Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ Lab3\_Z2

Дисциплина: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Block Level I/O Interface и Port Level I/O Interface

Выполнил студент гр. Шеметов С.А.

гр.5140901/21502

Руководитель, доцент Антонов А.П.

«\_\_» \_\_\_\_\_\_ 2023

Санкт-Петербург

2023

Оглавление

[1. Задание 3](#_Toc148196317)

[2. Исходный код функции 3](#_Toc148196318)

[3. Исходный код теста 4](#_Toc148196319)

[3.1. Моделирование 4](#_Toc148196320)

[4. Создание скрипта 6](#_Toc148196321)

[5. Сравнение результатов двух решений 7](#_Toc148196322)

[5.1. Сравнение временных параметров, производительности и аппаратных затрат 7](#_Toc148196323)

[5.2. Сравнение использованных интерфейсов 8](#_Toc148196324)

[5.3. Результаты планирования (Schedule viewer) 9](#_Toc148196325)

[5.4. C/RTL моделирование 12](#_Toc148196326)

[5.4.1. Отчет по производительности 12](#_Toc148196327)

[5.4.2. Временная диаграмма 13](#_Toc148196328)

[5.5. Анализ результатов 14](#_Toc148196329)

[6. Решение sol3 15](#_Toc148196330)

[7. Измерение времени выполнения 16](#_Toc148196331)

[7.1. Измерение времени выполнения на ПК 16](#_Toc148196332)

[7.2. Измерение времени выполнения на аппаратной реализации 20](#_Toc148196333)

[7.3. Сравнение результатов 21](#_Toc148196334)

[8. Вывод 22](#_Toc148196335)

1. Задание

* Создать на языке С++ функцию (lab3\_z2.cpp), которая по входному массиву размером N, заполненного случайными целыми числами от 0 до M-1 строит гистограмму в выходном массиве размером M элементов.
* Создать на языке С++ тест (lab3\_z2\_test.cpp) для проверки работы функции.
* Создать скрипт автоматизирующий процесс создания проекта и работы с ним.
* Создать, провести исследование и сравнительный анализ двух решений.
  + Микросхема: xa7a12tcsg325-1q
  + clock period 10; clock\_uncertainty 1
  + Block Level I/O и Port Level I/O interface по умолчанию (для решения sol1)
  + Port Level I/O interface ap\_fifo ТОЛЬКО для входного массива (для решения sol2)
* Создать третье решение (sol3) и осуществить его синтез.
  + Block Level I/O по умолчанию
  + Port Level I/O interface ap\_fifo ТОЛЬКО для выходного массива
* 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* *lab3\_z2*

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

*Рис.2. Исходный код заголовочного файла lab3\_z2*

Функция принимает два массива типа short и по входному массиву размером N, заполненного случайными целыми числами от 0 до M-1, строит гистограмму в выходном массиве размером M элементов.

3. Исходный код теста

Исходный код теста для проверки функции lab3\_z2, исправленный в соответствии с заданием, приведен на рисунке 3. Тест обеспечивает: запуск функции 3 раза; формирование исходного входного массива, заполненного 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, 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[inA\_ar[k]] = expted\_ar[inA\_ar[k]] + 2. 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 lab3\_z2*. Затем добавляется файл с исходным кодом *add\_files ./source/lab3\_z2.cpp* и устанавливается функция верхнего уровня в иерархии *set\_top lab3\_z2*. Добавляется тестовый файл - *add\_files -tb ./source/lab3\_z2\_test.cpp*.

Далее создаётся первое решение *- open\_solution -reset "sol1"*. Для него назначается микросхема, период 10 нс и неопределённость 1 нс. После запускаются Си моделирование (*csim\_design -clean*), синтез (*csynth\_design*) и моделирование cosim (*cosim\_design -trace\_level all -tool xsim*). Block Level I/O и Port Level I/O interface задаются по умолчанию.

Далее создаётся второе решение - *open\_solution -reset "sol2"*. Для него задаются такие же период, неопределённость и микросхема. Устанавливается Port Level I/O interface ap\_fifo для входного массива - *set\_directive\_interface -mode ap\_fifo "lab3\_z2" inA\_ar*. Протокол *ap\_fifo* используется, когда требуется доступ к массиву последовательным образом. Затем для этого решения выполняются синтез и моделирование cosim.

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, число, Шрифт

Автоматически созданное описание**

*Рис.6. Скрипт для работы с проектом*

5. Сравнение результатов двух решений

5.1. Сравнение временных параметров, производительности и аппаратных затрат

В данной работе были реализованы два решения с разными протоколами. В первом решении используется Block Level I/O и Port Level I/O interface по умолчанию, а во втором для Port Level I/O interface устанавливается протокол ap\_fifo. Результаты сравнения приведены на рисунке 7.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

*Рис.7. Сравнение двух решений*

Сравнение данных на рисунке 7 показывает, что Estimated timing у обоих решений одинаковое. Однако Latency значительно различаются. Для первого оно 49 тактов, для второго 33 такта.

Аппаратные затраты также различаются по числу FF триггеров и LUT таблиц перекодировок. Первое решение требует немного больше FF триггеров по сравнению со вторым (12 триггеров для sol1 и 11 для sol2) и меньше LUT таблиц перекодировки (89 таблиц для sol1 и 96 для sol2).

5.2. Сравнение использованных интерфейсов

Сравнение интерфейсов на рисунках 8–9 показывает, что для массива inA\_ar были использованы разные протоколы: ap\_memory для sol1 и ap\_fifo для sol2. Различие данных протоколов заключается в том, что ap\_memory может работать с данными массива любым образом, а ap\_fifo – последовательным. Также ap\_memory имеет порты: выходной address0 (адрес элемента массива), выходной ce0 (управляющий сигнал, разрешающий чтение) и входной q0 (входные данные). А ap\_fifo: входной dout (размер буфера, в который считываются данные), входной empty\_n (разрешение на чтение данных) и выходной read (завершение чтения данных).

Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

*Рис.8. Интерфейсы для sol1*

Изображение выглядит как текст, снимок экрана, число, меню

Автоматически созданное описание

*Рис.9. Интерфейсы для sol2*

5.3. Результаты планирования (Schedule viewer)

Результаты планирования для обоих решений представлены на рисунках 10-19.

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

Автоматически созданное описание

*Рис.10. Результаты планирования для sol1. Операция inA\_ar\_addr(getelementptr)*

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

Автоматически созданное описание

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

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

Автоматически созданное описание

*Рис.12. Результаты планирования для sol1. Операция out\_ar\_addr(getelementptr)*

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

Автоматически созданное описание

*Рис.13. Результаты планирования для sol1. Операция add\_ln11(+)*

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

Автоматически созданное описание

*Рис.14. Результаты планирования для sol1. Операция out\_ar\_addr\_write\_ln11(write)*

Результаты для первого решения показаны на рисунках 10-14. Отображено 4 такта. Цикл начинается со второго такта. На втором такте из массива *inA\_ar* получается i-ый элемент (рисунок 10). Затем данный элемент записывается в переменную *temp* на втором-третьем тактах (рисунок 11). Далее на третьем такте из выходного массива *out\_ar* извлекается элемент с индексом *temp* (рисунок 12). На четвёртом такте данный элемент инкрементируется (рисунок 13) и записывается обратно в *out\_ar[temp]* (рисунок 14).

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

Автоматически созданное описание

*Рис.15. Результаты планирования для sol2. Операция add\_ln8(+)*

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

Автоматически созданное описание

*Рис.16. Результаты планирования для sol2. Операция temp(read)*

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

Автоматически созданное описание

*Рис.17. Результаты планирования для sol2. Операция out\_ar\_addr(getelementptr)*

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

Автоматически созданное описание

*Рис.18. Результаты планирования для sol2. Операция add\_ln11(+)*

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

Автоматически созданное описание

*Рис.19. Результаты планирования для sol2. Операция out\_ar\_addr\_write\_ln11(write)*

Результаты для первого решения показаны на рисунках 15-19. В отличии от первого решения во втором решении представлено 3 такта и одна итерация цикла занимает 2 такта. Цикл начинается со второго такта. На втором такте происходит работа с итератором *i* (рисунок 15). Обращение к *inA\_ar* не выполняется, данные получаются из памяти.Также на втором такте элемент записывается в переменную *temp* (рисунок 16) и из выходного массива *out\_ar* извлекается элемент с индексом *temp* (рисунок 17). На третьем такте данный элемент инкрементируется (рисунок 18) и записывается обратно в *out\_ar[temp]* (рисунок 19).

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

5.4. C/RTL моделирование

5.4.1. Отчет по производительности

Отчеты по производительности для обоих решений, полученный при моделировании cosim, приведен на рисунке 20. Из отчётов следует, что полученные Latency и II совпадают со значениями при синтезе, показанными на рисунке 7.

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

Автоматически созданное описание

*Рис.20. Отчеты по производительности C/RTL*

5.4.2. Временная диаграмма

На рисунке 21 приведена временная диаграмма совместного C/RTL моделирования для первого решения. Она отображает как происходит чтение данных из массива по умолчанию. Поступает сигнал inA\_ar\_ce0 разрешающий чтение данных по адресу, записанному на шине inA\_ar\_address0[3:0]. После спада ce0 на шину inA\_ar\_q0[15:0] выводится прочитанное число. Это повторяется 16 раз. Latency занимает 49 тактов, а II – 50 тактов, что соответствует рисунку 7.

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

Автоматически созданное описание

*Рис.21. Временная диаграмма C/RTL для sol1*

На рисунке 22 приведена временная диаграмма совместного C/RTL моделирования для второго решения. На ней можно видеть, что процесс чтения данных с протоколом ap\_fifo. Сигнал inA\_ar\_empty\_n устанавливается в 1, если в массиве ещё есть непрочитанные данные, и начинается чтение. Когда элемент массива будет считан сигнал read устанавливается в 1. При новом чтении он сбрасывается в 0. Так повторяется, пока все данные из массива не будут прочитаны. Latency занимает 33 такта, а II – 34 тактов, что соответствует рисунку 7.

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

Автоматически созданное описание

*Рис.22. Временная диаграмма C/RTL для sol2*

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 23–24.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

*Рис.23. Электронная таблица для двух решений*

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

*Рис.24. График для двух решений*

С точки зрения производительности лучшим решением является второе, так как оно затрачивает меньше времени. Это обусловлено использованием протокола ap\_fifo, который позволяет быстрее считывать данные из массива.

С точки зрения аппаратных затрат второе решение имеет незначительное преимущество за счёт меньшего числа LUT, так что его не стоит рассматривать как оптимальное.

6. Решение sol3

В решении sol3 были установлены Block Level I/O по умолчанию и Port Level I/O interface ap\_fifo только для выходного массива out\_ar (рисунок 25), а затем для данного решения был выполнен синтез.

Результаты приведены на рисунке 26.

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

Автоматически созданное описание

*Рис.25. Директивы для sol3*

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Значок на компьютере

Автоматически созданное описание

*Рис.26. Результат синтеза sol3*

Как можно видеть на рисунке 26, синтез выполнить не удалось и возникла ошибка. Данная ошибка возникает из-за того, что протокол не поддерживает операции чтения и записи для выходного массива.

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 lab3\_z2\_test.cpp был создан отдельный, модернизированный, тест lab3\_z2\_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, программное обеспечение, число

Автоматически созданное описание

*Рис.27. Код теста* *lab3\_z2\_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 M и N – 4096. 8192, 16384 (рисунок 28). Результаты запуска представлены на рисунках 29–31.

Изображение выглядит как текст, снимок экрана, Шрифт

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 M=N=4096*

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

Автоматически созданное описание

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

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

Автоматически созданное описание

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

7.2. Измерение времени выполнения на аппаратной реализации

Для измерения времени выполнения на аппаратной реализации был осуществлён синтез решения sol2 для трёх значений M и N – 4096. 8192, 16384. Результаты запуска представлены на рисунках 32–34.

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

* для 4096: 8194 такта \* 8,660 нс = 70960,04 нс
* для 8192: 16386 такта \* 8,660 нс = 141902,76 нс
* для 16384: 49154 такта \* 8,660 нс = 425673,64 нс

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

Автоматически созданное описание

*Рис.32. Результаты теста на аппаратной реализации для M=N=4096*

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

Автоматически созданное описание

*Рис.33. Результаты теста на аппаратной реализации для M=N=8192*

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

Автоматически созданное описание

*Рис.34. Результаты теста на аппаратной реализации для M=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 (M и N) ко времени выполнения. Результаты представлены на рисунках 35–36.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

*Рис.23. Электронная таблица для ПК и аппаратной реализации*

Изображение выглядит как снимок экрана, текст, линия, График

Автоматически созданное описание

*Рис.24. 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 10–20 раз больше. Более медленное выполнения аппаратной реализации связано с тем, что она имеет меньшую частоту по сравнению с ПК (около 1 ГГц против 2 ГГц).

Для того, чтобы получить лучшие результаты, необходимо распараллеливать аппаратную реализацию.

8. Вывод

В результате выполнения данной лабораторной работы были получены навыки работы с Block Level I/O и Port Level I/O interface. Для этого на языке С++ была написана программа построения гистограммы в выходном массиве размером M элементов и тест к ней. Для создания проекта и работы с ним был создан TCL скрипт, содержащий два решения. В одном решении o Block Level I/O и Port Level I/O interface были установлены по умолчанию, а во втором задан протокол ap\_fifo для входных данных. В результате применение данного протокола дало преимущество в производительности во втором решении.

Кроме того, было создано третье решение, в котором ap\_fifo использовался для выходного массива, что привело к ошибке, так как данный протокол не поддерживает работу с выходными данными.

Также было выполнено измерение времени выполнения программы на ПК и аппаратной реализации. Результаты показали, что аппаратная реализация значительно проигрывает в производительности. Для улучшения показателей необходимо распараллеливать аппаратную реализацию.