Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологии
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ Lab5_Z2

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.

«25» ноября 2021

Санкт-Петербург
2021

1. Задание

Текст задания находится в файле «Задание lab5_z2.docx»

2. Исходный код функции

Исходный код синтезируемой функции lab5_z2 представлен на рисунке 1.

Рис. 1

```
#include "lab5_z2.h"

void lab5_z2(int d_in[N], int d_out[N-3])
{
    for_label0: for(int i = 0; i < (N-3); i=i+4) {
        d_out[i] = d_in[i] + d_in[i+1] + d_in[i+2] + d_in[i+3];
    }

}
```

Исходный код функции lab5_z2

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_z2 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» и «array_partition -type block -factor 4 "lab5_z2" d_in» для входного массива d_in и добавление директивы pipeline для цикла for_label0 в функции.

```c
int cmp_arr(int d_in[N], int d_cmp[N-3])
{
    for(int i = 0; i < (N-3); i+=4) {
        int tmp = d_in[i] + d_in[i+1] + d_in[i+2] + d_in[i+3];
        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-3];

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


        lab5_z2(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_z2_test.c тестирования функции

```
# Create a project
open_project -reset lab5_z2_prj

# The source file and test bench
add_files       ./source/lab5_z2.c
add_files -tb   ./source/lab5_z2_test.c

# Specify the top-level function for synthesis
set_top         lab5_z2

############################
# 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_z2" d_in
set_directive_array_partition -type block -factor 4 "lab5_z2" d_in
set_directive_pipeline "lab5_z2/for_label0"

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

Рис. 3 Текст команд для создания решения

5. Решение №1

Для размера массива N=8192 были директивы представлены на рисунке 4. На рисунке 5 представлен результат синтезирования performance и utilization estimates. На рисунке 6 представлена временная диаграмма для решения.

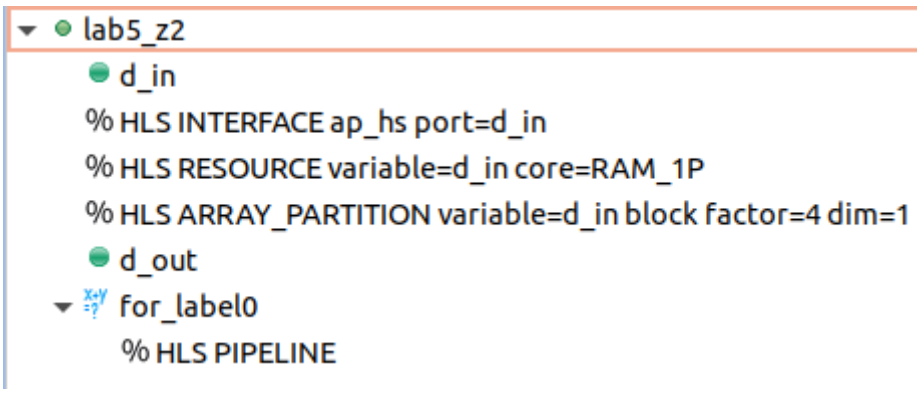

Рис. 4 Директивы для решения

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	20.00 ns	5.226 ns	2.50 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
2050	2050	41.000 us	41.000 us	2050	2050	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	36	-
FIFO	-	-	-	-	-
Instance	-	-	0	21	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	66	-
Register	-	-	17	-	-
Total	0	0	17	123	0
Available	40	40	16000	8000	0

Рис. 5 Результат синтезирования функции N=8192

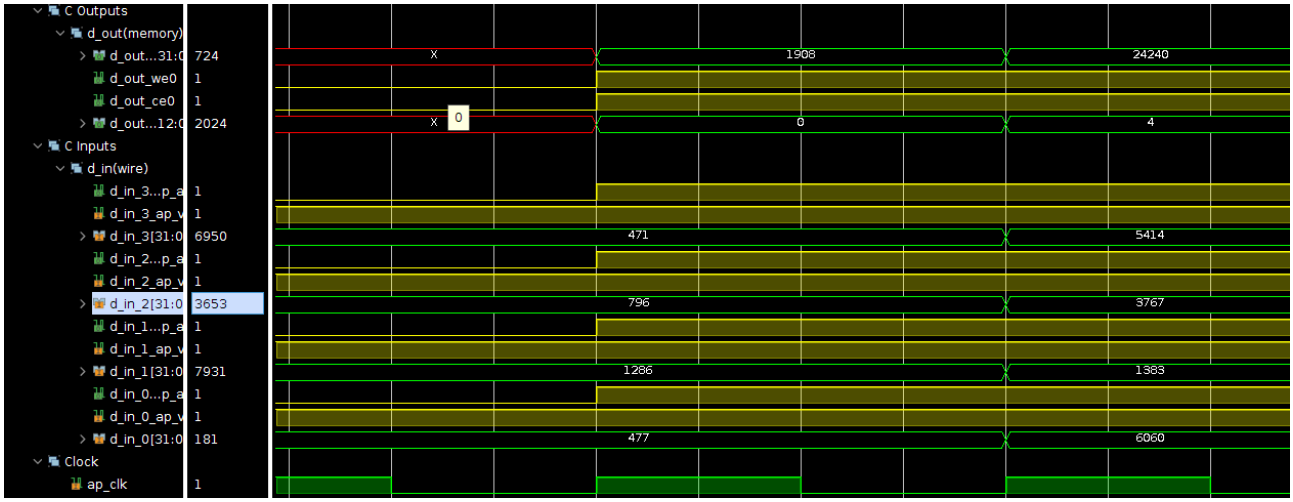

Рис. 6 Временная диаграмма для решения

5.2 Решение №2

Для размера массива N=131072 на рисунке 7 представлен результат синтезирования performance и utilization estimates. На рисунке 8 представлен schedule viewer solution1. На рисунке 9 представлена временная диаграмма для решения. Как видно из результатов синтезирования закономерно увеличилось количество latency cycle, то есть количество необходимых итерации. Количество аппаратных ресурсов не изменилось. Как видно из рисунка 7 время выполнения функции составило Latency = 171256.0 нс = 0.17 мс.

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	20.00 ns	5.226 ns	2.50 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
32770	32770	0.655 ms	0.655 ms	32770	32770	none

Detail

- Instance
- Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	45	-
FIFO	-	-	-	-	-
Instance	-	-	0	21	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	66	-
Register	-	-	21	-	-
Total	0	0	21	132	0

Рис. 7 Результат синтезирования функции N=131072

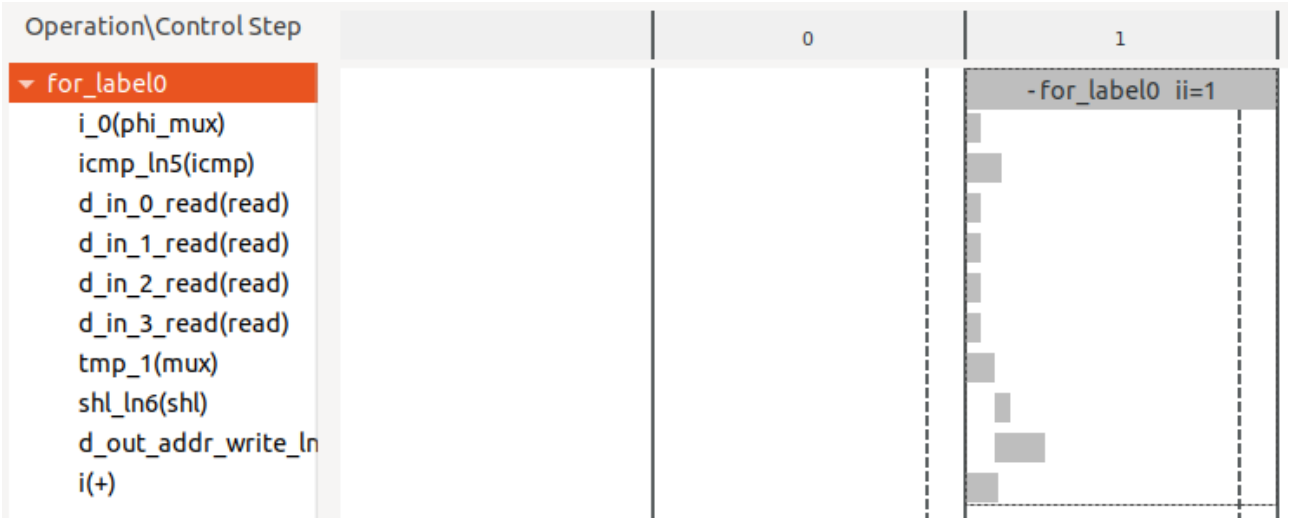

Рис. 8 Schedule viewer синтезирования функции

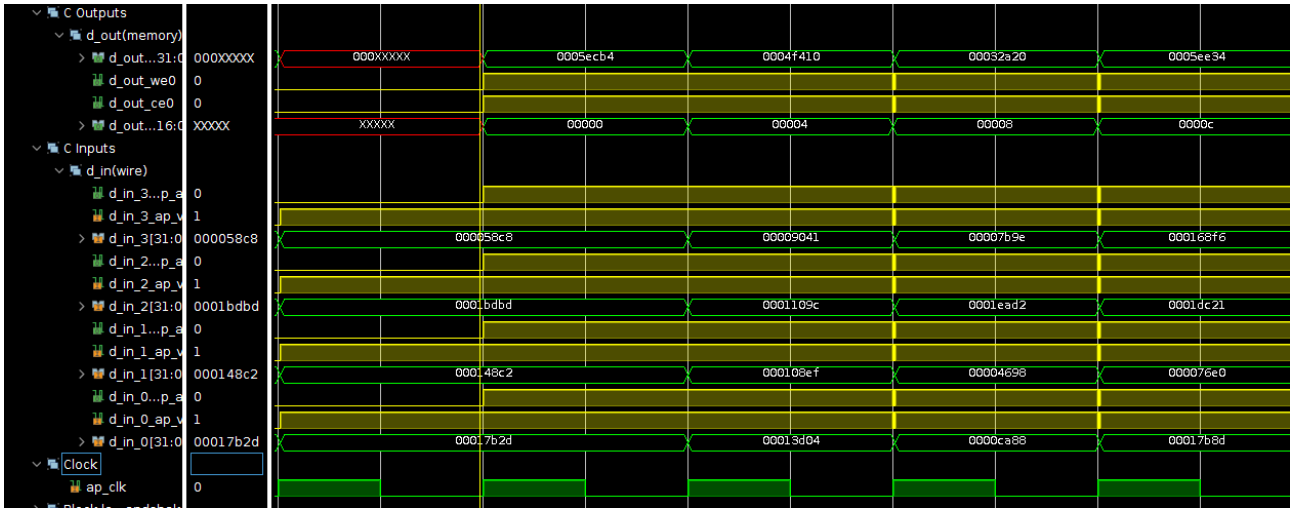

Рис. 9 Временная диаграмма для решения

6. Решение с модифицированным тестом на ПК

Исходный код модифицированного теста для проверки функции lab5_z2 приведен на рисунке 10. Тест обеспечивает проверку производительности функции на ПК. На рисунке 11 представлен результат запуска теста, как видно из рисунка среднее время выполнения составило 152933.2 нс = 0.15 мс. 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

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

    // Call the function for 32 transactions
    int d_in[N];
    int d_out[N-3];
    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_z2(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. 10 Исходный код модифицированного теста

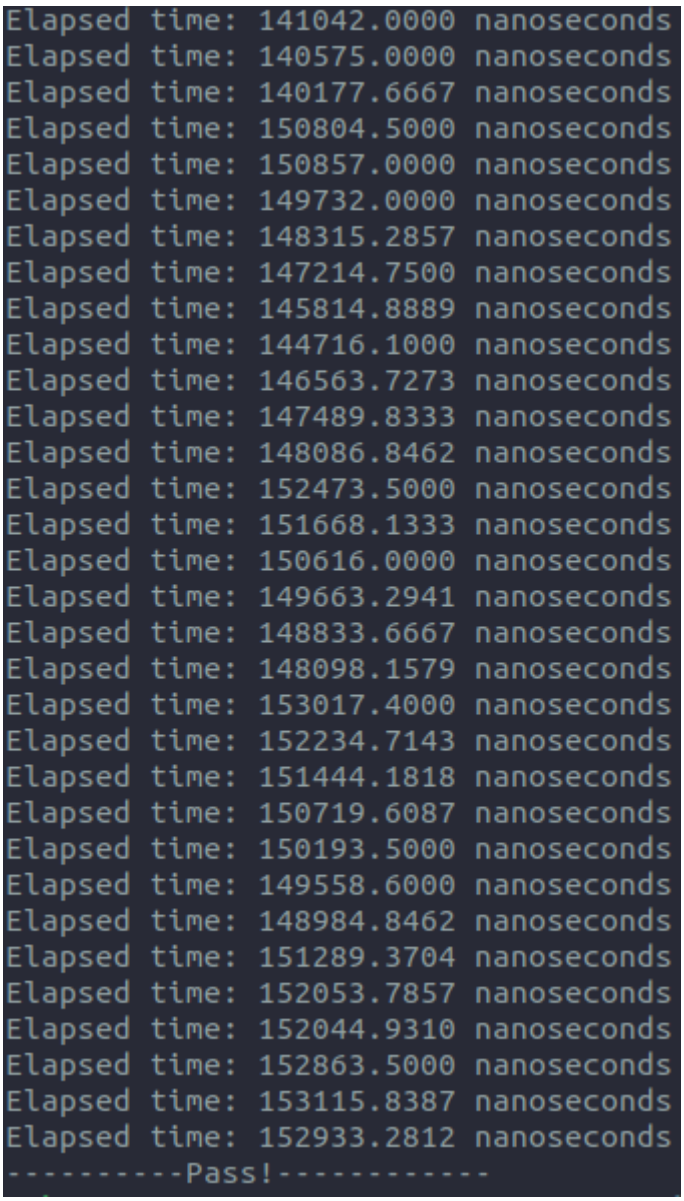

Рис. 11 Результат запуска модифицированного теста

Вывод

В данной работе была изучена возможность добавления директив по оптимизации работы с массивами для синтезируемой функции. Был подобраны директивы для достижения заданных показателей синтезирования функции. Также было произведено сравнение временных показателей между решением полученным Vivado HLS и программным решения на ПК. Как видно из результатов решением полученное на ПК почти такое же, как и у решения полученное аппаратно в Vivado HLS.