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

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ Lab3_Z2

Дисциплина: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Введение в Port-Level I/O Protocols

Выполнил студент гр. 01502 С.С. Гаспарян

Руководитель, доцент Антонов А.П.

«8» октября 2021

Санкт-Петербург
2021

Задание

Текст задания находится в файле «Задание lab3_z2.docx»

2. Исходный код функции

Код объявления синтезируемой функции и макроопределений представлен на рисунке 1.

```
1 #ifndef LAB3_z2_IO_H_
2 #define LAB3_z2_IO_H_
3     typedef  int data_sc;
4     #define N 256
5     #define M 16
6     void lab3_z2(data_sc D_I[N], data_sc* C_I, data_sc D_O[N]);
7 #endif
```

Рис. 1 Определение синтезируемой функции

Исходный код синтезируемой функции с обычным считыванием представлен на рисунке 2.

```
1 #include "lab3_z2.h"
2
3 void lab3_z2(data_sc D_I[N], data_sc* C_I, data_sc D_O[N])
4 {
5     for(int i = 0; i < N; ++i){
6         data_sc macc = D_I[i];
7         for(int j = 0; j < M; ++j){
8             macc *= C_I[j];
9         }
10        D_O[i] = macc;
11    }
12 }
```

Рис. 2. Исходный синтезируемой функции с обычным копированием

Исходный код синтезируемой функции с memcpy считыванием представлен на рисунке 3.

```
1 #include <string.h>
2 #include "lab3_z2.h"
3
4
5 void lab3_z2(data_sc D_I[N], data_sc* C_I, data_sc D_O[N])
6 {
7     data_sc buff[M];
8     memcpy(buff, C_I, M * sizeof(data_sc));
9
10    for(int i = 0; i < N; ++i){
11        data_sc macc = D_I[i];
12        for(int j = 0; j < M; ++j){
13            macc *= buff[j];
14        }
15        D_O[i] = macc;
16    }
17 }
```

Рис. 3 Исходный синтезируемой функции с memcpy копированием

3. Исходный код теста

Исходный код теста проверки функции lab3_z2 приведен на рисунке 4.

Тест обеспечивает проверку корректной работы функции.

4. Командный файл

На рисунке 5 представлен текст команд для автоматизированного создания следующих вариантов аппаратной реализации:

a. Для sol задается clock period 6: clock uncertainty 0.1 и подключается файл lab3_z2s.c

b. Для sol задается clock period 6: clock uncertainty 0.1 и подключается файл lab3_z2b.c

Также устанавливается директива интерфейса для указателя C_I — ap_bus с параметром depth = 1.

```
12 int cmp_arr(data_sc D_I[N], data_sc C_I[M], data_sc D_O[N])
13 {
14     data_sc cmp_buff[N];
15     for(int i = 0; i < N; ++i){
16         cmp_buff[i] = D_I[i];
17         for(int j = 0; j < M; ++j){
18             cmp_buff[i] *= C_I[j];
19         }
20         if (cmp_buff[i] != D_O[i]){
21             return 0;
22         }
23     }
24     return 1;
25 }
26
27 int main()
28 {
29     int pass=0;
30
31     // Call the function for 2 transactions
32     data_sc D_I[N];
33     data_sc C_I[M];
34     data_sc D_O[N];
35
36     for (int i = 0; i < 2; ++i){
37         set_random(D_I, N);
38         set_random(C_I, M);
39
40         lab3_z2(D_I, C_I, D_O);
41         pass = cmp_arr(D_I, C_I, D_O);
42         if (pass == 0){break;}
43     }
```

Рис. 4 Исходный код тестирования синтезируемой функции

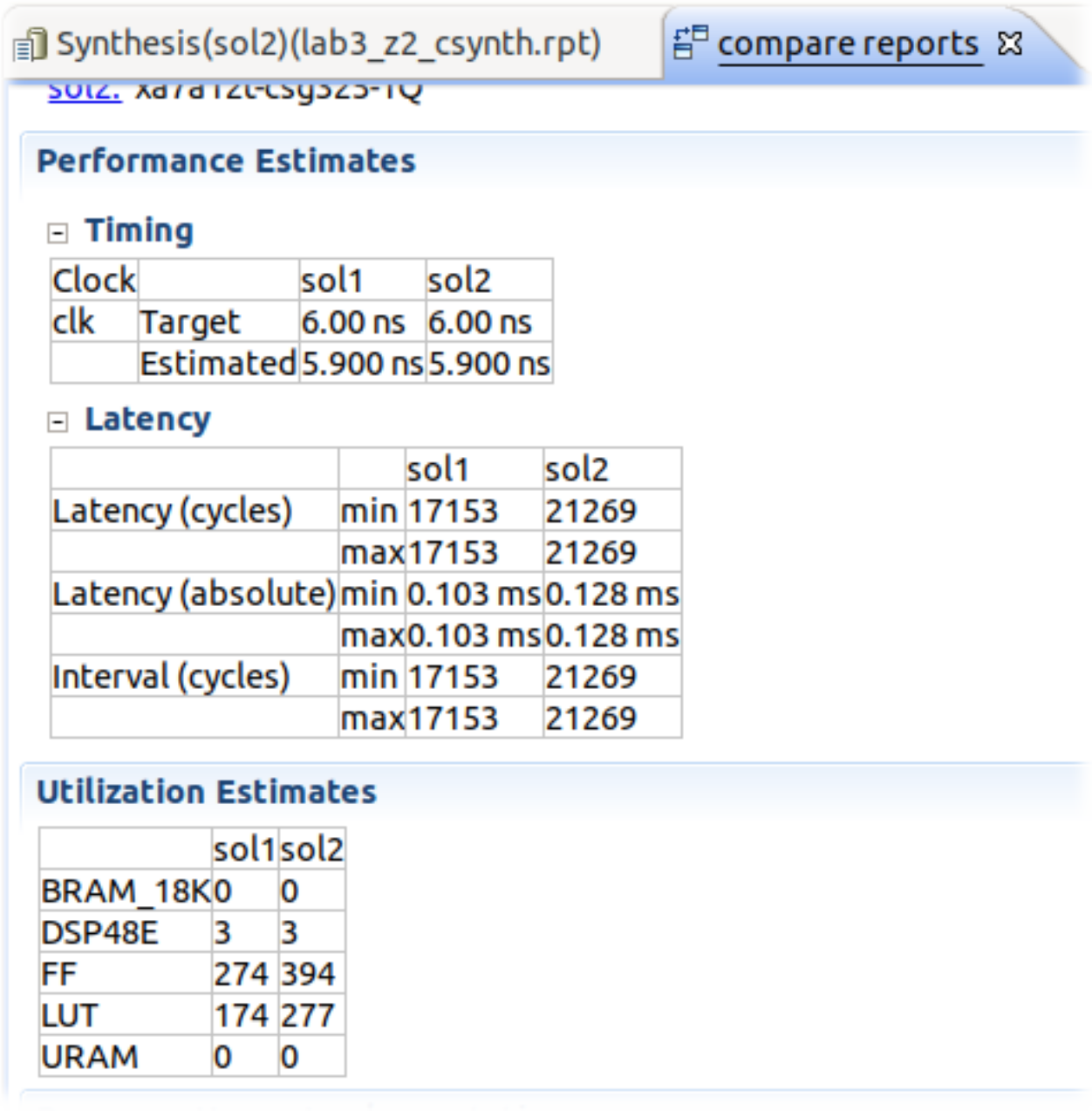
```
# Insert the command to add design file
add_files -tb ./source/lab3_z2_test.c

# Insert the command to specify the top-level function
set_top lab3_z2

# Insert the command to add testbench file


# Insert the command to create the solution named sol1
open_solution -reset "sol1"
add_files ./source/lab3_z2s.c

set_part {xa7a12tcsg325-1Q}
create_clock -period 6 -name clk
set_clock_uncertainty 0.1

set_directive_interface -depth 1 -mode ap_bus "lab3_z2" C_I
csim_design -clean
csynth_design


# Insert the command to create the solution named sol2
open_solution -reset "sol2"
#remove_files lab3_z2s.c
add_files ./source/lab3_z2b.c

set_part {xa7a12tcsg325-1Q}
create_clock -period 6 -name clk
set_clock_uncertainty 0.1

set_directive_interface -depth 1 -mode ap_bus "lab3_z2" C_I
csim_design -clean
csynth_design

exit
```

Рис. 5 Текст команд для создания решений

6. Результаты исследований

6.1 Сравнение решений

На рисунке 6 представлено сравнение из Vivado HLS GUI по аппаратным ресурсам и временным параметрам. На рисунке видно, что Estimated Time = 5,9 нс для обеих решений, Latency(cycle) = 17153 итерации для sol1 и 21269 итерации для sol2. По аппаратным ресурсам видно, что оба решения требуют 3 DSP48E модуля и для sol1 требуется 274 триггера FF и 174 LUT, а для sol2 394 триггера FF и 277 LUT.



Clock		sol1	sol2
clk	Target	6.00 ns	6.00 ns
	Estimated	5.900 ns	5.900 ns

		sol1	sol2
Latency (cycles)	min	17153	21269
	max	17153	21269
Latency (absolute)	min	0.103 ms	0.128 ms
	max	0.103 ms	0.128 ms
Interval (cycles)	min	17153	21269
	max	17153	21269

	sol1	sol2
BRAM_18K	0	0
DSP48E	3	3
FF	274	394
LUT	174	277
URAM	0	0

Рис. 6 Сравнение решений

6.2 Электронная таблица и график решений

На рисунке 7 представлена таблица с параметрами для двух решений. На рисунке 8 представлен график для сравнения двух решений.

		sol1	sol2
Clock	Target (ns)	6	6
	Estimated (ns)	5.9	5.9
Latency	(cycles)	17153	21269
	(ns)	102202	125487
Resources	BRAM_18K	0	0
	DSP48E	3	3
	FF	274	394
	LUT	174	277
	URAM	0	0

Рис. 7 Таблица данных для решений

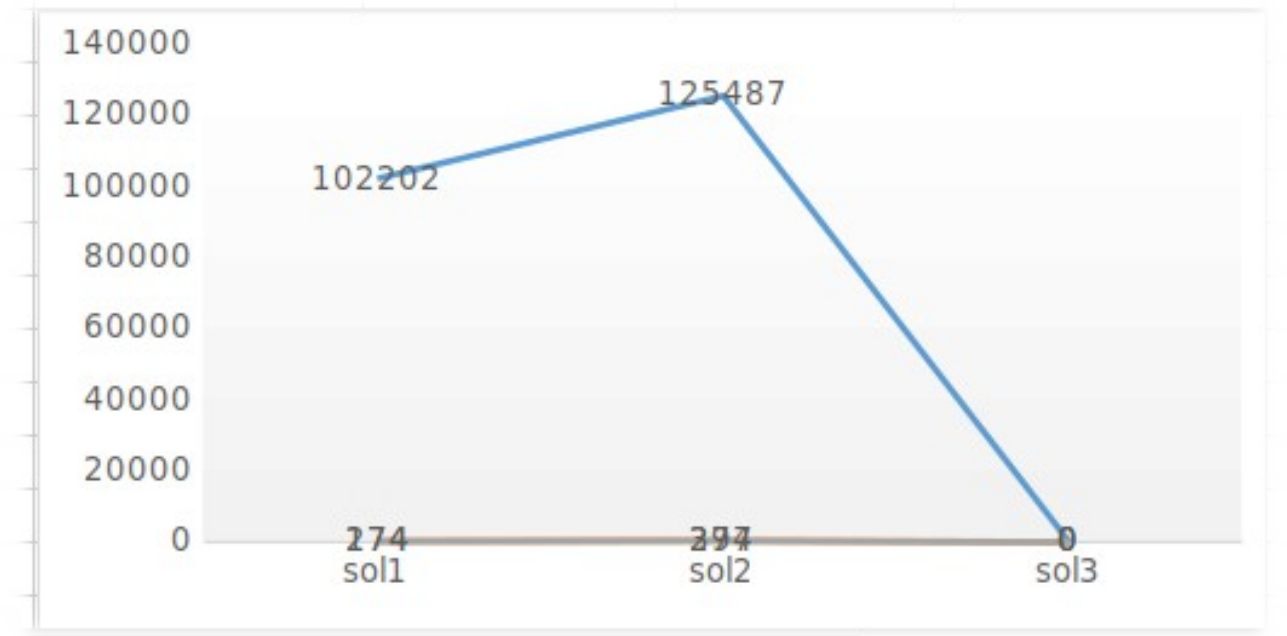


Рис. 8 График данных для решений

Как видно из таблицы и графика второе решение затрачивает большее время и большее количество почти всех аппаратных ресурсов, следовательно первое решение является лучшим по сравнению со вторым.

Вывод

В данной работе был произведен сравнительный анализ между решением с обычным копированием данных и копированием с использованием функции memcpy из стандартной библиотеки. Как видно по результатам решением с memcpy является менее предпочтительным, так как проигрывает по все показателям по сравнению с решением без него.