Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и технологии

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ Lab5\_z1

Дисциплина: Проектирование реконфигурируемых гибридных вычислительных систем

Выполнил студент Курякин Д.

Гр. 3540901/12001

Руководитель, доцент Антонов А.П.

«\_\_» \_\_\_\_\_\_ 2022

Санкт-Петербург

2022

Оглавление

[1. Задание 3](#_Toc116916474)

[2. Исходный код функции 3](#_Toc116916475)

[3. Исходный код теста 3](#_Toc116916476)

[3.1 Моделирование 4](#_Toc116916477)

[4. Первое решение — Solution1 5](#_Toc116916478)

[4.1 Исходные настройки 5](#_Toc116916479)

[4.2 Синтез 5](#_Toc116916480)

[5. Первое решение — Solution2 7](#_Toc116916481)

[5.1 Исходные настройки 7](#_Toc116916482)

[5.2 Синтез 8](#_Toc116916483)

[6. Сравнение Solution1 и Solution2 9](#_Toc116916484)

[7. Выводы 10](#_Toc116916485)

1. Задание

* Создать тест lab5\_z1\_test.c для проверки функции (тест с самопроверкой, входные данные – случайные числа).
* Тип данных (для всех аргументов он – int), значение N (***N = 16384***) следует задать в файле lab5\_z1.h
* Для функции **foo\_m**
  + задать: clock period 6, 10, 14; clock\_uncertainty 1
  + осуществить моделирование
  + осуществить синтез и выбрать лучший вариант (максимум производительности при наименьших аппаратных затратах), привести результаты сравнения вариантов + таблица.
    - Для выбранного варианта привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнить Zoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
  + **Написать tcl файл автоматизирующий исследование (решения sol1, sol2, sol3)**

для clock period выбранного на предыдущем этапе:

Решение sol4

* + осуществить синтез с DATAFLOW и параметром **FIFO для default\_channel**:
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнить Zoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * **Dataflow viewer**
  + Осуществить C|RTL моделирование
    - 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

Решение sol5

* + осуществить синтез с DATAFLOW и параметром **ping-pong для default\_channel** :
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнить Zoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * **Dataflow viewer**
  + Осуществить C|RTL моделирование
    - 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 двух вариантов foo\_m между собой

**2. Исходный код функции**

Исходный код заголовочного файла приведен ниже.

|  |
| --- |
| **typedef** **int** **int\_arr**;  **#define** **N** 16384  **void** **split**(**int\_arr** in[N], **int\_arr** out1[N], **int\_arr** out2[N]); |

Исходный код синтезируемой функции приведен ниже.

|  |
| --- |
| **#include** "lab5\_z1.h"  **#include** <stdio.h>  **void** **split**(**int\_arr** in[N], **int\_arr** out1[N], **int\_arr** out2[N])  {  L1:  **for** (**int** i = 0; i < N; i++)  {  **#pragma** HLS pipeline off  out1[i] = in[i];  out2[i] = in[i];  }  }  **void** **foo\_m**(**int** data\_in[N], **int** scale, **int** data\_out1[N], **int** data\_out2[N])  {  **int** temp1[N], temp2[N], temp3[N];  Loop1:  **for** (**int** i = 0; i < N; i++)  {  **#pragma** HLS pipeline off  temp1[i] = data\_in[i] \* scale;  }  **split**(temp1, temp2, temp3);  Loop2:  **for** (**int** j = 0; j < N; j++)  {  **#pragma** HLS pipeline off  data\_out1[j] = temp2[j] \* 22;  }  Loop3:  **for** (**int** k = 0; k < N; k++)  {  **#pragma** HLS pipeline off  data\_out2[k] = temp3[k] \* 33;  }  } |

**3. Исходный код теста**

Исходный код теста для проверки функции lab4\_z1 приведен ниже.

|  |
| --- |
| **#include** <stdio.h>  **#include** "lab5\_z1.h"  **void** **initialize\_int\_arr**(**int\_arr** temp\_arr[N])  {  **for** (**int** i = 0; i < N; i++)  temp\_arr[i] = 0;  }  **void** **check\_func**(**int** check\_in1[N], **int** check\_in2[N], **int** scale, **int** foo\_m\_out1[N], **int** foo\_m\_out2[N])  {  **for** (**int** j = 0; j < N; j++)  {  **#pragma** HLS pipeline off  check\_in1[j] = foo\_m\_out1[j] / 22 / scale;  }  **for** (**int** k = 0; k < N; k++)  {  **#pragma** HLS pipeline off  check\_in2[k] = foo\_m\_out2[k] / 33 / scale;  }  }  **int** **arentEqual**(**int\_arr** foo\_m\_in[N], **int\_arr** check\_in1[N], **int\_arr** check\_in2[N])  {  **for** (**int** i = 0; i < N; i++)  {  **if** (foo\_m\_in[i] != check\_in1[i] || check\_in1[i] != check\_in2[i])  {  **fprintf**(stdout, " ERROR: actual in1 = %d actual in2 = %d; expected in = %d\n",  check\_in1[i], check\_in2[i], foo\_m\_in[i]);  **return** 1;  }  }  **return** 0;  }  **int** **main**()  {  **int** pass = 0;  **int\_arr** scale, foo\_m\_in[N], foo\_m\_out1[N], foo\_m\_out2[N], check\_in1[N], check\_in2[N];  **for** (**int** i = 0; i < 2; i++)  {  **initialize\_int\_arr**(foo\_m\_in);  **initialize\_int\_arr**(foo\_m\_out1);  **initialize\_int\_arr**(foo\_m\_out2);  **initialize\_int\_arr**(check\_in1);  **initialize\_int\_arr**(check\_in2);  scale = **rand**() % 15 + 1;  **for** (**int** j = 0; j < N; j++)  {  foo\_m\_in[j] = **rand**() % 5;  }  **foo\_m**(foo\_m\_in, scale, foo\_m\_out1, foo\_m\_out2);  **check\_func**(check\_in1, check\_in2, scale, foo\_m\_out1, foo\_m\_out2);  }  **if** (!pass)  **fprintf**(stdout, "----------Pass!------------ \n");  **else**  **fprintf**(stderr, "----------Fail!------------ \n");  **return** pass;  } |

Тест обеспечивает проверку корректной работы функции.

В листинге приведен tcl скрипт.

|  |
| --- |
| open\_project -reset lab5\_z1  add\_files ./source/lab5\_z1.c  set\_top foo\_m  add\_files -tb ./source/lab5\_z1\_test.c  set all\_solutions {sol1 sol2 sol3}  set all\_period {{6} {10} {14}}  foreach the\_solution $all\_solutions the\_period $all\_period {  open\_solution -reset $the\_solution    create\_clock -period $the\_period -name clk  set\_clock\_uncertainty 1    set\_part {xa7a12tcsg325-1Q}  if {$the\_solution == "sol1"} {  csim\_design -clean  }    csynth\_design  cosim\_design -trace\_level port -tool xsim  }  open\_solution -reset "sol4\_fifo"  set\_part {xa7a12tcsg325-1Q}  create\_clock -period 10 -name clk  set\_clock\_uncertainty 1  config\_dataflow -default\_channel fifo  set\_directive\_dataflow "foo\_m"  csynth\_design  cosim\_design -trace\_level port -tool xsim  open\_solution -reset "sol5\_pingpong"  set\_part {xa7a12tcsg325-1Q}  create\_clock -period 10 -name clk  set\_clock\_uncertainty 1  config\_dataflow -default\_channel pingpong  set\_directive\_dataflow "foo\_m"  csynth\_design  cosim\_design -trace\_level port -tool xsim |

**6. Сравнение решений**

Были созданы три решения. На таблице 1 и рис. 1 приведены с аппаратные затраты и время работы.

Таблица 1. Три решения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | sol1 | sol2 | sol3 |
| Clock | Target ( ns) | 6 | 10 | 14 |
| Estimated ( ns) | 3,895 | 8,470 | 11,727 |
| Iteration Interval | (cycles) | 376836 | 212996 | 163844 |
| (ns) | 1844612 | 2017072 | 2085243 |
| Resources | BRAM\_18K | 96 | 96 | 96 |
| DSP48E | 4 | 4 | 4 |
| FF | 798 | 294 | 195 |
| LUT | 379 | 357 | 345 |
| URAM | 0 | 0 | 0 |

Рис. 1. Три решения

Исходя из таблицы наиболее оптимальным является решение 2 (sol2). На рис. 2 приведен scheduler viewer второго решения.

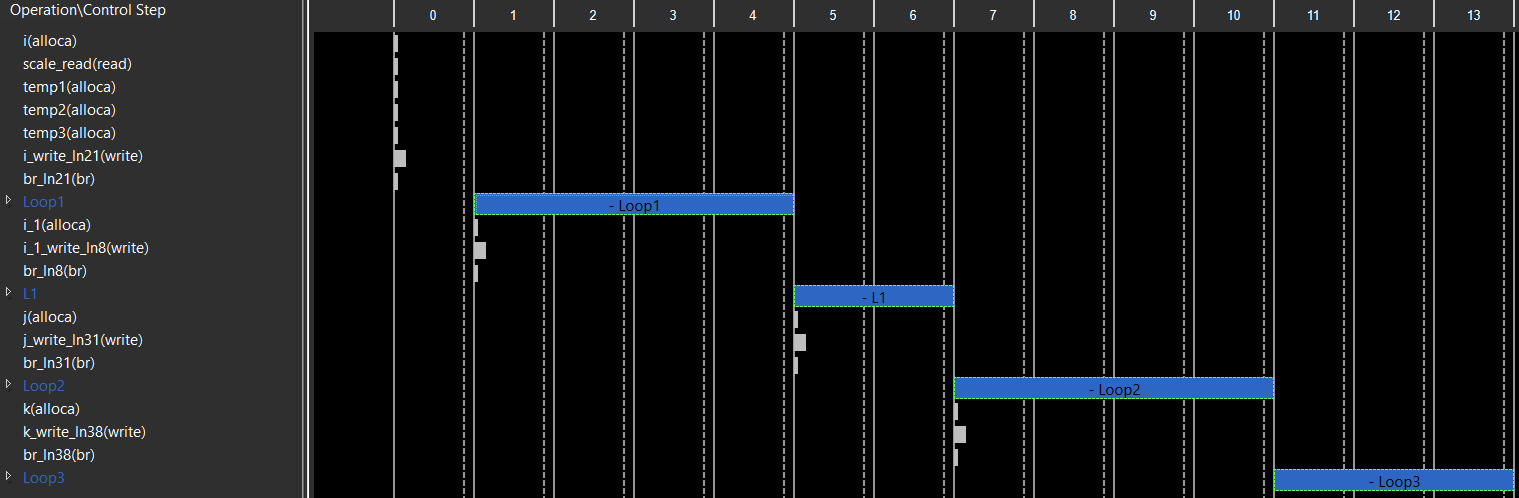


Рис. 2. Scheduler viewer второго решения

На рис. 3 приведен wave viewer для второго решения.



Рис. 3. Wave viewer для второго решения

Осуществим синтез с DATAFLOW параметром FIFO и ping-pong с target 10. На таблице 2 и рис. 4 приведены результаты.

Таблица 2. Пять решений

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | sol1 | sol2 | sol3 | sol4\_fifo | sol5\_pingpong |
| Clock | Target ( ns) | 6 | 10 | 14 | 10 | 10 |
| Estimated ( ns) | 3,895 | 8,470 | 11,727 | 8,470 | 8,470 |
| Iteration Interval | (cycles) | 376836 | 212996 | 163844 | 65541 | 163845 |
| (ns) | 1844612 | 2017072 | 2085243 | 620673 | 1551612 |
| Resources | BRAM\_18K | 96 | 96 | 96 | 96 | 96 |
| DSP48E | 4 | 4 | 4 | 4 | 4 |
| FF | 798 | 294 | 195 | 760 | 303 |
| LUT | 379 | 357 | 345 | 711 | 421 |
| URAM | 0 | 0 | 0 | 0 | 0 |

Рис. 4. Пять решений

Также конфигурация, performance estimates, scheduler viewer, wave viewer, Dataflow viewer для решения 4 приведены на рис. 5-9, а для решения 5 на рис 10-14.

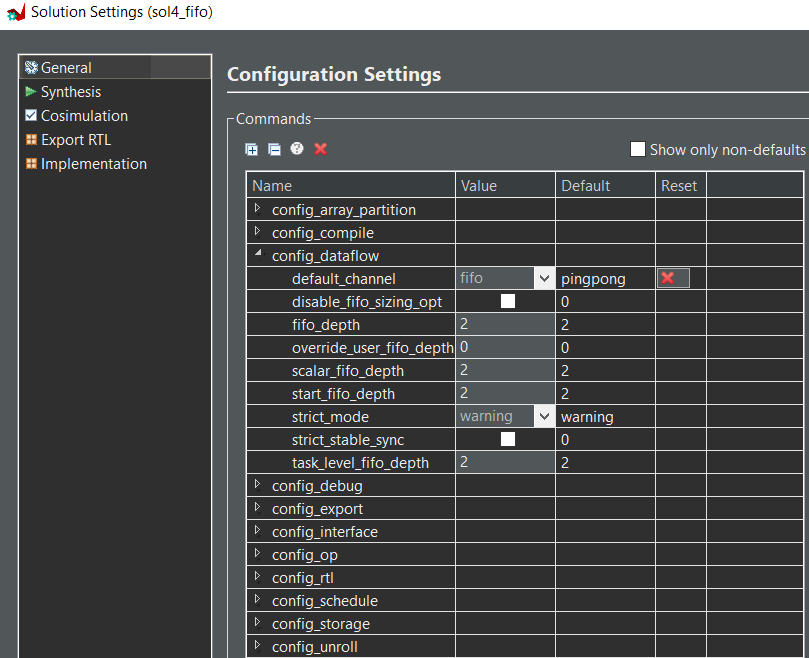


Рис. 5. Конфигурация решения 4

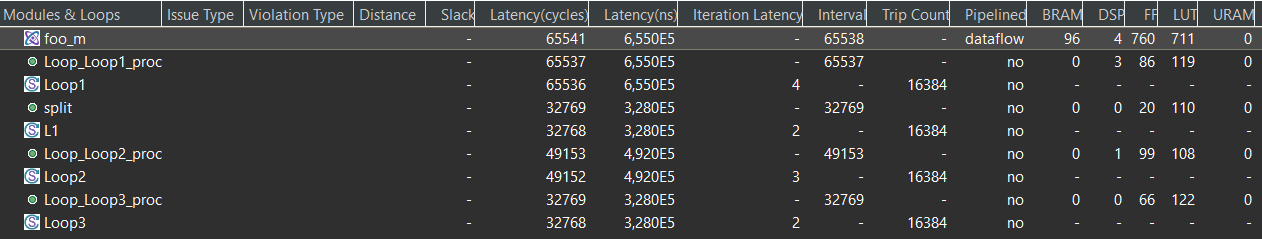


Рис. 6. Performance estimates. Решение 4

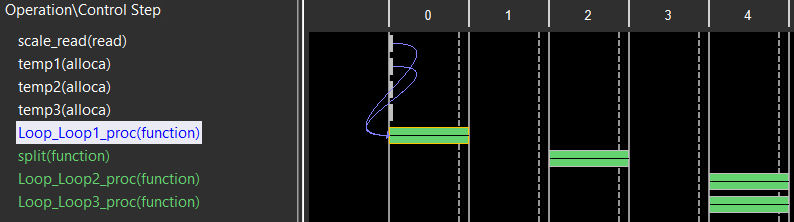


Рис. 7. Scheduler viewer. Решение 4



Рис. 8. Wave viewer. Решение 4

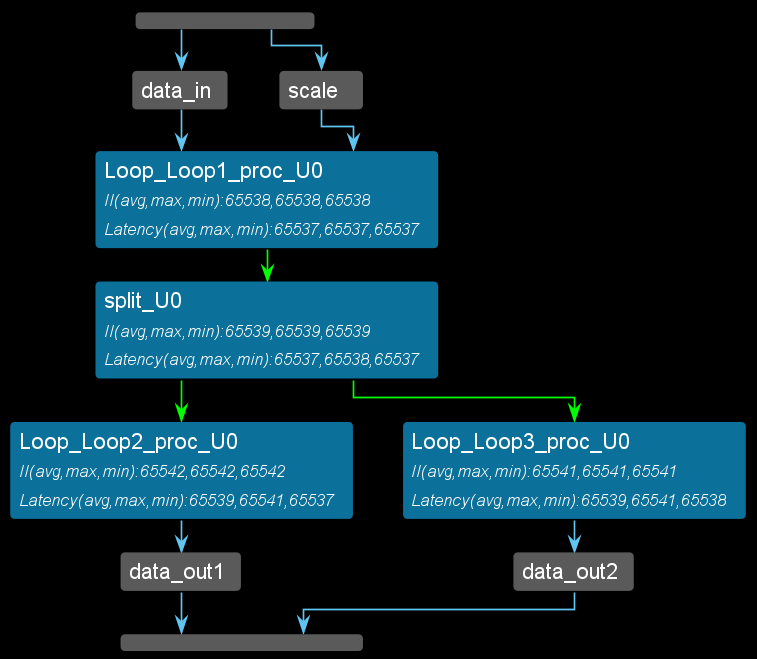


Рис. 9. Dataflow viewer. Решение 4

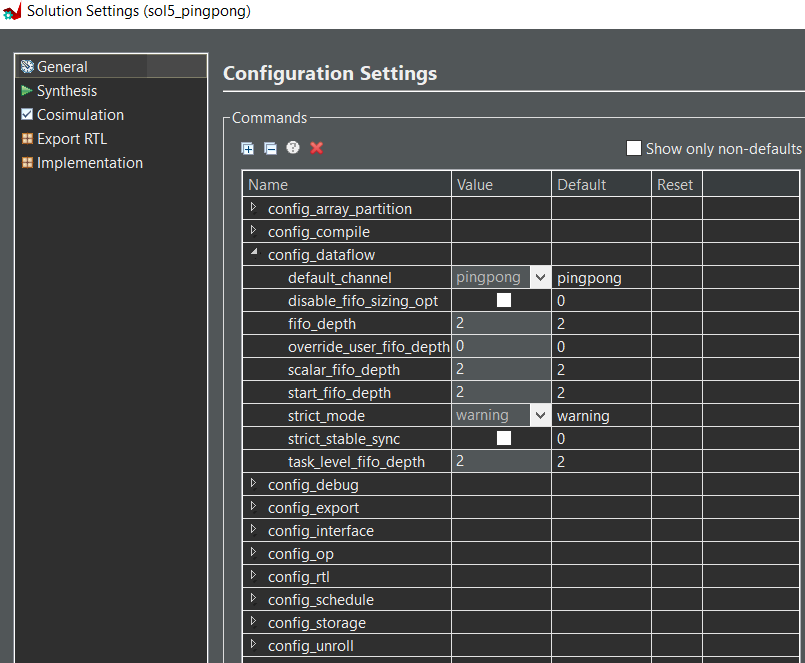


Рис. 10. Конфигурация решения 5

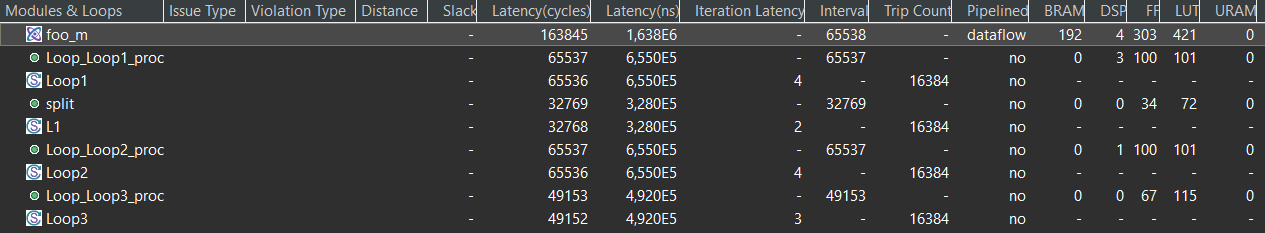


Рис. 11. Performance estimates. Решение 5

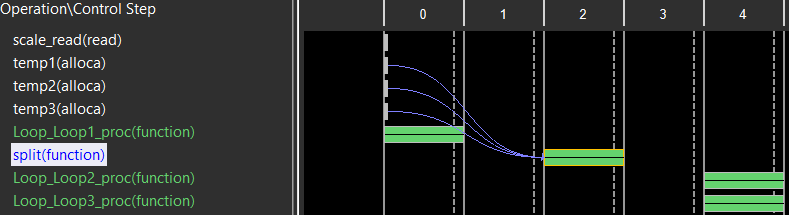


Рис. 12. Scheduler viewer. Решение 5

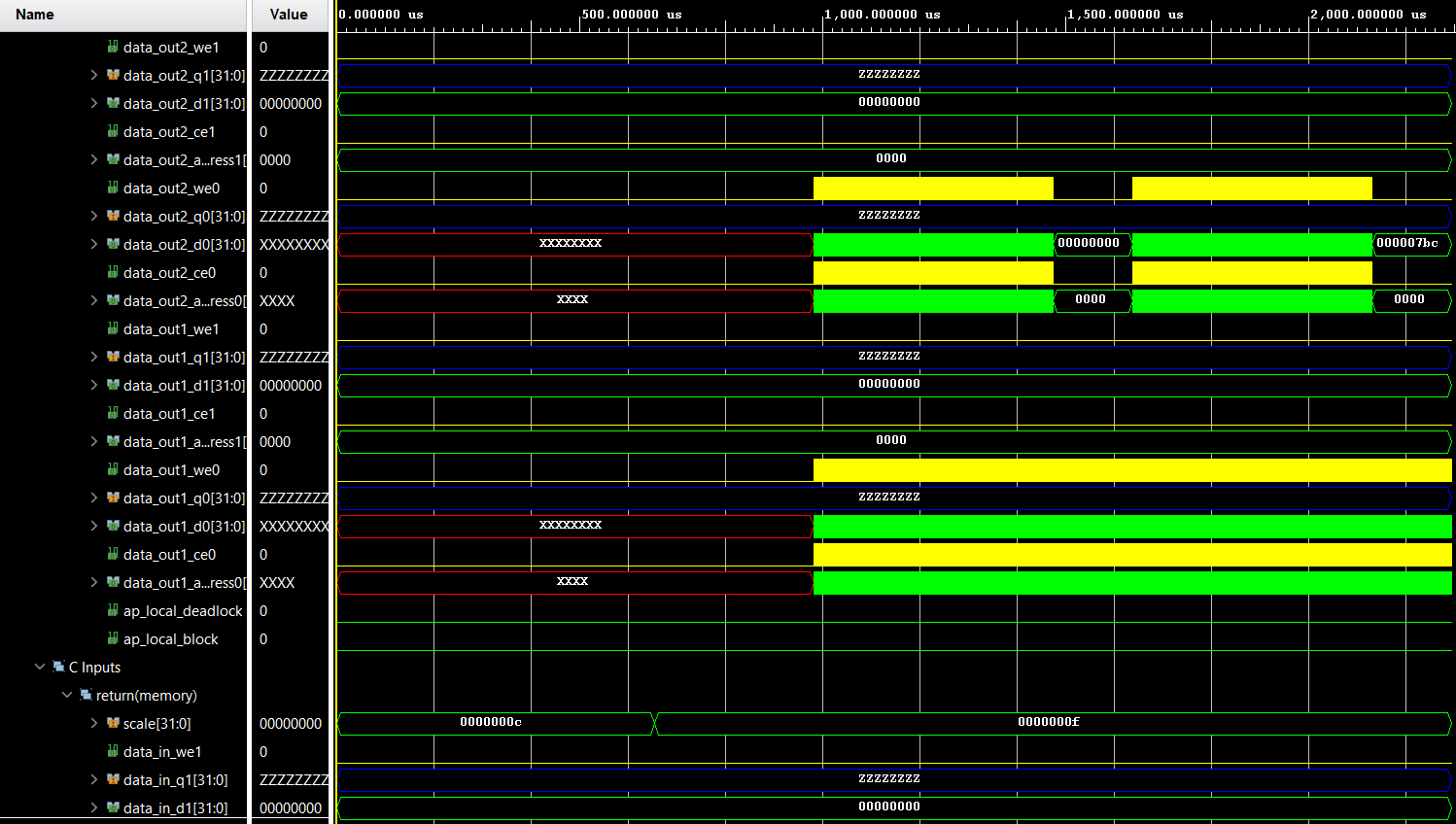


Рис. 13. Wave viewer. Решение 5

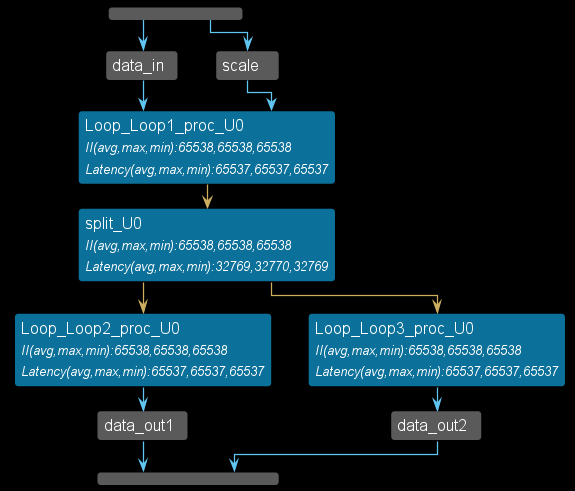


Рис. 14. Dataflow viewer. Решение 5

Исходя из результатов можно принять решение о выборе решения 5 (sol5).

**7. Исследование времени выполнения на ПК**

Было проверено время работы исследуемой программы. На листинге предствлен код программы .

Листинг 4. lab5\_z1.c измерение времени выполнения.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #include "lab5\_z1.h"  #include "lab5\_z1.c"  void initialize\_int\_arr(int\_arr temp\_arr[N])  {  for (int i = 0; i < N; i++)  temp\_arr[i] = 0;  }  int main()  {  int pass = 0;  int\_arr scale, foo\_m\_in[N], foo\_m\_out1[N], foo\_m\_out2[N], check\_in1[N], check\_in2[N];  long average = 0;  struct timespec t0, t1;  double min = 1000000.0, max = 0.0, acc\_time = 0.0;  for (int i = 0; i < 32; i++)  {  initialize\_int\_arr(foo\_m\_in);  initialize\_int\_arr(foo\_m\_out1);  initialize\_int\_arr(foo\_m\_out2);  initialize\_int\_arr(check\_in1);  initialize\_int\_arr(check\_in2);  scale = rand() % 15 + 1;  for (int j = 0; j < N; j++)  {  foo\_m\_in[j] = rand() % 5;  }  if (clock\_gettime(CLOCK\_REALTIME, &t0) != 0){  perror("Error in calling clock\_gettime\n");  exit(EXIT\_FAILURE);  }  foo\_m(foo\_m\_in, scale, foo\_m\_out1, foo\_m\_out2);  if (clock\_gettime(CLOCK\_REALTIME, &t1) != 0){  perror("Error in calling clock\_gettime\n");  exit(EXIT\_FAILURE);  }  double diff\_sec = (double)(t1.tv\_sec - t0.tv\_sec);  double diff\_sec\_mult = diff\_sec \* 1000000000.0;  double diff\_nsec = (double)(t1.tv\_nsec - t0.tv\_nsec);  double diff\_time = diff\_sec\_mult + diff\_nsec;  acc\_time = acc\_time + diff\_time;  double temp\_avg\_time = acc\_time / (i+1);  printf("%d Elapsed time: %.0f nanoseconds\n", i+1, temp\_avg\_time);    if (diff\_time<min)  min = diff\_time;  if (diff\_time>max)  max = diff\_time;  }  printf("Min %.4f nanosec\n",min);  printf("Max %.4f nanosec\n",max);  if (!pass)  fprintf(stdout, "----------Pass!------------ \n");  else  fprintf(stderr, "----------Fail!------------ \n");  return pass;  } |

Таблица 3. Характеристики ПК.

|  |  |  |  |
| --- | --- | --- | --- |
| Тип процессора | Базовая частота | Максимальная частота | Объем ОЗУ |
| 11th Gen Intel(R) Core(TM) i7-11800H | 2.3 ГГц | 4.5 ГГцд | 32 ГБ |

На рис 15 представлен фрагмент вывода программы.

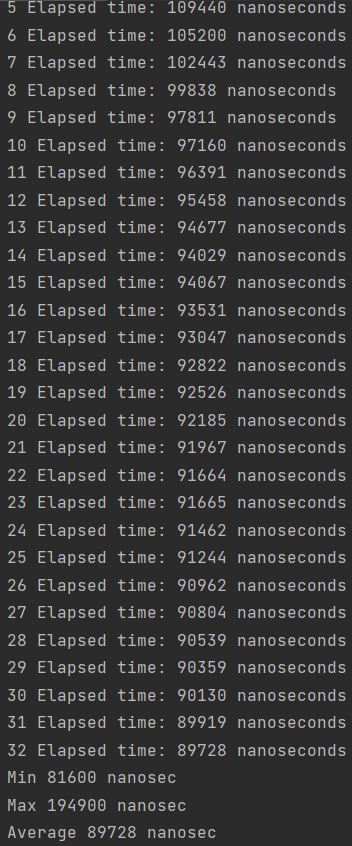


Рис. 15. Фрагмент вывода программы

Для наглядности приведена сводная таблица

Таблица 4. Время выполнения на ПК.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Min (ns) | Max (ns) | Average (ns) |
| N = 16384 | 81600 | 194900 | 89728 |

Посчитаем время выполнения на аппаратной реализации:

Таблица 5. Время выполнения на ПЛИС.

|  |  |  |  |
| --- | --- | --- | --- |
|  | II | Estimated | Time (ns) |
| Sol2 | 212996 | 8,470 | 2017072 |
| Sol4(fifo) | 65541 | 8,470 | 620673 |
| Sol5(ping-pong) | 6163845 | 8,470 | 1551612 |

**Выводы**

В данной работе были получен опыт работы с vitis hls. Из ходя из таблицы 4 и 5 видно, что время выполнения на плис лучше, чем на пк в 4 решении, но при это м высоки аппаратные затраты. В решении 5 ниже аппаратные затраты и сравнимое с процессором время выполнения. Исходя из анализа выбреем решение 5.