**Министерство образования и науки Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**(Университет ИТМО)**

О Т Ч Ё Т

**о выполнении курсовой работы**

Тема задания: Анализ алгоритмов и методов описания аппаратуры для реализации аппаратного расширителя процессора SCR1, производящего операцию извлечения логарифма из целого числа.

Выполнил: студент группы P41193

Хлыстун Д.С.

Проверил: Березина Екатерина Михайловна

Санкт-Петербург

2020

**СОДЕРЖАНИЕ**

[1 ПОИСК АЛГОРИТМОВ ВЫЧИСЛЕНИЯ ЛОГАРИФМА ЛЮБОЙ СТЕПЕНИ ИЗ ЧИСЛА И ИХ АНАЛИЗ С ЦЕЛЬЮ ВЫЯВИТЬ ОПТИМАЛЬНЫЕ ВАРИАНТЫ ДЛЯ РЕАЛИЗАЦИИ АППАРАТНОГО МОДУЛЯ ВЫЧИСЛЕНИЯ ЛОГАРИФМА ИЗ ЧИСЛА ТИПА INT 2](#_Toc44587154)

[1.1 Первый метод 3](#_Toc44587155)

[1.2 Второй метод 7](#_Toc44587156)

[2 РАЗРАБОТКА АППАРАТНОГО МОДУЛЯ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ СИ В СРЕДЕ VIVADO HLS. 11](#_Toc44587157)

[2.1 Разработка аппаратного блока на языке программирования Си 11](#_Toc44587158)

[2.2 Анализ реализованного варианта. 12](#_Toc44587160)

[3 UNIT-ТЕСТЫ В VIVADO 13](#_Toc44520973)

[3.1 Первое тестирование 13](#_Toc44520974)

[3.2 Второе тестирование 16](#_Toc44520975)

[3.3 Третье тестирование 18](#_Toc44520976)

4 ВНЕСЕНИЕ ИЗМЕНЕНИЙ В ПРОЕКТ ПРОЦЕССОРА SCR1 ДЛЯ ДОБАВЛЕНИЯ ВЫБРАННОГО РАНЕЕ МОДУЛЯ 21

[5 ТЕСТИРОВАНИЕ РАБОТОСПОСОБНОСТИ ДОБАВЛЕННОЙ КОМАНДЫ В СОСТАВЕ SCR1 24](#_Toc44520978)

[5.1 Тестирование частоты 24](#_Toc44520979)

[5.2 Unit-тест в составе SCR1 24](#_Toc44520980)

[ЗАКЛЮЧЕНИЕ 27](#_Toc44520981)

[ПРИЛОЖЕНИЕ 1 29](#_Toc44520983)

[ПРИЛОЖЕНИЕ 2 30](#_Toc44520984)

[ПРИЛОЖЕНИЕ 3 41](#_Toc44520985)

[ПРИЛОЖЕНИЕ 4 42](#_Toc44520985)

# 1 ПОИСК АЛГОРИТМОВ ВЫЧИСЛЕНИЯ ЛОГАРИФМА ЛЮБОЙ СТЕПЕНИ ИЗ ЧИСЛА И ИХ АНАЛИЗ С ЦЕЛЬЮ ВЫЯВИТЬ ОПТИМАЛЬНЫЕ ВАРИАНТЫ ДЛЯ РЕАЛИЗАЦИИ АППАРАТНОГО МОДУЛЯ ВЫЧИСЛЕНИЯ ЛОГАРИФМА ИЗ ЧИСЛА ТИПА INT

**Логарифм** положительного числа  по основанию   (обозначается ) — это показатель степени, в которую надо возвести , чтобы получить .

## **Первый метод**

Вычисление логарифмов довольно распространённая операция в цифровой обработке сигналов. Чаще приходится считать только свёртки (умножение с накоплением) и амплитуды с фазами. Как правило для вычисления логарифмов на FPGA применяется алгоритм CORDIC в гиперболическом варианте, требующий только таблицы и простых арифметических операций. Однако это не всегда бывает удобно, особенно если проект большой, кристалл маленький и начинаются проблемы с оптимизацией. В приведенном далее алгоритме не используется таблиц, но есть умножение, точнее возведение в квадрат. И поскольку схемотехнически возведение в квадрат проще общего случая умножения, возможно этот алгоритм представляет интерес для специализированных микросхем, хотя для FPGA разницы конечно нет.

Пусть есть число **X**. Требуется найти число **Y** такое, чтобы  .  
Положим так же что **X** лежит в интервале от 1 до 2. Это не слишком ограничивает общность, поскольку **X** всегда можно перевести в этот интервал умножением или делением на степень двойки. Для **Y** это будет означать добавление или вычитание целого числа, что делается легко. Итак, **X** лежит в интервале от 1 до 2. Тогда **Y** будет лежать в интервале от 0 до 1. Запишем **Y** как бесконечную двоичную дробь:

Коэффициенты  в этой записи есть ни что иное, как просто биты двоичного представления числа **Y**. Причём поскольку **Y** меньше 1, очевидно что .

Возведём первое уравнение в квадрат:   и как и ранее, запишем двоичное представление числа **2Y**. Очевидно, что

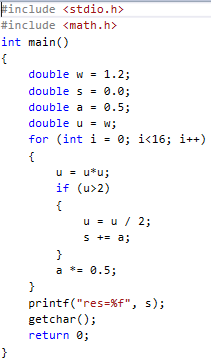
Т.е. биты    остались теми же, просто сдвинулись степени двойки. Бита  в представлении нет, потому что он равен нулю. Возможны два случая:

В первом случае в качестве нового значения **X** примем  , во втором-

В итоге задача свелась к прежней. Новое **X** опять лежит в интервале от 1 до 2, новое **Y** от 0 до 1. Но я узнал один бит результата. Делая такие же шаги в дальнейшем, можно получить сколько угодно битов **Y**.

После этого необходимо рассмотреть целочисленную реализацию.  
Пусть N-разрядные двоичные без знаковые числа, отображают интервал [0, 1]. Для удобства буду считать единицей число , а не , и соответственно двойкой число .

На листинге 1.1.1 представлен код программы такого алгоритма, написанный на языке С.



Листинг 1.1.1- код программы для первого алгоритма

Исследовав данный алгоритм, выявляется его серьезный недостаток- он может рассчитывать логарифм только лишь для значений, расположенных на интервале от [1;2].

В качестве примера посчитаем Как известно, любой логарифм от 1 равен 0. Удостоверимся в этом при помощи калькулятора(рис 1.1.2).

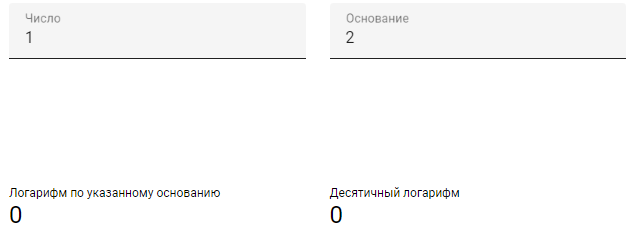


Рис. 1.1.2- расчет при помощи калькулятора.

Теперь посмотрим на результат вычисления программы (рис. 1.1.3).

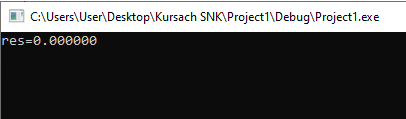


Рис. 1.1.3 – результат работы программы

Теперь вычислим .

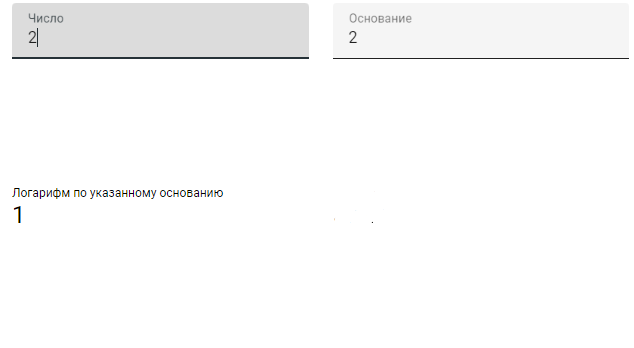


Рис. 1.1.4- расчет при помощи калькулятора.

И опять же вычислим при помощи программы (рис. 1.1.4).

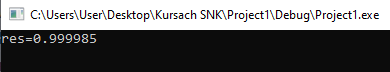


Рис. 1.1.4 –вычисление при помощи программы.

И, на всякий случай, проверим любое промежуточное значение, например- 1.5 (рис. 1.1.5).

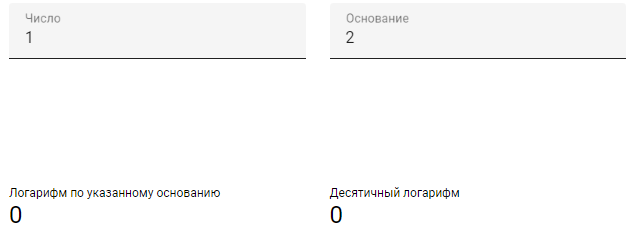


Рис. 1.1.5- расчет при помощи калькулятора.

Теперь посмотрим на результат вычисления программы (рис. 1.1.6).

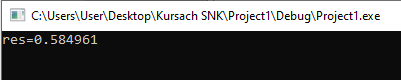


Рис. 1.1.6 – результат работы программы

Ну и в подтверждение слов, что данный метод не сможет посчитать логарифм из числа, которое больше двух, приведу еще пару скриншотов.

Вычислим логарифм из числа 2.1 (рис. 1.1.7).

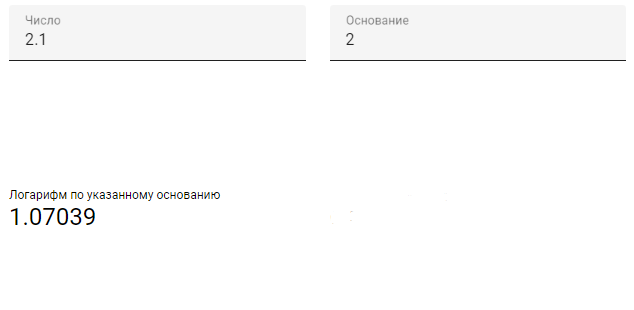


Рис. 1.1.7- расчет при помощи калькулятора.

Ну и то же самое вычисление при помощи программы (рис. 1.1.8).

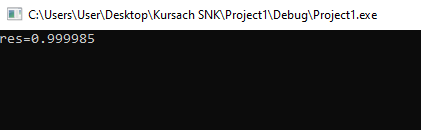
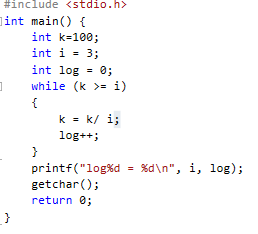


Рис. 1.1.8 – результат работы программы

Как можно заметить, погрешность ну уж слишком велика, соответственно можно сделать вывод, что алгоритм работает неверно и внедрять его куда-либо – заведомо гиблое дело.

## **Второй метод**

Второй метод был придуман непосредственно своей головой. Отталкиваясь от всем известного определения логарифма, был придуман новый алгоритм. Логарифм- это показатель степени, а если возвести число в определённую степень -это перемножить его само на себя несколько раз. То есть, чтобы взять логарифм по основанию от какого-либо числа, нужно это число делить на основание логарифма до тех пор, пока не перестанет происходить деление нацело. При этом, при каждом «удачном» делении на единицу будет наращиваться счетчик, значение которого в конечном итоге и будет результатом извлечения логарифма. Ниже приведен листинг программы для данного алгоритма (листинг 1.2.1).



Листинг 1.2.1 - код программы для второго алгоритма

К- это число, из которого нужно извлечь логарифм, i- основание логарифма, ну и непосредственно log- счетчик, который будет инкрементироваться, и, непосредственно, являться ответом.

Разберем работу алгоритма на примере кода программы из листинга.

В данном случае нужно найти целую часть . Есть цикл, который работает до тех пор, пока остаток от деления не станет меньше основания логарифма (то есть перестанет быть возможным деление нацело). В этом цикле происходит 2 действия- деление заданного числа на основание логарифма и при каждом успешном делении нацело- инкрементирование переменной log, которая является результатом. Ну и после выхода из цикла стоит вывод результата в консоль.

Сделаем пару тестовых вычислений.

Например посчитаем приведенный выше при помощи калькулятора и программы. Калькулятор выдал такой результат (рис. 1.2.2).

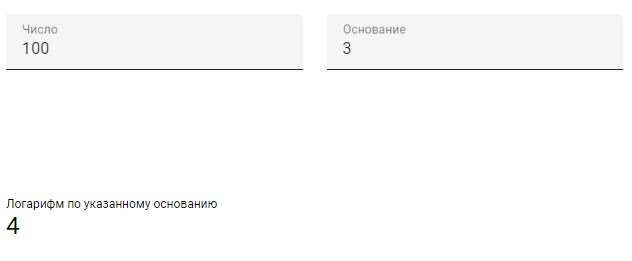


Рис. 1.2.2- расчет при помощи калькулятора.

Результат работы программы (рис. 1.2.3):

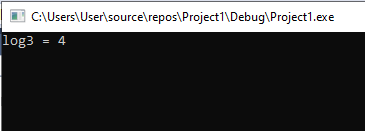


Рис. 1.2.3- результат работы программы

Ну и проверим еще какой-нибудь логарифм. Например, . Что выдаст калькулятор (рис. 1.2.4).

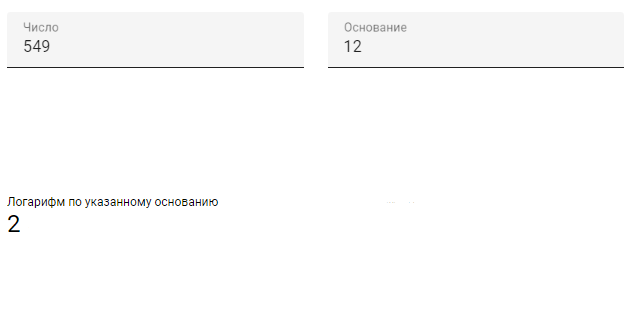


Рис. 1.2.2- расчет при помощи калькулятора.

Посмотрим на результат работы программы (рис.1.2.5).

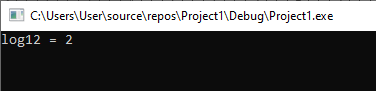


Рис. 1.2.5- результат работы программы

Все работает прекрасно, и, исходя из анализа работоспособности, простоты реализации и верности вычислений чашу весов в свою сторону склоняет второй алгоритм вычисления логарифма.

# 2 РАЗРАБОТКА АППАРАТНОГО МОДУЛЯ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ СИ В СРЕДЕ VIVADO HLS.

## **2.1 Разработка аппаратного блока на языке программирования Си**

Для разработки посредством синтеза высокого уровня использована система автоматизированного проектирования Vivado HLS. Она предназначена для создания цифровых устройств с применением языков высокого уровня.

Согласно алгоритму, был составлен программный код на языке программирования Си [5] (приложение 1). Он представляет из себя простой код на языке программирования высокого уровня, но был преобразован средой Vivado HLS в ходе синтеза в набор модулей \*.v (рисунок 2.1.). Код основного модуля представлен в приложении 2.

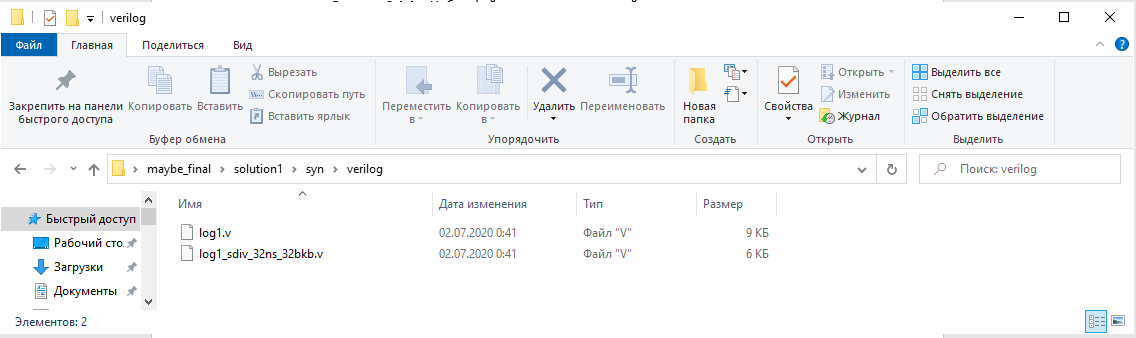


Рисунок 2.1.1 – Набор файлов, сгенерированный высокоуровневым синтезом

## **2.2 Анализ реализованного варианта.**

Для анализа реализованного аппаратного модуля использована среда разработки Vivado.

Необходимо добавить в проект тестирования сгенерированные средой Vivado HLS модули \*.v. После данных манипуляций следует запустить симуляцию, по окончании которой будет сгенерирована осциллограмма. Она является наглядным представлением правильной работы модуля, а также его производительности – предоставляет время выполнения извлечения логарифма исследуемым аппаратным блоком.

На рисунке 2.2 представлена осциллограмма, полученная после выполнения симуляции с модулем, разработанным посредством синтеза высокого уровня в среде Vivado HLS. В качестве числа для тестирования используется 549, основание логарифма будет 12. Ответ должен быть равен 2.