

# Программирование сигнальных процессоров компании Analog Devices в среде VisualDSP++

## Цифровая фильтрация

**Цифровая фильтрация является важнейшей областью цифровой обработки сигналов. В первую очередь цифровые фильтры отличаются высоким качеством формирования частотной характеристики, стабильностью параметров, простотой изменения параметров амплитудно-частотной характеристики и возможностью адаптации параметров фильтра. Наиболее полно эти преимущества используются в таких областях, как подавление различного рода шумов и помех, распознавание речи, и в особенности для подавления эха в устройствах передачи данных. В рамках этой статьи будут кратко рассмотрены общие подходы к проектированию и реализации цифровых фильтров на базе процессоров ADSP компании Analog Devices, а также пример реализации адаптивного цифрового фильтра с использованием отладочного комплекта ADDS-2181-EZ-KIT-LITE. Эти программы были реализованы совместно специалистами одной из групп, проходивших подготовку, и преподавателями центра.**

**Юрий Юрченко,  
Михаил Петров,  
Вадим Шатилев**

vadim@eltech.spb.ru

### Структура и реализация цифровых КИХ-фильтров

Различают два основных типа фильтров. Цифровые фильтры с конечной импульсной характеристикой (КИХ) реализуются путем весового суммирования  $N$  предшествующих отсчетов входного сигнала. Выходной сигнал фильтра  $y(n)$  имеет вид

$$y(n) = \sum_{k=0}^{N-1} h_k \times x(n-k),$$

где  $h_k$  — весовые коэффициенты фильтра,  $x(n-k)$  — отсчеты входного сигнала.

Цифровые фильтры с бесконечной импульсной характеристикой (БИХ) используют входной  $x(n-k)$  и выходной  $y(n-k)$  сигналы. Выходной сигнал такого фильтра описывается уравнением

$$y(n) = \sum_{k=0}^M b_k \times x(n-k) + \sum_{k=1}^N a_k \times y(n-k),$$

где  $b_k$  и  $a_k$  — весовые коэффициенты фильтра.

Цифровые фильтры БИХ требуют меньших вычислительных затрат по сравнению КИХ, однако они имеют нелинейную фазовую характеристику и при определенных условиях становятся неустойчивыми.

Фильтры КИХ при симметричном выборе весовых коэффициентов  $h_k$  имеют линейную фазовую характеристику и поэтому находят широкое применение в высококачественной аппаратуре. Если фильтр КИХ используется для децимации (понижения) частоты дискретизации в  $n$  раз, то вычислительные затраты также снижаются в  $n$  раз.

Рассмотрим основные методы проектирования цифровых фильтров КИХ. Для обработки сигналов во многих случаях требуется иметь частотную характеристику, близкую к прямоугольной. Для получения

прямоугольной амплитудно-частотной характеристики требуется фильтр с бесконечной импульсной характеристикой, которая физически не реализуема. Для того чтобы ограничить длину импульсной характеристики фильтра, используется метод окна. В практике проектирования фильтров используются окна Хэннинга, Хэмминга, Блэкмана и Кайзера. Эти окна различаются скоростью спада на границах импульсной характеристики и достижимым уровнем ослабления в полосе задержания. Очень подробно этапы проектирования описываются в книге «Теория и применение ЦОС» (Рабинер Л., Голд Б. М. Мир. 1978).

Метод окна является эмпирическим и не гарантирует достижение оптимальных результатов. Используя методы линейного программирования, Ремеза или Паркс-мак-Клеллана можно получить частотные характеристики более высокого качества.

Наиболее известные и описанные в литературе методы проектирования фильтров содержатся в пакете QED-2000. Это достаточно известный программный продукт, предназначенный для разработки цифровых фильтров и обладающий удобным пользовательским интерфейсом. Кроме него могут быть использованы и другие программы, например MatLab.

При проектировании цифрового фильтра необходимо задать следующие параметры:

1. Неравномерность частотной характеристики в полосе пропускания фильтра. Обычно уровень пульсаций характеристики составляет десятые доли дБ.
2. Ширину переходной области между границами полос пропускания и задержания. Эта ширина определяет крутизну ската характеристики фильтра и составляет десятки процентов от ширины полосы пропускания.
3. Величину ослабления в полосе задержания, составляющую обычно десятки дБ.

Количество звеньев фильтра N возрастает при снижении пульсаций в полосе пропускания, повышении крутизны ската частотной характеристики и увеличении ослабления сигнала в полосе задержания.

Проектирование фильтра с помощью пакета QED-2000 начинается с выбора с помощью окна Design необходимого фильтра (КИХ или БИХ) и метода проектирования. Затем производится выбор вида характеристики (ФНЧ, ФВЧ, полосовой и т. д.), задание частот, пульсаций и ослабления. После задания параметров фильтра происходит вычисление числа звеньев фильтра N, его весовых коэффициентов и графиков частотной и фазовой характеристик, весовой функции и переходного процесса.

Сложность вычислительной реализации фильтра определяется числом звеньев N. В цифровых сигнальных процессорах фирмы Analog Devices для вычисления отсчета y(n) требуется приблизительно N команд. Поэтому следует проверить возможность реализации такого числа операций за период дискретизации сигнала. Например, при частоте дискретизации сигнала 8 кГц период составляет 125 мкс. При частоте выполнения команд 50 МГц число команд, выполненных за период дискретизации равно 6250 и необходимо выполнить условие N<6250. Практически требуется иметь некоторый запас команд для обработки прерываний при вводе и выводе данных из процессора.

Если фильтр получается слишком сложным, приходится снижать одно из требований к частотной характеристике и проводить расчет заново.

После получения фильтра с необходимой характеристикой формируем файл коэффициентов с помощью окна Output и Create Coefficient Dataset. На этом этапе мы имеем набор коэффициентов, описывающих импульсную характеристику проектируемого нами фильтра.

Следующим этапом является практическая программная реализация этого фильтра. В данном примере для реализации алгоритма используется процессор серии ADSP-2181 компании Analog Devices. Этот выбор обусловлен тем, что на сегодня это один из массовых и доступных процессоров обработки сигналов. Кроме того, он имеет оптимизированную архитектуру ядра, в том числе для задач цифровой фильтрации. В данных примерах используются такие возможности процессора, как:

- Раздельная организация памяти данных и программ, то есть возможность выборки двух операндов за один цикл.
- Аппаратная поддержка организации циклических буферов.
- Возможность использования многофункциональных команд, которые в частности за один цикл позволяют выполнить следующие действия: умножение, сложение, выборку данных, обновление двух указателей на следующие данные и т. д.

Все эти возможности полностью использованы в приведенной ниже программе.

При программировании фильтра КИХ длиной N в памяти процессора организуются

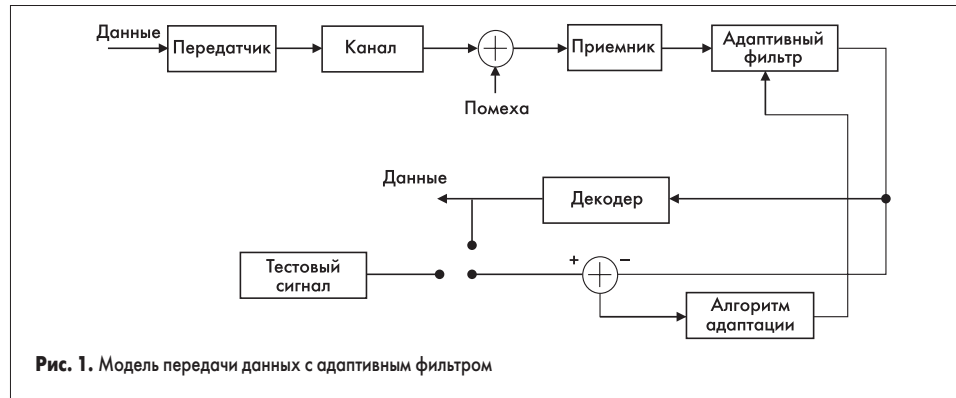


Рис. 1. Модель передачи данных с адаптивным фильтром

**Подпрограмма реализации КИХ-фильтра**

два буфера длиной N для хранения данных the\_data (в памяти данных DM) и коэффициентов фильтра fir1\_coefs (в памяти программы PM). В буфер коэффициентов вводятся данные из соответствующего файла fir4.dat. Благодаря использованию двух секций памяти считывание данных и коэффициентов выполняется одновременно в одном цикле процессора.

Вычисление весовой суммы выполняется с помощью многофункциональной команды, состоящей из умножения с накоплением в аккумуляторе MR и двух обращений к памяти (данных и программы). Так как данные из регистров умножителя MX0 и MY0 считываются в начале выполнения команды, а новые данные поступают в конце выполнения команды, перед запуском цикла необходимо ввести первые значения данных и коэффициентов, а последнее умножение с накоплением выполняется дополнительной командой после завершения цикла. Таким образом, для получения весовой суммы из N слагаемых выполняется цикл из N-1 шага и дополнительное умножение с накоплением вне цикла.

```
.GLOBAL sub;
.GLOBAL in_out;
#define taps 121
.SECTION/DM          buf_var2;
.var/circ  the_data[taps]; /*буфер данных фильтра*/
.SECTION/PM          pm_da;
.var/circ  fir1_coefs[taps] = «fir4.dat»; /*буфер коэффициентов*/

.SECTION/PM          seg_code;
sub: /* подпрограмма инициализации */
    imask = b#0000100001; /*прерывание приема SPORT0 */

    /* организация циклического буфера */
    i0=the_data;      m0=1; l0=taps;
    i4=fir1_coefs;   m4=1; l4=taps;
    rts;
in_out:
    /* основная программа обработки */

    sr1 = dm (rx_buf + 1); /*прием сигнала*/
    dm(i0,m0)=sr1;      /*запись сигнала в буфер*/
    cntnr=taps-1;      /*установка числа циклов*/
    mr=0, mx0=dm(i0,m0), my0=pm(i4,m4);

    /* основной цикл фильтра */
do firloop until ce;
firloop:  mr=mr+mx0*my0(ss), mx0=dm(i0,m0), my0=pm(i4,m4);

    mr=mr+mx0*my0(rnd); /*выполнение N-го
                                умножения-накопления*/

    if mv sat mr;
    dm (tx_buf + 1) = mr1;
    rts;
```

Полная версия данного примера находится в каталоге примеров, созданном после установки VisualDSP++. Для начала работы с ним необходимо открыть файл проекта Fir.dpj из каталога \\Visualdsp\218x\EZ-KITs\2181\Examples\FIRDEMO. Затем необходимо скомпилировать программу и загрузить ее в отладочную плату. Запустив ее на выполнение и подав внешний сигнал с генератора, можно оценить амплитудно-частотную характеристику полученного фильтра.

Необходимо хорошо разобраться в том, как реализованы и работают эти программы — это необходимо для правильного понимания последующих примеров.

**Структура адаптивного фильтра**

Структура КИХ-фильтра удобна для построения алгоритмов адаптивных фильтров. В качестве примера реализации адаптивного фильтра рассмотрим использование фильтра КИХ для коррекции искажений в канале связи при цифровой передаче данных.

На рис. 1 представлена общая модель, используемая при проектировании алгоритмов адаптивной фильтрации.

Перед началом передачи данных и периодически в процессе передачи, при необходимости, производится настройка фильтра путем сравнения эталонного тестового сигнала с принятым сигналом и полученная разность e(n) используется для формирования адаптированных коэффициентов фильтра. Затем начинается прием информации и для адаптации используется сигнал на выходах декодера и адаптивного фильтра.

Алгоритм адаптации реализован с использованием алгоритма стохастического градиента. Ошибка адаптации e(n) определяется как разность тестового сигнала и выхода фильтра

$$e(n) = d(n) - \sum_{k=1}^N h_k(n) \times x(n-k+1)$$

Коррекция параметров фильтра выполняется по правилу

$$h_k(n+1) = h_k(n) + \beta e_0(n) x(n-j+1)$$

Скорость коррекции определяется параметром β. При определении значения β выбирают оптимум между скоростью адаптации и влиянием флуктуационных ошибок.

**Формирование эталонного и искаженного сигналов**

Для демонстрации возможностей этого алгоритма были составлены программы формирования тестовой последовательности, то есть входного сигнала с помехой типа переотражения и адаптивный фильтр на основе стохастического градиента. В качестве тестового сигнала используется сумма синусоидальных сигналов, моделирующих стандартные виды модуляции, используемые в современных модемах. Для формирования помехи использовался этот же сигнал, задержанный по фазе и уменьшенный по амплитуде. Изменение коэффициентов задержки и амплитуды позволило моделировать величину искажений, вносимых в эталонный сигнал, и оценивать возможности фильтра по их коррекции.

Программа формирования тестовой последовательности содержит в буфере string кодовую последовательность, задающую тестовый сигнал, которая проходит фильтр КИХ из 7 звеньев для устранения межсимвольной интерференции. Затем с помощью буфера filtr формируется задержанный на 4 такта сигнал помехи. С помощью прерывания IRQE в регистре AX1 записывается +1 или -1. В зависимости от значения AX1 включаются или выключаются помеха и флаг, зажигающий сигнальный диод. В передающие регистры кодека поступают опорный сигнал и сигнал с помехой типа переотражения. Ниже приве-

ден фрагмент кода программы, формирующей эталонный и искаженный сигнал.

**Реализация адаптивного фильтра**

Программа адаптивного фильтра содержит алгоритм фильтра КИХ, в который первоначально вводятся коэффициенты ФНЧ из файла fir4.dat, вычисленные с помощью пакета QED-2000. При включении помехи на выходе разностного элемента появляется рассогласование и начинается адаптивная подстройка коэффициентов по минимуму рассогласования. Приведенный фрагмент кода программы показывает один из возможных вариантов практической реализации цифрового адаптивного фильтра.

На рис. 2-4, полученных с использованием встроенных средств отображения VisualDSP++, показана импульсная характеристика фильтра до подстройки коэффициентов и после работы алгоритма адаптивной подстройки. На рисунках видно, как изменяется импульсная характеристика фильтра в процессе работы алгоритма адаптивной подстройки. На последнем рисунке представлены результаты работы фильтра. Синим цветом показан исходный сигнал, подаваемый на вход кодека, красным цветом — сигнал с внесенными в него искажениями, а желтым — полученный после включения адаптивного фильтра сигнал ошибки.

```
.GLOBAL in_out;

#define taps 200
.SECTION/DM
.var/circ the_data[7]; /*буфер данных формирующего фильтра*/
.var/circ filtr[taps]; /*буфер данных*/

.SECTION/PM
.var/circ pm_da;
.var/circ fir1_coefs[7]= «sinf.dat»; /*буфер коэф-тов формирующего фильтра*/
.var/circ string[taps] = «string.dat»; /*буфер кодовой последовательности*/

.SECTION/PM
.in_out:
    /*обработка кодовой последовательности формирующим фильтром для устранения межсимвольной интерференции*/
    sr1 = pm (i7,m4);
    dm(i0,m0)=sr1;
    cntr=6;
    mr=0, mx0=dm(i0,m0), my0=pm(i4,m4);
    do fir1loop until ce;
fir1loop: mr=mr+mx0*my0(ss), mx0=dm(i0,m0), my0=pm(i4,m4);
    mr=mr+mx0*my0(rnd);
    if mv sat mr;
    if mv sat mr;
    dm(i3,m1) = mr1; /*запись сформированного кода в буфер данных*/

    /* формирование суммы сигнала и задержанной на m1 помехи*/
    si=dm(i3,m2);
    sr=ashift si by (-1)(hi);
    ay0=sr1;
    ar=ax1; /*проверка включения помехи*/
    ar=pass ar;
    if gt jump jumper;
    ar=mr1+ay0;
    jump jumper2;
jumper: ar=mr1;
jumper2: dm (tx_buf + 1) = mr1; /*код без помехи*/
    dm (tx_buf + 2) = ar; /*код с помехой*/
    rts;
```

```
.GLOBAL in_out;
#define taps 121
.SECTION/DM
.var/circ the_data[taps]; /*буфер данных адаптивного фильтра*/
.var/circ the_data_1[taps_2]; /*буфер данных*/

.SECTION/PM
.var/circ pm_da;
.var/circ fir1_coefs[taps] = «fir4.dat»; /*буфер коэффициентов адаптивного фильтра*/
.in_out:
    dm (tx_buf + 1) = ax1; /*опорный код*/
    dm (tx_buf + 2) = ay1; /*код на выходе адаптивного фильтра*/

    sr1 = dm (rx_buf + 1); /*прием кода с помехой*/
    ax0 = dm(rx_buf+2); /*код без помехи*/
    ay1=dm(i1,m1);
    dm(i1,m0)=ax0;

    /*обработка принятого сигнала адаптивным фильтром*/
    dm(i0,m0)=sr1; /* store sample in data buffer (delay line) */
    cntr=taps-1;
    mr=0, mx0=dm(i0,m0), my0=pm(i4,m4);
    do fir1loop until ce;
fir1loop: mr=mr+mx0*my0(ss), 0=dm(i0,m0), my0=pm(i4,m4);
    mr=mr+mx0*my0(rnd);
    if mv sat mr;
    ar=mr1-ay1; /*сравнение отфильтрованного сигнала с опорным кодом*/
    ay1=ar;

    /*использование вычисленного рассогласования для адаптивной подстройки всех коэффициентов фильтра*/
    ax1=mr1;
    my1=0xff00; /*установка скорости адаптации*/
    mf=ar*my1(rnd), mx0=dm(i0,m0);

    mr=mx0*mf(rnd), ay0=pm(i4,m4);
    cntr=taps;
    do ad until ce;
    ar=mr1+ay0, mx0=dm(i0,m0), ay0=pm(i4,m7);
    pm(i4,m6)=ar, mr=mx0*mf(rnd);
    modify(i0,m2);
    modify(i4,m7);
    rts;
```

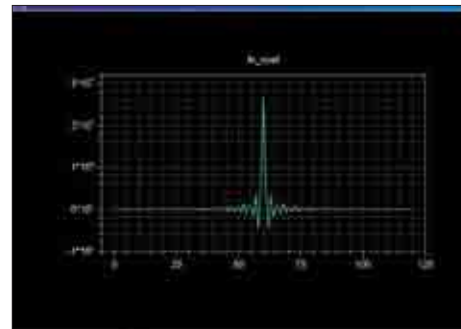


Рис. 2. Начальная импульсная характеристика адаптивного фильтра

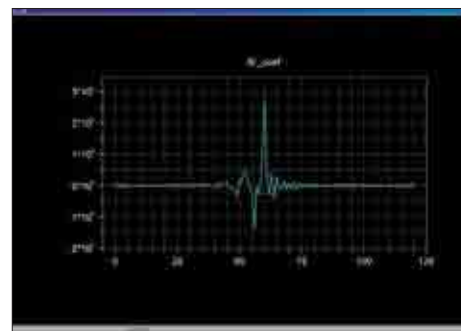


Рис. 3. Импульсная характеристика фильтра после изменения коэффициентов

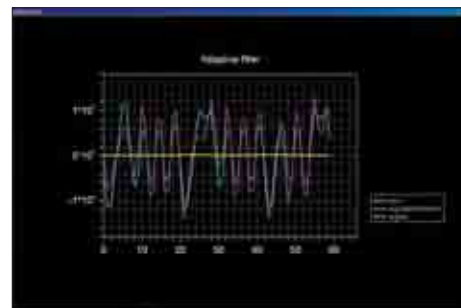


Рис. 4. Пример работы адаптивного фильтра

**Заключение**

Применение адаптивных фильтров выходит далеко за рамки приведенных примеров для задач передачи данных. Это может быть и обработка видеоизображений, и радиолокационные задачи, и различного рода телекоммуникационные приложения, объединенные общей задачей динамической фильтрации сигнала от вносимых в него шума и искажений. В рамках данной статьи были показаны лишь основные подходы к проблеме реализации адаптивных фильтров с использованием сигнальных процессоров компании Analog Devices и среды разработки VisualDSP++. Однако даже эти относительно несложные примеры достаточно успешно решают поставленные задачи, как и показано на приведенных рисунках. В рамках следующей статьи будет рассмотрено, каким образом на базе отладочного комплекта ADDS-2181-EZ-KIT-LITE с использованием среды разработки VisualDSP++ реализуются задачи получения спектра сигнала с использованием алгоритмов быстрого преобразования Фурье.