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 С.А.

(подпись)

Руководитель Антонов А.П.

(подпись)

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

Санкт – Петербург

2019

**ОГЛАВЛЕНИЕ**

[**1.Задание** 3](#_Toc27953352)

[1.1. Исходный код программы 3](#_Toc27953353)

[1.2. Задание 6](#_Toc27953354)

[**2.Решение №1** 7](#_Toc27953355)

[2.1Моделирование 7](#_Toc27953356)

[2.2. Синтез 7](#_Toc27953357)

[**3. Решение №2** 10](#_Toc27953358)

[3.2.Синтез 11](#_Toc27953359)

[**4.Решение №3** 14](#_Toc27953360)

[4.2. Синтез 14](#_Toc27953361)

[**5. Выводы** 15](#_Toc27953362)

# **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:

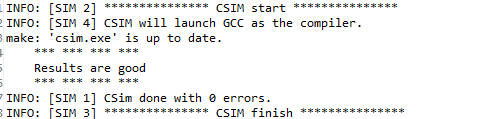


Рис.1.1.

## 2.2. Синтез

Приведем в отчете требуемые данные о проекте:

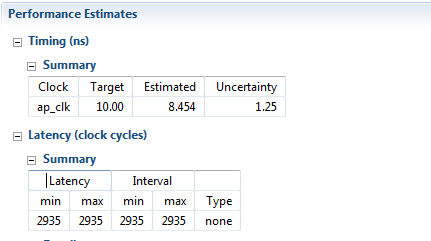


Рис. 1.2. Информация о производительности

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.

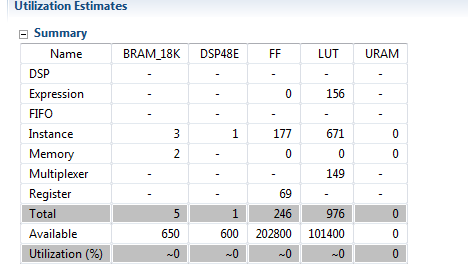


Рис. 1.3. Занимаемые ресурсы

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.

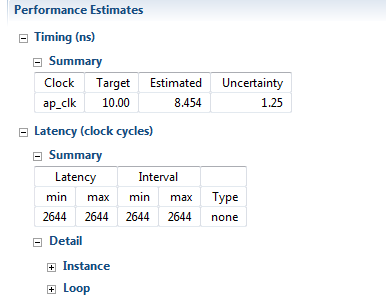


Рис. 1.4. Информация о производительности dct\_2d

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.

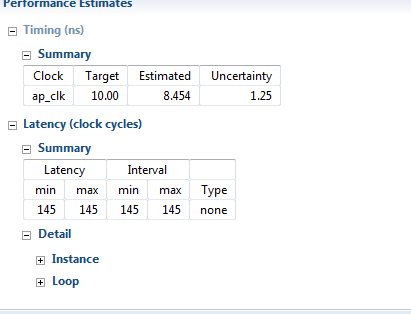


Рис. 1.5. Информация о производительности dct\_1d2

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.

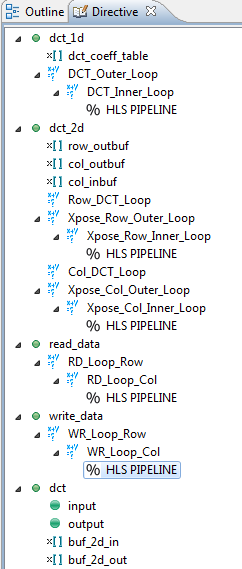


Рисунок 2.1. Добавленные директивы PIPELINE

## 3.2.Синтез

Приведем в отчете требуемые данные о результатах синтеза второго решения:

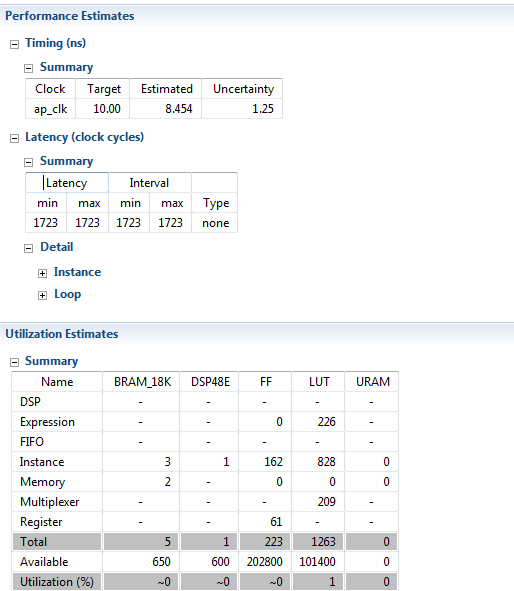


Рисунок 2.2. Отчет о синтезе второго решения. Производительность и занимаемые ресурсы.

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.

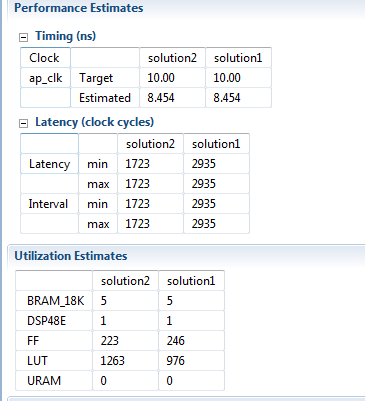


Рисунок 2.3. Сравнение отчетов синтеза первого и второго решений.

Отчет о синтезе функции dct\_2d представим на рисунке 2.4.

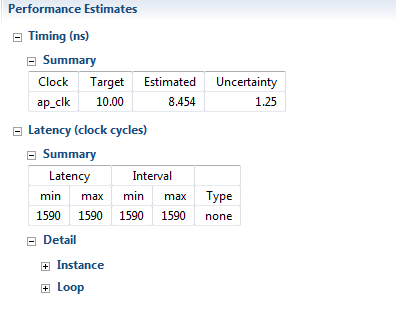


Рисунок 2.4. Отчет о синтезе функции dct\_2d

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.

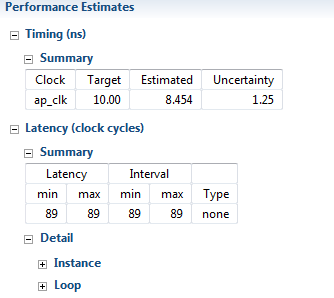


Рисунок 2.5. Отчет о синтезе функции dct\_1d2

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.

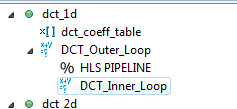


Рисунок 3.1. Применение директивы PIPELINE к внешнему циклу.

## 4.2. Синтез

Отчет о синтезе функции представим на рисунке 3.2.

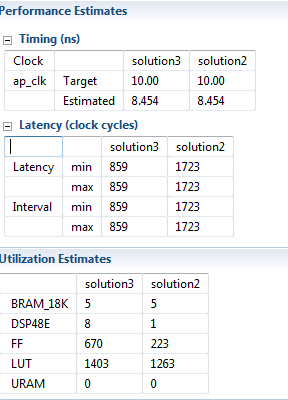


Рисунок 3.2. Сравнение отчетов второго и третьего решений

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.