Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе №7_1
Курс: «Проектирование реконфигурируемых гибридных вычислительных систем»
Тема: Pipeline

Выполнил студент гр. 3540901/81501 Селиверстов Я.А.
(подпись)

Руководитель Антонов А.П.
(подпись)
“___” _____2019 г.

Санкт – Петербург
2019

ОГЛАВЛЕНИЕ

1.Задание

1.1. Исходный код программы

Листинг. Исходный код основной программы и теста

```
#include "dct.h"

void dct_1d(dct_data_t src[DCT_SIZE], dct_data_t dst[DCT_SIZE])
{
   unsigned int k, n;
   int tmp;
   const dct_data_t dct_coeff_table[DCT_SIZE][DCT_SIZE] = {
#include "dct_coeff_table.txt"
   };

DCT_Outer_Loop:
   for (k = 0; k < DCT_SIZE; k++) {
DCT_Inner_Loop:
      for(n = 0, tmp = 0; n < DCT_SIZE; n++) {
         int coeff = (int)dct_coeff_table[k][n];
         tmp += src[n] * coeff;
      }
      dst[k] = DESCALE(tmp, CONST_BITS);
   }
}

void dct_2d(dct_data_t in_block[DCT_SIZE][DCT_SIZE],
      dct_data_t out_block[DCT_SIZE][DCT_SIZE])
{
   dct_data_t row_outbuf[DCT_SIZE][DCT_SIZE];
   dct_data_t col_outbuf[DCT_SIZE][DCT_SIZE], col_inbuf[DCT_SIZE][DCT_SIZE];
   unsigned i, j;

   // DCT rows
Row_DCT_Loop:
   for(i = 0; i < DCT SIZE; i++) {
      dct_1d(in_block[i], row_outbuf[i]);
   }
   // Transpose data in order to re-use 1D DCT code
Xpose_Row_Outer_Loop:
   for (j = 0; j < DCT_SIZE; j++)
Xpose_Row_Inner_Loop:
      for(i = 0; i < DCT_SIZE; i++)
         col_inbuf[j][i] = row_outbuf[i][j];
   // DCT columns
Col_DCT_Loop:
   for (i = 0; i < DCT_SIZE; i++) {
      dct_1d(col_inbuf[i], col_outbuf[i]);
   }
   // Transpose data back into natural order
Xpose_Col_Outer_Loop:
   for (j = 0; j < DCT_SIZE; j++)
Xpose_Col_Inner_Loop:
      for(i = 0; i < DCT_SIZE; i++)
         out_block[j][i] = col_outbuf[i][j];
}

void read_data(short input[N], short buf[DCT_SIZE][DCT_SIZE])
{
   int r, c;
```

```
RD_Loop_Row:
   for (r = 0; r < DCT_SIZE; r++) {
RD_Loop_Col:
      for (c = 0; c < DCT_SIZE; c++)
         buf[r][c] = input[r * DCT SIZE + c];
   }
}

void write_data(short buf[DCT_SIZE][DCT_SIZE], short output[N])
{
   int r, c;

WR_Loop_Row:
   for (r = 0; r < DCT_SIZE; r++) {
WR_Loop_Col:
      for (c = 0; c < DCT_SIZE; c++)
         output[r * DCT_SIZE + c] = buf[r][c];
   }
}

void dct(short input[N], short output[N])
{

   short buf_2d_in[DCT_SIZE][DCT_SIZE];
   short buf_2d_out[DCT_SIZE][DCT_SIZE];

   // Read input data. Fill the internal buffer.
   read_data(input, buf_2d_in);

   dct_2d(buf_2d_in, buf_2d_out);

   // Write out the results.
   write_data(buf_2d_out, output);
}
```

```
// Copyright (C) 2008 AutoESL Design Techonologies, Inc.
// All rights reserved.

#include <stdio.h>
#include "dct.h"


// ******************************************************
int main() {
   short a[N], b[N], b_expected[N];
   int retval = 0, i;
   FILE *fp;

   fp=fopen("in.dat","r");
   for (i=0; i<N; i++){
      int tmp;
      fscanf(fp, "%d", &tmp);
      a[i] = tmp;
   }
   fclose(fp);

   fp=fopen("out.golden.dat","r");
   for (i=0; i<N; i++){
      int tmp;
      fscanf(fp, "%d", &tmp);
      b_expected[i] = tmp;
   }
```

```
   fclose(fp);

   dct(a, b);

   for (i = 0; i < N; ++i) {
      if(b[i] != b expected[i]){
         printf("Incorrect output on sample %d. Expected %d, Received %d \n",
i, b_expected[i], b[i]);
         retval = 2;
      }
   }

#if 0 // Optionally write out computed values
   fp=fopen("out.dat","w");
   for (i=0; i<N; i++){
      fprintf(fp, "%d\n", b[i]);
   }
   fclose(fp);
#endif

   if(retval != (2)){
      printf("    *** *** *** *** \n");
      printf("    Results are good \n");
      printf("    *** *** *** *** \n");
   } else {
      printf("    *** *** *** *** \n");
      printf("    BAD!! %d \n", retval);
      printf("    *** *** *** *** \n");
   }
   return retval;
}
```

```
#ifndef __DCT_H__
#define __DCT_H__


#define DW 16
#define N 1024/DW
#define NUM_TRANS 16


typedef short dct_data_t;


#define DCT_SIZE 8    /* defines the input matrix as 8x8 */
#define CONST_BITS  13
#define DESCALE(x,n)  (((x) + (1 << ((n)-1))) >> n)


void dct(short input[N], short output[N]);


#endif // __DCT_H__ not defined
```

1.2. Задание

- Launch the Vivado® HLS tool.
- Open the provided **dct_prj** Vivado HLS tool project located at: *C:\training\pipeline\demo\dct_prj*
- Access and review the source files (*dct.c* and *dct.h*) from the Explorer pane.
- Run C synthesis.
- Review the Synthesis report
- Double-click **dct_2d_csynth.rpt** to open the Synthesis report available under the *dct_prj* > *solution1* > *syn* > *report* folder in the Explorer pane.
- Similarly, open the **dct_1d2_csynth.rpt** file under the *dct_prj* > *solution1* > *syn* > *report* folder in the Explorer pane.
- Create a new solution named *solution2*.
- Accept the default settings and click **Finish**.
- Select **Project** > **Close Inactive Solution Tabs**.
- Apply the **PIPELINE** directive on *DCT_Inner_Loop* of the *dct_1d* function (shown below).
- Similarly, apply the **PIPELINE** directive to the following loops:
- *Xpose_Row_Inner_Loop* of the *dct_2d* function
- *Xpose_Col_Inner_Loop* of the *dct_2d* function
- *RD_Loop_Col* of the *read_data* function
- *WR_Loop_Col* of the *write_data* function
- Run C synthesis.
- Compare the results of the two solutions (*solution1* and *solution2*).
- Select **Project** > **Compare Reports**.
- Add the reports you wish to compare.
- Double-click **dct_2d_csynth.rpt** to open the Synthesis report available under the *dct_prj* > *solution2* > *syn* > *report* folder in the Explorer pane.
- Similarly, open the **dct_1d2_csynth.rpt** file under the *dct_prj* > *solution2* > *syn* > *report* folder in the Project Explorer pane.
- Create a new solution named *solution3*.
- Accept the default settings and click **Finish**.
- Delete the **PIPELINE** directive from *DCT_Inner_Loop* of the *dct_1d* function.
- Apply the **PIPELINE** directive on *DCT_Outer_Loop* of the *dct_1d* function.
- Run C synthesis.
- It is safe to ignore the warnings if any.
- Compare the results of the two solutions (*solution2* and *solution3*).

2.Решение №1

2.1Моделирование

При запуске моделирования можно увидеть, что тест успешно пройден:

```
INFO: [SIM 2] ************** CSIM start **************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
make: 'csim.exe' is up to date.
    *** *** *** ***
    Results are good
    *** *** *** ***
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] ************** CSIM finish **************
```

Рис.1.1.

2.2. Синтез

Приведем в отчете требуемые данные о проекте:

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.454	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
2935	2935	2935	2935	none

Рис. 1.2. Информация о производительности

Здесь можно увидеть, что достигнутая задержка равна 8.454 + 0.1, что укладывается в заданные нами требования к тактовой частоте. Также вычислено наихудшее значение Latency равное 2935.

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	156	-
FIFO	-	-	-	-	-
Instance	3	1	177	671	0
Memory	2	-	0	0	0
Multiplexer	-	-	-	149	-
Register	-	-	69	-	-
Total	5	1	246	976	0
Available	650	600	202800	101400	0
Utilization (%)	~0	~0	~0	~0	0

Рис. 1.3. Занимаемые ресурсы

Данный проект займет на микросхеме 1 блок DSP48E, 5 BRAM_18K, 246 FF (69 регистров и 177 экземпляров) , 976 LUT.

Произведем синтез функции dct_2d и приведем его отчет на рис 1.4.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.454	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
2644	2644	2644	2644	none

Detail

Instance

Loop

Рис. 1.4. Информация о производительности dct_2d

Здесь можно увидеть, что достигнутая задержка равная 8.454 + 0.1 не изменилась по сравнению с первым вариантом. Также вычислено наихудшее значение Latency, которое немного лучше предыдущего равное 2644.

Произведем синтез функции dct_1d2 и приведем его отчет на рис 1.5.

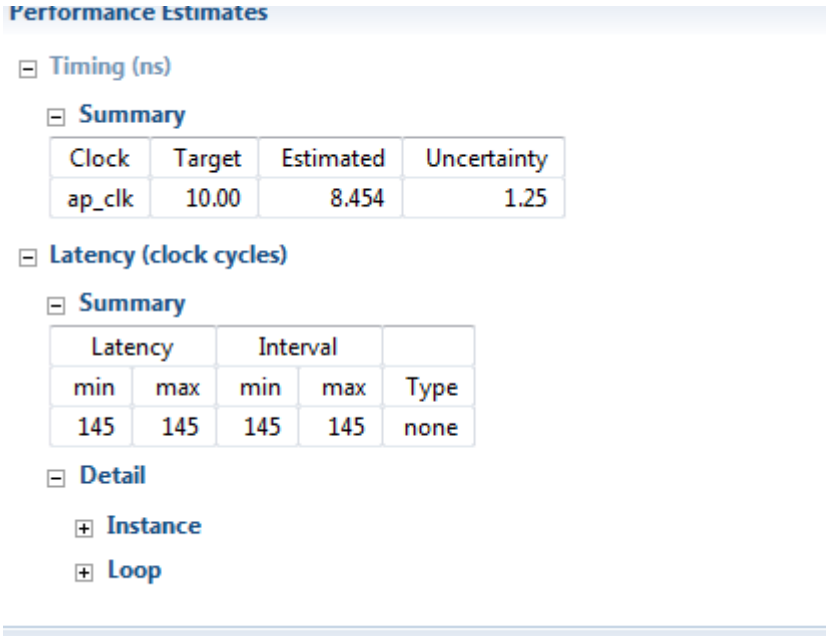

Рис. 1.5. Информация о производительности dct_1d2

Здесь можно увидеть, что достигнутая задержка равная 8.454 + 0.1 не изменилась по сравнению с первым вариантом. Значение Latency улучшилось и составило 145.

3. Решение №2

3.1.Директива

Добавим директиву PIPELINE: % HLSPIPELINE – рисунок 2.1.

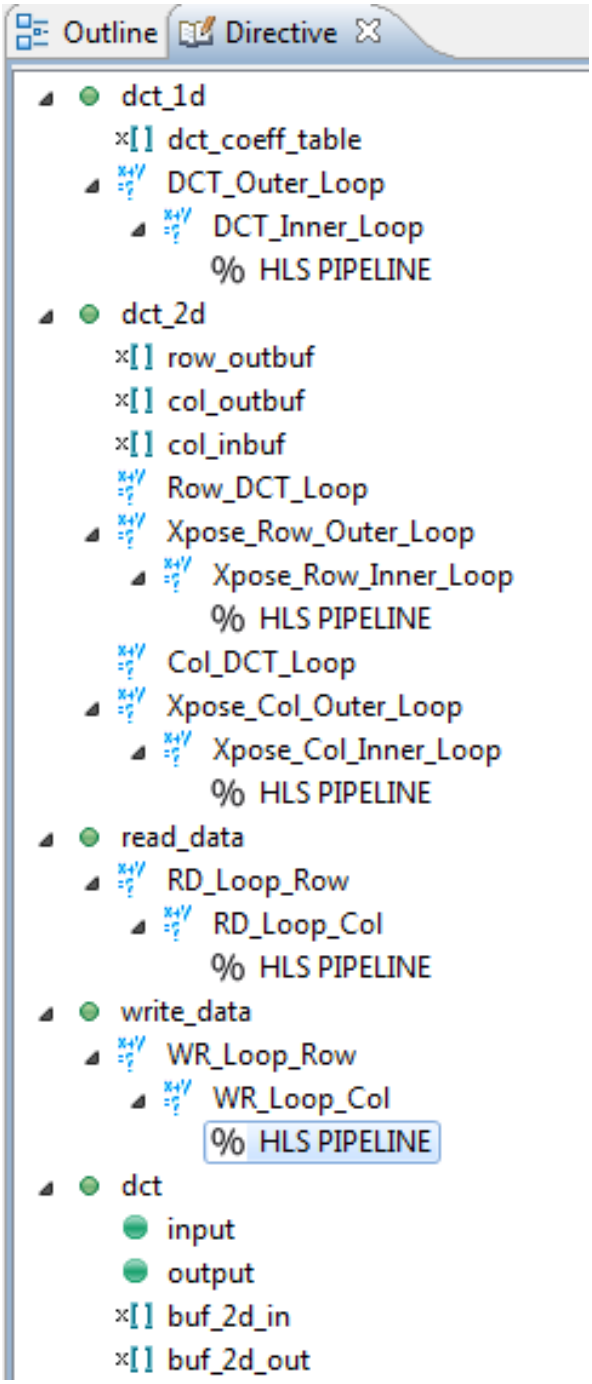

Рисунок 2.1. Добавленные директивы PIPELINE

3.2.Синтез

Приведем в отчете требуемые данные о результатах синтеза второго решения:

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.454	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
1723	1723	1723	1723	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	226	-
FIFO	-	-	-	-	-
Instance	3	1	162	828	0
Memory	2	-	0	0	0
Multiplexer	-	-	-	209	-
Register	-	-	61	-	-
Total	5	1	223	1263	0
Available	650	600	202800	101400	0
Utilization (%)	~0	~0	~0	1	0

Рисунок 2.2. Отчет о синтезе второго решения. Производительность и занимаемые ресурсы.

Здесь можно увидеть, что достигнутая задержка во-втром решении составила 8.454 + 0.1. Наихудшее значение Latency составило 1723.

Данный проект займет на микросхеме 1 блок DSP48E, 5блоков BRAM_18K, 223 FF (61 регистр и 162 экземпляров) , 1263 LUT.

Сравним результаты синтеза первого и второго решения на рисунке 2.3.

Performance Estimates

Timing (ns)

Clock		solution2	solution1
ap_clk	Target	10.00	10.00
	Estimated	8.454	8.454

Latency (clock cycles)

		solution2	solution1
Latency	min	1723	2935
	max	1723	2935
Interval	min	1723	2935
	max	1723	2935

Utilization Estimates

	solution2	solution1
BRAM_18K	5	5
DSP48E	1	1
FF	223	246
LUT	1263	976
URAM	0	0

Рисунок 2.3. Сравнение отчетов синтеза первого и второго решений.

Отчет о синтезе функции dct_2d представим на рисунке 2.4.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.454	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
1590	1590	1590	1590	none

Detail

Instance

Loop

Рисунок 2.4. Отчет о синтезе функции dct_2d

Здесь можно увидеть, что достигнутая задержка во-втром решении составила также 8.454 + 0.1. Наихудшее значение Latency составило 1590.

Отчет о синтезе функции dct_1d2представим на рисунке 2.5.

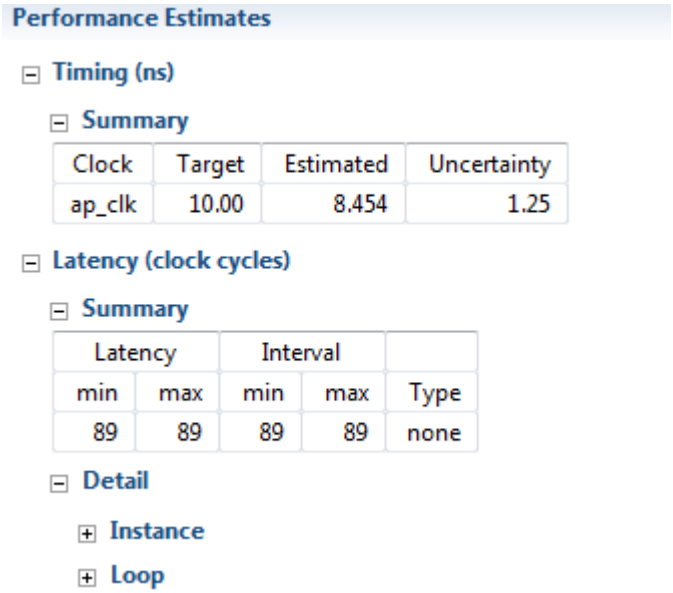

Рисунок 2.5. Отчет о синтезе функции dct_1d2

Здесь можно увидеть, что достигнутая задержка составила также 8.454 + 0.1, а наихудшее значение Latency – 89.

4.Решение №3

4.1.Директива.

Добавим директиву PIPELINE к внешнему циклу – рисунок 3.1.

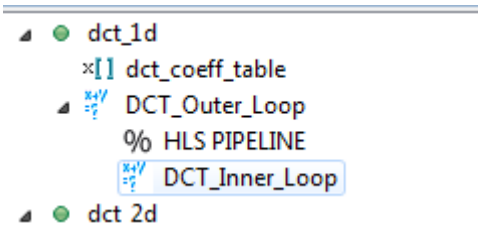

Рисунок 3.1. Применение директивы PIPELINE к внешнему циклу.

4.2. Синтез

Отчет о синтезе функции представим на рисунке 3.2.

Performance Estimates

Timing (ns)

Clock		solution3	solution2
ap_clk	Target	10.00	10.00
	Estimated	8.454	8.454

Latency (clock cycles)

		solution3	solution2
Latency	min	859	1723
	max	859	1723
Interval	min	859	1723
	max	859	1723

Utilization Estimates

	solution3	solution2
BRAM_18K	5	5
DSP48E	8	1
FF	670	223
LUT	1403	1263
URAM	0	0

Рисунок 3.2. Сравнение отчетов второго и третьего решений

В решении 3 величина задержки меньше, чем во –втором , Latency = 859.

Ресурсов в третьем решении используется больше чем во втором: FF = 670, LUT=1403, DSP48E – 8.

5. Выводы

Директива PIPELINE используется для добавления регистров конвейеризации что ведет к снижению значения Latency в проекте. Если данная директива применяется к внешнему циклу, автоматически будут развернуты внутренние циклы и будут добавлены регистры конвейеризации. При правильном применении директивы PIPELINE можно значительно увеличить пропускную способность проекта.