ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Дисциплина «Проектирование реконфигурируемых гибридных вычислительных систем»

**ОТЧЕТ**

по лабораторной работе №4+

Выполнил:

студент группы 3540901/02001

Кузьмичев Егор Сергеевич

« » \_\_\_\_\_\_\_\_\_\_ 2021г., \_\_\_\_\_\_\_\_\_\_

(подпись)

(подпись)

Проверила:

доцент Антонов Алексей Петрович

«\_\_» \_\_\_\_\_\_\_\_\_\_ 2021г., \_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2021

**Оглавление**

[Цели работы 5](#_Toc86771622)

[Часть Lab1\_z1 5](#_Toc86771623)

[ЧастьLab1\_z2 6](#_Toc86771624)

[Результаты работы 9](#_Toc86771625)

[Часть Lab1\_z1 9](#_Toc86771626)

[Solution 1 9](#_Toc86771627)

[Solution 2 12](#_Toc86771628)

[Выводы 15](#_Toc86771629)

[Часть Lab1\_z2 15](#_Toc86771630)

[Solution 1 15](#_Toc86771631)

[Solution 2 18](#_Toc86771632)

[Выводы 21](#_Toc86771633)

[Приложение 22](#_Toc86771634)

[1. lab1\_z1.c 22](#_Toc86771635)

[2. lab1\_z1\_test.c 22](#_Toc86771636)

[3. lab1\_z2.c 23](#_Toc86771637)

[4. lab1\_z2\_test.c 23](#_Toc86771638)

**Список иллюстраций**

[**Рис. 1.** Таблица Performance Estimates 10](#_Toc86771639)

[**Рис. 2.** Таблица Utilization Estimates 10](#_Toc86771640)

[**Рис. 3.** Таблица Performance Profile 11](#_Toc86771641)

[**Рис. 4.** Таблица Resource Profile 11](#_Toc86771642)

[**Рис. 5.** Таблица Schedule Viewer 11](#_Toc86771643)

[**Рис. 6.** Результаты CoSimulation 12](#_Toc86771644)

[**Рис. 7.** CoSimulation Report 12](#_Toc86771645)

[**Рис. 8.** Таблица Performance Estimates 13](#_Toc86771646)

[**Рис. 9.** Таблица Utilization Estimates 13](#_Toc86771647)

[**Рис. 10.** Таблица Performance Profile 14](#_Toc86771648)

[**Рис. 11.** Таблица Resource Profile 14](#_Toc86771649)

[**Рис. 12.** Таблица Schedule Viewer 14](#_Toc86771650)

[**Рис. 13.** Результаты CoSimulation 14](#_Toc86771651)

[**Рис. 14.** CoSimulation Report 15](#_Toc86771652)

[**Рис. 15.** Соотношение симуляций 15](#_Toc86771653)

[**Рис. 16.** Таблица Performance Estimates 16](#_Toc86771654)

[**Рис. 17.** Таблица Utilization Estimates 16](#_Toc86771655)

[**Рис. 18.** Таблица Performance Profile 17](#_Toc86771656)

[**Рис. 19.** Таблица Schedule Viewer 17](#_Toc86771657)

[**Рис. 20.** Таблица Resource Viewer 18](#_Toc86771658)

[**Рис. 21.** Результаты CoSimulation 18](#_Toc86771659)

[**Рис. 22.** CoSimulation Report 18](#_Toc86771660)

[**Рис. 23.** Таблица Performance Estimates 19](#_Toc86771661)

[**Рис. 24.** Таблица Utilization Estimates 19](#_Toc86771662)

[**Рис. 25.** Таблица Performance Profile 20](#_Toc86771663)

[**Рис. 26.** Таблица Schedule Viewer 20](#_Toc86771664)

[**Рис. 27.** Таблица Resource Viewer 20](#_Toc86771665)

[**Рис. 28.** Результаты CoSimulation 21](#_Toc86771666)

[**Рис. 29.** CoSimulation Report 21](#_Toc86771667)

[**Рис. 30.** Соотношение симуляций 21](#_Toc86771668)

# Цели работы

## Часть Lab4\_z3

* Создать проект lab4\_z3
* Микросхема: xa7a12tcsg325-1q
* ***N = 16384, scale = { три случайных числа).***
* Создать две функции (см. Текст ниже) – исходную и модифицированную - и провести их анализ.

**Single-producer-consumer**

***For Vivado HLS to perform the DATAFLOW optimization, all elements passed between tasks***

***must follow a single-producer-consumer model. Each variable must be driven from a single task***

***and only be consumed by a single task. In the following code example, temp1fans out and is***

***consumed by both Loop2and Loop3. This violates the single-producer-consumer model.***

Void foo\_b(int data\_in[N], int scale[3], int data\_out1[N], int data\_out2[N]) {

int temp1[N];

Loop1: for(inti = 0; i< N; i++) {

temp1[i] = data\_in[i] \* scale[0];

}

Loop2: for(int j = 0; j < N; j++) {

data\_out1[j] = temp1[j] \* scale[1];

}

Loop3: for(int k = 0; k < N; k++) {

data\_out2[k] = temp1[k] \* scale[2];

}

}

***A modified version of this code uses function Split to create a single-producer-consumer***

***design. In this case, data flows from Loop1 to Split and then to Loop2 and Loop3.***

***The data now flows between all four tasks, and Vivado HLS can perform the DATAFLOW***

***Optimization***

void Split (in[N], out1[N], out2[N]) {

// Duplicated data

L1:for(inti=1;i<N;i++) {

out1[i] = in[i];

out2[i] = in[i];

}

}

voidfoo\_m(intdata\_in[N], int scale[3], int data\_out1[N], int data\_out2[N]) {

int temp1[N], temp2[N]. temp3[N];

Loop1: for(inti = 0; i< N; i++) {

temp1[i] = data\_in[i] \* scale[0];

}

Split(temp1, temp2, temp3);

Loop2: for(int j = 0; j < N; j++) {

data\_out1[j] = temp2[j] \* scale[1];

}

Loop3: for(int k = 0; k < N; k++) {

data\_out2[k] = temp3[k] \* scale[2];

}

}

* Создать тест lab4\_z3\_test.c для проверки функций выше.
* Для функции **foo\_b**
  + задать: clock period 6, 8, 10, 12; clock\_uncertainty 0.1
  + осуществить моделирование (с выводом результатов в консоль)
  + осуществить синтез и выбрать лучший вариант (максимум производительности при наименьших аппаратных затратах), привести результаты сравнения вариантов + таблица.
    - Для выбранного варианта привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнитьZoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * resource viewer (выполнитьZoom to Fit)
  + **Написать tcl файл автоматизирующий исследование**
* Для функции **foo\_m**
  + задать: clock period, выбранный на предыдущем этапе; clock\_uncertainty 0.1
  + осуществить моделирование (с выводом результатов в консоль)
  + осуществить синтездляслучая **FIFO for the memory buffers**:
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнитьZoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * **Dataflow viewer**
  + осуществить синтез для случая **ping-pongbuffers**:
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнитьZoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * **Dataflow viewer**
  + Осуществить C|RTL моделирование для случая **FIFO for the memory buffers**
    - Привести результаты из консоли
    - Открыть временную диаграмму (все сигналы)
      * Отобразить два цикла обработки на одном экране
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
  + **Написать tcl файл автоматизирующий исследования**
* Выводы
  + Объяснить отличия в синтезе foo\_b и двух вариантов foo\_m между собой

Исследование времени выполнения на ПК

* **Используются исходные коды и результаты исследования проведенного ранее.**
* На базе использованного выше Си теста создать отдельный, модернизированный, тест для проверки времени выполнения синтезируемой функции на ПК:
  + добавить в тест операторы измерения **времени выполнения** синтезируемой функции (например, как-то так: <https://solarianprogrammer.com/2019/04/17/c17-programming-measuring-execution-time-delaying-program/> ).
  + Увеличить количество запусков синтезируемой функции до 32. Для каждого запуска измерить время, найти среднее значение и вывести как результат.
  + Точность измерения времени (наносекунды).
  + Провести исследование времени выполнения синтезируемой функции на Вашем ПК
    - Осуществить компиляцию модернизированного теста и запустить его как отдельное приложение
    - В отчете привести:
      * Параметры Вашего ПК: тип процессора, частота работы процессора, объем ОЗУ
      * результаты измерения времени выполнения
* Оформить отчет, который должен включать
  + Задание
  + Раздел с описанием исходного кода функции
  + Раздел с описанием теста
  + Раздел с описание созданного командного файла
  + Раздел с анализом результатов (со снимками экрана с заполненной таблицей и полученным графиком)
    - Анализ и выбор оптимального (критерий максимальная производительность) решения
  + Результаты исследования времени выполнения на ПК и сравнение с аппаратными решениями.
  + Выводы

Архив должен включать всю рабочую папку проекта (включая модернизированный тест и скомпилированное приложение), отчет

## ЧастьLab4\_z4

* Создать проект lab4\_z4
* Микросхема: xa7a12tcsg325-1q
* ***N = 16384, scale = { два случайных числа).***
* Создатьдвефункции (см. Текст ниже) – исходную и модифицированную - и провести их анализ.

**Bypassing Tasks**

**Data should generally flow from one task to another. If you bypass tasks, this reduces the performance of the DATAFLOW optimization. In the following example, Loop1 generates the values for temp1and temp2. However, the next task, Loop2, only uses the value of temp1.**

**The value of temp2is not consumed until after Loop2. Therefore, temp2bypasses the next task in the sequence, which limits the performance of the DATAFLOW optimization**

voidfoo\_b(intdata\_in[N], int scale[3], int data\_out1[N], int data\_out2[N]) {

int temp1[N], temp2[N], temp3[N];

Loop1: for(inti = 0; i< N; i++) {

temp1[i] = data\_in[i] \* scale[0];

temp2[i] = data\_in[i] \* scale[1];

}

Loop2: for(int j = 0; j < N; j++) {

temp3[j] = temp1[j] \*scale[2];

}

Loop3: for(int k = 0; k < N; k++) {

data\_out[k] = temp2[k] + temp3[k];

}

}

**You can modify the code so that Loop2consumes temp2and produces temp4as follows. This ensures that the data flow from one task to the next.**

voidfoo\_b(intdata\_in[N], int scale[3], int data\_out1[N], int data\_out2[N]) {

int temp1[N], temp2[N], temp3[N], **temp4[N]**;

Loop1: for(inti = 0; i< N; i++) {

temp1[i] = data\_in[i] \* scale[0];

temp2[i] = data\_in[i] \* scale[1];

}

Loop2: for(int j = 0; j < N; j++) {

temp3[j] = temp1[j] \* scale[2];

**temp4[j] = temp2[j];**

}

Loop3: for(int k = 0; k < N; k++) {

data\_out[k] = **temp4[k]** + temp3[k];

}

}

* Создать тест lab4\_z4\_test.cдля проверки функций выше.
* Для функции **foo\_b**
  + задать: clock period,**выбранный в lab4\_z3**; clock\_uncertainty 0.1
  + осуществить моделирование (с выводом результатов в консоль)
  + осуществить синтез:
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнитьZoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
* Для функции **foo\_m**
  + задать: clock period,**выбранный в lab4\_z3**; clock\_uncertainty 0.1
  + осуществить моделирование (с выводом результатов в консоль)
  + осуществить синтез для случая **FIFO for the memory buffers**:
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнитьZoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * **Dataflow viewer**
  + осуществить синтез для случая **ping-pongbuffers**:
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнитьZoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * **Dataflow viewer**
  + Осуществить C|RTL моделирование для случая **FIFO for the memory buffers**
    - Привести результаты из консоли
    - Открыть временную диаграмму (все сигналы)
      * Отобразить два цикла обработки на одном экране
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
  + **Написать tcl файл автоматизирующий исследование**
* Выводы
  + Объяснить отличия в синтезе foo\_b и двух вариантов foo\_m между собой

Исследование времени выполнения на ПК

* **Используются исходные коды и результаты исследования проведенного ранее.**
* На базе использованного выше Си теста создать отдельный, модернизированный, тест для проверки времени выполнения синтезируемой функции на ПК:
  + добавить в тест операторы измерения **времени выполнения** синтезируемой функции (например, как-то так: <https://solarianprogrammer.com/2019/04/17/c17-programming-measuring-execution-time-delaying-program/> ).
  + Увеличить количество запусков синтезируемой функции до 32. Для каждого запуска измерить время, найти среднее значение и вывести как результат.
  + Точность измерения времени (наносекунды).
  + Провести исследование времени выполнения синтезируемой функции на Вашем ПК
    - Осуществить компиляцию модернизированного теста и запустить его как отдельное приложение
    - В отчете привести:
      * Параметры Вашего ПК: тип процессора, частота работы процессора, объем ОЗУ
      * результаты измерения времени выполнения
* Оформить отчет, который должен включать
  + Задание
  + Раздел с описанием исходного кода функции
  + Раздел с описанием теста
  + Раздел с описание созданного командного файла
  + Раздел с анализом результатов (со снимками экрана с заполненной таблицей и полученным графиком)
    - Анализ и выбор оптимального (критерий максимальная производительность) решения
  + Результаты исследования времени выполнения на ПК и сравнение с аппаратными решениями.
  + Выводы

Архив должен включать всю рабочую папку проекта (включая модернизированный тест и скомпилированное приложение), отчет

## ЧастьLab4\_z5

* Создать проект lab4\_z5
* Микросхема: xa7a12tcsg325-1q
* ***N = 16384, scale = { два случайных числа).***
* Создатьдвефункции (см. Текст ниже) – исходную и модифицированную - и провести их анализ.

***Conditional Execution of Tasks***

***The DATAFLOW optimization does not optimize tasks that are conditionally executed. The following example highlights this limitation. In this example, the conditional execution of Loop1 and Loop2 prevents Vivado HLS from optimization the data flow between these loops, because the data does not flow from one loop into the next.***

voidfoo\_b(int data\_in1[N], intdata\_out[N], int scale[2], charsel) {

int temp1[N], temp2[N];

if (sel==0) {

Loop1: for(inti = 0; i< N; i++) {

temp1[i] = data\_in[i] \* scale[0];

temp2[i] = data\_in[i] \* scale[1];

}

}

else {

Loop2: for(int j = 0; j < N; j++) {

temp1[j] = data\_in[j] \* scale[1];

temp2[j] = data\_in[j]\* scale[0];

}

}

Loop3: for(int k = 0; k < N; k++) {

data\_out[k] = temp1[k] / temp2[k];

}

}

***To ensure each loop is executed in all cases, you must transform the code as shown in the following example. In this example, the conditional statement is moved into the first loop and second loop. Both loops are always executed, and data always flows from one loop to the next.***

void foo\_m(int data\_in[N], int data\_out[N],int scale[2],char sel) {

int temp1[N], temp2[N];

Loop1: for(inti = 0; i< N; i++) {

if (sel==0) {

temp1[i] = data\_in[i] \* scale[0];

}

else {

temp1[i] = data\_in[i] \* scale[1];

}

}

Loop2: for(int j = 0; j < N; j++) {

if (sel==0) {

temp2[j] = data\_in[j]\* scale[1];

}

else {

temp2[i] = data\_in[i] \* scale[0];

}

}

Loop3: for(int k = 0; k < N; k++) {

data\_out[k] = temp1[k] \* temp2[k];

}

}

* Создать тест lab4\_z5\_test.cдля проверки функций выше (тест должен обеспечивать запуск функций при двух значениях sel: =0 и !=0).
* Для функции **foo\_b**
  + задать: clock period ,**выбранный в lab4\_z3**; clock\_uncertainty 0.1
  + осуществить моделирование (с выводом результатов в консоль)
  + осуществить синтездля:
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнитьZoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
* Для функции **foo\_m**
  + задать: clock period,**выбранный в lab4\_z3**; clock\_uncertainty 0.1
  + осуществить моделирование (с выводом результатов в консоль)
  + осуществить синтездляслучая **FIFO for the memory buffers**:
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнитьZoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * **Dataflow viewer**
  + осуществить синтездляслучая **ping-pongbuffers**:
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнитьZoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * **Dataflow viewer**
  + Осуществить C|RTL моделирование дляслучая **FIFOforthememorybuffers**
    - Привести результаты из консоли
    - Открыть временную диаграмму (все сигналы)
      * Отобразить два цикла обработки на одном экране
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
  + **Написать tclфайл автоматизирующий исследование**
* Выводы
  + Объяснить отличия в синтезе foo\_b и двух вариантов foo\_m между собой

Исследование времени выполнения на ПК

* **Используются исходные коды и результаты исследования проведенного ранее.**
* На базе использованного выше Си теста создать отдельный, модернизированный, тест для проверки времени выполнения синтезируемой функции на ПК:
  + добавить в тест операторы измерения **времени выполнения** синтезируемой функции (например, как-то так: <https://solarianprogrammer.com/2019/04/17/c17-programming-measuring-execution-time-delaying-program/> ).
  + Увеличить количество запусков синтезируемой функции до 32. Для каждого запуска измерить время, найти среднее значение и вывести как результат.
  + Точность измерения времени (наносекунды).
  + Провести исследование времени выполнения синтезируемой функции на Вашем ПК
    - Осуществить компиляцию модернизированного теста и запустить его как отдельное приложение
    - В отчете привести:
      * Параметры Вашего ПК: тип процессора, частота работы процессора, объем ОЗУ
      * результаты измерения времени выполнения
* Оформить отчет, который должен включать
  + Задание
  + Раздел с описанием исходного кода функции
  + Раздел с описанием теста
  + Раздел с описание созданного командного файла
  + Раздел с анализом результатов (со снимками экрана с заполненной таблицей и полученным графиком)
    - Анализ и выбор оптимального (критерий максимальная производительность) решения
  + Результаты исследования времени выполнения на ПК и сравнение с аппаратными решениями.
  + Выводы

Архив должен включать всю рабочую папку проекта (включая модернизированный тест и скомпилированное приложение), отчет

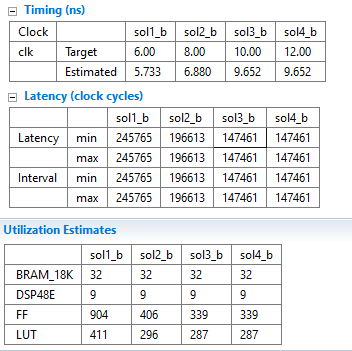
# Результаты работы

Модифицированные программные коды представлены в Приложении.

## Часть Lab4\_z3

### Solution 1

Результаты оценки быстродействия Timing/summary и Latency/Summary приведены на Рис. 1.



#### **Рис. 1.** Таблица Performance Estimates

Результаты оценки аппаратных ресурсов, требуемых для реализации синтезируемой функции, приведены на Рис. 2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | sol1 | sol2 | sol3 | sol4 |
| Clock | Target ( ns) | 6 | 8 | 10 | 12 |
| Estimated ( ns) | 5,73 | 6,88 | 9,65 | 9,65 |
| Latency | (cycles) | 245765 | 196613 | 147461 | 147461 |
| (ns) | 1408971 | 1352697 | 1423294 | 1423294 |
| Resources | BRAM\_18K | 32 | 32 | 32 | 32 |
| DSP48E | 9 | 9 | 9 | 9 |
| FF | 904 | 406 | 339 | 339 |
| LUT | 411 | 296 | 287 | 287 |
| URAM |  |  |  |  |

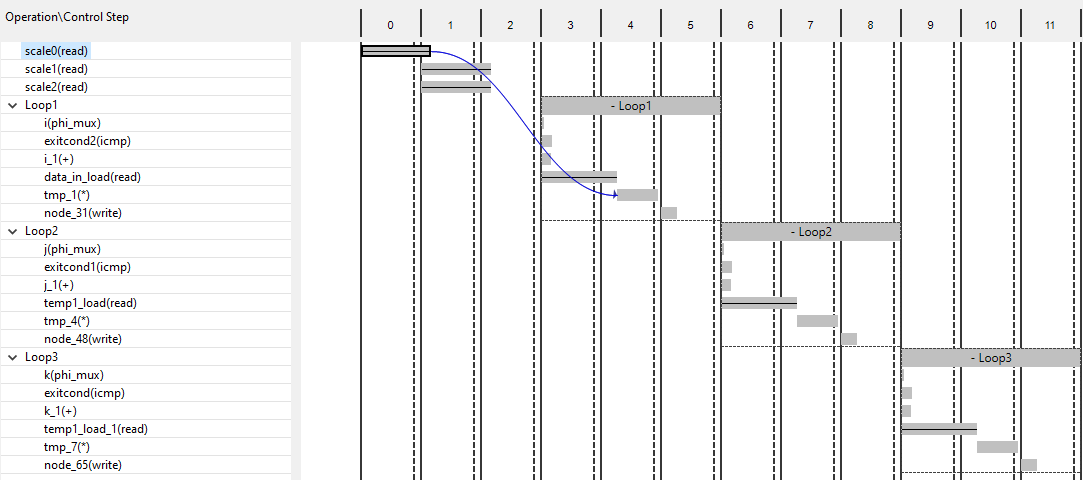
#### **Рис. 2.** Таблица Utilization Estimates

Графики приведены на рисунках 2-3.

#### **Рис. 2.** График значений Latency

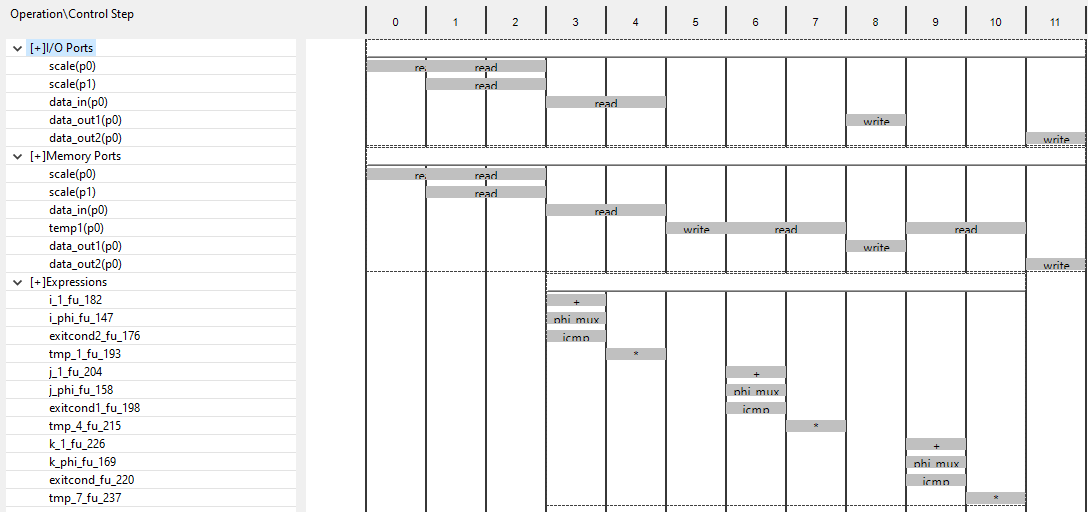
#### **Рис. 3.** График значений используемых триггеров FF и таблиц преобразований LUT

Результаты планирования выполнения функции приведены на Рис. 3.



#### **Рис. 4.** Таблица Schedule Viewer

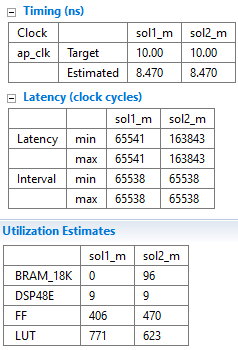
Результаты синтеза приведены на Рис.6 и показывают, что описание сформулировано безошибочно.



#### **Рис. 5.** Таблица Resource Viewer

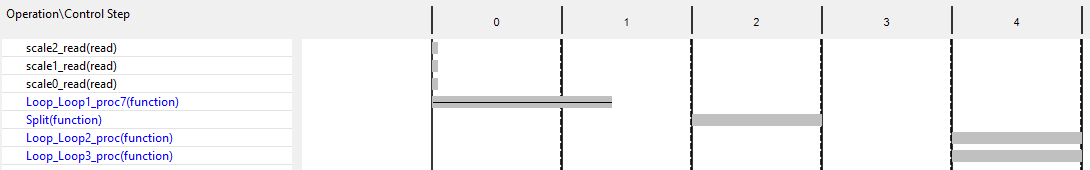
### Solution 2

Результаты оценки быстродействия Timing/summary и Latency/Summary приведены на Рис. 8.

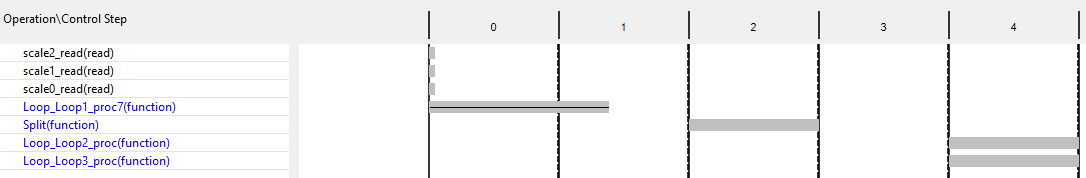


#### **Рис. 6.** Таблица Performance Estimates

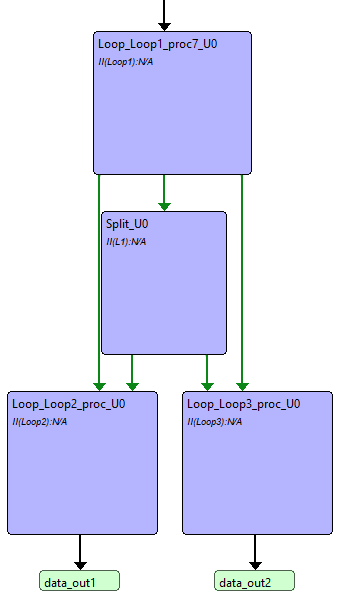
Результаты планирования выполнения функции приведены на Рис. 12.



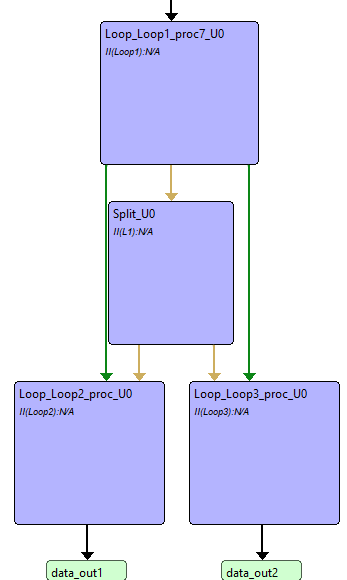
#### **Рис. 7.** Таблица Schedule Viewer для FIFO



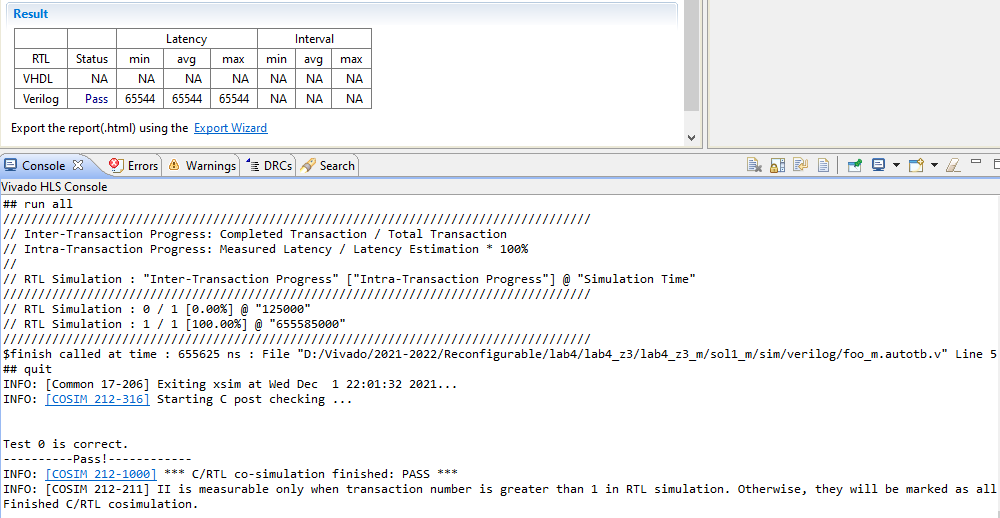
#### **Рис. 8.** Таблица Schedule Viewer для PingPong



#### **Рис. 9.** Dataflow Report для FIFO



#### **Рис. 10.** Dataflow Report для PingPong



**Рис. 11.** Результаты косимуляции для FIFO

### Выводы

Сопоставление симуляций приведено на Рис. 15. Вторая симуляция является более эффективной по причине меньшей ресурсозатратности относительно симуляции solution1.

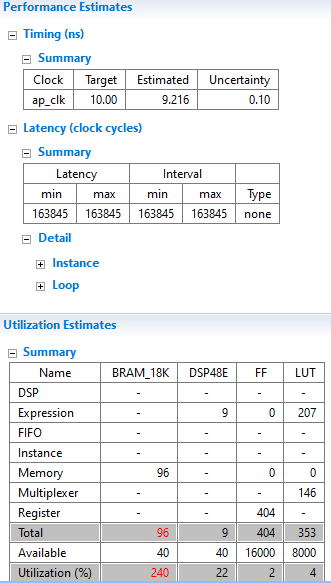
## Часть Lab4\_z4

### Solution 1

Входные значения первого решения:

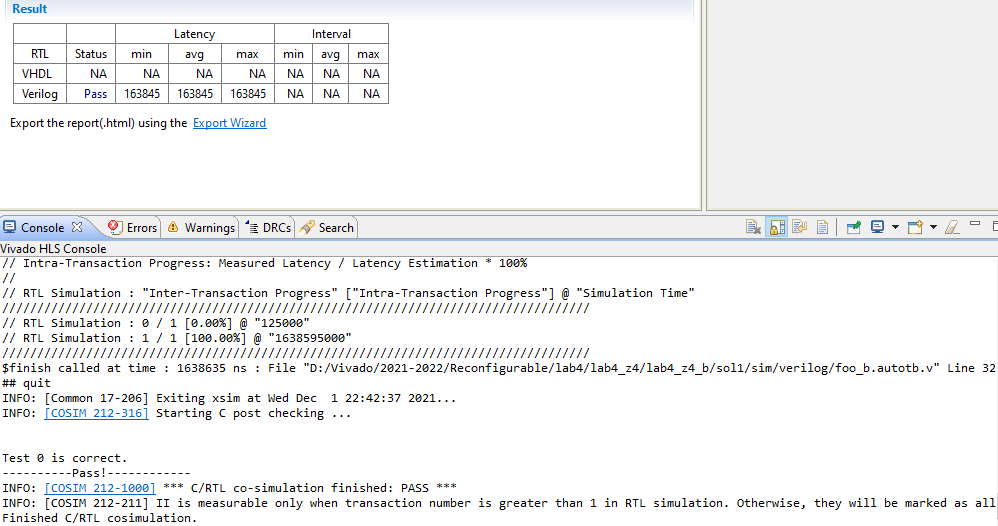
* clock\_period: 10.
* clock\_uncertainty: 0.1.
* part: xa7a12tcsg325-1q.

Результаты оценки быстродействия Timing/summary и Latency/Summary приведены на Рис. 16.



#### **Рис. 12.** Таблица Performance Estimates

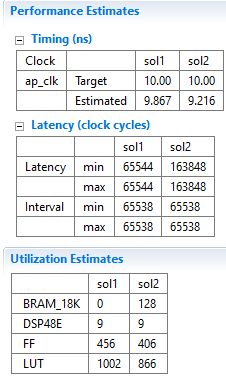
Результаты оценки аппаратных ресурсов, требуемых для реализации синтезируемой функции, приведены на Рис. 17.



#### **Рис. 13.** Результаты косимуляции

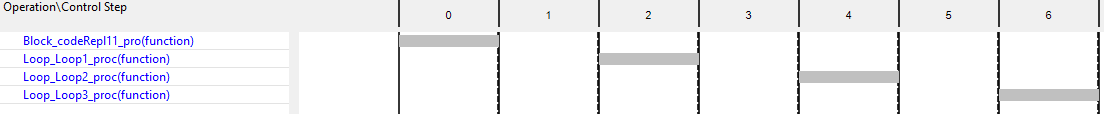
### Solution 2

Результаты оценки быстродействия Timing/summary и Latency/Summary приведены на Рис. 23.

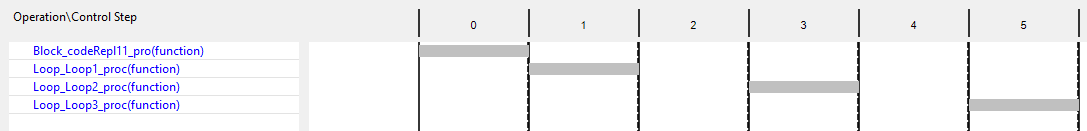


#### **Рис. 23.** Таблица Performance Estimates

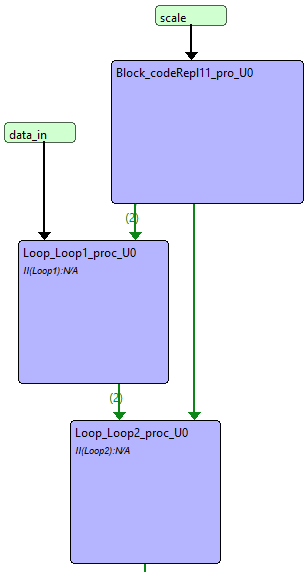
Результаты оценки аппаратных ресурсов, требуемых для реализации синтезируемой функции, приведены на Рис. 24.

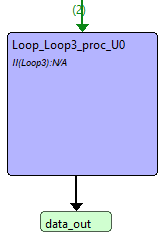


#### **Рис. 7.** Таблица Schedule Viewer для FIFO

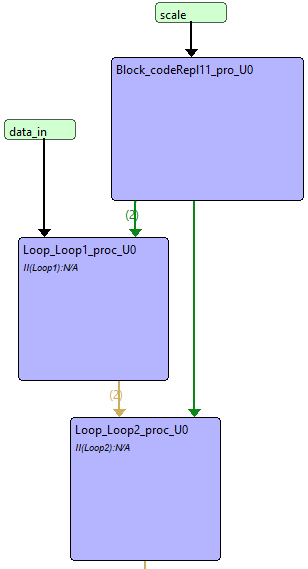


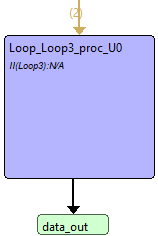
#### **Рис. 8.** Таблица Schedule Viewer для PingPong



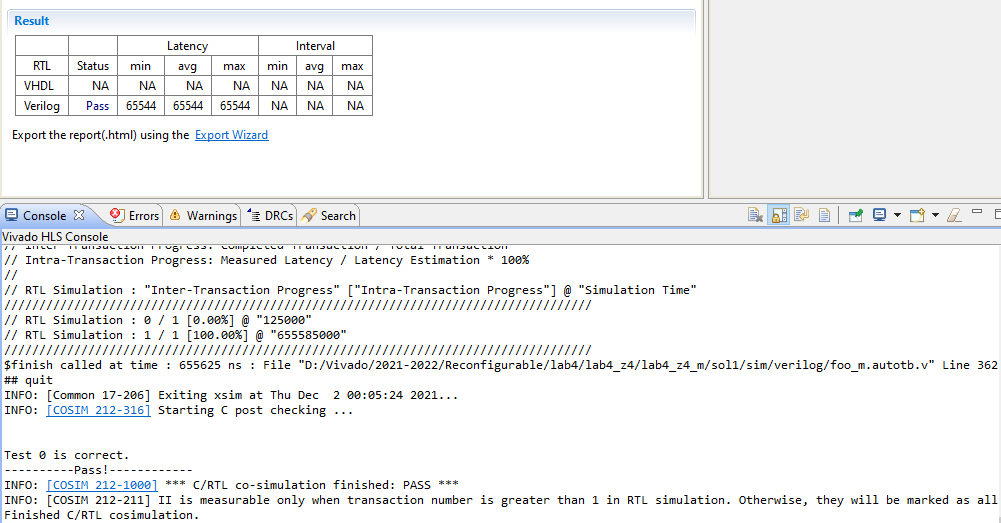


#### **Рис. 9.** Dataflow Report для FIFO





#### **Рис. 10.** Dataflow Report для PingPong



#### **Рис. 11.** Результаты косимуляции для FIFO

### Выводы

Сопоставление симуляций приведено на Рис. 30. Вторая симуляция является более эффективной по причине меньшей ресурсозатратности относительно симуляции solution1.

## Часть Lab4\_z4

### Solution 1

Входные значения первого решения:

* clock\_period: 10.
* clock\_uncertainty: 0.1.
* part: xa7a12tcsg325-1q.

Результаты оценки быстродействия Timing/summary и Latency/Summary приведены на Рис. 16.

#### **Рис. 12.** Таблица Performance Estimates

Результаты оценки аппаратных ресурсов, требуемых для реализации синтезируемой функции, приведены на Рис. 17.

#### **Рис. 13.** Результаты косимуляции

### Solution 2

Результаты оценки быстродействия Timing/summary и Latency/Summary приведены на Рис. 23.

#### **Рис. 23.** Таблица Performance Estimates

Результаты оценки аппаратных ресурсов, требуемых для реализации синтезируемой функции, приведены на Рис. 24.

#### **Рис. 7.** Таблица Schedule Viewer для FIFO

#### **Рис. 8.** Таблица Schedule Viewer для PingPong

#### **Рис. 9.** Dataflow Report для FIFO

#### **Рис. 10.** Dataflow Report для PingPong

#### **Рис. 11.** Результаты косимуляции для FIFO

### Выводы

Сопоставление симуляций приведено на Рис. 30. Вторая симуляция является более эффективной по причине меньшей ресурсозатратности относительно симуляции solution1.

# Приложение

## lab3\_1.tcl

# Insert the command to create new project

open\_project -reset lab3\_z1\_prj

# Insert the command to add design file

add\_files ./source/lab3\_z1.c

# Insert the command to specify the top-level function

set\_top lab3\_z1

# Insert the command to add test bench file

add\_file -tb ./source/lab3\_z1\_test.c

#solution sol1 ( default)

# Insert the command to create the solution sol1

open\_solution -reset "sol1"

# Insert the command to associate the device to the solution

set\_part {xa7a12tcsg325-1Q}

# Insert the command to associate clock period to the solution

create\_clock -period 6 -name clk

set\_clock\_uncertainty 0.1

# Insert the command to run C simulaiton

csim\_design -clean

# Insert the command to Synthesize the design

csynth\_design

# Insert the command to perform C/RTL Cosimulation

cosim\_design -trace\_level all -tool xsim

#solution sol2 ( default)

# Insert the command to create the solution sol2

open\_solution -reset "sol2"

# Insert the command to associate the device to the solution

set\_part {xa7a12tcsg325-1Q}

# Insert the command to associate clock period to the solution

create\_clock -period 6 -name clk

set\_clock\_uncertainty 0.1

# Insert the command to run C simulaiton

# Insert the command to Synthesize the design

csynth\_design

# Insert the command to perform C/RTL Cosimulation

cosim\_design -trace\_level all -tool xsim

exit

## lab3\_1.c

**#include** "lab3\_z1.h"

//typedef char array[N];

**void** **lab3\_z1**(data\_sc array\_in[N], data\_sc array\_out[M]){

data\_sc count;

**for**(**int** i=1; i<N; i++){

count = array\_in[i];

array\_out[count]++;

}

}

## lab3\_1\_test.c

**#include** <stdio.h>

**#include** "lab3\_z1.h"

**int** **main**()

{

**int** pass=0;

data\_sc inputArray[N];

data\_sc outputArray[M] = {0};

data\_sc specialArray[M] = {0};

// Call the function for 2 transactions

**for** (**int** i = 1; i<N; i++) {

inputArray[i] = rand()%(M);

specialArray[inputArray[i]]++;

}

**if**(pass == 0)

**for**(**int** j = 0; j<2; j++){

**for**(**int** c = 0; c<M; c++)

outputArray[c] = 0;

lab3\_z1(inputArray,outputArray);

**fprintf**(stdout, "\n\n");

**for**(**int** h = 0; h<M; h++){

**fprintf**(stdout, "%d:\t specialArray: %d \t outputArray: %d\n", h, specialArray[h], outputArray[h]);

**if** (outputArray[h] != specialArray[h])

pass = 1;

}

**if**(pass==0)

**fprintf**(stdout, "Test %d is correct.\n", j);

**else**

**fprintf**(stdout, "Test %d is incorrect.\n", j);

}

**if** (pass == 0) {

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

**return** 0;}

**else** {

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

**return** 1;

}

}

## lab2\_2.tcl

## # Insert the command to create new project

## open\_project -reset lab2\_2\_prj

## # Insert the command to add design file

## add\_files ./source/lab2\_2.c

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

## set\_top lab2\_2

## # Insert the command to add test bench file

## add\_files -tb ./source/lab2\_2\_test.c

## # Insert the command to create the solution named sol0

## open\_solution -reset "sol0"

## # Insert the command to associate the device to the solution1

## set\_part {xa7a12tcsg325-1Q}

## # Insert the command to associate clock period to the solution1

## create\_clock -period 6 -name clk

## set\_clock\_uncertainty 0.1

## # Insert the command to run C simulaiton

## csim\_design -clean

## # Insert the command to Synthesize the design

## csynth\_design

## # Insert the command to perform C/RTL Cosimulation

## cosim\_design -trace\_level all -tool xsim

## # Build list of solutions

## set all\_solutions {ex\_sol1 ex\_sol2 ex\_sol3 ex\_sol4 ex\_sol5 ex\_sol6 ex\_sol7}

## set all\_period {{8} {10} {12} {14} {16} {18} {20}}

## foreach the\_solution $all\_solutions the\_period $all\_period {

## open\_solution -reset $the\_solution

## # Insert the command to associate clock period to the solution1

## create\_clock -period $the\_period -name clk

## set\_clock\_uncertainty 0.1

## # Insert the command to associate the device to the solution1

## set\_part {xa7a12tcsg325-1Q}

## # Insert the command to Synthesize the design

## csynth\_design

## # Insert the command to perform C/RTL Cosimulation

## cosim\_design -trace\_level all -tool xsim

## }

## lab2\_2\_inc.c

**#define** N 8192

// Вариант 4

**typedef** **int** atype;

## lab2\_2.c

## lab2\_2\_test.c