Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ Lab3\_Z1

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](#_Toc147975554)

[2. Исходный код функции 3](#_Toc147975555)

[3. Исходный код теста 4](#_Toc147975556)

[3.1. Моделирование 4](#_Toc147975557)

[4. Создание скрипта 6](#_Toc147975558)

[5. Сравнение результатов двух решений 7](#_Toc147975559)

[5.1. Сравнение временных параметров, производительности и аппаратных затрат 7](#_Toc147975560)

[5.2. Сравнение использованных интерфейсов 8](#_Toc147975561)

[5.3. Результаты планирования (Schedule viewer) 9](#_Toc147975562)

[5.4. C/RTL моделирование 11](#_Toc147975563)

[5.4.1. Отчет по производительности 11](#_Toc147975564)

[5.4.2. Временная диаграмма 11](#_Toc147975565)

[6. Анализ результатов 12](#_Toc147975566)

[7. Вывод 14](#_Toc147975567)

1. Задание

* Создать на языке С++ функцию (lab3\_z1.cpp), выполняющую операции над тремя аргументами inA, inB, inC:

#include "lab3\_z1.h"

data\_ret lab3\_z1 (data\_in inA, data\_in inB, pnt\_in \*inC, pnt\_out \*res) {

data\_ret temp;

temp = inA \* \*inC;

\*res = temp + inB;

return temp;

}

* Создать на языке С++ тест (lab3\_z1\_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 8; clock\_uncertainty 1
  + Block Level I/O interface по умолчанию (для решения sol1)
  + Block Level I/O interface ap\_ctrl\_chain (для решения sol2)

2. Исходный код функции

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

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

Автоматически созданное описание*

*Рис.1. Исходный код функции lab3\_z1*

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

Автоматически созданное описание

*Рис.2. Исходный код заголовочного файла lab3\_z1*

Функция принимает два-аргумента скаляра и два указателя, выполняет операции арифметически операции над аргументами, записывает полученное значение по указателю \*res и возвращает полученный результат.

3. Исходный код теста

Исходный код теста для проверки функции lab3\_z1, исправленный в соответствии с заданием, приведен на рисунке 3. Тест обеспечивает: запуск функции 3 раза, заполнение данных случайными числами, очистку результата между запусками функции и 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, дисплей, программное обеспечение

Автоматически созданное описание*

*Рис.4. Лог результата выполнения моделирования*

Для проверки правильности работы теста в него было внесено изменение в ожидаемый результат - exp\_res = inA \* inC + inB + 1. Результат моделирования приведён на рисунке 5. Он показывает, что тест отрабатывает ошибку корректно.

Изображение выглядит как текст, электроника, 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\_z1*. Затем добавляется файл с исходным кодом *add\_files ./source/lab3\_z1.cpp*, файл *add\_files ./source/lab3\_z1.h* и устанавливается функция верхнего уровня в иерархии *set\_top lab3\_z1*. Добавляется тестовый файл - *add\_files -tb ./source/lab3\_z1\_test.cpp*.

Далее создаётся первое решение *- open\_solution -reset "sol1"*. Для него назначается микросхема, период 8 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 interface задаются по умолчанию.

Далее создаётся второе решение - *open\_solution -reset "sol2"*. Для него задаются такие же период, неопределённость и микросхема. Устанавливается Block Level I/O interface с протоколом ap\_ctrl\_chain - *set\_directive\_interface -mode ap\_ctrl\_chain "lab3\_z1"*. Протокол *ap\_ctrl\_chain* используется для поддержки цепочек конвейеров. Аналогичен протоколу *ap\_ctrl\_hs*, но предоставляет дополнительный входной порт *ap\_continue*. 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\_z1.tcl*.

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

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 interface по умолчанию, а во втором с протоколом ap\_ctrl\_chain. Результаты сравнения приведены на рисунке 7.

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

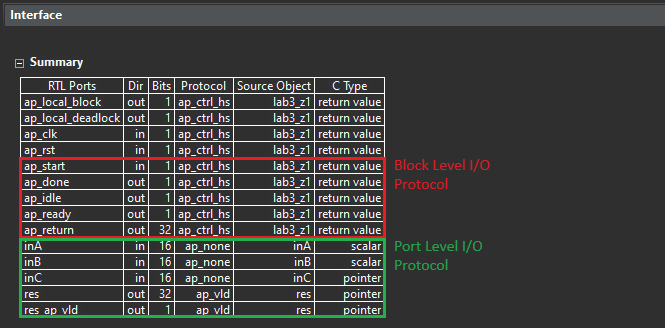
Автоматически созданное описание

*Рис.7. Сравнение двух решений*

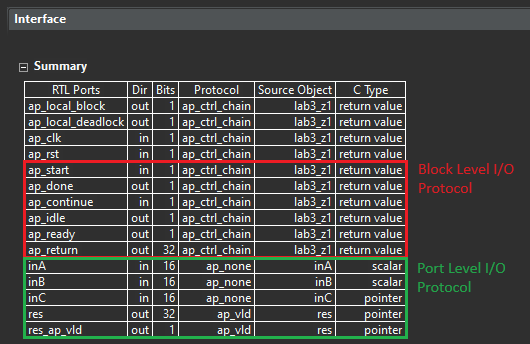
Сравнение данных на рисунке 7 показывает, что временные параметры и производительность у двух решений одинаковые. Однако аппаратные затраты различаются. Первое решение требует меньше FF триггеров по сравнению со вторым (4 триггера для sol1 и 69 для sol2) и меньше LUT таблиц перекодировки (60 таблиц для sol1 и 89 для sol2).

5.2. Сравнение использованных интерфейсов

Сравнение интерфейсов на рисунках 8–9 показывает, что для ap портов разные протоколы, а также во втором решении появляется новый порт ap\_continue. Интерфейсы для Block Level I/O interface и Port Level I/O interface отмечены на рисунках красным и зелёным соответственно.



*Рис.8. Интерфейсы для sol1*



*Рис.9. Интерфейсы для sol2*

5.3. Результаты планирования (Schedule viewer)

Результаты планирования для обоих решений совпадают. Они приведены на рисунках 10-13.

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

Автоматически созданное описание

*Рис.10. Результаты планирования. Операция inA\_read(read)*

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

Автоматически созданное описание

*Рис.11. Результаты планирования. Операция temp(\*)*

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

Автоматически созданное описание

*Рис.12. Результаты планирования. Операция inB\_read(read)*

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

Автоматически созданное описание

*Рис.13. Результаты планирования. Операция add\_ln6(+)*

На рисунках 10-13 представлено 4 такта. На первом такте происходит чтение *inA* и *inС* (рисунок 10). Со второго по четвёртый такт выполняется в переменной *temp* (рисунок 11). На втором такте также происходит чтение переменной *inB* (рисунок 12), выполнение сложения её с произведением и запись результата (рисунок 13).

Значение Latency совпадает с результатами полученными на рисунке 7 и равно 3 тактам, также совпадает и Initiation Interval (II), и он равен 4 тактам.

5.4. C/RTL моделирование

5.4.1. Отчет по производительности

Отчеты по производительности для обоих решений, полученный при моделировании cosim, приведен на рисунке 14. Из отчётов следует, что результаты для обоих решений одинаковые. Также 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.14. Отчеты по производительности C/RTL*

5.4.2. Временная диаграмма

На рисунке 15 приведена временная диаграмма совместного C/RTL моделирования для первого решения. Она отображает 3 цикла запуска аппаратной реализации созданной функции. На ней видно четыре управляющих сигнала: ap\_start – блок может начать выполнение операции; ap\_done - операция блока завершается; ap\_ready – блок готов к поступлению новых данных; ap\_idle – блок выполняет операции или находится в режиме ожидания. На шинах inA, inB, inC задаются входные значения. На шине res[31:0] отображаются правильные выходные значения после Latency. Latency занимает 3 такта (3\*8 нс = 24 нс), а II - 4 такта (32 нс).

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

Автоматически созданное описание

*Рис.15. Временная диаграмма C/RTL для sol1*

На рисунке 16 приведена временная диаграмма совместного C/RTL моделирования для второго решения. Кроме описанных для sol1 сигналов данное решение имеет новый – ap\_continue – указывает на то, готов ли следующий блок в конвейере принимать новые данные.

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

Автоматически созданное описание

*Рис.16. Временная диаграмма C/RTL для sol2*

6. 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 17–18.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

*Рис.17. Электронная таблица для двух решений*

Изображение выглядит как снимок экрана, линия, текст, График

Автоматически созданное описание

*Рис.18. График для двух решений*

С точки зрения производительности оба решения имеют одинаковые значения. Однако при сравнении аппаратных затрат видно, что оптимальным решением является первое, так как имеет меньшее число FF триггеров и таблиц перекодировок.

Увеличение аппаратных затрат во втором решении обусловлено тем, что протокол ap\_ctrl\_chain изменяет структуру самого интерфейса и на выходе получается отличное от sol1 аппаратное решение с большими затратами.

7. Вывод

В результате выполнения данной лабораторной работы были получены навыки работы с Block Level I/O Interface. Для этого на языке С++ была написана программа для выполнения арифметических операций над несколькими операндами и тест к ней. Для создания проекта и работы с ним был создан TCL скрипт, содержащий два решения. В одном решении Block Level I/O Interface был установлен по умолчанию, а во втором задан режим ap\_ctrl\_chain. В результате во втором решении была изменена аппаратная структура проекта, что увеличило аппаратные затраты.