

**Высшая школа электроники и микросистемной техники**

**КУРСОВАЯ РАБОТА**

**Перестраиваемый фильтр (дифференцирование, интегрирование)**

по дисциплине «Автоматизированная разработка цифровых интегральных схем»

Выполнил студент 4941104/20701

\_\_\_\_\_

А. Каврук

подпись студента

Дата сдачи пояснительной записки: «\_\_8\_\_» \_\_\_\_\_ декабря \_\_\_\_\_ 2023 г.

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

Преподаватель: доцент, к. т. н.

\_\_\_\_\_

М. С. Енученко

подпись

Дата защиты: «        » декабря 2023 г.

Оценка: \_\_\_\_\_

Члены комиссии:

доцент, ВШЭиМТ

\_\_\_\_\_

М. С. Енученко

доцент, ВШЭиМТ

\_\_\_\_\_

И. М. Пятак

доцент, ВШЭиМТ

\_\_\_\_\_

М. М. Пилипко

**САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО**  
**Институт электроники и телекоммуникаций**  
**Высшая школа электроники и микросистемной техники**

**ЗАДАНИЕ**  
**на выполнение курсовой работы по дисциплине**

Автоматизированная разработка цифровых интегральных схем

студенту Кавруку Александру

группа: 4941104/20701

семестр: 3

1. Тема работы:

Перестраиваемый фильтр (дифференцирование, интегрирование)

2. Срок сдачи студентом законченной работы: 08.12.2023

3. Исходные данные по работе:

Требования и результаты, полученные в ходе выполнения практической работы предыдущего семестра по курсу «Цифровые устройства обработки сигналов», в том числе математическая модель устройства в MATLAB/Simulink.

4. Содержание работы (перечень подлежащих разработке вопросов):

Выполнить разработку цифровой интегральной схемы (ASIC), выполняющую функцию блока, разработанного в предыдущем семестре, с использование пакета программ Cadence. В частности:

1. Разработать описание блока цифровой обработки сигналов на языке Verilog HDL (синтезируемое подмножество конструкций языка) согласно выданной теме работы, а также создать все файлы, требуемые для успешного выполнения последующих этапов курсовой работы;
2. Провести компьютерное моделирование (Cadence Incisive), логический синтез (Cadence RTL Compiler), генерацию топологии (Cadence Encounter) устройства и выполнить импорт устройства в среду разработки аналоговых ИС (Cadence Virtuoso) с последующей верификацией (Cadence Assura/PVS).

Перечень обязательного сопроводительного материала:

1. Исходный код блока на языке Verilog, в том числе модуля для тестирования (моделирования);
  2. Файлы описания временных ограничений (.sdc);
  3. Сценарии (TCL файлы) для проведения синтеза и генерации топологии;
  4. Результаты компьютерного моделирования на всех этапах проектирования (как для исходного RTL кода, так и для списка межсоединений ячеек библиотеки), подтверждающие соответствие устройства своей исходной математической модели;
  5. Отчёты статического временного анализа (STA) для этапов логического синтеза и всех этапов построения топологии, отчёты прохождения проверок DRC и LVS.
- Все файлы с исходными кодами (Verilog, SDC, TCL и т. п.) должны сопровождаться комментариями.

Структура пояснительной записки:

- Введение;
- Основная часть;
- Заключение;
- Список использованных источников.

Требования к разработке:

1. Разработку проекта вести с помощью системы контроля версий Git, в качестве удаленного хранилища (репозитория) использовать <https://github.com>. Ссылка на открытый репозиторий с выполненным проектом должна быть указана в работе;
2. Критический путь устройства должен быть определен и не должен превышать по длине критический путь исходной математической модели;
3. Тактовая частота устройства должна быть не менее 20 МГц, неопределённость тактового

сигнала – не более 10%;

4. Описание на языке Verilog должно содержать в явном виде понижение/повышение частоты дискретизации, если это предусмотрено назначением и/или функциональными особенностями устройства. Допустимо предполагать, что внешние тактовые сигналы синхронные.

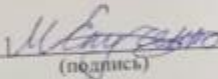
5. Перечень графического материала (с указанием обязательных рисунков):

6. Дата выдачи задания: 14.09.2023

7. Форма отчётности: пояснительная записка о курсовой работе

(данное задание прилагается к пояснительной записке)

Преподаватель

  
(подпись)

М. С. Енученко

(инициалы, фамилия)

Студент

  
(подпись)

А. Каврук

(инициалы, фамилия)

## Реферат

Отчёт 48 с., 15 рис., 1 табл., 3 источника, 8 прилож.

ЦИФРОВЫЕ УСТРОЙСТВА, ФИЛЬТРЫ, ИНТЕГРАЛЬНЫЕ СХЕМЫ, СИНТЕЗ, ВЕРИФИКАЦИЯ

Объектом исследования является автоматизация процесса проектирования цифрового устройства.

Цель работы – разработка цифровой интегральной схемы (ASIC).

В ходе работы на поведенческом уровне на языке System Verilog был описан фильтр и средствами САПР Cadence была разработана его топология. Устройство удовлетворяет критерию работоспособности на частоте 20 МГц, и полностью соответствует ранее разработанной математической модели. Также было проанализировано максимальное быстродействие схемы, данная реализация способна работать при тактовой частоте 40 МГц.

## Содержание

1	Введение .....	6
2	Основная часть .....	7
2.1	Опорная модель .....	7
2.2	Поведенческий уровень .....	7
2.3	Синтез .....	11
2.4	Топология .....	13
2.5	Верификация .....	15
3	Приложение .....	20
3.1	HDL-код упакованных параметров .....	20
3.2	HDL-код схемы .....	20
3.3	Файл с тестовыми воздействиями .....	29
3.4	Сценарий MATLAB для тестирования .....	32
3.5	Файл с ограничениями .....	39
3.6	Сценарий логического синтеза .....	40
3.7	Файл МММС .....	41
3.8	Сценарий создания топологии фильтра .....	43
	Заключение .....	48
4	Список использованных источников .....	49

# 1 Введение

В связи с развитием кремниевой технологии и математического аппарата анализа дискретных систем цифровые устройства все больше замещают аналоговую обработку. Преимущества цифровых устройств – высокая помехоустойчивость, возможность реализации достаточно сложных алгоритмов обработки сигнала (БПФ). В свою очередь для реализации сложных алгоритмов и окружающей периферии требуется создание достаточно больших интегральных схем (БИС и СБИС), которые человек не сможет собрать вручную и тем более создать топологию на кристалле. Эта задача отведена алгоритмам САПР, инженеру остается описать устройство на языках описания аппаратуры (HDL).

В данной работе предлагается пройти маршрут создания перестраиваемого фильтра (дифференцирование, интегрирование) ASIC, математическая модель которого была разработана ранее. Стадии создания маршрута проектирования велись в системе контроля версий GitHub, URL: [https://github.com/loris19905/projects/tree/main/adaptive\\_filter](https://github.com/loris19905/projects/tree/main/adaptive_filter)

## 2 Основная часть

### 2.1 Опорная модель

Ранее была разработана математическая модель фильтра с полосой 0,75 от частоты дискретизации, которая в зависимости от управляющего сигнала могла работать как в режиме дифференцирования, так и в режиме интегрирования. За основу был взят интегратор Тика [1] и дифференциатор в виде КИХ-фильтра 9-го порядка. Интегратор Тика взят был с тем расчетом, что он имеет в заданной полосе наименьшую среднеквадратичную ошибку в сравнении с идеальным интегратором. Коэффициенты КИХ-фильтра дифференциатора были получены при помощи fdatool в среде MATLAB, применяемый метод – метод Паркса-МакКлееллана.

Архитектурным решением для такой реализации является переключение коэффициентов импульсной характеристики при помощи мультиплексора по управляющему сигналу. При этом на вход части триггеров, которые находятся в сдвиговом регистре, необходимо подать нули) при работе в режиме интегратора. При работе в режиме дифференциатора данные в петле обратной связи подменяются нулями, что эквивалентно отсутствию обратной связи; при работе в режиме интегратора на последний сумматор подаются уже данные с триггеров. Схема фильтра в среде Simulink представлена на рисунке 1.

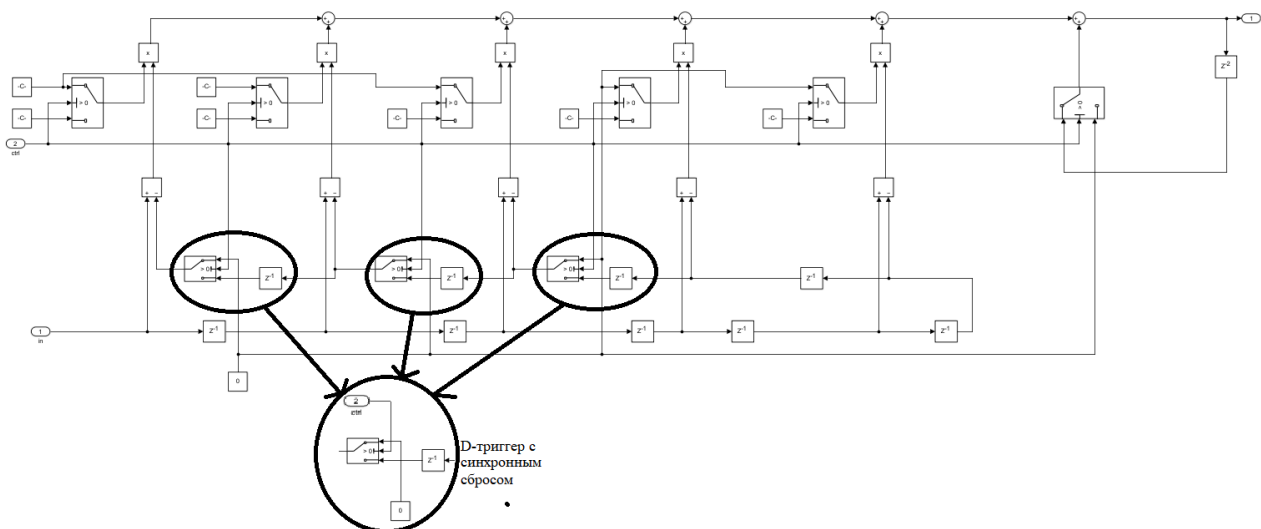


Рисунок 1 – Схема фильтра в среде Simulink

Фильтр был разработан с тем расчетом, что разрядность вычислительных блоков минимальна и в точности соответствует математической модели расширенной разрядности (64-бита с плавающей точкой) в соответствии с первоначальным ограничением по ширине данных (14 бит). Порог чувствительность фильтра в режиме интегратора варьируется от 1 МЗР (младший значащий разряд) на низких частотах до 2 МЗР на высоких частотах, в режиме дифференциатора – от 3 МЗР до 1 МЗР. Теперь математическую модель необходимо представить в виде кода на HDL.

### 2.2 Поведенческий уровень

На данном этапе производится описание фильтра на уровне регистров на одном из языков описания аппаратуры (HDL – hardware description language). Данный этап можно было реализовать при помощи встроенной утилиты HDL Coder в среде MATLAB или же самостоятельно. Ввиду отсутствия данного пакета код писался вручную. Структура кода состоит из описательной части (RTL) на языке System Verilog и файла-пакета с объявлением вспомогательных параметров таких как ширина дробной части и всего слова на выходе вычислительных блоков, количество коэффициентов импульсной характеристик КИХ-фильтра, количество «уникальных» коэффициентов интегратора. Для вычислительных блоков таких как вычитание, суммирование и умножение реализовано округление к ближайшему, точно так же как в самой модели. Файл-пакет с параметрами можно найти в репозитории и в приложении 3.1, файл с описанием фильтра представлен в приложении.

В исходном коде фильтра имеется как синтезируемая, так и несинтезируемая части, которая используется для проверки работы операции умножения и вычитания. Несинтезируемая часть генерируется только, если включен параметр SIM\_EN, во время синтеза данный параметр принимает значение 0 по умолчанию. К описанию фильтра также добавлен файл для верификации с набором тестовых воздействий (testbench), в котором происходит считывание данных модели и проверяется побитовое соответствие с выходом написанного модуля; в консоль выводится результат проверки каждого отсчета. Дополнительно для проверки в MATLAB записывается файл с выходными данными. В приложении 3.3 вынесен testbench. Дополнительно через параметр INCLUDE\_SDF вынесена возможность запуска симуляции с учетом паразитных параметров, которые хранятся в файле .sdf.

Также был написан сценарий в MATLAB, в котором можно задать параметры входного сигнала и режим работы фильтра, далее автоматически будет запущена математическая модель, и будет записан файл с входным воздействием и откликом.

Таким образом, на рисунке 2 изображен результат разработки (elaboration) в САПР Vivado, на рисунке 3 показан выходной сигнал в симуляторе Questa Advanced Simulator. Из диаграммы видна подача входных данных на порт s\_tdata, правильно считанные данные подчеркиваются сигналов s\_tvalid; аналогично мы видим данные на выходе фильтра по сигналу m\_tdata, правильные данные подчеркиваются сигналом m\_tvalid; в testbench также производится чтение файла, в котором записан выходной сигнал математической модели, и сбор этих данных в память, по которой в дальнейшем будут вытаскиваться значения по счетчику. Каждый раз при поступлении m\_tvalid счетчик инкрементируется.

Если выходная данные фильтра, подчеркнутая сигналом m\_tvalid, совпала с заданным значением математической модели, то сигналу output\_is\_valid\_to\_model присваивается значение 1. По окончании приема всех данных с выхода фильтра поднимается сигнал finish\_data\_transfer, по этому сигналу симуляция останавливается. Дополнительно в консоли можно увидеть отладочные сообщения с указанием номера данных и результатом проверки.



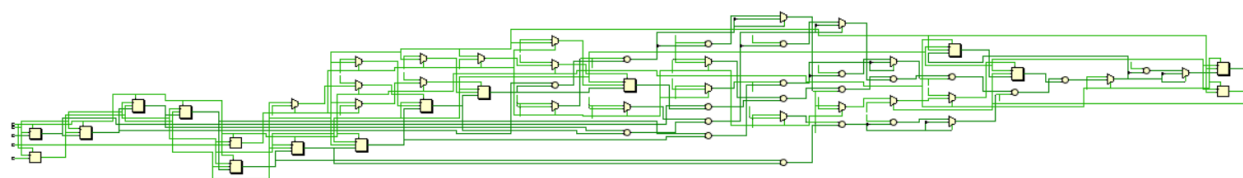


Рисунок 2 – Сгенерированная схема из Vivado

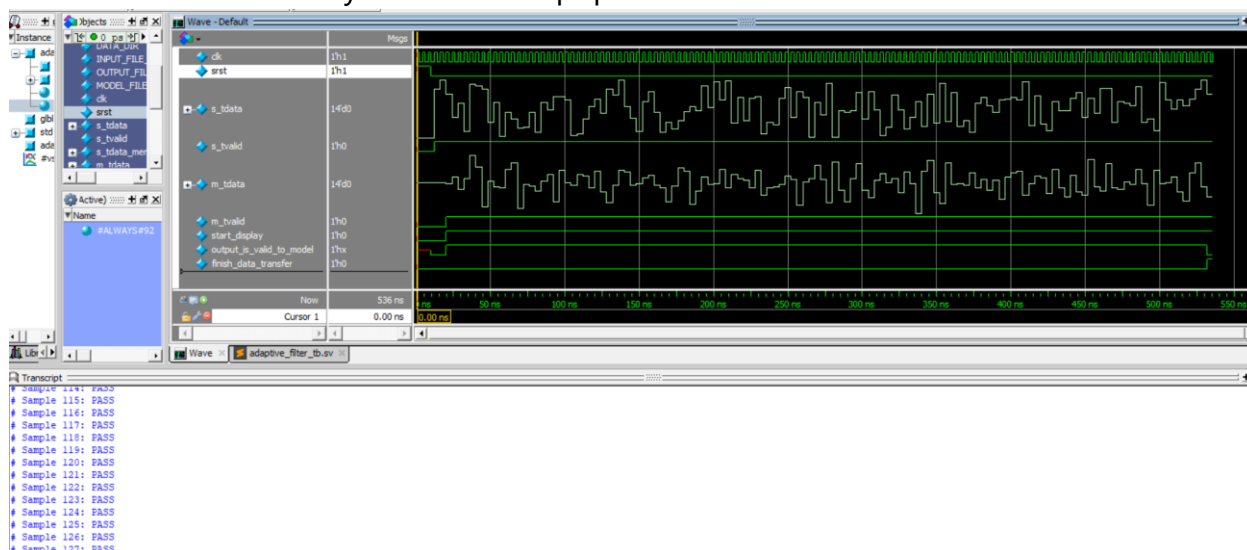


Рисунок 3 – Результат моделирования в Questa Advanced Simulator

Также можно проверить написанный модуль в MATLAB: задав все необходимые параметры, необходимо запустить сценарий; далее в консоли MATLAB появится сообщение «Запустите симуляцию»; после запуска в Questa Advanced Simulator и его завершения, нажав на произвольную кнопку, запустится проверка с выходов вычитания, перемножения и самого модуля. Код сценария представлен в приложении 3.4. На рисунке 4 – результат проверки в среде MATLAB.

```
Данные с выхода блоков вычитания совпали  
Данные с выхода умножителей совпали  
Simulink-модель совпала с RTL  
>>
```

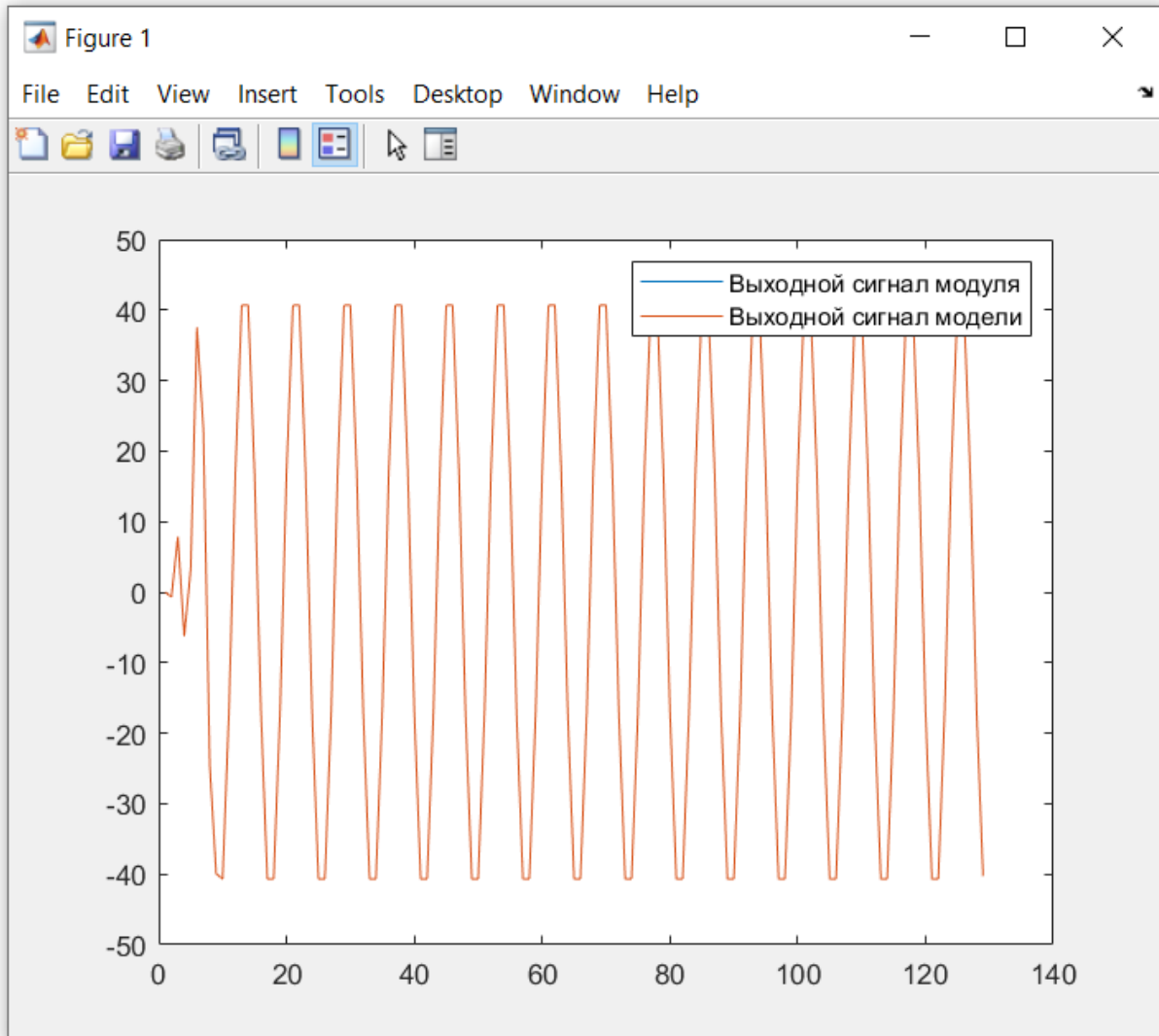


Рисунок 4 – Результат проверки в среде MATLAB

Аналогичный процесс моделирования можно сделать, воспользовавшись САПР Cadence Incisive. Загрузив исходные файлы проекта и testbench, проведем моделирование. Результаты представлены на рисунке 5. Для наглядности выведены аналогичные сигналы, по которым видно, что RTL соответствует математической модели.

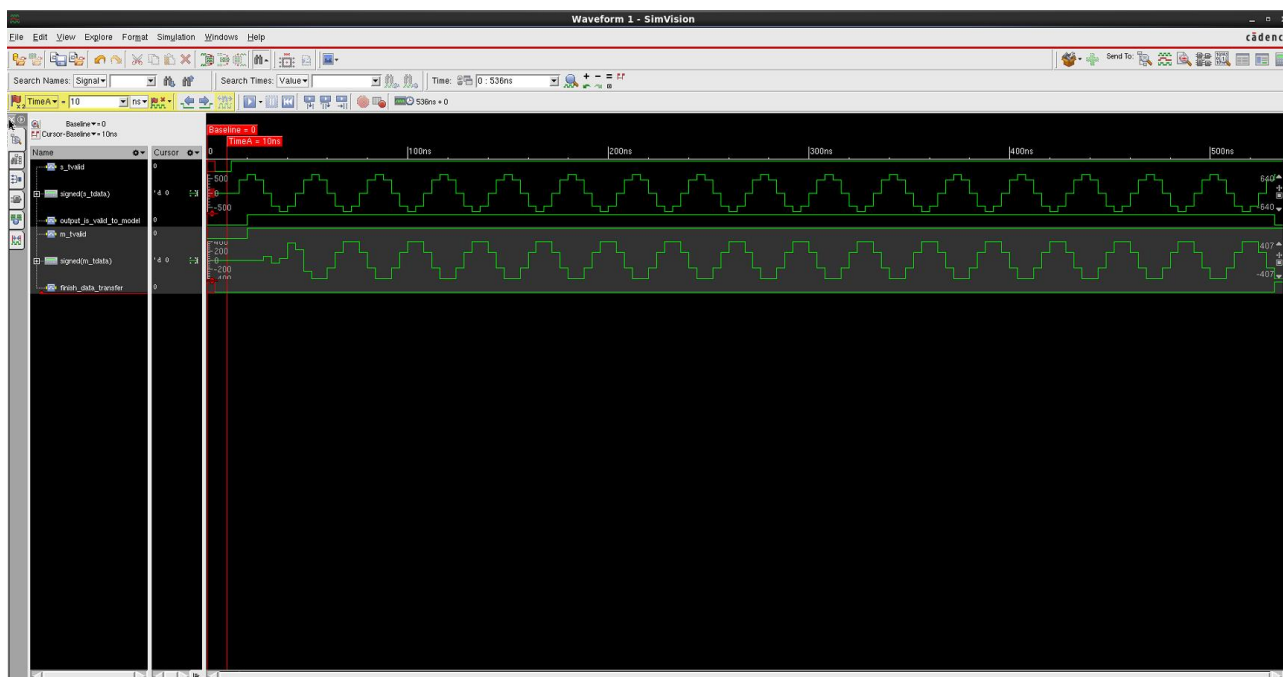


Рисунок 5 – Результат моделирования в Incisive до логического синтеза

### 2.3 Синтез

Закончив с разработкой RTL, можем перейти уже к физической реализации схемы. На первом шаге необходимо создать файл временных ограничений (.sdc). В качестве команд использовались [2]:

- `create_clock`: объявление тактовой частоты, по заданию тактовая частота равна 20 МГц, но ,чтобы определить максимальное быстродействие, мы ее определим равной 40 МГц; сама команда в качестве аргумента принимает период дискретизации, поэтому запишем 25 нс.

- `set_input_delay`: установление времени прибытия данных относительно возрастающего фронта тактового сигнала. В рамках задания выбрали величину равной 0,5 от периода тактового сигнала.

- `set_output_delay`: установление времени прибытия данных к данному выходному порту относительно возрастающего фронта тактового сигнала. В рамках задания выбрали величину равной 0,5 от периода тактового сигнала.

- `set_clock_uncertainty`: команда позволяет задать колебание тактового сигнала или задержку между тактовыми сигналами. Величина неоднозначности была задана равной 0,25 нс, считая, что нестабильность периода в 1% вероятна.

- `set_load`: при помощи данной команды можно задать выходную емкостную нагрузку устройства; величину нагрузки приняли равной 0,5 пФ.

После того как мы создали файл ограничений и RTL можно перейти к этапу логического синтеза. В результате синтеза мы должны получить список соединений (netlist), в котором производится замена «абстрактных» умножителей, сумматоров, триггеров на их реализации при помощи элементарных IP-ячеек библиотеки; и отчет о STA и оценке занимаемой площади. Если этот запас отрицательный, то в таком случае вероятность появления метастабильности [3] в ходе работы схемы велика. Если такое

происходит, то необходимо внести архитектурные изменения, которые бы уменьшили критический путь, либо, если это возможно, произвести конвейеризацию. Существует еще третий вариант – ослабление накладываемых ограничений, пересмотр путей данных (объявление т.н. false path, через которые данные в ходе работы идти не будут) или уменьшение тактовой частоты, что может противоречить изначальному техническому заданию.

Целью синтеза является получение netlist с положительным запасом по времени для наихудшего случая (corner): p- и n-транзисторы «медленные», температура 175 градусов, рабочее напряжение 1,62 В при номинальном 1,8 В. Также добавляются файлы, которые содержат информацию о блоках библиотеки (имя, геометрия, информация о портах, их направление и т.д.) (.lef), информацию о паразитных RC-элементах для расчета задержки (.captbl).

По заданию использовалась библиотека D\_CELL\_HD по технологии компании X-FAB. Файл ограничений представлен в приложении 3.5, сценарий запуска логического синтеза – в приложении 3.6. Отчет по STA показан на рисунке 6, оценка занимаемой площади – на рисунке 7.

```
Generated by:      Encounter(R) RTL Compiler RC14.27 - v14.20-s064_1
Generated on:      Nov 25 2022  02:38:13 pm
Module:           adaptive_filter
Technology library: D_CELLS_HD_LP5MOS_slow_1_62V_175C 4.0.0
Operating conditions: slow_1_62V_175C
Interconnect mode: global
Area mode:        physical library
=====
Cost Group   : 'clk' (path_group 'clk')
Timing slack :      10ps
Start-point  : mult_coeff_1_reg[2]/C
End-point    : m_tdata_reg[13]/D
```

Рисунок 6 – Результат STA после синтеза

Generated by: Encounter(R) RTL Compiler RC14.27 - v14.20-s064_1				
Generated on: Nov 25 2022 02:38:13 pm				
Module: adaptive_filter				
Technology library: D_CELLS_HD_LP5MOS_slow_1_62V_175C 4.0.0				
Operating conditions: slow_1_62V_175C				
Interconnect mode: global				
Area mode: physical library				
Instance	Cells	Cell Area	Net Area	Total Area
adaptive_filter	1511	40582	20056	60639
mul_186_52	188	4714	2116	6830
mul_190_52	167	4461	1790	6250
csa_tree_add_208_53_groupi	99	4335	1186	5521
mul_184_52	121	3841	1360	5201
mul_198_52	74	1701	789	2490
mul_194_52	73	1651	738	2389
sub_180_59_I2	68	953	532	1486
sub_178_56_I1	65	941	507	1448
sub_180_59_I4	58	1061	376	1437
sub_180_59_I5	58	1064	368	1432
sub_180_59_I3	54	1041	335	1376
inc_ADD_UNSP_OP_2	19	517	133	650

Рисунок 7 – Оценка занимаемой площади

Из результатов, представленных на рисунке 6, можно обнаружить, что критический путь в результате логического синтеза получился короче предполагаемого. На рисунке 8 черным цветом выделен предполагаемый путь, красным – путь в результате синтеза.

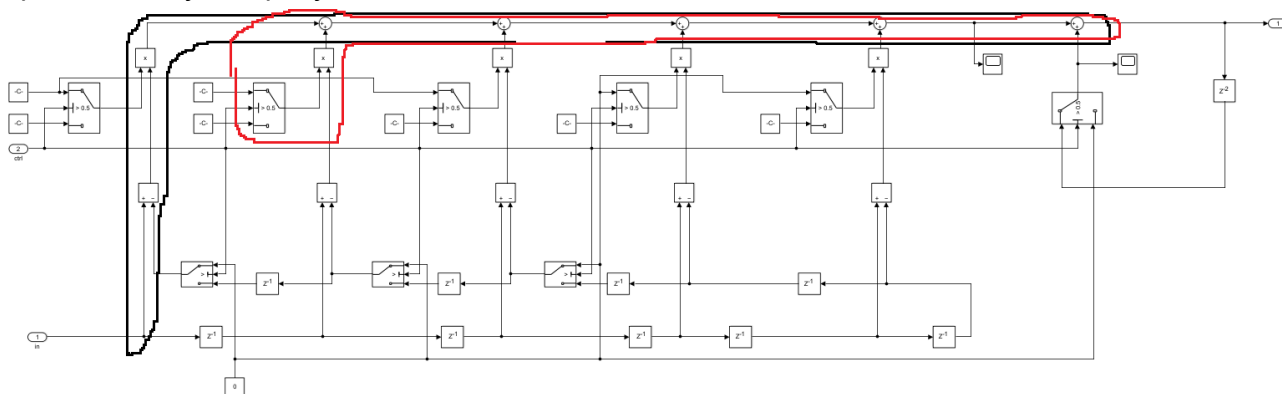


Рисунок 8 – Сравнение предполагаемого и полученного в результате синтеза критических путей

## 2.4 Топология

Следующий шаг – размещение на кристалле и построение топологии. Для выполнения данного шага необходимо проделать серию операций: объявление портов и их расположение, шин земли и питания, размещение блоков, построение тактового дерева, соединение элементов путем наложения слоев металлизации, конечная проверка STA, DRC, геометрии и соединений. Данные операции реализуются автоматически при помощи сценария.

Порты объявляются в отдельном файле и импортируются в Encounter. Помимо портов, на начальных этапах обозначают аспектное соотношение (отношение длины к ширине), плотность размещения, которую лучше не ставить слишком большой (не больше 0,7) для упрощения задачи синтезатору, и размер силовых шин (шин земли и шин питания).

После этого осуществляется первичное размещение блоков, после – детальное размещение с анализом возможных конфликтов при наложении металлизации и с оценкой по setup.

После размещения блоков производится построение тактового дерева. Глобальные задачи при построении дерева – минимизировать задержку относительно тактового источника (clock latency), минимизировать разность времени прихода тактового сигнала на триггеры (clock skew), минимизировать наводки, устранить возможные нарушения STA. Минимизация задержек решается путем построения регулярных структур, выравнивание по времени прихода – за счет буферов; снижение влияния наводок – увеличение расстояния между проводниками.

Следующий шаг – наложение слоев металлизации. Аналогично этапу размещения блоков производится глобальное соединение элементов и детальное. Главная цель – произвести соединение элементов в соответствии с DRC, а уже потом – так реализовать трассировку, чтобы удовлетворить STA.

На финальном шаге – проверка STA, проверка по DRC и по соединениям, извлечение паразитных компонент (signoff). Файл с подключением библиотек для построения топологии по принципу многорежимного анализа с множеством технологических углов (MMMC – multy-mode multy-corner) представлен в приложении 3.7, сценарий запуска процесса создания фильтра на кристалле – в приложении 3.8.

В результате получили файл с топологией (.def), netlist логического и физического уровней, файл с паразитными компонентами (.sdf), который в дальнейшем может быть применен, если не выполнены условия по STA и приходится заново проходить весь маршрут проектирования. На рисунке 9 изображена топология в Cadence Encounter. Размер кристалла составил 307,7 на 303,0 мкм, площадь кристалл – 0,09 мм<sup>2</sup>.



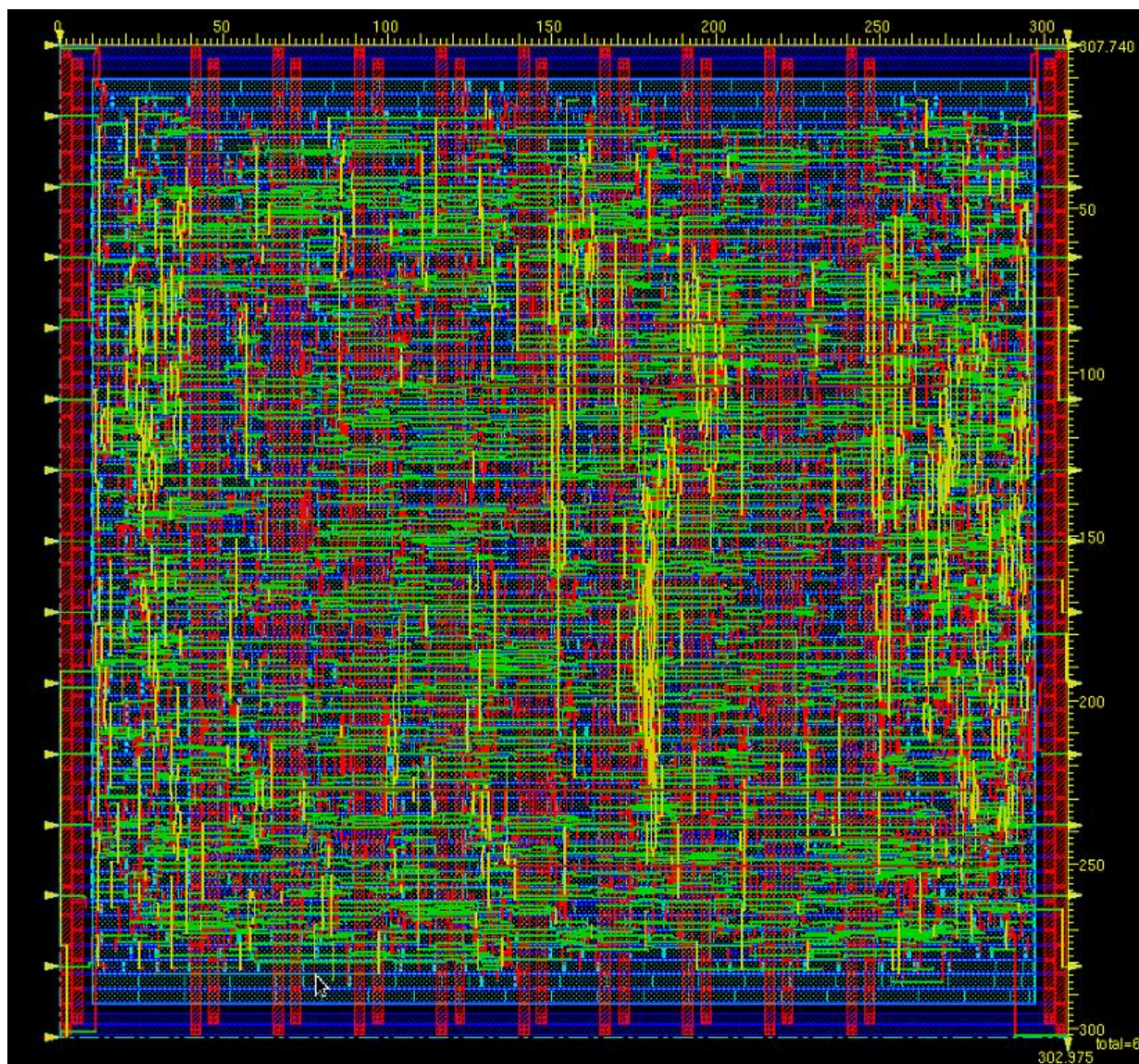


Рисунок 9 – Топология фильтра

В таблице 1 представлено изменение таймингов в ходе каждого из этапов создания топологии. Запас по времени оказался положительным, таким образом, максимальная частота работы фильтра без дополнительной конвейеризации составила 40 МГц при целевых 20 МГц.

Таблица 1 – Тайминги при синтезе схемы с тактовой частотой 40 МГц

Этап имплементации	Запас по setup, нс	Запас по hold, нс
Размещение модулей	4,300	---
Первичный синтез тактового дерева	1,709	0,049
Финальный синтез тактового дерева	0,164	0,065
Трассировка	0,213	0,033
Финальная проверка	0,025	0,033

## 2.5 Верификация

После этапа синтеза необходимо загрузить полученный «физический» netlist в Cadence Virtuoso и запустить проверки DRC и LVS. Чтобы пройти LVS необходимо

предварительно проставить наименования портов фильтра. Результат DRC представлен на рисунке 10. На рисунке 11 показан результат прохождения LVS.



Рисунок 10 – Результат прохождения DRC





Рисунок 11 – Результат прохождения LVS

Следующий этап верификации – проверка сходимости данных при симуляции netlist после логического синтеза и после получения топологии с математической моделью фильтра. Для проверки логического netlist логического синтеза необходимо подключить к симуляции примитивы библиотеки. Результаты симуляции представлен на рисунке 12 и 13 в режиме дифференциатора и в режиме интегратора соответственно. Для моделирования в режиме дифференциатора параметру FILTER\_MODE необходимо присвоить 0, в режиме интегратора – 1. Как видим, результаты симуляции полностью совпадает с результатами модели.

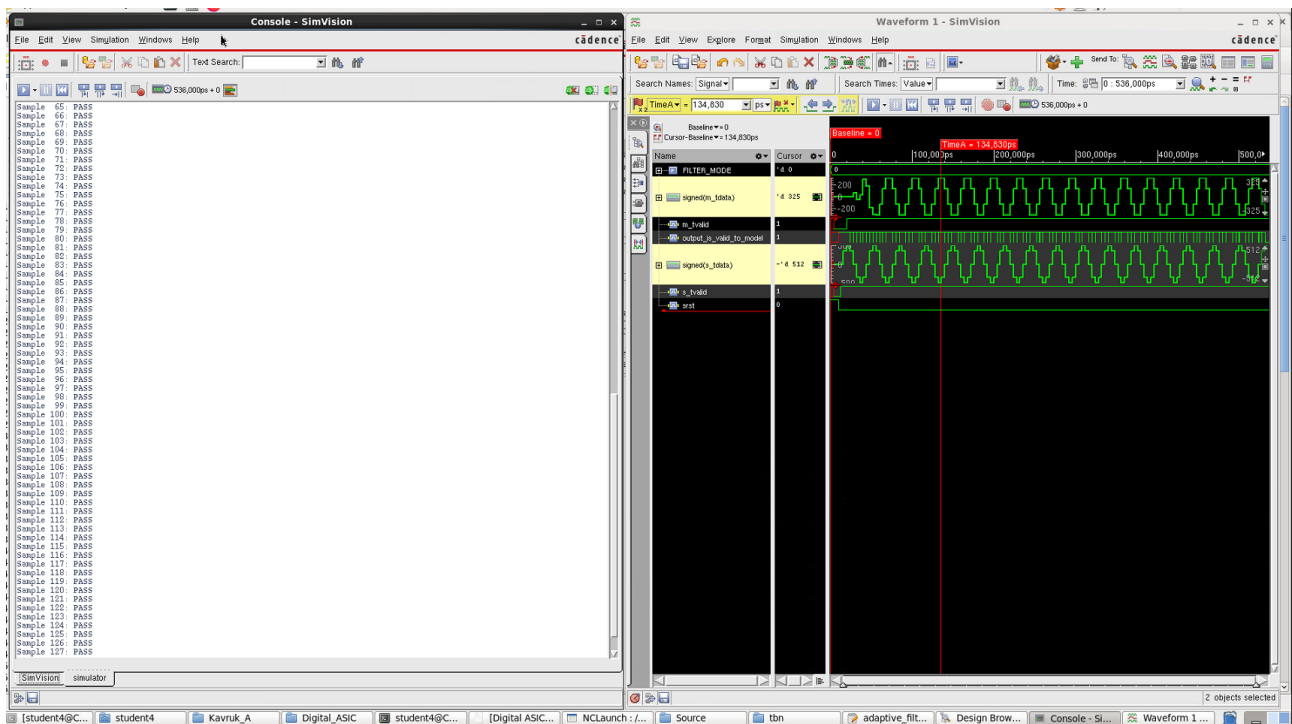


Рисунок 12 – Результат симуляции netlist после логического синтеза в режиме дифференциатора

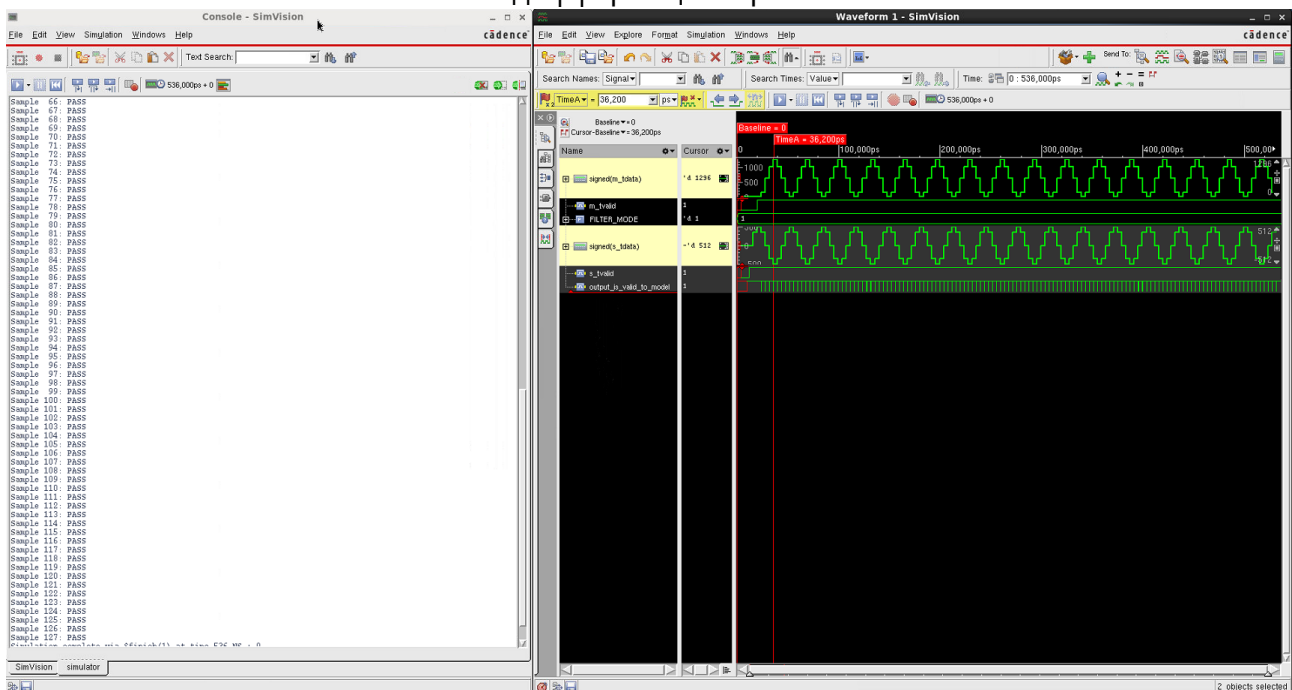


Рисунок 13 – Результат симуляции netlist после логического синтеза в режиме интегратора

На рисунках 14 и 15 представлены результаты моделирования netlist после получения топологии. В тестовом окружении необходимо присвоить параметру INCLUDE\_SDF значение 1. Как можно заметить из сообщений симулятора, в ходе компиляции возникает ошибка, связанная с тем, что модуль создан без параметра SIM\_EN, поэтому данный параметр надо убрать из тестового окружения до компиляции. Результаты симуляции полностью совпали с моделью.

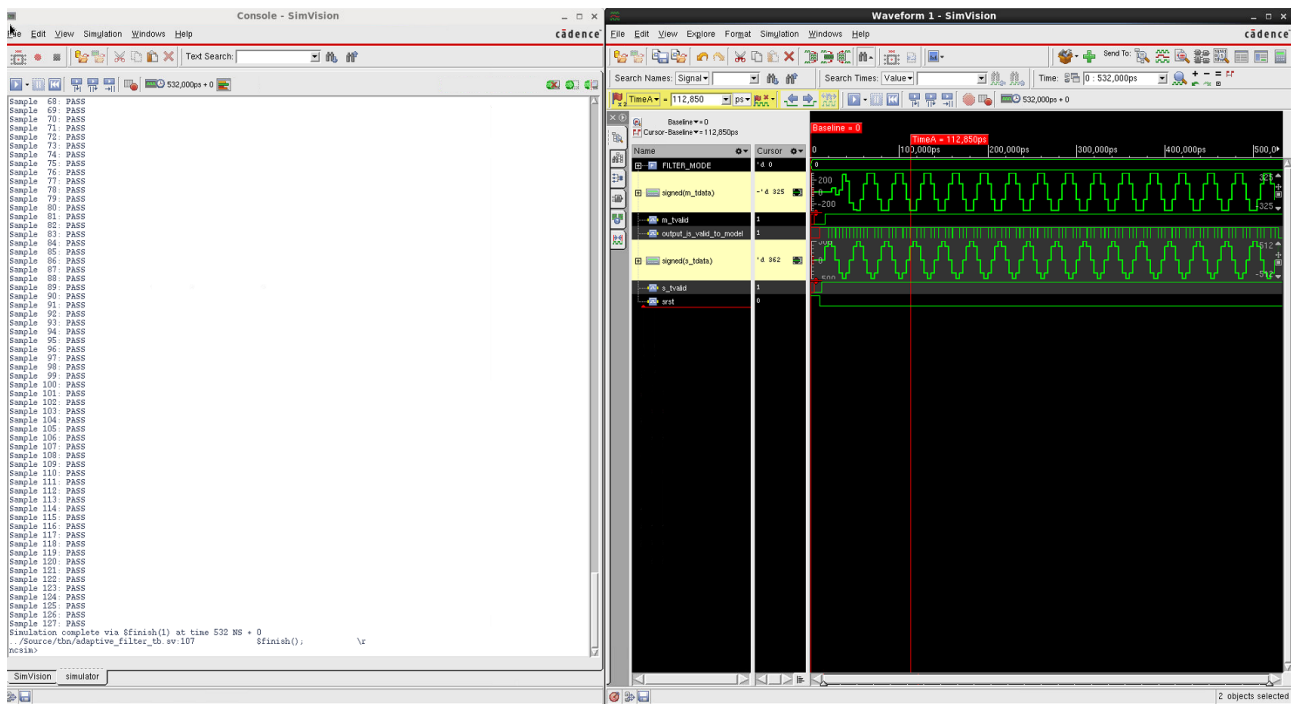


Рисунок 14 – Результат симуляции netlist после получения топологии в режиме дифференциатора

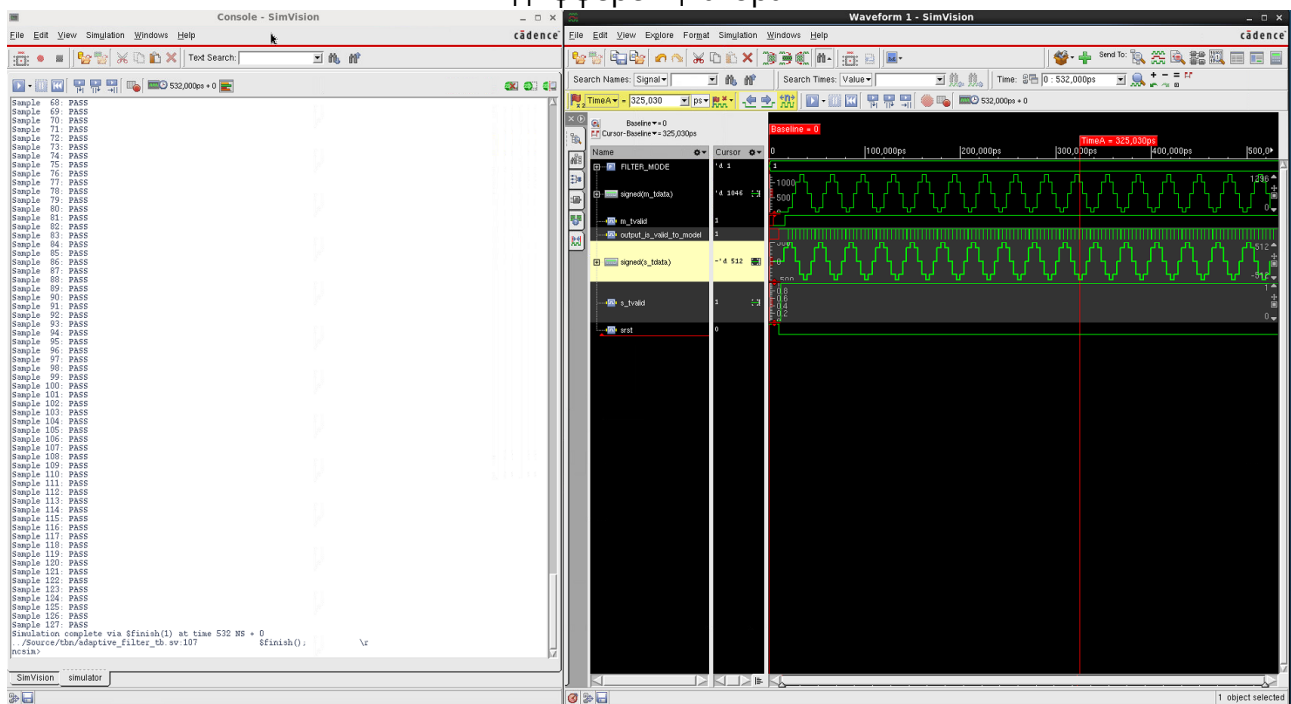


Рисунок 15 – Результат симуляции netlist после получения топологии в режиме интегратора

## 3 Приложение

### 3.1 HDL-код упакованных параметров

```
//Project name: "Adaptive filter"
//Author:      Kavruk A.
//File description: Package for RTL parameters

package adaptive_filter_pkg;

    localparam WORDLENGTH      = 14;
    localparam FRACTIONAL_LENGTH = 6;

    localparam FIR_DIFF_ORDER   = 9;
    localparam FIR_DIFF_COEFF_NUM = (FIR_DIFF_ORDER + 1) >> 1;

    parameter int DIFF_COEFF_WL [FIR_DIFF_COEFF_NUM-1:0] = '{7, 6, 9,
9, 8};
    parameter int DIFF_COEFF_FL [FIR_DIFF_COEFF_NUM-1:0] = '{5, 4, 7,
7, 6};

    localparam INTEGR_COEFF_NUM = 2;
    parameter int INTEGR_COEFF_WL [INTEGR_COEFF_NUM-1:0] = '{6, 7};
    parameter int INTEGR_COEFF_FL [INTEGR_COEFF_NUM-1:0] = '{5, 6};

    parameter int MULTYPLYERS_WL [FIR_DIFF_COEFF_NUM-1:0] = '{14, 14,
20, 19, 18};
    parameter int MULTYPLYERS_FL [FIR_DIFF_COEFF_NUM-1:0] = '{6 , 6 ,
12, 11, 12};
    localparam OP_SUMM_WL = 20;
    localparam OP_SUMM_FL = 12;

    localparam OP_DIFF_WL = 15;
    localparam OP_DIFF_FL = 6;

endpackage : adaptive_filter_pkg
```

### 3.2 HDL-код схемы

```
`timescale 1ns / 1ns

//Project name: "Adaptive filter"
//Author:      Kavruk A.
//File description: RTL

module adaptive_filter
    import adaptive_filter_pkg::*;
    #(SIM_EN = 0
    )(
        input  logic      clk,
        input  logic      srst,
        input  logic      ctrl,

        input  logic [13:0] s_tdata,
        input  logic      s_tvalid,

        output logic [13:0] m_tdata,
        output logic      m_tvalid
```

```

);

localparam REG_ZERO_START_ADDR = 6;
localparam DELAY_FEEDBACK_LOOP = 2;

//Коэффициенты импульсной характеристики дифференциатора
logic [DIFF_COEFF_WL[0]-1:0] fir_diff_coeff_a0;
logic [DIFF_COEFF_WL[1]-1:0] fir_diff_coeff_a1;
logic [DIFF_COEFF_WL[2]-1:0] fir_diff_coeff_a2;
logic [DIFF_COEFF_WL[3]-1:0] fir_diff_coeff_a3;
logic [DIFF_COEFF_WL[4]-1:0] fir_diff_coeff_a4;

assign fir_diff_coeff_a0 = 8'hFF;
assign fir_diff_coeff_a1 = 9'h019;
assign fir_diff_coeff_a2 = 9'h1CD;
assign fir_diff_coeff_a3 = 6'h07;
assign fir_diff_coeff_a4 = 7'h13;

//Коэффициенты импульсной характеристики интегратора
logic [INTEGR_COEFF_WL[0]-1:0] fir_integr_coeff_a0;
logic [INTEGR_COEFF_WL[1]-1:0] fir_integr_coeff_a1;

assign fir_integr_coeff_a0 = 7'h17;
assign fir_integr_coeff_a1 = 6'h29;

//Объявление регистровых переменных
logic [FIR_DIFF_ORDER-1:0] [WORDLENGTH-1:0] s_tdata_d;
logic [DELAY_FEEDBACK_LOOP-1:0] [OP_SUMM_WL-1:0] loop_tdata;
logic s_tvalid_d;

logic [WORDLENGTH-1:0] s_tdata_reg;
logic s_tvalid_reg;
logic ctrl_reg;

logic [FIR_DIFF_COEFF_NUM-1:0] [OP_SUMM_WL-1:0] summ_res;

//Зашелкивание входных данных
always_ff @(posedge clk) begin
    if (srst) begin
        s_tdata_reg <= '0;
        s_tvalid_reg <= '0;
        ctrl_reg <= '0;
    end else begin
        s_tdata_reg <= s_tdata;
        s_tvalid_reg <= s_tvalid;
        ctrl_reg <= ctrl;
    end
end

//Сдвиговый регистр. Начиная с регистра REG_ZERO_START_ADDR на
вход подаются нули при работе в режиме интегратора
always_ff @(posedge clk) begin
    if (srst) begin
        s_tdata_d <= '0;
    end else begin
        for (int i = 0; i < FIR_DIFF_ORDER; i++) begin
            if (i == 0) begin

```

```

s_tdata_d[i] <= (s_tvalid_reg) ? s_tdata_reg :
s_tdata_d[i];
    end else begin
        if (ctrl_reg) begin
            if (s_tvalid_reg) begin
                s_tdata_d[i] <= (i >=
REG_ZERO_START_ADDR) ? '0 : s_tdata_d[i-1];
            end else begin
                s_tdata_d[i] <= s_tdata_d[i];
            end
        end else begin
            s_tdata_d[i] <= (s_tvalid_reg) ?
s_tdata_d[i-1] : s_tdata_d[i];
        end
    end
end
end
end

//Защелкивание выходного потока данных в регистры с
одновременным приведением поступающих данных к ширине выходных данных +
округление
//В этом же блоке осуществляется передача данных по петле
обратной связи
always_ff @(posedge clk) begin
    if (srst) begin
        loop_tdata <= '0;
        m_tdata <= '0;
        s_tvalid_d <= '0;
        m_tvalid <= '0;
    end else begin
        if (s_tvalid_reg) begin
            loop_tdata[0] <= summ_res[FIR_DIFF_COEFF_NUM-1];
            m_tdata <= (summ_res[4][OP_SUMM_FL-
FRACTIONAL_LENGTH-1]) ? summ_res[4][WORDLENGTH+OP_SUMM_FL-
FRACTIONAL_LENGTH-1:OP_SUMM_FL-FRACTIONAL_LENGTH] + 1 :
summ_res[4][WORDLENGTH+OP_SUMM_FL-FRACTIONAL_LENGTH-1:OP_SUMM_FL-
FRACTIONAL_LENGTH];
        end else begin
            loop_tdata[0] <= loop_tdata[0];
            m_tdata <= m_tdata;
        end

        if (s_tvalid_d) begin
            loop_tdata[DELAY_FEEDBACK_LOOP-1] <= loop_tdata[0];
        end else begin
            loop_tdata[DELAY_FEEDBACK_LOOP-1] <=
loop_tdata[DELAY_FEEDBACK_LOOP-1];
        end

        m_tvalid <= s_tvalid_reg;
        s_tvalid_d <= s_tvalid_reg;
    end
end

/*

```

По приходящему сигналу ctrl происходит переключение между коэффициентами интегратора и дифференциатора

Блок умножения имеет фиксированный размер для поступающих операндов, поэтому ширина mult\_coeff\_i определяется наибольшей шириной двух коэффициентов, которые могут прийти на i-ый блок умножения

В данном случае ширина коэффициентов интегратора оказалась меньше, чем ширина коэффициентов дифференциатора.

Добавив справа "0", сами коэффициенты не поменяются, и при этом будут удовлетворять заявленной ширине операнда умножителя.

Дополнительный "0" слева - знаковый бит

```

*/

logic [DIFF_COEFF_WL[0]-1:0]          mult_coeff_0;
logic [DIFF_COEFF_WL[1]-1:0]          mult_coeff_1;
logic [DIFF_COEFF_WL[2]-1:0]          mult_coeff_2;
logic [DIFF_COEFF_WL[3]-1:0]          mult_coeff_3;
logic [DIFF_COEFF_WL[4]-1:0]          mult_coeff_4;

logic [DIFF_COEFF_FL[1]-INTEGR_COEFF_FL[1]-1:0]
zeros_coeff_mult_1;
logic [DIFF_COEFF_FL[2]-INTEGR_COEFF_FL[0]-1:0]
zeros_coeff_mult_2;

assign zeros_coeff_mult_1 = '0;
assign zeros_coeff_mult_2 = '0;

always_ff @(posedge clk) begin
    if (ctrl) begin
        mult_coeff_0 <= {'0, fir_integr_coeff_a0};
        mult_coeff_1  <= {'0,   fir_integr_coeff_a1,
zeros_coeff_mult_1};
        mult_coeff_2  <= {'0,   fir_integr_coeff_a0,
zeros_coeff_mult_2};
        mult_coeff_3 <= '0;
        mult_coeff_4 <= '0;
    end else begin
        mult_coeff_0 <= fir_diff_coeff_a0;
        mult_coeff_1 <= fir_diff_coeff_a1;
        mult_coeff_2 <= fir_diff_coeff_a2;
        mult_coeff_3 <= fir_diff_coeff_a3;
        mult_coeff_4 <= fir_diff_coeff_a4;
    end
end

//Вычисление результатов блоков вычитания, умножения и сложения
logic [FIR_DIFF_COEFF_NUM-1:0][OP_DIFF_WL-OP_DIFF_FL-1:-
OP_DIFF_FL] diff_res;

logic [MULTIPLYERS_WL[0]-1:0]
mult_res_0;
logic [MULTIPLYERS_WL[1]-1:0]
mult_res_1;
logic [MULTIPLYERS_WL[2]-1:0]
mult_res_2;
logic [MULTIPLYERS_WL[3]-1:0]
mult_res_3;

```

```

        logic                                [MULTIPLYERS_WL[4]-1:0]
mult_res_4;

        logic                                [OP_DIFF_WL+DIFF_COEFF_WL[1]-1:0]
mult_res_tmp_1;
        logic                                [OP_DIFF_WL+DIFF_COEFF_WL[2]-1:0]
mult_res_tmp_2;
        logic                                [OP_DIFF_WL+DIFF_COEFF_WL[3]-1:0]
mult_res_tmp_3;
        logic                                [OP_DIFF_WL+DIFF_COEFF_WL[4]-1:0]
mult_res_tmp_4;

        logic                                [MULTIPLYERS_FL[0]-MULTIPLYERS_FL[1]-1:0]
zeros_summ_res_0;
        logic                                [OP_SUMM_FL-MULTIPLYERS_FL[3]-1:0]
zeros_summ_res_2;
        logic                                [OP_SUMM_FL-MULTIPLYERS_FL[4]-1:0]
zeros_summ_res_3;

        logic                                [OP_SUMM_WL-1:0]
feedback_operand;

        assign zeros_summ_res_0 = '0;
        assign zeros_summ_res_2 = '0;
        assign zeros_summ_res_3 = '0;

/*
    При описании блока использовались встроенные функции Verilog
    $signed(), который учитывает при вычислении смену знака.
    При вычислении использовалось округление к ближайшему.
    УМНОЖЕНИЕ:
    Промежуточный результат рассчитывался с полной разрядностью
    (например, первый операнд имеет ширину W1,
    второй W2, тогда выходной результат - W1+W2). Конечный
    результат - результат округления промежуточного результата к заявленной
    ширине. Ход округления:
    -- расчет в полной разрядности,
    -- определяем диапазон бит, который необходимо взять из
    выходного результата, ориентируясь на то, что все-таки число дробное
    -- определение величины бита (результата полной
    разрядности), который располагается относительно фиксированной точки в
    позиции (-FRACTIONAL_LENGTH-1)
    -- если бит = "1", прибавляем к усеченному результату "1",
    если нет - выходной результат вычислительного блока равен усеченному
    результату полной разрядности
    СУММИРОВАНИЕ:
    При суммировании также необходимо помнить, что первоначально
    числа дробные, т.е. целую часть необходимо складывать с целой, дробную -
    с дробной. Соответственно,
    если дробные части не сходятся, то операнд с наименьшей
    дробной частью необходимо справа дополнить нулями
*/
    always_comb begin
        if (srst) begin
            mult_res_0      = '0;
            mult_res_1      = '0;

```



```

        mult_res_2      = '0;
        mult_res_3      = '0;
        mult_res_4      = '0;
        summ_res        = '0;
        mult_res_tmp_1   = '0;
        mult_res_tmp_2   = '0;
        mult_res_tmp_3   = '0;
        mult_res_tmp_4   = '0;
        feedback_operand = '0;
    end else begin
        for (int i = 0; i < FIR_DIFF_COEFF_NUM; i++) begin
            if (i == 0) begin
                diff_res[i]      =      $signed(s_tdata_reg)      -
$signed(s_tdata_d[FIR_DIFF_ORDER-1]);
            end else begin
                diff_res[i]      =      $signed(s_tdata_d[i-1])    -
$signed(s_tdata_d[FIR_DIFF_ORDER-i-1]);
            end
        end

        mult_res_0          =      $signed(mult_coeff_0)          *
$signed(diff_res[0]);

        mult_res_tmp_1      =      $signed(mult_coeff_1)          *
$signed(diff_res[1]);
        mult_res_1          =
(mult_res_tmp_1[DIFF_COEFF_FL[1]+OP_DIFF_FL-MULTIPLYERS_FL[1]-1]) ?
mult_res_tmp_1[MULTIPLYERS_WL[1]+DIFF_COEFF_FL[1]+OP_DIFF_FL-
MULTIPLYERS_FL[1]-1:DIFF_COEFF_FL[1]+OP_DIFF_FL-MULTIPLYERS_FL[1]] + 1 :
mult_res_tmp_1[MULTIPLYERS_WL[1]+DIFF_COEFF_FL[1]+OP_DIFF_FL-
MULTIPLYERS_FL[1]-1:DIFF_COEFF_FL[1]+OP_DIFF_FL-MULTIPLYERS_FL[1]];

        mult_res_tmp_2      =      $signed(mult_coeff_2)          *
$signed(diff_res[2]);
        mult_res_2          =
(mult_res_tmp_2[DIFF_COEFF_FL[2]+OP_DIFF_FL-MULTIPLYERS_FL[2]-1]) ?
mult_res_tmp_2[MULTIPLYERS_WL[2]+DIFF_COEFF_FL[2]+OP_DIFF_FL-
MULTIPLYERS_FL[2]-1:DIFF_COEFF_FL[2]+OP_DIFF_FL-MULTIPLYERS_FL[2]] + 1 :
mult_res_tmp_2[MULTIPLYERS_WL[2]+DIFF_COEFF_FL[2]+OP_DIFF_FL-
MULTIPLYERS_FL[2]-1:DIFF_COEFF_FL[2]+OP_DIFF_FL-MULTIPLYERS_FL[2]];

        mult_res_tmp_3      =      $signed(mult_coeff_3)          *
$signed(diff_res[3]);
        mult_res_3          =
(mult_res_tmp_3[DIFF_COEFF_FL[3]+OP_DIFF_FL-MULTIPLYERS_FL[3]-1]) ?
mult_res_tmp_3[MULTIPLYERS_WL[3]+DIFF_COEFF_FL[3]+OP_DIFF_FL-
MULTIPLYERS_FL[3]-1:DIFF_COEFF_FL[3]+OP_DIFF_FL-MULTIPLYERS_FL[3]] + 1 :
mult_res_tmp_3[MULTIPLYERS_WL[3]+DIFF_COEFF_FL[3]+OP_DIFF_FL-
MULTIPLYERS_FL[3]-1:DIFF_COEFF_FL[3]+OP_DIFF_FL-MULTIPLYERS_FL[3]];

        mult_res_tmp_4      =      $signed(mult_coeff_4)          *
$signed(diff_res[4]);
        mult_res_4          =
(mult_res_tmp_4[DIFF_COEFF_FL[4]+OP_DIFF_FL-MULTIPLYERS_FL[4]-1]) ?

```

```

mult_res_tmp_4[MULTIPLYERS_WL[4]+DIFF_COEFF_FL[4]+OP_DIFF_FL-
MULTIPLYERS_FL[4]-1:DIFF_COEFF_FL[4]+OP_DIFF_FL-MULTIPLYERS_FL[4]] + 1 :

mult_res_tmp_4[MULTIPLYERS_WL[4]+DIFF_COEFF_FL[4]+OP_DIFF_FL-
MULTIPLYERS_FL[4]-1:DIFF_COEFF_FL[4]+OP_DIFF_FL-MULTIPLYERS_FL[4]];

        summ_res[0]      =      $signed(mult_res_0      )      +
$signed({mult_res_1, zeros_summ_res_0});
        summ_res[1]      =      $signed(summ_res[0])      +
$signed(mult_res_2);
        summ_res[2]      =      $signed(summ_res[1])      +
$signed({mult_res_3, zeros_summ_res_2});
        summ_res[3]      =      $signed(summ_res[2])      +
$signed({mult_res_4, zeros_summ_res_3});

        feedback_operand      =      (ctrl_reg)      ?
loop_tdata[DELAY_FEEDBACK_LOOP-1] : '0;
        summ_res[4]      =      $signed(summ_res[3])      +
$signed(feedback_operand);
    end
end

/*
    Секция предназначенная для проверки промежуточных
    результатов блоков умножения и вычитания.
    В модуле объявляется память под каждый вычислительный блок.
    По окончании принятия данных происходит запись
    файл.

    Данный блок относится к несинтезируемым структурам. При
    симуляции параметр SIM_EN можно положить равными 1, в
    таком случае сгенерируется несинтезуемая часть.

    По умолчанию SIM_EN = 0 => на этапе синтеза данная часть не
    будет сгенерирована.
*/
generate
    if (SIM_EN) begin

        localparam DATA_LENGTH = 128;
        localparam      DATA_DIR      =
"C:\\MyFolder\\RemoteFolder\\projects\\adaptive_filter\\Source\\tbn\\da
ta\\";

        logic [$clog2(DATA_LENGTH)-1:0] cnt_data;
        logic      finish_data_transfer;

        logic      [MULTIPLYERS_WL[0]-1:0]      mult_res_0_mem
[DATA_LENGTH-1:0];
        logic      [MULTIPLYERS_WL[1]-1:0]      mult_res_1_mem
[DATA_LENGTH-1:0];
        logic      [MULTIPLYERS_WL[2]-1:0]      mult_res_2_mem
[DATA_LENGTH-1:0];
        logic      [MULTIPLYERS_WL[3]-1:0]      mult_res_3_mem
[DATA_LENGTH-1:0];
        logic      [MULTIPLYERS_WL[4]-1:0]      mult_res_4_mem
[DATA_LENGTH-1:0];

```

```

        logic [OP_DIFF_WL-OP_DIFF_FL-1:-OP_DIFF_FL]
diff_res_mem [DATA_LENGTH-1:0][FIR_DIFF_COEFF_NUM-1:0];

        always_ff @(posedge clk) begin
            if (srst) begin
                cnt_data <= '0;

                foreach (mult_res_0_mem[element]) begin
                    mult_res_0_mem[element] <= '0;
                end

                foreach (mult_res_1_mem[element]) begin
                    mult_res_1_mem[element] <= '0;
                end

                foreach (mult_res_2_mem[element]) begin
                    mult_res_2_mem[element] <= '0;
                end

                foreach (mult_res_3_mem[element]) begin
                    mult_res_3_mem[element] <= '0;
                end

                foreach (mult_res_4_mem[element]) begin
                    mult_res_4_mem[element] <= '0;
                end

                for (int i = 0; i < DATA_LENGTH; i++) begin
                    for (int j = 0; j < FIR_DIFF_COEFF_NUM; j++)
                        diff_res_mem[i][j] <= '0;
                end
            end

            end else begin
                if (s_tvalid_reg) begin
                    cnt_data <= cnt_data + 1;
                    mult_res_0_mem[cnt_data] <= mult_res_0;
                    mult_res_1_mem[cnt_data] <= mult_res_1;
                    mult_res_2_mem[cnt_data] <= mult_res_2;
                    mult_res_3_mem[cnt_data] <= mult_res_3;
                    mult_res_4_mem[cnt_data] <= mult_res_4;

                    for (int i = 0; i < FIR_DIFF_COEFF_NUM; i++)
                        diff_res_mem[cnt_data][i] <=
diff_res[i];

                    end

                    finish_data_transfer <= (cnt_data ==
(DATA_LENGTH - 1)) ? 1'b1 : 1'b0;
                end else begin
                    cnt_data <= cnt_data;
                    mult_res_0_mem <= mult_res_0_mem;
                    mult_res_1_mem <= mult_res_1_mem;
                    mult_res_2_mem <= mult_res_2_mem;
                    mult_res_3_mem <= mult_res_3_mem;
                    mult_res_4_mem <= mult_res_4_mem;
                end
            end
        end
    end
endmodule

```

```

diff_res_mem      <= diff_res_mem;
finish_data_transfer      <=
finish_data_transfer;
end
end
end
always @(*) begin
    if (finish_data_transfer) begin
        $writememb({DATA_DIR,      "mult_0.txt"},
mult_res_0_mem);
        $writememb({DATA_DIR,      "mult_1.txt"},
mult_res_1_mem);
        $writememb({DATA_DIR,      "mult_2.txt"},
mult_res_2_mem);
        $writememb({DATA_DIR,      "mult_3.txt"},
mult_res_3_mem);
        $writememb({DATA_DIR,      "mult_4.txt"},
mult_res_4_mem);
        $writememb({DATA_DIR,      "diff.txt"      },
diff_res_mem );
    end
end
end
endgenerate
endmodule

```

### 3.3 Файл с тестовыми воздействиями

```
`timescale 1ns / 1ns

//Project name: "Adaptive filter"
//Author:      Kavruk A.
//File description: Testbench

module adaptive_filter_tb (
);

    localparam FILTER_MODE          = 0; // если 0 -
дифференцирование, 1 - интегрирование
    localparam INCLUDE_SDF          = 0;
    localparam SIM_EN                = 1;

    localparam WORDLENGTH            = 14;
    localparam FRACTIONAL_LENGTH    = 6;

    localparam DATA_NUM             = 128;
    localparam DATA_DIR              =
"C:\\MyFolder\\RemoteFolder\\projects\\adaptive_filter\\Source\\t
bn\\data\\";

    localparam INPUT_FILE_NAME       = "data_in.txt";
    localparam OUTPUT_FILE_NAME      = "data_out.txt";
    localparam MODEL_FILE_NAME       =
"model_data_differentiator.txt";

    logic clk;
    logic srst;

    logic [WORDLENGTH-1:0] s_tdata;
    logic                   s_tvalid;
    logic [WORDLENGTH-1:0] s_tdata_mem [DATA_NUM-1:0];

    logic [WORDLENGTH-1:0] m_tdata;
    logic [WORDLENGTH-1:0] m_tdata_mem [DATA_NUM-1:0];
    logic                   m_tvalid;

    logic [WORDLENGTH-1:0] model_valid_tdata [DATA_NUM-1:0];

    logic                   start_display;
    logic                   output_is_valid_to_model;
    logic                   finish_data_transfer;

    always begin
        #2 clk = ~clk;
    end

    initial begin
        if (INCLUDE_SDF) begin
```

```

$sdf_annotate("../Outputs/Place_and_route/adaptive_filter.sdf",
dut);

    end

    clk = 1;
    srst = 1;
    #10;
    srst = 0;
    $readmemb({DATA_DIR, INPUT_FILE_NAME}, s_tdata_mem);
    $readmemb({DATA_DIR, MODEL_FILE_NAME},
model_valid_tdata);
    end

    adaptive_filter #(
        .SIM_EN (SIM_EN )
    ) dut (
        .clk (clk ),
        .srst (srst ),
        .ctrl (FILTER_MODE),

        .s_tdata (s_tdata ),
        .s_tvalid (s_tvalid ),

        .m_tdata (m_tdata ),
        .m_tvalid (m_tvalid )
    );

    logic [$clog2(DATA_NUM)-1:0] cnt_output_data;
    logic [$clog2(DATA_NUM)-1:0] cnt_input_data;

    always_ff @(posedge clk) begin
        if (srst) begin
            foreach (m_tdata_mem[element]) begin
                m_tdata_mem[element] <= '0;
            end

            s_tdata <= '0;
            cnt_output_data <= '0;
            cnt_input_data <= '0;
            finish_data_transfer <= '0;
            s_tvalid <= 1'b0;
        end else begin
            s_tvalid <= 1'b1;
            s_tdata <=
s_tdata_mem[cnt_input_data];
            cnt_input_data <= cnt_input_data +
1;
            m_tdata_mem[cnt_output_data] <= m_tdata;
            cnt_output_data <= (m_tvalid) ?
cnt_output_data + 1 : cnt_output_data;
            finish_data_transfer <= (cnt_output_data
== (DATA_NUM - 1)) ? 1'b1 : finish_data_transfer;;

```

```

        end
    end

    assign          output_is_valid_to_model          =
model_valid_tdata[cnt_output_data] == m_tdata;
    assign start_display          = m_tvalid;
    always_ff @(posedge clk) begin
        if (start_display && !finish_data_transfer) begin
            if (output_is_valid_to_model) begin
                $display("Sample %d: PASS",cnt_output_data);
            end else begin
                $display("Sample %d: FAIL",cnt_output_data);
            end
        end
    end

    if ((cnt_output_data == 0) && finish_data_transfer)
begin
        $writememb({DATA_DIR,          OUTPUT_FILE_NAME},
m_tdata_mem);
        $finish();
    end
end

endmodule : adaptive_filter_tb

```

### 3.4 Сценарий MATLAB для тестирования

```
%%
estimate_resolution;
clearvars -except coeff_integrator coeff;
clc;
close all;
%% Параметры

FILTER_MODE          = 'differentiator'; % 'integrator',
'differentiator'
GET_INPUT_DATA = 'generate';           % 'generate', 'read'
START_RTL       = 1;
DEBUG           = 1;

SIGNAL_TYPE = "sine"; % "sine", "noise"
N           = 128;
FS          = 1;
AMP         = 63.992;

%% Начальные данные
INT16_SIZE      = 16;
DATA_WIDTH      = 14;

DATA_PATH       = '..\data\';
INPUT_DATA_FILE_NAME = 'data_in.txt';
OUTPUT_DATA_FILE_NAME = 'data_out.txt';
MODEL_DATA_FILE_NAME   = ['model_data', '_', FILTER_MODE,
'.txt'];

FILTER_ORDER = 9;
MULT_NUM     = (FILTER_ORDER + 1) / 2;
DIFF_NUM     = MULT_NUM;
Time = N/FS;

WORDLENGTH      = 14;
FRACTIONAL_LENGTH = 6;

OP_DIFF_WL = 15;
OP_DIFF_FL = 6;

WORDLENGTH_MULT = [18, 19, 20, 14, 14];
FRACLENGTH_MULT = [12, 11, 12, 6, 6];

DATA_TYPE = 'int16';

%% Генерирование входных данных

if strcmp(GET_INPUT_DATA, 'generate')
    f = FS/8;
    t = 0:1/FS:(N-1)/FS;

    if (strcmp(SIGNAL_TYPE, "sine"))
```



```

        signal = AMP * sin(2*pi*f*t);
    elseif (strcmp(SIGNAL_TYPE, "noise"))
        signal = AMP * rand([1, N]);
    end

    file_id = fopen([DATA_PATH, INPUT_DATA_FILE_NAME], 'w');
    signal_fix = round(signal * 2^FRACTIONAL_LENGTH);
    for i = 1:N
        bin_repr = dec2bin(signal_fix(i));
        if length(bin_repr) < DATA_WIDTH
            zero_num = DATA_WIDTH - length(bin_repr);
            for j = 1:zero_num
                if (signal_fix(i) < 0)
                    bin_repr = ['1', bin_repr];
                else
                    bin_repr = ['0', bin_repr];
                end
            end
        elseif length(bin_repr) > DATA_WIDTH
            delete_bits_num = length(bin_repr) - WORDLENGTH; %
% во время представления                                % отрицательных
% чисел в двоичном                                     % формате в виде
% текста Matlab                                         % добавляет
% некоторое количество                                % единиц в конце
% строки
            bin_repr(1:delete_bits_num) = [];
        end

        fprintf(file_id, '%s\n', bin_repr);
    end
    fclose(file_id);
elseif strcmp(GET_INPUT_DATA, 'read')

    file_id = fopen([DATA_PATH, INPUT_DATA_FILE_NAME],
'r');
    signal_char = fscanf(file_id, '%s\n');
    fclose(file_id);

    file_id = fopen([DATA_PATH, MODEL_DATA_FILE_NAME],
'rb');
    model_dec = fread(file_id, N, 'double');
    fclose(file_id);

    signal_dec = zeros(1, N);
    for i = 1:N
        a = fi(0,1,WORDLENGTH, FRACTIONAL_LENGTH);
        a.bin = signal_char((i-1)*WORDLENGTH+1:i*WORDLENGTH);
        signal_dec(i) = double(a);
    end
end

```

```

else
    error("Неправильно выбран способ получения входного сигнала")
end

if (isequal(FILTER_MODE, 'differentiator'))
    ctrl = zeros(1, length(t));
elseif isequal(FILTER_MODE, 'integrator')
    ctrl = ones(1, length(t));
end
ctrl = [t', ctrl'];
test_signal = [t', signal'];

%% Запуск модели в Simulink
ts = 1/FS;
model_dec = start_simulink(signal, t, DEBUG, DATA_PATH);
file_id = fopen([DATA_PATH, MODEL_DATA_FILE_NAME], 'w');
for i = 1:N
    model_str = dec2bin(round(model_dec(i) *
2^FRACTIONAL_LENGTH));
    if (length(model_str) < WORDLENGTH)
        if (model_dec(i) >= 0)
            for j = 1 : WORDLENGTH - length(model_str)
                model_str = ['0', model_str];
            end
        elseif (model_dec(i) < 0)
            for j = 1 : WORDLENGTH - length(model_str)
                model_str = ['1', model_str];
            end
        end
    elseif (length(model_str) > WORDLENGTH)
        model_str(1:length(model_str)-WORDLENGTH) = [];
    end
    fprintf(file_id, '%s\n', model_str);
end
fclose(file_id);

%% Ожидание запуска симуляции
if (START_RTL)
    disp("Запустите симуляцию")
    pause();
end

%% Обработка данных симуляции
if (DEBUG)
    filter_output = read_data_from_sim([DATA_PATH,
OUTPUT_DATA_FILE_NAME], N, ...
WORDLENGTH,
FRACTIONAL_LENGTH);

    mult_rtl = zeros(N, MULT_NUM);

```

```

        mult_model = zeros(N, MULT_NUM);

        diff_rtl    = zeros(1, MULT_NUM*N);
        diff_model  = zeros(N, MULT_NUM);

        for i = 1:(FILTER_ORDER+1)/2
            mult_rtl(:, i) = read_data_from_sim([DATA_PATH,
'mult_', num2str(i-1), '.txt'], ...
                                                N,
WORDLENGTH_MULT(i), FRACLENGTH_MULT(i));

            file_id      = fopen([DATA_PATH, 'mult_', num2str(i-
1)], 'rb');
            mult_model(:,i) = fread(file_id, N, 'double');
            fclose(file_id);

            file_id      = fopen([DATA_PATH, 'diff_', num2str(i-
1)], 'rb');
            diff_model(:,i) = fread(file_id, N, 'double');
            fclose(file_id);

        end

        diff_rtl = read_data_from_sim([DATA_PATH, 'diff', '.txt'],
...
                                    N*MULT_NUM,      OP_DIFF_WL,
OP_DIFF_FL);

        diff_rtl = reshape(diff_rtl, DIFF_NUM, N);
        diff_rtl = diff_rtl';

        plot(filter_output)
        hold on
        plot(model_dec)
        legend("Выходной сигнал модуля", "Выходной сигнал модели")
        hold off

        if (isequal(diff_model, diff_rtl))
            disp("Данные с выхода блоков вычитания совпали")
        else
            disp("ОШИБКА!! Данные с выхода блоков вычитания не
совпали")
        end

        if (isequal(mult_model, mult_rtl))
            disp("Данные с выхода умножителей совпали")
        else
            disp("ОШИБКА!! Данные с выхода умножителей не
совпали")
        end

        if (isequal(filter_output, model_dec(1:N)))
            disp("Simulink-модель совпала с RTL")
        end

```

```

        else
            disp("ОШИБКА!! RTL не соответствует Simulink-модели")
        end

    end

%% Функции

function output_data = start_simulink(signal, t, debug,
file_path)

    if nargin < 3
        debug = 0;
    end

    if ((nargin < 4) && debug)
        file_path = '.';
    end

    test_signal = [t', signal'];

    sim("models_diff_integr.slx");
    output_data = ans.fp_out_signal.data;

    mult_0_data = ans.mult_0.data;
    mult_1_data = ans.mult_1.data;
    mult_2_data = ans.mult_2.data;
    mult_3_data = ans.mult_3.data;
    mult_4_data = ans.mult_4.data;

    diff_0_data = ans.diff_0.data;
    diff_1_data = ans.diff_1.data;
    diff_2_data = ans.diff_2.data;
    diff_3_data = ans.diff_3.data;
    diff_4_data = ans.diff_4.data;

    if (debug)

        file_id = fopen([file_path, 'mult_0'], 'wb');
        fwrite(file_id, mult_0_data, 'double');
        fclose(file_id);

        file_id = fopen([file_path, 'mult_1'], 'wb');
        fwrite(file_id, mult_1_data, 'double');
        fclose(file_id);

        file_id = fopen([file_path, 'mult_2'], 'wb');
        fwrite(file_id, mult_2_data, 'double');
        fclose(file_id);

        file_id = fopen([file_path, 'mult_3'], 'wb');
        fwrite(file_id, mult_3_data, 'double');
    end

```

```

        fclose(file_id);

        file_id = fopen([file_path, 'mult_4'], 'wb');
        fwrite(file_id, mult_4_data, 'double');
        fclose(file_id);

        file_id = fopen([file_path, 'diff_0'], 'wb');
        fwrite(file_id, diff_0_data, 'double');
        fclose(file_id);

        file_id = fopen([file_path, 'diff_1'], 'wb');
        fwrite(file_id, diff_1_data, 'double');
        fclose(file_id);

        file_id = fopen([file_path, 'diff_2'], 'wb');
        fwrite(file_id, diff_2_data, 'double');
        fclose(file_id);

        file_id = fopen([file_path, 'diff_3'], 'wb');
        fwrite(file_id, diff_3_data, 'double');
        fclose(file_id);

        file_id = fopen([file_path, 'diff_4'], 'wb');
        fwrite(file_id, diff_4_data, 'double');
        fclose(file_id);

    end

end

function data_decimal = read_data_from_sim(file_path,
data_length, wl, fl)

    file_id = fopen(file_path, 'r');
    data_char = fscanf(file_id, '%s');
    fclose(file_id);

    check_length = 32;
    mask = zeros(1, check_length);
    for i = 1:round(length(data_char)/check_length)
        alpha_pos = isstrprop(data_char(1:check_length),
'alpha');
        if(isequal(alpha_pos, mask))
            break;
        else
            data_char(1) = [];
            j = 2;
            while (alpha_pos(j))
                data_char(1) = [];
                j = j + 1;
            end
        end
    end
end
end

```

```
data_decimal = zeros(1, data_length);  
for i = 1:data_length  
    a = fi(0,1,wl, fl);  
    a.bin = data_char((i-1)*wl+1:i*wl);  
    data_decimal(i) = double(a);  
end
```

```
end
```

### 3.5 Файл с ограничениями

```
set_units -time ns -capacitance pF

create_clock -name clk -period 25 [get_ports clk]

set_input_delay      -clock clk -max 0.5 [get_ports  -filter
{@port_direction == in}]
set_output_delay     -clock clk -max 0.5 [get_ports  -filter
{@port_direction == out}]

set_clock_uncertainty -to clk 0.25

set_load             0.5 [all_outputs]
```

### 3.6 Сценарий логического синтеза

```
###Project name: "Adaptive filter"
###Author:      Kavruk A.
###Technology:   X-FAB 180nm CMOS, XT018
###Library:     "D_CELLS_HD, 1.8V"
###Tools:       "RTL Compiler"

###Stage: Synthesis
###File description: TCL script for Synthesis with slow corner
###Work Directory: /adaptive_filter/Scripts/

# Setup path for liberty CPF directory
set_attribute                                lib_search_path
/Cadence/Libs/X_FAB/XKIT/xt018/diglibs/D_CELLS_HD/v4_0/liberty_LP
5MOS/v4_0_0/PVT_1_80V_range

# Setup PVT corner .lib file
set_attribute library D_CELLS_HD_LP5MOS_slow_1_62V_175C.lib

# Setup LEF files
set_attribute                                lef_library
{/Cadence/Libs/X_FAB/XKIT/xt018/cadence/v7_0/techLEF/v7_0_1_1/xt0
18_xx43_MET4_METMID_METTHK.lef \
  /Cadence/Libs/X_FAB/XKIT/xt018/diglibs/D_CELLS_HD/v4_0/LEF/v4
_0_0/xt018_D_CELLS_HD.lef \
  /Cadence/Libs/X_FAB/XKIT/xt018/diglibs/D_CELLS_HD/v4_0/LEF/v4
_0_0/xt018_xx43_MET4_METMID_METTHK_D_CELLS_HD_mprobe.lef
}

# Setup Capacitance Table file
set_attribute                                cap_table_file
/Cadence/Libs/X_FAB/XKIT/xt018/cadence/v7_0/capTbl/v7_0_1/xt018_x
x43_MET4_METMID_METTHK_max.capTbl

# Setup error on blackbox
set_attribute hdl_error_on_blackbox true
```



### 3.7 Файл МММС

```
###Project name: "Adaptive filter"
###Author:      Kavruk A.
###Technology:   X-FAB 180nm CMOS, XT018
###Library:      "D_CELLS_HD, 1.8V"
###Tools:        "Cadence Encounter 14.28"

###Stage: Place_and_Route
###File description: TCL script for Place and Route in multy-mode
multy-corner
###Work Directory: /adaptive_filter/Scripts/

## Подключение файла-констрэйнов сгенерированного на этапе
логического синтеза
create_constraint_mode -name CONSTRAINTS -sdc_files
{../Source/rtl/Top_syn_out.sdc}

## Создание набора библиотек
create_library_set -name SLOWlib \
-timing
{/Cadence/Libs/X_FAB/XKIT/xt018/diglibs/D_CELLS_HD/v4_0/liberty_LP5MOS/
v4_0_0/PVT_1_80V_range/D_CELLS_HD_LP5MOS_slow_1_62V_175C.lib
}

create_library_set -name TYPlib \
-timing
{/Cadence/Libs/X_FAB/XKIT/xt018/diglibs/D_CELLS_HD/v4_0/liberty_LP5MOS/
v4_0_0/PVT_1_80V_range/D_CELLS_HD_LP5MOS_typ_1_80V_25C.lib
}

create_library_set -name FASTlib \
-timing
{/Cadence/Libs/X_FAB/XKIT/xt018/diglibs/D_CELLS_HD/v4_0/liberty_LP5MOS/
v4_0_0/PVT_1_80V_range/D_CELLS_HD_LP5MOS_fast_1_98V_m40C.lib
}

##create_op_cond -name PVT_slow_3_00V_175C \
##-library_file {<Path-to-cells-library>/liberty/.../<PVT corner
name>/<Library
##name>_<Slow corner name>.lib} \
##-P {1} -V {3} -T {175}

## Создание RC-угла из captable
create_rc_corner -name RCcornerMIN \
-cap_table
/Cadence/Libs/X_FAB/XKIT/xt018/cadence/v7_0/capTbl/v7_0_1/xt018_xx43_ME
T4_METMID_METTHK_min.capTbl \
-qx_tech_file
/Cadence/Libs/X_FAB/XKIT/xt018/cadence/v7_0/QRC_pvs/v7_0_3/XT018_1243/Q
RC-Min/qrcTechFile

create_rc_corner -name RCcornerTYP \
-cap_table
/Cadence/Libs/X_FAB/XKIT/xt018/cadence/v7_0/capTbl/v7_0_1/xt018_xx43_ME
T4_METMID_METTHK_typ.capTbl \
-qx_tech_file
/Cadence/Libs/X_FAB/XKIT/xt018/cadence/v7_0/QRC_pvs/v7_0_3/XT018_1243/Q
RC-Typ/qrcTechFile
```

```

create_rc_corner -name RCcornerMAX \
-cap_table
/Cadence/Libs/X_FAB/XKIT/xt018/cadence/v7_0/capTbl/v7_0_1/xt018_xx43_ME
T4_METMID_METTHK_max.capTbl \
-qx_tech_file
/Cadence/Libs/X_FAB/XKIT/xt018/cadence/v7_0/QRC_pvs/v7_0_3/XT018_1243/Q
RC-Max/qrcTechFile

## Создание корнера задержки
create_delay_corner -name DELAYcornerSLOW \
-library_set SLOWlib \
-rc_corner RCcornerMAX

create_delay_corner -name DELAYcornerTYP \
-library_set TYPlib \
-rc_corner RCcornerTYP

create_delay_corner -name DELAYcornerFAST \
-library_set FASTlib \
-rc_corner RCcornerMIN

## Создание типов анализа
create_analysis_view -name MAXview \
-delay_corner {DELAYcornerSLOW} \
-constraint_mode {CONSTRAINTS}

create_analysis_view -name TYPview \
-delay_corner {DELAYcornerTYP} \
-constraint_mode {CONSTRAINTS}

create_analysis_view -name MINview \
-delay_corner {DELAYcornerFAST} \
-constraint_mode {CONSTRAINTS}

## Выбор наихудшего корнера по setup и по hold
set_analysis_view -setup {MAXview} -hold {MINview}

```

### 3.8 Сценарий создания топологии фильтра

```
###Project name: "Adaptive filter"
###Author:      Kavruk A.
###Technology:   X-FAB 180nm CMOS, XT018
###Library:     "D_CELLS_HD, 1.8V"
###Tools:       "Cadence Encounter 14.28"
###Stage: Place_and_Route
###File description: TCL script for Place and Route
###Work Directory: /adaptive_filter/Scripts/
###Run command: Encounter ../Scripts/adaptive_filter_PaR.tcl

#Import design
set_global _enable_mmmc_by_default_flow      $CTE::mmmc_default
suppressMessage ENCEXT-2799
set_global _enable_mmmc_by_default_flow      $CTE::mmmc_default
suppressMessage ENCEXT-2799
win
set ::TimeLib::tsgMarkCellLatchConstructFlag 1
set defHierChar /
set distributed_client_message_echo 1
set gpsPrivate::dpgNewAddBufsDBUpdate 1
set gpsPrivate::lsgEnableNewDbApiInRestruct 1
set init_design_settop 0
set init_gnd_net VSS
set init_io_file ../Outputs/Place_and_Route/Module_pins
set                                     init_lef_file
{/Cadence/Libs/X_FAB/XKIT/xt018/cadence/v7_0/techLEF/v7_0_1_1/xt0
18_xx43_MET4_METMID_METTHK.lef
/Cadence/Libs/X_FAB/XKIT/xt018/diglibs/D_CELLS_HD/v4_0/LEF/v4
_0_0/xt018_D_CELLS_HD.lef}
set init_mmmc_file ../Scripts/MMMC.tcl
set init_oa_search_lib {}
set init_pwr_net VDD
set init_verilog ../Outputs/Synthesis/synth_hdl.v
set lsgOCPGainMult 1.000000
set pegDefaultResScaleFactor 1.000000
set pegDetailResScaleFactor 1.000000
set timing_library_float_precision_tol 0.000010
set timing_library_load_pin_cap_indices {}
set      tso_post_client_restore_command      {update_timing      ;
write_eco_opt_db ;}
init_design

#Floorplanning: chip geometry
getIoFlowFlag
setIoFlowFlag 0
floorPlan -fplanOrigin center -site core_hd -r 1 0.5 10 10 10
10
uiSetTool select
getIoFlowFlag
fit
globalNetConnect VDD -type pgpin -pin vdd -inst *
```

```

globalNetConnect VSS -type pgpin -pin gnd -inst *
globalNetConnect VDD -type tiehi -inst * -module {}
globalNetConnect VSS -type tielo -inst * -module {}

#Floorplanning: power rails
set sprCreateIeRingNets {}
set sprCreateIeRingLayers {}
set sprCreateIeRingWidth 1.0
set sprCreateIeRingSpacing 1.0
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin
Standardcell -stacked_via_top_layer METTPL -type core_rings -
jog_distance 3.15 -threshold 3.15 -nets {VSS VDD} -follow core -
stacked_via_bottom_layer MET1 -layer {bottom MET1 top MET1 right
MET2 left MET2} -width 3 -spacing 2.5 -offset 3.15
addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin
Standardcell -stacked_via_top_layer METTPL -type core_rings -
jog_distance 3.15 -threshold 3.15 -nets {VSS VDD} -follow core -
stacked_via_bottom_layer MET1 -layer {bottom MET1 top MET1 right
MET2 left MET2} -width 3 -spacing {bottom 0.23 top 0.23 right 0.28
left 0.28} -offset 3.15
set sprCreateIeStripeNets {}
set sprCreateIeStripeLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeSpacing 2.0
set sprCreateIeStripeThreshold 1.0
win
addStripe -skip_via_on_wire_shape Noshape -
block_ring_top_layer_limit MET3 -max_same_layer_jog_length 6 -
padcore_ring_bottom_layer_limit MET1 -set_to_set_distance 25 -
skip_via_on_pin Standardcell -stacked_via_top_layer METTPL -
padcore_ring_top_layer_limit MET3 -spacing 2.5 -xleft_offset 30 -
xright_offset 30 -merge_stripes_value 3.15 -layer MET2 -
block_ring_bottom_layer_limit MET1 -width 3 -nets {VDD VSS} -
stacked_via_bottom_layer MET1
sroute -connect { blockPin padPin padRing corePin
floatingStripe } -layerChangeRange { MET1 METTPL } -blockPinTarget
{ nearestTarget } -padPinPortConnect { allPort oneGeom } -
padPinTarget { nearestTarget } -corePinTarget { firstAfterRowEnd }
-floatingStripeTarget { blockring padring ring stripe ringpin
blockpin followpin } -allowJogging 1 -crossoverViaLayerRange { MET1
METTPL } -nets { VSS VDD } -allowLayerChange 1 -blockPin useLef -
targetViaLayerRange { MET1 METTPL }
editPowerVia -skip_via_on_pin Standardcell -bottom_layer MET1
-add_vias 1 -top_layer METTPL
set sprEpvLayers {}
redirect -quiet {set honorDomain [getAnalysisMode -
honorClockDomains]} > /dev/null

#Placement

```

```

timeDesign -prePlace -idealClock -pathReports -drvReports -
slackReports -numPaths 50 -prefix adaptive_filter_prePlace -outDir
../Reports/Place_and_Route
redirect -quiet {set honorDomain [getAnalysisMode -
honorClockDomains]} > /dev/null
timeDesign -prePlace -idealClock -pathReports -drvReports -
slackReports -numPaths 50 -prefix adaptive_filter_prePlace -outDir
../Reports/Place_and_Route
setPlaceMode -fp false
placeDesign -inPlaceOpt
redirect -quiet {set honorDomain [getAnalysisMode -
honorClockDomains]} > /dev/null

#CTS
timeDesign -preCTS -idealClock -pathReports -drvReports -
slackReports -numPaths 50 -prefix adaptive_filter_preCTS -outDir
../Reports/Place_and_Route
redirect -quiet {set honorDomain [getAnalysisMode -
honorClockDomains]} > /dev/null
timeDesign -preCTS -hold -idealClock -pathReports -
slackReports -numPaths 50 -prefix adaptive_filter_preCTS -outDir
../Reports/Place_and_Route
setOptMode -fixCap true -fixTran true -fixFanoutLoad true
optDesign -preCTS
createClockTreeSpec -bufferList {BUHDX0 BUHDX1 BUHDX12 BUHDX2
BUHDX3 BUHDX4 BUHDX6 BUHDX8 DLY1HDX0 DLY1HDX1 DLY2HDX0 DLY2HDX1
DLY4HDX0 DLY4HDX1 DLY8HDX0 DLY8HDX1 INHDX0 INHDX1 INHDX2 INHDX12
INHDX3 INHDX4 INHDX6 INHDX8 STEHDX0 STEHDX1 STEHDX2 STEHDX4} -file
Clock.ctstch
clockDesign -specFile Clock.ctstch -outDir clock_report -
fixedInstBeforeCTS
setCTSMode -engine ck
clockDesign -specFile Clock.ctstch -outDir clock_report -
fixedInstBeforeCTS
redirect -quiet {set honorDomain [getAnalysisMode -
honorClockDomains]} > /dev/null
timeDesign -postCTS -pathReports -drvReports -slackReports -
numPaths 50 -prefix adaptive_filter_postCTS -outDir
../Reports/Place_and_Route
redirect -quiet {set honorDomain [getAnalysisMode -
honorClockDomains]} > /dev/null
timeDesign -preCTS -hold -idealClock -pathReports -
slackReports -numPaths 50 -prefix adaptive_filter_preCTS -outDir
../Reports/Place_and_Route
setOptMode -fixCap false -fixTran false -fixFanoutLoad false
optDesign -postCTS -incr
optDesign -postCTS -hold -incr

#Routing
setNanoRouteMode -quiet -timingEngine {}
setNanoRouteMode -quiet -routeWithSiPostRouteFix 0
setNanoRouteMode -quiet -drouteStartIteration default
setNanoRouteMode -quiet -routeTopRoutingLayer default

```

```

setNanoRouteMode -quiet -routeBottomRoutingLayer default
setNanoRouteMode -quiet -routeEndIteration default
setNanoRouteMode -quiet -routeWithTimingDriven false
setNanoRouteMode -quiet -routeWithSiDriven false
routeDesign -globalDetail
setAnalysisMode -analysisType onChipVariation -skew true -
clockPropagation sdcControl
redirect -quiet {set honorDomain [getAnalysisMode -
honorClockDomains]} > /dev/null
timeDesign -postRoute -pathReports -drvReports -slackReports -
numPaths 50 -prefix adaptive_filter_postRoute -outDir
../Reports/Place_and_Route
redirect -quiet {set honorDomain [getAnalysisMode -
honorClockDomains]} > /dev/null
timeDesign -postRoute -hold -pathReports -slackReports -
numPaths 50 -prefix adaptive_filter_postRoute -outDir
../Reports/Place_and_Route
setOptMode -fixCap true -fixTran true -fixFanoutLoad true
optDesign -postRoute
optDesign -postRoute -hold
getFillerMode -quiet
addFiller -cell FEED7HD FEED5HD FEED3HD FEED2HD FEED25HD
FEED1HD FEED15HD FEED10HD -prefix FILLER

#Verification and parasitic extraction
setVerifyGeometryMode -area { 0 0 0 0 } -minWidth true -
minSpacing true -minArea true -sameNet true -short true -overlap
true -offRGrid false -offMGrid true -mergedMGridCheck true -minHole
true -implantCheck true -minimumCut true -minStep true -
viaEnclosure true -antenna false -insuffMetalOverlap true -
pinInBlkg false -diffCellViol true -sameCellViol false -
padFillerCellsOverlap true -routingBlkgPinOverlap true -
routingCellBlkgOverlap true -regRoutingOnly false -
stackedViasOnRegNet false -wireExt true -useNonDefaultSpacing
false -maxWidth true -maxNonPrefLength -1 -error 1000
verifyGeometry
setVerifyGeometryMode -area { 0 0 0 0 }
verify_drc -report adaptive_filter.drc.rpt -limit 1000
verifyConnectivity -type all -error 1000 -warning 50
setExtractRCMode -engine postRoute -effortLevel signoff
extractRC
redirect -quiet {set honorDomain [getAnalysisMode -
honorClockDomains]} > /dev/null
timeDesign -signoff -pathReports -drvReports -slackReports -
numPaths 50 -prefix adaptive_filter_signOff -outDir
../Reports/Place_and_Route
redirect -quiet {set honorDomain [getAnalysisMode -
honorClockDomains]} > /dev/null
timeDesign -signoff -hold -pathReports -slackReports -numPaths
50 -prefix adaptive_filter_signOff -outDir
../Reports/Place_and_Route
all_hold_analysis_views
all_setup_analysis_views

```

```
#SDF, DEF, logic and physical netlist
write_sdf -view MAXview ../Outputs/adaptive_filter.sdf
saveNetlist ../Outputs/adaptive_filter_logic
saveNetlist ../Outputs/asic_filter.v
global dbgLefDefOutVersion
set dbgLefDefOutVersion 5.8
defOut          -floorplan          -netlist          -routing
../Outputs/adaptive_filter.def
set dbgLefDefOutVersion 5.8
```

## Заключение

Таким образом, в ходе работы был спроектирован перестраиваемый цифровой фильтр с максимальной тактовой частотой 40 МГц при минимальной требуемой 20 МГц, средствами автоматического проектирования была создана его топология на кристалле.



## 4 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Lyons R.G. Understanding Digital Signal Processing / R.G. Lyons. – 3<sup>rd</sup> ed. – Boston: Prentice Hall, 2011. – pp. 787.
2. Intel® Quartus® Prime Pro Edition Help version 22.1. URL: <https://www.intel.com/content/www/us/en/programmable/quartushelp/22.1/index.htm#tafs/tafs/tafs.htm>
3. Харрис, Д. М. Цифровая схемотехника и архитектура компьютера [Электронный ресурс] / Дэвид М. Харрис и Сара Л. Харрис. - Нью-Йорк : Elsevier. inc : Изд-во Morgan Kaufman, 2013. - on-line. - ISBN = 978-0-12-394424-5