

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

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ Lab5\_Z3

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.

«26» ноября 2021

Санкт-Петербург  
2021

## 1. Задание

Текст задания находится в файле «Задание lab5\_z3.docx»

## 2. Исходный код функции

Исходный код синтезируемой функции lab5\_z3 представлен на рисунке 1.

```
#include "lab5_z3.h"

void lab5_z3(data d_in, data* d_out)
{
    looplabel0: for(int i = 0; i < N; ++i) {
        d_out->A[i] = d_in.A[i] + d_in.B[i];
        d_out->B[i] = d_in.A[i] - d_in.B[i];
    }
}
```

Рис. 1 Исходный код функции lab5\_z3

Функция принимает два аргумента структуры с массивами типа int — вычисляет сумму и разницу элементов массивов и записывает результат в массивы выходной структуры.

## 3. Исходный код теста

Исходный код теста проверки функции lab5\_z3 приведен на рисунке 2.

Тест обеспечивает проверку корректной работы функции.

## 4. Командный файл

На рисунке 3 представлен текст команд для автоматизированного создания варианта аппаратной реализации — solution1 с clock period 20;

```

int cmp_arr(data d_in, data* d_cmp)
{
    for(int i = 0; i < N; ++i) {
        int tmp1 = d_in.A[i] + d_in.B[i];
        int tmp2 = d_in.A[i] - d_in.B[i];
        if (d_cmp->A[i] != tmp1 || d_cmp->B[i] != tmp2)
            return 0;
    }

    return 1;
}

int main () {

    int pass = 0;
    // Create input data
    data d_in;
    data d_out;

    for (int i = 0; i < 3; ++i){
        for(int j = 0; j < N; j++){
            d_in.A[j] = rand() % (N - 1) + 1;
            d_in.B[j] = (rand() + (rand()/2 + 1)) % (N - 1) + 1;
        }

        lab5_z3(d_in, &d_out);
        pass = cmp_arr(d_in, &d_out);
        if (pass == 0) {
            fprintf(stderr, "-----Fail!-----\n");
            return 1;
        }
    }

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

```

Рис. 2 Исходный код lab5\_z3\_test.c тестирования функции

```
# Create a project
open_project -reset lab5_z3_prj

# The source file and test bench
add_files      ./source/lab5_z3.c
add_files -tb  ./source/lab5_z3_test.c

# Specify the top-level function for synthesis
set_top      lab5_z3

#####
# 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 {xa7a12tcsq325-1q}
create_clock -period 20

# Simulate the C code
csim_design

# synthesis and c/rtl sim
csynth_design

exit
```

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.


- ▼ ● lab5\_z3
  - ▼ ● d\_in
    - A
      - % HLS ARRAY\_RESHAPE variable=d\_in.A cyclic factor=64 dim=1
      - % HLS RESOURCE variable=d\_in.A core=RAM\_1P
    - B
      - % HLS ARRAY\_RESHAPE variable=d\_in.B cyclic factor=64 dim=1
      - % HLS RESOURCE variable=d\_in.B core=RAM\_1P
  - ▼ ● d\_out
    - A
      - % HLS ARRAY\_RESHAPE variable=d\_out->A cyclic factor=64 dim=1
      - % HLS RESOURCE variable=d\_out->A core=RAM\_1P
    - B
      - % HLS ARRAY\_RESHAPE variable=d\_out->B cyclic factor=64 dim=1
      - % HLS RESOURCE variable=d\_out->B core=RAM\_1P
  - ▼  looplabel0
    - % HLS UNROLL skip\_exit\_check factor=64
    - % HLS PIPELINE

Рис. 4 Директивы для решения

## Timing

### Summary

Clock	Target	Estimated	Uncertainty
ap_clk	20.00 ns	7.278 ns	2.50 ns

## Latency

### Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
130	130	2.600 us	2.600 us	130	130	none

### Detail

#### + Instance

#### + Loop

## Utilization Estimates

### Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	5030	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	45	-
Register	-	-	27	-	-
Total	0	0	27	5075	0
Available	40	40	16000	8000	0
Utilization (%)	0	0	~0	63	0

Рис. 5 Performance and utilization estimates для решения

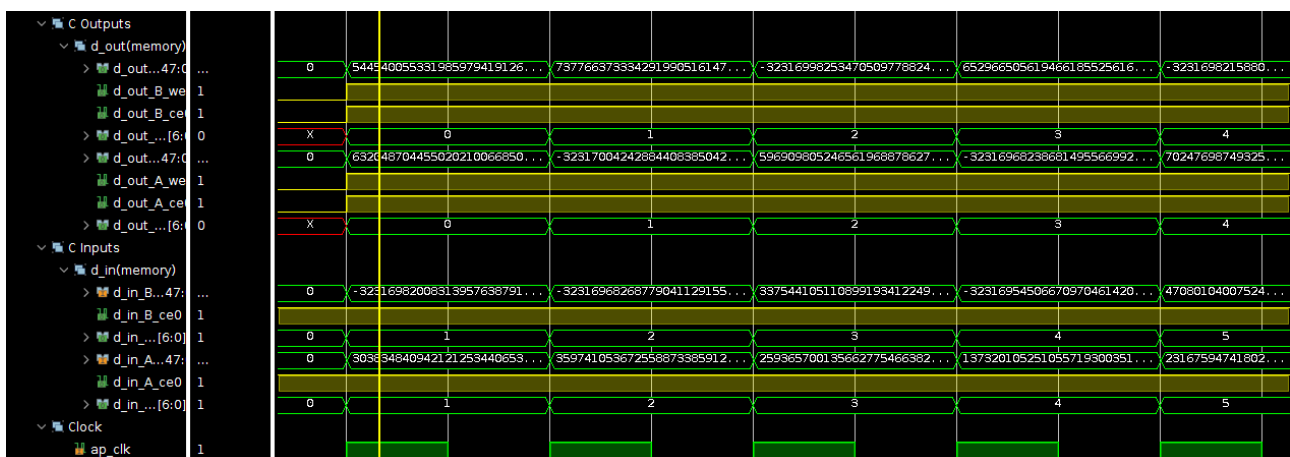


Рис. 6 Временная диаграмма решения

## 5.2 Решение №2

Для размера массива  $N=262144$  на рисунке 7 представлен результат синтезирования performance и utilization estimates. На рисунке 8 представлен schedule viewer solution1. 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  $\text{Latency} = 32103.7 \text{ нс} = 0.03 \text{ мс}$ .

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	20.00 ns	7.834 ns	2.50 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
4098	4098	81.960 us	81.960 us	4098	4098	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	5040	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	45	-
Register	-	-	37	-	-
Total	0	0	37	5085	0

Рис. 7 Результат синтезирования функции  $N=262144$

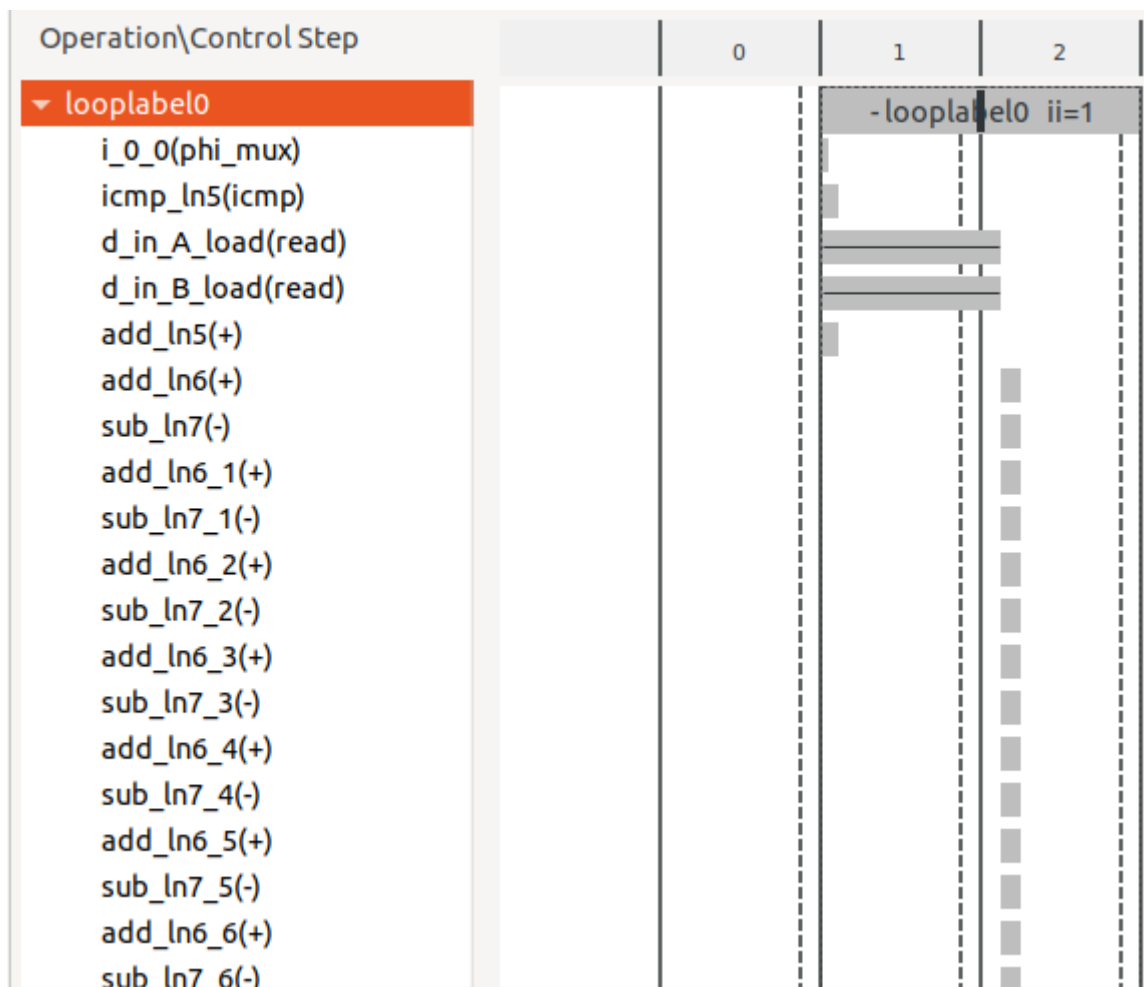


Рис. 8 Schedule viewer синтезирования функции

## 6. Решение с модифицированным тестом на ПК

Исходный код модифицированного теста для проверки функции lab5\_z3 приведен на рисунке 9. Тест обеспечивает проверку производительности функции на ПК. На рисунке 10 представлен результат запуска теста, как видно из рисунка среднее время выполнения составило 1295419.7 нс = 1.2 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



```

int main()
{
    int pass=0;

    // Call the function for 32 transactions
    data d_in;
    data d_out;
    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.A[j] = rand() % (N - 1) + 1;
            d_in.B[j] = (rand() + (rand()/2 + 1)) % (N - 1) + 1;
        }

        if(clock_gettime(CLOCK_REALTIME, &t0) != 0) {
            perror("Error in calling clock_gettime\n");
            exit(EXIT_FAILURE);
        }
        lab5_z3(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. 9 Исходный код модифицированного теста

```
Elapsed time: 2096174.0000 nanoseconds
Elapsed time: 1832031.0000 nanoseconds
Elapsed time: 1628617.3333 nanoseconds
Elapsed time: 1516676.2500 nanoseconds
Elapsed time: 1520987.2000 nanoseconds
Elapsed time: 1469652.0000 nanoseconds
Elapsed time: 1427253.0000 nanoseconds
Elapsed time: 1448369.8750 nanoseconds
Elapsed time: 1413905.8889 nanoseconds
Elapsed time: 1391971.6000 nanoseconds
Elapsed time: 1395698.2727 nanoseconds
Elapsed time: 1376253.8333 nanoseconds
Elapsed time: 1359470.6154 nanoseconds
Elapsed time: 1363461.0714 nanoseconds
Elapsed time: 1351591.8000 nanoseconds
Elapsed time: 1338426.6875 nanoseconds
Elapsed time: 1339036.4118 nanoseconds
Elapsed time: 1331741.2778 nanoseconds
Elapsed time: 1322110.2632 nanoseconds
Elapsed time: 1327638.8500 nanoseconds
Elapsed time: 1318250.9048 nanoseconds
Elapsed time: 1313017.6818 nanoseconds
Elapsed time: 1316036.5217 nanoseconds
Elapsed time: 1318928.0000 nanoseconds
Elapsed time: 1311006.9200 nanoseconds
Elapsed time: 1311206.0769 nanoseconds
Elapsed time: 1308297.4444 nanoseconds
Elapsed time: 1303649.5357 nanoseconds
Elapsed time: 1302505.6207 nanoseconds
Elapsed time: 1297065.7000 nanoseconds
Elapsed time: 1296130.5806 nanoseconds
Elapsed time: 1295419.7500 nanoseconds
-----Pass!-----
```

Рис. 10 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 синтеза функции. Также было произведено сравнение временных показателей между решением полученным Vivado HLS и программным решением на ПК. Как видно из результатов решением полученное на ПК хуже, чем решение полученное аппаратно в Vivado HLS.