Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ Lab5\_Z1

Дисциплина: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Optimizing performance

Выполнил студент гр. Шеметов С.А.

гр.5140901/21502

Руководитель, доцент Антонов А.П.

«\_\_» \_\_\_\_\_\_ 2023

Санкт-Петербург

2023

Оглавление

[1. Задание 4](#_Toc150297551)

[2. Исходный код функции 5](#_Toc150297552)

[3. Исходный код теста 5](#_Toc150297553)

[3.1. Моделирование 5](#_Toc150297554)

[4. Создание скрипта 7](#_Toc150297555)

[5. Исследование решений 9](#_Toc150297556)

[5.1. Решение 1 9](#_Toc150297557)

[5.1.1. Синтез 9](#_Toc150297558)

[5.1.2. C/RTL моделирование 10](#_Toc150297559)

[5.2. Решение 2 11](#_Toc150297560)

[5.2.1. Синтез 11](#_Toc150297561)

[5.2.2. C/RTL моделирование 12](#_Toc150297562)

[5.3. Решение 3 13](#_Toc150297563)

[5.3.1. Синтез 13](#_Toc150297564)

[5.3.2. C/RTL моделирование 15](#_Toc150297565)

[5.4. Решение 4 16](#_Toc150297566)

[5.4.1. Синтез 16](#_Toc150297567)

[5.4.2. C/RTL моделирование 18](#_Toc150297568)

[5.5. Решение 5 19](#_Toc150297569)

[5.5.1. Синтез 19](#_Toc150297570)

[5.5.2. C/RTL моделирование 21](#_Toc150297571)

[5.6. Решение 6 22](#_Toc150297572)

[5.6.1. Синтез 22](#_Toc150297573)

[5.6.2. C/RTL моделирование 24](#_Toc150297574)

[5.7. Решение 7 25](#_Toc150297575)

[5.7.1. Синтез 25](#_Toc150297576)

[5.7.2. C/RTL моделирование 28](#_Toc150297577)

[6. Сравнение результатов решений 29](#_Toc150297578)

[6.1. Сравнение временных параметров, производительности и аппаратных затрат 29](#_Toc150297579)

[6.2. Анализ результатов 29](#_Toc150297580)

[7. Измерение времени выполнения 30](#_Toc150297581)

[7.1. Измерение времени выполнения на ПК 30](#_Toc150297582)

[7.2. Измерение времени выполнения на аппаратной реализации 34](#_Toc150297583)

[7.3. Сравнение результатов 35](#_Toc150297584)

[8. Вывод 36](#_Toc150297585)

1. Задание

* Создать на языке С++ функцию (lab5\_z1.cpp), суммирующую элементы с индексами i, i+N/4, i+N/2, i+3\*N/4, где N – число элементов входного массива равное 512.
* Создать на языке С++ тест (lab5\_z1\_test.cpp) для проверки работы функции.
* Создать, провести исследование и сравнительный анализ семи решений.
  + Микросхема: xa7a12tcsg325-1q;
  + clock period 20; clock\_uncertainty не задавать;
  + Solution1:
    - Установить RAM\_1P для входного массива.
  + Solution2:
    - Установить RAM\_2P для входного массива;
    - Сравнить с Solution1.
  + Solution3:
    - Установить RAM\_1P;
    - Установить array\_partition block, factor =4 для входного массива;
    - Сравнить с Solution1, 2.
  + Solution4:
    - Установить RAM\_1P;
    - Установить array\_reshape block, factor =4 для входного массива;
    - Сравнить с Solution1, 2, 3.
  + Solution5:
    - Установить RAM\_2P;
    - Установить array\_partition block, factor =2 для входного массива;
    - Сравнить с Solution3.
  + Solution6:
    - Установить RAM\_2P;
    - Установить array\_reshape block, factor =2 для входного массива;
    - Сравнить с Solution2, 4.
  + Solution7:
    - Установить RAM\_2P;
    - Установить array\_partition block, factor =4 для входного массива;
    - Установить unroll, factor=2 для цикла;
    - Сравнить с Solution4.
  + Выключить конвейеризацию для цикла.
  + Синтез.
  + С/RTL cosimulation.
* 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* *lab5\_z1*

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

Автоматически созданное описание

*Рис.2. Исходный код заголовочного файла lab5\_z1*

Функция принимает два вектора типа short: входной размером N и выходной размером N/4. Затем с помощью цикла выполняется суммирование четырёх элементов входного массива с интервалом N/4 и полученный результат записывается в выходной массив.

3. Исходный код теста

Исходный код теста для проверки функции lab5\_z1, исправленный в соответствии с заданием, приведен на рисунке 3. Тест обеспечивает: запуск функции 3 раза; формирование входного массива, заполненного случайными числами; очистку входного и выходного массивов 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, программное обеспечение, дисплей

Автоматически созданное описание

*Рис.3. Исходный код теста*

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

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] = in\_ar[j] + in\_ar[j + N/4] + in\_ar[j + N/2] + in\_ar[j + 3\*N/4] + 1. Результат моделирования приведён на рисунке 5. Он показывает, что тест отрабатывает ошибку корректно.

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

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 lab5\_z1*. Затем добавляется файл с исходным кодом *add\_files ./source/lab5\_z1.cpp* и устанавливается функция верхнего уровня в иерархии *set\_top lab5\_z1*. Добавляется тестовый файл - *add\_files -tb ./source/lab5\_z1\_test.cpp*. Затем по очереди создаются 14 решений. Для каждого из них задаётся одинаковая микросхема и период 20 нс.

Для решений sol1-sol7 используются разные типы ОЗУ (однопортовое и двухпортовое), а также применяются директивы ARRAY\_PARTITION и ARRAY\_RESHAPE для увеличения пропускной способности данных. Обе директивы разделяют большие массивы на несколько меньших массивов. Однако Partitioning увеличивает количество портов памяти, таким образом увеличивая количество операций ввода-вывода, с которыми приходится иметь дело. А Reshaping увеличивает ширину слова данных и не увеличивает количество портов памяти. Кроме того, конвейеризация цикла да этих решений отключена.

Решения sol1\_1-sol7\_1 аналогичны предыдущим, но для них конвейеризация включена.

Запуск скрипта выполняется в командной строке Vitis HLS 2021.2 Command Prompt с помощью команды *Vitis\_hls -f lab5\_z1.tcl*.

**Изображение выглядит как текст, снимок экрана, документ

Автоматически созданное описание**

**Изображение выглядит как снимок экрана, текст

Автоматически созданное описание**

*Рис.6. Скрипт для работы с проектом*

5. Исследование решений

5.1. Решение 1

5.1.1. Синтез

В решении 1 для входного массива был установлен тип ОЗУ RAM\_1P – однопортовая память. Временные параметры, производительность и аппаратные затраты решения представлены на рисунке 7.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

*Рис.7. Результаты синтеза для sol1*

Estimated timing для решения составило 12.697 нс. Latency и II – 641 такт и 642 такта соответственно. Аппаратные затраты - 86 триггеров и 149 таблиц.

Результаты планирования для sol1 представлены на рисунке 8.

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

Автоматически созданное описание

*Рис.8. Результаты планирования для sol1*

Как можно видеть на рисунке 8, одна итерация длится 5 тактов. При этом на каждом такте происходит чтение нового суммирующегося элемента.

5.1.2. C/RTL моделирование

Временная диаграмма для sol1 представлена на рисунке 9.

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

Автоматически созданное описание

*Рис.9. Временная диаграмма C/RTL для sol1*

Так как чтение и запись в однопортовую память не могут выполняться одновременно, то в момент записи на временной диаграмме появляются «пустые» такты. Чтобы это исправить, можно выполнить конвейеризацию цикла. Временная диаграмма для sol1\_1 с конвейеризацией приведена на рисунке 10.

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

Автоматически созданное описание

*Рис.10. Временная диаграмма C/RTL для sol1\_1*

5.2. Решение 2

5.2.1. Синтез

Решения sol1 и sol2 различаются типом памяти. В первом используется RAM\_1P – однопортовая память, позволяющая за один такт считать один элемент массива. Во втором RAM\_2P – двухпортовая память, позволяющая и за один такт считать два элемента массива. Результаты сравнения двух решений приведены на рисунке 11.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

*Рис.11. Сравнение решений sol1 и sol2*

Сравнение данных на рисунке 11 показывает, что Estimated timing у обоих решений одинаковое. Однако Latency и II значительно различаются. Для первого значения - 641 и 642 такта, для второго – 385 и 386 тактов. Это связано с тем, что во втором решении за такт считывается большее количество данных.

Аппаратные затраты также различаются по числу FF триггеров и LUT таблиц перекодировок. Первое решение требует больше аппаратных затрат (86 триггеров и 149 таблиц). Второе требует немного меньше (68 триггеров и 146 таблиц).

Результаты планирования для sol2 представлены на рисунке 12.

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

Автоматически созданное описание

*Рис.12. Результаты планирования для sol2*

При сравнении результатов планирования для sol1 и sol2 (рисунки 8 и 12) видно, что благодаря использованию двухпортовой памяти для входного массива итерация цикла выполняется за меньшее число тактов, так как за один такт считывается два значения.

5.2.2. C/RTL моделирование

Временная диаграмма для sol2 представлена на рисунке 13.

Изображение выглядит как снимок экрана, текст, электроника, дисплей

Автоматически созданное описание

*Рис.13. Временная диаграмма C/RTL для sol2*

Двухпортовая память позволяет считывать два значения массива за такт и записывать значение в одну и туже память. Однако, как и в решении sol1 «пустой» такт отводится для записи. Чтобы его убрать, нужно выполнить конвейеризацию. Временная диаграмма для sol2\_1 с конвейеризацией приведена на рисунке 14.

Изображение выглядит как текст, снимок экрана, дисплей, электроника

Автоматически созданное описание

*Рис.14. Временная диаграмма C/RTL для sol2\_1*

5.3. Решение 3

5.3.1. Синтез

Как и в первом решении в третьем используется однопортовая память. Однако, в отличии от первых двух решений в третьем также используется директива array\_partition. Она разделяет массив на factor частей, что позволяет читать значение из всех четырёх одновременно. Для решения sol3 указаны параметры: block – разбиение массива на равные блоки и factor=4 – количество создаваемых массивов. Результаты сравнения трёх решений приведены на рисунке 15.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

*Рис.15. Сравнение решений sol1, sol2 и sol3*

Сравнение данных на рисунке 15 показывает, что Estimated timing у всех решений одинаковое. Однако третье решение имеет наименьшие значения Latency и II – 257 и 258 тактов. Аппаратные затраты также наименьшие у третьего решения - 19 триггеров и 107 таблиц.

Результаты планирования для sol3 представлены на рисунке 16.

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

Автоматически созданное описание

*Рис.16. Результаты планирования для sol3*

При сравнении результатов планирования для трёх решений (рисунки 8, 12 и 16) видно, что благодаря использованию разделения массива на 4 части, число тактов одной итерации цикла сократилось до двух. Это произошло благодаря тому, что теперь все четыре элемента считываются на одном такте цикла.

5.3.2. C/RTL моделирование

Временная диаграмма для sol3 представлена на рисунке 17.

Изображение выглядит как электроника, снимок экрана, текст, дисплей

Автоматически созданное описание

*Рис.17. Временная диаграмма C/RTL для sol3*

Для того, чтобы реализовать считывание данных на каждом такте и формирование результата на каждом такте, необходимо выполнить конвейеризацию. Временная диаграмма для sol3\_1 с конвейеризацией приведена на рисунке 18.

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

Автоматически созданное описание

*Рис.18. Временная диаграмма C/RTL для sol3\_1*

5.4. Решение 4

5.4.1. Синтез

В четвертом решении используется однопортовая память и директива array\_reshape с параметрами block и factor=4. Она позволяет разбить массив на factor частей, при этом блок останется один, но размер слова увеличится в N раз. Таким образом, элементов становится меньше, а длина слова увеличивается.

Результаты сравнения четырёх решений приведены на рисунке 19.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

*Рис.19. Сравнение решений sol1, sol2, sol3 и sol4*

Сравнение данных на рисунке 19 показывает, что Estimated timing у четвёртого решения равно 14.455 нс, что отличается от значений остальных решений. Значения Latency и II совпадают со вторым решением.

Аппаратные затраты четвёртого решения значительно превосходят все остальные. Если число FF триггеров соизмеримо с остальными – 42 триггера, то значение LUT больше в 18 раз – 1827 таблиц.

Результаты планирования для sol4 представлены на рисунке 20.

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

Автоматически созданное описание

*Рис.20. Результаты планирования для sol4*

При сравнении результатов планирования для всех решений (рисунки 8, 12, 16 и 20) видно, что аналогично решению 2, одна итерация цикла занимает 3 такта. Однако, во втором решении выполняется четыре операции чтения, по две на такт. В четвёртом решении – одна операция чтения на первом такте цикла.

5.4.2. C/RTL моделирование

Временная диаграмма для sol4 представлена на рисунке 21.

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

Автоматически созданное описание

*Рис.21. Временная диаграмма C/RTL для sol4*

Для того, чтобы реализовать считывание данных на каждом такте и формирование результата на каждом такте, необходимо выполнить конвейеризацию. Временная диаграмма для sol4\_1 с конвейеризацией приведена на рисунке 22.

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

Автоматически созданное описание

*Рис.22. Временная диаграмма C/RTL для sol4\_1*

5.5. Решение 5

5.5.1. Синтез

Как и в третьем решении в пятом используется директива array\_partition, только с параметрами block и factor=2. Также в пятом решении вместо однопротовой используется двухпортовая память. Результаты сравнения решений sol3 и sol5 приведены на рисунке 23.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

*Рис.23. Сравнение решений sol3 и sol5*

Сравнение данных на рисунке 23 показывает, что Estimated timing, Latency и II у двух решений совпадают. Это связано с тем, что в пятом решении массив разбивается на 4 части, как и в третьем, благодаря двухпортовой памяти и array\_partition.

Аппаратные затраты незначительно отличаются. Число FF триггеров для обоих решений одинаково и равно 19. А вот число LUT для sol3 – 107 таблиц, а для sol5 – 116.

Результаты планирования для sol5 представлены на рисунке 24.

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

Автоматически созданное описание

*Рис.24. Результаты планирования для sol5*

При сравнении результатов планирования для двух решений (рисунки 16 и 24) видно, число тактов на итерацию цикла для обоих решений совпадает и равно двум тактам. Также в обоих решениях выполняется четыре операции чтения на первом такте цикла.

5.5.2. C/RTL моделирование

Временная диаграмма для sol5 представлена на рисунке 25.

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

Автоматически созданное описание

*Рис.25. Временная диаграмма C/RTL для sol5*

Временные диаграммы для решений 3 и 5 (рисунки 17 и 25) различаются порядком разбиения данных на 4 части.

Для того, чтобы реализовать считывание данных на каждом такте и формирование результата на каждом такте, необходимо выполнить конвейеризацию. Временная диаграмма для sol5\_1 с конвейеризацией приведена на рисунке 26.

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

Автоматически созданное описание

*Рис.26. Временная диаграмма C/RTL для sol5\_1*

5.6. Решение 6

5.6.1. Синтез

Как и в четвёртом решении в шестом используется директива array\_reshape, только с параметрами block и factor=2. Также в шестом решении, как и во втором, вместо однопротовой используется двухпортовая память. Результаты сравнения решений sol2, sol4 и sol6 приведены на рисунке 27.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

*Рис.27. Сравнение решений sol2, sol4 и sol6*

Сравнение данных на рисунке 27 показывает, что Estimated timing у всех трёх различаются: 12.697 нс, 14.455 нс и 14.176 нс соответственно. Latency и II совпадают у трёх решений совпадают.

Аппаратные затраты также различаются по числу FF триггеров и LUT таблиц перекодировок. Второе решение требует меньше всего аппаратных затрат (68 триггеров и 146 таблиц). Шестое имеет средние показатели (40 триггеров и 463 таблиц). Четвёртое решение имеет наибольшие аппаратные затраты (42 триггера и 1827 таблиц).

Результаты планирования для sol6 представлены на рисунке 28.

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

Автоматически созданное описание

*Рис.28. Результаты планирования для sol6*

При сравнении результатов планирования для трёх решений (рисунки 12, 20 и 28) видно, число тактов на итерацию цикла для трёх решений совпадает и равно трём тактам. Однако количество операций чтений различается. В sol2 их четыре, и они происходят на первом и втором тактах цикла. В sol4 – одна операция чтения на первом такте цикла. В sol6 – две операции чтения на первом такте цикла.

5.6.2. C/RTL моделирование

Временная диаграмма для sol6 представлена на рисунке 29.

Изображение выглядит как снимок экрана, текст, дисплей, электроника

Автоматически созданное описание

*Рис.29. Временная диаграмма C/RTL для sol6*

Временные диаграммы для решений 2, 4 и 6 (рисунки 13, 21 и 29) различаются порядком разбиения данных. Решение sol2 – имеет два порта размерностью 16 бит. Решение sol4 – имеет один порт размерностью 64 бита. Решение sol6 – имеет два порта размерностью 32 бита.

Для того, чтобы реализовать считывание данных на каждом такте и формирование результата на каждом такте, необходимо выполнить конвейеризацию. Временная диаграмма для sol6\_1 с конвейеризацией приведена на рисунке 30.

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

Автоматически созданное описание

*Рис.30. Временная диаграмма C/RTL для sol6\_1*

При увеличении block до 4 производительность не изменится, так как длина слова увеличится, но длина массива уменьшится.

5.7. Решение 7

5.7.1. Синтез

В седьмом решении используется двухпортовая память и директива array\_partition с параметрами block и factor=4. Однако в отличии от остальных в данном решении добавлена директива unroll, которая позволяет полностью или частично выполнить развёртывание цикла. Результаты сравнения решений sol4 и sol7 приведены на рисунке 31.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

*Рис.31. Сравнение решений sol4 и sol7*

Сравнение данных на рисунке 31 показывает, что Estimated timing у двух решений различаются: 14.455 нс и 12.697 нс соответственно. Latency и II также различаются. Для sol4 – 385 и 386 тактов и для sol7 – 129 и 130 тактов.

Аппаратные затраты также различаются по числу FF триггеров и LUT таблиц перекодировок. Четвёртое решение требует значительно больше аппаратных затрат (42 триггеров и 1827 таблиц). Седьмое наименьшие аппаратные затраты (35 триггеров и 217 таблиц).

Результаты планирования для sol7 представлены на рисунке 32.

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

Автоматически созданное описание

*Рис.32. Результаты планирования для sol7*

При сравнении результатов планирования для двух решений (рисунки 20 и 32) видно, седьмое решение имеет меньше число тактов на итерацию цикла (sol4 – 3 и sol7 – 2). Количество операций чтения также различается. В sol4 - одна операция на первом такте цикла. В sol7 – восемь операций чтения на первом такте цикла.

5.7.2. C/RTL моделирование

Временная диаграмма для sol7 представлена на рисунке 33.

Изображение выглядит как электроника, текст, снимок экрана, дисплей

Автоматически созданное описание

*Рис.33. Временная диаграмма C/RTL для sol7*

Временные диаграммы для решений 4 и 7 (рисунки 21 и 33) различаются порядком разбиения данных. Решение sol4 – имеет один порт размерностью 64 бита. Решение sol7 – имеет восемь портов размерностью 16 бит.

Для того, чтобы реализовать считывание данных на каждом такте и формирование результата на каждом такте, необходимо выполнить конвейеризацию. Временная диаграмма для sol7\_1 с конвейеризацией приведена на рисунке 34.

Изображение выглядит как электроника, снимок экрана, текст, дисплей

Автоматически созданное описание

*Рис.34. Временная диаграмма C/RTL для sol7\_1*

6. Сравнение результатов решений

6.1. Сравнение временных параметров, производительности и аппаратных затрат

В данной работе были реализованы 14 решений с разными типами ОЗУ и директивами оптимизации пропускной способности. Результаты сравнения приведены на рисунке 35.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

*Рис.35. Сравнение всех решений*

Сравнение данных на рисунке 35 показывает, что оптимальным с точки зрения производительности и аппаратных затрат является решение sol7\_1, так как оно имеет наименьшие значения Estimated timing, Latency и II и небольшие аппаратные затраты.

6.2. Анализ результатов

На основе результатов сравнения решений была составлена электронная таблица и построен график, в котором для всех решений отображены: Latency, II и использованные ресурсы. Результаты представлены на рисунках 36–37.

Изображение выглядит как текст, число, линия, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

*Рис.36. Электронная таблица для всех решений*

Изображение выглядит как График, диаграмма, линия, Шрифт

Автоматически созданное описание

*Рис.37. График для всех решений*

С точки зрения производительности оптимальным решением является sol7\_1, так как оно затрачивает меньше времени. Это обусловлено использованием двухпортовой памяти, array\_partition, развертывания цикла и конвейеризации.

С точки зрения аппаратных затрат решение sol3 является оптимальным, так как оно имеет меньшее число триггеров и таблиц перекодировок.

7. Измерение времени выполнения

7.1. 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 lab5\_z1\_test.cpp был создан отдельный, модернизированный, тест lab5\_z1\_testSW.cpp, который содержит операторы измерения времени выполнения и имеет большее число запусков равное 32. Код теста представлен на рисунке 38.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

*Рис.38. Код теста* *lab5\_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 39). Результаты запуска представлены на рисунках 40–42.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

*Рис.39. Выполнение компиляции теста*

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

Автоматически созданное описание

*Рис.40. Результаты теста на ПК для N=4096*

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

Автоматически созданное описание

*Рис.41. Результаты теста на ПК для N=8192*

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

Автоматически созданное описание

*Рис.42. Результаты теста на ПК для N=16384*

7.2. Измерение времени выполнения на аппаратной реализации

Для измерения времени необходимо выбрать решение, лучшее с точки зрения производительности. Им является sol7\_1.

Синтез решения sol7\_1 был осуществлён для трёх значений N – 4096. 8192, 16384. Результаты запуска представлены на рисунках 43–45.

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

* для 4096: 515 такта \* 12.697 нс = 6538.955 нс
* для 8192: 1027 такта \* 12.697 нс = 13039.819 нс
* для 2051: 268435461 такта \* 12.697 нс = 26041.547 нс

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

Автоматически созданное описание

*Рис.43. Результаты теста на аппаратной реализации для N=4096*

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

Автоматически созданное описание

*Рис.44. Результаты теста на аппаратной реализации для N=8192*

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

Автоматически созданное описание

*Рис.45. Результаты теста на аппаратной реализации для N=16384*

7.3. 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 46–47.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

*Рис.46. Электронная таблица для ПК и аппаратной реализации*

Изображение выглядит как снимок экрана, линия, График, текст

Автоматически созданное описание

*Рис.47. График для ПК и аппаратной реализации*

На графике можно видеть, что результаты выполнения на ПК растут линейно и на небольших значениях они больше, чем у аппаратной реализации. Но на значении 8192 в аппаратной реализации происходит «перегиб» и у неё показатели производительности становятся хуже.

8. Вывод

В результате выполнения данной лабораторной работы была изучена оптимизация пропускной способности данных. Для этого была создана программа на языке C++, выполняющая суммирование четырёх значений в цикле с шагом N/4.

Для создания проекта и работы с ним был создан TCL скрипт, содержащий 14 решений. В решениях использовались разные типы памяти однопортовая и двухпортовая, директивы для разбиения массива на блоки и развертывания цикла. Также для разных решений включалась и выключалась конвейеризация. Оптимальным с точки зрения производительности оказалось sol7\_1 за счёт использования конвейеризации, двухпортовой памяти, array\_partition и развертывания цикла.

Также было выполнено измерение времени выполнения программы на ПК и аппаратной реализации. В качестве аппаратной реализации было выбрано решение sol7\_1, так как оно имело оптимальную производительность. Результаты показали, что аппаратная реализация выигрывает при небольших значениях, но с ростом числа элементов массива начинает уступать выполнению на ПК.