САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ПЕТРА ВЕЛИКОГО

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

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Лабораторная работа lab4\_z3

Дисциплина:

«Проектирование реконфигурируемых гибридных вычислительных систем»

Тема: Введение в Pipeline of Perfomance Dataflow

Выполнил:

Дроздов Н.Д.

Группа: 3540901/02001

Преподаватель: А. П. Антонов

Санкт-Петербург

2021

**Оглавление**

[1. Задание 4](#_Toc89958120)

[2. Исходный код функции 4](#_Toc89958121)

[3. Исходный код теста 5](#_Toc89958122)

[4. Исходный код командного файла 6](#_Toc89958123)

[5. Результаты исследования и сравнение решений 6](#_Toc89958124)

[6. Анализ результатов 8](#_Toc89958125)

[7. Исходный код и результат модифицированного теста 9](#_Toc89958126)

[8. Добавление dataflow-конвейеризации для решения 11](#_Toc89958127)

[8.1 Dataflow с FIFO memory buffer 11](#_Toc89958128)

[8.2 Dataflow с ping-pong memory buffer 15](#_Toc89958129)

[9. Исходный код и результат модифицированного теста 17](#_Toc89958130)

[10. Выводы 19](#_Toc89958131)

**Список иллюстраций**

[Рисунок 1- Исходный код функций foo\_b и foo\_m (файл lab4\_z3.c) 4](#_Toc89957718)

[Рисунок 2 - Исходный код теста (файл lab4\_z3\_test.c) 5](#_Toc89957719)

[Рисунок 3 - Исходный код командного файла для создания проекта (la4\_z3.tcl) 6](#_Toc89957720)

[Рисунок 4 - Сравнение полученных решений 7](#_Toc89957721)

[Рисунок 5 - Таблица данных для всех решений 8](#_Toc89957722)

[Рисунок 6 - График зависимости данных для всех решений 8](#_Toc89957723)

[Рисунок 7 - Schedule viewer для функции foo\_b 9](#_Toc89957724)

[Рисунок 8 - Resource viewer для функции foo\_b 9](#_Toc89957725)

[Рисунок 9 - Параметры ПК (Частота = 2.7 Гц) 9](#_Toc89957726)

[Рисунок 10 - Исходный код модифицированного теста для ПК 10](#_Toc89957727)

[Рисунок 11 - Временные показатели для модифицированного теста для foo\_b 11](#_Toc89957728)

[Рисунок 12 - Cинтезирование функции foo\_m для FIFO memory buffers 12](#_Toc89957729)

[Рисунок 13 - Schedule viewer для функции foo\_m 12](#_Toc89957730)

[Рисунок 14 - Resource viewer для функции foo\_m 12](#_Toc89957731)

[Рисунок 15 - Временная диаграмма для функции foo\_m 13](#_Toc89957732)

[Рисунок 16 - Dataflow viewer FIFO для функции foo\_m 14](#_Toc89957733)

[Рисунок 17 - Cинтезирование функции foo\_m для ping-pong memory buffers 15](#_Toc89957734)

[Рисунок 18 - Schedule viewer для ping-pong memory buffers 15](#_Toc89957735)

[Рисунок 19 - Resource viewer для ping-pong memory buffers 16](#_Toc89957736)

[Рисунок 20 - Dataflow viewer ping-pong memory buffers для функции foo\_m 16](#_Toc89957737)

[Рисунок 21 - Параметры ПК (Частота = 2.7 Гц) 17](#_Toc89957738)

[Рисунок 22 - Исходный код модифицированного теста для ПК 18](#_Toc89957739)

[Рисунок 23 - Временные показатели для модифицированного теста для foo\_m 19](#_Toc89957740)

1. Задание

Задание описано в файле «Задание lab4\_z3.docx», лежащее в рабочей папке.

1. Исходный код функции

Исходный код синтезируемых функций foo\_b и foo\_m приведены на рисунке 1.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1- Исходный код функций foo\_b и foo\_m (файл lab4\_z3.c)

Функции принимают четыре аргумента массива типа int — вычисляют для входного массива произведение на элементы второго массива, и записывают результаты в выходные массивы.

1. Исходный код теста

Исходный код теста для проверки функций foo\_b и foo\_m приведен на рисунке 2. Тест обеспечивает проверку корректной работы функций.

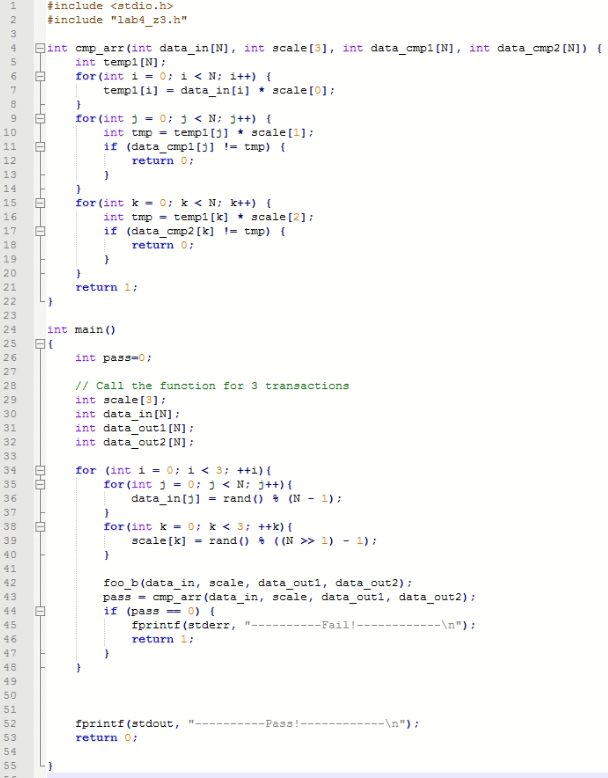


Рисунок 2 - Исходный код теста (файл lab4\_z3\_test.c)

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 стол

Автоматически созданное описание

Рисунок 3 - Исходный код командного файла для создания проекта (la4\_z3.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.

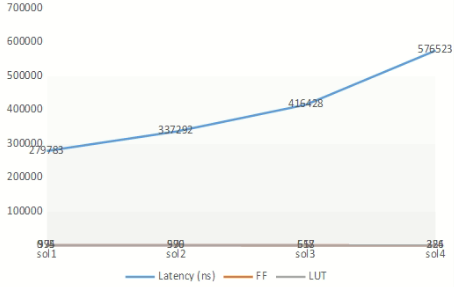


Рисунок 6 - График зависимости данных для всех решений

Исходя из результатов, видно, что наилучшее решение - sol3 с clock period 10 нс. У этого решения хороший результат по времени и почти самое низкое потребление аппаратных ресурсов.

На рисунке 7 и 8 представлены Schedule и Resource Viewer.

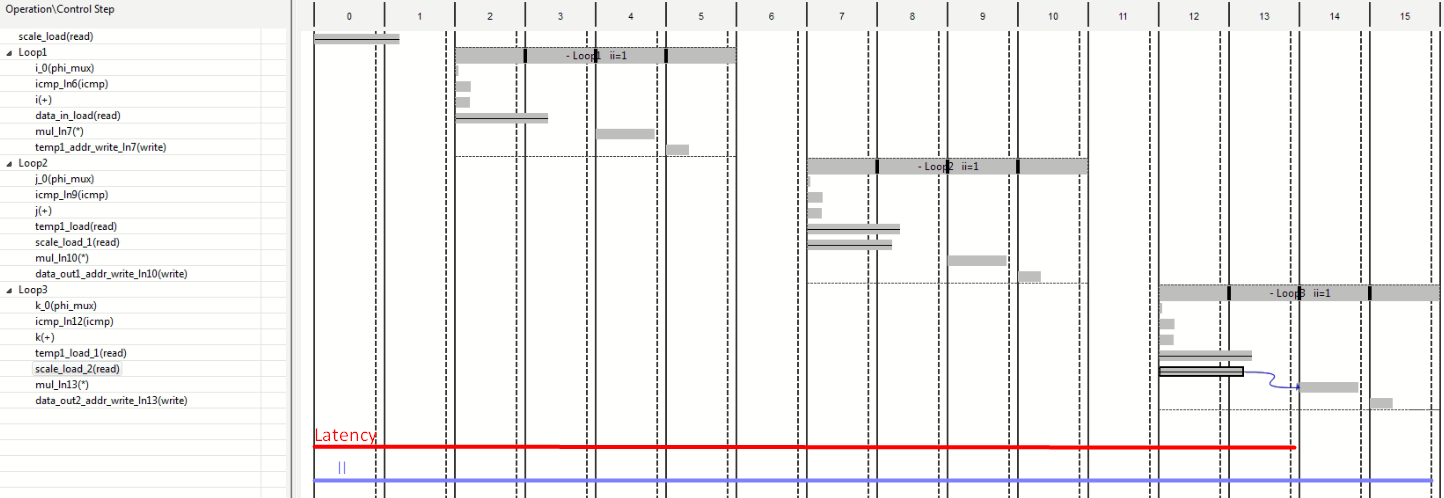


Рисунок 7 - Schedule viewer для функции foo\_b

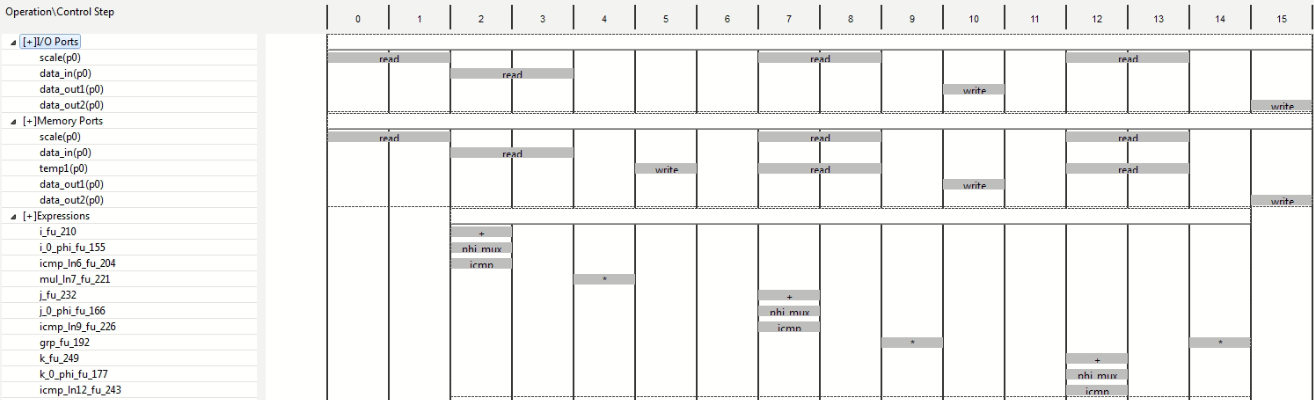


Рисунок 8 - Resource viewer для функции foo\_b

1. Исходный код и результат модифицированного теста

На рисунке 10 представлен исходный код модифицированного теста для проверки функции foo\_m, который используется также и при проверке функции foo\_b. Тест обеспечивает проверку производительности функции (Компилятор gcc-9.3.0).

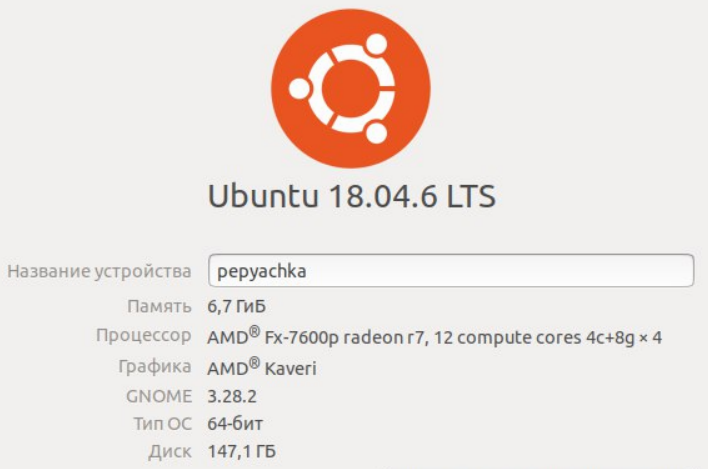


Рисунок 9 - Параметры ПК (Частота = 2.7 Гц)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 - Исходный код модифицированного теста для ПК

На рисунке 11 представлены результаты запуска функции на ПК. Как видно из рисунка, среднее время выполнения функции после 32 итерации равно 262050.41 нс, что в 1.6 раз быстрее, чем решение, полученное при синтезировании функции.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 - Временные показатели для модифицированного теста для foo\_b

1. Добавление dataflow-конвейеризации для решения

8.1 Dataflow с FIFO memory buffer

На рисунке 12 представлен результат синтезирования функции *foo\_m* для FIFO memory buffers в виде *Perfomance* и *Utilization Estimates* соответственно.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 12 - Cинтезирование функции foo\_m для FIFO memory buffers

Далее на рисунке 13 и 14 представлены Schedule и Resource viewer соотвственно.

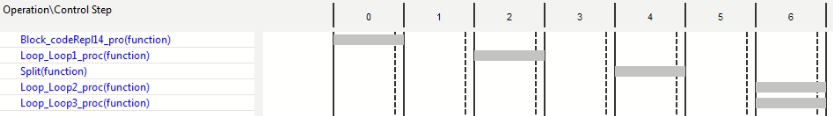


Рисунок 13 - Schedule viewer для функции foo\_m

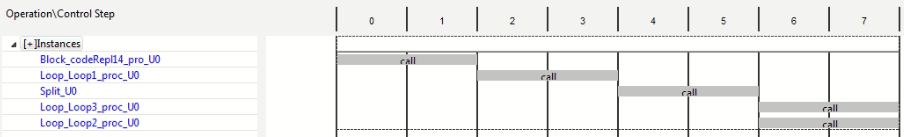


Рисунок 14 - Resource viewer для функции foo\_m

На рисунке 15 представлена временная диаграмма для решения с несколькими тактами работы функции после выполнения C/RTL Сosimulation.

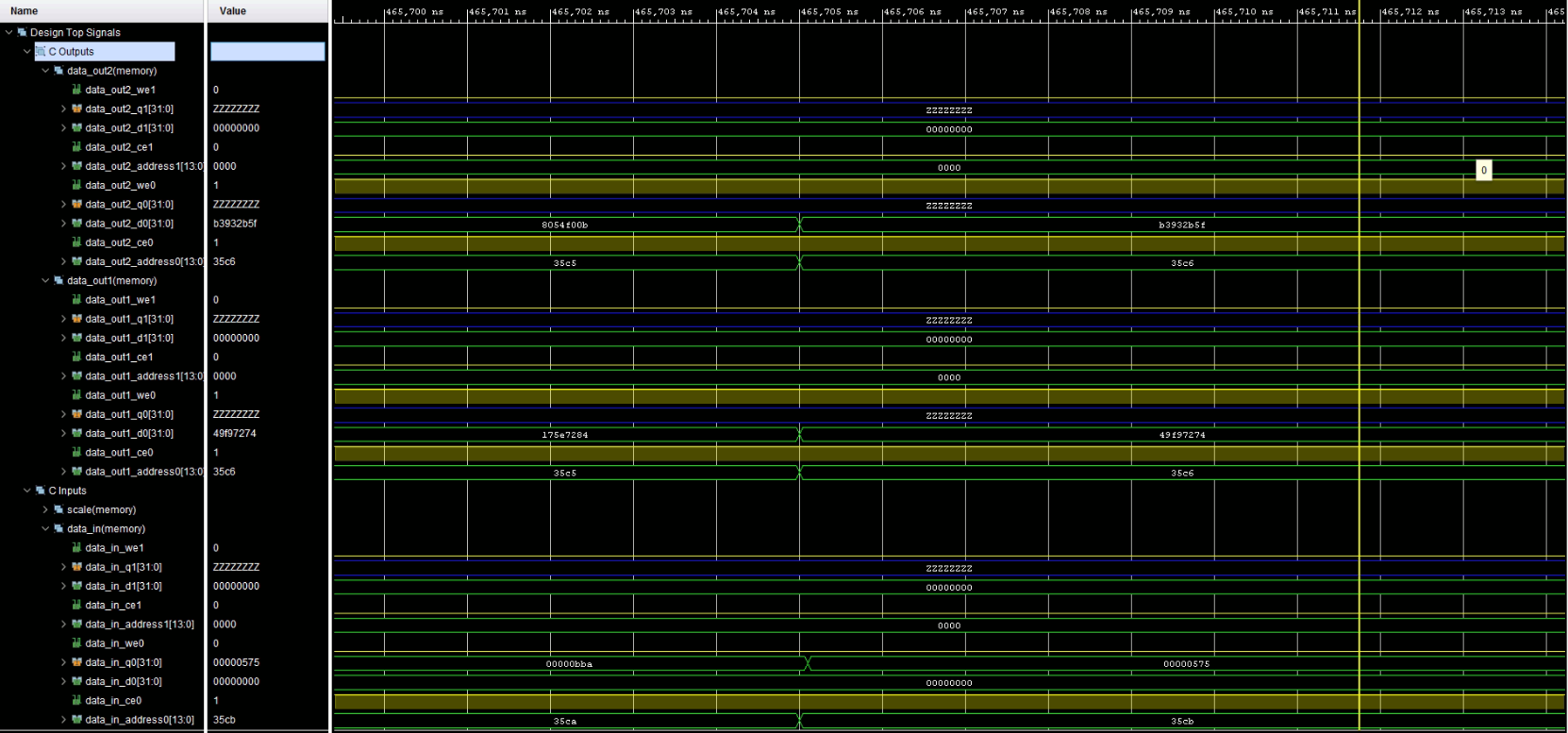


Рисунок 15 - Временная диаграмма для функции foo\_m

На рисунке 16 приведен Dataflow Viewer для случая FIFO the memory buffers.

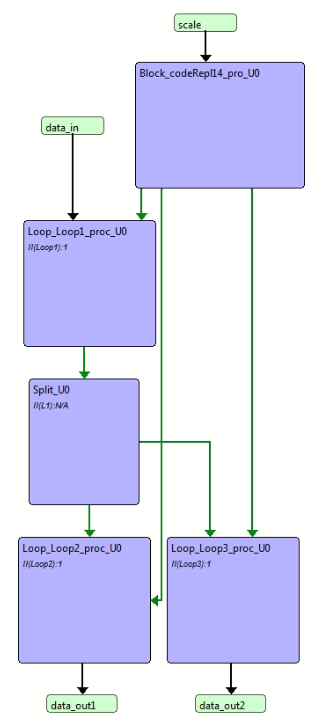


Рисунок 16 - Dataflow viewer FIFO для функции foo\_m

* 1. Dataflow с ping-pong memory buffer

На рисунке 17 представлен результат синтезирования функции **foo\_m** для **ping-pong memory buffers** в виде *Perfomance и Utilization Estimates* соответственно.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 17 - Cинтезирование функции foo\_m для ping-pong memory buffers

Как видно из результатов, для ping-pong memory buffer результат стал хуже по времени и по аппаратным ресурсам в сравнении с предыдущим решением.

На рисунке 18 и 19 приведены Schedule и Resource viewer соответственно.

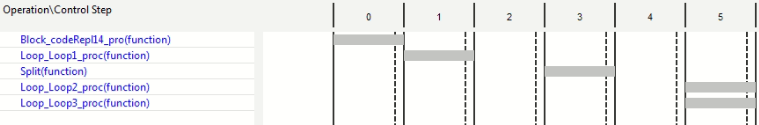


Рисунок 18 - Schedule viewer для ping-pong memory buffers

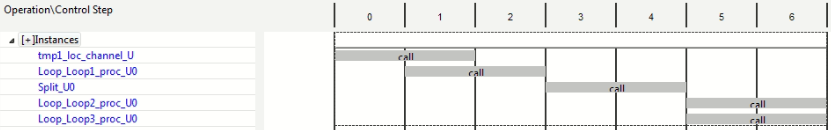


Рисунок 19 - Resource viewer для ping-pong memory buffers

На рисунке 20 и 21 приведен *Dataflow Viewer* для решения **ping-pong memory buffer.**

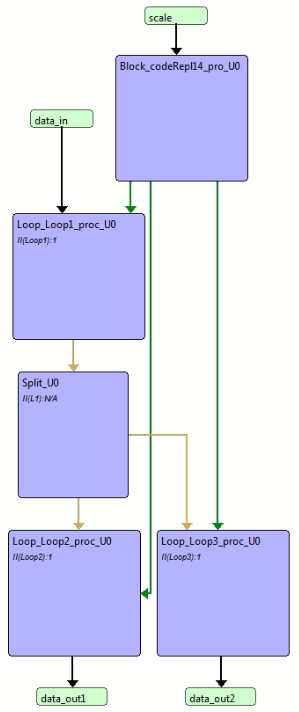


Рисунок 20 - Dataflow viewer ping-pong memory buffers для функции foo\_m

1. Исходный код и результат модифицированного теста

На рисунке 22 представлен исходный код модифицированного теста для проверки функции foo\_m, который используется также и при проверке функции foo\_b. Тест обеспечивает проверку производительности функции (Компилятор gcc-9.3.0).

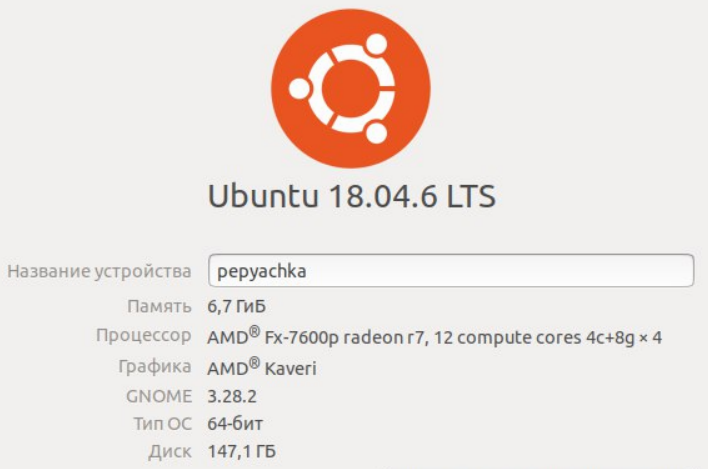


Рисунок 21 - Параметры ПК (Частота = 2.7 Гц)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 22 - Исходный код модифицированного теста для ПК

На рисунке 23 представлены результаты запуска функции на ПК.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 23 - Временные показатели для модифицированного теста для foo\_m

1. Выводы

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