Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчет по лабораторной работе №8\_3**

**Курс: «Проектирование реконфигурируемых гибридных вычислительных систем»**

**Тема: Dataflow.** **Conditional Execution of Tasks**

Выполнил студент гр. 3540901/81501 Селиверстов Я.А.

(подпись)

Руководитель Антонов А.П.

(подпись)

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

Санкт – Петербург

2019

Оглавление

[1. Задание 4](#_Toc30294086)

[2. Первое решение 7](#_Toc30294087)

[2.1. Моделирование 8](#_Toc30294088)

[2.2. Синтез 8](#_Toc30294089)

[3. Второе решение 10](#_Toc30294090)

[3.1. Синтез 11](#_Toc30294091)

[3.2. C\RTL моделирование 13](#_Toc30294092)

[4. Третье решение 14](#_Toc30294093)

[4.1. Синтез 14](#_Toc30294094)

[5. Выводы 17](#_Toc30294095)

1. Задание

* Создать проект lab8\_3
* Микросхема: xa7a12tcsg325-1q
* Создать две функции (см. Текст ниже) – исходную и модифицированную - и провести их анализ.

***Conditional Execution of Tasks***

***The DATAFLOW optimization does not optimize tasks that are conditionally executed. The***

***following example highlights this limitation. In this example, the conditional execution of Loop1***

***and Loop2 prevents Vivado HLS from optimization the data flow between these loops, because***

***the data does not flow from one loop into the next.***

**void foo\_b(int data\_in1[N], int data\_out[N], int sel) {**

**int temp1[N], temp2[N];**

**if (sel) {**

**Loop1: for(int i = 0; i < N; i++) {**

**temp1[i] = data\_in[i] \* 123;**

**temp2[i] = data\_in[i];**

**}**

**} else {**

**Loop2: for(int j = 0; j < N; j++) {**

**temp1[j] = data\_in[j] \* 321;**

**temp2[j] = data\_in[j];**

**}**

**}**

**Loop3: for(int k = 0; k < N; k++) {**

**data\_out[k] = temp1[k] \* temp2[k];**

**}**

**}**

***To ensure each loop is executed in all cases, you must transform the code as shown in the***

***following example. In this example, the conditional statement is moved into the first loop. Both***

***loops are always executed, and data always flows from one loop to the next.***

**void foo\_m(int data\_in[N], int data\_out[N], int sel) {**

**int temp1[N], temp2[N];**

**Loop1: for(int i = 0; i < N; i++) {**

**if (sel) {**

**temp1[i] = data\_in[i] \* 123;**

**} else {**

**temp1[i] = data\_in[i] \* 321;**

**}**

**Loop2: for(int j = 0; j < N; j++) {**

**temp2[j] = data\_in[j];**

**}**

**Loop3: for(int k = 0; k < N; k++) {**

**data\_out[k] = temp1[k] \* temp2[k];**

**}**

**}**

* Создать тест lab8\_3\_test.c для проверки функций выше.
* Для функции **foo\_b**
  + задать: clock period 10; clock\_uncertainty 0.1
  + осуществить моделирование (с выводом результатов в консоль)
  + осуществить синтез для:
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнить Zoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * resource viewer (выполнить Zoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
* Для функции **foo\_m**
  + задать: clock period 10; clock\_uncertainty 0.1
  + осуществить моделирование (с выводом результатов в консоль)
  + осуществить синтез для случая **FIFO for the memory buffers**:
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнить Zoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * resource viewer (выполнить Zoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * Dataflow viewer
  + осуществить синтез для случая **ping-pong buffers**:
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнить Zoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * resource viewer (выполнить Zoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * Dataflow viewer
  + Осуществить C|RTL моделирование для случая **FIFO for the memory buffers**
    - Привести результаты из консоли
    - Открыть временную диаграмму (все сигналы)
      * Отобразить два цикла обработки на одном экране
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
* Выводы
  + Объяснить отличия в синтезе foo\_b и двух вариантов foo\_m между собой

1. Первое решение

Исходный текст подготовленной для синтеза функции и теста приведён ниже:

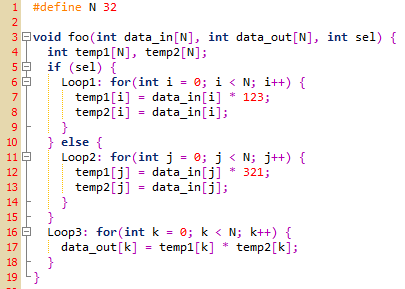


Рис. 2.1. Исходный код синтезируемой функции

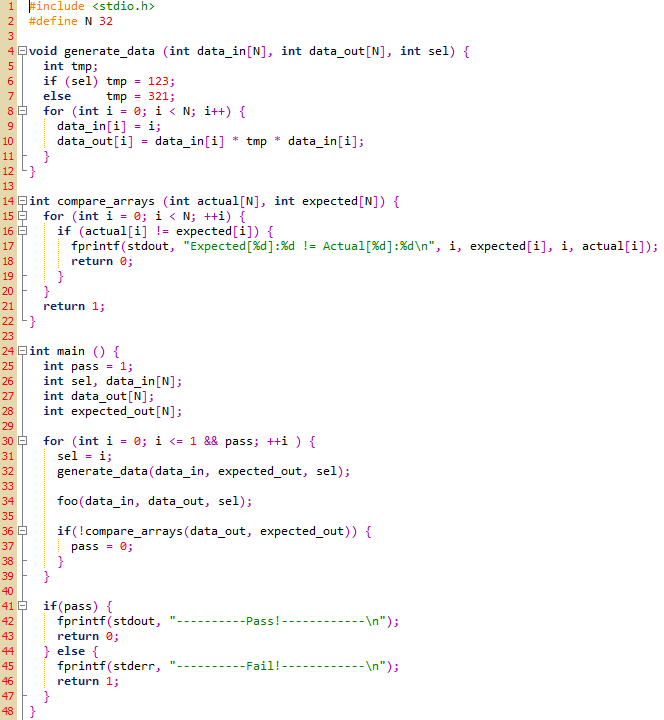


Рис. 2.2. Исходный код теста

* 1. Моделирование

При запуске моделирования можно увидеть, что тест успешно пройден:

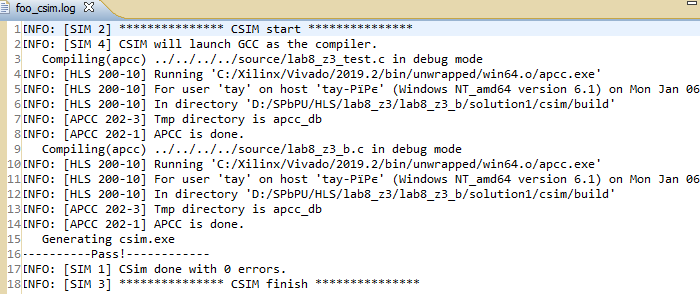


Рис. 2.3. Результаты моделирования

* 1. Синтез

Приведем в отчете требуемые данные о проекте:

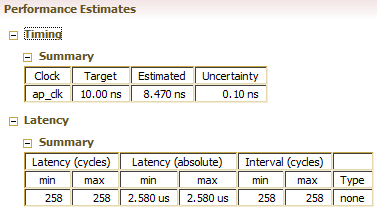


Рис. 2.4. Производительность

Здесь можно увидеть, что достигнутая задержка равна 8.47 + 0.1, что укладывается в заданные нами требования к тактовой частоте.

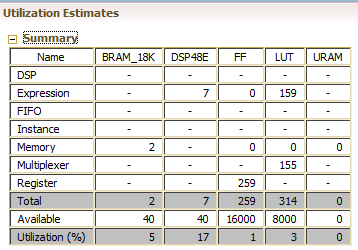


Рис. 2.5. Занимаемые ресурсы

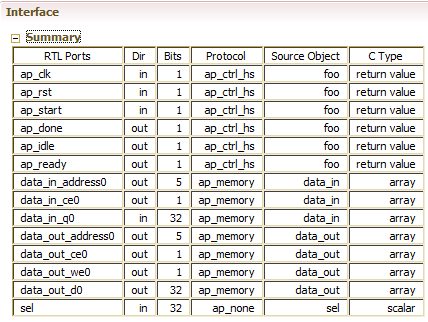


Рис. 2.6. Применяемые интерфейсы

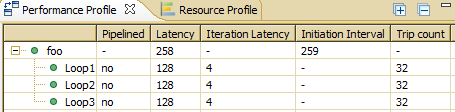


Рис. 2.7. Профиль производительности

На этом рисунке видно, что задержка получения выходного значения составляет 258 тактов с момента старта, а задержка после старта до готовности приема новых данных – 259:

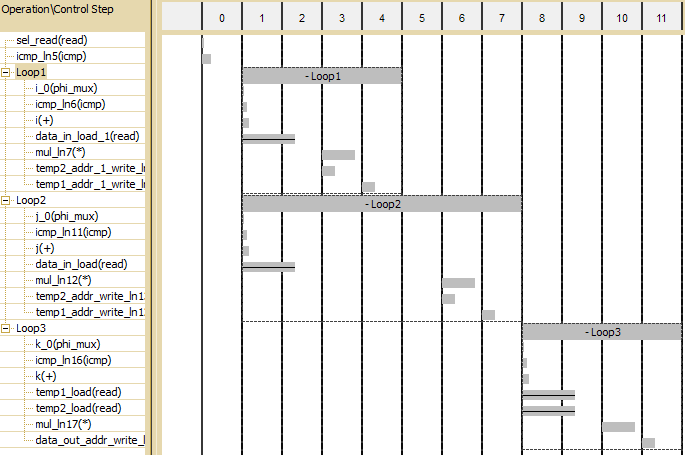


Рис. 2.8. Временная диаграмма

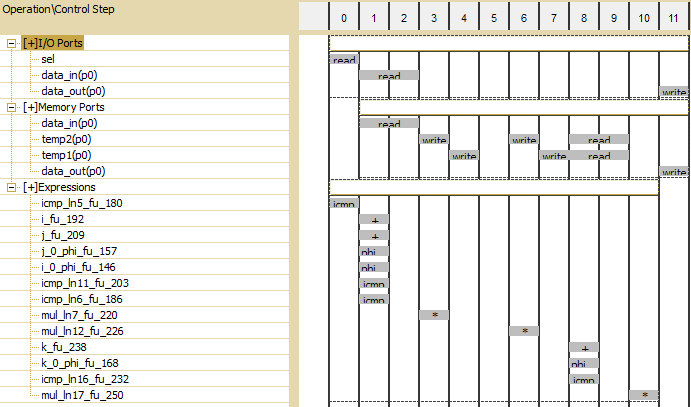


Рис. 2.9. Диаграмма использования ресурсов

1. Второе решение

Исходный текст подготовленной для синтеза функции приведён ниже:

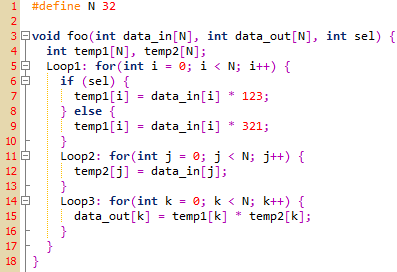


Рис. 3.1. Исходный код синтезируемой функции

Добавим директиву, которая изменяет способ передачи данных.

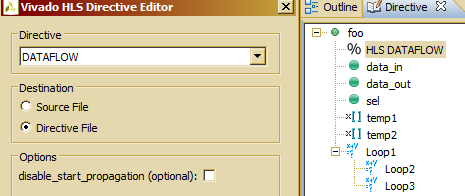


Рис. 3.2. Добавление директивы

* 1. Синтез

Приведем в отчете требуемые данные о проекте:

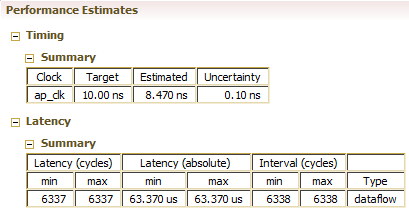


Рис. 3.3. Производительность

Здесь можно увидеть, что достигнутая задержка равна 8.47 + 0.1, что укладывается в заданные нами требования к тактовой частоте.

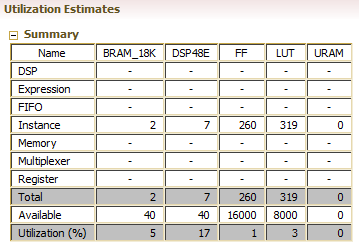


Рис. 3.4. Затрачиваемые ресурсы

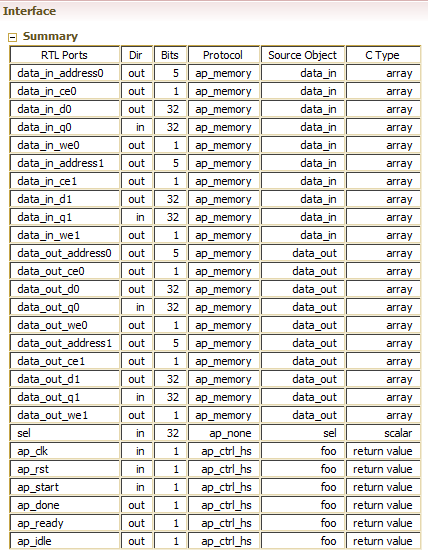


Рис. 3.5. Применяемые интерфейсы



Рис. 3.6. Профиль производительности

На этом рисунке видно, что задержка получения результатов с момента старта – 6337 тактов, а задержка после старта до готовности приема новых данных – 6338:

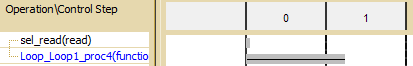


Рис. 3.7. Временная диаграмма

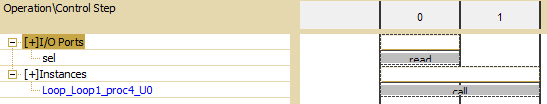


Рис. 3.8. Диаграмма использования ресурсов

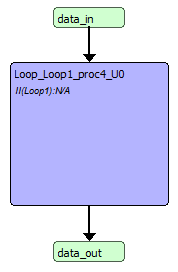


Рис. 3.9. Диаграмма потоков данных

* 1. C\RTL моделирование

При совместном моделировании, программа отобразила те же самые, ожидаемые нами значения Latency и II:

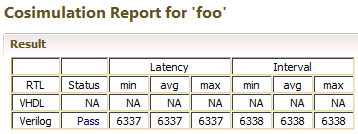


Рис. 3.10. C\RTL моделирование

Покажем временную диаграмму совместного моделирования с отмеченными на ней Latency и Initiation Interval:

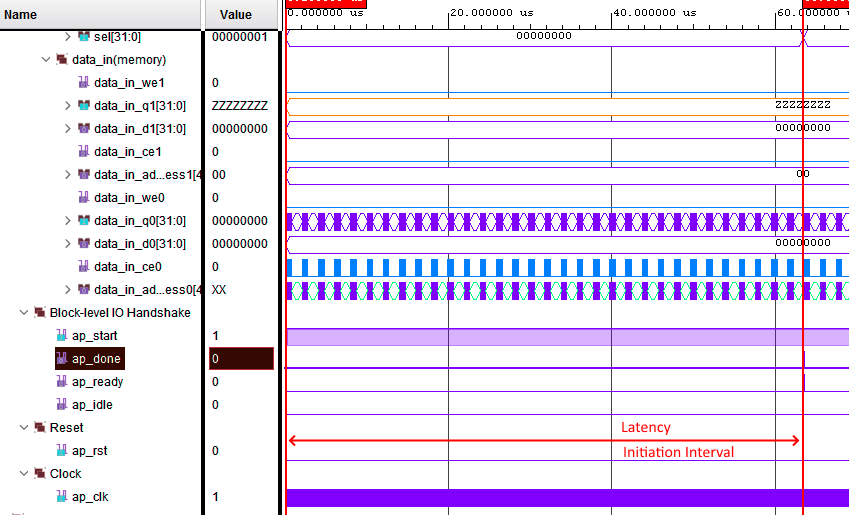


Рис. 3.11. Временная диаграмма совместного моделирования

1. Третье решение

Добавим директиву, которая изменяет способ обмена данными.

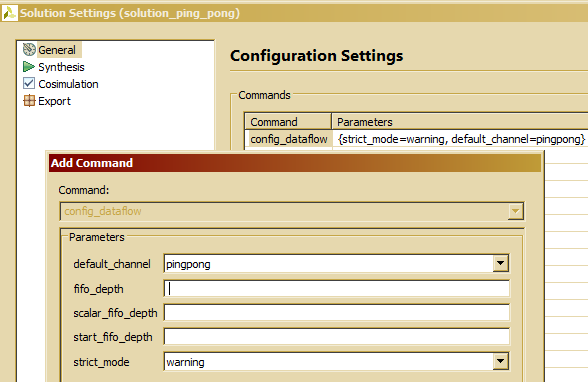


Рис. 4.1. Добавление директивы

* 1. Синтез

Приведем в отчете требуемые данные о проекте:

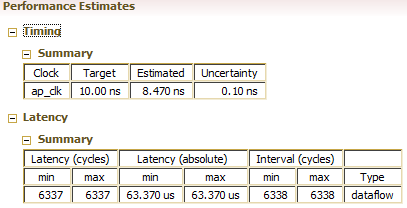


Рис. 4.2. Производительность

Здесь можно увидеть, что достигнутая задержка равна 8.47 + 0.1, что укладывается в заданные нами требования к тактовой частоте.

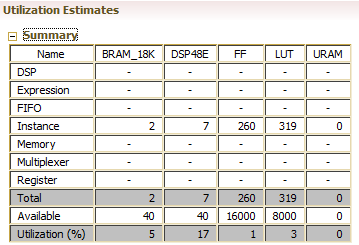


Рис. 4.3. Затрачиваемые ресурсы

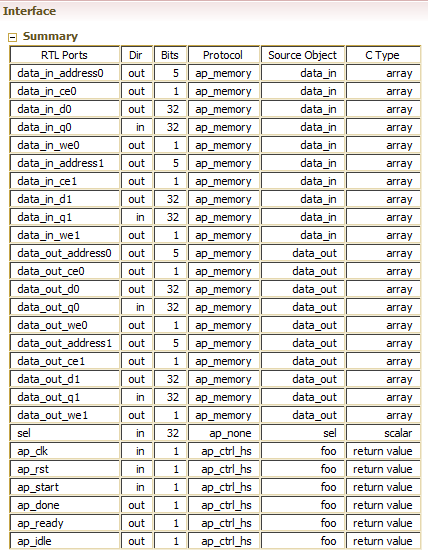


Рис. 4.4. Применяемые интерфейсы



Рис. 4.5. Профиль производительности

На этом рисунке видно, что задержка получения результатов с момента старта – 6337 тактов, а задержка после старта до готовности приема новых данных – 6338:

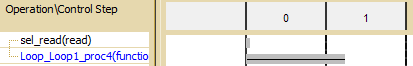


Рис. 4.6. Временная диаграмма

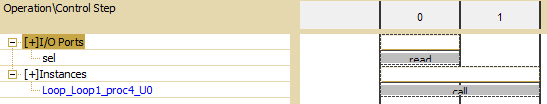


Рис. 4.7. Диаграмма использования ресурсов

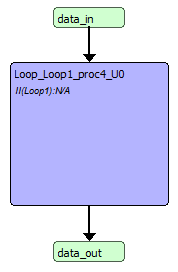


Рис. 4.8. Диаграмма потоков данных

Видно, что решение полностью соответствует предыдущему.

1. Выводы

В данной лабораторной работе были рассмотрены варианты применения директивы Dataflow. В первом решении не используются директивы, выполнение циклов в функции происходит последовательно.

Основное отличие данной работы от предыдущей (lab8\_z2) состоит в применении операции ветвления, что негативно сказалось на производительности полученного устройства, в особенности при использовании директивы Dataflow.