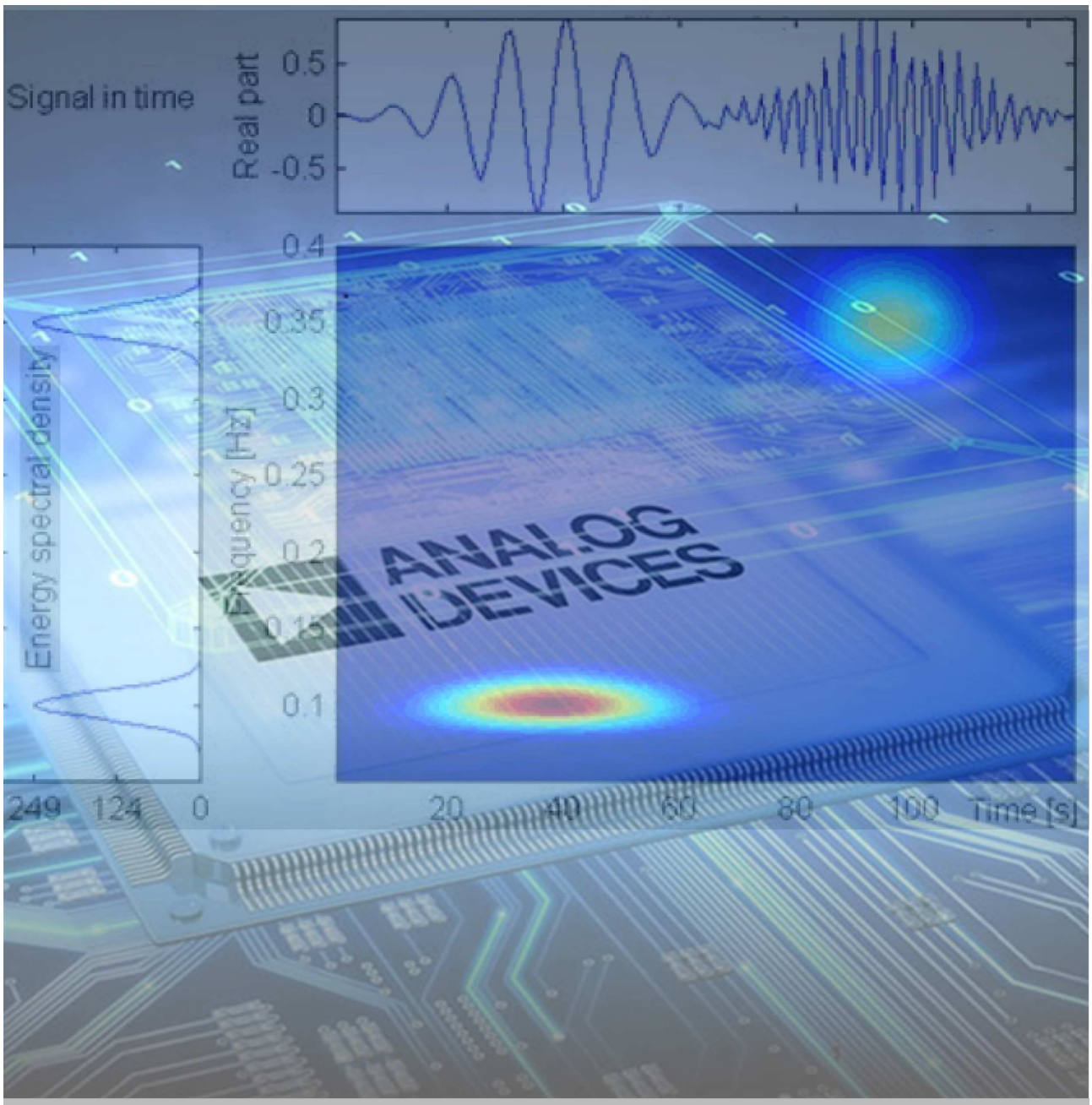


ПРОГРАММИРУЕМЫЕ МИКРОЭЛЕКТРОННЫЕ СИСТЕМЫ

Часть III. Цифровые сигнальные процессоры

РУКОВОДСТВО К ПРАКТИКУМУ

КФУ 2019

МИНИСТЕРСВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное учреждение высшего образования

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ФИЗИКИ

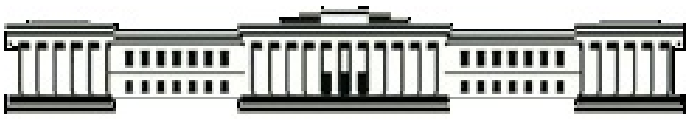
ОТДЕЛЕНИЕ РАДИОФИЗИКИ И ИНФОРМАЦИОННЫХ СИСТЕМ

Р. И. Гумеров, Д. А. Когогин

ЦИФРОВЫЕ СИГНАЛЬНЫЕ ПРОЦЕССОРЫ

РУКОВОДСТВО К ПРАКТИКУМУ

Электронное учебное пособие



Казань
2019

УДК 004.318-181.4

Рекомендовано к изданию
Учебно–методической комиссией
Института физики
Казанского (Приволжского) федерального университета
(*протокол № 5 от 24 января 2019 г.*)

Рецензент:
доцент кафедры компьютерных систем КНИТУ–КАИ им. А. Н. Туполева, к.ф.-м.н.
Е. С. Белашова

Гумеров Р. И., Когогин Д. А.
Цифровые сигнальные процессоры Руководство к практикуму. *Электронное учебное пособие* — Казань: Институт физики КФУ, 2019. — 85 с.

Учебное пособие предназначено для студентов, обучающихся по направлению 03.03.03 — Радиофизика. В руководстве в краткой форме изложены основные понятия теории сигналов, приведены методы представления сигналов, описания линейных систем, линейных цифровых фильтров; рассмотрены базовые алгоритмы цифровой обработки сигналов, архитектура и аппаратная реализация сигнальных микропроцессоров. Приводится описание инструментария разработки приложений. Даны примеры построения приложений.

Оглавление

ГЛАВА 1

ВВЕДЕНИЕ

Цифровой сигнальный процессор (англ. Digital signal processor, DSP) — специализированный микропроцессор, предназначенный для цифровой обработки сигналов в режиме реального времени, откуда и получил свое название. Цифровые сигнальные процессоры принимают на вход предварительно оцифрованные физические сигналы, например, звук, видеоизображение, показания температуры, давления и положения, и производят над ними математические манипуляции. Внутренняя структура цифровых сигнальных процессоров специально разрабатывается таким образом, чтобы они могли очень быстро выполнять такие математические функции, как «сложение», «вычитание», «умножение» и «деление».

Цифровая обработка сигналов, ее методы и средства представляют собой одну из наиболее мощных технологий, которые формируют науку и технологию в 21 веке. Революционные изменения с ее помощью произошли в целом ряде областей, таких как связь, медицинская диагностика, техника радаров и сонаров (радио и звуковая (эхо) локация), геолого- и нефтеразведка, высококачественное звуковоспроизведение и т.д. Одной и той же аббревиатурой «DSP» обозначают и цифровую обработку сигналов (Digital Signal Processing), и цифровые сигнальные процессоры (Digital Signal Processors). Какое именно значение используется — ясно из текста и обычно не комментируется.

Цифровая обработка сигналов отличается от других «компьютерных наук» (computer science — термин, принятый на западе и связанный как с проблемами вычислительной техники, так и с применением компьютеров в различных областях науки) уникальным типом используемых данных —сигналами. В большинстве случаев происхождение этих сигналов обусловлено и соответствует данным, которые мы получаем из ощущений реального мира: сейсмические колебания, визуальные изображения, звуковые колебания, и многое, многое другое. *DSP есть математические методы и алгоритмы обработки сигналов после их преобразования в цифровую форму, а также аппаратные средства для эффективной их реализации.*

Цифровую обработку сигналов можно определить, как подобласть вычислений общего назначения (обычных компьютерных вычислений), в которых архитектурные усовершенствования процессора, в том числе и такие как конвейеризация и параллелизм, необходимы для обеспечения скорости обработки сигналов в реальном времени. Корни, истоки цифровой обработки сигналов находятся в 1960-1970 годах, когда впервые стали доступны цифровые ЭВМ. В течение этой эпохи компьютеры были дорогими, и их применение для обработки сигналов было ограничено немногочисленными применениями для решения критических задач. Пионерские прорывы были сделаны по четырем ключевым областям: радары и сонары для целей национальной безопасности; нефтеразведка, где были сосредоточены огромные деньги; исследования космоса, где данные невосстановимы и их потеря повлекла бы огромные убытки; и медицина, а именно, диагностика (в особенно-

Рисунок 1.1 — Области применения DSP

сти «medical imaging»), где стоит вопрос о сохранении жизни. Революция персональных компьютеров 80-90-х годов вызвала взрыв новых приложений для DSP. Обогнав военные и правительственные применения, DSP неожиданно вышло на коммерческий рынок. DSP добралось до широкой публики в виде таких товаров, как мобильные телефоны, плееры компакт и DVD дисков, голосовая электронная почта. . . На рис. 1.1. иллюстрируются некоторые из этих применений.

ГЛАВА 2

СИГНАЛЫ И СИСТЕМЫ

Сигналы можно классифицировать по трем группам. Непрерывные сигналы — сигналы, область определения и область значений которых непрерывны, т.е. для каждой точки области определения и области значений можно найти точку, удаленную от нее на бесконечно малое расстояние. Непрерывные сигналы часто называют аналоговыми сигналами, отмечая, что они являются как бы аналогами порождающих их природных объектов. Примерами аналоговых сигналов могут служить изображения, сейсмические колебания, сигналы радаров, речевые сигналы. Вторая группа это дискретные сигналы. Область определения таких сигналов состоит из отдельных точек (отдельных чисел), называемых элементами дискретного сигнала. Обычно дискретный сигнал получают путем выборки аналогового сигнала. К третьей группе относят сигналы, область значений которых принимает только определенные квантованные значения. Обычно сигнал перед квантованием дискретизируют, а дискретный и квантованный сигнал называют цифровым сигналом. Дискретные, квантованные и цифровые сигналы — это, как правило, искусственные и в определенном смысле абстрактные объекты. Цифровые системы и компьютеры имеют дело только с цифровыми сигналами. Методы цифровой обработки сигналов, базирующиеся на цифровых сигналах, требуют детального обоснования амплитудного квантования. Последнее представляет собой нелинейный процесс, поэтому является весьма сложным и громоздким при математическом описании методов цифровой обработки. В большинстве случаев методы цифровой обработки базируются на дискретных сигналах. Опыт показывает, что теории, основанные на дискретных сигналах, часто применимы и для цифровых сигналов.

2.1 Представление сигналов

Сигналы (чаще всего непрерывные) обозначают первыми буквами латинского алфавита **a, b, c,** а их аргументы буквой x, подразумевая, что для многомерных непрерывных сигналов **x** — векторная переменная, с компонентами $x_1, \ldots x_n$, где n - размерность сигнала. Дискретные и цифровые сигналы обозначаются векторами, компонентами которых являются элементы сигнала, например **a** $=\{a_0, a_1, \ldots a_k, \ldots, a_{N-1}\}$, где N —количество элементов сигнала.

Любой оптический или электрический сигнал — это непрерывный сигнал. Примером цифрового сигнала может служить последовательность чисел, записанных в память цифровой вычислительной машины. Если бы эти числа можно было записывать с неограниченным количеством цифр, то есть с неограниченной точностью, то это был бы дискретный сигнал. С точки зрения математического описания сигналов различают также детерминированное и вероятностное описание. При детерминированном описании сигналы рассматриваются индивидуально, независимо друг от друга, и считается, что значение

сигнала может быть задано в любой точке, где он определен. Однако иногда индивидуальное рассмотрение характеристик физических объектов невозможно, а можно измерить и учесть только некоторое число макропараметров, характеризующих объекты в среднем. В этих случаях используется вероятностное описание, т.е. сигналы рассматриваются как выборочные функции, или реализации из некоторого ансамбля сигналов, и строится математическое описание не каждого отдельного сигнала, а ансамбля в целом.

Сигналы удобно рассматривать как точки или векторы в некотором функциональном пространстве (пространстве сигналов), а преобразования сигналов — как отображения в этом пространстве [1]. При этом свойства сигналов трактуются как свойства пространства. Термин *пространство* применяется для того, чтобы придать понятию множества сигналов геометрический смысл и, тем самым, наглядность. С пространством связано и понятие метрики пространства, т. е. способа, в соответствии с которым каждой паре точек пространства может быть поставлено в соответствие некоторое неотрицательное вещественное число, имеющее смысл расстояния между ними и удовлетворяющее следующим условиям:

$$\mathbf{a}_1, \mathbf{a}_2 = 0, \text{ если } \mathbf{a}_1 = \mathbf{a}_2;$$

$$\mathbf{d}(\mathbf{a}_1, \mathbf{a}_2) = \mathbf{d}(\mathbf{a}_2, \mathbf{a}_1);$$

$$\mathbf{d}(\mathbf{a}_1, \mathbf{a}_3) \leq \mathbf{d}(\mathbf{a}_1, \mathbf{a}_2) + \mathbf{d}(\mathbf{a}_2, \mathbf{a}_3).$$

Смысл первых двух условий очевиден; смысл третьего условия (правило треугольника) в том, что если две точки близки к третьей, то они должны быть близки и между собой. В теории сигналов наиболее часто встречается эвклидова метрика, поскольку удобна в расчетах и имеет определенный физический смысл: это мера энергии разности двух сигналов, измерение которой нетрудно воплотить в физическом приборе.

Обычно для сигналов как физических объектов выполняется принцип суперпозиции. Математически он формулируется как свойство линейности пространства сигналов. С понятием сигнала непосредственно связано понятие *системы*. Система преобразует входной сигнал в выходной. Основным элементом в применениях, связанных с обработкой сигналов, является анализ, проектирование и реализация системы, которая преобразует входной сигнал в наиболее подходящий для данного приложения выходной сигнал. При разработке теоретических обоснований системы на нее часто накладываются условия линейности и инвариантности к сдвигу. И хотя эти условия весьма жесткие, полученные с ними теоретические результаты на практике применимы к очень многим системам (по крайней мере, приближенно).

2.1.1 Линейность

Для пространства сигналов линейность определяется следующими условиями:

1. для двух любых его элементов **a** и **b** однозначно определен третий **c**, называемый суммой **a+b**, причем операция суммирования подчиняется закону коммутативности:

$$\mathbf{a}+\mathbf{b}=\mathbf{b}+\mathbf{a}, \quad \text{и ассоциативности:} \quad \mathbf{a}+(\mathbf{b}+\mathbf{c})=(\mathbf{a}+\mathbf{b})+\mathbf{c};$$

2. существует такой элемент **0**, что **a + 0 = a** для всех элементов пространства;
3. каждому элементу пространства можно поставить в соответствие противоположный ему элемент **−a** такой, что **a + (−a) = 0**;
4. для любого числа α и любого элемента пространства **a** определен принадлежащий этому пространству элемент $\alpha\mathbf{a}$, причем так, что

$$\alpha_1(\alpha_2\mathbf{a})=(\alpha_1\alpha_2)\mathbf{a}; \quad \mathbf{1a}=\mathbf{a};$$

$$(\alpha_1+\alpha_2)\mathbf{a}=\alpha_1\mathbf{a}+\alpha_2\mathbf{a};$$

$$\alpha(\mathbf{a}_1+\mathbf{a}_2)=\alpha\mathbf{a}_1+\alpha\mathbf{a}_2.$$

Элементы линейного пространства называются векторами. Вектор, образованный суммированием нескольких векторов со скалярными коэффициентами, называется линейной комбинацией:

$$\mathbf{a}=\sum_{k=0}^{N-1}\alpha_k\varphi_k. \tag{2.1}$$

Пространство $\mathbf{A}_N$, составленное из N линейно-независимых векторов $\{\varphi_k\}$, называется N -мерным линейным пространством. Множество линейно-независимых векторов $\{\varphi_k\}$ называется *базисом* этого пространства. Любое множество N линейно-независимых векторов в $\mathbf{A}_N$ может служить его базисом, и каждый вектор в $\mathbf{A}_N$ соответствует единственной линейной комбинации векторов $\{\varphi_k\}$ и единственному множеству скалярных коэффициентов $\{\alpha_k\}$ ($\{\alpha_k\}$ — *представление* вектора по отношению к данному базису). Если в пространстве определено скалярное произведение, то, пользуясь им, можно установить простое соотношение между сигналом и его представлением.

Представление сигналов как элементов линейного конечномерного пространства удобно потому, что позволяет описать любой сигнал набором стандартных базисных функций и набором чисел. Выбор базиса определяется удобством нахождения представления сигналов и, конечно, существом задачи, то есть особенностями сигналов.

Наиболее употребительны два класса базисных функций: сдвиговые базисные функции и мультипликативные. Первые строятся из одной функции путем сдвига по ее аргументу. Вторые же обладают тем свойством, что произведение двух функций дает также базисную функцию из той же системы. Это свойство используется для построения системы базисных функций путем многократного перемножения одной и той же функции. Примером первых могут служить импульсные базисные функции и функции отсчетов; примером вторых — экспоненциальные функции.

Взаимный базис. Пусть $\mathbf{A}_N$ — N-мерное пространство с базисом $\{\varphi_k\}$, $k = 0, 1, \ldots N - 1$, т. е. состоящее из векторов вида (2.1), а также $\{\psi_k\}$ — векторы, которые попарно ортогональны к $\{\varphi_k\}$ и нормированы так, что

$$\langle \varphi_k , \psi_k \rangle = \delta_{k,l} = \begin{cases} 1, & \text{если } k = l; \\ 0, & \text{если } k \neq l. \end{cases} \tag{2.2}$$

Символ $\delta_{k,l}$ есть символ Кронекера. Базис $\{\psi_k\}$, удовлетворяющий этому соотношению, называется взаимным к $\{\varphi_k\}$, и его можно использовать для вычисления коэффициентов представления $\{\alpha_k\}$:

$$\langle \mathbf{a}, \psi_l \rangle = \delta_{k,l} = \sum_{k=0}^{N-1} \alpha_k \langle \varphi_k, \psi_l \rangle = \sum_{k=0}^{N-1} \alpha_k \delta_{k,l} = \alpha_l. \tag{2.3}$$

Двумерные базисные функции обычно получают как произведение одномерных. Это делается для того, чтобы упростить вычисление коэффициентов представления сигналов по таким функциям: в случае разделяемых базисов, являющихся произведением функций одной переменной, вычисление двумерного интеграла скалярного произведения сводится к вычислению двух одномерных интегралов.

2.2 Линейные системы и системы, инвариантные к сдвигу

Взаимосвязь вход/выход определяется как система, если имеется однозначное соответствие между выходным сигналом и каким-либо входным [2]. Система T, которая соотносит вход $x(n)$ с выходом $y(n)$, может быть представлена как:

$$y(n) = T[x(n)].$$

Это очень широкое определение системы. Если нет каких-либо ограничений, характеризующих систему, то требуется полное задание взаимосвязей между входом и выходом. Знание состояний системы на выходе для некоторого множества входных сигналов, в общем, не позволяет нам определить выход системы для других наборов сигналов. Есть два ограничения, которые очень упрощают описание и анализ систем, — это линейность и инвариантность к сдвигу. И на практике, к счастью, множество систем могут (хотя бы приближенно) рассматриваться как линейные и инвариантные к сдвигу. Линейность системы определяется в следующем виде:

$$\textit{Линейность} \iff T[ax_1(n) + bx_2(n)] = ay_1(n) + by_2(n), \tag{2.4}$$

где $T[x_1(n)] = y_1(n), T[x_2(n)] = y_2(n)$, a и b — любые скалярные константы, а $\mathbf{A} \iff \mathbf{B}$ означает, что $\mathbf{A}$ является следствием $\mathbf{B}$ и $\mathbf{B}$ является следствием $\mathbf{A}$. Это условие также называется принципом суперпозиции.

Инвариантность к сдвигу, или пространственная инвариантность системы определяются следующим образом:

$$\textit{Инвариантность} \Longleftrightarrow T[x(n-m)] = y(n-m), \tag{2.5}$$

где $y(n) = T[x(n)]$, а m — любое целое число.

Примеры:

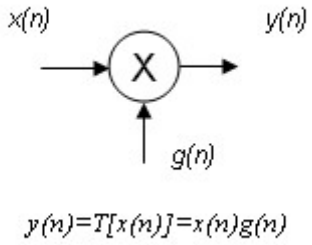


Рисунок 2.1 — Пример линейной системы

Система на рис. 2.1 не является инвариантной к сдвигу, поскольку

$$T[x(n-m)] = x(n-m)g(n),\ \ y(n-m) = x(n-m)g(n-m),$$

так как

$$T[x(n-m)] = x^2(n-m)y(n-m) = x^2(n-m).$$

Рассмотрим линейную систему T. При входном воздействии $x(n)$ выход системы $y(n)$ с учетом 2.5 будет

$$y(n) = T[x(n)] = T\Big[\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)\Big] = \sum_{k=-\infty}^{\infty} x(k)T[\delta(n-k)] = \sum_{k=-\infty}^{\infty} x(k)h_k(n). \tag{2.6}$$

Из 2.6 следует, что линейная система может быть полностью охарактеризована ее реакцией на импульс $\delta(n)$ и его сдвиг $\delta(n-k)$. $h_k(n)$ — отклик системы на $\delta(n)$ в момент n=k. Система является инвариантной к сдвигу, когда задержка входной последовательности $x(n-k)$ вызывает аналогичный временной сдвиг $y(n-k)$ на выходе. Тогда для $h(n)$

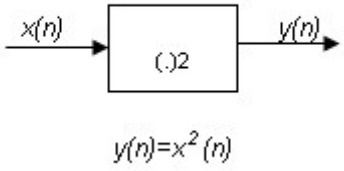


Рисунок 2.2 — Система нелинейная, однако является инвариантной к сдвигу

и $\delta(n)$ откликом на $\delta(n-k)$ будет $h(n-k)$, и тогда

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k). \tag{2.7}$$

Это выражение имеет важнейшее значение в цифровой обработке сигналов, представляет собой сумму парных произведений и называется сверткой. При цифровой обработке количество слагаемых в 2.7 должно быть ограничено. Удобно это ограничение связывать не с протяженностью сигнала, как в 2.7, а с протяженностью импульсной реакции фильтра, которая, как правило, меньше протяженности сигнала. Заменив в 2.7 индексы суммирования и введя ограничения по количеству отсчетов импульсной характеристики, получим конечномерное приближение к непрерывному фильтру

$$y(n) = \sum_{k=-0}^{N-1} x(n-k)h(k). \tag{2.8}$$

Эта формула называется формулой цифровой свертки и является базовой вычислительной процедурой в цифровой обработке сигналов. Операция свертки лежит в основе цифровых фильтров (КИХ и БИХ фильтров), вычисления авто- и взаимно корреляционных функций, умножения многочленов, интегральных преобразований (преобразование Фурье, вейвлет-преобразования и т. п.) и многого другого.

2.3 Передаточные функции

Помимо импульсной и частотной характеристики, представляющей импульсную характеристику в частотной области, линейную систему удобно представлять с помощью передаточной функции, которая базируется на методе z-преобразования [3]. Односторонее z-преобразование дискретной последовательности $x(nT)$, $n = 0, 1, 2, \ldots$, определяется рядом [3]:

$$X(z) = Z[x(nt)] = \sum_{n=0}^{\infty} x(nt)z^{-n}, \tag{2.9}$$

где $z = e^{j\varphi} = \alpha + j\beta$ — комплексная переменная. То есть z-преобразование представляет собой интегральное преобразование, где базисной функцией является z. Для равномерной сходимости ряда 2.9 достаточно, чтобы

$$\sum_{n=0}^{\infty} \left| x(nT)z^{-n} \right| = \sum_{n=0}^{\infty} \left| x(nT) \right| r^{-n} < \infty. \tag{2.10}$$

Область сходимости определяется кругом радиуса R в z-плоскости, вне которого ряд 2.9 сходится.

2.3.1 Основные свойства z-преобразования

1. *Линейность.* Пусть для последовательностей $x_1(nT)$ и $x_2(nT)$ z-преобразования — $X_1(z)$ и $X_2(z)$, соответственно, а a_1 и a_2 — постоянные и независящие от n коэффи-

циенты, тогда

$$y(nT) = a_1 x_1(nT) + a_2 x_2(nT)$$

имеет z-преобразование

$$Y(z) = a_1 X_1(z) + a_2 X_2(z).$$

2. *Сдвиг последовательности*. Если $Z\{x(nT)\} = X(z)$ и $x(nT) = 0$ при $n < 0$, то

$$y(nT) = x(nT - mT)$$

имеет z-преобразование

$$Y(z) = Z\{x(nT - mT)\} = z^{-m} X(z).$$

3. *Свертка последовательностей*. Свертка $x_1(nT)$ и $x_2(nT)$ во временной области:

$$y(nT) = \sum_{m=0}^{n} x_1(nT) x_2(nT - mT) = \sum_{m=0}^{n} x_1(nT - mT) x_2(nT)$$

может быть представлена в z-области в виде произведения

$$Y(z) = X_1(z) X_2(z).$$

Все это очень похоже на преобразование Фурье. Для того чтобы перейти из z-области во временную, необходимо выполнить обратное z-преобразование, которое ставит в соответствие функции комплексной переменной $X(z)$ решетчатую функцию

$$x(nT) = z^{-1}\{X(z)\}.$$

Эта последовательность определяется как

$$x(nT) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz, \tag{2.11}$$

где C – контур в области сходимости $X(z)z^{n-1}$, который охватывает начало координат в z-плоскости. Интеграл 2.11 вычисляют при помощи теоремы о вычетах. $x(nT)$ есть сумма вычетов подынтегрального выражения в полюсах в области внутри контура C:

$$x(nT) = \sum_{s_k} Res_k X(z) z^{n-1}.$$

Более удобно вычислять обратное z-преобразование через разложение на простые дроби. Для этого $X(z)$ должно быть рациональной функцией, тогда

$$X(z) = \frac{\beta_k}{1 - \alpha_k z^{-1}}.$$

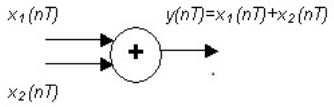


Рисунок 2.3 — Суммирование

При этом, используя свойство линейности и сходимость ряда, можно привести данное выражение к виду

$$x(nT) = \sum_{k=1}^{N} \beta_k (\alpha_k)^n. \tag{2.12}$$

Передаточной функцией линейной дискретной системы $H(z)$ называют отношение выходной последовательности, представленной в z-области (z-образ), к входной последовательности в z-области:

$$H(Z) = \frac{Y(Z}{X(Z)}.$$

Если применить z -преобразование к решению линейных разностных уравнений вида

$$\sum_{m=0}^{M-1} a_m y(nT - mT) = \sum_{k=0}^{N-1} b_k x(nT - kT),$$

которыми описываются линейные дискретные фильтры, то получим

$$\sum_{m=0}^{M-1} a_m z^{-m} Y(z) = \sum_{k=0}^{N-1} b_k z^{-k} X(z).$$

Поскольку $Y(z) = H(z)X(z)$, то

$$H(z) = \frac{\sum_{k=0}^{N-1} b_k z^{-k}}{\sum_{m=0}^{M-1} a_m z^{-m}} \tag{2.13}$$

— есть передаточная функция рекурсивного цифрового фильтра, и

$$H(z) = \sum_{k=0}^{N-1} b_k z^{-k} \tag{2.14}$$

— передаточная функция нерекурсивного фильтра.

2.3.2 Структурные схемы дискретных фильтров

Алгоритмы рекурсивных и нерекурсивных фильтров могут быть представлены в виде структурных схем, построенных на трех операциях: алгебраического сложения, умножения сигнала на константу и задержки сигнала на один интервал дискретизации.
Графически эти операции изображаются в следующем виде:

Суммирование, рис. 2.3

Умножение на коэффициент, рис. 2.4

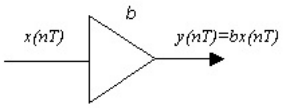


Рисунок 2.4 — Умножение на коэффициент

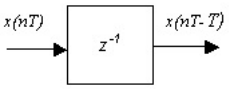


Рисунок 2.5 — Задержка на такт

Задержка на такт, рис. 2.5

Рассмотрим наиболее часто используемые структурные схемы фильтров.

Прямая форма структурной схемы реализуется непосредственно по разностному уравнению

$$\sum_{m=0}^{M-1} a_m y(nT - mT) = \sum_{k=0}^{N-1} b_k x(nT - kT),$$

или по передаточной функции

$$H(z) = \frac{\sum_{k=0}^{N-1} b_k z^{-k}}{1 + \sum_{m=1}^{M-1} a_m z^{-m}},$$

и имеет вид, представленный на рис. 2.6.

Прямая каноническая форма. Канонической называют структурную форму фильтра, в которой содержится минимальное число элементов задержки.

Структурные схемы нерекурсивных фильтров. Прямая форма является непосредственной реализацией передаточной характеристики нерекурсивного фильтра

$$H(z) = \sum_{k=0}^{N-1} b_k z^{-k},$$

или соответствующего разностного уравнения

$$y(nT) = \sum_{k=0}^{N-1} b_k x(nT - kT).$$

Прямая форма содержит $N - 1$ элементов задержки, N умножителей и сумматор с N входами. Другое название такой схемы — трансверсальный фильтр, или фильтр с многоотводной линией задержки.

2.3.3 Соединение фильтров

Эквивалентными называют фильтры, у которых при нулевых начальных условиях и одинаковых входных сигналах выходные сигналы тоже одинаковы.

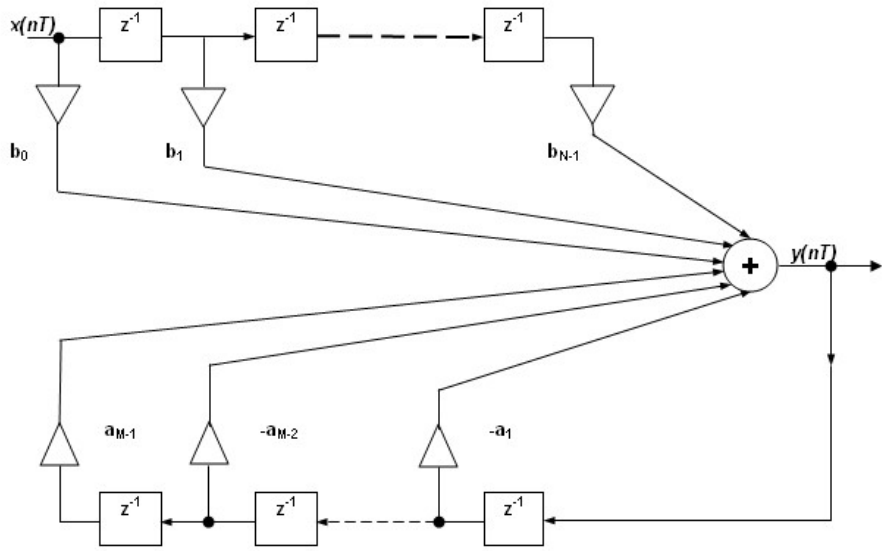


Рисунок 2.6 — Прямая форма

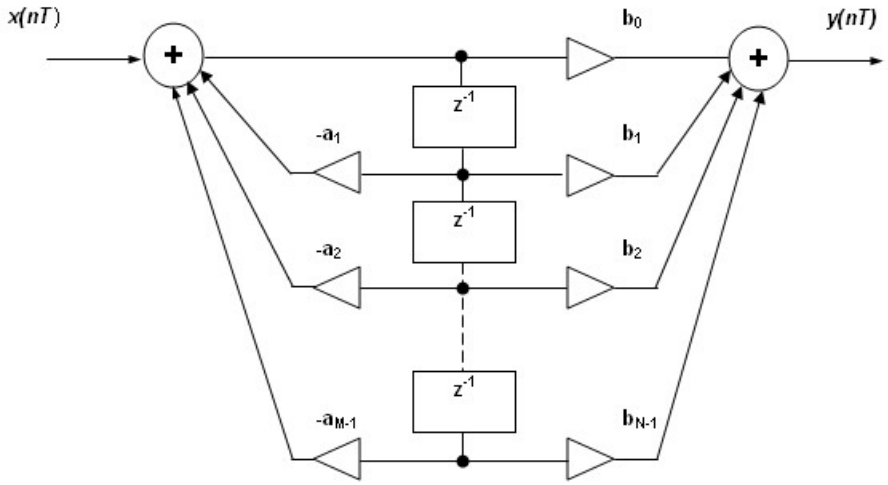


Рисунок 2.7 — Каноническая форма

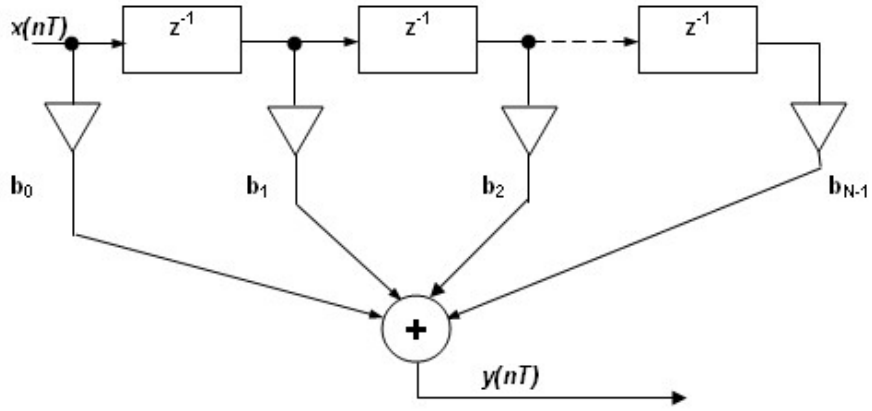


Рисунок 2.8 — Структурная схема нерекурсивного фильтра

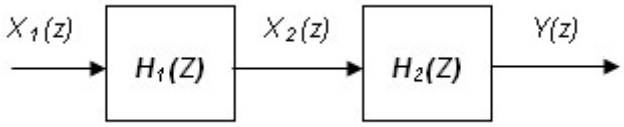


Рисунок 2.9 — Последовательное соединение фильтров

2.3.3.1 Последовательное соединение

Здесь выходной сигнал предшествующего фильтра является входным для последующего (рис. 2.9). Для такого соединения эквивалентная передаточная функция $H_e(z)$ равна произведению передаточных функций $H_1(z)$ и $H_2(z)$, то есть

$$H_e(z) = H_1(z) \times H_2(z) \text{ или } H_e(z) = \prod_{i=1}^{p} H_i(z). \tag{2.15}$$

2.3.3.2 Параллельное соединение

Входной сигнал для всех фильтров один и тот же, а выходной сигнал равен сумме выходных сигналов отдельных фильтров (рис. 2.10).

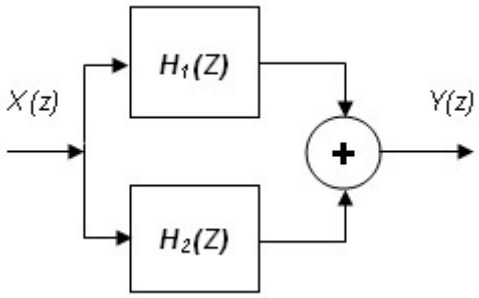


Рисунок 2.10 — Параллельное соединение фильтров

Для параллельного соединения фильтров эквивалентная передаточная функция равна сумме передаточных функций отдельных фильтров:

$$H_e(z) = H_1(z) + H_2(z) \text{ или } H_e(z) = \sum_{i=1}^{p} H_i(z). \tag{2.16}$$

Следует отметить еще один вариант соединения фильтров.

2.3.3.3 Соединение обратной связи

Представлено на рис. 2.11; возможна как отрицательная, так и положительная обратная связь.

Для такого соединения

$$H_e(z) = \frac{H_1(z)}{1 \pm H_1(z) \times H_2(z)}; \tag{2.17}$$

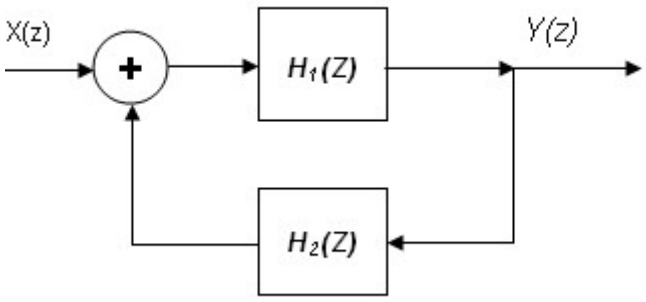


Рисунок 2.11 — Соединение обратной связи

в знаменателе знак плюс соответствует отрицательной обратной связи, а знак минус — положительной.

2.4 КИХ и БИХ фильтры

Фильтром с конечной импульсной характеристикой (КИХ-фильтром) называется фильтр, у которого импульсная характеристика представляет собой конечный дискретный сигнал (N-точечный дискретный сигнал), то есть может принимать отличные от нуля значения лишь при $n = 0, 1, 2, \ldots N - 1$.

Фильтром с бесконечной импульсной характеристикой (БИХ-фильтром) называют фильтр, у которого импульсная характеристика может принимать отличные от нуля значения на бесконечном множестве значений $n = 0, 1, 2, \ldots$

Нерекурсивный фильтр — всегда КИХ-фильтр. Рекурсивный фильтр может быть как БИХ-, так и КИХ-фильтром. Поскольку основные особенности проектирования и применения фильтров связаны с видом их импульсной характеристики (КИХ и БИХ), то, как правило, при описании фильтров используются термины БИХ и КИХ (IIR и FIR в английской аббревиатуре).

2.4.1 Реализуемость дискретных фильтров

Линейный дискретный фильтр физически реализуем, если его реакция (выходной сигнал) не опережает входного, то есть в момент времени nT реакция $y(nT)$ зависит лишь от значений $x(nT)$ в моменты $n'T \leq nT$ и не зависит от их значений в последующие моменты. Критерием реализуемости фильтра является равенство нулю отсчетов импульсной характеристики при отрицательных значениях моментов отсчетов, то есть $h(nT) = 0$ при $n < 0$.

2.4.2 Критерий устойчивости

Фильтр устойчив, если при любых начальных условиях реакция фильтра на любой ограниченный сигнал $x(nT)$ также ограничена. То есть, если $|x(nT)| \leq M_x < \infty$ для всех n=0,1,2,. . . , то $|y(nT)| \leq M_y < \infty$ для всех n, причем M_x и M_y постоянные, независящие от n. Отсюда $|y(nT)| \leq \sum_{m=0}^{\infty} |h(mT)||x(nT - mT)| \leq M_x \sum_{m=0}^{\infty} |h(mT)|$. Следовательно, критерием устойчивости дискретного линейного фильтра является абсо-

лютная сходимость ряда отсчетов импульсной характеристики:

$$\sum_{m=0}^{\infty} |h(mT)| \leq \infty. \tag{2.18}$$

Для передаточной функции можно аналогично записать: $|H(z)| \leq \sum_{k=0}^{\infty} |h(kT)||z^{-1}|$. Если $|z^{-1}| < 1$, то $|H(z)| \leq \sum_{k=0}^{\infty} |h(kT)|$. Это значит, что в устойчивой системе $H(z)$ конечна во всех точках z-плоскости, где $|z| \geq 1$, то есть $H(z)$ не должна иметь особых точек (полюсов) при $|z| \geq 1$. Система будет устойчива только тогда, когда все полюсы $H(z)$ расположены внутри единичного круга z-плоскости.

2.5 Частотные характеристики линейных дискретных фильтров

Для входной $x(nT)$ и выходной $y(nT)$ последовательностей дискретного фильтра с помощью преобразования Фурье получим их спектры $X(e^{j\omega T})$ и $Y(e^{j\omega T})$, то есть

$$X(e^{j\omega T}) = \sum_{n=0}^{\infty} x(nT)e^{j\omega nT} \text{ и } Y(e^{j\omega T}) = \sum_{n=0}^{\infty} y(nT)e^{j\omega nT}.$$

Тогда частотной характеристикой системы будет отношение

$$Z(e^{j\omega T}) = \frac{Y(e^{j\omega T})}{X(e^{j\omega T})}.$$

Это означает, что частотная характеристика есть передаточная функция при $z = e^{j\omega T}$ или $H(e^{j\omega T}) = H(z)|_{z=j\omega T}$.

Для рекурсивного фильтра в соответствии с 2.10

$$H(e^{j\omega T}) = \frac{\sum_{k=0}^{N-1} b_k e^{j\omega kT}}{1 + \sum_{m=1}^{M-1} a_m e^{j\omega mT}}, \tag{2.19}$$

а для нерекурсивного согласно 2.11:

$$H(e^{j\omega T}) = \sum_{k=0}^{N-1} b_k e^{j\omega kT} \quad \text{или} \quad H(e^{j\omega T}) = \sum_{k=0}^{N-1} h(kT)e^{j\omega kT}. \tag{2.20}$$

Поскольку $H(e^{j\omega T})$ — комплексная функция, то она может быть записана в виде $H(e^{j\omega T}) = A(\omega)e^{j\varphi(\omega)} = \Re(\omega) + j\Im(\omega)$, где $A(\omega) = |H(e^{j\omega T})|$ — модуль частотной характеристики или *амплитудно-частотная характеристика – АЧХ*; $\varphi(\omega)$— аргумент частотной характеристики — *фазо-частотная характеристика — ФЧХ*. Также $\Re(\omega) = A(\omega)\cos\varphi(\omega)$ и $\Im(\omega) = A(\omega)\sin\varphi(\omega)$. Тогда

$$\varphi(\omega) = \operatorname{arctg}\frac{\Im(\omega)}{\Re(\omega)}. \tag{2.21}$$

Нередко применяется также параметр, который называется ГВЗ — групповое время замедления:

$$\tau(\omega) = -\frac{d\varphi(\omega)}{d\omega}.$$

Очевидно, что для строго линейной фазочастотной характеристики ГВЗ есть величина постоянная.

2.5.1 Основные свойства частотных характеристик

1. Все частотные характеристики дискретных линейных фильтров являются непрерывными функциями частоты.
2. Все частотные характеристики есть периодические функции с периодом, равным частоте дискретизации.
3. Для вещественных фильтров (фильтров с вещественными коэффициентами) АЧХ ($A(\omega)$) и ГВЗ ($\tau(\omega)$) есть четные функции частоты, а ФЧХ ($\varphi(\omega)$) – нечетная функция частоты.

Для сравнения частотных характеристик различных фильтров частоту обычно нормируют. Нормирование производится исходя из следующих соображений: период дискретизации, определяемый теоремой Котельникова, есть $T = \frac{1}{f_s} = \frac{1}{2f_{max}}$, а рассматриваемый спектр частот может находиться в интервале $0 < f \leq f_{max}$ (здесь f_s – частота выборки сигнала, а f_{max} – максимальная гармоника, содержащаяся в спектре сигнала). Тогда, приведя этот интервал к частоте выборки, получим для нормированной частоты $\tilde{f}_s$ область задания $[0, 0.5]$. Для круговой частоты ω частота дискретизации ω_s и нормированный интервал частот будут находиться в пределах $[0, \pi]$.

В заключении данного раздела выделим основные положения:

1. Цифровые сигналы представляют собой, как правило, квантованные выборки аналоговых сигналов; выборки производятся в соответствии с теоремой Котельникова.
2. К этим сигналам с определенной степенью точности можно применить методы описания и анализа дискретных сигналов.
3. Преобразования сигналов осуществляются посредством систем. Наиболее широко разработаны методы описания, анализа и синтеза для линейных систем. Поэтому при рассмотрении системы нужно стремиться к ее линейному представлению, хотя бы приближенно.
4. Линейные системы полностью характеризуются во временной области импульсной характеристикой, в частотной области – частотной характеристикой. Переход от импульсной характеристики к частотной (и обратно) производится при помощи преобразования Фурье. Весьма удобной и информативной характеристикой линейной системы является передаточная функция, которая представляет реакцию системы с помощью z-преобразования.
5. Благодаря свойству линейности система может быть разбита на более простые элементы (подсистемы) с последовательным или параллельным соединением, или соединением обратной связи.

6. К системам, состоящим из подсистем с последовательным или параллельным соединением, можно применить методы конвейерной и параллельной обработки, соответственно. Оба подхода позволяют упростить реализацию системы, а также повысить ее производительность, либо снизить требования к производительности подсистем.
7. Одним из наиболее распространенных классов линейных дискретных систем являются фильтры.
8. Дискретные и цифровые фильтры подразделяются на рекурсивные и нерекурсивные, и описываются соответствующими линейными разностными уравнениями.
9. По виду импульсной характеристики фильтры разделяют на два класса: БИХ-фильтры – фильтры с бесконечной импульсной характеристикой, и КИХ-фильтры – фильтры с конечной импульсной характеристикой. Нерекурсивные фильтры всегда КИХ-фильтры, рекурсивные фильтры могут быть как КИХ- так и БИХ-фильтрами .
10. КИХ-фильтры всегда устойчивы, при проектировании БИХ-фильтров необходимо оценивать их устойчивость.
11. КИХ-фильтры могут иметь строго линейную ФЧХ (постоянное ГВЗ) и исключить частотные искажения, БИХ-фильтры могут иметь только приближенно линейную ФЧХ.

2.6 Представление сигналов в других базисах: вейвлет-преобразование

Вейвлет-преобразование — интегральное преобразование, которое представляет собой свертку вейвлет-функции с сигналом. Вейвлет-преобразование переводит сигнал из временного представления в частотно-временное. Термин «вейвлет» (англ. wavelet) в переводе с английского означает «маленькая (короткая) волна». Вейвлеты — это обобщенное название семейств математических функций определенной формы, которые локализованы как по времени, так и по частоте. Все базисные функции в вейвлет-преобразовании получаются из одной, так называемой «материнской функции» ψ путем её сдвига и масштабирования по оси времени:

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right), \tag{2.22}$$

где a – параметр масштаба анализирующей функции, b – параметр сдвига. Таким образом, вейвлет-преобразование некоторой функции $F(t)$ может быть записано следующим образом:

$$W(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty} F(x)\psi^*\left(\frac{x-b}{a}\right), \tag{2.23}$$

где ψ^* - функция, комплексно сопряженная к ψ.

2.6.1 Непрерывное вейвлет-преобразование

Пусть имеется определенная на всей действительной оси $(-\infty;\ +\infty)$ функция $s(t)$, имеющая конечную энергию (норму) и среднее значение, стремящееся к нулю на $\pm\infty$. Тогда непрерывным вейвлет-преобразованием (или вейвлетным образом) функции $s(t)$

называют функцию двух переменных:

$$C(a,b) = \langle s(x)\psi(a,b,x)\rangle = \int_{-\infty}^{+\infty} s(x)\psi(a,b,x)dx, \tag{2.24}$$

где $\psi(a,b,x) = \psi_{a,b}(x)$ – совокупность масштабированных и сдвинутых копий «материнского» (порождающего) вейвлета ψ, составляющая базис разложения. В качестве порождающих функций могут быть выбраны самые различные функции в зависимости от типа решаемых задач, но все они должны удовлетворять следующим требованиям:

1. Анализирующая вейвлет-функция должна иметь нулевое среднее значение:

$$\int_{-\infty}^{+\infty} \psi(x)dx = 0.$$

2. Все базисные функции должны быть получены из одной основной функции путем масштабирования и сдвига:

$$\psi_{a,b}(x) = \psi\left(\frac{x-b}{a}\right).$$

3. Наличие однозначного обратного преобразования, восстанавливающего исходный сигнал:

$$F(x) = \frac{1}{C_\psi a^2}\iint\limits_{\infty} W(a,b)\psi\left(\frac{x-b}{a}\right) da\ db.$$

C_ψ — нормирующий коэффициент (аналог коэффициента $\sqrt{2}\pi$ в обратном преобразовании Фурье):

$$C_\psi = \int_{-\infty}^{+\infty} \frac{|\psi(\omega)|^2}{\omega} d\omega < \infty.$$

$\psi(\omega)$ — Фурье-образ функции $\psi(x)$.

4. Хорошая локализация функции $\psi(x)$ по времени и частоте. Для того, чтобы перекрыть анализирующей функцией всю временную ось пространства сигнала, применяется операция сдвига (изменение коэффициента b): $\psi(a,x) = \psi(x-b)$, где значение b изменяется непрерывно для непрерывного вейвлет-преобразования. Чтобы перекрыть весь частотный диапазон пространства сигнала, применяется операция масштабирования (изменение коэффициента a): $\psi(a,x) = \frac{1}{\sqrt{a}}\psi(\frac{x}{a})$, где значение параметра a также изменяется непрерывно. Так, изменяя значение переменной $(x-b)$, можно перемещать вейвлет по всей временной оси, а варьируя значение переменной a, можем просматривать частотный спектр сигнала.

2.6.2 Дискретное вейвлет-преобразование

Отличие дискретного вейвлет-преобразования в том, что параметры a и b принимают дискретные значения и обычно представляют собой функции вида:

$$a = a_0^{-m}, \quad b = ka_0^{-m},$$

где $a_0 > 1, \quad m,\ k$ – множество целых чисел $(-\infty \ +\infty)$. Ширина вейвлетов зависит от параметра m, поэтому, для более эффективного перекрытия временной оси анализируемого сигнала, параметр сдвига также должен зависеть от m. Таким образом, получаем, что узкие вейвлеты (малое значение m, соответствует высоким частотам) сдвигаются на меньшие расстояния по сравнению с более широкими (большее значение параметра m, соответствует низким частотам). В общем случае значение параметра масштаба a_0 может принимать любые значения, превышающие единицу, однако на практике, в основном, это значение принимается равным 2. В таком случае вейвлет-преобразование называется диадным. Для него разработаны алгоритмы быстрого преобразования, аналогичные быстрому преобразованию Фурье, позволяющие существенно повысить быстродействие.

Базисные функции для дискретного вейвлет-преобразования выглядят следующим образом:

$$\psi_{m,k}(x) = a_0^{-\frac{m}{2}} \psi(a_0^{-m} x - k),$$

а вейвлет-коэффициенты для непрерывных сигналов $s(x)$:

$$C_{m,k} = \int_{-\infty}^{+\infty} s(x) \psi_{m,k}(x) dx.$$

Восстановить исходный сигнал $s(x)$ можно используя обратное дискретное вейвлет - преобразование:

$$s(x) = \sum_k \sum_m C_{m,k} \psi_{m,k}(x).$$

Основным преимуществом вейвлет-преобразования является то, что базисные функции здесь хорошо локализованы как по времени, так и по частоте. Это позволяет получить информацию о сигнале не только в частотной области (как для Фурье преобразования), но также и определить его временны́е характеристики (рис. 2.12).

К одной из наиболее общих и важных задач цифровой обработки сигналов относится очистка сигнала от шума. Методы шумоподавления весьма разнообразны и зависят как от характера шума, так и от особенностей сигнала, реализуются как с помощью линейных дискретных фильтров, так и посредством нелинейных алгоритмов. Поэтому нет особенного смысла рассматривать эти вопросы в отрыве от конкретной задачи. И, тем не менее, рассмотрим один из подходов, который не использует частотную фильтрацию непосредственно, а базируется на вейвлет-преобразовании и итерационных алгоритмах. Этот весьма эффективный аппарат, разработанный специалистами, участвовавшими в проекте HST (Hubble Space Telescope – космический телескоп Хаббла), рассмотрим в качестве примера, иллюстрирующего:

1. использования вейвлет-преобразования в цифровой обработке сигналов;
2. применения итерационных алгоритмов для выделения шума из смеси сигнал-шум.

Не вдаваясь в детали строгих математических выводов, рассмотрим вейвлет-преобразование с точки зрения цифровой обработки сигналов. Для наглядности будем сопоставлять с дискретным или быстрым преобразованием Фурье (БПФ). Аналогично

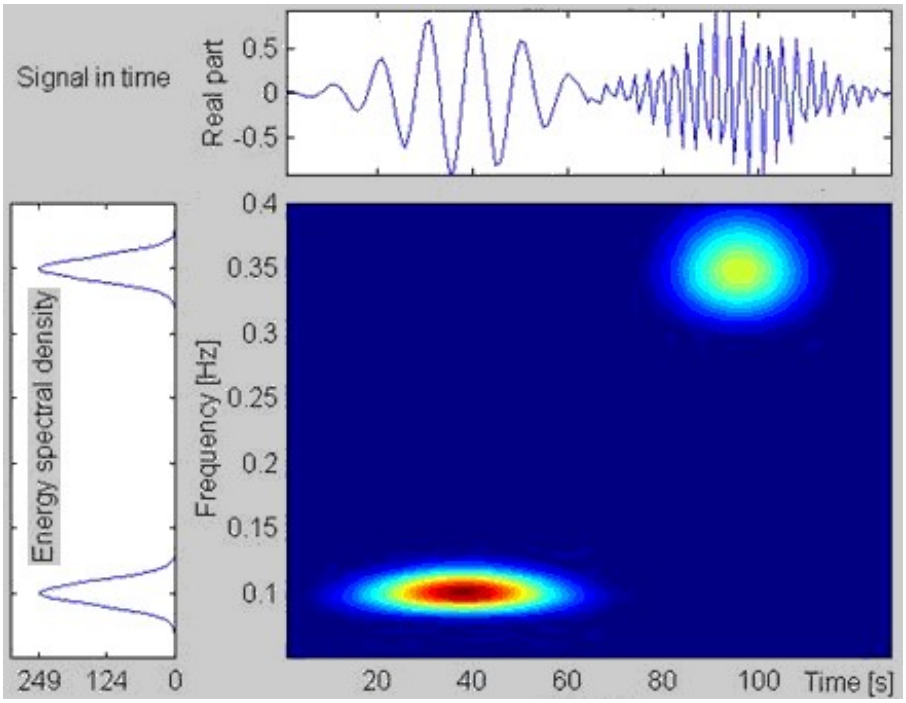


Рисунок 2.12 — Фурье преобразование (слева) и вейвлет-преобразование (справа) модулированного входного сигнала (сверху).

БПФ дискретное вейвлет-преобразование является быстрой линейной операцией, которая оперирует с вектором данных, длина которого есть целая степень двойки, преобразуя его в численно отличающийся вектор той же длины. Также по аналогии с БПФ вейвлет-преобразование обратимо и ортогонально обратному преобразованию. Если рассматривать в виде матричного представления, то обратное преобразование есть просто транспонирование матрицы преобразования. Поэтому и БПФ, и дискретное вейвлет-преобразование могут рассматриваться как поворот функционального пространства; как переход из пространственной или временной области, где базисными функциями являются единичные векторы (или дельта-функция в пределах континуума), в другую область. С этой точки зрения базисные функции также определяют как поворачивающие множители (коэффициенты поворота). Для БПФ эта другая область имеет базисные функции, являющиеся привычными для нас синусами и косинусами. Для вейвлет-преобразования базисными функциями являются некоторые более сложные функции, которые имеют несколько забавные названия: «материнские функции» и вейвлеты (wavelet – небольшая волна, всплеск, импульсоид). Конечно, число базисов не ограничено для функционального пространства, но в основном они не вызывают интереса с точки зрения цифровой обработки сигналов.

Что же делает интересными вейвлет-базисы? — Это то, что в отличие от синусов и косинусов отдельная вейвлет-функция вполне локализована в пространстве и в то же время, подобно синусам и косинусам, вейвлет-функции вполне локализованы по частоте, или, что более точно, по характеристической шкале.

Введем основные понятия. Вейвлет-преобразованием некоторой функции F называется множество функций вида

$$W(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} F(x)\psi^* \left(\frac{x-b}{a} \right) dx, \tag{2.25}$$

где ψ^* — комплексно-сопряженная анализирующая вейвлет-функция. Выражение 2.25 можно рассматривать как свертку сигнала с фильтром, зависящим от параметра a, называемым пространственным масштабом вейвлет-функции, и параметра b — параметра положения. Параметр пространственного масштаба определяется размером структур, которые мы хотим детектировать.

Преобразование характеризуется следующими основными свойствами:

1. это линейное преобразование;
2. это преобразование инвариантно к сдвигу:

$$f(x) \to f(x-u) \qquad W(a,b) \to W(a,b-u);$$

3. это преобразование инвариантно к растяжению:

$$f(x) \to f(sx) \qquad W(a,b) \to s^{-\frac{1}{2}} W(sa,sb).$$

Последнее свойство делает вейвлет-преобразование очень удобным для анализа иерархических структур. Это можно сравнить с математическим микроскопом, свойства которого не зависят от увеличения. Применив преобразование Фурье, перейдя в частотную область, получим:

$$\hat{W}(a,b) = \sqrt{a}\hat{f}(\omega)\hat{\psi}^*(a\omega).$$

То есть, при изменении масштаба a вейвлета, фильтр $\hat{\psi}^*(a\omega)$ только понижает частоту при сохранении своей формы. И с этой точки зрения вейвлет-преобразование можно рассматривать как анализ сигнала, проходящего через систему полосовых фильтров. Функция ψ должна удовлетворять необходимым условиям для применения формул реконструкции: для дифференцируемой функции это нулевое среднее значение. Аналогично преобразованию Фурье, применив обратное вейвлет-преобразование, получим исходную функцию.

Классический анализирующий вейвлет — это так называемая «мексиканская шляпа», которая определяется следующим выражением:

$$\psi(x) = (1-2x^2)\exp(-x^2). \tag{2.26}$$

Это вторая производная гауссианы; имеет два убывающих момента. С точки зрения обработки двумерных сигналов, например, изображений, вейвлет-анализ соответствует системе изображений с нерезким маскированием с последовательными переходами от мелких к более крупным масштабам. Результат может в совокупности рассматриваться как частотный анализ вблизи точки b и как пространственный анализ для масштаба a. Есть,

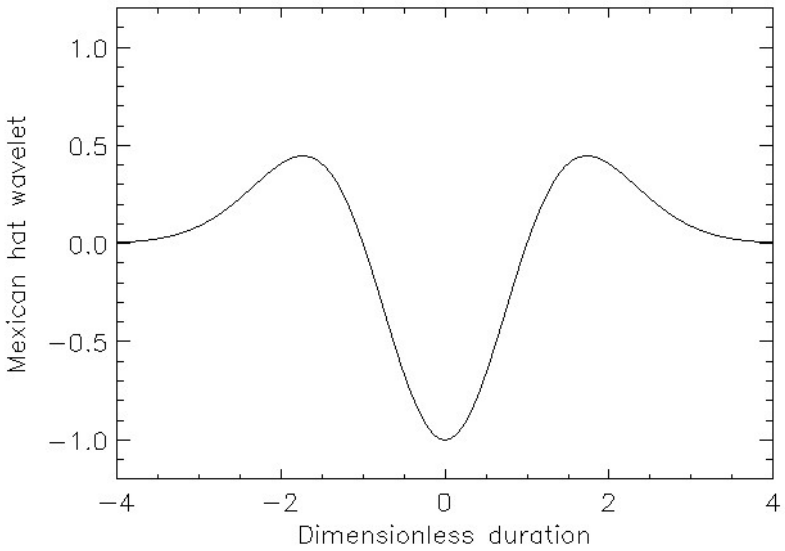


Рисунок 2.13 — Вейвлет "мексиканская шляпа"

по крайней мере, два пути введения вейвлет-преобразования для обработки сигналов: на основе дискретного представления выражения (2.25) и на основе метода, называемого анализом мультиразрешения (multiresolution analysis) — анализом с переменной разрешающей способностью. Для последнего разработано несколько способов реализации. Один из них предложен J.-L. Starck [4] и представляет собой алгоритм под названием «á trous». Этот способ хорош тем, что не требует существенных вычислительных ресурсов и на его основе можно реализовать эффективный итерационный механизм подавления шума, кроме того, он очень нагляден и позволяет наблюдать этапы обработки данных. Максимальное частотное разрешение сигналов определяется шагом (или частотой) выборки, который задает максимальное разрешение — наименьший масштаб $a_0 = 2^0$. Обозначив исходную выборку через $\{c_0(k)\}$, мы можем записать ее в виде представления исходного сигнала по базису $\varphi(x-k)$:

$$c_0(k) = \langle f(x), \varphi(x-k) \rangle.$$

Функция $\varphi(x)$ выбирается соответствующей низкочастотному фильтру, и для диадного разложения (с разрешением, понижающимся с кратностью 2 от одного масштаба к следующему) должна удовлетворять уравнению растяжения:

$$\frac{1}{2}\varphi\left(\frac{x}{2}\right) = \sum_n h(n)\varphi(x-n), \tag{2.27}$$

тогда представление исходного сигнала $f(x)$ с разрешением $a = 2^i$ может быть записано как

$$c_i(k) = \left\langle f(x), \frac{1}{2^i}\varphi\left(\frac{x-k}{2^i}\right)\right\rangle, \tag{2.28}$$

а для следующего уровня разрешения $a = 2^{i+1}$, соответственно

$$c_{i+1}(k) = \left\langle f(x), \frac{1}{2^{i+1}}\varphi\left(\frac{x-k}{2^{i+1}}\right)\right\rangle. \tag{2.29}$$

На основании 2.27 — 2.29 можно прийти к рекурсивной формуле:

$$c_{i+1}(k) = \sum_n h(n)c_i(k+2^i n), \tag{2.30}$$

где шаг между двумя последовательными точками свертки соответствует наилучшему масштабу. Далее, функция $\psi(x)$ также может быть задана в базисе $\{\varphi(x-k)\}$с коэффициентами представления $g(n)$:

$$\frac{1}{2}\psi\left(\frac{x}{2}\right) = \sum_n g(n)\varphi(x-n). \tag{2.31}$$

Тогда коэффициенты вейвлет-преобразования для масштаба $a = 2^{i+1}$ могут быть записаны как:

$$w_{i+1}(k) = \left\langle f(x), \frac{1}{2^{i+1}}\psi\left(\frac{x-k}{2^{i+1}}\right)\right\rangle. \tag{2.32}$$

Коэффициенты фильтра $h(n)$ полностью определяют базисную функцию $\varphi(x)$. Для этой цели разработаны компактные, гладкие, аппроксимируемые кубическим B-сплайном быстроспадающие фильтры. Поскольку $w_i(k)$ содержат детали сигнала, исчезающие между двумя аппроксимациями с соседними уровнями разрешения, то вполне корректно было бы принять, чтобы $G(\omega) = 1 - H(\omega))$, где $G(\omega)$ и $H(\omega)$ — преобразование Фурье для $g(n)$ и $h(n)$, соответственно. Это приводит к простому алгоритму вычисления коэффициентов $w_i(k)$ путем получения разности двух аппроксимаций сигнала с соседними уровнями разрешения попиксельно:

$$w_{i+1} = c_i(k) - c_{i+1}(k). \tag{2.33}$$

Для каждого масштаба i совокупность $\{w_i(k)\}$ принято называть вейвлет-плоскостью. Множество $W = \{w_0, w_1, \ldots, w_p, c_p\}$ есть вейвлет-преобразование, а исходная выборка может быть представлена в виде:

$$c_0(k) = c_p + \sum_{i=1}^{p} w_i(k), \tag{2.34}$$

где c_p — результат последнего сглаживания на уровне разрешения p. Для двумерного случая, каким является изображение, имеем:

$$c_0(x,y) = c_p(x,y) + \sum_{i=1}^{p} w_i(x,y). \tag{2.35}$$

На рис. 2.14 показан пример разложения изображения (двумерного сигнала) на вейвлет-плоскости. Такое представление сигнала очень эффективно для удаления из него шума методом жесткого порога [4].

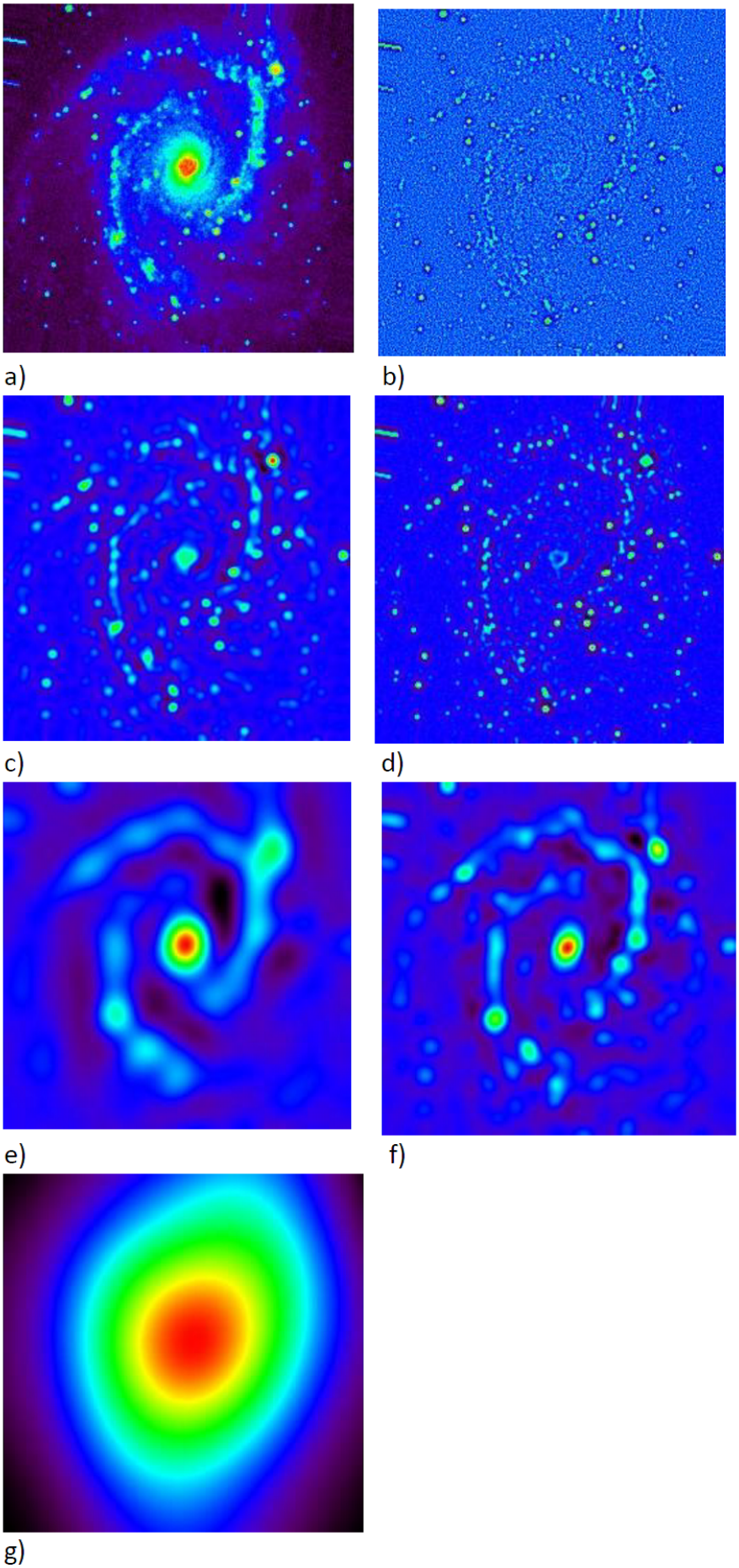


Рисунок 2.14 — Разложение сигнала на вейвлет-плоскости: a - исходный сигнал; b, c, d, e, f, соответственно, — w_1, w_2, w_3, w_4, w_5, a g — c_p.

Порог, разделяющий сигнал и шум, задается функцией:

$$\alpha(\sigma_j, w_j(x)) = \begin{cases} 1, & |w_j(x)| \geq k\sigma_j \\ 0, & |w_j(x)| < k\sigma_j. \end{cases} \tag{2.36}$$

Здесь σ_j — уровень (среднеквадратичное значение) шума, задающий порог, отделяющий *значимые* коэффициенты вейвлет-преобразования от *незначимых*. Тогда отфильтрованный сигнал получается следующим образом:

$$\tilde{I}(x) = c_p + \sum_{i=1}^{p} \alpha(\sigma_i, w_i(x)) w_i(x). \tag{2.37}$$

В этом выражении проявляется главное достоинство метода: мелкие, малозначимые детали сигнала, связанные с шумом, удаляются без искажения значимых деталей, в отличие от обычного сглаживания или низкочастотной фильтрации. Однако проблема заключается в том, как их определять. В зависимости от характера шума существует несколько подходов. Один из них заключается в среднеквадратичной оценке шума по вейвлет-плоскости, а затем шум удаляется итерационно, по алгоритму Ван Циттерта (Van Cittert). Суть этого алгоритма в следующем:

1. исходный сигнал подвергается вейвлет-преобразованию по изложенной методике, получаем набор вейвлет-плоскостей;
2. для каждой плоскости оценивается шум*;
3. по формуле 2.37 получаем отфильтрованный сигнал $\tilde{I}^{(n)}$, индекс n — номер итерации;
4. определяется невязка $E^{(n)} = I - \tilde{I}^{(n)}$;
5. пункты 1,2,3 выполняются для невязки $E^{(n)}$, то есть из невязки, в которой присутствует смесь шума и значимых элементов сигнала, выделяем последние;
6. добавляем эту невязку к ранее отфильтрованному сигналу: $\tilde{I}^{(n)} \Leftarrow \tilde{I}^{(n)} + \tilde{E}^{(n)}$;
7. $if \left| \frac{\sigma_{E^{(n-1)}} - \sigma_{E^{(n)}}}{\sigma_{E^{(n)}}} \right| > \varepsilon, \quad then \quad n = n + 1; \quad goto \quad 4.$

*Первоначальная оценка гауссова шума может быть получена, например, фильтрацией сигнала для заданного уровня разрешения путем сглаживания, или низкочастотной фильтрацией с прямоугольным окном. Затем отфильтрованный сигнал вычитается из исходного, и остаток считается шумом. Для более точной оценки можно воспользоваться алгоритмом, рассмотренным выше [4].

В завершение данного раздела следует отметить, что для «метода мультиразрешения» в качестве низкочастотного фильтра $h(k)$ необязательно использовать линейный фильтр, можно применить и нелинейный, например, медианный фильтр, размер окна которого также растягивается с понижением уровня разрешения.

Рассмотрев наиболее распространенные алгоритмы цифровой обработки сигналов, можно прийти к заключению, что базовыми вычислительными процедурами для этих алгоритмов являются вычисления вида 2.8, то есть операции свертки. Таким образом, для целей обработки сигналов процессор должен эффективно (с минимальными временны̀ми затратами) вычислять выражения вида 2.35. Более того, если входной сигнал $x(n)$ пред-

ставляет собой дискретно - временнỳю последовательность $x(nT)$ (T — период выборки), то выходной сигнал $y(n)$ должен формироваться в том же темпе, то есть за время, не превышающее T. Отсюда DSP - системы должны быть системами реального времени, причем время выполнения команды процессора должно быть не только малым, но и строго предсказуемым.

ГЛАВА 3

ПРОЦЕССОРЫ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ

Для построения систем цифровой обработки сигналов используются специализированные микропроцессоры — цифровые сигнальные микропроцессоры. Невозможность или неэффективность применения для решения задач цифровой обработки сигналов универсальных микропроцессоров связана, с одной стороны, с их низкой производительностью на указанных задачах, а с другой стороны — с их чрезмерной функциональной избыточностью для данных задач. Поэтому для систем ЦОС используют так называемые сигнальные микропроцессоры. К их особенностями относятся малоразрядная (40 и менее разрядов) обработка чисел с плавающей точкой, преимущественное использование чисел с фиксированной точкой разрядности 32 и менее, а также ориентация на несложную обработку больших массивов данных. Отличительной особенностью задач цифровой обработки сигналов является поточный характер обработки больших объемов данных в режиме реального времени, требующий высокой производительности процессора и обеспечения возможности интенсивного обмена с внешними устройствами. Соответствие данным требованиям достигается в настоящее время благодаря специфической архитектуре сигнальных процессоров и проблемно-ориентированной системе команд. Сигнальные процессоры обладают высокой степенью специализации. В них широко используются методы сокращения длительности командного цикла (характерные и для универсальных RISC-процессоров), такие как конвейеризация на уровне отдельных микрокоманд и команд, размещение операндов большинства команд в регистрах, использование скрытых (теневых) регистров для сохранения состояния вычислений при переключении контекста, разделение памяти команд и данных (гарвардская архитектура). В то же время для сигнальных процессоров характерным является наличие аппаратного умножителя, позволяющего выполнять умножение двух чисел за один такт. В универсальных процессорах умножение обычно реализуется в течение нескольких тактов, как последовательность операций сдвига и сложения.

Другой особенностью сигнальных процессоров является включение в систему команд таких операций, как умножение с накоплением (multiplier- accumulator) — MAC ($C := A \times B + C$), реверсирование порядка расположения битов адреса, операции над битами.

В сигнальных процессорах реализуется аппаратная поддержка программных циклов, кольцевых буферов, обработки прерываний. Реализация однотактного умножения, а также команд, использующих в качестве операндов содержимое ячеек памяти, обусловливает сравнительно низкие тактовые частоты работы этих процессоров.

Сигнальные процессоры различных компаний-производителей образуют два класса, существенно отличающихся по цене: более дешевые микропроцессоры обработки данных в формате с фиксированной точкой и более дорогие микропроцессоры, аппаратно под-

держивающие операции над данными в формате с плавающей точкой. Использование в сигнальной обработке данных в формате с плавающей точкой обусловлено несколькими причинами. Для многих задач, связанных с выполнением интегральных и дифференциальных преобразований, особую значимость имеет точность вычислений, обеспечить которую позволяет экспоненциальный формат представления данных. Алгоритмы компрессии, декомпрессии, адаптивной фильтрации в цифровой обработке сигналов связаны с определением логарифмических зависимостей и весьма чувствительны к точности представления данных в широком динамическом диапазоне значений. Работа с данными в формате с плавающей точкой существенно упрощает обработку, поскольку не требует выполнения операций округления и нормализации данных, отслеживания ситуаций потери точности и переполнения. Платой за комфорт является высокая сложность функциональных устройств, выполняющих обработку данных в формате с плавающей точкой, необходимость использования более сложных технологий производства микросхем и, как следствие, дороговизна микропроцессоров.

В настоящее время стал популярен и другой подход к получению высокой производительности. Большое количество транзисторов на кристалле может быть использовано для создания симметричной мультипроцессорной системы с более простыми процессорами, обрабатывающими целочисленные операнды. Примерами таких так называемых медийных процессоров служат чипы типа Mediaprocessor компании MicroUnity, Trimedia компании Philips, Impact Media Engine компании Chromatic Research, NVI компании Nvidia, MediaGx компании Cyrix. Эти процессоры создавались исходя из потребности обработки в реальном времени видео- и аудиоинформации в мультимедийных ПК, игровых приставках, бытовых радиоэлектронных приборах. Ввиду более простой схемотехники по сравнению с универсальными сигнальными процессорами, стоимость медийных процессоров достаточно низкая (порядка $100), а значение показателя производительность/стоимость на 2 – 3 порядка больше. Пиковое значение производительности медийных процессоров составляет несколько миллиардов целочисленных операций в секунду. К наиболее крупным производителям сигнальных микропроцессоров относятся компании Motorola, Texas Instruments, Analog Devices, Lucent Technologies. Каждая из указанных компаний выпускает целый спектр устройств, ориентированных на решение широкого круга задач. При выборе микропроцессора ЦОС для реализации конкретного проекта необходим учет многих параметров. Использование продукции той или иной компании во многом определяется предпочтениями разработчиков, однако следует учитывать определенные преимущества каждого микропроцессорного семейства.

3.1 Архитектура сигнальных процессоров

Практика использования компьютеров в последние десятилетия определила две наиболее широкие области применения: (1) *манипуляции с данными* — текстовые процессоры, организация и управление базами данных, и (2) *математические вычисления*, используемые в науке, технике и цифровой обработке сигналов. Все процессоры могут работать

Типовые приложения	**Обработка текстов, управление базами данных, электронные таблицы, операционные системы (банковские, бухгалтерские…)**	**Цифровая обработка сигналов, управление движением, моделирование, науке и технике и т.д.**
Основные операции	**Перемещение данных:** $A \Rightarrow B$. **Тестирование значений:** ***(If A=B then ...)***	**Сложение:** ***(A+B=C)*** **Умножение:** ***(A×B=C)***

Рисунок 3.1 — Области применения и основные процедуры

в обеих областях, однако трудно (дорого) сделать прибор, оптимальный для работы везде. Нужно преодолеть много технических противоречий в размерах команд, обработке прерываний.

Одним из наиболее узких мест в реализации DSP алгоритмов является обмен информацией с памятью. Сюда относятся данные, такие как выборки входного сигнала и коэффициенты фильтра, а также команды программы. Например, операция умножения двух чисел, размещенных где-либо в памяти, требует выборки трех двоичных величин из памяти: перемножаемые числа плюс команда, определяющая действия. На рис. 3.2a представлена наиболее простая схема выполнения, реализуемая традиционными процессорами. Это архитектура Фон Неймана; названа в честь блестящего американского математика Джона Фон Неймана (1903-1957). Архитектура Фон Неймана состоит из единственной памяти и единственной шины для обмена данными между памятью и центральным процессором (CPU). Перемножение здесь требует, по крайней мере, трех тактовых циклов: по одному для передачи каждой из величин по шине из памяти в CPU. При этом не учитывается время передачи результата обратно в память, поскольку предполагается, что он остается в CPU для дальнейшей обработки (например, суммирование произведений в алгоритмах КИХ-фильтров). Архитектура Фон Неймана вполне удовлетворительна, если вас устраивает выполнение всех требуемых этапов последовательно. Но если требуется очень быстрая обработка, то возникает необходимость в другой архитектуре (за которую придется заплатить определенную цену вследствие увеличения сложности). Другая архитектура — это Гарвардская архитектура, показана на рис. 3.2b и названа в честь работ, выполненных в Гарвардском университете в 40-х годах 20-го века под руководством Ховарда Айкена (1900 - 1973). Как показано на рисунке, Айкен ввел два отдельных блока памяти для данных и команд программы с раздельными шинами для каждого блока. Шины действуют независимо друг от друга, команды и данные могут выбираться из памяти одновременно. До настоящего времени DSP используют эту двухшинную архитектуру.

На рис. 3.2c представлен следующий этап усовершенствования архитектуры — Супер-гарвардская архитектура. Этот термин был введен компанией Analog Devices для описания своих DSP семейства ADSP-2106x и нового семейства ADSP-211xx. Они имеют название SHARC® (аббревиатура от Super Harvard ARChitecture). SHARC DSP опти-

мизированы для ЦОС по десяткам направлений, но есть два наиболее важных элемента: командный кэш и контроллер ввода/вывода. При перемножении двух чисел требуется чтение из памяти данных двух сомножителей и только одной команды из памяти программ. Следовательно, шина, связанная с памятью данных, более загружена, чем шина программной памяти. Как же командный кэш разрешает эту проблему. Во-первых, размещением части «данных» в программной памяти. Например, можно разместить коэффициенты (ядро) КИХ-фильтра, оставляя входной сигнал в памяти данных. Эти перемещенные данные

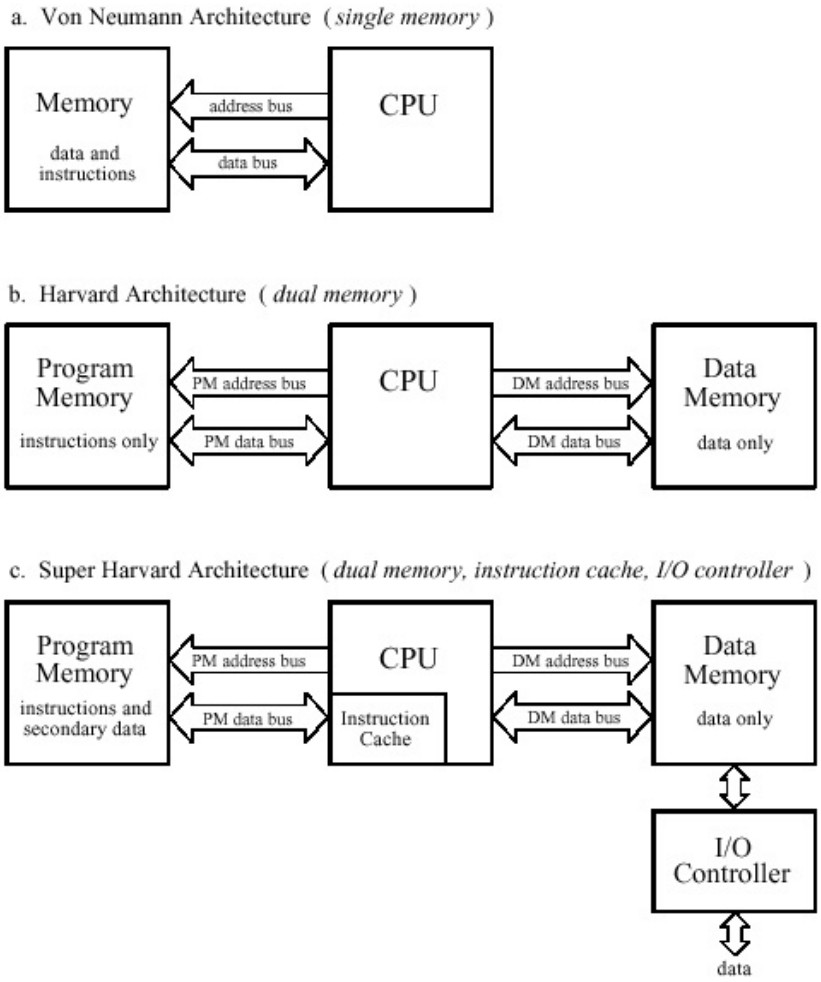


Рисунок 3.2 — Архитектуры процессоров: (a) – Фон-Неймана, (b) – гарвардская, (c) – супер-гарвардская

называются на рисунке «вторичными данными» (secondary data). На первый взгляд кажется, что это не разрешает ситуацию, поскольку перегруженной становится другая шина — шина программной памяти. И это так, если выполняются произвольные команды. Однако DSP - алгоритмы в большинстве своем имеют циклы, это означает, что определенный набор команд многократно в цикле вызывается из памяти программ в CPU. И в этом случае оказывается эффективным командный кэш — небольшая память в CPU, вмещающая 32 последние команды. Проходя по циклу в первый раз, команды должны пройти через шину программной памяти, это приводит к снижению производительности вследствие конфликта с коэффициентами фильтра, которые передаются по этому же пути. Однако

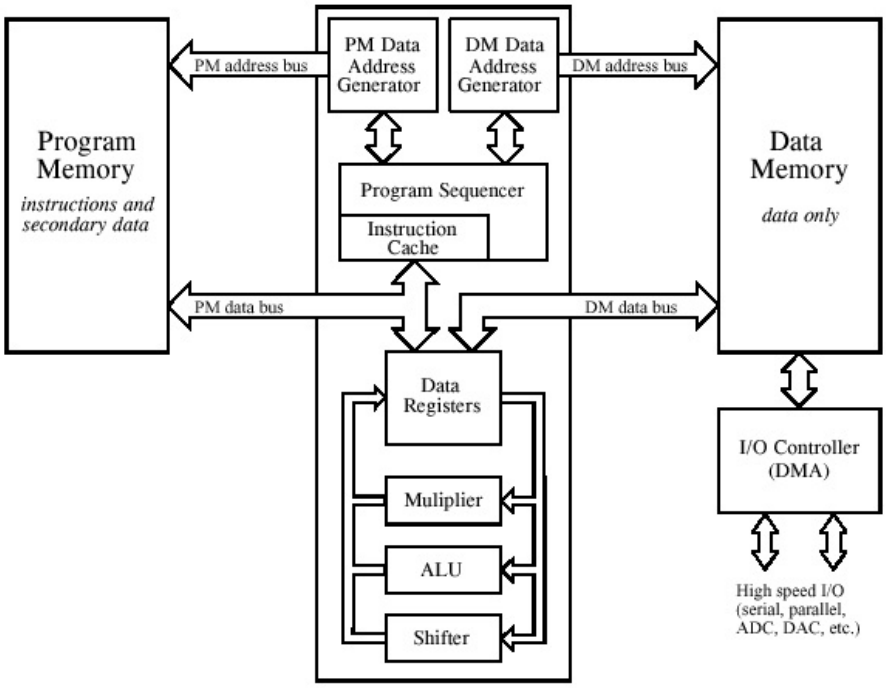


Рисунок 3.3 — Типичная DSP – архитектура с параллельным выполнением основных операций

при последующих прохождениях цикла команды выбираются уже из командного кэша. А это означает, что вся информация может быть доставлена в CPU за один тактовый цикл.

На рис. 3.3 представлена более детальная схема SHARC, где показан контроллер ввода/вывода, связанный с памятью данных. Он обеспечивает ввод и вывод сигналов в DSP системе. Для SHARC DSP предусмотрены как последовательные, так и параллельные порты. Это чрезвычайно скоростные связи, например, при 40 Мгц тактовой частоте два последовательных порта работают на скорости 40 Мбит/сек каждый, кроме того, шесть параллельных портов обеспечивают скорость передачи данных 40 Мбайт/сек каждый, а когда используются все шесть параллельных портов вместе, то скорость ввода/вывода может достигать 240 Мбайт/сек! Очень важно то, что эти скоростные каналы обеспечивают передачу данных в память непосредственно (минуя регистры CPU в режиме прямого доступа) и независимо, благодаря двухпортовой памяти. Поэтому для операций ввода выборки входного сигнала и выдачи выходного не требуется ни одного дополнительного тактового цикла CPU. Основные шины (памяти данных и программной памяти) доступны извне, обеспечивая дополнительный интерфейс для внешней дополнительной памяти и периферийных устройств. Такой высокоскоростной ввод/вывод является ключевой характеристикой DSP. Цель — ввести данные, выполнить математические преобразования и выдать данные до прихода следующей выборки. Некоторые DSP имеют встроенные АЦП, ЦАП для аналогового ввода/вывода, но все они обеспечены последовательными или параллельными портами для подключения преобразователей.

Обратим внимание на то, что внутри CPU (рис. 3.3) имеются также адресные генераторы для обоих блоков памяти — Data Address Generator (DAG). Они формируют

адреса операндов, расположенных в DM и PM. В простых микропроцессорах эту задачу выполняет счетчик команд. Применение DAG позволяет управлять кольцевыми буферами без дополнительных затрат тактовых циклов. Кроме того, DAG позволяет выполнять процедуру реверсирования битов адреса операндов — необходимую часть алгоритма БПФ.

Регистры данных CPU используются для тех же целей, что и в традиционных микропроцессорах: для хранения промежуточных результатов вычислений, подготовки данных для арифметических блоков, служат буферами при передаче данных, хранят флаги управления программой и т.д. Эти регистры также могут использоваться как счетчики и для организации циклов, но SHARC DSP имеет специальные аппаратные регистры для выполнения этих и других функций.

Математическая обработка выполняется в трех блоках: умножителе, арифметико-логическом блоке (ALU) и сдвигателе барабанного типа. Особенностью SHARC DSP является то, что все эти устройства могут работать параллельно.

Другой интересной особенностью является наличие скрытых (теневых) регистров, копирующих набор всех ключевых регистров CPU. Они используются для быстрого переключения контекста при быстрой обработке прерываний. Переключение контекста выполняется за один тактовый цикл. По завершении процедуры обработки прерываний восстановление регистров происходит также быстро. В обычных микропроцессорах для этого используется стек и требуется, соответственно, немало тактовых циклов. От архитектуры самой общей функциональной схемы перейдем к более подробному рассмотрению главных элементов DSP.

3.2 Основные элементы архитектуры DSP

Рассмотрим их на примере процессоров от Analog Devices, подробно описанных в [5].

3.2.1 Умножитель-аккумулятор (Multiplier-Accumulator – MAC)

MAC представляет собой совокупность множительного устройства и накапливающего сумматора. Сочетание множительного устройства (умножителя) с аккумулятором (сумматором результата) эффективно для выполнения основной процедуры обработки сигналов: вычисления выражений типа $\sum b(n)x(n-k)$.

Каковы же особенности умножителя-аккумулятора в DSP?

- Выполнение операции умножение – накопление за один цикл;
- достаточное число охранных битов (битов расширения) в аккумуляторе;
- детектирование переполнения до его наступления;
- округление без смещения;
- арифметика насыщения;
- внутренние магистрали обратной связи, предназначенные для сокращения временных потерь в циклах и ускорения вычислений общих «DSP примитивов».

В упрощенном виде структурная схема MAC приведена на рис. 3.4 и демонстрирует то, как выполняются повторяющиеся DSP процедуры независимо от других вычислительных элементов.

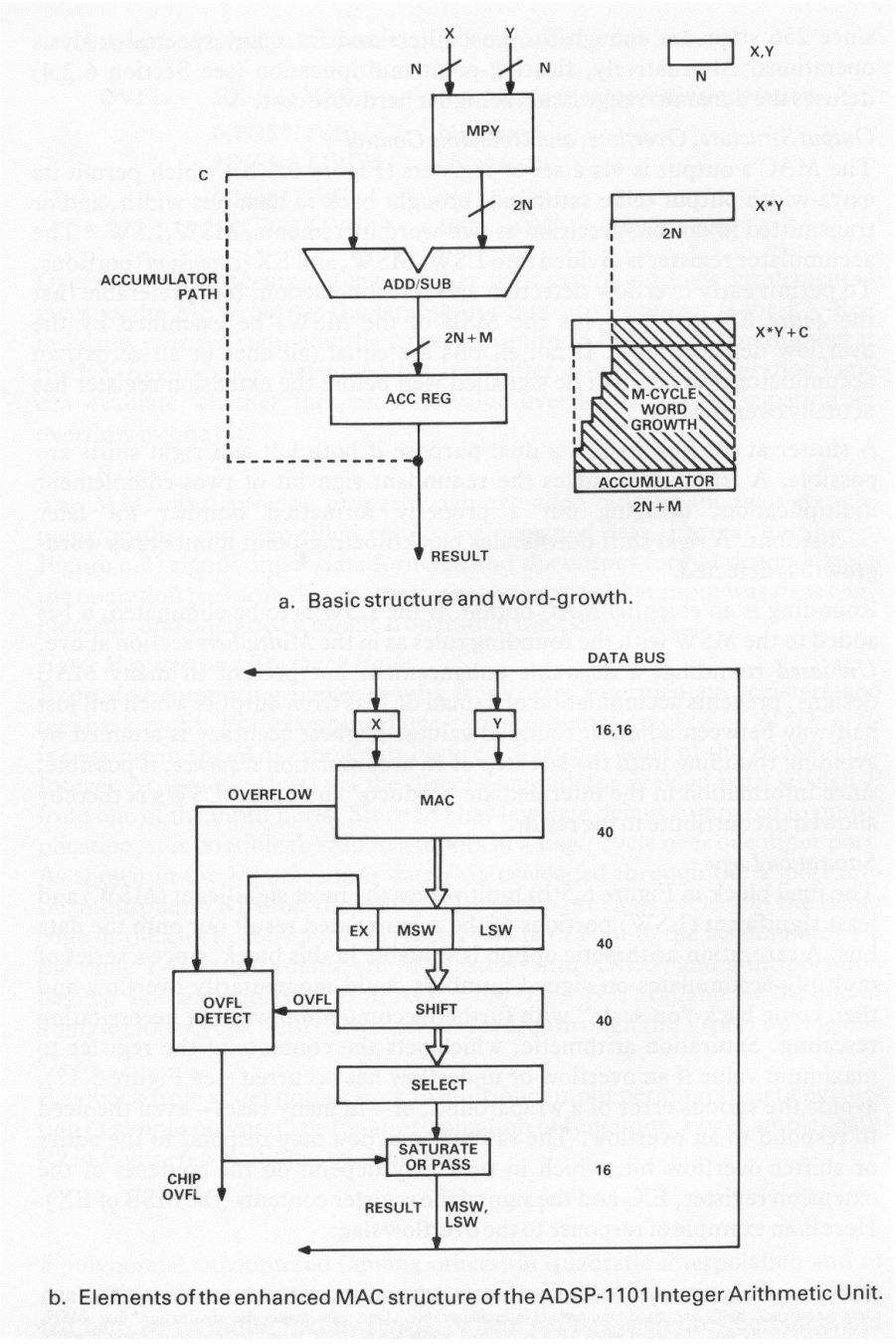


Рисунок 3.4 — Структурная схема MAC

Все 2N бит произведения поступают из умножителя на сумматор/вычитатель. Обратная связь по магистрали C через другой вход сумматора обеспечивает действие аккумулятора:

$$R(n+1) = X(n+1) \times Y(n+1) \pm R(n),$$

где результат предыдущих вычислений $R(n)$ возвращается для суммирования по магистрали C. Выход сумматора ADD/SUB имеет увеличенную на M битов разрядность, так что, по крайней мере, $2M$ повторяющихся MPY/ACC операций могут выполняться без переполнения (рис. 3.4a). Например, 40-бит аккумулятор с 32-бит умножителем позволяет выполнить 256 командных циклов без переполнения, что достаточно для большинства фильтров и многих приложений спектрального анализа. Выход MAC осуществляется через систему регистров (рис. 3.4b), которые реализуют «насыщение» при превышении разрядности результата операции, приводят результат к формату шины данных и/или передают число двойной точности в виде двух слов MSW, LSW (most - significant word, least - significant word = старшее слово, младшее слово). Для этого регистр аккумулятора разделен на части: LSW, MSW и EX – (расширение). Для детектирования предстоящего переполнения весь регистр EX и старший бит MSW тестируются логикой детектора переполнения. Сдвигатель предназначен для двух случаев: левый сдвиг устраняет избыточные знаковые биты (для чисел в дополнительном коде), возвращая сформатированное для дальнейших вычислений значение. Правый сдвиг понижает масштабы чисел в блоковых операциях с плавающей точкой. Логика «насыщения» преобразует данные, полученные в результате вычислений и выходящие за разрядную сетку (за формат) шины данных (см. рис. 3.4b). Если имеется переполнение, логика «насыщения» устанавливает в регистр устройства максимальное значение (или минимальное, в зависимости от знака). Функция насыщения аналогична насыщению аналоговых сигналов, например, в линейных усилительных схемах и предотвращает серьезные ошибки в работе DSP систем. Расширение вычислительных возможностей MAC основывается на применении дополнительных промежуточных регистров и трактов обратной связи. Например, вычисления типа $Z = M \times D + B$ выполняются в MAC, изображенном на рис. 3.5, путем прямой загрузки аккумулятора через один из входных портов по шине P.

Поскольку ввод/вывод имеет задержки в пределах задержки нескольких вентилей, соизмеримые с временем выполнения умножения, то реализуется возможность считывания двух входных операндов в одном цикле и через один вход. Как показано на рисунке, входные данные D загружаются через Y-порт в умножитель, в регистр $YREG$, по спаду предыдущего тактового импульса. По фронту импульса константа B загружается в аккумулятор по магистрали P через тот же Y-порт, а коэффициент M загружается в X-порт и выполняется умножение. Результат: $ACC = M \times D + B$ формируется в конце цикла. Обратная связь с сумматора на вход умножителя позволяет эффективно вычислять простые полиномы для целей аппроксимации функций. Например, магистраль $R2$ (рис. 3.5) способствует вычислениям вида $Z = 1 + aY + bY^2$ — полином, встречающийся в квадратичной интерполяции и при вычислении квадратных корней.

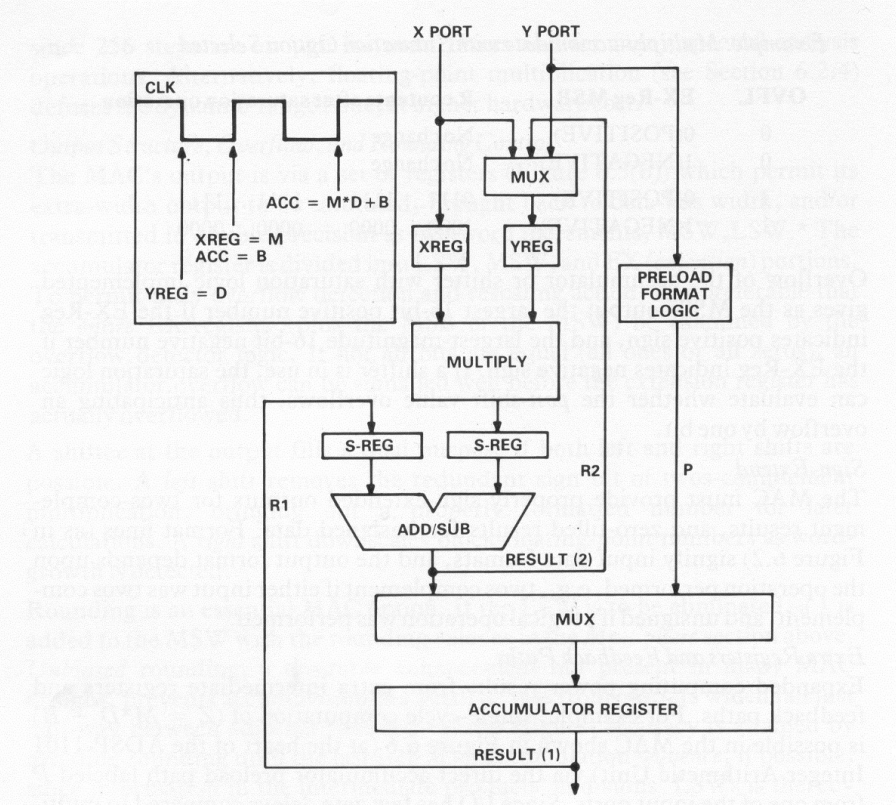


Рисунок 3.5 — Упрощенная схема MAC целочисленной арифметики с трактами обратной связи и прямой загрузкой аккумулятора.

3.2.2 Арифметическое и логическое устройство — ALU (Arithmetic Logic Unit)

В цифровых сигнальных процессорах *ALU*, в основном, такое же, что и в универсальных микропроцессорах, и выполняет обычный набор команд. Отличает возможность выполнения команд за один такт и конвейеризация повторяющихся операций, но с сохранением прозрачности для сквозных потоков команд. Кроме того, DSP *ALU* тесно связаны с масштабирующими устройствами, такими как сдвигатели, которые для сохранения динамического диапазона поддерживают требуемый масштаб данных.

Каким же требованиям должен удовлетворять ALU DSP? — Обеспечивать:

- достаточную точность с возможностью наращивания разрядности для высокоточных вычислений;
- выполнение арифметических функций: сложение, вычитание, сложение с переносом, вычитание с займом, формирование абсолютного значения;
- выполнение логических функций: AND, OR, XOR, логическое отрицание;
- формирование флагов: нуль, равно, больше чем, меньше чем, перенос. . . ;
- возможность выполнения деления;
- эффективная пересылка данных и вычисления: пересылки данных плюс арифметика в едином командном цикле;
- возможность загрузки двух операндов в *ALU* в каждом цикле;

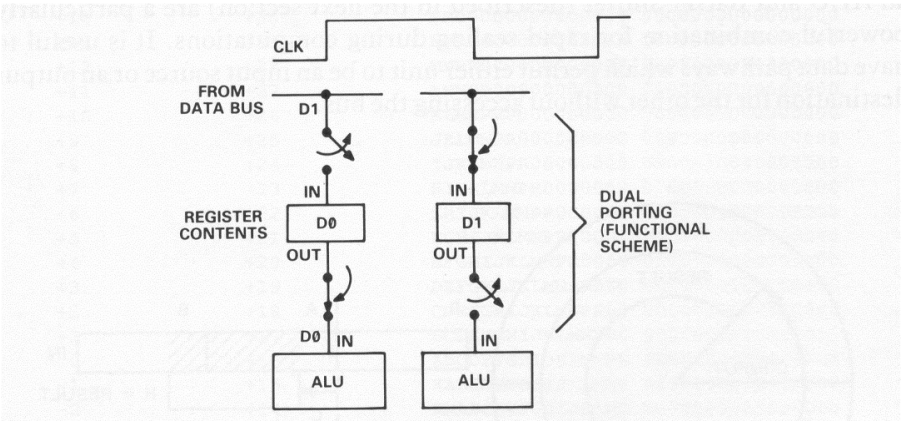


Рисунок 3.6 — Двухпортовая схема загрузки

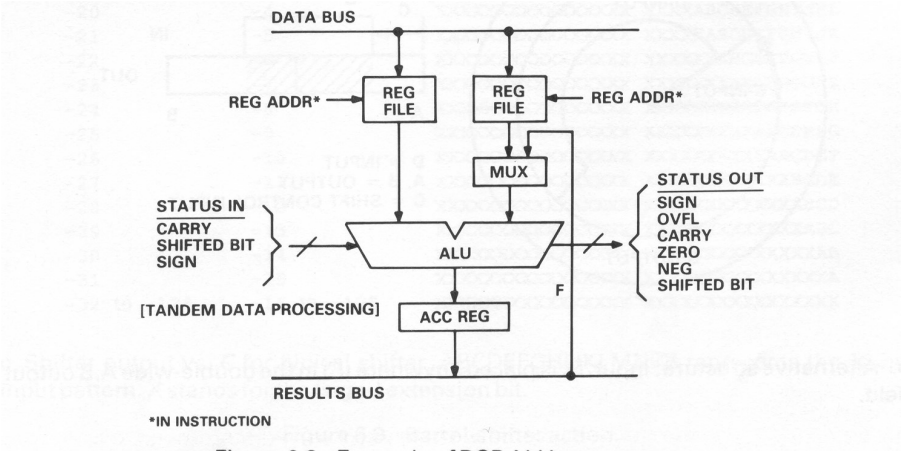


Рисунок 3.7 — Пример структуры ALU DSP.

- достаточный размер регистровых файлов: двойной набор для переключения контекста (сохранения данных на время прерываний), или наличие регистрового файла для быстрого доступа и хранения промежуточных результатов;
- выполнение условных операций: например, сложение плюс сдвиг, если установлен флаг;
- наличие необходимых магистралей обратной связи: выход на вход (аккумулятор), и если устанавливается связь со сдвигателем, то входы ALU связываются с выходом сдвигателя, или входы сдвигателя с выходом ALU;
- опции конвейерного и сквозного проходов на входные регистры;
- тесное взаимодействие со сдвигателем для операций масштабирования;
- возможности выполнения вычислений с двойной точностью с минимальными временными затратами.

Схема, позволяющая выполнять загрузку данных и арифметические команды в едином цикле, представлена на рис. 3.6. Здесь адресуемые регистровые файлы, которые обеспечивают быстрый доступ к повторно используемой информации и к промежуточным результатам без обращения к памяти. Гибкие внутренние магистрали данных — им присущи свойства ALU в DSP. Например, выход аккумулятора может быть входом для следующего цикла ALU благодаря магистрали F (см. рис. 3.7).

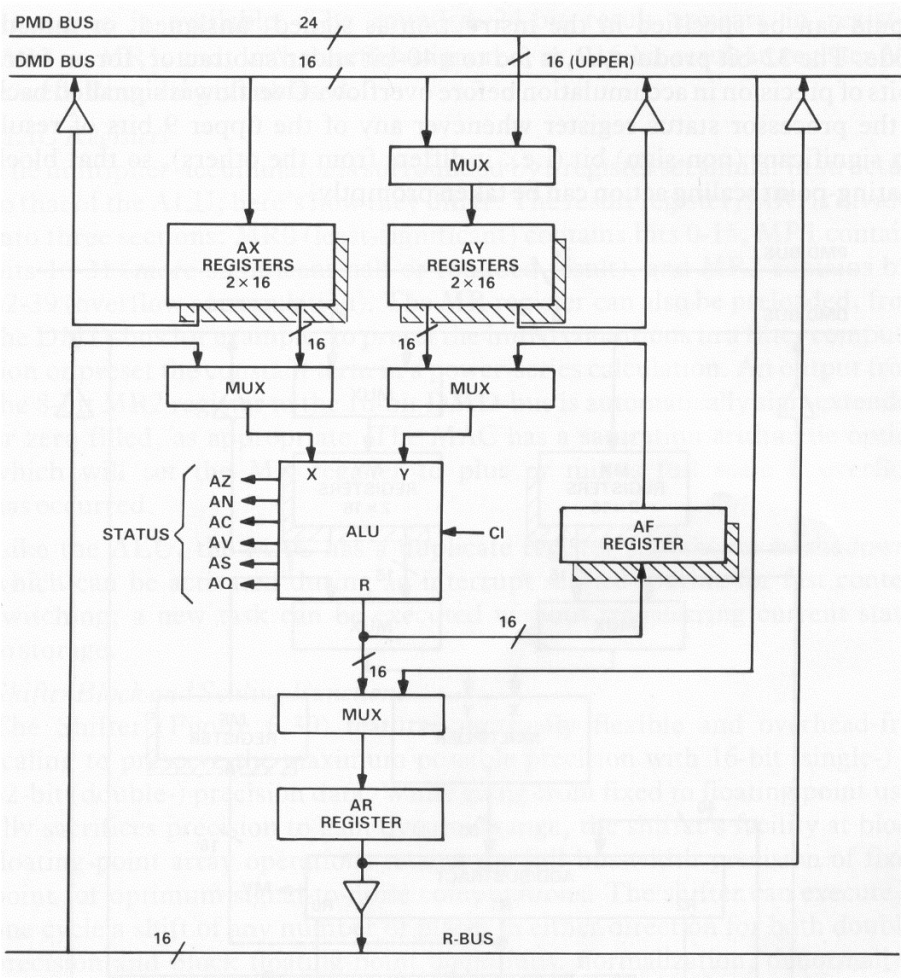
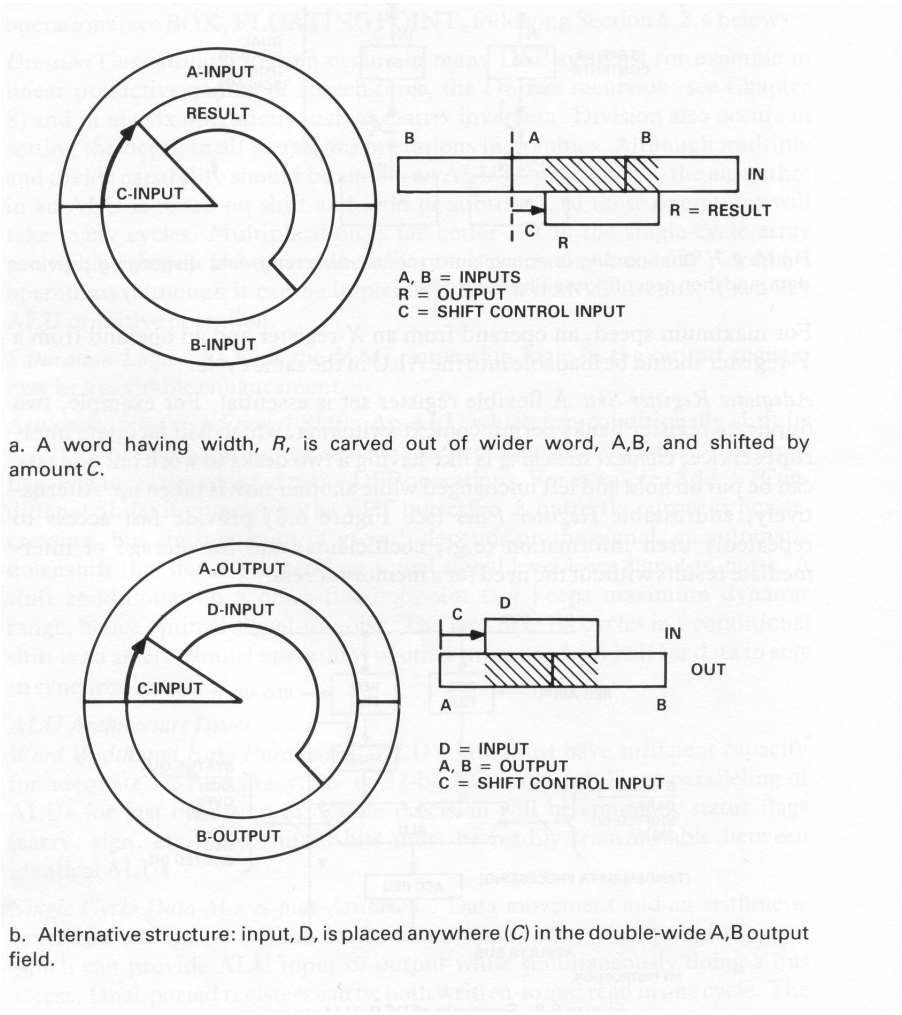


Рисунок 3.8 — Блок-схема ALU ADSP2100

Блок-схема ALU ADSP-2100 приведена на рис. 3.8. Здесь есть очень гибкий набор регистров, обрамляющий собственно ALU. (Аналогичные наборы обрамляют MAC и сдвигатель). Конфигурация регистров предоставляет возможность компоновки одноцикловых операций. Каждый из регистров $X0, X1, Y0, Y1$ обладает двумя связанными или независимыми 16 бит значениями. Y–вход может иметь доступ либо к шине данных памяти данных, либо к шине данных программной памяти, и таким образом, может обрабатывать информацию, хранимую в программной памяти. Любой регистр может выдавать данные на магистраль данных. При этом регистры снабжены двумя независимыми выходами, и таким образом, один регистр может снабжать данными ALU, в то время как содержимое других возвращается на шину DMD. Непосредственная обратная связь результата ALU устанавливается через другой Y-вход и регистр AF, то есть связь без прохождения через входные регистры ALU. Выходы ALU через регистр AR подключаются либо к внутренней шине R, либо возвращаются на шину DMD. Регистр AR может реализовывать функцию, называемую «арифметикой насыщения».

3.2.3 Сдвигатель (Shifter) и управление масштабированием чисел

Сдвигатель в DSP предназначен для масштабирования чисел с целью предотвращения переполнения и потери значащих разрядов, для выполнения преобразований чисел



a. A word having width, *R*, is carved out of wider word, A,B, and shifted by amount *C*.

b. Alternative structure: input, D, is placed anywhere (*C*) in the double-wide A,B output field.

Рисунок 3.9 — Устройство сдвига (сдвигатель) барабанного типа

с фиксированной точкой в числа с плавающей точкой и наоборот. В отличие от микропроцессоров общего назначения, где сдвиг осуществляется на один бит за цикл, в DSP сдвигатель должен иметь способность выполнять сдвиг слова на заданное количество бит за один цикл.

Можно выполнять сдвиги с помощью множительного устройства (умножителя), умножая на 2, 4, $\ldots 2^n$, однако полный комплект операций сдвига лучше выполнять сдвигателем, специально спроектированным для этих целей.

Работа сдвигателя барабанного типа (barrel shifter) схематически показана на рис. 3.9a. Здесь отображается часть длинного входного слова A, B на выходное слово R, с разрядностью, равной разрядности шины, и величиной сдвига, устанавливаемой управляющим словом C. Другое использование данной топологии — это когда входное слово D сдвигается в расположение выходного слова A, B, имеющего удвоенную разрядность. Стартовая точка устанавливается управляющим словом по входу C (рис. 3.9b). Сдвигатель выполняет арифметические и логические сдвиги. Арифметические сдвиги масштабируют числа (выполняя сдвиги вправо и влево), как показано на рис. 3.10.

В универсальных микропроцессорах, если старший бит числа (MSB – most significant bit) не равен S, то устанавливается флаг переполнения. А в DSP для вы-

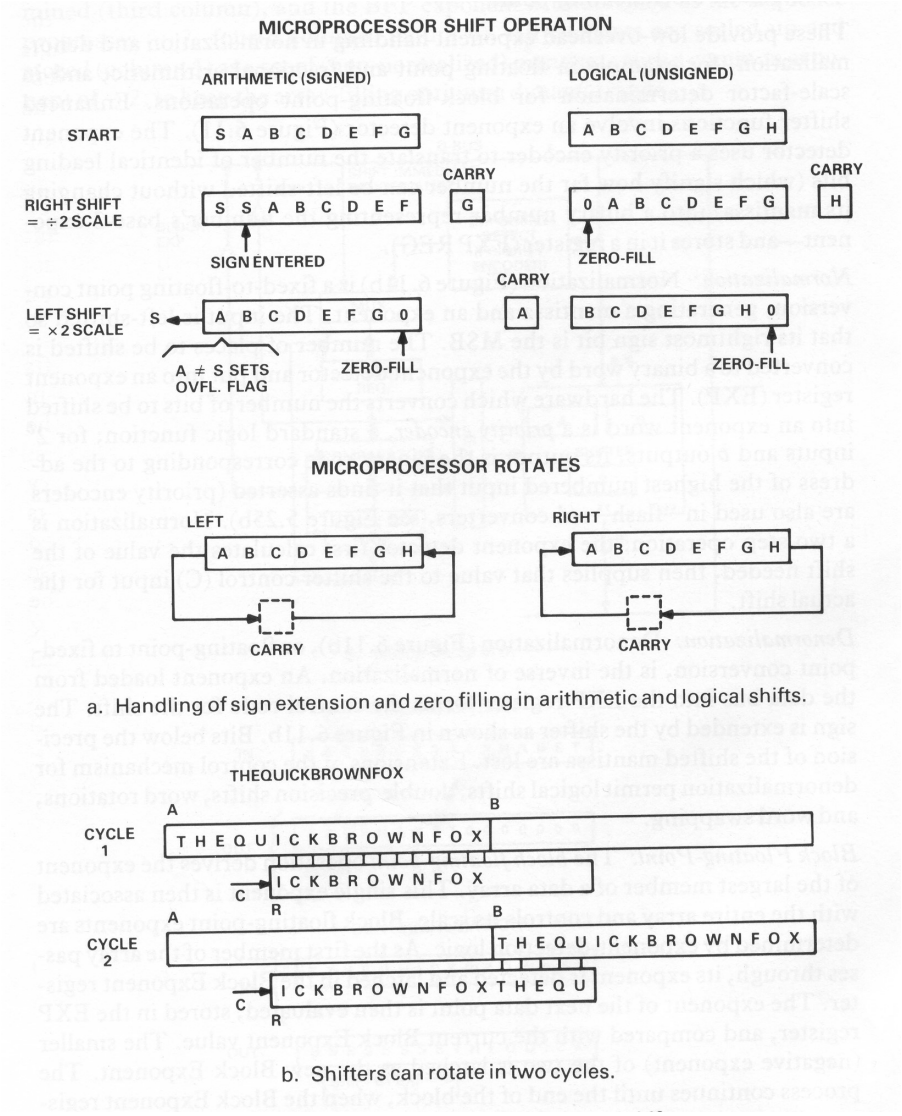


Рисунок 3.10 — Сдвиги в обычном микропроцессоре и сдвигателе

ходного регистра R сдвигатель имеет регистр врѐменного хранения состояния тестового бита (test - bit status), который сигнализирует о надвигающемся переполнении. Таким образом, в следующем цикле могут быть предприняты действия, снимающие угрозу переполнения.

Логические сдвиги работают с числами без знака (например, в операциях маскирования). При левом или правом сдвигах младшие (или старшие) биты заполняются нулями.

В DSP обычно используют сдвигатель барабанного типа (рис. 3.9), который благодаря своей архитектуре позволяет осуществлять циклические сдвиги на величину от одного до b бит всего за два такта, загружая слово сначала во входной регистр A, и затем в регистр B.

3.2.3.1 Операции сдвига

К основным операциям относятся:

- n – разрядные арифметические и логические сдвиги;
- циклические сдвиги слов (вращение и слияние);

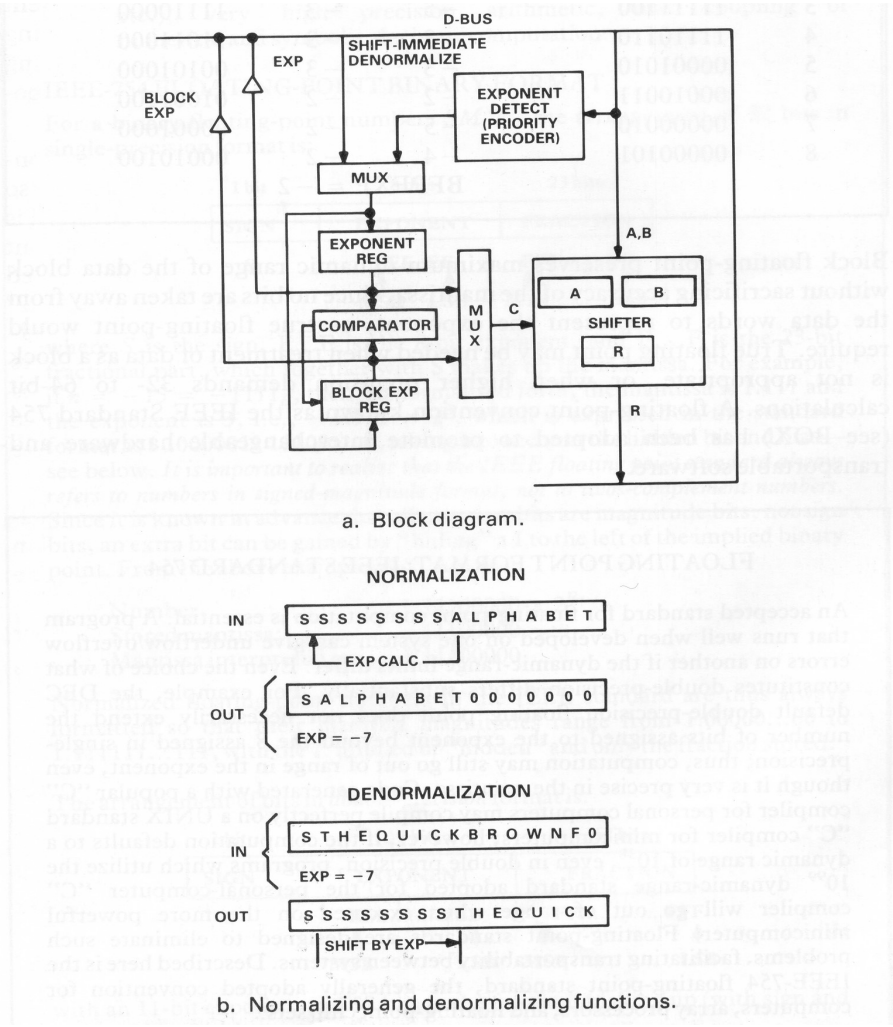


Рисунок 3.11 — Сдвигатель с детектированием порядка числа и блоковыми преобразованиями с плавающей точкой.

- тестирование бита, установка бита, очистка бита.

Здесь непосредственно возвращается значение, сдвинутое на управляющую величину C.

Сдвигатель DSP имеет также и расширенные функции, к которым относятся:

- нормализация,
- денормализация,
- блоковые операции с плавающей точкой.

Это реализуется благодаря схеме (рис. 3.11a). Основным элементом здесь является детектор порядка числа. Он представляет собой приоритетный шифратор и информирует о наличии идентичных старших разрядов слова, и, тем самым, определяет количество разрядов, на которое может быть сдвинуто слово влево без изменения мантиссы.

Нормализация (рис. 3.11b) — есть преобразование числа с фиксированной точкой в число с плавающей точкой, формирующее мантиссу и порядок. Входное значение сдвигается влево так, чтобы самый правый знаковый бит занял позицию MSB. Количество разрядов, на которое нужно сдвинуть число, преобразуется в двоичное слово (код) с помощью «детектора порядка» и записывается в регистр порядка EXP. Схема, которая преобразует число сдвигаемых битов в порядок, представляет собой приоритетный шифратор,

Example of Block-Floating-Point Exponent Determination and BFP Scaling

Array Index	Input Data S	Stored Exponent EXP	BFP	BFP-Adjusted Output Data S
1	00001001	−3	−3	00100100
2	11110001	−3	−3	11000100
3	11111100	−5	−3	11110000
4	11110110	−3	−3	11011000
5	00001010	−3	−3	00101000
6	00010011	−2	−2	01001100
7	00000010	−5	−2	00001000
8	00000101	−4	−2	00010100
		BFPEXP = −2		

Рисунок 3.12 — Пример определения порядка для блоковой операции с плавающей точкой.

то есть тривиальную комбинационную логическую схему. Для $2b$ входов и b выходов такой схемы выходное значение будет представлять собой двоичный код, соответствующий номеру (адресу) наивысшего установленного разряда.

Нормализация — двухшаговая операция: во-первых, детектор порядка вычисляет требуемое значение сдвига, а затем это значение применяется для управления сдвигателем (величина C).

Денормализация (или преобразование числа с плавающей точкой в число с фиксированной точкой) — есть операция обратная нормализации. Порядок числа, загруженный через шину данных $D - bus$ в регистр порядка EXP, формирует управляющий код для сдвига. Знак числа повторяется, как показано на рис. 3.11b. Младшие биты мантиссы, а следовательно и точность, при этом теряются. Дополнительные функции механизма управления денормализацией позволяют выполнять логические сдвиги, сдвиги для слов двойной длины (double-precision), циклические сдвиги, перестановку байтов в слове (свопинг).

3.2.3.2 Блоковые преобразования с плавающей точкой

Здесь определяется порядок наибольшего числа из некоторого массива данных, и этот единственный порядок числа затем связывают со всем массивом (блоком) и управляют его масштабом. Для этих операций порядок определяется с помощью специальной логики (рис. 3.11a). При прохождении первого элемента массива его порядок определяется и «защелкивается» в регистре — $BLOCK\ EXP\ REG$. Далее определяется порядок следующего элемента массива и помещается в регистр $EXP\ REG$, затем сравнивается с текущей величиной, находящейся в $BLOCK\ EXP\ REG$. Наименьшее из двух этих значений (поскольку порядок — величина отрицательная) запоминается в $BLOCK\ EXP\ REG$, как новый блоковый порядок. И так далее до конца массива, пока не будет найдено наименьшее значение. Пример определения блокового порядка приведен в таблице (рис. 3.12).

Очевидно, что применение блоковых преобразований с плавающей точкой возможно только тогда, когда данные представляют собой достаточно однородный по масштабу массив. Сигналы (или их фрагменты), весовые коэффициенты обычно удовлетворяют этому условию. В других случаях используют непосредственные преобразования с плавающей точкой.

3.2.4 Генератор адреса данных DAG (Data Address Generator)

Генератор адреса данных предназначен для формирования адресов операндов, расположенных в памяти данных DM и в памяти программ PM.

Обобщенная блок-схема DAG приведена на рис. 3.13. DAG имеет один или более выходных регистров с возможностью записи и чтения и содержащих числа, указывающие на расположение данных и констант (коэффициентов) в памяти. Он обеспечивает быструю и гибкую адресацию, весьма существенную в алгоритмах цифровой обработки сигналов, которые обычно требуют несколько операций пересылок, записи и чтения с элементами обработки в каждом командном цикле. Зачем нужны специальные формирователи адреса? - Примером может служить сканирование массива данных и генерация адресов для алгоритмов быстрого преобразования Фурье — одного из наиболее широко применяемых в цифровой обработке сигналов. Здесь при перемещении задач формирования адресов из ALU в специальное устройство существенно повышается скорость обработки.

Сканирование массива данных относится к наиболее универсальным операциям, используемым в алгоритмах цифровой обработки сигналов. Не все «сканы» являются последовательными, поэтому DAG со своим собственным ALU делает возможным сканирование по заданной схеме (например, в операциях децимации сканируются каждый 2-й, 4-й, 8-й и т. д. адреса). Большинство «таблиц данных» в цифровой обработке сигналов выглядят как циклические (или кольцевые). Таблицы имеют конечную длину; программа обработки выполняется в цикле, то есть, по достижении конца таблицы выполняется переход на ее начало. Автоматическое определение конечного пункта цикла освобождает CPU от множества дополнительных вычислений и контрольных шагов. В высокопроизводительных DSP генератор адреса данных комбинирует эти функции в пределах одного составного выражения:

$$Y = Y + R; \quad IF \quad R \geq END \quad THEN \quad Y = BEGIN,$$

где Y – значение адреса, R – величина смещения, $BEGIN$ и END задают границы таблицы. Основная часть программ в процессе их выполнения требует модификации величины R. ALU в составе DAG может выполнять такую задачу. Представим ее в виде выражения: $Y = Y + R; \quad modify \quad R$. Модификация, например, может представлять собой фиксированное приращение $R = R + B$. В случае быстрого преобразования Фурье данные из достаточно сильно различающихся областей адресного пространства выбираются и комбинируются для вычислений по операциям типа «бабочка». Здесь ALU генератора адреса данных может осуществлять сдвиги (вверх или вниз), смещения B для реализации

последовательности $RADIX-2$. *Выделение генератора адресов данных в отдельный блок в настоящее время используется в DSP повсеместно.*

3.2.4.1 Свойства DAG

- Сдвиги: способность выполнять сдвиги обеспечивает алгоритмы типа «бабочка» или вращающейся таблицы (twiddle-table) формированием парных адресов. Поскольку при последовательных проходах в алгоритмах *БПФ* адреса смещаются вверх или вниз с множителем 2 (то есть сдвигом на двоичный разряд).
- Бит–реверсирование: перестановка битов в адресном слове используется в *БПФ*, в вычислениях, связанных с децимацией. Прежде требовались отдельные вычисления для пересортировки данных в требуемый порядок. Бит-реверсирующая логика, реализованная в DAG, может это делать автоматически.
- Логические функции и маскирование —эти операции используются при обращении к таблице данных, например, в алгоритме интерполяции по методу Ньютона (рекурсия Ньютона–Рафсона). Способность DAG выполнять логические операции позволяет выполнять частичное маскирование адресного поля для адресации сохраняемого значения функции.
- Гибкая модульная арифметика. При сканировании массива данных DAG вычисляет приращение указателя, и при достижении конца буфера осуществляет его сброс до базового значения. Благодаря тому, что DAG на основании предварительно загруженных в его регистры размеров циклов и адресных смещений, отслеживает состояние программы в цикле, обеспечиваются непрерывные циклы без зависания.
- Реализация цикла (с помощью «круговой буферизации»), являющегося базовым элементом практически для всех программ цифровой обработки сигналов, основывается на «адресации по модулю».

Адрес, превышающий размер кругового буфера, становится первым или начальным адресом таблицы (для 4-х битовой арифметики 1111+ 0001=0000). Адресация по модулю с произвольным значением модуля (не только равным степени 2) есть очень важная особенность, которая позволяет выполнять циклический возврат кругового буфера. Пример:

$$R = B + (R - B + M)MOD(L).$$

Где R – содержимое регистра DAG, B – базовый адрес, M – вычисляемое приращение смещения, L — длина буфера (значение модуля).

Блок-схема DAG для семейства $ADSP-21xx$ приведена на рис. 3.14. Здесь присутствуют отмеченные нами регистры: M (модификации), L (длины цикла), I (индексный), B (базового адреса). Адрес формируется в соответствии с выражением:

$$Next\ \ address = (I + M - B)Modulo(L) + B.$$

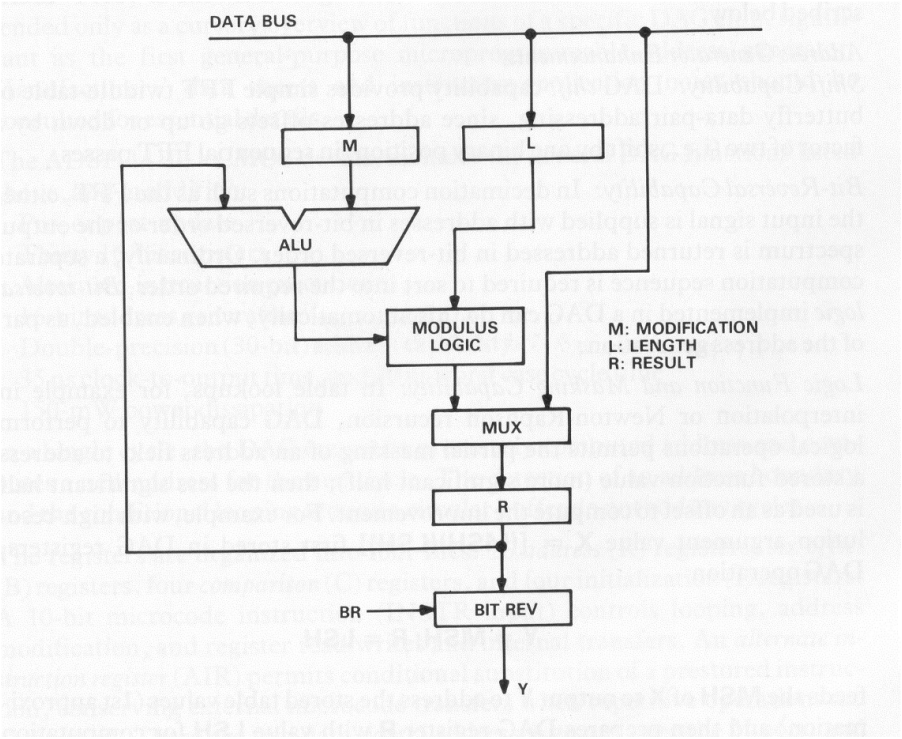


Рисунок 3.13 — Обобщенная схема генератора адреса данных (DAG)

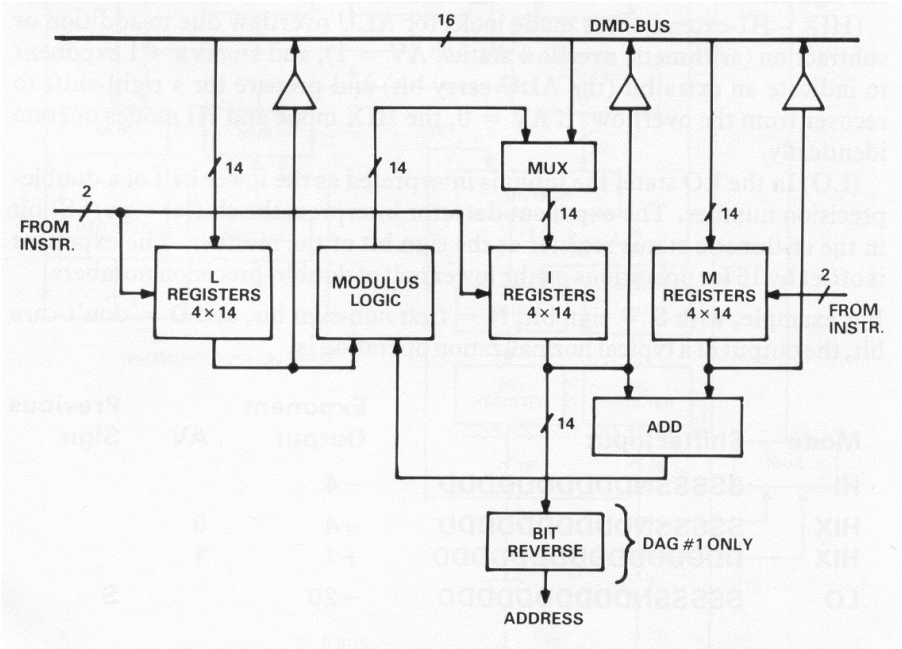


Рисунок 3.14 — Блок-схема DAG ADSP-21xx

3.2.5 Формирователь последовательности команд – PSqr (Program Sequencer)

Формирователь последовательности (потока) команд — $PSqr$ (Program Sequencer) предназначен для генерации адреса команд, которые определяют ход программы. Он устраняет временные издержки, связанные с отслеживанием изменения (наращивания) содержимого счетчика команд (включая условную логику и циклы), управление подпрограммами и внешними прерываниями. Например, программа может иметь счетчики, которые тестируются для определения моментов завершения циклов. Эти функции поддерживаются в $PSqr$ регистровыми стеками (стек счетчика или стек цикла). Адреса возврата восстанавливаются из стека счетчика команд (PC) по завершении подпрограммы и обслуживания запроса на прерывание. $PSqr$ имеет также доступ к статусной информации (знаки, переполнение, заполнение стеков), генерируемой различными вычислительными модулями. Эта статусная информация используется в условных командах (например, в условных переходах: $Jump\ If\ldots$). Таким образом, для $PSqr$ определено несколько главных функций:

- управление нормальным ходом программы, инкрементирование счетчика команд в каждом цикле;
- отслеживание адресации при обращении к подпрограммам, управление стеком возврата;
- управление программными циклами, контроль счетчиков циклов;
- переход на соответствующие специальные программы при обнаружении переполнения или необходимости перемасштабирования данных;
- обслуживание прерываний от устройств ввода/вывода, переходы на соответствующие подпрограммы и возврат к прерванной задаче по завершении.

Для реализации этих функций $PSqr$ должен содержать следующие компоненты:

- счетчик команд PC;
- логика проверки условий;
- стеки для адресов возврата подпрограмм (и прерываний), содержимого счетчиков циклов, адресов переходов, состояний (флагов).

Стеки должны быть достаточно большими, чтобы обеспечить вложение подпрограмм и обеспечивать несколько уровней прерывания.

Следует отметить, что конвейерные схемы для повышения производительности в $PSqr$ применять не рекомендуется, поскольку задержки конвейеризации создают трудности для разработчиков программного обеспечения системы как целого. К примеру, непредсказуемая задержка обслуживания прерываний или реакции на переполнение на несколько тактовых циклов может привести к полному хаосу в программе. $PSqr$ обеспечивает расширение возможностей прямой адресации через внешний порт. При нормальном ходе программы адрес результата выполнения команды хранится в $PSqr$, однако иногда возникает потребность иметь доступ к адресу или смещению адреса через внешний порт. Например, внешняя прямая адресация используется для расширения возможностей ветвления при отладке программ с помощью отладчика. $PSqr$ предоставляет возможность

ветвления с нулевыми затратами, осуществляя переходы по двум – трем направлениям в пределах одной команды, ускоряют выполнение программы и облегчают программирование, снимая явный контроль ветвления. Пример двунаправленного тестирования условия и модификации:

$$IF\ \ LE\ \ BRANCH,\ \ UPDATE\ \ C.$$

$PSqr$ тестирует счетчик цикла ($Loop\ \ End$), если циклы не завершены, то продолжается нормальное наращивание программного счетчика PC — рис. 3.16b.

Функции мощного $PSqr$ поясняются блок-схемами (рис. 3.15 и рис. 3.16). $PSqr$ формирует очередной адрес, комбинируя коды команды ($Jamp\ldots$), логику условий, прерывания и информацию о циклах. Мультиплексор очередного адреса (рис. 3.15) выбирает один из 4-х источников:

- адрес вектора подпрограммы обслуживания прерывания;
- значение (инкрементированное) программного счетчика (PC);
- стек PC (возврат из подпрограммы или из прерывания);
- команда, например, адрес перехода, указанный в команде.

Можно указать и пятую альтернативу — индексированный переход, то есть следующий адрес устанавливается на шину адреса программ извне, а затем «защелкивается» в PC. Нормальный (последовательный) ход программы использует программный счетчик PC и стек PC. Программный счетчик содержит адрес текущей программы и его содержимое инкрементируется. Инкрементированное значение PC адресует следующую команду, а в случае перехода к подпрограмме или по вектору прерывания «заталкивается» в стек PC, как адрес возврата. В операциях с индексированными адресами новое значение PC, сформированное адресным генератором, появляется на шине $PSqr$ для загрузки в PC.

Выбор очередного адреса определяется, как было отмечено выше, одним из четырех источников (рис. 3.16b). Блок логики условий (рис. 3.16c) сравнивает статусную информацию от ALU, MAC с условной частью команды ($Jump\ \ If\ldots$) и группирует соответствующие выходные биты. Сигналы от счетчика CE (Count Expired — счет завершен), от стека циклов и компаратора циклов сравниваются с кодом условия команды, что позволяет выходить из циклов типа DO без затрат времени.

Контроллер прерываний выполняет все функции, связанные с обработкой прерываний, такие как обнаружение и маскирование прерываний, генерация соответствующего адреса вектора прерывания, определение приоритета. Прерывания могут осуществляться по фронту или по уровню сигнала запроса. По фронту проще в реализации и точнее задается момент запроса, однако при наличии шумов более надежным считается запрос по уровню.

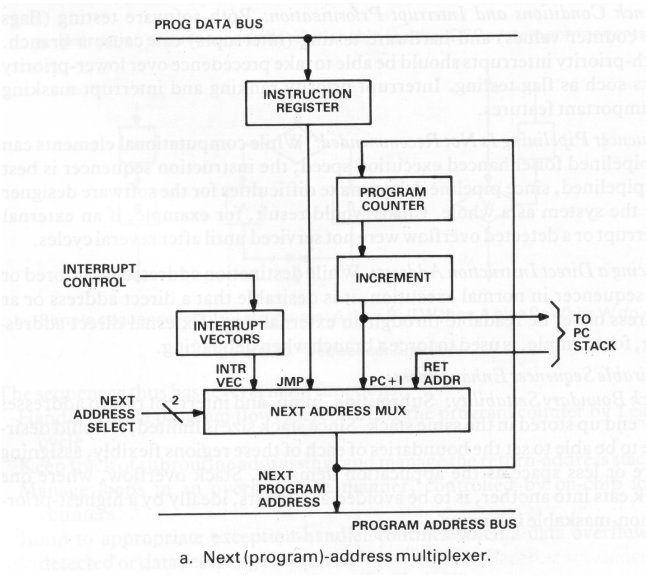


Рисунок 3.15 — Мультиплексор очередного адреса

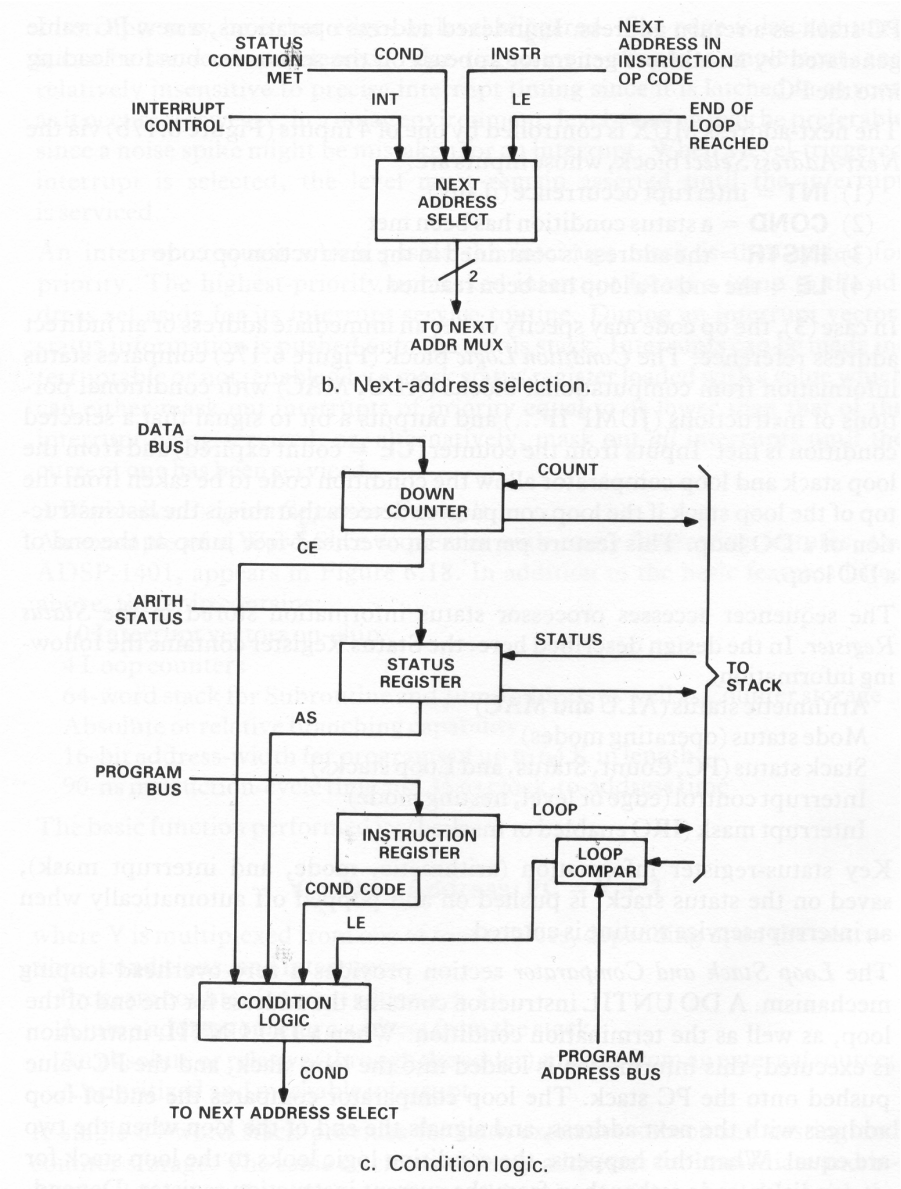


Рисунок 3.16 — Логика выбора очередного адреса (b) и логика условных переходов (c)

3.2.6 Память

Очевидно, что выбор типа памяти, ее конфигурации для высокоскоростной обработки сигналов является одним из ключевых решений DSP и требует рассмотрения множества особенностей. Отметим основные из них на примере памяти семейства ADSP-2106x.

Процессоры ADSP-2106x имеют два независимых блока двухпортовой памяти (рис. 3.17), обозначенных БЛОК 0 и БЛОК 1. Имеется также возможность подключения внешней памяти с адресным полем до 4-х гигаслов. Для хранения данных с плавающей точкой обычной длины используются 32-бит слова, 48-бит слова предназначены для хранения команд либо 40 бит данных повышенной точности.

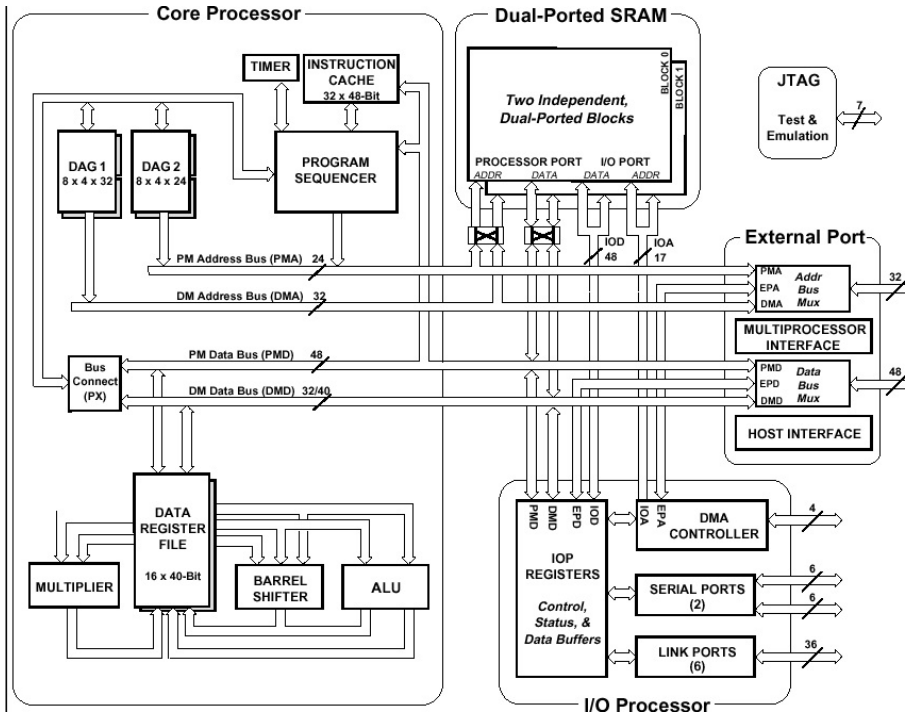


Рисунок 3.17 — Блок-схема ADSP-2106x.

Кроме того, в процессорах этого семейства поддерживается формат 16 бит слов, которые могут использоваться для целых или дробных значений данных с фиксированной точкой. ADSP-2106x имеет три внутренние шины, связанные с двухпортовой памятью $PM-bus, DM-bus$ и контроллером ввода/вывода $I/O-bus$. Один порт — это процессорный порт, куда подключаются $PM-bus$ и $DM-bus$, а к другому порту подключается $I/O-bus$. Внутренние шины памяти данных и памяти программ управляются процессором, тогда как шины ввода/вывода управляются контроллером ввода/вывода, расположенном в том же чипе. Шины ввода/вывода обеспечивают одновременные передачи данных между и коммуникационными портами (последовательные порты, параллельные порты, порты связи (link ports)). Благодаря этой двухпортовой структуре обращения к внутренней памяти со стороны процессора и контроллера ввода/вывода являются независимыми и прозрачными по отношению друг к другу. В каждом тактовом цикле к каждому блоку памяти могут обращаться и процессор, и контроллер ввода/вывода. Обращение к одному и тому же блоку не требует каких-либо дополнительных циклов. Оба (процессор и контроллер ввода/вывода) имеют доступ к внешней памяти по внешней шине через внешний порт. Такая схема магистралей позволяет иметь единое унифицированное адресное пространство для хранения кода и данных.

Для того, чтобы обеспечить параллельную работу рассмотренных основных элементов архитектуры, система команд DSP обеспечивается так называемыми **многофункциональными** *командами. Это команды, с помощью которых выполняется параллельно несколько операций, например, арифметические (используют одновременно MAC, ALU или SHIFTER) и пересылки данных (DAG).*

ГЛАВА 4

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Практикум предназначен для ознакомления студентов с основными этапами разработки приложений ЦОС, для их реализации на сигнальных процессорах.

4.1 Инструментарий DSP и разработка приложений

В основе практикума — изучение аппаратной базы, а также интегрированной среды разработки исполняемого кода и его отладки. Здесь будут использованы стартовые модули от Analog Devices на сигнальных процессорах типа Blackfin и SHARC, а в качестве среды разработки — VisualDSP++.

4.1.1 Модуль ADSP-BF-533-EZLITE фирмы Analog Devices

4.1.1.1 Процессор: архитектура и характеристики

Цифровой сигнальный процессор ADSP-BF533 принадлежит к семейству сигнальных процессоров Blackfin (Черный плавник). 16-ти или 32-х разрядные встраиваемые процессоры Blackfin обеспечивают гибкость и масштабируемость программного обеспечения, необходимые для задач конвергентной обработки (комбинация цифровой обработки сигналов и задач управления): мультиформатной обработки звука, речи, фото- и видеоизображений, мультирежимной обработки данных в широкополосных и пакетных сетях, управления технологическими процессами и охранных системах реального времени. Еще одной важной особенностью продуктов семейства Blackfin является динамическое управление питанием. Возможность изменения как напряжения питания, так и рабочей частоты позволяет оптимизировать потребление мощности в соответствии с конкретной задачей [6].

В основе стартового модуля ADSP-BF533-EZLITE — отмеченный выше процессор, а блок-схема ядра процессора представлена на рис. 4.1. Здесь два умножителя–аккумулятора (MAC), два арифметико-логических устройства (ALU), сдвигатель и четыре видео-ALU; потоками данных управляют два мощных мультиплексора. Это наиболее производительный представитель семейства и предназначен в первую очередь для встроенных видеосистем, например, для систем безопасности (видеонаблюдения), для широкополосных информационных систем и т. д. В отличие от процессоров, созданных на основе традиционных для DSP подходах, в системе команд ADSP-BF533 есть инструкции, ориентированные на поддержку работы операционной системы. Система команд, исполняемых процессорным ядром, включает в себя также специализированные инструкции, позволяющие ускорить обработку аудио- и видеосигналов при использовании алгоритмов в стандартах MPEG2, MPEG4 и JPEG. Кроме того, система команд включает инструкции, адаптированные к выполнению функций управления/контроля, что традиционно присуще микроконтроллерам. Структура вычислительного ядра и набор встроенных периферийных

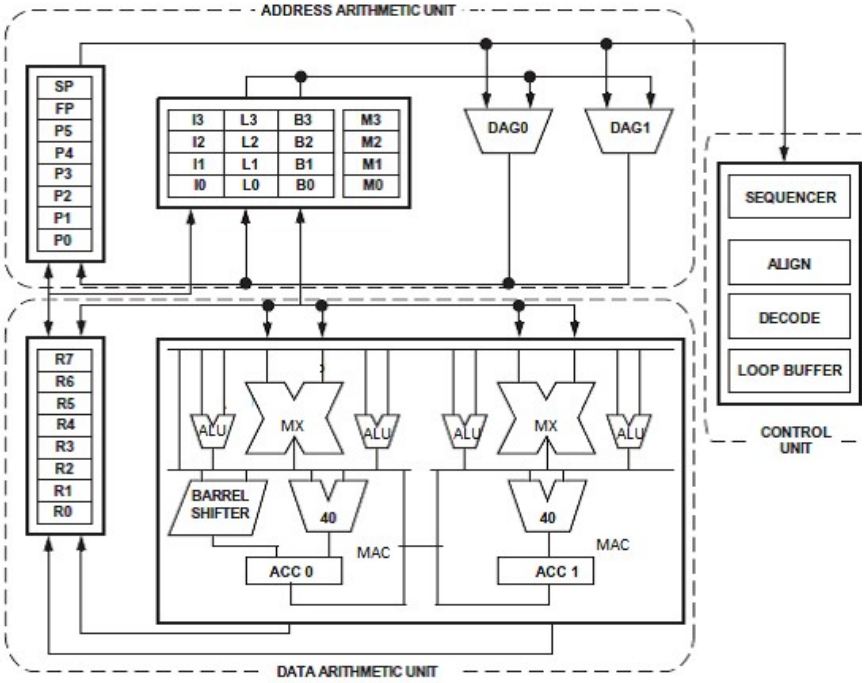


Рисунок 4.1 — Ядро процессора ADSP-BF533

устройств позволяют эффективно работать не только по алгоритмам цифровой обработки сигналов, но и по алгоритмам управления/контроля различными объектами, а также приема/передачи данных.

Регистровый файл содержит восемь 32-х разрядных регистров, которые при работе с 16-бит операндами могут действовать как 16 независимых 16-разрядных регистров. Из них шесть регистров-указателей общего назначения ($P5 - P0$) и специальные регистры для хранения указателя стека (SP) и кадра стека (FP). Непосредственно формированием адреса занимаются два генератора адреса данных — DAG. Они поддерживают режимы прямой адресации, косвенной адресации с пред- и постинкрементом, а такжже специфические для задач цифровой обработки сигналов режимы бит-реверсной и циклической адресации. Адресация может выполняться словами (16 бит), двойными словами (32 бит) или побайтно. Функциональная схема процессора с периферией приведена на рис. 4.2.

Основные периферийные устройства системы процессора ADSP-BF533:

- Модуль интерфейса внешней шины ($EBIU$ — $External\ \ Bus\ \ InterfaceUnit$) — предназначен для подключения внешней памяти к процессору ADSP-BF533. Включает в себя 16-разрядную шину данных, адресную шину и управляющую шину. Поддерживается как 16-разрядный, так и 8-разрядный доступ.
- Параллельный периферийный интерфейс (PPI — $Parallel\ \ Peripheral\ \ Interface$) — отличительная особенность ADSP-BF533 по сравнению с процессорами аналогичного класса. PPI обеспечивает непосредственное подключение внешних устройств (АЦП и ЦАП, видеокодеров и декодеров, фотокамер и т.д.), поддерживающих протоколы передачи данных ITU-R601/656. Имеет несколько режимов передачи данных,

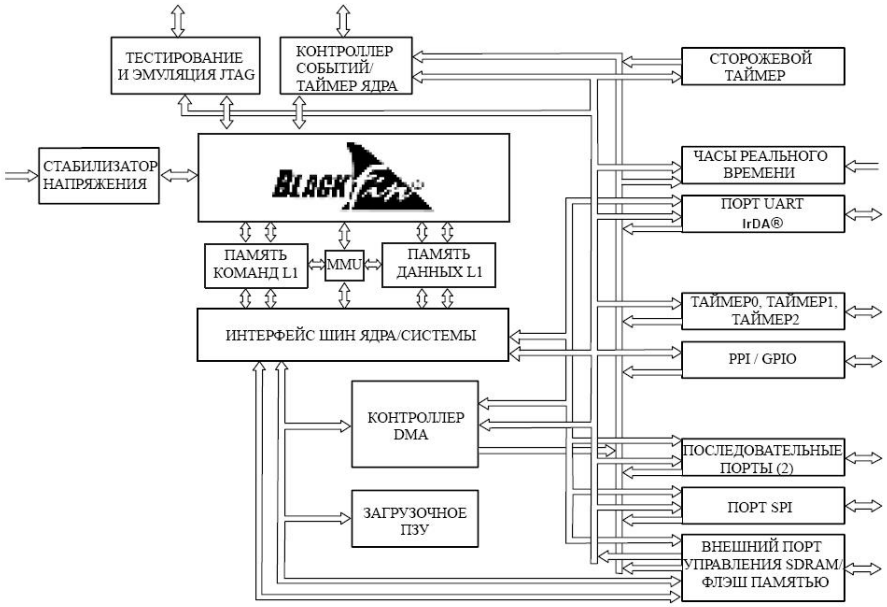


Рисунок 4.2 — Функциональная схема процессора ADSP-BF533

каждый из которых позволяет передавать до 16 бит данных за один такт. Структура PPI представлена на рис. 4.3.

- Порты $SPORT$ — это последовательные порты ввода/вывода; поддерживают большое количество протоколов передачи данных, могут использоваться для связи процессоров в составной вычислительной системе. Каждый $SPORT$ имеет две группы выводов: одну для передачи, другую для приема. Причем прием и передача происходят одновременно, более того, эти функции (приема и передачи) программируются по отдельности, что позволяет одновременно передавать данные в двух направлениях. Слова данных могут иметь размерность от 3 до 32 бит и передаваться со скоростью до 100 Мбит/с. Интерфейс $SPORT0$ подключается к аудиокодеку AD1836 и интерфейсу расширения.
- Последовательный периферийный интерфейс: SPI — $Serial\ \ Peripheral\ \ Interface$ — один из популярных коммуникационных интерфейсов. Имеет три вывода для передачи данных: два вывода данных и один вывод тактового сигнала. Позволяет передавать информацию как в одном направлении, так и одновременно в двух направлениях со скоростью передачи до 20 Мбит/с. SPI подключается к интерфейсу расширения и к аудиокодеку AD1836 для доступа к управляющим регистрам устройства.
- Таймеры общего назначения. ADSP-BF533 имеет 4 таймера общего назначения. Три из них имеют внешние выводы и могут быть использованы в качестве входных или выходных сигналов таймера; могут также синхронизироваться внешним или внутренним сигналами, или совместно с UART могут быть использованы для определения скорости передачи данных в последовательном канале. Четвертый таймер — внутренний. Обычно он используется для создания периодических прерываний операционной системы.

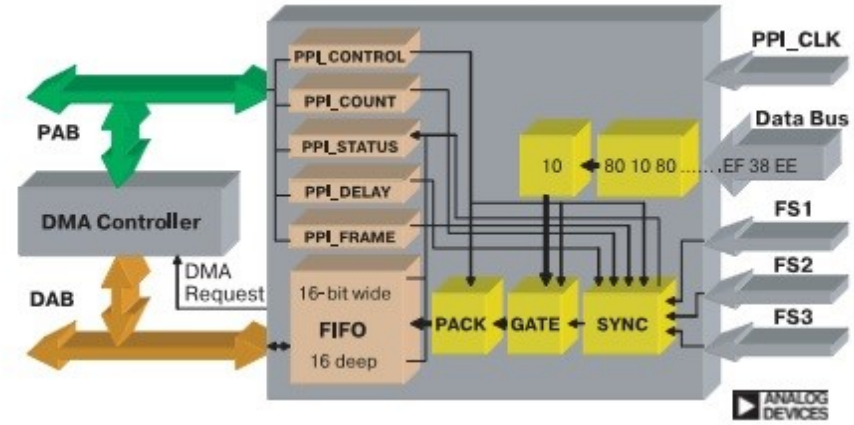


Рисунок 4.3 — Структура контроллера PPI

- Универсальный асинхронный приемник-передатчик ($UART$ – $Universal\ Asynchronous\ Receiver/Transmitter$) — обеспечивает упрощенную связь с другими периферийными устройствами (микросхемы, АЦП, ЦАП, ПК) и реализует полудуплексную асинхронную передачу данных в режиме с поддержкой прямого доступа к памяти (DMA).
- Часы реального времени (RTC — $Real\ Time\ Clock$) — система, включающая в себя функции будильника, секундомера и идентификатора текущего времени. Тактирование происходит от внешнего кварцевого генератора с частотой 32,768 кГц.
- Сторожевой таймер — 32-разрядный таймер — повышает устойчивость системы посредством аппаратного сброса или прерывания общего назначения, если установленное время таймера вышло до срока его программного сброса (защита от «зависания»).
- Порт ввода/вывода общего назначения (программируемые флаги). Процессор имеет 16 портов ввода/вывода общего назначения $PF[15:0]$, каждый из которых может конфигурироваться индивидуально. Данные устройства соединяются с ядром посредством нескольких шин, что обеспечивает меньшую их зависимость друг от друга и высокоскоростную передачу данных.
- Порт эмуляции $JTAG$ позволяет эмулятору получить доступ к внутренней и внешней памяти процессора. Используется для эмуляции, тестирования и отладки программ. Порт также подключается к интерфейсу USB-отладки.

Важным свойством процессора ADSP-BF533 является возможность динамического управления энергопотреблением, основанная на изменении тактовой частоты процессорного ядра и периферийных устройств, а также возможности на программном уровне регулировать напряжение питания ядра. В общем случае суммарная потребляемая процессором мощность определяется режимом его работы (динамическим или статическим) и величиной сквозных токов, протекающих через транзисторы микросхемы. Доминирующей является мощность потребления в динамическом режиме работы: $P_{\text{динам}} = CfU \times 2$, где C – емкость нагрузки, f – частота, U – напряжение питания. Для изменения частоты

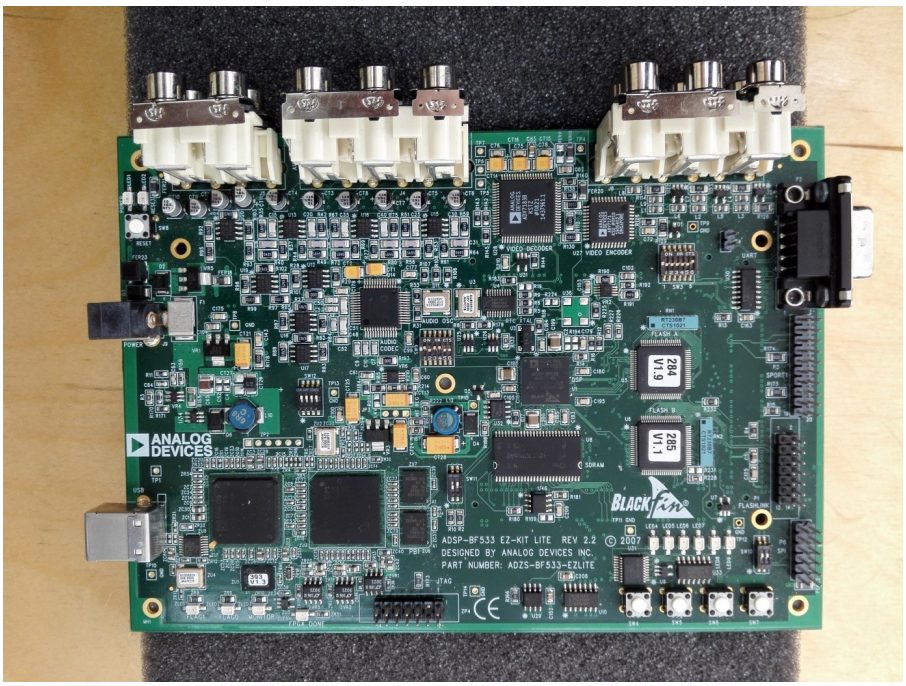


Рисунок 4.4 — Внешний вид платы ADSP-BF533 EZ-KIT Lite

сигналов тактирования процессорного ядра и периферийных контроллеров используется система ФАПЧ. Максимальная тактовая частота (SCLK), используемая для работы периферийных контроллеров, составляет 133 МГц. Имеется также возможность отключения тактового сигнала от периферийных контроллеров, которые не используются в данный момент. Поскольку потребляемая мощность находится в прямой зависимости от величины тактовой частоты и в квадратичной от напряжения питания, наиболее эффективного управления потребляемой мощностью можно добиться за счет изменения напряжения питания ядра. Для этой цели в сигнальном процессоре ADSP-BF533 реализован встроенный импульсный стабилизатор напряжения, позволяющий на программном уровне изменять напряжение питания ядра с шагом 50 мВ. В процессоре ADSP-BF533 реализовано пять программно-управляемых режимов работы со сниженным уровнем энергопотребления. Внешний вид и функциональная схема модуля ADSP-BF533-EZLITE [8] представлены на рис. 4.4 и рис. 4.5.

4.1.1.2 Основные характеристики модуля

- *процессор Analog Devices ADSP-BF533:*
  - производительность до 600 МГц;
  - генератор 27 МГц.
- *Синхронное динамическое оперативное запоминающее устройство* (*SDRAM* — T48LC32M16 — 64 MB).
- Флеш-память — 2 Мбайт
- *Аналоговый аудиоинтерфейс:*
  - аудиокодек AD1836 96кГц;
  - 4 входных RCA-гнезда (2 канала);

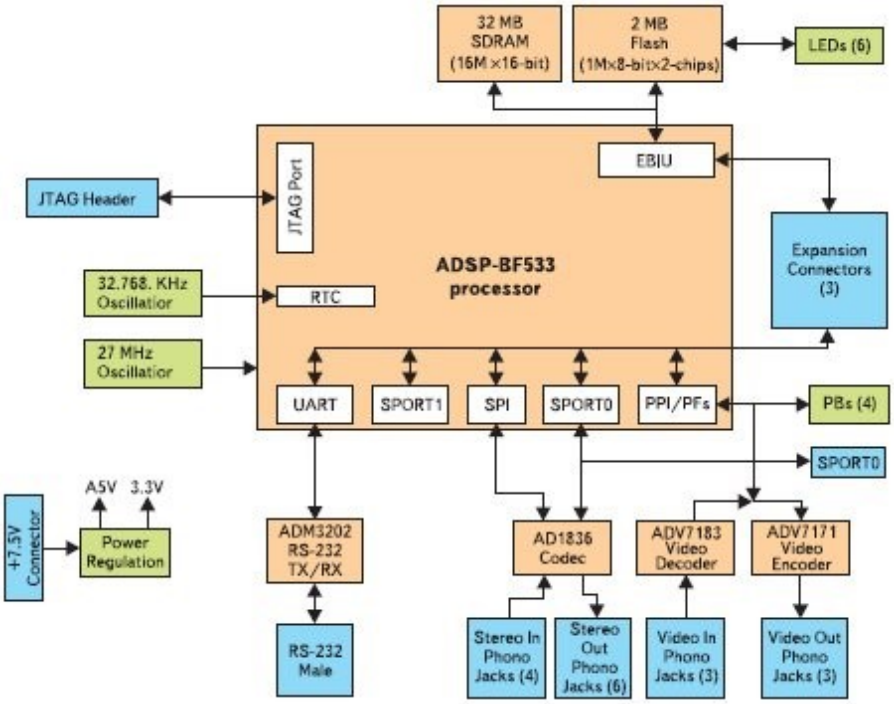


Рисунок 4.5 — Функциональная схема ADSP-BF533-EZLITE

  - 6 выходных RCA-гнезда (3 канала).
- *Аналоговый видеоинтерфейс:*
  - видеодекодер ADV7183 с 3 RCA-гнездами;
  - видеокодер ADV7171 с 3 RCA-гнездами.
- *Универсальный асинхронный приемник/передатчик (UART):*
  - линейный драйвер/приемник ADM3202 RS-232;
  - штекерный разъем DB9 male.
- *Светодиоды (10 светодиодов):*
  - питание (зеленый);
  - сброс (красный);
  - USB (красный);
  - 6 универсальных (желтый);
  - USB монитор (желтый).
- *Кнопки (5 кнопок):*
  - сброс;
  - 4 программируемые — флаги.
- *Интерфейс расширения:*
  - $PPI, SPI, EBIU$;
  - $Timers$ $[2-0]$;
  - программируемые флаги и $SPORT0$, $SPORT1$;
  - дополнительно: 14-контактный разъем $JTAG$ ICE.

Модуль ADSP-BF533-EZLITE также использует два PSD4256G6V flash устройства ввода/вывода общего назначения от STMicroelectronics. Эти устройства обеспечивают

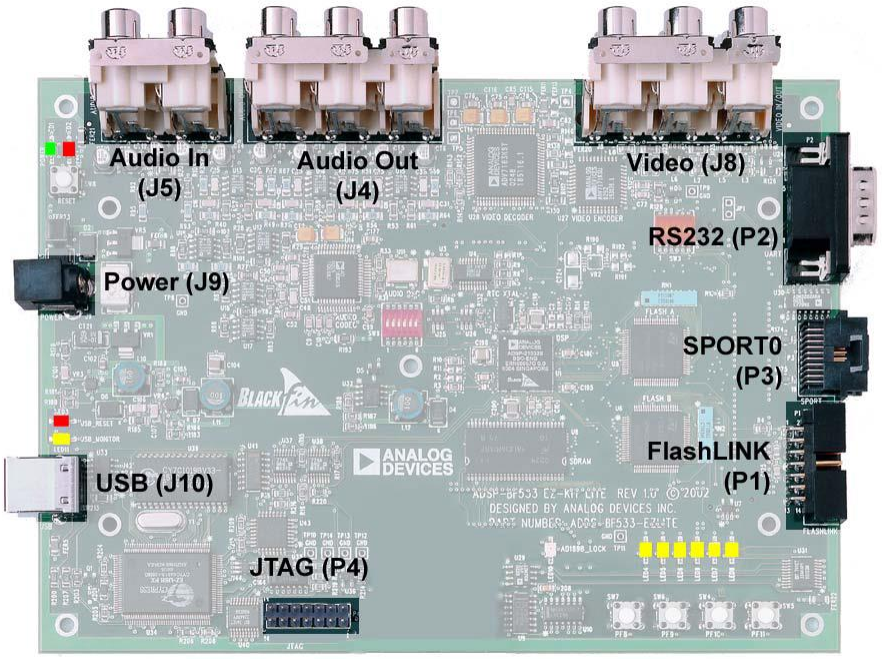


Рисунок 4.6 — Элементы ввода/вывода модуля ADSP-BF533-EZLITE

не только флэш-память, но и дают дополнительные контакты ввода/вывода [8]. Каждое устройство включает в себя следующие сегменты памяти:

- 1 Мбайт основной флэш-памяти;
- 64 Кбайт вторичной флэш-памяти;
- 32 Кбайт встроенной SRAM;
- 256 байт регистров контроля ввода/вывода.

Расположение и назначение основных элементов модуля.

Расположение основных элементов модуля представлено на рис. 4.6.

4.1.1.3 Переключатели

Переключатели (SW) и перемычки (JP), — их расположение показано на рис. 4.7

- Перемычка $JP4 -- UART$ — позволяет устанавливать соединение обратной связи для сигналов передачи и приема, которое предназначено для определения работоспособности последовательной связи между модулем и компьютером, устранения неполадок.
- Переключатель режима загрузки — $SW11$:
  (on,on) — 16-разрядная внешняя память;
  (off,on) — флэш-память (настойка по умолчанию);
  (on,off) — хост-сервер SPI;
  (off,off) — SPI EEPROM (тип последовательной памяти).
- Тестовые DIP-переключатели: $SW1, SW2$ — два переключателя, находящихся на нижней части платы. Используются только для тестирования и должны находится в положении OFF по умолчанию.

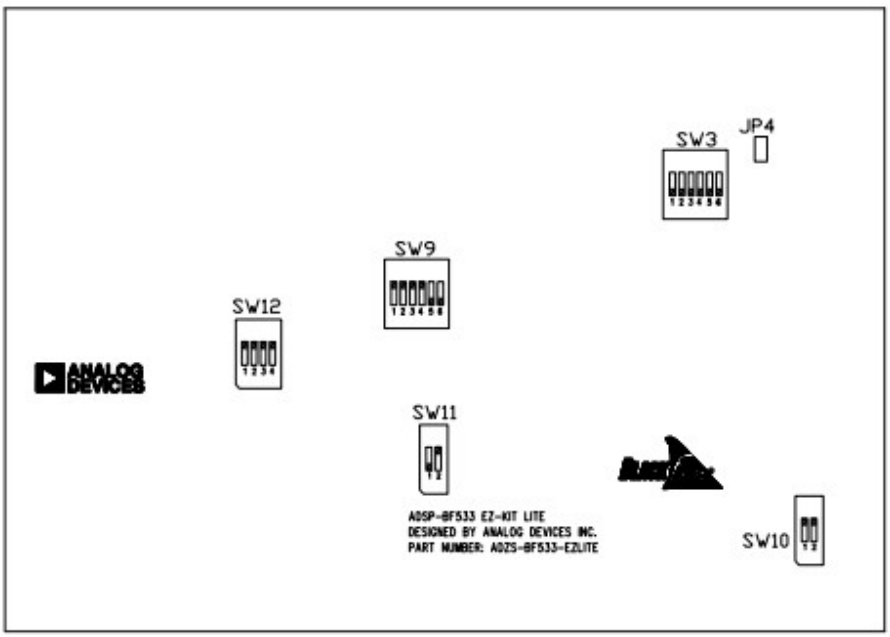


Рисунок 4.7 — Расположение переключателей

- Переключатель конфигурации видео $SW3$ — управляет тем, как сигналы от видеодекодера и видеокодера будут передаваться на PPI процессора.
- Переключатель $SW9$ — используется для управления режимами драйверов процессора, синхронизации приема и передачи.
- Переключатель SPORT0 — $SW12$:
  off — отключает $SPORT0$ от аудиокодека;
  on — переключает сигналы $SPORT0$ на интерфейс расширения.
  По умолчанию используется положение on.

4.1.1.4 Светодиоды и кнопки.

Их расположение показано на рис. 4.8 и назначение следующее:

- Программируемые кнопки $SW4 - SW7$ — предназначены для ввода данных общего назначения. Подключены к программируемым контактам $PF8 - PF11$. Конфигурация $SW9$ регулирует подключение $SW4 - SW7$ к $PF8 - PF11$.
- Кнопка сброса $SW8$ — производит сброс всех схем на плате, кроме интерфейса USB $(U34)$.
- Индикатор питания $LED1$; если горит (зелёный) — питание подается на плату правильно.
- Светодиод сброса $LED2$; если горит — происходит сброс схем.
- Пользовательские светодиоды $LED4 - LED9$: шесть светодиодов подключены к шести выводам общего назначения флэш-памяти (U5). Если светодиод горит, то по соответствующему адресу флэш-памяти записана 1.
- Светодиодный индикатор USB (ZLED3) — показывает, что связь USB успешно инициализирована и возможно подключение к процессору с помощью сеанса VisualDSP++ & EZ-KIT Lite.

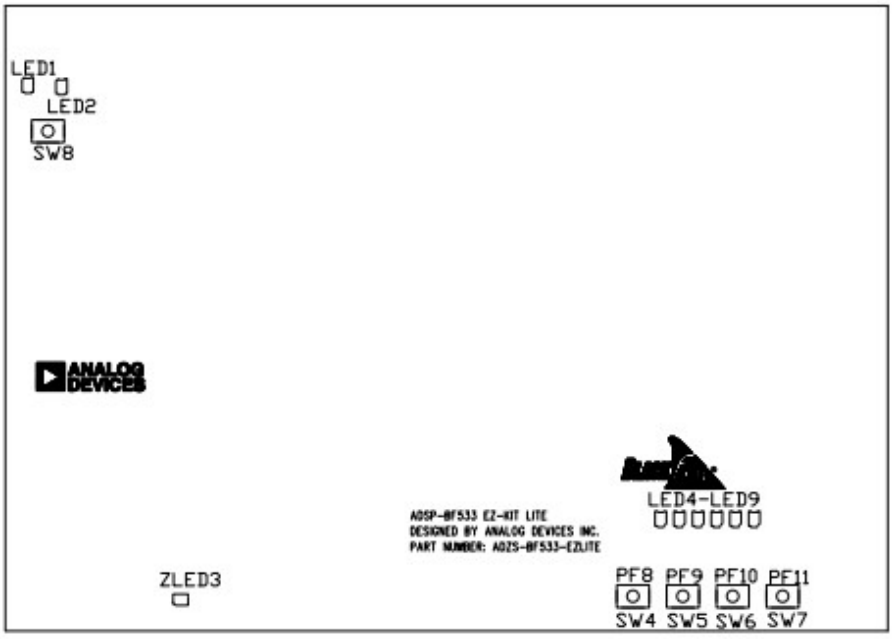


Рисунок 4.8 — Расположение кнопок и светодиодов

4.1.1.5 Разъемы (соединители).

Здесь дается описание соединителей, их функции. Расположение показано на рис. 4.9:

- $J4$, $J5$: $J5$ — аудиовход, $J4$ — аудиовыход;
- $J8$ — видеовход/видеовыход;
- $J9$ — питание модуля;
- $P1$ — FlashLINK — предназначен для настройки и программирования микросхемы, обновления прошивок устройств;
- $P2$ — RS-232 — стандартный вход для интерфейса UART. (Используется для подключения специального или устаревшего оборудования).
- $P3$ — $SPORT1$ — последовательный порт ввода вывода; предназначен для передачи данных и построения многопроцессорной вычислительной системы.
- $ZP4$ — $JTAG$ — разъем для подключения внутрисхемного эмулятора $JTAG$ (когда эмулятор подключен к порту $JTAG$, интерфейс отладки USB отключается);
- $P6$ — SPI — разъем для передачи данных через последовательный периферийный интерфейс.

4.1.2 Отладочный модуль ADSP-21262-EZLITE

4.1.2.1 Процессор: архитектура и характеристики

Основой комплекта ADSP-21262-EZLITE (рис. 4.10) является сигнальный процессор ADSP-21262 (SHARC), представленный на рис. 4.11. ADSP-21262 является первым представителем 3-го поколения 32-разрядных DSP SHARC, с плавающей точкой, основанных на SIMD-архитектуре ядра (с одним потоком команд и множеством потоков данных), которое оптимизировано для высокопроизводительной цифровой обработки сигналов. По-

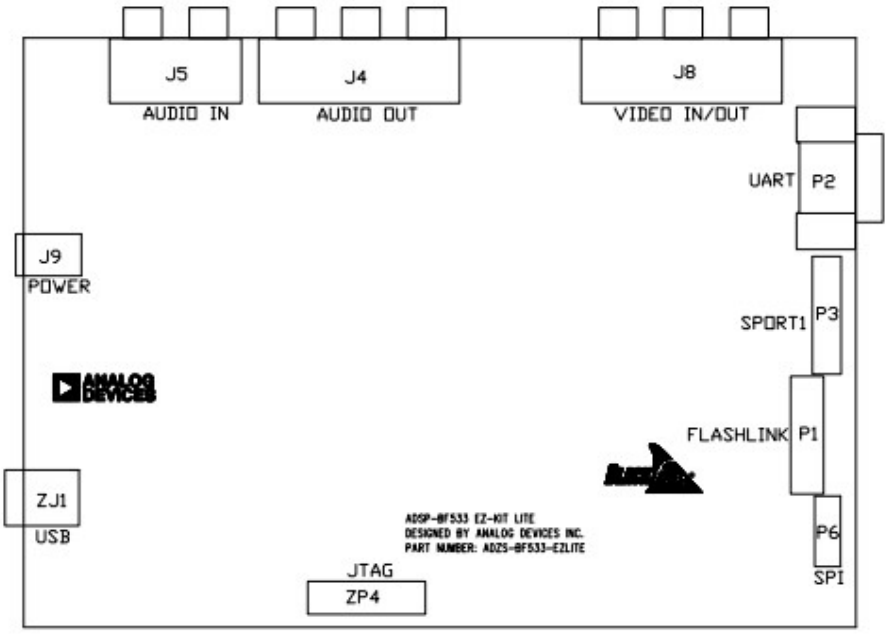


Рисунок 4.9 — Расположение разъемов

добно другим SHARC процессорам, ADSP-21262 совместим по набору инструкций с другими представителями семейства и поддерживает как целочисленные, так и вещественные типы данных.

В архитектуре SHARC (Super Harvard Architecture) концепция использования раздельных шин памяти данных и памяти программ расширена за счет добавления процессора ввода/вывода (IOP - I/O processor) с эксклюзивным набором шин. Все процессоры SHARC независимо от поколения и порядкового номера имеют следующие общие архитектурные компоненты [7]:

- два вычислительных элемента, каждый из которых содержит умножитель, АЛУ и устройство сдвига, поддерживающие операции над 32-разрядными операндами с ПТ формата IEEE, а также 10-портовый регистровый файл данных;
- программный автомат, взаимодействующий с кэш памятью команд, интервальным таймером и двумя генераторами адреса данных (DAG1 и DAG2);
- процессор ввода/вывода с интегрированным контроллером DMA и набором портов для подключения внешних устройств;
- двухпортовая память SRAM, к которой одновременно может обращаться ядро и процессор ввода/вывода;
- порт JTAG для внутрисхемной эмуляции.

Наличие в процессоре трех шин: памяти данных (ПД), памяти программ (ПП) и ввода/вывода (IO), а также возможность обращения по шине ПП к данным позволяют за один цикл выбирать из памяти два операнда данных (один из ПП, другой из ПД), выбирать одну команду из кэш-памяти и выполнять пересылку в режиме DMA.

Процессоры третьего поколения, к которым относится процессор ADSP-21262, ориентированы в первую очередь на применение в аудиосистемах. Они имеют ту же архитек-

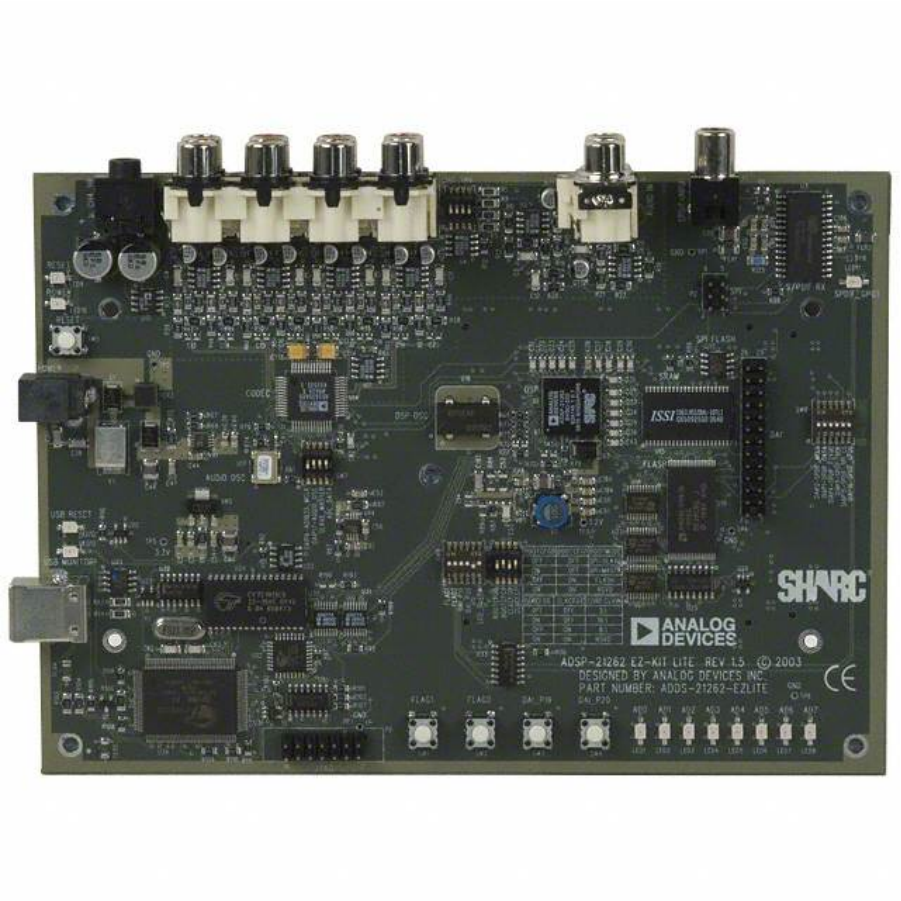


Рисунок 4.10 — Внешний вид модуля ADSP-21262 EZLITE

туру ядра, что и процессоры второго поколения, но существенно отличаются от них в части организации памяти и архитектуры процессора ввода/вывода. Архитектура процессора ADSP-21262 показана на рис. 4.11. Как видно на приведенной блок-схеме, память процессора дополнена блоками внутреннего масочно-программируемого ПЗУ. При изготовлении некоторых микросхем семейства в эту память записываются различные алгоритмы обработки звука (Dolby Digital/Pro Logic, DTS, WMA, MPEG2 AAC). Для поддержки высокой тактовой частоты работы ядра (до 200 МГц) в состав процессора включена программно-конфигурируемая схема ФАПЧ. Однако основные изменения затронули процессор ввода/вывода. Процессор ввода/вывода в процессоре ADSP-21262 включает в себя 4 выделенных вывода GPIO, порт SPI и 16-битный параллельный порт, выводы которого могут также использоваться в режиме GPIO. Остальные интегрированные периферийные модули ADSP-21262, к которым относятся последовательные порты, таймеры входа прерываний, прецизионные генераторы тактовых частот, дополнительные линии GPIO, а также порт ввода данных/PDAP, не имеют жестко назначенных выводов на корпусе. Вместо этого отдельные сигналы периферийных модулей мультиплексируются в блоке маршрутизации сигналов (SRU - Signal Routing Unit) на 20 внешних выводах. Это позволяет сократить количество выводов и, следовательно, снизить его стоимость. Очевидно, по этим же соображениям из состава процессора ввода/вывода ADSP-21262 исключен контроллер SDRAM. Линк-порты и поддержка мультипроцессорных систем в процессорах

третьего поколения также отсутствуют, поскольку для организации высокопроизводительных параллельных вычислительных систем компания Analog Devices предлагает другое семейство процессоров — TigerSHARC.

ADSP-21262 включает в себя шесть сдвоенных последовательных синхронных портов, которые обеспечивают эффективное взаимодействие с широким спектром периферийных устройств с цифровым или смешанным сигналом. Каждая шина данных имеет свой собственный DMA-канал и может быть запрограммирована либо на передачу, либо на получение данных. Последовательные порты включают в себя 12 программируемых контактов для синхронной передачи либо получения сигналов, которые поддерживают до 24-х передающих либо принимающих каналов аудио данных, когда все шесть портов SPORT активны [7]. Последовательные порты используют до четверти тактовой частоты ядра. Каждый последовательный порт может работать в тандеме с другим последовательным портом для оказания помощи TDM, т. е. один SPORT выполняет две передачи сигнала, в то время как другой производит два приема сигнала. Последовательные порты работают в четырех режимах:

- стандартный последовательный режим DSP;
- мультиканальный (TDM) режим;
- $I2S$ режим;
- лево-ориентированный парный режим.

Лево-ориентированный парный режим — это режим, где в каждом цикле синхронизации кадров операнды могут либо перемещаться, либо приниматься: один операнд на старшем сегменте кадровой синхронизации данных, а другой на младшем. Этот режим программно контролируется.

4.1.2.2 Состав модуля

ADSP-21262 EZ-KITLITE содержит три типа памяти: параллельную flash-память объемом 1 Мбайт, flash-память SPI объемом 2 Мбит и $SRAM$ память объемом 512 Кбит. Flash-память позволяет хранить загрузочные коды пользователя, что дает ей возможность работать в качестве автономного устройства. Параллельная flash-память и SRAM подключена к параллельному порту процессора. Параллельный порт является мультиплексируемым портом адреса и данных. Порт может быть подключен к 8-ми и 16-ти разрядным устройствам памяти. Модуль ADSP-21262-EZLITE, как и ADSP-BF533-EZLITE, оснащен большим количеством периферийных контроллеров:

- параллельный периферийный интерфейс (PPI);
- последовательный периферийный интерфейс (SPI);
- последовательные порты ($SPORT$);
- порт ввода/вывода общего назначения (программируемые флаги);
- универсальный асинхронный приёмник-передатчик ($UART$);
- часы реального времени (RTC);
- таймеры общего назначения;
- сторожевой таймер.

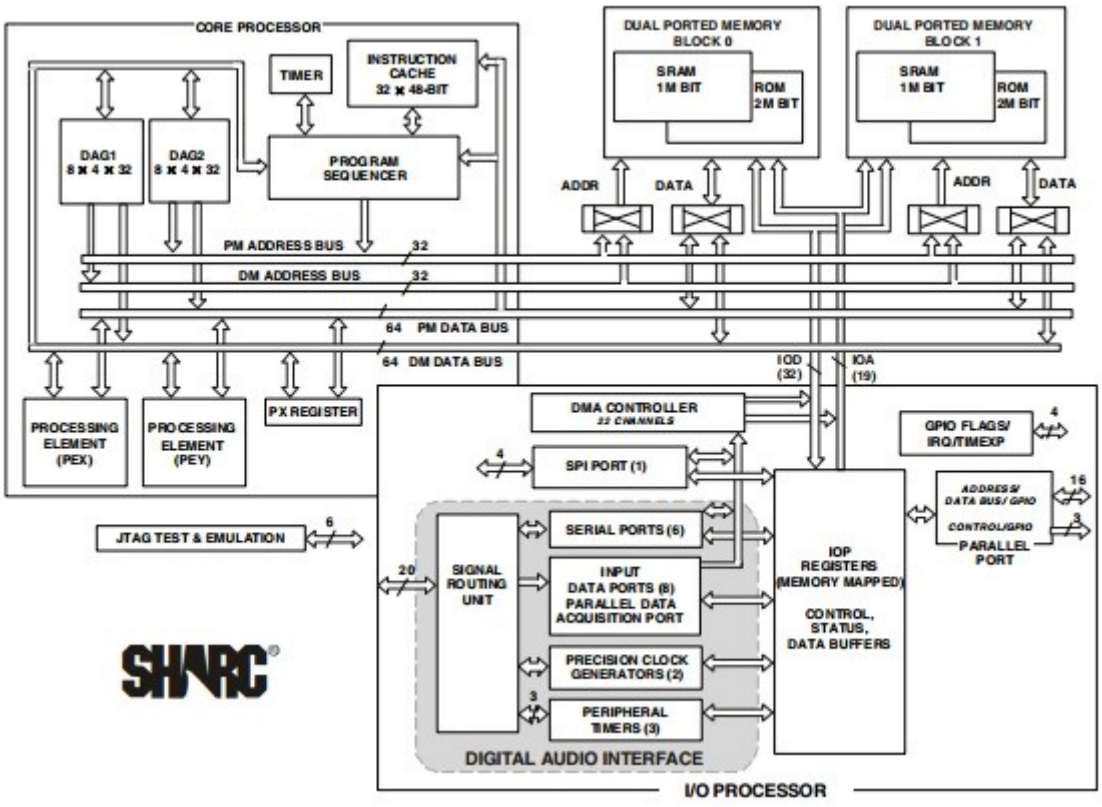


Рисунок 4.11 — Функциональная схема процессора ADSP-21262 SHARK

Для работы с аналоговыми сигналами, в частности со звуком имеется микросхема AD1835A, которая является высокопроизводительным кодеком с четырьмя стерео цифро-аналоговыми преобразователями (ЦАП) для аудио выхода и одним стерео аналого-цифровым преобразователем (АЦП) для аудио входа. Кодек способен получать и передавать информацию с частотой до 96 кГц на всех каналах. Один из каналов ЦАП может работать на частоте 196 кГц [7]. Процессор соединен с аудиокодеком через цифровой аудио интерфейс DAI (Digital audio interface) и способен передавать данные к аудио кодеку в режиме с временным разделением каналов (TDM) или режиме двухпроводного интерфейса (TWI). Также ADSP-21262 EZ-KITLITE содержит устройство CS8416, которое представляет собой монолитный CMOS чип, способный принимать и декодировать один из восьми аудио каналов данных в соответствии со стандартами IEC0958, S/PDIF (Sony/Phillips Digital Interface), EIAJ CP1201 или AES3. Чип S8416 получает данные от передающей шины, восстанавливает тактовые сигналы синхронизации и демультиплексирует аудио- и цифровые данные и соединяется с процессором через порт DAI.

В модуле ADSP-21262 EZLITE имеются также 8 светодиодов и 4 пользовательские кнопки общего назначения. Две кнопки отвечают за FLAG-регистры процессора, а оставшиеся две за DAI-регистры. Нажатие на кнопки может генерировать прерывания [7].

4.1.2.3 Функциональная схема и основные характеристики модуля

- SIMD-ядро: 200 МГц (5 нс);
- производительность 1200 MFLOPS, 800 MMACS;

– совместимость по коду с предыдущими поколениями SHARC;
– поддержка чисел с 32/40-разрядной плавающей точкой и 32-разрядной фиксированной точкой;
– выполнение инструкции за один такт, включая SIMD операции в обоих вычислительных модулях;
– 512 Кбит внутренней SRAM;
– 22 канала ПДП, работающих без тактов ожидания;
– 6 синхронных последовательных портов с поддержкой стандарта I2S и 128-канальной мультиплексной передачи с временным уплотнением;
– обращение за один цикл к 100 МГц 48-разрядной внешней памяти;
– два 100 Мб/с link-порта;
– поддержка стандарта SPI;
– 16-битный параллельный порт;
– 4 таймера с возможностью ШИМ;
– поддержка оптимизированными C и C++ компиляторами;
– поддержка средствами разработки VisualDSP;
– JTAG отладочный интерфейс;
– 8 LED- светодиодов;
– 4 кнопки управления;
– flash-память 1Мб;
– 2 микрофонных входа и выход на стереонаушники;
– 24-х разрядный стереокодек AD1835 (96 кГц);
– ресивер CS8416.

4.1.2.4 Переключатели

Здесь дано описание переключателей (SW) и перемычек (JP); их расположение показано на рис. 4.13.

– SW6 — для подключения электретного микрофона к аудиовходу нужно установить все позиции переключателя SW6 в положение on. По умолчанию все позиции в положении off. Когда *все* переключатели находятся в положении on, к сигналу добавляется постоянное смещение 2.5V, а коэффициенты усиления входных усилителей изменяются от 1 до 10.
– SW7 — переключатель настройки кодека (SW7) используется для маршрутизации сигналов, идущих к кодеку AD1835A, и настройки протокола связи кодека. Позиции 1 и 2 определяют маршрут тактирования аудиоосциллятора к кодеку и процессору. Рис. 4.13 иллюстрирует, как позиции 1 и 2 переключателя установлены на плате. По умолчанию для тактирования AD1835A — это маршрут от контакта DAI_P17 к DAI_P6. Позиция 3 переключателя SW7 определяет, является ли устройство AD1835A ведущим или ведомым. Если AD1835A является ведущим устройством, то последовательный интерфейс устройства генерирует синхроимпульсы кадров и тактовые сигналы, необходимые для передачи данных. Когда устройство

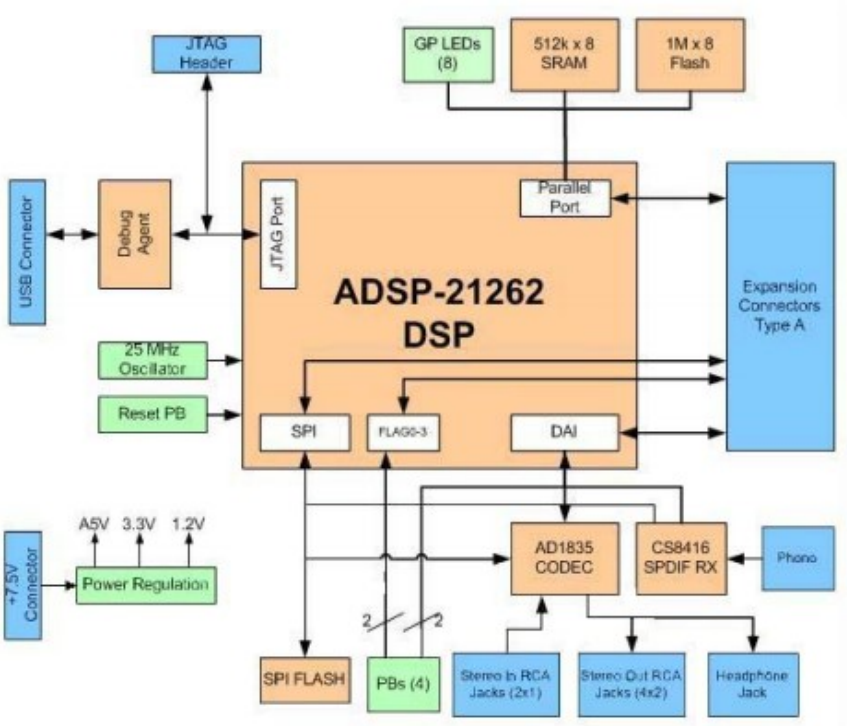


Рисунок 4.12 — Функциональная схема модуля ADSP-21262-EZLITE

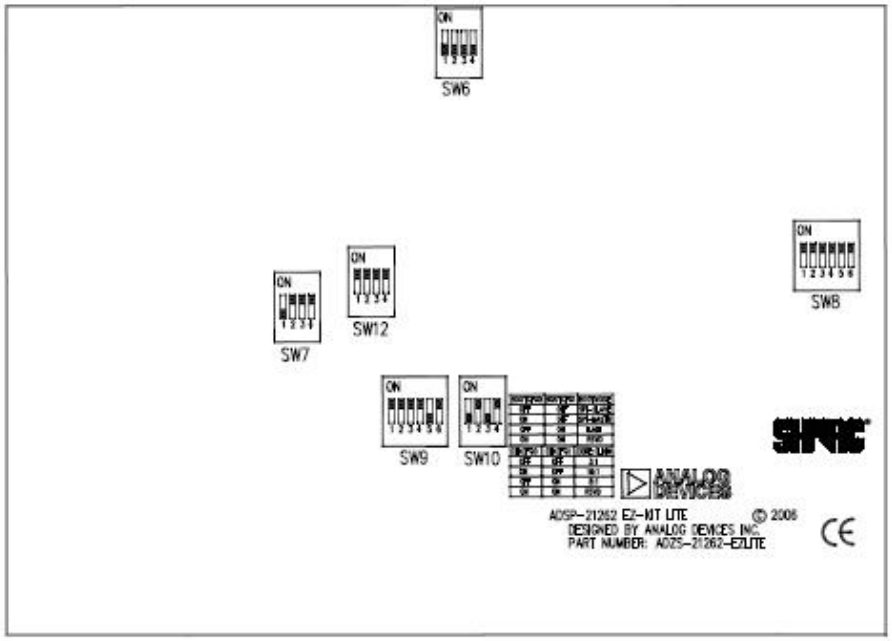


Рисунок 4.13 — ADSP-21262 EZLITE — расположение переключателей; заводская установка

Switch Position	Processor Pin	S/PDIF RX Pin
1	DAI_P2	MCK
2	DAI_P1	SDATA
3	DAI_P4	FSYNC
4	DAI_P3	SCK
5	DAI_P15	SPI_CS
6	DAI_P16	GPOO

Рисунок 4.14 — Функции переключателя SW8

является ведомым, тогда генерировать синхроимпульсы кадра и тактовые сигналы должен процессор. По умолчанию позиция 3 в положении on, а AD1835A генерирует эти управляющие сигналы. Позиция 4 SW7 отсоединяет вывод adc_data кодека AD1835A от интерфейса DAI. Это полезно, когда интерфейс DAI используется для связи с другим устройством.

- SW8 — переключатель разрешения сигнала S/PDIF; размыкает цепь сигнала приемника последовательного интерфейса S/PDIF чипа CS8416. В таблице на рис.4.14 показано, какие сигналы S/PDIF подключаются к выводам процессора при соответствующих положения переключателя.
- Переключатель SW9 отключает кнопки от выводов процессора в соответствии с рис. 4.15. Это позволяет использовать как сигналы от кнопок, так и выводы процессора для иных целей. Таблица показывает сигналы и соединения для SW9. По умолчанию позиции 1-4 переключателя в положении on, то есть кнопки подключены к процессору. Позиция 6 переключателя SW9 подключает/отключает «защелкиваемые» пины LED для реализации логического OR сигналов WE и LED_CS. В положении off «защелкиваемый» пин LED (U24) подтягивается к высокому уровню, делая «защелку» прозрачной. В этом положении LED-ы непосредственно подсоединены к AD7-0. Когда позиция 6 в положении on, значения LED устанавливаются операцией записи в память: младшие 8 бит данных, записанные по адресу 0x1400 0000, устанавливают значения LED. По умолчанию позиция 6 переключателя SW9 в положении ON.
- SW10 устанавливает режим загрузки и коэффициент умножения частоты тактовых импульсов. Таблица (рис.4.16) показывает, как настроить режим загрузки, используя позиции 1 и 2 переключателя. По умолчанию, загрузки модуля EZ-KIT Lite производятся в режиме ведущего SPI или параллельного порта, а процессор загружается из флэш-памяти.

  Следующая таблица (рис. 4.17) поясняет, как установить нужную частоту тактирования.

Switch Position	Push Button Reference Designator	Processor Pin
1	SW1	FLAG1
2	SW2	FLAG2
3	SW3	DAI_P19
4	SW4	DAI_P20

Рисунок 4.15 — Функции переключателя SW9

BOOTCFG1 Pin (Position 2)	BOOTCFG0 Pin (Position 1)	Boot Mode
OFF	OFF	SPI slave
OFF	ON	SPI master
ON	OFF	Parallel flash boot (default)
ON	ON	Internal

Рисунок 4.16 — Переключатель SW10 — режимы загрузки

4.1.2.5 Светодиоды и кнопки

В этом подразделе описывается функциональность светодиодов и кнопок. Их расположение показано на рис. 4.20. Светодиоды подключаются к контактам параллельного порта AD7-0 через регистр-защелку. Параллельный порт процессора может быть настроен как шина памяти, или как выводы флагов общего назначения — FLAG. Регистр-защелка может работать в обоих случаях. Если к светодиодам идет обращение через регистр FLAG, то нужно настроить параллельный порт на FLAG-пины, а для этого нужно установить бит PPFLG в регистре SYSCTL. В таблице на рис. 4.18 показано, как подключать LED.

Две кнопки общего назначения подключены к пинам FLAG процессора, а две других — к контактам DAI. Кнопки подключаются к процессору через DIP-переключатель SW9 (см. п. 4.1.3.3). Состояние кнопок, подключенных к пинам FLAG, может быть получено чтением регистра FLAG, а кнопки, подключенные к шинам DAI, должны быть сконфигурированы как источники внешних прерываний (см. рис. 4.19).

- LED1-LED8 — 8 светодиодов общего назначения; управляются через регистр-защелку сигналами AD7-0. Доступ через запись в регистры флагов (FLAG), либо посредством адресации к памяти.
- LED9 — сброс; когда светится красным — сброс всей основной периферии.

CLKCFG1 (Position 4)	CLKCFG0 (Position 3)	Core to CLKIN Ratio
OFF	OFF	3:1
OFF	ON	16:1
ON	OFF	8:1 (default)
ON	ON	NA

Рисунок 4.17 — Переключатель SW10 — режимы тактирования

LED Reference Designator	Processor Pin	Mapped as Flag
LED1	AD0	FLAG8
LED2	AD1	FLAG9
LED3	AD2	FLAG10
LED4	AD3	FLAG11
LED5	AD4	FLAG12
LED6	AD5	FLAG13
LED7	AD6	FLAG14
LED8	AD7	FLAG15

Рисунок 4.18 — Управление светодиодами

Push Button Reference Designator	Processor Pin
SW1	FLAG1
SW2	FLAG2
SW3	DAI_P19
SW4	DAI_P20

Рисунок 4.19 — Подключение кнопок

- LED10 — питание: светится зеленым — питание модуля в норме.
- LED11 – S/PDIF GPO1; связан с сигналом GPO1 чипа CS8416. Функциональность GPO1 программируется через SPI.
- LED (ZLED3) — USB монитор: индицирует USB связь, зажигается в течении 15 сек после подключения кабеля (если нет, то выключить => включить питание или переустановить драйвер).
- Программируемые кнопки (SW1-SW4) — предназначены для ввода данных общего назначения. Две подключены к FLAG-пинам процессора, две другие к DAI (кнопки доступны через переключатель SW9). Сигналы процессора, связанные с кнопками, приведены в таблице на рис. 4.19.
- Кнопка сброса (SW5) — производит сброс всех схем на плате.

4.1.2.6 Разъемы (соединители)

Их расположение и обозначение показаны на рисунке 4.21:

- J4 и J5: J4 — аудиовход, J5 — аудиовыход;
- J6 — наушники;
- J7 — питание модуля;
- J8 — коаксиальный разъем SPDIF;
- P2 — SPI;
- P3 — DAI;

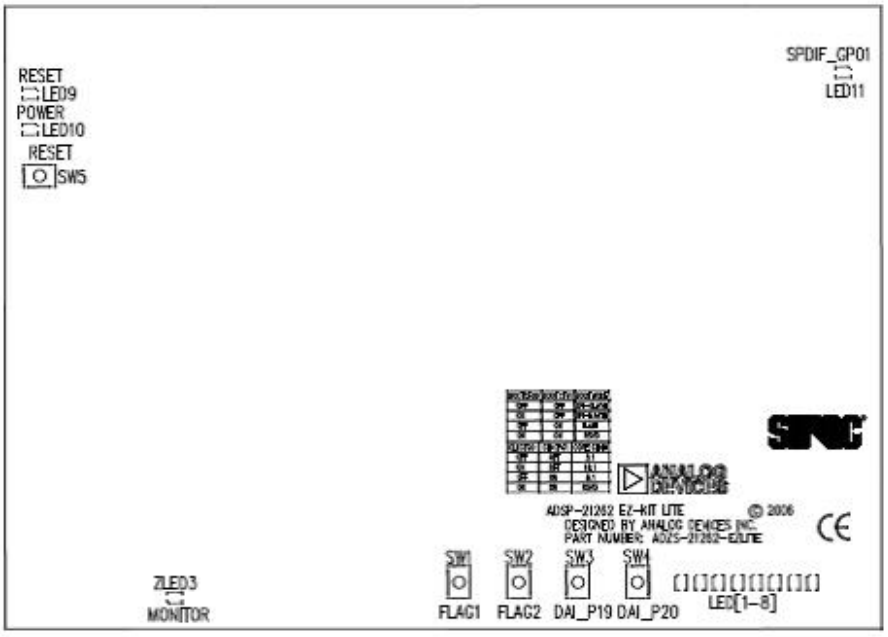


Рисунок 4.20 — ADSP-21262-EZLITE — расположение кнопок и светодиодов []

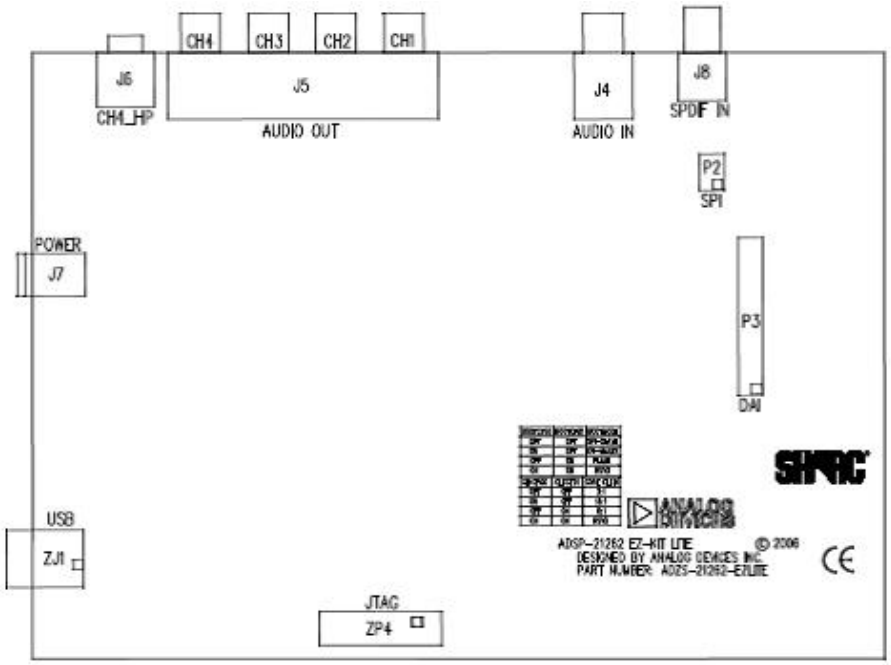


Рисунок 4.21 — ADSP-21262-EZLITE — расположение разъемов и соединителей [7]

– ZJ1 — USB;
– ZP4 — JTAG.

4.1.3 VisualDSP++ — интегрированная среда разработки приложений

Характеристики [9]:

– Широкие возможности редактирования. Среда разработки VisualDSP++ позволяет создавать и изменять исходные файлы, используя многовариантную подсветку синтаксиса языка, применять закладки, метод drag-and-drop и другие стандартные приемы редактирования. Вы можете видеть и редактировать файлы, создаваемые встроенными средствами автоматической генерации кода.
– Гибкое управление проектом. Вы можете задать полное описание программного проекта, включающее необходимые файлы, их зависимости и указать инструментальные средства, которые необходимо использовать при создании проекта. В дальнейшем в это описание можно внести изменения в соответствии с изменившимися требованиями.
– Простой доступ к средствам разработки программного кода. Среда VisualDSP++ позволяет использовать следующие инструменты разработки: компилятор C/C++, VIDL компилятор (компилятор специального языка интерфейса с ранее разработанными программными компонентами), ассемблер, компоновщик (линкер), загрузчик и симулятор – отладчик. Опции, определяющие варианты использования этих инструментальных средств, могут быть заданы как в диалоговых окнах, так и указаны в командной строке в виде ключей. Можно указать опции для каждого отдельного файла и для всего проекта в целом, а впоследствии их изменить.
– Гибкое управление вариантами построения программного проекта. Среда разработки дает возможность управлять построением кода на уровне отдельного файла и всего проекта. VisualDSP ++ позволяет вносить изменения в проект путем добавления новых зависимостей, компилировать и создавать код только новых и измененных файлов. Среда позволяет наблюдать за процессом построения проекта. Если в процессе компиляции и компоновки возникли ошибки, то двойным щелчком мыши можно открыть исходный код файла, указанного в сообщении об ошибке. Исправив ошибку, можно заново откомпилировать файл или проект и начать сессию отладки.
– Поддержка ядра VisualDSP++ (VisualDSP Kernel – VDK). Вы можете добавить к проекту поддержку VDK, для того чтобы структурировать и масштабировать разрабатываемое приложение. Вкладка Kernel окна Project позволяет управлять такими объектами ядра как события, управляющие событиями биты, приоритеты, семафоры и потоки.
– Гибкое управление рабочей средой. Можно создать несколько рабочих сред (пространств) и быстро переключаться между ними. Назначение каждому проекту отдельного рабочего пространства (workspace) позволяет строить и выполнять отладку нескольких проектов в одной сессии.

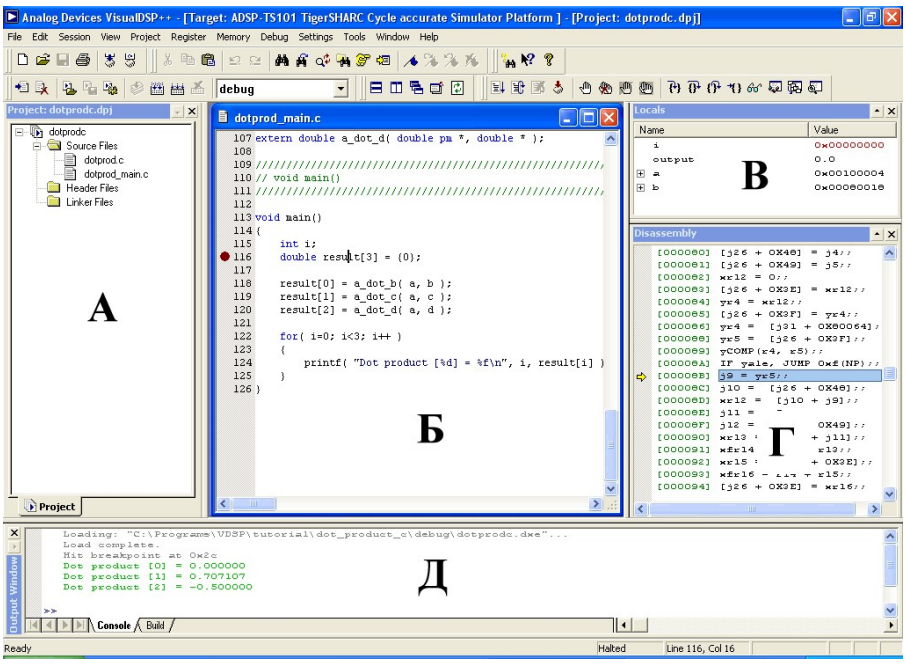


Рисунок 4.22 — Окно с открытым проектом

– Простое переключение между отладкой и внесением изменений. Начав сессию отладки, вы можете свободно переключиться на редактирование исходных файлов, компиляцию, компоновку и выполнить другие отладочные действия.

4.1.3.1 Рабочее пространство — окна

На рис.4.22 представлен вид рабочего пространства среды VisualDSP++ с открытым в ней проектом. Верхняя часть основного окна состоит из строки меню (File, Edit, Session, View, Project, Register, Memory, Debug, Settings, Tools, Window, Help) и панелей инструментов, обеспечивающих быстрый вызов требуемых функций, а также выполнение основных действий по созданию кода и его отладке.

Представленная на этом рисунке раскладка рабочего пространства включает 5 открытых окон (отмечены буквами русского алфавита):

А – окно проекта. Здесь представлено дерево каталогов и файлов, составляющих проект;

Б – окно редактора исходного кода одного их файлов проекта;

В – окно вывода значений локальных переменных в процессе отладки программы;

Г – окно дизассемблера. Здесь отображается исполняемый код программы на языке ассемблера того сигнального процессора, для которого создается данный проект.

Д – одна из вкладок окна сообщений среды VisualDSP, в данном случае — вкладка консольных сообщений с результатами вывода отлаживаемой программы. В это окно выводится также информация о ходе компиляции и компоновки программы и сообщения об ошибках построения исполняемого кода.

4.1.3.2 Начало работы с VisualDSP ++

После запуска приложения VisualDSP ++ должно открыться рабочее окно (рис.4.22). Затем нужно установить режим работы приложения. Режим задается через пункты меню $Session \rightarrow New\ Session$. Если вы не используете отладочный модуль, например, ADSP-BF533 EZLITE, то можно задать режим симуляции. В этом режиме можно работать с инструментальной системой и изучать особенности архитектуры DSP. Для этого достаточно открыть демонстрационные файлы: проекты *.dpj находятся в директории Example среды разработки.

Для работы с реальным устройством задаем режим EZ-KIT-Lite. Но прежде подключимся к модулю через USB-интерфейс. Здесь все просто, поскольку работает технология plug and play (подключи и работай), что упрощает работу с драйверами. Убедившись, что связь между компьютером и отладочным модулем установлна, можно задавать режим "EZ-KIT Lite". Таким образом, создав файл-проект, либо выбрав пример в директории Example, запускаем проект на построение приложения командой меню:$Project \rightarrow Build\ Project$. При этом модули проекта компилируются, затем связываются, размещаются и загружаются в память устройства, например, ADSP-BF533 EZ-KIT Lite. Остается запустить программу на выполнение: выбираем пункт меню $Debug \rightarrow Run$ и наблюдаем работу приложения.

4.2 Лабораторные работы

В данном разделе на основе программ из каталогов «$\ldots \backslash VisualDSP++\backslash$ тип процессора, например, 212XX $\backslash Examples \ldots$» — выбираем соответствующий пример на языке ассемблера или Си. Затем, используя пример в качестве прототипа, выполняем лабораторную работу.

4.2.1 Работа с портами ввода/вывода (GPIO) общего назначения

Здесь нужно продемонстрировать управление светодиодами (LED) посредством кнопок.

4.2.1.1 Задание 1

В этом задании для модуля ADSP-21262-EZLITE нужно написать программу, которая обеспечивает загорание определенных светодиодов (LED) при нажатии на пользовательские кнопки (SW4—1).

Кнопки SW1, SW2, SW3 и SW4 соединены с регистрами IRQ1, IRQ2, — выводы DAI_P19, и DAI_P20 соответственно. Нажатие на определенную кнопку вызывает прерывание, подпрограмма обработки прерывания включает/выключает LED. За основу программы можно взять пример EZkit Push Button (C), который находится в указанном выше каталоге. Фрагмент программы, отвечающий за зажигание светодиодов:

```
#include <def21262.h>
#include <cdef21262.h>
```

```
#define SRUDEBUG  // контроль маршрутизации сигналов
#include <SRU.h>
#include <signal.h>

#define LED1 1
#define LED2 2
#define LED3 4
#define LED4 8

void DAIroutine(int);
void IRQ1_routine(int);
void IRQ2_routine(int);
void handle_LED(int);

main() // основная программа
{
     *pDAI_IRPTL_PRI = SRU_EXTMISCB1_INT|SRU_EXTMISCB2_INT; //это задание п
     *pDAI_IRPTL_RE = SRU_EXTMISCB1_INT|SRU_EXTMISCB2_INT; //демаскирование
     *pSYSCTL |= IRQ1EN|IRQ2EN; // прерывания по фронту сигнала

     asm("#include <def21262.h>");
     asm("bit set mode2 IRQ1E|IRQ2E;");

     interrupt(SIG_DAIH,DAIroutine);
     interrupt(SIG_SPIH,DAIroutine);

     interrupt(SIG_IRQ1,IRQ1_routine);
     interrupt(SIG_IRQ2,IRQ2_routine);

     // назначение выводов SRU_PIN3 (Group D)
     SRU(LOW,DAI_PB19_I); //буфер пина 19  на ввод
     SRU(LOW,DAI_PB20_I); //буфер пина 20  на ввод

     //маршрутизация сигналов MISCB в SRU_EXT_MISCB (Group E)
     SRU(DAI_PB19_O,MISCB1_I); //маршрутизация DAI пина (через буфер) 19 на
     SRU(DAI_PB20_O,MISCB2_I); //маршрутизация DAI пина (через буфер) 20 на
     SRU(LOW,PBEN19_I); // пин 19 enable
     SRU(LOW,PBEN20_I); // пин 20 enable

     for(;;)
```

```
     {}
 }

// подпрограммы обслуживания прерываний
void IRQ1_routine(int sig_int){
handle_LED(LED1);
}

void IRQ2_routine(int sig_int){
handle_LED(LED2);
}

// связывание подпрограмм с LED 3 и  LED 4.
void DAIroutine(int sig_int){
static int interrupt_reg;

interrupt_reg = *pDAI_IRPTL_H;

//проверка  SRU_EXTMISCB1_INT
if ((interrupt_reg & SRU_EXTMISCB1_INT) != 0)
handle_LED(LED3);

//проверка SRU_EXTMISCB2_INT
if ((interrupt_reg & SRU_EXTMISCB2_INT) != 0)
handle_LED(LED4);
}

void handle_LED(int led_value){
*pPPCTL=0;

*pIIPP=(int) &led_value;
*pIMPP=1;
*pICPP=1;
*pEMPP=1;
*pECPP=1;
*pEIPP=0x400000;

*pPPCTL=PPTRAN|PPBHC|PPDUR20|PPDEN|PPEN;
}
```

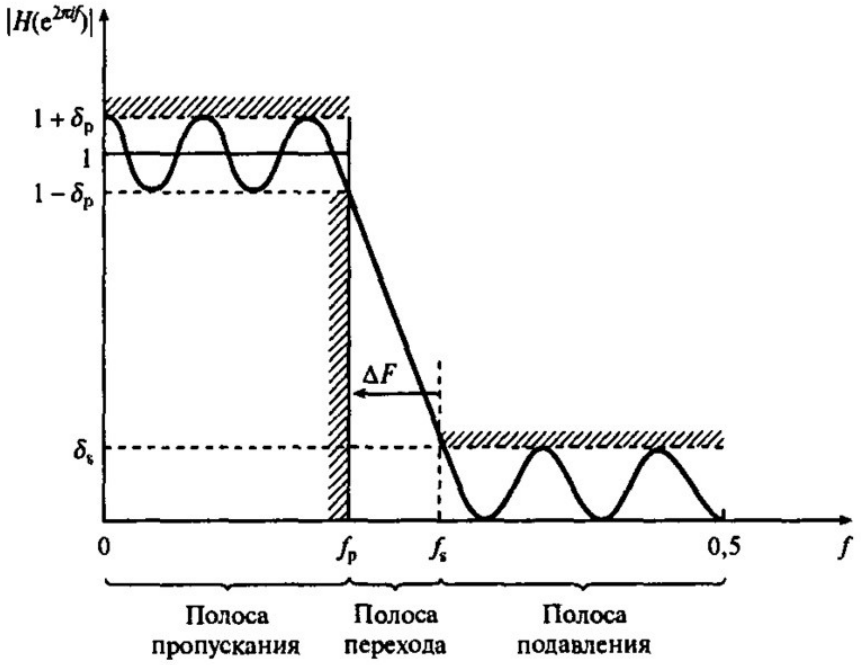


Рисунок 4.23 — ФНЧ: амплитудо-частотная характеристика

Используя этот код, создать проект и запустить приложение на модуле. Пояснить работу кода (проекта) и модифицировать его, изменяя режимы свечения светодиодов.

4.2.1.2 Задание 2

То же что и в задании 1, только для ADSP-BF533-EZLITE.

4.2.2 Задание 3

Здесь рассмотрим реализацию цифровых фильтров: низкочастотный КИХ фильтр первого порядка. Цифровые фильтры — это фильтры с конечной импульсной характеристикой (КИХ) и бесконечной импульсной характеристикой (БИХ), и строят их по нерекурсивной и по рекурсивной схемам соответственно. В данной лабораторной работе будем проектировать низкочастотный КИХ-фильтр. Его частотная характеристика приведена на рис. 4.23.

Далее представлен фрагмент программы, отвечающий непосредственно за преобразование сигнала.

4.2.2.1 Реализация КИХ-фильтра на модуле ADSP-BF533-EZLITE

Перед запуском программы необходимо выставить на отладочной плате переключатели согласно значениям, указанным в файле readme.txt. Программный код и пояснения:

```
#include "Talkthrough.h"// подключение основной библиотеки
#defineW 14        // задание размерности фильтра
Process_Data()
void Process_Data(void)
{
        int i;// задание целочисленной переменной
        float S[W];// задание массива входного сигнала
        float H[W] = {0.0175, 0.0072, 0.0119, 0.0187, 0.0261, 0.0372, 0.038
        0.0369, 0.0322, 0.0256, 0.0181, 0.0109, 0.0049, 0.0008};// задание
        коэффициентов КИХ фильтра
        volatile long unsigned Sum_ = 0; // задание типа и начального значе
        переменной Sum_
        for (i=1; i<W; i++)// цикл для корректировки индексов переменных S
        {
                S[i-1] = S[i] ;
        }
        S[W-1] = iChannel0LeftIn;// запись входного сигнала
        double SUM=0;
        for (i=0; i<W; i++) SUM +=H[i];// нормировка импульсной характерист
        for (i=0; i<W; i++) H[i]/=SUM;
        for(i=0; i<W; i++)// цикл, реализующий перемножение входного сигнал
         с коэффициентами фильтра
        {
                Sum_ += S[i]*H[i];
        }
        iChannel0LeftOut = Sum_;// вывод результата
}
```

В качестве входного сигнала можно использовать сигнал калибровки осциллографа (прямоугольный сигнал частотой 1 кГц и амплитудой 2 В). Этот сигнал подается на вход модуля (In) (см. рис. 4.24). Здесь In — канал 1, Out — канал 2. На выходе Out сигнал после обработки (прямоугольные импульсы после прохождения через фильтр) показан на рис. 4.25.

4.2.2.2 Ввод/вывод аналогового сигнала на модуле ADSP-21262-EZLITE

Здесь за прототип берем пример Block Based TolkThru, устанавливаем переключатели в соответствии с описанием программы и рис. 4.15, подключаем порты входа и

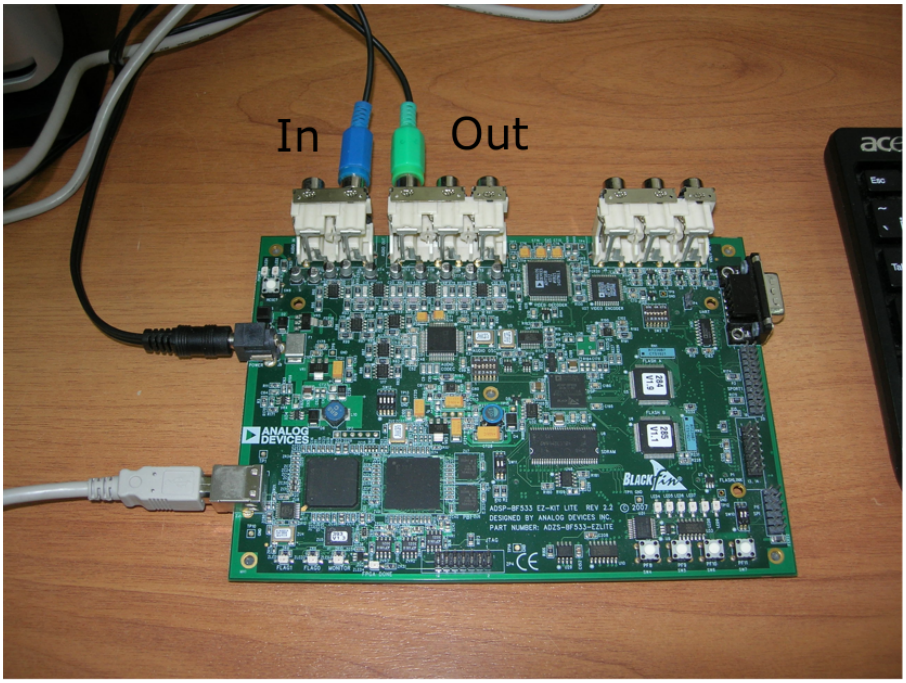


Рисунок 4.24 — Подключение к осциллографу

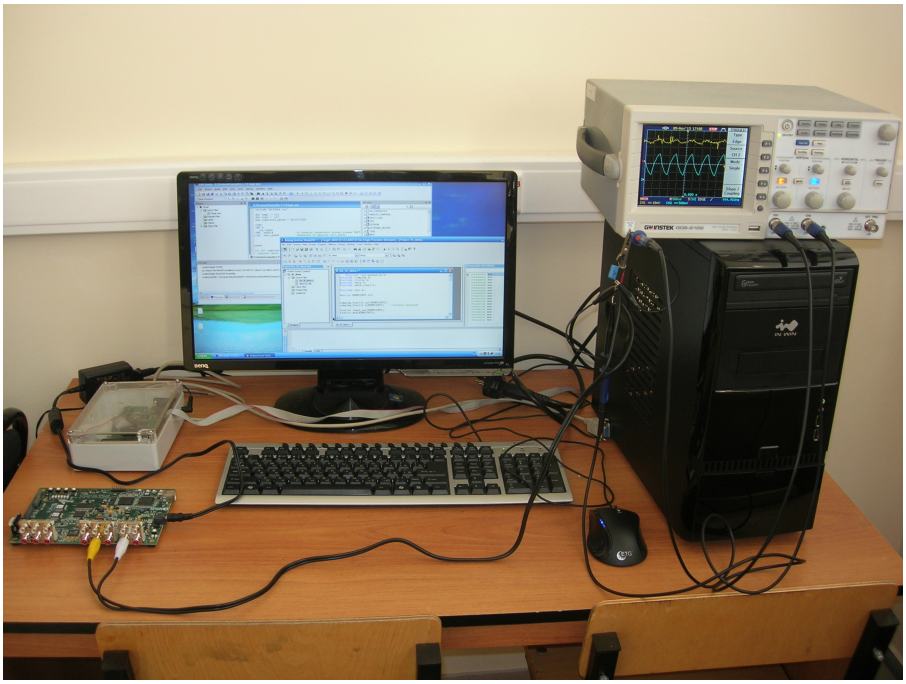


Рисунок 4.25 — Результат

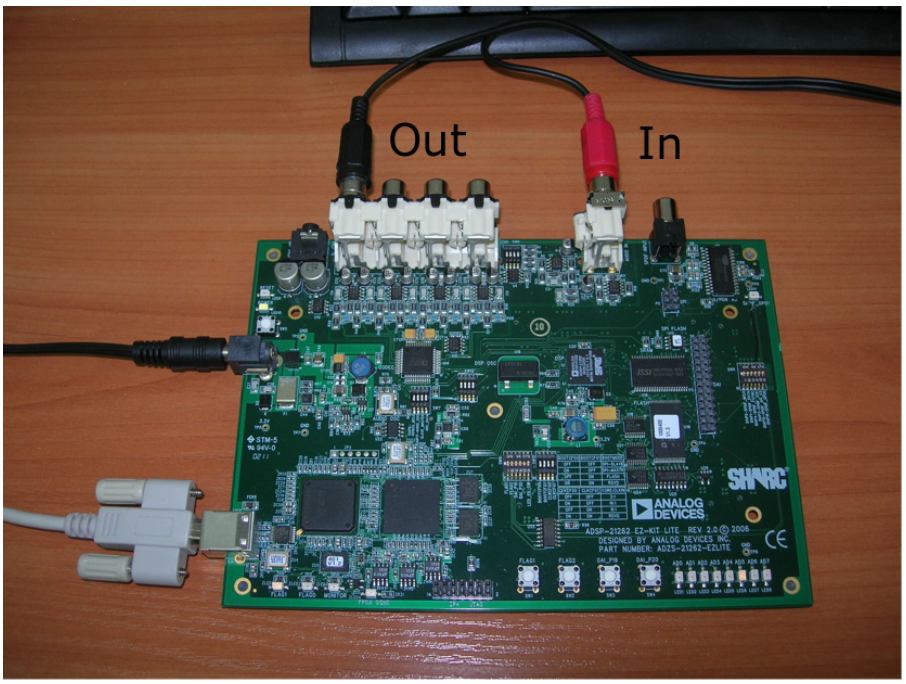


Рисунок 4.26 — Аналоговые вход и выход для ADSP-21262-EZLITE

выхода аналоговых сигналов к осциллографу по аналогии с предыдущим заданием и в соответствии с рисунком 4.26.

4.2.3 Задание 4. ФВЧ

В этом задании нужно реализовать фильтр верхних частот (ФВЧ): рассчитать на основе метода Кайзера значения ядра фильтра и по аналогии с предыдущим заданием (т. е. ФНЧ), используя те же аппаратные средства и ПО, рассмотреть его работу.

4.2.4 Задание 5. BTC

BTC (Background Telemetry Channel) представляет собой способ обмена данными между процессором и хостом (вашим компьютером). BTC, хост и целевое приложение имеют доступ к чтению/записи общих регистров и используются BTC в реальном времени. А это позволяет просматривать входные и выходные данные через интерфейс отладчика тоже в реальном времени. Для преобразования сигнала будем использовать метод экспоненциального скользящего. Это простейший сглаживающий цифровой фильтр. α — параметр фильтра, определяет степень сглаживания. Он может принимать значения от 0 до 1. Чем он меньше, тем сильнее будет сглаживаться входной сигнал. В частных случаях при $\alpha = 0$ фильтр перестает реагировать на изменения входного сигнала, а при $\alpha = 1$ он просто повторяет входной сигнал. Выражение для него выглядит следующим образом:

$$Y(n) = \alpha * Y(n) + (1 - \alpha) * X(n)$$

Для модуля ADSP-BF533-EZLITE перед началом работы необходимо установить переключатели SW9 и SW3 в положение:

$$SW3- > OFF, OFF, OFF, OFF, OFF, ON$$

$$SW9- > ON, OFF, ON, ON, OFF, OFF$$

Освоение BTC начнем с примера в $\ldots\backslash AnalogDevices\backslash VisualDSP5.0\backslash Blackfin \backslash Examples\backslash ADSP-BF533-EZLITE\backslash\ Background\_Telemetry\backslash AudioDemo$:

```
#include "Talkthrough.h" // основные библиотеки
#include <btc.h>
#define ALPHA 0.8 //коэффициент сглаживания
void Process_Data(void)
{
// данные в каналы BTC
intnValue;
extern int BTCLeftVolume;
extern int BTCRightVolume;
//запись данных входного сигнала
nValue = (iChannel0LeftIn >> 8);  //upper 24-bits
nValue = (nValue>>BTCLeftVolume); //амплитуда
btc_write_value(0, (unsignedint*)&nValue, sizeof(nValue));
 //преобразование входного сигнала
iChannel0LeftOut = (ALPHA*iChannel0LeftOut + (1-ALPHA)*((nValue<<8)/2))
//запись данных выходного сигнала
nValue = (iChannel0LeftOut >> 8);  //upper 24-bits
nValue = (nValue>>BTCRightVolume); //амплитуда
btc_write_value(1, (unsignedint*)&nValue, sizeof(nValue));
iChannel0LeftOut = (nValue<< 8);
}
```

Для наблюдения входного и выходного сигналов на мониторе компьютера выполняем следующие действия:

- открываем окно каналов BTC для входного и выходного сигналов (View->Debug Windows->Plot-Restore), рис. 4.27.
- дважды кликнув «мышкой» на файл left.vps и right.vps, открываем окна входного и выходного сигналов. (рис. 4.28 и 4.29).
- для того, чтобы увидеть входной и выходной сигналы, запускаем программу и устанавливаем авто-обновление графиков; для этого кликнем правой кнопкой мыши в окне BTC и выберем функцию “AutoRefresh".
- в результате получим изображения входного и выходного (преобразованного) сигналов.

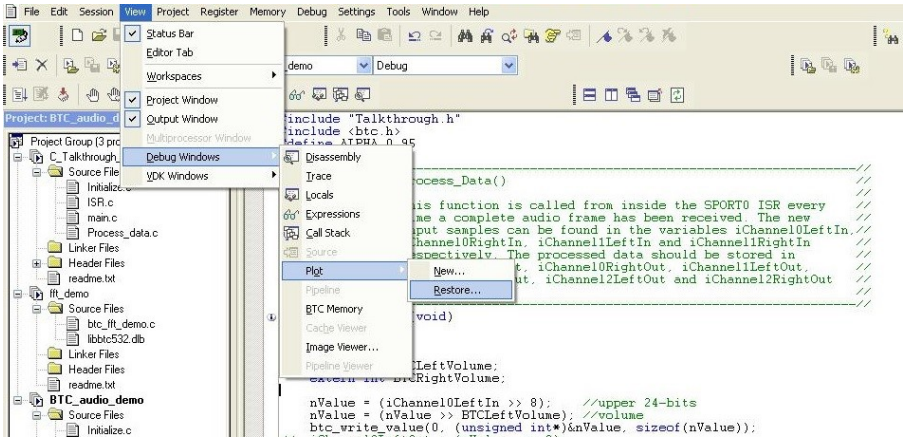


Рисунок 4.27 — BTC – окно установки режима

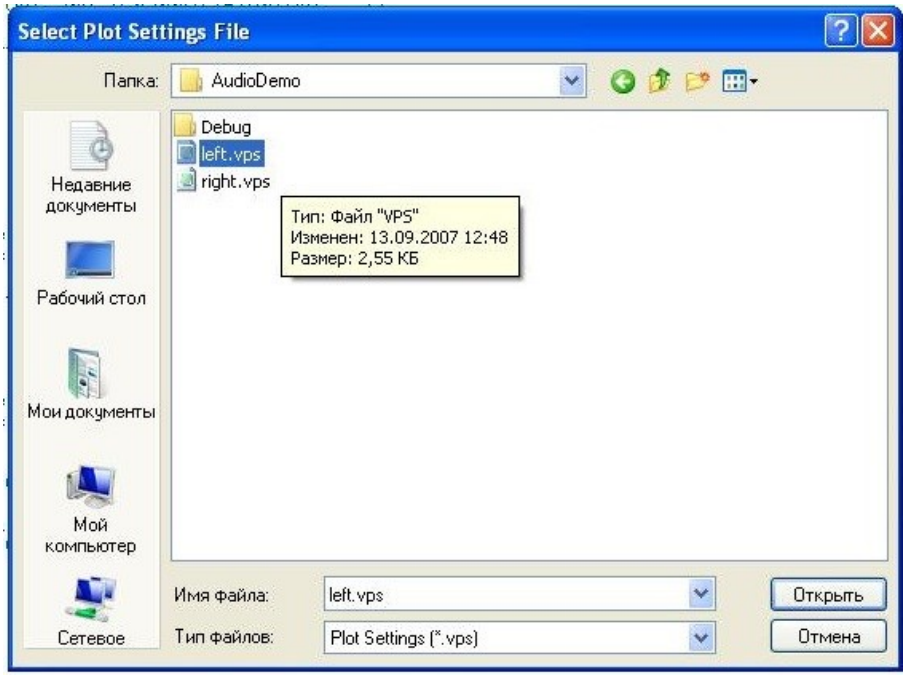


Рисунок 4.28 — Файлы left.vps и right.vps

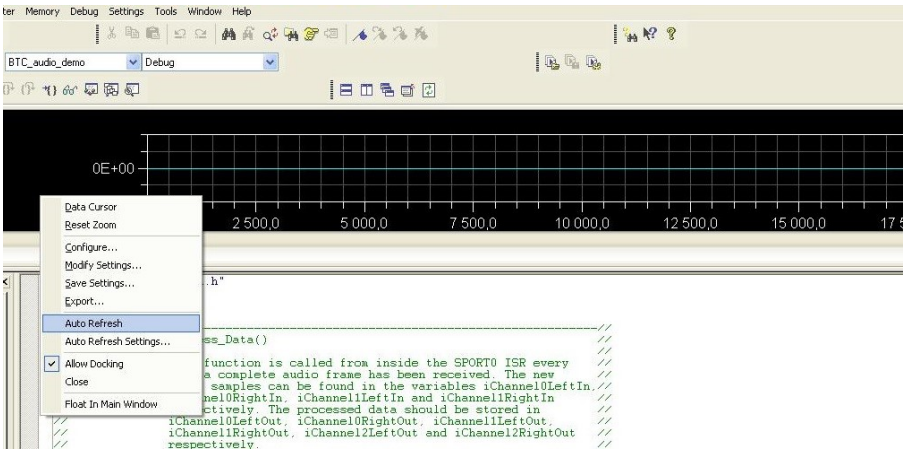


Рисунок 4.29 — Окно канала BTC

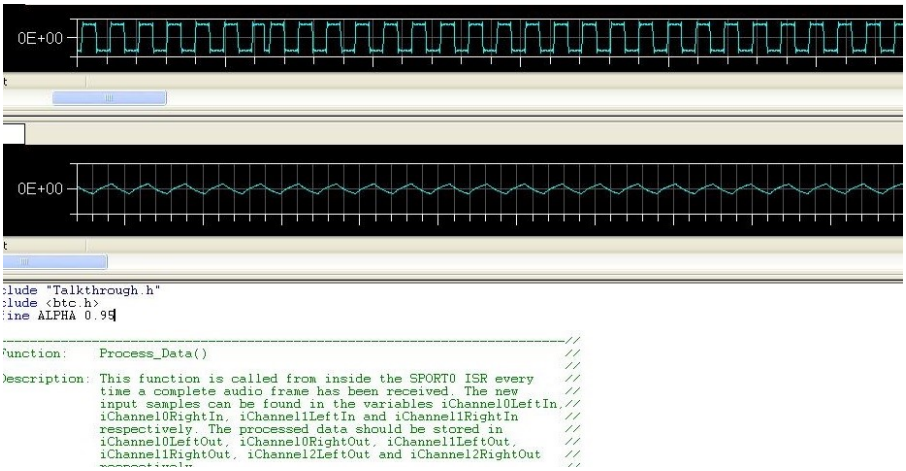


Рисунок 4.30 — Входной и выходной сигналы при значении $\alpha = 0.95$

Рисунок 4.31 — Входной и выходной сигналы при значении $\alpha = 0.5$

– изменяя значение коэффициента сглаживания α, будем получать разный выходной сигнал; на рис. 4.30 и 4.31 представлены изображения входного и выходного сигналов при значении α 0.95 и 0.5, соответственно.

Литература

1. Ярославский Л. П. Цифровая обработка сигналов в оптике и голографии. – Радио и связь, 1987. – 296 с.

2. Гольденберг Л. М. и др. Цифровая обработка сигналов: Учеб. пособ. для вузов / Гольденберг Л. М. – Радио и связь, 1990 – 256 с.

3. Оппенгейм А., Шафер Р. Цифровая обработка сигналов. Издание 2-е, исправленное. – Москва: Техносфера, 2007. – 856 с.

4. Starck J., L., Murtagh F. Image restoration with noise suppression using the wavelet transform / J. L. Starck, F. Murtagh // Astronomy and Astrophysics. – 1994. – 288. – P. 343–348.

5. Richard J. Higgins. Digital Signal Processing in VLSI / Richard J. Higgins, – Prentice Hall, Englewood Cliffs, – 1990. – 591 p.

6. Сайт компании «Analog Devices» [Электронный ресурс]. ADSP-BF531 BF532 BF533. – 2013. – 64 с. Режим доступа: $http : //www.analog.com/media/en/technical - documentation/data - sheets/ADSP - BF531\_BF532\_BF533.pdf$, свободный. – Яз. англ. (дата обращения 9.02.2017)

7. Сайт компании «Analog Devices» [Электронный ресурс]. ADSP-21262 EZ-KIT Lite Evaluation System Manual. – 2004. – 105 с. Режим доступа: $http : //www.analog.com/media/en/dsp - documentation/legacy - evaluation - kit - manuals/62200738ADSP\_21262\_EZ\_KIT\_Lite\_Manual\_Rev\_1.2\_.pdf$, свободный. – Яз. англ. (дата обращения 25.01.2017)

8. Сайт компании «Analog Devices» [Электронный ресурс]. ADSP-BF533 EZ-KIT Lite Evaluation System Manual. – 2012. – 91 с. Режим доступа: $http : //www.analog.com/media/en/dsp-documentation/evaluation-kit-manuals/ADSP-BF533\_ezkit\_man\_rev.3.2.pdf$, свободный. – Яз. англ. (дата обращения 25.04.2017)

9. Сайт компании «Analog Devices» [Электронный ресурс]. VisualDSP++5.0 User's Guide. – 2007. – 441 с. Режим доступа: $http : //www.analog.com/media/en/dsp - documentation/software - manuals/719705850\_ug.pdf$, свободный. – Яз. англ. (дата обращения 1.04.2017)