Электронное учебное издание

Коковин Валерий Аркадьевич
Холопов Владимир Анатольевич

Электротехника и электроника (Цифровая электроника)

ЭЛЕКТРОННОЕ МЕТОДИЧЕСКОЕ ПОСОБИЕ

Филиал «Протвино»
государственного университета «Дубна
142281 г. Протвино Московской обл.,
Северный проезд, д. 9

Государственное бюджетное образовательное учреждение
высшего образования Московской области
«Университет «Дубна»
Филиал «Протвино»
Кафедра «Автоматизация технологических процессов и производств»

В.А. Коковин, В.А. Холопов

ЛАБОРАТОРНЫЕ РАБОТЫ ПО ДИСЦИПЛИНЕ «ЭЛЕКТРОТЕХНИКА И ЭЛЕКТРОНИКА» (ЦИФРОВАЯ ЭЛЕКТРОНИКА)

ЭЛЕКТРОННОЕ МЕТОДИЧЕСКОЕ ПОСОБИЕ

Рекомендовано
кафедрой автоматизации технологических процессов и производств
филиала «Протвино» государственного университета «Дубна»
в качестве методического пособия для студентов,
обучающихся по направлению
«Автоматизация технологических процессов и производств»

Протвино
2015

ББК 31.2я73
К59

Рецензент:
кандидат технических наук, доцент, доцент кафедры
«Транспортные средства и бортовые информационно-управляющие системы»
ФГБОУ ВО «Московский технологический университет»
А.В. Меркулов

Коковин, В.А.
К59 Лабораторные работы по дисциплине «Электротехника и электроника» (Цифровая электроника): электронное методическое пособие / В.А. Коковин, В.А. Холопов. — Протвино, 2015. — 79 с.: ил.

Методическое пособие содержит описания четырех лабораторных работ по дисциплине «Электротехника и электроника» по изучению принципов проектирования и функционирования логических элементов, устройств комбинационного и последовательного типов на базе современных программируемых логических интегральных схем с использованием интегрированного пакета MAX+plus II.

Методическое пособие предназначено для студентов, обучающихся по направлению «Автоматизация технологических процессов и производств».

ББК 31.2я73

Библиографический список

1. Коковин, В.А. Контроллер таймерной сети общей таймерной системы ускорительного комплекса ИФВЭ / В.А. Коковин, В.В. Комаров // Приборы и системы. Управление, контроль, диагностика. — 2005. — № 6. — С. 15—17.

2. Уэйкерли, Дж. Ф. Проектирование цифровых устройств : в 2 т. / Дж.Ф. Уэйкерли. — М. : Постмаркет, 2002.

3. ACEX 1K Programmable Logic Device Family Data Sheet. — Электрон. дан. — Режим доступа: http://www.altera.com.

4. ByteBlasterMV Parallel Port Download Cable Data Sheet. — Электрон. дан. — Режим доступа: http://www.altera.com.

Вариант 5																
Объект	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*T*1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
*T*2	0	0	0	1	1	1	1	0	0	0	0	1	1	1	0	0
*T*3	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
*T*4	0	0	0	0	1	1	0	0	0	1	1	1	1	1	1	0

Вариант 6																
Объект	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*T*1	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0
*T*2	0	0	0	1	1	1	1	0	0	0	0	1	1	1	0	0
*T*3	0	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0
*T*4	0	0	0	1	1	0	0	0	0	0	1	1	1	1	1	0

Вариант 7																
Объект	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*T*1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
*T*2	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0
*T*3	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0
*T*4	0	0	1	1	1	0	0	0	0	0	1	1	1	1	1	0

Вариант 8																
Объект	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*T*1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
*T*2	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0
*T*3	0	1	1	0	0	1	1	1	1	0	0	0	0	0	0	0
*T*4	0	0	0	0	1	1	0	0	0	0	1	1	1	1	1	0

Оглавление

Г. Варианты для выполнения лабораторной работы № 4

Вариант 1																
Объект	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*T*1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
*T*2	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0
*T*3	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
*T*4	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0

Вариант 2																
Объект	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*T*1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
*T*2	0	1	0	0	1	1	0	0	0	1	1	1	1	1	0	0
*T*3	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
*T*4	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0

Вариант 3																
Объект	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*T*1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
*T*2	0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	0
*T*3	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
*T*4	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0

Вариант 4																
Объект	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*T*1	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0
*T*2	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
*T*3	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
*T*4	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

В. Варианты для выполнения лабораторной работы № 3

Вариант	1	2	3	4	5	6	7	8
$K_{сч}$, ед.	53	24	19	109	135	106	48	38

Вариант	9	10	11	12	13	14	15	16
$K_{сч}$, ед.	74	126	83	124	119	99	65	145

Введение

Развитие технологии элементной базы цифровой схемотехники позволило создать программируемую логическую интегральную схему (ПЛИС) и, в большинстве случаев, отказаться от специализированных микросхем, минимизировать объем аппаратуры автоматизированных систем управления и повысить ее функциональность. Возрастание сложности интегральных схем, в частности ПЛИС, дает возможность иметь в аппаратуре все большее число компонентов, схемотехнически реализовывать всё более многообразные и сложные функции. Разработка цифровых узлов на таких емких (миллионы вентилей) и быстрых схемах, вполне доступных по цене, позволяет проектировать унифицированную аппаратуру различного назначения. Высокая надежность работы ПЛИС позволила, например, реализовать электронику общей таймерной системы управляющей аппаратуры ускорительного комплекса ИФВЭ на программируемых логических интегральных схемах [1].

Инструментальные средства разработки нового поколения — такие, как MAX+plus II или QUARTUS фирмы Altera, дают возможность проектировать функционально законченные блоки. Эти блоки представляют собой так называемые мегафункции, которые можно использовать без изменения в различных проектах, содержащих ПЛИС. Такой подход использован при разработке регистратора таймерных сообщений общей таймерной системы ускорительного комплекса ИФВЭ [1].

Созданные ПЛИС находят все большее применение при разработке цифровых устройств самого различного назначения. В ПЛИС заложены возможности, которые позволяют реализовать на ее основе интегральные схемы с любой функцией цифровой логики. В результате эволюции развития ПЛИС к настоящему времени разработаны и применяются микросхемы, которые можно разбить на два больших класса: ПЛИС с архитектурой CPLD — это сложные программируемые логические приборы (Complex Programmable Logic Device) и программируемые пользователем вентильные матрицы, которые имеют аббревиатуру FPGA (Field Programmable Gate Array). Использование ПЛИС обеспечивает максимальную гибкость при необходимости модификации аппаратуры. Применение ПЛИС позволяет сократить процесс проектирования и отладки цифровых устройств.

Проектирование цифровых устройств с применением ПЛИС имеет свои особенности. Для разработки конкретных схем использу-ются специально созданные системы автоматизированного проекти-рования, в которых для ввода могут использоваться языки описания схем или универсальные схемные редакторы. Обязательным этапом является моделирование, во время которого проверяется правильность разработанной схемы. Трансляция введенной схемы в битовую загру-зочную последовательность часто осуществляется автоматически без вмешательства пользователя. Для программирования микросхем при-меняются достаточно простые устройства, в том числе использующие четырехпроводной интерфейс JTAG, который позволяет не только достаточно просто производить загрузку ПЛИС, но и осуществлять тестирование микросхемы. Ведущие фирмы распространяют бесплат-ные системы проектирования, которые хотя и имеют ограничения по мощности и функциональному назначению по сравнению с платными, тем не менее позволяют разрабатывать проекты для многих практиче-ских приложений. При выполнении лабораторных работ по данному курсу используется бесплатный интегрированный пакет MAX+plus II фирмы Альтера.

Целью данного лабораторного практикума является освоение элементов математического аппарата цифровой схемотехники и алгеб- ры логики (булевой алгебры), выполнение анализа и синтеза базовых комбинационных и последовательностных устройств, изучение основ-ных приемов и методов создания проектов в интегрированном пакете MAX+plus II и наблюдение на экране осциллографа сигналов цифро-вой схемы в контрольных точках.

Данное пособие предназначено для студентов старших курсов специальностей «Автоматизация технологических процессов и про-изводств в машиностроении» и «Программное обеспечение вычисли-тельной техники и автоматизированных систем». Все лабораторные работы выполняются на учебном стенде, в составе которого исполь-зуются современные ПЛИС классов CPLD и FPGA. При подготовке и выполнении лабораторных работ данного курса необходимо пользо-ваться следующей литературой: данным лабораторным практикумом, конспектом лекций по курсу «Электротехника и электроника (цифро-вая электроника)», описанием учебного стенда и паспортом и руковод-ством по эксплуатации учебного стенда «УС-ПЛИС-2-96» (далее —

Вариант 7										
x_2	x_1	x_0	*A*	*B*	*C*	*D*	*E*	*F*	*G*	Символ
0	0	0	0	0	0	0	1	0	0	9
0	0	1	0	0	0	1	0	0	0	*A*
0	1	0	1	1	0	0	0	0	0	*B*
0	1	1	1	0	0	1	1	1	1	1
1	0	0	0	0	1	0	0	1	0	2
1	0	1	0	0	0	0	1	1	0	3
1	1	0	0	0	0	1	1	1	1	7
1	1	1	0	0	0	0	0	0	0	8

Вариант 8										
x_2	x_1	x_0	*A*	*B*	*C*	*D*	*E*	*F*	*G*	Символ
0	0	0	0	0	0	0	1	0	0	9
0	0	1	0	0	0	1	0	0	0	*A*
0	1	0	1	0	0	1	1	0	0	4
0	1	1	0	1	0	0	1	0	0	5
1	0	0	0	1	0	0	0	0	0	6
1	0	1	0	0	0	1	1	1	1	7
1	1	0	1	0	0	1	1	1	1	1
1	1	1	0	0	1	0	0	1	0	2

Вариант 4										
x_2	x_1	x_0	*A*	*B*	*C*	*D*	*E*	*F*	*G*	Символ
0	0	0	0	0	0	0	1	0	0	9
0	0	1	0	0	0	1	0	0	0	*A*
0	1	0	1	1	0	0	0	0	0	*B*
0	1	1	0	1	1	0	0	0	1	*C*
1	0	0	1	0	0	0	0	1	0	*D*
1	0	1	0	1	1	0	0	0	0	*E*
1	1	0	0	1	1	1	0	0	0	*F*
1	1	1	0	0	0	0	0	0	1	*O*

Вариант 5										
x_2	x_1	x_0	*A*	*B*	*C*	*D*	*E*	*F*	*G*	Символ
0	0	0	1	0	0	1	1	1	1	1
0	0	1	0	0	1	0	0	1	0	2
0	1	0	0	0	0	0	1	1	0	3
0	1	1	0	1	1	0	0	0	1	*C*
1	0	0	1	0	0	0	0	1	0	*D*
1	0	1	0	1	1	0	0	0	0	*E*
1	1	0	0	1	1	1	0	0	0	*F*
1	1	1	0	0	0	0	0	0	1	*O*

Вариант 6										
x_2	x_1	x_0	*A*	*B*	*C*	*D*	*E*	*F*	*G*	Символ
0	0	0	0	0	0	0	1	0	0	9
0	0	1	0	0	0	1	0	0	0	*A*
0	1	0	1	1	0	0	0	0	0	*B*
0	1	1	0	1	1	0	0	0	1	*C*
1	0	0	1	0	0	0	0	1	0	*D*
1	0	1	0	1	1	0	0	0	0	6
1	1	0	0	1	1	1	0	0	0	7
1	1	1	0	0	0	0	0	0	1	8

инструкция). Кроме того, для более углубленного изучения цифровой схемотехники, необходимо использовать литературу, приведенную в библиографическом списке к практикуму.

Каждая лабораторная работа состоит из подготовки проекта цифровой схемы в пакете MAX+plus II, загрузки готового проекта в ПЛИС стенда и наблюдения на экране осциллографа сигналов цифровой схе-мы в контрольных точках. Разработка проекта состоит из нескольких этапов: организации проекта, схемного ввода, компиляции (синтез, функциональное симулирование, размещение в кристалле), временного моделирования и загрузки (конфигурирования) в ПЛИС стенда. Рас-смотрим более подробно основные этапы разработки проекта.

Этапы создания проекта в интегрированном пакете MAX+plus II

Использование современных CAD пакетов — таких, как MAX+plus II, значительно снижает трудоемкость разработок проектов цифровых устройств и повышает качество этих разработок. Одним из способов введения исходных данных проекта является схемный ввод. На рис. 1 определены этапы разработки проекта.

Выполнение проекта в системе MAX+plus II включает следующие этапы:

• **Схемный ввод проекта** — ввод элементов цифрового устрой-ства в виде схемных блоков (проект может вводится в текстовом виде на языках описания аппаратуры VHDL или Verilog).

• **Синтез** — введенные схемные блоки синтезируются в виде логических элементов LE (Logical Elements), из которых состоит FPGA и CPLD.

• **Функциональное симулирование** — проверка синтезированной схемы на функциональную корректность. Если ввод схемных блоков выполнен некорректно, то необходимо вернуться на этап схемного ввода.

• **Размещение в кристалле (fitting)** — размещение синтезирован-ных LE в выбранный кристалл.

• **Временной анализ** — анализ, при котором определяется оптимальный путь распространения сигналов в кристалле.

• **Временная| симуляция (моделирование)** — проверка функционального и временного| соответствия размещенной в кристалле схемы заданным условиям.

• **Программирование и конфигурирование** — программирование конфигурационных переключателей, которые конфигурируют LE в виде заданной функции и устанавливают соединения внутри кристалла.

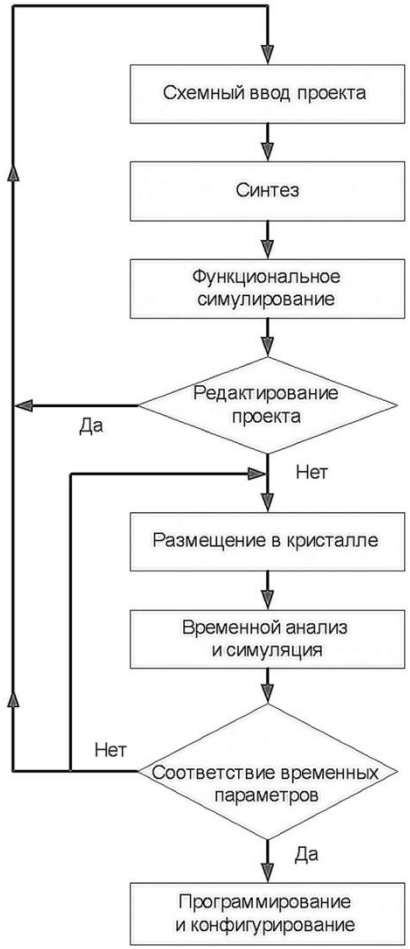


Рис. 1. Этапы выполнения проекта цифрового устройства

Б. Варианты для выполнения лабораторной работы № 2

Вариант 1										
x_2	x_1	x_0	*A*	*B*	*C*	*D*	*E*	*F*	*G*	Символ
0	0	0	0	0	0	0	1	1	0	3
0	0	1	1	0	0	1	1	0	0	4
0	1	0	0	1	0	0	1	0	0	5
0	1	1	0	1	0	0	0	0	0	6
1	0	0	0	0	0	1	1	1	1	7
1	0	1	0	0	0	0	0	0	0	8
1	1	0	0	0	0	0	1	0	0	9
1	1	1	0	0	0	1	0	0	0	*A*

Вариант 2										
x_2	x_1	x_0	*A*	*B*	*C*	*D*	*E*	*F*	*G*	Символ
0	0	0	0	0	0	0	1	1	0	*C*
0	0	1	1	0	0	1	1	0	0	*D*
0	1	0	0	1	0	0	1	0	0	*E*
0	1	1	0	1	0	0	0	0	0	*F*
1	0	0	0	0	0	1	1	1	1	*O*
1	0	1	0	0	0	0	0	0	0	1
1	1	0	0	0	0	0	1	0	0	2

Вариант 3										
x_2	x_1	x_0	*A*	*B*	*C*	*D*	*E*	*F*	*G*	Символ
0	0	0	0	1	0	0	0	0	0	6
0	0	1	0	0	0	1	1	1	0	7
0	1	0	0	0	0	0	0	0	0	8
0	1	1	0	0	0	0	1	0	0	9
1	0	0	0	0	0	1	0	0	0	*A*
1	0	1	1	1	0	0	0	0	0	*B*
1	1	0	0	1	1	0	0	0	1	*C*
1	1	1	1	0	0	0	0	1	0	*D*

Вариант 8																
v_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
y	0	0	0	1	0	0	0	1	1	1	1	0	0	1	1	1

Организация проекта в MAX+plus II

Каждая схема, введенная в CAD MAX+plus II, называется проектом и сохраняется в отдельном файле. MAX+plus II в каждый момент времени работает только с одним проектом. Перед началом ввода новой схемы необходимо создать директорию, в которой будут сохраняться файлы проек-та. Наша директория будет называться **Proba** (или любое другое имя).

Для начала работы необходимо запустить программу MAX+plus II на исполнение. На экране монитора должно открыться окно, представ-ленное на рис. 2.

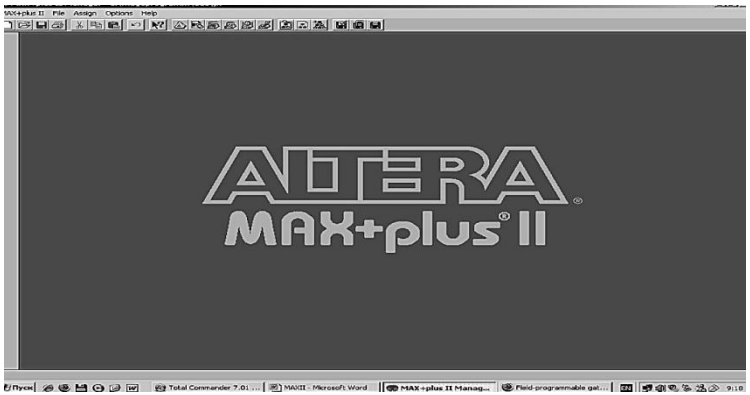


Рис. 2. Основное интерфейсное окно пакета MAX+plus II

Основные команды, обеспечивающие работу пакета MAX+plus II, расположены в верхнем меню. Например, если нажать левой кнопкой мыши на меню **File**, то откроется меню, изображенное на рис. 3. С помощью этого меню можно, например, открыть новый файл **(New)** или выйти из пакета MAX+plus II, нажав левой кнопкой на опцию **Exit MAX+plus II**.

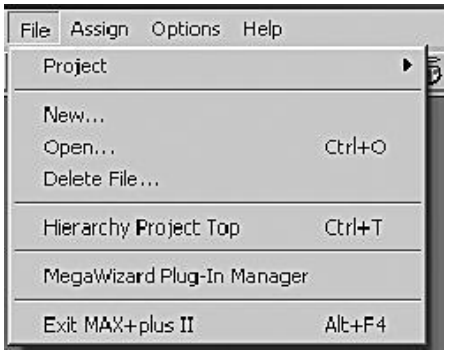


Рис. 3. Окно меню File

Для некоторых команд необходимо нажатие левой кнопкой несколько раз, поскольку одно меню может быть вложено в другое. Например, в основном меню мы нажимаем левой кнопкой **File** (Меню 1), переме-щаемся вниз до **Open**, выбираем требуемый файл и нажимаем снова левую кнопку мыши (Меню 2).

Создание и компиляция нового проекта

Для начала работы над новым проектом необходимо определить редактор, в котором будет выполняться ввод элементов цифрового устройства.

Шаг 1. Выбор редактора. Из меню, представленном на рис. 3, выберите: **File > New**.

Откроется окно, изображенное на рис. 4. В верхней строчке обозначен ввод проекта в графическом редакторе **(Graphic Editor file)**. В следующей строчке — символьный редактор **(Symbol Editor file)**, далее текстовый **(Text Editor file)** и последний — редактор временных форм (осциллограмм) **(Waveform Editor file)**. Выбираем **Graphic Editor file** с расширением **.gdf (graphic design file)** и нажимаем **OK.** В графическом редакторе откроется «чистый лист» нового файла, имеющего название Untitled1. Сохраним его под именем **test1.gdf** в созданной папке **Proba.** Далее можно осуществлять непосредственный ввод элементов схемы.

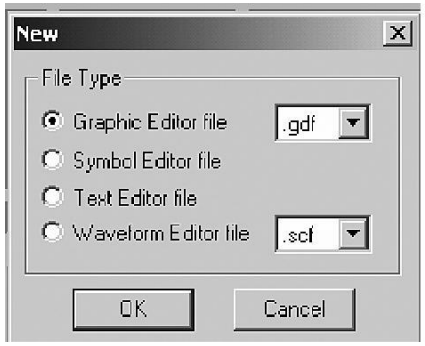


Рис. 4. Выбор редактора ввода

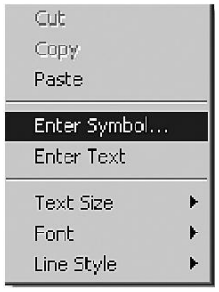


Рис. 5. Выбор символьных элементов из библиотек пакета MAX+plus II

Вариант 4																
v_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
y	1	0	0	1	0	0	0	1	1	1	0	0	0	1	0	1

Вариант 5																
v_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
y	1	0	0	1	1	1	0	1	0	1	0	1	0	1	0	1

Вариант 6																
v_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
y	1	1	1	1	0	0	0	0	1	1	0	0	0	1	0	1

Вариант 7																
v_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
y	1	0	0	1	1	1	0	1	1	1	1	0	0	1	1	1

Приложения

А. Варианты для выполнения лабораторной работы № 1

Вариант 1																
v_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
y	0	0	1	0	1	0	1	0	1	1	0	0	0	1	0	1

Вариант 2																
v_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
y	0	1	0	1	0	0	0	0	1	1	0	1	1	1	0	0

Вариант 3																
v_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
y	0	0	0	1	0	1	0	1	0	1	0	0	0	1	0	1

Шаг 2. Ввод библиотечных элементов. Пакет MAX+plus II содержит разнообразные библиотечные элементы, расположенные в специальных библиотечных директориях (первый символ **c**:, если пакет установлен на диске c):

- c:\maxplus2\max2lib\prim — библиотека примитивов (элементы AND, OR, NOT, GND, VCC, входы, выходы и т.□д.);
- c:\maxplus2\max2lib\mf — библиотека макрофункций, включающая большое разнообразие макроэлементов — таких, как счетчики, сумматоры, регистры, мультиплексоры и т.□д.;
- c:\maxplus2\max2lib\mega_lpm — библиотека мегафункций, включающая параметризованные функциональные элементы. Эта библиотека дает пользователю наиболее оптимизированные и мощные ресурсы при создании проекта. Более подробно о мегафункциях будет рассказано в последующих главах.

Кроме того, пользователь может создать свою собственную библиотеку (user library), которая будет содержать символы и элементы, разработанные пользователем.

Для ввода символа из библиотеки необходимо на рабочем поле проекта нажать правую кнопку мыши, после чего в открывшемся меню выбрать **Enter Symbol** (рис. 5). Второй способ ввода библиотечных элементов — выбрать в основном меню **Symbol > Enter Symbol**.

Откроется окно **Enter Symbol**, изображенное на рис. 6. Для ввода символа необходимо два раза кликнуть на выбранной библиотеке (в нашем случае это c:\maxplus2\max2lib\prim) и в поле Symbol Files откроется перечень символов. Выберем символ элемента **and2**, при этом в поле Symbol Name появится название элемента. Если выбран нужный элемент, то нажимаем **OK**.

После ввода символа **and2** на рабочем поле появится выбранный элемент. Для полноценной схемы необходимо присутствие элементов ввода (символ **input**) и элементов вывода сигналов (символ **output**). Выберем эти символы из библиотеки c:\maxplus2\max2lib\prim и разместим на рабочем поле. После ввода символов ввода и вывода необходимо нарисовать связи выбранных элементов. Для этого нужно навести курсор на какой-нибудь вывод элемента и при появлении маркера «крест» нажать левую кнопку мыши и рисовать линию связи.

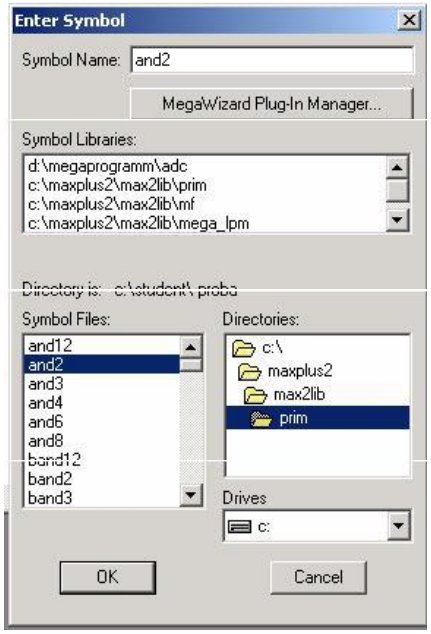


Рис. 6. Ввод символа **and2**

На рис. 7 изображен элемент **and2** (логическое умножение двух сигналов). Входы элемента **and2** соединены с элементами **input**, а выход — с элементом **output**. Далее необходимо присвоить входным и выходным элементам имена (например **x1, x2** и **y**), для чего два раза кликнуть на PIN_NAME и ввести новое название сигнала.

Если схема проекта небольшая, то вполне достаточно рабочего поля пакета **MAX+plus II**. Но, как правило, реальные проекты требу-ют гораздо больше места для своего ввода, чем отображаемое рабочее поле. В этом случае удобно часть схемы объединить в виде графиче-ского символа. Например, рассмотрим создание графического символа на базе схемы, представленной на рис. 7. Для этого введем схему, изо-браженную на этом рисунке, зададим ей имя **test2** (или любое другое) и выполним компиляцию. После этого выберем в меню: **File > Create Default Symbol** и, нажимая левую кнопку мыши, создадим по умол-чанию символ **test2**. Теперь можно этот символ ввести в какой-нибудь другой проект, открытый в той же самой директории.

Требования к отчету

Отчет по лабораторной работе должен выполняться в отдельной тетради и содержать:

- Название лабораторной работы, ее цель, задачи.
- Вариант задания.
- Схему цифрового автомата.
- Временные| диаграммы работы цифрового автомата.

Вопросы и задания для самопроверки

- К какому типу цифровых схем относятся конечные автоматы?
- К какому типу цифровых схем относятся микропрограммные ав-томаты?
- Написать характеристическое уравнение *RS*-триггера.
- Написать характеристическое уравнение *T*-триггера с входом разрешения.
- Разработать цифровой автомат на 10 состояний, таблица состоя-ний которого задается статической памятью.
- Разработать цифровой автомат на 16 состояний, таблица состоя-ний которого задается постоянной запоминающей памятью.

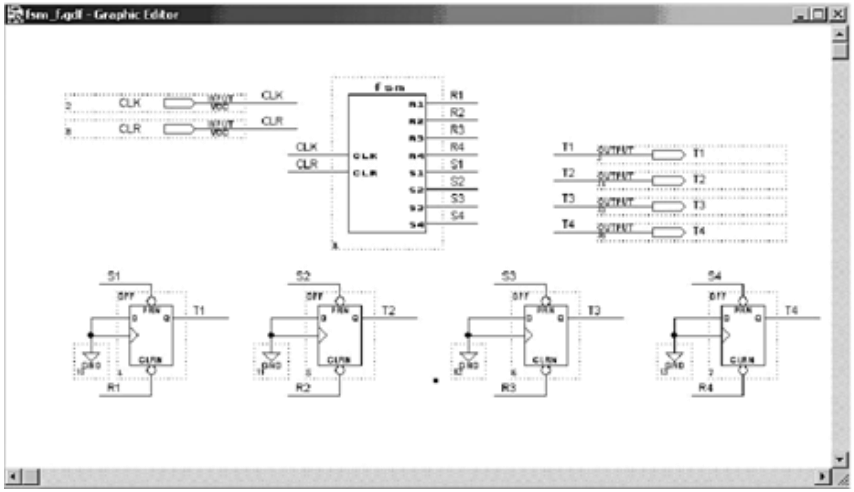


Рис. 48. Полная схема цифрового автомата и объектов управления

2. Выполним временное| моделирование разработанной схемы.

Анализируя работу цифрового автомата по временной| диаграмме (рис. 49), мы видим, что состояния объектов управления (триггеров) полностью соответствуют заданным.

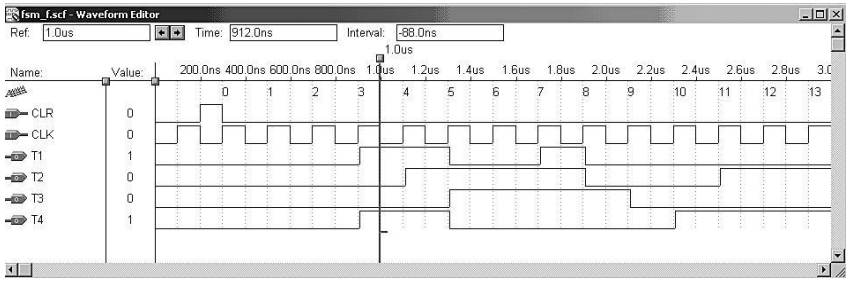


Рис. 49. Временная| диаграмма работы цифрового автомата

3. Загрузим готовый проект в ПЛИС стенда и проверим правильность работы цифрового автомата.

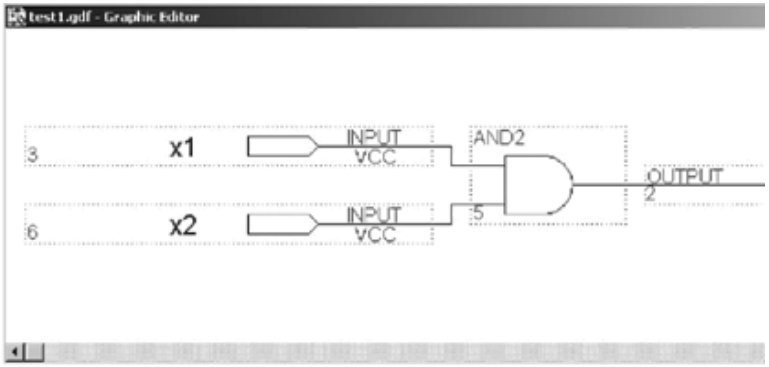


Рис. 7. Элемент логического умножения входных сигналов **x1** и **x2**

Создадим новый проект под именем **test3**. Откроем рабочее поле проекта и введем созданный символ **test2** (рис. 8). Для ввода нового символа выберем в окне **Symbol Files** имя символа, при этом имя выбранного символа появится в окне **Symbol Name**, после чего нажмем кнопку **OK.**

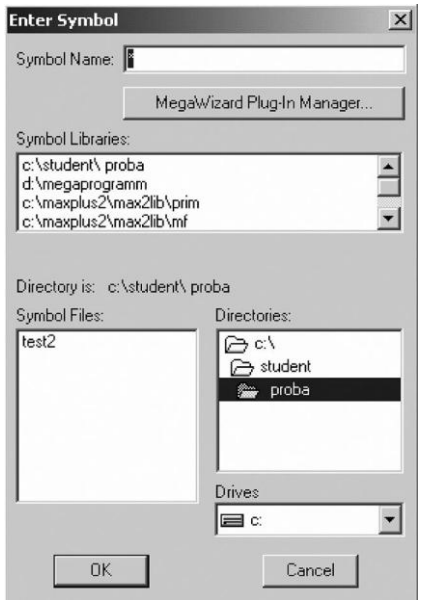


Рис. 8. Ввод созданного символа **test2**

Шаг 3. Выбор типа ПЛИС. Фирма Альтера выпускает широкую номенклатуру устройств с различной архитектурой. Для выбора конкретного типа ПЛИС необходимо выполнить следующие действия.
В верхнем основном меню (где **File**) выбрать **Assign > Device**. Откроется меню, изображенное на рис. 9. В выпадающем меню **Device Family** выбрать семейство устройств ACEX1. В меню **Devices** выбрать ту микросхему, которая указана в инструкции на учебный стенд. Например, на рис. 8 выбрана микросхема EP1K50QC208-3 семейства ACEX 1K [2].

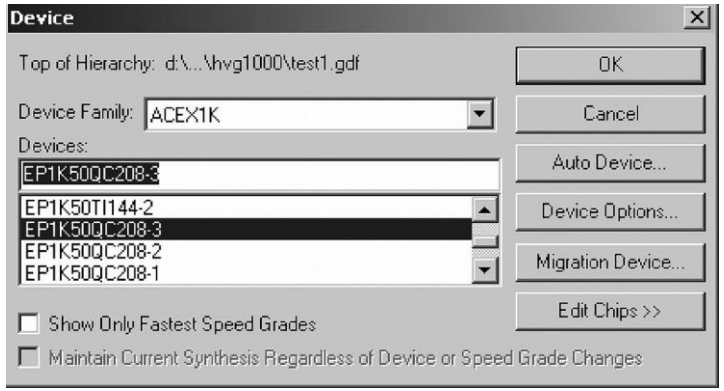


Рис. 9. Выбор типа ПЛИС

Шаг 4. Компиляция проекта. Наш проект содержит всего один файл и готов для компиляции. Компиляция проекта выполняется в окне **Compiler** и содержит несколько этапов:

• синтез введенных схемных элементов в виде логических элементов LE (logical elements);

• размещение проекта в выбранном кристалле (выбор микросхем будет описан позднее);

• создание списка связей для симуляции;

• создание ассемблерного файла для программирования выбранного кристалла.

Окно **Compiler** можно открыть двумя способами: Первый способ:

• Выбрать из основного меню **MAX+plus II > Compiler > Start**. На вопрос о необходимости сохранения проекта — нажать **Да**.

Составим логические выражения для управляющих сигналов:

$R1 = Y5 \,\&\, Y12$, $S1 = Y3 \,\&\, Y7$,
$R2 = Y2 \,\&\, Y14$, $S2 = Y4 \,\&\, Y11$,
$R3 = Y9$, $S3 = Y5$,
$R4 = Y5 \,\&\, Y15$, $S4 = Y3 \,\&\, Y10$.

Разработаем в пакете MAX+plus II схему цифрового автомата. На счетный вход счетчика подадим тактовый сигнал *CLK*. На вход сброса подадим сигнал *CLR*. Нарисуем цифровой автомат и сохраним схему как отдельный символ под именем fsm.gdf. На рис. 47 представлена схема цифрового автомата с синтезированной выходной логикой.

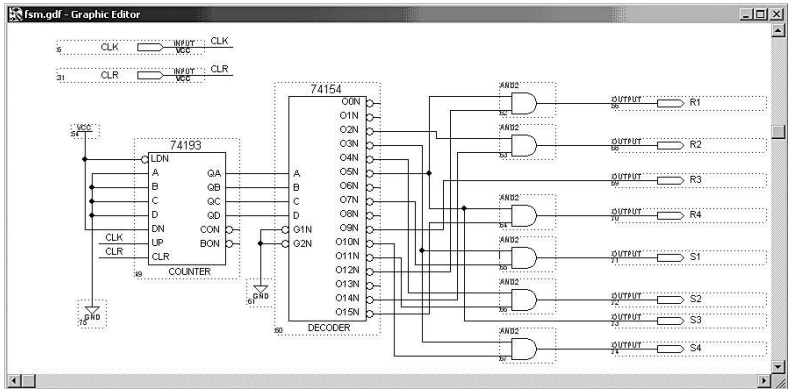


Рис. 47. Цифровой автомат с синтезированной выходной логикой

Синтезируя выходную комбинационную логику, мы однознач-но определяем работу цифрового автомата по заданному алгоритму (см. таблицу состояний). Если необходимо оперативно менять управля-ющие воздействия (*R* и *S*), то можно использовать вместо дешифратора и комбинационной логики перезаписываемую память или ПЗУ. В этом случае можно говорить о работе микропрограммного автомата. Управляющие сигналы *R* и *S* управляют выходными триггерами *T*1, *T*2, *T*3 и *T*4 в каждый такт управляющего цикла. Состояние триггеров меняется по положительному фронту тактового сигнала *CLK*. На рис. 48 представлена полная схема, реализующая заданный алгоритм.

В табл. 7 $Q1$, $Q2$, …, $Q3$ — выходы счетчика, $T1$, $T2$, …, $T4$ — триггеры, $S1$, $S2$, …, $S4$ — управляющие сигналы, подаваемые на S-входы триггеров, $R1$, $R2$, …, $R4$ — управляющие сигналы, подавае-мые на R-входы триггеров. Синтезируем функции R и S согласно задан-ному алгоритму. В нашем примере эти функции достаточно простые.
В более сложных задачах комбинационная логика минимизируется с помощью карт Карно.

Таблица 7. Таблица соответствия кодовых комбинаций выходов счетчика и управляющих сигналов

$Q3$	$Q2$	$Q1$	$Q0$	Номер состояния	$T4$	$T3$	$T2$	$T1$
0	0	0	0	0				
0	0	0	1	1				
0	0	1	0	2				
0	0	1	1	3	$S4$			$S1$
0	1	0	0	4			$S2$	
0	1	0	1	5	$R4$	$S3$		$R1$
0	1	1	0	6				
0	1	1	1	7			$R2$	$S1$
1	0	0	0	8				
1	0	0	1	9		$R3$		
1	0	1	0	10	$S4$			
1	0	1	1	11			$S2$	
1	1	0	0	12				$R1$
1	1	0	1	13				
1	1	1	0	14			$R2$	
1	1	1	1	15	$R4$			

Второй способ:
• Выбрать из основного меню **File > Project > Save & Compile**.
Второй способ более удобный, поскольку совмещает операцию сохранения файла и запуска процесса компиляции. На рис. 10 изображено окно с выполненными этапами, перечисленными выше.

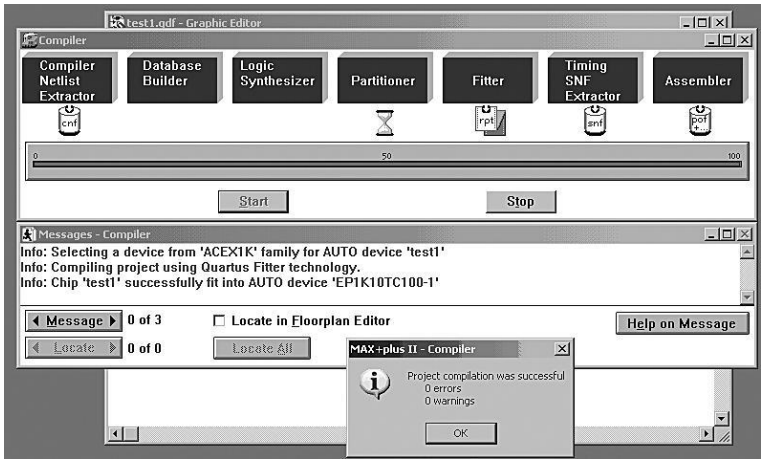


Рис. 10. Выполнение компиляции проекта **test1**

Если при вводе элементов схемы не было ошибок, то после компиляции проекта появится надпись «компиляция проекта выполне-на успешно, 0 ошибок, 0 предупреждений» (Project compilation was successful, 0 errors, 0 warnings). Нажимаем кнопку **OK** и закрываем окно **Compiler**. После компиляции будут созданы различные файлы для выполнения моделирования, конфигурирования и файл отчета с расширением **.rpt**.

Временное| моделирование проекта

После безошибочной компиляции проекта будет создан специальный файл **test1.cnf (compiler netlist file)**, описывающий все связи в схеме проекта. Это позволяет выполнить временное| моделирование проекта. Основная цель моделирования — проверить правильность функциониро-

вания созданной схемы и оценить временные| задержки распространения сигналов. Симуляция проекта выполняется в окне **Waveform Editor**.

Чтобы открыть окно **Waveform Editor**, выполним следующие действия: выберем из основного меню **MAX+plus II > Waveform Editor**. В открывшемся окне на поле редактора правой кнопкой мыши открываем контекстное меню и выбираем **Enter Nodes from SNF > List**. В левом окне **Available Nodes & Groups** интерфейса **Enter Nodes from SNF** выберем необходимые сигналы (в нашем случае **x1**, **x2** и **y**) и с по-мощью стрелки → перетаскиваем их в правое окно. Нажимаем кнопку **OK** и возвращаемся в окно **Waveform Editor**.

В верхней части **Waveform Editor** содержится три окна:

• Reference (Ref) — начало моделирования.

• Time — временной| интервал между началом отсчета и специальным маркером (маркер по умолчанию находится в крайней левой точке временной| оси).

• Interval — временной| интервал между специальным маркером и курсором мыши.

Для того чтобы задать входные сигналы, необходимо нажать на кнопку и нарисовать поведение соответствующего входного сигнала. Например так, как изображено на рис. 11. После задания входных сигналов необходимо сохранить параметры моделирования в специальном файле, нажимая (находясь в окне **Waveform Editor**) **File > Project > Save & Simulate**. Далее с помощью кнопки Start запускаем симуляцию. Результат временной| симуляции мы видим на рис. 11. Переменная **y** принимает значения согласно логической функции **y**□=□**x1** & **x2**.

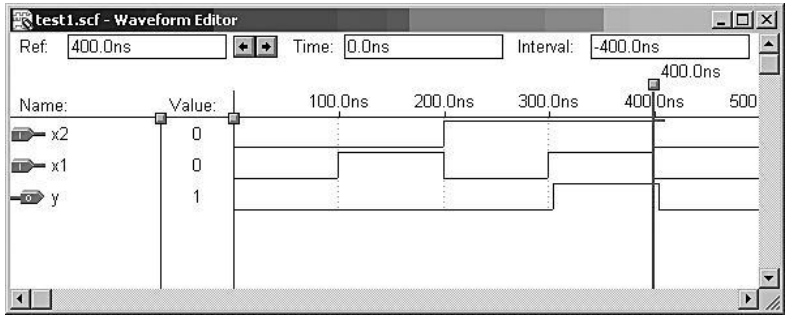


Рис. 11. Задание поведения входных сигналов **x1** и **x2**

Выполнить следующие действия:

• в графическом редакторе Graphic Editor пакета MAX+plus II создать проект, реализующий цифровой автомат для управления четырьмя объектами (поведение объектов индицировать на светодиодах стенда);

• выполнить компиляцию проекта;

• в редакторе Waveform Editor задать входные и выходные сигналы цифрового автомата и выполнить временное| моделирование схемы (сравнить временную| диаграмму работы модели и заданную таблицу состояний объектов);

• загрузить конфигурационный файл в ПЛИС;

• проверить правильность работы автомата, наблюдая за работой объектов с помощью светодиодов стенда.

3 Пример выполнения лабораторной работы № 4

1. Выполним анализ таблицы состояний объектов (табл. 6) и разработаем цифровой автомат.

Поведение объектов во времени будем описывать с помощью ука-занной таблицы, в которой приведен один из вариантов. Обозначим объекты через *T* (*T*1, …, *T*4).

Используя данные табл. 6, составим таблицу управляющих сигналов. Поскольку объектами являются триггеры, то удобно управлять ими через *R*-входы (установка в «0») и *S*-входы (установка в «1»), подавая на них управляющие сигналы. Выходную логику автомата построим на дешифраторе 74154, у которого выходы *Y*0, *Y*1, …, *Y*15 име-ют активный сигнал — логический «0». Память состояния цифрового автомата выполним на счетчике 74193.

Таблица 6. Таблица состояний объектов

Объект	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T 1	0	0	0	1	1	0	0	1	1	1	1	1	0	0	0	0
T 2	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0
T 3	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
T 4	0	0	0	1	1	0	0	0	0	0	1	1	1	1	1	0

• За один период тактовой частоты должны успеть сработать регистр и ПЗУ.

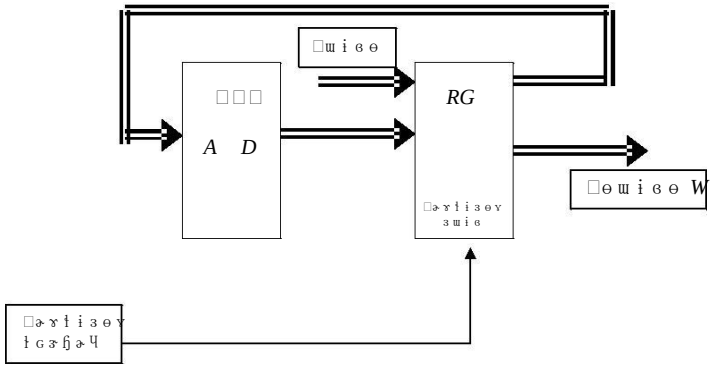


Рис. 46. Структура микропрограммного автомата

Отметим некоторые простые последовательные действия, из которых могут складываться алгоритмы работы микропрограммного автомата:

1. Последовательный перебор адресов ПЗУ для выдачи последовательности выходных сигналов.

2. Периодическое повторение последовательности адресов ПЗУ для повторения последовательности выходных сигналов.

3. Остановка в каком-то адресе ПЗУ для ожидания изменения входного сигнала.

На микропрограммных автоматах, разработанных по структурной схеме (рис. 46), можно построить сложные устройства. Например, можно построить цифровой генератор функции или автомат, перекоди-рующий код Манчестер-II в двоичный.

2 Задание для лабораторной работы № 4

Изучить структуру и характеристические уравнения конечных автоматов. Разработать проект цифрового автомата, управляющего на-бором объектов и работающего по цикловому алгоритму. В качестве объектов управления использовать четыре *D*-триггера. Поведение объ-ектов задается таблицей состояний соответствующего варианта. Нари-совать циклограмму работы автомата.

Программирование и конфигурирование ПЛИС стенда

Учебный стенд содержит две ПЛИС, имеющих разные архитектуры — CPLD и FPGA. Тип микросхемы, используемой в учебном стен-де, и ее характеристики можно узнать из инструкции на учебный стенд. Лабораторные проекты будут загружаться из персонального компью-тера в ПЛИС, выполненную на базе архитектуры FPGA. Архитектура FPGA предполагает наличия в ПЛИС внутреннего ОЗУ, содержимое которого определяет логические функции и внутренние соединения. Для работы созданного проекта цифрового устройства необходимо за-писать в это внутреннее ОЗУ (SRAM) конфигурационный файл, соз-данный при компиляции. При успешной компиляции проекта созда-ются два конфигурационных файла **test1.pof (programmer object file)** и **test1.sof (sram object file).** Файл с расширением **.sof** предназначен для непосредственной загрузки в ПЛИС (конфигурирование), а файл с расширением **.pof** — для программирования конфигурационной памяти (внешнего ПЗУ). В последнем случае загрузка ПЛИС осуществляется при подаче питания, при этом информация из конфигурационной памяти копируется в ПЛИС. Архитектура CPLD предполагает наличие конфигурационной памяти прямо в ПЛИС.

Существует несколько способов программирования и конфигурирования ПЛИС, определенных техническим руководством ПЛИС [2]. Рассмотрим основной режим загрузки проекта в ПЛИС (FPGA) стенда через параллельный порт компьютера с помощью адаптера ByteBlasterMV. Такая загрузка предусмотрена как штатная процедура и обеспечивается средствами пакета MAX+plus II [3].

Для загрузки проекта лабораторной работы необходимо соединить соответствующий разъем учебного стенда (см. описание стенда [2]) и параллельный порт LPT используемого компьютера кабелем адапте-ра ByteBlasterMV. В рабочем окне пакета MAX+plus II (при открытом и подготовленном для загрузки проекте) выбрать в основном меню за-кладку **MAX+plus II > Programmer**. Откроется окно, представленное на рис. 12. В этом окне содержится информация о загружаемом файле **(test1.sof),** выбранном типе ПЛИС **(Device:EP1K50QC208-3**) и контроль-ной сумме. Для загрузки проекта необходимо нажать кнопку **Program.**

Окно **Programmer** (рис. 12) содержит дополнительные опции: **Examine** и **Verify**. Опция **Examine** используется для считывания данных из ПЛИС с архитектурой CPLD. При считывании данных из ПЛИС, ее содержимое помещается во временный файл, а затем сохра-няется в рабочей директории. Сохраненный файл можно использовать для программирования других CPLD с такими же характеристиками, при условии, что в проекте не используется секретный бит (Security Bit). Опция **Verify** используется для проверки результатов программирования конфигурационной памяти, выполненной по технологии Flash, или ПЛИС с технологией EEPROM.

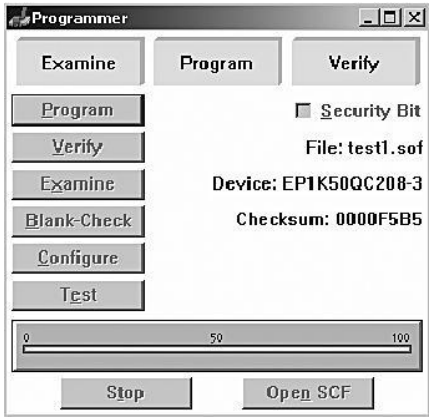


Рис. 12. Окно загрузки проекта в ПЛИС

Анализ тактируемых синхронных конечных автоматов выполняется в три основных этапа:

• Определяются функции переходов *F* и выхода *G*.

• Функции *F* и *G* используются для построения таблицы «состояние/выход», которой полностью задаются последующие состояния и выходы схемы при всех возможных комбинациях текущего состояния и текущих значений входных сигналов.

• Вычерчивается диаграмма состояний, которая представляет информацию, полученную на предыдущем шаге, в графической форме.

Последний этап необязательный.

1.2 Микропрограммные автоматы на постоянных запоминающих устройствах

На основе микропрограммных автоматов можно проектировать устройства, которые работают по довольно сложным алгоритмам и выполняют различные функции, определяемые входными сигналами. В микропрограммном автомате выходные сигналы зависят от входных сигналов и текущего состояния автомата. Но логика работы памяти состояния автомата задается микропрограммой, зашитой в постоянных запоминающих устройствах (ПЗУ). Структура микропрограммных автоматов, как правило, содержит три основных узла:

• ПЗУ;

• регистр, срабатывающий по фронту (например, на *D*-триггерах);

• устройство, вырабатывающее тактовый сигнал.

На рис. 46 приведена структура микропрограммного автомата.

ПЗУ имеет *M* адресных разрядов и *N* разрядов данных. Если входы автомата имеют *L* разрядов, то регистр должен иметь (*N*□+□*L*) раз-рядов. Данные записываются в регистр *RG* по положительному фронту тактового сигнала. Часть выходных разрядов регистра используется для управления адресом ПЗУ, другая часть служит для формирования выходных сигналов.

Алгоритм и условия правильной работы схемы следующие:

• В каждом такте ПЗУ выдает код данных, определяя не только вы-ходные сигналы, но и адрес ПЗУ в следующем такте.

• На формирование адреса в следующем такте влияют также вход-ные сигналы.

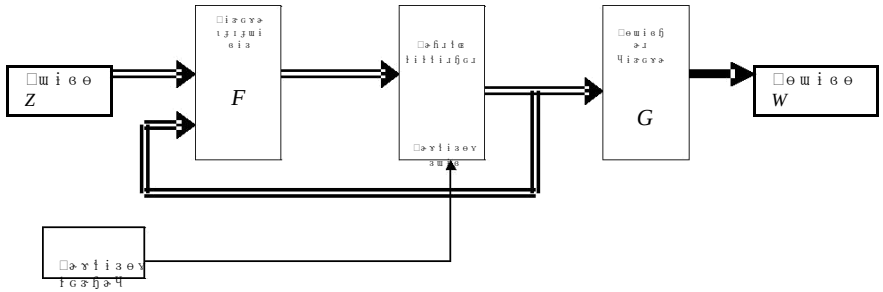


Рис. 45. Структура конечного автомата (автомат Мура)

Функциональное поведение триггера можно описать формально с помощью характеристического уравнения, посредством которого следующее состояние триггера задается как функция его текущего состояния и значений сигналов на его входах. В табл. 5 перечислены характеристические уравнения для различных триггеров. Принято считать, что символ «*» (звездочка) в записи Q^* означает «следующее значение Q».

Таблица 5. Характеристические уравнения триггерных устройств

Тип схемы	Характеристическое уравнение
Переключающийся по фронту *D*-триггер	$Q^* = D$
D-триггер с входом разрешения	$Q^* = (EN \,\&\, D) + (\overline{EN} \,\&\, Q)$
Двухтактный *RS*-триггер	$Q^* = S + (\overline{R} \,\&\, Q)$
Двухтактный *JK*-триггер	$Q^* = (J \,\&\, \overline{Q}) + (\overline{K} \,\&\, Q)$
Переключающийся по фронту *JK*-триггер	$Q^* = (J \,\&\, \overline{Q}) + (\overline{K} \,\&\, Q)$
T-триггер	$Q^* = \overline{Q}$
T-триггер с входом разрешения	$Q^* = (EN \,\&\, \overline{Q}) + (\overline{EN} \,\&\, Q)$

Лабораторная работа №□1

Комбинационные логические схемы

Цель работы: изучение методов анализа и синтеза комбинацион-ных логических схем (КЛС).

1 Краткие сведения из теории

Логические схемы подразделяются на два класса: комбинационные и последовательностные. Комбинационной является такая логическая схема, сигналы на выходах которой зависят только от текущих значений входных сигналов. Комбинационная схема может состоять из произвольного числа логических вентилей и инверторов, но в ней нет обратных связей. Под обратной связью понимают наличие в схеме пути, по которому сигнал с выхода вентиля может пройти на вход того же вентиля. В общем случае такие петли обратной связи делают схему последовательностной [4].

Формальные методы анализа цифровых схем впервые представлены в 1854 году английским математиком Джорджем Булем, который ввел двузначную алгебраическую систему, называемую теперь булевой алгеброй. Позднее, в 1938 году Клод Э. Шеннон показал, как приспособить булеву алгебру для описания поведения и анализа схем.

В алгебре логики рассматриваются следующие компоненты:

– **переменные**, могут принимать только два значения **0** и **1**, их будем обозначать латинскими буквами *x*, *y*, *z*, …, а также x_0, x_1, …, x_n, y_0, y_1, …, y_n и т.□д.;

– **отношение эквивалентности** (равенства «=»), удовлетворяет следующим свойствам:

– **рефлексивность** — *x*□=□*x*;

– **симметричность** — если *x*□=□*y*, то *y*□=□*x*;

– **транзитивность** — если *x*□=□*y* и *y*□=□*z*, то *x*□=□*z*, отсюда следует принцип: если *x*□=□*y*, то в любой формуле, содержащей *x*, вместо *x* можно подставить *y* и в результате будет получена эквивалентная формула;

– **три операции:**

– **дизъюнкция** — операция ИЛИ (логическое сложение), обозначают знаком ∃ или□+;

– **конъюнкция** — операция И (логическое умножение), обозначается знаком ӟ , или &, или *, или опускается;

– **отрицание** — инверсия (операция НЕ), обозначается чертой над переменной, или над элементами 0 и 1, или над операциями, охватыва-ющими все переменные, входящие в операцию (*x*, *y*, или1, 0, или *x* + *y*). Отрицание может обозначаться знаком апостроф «’» после имени пере-менной.

1.1 Аксиомы

Аксиомы или постулаты математической системы — это на-бор основных утверждений, про которые мы предполагаем, что они справедливы, и из которых можно вывести все другие свойства системы .

$\uparrow x\ 0,$ ⨍ ł Ч G $x\ \zeta\ 1;$ (1.1)

$\rightarrow$

$\downarrow x\ 1,$ ⨍ ł Ч G x

$\zeta\ 0.$

$\uparrow 0\ 0\quad 0,$ (1.3)

$\rightarrow$

$\downarrow 1*1\quad 1.$

$\uparrow\bar{\rightarrow} 0\quad 1,$ (1.5)

$\downarrow\bar{1}\quad 0.$

$\uparrow 1\ \ 1\quad 1,$ (1.2)

$\rightarrow$

$\downarrow 0 * 0\quad 0.$

$\uparrow 0\ 1\quad 1\ 0\quad 1,$ (1.4)

$\rightarrow$

$\downarrow 1* 0\quad 0 *1\quad 0.$

Формула (1.1) утверждает, что в алгебре логики рассматриваются только двоичные переменные.

Формулы (1.2)—(1.4) определяют операции дизъюнкции и конъюнкции.

Формула (1.5) определяет операцию отрицания.

1.2 Теоремы и тождества

На основании аксиом алгебры логики можно вывести ряд теорем и законов.

Тогда

$$a_S = F(a_m, z),$$

где a_m — состояние автомата в момент времени t, z — входной сигнал в момент времени t,

a_S — состояние автомата в момент времени t + 1; $w_m = G(a_q, z_n)$.

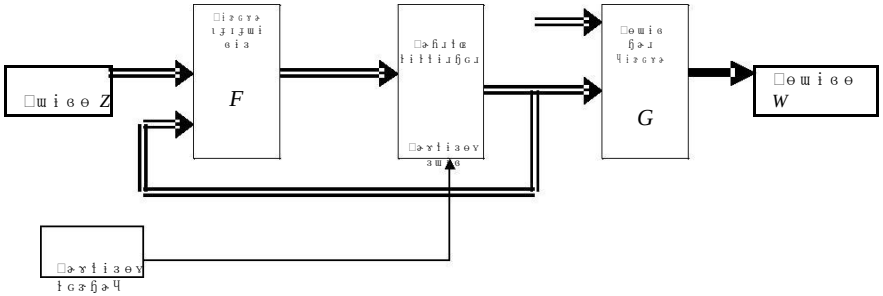


Рис. 44. Общая структура конечного автомата (автомат Мили)

Последовательностная схема (рис. 44), выход которой зависит как от состояния, так и от входа, называется *автоматом Мили* (Mealy machine)

Память состояния конечного автомата может быть построена на *D*-триггерах, переключающихся по положительному фронту.
В этом случае все события происходят в моменты времени, соответствующие нарастающим фронтам тактового сигнала. Возможно также использование в памяти состояния *JK*-триггеров и *RS*-триггеров.

В некоторых приложениях, где используются конечные автоматы, выход зависит только от текущего состояния автомата:

$$w_m = G(a_q).$$

Такая схема называется *автоматом Мура* (Moore machine) (рис. 45).

Единственное различие между автоматом Мили и автоматом Мура заключается в способе выработки выходных сигналов.

Лабораторная работа № 4

Цифровые автоматы

Цель работы: изучение и синтез цифровых схем на базе цифровых автоматов. Изучение особенностей реализации конечных автоматов.

1 Краткие сведения из теории

Конечный автомат — это общее название синхронных тактируемых последовательностных цифровых схем, которые принимают последующее состояние в зависимости от входных сигналов и текущего состояния. Слово «тактируемый» указывает на тот факт, что элементы памяти в конечном автомате (триггеры) имеют тактовый вход. Слово «синхронный » означает, что на тактовые входы всех триггеров подается один и тот же тактовый сигнал. Состояние такого конечного автомата изменяется только на очередном такте, а между тактами — остается неизменным.

1.1 Тактируемые синхронные конечные автоматы

На рис. 44 приведена общая структура синхронного тактируемого конечного автомата Мили. Память состояния представляет собой набор из n триггеров, в которых хранится текущее состояние автомата; всего имеется 2^n различных состояний. Все триггеры подключены к общему источнику тактового сигнала, который позволяет им изменять состояние на каждом такте. Следующее состояние конечного автомата определяется *логикой (функцией) переходов F* и является функ-цией текущего состояния и входного воздействия. Выходные сигналы определяются *выходной логикой G* и также зависят от текущего состоя-ния и входного воздействия. Оба блока *F* и *G* являются строго комби-национными схемами.

Пусть $A = \{a_1, a_2, \ldots, a_q\}$ — множество внутренних состояний ко-нечного автомата, $Z = \{z_1, z_2, \ldots, z_n\}$ — множество входных сигналов, $W = \{w_1, w_2, \ldots, w_m\}$ — множество выходных сигналов.

1. Идемпотентные законы: $x \cdot x = x$, $x + x = x$.
2. Коммутативные законы: $x \cdot y = y \cdot x$, $x + y = y + x$.
3. Ассоциативные законы: $(x \cdot y) \cdot z = x \cdot (y \cdot z)$, $(x + y) + z = x + (y + z)$.
4. Дистрибутивные законы: $x + y \cdot z = (x + y) \cdot (x + z)$, $x \cdot (y + z) = x \cdot y + x \cdot z$.
5. Законы отрицания: $x + \bar{x} = 1$, $x \cdot \bar{x} = 0$.
   $0 + x = x$, $0 \cdot x = 0$.
   $1 + x = 1$, $1 \cdot x = x$.
6. Законы двойственности (теоремы де Моргана): $\overline{x \cdot y} = \bar{x} + \bar{y}$, $\overline{x + y} = \bar{x} \cdot \bar{y}$.
7. Закон двойного отрицания: $\bar{\bar{x}} = x$.
8. Законы поглощения: $x + xy = x$, $x \cdot (x + y) = x$.
9. Операции склеивания: $xy + x\bar{y} = x$, $(x + y)(x + \bar{y}) = x$;
   $x + \bar{x}y = x + y$, $x(x + y) = xy$.

1.3 Логические функции одной и двух переменных

Логические функции одной переменной

Рассмотрим каждую функцию:

1. $f_0 = 0$ — нулевая функция.
2. $f_1 \quad x$ — функция повторения.
3. $f_2 \quad \bar{x}$ — функция отрицания.
4. $f_3 = 1$ — единичная функция.

Логические функции двух переменных (см. табл. 1)

Таблица 1. Таблица истинности всех функций двух переменных

v_i	x_1, x_0	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0 0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0 1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
2	1 0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
3	1 1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

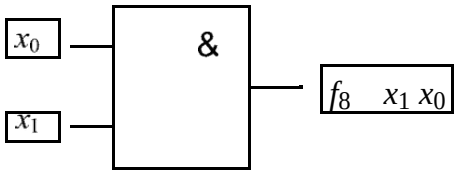
где v_i — десятичный номер комбинаций переменных x_1, x_0.

Следующие функции разберем подробно.

Функция логического умножения (конъюнкция)

$f_8 \quad x_1 x_0$ — логическое умножение, описывает работу логического элемента И.

v_i	x_1, x_0	f_8
0	0 0	0
1	0 1	0
2	1 0	0
3	1 1	1



Функция логического сложения (дизъюнкция)

$f_{14} \; x_1 \quad x_0$ — логическое сложение, описывает работу логического элемента ИЛИ.

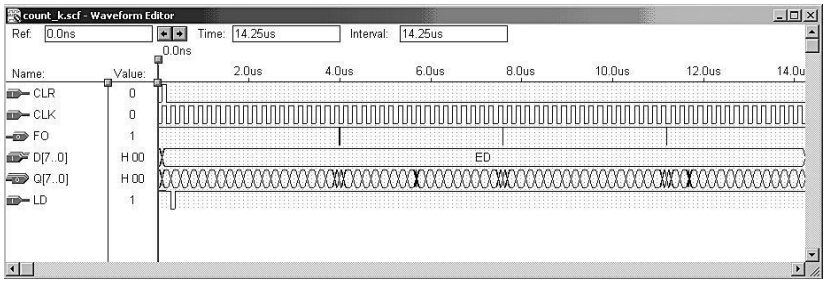


Рис. 43. Временная| диаграмма работы синхронного счетчика с $К_{сч} = 19$

6. Исследуем с помощью осциллографа разработанную схему. Для отображения входных и выходных сигналов делителя используем раз-ные каналы осциллографа. Замеряем длительность периода входного и выходного сигналов.

Требования к отчету

Отчет по лабораторной работе должен выполняться в отдельной тетради и содержать:

- Название лабораторной работы, ее цель, задачи.
- Вариант задания.
- Схему синхронного счетчика.
- Временные| диаграммы работы счетчика.
- Схему делителя частоты с произвольным коэффициентом счета.
- Временные| диаграммы работы схемы делителя.
- Значения периодов входного и выходного сигналов, измеренные на осциллографе.

Вопросы и задания для самопроверки

- К какому типу цифровых схем относятся синхронные счетчики?
- К какому типу цифровых схем относятся асинхронные счетчики?
- Разработать вычитающий трехразрядный счетчик с последовательным переносом на *JK*-триггерах.
- Построить временные| диаграммы работы восьмиразрядного ре-гистра 74273 (библиотека mf).
- Построить временные| диаграммы работы восьмиразрядного ре-гистра 74374.

Разработаем схему делителя частоты на базе счетчика 74193. Прежде всего необходимо вычислить двоичный код для загрузки. Пусть требуемый коэффициент счета $К_{сч} = 19$.

• Определим требуемое количество разрядных триггеров:

$$n = [\log_2 К_{сч}] = [\log_2 19] = [\sim 5] = 5.$$

Поскольку счетчик 74193 имеет число разрядов кратное 4, то нам необходимо использовать два элемента 74193. С учетом этого примем $n = 8$.

• Определим число избыточных состояний M:

$$M = 2^n - К_{сч} = 2^8 - 19 = 256 - 19 = 237.$$

• Переведем десятичное число 237 в двоичный код восьмиразряд-ного числа ($n = 8$):

$$237_{10} = 10001001_2.$$

Таким образом, получили следующие исходные данные для разработки делителя:

• число разрядов счетчика делителя $n = 8$;

• параллельный код для предзагрузки в счетчики — 11101101 (0*xED* — шестнадцатеричный код числа).

На рис. 42 изображена схема делителя с коэффициентом пересчета, определяемым двоичным числом, подаваемым на линии данных *D* [7..0].

5. Выполняем временное| моделирование синтезированной схемы (рис. 43).

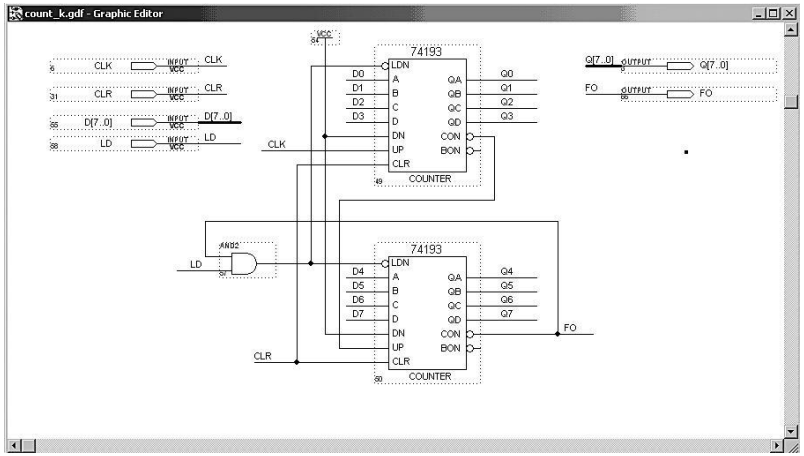
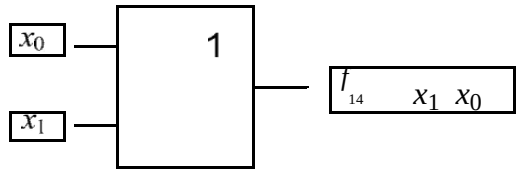


Рис. 42. Схема делителя частоты

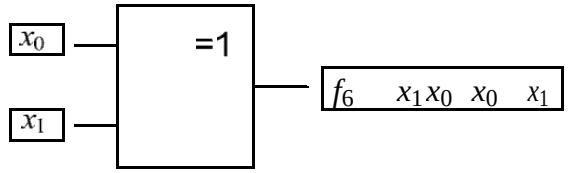
v_i	x_1, x_0	f_{14}
0	0 0	0
1	0 1	1
2	1 0	1
3	1 1	1



Функция сложения по модулю два (исключающее ИЛИ, неравнозначность)

f_6 x_1 x_0 x_0 x_1 x_0 $\overline{x_1}$ — сложение по модулю два, применя-ется для арифметического сложения.

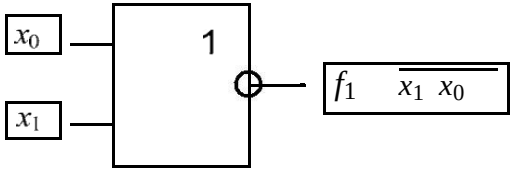
v_i	x_1, x_0	f_6
0	0 0	0
1	0 1	1
2	1 0	1
3	1 1	0



Функция Пирса — логическое сложение с отрицанием, отрицание дизъюнкции (стрелка Пирса ИЛИ-НЕ)

f_1 x_1 π x_0 x_0 x_1 — логическое сложение с отрицанием ИЛИ-НЕ.

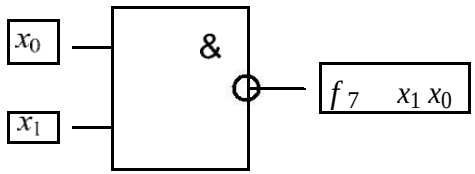
v_i	x_1, x_0	f_1
0	0 0	1
1	0 1	0
2	1 0	0



Функция Шеффера — отрицание от логического умножения (штрих Шеффера И-НЕ)

f_7 x_1 _ x_0 $\overline{x_0\ x_1}$ — логическое умножение с отрицанием И-НЕ.

v_i	x_1, x_0	f_7
0	0 0	1
1	0 1	1
2	1 0	1
3	1 1	0



Функции двух переменных исключительно важны в силу того, что любая логическая функция *n* переменных может быть получена из них методом суперпозиции — подстановкой этих функций вместо пере-менных в другие функции.

1.4 Анализ комбинационных схем

Анализ комбинационной схемы заключается в первую очередь в формальном описании логической функции, которую реализует эта схема. Получив описание логической функции, можно:

• Определить реакцию схемы на различные комбинации входных воздействий.

• Преобразовать алгебраическую запись и создать другую структуру схемы, реализующей эту логическую функцию.

• Преобразовать алгебраическую запись так, чтобы подогнать ее под имеющийся набор цифровых схем.

Если имеется графическое изображение комбинационной схемы, например такое, как на рис. 13, то существует несколько способов по-лучить формальное описание функции, которую реализует эта схема. Самым простым функциональным описанием является таблица истин-ности.

На рис. 13 представлена комбинационная схема с тремя входами и одним выходом. Используя только основные аксиомы алгебры логики можно составить таблицу истинности для схемы с *n* входами, прослеживая путь от входов к выходам для всех 2^n комбинаций входных сигналов. Для каждой такой комбинации определяются сигналы, возникающие на выходах всех вентилей под действием данных входных сигналов.

На рис. 14 у каждой входной линии ($x1$, …, $x3$) выписаны после-довательности из восьми логических значений, когда на эти входные линии по очереди подаются сигналы 000, 001, …, 111.

74193 (аналог К155ИЕ7) является 4-разрядным синхронным счетчиком с параллельной загрузкой. Параллельная загрузка (сигналом *ld*) предполагает предварительную установку выходов счетчика в состояние, кодовая комбинация которого задана на входах счетчика.

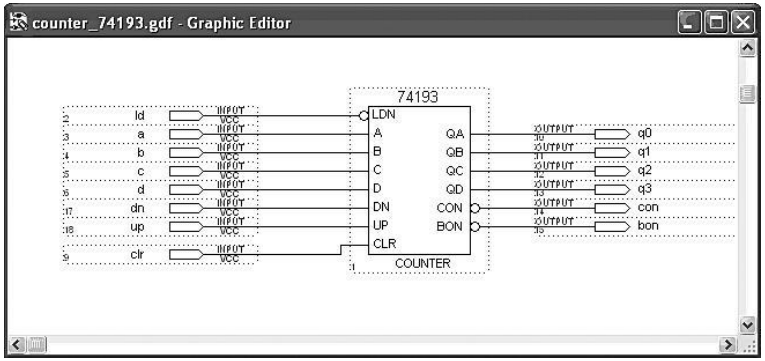


Рис. 40. Синхронный счетчик со входом параллельной загрузки

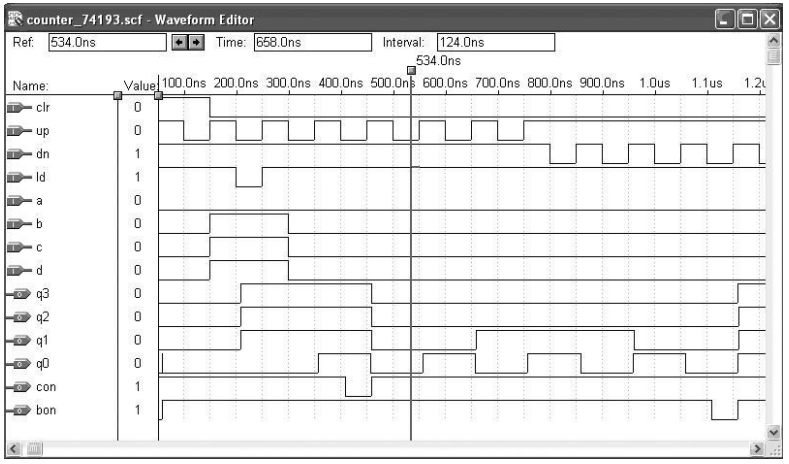


Рис. 41. Временная| диаграмма работы синхронного счетчика 74193

3. Загружаем конфигурационный файл в ПЛИС и исследуем временные| характеристики счетчика с помощью осциллографа.

4. Делаем расчет делителя по заданному коэффициенту счета $К_{сч}$.

ет несколько методов получения счетчиков с заданным коэффициентом счета $К_{сч}$. Один их этих методов заключается в немедленном сбросе в «0» счетчика, установившегося в комбинацию, соответствующую $К_{сч}$.

2 Задание для лабораторной работы № 3

Изучить структуру и алгоритм работы синхронных счетчиков. Разработать проект цифровой схемы, включающий делитель частоты с переменным коэффициентом деления на синхронных счетчиках. Ко-эффициент счета $К_{сч}$ задается соответствующим вариантом.

Выполнить следующие действия:

• для изучения работы синхронных счетчиков в пакете MAX+plus II создать проект, включающий синхронный счетчик 74193 из библиотеки mf;

• выполнить компиляцию проекта;

• в редакторе Waveform Editor задать входные и выходные сигналы счетчика и выполнить временное моделирование схемы;

• загрузить конфигурационный файл в ПЛИС;

• исследовать временные характеристики счетчика с помощью осциллографа и сравнить их с временной моделью, полученной в Waveform Editor;

• разработать проект цифровой схемы, включающий делитель частоты с переменным коэффициентом деления на синхронных счетчиках 74193;

• выполнить компиляцию проекта;

• в редакторе Waveform Editor задать входные и выходные сигналы делителя и выполнить временное моделирование схемы;

• с помощью входных переключателей стенда (см. инструкцию) задавать коды делителя, соответствующие коэффициенту счета $К_{сч}$, и смотреть на осциллографе входные и выходные частоты делителя.

3 Пример выполнения лабораторной работы № 3

1. Создаем проект count_74193.gdf (см. рис. 40) и выполняем ком-пиляцию.

2. В редакторе Waveform Editor задаем входные и выходные сигналы (рис. 41) счетчика и выполняем временное моделирование. Счетчик

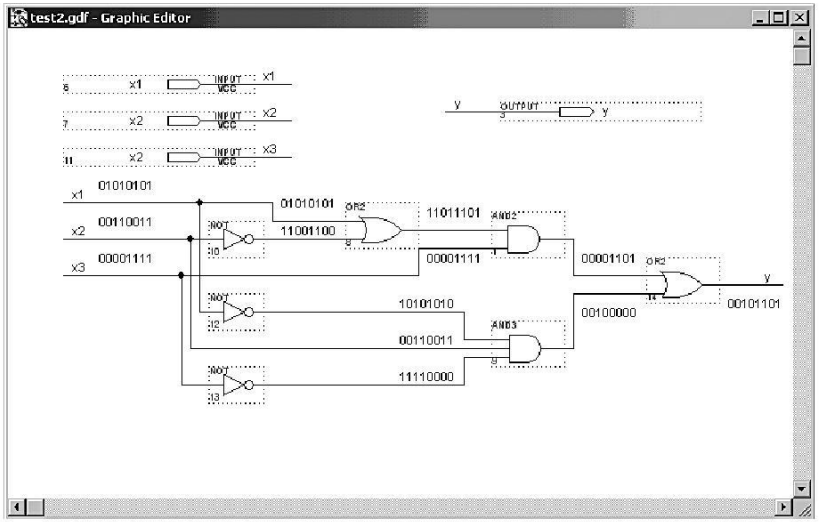


Рис. 13. Логическая схема с тремя входами *x* [3..1] и одним выходом *y*

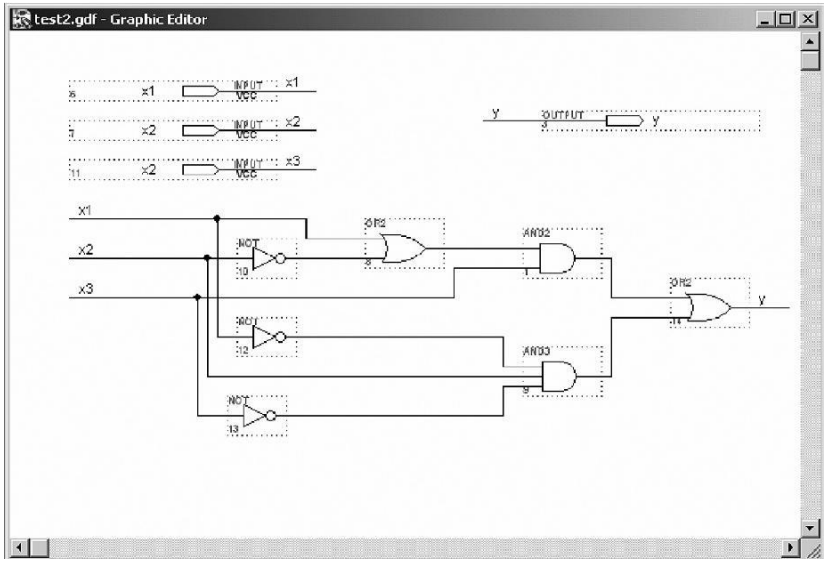


Рис. 14. Логическая схема с комбинациями входных значений (восемь комбинаций)

Составив таблицу истинности для данной схемы, далее можно прямо написать логическое выражение в виде канонической суммы или канонического произведения. Данная таблица истинности представлена в табл. 2.

Таблица 2. Таблица истинности для схемы, представленной на рис. 14

v_i	$x3$	$x2$	$x1$	y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Каноническая сумма логической функции есть сумма минтермов, соответствующих тем строкам таблицы истинности (комбинациям входных сигналов x [3..1]), для которых значение функции (y) равно 1. Для нашего случая

$$y = x_1' \,\&\, x_2 \,\&\, x_3' + x_1' \,\&\, x_2' \,\&\, x_3 + x_1 \,\&\, x_2' \,\&\, x_3 + x_1 \,\&\, x_2 \,\&\, x_3.$$

Знак апостроф «’» после переменной означает инверсию этой переменой. Знак «&» между переменными означает одну из форм записи логического умножения (конъюнкции). Далее запишем каноническое произведение, то есть произведение макстермов, соответствующее тем комбинациям входных сигналов, для которых значение функции равно 0.

$$y = (x_1' + x_2' + x_3') \,\&\, (x_1 + x_2' + x_3') \,\&\, (x_1 + x_2 + x_3') \,\&\, (x_1' + x_2 + x_3).$$

Поскольку число комбинаций входных сигналов растет экспоненциально с увеличением числа входов, то перебирать все возможные комбинации не оптимально. Другой способ получения логического выражения — алгебраический. Для каждого вентиля пишется логическое выражение, начиная от входа и до выхода (рис. 15), при этом учитываются значения функции каждого элемента.

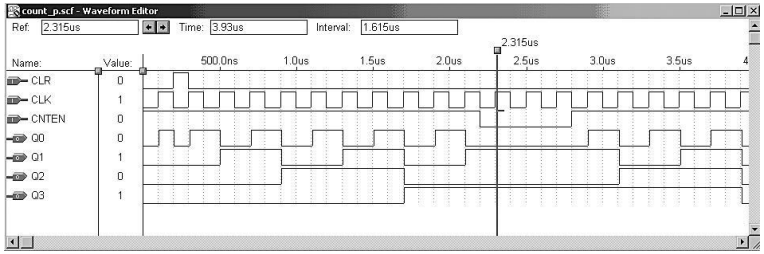


Рис. 39. Временная| диаграмма работы синхронного счетчика

Поскольку счетчик имеет одну общую линию тактирования (синхронизации), состояние триггеров меняется синхронно, то есть те триггеры, которые по переднему фронту тактового сигнала должны изменить свое состояние, делают это одновременно, что существенно повышает быстродействие синхронных счетчиков.

1.8 Счетчики с произвольным коэффициентом счета

Принцип построения счетчиков с произвольным коэффициентом счета состоит в исключении нескольких состояний обычного двоично-го счетчика, являющихся избыточными для счетчика с коэффициентом пересчета, отличающегося от двоичного. При этом избыточные состояния исключаются с помощью обратных связей внутри счетчика.

Число избыточных состояний (M) для любого счетчика определя-ется из следующего выражения:

$$M = 2^m - К_{сч},\ m$$

где $К_{сч}$ — требуемый коэффициент счета; 2 — число устойчивых состояний двоичного счетчика.

Задача синтеза счетчика с произвольным коэффициентом счета заключается в определении необходимых обратных связей и минимизации их числа. Требуемое количество триггеров (n) определяется из выражения

$$n = [\log_2 К_{сч}],$$

где $[\log_2 К_{сч}]$ — двоичный логарифм заданного коэффициента сче-та $К_{сч}$, округленный до ближайшего целого числа.

В каждом отдельном случае приходится применять какие-то конкретные методы получения требуемого коэффициента счета. Существу-

довательные счетчики быстрее асинхронных последовательных счетчиков, но если период тактового сигнала слишком мал, то изменение в младшем разряде (перенос информации) может не успеть дойти за это время до старшего разряда.

Разработаем схему четырехразрядного синхронного параллельно-го счетчика. Для этого из библиотеки примитивов пакета **MAX+plus II** на рабочее поле скопируем элементы **dffe** — *D*-триггер с входами предустановки и разрешения, **and2** — элемент 2И, **and3** — элемент 3И, **and4** — элемент 4И, **not** — инвертор. Счетчик будет иметь сигнал сброса *CLR*, тактовый сигнал *CLK* и сигнал разрешения счета *CNTEN*. На рис. 38 представлен синхронный параллельный счетчик на *D*-триг-герах, включенных в счетном режиме. Каждый триггер имеет вход раз-решения работы и входы предустановки.

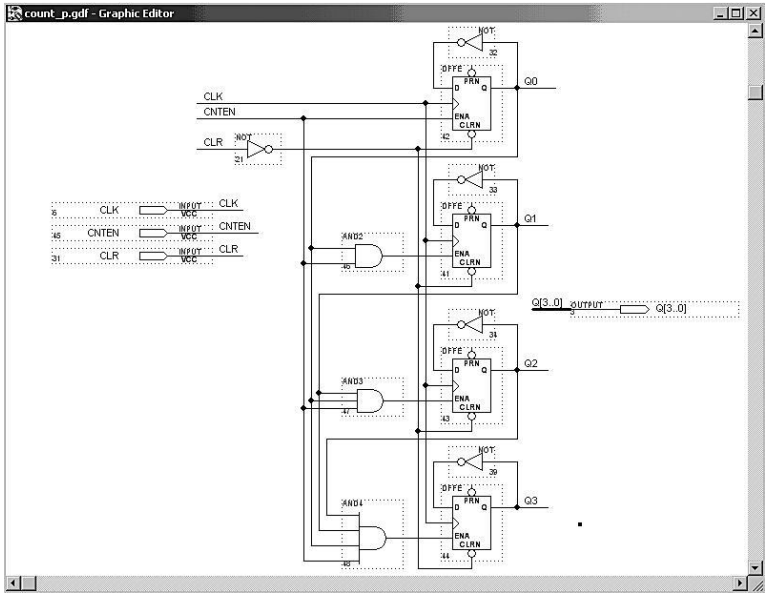


Рис. 38. Четырехразрядный синхронный параллельный счетчик на *D*-триггерах

Если сигнал разрешения счета *CNTEN* активный (лог. 1), то счетчик меняет свое состояние (см. рис. 39), в противном случае выходы счетчика остаются без изменений.

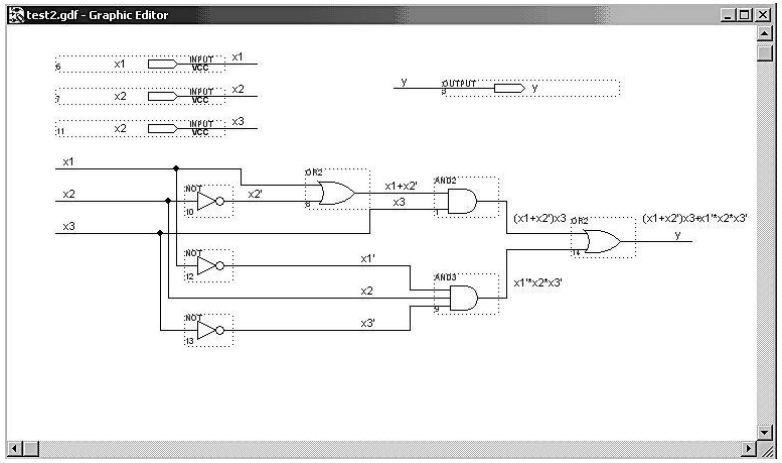


Рис. 15. Логические выражения для различных сигнальных линий

Таким образом, логическое выражение для приведенной схемы выглядит так:

$$y = (x_1 + x_2') \,\&\, x_3 + (x_1' \,\&\, x_2 \,\&\, x_3').$$

1.5 Минимизация комбинационных схем

Одной из основных задач, возникающих при синтезе комбинационных схем (КС), является минимизация логических функций, которые эти КС реализуют. Чем проще логическое выражение, описывающее функцию, тем проще и дешевле реализующая ее КС.

Существуют два метода минимизации:

– аналитический, весьма трудоемкий и требующий сложного под-хода, который не всегда виден;

– графический, наиболее наглядный, простой в использовании, но имеющий некоторые ограничения.

Очевидно, что любой метод минимизации может основываться только на тождественном преобразовании логических выражений. Су-ществует несколько способов минимизации булевых функций. Прежде всего это метод Квайна—МакКласки и метод минимизации с помощью карт Карно или диаграмм Вейча [4].

Рассмотрим минимизацию КС с помощью графического метода — карт Карно. Карты Карно представляют собой один из таблич-ных способов задания функций и состоят из клеток, каждая из которых соответствует определенной точке v_i области определения функций. Карты Карно для функции n переменных состоят из 2^n клеток, которые нумеруются числами от 0 до $2^n - 1$. Чтобы с помощью такой карты задать функцию $f(v)$, необходимо в каждую клетку с номером i занести значение функции $f(v_i) = 0$ или 1, которое оно принимает в точке v_i.

Существуют карты Карно на 2, 3, 4, 5 и 6 переменных [3]. На рис. 16 представлены такие карты Карно.

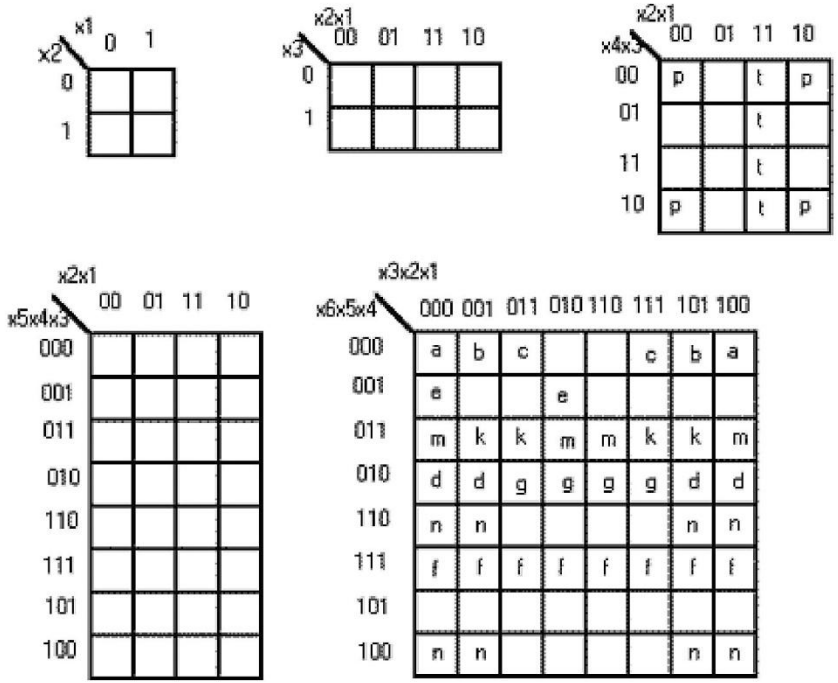


Рис. 16. Карты Карно для 2, 3, 4, 5 и 6 переменных

Метод Карно основан на законе склеивания. Склеиваются наборы, отличающиеся друг от друга лишь значением одного разряда. Такие наборы называются соседними. Карно закодировал клетки своей карты

Такой счетчик называется *счетчиком с последовательным переносом*, поскольку информация о переносе поочередно передается от младшего разряда к старшему.

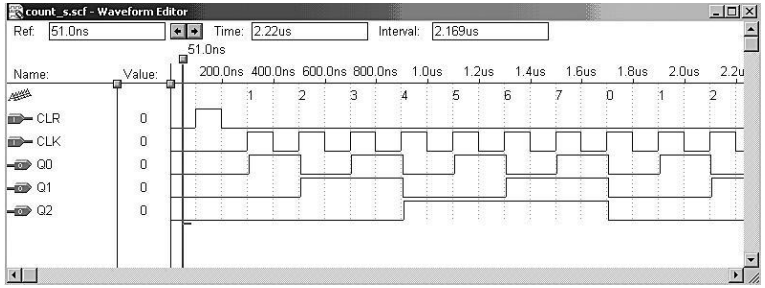


Рис. 37. Временная диаграмма работы суммирующего счетчика

Из рис. 37 видно, что при активном сигнале сброса *CLR* все выхо-ды счетчика принимают нулевое состояние. При каждом положительном фронте сигнала *CLK* происходит увеличение значения счетчика от 0 до 7.

Рассмотренный тип счетчиков может быть использован в цифровых устройствах «умеренного» быстродействия, когда частота следования синхроимпульсов не превышает критического значения, при котором время задержки установки триггеров последних (старших) разрядов счетчика становится соизмеримым с длительностью периода входных тактовых импульсов. В связи с этим, асинхронные счетчики строятся на относительно небольшое количество разрядов, так как при большем количестве разрядов выходные сигналы триггеров старших разрядов появляются позднее, чем управляющие фронты синхроим-пульсов (поступающих на вход первого триггера).

1.7 Синхронные счетчики

В синхронном счетчике к тактовым входам всех триггеров подводится один и тот же тактовый сигнал *CLK*, так что изменение значений сигналов на выходах всех триггеров происходит в один и тот же мо-мент времени. Синхронные счетчики могут быть с последовательным переносом (синхронный последовательный счетчик) и параллельным переносом (синхронный параллельный счетчик). Синхронные после-

вательную синхронизацию, то есть каждый последующий разрядный триггер синхронизируется выходными импульсами триггера предыду-щего разряда. Асинхронные счетчики иногда называют счетчиками с последовательным переносом или последовательными, а синхронные счетчики — параллельными.

Разработаем трехразрядный счетчик с последовательным перено-сом на *D*-триггерах и общим сигналом сброса *CLR* (см. рис. 36). У это-го счетчика каждый разрядный *D*-триггер включен по схеме счетного триггера, то есть инверсный выход соединен с *D*-входом. Прямые вы-ходы триггеров служат выходами счетчика соответствующего разряда (*Q*0, *Q*1, *Q*2), а инверсные выходы каждого триггера являются такто-выми сигналами для следующего триггера. Поскольку счетчик суммиру-ющий, то каждый положительный фронт сигнала *CLK* увеличивает значение счетчика на единицу.

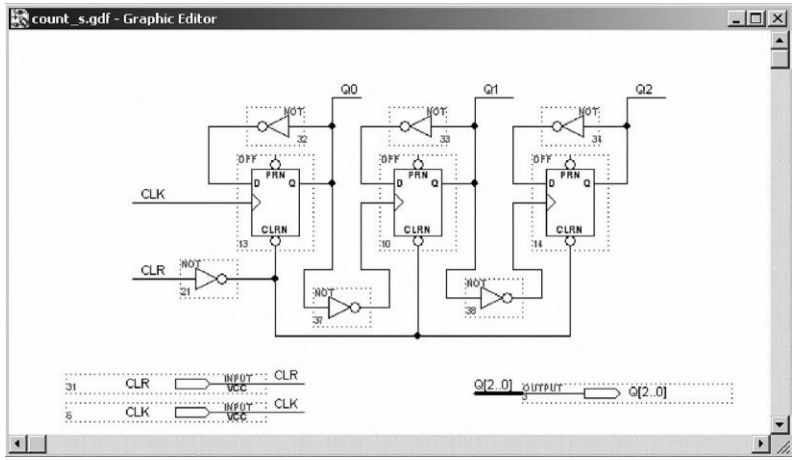


Рис. 36. Трехразрядный счетчик на *D*-триггерах

Из временной| диаграммы (рис. 37) следует, что содержимое того или иного разряда счетчика меняется на противоположное тогда и только тогда, когда значение инверсного выхода триггера предыдущего разряда меняется с 0 на 1. Это соответствует двоичному счету в пря-мом направлении, когда бит, хранящийся в данном разряде, изменя-ется с 1 на 0 и возникает перенос информации в следующий разряд.

так, что в соседних клетках оказались соседние, а значит склеивающие-ся наборы. Соседними могут быть не только отдельные клетки, которые мы назовем элементарными квадратами Карно, но и целые группы кле-ток (назовем их прямоугольниками Карно). Под прямоугольником Кар-но будем понимать некоторую, зачастую разрозненную, фигуру покры-тия, все соседние клетки которой закодированы соседними наборами. Например, на рис. 16 в поле карты для 4 переменных изображен пря-моугольник Карно *P*. Этот прямоугольник- состоит из четырех элемен-тарных квадратов Карно, описываемых наборами *x*4’ & *x*3’ & *x*2’ & *x*1’; *x*4’ & *x*3’ & *x*2 & *x*1’; *x*4 & *x*3’ & *x*2’ & *x*1’; *x*4 & *x*3’ & *x*2 & *x*1’. Если над логи-ческой суммой этих четырех наборов произвести последовательно опе-рации склеивания, то в результате получим *x*3’ & *x*1’. Карта Карно позво-ляет получить этот результат графически значительно быстрее и проще. Для решения этой задачи используем алгоритм графической миними-зации с определенными правилами:

- Заполнить карту Карно нулями и единицами в соответствии с таблицей истинности.
- Покрыть все единичные наборы минимальным количеством пря-моугольников Карно, каждый из которых имеет максимальную пло-щадь.
- Сторона прямоугольника Карно должна быть кратна степени числа 2 (то есть 1, 2, 4, 8, …). Таким образом, прямоугольник 3 × 4 не является прямоугольником Карно.

1.6 Синтез комбинационных логических схем

Синтезом комбинационных логических схем будем называть ме-тодику создания КЛС, включающую следующие этапы:

1) задание функции (функций) с помощью таблицы истинности на основе поставленной задачи;
2) алгебраическая запись функции в виде совершенной дизъюнк-тивной нормальной формы (СДНФ) и совершенной конъюнктивной нормальной формы (СКНФ), определение более выгодного варианта;
3) минимизация функции (функций);
4) анализ и при необходимости изменение функций на возмож-ность совместной реализации;
5) выбор базиса функции (функций) и приведение ее к этому базису;

6) построение КЛС на логических элементах, удовлетворяющих выбранному базису.

2 Задание для лабораторной работы № 1

Провести синтез комбинационной логической схемы, реализующей функцию $y = f(x_1, x_2, x_3, x_4)$ в базисах И-ИЛИ-НЕ. Функция $y = f(x_1, x_2, x_3, x_4)$ задана входными наборами таблицы истинности, определяемой вариантом (см. приложение А).

Выполнить следующие действия:

• записать СДНФ функции;

• минимизировать заданную функцию с помощью карт Карно;

• в пакете MAX+plus II создать проект, из библиотечных элементов нарисовать получившуюся минимальную функцию в графическом редакторе Graphic Editor;

• выполнить компиляцию проекта;

• в редакторе Waveform Editor выполнить временное| моделирова-ние КЛС;

• загрузить конфигурационный файл в ПЛИС;

• с помощью входных переключателей стенда (см. инструкцию) задать входные последовательности КЛС (переменные x_1, x_2, x_3, x_4) и смотреть на светодиодном индикаторе значение функции (y).

3 Пример выполнения лабораторной работы № 1

Задание

Провести синтез КЛС, реализующей функцию $y = f(x_1, x_2, x_3, x_4)$ в базисах И-ИЛИ. Функция $y = f(x_1, x_2, x_3, x_4)$ задана входными наборами (2, 4, 6, 8, 13, 14, 15) таблицы истинности, на которых она равна 1.

1. СДНФ для заданной функции имеет вид:

$y = x4' \,\&\, x3' \,\&\, x2 \,\&\, x1' + x4' \,\&\, x3 \,\&\, x2' \,\&\, x1' +$

$+ x4' \,\&\, x3 \,\&\, x2 \,\&\, x1' + x4 \,\&\, x3' \,\&\, x2' \,\&\, x1' +$

$+ x4 \,\&\, x3 \,\&\, x2' \,\&\, x1 + x4 \,\&\, x3 \,\&\, x2 \,\&\, x1' +$

$+ x4 \,\&\, x3 \,\&\, x2 \,\&\, x1.$

формация подается по линиям данным D [7..0]. Как видно из рис. 34 регистр имеет восемь D-триггеров, которые одновременно фиксируют входную информацию на выходы по нарастающему фронту сигнала *CLK*. Установка всех триггеров в исходное состояние выполняется с помощью сигнала *CLR*. Рассмотрим временные диаграммы работы регистра. Для этого зададим в Waveform Editor значения входных сигналов D [7..0] равными 0x34 (Hex). В двоичном коде это соответствует значению 00110100 (справа всегда младший разряд D0). По передне-му фронту сигнала *CLK* (вертикальный маркер) код 0x34 фиксируется на выходах D-триггеров (рис. 35). На выходах регистра Q [7..0] инфор-мация достоверная только при активном управляющем сигнале *OE*. Все остальное время выходы регистра находятся в третьем состоянии (третье состояние обозначено *ZZ*).

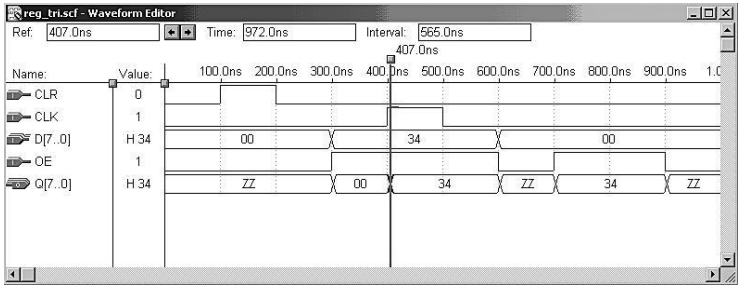


Рис. 35. Временная| диаграмма работы параллельного регистра

1.6 Счетчики с последовательным переносом

Счетчиком называется тактируемая последовательностная схема, выполненная на *n-триггерах*, диаграмма состояний которых представ-ляет собой кольцо. То есть за последним состоянием следует первое. Счетчики позволяют вести подсчет электрических импульсов, количе-ство которых (поступивших на тактовый вход счетчика) представляет-ся, обычно, в параллельном коде.

Счетчики подразделяются на *синхронные* и *асинхронные*. У синхронных счетчиков все разрядные триггеры синхронизируются параллельно одними и теми же синхроимпульсами, поступающими из источника этих импульсов. Асинхронные счетчики имеют последо-

используются специальные буферные элементы с тремя состояниями. Такие элементы имеют на выходе кроме логических состояний 0 и 1 состояние «отключено», в котором ток выходной цепи пренебрежимо мал. В это состояние (третье) элемент переводится с помощью специального управляющего сигнала. Таким образом, при считывании информации из 8-разрядных регистров по шине данных только один регистр подключен, а выходы остальных находятся в третьем состоянии.

Разработаем восьмиразрядный регистр, состоящий из переключающихся по фронту *D*-триггеров. Для этого из библиотеки примитивов на рабочее поле скопируем элементы: **dff** — *D*-триггер с входами предустановки, **tri** — буферный элемент с тремя состояниями и **not** — инвертор.

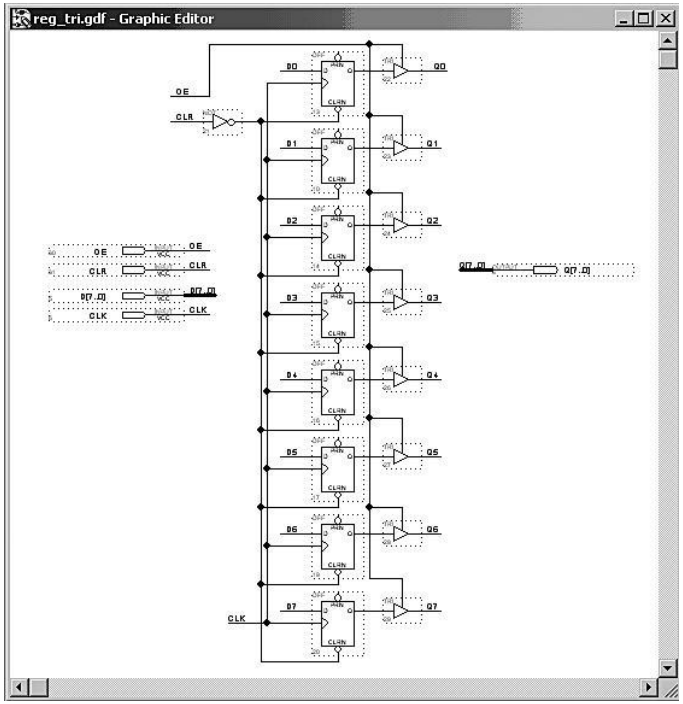


Рис. 34. Восьмиразрядный регистр на переключающихся по фронту *D*-триггерах

Сигнал с выхода каждого триггера поступает на буфер с тремя состояниями и далее на выходной элемент схемы (output). Входная ин-

Таблица истинности функции:

v_i	x_4	x_3	x_2	x_1	y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

2. Минимизируем заданную функцию с помощью карты Карно. Для получения минимальной дизъюнктивной нормальной функции (ДНФ) объединяем в группы смежные (соседние) единичные клетки.

Минимальная ДНФ имеет вид:

*x*4’ & *x*2 & *x*1’ + *x*4 & *x*3 & *x*1□+□*x*4 & *x*3 & *x*2□+□
+ *x*4’ & *x*3 & *x*1’□+□*x*4 & *x*3’ & *x*2’ & *x*1’.

x_4x_3 \ x_2x_1	00	01	11	10
00	0	0	0	1
01	1	0	0	1
11	0	1	1	1
10	1	0	0	0

3. Создаем проект в пакете MAX+plus II в Graphic Editor (рис. 17).

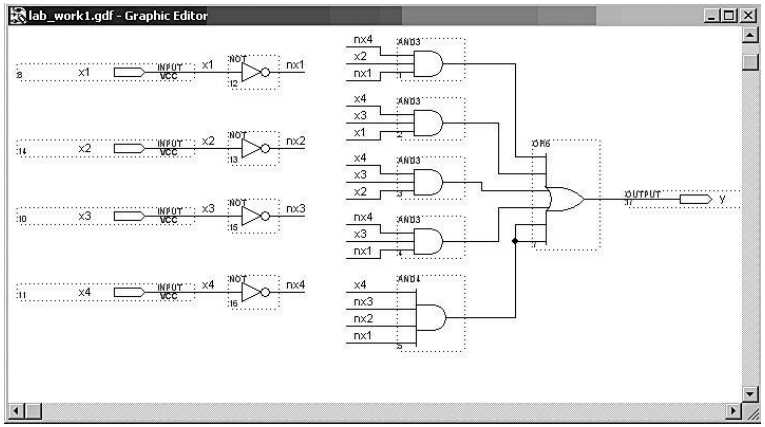


Рис. 17. Синтезированная КЛС

4. Моделируем в редакторе Waveform Editor.

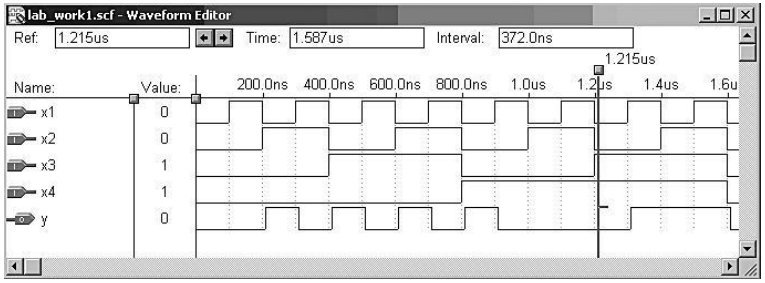


Рис. 18. Временное| моделирование синтезированной КЛС

Требования к отчету

Отчет по лабораторной работе должен выполняться в отдельной тетради и содержать:

1. Название лабораторной работы, ее цель, задачи.
2. Вариант задания.
3. СДНФ исходной функции.

на входе *D* не должен меняться. Этот сигнал находится в окрестности положительного фронта сигнала *CLK*.

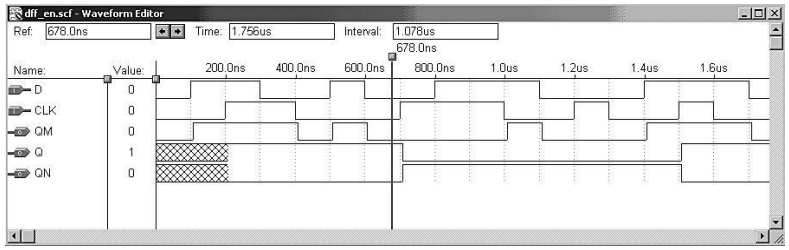


Рис. 33. Временная| диаграмма работы *D*-триггера

1.5 Регистр, переключающийся по фронту

Регистром называют совокупность из двух или более триггеров с общим входом тактового сигнала. Регистры часто применяют для промежуточного хранения набора связанных между собой (функционально однотипных) битов, например байт данных в компьютере. Но можно в регистре сохранять и не связанные биты, единственное ограничение — все биты будут запоминаться в регистре в один и тот же момент времени. Регистры можно классифицировать по способу приема и выдачи данных. По этому признаку различают параллельные, последовательные (сдвиговые) и параллельно-последовательные регистры. В параллельных регистрах прием и выдача слов производится одновременно. В последовательных регистрах биты информации принимаются и выдаются разряд за разрядом, то есть происходит сдвиг данных по разрядам от входа к выходу или обратно (реверсивный сдвиговый регистр). Параллельно-последовательные регистры име-ют входы-выходы одновременно параллельного и последовательного типа. В таких регистрах можно принимать информацию в параллель-ном виде, а передавать в последовательном.

Как правило, обрабатываемая в цифровых системах информация [4] представляется словами, состоящими из 8, 16 или 32 бит. Поэтому разрядность регистров кратна восьми. В сложных проектах использу-ется много регистров, выходы которых объединены в так называемые шины данных. Для считывания информации из определенного регистра

фронтом сигнала *CLK*. В этой схеме первая защелка называется ведущей (master): при значении *CLK*, равном 0, она открыта и ее выходной сигнал повторяет входной. Когда сигнал *CLK* переходит в состояние 1 (положительный фронт), ведущая защелка запирается и ее выходной сигнал переносится во вторую защелку, называемую ведомой (slave), и защелкивается там. Ведомая защелка открыта в течение всего времени, пока значение *CLK* остается равным 1, но изменение сигнала на ее выходе возможно только в самом начале этого интервала, поскольку ведущая защелка заперта и сигнал на ее выходе остается неизменным. Для изучения работы *D*-триггера в схему добавлен промежуточный сигнал *QM* с выхода ведущей защелки. Проект схемы, показанной на рис. 32, сохранен под именем dff_en.gdf.

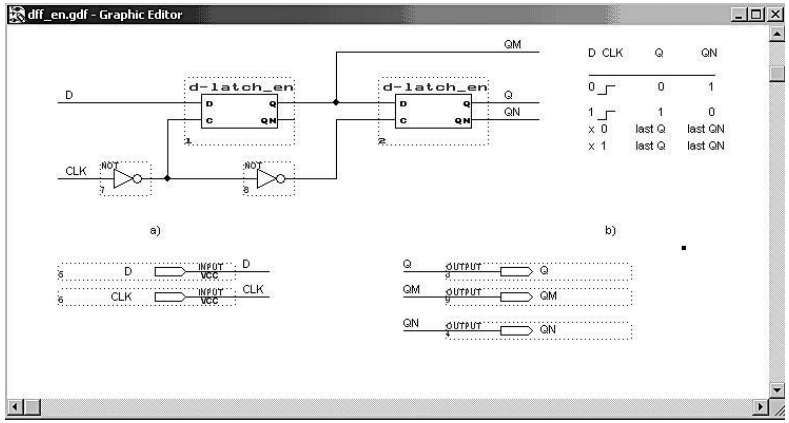


Рис. 32. *D*-триггер: *a*) принципиальная схема на *D*-защелках; *b*) таблица переходов

Рассмотрим работу приведенной схемы в окне редактора Waveform Editor. На рис. 33 показана временная| диаграмма работы *D*-триггера. Необходимо обратить внимание, что сигнал *QM* изменяется только при *CLK*, равном 0. Когда *CLK* становится равным 1, текущее значение *QM* переносится на выход *Q*, тогда как изменение сигнала *QM* невозможно до тех пор, пока *CLK* снова не станет равным 0. Так же, как и в случае *D*-защелки, у *D*-триггера есть интервал времени, состоящий из времени установления и времени удержания, в течение которого сигнал

4. Карту Карно, минимальную функцию, синтезированную схему.
5. Временную| диаграмму синтезированной КЛС.

Вопросы и задания для самопроверки

1. Назовите основные логические функции двух переменных.
2. Дайте определение логической функции СДНФ.
3. Назовите правила минимизации с помощью карт Карно.
4. Дано алгебраическое выражение функции: $y = (x1 + x2 + x3) \& (x3 + x4 + x5 + x6) \& (x6 + x7 + x8)$. Используя графические символы из библиотеки пакета MAX+plus II, разработайте и введите схему, эквивалентную заданному выражению. После компиляции создайте символ с именем проекта.

Лабораторная работа № 2

Дешифраторы

Цель работы: изучение работы дешифраторов, синтез логической схемы, заданной таблицей истинности, формирование селектора входных кодов.

1 Краткие сведения из теории

Дешифратор — это логическая схема с несколькими входами и несколькими выходами, которая преобразует кодированные входные сигналы в кодированные выходные сигналы, причем входные и выходные коды различны. Входной код обычно имеет меньшее число разрядов, чем выходной код, и между входными и выходными кодовыми словами имеет-ся взаимно-однозначное соответствие. В большинстве случаев роль вход-ного кода играет *N*-разрядный двоичный код, где *N*-разрядное двоичное слово представляет одну из 2_N различных кодированных величин.

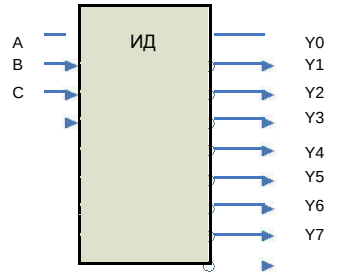


Рис. 19. Полный дешифратор 3 × 8

Самым распространенным является дешифратор $N \times 2_N$ или полный дешифратор. Полный дешифратор применяется в том случае, если необходимо активизировать только один из 2_N выходов. Каждой комбинации логических уровней на входах будет соответствовать активный уровень на одном из 2_N выходов. Обычно *N* равно 2, 3 или 4. На рис. 19 изображен дешифратор с $N = 3$ и 8 выходами.

Активным выходным уровнем является уровень логического нуля.

На входы *C*, *B*, *A* можно подать следующие комбинации логических уровней: 000, 001, 010, …, 111, всего 8 комбинаций. Схема имеет 8 выходов, на одном из которых формируется низкий потенциал, на остальных — высокий. Номер этого единственного выхода, на кото-ром формируется активный (нулевой) уровень, соответствует числу *N*, определяемому состоянием входов *C*, *B*, *A* следующим образом:

$$Y_j = m_j\,(C,B,A), \qquad (1)$$

Пример поведения *D*-защелки показан на рис. 31. Когда сигнал *C* активный, выходной сигнал *Q* повторяет значение входного сигнала *D*. Когда сигнал *C* снимается, защелка запирается; выходной сигнал *Q* сохраняет свое последнее значение и больше не реагирует на измене-ние сигнала *D*. В *D*-защелке нет проблемы, имеющейся в *SR*-защелке ($S = R = 1$), но остаются затруднения, связанные с метастабильностью.

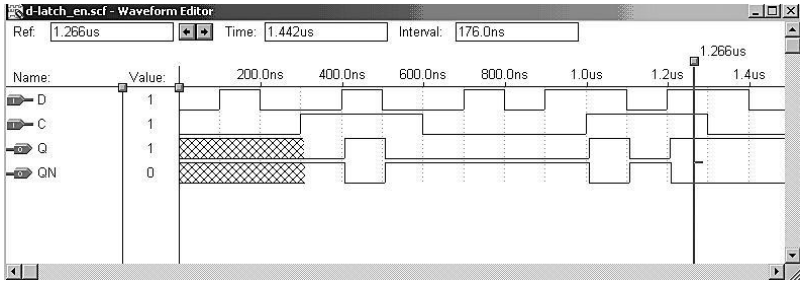


Рис. 31. Временная диаграмма работы *D*-защелки

В частности, в окрестности отрицательного фронта сигнала *C* (вертикальный маркер на рис. 31) существует интервал времени, в котором входной сигнал *D* не должен изменяться. Этот интервал начинается за время t_{setup} до отрицательного фронта сигнала *C* (это время называется *временем установления*) и заканчивается спустя время t_{hold} после отрицательного фронта *C* (это время называется *временем удержания*). Если входной сигнал *D* изменяется внутри этого интервала ($t_{setup} + t_{hold}$), то значение сигнала на выходе защелки непредсказуемо. Проект схемы, показанной на рис. 31, сохранен под именем d-latch_en.gdf.

Создадим из схемы проекта d-latch_en.gdf символ, на основе которого будем строить следующие устройства.

1.4 *D*-триггер, переключающийся по фронту

Рассмотрим схему, состоящую из двух *D*-защелок и двух инверторов. Эта схема называется *D-триггером*, переключающимся по положительному фронту тактового сигнала. На рис. 32 показа схема, в которой опрос ее входа *D* и изменение ее выходных сигналов *Q* и *QN* происходит только в моменты времени, задаваемые положительным

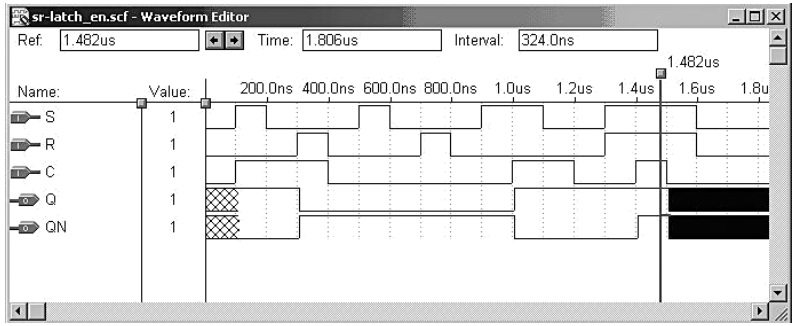


Рис. 29. Временная| диаграмма работы *SR*-защелки с входом разрешения

1.3 *D*-защелка

В цифровой схемотехнике очень часто бывают нужны защелки, чтобы просто запомнить биты информации, когда каждый бит поступает по отдельной сигнальной линии и его необходимо сохранить. В этом случае удобно воспользоваться *D*-защелкой. На рис. 30 показана схема *D*-защелки, состоящей из *SR*-защелки с входом разрешения и дополнительного инвертора, который формирует *S* и *R* сигналы из единственного входа *D*.

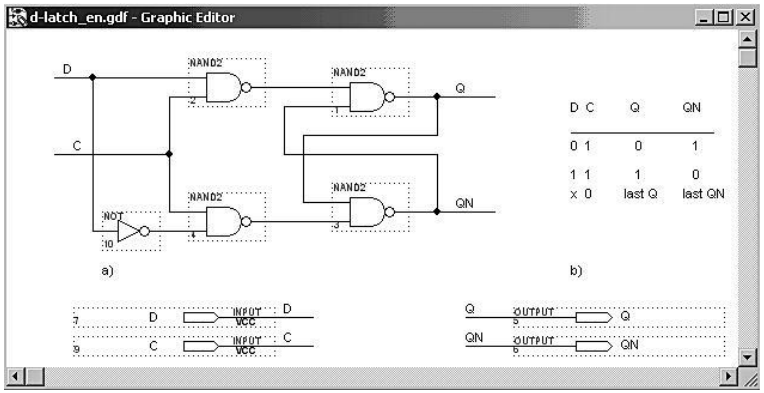


Рис. 30. *D*-защелка с входом разрешения *C*: *a*) принципиальная схема на вентилях И-НЕ; *b*) таблица переходов

где $j = 0, 1, \ldots, 7$), m_j — конституенты единицы переменных *C*, *B*, *A*, соответствующего номера.

Например, для номера $j = 0$ выражение (1) будет иметь вид:

$$Y_0 = \overline{C}\,\overline{B}\,\overline{A},$$

а для $j = 3$:

$$Y_3 = \overline{C}BA.$$

Таким образом, если на входы подана комбинация логических уровней 011, то из восьми выходов микросхемы (Y_0, Y_1, …, Y_7) на выходе Y_3 установится логический ноль ($Y_3 = 0$), а все остальные выходы будут иметь уровень логической единицы. Этот дешифратор имеет от-рицательный активный выходной сигнал. Дешифраторы могут иметь несколько стробирующих входов.

Рассмотрим дешифратор с $N = 2$ и положительными активными выходными сигналами. Дешифратор, как и любая комбинационная схема, задается таблицей истинности. На рис. 20 представлена схема дешифратора (*a*) и таблица истинности (*b*).

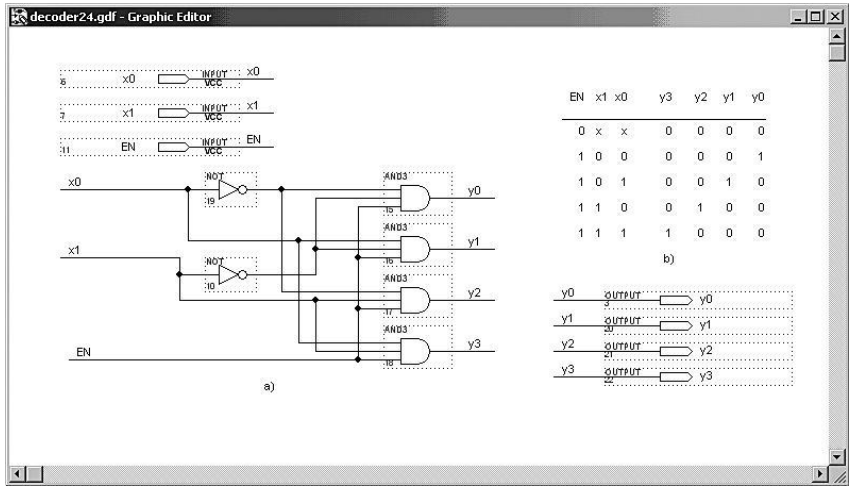


Рис. 20. Схема полного 2 × 4 дешифратора и таблица истинности

В таблице истинности полного дешифратора среди входных комбинаций фигурирует символ «безразличного состояния». Если одна или большее число входных комбинаций не влияют на значения вы-

ходных сигналов, то такие входные сигналы в данной комбинации отмечаются символом «*X*». Запишем алгебраические выражения для вы-ходных функций дешифратора:

$$Y0 = EN \times \overline{x1} \times \overline{x0}; \qquad Y1 = EN \times \overline{x1} \times x0;$$

$$Y2 = EN \times x1 \times \overline{x0}; \qquad Y3 = EN \times x1 \times x0.$$

Построим временн|ые диаграммы работы дешифратора (рис. 21). При активном сигнале *EN* (*EN*□=□1) на выходах дешиф-ратора появляются активные положительные сигналы, причем каж-дый соответствует своей кодовой комбинации входных сигналов. При обратном разрешающем сигнале *EN* (*EN*□=□0) все выходы де-шифратора принимают нулевое значение, что соответствует табли-це истинности.

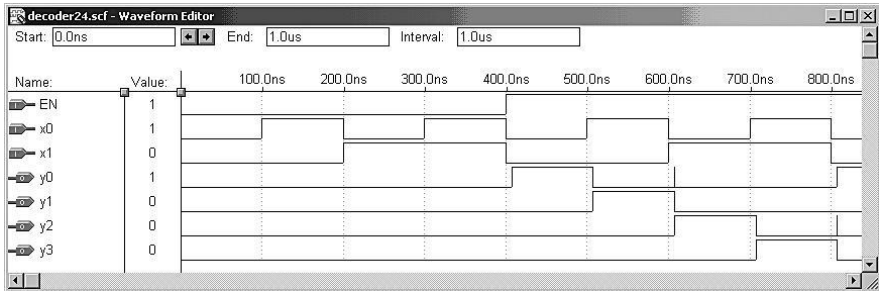


Рис. 21. Временная| диаграмма работы дешифратора 2 × 4

Дешифраторы изготавливаются в интегральном исполнении и имеют различное функциональное назначение. Например, дешиф-ратор КР514ИД2 является дешифратором семисегментного кода и ис-пользуется для преобразования двоичного или двоично-десятичного кода в код формирования символов на семисегментных индикаторах (рис. 22). Графическое изображение этой микросхемы представлено на рис. 23, а неполная таблица истинности дешифратора — табл. 3. Семисегментные индикаторы бывают с общим анодом и с общим като-дом. КР514ИД2 работает с индикаторами с общим анодом.

1.2 *SR*-защелка с входом разрешения

SR-защелки чувствительны к входным сигналам *S* и *R* в течение всего времени. Если мы хотим, чтобы выходные сигналы изменялись только в строго определенное время (во время действия разрешающе-го сигнала, например *C*), то схему защелки необходимо видоизменить. Реализуем защелку видоизмененной схемы не на элементах ИЛИ-НЕ (NOR2), а на элементах И-НЕ (NAND2). На рис. 28 представлена схе-ма, имеющая разрешающий сигнал *C*, с элементами стробирования и *SR*-защелка. Проект сохранен под именем sr-latch_en.gdf.

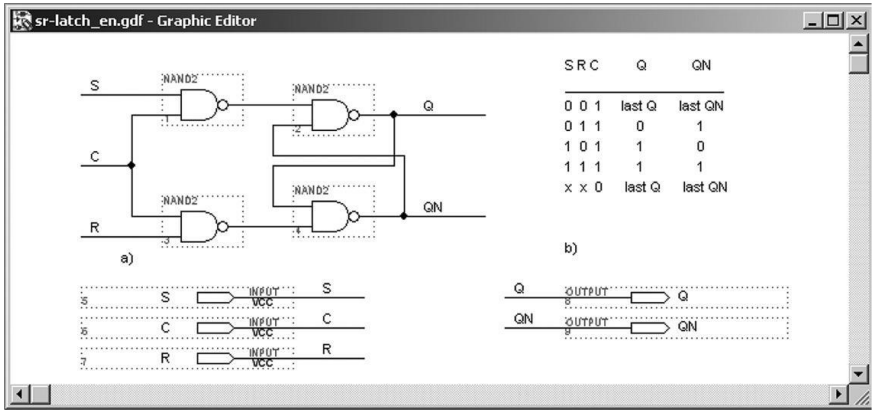


Рис. 28. *SR*-защелка с входом разрешения *C*: *a*) принципиальная схема на вентилях И-НЕ; *b*) таблица переходов

Как видно из таблицы переходов, описывающей работу схемы, при *C* = 1 данная схема ведет себя как *SR*-защелка, а при *C* = 0 она удер-живается в прежнем состоянии. На рис. 29 приведены временные| диа-граммы работы схемы при различных входных сигналах. Если оба сигнала равны 1 в момент, когда сигнал *C* переходит из 1 в 0 (отри-цательный фронт сигнала), то схема ведет себя подобно *SR*-защелке при одновременном переходе сигналов *S* и *R* на неактивный уровень. В этом случае следующее состояние непредсказуемо и выходная цепь может стать метастабильной (это состояние выделено черным цветом на экране монитора и находится правее вертикального маркера).

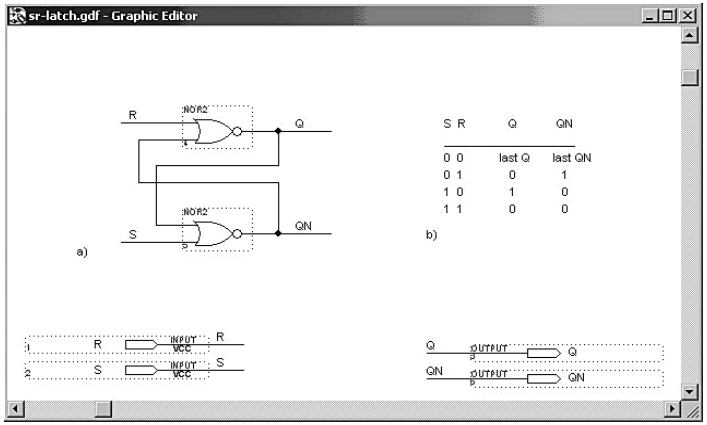


Рис. 26. *SR*-защелка: *a*) принципиальная схема на элементах NOR2; *b*) таблица переходов

Если оба входных сигнала *S* и *R* равны 0, то схема ведет себя аналогично элементу с двумя устойчивыми состояниями, то есть на выходах сохраняется последнее состояние (last *Q* и last *QN*). Меняя состояния *S* и *R* можно заставить схему переходить в требуемое состояние. Сигнал *S* (активное состояние 1) устанавливает (set) состояние, при котором выходной сигнал *Q* равен 1. Сигнал *R* сбрасывает (reset) или очищает схему, и выход *Q* становится равным 0. Сохраним проект под именем sr-latch.gdf и выпол-ним компиляцию. В окно редактора Waveform Editor загрузим входные и выходные сигналы схемы и выполним моделирование работы *SR*-защел-ки. На рис. 27 представлена временная| диаграмма работы *SR*-защелки.

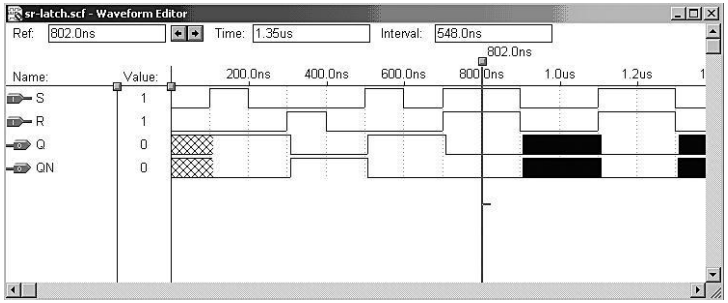


Рис. 27. Временная| диаграмма работы *SR*-защелки

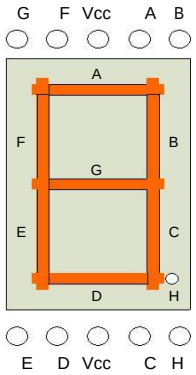


Рис. 22. Семисегментный индикатор

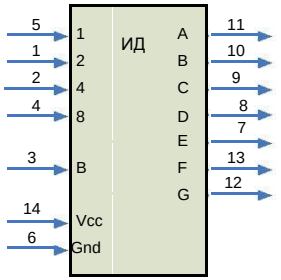


Рис. 23. Дешифратор КР514ИД2

Таблица 3. Таблица истинности дешифратора КР514ИД2

Входы					Выходы							
Выбор					(0)	(1)	(2)	(3)	(4)	(5)	(6)	Символ
8	4	2	1	*B*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	
X	*X*	*X*	*X*	0	1	1	1	1	1	1	1	
0	0	0	0	1	1	0	0	1	1	1	1	1
0	0	0	1	1	0	0	1	0	0	1	0	2
0	0	1	0	1	0	0	0	0	1	1	0	3
0	0	1	1	1	1	0	0	0	1	0	1	4
0	1	0	0	1	1	1	1	1	0	1	1	5
0	1	0	1	1	1	1	1	1	1	0	1	6
0	1	1	0	1	1	1	1	1	1	1	0	7
0	1	1	1	1	1	1	1	1	1	1	1	8

2 Задание для лабораторной работы № 2

Провести синтез дешифратора двоичного трехразрядного входного кода в семиразрядный выходной код для управления семисегментным индикатором. Функция определяется таблицей истинности (табл. 4).

Таблица 4. Таблица истинности функции

x_2	x_1	x_0	*A*	*B*	*C*	*D*	*E*	*F*	*G*	Символ
0	0	0	0	0	0	0	0	0	1	0
0	0	1	1	0	0	1	1	1	1	1
0	1	0	0	0	1	0	0	1	0	2
0	1	1	0	0	0	0	1	1	0	3
1	0	0	1	0	0	1	1	0	0	4
1	0	1	0	1	0	0	1	0	0	5
1	1	0	0	1	0	0	0	0	0	6
1	1	1	0	0	0	1	1	1	1	7

Выполнить следующие действия:

• записать СДНФ для каждого выхода дешифратора;

• минимизировать заданную функцию с помощью карт Карно для каждого выхода;

• в пакете MAX+plus II создать проект, из библиотечных элементов нарисовать схему дешифратора в графическом редакторе Graphic Editor;

• выполнить компиляцию проекта;

• в редакторе Waveform Editor выполнить временное| моделирова-ние схемы;

• загрузить конфигурационный файл в ПЛИС;

• с помощью входных переключателей стенда (см. инструкцию) задавать входные последовательности дешифратора и смотреть на ин-дикаторе отображение символов, заданных таблицей истинности.

3 Пример выполнения лабораторной работы № 2

Задание

Провести синтез дешифратора двоичного трехразрядного входно-го кода в семиразрядный выходной код для управления семисегмент-ным индикатором. Функция определяется таблицей истинности.

Лабораторная работа № 3

Последовательностные устройства: триггеры, регистры, счетчики

Цель работы: изучение работы последовательностных устройств — триггеров, регистров, счетчиков. Синтез цифровой схемы на базе синхронных счетчиков, разработка делителя частоты с переменным коэффициентом деления.

1 Краткие сведения из теории

Последовательностные цифровые устройства — это логические схемы, выходные сигналы которых не только определяются текущи-ми значениями входных сигналов, но и зависят от последовательности значений входных сигналов в прошлом. В большинстве последовательностных схем изменение выходов происходит в моменты времени, задаваемые внешним тактовым сигналом от независимого источника. Различают два типа элементарных последовательностных устройств — защелки и триггеры. В цифровой электронике принято триггером (flip-flop) называть последовательностную схему, в которой значения выходных сигналов изменяются только в моменты времени, задаваемые тактовым сигналом. Название «защелка» (latch) используется для последовательностной схемы, выходы которой чувствительны к измене-нию входных сигналов (но зависят не только от входных сигналов, но и от текущего состояния) непрерывно в течение всего времени [2].

1.1 *SR*-защелка

Рассмотрим простейшую, так называемую *SR*-защелку. На рис. 26 показана *SR*-защелка (set-reset latch) на вентилях ИЛИ-НЕ. У этой схе-мы два входа *S* и *R* и два выхода *Q* и *QN*. Сигнал *QN* представляет со-бой инверсию сигнала *Q*.

Требования к отчету

Отчет по лабораторной работе должен выполняться в отдельной тетради и содержать:

- Название лабораторной работы, ее цель, задачи.
- Вариант задания.
- СДНФ исходных функций.
- Минимизированные функции, синтезированную схему дешифратора.
- Временную| диаграмму работы дешифратора.

Вопросы и задания для самопроверки

1. К какому типу цифровых схем относятся дешифраторы?
2. Дайте определение полного дешифратора.
3. Дайте определение логической функции СДНФ.
4. Напишите функциональную зависимость *i*-го выхода полного трехразрядного дешифратора от его входов.
5. Сформулируйте принцип работы селектора входного кода.
6. Разработайте схему полного дешифратора 4 × 16.

1. Напишем СДНФ для функции каждого выхода.

$Y_0 = A = \bar{x}_2\,\bar{x}_1\,x_0 + x_2\,\bar{x}_1\,\bar{x}_0$;

$Y_1 = B = x_2\,\bar{x}_1\,x_0 + x_2\,x_1\,\bar{x}_0$;

$Y_2 = C = \bar{x}_2\,x_1\,\bar{x}_0$;

$Y_3 = D = \bar{x}_2\,\bar{x}_1\,x_0 + x_2\,\bar{x}_1\,\bar{x}_0 + x_2\,x_1\,x_0$;

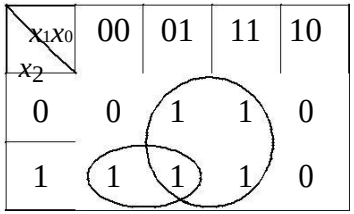
$Y_4 = E = \bar{x}_2\,\bar{x}_1\,x_0 + \bar{x}_2\,x_1\,x_0 + x_2\,\bar{x}_1\,\bar{x}_0 + x_2\,\bar{x}_1\,x_0 + x_2\,x_1\,x_0$;

$Y_5 = F = \bar{x}_2\,\bar{x}_1\,x_0 + \bar{x}_2\,x_1\,\bar{x}_0 + \bar{x}_2\,x_1\,x_0 + x_2\,x_1\,x_0$;

$Y_6 = G = \bar{x}_2\,\bar{x}_1\,\bar{x}_0 + \bar{x}_2\,\bar{x}_1\,x_0 + x_2\,x_1\,x_0$.

2. Минимизируем получившиеся функции выходов с помощью карт Карно. Функции выходов *A*, *B*, *C*, *D* имеют уже минимальную форму, поэтому построим карты Карно для выходов *E*, *F* и *G*.

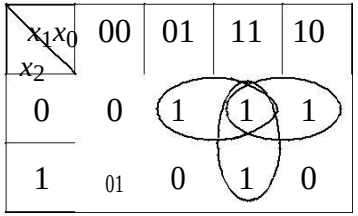
Для функции выходов *E*:



Склеивая соответствующие наборы, получим:

$E \quad x_0\;\; x_2\,\bar{x}_1$.

Для функции выходов *F*:



После склейки получим:

$F \quad x_2\,x_1\;\; x_2\,x_0\;\; x_1\,x_0$.

Для функции выходов *G*:

x_1x_0 / x_2	00	01	11	10
0	1	1	0	0
1	01	0	1	0

Склеивая наборы, получим:

$G = x_2 x_1 x_2 x_1 x_0$.

Таким образом получили минимальные функции выходов *A*, *B*, *C*, *D*, *E*, *F*, *G* дешифратора:

$A = x_2 x_1 x_0 + x_2 x_1 x_0$;

$B = x_2 x_1 x_0 + x_2 x_1 x_0$;

$C = x_2 x_1 x_0$;

$D = x_2 x_1 x_0 + x_2 x_1 x_0 + x_2 x_1 x_0$;

$E = x_0 + x_2 x_1$;

$F = x_2 x_1 + x_2 x_0 x_1 x_0$;

$G = x_2 x_1 + x_2 x_1 x_0$.

Анализируя получившиеся выражения, мы видим, что в разных функциях выходов есть однородные слагаемые. При составлении схемы это необходимо учесть.

3. Создадим проект lab_work2.gdf в пакете Max+plus II и нарисуем схему (рис. 24). После ввода схемы, выполним компиляцию.

4. Выполним временное| моделирование в редакторе Waveform Editor (рис. 25).

5. Загрузим конфигурационный файл lab_work2.pof в ПЛИС учеб-ного стенда.

6. С помощью входных переключателей стенда (см. инструкцию) зададим входные коды разработанного дешифратора и проверим соот-ветствие символов, отображаемых на индикаторе, заданным в таблице истинности.

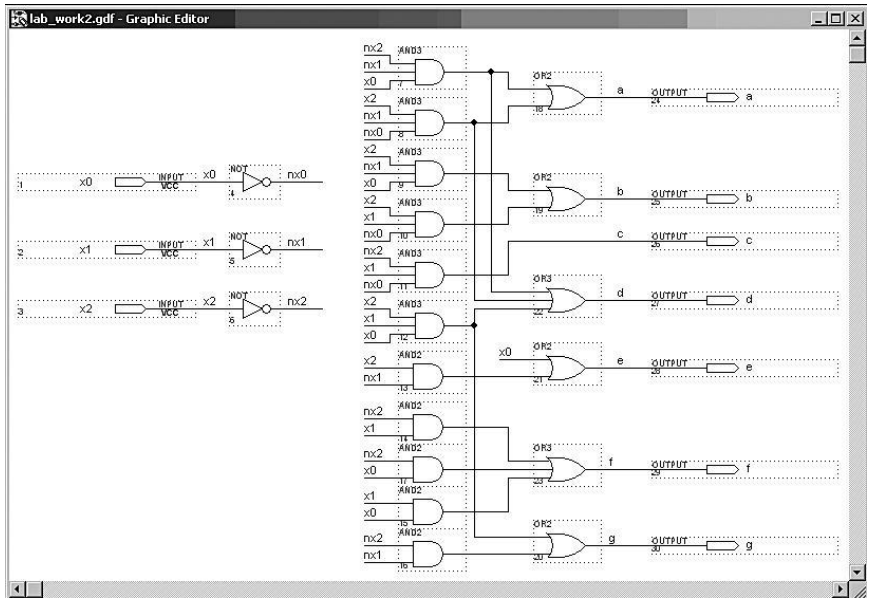


Рис. 24. Схема семисегментного дешифратора

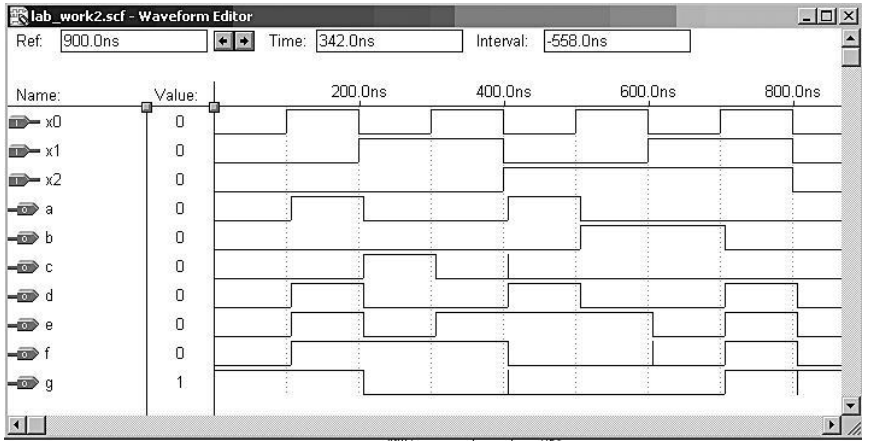


Рис. 25. Временная| диаграмма работы дешифратора