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

ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ

Дисциплина: Проектирование реконфигурируемых гибридных вычислительных систем

Вариант - 8

Выполнил студент гр. 02001 Дроздов Н.Д.
Руководитель, доцент Антонов А.П.

«23» декабря 2021

Санкт-Петербург
2021

Оглавление

Список иллюстраций

Задание

Даны матрицы A и B, квадратные, размер NxN. Тип элементов int

Найти их произведение и транспонировать. В тесте, для проверки, использовать свойство $(AB)^T=B^T*A^T$

Исследование включает:

- оптимизацию аппаратной реализации и оценку производительности,
- анализ производительности программной реализации на ПК,
- сравнительный анализ программной и аппаратной реализаций.

Оптимизация аппаратной реализации и оценка ее производительности:

- Параметр N и тип данных должны быть определены в include файле
- Оптимизация аппаратной реализации и оценка ее производительности осуществляется в пакете Vivado HLS.
- Способ оценки производительности: II (число тактов) * Estimated time period
- В тесте, в рамках пакета Vivado HLS, исследуемую функцию следует запускать не менее двух раз.
- Заполнение матриц в тестах осуществлять случайными числами.
- Тест должен быть с самопроверкой результата (обязательно предусмотреть и отразить в записке проверку теста – при неправильных данных должен формировать ошибку).
- Оптимизация аппаратной реализации осуществлять для N=8192.
- Цель оптимизации – максимальная производительность. Ограничения – доступные аппаратные ресурсы микросхемы xa7a100tfgg484-2i
- Все варианты, созданные при исследовании производительности должны быть представлены в проекте и описаны в пояснительной записке. Должно быть приведено сравнение результатов (+заполнена xls таблица и приведены графики зависимости производительности и аппаратных затрат для созданных вариантов реализаций)

- Следует обосновать выбор директив для оптимизации.
- Следует обосновать достижения максимума производительности (невозможность дальнейшего увеличения производительности без превышения ограничений по аппаратным затратам; или обосновать особенностью алгоритма)
- Оценку производительности аппаратной реализации осуществить для выбранного, оптимального решения, при N=8192, 32768, 131072

Анализ производительности программной реализации на ПК:

- Для тестирования производительности программной реализации синтезируемой функции необходимо создать:
  - отдельный, модернизированный, тест для проверки времени выполнения синтезируемой функции на ПК:
  - добавить в тест операторы измерения времени выполнения синтезируемой функции
  - Увеличить количество запусков синтезируемой функции до 32. Для каждого запуска измерить время, найти среднее значение и вывести как результат.
  - Точность измерения времени (наносекунды).
  - Тестирование проводить при N=8192, 32768, 131072
- Провести исследование времени выполнения синтезируемой функции на Вашем ПК
- Осуществить компиляцию модернизированного теста и запустить его как отдельное приложение при N=8192, 32768, 131072
- В отчете привести:
  - Параметры Вашего ПК: тип процессора, частота работы процессора, объем ОЗУ
  - результаты измерения времени выполнения при N=8192, 32768, 131072

Сравнительный анализ программной и аппаратной реализаций

- Привести табличные данные по производительности аппаратной и программной реализации (при N=8192, 32768, 131072), табличные данные для аппаратных затрат.
- Привести графики зависимости производительности (времени выполнения) аппаратной и программной реализации при N=8192, 32768, 131072. (По возможности совмещенные данные, можно использовать логарифмический масштаб).
- Сделать вывод

Исходный код функции

Исходный код синтезируемой функции *kurs* приведен на рисунке ниже.

Функция принимает 3 аргумента массива типа int — вычисляет произведение двух матриц, после чего транспонирует полученную матрицу.

```
1   #include "kurs.h"
2
3   void kursFunc(data_sc inA[N][N], data_sc inB[N][N], data_sc outC[N][N]) {
4
5       L0: for (int i = 0; i < N; i++) {
6           L1: for (int j = 0; j < N; j++) {
7                   outC[i][j] = 0;
8                   L2: for (int k = 0; k < N; k++) {
9                       outC[i][j] += inA[i][k] * inB[k][j];
10                  }
11          }
12      }
13
14      L3: for (int i2 = 0; i2 < N; i2++) {
15              L4: for (int j2 = i2 + 1; j2 < N; j2++) {
16                  int temp = outC[i2][j2];
17                  outC[i2][j2] = outC[j2][i2];
18                  outC[j2][i2] = temp;
19          }
20      }
21  }
```

Рисунок 1 - Исходный код функции (файл kurs.c)

Исходный код теста

Исходный код теста для проверки функции *kurs* приведен на рисунке ниже. Для проверки результата была написана проверочная функция, в которой сначала транспонируются входные матрицы, а уже после этого

вычисляется их произведение и сравнивается с полученной ранее выходной матрицей.

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

int func_T(data_sc inArr[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = i + 1; j < N; j++) {
            int temp = inArr[i][j];
            inArr[i][j] = inArr[j][i];
            inArr[j][i] = temp;
        }
    }
}

int cmp_arr(data_sc inA[N][N], data_sc inB[N][N], data_sc outC[N][N]) {

    data_sc BTAT[N][N];

    func_T(inB);
    func_T(inA);

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            BTAT[i][j] = 0;
            for (int k = 0; k < N; k++) {
                BTAT[i][j] += inB[i][k] * inA[k][j];
            }
        }
    }

    for(int i2 = 0; i2 < N; i2++) {
        for(int j2 = 0; j2 < N; j2++) {
            if (outC[i2][j2] != BTAT[i2][j2]) {
                return 0;
            }
        }
    }

    return 1;
}
```

```
40
41  int main() {
42      int pass = 0;
43
44      data_sc inA[N][N];
45      data_sc inB[N][N];
46      data_sc outC[N][N];
47
48      for (int i = 0; i < 3; i++) {
49
50          //set inA inB inC
51          for (int j = 0; j < N; j++) {
52              for (int k = 0; k < N; k++) {
53                  inA[j][k] = rand() % (N - 1);
54                  inB[j][k] = rand() % (N - 1);
55              }
56          }
57
58          kursFunc(inA, inB, outC);
59          pass = cmp_arr(inA, inB, outC);
60      }
61
62      if (pass == 1) {
63          fprintf(stdout, "----------Pass!-----------\n");
64          return 0;
65      } else {
66          fprintf(stderr, "----------Fail!-----------\n");
67          return 1;
68      }
69  }
70
```

Рисунок 2 - Исходный код теста (файл kurs_test.c)

Исходный код командного файла

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

a. Для sol1 задается clock period 6: clock uncertainty 0.1
b. Для sol2 задается clock period 8. clock uncertainty 0.1
c. Для sol3 задается clock period 10. clock uncertainty 0.1
d. Для sol4 задается clock period 12. clock uncertainty 0.1
e. Для sol5 задается clock period 16. clock uncertainty 0.1

```
##############################################
#                    Lab                     #
##############################################
open_project -reset kr_02001_8
set_top kursFunc
add_files ./source/kurs.c
add_files -tb ./source/kurs_test.c

open_solution -reset sol1
create_clock -period 6 -name clk
set_clock_uncertainty 0.1
set_part {xa7a100tfgg484-2i}

csim_design
csynth_design
#cosim_design -trace_level all
##############################################
#                  Solutions                 #
##############################################
set all_solutions {sol2 sol3 sol4 sol5}
set all_periods   {{8}  {10} {12} {16}}

foreach the_solution $all_solutions the_period $all_periods {
open_solution -reset $the_solution
create_clock -period $the_period -name clk
set_clock_uncertainty 0.1
set_part {xa7a100tfgg484-2i}

csim_design
csynth_design
#cosim_design -trace_level all
}

exit
```

Рисунок 3 - Исходный код командного файла для создания проекта (файл kurs.tcl)

Результаты исследования и сравнение решений

На рисунке ниже представлено сравнение из Vivado HLS GUI по аппаратным ресурсам, требуемых для реализации синтезируемой функции, и временным параметрам.

All Compared Solutions

sol1: xa7a100tfgg484-2i
sol2: xa7a100tfgg484-2i
sol3: xa7a100tfgg484-2i
sol4: xa7a100tfgg484-2i
sol5: xa7a100tfgg484-2i

Performance Estimates

Timing (ns)

Clock		sol1	sol2	sol3	sol4	sol5
clk	Target	6.00	8.00	10.00	12.00	16.00
	Estimated	5.733	7.047	9.652	11.832	11.832

Latency (clock cycles)

		sol1	sol2	sol3	sol4	sol5
Latency	min	4398247870466	3848492056578	3298736242690	2748980428802	2748980428802
	max	4398650474498	3848827559938	3299071746050	2749315932162	2749315932162
Interval	min	4398247870466	3848492056578	3298736242690	2748980428802	2748980428802
	max	4398650474498	3848827559938	3299071746050	2749315932162	2749315932162

Utilization Estimates

	sol1	sol2	sol3	sol4	sol5
BRAM_18K	0	0	0	0	0
DSP48E	3	3	3	3	3
FF	624	457	392	359	359
LUT	655	618	598	595	595

Рисунок 4 - сравнение результатов

Target – планируемое время на один такт.

Estimated – оценочное время.

Latency (cycle) – количество тактов latency за один цикл.

Latency (absolute) – время затраченное на latency.

Анализ результатов

На рисунке ниже представлена таблица с параметрами для всех решений, где рассчитывается Latency в нс.

		sol1	sol2	sol3	sol4	sol5
Clock	Target (ns)	6	8	10	12	16
	Estimated (ns)	5,73	7,05	9,65	11,83	11,83
Latency	(cycles)	4398650474498	3848827559938	3299071746050	2749315932162	2749315932162
	(ns)	25217463170297	27134234297563	31842640492875	32524407477477	32524407477477
Resources	BRAM_18K	0	0	0	0	0
	DSP48E	3	3	3	3	3
	FF	538	443	277	244	212
	LUT	429	424	392	389	389
	URAM					

Рисунок 5 - Таблица с параметрами и график для сравнения всех решений

На рисунке 5 представлен график данных для сравнения всех решений.

Исходя из результатов, видно, что наилучшим решением является 3 решение, так как у него среднее время и средние затраты по ресурсам среди всех остальных решений.

На основе него и производилась оптимизация

Оптимизация, 1 вариант

Были использованы следующие директивы

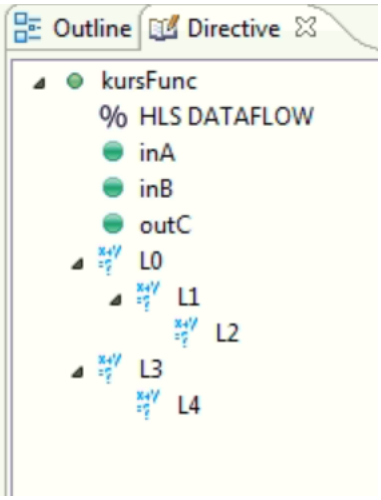


Рисунок 6 - Использованные директивы для выбранного решения

General Information

Date:	Fri Dec 24 01:27:46 2021
Version:	2018.3 (Build 2405991 on Thu Dec 06 23:56:15 MST 2018)
Project:	kr_02001_8
Solution:	sol2
Product family:	aartix7
Target device:	xa7a100tfgg484-2i

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
clk	10.00	9.652	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
201359362	536862722	3298736242691	3299071746051	dataflow

Рисунок 7 - Анализ полученного решения с оптимизацией

Оптимизация, 2 вариант

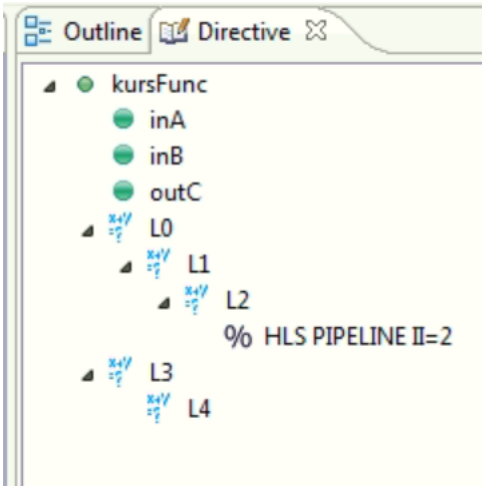
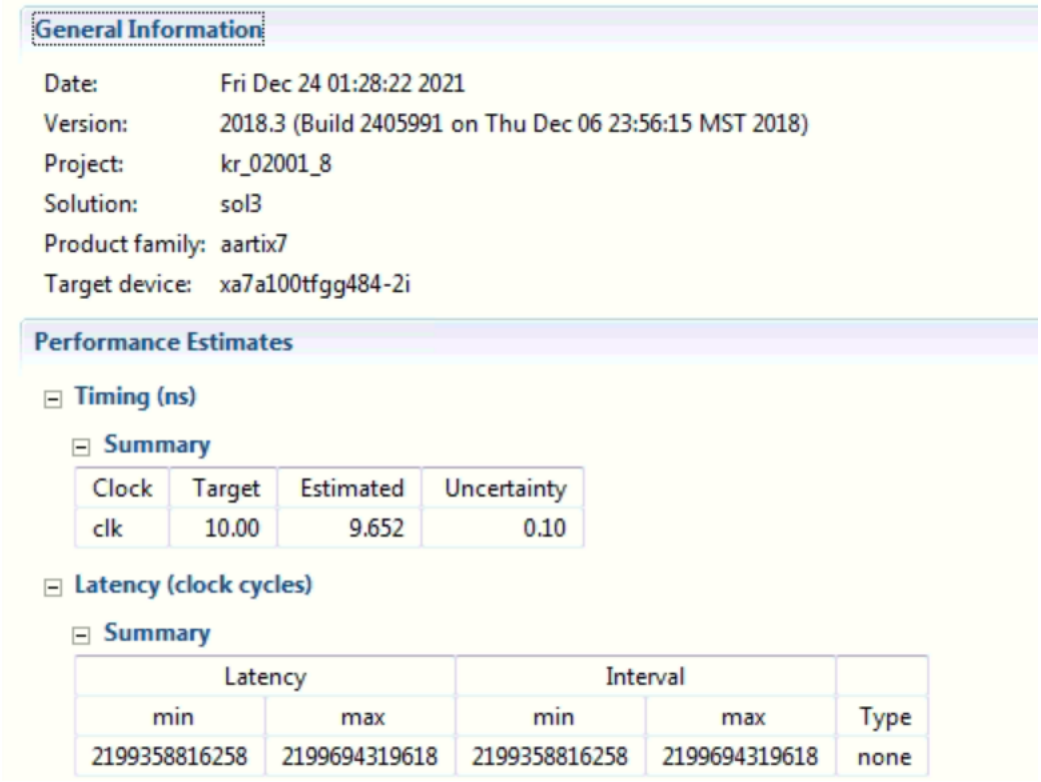


Рисунок 8 - Использованные директивы для выбранного решения



General Information

Date:	Fri Dec 24 01:28:22 2021
Version:	2018.3 (Build 2405991 on Thu Dec 06 23:56:15 MST 2018)
Project:	kr_02001_8
Solution:	sol3
Product family:	aartix7
Target device:	xa7a100tfgg484-2i

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
clk	10.00	9.652	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
2199358816258	2199694319618	2199358816258	2199694319618	none

Рисунок 9 - Анализ полученного решения с оптимизацией

Оптимизация, 3 вариант

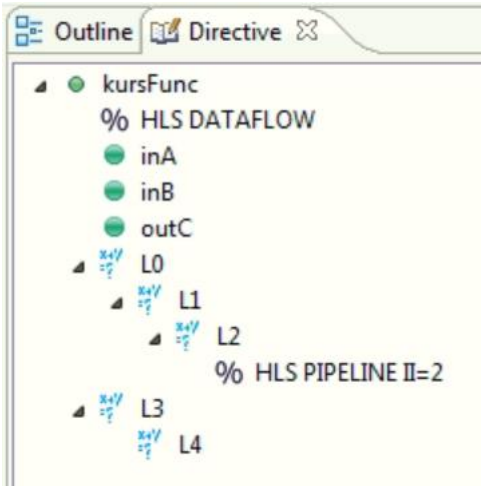


Рисунок 10 - Использованные директивы для выбранного решения

General Information

Date:	Fri Dec 24 01:32:16 2021
Version:	2018.3 (Build 2405991 on Thu Dec 06 23:56:15 MST 2018)
Project:	kr_02001_8
Solution:	sol4
Product family:	aartix7
Target device:	xa7a100tfgg484-2i

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
clk	10.00	9.652	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
335560706	671064066	2199358816259	2199694319619	dataflow

Рисунок 11 - Анализ полученного решения с оптимизацией

Оптимизация, 4 вариант

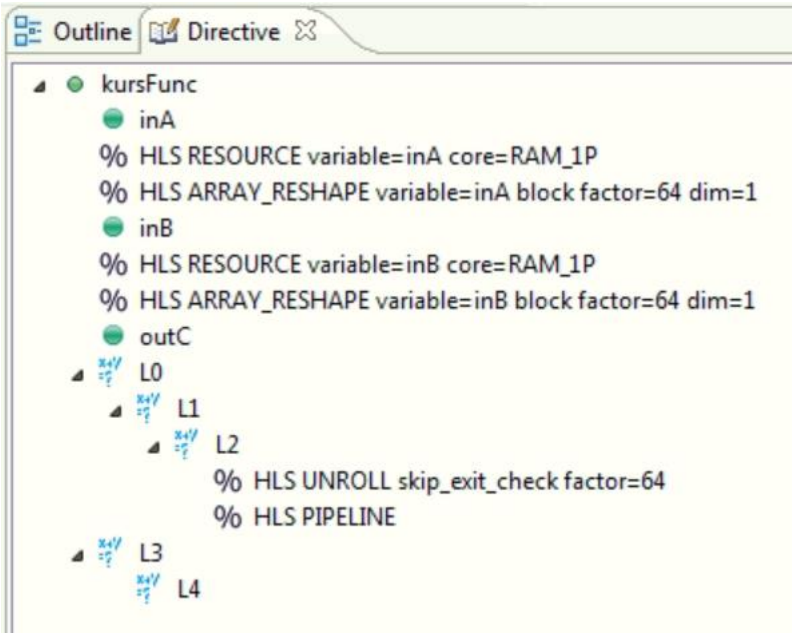


Рисунок 12 - Использованные директивы для выбранного решения

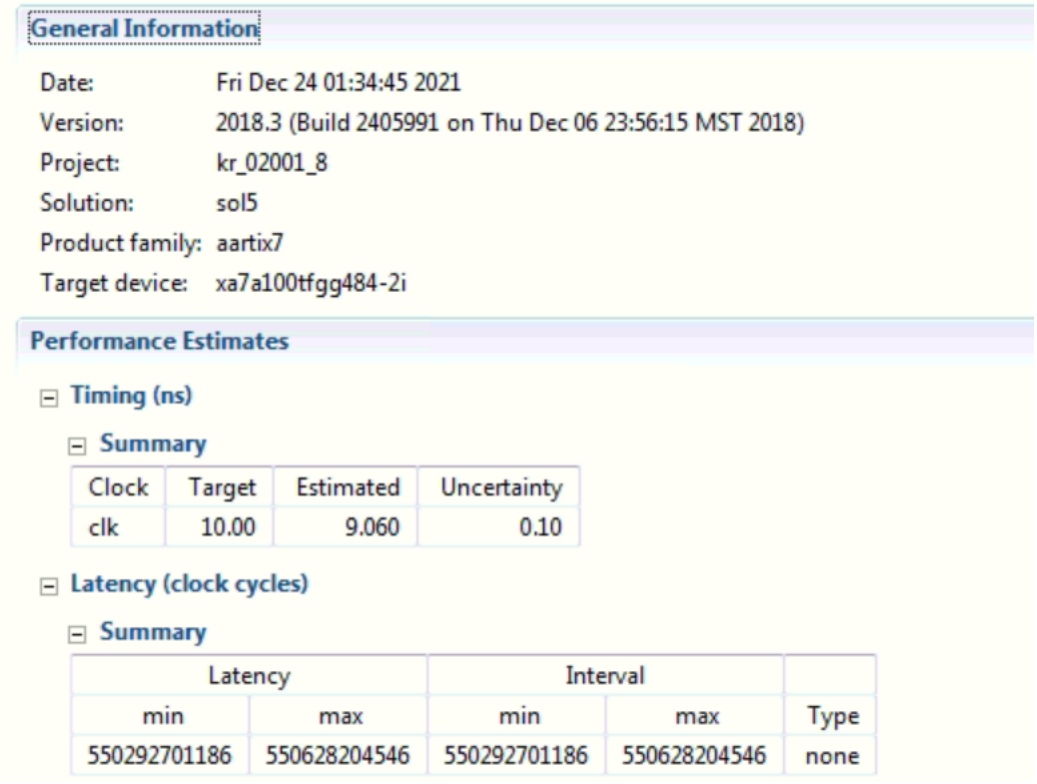


Рисунок 13 - Анализ полученного решения с оптимизацией

Оптимизация, 5 вариант

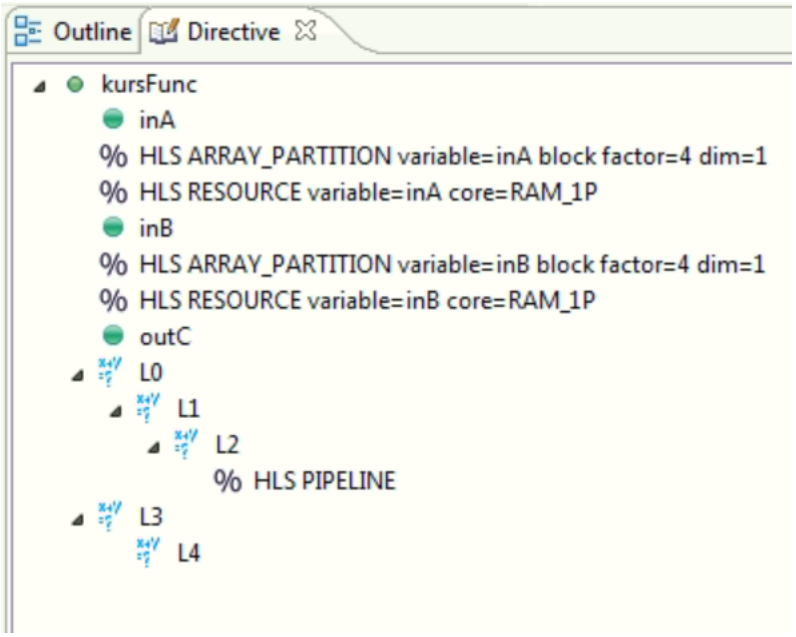


Рисунок 14 - Использованные директивы для выбранного решения

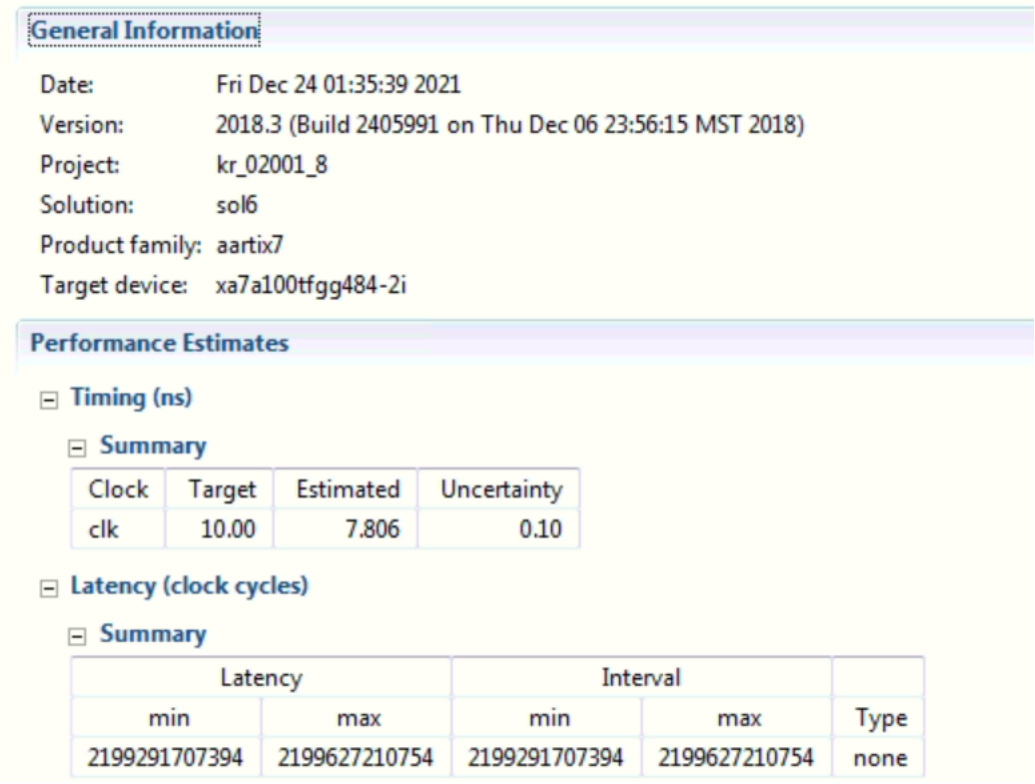


Рисунок 15 - Анализ полученного решения с оптимизацией

Оптимизация, 6 вариант

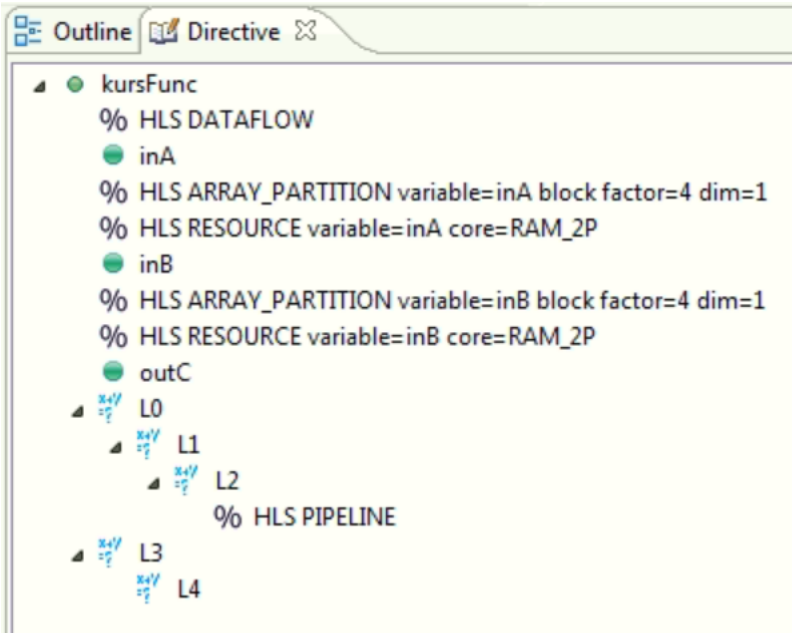


Рисунок 16 - Использованные директивы для выбранного решения

General Information

Date:	Fri Dec 24 01:36:11 2021
Version:	2018.3 (Build 2405991 on Thu Dec 06 23:56:15 MST 2018)
Project:	kr_02001_8
Solution:	sol7
Product family:	aartix7
Target device:	xa7a100tfgg484-2i

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
clk	10.00	7.806	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
268451842	603955202	2199291707395	2199627210755	dataflow

Рисунок 17 - Анализ полученного решения с оптимизацией

Общий анализ всех полученных решений

Performance Estimates

Timing (ns)

Clock		sol1	sol2	sol3	sol4	sol5	sol6	sol7
clk	Target	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	Estimated	9.652	9.652	9.652	9.652	9.060	7.806	7.806

Latency (clock cycles)

		sol1	sol2	sol3	sol4	sol5	sol6	sol7
Latency	min	3298736242690	201359362	2199358816258	335560706	550292701186	2199291707394	268451842
	max	3299071746050	536862722	2199694319618	671064066	550628204546	2199627210754	603955202
Interval	min	3298736242690	3298736242691	2199358816258	2199358816259	550292701186	2199291707394	2199291707395
	max	3299071746050	3299071746051	2199694319618	2199694319619	550628204546	2199627210754	2199627210755

Utilization Estimates

	sol1	sol2	sol3	sol4	sol5	sol6	sol7
BRAM_18K	0	0	0	0	0	0	0
DSP48E	3	3	3	3	192	3	3
FF	392	393	416	417	7215	497	498
LUT	598	615	700	717	708675	728	745

Resource Usage Implementation

Рисунок 18 - Общий анализ полученных решений

Также график, для сравнения (при N = 8192) всех решений (некоторые значения были переназначены на число, близкое к нулю для того, чтобы можно было видеть разницу самых эффективных решений)

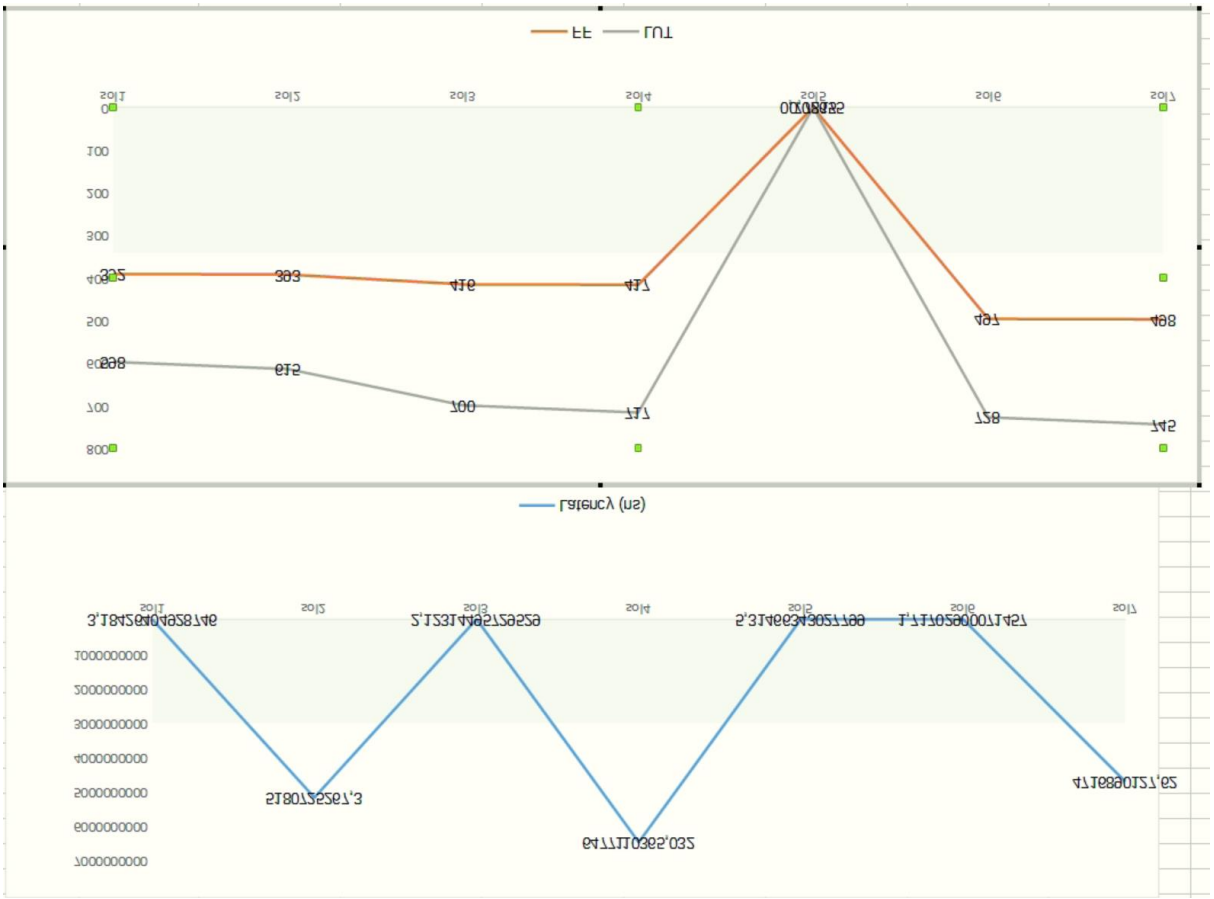


Рисунок 19 - Графики для сравнения полученных решений с оптимизацией

Из графиков можно сделать вывод, что оптимальным решением (при N=8192) является sol7, так как время является наилучшим, а аппаратные затраты не сильно отличаются от остальных решений.

При N = 32768

Performance Estimates

Timing (ns)

Clock		sol1	sol2	sol3	sol4	sol5	sol6	sol7
clk	Target	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	Estimated	9.652	9.652	9.652	9.652	9.060	8.058	8.058

Latency (clock cycles)

		sol1	sol2	sol3	sol4	sol5	sol6	sol7
Latency	min	211109453889538	?	140742857129986	1073807362	35192962088962	140741783388162	65538
	max	211114822434818	?	140748225675266	2147385346	35198330634242	140747151933442	1073643522
Interval	min	211109453889538	211109453889539	140742857129986	140742857129987	35192962088962	140741783388162	140741783388163
	max	211114822434818	211114822434819	140748225675266	140748225675267	35198330634242	140747151933442	140747151933443

Utilization Estimates

	sol1	sol2	sol3	sol4	sol5	sol6	sol7
BRAM_18K	0	0	0	0	0	0	0
DSP48E	3	3	3	3	192	3	3
FF	428	429	460	461	7269	543	544
LUT	646	663	751	768	709102	779	796

Рисунок 20 - Результат аппаратных решений при N = 32768

При N = 131072

Performance Estimates

Timing (ns)

Clock		sol1	sol2	sol3	sol4	sol5	sol6	sol7
clk	Target	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	Estimated	8.635	8.635	8.635	8.635	9.060	6.880	6.880

Latency (clock cycles)

		sol1	sol2	sol3	sol4	sol5	sol6	sol7
Latency	min	6755433801318402	524290	4503651167240194	262146	2251902893162498	4503668347109378	262146
	max	6755468160794626	262146	4503685526716418	2	2251937252638722	4503702706585602	2
Interval	min	6755433801318402	6755433801318403	4503651167240194	4503651167240195	2251902893162498	4503668347109378	4503668347109379
	max	6755468160794626	6755468160794627	4503685526716418	4503685526716419	2251937252638722	4503702706585602	4503702706585603

Utilization Estimates

	sol1	sol2	sol3	sol4	sol5	sol6	sol7
BRAM_18K	0	0	0	0	0	0	0
DSP48E	3	3	3	3	192	3	3
FF	354	355	358	359	7201	465	466
LUT	445	462	509	526	709376	595	612

Рисунок 21 - Результат аппаратных решений при N = 131072

Анализ производительности программной реализации на ПК

Оценка производилась при N = 1024.

Было выбрано N = 1024, так как при увеличении N результаты приходится ждать слишком долго.

На рисунке представлен исходный код модифицированного теста для ПК. Тест обеспечивает проверку производительности функции на ПК (Компилятор gcc-9.3.0).

Рисунок 22 - Параметры ПК (Частота = 2.7 Гц)

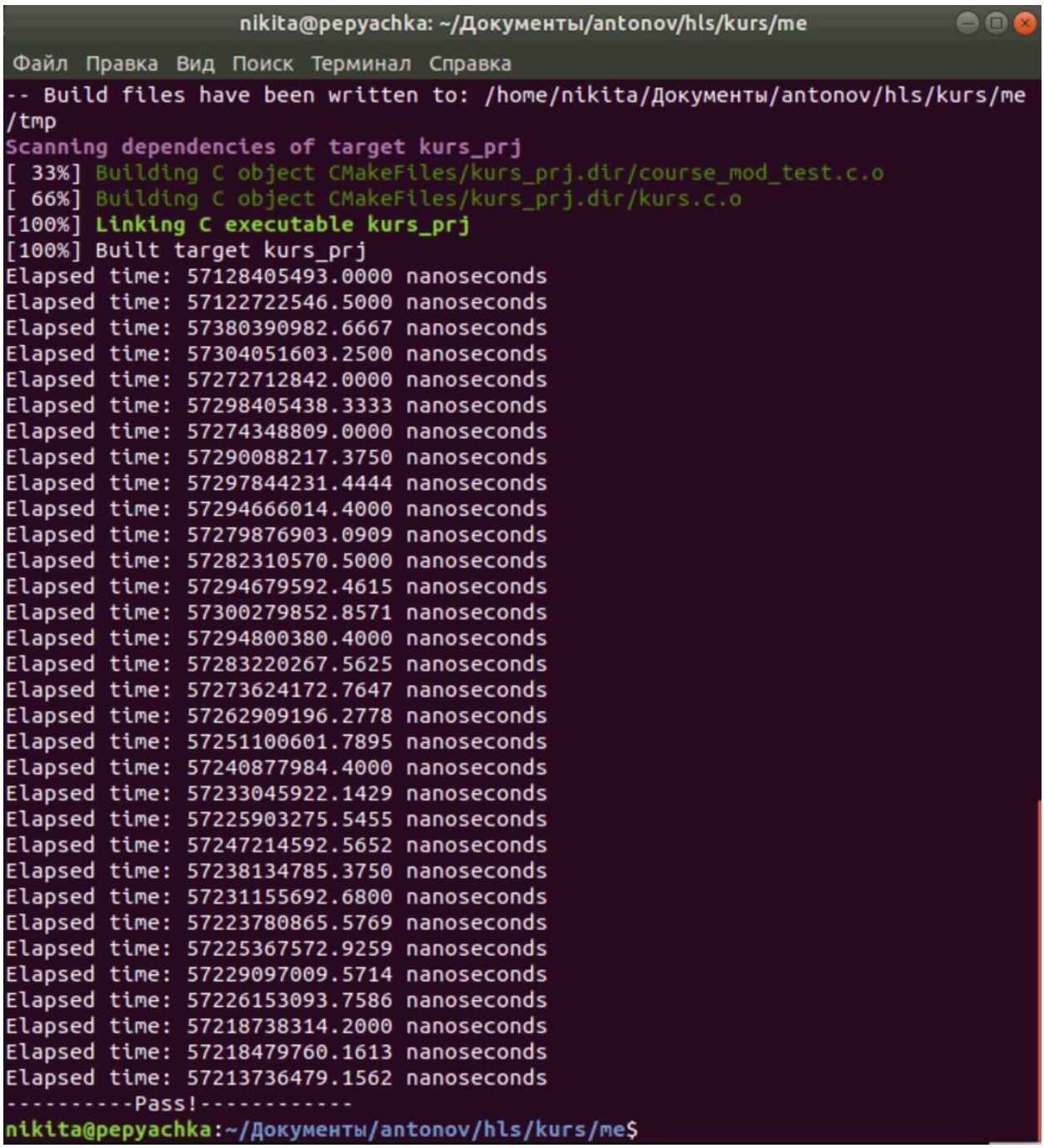


Рисунок 23 - Результат запуска модифицированного теста

		sol1	sol2	sol3	sol4	sol5	sol6	sol7	PC
Clock	Target (ns)	10	10	10	10	10	10	10	
	Estimated (ns)	9,65	9,65	9,65	9,65	9,61	7,72	7,72	
Latency	(cycles)	6445600770	?	4300212226	5244930	1082132482	4299163650	4196354	
	(ns)	62212938632	#ЗНАЧ!	41505648405	50624064	10403621682	33206740033	32412638	57304051603
Resources	BRAM_18K	0	0	0	0	0	0	0	
	DSP48E	3	3	3	3	192	3	3	
	FF	311	312	319	320	244106	387	388	
	LUT	524	541	610	627	707074	638	655	
	URAM								

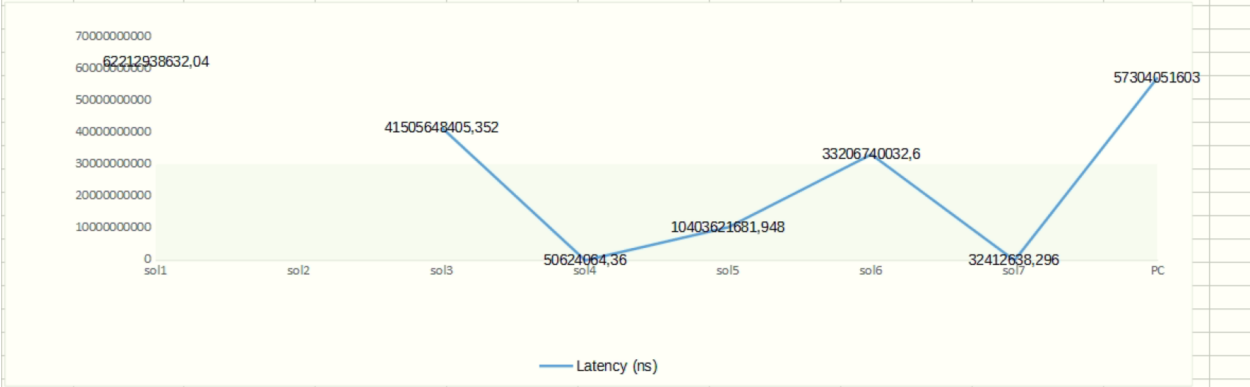
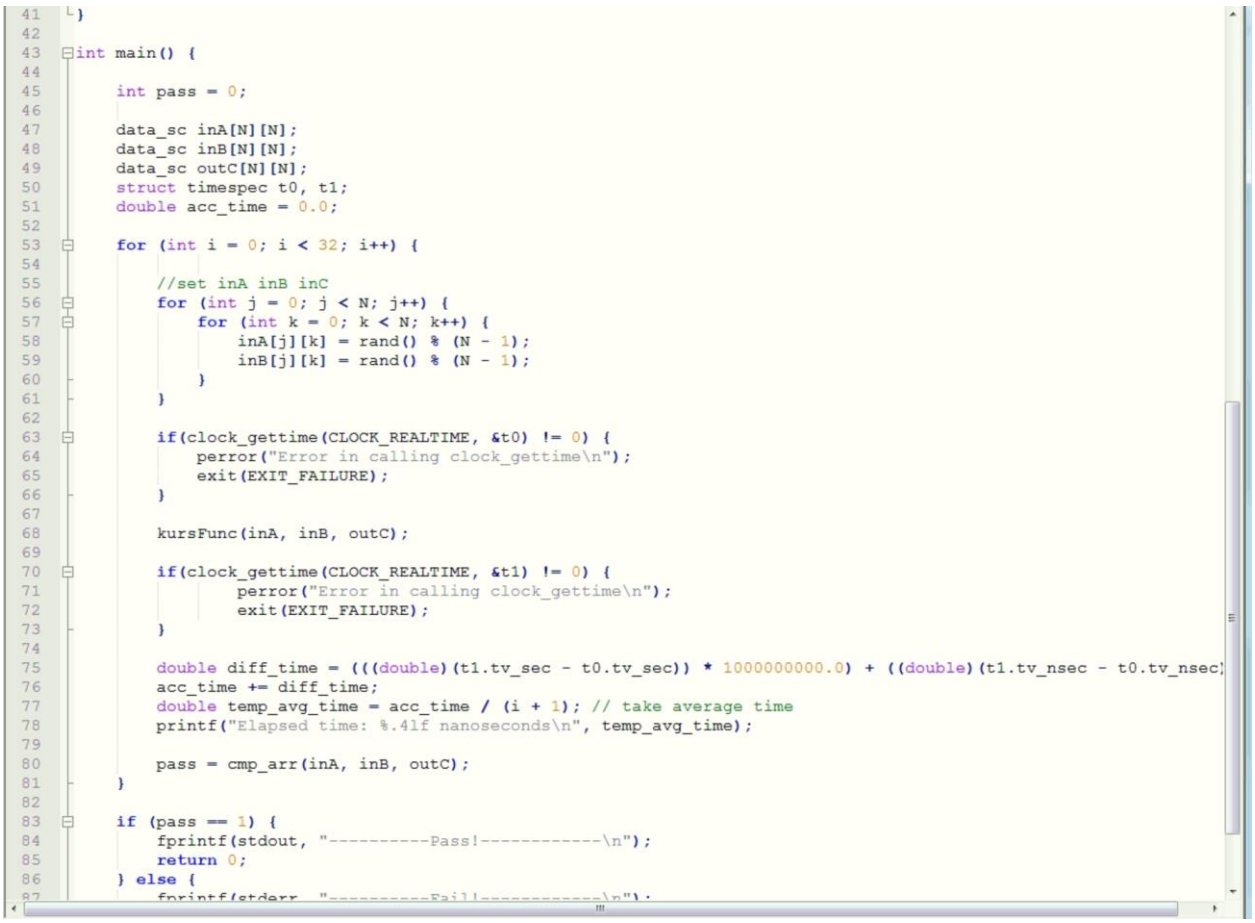


Рисунок 24 - Сравнение аппаратных решений с программных решением

На рисунке представлен график для сравнения аппаратных решений с программным решением (при N = 1024). Исходя из графика можно сделать вывод, что программное решение ненамного лучше аппаратное решения без каких-либо оптимизаций, но в несколько раз хуже, чем самое эффективное аппаратное решение.



```
}

int main() {

    int pass = 0;

    data_sc inA[N][N];
    data_sc inB[N][N];
    data_sc outC[N][N];
    struct timespec t0, t1;
    double acc_time = 0.0;

    for (int i = 0; i < 32; i++) {

        //set inA inB inC
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
                inA[j][k] = rand() % (N - 1);
                inB[j][k] = rand() % (N - 1);
            }
        }

        if(clock_gettime(CLOCK_REALTIME, &t0) != 0) {
            perror("Error in calling clock_gettime\n");
            exit(EXIT_FAILURE);
        }

        kursFunc(inA, inB, outC);

        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(inA, inB, outC);
    }

    if (pass == 1) {
        fprintf(stdout, "----------Pass!----------\n");
        return 0;
    } else {
        fprintf(stderr, "----------Fail!----------\n");
```

Рисунок 25 - Исходный код модернизированного теста

На рисунке представлен исходный код модернизированного теста.

Вывод

В ходе выполнения курсовой работы, были применены знания, полученные на лабораторных работах. Были написаны исходный код для заданного задания, исходный код теста для него, а также tcl файл для первоначального запуска программы.

После чего было выявлено лучшее решение, на основе которого произвели оптимизацию. Были использованы изученные ранее директивы для оптимизации и было выбрано лучшее решение из предложенных.

Осуществили запуск аппаратных и программного решений и сравнили полученные результаты. Из результатов видно , что решение, полученное на ПК, медленнее, чем решение, полученное аппаратным путем с оптимизацией в Vivado HLS.

Приложение

```
#include "kurs.h"

void kursFunc(data_sc inA[N][N], data_sc inB[N][N], data_sc outC[N][N]) {

	L0: for (int i = 0; i < N; i++) {
		L1: for (int j = 0; j < N; j++) {
			outC[i][j] = 0;
			L2: for (int k = 0; k < N; k++) {
			outC[i][j] += inA[i][k] * inB[k][j];
			}
	}
	}

	L3: for (int i2 = 0; i2 < N; i2++) {
	L4: for (int j2 = i2 + 1; j2 < N; j2++) {
			int temp = outC[i2][j2];
			outC[i2][j2] = outC[j2][i2];
			outC[j2][i2] = temp;
	}
	}
}
```

Листинг исходного кода

```
#include "kurs.h"
#include <stdio.h>

int func_T(data_sc inArr[N][N]) {
	for (int i = 0; i < N; i++) {
	for (int j = i + 1; j < N; j++) {
			int temp = inArr[i][j];
			inArr[i][j] = inArr[j][i];
			inArr[j][i] = temp;
	}
	}
}

int cmp_arr(data_sc inA[N][N], data_sc inB[N][N], data_sc outC[N][N]) {

	data_sc BTAT[N][N];

	func_T(inB);
	func_T(inA);
```

```
        for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
                BTAT[i][j] = 0;
                        for (int k = 0; k < N; k++) {
                        BTAT[i][j] += inB[i][k] * inA[k][j];
                        }
        }
        }

        for(int i2 = 0; i2 < N; i2++) {
                for(int j2 = 0; j2 < N; j2++) {
                                if (outC[i2][j2] != BTAT[i2][j2]) {
                                        return 0;
                        }
                }
        }

        return 1;
}

int main() {
        int pass = 0;

        data_sc inA[N][N];
        data_sc inB[N][N];
        data_sc outC[N][N];

        for (int i = 0; i < 3; i++) {

                //set inA inB inC
                for (int j = 0; j < N; j++) {
                        for (int k = 0; k < N; k++) {
                                inA[j][k] = rand() % (N - 1);
                                inB[j][k] = rand() % (N - 1);
                        }
                }

                kursFunc(inA, inB, outC);
                pass = cmp_arr(inA, inB, outC);
        }

        if (pass == 1) {
                fprintf(stdout, "----------Pass!----------\n");
                return 0;
        } else {
```

```
                fprintf(stderr, "----------Fail!-----------\n");
                return 1;
        }
}
```

Листинг исходного кода теста

```
############################################
#                Lab                 #
############################################
open_project -reset kr_02001_8
set_top kursFunc
add_files ./source/kurs.c
add_files -tb ./source/kurs_test.c

open_solution -reset sol1
create_clock -period 10 -name clk
set_clock_uncertainty 0.1
set_part {xa7a100tfgg484-2i}

csim_design
csynth_design
#cosim_design -trace_level all
############################################
#              Solutions              #
############################################
set all_solutions {sol2 sol3 sol4 sol5}
set all_periods   {{10}  {10} {10} {10}}

foreach the_solution $all_solutions the_period $all_periods {
open_solution -reset $the_solution
create_clock -period $the_period -name clk
set_clock_uncertainty 0.1
set_part {xa7a100tfgg484-2i}

csim_design
csynth_design
#cosim_design -trace_level all
}

Exit
```

Листинг tcl скрипта

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
```

```
#include "kurs.h"

int cmp_arr(data_sc inA[N][N], data_sc inB[N][N], data_sc inC[N][N], data_sc outD[N][N]) {

	static data_sc AC[N][N];
	static data_sc BC[N][N];

	for(int i = 0; i < N; i++) {
		data_sc temp_res = 0;
		for(int j = 0; j < N; j++) {
			AC[i][j] = 0;
			BC[i][j] = 0;
			for(int k = 0; k < N; k++) {
				AC[i][j] += inA[i][k] * inC[k][j];
				BC[i][j] += inB[i][k] * inC[k][j];
			}
			temp_res = AC[i][j] + BC[i][j];
			if (outD[i][j] != temp_res) {
				return 0;
			}
		}
	}

	return 1;
}

int main() {
	int pass = 0;


	static data_sc inA[N][N];
	static data_sc inB[N][N];
	static data_sc inC[N][N];
	static data_sc outD[N][N];
	struct timespec t0, t1;
	double acc_time = 0.0;

	for (int i = 0; i < 32; i++) {

		for (int j = 0; j < N; j++) {
			for (int k = 0; k < N; k++) {
				inA[j][k] = rand() % (N - 1);
				inB[j][k] = rand() % (N - 1);
				inC[j][k] = rand() % (N - 1);
```

```
                        }
                }

                if(clock_gettime(CLOCK_REALTIME, &t0) != 0) {
                        perror("Error in calling clock_gettime\n");
        exit(EXIT_FAILURE);
        }

                kursFunc(inA, inB, inC, outD);

                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(inA, inB, inC, outD);
        }

        if (pass == 1) {
                fprintf(stdout, "----------Pass!----------\n");
                return 0;
        } else {
                fprintf(stderr, "----------Fail!----------\n");
                return 1;
        }
}
```

Исходный код модернизированного теста