А. Б. СТЕПАНОВ

ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ В РАДИОТЕХНИЧЕСКИХ СИСТЕМАХ

УЧЕБНОЕ ПОСОБИЕ

САНКТ-ПЕТЕРБУРГ
2021

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,
СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное
бюджетное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ
им. проф. М. А. БОНЧ-БРУЕВИЧА»

А. Б. Степанов

ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ В РАДИОТЕХНИЧЕСКИХ СИСТЕМАХ

УЧЕБНОЕ ПОСОБИЕ

СПб ГУТ)))

САНКТ-ПЕТЕРБУРГ
2021

УДК 621.391(075.8)
ББК 32.811.3я73
С 79

Рецензенты:
доктор технических наук,
заведующий кафедрой теоретических основ электротехники
Санкт-Петербургского государственного
электротехнического университета «ЛЭТИ»
Е. Б. Соловьева,
доктор технических наук,
профессор кафедры радиосистем и обработки сигналов СПбГУТ
С. В. Томашевич

Автор выражает благодарность за оказанную помощь
при написании настоящего учебного пособия Е. В. Нечаеву

Утверждено редакционно-издательским советом СПбГУТ
в качестве учебного пособия

Степанов, А. Б.

С 79 Цифровая обработка сигналов в радиотехнических системах : учебное пособие / А. Б. Степанов ; СПбГУТ. – Санкт-Петербург, 2021. – 42 с.

Написано в соответствии с рабочей программой дисциплины «Цифровая обработка сигналов в радиотехнических системах». Рассматриваются основные этапы проектирования систем цифровой обработки сигналов с их реализацией на цифровых сигнальных процессорах.

Предназначено для студентов 1-го курса, обучающихся по программам магистратуры направления подготовки 11.04.01 «Радиотехника», а также аспирантов и специалистов в области радиотехники.

УДК 621.391(075.8)
ББК 32.811.3я73

СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ

АФНЧ	– аналоговый фильтр нижних частот.
АЦП	– аналого-цифровой преобразователь.
БИХ-фильтр	– фильтр с бесконечной импульсной характеристикой.
БПФ	– быстрое преобразование Фурье.
ИНС	– искусственная нейронная сеть.
ИАС	– источник аналогового сигнала.
ИСР	– интегрированная среда разработки.
КИХ-фильтр	– фильтр с конечной импульсной характеристикой.
ПЛИС	– программируемая логическая интегральная схема.
ПТ	– плавающая точка.
СФ	– сглаживающий фильтр.
ФВЧ	– фильтр верхних частот.
ФНЧ	– фильтр нижних частот.
ФТ	– фиксированная точка.
ЦАП	– цифро-аналоговый преобразователь.
ЦОС	– цифровая обработка сигналов.
ЦПОС	– цифровой процессор обработки сигналов.
ЦПУ	– центральное процессорное устройство.
ЦСП	– цифровой сигнальный процессор.
ЦФ	– цифровой фильтр.
CCS	– Code Composer Studio (интегрированная среда разработки фирмы Texas Instruments).
DSP	– Digital Signal Processing (цифровой сигнальный процессор).

1. ЦИФРОВЫЕ СИГНАЛЬНЫЕ ПРОЦЕССОРЫ

1.1. Обобщенная схема цифровой обработки сигналов

На рис. 1.1 представлена обобщенная схема цифровой обработки сигналов (ЦОС) [1–5].

Рис. 1.1. Обобщенная схема цифровой обработки сигналов

На выходе источника аналогового сигнала (ИАС) (например, микрофона) формируется аналоговый сигнал $x(t)$.

Аналоговый сигнал – это сигнал, непрерывный по времени и состоянию (рис. 1.2) [6]. Аналоговый сигнал является непрерывной или кусочно-непрерывной функцией.

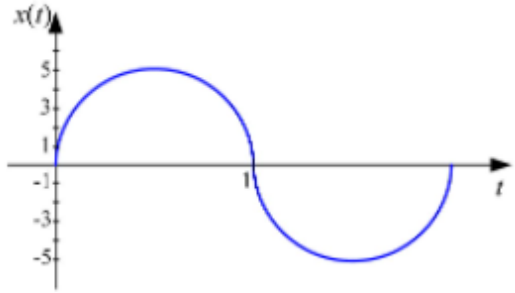


Рис. 1.2. График аналогового сигнала

Аналоговый сигнал поступает на вход аналогового фильтра нижних частот (АФНЧ), где его спектр ограничивается до частоты $f_в$, а далее поступает на вход аналого-цифрового преобразователя (АЦП), где подвергается дискретизации и квантованию. Формируется цифровой сигнал.

Дискретный сигнал $x(nT)$ – это сигнал, дискретный по времени и непрерывный по состоянию (рис. 1.3), представляется последовательностью чисел с бесконечной разрядностью. При моделировании дискретных сигналов в математических пакетах разрядность чисел, описывающих отсчеты, ограничена максимальной разрядностью универсального процессора общего назначения, используемого в персональном компьютере.

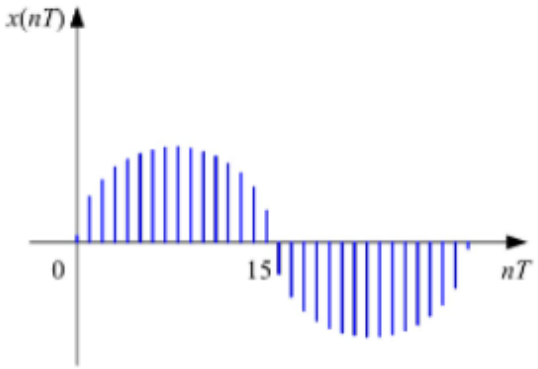


Рис. 1.3. График дискретного сигнала

Цифровой сигнал $\tilde{x}(nT)$ – это сигнал, дискретный по времени и квантованный по состоянию (рис. 1.4), описывается последовательностью чисел конечной разрядности.

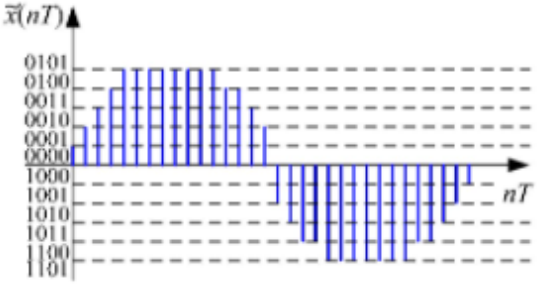


Рис. 1.4. График цифрового сигнала

Отличие в обозначении дискретного и цифрового сигналов указывает на наличие ошибки квантования.

На выходе АЦП формируется цифровой сигнал $\tilde{x}(nT)$, который поступает на вход системы ЦОС (вычислителя). В некоторых случаях входная аналоговая часть обобщенной схемы ЦОС может отсутствовать, а цифровой сигнал поступает на вход системы ЦОС (вычислителя) через порт ввода.

Под системой ЦОС понимается некоторая математическая модель, преобразующая входной цифровой сигнал $\tilde{x}(nT)$ в выходной цифровой сигнал $\tilde{y}(nT)$ в соответствии с заданным алгоритмом, или некоторое устройство, например цифровой сигнальный процессор (ЦСП). В последнем случае система ЦОС называется вычислителем.

На выходе системы ЦОС (вычислителя) формируется цифровой сигнал $\tilde{y}(nT)$ (рис. 1.5), который поступает на вход цифро-аналогового преобразователя (ЦАП).

Рис. 1.5. График цифрового сигнала
на выходе системы ЦОС (вычислителя)

На выходе ЦАП формируется аналоговый сигнал $\tilde{y}(nt)$ (рис. 1.6). Как правило, он имеет ступенчатый вид. Для его сглаживания используется сглаживающий фильтр (СФ). Сглаженный сигнал (рис. 1.7) передается потребителю (П).

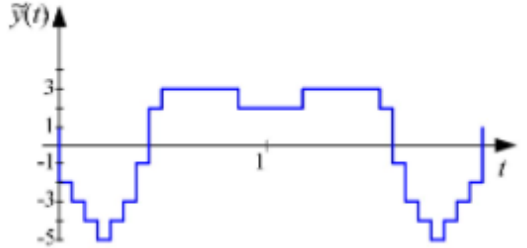


Рис. 1.6. График аналогового сигнала
на выходе ЦАП

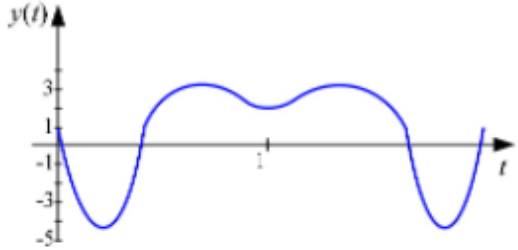


Рис. 1.7. График сигнала
на выходе сглаживающего фильтра

Выходная аналоговая часть схемы ЦОС может отсутствовать. В этом случае преобразованный цифровой сигнал выводится через порт вывода.

Как правило, на отладочных платах АЦП и ЦАП объединяются в виде одного устройства – кодека.

При реализации алгоритма цифровой обработки сигналов разработчик стремится к тому, чтобы разработанный им алгоритм выполнялся в реальном времени. В соответствии с этим, а также рядом других требований, предъявляемым к реализуемой системе, выбирается та или иная элементная база.

Алгоритмы цифровой обработки сигналов могут быть реализованы на следующей элементной базе:

1) цифровой сигнальный процессор (ЦСП, Digital Signal Processor, DSP) – это устройство, предназначенное специально для программной реализации алгоритмов цифровой обработки сигналов. В отечественной литературе закрепился термин – цифровой процессор обработки сигналов (ЦПОС);

2) программируемые логические интегральные схемы (ПЛИС) – это устройства, позволяющие выполнять аппаратную реализацию систем ЦОС;

3) универсальные процессоры общего назначения – эти устройства используются в персональных компьютерах и предназначены для решения самых различных задач;

4) системы на кристалле (System on a Chip, SoC) – это устройства, которые, как правило, имеют комбинированную архитектуру. Они могут совмещать в себе несколько ядер DSP, ядро DSP и ядро ARM, DSP и элементы логики;

5) графические процессоры – эти устройства получили широкое распространение, например, при реализации искусственных нейронных сетей (ИНС);

6) микроконтроллеры. Эти устройства предназначены в первую очередь для реализации систем управления. Однако производительность современных микроконтроллеров позволяет применять их и при реализации систем цифровой обработки сигналов;

7) другие устройства. Современный рынок элементной базы постоянно развивается, появляются новые устройства, пригодные для выполнения цифровой обработки сигналов.

Далее более подробно будут рассмотрены цифровые сигнальные процессоры.

1.2. Определение цифровых сигнальных процессоров

Цифровые сигнальные процессоры – это устройства, разработанные специально для реализации алгоритмов цифровой обработки сигналов. Архитектура ЦСП включает элементы, позволяющие выполнять алгоритмы ЦОС с максимальной скоростью.

На быстродействие ЦСП оказывают влияние два фактора:

- тактовая частота;
- архитектура.

По сравнению с процессорами общего назначения, тактовые частоты ЦСП, как правило, ниже. Современные ЦСП обладают тактовой частотой от 120 МГц до 1,4 ГГц.

Однако ЦСП имеют аппаратную реализацию простейших операций: умножение, сложение, задержка (сдвиг). Это является одним из основных достоинств ЦСП, позволяющих значительно сократить время выполнения алгоритма ЦОС.

Если для универсального процессора общего назначения i8086 для выполнения операции умножения требуется более 100 тактов [6], то для ЦСП – менее 1 такта.

Кроме базовых операций, некоторые ЦСП имеют аппаратную реализацию алгоритма быстрого преобразования Фурье (БПФ), видеокодеков и др.

Цифровые сигнальные процессоры применяются для реализации самых разнообразных устройств: от игрушек и сотовых телефонов до высокоточных измерительных приборов и систем управления космическими аппаратами.

Программируемые логические интегральные схемы целесообразно применять при изготовлении малотиражной продукции (порядка 10 шт.), для проверки работоспособности алгоритма после его моделирования, а также при необходимости распараллеливания вычислительных потоков. Для выпуска продукции большим тиражом (порядка 1000 шт.) целесообразно использовать ЦСП, что также позволяет снизить стоимость конечного продукта. Кроме того, ЦСП должны использоваться и при изготовлении особо ответственного оборудования, где требуется высокая надежность.

1.3. Архитектура ЦСП

В основе многих устройств, пригодных для реализации алгоритмов цифровой обработки сигналов, лежит одна из базовых архитектур [6]:

- архитектура фон Неймана;
- гарвардская архитектура;
- модифицированная гарвардская архитектура.

Цифровые сигнальные процессоры также относятся к подобным устройствам.

Рассмотрим более подробно каждую из упомянутых архитектур и отметим их достоинства и недостатки.

1. Архитектура фон Неймана (рис. 1.8).

Рис. 1.8. Архитектура фон Неймана

Достоинством данной архитектуры является ее простота и низкая стоимость. В качестве ее основного недостатка необходимо отметить низкое быстродействие. Это связано с тем, что память программ и память данных объединены в один блок, который соединен с блоком центрального процессорного устройства (ЦПУ) только одним набором шин: адреса (ША) и данных (ШД). Эта особенность архитектуры не позволяет одновременно передавать в ЦПУ данные и программы.

2. Гарвардская архитектура (рис. 1.9).

Рис. 1.9. Гарвардская архитектура

Особенностью гарвардской архитектуры является разделение памяти программ и памяти данных в разные блоки. В результате появляется дополнительный набор шин. Архитектура позволяет одновременно передавать в ЦПУ данные и программы. Все это значительно ускоряет процесс выполнения алгоритмов ЦОС. Однако данная архитектура имеет и существенный недостаток – сложность и, как следствие, высокую стоимость вычислителя. Как известно, увеличение стоимости элементной базы приводит к повышению стоимости конечного продукта.

Решить эту проблему позволила разработка модифицированной гарвардской архитектуры.

3. Модифицированная гарвардская архитектура совмещает в себе достоинства каждой из рассмотренных ранее. Добиться высокой скорости работы и снизить себестоимость позволило уменьшение числа внешних шин: данная архитектура использует полный набор шин для взаимодействия с ЦПУ и один набор для работы с внешними устройствами.

Значительная часть современных цифровых сигнальных процессоров используют в своей основе модифицированную гарвардскую архитектуру. Тем не менее архитектура фон Неймана до сих пор используется в некоторых устройствах, пригодных для выполнения цифровой обработки сигналов, например в микроконтроллерах фирмы Texas Instruments MSP430.

Говоря об архитектуре современных ЦСП, сложно дать описание некоторого обобщенного варианта, тем более изобразить ее в виде конкретной структуры. Поэтому приведем список блоков, которые, как правило, присутствуют в ЦСП, а в ряде случаев даже дублируются:

- арифметико-логическое устройство (АЛУ);
- устройство генерации адреса (УГА);
- умножитель;
- блоки памяти;
- таймеры и пр.

Архитектура современных ЦСП активно развивается, в связи с этим сложно сформулировать конкретное определение ЦСП. Выделим его основные особенности по сравнению с другими устройствами:

1) архитектура ЦСП предназначена специально для выполнения алгоритмов ЦОС;

2) аппаратная реализация простейших операций, таких как: умножение, сложение и задержка (сдвиг). Данным операциям соответствуют устройства: умножитель (рис. 1.10, а), сумматор (рис. 1.10, б), элемент задержки (сдвигатель) (рис. 1.10, в).

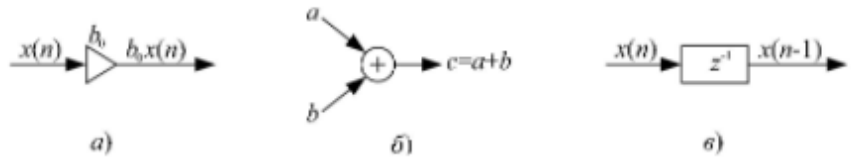


Рис. 1.10. Основные элементы ЦСП:
а) умножитель; б) сумматор; в) элемент задержки (сдвигатель)

Данный отличительный признак ЦСП можно продемонстрировать на примере структуры нерекурсивного цифрового фильтра (рис. 1.11), который состоит их этих 3 видов элементов, где ММ – масштабный множитель;

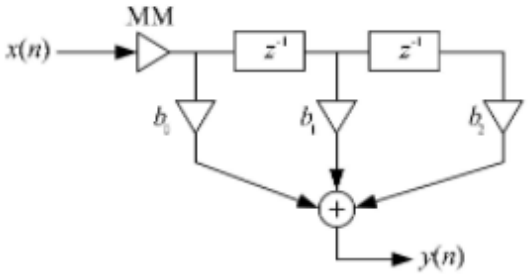


Рис. 1.11. Прямая структура
нерекурсивного цифрового фильтра

3) наличие блока MAC (multiplier–accumulator, операция умножения с накоплением в аккумуляторе). Ранее считалось, что наличие блока MAC является показателем того, что данное устройство – это ЦСП. Действительно, многие ЦСП обладают одним или несколькими блоками MAC. Например, ЦСП фирмы Texas Instruments TMS320C54xx и TMS320C55xx имеют один и два блока MAC соответственно. В тот же момент ЦСП TMS320C6xxx этой же фирмы его не имеет, но также считается ЦСП. Противоположным примером может считаться TMS320C2xxx – в прошлом относящийся к ЦСП, обладающим блоком MAC, а в настоящее время – это устройство, позиционирующееся фирмой Texas Instruments как микроконтроллер;

4) Ограниченный набор периферийных устройств. Традиционно считается, что если устройство обладает ограниченным набором периферии, то это ЦСП, а если периферийных устройств достаточно много, то это устройство более подходит для реализации систем управления, что характерно для микроконтроллеров. Отчасти это условие сохранилось. TMS320C2xxx при переходе из класса ЦСП в класс микроконтроллеров получил специфический набор периферийных устройств. Однако в силу того, что современный ЦСП должен решать сразу комплекс задач, эта граница весьма размыта. Кроме того, по этой же причине появляются устройства с комбинированной архитектурой – системы на кристалле.

1.4. Конвейерный принцип выполнения команд

Каждая команда в ЦСП выполняется по этапам. В простейшем случае таких этапов может быть 4 (рис. 1.12).

На первом этапе выполняется выборка команды. На втором – ее декодирование. На третьем этапе осуществляется подготовка операндов (переменных, с которыми работает команда). Четвертый этап – непосредственное выполнение команды.

Выборка команды	Декодирование команды	Подготовка операндов	Выполнение команды
1 этап	2 этап	3 этап	4 этап

Рис. 1.12. Этапы выполнения команды

Принцип поэтапного выполнения команды может быть продемонстрирован на примере сложения двух чисел: *a* и *b*.

На первом этапе выполняется выборка команды. Далее, на втором этапе, при декодировании команды ЦСП определяет, что ему необходимо выполнить операцию сложения. На 3 этапе производится подготовка переменных *a* и *b*. На последнем, 4 этапе, выполняется само сложение. Неконвейерный принцип выполнения команд означает, что при выполнении нескольких команд сначала выполняются все этапы первой команды, и только после этого процессор переходит к выполнению последующих команд: второй, третьей и т. д. (рис. 1.13).

1 команда				2 команда				3 команда			
1 этап	2 этап	3 этап	4 этап	1 этап	2 этап	3 этап	4 этап	1 этап	2 этап	3 этап	4 этап

Рис. 1.13. Неконвейерный принцип выполнения команд

Конвейерный принцип выполнения команд подразумевает, что в одно и то же время выполняются различные этапы следующих друг за другом команд (рис. 1.14).

Рассмотрим конвейерный принцип выполнения команд более подробно.

В то время, когда выполняется 1 этап 1 команды, все остальные команды не выполняются. Параллельно с выполнением 2 этапа 1 команды выполняется 1 этап 2 команды. При выполнении 3 этапа 1 команды выполняется 2 этап 2 команды и 1 этап 3 команды. И так далее. Таким образом, даже самый простой 4-этапный конвейер позволяет в несколько раз увеличить скорость выполнения программного кода. Четырехэтапный конвейер был внедрен уже в первом в мире цифровом сигнальном процессоре TMS32010, разработанным фирмой Texas Instruments в 1983 году. Современные ЦСП могут иметь конвейер с числом этапов более десяти.

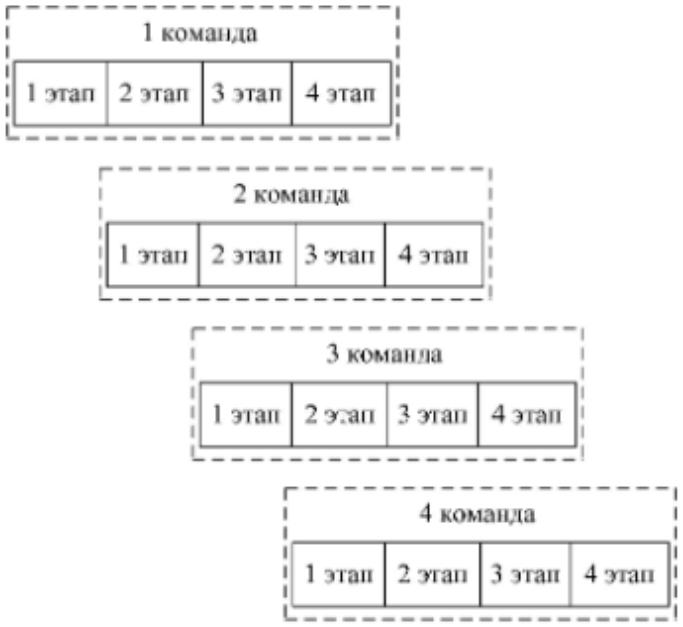


Рис. 1.14. Конвейерный принцип выполнения команд

При работе конвейера может возникнуть ситуация, когда при декодировании очередной команды ЦСП определяет, что необходимо прекратить выполнение текущего набора команд и перейти по команде условного перехода к другой части памяти, где хранится другой набор команд. Это может приводить к сбоям в работе ЦСП. Такая ситуация называется конфликтом конвейера и должна быть учтена разработчиком. В современных ЦСП существуют специальные механизмы, позволяющие избежать серьезных ошибок при возникновении конфликта конвейера.

2. РЕАЛИЗАЦИЯ АЛГОРИТМОВ ЦОС НА ЦСП

2.1. Языки программирования для ЦСП

Все языки программирования, применяемые при реализации систем ЦОС на ЦСП, можно разделить на 2 группы: языки низкого уровня (машино-ориентированные) и языки высокого уровня. К языкам низкого уровня относится язык ассемблера. К языкам высокого уровня – языки С и С++.

Необходимо различать язык ассемблера и языки ассемблера. В первом случае подразумевается совокупность команд, которые относятся к данному языку и могут быть сведены в единый справочник. Языками ассемблера называют ограниченные наборы команд, которые поддерживаются конкретными цифровыми сигнальными процессорами. Каждая фирма, выпуская на рынок новый ЦСП, на своем сайте или в сопроводительной документации приводит информацию о наборе команд, поддерживаемом данным ЦСП.

Достоинством языка ассемблера при реализации систем ЦОС на ЦСП является возможность работы с ЦСП на низком уровне. Программный код, написанный на языках ассемблера, может быть оптимальным.

Недостатком языка ассемблера является относительная сложность написания программного кода на нем. Обычно для этого требуется значительное время.

Языки С и С++ относятся к языкам высокого уровня. Написание программного кода с их применением, как правило, требует меньше времени, чем при использовании языка ассемблера. Однако программный код, написанный на С/С++, даже при мерах, направленных на его оптимизацию, не может считаться в полной мере оптимальным, что является их недостатком. Язык С++ является объектно-ориентированным языком. Языки С/С++ выбираются тогда, когда требуется реализовать систему ЦОС за минимальное время и когда при этом может быть выбрана заведомо более мощная (и как следствие более дорогая) элементная база.

Обратной ситуацией является случай, когда необходимо выпустить систему ЦОС, обладающую минимальной стоимостью, при этом сроки разработки конечного продукта и выпуск его на рынок не имеют значения. Типичным примером таких систем являются различного рода охранные и пожарные датчики. Одним из самых дорогих элементов в таких системах является ЦСП или микроконтроллер. На его стоимость оказывает влияние его быстродействие и объем внутренней памяти. Именно поэтому каждый разработчик таких систем стремится оптимизировать программный код для них по объему и быстродействию. Связано это и с тем, что программный код целесообразно размещать во внутренней памяти устройства, так как работа с ней осуществляется быстрее, чем с внешней.

Для ускорения работы программного кода, написанного на языке высокого уровня, могут использоваться фрагменты кода на языке ассемблера, так называемые «ассемблерные вставки». Данные вставки заменяют часть кода на языке высокого уровня, который выполняется медленно и задерживает работу всей системы ЦОС.

Также может быть использована технология реализации систем ЦОС, при которой программный код вначале пишется на языке высокого уровня, а потом полностью или частично переписывается на языке ассемблера. Такой подход позволяет быстро получить работающую систему, а далее выполнить оптимизацию программного кода и добиться повышения скорости работы системы и снизить требования к используемой элементной базе.

Кроме применения языков низкого и высокого уровней для реализации систем ЦОС могут применяться языки «сверхвысокого» уровня. На базе функций, написанных с их применением, может выполняться генерация программного кода на языках C/C++. К языкам «сверхвысокого» уровня относится язык MATLAB. О процедуре генерации С-кода на основе кода MATLAB и Simulink-модели будет рассказано далее.

2.2. Основные этапы проектирования систем ЦОС с их реализацией на ЦСП

Проектирование систем ЦОС с их реализацией на ЦСП включает в себя следующие этапы:

1) формирование общих требований к системе ЦОС;
2) разработка математического аппарата, на котором будет основан принцип работы предлагаемой системы ЦОС;
3) моделирование системы ЦОС в одном из математических пакетов, например MATLAB или LabVIEW;
4) разработка основного алгоритма работы системы ЦОС;
5) выбор ЦСП для реализации системы ЦОС;
6) теоретическая оценка требуемого ресурса;
7) уточнение модели ЦСП и отладочной платы;
8) разработка детализированного алгоритма работы системы ЦОС;
9) выбор языка программирования;
10) создание проекта в интегрированной среде разработки;
11) написание программного кода;
12) проверка проекта на наличие ошибок и их исправление;
13) получение бинарного исполняемого файла;
14) загрузка выходного файла в память ЦСП;
15) проверка работы системы ЦОС на тестовых сигналах;

16) проверка работы системы ЦОС на реальных сигналах (например, в составе комплекса).

При разработке программного кода может использоваться один из принципов его написания: «сверху вниз» или модульный.

Первый метод подходит для написания небольших программ. При его применении программный код пишется полностью от первой до последней команды.

В случае, если программный код ожидается объемным, целесообразно разделить его на части – модули. Такой принцип написания программного кода называется модульный. Модульный принцип удобно использовать для написания однотипных функций, а также при реализации проекта группой программистов (разработчиков). В этом случае каждый разработчик пишет свою часть программного кода (модуль). Такой подход позволяет значительно ускорить процесс реализации системы ЦОС.

Также интерес может представлять использование так называемой «файловой модели». В этом случае при использовании языка высокого уровня функция *main* содержит только минимальный объем команд, который позволяет: загрузить сигнал, вызвать необходимые функции для его обработки (например, фильтрации или вычисления быстрого преобразования Фурье) и записать обработанный сигнал во внешнюю память. Применение файловой модели позволяет значительно сократить время на разработку новых проектов при наличии уже написанных базовых функций. Кроме того, функция main становится удобной для чтения и быстрой корректировки.

Кроме написания функций также могут использоваться специальные библиотеки. Подключение таких библиотек к проекту позволяет значительно упростить и ускорить процесс реализации системы ЦОС.

2.3. Отладочные средства

Отладочные средства можно разделить на программные и аппаратные.

К программным средствам отладки относятся:

- симуляторы;
- компиляторы;
- отладчики;
- линкеры;
- интегрированная среда разработки (ИСР) и пр.

К современным аппаратным средствам отладки относятся:

- внутрисхемные эмуляторы;
- внутрикристальные эмуляторы;
- отладочные модули.

Симулятор – это программа, имитирующая работу ЦСП. Она разработана таким образом, чтобы учитывать архитектуру ЦСП, его особенности и даже временные задержки, возникающие в процессе его работы. Ранее симуляторы использовались для первоначальной отладки программного кода, перед загрузкой программы в память ЦСП.

Основные достоинства симулятора:

- доступность – он, как правило, распространяется свободно и может быть установлен на неограниченное число персональных компьютеров;
- симулятор сложно повредить (в отличие от реальной отладочной платы). В случае необходимости симулятор может быть оперативно переустановлен.

Недостатком симуляторов является скорость их работы. Несмотря на стремление разработчиков учесть все временные задержки, возникающие при его работе, тем не менее, они могут быть слишком большими. Это приводит к существенным временным потерям при отладке проектов. Поэтому в настоящее время симуляторы в основном используются для обучения, а отладка проектов выполняется сразу с применением отладочных модулей.

Интегрированная среда разработки – это совокупность всех инструментов, собранных в одной оболочке, предназначенных для разработки проектов ЦОС и для их реализации на ЦСП.

Интегрированная среда разработки включает все необходимое для создания проекта.

В качестве примера рассмотрим составляющие и основные этапы создания проекта в интегрированной среде разработки Code Composer Studio (CCS), используемой для ЦСП и микроконтроллеров фирмы Texas Instruments. Интегрированная среда разработки Code Composer Studio реализована на основе кросс-платформы Eclipse. Достоинствами данной платформы являются ее открытость, бесплатность, кроме того, она известна многим разработчикам. Последнее и послужило одной из причин применения данной платформы. Интегрированная среда разработки должна обладать интерфейсом, который известен для разработчиков и претерпевает незначительные изменения от версии к версии.

Рассмотрим основные составляющие ИСР Code Composer Studio версии 5 (рис. 2.1).

Интерфейс CCS включает следующие элементы:

- перспектива;
- проводник проекта;
- редактор файлов исходных кодов;
- консоль проекта;
- кнопки переключения режимов.

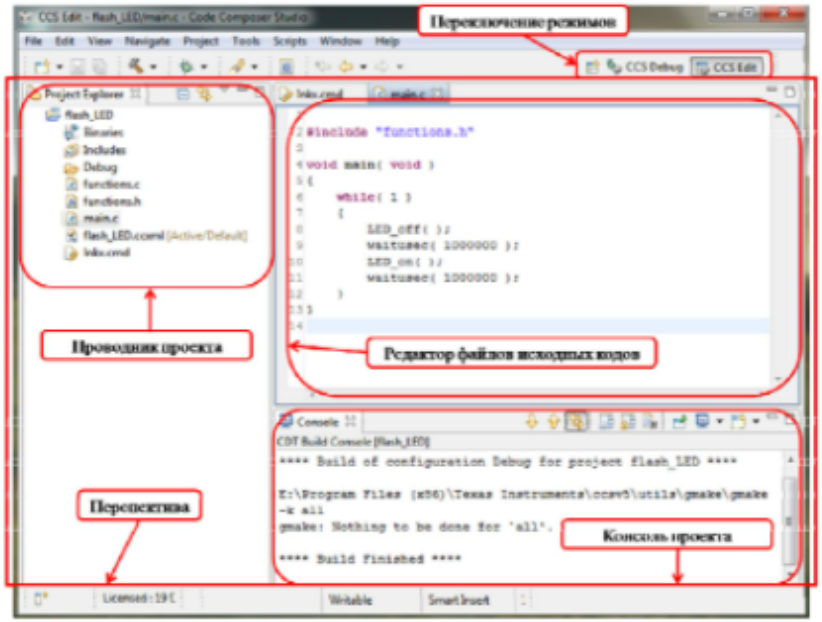


Рис. 2.1. Окно интегрированной среды разработки Code Composer Studio v.5

Реализация проекта в CCS включает следующие этапы:

1) выбор рабочей области;

2) создание проекта;

3) создание или добавление файлов с исходным кодом, заголовочных и командного файлов;

4) создание и настройка файла конфигурации;

5) компиляция проекта, получение бинарного исполняемого файла.

К аппаратным средствам отладки относятся эмуляторы: внутрисхемные и внутрикристальные.

Внутрисхемный эмулятор располагается на отладочной плате и может занимать значительную ее часть.

Внутрикристальный эмулятор находится непосредственно на ЦСП или на микроконтроллере (на самом кристалле).

Внутрисхемный и внутрикристальный эмуляторы работают совместно.

К аппаратным средствам отладки также относятся отладочные модули. Они делятся на 2 типа: стартовые наборы и профессиональные отладочные платы.

Стартовые наборы – это отладочные модули, предназначенные для первоначального знакомства с отладочными средствами данной линейки.

Стартовые наборы включают в себя цифровой сигнальный процессор или микроконтроллер – один из наиболее дешевых в линейке, недорогую отладочную плату, программное обеспечение на диске или свободное для скачивания, как правило, с ограниченной лицензией (по объему кода, функциональности или длительности использования), необходимое коммутационное оборудование, инструкцию, привлекательную упаковку. В Интернете для таких наборов можно найти много примеров программного кода. Цена таких комплектов невелика. Цель производителей – привлечь потенциальных разработчиков к своей продукции.

Наборы профессиональных отладочных плат включают в себя:

- цифровой сигнальный процессор, предназначенный для решения конкретной задачи (обработки аудио, видео и т. д.);
- отладочная плата, предназначенная для решения конкретной задачи и включающая соответствующий набор периферийных устройств;
- программное обеспечение на диске, которое может приобретаться отдельно;
- кабель. Как правило, в комплект поставки входит только основной набор коммутационного оборудования. В ряде случаев кабель поставляется без разъемов на одном из концов с облужением;
- часть оборудования для профессиональной отладки может не поставляться, а должна приобретаться отдельно (например, внешние эмуляторы);
- переходники и прочее оборудование для соединения с другими отладочными платами приобретаются отдельно;
- инструкция;
- упаковочная коробка.

3. СРЕДСТВА АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ ПРОГРАММНОГО КОДА ДЛЯ ЦСП

Помимо прямого написания программного кода для ЦСП с использованием машинно-ориентированного языка и языков высокого уровня, может также использоваться технология автоматической генерации программного кода на языках C/C++ средствами системы MATLAB.

MATLAB – это система, предназначенная для математического моделирования, включающая в себя 3 части:

- ядро MATLAB;
- подсистему блочного (динамического) моделирования Simulink;
- пакеты расширения (Toolbox).

Ядро MATLAB содержит набор встроенных команд для языка сверхвысокого уровня, позволяющих выполнять матричные вычисления. Также при необходимости помимо имеющихся встроенных функций пользователь может использовать свои, написанные им в соответствии с разработанным алгоритмом.

Подсистема блочного (динамического) моделирования Simulink позволяет работать с моделями систем (Simulink-моделями). Необходимая структура может быть собрана из набора существующих или созданных пользователем блоков.

Наборы инструментов (пакеты расширения) Toolbox позволяют выполнять моделирование систем в более узком (специализированном) направлении, например:

- проектирование фильтров (FDATool);
- проектирование искусственных нейронных сетей (NNTool);
- вейвлет-анализ сигналов (Wavelet Toolbox) и пр.

3.1. Технология автоматической генерации C/C++-кода

В MATLAB предусмотрена технология, позволяющая выполнять автоматическую генерацию программного кода на языке C/C++ для выбранного цифрового сигнального процессора.

Генерация программного кода на языке C/C++ может осуществляться на основе программного кода, написанного на языке MATLAB или на основе Simulink-модели. Рассмотрим оба подхода более подробно.

3.1.1. Генерация C/C++-кода на основе программного кода MATLAB

При работе с системой MATLAB можно выделить файлы с расширением *.mat и *.m. Файлы с расширением *.mat используются для хранения данных. Файлы с расширением *.m, как правило, содержат последовательности команд и делятся на два вида: скрипты (script-файл) и функции (function-файл). На основе function-файла может осуществляться генерация программного кода на языке C/C++. Для этого разработчику необходимо написать function-файл, реализующий алгоритм системы ЦОС, проверить его работоспособность, а далее произвести настройку специального инструмента – MATLAB Coder – и выполнить генерацию программного кода на языке C/C++ для целевого устройства.

Приведем процедуру генерации программного кода для ЦСП с применением MATLAB Coder. Она включает следующие этапы.

1. Разработка алгоритма работы системы ЦОС.
2. Написание m-файла.
3. Проверка работоспособности m-файла.
4. Настройка параметров MATLAB Coder.
5. Генерация C/C++-кода для целевого устройства.
6. Реализация проекта в интегрированной среде разработки.
7. Модернизация программного кода для работы с периферийными устройствами.
8. Отладка программного кода.
9. Получение выходного файла.
10. Загрузка выходного файла в память ЦСП.

Важно отметить, что при написании function-файла необходимо учитывать, что не все команды на языке MATLAB поддерживаются MATLAB Coder и могут быть использованы при генерации C/C++-кода.

Кроме того, необходимо понимать основные принципы работы ЦСП. С этой целью необходимо предусмотреть организацию ввода/вывода сигналов портами ЦСП.

К достоинствам использования m-файла при генерации C/C++-кода для ЦСП можно отнести:

1) выполнение автоматической генерации C/C++-кода на основе m-файла. При корректной настройке параметров в MATLAB Coder процедура генерации кода занимает несколько минут, в отличие от написания кода на языках C/C++ вручную. Генерация C/C++-кода может осуществляться с учетом особенностей целевого устройства с возможностью оптимизации программного кода по объему и быстродействию;

2) в общем случае полученный программный код может быть портирован на любой ЦСП, так как он является стандартным кодом C/C++ (ANSI/ISO C/C++);

3) легкость чтения программного кода на языках C/C++. Это связано с тем, что при переходе от программного кода на языке MATLAB в код на языке C/C++ сохраняется последовательность основных операций. Код генерируется последовательно (по принципу «сверху вниз»);

4) генерация кода может осуществляться для работы с данными с фиксированной точкой.

3.1.2. Генерация C/C++-кода на основе Simulink-модели

Приведем процедуру генерации программного кода для ЦСП с применением Simulink Coder. Она включает следующие этапы:

1) разработка алгоритма работы системы ЦОС;
2) создание Simulink-модели;
3) настройка Simulink-модели;
4) проверка работоспособности Simulink-модели;
5) настройка параметров Simulink Coder;
6) генерация C/C++кода для целевого устройства;
7) реализация проекта в интегрированной среде разработки;
8) модернизация программного кода для работы с периферийными устройствами;
9) отладка программного кода;
10) получение выходного файла;
11) загрузка выходного файла в память ЦСП.

Основным достоинством данной технологии является возможность работы с системой ЦОС в режиме блочного моделирования с последующей генерацией программного кода для ЦСП. Недостатком данной технологии является сложность чтения полученного программного кода.

3.2. Реализация нерекурсивного и рекурсивного фильтров на ЦСП с применением Simulink

В качестве примера рассмотрим процедуру проектирования цифровых КИХ- и БИХ-фильтров для ЦСП TMS320C5515 с применением Simulink. Для синтеза и реализации на ЦСП может быть применена технология автоматической генерации C/C++-кода в среде MATLAB с последующим переносом в интегрированную среду разработки Code Composer Studio. Синтез и реализация КИХ- и БИХ-фильтров на базе ЦСП TMS320C5515 в соответствии с данной технологией состоит из следующих четырех этапов:

1) создание Simulink-моделей КИХ- и БИХ-фильтров;

2) симуляция Simulink-моделей КИХ- и БИХ-фильтров;

3) автоматическая генерация программного кода из Simulink-моделей КИХ- и БИХ-фильтров с последующим переносом в интегрированную среду разработки Code Composer Studio;

4) тестирование и верификация синтезированного кода в среде Code Composer Studio.

Рассмотрим каждый из этих этапов более подробно.

3.2.1. Создание Simulink-моделей КИХ- и БИХ-фильтров

Simulink-модели КИХ- и БИХ-фильтров представлены на рис. 3.1 и включают следующие блоки:

- Discrete Impulse – источник цифрового единичного импульса;
- Scope – осциллограф;
- Vector Scope – блок, позволяющий строить графики исследуемых сигналов как функции времени или частоты;
- Digital Filter Design – средство проектирования цифровых фильтров;
- fir_filter – блок КИХ-фильтра, реализованный средствами Digital Filter Design;
- iir_filter – блок БИХ-фильтра, реализованный средствами Digital Filter Design.

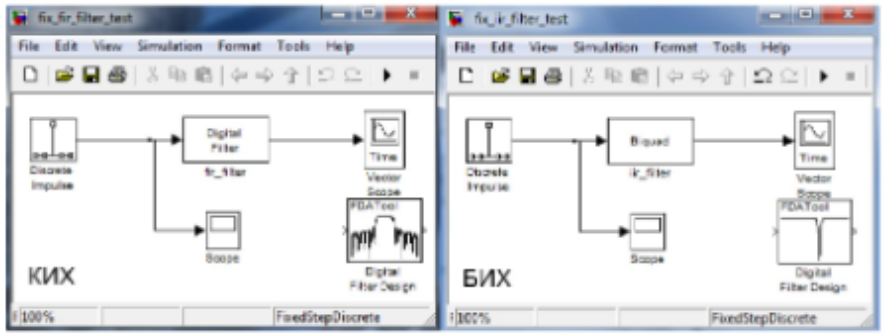


Рис. 3.1. Simulink-модели КИХ- и БИХ-фильтров

Блок Discrete Impulse генерирует цифровой единичный импульс для проверки работоспособности КИХ- и БИХ-фильтров. Для данного блока устанавливаются следующие настройки (рис. 3.2):

- Delay (samples): 0 – указывает задержку сигнала по оси абсцисс (время) в отсчетах сигнала;
- Sample time: 1/8000 – период дискретизации (для частоты дискретизации 8000 Гц);

- Samples per frame: 1 – количество отсчетов сигнала за одну выборку;
- Output data type: fixdt (1, 16, 15) – формат представления данных Q15 (с фиксированной точкой; 16 бит – длина слова данных; 1 знаковый бит; 15 бит – число значащих разрядов справа от разделительной точки).

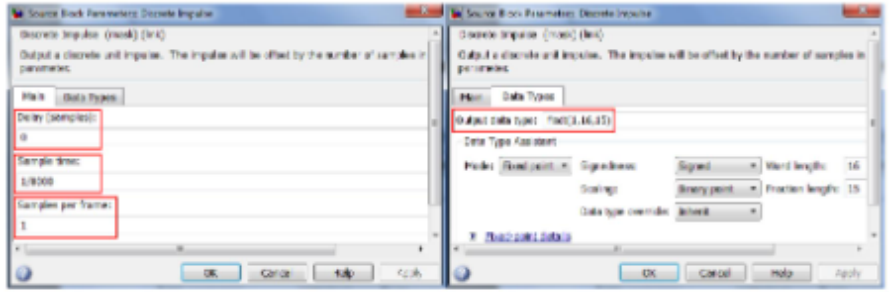


Рис. 3.2. Установленные настройки блока Discrete Impulse

Блок Scope не требует каких-либо настроек.

В свойствах блока Vector Scope на вкладке Scope Properties устанавливаются настройки, приведенные на рис. 3.3:

- Input Domain: Time – указывает, в какой области отображать данные (во временной или в частотной); необходимо выбрать – во временной;
- Time display span (number of frames): 50 – временной интервал отображения;

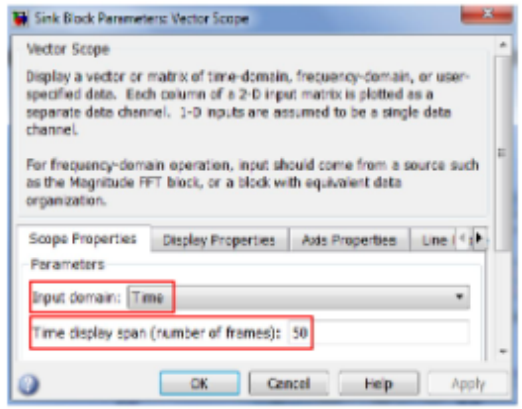


Рис. 3.3. Настройка блока Vector Scope

Блок Digital Filter Design позволяет выполнять проектирование цифровых фильтров.

Для КИХ-фильтра устанавливаются следующие настройки блока Digital Filter Design (рис. 3.4):

- Response Type: Bandpass – выбор типа избирательности цифрового фильтра (полосовой фильтр);
- Design Method: FIR – выбор типа цифрового фильтра (КИХ);
- Design Method: Window – выбор метода синтеза цифрового фильтра (метод окон);
- Filter Order/Specify order: 40 – порядок цифрового фильтра;
- Options/Window: Rectangular – выбор типа окна (прямоугольное окно);
- Frequency Specifications/Fs: 8000 – частота дискретизации (8 кГц);
- Frequency Specifications/Fc1: 1500 – граничная частота (Гц);
- Frequency Specifications/Fc2: 2500 – граничная частота (Гц).

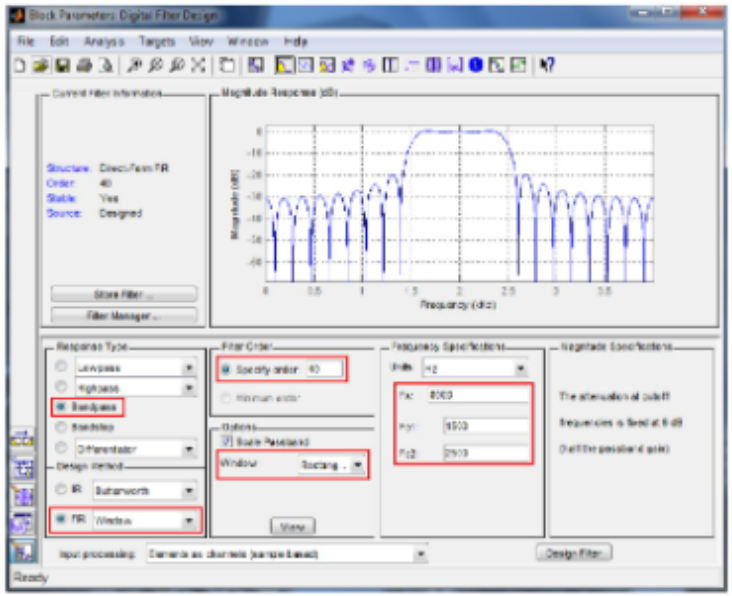


Рис. 3.4. Установка параметров КИХ-фильтра
в блоке Digital Filter Design

Для БИХ-фильтра установлены следующие настройки блока Digital Filter Design (рис. 3.5):

- Response Type: Bandstop – выбор типа избирательности цифрового фильтра (режекторный фильтр);
- Design Method: IIR – выбор типа цифрового фильтра (БИХ);

- Design Method: Butterworth – выбор аналогового фильтра-прототипа (фильтр Баттерворта);
- Filter Order/Specify order: 10 – порядок цифрового фильтра;
- Frequency Specifications/Fs: 8000 – частота дискретизации (Гц);
- Frequency Specifications/Fc1: 2000 – граничная частота (Гц);
- Frequency Specifications/Fc2: 2500 – граничная частота (Гц).

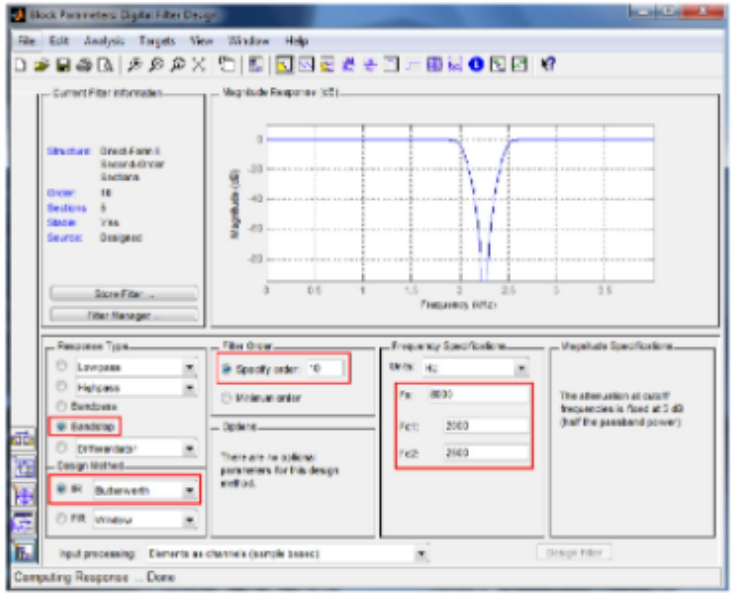


Рис. 3.5. Установка параметров БИХ-фильтра
в блоке Digital Filter Design

Блоки fir_filter и iir_filter, сгенерированные на основе блока Digital Filter Design, не требуют дополнительных настроек.

3.2.2. Симуляция Simulink-моделей КИХ- и БИХ-фильтров

Так как выбранный ЦСП работает с данными в формате с фиксированной точкой, перед проведением симуляции Simulink-моделей, работающих с данными в формате с плавающей точкой, их необходимо преобразовать.

С этой целью следует выбрать пункт меню Tools/Fixed-Point Tool… В окне Fixed-Point Tool необходимо нажать кнопку Fixed-Point Adviser . В результате откроется диалоговое окно Fixed-Point Adviser (рис. 3.6), где

требуется пройти все этапы перехода от формата представления данных с плавающей точкой к формату представления данных с фиксированной точкой, поочередно нажимая для каждого пункта кнопку Run This Task. По мере преобразования моделей для работы с данными в формате с фиксированной точкой каждый пункт перехода должен отмечаться значком , свидетельствующей о том, что этап успешно пройден. Если какой-либо из этапов не пройден, то необходимо определить причину: в ручным режиме или автоматическом, нажав кнопку Modify All в правой части диалогового окна Fixed-Point Adviser. Второй вариант более предпочтителен, так как значительно ускоряет процесс перехода.

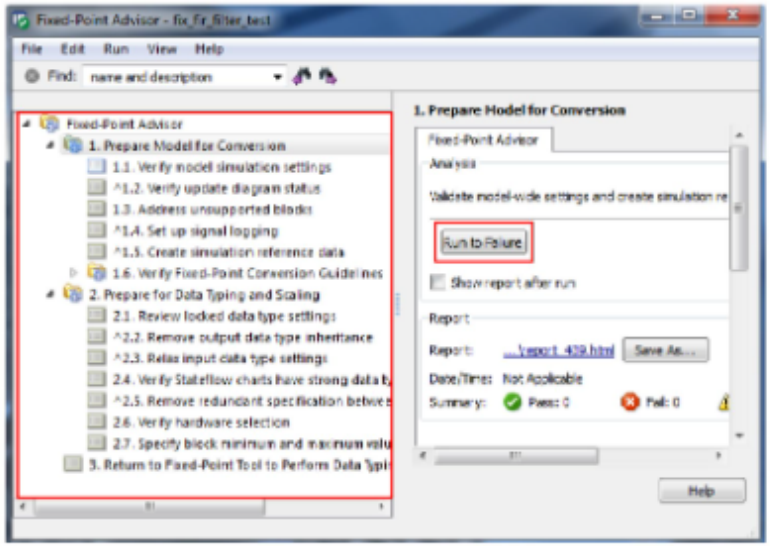
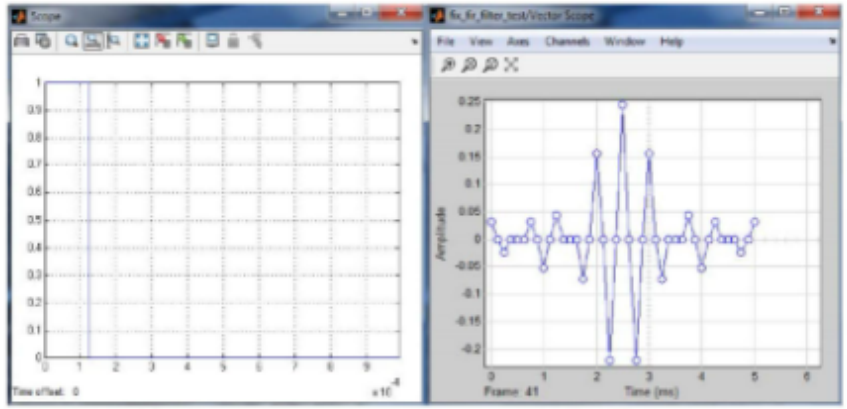


Рис. 3.6. Диалоговое окно Fixed-Point Adviser

Запуск симуляции Simulink-моделей осуществляется нажатием кнопки Start Simulation , расположенной на панели инструментов, либо комбинацией клавиш Ctrl + T.

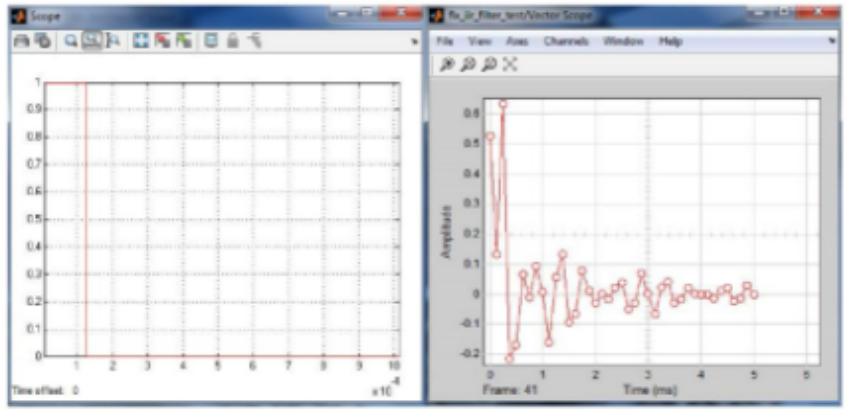
Результатом симуляции Simulink-моделей являются графики, полученные с помощью блоков Scope и Vector Scope. График на выходе блока Scope отображает сигнал на входе цифрового фильтра – цифровой единичный импульс, а график на выходе блока Vector Scope – сигнал на выходе цифрового фильтра – импульсную характеристику.

Графики сигналов на входе и выходе КИХ- и БИХ-фильтров представлены на рис. 3.7 и 3.8 соответственно.



Блок Scope Блок Vector Scope

Рис. 3.7. Результаты симуляции Simulink-модели КИХ-фильтра



Блок Scope Блок Vector Scope

Рис. 3.8. Результаты симуляции Simulink-модели БИХ-фильтра

3.2.3. Автоматическая генерация программного кода

Перед генерацией программного кода из Simulink-моделей КИХ- и БИХ-фильтров необходимо их модифицировать, удалив лишние блоки (Discrete Impulse, Scope, Vector Scope, Digital Filter Design), и добавить блоки, имитирующие входной и выходной порты процессора (блоки In1 и Out1 соответственно).

Преобразованные модели Simulink КИХ- и БИХ-фильтров показаны на рис. 3.9.

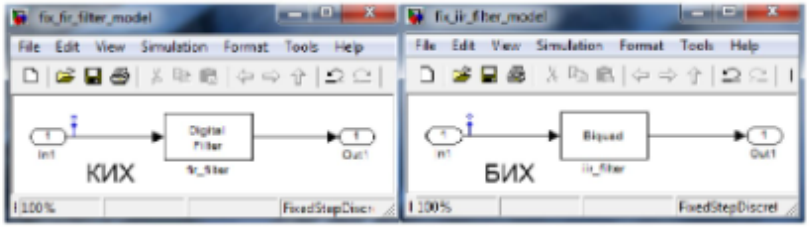


Рис. 3.9. Преобразованные модели Simulink КИХ- и БИХ-фильтров

Параметры сигналов, проходящих через блоки In1, показаны на рис. 3.10:

- Data type: fixdt (1, 16, 15) – формат представления данных Q15;
- Port dimensions (–1 for inherited): 1 – определяет размерность входного сигнала – вектор размером 1×1 ;
- Sample time (–1 for inherited): 1/8000 – период дискретизации (частота дискретизации 8000 Гц);
- Signal Type: real – выбор типа сигнала – реальный;
- Sampling mode: Sample based – выбор режима выборки сигнала – по одному отсчету.

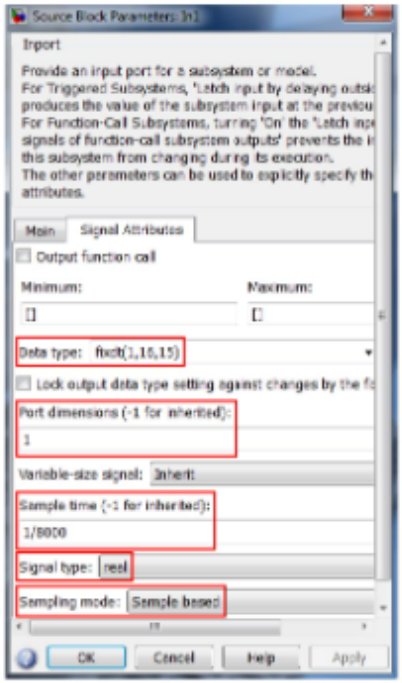


Рис. 3.10. Установленные настройки блоков In1

Для блоков Out1 настройка параметров сигналов не требуется.

Поскольку исходные Simulink-модели КИХ- и БИХ-фильтров были модифицированы, то их необходимо заново преобразовать в модели, оперирующие данными в формате с фиксированной точкой.

Далее следует провести предварительные настройки моделей, связанные с генерацией кода. Для этого необходимо нажать комбинацию клавиш Ctrl + E. В открывшемся окне (рис. 3.11) необходимо установить следующие параметры:

- Hardware Implementation/Device vendor: Texas Instruments – выбор производителя микропроцессоров;
- Hardware Implementation/Device type: C5000 – выбор серии микропроцессоров;
- Code Generation/System target file: ert.tcl – выбор цели, преследуемой в процессе синтеза кода; в этом случае все настройки оптимальным образом подобраны для синтеза кода именно для систем ЦОС, а также для встраиваемых систем;
- Code Generation/Language: C – выбор языка программирования.

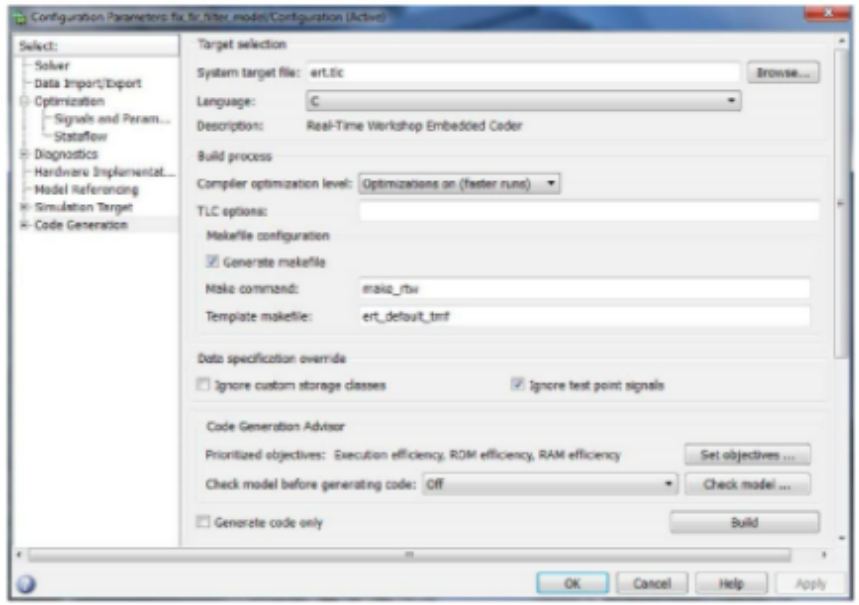


Рис. 3.11. Окно настроек Simulink-моделей КИХ- и БИХ-фильтров

После выбора настроек может быть выполнена генерация кода из Simulink-моделей. Для запуска диалогового окна Model Explorer (рис. 3.12),

вызываемого выбором пункта меню View/Model Explorer либо комбинацией клавиш Ctrl + H, собраны все неграфические параметры модели. Для того, чтобы синтезировать код, необходимо выбрать Configuration, далее выбрать Code Generation, после чего в правой части окна появится набор вкладок, сходный с набором вкладок настроек Simulink-моделей (рис. 3.11), которые уже прошли предварительную настройку. В правой нижней части окна Model Explorer находится группа Code Generation Advisor. В этой группе, нажав кнопку Set Objectives…, можно выбрать вид оптимизации. В данном случае выбран вид оптимизации по производительности и по компактности кода в части RAM (ОЗУ) и в части ROM (ПЗУ). Нажатие кнопки Check Model… запускает интерфейс консультанта, который последовательно проверит все настройки в автоматическом режиме. Если какие-то тесты были выполнены с ошибкой, производится поиск причины: ручным способом или нажатием кнопки Modify Parameters.

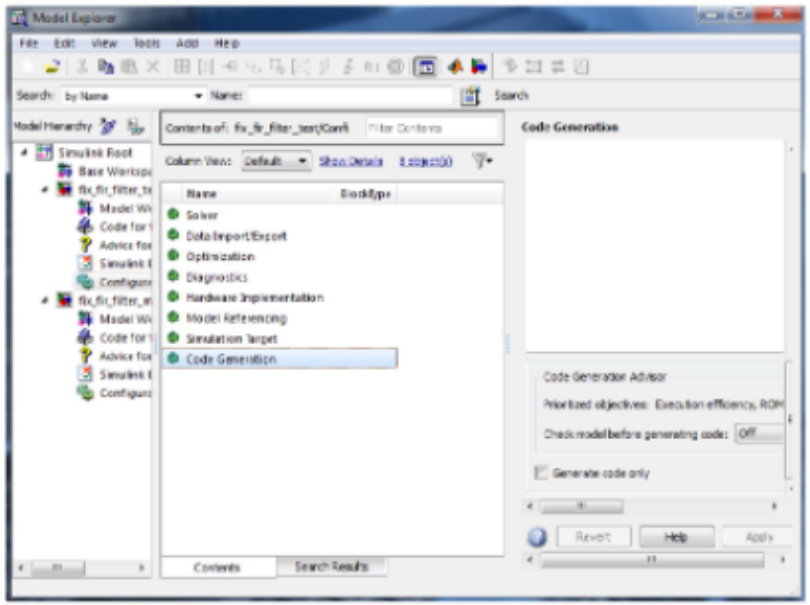


Рис. 3.12. Диалоговое окно Model Explorer

Процесс синтеза кода производится нажатием кнопки Build, расположенной под группой Code Generation Advisor. По окончании генерации кода выводится отчет, представленный на рис. 3.13. Отчет содержит список файлов программного кода, а также описание основных объектов кода: переменные, константы, функции.

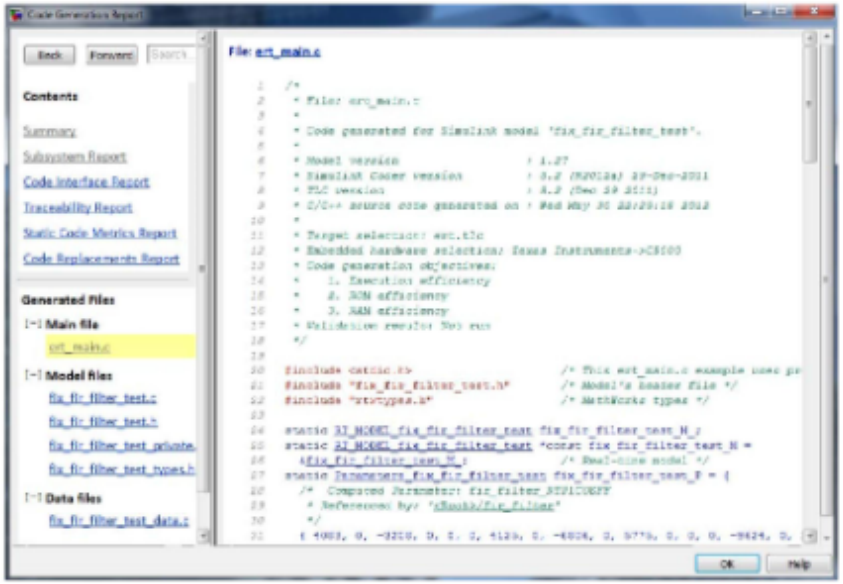


Рис. 3.13. Отчет о генерированном коде

3.2.4. Тестирование и верификация синтезированного кода в среде Code Composer Studio

Полученный программный код подвергается тестированию и верификации в ИСР Code Composer Studio на ЦСП TMS320C5515. При этом выполняются следующие операции.

1. Выбор рабочей области.

Перед созданием проекта необходимо выбрать рабочую область – папку, в которой будут сохраняться все персональные настройки для среды Code Composer Studio. Для этого нужно выбрать пункт меню File/Switch Workspace/Other… Отобразится диалоговое окно выбора рабочей области, где в поле Workspace необходимо указать путь к папке с будущим проектом.

2. Создание проекта.

Для создания нового проекта необходимо выбрать Project/New CCS Project в главном окне среды Code Composer Studio. Появится диалоговое окно мастера создания проектов, где в поле Project Name необходимо ввести название проекта, затем нажать кнопку Finish.

3. Добавление в проект файлов с исходным кодом, заголовочных файлов, а также командного файла компоновки.

Далее в проект подключаются заранее подготовленные файлы с исходным кодом, заголовочные файлы, полученные посредством автоматической генерации программного кода, и командный файл компоновки для указания порядка соединения секций и используемых областей памяти для их размещения. Для этого эти файлы нужно скопировать в папку с проектом из рабочей папки MATLAB, где они были размещены сразу после синтеза.

4. Создание и настройка файла конфигурации.

Для отладки проекта требуется создать файл конфигурации: выбрать пункт меню File/New/Target Configuration File. В следующем окне нужно указать его имя в поле File Name, нажать кнопку Finish, после чего файл будет создан, и откроется окно, в котором осуществляется настройка файла конфигурации, где необходимо выбрать требуемый тип эмулятора из выпадающего списка Connection – Spectrum Digital DSK–EVM-eZdsp onboard USB Emulator, в списке Device выбрать отладочную плату – EVM5515, сохранить настройки подключения, нажав кнопку Save.

5. Компиляция проекта; получение бинарного исполняемого файла с расширением *.out.

Чтобы скомпилировать проект нужно нажать кнопку на панели быстрых кнопок. Произойдет компиляция проекта, после чего в разделе Binaries окна проводника проекта будет находиться бинарный исполняемый файл с расширением *.out. При нажатии кнопки произойдет подключение к отладочной плате и загрузка в память процессора программного кода, среда Code Composer Studio перейдет в режим отладки.

6. Подготовка входных данных – файла в формате *.dat с отсчетами входного сигнала (цифрового единичного импульса) в системе MATLAB, а также установка контрольных точек для ввода входных данных из файла и вывода результатов в файл.

7. Запуск программного кода на исполнение и просмотр результатов тестирования средствами визуализации ИСР Code Composer Studio.

Графические результаты тестирования и верификации автоматически синтезированного программного кода, реализующего алгоритмы КИХ- и БИХ-фильтрации, представлены на рис. 3.14 и 3.15 соответственно.

Данные, полученные на этапе тестирования, совпадают с данными, полученными на этапе моделирования, что свидетельствует о корректной работе программного кода, сгенерированного автоматически средствами MATLAB.

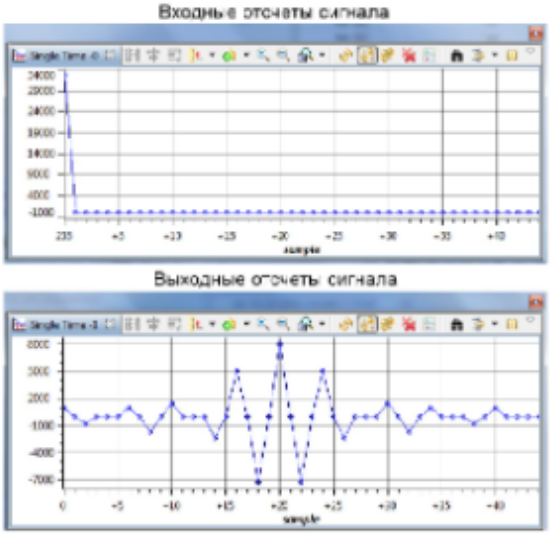


Рис. 3.14. Результаты тестирования автоматически сгенерированного программного кода, реализующего алгоритм КИХ-фильтрации

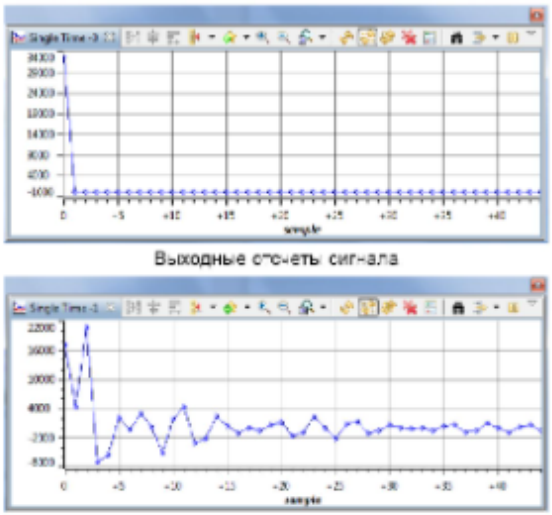


Рис. 3.15. Результаты тестирования автоматически сгенерированного программного кода, реализующего алгоритм БИХ-фильтрации

3.3. Реализация алгоритма быстрого преобразования Фурье на ЦСП с применением Simulink

Для реализации алгоритма БПФ на базе ЦСП TMS320C5515 также, как и для реализации КИХ- и БИХ-фильтров, применена технология автоматической генерации С-кода с последующим переносом в интегрированную среду разработки Code Composer Studio. Реализация алгоритма БПФ на базе ЦСП TMS320C5515 в соответствии с данной технологией состоит из следующих этапов.

1. Создание Simulink-модели алгоритма БПФ.
2. Симуляция Simulink-модели алгоритма БПФ.
3. Автоматическая генерация программного кода из Simulink-модели алгоритма БПФ с последующим переносом в интегрированную среду разработки Code Composer Studio.
4. Тестирование и верификация синтезированного кода в среде Code Composer Studio.

3.3.1. Создание Simulink-модели алгоритма БПФ

Simulink-модель алгоритма БПФ представлена на рис. 3.16 и включает следующие блоки:

- Sine Wave_1 и Sine Wave_2 – источники цифрового синусоидального сигнала;
- Sum – блок, выполняющий сложение двух сигналов;
- Scope – осциллограф;
- Vector Scope – блок, позволяющий строить графики исследуемых сигналов как функции частоты;
- Buffer – блок, осуществляющий накопление входных отсчетов сигнала в размере, указываемом в настройках *N*;
- Magnitude FFT – блок, выполняющий вычисление 256-точечного БПФ.

Блок Sum выполняет сложение двух синусоидальных сигналов, генерируемых блоками Sine Wave_1 и Sine Wave_2. Блок Sine Wave_1 имеет следующие настройки:

- Amplitude: 0,6 – амплитуда синусоидального сигнала (данный параметр может принимать значения от 0 до 1);
- Frequency (Hz): 1000 – частота синусоидального сигнала (Гц);
- Phase offset (rad): 0 – начальная фаза синусоидального сигнала (рад);
- Computation method: Table lookup – метод генерации сигнала (метод прямого цифрового синтеза (DDS));

- Sample time: 1/8000 – период дискретизации (для частоты дискретизации 8000 Гц);
- Samples per frame: 1 – количество отсчетов сигнала за одну выборку;
- Output data type: fixdt (1, 16, 15) – формат представления данных Q15.

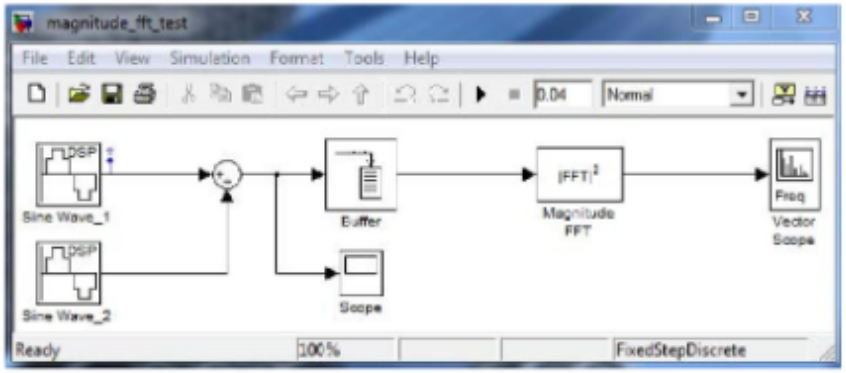


Рис. 3.16. Simulink-модель алгоритма БПФ

Блок Sine Wave_2 имеет те же настройки, что и блок Sine Wave_1, за некоторым исключением:

- Amplitude: 0,3 – амплитуда синусоидального сигнала;
- Frequency (Hz): 2000 – частота синусоидального сигнала (Гц).

Блок Scope не требует настроек.

У блока Vector Scope устанавливаются следующие настройки:

- Scope Properties/Input Domain: Frequency – указывает, в какой области отображать данные (в частотной);
- Axis Properties/Frequency Range: [0…Fs/2] – задает отображение спектра сигнала от 0 до Fs/2 Гц (0…4000 Гц).

В свойствах блока Buffer в группе Parameters нужно ввести в поле Output buffer size (per channel) число 256, что означает накопление отсчетов входного сигнала до 256, после чего блок выдает их на выход в виде вектора размером 256 × 1.

В свойствах блока Magnitude FFT в группе Parameters нужно выбрать из выпадающего списка Output параметр Magnitude squared и отметить галочкой пункт Inherit FFT length from input dimensions.

На этом этапе построение Simulink-модели алгоритма БПФ, а также установки настроек, связанных с этой моделью, завершены.

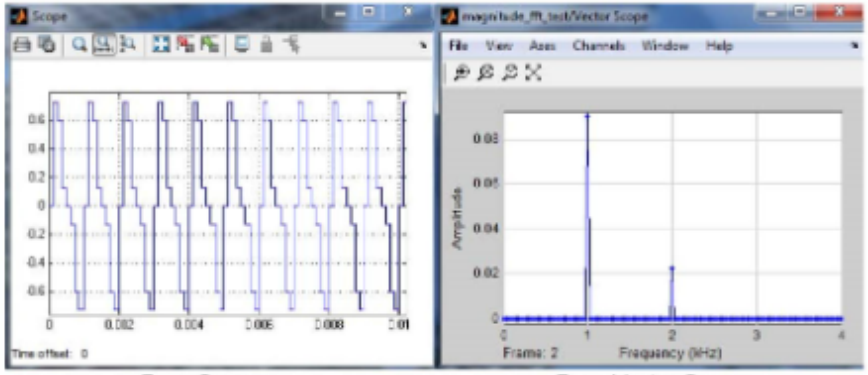
3.3.2. Симуляция Simulink-модели алгоритма БПФ

Преобразование Simulink-модели алгоритма БПФ, работающую с данными в формате с плавающей точкой, в модель, оперирующую данными в формате с фиксированной точкой, осуществляется способом, описанным ранее.

Запуск симуляции Simulink-модели осуществляется нажатием кнопки Start simulation , расположенной на панели инструментов, либо комбинацией клавиш Ctrl + T.

Результатом симуляции Simulink-модели являются 2 графика, полученные с помощью блоков Scope и Vector Scope. График слева (блок Scope) отображает сигнал (сумма двух синусоидальных сигналов с частотами 1 и 2 кГц) на входе модели, выполняющую БПФ, а график справа (блок Vector Scope) – данные на выходе модели (спектр входного сигнала).

Графики входных/выходных сигналов модели представлены на рис. 3.17.



Блок Scope Блок Vector Scope

Рис. 3.17. Результаты симуляции Simulink-модели алгоритма БПФ

3.3.3. Автоматическая генерация программного кода из Simulink-модели алгоритма БПФ

Перед генерацией программного кода из Simulink-модели алгоритма БПФ необходимо ее представить в преобразованном виде, так как она содержит много лишних блоков: Sine Wave_1, Sine Wave_2, Sum, Scope, Vector Scope.

Преобразованная Simulink-модель алгоритма БПФ показана на рис. 3.18.

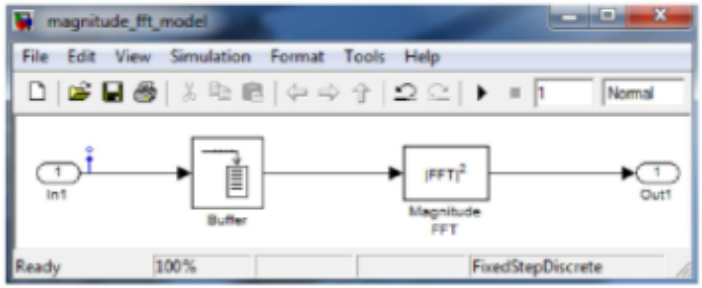


Рис. 3.18. Преобразованная Simulink-модель алгоритма БПФ

На рис. 3.18 видно, что в модели присутствуют два новых блока – In1 и Out1, представляющие собой порты ввода и вывода соответственно. Параметры сигналов, проходящих через блок In1, следующие:

- Data type: fixdt (1, 16, 15) – формат представления данных Q15;
- Port dimensions (-1 for inherited): 1 – определяет размерность входного сигнала – вектор размером 1×1 ;
- Sample time (-1 for inherited): 1/8000 – период дискретизации (для частоты дискретизации 8000 Гц);
- Signal Type: real – выбор типа сигнала – реальный;
- Sampling mode: Sample based – выбор режима выборки сигнала – по одному отсчету.

Для блока Out1 настройка параметров сигналов не требуется.

Поскольку исходная модель Simulink алгоритма БПФ была модифицирована, то ее необходимо заново преобразовать в модель, оперирующую данными в формате с фиксированной точкой.

Далее следует провести предварительные настройки модели, связанные с генерацией кода.

3.3.4. Тестирование синтезированного кода в среде Code Composer Studio

Код, полученный на предыдущем этапе, подвергается тестированию и верификации в ИСР Code Composer Studio на ЦСП TMS320C5515.

Графические результаты тестирования и верификации автоматически синтезированного программного кода, реализующего алгоритм БПФ, представлены на рис. 3.19. График исследуемого сигнала выводится средствами ИСР Code Composer Studio, а график спектра исследуемого сигнала выводится в результате выполнения программы, написанной на языке MATLAB.

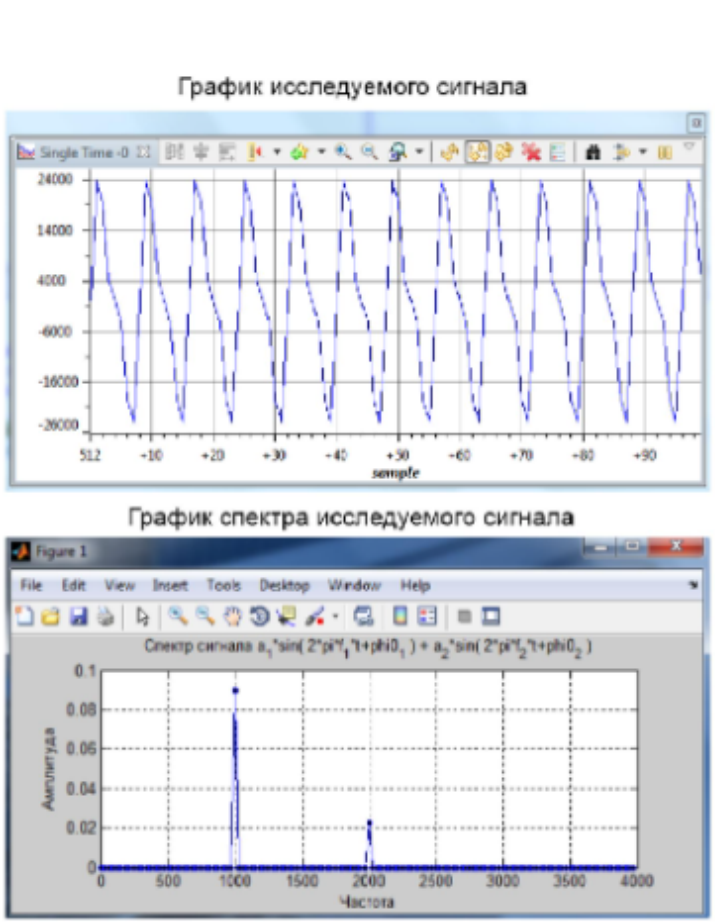


Рис. 3.19. Результаты тестирования
автоматически синтезированного программного кода,
реализующего алгоритм БПФ

Данные, полученные на этапе тестирования, совпадают с данными, полученными на этапе моделирования, что свидетельствует о корректной работе программного кода, сгенерированного автоматически средствами MATLAB.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Айфичер, Э. С.* Цифровая обработка сигналов : практический подход / Э. С. Айфичер, Б. У. Джервис. – 2-е изд. ; пер. с англ. – Москва : Издательский дом «Вильямс», 2004. – 992 с.

2. *Голд, Б.* Цифровая обработка сигналов / Б. Голд, Ч. Рэйдер ; пер. с англ. ; под ред. А. М. Трахтмана. – Москва : Советское радио, 1973. – 368 с.

3. *Гольденберг, Л. М.* Цифровая обработка сигналов : справочник / Л. М. Гольденберг, Б. Д. Матюшкин, М. Н. Поляк. – Москва : Радио и связь, 1985. – 312 с.

4. *Оппенгейм, А. В.* Цифровая обработка сигналов / А. В. Оппенгейм, Р. В. Шафер ; пер. с англ.; под ред. С. Я. Шаца. – Москва : Связь, 1979. – 416 с.

5. *Солонина, А. И.* Основы цифровой обработки сигналов : курс лекций / А. И. Солонина, Д. А. Улахович, С. М. Абузов, Е. Б. Соловьева. – 2-е изд., испр. и перераб. – Санкт-Петербург : БХВ-Петербург, 2005. – 768 с.

6. *Солонина, А. И.* Алгоритмы и процессоры цифровой обработки сигналов / А. И. Солонина, Д. А. Улахович, Л. А. Яковлев. – Санкт-Петербург : БХВ-Петербург, 2002. – 464 с.