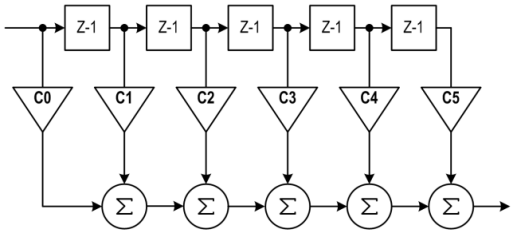
Стандартные приемы оптимизации арифметики FIR фильтров

В общем виде расчет свертки (FIR фильтр) имеет вид:



Блоки задержки Z-1 – образуют буфер FIFO, содержащий входную последовательность семплов, а треугольники задают усиление (умножая значения на дробную величину). Соот-но кол-во треугольников обозначает кол-во умножений.

Для формирования выходного семпла входные отсчеты данных перемножаются на ряд коэффициентов C0..C5 и суммируются.

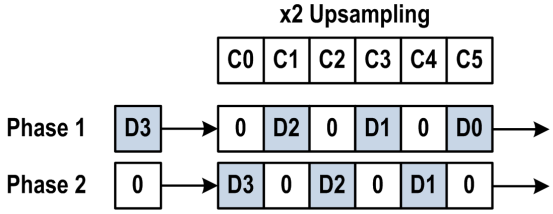
Порядок такого фильтра на единицу меньше кол-ва отводов (коэффициентов), и для данного примера равен 5.

Полифазность при апсемплинге (интерполяции)

Апсемплинг осуществляется прореживанием исходных семплов нулевыми, с последующим пропусканием полученной последовательности через FIR фильтр. Кратность увеличения кол-ва семплов за счет добавления новых определяет кратность интерполяции.

Т.к. добавленные семплы нулевые, то они не влияют на результат суммы свертки и их умножение можно не выполнять.

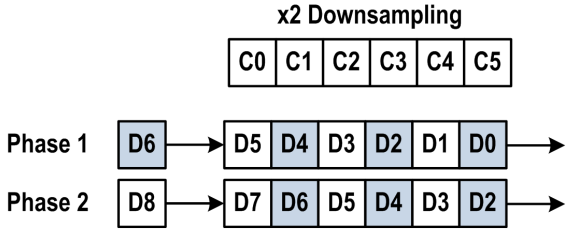
В результате процедура обсчета разбивается на фазы: на каждом сдвиге данных в FIFO необходимо перемножить только часть коэффициентов, расположенных напротив исходных отсчетов в FIFO.



Полифазность при даунсемплинге (децимации)

Даунсемплинг осуществляется фильтрацией исходного потока данных, с последующим отбрасыванием части семплов. А значит семплы, которые выбрасываются, можно не обсчитывать.

Таким образом децимация вырождается в однократное перемножение ряда данных на ряд коэффициентов после многократного сдвига. Кол-во сдвигов перед обсчетом определяет кратность децимации.

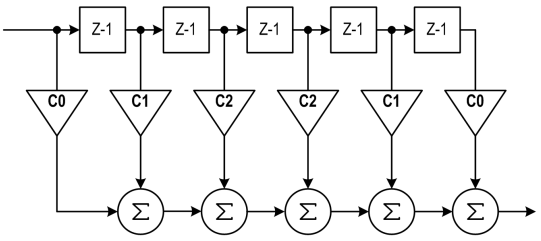


Оптимизация фазолинейных фильтров

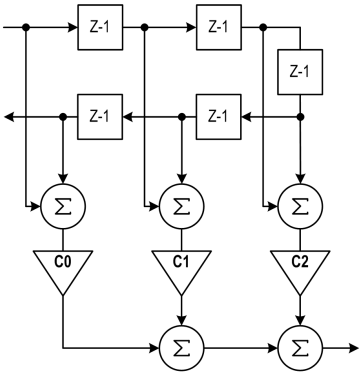
Данные фильтры отличаются симметричной импульсной хар-кой (свойство линейной фазы). Это означает, что половина коэффициентов относительно центра импульсной хар-ки зеркально повторяет вторую половину.

За счет этого свойства два умножения на одинаковые коэффициенты можно заменить одним умножением на сумму двух соотв. отсчетов.

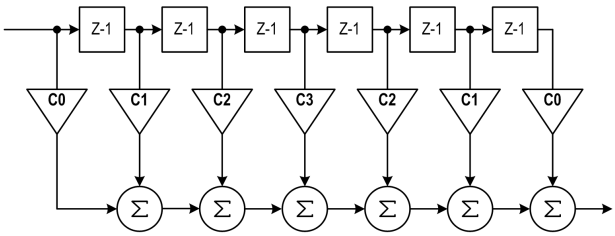
Для фильтра с четным числом коэф-тов каноническая схема имеет вид:



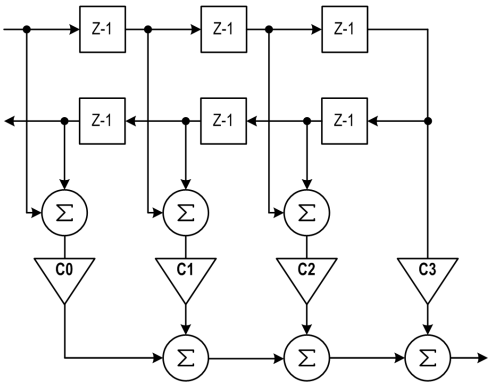
Его оптимизированная схема:



Каноническая схема фильтра с нечетным числом коэффициентов:



Его оптимизированная схема:

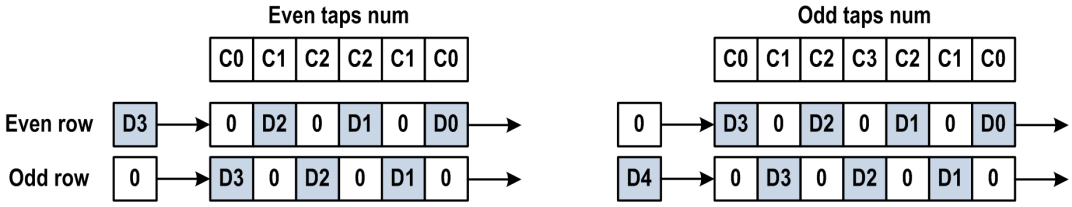


Таким образом кол-во умножений сокращается вдвое (или почти вдвое).

Оптимизация коэффициентов

Выбор порядка фильтра при интерполяции

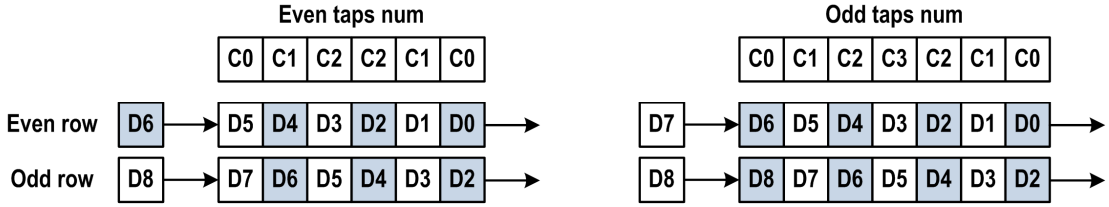
На примере x2 интерполяции ниже можно видеть, что для того чтобы использовать свойство симметрии коэффициентов необходимо чтобы фильтр имел четный порядок (нечетное число отводов)



Выбор порядка фильтра при децимации

При децимации прореживание нулями отсутствует и для применения свойства симметрии можно использовать фильтр с любым числом отводов. Но если необходимо выполнять считывание двух семплов за один такт, то удобно разбить семплы на два ряда: четный и нечетный, и писать каждый ряд в отдельный буфер FIFO.

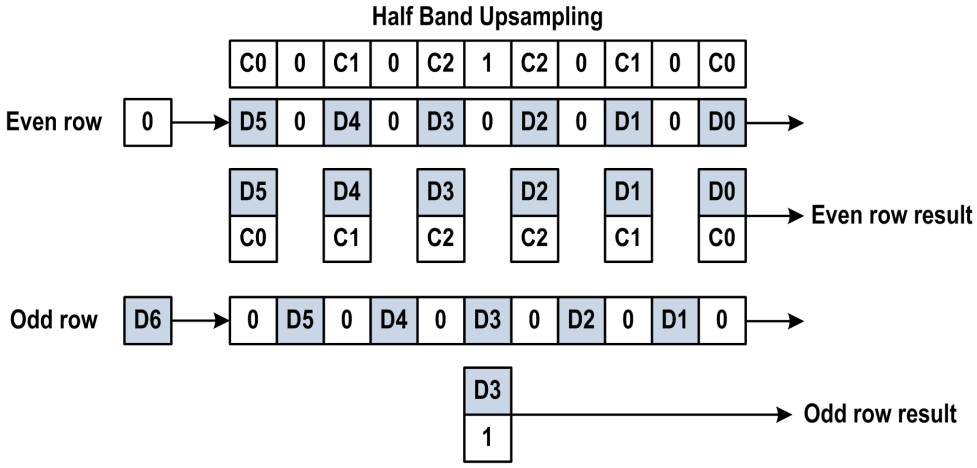
Тогда при использовании фильтра с четным числом отводов (нечетного порядка) данные необходимые для суммированием перед умножением будут всегда в разных FIFO буферах и их можно считать параллельно, за один такт, без дополнительных затрат ресурсов на организацию блоков памяти с двухканальными портами.



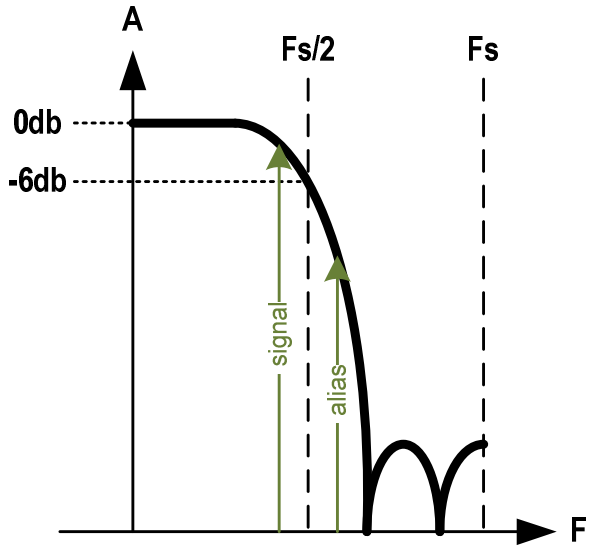
Полуполосные (half-band) фильтры

Получили наибольшее распространение как в апсемплерах, так и в даунсемлперах за счет того, что у них половина коэффициентов равна нулю: нечетный ряд такого фильтра содержит только один единичный коэффициент. Поэтому, после каждого второго сдвига вместо умножения на свертку можно выполнять копирование одного семпла.

Это означает, что на выходе такого фильтра исходные отсчеты остаются неизменными. Кроме того, полуполосный фильтр всегда фазолинейный и четного порядка, что позволяет также сократить кол-во умножений за счет симметрии не только при децимации, но и при интерполяции, снижая таким образом суммарно кол-во умножений вчетверо:



Недостаток полуполосного ФНЧ в том, что его частотная характеристика всегда пересекает половину частоты семплирования (частоту Найквиста) на уровне -6дБ от уровня полосы пропускания, что делает невозможным полную фильтрацию зеркальных составляющих, попадающих в переходную полосу.

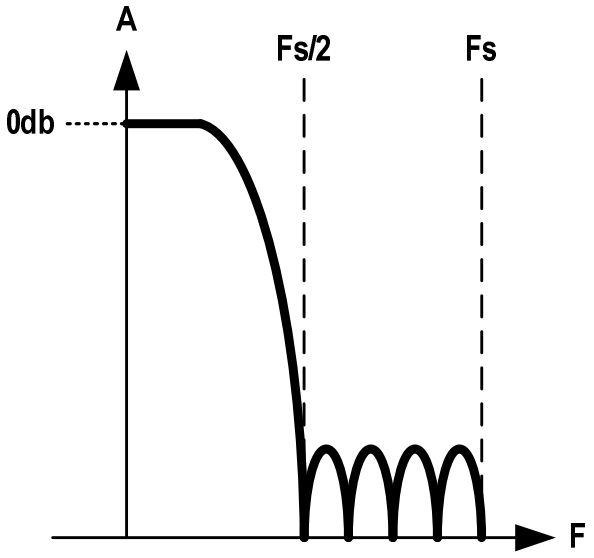


В значительной мере эта проблема решается сужением переходной полосы, что однако увеличивает ресурсоемкость фильтра и удлиняет его импульсную хар-ку.

Помимо свойств четности и полуполосности ресурсоемкость можно сократить разбивая процедуру апсемплинга или даунсемплинга на x2 каскады.

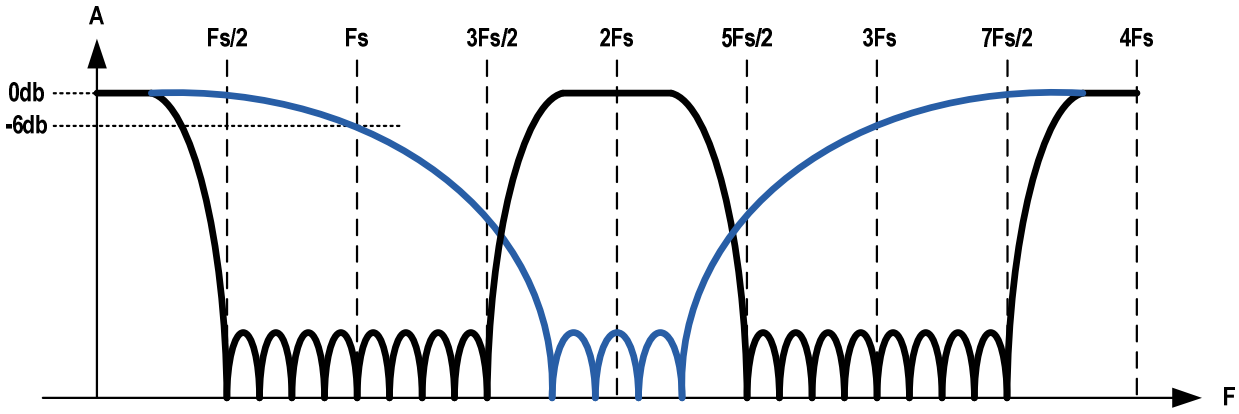
Тогда большая длина потребуется только от одной ступени: первой для интерполятора и последней для дециматора. Для остальных каскадов относительная полоса пропускания получается кратно уже, поэтому переходную полосу можно расширить, укоротив фильтр.

В случае, если необходимо обеспечить ослабление на частоте Найквиста больше 6дБ, то для первой ступени не получится использовать фильтр полуполосного типа. В таком случае придется применить более длинный FIR фильтр без нулевого ряда коэффициентов:

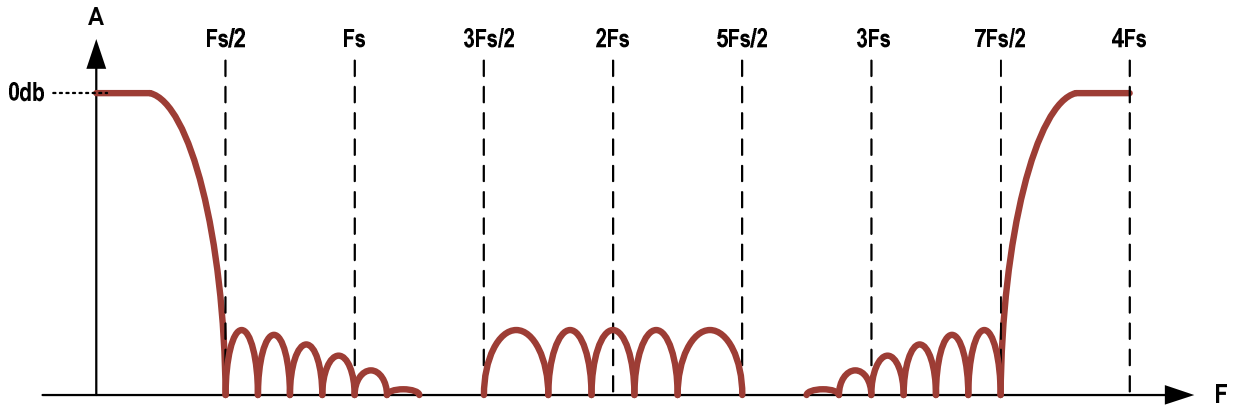


Но т.к. после первой ступени сигнал получается уже отфильтрован, то следующему x2 каскаду достаточно отфильтровать только отражение предыдущего. Это означает, что все последующие ступени x2 апсемплинга можно выполнить максимально короткими half-band фильтрами с широкой переходной полосой.

Пример x4 апсемплинга с первой длинной x2 ступенью equiripple (черный) и второй x2 half-band (синий) с исходной частотой семплирования Fs.



Результирующая частотная характеристика будет иметь вид:



Кроме того, в многокаскадной схеме импульсную характеристику всего фильтра определяет каскад с самой низкой частотой среза, т.е. – самый длинный каскад.

Это значит, что если в качестве первого x2 интерполятора использовать минимально-фазовый тип фильтра, то при добавлении фазолинейных каскадов импульсная хар-ка на выходе фильтра останется минимально-фазовой.

Поэтому для реализации фильтров с разным типом ФЧХ достаточно сделать разные варианты основного длинного x2 каскада интерполятора.

Оптимизация логики поочередной обаработкой данных

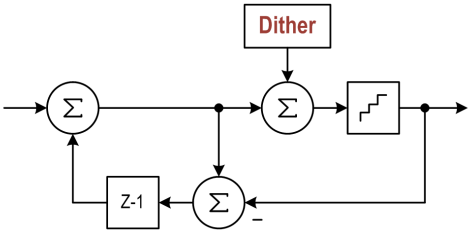
Т.к. арифметика FIR фильтров любого типа схожа, то при построении многокаскадной структуры выгодно использовать один общий блок ALU и общий (поделенный на сектора) буфер памяти.

Конечный автомат для запуска и поочередной обработки данных получается достаточно сложный, но экономия на арифметике делает такую структуру выгоднее распараллеленной (когда все каскады обсчитываются одновременно).

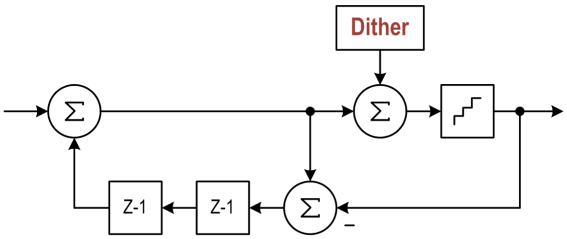
Кроме того, т.к. аудио-сигналы содержат два и более каналов, то выгодно и обработку каналов также выполнить поочередно.

Такая последовательная обработка данных «левый/правый» позволяет сэкономить не только на общем блоке FIR апсемплера, но и на любой другой логике.

Рассмотрим на примере шейпера 1-го порядка, каноничная форма которого имеет вид:

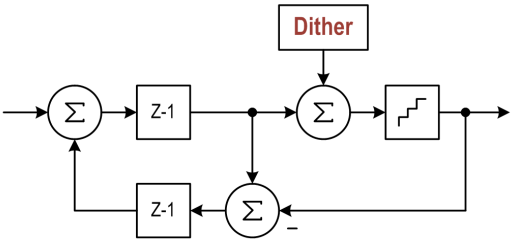


Для того, чтобы данной схемой обрабатывать сразу два канала, достаточно удвоить кол-во элементов задержки и чередовать семплы данных на входе «левый/правый», получая аналогичную последовательность семплов на выходе:

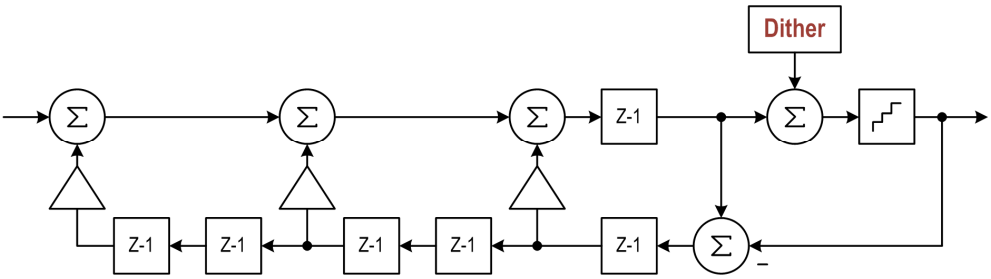


Таким образом можно последовательно обработать любое число каналов, создав соответствующее кол-во элементов задержки. А при достаточно большом кол-ве элементов блок задержки вырождается в FIFO ОЗУ, которое может быть синтезировано на блочной или распределенной памяти, снижая ресурсы логических ячеек.

При необходимости повысить быстродействие логики элементы задержки можно перераспределить:

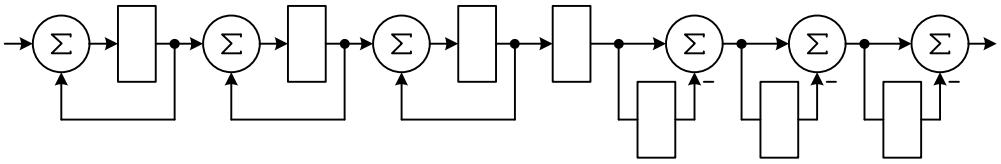


Или использовать комбинированный вариант. Пример стерео-шейпера 3-го порядка, реализованного в DF3E проекте:

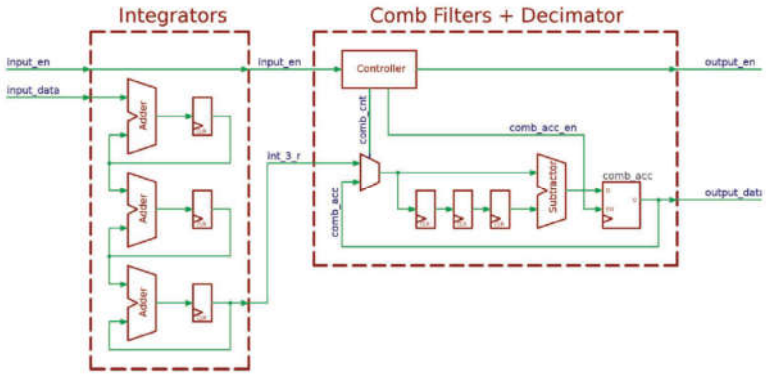


Данный подход можно обобщить на узлы арифметики других модулей: модуляторы, апсемплеры, и т.п. Применение поочередной обработки для двух каналов позволяет радикально (более чем в полтора раза) снизить ресурсы логики практически для любой схемы.

Рассмотрим типовую схему трехкаскадного CIC фильтра- дециматора:

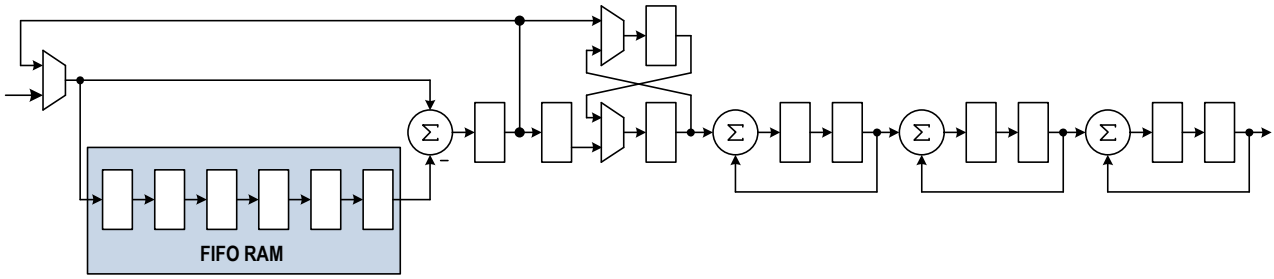


Его экономичную реализацию можно выполнить следующим способом [1]:

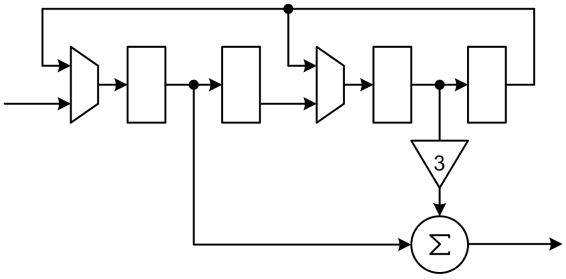


Экономия достигается за счет поочередного вычитания в каскадах дифференциатора. В результате три сумматора заменяются одним. *Строго говоря, аналогичный прием можно использовать и для секции интеграторов, но т.к. они обычно работают на высоких частотах, то может не хватить частоты системного клока.*

Для данной схемы так же возможно организовать последовательную обработку каналов аудио-данных, удвоив количество элементов задержки в блоке дифференциатора. По такому принципу построен 3-х секционный CIC интерполятор проекта DF3E:



Аналогичным образом (с чередованием каналов) в проекте DF3E модернизирован x2 FIR интерполятор, со сверткой эквивалентной x2 CIC3 апсемплеру (0.25/0.75/0.75/0.25), примененному в проекте DF2:



Литература

1. Simone Impagnatiello “Master theses CIC filter design with HLS”.

Фисько Алексей.
Орша 02/2025