САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ПЕТРА ВЕЛИКОГО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Лабораторная работа lab4\_z1

Дисциплина:

«Проектирование реконфигурируемых гибридных вычислительных систем»

Тема: Введение в Pipeline for Perfomance

Выполнил:

Бараев Д. Р.

Группа: 3540901/02001

Преподаватель: А. П. Антонов

Санкт-Петербург

2021

**Оглавление**

[1. Задание 4](#_Toc86510243)

[2. Исходный код функции 4](#_Toc86510244)

[3. Исходный код теста 4](#_Toc86510245)

[4. Исходный код командного файла 5](#_Toc86510246)

[5. Результаты исследования и сравнение решений 6](#_Toc86510247)

[6. Анализ результатов 8](#_Toc86510248)

[7. Добавление программной конвейеризации для решений 9](#_Toc86510249)

[8. Исходный код и результат модифицированного теста 11](#_Toc86510250)

[9. Выводы 13](#_Toc86510251)

**Список иллюстраций**

[Рисунок 1 Исходный код функции (файл lab4\_z1.c) 4](#_Toc86510229)

[Рисунок 2 - Исходный код теста 5](#_Toc86510230)

[Рисунок 3 Исходный код командного файла для создания проекта (la4\_z1.tcl) 6](#_Toc86510231)

[Рисунок 4 - Сравнение полученных решений 7](#_Toc86510232)

[Рисунок 5 - Таблица данных для всех решений 8](#_Toc86510233)

[Рисунок 6 График зависимости данных для всех решений 8](#_Toc86510234)

[Рисунок 7 - Директива Pipeline для sol5 9](#_Toc86510235)

[Рисунок 8 - Директива Pipeline для sol6 9](#_Toc86510236)

[Рисунок 9 - Сравнение отчетов для pipeline решений sol5 и sol6 9](#_Toc86510237)

[Рисунок 10 - Сравнение результатов всех решений в виде таблицы 10](#_Toc86510238)

[Рисунок 11 - График сравнения результатов всех решений 10](#_Toc86510239)

[Рисунок 12 - Параметры ПК (Частота = 2.7 Гц) 11](#_Toc86510240)

[Рисунок 13 - Исходный код модифицированного теста для ПК 12](#_Toc86510241)

[Рисунок 14 - Временные показатели для модифицированного теста 13](#_Toc86510242)

1. Задание

Задание описано в файле «Задание lab4\_z1.docx», лежащее в рабочей папке.

1. Исходный код функции

Исходный код синтезируемой функции приведен на рисунке 1.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 Исходный код функции (файл lab4\_z1.c)

Функция принимает два аргумента массива типа int — вычисляет скалярное произведение входных массивов и возвращает результат.

1. Исходный код теста

Исходный код теста для проверки функции lab4\_z1 приведен на рисунке 2. Тест обеспечивает проверку корректной функции.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 - Исходный код теста

1. Исходный код командного файла

На рисунке 3 представлен текст команд для автоматизированного создания следующих вариантов аппаратной реализаций:

1. Для sol1 задается clock period 6: clock uncertainty 0.1
2. Для sol2 задается clock period 8. clock uncertainty 0.1
3. Для sol3 задается clock period 10. clock uncertainty 0.1
4. Для sol4 задается clock period 12. clock uncertainty 0.1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 Исходный код командного файла для создания проекта (la4\_z1.tcl)

1. Результаты исследования и сравнение решений

На рисунке 4 представлено сравнение из Vivado HLS GUI по аппаратным ресурсам, требуемых для реализации синтезируемой функции, и временным параметрам.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 4 - Сравнение полученных решений

Target – планируемое время на один такт.

Estimated – оценочное время.

Latency (cycle) – количество тактов latency за один цикл.

Latency (absolute) – время затраченное на latency.

1. Анализ результатов

На рисунке 5 представлена таблица с параметрами для всех решений, где рассчитывается Latency в нс.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 5 - Таблица данных для всех решений

На рисунке 6 представлен график данных для сравнения всех решений.

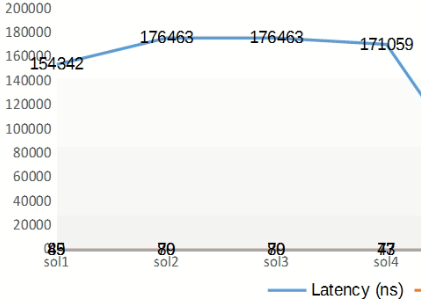


Рисунок 6 График зависимости данных для всех решений

Исходя из результатов, видно, что наилучшее решение было для sol1, поэтому для решений sol5 и sol6 с использованием конвейеризации заданный период будет равен 6нс. На приведенных ниже рисунках представлены директивы для добавления pipeline в решения.

1. Добавление программной конвейеризации для решений

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 - Директива Pipeline для sol5

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 - Директива Pipeline для sol6

На рисунке 9 представлено сравнение отчетов двух решений после добавление pipeline.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 9 - Сравнение отчетов для pipeline решений sol5 и sol6

На рисунке 9 видно, что между решениями sol5 и sol6 отсутствует разница почти по все параметрам.

На рисунке 10 представлена таблица сравнения всех решений.

На рисунке 11 представлен график с временными и аппаратными затратами.

Смотря на результаты, приведенные в рисунках, можно сделать вывод, что добавление Pipeline значительно улучшает производительность системы, но при этом увеличивает потребление аппаратных ресурсов.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 10 - Сравнение результатов всех решений в виде таблицы

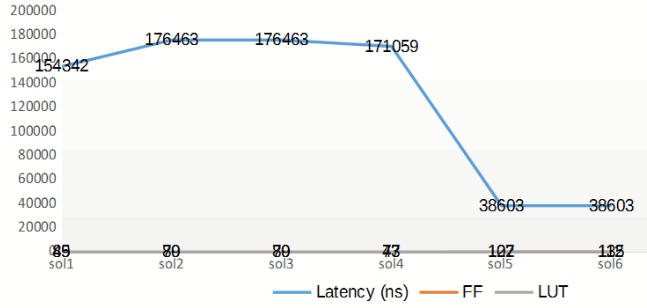


Рисунок 11 - График сравнения результатов всех решений

1. Исходный код и результат модифицированного теста

На рисунке 13 представлен исходный код модифицированного теста для проверки функции lab4\_z1. Тест обеспечивает проверку производительности функции (Компилятор gcc-9.3.0).

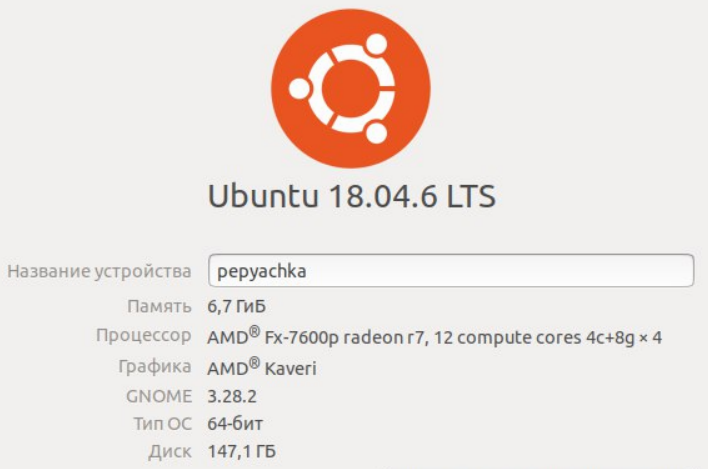


Рисунок 12 - Параметры ПК (Частота = 2.7 Гц)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13 - Исходный код модифицированного теста для ПК

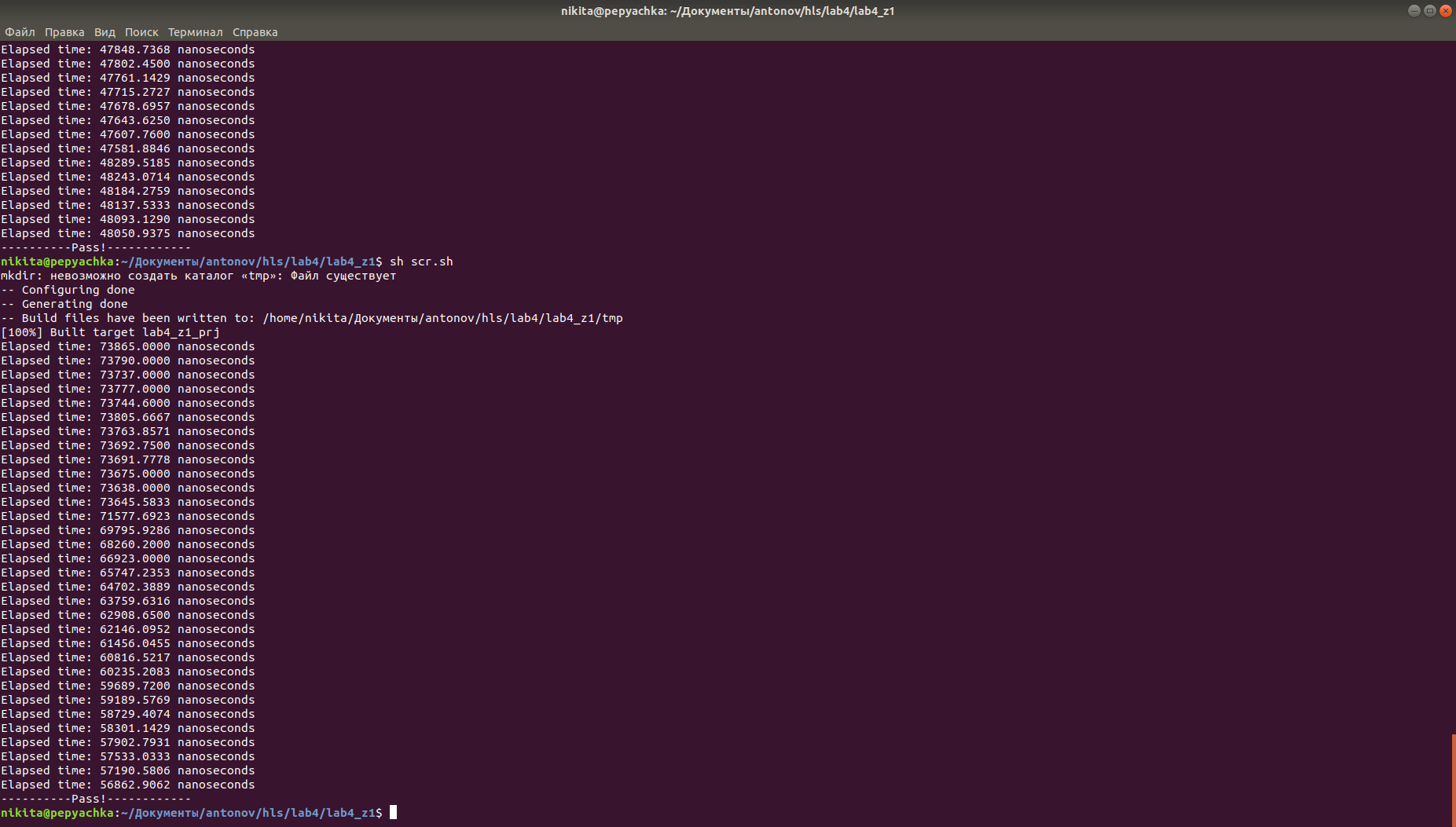


Рисунок 14 - Временные показатели для модифицированного теста

1. Выводы

В ходе данной работы была изучена возможность добавления pipeline-директивы для синтезируемой функции. Был произведен сравнительный анализ между решением без добавления и с добавлением pipeline. Также было произведено сравнение временных показателей между решением полученным Vivado HLS и тестированием решения на ПК. Как видно из результатов решением полученное на ПК быстрее, чем решением после синтезирования в Vivado HLS.