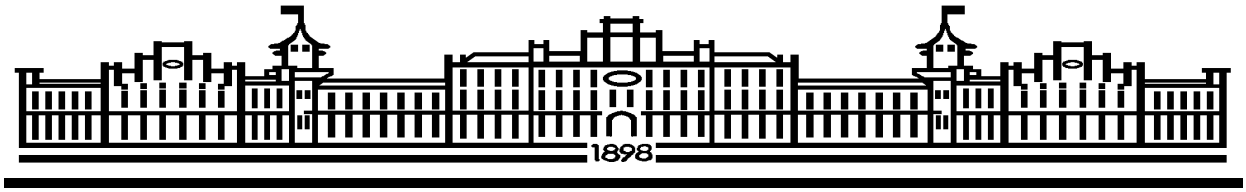


МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Методичні вказівки до виконання лабораторних робіт

по курсу

Технологія проектування комп'ютерних систем — 2

(назва дисципліни)

для напряму підготовки (спеціальностей)

123 Комп'ютерна інженерія .

(шифр та назва напряму, спеціальностей)

(російською мовою)

Уклав
професор Анатолій Михайлович
Сергієнко, д.т.н., с.н.с. кафедри ОТ

Рекомендовано
Вченою радою факультету
інформатики та обчислювальної техніки
НТУУ «КПІ ім. Ігоря Сікорського»
Протокол № 11_ від _7_.__05_____.2019 р.

Київ - 2020

Лабораторная работа 1

Проектирование рекурсивного цифрового фильтра

Цель работы: получить знания и навыки в разработке и тестировании быстродействующих цифровых фильтров на ПЛИС.

Теоретические сведения

Передаточная функция

В цифровой обработке сигналов часто используются рекурсивные цифровые фильтры (РЦФ). РЦФ на базе программируемой логической интегральной схемы (ПЛИС) имеет высокую пропускную способность. РЦФ описывается разностным уравнением N-го порядка с постоянными коэффициентами:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{r=0}^{M} b_r x(n-r). \quad (1)$$

Итак, n-е значение выхода можно вычислить на основе n-го значения входа $x(n)$ и соответственно, N и M прошлых значений выхода $y(n–k)$ и входа $x(n–r)$. Тогда ***импульсную реакцию*** такой системы можно определить как:

$$h(n) = \frac{y(n)}{x(n)} \ . \quad (2)$$

Фильтрацию сигнала x фильтром с импульсной реакцией h называют сверткой. Импульсная реакция (2) в общем случае, является бесконечной и тогда это — система с бесконечной импульсной характеристикой (БИХ).

Операции свертки $h(n)*x(n)$ соответствует умножения: $H(z)X(z)$ в z-пространстве. Следует отметить, что эти свойства свертки справедливы лишь для области распространения переменной z, где рассмотренные функции не расходятся.

Импульсная реакция $h(n)$ системы (2) имеет отображение в z-пространстве как ***передаточная функция***:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{r=0}^{M} b_r z^{-r}}{1 + \sum_{k=0}^{N} a_k z^{-k}}, \qquad (3)$$

где a_k, b_r – действительные числа, функции z^{-m} отвечает задержка сигнала на m тактов дискретизации.

Передаточную функцию (3) можно разложить на сумму элементарных дробей. Чаще всего, например, если $N \geq M$; $Q = N/2$, она факторизуется на произведение дробей:

$$H(z) = \prod_{k=0}^{Q} \frac{b_{0,k} + b_{1,k} z^{-1} + b_{2,k} z^{-2}}{1 + a_{1,k} z^{-1} + a_{2,k} z^{-2}} \quad . \qquad (4)$$

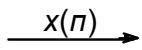
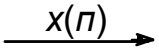
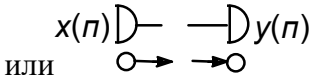
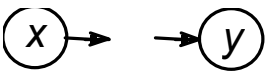
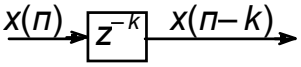
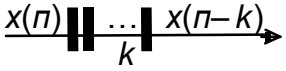
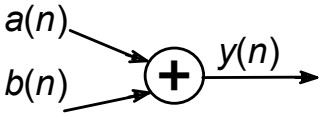
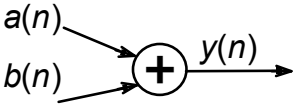
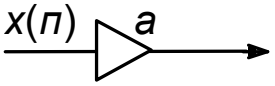
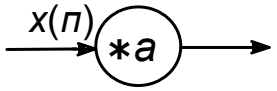
Передаточная функция свидетельствует о спектральных свойствах рассматриваемой линейной системы. На ее основе находят ***амплитудо-частотную*** $|H(e^{-j\omega t})|$ и ***фазо-частотную*** $\arg(H(e^{-j\omega t}))$ характеристики системы.

Графовое представление алгоритма РЦФ

В подавляющем большинстве алгоритмов ЦОС сигнальные потоки являются синхронными. Поэтому такие алгоритмы можно представить графом синхронных потоков данных (ГСПД). Если в результирующем потоке наличие данных условно зависит от входного потока, то такие потоки могут быть несинхронными. Это, например, потоки в компрессоре сигналов, который заменяет цепочки нулевых отсчетов кодами длины этих цепочек.

Сигнальный граф, обычно используемый в ЦОС и однородный ГСПД являются эквивалентными моделями. В табл. 1 показаны графические обозначения элементов сигнального графа, ГСПД и их соответствие частям алгоритма ЦОС.

Таблица 1. Обозначения элементов сигнального графа и ГСПД

Элемент алгоритма	Сигнальный граф	Однородный ГСПД
Сигнал $x(n)$		
Входной и выходной порты, источник и приемник сигналов $x(n)$, $y(n)$		
Задержка на k циклов		
Сложение сигналов, вершина сумматора $y(n) = a(n)+b(n)$		
Умножение сигнала на константу $y(n) = a \cdot x(n)$, вершина блока умножения		

Рассмотрим пример задания алгоритма РЦФ высоких частот с помощью сигнального графа и ГСПД. Передаточная характеристика такого фильтра равна

$$H(z) = \frac{1-z^{-2}}{1+a_1z^{-1}+a_2z^{-2}} = (1-z^{-2})\cdot \frac{1}{1+a_1z^{-1}+a_2z^{-2}}\ , \qquad (5)$$

(сравните с (4)). Сомножителям передаточной функции фильтра соответствуют разностные уравнения (сравните с (1)):

$u(n) = x(n) - x(n-2)$;

$y(n) = u(n) - a_1y(n-1) - a_2y(n-2)$.

Уравнения реализуются в сигнальном графе (рис.1). Считается, что перед началом выполнения все элементы задержки сохраняют нулевые значения. Как только поступает очередное входное данное $x(n)$, дискретизованное с частотой дискретизации f_S, оно сразу идет на элемент задержки на два цикла z^{-2} и на сумматор «+», где оно прибавляется к задержанному данному $x(n-2)$. Остальные элементы модели функционируют так же: как только есть в наличии входные данные —они сразу срабатывают и выдают результат на свой выход.

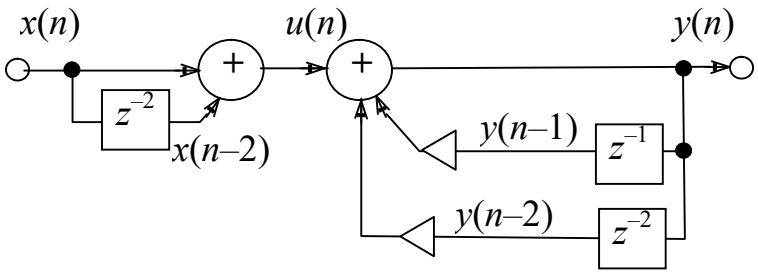
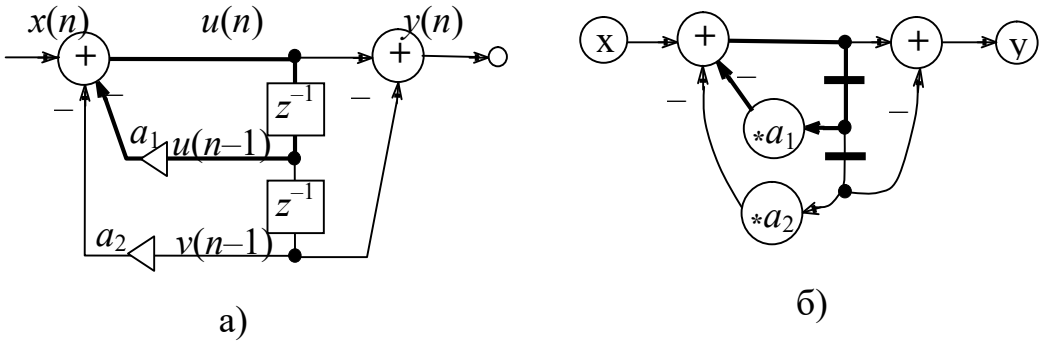


Рис. 1. Сигнальний граф алгоритма фильтра верхних частот

Граф на рис.1 можно рассматривать как структурную схему некоторого специализированного вычислителя, у которого сумматоры — это отображение вершин сложения, блоки умножения — вершин умноженя, а регистры промежуточных операндов — элементов задержки. С этой точки зрения такой вариант алгоритма не является рациональным за чрезмерного количества задержек. Для его оптимизации можно пере¬ста¬вить местами множители в формуле (5) и представить задержку z^{-2} как две последовательные задержки z^{-1}:

$u(n)= x(n) – a_1u(n–1) – a_2v(n–1);$
$v(n) =u(n–1);$
$y(n) = u(n) – v(n–1);$

Результирующий сигнальный граф показано на рис. 2,а. Ему соответствует ГСПД на рис. 2,б. Этот граф принято называть канонической формой рекур-сивного фильтра, так как он содержит минимальное число элементов задержки для хранения промежуточных результатов — задержанных сигналов $u(n)$ и $v(n)$.



а) б)

Рис. 2. Сигнальный граф алгоритма фільтра верхних частот, а) и соответствующий ГСПД, б)

Сигнальный граф и ГСПД могут иметь замкнутые циклы. На рис. 2 такой цикл выделено толстой линией. Если в замкнутом цикле нет ни одного элемента задержки, то сигналы в нем будут бесконечно переприсваиваться в течение одного такта, то есть возникает ***блокировка*** алгоритма. Поэтому обязательным условием отсутствия блокировок в сигнальном графе или ГСПД является наличие, по крайней мере, по одному элементу задержки во каждом замкнутом цикле. Другим условием отсутствия блокировок является наличие начальных, например, нулевых данных во всех элементах задержки, которые входят в такие циклы.

РЦФ на базе фазовых фильтров

Фазовый фильтр имеет модуль передаточной функции $H(z)$, равная $|H(z)| = 1$ Его фазо-частотная характеристика (ФЧХ) на частоте f_R имеет фазовый сдвиг, равный 180°. Если складываются сигналы от двух параллельныъ фазовых фильтров, выходной сигнал подавляется на частотах, для которых разность фаз равна 180°. Результирующая передаточная функция:

$$H_S = (H_1(z) \pm H_2(z))/2, \qquad (6)$$

соответствует различным фильтрам нижних (ФНЧ), высоких частот (ФВЧ), полосового или режекторного фильтра в зависимости знака сложения (ФНЧ или ФВЧ) и от порядка фазовых фильтров. Например, если функция $H_1(z)$ — второго порядка, как в данной лабораторной работе, то при $H_2(z) = 1$ получаем режекторный фильтр, при $H_2(z) = -1$ — полосовой фильтр, при $H_2(z) = \pm z^{-1}$ — ФНЧ (+) и ФВЧ (-).

РЦФ на базе фазовых фильтров отличаются устойчивостью при малой разрядности коэффициентов, высокой линейностью АЧХ и ФЧХ в полосе пропускания, а также высоким быстродействием. Параметры их АЧХ, такие как положение частоты среза, крутизна переходной полосы, напрямую зависят от коэффициентов фильтра.

Передаточная функция ФНЧ, параметры которого можно перенастроить, равна:

$$H_S = (H_1(z) + z^{-1})/2,$$

$$H_1(z) = \frac{b + a(b+1)z^{-1} + z^{-2}}{1 + a(b+1)z^{-1} + bz^{-2}} \qquad (7)$$

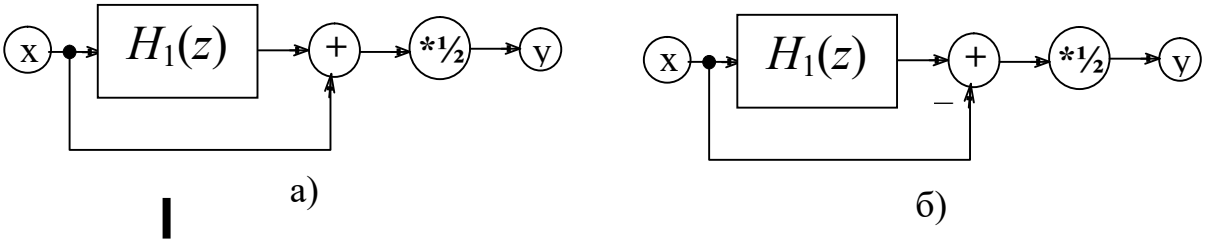
$$a = \cos(2\pi f_R),$$

$$b = (1-t)/(1+t), \qquad (8)$$

$$t = \mathrm{tg}(\pi \Delta f),$$

причем a — регулирует частоту среза f_R, b — задает ширину переходной полосы Δf. Таким образом, изменяя a в (7), полоса среза регулируется в пределах (0,1 – 0,4) f_S с подавлением в полосе задержания до 50 дБ.

ГСПД ФНЧ и полосового фильтров, которые построены в соответствии с (6) показаны на рис.3. Следовательно, эти ГСПД отличаются только знаком сложения и наличием или отсутствием задержки во второй ветке распространения входного сигнала.



а) б)

Рис.3. ГСПД ФНЧ (а) и полосового фильтра (б)

Полосовой фильтр работает следующим образом. Один и тот же сигнал проходит через две ветви графа и от сигнала вычитается его копия на всех частотах кроме резонансной, давая нулевой результат. На резонансной частоте f_R фазовый фильтр возвращает сигнал на 180° и в результате сигнал и его копия складываются. Аналогично работает ФНЧ, но сигнал в фазовом фильтре возвращается на 180° на частотах выше f_R.

Рассмотрим подробно несколько примеров ГСПД звеньев фазового фильтра второго порядка. При непосредственном реализации формулы (7)

в ГСПД необходимо, как минимум, 6 вершин умножения и 6 вершин сумматоров. Есть более эффективный ГСПД для этой формулы, который называют графом волнового фильтра за то, что он является моделью волновода (рис.4).

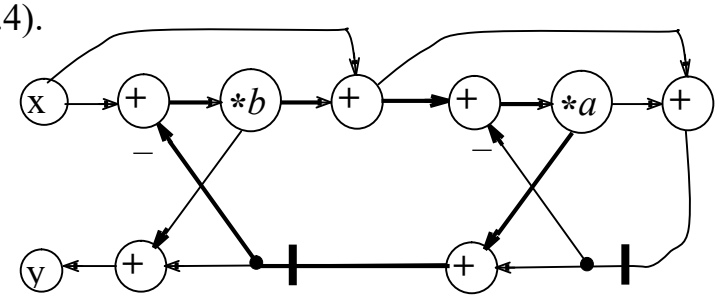


Рис.4. ГСПД для передаточной функции (7)

Формулу (7) можно сократить за счет отказа от возможности прямого регулирования характеристик фильтра:

$$H_{12}(z) = \frac{b + cz^{-1} + z^{-2}}{1 + cz^{-1} + bz^{-2}}, \tag{9}$$

где $c = a(b + 1)$. Соответствующий ГСПД, который представляет каноническую структуру фильтра, показан на рис. 5.

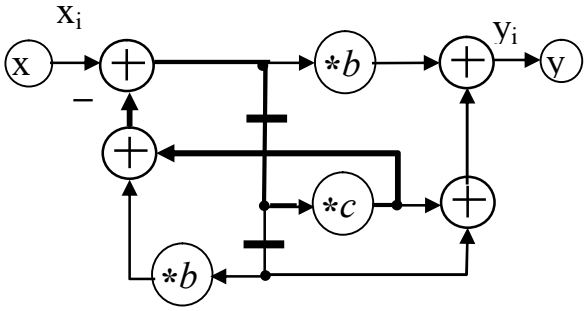


Рис.5. ГСПД для передаточной функции (9) в канонической форме

Формуле (9) соответствует разностное уравнение (10).

$$y_i = b \cdot x_i + c \cdot x_{i-1} + x_{i-2} - c \cdot y_{i-1} - b \cdot y_{i-2} \ . \tag{10}$$

Уравнение (10) можно переписать так

$$q_i = x_i - c \cdot q_{i-1} - b \cdot q_{i-2} \ ; \tag{11}$$

$$y_i = b \cdot q_i + c \cdot q_{i-1} + q_{i-2}.$$

Если не экономить задержки в дугах, то уравнения (11) выполняются в ГСПД, таком как на рис.6. Зато получим лишь две операции умножения.

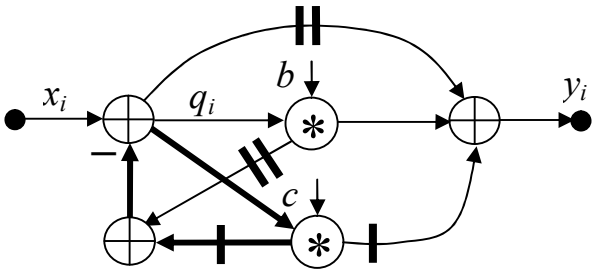


Рис.6. ГСПД для передаточной функции (9) с избыточным количеством задержек

При вынесении в (10) общих коэффициентов за скобки получим уравнение:

$$y_i = b\cdot(x_i - y_{i-2}) + c\cdot(x_{i-1} - y_{i-1}) + x_{i-2}\,. \qquad (12)$$

Непосредственно по уравнению (12) построен ГСПД, показанный на рис. 7. За счет повторного использования задержек авторам Mitra и Hirano удалось построить ГСПД под названием MH2B, который показан на рис. 8.

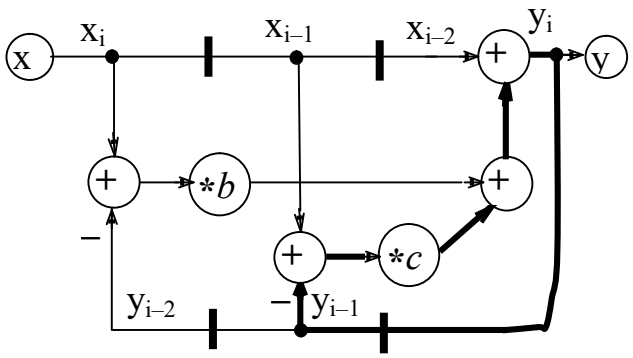


Рис.7. ГСПД для уравнения 12

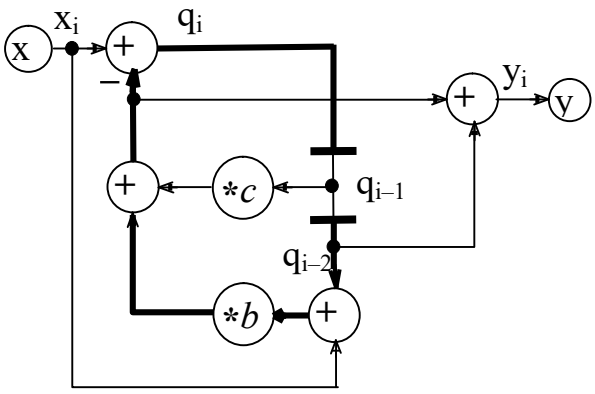


Рис.8. ГСПД MH2B

Толстой линией в ГСПД на рис. 4 – 8 показан критический путь. Длина этого пути максимальна — $4t_S + 2t_M$ для ГСПД на рис. 4 и минимальна — для ГСПД на рис. 5 и 6 — $2t_S + t_M$ за счет большего числа вершин умножения.

Тестирование модели цифрового фильтра

Определение АЧХ фильтра является типичной процедурой его диагностики и тестирования. Для анализа или измерения АЧХ и ФЧХ на вход проверяемого РЦФ следует подавать комплексный сигнал $e^{-j\omega n}$, который называют ***аналитическим***. Часто используют простой способ проверки, когда вместо аналитического сигнала подают только его составляющую – $\cos(\omega n)$, а АЧХ измеряют на выходе системы, как максимум результирующего сигнала $Re(H(\omega))$, то есть в моменты, когда вторая составляющая – $\sin(\omega n) = 0$. Недостаток этого способа — в неточности измерения максимума сигнала.

Более точный способ заключается в получении мнимой составляющей $Im(H(\omega))$ после пропускания результата $Re(H(\omega))$ через фильтр Гильберта, поворачивающий фазу сигнала на 90°. Но такой фильтр вносит существенные искажения в частотную характеристику. Поэтому для анализа РЦФ следует применить сигнальный граф, такой как на рис. 9. В нем используются два идентичных экземпляра РЦФ с функцией $H(z)$, на которые подаются синусная и косинусная составляющие аналитического сигнала.

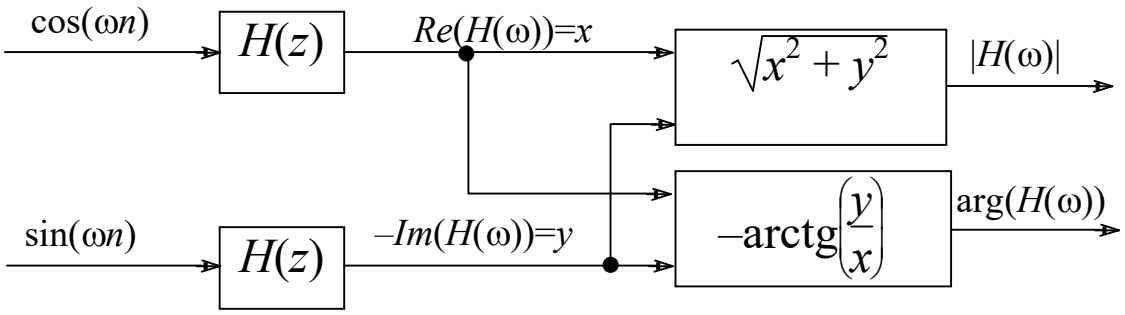


Рис. 9. Измерение АЧХ, ФЧХ реальной системы $H(z)$

Соответственно, с выходов РЦФ снимаются составляющие аналитического сигнала отклика РЦФ.

В лабораторной работе предлагается стенд для испытаний, имеющий структуру как на рис. 10, который можно скачать по адресу: http://kanyevsky.kpi.ua/en/useful-ip-cores/testbench-for-the-filter-testing/
Порты и настроечные константы этого стенда представлены в табл.2.

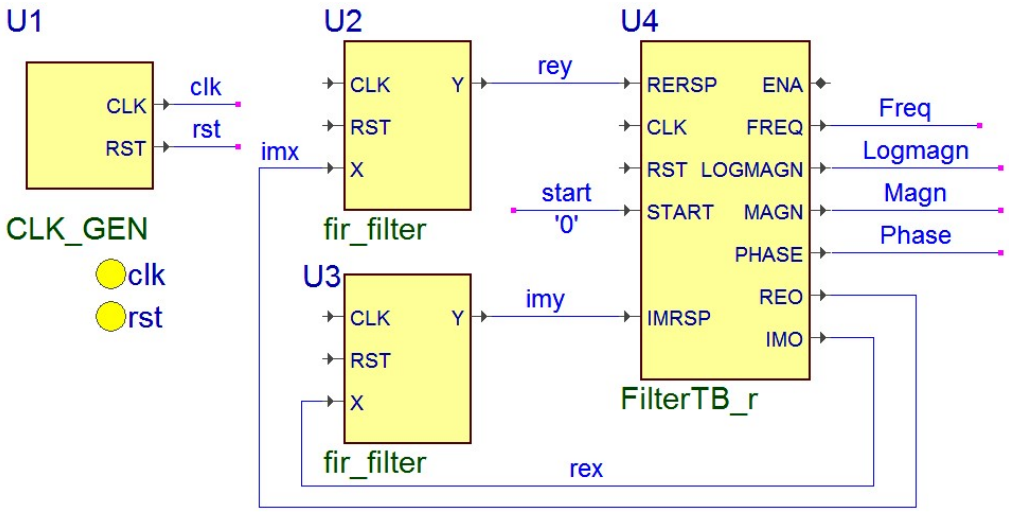


Рис.10. Испытательный стенд для фильтров

Выбор разрядности фильтра

Подавляющее большинство РЦФ вычисляется в компьютерах или ПЛИС в арифметике целых чисел или чисел с фиксированной запятой. При программировании такого фильтра синтезируют набор коэффициентов, удовлетворяющих заданным требованиям и представленных с плавающей запятой. Затем выбирают количество битов квантования коэффициентов $n_к$, входных данных n_x и результатов n_y.

Как правило, $n_x, n_y \geq \log_2 10 \cdot D/20$, где D — динамический диапазон изменения сигнала, dB. То есть на каждые 6 децибел динамического диапазона приходится не менее одного разряда.

Таблица 2. Порты и настроечные константы виртуального модуля стенда для испытаний цифровых фильтров

Имя порта	**Назначение порта**
fsampl	Частота дискретизації f_o, наприклад 1000 кГц
fstrt	Початкова частота діапазону аналізу
deltaf	Приріст частоти d, так що через k кроків генератор видаватиме частоту $f_o + k \cdot d$
maxdelay	Затримка у кількості відліків від початку генерації частоти, після якої виконується оцінка результатів фільтрації. Має бути більшою за подвоєну максимальну групову затримку фільтра, який тестується.
slowdown	Коефіцієнт зменшення швидкості фільтра відносно тактової частоти. Якщо вхідні відліки поступають у кожному такті, то slowdown=1, якщо вони приходять у кожному парному такті, то slowdown=2 і т.д.
magnitude	Амплітуда синусного і косинусного сигналів, що генеруються.
REO,IMO	Синусний та косинусний сигнали, що генеруються
RERSP, IMRSP	Сигнали з виходів фільтрів, які є відгуками на синусний та косинусний сигнали, відповідно
FREQ	Код частоти сигналів, що генеруються, який дорівнює $f_o + k \cdot d$
MAGN	Обчислена амплітуда комплексного сигналу RERSP, IMRSP на частоті FREQ
LOGMAGN	Обчислена амплітуда комплексного сигналу RERSP, IMRSP на частоті FREQ у логарифмічному масштабі, тобто у децибелах, причому сигнал з амплітудою magnitude приймається за 0 дБ.
PHASE	Обчислена фаза комплексного сигналу RERSP, IMRSP на частоті FREQ, представлена у діапазоні $\pm\pi$
RST,START	Сигнали початкового встановлення
ENA	Сигнал дозволу прийому RERSP, IMRSP при slowdown >1

Коэффициенты масштабируют и округляют, так что при $|b_i| < 1$ целый коэффициент был равен

$$b'_i =]2^{n_x} \cdot b_i + 0.5[\, . \qquad (11)$$

Результаты фильтра вычисляют по формуле (1). При этом сумматор, накапливающий результат, должен иметь такую разрядность, чтобы не возникало переполнения. Причем разрядность произведения равна $n_д = n_к + n_х$, а разрядность сумматора должна быть не менее величины $n_с = \log_2 S + n_к + n_х$, где S — теоретически возможный максимальный результат формулы (1) при условии, что не утрачивается информация при

округлении результатов. Для правой половины формулы, соответствующей КИХ-фильтру, максимум суммы равен сумме модулей всех коэффициентов фильтра, стоящих в числителе формулы передаточной функции, т.е.

$$n_{\text{с}} = \log_2 \sum_{i=0}^{M} |b_{\text{i}}| + n_{\text{к}} + n_{\text{х}}. \qquad (12)$$

Для левой половины формулы (1) максимум суммы может быть значительно больше из-за усиления сигнала из-за обратной связи. Такое усиление пропорционально ***добротности фильтра*** — коэффициенту усиления сигнала на резонансной частоте. Для фазового фильтра (7), (8) добротность фильтра резко возрастает при $b \to 1$.

На практике, для фазовых фильтров и фильтров на их основе, благодаря их малой чувствительности к ошибкам округления, разрядность промежуточных результатов выбирают как

$$n_{\text{с}} = n_{\text{д}} + n_{\text{х}} + 3, \qquad (13)$$

где $n_{\text{д}} = 1,...,8$ выбирается в зависимости от добротности фильтра и уточняется во время его моделирования. То есть, это значение уменьшается до такой величины, пока не произойдет событие переполнения разрядной сетки.

Разрядность коэффициентов фазовых фильтров может быть меньше разрядности коэффициентов в других моделях РЦФ. Достаточно разрядности $n_{\text{к}} = n_{\text{х}}$. Эта разрядность может быть уменьшена по результатам тестирования фильтра при условии выполнения требований к АЧХ.

Результат фильтра $y(n)$ берется как старшие $n_{\text{у}}$ разрядов суммы (1) с отбрасыванием младших разрядов, то есть с усечением или по другому алгоритму округления.

Поскольку $n_{\text{с}}$ может быть довольно большим числом, а вероятность достижения результатом максимального значения может быть мала (на резонансных частотах), то на практике $n_{\text{с}}$ выбирают несколько меньше, а

сложение в (1) выполняют по алгоритму накопления с насыщением. По этому алгоритму, если возникает переполнение суммы, то результат заменяется максимальным числом разрядности $n_с$ с соответствующим знаком. Насыщение сигнала моделирует аналогичный процесс в аналоговых схемах и дает значительно меньше искажения сигнала чем переполнение разрядной сетки.

Следующий пример показывает реализацию насыщения в VHDL. Пусть $n_с = 14$, $n_д = 3$ и $n_у = 8$, причем считается, что результат по амплитуде меньше 1, а фиксированная запятая стоит перед 8-м разрядом. Тогда операция насыщения накопленного результата S программируется как

```
Y <= x”7F” when S(13 downto 10) > signed(“0001”),
     X”80” when S(13 downto 10) < signed(“1111”), else
     S(11 downto 3);
```

Здесь константы x”7F” и x”80” представляют максимальне число 0,99 и мінімальное число –0,99.

Задания для работы

Разработать VHDL-проект цифрового фильтра с передаточной функцией (6).

Вид составляющих $H_1(z)$ и $H_2(z)$ задается по варианту из таблицы 3. Номер задания этой и других лабораторных работ совпадает с номером студента в списке группы. При этом в таблице 2 задано $H_2(z)$, частота среза (резонанса) f_R, ширина переходной полосы Δf и вид ГСПД. Коэффициенты *a, b* для функции (7) рассчитываются по формулам (8), а коэффициент *c* для функции (9) вычисляется как $c = a\,(b + 1)$.

Разрядность входных и выходных данных n_x, n_y:

в первой группе потока разрядность – 14,

во второй – разрядность – 16,

в третьей – разрядность – 24,

в четвертой – разрядность – 18.

Таблица 3. Параметры и функции к лабораторной работе

№ вар.	f_R,	Δf	**Рис. ГСПД** $H_1(z)$	$H_2(z)$
1	0,1	0,05	4	z^{-1}
2	0,125	0,05	5	z^{-1}
3	0,15	0,05	6	z^{-1}
4	0,175	0,05	7	z^{-1}
5	0,20	0,05	8	z^{-1}
6	0,225	0,05	4	z^{-1}
7	0,25	0,05	5	z^{-1}
8	0,275	0,05	6	z^{-1}
9	0,1	0,05	7	$-z^{-1}$
10	0,125	0,05	8	$-z^{-1}$
11	0,15	0,05	4	$-z^{-1}$
12	0,175	0,05	5	$-z^{-1}$
13	0,20	0,05	6	$-z^{-1}$
14	0,225	0,05	7	$-z^{-1}$
15	0,25	0,05	8	$-z^{-1}$
16	0,275	0,05	4	$-z^{-1}$
17	0,1	0,1	6	1
18	0,125	0,1	7	1
19	0,15	0,1	8	1
20	0,175	0,1	4	1
21	0,20	0,1	5	1
22	0,225	0,1	6	1
23	0,25	0,1	7	1
24	0,275	0,1	8	1
25	0,125	0,1	4	–1
26	0,15	0,1	5	–1
27	0,175	0,1	6	–1
28	0,20	0,1	7	–1
29	0,225	0,1	8	–1
30	0,25	0,1	4	–1

Разрядность коэффициентов равна $n_к = n_x - 2$. Разрядность внутренних промежуточных результатов определяется отношением (13).

Модель фильтра является описанием заданного ГСПД на языке VHDL. ГСПД оптимизируется путем ресинхронизации и конвейеризации. Например, целесообразно добавить задержки на входе и выходе ГСПД, которые отражаются в соответствующие регистры входного и выходного сигналов.

VHDL-описание фильтра должно иметь соответствующие комментарии, которые указывают автора и объясняют выполнения алгоритма.

Разработанный фильтр протестировать на стенде для испытаний, таком, как на рис. 8.

Также фильтр нужно синтезировать в САПР ПЛИС (Xilinx или Intel) для ПЛИС, выбранной произвольно, с размещением и трассировкой.

Оформить протокол лабораторной работы, в котором привести

— алгоритм фильтра, оптимизированный ГСПД фильтра;

— VHDL-текст описания фильтра;

— графики АЧХ испытания фильтра в линейном и логарифмическом масштабах;

— результаты синтеза — аппаратные затраты и минимальный период тактовой частоты выбранной ПЛИС как скриншот работы САПР ПЛИС.

Пример выполнения

Пусть имеем ГСПД такой как на рис. 8 и необходимо разработать ФНЧ с частотой среза f_R =0,25 (это, так называемый, полуполосный фильтр) и переходной полосой Δf = 0,1. Разрядность входных даннных – 12.

Тогда, согласно (8) и (9), $a = \cos(2\pi \cdot 0{,}25) = 0$; $t = \operatorname{tg}(\pi \cdot 0{,}1) = 0{,}325$; $b = (1–0{,}325)/(1+0{,}325) = 0{,}509$; $c = a(1 + b) = 0$. Итак, умножение на c удаляется. Результирующий ГСПД фильтра показан на рис. 11.

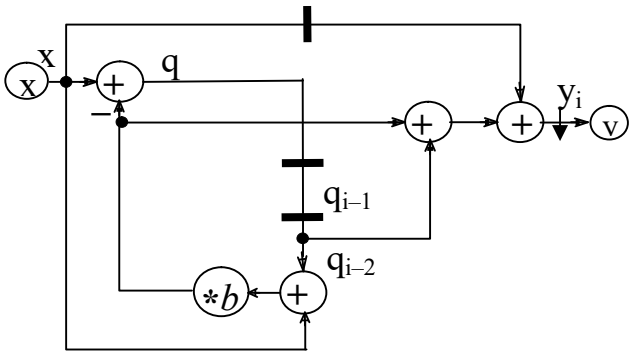


Рис. 11. ГСПД полуполосного ФНЧ

На нем стрелкой поперек дуги представлен сдвиг вправо на один разряд, то есть, деление пополам.

Далее этот ГСПД подлежит ресинхронизации путем конвейеризации. При этом дуги графа нагружаются задержками таким образом, чтобы алгоритм оставался неизменным, за исключением латентной задержки и чтобы критический путь был минимизирован. Результирующий ГСПД фильтра показан на рис. 12. На нем показаны критический путь и все сигналы, которые участвуют в вычислениях.

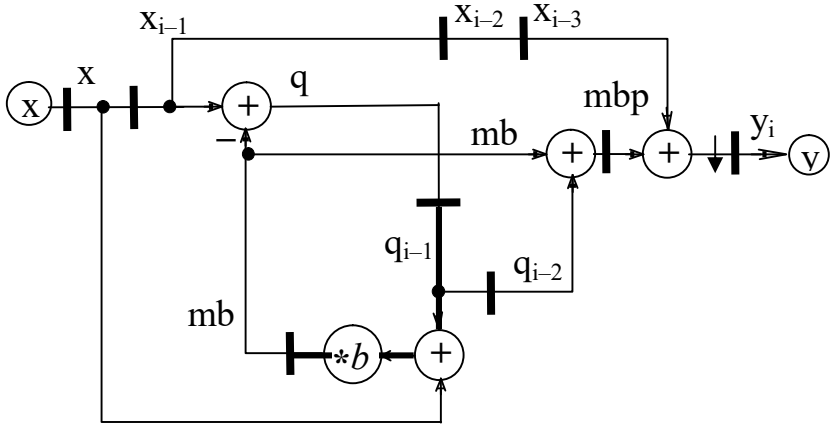


Рис. 12. Конвееризованный ГСПД полуполосного ФНЧ

Выбирается разрядность промежуточных данных $n_с = n_д + n_х + 3 =$ $= 2+12+3 = 17$. Разрядность коэффициентов $n_к = 12$. Разрядность произведения $n_п = n_к + n_с = 17+12= 29$.

Результирующее VHDL-описание фильтра представлено ниже.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
entity LPF_HB_LAB is
     port(     CLK : in STD_LOGIC;
          RST : in STD_LOGIC;
          X : in STD_LOGIC_VECTOR(11 downto 0);
          Y : out STD_LOGIC_VECTOR(11 downto 0)
          );
end LPF_HB_LAB;

architecture synt of LPF_HB_LAB is
     constant b:signed(11 downto 0):=
           to_signed(integer(0.509*2.0**11),12);
     constant max:signed(3 downto 0):="0111";
     constant min:signed(3 downto 0):="1100";
```

```
     signal xi,xi_1,xi_2,xi_3:signed(16 downto 0);
     signal qi,qi_1,qi_2,mbp:signed(16 downto 0);
     signal mb:signed(28 downto 0);
     signal yi:signed(11 downto 0);
begin
     LPF:process(CLK,RST)
       variable yt:signed(16 downto 0);
     begin
          if RST = '1' then
               xi<=(others=>'0');
               xi_1<=(others=>'0');
               xi_2<=(others=>'0');
               xi_3<=(others=>'0');
               yi<=(others=>'0');
               qi_1<=(others=>'0');
               qi_2<=(others=>'0');
               mb<=(others=>'0');
               mbp<=(others=>'0');
          elsif CLK='1' and CLK'event then
               xi<= RESIZE(signed(X&"000"),17);
               xi_1<= xi;
               xi_2<= xi_1;
               xi_3<= xi_2;
               mb  <= b*(xi + qi_1);
               qi_1<= xi_1 - mb(27 downto 11);
               qi_2<= qi_1;
               mbp <= qi_2 + mb(27 downto 11);

               yt:= mbp + xi_3;
               if yt(16 downto 13) > max  then
                    yi<= x"7ff";
               elsif yt(16 downto 13) < min  then
                    yi<= x"800";
               else
                    yi<=yt(14 downto 3);
               end if;
          end if;
     end process;
      Y<= std_logic_vector(yi);

end synt;
```

Результаты испытаний модели фильтра в виде АЧХ, логарифмической АЧХ и ФЧХ показаны на рис. 13. При этом входной сигнал имеет амплитуду $2000 < 2^{11}$, частота дискретизации равна 1000 условных единиц.

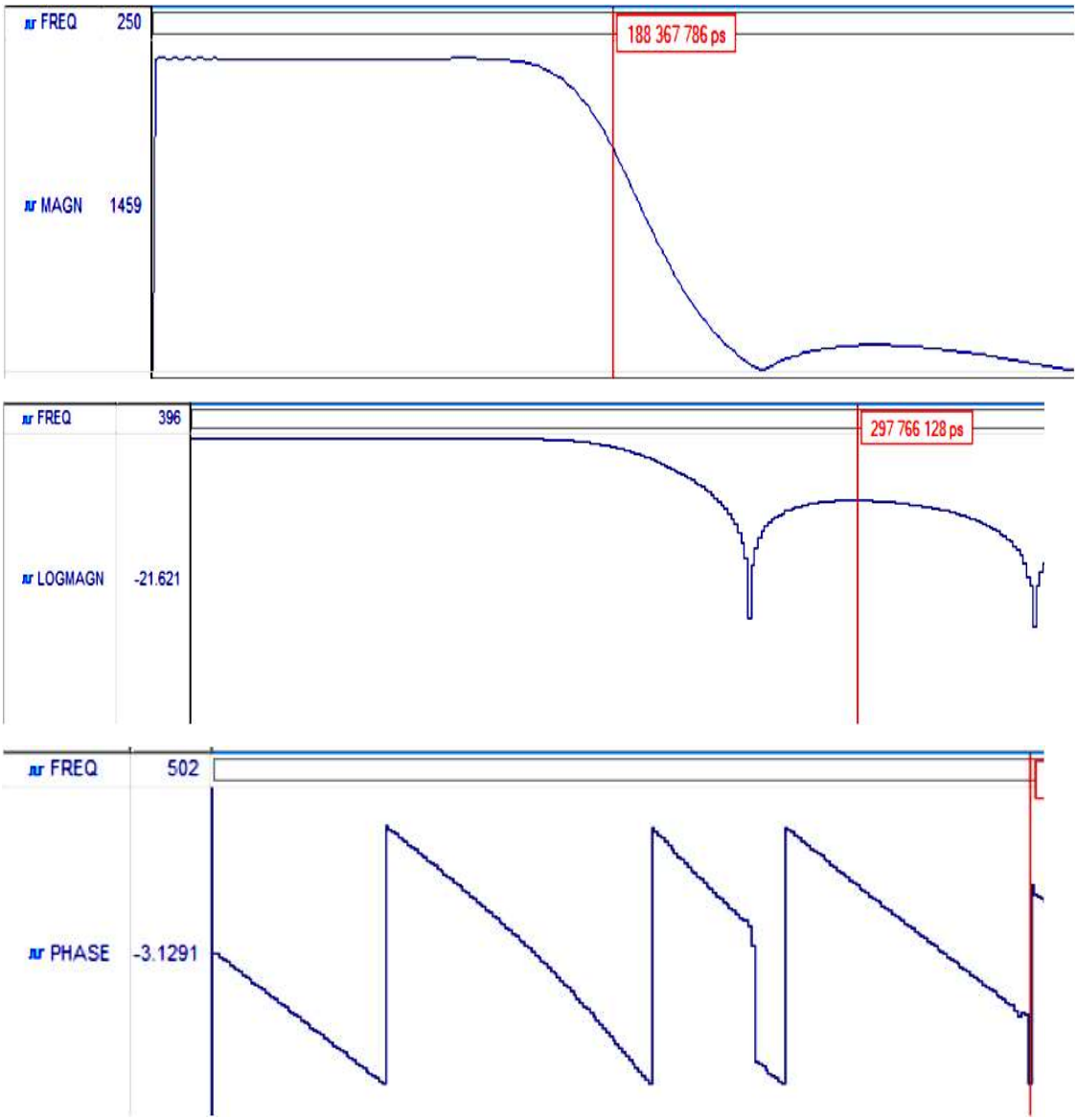


Рис.13. АЧХ, логарифмическая АЧХ и ФЧХ построенного фильтра

Из полученных графиков видно, что фильтр имеет уровень подавления 21 дБ, является действительно полуполосным (на частоте 250 передаточная функция равна $1459/2000 \approx \sqrt{0{,}5}$), его фазовая характеристика более или менее линейная, но на частоте 332 фаза резко меняется на угол π, причем на этой частоте как раз происходит провал в АЧХ.

При конфигурировании фильтра в ПЛИС Xilinx Spartan-6 получены следующие результаты.

Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	125	18,224	1%
Number used as Flip Flops	125		
Number used as Latches	0		
Number used as Latch-thrus	0		
Number used as AND/OR logics	0		
Number of Slice LUTs	91	9,112	1%
Number used as logic	68	9,112	1%
Number using O6 output only	59		
Number using O5 output only	0		
Number using O5 and O6	9		
Number used as ROM	0		
Number used as Memory	0	2,176	0%
Number used exclusively as route-thrus	23		
Number with same-slice register load	23		
Number with same-slice carry load	0		
Number with other load	0		
Number of occupied Slices	31	2,278	1%
Nummber of MUXCYs used	72	4,556	1%
Number of LUT Flip Flop pairs used	107		
Number of DSP48A1s	1	32	3%

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
ts_clk = PERIOD TIMEGRP "CLK" 7.93 ns HIGH 50%	SETUP	0.000ns	7.930ns	0	0
	HOLD	0.382ns		0	0

Итак, аппаратные затраты синтезированного фильтра составляют 31 CLB slices и 1 блок DSP48. Тактовая частота фильтра достигает f_C = 1/7.93 = 126 МГц.

Лабораторная работа 2

Проектирование цифрового фильтра без блоков умножения

Цель работы: получить знания и навыки в разработке и тестировании быстродействующих цифровых фильтров на ПЛИС с использованием техник уменьшения аппаратных затрат.

Теоретические сведения

Маскирующие фильтры

При последовательном соединении ступеней цифровых фильтров результирующая АЧХ является пересечением АЧХ этих ступеней. При этом говорят, что АЧХ ступени маскирует АЧХ других ступеней, то есть, такая ступень является маскирующим фильтром (рис. 14). Благодаря маскированию, результирующий фильтр, состоящий из простых ступеней фильтра, имеет высокое качество АЧХ.

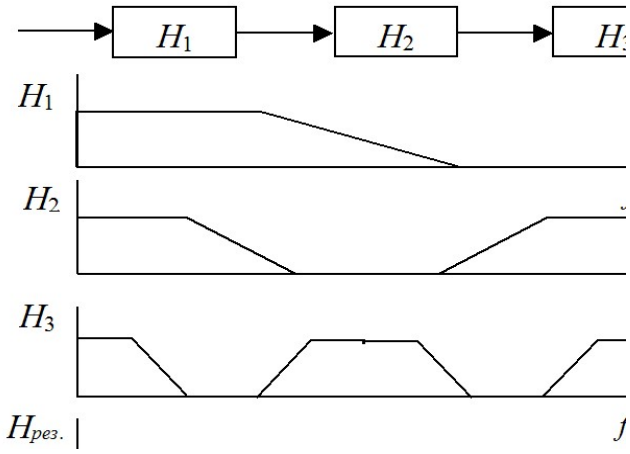


Рис.14. Пример трехступенчатого фильтра с маскирующими фильтрами

Фильтры с кратными задержками

Каждому члену z^{-k} в передаточной функции $H(z)$ в сигнальном графе фильтра соответствует задержка на k циклов или цепочку k регистров задержки в структуре фильтра. Если в фильтре количество регистров задержки увеличить в n раз, то получается фильтр с АЧХ $H_n(z) = H(zn)$.

АЧХ этого фильтра имеет форму такую же, как у фильтра-прототипа $H(z)$, но в диапазоне 0 — f_S она повторяется n раз, где f_S — частота дискретизации. На рис. 14 $H_1 = H(z)$, $H_2 = H(2z)$, $H_3 = H(4z)$.

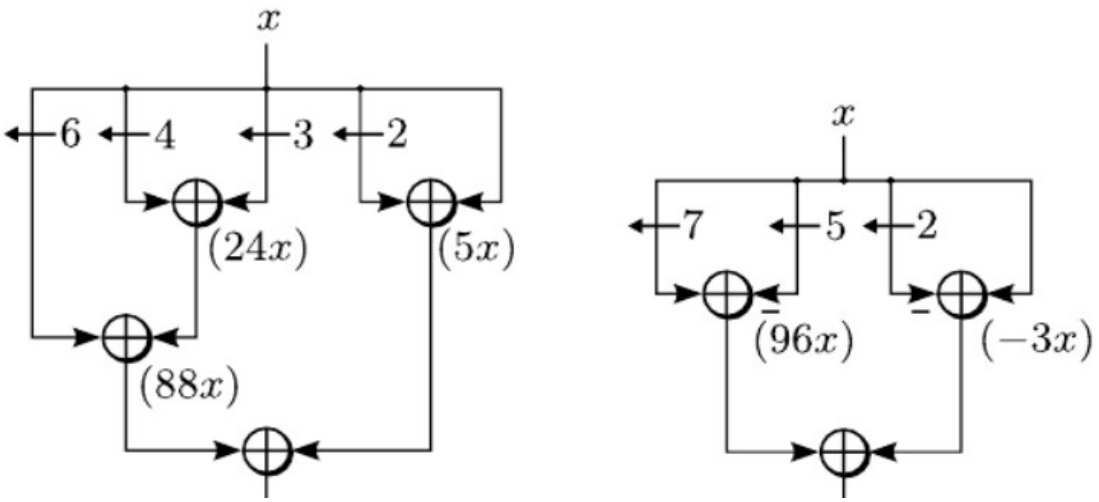
Замена блока умножения на константу

Большинство блоков умножения в цифровой обработке данных — это блоки умножения на константу. Если константа может быть представлена в канонической системе счисления с небольшим количеством ненульвих разрядов, то стоит блок умножения заменить на специализированный умножитель в виде дерева сумматоров частичных произведений.

Например, пусть имеем константу $y = 93_{10} = 1011101_2$. Тогда произведение будет

$$y \cdot x = 93x = (2^6 + 2^4 + 2^3 + 2^2 + 1)x,$$

то есть, при представлении множителя y в двоичной системе имеем 5 ненулевых разрядов, умножения на которые заключается в сдвиге множимого x на соответствующее количество разрядов и сложении на дереве из 4 сумматоров. Схема такого дерева показана на рис. 15, а, на котором горизонтальными стрелками показано сдвиг вправо на соответствующее количество разрядов.



а) б) в)

Рис.15. Примеры реализации блока умножения на коэффициент

Если константу представить в знаковой канонической системе счисления, то количество ненулевых разрядов уменьшается. При этом следует заметить, что сложности сумматора и вычитателя одинаковы. Например,

$$y = 93_{10} = 1011101_2 = 1100\bar{1}01_2 = 10\bar{1}00\bar{1}01_2 = (2^7 - 2^5 - 2^2 + 1).$$

Тогда количество сумматоров уменьшается до трех, как на рис. 15, д. Еще улучшить схему можно после факторизации множителя. Например,

$$10\bar{1}00\bar{1}01_2 = (2^1 + 1)\cdot(2^5 - 1).$$

Тогда количество сумматоров уменьшается до двух, как на рис. 15 в.

Завдание для работы

Разработать VHDL-проект цифрового фильтра без использования блоков умножения с передаточной функцией

$H(z) = H_4(z)\cdot H_M(z)$,

где $H_4(z) = H_3(zk)$,

$H_3(z) = (H_1(z) + H_2(z))/2$,

$H_1(z)$ является такой же, как в лабораторной работе 1 и реализуется в соответствующем ГСПД, а $H_M(z)$, $H_2(z)$, *a, b, k* выбираются из таблицы 4.

Модель фильтра является описанием заданного ГСПД на языке VHDL. ГСПД обязательно оптимизируется путем ресинхронизации или конвейеризации, так как этому способствует использование задержек кратности *k*.

Разработанный фильтр протестировать на стенде для испытаний, таком, как на рис. 8.

Также фильтр просинтезировать с размещением и трассировкой в САПР ПЛИС (Xilinx или Intel) для ПЛИС, выбранной произвольно.

Оформить протокол лабораторной работы, в котором привести

— алгоритм фильтра, ГСПД фильтра;

— VHDL-текст описания фильтра, который должен иметь соответствующие комментарии, указывающие автора и объясняют выполнения алгоритма.

Таблица 4. Параметры и функции к лабораторной работе

№ вар.	a	b	$H_2(z)$	k	$H_M(z)$
1	–0,3125	0,75	–1	2	$(1+z^{-1}+z^{-2}+z^{-3}+z^{-4})/8$
2	–0,125	0,75	–1	2	$(1-z^{-1}+z^{-2}-z^{-3}+z^{-4})/8$
3	–0,625	0,75	–1	2	$(1+4z^{-1}+6z^{-2}+4z^{-3}+z^{-4})/16$
4	–0,875	0,75	–1	2	$(1-4z^{-1}+6z^{-2}-4z^{-3}+z^{-4})/16$
5	–0,3125	0,5	$-z^{-1}$	2	$(1+4z^{-1}+6z^{-2}+4z^{-3}+z^{-4})/16$
6	–0,75	0,5	$-z^{-1}$	2	$(1+z^{-1}+z^{-2}+z^{-3}+z^{-4})/8$
7	–0,5	0,5	$-z^{-1}$	2	$(-1+3z^{-1}+5z^{-2}+3z^{-3}-z^{-4})/8$
8	–0,25	0,5	$-z^{-1}$	2	$(1+5z^{-1}+10z^{-2}+10z^{-3}+5z^{-4}+z^{-5})/32$
9	–0.875	0,5	z^{-1}	2	$(1+z^{-1}+z^{-2}+z^{-3}+z^{-4}+z^{-5}+z^{-6}+z^{-7})/8$
10	0.5	0,5	z^{-1}	2	$(1+z^{-1})(1+z^{-1}+z^{-2}+z^{-3})/8$
11	–0,25	0,25	z^{-1}	3	$(2+5z^{-1}+7z^{-2}+5z^{-3}+2z^{-4})/32$
12	–0,5	0,25	z^{-1}	3	$(1+z^{-1}+z^{-2}+z^{-3}+z^{-4}+z^{-5})/8$
13	–0,125	0,25	z^{-1}	3	$(1+z^{-1}+2z^{-2}+z^{-3}+z^{-4})/8$
14	–0,75	0,25	z^{-1}	3	$(1-z^{-1}+z^{-3}-z^{-4})/4$
15	–0,75	0,25	z^{-1}	3	$(1+0.7z^{-1}-0.7z^{-3}-z^{-4})/4$
16	–0,625	0,25	z^{-1}	3	$(1-z^{-1}+z^{-2}-z^{-3}+z^{-4}-z^{-5})/8$
17	–0,75	0,25	z^{-1}	3	$(1-0.7z^{-1}+0.7z^{-3}-z^{-4})/4$
18	–0,5	0,25	z^{-1}	2	$(1+z^{-1}+z^{-3}+z^{-4})/4$
19	0.25	0,25	z^{-1}	2	$(-1+z^{-1}+z^{-3}-z^{-4})/4$
20	–0,625	0,25	z^{-1}	2	$(1-z^{-1}+z^{-3}-z^{-4})/4$
21	–0,5	0,25	z^{-1}	2	$(-1+2z^{-2}-z^{-4})/4$
22	–0,75	0,25	z^{-1}	2	$(1+2z^{-1}+2z^{-2}+2z^{-3}+z^{-4})/8$
23	–0,6	0,25	z^{-1}	2	$(1-2z^{-1}+2z^{-2}-2z^{-3}+z^{-4})/8$
24	0	0,25	z^{-1}	2	$(1+z^{-1}+z^{-2}+z^{-3}+z^{-4}+z^{-5}+z^{-6})/8$
25	–0.1	0,25	z^{-1}	2	$(1-z^{-2}+z^{-4}-z^{-6})/4$
26	–0.15	0,25	z^{-1}	3	$(1+1.4z^{-1}+z^{-2}-z^{-4}-1.4z^{-5}-z^{-6})/8$
27	–0,5	0,25	z^{-1}	3	$(1+z^{-1}+z^{-2}+z^{-3}+z^{-4})/8$
28	–0,625	0,25	z^{-1}	3	$(-3-2z^{-1}+5z^{-2}+5z^{-3}-2z^{-4}-3z^{-5})/32$
29	–0,5	0,25	z^{-1}	3	$(1+5z^{-1}+10z^{-2}+10z^{-3}+5z^{-4}+z^{-5})/32$
30	–0,75	0,25	z^{-1}	3	$(3-2z^{-1}+5z^{-2}+5z^{-3}-2z^{-4}+3z^{-5})/32$

— графики АЧХ испытания фильтра в линейном и логарифмическом масштабах;

— результаты синтеза — аппаратные затраты и минимальный период тактовой частоты выбранной ПЛИС как скриншот работы САПР ПЛИС.

Номер задания этой лабораторной работы совпадает с номером студента в списке группы.

Разрядность входных, выходных и промежуточных данных такая же, как в лабораторной работе 1.

Пример выполнения

Пусть имеем ГСПД части БИХ-фильтра такую, как на рис. 11 и надо разработать ФНЧ с $a = 0$, $b = 0{,}5625$, $k = 2$;

$H_M(z) = (1 + 4z^{-1} + 6z^{-2} + 4z^{-3} + z^{-4})/16$. Разрядность входных данных - 12.

ГСПД фильтра показан на рис. 16. Он состоит из БИХ-части (слева) и КИХ-части (справа).

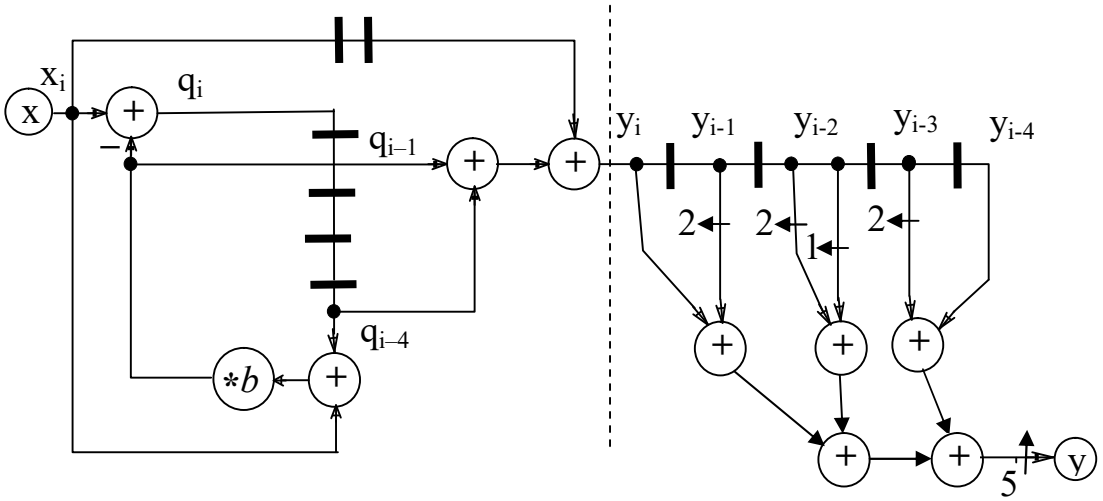


Рис. 16. ГСПД ФНЧ

Далее этот ГСПД подвергается ресинхронизации с использованием конвейеризации. Также блок умножения на b заменяется на специализированный блок умножения на основе сумматора. При этом $b = 0{,}4375 = 0{,}100\bar{1}$ означает, что умножение выполняется как вычитание

операнда, сдвинутого на 4 разряда из операнда, сдвинутого на 1 разряд. Аналогично реализовано умножения на коэффициенты в КИХ-части. Результирующий ГСПД фильтра показан на рис. 17. На нем показаны все сигналы, которые участвуют в вычислениях.

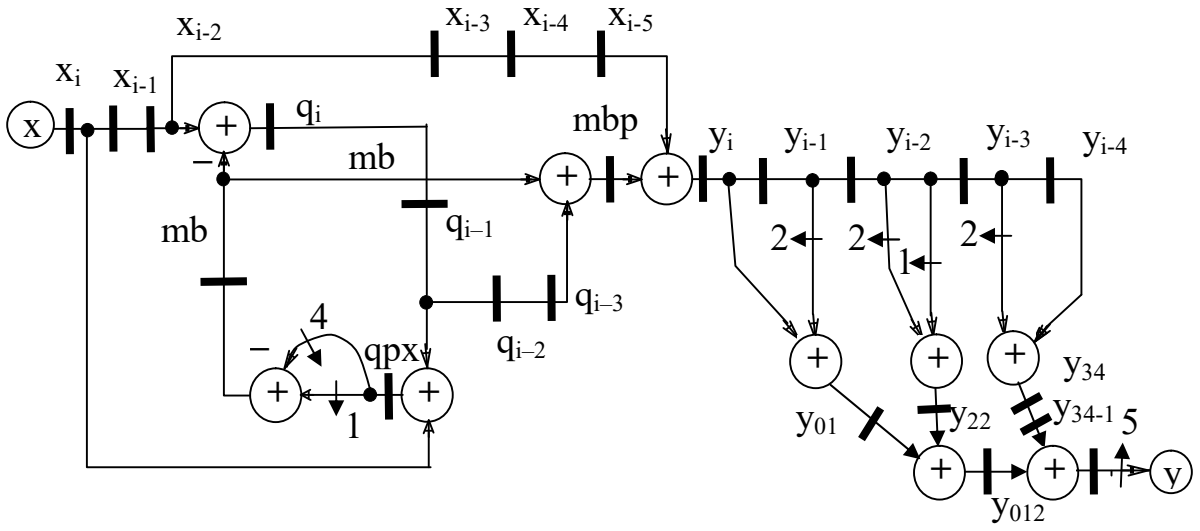


Рис. 17. Конвейеризированный ГСПД ФНЧ

Результат фильтрации делится на 32 с помощью сдвига на 5 разрядов, принимая во внимание коэффициенты передачи БИХ и КИХ-частей. Критический путь минимизирован до задержки одного сумматора, благодаря ресинхронизации и конвейеризации, а также замене умножения на сложение.

Выбирается разрядность промежуточных данных в КИХ-части $n_с = n_д + n_х + 3 = 2+12+3 = 17$. С учетом формулы (12) разрядность промежуточных данных в КИХ-части $n_у = n_с + 4 = 17+4 = 21$. Результирующее VHDL-описание фильтра представлено ниже.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
entity LPF_HB_LAB2 is
     port(
          CLK : in STD_LOGIC;
```

```
          RST : in STD_LOGIC;
          X : in STD_LOGIC_VECTOR(11 downto 0);
          Y : out STD_LOGIC_VECTOR(11 downto 0)
          );
end LPF_HB_LAB2;

architecture synt of LPF_HB_LAB2 is
     constant max:signed(3 downto 0):="0111";
     constant min:signed(3 downto 0):="1100";
     signal xi,xi_1,xi_2,xi_3,xi_4,xi_5:signed(16 downto 0);
     signal qi,qi_1,qi_2,qi_3,mb,qpx,mbp:signed(16 downto 0);
     signal yi,yi_1,yi_2,yi_3,yi_4:signed(16 downto 0);
     signal y01,y22,y34,y012,y34_1:signed(20 downto 0);
     signal ys:signed(11 downto 0);
begin
     IIR:process(CLK,RST)
     begin
          if RST = '1' then
               xi<=(others=>'0');       xi_1<=(others=>'0');
               xi_2<=(others=>'0');     xi_3<=(others=>'0');
               xi_4<=(others=>'0');     xi_5<=(others=>'0');
               yi<=(others=>'0');       qi<=(others=>'0');
               qi_1<=(others=>'0');     qi_2<=(others=>'0');
               qi_3<=(others=>'0');     qpx<=(others=>'0');
               mb<=(others=>'0');       mbp<=(others=>'0');
          elsif CLK='1' and CLK'event then
               xi<= RESIZE(signed(X&"000"),17);
               xi_1<= xi;
               xi_2<= xi_1;
               xi_3<= xi_2;
               xi_4<= xi_3;
               xi_5<= xi_4;
               qpx <= xi + qi_1;
               mb  <= shift_right(qpx,1) - shift_right(qpx,4);
               qi  <= xi_2 - mb;
               qi_1<= qi;
               qi_2<= qi_1;
               qi_3<= qi_2;
               mbp <= qi_3 + mb;
               yi<= mbp + xi_5;
          end if;
     end process;

     FIR:process(CLK,RST)
       variable yt:signed(20 downto 0);
     begin
          if RST = '1' then
               yi_1<=(others=>'0');     yi_2<=(others=>'0');
               yi_3<=(others=>'0');     yi_4<=(others=>'0');
               y01 <=(others=>'0');     y012 <=(others=>'0');
               y22 <=(others=>'0');     y34 <=(others=>'0');
               y34_1<=(others=>'0');    ys   <=(others=>'0');
          elsif CLK='1' and CLK'event then
               yi_1<= yi;
               yi_2<= yi_1;
               yi_3<= yi_2;
               yi_4<= yi_3;
               y01 <= yi + resize((yi_1 &"00"),21) ;
```

```
                y22 <= resize((yi_2 &"00"),21) + (yi_2 & "0");
                y34 <= yi_4 + resize((yi_3 & "00"),21) ;
                y34_1<= y34;
                y012 <= y01 + y22;
                yt:= y012 + y34_1;
                ys<=yt(19 downto 8);
          end if;
     end process;
           Y<= std_logic_vector(ys);
end synt;
```

Результаты испытания на испытательном стенде, таком как на рис.10, показаны на рис. 18.

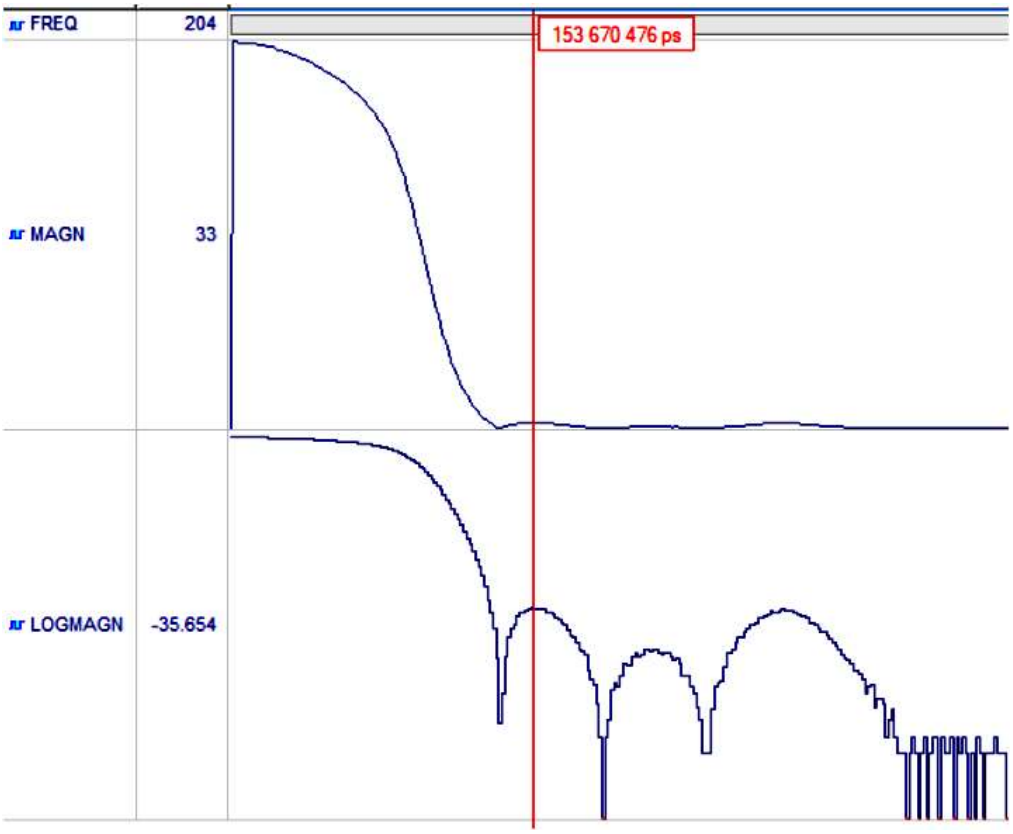


Рис. 18. АЧХ и логарифмическая АЧХ фильтра на рис. 17

Из полученных графиков видно, что фильтр имеет уровень подавления 35,6 дБ, частоту среза 0,112, частоту начала полосы подавления 0,175.

При конфигурировании фильтра в ПЛИС Xilinx Spartan-6 получены следующие результаты.

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	382	18,224	2%
Number used as Flip Flops	382		
Number used as Latches	0		
Number used as Latch-thrus	0		
Number used as AND/OR logics	0		
Number of Slice LUTs	263	9,112	2%
Number used as logic	173	9,112	1%
Number using O6 output only	169		
Number using O5 output only	0		
Number using O5 and O6	4		
Number used as ROM	0		
Number used as Memory	1	2,176	1%
Number used as Dual Port RAM	0		
Number used as Single Port RAM	0		
Number used as Shift Register	1		
Number using O6 output only	1		
Number using O5 output only	0		
Number using O5 and O6	0		
Number used exclusively as route-thrus	89		
Number with same-slice register load	88		
Number with same-slice carry load	1		
Number with other load	0		
Number of occupied Slices	77	2,278	3%
Nummber of MUXCYs used	192	4,556	4%
Number of LUT Flip Flop pairs used	286		
Number with an unused Flip Flop	9	286	3%
Number with an unused LUT	23	286	8%
Number of fully used LUT-FF pairs	254	286	88%

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
ts_clk = PERIOD TIMEGRP "CLK" 2.8 ns HIGH 50%	SETUP	0.248ns	2.552ns	0	0
	HOLD	0.388ns		0	0
	MINPERIOD	0.134ns	2.666ns	0	0

Итак, аппаратные расходы синтезированного фильтра составляют 77 CLB slices, в том числе 382 триггера и 263 логических таблицы.

Тактовая частота фильтра достигает f_C = 1 / 2.666 = 375 МГц. Это почти втрое больше, чем для фильтра, который использует блок умножения (см. Лабораторную работу 1). Таким образом, методика использования специализированных блоков умножения не только уменьшает аппаратные расходы (блоки умножения DSP48, каждый из которых эквивалентен 20 сумматорам), но и существенно увеличивает быстродействие фильтра.

Литература

1. Сергиенко А.М. VHDL для проектирования вычислительных устройств. Киев: ЧП "Корнейчук", ТИД ДС, 2003 — 208 с.

2. Сергієнко А.М. Генератор рекурсивних фільтрів без блоків множення. 2014. [електронний ресурс]
http://kanyevsky.kpi.ua/GEN_MODUL/APgen/APMF_help_ukr.php

3. Сергієнко А. М., Сергієнко А.А. Методика проектування цифрових фільтрів з застосуванням VHDL. //Праці 3 міжнародної конференції InfoCom'2016, 1 – 2 грудня 2016 р. -К.:НТУУ "КПІ", ВПІ "Політехніка". – 2016. -С. 56-57. [електронний ресурс]
https://iconfs.net/w.infocom2016/metodyka-proektuvannya-tsyfrovykh-filtriv-z-zastosuvannyam-vhdl

4. Сергієнко А.М., Виноградов Ю.М., Лесик Т.М. Цифрова обробка сигналів. Комп'ютерний практикум мовою VHDL. – Київ. – 2012. – 106 с. [електронний ресурс]
http://kanyevsky.kpi.ua/wp-content/uploads/2017/11/DSP_LabS.pdf

5. Schlichthärle D. Digital Filters: Basics and Design. – Springer. Berlin Heidelberg, –2011. – 527 p.