МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №2

«Разработка аппаратных ускорителей математических вычислений»
по дисциплине
«Функциональная схемотехника»

Вариант №9

Выполнила:
студентка группы P33082
Савельева Диана Александровна

Преподаватель:
Кустарев Павел Валерьевич

Санкт-Петербург, 2024

1. Цель работы

Получить навыки описания арифметических блоков на RTL-уровне с использованием языка описания аппаратуры Verilog HDL

2. Задание (Вариант №9)

Порядок выполнения работы

1. Разработайте и опишите на Verilog HDL схему, вычисляющую значение функции в соответствии с заданными ограничениями.

2. Определите область допустимых значений функции.

3. Разработайте тестовое окружение для разработанной схемы. Тестовое окружение должно проверять работу схемы не менее, чем на 10 различных тестовых векторах.

4. Проведите моделирование работы схемы и определите время вычисления результата. Схема должна тактироваться от сигнала с частотой 100 МГц.

5. Составьте отчет по результатам выполнения работы.

Вариант	*Функция*	*Ограничения*
9	$y = \sqrt[3]{a} + \sqrt{b}$	2 сумматора и 1 умножитель

В качестве входных данных необходимо использовать целые беззнаковые числа с разрядностью 8 бит. Разрядность выходного значения выбирается в соответствии с областью допустимых значений функции. Результат вычислений не должен выходить за границы формата представления выходного значения блока.

Ограничения накладываются на количество используемых блоков суммирования и умножения. В разработанной схеме должен быть использован блок умножения, реализующий последовательный алгоритм умножения «в столбик». При выполнении заданий необходимо использовать беззнаковую целочисленную арифметику.

3. Схема разработанного блока вычисления функции, заданной вариантом, в терминах базовых операционных элементов (БОЭ)

На рисунке 1 представлена схема разработанного блока вычисления функции $y = \sqrt[3]{a} + \sqrt{b}$ в терминах базовых операционных элементов. На данной схеме также представлены разработанные для заданной функции модули извлечения квадратного корня и кубического корней. В модуле кубического корня также используется разработанный модуль умножителя.

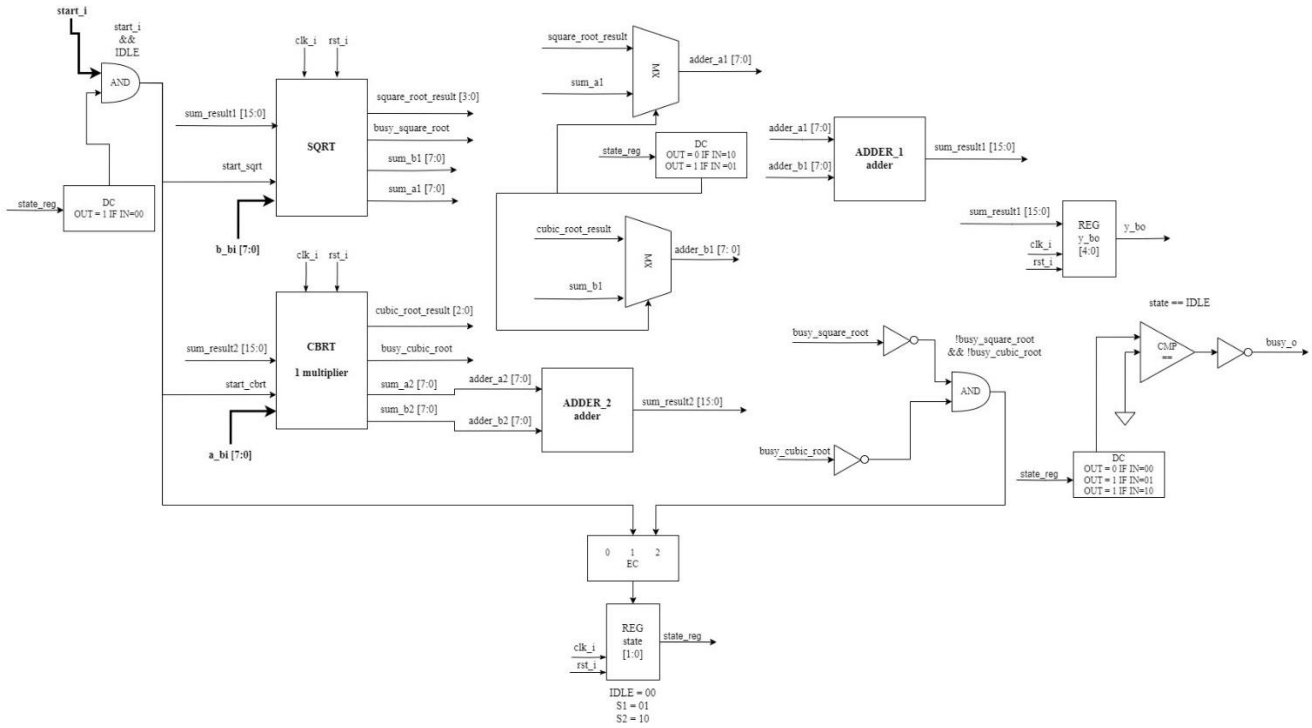

Рисунок 1 – Схема разработанного блока вычисления функции

На рисунке 2 и 3 представлена схема разработанного блока вычисления кубического корня.

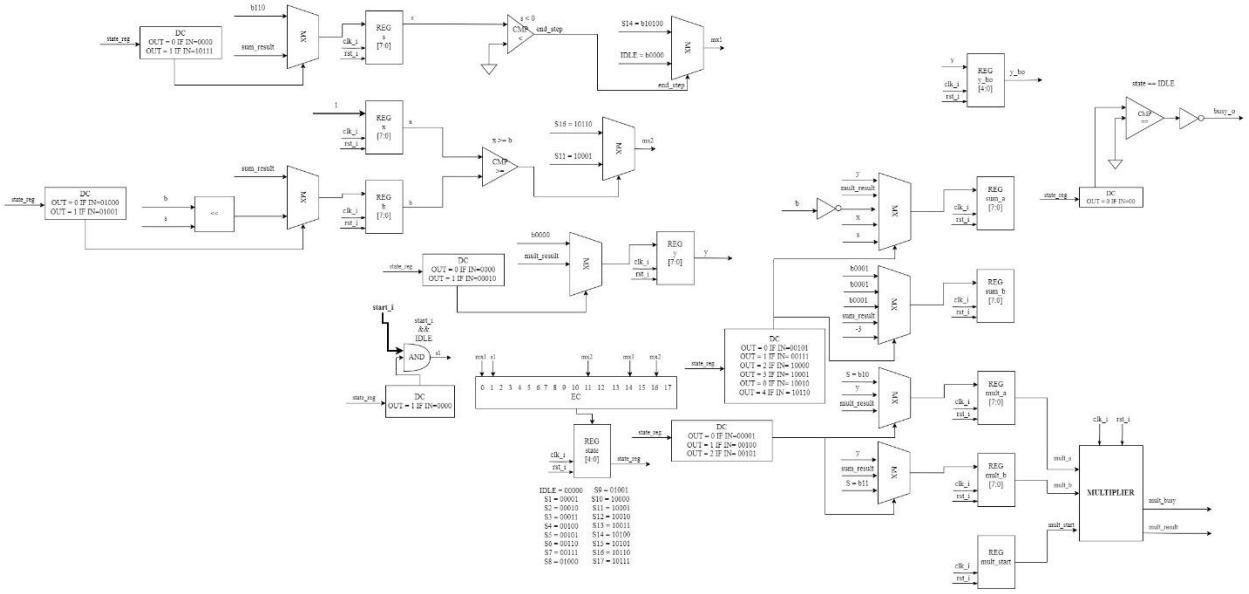

Рисунок 2 – Схема разработанного блока вычисления кубического корня

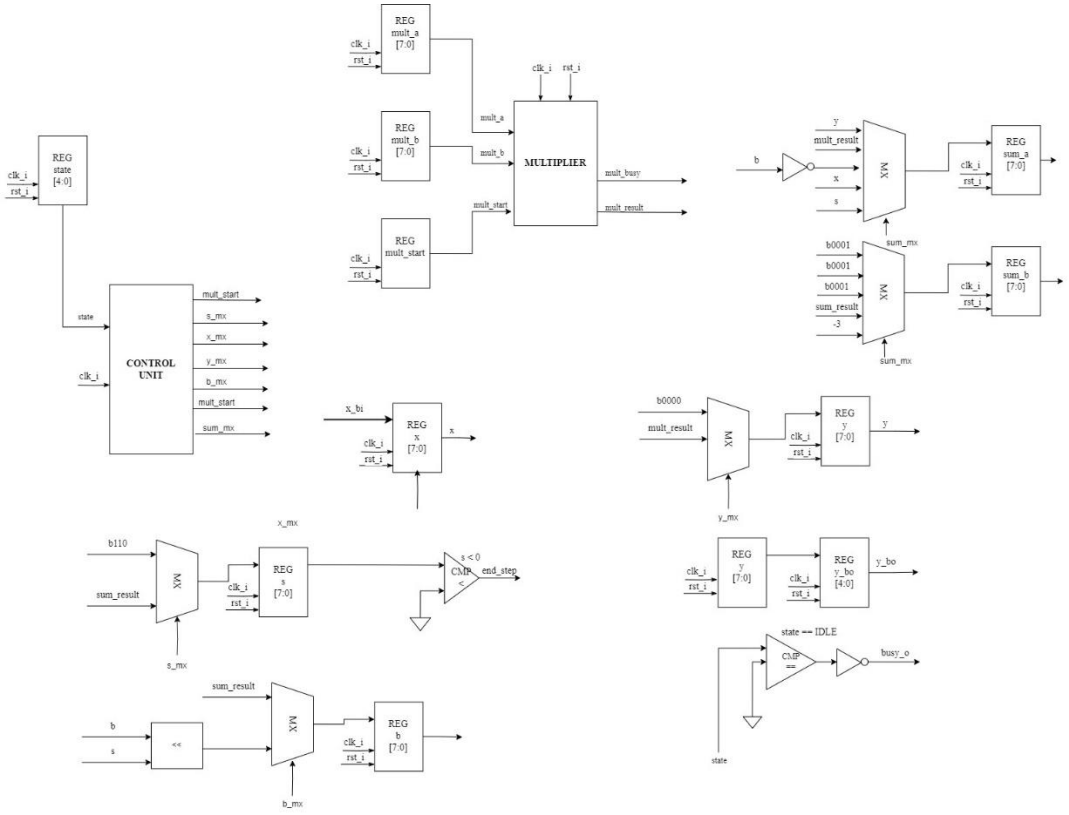

Рисунок 3 – Обобщенная разработанного блока вычисления кубического корня

4. Описание работы разработанного блока, начиная с подачи входных данных и заканчивая получением результата

На вход разработанного блока функции подаются следующие сигналы:

- start_i – сигнал, который сообщает о начале вычислений в блоке.
- rst_i – сигнал сброса.
- clk_i – сигнал, посылающий синхроимпульсы.
- a_bi – аргумент a (8 бит) для вычисления значения функции (данное значение далее подается на вход кубическому корню).
- b_bi – аргумент b (8 бит) для вычисления значения функции (данное значение далее подается на вход квадратному корню).

На выходе разработанного блока функции имеем:

- busy_o – сообщает о занятости блока вычислениями.
- y_bo – регистр, который хранит результат вычислений.

Модуль функции использует **2 сумматора**, 1 модуль вычисления квадратного корня (внутри которого используется сумматор, переданный из функции) и 1 модуль вычисления кубического корня (внутри которого используется сумматор, переданный из функции, и **1 умножитель**).

На вход разработанного модуля кубического и квадратного корней подаются следующие сигналы:

- start_i – сигнал, который сообщает о начале вычислений в блоке.
- rst_i – сигнал сброса.
- clk_i – сигнал, посылающий синхроимпульсы.
- x_bi– аргумент x (8 бит) для вычисления значения корня.
- sum_result – вход, который используется для передачи значения результата работы сумматора.

На выходе разработанных блоков корня имеем:

- busy_o – сообщает о занятости блока вычислениями.
- y_bo – регистр, который хранит результат вычислений.
- sum_a, sum_b – регистры, которые хрaят аргументы для сложения сумматора. С помощью данных регистров можно передавать значение из нисходящего модуля в модуль функции.

Основной алгоритм вычисления значения функции:

1) Модуль вычисления значения функции получает на вход «1» по сигналу start_i.
2) После того, как на start_i была подана «1», выход busy_o принимает значение «1», что говорит о том, что модуль занят вычислением значения функции. Как только выход busy_o принимает значение «0» - он доступен для новых вычислений.
3) На вход start_sqrt и start_cbrt модулям квадратного корня и кубического корня соответственно также подается «1». Далее начинается вычисление значения квадратного корня для поданного аргумента «b» и вычисление значения кубического корня для поданного аргумента «a» одновременно. А busy_square_root и busy_cubic_root устанавливаются в «1», сигнализируют о том, что проводятся вычисления.
4) О готовности данных сигнализирует установка значения busy_square_root и busy_cubic_root в значение «0». Результаты вычисления заносятся в square_root_result и cubic_root_result соответственно.
5) Далее значения из square_root_result и cubic_root_result подаются на вход сумматору для расчета суммы данных значений.
6) Результаты работы сумматора подаются на выход.

Конечный автомат Мили для разработанной функции

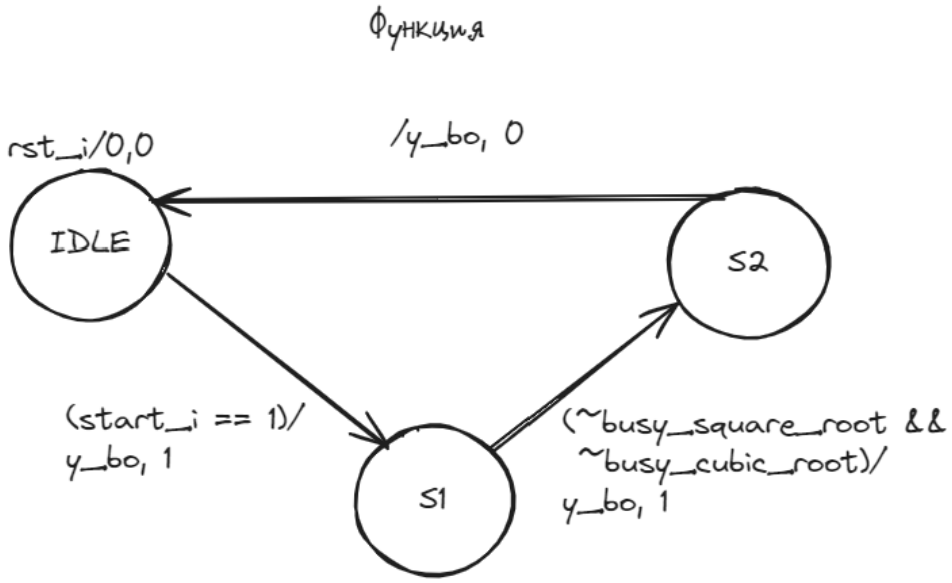

Рисунок 3 – Конечный автомат Мили для разработанного модуля функции

Каждый из модулей реализован как конечный автомат.

Модуль умножителя реализован с помощью алгоритма, представленного на рисунке ниже.

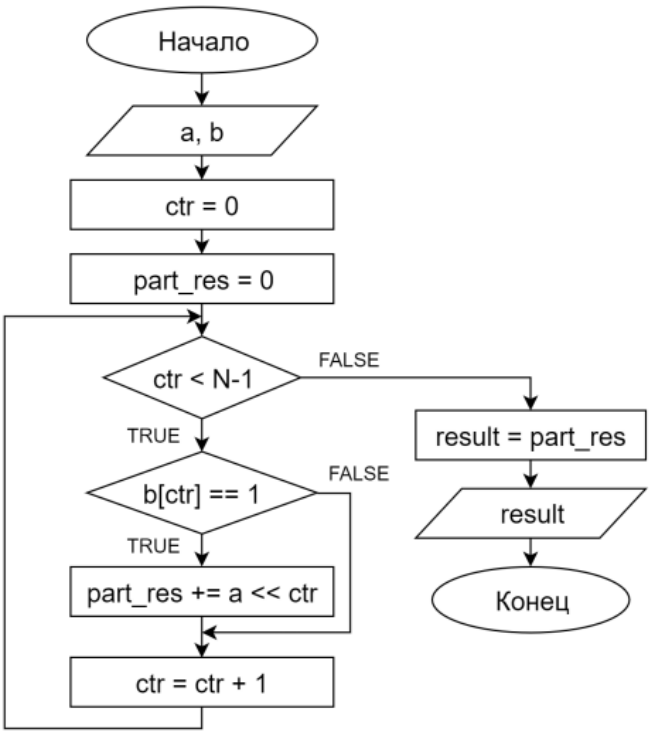

Рисунок 4 – Алгоритм умножителя

Модуль квадратного корня реализован с помощью алгоритма, представленного на рисунке ниже.

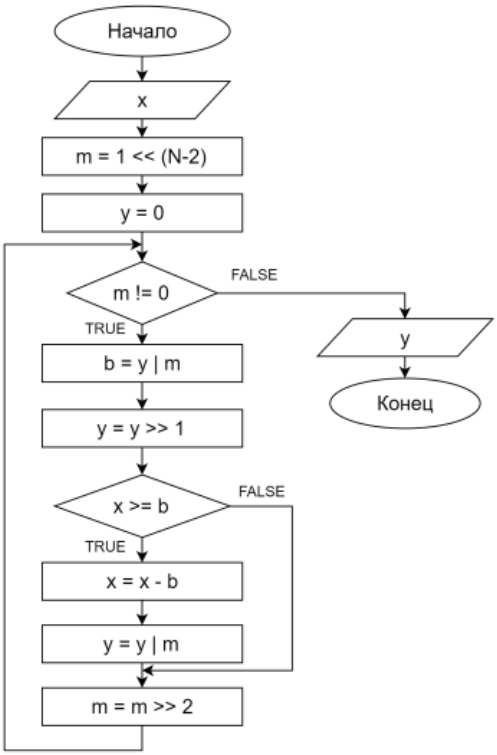

Рисунок 5 – Алгоритм нахождения квадратного корня

Модуль квадратного корня реализован с помощью алгоритма, представленного на рисунке ниже.

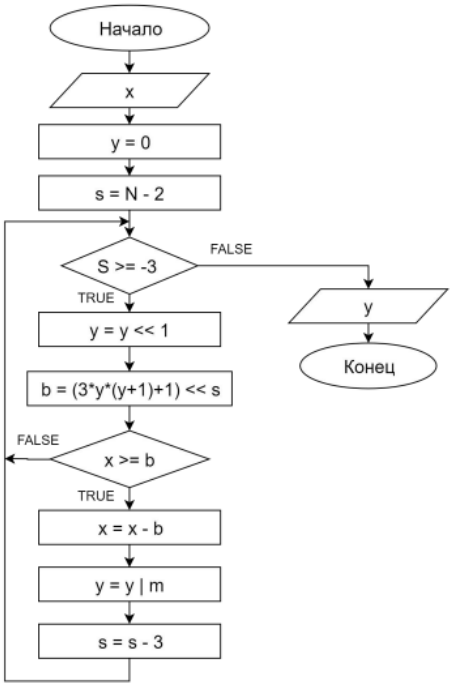

Рисунок 6 – Алгоритм нахождения кубического корня

5. Область допустимых значений для разработанного блока

Область допустимых значений для разработанного блока функции $y = \sqrt[3]{a} + \sqrt{b}$ была определена исходя из областей допустимых значений модулей, составляющих модуль функции. По итогу расчетов, представленных ниже, для хранения значения функции необходимо выделить **5 бит.**

1. Разработанный модуль **умножителя**

   INPUT: OUTPUT:

   a: 8 бит y: 16 бит

   b: 8 бит

2. Разработанный модуль **сумматора**

   INPUT: OUTPUT:

   a: 8 бит y: 16 бит

   b: 8 бит

3. Разработанный модуль **квадратного корня**

   Так как максимальное значение b, которое может быть передано для вычисления функции, составляет 255 (b: 8 бит), то максимальное значение корня, которое мы можем получить:

   $$[\sqrt{255}] = 15.$$

   Для хранения данного числа нам необходимо 4 бита.

   INPUT: OUTPUT:

   b: 8 бит y: 4 бита

4. Разработанный модуль **кубического корня**

   Так как максимальное значение a, которое может быть передано для вычисления функции, составляет 255 (a: 8 бит), то максимальное значение кубического корня, которое мы можем получить:

   $$[\sqrt[3]{255}] = 6.$$

   Для хранения данного числа нам необходимо 3 бита.

INPUT: OUTPUT:

a: 8 бит y: 3 бита

5. Разработанный модуль **функции**

   Так как для значения квадратного корня необходимо хранить 4 бита, а для кубического корня – 3 бита, то, вследствие суммирования двух значений, максимальный результат, который мы можем получить:

$$6 + 15 = 21$$

   Для хранения значения 21 нам необходимо 5 бит, так как:

$$2^4 - 1 < 21 < 2^5 - 1$$

$$15 < 21 < 31$$

   Получается, что 4 бита для хранения значения функции мало. Тогда для хранения значения функции $y = \sqrt[3]{a} + \sqrt{b}$ необходимо **5 бит.**

   INPUT: OUTPUT:

   a: 8 бит y: 5 бит

   b: 8 бит

6. Результат тестирования разработанного блока (временные диаграммы)

Тестирование выполнялось поэтапно, для каждого модуля. Далее рассмотрим тестирование каждого модуля отдельно. После вывода результата значение Y обнуляется со следующим синхроимпульсом. Все входные и выходные данные представлены в 10-ричной системе счисления.

1. Разработанный модуль **умножителя**

   Для модуля умножителя реализовано полное тестовое покрытие. На рисунке представлен скриншот с частью пройденных тестов. Ожидаемое значение рассчитываем с помощью встроенного умножителя *.

```
Test passed. On values : a=255;b=225. Expected 57375 found 57375.
Test passed. On values : a=255;b=226. Expected 57630 found 57630.
Test passed. On values : a=255;b=227. Expected 57885 found 57885.
Test passed. On values : a=255;b=228. Expected 58140 found 58140.
Test passed. On values : a=255;b=229. Expected 58395 found 58395.
Test passed. On values : a=255;b=230. Expected 58650 found 58650.
Test passed. On values : a=255;b=231. Expected 58905 found 58905.
Test passed. On values : a=255;b=232. Expected 59160 found 59160.
Test passed. On values : a=255;b=233. Expected 59415 found 59415.
Test passed. On values : a=255;b=234. Expected 59670 found 59670.
Test passed. On values : a=255;b=235. Expected 59925 found 59925.
Test passed. On values : a=255;b=236. Expected 60180 found 60180.
Test passed. On values : a=255;b=237. Expected 60435 found 60435.
Test passed. On values : a=255;b=238. Expected 60690 found 60690.
Test passed. On values : a=255;b=239. Expected 60945 found 60945.
Test passed. On values : a=255;b=240. Expected 61200 found 61200.
Test passed. On values : a=255;b=241. Expected 61455 found 61455.
Test passed. On values : a=255;b=242. Expected 61710 found 61710.
Test passed. On values : a=255;b=243. Expected 61965 found 61965.
Test passed. On values : a=255;b=244. Expected 62220 found 62220.
Test passed. On values : a=255;b=245. Expected 62475 found 62475.
Test passed. On values : a=255;b=246. Expected 62730 found 62730.
Test passed. On values : a=255;b=247. Expected 62985 found 62985.
Test passed. On values : a=255;b=248. Expected 63240 found 63240.
Test passed. On values : a=255;b=249. Expected 63495 found 63495.
Test passed. On values : a=255;b=250. Expected 63750 found 63750.
Test passed. On values : a=255;b=251. Expected 64005 found 64005.
Test passed. On values : a=255;b=252. Expected 64260 found 64260.
Test passed. On values : a=255;b=253. Expected 64515 found 64515.
Test passed. On values : a=255;b=254. Expected 64770 found 64770.
Test passed. On values : a=255;b=255. Expected 65025 found 65025.
All tests passed!
```

Рисунок 7 – Тестирование модуля умножителя

Далее на рисунке представлены временные диаграммы тестирования модуля умножителя.

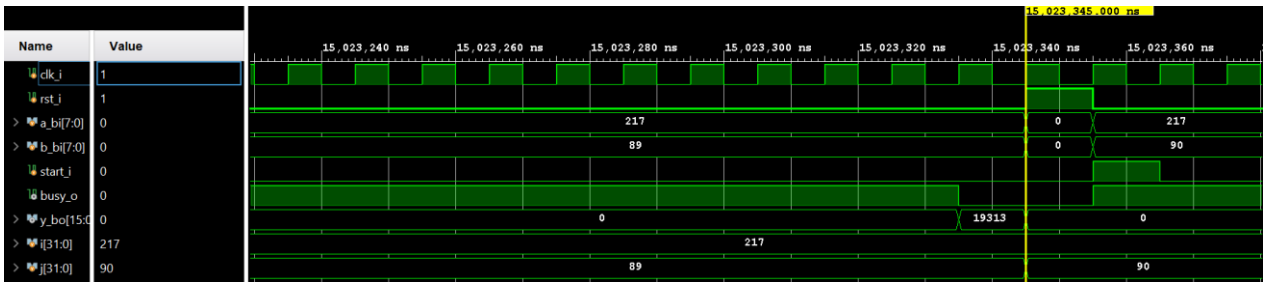

Рисунок 8 – Временная диаграмма тестирования модуля умножителя

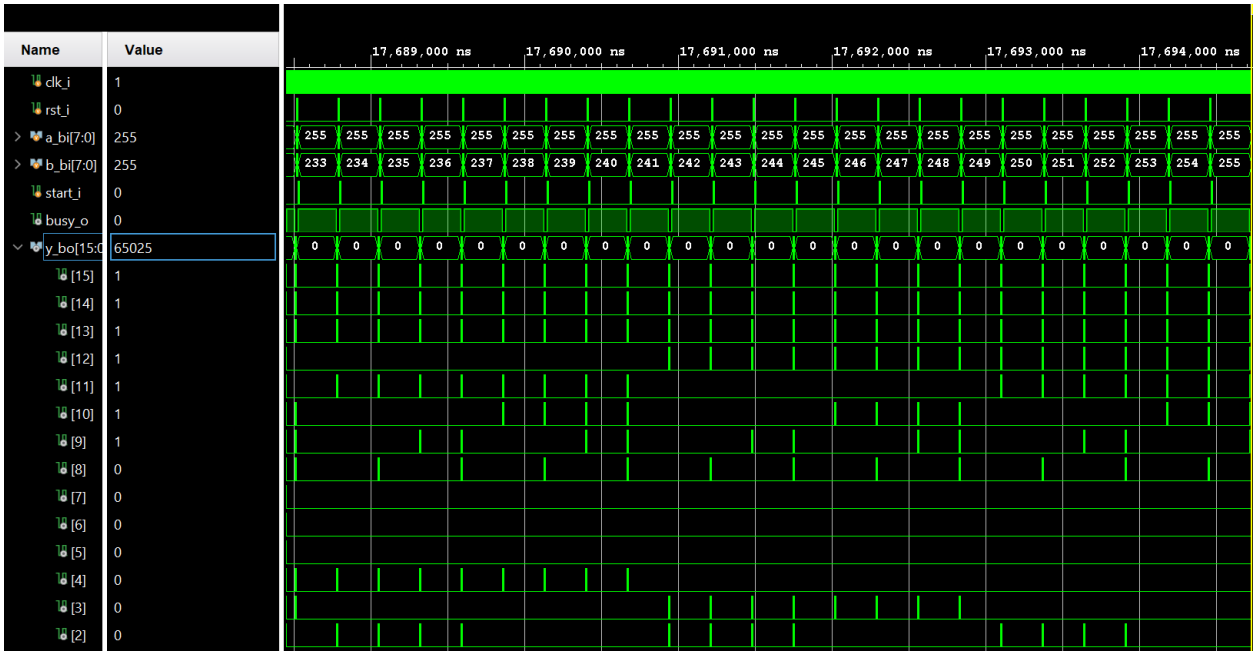

Рисунок 9 – Временная диаграмма тестирования модуля умножителя

2. Разработанный модуль **сумматора 8 бит**

Для модуля сумматора реализовано полное тестовое покрытие. На рисунке представлен скриншот с частью пройденных тестов. Ожидаемое значение рассчитываем с помощью встроенного сумматора +.

```
Test passed. On values : a=  2;b=135. Expected 137 found   137.
Test passed. On values : a=  2;b=136. Expected 138 found   138.
Test passed. On values : a=  2;b=137. Expected 139 found   139.
Test passed. On values : a=  2;b=138. Expected 140 found   140.
Test passed. On values : a=  2;b=139. Expected 141 found   141.
Test passed. On values : a=  2;b=140. Expected 142 found   142.
Test passed. On values : a=  2;b=141. Expected 143 found   143.
Test passed. On values : a=  2;b=142. Expected 144 found   144.
Test passed. On values : a=  2;b=143. Expected 145 found   145.
Test passed. On values : a=  2;b=144. Expected 146 found   146.
Test passed. On values : a=  2;b=145. Expected 147 found   147.
Test passed. On values : a=  2;b=146. Expected 148 found   148.
Test passed. On values : a=  2;b=147. Expected 149 found   149.
Test passed. On values : a=  2;b=148. Expected 150 found   150.
Test passed. On values : a=  2;b=149. Expected 151 found   151.
Test passed. On values : a=  2;b=150. Expected 152 found   152.
Test passed. On values : a=  2;b=151. Expected 153 found   153.
```

Рисунок 10 – Тестирование модуля сумматора 8 бит

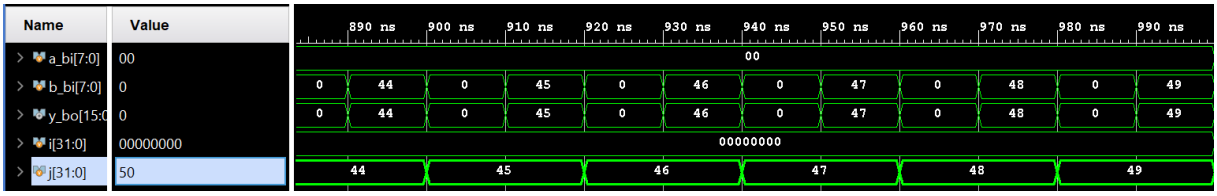

Рисунок 11 – Временная диаграмма тестирования модуля сумматора 8 бит

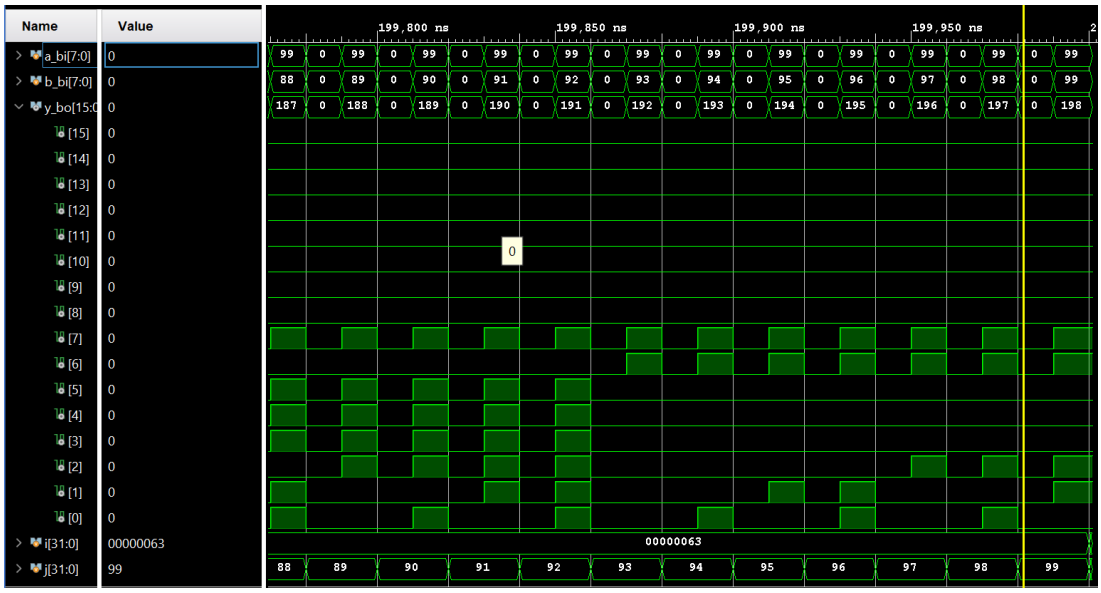

Рисунок 12 – Временная диаграмма тестирования модуля сумматора 8 бит

3. Разработанный модуль **квадратного корня**

Для модуля сумматора реализовано полное тестовое покрытие для 8-битного входа. На рисунке представлен скриншот с частью пройденных тестов. Для тестирования используем цикл, в котором перебираем значения от 0 до 15. Для формирования ожидаемого результата передаем переменную цикла, в качестве аргумента для излечения корня передаем число, умноженное само на себя.

```
Test passed. On values : a= 25. Expected   5 found   5.
Test passed. On values : a= 36. Expected   6 found   6.
Test passed. On values : a= 49. Expected   7 found   7.
Test passed. On values : a= 64. Expected   8 found   8.
Test passed. On values : a= 81. Expected   9 found   9.
Test passed. On values : a=100. Expected  10 found  10.
Test passed. On values : a=121. Expected  11 found  11.
Test passed. On values : a=144. Expected  12 found  12.
Test passed. On values : a=169. Expected  13 found  13.
Test passed. On values : a=196. Expected  14 found  14.
Test passed. On values : a=225. Expected  15 found  15.
All tests passed!
```

Рисунок 13 – Тестирование модуля квадратного корня

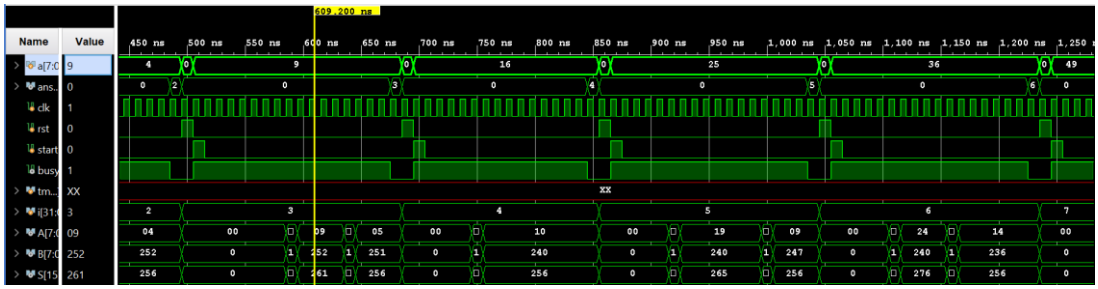

Рисунок 14 - Временная диаграмма тестирования модуля квадратного корня

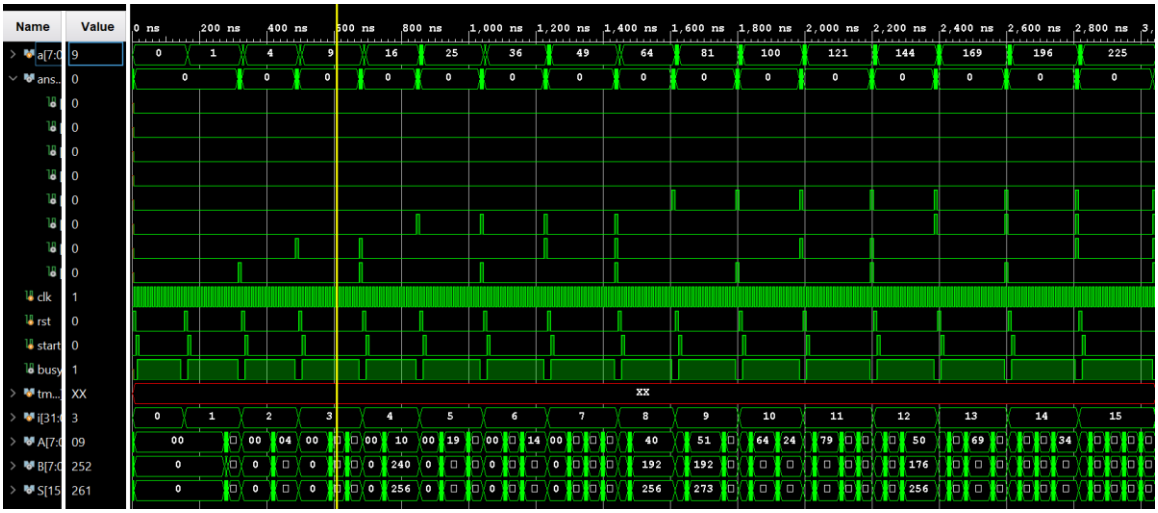

Рисунок 15 - Временная диаграмма тестирования модуля кубического корня

4. Разработанный модуль **кубического корня**

Для модуля сумматора реализовано полное тестовое покрытие для 8-битного входа. На рисунке представлен скриншот с частью пройденных тестов. Для тестирования используем цикл, в котором перебираем значения от 0 до 7 Для формирования ожидаемого результата передаем переменную цикла, в качестве аргумента для излечения корня передаем число, умноженное само на себя три раза.

```
Test passed. On values : a=  0. Expected   0 found 0.
Test passed. On values : a=  1. Expected   1 found 1.
Test passed. On values : a=  8. Expected   2 found 2.
Test passed. On values : a= 27. Expected   3 found 3.
Test passed. On values : a= 64. Expected   4 found 4.
Test passed. On values : a=125. Expected   5 found 5.
Test passed. On values : a=216. Expected   6 found 6.
All tests passed!
```

Рисунок 16 - Тестирование модуля кубического корня

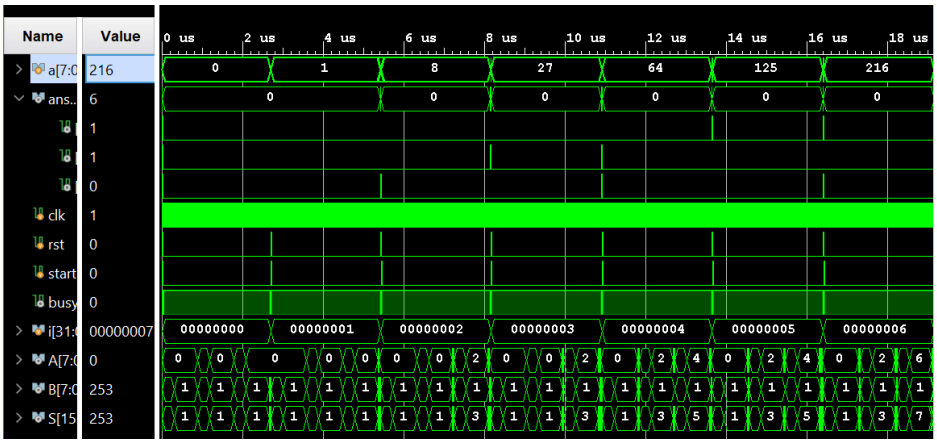

Рисунок 17 - Временная диаграмма тестирования модуля кубического корня

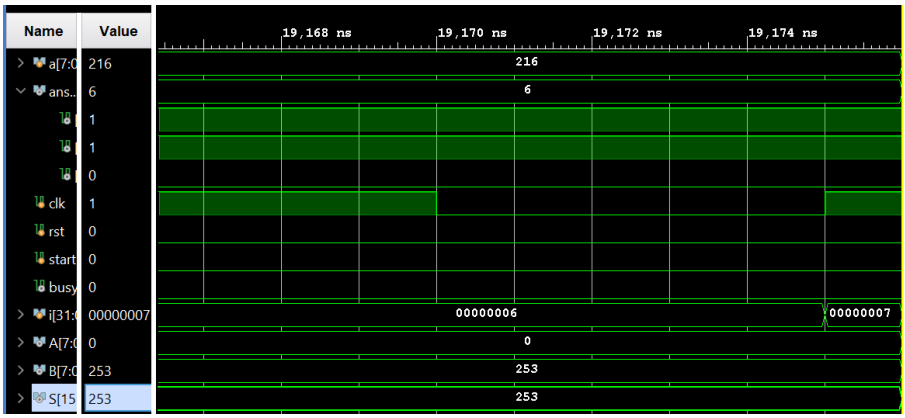

Рисунок 18 - Временная диаграмма тестирования модуля кубического корня

5. Разработанный модуль **функции** $y = \sqrt[3]{a} + \sqrt{b}$

Для модуля функции предусмотрено 10 тестов с различным набором входных данных. На рисунке представлен скриншот с частью пройденных тестов.

```
Test passed. On values : a=126;b= 26. Expected    10 found 10.
Test passed. On values : a=216;b=255. Expected    21 found 21.
Test passed. On values : a=255;b= 24. Expected    10 found 10.
Test passed. On values : a= 19;b= 81. Expected    11 found 11.
Test passed. On values : a= 96;b= 50. Expected    11 found 11.
Test passed. On values : a=  8;b=100. Expected    12 found 12.
Test passed. On values : a= 28;b=225. Expected    18 found 18.
Test passed. On values : a=  0;b=  1. Expected     1 found  1.
Test passed. On values : a=218;b=196. Expected    20 found 20.
Test passed. On values : a= 93;b= 10. Expected     7 found  7.
All tests passed!
```

Рисунок 19 - Тестирование модуля функции

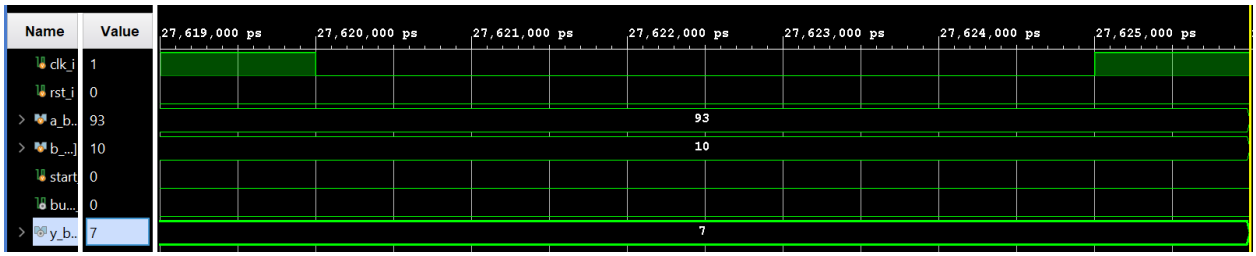

Рисунок 20 - Временная диаграмма тестирования модуля функции

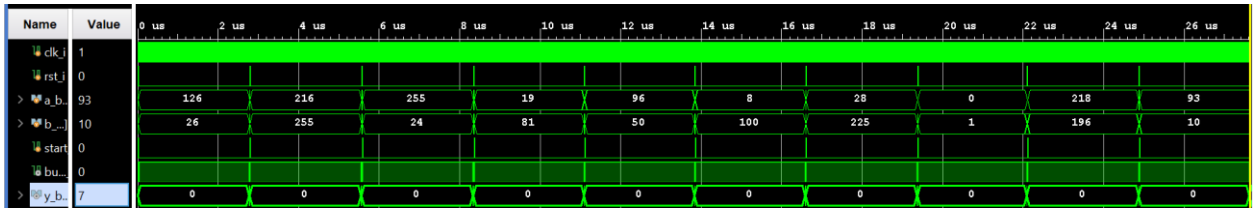

Рисунок 21 - Временная диаграмма тестирования модуля функции

7. Время вычисления результата при частоте тактового сигнала в 100 МГц

Тактовая частота в 100 МГ равна 1 синхроимпульс в 10 нс. Выставим ограничение на период в 5 нс. Временная диаграмма тестирования:

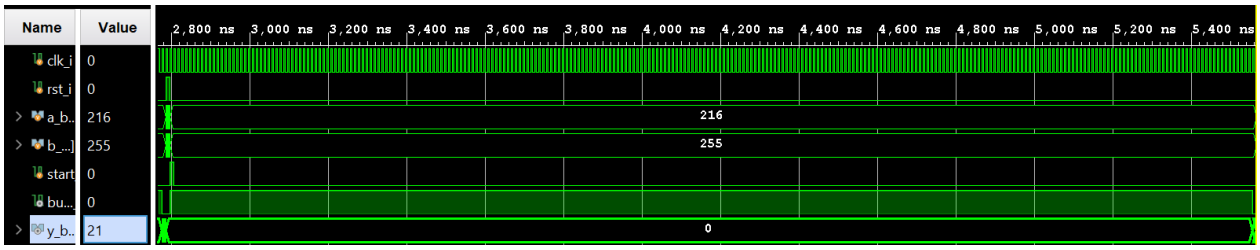

Рисунок 22 – Расчет частоты тактового сигнала при 100 МГц

По временной диаграмме видно, что на вычисление результата функции при тактовой частоте 100 МГц потребовалось 2600 нс.

8. Выводы по работе

В ходе выполнения данной лабораторной работы я столкнулась с некоторыми проблемами при разработке модуля функции, умножителя, квадратного и кубического корней. Я разработала модуль функции $y = \sqrt[3]{a} + \sqrt{b}$ с заданными ограничениями (2 сумматора и 1 умножитель). При этом оба сумматора находятся в модуле функции и передаются в нисходящие модули с помощью портов input и output. Первично возникли вопросы именно о передаче значений в другие модули из главного. Также возникли небольшие проблемы с реализацией модуля кубического корня. Алгоритм оказался очень трудоемким и потребовал много состояний для автомата в связи с особенностью разработки на Verilog.

Для проверки правильности реализации модулей были разработаны тесты с различными наборами данных. Все тесты пройдены успешно.