МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное
учреждение высшего образования
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ РАДИОЭЛЕКТРОНИКИ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Кафедра «Электронная техника»

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ ИНТЕГРАЛЬНЫЕ СХЕМЫ.
Лабораторный практикум

Учебно-методическое пособие
по дисциплине для обучающихся по направлениям подготовки
11.03.04 Электроника и наноэлектроника,
11.03.03 Конструирование и технология электронных средств

Севастополь
СевГУ
2020

УДК 621.382(06)
П78

Р е ц е н з е н т:
Ю.П. Михайлюк - заведующий кафедрой «Электронная техника»,
канд. техн. наук, доцент

Ответственный за выпуск:
Ю.П. Михайлюк - заведующий кафедрой «Электронная техника»,
канд. техн. наук, доцент

Составители: Д.В. Начаров, И.Б. Широков, И.В. Скорик

П78 Программируемые логические интегральные схемы. Лабораторный практикум : учебно-методическое пособие по дисциплине для обучающихся по направлениям подготовки 11.03.04 Электроника и наноэлектроника, 11.03.03 Конструирование и технология электронных средств / Сост. Д.В. Начаров, И.Б. Широков, И.В. Скорик ; Министерство науки и высшего образования Российской Федерации, Севастопольский государственный университет, Институт радиоэлектроники и информационной безопасности. – Севастополь: СевГУ, 2020. – 30 с. – Текст : электронный.

Цель пособия: оказание методической помощи студентам при выполнении лабораторного практикума по дисциплине «Программируемые логические интегральные схемы», подготовке, оформлении и защиты отчетов по лабораторным работам.

УДК 621.382(06)

Учебно-методическое пособие утверждено на заседании кафедры «Электронная техника», протокол № 8 от 17 марта 2020 г.

Учебно-методическое пособие одобрено на заседании методической комиссии Института радиоэлектроники и информационной безопасности, протокол № 11 от 29 мая 2020 г.

СОДЕРЖАНИЕ

Введение 4
1. Лабораторная работа № 1 5
2. Лабораторная работа № 2. 9
3. Лабораторная работа № 3. 15
4. Лабораторная работа № 4. 18
Приложение А. Инструкция по использованию САПР Quartus II 23
Библиографический список 30

ВВЕДЕНИЕ

Дисциплина «Программируемые логические интегральные схемы» (ПЛИС) относится к вариативной части учебного плана по направлениям подготовки высшего образования направлений 11.03.03 Конструирование и технология электронных средств и 11.03.04 Электроника и наноэлектроника.

Целью учебно-методического пособия является методической помощи студентам при выполнении лабораторного практикума по дисциплине «ПЛИС», подготовке, оформлении и защиты отчетов по лабораторным работам.

В учебно-методическое пособие включены пять лабораторных работ.

Лабораторная работа № 1 посвящена изучению поведенческого стиля описания модулей при проектировании цифровых систем на базе ПЛИС на примере проектирования цифрового генератора сигналов.

Лабораторная работа № 2 посвящена изучению структурного и поведенческого стилей описания модулей при проектировании цифровых систем на базе ПЛИС на примере проектирования двоичного сумматора.

Лабораторная работа № 3 посвящена изучению принципов проектирования регистровых цифровых устройств на базе ПЛИС на примере проектирования таймера реакции человека.

Лабораторная работа № 4 посвящена изучению принципов проектирования модулей памяти при реализации цифровых систем на базе ПЛИС.

Отчеты по лабораторным работам должны иметь титульный лист и содержать следующие разделы:

— цель работы;

— Verilog-описания разработанных цифровых устройств и модулей тестового окружения, временные диаграммы работы разработанных устройств;

— синтезированные схемы разработанных устройств;

— результаты выполнения индивидуальных заданий;

— выводы.

ЛАБОРАТОРНАЯ РАБОТА № 1
ИЗУЧЕНИЕ ГЕНЕРАТОРА СИГНАЛОВ

1.1. Цель работы:

— на примере цифрового генератора сигналов изучить применение поведенческого описания модулей при проектировании цифровых систем на базе ПЛИС.

1.2. Теоретические сведения

Широтно-импульсная модуляция (ШИМ, также PWM — Pulse-Width Modulation) является эффективным способом цифрового управления силовыми системами. При этом регулирование мощности осуществляется путем управления отношением времени, в течение которого сигнал включен, ко времени, в течение которого он выключен. Такой способ управления обладает как минимум двумя достоинствами: он удобен для реализации в цифровой системе, и является энергоэффективным, поскольку в ключевом режиме работы управляющий элемент потребляет минимальную мощность (в закрытом состоянии ток через него стремится к нулю, а в открытом стремится к нулю падение напряжения, поскольку сопротивление переключающих элементов стремятся уменьшить).

Модуль ШИМ разрабатывается следующим образом. Входной сигнал *data* задает число тактов, в течение которых следует удерживать высокий логический уровень на выходе модуля. Сигнал *cnt* внутреннего счетчика циклически перебирает состояния от 0 до максимального значения, которое может быть подано в качестве входного. Допустим, что внутренний счетчик является 8-разрядным (т.е. полный цикл счета содержит 256 тактов). Тогда при подаче на вход *data* числа 100 на выходе такого модуля будет логическая единица в течение 100 тактов, а в течение остальных 156 — логический ноль. Увеличивая значение числа, поданного на вход *data*, можно увеличивать отношение времени включения выходного сигнала к общему времени цикла счета.

Поведенческое описание 8-разрядного ШИМ-модулятора на языке Verilog и временные диаграммы его работы приведены на рис.1.1, 1.2.

Период следования T_{pwm} импульсов ШИМ определяется разрядностью внутреннего счетчика и составляет

$$T_{pwm} = \frac{T_{clk}}{N},$$

где T_{clk} — период тактового сигнала внутреннего счетчика;

N — модуль счета внутреннего счетчика.

Широтно-импульсная модуляция позволяет осуществлять имитацию аналогового напряжения в полностью цифровых системах. При этом ШИМ-модулятор выполняет роль цифро-аналогового преобразователя (ЦАП).

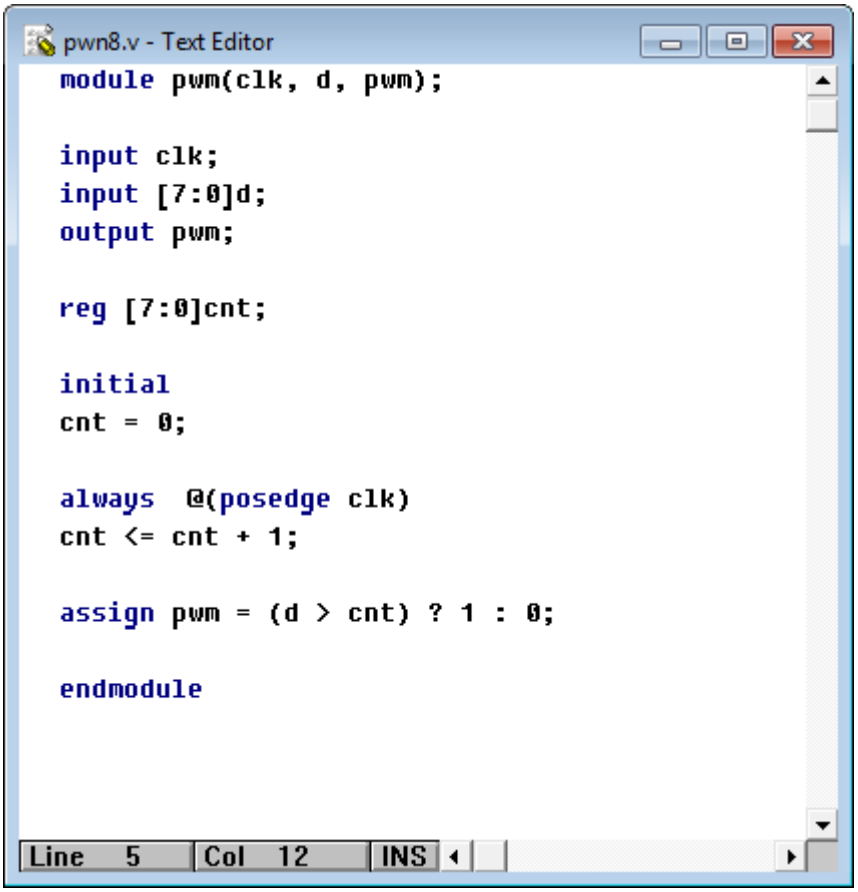

Рис.1.1 — Поведенческое описание ШИМ-модулятора

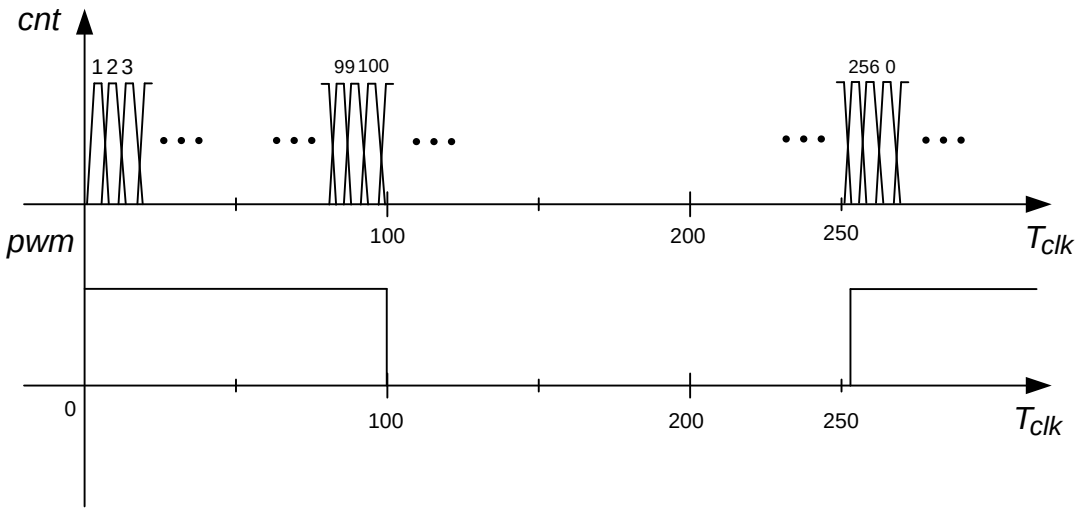

Рис.1.2 — Временные диаграммы работы ШИМ-модулятора при *data* = 100

Одним из возможных применений ШИМ является цифровой генератор сигналов произвольной формы. Принцип работы цифрового генераторов сигналов на базе ШИМ-модулятора заключается в представлении отсчетов дискретного сигнала ШИМ-сигналом. При этом форма выходного сигнала задается величиной и порядком следования дискретных отсчетов.

Простейшим примером цифрового генератора сигналов является генератор пилообразного напряжения.

Структурная схема генератора пилообразного напряжения показана на рис.1.3.

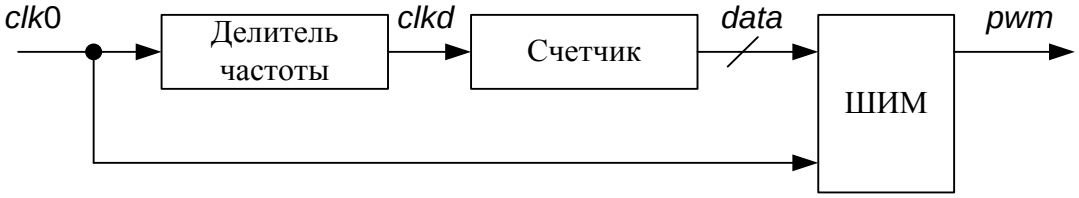

Рис.1.3 — Структурная схема генератора пилообразного напряжения

Последовательность счетных состояний на выходе счетчика используется в качестве отсчетов пилообразного напряжения, подаваемых на вход *data* ШИМ-модулятора. Поскольку период следования импульсов ШИМ отличается от частоты сигнала тактового генератора *clk*0 в N раз, для формирования сигнала *clkd* в цепи синхронизации счетчика присутствует делитель частоты с соответствующим коэффициентом деления N.

1.3 Порядок выполнения работы

Необходимо создать модуль генератора пилообразного напряжения, выполнить правильности его работы путем моделирования и экспериментальной проверки путем загрузки проекта в макет.

Модуль генератора пилообразного напряжения выполняется в виде модуля верхнего уровня иерархии, внутри которого происходит создание экземпляров модулей делителя частоты, счетчика, ШИМ-модулятора, соединительных проводников и портов ввода/вывода согласно структурной схеме на рис. 1.3.

Внутренняя реализация модулей счетчика и делителя частоты должны быть выполнены в виде поведенческого описания на языке Verilog.

Пример поведенческого описание ШИМ-модулятора показан на рис. 1.1.

Для выполнения моделирования в проекте также следует создать модуль тестового окружения (testbench).

При загрузке проекта в макет выходной сигнал генератора пилообразного напряжения следует выводить на один или группу светодиодов, например, на сегменты A-G индикатора 0. Для дублирования выходного одноразрядного сигнала генератора на несколько светодиодов в Verilog-описании следует применять оператор конкатенации (репликации).

Для того, чтобы изменение градаций яркости свечения было заметно визуально, необходимо добавить в проект дополнительный делитель напряжения для сигнала *clk*0, коэффициент деления которого следует выбрать таким, чтобы периода пилообразного напряжения составлял не менее 1 с при частоте тактового генератора макета, равной 1кГц.

1.4. Индивидуальные задания

Значения параметров генератора пилообразного напряжения при моделировании:

— разрядности ШИМ модулятора и счетчика — $(M \,\%\, 8)$,

где M — номер студента в списке группы;

% — операция взятия остатка от деления нацело.

Значения параметров генератора пилообразного напряжения при моделировании:

— разрядности ШИМ модулятора и счетчика — 4.

1.5. Содержание отчета

Отчет по лабораторной работе должен содержать:

— цель работы;

— задание на лабораторную работу;

— описание и временные диаграммы работы модулей делителя частоты и ШИМ-модулятора;

— Verilog-описание модулей тестового окружения;

— временные диаграммы работы генератора сигналов;

— синтезированные схемы модулей делителя частоты, счетчика, ШИМ-модулятора и генератора сигналов, полученные с помощью инструмента RTL Viewer;

— подробные выводы.

ЛАБОРАТОРНАЯ РАБОТА № 2
ИЗУЧЕНИЕ АРИФМЕТИЧЕСКИХ СХЕМ

2.1. Цель работы:

— на примере двоичного сумматора изучить применение структурного и поведенческого стилей описания модулей при проектировании цифровых систем на базе ПЛИС.

2.2. Теоретические сведения

Арифметические схемы являются основным функциональным узлом любого вычислительного устройства. Компьютеры и цифровые схемы выполняют множество арифметических операций: сложение, вычитание, сравнение, сдвиги, умножение и деление. В лабораторной работе на примере двоичного сумматора с последовательным переносом изучаются принципы построения цифровых арифметических схем, применяемых при проектировании цифровых систем на базе ПЛИС.

Сложение — одна из самых распространенных операций в цифровых системах. Базовой схемой сложения является полусумматор (англ. *half-adder*), позволяющий производить сложение двух 1-разрядных двоичных чисел.

Входными сигналами полусумматора являются 1-разрядные двоичные числа *a* и *b*, а выходными — бит суммы *s* и бит переноса *c* (таблица 2.1).

Таблица 2.1 — Таблица истинности полусумматора

a	*b*	*s*	*c*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Выходной сигнал *s* представляет собой сумму чисел *a* и *b*. Если $a = b = 1$, то сигнал *s* должен быть равным 2. Однако, поскольку число 2 нельзя представить с помощью одного двоичного разряда, в этом случае результат сложения указывается вместе с битом переноса единицы в следующий разряд. Таким образом при $a = b = 1$ выходной сигнал переноса *c* равен 1.

По таблице 2.1 могут быть записаны выражения, описывающие работу полусумматора,

$$s = \bar{a}b + a\bar{b} = a \oplus b;$$

$$c = ab,$$

где символом «⊕» обозначается операция Исключающее ИЛИ.

Таким образом, схема полусумматора может быть построена с помощью вентиля Исключающее ИЛИ и вентиля И (рисунок 2.1, а). Условное обозначение полусумматора показано на рис.2.1, б.

а) б)

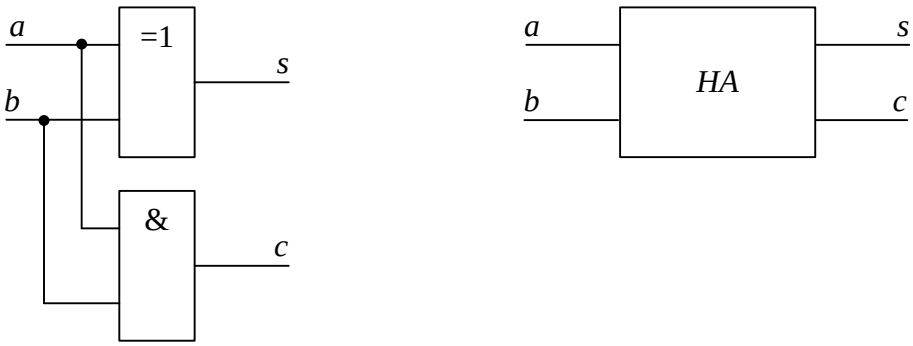

Рис.2.1 — Схема (а) и условное обозначение (б) полусумматора

При сложении многоразрядных чисел выход переноса текущего разряда соединяется со входом переноса следующего разряда. Схема сложения, содержащая вход переноса c_{in} и выход переноса c_{out}, называется полным сумматором (*full-adder*). Таблица истинности полного сумматора содержит дополнительный вход c_{in} (таблица 2.2).

Таблица 2.2 — Таблица истинности полного сумматора

c_{in}	a	b	s	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

По таблице 2.2 могут быть записаны выражения, описывающие работу полного сумматора,

$$a = (a \oplus b)\overline{c_{in}} + \overline{(a \oplus b)}c_{in} = (a \oplus b) \oplus c_{in};$$

$$c_{out} = \overline{c_{in}}ab + c_{in}\overline{a}b + c_{in}a\overline{b} + c_{in}ab = ab + c_{in}(a \oplus b).$$

Полный сумматор может быть получен за счет объединения двух полусумматоров (рис.2.2, а). Условное обозначение полного сумматора показано на рис.2.2, б.

Самый простой способ реализации N-разрядного сумматора – это объединение в цепь N полных сумматоров. Выход c_{out} i-го разряда соединяется с входом c_{in} $(i+1)$-го разряда. Такая схема называется сумматором с последовательным переносом (*ripple-carry adder*).

На рис.2.3 показана схема 4-разрядного сумматором с последовательным переносом, на входы которого подаются числа $A = \{A_3, A_2, A_1, A_0\}$ и

$B = \{B_3, B_2, B_1, B_0\}$ и одноразрядный сигнал c_{in}. На выходах 4-разрядного сумматора формируется число $S = \{S_3, S_2, S_1, S_0\}$ и одноразрядный сигнал c_{out}. При сложении младших разрядов A_0 и B_0 входной бит переноса отсутствует, поэтому сигнал $c_{in} = 0$ (вход c_{in} соединен с «землей»).

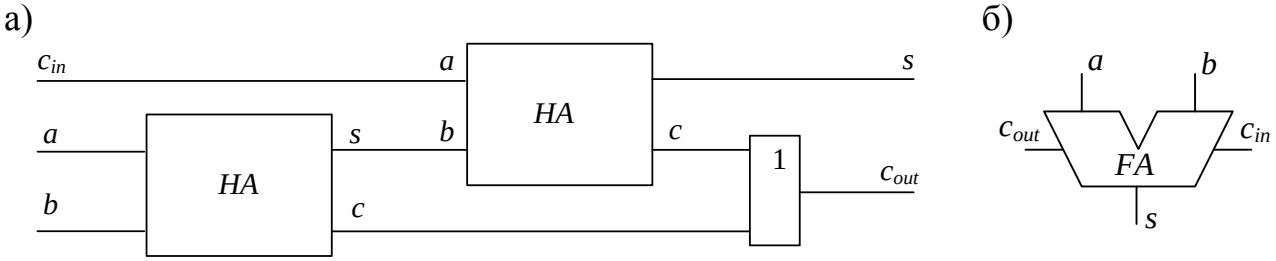

Рис. 2.2 — Схема (а) и условное обозначение (б) полного сумматора

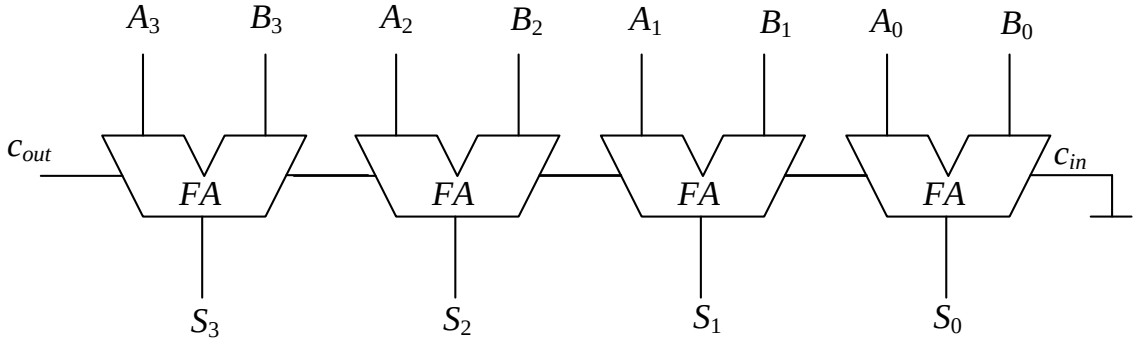

Рис. 2.3 — 4-разрядный сумматор с последовательным переносом

На рис. 2.4 показано условное обозначение 4-разрядного сумматора с последовательным переносом.

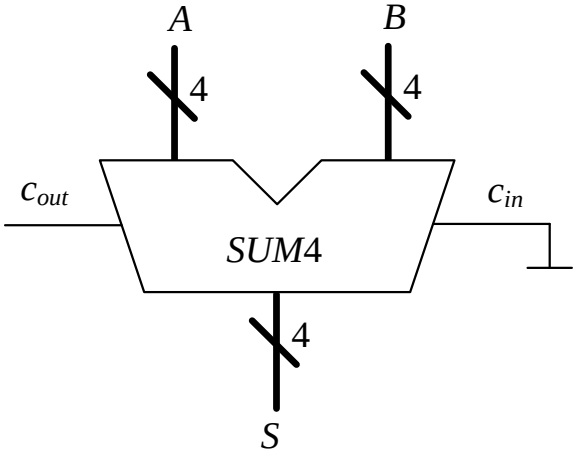

Рис. 2.4 — Условное обозначение 4-разрядного сумматора

Функционально идентичный модуль 4-разрядного сумматора может быть описан поведенчески с использованием арифметических операций в процедурном блоке `always`.

2.3. Порядок выполнения работы

В ходе выполнения работы необходимо:

— выполнить структурное описание модуля сумматора с последовательным переносом, выполнить моделирование его работы, а также сохранить и проанализировать синтезированную схемы с помощью инструмента RTL Viewer; разрядность сумматора задать с помощью параметра;

— выполнить поведенческое описание модуля сумматора с последовательным переносом, выполнить моделирование его работы, а также сохранить и проанализировать синтезированную схемы с помощью инструмента RTL Viewer; разрядность сумматора задать с помощью параметра;

— выполнить сравнительный анализ синтезированных схем 4-разрядного сумматора для структурного и поведенческого описаний;

— реализовать схему сложения двух 4-разрядных двоичных чисел с выводом входных операндов и суммы на семисегментные индикаторы.

Проект схемы сложения выполняется в виде файла верхнего уровня иерархии, содержащего порты ввода/вывода, соединительные проводники и модули сумматора и дешифратора семисегментного индикатора в соответствии со структурной схемой (рис.2.5).

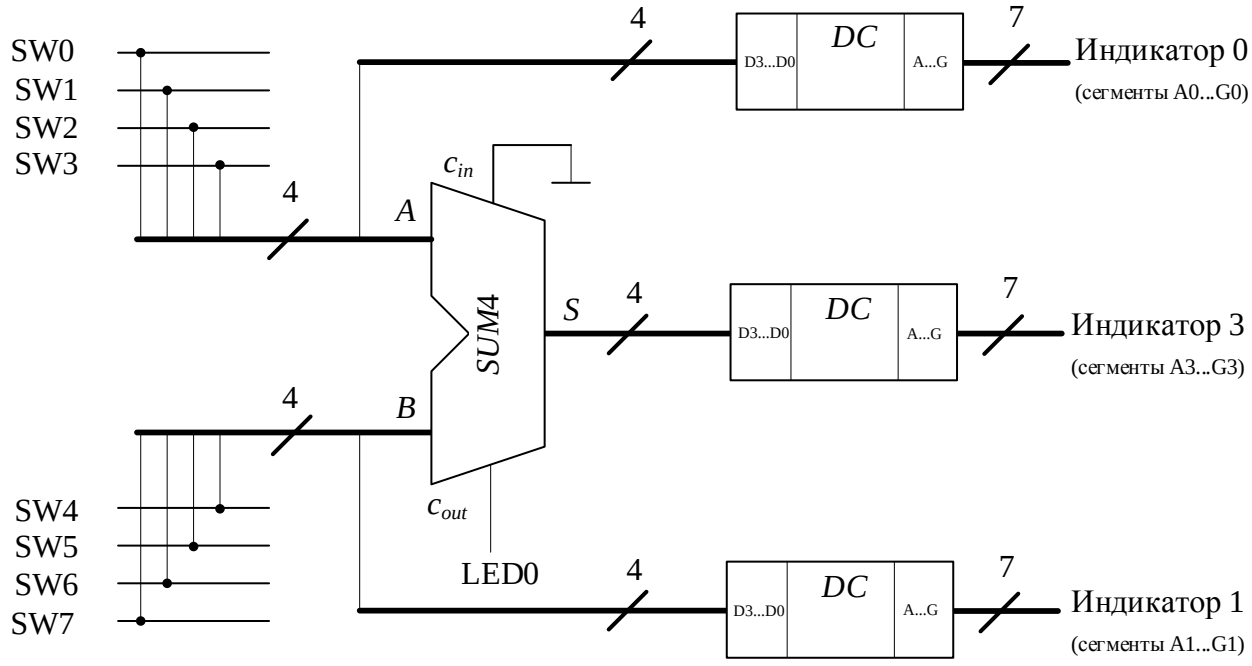

Рис. 2.5 — Схема сложения двух 4-разрядных двоичных чисел

Числа *A* и *B* в двоичном представлении задаются положением переключателей SW0…SW7 и подаются параллельно на входы 4-разрядного сумматора с последовательным переносом *SUM*4 и входы двух дешифраторов семисегментного индикатора, выходы которых связаны с индикаторами 0 и 1. Результат сложения *S* подается на дешифратор семисегментного индикатора и выводится на индикатор 3. Сигнал выходного бита переноса c_{out} сумматора подается на светодиод LED0.

Пример структурного описания модулей полусумматора и полного сумматора показаны на рис. 2.6, 2.7.

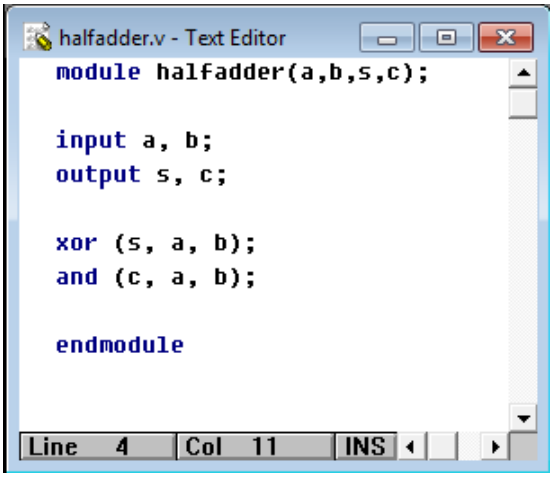

Рис. 2.6 — Пример структурного описания модуля полусумматора

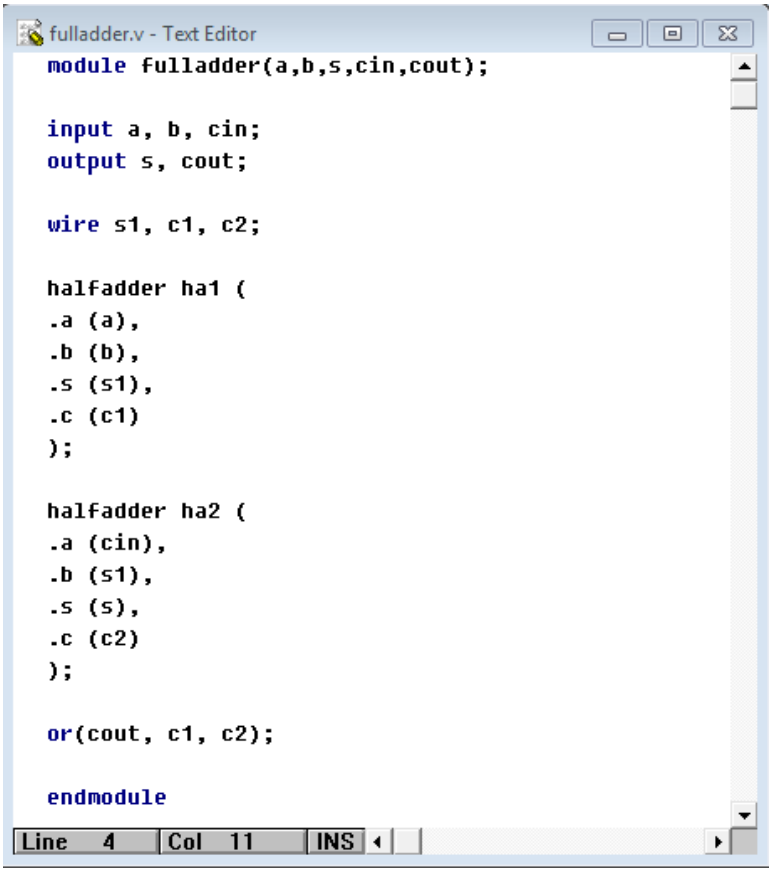

Рис. 2.7 — Пример структурного описания модуля полного сумматора

2.4. Индивидуальные задания

Значения разрядности сумматора с последовательным переносом при моделировании — $(2M \,\%\, 8)$, где M — номер студента в списке группы; % — операция взятия остатка от деления нацело.

2.5. Содержание отчета

Отчет по лабораторной работе должен содержать:

— цель работы;

— задание на лабораторную работу;

— описание работы и временные диаграммы работы модулей полусумматора, полного сумматора и сумматора с последовательным переносом;

— Verilog-описание модулей тестового окружения;

— синтезированные схемы 4-разрядного сумматора для структурного и поведенческого описаний, полученные с помощью инструмента RTL Viewer;

— подробные выводы.

ЛАБОРАТОРНАЯ РАБОТА № 3
ПРОЕКТИРОВАНИЕ РЕГИСТРОВЫХ УСТРОЙСТВ НА ПЛИС

3.1. Цель работы:

— на примере разработки таймера реакции изучить принципы проектирования регистровых цифровых устройств на базе ПЛИС.

3.2. Задание на лабораторную работу

Разработать таймер реакции человека, предполагающий измерение времени от момента включения светодиода до момента нажатия на кнопку PUSH.

На рис. 3.1 показана функциональная схема, поясняющая принцип работы таймера реакции. Модуль таймера реакции на рис.3.1 выделен штриховой линией.

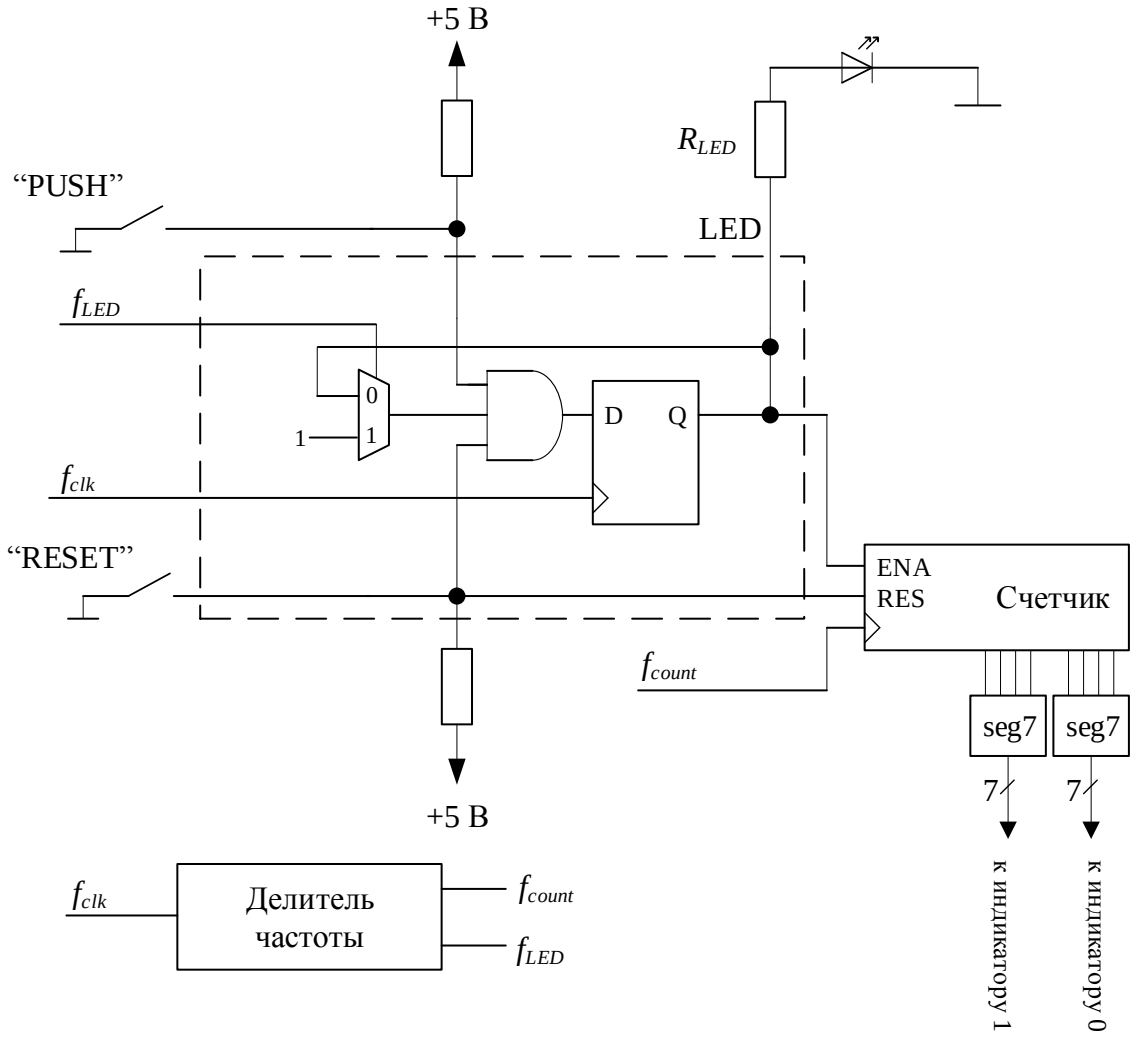

Рис. 3.1 — Функциональная схема проекта

В качестве опорного тактового сигнала используется сигнал тактового генератора макета с частотой $f_{clk} = 1000$ Гц. Для того чтобы изменения состояния

светодиода были заметны человеку, для управления светодиодом используется сигнал с частотой f_{LED}, полученный в результате деления частоты f_{clk}. При этом

$$f_{LED} = \frac{f_{clk}}{1000} = 1 \text{ Гц}.$$

Таким образом, изменение состояния светодиода производится не чаще, чем 1 раз в секунду.

Для измерения времени реакции используется двоично-десятичный Счетчик, тактируемый сигналом с частотой f_{count}. При этом

$$f_{count} = \frac{f_{clk}}{10} = 100 \text{ Гц}.$$

Таким образом, время реакции может быть измерено с точностью до сотых долей секунды. Помимо входа тактового сигнала Счетчик также имеет вход разрешения счета ENA и вход сброса RES.

Для индикации измеренного времени используются два семисегментных индикатора. Для преобразования двоичного кода с выхода Счетчика в код семисегментного индикатора используются дешифраторы **seg7**.

Формирование тактовых сигналов с частотами f_{LED} и f_{count} используется Делитель частоты. Поскольку коэффициенты деления частоты не является степенями 2, для реализации делителя следует применять счетчики с асинхронным сбросом (произвольным модулем счета). Один из возможных вариантов функциональной схемы модуля Делителя показана на рис.3.2.

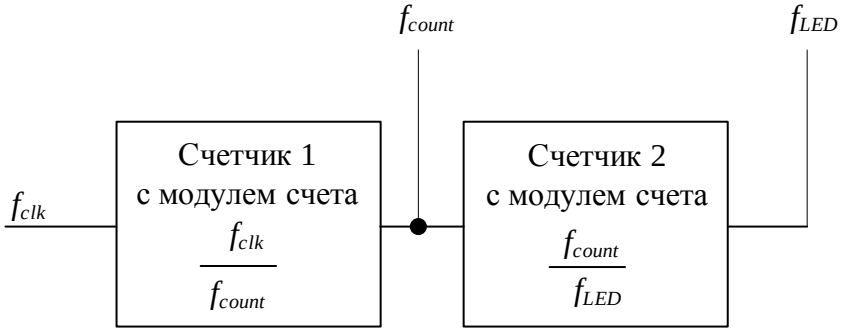

Рис. 3.2 — Функциональная схема модуля Делителя

Включение и выключение светодиода осуществляется установкой, соответственно, сигнала логической 1 или логического 0 на линии LED, подключенной к выходу Q D-триггера. На выход D-триггера через элемент 3-И поступают сигналы кнопок PUSH и RESET, а также сигнал с выхода мультиплексора. На входы мультиплексора подключен сигнал логической 1 и сигнал с линии LED. Для управления мультиплексором используется сигнал f_{LED}.

Установка D-триггера (включение светодиода) осуществляется высоким уровнем сигнала f_{LED} — при этом к выходу мультиплексора будет подключен сигнал логической 1. Сброс D-триггера (выключение светодиода) происходит либо по нажатию кнопки PUSH, либо по нажатию кнопки RESET.

Сигнал линии LED подается на вход разрешения счета ENA Счетчика. Низкий уровень линии LED останавливает счет. При этом на выходе Счетчика сохраняется текущее счетное состояние.

Verilog-описание модуля таймера реакции приведено на рис.3.3.

```
1  module reaction_v2(PUSH, fled, fclk, RESET, LED);
2  input PUSH, fled, fclk, RESET;
3  output reg LED;
4
5  always @(posedge fclk)
6  begin
7      if(fled)
8          LED <= 1;
9      else
10         LED <= LED & PUSH & RESET;
11 end
12 endmodule
```

Рис. 3.3 — Verilog-описание модуля таймера реакции

3.3. Порядок выполнения работы

Для реализации таймера реакции необходимо разработать Verilog-описания модулей Делителя, двоично-десятичного Счетчика и дешифратора.

Модуль Делителя строится на основе двух многоразрядных счетчиков с асинхронным сбросом (с произвольным модулем счета) и может быть описан на поведенческом уровне.

Модуль двоично-десятичного Счетчика должен иметь одноразрядные тактовый вход, входы разрешения счета и сброса, а также 4-разрядные выходы для формирования двоичного представления двух десятичных разрядов.

Модули двоично-десятичного Счетчика и дешифратора могут быть описаны на поведенческом уровне.

3.4. Содержание отчета

Отчет по лабораторной работе должен содержать:

— цель работы;

— задание на лабораторную работу;

— Verilog-описания, результаты моделирования и синтезированные схемы модулей Делителя, двоично-десятичного Счетчика, таймера реакции и дешифратора семисегментного индикатора;

— подробные выводы.

ЛАБОРАТОРНАЯ РАБОТА № 4
ПРОЕКТИРОВАНИЕ МОДУЛЕЙ ПАМЯТИ

4.1. Цель работы:

— изучить принципы проектирования модулей памяти при реализации цифровых систем на базе ПЛИС.

4.2. Теоретические сведения

Память является важным и часто используемым элементом современных цифровых устройств. Она представляет собой массив однотипных ячеек, хранящих число фиксированной разрядности.

По характеру выполняемых с ней операций память подразделяется на:

— постоянные запоминающие устройства (ПЗУ, также ROM — Read-Only Memory), которые хранят фиксированные данные без возможности их изменения;

— оперативные запоминающие устройства (ОЗУ, также RAM — Random Access Memory), допускающие изменение записанных данных.

Типы памяти также разделяют на энергозависимую (сохраняющую данные только при поданном питании) н энергонезависимую. Для ранних вариантов исполнения энергозависимая память являлась практически синонимом ПЗУ, поскольку техническая реализация таких микросхем подразумевала хранение данных в ячейках с пережигаемыми перемычками, ультрафиолетовым стиранием н т.п. Все эти принципы исполнения обеспечивали сохранность данных при отключении питания, но не позволяли изменять содержимое памяти без специального оборудования — например, память с ультрафиолетовым (УФ) стиранием требовала, как следует из ее названия, источника УФ излучения для стирания данных н специального программатора.

В настоящее время существуют устройства энергонезависимой памяти, которые допускают изменение содержимого без специального программатора. Например, Flash-память, память с электрическим стиранием, память FRAM, другие перспективные типы памяти. Часть из них требует отдельного цикла стирания, а часть позволяет произвольно перезаписывать данные, как для микросхем ОЗУ.

По способу организации интерфейса модули памяти подразделяются на модули с асинхронным или синхронным интерфейсом, а также с параллельным или последовательным доступом. Наиболее простой вариант — память с параллельным асинхронным доступом. Графическое изображение такого модуля показано на рисунке 4.1.

Рис.4.1 — Графическое изображение модуля ПЗУ
с асинхронным интерфейсом

Постоянное запоминающее устройство с асинхронным интерфейсом может быть описано с помощью оператора **`case`** (рис.4.2).

```
module rom(
    input [3:0] a,
    output [7:0] d
    );

reg [7:0] data;

always @(a)
case (a)
  0 : data <= 1;
  1 : data <= 3;
  2 : data <= 4;
  default : data <= 0;
endcase

assign d = data;
endmodule
```

Рис.4.2 — Verilog-описание модуля ПЗУ с асинхронным интерфейсом

При описании асинхронного ПЗУ с помощью оператора **case** используются строки вида **<addr> : <data>** — для каждого варианта адреса записывается то значение, которое хранится по этому адресу. Можно заметить, что при таком подходе сложно описать массивы памяти большого объема без применения средств автоматизации.

В ПЛИС модули ПЗУ небольшого объема обычно реализуются на таблицах соответствия программируемых ячеек. Таблица с 6 входами может хранить 64 бита, а с 4 — 16 бит. Необходимо иметь в виду, что из-за реализации в виде мелких блоков память на базе программируемых ячеек не может обеспечить высокие характеристики производительности.

Синхронный интерфейс памяти обеспечивает более высокую производительность, поэтому память такого вида используется чаще. Блочная память, размещаемая в FPGA, является памятью с синхронным интерфейсом. Отличием такого типа интерфейса является выполнение всех действий по фронту тактового сигнала.

Для памяти с возможностью чтения и записи (ОЗУ) используются следующие дополнительные сигналы:

— **din** — данные для записи;

— **we** — разрешение записи (Write Enable). Память такого типа работает следующим образом: если по фронту тактового сигнала активен сигнал **we**, то производится запись данных **din** в ячейку памяти с адресом **addr**. Иначе производится чтение из памяти, и на выходе **dout** появляется содержимое ячейки памяти с адресом **addr**. Пример описания на Verilog памяти с произвольным доступом и синхронным интерфейсом показан на рис. 4.3.

```
module ram(
 input clk,
 input [7:0] addr,
 input [15:0] din,
 input we,
 output reg [15:0] dout
 );

reg [15:0] ram_array [7:0];

always @(posedge clk)
    begin
    if (we) begin
      ram_array[addr] <= din;
    end
    dout = ram_array[addr];
    end

endmodule
```

Рис. 4.3 — Verilog-описание модуля ОЗУ с синхронным интерфейсом

Графическое изображение модуля ОЗУ с синхронным интерфейсом показано на рисунке 4.4.

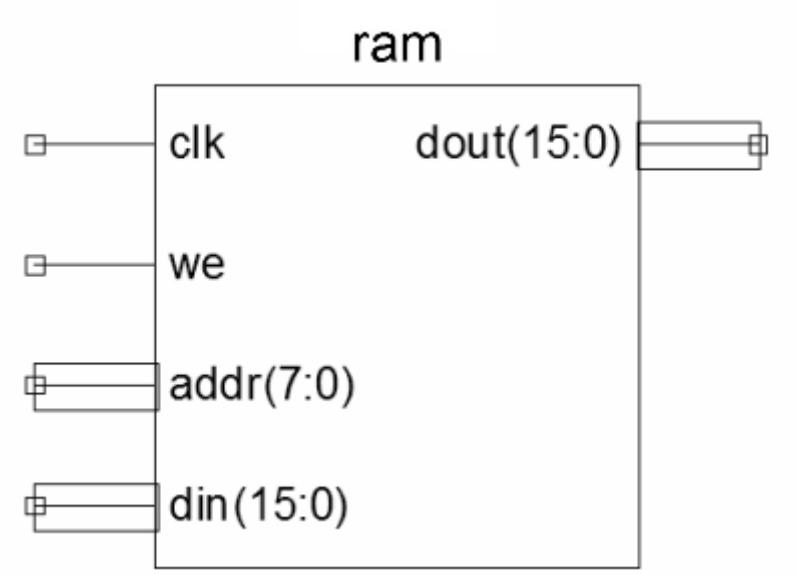

Рис. 4.4 — Графическое изображение модуля ОЗУ с синхронным интерфейсом

Из описания порядка работы модуля ОЗУ виден его недостаток, проявляющийся в том, что при записи в память невозможно одновременно читать ее содержимое. Например, при использовании блока памяти для хранения таблицы значений, непрерывно выдаваемых на цифро-аналоговый преобразователь, устройство отображения информации или на иное устройство, требующее непрерывного потока данных, возникнет проблема, связанная с тем, что для обновления содержимого памяти придется прервать процесс чтения записанных в нее данных.

От этого недостатка свободны многопортовые модули памяти, которые позволяют обращаться к одному н тому же массиву ячеек с помощью нескольких наборов линий **addr**, , **din**, **we**, и имеют соответствующее количество выходных шин **dout**.

Простейшим вариантом многопортовой памяти является двупортовая (dual-port memory), которая имеет достаточно много разновидностей. По функциональным возможностям второго порта двупортовая память подразделяется на simple dual-port, или pseudo dual-port («простая двупортовая», или «псевдо-двупортовая» память), и true dual-port («истинно двупортовая память»). Их отличием является то, что память true dual-port имеет два независимых и равноправных порта, по каждому из которых возможно проведение операций чтения и записи. У памяти simple dual-port один порт является универсальным (чтение н запись), а второй — только для чтения. Память такого типа вполне может быть использована в проектах, где требуется обеспечение непрерывного потока читаемых данных. В этом случае при необходимости перезаписи используется универсальный порт, а второй н используется для постоянного считывания. Графическое изображение simple dual-port памяти показано на рисунке 4.5.

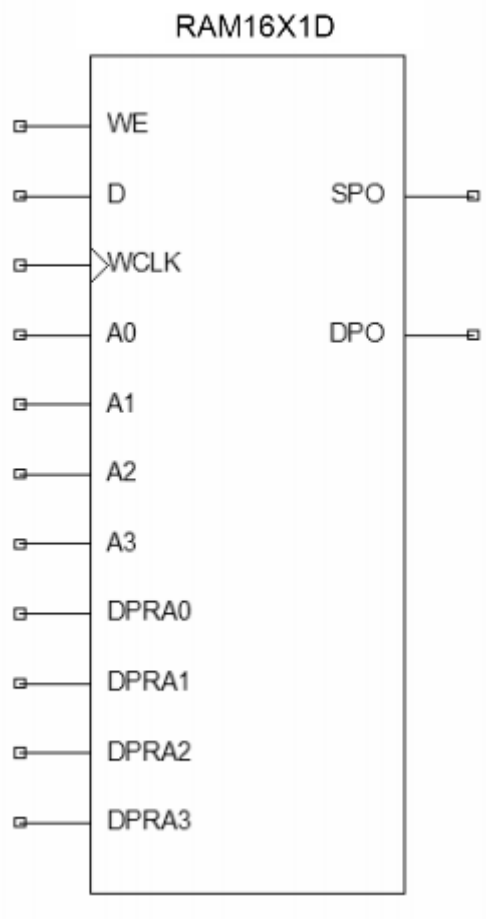

Рис. 4.5 — Двупортовая память в конфигурации simple dual-port

Таблицы соответствия программируемых ячеек FPGA представляют собой массивы статической памяти, хранящие таблицы истинности. Поэтому они могут быть использованы и в качестве модулей памяти, являясь при этом simple dual-port памятью. Такая память в терминологии FPGA называется также распределён-

ной (dislributed), поскольку распределена по программируемым ячейкам. Блочная память, размещаемая в FPGA, является true dual-port памятью.

4.3. Порядок выполнения работы

Разработать Verilog-описания модулей ПЗУ с асинхронным интерфейсом и однопортовой ОЗУ с синхронным интерфейсом. Модули должны быть параметризированы, при этом с использованием параметров следует задать разрядности шин адреса и данных.

Проверить правильность разработанных Verilog-описаний путем моделирования работы в программе ModelSim, а также запрограммировав лабораторный макет. При программировании макета включить в проект модуль дешифратора семисегментного индикатора.

Значения параметров при моделировании:

1) модуль ПЗУ:

— разрядность шины данных — (**4 + M%5**);

— разрядность шины адреса — (**2 + M%3**);

2) модуль ОЗУ:

— разрядность шины данных — **M**;

— разрядность шины адреса — (**2 + M%3**),

где **M** — номер студента в списке группы.

Значения параметров при программировании макета:

1) модуль ПЗУ:

— разрядность шины данных — 4;

— разрядность шины адреса — 3;

2) модуль ОЗУ:

— разрядность шины данных — 4;

— разрядность шины адреса — 3.

4.4. Содержание отчета

Отчет по лабораторной работе должен содержать:

— цель работы;

— задание на лабораторную работу;

— Verilog-описания, результаты моделирования и синтезированные схемы модулей ПЗУ и ОЗУ;

— подробные выводы.

ПРИЛОЖЕНИЕ А
ИНСТРУКЦИЯ ПО ИСПОЛЬЗОВАНИЮ САПР QUARTUS II

А.1. Создание проекта

После запуска Quartus II для создания нового проекта необходимо запустить мастер **New Project Wizard…** из меню **File**. В открывшемся окне нажать кнопку **Next**, далее указать путь к рабочей папке проекта, название проекта и имя файла верхнего уровня иерархии (имя основного файла) проекта о файлах проекта (рис.А.1). При этом следует использовать только латинские символы для именования файлов и папок. Имя основного файла и название проекта должны совпадать. После этого следует нажать кнопку **Next**.

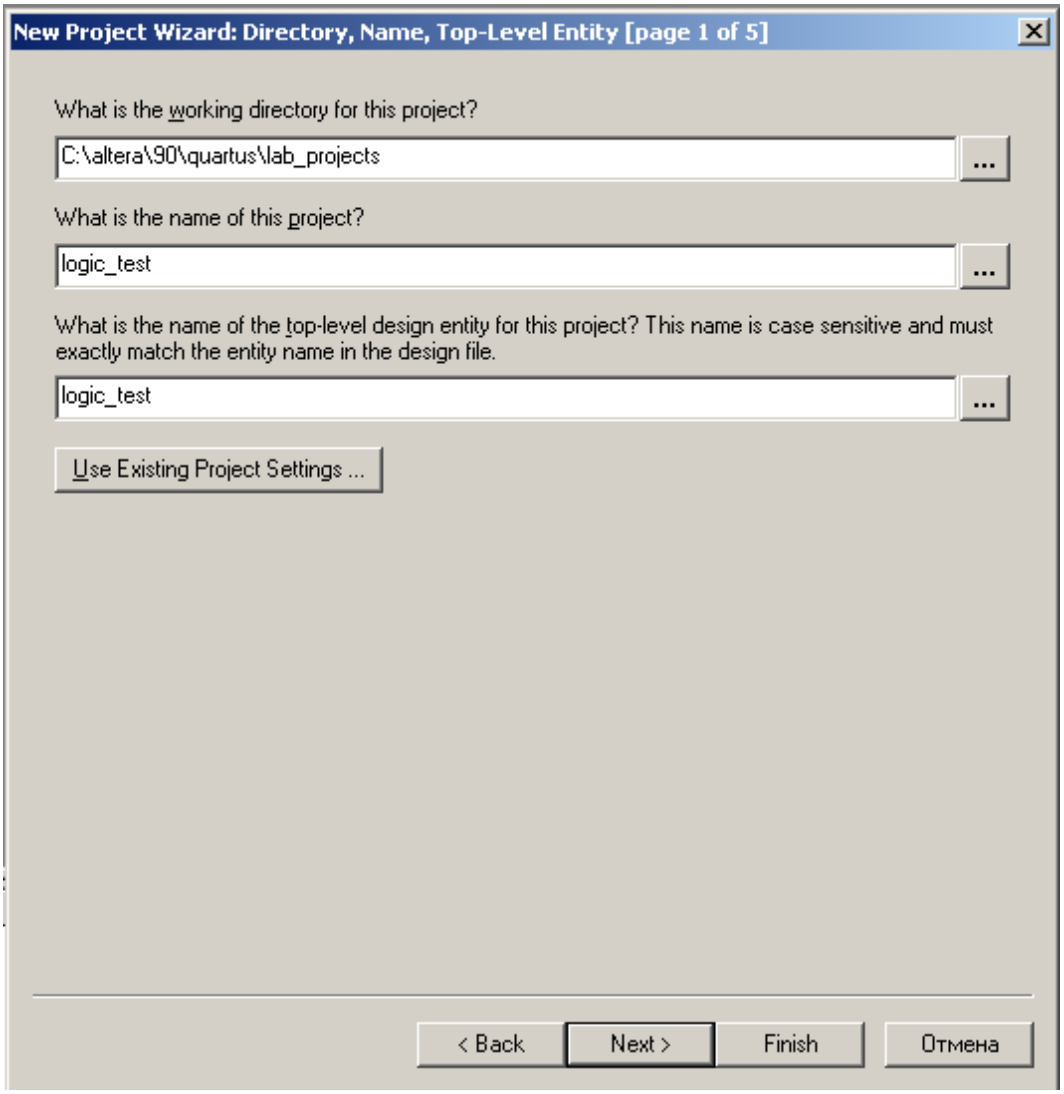

Рис. А.1

В открывшемся окне (рис.А.2) в создаваемый проект можно добавить существующие файлы. Для этого следует нажать кнопку ..., расположенную рядом с полем **File name**. Кнопка **Add All…** позволяет добавить в проект все файлы, на-

ходящиеся в выбранной папке проекта. При создании проекта впервые этот шаг можно пропустить, нажав кнопку **Next.**

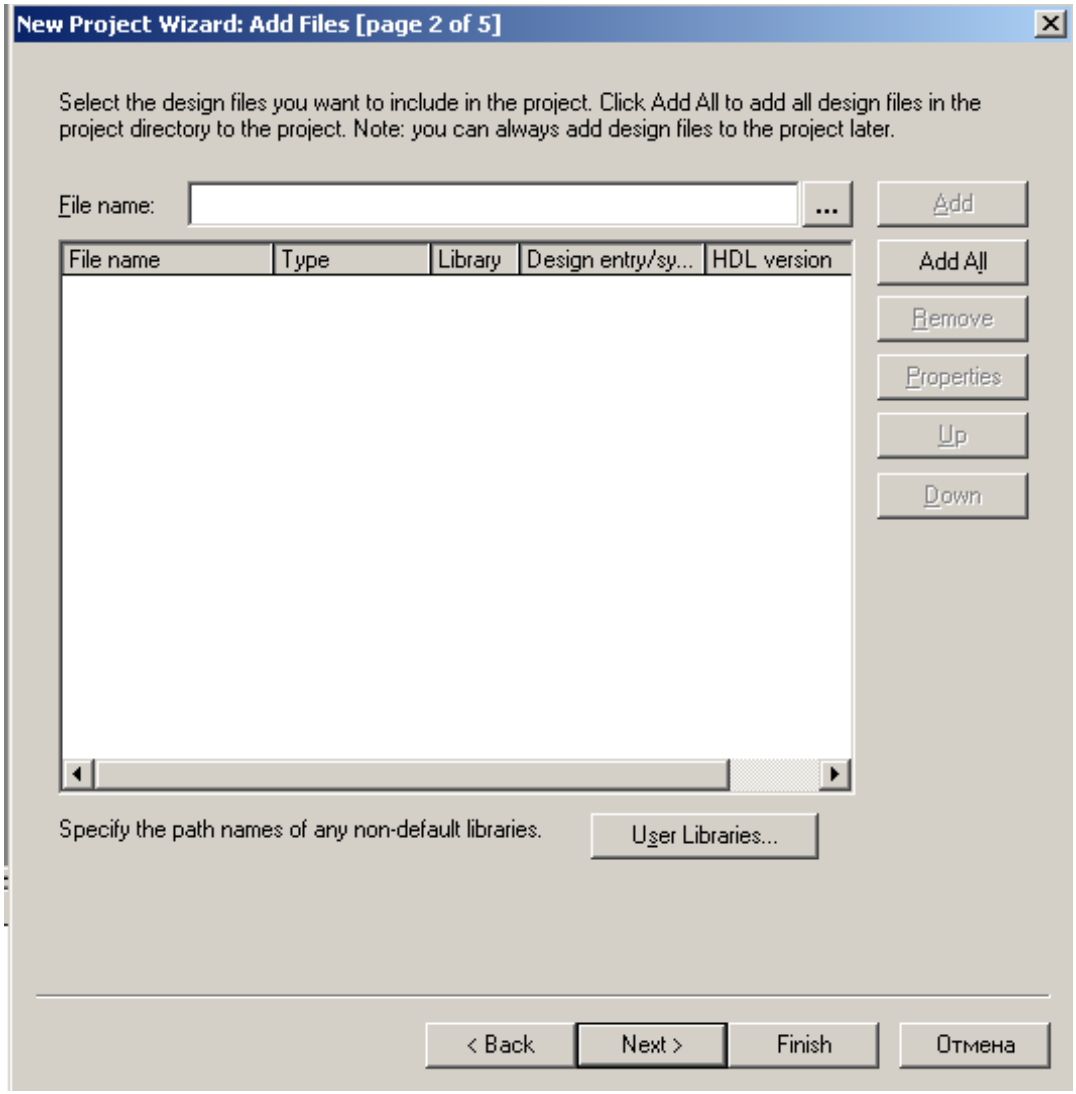

Рис. А.2

Следующее окно мастера (рис.А.3) позволяет выбрать семейство и наименование микросхемы ПЛИС, в которую будет загружен проект. В выпадающем списке **Family** следует выбрать семейство микросхем *MAX7000S*, а в поле **Available Devices** выбрать микросхему *EPM7128SLC84-15*. После этого нажать кнопку **Finish**.

Чтобы создать новый файл проекта следует выбрать пункт **New** из меню **File**. В случае создания текстового описания модуля на языке Verilog следует выбрать тип *Verilog HDL File* (рис.А.4) из раздела *Design Files*. Нажмите кнопку **OK**, после чего открывается окно текстового редактора. Сохраните файл схемы с расширением *.v*, выбрав пункт **Save As** из меню **File**.

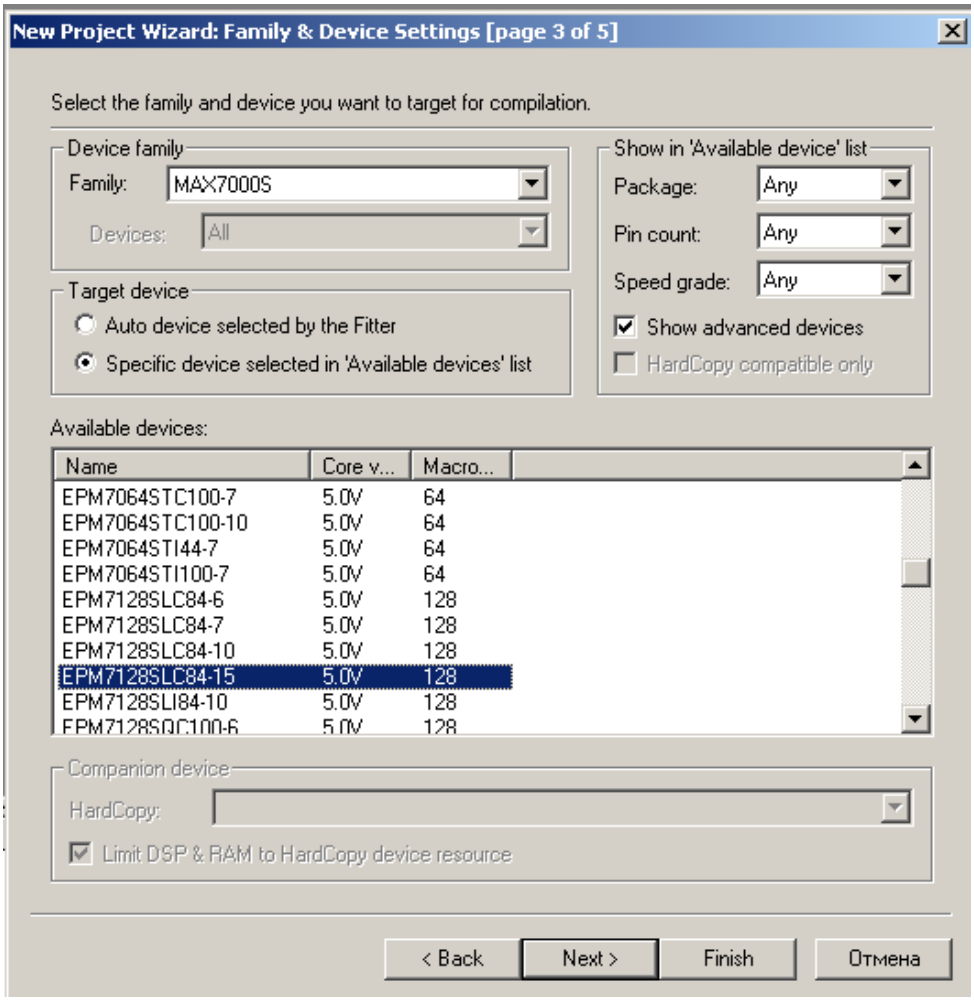
New Project Wizard: Family & Device Settings [page 3 of 5]
Select the family and device you want to target for compilation.
Device family
Family: MAX7000S
Devices: All
Target device
Auto device selected by the Fitter
Specific device selected in 'Available devices' list
Show in 'Available device' list
Package: Any
Pin count: Any
Speed grade: Any
Show advanced devices
HardCopy compatible only
Available devices:
Name Core v... Macro...
EPM7064STC100-7 5.0V 64
EPM7064STC100-10 5.0V 64
EPM7064STI44-7 5.0V 64
EPM7064STI100-7 5.0V 64
EPM7128SLC84-6 5.0V 128
EPM7128SLC84-7 5.0V 128
EPM7128SLC84-10 5.0V 128
EPM7128SLC84-15 5.0V 128
EPM7128SLI84-10 5.0V 128
Companion device
HardCopy:
Limit DSP & RAM to HardCopy device resource
< Back
Next >
Finish
Отмена

Рис. А.3

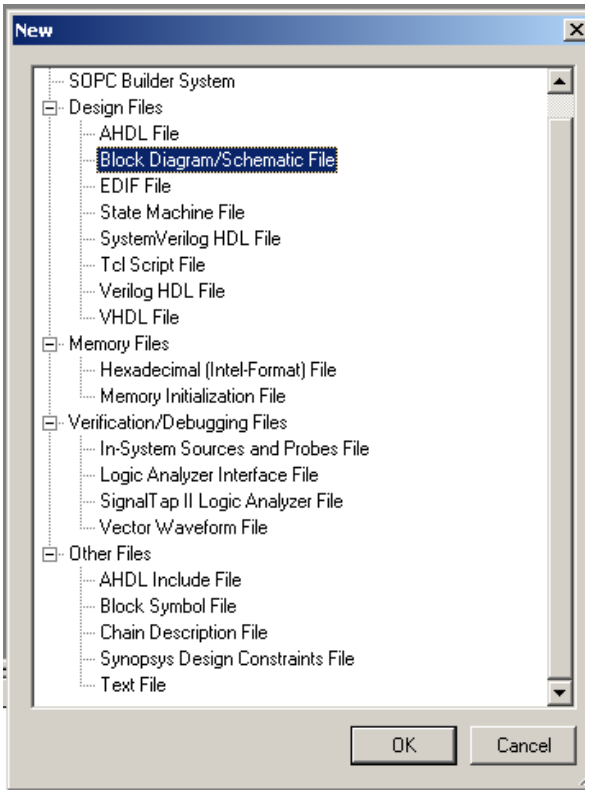
New
SOPC Builder System
Design Files
AHDL File
Block Diagram/Schematic File
EDIF File
State Machine File
SystemVerilog HDL File
Tcl Script File
Verilog HDL File
VHDL File
Memory Files
Hexadecimal (Intel-Format) File
Memory Initialization File
Verification/Debugging Files
In-System Sources and Probes File
Logic Analyzer Interface File
SignalTap II Logic Analyzer File
Vector Waveform File
Other Files
AHDL Include File
Block Symbol File
Chain Description File
Synopsys Design Constraints File
Text File
OK
Cancel

Рис. А.4

Для того, чтобы добавить только что созданный файл в проект, следует выбрать пункт **Add/Remove Files in Project** меню **Project**.

А.2. Конфигурирование ПЛИС

Для загрузки проекта в макет и конфигурирования микросхемы ПЛИС в соответствии с разработанным описанием необходимо установить соответствие между входными и выходными портами проекта и выводами микросхемы. Для этого выберите пункт **Pins** меню **Assignments** (рис.А.5).

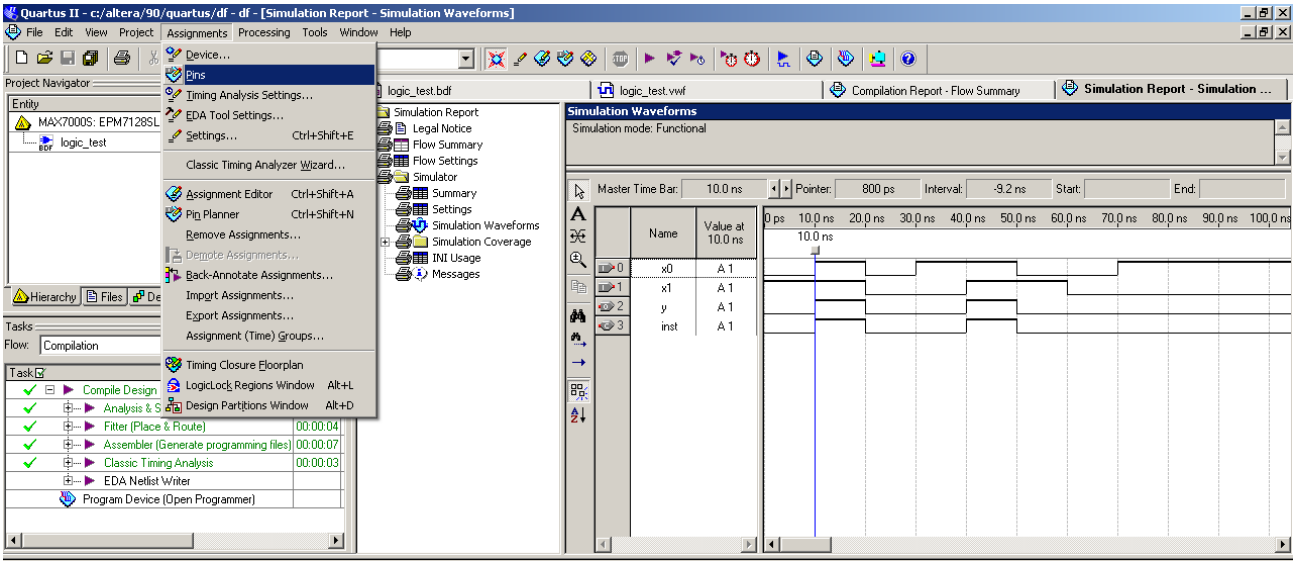

Рис. А.5

В открывшемся диалоговом окне **Pin Planner** (рис. А.6) содержится общий вид микросхемы с указанием расположения и типа выводов.

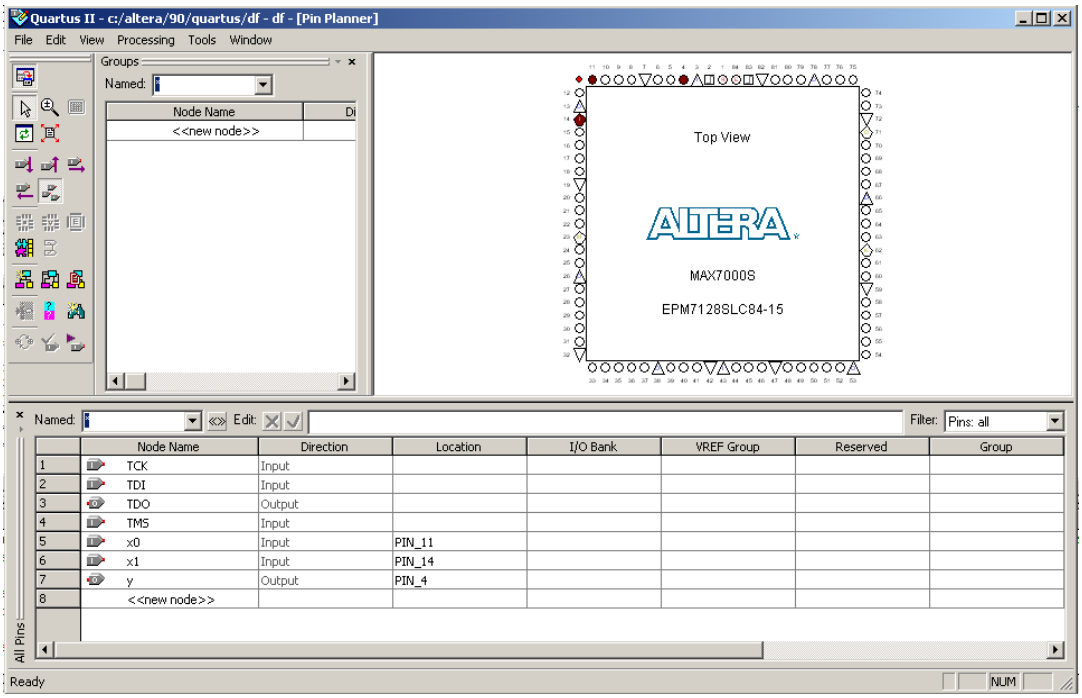

Рис. А.6

На панели **All pins** приведена таблица, в которой в столбце **Node Name** перечислены выводы конфигурационного интерфейса JTAG (сигналы TCK, TDI, TDO, TMS), а также порты пользовательского проекта. В столбце Location в соответствии с таблицей А.1 необходимо назначить выводы микросхемы, соответствующие портам проекта. После этого окно **Pin Planner** можно закрыть.

Таблица А.1 — Назначение выводов ПЛИС макета

Обозначение	**Номер вывода**	**Подключенный компонент**	**Обозначение**	**Номер вывода**	**Подключенный компонент**
BUTTON3	54	Кнопка 3	Индикатор 1		
BUTTON2	57	Кнопка 2	A1	31	Сегмент A
BUTTON1	56	Кнопка 1	B1	28	Сегмент B
BUTTON0	58	Кнопка 0	C1	16	Сегмент C
LED0	79	Светодиод 0	D1	15	Сегмент D
LED1	80	Светодиод 1	E1	12	Сегмент E
LED2	77	Светодиод 2	F1	30	Сегмент F
LED3	75	Светодиод 3	G1	34	Сегмент G
LED4	76	Светодиод 4	DP1	17	Сегмент DP
LED5	74	Светодиод 5	Индикатор 2		
LED6	73	Светодиод 6	A2	36	Сегмент A
LED7	70	Светодиод 7	B2	33	Сегмент B
SW0	68	Кнопка с фикс. 0	C2	10	Сегмент C
SW1	69	Кнопка с фикс. 1	D2	9	Сегмент D
SW2	67	Кнопка с фикс. 2	E2	8	Сегмент E
SW3	64	Кнопка с фикс. 3	F2	35	Сегмент F
SW4	65	Кнопка с фикс. 4	G2	37	Сегмент G
SW5	63	Кнопка с фикс. 5	DP2	11	Сегмент DP
SW6	60	Кнопка с фикс. 6	Индикатор 3		
SW7	61	Кнопка с фикс. 7	A3	39	Сегмент A
Индикатор 0			B3	40	Сегмент B
A0	24	Сегмент A	C3	5	Сегмент C
B0	25	Сегмент B	D3	4	Сегмент D
C0	20	Сегмент C	E3	81	Сегмент E
D0	21	Сегмент D	F3	41	Сегмент F
E0	18	Сегмент E	G3	44	Сегмент G
F0	27	Сегмент F	DP3	6	Сегмент DP
G0	29	Сегмент G			
DP0	22	Сегмент DP	CLK	83	Тактовый генератор

Обязательным шагом является выбор способа конфигурирования неиспользуемых в проекте выводов микросхемы. Для этого выберите пункт **Device** меню **Assignments**, далее в открывшемся диалоговом окне (рис.А.7) нажмите кнопку **Device and Pin Options…** .

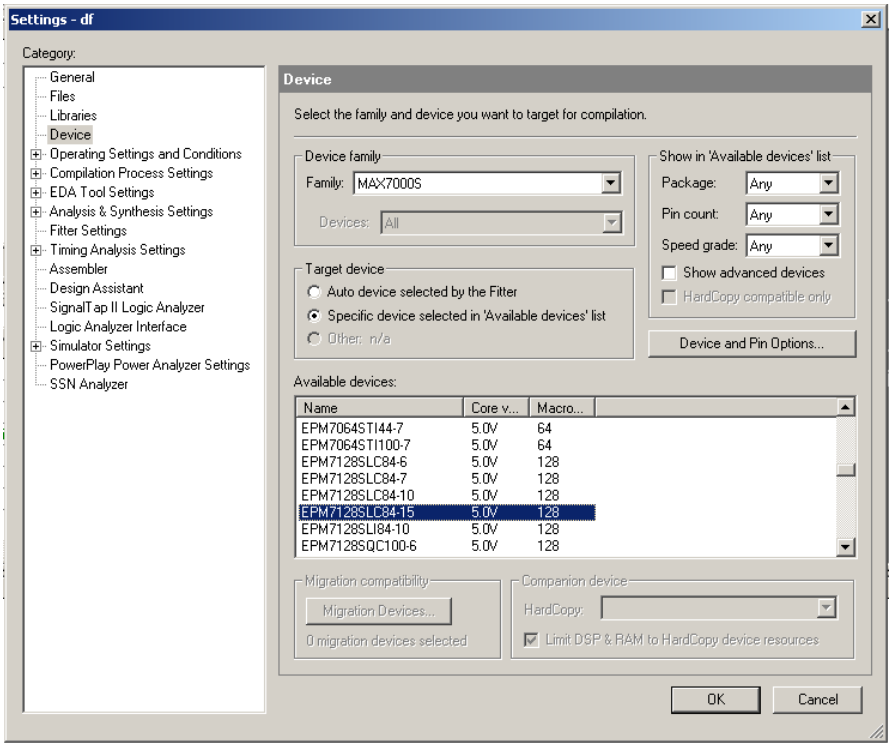

Рис. А.7

На вкладке **Unused Pins** в выпадающем списке **Reserve all unused pins** выберите пункт *As input tri-stated* (рис.А.7) и нажмите кнопку **OK**. Эта опция настроит все неиспользуемые в проекте выводы микросхемы как входы.

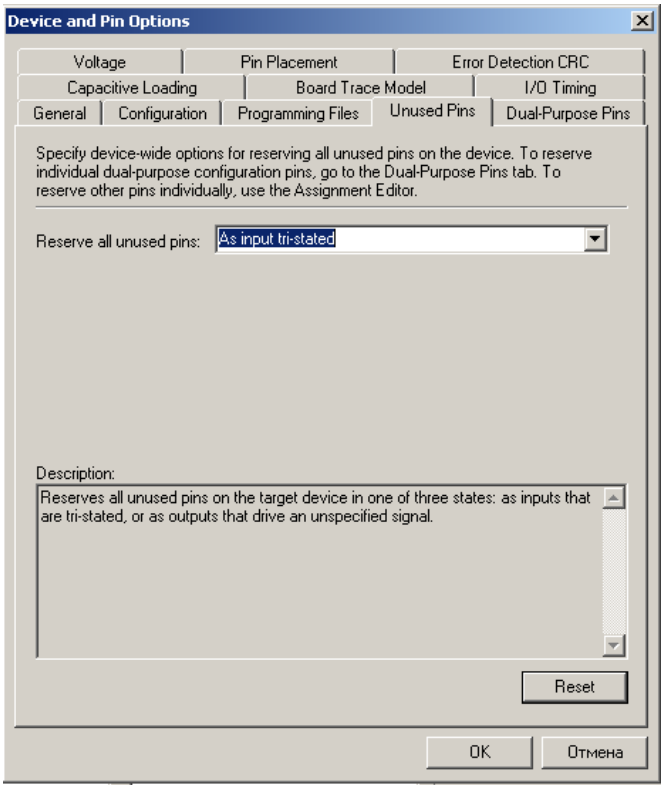

Рис. А.8

Выполните компиляцию проекта, щелкнув по пиктограмме  .

В случае успешной компиляции запустите окно программатора (рис.А.9), выбрав пункт меню **Programmer** меню **Processing**. Нажмите кнопку **Hardware Setup…**, далее кнопку **Add Hardware**, и затем кнопку **OK** (рис.А.10).

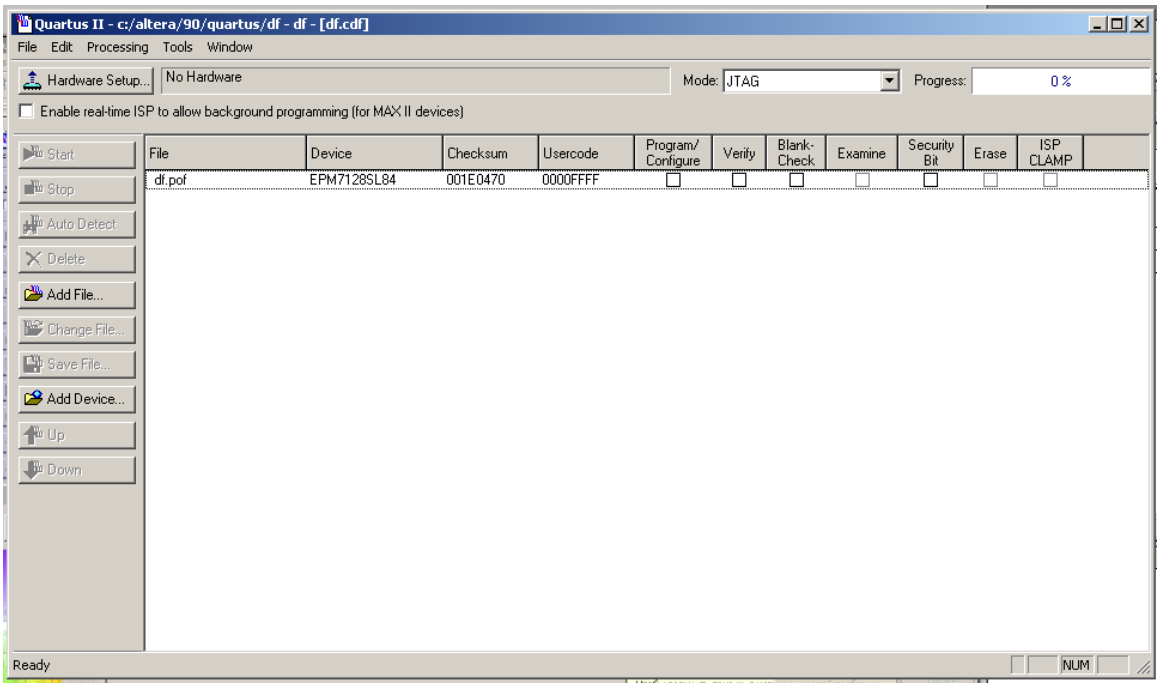

Рис. А.9

После этого в окне программатора установите галочку у пункта *Program/Configure* (рис.А.10) и нажмите кнопку **Start** Start. Статус загрузки отображается в поле **Progress** в правом верхнем углу окна программатора.

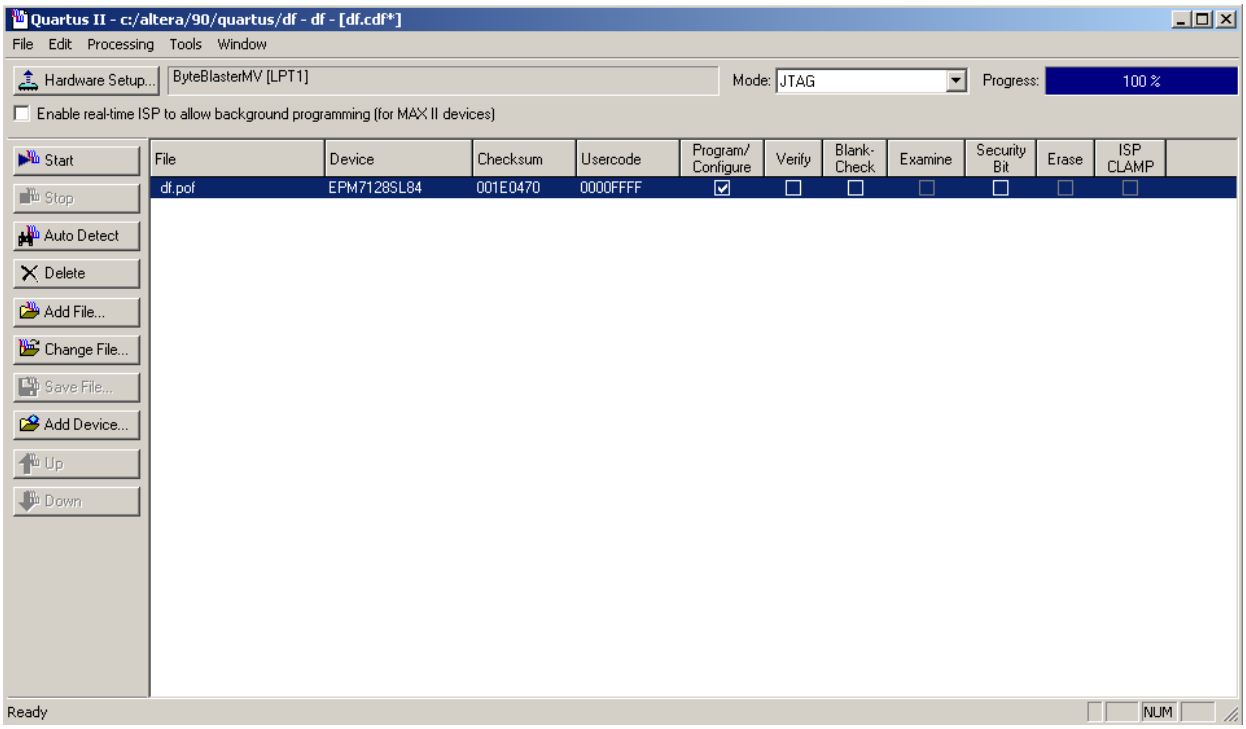

Рис. А.10

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Грушвицкнй Р.И. Проектирование систем на микросхемах программируемой логики / Р.И. Грушвицкий, А. Х. Мурсаев, Е. П. Угрюмое. — СПб.: БХВ-Петербург, 2002. — 608 с.

2. Максфилд К. Проектирование на базе ПЛИС. Архитектура, средства и методы / К. Максфилд. — М.: Додэка-XXI, 2007. — 408 с.

3. Угрюмов Е.П. Цифровая схемотехника / Е.П. Угрюмов. — СПб.: БХВ-Петербург, 2010. — 816 с.

4. Стешенко В.Б. ПЛИС фирмы Altera: элементная база, система проектирования и языки описания аппаратуры / В.Б. Стешенко. — М.: Додэка-XXI, 2002. — 576 с.

5. Глазков В. В. Программируемые логические интегральные схемы фирмы Altera: учеб. пособие по дисциплине «Технология и схемотехника средств управления в технических системах» / В. В. Глазков. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2014. — 133 с.

6. Brown S. Fundamentals of digital logic with Verilog design / S. Brown, Z. Vranesic. — New York: McGraw-Hill, 2012. — 864 p.

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ ИНТЕГРАЛЬНЫЕ СХЕМЫ.
Лабораторный практикум

Учебно-методическое пособие

Составители: **Начаров** Денис Владимирович,
Широков Игорь Борисович, **Скорик** Иван Викторович

В авторской редакции

Изд. № 179/2020. Объем 2 п.л.
РИИЦМ ФГАОУ ВО «Севастопольский государственный университет»