САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ПЕТРА ВЕЛИКОГО

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

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Лабораторная работа lab4\_z5

Дисциплина:

«Проектирование реконфигурируемых гибридных вычислительных систем»

Тема: Введение в Pipeline of Perfomance Dataflow

Выполнил:

Бараев Д. Р.

Группа: 3540901/02001

Преподаватель: А. П. Антонов

Санкт-Петербург

2021

**Оглавление**

[1. Задание 4](#_Toc90064938)

[2. Исходный код функции 4](#_Toc90064939)

[3. Исходный код теста 5](#_Toc90064940)

[4. Исходный код командного файла 6](#_Toc90064941)

[5. Результаты исследования и сравнение решений 6](#_Toc90064942)

[6. Добавление dataflow-конвейеризации для решения 8](#_Toc90064943)

[6.1 Dataflow с FIFO memory buffer 8](#_Toc90064944)

[6.2 Dataflow с ping-pong memory buffer 11](#_Toc90064945)

[7. Исходный код и результат модифицированного теста 13](#_Toc90064946)

[8. Выводы 15](#_Toc90064947)

**Список иллюстраций**

[Рисунок 1- Исходный код функций foo\_b и foo\_m (файл lab4\_z5.c) 4](#_Toc90064918)

[Рисунок 2 - Исходный код теста (файл lab4\_z5\_test.c) 5](#_Toc90064919)

[Рисунок 3 - Исходный код командного файла для создания проекта (la4\_z5.tcl) 6](#_Toc90064920)

[Рисунок 4 – Результат синтезирования функции 7](#_Toc90064921)

[Рисунок 5 - Schedule viewer для функции foo\_b 8](#_Toc90064922)

[Рисунок 6 - Cинтезирование функции foo\_m для FIFO memory buffers 9](#_Toc90064923)

[Рисунок 7 - Schedule viewer для функции foo\_m 9](#_Toc90064924)

[Рисунок 8 - Resource viewer для функции foo\_m 9](#_Toc90064925)

[Рисунок 9 - Временная диаграмма для функции foo\_m 10](#_Toc90064926)

[Рисунок 10 - Dataflow viewer FIFO для функции foo\_m 10](#_Toc90064927)

[Рисунок 11 - Cинтезирование функции foo\_m для ping-pong memory buffers 11](#_Toc90064928)

[Рисунок 12 - Schedule viewer для ping-pong memory buffers 12](#_Toc90064929)

[Рисунок 13 - Resource viewer для ping-pong memory buffers 12](#_Toc90064930)

[Рисунок 14 - Dataflow viewer ping-pong memory buffers для функции foo\_m 12](#_Toc90064931)

[Рисунок 15 - Параметры ПК (Частота = 2.7 Гц) 13](#_Toc90064932)

[Рисунок 16 - Исходный код модифицированного теста для ПК 14](#_Toc90064933)

[Рисунок 17 - Временные показатели для модифицированного теста для foo\_m 15](#_Toc90064934)

1. Задание

Задание описано в файле «Задание lab4\_z5.docx», лежащее в рабочей папке.

1. Исходный код функции

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

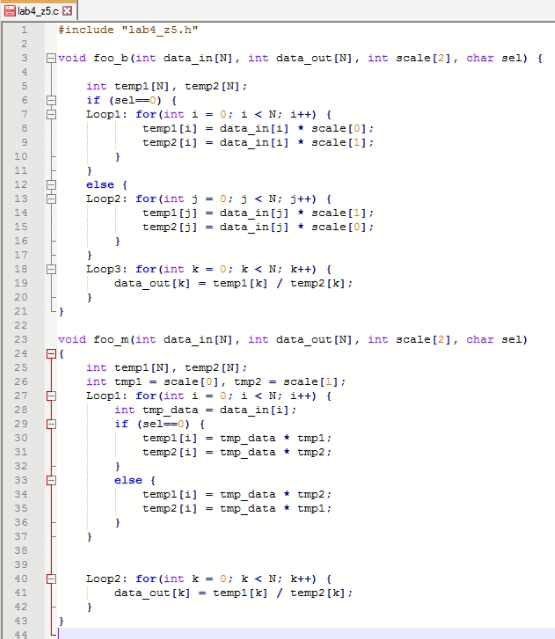


Рисунок 1- Исходный код функций foo\_b и foo\_m (файл lab4\_z5.c)

Функции принимают три аргумента массива типа int — вычисляют для входного массива произведение на элементы второго массива, и записывают результаты в выходной массив.

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.

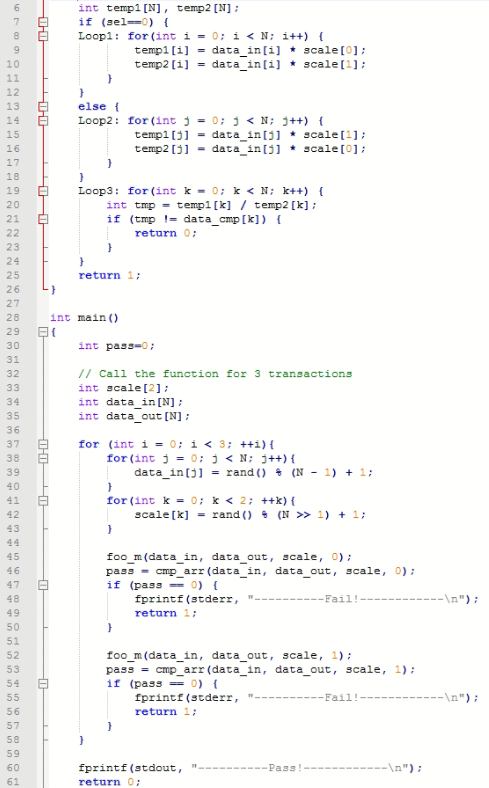


Рисунок 2 - Исходный код теста (файл lab4\_z5\_test.c)

1. Исходный код командного файла

На рисунке 3 представлен текст команд для автоматизированного создания следующего варианта аппаратной реализации:

1. Для sol1 задается clock period 10: clock uncertainty 0.1

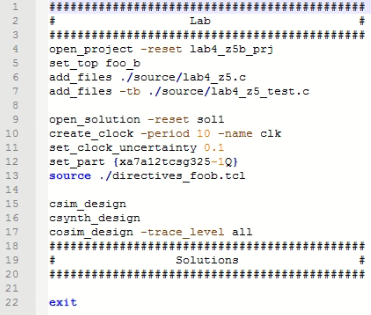


Рисунок 3 - Исходный код командного файла для создания проекта (la4\_z5.tcl)

1. Результаты исследования и сравнение решений

На рисунке 4 представлены результаты из Vivado HLS GUI по аппаратным ресурсам, требуемых для реализации синтезируемой функции, и временным параметрам. После синтезирования время Latency составляет 277900.7 нс.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 4 – Результат синтезирования функции

Target – планируемое время на один такт.

Estimated – оценочное время.

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

Latency (absolute) – время затраченное на latency.

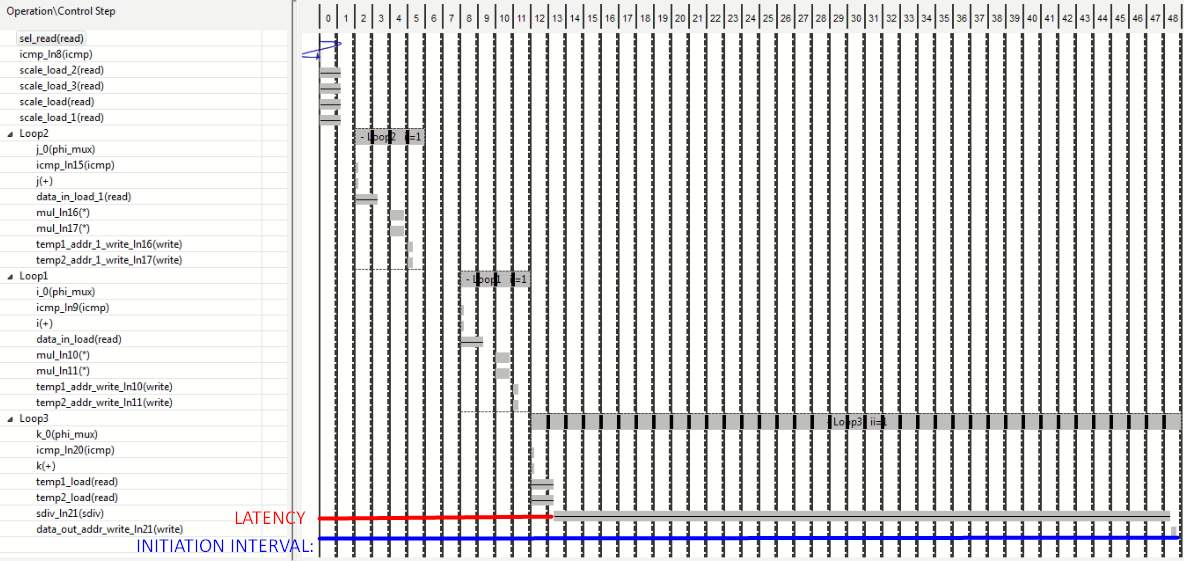


Рисунок 5 - Schedule viewer для функции foo\_b

1. Добавление dataflow-конвейеризации для решения

6.1 Dataflow с FIFO memory buffer

На рисунке 6 приведен результат синтезирования функции *foo\_m* для FIFO memory buffers в виде *Perfomance* и *Utilization Estimates* соответственно. После запуска синтезирования время Latency составляет 139119.75 нс.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 6 - Cинтезирование функции foo\_m для FIFO memory buffers

Далее на рисунке 7 и 8 представлены Schedule и Resource viewer соотвственно.

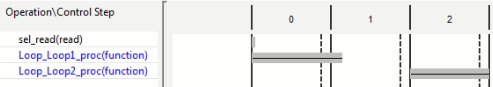


Рисунок 7 - Schedule viewer для функции foo\_m

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 8 - Resource viewer для функции foo\_m

На рисунке 9 представлена временная диаграмма для решения с несколькими тактами работы функции после выполнения C/RTL Сosimulation.

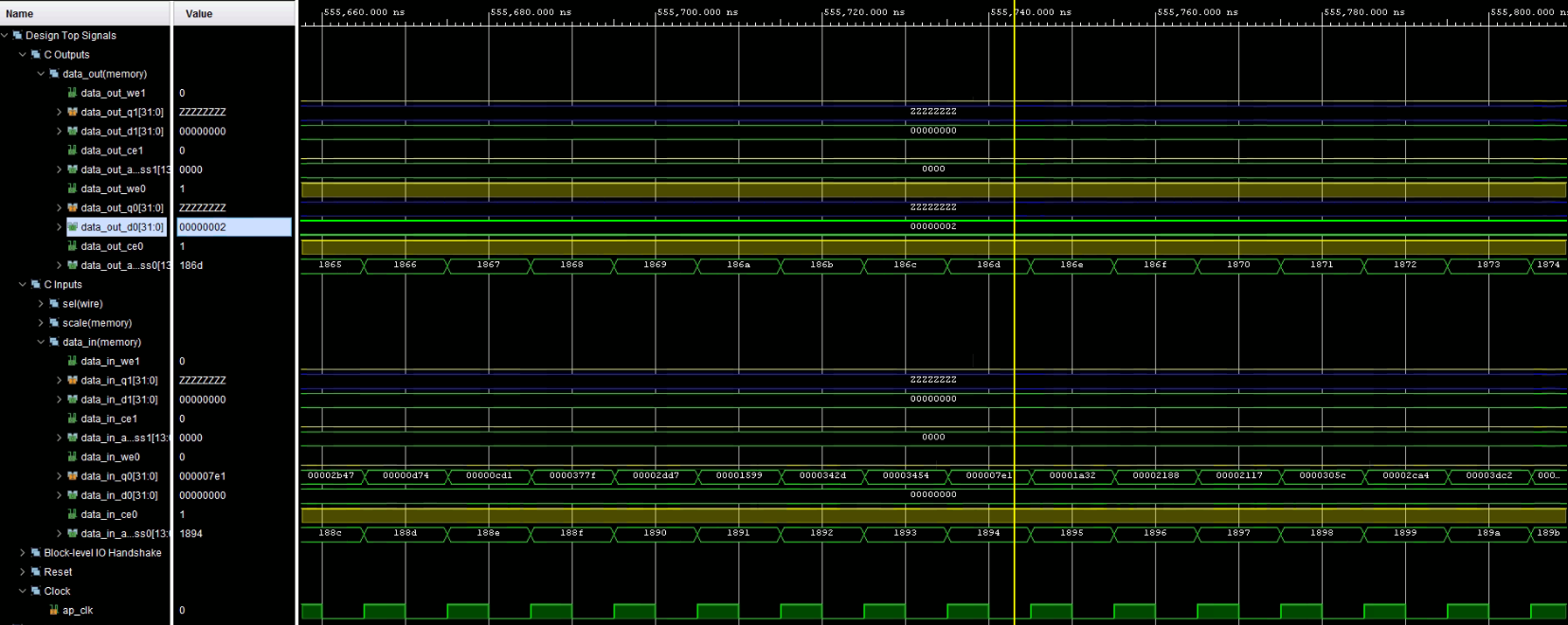


Рисунок 9 - Временная диаграмма для функции foo\_m

На рисунке 10 приведен Dataflow Viewer для случая FIFO the memory buffers.

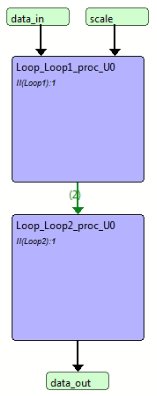


Рисунок 10 - Dataflow viewer FIFO для функции foo\_m

* 1. Dataflow с ping-pong memory buffer

На рисунке 11 представлен результат синтезирования функции **foo\_m** для **ping-pong memory buffers** в виде *Perfomance и Utilization Estimates* соответственно. Время Latency составляет 277909.17 нс.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 11 - 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 12 и 13 приведены Schedule и Resource viewer соответственно.



Рисунок 12 - Schedule viewer для ping-pong memory buffers

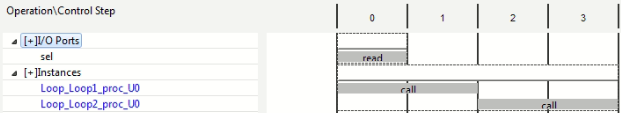


Рисунок 13 - Resource viewer для ping-pong memory buffers

На рисунке 14 приведен *Dataflow Viewer* для решения **ping-pong memory buffer.**

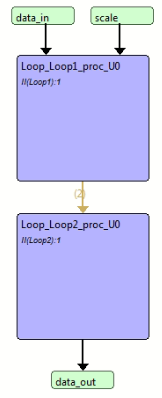


Рисунок 14 - Dataflow viewer ping-pong memory buffers для функции foo\_m

1. Исходный код и результат модифицированного теста

На рисунке 16 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).

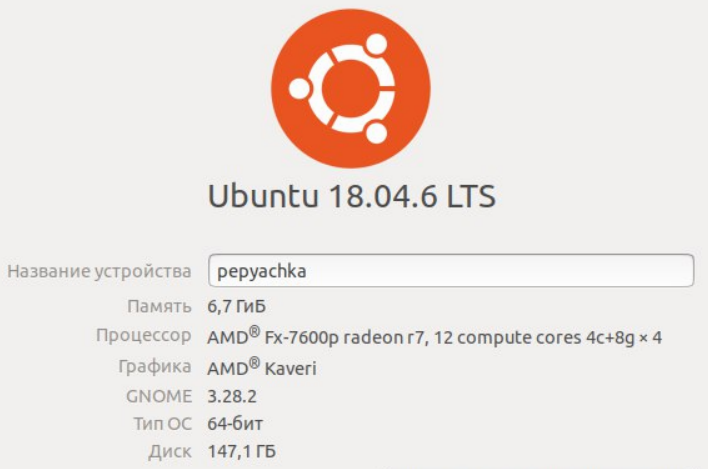


Рисунок 15 - Параметры ПК (Частота = 2.7 Гц)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 16 - Исходный код модифицированного теста для ПК

На рисунке 17 представлены результаты запуска функции на ПК.

Среднее время выполнения функции составило 455278.87 нс, что в разы медленнее, чем решения, полученные при синтезировании функции для ping-pong memory buffer и для FIFO memory buffer.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 17 - 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, медленнее, чем решения после синтезирования в Vivado HLS для Ping-pong memory buffer и для FIFO memory buffer.