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

Тема: Введение в Optimizing Structure for performance

Выполнил студент гр. 01502 С.С. Гаспарян

Руководитель, доцент Антонов А.П.

«24» ноября 2021

Санкт-Петербург
2021

1. Задание

Текст задания находится в файле «Задание lab5_z1.docx»

2. Исходный код фукнции

Исходный код синтезируемой функции lab5_z1 представлен на рисунке 1.

```
#include "lab5_z1.h"

void lab5_z1(int d_in[N], int d_out[N/4])
{
    for_label0: for(int i = 0; i < N/4; ++i) {
        d_out[i] = d_in[i] + d_in[i + N/4] + d_in[i+N/2] + d_in[i+(3*N)/4];
    }

}
```

Рис. 1 Исходный код функции lab5_z1

Функция принимает два аргумента массива типа int — вычисляет сумму отдельных элементов массивов и записывает результат в выходной массив.

3. Исходный код теста

Исходный код теста проверки функции lab5_z1 приведен на рисунке 2.

Тест обеспечивает проверку корректной работы функции.

4. Командный файл

На рисунке 3 представлен текст команд для автоматизированного создания варианта аппаратной реализаций — solution1 с clock period 20; В котором устанавливает директива «resource -core RAM_1P» для входного массива d_in.

```c
#include <stdio.h>
#include "lab5_z1.h"

int cmp_arr(int d_in[N], int d_cmp[N]) {
    for_label0: for(int i = 0; i < N/4; ++i) {
        int tmp = d_in[i] + d_in[i + N/4] + d_in[i+N/2] + d_in[i+(3*N)/4];
        if (tmp != d_cmp[i])
            return 0;
    }
    return 1;
}

int main () {

    int pass = 0;
    // Create input data
    int d_in[N];
    int d_out[N];

    for (int i = 0; i < 3; ++i){
        for(int j = 0; j < N; j++){
            d_in[j] = rand() % (N - 1) + 1;
        }


        lab5_z1(d_in, d_out);
        pass = cmp_arr(d_in, d_out);
        if (pass == 0) {
            fprintf(stderr, "----------Fail!------------\n");
            return 1;
        }
    }
```

Рис. 2 Исходный код lab5_z1_test.c тестирования функции

```
# Create a project
open_project -reset lab5_z1_prj

# The source file and test bench
add_files        ./source/lab5_z1.c
add_files -tb    ./source/lab5_z1_test.c

# Specify the top-level function for synthesis
set_top          lab5_z1

############################
# Solution settings

# Create solution1
open_solution -reset solution1

# Specify a Xilinx device and clock period
# - Do not specify a clock uncertainty (margin)
# - Let the  margin to default to 12.5% of clock period
set_part  {xa7a12tcsg325-1q}
create_clock -period 20

# Simulate the C code
csim_design

# Directives - RESOURCE (RAM_1P);
set_directive_resource -core RAM_1P "lab5_z1" d_in

# synthesis and c/rtl sim
csynth_design
cosim_design -trace_level all

exit
```

Рис. 3 Текст команд для создания решения

5. Исследование часть 1

5.1 solution1 с директивой «resource -core RAM_1P "lab5_z1" d_in». На рисунке 4 представлена временная диаграмма solution1. На временной диаграмме видны пустые промежутки во время записи данных у считывания данных. В этот момент можно брать уже следующие данные из массива. Для решения данного вопроса нужно просто добавить директиву конвейеризации. Временная диаграмма с добавлением «pipeline» представлена на рисунке 5 для решения solution1_1.

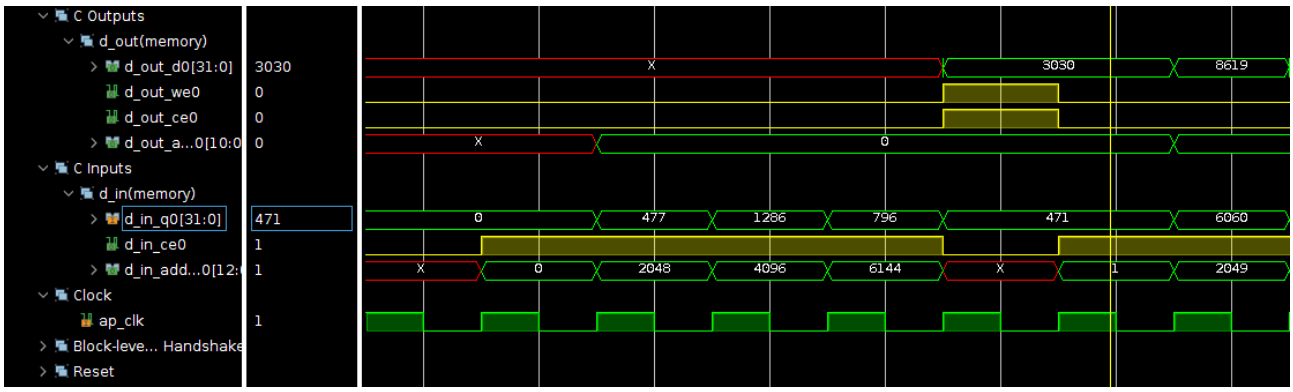

Рис. 4 Временная диаграмма solution1

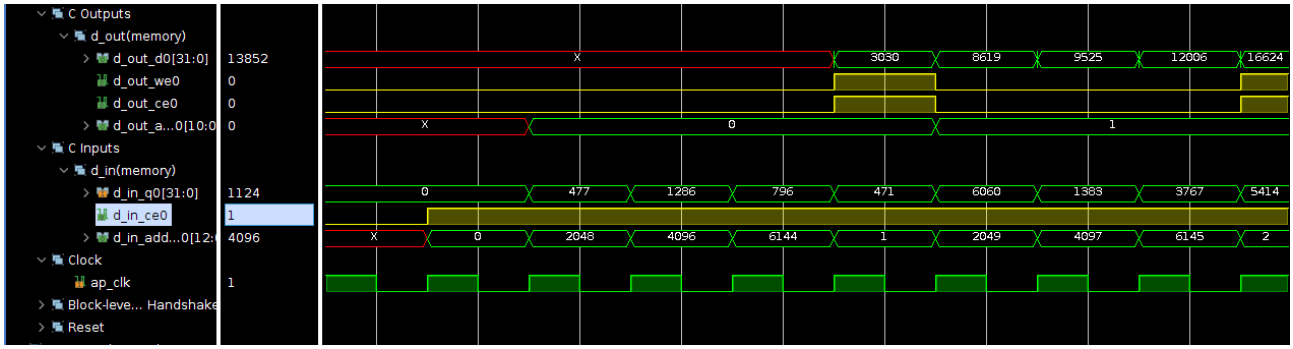

Рис. 5 Временная диаграмма solution1_1

5.2 solution2 с директивой «resource -core RAM_2P "lab5_z1" d_in». На рисунке 6 представлено сравнение временных и аппаратных параметров, как видно разделение входного массива на два порта уменьшает количество тактов до записи в выходной массив. И следовательно уменьшается общее количество тактов. На рисунке 7 представлена временная диаграмма solution2. На временной диаграмме видны пустые промежутки во время записи данных у считывания данных. В этот момент можно брать уже следующие данные из массива. Для решения данного вопроса нужно просто добавить директиву конвейеризации. Временная диаграмма с добавлением «pipeline» представлена на рисунке 8 для решения solution2_1.

Performance Estimates

Timing

Clock		solution1	solution2
ap_clk	Target	20.00 ns	20.00 ns
	Estimated	13.030 ns	13.030 ns

Latency

		solution1	solution2
Latency (cycles)	min	10241	6145
	max	10241	6145
Latency (absolute)	min	0.205 ms	0.123 ms
	max	0.205 ms	0.123 ms
Interval (cycles)	min	10241	6145
	max	10241	6145

Utilization Estimates

	solution1	solution2
BRAM_18K	0	0
DSP48E	0	0
FF	150	116
LUT	222	214
URAM	0	0

Рис. 6 Сравнение решении solution1 и solution2

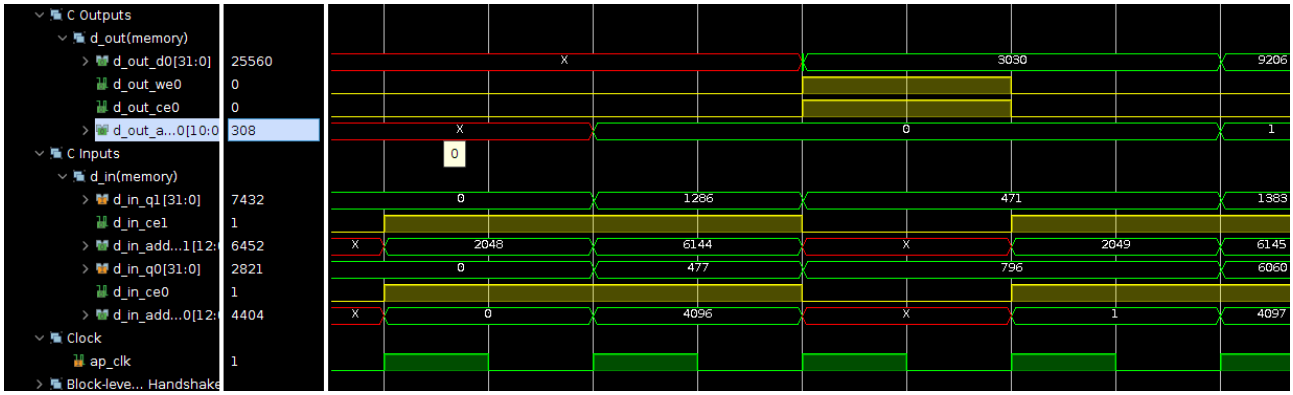

Рис. 7 Временная диаграмма solution2

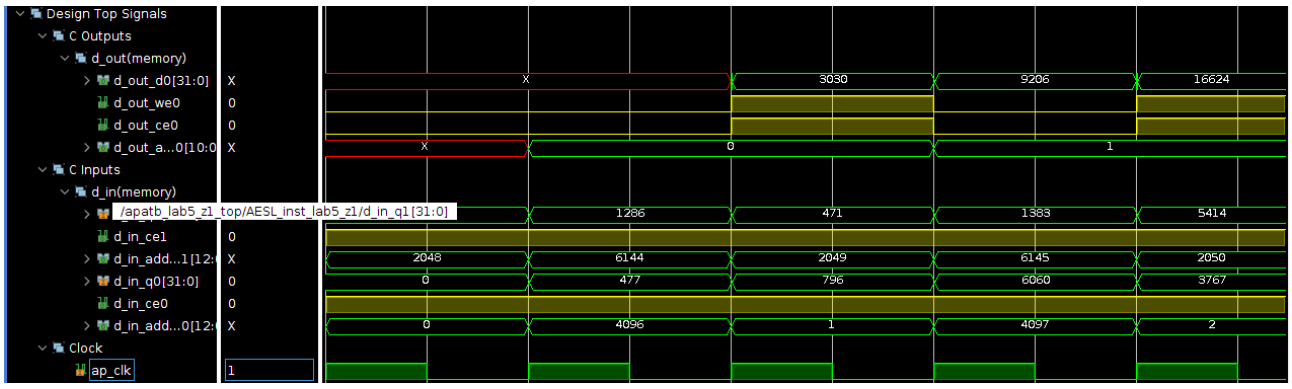

Рис. 8 Временная диаграмма solution2_1

5.3 solution3 с директивой «resource -core RAM_1P "lab5_z1" d_in» и «array_partition -type block -factor 4 "lab5_z1" d_in». На рисунке 8 представлено сравнение временных и аппаратных параметров, как видно разделение входного массива одиночного порта на четыре подмассива уменьшает количество тактов до записи в выходной массив и выходной массив теперь записывается одним тактом в два порта. И следовательно уменьшается общее количество тактов. На рисунке 9 представлена временная диаграмма solution3. На временной диаграмме видны пустые промежутки во время записи данных у считывания данных. В этот момент можно брать уже следующие данные из массива. Для решения данного вопроса нужно просто добавить директиву конвейеризации. Временная диаграмма с добавлением «pipeline» представлена на рисунке 10 для решения solution3_1.

Performance Estimates

Timing

Clock		solution1	solution2	solution3
ap_clk	Target	20.00 ns	20.00 ns	20.00 ns
	Estimated	13.030 ns	13.030 ns	13.030 ns

Latency

		solution1	solution2	solution3
Latency (cycles)	min	10241	6145	4097
	max	10241	6145	4097
Latency (absolute)	min	0.205 ms	0.123 ms	81.940 us
	max	0.205 ms	0.123 ms	81.940 us
Interval (cycles)	min	10241	6145	4097
	max	10241	6145	4097

Utilization Estimates

	solution1	solution2	solution3
BRAM_18K	0	0	0
DSP48E	0	0	0
FF	150	116	39
LUT	222	214	165
URAM	0	0	0

Рис. 8 Сравнение решении 1, 2 и 3

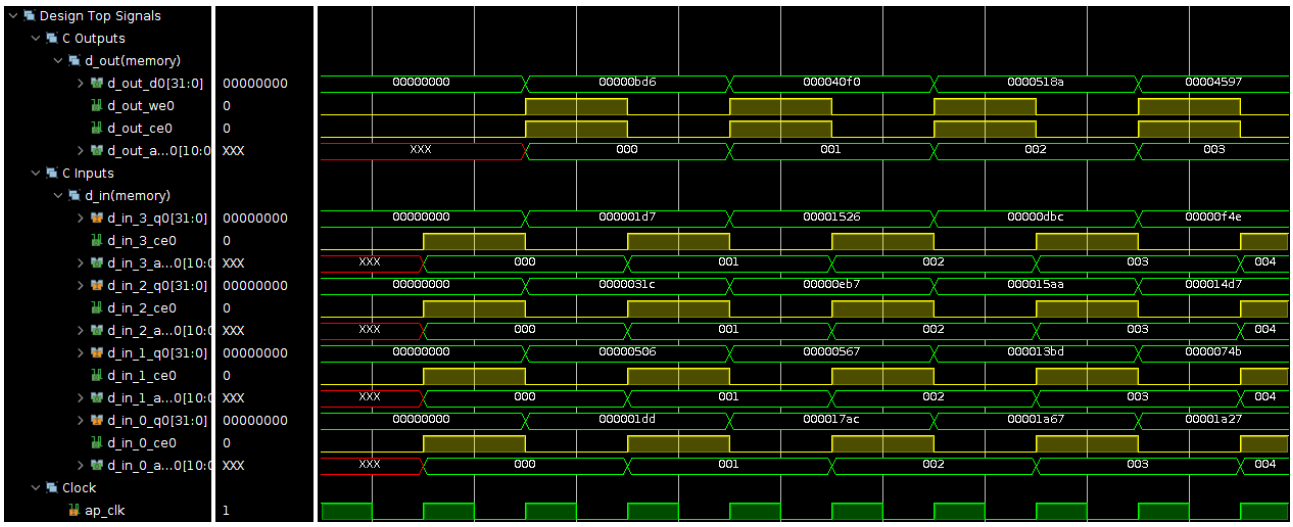

Рис. 9 Временная диаграмма solution3

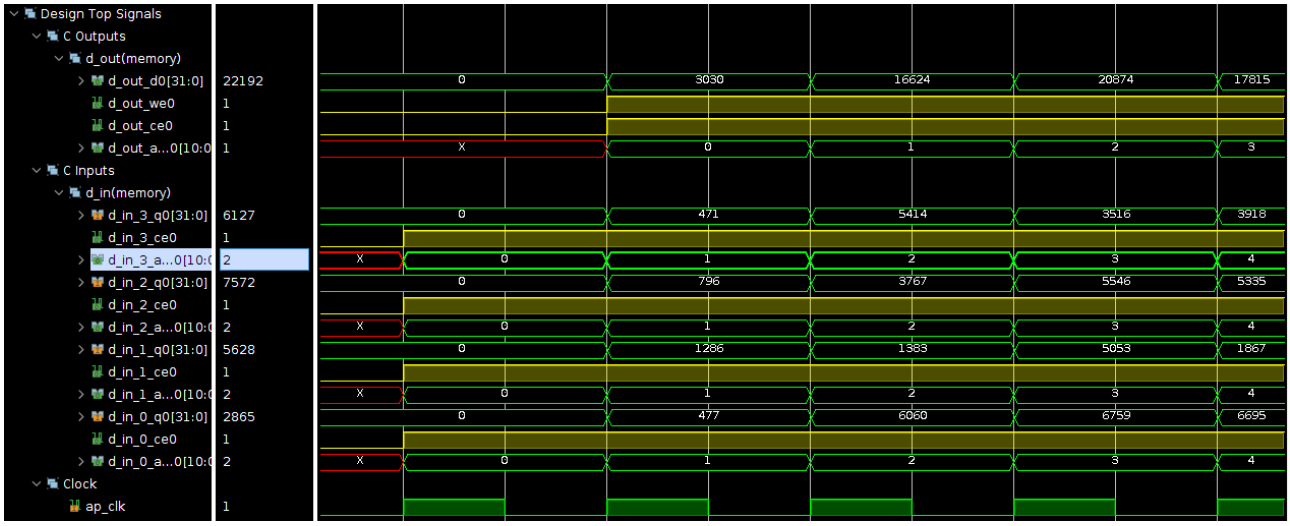

Рис. 10 Временная диаграмма solution3_1

5.4 solution4 с директивой «resource -core RAM_1P "lab5_z1" d_in» и «array_reshape -type block -factor 4 "lab5_z1" d_in». На рисунке 11 представлено сравнение временных и аппаратных параметров, как видно разделение входного массива упаковка одного порта в многоразрядное значение уменьшает количество тактов до записи в выходной массив и выходной массив теперь записывается одним тактом в два порта. И следовательно уменьшается общее количество тактов. На рисунке 12 представлена временная диаграмма solution4. На временной диаграмме видны пустые промежутки во время записи данных у считывания данных. В этот момент можно брать уже следующие данные из

массива. Для решения данного вопроса нужно просто добавить директиву конвейеризации. Временная диаграмма с добавлением «pipeline» представлена на рисунке 13 для решения solution4_1.

Performance Estimates

Timing

Clock		solution1	solution2	solution3	solution4
ap_clk	Target	20.00 ns	20.00 ns	20.00 ns	20.00 ns
	Estimated	13.030 ns	13.030 ns	13.030 ns	13.030 ns

Latency

		solution1	solution2	solution3	solution4
Latency (cycles)	min	10241	6145	4097	4097
	max	10241	6145	4097	4097
Latency (absolute)	min	0.205 ms	0.123 ms	81.940 us	81.940 us
	max	0.205 ms	0.123 ms	81.940 us	81.940 us
Interval (cycles)	min	10241	6145	4097	4097
	max	10241	6145	4097	4097

Utilization Estimates

	solution1	solution2	solution3	solution4
BRAM_18K	0	0	0	0
DSP48E	0	0	0	0
FF	150	116	39	39
LUT	222	214	165	165
URAM	0	0	0	0

Рис. 11 Сравнение решении 1, 2, 3 и 4

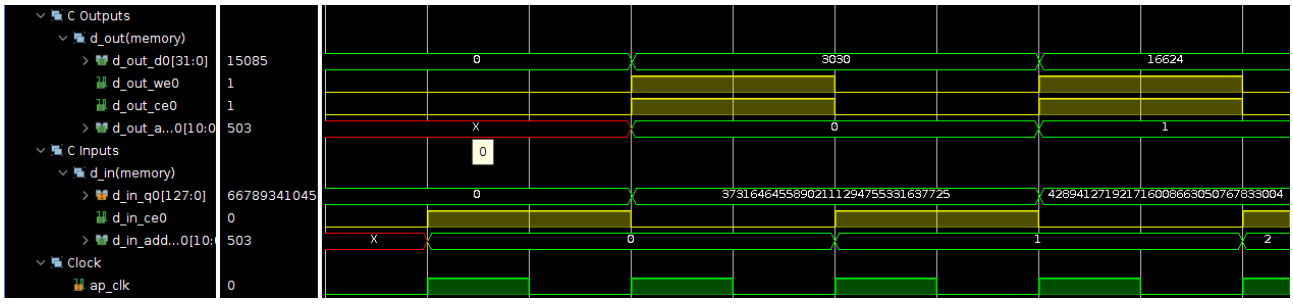

Рис. 12 Временная диаграмма solution4

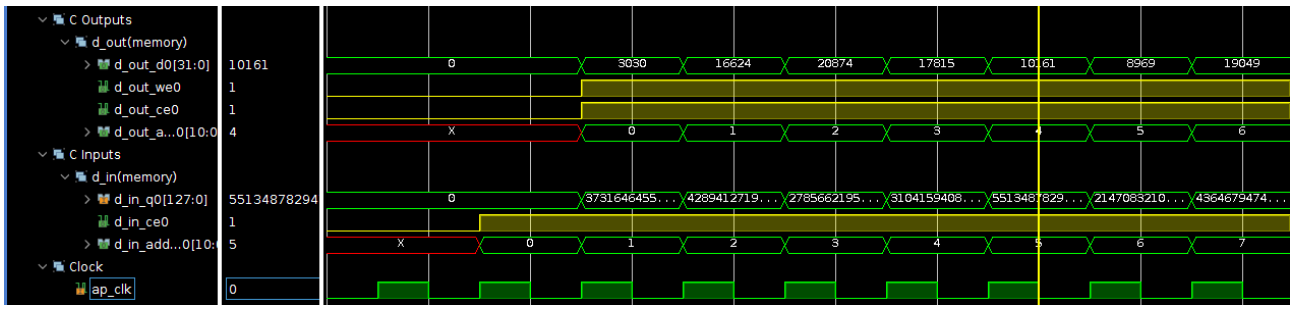

Рис. 13 Временная диаграмма solution4_1

5.5 solution5 с директивой «resource -core RAM_2P "lab5_z1" d_in» и «array_partition -type block -factor 2 "lab5_z1" d_in». На рисунке 14 представлено сравнение временных и аппаратных параметров. На рисунке 15 представлена временная диаграмма solution5. На временной диаграмме видны пустые промежутки во время записи данных у считывания данных. В этот момент можно брать уже следующие данные из массива. Для решения данного вопроса нужно просто добавить директиву конвейеризации. Временная диаграмма с добавлением «pipeline» представлена на рисунке 15 для решения solution5_1. Если изменить block до 4, то производительность не изменится.

Performance Estimates

Timing

Clock		solution5	solution3
ap_clk	Target	20.00 ns	20.00 ns
	Estimated	13.030 ns	13.030 ns

Latency

		solution5	solution3
Latency (cycles)	min	4097	4097
	max	4097	4097
Latency (absolute)	min	81.940 us	81.940 us
	max	81.940 us	81.940 us
Interval (cycles)	min	4097	4097
	max	4097	4097

Utilization Estimates

	solution5	solution3
BRAM_18K	0	0
DSP48E	0	0
FF	39	39
LUT	178	165
URAM	0	0

Рис. 14 Сравнение решении 3 и 5

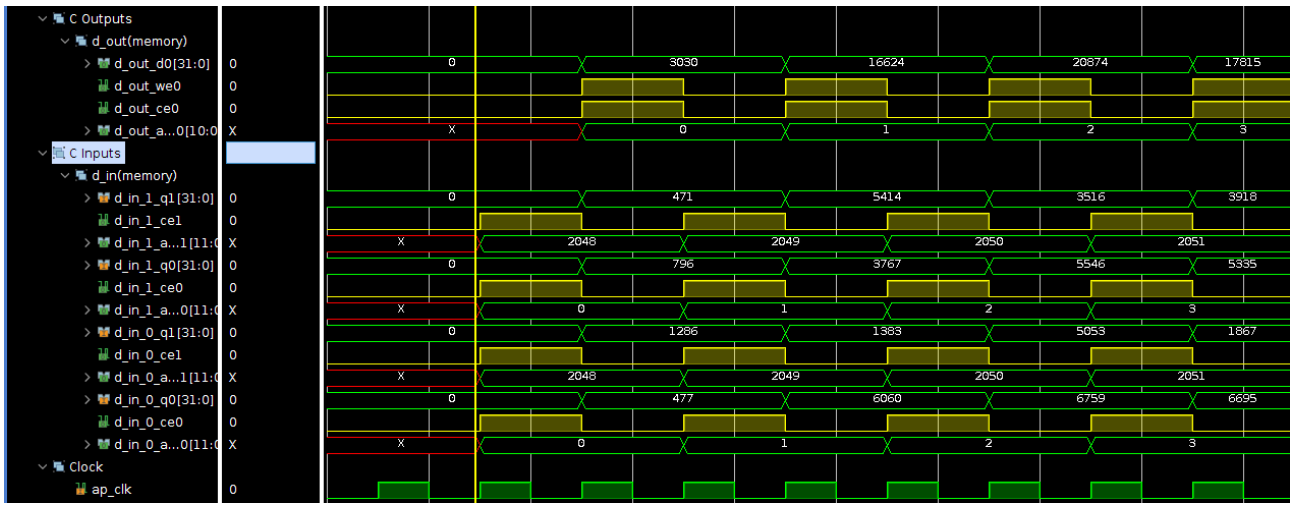

Рис. 15 Временная диаграмма solution5

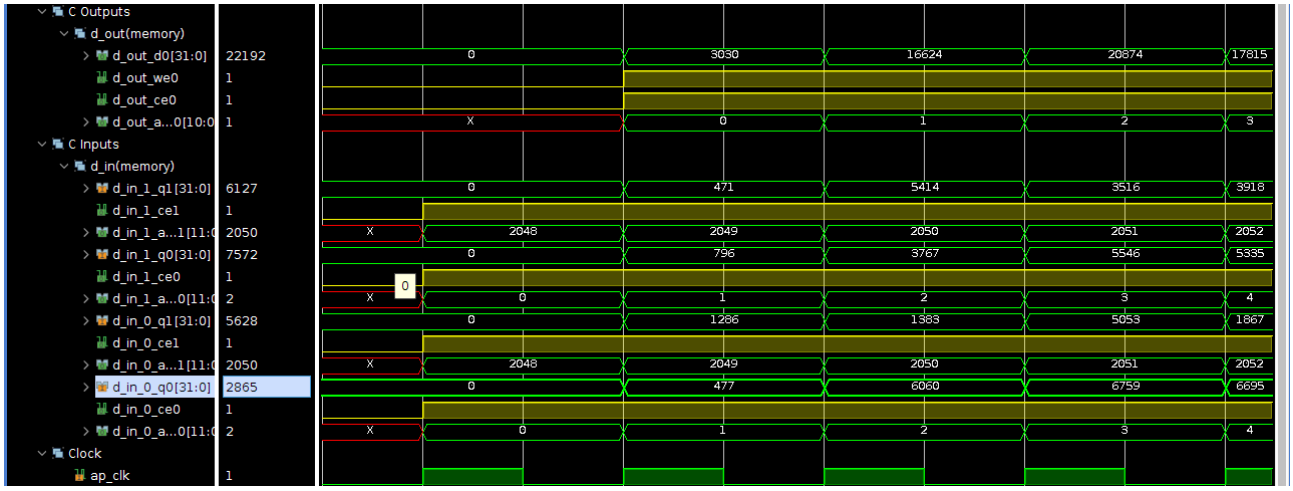

Рис. 16 Временная диаграмма solution5_1

5.6 solution6 с директивой «resource -core RAM_2P "lab5_z1" d_in» и «array_reshape -type block -factor 2 "lab5_z1" d_in». На рисунке 17 представлено сравнение временных и аппаратных параметров. На рисунке 18 представлена временная диаграмма solution6. На временной диаграмме видны пустые промежутки во время записи данных у считывания данных. В этот момент можно брать уже следующие данные из массива. Для решения данного вопроса нужно просто добавить директиву конвейеризации. Временная диаграмма с добавлением «pipeline» представлена на рисунке 19 для решения solution6_1. Если изменить block до 4, то производительность не изменится.

Performance Estimates

Timing

Clock		solution6	solution4	solution2
ap_clk	Target	20.00 ns	20.00 ns	20.00 ns
	Estimated	17.489 ns	13.030 ns	13.030 ns

Latency

		solution6	solution4	solution2
Latency (cycles)	min	4097	4097	6145
	max	4097	4097	6145
Latency (absolute)	min	81.940 us	81.940 us	0.123 ms
	max	81.940 us	81.940 us	0.123 ms
Interval (cycles)	min	4097	4097	6145
	max	4097	4097	6145

Utilization Estimates

	solution6	solution4	solution2
BRAM_18K	0	0	0
DSP48E	0	0	0
FF	45	39	116
LUT	753	165	214
URAM	0	0	0

Рис. 17 Сравнение решении 2, 4 и 6

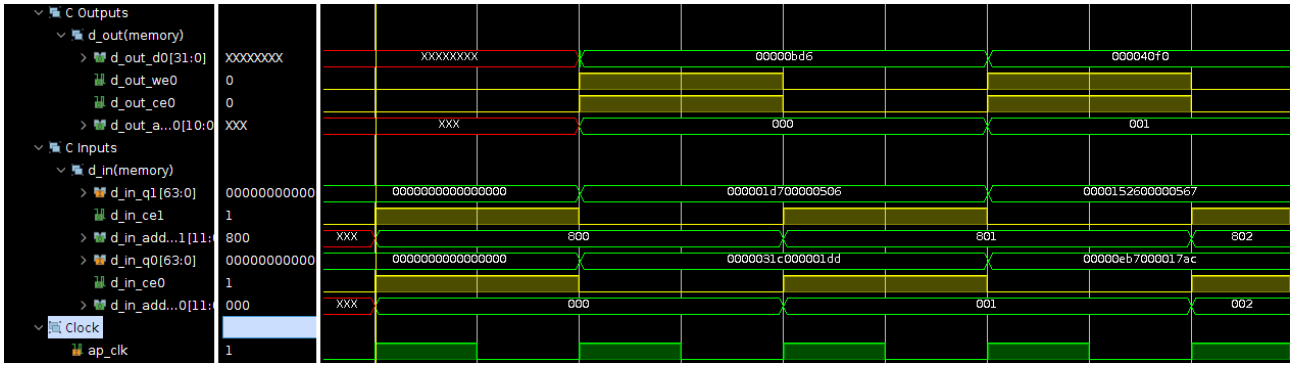

Рис. 18 Временная диаграмма solution6

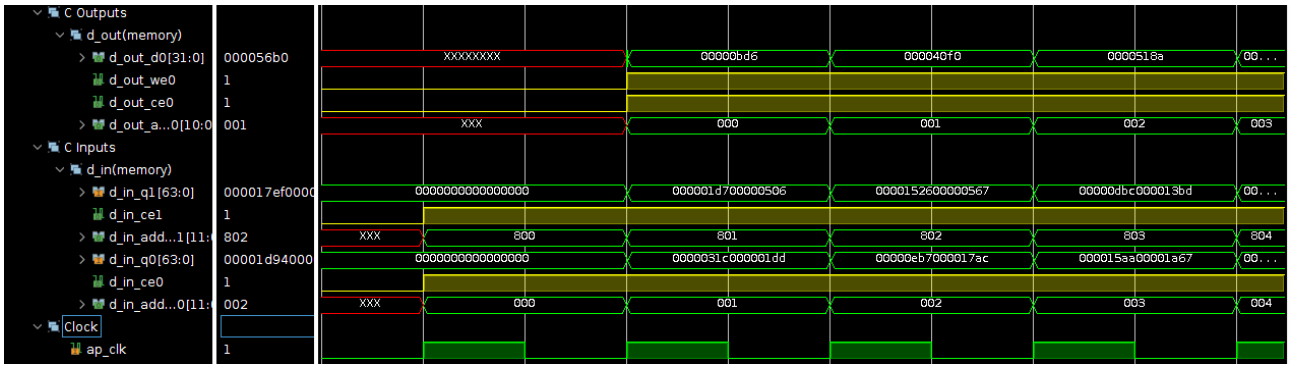

Рис. 19 Временная диаграмма solution6_1

5.7 solution7 с директивой «resource -core RAM_2P "lab5_z1" d_in», «array_partition -type block -factor 2 "lab5_z1" d_in» и «unroll -skip_exit_check -factor 2 "lab5_z1/for_label0" d_in». На рисунке 20 представлено сравнение временных и аппаратных параметров. На рисунке 21 представлена временная диаграмма solution7. На временной диаграмме видны пустые промежутки во время записи данных у считывания данных. В этот момент можно брать уже следующие данные из массива. Для решения данного вопроса нужно просто добавить директиву конвейеризации. Временная диаграмма с добавлением «pipeline» представлена на рисунке 22 для решения solution7_1.

Performance Estimates

Timing

Clock		solution7	solution4
ap_clk	Target	20.00 ns	20.00 ns
	Estimated	13.030 ns	13.030 ns

Latency

		solution7	solution4
Latency (cycles)	min	2049	4097
	max	2049	4097
Latency (absolute)	min	40.980 us	81.940 us
	max	40.980 us	81.940 us
Interval (cycles)	min	2049	4097
	max	2049	4097

Utilization Estimates

	solution7	solution4
BRAM_18K	0	0
DSP48E	0	0
FF	26	39
LUT	403	165
URAM	0	0

Рис. 20 Сравнение решении 4 и 7

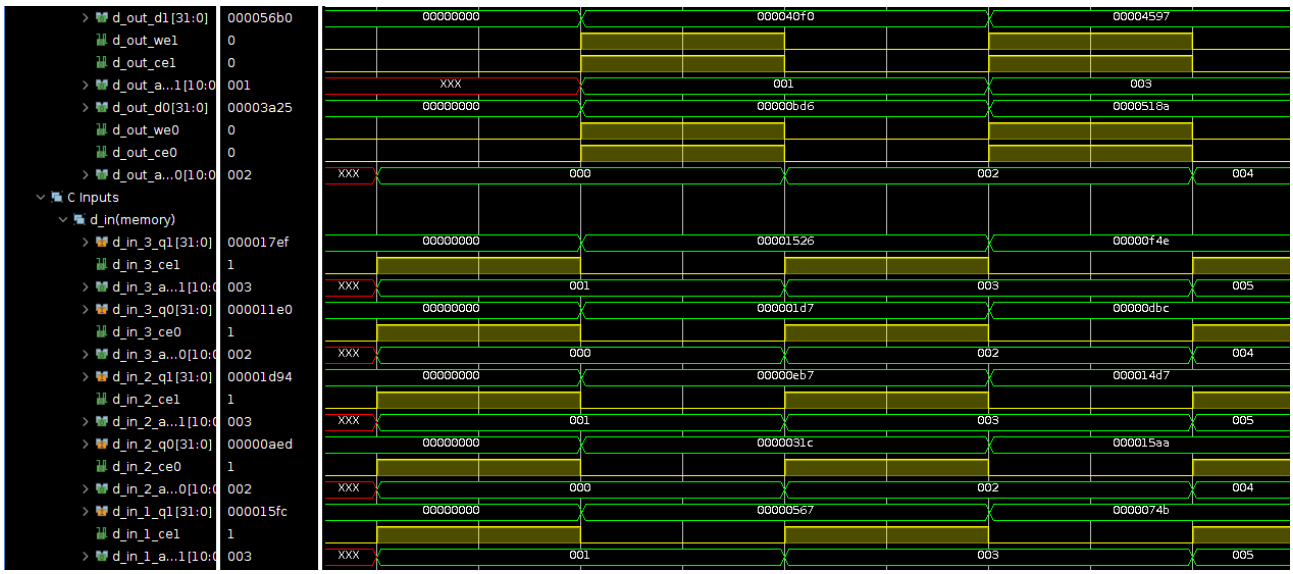

Рис. 21 Временная диаграмма solution7

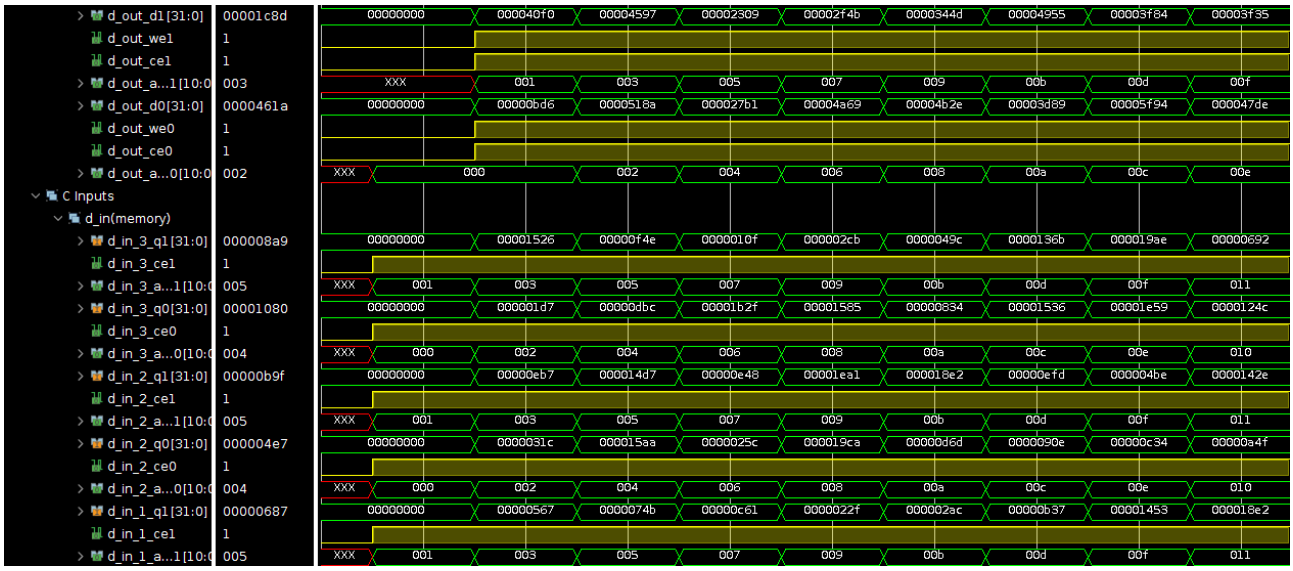

Рис. 22 Временная диаграмма solution7_1

5.8 Сравнение всех результатов

На рисунке 23 приведено сравнение отчетов всех решении. На рисунке 24 приведено сравнение всех решений с расчетом Latency в нс и график зависимости производительности с аппаратной частью от номера решения.

Performance Estimates

Timing

Clock		solution1	solution1_1	solution2	solution2_1	solution3	solution3_1	solution4	solution4_1	solution5	solution5_1	solution6	solution6_1	solution7	solution7_1
ap_clk	Target	20.00 ns	20.00 ns	20.00 ns	20.00 ns	20.00 ns	20.00 ns	20.00 ns	20.00 ns	20.00 ns	20.00 ns	20.00 ns	20.00 ns	20.00 ns	20.00 ns
	Estimated	13.030 ns	13.030 ns	13.030 ns	13.030 ns	13.030 ns	13.030 ns	13.030 ns	13.030 ns	13.030 ns	13.030 ns	17.489 ns	17.489 ns	13.030 ns	13.030 ns

Latency

		solution1	solution1_1	solution2	solution2_1	solution3	solution3_1	solution4	solution4_1	solution5	solution5_1	solution6	solution6_1	solution7	solution7_1
Latency (cycles)	min	10241	8194	6145	4098	4097	2050	4097	2050	4097	2050	4097	2050	2049	1026
	max	10241	8194	6145	4098	4097	2050	4097	2050	4097	2050	4097	2050	2049	1026
Latency (absolute)	min	0.205 ms	0.164 ms	0.123 ms	81.960 us	81.940 us	41.000 us	81.940 us	41.000 us	81.940 us	41.000 us	81.940 us	41.000 us	40.980 us	20.520 us
	max	0.205 ms	0.164 ms	0.123 ms	81.960 us	81.940 us	41.000 us	81.940 us	41.000 us	81.940 us	41.000 us	81.940 us	41.000 us	40.980 us	20.520 us
Interval (cycles)	min	10241	8194	6145	4098	4097	2050	4097	2050	4097	2050	4097	2050	2049	1026
	max	10241	8194	6145	4098	4097	2050	4097	2050	4097	2050	4097	2050	2049	1026

Utilization Estimates

	solution1	solution1_1	solution2	solution2_1	solution3	solution3_1	solution4	solution4_1	solution5	solution5_1	solution6	solution6_1	solution7	solution7_1
BRAM_18K	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DSP48E	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FF	150	165	116	131	39	30	39	30	39	30	45	36	26	40
LUT	222	242	214	234	165	184	165	184	178	197	753	772	403	431
URAM	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Рис. 23 Сравнение отчетов всех решении

		sol1	sol1_1	sol2	sol2_1	sol3	sol3_1	sol4	sol4_1	sol5	sol5_1	sol6	sol6_1	sol7	sol7_1
Clock	Target (ns)	20	20	20	20	20	20	20	20	20	20	20	20	20	20
	Estimated (ns)	13,03	13,03	13,03	13,03	13,03	13,03	13,03	13,03	13,03	13,03	17,48	17,48	13,03	13,03
Latency	(cycles)	10241	8194	6145	4098	4097	2050	4097	2050	4097	2050	4097	2050	2049	1026
	(ns)	133440	106768	80069	53397	53384	26712	53384	26712	53384	26712	71616	35834	26698	13369
Resources	BRAM_18K	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	DSP48E	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	FF	150	165	116	131	39	30	39	30	39	30	45	36	26	40
	LUT	222	242	214	234	165	184	165	184	178	197	753	772	403	431
	URAM	0	0	0	0	0	0	0	0	0	0	0	0	0	0

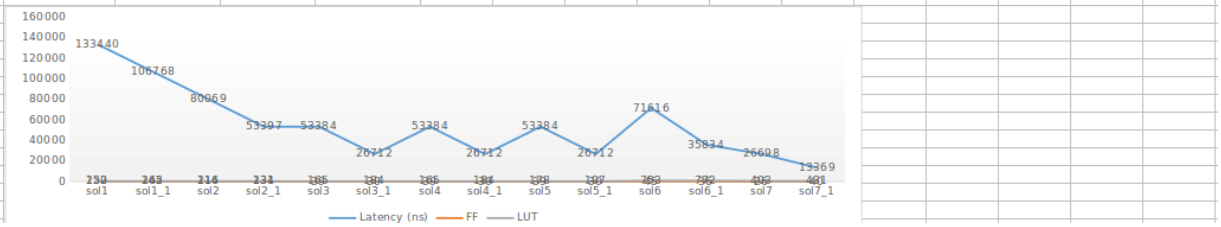

Рис. 24 Сравнение отчетов всех решении в виде таблицы и графика

Как видно из рисунка 24 самый лучший результат по производительности и по аппаратным затратам у решения №3_1 и №4_1.

6. Исследование часть 2

6.1 Сравнение в Vivado_HLS

На рисунке 25 представлено сравнение отчетов для решении №1, 7 и 7_1 для разных clock period = {4, 12, 20} и clock_uncertainty = 1, после увеличения N = 131072. На рисунке 26 представлено сравнение решении в виде графика и таблицы с Latency в нс. Как видно из рисунка лучшим решением по временным показателям является solution 7_1_3.

Performance Estimates

Timing

Clock		solution1	solution1_2	solution1_3	solution7	solution7_2	solution7_3	solution7_1	solution7_1_2	solution7_1_3
ap_clk	Target	20.00 ns	12.00 ns	4.00 ns	20.00 ns	12.00 ns	4.00 ns	20.00 ns	12.00 ns	4.00 ns
	Estimated	13.030 ns	9.773 ns	2.702 ns	13.030 ns	9.773 ns	2.702 ns	13.122 ns	10.328 ns	2.702 ns

Latency

		solution1	solution1_2	solution1_3	solution7	solution7_2	solution7_3	solution7_1	solution7_1_2	solution7_1_3
Latency (cycles)	min	163841	196609	262145	32769	49153	81921	16386	16387	16389
	max	163841	196609	262145	32769	49153	81921	16386	16387	16389
Latency (absolute)	min	3.277 ms	2.359 ms	1.049 ms	0.655 ms	0.590 ms	0.328 ms	0.328 ms	0.197 ms	65.556 us
	max	3.277 ms	2.359 ms	1.049 ms	0.655 ms	0.590 ms	0.328 ms	0.328 ms	0.197 ms	65.556 us
Interval (cycles)	min	163841	196609	262145	32769	49153	81921	16386	16387	16389
	max	163841	196609	262145	32769	49153	81921	16386	16387	16389

Utilization Estimates

	solution1	solution1_2	solution1_3	solution7	solution7_2	solution7_3	solution7_1	solution7_1_2	solution7_1_3
BRAM_18K	0	0	0	0	0	0	0	0	0
DSP48E	0	0	0	0	0	0	0	0	0
FF	166	199	233	34	99	488	52	163	699
LUT	235	238	219	434	440	479	462	465	589
URAM	0	0	0	0	0	0	0	0	0

Рис. 25 Сравнение отчетов всех решении

		Sol1	Sol1_2	Sol1_3	Sol7	Sol7_2	Sol7_3	Sol7_1	Sol7_1_2	Sol7_1_3
Clock	Target (ns)	20	12	4	20	12	4	20	12	4
	Estimated (ns)	13,03	9,77	2,70	13,03	9,77	2,70	13,12	10,32	2,70
Latency	(cycles)	163841	196609	262145	32769	49153	81921	16386	16837	16839
	(ns)	2134848	1920870	707792	426980	480225	221187	214984	173758	45465
Resources	BRAM_18K	0	0	0	0	0	0	0	0	0
	DSP48E	0	0	0	0	0	0	0	0	0
	FF	166	199	233	34	99	488	52	163	699
	LUT	235	238	219	434	440	479	462	465	589
	URAM	0	0	0	0	0	0	0	0	0

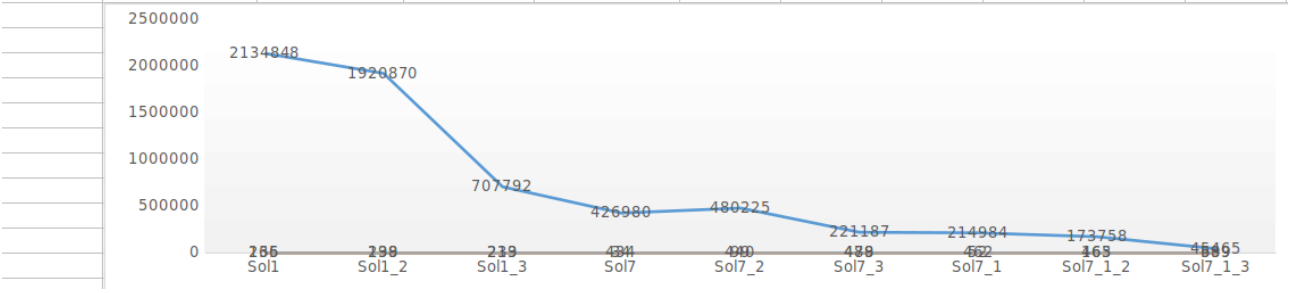

Рис. 26 Сравнение отчетов всех решении в виде таблицы и графика

6.2 Запуск теста на ПК

На рисунке 27 представлен исходный код модифицированного теста для ПК. Тест обеспечивает проверку производительности функции на ПК. Функция была скомпилирована компилятором gcc-9.3.0. В таблице 1 представлены характеристики ПК:

Таблица 1

CPU	Intel Core i5-6200U 2.3 GHz
Core	2
Threads	4
RAM	8 Gb

```
int main()
{
    int pass=0;

    // Call the function for 32 transactions
    int d_in[N];
    int d_out[N];
    struct timespec t0, t1;
    double acc_time = 0.0;

    for (int i = 0; i < 32; ++i){
        for(int j = 0; j < N; j++){
            d_in[j] = rand() % (N - 1) + 1;
        }

        if(clock_gettime(CLOCK_REALTIME, &t0) != 0) {
            perror("Error in calling clock_gettime\n");
            exit(EXIT_FAILURE);
        }
        lab5_z1(d_in, d_out);
        if(clock_gettime(CLOCK_REALTIME, &t1) != 0) {
            perror("Error in calling clock_gettime\n");
            exit(EXIT_FAILURE);
        }
        double diff_time = (((double)(t1.tv_sec - t0.tv_sec))*1000000000.0) + (double)(t1.tv_nsec - t0.tv_nsec);
        acc_time += diff_time;
        double temp_avg_time = acc_time / (i + 1); // take average time
        printf("Elapsed time: %.4lf nanoseconds\n", temp_avg_time);

        pass = cmp_arr(d_in, d_out);
        if (pass == 0)  {
            fprintf(stderr, "----------Fail!-----------\n");
            return 1;
        }
    }


    fprintf(stdout, "----------Pass!-----------\n");
    return 0;
}
```

Рис. 27 Исходный код модифицированного теста для ПК

На рисунке 28 представлен результат запуска модифицированного теста на ПК. Как видно среднее время выполнения составляет 148360 нс, что на ~100000 нс дольше, чем лучшее аппаратное решение.

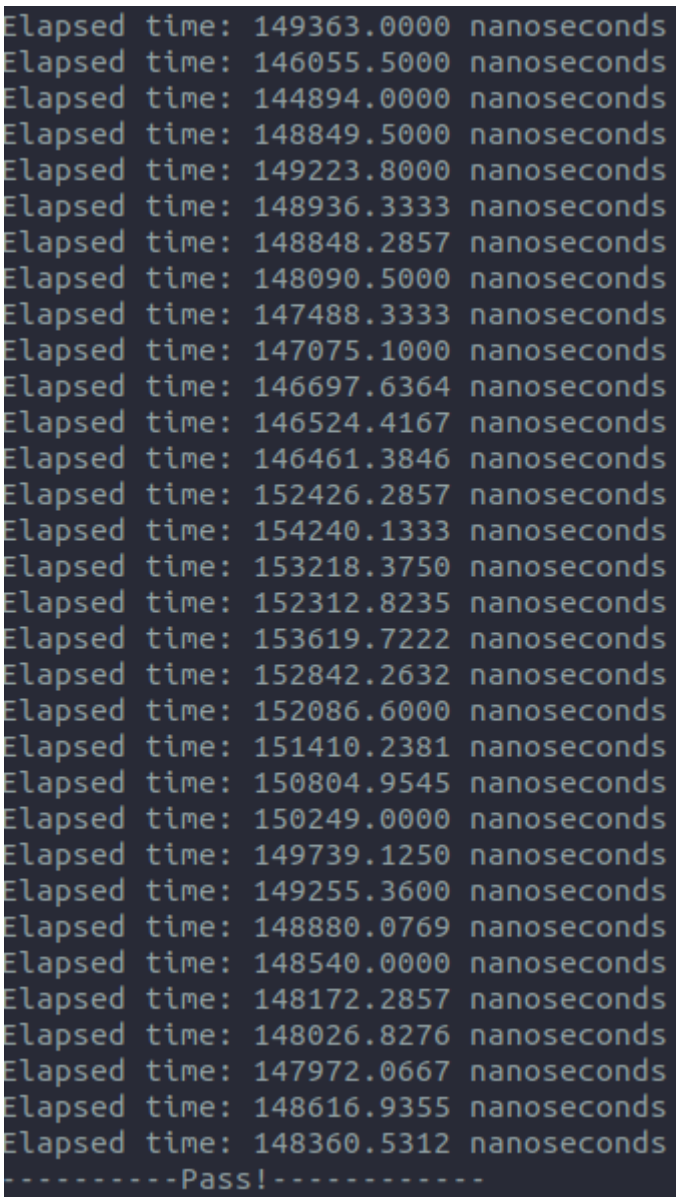

Рис. 28 Результат запуска теста на ПК

Вывод

В данной работе была изучена возможность добавления директив по оптимизации работы с массивами для синтезируемой функции. Был произведен сравнительный анализ между решением без добавлением и с добавлением различных директив с одним и двумя портами для входного массива, раскруткой цикла и разбиением массива на блоки разного размера. Также было произведено сравнение временных показателей между решением полученным Vivado HLS и программным решения на ПК. Как видно из результатов решением полученное на ПК медленнее, чем лучшее решение полученное синтезированием в Vivado HLS.