Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Проектирование реконфигурируемых гибридных вычислительных систем

Отчёт по лабораторной работе №4.1

«Исследование Dataflow»

Выполнила студентка гр. 3540901/11501 Дуботолкова Н.Д.

Принял преподаватель Антонов А. П.

Санкт-Петербург

2022

Содержание

[1. Задание 3](#_Toc119173613)

[2. Создание скрипта 6](#_Toc119173614)

[3. Cравнение решений 7](#_Toc119173615)

[3.1. Аппаратные затраты 7](#_Toc119173616)

[3.2. Scheduler viewer 8](#_Toc119173617)

[4. Временные диаграммы 12](#_Toc119173618)

[5. Сравнительный анализ 13](#_Toc119173619)

[Выводы 15](#_Toc119173620)

# Задание

Создать на языке Си функцию (см.Рисунок 1), в котором входной массив умножается на число. В качестве выходных значений – 2 массива типа int, где первый – это умножение полученного массива на 22, а второй – умножение полученного массива на 33.

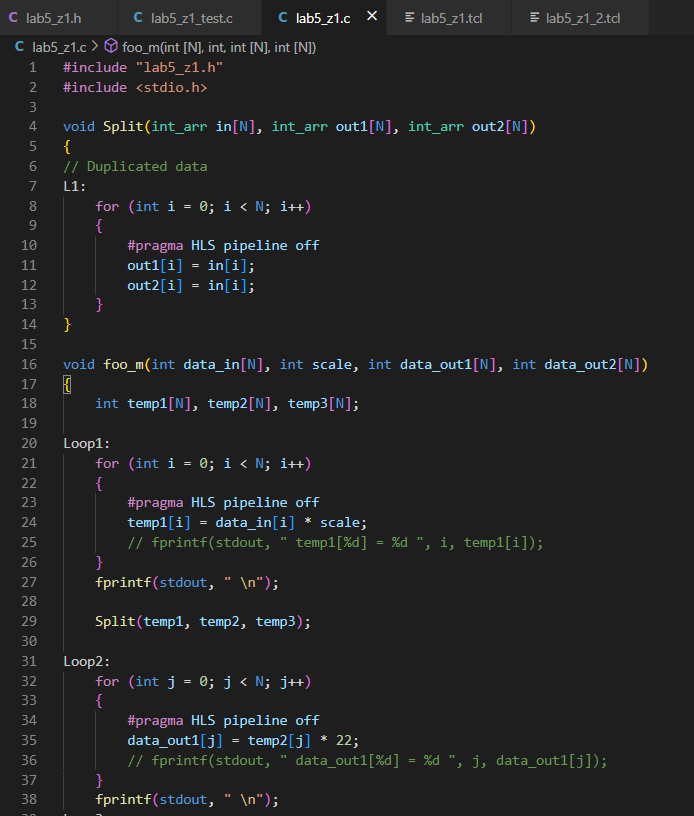
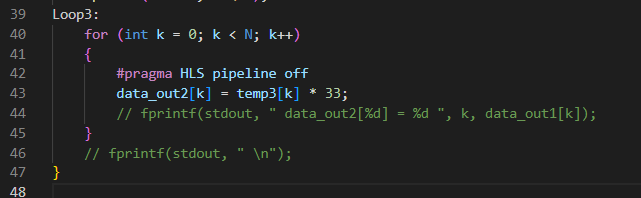
  


Рисунок . Функция умножения двух векторов

В файле lab5\_1.h (см. Рисунок 2) должны быть определены: N = 16384, тип данных int\_arr, имеющий тип int.

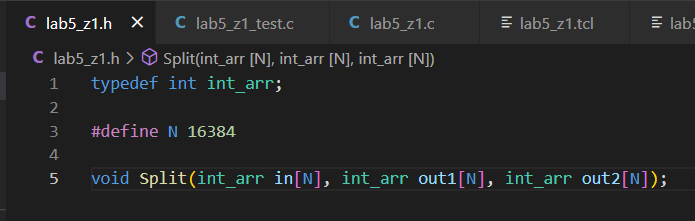
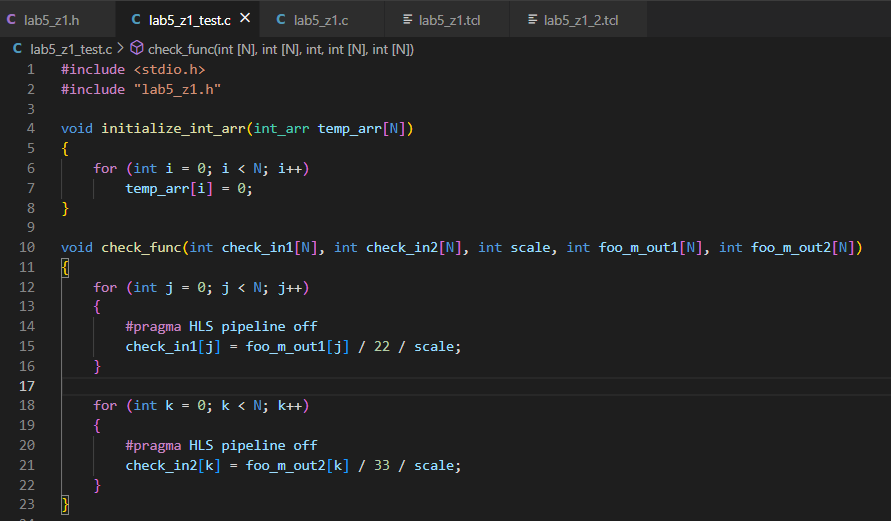


Рисунок . Файл заголовков

Написать тест, который при неправильном результате в любом из запусков функции тест должен сообщать об ошибке (см. Рисунок 3). Тест содержит обратные действия: сначала выходные массивы делятся на 22 и 33 соотвественно, а потом делятся на рандомное число scale. Полученные массивы должны совпадать между собой и c исходным входным массивом.



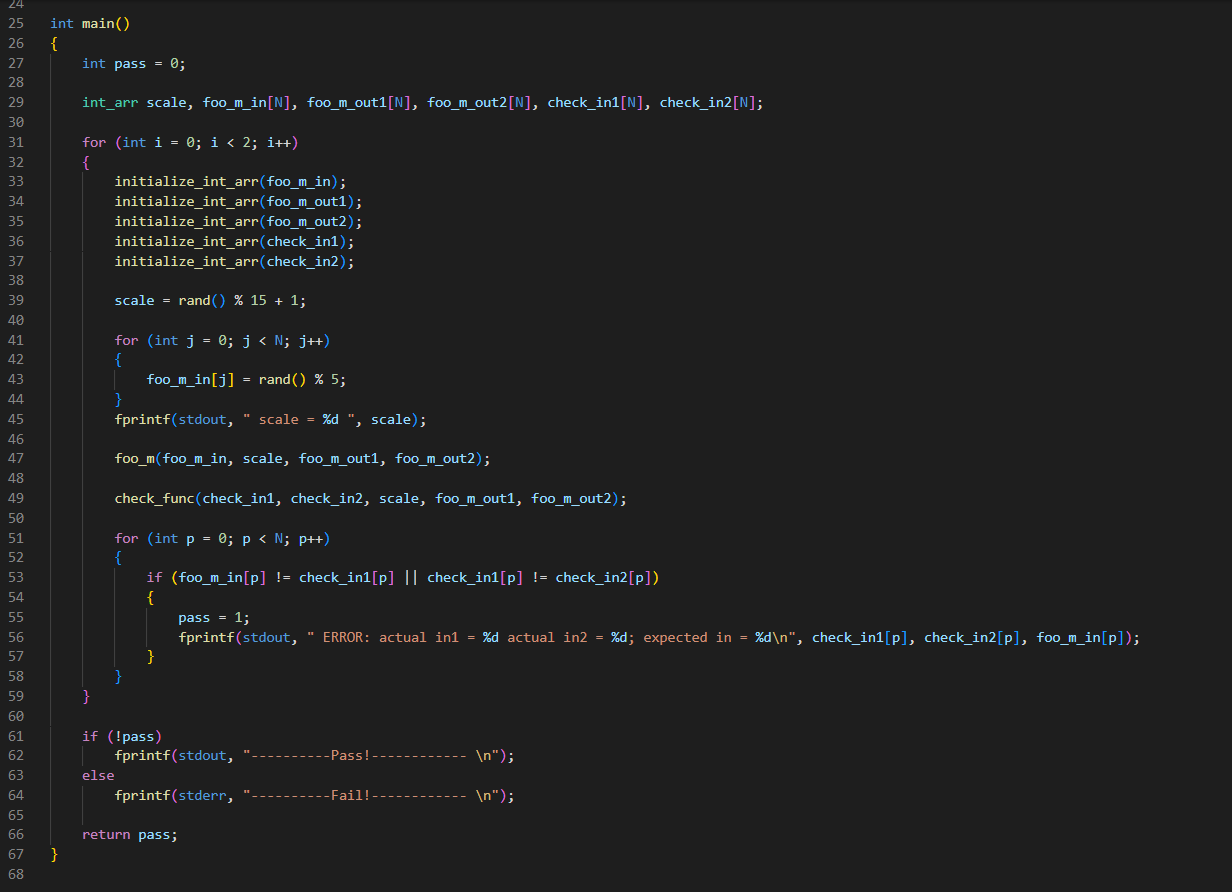
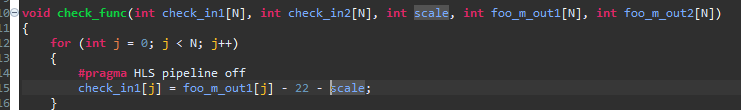


Рисунок . Тест функции

* 1. Проверка теста на корректность

Сделаем так, чтобы тест упал, заменив в нём операцию деления на вычитание (Рисунок 4).

Рисунок . Редактирование теста

Запустим проект и убедимся, тест провален (Рисунок 5).

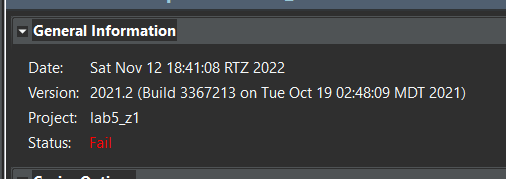


Рисунок . Результат теста

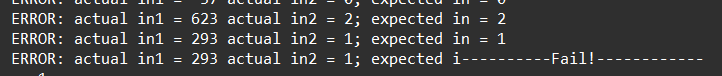


Рисунок . Вывод в консоли

# Создание скрипта

Необходимо создать скрипт с 3-мя решениями, для того, чтобы выбрать оптимальный clock period:

* clock period 6, 10, 14; clock\_uncertainty 1
* осуществить моделирование
* осуществить синтез

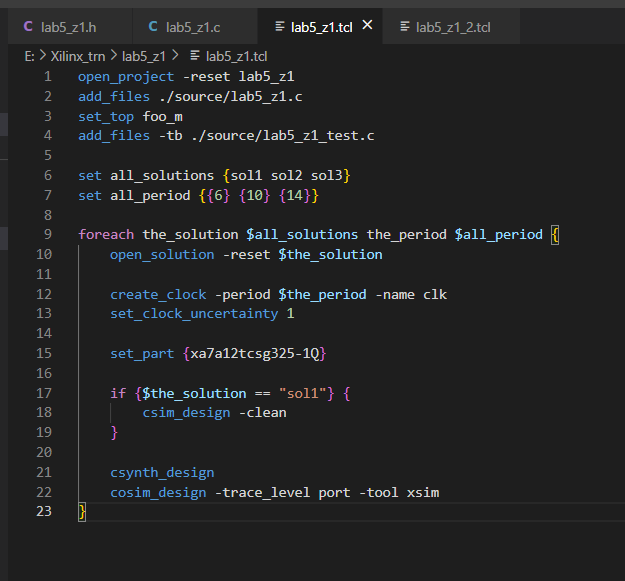


Рисунок . Командный файл для выявления оптимального решения

После запуска командного файла убеждаемся, что все решения созданы (Рисунок 8).

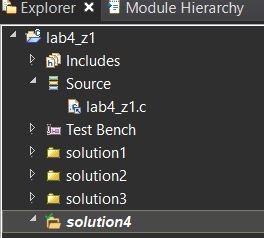


Рисунок . Проверка создания решений

# Cравнение решений

## Аппаратные затраты

Сравним полученные 3 решение и выберем оптимальное (Рисунок 9)

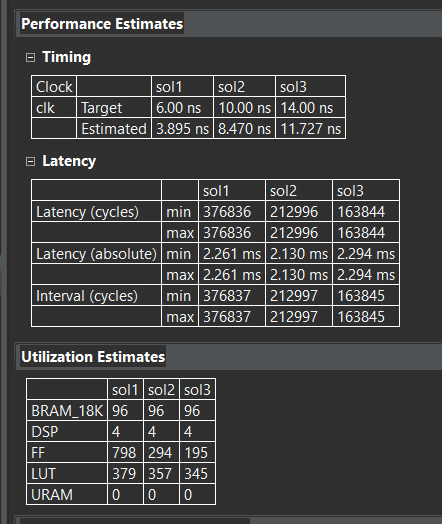


Рисунок . Сравнение решений по аппаратным затратам

Для удобства определения построим таблицу (Таблица 1). Из-за разницы в цифрах были посмтроены два графика (Рисунок 10, Рисунок 11).

Таблица Сравнение 3х решений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | solutionl 1 | solutionl 2 | solutionl 3 |  |
| Clock | Target ( ns) | 6 | 10 | 14 |  |
| Estimated ( ns) | 3,895 | 8,470 | 11,727 |  |
| Iteration Interval | (cycles) | 376837 | 212997 | 163845 |  |
| (ns) | 1467780,12 | 1804084,59 | 1921410,32 |  |
| Resources | FF | 798 | 294 | 195 |  |
| LUT | 379 | 357 | 345 |  |

Рисунок Iteration Interval (ns)

Рисунок Аппаратные затраты

По результатам видно, что оптимальным будет второе решение, так как имеет низкие аппаратные затраты и среднее время выполнения.

## Scheduler viewer

Посмотрим scheduler viewer второго решения.

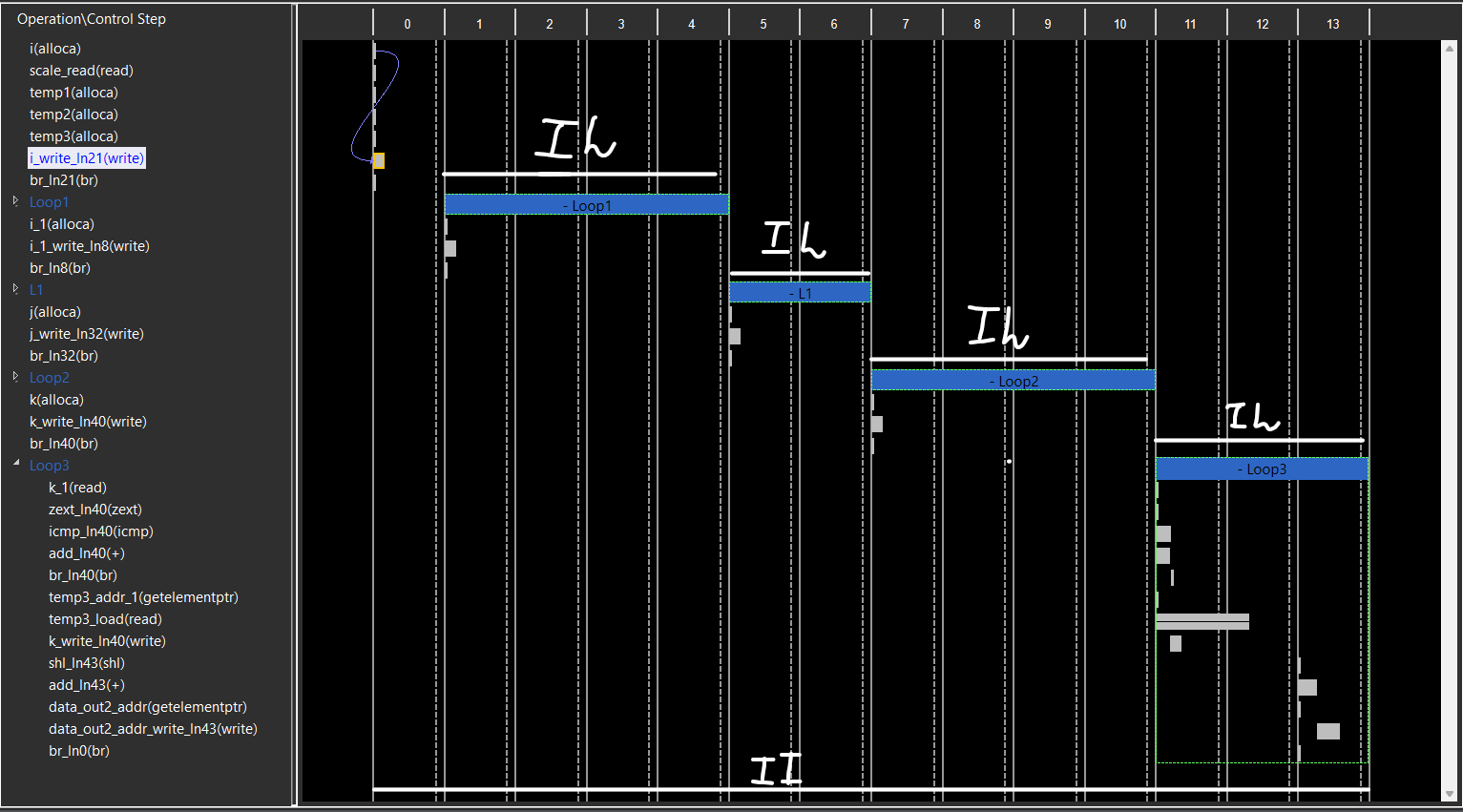


Рисунок Окно планировщика для второго решения

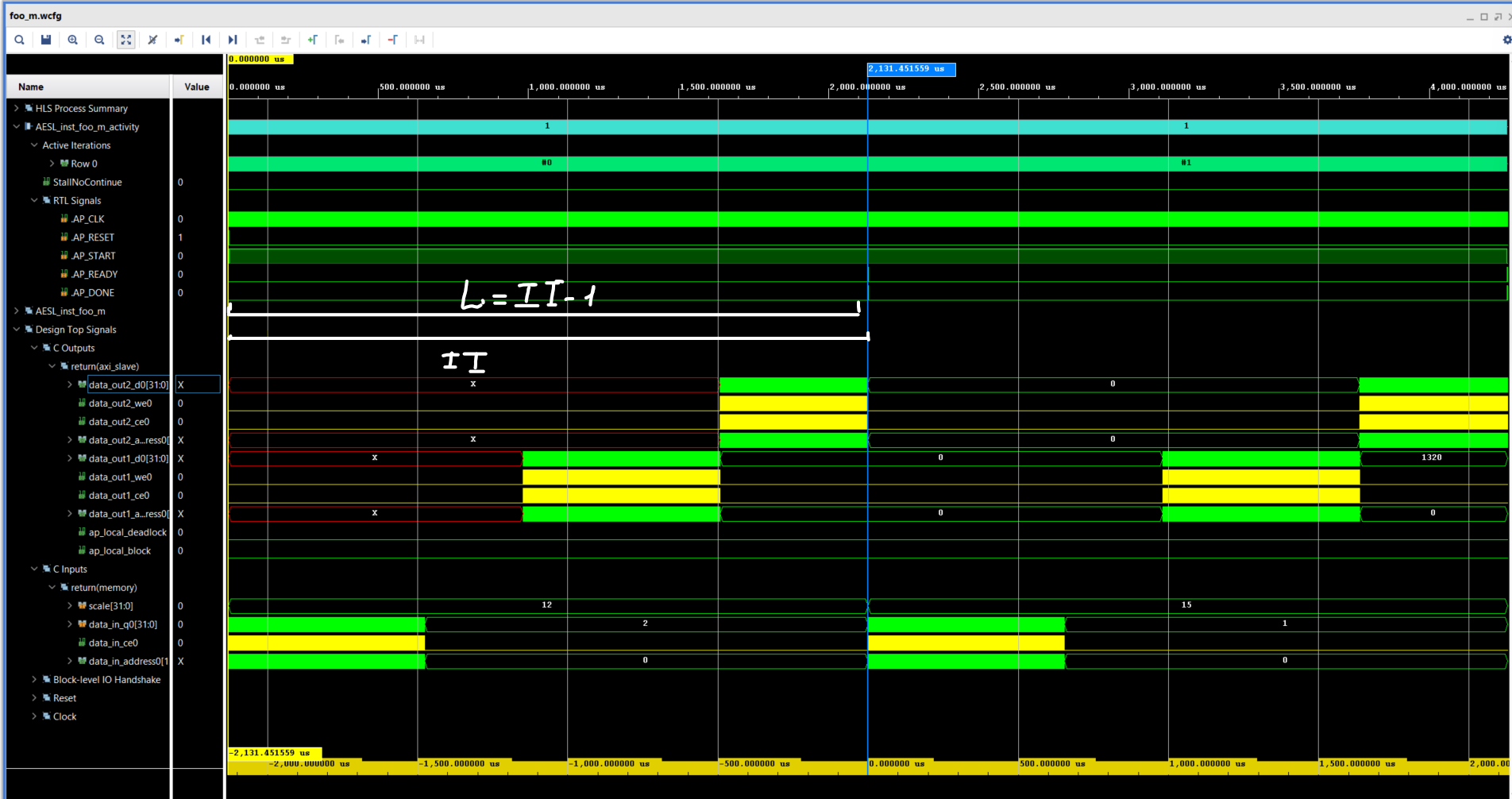


Рисунок Wave viewer

# Синтез Dataflow

Создадим ещё два решения.

* Решение sol4
  + осуществить синтез с DATAFLOW и параметром **FIFO для default\_channel**:
  + Осуществить C|RTL моделирование
* Решение sol5
  + осуществить синтез с DATAFLOW и параметром **ping-pong для default\_channel** :
  + Осуществить C|RTL моделирование

Для этого создадим новый командный файл.

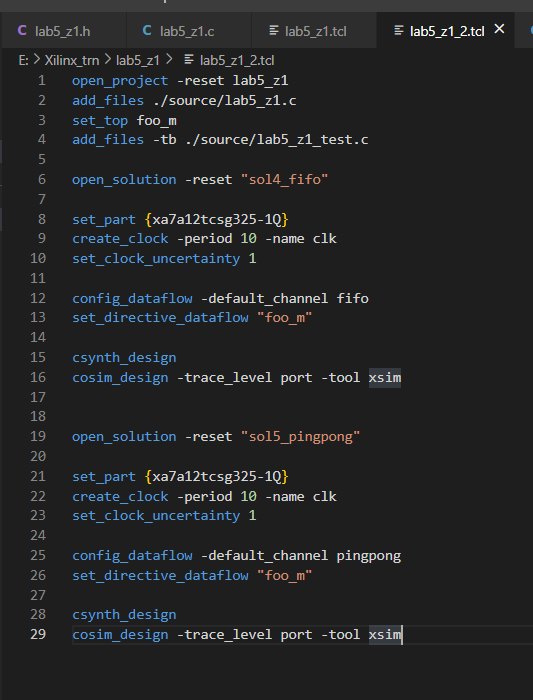


Рисунок Командный файл для параметризации Dataflow

## Решение 4

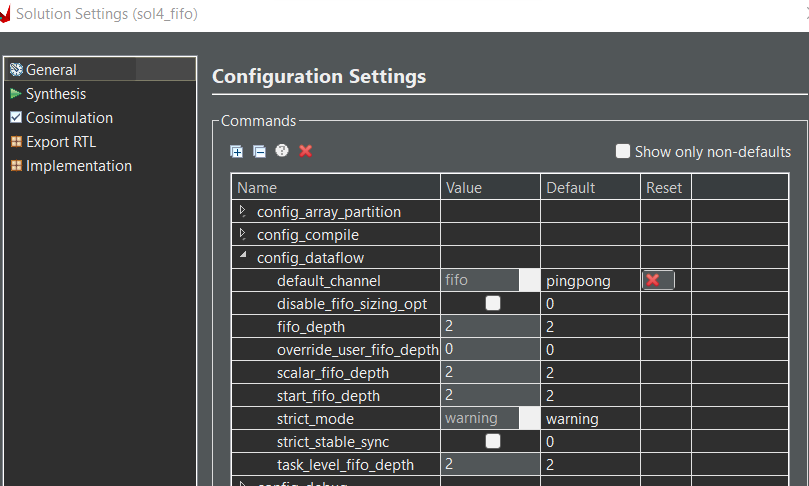


Рисунок Просмотр правильности конфигурации dataflow

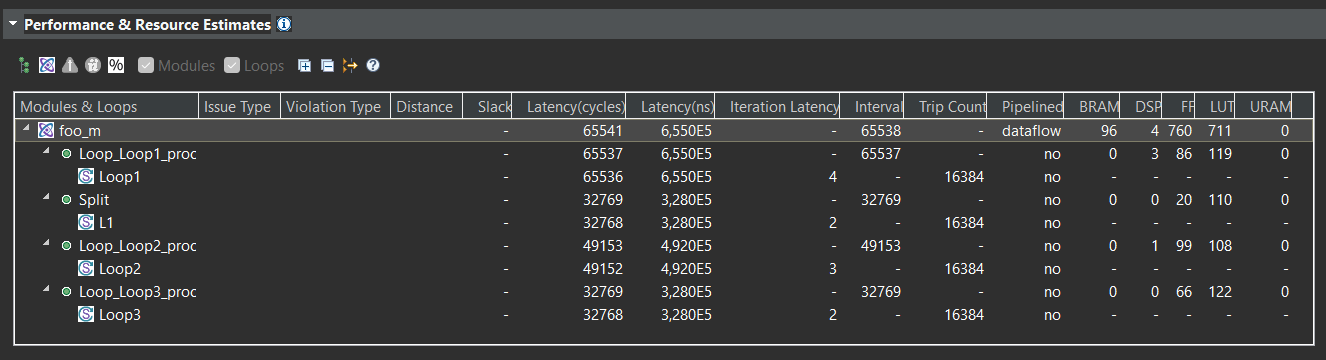


Рисунок 16 Performance estimates

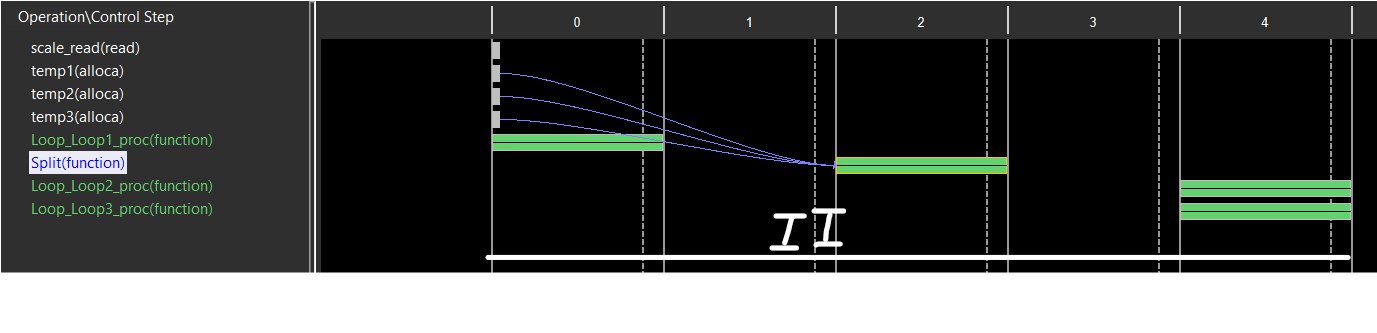


Рисунок 17 Scheduler viewer

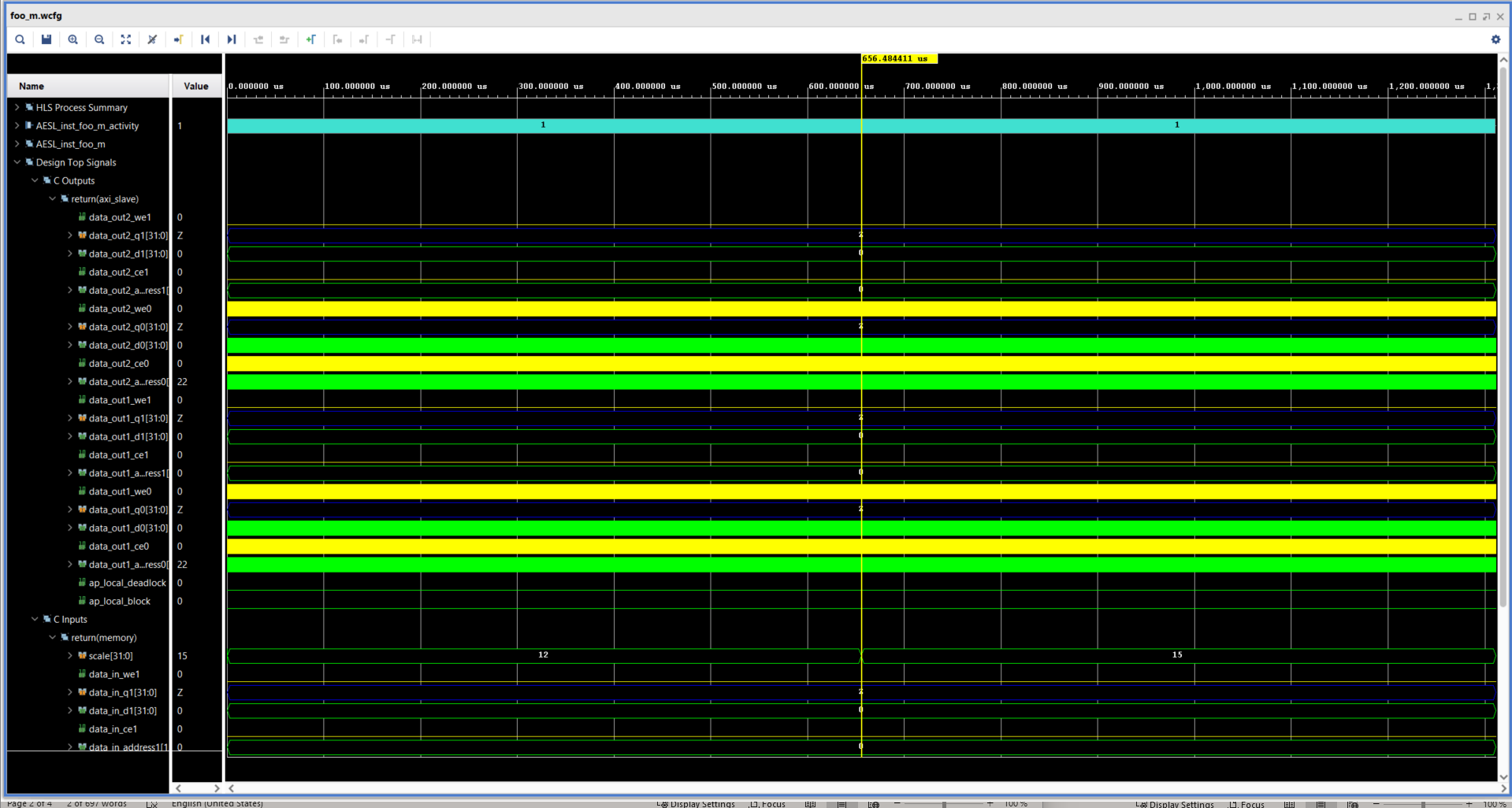
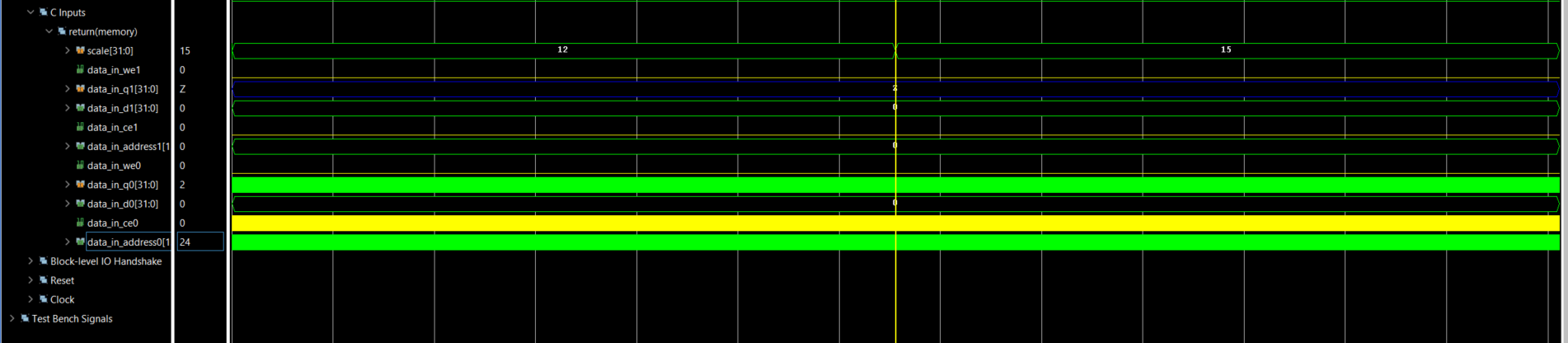
 

Рисунок 18 Wave viewer

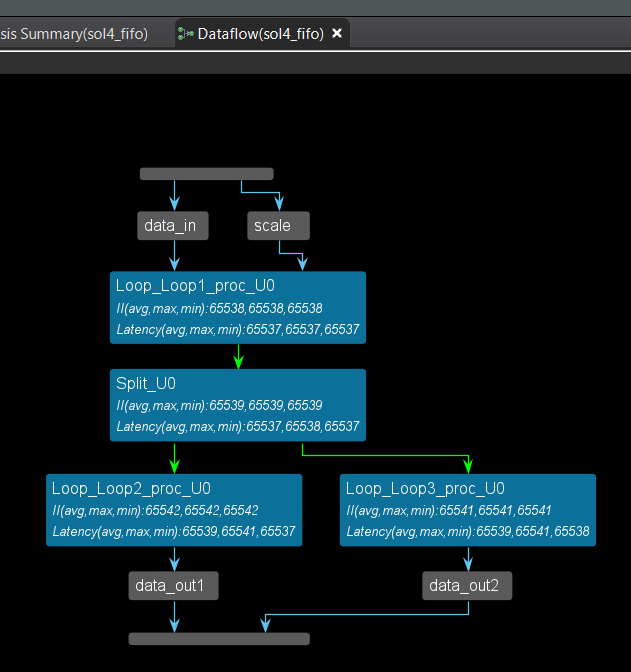


Рисунок Dataflow viewer

* 1. Решение 5

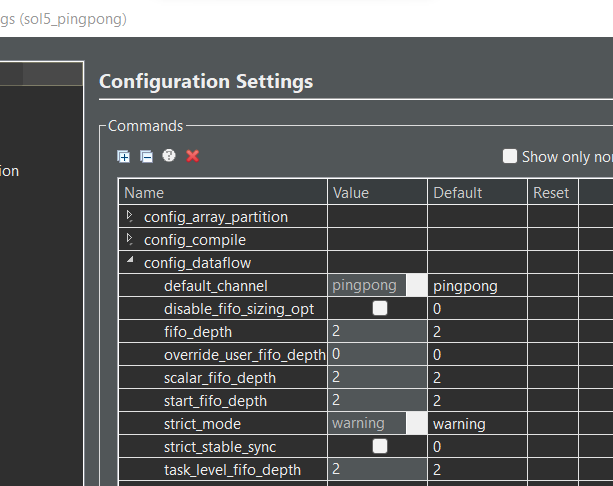


Рисунок Просмотр правильности конфигурации dataflow

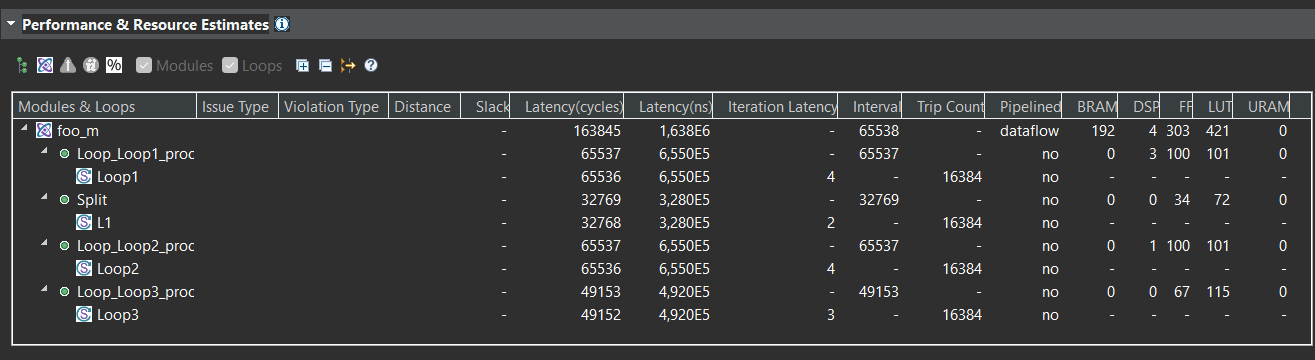


Рисунок 21 Performance estimates

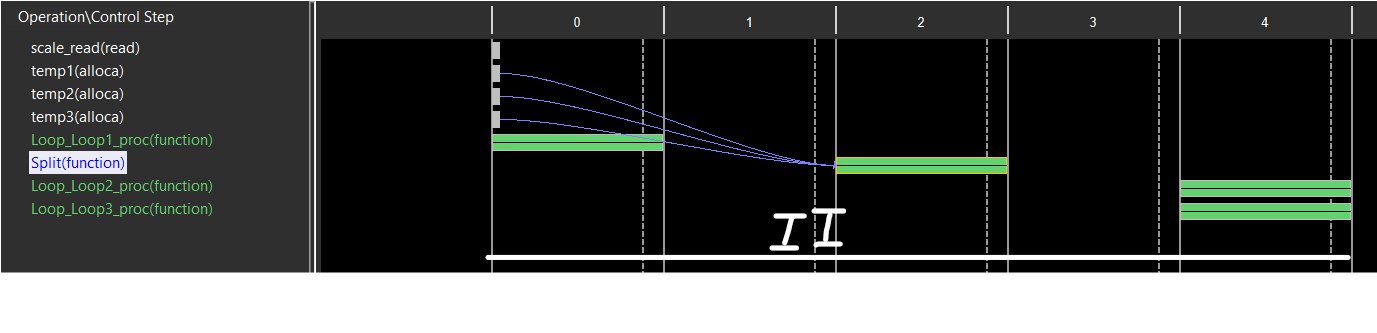


Рисунок Scheduler viewer

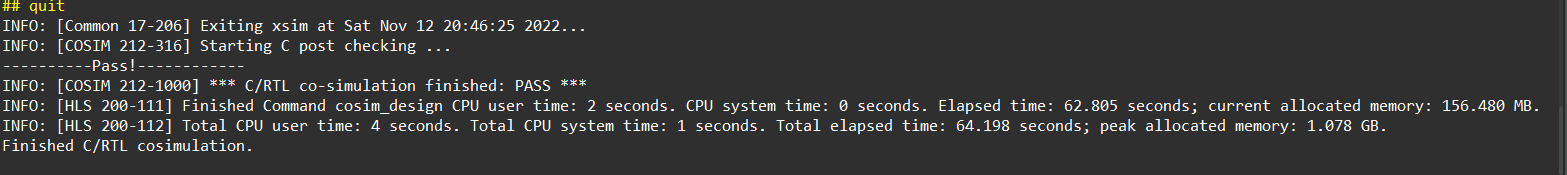


Рисунок Вывод в консоли результата операции C|RTL моделирования

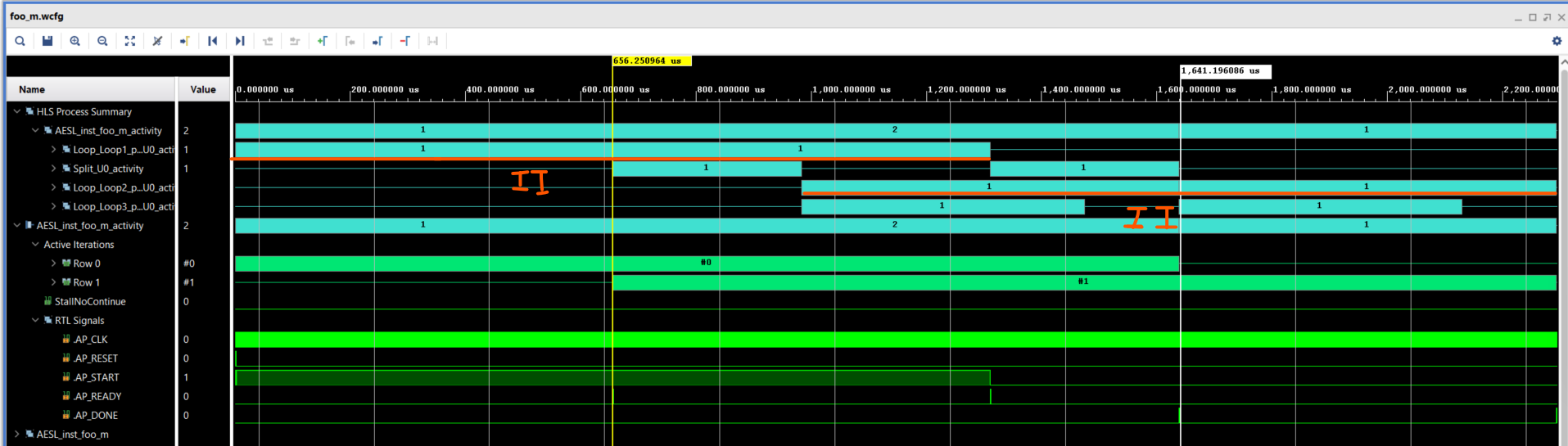
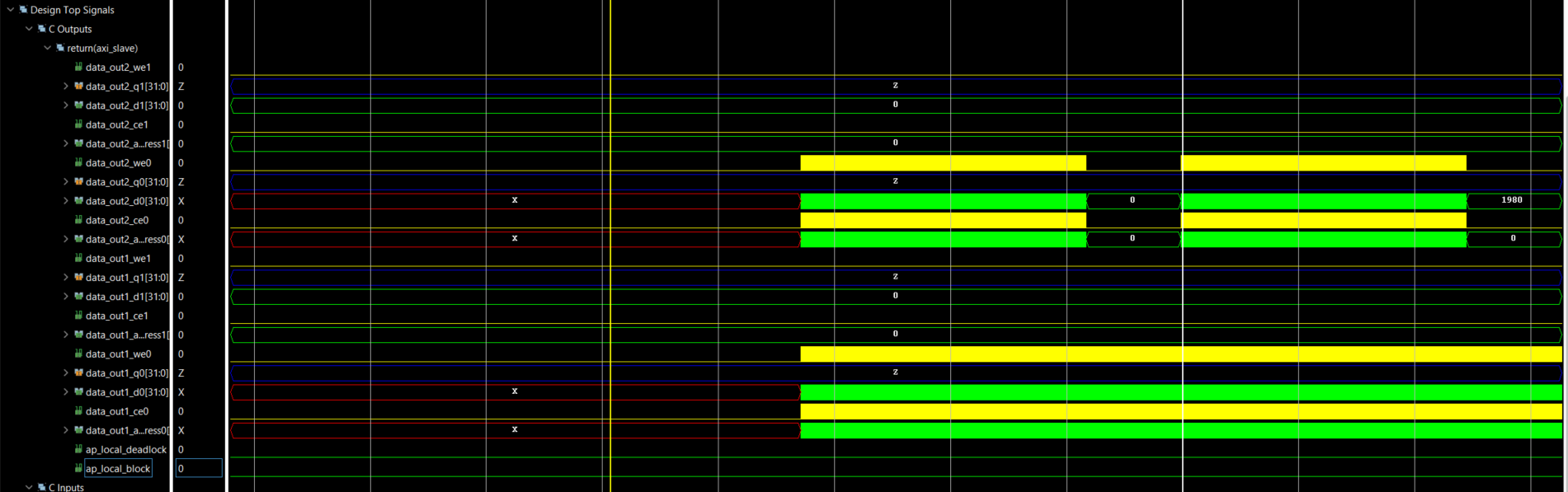
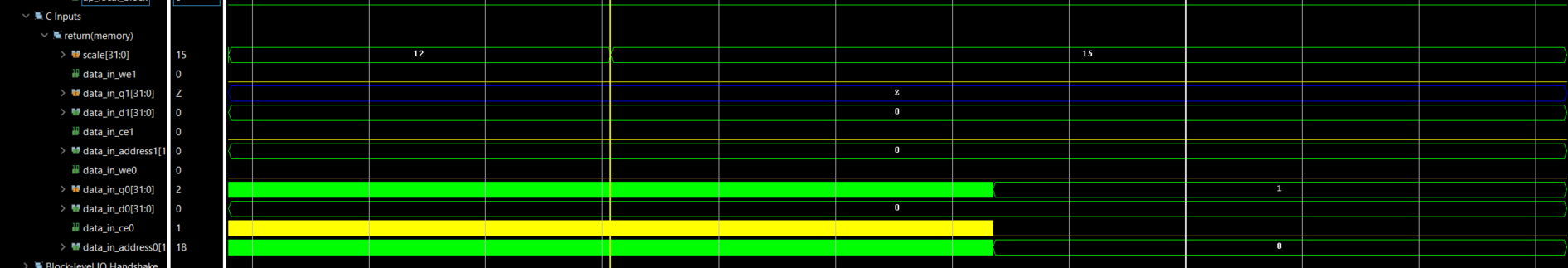
  
 

Рисунок Wave viewer

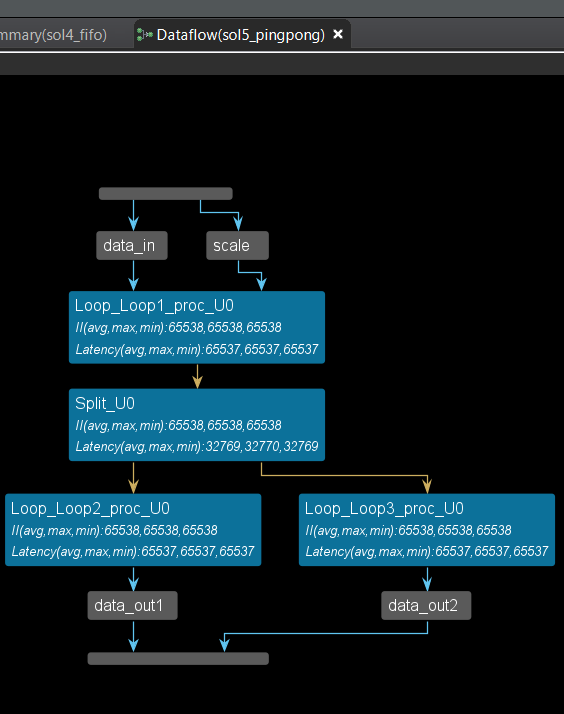


Рисунок Dataflow viewer

## Сравним два решения

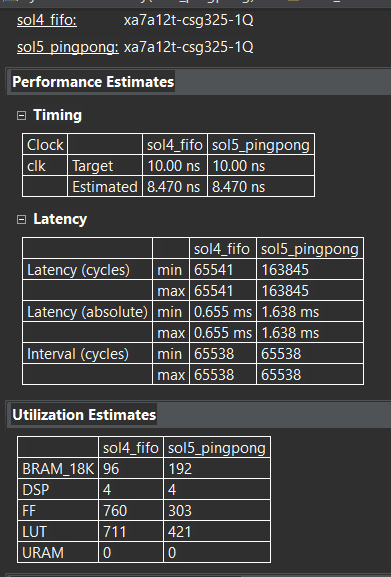


Рисунок Сравнение решени Ping-pong и FIFO

## Исследование времени выполнения на ПК

На базе использованного выше Си теста был создан отдельный, модернизированный, тест для проверки времени выполнения синтезируемой функции на ПК:

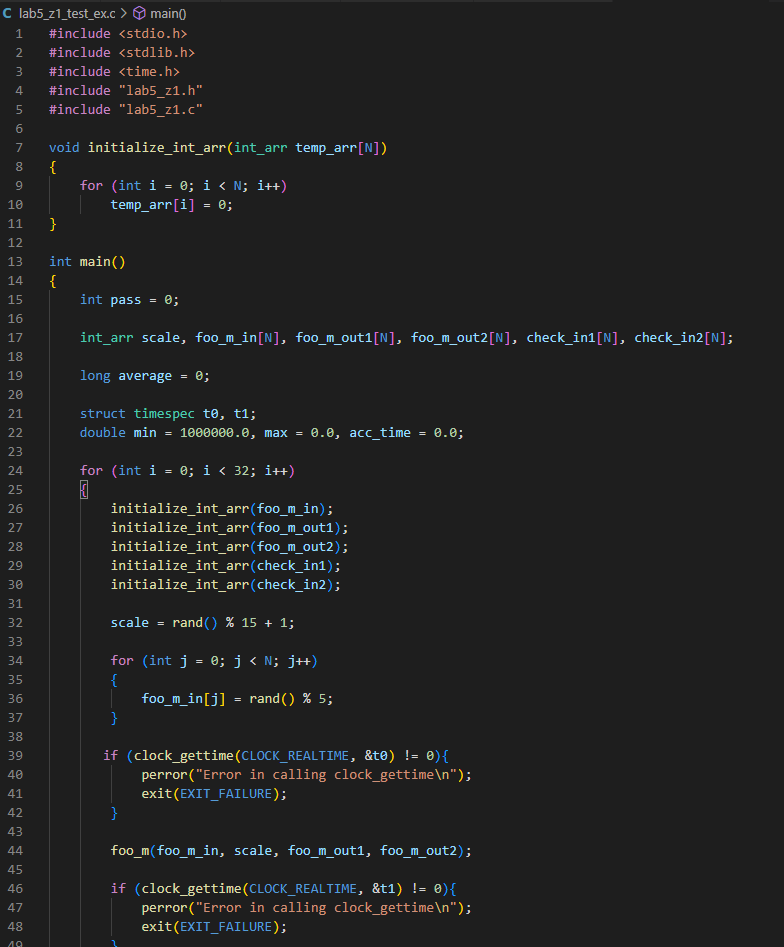


Рисунок Тест для измерения времени

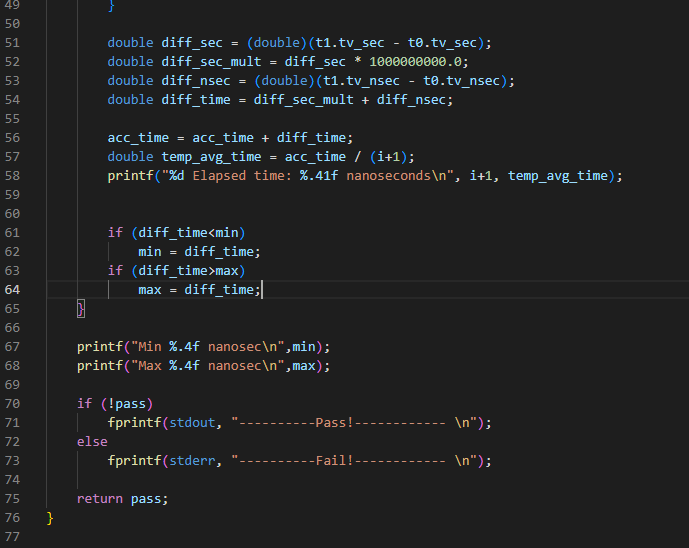


Рисунок Тест для измерения времени

В тесте:

* добавлены операторы измерения времени выполнения синтезируемой функции.
* увеличено количество запусков синтезируемой функции до 32. Для каждого запуска измерено время, найдено среднее значение, которое было выведено в качестве результата.
* Точность измерения времени (наносекунды).

Характеристики ПК, на котором производилось тестирование.

|  |  |  |  |
| --- | --- | --- | --- |
| Тип процессора | Базовая частота | Максимальная частота | Объем ОЗУ |
| 11th Gen Intel(R) Core(TM) i5-1155G7 | 2.5 ГГц | 3.2 ГГц | 16 ГБ |

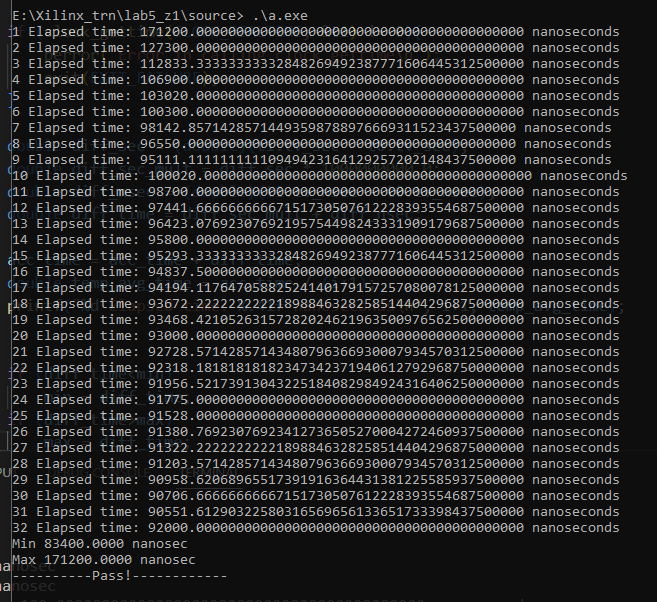


Рисунок Результат измерения времени выполнения функции

Для наглядности приведена сводная таблица

|  |  |  |  |
| --- | --- | --- | --- |
|  | Min (ns) | Max (ns) | Average (ns) |
| N = 16384 | 83400 | 171200 | 92000 |

Посчитаем время выполнения на аппаратной реализации:

|  |  |  |  |
| --- | --- | --- | --- |
|  | II | Estimated | Time (ns) |
| Sol2 | 212997 | 8,470 | 1804084.59 |
| Sol4(fifo) | 65538 | 8,470 | 555106.86 |
| Sol5(ping-pong) | 65538 | 8,470 | 555106.86 |

# Выводы

В ходе выполнения лабораторной работы, было выполнено исследование и сравнительный анализ времени выполнения функции lab5\_z1. Для определения оптимального clock period были созданы 3 решения и выбрано оптимальное – второе с clock period = 10.

Затем для оптимального решения были заданы Dataflow FIFO и Ping-Pong. FIFO лучше всего использовать тогда, когда мы последовательно обращаемся к адресам, в противном случае следует использовать Ping-pong. По сравнительному анализу можно сделать вывод, что в данном случае оптимальнее использовать FIFO – меньше аппаратных затрат и меньше Latency, по времени выполнялись одинаково.

Также функция была запущена на ПК. Результат запуска был сравнен с результатом запуска на аппаратной реализации. Получилось, что запуск на ПК выполняется быстрее.