Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ Lab2\_Z1

Дисциплина: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Введение в Vitis HLS Tool CLI Flow

Выполнил студент гр. Шеметов С.А.

гр.5140901/21502

Руководитель, доцент Антонов А.П.

«\_\_» \_\_\_\_\_\_ 2023

Санкт-Петербург

2023

Оглавление

[1. Задание 3](#_Toc146937429)

[2. Исходный код функции 3](#_Toc146937430)

[3. Исходный код теста 3](#_Toc146937431)

[3.1. Моделирование 4](#_Toc146937432)

[4. Создание скрипта 5](#_Toc146937433)

[5. Сравнение результатов четырёх решений 6](#_Toc146937434)

[6. Анализ результатов 7](#_Toc146937435)

[7. Временные диаграммы 8](#_Toc146937436)

[8. Вывод 10](#_Toc146937437)

1. Задание

* Создать на языке С++ функцию (lab2\_z1.cpp) поиска факториала (без рекурсии).
  + Базовый тип данных: n – unsigned char; возвращаемое значение – unsigned int
* Создать на языке С++ тест (lab2\_z1\_test.cpp) для проверки работы функции.
* Создать скрипт автоматизирующий процесс создания проекта и работы с ним.
* Создать, провести исследование и сравнительный анализ четырёх аппаратных реализаций разработанного на языке С++ описания функции.
  + Микросхема: xa7a12tcsg325-1q
  + clock period 4; clock\_uncertainty 1 (для решения Solution 1)
  + clock period 8; clock\_uncertainty 1 (для решения Solution 2)
  + clock period 12; clock\_uncertainty 1 (для решения Solution 3)
  + clock period 40; clock\_uncertainty 1 (для решения Solution 4)

2. Исходный код функции

Исходный код синтезируемой функции приведен на рисунке 1.

*Изображение выглядит как программное обеспечение, Мультимедийное программное обеспечение, текст, Графическое программное обеспечение

Автоматически созданное описание*

*Рис.1. Исходный код функции lab2\_z1*

Функция принимает один аргумент-скаляр, вычисляет факториал в цикле и возвращает результат расчета.

3. Исходный код теста

Исходный код теста для проверки функции lab2\_z1, исправленный в соответствии с заданием, приведен на рисунке 2. Тест обеспечивает проверку корректной работы функции с помощью алгоритма с рекурсией и формирование признака успешного/неуспешного выполнения для каждого запуска функции.

3.1. Моделирование

Результаты моделирования исходного кода синтезируемой функции приведены на рисунке 3. Результаты моделирования показывают, что тест успешно пройден — так как условия теста выполняются (функция *main* возвращает 0) и результат расчета совпадает с ожидаемыми значениями.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

*Рис.2. Исходный код теста*

*Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание*

*Рис.3. Лог результата выполнения моделирования*

Для проверки правильности работы теста в него было внесено изменение в ожидаемый результат - expected = factorial(inN) + 1. Результат моделирования приведён на рисунке 4. Он показывает, что тест отрабатывает ошибку корректно.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

*Рис.4. Лог результата при ошибке в тесте*

4. Создание скрипта

Созданный скрипт для автоматизации работы с проектом представлен на рисунке 5.

Данный скрипт создаёт или открывает проект, предварительно очищая его, если он уже существует, командой *open\_project -reset lab2\_z1*. Затем добавляется файл с исходным кодом *add\_files ./source/lab2\_z1.cpp* и устанавливается функция верхнего уровня в иерархии *set\_top lab2\_z1*. Добавляется тестовый файл - *add\_files -tb ./source/lab2\_z1\_test.cpp*. Параметр *-tb* указывает на то, что файл является тестовым.

Далее создаётся первое решение *- open\_solution -reset "ex\_sol1"*. Для него назначается микросхема, период 4 нс и неопределённость 1 нс. После запускаются Си моделирование (*csim\_design -clean*), синтез (*csynth\_design*) и моделирование cosim (*cosim\_design -trace\_level all -tool xsim*).

Затем создаются остальные три решения. Для них в цикле задаётся период, микросхема и выполняются синтез и моделирование cosim.

Запуск скрипта выполняется в командной строке Vitis HLS 2021.2 Command Prompt с помощью команды *Vitis\_hls -f lab2\_z1.tcl*.

**Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание**

*Рис.5. Скрипт для работы с проектом*

5. Сравнение результатов четырёх решений

В данной работе были реализованы четыре решения с разной частотой, которые могут отличаться в зависимости от задаваемого параметра. Результаты сравнения параметров двух решений приведены на рисунке 6.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

*Рис.6. Сравнение четырёх решений*

Сравнение данных на рисунке 6 показывает, что решение 3 и 4 имеет одинаковые аппаратные затраты: 52 FF триггера и 95 LUT таблиц перекодировок. Решение 2 имеет – максимальные: 251 FF и 132 LUT.

Первое решение не укладывается в заданные период, так как Estimated + Uncertainty = 3.872 + 1 = 4.872 нс, что превышает 4 нс.

Третье и четвёртое решение имеют минимальный IImax составляющий 257 тактов. У второго решения – это значение составляет 767 тактов, а первое имеет максимальное значение 1532 такта.

6. Анализ результатов

На основе результатов сравнения решений была составлена электронная таблица и построен график, в котором для всех решений отображены: Iteration Interval и использованные ресурсы. Результаты представлены на рисунках 7–8.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

*Рис.7. Электронная таблица для четырёх решений*

Изображение выглядит как линия, диаграмма, График, снимок экрана

Автоматически созданное описание

*Рис.8. График для четырёх решений*

* Оптимальное решение по критерию максимальная производительность – решения 3 и 4
* Оптимальное решение по критерию минимальные аппаратные затраты – 3 и 4
* Оптимальное решение по критерию максимальная производительность и минимальные аппаратные затраты – 3 и 4.

Таким образом, если заданный период будет 12 нс и выше, то умножение будет укладываться в один период и решение будет иметь оптимальные значения для производительности и аппаратных затрат.

7. Временные диаграммы

На рисунках 9-12 приведены временные диаграммы для всех четырёх решений. На временной диаграмме приведено 3 цикла запуска аппаратной реализации созданной функции. Чем больше входное значение, тем большее число тактов требуется для формирования вывода.

Изображение выглядит как снимок экрана, текст, программное обеспечение, дисплей

Автоматически созданное описание

*Рис.9. Временная диаграмма для ex\_sol1*

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

*Рис.10. Временная диаграмма для ex\_sol2*

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

*Рис.11. Временная диаграмма для ex\_sol3*

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

*Рис.12. Временная диаграмма для ex\_sol4*

8. Вывод

В результате выполнения данной лабораторной работы были получены навыки работы с Vitis HLS Tool CLI. Для этого на языке С++ была написана программа для нахождения факториала и тест к ней. Для создания проекта и работы с ним был создан TCL скрипт, позволяющий задать необходимые параметры для нескольких решений и выполнить синтез и моделирование cosim для каждого из них. По окончании синтеза и моделирования cosim были проанализированы решения в соответствии с критериями оптимальности.