Министерство образования и науки Российской Федерации
«Южно-Уральский государственный университет»
(Национальный исследовательский университет)
«Высшая школа электроники и компьютерных наук»
Кафедра «Инфокоммуникационные технологии»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
_____ Даровских С.Н
“____”_____ 2018 год

Разработка Simulink модели цифровых приемника и передатчика с QPSK модуляцией

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ – Д 11.03.02.2018.211.00 ПЗ ВКР

Руководитель работы:
НиколаевА.Н. _____
“____”_____ 2018 год
Автор работы:
студентка группы КЭ -479
ЮрьеваП.А. _____
“____”_____ 2018 год
Нормоконтролер:
Спицына В.Д. _____
“____”_____ 2018 год

Челябинск 2018

РЕФЕРАТ

Юрьева П.А. Разработка Simulink модели цифровых приемника и передатчика с QPSK модуляцией – Челябинск: ЮУрГУ, ВШЭКН, 2018,15 илл.,62 с.– Библиографический список – 18наименований, 8 листов приложений, 2 плаката формата А1 (основные сведения о QPSK модуляции, структурная схема приемника и передатчика).

В выпускной квалификационной работе рассматриваются основные теоретические сведения о системах цифровой связис QPSK модуляцией. Описана структурная схема модели приемника и передатчика системы цифровой связи с QPSK модуляцией. Разработаны отдельные функциональные компоненты, входящие в состав приемника и передатчика, также приведены результаты их моделирования. Получено VHDL описание функциональных элементов модели приемника и передатчика системы цифровой связи с QPSK модуляцией для дальнейшей реализации на ПЛИС.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время современные достижения радиоэлектроники обеспечивают возможность реализовать в передатчике и приемнике системы связи достаточно сложные алгоритмы цифровой обработки электрических сигналов. В результате качество передачи практически любых сообщений в цифровых системах оказывается выше, чем качество передачи этих сообщений с помощью аналоговых систем связи. Например, оказалось возможным передавать сообщения в присутствии шума и помех с большей точностью или передавать больше сообщений при прочих равных условиях. При использовании цифровых технологий очень низкая частота ошибок делает возможным высокую точность сигнала. Так же цифровые каналы надежнее, допускают более гибкую реализацию. Технологии не стоят на месте и чем стремительнее развиваются компьютеры, тем сложнее они становятся. На одном кристалле могут размещаться тысячи, а то и миллионы элементов, и разработка таких устройств не может осуществляться на уровне изображенной на бумаге схемы. Поэтому были созданы специальные языки описания аппаратуры, включающие в себя основные понятия, существенные для проектирования цифровых систем.

В настоящее время существует проблема о наличие в свободном доступе VHDL кодов, которые необходимы для проектирования цифровых систем на программированных логических интегральных схемах (ПЛИС). Для решения данной проблемы был разработан код на языке VHDL для реализации приемника и передатчика цифровой связи на базе ПЛИС.

Цель выпускной квалификационной работы состоит в разработке приемника и передатчика системы цифровой связи с QPSK модуляцией на базе программируемой логической интегральной схемы. QPSK модуляция широко применяется в беспроводных профессиональных сетях (IEEE 802.15.4c и IEEE 802.15.4d), в автоматизации технологических процессов и жилья, идентификация товаров на складе и отслеживание движения товара, в различных технологиях, включая ZigBee, WiFi и радиочастотную идентификацию. Данный вид модуляции

является основным в стандарте спутникового цифрового телевидения. А также особое значение QPSK модуляция имеет в радиолокации.

Для достижения этой цели необходимо выполнить ряд задач и решить последовательно вопросы проектирования данных элементов в программно-инструментальной среде MATLAB Simulink.

Simulink – это графическая среда имитационного моделирования, позволяющая при помощи блок-диаграмм в виде направленных графов, строить динамические модели, включая дискретные, непрерывные и гибридные, нелинейные и разрывные системы. Дополнительные пакеты расширения Simulink позволяют решать весь спектр задач от разработки концепции модели до тестирования, проверки, генерации кода и аппаратной реализации. Simulink интегрирован в среду MATLAB, что позволят использовать встроенные математические алгоритмы, мощные средства обработки данных и научную графику.

Задачи:

а) разработка отдельных функциональных компонентов, входящих в состав приемника и передатчика;

б) тестирование отдельных компонентов;

в) сборка и тестирование полной готовой модели;

Исследование устройств в данном случае подразумевает под собой схемотехническое решение проблемы, выбор элементной базы, моделирование разрабатываемого устройства и экспериментальное измерение его параметров. Данный процесс можно считать итерационным, поскольку на каждом этапе возможно возвращение к этапу предыдущему для внесения изменений в концепцию или конструкцию устройства.

Пояснительная записка состоит из трех разделов. В первом разделе рассматриваются основные теоретические сведения, а также описана структурная схема модели приемника и передатчика системы цифровой связи. Во втором описываются отдельные функциональные схемы и результаты моделирования. Третий раздел посвящен VHDL описанию модели и результатам компиляции.

1 Описание структурной схемы модели приемника и передатчика системы цифровой связи с QPSK модуляции

В первом разделе рассматриваются основные теоретические сведения о модуляции и кодирование в системе цифровой связи с QPSK модуляцией, а также описана структурная схема модели приемника и передатчика системы цифровой связи с QPSK модуляцией.

1.1 Основные сведения о QPSK модуляции.

Фазовая манипуляция PSK

Фазовая манипуляция (phase-shift keying – PSK) является широко используемым способом цифровой модуляции, при которой данные кодируются путем изменения фазы несущего сигнала. Сигнал на несущей частоте с фазой 0° декодируется как один бит или набор битов, а с фазой 90° - как другой бит. Имеется два основных метода декодирования таких сигналов: дифференциальный и недифференциальный. Демодулятор, содержащий специальную схему PSK модуляции, может сравнивать фазу входящего сигнала с опорным сигналом или же сравнивать только изменения в фазе сигнала на несущей частоте, как это происходит при определении следующего символа.

Кодирование PSK

PSK – это метод, при котором фаза несущего сигнала соответствует определенному символу. При недифференциальном кодировании на демодулятор подается опорный сигнал, который сравнивается с входящим сигналом. Разность фаз, которая сопоставляется одному символу, впоследствии, если фаза принимаемого сигнала сдвигается на какую-то величину от фазы опорного сигнала, сопоставляется другому символу или набору символов. Таким образом, фазу предыдущего символа знать необязательно. Это облегчает декодирование, так как ненужным становится механизм запоминания [1].

Квадратурная фазовая манипуляция QPSK

QPSK модуляция строится на основе кодирования двух бит передаваемой информации одним символом (рисунок 1).

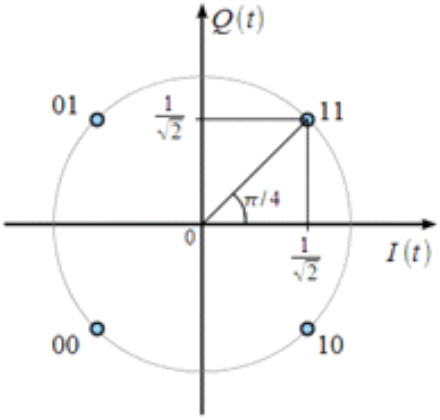

Рисунок 1 – Векторная диаграмма QPSK-сигнала

На векторной диаграмме изображено созвездие фазовой манипуляции. Для кодирования одним символом двух бит информации, необходимо, чтобы созвездие состояло из четырех точек. Все точки созвездия расположены на единичной окружности. Кодирование осуществляется следующим образом: разбивается битовый поток на четные и нечетные биты. Два последовательно идущих друг за другом бита информации кодируются одновременно синфазным и квадратурным сигналами. Если нечетный бит равен единице (биты нумеруются с нуля, а не с единицы, поэтому первый в очереди бит имеет номер 0, а значит он четный по порядку) и, если четный бит равен 0. Аналогично строится квадратурный канал, но только по нечетным битам. Длительность одного символа в два раза больше длительности одного бита исходной информации [2].

Устройство выполняющее такое кодирование и согласно созвездию QPSK условно показано на рисунке 2.

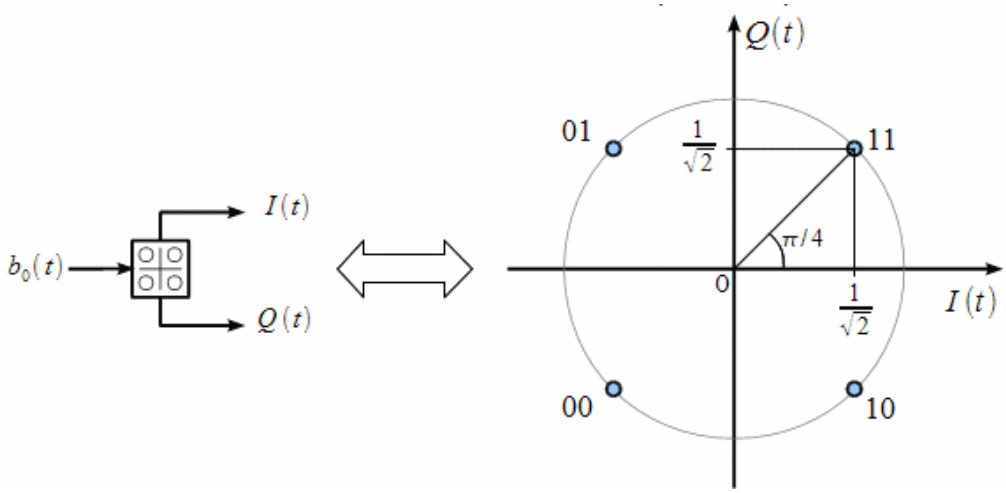

Рисунок 2 – Устройство кодирования синфазной и квадратурной составляющих на основе созвездия QPSK

Структурная схема QPSK модулятора на основе универсального квадратурного модулятора показана на рисунке 3.

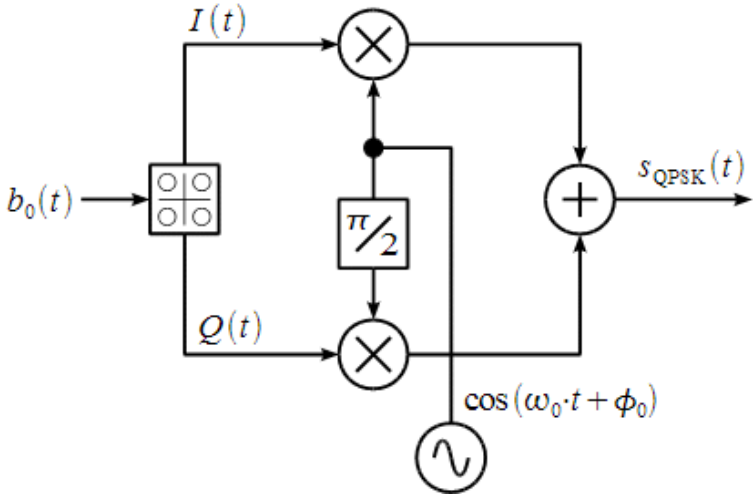

Рисунок 3 – Структурная схема QPSK модулятора

Сигнал имеет вид:

$$S_{QPSK}(t) = I(t) \cdot \cos(\omega_0 \cdot t + \phi_0) - Q(t) \cdot \sin(\omega_0 \cdot t + \phi_0). \quad (1)$$

Синфазная I(t) и квадратурная Q(t) составляющие это ничто иное, как реальная и мнимая части комплексной огибающей QPSK сигнала, которые

являются входными сигналами квадратурного модулятора. Тогда можно представить сигнал через его комплексную огибающую Z(t):

$$S_{QPSK}(t) = R[Z(t) \cdot \exp(j \cdot \omega_0 \cdot t)] . \quad (2)$$

Из комплексной огибающей можно выделить фазовую огибающую как:

$$\phi(t) = arctan\left(\frac{J[Z(t)]}{R[Z(t)]}\right) = \left(\frac{Q(t)}{I(t)}\right) \quad (3)$$

Важно отметить, что арктангенс должен вычисляться с учетом четверти комплексной плоскости. Вид фазовой огибающей для информационного потока «1100101101100001» показан на рисунке 4.

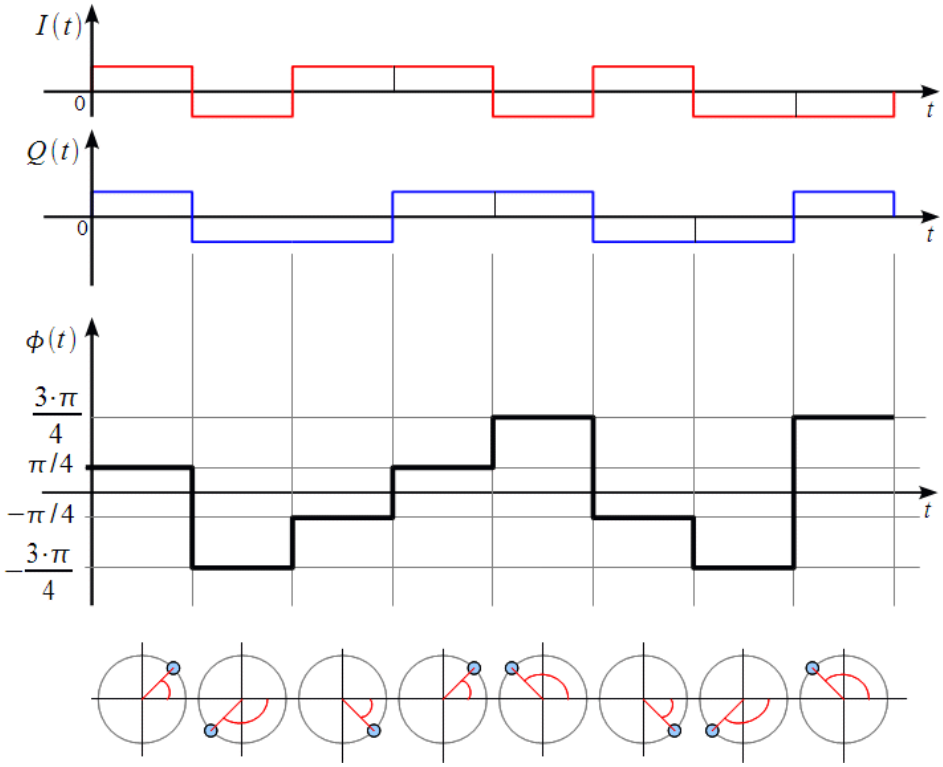

Рисунок 4 – Фазовая огибающая информационного потока «1100101101100001»

Фазовая огибающая представляет собой ступенчатую функцию времени, претерпевающую разрывы в моменты смены символа QPSK. При этом в пределах одного символа векторная диаграмма QPSK находится всегда в одной точке созвездия, как это показано внизу, а при смене символа – скачкообразно переходит в точку, соответствующую следующему символу. Поскольку у QPSK всего четыре точки в созвездии, то фазовая огибающая может принимать всего

четыре значения.

Амплитудная огибающая QPSK сигнала может быть получена из комплексной огибающей:

$$a(t) = \sqrt{I^2(t) + Q^2(t)} \quad (4)$$

Отметим, что амплитудная огибающая QPSK сигнала равна единице всюду, за исключением моментов смены передаваемых символов, т. е. в моменты перескока фазы и перехода очередной точке созвездия.

Пример осциллограммы QPSK сигнала при входном битовом потоке «1100101101100001» при скорости передачи информации 10 кбит/с и несущей частоте 20 кГц показан на рисунке 5.

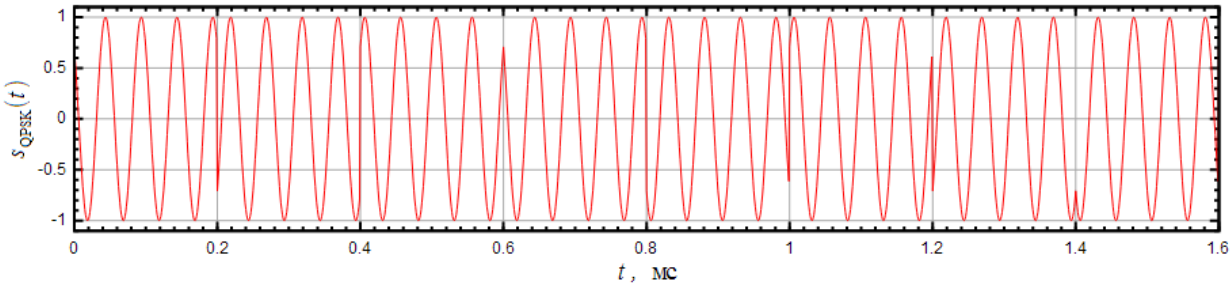

Рисунок 5 – Осциллограмма QPSK сигнала

Фаза несущего колебания может принимать четыре значения:

$$-\frac{\pi}{4}; +\frac{\pi}{4} \text{ и } -\frac{3\pi}{4}; +\frac{3\pi}{4} \text{ радиан.}$$

При этом фаза следующего символа относительно предыдущего может не изменится, или измениться на: $+\frac{\pi}{2}$, $-\frac{\pi}{2}$ или на $\pm\pi$ радиан. При скорости передачи информации 10 кбит/с мы имеем символьную скорость5 кбит/с и длительность одного символа T=0,2 мс, что отчетливо видно на осциллограмме (скачок фазы происходит через 0,2 мс).

1.2 Структурная схема приемника и передатчика с QPSK модуляцией

Основные элементы, из которых состоит система цифровой связи в общем виде показаны на рисунке 6.

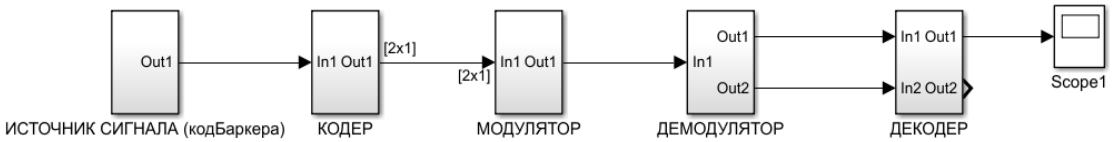

Рисунок 6 – Структурная схема приемника и передатчика с QPSK модуляцией

«Источник сигнала» выдает данные для передачи по каналу связи в цифровом виде (современные носители цифровой информации, различные датчики с цифровым интерфейсом и т. д.). В независимости от типа источника информации данные должны быть представлены в как можно более сжатом цифровом виде. В блоке «Кодер» происходит процесс эффективного преобразования данных в последовательность двоичных символов, который называется кодированием источника или сжатием данных. С выхода кодера данные поступают на передатчик цифровой системы связи, в котором выполняется цифровая модуляция – блок «Модулятор». Приемная часть цифровой системы связи содержит цифровые «Демодулятор» и «Декодер», интерфейс выдачи полезных данных пользователю (осциллограф) – блок «Scope» [3].

2. Моделирование функциональных узлов системы цифровой связи с QPSK модуляции

В данном разделе описываются отдельные функциональные схемы кодирующего устройства, модулятора, демодулятора, декодера. Приведены результаты моделирования, которые были получены в основных узлах модели приемника и передатчика системы связи с QPSK модуляцией.

2.1 Кодирующее устройство

Кодирующее устройство - устройство, накладывающее информацию одного электронного сигнала на другие электронные сигналы.

Первый и третий входные порты представляют собой порты данных, а второй входной порт - порт управления. Кодовая последовательность сигнала Баркера состоит из символов ± 1 и характеризуется нормированной АКФ вида:

$$B(\tau) = \begin{cases} 1 & \text{для } \tau = 0, \\ 0 & \text{для } \tau = 2l + 1, \\ \pm 1/N & \text{для } \tau = 2l, \end{cases}$$

где $l = 0, 1, \ldots (N-1)/2$.

Знак в последней строчке зависит от величины N. В таблице 1 приведены известные кодовые последовательности Баркера и их уровни боковых типов АКФ. Кодовые последовательности, обладающие свойствами (18), для N > 13 не найдены.

Таблица 1 – Кодовые последовательности Баркера

Код	Кодовая последовательность	Уровень боковых лепестков
3	1 1 -1	-1/3
4	1 1 -1 1	1/4
5	1 1 1 -1 1	1/5
7	1 1 1 -1 -1 1 –1	-1/7
11	1 1 1 -1 -1 -1 1 -1 -1 1	-1/11
13	1 1 1 1 1 -1 -1 1 1 -1 1 -1 1	1/13

На первый входной порт коммутатора Switch1 поступает источник сигнала код Баркера «Barker Code» с длинной последовательности 7 и с шагом модельного времени (Sample time) равным 0,007, а на третий входной порт поступает инверсный код Баркера «Barker Code» с длиной последовательности 7 (см. рис. 7).

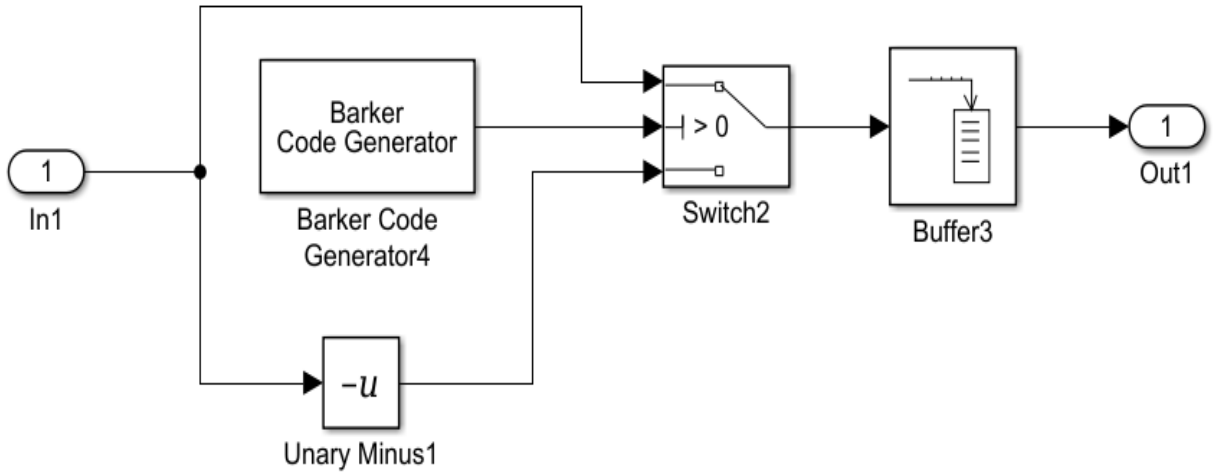

Рисунок 7 – Устройство кодирования синфазной и квадратурной составляющих на основе созвездия QPSK

Для разработки VHDL кода для дальнейшей реализации устройства кодирования синфазной и квадратурной составляющей на ПЛИС необходимо было реализовать блок «Barker Code» на простых логических операциях, так как стандартный блок MATLAB «Barker Code» не поддавался для генерации VHDL кода. Созданная на простых логических операциях модель блока «Barker Code» изображена на рисунке 8.

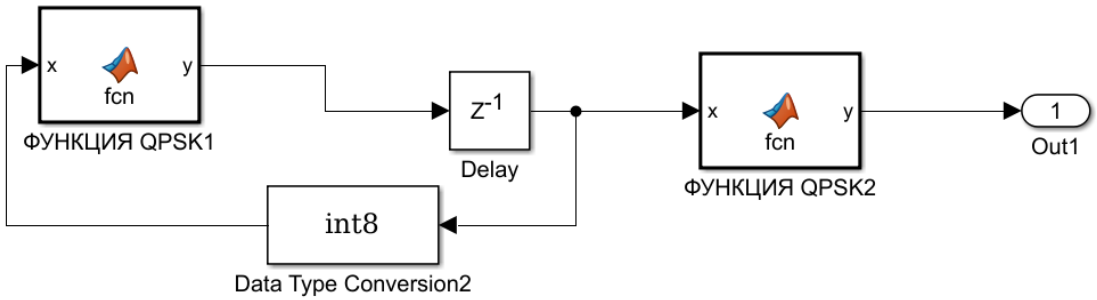

Рисунок 8 – Модель блока «Barker Code», созданная на простых логических операциях

На рисунке 9 изображена осциллограмма последовательности кода Баркера.

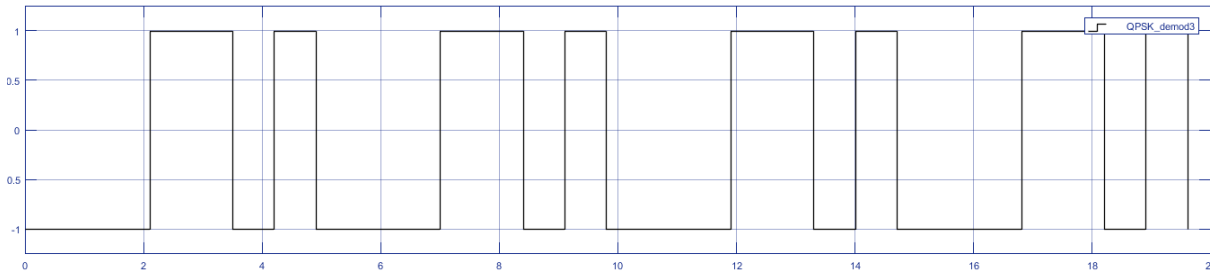

Рисунок 9 – Осциллограмма последовательности кода Баркера

На второй входной порт управления поступает сигнал кода Баркера «Barker Code1» с шагом модельного времени 0,001, который разрешает или запрещает прохождение входного сигнала на выход. На рисунке 10 изображена осциллограмма, полученная после прохождения сигнала через коммутатор.

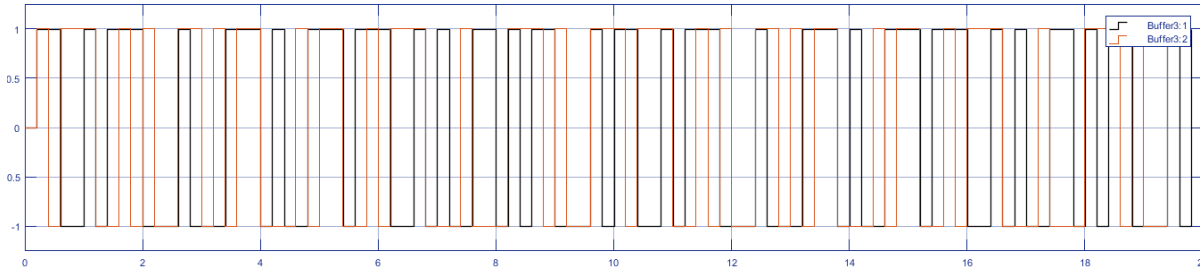

Рисунок 10 – Осциллограмма на выходе кодирующего устройства

Сигнал с выхода коммутатора Switch1 поступает в блок Buffer, где происходит преобразование скалярных выборок. Размер выходного буфера равен двум.

2.2 Модулятор

Модуля́тор (лат. modulator – соблюдающий ритм) – устройство, изменяющее параметры несущего сигнала в соответствии с изменениями передаваемого

(информационного) сигнала. Этот процесс называют модуляцией, а передаваемый сигнал модулирующим.

Модулятор является одной из составных частей передающих устройств радиосвязи, радиовещания и телевещания. Несущими являются высокочастотные гармонические колебания, а модулирующими — колебания звуковой частоты и видеосигналы. Также применяют в радиолокации, системах кодово-импульсной связи, телеуправления и телеметрии. Модуляторы, преобразующие постоянные напряжения в переменные, применяются в усилителях постоянного тока и нуль-органах, работающих по принципу модуляции — демодуляции, для устранения дрейфа нуля и повышения чувствительности аналоговых вычислительных устройств [4].

Схема модулятора в среде моделирования Matlab Simulink представлена на рисунке 11.

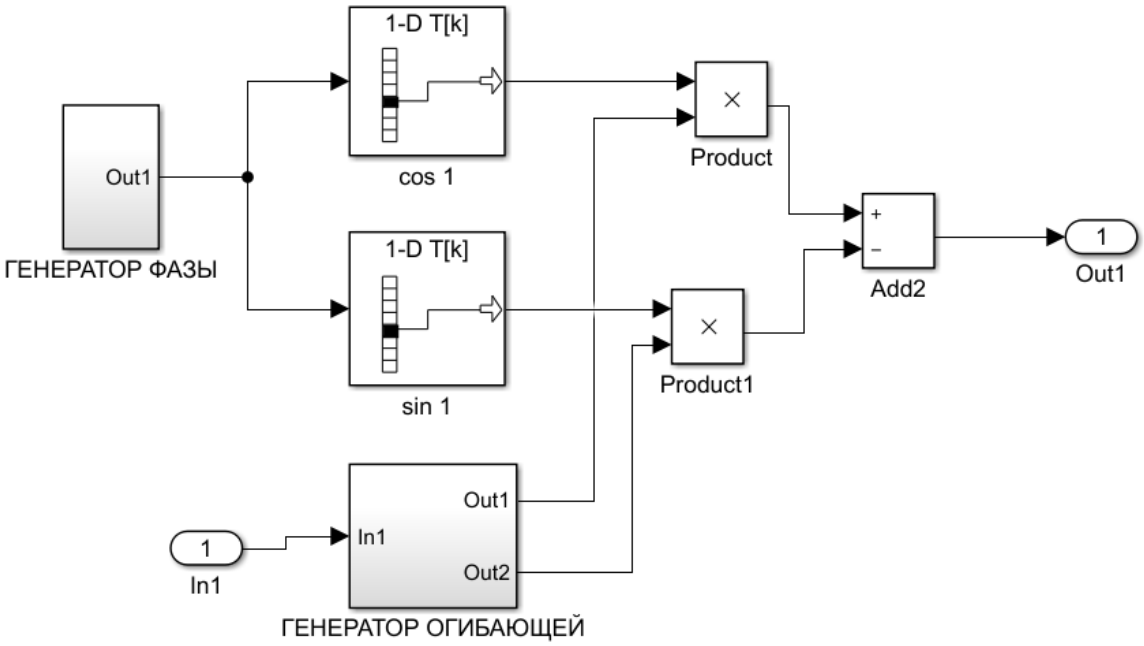

Рисунок 11 – Модель модулятора в среде моделирования Matlab Simulink

Два последовательно идущих друг за другом бита информации разбиваются на четные и нечетные биты и в соответствии с принимаемыми

значениями в блоке MATLAB Function кодируются одновременно синфазным и квадратурным сигналами. Синфазная и квадратурная составляющие ничто иное, как реальная и мнимая части сигнала.

На входе и выходе блока MATLAB Function сигнал имеет целочисленный тип данных - int8.

После преобразования действительная часть сигнала поступает на умножитель Product, где умножается на сигнал косинусоидальной формы, а мнимая часть поступает на умножитель Product1, где аналогично умножается, но только на сигнал синусоидальной формы [3].

Сигналы косинуса и синуса формируются в блоке Direct Lookup Table1 (cos 2) и Direct Lookup Table2 (-sin 2) соответственно. На их входы подается сигнал с постоянным значением, который проходит через линию задержки. Осциллограма косинуса и синуса изображена на рисунке 12.

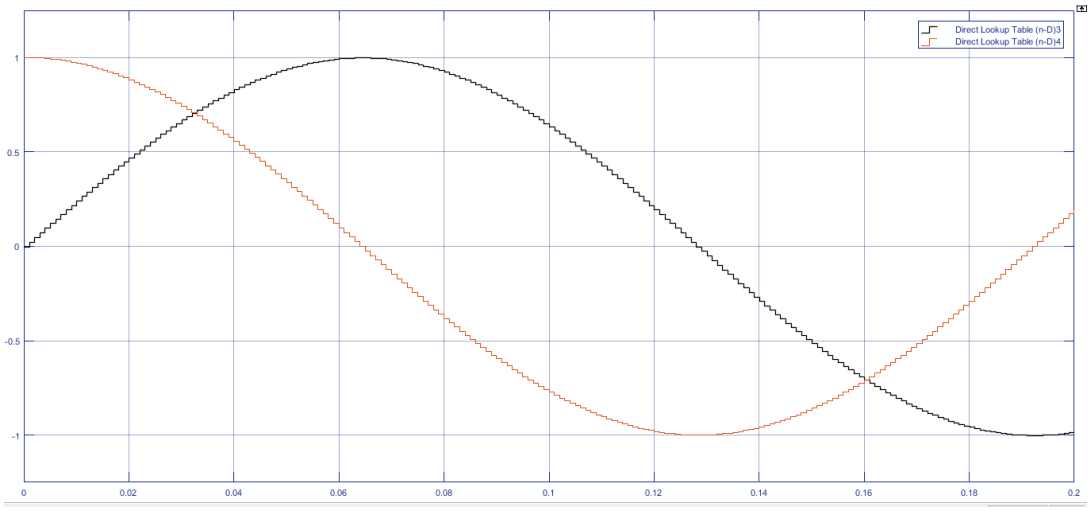

Рисунок 12 – Осциллограмма косинуса и синуса

Осциллограмма на выходе модулятора канала связи представлена на рисунке 13. Фаза несущего колебания может принимать четыре значения $\pm\frac{\pi}{4}$ и $\pm\frac{3\pi}{4}$ радиан, при этом на осциллограмме отчетливо видно, что фаза следующего символа относительно предыдущего меняется через 0,2 мс.

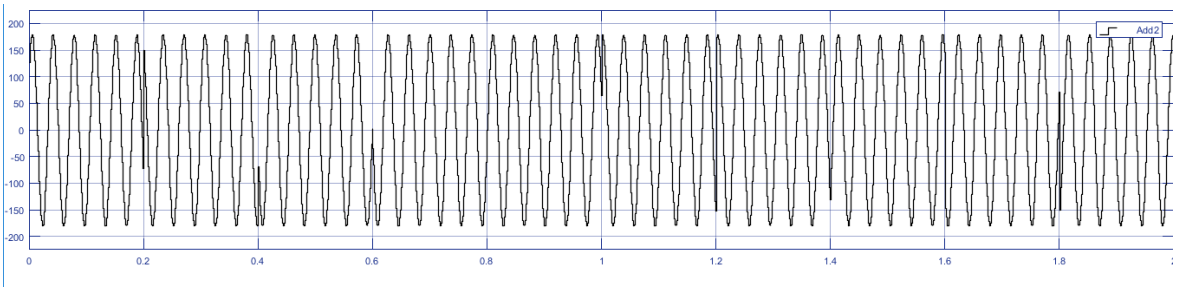

Рисунок 13 – Осциллограмма на выходе сумматора модулятора канала связи

2.3 Демодулятор

Демодулятор – радиотехническое устройство, предназначенное для выделения информационного сигнала из модулированного высокочастотного колебания. Процесс получения напряжение (тока), изменяющийся по закону модуляции, из модулированного напряжения высокой частоты называется демодуляцией (детектированием). В зависимости от вида модуляции в передающей части, в демодуляторе осуществляется амплитудная, частотная или фазовая демодуляция.

Фазовые демодуляторы. При демодуляции фазо-модулированного сигнала используют фазовые детекторы (демодуляторы). Фазовый детектор – это устройство, напряжение на выходе которого зависит от разности фаз двух сравниваемых напряжений одной частоты. На вход детектора должны подаваться сигналы с одной и той же частотой. Одним сигналом является фазо-манипулированный сигнал, а вторым – опорное колебание. Традиционное применение фазового демодулятора: в следящих системах автоподстройки частоты, где фазовый демодулятор, совместно с генератором переменной частоты, управляемый напряжением (ГУН) включены в контур отрицательной обратной связи. Сигналом задания для этой системы автоматического регулирования является частота входного сигнала, а фазовый демодулятор является сравнивающим устройством.

На рисунке 14 изображена модель демодулятора в среде моделирования

Matlab Simulink.

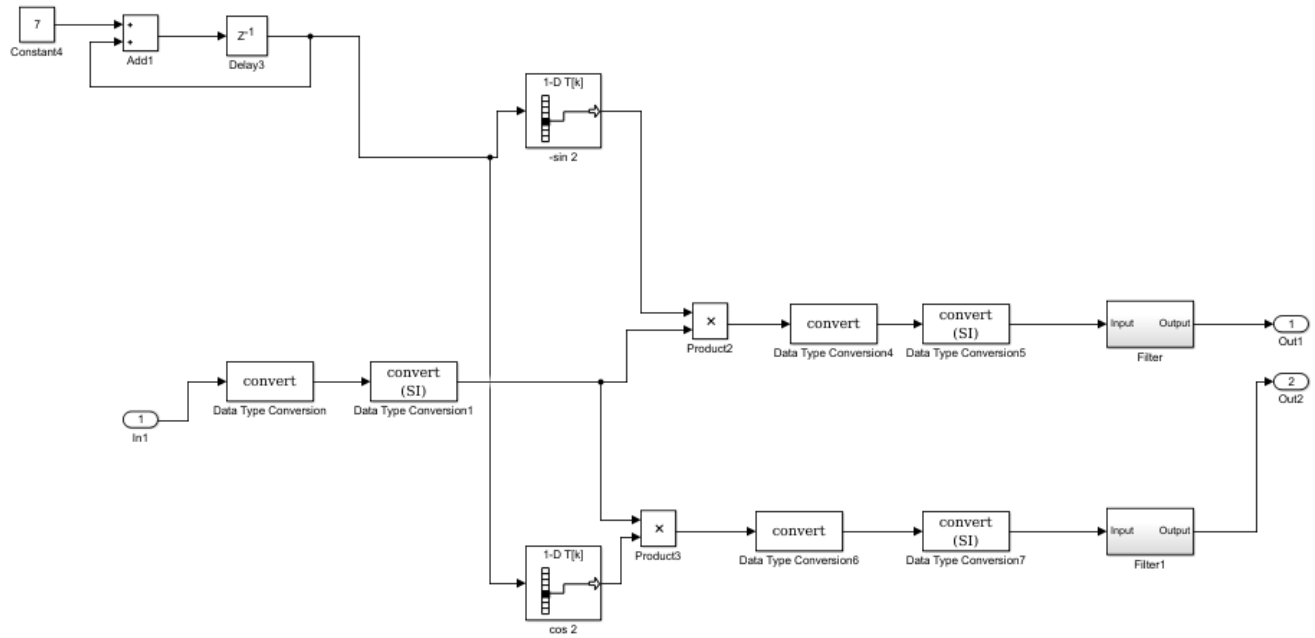

Рисунок 14 – Модель демодулятора в среде моделирования Matlab Simulink

Устройство, генерирующее фазу, состоит из блока Constant и линии задержки – блока Add и блока Delay. После часть сигнала из генератора фазы поступает на вход блока Direct Lookup Table (-sin 2), а другая часть на вход блока Direct Lookup Table (cos 2). Далее сигналы поступают на умножители Product и Product2, где генератор опорного колебания умножается на входной сигнал, произведение этих сигналов имеет целочисленный тип данных. После в фильтрах подавляется спектр, составленный на удвоенную частоту [5].

2.4 Декодирующее устройство

Декодер – устройство, которое преобразует информацию из одного внешнего вида в другой вид, применяемый в каком-нибудь устройстве.

На рисунке 15 изображена модель декодера в среде моделирования Matlab Simulink.

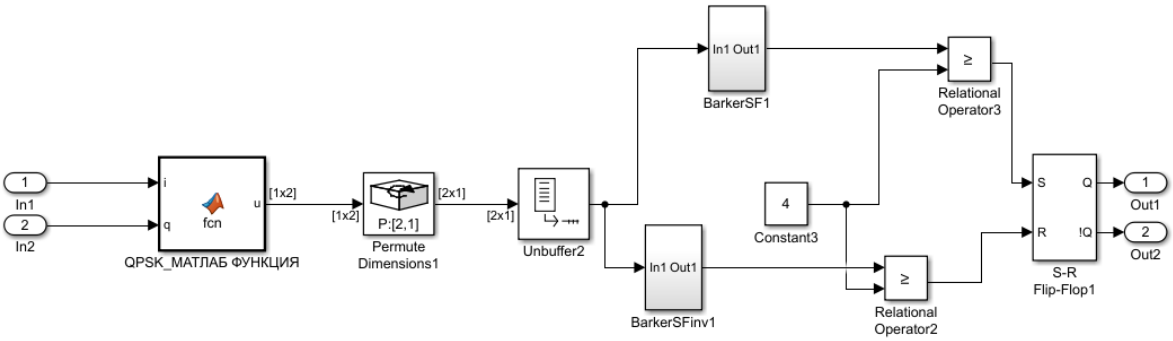

Рисунок 15 – Модель декодера в среде моделирования Matlab Simulink

На основе знака I и Q формируется два бита информации, далее в блоке Unbuffer2 формируется информационный поток, который поступает на согласованные фильтры для принятия решения о логическом уровне текущего знака 0 или 1. Далее сравнивается с порогом, в нашем случае порог равен 4, и поступает на вход RS триггера. Триггер – это электронное устройство, которое предназначается для записи и хранения информации, имеет два выхода: прямой и инверсный; и некоторое количество входов, в зависимости от выполняемой задачи [6].

3 VHDL описание модели приемника и передатчика системы цифровой связи с QPSK модуляцией

Данный раздел посвящен VHDL описанию модели приемника и передатчика системы цифровой связи с QPSK модуляцией, подробно описаны все функциональные составляющие модели, приведены значения портов и сигналов, а также их назначение и разрядность. Описание модели получено в среде программирования Quartus II, которую разработала компания Altera. Данная программа позволяет проектировать логику работы микросхем схемотехнически и на языках программирования AHDL, VHDL, Verilog и других. Среда программирования Altera Quartus II так же позволяет производить симуляцию проектов, программировать микросхемы и многое другое. Альтера на протяжении многих лет совершенствовала среду Quartus II и в настоящее время, то есть последняя версия этой программы, теперь называется Altera Quartus Prime [7].

3.1 VHDL описание кодирующего устройства

«Источник сигнала» выдает данные для передачи по каналу связи в цифровом виде. А в блоке «Кодер» происходит процесс эффективного преобразования данных в последовательность двоичных символов. В приложение А приведено VHDL описание источника сигнала. В приложение Б приведено VHDL описание кодирующего устройства. Для разработки VHDL кода устройства кодирования необходимо было реализовать блок «Barker Code» на простых логических операциях, так как стандартный блок MATLAB «Barker Code» не поддавался для генерации VHDL кода.

Основные особенности блока «Источник сигнала», описание назначений и разрядности портов приведены в таблице 2.

Таблица 2 – Описание портов блока «Источник сигнала»

Порт	Назначение	Разрядность
clk	Входной сигнал (синхронизация)	1
reset	Входной сигнал (сброс)	1
clk_enable	Входной сигнал (разрешение синхронизации)	1
ce_out	Выходной сигнал (разрешение синхронизации)	1

Кодирующее устройство состоит щ – блок «Источник сигнала», блок «Смена знака», «Коммутатор» и «Буфер». В коммутаторе первый и третий входные порты представляют собой порты данных, а второй входной порт - порт управления. Далее в блоке «Буфер» осуществляется кодирование следующим образом: разбивается битовый поток на четные и нечетные биты. И разбитый по паре бит информационный поток поступает на вход модулятора.

Основные особенности блока «Кодирующее устройство», описание назначений и разрядности портов приведены в таблице 3.

Таблица 3 – Описание портов блока «Кодирующее устройство»

Порт	Назначение	Разрядность
clk	Входной сигнал (синхронизация)	1
reset	Входной сигнал (сброс)	1
clk_enable	Входной сигнал (разрешение синхронизации)	1
ce_out	Выходной сигнал (разрешение синхронизации)	1

Основные особенности, описание назначений и разрядности портов блока «Буфер», входящего в состав кодирующего устройства приведены в таблице 4. А в приложение В приведено VHDL описание блока «Буфер».

Таблица 4 – Описание портов блока «Буфер»

Порт	Назначение	Разрядность
u	Отсчеты входного сигнала	8
y	Отсчеты выходного сигнала	2

3.2 VHDL описание модулятора

Модель цифрового модулятора состоит из блока «Генератор фазы», блока «Генератор огибающей», таблица для расчетов cos, таблица для расчетов sin, двух умножителей и сумматора. С выхода кодера данные поступают на передатчик цифровой системы связи, в котором выполняется цифровая модуляция – блок «Модулятор». В приложение Г приведено VHDL описание блока «Модулятор». Основные особенности, описание назначений и разрядности портов блока «Модулятор» приведены в таблице 5.

Таблица 5 – Описание портов блока «Модулятор»

Порт	Назначение	Разрядность
clk	Входной сигнал (синхронизация)	1
reset	Входной сигнал (сброс)	1
clk_enable	Входной сигнал (разрешение синхронизации)	1
ce_out	Выходной сигнал (разрешение синхронизации)	8
Out1	Отсчеты выходного сигнала	1

VHDL описание блока «Модулятор» включает в себя две компоненты - блок «Генератор фазы» (приложение Д) и «Генератор огибающей» (приложение Е). Основные особенности, описание назначений и разрядности портов блока «Генератор фазы», входящего в состав блока «Модулятор» приведены в таблице

6.

Таблица 6 – Описание портов блока «Генератор фазы»

Порт	Назначение	Разрядность
clk	Входной сигнал (синхронизация)	1
reset	Входной сигнал (сброс)	1
enb	Входной сигнал (разрешение синхронизации)	1
Out1	Отсчеты выходного сигнала генератора фазы	8

Основные особенности, описание назначений и разрядности портов блока «Генератор огибающей», входящего в состав блока «Модулятор» приведены в таблице 7.

Таблица 7 – Описание портов блока «Генератор огибающей»

Порт	Назначение	Разрядность
u	Отсчеты входного сигнала	8
i	Отсчеты cos	8
q	Отсчеты sin	8

3.3 VHDL описание демодулятора

Демодулятор предназначен для выделения информационного сигнала из модулированного высокочастотного колебания.

В приложение Ж приведено VHDL описание блока «Демодулятор». Основные особенности, описание назначений и разрядности портов блока «Демодулятор» приведены в таблице 8.

Таблица 8 – Описание портов блока «Демодулятор»

Порт	Назначение	Разрядность
clk	Входной сигнал (синхронизация)	1
reset	Входной сигнал (сброс)	1
clk_enable	Входной сигнал (разрешение синхронизации)	1
ce_out	Выходной сигнал (разрешение синхронизации)	8
Out1	Отсчеты выходного сигнала	1

3.4 VHDL описание декодирующего устройства

Декодер – устройство, преобразующее информацию из одного внешнего вида в другой вид, применяемый в каком-нибудь устройстве.

На основе знака I и Q формируется два бита информации, в блоке «Буфер» формируется информационный поток, который поступает на согласованные фильтры для принятия решения о логическом уровне текущего знака 0 или 1. Далее происходит сравнение с порогом и после этого информационный поток поступает на вход RS триггера. В приложение 3 приведено VHDL описание блока «Декодирующее устройство». Основные особенности, описание назначений и разрядности портов блока «Декодирующее устройство» приведены в таблице 9.

Таблица 9 – Описание портов блока «Декодирующее устройство»

Порт	Назначение	Разрядность
clk	Входной сигнал (синхронизация)	1
reset	Входной сигнал (сброс)	1
enb	Входной сигнал (разрешение синхронизации)	1
In1	Выходной сигнал (разрешение синхронизации)	16
Out1	Отсчеты выходного сигнала	32

Для того чтобы проанализировать модель приемника и передатчика системы цифровой связи с QPSK модуляцией и определить, какая производительность необходима тому или иному блоку в среде проектирования Altera Quartus II был произведен процесс компиляции модели, полученные результаты представлены в таблице 10. Данный анализ производился в семействе FPGA Stratix III фирмы Altera.

Таблица 10 – Описание требуемых ресурсов ПЛИС

Название блока	ALUTs	Dedicated Logic Registers	Blok Memory Bits	DSP Elements
Источник сигнала	4	3	0	0
Кодирующее устройство	12	3	0	0
Модулятор	75	8	0	2
Демодулятор	979	314	0	6
Декодирующее устройство	13	198	0	0
Итого	1083	526	0	8

ALUTs – это комбинационные арифметические логические ячейки, которые могут использоваться для реализации памяти, логики. Использование памяти ALUT 0 не означает, что ALUT не используются, это означает, что они не используются в качестве памяти.

Dedicated Logic Registers – количество задействованных триггеров.

Blok Memory Bits – количество ячеек памяти.

DSP Elements – количество аппаратных умножителей.

По результатам компиляции, полученной в среде программирования Quartus II, можно сделать вывод, что для реализации приемника и передатчика системы цифровой связи с QPSK модуляцией требуются следующие логические ресурсы ПЛИС: 1083 ALUTs, 8 DSP Elements, 526 Dedicated Logic Registers. Основные ресурсы используются для реализации фильтров.

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе решается проблема о наличие в свободном доступе VHDL кодов, которые необходимы для проектирования цифровых систем на программированных логических интегральных схемах (ПЛИС).

Практическая значимость данной работы заключается в разработке таких кодов на языке VHDL для реализации приемника и передатчика цифровой связи на базе ПЛИС.

В ходе решения поставленной проблемы, были выполнены задачи в полном объеме. Изучены основные теоретические сведения, проанализирована модель приемника и передатчика системы цифровой связи с QPSK модуляцией. Описана структурная схема модели приемника и передатчика системы цифровой связи. Разработаны и подробно описаны отдельные функциональные компоненты, входящих в состав приемника и передатчика.

Полученные результаты позволяют сделать вывод, что для реализации приемника и передатчика системы цифровой связи с QPSK модуляцией необходимы следующие логические ресурсы ПЛИС: 1083 ALUTs, 8 DSP Elements.

На следующем этапе предполагается дальнейшая работа в разработке, моделирование и отладке схемы синхронизации, также разработать и произвести отладку программного обеспечения для ПЛИС в составе макета на базе отладочной платы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Деев, В. В. Методы модуляции и кодирования в современных системах связи [Текст] / В. В. Деев. – Санкт-Петербург: Наука, 2007. – 266 с. : ил., табл.
2. Скляр , Б. Цифровая связь [Текст] / Б. Скляр. — М.: Вильямс, 2003. – 1104 с.: ил.
3. Черных, И.В. Simulink: Инструмент моделирования динамических систем [Электронный ресурс]/ И.В. Черных. – М., 2003. – 252 с.
4. Основы построения телекоммуникационных систем и сетей [Текст]: учебник для вузов / В. В. Крухмалев. – М.: Горячая линия – Телеком, 2004. – 510 с.: ил.
5. Худяков, В.Ф Моделирование источников вторичного электропитания в среде MATLAB 7.x [Текст]: учебное пособие/ В.Ф. Худяков. – СПб.: ГУАП, 2008. – 332 с.
6. Сергиенко, А.Б. Цифровая обработка сигналов [Текст]: учеб. пособие/ А.Б. Сергиенко. – СПб: Питер, 2002. – 606 с.
7. Системы автоматизированного проектирования фирмы Altera MAX+plus II и Quartus II. Краткое описание и самоучитель [Текст]/ Комолов Д.А. [и др.]. – М.: РадиоСофт, 2002. – 352 с.: ил.
8. Джакония, В.Е. Телевидение учебник для вузов / В.Е. Джакония, А.А. Гоголь, Я.В. Друзин; под ред. В.Е. Джакония. – М.: Горячая линия – Телеком, 2007. – 616 с.: ил.
9. Дьяконов, В. П. MATLAB. Полный самоучитель [Текст]/ В.П. Дьяконов. – М.: ДМК Пресс, 2012. – 768 с.: ил.
10. Ефимов С.Н., Цифровая обработка видеоинформации [Текст] / С.Н. Ефимов. – М.: Сайнс-Пресс, 2007. – 272 с.
11. Логические ИС КР1533, КР1554 [Текст]: справочник/ Петровский И.И. [и др.]. – М.: Бином, 1998. – 254 с.

12. Птачек, М. Цифровое телевидение: Теория и техника [Текст] / М. Птачек. – М.: Радио и связь, 1990. – 528 с.
13. Воробьев, М.С. Основы цифрового телевидения [Текст]: учебное пособие/ М.С. Воробьев. – Челябинск: Издательский центр: ЮУрГУ, 2002. – 74 с..
14. Смирнов А.В. Основы цифрового телевидения [Текст]: учебное пособие / А. В. Смирнов. – М.: Горячая Линия – Телеком, 2001. – 224 с.
15. Цифровая фазовая модуляция [Текст]: монография / В.И. Журавлев, А.Н. Руднев. – М.: Радиотехника. – 2012. – 246 с.
16. Угрюмов, Е. П. Цифровая схемотехника [Текст]: учеб. пособие для вузов / Е. П. Угрюмов. – СПб.: БХВ-Петербург, 2010. – 816 с.: ил.
17. Уэйкерли, Дж. Ф. Проектирование цифровых устройств [Текст]: в 2 т. / Дж. Ф. Уэйкерли. – М.: Постмаркет, 2002. – 1080 с.
18. Шило, В.Л. Популярные цифровые микросхемы [Текст]: справочник / Шило В.Л. – М.: Радио и связь, 1987. – 350 с.

ПРИЛОЖЕНИЕ А

Листингпрограммы «Источник сигнала»

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

ENTITY istoc_signala IS
PORT( clk                               :   IN    std_logic;
        reset                             :   IN    std_logic;
        clk_enable                        :   IN    std_logic;
        ce_out                            :   OUT   std_logic;
        Out1                              :   OUT   std_logic_vector(7 DOWNTO 0)
        );
END istoc_signala;


ARCHITECTURE rtl OF istoc_signala IS

  COMPONENT not_ascii_QPSK1
PORT( x                           :   IN    std_logic_vector(7 DOWNTO 0);
          y                         :   OUT   std_logic_vector(7 DOWNTO 0)
          );
  END COMPONENT;

  COMPONENT not_ascii_QPSK2
PORT( x                               :   IN    std_logic_vector(7 DOWNTO 0);
          y                             :   OUT   std_logic_vector(7 DOWNTO 0)
          );
  END COMPONENT;


  FOR ALL : not_ascii_QPSK1
    USE ENTITY work.not_ascii_QPSK1(rtl);

  FOR ALL : not_ascii_QPSK2
    USE ENTITY work.not_ascii_QPSK2(rtl);


  SIGNAL enb                            : std_logic;
  SIGNAL Delay_out1                     : signed(7 DOWNTO 0);
  SIGNAL y                              : std_logic_vector(7 DOWNTO 0);
  SIGNAL y_signed                       : signed(7 DOWNTO 0);
  SIGNAL y_1                            : std_logic_vector(7 DOWNTO 0);

BEGIN
  u_not_ascii_QPSK1 : not_ascii_QPSK1
    PORT MAP( x => std_logic_vector(Delay_out1),
              y => y
              );

  u_not_ascii_QPSK2 : not_ascii_QPSK2
    PORT MAP( x => std_logic_vector(Delay_out1),
              y => y_1
```

```
              );

  enb <= clk_enable;

  y_signed <= signed(y);

  Delay_process : PROCESS (clk, reset)
  BEGIN
    IF reset = '1' THEN
      Delay_out1 <= to_signed(16#00#, 8);
    ELSIF clk'EVENT AND clk = '1' THEN
      IF enb = '1' THEN
        Delay_out1 <= y_signed;
      END IF;
    END IF;
  END PROCESS Delay_process;


  ce_out <= clk_enable;

  Out1 <= y_1;

END rtl;
```

ФУНКЦИЯ 1

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

ENTITY not_ascii_QPSK1 IS
PORT( x                                 :   IN    std_logic_vector(7 DOWNTO 0);
        y                                 :   OUT   std_logic_vector(7 DOWNTO 0)
        );
END not_ascii_QPSK1;


ARCHITECTURE rtl OF not_ascii_QPSK1 IS

  SIGNAL x_signed                         : signed(7 DOWNTO 0);
  SIGNAL y_tmp                            : signed(7 DOWNTO 0);

BEGIN
  x_signed <= signed(x);

  not_ascii_QPSK1_1_output : PROCESS (x_signed)
  BEGIN
    IF x_signed = to_signed(16#00000000#, 8) THEN
      y_tmp <= to_signed(16#01#, 8);
    ELSIF x_signed = to_signed(16#00000001#, 8) THEN
      y_tmp <= to_signed(16#02#, 8);
    ELSIF x_signed = to_signed(16#00000002#, 8) THEN
      y_tmp <= to_signed(16#03#, 8);
    ELSIF x_signed = to_signed(16#00000003#, 8) THEN
      y_tmp <= to_signed(16#04#, 8);
    ELSIF x_signed = to_signed(16#00000004#, 8) THEN
      y_tmp <= to_signed(16#05#, 8);
    ELSIF x_signed = to_signed(16#00000005#, 8) THEN
      y_tmp <= to_signed(16#06#, 8);
    ELSE
```

```
      y_tmp <= to_signed(16#00#, 8);
    END IF;

  END PROCESS not_ascii_QPSK1_1_output;

  y <= std_logic_vector(y_tmp);

END rtl;
```

ФУНКЦИЯ 2

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

ENTITY not_ascii_QPSK2 IS
PORT( x                                 :   IN    std_logic_vector(7 DOWNTO 0);
       y                                  :   OUT   std_logic_vector(7 DOWNTO 0)
       );
END not_ascii_QPSK2;


ARCHITECTURE rtl OF not_ascii_QPSK2 IS

  SIGNAL x_signed                         : signed(7 DOWNTO 0);
  SIGNAL y_tmp                            : signed(7 DOWNTO 0);

BEGIN
  x_signed <= signed(x);

  not_ascii_QPSK2_1_output : PROCESS (x_signed)
  BEGIN
    IF x_signed = to_signed(16#00000000#, 8) THEN
      y_tmp <= X"FF";
    ELSIF x_signed = to_signed(16#00000001#, 8) THEN
      y_tmp <= X"FF";
    ELSIF x_signed = to_signed(16#00000002#, 8) THEN
      y_tmp <= X"FF";
    ELSIF x_signed = to_signed(16#00000003#, 8) THEN
      y_tmp <= to_signed(16#01#, 8);
    ELSIF x_signed = to_signed(16#00000004#, 8) THEN
      y_tmp <= to_signed(16#01#, 8);
    ELSIF x_signed = to_signed(16#00000005#, 8) THEN
      y_tmp <= X"FF";
    ELSIF x_signed = to_signed(16#00000006#, 8) THEN
      y_tmp <= to_signed(16#01#, 8);
    ELSE
      y_tmp <= X"FF";
    END IF;

  END PROCESS not_ascii_QPSK2_1_output;

  y <= std_logic_vector(y_tmp);

END rtl;
```

ПРИЛОЖЕНИЕ Б

Листинг программы «Кодирующее устройство»

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

ENTITY not_ascii_2 IS
PORT( clk                               :   IN    std_logic;
        reset                             :   IN    std_logic;
        enb                               :   IN    std_logic;
        Out1                              :   OUT   std_logic_vector(7 DOWNTO 0)
-- int8
        );
END not_ascii_2;


ARCHITECTURE rtl OF not_ascii_2 IS

  -- Component Declarations
  COMPONENT QPSK_demod2
PORT( x                                 :   IN    std_logic_vector(7 DOWNTO 0);  --
int8
          y                               :   OUT   std_logic_vector(7 DOWNTO 0)
-- int8
          );
  END COMPONENT;

  COMPONENT QPSK_demod3
PORT( x                                 :   IN    std_logic_vector(7 DOWNTO 0);  --
int8
          y                               :   OUT   std_logic_vector(7 DOWNTO 0)
-- int8
          );
  END COMPONENT;

  -- Component Configuration Statements
  FOR ALL : QPSK_demod2
    USE ENTITY work.QPSK_demod2(rtl);

  FOR ALL : QPSK_demod3
    USE ENTITY work.QPSK_demod3(rtl);

  -- Signals
  SIGNAL Delay_out1                       : signed(7 DOWNTO 0);  -- int8
  SIGNAL y                                : std_logic_vector(7 DOWNTO 0);  --
ufix8
  SIGNAL y_signed                         : signed(7 DOWNTO 0);  -- int8
  SIGNAL y_1                              : std_logic_vector(7 DOWNTO 0);  --
ufix8

BEGIN
  u_QPSK_demod2 : QPSK_demod2
    PORT MAP( x => std_logic_vector(Delay_out1),  -- int8
              y =>y  -- int8
              );
```

```
  u_QPSK_demod3 : QPSK_demod3
    PORT MAP( x => std_logic_vector(Delay_out1),  -- int8
              y => y_1  -- int8
              );

  y_signed <= signed(y);

  Delay_process : PROCESS (clk, reset)
  BEGIN
    IF reset = '1' THEN
      Delay_out1 <= to_signed(16#00#, 8);
    ELSIF clk'EVENT AND clk = '1' THEN
      IF enb = '1' THEN
        Delay_out1 <= y_signed;
      END IF;
    END IF;
  END PROCESS Delay_process;


  Out1 <= y_1;

END rtl;



LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
USE work.coder_pkg.ALL;

ENTITY coder IS
PORT( clk                               :   IN    std_logic;
        reset                             :   IN    std_logic;
        clk_enable                        :   IN    std_logic;
        In1                               :   IN    std_logic_vector(7 DOWNTO 0);
        ce_out                            :   OUT   std_logic;
        Out1                              :   OUT   vector_of_std_logic_vector8(0
TO 1)
        );
END coder;


ARCHITECTURE rtl OF coder IS

  COMPONENT not_ascii_2
PORT( clk                             :   IN    std_logic;
          reset                           :   IN    std_logic;
          enb                             :   IN    std_logic;
          Out1                            :   OUT   std_logic_vector(7 DOWNTO 0)
          );
  END COMPONENT;

  COMPONENT MATLAB_Function1
PORT( u                               :   IN    std_logic_vector(7 DOWNTO 0);
          y                               :   OUT   vector_of_std_logic_vector8(0
TO 1)
          );
  END COMPONENT;


  FOR ALL : not_ascii_2
```

```
    USE ENTITY work.not_ascii_2(rtl);

  FOR ALL : MATLAB_Function1
    USE ENTITY work.MATLAB_Function1(rtl);

  SIGNAL y                                : std_logic_vector(7 DOWNTO 0);
  SIGNAL y_signed                         : signed(7 DOWNTO 0);
  SIGNAL switch_compare_1                 : std_logic;
  SIGNAL In1_signed                       : signed(7 DOWNTO 0);
  SIGNAL Unary_Minus1_in0                 : signed(8 DOWNTO 0);
  SIGNAL Unary_Minus1_out1                : signed(7 DOWNTO 0);
  SIGNAL Switch2_out1                     : signed(7 DOWNTO 0);
  SIGNAL y_1                              : vector_of_std_logic_vector8(0 TO 1);

BEGIN
  u_not_ascii_2 : not_ascii_2
    PORT MAP( clk => clk,
              reset => reset,
              enb => clk_enable,
              Out1 => y
              );

  u_MATLAB_Function1 : MATLAB_Function1
    PORT MAP( u => std_logic_vector(Switch2_out1),
              y => y_1
              );

  y_signed <= signed(y);


  switch_compare_1 <= '1' WHEN y_signed > to_signed(16#00#, 8) ELSE
      '0';

  In1_signed <= signed(In1);

  Unary_Minus1_in0 <=  - (resize(In1_signed, 9));
  Unary_Minus1_out1 <= Unary_Minus1_in0(7 DOWNTO 0);


  Switch2_out1 <= Unary_Minus1_out1 WHEN switch_compare_1 = '0' ELSE
      In1_signed;

  ce_out <= clk_enable;

  Out1 <= y_1;

END rtl;
```

СИГНАЛЫ

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

PACKAGE coder_pkg IS
  TYPE vector_of_std_logic_vector8 IS ARRAY (NATURAL RANGE <>) OF
std_logic_vector(7 DOWNTO 0);
  TYPE vector_of_signed8 IS ARRAY (NATURAL RANGE <>) OF signed(7 DOWNTO 0);
ENDcoder_pkg;
```

ПРИЛОЖЕНИЕ В

Листинг программы «Буфер»

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
USE work.coder_pkg.ALL;

ENTITY MATLAB_Function1 IS
PORT( u                                 :   IN    std_logic_vector(7 DOWNTO 0);
        y                                 :   OUT   vector_of_std_logic_vector8(0
TO 1)
        );
END MATLAB_Function1;


ARCHITECTURE rtl OF MATLAB_Function1 IS

  SIGNAL u_signed                         : signed(7 DOWNTO 0);
  SIGNAL y_tmp                            : vector_of_signed8(0 TO 1);

BEGIN
  u_signed <= signed(u);

  MATLAB_Function1_1_output : PROCESS (u_signed)
  BEGIN
    y_tmp <= (OTHERS => to_signed(16#00#, 8));
    y_tmp(0) <= u_signed;
    y_tmp(1) <= u_signed;
  END PROCESS MATLAB_Function1_1_output;


  outputgen: FOR k IN 0 TO 1 GENERATE
    y(k) <= std_logic_vector(y_tmp(k));
END GENERATE;

END rtl;
```

ПРИЛОЖЕНИЕГ

Листинг программы «Модулятор»

```
LIBRARYIEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
USE work.mod_rsvd_pkg.ALL;

ENTITY mod_rsvd IS
PORT( clk                               :   IN    std_logic;
        reset                           :   IN    std_logic;
        clk_enable                      :   IN    std_logic;
        In1                             :   IN    std_logic_vector(7 DOWNTO 0);
        ce_out                          :   OUT   std_logic;
        Out1                            :   OUT   std_logic_vector(15 DOWNTO 0)
        );
END mod_rsvd;


ARCHITECTURE rtl OF mod_rsvd IS

  COMPONENT generatorphazi
PORT( clk                             :   IN    std_logic;
          reset                       :   IN    std_logic;
          enb                         :   IN    std_logic;
          Out1                        :   OUT   std_logic_vector(7 DOWNTO 0)
          );
  END COMPONENT;

  COMPONENT MATLAB_Function
PORT( u                               :   IN    std_logic_vector(7 DOWNTO 0);
          i                           :   OUT   std_logic_vector(7 DOWNTO 0);
          q                           :   OUT   std_logic_vector(7 DOWNTO 0)
          );
  END COMPONENT;

  FOR ALL : generatorphazi
    USE ENTITY work.generatorphazi(rtl);

  FOR ALL : MATLAB_Function
    USE ENTITY work.MATLAB_Function(rtl);

  CONSTANT nc                             : vector_of_signed8(0 TO 255) :=
    (to_signed(16#7F#, 8), to_signed(16#7F#, 8), to_signed(16#7F#, 8),
to_signed(16#7F#, 8), to_signed(16#7E#, 8),
     to_signed(16#7E#, 8), to_signed(16#7E#, 8), to_signed(16#7D#, 8),
to_signed(16#7D#, 8), to_signed(16#7C#, 8),
     to_signed(16#7B#, 8), to_signed(16#7A#, 8), to_signed(16#7A#, 8),
to_signed(16#79#, 8), to_signed(16#78#, 8),
     to_signed(16#76#, 8), to_signed(16#75#, 8), to_signed(16#74#, 8),
to_signed(16#73#, 8), to_signed(16#71#, 8),
     to_signed(16#70#, 8), to_signed(16#6F#, 8), to_signed(16#6D#, 8),
to_signed(16#6B#, 8), to_signed(16#6A#, 8),
     to_signed(16#68#, 8), to_signed(16#66#, 8), to_signed(16#64#, 8),
to_signed(16#62#, 8), to_signed(16#60#, 8),
     to_signed(16#5E#, 8), to_signed(16#5C#, 8), to_signed(16#5A#, 8),
to_signed(16#58#, 8), to_signed(16#55#, 8),
     to_signed(16#53#, 8), to_signed(16#51#, 8), to_signed(16#4E#, 8),
```

```
to_signed(16#4C#, 8), to_signed(16#49#, 8),
     to_signed(16#47#, 8), to_signed(16#44#, 8), to_signed(16#41#, 8),
to_signed(16#3F#, 8), to_signed(16#3C#, 8),
     to_signed(16#39#, 8), to_signed(16#36#, 8), to_signed(16#33#, 8),
to_signed(16#31#, 8), to_signed(16#2E#, 8),
     to_signed(16#2B#, 8), to_signed(16#28#, 8), to_signed(16#25#, 8),
to_signed(16#22#, 8), to_signed(16#1F#, 8),
     to_signed(16#1C#, 8), to_signed(16#19#, 8), to_signed(16#16#, 8),
to_signed(16#13#, 8), to_signed(16#10#, 8),
     to_signed(16#0C#, 8), to_signed(16#09#, 8), to_signed(16#06#, 8),
to_signed(16#03#, 8), to_signed(16#00#, 8),
     to_signed(-16#03#, 8), to_signed(-16#06#, 8), to_signed(-16#09#, 8),
to_signed(-16#0C#, 8),
     to_signed(-16#10#, 8), to_signed(-16#13#, 8), to_signed(-16#16#, 8),
to_signed(-16#19#, 8),
     to_signed(-16#1C#, 8), to_signed(-16#1F#, 8), to_signed(-16#22#, 8),
to_signed(-16#25#, 8),
     to_signed(-16#28#, 8), to_signed(-16#2B#, 8), to_signed(-16#2E#, 8),
to_signed(-16#31#, 8),
     to_signed(-16#33#, 8), to_signed(-16#36#, 8), to_signed(-16#39#, 8),
to_signed(-16#3C#, 8),
     to_signed(-16#3F#, 8), to_signed(-16#41#, 8), to_signed(-16#44#, 8),
to_signed(-16#47#, 8),
     to_signed(-16#49#, 8), to_signed(-16#4C#, 8), to_signed(-16#4E#, 8),
to_signed(-16#51#, 8),
     to_signed(-16#53#, 8), to_signed(-16#55#, 8), to_signed(-16#58#, 8),
to_signed(-16#5A#, 8),
     to_signed(-16#5C#, 8), to_signed(-16#5E#, 8), to_signed(-16#60#, 8),
to_signed(-16#62#, 8),
     to_signed(-16#64#, 8), to_signed(-16#66#, 8), to_signed(-16#68#, 8),
to_signed(-16#6A#, 8),
     to_signed(-16#6B#, 8), to_signed(-16#6D#, 8), to_signed(-16#6F#, 8),
to_signed(-16#70#, 8),
     to_signed(-16#71#, 8), to_signed(-16#73#, 8), to_signed(-16#74#, 8),
to_signed(-16#75#, 8),
     to_signed(-16#76#, 8), to_signed(-16#78#, 8), to_signed(-16#79#, 8),
to_signed(-16#7A#, 8),
     to_signed(-16#7A#, 8), to_signed(-16#7B#, 8), to_signed(-16#7C#, 8),
to_signed(-16#7D#, 8),
     to_signed(-16#7D#, 8), to_signed(-16#7E#, 8), to_signed(-16#7E#, 8),
to_signed(-16#7E#, 8),
     to_signed(-16#7F#, 8), to_signed(-16#7F#, 8), to_signed(-16#7F#, 8),
to_signed(-16#7F#, 8),
     to_signed(-16#7F#, 8), to_signed(-16#7F#, 8), to_signed(-16#7F#, 8),
to_signed(-16#7E#, 8),
     to_signed(-16#7E#, 8), to_signed(-16#7E#, 8), to_signed(-16#7D#, 8),
to_signed(-16#7D#, 8),
     to_signed(-16#7C#, 8), to_signed(-16#7B#, 8), to_signed(-16#7A#, 8),
to_signed(-16#7A#, 8),
     to_signed(-16#79#, 8), to_signed(-16#78#, 8), to_signed(-16#76#, 8),
to_signed(-16#75#, 8),
     to_signed(-16#74#, 8), to_signed(-16#73#, 8), to_signed(-16#71#, 8),
to_signed(-16#70#, 8),
     to_signed(-16#6F#, 8), to_signed(-16#6D#, 8), to_signed(-16#6B#, 8),
to_signed(-16#6A#, 8),
     to_signed(-16#68#, 8), to_signed(-16#66#, 8), to_signed(-16#64#, 8),
to_signed(-16#62#, 8),
     to_signed(-16#60#, 8), to_signed(-16#5E#, 8), to_signed(-16#5C#, 8),
to_signed(-16#5A#, 8),
     to_signed(-16#58#, 8), to_signed(-16#55#, 8), to_signed(-16#53#, 8),
to_signed(-16#51#, 8),
     to_signed(-16#4E#, 8), to_signed(-16#4C#, 8), to_signed(-16#49#, 8),
to_signed(-16#47#, 8),
```

```
     to_signed(-16#44#, 8), to_signed(-16#41#, 8), to_signed(-16#3F#, 8),
to_signed(-16#3C#, 8),
     to_signed(-16#39#, 8), to_signed(-16#36#, 8), to_signed(-16#33#, 8),
to_signed(-16#31#, 8),
     to_signed(-16#2E#, 8), to_signed(-16#2B#, 8), to_signed(-16#28#, 8),
to_signed(-16#25#, 8),
     to_signed(-16#22#, 8), to_signed(-16#1F#, 8), to_signed(-16#1C#, 8),
to_signed(-16#19#, 8),
     to_signed(-16#16#, 8), to_signed(-16#13#, 8), to_signed(-16#10#, 8),
to_signed(-16#0C#, 8),
     to_signed(-16#09#, 8), to_signed(-16#06#, 8), to_signed(-16#03#, 8),
to_signed(16#00#, 8),
     to_signed(16#03#, 8), to_signed(16#06#, 8), to_signed(16#09#, 8),
to_signed(16#0C#, 8), to_signed(16#10#, 8),
     to_signed(16#13#, 8), to_signed(16#16#, 8), to_signed(16#19#, 8),
to_signed(16#1C#, 8), to_signed(16#1F#, 8),
     to_signed(16#22#, 8), to_signed(16#25#, 8), to_signed(16#28#, 8),
to_signed(16#2B#, 8), to_signed(16#2E#, 8),
     to_signed(16#31#, 8), to_signed(16#33#, 8), to_signed(16#36#, 8),
to_signed(16#39#, 8), to_signed(16#3C#, 8),
     to_signed(16#3F#, 8), to_signed(16#41#, 8), to_signed(16#44#, 8),
to_signed(16#47#, 8), to_signed(16#49#, 8),
     to_signed(16#4C#, 8), to_signed(16#4E#, 8), to_signed(16#51#, 8),
to_signed(16#53#, 8), to_signed(16#55#, 8),
     to_signed(16#58#, 8), to_signed(16#5A#, 8), to_signed(16#5C#, 8),
to_signed(16#5E#, 8), to_signed(16#60#, 8),
     to_signed(16#62#, 8), to_signed(16#64#, 8), to_signed(16#66#, 8),
to_signed(16#68#, 8), to_signed(16#6A#, 8),
     to_signed(16#6B#, 8), to_signed(16#6D#, 8), to_signed(16#6F#, 8),
to_signed(16#70#, 8), to_signed(16#71#, 8),
     to_signed(16#73#, 8), to_signed(16#74#, 8), to_signed(16#75#, 8),
to_signed(16#76#, 8), to_signed(16#78#, 8),
     to_signed(16#79#, 8), to_signed(16#7A#, 8), to_signed(16#7A#, 8),
to_signed(16#7B#, 8), to_signed(16#7C#, 8),
     to_signed(16#7D#, 8), to_signed(16#7D#, 8), to_signed(16#7E#, 8),
to_signed(16#7E#, 8), to_signed(16#7E#, 8),
     to_signed(16#7F#, 8), to_signed(16#7F#, 8), to_signed(16#7F#, 8));
  CONSTANT nc_2                           : vector_of_signed8(0 TO 255) :=
    (to_signed(16#00#, 8), to_signed(16#03#, 8), to_signed(16#06#, 8),
to_signed(16#09#, 8), to_signed(16#0C#, 8),
     to_signed(16#10#, 8), to_signed(16#13#, 8), to_signed(16#16#, 8),
to_signed(16#19#, 8), to_signed(16#1C#, 8),
     to_signed(16#1F#, 8), to_signed(16#22#, 8), to_signed(16#25#, 8),
to_signed(16#28#, 8), to_signed(16#2B#, 8),
     to_signed(16#2E#, 8), to_signed(16#31#, 8), to_signed(16#33#, 8),
to_signed(16#36#, 8), to_signed(16#39#, 8),
     to_signed(16#3C#, 8), to_signed(16#3F#, 8), to_signed(16#41#, 8),
to_signed(16#44#, 8), to_signed(16#47#, 8),
     to_signed(16#49#, 8), to_signed(16#4C#, 8), to_signed(16#4E#, 8),
to_signed(16#51#, 8), to_signed(16#53#, 8),
     to_signed(16#55#, 8), to_signed(16#58#, 8), to_signed(16#5A#, 8),
to_signed(16#5C#, 8), to_signed(16#5E#, 8),
     to_signed(16#60#, 8), to_signed(16#62#, 8), to_signed(16#64#, 8),
to_signed(16#66#, 8), to_signed(16#68#, 8),
     to_signed(16#6A#, 8), to_signed(16#6B#, 8), to_signed(16#6D#, 8),
to_signed(16#6F#, 8), to_signed(16#70#, 8),
     to_signed(16#71#, 8), to_signed(16#73#, 8), to_signed(16#74#, 8),
to_signed(16#75#, 8), to_signed(16#76#, 8),
     to_signed(16#78#, 8), to_signed(16#79#, 8), to_signed(16#7A#, 8),
to_signed(16#7A#, 8), to_signed(16#7B#, 8),
     to_signed(16#7C#, 8), to_signed(16#7D#, 8), to_signed(16#7D#, 8),
to_signed(16#7E#, 8), to_signed(16#7E#, 8),
     to_signed(16#7E#, 8), to_signed(16#7F#, 8), to_signed(16#7F#, 8),
```

```
to_signed(16#7F#, 8), to_signed(16#7F#, 8),
     to_signed(16#7F#, 8), to_signed(16#7F#, 8), to_signed(16#7F#, 8),
to_signed(16#7E#, 8), to_signed(16#7E#, 8),
     to_signed(16#7E#, 8), to_signed(16#7D#, 8), to_signed(16#7D#, 8),
to_signed(16#7C#, 8), to_signed(16#7B#, 8),
     to_signed(16#7A#, 8), to_signed(16#7A#, 8), to_signed(16#79#, 8),
to_signed(16#78#, 8), to_signed(16#76#, 8),
     to_signed(16#75#, 8), to_signed(16#74#, 8), to_signed(16#73#, 8),
to_signed(16#71#, 8), to_signed(16#70#, 8),
     to_signed(16#6F#, 8), to_signed(16#6D#, 8), to_signed(16#6B#, 8),
to_signed(16#6A#, 8), to_signed(16#68#, 8),
     to_signed(16#66#, 8), to_signed(16#64#, 8), to_signed(16#62#, 8),
to_signed(16#60#, 8), to_signed(16#5E#, 8),
     to_signed(16#5C#, 8), to_signed(16#5A#, 8), to_signed(16#58#, 8),
to_signed(16#55#, 8), to_signed(16#53#, 8),
     to_signed(16#51#, 8), to_signed(16#4E#, 8), to_signed(16#4C#, 8),
to_signed(16#49#, 8), to_signed(16#47#, 8),
     to_signed(16#44#, 8), to_signed(16#41#, 8), to_signed(16#3F#, 8),
to_signed(16#3C#, 8), to_signed(16#39#, 8),
     to_signed(16#36#, 8), to_signed(16#33#, 8), to_signed(16#31#, 8),
to_signed(16#2E#, 8), to_signed(16#2B#, 8),
     to_signed(16#28#, 8), to_signed(16#25#, 8), to_signed(16#22#, 8),
to_signed(16#1F#, 8), to_signed(16#1C#, 8),
     to_signed(16#19#, 8), to_signed(16#16#, 8), to_signed(16#13#, 8),
to_signed(16#10#, 8), to_signed(16#0C#, 8),
     to_signed(16#09#, 8), to_signed(16#06#, 8), to_signed(16#03#, 8),
to_signed(16#00#, 8),
     to_signed(-16#03#, 8), to_signed(-16#06#, 8), to_signed(-16#09#, 8),
to_signed(-16#0C#, 8),
     to_signed(-16#10#, 8), to_signed(-16#13#, 8), to_signed(-16#16#, 8),
to_signed(-16#19#, 8),
     to_signed(-16#1C#, 8), to_signed(-16#1F#, 8), to_signed(-16#22#, 8),
to_signed(-16#25#, 8),
     to_signed(-16#28#, 8), to_signed(-16#2B#, 8), to_signed(-16#2E#, 8),
to_signed(-16#31#, 8),
     to_signed(-16#33#, 8), to_signed(-16#36#, 8), to_signed(-16#39#, 8),
to_signed(-16#3C#, 8),
     to_signed(-16#3F#, 8), to_signed(-16#41#, 8), to_signed(-16#44#, 8),
to_signed(-16#47#, 8),
     to_signed(-16#49#, 8), to_signed(-16#4C#, 8), to_signed(-16#4E#, 8),
to_signed(-16#51#, 8),
     to_signed(-16#53#, 8), to_signed(-16#55#, 8), to_signed(-16#58#, 8),
to_signed(-16#5A#, 8),
     to_signed(-16#5C#, 8), to_signed(-16#5E#, 8), to_signed(-16#60#, 8),
to_signed(-16#62#, 8),
     to_signed(-16#64#, 8), to_signed(-16#66#, 8), to_signed(-16#68#, 8),
to_signed(-16#6A#, 8),
     to_signed(-16#6B#, 8), to_signed(-16#6D#, 8), to_signed(-16#6F#, 8),
to_signed(-16#70#, 8),
     to_signed(-16#71#, 8), to_signed(-16#73#, 8), to_signed(-16#74#, 8),
to_signed(-16#75#, 8),
     to_signed(-16#76#, 8), to_signed(-16#78#, 8), to_signed(-16#79#, 8),
to_signed(-16#7A#, 8),
     to_signed(-16#7A#, 8), to_signed(-16#7B#, 8), to_signed(-16#7C#, 8),
to_signed(-16#7D#, 8),
     to_signed(-16#7D#, 8), to_signed(-16#7E#, 8), to_signed(-16#7E#, 8),
to_signed(-16#7E#, 8),
     to_signed(-16#7F#, 8), to_signed(-16#7F#, 8), to_signed(-16#7F#, 8),
to_signed(-16#7F#, 8),
     to_signed(-16#7F#, 8), to_signed(-16#7F#, 8), to_signed(-16#7F#, 8),
to_signed(-16#7E#, 8),
     to_signed(-16#7E#, 8), to_signed(-16#7E#, 8), to_signed(-16#7D#, 8),
to_signed(-16#7D#, 8),
```

```
     to_signed(-16#7C#, 8), to_signed(-16#7B#, 8), to_signed(-16#7A#, 8),
to_signed(-16#7A#, 8),
     to_signed(-16#79#, 8), to_signed(-16#78#, 8), to_signed(-16#76#, 8),
to_signed(-16#75#, 8),
     to_signed(-16#74#, 8), to_signed(-16#73#, 8), to_signed(-16#71#, 8),
to_signed(-16#70#, 8),
     to_signed(-16#6F#, 8), to_signed(-16#6D#, 8), to_signed(-16#6B#, 8),
to_signed(-16#6A#, 8),
     to_signed(-16#68#, 8), to_signed(-16#66#, 8), to_signed(-16#64#, 8),
to_signed(-16#62#, 8),
     to_signed(-16#60#, 8), to_signed(-16#5E#, 8), to_signed(-16#5C#, 8),
to_signed(-16#5A#, 8),
     to_signed(-16#58#, 8), to_signed(-16#55#, 8), to_signed(-16#53#, 8),
to_signed(-16#51#, 8),
     to_signed(-16#4E#, 8), to_signed(-16#4C#, 8), to_signed(-16#49#, 8),
to_signed(-16#47#, 8),
     to_signed(-16#44#, 8), to_signed(-16#41#, 8), to_signed(-16#3F#, 8),
to_signed(-16#3C#, 8),
     to_signed(-16#39#, 8), to_signed(-16#36#, 8), to_signed(-16#33#, 8),
to_signed(-16#31#, 8),
     to_signed(-16#2E#, 8), to_signed(-16#2B#, 8), to_signed(-16#28#, 8),
to_signed(-16#25#, 8),
     to_signed(-16#22#, 8), to_signed(-16#1F#, 8), to_signed(-16#1C#, 8),
to_signed(-16#19#, 8),
     to_signed(-16#16#, 8), to_signed(-16#13#, 8), to_signed(-16#10#, 8),
to_signed(-16#0C#, 8),
     to_signed(-16#09#, 8), to_signed(-16#06#, 8), to_signed(-16#03#, 8));


  SIGNAL generatorphazi_out1              : std_logic_vector(7 DOWNTO 0);
  SIGNAL generatorphazi_out1_unsigned     : unsigned(7 DOWNTO 0);
  SIGNAL cos_1_out1                       : signed(7 DOWNTO 0);
  SIGNAL i                                : std_logic_vector(7 DOWNTO 0);
  SIGNAL q                                : std_logic_vector(7 DOWNTO 0);
  SIGNAL sin_1_out1                       : signed(7 DOWNTO 0);
  SIGNAL i_signed                         : signed(7 DOWNTO 0);
  SIGNAL Product_out1                     : signed(15 DOWNTO 0);
  SIGNAL q_signed                         : signed(7 DOWNTO 0);
  SIGNAL Product1_out1                    : signed(15 DOWNTO 0);
  SIGNAL Add2_out1                        : signed(15 DOWNTO 0);

BEGIN
  u_generatorphazi : generatorphazi
    PORT MAP( clk => clk,
              reset => reset,
              enb => clk_enable,
              Out1 => generatorphazi_out1
              );

  u_MATLAB_Function : MATLAB_Function
    PORT MAP( u => In1,
              i => i,
              q => q
              );

  generatorphazi_out1_unsigned <= unsigned(generatorphazi_out1);

  cos_1_out1 <= nc(to_integer(generatorphazi_out1_unsigned));

  sin_1_out1 <= nc_2(to_integer(generatorphazi_out1_unsigned));

  i_signed <= signed(i);
```

```
  Product_out1 <= cos_1_out1 * i_signed;

  q_signed <= signed(q);

  Product1_out1 <= sin_1_out1 * q_signed;

  Add2_out1 <= Product_out1 - Product1_out1;

  Out1 <= std_logic_vector(Add2_out1);

  ce_out <= clk_enable;

END rtl;
```

СИГНАЛЫ

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

PACKAGE mod_rsvd_pkg IS
  TYPE vector_of_signed8 IS ARRAY (NATURAL RANGE <>) OF signed(7 DOWNTO 0);
END mod_rsvd_pkg;
```

ПРИЛОЖЕНИЕ Д

Листинг программы «Генератор фазы»

```
COMPONENT generatorphazi
PORT( clk                               :   IN    std_logic;
          reset                             :   IN    std_logic;
          enb                               :   IN    std_logic;
          Out1                              :   OUT   std_logic_vector(7 DOWNTO 0)
          );
  END COMPONENT;


  FOR ALL : generatorphazi
    USE ENTITY work.generatorphazi(rtl);
```

ПРИЛОЖЕНИЕ Е

Листинг программы «Генератор огибающей»

```
COMPONENT generatorogiba                                    :   IN
std_logic_vector(7 DOWNTO 0);
          i                                 :   OUT   std_logic_vector(7 DOWNTO 0);
          q                                 :   OUT   std_logic_vector(7 DOWNTO 0)
          );
  END COMPONENT;

  FOR ALL : generatorogiba
    USE ENTITY work.MATLAB_Function(rtl);
```

ПРИЛОЖЕНИЕ Ж

Листинг программы «Демодулятор»

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
USE work.demod_pkg.ALL;

ENTITY demod IS
PORT( clk                               :   IN    std_logic;
        reset                             :   IN    std_logic;
        clk_enable                        :   IN    std_logic;
        In1                               :   IN    std_logic_vector(7 DOWNTO 0);
-
        ce_out                            :   OUT   std_logic;
        Out1                              :   OUT   std_logic_vector(15 DOWNTO 0);
        Out2                              :   OUT   std_logic_vector(15 DOWNTO 0)
        );
END demod;


ARCHITECTURE rtl OF demod IS


  COMPONENT MATLAB_Function
PORT( u                                 :   IN    std_logic_vector(7 DOWNTO 0);
          y                                 :   OUT   std_logic_vector(15 DOWNTO 0)
          );
  END COMPONENT;

  COMPONENT MATLAB_Function1
PORT( u                                 :   IN    std_logic_vector(7 DOWNTO 0);
          y                                 :   OUT   std_logic_vector(15 DOWNTO 0)
          );
  END COMPONENT;


  FOR ALL : MATLAB_Function
    USE ENTITY work.MATLAB_Function(rtl);

  FOR ALL : MATLAB_Function1
    USE ENTITY work.MATLAB_Function1(rtl);

  CONSTANT nc                             : vector_of_signed8(0 TO 255) :=
    (to_signed(16#00#, 8), to_signed(-16#03#, 8), to_signed(-16#06#, 8),
to_signed(-16#09#, 8),
     to_signed(-16#0C#, 8), to_signed(-16#10#, 8), to_signed(-16#13#, 8),
to_signed(-16#16#, 8),
     to_signed(-16#19#, 8), to_signed(-16#1C#, 8), to_signed(-16#1F#, 8),
to_signed(-16#22#, 8),
     to_signed(-16#25#, 8), to_signed(-16#28#, 8), to_signed(-16#2B#, 8),
to_signed(-16#2E#, 8),
     to_signed(-16#31#, 8), to_signed(-16#33#, 8), to_signed(-16#36#, 8),
to_signed(-16#39#, 8),
     to_signed(-16#3C#, 8), to_signed(-16#3F#, 8), to_signed(-16#41#, 8),
to_signed(-16#44#, 8),
     to_signed(-16#47#, 8), to_signed(-16#49#, 8), to_signed(-16#4C#, 8),
```

```
to_signed(-16#4E#, 8),
     to_signed(-16#51#, 8), to_signed(-16#53#, 8), to_signed(-16#55#, 8),
to_signed(-16#58#, 8),
     to_signed(-16#5A#, 8), to_signed(-16#5C#, 8), to_signed(-16#5E#, 8),
to_signed(-16#60#, 8),
     to_signed(-16#62#, 8), to_signed(-16#64#, 8), to_signed(-16#66#, 8),
to_signed(-16#68#, 8),
     to_signed(-16#6A#, 8), to_signed(-16#6B#, 8), to_signed(-16#6D#, 8),
to_signed(-16#6F#, 8),
     to_signed(-16#70#, 8), to_signed(-16#71#, 8), to_signed(-16#73#, 8),
to_signed(-16#74#, 8),
     to_signed(-16#75#, 8), to_signed(-16#76#, 8), to_signed(-16#78#, 8),
to_signed(-16#79#, 8),
     to_signed(-16#7A#, 8), to_signed(-16#7A#, 8), to_signed(-16#7B#, 8),
to_signed(-16#7C#, 8),
     to_signed(-16#7D#, 8), to_signed(-16#7D#, 8), to_signed(-16#7E#, 8),
to_signed(-16#7E#, 8),
     to_signed(-16#7E#, 8), to_signed(-16#7F#, 8), to_signed(-16#7F#, 8),
to_signed(-16#7F#, 8),
     to_signed(-16#7F#, 8), to_signed(-16#7F#, 8), to_signed(-16#7F#, 8),
to_signed(-16#7F#, 8),
     to_signed(-16#7E#, 8), to_signed(-16#7E#, 8), to_signed(-16#7E#, 8),
to_signed(-16#7D#, 8),
     to_signed(-16#7D#, 8), to_signed(-16#7C#, 8), to_signed(-16#7B#, 8),
to_signed(-16#7A#, 8),
     to_signed(-16#7A#, 8), to_signed(-16#79#, 8), to_signed(-16#78#, 8),
to_signed(-16#76#, 8),
     to_signed(-16#75#, 8), to_signed(-16#74#, 8), to_signed(-16#73#, 8),
to_signed(-16#71#, 8),
     to_signed(-16#70#, 8), to_signed(-16#6F#, 8), to_signed(-16#6D#, 8),
to_signed(-16#6B#, 8),
     to_signed(-16#6A#, 8), to_signed(-16#68#, 8), to_signed(-16#66#, 8),
to_signed(-16#64#, 8),
     to_signed(-16#62#, 8), to_signed(-16#60#, 8), to_signed(-16#5E#, 8),
to_signed(-16#5C#, 8),
     to_signed(-16#5A#, 8), to_signed(-16#58#, 8), to_signed(-16#55#, 8),
to_signed(-16#53#, 8),
     to_signed(-16#51#, 8), to_signed(-16#4E#, 8), to_signed(-16#4C#, 8),
to_signed(-16#49#, 8),
     to_signed(-16#47#, 8), to_signed(-16#44#, 8), to_signed(-16#41#, 8),
to_signed(-16#3F#, 8),
     to_signed(-16#3C#, 8), to_signed(-16#39#, 8), to_signed(-16#36#, 8),
to_signed(-16#33#, 8),
     to_signed(-16#31#, 8), to_signed(-16#2E#, 8), to_signed(-16#2B#, 8),
to_signed(-16#28#, 8),
     to_signed(-16#25#, 8), to_signed(-16#22#, 8), to_signed(-16#1F#, 8),
to_signed(-16#1C#, 8),
     to_signed(-16#19#, 8), to_signed(-16#16#, 8), to_signed(-16#13#, 8),
to_signed(-16#10#, 8),
     to_signed(-16#0C#, 8), to_signed(-16#09#, 8), to_signed(-16#06#, 8),
to_signed(-16#03#, 8),
     to_signed(16#00#, 8), to_signed(16#03#, 8), to_signed(16#06#, 8),
to_signed(16#09#, 8), to_signed(16#0C#, 8),
     to_signed(16#10#, 8), to_signed(16#13#, 8), to_signed(16#16#, 8),
to_signed(16#19#, 8), to_signed(16#1C#, 8),
     to_signed(16#1F#, 8), to_signed(16#22#, 8), to_signed(16#25#, 8),
to_signed(16#28#, 8), to_signed(16#2B#, 8),
     to_signed(16#2E#, 8), to_signed(16#31#, 8), to_signed(16#33#, 8),
to_signed(16#36#, 8), to_signed(16#39#, 8),
     to_signed(16#3C#, 8), to_signed(16#3F#, 8), to_signed(16#41#, 8),
to_signed(16#44#, 8), to_signed(16#47#, 8),
     to_signed(16#49#, 8), to_signed(16#4C#, 8), to_signed(16#4E#, 8),
to_signed(16#51#, 8), to_signed(16#53#, 8),
```

```
     to_signed(16#55#, 8), to_signed(16#58#, 8), to_signed(16#5A#, 8),
to_signed(16#5C#, 8), to_signed(16#5E#, 8),
     to_signed(16#60#, 8), to_signed(16#62#, 8), to_signed(16#64#, 8),
to_signed(16#66#, 8), to_signed(16#68#, 8),
     to_signed(16#6A#, 8), to_signed(16#6B#, 8), to_signed(16#6D#, 8),
to_signed(16#6F#, 8), to_signed(16#70#, 8),
     to_signed(16#71#, 8), to_signed(16#73#, 8), to_signed(16#74#, 8),
to_signed(16#75#, 8), to_signed(16#76#, 8),
     to_signed(16#78#, 8), to_signed(16#79#, 8), to_signed(16#7A#, 8),
to_signed(16#7A#, 8), to_signed(16#7B#, 8),
     to_signed(16#7C#, 8), to_signed(16#7D#, 8), to_signed(16#7D#, 8),
to_signed(16#7E#, 8), to_signed(16#7E#, 8),
     to_signed(16#7E#, 8), to_signed(16#7F#, 8), to_signed(16#7F#, 8),
to_signed(16#7F#, 8), to_signed(16#7F#, 8),
     to_signed(16#7F#, 8), to_signed(16#7F#, 8), to_signed(16#7F#, 8),
to_signed(16#7E#, 8), to_signed(16#7E#, 8),
     to_signed(16#7E#, 8), to_signed(16#7D#, 8), to_signed(16#7D#, 8),
to_signed(16#7C#, 8), to_signed(16#7B#, 8),
     to_signed(16#7A#, 8), to_signed(16#7A#, 8), to_signed(16#79#, 8),
to_signed(16#78#, 8), to_signed(16#76#, 8),
     to_signed(16#75#, 8), to_signed(16#74#, 8), to_signed(16#73#, 8),
to_signed(16#71#, 8), to_signed(16#70#, 8),
     to_signed(16#6F#, 8), to_signed(16#6D#, 8), to_signed(16#6B#, 8),
to_signed(16#6A#, 8), to_signed(16#68#, 8),
     to_signed(16#66#, 8), to_signed(16#64#, 8), to_signed(16#62#, 8),
to_signed(16#60#, 8), to_signed(16#5E#, 8),
     to_signed(16#5C#, 8), to_signed(16#5A#, 8), to_signed(16#58#, 8),
to_signed(16#55#, 8), to_signed(16#53#, 8),
     to_signed(16#51#, 8), to_signed(16#4E#, 8), to_signed(16#4C#, 8),
to_signed(16#49#, 8), to_signed(16#47#, 8),
     to_signed(16#44#, 8), to_signed(16#41#, 8), to_signed(16#3F#, 8),
to_signed(16#3C#, 8), to_signed(16#39#, 8),
     to_signed(16#36#, 8), to_signed(16#33#, 8), to_signed(16#31#, 8),
to_signed(16#2E#, 8), to_signed(16#2B#, 8),
     to_signed(16#28#, 8), to_signed(16#25#, 8), to_signed(16#22#, 8),
to_signed(16#1F#, 8), to_signed(16#1C#, 8),
     to_signed(16#19#, 8), to_signed(16#16#, 8), to_signed(16#13#, 8),
to_signed(16#10#, 8), to_signed(16#0C#, 8),
     to_signed(16#09#, 8), to_signed(16#06#, 8), to_signed(16#03#, 8));
  CONSTANT nc_2                           : vector_of_signed8(0 TO 255) :=
    (to_signed(16#7F#, 8), to_signed(16#7F#, 8), to_signed(16#7F#, 8),
to_signed(16#7F#, 8), to_signed(16#7E#, 8),
     to_signed(16#7E#, 8), to_signed(16#7E#, 8), to_signed(16#7D#, 8),
to_signed(16#7D#, 8), to_signed(16#7C#, 8),
     to_signed(16#7B#, 8), to_signed(16#7A#, 8), to_signed(16#7A#, 8),
to_signed(16#79#, 8), to_signed(16#78#, 8),
     to_signed(16#76#, 8), to_signed(16#75#, 8), to_signed(16#74#, 8),
to_signed(16#73#, 8), to_signed(16#71#, 8),
     to_signed(16#70#, 8), to_signed(16#6F#, 8), to_signed(16#6D#, 8),
to_signed(16#6B#, 8), to_signed(16#6A#, 8),
     to_signed(16#68#, 8), to_signed(16#66#, 8), to_signed(16#64#, 8),
to_signed(16#62#, 8), to_signed(16#60#, 8),
     to_signed(16#5E#, 8), to_signed(16#5C#, 8), to_signed(16#5A#, 8),
to_signed(16#58#, 8), to_signed(16#55#, 8),
     to_signed(16#53#, 8), to_signed(16#51#, 8), to_signed(16#4E#, 8),
to_signed(16#4C#, 8), to_signed(16#49#, 8),
     to_signed(16#47#, 8), to_signed(16#44#, 8), to_signed(16#41#, 8),
to_signed(16#3F#, 8), to_signed(16#3C#, 8),
     to_signed(16#39#, 8), to_signed(16#36#, 8), to_signed(16#33#, 8),
to_signed(16#31#, 8), to_signed(16#2E#, 8),
     to_signed(16#2B#, 8), to_signed(16#28#, 8), to_signed(16#25#, 8),
to_signed(16#22#, 8), to_signed(16#1F#, 8),
     to_signed(16#1C#, 8), to_signed(16#19#, 8), to_signed(16#16#, 8),
```

```
to_signed(16#13#, 8), to_signed(16#10#, 8),
     to_signed(16#0C#, 8), to_signed(16#09#, 8), to_signed(16#06#, 8),
to_signed(16#03#, 8), to_signed(16#00#, 8),
     to_signed(-16#03#, 8), to_signed(-16#06#, 8), to_signed(-16#09#, 8),
to_signed(-16#0C#, 8),
     to_signed(-16#10#, 8), to_signed(-16#13#, 8), to_signed(-16#16#, 8),
to_signed(-16#19#, 8),
     to_signed(-16#1C#, 8), to_signed(-16#1F#, 8), to_signed(-16#22#, 8),
to_signed(-16#25#, 8),
     to_signed(-16#28#, 8), to_signed(-16#2B#, 8), to_signed(-16#2E#, 8),
to_signed(-16#31#, 8),
     to_signed(-16#33#, 8), to_signed(-16#36#, 8), to_signed(-16#39#, 8),
to_signed(-16#3C#, 8),
     to_signed(-16#3F#, 8), to_signed(-16#41#, 8), to_signed(-16#44#, 8),
to_signed(-16#47#, 8),
     to_signed(-16#49#, 8), to_signed(-16#4C#, 8), to_signed(-16#4E#, 8),
to_signed(-16#51#, 8),
     to_signed(-16#53#, 8), to_signed(-16#55#, 8), to_signed(-16#58#, 8),
to_signed(-16#5A#, 8),
     to_signed(-16#5C#, 8), to_signed(-16#5E#, 8), to_signed(-16#60#, 8),
to_signed(-16#62#, 8),
     to_signed(-16#64#, 8), to_signed(-16#66#, 8), to_signed(-16#68#, 8),
to_signed(-16#6A#, 8),
     to_signed(-16#6B#, 8), to_signed(-16#6D#, 8), to_signed(-16#6F#, 8),
to_signed(-16#70#, 8),
     to_signed(-16#71#, 8), to_signed(-16#73#, 8), to_signed(-16#74#, 8),
to_signed(-16#75#, 8),
     to_signed(-16#76#, 8), to_signed(-16#78#, 8), to_signed(-16#79#, 8),
to_signed(-16#7A#, 8),
     to_signed(-16#7A#, 8), to_signed(-16#7B#, 8), to_signed(-16#7C#, 8),
to_signed(-16#7D#, 8),
     to_signed(-16#7D#, 8), to_signed(-16#7E#, 8), to_signed(-16#7E#, 8),
to_signed(-16#7E#, 8),
     to_signed(-16#7F#, 8), to_signed(-16#7F#, 8), to_signed(-16#7F#, 8),
to_signed(-16#7F#, 8),
     to_signed(-16#7F#, 8), to_signed(-16#7F#, 8), to_signed(-16#7F#, 8),
to_signed(-16#7E#, 8),
     to_signed(-16#7E#, 8), to_signed(-16#7E#, 8), to_signed(-16#7D#, 8),
to_signed(-16#7D#, 8),
     to_signed(-16#7C#, 8), to_signed(-16#7B#, 8), to_signed(-16#7A#, 8),
to_signed(-16#7A#, 8),
     to_signed(-16#79#, 8), to_signed(-16#78#, 8), to_signed(-16#76#, 8),
to_signed(-16#75#, 8),
     to_signed(-16#74#, 8), to_signed(-16#73#, 8), to_signed(-16#71#, 8),
to_signed(-16#70#, 8),
     to_signed(-16#6F#, 8), to_signed(-16#6D#, 8), to_signed(-16#6B#, 8),
to_signed(-16#6A#, 8),
     to_signed(-16#68#, 8), to_signed(-16#66#, 8), to_signed(-16#64#, 8),
to_signed(-16#62#, 8),
     to_signed(-16#60#, 8), to_signed(-16#5E#, 8), to_signed(-16#5C#, 8),
to_signed(-16#5A#, 8),
     to_signed(-16#58#, 8), to_signed(-16#55#, 8), to_signed(-16#53#, 8),
to_signed(-16#51#, 8),
     to_signed(-16#4E#, 8), to_signed(-16#4C#, 8), to_signed(-16#49#, 8),
to_signed(-16#47#, 8),
     to_signed(-16#44#, 8), to_signed(-16#41#, 8), to_signed(-16#3F#, 8),
to_signed(-16#3C#, 8),
     to_signed(-16#39#, 8), to_signed(-16#36#, 8), to_signed(-16#33#, 8),
to_signed(-16#31#, 8),
     to_signed(-16#2E#, 8), to_signed(-16#2B#, 8), to_signed(-16#28#, 8),
to_signed(-16#25#, 8),
     to_signed(-16#22#, 8), to_signed(-16#1F#, 8), to_signed(-16#1C#, 8),
to_signed(-16#19#, 8),
```

```
     to_signed(-16#16#, 8), to_signed(-16#13#, 8), to_signed(-16#10#, 8),
to_signed(-16#0C#, 8),
     to_signed(-16#09#, 8), to_signed(-16#06#, 8), to_signed(-16#03#, 8),
to_signed(16#00#, 8),
     to_signed(16#03#, 8), to_signed(16#06#, 8), to_signed(16#09#, 8),
to_signed(16#0C#, 8), to_signed(16#10#, 8),
     to_signed(16#13#, 8), to_signed(16#16#, 8), to_signed(16#19#, 8),
to_signed(16#1C#, 8), to_signed(16#1F#, 8),
     to_signed(16#22#, 8), to_signed(16#25#, 8), to_signed(16#28#, 8),
to_signed(16#2B#, 8), to_signed(16#2E#, 8),
     to_signed(16#31#, 8), to_signed(16#33#, 8), to_signed(16#36#, 8),
to_signed(16#39#, 8), to_signed(16#3C#, 8),
     to_signed(16#3F#, 8), to_signed(16#41#, 8), to_signed(16#44#, 8),
to_signed(16#47#, 8), to_signed(16#49#, 8),
     to_signed(16#4C#, 8), to_signed(16#4E#, 8), to_signed(16#51#, 8),
to_signed(16#53#, 8), to_signed(16#55#, 8),
     to_signed(16#58#, 8), to_signed(16#5A#, 8), to_signed(16#5C#, 8),
to_signed(16#5E#, 8), to_signed(16#60#, 8),
     to_signed(16#62#, 8), to_signed(16#64#, 8), to_signed(16#66#, 8),
to_signed(16#68#, 8), to_signed(16#6A#, 8),
     to_signed(16#6B#, 8), to_signed(16#6D#, 8), to_signed(16#6F#, 8),
to_signed(16#70#, 8), to_signed(16#71#, 8),
     to_signed(16#73#, 8), to_signed(16#74#, 8), to_signed(16#75#, 8),
to_signed(16#76#, 8), to_signed(16#78#, 8),
     to_signed(16#79#, 8), to_signed(16#7A#, 8), to_signed(16#7A#, 8),
to_signed(16#7B#, 8), to_signed(16#7C#, 8),
     to_signed(16#7D#, 8), to_signed(16#7D#, 8), to_signed(16#7E#, 8),
to_signed(16#7E#, 8), to_signed(16#7E#, 8),
     to_signed(16#7F#, 8), to_signed(16#7F#, 8), to_signed(16#7F#, 8));

  SIGNAL enb                              : std_logic;
  SIGNAL Constant4_out1                   : unsigned(7 DOWNTO 0);
  SIGNAL Delay3_out1                      : unsigned(7 DOWNTO 0);
  SIGNAL Add1_out1                        : unsigned(7 DOWNTO 0);
  SIGNAL alphasin_2_out1                  : signed(7 DOWNTO 0);
  SIGNAL In1_signed                       : signed(7 DOWNTO 0);
  SIGNAL Data_Type_Conversion1_out1       : signed(7 DOWNTO 0);
  SIGNAL Product2_out1                    : signed(15 DOWNTO 0);
  SIGNAL Data_Type_Conversion4_out1       : signed(7 DOWNTO 0);
  SIGNAL Data_Type_Conversion5_out1       : signed(7 DOWNTO 0);
  SIGNAL y                                : std_logic_vector(15 DOWNTO 0);
  SIGNAL cos_2_out1                       : signed(7 DOWNTO 0);
  SIGNAL Product3_out1                    : signed(15 DOWNTO 0);
  SIGNAL Data_Type_Conversion6_out1       : signed(7 DOWNTO 0);
  SIGNAL Data_Type_Conversion7_out1       : signed(7 DOWNTO 0);
  SIGNAL y_1                              : std_logic_vector(15 DOWNTO 0);

BEGIN
  u_MATLAB_Function : MATLAB_Function
    PORT MAP( u => std_logic_vector(Data_Type_Conversion5_out1),
              y => y
              );

  u_MATLAB_Function1 : MATLAB_Function1
    PORT MAP( u => std_logic_vector(Data_Type_Conversion7_out1),
              y => y_1
              );

  Constant4_out1 <= to_unsigned(16#07#, 8);

  enb <= clk_enable;
```

```
  Add1_out1 <= Constant4_out1 + Delay3_out1;

  Delay3_process : PROCESS (clk, reset)
  BEGIN
    IF reset = '1' THEN
      Delay3_out1 <= to_unsigned(16#00#, 8);
    ELSIF clk'EVENT AND clk = '1' THEN
      IF enb = '1' THEN
        Delay3_out1 <= Add1_out1;
      END IF;
    END IF;
  END PROCESS Delay3_process;


  alphasin_2_out1 <= nc(to_integer(Delay3_out1));

  In1_signed <= signed(In1);

  Data_Type_Conversion1_out1 <= In1_signed;

  Product2_out1 <= alphasin_2_out1 * Data_Type_Conversion1_out1;

  Data_Type_Conversion4_out1 <= Product2_out1(8 DOWNTO 1);

  Data_Type_Conversion5_out1 <= Data_Type_Conversion4_out1;

  cos_2_out1 <= nc_2(to_integer(Delay3_out1));

  Product3_out1 <= Data_Type_Conversion1_out1 * cos_2_out1;

  Data_Type_Conversion6_out1 <= Product3_out1(8 DOWNTO 1);

  Data_Type_Conversion7_out1 <= Data_Type_Conversion6_out1;

  ce_out <= clk_enable;

  Out1 <= y;

  Out2 <= y_1;

END rtl;
```

СИГНАЛЫ

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

PACKAGE demod_pkg IS
  TYPE vector_of_signed8 IS ARRAY (NATURAL RANGE <>) OF signed(7 DOWNTO 0);
END demod_pkg;
```

СОГЛАСОВАННЫЙФИЛЬТР

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
```

```
USE IEEE.numeric_std.ALL;

ENTITY filter IS
PORT( clk                               :   IN    std_logic;
         clk_enable                     :   IN    std_logic;
         reset                          :   IN    std_logic;
         filter_in                      :   IN    std_logic_vector(7 DOWNTO 0); -
- sfix8
         filter_out                     :   OUT   std_logic_vector(16 DOWNTO 0)
-- sfix17_En8
         );

END filter;


-------------------------------------------------------------------
--Module Architecture: filter
-------------------------------------------------------------------
ARCHITECTURE rtl OF filter IS
  -- Local Functions
  -- Type Definitions
  TYPE delay_pipeline_type IS ARRAY (NATURAL range <>) OF signed(7 DOWNTO 0); --
sfix8
  -- Constants
  CONSTANT coeff1                       : signed(7 DOWNTO 0) := to_signed(-1,
8); -- sfix8_En8
  CONSTANT coeff2                       : signed(7 DOWNTO 0) := to_signed(-5,
8); -- sfix8_En8
  CONSTANT coeff3                       : signed(7 DOWNTO 0) := to_signed(-9,
8); -- sfix8_En8
  CONSTANT coeff4                       : signed(7 DOWNTO 0) := to_signed(-11,
8); -- sfix8_En8
  CONSTANT coeff5                       : signed(7 DOWNTO 0) := to_signed(-5,
8); -- sfix8_En8
  CONSTANT coeff6                       : signed(7 DOWNTO 0) := to_signed(13,
8); -- sfix8_En8
  CONSTANT coeff7                       : signed(7 DOWNTO 0) := to_signed(38,
8); -- sfix8_En8
  CONSTANT coeff8                       : signed(7 DOWNTO 0) := to_signed(61,
8); -- sfix8_En8
  CONSTANT coeff9                       : signed(7 DOWNTO 0) := to_signed(70,
8); -- sfix8_En8
  CONSTANT coeff10                      : signed(7 DOWNTO 0) := to_signed(61,
8); -- sfix8_En8
  CONSTANT coeff11                      : signed(7 DOWNTO 0) := to_signed(38,
8); -- sfix8_En8
  CONSTANT coeff12                      : signed(7 DOWNTO 0) := to_signed(13,
8); -- sfix8_En8
  CONSTANT coeff13                      : signed(7 DOWNTO 0) := to_signed(-5,
8); -- sfix8_En8
  CONSTANT coeff14                      : signed(7 DOWNTO 0) := to_signed(-11,
8); -- sfix8_En8
  CONSTANT coeff15                      : signed(7 DOWNTO 0) := to_signed(-9,
8); -- sfix8_En8
  CONSTANT coeff16                      : signed(7 DOWNTO 0) := to_signed(-5,
8); -- sfix8_En8
  CONSTANT coeff17                      : signed(7 DOWNTO 0) := to_signed(-1,
8); -- sfix8_En8

  -- Signals
  SIGNAL delay_pipeline                 : delay_pipeline_type(0 TO 16); -- sfix8
  SIGNAL product17                      : signed(14 DOWNTO 0); -- sfix15_En8
  SIGNAL mulpwr2_temp                   : signed(8 DOWNTO 0); -- sfix9
```

```
SIGNAL product16                        : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp                         : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product15                        : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_1                       : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product14                        : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_2                       : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product13                        : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_3                       : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product12                        : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_4                       : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product11                        : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_5                       : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product10                        : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_6                       : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product9                         : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_7                       : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product8                         : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_8                       : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product7                         : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_9                       : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product6                         : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_10                      : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product5                         : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_11                      : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product4                         : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_12                      : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product3                         : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_13                      : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product2                         : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mul_temp_14                      : signed(15 DOWNTO 0); -- sfix16_En8
SIGNAL product1_cast                    : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL product1                         : signed(14 DOWNTO 0); -- sfix15_En8
SIGNAL mulpwr2_temp_1                   : signed(8 DOWNTO 0); -- sfix9
SIGNAL sum1                             : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp                         : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum2                             : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_1                       : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum3                             : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_2                       : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum4                             : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_3                       : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum5                             : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_4                       : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum6                             : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_5                       : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum7                             : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_6                       : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum8                             : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_7                       : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum9                             : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_8                       : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum10                            : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_9                       : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum11                            : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_10                      : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum12                            : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_11                      : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum13                            : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_12                      : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum14                            : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_13                      : signed(17 DOWNTO 0); -- sfix18_En8
SIGNAL sum15                            : signed(16 DOWNTO 0); -- sfix17_En8
SIGNAL add_temp_14                      : signed(17 DOWNTO 0); -- sfix18_En8
```

```
  SIGNAL sum16                            : signed(16 DOWNTO 0); -- sfix17_En8
  SIGNAL add_temp_15                      : signed(17 DOWNTO 0); -- sfix18_En8
  SIGNAL output_typeconvert               : signed(16 DOWNTO 0); -- sfix17_En8
  SIGNAL output_register                  : signed(16 DOWNTO 0); -- sfix17_En8


BEGIN

  -- Block Statements
  Delay_Pipeline_process : PROCESS (clk, reset)
  BEGIN
    IF reset = '1' THEN
      delay_pipeline(0 TO 16) <= (OTHERS => (OTHERS => '0'));
    ELSIF clk'event AND clk = '1' THEN
      IF clk_enable = '1' THEN
        delay_pipeline(0) <= signed(filter_in);
        delay_pipeline(1 TO 16) <= delay_pipeline(0 TO 15);
      END IF;
    END IF;
  END PROCESS Delay_Pipeline_process;

  mulpwr2_temp <= ('0' & delay_pipeline(16)) WHEN delay_pipeline(16) = "10000000"
ELSE -resize(delay_pipeline(16),9);

  product17 <= resize(mulpwr2_temp, 15);

  mul_temp <= delay_pipeline(15) * coeff16;
  product16 <= mul_temp(14 DOWNTO 0);

  mul_temp_1 <= delay_pipeline(14) * coeff15;
  product15 <= mul_temp_1(14 DOWNTO 0);

  mul_temp_2 <= delay_pipeline(13) * coeff14;
  product14 <= mul_temp_2(14 DOWNTO 0);

  mul_temp_3 <= delay_pipeline(12) * coeff13;
  product13 <= mul_temp_3(14 DOWNTO 0);

  mul_temp_4 <= delay_pipeline(11) * coeff12;
  product12 <= mul_temp_4(14 DOWNTO 0);

  mul_temp_5 <= delay_pipeline(10) * coeff11;
product11 <= mul_temp_5(14 DOWNTO 0);

  mul_temp_6 <= delay_pipeline(9) * coeff10;
  product10 <= mul_temp_6(14 DOWNTO 0);

  mul_temp_7 <= delay_pipeline(8) * coeff9;
  product9 <= mul_temp_7(14 DOWNTO 0);

  mul_temp_8 <= delay_pipeline(7) * coeff8;
  product8 <= mul_temp_8(14 DOWNTO 0);

  mul_temp_9 <= delay_pipeline(6) * coeff7;
  product7 <= mul_temp_9(14 DOWNTO 0);

  mul_temp_10 <= delay_pipeline(5) * coeff6;
product6 <= mul_temp_10(14 DOWNTO 0);

  mul_temp_11 <= delay_pipeline(4) * coeff5;
  product5 <= mul_temp_11(14 DOWNTO 0);

  mul_temp_12 <= delay_pipeline(3) * coeff4;
```

```
  product4 <= mul_temp_12(14 DOWNTO 0);

  mul_temp_13 <= delay_pipeline(2) * coeff3;
  product3 <= mul_temp_13(14 DOWNTO 0);

  mul_temp_14 <= delay_pipeline(1) * coeff2;
  product2 <= mul_temp_14(14 DOWNTO 0);

  product1_cast <= resize(product1, 17);

  mulpwr2_temp_1 <= ('0' & delay_pipeline(0)) WHEN delay_pipeline(0) = "10000000"
      ELSE -resize(delay_pipeline(0),9);

  product1 <= resize(mulpwr2_temp_1, 15);

  add_temp <= resize(product1_cast, 18) + resize(product2, 18);
  sum1 <= add_temp(16 DOWNTO 0);

  add_temp_1 <= resize(sum1, 18) + resize(product3, 18);
  sum2 <= add_temp_1(16 DOWNTO 0);

  add_temp_2 <= resize(sum2, 18) + resize(product4, 18);
  sum3 <= add_temp_2(16 DOWNTO 0);

  add_temp_3 <= resize(sum3, 18) + resize(product5, 18);
  sum4 <= add_temp_3(16 DOWNTO 0);

  add_temp_4 <= resize(sum4, 18) + resize(product6, 18);
  sum5 <= add_temp_4(16 DOWNTO 0);

  add_temp_5 <= resize(sum5, 18) + resize(product7, 18);
  sum6 <= add_temp_5(16 DOWNTO 0);

  add_temp_6 <= resize(sum6, 18) + resize(product8, 18);
  sum7 <= add_temp_6(16 DOWNTO 0);

  add_temp_7 <= resize(sum7, 18) + resize(product9, 18);
sum8 <= add_temp_7(16 DOWNTO 0);

  add_temp_8 <= resize(sum8, 18) + resize(product10, 18);
  sum9 <= add_temp_8(16 DOWNTO 0);

  add_temp_9 <= resize(sum9, 18) + resize(product11, 18);
  sum10 <= add_temp_9(16 DOWNTO 0);

  add_temp_10 <= resize(sum10, 18) + resize(product12, 18);
  sum11 <= add_temp_10(16 DOWNTO 0);

  add_temp_11 <= resize(sum11, 18) + resize(product13, 18);
  sum12 <= add_temp_11(16 DOWNTO 0);

  add_temp_12 <= resize(sum12, 18) + resize(product14, 18);
  sum13 <= add_temp_12(16 DOWNTO 0);

  add_temp_13 <= resize(sum13, 18) + resize(product15, 18);
  sum14 <= add_temp_13(16 DOWNTO 0);

  add_temp_14 <= resize(sum14, 18) + resize(product16, 18);
  sum15 <= add_temp_14(16 DOWNTO 0);

  add_temp_15 <= resize(sum15, 18) + resize(product17, 18);
  sum16 <= add_temp_15(16 DOWNTO 0);
```

```
  output_typeconvert <= sum16;

  Output_Register_process : PROCESS (clk, reset)
  BEGIN
    IF reset = '1' THEN
      output_register <= (OTHERS => '0');
    ELSIF clk'event AND clk = '1' THEN
      IF clk_enable = '1' THEN
        output_register <= output_typeconvert;
END IF;
    END IF;
  END PROCESS Output_Register_process;

  -- Assignment Statements
  filter_out <= std_logic_vector(output_register);
END rtl;
```

ПРИЛОЖЕНИЕ 3

Листинг программы «Декодирующее устройство»

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
USE work.DECODER_pkg.ALL;

ENTITY DECODER IS
PORT( clk                               :   IN    std_logic;
        reset                           :   IN    std_logic;
        clk_enable                      :   IN    std_logic;
        In1                             :   IN    std_logic_vector(15 DOWNTO 0);
-- int16
        In2                             :   IN    std_logic_vector(15 DOWNTO 0);
-- int16
        ce_out                          :   OUT   std_logic;
        Out1                            :   OUT   std_logic_vector(15 DOWNTO 0);
-- int16
        Out2                            :   OUT   real  -- double
        );
END DECODER;


ARCHITECTURE rtl OF DECODER IS

  -- Component Declarations
  COMPONENT QPSK_not_ascii
PORT( i                           :   IN    std_logic_vector(15 DOWNTO 0);  --
int16
          q                               :   IN    std_logic_vector(15 DOWNTO 0);
-- int16
          u                               :   OUT   vector_of_std_logic_vector16(0
TO 1)  -- int16 [2]
          );
  END COMPONENT;

  COMPONENT MATLAB_Function
PORT( u                           :   IN    vector_of_std_logic_vector16(0 TO
1);  -- int16 [2]
          y                               :   OUT   std_logic_vector(15 DOWNTO 0)
-- int16
          );
  END COMPONENT;

  COMPONENT BarkerSF1
PORT( clk                         :   IN    std_logic;
          reset                           :   IN    std_logic;
          enb                             :   IN    std_logic;
          In1                             :   IN    std_logic_vector(15 DOWNTO 0);
-- int16
          Out1                            :   OUT   std_logic_vector(31 DOWNTO 0)
-- sfix32_En14
          );
  END COMPONENT;

  COMPONENT BarkerSFinv1
PORT( clk                         :   IN    std_logic;
```

```
          reset                           :   IN    std_logic;
          enb                             :   IN    std_logic;
          In1                             :   IN    std_logic_vector(15 DOWNTO 0);
-- int16
          Out1                            :   OUT   std_logic_vector(31 DOWNTO 0)
-- sfix32_En14
          );
  END COMPONENT;

  COMPONENT MATLAB_Function1
PORT( u                                :   IN    std_logic;
          w                               :   IN    std_logic;
          y                               :   OUT   std_logic_vector(15 DOWNTO 0)
-- int16
          );
  END COMPONENT;

  -- Component Configuration Statements
  FOR ALL : QPSK_not_ascii
    USE ENTITY work.QPSK_not_ascii(rtl);

  FOR ALL : MATLAB_Function
    USE ENTITY work.MATLAB_Function(rtl);

  FOR ALL : BarkerSF1
    USE ENTITY work.BarkerSF1(rtl);

  FOR ALL : BarkerSFinv1
    USE ENTITY work.BarkerSFinv1(rtl);

  FOR ALL : MATLAB_Function1
    USE ENTITY work.MATLAB_Function1(rtl);

  -- Signals
  SIGNAL u                                : vector_of_std_logic_vector16(0 TO 1);
-- ufix16 [2]
  SIGNAL y                                : std_logic_vector(15 DOWNTO 0);  --
ufix16
  SIGNAL BarkerSF1_out1                   : std_logic_vector(31 DOWNTO 0);  --
ufix32
  SIGNAL BarkerSF1_out1_signed            : signed(31 DOWNTO 0);  -- sfix32_En14
  SIGNAL Constant3_out1                   : signed(7 DOWNTO 0);  -- int8
  SIGNAL Relational_Operator3_1_cast      : signed(31 DOWNTO 0);  -- sfix32_En14
  SIGNAL Relational_Operator3_relop1      : std_logic;
  SIGNAL BarkerSFinv1_out1                : std_logic_vector(31 DOWNTO 0);  --
ufix32
  SIGNAL BarkerSFinv1_out1_signed         : signed(31 DOWNTO 0);  -- sfix32_En14
  SIGNAL Relational_Operator2_1_cast      : signed(31 DOWNTO 0);  -- sfix32_En14
  SIGNAL Relational_Operator2_relop1      : std_logic;
  SIGNAL y_1                              : std_logic_vector(15 DOWNTO 0);  --
ufix16
  SIGNAL TmpGroundAtOut2Inport1_out1      : real := 0.0;  -- double

BEGIN
  u_QPSK_not_ascii : QPSK_not_ascii
    PORT MAP( i => In1,  -- int16
              q => In2,  -- int16
              u =>u  -- int16 [2]
              );

  u_MATLAB_Function : MATLAB_Function
```

```
    PORT MAP( u => u,  -- int16 [2]
              y =>y  -- int16
              );

  u_BarkerSF1 : BarkerSF1
    PORT MAP( clk => clk,
              reset => reset,
              enb => clk_enable,
              In1 =>y,  -- int16
              Out1 => BarkerSF1_out1  -- sfix32_En14
              );

  u_BarkerSFinv1 : BarkerSFinv1
    PORT MAP( clk => clk,
              reset => reset,
              enb => clk_enable,
              In1 =>y,  -- int16
              Out1 => BarkerSFinv1_out1  -- sfix32_En14
              );

  u_MATLAB_Function1 : MATLAB_Function1
    PORT MAP( u => Relational_Operator3_relop1,
              w => Relational_Operator2_relop1,
              y => y_1  -- int16
              );

  BarkerSF1_out1_signed <= signed(BarkerSF1_out1);

  Constant3_out1 <= to_signed(16#04#, 8);

  Relational_Operator3_1_cast <= resize(Constant3_out1 & '0' & '0' & '0' & '0' &
'0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0', 32);

  Relational_Operator3_relop1 <= '1' WHEN BarkerSF1_out1_signed >=
Relational_Operator3_1_cast ELSE
      '0';

  BarkerSFinv1_out1_signed <= signed(BarkerSFinv1_out1);

  Relational_Operator2_1_cast <= resize(Constant3_out1 & '0' & '0' & '0' & '0' &
'0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0', 32);

  Relational_Operator2_relop1 <= '1' WHEN BarkerSFinv1_out1_signed >=
Relational_Operator2_1_cast ELSE
      '0';

  TmpGroundAtOut2Inport1_out1 <= 0.0;

  ce_out <= clk_enable;


  Out1 <= y_1;

  Out2 <= TmpGroundAtOut2Inport1_out1;

END rtl;
```

RS ТРИГГЕР

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

ENTITY MATLAB_Function1 IS
PORT( u                                 :   IN    std_logic;
        w                                 :   IN    std_logic;
        y                                 :   OUT   std_logic_vector(15 DOWNTO 0)
-- int16
        );
END MATLAB_Function1;


ARCHITECTURE rtl OF MATLAB_Function1 IS

  -- Signals
  SIGNAL y_tmp                            : signed(15 DOWNTO 0);  -- int16

BEGIN
  y_tmp <= '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' &
'0' & '0' & '0' & u;

  y <= std_logic_vector(y_tmp);

END rtl;
```

Сигналы

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

PACKAGE DECODER_pkg IS
  TYPE vector_of_std_logic_vector16 IS ARRAY (NATURAL RANGE <>) OF
std_logic_vector(15 DOWNTO 0);
  TYPE vector_of_signed16 IS ARRAY (NATURAL RANGE <>) OF signed(15 DOWNTO 0);
END DECODER_pkg;
```