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

Факультет радиотехники и электроники

Кафедра радиотехнических систем

СИГНАЛЬНЫЕ ПРОЦЕССОРЫ В УСТРОЙСТВАХ ЦИФРОВОЙ РАДИОСВЯЗИ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Рекомендовано УМО по образованию
в области информатики и радиоэлектроники
в качестве учебно-методического пособия
для специальностей 1-39 01 01 «Радиотехника (по направлениям)»,
1-39 01 02 «Радиоэлектронные системы», 1-39 01 03 «Радиоинформатика»

Минск БГУИР 2015

УДК 004.383.3:621.396(076.5)
ББК 32.973.26-04я73+32.884.1я73
С34

А в т о р ы:
А. А. Казека, Е. Н. Каленкович, В. Н. Левкович, И. Г. Давыдов

Р е ц е н з е н т ы:

кафедра информатики и компьютерных систем Белорусского государственного университета (протокол №4 от 06.11.2014);

профессор кафедры программного обеспечения учреждения образования «Высший государственный колледж связи», кандидат технических наук, доцент О. Р. Ходасевич

Сигнальные процессоры в устройствах цифровой радиосвязи. С34 Лабораторный практикум : учеб.-метод. пособие / А. А. Казека [и др.]. – Минск : БГУИР, 2015. – 115 с. : ил.
ISBN 978-985-543-008-8.

Включает методические материалы для практического выполнения цикла из восьми лабораторных работ по курсу «Сигнальные процессоры в устройствах цифровой радиосвязи». Содержание издания направлено на получение студентами начальных знаний и приобретение умений по проектированию устройств формирования и обработки сигналов на современной цифровой элементной базе.

УДК 004.383.3:621.396(076.5)
ББК 32.973.26-04я73+32.884.1я73

ISBN 978-985-543-008-8

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современных радиоэлектронных системах формирование и обработка сигналов производится исключительно цифровыми методами, реализуемыми практически в виде сигнальных процессоров. В свою очередь сигнальные процессоры или их отдельные функциональные блоки физически реализуются на программируемых логических интегральных схемах (ПЛИС) и однокристальных сигнальных контроллерах.

Учебно-методическое пособие содержит методические материалы для выполнения цикла из восьми лабораторных работ. Первые четыре работы посвящены вопросам реализации цифровых устройств на ПЛИС, а остальные – реализации алгоритмов цифровой обработки сигналов на сигнальных контроллерах.

Тематика и наполнение работ подобраны таким образом, чтобы обучающиеся получили начальные знания и приобрели умения, достаточные для дальнейшего самостоятельного профессионального совершенствования.

1. РЕАЛИЗАЦИЯ ЦИФРОВЫХ УСТРОЙСТВ НА ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМАХ

1.1. Описание ПЛИС производства фирмы Xilinx

Одним из самых интересных и быстро развивающихся направлений современной цифровой микроэлектроники являются программируемые логические интегральные схемы (ПЛИС). С их появлением проектирование цифровых микросхем перестало быть уделом исключительно крупных предприятий с объемами выпуска в десятки и сотни тысяч кристаллов. Проектирование и выпуск небольшой партии уникальных цифровых устройств стало возможным в условиях проектно-конструкторских подразделений промышленных предприятий, в исследовательских и учебных лабораториях.

Сегодня существуют различные производители микросхем, которые выпускают ПЛИС, такие, как *Xilinx, Altera, Actel* и др. У каждой фирмы свои названия выпускаемых устройств и собственные системы проектирования, позволяющие проектировать весь спектр цифровых устройств типа FPGA/CPLD. Основные отличия производителей устройств ПЛИС друг от друга заключаются в архитектуре построения внутренних программируемых комбинационных схем, способах программирования ПЛИС, числе эквивалентных вентилей, технологии изготовления кристаллов, типах корпусов ПЛИС и т. д.

Фирма *Xilinx* предоставляет разработчикам широкий спектр кристаллов с различной технологией производства, степенью интеграции, архитектурой, быстродействием, потребляемой мощностью и напряжением питания, выпускаемых в различных типах корпусов и в нескольких вариантах исполнения, включая промышленное, военное и радиационно стойкое. *Xilinx* выпускает различные семейства ПЛИС, одним из которых является семейство программируемых логических интегральных схем структуры FPGA Spartan-3E с напряжением питания ядра 1,2 В, предназначенное для реализации проектов большого объема, в которых особенно важна стоимость реализации. Семейство ПЛИС Spartan-3E по объему системных вентилей подразделяется на 5 типов от 100 000 до 1 600 000 (табл. 1.1). Семейство Spartan-3E является модернизацией предыдущей версии ПЛИС семейства Spartan-3, в котором улучшены некоторые характеристики.

Особенности данного семейства:

1. Улучшенная 90 нм КМОП-технология изготовления.

2. Очень дешевое быстродействующее логическое решение для реализаций приложений, требующих большого объема системных вентилей.

3. Технология передачи сигналов SelectIO™:

– до 376 портов ввода/вывода или до 156 дифференциальных сигнальных пар;

– LVCMOS, LVTTL, HSTL, SSTL – линейные стандарты ввода/вывода;

– поддерживаемые напряжения портов ввода/вывода 3,3 В, 2,5 В, 1,8 В, 1,5 В, и 1,2 В;

– скорость передачи данных 622+ Мбит/с;

– True LVDS, RSDS, mini-LVDS, differential HSTL/SSTL дифференциальные стандарты ввода/вывода;

– поддержка Enhanced Double Data Rate (DDR);

– поддержка DDR SDRAM до 333 Мбит/с.

Таблица 1.1

ПЛИС	Объем системных вентилей	Эквивалентное число логических ячеек	Конфигурируемые логические ячейки (КЛБ)			Распределенная память, кбит	Блочная память, кбит	Количество умножителей	Количество DCM	Максимально доступное пользователю количество выводов	Максимальное число дифференциальных входов/выходов
			Ряды	Колонки	Всего КЛБ						
XC3S100E	100 к	2 160	22	16	240	15	72	4	2	108	40
XC3S250E	250 к	5 508	34	26	612	38	216	12	4	172	68
XC3S500E	**500 к**	**10 476**	**46**	**34**	**1 164**	**73**	**360**	**20**	**4**	**232**	**92**
XC3S1200E	1 200 к	19 512	60	46	2 168	136	504	28	8	304	124
XC3S1600E	1 600 к	33 192	76	58	3 688	231	648	36	8	376	156

4. Логические ресурсы:

– большое количество логических ячеек (до 33 192), включая опционально регистры сдвига или поддержку распределенной RAM;

– многоразрядные мультиплексоры;

– быстродействующие логические ячейки;

– встроенные 18×18 бит умножители;

– JTAG-порт для программирования и отладки, совместимый со стандартом IEEE 1149.1/1532.

5. Иерархическая память SlectRAM™:

– объем быстрой блочной памяти до 648 кбит;

– объем распределенной памяти до 231 кбит.

6. До восьми блоков цифрового управления тактовой частоты (Digital Clock Managers – DCM):

– устранение задержек;

– синтез частоты, умножение и деление частоты;

– сдвиг фазы с высоким разрешением;

– широкий частотный диапазон от 5 до 300 МГц.

7. Восемь глобальных тактовых шин.

8. Полная поддержка программным обеспечением Xilinx ISE®.

9. Поддержка встраиваемых процессорных ядер MicroBlaze™ и PicoBlaze™.

10. Полная поддержка 32/64 бит 33 МГц PCI (66 МГц в некоторых устройствах).

Архитектура ПЛИС семейства Spartan-3E состоит из пяти фундаментальных программируемых функциональных элементов.

Конфигурируемые логические блоки (КЛБ, CLB) – содержат основанные на элементах памяти таблицы (LUT) для реализации логики и элементы хранения, которые могут использоваться как триггеры или «защелки». КЛБ могут быть запрограммированы для реализации логической функции или же элемента памяти.

Блоки ввода/вывода (БВВ, IOB) – управляют потоком данных между блоками ввода/вывода и внутренней логикой. Каждый БВВ поддерживает возможность двунаправленного обмена данными с возможностью перевода вывода в 3-е состояние. БВВ поддерживают различные стандарты, которые могут быть запрограммированы пользователем. В каждом БВВ имеется возможность поддержки удвоенной тактовой частоты (стандарт DDR).

Блочная память (Block RAM) обеспечивает хранение данных в форме 18-килобитных двухпортовых блоков.

Умножители (Multiplier Block) принимают два 18-битных двоичных числа на входе и производят их умножение.

Блоки управления тактовой частотой (Digital Clock Manager, DCM) – обеспечивают автокалибровку, полностью реализованные цифровым методом задержку, умножение, деление и фазовый сдвиг тактового сигнала.

Расположение элементов на кристалле показано на рис. 1.1.

Кольцо блоков ввода/вывода окружает регулярную матрицу конфигурируемых логических блоков. Каждый из кристаллов ПЛИС семейства Spartan-3E имеет две колонки блочной памяти за исключением XC3S100E, содержащего одну колонку. Каждая колонка составлена из блоков RAM на 18 кбит. Каждый блок памяти связан с умножителем 18×18 бит. Модули управления синхронизацией (DCM) расположены в центре по два сверху и снизу кристалла ПЛИС. Кристалл ПЛИС XC3S100E имеет только по одному блоку управления тактовой частотой сверху и снизу кристалла, а кристаллы ПЛИС XC3S1200E и XC3S1600E имеют также дополнительно по два блока в центре кристалла справа и слева.

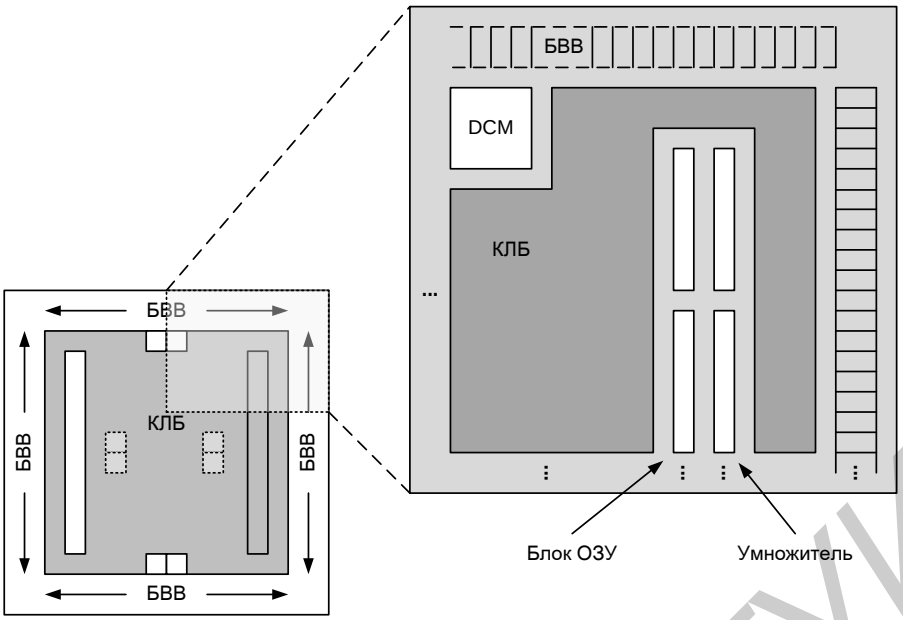


Рис. 1.1. Архитектура ПЛИС семейства Spartan-3E

1.2. Описание лабораторного макета

Лабораторный макет для выполнения лабораторных работ по разработке цифровых устройств на ПЛИС представляет собой отладочную плату ***Nexys2*** производства компании *Digilent Inc.* Данная отладочная плата позволяет производить разработку и отладку цифровых устройств и систем, выполненных на базе ПЛИС.

Основным элементом отладочной платы является кристалл ПЛИС XC3S500E семейства Spartan-3E производства фирмы *Xilinx* с числом системных вентилей 500 к. Кроме него на отладочной плате имеется набор периферийных устройств, позволяющих расширить возможности разработчика при разработке и отладке цифровых устройств. Структурная схема отладочной платы представлена на рис. 1.2.

В состав отладочной платы входят следующие основные элементы:

1) кристалл ПЛИС Spartan-3E-500 в корпусе FG320;

2) высокоскоростной интерфейс USB2.0, который используется для конфигурирования кристалла ПЛИС и внешней микросхемы конфигурации по JTAG, а также для передачи данных между разработанным устройством на базе ПЛИС и ПК;

3) внешняя Flash ROM память для хранения файла конфигурации ПЛИС;

4) тактовый генератор частотой 50 МГц;

5) микросхема памяти Flash емкостью 16 Мбайт и оперативной памяти SDRAM емкостью 16 Мбайт;

6) устройства для ввода/вывода информации (светодиодный семисегментный индикатор, светодиодные индикаторы, тактовые кнопки, ползунковые переключатели);

7) интерфейс PS/2;

8) видеоинтерфейс VGA;

9) интерфейс RS-232C;

10) разъемы для подключения дополнительных модулей расширения *Pmod* (4 шт.);

11) высокоскоростная шина передачи данных.

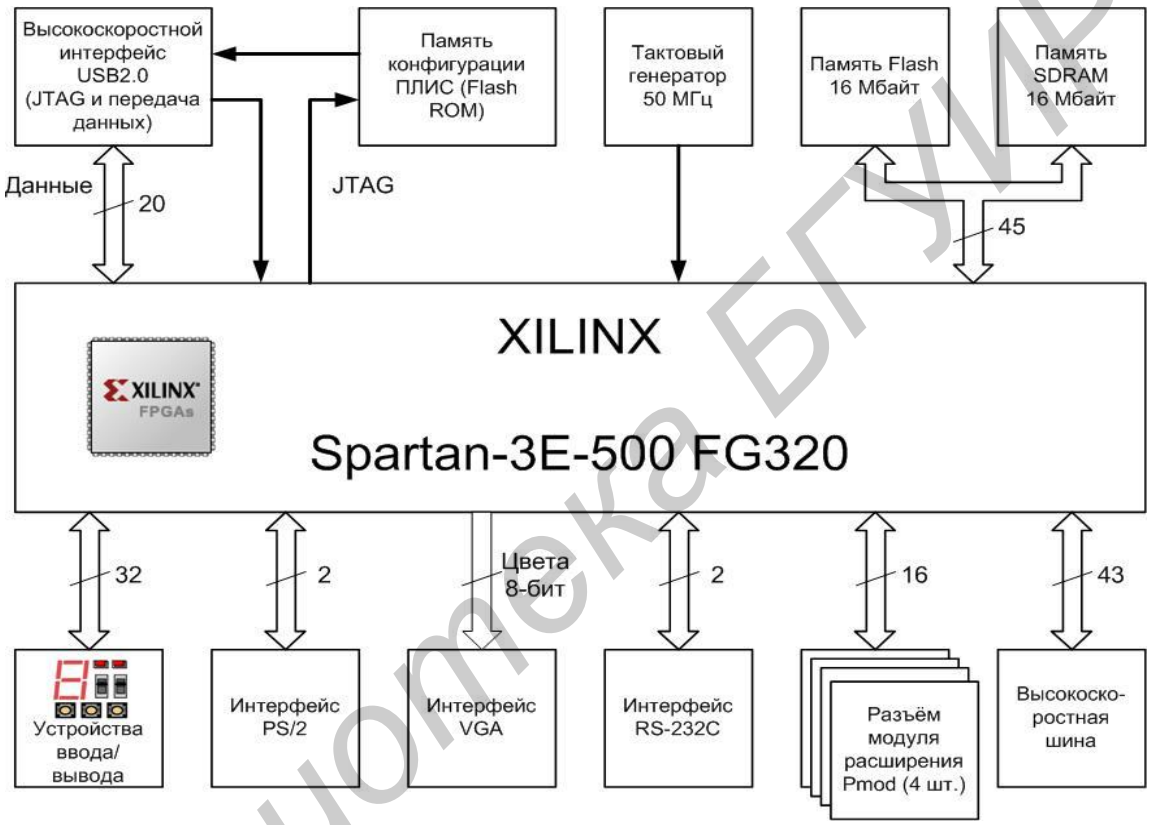


Рис. 1.2. Структура отладочной платы

Внешний вид и расположение основных элементов на отладочной плате показаны на рис. 1.3.

Номерами на рисунке отмечены:

1) перемычка для переключения типа питания отладочной платы (батарея, от шины USB, внешний источник питания);

2) разъем видеоинтерфейса VGA;

3) DIP-панель для подключения дополнительного тактового генератора;

4) разъем интерфейса RS-232C;

5) разъем интерфейса USB2.0;

6) разъем интерфейса PS/2;

7) восемь светодиодных индикаторов;

8) восемь ползунковых переключателей;

9) четыре тактовых кнопочных переключателя;

10) четырехразрядный семисегментный светодиодный индикатор;

11) разъем высокоскоростной шины для подключения дополнительных устройств;

12) тактовый генератор 50 МГц;

13) микросхема Flash ROM для хранения файла конфигурации кристалла ПЛИС;

14) кнопка сброса «RESET»;

15) стандартный штыревой разъем для подключения JTAG кабеля при программировании и отладке;

16) четыре 12-выводных разъема для подключения дополнительных модулей расширения *Pmod*;

17) выключатель питания отладочной платы;

18) разъем для подключения внешнего источника питания напряжением +5…15 В;

19) микросхема памяти Flash емкостью 16 Мбайт и SDRAM емкостью 16 Мбайт (установлена с обратной стороны отладочной платы);

20) кристалл ПЛИС XC3S500E семейства Spartan-3E.

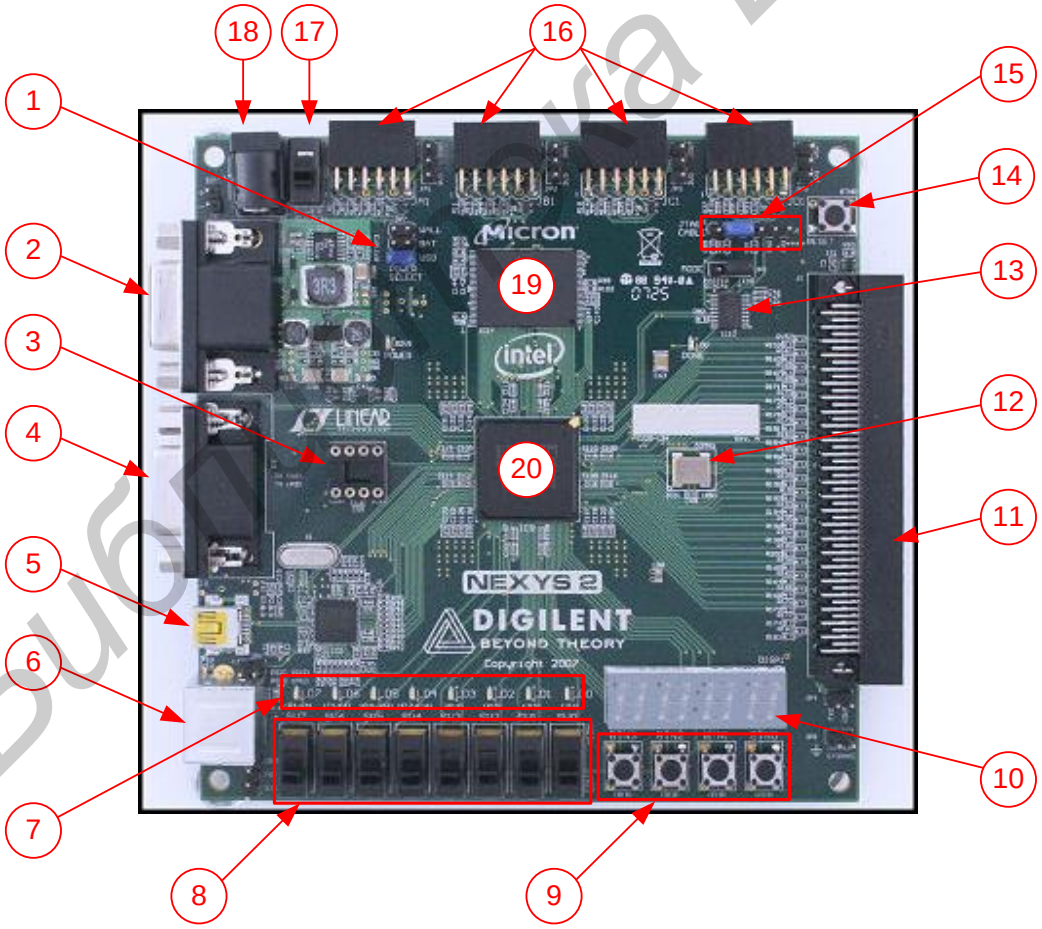


Рис. 1.3. Внешний вид отладочной платы ***Nexys2***

На плате кроме указанных элементов установлены некоторые другие, которые позволяют управлять режимами работы отладочной платы. Более подробную информацию можно получить в техническом описании.

Для конфигурирования и отладки устройства, реализованного на ПЛИС, используется JTAG интерфейс. По этому интерфейсу производится загрузка файла конфигурации в энергозависимую память кристалла ПЛИС или загрузка файла конфигурации во внешнюю микросхему памяти Flash ROM. При включении питания и соответствующем режиме загрузки информация из внешней памяти переписывается во внутреннюю энергозависимую память ПЛИС. При успешной загрузке на отладочной плате загорается светодиод «DONE» желтого свечения. При нажатии кнопки «RESET» производится сброс энергозависимой памяти ПЛИС и ее перезапись, если выбрана загрузка из внешней памяти. При выполнении лабораторных работ производится запись только в энергозависимую память ПЛИС, т. к. ресурс циклов записи внешней памяти Flash ROM ограничен.

Тактирование работы цифрового устройства может производиться от расположенного на отладочной плате тактового генератора частотой 50 МГц, а также от дополнительного генератора, который может быть установлен на плате в DIP-панели. Генератор 50 МГц подключен к выводу **B8** кристалла ПЛИС, дополнительный генератор подключается к выводу **U9** ПЛИС.

На отладочной плате предусмотрен набор устройств, позволяющий разработчику производить индикацию и ввод данных. К ним относится набор из восьми одиночных светодиодов, восьми ползунковых переключателей, четырех тактовых кнопок, четырехразрядного семисегментного светодиодного индикатора с общим анодом. Схема подключения этих элементов к ПЛИС показана в табл. 1.2, 1.3, 1.4, 1.5 и 1.6 соответственно.

Таблица 1.2

Светодиод	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
Вывод ПЛИС	R4	F4	P15	E17	K14	K15	J15	J14

Каждый светодиод подключен катодом к общей «земле» через токоограничительный резистор сопротивлением 390 Ом. Ток индикации составляет около 3…4 мА на каждый светодиод. Для управления светодиодом необходимо выставить уровень логической «1» на соответствующем выводе ПЛИС. Светодиоды маркированы на плате символами LD0…LD7.

Таблица 1.3

Переключатель	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
Вывод ПЛИС	R17	N17	L13	L14	K17	K18	H18	G18

Переключатели маркированы на отладочной плате символами SW0…SW7. При установке переключателя в положение «ON» производится

подача логической «1» +3,3 В на указанный вывод кристалла ПЛИС, при установке в положение «OFF» – логического «0». Защита от дребезга контактов отсутствует. Защита по току осуществляется при помощи последовательно включенного резистора сопротивлением 10 кОм.

Таблица 1.4

Кнопка	BTN3	BTN2	BTN1	BTN0
Вывод ПЛИС	H13	E18	D18	B18

Тактовые кнопки подключены одним выводом к общей «земле» через резисторы сопротивлением 10 кОм, другой вывод подключен к источнику питания напряжением +3,3 В и обеспечивает подачу логической «1» на соответствующий вывод кристалла ПЛИС при замыкании кнопки. Защита по току осуществляется последовательно включенным резистором сопротивлением 10 кОм.

Четырехразрядный семисегментный индикатор имеет общую шину данных для засветки определенного сегмента и отдельные выводы управления для разрешения вывода информации в определенный разряд. Светодиоды сегментов индикатора включены по схеме с общим анодом. В табл. 1.5 показано подключение выводов сегментов к выводам ПЛИС, а в табл. 1.6 – подключение общих выводов управления.

Таблица 1.5

Сегмент	A	B	C	D	E	F	G	DP
Вывод ПЛИС	L18	F18	D17	D16	G14	J14	H14	C17

Таблица 1.6

Общий вывод индикатора	AN0	AN1	AN2	AN3
Вывод ПЛИС	F17	H17	C18	F15

Схема расположения элементов на индикаторе показана на рис. 1.4.

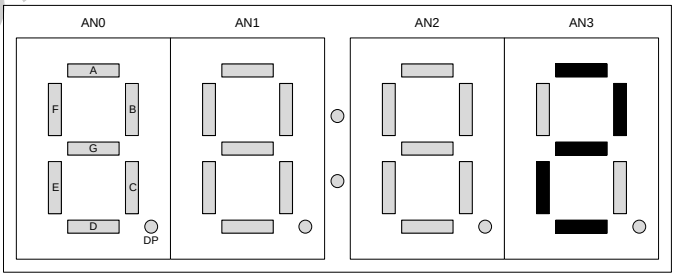


Рис. 1.4. Расположение элементов на семисегментном индикаторе

Для управления индикатором используется принцип динамической индикации, последовательность управляющих сигналов, реализующая этот способ, показана на рис. 1.5.

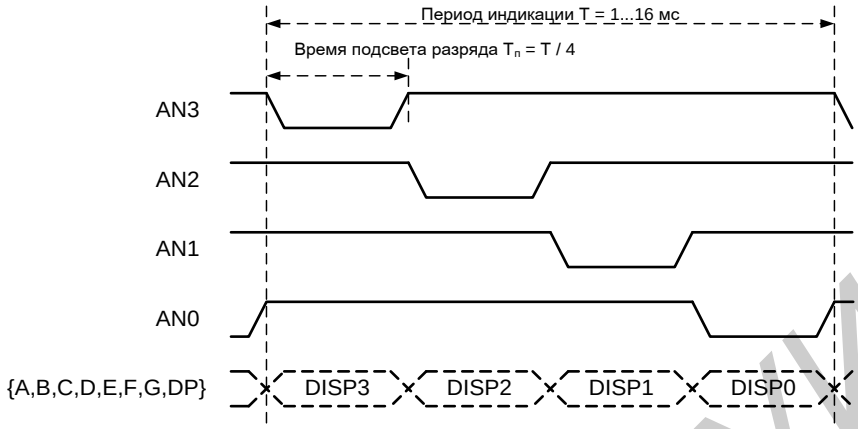


Рис. 1.5. Управляющие сигналы при динамической индикации

Для расширения возможностей отладочной платы имеется возможность подключения внешних модулей расширения *Peripheral Module* или *Pmod* посредством четырех 12-выводных разъемов. Всего имеется возможность подключения до 8 модулей расширения. Каждый модуль подключается к отладочной плате через 6-выводной штыревой разъем.

Все выводы, использующиеся для передачи и приема данных, имеют защиту от статического электричества в виде защитных диодов и защиту по току в виде последовательно включенных резисторов сопротивлением 200 Ом. Схема включения одного порта расширения с указанием расположения выводов показана на рис. 1.6.

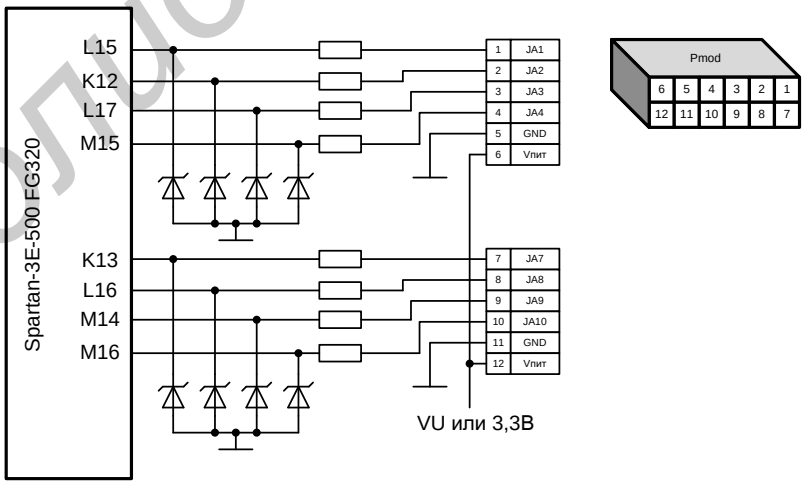


Рис. 1.6. Схема порта расширения *Pmod*

В табл. 1.7 приведен список контактов ПЛИС и соответствующих им выводов портов расширения.

Таблица 1.7

Pmod JA								
Вывод разъема	JA1	JA2	JA3	JA4	JA7	JA8	JA9	JA10
Вывод ПЛИС	L15	K12	L17	M15	K13	L16	M14	M16
Pmod JB								
Вывод разъема	JB1	JB2	JB3	JB4	JB7	JB8	JB9	JB10
Вывод ПЛИС	M13	R18	R15	T17	P17	R16	T18	U18
Pmod JC								
Вывод разъема	JC1	JC2	JC3	JC4	JC7	JC8	JC9	JC10
Вывод ПЛИС	G15	J16	G13	H16	H15	F14	G16	J12
Pmod JD								
Вывод разъема	JD1	JD2	JD3	JD4	JD7	JD8	JD9	JD10
Вывод ПЛИС	J13	M18	N18	P18	K14*	K15*	J15*	J14*
*K14 – соединен с LD3; K15 – соединен с LD2; J15 – соединен с LD1; J14 – соединен с LD0.								

Для выполнения лабораторных работ используются два дополнительных модуля расширения: модуль цифроаналогового преобразователя ***PmodDA2*** и модуль аналого-цифрового преобразователя ***PmodAD1*** (рис. 1.7).

а *б*

Рис. 1.7. Модуль ЦАП (*а*) и модуль АЦП (*б*)

Модуль ЦАП ***PmodDA2*** представляет собой двухканальный 12-разрядный преобразователь сигналов из цифровой формы в аналоговую, выполненный на основе микросхем DAC121S101, и обеспечивает формирование аналоговых сигналов напряжением от 0 до +3,3 В. Назначение выводов входного и выходного разъемов модуля представлено в табл. 1.8.

Модуль АЦП ***PmodAD1*** предназначен для преобразования аналогового сигнала в цифровой код и представляет собой двухканальный 12-разрядный АЦП, выполненный на микросхемах ADCS7476MSPS. Данный модуль преобразует входной аналоговый сигнал напряжением в диапазоне от 0 до 3,3 В в цифровой 12-разрядный последовательный код. Назначение выводов модуля показано в табл. 1.9.

Таблица 1.8

Разъем ***J1*** – цифровой интерфейс		Разъем ***J2*** – аналоговый интерфейс	
Вывод	Назначение	Вывод	Назначение
1	$\overline{SYNC}$ – вход синхронизации кадров (общий для двух каналов)	1	*VOUTA* – выход аналогового сигнала канала А
2	*DINA* – вход последовательных данных канала А	2	*N/C* – не подключен
3	*DINB* – вход последовательных данных канала В	3	*VOUTB* – выход аналогового сигнала канала В
4	*SCLK* – вход синхронизации передаваемых бит данных (общий для двух каналов)	4	*N/C* – не подключен
5	*GND* – общий провод шины питания	5	*GND* – общий провод шины питания
6	*VCC* – напряжение питания модуля ЦАП	6	*VCC* – напряжение питания модуля ЦАП

Таблица 1.9

Разъем ***J1*** – цифровой интерфейс		Разъем ***J2*** – аналоговый интерфейс	
Вывод	Назначение	Вывод	Назначение
1	$\overline{CS}$ – вход выбора микросхемы (общий для двух каналов)	1	*VINA* – вход аналогового сигнала канала А
2	*SDATA1* – выход последовательных данных канала А	2	*GND* – общий провод шины питания
3	*SDATA2* – выход последовательных данных канала В	3	*VINB* – вход аналогового сигнала канала В
4	*SCLK* – вход синхронизации передаваемых бит данных (общий для двух каналов)	4	*GND* – общий провод шины питания
5	*GND* – общий провод шины питания	5	*GND* – общий провод шины питания
6	*VCC* – напряжение питания модуля ЦАП	6	*VCC* – напряжение питания модуля ЦАП

Более подробную информацию можно найти в техническом описании на данные модули. Принцип работы с данными модулями расширения будет рассмотрен далее в описании лабораторных работ.

1.3. *Лабораторная работа №1*. Реализация последовательностных устройств на ПЛИС

Цель работы

1. Углубление теоретических знаний по схемотехническому проектированию цифровых устройств на ПЛИС и закрепление их на практике.

2. Формирование практических навыков создания цифровых устройств на основе ПЛИС путем схемотехнического и поведенческого описания работы устройства, а также компьютерного моделирования их работы.

3. Приобретение практических навыков работы с реальными устройствами на базе ПЛИС и контрольно-измерительными приборами.

Краткие теоретические сведения

Все цифровые устройства можно разделить на два класса: *последовательностные* и *комбинационные*.

К *последовательностным* устройствам, или цифровым автоматам с памятью, относят цифровые устройства, выходные сигналы которых описываются внутренним состоянием автомата и комбинацией сигналов на его входах.

Цифровой автомат – дискретный преобразователь информации, способный принимать различные состояния, переходить под воздействием входных сигналов или команд программы решения задачи из одного состояния в другое и выдавать выходные сигналы. Автомат называется конечным, если множество его внутренних состояний, а также множества значений входных и выходных сигналов конечны. Примерами цифровых последовательностных устройств являются триггеры, а также регистры и счетчики, выполненные на их основе.

В ходе лабораторной производится ознакомление со средой сквозного проектирования **Xilinx ISE Design Suite** версий 12.x и выше на примере реализации последовательностного устройства в виде двоичного делителя частоты.

В данной работе рассматривается пример реализации делителя частоты на ПЛИС. Под делителем частоты понимается цифровое последовательностное устройство, которое производит формирование частоты в целое число N раз меньше входного тактового сигнала. Простейшим делителем частоты в цифровой технике выступает T-триггер, который из входного импульсного сигнала формирует выходной сигнал с частотой, в два раза меньше входного.

Создание проекта и порядок работы с Xilinx ISE Design Suite

Для начала работы с пакетом сквозного проектирования **Xilinx ISE Design Suite** необходимо запустить управляющую оболочку – *Навигатор проекта (Project Navigator)*. Программа *Навигатора проекта* предназначена для

организации эффективного управления процессом проектирования цифровых устройств на базе ПЛИС *Xilinx*. *Навигатор проекта* представляет собой удобный графический интерфейс для работы с проектом и управления всеми процессами в ходе проектирования, включая программирование ПЛИС. Запуск всех необходимых программных модулей пакета осуществляется непосредственно в среде *Навигатора проекта*. Для запуска *Навигатора проекта* необходимо на рабочем столе запустить ярлык ISE Design Suite (рис. 1.8).

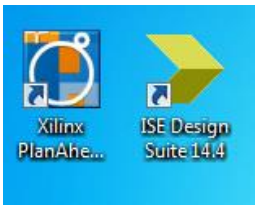


Рис. 1.8. Ярлык для запуска *Навигатора проекта* ISE Design Suite

После запуска программы появится основное окно, при первом запуске имеющее вид, показанный на рис. 1.9.

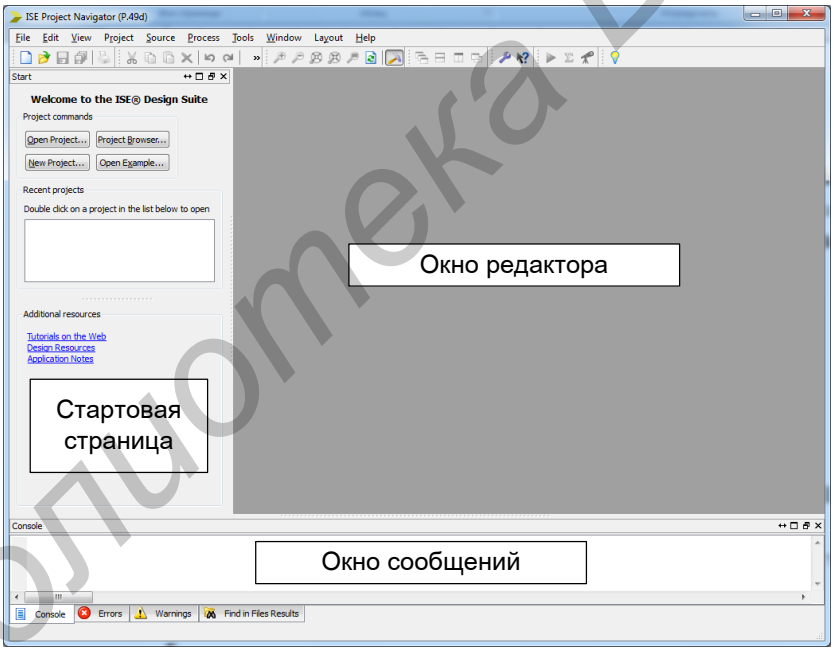


Рис. 1.9. Главное окно программы

На стартовой странице пользователю предлагается выбрать одно из следующих действий:

– открыть ранее созданный проект (кнопка «***Open Project…***»);
– создать новый проект (кнопка «***New Project…***»);
– запустить менеджер проектов (кнопка «***Project Browser…***»);
– открыть пример (кнопка «***Open Example…***»).

Названия последних проектов, с которыми производились какие-либо операции, отображаются в окне «***Recent project***».

Окно редактора – окно, в котором происходят все действия с проектом: ввод описаний элементов на языке VHDL, создание схем при помощи схемотехнического редактора, просмотр отчетов и т. д.

Окно сообщений – окно, в котором выводятся все сообщения о процессе выполнения операции или об ошибках, возникших во время ее выполнения.

Для создания нового проекта необходимо нажать кнопку ***New Project*** на стартовой странице либо выбрать в основном меню *Навигатора проекта* ***File*** > ***New Project***. В появившемся окне (рис. 1.10) в графе **Name** указывается название создаваемого проекта, в графе **Location** – расположение проекта на жестком диске. Графа **Working Directory** заполняется автоматически с графой **Location**. Для описания верхнего уровня проекта выбираем схемотехнический тип (Schematic) в графе **Top-level source type**.

ВНИМАНИЕ!!! При создании проекта в его названии и пути расположения проекта на диске нельзя использовать кодировку Unicode или КОИ-8 (кириллические символы). Новый проект необходимо создавать в отдельном каталоге.

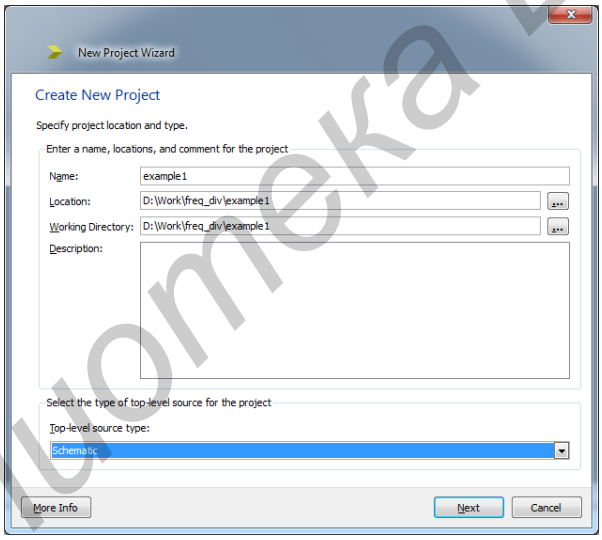


Рис. 1.10. Окно *New Project*

После нажатия кнопки **Далее** (**Next**) появляется новое окно (рис. 1.11), в котором предлагают выбрать используемую отладочную плату (**Evaluation Development Board**) либо указать семейство ПЛИС (**Family**), ПЛИС в выбранном семействе (**Device**) и тип корпуса (**Package**). Для используемой в лабораторных работах отладочной платы ***Nexys 2*** производства Digilent Inc. необходимо выбрать *Spartan-3E Starter Board* в графе **Evaluation Development Board** или вручную указать настройки, как на рис. 1.11.

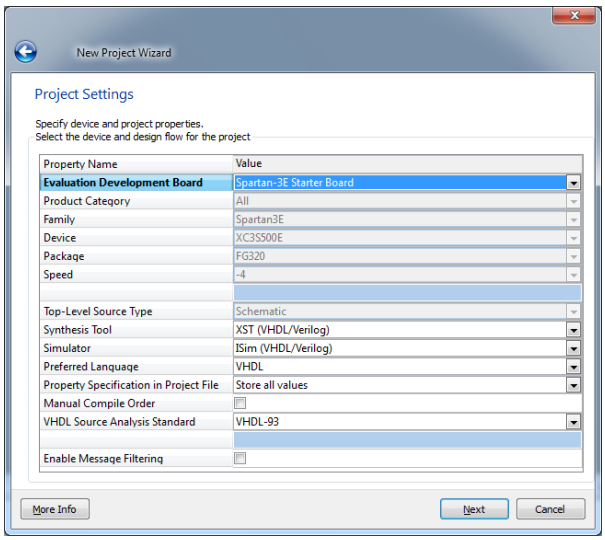


Рис. 1.11. Окно выбора базовых параметров проекта

В следующем окне (рис. 1.12) отображается суммарная информация по создаваемому проекту. Убеждаемся в правильности выбранных параметров и нажимаем кнопу **Готово** (**Finish**).

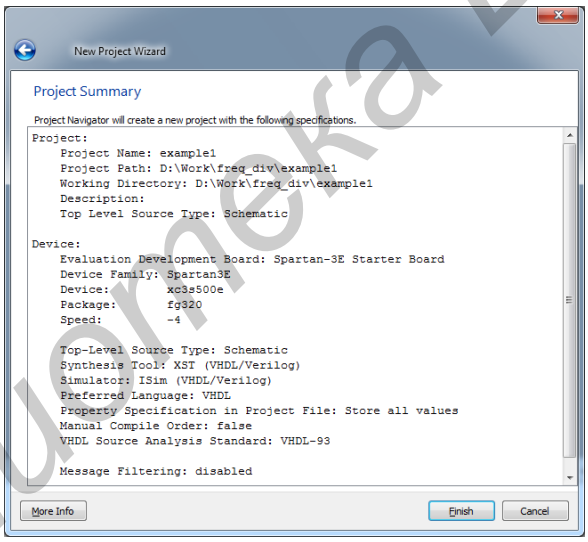


Рис. 1.12. Окно верификации настроек проекта

После создание проекта главное окно *Навигатора проекта* примет вид, показанный на рис. 1.13.

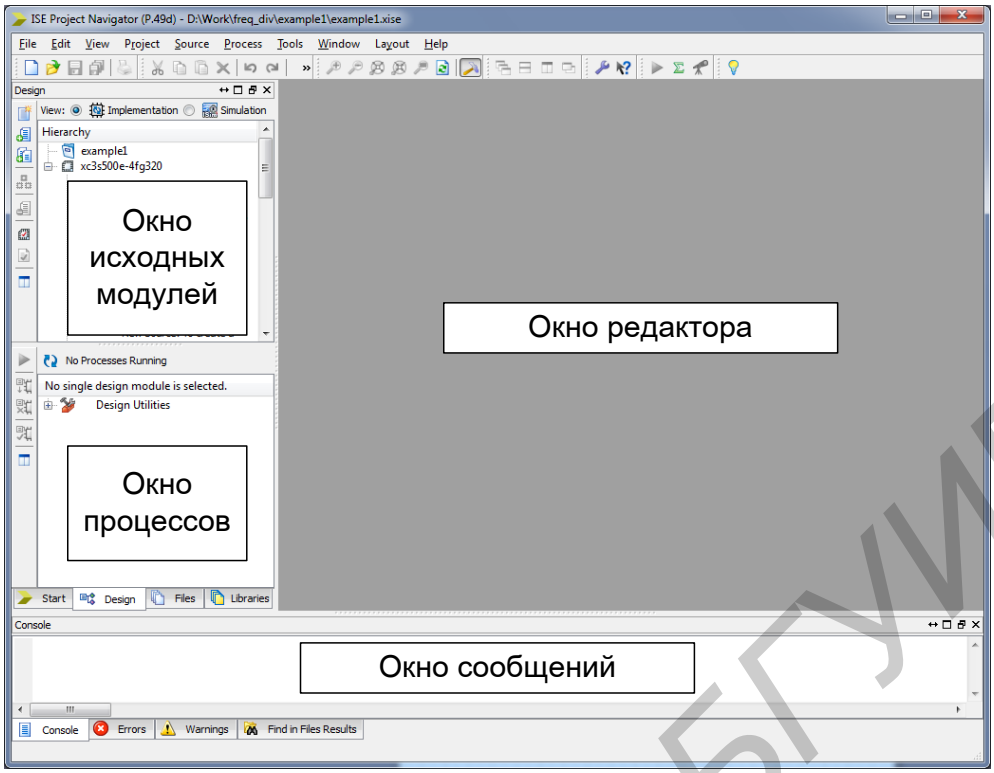


Рис. 1.13. Главное окно проекта

В этом окне кроме описанных окон отображаются следующие:

Окно исходных модулей – окно с отображением иерархической структуры проекта, состоящего из модулей, в которых содержится описание проектируемого устройства и входных тестовых воздействий.

Окно процессов – окно, с отображением всех возможных действий над выбранным модулем, а также всех этапов процесса разработки и конфигурирования ПЛИС. Процесс активизируется двойным нажатием на нем левой клавиши мыши, и, в зависимости от конечного результата, его состояние отображается одной из трех пиктограмм:

– процесс завершен успешно;

– процесс завершен успешно, но имеются предупреждения;

– процесс не завершен, имеются ошибки.

Более подробно ошибки можно посмотреть в отчете или же в ***окне сообщений***.

После создания проекта можно приступить к вводу модулей. Для этого на окне исходных модулей правой кнопкой мыши вызываем контекстное меню и выбираем источник нового модуля (рис. 1.14).

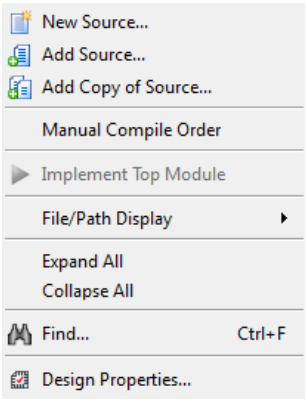


Рис. 1.14. Подключение модулей к проекту

Если модуль создается, то выбираем пункт меню **New Source…**, если же модуль уже существует, то выбираем пункт меню **Add Source…**, после чего система попросит указать расположение подключаемого файла. Эти же действия можно произвести и через основное меню *Навигатора проекта*, вызвав **Project > New Source…** или **Project > Add Source…**. Если выбран новый модуль, то в появившемся окне **New Source Wizard** (рис. 1.15) выбираем тип модуля и присваиваем ему какое-либо имя в графе **File name**.

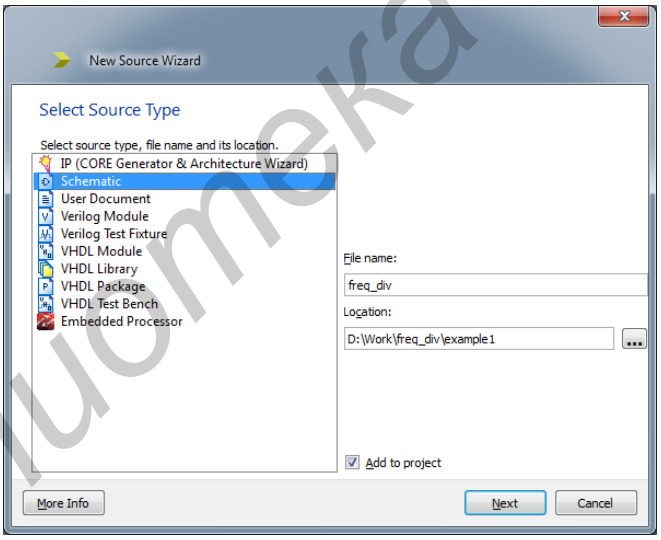


Рис. 1.15. Создание нового модуля

Расположение модуля (**Location**) по умолчанию указывает на директорию проекта. Если его не изменять, то модуль будет физически расположен в корневом каталоге проекта. Флажок возле метки **Add to project** автоматически присоединяет создаваемый файл к проекту.

Существует три основных типа ввода проекта: схемотехнический (*Schematic*), с помощью языка высокого уровня (*VHDL Module*) и с помощью диаграмм состояний (*State Diagram*). Проект может быть как целиком описан одним из трех методов, так и иметь смешанный тип. В этом случае из модулей генерируются библиотечные элементы редактора схем. Для выполнения лабо-

раторных работ ограничимся описанием работы с редактором схем (*Schematic*) и описанием работы устройства на языке высокого уровня VHDL.

Проектирование цифрового устройства при помощи схемотехнического описания в редакторе схем Schematic Editor

Чтобы добавить в проект модуль с электрической принципиальной схемой проектируемого цифрового устройства, в окне создания нового модуля выбирается пункт **Schematic**. После нажатия кнопок **Далее** (**Next**) и **Готово** (**Finish**) автоматически запускается схемный редактор *Schematic Editor* (рис. 1.16).

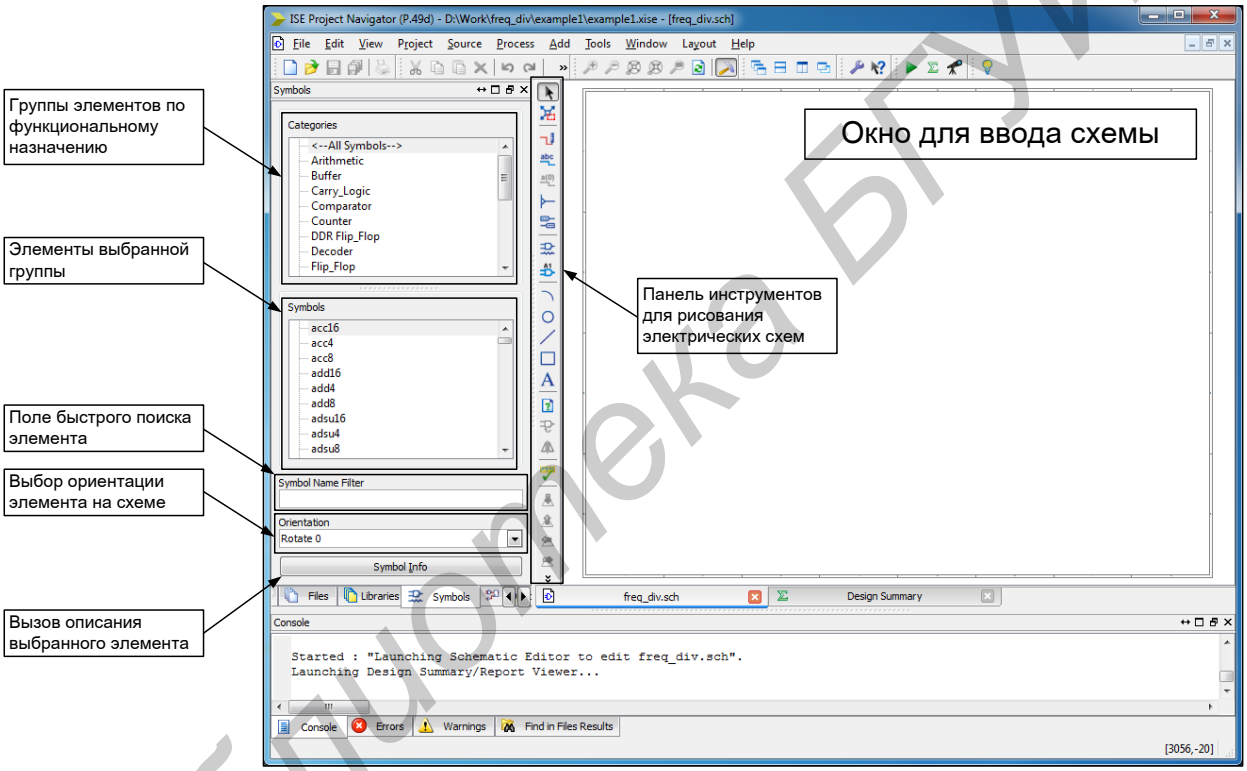


Рис. 1.16. Схемный редактор *Schematic Editor*

Большинство кнопок панелей инструментов редактора схем интуитивно понятны. Для ввода электрических схем используются следующие (по порядку элементов в панели сверху вниз):

– указатель (Esc);

– увеличение элементов схемы (*Zoom Select*) (CTRL+колесо прокрутки мыши);

– добавить цепь/шину (*Add Wire*) (CTRL+W);

– добавить имя цепи (*Add Net Name*) (CTRL+D);

– переименовать выбранную шину (*Rename Selected Bus*) (ALT+E,L);

– добавить ввод в шину (*Add Bus Tap*) (CTRL+B);
– добавить внешний порт ввода/вывода (*Add I/O Marker*) (CTRL+G);
– добавить элемент (*Add Symbol*);
– добавить позиционное обозначение (*Add Instance Name*) (CTRL+J).

Для ввода элементов используются библиотеки элементов, расположенные в окне **Categories**. Элементы библиотек сгруппированы (например, триггеры, счетчики, аккумуляторы, дешифраторы, логические элементы и т. д.). После выбора конкретной группы элементов в окне **Symbols** выводится список доступных элементов. Для быстрого поиска необходимого элемента используется поле **Symbol Name Filter**, в которое вписывается название элемента или его первые буквы.

Используя стандартные библиотечные элементы, создадим делитель частоты, который имеет входную частоту 50 МГц (задается тактовым генератором на отладочной плате ***Nexys 2***) и выходную частоту 1000 Гц.

Для выполнения этого задания нам понадобятся, например, следующие элементы: входной и выходной буферные элементы (**IBUF** и **OBUF**), 16-битный двоичный счетчик, элементы логической «1» (**VCC**) и логического «0» (**GND**). В качестве двоичного счетчика мы можем использовать из имеющихся библиотечных элементов следующие типы: **cb16ce** – 16-разрядный счетчик с асинхронным сбросом; **cb16cle** – 16-разрядный счетчик с предустановкой и асинхронным сбросом; **cb16cled** – 16-разрядный реверсивный счетчик с предустановкой и асинхронным сбросом; **cb16re** – 16-разрядный счетчик с синхронным сбросом.

Для примера используем счетчик **cb16cle** (рис. 1.17). Он имеет следующие входы и выходы:

D[15:0] – входы предустановки начального значения счетчика;

L – вход разрешения загрузки значений на выводах D[15:0] в счетчик;

CE – вход разрешения счета;

C – вход тактирования счетчика;

CLR – вход сброса;

Q[15:0] – выходы разрядов счетчика;

TC – выход переполнения счета (для каскадирования нескольких счетчиков);

CEO – выход разрешения счета (для каскадирования нескольких счетчиков).

Более подробную информацию о работе того или иного элемента можно получить, вызвав описание элемента кнопкой **Symbol Info**.

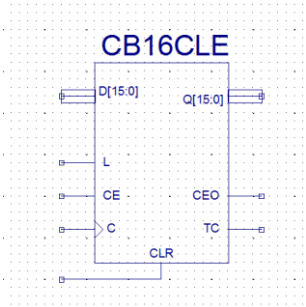


Рис. 1.17. 16-разрядный счетчик
с предустановкой и асинхронным сбросом

Используем включение данного счетчика в режиме работы с предварительной установкой начального значения счета. В этом случае выходной сигнал необходимо снимать с выхода *TC* (*Terminal Counter*) счетчика. Для получения необходимой частоты мы должны иметь следующий коэффициент деления:

$$K_{\text{дел}} = \frac{F_{\text{вх}}}{F_{\text{вых}}} = \frac{50 \cdot 10^6 \text{ Гц}}{1000 \text{ Гц}} = 50\,000 .$$

Имеющийся счетчик позволяет получить данный коэффициент деления, однако в силу принципа своей работы на выходе *TC* мы будем иметь кратковременные импульсы, а не меандр с частотой 1000 Гц. Для устранения этого недостатка используем счетный *T*-триггер. Ввиду того, что *T*-триггер является делителем частоты на 2, суммарный коэффициент деления будет состоять из коэффициента деления счетчика и коэффициента деления триггера:

$$K_{\text{дел}} = K_{\text{сч}} \cdot K_{\text{тр}} = K_{\text{сч}} \cdot 2 \ .$$

Для нашего случая требуемый коэффициент деления счетчика составит 25 000. Так как выбранный счетчик является суммирующим, то начальное значение счета необходимо искать из выражения

$$K = M - K_{\text{сч}} ,$$

где $M = 2^N$ – модуль счета; N – разрядность двоичного счетчика.

Подставляя необходимые значения, получим начальное значение счетчика, равное $2^{16} - 25\,000 = 65\,536 - 25\,000 = 40\,536$, или в двоичной системе счисления «1001111001011000».

Определив начальные параметры для работы делителя частоты, соберем схему, представленную на рис. 1.18.

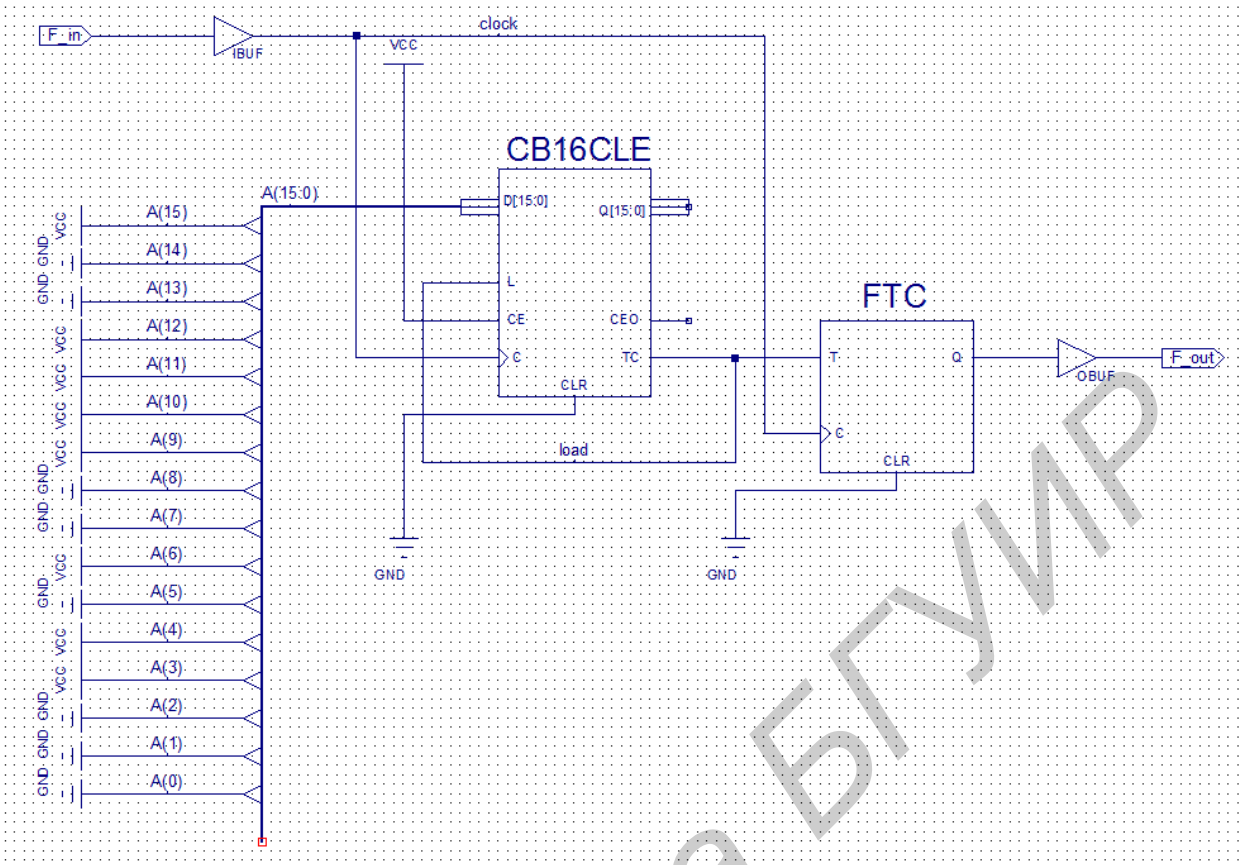


Рис. 1.18. Схема двоичного делителя частоты

Для обозначения имен шин необходимо учитывать некоторые правила. Шины в редакторе схем обозначаются следующим образом:

имя_шины(старший бит:младший бит),

например: **A(15:0)** – 16-разрядная шина.

Проводник в шине обозначается следующим образом:

имя шины(бит),

например: **A(2)** – 2-й бит шины **A.**

Схему после ее создания необходимо запустить на синтез. Параметры синтеза устанавливаются в меню, вызываемом нажатием правой клавиши мыши на процессе **Synthesize**. Здесь, в частности, можно выбрать, по какой оптимизации будет осуществляться синтез (по максимальной скорости или минимальной площади) (рис. 1.19).

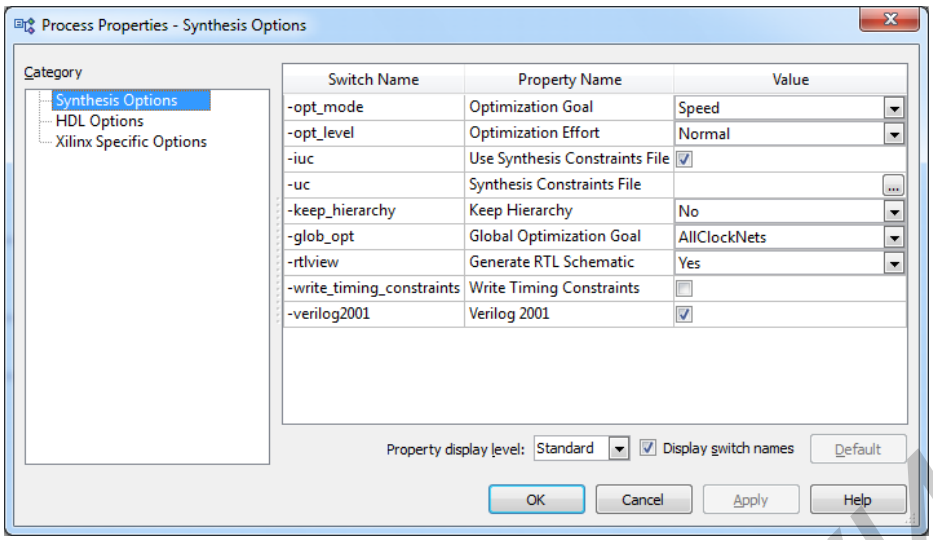


Рис. 1.19. Окно опций синтеза

Для запуска синтеза в окне процессов дважды щелкаем левой клавишей мыши по **Synthesize**. При успешном завершении процесса синтеза в окне сообщений должно появиться уведомление «**Process "Synthesize – XST" completed successfully**». В противном случае требуется устранить имеющиеся ошибки и повторить синтез.

Далее можно произвести моделирование работы схемы в каком-либо пакете (например *Modelsim* или *ISim*), а также выполнить трассировку выводов полученного устройства к физическим выводам кристалла ПЛИС. Об этом будет сказано далее.

Описание работы цифрового устройства при помощи языка высокого уровня

Рассмотрим реализацию той же задачи (делитель частоты с 50 МГц до 1 кГц) с использованием языка высокого уровня – VHDL. Для этого при создании нового модуля проекта (см. рис. 1.15) необходимо выбрать пункт **VHDL Module**. После ввода названия модуля и нажатия кнопки **Далее** (**Next**) в окне определения сигналов ввода/вывода можно описать интерфейс модуля (рис. 1.20).

После нажатия кнопки **Далее** (**Next**) появляется еще одно окно, в котором представлена суммарная информация о будущем модуле (рис. 1.21).

Если допущена какая-либо ошибка при вводе параметров модуля, можно вернуться назад и скорректировать параметры, если все правильно, то после нажатия кнопки **Готово** (**Finish**) будет сгенерирован VHDL файл-«пустышка», в котором содержится описание интерфейса создаваемого модуля и отсутствует только описание процесса или процессов, которые описывают работу устройства:

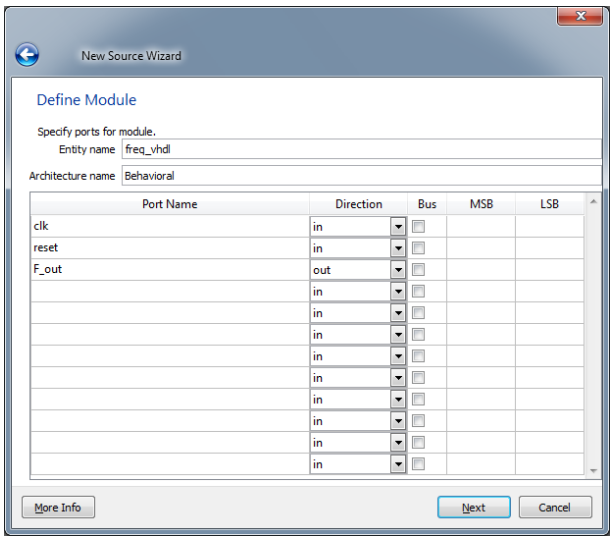


Рис. 1.20. Описание сигналов ввода/вывода

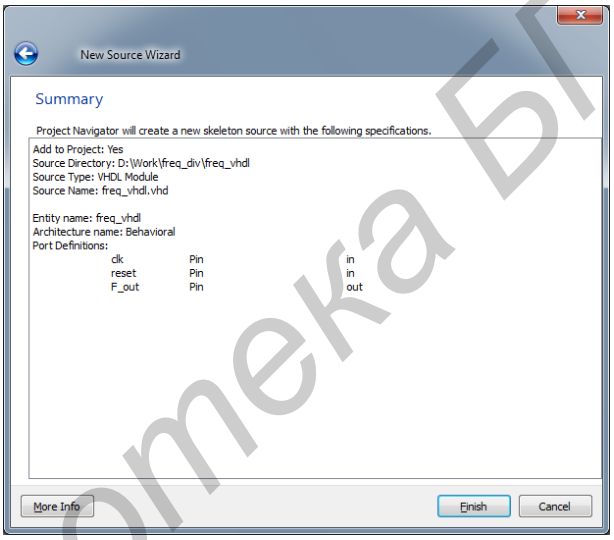


Рис. 1.21. Окно сведений о создаваемом модуле

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity freq_vhdl is
   Port (clk :          in  STD_LOGIC;
         reset :        in  STD_LOGIC;
         F_out :        out  STD_LOGIC);
end freq_vhdl;

architecture Behavioral of freq_vhdl is
begin

end Behavioral;
```

Для облегчения написания кода программы на языке VHDL в системе предусмотрена помощь (в главном меню **Edit > Language Templates…**) с приведением примеров языкового описания большинства компонентов схем (рис. 1.22).

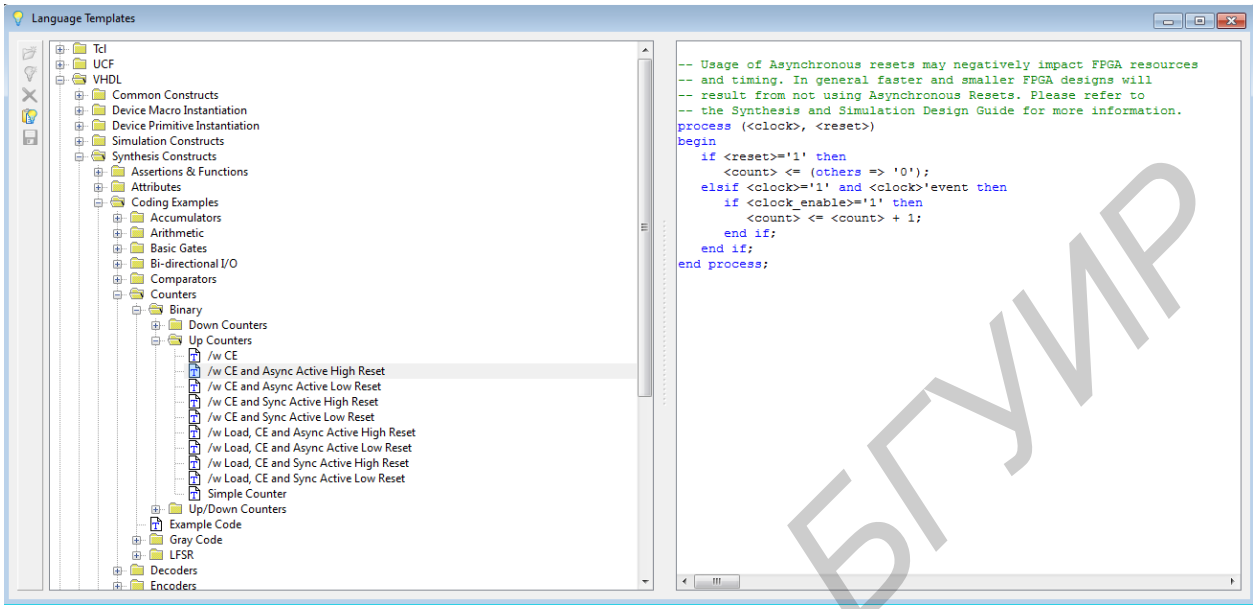


Рис. 1.22. Окно *Language Templates*

Скопировав шаблон счетчика, представленного на рис. 1.22 справа, и отредактировав его текст под условие задачи, получим следующее описание разрабатываемого устройства на языке VHDL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity freq_vhdl is
   Port ( clock_en : in STD_LOGIC;
          clk : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          F_out : out  STD_LOGIC);
end freq_vhdl;

architecture Behavioral of freq_vhdl is
        signal  counter:       std_logic_vector (15 downto 0);
        signal  F_out_temp:  std_logic;

begin

process (clk, reset)
begin
```

```
if reset = '1' then
counter <= (others => '0');
elsif clk = '1' and clk'event then
if clock_en = '1' then
counter <= counter + 1;
end if;
if counter = "0110000110101000" then
                        counter <= (others => '0');
                        F_out_temp <= not F_out_temp;
                end if;
end if;
        F_out <= F_out_temp;
end process;

end Behavioral;
```

Как видно из представленного кода, в интерфейсе устройства появился новый вывод ***clock_en***, который отвечает за разрешение счета.

Проверить отсутствие грубых ошибок можно при помощи опции **Check Syntax** в окне процессов (рис. 1.23).

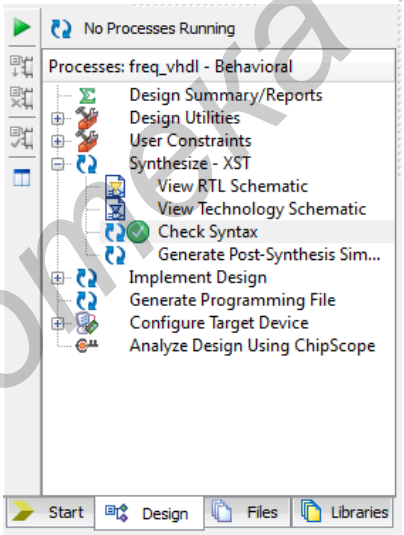


Рис. 1.23. Проверка синтаксиса VHDL-модуля

При наличии ошибок их список появится в ***окне сообщений*** (рис. 1.24), из которого путем двойного щелчка левой кнопкой мыши по выбранной ошибке можно переместиться к соответствующей точке описываемого модуля.

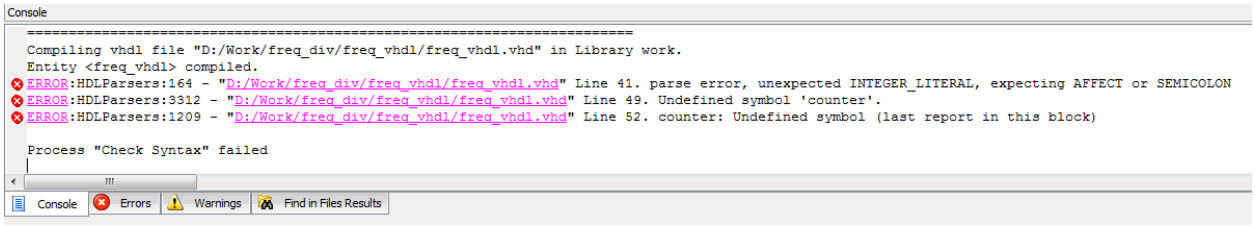


Рис. 1.24. Окно сообщений при наличии ошибок в коде VHDL-описания

Следует отметить, что отсутствие ошибок не всегда гарантирует возможность получения работающего проекта. В случае неожиданного отсутствия каких-либо частей проекта после компиляции и укладки на кристалл необходимо проанализировать предупреждения в ***окне сообщений***.

Далее из VHDL-описания устройства можно получить библиотечный компонент, который можно будет использовать при схемотехническом описании. Для этого в окне процессов на вкладке **Design Utilities** нужно дважды щелкнуть на опцию **Create Schematic Symbol** (рис. 1.25).

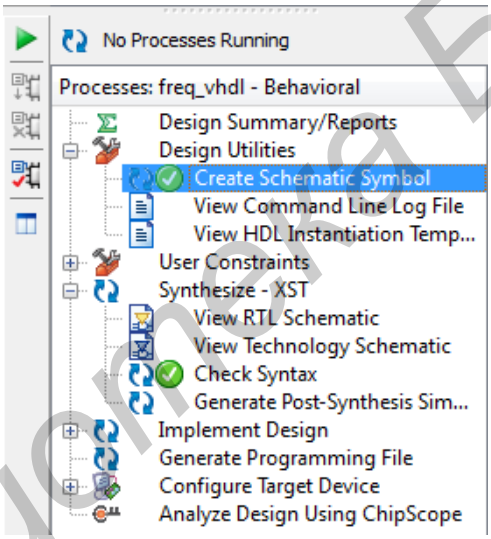


Рис. 1.25. Создание библиотечного элемента для устройства

После этого в редакторе схем в библиотеке элементов будет доступен созданный VHDL-модуль, с которым можно выполнять те же действия, что и с другими элементами схемы.

По окончании описания модуля (схемотехнического или на VHDL) его надо запустить на синтез. Перед запуском можно установить параметры синтеза, вызвав правой клавишей мыши на процессе **Synthesize** контекстное меню и выбрав в нем пункт **Properties…**. Для запуска синтеза в окне процессов дважды щелкаем левой клавишей мыши по процессу **Synthesize**. При успешном завершении процесса синтеза в окне сообщений появится предварительная оценка максимальной тактовой частоты проекта (рис. 1.26).

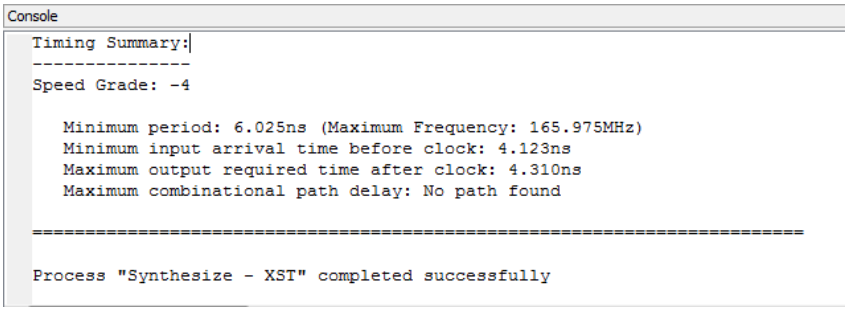


Рис. 1.26. Окно сообщений

Компиляция проекта и конфигурирование кристалла ПЛИС

Когда прошли этапы синтеза и моделирования устройства, можно приступить непосредственно к назначению выводам разработанного цифрового устройства физических выводов кристалла ПЛИС. Для этого действия предназначена утилита **Xilinx PlanAhead**. Она позволяет произвести назначение входных и выходных сигналов устройства по выводам ПЛИС. Для ее вызова необходимо запустить опцию **I/O Pin Planning (PlanAhead) – Post-Synthesis** в окне процессов (рис. 1.27). При ее запуске система автоматически подключает данные из проекта.

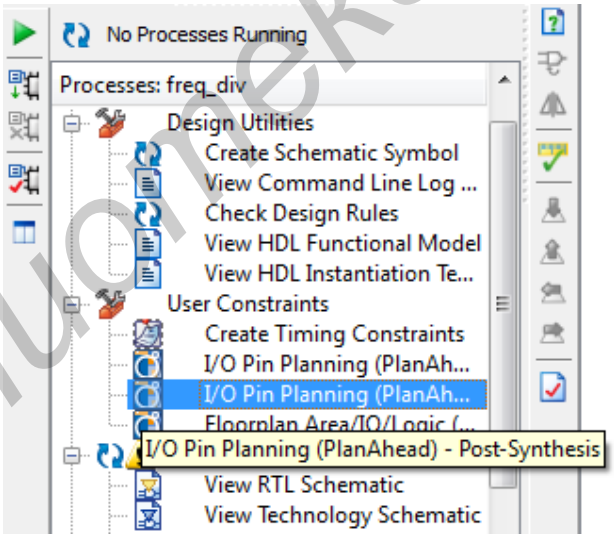


Рис. 1.27. Запуск утилиты **Xilinx PlanAhead**

В случае запуска утилиты **Xilinx PlanAhead** при помощи ярлыка на рабочем столе или через меню ***Пуск*** необходимо в появившемся окне выбрать пункт ***Create New Project***, указать место расположения проекта и дать ему имя. После нажатия кнопки *Далее*, в следующем окне установить флажок напротив «***Post-synthesis Project***» и нажать кнопку *Далее*. В появившемся окне необходимо добавить файл с описанием синтезированного проекта. Для этого в диалоговом окне «***Add Files…***» выбрать в директории с проектом файл с именем

«имя_проекта***.ngc***», после чего продолжить работу с менеджером проекта. После указания типа используемого кристалла ПЛИС откроется главное окно утилиты **Xilinx PlanAhead**. В меню ***Project Manager*** слева окна необходимо открыть результат синтеза кнопкой «***Open Synthesized Design***». Дальнейшие действия не отличаются от действий при запуске утилиты из среды разработки **Xilinx ISE Design Suite**.

Главное окно утилиты **Xilinx PlanAhead** изображено на рис. 1.28. На нем показан выбранный тип корпуса ПЛИС, а также все сигнальные цепи проекта.

На вкладке ***I/O Ports*** отображаются все сигналы, которым можно присвоить какие-либо физические выводы ПЛИС. Для назначения конкретного сигнала выводу кристалла необходимо выбрать интересующий сигнал и, удерживая левую клавишу мыши, перетащить его на нужный вывод микросхемы. Альтернативным способом назначения вывода кристалла сигналу является запись информации в поле ***Site***. Помимо расположения выводов данная утилита позволяет определить и другие параметры, такие, как тип логики (I/O STD), нагрузочная способность (Drive Strength), подключение подтягивающих резисторов (Pull Type) и др.

Для схемы, представленной на рис. 1.18, назначим выводы разработанного устройства следующим образом:

F_in – B8; F_out – L15.

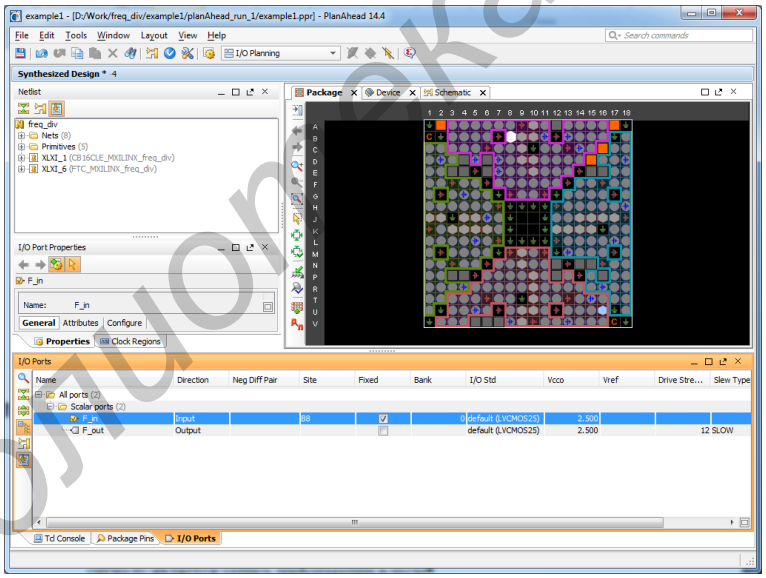


Рис. 1.28. Главное окно утилиты **Xilinx PlanAhead**

После окончания расстановки выводов сохраняем результаты работы. В итоге в окне исходных модулей должен появиться новый текстовый файл с расширением ***.ucf**, в данном случае файл **freq_div.ucf**, который можно редактировать вручную без вызова программы **Xilinx PlanAhead** при вызове опции **Edit Constraints** (рис. 1.29).

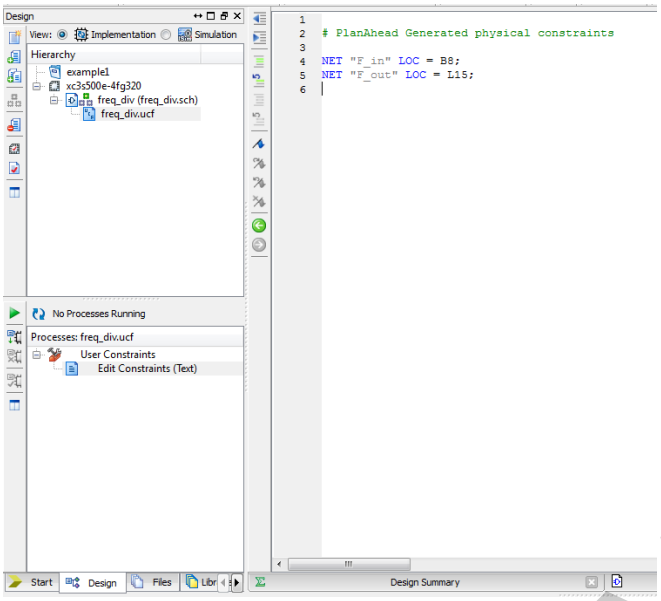


Рис. 1.29. Файл описания констант

Следующим этапом в создании устройства является этап размещения элементов спроектированного устройства и трассировки связей между ними на кристалле. Для этого необходимо запустить процесс **Implement Design**. В его настройках можно указать, каким образом осуществить оптимизацию размещения на кристалле.

Система проектирования построена таким образом, что если запускается на исполнение какой-нибудь этап, то предыдущие будут автоматически выполнены. Для завершения процедуры формирования конфигурационного файла для ПЛИС достаточно запустить процесс **Generate Programming File** в окне процессов. Результатом работы этого процесса будет ***.bit**-файл конфигурации, который будет располагаться в корневом каталоге проекта.

После этих процедур можно приступить к завершающему этапу – конфигурированию кристалла ПЛИС. Для конфигурирования кристалла можно воспользоваться возможностями самой среды разработки **Xilinx ISE Design Suite**, запустив на исполнение процесс **Configure Target Device** в случае, если ПЛИС подключена посредством стандартного кабеля по JTAG-интерфейсу. В нашем случае кристалл ПЛИС и ПЗУ подключаются для программирования к ПК через преобразователь USB/JTAG, размещенный на самой отладочной плате. Для программирования необходимо задействовать специальную утилиту **Digilent Adept**, поставляемую производителем отладочной платы Digilent Inc. Ее можно запустить через меню **Пуск>Программы> Digilent>Adept>Adept** операционной системы. Внешний вид окна данной программы показан на рис. 1.30.

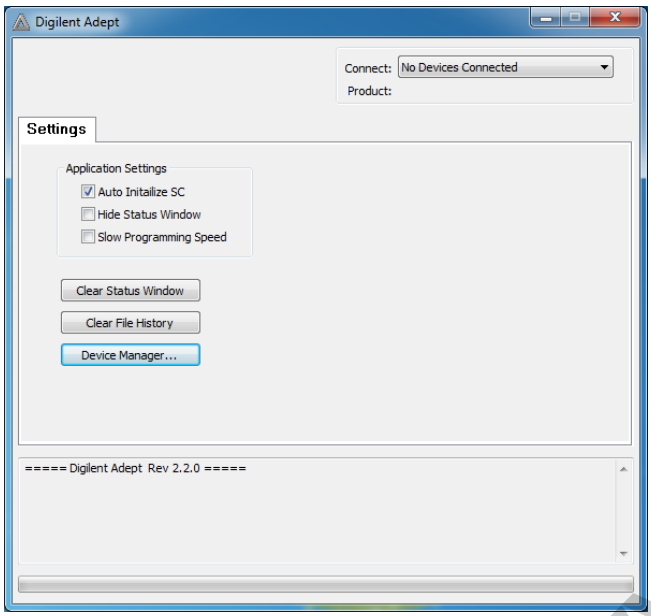


Рис. 1.30. Окно программы **Digilent Adept**

При запуске данная программа автоматически определяет все подключенные устройства. Чтобы определить подключенные устройства в ручном режиме, необходимо выбрать устройство на выпадающем списке возле графы **Connect**. В нашем случае это будет отладочная плата ***Nexys2*** (рис. 1.31).

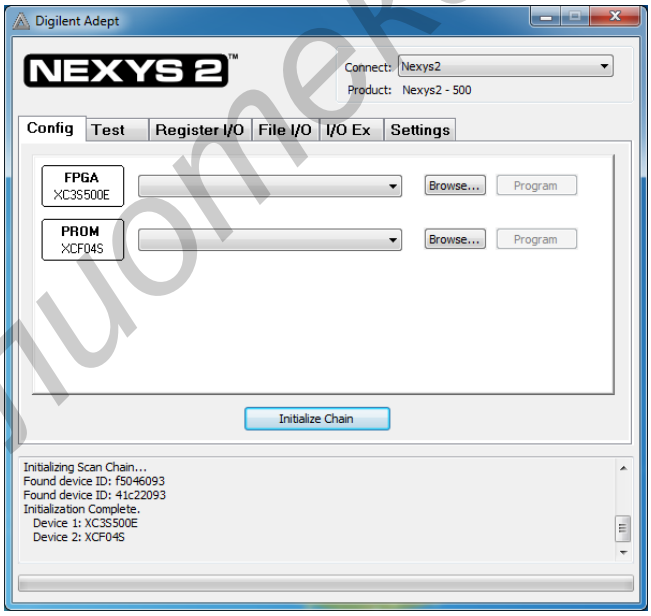


Рис. 1.31. Подключенные устройства для программирования

Как видно из рис. 1.31, программа обнаруживает два устройства: **FPGA XC3S500E** и **PROM XCF04S**. Первое из этих устройств является электрически зависимой памятью кристалла ПЛИС, второе – внешняя микросхема памяти. Конфигурировать необходимо память кристалла ПЛИС.

Для того чтобы сконфигурировать ПЛИС, необходимо указать ***.bit** -файл конфигурации устройства в графе рядом со значком ПЛИС. Путь к файлу конфигурации указывается путем вызова проводника посредством кнопки **Browse…**. После этого остается только нажать кнопку **Program** для программирования. В результате всего этого должна появиться надпись **Programming Successful** в строке состояния (рис. 1.32), а на указанном при конфигурировании выводе отладочной платы – сигнал с заданной частотой.

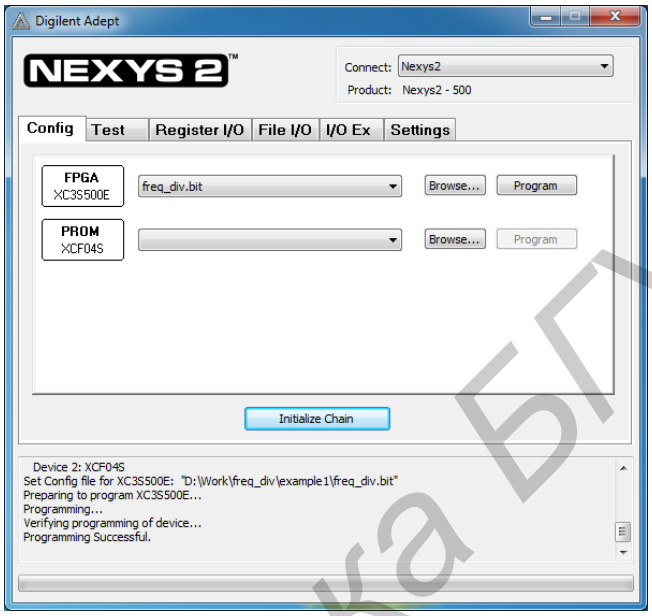


Рис. 1.32. Успешное завершение программирования

Лабораторное задание

Сформировать сигнал в виде меандра на указанных преподавателем выводах ПЛИС с заданными частотами. Устройство формирования сигнала реализовать двумя способами: при помощи схемотехнического описания *Schematic* и при помощи описания на языке VHDL. Проверить работу устройства в среде моделирования *ISim* и на отладочной плате.

Дополнительное задание – реализовать генератор псевдослучайной последовательности (ПСП) на основе регистра сдвига с обратными связями и тактовой частотой, заданной преподавателем в основном задании. Генераторный полином ПСП получить у преподавателя.

Порядок выполнения работы

1. Изучить порядок работы со средой разработки цифровых устройств на ПЛИС **Xilinx ISE Design Suite** и средой моделирования **ISim.**

2. Получить у преподавателя задание на лабораторную работу. Возможные варианты заданий представлены в табл. 1.10.

Таблица 1.10

Номер варианта	Частота F_1, Гц	Частота F_2, Гц	Вывод F_1	Вывод F_2
1	0,5	10000	JA1, LED0	JB4
2	1	1000	JA2, LED1	JB3
3	1,5	150	JA3, LED2	JB2
4	2	5000	JA4, LED3	JB1
…				

3. Произвести расчет делителя частоты на указанную частоту(ы), учитывая, что входная тактовая частота составляет 50 МГц.

4. Зная требуемый коэффициент деления, в редакторе *Schematic* реализовать устройство деления частоты на основе последовательностных цифровых устройств.

5. Произвести синтез собранного устройства и промоделировать его работу в среде **ISim**.

6. Создать делитель частоты с аналогичными параметрами, используя поведенческое описание работы устройства на языке VHDL.

7. Изучить описание отладочной платы ***Nexus2*** и произвести конфигурирование выводов кристалла ПЛИС в соответствии с заданием в **Xilinx PlanAhead**. Произвести формирование файла конфигурации ПЛИС.

8. Используя программное обеспечение **Digilent Adept**, загрузить полученный ***.bit**-файл конфигурации ПЛИС в энергозависимую память кристалла FPGA и убедиться в работоспособности разработанного устройства.

9. Используя осциллограф, измерить временные параметры формируемого сигнала.

10. Оформить отчет по лабораторной работе.

Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Лабораторное задание.
4. Схема реализованного устройства формирования сигнала в редакторе *Schematic*.
5. Код программы описания работы устройства на языке VHDL.
6. Результаты моделирования в среде **ISim**.
7. Результаты работы устройства на отладочной плате (осциллограммы полученных сигналов).
8. Выводы.

Контрольные вопросы и задания

1. Дайте определение ПЛИС и опишите архитектуру ПЛИС семейства FPGA Spartan-3E.
2. Приведите структуру конфигурируемой логической ячейки ПЛИС.
3. Приведите структуру блока ввода/вывода.
4. Расскажите порядок работы со средой проектирования устройств на ПЛИС **Xilinx ISE Design Suite**.
5. Перечислите процессы, которые необходимо осуществить при проектировании устройства на ПЛИС, и дайте их характеристику.
6. Что представляет собой утилита **Xilinx PlanAhead** и как при помощи ее осуществляется привязка входных и выходных сигналов к физическим выводам кристалла ПЛИС?
7. Поясните, что такое JTAG-интерфейс и каким образом через него осуществляется программирование и отладка устройств?

1.4. *Лабораторная работа №2*. Реализация комбинационных устройств на ПЛИС

Цель работы

1. Углубление и закрепление теоретических знаний по схемотехническому проектированию цифровых устройств на логических элементах и описанию их работы на языке высокого уровня.

2. Формирование практических навыков создания цифровых устройств на основе ПЛИС путем схемотехнического описания и описания работы устройства на языке VHDL, а также компьютерного моделирования их работы.

3. Приобретение практических навыков работы с реальными устройствами на базе ПЛИС и контрольно-измерительными приборами.

Краткие теоретические сведения

Комбинационными устройствами, или цифровыми автоматами без памяти, называют устройства, сигналы на выходе которых зависят только от комбинации сигналов на входе устройства. Внутреннее состояние устройства при этом не влияет на формируемые выходные сигналы.

Как известно из предыдущих курсов, к комбинационным устройствам относят следующие типы цифровых устройств: преобразователи кодов, суммирующие и вычитающие устройства, мультиплексоры и демультиплексоры, шифраторы и дешифраторы.

Преобразователи кодов являются комбинационными логическими устройствами и предназначены для изменения вида кодирования (преобразования) информации в цифровых устройствах. В общем случае преобразователи осуществляют преобразование информации из одного кода в другой и используются для шифрации и дешифрации цифровой информации.

Для кодирования информации в различных устройствах используются двоичные коды, наиболее часто применяемые из них приведены в табл. 1.11.

Суммирующие и вычитающие устройства – это комбинационные логические устройства, выполняющие арифметическое сложение либо вычитание чисел. Сумматоры и вычитающие устройства широко используются в цифровой технике и применяются как самостоятельно, так и в составе арифметико-логических устройств. Суммирующие и вычитающие устройства подразделяются на неполные (полусумматоры и полувычитатели) и полные (полный сумматор и полный вычитатель).

К группе арифметических устройств относятся также и *перемножители* двоичных чисел. Они могут быть реализованы как схемными решениями, так и готовыми блоками в составе ПЛИС. В используемой ПЛИС *Spartan-3E* имеются аппаратные перемножители двух 18-разрядных чисел.

Таблица 1.11

Десятичное число	Код 8421	Обратный код	Дополнит. код	Код Грея	Код с изб. 3	Код 7421	Код 2421
0	0000	1111	0000	0000	0011	0000	0000
1	0001	1110	1111	0001	0100	0001	0001
2	0010	1101	1110	0011	0101	0010	0010
3	0011	1100	1101	0010	0110	0011	0011
4	0100	1011	1100	0110	0111	0100	0100
5	0101	1010	1011	0111	1000	0101	1011
6	0110	1001	1010	0101	1001	0110	1100
7	0111	1000	1001	0100	1010	1000	1101
8	1000	0111	1000	1100	1011	1001	1110
9	1001	0110	0111	1101	1100	1010	1111
10	1010	0101	0110	1111	1101	–	–
11	1011	0100	0101	1110	1110	–	–
12	1100	0011	0100	1010	1111	–	–
13	1101	0010	0011	1011	10000	–	–
14	1110	0001	0010	1001	10001	–	–
15	1111	0000	0001	1000	10010	–	–

Мультиплексор – это комбинационное устройство, которое по заданному адресному двоичному коду осуществляет выбор одного из информационных каналов и подключает его к своему выходу.

Демультиплексор осуществляет передачу сигналов с одного информационного входа на один из выходов, имеющий заданный n-разрядный адресный код. В общем случае число выходов равно 2^n и определяется количеством адресных входов n.

Шифратор предназначен для преобразования входных сигналов в выходной n-разрядный двоичный код. Шифратор также называют кодером и применяют в устройствах ввода информации в цифровых системах. Шифратор называется полным, если число входов $m = 2^n$, а число выходов равно n (n – разрядность двоичного кода).

Дешифратор преобразует входной n-разрядный двоичный код в кодированные выходные сигналы. Входной код имеет меньшее число разрядов, чем выходной. Дешифратор называется полным, если при n входах имеется $m = 2^n$ выходов, где n – разрядность двоичного кода.

Рассмотрим пример реализации преобразователя двоичного кода 8421 в двоичный код 7421 на языке VHDL для 4-битного входного числа.

Устройство в этом случае должно иметь четыре входа и четыре выхода (см. табл. 1.11), кроме этого, можно предусмотреть вход разрешения. Интерфейс разрабатываемого комбинационного устройства можно описать следующим образом:

```
entity test is
        Port (   Data_in : in  STD_LOGIC_VECTOR (3 downto 0);
                 EN : in  STD_LOGIC;
```

```
                Data_out : out  STD_LOGIC_VECTOR (3 downto 0));
end test;
```

В данном примере для реализации устройства достаточно использовать один оператор процесса, который будет запускаться сигналом разрешения *EN* либо изменением входных данных на линиях *Data_in*. Преобразование входного кода в выходной можно осуществить посредством использования оператора **case**. Текст кода, реализующий данный преобразователь, приведен ниже.

```
architecture Behavioral of test is
begin
        process (EN, Data_in)
        begin
                if EN = '1' then
                        case (Data_in) is
                                when "0000" => Data_out <= "0000";
                                when "0001" => Data_out <= "0001";
                                when "0010" => Data_out <= "0010";
                                when "0011" => Data_out <= "0011";
                                when "0100" => Data_out <= "0100";
                                when "0101" => Data_out <= "0101";
                                when "0110" => Data_out <= "0110";
                                when "0111" => Data_out <= "1000";
                                when "1000" => Data_out <= "1001";
                                when "1001" => Data_out <= "1010";
                                when others => Data_out <= "0000";
                        end case;
                else Data_out <= "0000";
                end if;
        end process;
end Behavioral;
```

Лабораторное задание

Возможные варианты заданий:

1. Реализовать на основе ПЛИС устройство для выполнения арифметических операций с использованием встроенных аппаратных блоков ПЛИС (умножителей, таблиц соответствия – LUT, ячеек ОЗУ). Ввод информации для вычисления производить при помощи ползунковых переключателей, результаты выводить на четырехразрядный семисегментный индикатор. Описание устройства на верхнем уровне выполнить в редакторе *Schematic*, остальные необходимые блоки описать на языке VHDL. Проверить работу устройства на отладочной плате. Один из возможных вариантов структурной схемы предлагаемого устройства представлен на рис. 1.33.

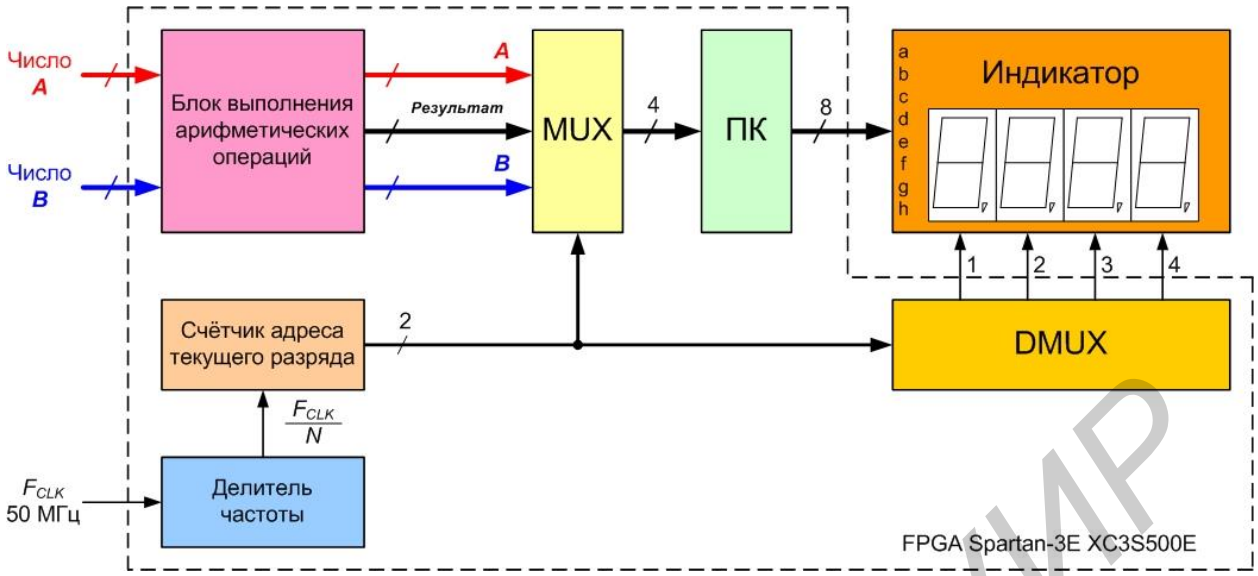


Рис. 1.33. Структурная схема арифметического устройства

2. Реализовать на основе ПЛИС устройство, которое находит заданную двоичную последовательность (или последовательности) во входном сигнале, формируемом генератором ПСП. Искомая последовательность (либо ее номер) задается ползунковыми переключателями. Разрядность последовательности и их число указывает преподаватель. Организовать подсчет количества найденных последовательностей и индикацию их числа на семисегментном индикаторе. Генератор ПСП использовать из предыдущей работы. Вариант структуры предлагаемого устройства показан на рис. 1.34.

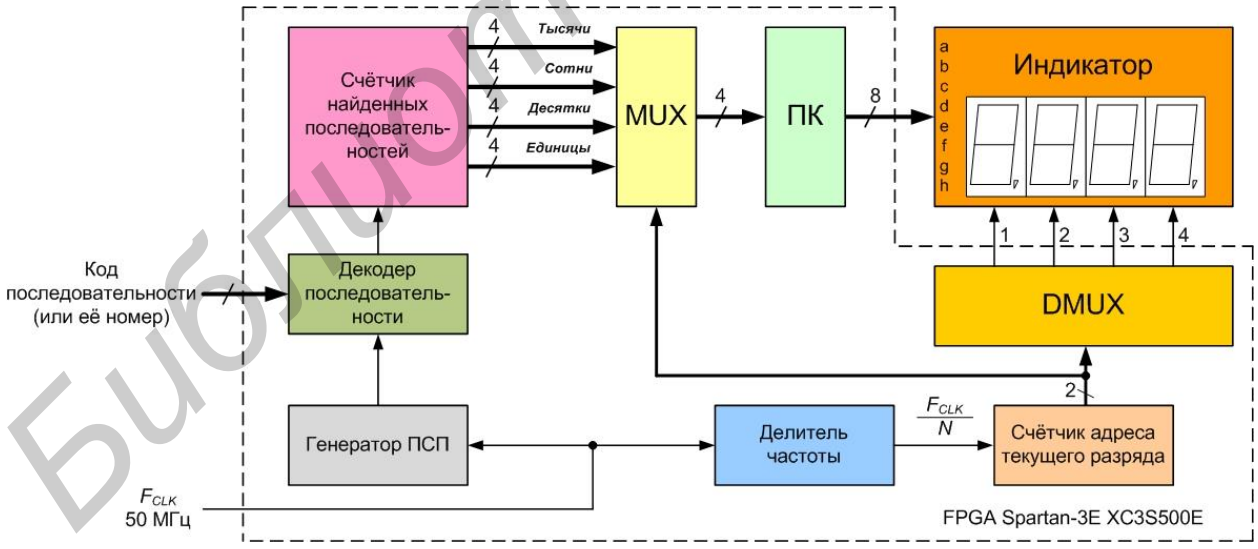


Рис. 1.34. Структура устройства

Порядок выполнения работы

1. Ознакомиться со встроенными аппаратными блоками ПЛИС. Повторить принципы работы семисегментного индикатора в динамическом режиме, повторить теоретические сведения о работе шифраторов, дешифраторов, мультиплексоров, демультиплексоров, преобразователей кодов, арифметических устройств.
2. Получить у преподавателя задание на лабораторную работу.
3. Реализовать предложенное устройство в редакторе *Schematic* и на языке VHDL.
4. Подготовить и загрузить в кристалл ПЛИС файл конфигурации.
5. Убедиться в работоспособности полученного устройства и произвести измерение необходимых сигналов при помощи цифрового осциллографа.
6. Оформить отчет по лабораторной работе.

Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Лабораторное задание.
4. Схема реализованного устройства в редакторе *Schematic*.
5. Код программы описания работы отдельных блоков на языке VHDL.
6. Результаты моделирования в среде **ISim**.
7. Результаты работы устройства на отладочной плате (осциллограммы сигналов).
8. Выводы.

Контрольные вопросы и задания

1. Какие устройства относят к комбинационным?
2. Назовите основные способы описания цифровых устройств, реализуемых на ПЛИС.
3. Дайте характеристику языка VHDL.
4. Назовите основные операторы языка VHDL, позволяющие производить выбор одного из нескольких условий, и дайте их характеристику.
5. В чем состоит отличие последовательных и параллельных операторов VHDL?
6. Что такое интерфейс и что такое архитектура объекта в VHDL?

1.5. *Лабораторная работа №3*. Формирование аналоговых сигналов на ПЛИС

Цель работы

1. Углубление и закрепление теоретических знаний по схемотехническому проектированию цифроаналоговых устройств.
2. Формирование практических навыков создания цифроаналоговых устройств на основе ПЛИС.
3. Приобретение практических навыков работы с реальными устройствами на базе ПЛИС и контрольно-измерительными приборами.

Краткие теоретические сведения

Формирование аналоговых сигналов при помощи цифровых устройств можно производить различными методами. Наиболее распространены следующие: формирование сигналов с помощью широтно-импульсной модуляции (ШИМ) и формирование при помощи блоков цифроаналогового преобразования (ЦАП) методом прямого цифрового синтеза.

Генератор с прямым цифровым синтезом – это электронный прибор, предназначенный для синтеза сигналов произвольной формы и частоты из единственной опорной частоты, задаваемой генератором тактовых импульсов. Способ получения аналогового сигнала из его дискретных отсчетов называют прямым цифровым синтезом (англ. – *Direct Digital Synthesis*, *DDS*). Исходный сигнал оцифровывается или его отсчеты формируются цифровыми методами. Потом отсчеты сигнала заносятся в память устройства. Далее они последовательно считываются из памяти с тактовой частотой F_T и поступают в ЦАП. На его выходе ставят ФНЧ, который убирает ступеньки и высшие гармоники; в результате чего восстанавливается исходная форма сигнала. Общая структурная схема *DDS*-генератора представлена на рис. 1.35.

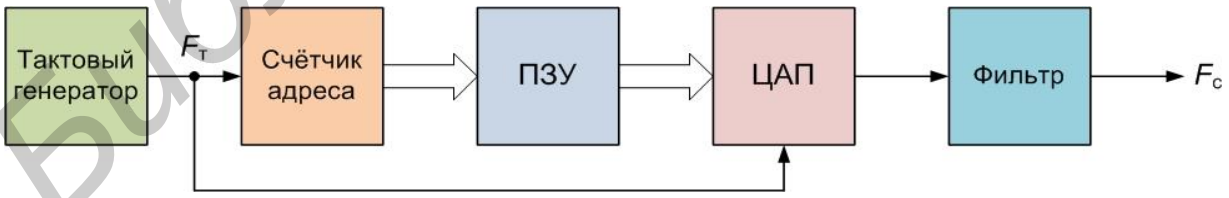


Рис. 1.35. Обобщенная структурная схема *DDS*-генератора

Вариант построения *DDS*-генератора, когда вычисление отсчета производится по некоторому полиному при помощи АЛУ (например тригонометрических функций sin и cos), в большинстве случаев достаточно сложен и будет иметь низкое быстродействие. Поэтому в настоящее время в основном приме-

няются способы формирования отсчетов сигнала табличными методами. Таблица отсчетов обычно хранится в ПЗУ. На адресные входы ПЗУ подается код, который является аргументом функции, а выходной код ПЗУ равен значению функции для данного аргумента. Аргумент функции (или его фаза в случае формирования гармонического сигнала) в отличие от значения функции меняется во времени линейно. Сформировать линейно изменяющуюся кодовую последовательность для формирования адреса (аргумента функции) текущего отсчета возможно при помощи обыкновенного двоичного счетчика. Изменять частоту формируемого сигнала можно путем изменения тактовой частоты F_T. Для ее изменения достаточно иметь делитель частоты с переменным коэффициентом деления. Схема простейшего *DDS*-генератора сигнала, имеющего возможность перестройки по частоте, показана на рис. 1.36.

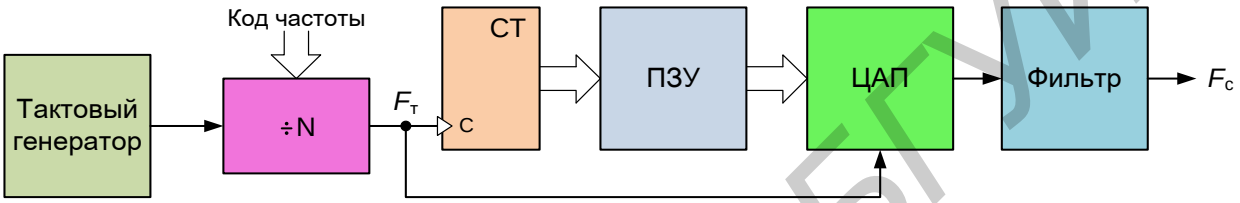


Рис. 1.36. Схема простейшего *DDS*-генератора

Такая структура *DDS* имеет очевидные недостатки, основным из которых является неудовлетворительная способность к перестройке по частоте. Действительно, поскольку тактовая частота испытывает деление на целое число, шаг перестройки будет переменным, причем, чем меньше коэффициент деления, тем больше относительная величина шага. Этот шаг будет недопустимо грубым при малых коэффициентах деления.

Кроме того, при перестройке выходной частоты будет меняться и частота дискретизации. Это затрудняет фильтрацию выходного сигнала, а также ведет к неоптимальному использованию скоростных характеристик ЦАП – они будут в полной мере использованы лишь на максимальной выходной частоте. Гораздо логичнее, независимо от выходной частоты, всегда работать на постоянной частоте дискретизации, близкой к максимальной для используемого ЦАП.

Все недостатки описанной выше структуры могут быть устранены путем замены адресного счетчика ПЗУ другим цифровым устройством – накапливающим сумматором. Накапливающий сумматор представляет собой регистр, который в каждом такте работы устройства перезагружается величиной, равной старому содержимому, плюс некоторая постоянная добавка (рис. 1.37). Как и в случае со счетчиком, содержимое регистра линейно увеличивается во времени, только теперь приращение не всегда является единичным, а зависит от величины постоянной добавки. Когда накапливающий сумматор используется для формирования кода фазы, его еще называют аккумулятором фазы. Выходной код аккумулятора фазы представляет собой код мгновенной фазы выходного сигнала. Постоянная добавка, которая используется при работе аккумулятора

фазы, представляет собой приращение фазы за один такт работы устройства. Чем быстрее изменяется фаза во времени, тем больше частота генерируемого сигнала. Поэтому значение приращения фазы фактически является кодом выходной частоты.

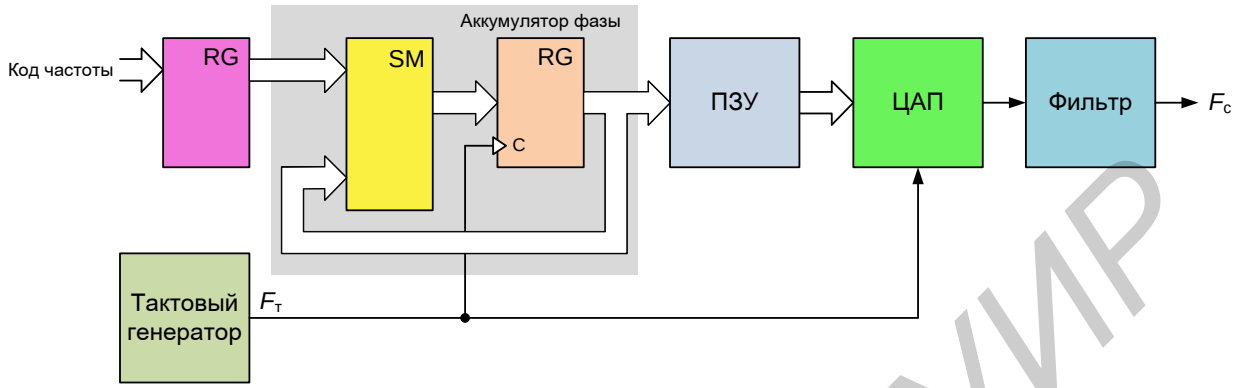


Рис. 1.37. *DDS*-генератор на основе накапливающего сумматора

Рассмотренная выше структура применяется во всех современных *DDS*. Объединение в одном чипе быстродействующего ЦАП и собственно *DDS* (так называемый полный *DDS* или *Complete DDS*) позволило получить весьма заманчивую альтернативу обычным синтезаторам на основе ФАПЧ. *DDS*, не имеющие встроенного ЦАП, иногда называют *Numerically Controlled Oscillator* (*NCO*), несмотря на то что *DDS* не содержит никаких генераторов.

Кроме интегрированного ЦАП, *DDS* могут иметь некоторые дополнительные цифровые блоки, выполняющие над сигналом различные дополнительные операции. Эти блоки обеспечивают большую функциональность и улучшенные пользовательские характеристики *DDS*.

DDS-генераторы, формирующие гармонический синусоидальный сигнал, также называют цифровым синтезатором частоты. Синтезаторы частот широко используются в различных радиоэлектронных системах в качестве задающего генератора в радиопередающих устройствах, в качестве перестраиваемого гетеродина в радиоприемных устройствах, генераторах сигналов стандартных частот и др.

Параметры синтезатора частоты очень важны для аппаратуры связи. Являясь сердцем системы настройки, синтезатор в основном определяет потребительские свойства конкретного аппарата. Как с технической, так и с экономической стороны *DDS* удовлетворяет большинству критериев идеального синтезатора частоты: простой, высокоинтегрированный, с малыми габаритами. Кроме того, многие параметры *DDS* программно-управляемые, что позволяет заложить в устройство новые возможности и делает *DDS* перспективными приборами.

Описание работы модуля цифроаналогового преобразования PmodDA2

Для преобразования сигнала, представленного в виде отдельных дискретных значений в цифровом коде, в аналоговый сигнал используется цифроаналоговый преобразователь. В данном случае ЦАП представляет собой дополнительный модуль расширения ***PmodDA2***, подключаемый к отладочной плате ***Nexys2***. ЦАП имеет два канала и выполнен на основе микросхем DAC121S101 и обеспечивает преобразование с разрядностью 12 бит. Структурная схема модуля представлена на рис. 1.38.

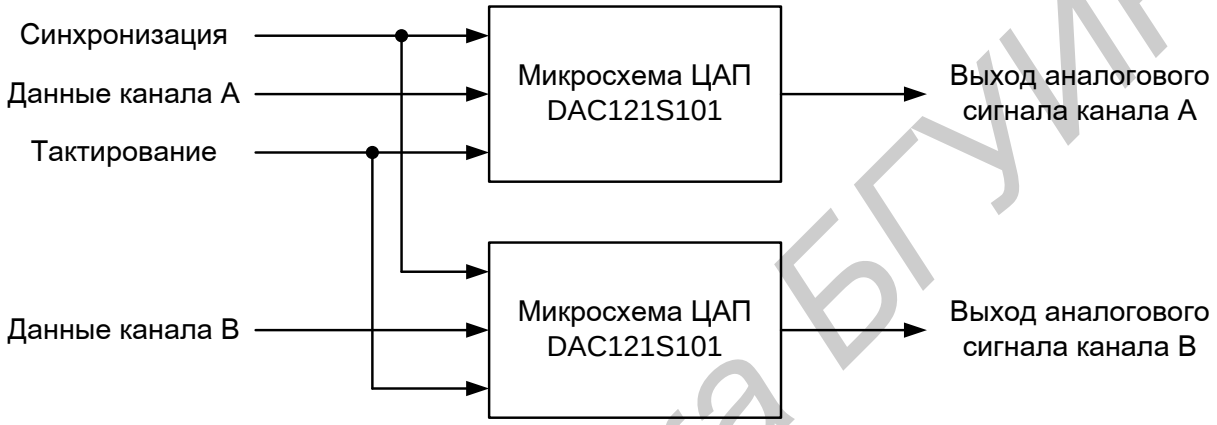


Рис. 1.38. Структура модуля ЦАП

Модуль ***PmodDA2*** содержит две микросхемы цифроаналогового преобразователя, которые имеют общие входы синхронизации и тактирования и раздельные входы данных. Вывод сформированного аналогового сигнала осуществляется на отдельный вывод выходного разъема.

Рассмотрим устройство и принцип функционирования микросхемы DAC121S101. Данная микросхема представляет собой цифроаналоговый преобразователь, построенный на принципе делителя напряжения. Внутренняя структура микросхемы показана на рис. 1.39.

В состав микросхемы входят следующие основные блоки: блок управления сбросом по включению питания (*POWER-ON RESET*), 12-разрядный регистр цифроаналогового преобразователя (*DAC REGISTER*), непосредственно сам ЦАП (*12-BIT DAC*), выходной буферный усилитель (*BUFFER*), блок контроля входных сигналов (*INPUT CONTROL LOGIC*), блок логики отключения питания (*POWER-DOWN CONTROL LOGIC*). В состав микросхемы также входят входные буферные элементы входов интерфейса SPI и ключи для коммутации режима работы аналогового выхода при отключении питания.

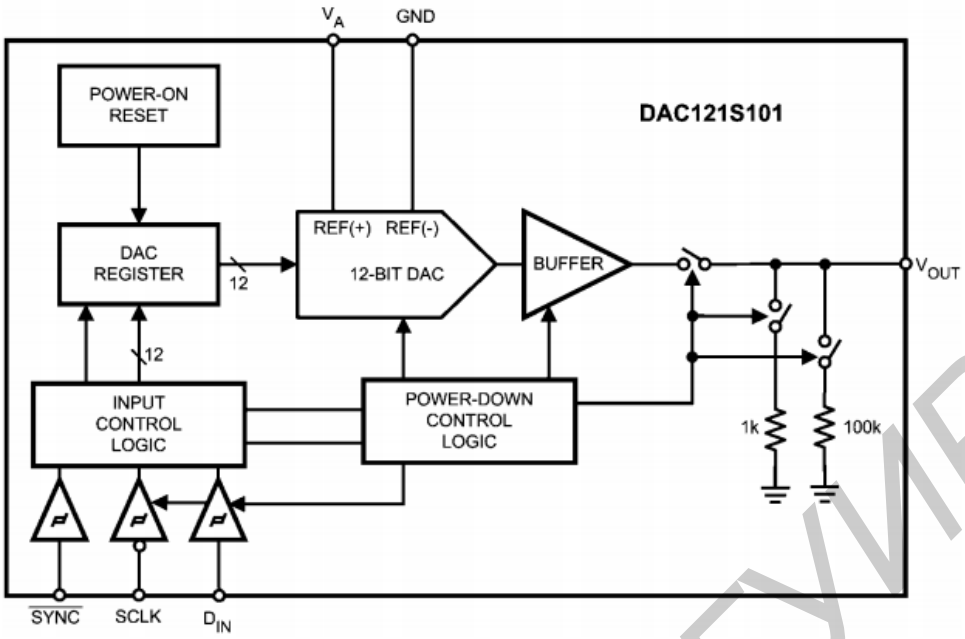


Рис. 1.39. Структура микросхемы ЦАП DAC121S101

Блок цифроаналогового преобразователя состоит из последовательно соединенных 4096 резисторов с коммутируемым средним выводом. Напряжение с коммутируемого делителя подается на выходной усилитель и далее на выход микросхемы. Схема управляемого делителя показана на рис. 1.40.

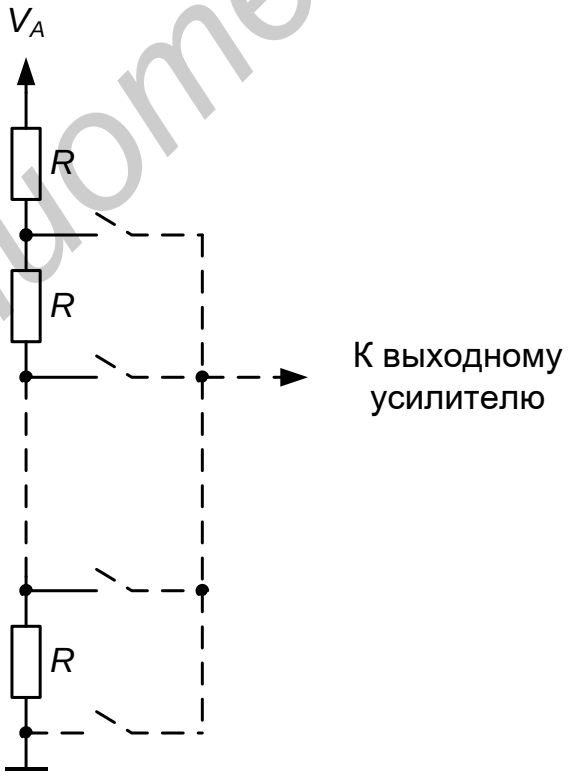


Рис. 1.40. Принцип преобразования цифрового кода в аналоговое напряжение

Напряжение на выходе ЦАП формируется за счет коммутации определенного отвода делителя напряжения к выходному усилителю, т. е. входной цифровой 12-разрядный код является номером соответствующего ключа, осуществляющего коммутацию.

Выходное напряжение ЦАП может быть определено по следующей формуле:

$$V_{OUT} = V_A \cdot \frac{D}{4096},$$

где V_A – напряжение питания микросхемы (в диапазоне от 2,7 до 5,5 В);

D – десятичный эквивалент двоичного числа, загружаемого во входной регистр ЦАП.

Напряжение питания модуля ***PmodDA2*** V_A составляет 3,3 В.

Загрузка информации во входной регистр микросхемы осуществляется по синхронному последовательному интерфейсу SPI. Для передачи данных по такому интерфейсу задействуются сигнальные линии $\overline{SYNC}$, *SCLK* и D_{IN} микросхемы. Вывод $\overline{SYNC}$ предназначен для кадровой синхронизации передаваемых данных. Активным уровнем для записи во входной регистр микросхемы является уровень логического «0». Вывод *SCLK* является входом тактирования при передаче информации. Запись в регистр осуществляется по заднему фронту *SCLK*. Вывод D_{IN} является последовательным входом данных для ЦАП. Для управления данным ЦАП необходимо передать во входной регистр 16 бит информации. На рис. 1.41 представлены временные диаграммы записи данных во входной регистр микросхемы преобразователя, а на рис. 1.42 – структура передаваемых данных.

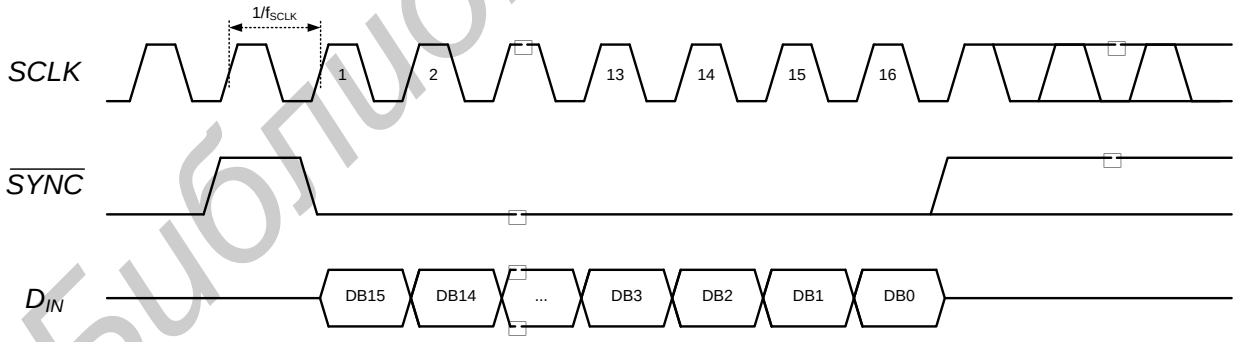


Рис. 1.41. Временные диаграммы записи данных в микросхему ЦАП

Данные по интерфейсу SPI передаются старшим битом (*DB*15) вперед. Максимальная частота тактирования (передачи данных) составляет 30 МГц.

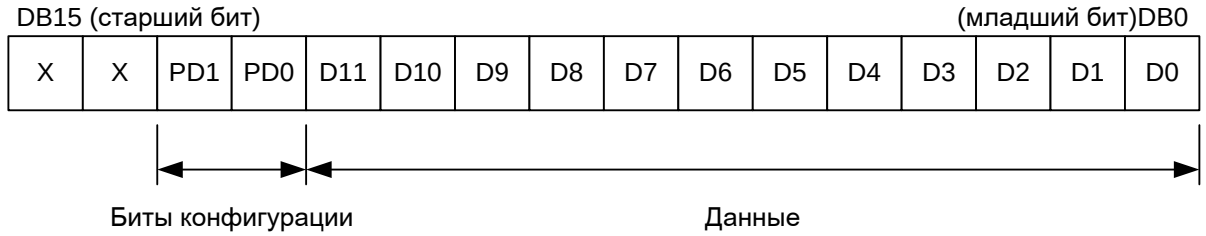


Рис. 1.42. Структура передаваемых данных

Блок передаваемых данных состоит из 16 бит. Из них 12 бит являются непосредственно данными об уровне сигнала на выходе ЦАП, 2 бита – биты конфигурации при отключении питания (Power-Down Modes) и 2 бита не используются. В зависимости от состояния битов конфигурации возможно четыре варианта работы микросхемы ЦАП (табл. 1.12).

Таблица 1.12

DB13	**DB12**	**Режим работы**
0	0	Нормальный режим работы
0	1	Выключение питания с подключением выхода аналогового сигнала на общий провод через резистор 1 кОм
1	0	Выключение питания с подключением выхода аналогового сигнала на общий провод через резистор 100 кОм
1	1	Выключение питания с переводом выхода аналогового сигнала в *Z*-состояние (с высоким выходным импедансом)

Для управления данным модулем можно использовать библиотечный элемент *DA2RefComp*, который имеет описание работы на языке VHDL и должен быть добавлен к проекту. Вид условного графического обозначения данного элемента показан на рис. 1.43.

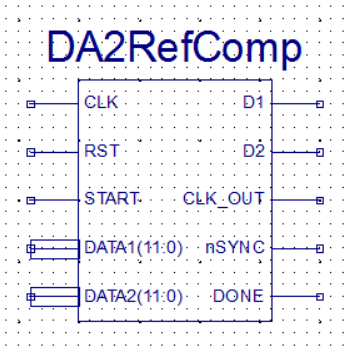


Рис. 1.43. Условное графическое обозначение компонента *DA2RefComp*

Данный элемент имеет вход синхроимпульсов *CLK*, на который подается тактирование от генератора с частотой 50 МГц, вход сброса *RST*, вход запуска передачи данных на микросхему ЦАП *START* и две 12-разрядные шины для передаваемых данных *DATA1* и *DATA2*. Информационные сигналы на входы *DATA1* и *DATA2* подаются в параллельном коде.

Из выходных сигналов компонент имеет следующие: *D1* и *D2* – выход последовательных данных на первый и второй канал ЦАП соответственно, *CLK_OUT* – выход тактирования, *nSYNC* – выход сигнала кадровой синхронизации, *DONE* – выход готовности компонента к загрузке новых данных. Для управления модулем ***PmodDA2*** необходимо вывести на соответствующие выводы разъемов расширения сигналы *D1, D2, CLK_OUT* и *nSYNC*.

Лабораторное задание

Реализовать генератор сигнала заданной формы на ПЛИС с заданной частотой, построенного по принципу генератора с прямым цифровым синтезом (*DDS*). За основу генератора использовать структурную схему, представленную на рис. 1.36, исключив возможность перестройки по частоте. Реализацию функциональных блоков генератора осуществить путем описания их на языке VHDL. Для связи с внешним модулем ЦАП использовать компонент ***DA2RefComp.vhd***. Связи между блоками выполнить в схемотехническом редакторе *Schematic*. Проверить работу устройства на отладочной плате.

Параметры формируемого сигнала, указываемые преподавателем:

1) форма сигнала;
2) период повторения (частота) формируемого сигнала;
3) амплитуда сигнала;
4) количество отсчетов сигнала на один период.

Порядок выполнения работы

1. Ознакомиться с принципами построения генераторов сигнала прямого синтеза и принципами работы цифроаналоговых преобразователей.
2. Получить у преподавателя форму сигнала (треугольник, трапеция, синусоидальный и т. д.), временные (период повторения, количество отсчетов на период) и амплитудные (амплитуда, количество уровней квантования сигнала по амплитуде) параметры сигнала.
3. На основе полученных данных произвести расчет сигнала заданной формы в среде *Matlab, Matcad, Maple* или *MS Exel* (получить отсчеты сигнала). Пример формирования сигнала функции sin*(x)* в среде *Matlab* с записью отсчетов в файл *Signal.txt* приведен ниже.

```
clc;                    %очистка командной строки
close all;              %закрываем все окна
clear all;              %очищаем переменные
format compact;         %вывод в сокращенной форме

N = 4096;       %количество точек на период сигнала
M = 4096;       %количество уровней квантования по амплитуде (разрядность ЦАП)
i = 0:N-1;      %номера отсчетов

X = 2*pi*i/N;                   %вычисляем аргумент функции sin(x)
Y = (M-1)/2*(sin(X)+1);         %вычисляем значения функции с масштабным
                                %коэффициентом
Z = round(Y');                  %округляем до целого числа

figure('color',[1 1 1]), plot(i,Z),grid;   %строим график функции Y = A*SIN(X) + B

axis tight;     %задаем диапазон изменения координат, строго соответствующий
                %диапазону изменения данных

%выводим в командную строку минимальное и максимальное значения
min(Y)          %проверяем на минимальное значение
max(Y)          %проверяем на максимальное значение

%запись файла сформированного сигнала
fid = fopen('Signal.txt','w');  %создаем файл, в который будем сохранять значения
                                %отсчетов
%записываем отсчеты
for k = 1:N-1
        fprintf(fid,'%.0f,',Z(k));
end
fprintf(fid,'%.0f',Z(length(Z)));          % пишем последний отсчет
status = fclose(fid);                       %закрываем файл
```

4. Реализовать блоки генератора на языке VHDL и произвести их соединение в *Schematic*.

5. Подготовить и загрузить в кристалл ПЛИС файл конфигурации.

6. Убедиться в работоспособности полученного устройства и произвести измерение временных и амплитудных параметров сигнала при помощи цифрового осциллографа.

7. Оформить отчет по лабораторной работе.

Содержание отчета

1. Титульный лист.
2. Цель работы.

3. Задание на лабораторную работу.

4. Расчет формы и необходимых параметров сигнала.

5. Схема реализованного устройства формирования сигнала в редакторе *Schematic*.

6. Тексты кодов программ функциональных блоков на языке VHDL.

7. Результаты работы устройства на отладочной плате (осциллограммы сигналов).

8. Схема электрическая принципиальная разработанного устройства (в необходимом объеме).

9. Выводы по работе.

Контрольные вопросы и задания

1. Назовите основные способы реализации генераторов сигналов заданной формы.

2. Какими способами производится преобразование сигнала, представленного в дискретной цифровой форме в непрерывный аналоговый сигнал?

3. Дайте определение генератора сигнала с прямым цифровым синтезом.

4. Приведите основные структурные схемы *DDS*-генераторов и укажите их достоинства и недостатки.

5. Перечислите способы реализации цифроаналоговых преобразователей сигналов.

6. Поясните принцип работы схемы *DDS*-генератора, разработанного в ходе выполнения лабораторной работы.

7. Поясните, как работает модуль ЦАП, использованный в работе.

8. Поясните, каким образом осуществлялось формирование отсчетов сигнала заданной формы.

1.6. *Лабораторная работа №4*. Преобразование аналоговых сигналов для обработки на ПЛИС

Цель работы

1. Изучение особенностей аппаратной реализации аналого-цифрового преобразования на ПЛИС.
2. Формирование практических навыков сопряжения модуля расширения АЦП с ПЛИС.
3. Приобретение практических навыков работы с реальными устройствами на базе ПЛИС и контрольно-измерительными приборами.

Краткие теоретические сведения

Множество цифровых устройств, разрабатываемых на ПЛИС, предназначено для задач цифровой обработки сигналов. Ввиду того, что ПЛИС – это цифровое устройство, то по определению оно работает только с цифровыми сигналами, имеющими два уровня квантования – уровень логического «0» и уровень логической «1». Для работы с аналоговыми сигналами необходимо иметь устройство, которое обеспечило бы преобразование входного аналогового сигнала в цифровую форму для дальнейшей обработки. Для этой цели служат аналого-цифровые преобразователи.

Аналого-цифровой преобразователь (АЦП, англ. – *Analog-to-digital converter, ADC*) – устройство, преобразующее входной аналоговый сигнал в дискретный код (цифровой сигнал). Как правило, АЦП – электронное устройство, преобразующее напряжение в двоичный цифровой код. Простейшим одноразрядным двоичным АЦП является компаратор.

Наиболее широкое распространение получили следующие типы АЦП:

1) прямого преобразования;
2) последовательного приближения;
3) дифференциального кодирования;
4) сравнения с пилообразным сигналом;
5) с уравновешиванием заряда;
6) с промежуточным преобразованием в частоту следования импульсов;
7) сигма-дельта-АЦП.

Основными характеристиками АЦП являются разрядность преобразования, от которой зависит разрешающая способность; точность, связанная с величиной ошибок квантования; нелинейность АЦП (из-за несовершенства физической структуры); апертурная погрешность (джиттер) – флуктуации из-за дрожания фронта синхросигнала; максимальная частота дискретизации входного сигнала – частота, с которой производится выдача цифровых значений сигнала.

Так как ПЛИС не содержит в своем составе аппаратных блоков АЦП, как, например, сигнальные процессоры и микроконтроллеры, то для обеспечения работы с аналоговыми сигналами необходимо использовать внешние АЦП, выполненные в виде отдельных микросхем. Каждая из таких микросхем АЦП имеет свой определенный интерфейс для управления и выдачи потока преобразованных данных. АЦП могут иметь интерфейсы с последовательной или параллельной выдачей цифрового кода. Задачей разработчика системы является написание программного кода или схемная реализация стыка между ПЛИС и микросхемой АЦП.

Для работы с аналоговыми сигналами, т. е. для преобразования их в цифровой код, в составе отладочной платы ***Nexys2*** используется дополнительный модуль расширения ***PmodAD1***. Этот модуль представляет собой двухканальный АЦП, выполненный на основе микросхем одноканального 12-разрядного АЦП ADCS7476MSPS. Структурная схема модуля расширения показана на рис. 1.44.

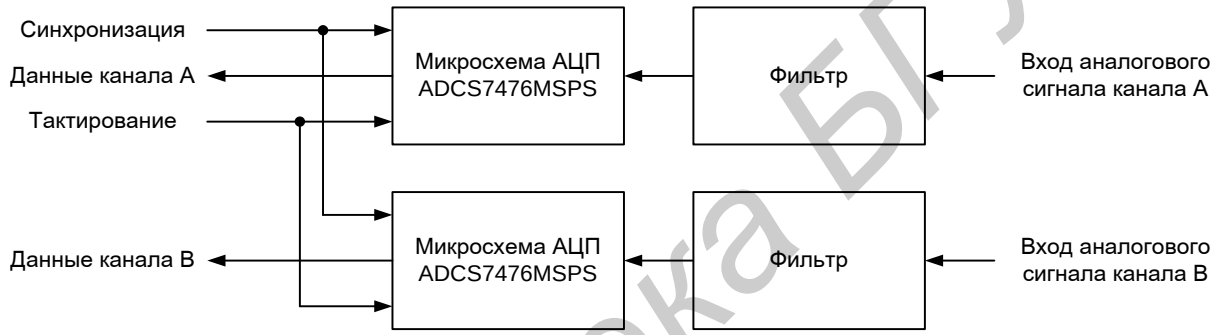


Рис. 1.44. Структура модуля расширения АЦП

Аналоговые сигналы подаются на входные антиалайзинговые фильтры (ФНЧ, уменьшающий вклад побочных частотных компонентов в выходном сигнале до пренебрежимо малых уровней) и далее на микросхему АЦП. Интерфейс взаимодействия с АЦП – SPI. Протокол обмена мало отличается от рассмотренного ранее протокола обмена данными с ЦАП. Максимальная частота тактирования при обмене данными составляет 20 МГц.

Микросхема АЦП ADCS7476MSPS представляет собой АЦП последовательного приближения с максимальной частотой 1 млн выборок в секунду. Процедура аналого-цифрового преобразования выполняется за 16 тактов синхросигнала на выводе синхронизации *SCLK*. Запуск АЦП производится переводом линии $\overline{CS}$ из уровня логической «1» в уровень логического «0». Сигнал $\overline{CS}$ должен удерживаться на линии на уровне логического «0» в течение всего времени преобразовании. При переводе $\overline{CS}$ в состояние логической «1» ранее 10-го такта на линии *SCLK* микросхема переводится в режим пониженного энергопотребления, а прочитанные данные в этом случае считаются недействительными.

Чтение данных на линии *SDATA* производится по заднему фронту тактового сигнала *SCLK*. Процедура чтения информации по шине SPI показана на рис. 1.45.

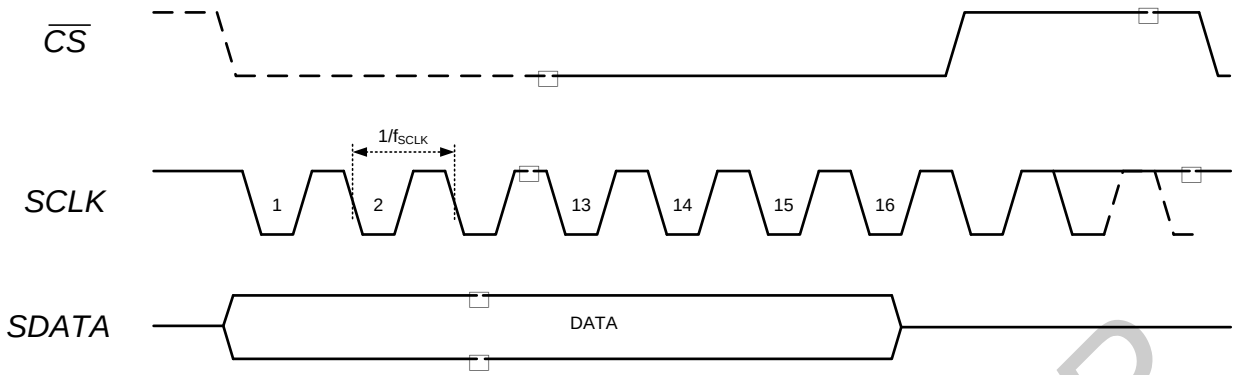


Рис. 1.45. Чтение данных из микросхемы АЦП

Из АЦП производится чтение 16 бит информации за 16 тактов. Структура данных представлена на рис. 1.46.

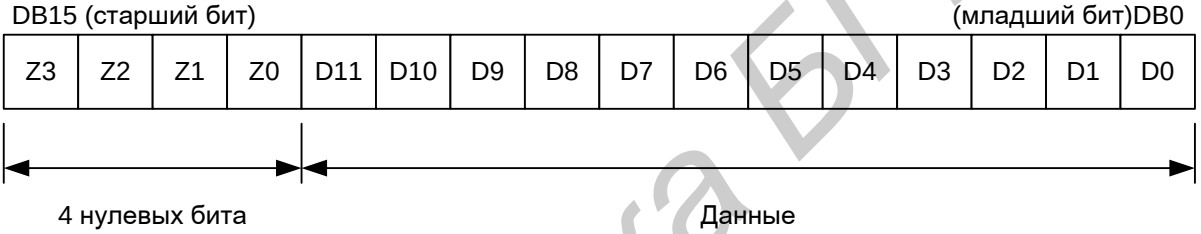


Рис. 1.46. Структура читаемых данных из АЦП

АЦП выдает цифровой код, пропорциональный поданному входному аналоговому сигналу. Идеальная передаточная характеристика АЦП показана на рис. 1.47.

Диапазон измеряемых значений входного сигнала лежит в пределах от 0 до V_{DD}, где V_{DD} – напряжение питания микросхемы АЦП, в данном случае V_{DD} равно 3,3 В.

Разрешающая способность по напряжению (цена деления 1 бита) для микросхемы ADCS7476MSPS составляет

$$LSB = \frac{V_{DD}}{4096}\ [\text{В}]\,.$$

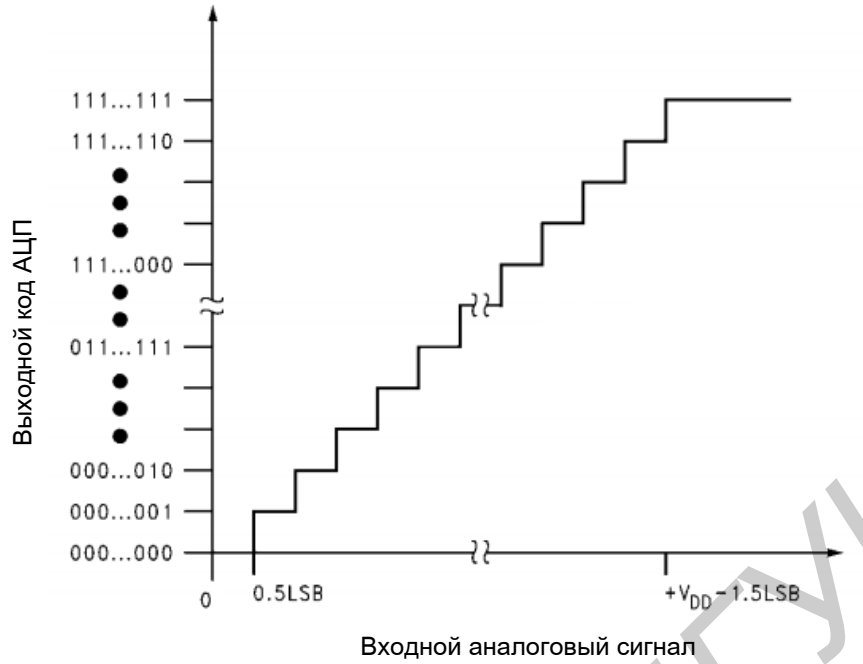


Рис. 1.47. Идеальная передаточная характеристика АЦП

Лабораторное задание

Реализовать протокол обмена данными по интерфейсу SPI между ПЛИС и модулем АЦП ***PmodAD1***. Описание его работы выполнить на языке VHDL. Организовать хранение получаемых данных в буфере либо произвести их обработку и выдачу на какое-либо устройство (например на индикатор). Проверить работу устройства на отладочной плате.

Порядок выполнения работы

1. Ознакомиться с теоретическими сведениями о принципах работы модуля аналого-цифрового преобразования.
2. Получить задание у преподавателя на лабораторную работу.
3. Реализовать протокол обмена между ПЛИС и АЦП на языке VHDL.
4. Подготовить и загрузить в кристалл ПЛИС файл конфигурации.
6. Проверить работоспособность полученного устройства.
7. Оформить отчет по лабораторной работе.

Содержание отчета

1. Титульный лист.
2. Цель работы.

3. Задание на лабораторную работу.

4. Коды текстов программ на языке VHDL.

5. Результаты работы устройства на отладочной плате.

6. Схема электрическая принципиальная разработанного устройства (в необходимом объеме).

7. Выводы по работе.

Контрольные вопросы и задания

1. Назовите основные типы аналого-цифровых преобразователей.

2. Поясните принцип работы АЦП ADCS7476MSPS.

3. Что такое антиалайзинговый фильтр?

4. Каким образом осуществляется связь модуля расширения АЦП ***PmodAD1*** с ПЛИС?

5. Поясните работу спроектированного модуля связи между ПЛИС и АЦП.

6. Какие параметры АЦП влияют на точность преобразования?

2. РЕАЛИЗАЦИЯ АЛГОРИТМОВ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ НА СИГНАЛЬНЫХ КОНТРОЛЛЕРАХ

2.1. Описание лабораторной установки

Лабораторная установка представляет собой аппаратно-программный комплекс, структура которого показана на рис. 2.1.

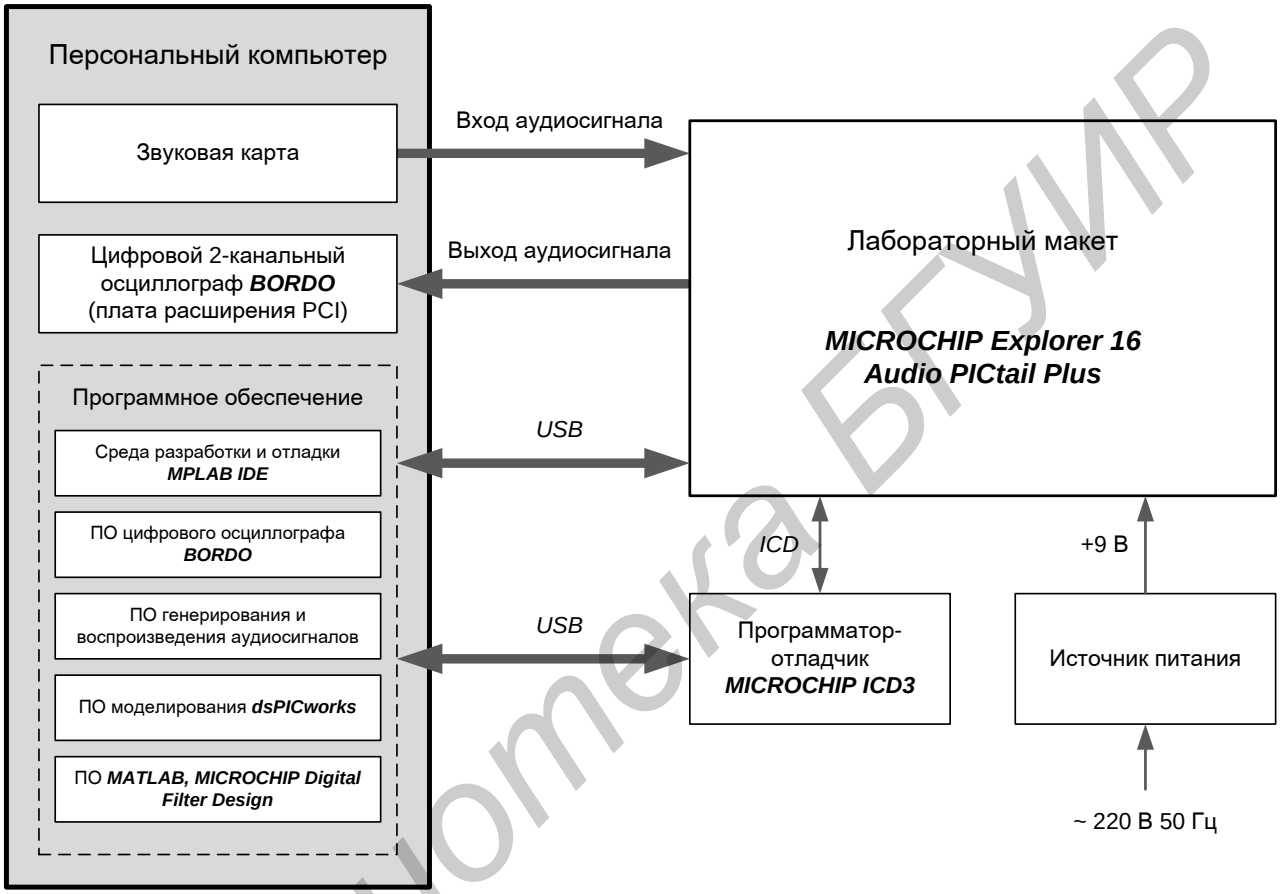


Рис. 2.1. Структурная схема лабораторной установки

Аппаратно-программный комплекс состоит из персонального компьютера (ПК) и лабораторного макета на базе отладочной платы ***Explorer 16***. Персональный компьютер включает аппаратную часть и специализированное программное обеспечение (ПО). В аппаратную часть ПК входят звуковая карта, которая предназначена для формирования акустических сигналов в диапазоне частот от 20 Гц до 20 кГц, и цифровой осциллограф ***Bordo*** для анализа сигналов, формируемых лабораторным макетом. Программное обеспечение включает среду разработки и отладки для написания программ **MPLAB IDE**, ПО для формирования аудиосигнала звуковой картой, ПО для управления осциллографом ***Bordo***, пакеты моделирования **dsPICworks** и **Digital Filter Desine**.

Лабораторный макет состоит из отладочной платы ***Explorer 16*** с платой расширения для обработки аудиосигналов ***Audio PICtail Plus***, устройства про-

граммирования и отладки ***Microchip ICD3*** и источника питания. Структурная схема отладочной платы ***Explorer 16*** показана на рис. 2.2.

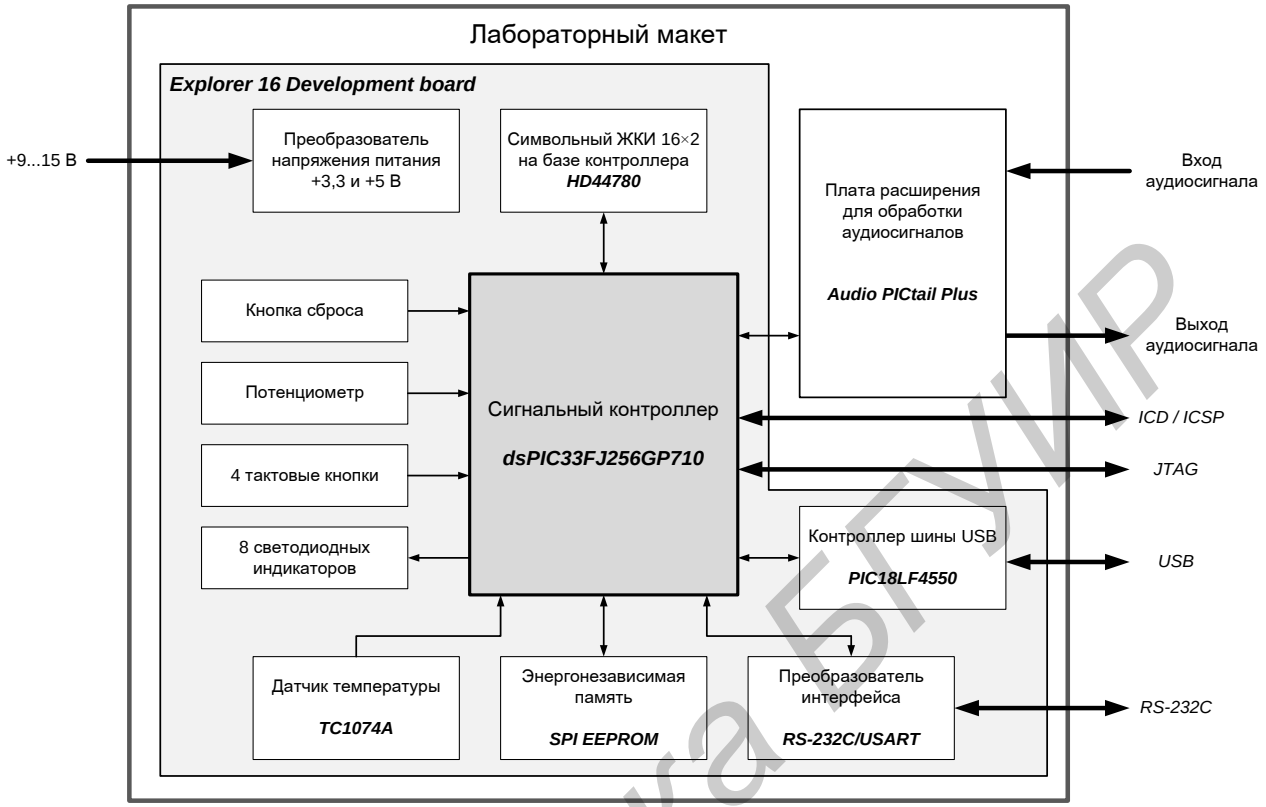


Рис. 2.2. Структурная схема отладочной платы ***Explorer 16***

Для решения задач цифровой обработки сигналов применяется плата расширения ***Audio PICtail Plus***, которая подключается через слот расширения к основной отладочной плате ***Explorer 16***, что позволяет сигнальному процессору dsPIC33F оцифровывать аудиосигнал с внешних источников, выполнять цифровую обработку и обратно преобразовывать цифровой сигнал в аналоговую форму. Ее структурная схема показана на рис. 2.3.

Более полное описание платы расширения ***Audio PICtail Plus*** приведено в прил. 1.

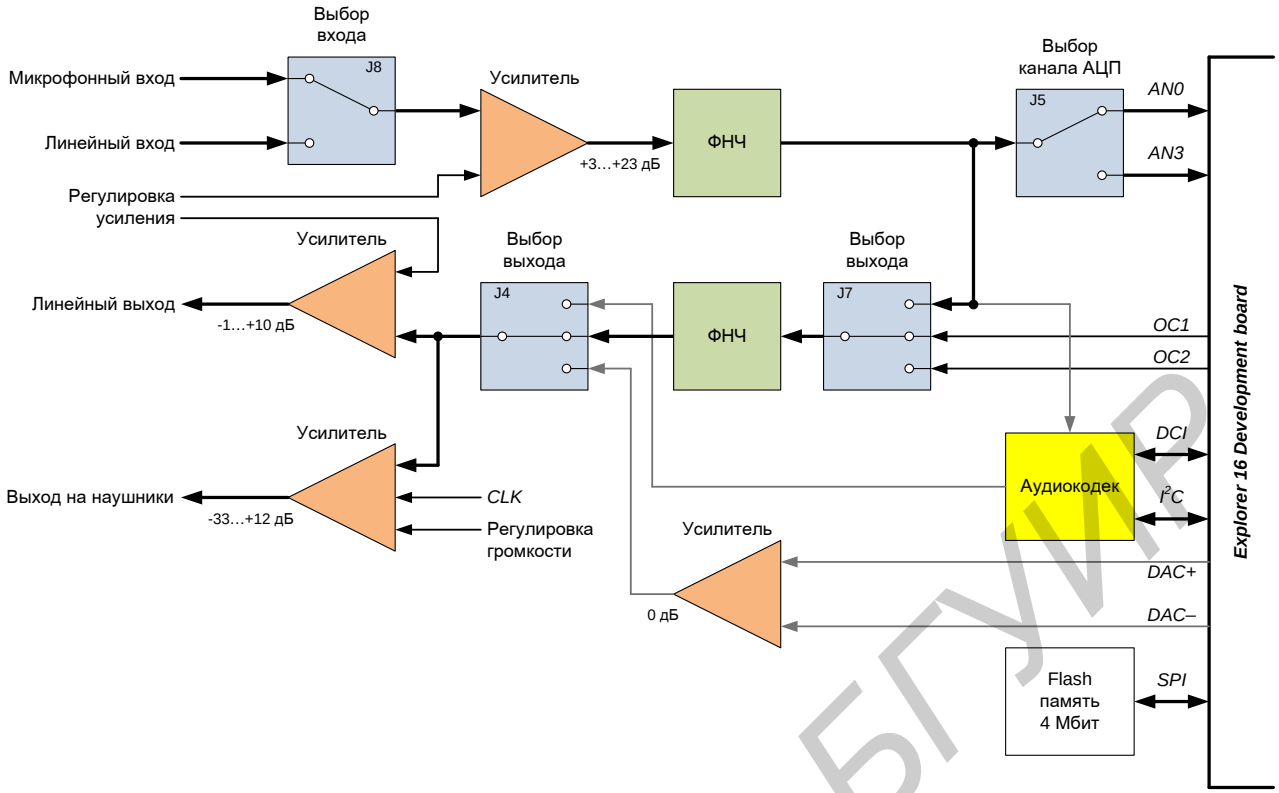


Рис. 2.3. Структурная схема платы расширения ***Audio PICtail Plus***

2.2. *Лабораторная работа №5*. Преобразование аналоговых сигналов для обработки на сигнальном контроллере

Цель работы

1. Изучение особенностей аппаратной реализации аналого-цифрового преобразования в системах цифровой радиосвязи.
2. Приобретение практических навыков программирования встроенных аппаратных средств сигнального контроллера, предназначенных для аналого-цифрового преобразования.

Краткие теоретические сведения

Модуль АЦП сигнального контроллера dsPIC33

Высокоскоростной АЦП основан на регистре последовательного приближения (SAR – *Successive Approximation Register*), что позволяет преобразовать аналоговый входной сигнал в 10-битный или 12-битный цифровой код с максимальной частотой 10,1 МГц.

Структурная схема модуля АЦП сигнального контроллера dsPIC33 показана на рис. 2.4.

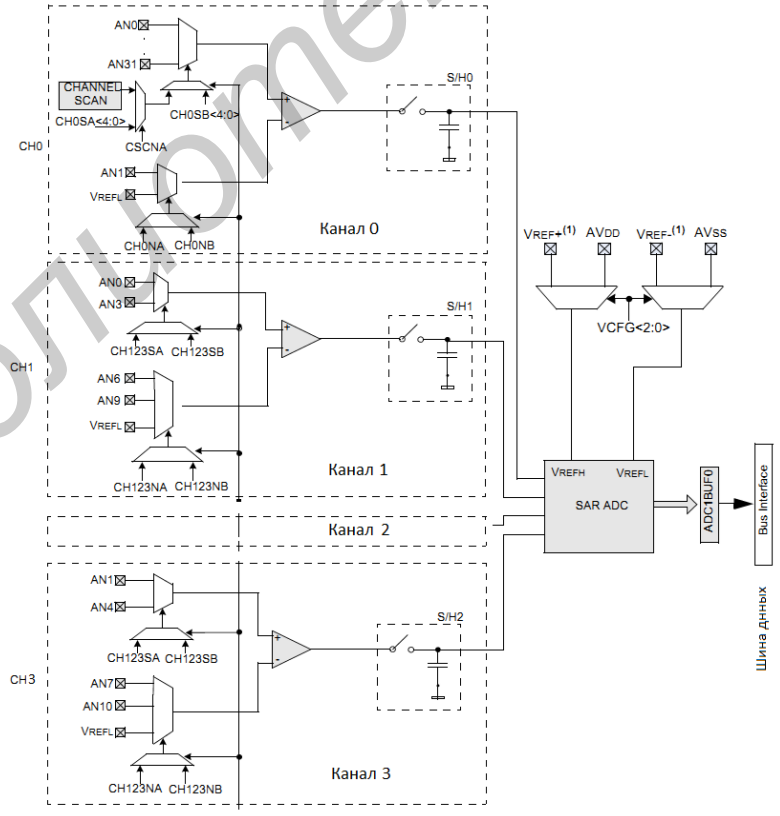


Рис. 2.4. Структурная схема модуля АЦП

Модуль АЦП имеет до 32 аналоговых входов AN0…AN31, которые через мультиплексоры подключаются к четырем усилителям накопителям S/H каналов CH0…CH3.

Модуль АЦП имеет десять регистров управления и состояния:

- **AD1CON1** – регистр управления 1 АЦП;
- **AD1CON2** – регистр управления 2 АЦП;
- **AD1CON3** – регистр управления 3 АЦП;
- **AD1CON4** – регистр управления 4 АЦП;
- **AD1CHS123** – регистр выбора канала CH0-CH3;
- **AD1CHS0** – регистр выбора канала 0;
- **AD1PCFGH** – старший регистр конфигурации портов АЦП;
- **AD1PCFGL** – младший регистр конфигурации входов АЦП;
- **AD1CSSH** – старший регистр настройки сканирования входных каналов;
- **AD1CSSL** – младший регистр настройки сканирования входных каналов.

Регистры **AD1CON1…AD1CON4** управляют ходом выполнения преобразования АЦП. С их помощью устанавливается частота преобразования, выбирается конфигурация источников опорного напряжения, а также выполняется программное управление выборкой и преобразованием входного сигнала.

Регистр **AD1CHS** позволяет выбирать каналы, которые подключаются к устройству выборки-хранения сигнала и преобразователя SAR, а также к мультиплексору для работы с входными сигналами.

С помощью регистра **AD1PCFG** осуществляется настройка выводов микроконтроллера для работы как с аналоговыми, так и с цифровыми сигналами.

Регистр **AD1CSSL** позволяет выбрать каналы входных сигналов для режима сканирования. Основные регистры и биты управления показаны ниже.

Регистры и биты конфигурации модуля АЦП приведены в прил. 2.

Фазы преобразования АЦП

На рис. 2.5 показаны две независимо контролируемые фазы работы АЦП: выборка и преобразование.

Фаза выборки – это когда сигнал с аналогового входа подается на конденсатор устройства выборки-хранения. Фазу выборки можно настроить на автоматический запуск после преобразования или программный запуск (битом *SAMP* в регистре управления **ADxCON1** <1>). Выборка контролируется битом *Auto-Sample* (*ASAM*) в регистре управления **ADxCON1** <2> (табл. 2.1).

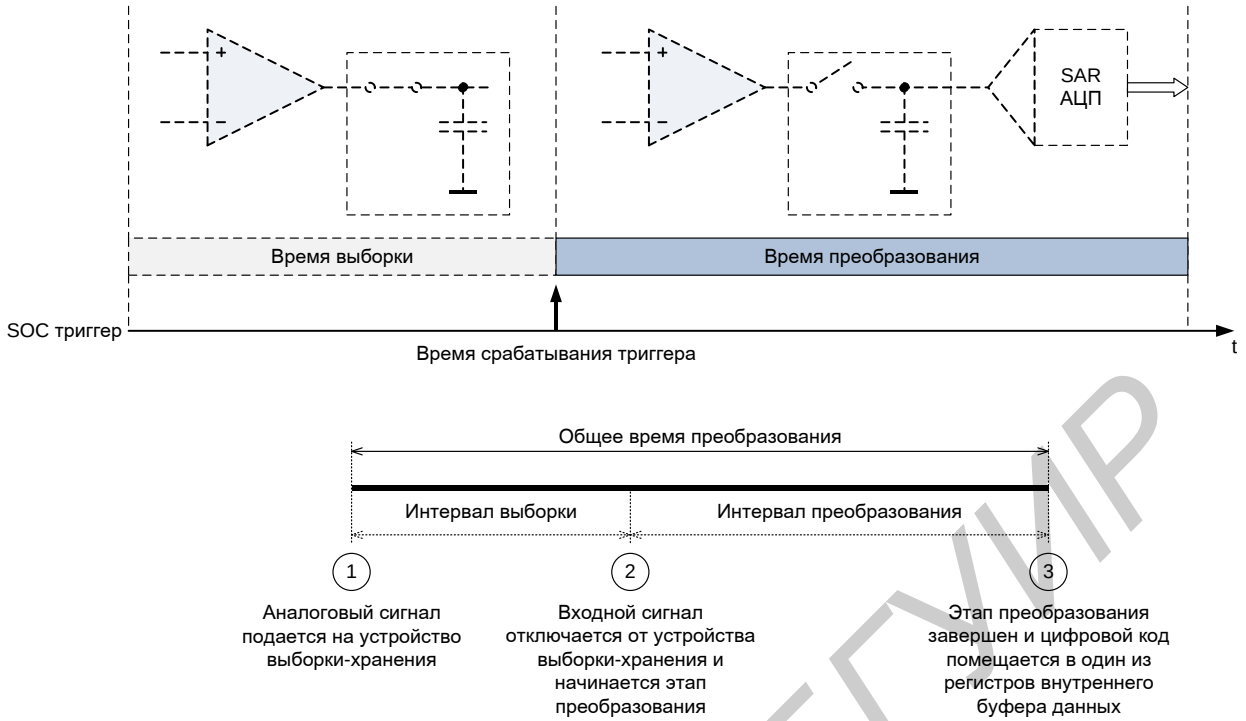


Рис. 2.5. Фазы работы модуля АЦП

Таблица 2.1

Бит ASAM	**Запуск выборки**
0	Программная выборка
1	Автоматическая выборка

Если включена автоматическая выборка, время выборки (T_{SMP}) АЦП равно числу тактов АЦП T_{AD}, умноженное на значение бит *SAMC* <4:00> (**ADxCON3** <12:08>):

$$T_{SMP} = \text{SAMC<4:0>} \cdot T_{AD}.$$

Если используется режим программной выборки, то необходимо программно обеспечить необходимую задержку для зарядки конденсатора в устройстве выборки-хранения.

Фаза преобразования – во время преобразования конденсатор отключается от мультиплексора и сохраняет накопленное напряжение, которое затем преобразовывается в цифровой код. Уравнение расчета времени преобразования для 10-битного и 12-битного режимов имеет следующий вид:

$$T_{CONV} = 12 \cdot T_{AD} \text{ (10 бит)};$$

$$T_{CONV} = 14 \cdot T_{AD} \text{ (12 бит)},$$

где T_{CONV} – время преобразования;

T_{AD} – период тактовых сигналов модуля АЦП.

Начало преобразования может быть инициализировано различными аппаратными источниками или же программно. Источник начала преобразования выбирается битами *SSRC* <2:00> в регистре управления АЦП (**ADxCON1** <7:05>).

Программное управление выборкой и преобразованием

Программное управление **выборкой и преобразованием** осуществляется установкой и сбросом бита *SAMP* в регистре **ADxCON1** <1>. В этом случае необходимо программно обеспечить задержку для обеспечения работы фазы выборки, что также окажет влияние на интервал дискретизации входного сигнала. Фазы работы модуля АЦП в режиме программного управления показаны на рис. 2.6.

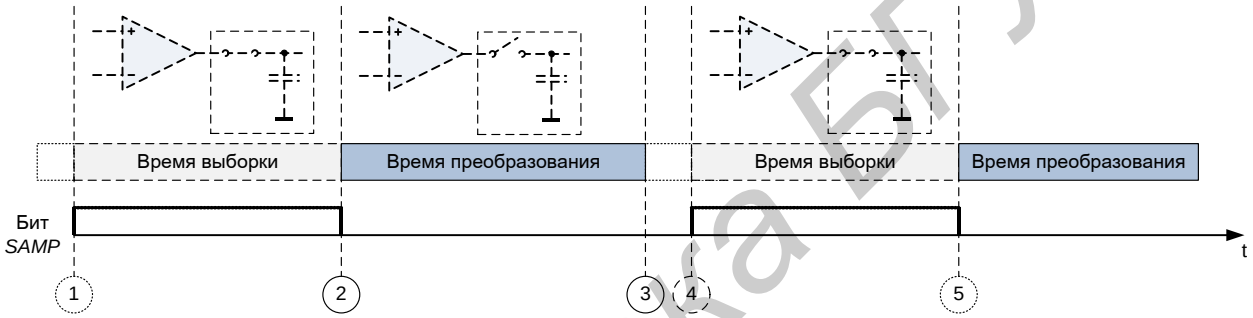


Рис. 2.6. Фазы преобразования АЦП:
1 – выборка запускается после программной установки бита *SAMP*;
2 – преобразование запускается после программной очистки бита *SAMP*;
3 – преобразование закончено;
4 – процесс повторяется согласно п. 1;
5 – процесс повторяется согласно п. 2

Пример программы:

```
AD1CON1bits.SAMP = 1;                  // запуск выборки
DelayUs(10);                           // задержка длительностью 10 мкс
AD1CON1bits.SAMP = 0;                  // запуск преобразования
while (!AD1CON1bits.DONE);             // ожидание полного преобразования
ADCValue = ADC1BUF0;                   // чтение результата преобразования
```

Автоматическая выборка и программное управление преобразованием

В данном режиме работы выборка начинается автоматически после завершения преобразования предыдущей выборки. Программно необходимо

обеспечить задержку для осуществления выборки до очистки бита *SAMP*, которое инициализирует преобразование (рис. 2.7).

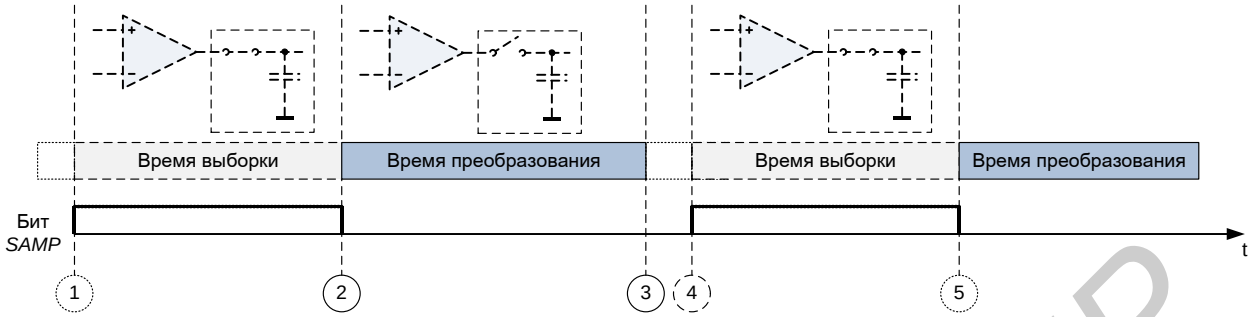


Рис. 2.7. Фазы преобразования АЦП:
1 – выборка запускается автоматически после завершения преобразования предыдущей выборки;
2 – преобразование запускается после программной очистки бита SAMP;
3 – преобразование закончено;
4 – процесс повторяется согласно п. 1;
5 – процесс повторяется согласно п. 2

Пример программы:

```
While (1)                                   // непрерывно повторяющийся цикл
  {
    DelayNmSec(100);                        // задержка 100 мс
      AD1CON1bits.SAMP = 0;                 // начало преобразования
      while (!AD1CON1bits.DONE);            // преобразование выполнено?
    AD1CON1bits.DONE = 0;                   // очистка бита состояния преобразования
                                            // (DONE)
    ADCValue = ADC1BUF0;                    // сохранение результата
  }
```

Автоматическая выборка и автоматическое преобразование

Данный режим обеспечивает более полный автоматизированный процесс выборки и преобразования аналоговых сигналов. Период выборки самосинхронизирован, и преобразование запускается автоматически после завершения периода выборки (рис. 2.8). Биты *SAMC* <4:0> в регистре **ADxCON3** <12:8> задают период выборки равный от 0 до 31 тактов АЦП (T_{AD}), при этом биты *SSRC* <2:0> должны быть установлены в состояние «111».

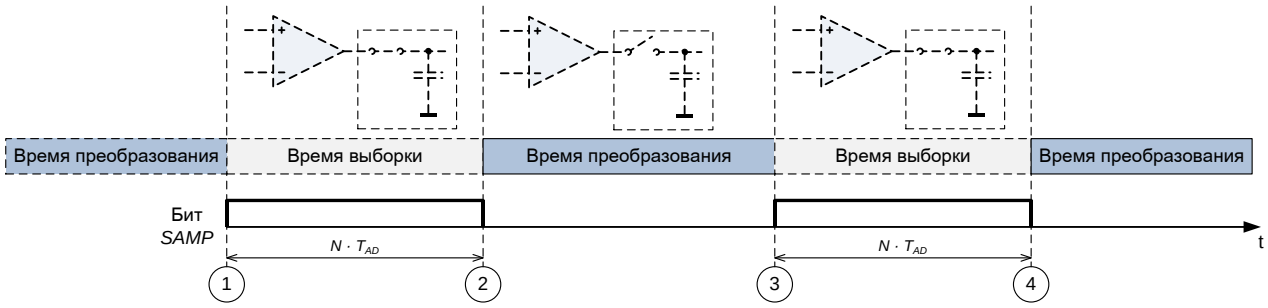


Рис. 2.8. Фазы преобразования АЦП:
1 – выборка запускается автоматически после преобразования;
2 – преобразование запускается автоматически после завершения периода выборки;
3 – процесс повторяется согласно п. 1;
4 – процесс повторяется согласно п. 2

Преобразование по событию от периферии

В режиме работы преобразования по событию от периферии модуля АЦП (рис. 2.9) фаза выборки начинается автоматически после предыдущего преобразования. Следующее преобразование начинается по событию от периферии (триггера). Триггер преобразования выбирается конфигурацией бит *SSRC* <2:0>.

Бит *ASAM* не должен изменяться во время преобразования модуля АЦП и должен быть установлен перед его включением. Также необходимо выполнить программную задержку для стабилизации процессов в модуле АЦП, поэтому если автоматическая выборка включена, то первый результат АЦП может быть некорректный. В таком случае необходимо отбросить первый результат в зависимости от тактовой частоты АЦП.

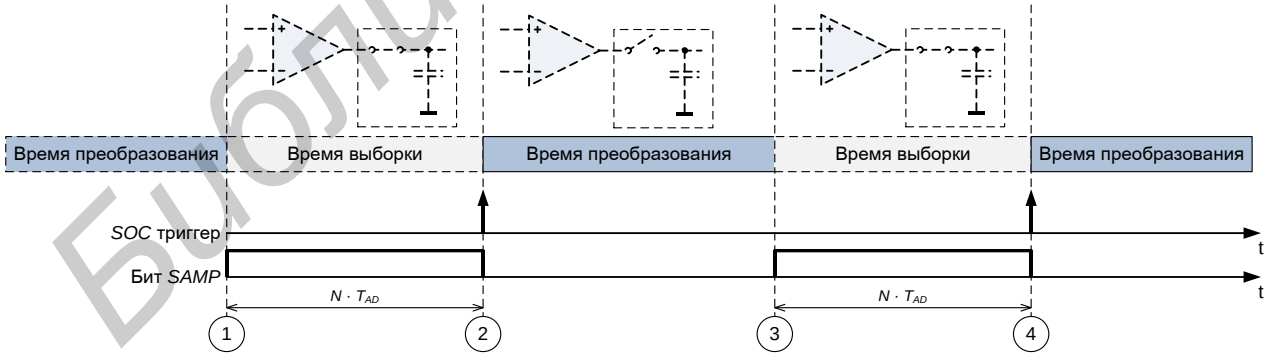


Рис. 2.9. Фазы преобразования АЦП:
1 – выборка запускается автоматически после преобразования;
2 – преобразование запускается на событие триггера;
3 – выборка запускается автоматически после преобразования;
4 – преобразование запускается на событие триггера

Выбор режима работы АЦП

Бит *AD*12*B* в регистре команд **ADC1** (**ADxCON1** <10>) переключает модуль АЦП между 10-разрядным и 12-разрядным режимом работы, что позволяют ему работать с 4 каналами (конфигурация по умолчанию) или с одним каналом соответственно (табл. 2.2).

Таблица 2.2

Бит AD12B	Режим работы АЦП
0	10-разрядный режим, поддерживается до 4 каналов (CH0…CH3)
1	12-разрядный режим, поддерживается только один канал CH0

В 10-разрядном режиме (*AD*12*B* = 0), используя биты *CHPS* <1:0> регистра **ADxCON2** <9:8>, можно выполнять преобразования с одним (CH0), двумя (CH0, CH1) или четырьмя каналами (CH0…CH3) (табл. 2.3).

Таблица 2.3

Биты CHPS <1:0>	Выбор каналов
00	Один канал CH0
01	Два канала CH0, CH1
1x	Четыре канала CH0…CH3 (многоканальное преобразование)

Многоканальное преобразование

Модуль АЦП в сигнальном контроллере dsPIC33 позволяет производить аналого-цифровое преобразование в многоканальном режиме. При этом фаза выборки может быть осуществлена одновременно для всех каналов либо поочередно для каждого из каналов, участвующих в преобразовании.

При одновременной выборке производится «снимок» всех аналоговых каналов, а затем осуществляется само преобразование. Последовательная выборка делает «снимок» каждого аналогового входа непосредственно перед преобразованием (рис. 2.10).

Модуль АЦП выполняет одновременную выборку, используя два или четыре канала, и затем осуществляет преобразование для каждого канала последовательно.

Одновременная выборка может использоваться для получения информации о фазе сигнала между различными каналами.

Одновременная выборка выбирается установкой бита *SIMSAM* в регистре **ADxCON1** <3>. По умолчанию выборка и преобразование выполняются последовательно. В табл. 2.4 приведены настройки по конфигурации бита *SIMSAM*.

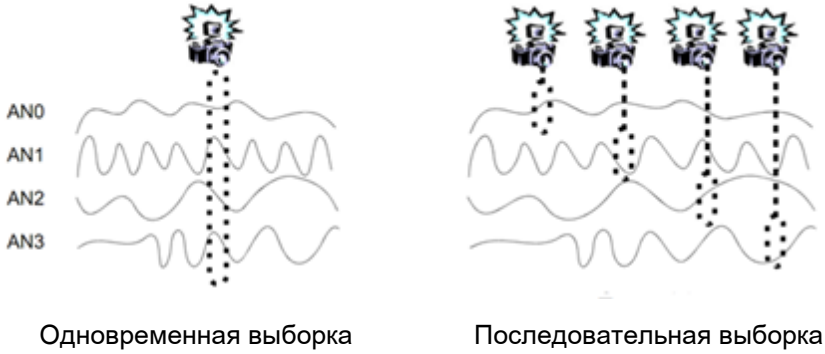


Рис. 2.10. Этапы преобразования АЦП

Таблица 2.4

SIMSAM	Режим работы
1	Одновременная выборка
0	Последовательная выборка

Процесс одновременной выборки для аналого-цифрового преобразования с двумя каналами показан на рис. 2.11 и состоит из следующих этапов:

1. Входной мультиплексор каналов CH0…CH1 подключает аналоговый вход для выполнения выборки сигналов.

2. По событию триггера начала преобразования конденсаторы каналов CH0…CH1 отсоединяются одновременно от аналоговых входов. Значение напряжения, полученное в канале CH0, преобразовывается в эквивалентный цифровой код.

3. Значение напряжения, полученное в канале CH1, преобразовывается в эквивалентный цифровой код.

4. Процесс повторяется согласно п. 1.

5. Процесс повторяется согласно п. 2.

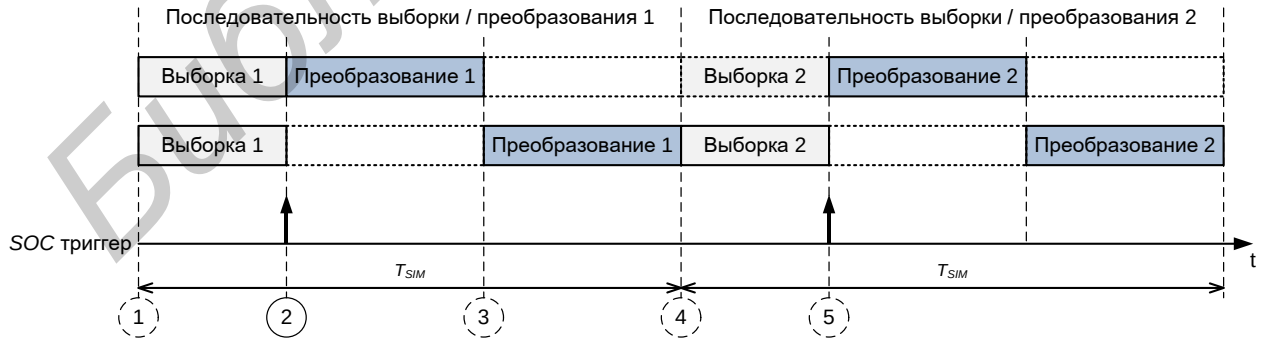


Рис. 2.11. Фазы преобразования АЦП

Время, затрачиваемое на полное преобразование, вычисляется следующим образом:

$$T_{SIM} = T_{SIM} + (M \cdot T_{CONV}),$$

где T_{SIM} – полное время, затраченное на выборку и преобразование нескольких каналов (каналы с одновременной выборкой);

T_{SMP} – время, затраченное на выборку;

M – количество каналов;

T_{CONV} – время, затраченное на преобразования.

Выбор источника тактовых сигналов

Модуль АЦП может быть синхронизирован от системных тактовых сигналов T_{CY} (командного цикла) или от внутреннего тактового *RC*-генератора (рис. 2.12). При использовании тактовых сигналов T_{CY} применяется делитель, что позволяет выбрать более низкую частоту для АЦП. Делитель тактовых сигналов T_{CY} управляется битами *ADCS* <7:0> в регистре **ADxCON3** <7:0>, и задает коэффициент деления от 1 до 64.

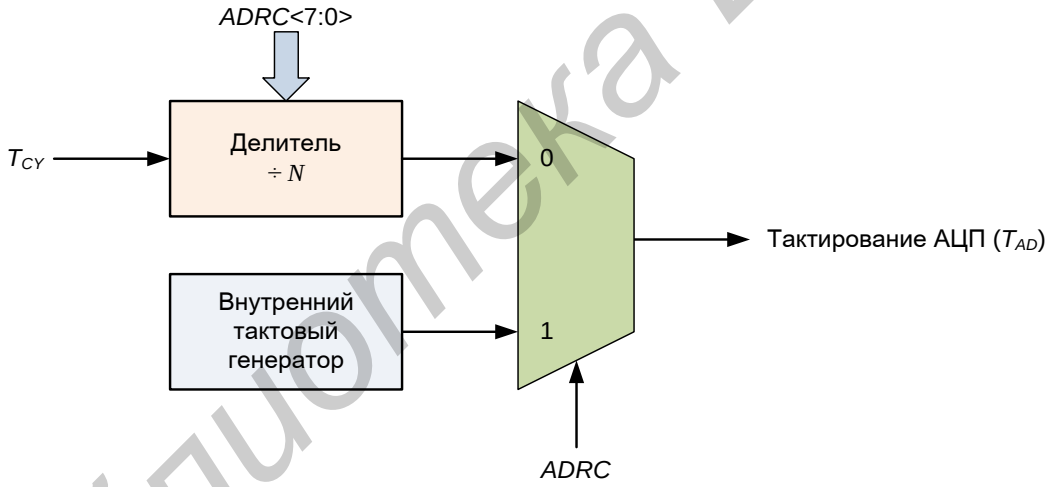


Рис. 2.12. Схема тактирования АЦП

Для правильного преобразования АЦП период тактовых сигналов не должен быть менее 75 нс.

Формула для расчета:

если бит *ADRC* = 0, то тактовый период АЦП

$$T_{AD} = T_{CY} \cdot (ADCS + 1);$$

если бит *ADRC* = 1:

$$T_{AD} = T_{AD} \cdot RC.$$

Внутренний источник тактовых сигналов используется, когда необходимо выполнять преобразования во время спящего режима (Sleep). Внутренний *RC*-генератор выбирается установкой бита *ADRC* в регистре **ADxCON3** <15>. Если бит *ADRC* = 1, то биты *ADCS* <7:0> не влияют на работу АЦП.

Формат выходных данных

Результат АЦП доступен в четырех различных числовых форматах, которые выбираются битами *FORM* <1:0> в регистре **ADxCON1** <9:8>. Примеры форматов данных 10-битного АЦП показаны на рис. 2.13.

Модуль АЦП содержит однословный регистр чтения результата преобразования **ADCxBUF0**. Если надо сохранить более одного результата преобразования прежде, чем инициализируется прерывание, то необходимо применять модуль передачи данных DMA. Каналы АЦП могут инициировать передачу данных в память DMA. Таким образом, программно можно обработать несколько результатов преобразования с минимальными задержками, тем самым экономя ресурсы сигнального контроллера.

Лабораторное задание

Требуется выполнить аналого-цифровое преобразование сигнала, подаваемого на вход AN0/RB0 сигнального контроллера с помощью встроенного АЦП, используя различные режимы работы. Результат записать в буфер RAM сигнального контроллера. Размер буфера имеет 256 значений. После записи последнего 256 результата буфер перезаписывается заново. Частота дискретизации АЦП задается в пределах 6…44 кГц.

Необходимо отредактировать исходный код программы для управления АЦП, приведенный ниже, согласно индивидуальному заданию.

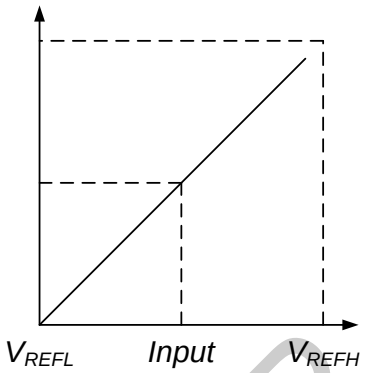
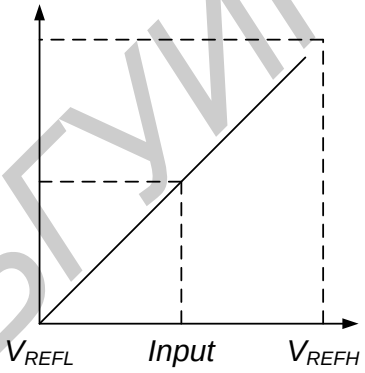
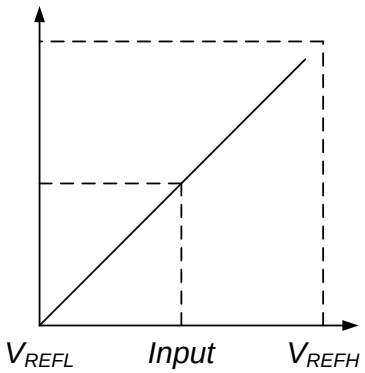
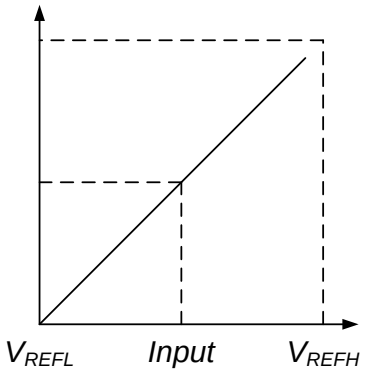
***FORM* = 0b11** Signed Fraction (Q15) Знаковый дробный	0111 1111 1100 0000 **(+0,999)** . . . 0000 0000 0000 0000 **(0)** . . . 1000 0000 0000 0000 **(-1)** 
***FORM* = 0b10** Unigned Fraction (Q16) Беззнаковый дробный	1111 1111 1100 0000 **(+0,999)** . . . 1000 0000 0000 0000 **(+0,5)** . . . 0000 0000 0000 0000 **(0)** 
***FORM* = 0b01** Signed Integer Знаковый целый	0000 0001 1111 1111 **(+511)** . . . 0000 0000 0000 0000 **(0)** . . . 1111 1110 0000 0000 **(-512)** 
***FORM* = 0b00** Unigned Integer Беззнаковый целый	0000 0011 1111 1111 **(+1023)** . . . 0000 0010 0000 0000 **(+512)** . . . 0000 0000 0000 0000 **(0)** 

Рис. 2.13. Схематичное отображение формата чисел 10-битного АЦП

Файл **main.c**

```
#include “p33Fxxxx.h”
#include <stdint.h>
void __delay32(unsigned long cycles);
#define SYS_FREQ 80000000UL
#define FCY SYS_FREQ/2
#define delay_us(x) __delay32(((x*FCY)/1000000L))
_FOSCSEL(0x02);
_FOSC(0xE2);
unsigned int Buffer[256], i = 0;

int main(void)
{

    AD1PCFGL = 0b011111111;          // настраиваем AN8 как аналоговый вход
    AD1CON2bits.VCFG = 0b000;        // внутреннее опорное напряжение
    AD1CON2bits.CHPS = 0b00;         // преобразование через канал CH0
    AD1CON1bits.SSRC = 0b000;        // ручной запуск преобразования
    AD1CON1bits.ASAM = 0;            // ручная выборка
    AD1CON3bits.ADRC = 1;            // для тактирования АЦП используется системная
                                     // тактовая частота
    AD1CON2bits.SMPI = 0b0000;       // прерывания после каждой выборки
    AD1CON3bits.SAMC = 0b11111;      // выборка через 31Tad
    AD1CON3bits.ADCS = 0b11111111;   // Tad = 256*Tcy
    AD1CHS0bits.CH0SA = 8;           // положительный сигнал берется с канала AN8
    AD1CHS0bits.CH0NA = 0;           // отрицательный сигнал берется с Vref-
    AD1CON1bits.FORM = 0b00;         // результат беззнаковый сдвинутый вправо
    _AD1IF = 0;                      // сбрасываем флаг прерывания АЦП
    _AD1IP = 1;                      // устанавливаем приоритет прерывания
    _AD1IE = 1;                      // разрешаем прерывания по АЦП
    AD1CON1bits.ADON = 1;            // включаем модуль АЦП
    // AD1CON1bits.SAMP = 1;         // запускаем преобразование

    while(1)
    {
        AD1CON1bits.SAMP = 1;        // начало выборки
        delay_us(10);                // ожидание зарядки конденсатора
        AD1CON1bits.SAMP = 0;        // начало преобразования
        while (!AD1CON1bits.DONE);   // чтение результата преобразования
    }
}

    // обработчик прерывания АЦП
    void __attribute__((__interrupt__,auto_psv)) _ADC1Interrupt(void)
```

```
{
    Buffer[i] = ADC1BUF0;          // записываем данные в массив
    i++;                           // переходим к следующему элементу
    i %= 256;                      // циклическая запись
    _AD1IF = 0;                    // сбрасываем флаг прерывания АЦП
}
```

Требования к отчету

1. Формулировка индивидуального задания, полученного у преподавателя.
2. Описание решения задачи с привлечением аналитических формул, временных диаграмм, функциональной и принципиальной схем, а также блок-схем алгоритмов.
3. Текст отлаженной программы с комментариями.
4. Выводы по работе.

Контрольные вопросы и задания

1. Поясните особенности реализации аналого-цифрового преобразования в системах цифровой радиосвязи.
2. Прокомментируйте режимы, фазы работы модуля АЦП, устанавливаемые с помощью управляющих регистров.
3. Приведите последовательность инициализации модуля АЦП.
4. Назовите и поясните факторы, влияющие на точность и скорость аналого-цифрового преобразования.

2.3. *Лабораторная работа №6*. Формирование аналоговых сигналов на сигнальном контроллере

Цель работы

1. Изучение принципов организации цифроаналогового преобразования на основе ШИМ при использовании сигнального контроллера.

2. Приобретение практических навыков настройки и программирования встроенных аппаратных средств сигнального контроллера для формирования аналоговых сигналов.

Краткие теоретические сведения

Для цифроаналогового преобразования можно применять цифровой сигнал с широтно-импульсной модуляцией (ШИМ) совместно с ФНЧ.

Импульсы формируются с одинаковым периодом T, который обеспечивается таймером и регистром периода. Длительность импульса $T_и$ изменяется программно в соотношении от 0 до 100 % от периода таймера. Отношение между периодичностью сигнала T и длительностью импульса $T_и$ называют рабочим циклом (скважностью) ШИМ.

Разрешение ШИМ – это количество возможных значений $T_и$, выраженное как логарифм по основанию 2. Например, ШИМ с 8-разрядным разрешением будет иметь 256 значений $T_и$.

Спектр сигнала с ШИМ и заданным рабочим циклом содержит три основные составляющие (рис. 2.14):

– постоянная составляющая с амплитудой, прямо пропорциональной рабочему циклу;

– гармоника на основной частоте ($f = 1/T$);

– бесконечное число гармоник, частота которых является кратным числом основной частоте ($2f, 3f, 4f, 5f, 6f, \ldots$).

Поэтому, если подсоединить «идеальный» фильтр низких частот к выводу генератора сигнала ШИМ, чтобы подавить все остальные составляющие, то возможно получить чистый аналоговый сигнал, амплитуда которого будет прямо пропорциональна рабочему циклу.

Однако такого идеального фильтра не существует, но можно применить некоторое к нему приближение, чтобы подавить как можно больше нежелательных частотных компонентов. Этот фильтр может быть выполнен в виде фильтра низких частот первого порядка (R / C).

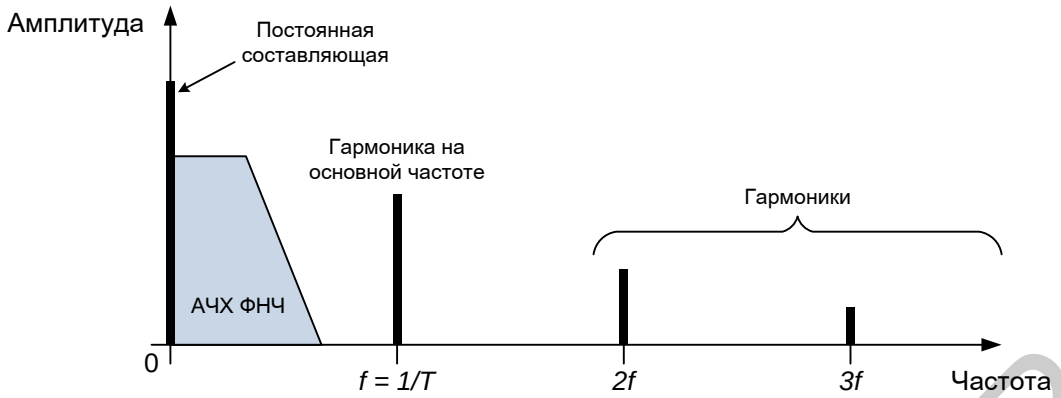


Рис. 2.14. Спектр ШИМ-сигнала

Для воспроизведения аудиосигнала выбирается частота ШИМ таким образом, чтобы она превышала диапазон слышимых частот 20 кГц, что позволит использовать более простые и недорогие фильтры.

Использование модуля сравнения dsPIC в режиме ШИМ

Структурная схема формирователя ШИМ сигнала на основе модуля сравнения (OCx) показана на рис. 2.15.

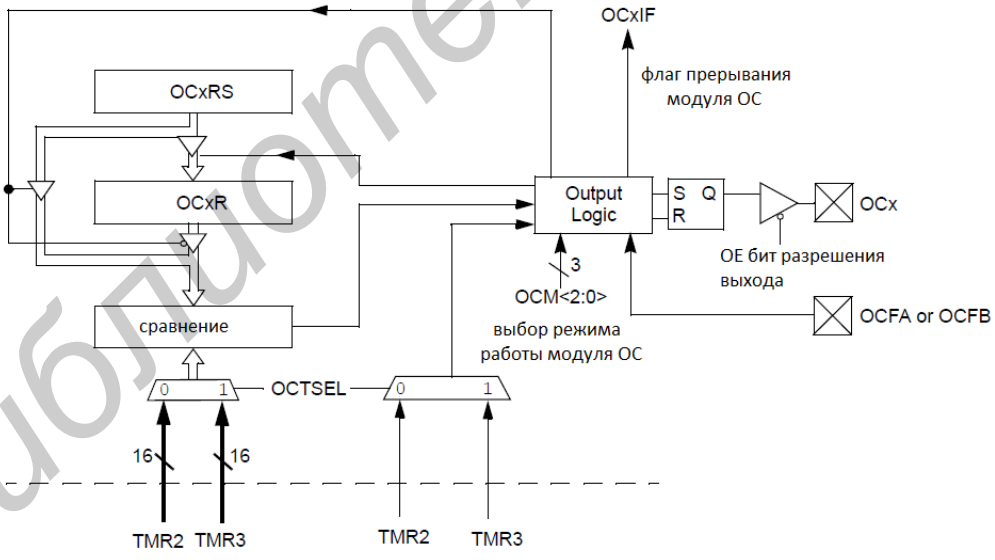


Рис. 2.15. Структурная схема модуля ОС для использования в режиме ШИМ

Этапы работы модуля сравнения в режиме ШИМ показаны на рис. 2.16.

1. Осуществляется запись скважности ШИМ в регистр **OCxR**.
2. Скважность второго цикла ШИМ записывается в регистр **OCxRS**.

3. Включается режим генерации ШИМ, вывод **OCx** устанавливается в высокий уровень.

4. Запускается Таймер (Timer) и начинает увеличиваться.

5. При совпадении Таймера со значением регистра **OCxR**, вывод **OCx** устанавливается в низкий уровень. В **OCxR** загружается новое значение из **OCxRS**.

6. При совпадении Таймера со значением регистра **PR** вывод **OCx** устанавливается в высокий уровень.

7. Загружается новое значение из **OCxRS** в **OCxR** и цикл повторяется.

Таким образом, длительность импульса ШИМ (скважности ШИМ) определяется битами в регистрах **OCxRS** и **OCxR**, а период ШИМ регистрами **PR2** или **PR3**.

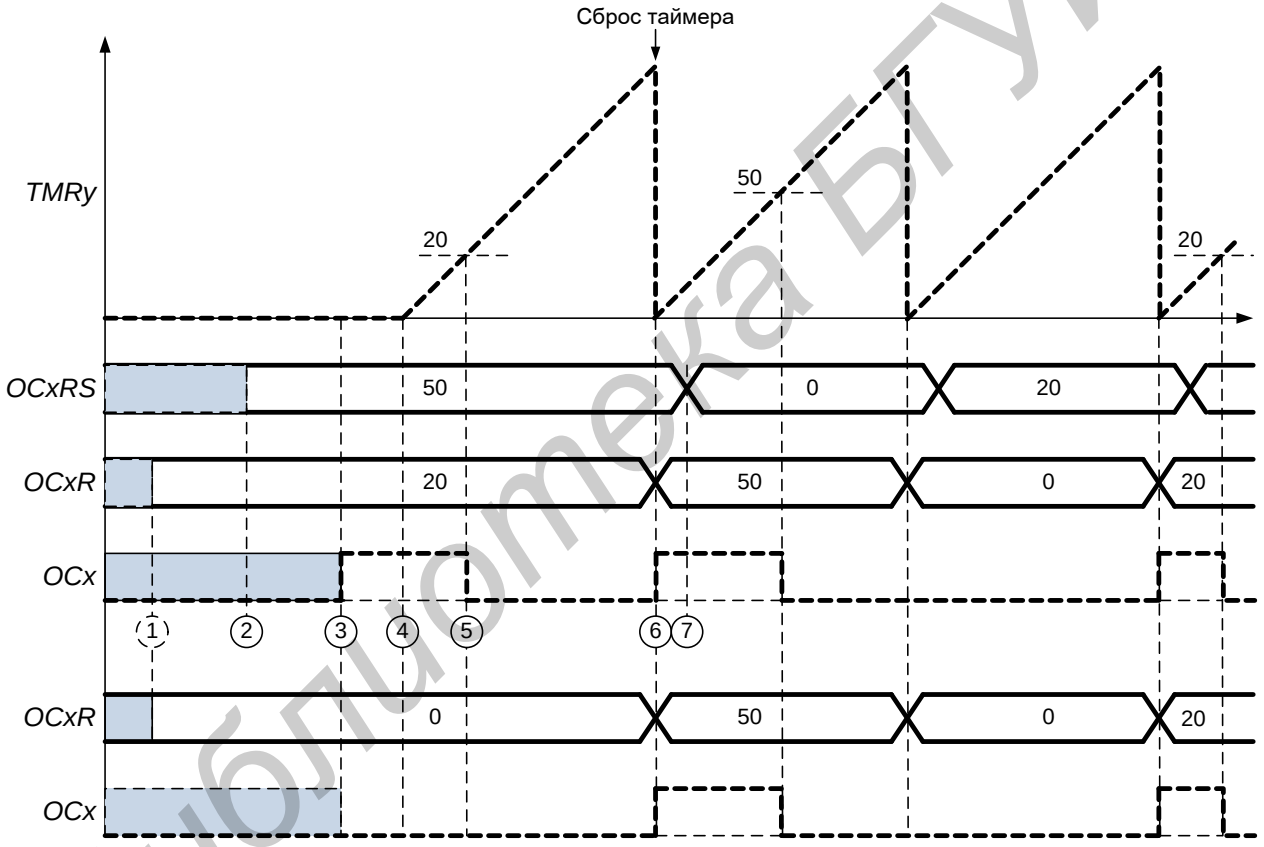


Рис. 2.16. Этапы работы модуля сравнения в режиме ШИМ

Для настройки модуля ШИМ используется регистр **OCxCON** (табл. 2.5).

Таблица 2.5

Бит 3 OCTSEL:	**Бит выбора таймера сравнения:** 1 = Timer3 источник тактовых сигналов; 0 = Timer2 источник тактовых сигналов
Биты 2-0 OCM<2:0>:	**Биты выбора режима модуля ОС в режиме ШИМ:** 111 = ШИМ-режим с защитой от ошибки; 110 = ШИМ-режим без защиты от ошибки; 000 = Модуль выключен. Генерация цифровых сигналов остановлена

На первом этапе установкой бита *OCTSEL* выбирается источник тактовых сигналов, который задает период и скважность ШИМ (Timer2 или Timer3).

Затем для инициализации ШИМ устанавливаются три бита OCM.

Период ШИМ вычисляется с учетом предделителя частоты и регистрами периода повторения таймера **PR2** или **PR3**.

Период ШИМ можно рассчитать по следующей формуле:

$$PWM_{Period} = [PRy + 1] \cdot T_{CY} \cdot TMRy_{Prescale\ Value};$$

$$PWM_{Frequency} = 1/PWM_{Period},$$

где PWM_{Period} – период ШИМ;
PRy – регистр периода таймера;
T_{CY} – время командного цикла сигнального контроллера;
$TMRy_{Prescale\ Value}$ – предделитель таймера;
$PWM_{Frequency}$ – частота ШИМ.

Например, при частоте тактирования сигнального контроллера 16 МГц, предделителя частоты 1:1 и периоде **PR** = 400 частота формирования ШИМ-сигнала будет составлять 40 кГц.

Если рассматривать возможность хранения речевых сообщений, зная, что спектр человеческой речи содержится в диапазоне частот от 400 Гц до 4 кГц, то можно ограничить воспроизведение ШИМ-сигнала 8 000 выборок в секунду. Обратите внимание на то, что для восстановления сигнала необходимо поддерживать более высокую частоту ШИМ вне диапазона звуковых частот и низких частот фильтра.

Таким образом, частота выборки зависит от рабочего цикла ШИМ и соответственно от частоты чтения новых данных из таблицы значений уровней квантования восстанавливаемого сигнала. Рассмотрим случай, когда новое значение из таблицы загружается с частотой 8 кГц при частоте ШИМ 40 кГц, т. е. каждые пять прерываний (40 000/8 000 = 5), тогда теоретически можно воспроизвести около 16 секунд речевого сообщения, записанного во Flash-память сиг-

нального контроллера. Для воспроизведения более длительных сообщений используются различные методы сжатия аудиоданных или внешние накопители.

Ниже приведен пример инициализации модуля ШИМ.

```
OC1CONbits.OCM = 0b000;          // выключение модуля Output Compare
OC1R = 100;                      // запись первого значения длительности
                                 // импульса ШИМ
OC1RS = 200;                     // запись второго значения длительности
                                 // импульса ШИМ
OC1CONbits.OCTSEL = 0;           // выбор Timer 2
OC1CONbits.OCM = 0b110;          // выбор модуля Output Compare
T2CONbits.TON = 0;               // отключение Timer2
T2CONbits.TCS = 0;               // выбор внутреннего источника тактовых сигналов
T2CONbits.TGATE = 0;             // запрет Gated Timer mode
T2CONbits.TCKPS = 0b00;          // выбор предделителя 1:1 таймера
TMR2 = 0x00;                     // очистка таймера
PR2 = 500;                       // запись периода ШИМ
IPC1bits.T2IP = 0x01;            // задание приоритета прерывания Timer2
IFS0bits.T2IF = 0;               // очистка флага прерывания Timer2
IEC0bits.T2IE = 1;               // разрешение таймера прерывания Timer2
T2CONbits.TON = 1;               // запуск Timer2

/* Пример кода обработчика прерывания для Timer2 */
void __attribute__((__interrupt__)) _T2Interrupt( void )
{
OC1RS = 300;  // запись следующего значения длительности импульса ШИМ
IFS0bits.T2IF = 0;       // очистка флага прерывания Timer2
}
```

Формирование сигнала с помощью пакета моделирования dsPICworks

Рассмотрим пример синтеза аналогового сигнала с последующим формированием отсчетов с помощью пакета моделирования **dsPICworks** для дальнейшего его воспроизведения на сигнальном контроллере.

Для этого необходимо создать новый или открыть существующий проект. Рассмотрим этот процесс на примере синусоидального сигнала.

1. Сформируем синусоидальный сигнал с частотой 80 Гц, *обязательно указав имя файла*, в котором будут сохраняться все промоделированные результаты (рис. 2.17).

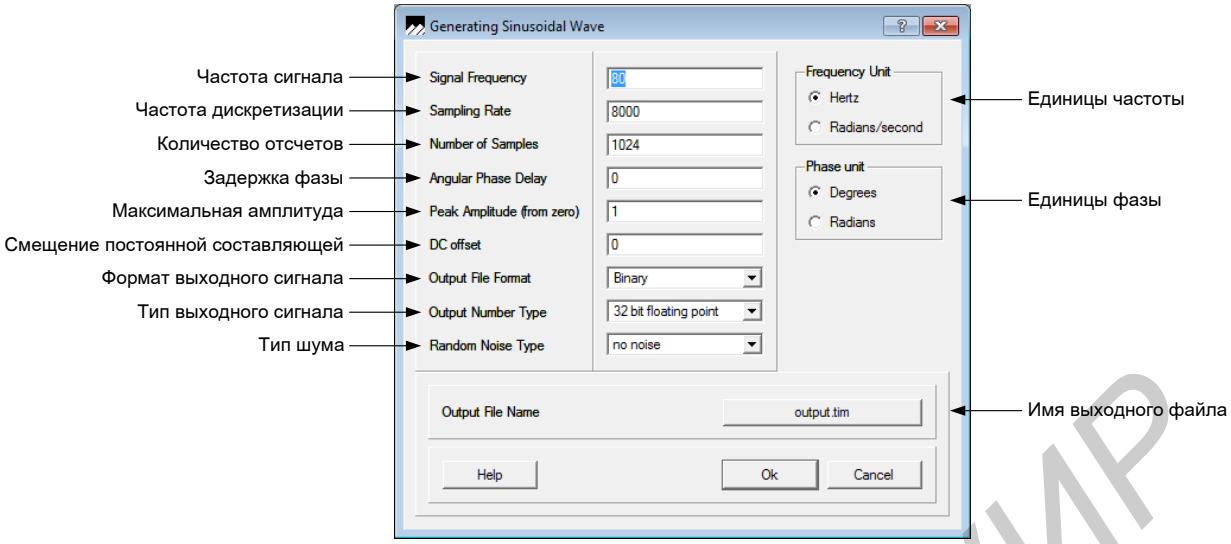


Рис. 2.17. Окно программы для задания параметров сигнала

Получаем сформированный сигнал:

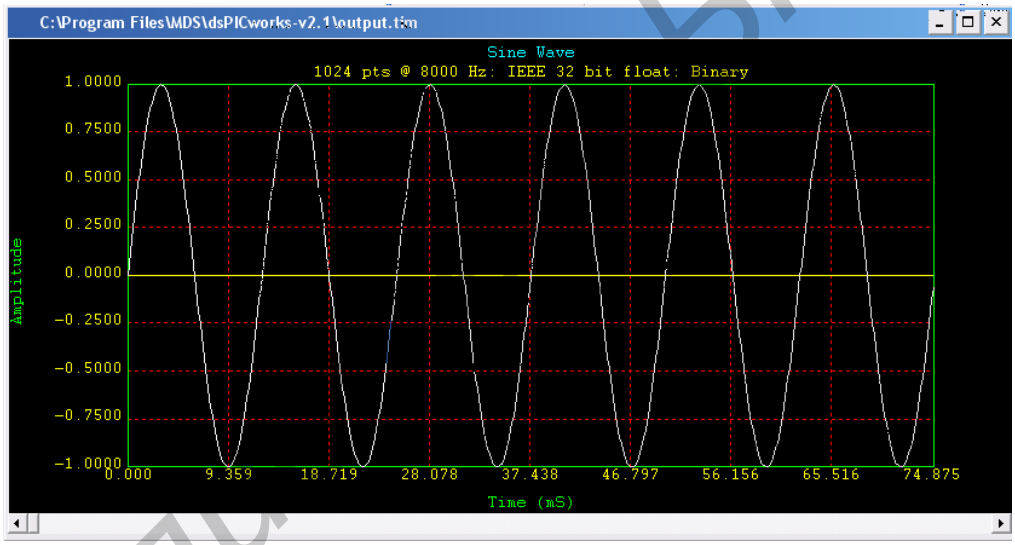


Рис. 2.18. Сформированный синусоидальный сигнал

2. Добавим к сигналу шум. Для чего открываем «**Generator>Noise Function**» и задаем параметры, как показано на рис. 2.19.

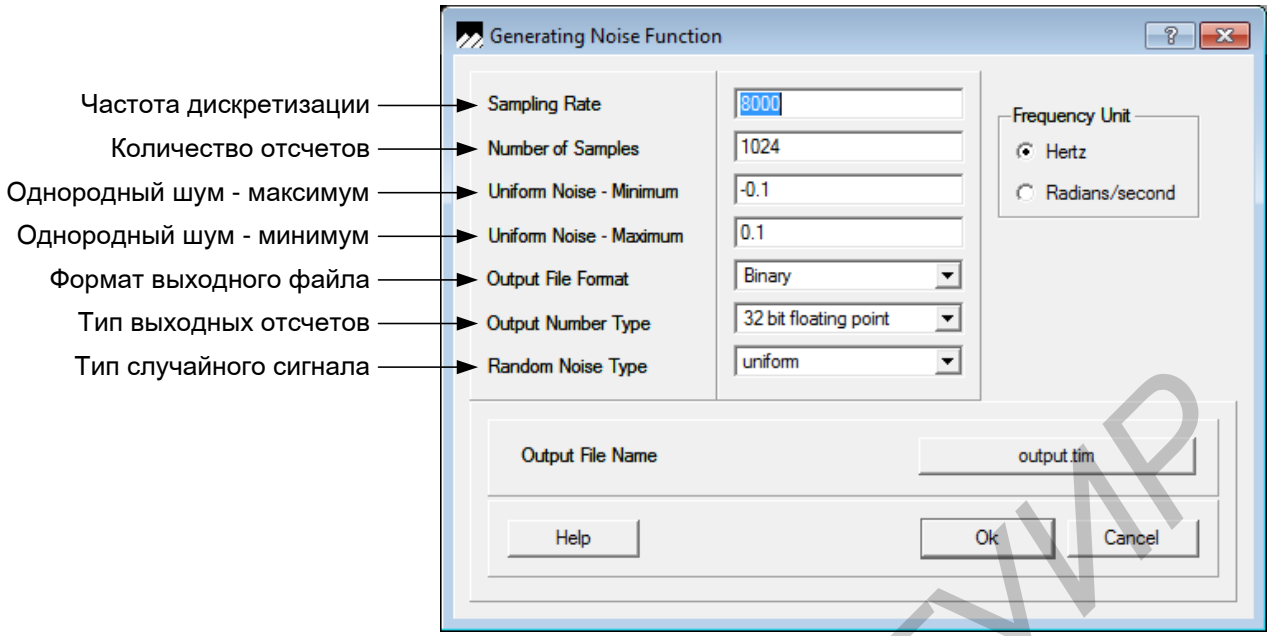


Рис. 2.19. Окно программы для задания параметров сигнала

Полученный сформированный сигнал показан на рис. 2.20.

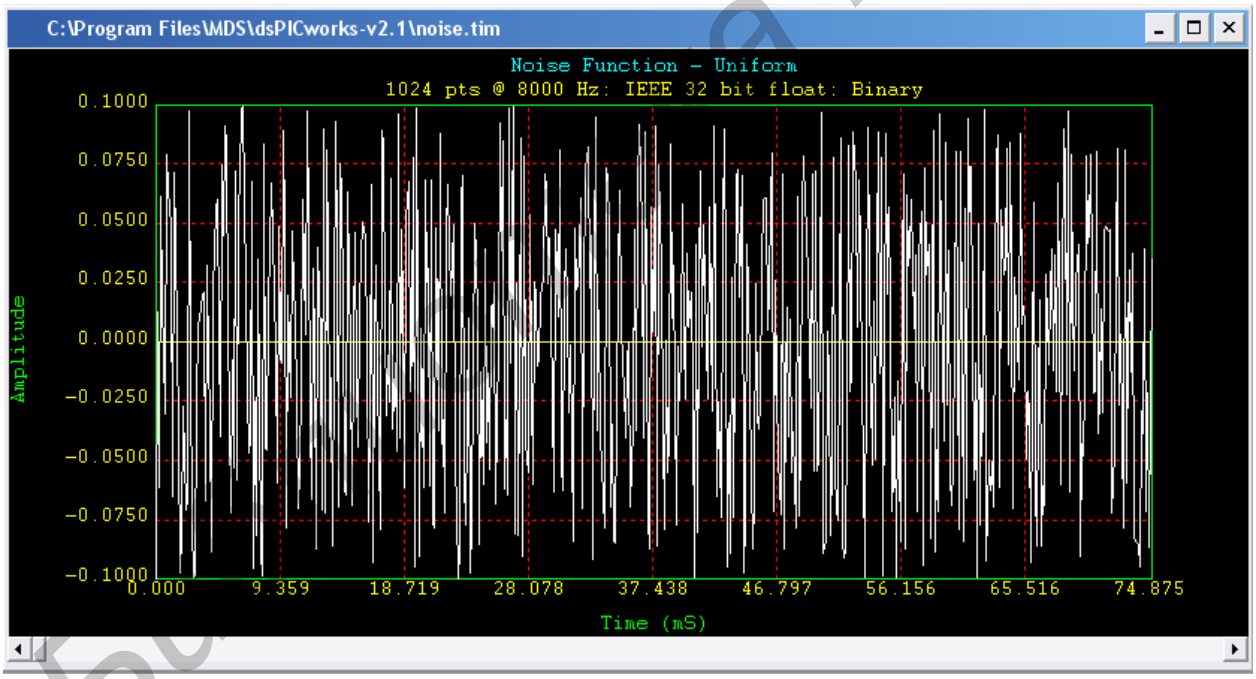


Рис. 2.20. Сформированный шум

3. Для получения смеси шума и синусоидального сигнала выбираем «**Operation>Arithmetic>Linear Combo…**». Из открытого окна указываем путь к сформированному сигналу и шуму (рис. 2.21).

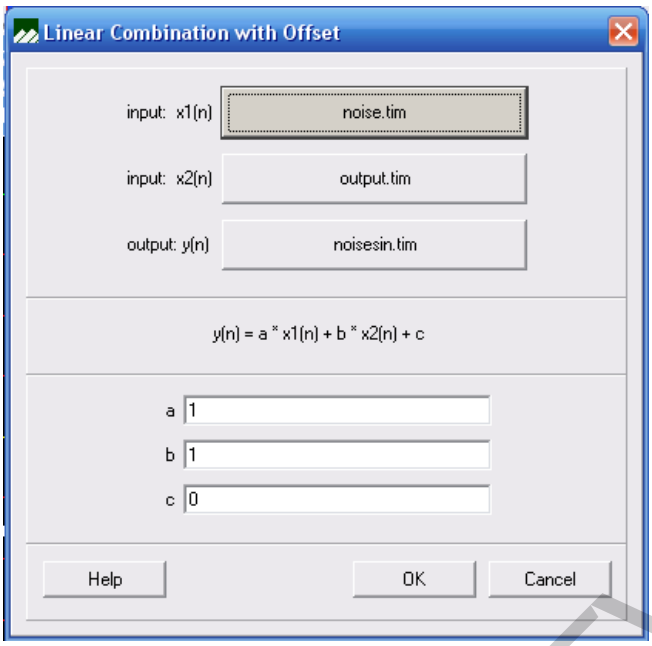


Рис. 2.21. Окно программы для формирования смеси сигнала и шума

Полученный сигнал показан на рис. 2.22.

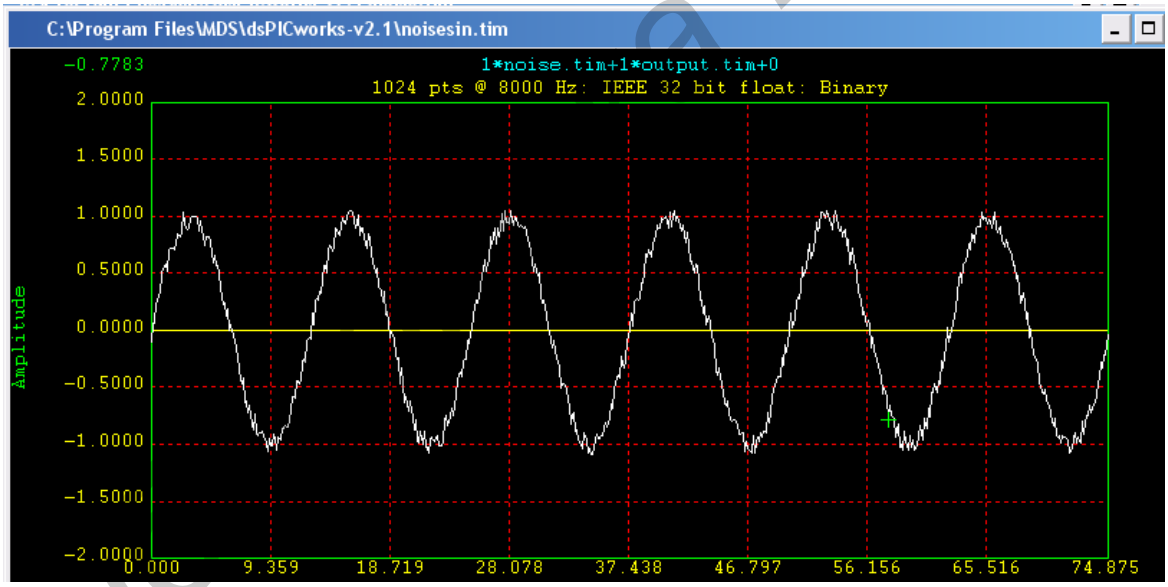


Рис. 2.22. Сформированная смесь сигнала и шума

4. Для передачи отсчетов полученного сигнала в память сигнального контроллера необходимо экспортировать полученные данные, как показано на рис. 2.23. Будет сформирован файл, который затем можно подключить к проекту в **MPLAB**.

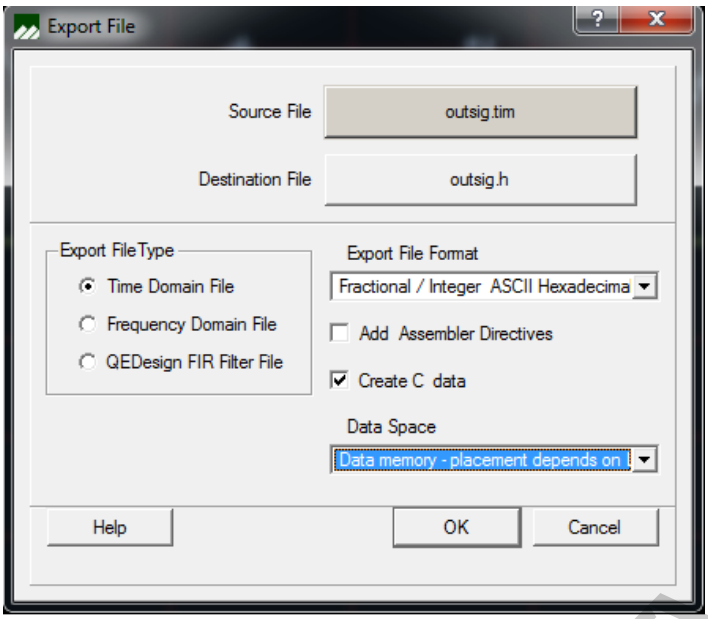


Рис. 2.23. Окно для формирования кода программы

Лабораторное задание

Требуется сформировать аналоговый сигнал, подаваемый на аудиовход ПК или осциллографа с помощью сигнального контроллера, используя его в режиме работы ШИМ. Частота выходного сигнала задается в пределах от 400 Гц до 4 кГц.

Сформировать таблицы значений уровней квантования восстанавливаемого сигнала в программе **dsPICWorks** и результат записать в буфер RAM сигнального контроллера. Размер буфера составляет 256 значений.

Необходимо отредактировать исходный код программы для управления ЦАП, приведенный ниже, согласно индивидуальному заданию.

Файл **main.c**

```
#include "p33Fxxxx.h"
#include <stdint.h>
#define SYS_FREQ 80000000UL
#define FCY SYS_FREQ/2
#define delay_us(x) __delay32(((x*FCY)/1000000L))
_FOSCSEL(0x02);
_FOSC(0xE2);
void __delay32(unsigned long cycles);
unsigned int i = 0,
sinus64[64] = {500, 549, 598, 645, 691, 736, 778, 817, 854, 887, 916, 941, 962, 978, 990,
998, 1000, 998, 990, 978, 962, 941, 916, 887, 854, 817, 778, 736, 691, 645, 598, 549, 500, 451,
402, 355, 309, 264, 222, 183, 146, 113, 84, 59, 38, 22, 10, 2, 0, 2, 10, 22, 38, 59, 84, 113, 146, 183,
222, 264, 309, 355, 402, 451},
```

```
sinus16[16]= {500, 691, 854, 962, 1000, 962, 854, 691, 500, 309, 146, 38, 0, 38, 146, 309};
// отсчеты формируемого сигнала

    int main(void)
    {                                   // инициализация модуля ШИМ
    OC1CONbits.OCM = 0b000;             // выключение модуля Output Compare
    OC1R = 100;                         // запись цикла первого импульса ШИМ
    OC1RS = 200;                        // запись цикла второго импульса ШИМ
                                        // Рабочий цикл ШИМ
    OC1CONbits.OCTSEL = 0;              // выбор Timer2 (таймер сравнения)
    OC1CONbits.OCM = 0b110;             // выбор модуля Output Compare
                                        // 110 - ШИМ режим без защиты от ошибки
                                        // 111 - ШИМ режим с защитой от ошибки
    T2CON = 0;                          // сброс состояния регистра T2CON
    T2CONbits.TON = 0;                  // отключить Timer2
    T2CONbits.TCS = 0;                  // выбор внутреннего источника тактовых сигналов
    T2CONbits.TGATE = 0;                // запрет Gated Timer mode
    T2CONbits.TCKPS = 0b00;             // выбор предделителя 1:1 таймера
    TMR2 = 0x00;                        // очистка таймера
    PR2 = 500;                          // загрузка периода
    IPC1bits.T2IP = 1;                  // задание приоритета прерывания Timer2
    IFS0bits.T2IF = 0;                  // очистка флага прерывания таймера Timer2
    IEC0bits.T2IE = 1;                  // разрешение таймера прерывания Timer2
    T2CONbits.TON = 1;                  // запуск Timer2
                                        // 100Hz, 1.5V
    while(1){ };
    }
    // обработчик прерывания по Timer2
    void __attribute__((__interrupt__)) _T2Interrupt( void )
    {
    OC1RS = sinus64[i];                 // запись очередного цикла ШИМ
    i++;
    i %= 64;
    IFS0bits.T2IF = 0;                  // очистка флага прерывания таймера Timer2
    }
```

Требования к отчету

1. Формулировка индивидуального задания, полученного у преподавателя.

2. Описание решения задачи с привлечением аналитических формул, временных диаграмм, функциональной и принципиальной схем, а также блок-схем алгоритмов.

3. Текст отлаженной программы с комментариями.

4. Выводы по работе.

Контрольные вопросы и задания

1. Поясните принцип организации цифроаналогового преобразования на основе ШИМ, используя сигнальный контроллер.
2. Приведите последовательность настройки модуля сравнения в режиме ШИМ.
3. Как осуществляется формирование сигнала заданной формы с помощью ШИМ?

2.4. *Лабораторная работа №7*. Реализация алгоритмов цифровой фильтрации на сигнальном контроллере

Цель работы

1. Изучение принципов построения цифровых фильтров на сигнальном контроллере.

2. Приобретение практических навыков в аппаратно-программной реализации цифровых фильтров.

Краткие теоретические сведения

Цифровая фильтрация – один из мощных инструментов в цифровой обработке сигналов (ЦОС). С помощью цифровых фильтров появилась возможность решать такие задачи, которые было бы чрезвычайно сложно или даже невозможно решить с помощью аналоговых устройств. Кроме того, характеристики цифрового фильтра могут легко изменяться программно, поэтому они широко применяются в системах телекоммуникаций, устройствах адаптивной фильтрации, подавлении шумов и распознавании речевых сигналов.

Процесс проектирования цифровых фильтров состоит из тех же этапов, что и процесс проектирования аналоговых фильтров. Сначала формулируются требования к желаемым характеристикам фильтра, на основании которых затем рассчитываются параметры фильтра.

На рис. 2.24 представлены основные этапы обработки сигнала, содержащего высокочастотный шум. Вначале сигнал оцифровывается с помощью АЦП для получения выборки отсчетов $x(n)$. Далее эта выборка поступает на цифровой ФНЧ. Отсчеты выходных данных $y(n)$ с помощью ЦАП преобразуются в выходной аналоговый сигнал.

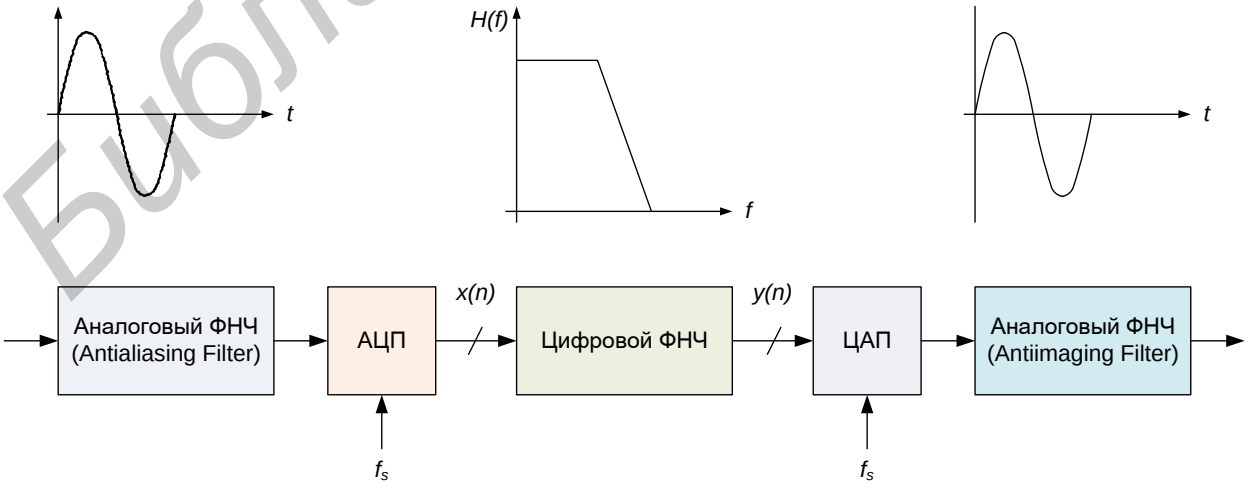


Рис. 2.24. Этапы цифровой фильтрации

При проектировании цифровых фильтров необходимо учитывать то, что они должны работать в реальном масштабе времени. В соответствии с этим в зависимости от частоты дискретизации и сложности фильтра накладывается ряд ограничений на параметры сигнального контроллера.

Основным моментом является то, что сигнальный контроллер должен проводить все вычисления в течение интервала дискретизации, чтобы быть готовым к обработке следующего отсчета данных. Пусть ширина полосы частот обрабатываемого сигнала равна f. Тогда частота дискретизации АЦП f_s и должна быть по крайней мере в два раза больше, т. е. $2f$. Интервал дискретизации равен $1/f_s$. Все вычисления, связанные с реализацией фильтра (включая все дополнительные операции), должны быть закончены в течение этого интервала. Время вычислений зависит от числа звеньев фильтра и быстродействия и эффективности сигнального контроллера. Каждое звено при реализации фильтра требует одну операцию умножения и одну операцию сложения (умножения с накоплением).

Высокопроизводительный сигнальный контроллер dsPIC33f обладает дополнительными возможностями для реализации на них цифровых фильтров, такими, как циклическая адресация и выполнение операции умножения с накоплением за один такт, что минимизирует количество инструкций.

Например, dsPIC33f с быстродействием 40 MIPS способен выполнить операцию умножения с накоплением при реализации одного каскада фильтра за 25 нс. Он затрачивает N+5 инструкций при реализации фильтра с количеством каскадов N. Для 100-каскадного фильтра полное время вычисления составляет приблизительно 2,6 мкс. Это соответствует максимально возможной частоте дискретизации 384 кГц, ограничивая, таким образом, ширину полосы частот обрабатываемого сигнала несколькими сотнями килогерц.

Основные требования к сигнальному контроллеру для реализации на нем цифровых фильтров в реальном масштабе времени:

1) полоса сигнала = f_a;

2) частота дискретизации $f_s > 2f_a$;

3) период дискретизации = $1/f_s$;

4) время вычисления фильтра и дополнительные операции должны быть меньше периода дискретизации, который зависит от:

– числа коэффициентов фильтра;

– скорости операций умножения с накоплением (MAC);

– эффективности ЦОС;

– поддержки циклических буферов;

– отсутствия дополнительных операций.

Фильтры с конечной импульсной характеристикой (КИХ)

Существует два основных типа цифровых фильтров: фильтры с конечной импульсной характеристикой (КИХ) и фильтры с бесконечной импульсной характеристикой (БИХ).

Как следует из терминологии, эта классификация относится к импульсным характеристикам фильтров. Изменяя вес коэффициентов и число звеньев КИХ-фильтра, можно реализовать практически любую частотную характеристику. КИХ-фильтры могут иметь такие свойства, которых невозможно достичь методами аналоговой фильтрации (в частности, совершенно линейную фазовую характеристику). Но высокоэффективные КИХ-фильтры строятся с большим числом операций умножения с накоплением и поэтому требуют использования быстрых и эффективных сигнальных контроллеров или сигнальных процессоров (*Digital Signal Processor – DSP*). БИХ-фильтры схожи по характеристикам с аналоговыми фильтрами, а их импульсная характеристика имеет бесконечную длительность. Благодаря использованию обратной связи БИХ-фильтры могут быть реализованы с использованием меньшего количества коэффициентов, чем КИХ-фильтры.

Проектирование КИХ-фильтров с использованием программы MICROCHIP Digital Filter Design

Для проектирования цифровых фильтров существует множество программ, одной из которых является **MICROCHIP Digital Filter Design**. На ее примере спроектируем низкочастотный (НЧ) КИХ-фильтр. Основные параметры, которые задаются при его проектировании, – это частота дискретизации, полоса пропускания, полоса затухания (задержки), неравномерность полосы пропускания, неравномерность полосы затухания (показаны на рис. 2.25).

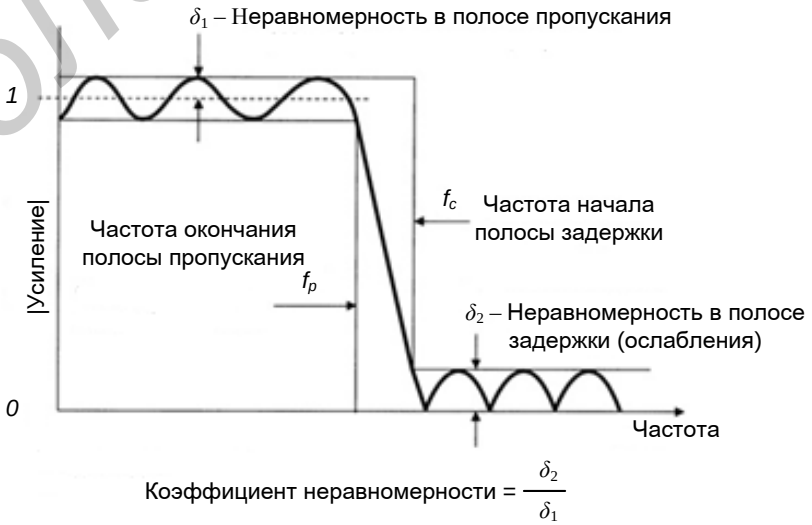


Рис. 2.25. Иллюстрация входных данных для проектирования КИХ-фильтра

Выберем в меню программы «**Filter>Lowpass Filter**». Зададим частоту дискретизации 44,1 кГц. Граничная частота полосы пропускания составляет 80 Гц. Полоса задержки начинается при 1 кГц, неравномерность полосы пропускания равна 0,5 дБ, а неравномерность полосы задержки (ослабление) – 10 дБ. Также необходимо определить разрядность коэффициентов, которая в данном случае составляет 16 разрядов, принимая во внимание, что используется 16-разрядный сигнальный контроллер с фиксированной точкой.

Заданные параметры для фильтра показаны на рис. 2.26.

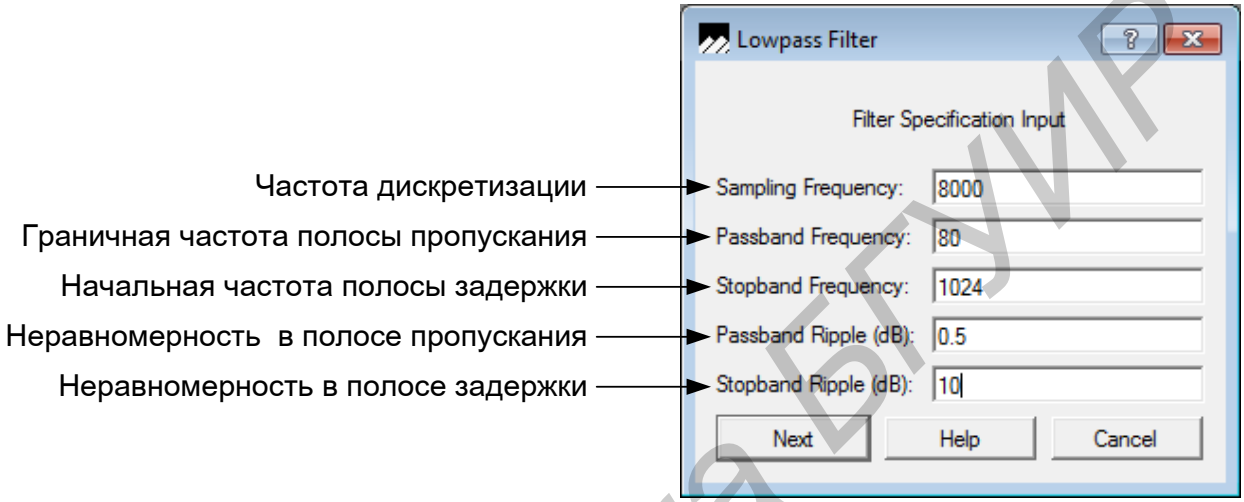


Рис. 2.26. Окно программы для задания параметров фильтра

В следующем окне зададим аналоговый эквивалент и порядок КИХ-фильтра (Хэмминга и др.). Выберем Хэмминга 10-го порядка. Частотные характеристики полученного фильтра показаны на рис. 2.27.

Если необходимо продолжить работу с фильтром в **dsPICWorks**, то сохраняем полученные результаты проектирования в формате (*.flt), для этого нажать соответствующую иконку на панели задач 

. Если нужно подключить полученные данные к проекту с программой для сигнального контроллера, то они сохраняются в виде кода, для этого выбираем «**Codegen > C code**» и сохраняем файл в необходимой папке.

Лабораторное задание

Требуется сформировать аналоговый сигнал в программе **dsPICWorks**. Частота выходного сигнала задается в пределах от 400 Гц до 4 кГц. Наложить на него шум. Сформировать таблицы значений уровней квантования восстанавливаемого сигнала, результат записать в буфер RAM сигнального контроллера. Размер буфера должен иметь 256 значений. Спроектировать фильтр для очистки сигнала от шума. Полученные коэффициенты фильтра записать в буфер RAM. Загрузить программу в сигнальный контроллер и с помощью отладчика

проверить ее выполнение. Проанализировать и убедиться в правильности обработки сигнала путем импортирования полученных данных в буфер RAM сигнального контроллера, например в **MS Excel**.

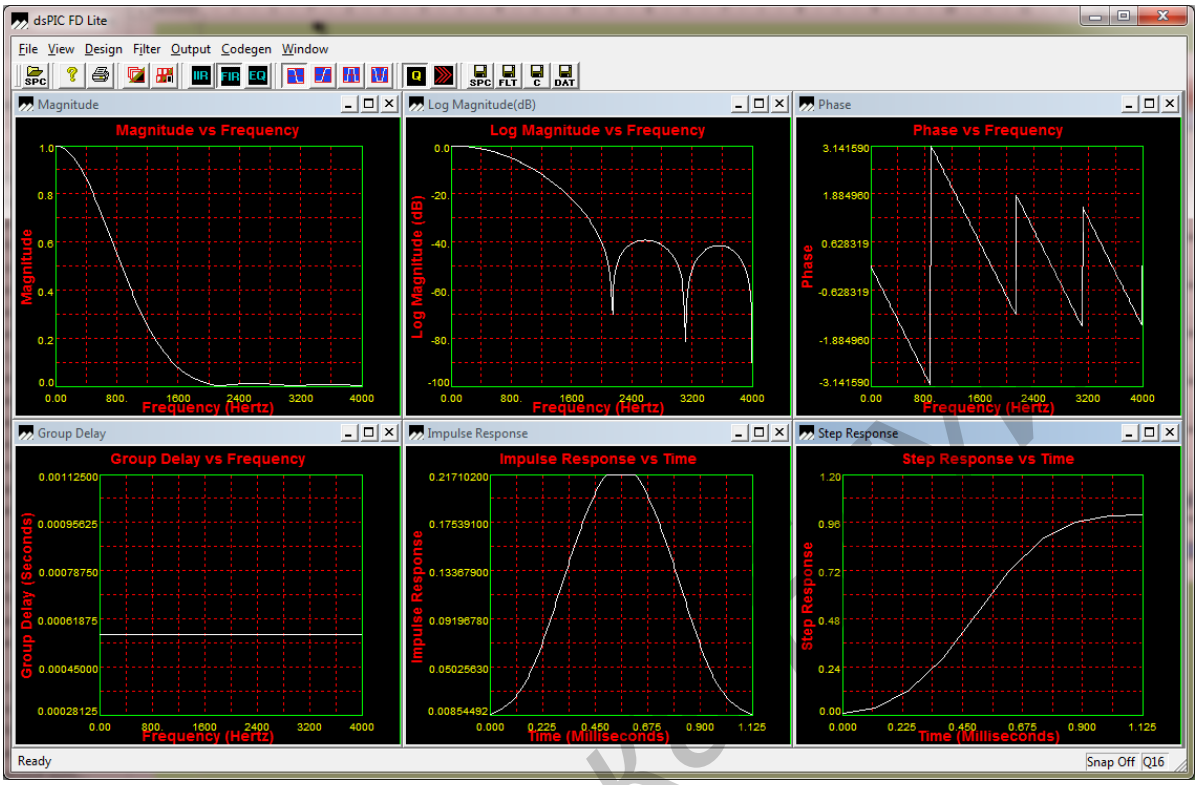


Рис. 2.27. Окно программы с характеристиками полученного фильтра

Согласно индивидуальному заданию необходимо отредактировать исходный код программы для реализации цифрового фильтра, который приведен ниже.

Файл **main.c**

```
#include "p33Fxxxx.h"
#include "dsp.h"

    /* настройка тактового генератора и других параметров сигнального контроллера*/
    _FOSCSEL(FNOSC_FRC);
    _FOSC(FCKSM_CSECMD & OSCIOFNC_OFF & POSCMD_XT);
    _FWDT(FWDTEN_OFF);
    _FPOR(FPWRT_PWR1);
    _FGS(GCP_OFF);

    /* объявление переменных*/
int     incr;
```

```
int     countCycles;
extern void    initOC1(void);
extern void    __attribute__((__interrupt__)) _T20Interrupt(void);

        /* определение основных параметров фильтра
#define BLOCK_LENGTH 256                    // число выборок
#define LOWPASSFILTER                       // ФНЧ

        /* расположение коэффициентов фильтра в памяти программ*/
#define FILTERCOEFFS_IN_PROGMEM
extern fractional square1k[256];            // объявление массива со значениями выборки
extern FIRStruct lowpassexample_psvFilter;

        /* объявление массива с выходными значениями сигнала*/
fractional FilterOut[BLOCK_LENGTH];

        /*начало основной программы*/
int main(void)
{
        /*настройка тактового генератора */
        PLLFBD=38;                          // M=40
        CLKDIVbits.PLLPOST=0;               // N1=2
        CLKDIVbits.PLLPRE=0;                // N2=2
        OSCTUN=0;
        RCONbits.SWDTEN=0;
        __builtin_write_OSCCONH(0x03);
        __builtin_write_OSCCONL(0x01);
        while (OSCCONbits.COSC != 0b011);
        while(OSCCONbits.LOCK!=1) {};
        TRISD=0xFF00;
        incr=0;
        countCycles=0;
        initOC1();                          // инициализация ШИМ

        FIRDelayInit(&lowpassexample_psvFilter);

        /* выполнение фильтрации входного сигнала*/
        FIR(BLOCK_LENGTH,&FilterOut[0],&square1k[0],&lowpassexample_psvFilter);
        while (1)
        {
                if (countCycles == 10)      // формирование периода ШИМ
                {
```

```
                incr++;
                if (incr==256) incr=0;
            }
            countCycles=0;
        }
}       return 0;
}

        /* Ожидание прерывания для загрузки в модуль сравнения нового значения (ШИМ)*/
void __attribute__((__interrupt__)) _T2Interrupt( void )
{
        countCycles++;
        OC1RS = FilterOut[incr];
        IEC0bits.T2IE = 1;
        IFS0bits.T2IF = 0;
}

        /* инициализация модуля сравнения для ШИМ*/
void initOC1(void)
{
        OC1CONbits.OCM = 0b000;
        OC1R = FilterOut[incr];
        OC1RS = FilterOut[incr];
        OC1CONbits.OCTSEL = 0;
        OC1CONbits.OCM = 0b110;
        T2CONbits.TON = 0;
        T2CONbits.TCS = 0;
        T2CONbits.TGATE = 0;
        T2CONbits.TCKPS = 0b00;
        TMR2 = 0x00;
        PR2 = 65535;
        IPC1bits.T2IP = 0x01;
        IFS0bits.T2IF = 0;
        IEC0bits.T2IE = 1;
        T2CONbits.TON = 1;
}
```

Требования к отчету

1. Формулировка индивидуального задания, полученного у преподавателя.

2. Описание решения задачи с привлечением аналитических формул, временных диаграмм, схем функциональной и принципиальной, а также схем алгоритмов.

3. Текст отлаженной программы с комментариями.

4. Выводы по работе.

Контрольные вопросы и задания

1. Поясните этапы построения цифровых фильтров на сигнальном контроллере.

2. Приведите последовательность проектирования КИХ-фильтра.

3. Какие характеристики цифровых фильтров вы знаете?

4. Назовите и поясните факторы, влияющие на точность реализации цифровых фильтров.

5. Поясните особенности аппаратной реализации цифровых фильтров на сигнальном контроллере.

2.5. *Лабораторная работа №8*. Реализация алгоритмов быстрого преобразования Фурье на сигнальном контроллере

Цель работы

1. Изучение принципов реализации алгоритмов спектрального анализа сигналов в сигнальных контроллерах.
2. Приобретение практических навыков реализации алгоритмов спектрального анализа сигналов в сигнальных контроллерах.

Краткие теоретические сведения

Дискретное преобразование Фурье (ДПФ) закладывает основы многих методов, применяемых в области цифровой обработки сигналов. Преобразование Фурье позволяет сопоставить сигналу, заданному во временной области, его эквивалентное представление в частотной области и наоборот, по известной частотной характеристике сигнала с помощью обратного преобразования Фурье можно определить соответствующий сигнал во временной области.

В дополнение к частотному анализу такие преобразования применяются при проектировании фильтров. Частотная характеристика фильтра может быть получена с помощью преобразования Фурье его импульсной характеристики. И наоборот, если определена частотная характеристика фильтра, то требуемая импульсная характеристика может быть получена с помощью обратного преобразования Фурье над его частотной характеристикой.

Фундаментальное уравнение для получения N-точечного ДПФ выглядит следующим образом:

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\, e^{-j2\pi n/N} = \frac{1}{N}\sum_{n=0}^{N-1} x(n)[\cos(2\pi nk/N) - j \cdot \sin(2\pi nk/N)],$$

где $X(k)$ – частотный выход ДПФ в k-й точке спектра, k находится в диапазоне от 0 до $N-1$;

N – число отсчетов при вычислении ДПФ;

$x(n)$ – n-й отсчет во временной области, где n также находится в диапазоне от 0 до $N-1$.

В общем уравнении $x(n)$ может быть вещественным или комплексным. Выходной спектр ДПФ $X(k)$ является результатом вычисления свертки между выборкой, состоящей из входных отсчетов во временной области, и набором из N пар гармонических базисных функций (косинус и синус).

Подобная процедура применяется при вычислении обратного ДПФ для восстановления отсчетов во временной области $x(n)$ из отсчетов в частотной области $X(k)$.

Соответствующее уравнение выглядит следующим образом:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, e^{j2\pi n/N} = \sum_{k=0}^{N-1} X(k)[\cos(2\pi nk/N) + j \cdot \sin(2\pi nk/N)].$$

Быстрое преобразование Фурье (БПФ, FFT) является не более чем алгоритмом для ускоренного вычисления ДПФ путем сокращения требуемого числа операций умножения и сложения. Концепция БПФ заключается в том, что ДПФ в развернутом виде может быть сильно упрощено, если использовать свойства симметрии и периодичности коэффициентов поворота:

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\, e^{-j2\pi n/N} = \frac{1}{N}\sum_{n=0}^{N-1} x(n) \cdot W_N\, nk;\ W_N = e^{-j2\pi/N}.$$

В отличие от ДПФ БПФ требует только $(N/2)log_2(N)$ умножений комплексных чисел.

Алгоритм БПФ по основанию 2 разделяет полное вычисление ДПФ на комбинацию 2-точечных ДПФ. Каждое 2-точечное ДПФ содержит базовую операцию умножения с накоплением, называемую «бабочкой» и иллюстрируемую на рис. 2.28.

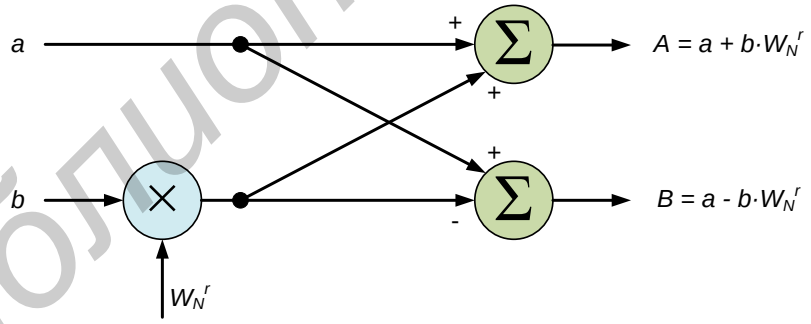


Рис. 2.28. Базовая операция «бабочка» в алгоритме БПФ с прореживанием по времени

Восьмиточечное БПФ с прореживанием во времени (*decimation-in-time*, DIT) вычисляет окончательный результат с использованием трех каскадов. Восемь входных отсчетов из временной области сначала разделяются (или прореживаются) на четыре группы 2-точечных ДПФ. Затем четыре 2-точечных ДПФ объединяются в два 4-точечных ДПФ. Затем два 4-точечных ДПФ объединяются для того, чтобы получить окончательный результат $X(k)$. Подробно процесс рассмотрен на рис. 2.29, где показаны все операции умноже-

ния и суммирования. Нетрудно заметить, что базовая операция «бабочки» 2-точечного ДПФ формирует основу для всего вычисления. Вычисление осуществляется в трех каскадах. После того как заканчивается вычисление первого каскада, нет необходимости сохранять какие-либо предыдущие результаты.

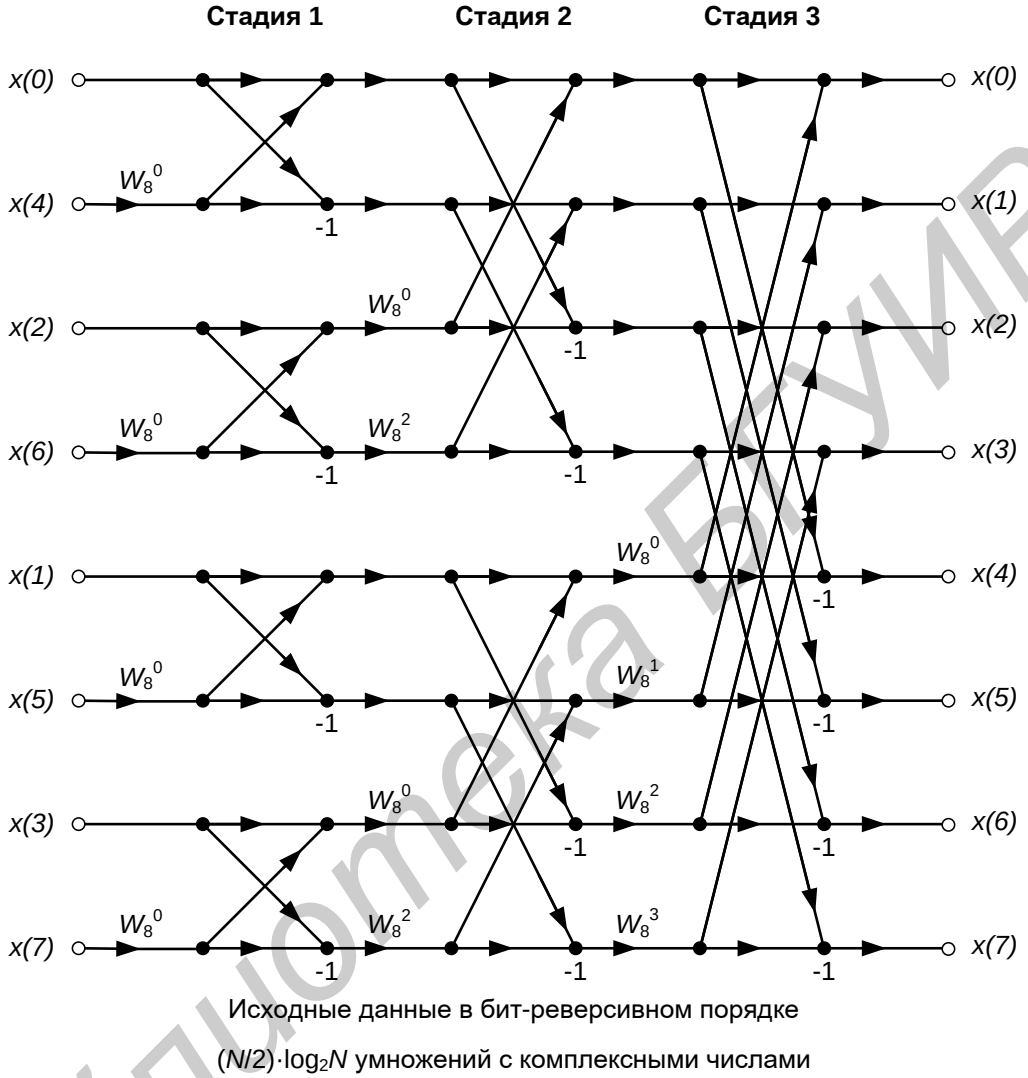


Рис. 2.29. Алгоритм 8-точечного БПФ с прореживанием по времени

Результаты вычисления первого каскада могут быть сохранены в тех же самых регистрах или ячейках памяти, которые первоначально хранили исходные отсчеты из временной области $x(n)$. Точно так, когда заканчивается вычисление второго каскада, результаты вычисления первого каскада могут быть удалены. Таким же образом осуществляется и вычисление последнего каскада: заменой в памяти промежуточного результата вычисления предыдущего каскада. Чтобы алгоритм функционировал, входные отсчеты по времени $x(n)$ должны быть упорядочены определенным образом с использованием алгоритма реверсии бит.

Алгоритм реверсии бит, используемый для реализации прореживания по времени, представлен в табл. 2.6.

Таблица 2.6

Десятичное число	0	1	2	3	4	5	6	7
Двоичный эквивалент	000	001	010	011	100	101	110	111
Двоичный эквивалент с реверсированием	000	100	010	110	001	101	011	111
Десятичный эквивалент с реверсированием	0	4	2	6	1	5	3	7

Десятичное число преобразуется в двоичный эквивалент. Затем двоичные разряды располагаются в обратном порядке и преобразуются обратно в десятичное число. Реверсия бит выполняется аппаратно в генераторе адреса данных сигнального контроллера, упрощая таким образом код программы и сокращая количество дополнительных операций, что ускоряет вычисления.

Необходимо отметить, что алгоритмы, требуемые для вычисления обратного БПФ, почти идентичны тем, которые необходимы для вычисления прямого БПФ, если использовать комплексное БПФ.

Проверка вычисления комплексного БПФ полученных отсчетов из временной области $x(n)$ возможна путем обратного вычисления БПФ с отсчетами из частотной области $X(k)$. В конце вычислений должны быть получены первоначальные отсчеты из временной области $Re\ x(n)$, при этом мнимая часть $Im\ x(n)$ должна быть равна нулю (в пределах ошибки математического округления).

Аппаратная реализация и время выполнения алгоритмов БПФ

В общем случае требования по используемой памяти для *N*-точечного БПФ следующие:

– число *N* ячеек памяти RAM для хранения вещественных данных;

– *N* ячеек памяти RAM для хранения мнимых данных;

– *N* ячеек для синусоидальных базисных функций (иногда упоминаемых как вращающие коэффициенты).

Дополнительный объем памяти может потребоваться в случае использования взвешивания с использованием оконных функций.

В 16-разрядном цифровом сигнальном процессоре с фиксированной точкой после умножения формируется 32-разрядное слово. Семейство цифровых сигнальных контроллеров характеризуется расширенным динамическим диапазоном, который реализуется в операциях умножения с накоплением посредством 40-разрядного внутреннего регистра аккумулятора.

Для обеспечения работы сигнального контроллера в реальном масштабе времени полный расчет БПФ должен выполняться в промежутке, соответствующем времени накопления одного пакета данных. Предполагается, что, пока производится вычисление БПФ текущего пакета данных, сигнальный контрол-

лер должен накапливать данные для следующего преобразования. Непрерывное получение данных облегчается благодаря возможностям гибкой адресации данных в сигнальных контроллерах в сочетании с использованием различных каналов прямого доступа к памяти (DMA).

Рассмотрим сигнальный контроллер, который вычисляет 1024-точечное 32-разрядное комплексное БПФ с плавающей запятой за 69 мкс. Очевидно, что максимальная частота дискретизации равна 1024/69 мкс = 14,8 MSPS. Это подразумевает, что сигнал имеет ширину полосы частот меньше 7,4 МГц. Данные вычисления предполагают, что нет дополнительных затрат процессорного времени, связанных с БПФ, или нет задержек, связанных с передачей данных.

Приведенный пример дает оценку максимальной ширины полосы сигнала, который может быть обработан сигнальным контроллером. Другой подход состоит в том, чтобы, задаваясь шириной полосы сигнала, разработать требования к сигнальным контроллерам для обработки сигнала в рассматриваемой полосе. Если ширина полосы частот сигнала известна, требуемая частота дискретизации может быть определена путем ее умножения на коэффициент 2…2,5 (увеличение частоты дискретизации может требоваться для ослабления требований к АЦП и ФНЧ, устраняющему эффект наложения спектра (*antialiasing filter*)). Следующим шагом определяется число точек БПФ, требуемое для достижения желаемой разрешающей способности по частоте. Разрешающая способность по частоте получается делением частоты дискретизации f_s на число точек БПФ N.

Приведем основные требования:

– ширина полосы сигнала, f;

– частота дискретизации, f_s;

– количество точек БПФ, N;

– разрешающая способность по частоте, $f_s\ /N$;

– максимальное время вычисления N-точечного БПФ, N/f_s;

– вычисление в формате с фиксированной или плавающей запятой;

– время выполнения алгоритма БПФ по основанию 2;

– выигрыш БПФ в отношении сигнал/шум, $10log_{10}(N/2)$;

– требования взвешивания с использованием оконной функции (*Windowing*).

Расширение спектра анализируемого сигнала и взвешивание с использованием оконной функции

Расширение спектра анализируемого сигнала при вычислении БПФ можно проиллюстрировать на синусоидальном сигнале.

Следует напомнить, что вычисление ДПФ предполагает, что выборка повторяется бесконечное число раз до и после исследуемого фрагмента сигнала,

формируя таким способом бесконечный непрерывный периодический сигнал, как показано на рис. 2.30.

При таких условиях форма входного сигнала представляет собой непрерывную синусоидальную функцию, и на выходе ДПФ или БПФ будет один ненулевой частотный отсчет, соответствующий частоте входного сигнала.

На рис. 2.31 отражена ситуация, когда в выборке нет целого числа периодов синусоидального сигнала. Разрывы, которые образуются в конечных точках выборки, приводят к расширению спектра анализируемого сигнала вследствие появления дополнительных гармоник. В дополнение к появлению боковых лепестков происходит расширение основного лепестка, что приводит к снижению разрешающей способности по частоте.

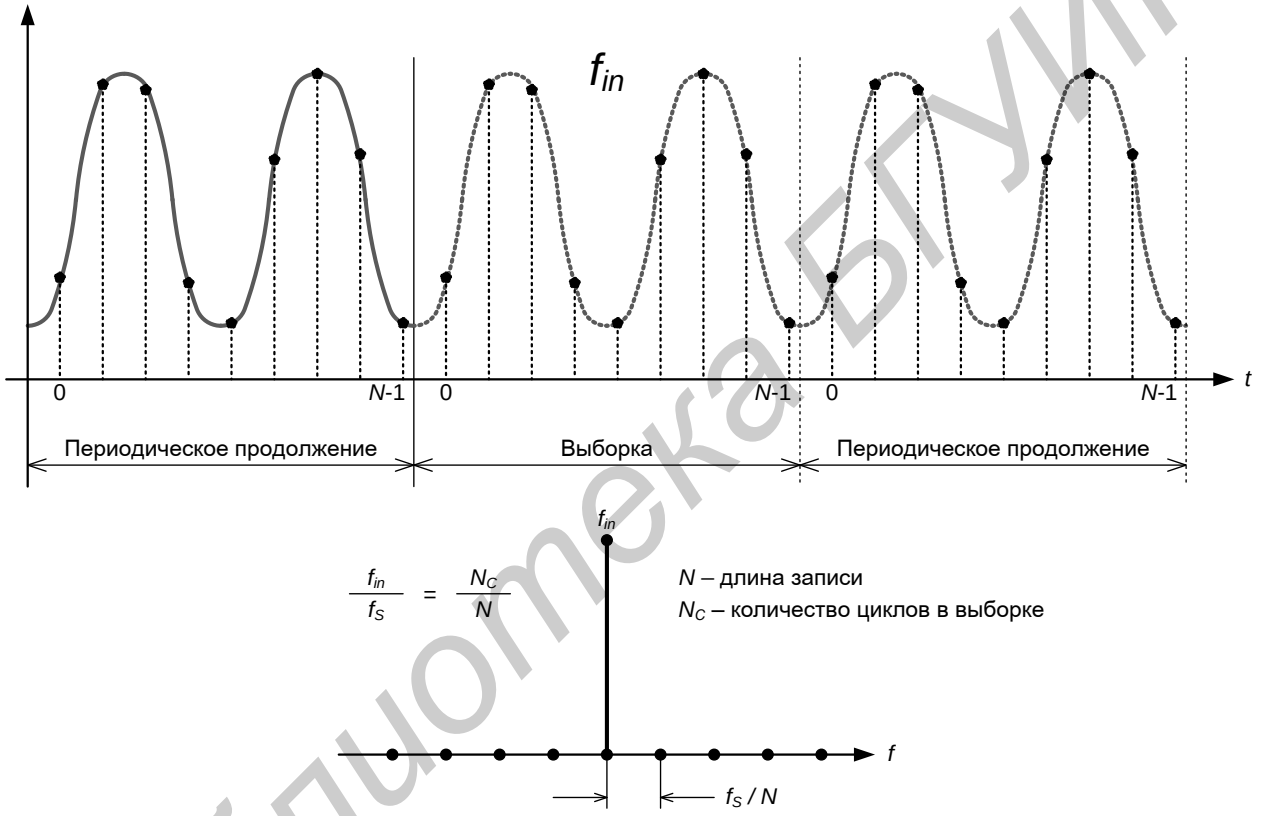


Рис. 2.30. БПФ синусоидального сигнала с целым числом периодов в выборке

Этот процесс эквивалентен перемножению входного синусоидального сигнала с прямоугольным импульсом, который имеет известную частотную характеристику $\sin(x)/x$ и связанные с этим широкий основной лепесток и боковые лепестки.

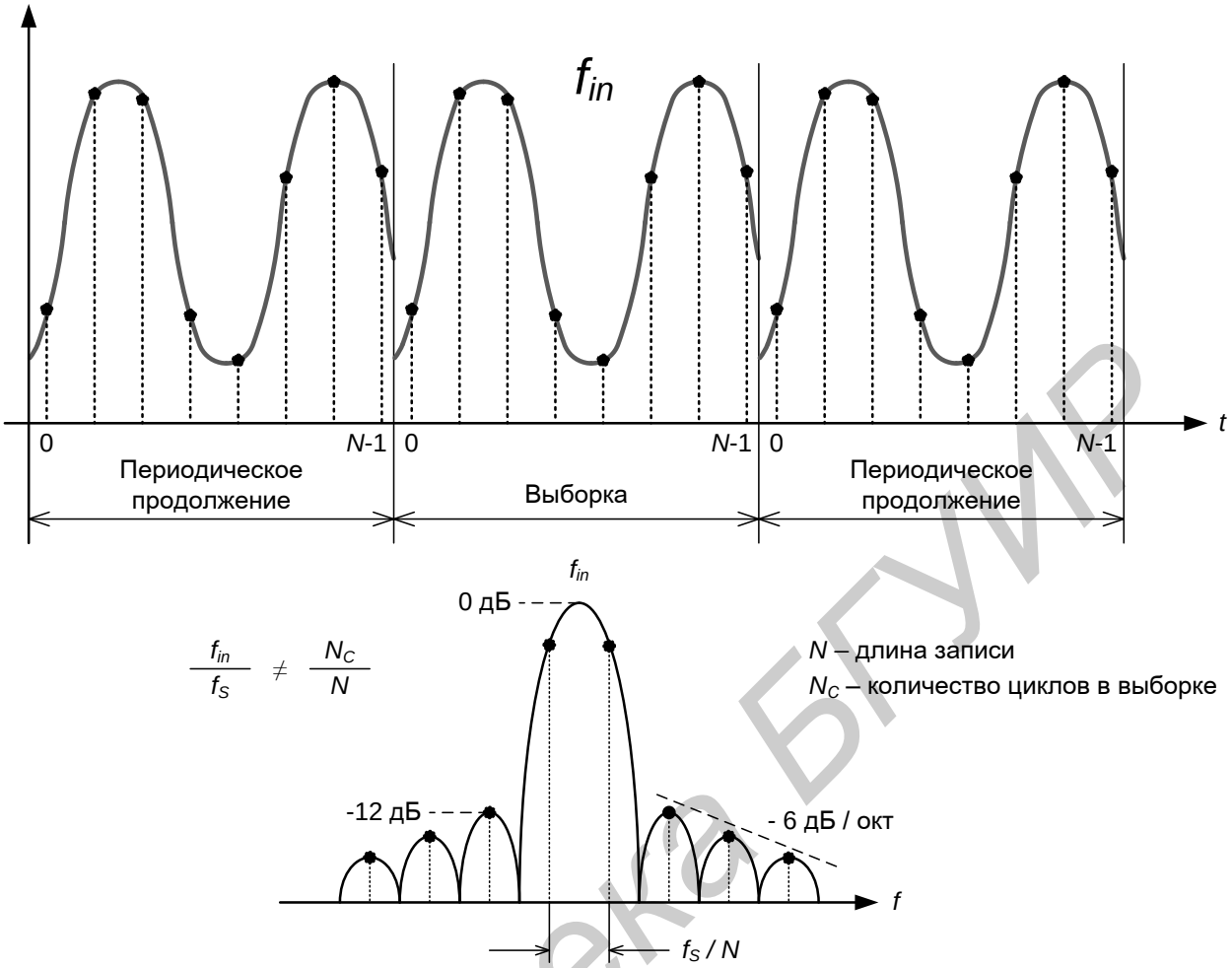


Рис. 2.31. БПФ синусоидального сигнала с нецелым числом периодов в выборке

Поскольку в практических приложениях БПФ для спектрального анализа точные входные частоты не известны, следует предпринять определенные шаги к уменьшению боковых лепестков. Оно достигается выбором оконной функции с более сложной формой. Входные отсчеты по времени умножаются на соответствующую функцию окна, что влечет за собой обнуление сигнала на краях выборки, как показано на рис. 2.32.

Выбор функции окна является прежде всего компромиссом между увеличением ширины основного лепестка и размером боковых лепестков.

Математические функции, описывающие четыре популярные оконные функции (Хемминга, Блэкмана, Хеннинга и минимальная 4-элементная Блэкмана – Харриса):

Хемминга:

$$w(n) = 0{,}54 - 0{,}46 \cdot \cos\left[\frac{2\pi n}{N}\right];$$

Блэкмана:

$$w(n) = 0{,}42 - 0{,}5 \cdot \cos\left[\frac{2\pi n}{N}\right] + 0{,}08 \cdot \cos\left[\frac{4\pi n}{N}\right];$$

Хеннинга:

$$w(n) = 0{,}5 - 0{,}5 \cdot \cos\left[\frac{2\pi n}{N}\right];$$

минимальная 4-элементная Блэкмана – Харриса:

$$w(n) = 0{,}35875 - 0{,}48829 \cdot \cos\left[\frac{2\pi n}{N}\right] +$$

$$+0{,}14128 \cdot \cos\left[\frac{4\pi n}{N}\right] - 0{,}01168 \cdot \cos\left[\frac{6\pi n}{N}\right].$$

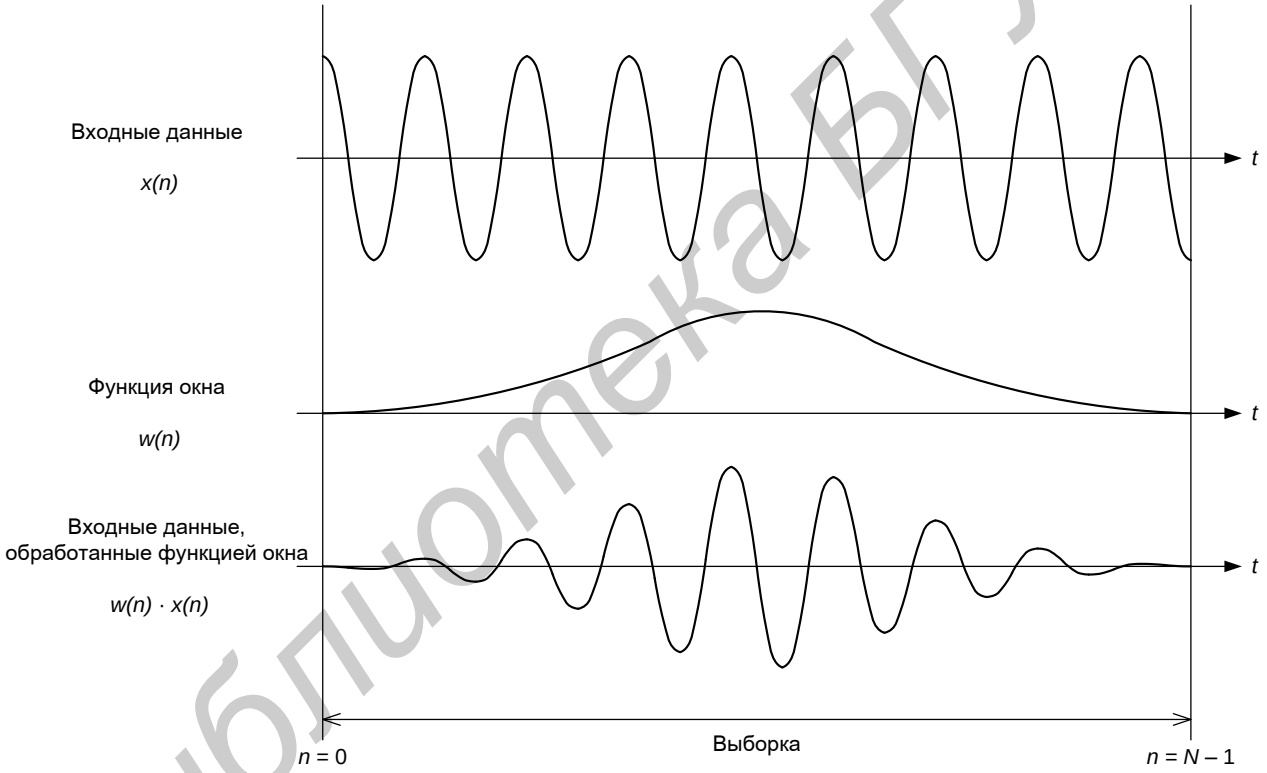


Рис. 2.32. Взвешивание с использованием функции окна для уменьшения эффекта расширения спектра

Оцифрованные оконные функции обычно вычисляются предварительно и сохраняются в памяти сигнального контроллера с целью минимизации вычислений непосредственно при реализации БПФ. Частотные характеристики прямоугольного окна, окон Хемминга и Блэкмана требуют компромисс между увеличением ширины основного лепестка, амплитудой первого бокового лепестка и спадом уровня боковых лепестков для популярных функций окна.

Лабораторное задание

Требуется сформировать аналоговый сигнал в программе **dsPICWorks.** Частота выходного сигнала задается в пределах от 400 Гц до 4 кГц. Сформировать таблицы значений уровней квантования восстанавливаемого сигнала результат записать в буфер RAM сигнального контроллера. Размер буфера должен иметь 256 значений. Загрузить программу в сигнальный контроллер и с помощью отладчика проверить ее выполнение. Проанализировать и убедиться в правильности БПФ сигнала путем импортирования полученных данных в буфер RAM сигнального контроллера, например в **MS Excel**.

Согласно индивидуальному заданию необходимо отредактировать исходный код программы для реализации алгоритма БПФ, который приведен ниже.

Файл **main.c**

```
#include <p33Fxxxx.h>
#include <dsp.h>
#include "fft.h"

/* объявление внешнего массива с отсчетами входного сигнала */
extern fractcomplex sigCmpx[FFT_BLOCK_LENGTH]
__attribute__ ((section (".ydata, data, ymemory"),
aligned (FFT_BLOCK_LENGTH * 2 *2)));

/* объявление внешнего массива с поворачивающими множителями */
extern const fractcomplex twiddleFactors[FFT_BLOCK_LENGTH/2]
__attribute__ ((space(auto_psv), aligned (FFT_BLOCK_LENGTH*2)));

/* объявление переменных */
int peakFrequencyBin = 0;
unsigned long peakFrequency = 0;

/* объявление массива с данными спектра */
fractional FFT_Waveform [FFT_BLOCK_LENGTH];

/* выполнение основной программы */
int main(void) {
int i = 0, break_p;
fractional *p_real = &sigCmpx[0].real;          // объявление указателей
fractcomplex *p_cmpx = &sigCmpx[0];             // ограничение входных данных в
                                                // диапазоне -0.5, +0.5
for ( i = 0; i < FFT_BLOCK_LENGTH; i++ )
{
*p_real = *p_real >> 1 ;                        // сдвигаем данные на один бит
*p_real++;
```

```
}
/* устанавливаем указатель на действительную и комплексную составляющую массива */
p_real = &sigCmpx[(FFT_BLOCK_LENGTH/2)-1].real ;
p_cmpx = &sigCmpx[FFT_BLOCK_LENGTH-1] ;

/* распределение действительной и обнуление мнимой части*/
for ( i = FFT_BLOCK_LENGTH; i > 0; i-- )
{
(*p_cmpx).real = (*p_real--);
(*p_cmpx--).imag = 0x0000;
}

/* вычисление БПФ */
FFTComplexIP (LOG2_BLOCK_LENGTH, &sigCmpx[0], (fractcomplex *)
__builtin_psvoffset(&twiddleFactors[0]), (int) __builtin_psvpage (&twiddleFactors[0]));

/* сохранение полученных данных с битреверсной адресацией */
BitReverseComplex (LOG2_BLOCK_LENGTH, &sigCmpx[0]);

/*вычисление амплитудного спектра и сохранение значений в массив
FFT_Waveform */
SquareMagnitudeCplx(FFT_BLOCK_LENGTH, &sigCmpx[0], FFT_Waveform);
/* поиск максимального значения амплитудного спектра*/
VectorMax(FFT_BLOCK_LENGTH/2, FFT_Waveform, &peakFrequencyBin);
/* вычисление частоты найденного максимального значения */
peakFrequency = peakFrequencyBin*(SAMPLING_RATE/FFT_BLOCK_LENGTH);

while (1);
}
```

Файл **fft.h**

```
/* определение основных параметров для вычисления БПФ*/
#define FFT_BLOCK_LENGTH 256

/* число этапов преобразования log2 256=8 */
#define LOG2_BLOCK_LENGTH 8
#define SAMPLING_RATE 4000                    // частота выборки сигнала
```

Использовалась функция *FFTComplexIP*, которая вычисляет ДПФ и сохраняет результат в исходный вектор(*In Place*).

Требования к отчету

1. Формулировка индивидуального задания, полученного у преподавателя.

2. Описание решения задачи с привлечением аналитических формул, временных диаграмм, схем функциональной и принципиальной, а также схем алгоритмов.

3. Текст отлаженной программы с комментариями.

4. Выводы по работе.

Контрольные вопросы и задания

1. Поясните принципы реализации алгоритмов спектрального анализа сигналов на сигнальных контроллерах.

2. Поясните реализацию алгоритма 8-точечного БПФ с прореживанием по времени и частоте.

3. Какие требования предъявляются к аппаратной реализации алгоритмов БПФ?

4. Обоснуйте применение оконных функций.

ПРИЛОЖЕНИЕ 1

Описание работы платы расширения для обработки аудиосигналов AUDIO PICtail ™ PLUS

Входной аудиосигнал с линейного входа или электретного микрофона выбирается перемычкой *J*8 (рис. П.1). Выбранный сигнал усиливается неинвертирующим усилителем переменного тока *U*4 (MCP6024) и подается через фильтр нижних частот (ФНЧ) (рис. П.2), частота среза которого равна 3300 Гц и далее на модуль аналого-цифрового преобразователя (АЦП) контроллера dsPIC33F. Выбор канала АЦП (AN0 или AN3) осуществляется перемычкой *J*5. Если к усилителю подключается электретный микрофон, то перемычкой *J*8 (3-2) он подключается к цепи питания *R*25 *C*27.

Регулировка усиления *U*4 изменяется резистором *R*29 (Mic ADJ) от 3 до 23 дБ.

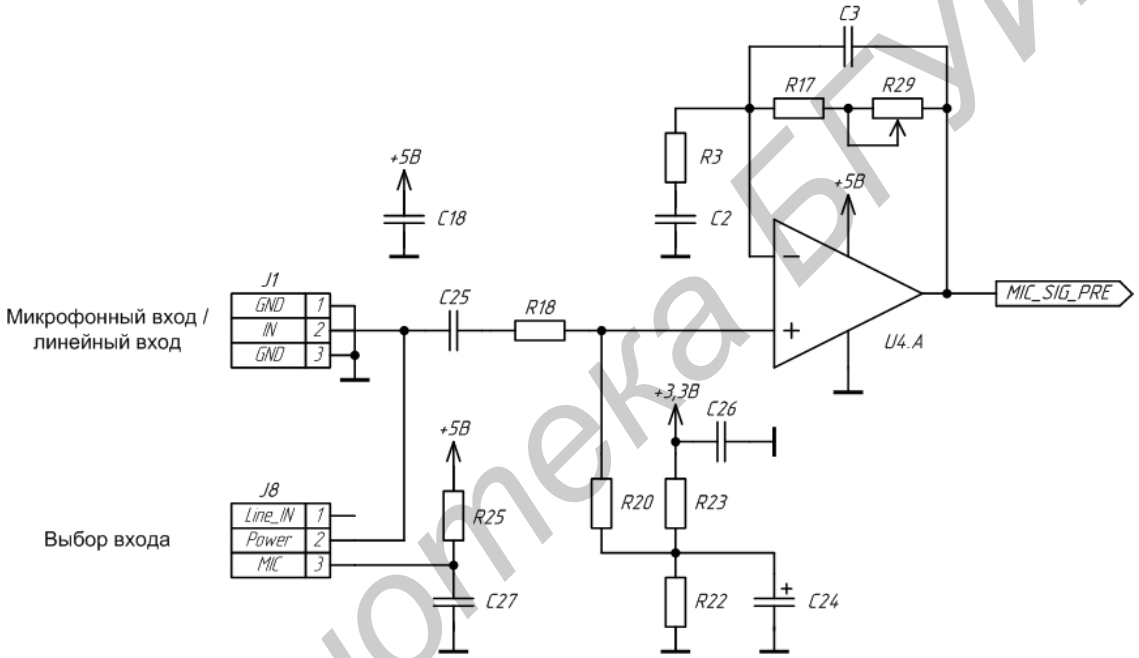


Рис. П.1. Схема входного усилителя

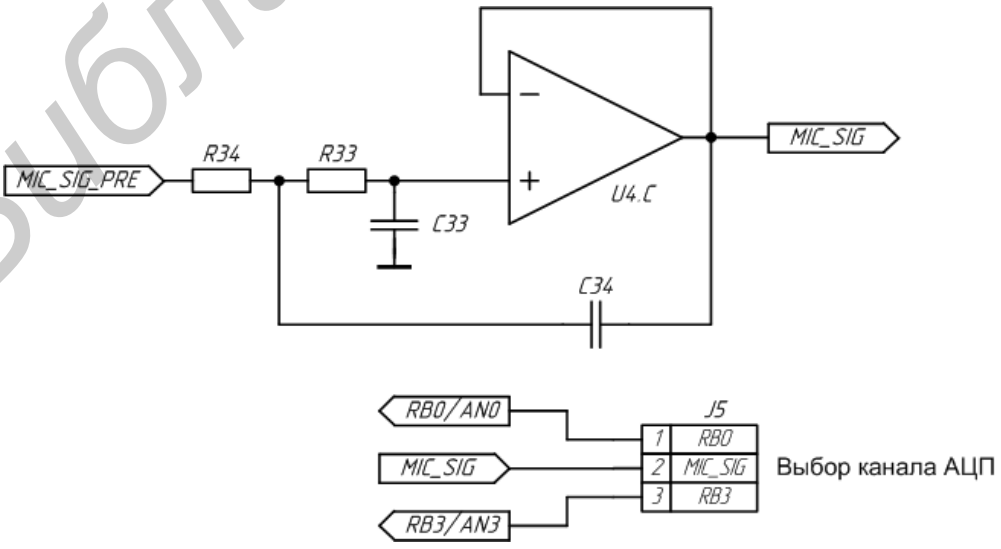


Рис. П.2. Схема ФНЧ (антиалайзинговый фильтр)

Воспроизведение речевого сигнала может осуществляться за счет широтно-импульсного модулированного (ШИМ) сигнала с вывода устройства модуля сравнения (ОС) dsPIC33F. Фильтр нижних частот преобразует ШИМ-сигнал в аналоговый, как показано на рис. П.3. Фильтр нижних частот, схема электрическая которого показана на рис. П.4, представляет собой интегратор, амплитуда выходного сигнала которого зависит от длительности импульса ШИМ. Частота ШИМ должна превышать, как правило, в пять раз максимальную частоту воспроизводимого аудиосигнала.

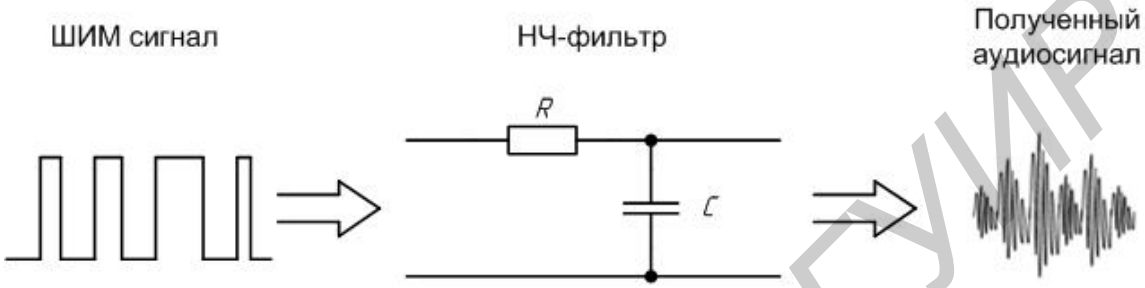


Рис. П.3. Схематичное изображение преобразования сигнала ШИМ в аудиосигнал

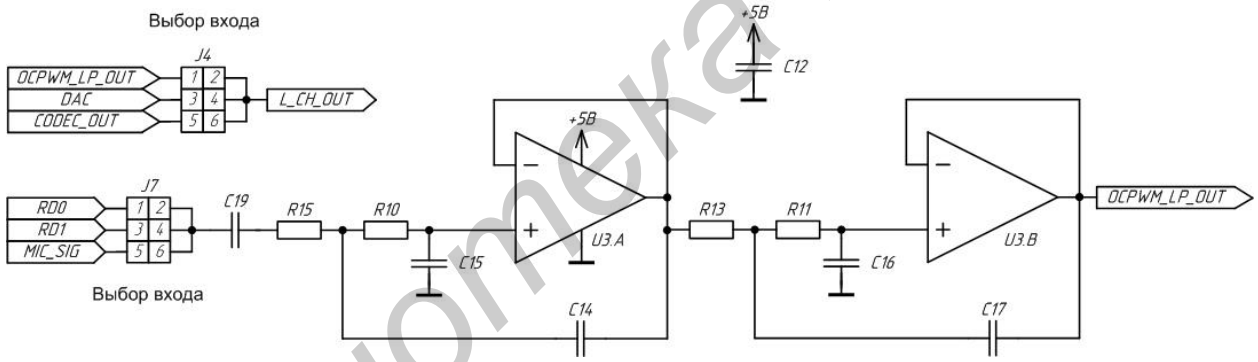


Рис. П.4. Схема ФНЧ

Сигнал с ФНЧ подается на усилитель линейного вывода или наушников (рис. П.5) через выходную перемычку *J*4. Выход оконечного усилителя подключается к внешнему аудиозаписывающему устройству или на внешний усилитель мощности звуковой частоты. Усилитель работает как инвертирующий с усилением по переменному току и с корректируемым коэффициентом усиления от –1 до +10 дБ.

Усилитель подключается к наушникам и может выдавать мощность до 75 мВт на нагрузку 32 Ом. Усилитель использует цифровой регулятор громкости, который управляется кнопкой *CLK S*1 и кнопкой управления *S*2 (рис. П.6).

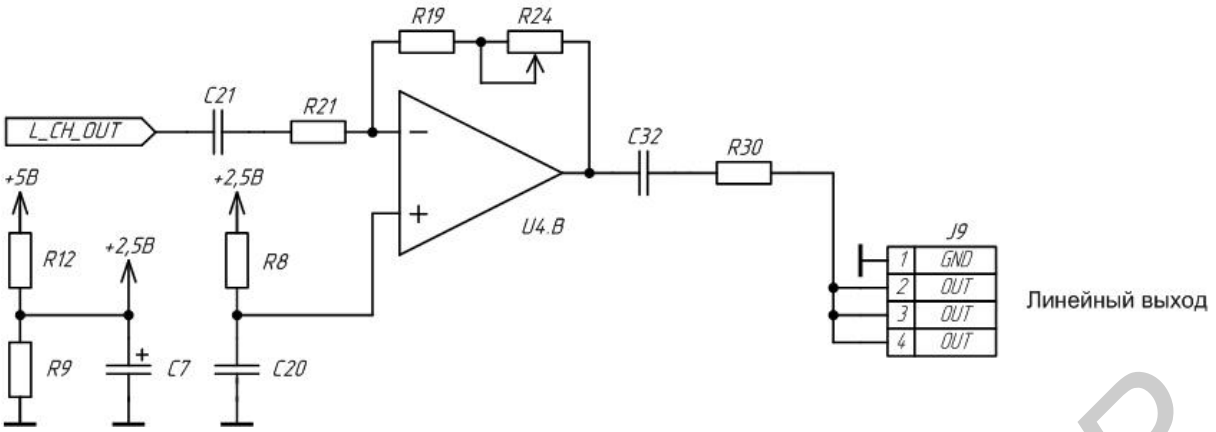


Рис. П.5. Схема выходного усилителя

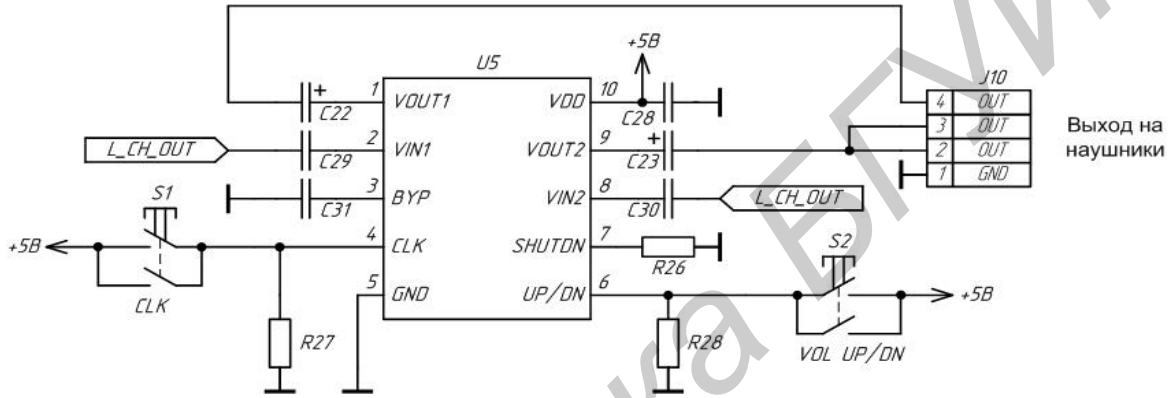


Рис. П.6. Схема усилителя для подключения наушников

На плате *Audio PICtail Plus Daughter Board* присутствует дифференциальный усилитель (рис. П.7), который используется для преобразования симметричного аудиосигнала, генерируемого встроенным модулем ЦАП сигнального контроллера dsPIC, в асимметричный. Полученный сигнал подается на оконечный усилитель.

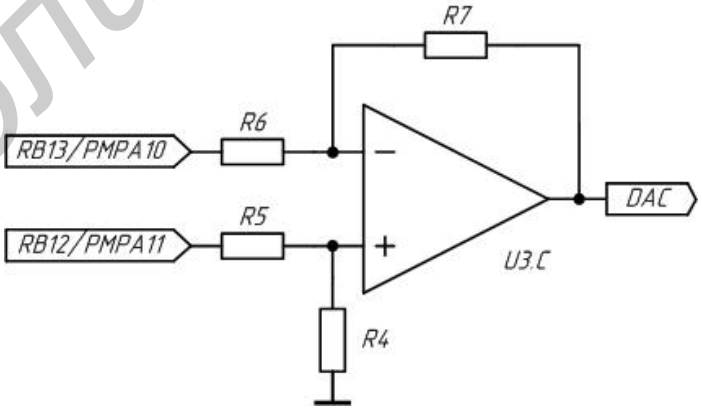


Рис. П.7. Схема усилителя на выходе ЦАП

Также для работы с аудиосигналами можно использовать дополнительный кодек WM8510G (рис. П.8), на вход которого подается усиленный сигнал с

линейного или микрофонного усилителя. Сигнал с выхода кодека подается на оконечный усилитель линейного выхода или наушников (переключается *J*4). Управление кодеком (частота дискретизации, регулировка громкости, регулирование уровня, настройка фильтров и т. д.) выполняется через интерфейс I^2C, а аудиоданные между кодеком и сигнальным контроллером передаются по интерфейсу DCI.

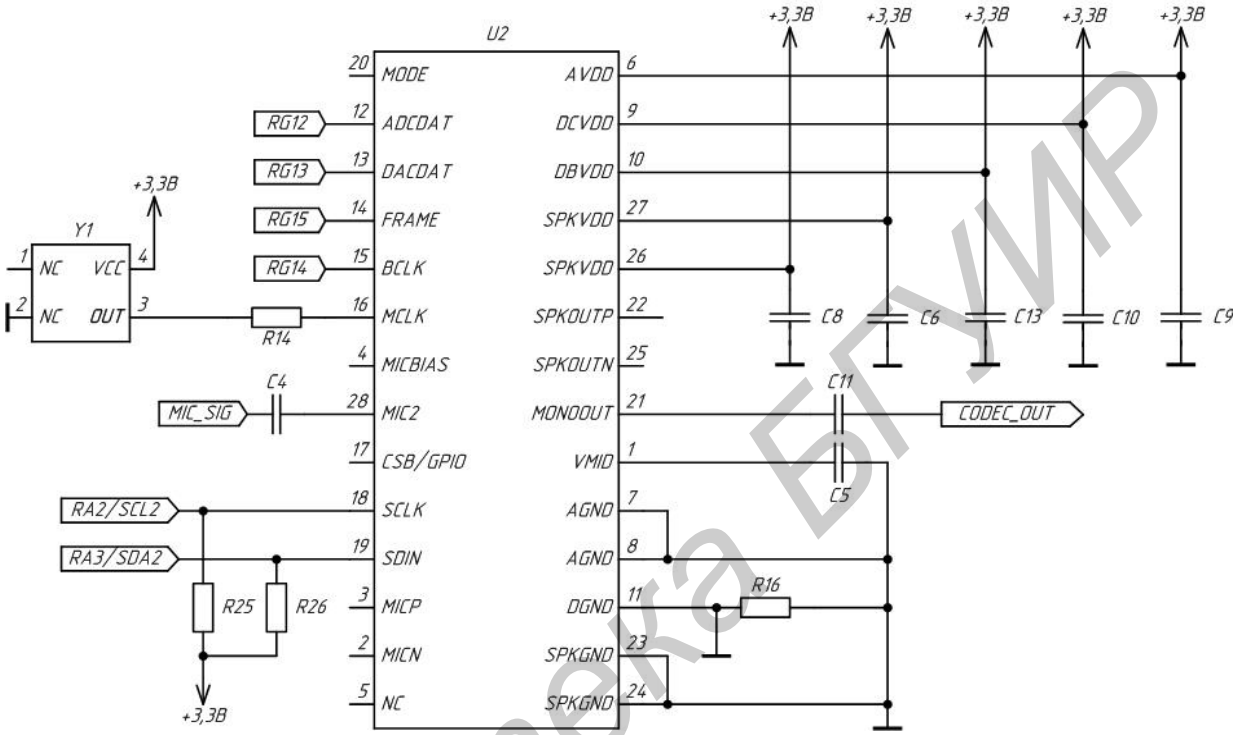


Рис. П.8. Схема кодека

Audio PICtail Plus Daughter Board содержит 4-мегабитную последовательную Flash-память (рис. П.9), которая может использоваться для хранения аудиоданных. Связь dsPIC33F с памятью осуществляется по интерфейсу SPI.

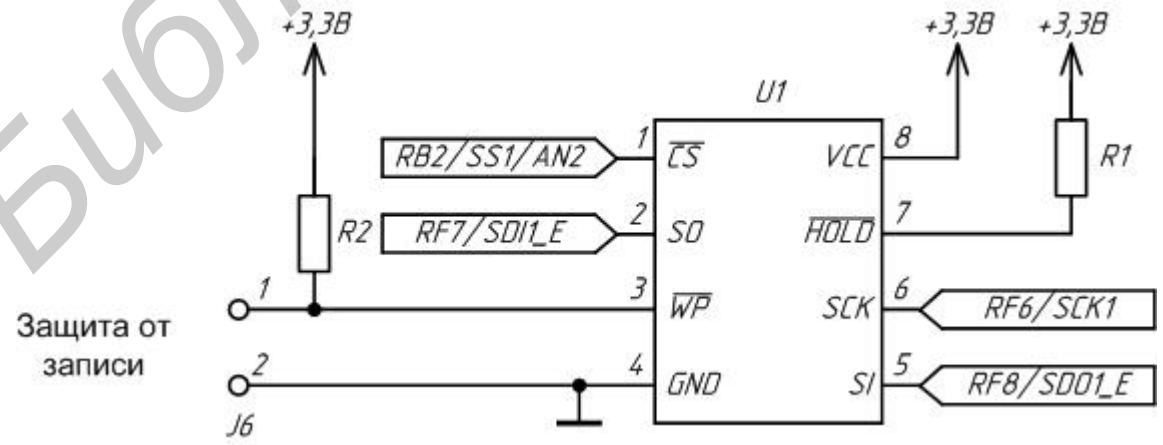


Рис. П.9. Схема подключения Flash-памяти

ПРИЛОЖЕНИЕ 2

Описание регистров управления сигнального микроконтроллера dsPIC33F

Биты регистра управления и состояния AD1CON1

Бит 15 **ADON**	**Бит включения модуля АЦП** 1 – модуль АЦП включен 0 – модуль АЦП отключен
Бит 12 **ADDMABM**	**Бит режима построения буфера DMA** 1 – буфер DMA записывается в порядке преобразования 0 – буфер DMA записывается в режиме накопления
Бит 10 **AD12B**	**Бит выбора 10-битного или 12-битного АЦП** 1 – 12 бит, 1-канальный АЦП 0 – 10 бит, 4-канальный АЦП
Биты 9-8 **FORM1–FORM0**	**Бит выбора формата выходных данных** **для 10-битных операций:** 11 = знаковое дробное 10 = дробное 01 = знаковое целое 00 = целое **для 12-битных операций:** 11 = знаковое дробное 10 = дробное 01 = знаковое целое 00 = целое
Биты 7-5 **SSRC2 – SSRC0**	**Бит выбора источника запуска АЦП** 111 = внутренний счетчик конца выборки и начала преобразования (автопреобразование) 100 = таймер (Timer 5 для АЦП1, Timer 3 для АЦП2) контролирует окончание выборки и запускает преобразование 010 = таймер (Timer 3 для АЦП1, Timer 5 для АЦП2) контролирует окончание выборки и запускает преобразование 001 = активный переход на входе INT0 приводит к завершению выборки и началу преобразования 000 = очистка бита SAMP приводит к завершению выборки и началу преобразования
Бит 3 **SIMSAM**	**Бит одновременной выборки** (только когда CHPS <1:0> = 01 или 1x) Когда AD12B = 1, SIMSAM устанавливается 0, не используется и читается как «0» 1 = Одновременная выборка CH0, CH1, CH2, CH3 (при

	CHPS <1:0> = 1x); или одновременная выборка CH0 и CH1 (при CHPS <1:0> = 01) 0 = последовательная выборка
Бит 2 **ASAM**	**Бит АЦП автозапуск выборки** 1 = выборка начинается сразу же после преобразования SAMP бит автоматически устанавливается 0 = выборка начинается, когда SAMP бит установлен
Бит 1 **SAMP**	**Бит разрешения выборки АЦП** 1 = начинает выборки сигнала 0 = режим хранения сигнала
Бит 0 **DONE**	**Бит статуса АЦП** 1 = преобразование модуля АЦП завершено 0 = преобразование модуля АЦП не запущено или в процессе аппаратно сбрасывается по завершении преобразования. Программно может быть записано «0». Очистка бита не влияет на процесс преобразования. Автоматическая очистка происходит в начале нового преобразования

Биты регистра управления и состояния ADxCON2

Биты 15-13 **VCFG<2:0>**	**Биты конфигурации источника опорного напряжения** **VCFG<2:0>** / **VREFH** / **VREFL** 000 / AVDD / AVSS 001 / Внешний VREF+ / AVSS 010 / AVDD / Внешний VREF- 011 / Внешний VREF+ / Внешний VREF- 1xx / AVDD / AVSS
Бит 10 **CSCNA**	**Бит выбора входа сканирования** 1 = вход сканирования для CH0 0 = не сканировать входы
Биты 9-8 **CHPS<1:0>**	**Биты выбора канала** Когда AD12B = 1, CHPS <1:0> не используются, читаются как «0» 1x = преобразует CH0, CH1, CH2 и CH3 01 = преобразует CH0 и CH1 00 = преобразует CH0

Биты регистра управления и состояния ADxCON3

Бит 15 **ADRC**	**Бит выбора источника тактовых сигналов АЦП** 1 = внутренний RC-генератор модуля АЦП 0 = системная тактовая частота
Биты 12-8 **SAMC <4:00>**	**Биты выбора делителя автовыборки** 11 111 = 31 TAD …… 00 001 = 1 TAD 00000 = 0 TAD
Биты 7-0 **ADCS <7:00>**	**Биты выбора тактовых сигналов преобразования АЦП** 00111111 = TCY • (**ADCS** <7:00> + 1) = 64 • TCY = TAD 00000010 = TCY • (**ADCS** <7:00> + 1) = 3 • TCY = TAD 00000001 = TCY • (**ADCS** <7:00> + 1) = 2 • TCY = TAD 00000000 = TCY • (**ADCS** <7:00> + 1) = 1 • TCY = TAD

Биты регистра управления и состояния ADxCON4

Биты 2-0 **DMABL<2:0>**	**Биты расположения буфера DMA по аналоговым входам** 111 = выделяет 128 слов из буфера для каждого аналогового входа 110 = выделяет 64 слова из буфера для каждого аналогового входа 101 = выделяет 32 слова буфера для каждого аналогового входа 100 = выделяет 16 слов буфера для каждого аналогового входа 011 = выделяет 8 слов из буфера для каждого аналогового входа 010 = выделяет 4 слова из буфера для каждого аналогового входа 001 = выделяет 2 слова из буфера для каждого аналогового входа 000 = выделяет 1 слово буфера для каждого аналогового входа

Биты регистра выбора входного канала ADxCHS123

Биты 10-9 **CH123NB <1:0>**	**Биты выбора отрицательного входа канала 1, 2, 3 для выборок B** Когда AD12B = 1, CHxNB не используется, читается как «0» 11 = CH1 отрицательный вход AN9, CH2 отрицательный вход AN10, CH3 отрицательный вход AN11 10 = CH1 отрицательный вход AN6, CH2 отрицательный вход AN7, CH3 отрицательный вход AN8 0x = CH1, CH2, CH3 отрицательный вход VREFL
Бит 8 **CH123SB**	**Биты выбора положительного входа канала 1, 2, 3 для выборок B** Когда AD12B = 1, ChxSA не используется, читается как «0» 1 = CH1 положительный вход AN3, CH2 положительный вход AN4, CH3 положительный вход AN5 0 = CH1 положительный вход AN0, CH2 положительный вход AN1, CH3 положительный вход AN2
Биты 2-1 **CH123NA <1:0>:**	**Биты выбора отрицательного входа канала 1, 2, 3 для выборок A** когда AD12B = 1, ChxNA не используется, читается как «0» 11 = CH1 отрицательный вход AN9, CH2 отрицательный вход AN10, CH3 отрицательный вход AN11 10 = CH1 отрицательный вход AN6, CH2 отрицательный вход AN7, CH3 отрицательный вход AN8 0x = CH1, CH2, CH3 отрицательный вход VREFL
Бит 0 **CH123SA**	**Биты выбора положительного входа канала 1, 2, 3 для выборок A** Когда AD12B = 1, ChxSA не используется, читается как «0» 1 = CH1 положительный вход AN3, CH2 положительный вход AN4, CH3 положительный вход AN5 0 = CH1 положительный вход AN0, CH2 положительный вход AN1, CH3 положительный вход AN2

Биты регистра выбора входного канала 0 ADxCHS0

Бит 15 **CH0NB**	**Бит выбора отрицательного входа канала 0 для выборок B** То же значение, что и бит 7
Биты 12-18 **CH0SB <4:00>**	**Биты выбора положительного входа канала 0 для выборок B** То же значение, что и биты <4:0>

Бит 7 **CH0NA**	**Бит выбора отрицательного входа канала 0 для выборок A** 1 = 0, отрицательный вход AN1 0 = 0, отрицательный вход VREFL не используется: читается как «0»
Биты 6-5 **CH0SA <4:00>**	**Биты выбора положительного входа канала 0 для выборок A** 11111 = канала 0 положительный вход AN31 11110 = канала 0 положительный вход AN30 … 00010 = канала 0 положительный вход AN2 00001 = канала 0 положительный вход AN1 00000 = канала 0 положительный вход AN0

Биты регистра конфигурации входов АЦП AD1PCFGH

Биты 15-0 **PCFG <31:16>**	**Биты конфигурации управления порта АЦП** 1 = вывод порта в цифровом режиме, порт разрешен для чтения, АЦП мультиплексора подключен к AVSS 0 = вывод порта в аналоговом режиме, порт запрещен для чтения

Биты регистра конфигурации входов АЦП AD1PCFGL

Биты 15-0 **PCFG <15:0>**	**Биты конфигурации управления порта АЦП** 1 = вывод порта в цифровом режиме, порт разрешен для чтения, АЦП мультиплексора подключен к AVSS 0 = вывод порта в аналоговом режиме, порт запрещен для чтения

Биты регистра настройки сканирования входных каналовAD1CSSH

Биты 15-0 **CSS <31:16>**	**Биты выбора сканирования каналов** 1 = выбор для сканирования входа ANX 0 = пропуск сканирования входа ANX

Биты регистра настройки сканирования входных каналовAD1CSSL

Биты 15-0 **CSS <15:0>**	**Биты выбора сканирования каналов** 1 = выбор для сканирования входа ANX 0 = пропустить сканирование входа ANX

Биты управления режимами работы АЦП

SSRC<2:0>	**Режимы работы АЦП**
000	Программный запуск (управление битом SAMP)
001	Запуск по внешнему источнику прерывания (INT0)
010	Запуск прерывания по таймеру (5,3)
011	Запуск управления электродвигателями ШИМ
100	Запуск прерывания по таймеру (3,5)
111	Автоматический запуск

ASAM	**SSRC<2:0>**	**Режимы работы АЦП**
0	000	Программный запуск выборки и программный запуск преобразования
0	111	Программный запуск выборки и автоматическое преобразование
0	001 010 011 100	Программный запуск выборки и преобразование по событию от периферии
1	000	Автоматическая выборка и программный запуск преобразования
1	111	Автоматическая выборка и автоматическое преобразование
1	001 010 011 100	Автоматическая выборка и преобразование по событию от периферии

ЛИТЕРАТУРА

1. Шабров, О. В. Среда проектирования WEbPAC ISE : метод. пособие к лаб. работам по курсу «Проектирование устройств на ПЛИС» для студ. спец. 1-39 01 03 «Радиоинформатика» дневн. формы обуч. / О. В. Шабров, А. И. Бурак. – Минск : БГУИР, 2007. – 75 с.

2. Максфилд, К. Проектирование на ПЛИС. Курс молодого бойца / К. Максфилд. – М. : Издательский дом «Додэка-XXI», 2007. – 408 с.

3. Зотов, В. Ю. Проектирование цифровых устройств на основе ПЛИС фирмы XILINX в САПР WebPACK ISE / В. Ю. Зотов. – М. : Горячая линия – Телеком, 2003. – 624 с.

4. Сергиенко, А. М. VHDL для проектирования вычислительных устройств / А. М. Сергиенко. – Киев : ЧП «Корнейчук», ООО «ТИД «ДС», 2003. – 208 с.

5. Зотов, В. Ю. Проектирование встраиваемых микропроцессорных систем на основе ПЛИС фирмы XILINX / В. Ю. Зотов. – М. : Горячая линия – Телеком, 2006. – 520 с.

6. Бибило, П. Н. Основы языка VHDL / П. Н. Бибило. – 3-е изд., доп. – М. : ЛКИ, 2007. – 328 с.

7. Spartan-3E FPGA Family Data Sheet. DS312, Xilinx [Электронный ресурс]. – 2012. – Режим доступа : http://www.xilinx.com/. – Дата доступа : 20.05.2013.

8. dsPIC33F. – Data Sheet. Microchip technology [Электронный ресурс]. – 2013. – Режим доступа: http:// www.microchip.com.

9. Казека, А. А. Разработка программ для микроконтроллеров фирмы MICROCHIP в интегрированной среде MPLAB IDE : метод. указание к лаб. работам по курсу «Микропроцессорные устройства» для студ. радиотехнич. спец. всех форм обуч. / А. А. Казека, А. В. Мартинович, И. Г. Давыдов. – Минск : БГУИР, 2008. – 31 с.

10. Кестер, У. Проектирование систем цифровой и смешанной обработки сигналов / У. Кестер. – М. : Техносфера, 2010.– 326 с.

11. Шпак, Ю. А. Программирование на языке С для AVR и PIC микроконтроллеров / Ю. А. Шпак. – 2-е изд. – Киев : МК-Пресс, 2011. – 544 с.

Св. план 2013, поз. 39

Учебное издание

Казека Александр Анатольевич
Каленкович Евгений Николаевич
Левкович Василий Николаевич
Давыдов Игорь Геннадьевич

СИГНАЛЬНЫЕ ПРОЦЕССОРЫ В УСТРОЙСТВАХ ЦИФРОВОЙ РАДИОСВЯЗИ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редакторы *И. В. Ничипор, М. А. Зайцева*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *А. А. Лущикова*

Подписано в печать 09.03.2015. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 6,86. Уч.-изд. л. 6,0. Тираж 150 экз. Заказ 275.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
ЛП №02330/264 от 14.04.2014.
220013, Минск, П. Бровки, 6