Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ Lab4\_Z1

Дисциплина: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Pipelines

Выполнил студент гр. Шеметов С.А.

гр.5140901/21502

Руководитель, доцент Антонов А.П.

«\_\_» \_\_\_\_\_\_ 2023

Санкт-Петербург

2023

Оглавление

[1. Задание 3](#_Toc148840636)

[2. Исходный код функции 3](#_Toc148840637)

[3. Исходный код теста 4](#_Toc148840638)

[3.1. Моделирование 4](#_Toc148840639)

[4. Создание скрипта 6](#_Toc148840640)

[5. Сравнение результатов решений 8](#_Toc148840641)

[5.1. Сравнение временных параметров, производительности и аппаратных затрат 8](#_Toc148840642)

[5.2. Сравнение использованных интерфейсов 9](#_Toc148840643)

[5.3. Результаты планирования (Schedule viewer) 10](#_Toc148840644)

[5.4. C/RTL моделирование 14](#_Toc148840645)

[5.4.1. Отчет по производительности 14](#_Toc148840646)

[5.4.2. Временная диаграмма 16](#_Toc148840647)

[5.5. Анализ результатов 18](#_Toc148840648)

[6. Измерение времени выполнения 19](#_Toc148840649)

[6.1. Измерение времени выполнения на ПК 19](#_Toc148840650)

[6.2. Измерение времени выполнения на аппаратной реализации 23](#_Toc148840651)

[6.3. Сравнение результатов 25](#_Toc148840652)

[7. Вывод 25](#_Toc148840653)

1. Задание

* Выбрать одну из трёх предложенных функций и реализовать на языке С++ (lab4\_z1.cpp). Функция умножает вектор А (N элементов) на вектор В (N элементов) по следующему алгоритму:

A[a1 a2 a3 a4] \* B[b1 b2 b3 b4] =

res[

a1\*b1 + a1\*b2 + a1\*b3 +a1\* b4

a2\*b1 + a2\*b2 + a2\*b3 +a2\* b4

a3\*b1 + a3\*b2 + a3\*b3 +a3\* b4

a4\*b1 + a4\*b2 + a4\*b3 +a4\* b4]

* Создать на языке С++ тест (lab4\_z1\_test.cpp) для проверки работы функции.
* Создать скрипт автоматизирующий процесс создания проекта и работы с ним.
* Создать, провести исследование и сравнительный анализ трех решений.
  + Микросхема: xa7a12tcsg325-1q
  + clock period 10; clock\_uncertainty 1
  + Pipeline off – для каждого цикла (для решения sol1)
  + для цикла L1 задать директиву Pipeline II=1 (для решения sol2)
  + для цикла L1 задать директиву Pipeline off и для внешнего цикла задать Pipeline II=1 и rewind (для решения sol3)
  + Синтез
  + С/RTL cosimulation (с опцией Dump Trace = Port)
* Измерить время выполнения на ПК и на аппаратной реализации.

2. Исходный код функции

Исходный код синтезируемой функции приведен на рисунке 1. Код заголовочного файла указан на рисунке 2.

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

Автоматически созданное описание*

*Рис.1. Исходный код функции* *lab4\_z1*

Изображение выглядит как текст, снимок экрана, Шрифт, Мультимедийное программное обеспечение

Автоматически созданное описание

*Рис.2. Исходный код заголовочного файла lab4\_z1*

Функция принимает два входных вектора и один выходной типа int и размером N = 4, выполняет перемножение входных векторов по алгоритму и записывает результат в выходной.

Из трёх предложенных реализаций самой производительной была выбрана первая, так как в ней происходит меньше обращений к памяти. Значение in\_a[j] получается один раз за итерацию внешнего цикла L2 и записывается в переменную temp\_a, а затем используется во внутреннем цикле L1. Во второй и третей предложенной реализации обращение к ячейке памяти in\_a[j] происходит N раз на каждой итерации внешнего цикла L2.

3. Исходный код теста

Исходный код теста для проверки функции lab4\_z1, исправленный в соответствии с заданием, приведен на рисунке 3. Тест обеспечивает: запуск функции 3 раза; формирование двух входных массивов, заполненных случайными числами; очистку выходного массива между запусками функции; проверку правильности вычисленного результата и формирование признака успешного/неуспешного выполнения для каждого запуска функции.

3.1. Моделирование

Результаты моделирования исходного кода синтезируемой функции приведены на рисунке 4. 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, дисплей, программное обеспечение

Автоматически созданное описание

*Рис.3. Исходный код теста*

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

Автоматически созданное описание*

*Рис.4. Лог результата выполнения моделирования*

Для проверки правильности работы теста в него было внесено изменение в ожидаемый результат - expted\_ar[j] += inA\_ar[j] \* inB\_ar[k] + 1. Результат моделирования приведён на рисунке 5. Он показывает, что тест отрабатывает ошибку корректно.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

*Рис.5. Лог результата при ошибке в тесте*

4. Создание скрипта

Созданный скрипт для автоматизации работы с проектом представлен на рисунке 6.

Данный скрипт создаёт или открывает проект, предварительно очищая его, если он уже существует, командой *open\_project -reset lab4\_z1*. Затем добавляется файл с исходным кодом *add\_files ./source/lab4\_z1.cpp* и устанавливается функция верхнего уровня в иерархии *set\_top lab4\_z1*. Добавляется тестовый файл - *add\_files -tb ./source/lab4\_z1\_test.cpp*. Затем по очереди создаются три решения. Для каждого из них задаётся одинаковая микросхема, период 10 нс и неопределённость 1 нс.

Первое решение *- open\_solution -reset "sol1"*. Для него запускается Си моделирование (*csim\_design -clean*), отключается конвейеризация для всех циклов (*set\_directive\_pipeline -off*) и выполняется синтез (*csynth\_design*), а затем моделирование cosim с опцией Dump Trace = Port (*cosim\_design -trace\_level port*).

Второе решение - *open\_solution -reset "sol2"*. Для него выполняется только синтез и моделирование cosim. Перед началом синтеза для цикла L1 задаётся директива Pipeline II=1 (*set\_directive\_pipeline -II 1 "lab4\_z1/L1"*). Директива указывает желаемый интервал запуска конвейера.

Третье решение - *open\_solution -reset "sol3"*. Для него выполняется только синтез и моделирование cosim. Перед началом синтеза задаются 2 директивы: для цикла L1 директива Pipeline off (*set\_directive\_pipeline -off "lab4\_z1/L1"*), отключающая конвейеризацию, и для цикла L2 директива Pipeline II=1 и rewind (*set\_directive\_pipeline -II 1 -rewind "lab4\_z1/L2"*). Ключевое слово rewind обеспечивает непрерывную конвейерную обработку цикла без пауз между одним выполнением окончания цикла и началом следующего выполнения.

Запуск скрипта выполняется в командной строке Vitis HLS 2021.2 Command Prompt с помощью команды *Vitis\_hls -f lab3\_z3.tcl*.

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

Автоматически созданное описание**

*Рис.6. Скрипт для работы с проектом*

5. Сравнение результатов решений

5.1. Сравнение временных параметров, производительности и аппаратных затрат

В данной работе были реализованы три решения с разными директивами конвейеризации. В первом решении для обоих циклов конвейеризация отключена, во втором для внутреннего цикла задан интервал запуска конвейера, а в третьем – для внутреннего конвейеризация отключена, для внешнего задан интервал и указана опция rewind. Результаты сравнения приведены на рисунке 7.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

*Рис.7. Сравнение трёх решений*

Сравнение данных на рисунке 7 показывает, что Estimated timing у всех решений одинаковое. Однако Latency и II значительно различаются. Для первого значения 77 и 78 тактов, для второго – 20 и 21 такт, для третьего 11–12 и 8 тактов.

Аппаратные затраты также различаются по числу FF триггеров и LUT таблиц перекодировок. Первое решение требует меньше всего аппаратных затрат (150 триггеров и 160 таблиц). Второе имеет средние показатели (342 триггера и 346 таблиц). Третье решение имеет наибольшие аппаратные затраты (590 триггеров и 345 таблиц).

5.2. Сравнение использованных интерфейсов

Сравнение интерфейсов представлено на рисунках 8–9. Для первых двух решений интерфейсы полностью совпадают. Входные массивы in\_a и in\_b имеют по три стандартных порта - выходной address0 (адрес элемента массива), выходной ce0 (управляющий сигнал, разрешающий чтение) и входной q0 (входные данные). Выходной массив res имеет 4 выходных порта: address0 (адрес элемента массива), ce0 (управляющий сигнал, разрешающий чтение), we0 (сигнал для записи) и d0 (данные).

Третье решение имеет на три порта больше. Для массива in\_b добавляются ещё три порта: address1, ce1, q1. Это происходит из-за применения опции rewind.

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

*Рис.8. Интерфейсы для sol1 и sol2*

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

*Рис.9. Интерфейсы для sol3*

5.3. Результаты планирования (Schedule viewer)

Результаты планирования для первого решения представлены на рисунках 10–15.

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

Автоматически созданное описание

*Рис.10. Результаты планирования для sol1. Операция in\_a\_addr(getelementptr)*

Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение, Графическое программное обеспечение, 3D-моделирование

Автоматически созданное описание

*Рис.11. Результаты планирования для sol1. Операция temp\_a(read)*

Изображение выглядит как снимок экрана, текст, 3D-моделирование, Графическое программное обеспечение

Автоматически созданное описание

*Рис.12. Результаты планирования для sol1. Операция in\_b\_addr(getelementptr)*

Изображение выглядит как снимок экрана, текст, Графическое программное обеспечение, 3D-моделирование

Автоматически созданное описание

*Рис.13. Результаты планирования для sol1. Операция mul\_ln10(\*)*

Изображение выглядит как снимок экрана, Графическое программное обеспечение, Мультимедийное программное обеспечение, 3D-моделирование

Автоматически созданное описание

*Рис.14. Результаты планирования для sol1. Операция product\_1(+)*

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

Автоматически созданное описание*

*Рис.15. Результаты планирования для sol1. Операция res\_addr\_write\_ln12(write)*

По результатам первого решения видно, что в данном случае конвейеризация была отключена для обоих циклов. Внешний цикл L2 начинается спустя один такт. Одна итерация внешнего цикла занимает 19 тактов.

Цикл L2. На первых двух тактах получается значение *in\_a[j]* (рисунок 10) и записывается в *temp\_a* (рисунок 11). Затем 16 тактов выполняется внутренний цикл L1. На последнем такте L2 происходит запись в *res[j]* (рисунок 15).

Цикл L1. Одна итерация занимает 4 такта. На первых двух выполняется чтение *in\_b[i]* (рисунок 12). На третьем – перемножение элементов (рисунок 13). На четвёртом – суммирование результата умножения со значением переменной product (рисунок 14).

Значение Latency и Initiation Interval (II) соответствуют полученным на рисунке 7.

Результаты планирования для второго решения представлены на рисунках 16–17.

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

Автоматически созданное описание

*Рис.16. Результаты планирования для sol2*

Изображение выглядит как текст, снимок экрана, Графическое программное обеспечение

Автоматически созданное описание

*Рис.17. Результаты планирования для sol2. Цикл L2\_L1*

По результатам видно, что была выполнена конвейеризация внутреннего цикла L1. Теперь одна итерация требует 4 такта и между итерациями 1 такт. Таким образом, для выполнения всего цикла требуется 20 тактов, что соответствует рисунку 7.

Результаты планирования для третьего решения представлены на рисунке 18.

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

Автоматически созданное описание

*Рис.18. Результаты планирования для sol3*

По результатам видно, что была выполнена конвейеризация внешнего цикла L2 и применена опция rewind.

5.4. C/RTL моделирование

5.4.1. Отчет по производительности

Отчеты по производительности для трёх решений, полученный при моделировании cosim, приведен на рисунках 19–21. Из отчётов следует, что полученные Latency и II для первого и третьего решений совпадают со значениями при синтезе, показанными на рисунке 7. Для второго решения Loop Latency и II меньше на 1 такт.

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

Автоматически созданное описание

*Рис.19. Отчет по производительности C/RTL для sol1*

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

Автоматически созданное описание

*Рис.20. Отчет по производительности C/RTL для sol2*

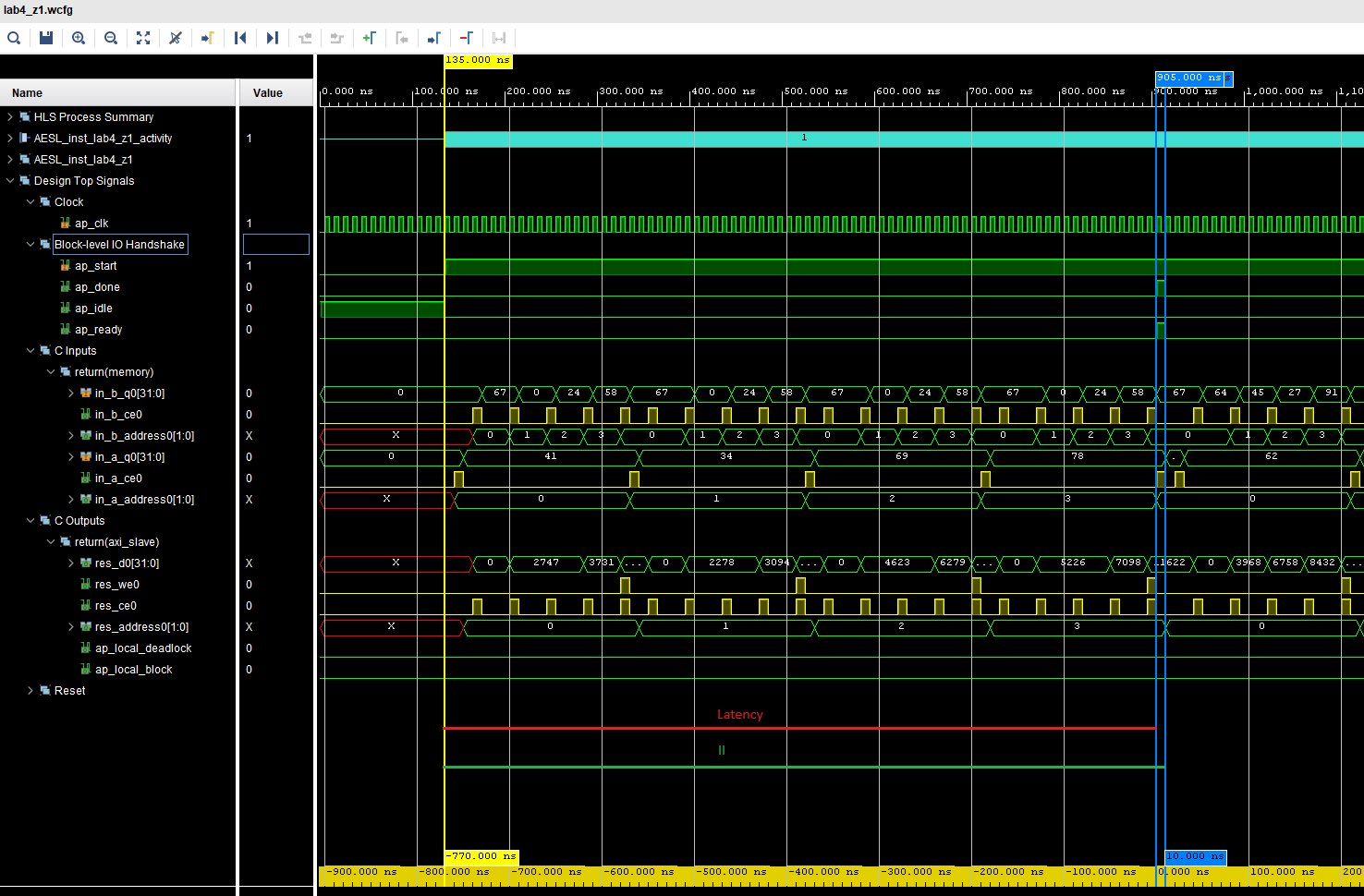
Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

*Рис.21. Отчет по производительности C/RTL для sol3*

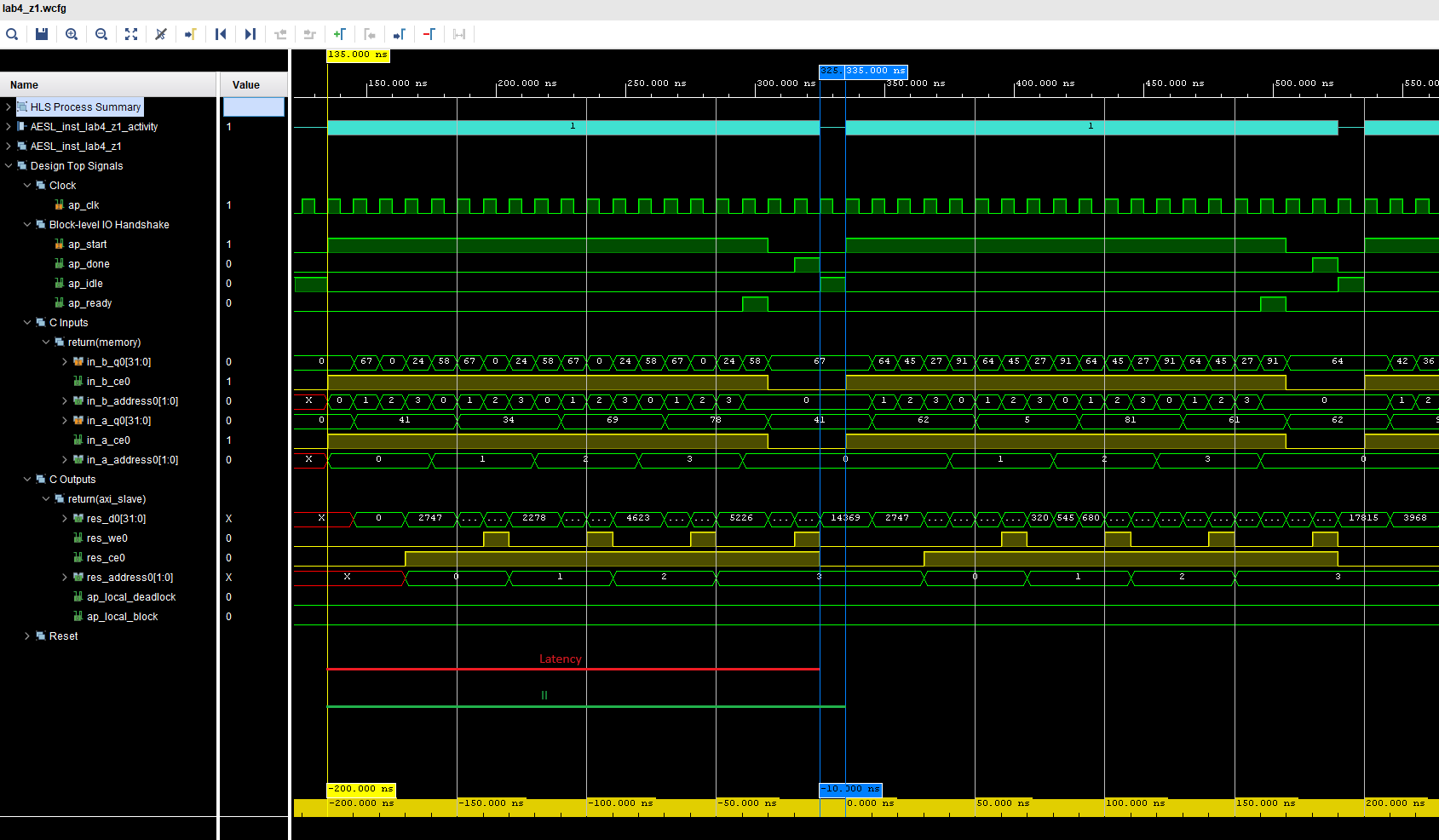
5.4.2. Временная диаграмма

На рисунке 22 приведена временная диаграмма совместного C/RTL моделирования для первого решения. Она отображает первый запуск функции. Поступает сигнал in\_a\_ce0 разрешающий чтение данных по адресу, записанному на шине in\_a\_address0[1:0]. После спада ce0 на шину in\_a\_q0[31:0] выводится прочитанное число. Затем аналогично считывается число in\_b. Далее происходит умножение и суммирование. Результат записывается по res\_address0[1:0] после одновременной подачи сигналов res\_ce0 и res\_we0. Latency занимает 77 тактов, а II – 78 тактов, что соответствует рисунку 19.



*Рис.22. Временная диаграмма C/RTL для sol1*

На рисунке 23 приведена временная диаграмма совместного C/RTL моделирования для второго решения. Она отображает два запуска функции. Между запусками проходит один такт, как указано в директиве. Latency занимает 19 тактов, а II – 20 тактов, что соответствует рисунку 20.



*Рис.23. Временная диаграмма C/RTL для sol2*

На рисунке 24 приведена временная диаграмма совместного C/RTL моделирования для третьего решения. Она отображает три запуска функции.

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

Автоматически созданное описание

*Рис.24. Временная диаграмма C/RTL для sol3*

5.5. Анализ результатов

На основе результатов сравнения решений была составлена электронная таблица и построен график, в котором для всех решений отображены: Latency и использованные ресурсы. Результаты представлены на рисунках 25–26.

Изображение выглядит как текст, число, Шрифт, снимок экрана

Автоматически созданное описание

*Рис.25. Электронная таблица для трёх решений*

Изображение выглядит как снимок экрана, линия, диаграмма, График

Автоматически созданное описание

*Рис.26. График для трёх решений*

С точки зрения производительности оптимальным решением является третье, так как оно затрачивает меньше времени. Это обусловлено использованием конвейеризации и rewind, которые позволяют быстрее выполнять арифметические операции.

С точки зрения аппаратных затрат первое решение является оптимальным, так как оно имеет меньшее число триггеров и таблиц перекодировок.

6. Измерение времени выполнения

6.1. Измерение времени выполнения на ПК

Измерение времени выполнения синтезируемой функции было выполнено на ПК со следующими характеристиками:

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип процессора** | **Базовая частота работы** | **Максимальная частота работы** | **ОЗУ** |
| 12th Gen Intel(R) Core(TM) i5-12450H | 2 ГГц | 4,4 ГГц | 16 Гб |

На базе теста lab4\_z1\_test.cpp был создан отдельный, модернизированный, тест lab4\_z1\_testSW.cpp, который содержит операторы измерения времени выполнения и имеет большее число запусков равное 32. Код теста представлен на рисунке 27.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

*Рис.27. Код теста* *lab4\_z1\_testSW.cpp*

Для компиляции теста был установлен компилятор GCC версии 13.2.0. После этого была выполнена сама компиляции и запуск теста для трёх значений N – 4096. 8192, 16384 (рисунок 28). Результаты запуска представлены на рисунках 29–31.

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

*Рис.28. Выполнение компиляции теста*

Изображение выглядит как текст, снимок экрана, монохромный, шаблон

Автоматически созданное описание

*Рис.29. Результаты теста на ПК для N=4096*

Изображение выглядит как текст, снимок экрана, черно-белый, шаблон

Автоматически созданное описание

*Рис.30. Результаты теста на ПК для N=8192*

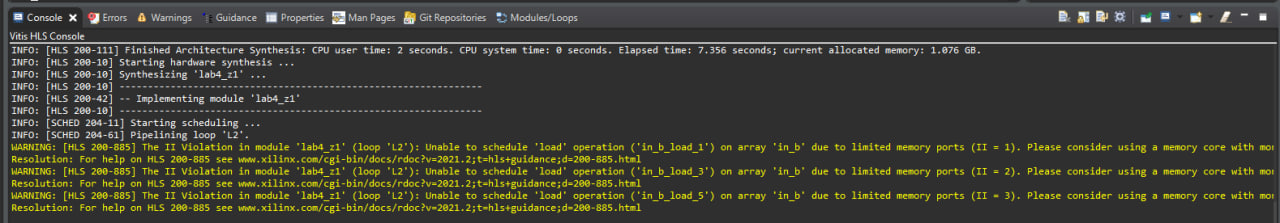
Изображение выглядит как текст, снимок экрана, черно-белый, шаблон

Автоматически созданное описание

*Рис.31. Результаты теста на ПК для N=16384*

6.2. Измерение времени выполнения на аппаратной реализации

Для измерения времени необходимо выбрать решение, лучшее с точки зрения производительности. Им является sol3. Однако при больших значениях синтез в Vitis происходил бесконечно, лишь выдавая Warning (рисунок 32). Поэтому измерение времени было выполнено на втором по производительности решении – sol2.



*Рис.32. Синтез sol3 для больших значений*

Синтез решения sol2 был осуществлён для трёх значений N – 4096. 8192, 16384. Результаты запуска представлены на рисунках 33–35.

Время выполнения = II \* Estimated period для каждого случая:

* для 4096: 16777221 такта \* 8.470 нс = 142103061.87 нс
* для 8192: 67108869 такта \* 8.470 нс = 568412120.43 нс
* для 16384: 268435461 такта \* 8.470 нс = 2273648354.67 нс

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

Автоматически созданное описание

*Рис.33. Результаты теста на аппаратной реализации для N=4096*

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

Автоматически созданное описание

*Рис.34. Результаты теста на аппаратной реализации для N=8192*

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

Автоматически созданное описание

*Рис.35. Результаты теста на аппаратной реализации для N=16384*

6.3. Сравнение результатов

На основе результатов, полученных при измерениях времени выполнения на ПК и аппаратной реализации, была составлена электронная таблица и построен график, в котором отражено соотношение размера массивов данных задачи (N) ко времени выполнения. Результаты представлены на рисунках 36–37.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

*Рис.36. Электронная таблица для ПК и аппаратной реализации*

Изображение выглядит как снимок экрана, линия, График, текст

Автоматически созданное описание

*Рис.37. График для ПК и аппаратной реализации*

На графике можно видеть, что результаты аппаратной реализации растут экспоненциально, а на ПК более линейно. При этом на аппаратной реализации время выполнения в 5 раз больше. Более медленное выполнения аппаратной реализации связано с тем, что она имеет меньшую частоту по сравнению с ПК.

7. Вывод

В результате выполнения данной лабораторной работы были получены навыки работы с конвейеризацией. Для этого была выбрана наиболее производительная из трёх предложенных реализаций программы, перемножающей пару векторов по определённому алгоритму. Это первая реализация, так как она имеет меньшее число обращений к in\_a[j] в цикле.

Для создания проекта и работы с ним был создан TCL скрипт, содержащий три решения. Каждое решение имело разные настройки конвейеризации: Pipeline off для каждого цикла в sol1; Pipeline II=1 для L1 в sol2; Pipeline off для L1 и Pipeline II=1 и rewind для L2 в sol3. Оптимальным с точки зрения производительности оказалось sol3 за счёт использования конвейеризации и параметра rewind.

Также было выполнено измерение времени выполнения программы на ПК и аппаратной реализации. В качестве аппаратной реализации было выбрано решение sol2, так как при sol3 программа Vitis бесконечно выполняло синтез. Результаты показали, что аппаратная реализация значительно проигрывает в производительности.