DF2 DIY Digital Filter (alpha 1.2 10/2023)

Опыт предыдущего проекта DF1 показал, что ЦФ на FPGA интересен любителям не как готовый проект, а как «заготовка» для применения в кастомном дизайне. Поэтому новый проект DF2 является развитием предыдущего проекта, но выполнен на подобие фреймворка, в котором опции задаются условной компиляцией.

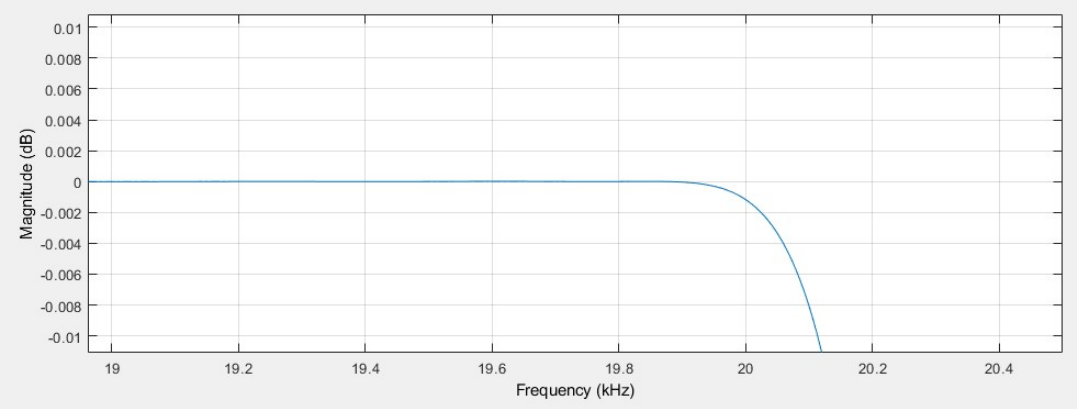
Как и в предыдущем проекте в DF2 применена оптимизация фазолинейных фильтров четного порядка: маки дополнены сумматорами для предварительного суммирования семплов, соответствующих одинаковым коэф-там. Реализуется такая арифметика за счет сложной схемы адресации FIFO буферов в модуле **ADR_GEN**, которую упростить особо не получилось. Но за счет экономии на умножителях такая схема все равно получается выгоднее, особенно на ПЛИС без выделенных блоков умножения.

Что нового в DF2:

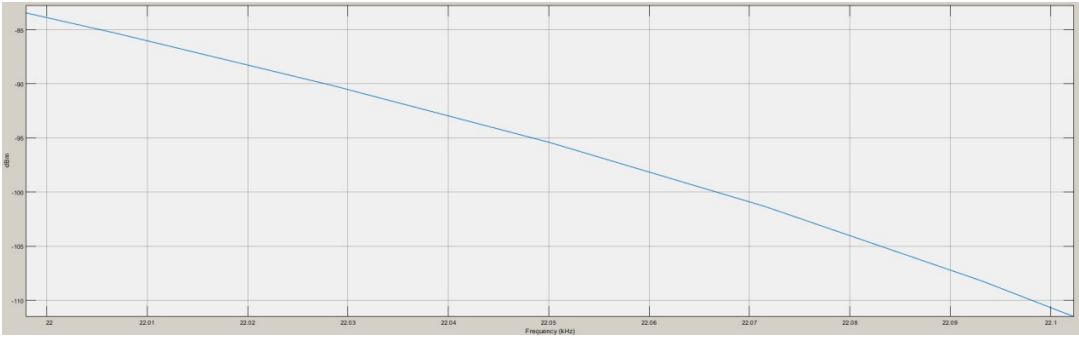
1. Значительно упрощена структура проекта. Ядро фильтра **DF2_FIR_CORE**, помимо конфигурационных входов, теперь имеет на входе и выходе только параллельные шины данных. Дополнительных синхронизирующих сигналов не требуется.
2. Аттенюатор из входного модуля **SAI_INPUT** исключен, теперь его ф-ю выполняет мак, на что расходуется дополнительно пара тактов в процессе обсчета фильтров. А сам модуль теперь распознает длину фрейма автоматически. Достаточно задать нужный формат I2S интерфейса.
3. Умножение теперь выполняется за два такта, что позволило почти в полтора раза поднять производительность. Для первого интерполятора выделено FIFO вдвое больше: на максимальную длину до 512 тапов (без учета допустимого порядка фильтра).
4. Добавлена поддержка коэф-тов минимальнофазовых фильтров. А в самом ЦФ сделано 4 режима фильтрации: **normal**, **sharp**, **short**, **slow**.
5. Добавлен **SPI** интерфейс для записи в ОЗУ произвольного значения аттенюации, а так же кастомных коэффициентов, которые можно залить вместо фильтра **slow**.
6. Оптимизирован МАК. Теперь у него абстрактное описание, что позволяет параметризировать не только разрядность данных, но и разрядность коэф-тов.
7. Упрощено описание ROM коэф-тов. Теперь они вынесены в отдельный файл с табличным представлением.
8. Округление данных теперь выполняется с шейпингом, с возможностью задания порядка от 1 до 3. И с возможностью задания произвольной разрядности округления от приблизительно 4-х до 24-х бит. А перед шейпером и дизером добавлена отдельная аттенюация для предотвращения переполнения из-за добавления шума.
9. Добавлены опциональные апсемплеры, позволяющие нарастить кратность оверсемплинга вплоть до частоты тактирования.
10. Модули вывода написаны в двух версиях: **SSAI_OUTPUT** – упрощенный модуль для максимальной экономии ресурсов с непрерывным битклоком, для вывода на аудио-ЦАПы. И **USAI_OUTPUT** – универсальный модуль вывода данных, поддерживающий функционал как аудио, так и SPI ЦАП-ов, в том числе сдвоенных, с заголовками перед данными. Второй модуль аналогичен модулю вывода DF1, но для упрощения выполнен с конфигурацией условной компиляцией и без опции конвейеризации.

Частотные характеристики DF2 в режиме Sharp

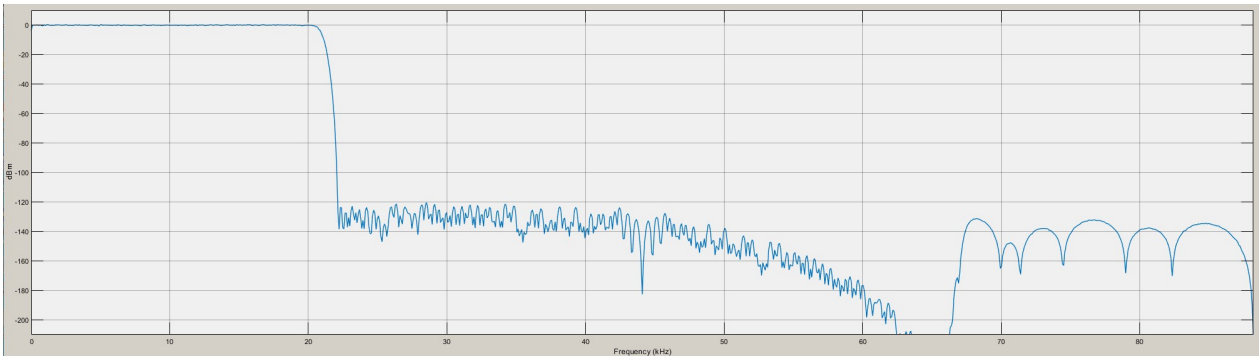
Sharp PassBand



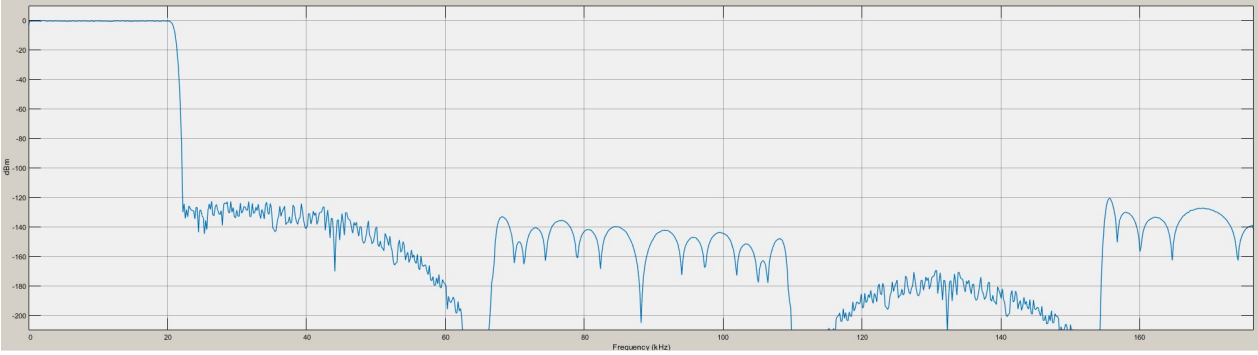
Sharp Nyquist Frequency



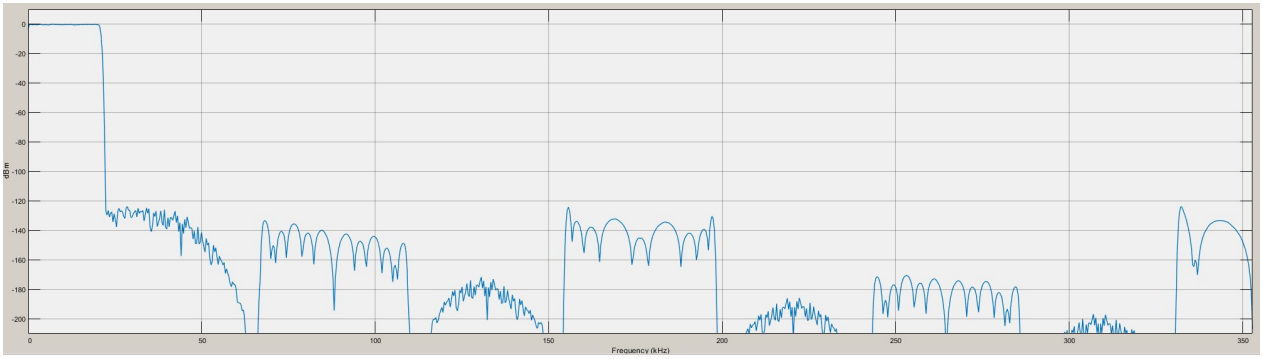
Sharp x4



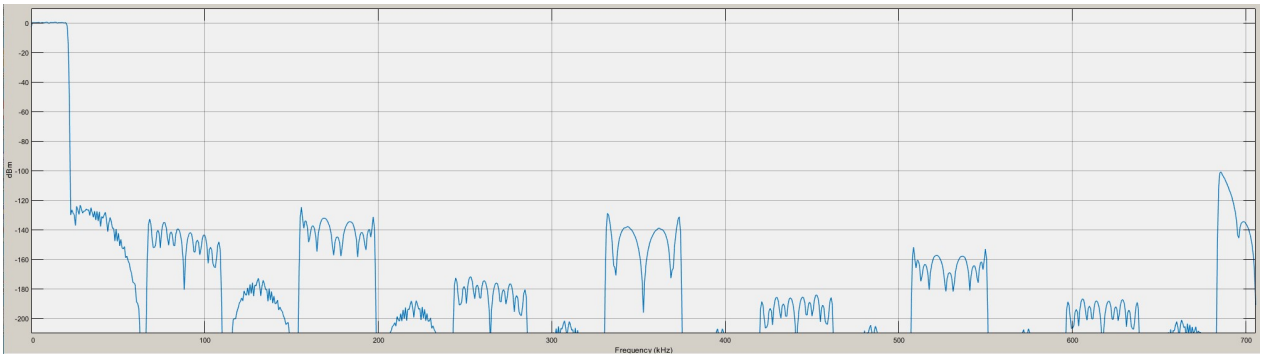
Sharp x8



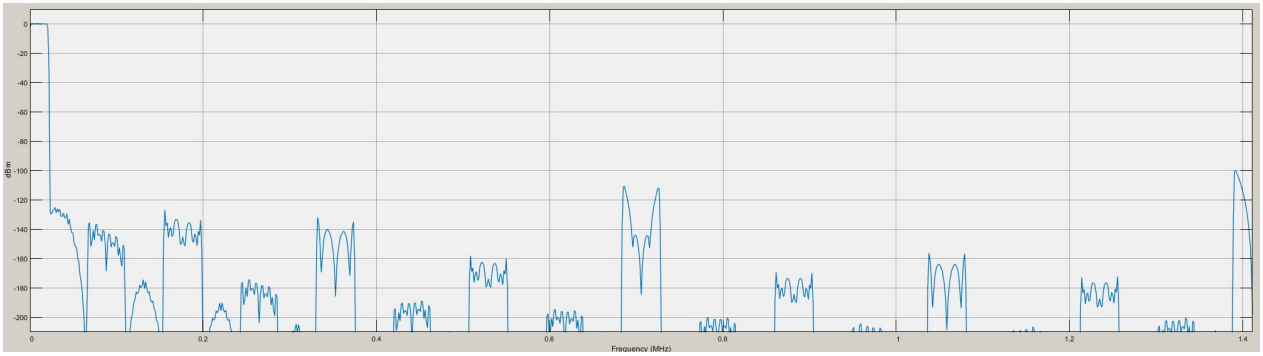
Sharp x16



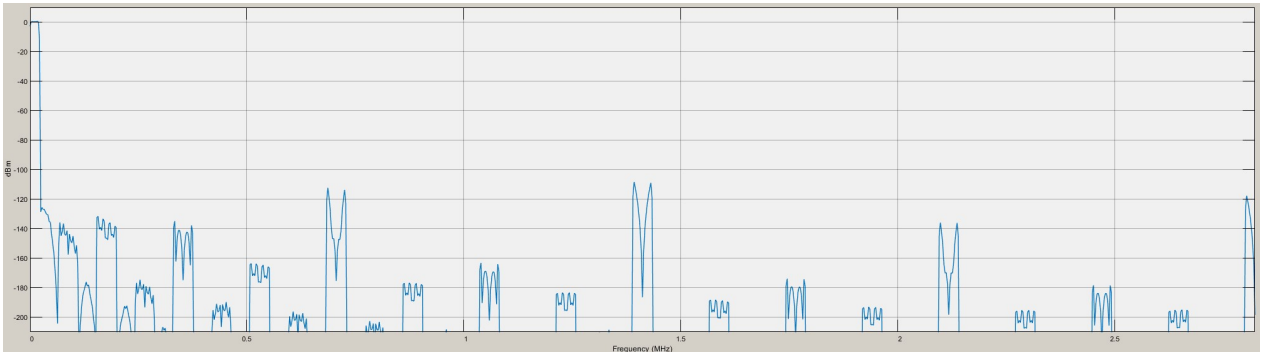
Sharp x32



Sharp x64



Sharp x128

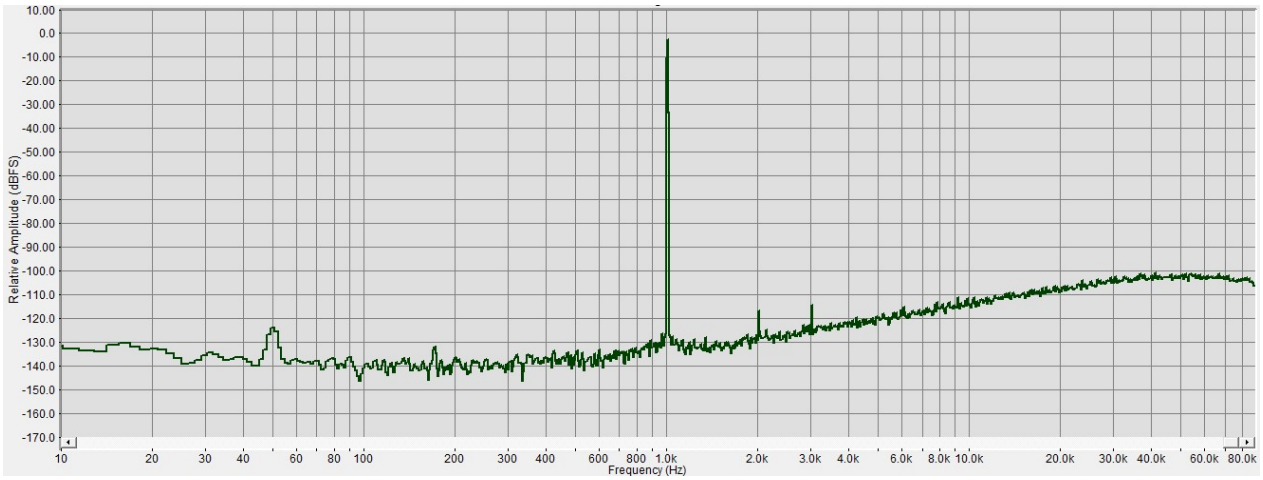


Неравномерность в полосе пропускания в режиме **sharp** с x32 оверсемплингом (без учета дополнительных интерполяторов) менее +/-0,00002дБ.

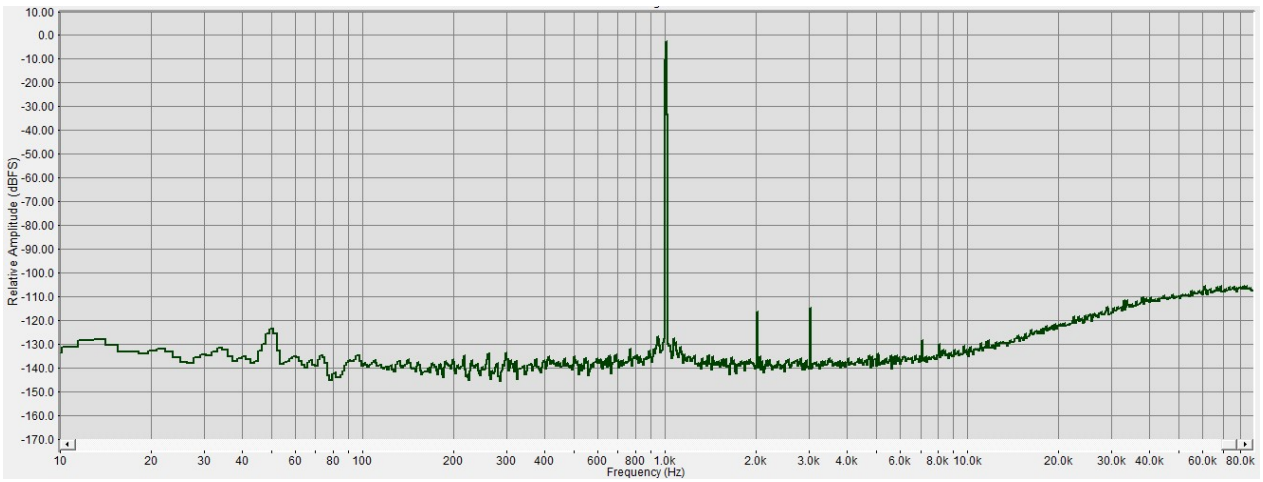
Задержка сигнала в режиме x16 оверсемплинга для фильтра **sharp**: около 1,8мс, для фильтра **short** – около 0,3мс.

Замеры шейпера на параллельном ЦАП в режиме 8 бит с оверсемплингом x16 (705,6кГц).

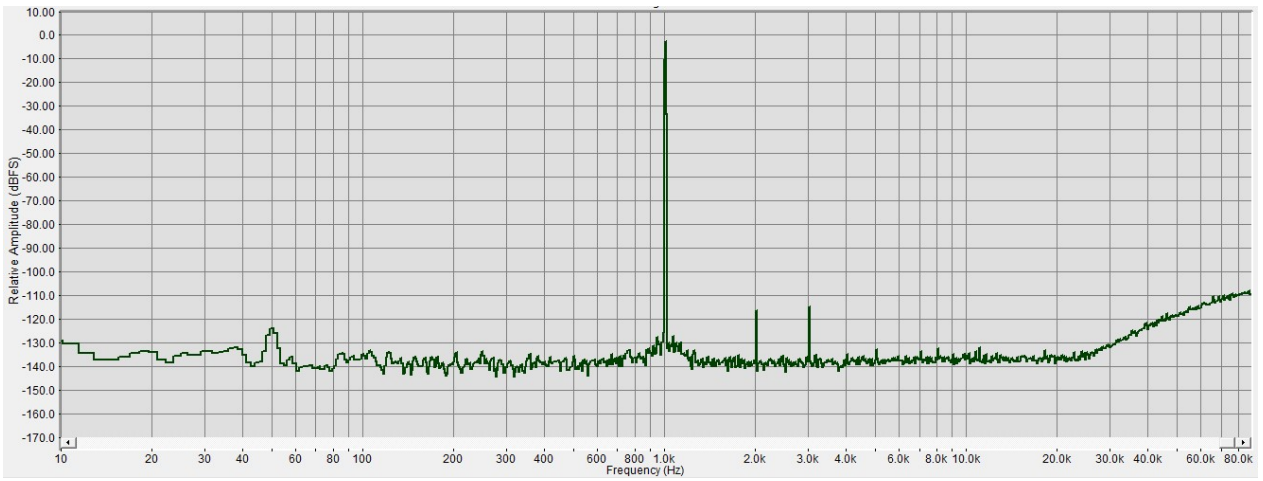
1-ый порядок



2-ой порядок



3-ий порядок



Настройка проекта в IDE.

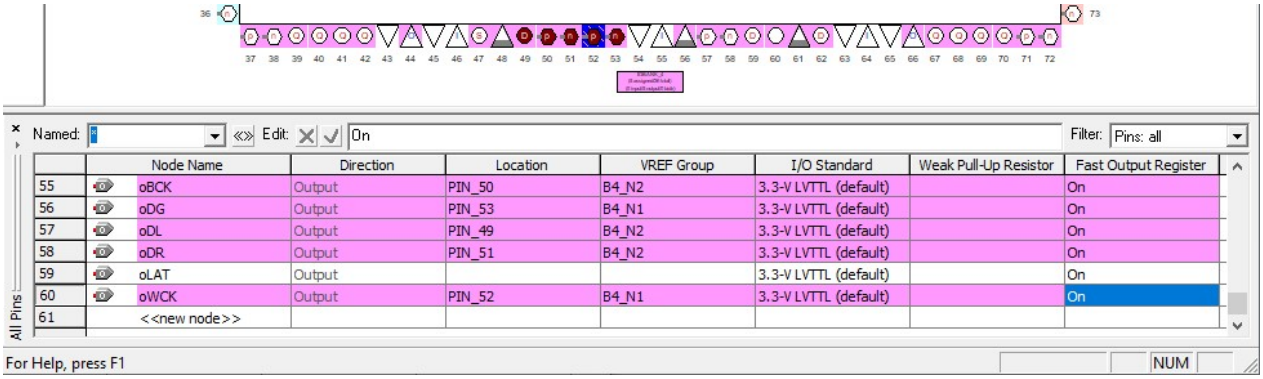
Для FPGA с высоким грейдом какой-то тонкой настройки не требуется. А вот для более медленных ПЛИС (FPGA Cyclone с грейдом **8**, или FPGA Lattice с грейдом **Z**), во избежание больших разбежек фронтов на выходах, необходимо настроить трассировщик и задать констрейнты таймингов выходных сигналов.

Для тактового входа **iCLK** необходимо выбирать порт с опцией глобального клока.

Для плис Lattice необходимо использовать компилятор **Synplify Pro** (выбирается в св-вах проекта, или при создании проекта).

Настройка констрейнтов и оптимизации маппингом.

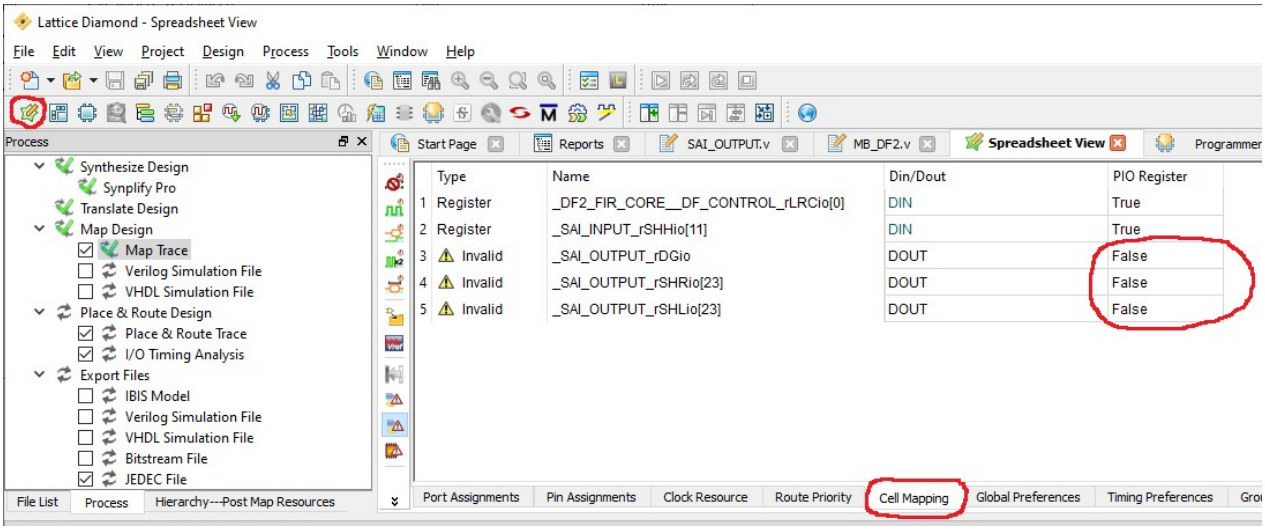
1. Необходимо сконфигурировать триггеры на выходных портах.
   Для FPGA **Cyclone** они по умолчанию отключены, поэтому нужно их включить.
   Для этого в планировщике пинов включить столбец **Fast Output Register** и для выходных портов данных задать значение **On**.



	Node Name	Direction	Location	VREF Group	I/O Standard	Weak Pull-Up Resistor	Fast Output Register
55	oBCK	Output	PIN_50	B4_N2	3.3-V LVTTL (default)		On
56	oDG	Output	PIN_53	B4_N1	3.3-V LVTTL (default)		On
57	oDL	Output	PIN_49	B4_N2	3.3-V LVTTL (default)		On
58	oDR	Output	PIN_51	B4_N2	3.3-V LVTTL (default)		On
59	oLAT	Output			3.3-V LVTTL (default)		On
60	oWCK	Output	PIN_52	B4_N1	3.3-V LVTTL (default)		On
61	<<new node>>						

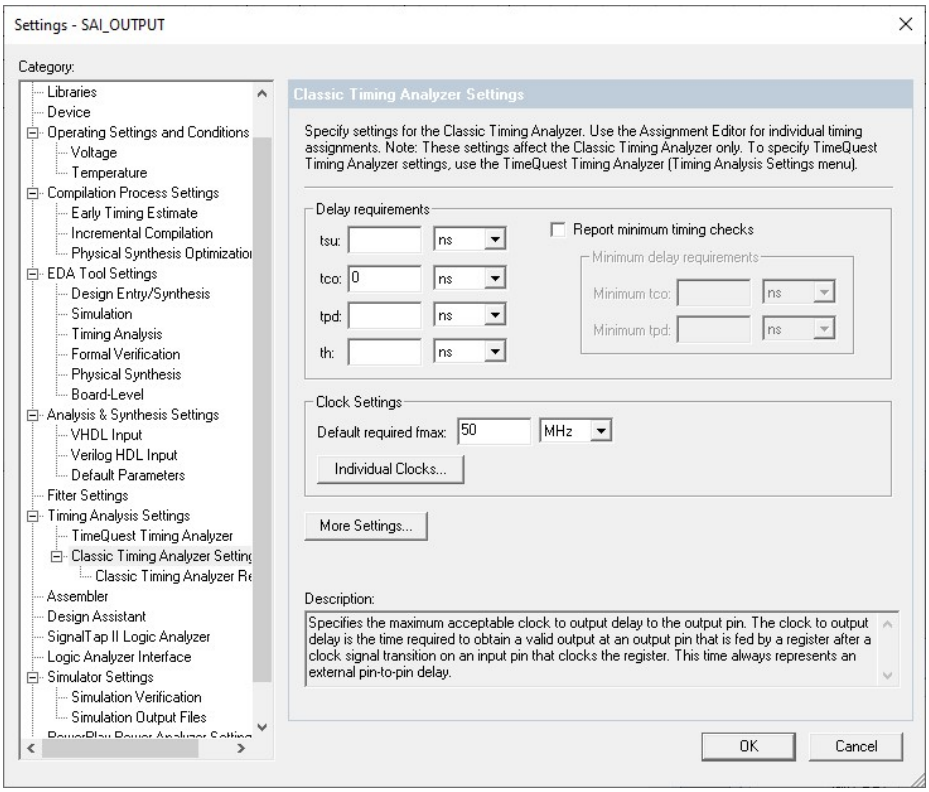
Для FPGA **MachXO2** в IDE **Diamond** эти триггеры нужно наоборот выключить.

Сначала необходимо собрать логику проекта – выполнить процедуры до **Map Trace** включительно. Затем открываем **Spreadsheet View**, переходим на вкладку **Cell Mapping** и для выходных портов задаем значения **False**:

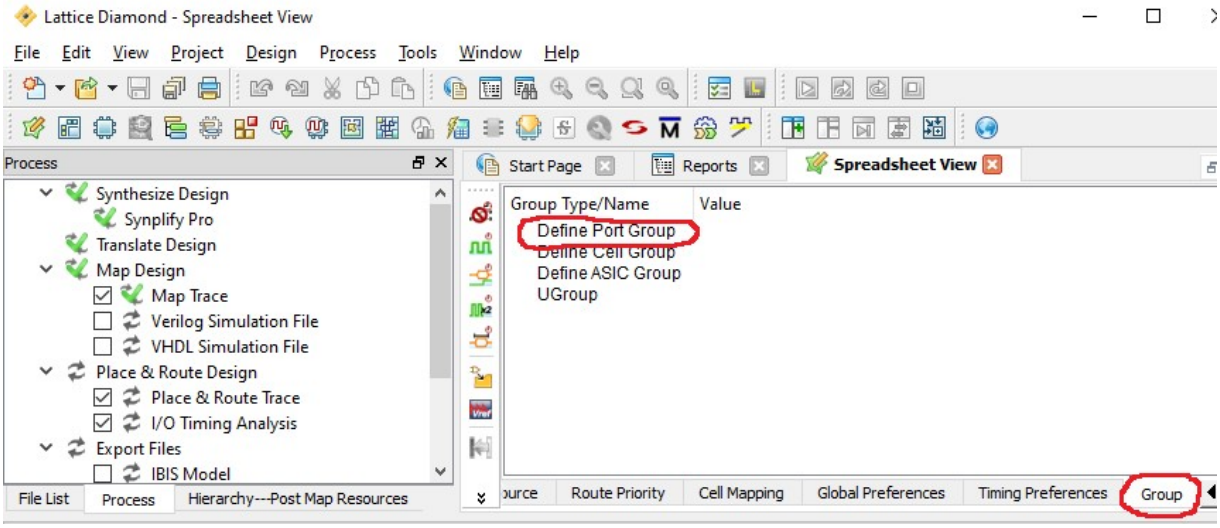


	Type	Name	Din/Dout	PIO Register
1	Register	_DF2_FIR_CORE__DF_CONTROL_rLRCio[0]	DIN	True
2	Register	_SAI_INPUT_rSHHio[11]	DIN	True
3	⚠ Invalid	_SAI_OUTPUT_rDGio	DOUT	False
4	⚠ Invalid	_SAI_OUTPUT_rSHRio[23]	DOUT	False
5	⚠ Invalid	_SAI_OUTPUT_rSHLio[23]	DOUT	False

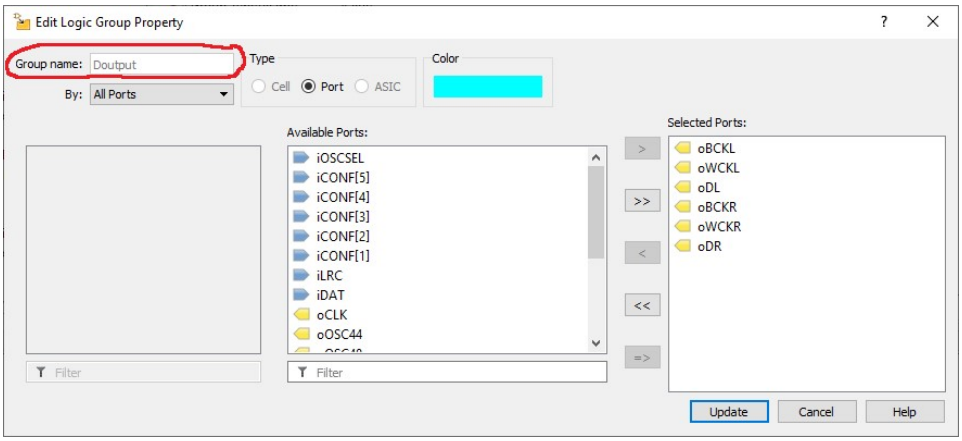
2. Через констрейнты таймингов необходимо задать желаемую тактовую частоту и минимальную задержку клока относительно выходов.
   В **Quartus** для этого самый простой способ – через **Classic Timing Analyser Settings** в свойствах проекта задаем опции как на скрине:



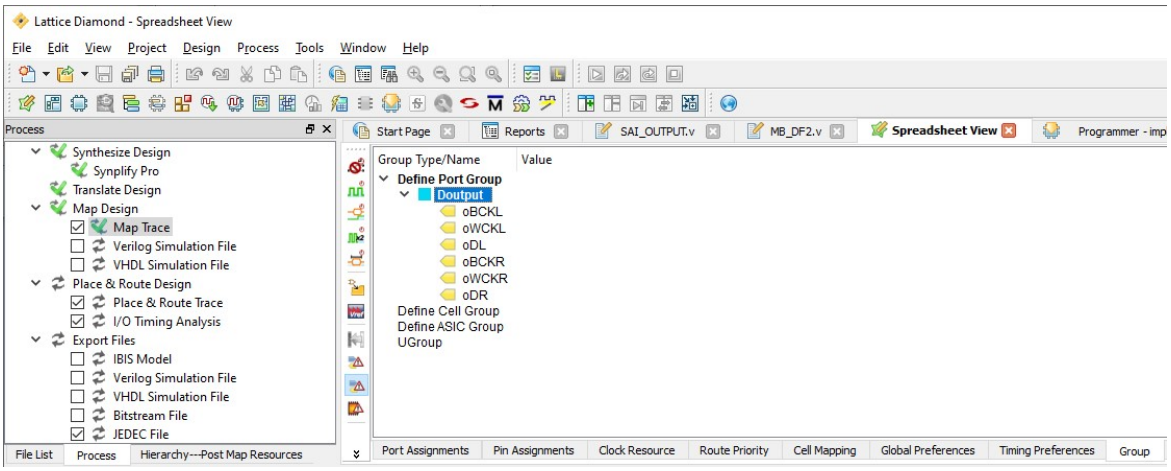
В **Diamond** опять сложнее. Общей опции для всех пинов нету, поэтому сперва нужно создать соответствующую группу из нужных пинов. Для этого в **Spreadsheet View** переходим на вкладку Group и кликаем на вкладку **Define Port Group**:



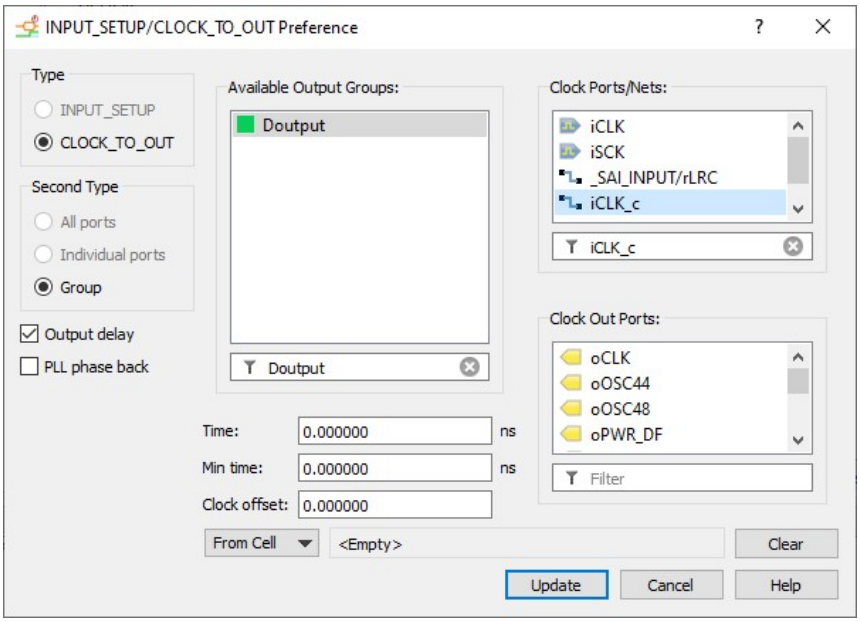
Выбираем нужные выходные порты и в окне **Group name** задаем имя группы:



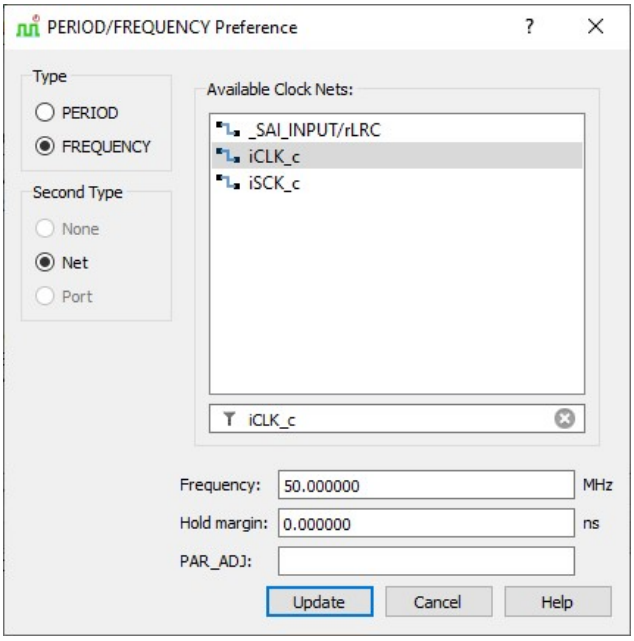
Должно получиться примерно так:



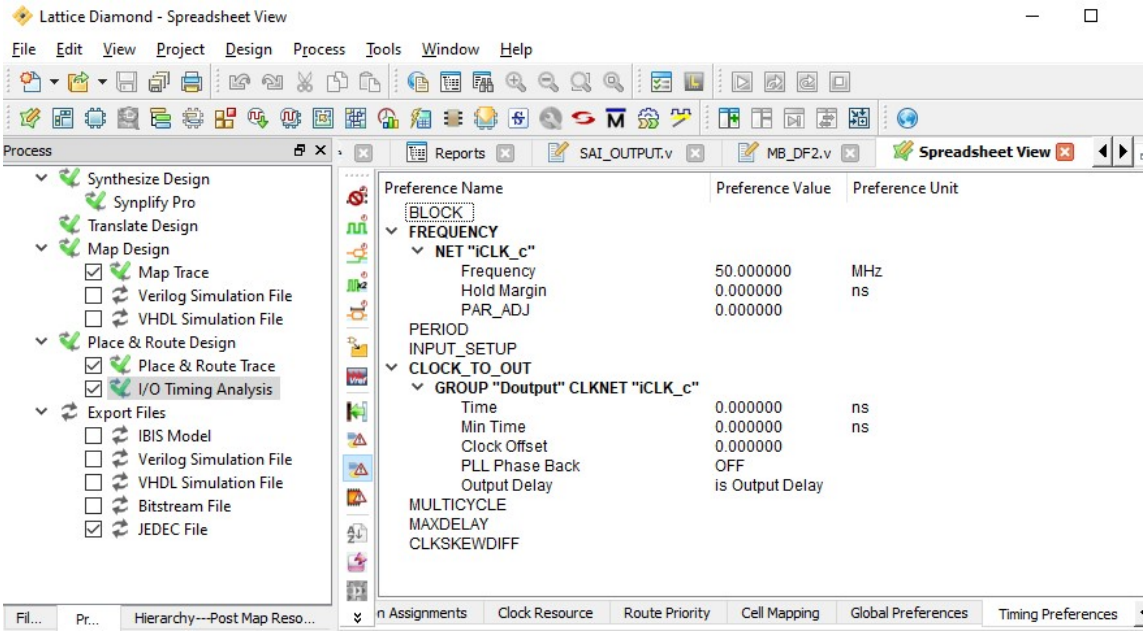
Затем переходим на вкладку **Timing Preferences** и кликаем на опцию **CLOCK_TO_OUT**.



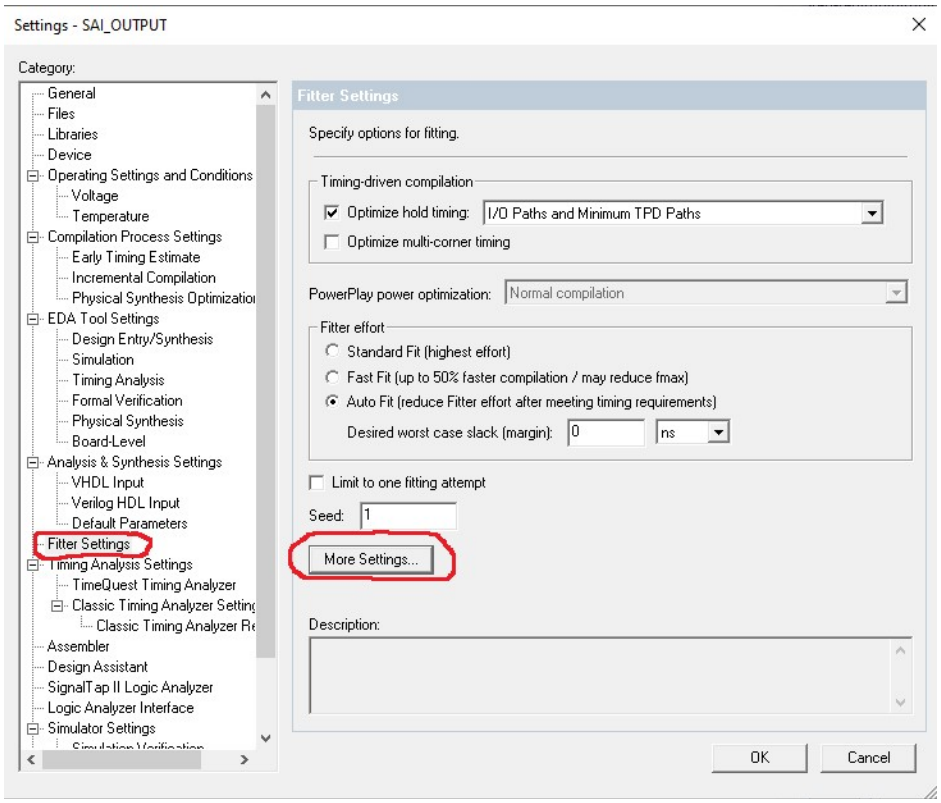
В окошке справа выбираем клок соединение **iCLK_c**, слева – созданную группу портов и задаем опции как на скрине. Закрываем окно и кликаем на опцию **FREQUENCY**:



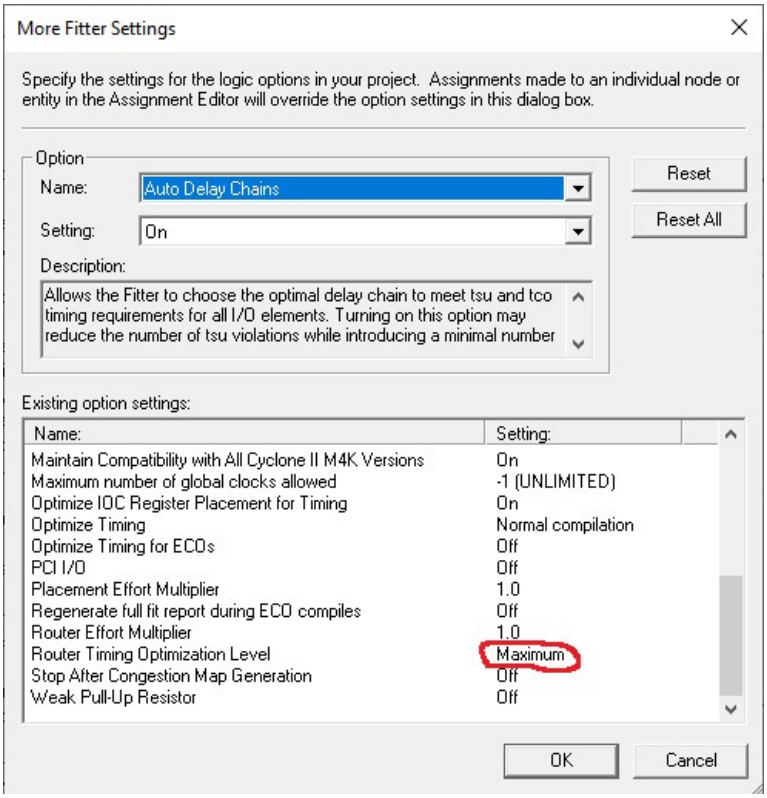
Опять выбираем **Net iCLK_c** и задаем для него частоту 50МГц.
Должно получиться примерно так:



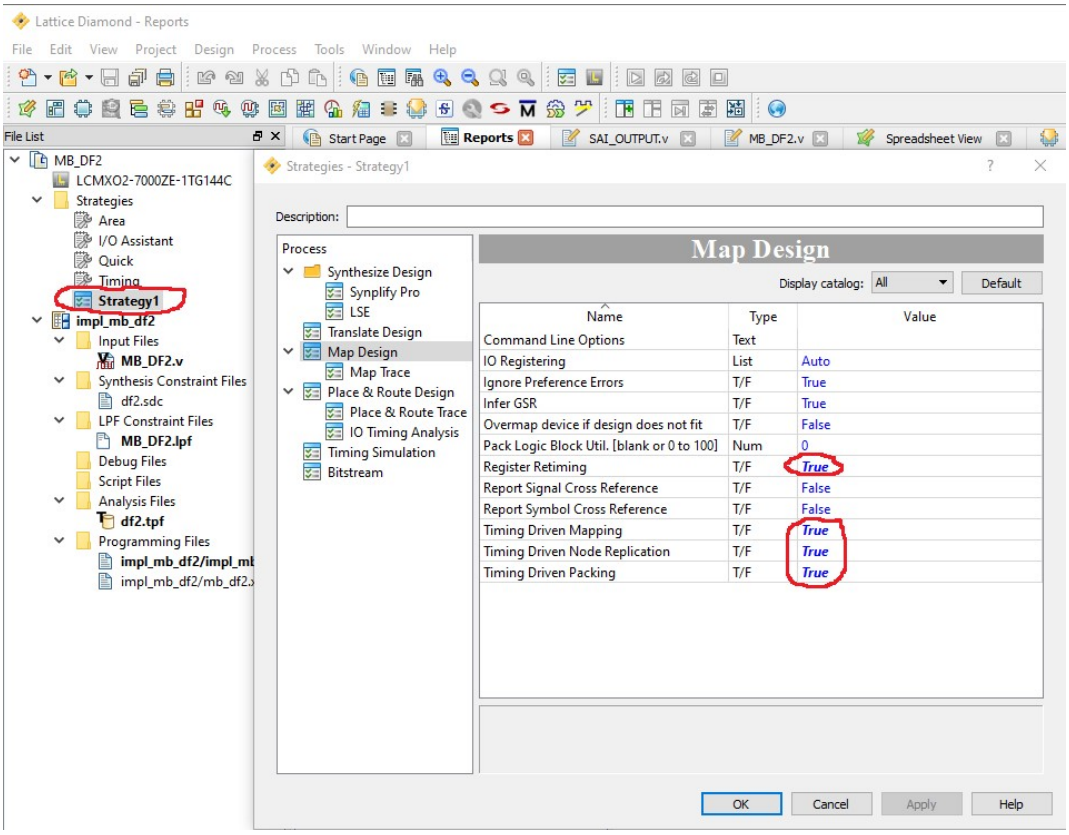
3. Настройка оптимизации трассировщика.
   Для FPGA **Cyclone** не обязательно, можно этот шаг пропустить.
   В **Quartus** находится в разделе **Fitter Setting** в св-вах проекта.



Открываем расширенные настройки и задаем максимальную оптимизацию:



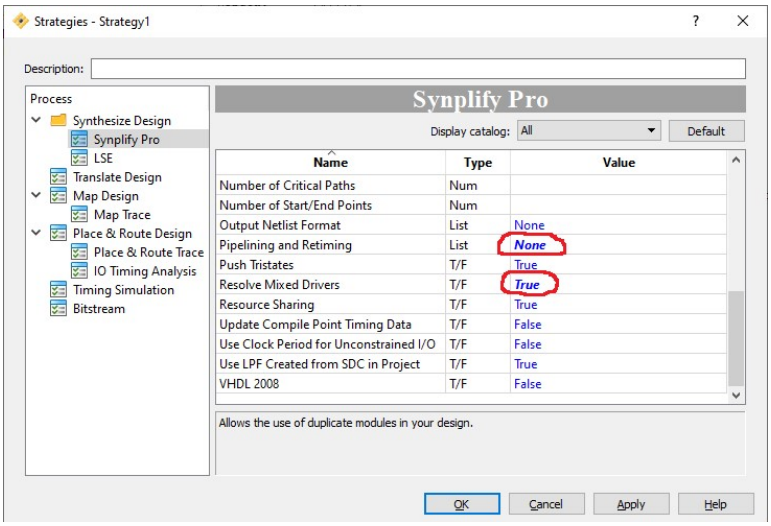
В **Diamond** открываем конфигурацию компилятора **Strategy**, переходим к меню **Map Design** и включаем опции как на скрине (для плис с высоким грейдом быстродействия (**HE**, **HC**) этот пункт можно пропустить, что позволит сэкономить ресурсы):



Для ПЛИС **Lattice** необходимо рассмотреть еще одну настройку синтеза. Это автоматическая конвейеризация логики. Данная опция повышает быстродействие логики за счет большего расхода ресурсов, что позволяет использовать FPGA Lattice с самым низким грейдом скорости **ZE**. Но не позволяет вместить проект DF2 в микросхему LcmXo2-2000ZE.

В таком случае взамен данной опции можно использовать более быстрые ПЛИС с индексом **HE** или **HC**, либо для грейда **ZE** понизить тактовую частоту. А конвейеризацию отключить в св-вах компилятора **Synplify Pro**.

Также, для исключения конфликтов компиляции в Diamond следует включить опцию **Resolve Mixed Drivers**.



Функционал ядра фильтра DF2_FIR_CORE.

[1:0] **iATT** – входы задания аттенюации входных аудио-данных.

b00: 0дБ

b01: -1дБ

b10: -2дБ

b11: -3дБ (для данного значения доступна перезапись через SPI интерфейс)

[1:0] **iDF_MODE** – входы выбора режима ЦФ

b00: **Normal** – Режим стандартного полуполосного фильтра-интерполятора.

b01: **Sharp** – Режим фазолинейного ЦФ с крутым спадом (-90дБ на частоте Найквиста)

b10: **Short** – Режим минимальнофазового ЦФ с малой задержкой (-90дБ на частоте Найквиста)

b11: **Slow** – Режим полуполосного ЦФ с пологим спадом (доступна перезапись через SPI интерфейс).

[2:0] **iOVS_MAX** – Максимальный оверсемплинг ЦФ.

Данный параметр задает максимальную кратность оверсемплинга **OVS RATIO** по формуле: **OVS RATIO** = 2 ^ **iOVS_MAX**. Диапазон допустимых значений от 0 – x1, до 5 – x32.

Фактическая кратность оверсемплинга задается автоматически с учетом входной частотой семплирования.

[2:0] **oINOVS** – выходы индикации входной частоты семплирования, определяемой как кратность оверсемплинга на входе: **IN OVS RATIO** = 2 ^ **oINOVS**.

Диапазон выходных значений 0..4. Кратность 0 соотв. частоте семплирования на входе 44.1/48кГц. Кратность 4 соответствует – 705.6/768кГц.

oOVFL, **oOVFR** – выходы индикации переполнения в шине данных входным сигналом. Используется для индикации клипа входного сигнала.

При наличии переполнения в выходном семпле индикатор соответствующего канала устанавливается в единицу, пока выходной семпл не сменит следующий, без переполнения.

MCLKSEL – кратность частоты мастерклока. Задается как **`MCLK_SEL_x1024** или **`MCLK_SEL_x768**. Для кратности 1024 доступны частоты тактирования 1024Fs, 512Fs, 256Fs. Для Кратности 768 доступны частоты тактирования 768Fs, 384Fs.

UPSAMPLE – параметр активирующий дополнительный апсемплер, включаемый на выход основного фильтра-интерполятора. Рекомендуется включать только при необходимости оверсемплинга выше x32. Суммарный оверсемплинг равен произведению оверсемплинга основного ЦФ и дополнительного интерполятора.

Например, если **iOVS_MAX** = 5, **UPSAMPLE** = **`DF2_UPSAMPLE_x4**, то результирующий оверсемплинг на выходе модуля **DF2_FIR_CORE** равен OVS RATIO = (2 ^ 5) x 4 = x128.

NSHOR – порядок ноиз-шейпинга, используемого при округлении. Доступные значения от 0 (шейпер выключен), до 3 (3-ий порядок).

LENMIN – минимальная разрядность округления выходных данных в битах. Определяет уровень дополнительной аттенюации перед округлением.

Если разрядность выходных данных **iODW** задать меньше параметра **LENMIN**, то возможно переполнение сумматора шейпера и дизера.

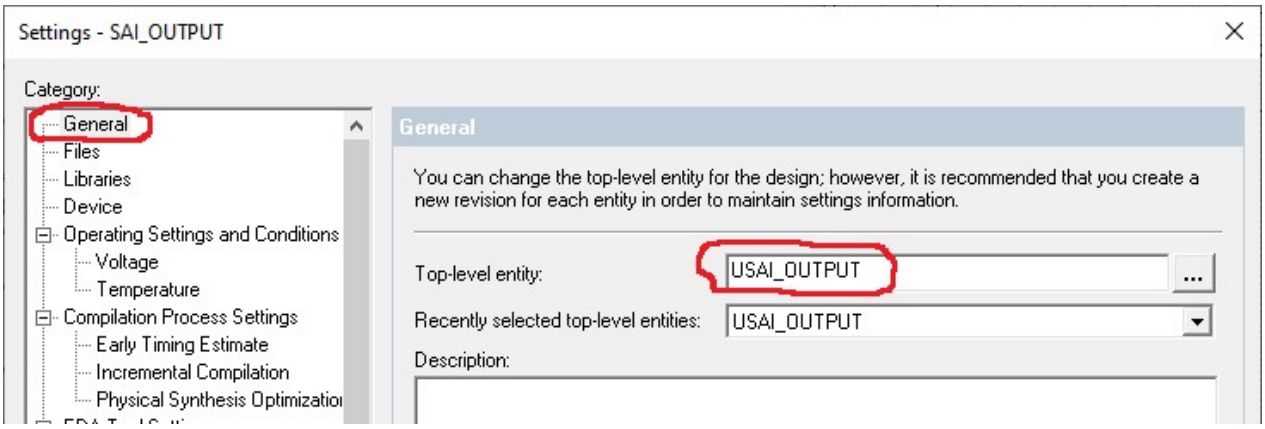
Функционал модулей вывода аудиопотока.

Т.к. модули вывода имеют достаточно сложный функционал, то вместо их полного описания к исходникам **DF2** прилагается проект квартуса с настроенными временными диаграммами для моделирования результата конфигурации выхода.

Проект содержит два разных модуля на выбор:

1. **SSAI_OUTPUT** – максимально простой модуль с непрерывным битклоком и правым выравниванием данных, рассчитанный на работу с аудио ЦАП-ами.
2. **USAI_OUTPUT** – универсальный модуль, поддерживающий режим как с разрывным, так и с непрерывным битклоком, имеющий более сложную схему делителя битклока и работающий не только с аудио, но и с разными SPI ЦАПами.

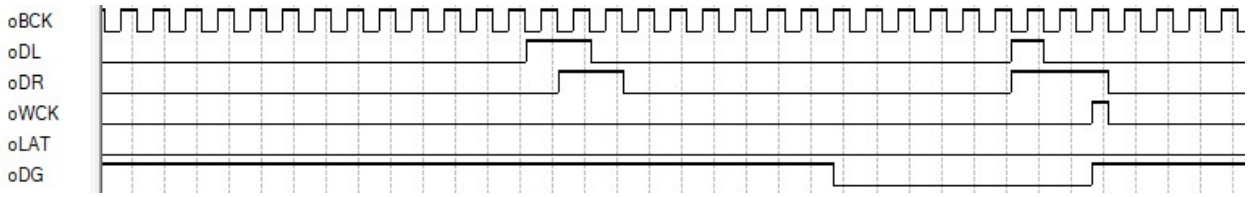
Для подключения нужного модуля в проекте симуляции в Quartus необходимо в настройках проекта задать имя модуля как верхнего модуля в иерархии проекта:



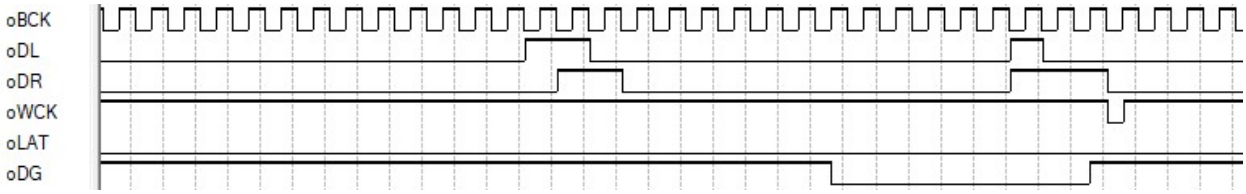
Параметризация модуля SSAI_OUTPUT

WCKW (word clock width) – Задает ширину строба **WCK** в тактах iCLK.

Минимальное значение 1:



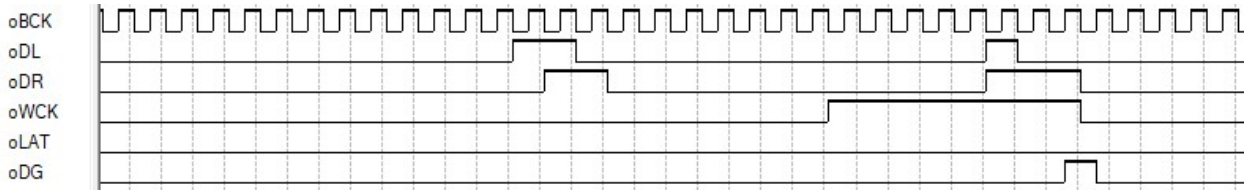
Максимальное значение зависит от периода фрейма (от выходной частоты семплирования), и рассчитывается как N-1, где N – число тактов iCLK за период выходного фрейма.



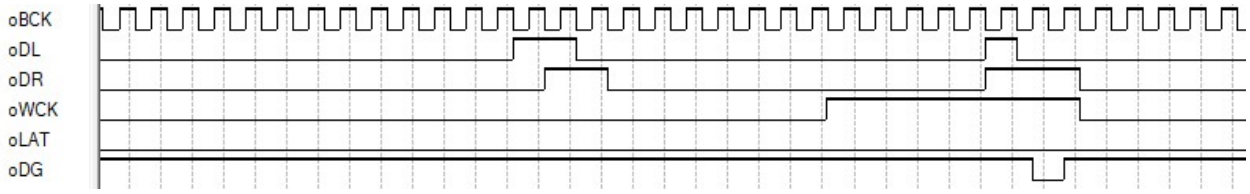
DGHW (deglitcher hold width) – Задает ширину уровня хранения для сигнала деглитчера **DG**.

Данный параметр задает ширину строба хранения с дискретностью 32 шага для фрейма любой ширины.

Минимальное значение 1:



Максимальное 31:



Параметризация модуля USAI_OUTPUT

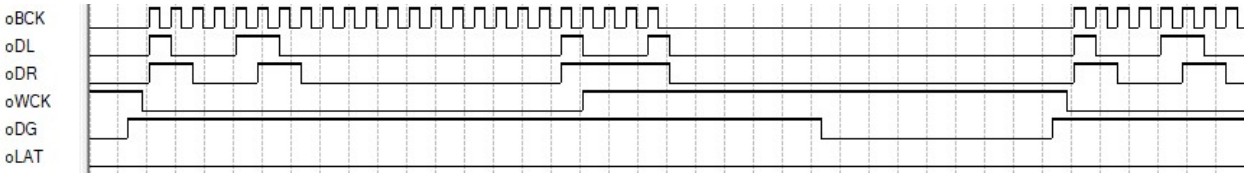
SPIMODE – Задает режим вывода данных.

0 – аудио-интерфейс:

Фрейм **WCK** формируется для загрузки в аудио ЦАПы, с переходом в единицу до того как завершится битклок фрейма.

Сигнал **LAT** отключается (фиксируется в нуль).

Сигнал **DG** выравнивается по спадающему фронту **WCK**.



1 – SPI интерфейс:

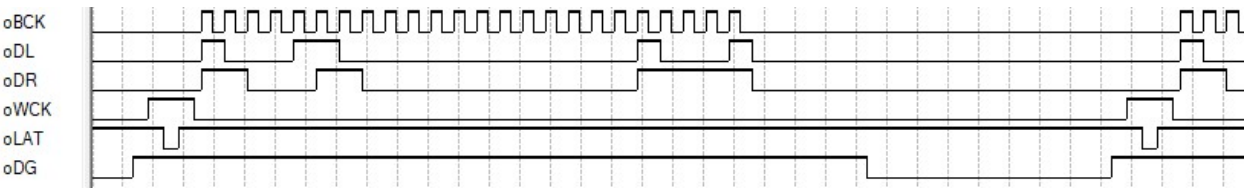
Фрейм **WCK** формируется для загрузки в SPI ЦАПы.

Строб **WCK** используется как сигнал **CS** (chip select) SPI шины.

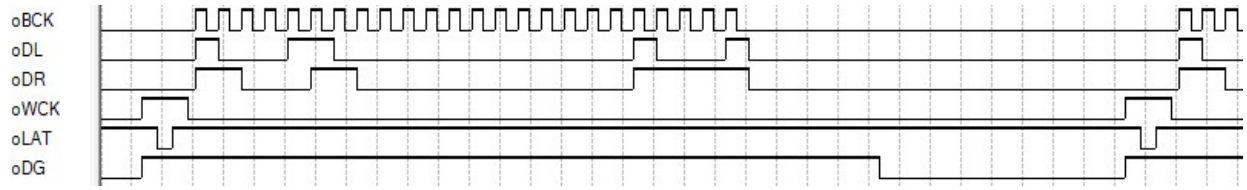
Сигнал **LAT** активируется и может быть использован для синхронного обновления выхода SPI ЦАП.

Сигнал **DG** получает выравнивание в соответствии с параметром **DGMODE**:

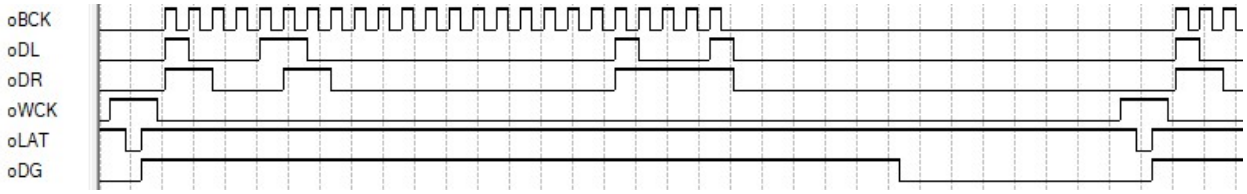
DGMODE = 0 – выравнивание **DG** по возрастающему фронту **WCK**:



DGMODE = 1 – выравнивание **DG** по спадающему фронту **LAT**:



DGMODE = 2 – выравнивание **DG** по спадающему фронту **WCK**:

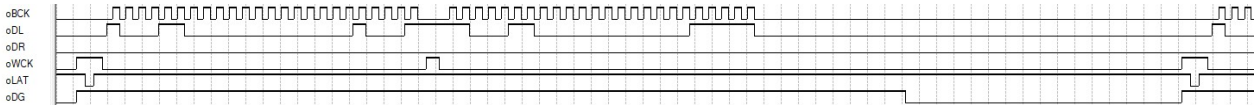


DDM (dual data mode) – Активирует режим последовательной загрузки данных в сдвоенные SPI ЦАПы.

Для использования такого режима необходимо активировать режим **SPIMODE**.

Синхронное обновление обоих ЦАП в таком режиме выполняется по сигналу **LAT**.

DGMODE задается 1 – выравнивание по спадающему фронту **LAT**.



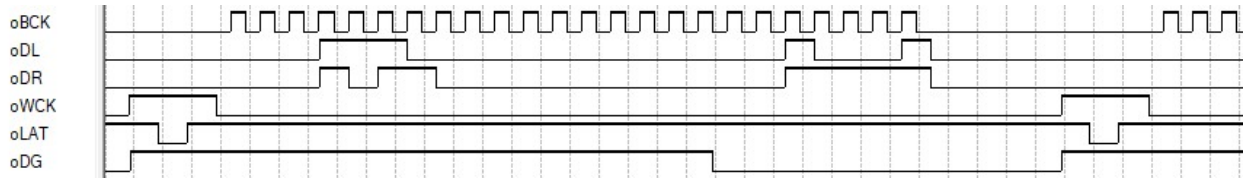
HEADLEN – задает размер заголовка в битах для SPI ЦАП.

Для ЦАП, не требующего заголовка, данный параметр задается нулем.

HEADL – задает значение заголовка для левого канала.

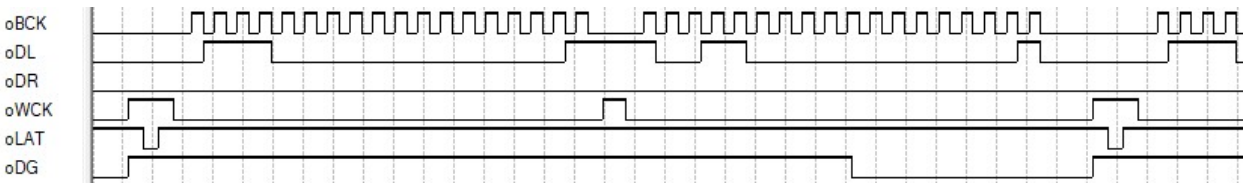
HEADR – задает значение заголовка для правого канала.

Пример SPI фрейма с длинной данных 20 бит, с заголовками длинной 4 бита, со значением 0x1 (AD5791):



В режиме **DDM** параметры **HEADL** и **HEADR** задают заголовки для первого и второго блока данных во фрейме соответственно.

Пример SPI фрейма с данными длинной 16 бит и заголовками 0x1 и 0x2 длиной 2 бита для загрузки данных в ЦАП DAC8812:



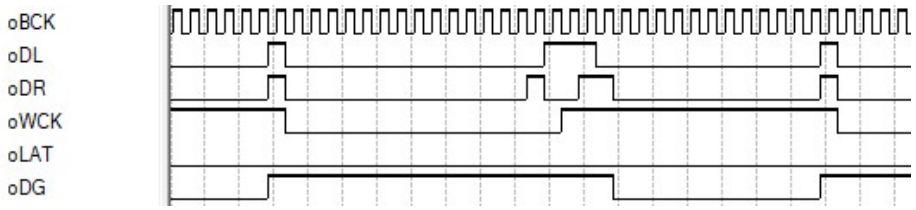
DGHW (deglitcher hold width) – Задает ширину уровня хранения для сигнала деглитчера **DG**, аналогично модулю **SSAI_OUTPUT**, описанному выше.

CONT (continuous bck) – Активирует режим непрерывного битклока.

Для данного режима не допускаются значения **BCKDIV** 2 (**iCLK**/3) и 4 (**iCLK**/6).

Параметр **SPIMODE** должен быть в нуле.

Данные выводятся с выравниванием вправо по спадающему фронту строба **WCK**.



SPI интерфейс

Коэффициенты первого каскада интерполятора, а так же значение аттенюатора можно загрузить через SPI интерфейс.

Данный интерфейс активируется низким уровнем на входе chip select (**CS**), и работает не зависимо от мастерклока (вся логика тактируется **SCK** сигналом). Данные **DATA** задвигаются по положительному перепаду сигнала **SCK**.

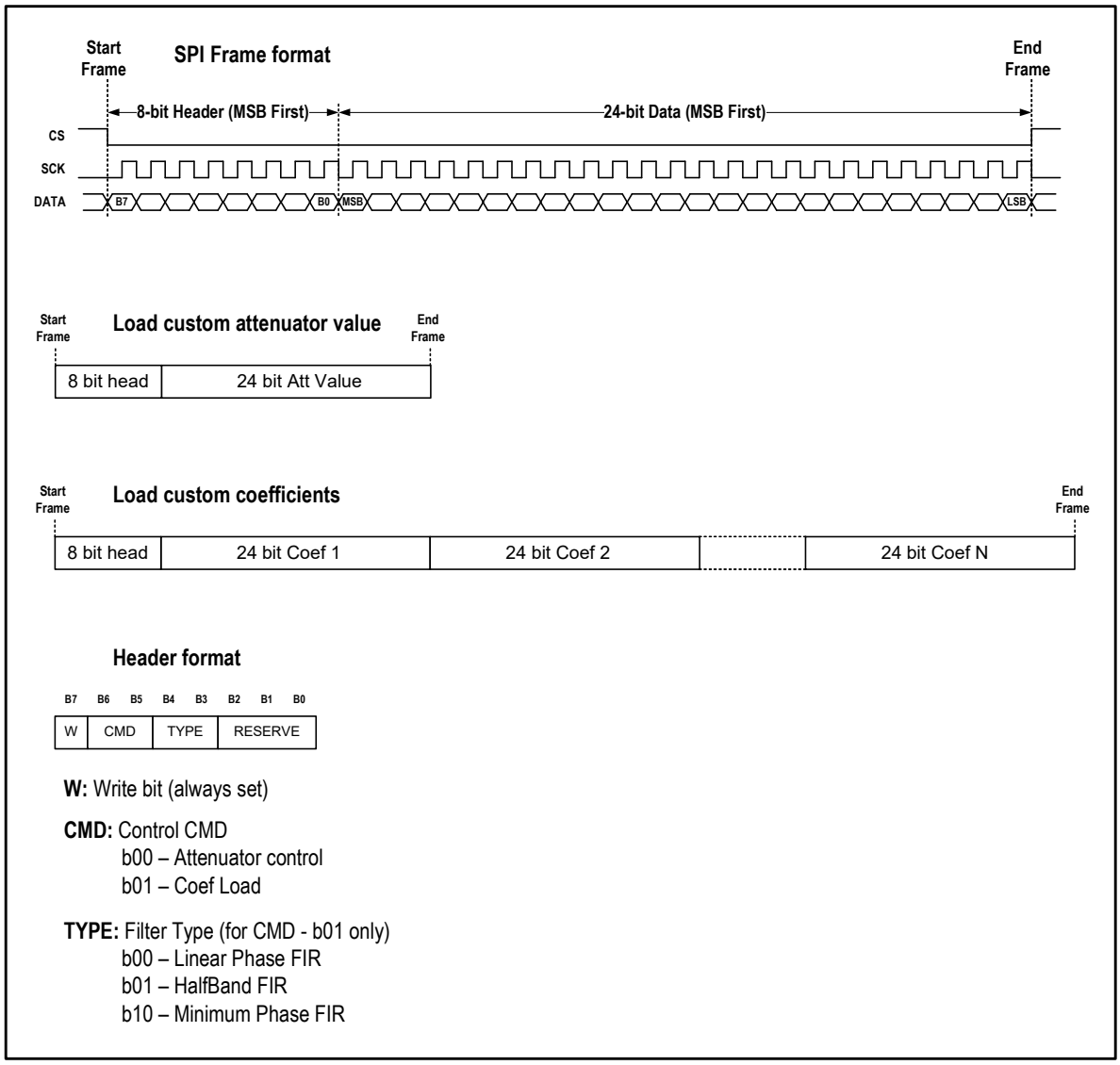
SPI фрейм состоит из 8-битного заголовка и одного или нескольких 24-битных слов данных, следующих за заголовком.

Как заголовок, так и блоки данных выводятся старшим битом вперед.

Фрейм с командой задания аттенюации содержит один блок данных.

Фрейм с командой записи коэф-тов может содержать любое число 24-хбитных блоков с коэф-тами. Подсчет кол-ва коэф-тов фильтра выполняется автоматически. В заголовке указывается только тип фильтра.

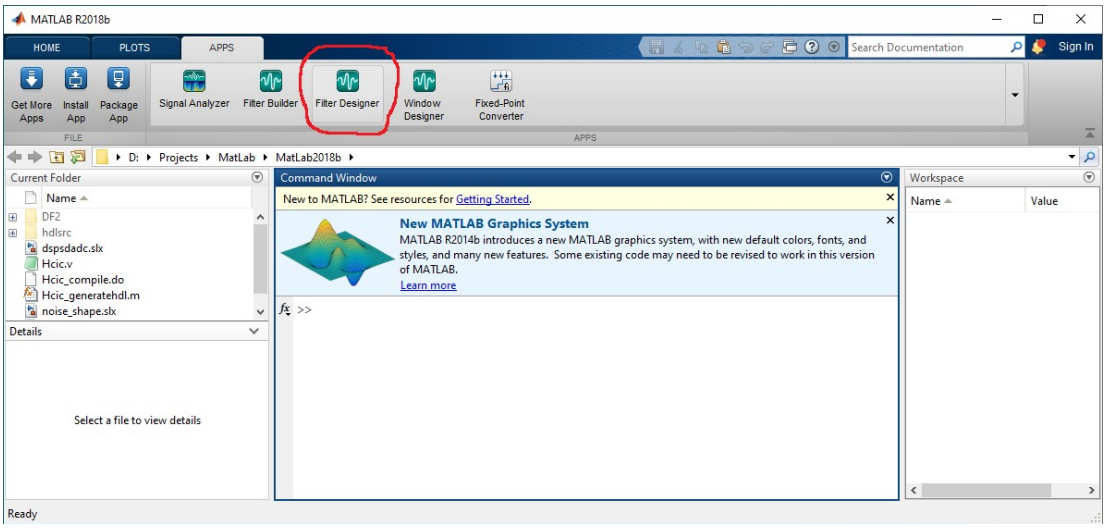
Для минимально-фазового фильтра пишется весь ряд коэф-тов в прямом порядке импульсной хар-ки. Для фазолинейныйх фильтров (простой FIR или HalfBand) пишется только половина ряда, начиная с центрального максимума.



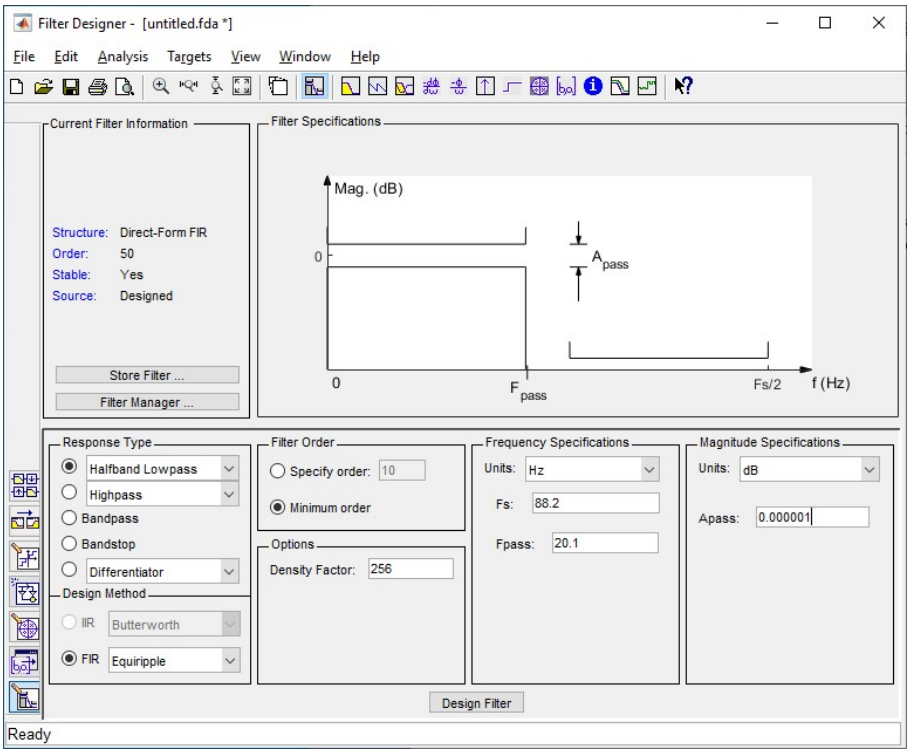
Данные, загружаемые через SPI, сохраняются только в ОЗУ FPGA. Поэтому после сброса питания ОЗУ восстанавливается в изначальное состояние.

Расчет коэффициентов фильтров в MatLab и интеграция в проект.

Из компонентов матлаба достаточно установить симулинк и **DSP system toolbox**.
Для расчета, анализа фильтров и генерации коэф-тов удобно использовать фреймворк **Filter Designer**:

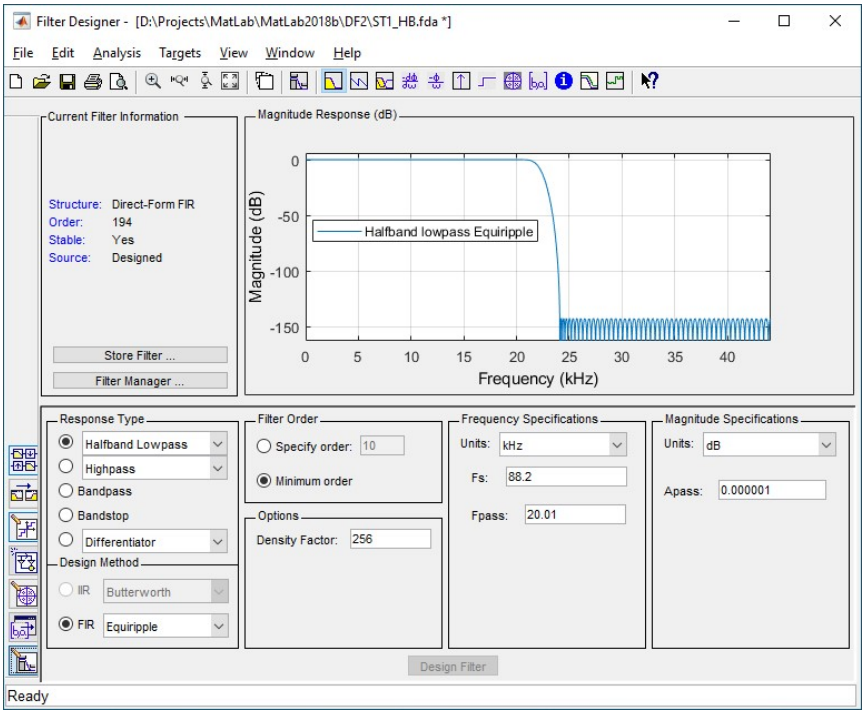


Пример задания параметров для расчета полуполосного фильтра первого каскада интерполятора:



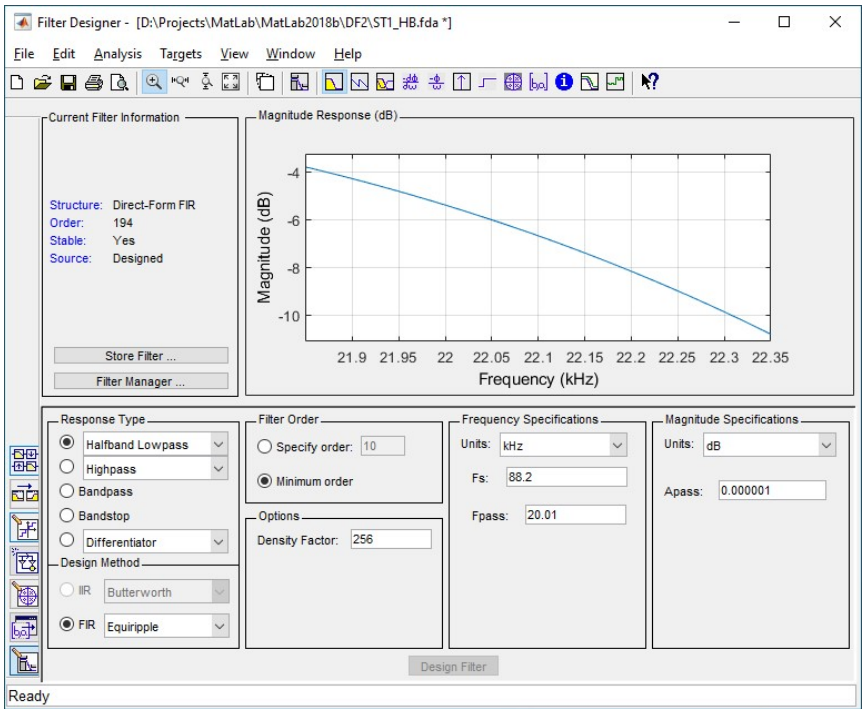
Выбираем тип фильтра: например полуполосный **Halfband Lowpass** с птимизацей **Equiripple**.
Т.к. для обеих сеток частот (44,1 и 48кГц) используются общие фильтры, то расчет выполняется в сетке 44,1кГц. Частота семплирования **Fs** – это частота на выходе интерполятора, поэтому ее выбираем вдвое выше входной, т.е. 88,2кГц. Порядок минимальнофазового фильтра можно выбирать произвольно. **Для фазолинейных фильтров в проекте DF2 допустим только четный порядок!**

Fpass – полоса пропускания.
Apass – неровномерность в полосе пропускания (размах).
Density Factor – кол-во точек на частотной хар-ке по которым ведется расчет (на расчет особо не влияет).
После задания параметров жмем кнопку **Design Filter**:

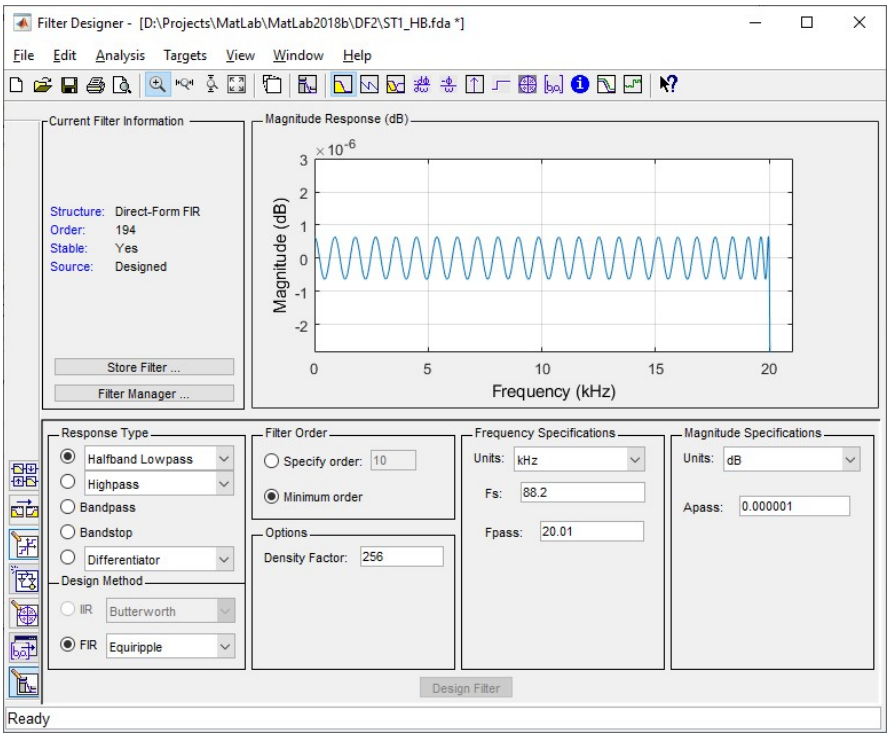


Получаем рассчитанный фильтр. Слева можно видеть порядок фильтра (**Order**). Можно посмотреть его хар-ки.

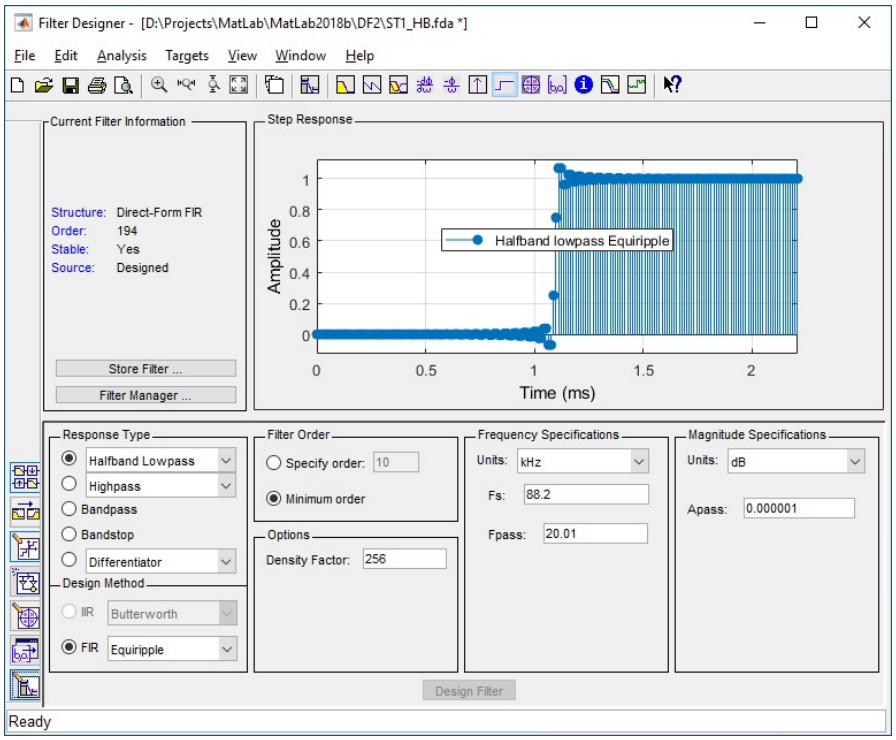
Проверить ослабление на частоте Найквиста:



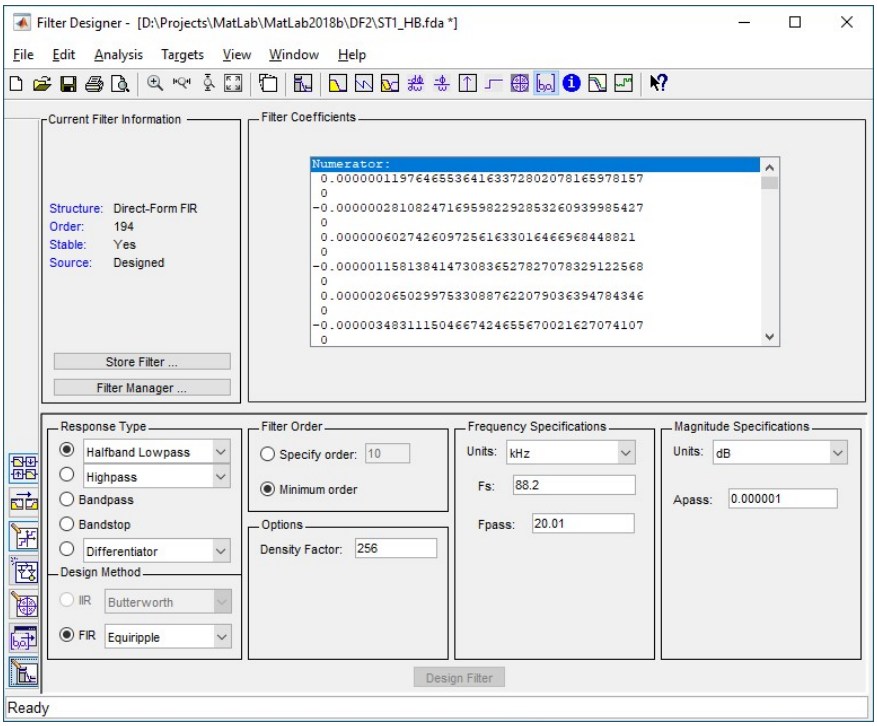
Неравномерность в полосе пропускания:



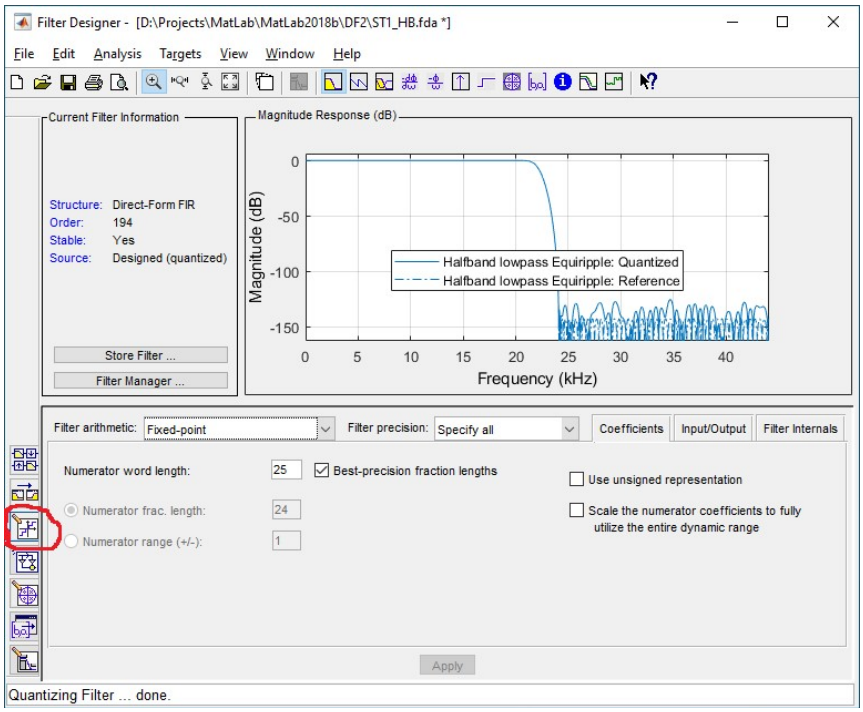
Посмотреть переходную хар-ку:



Коэф-ты:

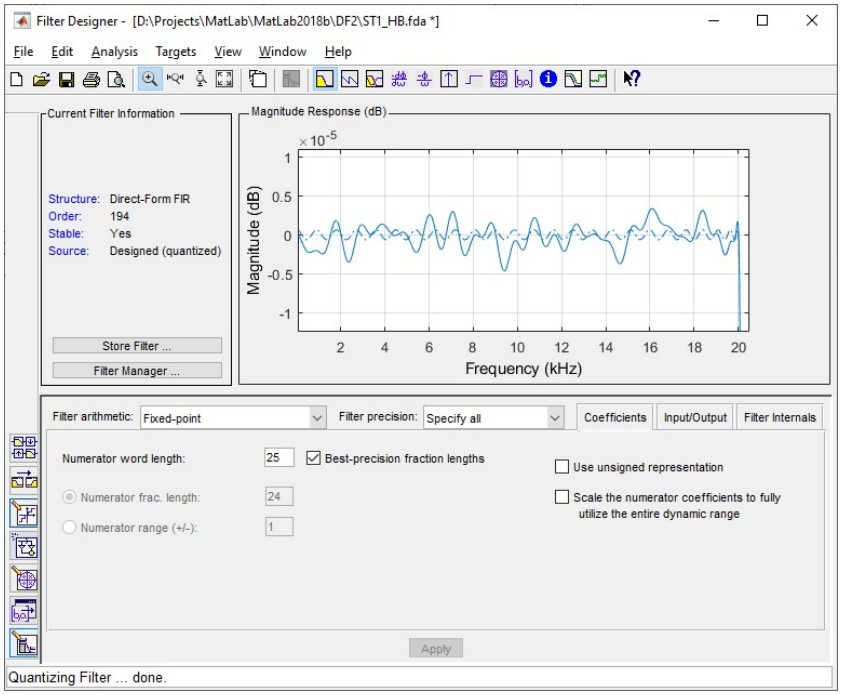


Но данный фильтр рассчитан в формате **double** с плавающей точкой.
Для использования коэф-тов необходимо выполнить расчет с целочисленной разрядностью коэф-тов.
Переходим в меню параметров квантования и выбираем режим **Fixed-point**:

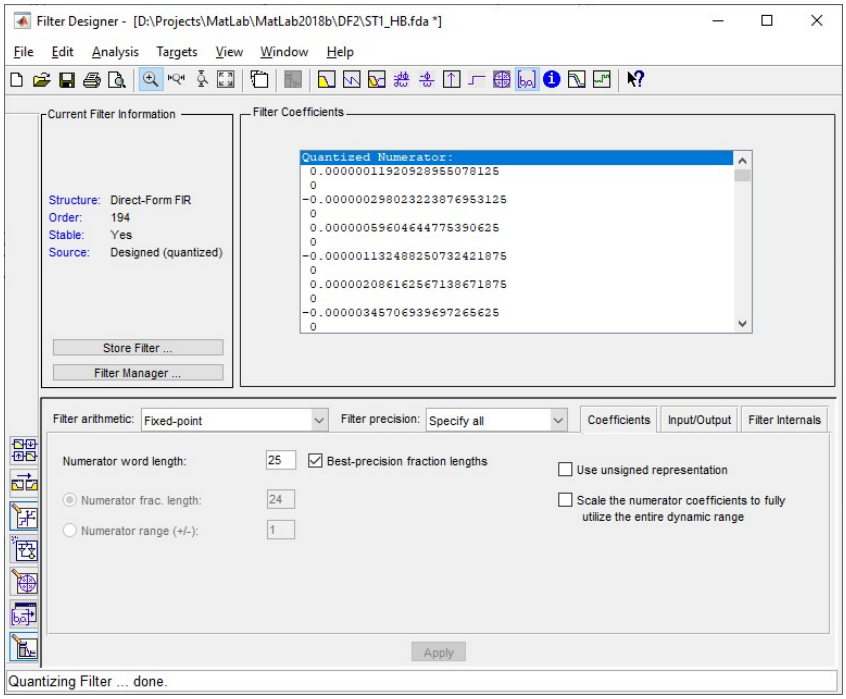


Для коэф-тов полуполосных фильтров разрядность задаем 25 бит. **Numerator range (+/-)** при этом должен получиться единица. Если убрать галку **Best-precision**, то диапазон нумератора можно задать вручную.

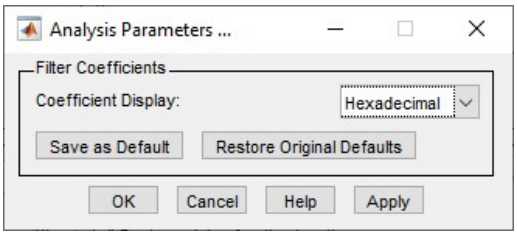
После задания разрядности график АЧХ фильтра приобретает новый вид, соответствующий квантованным коэф-там. Неквантованный график при этом тоже остается, но становится пунктирным. Можно видеть как ухудшаются хар-ки фильтра при ограничении разрядности коэф-тов.
В том числе в полосе пропускания:



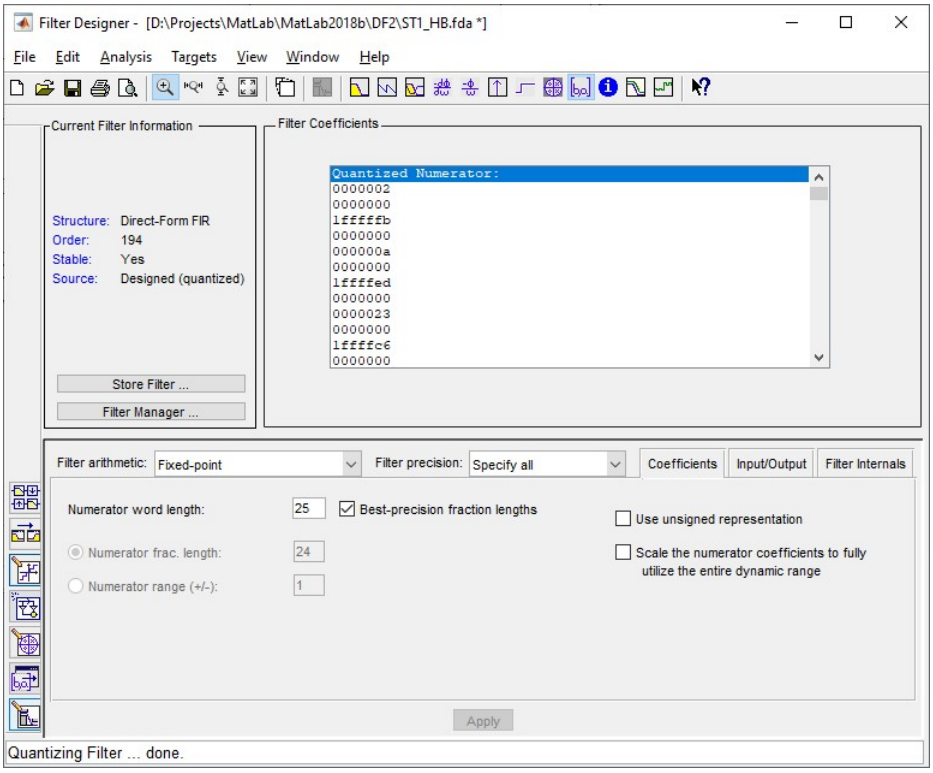
В окне коэф-тов при этом появляется ряд квантованных значений. Теперь нужно преобразовать эти квантованные коэф-ты в шестнадцатиричную форму и сохранить в файл. Для этого переходим к окну коэф-тов:



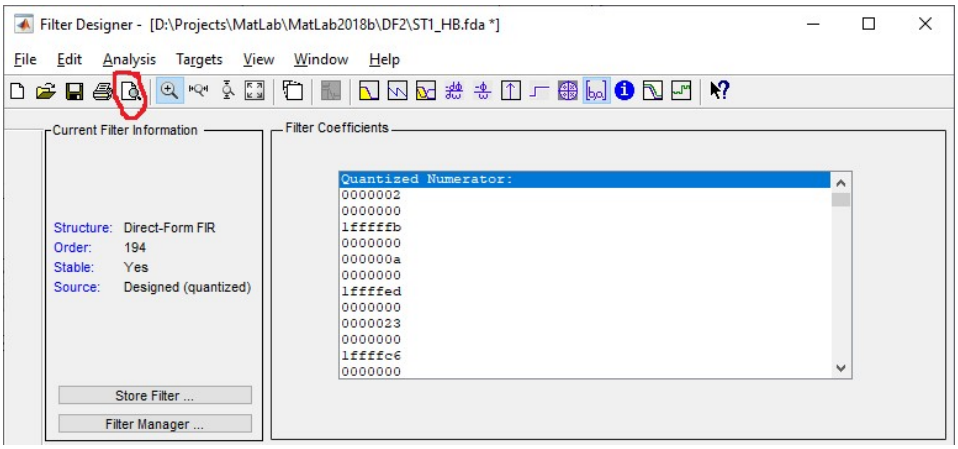
Далее: **Analysis** -> **Analysis Parameters** -> выбираем **Hexadecimal**



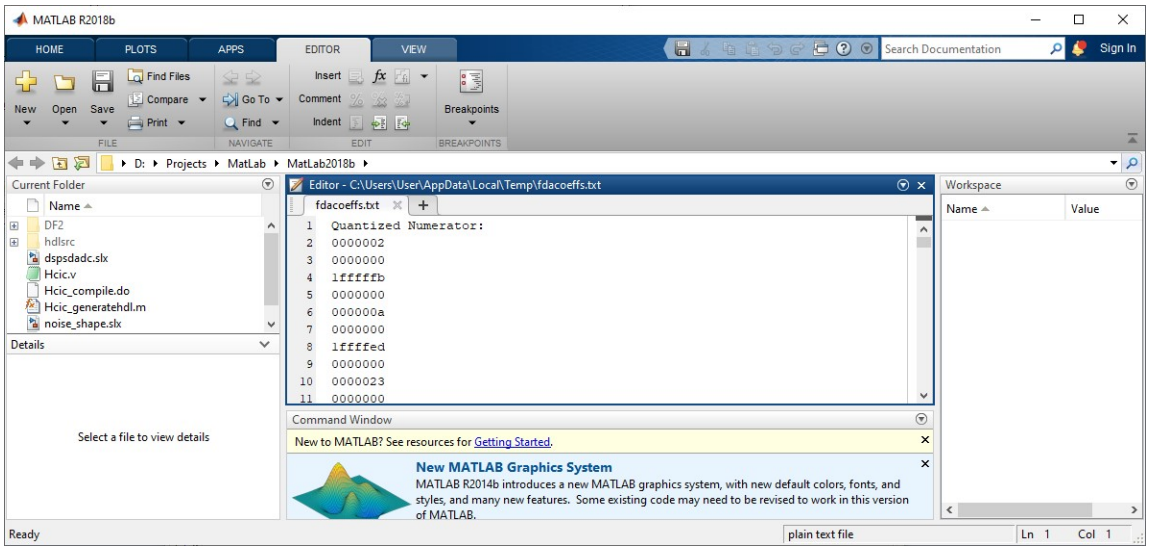
Получаем ряд коэф-тов в хексе:



Теперь выводим в файл и сохраняем. Для этого жмем кнопку предварительного просмотра печати:

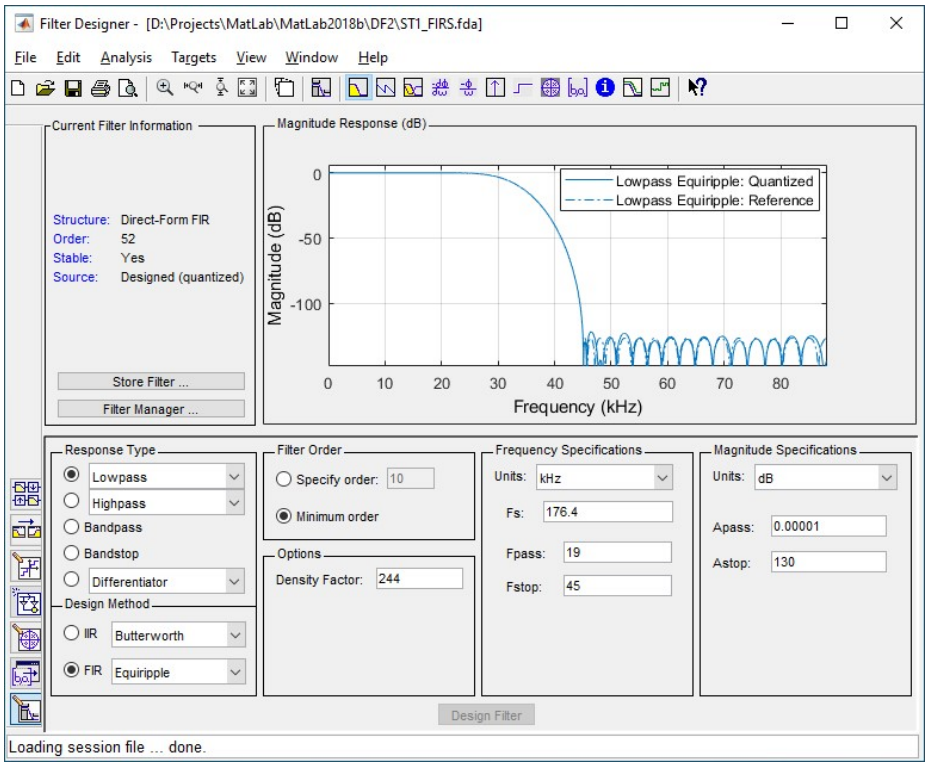


В главном окне матлаба выводится файл с коэффициентами:

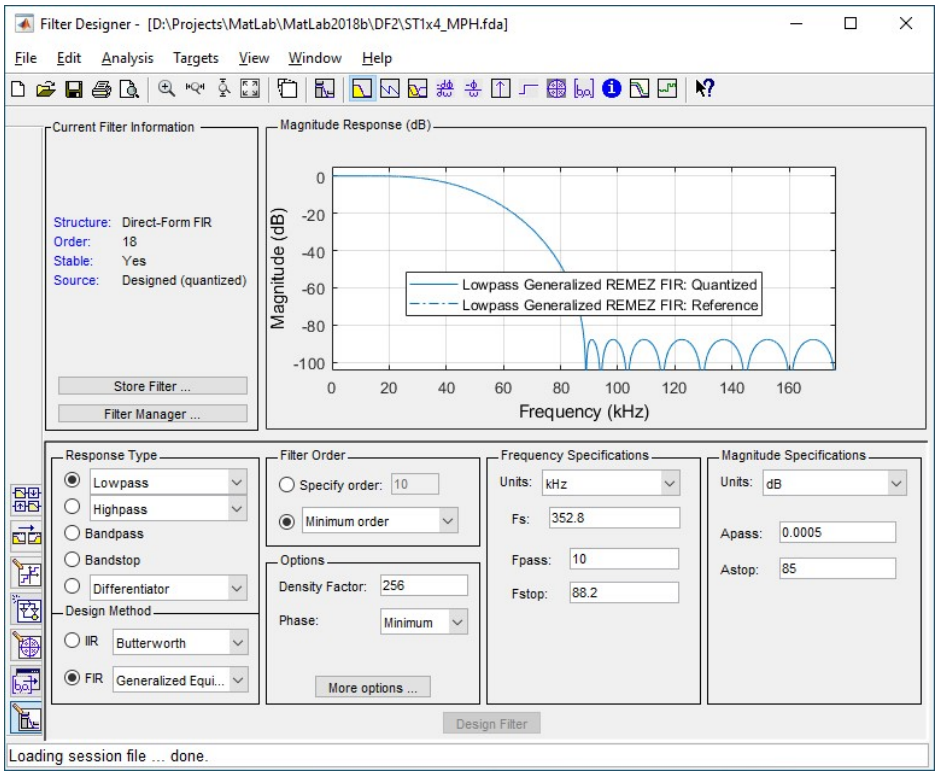


Сохраняем его под нужным именем с расширением txt.
Файл с коэффициентами готов.
Аналогично делаются коэф-ты для других типов фильтров.
Разрядность коэф-тов 25 бит с диапазоном +/- 1 выбирается только для полуполосных фильтров!
Для фильтров остальных типов разрядность выбирается 24 бита с диапазоном нумератора +/- 0.5!

Пример выбора обычного фазолинейного FIR:



Пример выбора минимальнофазового FIR:

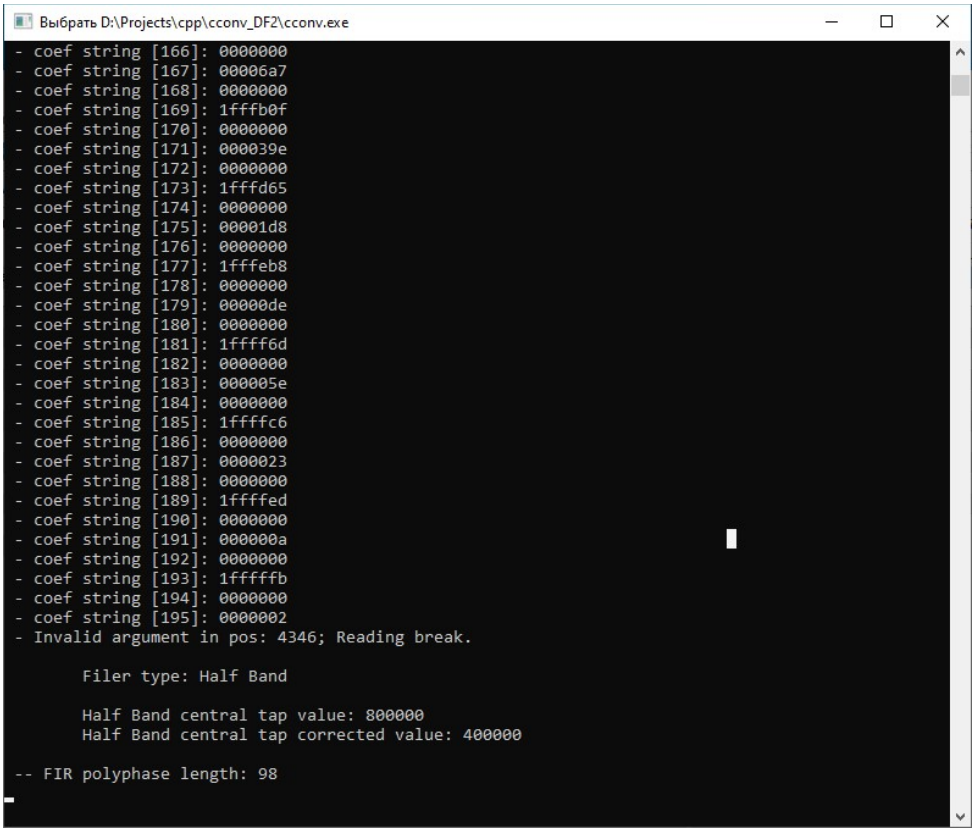


Для добавления коэф-тов в проект необходимо значения из текстового файла преобразовать в формат верилог-функции “ $readmemh” с 12-битными словами.
Такое преобразование автоматически выполняется с помощью утилиты **cconv.exe**.

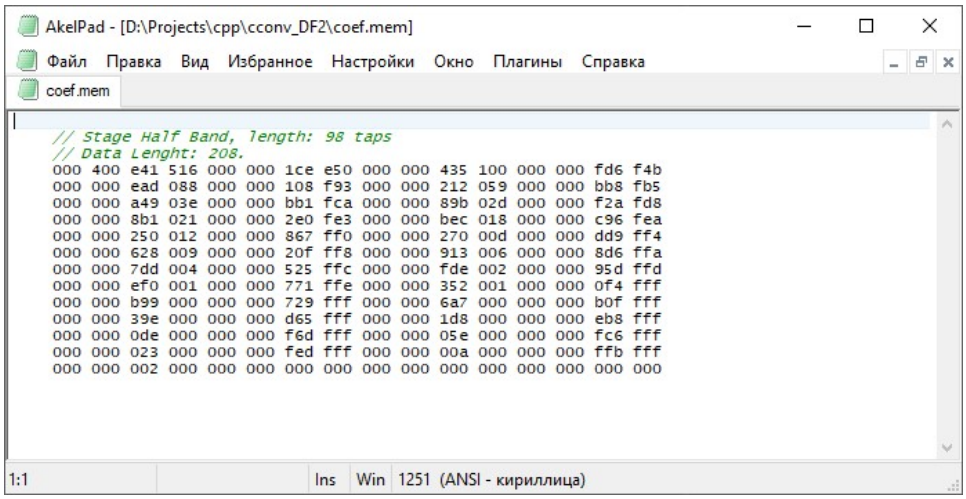
В папку с утилитой копируем текстовый файл с коэффициентами из матлаба, и задаем этому файлу имя **coef.txt**. Утилита ищет файл именно с таким именем.
Запускаем утилиту, видим параметры преобразования:

Жмем любую кнопку и смотрим результат конверсии.
Утилита по ряду коэф-тов автоматически распознает тип фильтра, вырезает нулевые значения в начале и в конце ряда, формирует полифазный ряд для применения в проекте DF2 и выполняет выравнивание значений в соответствии с параметром **Align** в файле **param.txt**. **Данный файл с параметрами по умолчанию создается автоматически.**

Кроме того, утилита пакует 25 разрядные коэффициенты полуполосного фильтра в 24-битные значения (делит на 2 центральный коэффициент).



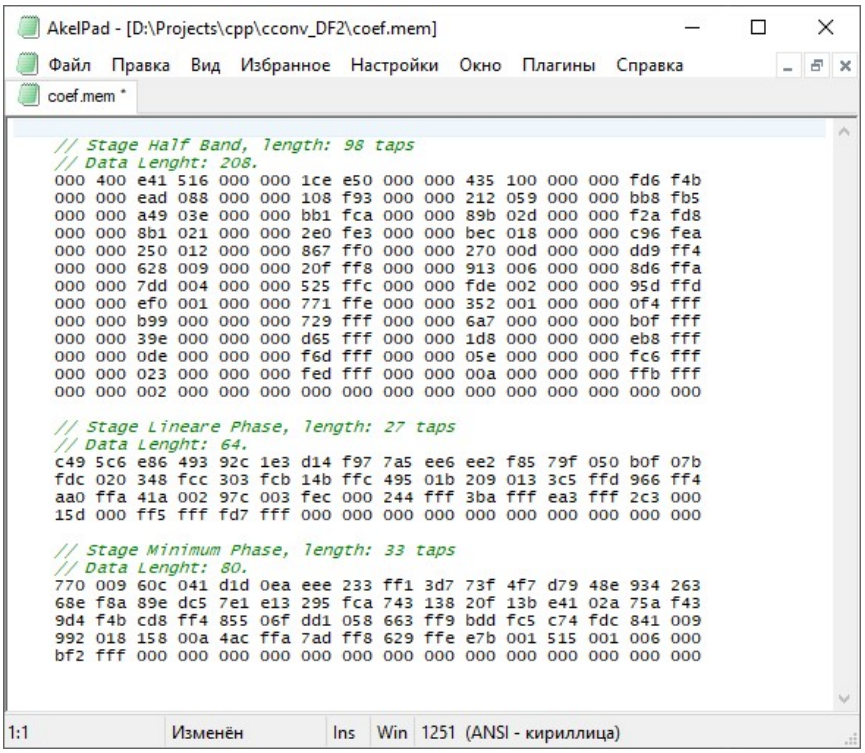
По завершении конвертации коэф-тов утилиту можно закрывать. В папке с утилитой появляется файл со сгенерированными коэф-тами **coef.mem**.



Внутри файла можно видеть ряд 24-хбитных коэф-тов, разбитых на 12-битные значения младшим словом вперед. В конце ряд дополняется нулевыми значения до заданного выравнивания **Align** в файле **param.txt**. Сверху ряда заголовок, в котором указан тип фильтра, полифазная длина (**length**), а так же фактическое кол-во 12-битных слов (с учетом выравнивания) для формирования смещений адресации к коэф-там в проекте DF2 (**Data Length**).

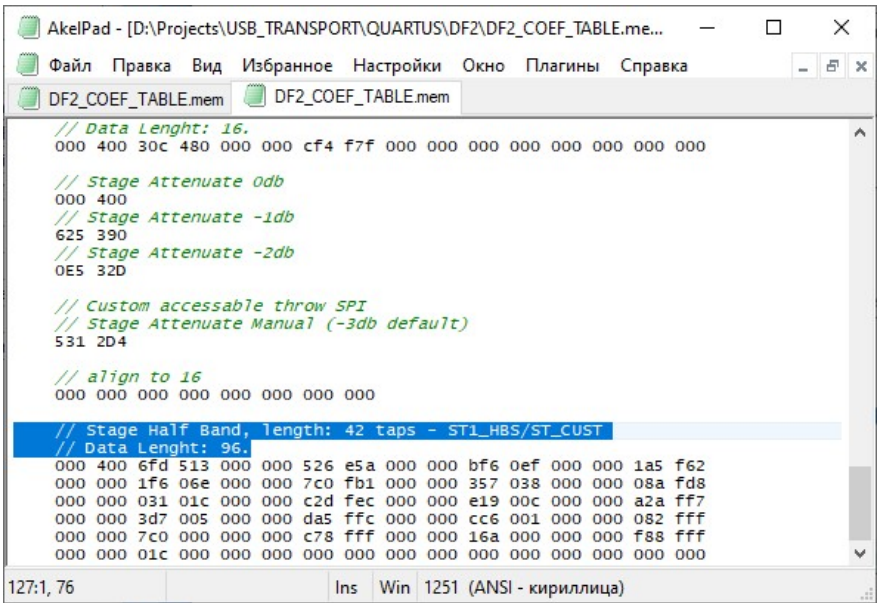
Для коэф-тов фильтров проекта DF2 выравнивание стартового адреса данных в памяти должно быть кратно 4! Для коэффициентов аттенюатора – кратно 2!

При повторном запуске утилиты файл **coef.mem** не затирается, новый ряд дописывается в конец файла.



Для добавления фильтра в проект DF2 открываем файл **DF2_COEF_TABLE.mem**.

В конце видим ряд коэф-тов полуполосного фильтра. Это фильтр для режима **Slow**. Этот же фильтр перезаписывается через **SPI** интерфейс:



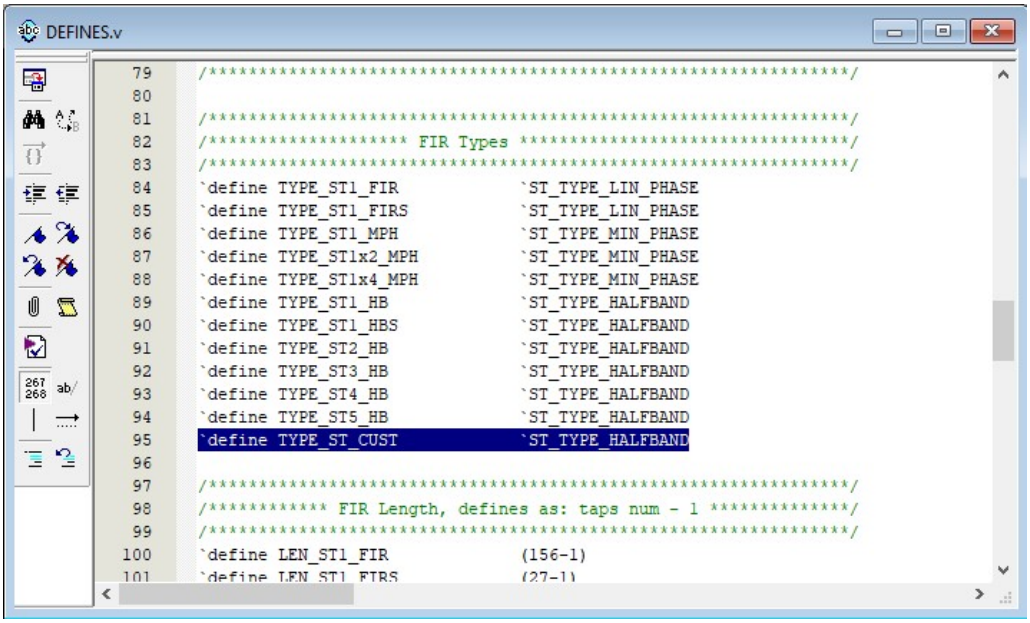
Можно вместо него вписать новый сгенерированный фильтр.

Выше располагаются коэффициенты для аттенюаторов, в том числе кастомного. Их так же можно поменять (вручную) на любые другие.

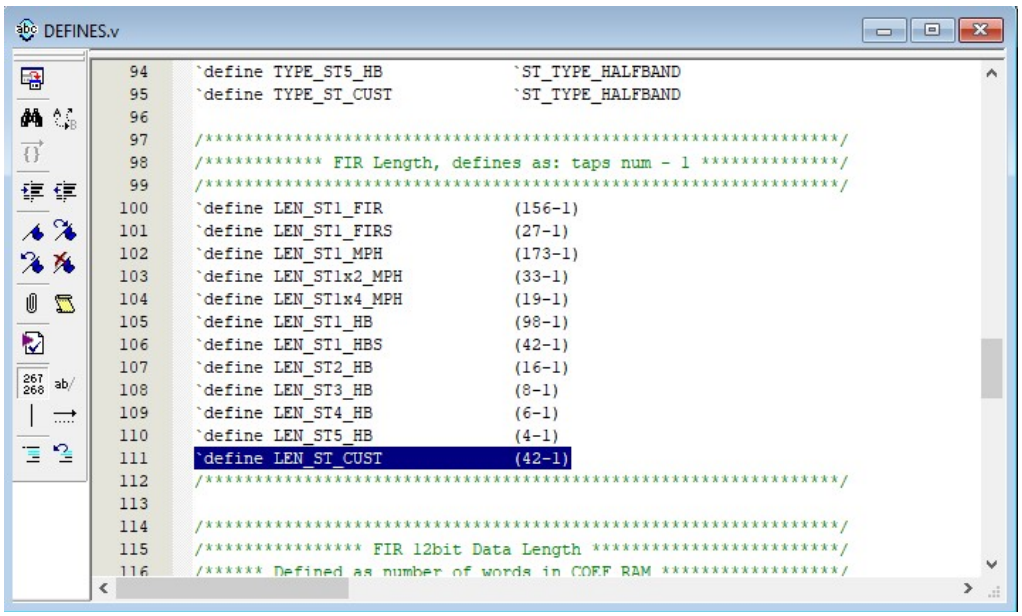
Небольшое отступление по заданию кастомной аттенюации:
Аттенюация выполняется умножением входных данных на коэффициент аттенюации. Единичный коэффициент (для аттенюации 0дБ) равен 0x400000 (4 194 304). Для получения коэффициента заданной аттенюации необходимо пересчитать единичное значение в соответствующее дробное, а затем домножить его на 4194304.
***Например**, необходимо выполнить аттенюацию -0,5дБ.*
Такой аттенюации соответствует множитель 0,944.
Расчитаем коэф-т: 4 194 304 х 0,944 = 3 959 423.
Переводим в хекс: 3C6 A7F, и задаем выравниванием младшим словом вперед: A7F 3C6. Готово.

Далее нужно задать параметры нового фильтра в фале DEFINES.v

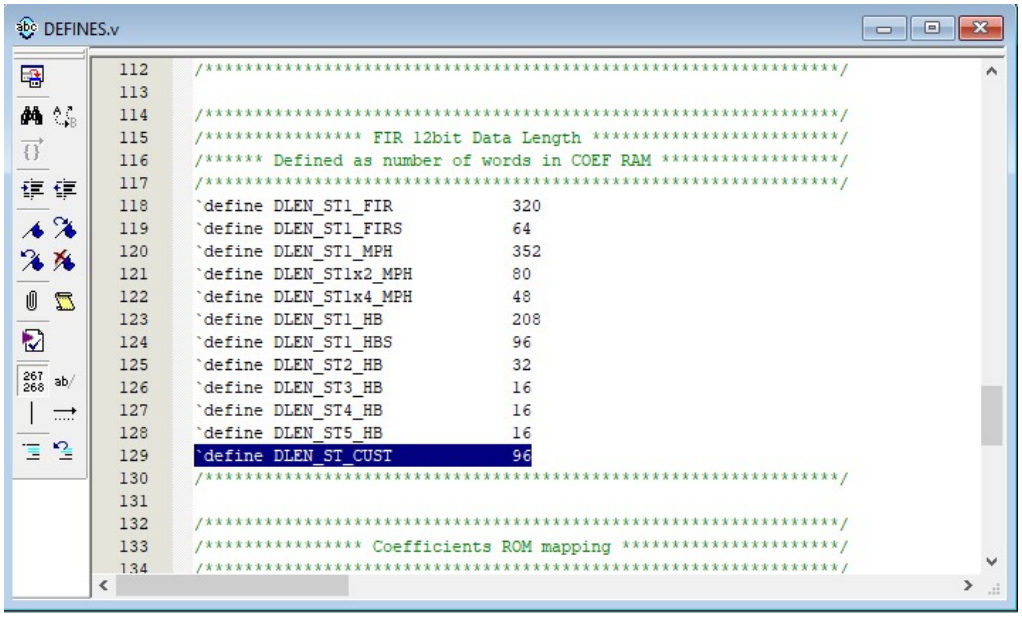
Тип нового фильтра в дефайне **TYPE_ST_CUST**:



Длину нового фильтра в дефайне **LEN_ST_CUST**. Длина задается как значение **length** из файла ***.mem** минус единица:



Для дефайна **DLEN_ST_CUST** нужно задать новую длину ряда в 12-битных словах, как значение **Data Lenght** из файла ***.mem**:



На этом процедура замены коэффициентов кастомного фильтра окончена. Теперь при выборе режима ЦФ «**Slow**» для первой ступени интерполятора будет задействован новый фильтр.

Аналогично можно заменить и любой другой ряд коэф-тов в проекте. Но нужно учитывать производительность фильтра, т.к. он может не успеть обсчитать фильтр заданной длины.

Также нужно учитывать, что выбор того или иного каскада фильтра выполняется в зав-ти от входной частоты семплирования (алгоритм выбора реализован в файле **DF_CONTROL.v**).

Оценивается производительность исходя из кол-ва умножений за период входной ЧД.

Например:
Частота мастер-клока 1024Fs.
Входная частота семплирования 2Fs (96kHz).
Выходная частота семплирования 16Fs (768kHz).

Получаем оверсемплинг 16/2 = x8.

В таком режиме работает 3 ступени интерполятора: 2->4, 4->8, 8->16. Первая ступень выполняет однократно умножения всех тапов, вторая ступень выполняет два прохода, третья – четыре. Каждое умножение это два такта мастерклока.

Т.к. входная ЧД 2Fs, то все эти умножения нужно успеть выполнить за 1024/2 = 512 тактов мастерклока. Нулевые коэф-ты при умножении не используются, поэтому для полуполосных фильтров их нужно вычесть.

Допустим первая ступень имеет 40 ненулевых тапов, вторая – 12, третья – 5. Получаем кол-во тактов мастерклока для первой ступени: (40x2)x1 = 80, для второй: (12x2)x2 = 48, для третьей: (5x2)x4 = 40. Всего получаем: 80+48+40 = 168 тактов на обсчет фильтра.

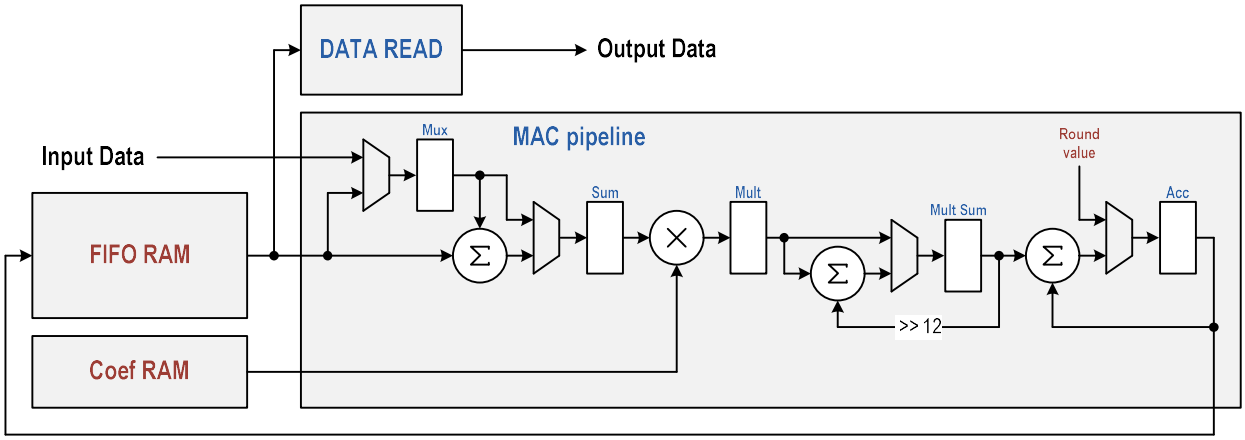
Помимо этого нужно добавить такты на чтение данных для вывода, т.к. для всех буферов FIFO используется общий блок памяти. Для x8 оверсемплинга нужно прочитать соот-но 8 отсчетов. Чтение каждого семпла занимает 1 такт. Т.е. требуется дополнительно 8 тактов.

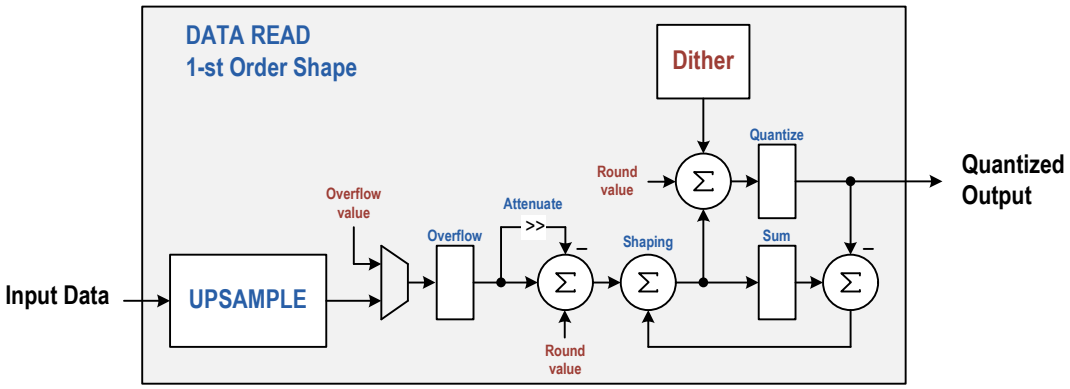
И плюс два такта нужно для аттенюации и загрузки в FIFO семпла входных данных.

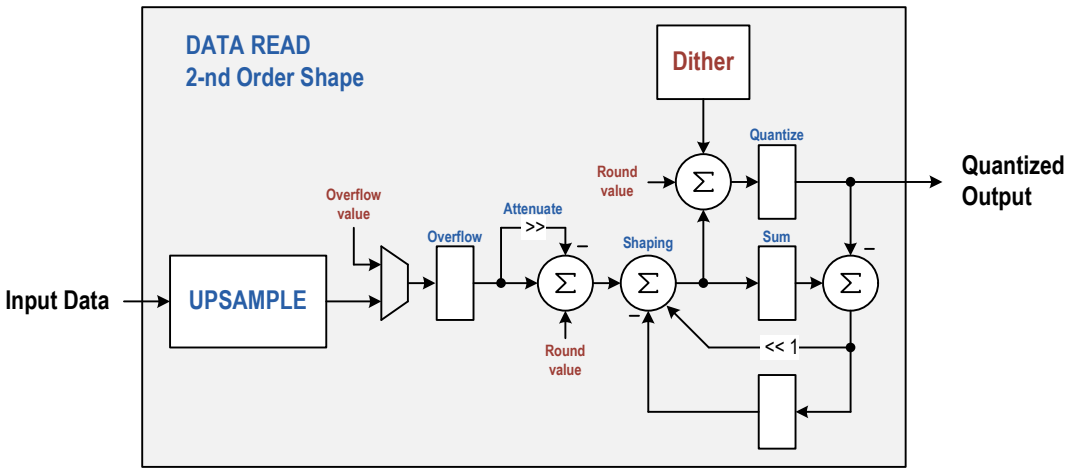
Итого получаем 168+8+2 = 178 тактов, которые легко помещаются в заданное ограничение 512 тактов.

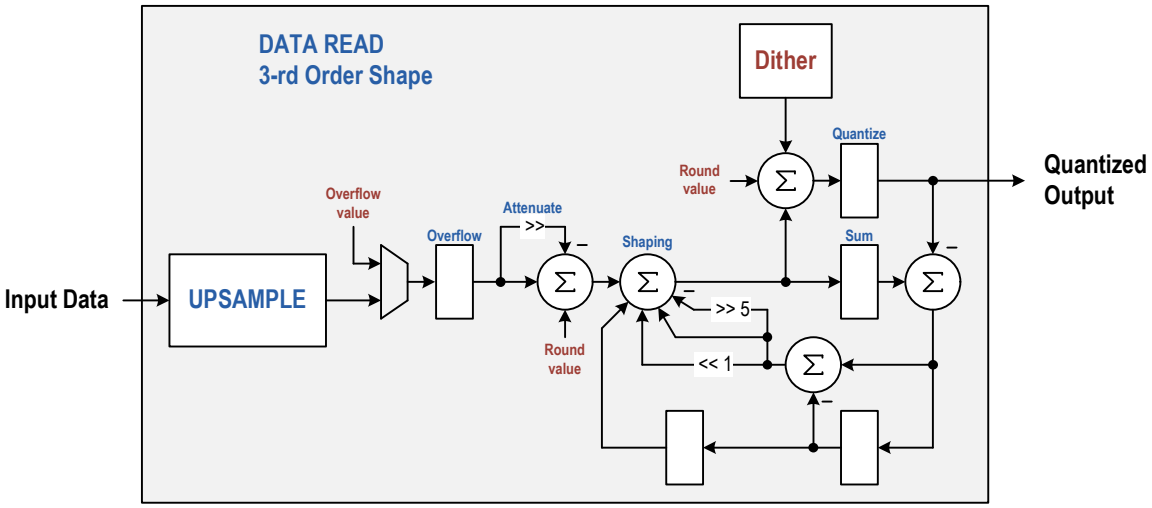
Структура DF2

Структура 5-каскадного (x32) интерполятора

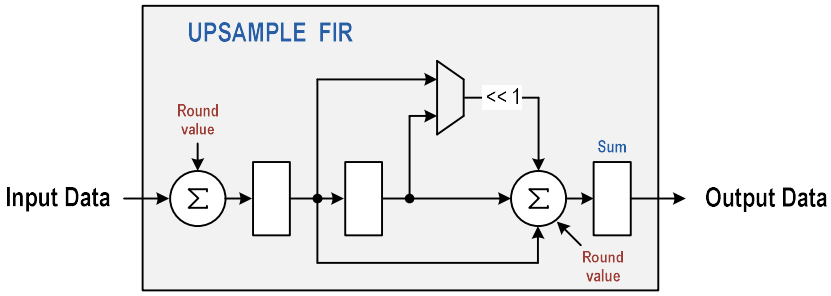








x2 FIR интерполятор, используемый для апсемплинга x64



CIC интерполятор, используемый для апсемплинга x128 и выше