**Алгоритм децимации на 20**

Предлагается используя HLS реализовать алгоритм децимации с коэффициентом 20 входного массива:

- 24576 пар слов х 2 канала (24576 пар слов в каждом канале, т.е. 24576х2 пар слов на 2 канала, т.е. на два канала всего 98304 слова. В каждом канале пара - это I и Q (т.е. комплексный сигнал) ) с плавающей запятой – массив исходных данных для двух каналов источника сигнала. Это базовый вариант размера входного массива.

- 49192 пар слов х 2 канала. Это дополнительный, для исследования, вариант размера входного массива.

- 98304 пар слов х 2 канала. Это дополнительный, для исследования, вариант размера входного массива.

Децимация выполняется методом последовательного поэлементного умножения выборки из 176 очередных элементов на предварительно рассчитанную импульсную характеристику длительностью 176 элементов (в импульсной характеристике пара - это I и Q (т.е. комплексные коэф фильтра) ). Коэффициента фильтра приложены в файле. Затем отбрасывание 19 из 20 входных отсчетов.

Выходными данными является массив из 1228х2 пар слов для двух каналов. Это для базового варианта размера входного массива.

Разрядность: входной, выходной массивы, массив коэффициентов: float (32 разряда, стандартный Си),

Импульсная характеристика симметричная – это д.б. учтено в реализации.

Размер входного массива задается define (значения определены выше), выходной считается.

Что надо сделать:

1. Написать на Си функцию, реализующую децимирующий фильтр (д.б. 2 функции: фильтр и прореживание)
2. Написать на С++ децимирующий фильтр (классы)
3. Написать тесты, проверяющие правильность работы созданных описаний
   1. Тест для проверки функции фильтра, в котором в массиве исходных данных будет только дельта функция – результат обработки, до прореживания, должен соответствовать набору коэффициентов.
   2. Тест, в котором исходные массивы считывается из памяти, результирующие массивы после прореживания сравниваются с ожидаемыми массивами (в Matlab сделать аналогичный децимирующий фильтр и для него получить входные и выходные массивы).

Тест должен запускать тестируемый дециматор несколько (как минимум 4) раза.

Тест должен позволять, при запуске на ПК: оценивать время выполнения дециматора (например time.h) и иметь возможность управлять запуском выполнения дециматора на 1 – 4х ядрах одновременно.

1. Провести тестирование для двух версий функции (описанной на Си и С++).
2. Синтезировать дециматор, описанный на Си, с настройками по умолчанию (без оптимизации). Микросхема **XCVU 125-flvc2104-3**
   1. Оптимизировать (как сможете) и дать сравнение аппаратных затрат, периода, II для базового и оптимизированного вариантов
   2. Осуществить моделирование после синтеза
3. Синтезировать дециматор, описанный на С++, с настройками по умолчанию (без оптимизации). Микросхема **XCVU 125-flvc2104-3**
   1. Оптимизировать (как сможете) и дать сравнение аппаратных затрат, периода, II для базового и оптимизированного вариантов
   2. Осуществить моделирование после синтеза
4. Привести сравнительный анализ времени выполнения функции фильтра на ПК и его аппаратной реализации.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Время выполнения  98302 отсчетов | Время выполнения  49192 отсчетов | Время выполнения  24576 отсчетов | Аппаратные затраты | период | II |
| Процессор  1 core |  |  |  | NA | NA | NA |
| Процессор  2 core |  |  |  | NA | NA | NA |
| Процессор  4 core |  |  |  | NA | NA | NA |
| Аппаратная реализация описания на Си без оптимизации |  |  |  |  |  |  |
| Аппаратная реализация описания на Си с оптимизацией |  |  |  |  |  |  |
| Аппаратная реализация описания на С++ без оптимизации |  |  |  |  |  |  |
| Аппаратная реализация описания на С++ с оптимизацией |  |  |  |  |  |  |