

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Нижегородский государственный университет им. Н.И. Лобачевского  
Национальный исследовательский университет

**Е.И. Шкелев**  
**В.Н. Бугров**  
**В.В. Артемьев**

**СИНТЕЗ И РЕАЛИЗАЦИЯ  
ЦЕЛОЧИСЛЕННОГО ЦИФРОВОГО ФИЛЬТРА  
В БАЗИСЕ ПЛИС**

*Практикум*

Рекомендовано методической комиссией  
радиофизического факультета для студентов ННГУ,  
обучающихся по специальности и направлению  
010801 «Радиофизика и электроника»,  
010400 «Информационные технологии»

Нижний Новгород  
2015

УДК 681.518

ББК 32.97

Б-90

Б-90 Синтез и реализация целочисленного цифрового фильтра в базе ПЛИС: Составители: Шкелев Е.И., Бугров В.Н., Артемьев В.В. Практикум. – Нижний Новгород: Нижегородский госуниверситет, 2015. – 54с.

Рецензент: к.ф-м.н., В.И. Пройдаков

Практикум подготовлен в соответствии с программой и учебным планом радиофизического факультета по специальностям «Радиофизика», «Информационные технологии». Практикум предназначен для студентов всех форм обучения и содержит понятия программируемого логического устройства, цифрового фильтра, методики его целочисленного проектирования, структуры аппаратной реализации, а так же последовательность операций для программирования ПЛИС, необходимая для реализации заданного типа цифрового фильтра. Данный практикум соответствует учебным программам курсов «Микропроцессоры», «Цифровая обработка сигналов» и «Компьютерные системы автоматизированного проектирования РЭА».

УДК 681.518

ББК 32.97

**© Нижегородский государственный университет  
им. Н.И. Лобачевского, 2015**

## Целочисленное проектирование цифровых фильтров

Цель данной лабораторной работы состоит в изучении современной методологии проектирования цифровых фильтров методами целочисленного нелинейного программирования (ЦНП), способов их построения, в получении практических навыков реализации как рекурсивных, так и нерекурсивных целочисленных цифровых фильтров на программируемых логических интегральных схемах (ПЛИС), а также в освоении навыков автоматизированного измерения характеристик синтезированных фильтров на реальном сигнале.

Цифровая фильтрация является одним из наиболее мощных инструментальных средств цифровой обработки сигналов. Являясь устройствами частотной селекции входного сигнала, цифровые фильтры обычно разрабатываются на основе требований к их частотным характеристикам. В частотной области, определяющей селективные свойства, комплексный частотный коэффициент передачи цифрового фильтра стандартно можно записать как:

$$H(e^{j\omega}) = |H(e^{j\omega})| e^{j\varphi(\omega)}$$

Таким образом основными характеристиками фильтра в частотной области являются:

1. Амплитудно-частотная характеристика (АЧХ), как модуль коэффициента передачи фильтра  $|H(e^{j\omega})|$ ;
2. Фазо-частотная характеристика (ФЧХ), как аргумент коэффициента передачи  $\varphi(\omega)$ ;
3. Фазовая задержка, как прямая характеристика временной задержки фильтром гармонических колебаний:  $\tau_{\text{фаз}} = -\varphi(\omega) / \omega$ ;
4. Время групповой задержки (ГВЗ):  $\tau_{\text{гп}} = -\partial\varphi / \partial\omega$ .

Современные требования к функционированию цифровых фильтров весьма высоки. По условиям работы в современных устройствах цифровой обработки сигналов (ЦОС) цифровые фильтры должны обладать совокупностью требуемых характеристик, таких как АЧХ, ФЧХ, требуемые характеристики групповой или фазовой задержки. Поэтому актуальной является задача изучения методов синтеза цифровых фильтров с учётом совокупности требуемых характеристик. Такой синтез принято называть многофункциональным в отличие от многокритериального синтеза, синтеза только по одной частотной характеристике. Многофункциональный синтез цифровых фильтров с учётом требований их практической реализуемости на современных цифровых платформах возможен в настоящее время только методами нелинейного математического программирования [1-5].

Математическое программирование (МП) - это инвариантная и эффективная методология решения формализованных задач, задач проектирования в частности, методология, максимально ориентированная на современные вычислительные системы. Общая идея МП состоит в привязке решения задачи к чёткому инвариантному математическому признаку – экстремуму функции качества объекта  $F(X)$ , которую также называют целевой функцией, где  $X$  – вектор искомых параметров устройства. Для любой проектной задачи такую функцию всегда можно сформировать исходя из требуемого функционирования устройства (в компьютерных пакетах это обычно делает функциональный редактор). Имея такую целевую функцию, решение задачи синтеза устройства сводят к процедуре минимизации  $F(X)$ , то есть отысканию координат глобального экстремума (оптимальных параметров устройства  $X^0$ ), что обычно делается поисковыми методами [2, 5].

Другим не менее важным требованием к цифровому фильтру является выполнение цифровой фильтрации в реальном времени. Это означает, что все операции алгоритма фильтрации должны выполняться за время, не превышающее период дискретизации входного сигнала. С точки зрения эффективной работы цифрового фильтра в реальном времени, обеспечения минимума затрат (по времени и ресурсам памяти) для расчёта отклика фильтра, особый интерес представляет случай целочисленной дискретизации пространства параметров (коэффициентов) фильтра. В этом случае задача МП трансформируется в задачу целочисленного нелинейного программирования. Целевая функция  $F(\mathbf{IX})$  в задачах ЦНП записана относительно вектора искомых целочисленных параметров фильтра  $\mathbf{IX}$  и наиболее часто формируется в виде аддитивной свёртки (1) частных целевых функций  $f_i(\mathbf{IX})$ , которые определяют выполнение функциональных требований по той или иной частотной характеристике фильтра [3, 4]:

$$F(\mathbf{IX}) = \sum_i \beta_i \cdot f_i(\mathbf{IX}). \quad (1)$$

Коэффициент  $\beta_i$  задает значимость (вес) характеристики ( $i$ -го частотного окна). Сами частные целевые функции  $f_i(\mathbf{IX})$  формирует функциональный редактор пакета синтеза по критерию минимума среднеквадратичного отклонения (ненормированная (2) или нормированная (3) форма) либо в форме минимаксного критерия (4):

$$f_i(\mathbf{IX}) = \sqrt{\frac{1}{p} \cdot \sum_{n=1}^p [Y_n(\mathbf{IX}) - Y_n^T]^2}, \quad (2)$$

$$f_i(\mathbf{IX}) = \sqrt{\frac{1}{p} \cdot \sum_{n=1}^p \left[ \frac{Y_n(\mathbf{IX}) - Y_n^T}{Y_n^T} \right]^2}, \quad (3)$$

$$f_i(\mathbf{IX}) = \max_n \left\{ |Y_n(\mathbf{IX}) - Y_n^T|^2 \right\}, \quad (4)$$

где  $Y_n(\mathbf{IX})$  – текущее значение характеристики на  $n$ -ой дискретной частоте диапазона определения, а  $Y_n^T$  – требуемое значение частотной характеристики.

Таким образом, целочисленный цифровой фильтр (ЦЦФ) обеспечивает максимальное качество цифровой фильтрации (с учётом всей совокупности требуемых характеристик) при гарантированном целочисленном решении  $\mathbf{IX}^0$ , т.е. при использовании только целочисленной арифметики для вычисления отклика фильтра на входное воздействие. Это обстоятельство определяет минимизацию времени расчёта отклика фильтра, так как базовая арифметика любого цифрового вычислителя – это целочисленная арифметика, оперирующая только с целыми числами в двоичном их представлении (битами, байтами, словами) и позволяющая осуществлять вычисления за минимальное время при минимальных ресурсах оперативной памяти. И только отсутствие целочисленных алгоритмов ЦОС вынуждает переходить к вещественной или комплексной арифметике цифровых вычислений, требующих значительных ресурсов как по памяти, так и по тактовой частоте вычислителя с существенным усложнением его структуры (за счёт введения FPU- арифметических сопроцессоров и т.п.) и значительным увеличением энергопотребления. При этом важным фактором, определяющим быстродействие, является разрядность данных, с которыми оперирует целочисленный цифровой фильтр. Снижение разрядности данных даже на 1 бит может сэкономить до 50% оперативной памяти при существенном уменьшении времени расчёта отклика фильтра, поэтому для высокоскоростных специализированных систем ЦОС разрядность имеет большое значение. Идеология ЦНП-синтеза позволяет эффективно проектировать цифровые фильтры с заданной разрядностью представления данных (целочисленных коэффициентов) при максимальном выполнении требований к частотным характеристикам фильтра. Такое целочисленное проектное решение даёт принципиальную возможность реализации фильтров не только на специализированных сигнальных процессорах (DSP), но и на простых и дешёвых микропроцессорных контроллерах (МК), в которых нет возможности вычислений в формате с плавающей точкой. Целочисленное проектное решение является безусловно предпочтительным при аппаратной реализации целочисленного фильтра на ПЛИС, а также на заказных или полужаказных СБИС.

Вопросам моделирования и многофункционального синтеза цифровых целочисленных фильтров и посвящается данная лабораторная работа.

## Построение цифровых фильтров

Цифровой фильтр (рис.1) является дискретной линейной системой, для которой соотношение между входной  $x_n$  и выходной  $y_n$  временными последовательностями определяется разностным уравнением

$$y_n = -\sum_{k=1}^N \frac{a_k}{a_0} \cdot y_{n-k} + \sum_{k=0}^N \frac{b_k}{a_0} \cdot x_{n-k} , \quad (5)$$

а передаточная функция – его z-преобразованием [3].

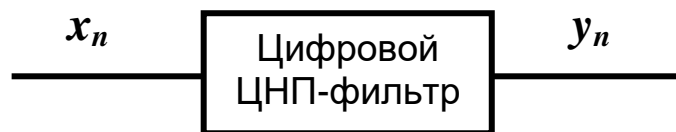


Рис.1 Цифровой фильтр

Принципиальной особенностью целочисленных фильтров, как уже было отмечено выше, является принадлежность его коэффициентов  $b_k$  и  $a_k$  знакопеременному ряду целых чисел, который может быть как натуральным, так и биномиальным (для нормирующего коэффициента  $a_0$ ). Интервал изменения коэффициентов определяется разрядностью представления данных в целочисленной цифровой платформе. Уравнение (5) можно трактовать и как расчетный алгоритм, в котором задержанные величины входной и выходной последовательностей умножаются на постоянные целочисленные коэффициенты  $b_k$  и  $a_k$  соответственно и результаты умножения суммируются.

По своему построению ЦЦФ могут быть реализованы как по рекурсивной, так и по нерекурсивной схемам.

### *Рекурсивные целочисленные фильтры*

Фильтры, которые описываются полным разностным уравнением (5), принято называть **рекурсивными** цифровыми фильтрами, так как в вычислении текущих выходных значений участвуют не только входные данные, но и значения выходной последовательности, вычисленные в предшествующих циклах расчетов.

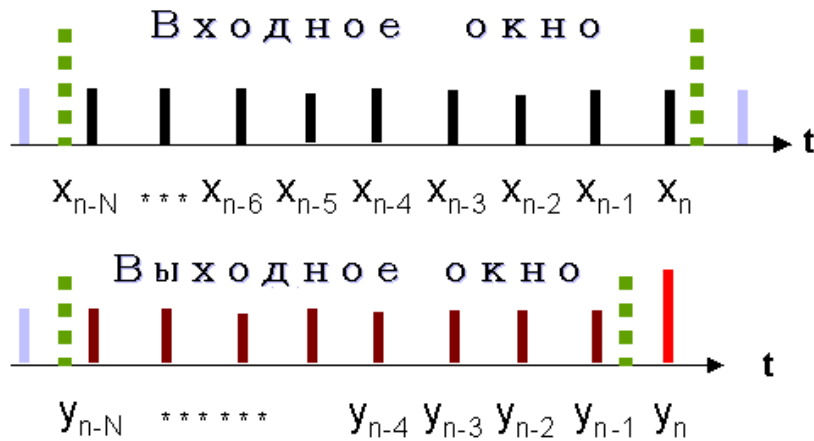


Рис. 2. Входное и выходное окно рекурсивного фильтра

Интервал суммирования по  $k$  получил название "окна" фильтра. Входное окно фильтра составляет  $N+1$  отсчётов (рис. 2), выходное –  $N$  отсчётов, при этом значение  $N$  определяет порядок рекурсивного фильтра. Наличие обратной связи (рекурсии) в (5) определяет бесконечный характер импульсной характеристики фильтра (поэтому рекурсивные фильтры также часто называют БИХ-фильтрами или в английской транскрипции ПИР-фильтрами (infinite impulse response)), причём его частотный коэффициент передачи

$$H(e^{j\omega}) = A \frac{\prod_{i=1}^N (1 - z_i e^{-j\omega})}{\prod_{i=1}^N (1 - p_i e^{-j\omega})}$$

полностью описывается распределением полюсов и нулей в  $z$ -плоскости. Если система устойчива, то все полюсы  $p_i$  должны лежать внутри единичного круга [3, 6]. Таким образом, условие устойчивости рекурсивного фильтра может быть записано как система неравенств (функциональных ограничений) по всем полюсам  $p_i$  коэффициента передачи в  $z$ -плоскости:

$$|Zp_i| < 1. \quad (6)$$

Линейно-разностное уравнение (5) соответствует прямой форме аппаратной реализации ЦЦФ. Однако наиболее выгодной как рекурсивного, так и для нерекурсивного фильтров является последовательная форма построения в виде каскадного включения звеньев второго порядка. Каскадная структура является наилучшей, поскольку:

- позволяет реализовывать ряд передаточных функций цифрового фильтра небольшим набором основных функциональных элементов (звеньев) низкого порядка;

- чувствительность характеристик фильтра к изменению параметров (коэффициентов) наименьшая при последовательной структуре построения фильтра;

- каскадная структура удобна в случае подстройки после синтеза, поскольку каждое звено фильтра изолировано друг от друга.

Поэтому в настоящее время построение рекурсивных фильтров в форме каскадного соединения (рис. 3) звеньев второго порядка прямой или обращённой формы на практике используется наиболее часто.

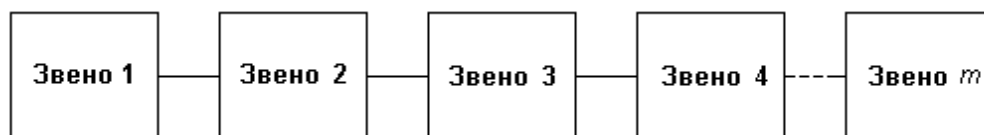


Рис. 3. Каскадная (последовательная) форма построения

Передаточная функция для рекурсивного ЦЦФ, состоящего из каскадного соединения  $m$ -звеньев второго порядка ( $m=N/2$ ), имеет следующий вид:

$$H(z) = \prod_{i=1}^m \frac{b_{0i} + b_{1i}z^{-1} + b_{2i}z^{-2}}{a_{0i} + a_{1i}z^{-1} + a_{2i}z^{-2}}, \quad (7)$$

где комплексная переменная

$$z = e^{j\omega}, \text{ а } \omega = \frac{2\pi f}{F_d} - \text{цифровая частота.}$$

Из соотношения (7) легко получается разностное уравнение для одного звена рекурсивного ЦЦФ:

$$y_n = (b_0x_n + b_1x_{n-1} + b_2x_{n-2} - a_1y_{n-1} - a_2y_{n-2})/a_0 \quad (8)$$

Как видно из (8), при вычислении отклика фильтра должна выполняться операция деления на целочисленный коэффициент  $a_0$ , которая может быть реализована операцией сдвига при условии принадлежности каждого  $i$ -го коэффициента биномиальному целочисленному ряду:

$$a_{0i} \in \{2^q\}, \quad q = \overline{0, W_k} \quad i = \overline{1, m}, \quad (9)$$

где  $W_k$  – длина битового слова целочисленных коэффициентов фильтра.

На рис. 4 приведена типичная структура звеньев рекурсивного целочисленного фильтра, соответствующая разностному уравнению (8). Как видно, для вычисления отклика фильтра  $y_n$  кроме традиционных операций сложения, умножения и задержки на такт присутствует операция сдвига на  $B = \log_2 a_0$  бит, с помощью которой, как уже сказано, реализуется целочисленное деление на биномиальный нормирующий коэффициент  $a_0$ .



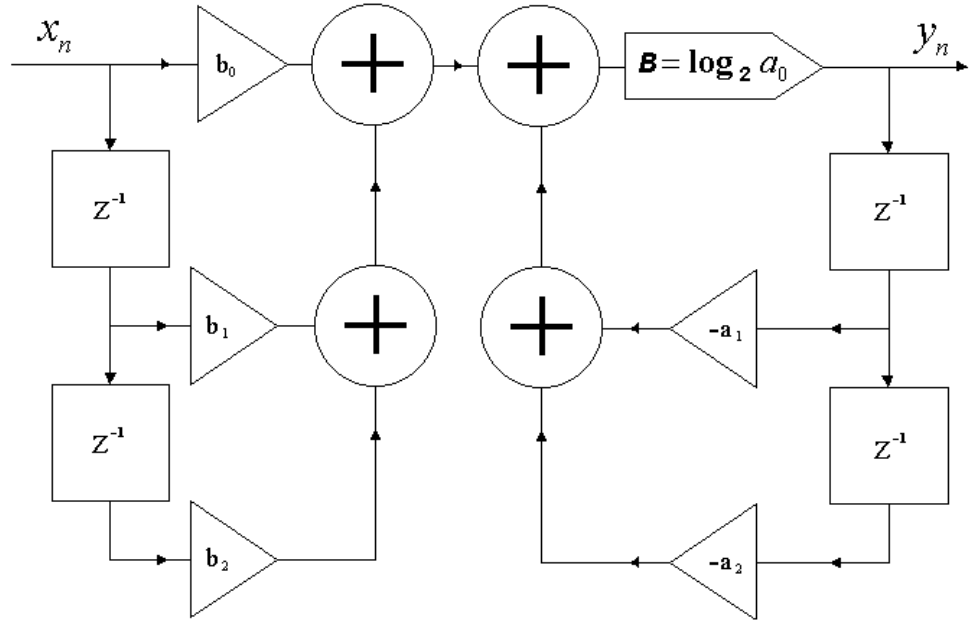


Рис. 4. Структура звена целочисленного БИХ-фильтра

Задачу целочисленного нелинейного программирования при машинном синтезе рекурсивного ЦЦФ можно записать так:

$$F^o(IX^o) = \min F(IX) \quad (10)$$

$$IX \in I^{6m}$$

$$\begin{aligned} -2^{W_k} - 1 &\leq a_{di} \leq 2^{W_k} - 1 & d=1,2 \quad i=\overline{1,m} \\ -2^{W_k} - 1 &\leq b_{di} \leq 2^{W_k} - 1 & d=0,2 \quad i=\overline{1,m} \end{aligned} \quad (11)$$

$$a_{0i} \in \{2^q\}, \quad q = \overline{0, W_k} \quad i = \overline{1, m}, \quad (12)$$

$$|Z_{pi}| < 1 \quad i = \overline{1, m}, \quad (13)$$

$$K_i^{\min} \leq |K_i(e^{j\omega})| \leq K_i^{\max} \quad i = \overline{1, m}, \quad (14)$$

где  $m$  - число звеньев второго порядка,

$d$  - индекс коэффициента передаточной функции звена (7),

$K_i^{\min}, K_i^{\max}$  - границы изменения коэффициента усиления  $i$ -го звена.

Экстремальная задача синтеза (10) записана относительно целочисленного пространства  $I^{6m}$  параметров (коэффициентов фильтра), размерностью  $6m$ . Ограничения (11) задают границы изменения этих целочисленных коэффициентов, а соотношение (12) определяет принадлежность коэффициентов  $a_{0i}$  биномиальному ряду (9). Функциональные ограничения (13) контролируют в процессе синтеза условие устойчивости рекурсивного фильтра по всем полюсам коэффициента передачи, а ограничения (14) определяют масштабирование коэффициентов передачи звеньев каскадных

фильтров в заданный интервал [6 –8]. Многофункциональное задание целевой функции определяется соотношением (1), хотя здесь могут быть применены и другие известные способы скаляризации векторных задач [2, 5]. Поисковое итеративное решение экстремальной задачи (10) в заданном пространстве параметров осуществляет программный алгоритмический комплекс [4, 5], обращаясь к модельному блоку программы для расчёта текущих функциональных характеристик фильтра. Вектор  $\mathbf{IX}^0$ , минимизирующий скалярную целевую функцию  $F(\mathbf{IX})$  на множестве допустимых целочисленных решений (11) и (12), является эффективным решением задачи параметрического синтеза рекурсивного целочисленного фильтра.

### *Нерекурсивные целочисленные фильтры*

В нерекурсивных фильтрах текущие значения выходной последовательности  $y_n$  вычисляются через прямую линейную свёртку (15) значений входной КИХ-последовательности  $x_n$

$$y_n = \sum_{k=0}^N \frac{b_k}{a_0} \cdot x_{n-k}, \quad (15)$$

поэтому такие фильтры всегда устойчивы и имеют конечную импульсную характеристику (finite impulse response - FIR). Входное окно КИХ-фильтра (рис. 5) составляет  $N+1$  отсчётов, при этом значение  $N$  определяет порядок фильтра. Фильтр является односторонним каузальным, т.е. причинно обусловленным текущими и "прошлыми" значениями входного сигнала, и выходной сигнал не опережает входного. Каузальный фильтр может быть реализован физически в реальном масштабе времени.



Рис. 5 Входное окно нерекурсивного фильтра

Принципиальной особенностью является принадлежность коэффициентов  $b_k$  и  $a_0$  знакопеременному ряду целых чисел, который может быть как натуральным, так и биномиальным (для нормирующего коэффициента  $a_0$ ). Интервал изменения коэффициентов при этом определяется длиной битового слова  $Wk$  целочисленных коэффициентов фильтра.

Передаточная функция каскадного соединения  $m$ -звеньев второго порядка нерекурсивного ЦЦФ может быть записана так:

$$H(z) = \prod_{i=1}^m \frac{b_{0i} + b_{1i}z^{-1} + b_{2i}z^{-2}}{a_{0i}}. \quad (16)$$

Уравнение одного звена фильтра имеет вид:

$$y_n = (b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2}) / a_0, \quad (17)$$

где коэффициент  $a_0$  во всех  $m$  звеньях также принадлежит биномиальному целочисленному ряду (9). На рис. 6 приведена типичная структура звеньев нерекурсивного ЦЦФ.

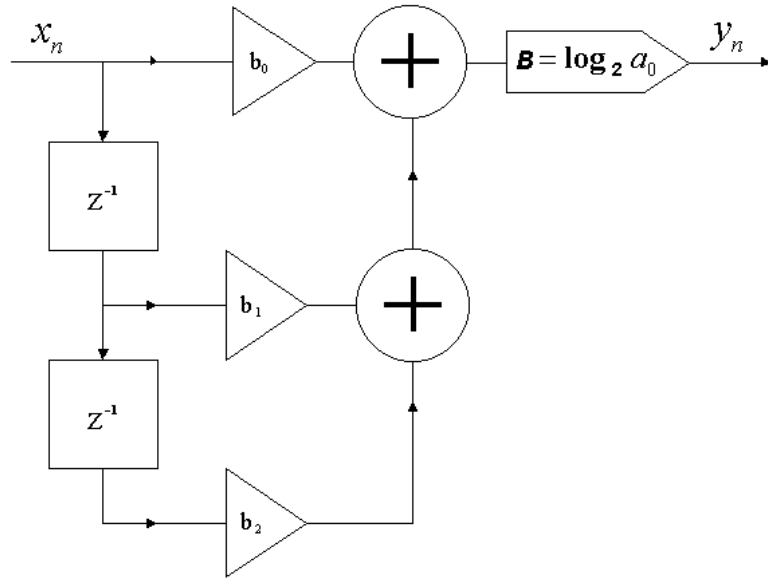


Рис. 6. Структура звена целочисленного КИХ-фильтра

Постановка задачи целочисленного нелинейного программирования для машинного синтеза нерекурсивного ЦЦФ выглядит следующим образом:

$$F^o(IX^o) = \min_{IX \in I^{4m}} F(IX) \quad (18)$$

$$-2^{Wk} - 1 \leq b_{di} \leq 2^{Wk} - 1 \quad d=0,2 \quad i=\overline{1,m}, \quad (19)$$

$$a_{0i} \in \{2^q\}, \quad q = \overline{0, W_k} \quad i = \overline{1, m}, \quad (20)$$

$$K_i^{\min} \leq |K_i(e^{j\omega})| \leq K_i^{\max} \quad i = \overline{1, m}, \quad (21)$$

где  $m$  - число КИХ-звеньев второго порядка,

$d$  - индекс коэффициента передаточной функции звена (16).

Экстремальная задача синтеза (18) записана относительно целочисленного пространства коэффициентов фильтра  $I^{4m}$ , размерностью  $4m$ . Ограничения (19) задают границы изменения этих целочисленных коэффициентов, а соотношение (20) определяет принадлежность коэффициентов  $a_{0i}$  биномиальному ряду. Функциональные ограничения (21) масштабируют усиление звеньев в каскадном КИХ-фильтре в заданный интервал. Многофункциональное задание целевой функции здесь также определяется соотношением (1). Вектор  $IX^0$ , минимизирующий целевую функцию  $F(IX)$  на множестве допустимых решений (19) и (20), является эффективным решением задачи параметрического синтеза нерекурсивного фильтра.

Таким образом, синтез как рекурсивного, так и нерекурсивного ЦЦФ осуществляется одной и той же методологией. При этом для расчёта отклика фильтра используется минимальное количество базовых операций, причём все эти операции целочисленные. Расчёт отклика фильтра на реальном сигнале идёт очень быстро, прямо по определению (модели) фильтра (8) и (17), используя для этого вычисленные заранее целочисленные коэффициенты.

Другим важным достоинством целочисленных фильтров является отсутствие процедуры квантования данных (как коэффициентов фильтра, так и результатов промежуточных вычислений) в ходе расчёта отклика фильтра в реальном времени, а, следовательно, и отсутствие всех негативных последствий квантования данных, свойственных цифровым фильтрам с вещественной арифметики вычислений в формате с фиксированной точкой [6 - 8], полученных, например, билинейным преобразованием аналогового прототипа или методом частотной выборки. Такими последствиям квантования прежде всего являются появление шумов квантования, вызывающих искажение частотных характеристик фильтра, необходимость масштабирования коэффициентов фильтра, а также возможность появления малых предельных циклов при квантовании результатов внутренних вычислений. То есть квантование, как необходимая процедура округления вещественных коэффициентов после их нахождения билинейным преобразованием, в ЦЦФ заменена целочисленной дискретизацией многомерного пространства коэффициентов перед синтезом фильтра с получением целочисленного решения (оптимального вектора целочисленных коэффициентов  $IX^0$ ) с нулевой ошибкой его реализации на цифровой платформе

или кристалле с заданной длиной  $W_k$  слова целочисленных коэффициентов.

Что же касается результатов необходимых для расчёта отклика ЦЦФ промежуточных вычислений, то все они также являются целочисленными, и при заданной битовой разрядности квантования входного сигнала  $W_x$  (в аналого-цифровом преобразователе, например) легко выделить внутренний аккумуляторный регистр с разрядностью

$$W_{ak} = W_x + W_k + 2 \text{ [бит]}$$

для хранения результата целочисленного умножения с накоплением, осуществляемого по алгоритмам (8) или (17). Колебаний переполнения, то есть возникновения больших предельных циклов, вызванных переполнением разрядной сетки регистра-аккумулятора, при таком расчёте его разрядности практически никогда не возникает, особенно если учесть, что накопление результата целочисленного умножения осуществляется алгебраически, с учётом знака слагаемых, что существенно понижает разрядность итогового результата.

Таким образом, все негативные проблемы квантования, присущие цифровым фильтрам с вещественной арифметикой вычислений, отсутствуют в целочисленных цифровых фильтрах ввиду отсутствия самой процедуры квантования данных при реализации алгоритма целочисленной фильтрации.

В каскадных структурах цифровых фильтрах необходимо, как известно, каскадное масштабирование сигнала, что позволяет цифровому фильтру работать в широком динамическом диапазоне изменения входного сигнала. В каскадных фильтрах с вещественной арифметикой вычислений масштабирование сигнала обычно осуществляется по методике Джексона путём введения в структуру звена специального масштабирующего множителя с коэффициентом умножения  $\lambda$ , равного степени числа два, при котором не возникают эффекты переполнения. То есть здесь масштабирующие умножения в звеньях сводятся к простым сдвигам [6, 7]. Но точно такая же процедура регистрового сдвига является последней в расчёте отклика целочисленного звена, то есть вполне может быть использована и для масштабирования его усиления. Однако расчёт такого масштабирования целочисленного звена гораздо легче осуществлять не применением, например,  $L_p$ -нормы, а прямым введением требования обеспечения малого разброса коэффициентов передачи отдельных звеньев непосредственно в ходе ЦНП-синтеза каскадного ЦЦФ. Формально требование масштабирования усиления записываются двусторонними функциональными ограничениями (11) и (21) экстремальных задач синтеза.

## Учебная компьютерная программа анализа и синтеза цифровых фильтров

Программа позволяет осуществлять параметрический синтез как рекурсивных (IIR), так и нерекурсивных (FIR) цифровых фильтров различного порядка в широкой области допустимых изменений целочисленных параметров (коэффициентов) фильтра, проводить подробный анализ полученного оптимального решения в частотной области, выводить на печать графики частотных характеристик синтезированного фильтра, а также формировать файл протокола решения текущей задачи синтеза. Блок-схема компьютерной программы приведена на рис. 7.

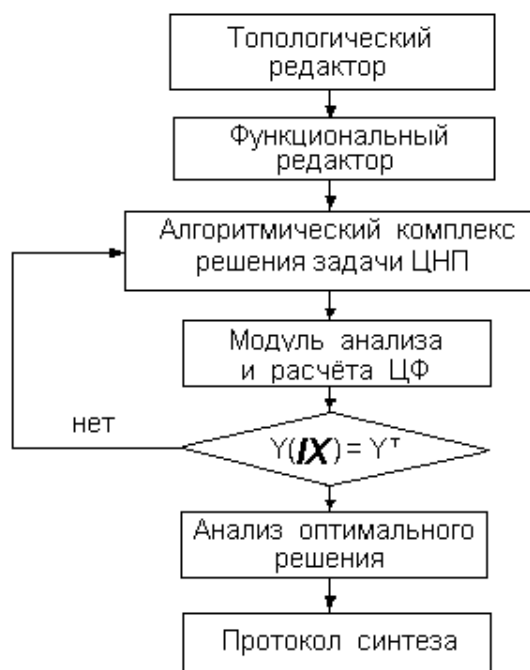


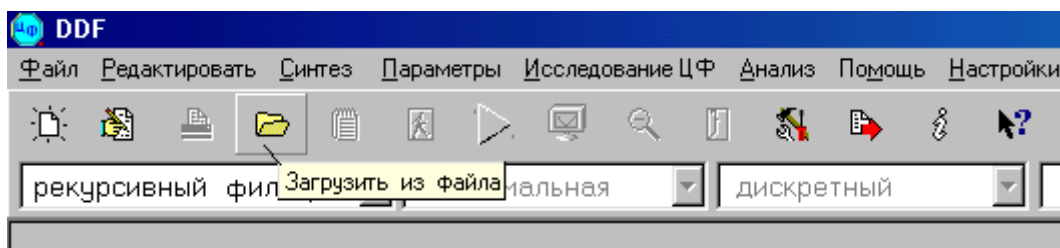
Рис. 7. Блок-схема учебной программы синтеза

Рассмотрим взаимодействие основных модулей программы в контексте выполнения основных этапов решения конкретной задачи синтеза ЦЦФ.

На первом этапе формируется типовой топологический файл задания на синтез `name.top`, имя которого определяет пользователь. Для выполнения конкретной лабораторной работы в программе сформированы типовые файлы заданий на синтез целочисленных фильтров:

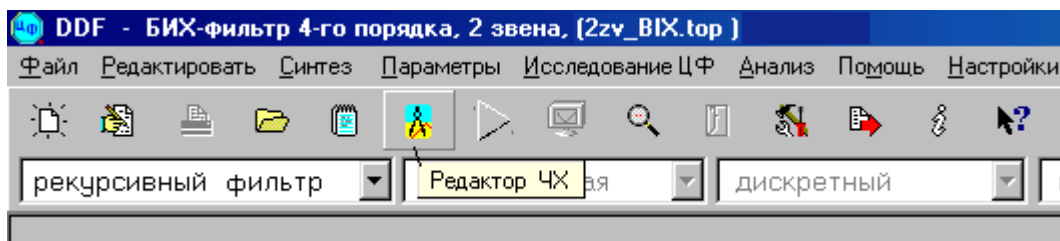
- а) нерекурсивного второго порядка (файл **FIR\_2p.top**),
- б) нерекурсивного четвёртого порядка (файл **FIR\_4p.top**),
- в) рекурсивного фильтра второго порядка (файл **IIR\_2p.top**) и
- г) рекурсивного фильтра четвёртого порядка (файл **IIR\_4p.top**).

В данных файлах содержится описание структуры синтезируемого фильтра, определяются границы изменения его варьируемых коэффициентов и их начальное значения, указывается порядок и тип синтезируемого ЦЦФ. Формирование и редактирование файла задания на синтез осуществляет топологический редактор программы, вызов которого осуществляется соответствующей «горячей» кнопкой основного меню программы. После редактирования файл задания необходимо загрузить в программу на выполнение:



После этого исходные данные можно просмотреть на панели варьируемых параметров.

На втором этапе необходимо осуществить ввод требуемых характеристик синтезируемого фильтра и сформировать целевую функцию в форме (1). Для этого служит графический редактор функциональных характеристик (функциональный редактор), который необходимо вызвать из основного меню программы следующим образом:

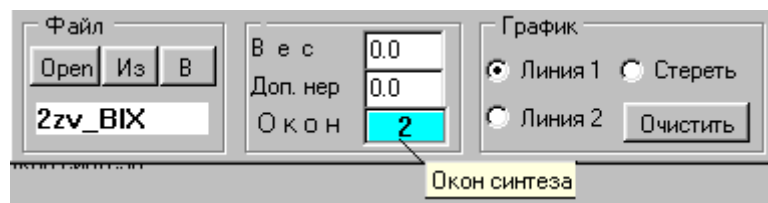


На переднюю панель функционального редактора выведены:

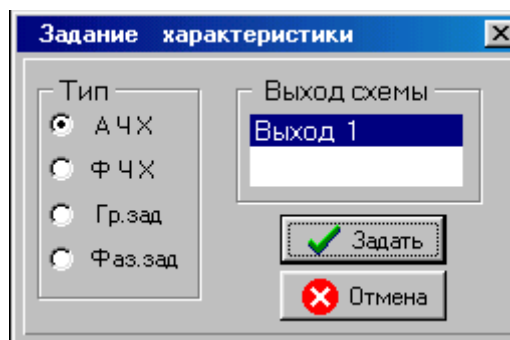
- графическое поле ввода графика требуемой характеристики фильтра;
- элементы управления, задающий тип характеристики и номер частотного окна;
- поля ввода веса характеристики и допустимой её неравномерности;
- элемент управления для задания типа сходимости в каждой точке оцифровки;
- элементы графического редактирования текущего окна редактора.

Для ввода требуемых характеристик ЦЦФ можно рекомендовать следующую последовательность работы в функциональном редакторе:

1. Задание числа используемых функциональных окон (вводимых частотных характеристик синтезируемого фильтра):



2. Задание типа частотной характеристики окна:



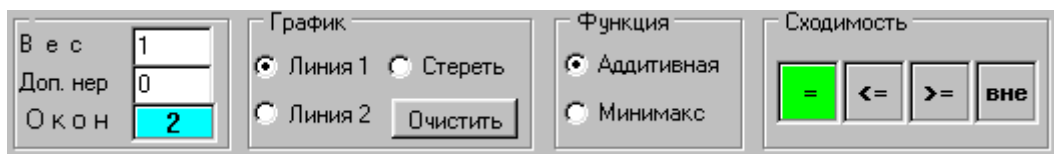
3. Задание границ функционального окна по частоте, выбранной размерности (Гц, кГц или МГц) и типа частотной шкалы (ЛИН или ЛОГ) окна:



4. Задание границ функционального окна по Y, размерности и типа шкалы (например, ЛИН или ДБ для АЧХ):

5. Кусочно-линейная (режим «Линия 1») отрисовка требуемой ЧХ фильтра с помощью левой клавиши мыши. График требуемой характеристики должен занимать всю частотную ось и не должен иметь разрывов. Для ввода сложных графиков рекомендуется использовать шаблоны характеристик функционального редактора.

6. Задание типов частных критериев (типа сходимости текущей характеристики к требуемой) в каждой точке оцифровки ЧХ. Типы критериев задаются цветом с помощью кнопочной панели СХОДИМОСТЬ:



7. Формирование целевой функции окна в аддитивной (2) или минимаксной (4) форме задаётся с помощью кнопочной панели ФУНКЦИЯ;

8. Указание веса текущего функционального окна в поле «Вес»;



9. Оцифровка введенной частотной характеристики осуществляется нажатием кнопки «Оцифровка». После оцифровки введенную характеристику рекомендуется сохранить на диске (панель ФАЙЛ).

В приведенной выше последовательности необходимо осуществить ввод требуемых характеристик ЦЦФ во все заказанные в п.1 окна, после чего завершить работу в функциональном редакторе (кнопка «Выход»).

На третьем этапе программный алгоритмический комплекс (см. рис.7) осуществляет поисковое итеративное решение экстремальной задачи синтеза (10) или (18) в заданном пространстве целочисленных варьируемых коэффициентов фильтра, обращаясь к модельному блоку программы для расчёта текущих функциональных характеристик фильтра по заданной его модели. Старт синтеза осуществляется нажатием соответствующей «горячей» кнопкой основного меню программы.

На четвёртом этапе осуществляется подробное исследование найденного эффективного решения задачи ЦНП-синтеза в модуле анализа пакета (кнопка основного меню «Анализ») с построением графиков всех характеристик цифрового фильтра, их распечаткой и формированием стандартного протокола решения задачи синтеза.

The screenshot shows a software interface with six input fields for coefficients: Козф. A0, Козф. A1, Козф. A2, Козф. B0, Козф. B1, and Козф. B2. Each field has a yellow top section for the coefficient value and a blue bottom section for two alternative values (1) and 2). To the right, there is a field for 'Фд, кГц' set to 10.0. Below these fields are four buttons: 'Анализ' (Analysis), 'Протокол' (Protocol), 'Panel', and 'Восстан' (Restore). At the bottom right are 'View' and 'Print' buttons.

Козф. A0	Козф. A1	Козф. A2	Козф. B0	Козф. B1	Козф. B2
8192	6376	4481	-2119	63	2293
1) 8192 2) 8192	1) 6376 2) -5654	1) 4481 2) 4413	1) -2119 2) -5754	1) 63 2) 1101	1) 2293 2) 4935

В приложении 1 приведён пример протокола синтеза полосового БИХ-фильтра четвёртого порядка с указанием его основных функциональных показателей и оптимальных параметров – целочисленных коэффициентов полосового фильтра. В протоколе также приводятся эти коэффициенты в формате, необходимом для программирования ПЛИС.

### Программируемая логическая интегральная схема

В данной лабораторной работе изучаются вопросы программной реализации синтезированного ЦЦФ на программируемой логической интегральной схеме, широко используемой в современной радиоэлектронике в качестве встроенного средства контроля, цифровой обработки или управления характеристиками различных объектов или процессов.

ПЛИС первого поколения имели достаточно простую структуру и содержали программируемую матрицу соединений с триггерами на выходах. Они использовались в основном для разработки несложных устройств управления. В настоящее время разработаны и применяются более слож-

ные микросхемы, в том числе программируемые пользователем вентильные матрицы (ППВМ), которые по структуре приближаются к базовым матричным кристаллам (БМК). Однако в отличие от последних каждая ППВМ может программироваться многократно, что значительно снижает затраты на разработку новых устройств. Также отличительной особенностью является необходимость загружать структуру ППВМ каждый раз после подачи питания, т.к. структура в ППВМ не сохраняется после снятия питания. Обычно память структуры ППВМ хранится в энергонезависимой флэш-памяти.

Проектирование цифровых устройств в базисе ПЛИС имеет свои особенности. Для разработки конкретных схем используются специально созданные системы автоматизированного проектирования, в которых для ввода могут использоваться языки описания аппаратуры интегральных схем или универсальные схемные редакторы. Для программирования микросхем применяются программаторы, использующие стандарт IEEE 1149.4 JTAG. Этот стандарт позволяет не только производить загрузку ПЛИС, но и проверять правильность работы микросхемы.

Использование ПЛИС обеспечивает максимальную гибкость при необходимости модификации аппаратуры. Применение ПЛИС позволяет сократить процесс проектирования и отладки цифровых устройств. При этом время, требуемое для получения работающей микросхемы, составляет от нескольких часов до нескольких дней, весь процесс разработки и получения готовой микросхемы производится на одном рабочем месте.

Примером современных ПЛИС является XC3S700AN линейки Spartan-3AN фирмы Xilinx [8, 9, 12, 15], на базе которого предполагается реализация синтезированных в данной лабораторной работе цифровых фильтров.

Характерной особенностью ПЛИС является использование для вычислений только целых чисел, включая представление чисел с фиксированной точкой, т.к. они все непосредственно реализуются на логике. Никакие вычисления в формате чисел с плавающей точкой напрямую невозможны. Это обстоятельство определяет возможность построения на данной ПЛИС только цифровых фильтров, спроектированных принципиально под целочисленную арифметику аппаратного комплекса [3].

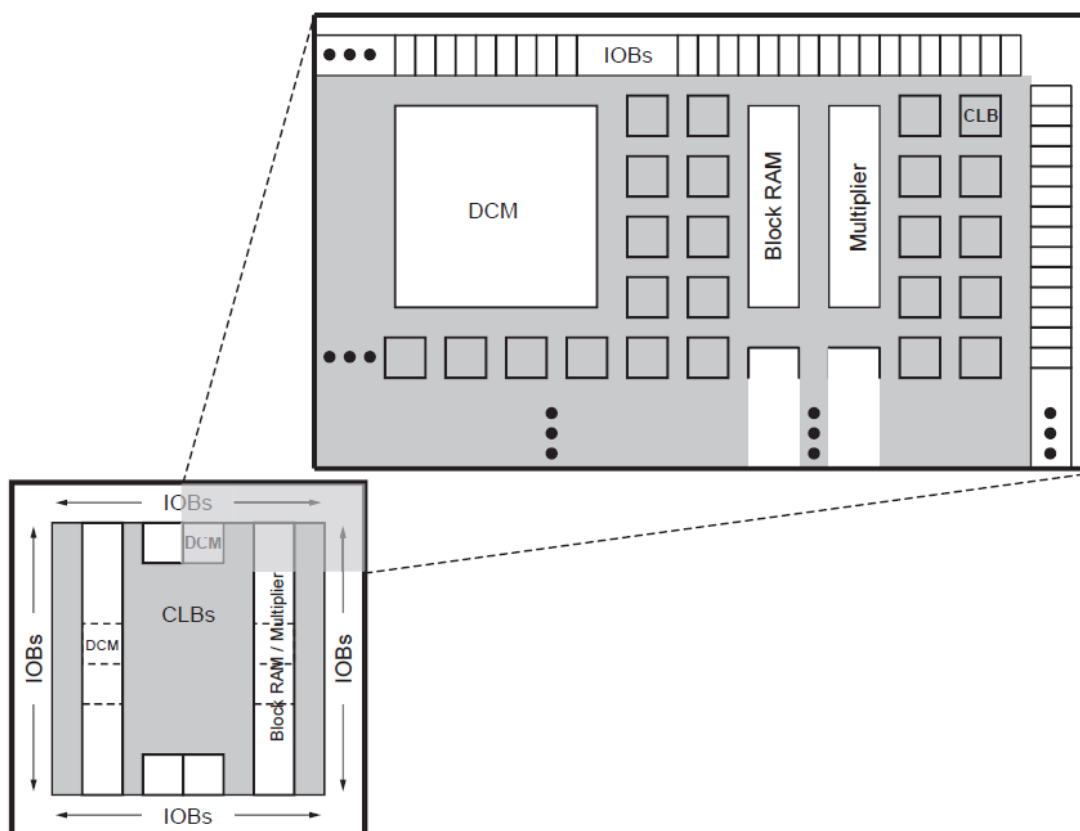


Рис. 8. Структура ПЛИС Spartan-3AN

На рис. 8 приведена общая структура ПЛИС XC3S700AN. В его состав входят:

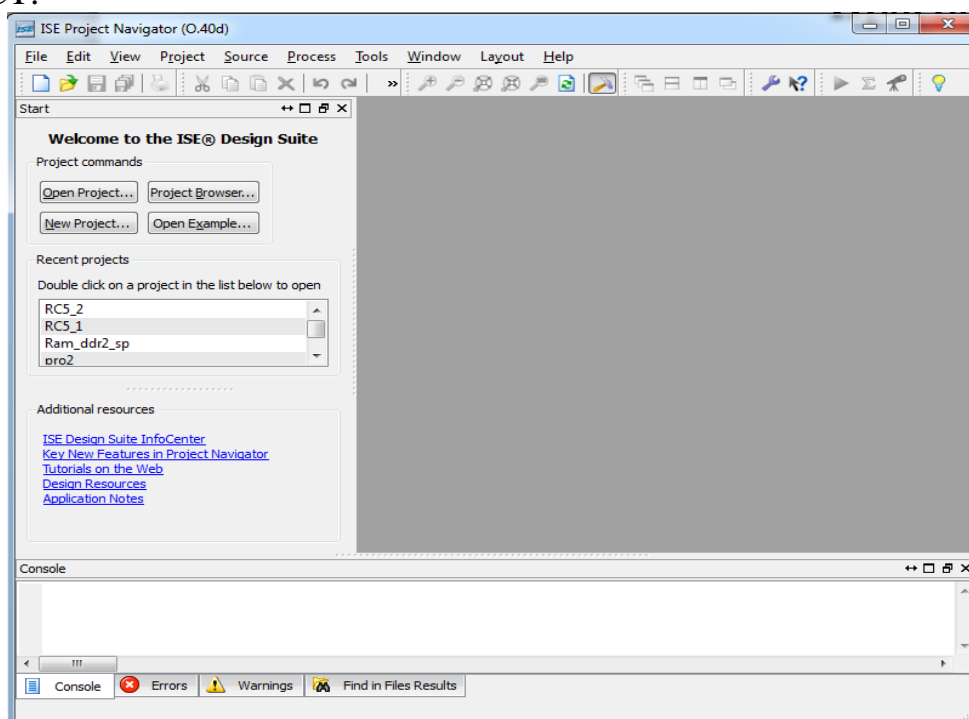
- Настраиваемые логические Блоки (CLBs) содержат гибкие Look-Up table (LUTs), которые реализуют логику плюс хранение элементов, используемых в качестве триггеров или защелки.
- Блоки ввода/вывода (IOBs) управляют потоком данных между контактами микросхемы и внутренней логикой устройства. Блоки ввода/вывода могут обеспечивать двунаправленный поток данных и высокоимпедансное состояние вывода. Они поддерживают различные стандарты сигнала, в том числе несколько высокопроизводительных дифференциальных стандартов. В состав блоков входят регистры двойной скоростью передачи данных (DDR).
- Блоки оперативной памяти. Каждый блок обеспечивает хранение данных в виде 18-Кбитных блоков с двойными портами.
- Блоки аппаратного умножителя. Каждый аппаратный умножитель производит умножение двух 18-разрядных чисел.
- Блоки цифрового управления тактовым сигналом (DCM). Блоки обеспечивают самокалибрующиеся, полностью цифровые решения для распространения, задержки, умножения, деления и фазового сдвига тактовых сигналов.

Подробную информацию об устройстве ПЛИС XC3S700AN можно найти в [9, 14, 15].

### Среда проектирования Xilinx ISE

Реализация синтезированного фильтра сводится к программированию ПЛИС, т.е. занесению в ПЗУ найденных целочисленных коэффициентов фильтра и алгоритма их обработки – расчёта выходного отклика фильтра по его линейно-разностному уравнению (8) для рекурсивного фильтра либо по прямой свёртке (15) – для нерекурсивного.

Программирование ПЛИС осуществляется в среде Xilinx ISE 14 на языке VHDL. В состав среды программирования входят менеджер проектов, редактор кода с подсветкой синтаксиса, схемотехнический редактор, синтезатор HDL описания XST, компоновщик, трассировщик, симулятор ISIM, программатор iMPACT, инструментальные средства отладки. Синтезатор XST осуществляет создание из VHDL описания принципиальной схемы на простых элементах. Компоновщик и трассировщик осуществляют размещение элементов в базе ПЛИС и трассировку сигналов между элементами. Загрузка структуры ППВМ осуществляется с помощью iMPACT.

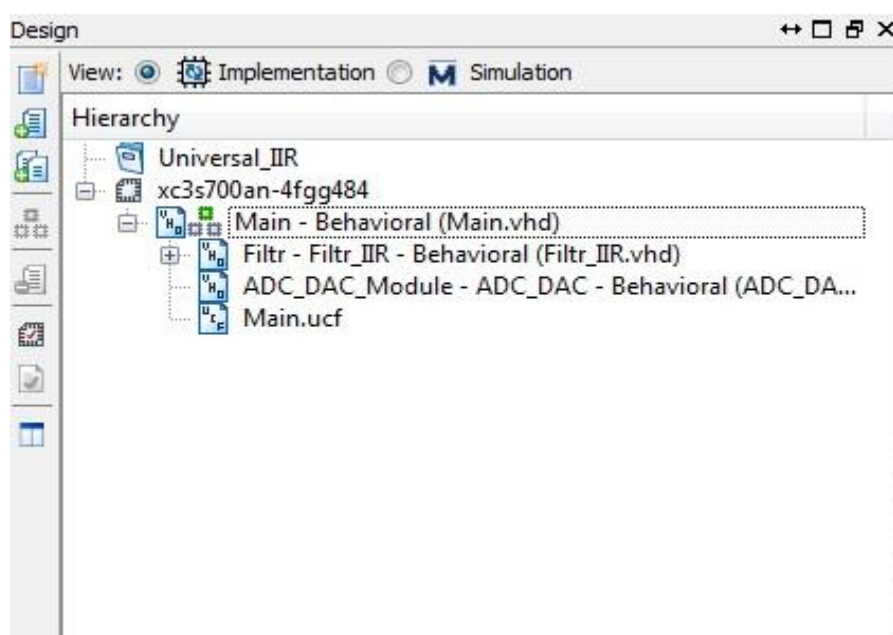


Для реализации цифрового фильтра на ПЛИС можно рекомендовать следующую последовательность работы в среде проектирования Xilinx ISE:

1. Загрузить проект фильтра *Universal\_IIR.xise*. Проект можно открыть следующим образом File -> Open project.

Проект состоит из четырех модулей:

- 1) *Main* (файл *Main.vhd*) – верхний файл в иерархии. Содержит данные о соединении модулей друг с другом и с внешними сигналами.
  - 2) *Filtr* (файл *Filtr\_IIR.vhd*) - отвечает за цифровую фильтрацию сигнала.
  - 3) *ADC\_DAC\_Module* (файл *ADC\_DAC.vhd*) – модуль управления АЦП и ЦАП. В этом же модуле задается частота дискретизации.
  - 4) *Main.ucf* – содержит данные о привязки внешних сигналов проекта к контактным площадкам микросхемы ПЛИС.
2. Задать коэффициенты цифрового фильтра. Открытие модуля *Filtr* (файл *Filtr\_IIR.vhd*) в текстовом редакторе осуществляется двойным кликом левой кнопкой мышки по нему в окне Hierarchy вкладки Design.



Далее в модуле *Filtr* найти и заменить текущие коэффициенты на коэффициенты фильтра из протокола программы синтеза фильтра. При вводе коэффициентов фильтра (строки 75 и далее) необходимо инвертировать знаки у коэффициентов  $a_1$  и  $a_2$  всех звеньев и следить за наличием или отсутствием запятых в строках данных. Для нормирующих коэффициентов звеньев  $a_0$  при этом вводится не их численное значение, а количество бит операции сдвига ( $B = \log_2 a_0$  бит), с помощью которой реализуется целочисленное деление на биномиальный нормирующий коэффициент  $a_0$ .

```

74
75 constant a0:coeff_type:=( 13, -- Zveno 1
76                             13 -- Zveno 2
77                             15, -- Zveno 3
78                             15, -- Zveno 4
79                             15, -- Zveno 5
80                             15, -- Zveno 6
81                             15, -- Zveno 7
82                             15, -- Zveno 8
83                             15, -- Zveno 9
84                             15 -- Zveno 10
85                             );
86 constant a1:coeff_type:=( 7143, -- Zveno 1
87                             2604 -- Zveno 2
88                             0, -- Zveno 3
89                             0, -- Zveno 4
90                             0, -- Zveno 5
91                             0, -- Zveno 6
92                             0, -- Zveno 7
93                             0, -- Zveno 8
94                             0, -- Zveno 9
95                             0 -- Zveno 10
96                             );
97 constant a2:coeff_type:=( -2086, -- Zveno 1
98                             -5172 -- Zveno 2
99                             0, -- Zveno 3
100                            0, -- Zveno 4
101                            0, -- Zveno 5
102                            0, -- Zveno 6
103                            0, -- Zveno 7
104                            0, -- Zveno 8
105                            0, -- Zveno 9
106                            0 -- Zveno 10
107                            );
108 constant b0:coeff_type:=( 2089, -- Zveno 1
109                             -1071 -- Zveno 2

```

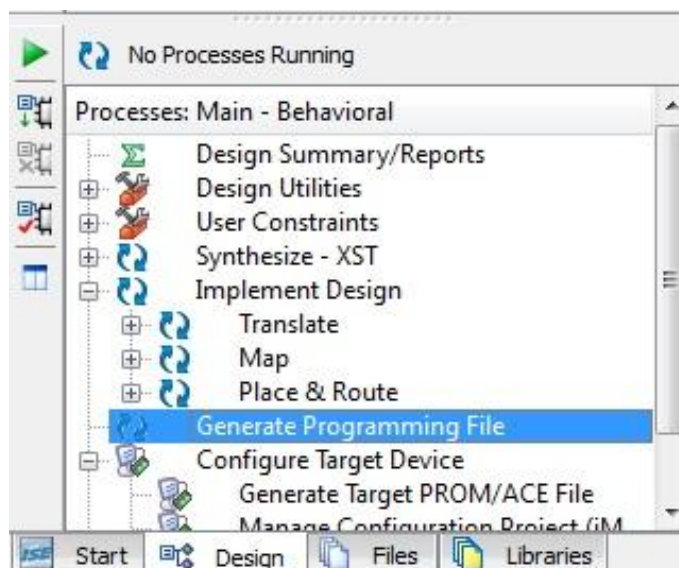
После ввода коэффициентов необходимо указать их разрядность (строка 45) и число звеньев синтезированного цифрового фильтра (строка 50):

```

41
42 architecture Behavioral of Filtr_IIR is
43
44 -- Разрядность коэффициентов цифрового фильтра
45 constant Coeff_Raz:integer:=16;
46
47 --constant Data_Raz:integer:=16;
48
49 -- Число звеньев цифрового фильтра
50 constant Zveno_count:integer:=2;
51

```

3. Выполнить синтез структуры ПЛИС, её компоновку и размещение в ПЛИС, а также подготовить файл для программирования. Для этого в окне Hierarchy с помощью одинарного клика левой кнопкой мышки выделить верхний модуль *Main* и в окне Processes дважды кликнуть левой кнопкой мышки по Generate Programming File:

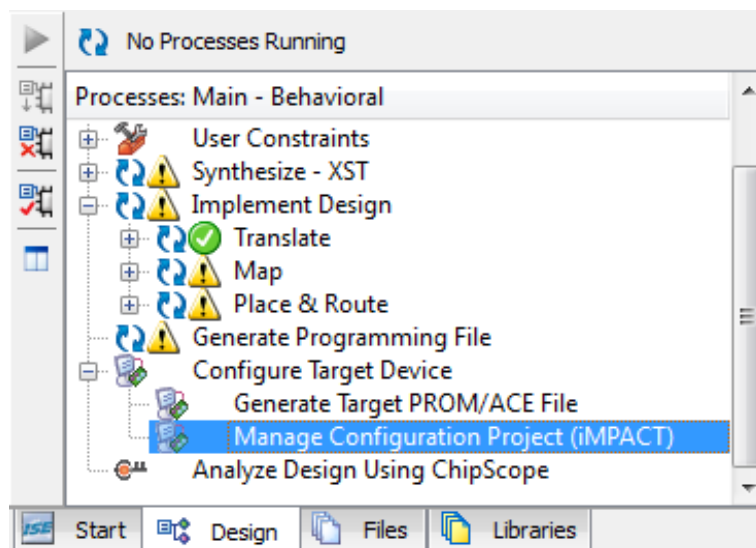


По завершению всех процессов будут подготовлены файлы для программирования ПЛИС.

### ***Программирование ПЛИС***

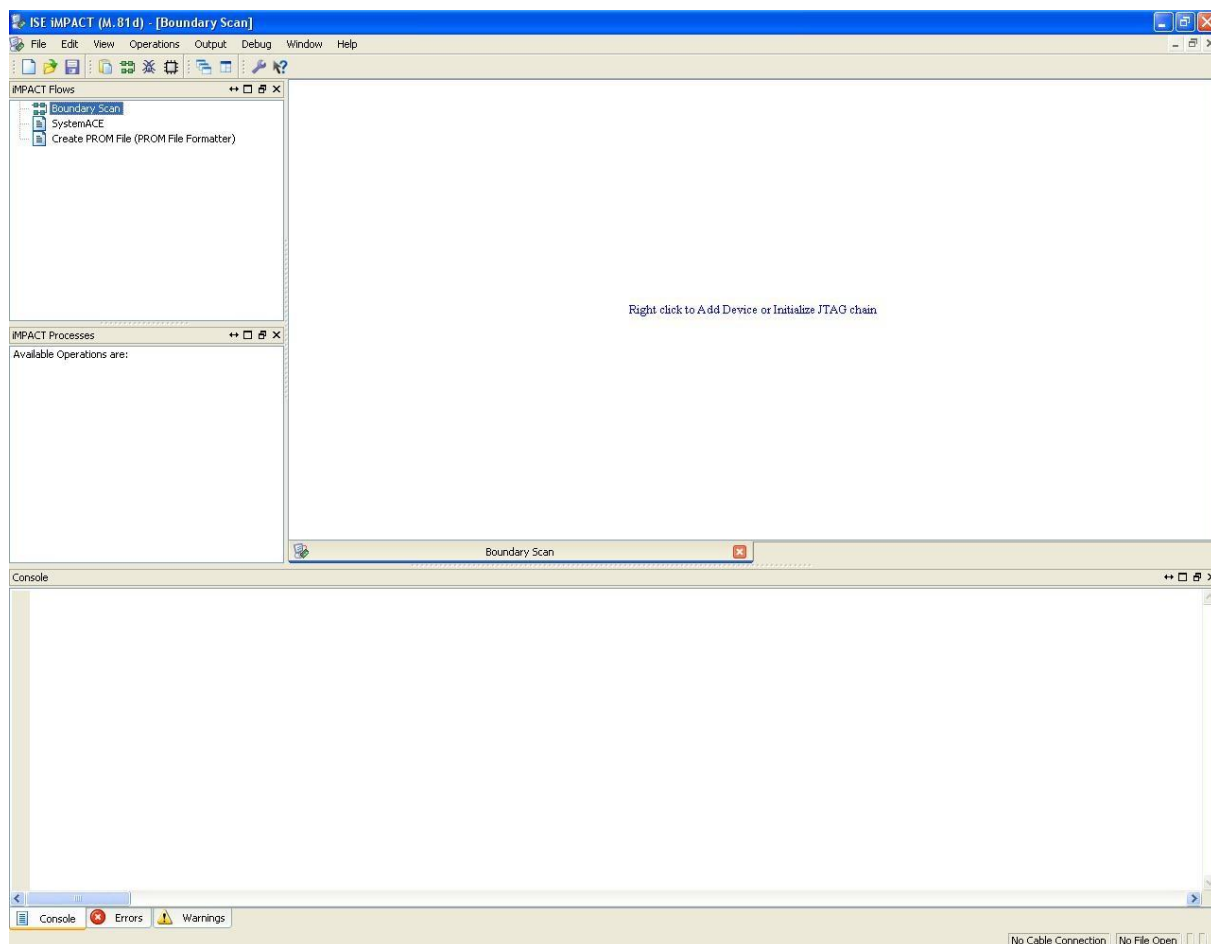
Для программирования ПЛИС можно рекомендовать следующую последовательность работы:

1. Запустить программатор iMPACT, два раза кликнув левой кнопкой мышки по Manage Configuration Project (iMPACT):



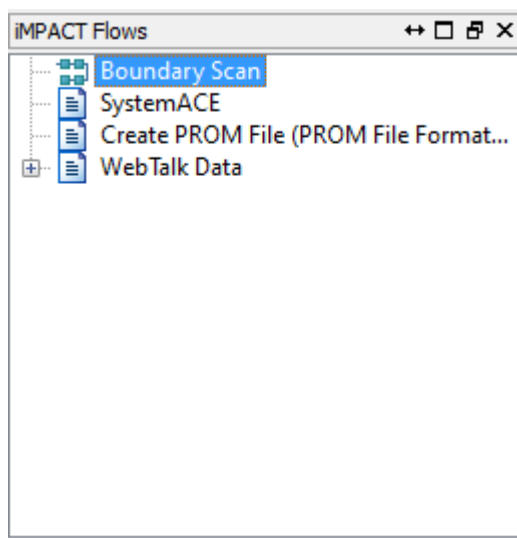
В отдельном окне запустится программатор iMPACT.





Будет задан вопрос об автоматическом создании конфигурационного проекта, на что следует ответить отказом “No”.

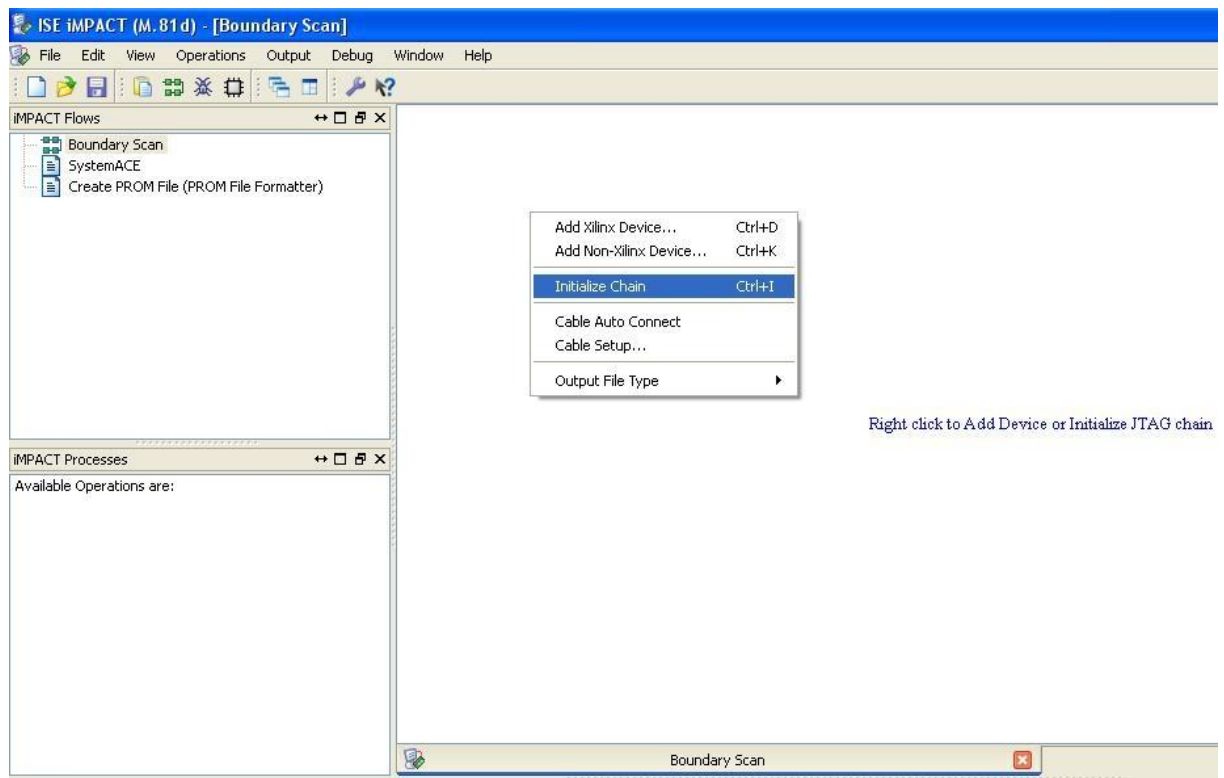
2. Открыть окно работы программатора с JTAG интерфейсом двойным кликом мышки по Boundary Scan в окне iMPACT Flows:



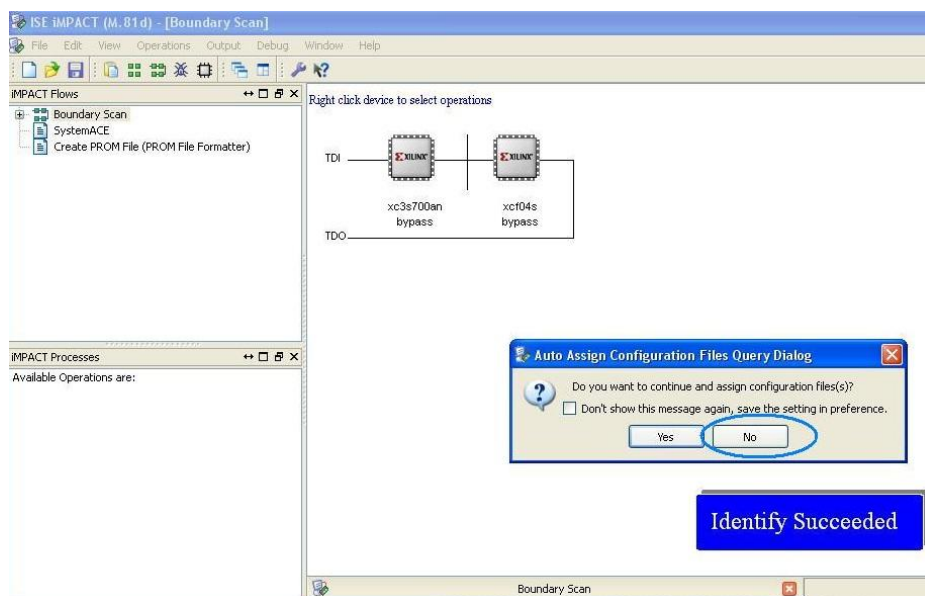
3. Установить связь с ПЛИС. В первую очередь необходимо удостовериться, что отладочная плата со Spartan-3AN включена и подключена к компьютеру с помощью USB кабеля. Правым кликом мышки по свобод-



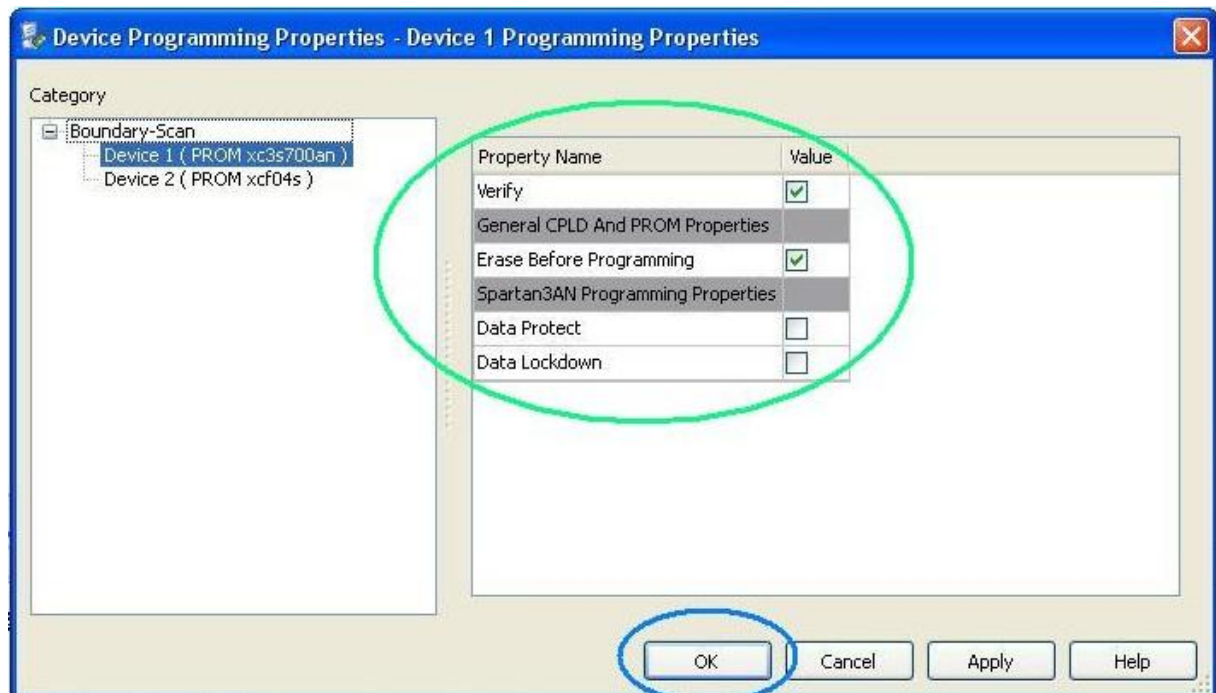
ному полю вкладки Boundary Scan вызвать меню и там выбрать Initialize Chain:



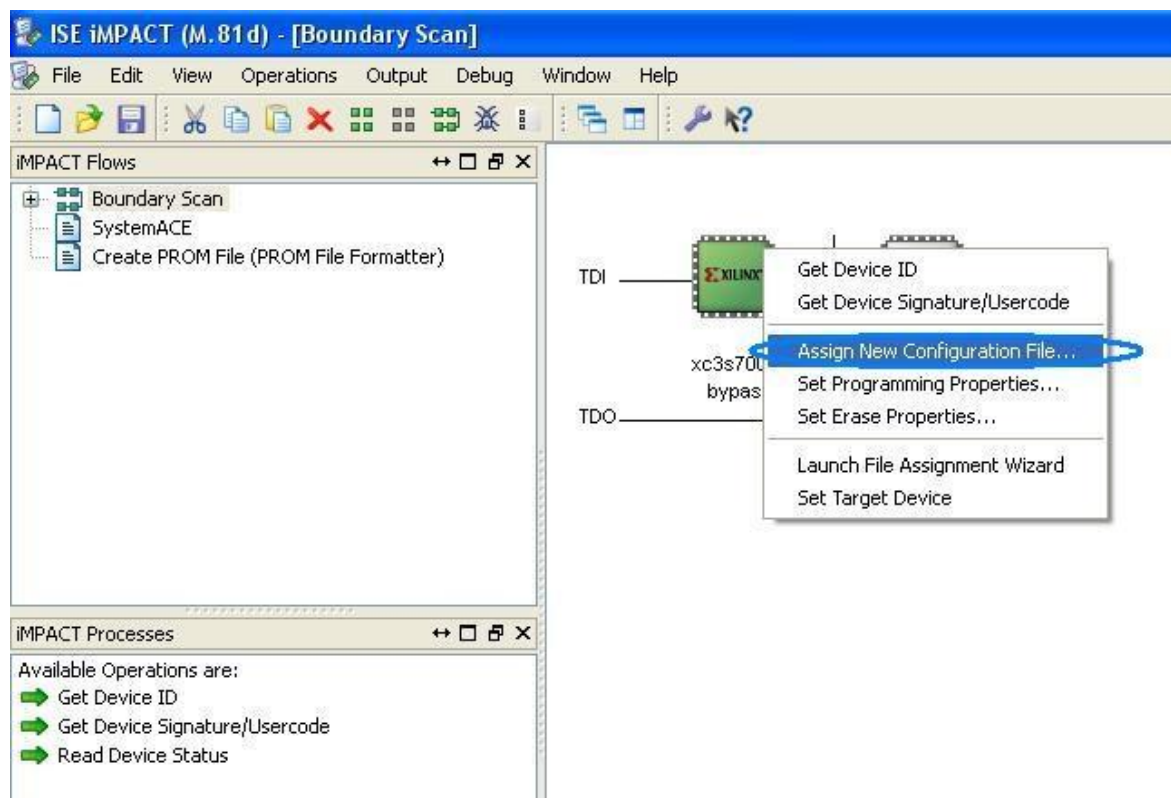
Программатором iMPACT будет произведено сканирование интерфейса JTAG цепочки. В случае успеха на поле появятся два изображения микросхем с подписями под ними. Одна микросхема с подписью XC3S700AN является ПЛИС, другая xc3s700an – внешняя конфигурационная память. На запрос iMPACT о дальнейшем предложении автоматического ассоциирования конфигурационных файлов с микросхемами необходимо ответить отрицательно (“NO”):



4. Убедиться, что в окне свойств настроек программирования установлены опции Verify и Erase before programming:



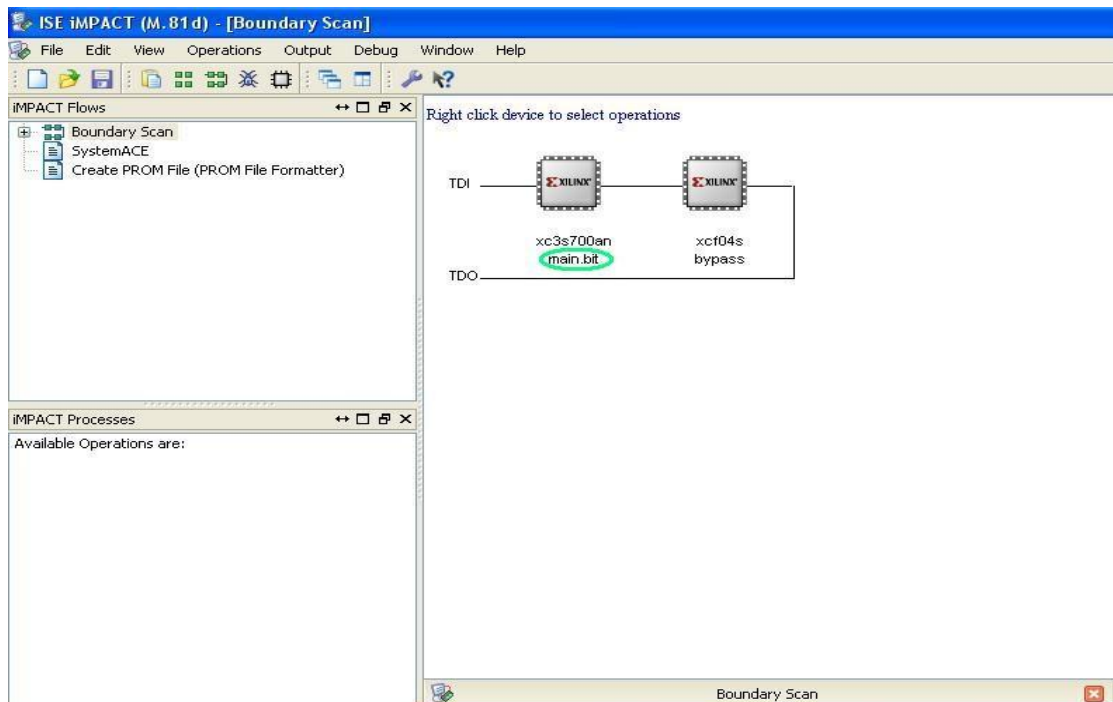
5. Привязать сгенерированную структуру с цифровым фильтром к ПЛИС. Для этого в меню, открываемся кликом правой клавиши мышки по микросхеме ПЛИС XC3S700AN, нужно выбрать Assign new configuration file:



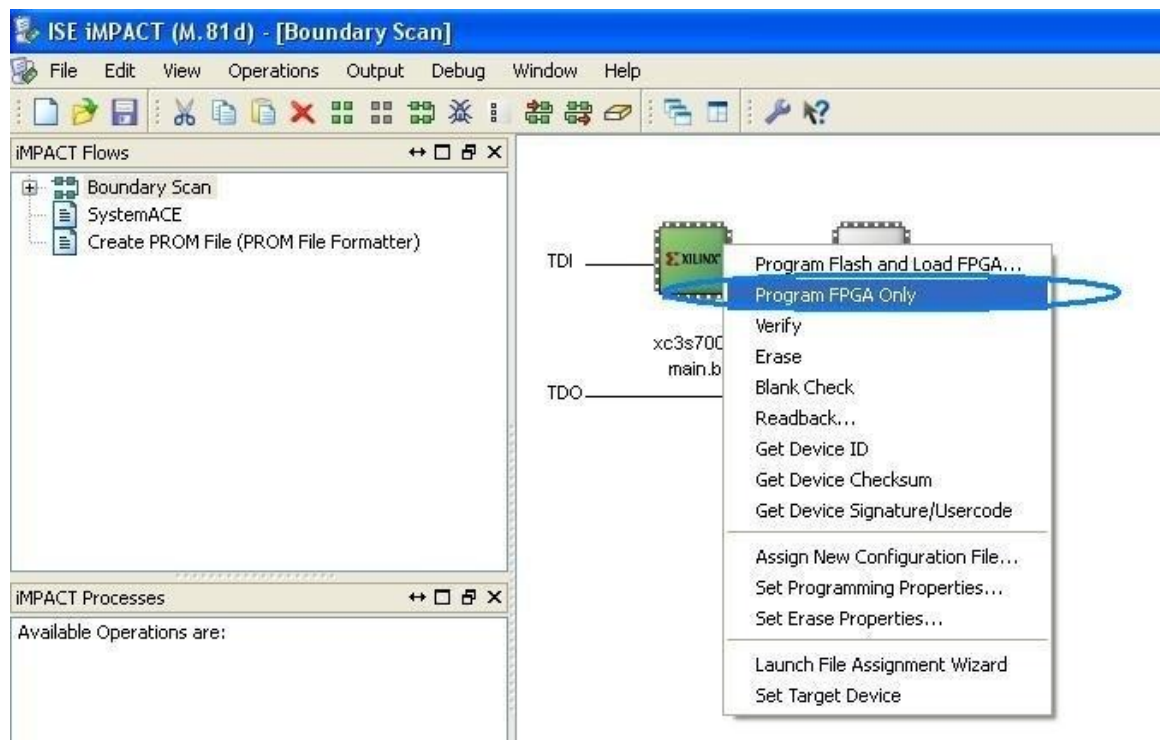
В открывшемся окне выбрать подготовленную к программированию ПЛИС структуру. Она содержится в файле main.bit, который находится в папке проекта:



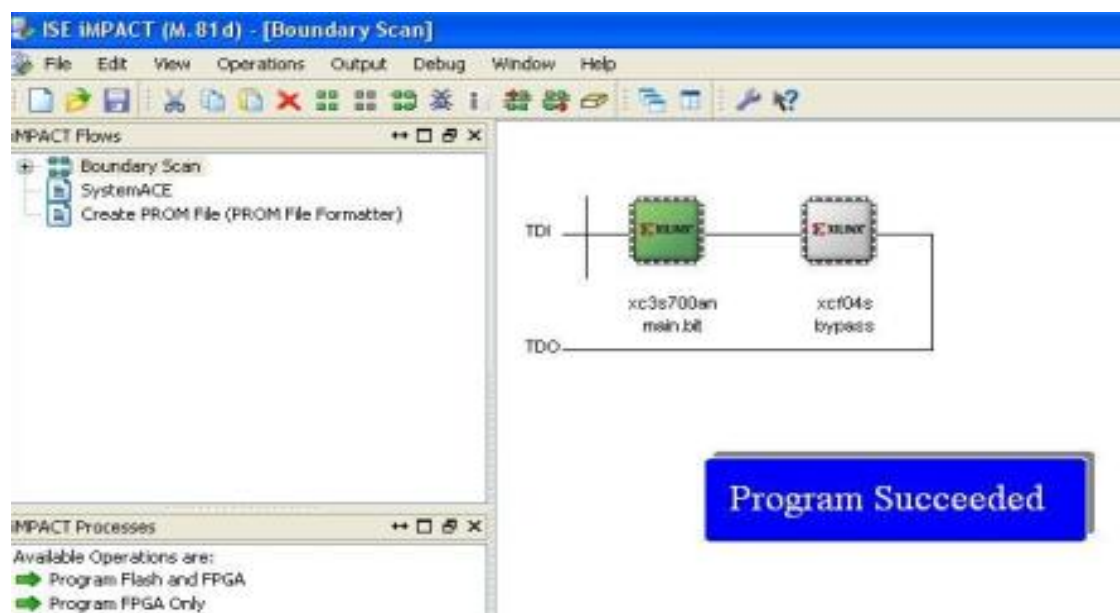
После успешного ассоциирования, подпись bypass под микросхемой ПЛИС XC3S700AN изменится на название файла - main.bit



6. Далее кликом правой клавиши мышки по микросхеме XC3S700AN в меню выбрать Program FPGA only для осуществления программирования ПЛИС:



После успешного завершения программирования синтезированная структура будет реализована в микросхеме ПЛИС:



После отключения питания отладочной платы, данные о структуре ПЛИС будут потеряны, поскольку структура загружалась с компьютера и не сохранялась в конфигурационной ПЗУ.

## Описание лабораторной установки

Для выполнения данной лабораторной работы используются три программных модуля:

- программа дискретного синтеза цифровых фильтров,
- среда Xilinx ISE программирования ПЛИС,
- цифровой измеритель частотных характеристик фильтра.

Перед началом работы эти три программы должны быть загружены в оперативную память компьютера и могут непосредственно использоваться для выполнения конкретных заданий лабораторной работы. Схема самой лабораторной установки представлена на рис. 9.

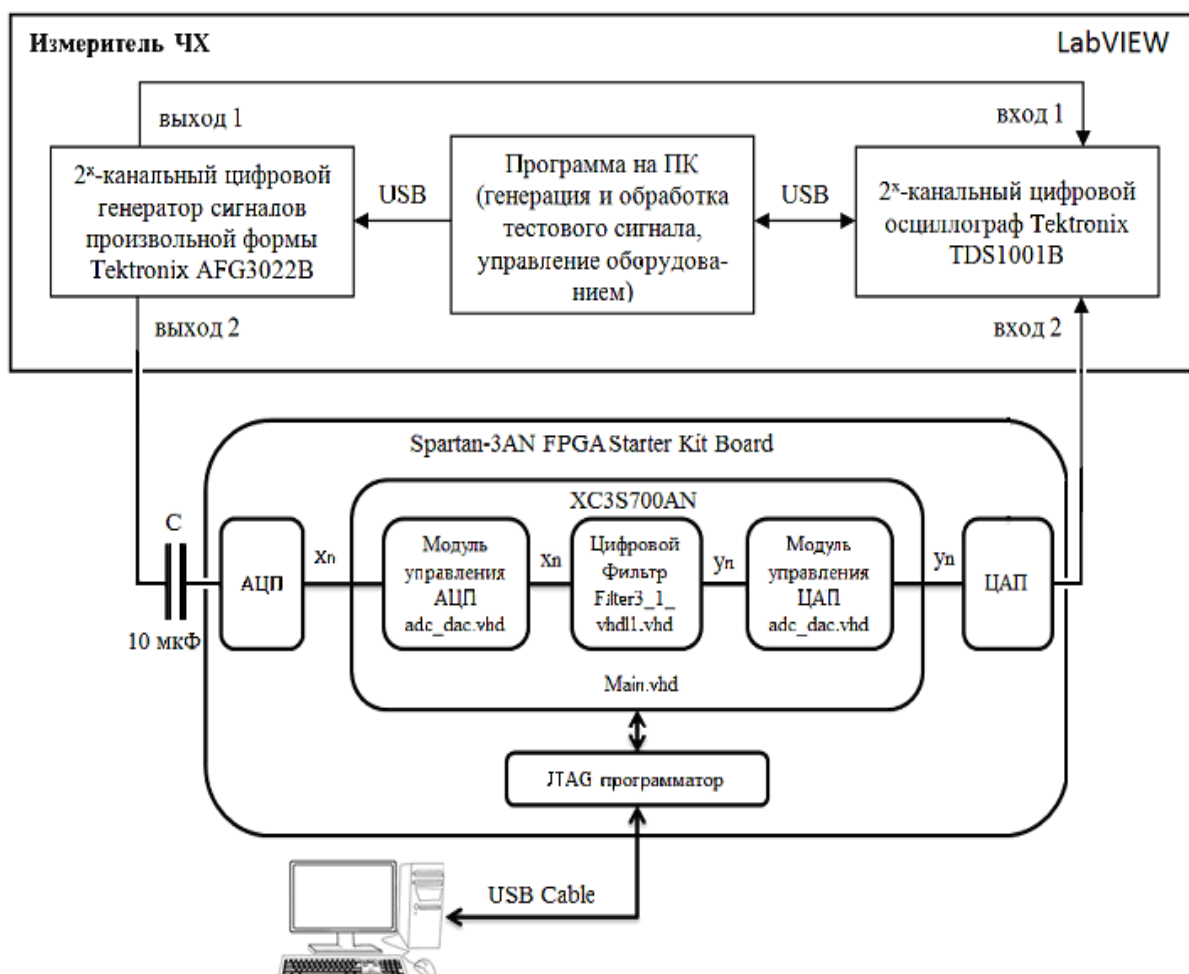


Рис. 9. Схема лабораторной установки

В лабораторную установку входят отладочная плата *Spartan-3AN FPGA Starter Kit Board* с ПЛИС XC3S700AN, на которой реализован исследуемый фильтр. Программирование ПЛИС осуществляется через интерфейс JTAG с помощью программатора, находящегося на отладочной плате ПЛИС. Измерение частотных характеристик (ЧХ) фильтра осуществляется на реальном сигнале с помощью автоматизированной измеритель-



ной системы, включающей в себя  $2^x$ -канальные цифровые генератор сигналов и осциллограф, а также разработанную в среде виртуальных приборов LabVIEW программу для обработки сигналов, вычисления частотных характеристик и управления генератором и осциллографом.

По команде запуска от программы управления (рис. 10), гармонический сигнал с частотой **fmin** подаётся с 1-го выхода генератора на 1-й вход цифрового осциллографа, а со 2-го выхода – на вход АЦП отладочной платы. Прошедший через реализованный на ПЛИС цифровой фильтр сигнал переводится в аналоговую форму с помощью ЦАП и поступает на 2-й вход осциллографа, при этом сигнал на входе 1 осциллографа дублирует сигнал на входе АЦП. Осциллограф оцифровывает сигналы одновременно на обоих входах и передаёт их в управляющую программу, которая вычисляет модуль и фазу коэффициента передачи исследуемого цифрового фильтра от входа АЦП до выхода ЦАП. Далее процесс измерения повторяется на всех дискретных частотах заданного пользователем интервала измерения от частоты **fmin** до частоты **fmax** с шагом **fs** (все частоты задаются в Гц). После завершения измерений в заданном интервале частот производится построение графиков амплитудно-частотной (АЧХ) и фазо-частотной (ФЧХ) характеристик цифрового фильтра. Осциллограммы сигналов на входе и выходе тестируемого устройства также отображаются на графиках. Любой график может быть скопирован в буфер обмена в графическом виде или в виде последовательностей числовых значений выборок средствами LabVIEW.

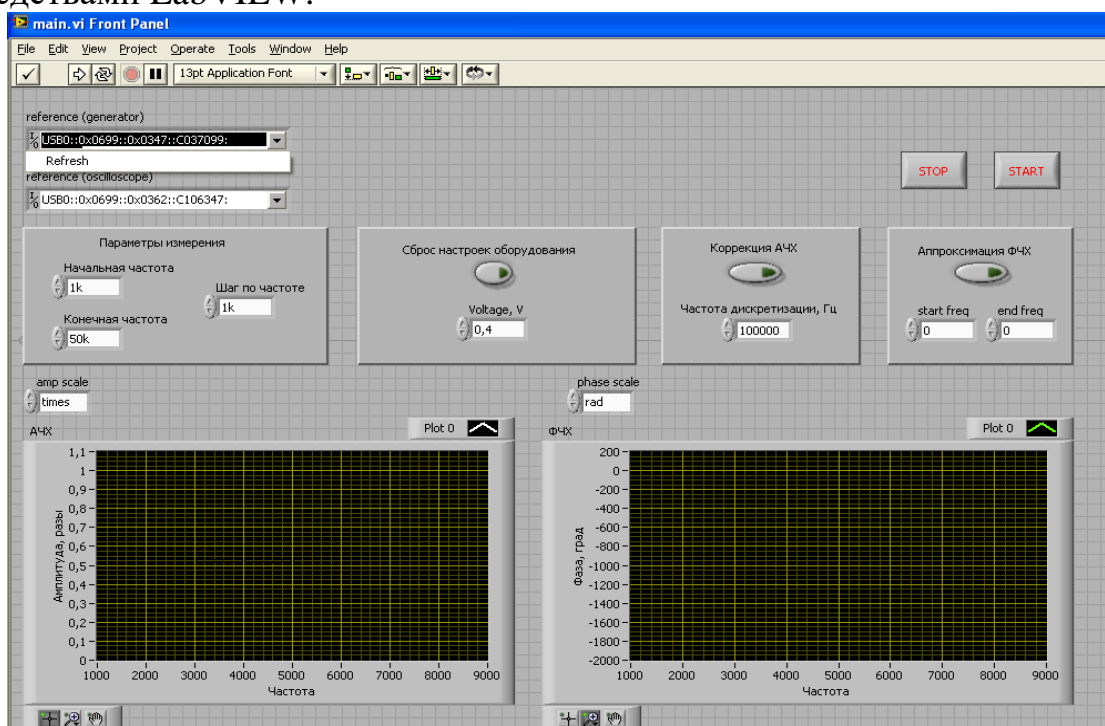


Рис. 10. Основная панель управляющей программы

Вид интерфейса программного модуля измерителя частотных характеристик показан на рис. 10. При использовании программы необходимо выполнить следующую последовательность действий:

- включить цифровые осциллограф и генератор сигналов и дождаться перехода их в состояние готовности;
- открыть программу панорамного измерителя характеристик;
- на вкладке «Reference» в поле «Generator» выполнить команду «Refresh» и выбрать строку, содержащую серийный номер генератора (C037099 - данный номер указан на задней панели генератора);
- аналогичным образом выбрать строку с серийным номером осциллографа :C106347 в поле «Oscilloscope»;
- запустить программу, нажав на кнопку с изображением стрелки, которая находится сверху под строкой меню;
- в соответствующих полях задать параметры измерения: частотный диапазон **fmin**, **fmax** и шаг измерения **fs** при  $U_{вх}=0,4$  в. На панели коррекции АЧХ задать частоту дискретизации 100 кГц;
- выставить произведённые настройки нажатием кнопки на панели настроек оборудования;
- нажать кнопку START для запуска процесса автоматического измерения модуля и фазы коэффициента передачи исследуемого ЦЦФ.

## Задание и порядок выполнения работы

### I. Синтез и оценка селективных свойств рекурсивных и нерекурсивных цифровых фильтров различного порядка.

Данное задание выполняется с помощью программы параметрического синтеза цифровых целочисленных фильтров. Для выполнения задания необходимо синтезировать ЦЦФ различной частотной селекции, а затем оценить  $\sigma$  - среднеквадратичную ошибку (СКО)

$$\sigma = \sqrt{\frac{1}{p} \cdot \sum_{n=1}^p [Y_n(\mathbf{IX}) - Y_n^T]^2} \quad (21)$$

выполнения заданных требований (требуемой АЧХ фильтра). Значение СКО выводится на панель синтеза компьютерной программы. Частота дискретизации  $F_d = 10$  кГц.

#### Синтез полосового фильтра

1). Осуществить синтез КИХ-фильтра второго порядка. Порядок выполнения задания в пакете синтеза следующий:

- Загрузить файл **FIR\_2p.top** исходных данных к синтезу;
- В функциональном редакторе ввести требуемую АЧХ фильтра (рис. 11). Правила работы в редакторе приведены выше;
- Синтезировать фильтр. Зафиксировать СКО выполнения заданных требований;

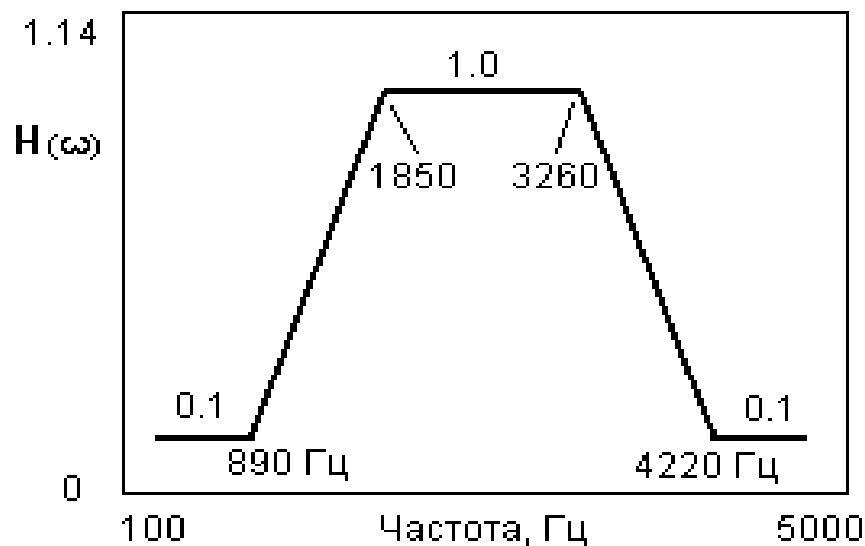


Рис. 11. Требуемая АЧХ полосового фильтра

2). В той же последовательности синтезировать полосовой КИХ-фильтр четвертого порядка (файл данных **FIR\_4p.top**).



3). Синтезировать полосовой БИХ-фильтр второго порядка (файл исходных данных **PIR\_2p.top**) по той же АЧХ (рис. 10).

4). В той же последовательности синтезировать полосовой БИХ-фильтр четвёртого порядка (файл исходных данных **PIR\_4p.top**). После чего в модуле анализа программы подробно исследовать полученное оптимальное решение:

- Задавая различные значения коэффициентов ЦЦФ, оценить изменение его АЧХ, получить неустойчивое решение задачи.
- Для оптимальных коэффициентов сохранить график АЧХ синтезированного БИХ-фильтра четвёртого порядка для отчёта как на всём главном интервале изменения цифровой частоты  $f=f_N$  ( $f_N=0.5F_d$  – частота Найквиста), так и на интервале частот вплоть до  $f=F_d$ ,  $f=2F_d$  и  $f=4F_d$ . Сформировать протокол решения данной задачи. Сохранить для отчёта найденные значения оптимальных коэффициентов полосового рекурсивного ЦЦФ четвёртого порядка.
- Синтезировать по той же АЧХ (рис. 11) полосовые БИХ-фильтры четвёртого порядка с разрядностью представления данных в 16, 8 и 4 бита (файлы исходных данных соответственно **PIR\_4p\_R16.top**, **PIR\_4p\_R8.top** и **PIR\_4p\_R4.top**). Зафиксировать СКО выполнения заданных требований в каждом случае. Сохранить графики АЧХ для отчёта.

Сравнивая СКО реализации заданной АЧХ, оценить селективные возможности рекурсивных и нерекурсивных полосовых ЦЦФ различного порядка и различной разрядности представления данных.

### *Синтез гауссова фильтра*

Нормированная резонансная характеристика для гауссовой кривой определяется следующим образом:

$$y(\xi) = e^{-\frac{\xi^2}{\alpha}}, \quad (22)$$

где  $\xi = f - f_0$  – абсолютная расстройка от резонансной частоты, а параметр  $\alpha$  определяет нормированную полосу пропускания гауссовой кривой:

$$\alpha = \frac{P^2}{4 \ln \sqrt{2}},$$

здесь  $P$  – абсолютная полоса пропускания по уровню 0,7. Выполнение задания в пакете синтеза осуществляется в следующей последовательности.

1) Осуществить синтез гауссова КИХ-фильтра второго порядка (файл исходных данных **FIR\_2p.top**). Зафиксировать СКО. Для ввода тре-

буемой гауссовой АЧХ (рис. 12) использовать панель шаблонов характеристик функционального редактора либо файл характеристики **Gauss.x**.

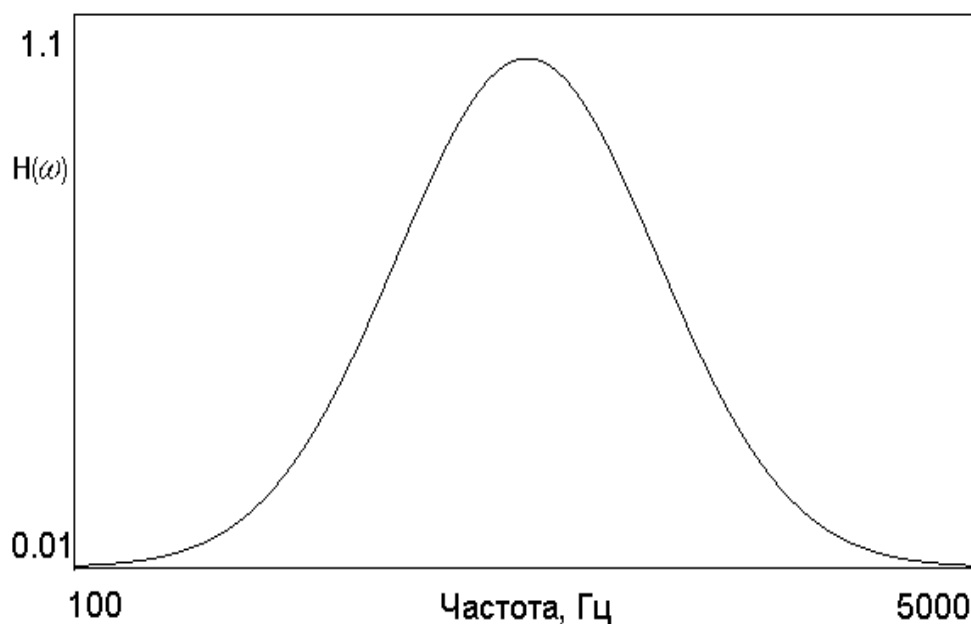


Рис. 12. Требуемая АЧХ гауссова фильтра

2). В той же последовательности синтезировать гауссов КИХ-фильтра четвертого порядка (файл данных **FIR\_4p.top**).

3). Синтезировать гауссов БИХ-фильтра второго порядка (файл исходных данных **IIR\_2p.top**) по той же АЧХ (рис. 12).

4). В той же последовательности синтезировать гауссов БИХ-фильтра четвертого порядка (файл исходных данных **IIR\_4p.top**). Сохранить для отчёта график АЧХ и найденные значения оптимальных коэффициентов рекурсивного гауссового ЦЦФ четвертого порядка.

Синтезировать по той же АЧХ (рис. 11) гауссовы БИХ-фильтры четвертого порядка с разрядностью представления данных в 16, 8 и 4 бита (файлы исходных данных соответственно **IIR\_4p\_R16.top**, **IIR\_4p\_R8.top** и **IIR\_4p\_R4.top**). Зафиксировать СКО выполнения заданных требований в каждом случае. Сохранить графики АЧХ для отчёта.

Сравнивая СКО реализации заданной АЧХ, оценить селективные возможности рекурсивных и нерекурсивных гауссовых ЦЦФ различного порядка и различной разрядностью представления данных.

5). Для БИХ-фильтра четвертого порядка (файл данных **IIR\_4p.top**) осуществить последовательно синтез фильтров различных видов селекции, используя библиотеку шаблонов характеристик функционального редактора пакета синтеза.

**II. Многофункциональный синтез рекурсивного цифрового фильтра нижних частот с линейной фазой.** В данном задании необходимо осуществить синтез рекурсивного ЦЦФ четвёртого порядка сначала по одной характеристике (АЧХ), а затем по двум его частотным характеристикам (АЧХ и ФЧХ) для частоты дискретизации  $F_d=10$  кГц. Порядок выполнения задания следующий.

1. Загрузить файл **IIR\_4p.top** исходных данных к синтезу ФНЧ;
2. В функциональном редакторе заказать два окна синтеза и ввести требуемую АЧХ фильтра (рис. 14) в первое окно, а требуемую ФЧХ (рис. 15) - во второе. Как видно, данная характеристика определяет требование линейности ФЧХ в полосе пропускания цифрового фильтра нижних частот;

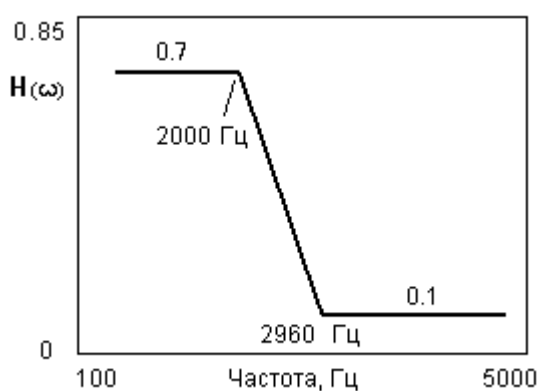


Рис. 14. Требуемая АЧХ

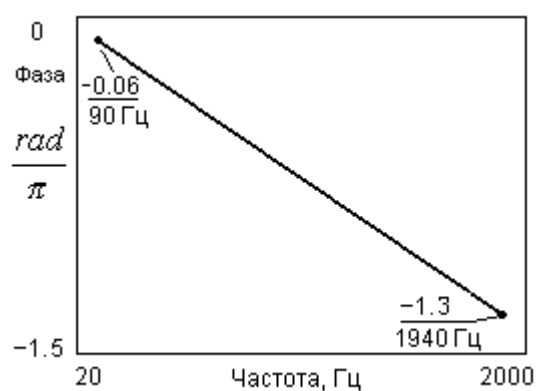


Рис. 15. Требуемая ФЧХ

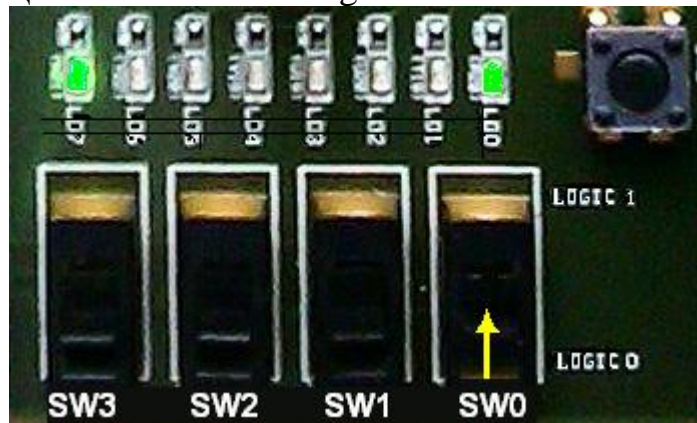
3. Задать вес  $\beta_1=0,5$  для окна АЧХ и вес  $\beta_2=0$  для окна ФЧХ и синтезировать фильтр. В модуле анализа исследовать полученное оптимальное решение. Оценить фазовые искажения в полосе пропускания (от 10 до 2000 Гц) фильтра по синтезу. Сохранить график АЧХ и ФЧХ синтезированного фильтра для отчёта. Сформировать протокол решения задачи;
4. Перезагрузив исходные данные (файл **IIR\_4p.top**), задать вес  $\beta_1=0,5$  для АЧХ и вес  $\beta_2=1$  для ФЧХ и синтезировать цифровой ФНЧ по двум требуемым характеристикам. В модуле анализа исследовать полученное оптимальное решение многофункционального синтеза. Оценить фазовые искажения в полосе пропускания (от 100 до 2000 Гц) фильтра по синтезу. Сохранить графики АЧХ и ФЧХ синтезированного ЦЦФ, а также найденные значения коэффициентов для отчёта. Сформировать протокол решения задачи;
5. Из протокола синтеза ввести в VHDL-программу расчёта отклика (файл *Filtr\_IIR.vhd*) число звеньев синтезированного фильтра, оптимальные его коэффициенты при разрядности представления данных  $R=16$  (приложение 2) и частоты дискретизации  $F_d=100$  кГц (при-

- ложение 3). Осуществить трансляцию и загрузку программы в ПЛИС. Убедиться, что на отладочной плате переключатель SW0 (V8) «режим контроля АЦП» находится в положении Logic 0;
6. Инициализировав работу цифрового устройства, с помощью панорамного измерителя произвести измерение АЧХ и ФЧХ синтезированного фильтра на интервале цифровой частоты от  $f_{\min}=1$  кГц до  $f_{\max}=45$  кГц с шагом  $f_s=1$  кГц. Измерить ФЧХ фильтра только в полосе пропускания ( $f_{\min}=1$  кГц,  $f_{\max}=20$  кГц, шаг  $f_s=0,5$  кГц). По результатам измерений оценить амплитудные и фазовые искажения в полосе пропускания фильтра. Сохранить графики измерения АЧХ и ФЧХ синтезированного фильтра для задачи многофункционального синтеза;

### III. Исследование частотной характеристики цифрового тракта.

Частотная характеристика тракта измеряется на реальном сигнале с помощью панорамного измерителя ЧХ. Частота дискретизации  $F_d=100$  кГц. Порядок выполнения задания следующий:

1. Осуществить трансляцию и загрузку VHDL-программы в ПЛИС. Перевести на отладочной плате переключатель SW0 (V8) «режим контроля АЦП» в положение Logic 1:



2. С помощью панорамного измерителя ЧХ измерить АЧХ цифрового тракта в диапазоне от  $f_{\min}=100$  Гц до  $f_{\max}=1$  кГц с шагом  $f_s=50$  Гц. Сохранить график АЧХ для отчёта;
3. Измерить АЧХ и ФЧХ цифрового тракта в диапазоне от  $f_{\min}=1$  кГц до  $f_{\max}=45$  кГц с шагом  $f_s=1$  кГц. Сохранить графики для отчёта;
4. С помощью встроенного в панорамный измеритель осциллографа зафиксировать форму входного и выходного (с ЦАП) сигналов на частотах  $f_{\max}=1, 10, 20$  и  $45$  кГц ( $f_{\min}=1$  кГц, шаг  $f_s=5$  кГц).

На основании проведённых измерений определить:

- а) рекомендуемый рабочий частотный диапазон реализации цифровых фильтров на данном микроконтроллере;

- б) среднее затухание  $A_{ср}$ , вносимое цифровым трактом;
- в) амплитудные  $\Delta H(\omega)$  и фазовые  $\Delta \varphi(\omega)$  искажения цифрового тракта в рабочем диапазоне.

### **Содержание отчета**

1. Модель и структура звена рекурсивного и нерекурсивного цифрового целочисленного фильтра.
2. Постановка задачи синтеза рекурсивного ЦЦФ.
3. Постановка задачи синтеза нерекурсивного ЦЦФ.
4. Описание блок-схемы и компьютерной программы синтеза фильтра.
5. Результаты синтеза и оценка селективных возможностей рекурсивных и нерекурсивных ЦЦФ различного порядка и различной разрядности представления данных. Частотные характеристики синтезированных цифровых фильтров.
6. Измерение частотных характеристик цифрового тракта.
7. Результаты синтеза и измерения частотных характеристик целочисленного рециркулятора.
8. Результаты многофункционального синтеза рекурсивного фильтра нижних частот с линейной фазой. Измерения АЧХ и ФЧХ синтезированного ЦЦФ. Оценка амплитудных (СКО) и фазовых искажений в полосе пропускания фильтра.
9. Интерпретация результатов, общие выводы по работе.

### **Контрольные вопросы**

1. Назначение и определение цифрового фильтра.
2. Какие характеристики имеет цифровой фильтр в частотной области ?
3. Современные требования к устройствам цифровой фильтрации.
4. В чём особенности математического программирования как методологии проектирования радиоэлектронных устройств ?
5. Какие достоинства имеют цифровые фильтры, спроектированные методологией целочисленного нелинейного программирования ?
6. Модели рекурсивных целочисленных фильтров. Структура их построения и условие устойчивости.
7. Модели нерекурсивных целочисленных фильтров. Структура их построения.
8. Постановка задачи синтеза фильтра методом ЦНП.
9. В чём состоит методология поискового решения экстремальной задачи синтеза целочисленных цифровых фильтров?
10. Какие основные требования предъявляются к алгоритмам поисковой минимизации?.

11. Общая структура программы синтеза целочисленного фильтра.
12. ПЛИС. Её структура и характеристики.
13. Оценка быстродействия цифровых целочисленных фильтров, реализованных на микроконтроллере.
14. Как влияет разрядность представления данных на частотных характеристиках ЦЦФ и его быстродействие?
15. Измерение характеристик цифрового фильтра на реальном сигнале.

### **Литература основная**

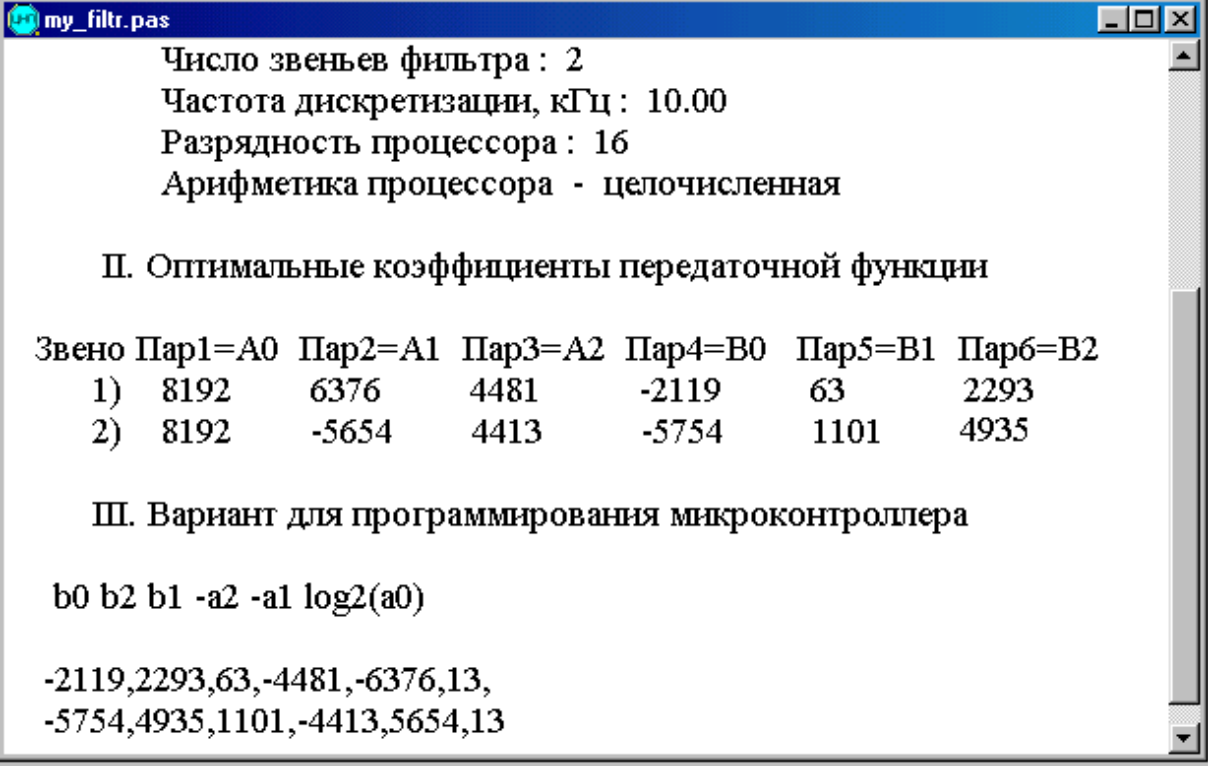
1. Мину М. Математическое программирование. Теория и алгоритмы. М., Наука, 1990, 488 с.
2. Воинов Б.С., Бугров В.Н., Воинов Б.Б. Информационные технологии и системы: поиск оптимальных, оригинальных и рациональных решений. М., Наука, 2007, 730 с.
3. Бугров В.Н. Проектирование цифровых фильтров методами целочисленного нелинейного программирования. // Вестник ННГУ, 2009, № 6. с. 61 – 70.
4. Бугров В.Н., Пройдаков В.И., Артемьев В.В. Поисковые технологии проектирования целочисленных цифровых фильтров. Часть 2. М., Компоненты и технологии, № 10, 2014, с. 142-150.
5. Шкелев Е.И., Бугров В.Н., Ивлев Д.Н. Цифровая обработка сигналов с применением цифровых сигнальных процессоров. Электронное учебно-методическое пособие. Регист.номер 452 12 04, //Н.Новгород, ННГУ, 2012, 83 С.
6. Рабинер Л., Гоулд Б. Теория и применение цифровой обработки сигналов. - М.: Мир, 1978.-848 с.
7. Антонию А. Цифровые фильтры: анализ и проектирование. М., Радио и Связь , 1983.
8. Мингазин А.Т. Синтез передаточных функций цифровых фильтров в области дискретных значений коэффициентов (обзор). // Электронная техника. Сер. 10. 1993. № 1,2. С. 3-35.
9. Электронная версия руководства пользователя (оригинал) ug334.pdf.
10. Шкелев Е.И. Электронные цифровые системы и микропроцессоры. Учебное пособие. //Н.Новгород: Изд.ННГУ, 2004 – 152 с.

### **Литература дополнительная**

11. Корбут А.А., Финкельштейн Ю.Ю. Дискретное программирование. - М.: Наука, 1959, 370 с.
12. Баскаков С.И. Радиотехнические цепи и сигналы. – М.: Высшая школа, 2005 – 270 с..

13. Шкелев Е.И., Бугров В.Н., Пройдаков В.И., Артемьев В.В. Целочисленные цифровые фильтры—эффективное решение для 8-битовых цифровых платформ. М., Компоненты и технологии, № 10, 2013, с.104-110.
14. Электронная версия руководства пользователя (оригинал) ug331.pdf.
15. Электронная версия технических условий (оригинал) ds557.pdf.

**Протокол синтеза полосового рекурсивного целочисленного  
фильтра четвёртого порядка**



```
my_filtr.pas
Число звеньев фильтра : 2
Частота дискретизации, кГц : 10.00
Разрядность процессора : 16
Арифметика процессора - целочисленная

II. Оптимальные коэффициенты передаточной функции

Звено Пар1=A0 Пар2=A1 Пар3=A2 Пар4=B0 Пар5=B1 Пар6=B2
1) 8192 6376 4481 -2119 63 2293
2) 8192 -5654 4413 -5754 1101 4935

III. Вариант для программирования микроконтроллера

b0 b2 b1 -a2 -a1 log2(a0)

-2119,2293,63,-4481,-6376,13,
-5754,4935,1101,-4413,5654,13
```



## Листинг VHDL кода модуля Filtr\_IIR.vhd

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
--use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Filtr_IIR is
    generic (Data_Raz:integer:=16);
    Port ( clk : in  STD_LOGIC;
           clr : in  STD_LOGIC;
           din : in  STD_LOGIC_VECTOR (Data_Raz-1 downto 0);
           ce : in  STD_LOGIC;
           ceo : out STD_LOGIC;
           dout : out STD_LOGIC_VECTOR (Data_Raz-1 downto 0));
    --      dout : out STD_LOGIC_VECTOR (31 downto 0));
end Filtr_IIR;

architecture Behavioral of Filtr_IIR is

    -- Разрядность коэффициентов цифрового фильтра
    constant Coeff_Raz:integer:=16;

    --constant Data_Raz:integer:=16;

    -- Число звеньев цифрового фильтра
    constant Zveno_count:integer:=2;

    component IIR_Zveno
        generic (Data_Raz:integer;
                 Coeff_Raz:integer;
                 a0:integer;
                 a1:integer;
                 a2:integer;
                 b0:integer;
                 b1:integer;
                 b2:integer);
        Port ( clk : in  STD_LOGIC;
              clr : in  STD_LOGIC;

```

```

        data_in : in std_logic_vector(Data_Raz-1 downto 0);
        data_out : out std_logic_vector(Data_Raz-1 downto 0);
                ceo : out STD_LOGIC;
        ce : in STD_LOGIC);
end component;

type Data_net_type is array (0 to Zveno_count-1) of std_logic_vector(Data_Raz-1 downto 0);
signal Data_nets:Data_net_type;
type sig_net_type is array (0 to Zveno_count-1) of std_logic;
signal Sig_nets:sig_net_type;
type coeff_type is array (0 to Zveno_count-1) of integer;

constant a0:coeff_type:=(    13, -- Zveno 1
                           13 -- Zveno 2
--                           15, -- Zveno 3
--                           15, -- Zveno 4
--                           15, -- Zveno 5
--                           15, -- Zveno 6
--                           15, -- Zveno 7
--                           15, -- Zveno 8
--                           15, -- Zveno 9
--                           15 -- Zveno 10
                                   );

constant a1:coeff_type:=(   7143, -- Zveno 1
                           2604 -- Zveno 2
--                           0, -- Zveno 3
--                           0, -- Zveno 4
--                           0, -- Zveno 5
--                           0, -- Zveno 6
--                           0, -- Zveno 7
--                           0, -- Zveno 8
--                           0, -- Zveno 9
--                           0 -- Zveno 10
                                   );

constant a2:coeff_type:=(  -2086, -- Zveno 1
                           -5172 -- Zveno 2
--                           0, -- Zveno 3
--                           0, -- Zveno 4
--                           0, -- Zveno 5
--                           0, -- Zveno 6
--                           0, -- Zveno 7
--                           0, -- Zveno 8
--                           0, -- Zveno 9
--                           0 -- Zveno 10
                                   );

constant b0:coeff_type:=(   2089, -- Zveno 1
                           -1071 -- Zveno 2

```



```

                                clr => clr,
                                data_in => din,
                                data_out => Data_nets(i),
                                ceo => Sig_nets(i),
                                ce => ce);

end generate Gen_Zveno1;
Gen_Zveno: if i>0 generate
    Zveno:IIR_Zveno
        generic map( Data_Raz => Data_Raz,
                    Coeff_Raz => Coeff_Raz,
                    a0 => a0(i),
                    a1 => a1(i),
                    a2 => a2(i),
                    b0 => b0(i),
                    b1 => b1(i),
                    b2 => b2(i))
        port map( clk => clk,
                    clr => clr,
                    data_in => Data_nets(i-1),
                    data_out => Data_nets(i),
                    ceo => Sig_nets(i),
                    ce => Sig_nets(i-1));

    end generate Gen_Zveno;
end generate Gen_Zveno_Chain;

process(clk)
begin
    if clk'event and clk='1' then
        if Sig_nets(Zveno_count-1)='1' then
            dout<=Data_nets(Zveno_count-1);
            ceo<='1';
        else
            ceo<='0';
        end if;
    end if;
end process;

end Behavioral;

```

### Листинг VHDL кода модуля ADC\_DAC.vhd

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ADC_DAC is
  Port ( clk : in  STD_LOGIC;
        reset : in  STD_LOGIC;
        AD_CONV : out STD_LOGIC;
        SPI_SCK : out STD_LOGIC;
        SPI_MOSI : out STD_LOGIC;
        AMP_CS : out STD_LOGIC;
        AMP_SHDN : out STD_LOGIC;
        AMP_DOUT : in  STD_LOGIC;
        TEST_ADC : in  STD_LOGIC;
        TEST : out STD_LOGIC;
        TEST_8 : out STD_LOGIC_VECTOR(7 downto 0);
        READY : out STD_LOGIC;
        DAC_CS : out STD_LOGIC;
        DAC_CLR : out STD_LOGIC;
        ADC: out STD_LOGIC_VECTOR(13 downto 0);-- from ADC
        CEO : out STD_LOGIC;
        DAC: in  STD_LOGIC_VECTOR(11 downto 0);-- to DAC
        CE : in  STD_LOGIC;
        DAC_OUT : in  STD_LOGIC;
        AD_DOUT : in  STD_LOGIC);
end ADC_DAC;

architecture Behavioral of ADC_DAC is
  attribute BOX_TYPE : string ;
  component BUFG
    port (I:in STD_LOGIC;
          O:out STD_LOGIC);
  end component;

```

```

attribute BOX_TYPE of BUFG : component is "BLACK_BOX";

constant Fdf:integer:=100; -- кГц (Частота дискретизации)

constant td:integer:=(1000000/20/Fdf)-1;

--AMP
signal SPI_MOSI_AMP:std_logic:= '0';
signal amp_ready:std_logic:= '0';
signal dclk:std_logic:= '0';
signal gdclk:std_logic:= '0';
signal amp_clk:std_logic:= '0';
signal ac:std_logic:= '0';
signal d:std_logic_vector(1 downto 0):=(others=>'0');
signal c:std_logic:= '0';
signal gc:std_logic:= '0';
signal ar:integer range 0 to 31:=0;
signal amp_check:std_logic:= '0';
signal AMP_REG:std_logic_vector(7 downto 0):=(others=>'0');
signal gamp_clk:std_logic:= '0';

--ADC
signal snc:integer range 0 to 63:=0;
signal adc_busy:std_logic:= '0';
signal conv:std_logic:= '0';
signal sn:std_logic:= '0';
signal R1:std_logic:= '0';
signal cn:std_logic:= '1';
signal ADC_REG1:std_logic_vector(13 downto 0):=(others=>'0');
signal ADC_REG1_TEMP:std_logic_vector(13 downto 0):=(others=>'0');
signal ADC_REG1_TMP1:std_logic_vector(15 downto 0):=(others=>'0');
signal NE:std_logic:= '0';

--DAC
--signal sc:integer range 0 to 63:=0;
--signal TEST_LACH:std_logic:= '0';
signal spc:integer range 0 to 31:=0;
signal sd:std_logic:= '0';
signal sde:std_logic:= '0';
signal sdc:std_logic:= '0';
signal bclk:std_logic:= '0';
signal SPI_MOSI_DAC:std_logic:= '0';
signal New_Data:std_logic:= '0';

constant AMP_GAIN:std_logic_vector(7 downto 0):="00010001";
--signal SPI_MOSI_DAC:std_logic:= '0';
--signal DAC_COM:std_logic_vector(3 downto 0):="0011";
--signal DAC_ADR:std_logic_vector(3 downto 0):="0000";

```

```

constant DAC_COM:std_logic_vector(3 downto 0):="0011";
constant DAC_ADR:std_logic_vector(3 downto 0):="0000";

--signal DAC_DAT:std_logic_vector(11 downto 0):="000000000001";
signal DAC_DAT:std_logic_vector(11 downto 0):=(others=>'0');
signal TC:integer range 0 to 1023:=0;
begin

    dclkbuf:BUFG
    port map (I=>dclk,
              O=>gdclk);

    gcbuf:BUFG
    port map (I=>c,
              O=>gc);

    gamp_clkbuf:BUFG
    port map (I=>amp_clk,
              O=>gamp_clk);

-- Common
    process(clk)
    begin
        if clk'event and clk='1' then
            if c=0 then
--                if c='0' then
--                    if d="00" then
                        dclk<=not dclk;
                    end if;
--                c<=c+1;
--                d<=d+1;
--                c<=not c;
            end if;
        end process;

-- Controll
    process(clk,reset)
    begin
        if reset='1' then
            conv<='0';
            New_Data<='0';
        elsif clk'event and clk='1' then
            if sde='1' then
                New_Data<='0';
            end if;
            conv<='0';
            tc<=tc+1;
--            if tc=199 then -- 250 KHz
--            if tc=249 then -- 200 KHz
            if tc=td then -- Частота дискретизации
                tc<=0;
                if cn='1' and amp_ready='1' then

```

```

                                conv<='1';
                                New_Data<='1';
                            end if;
                        end if;-- tc
                    end if;--clk
                end process;

-- Amplifier
    process (gdclk,reset)
    begin
        if reset='1' then
            AMP_CS<='1';
            amp_ready<='0';
            amp_check<='0';
            amp_clk<='0';
            ac<='0';
            ar<=0;

            SPI_MOSI_AMP<='0';
        elsif gdclk'event and gdclk='1' then
            if ac='1' then
                amp_clk<=not amp_clk;
            end if;
            if amp_ready='0' then
                SPI_MOSI<='0';
                ar<=ar+1;
                AMP_CS<='0';
                ac<='1';
            end if;
            if ar=6 then
                SPI_MOSI_AMP<='1';
            elsif ar=8 then
                SPI_MOSI_AMP<='0';
            elsif ar=14 then
                SPI_MOSI_AMP<='1';
            elsif ar=16 then
                SPI_MOSI_AMP<='0';
            end if;
            if ar=17 then
                if amp_check='0' then
                    amp_check<='1';
                else
                    if amp_reg=AMP_GAIN then
                        amp_ready<='1';
                    else
                        amp_ready<='0';
                    end if;
                end if;
            end if;
        end if;
    end process;

```



```

        ar<=0;
        amp_clk<='0';
        ac<='0';
        AMP_CS<='1';
    end if;
end if;
end process;
process(gamp_clk,reset)
begin
    if reset='1' then
        AMP_REG<=(others=>'0');
    elsif gamp_clk'event and gamp_clk='0' then
        amp_reg<=amp_reg(6 downto 0)&AMP_DOUT;
    end if;
end process;
-- ADC
process(clk,reset)
begin
    if reset='1' then
--        conv<='0';
--        adc_busy<='0';
    elsif clk'event and clk='1' then
--        if sn='1' and adc_busy='1' then
--
--            end if;
--            if conv='1' then
--                conv<='0';
--                adc_busy<='1';
--                sn<='1';
--            end if;
--            if sn='1' then
--                adc_busy<='0';
--            end if;
--            if amp_ready='1' then
--                if adc_busy='0' and conv='0' and cn='1' then
--                    adc_busy<='1';
--                    conv<='1';
--                end if;
--            end if;
--        end if;
    end if;
end process;
process(bclk,adc_busy,reset)
begin
    if reset='1' then
        NE<='0';
        snc<=0;
    elsif adc_busy='0' then
        sn<='0';

```

```

    elsif (bclk'event and bclk='1') then
        if R1='1' then
            ADC_REG1<=ADC_REG1(12 downto 0)&AD_DOUT;
        end if;
        -- if sn='1' then
        if adc_busy='1' then
            snc<=snc+1;
            if snc=3 then
                R1<='1';
            elsif snc=17 then
                R1<='0';
            elsif snc=35 then
                -- snc<=0;
                -- sn<='0';
                -- sn<='1';
                -- NE<='1';
                -- adc_busy<='0';
                -- CEO<='1';
            end if;
        end if;
        -- if adc_busy='1' and sn='0' then
        -- sn<='1';
        -- ss<='0';
        -- end if;
    end if;
end process;
-- DAC
process(gc,reset)
begin
    if reset='1' then
        SPI_MOSI_DAC<='0';
        sdc<='0';
        spc<=0;
    elsif (gc'event and gc='1') then
        sdc<=sd;
        if (sdc='1')and(amp_ready='1') then
            spc<=spc+1;
            case spc is
                when 0 to 6=>
                    SPI_MOSI_DAC<='0';
                when 7 =>
                    SPI_MOSI_DAC<=DAC_COM(3);
                when 8 =>
                    SPI_MOSI_DAC<=DAC_COM(2);
                when 9 =>
                    SPI_MOSI_DAC<=DAC_COM(1);
                when 10 =>
                    SPI_MOSI_DAC<=DAC_COM(0);
            end case;
        end if;
    end if;
end process;

```



```

        sde<='0';
    end if;--sd
    sc<=sc+1;
    if (sc=63) then
        DAC_DAT(11 downto 0)<=DAC_DAT(10 downto
0)&(DAC_DAT(7) xor DAC_DAT(11));
        SD<='1';
    end if;--sc
end if;--clk
end process;

process(clk,reset)
begin
    if reset='1' then
        bclk<='0';
        sd<='0';
        cn<='1';
        DAC_DAT<=(others=>'0');
    elsif (clk'event and clk='1') then
        if (sd='1')or(adc_busy='1') then
            bclk<=not bclk;
        else
            bclk<='0';
        end if;
        if TEST_LACH='1' then
            TEST_LACH<='0';
            sd<='1';
            cn<='0';
        end if;
        if TEST_ADC='1' then
            DAC_DAT(11 downto 0)<=ADC_REG1(13 downto 2);
            DAC_DAT(11 downto 0)<=conv_std_logic_vector(signed((not
ADC_REG1(13 downto 2))+2048),12);
            DAC_DAT(11 downto 0)<=conv_std_logic_vector(signed((not
ADC_REG1(13 downto 2))),12);
            DAC_DAT(11 downto 0)<=conv_std_logic_vector(signed((not
ADC_REG1(11 downto 0))+2048),12);
            DAC_DAT(11 downto 0)<=conv_std_logic_vector(signed((not
ADC_REG1(12 downto 1))+2048),12);
            DAC_DAT(11 downto 0)<=conv_std_logic_vector(signed((not
ADC_REG1(12 downto 2))+2048),12);
            DAC_DAT(11 downto 0)<=conv_std_logic_vector(signed((not
ADC_REG1(13 downto 2))+1),12);
            DAC_DAT(11 downto 0)<=conv_std_logic_vector(signed((not
ADC_REG1(13 downto 2))+1024),12);
            DAC_DAT(11
downto
0)<=conv_std_logic_vector(signed(('0'&(not ADC_REG1(12 downto 2)))+2048),12);

```

```

DAC_DAT(11 downto 0)<=conv_std_logic_vector(signed((not
(ADC_REG1_TEMP))+2048),12);
    if (sn='1' or (adc_busy='0' and conv='0' and amp_ready='1' and
NE='1'))and New_Data='1' then
--
        TEST_LACH<='1';
        sd<='1';
        cn<='0';
    end if;
else
    if ce='1' then
        sd<='1';
        cn<='0';
        DAC_DAT(11 downto 0)<=DAC;
    end if;
end if;
if sde='1' then
    sd<='0';
    cn<='1';
end if;
if sn='1' then
    cn<='0';
end if;
end if;
end process;
--AMP_SHDN<='0';
AMP_SHDN<=reset;
AD_CONV<=conv;
DAC_CS<=not sd;
--DAC_CLR<='1';
READY<=amp_ready;
DAC_CLR<=not reset;
--ADC_REG1_TEMP<=conv_std_logic_vector(5*conv_integer(ADC_REG1(11
downto
0)),14);
--ADC_REG1_TEMP<=conv_std_logic_vector(conv_integer(ADC_REG1(13
downto
3)),14);-- >1
--ADC_REG1_TEMP<=conv_std_logic_vector(conv_integer(ADC_REG1(13
downto
4)),14);-- <1
--ADC_REG1_TEMP<=conv_std_logic_vector(3*conv_integer(ADC_REG1(13
downto
5)),14);
ADC_REG1_TMP1<=ADC_REG1(13 downto 0)&'0'+ADC_REG1(13 downto 0);
ADC_REG1_TEMP<=conv_std_logic_vector(conv_integer(ADC_REG1_TMP1(15
downto
5)),14);
--ADC_REG1_TEMP<=ADC_REG1_TMP1(15 downto 5);
SPI_MOSI<=SPI_MOSI_DAC when amp_ready='1' else SPI_MOSI_AMP;
--SPI_SCK<=bclk;
--SPI_SCK<=clk when sd='1' else '0';
--TEST<=sde;
CEO<=sn;

```

```
--*ADC<=ADC_REG1;  
--*ADC<=conv_std_logic_vector(signed((not (ADC_REG1_TEMP))+2048),12);  
ADC<=ADC_REG1_TEMP;  
--TEST_8<=conv_std_logic_vector(spc,8);  
--SPI_SCK<=dclk when amp_ready='0' else '0';  
SPI_SCK<=amp_clk or bclk;  
end Behavioral;
```