Санкт-Петербургский Политехнический Университет Петра Великого  
Институт Компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий

Лабораторная работа 1

Предмет: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Введение в Vivado HLS

Задание 2

Студент: Ерниязов Т.Е.

Гр. № 3540901/81501

Преподаватель: Антонов А.П.

Санкт-Петербург

2019

Оглавление

[Задание 4](#_Toc20774223)

[Ход работы 5](#_Toc20774224)

[Решение 1 5](#_Toc20774225)

[Моделирование 6](#_Toc20774226)

[Синтез 7](#_Toc20774227)

[C/RTL моделирование 9](#_Toc20774228)

[Решение 2 10](#_Toc20774229)

[Моделирование 10](#_Toc20774230)

[Синтез 10](#_Toc20774231)

[С/RTL моделирование 12](#_Toc20774232)

[Выводы 13](#_Toc20774233)

# Задание

Создать проект lab1\_2

Подключить файл lab1\_2.c (папка source)

Написать тест lab1\_2\_test.c по аналогии с тестом lab1\_1\_test.c

Микросхема: xa7a12tcsg325-1q

Сделать solution1

задать: clock period 6; clock\_uncertainty 0.1

осуществить моделирование

осуществить синтез\

привести в отчете:

• performance estimates=>summary

• utilization estimates=>summary

• Performance Profile

• scheduler viewer (выполнить Zoom to Fit)

На скриншоте показать Latency

На скриншоте показать Initiation Interval

• resource viewer (выполнить Zoom to Fit)

На скриншоте показать Latency

На скриншоте показать Initiation Interval

Осуществить C|RTL моделирование

Открыть временную диаграмму (все сигналы)

• Отобразить два цикла обработки на одном экране

На скриншоте показать Latency

На скриншоте показать Initiation Interval

Сделать solution2

задать: clock period 10; clock\_uncertainty 0.1

осуществить моделирование

осуществить синтез

привести в отчете:

• performance estimates=>summary

• utilization estimates=>summary

• Performance Profile

• scheduler viewer (выполнить Zoom to Fit)

На скриншоте показать Latency

На скриншоте показать Initiation Interval

• resource viewer (выполнить Zoom to Fit)

На скриншоте показать Latency

На скриншоте показать Initiation Interval

Осуществить C|RTL моделирование

Открыть временную диаграмму (все сигналы)

•Отобразить два цикла обработки на одном экране

На скриншоте показать Latency

На скриншоте показать Initiation Interval

Выводы

Объяснить отличие двух solutions

# Ход работы

## Решение 1

1. Создание проекта lab1\_2.
2. Подключение файлов lab1\_2.c, lab1\_2\_test.c.

Lab1\_2.c

void lab1\_2 (int in[3], char a, char b, char c, int out[3]) {

int x,y;

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

x = in[i];

y = a\*x + b + c;

out[i] = y;

}

}

Lab1\_2\_test.c

#include <stdio.h>

int main()

{

int inA, inB, inC;

int out[3];

// For adders

int refOut[3][3] = {{6,7,8}, {12,14,16}, {24,28,32}};

int pass;

int i;

inA = 1;

inB = 2;

inC = 3;

int in[3] = {1, 2, 3};

int res[3];

// Call the adder for 5 transactions

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

{

lab1\_2(in, inA, inB, inC, out);

for (int j=0; j<3; j++)

{

fprintf(stdout, " %d\*1+%d+%d=%d \n", inA, inB, inC, out[j]);

}

// Test the output against expected results

if (equals(out, refOut[i]) == 1)

pass = 1;

else

pass = 0;

inA=inA\*2;

inB=inB\*2;

inC=inC\*2;

}

if (pass)

{

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

return 0;

}

else

{

fprintf(stderr, "----------Fail!------------\n");

return 1;

}

}

int equals(int a[], int b[])

{

int res = 1;

for (int i = 0; i<sizeof(a)/sizeof(int); i++)

{

if (a[i] == b[i])

{

res = 1;

}

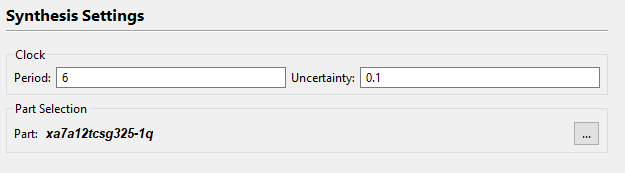
else return 0;

}

return res;

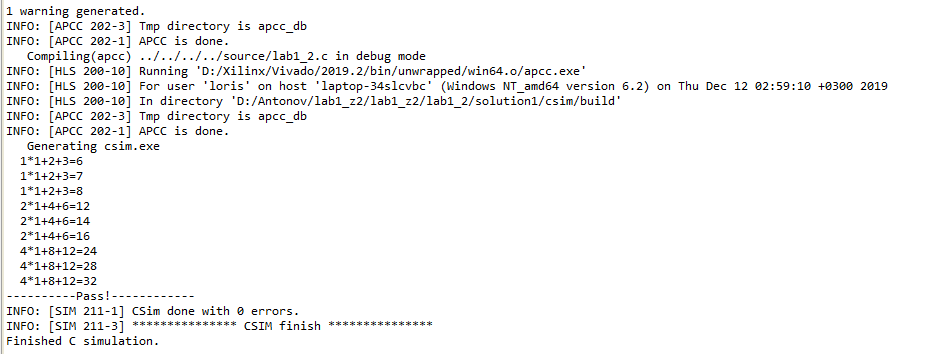
}

1. Конфигурирование решения.



### Моделирование

1. Результат моделирования заданного решения.

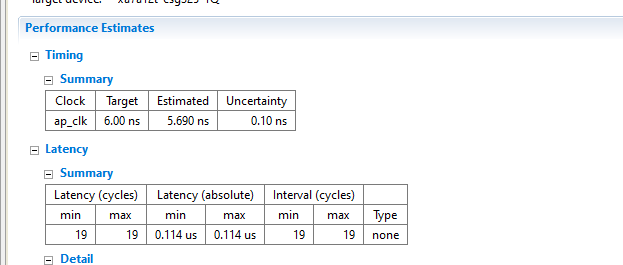


Моделирование выполнено без ошибок. Тест пройден успешно.

### Синтез

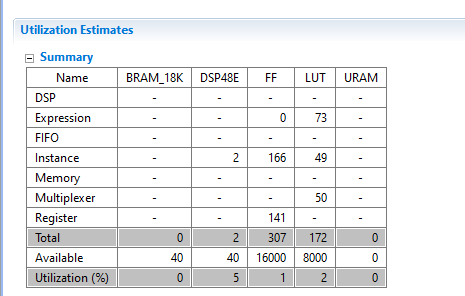
1. Выполним команду Solution -> Run C Synthesis -> Active solution

Производительность



Достигнутая задержка (estimated) равна 5,690 + (погрешность - uncertainty) 0,10. Величина задержки укладывается в заданные требования к тактовой частоте, но является значительной для заданных ограничений.

Занимаемые ресурсы



Данный проект будет занимать на микросхеме:

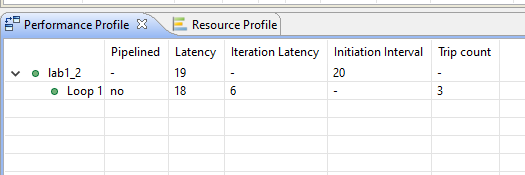
4 DSP блока.

DSP блок – цифровой процессор обработки сигналов, специализированный микропроцессор, особенностью работы которого является поточный характер обработки больших объемов данных в реальном масштабе времени и, обычно, с интенсивным обменом данных с другими внешними устройствами. (Нужен для преобразования сигналов, представленных в виде цифр, как правило, в режиме реального времени). В данном блоке будут использованы сумматор и умножитель.

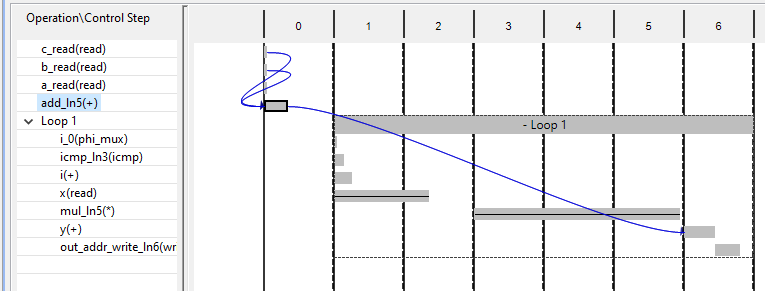
307 регистров для хранения и считывания данных (чисел).

172 LUT

1. Перейдем на вкладку Analysis.



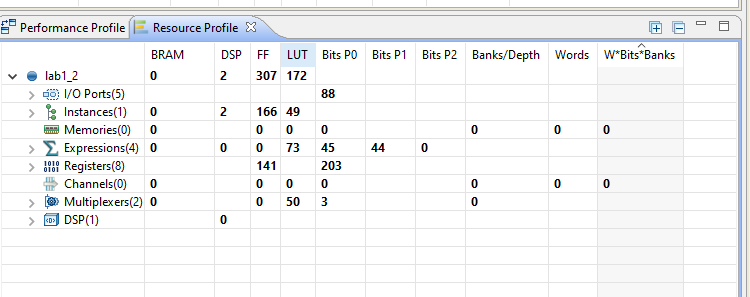
Задержка времени со старта до момента получения результата составляет 19 тактов. Задержка времени до готовности получения новых данных – 20 тактов. Временная диаграмма:



Получение результата происходит следующим образом:

* Первый такт
  + Считывание параметра c
  + Считывание параметра b
  + Считывание параметра a
  + Инициализация параметра
* Второй – седьмой такты
  + Выполнение действий в цикле

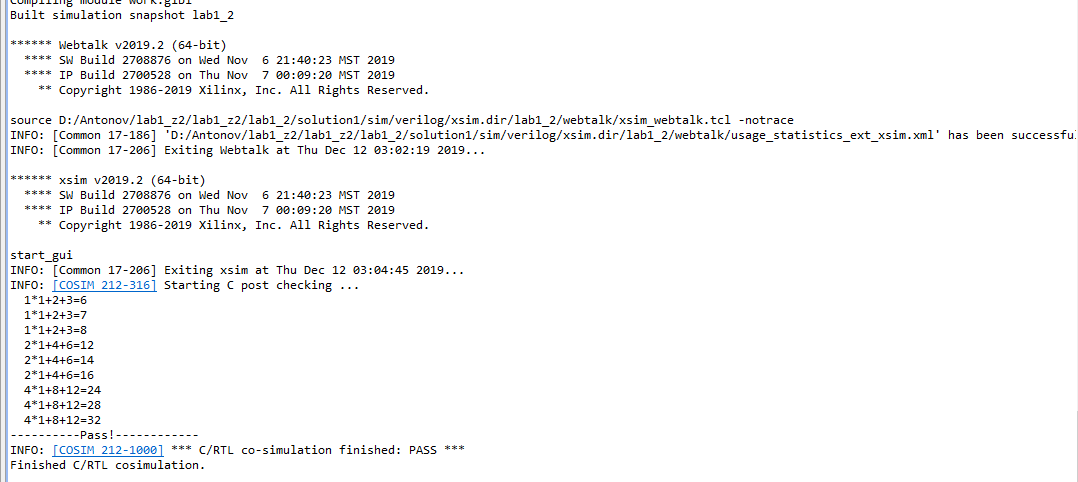
1. Профиль ресурсов



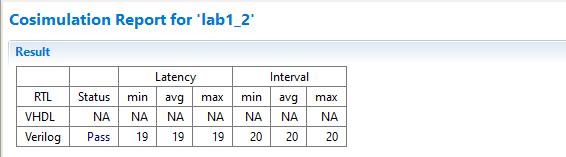
Значения в отчете по ресурсам идентичные значениям в пункте 5.

### C/RTL моделирование

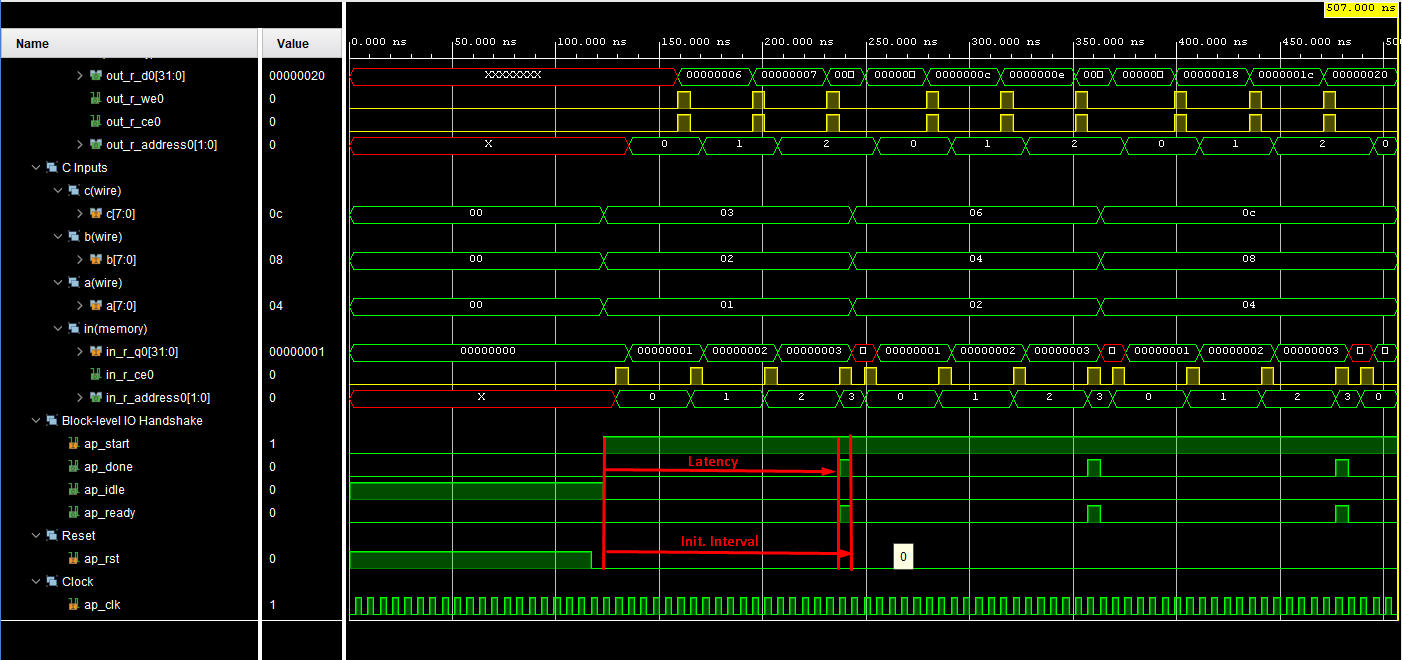
1. Результат выполнения.



1. Отчет о выполнении



Результаты идентичны полученным в пункте 6.

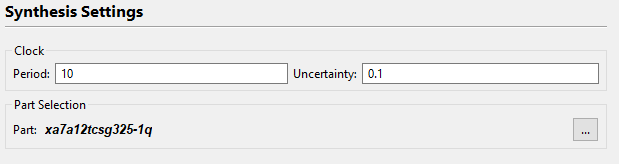


## 

## Решение 2

### Моделирование

1. Создание и конфигурирование решения.

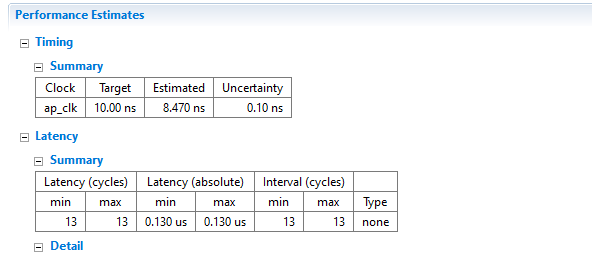


Исходные файлы соответствуют файлам решения 1.

### Синтез

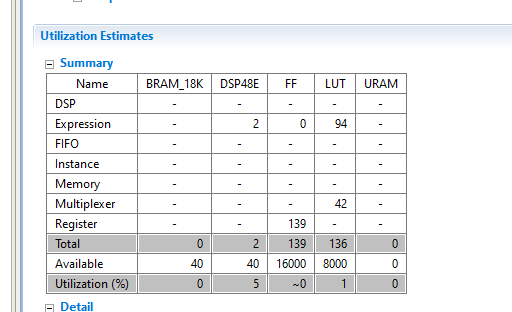
1. Выполним команду Solution-> Run C Synthesis -> Active solution

Производительность



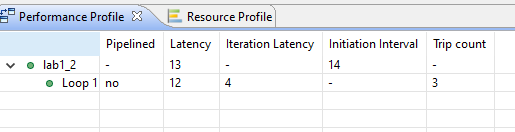
Отношение величины полученной задержки к заданному значению значительно отличаются по отношению к первому решению. Результаты свидетельствуют о том, что измененные параметры влекут за собой изменение устройства.

Затрачиваемые ресурсы

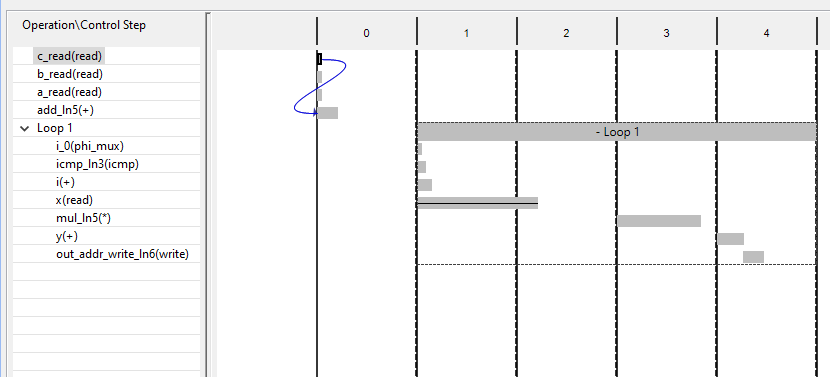


По сравнению с решением 1, использование ресурсов значительно меньше. Используется более чем в 2 раза меньше регистров и значительно меньше LUT.

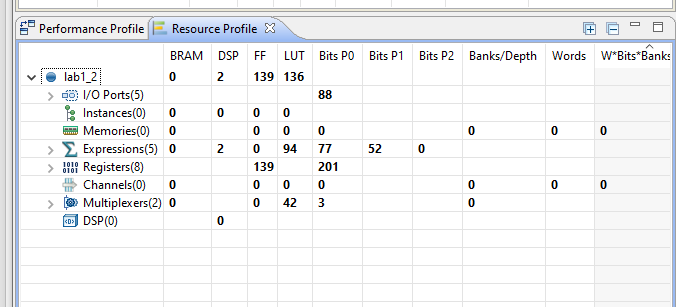
1. Перейдем на вкладку Analysis.



На изображении видно, что до получения ответа требуется 13 тактов, а готовность получения новых данных наступает через 14 такт. Действия в цикле ограничиваются 4 тактами вместо 6.



Общий отчет о затраченных ресурсах

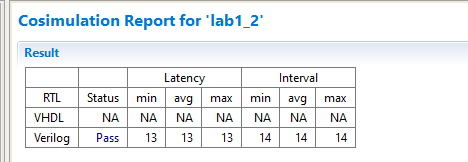


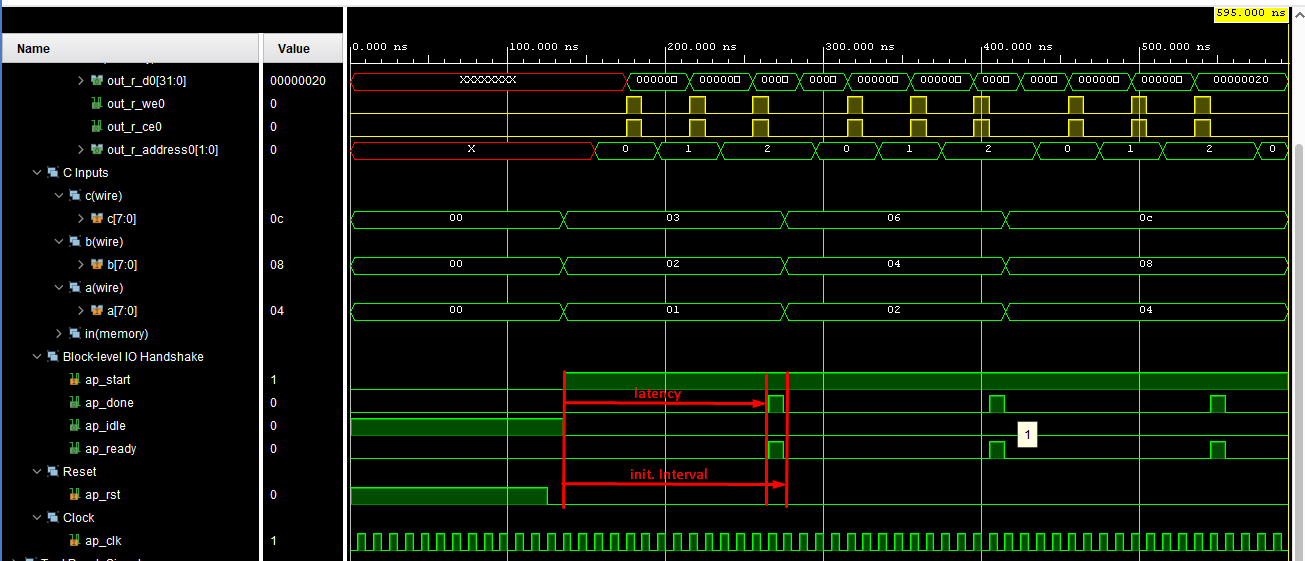
### С/RTL моделирование

1. Результат выполнения



1. Отчет решения



На временной диаграмме изображены Latency, Initiation interval. 

# Выводы

В ходе работы были рассмотрены два решения, отличающиеся лишь параметром периода тактовой частоты. В ходе выполнения лабораторной работы был получен результат, что при большем периоде тактов, программа сумела вместить в 1 период чисто комбинаторную схему, в то время как при меньшем периоде это оказалось невозможным. Были получены 2 решения: первое – полный цикл выполнения составил 19 тактов, а максимальная задержка обработки сигнала на такте составляет 5.69 нс, и второе –решение, задержка в котором составила большую величину 8.47 нс, но при этом все действия выполняются за 13 тактов.