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

Лабораторная работа 7

Предмет: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Pipelining for Performance Demo Script

Задание 2

Студент: Безрукова Ю.С.

Гр. № 3540901/81502

Преподаватель: Антонов А.П.

Санкт-Петербург

2019

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

Лабораторная работа 7

Предмет: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Pipelining for Performance Demo Script

Задание 2

Студент: Белоглазов К.И.

Гр. 3540901/81501

Преподаватель: Антонов А.П.

Санкт-Петербург

2019

Оглавление

[Задание 4](#_Toc26222994)

[Решение 1 без использования директивы 5](#_Toc26222995)

[Решение 2 8](#_Toc26222996)

[Решение 3 11](#_Toc26222997)

[Вывод 14](#_Toc26222998)

# Задание

* Создать проект lab7\_2
* Микросхема: xa7a12tcsg325-1q
* Создать функцию на основе приведенных ниже слайдов.



* Создать тест lab7\_2\_test.c для проверки функций выше.
  + осуществить моделирование (с выводом результатов в консоль)
* Сделать свой solution (для каждого варианта задания директивы и для варианта без директивы)
  + задать: clock period 10; clock\_uncertainty 0.1
  + осуществить синтез
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * 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 (для каждого варианта задания директивы)
    - 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
* Выводы
  + Привести обобщенную таблицу зависимости utilization и performance от каждого варианта задания директивы и для варианта без директивы.
  + Объяснить отличие процедур обращения к элементам массива для каждого случая

# Решение 1 без использования директивы

Исходный файл

void foo1(int m, int n, int in1[n][m], int in2[n][m], int out[n][m])

{

for (int i = 0; i < n; i++)

{

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

{

out[i][j] = in1[i][j] + in2[i][j];

}

}

}

Файл – тест

#include <stdio.h>

int main()

{

int a1[3][3] = {{1,2,3}, {1,3,2}, {2,3,1}};

int a2[3][3] = {{6,0,5},{2,1,2},{3,5,0}};

int out[3][3];

int res[3][3] = {{7,2,5}, {3,4,4}, {5,8,1}};

foo1(3, 3, a1, a2, out);

int pass;

fprintf(stdout, "Expected Actual\n");

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

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

fprintf(stdout, "res[%d][%d]: %d == out[%d][%d]: %d\n", i, j, res[i][j], i, j, out[i][j]);

if (res[i][j] == out[i][j]) {

pass = 1;

} else {

pass = 0;

break;

}

}

}

if (pass) {

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

return 0;

} else {

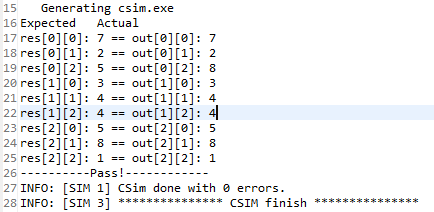
fprintf(stderr, "----------Fail!------------\n");

return 1;

}

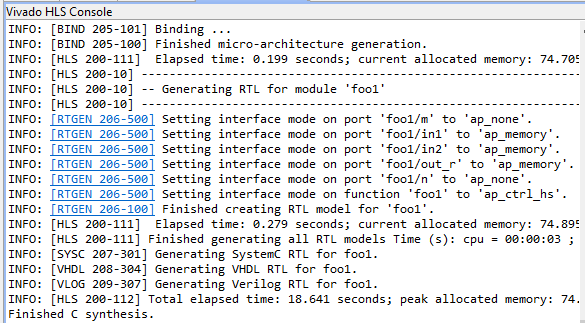
}

Моделирование

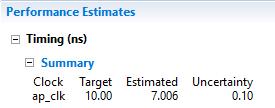


На изображении выше видно, что моделирование выполнено без ошибок.

Синтез

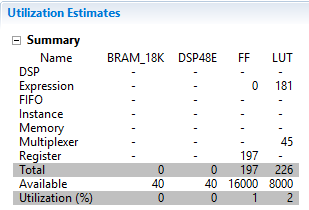


Производительность

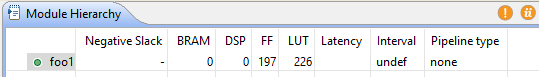


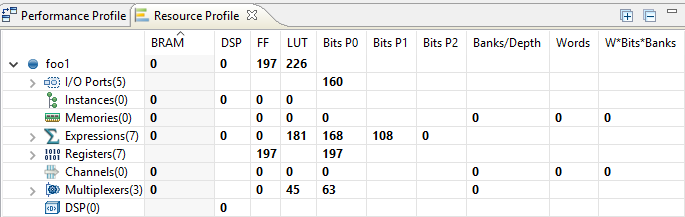
На изображении видно, что полученная задержка укладывается в заданное значение.

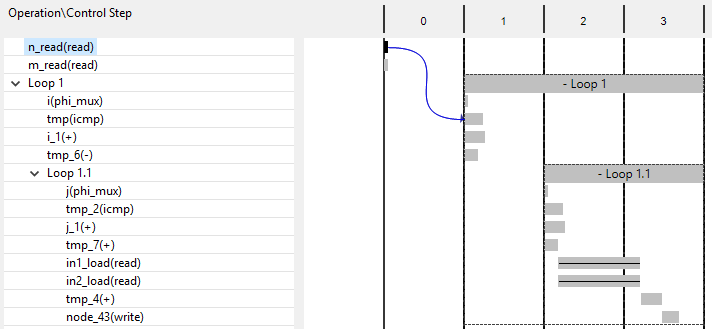
Использование ресурсов



Данное решение потребует на микросхеме 197 регистров, 226 элементов LUT.



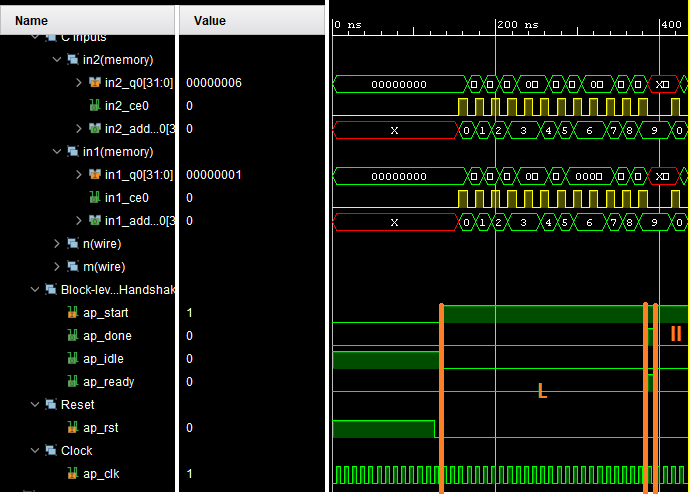




На рисунке выше изображено пошаговое выполнение программы

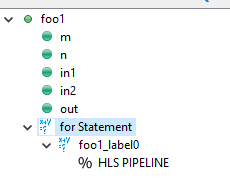
C/RTL моделирование

На временной диаграмме отображены интервал инициализации и задержка.

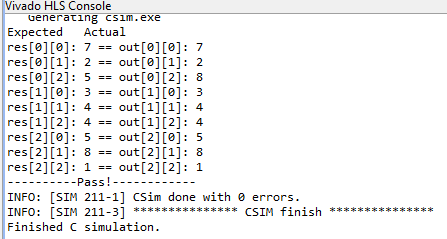


# Решение 2

Создание директивы

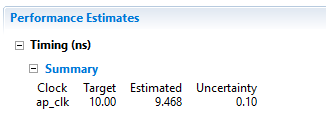


Моделирование



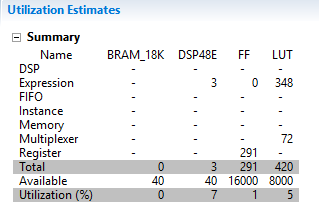
Синтез

Производительность

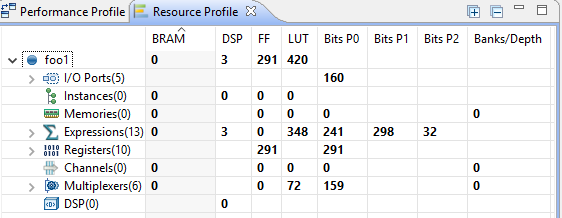


По сравнению с предыдущим решением значение задержки значительно увеличено, но все еще укладывается в заданное значение.

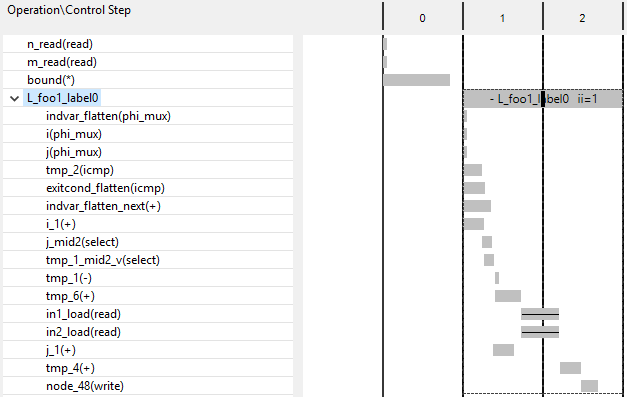
Использование ресурсов



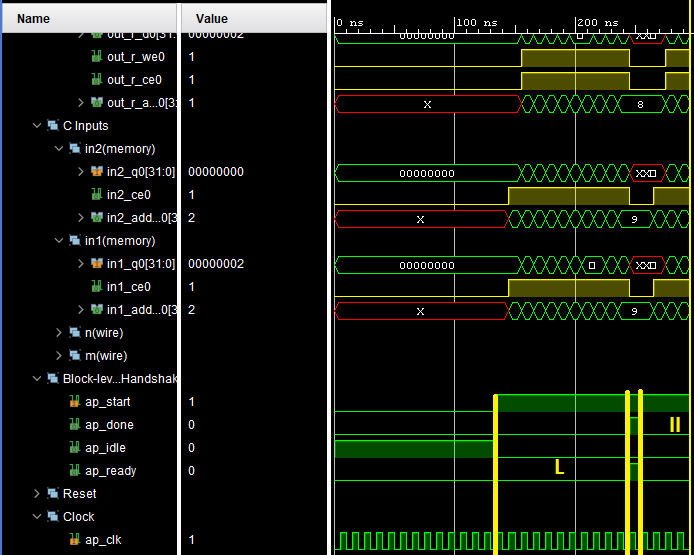
Данное решение потребует на микросхеме 3 блока DSP48E, 291 регистр и 420 элементов LUT.



Пошаговое выполнение:



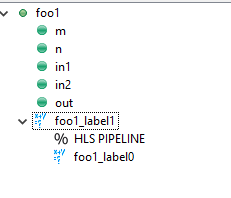
С/RTL моделирование



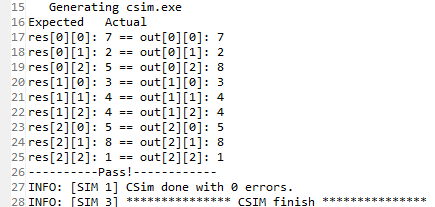
На временной диаграмме отображена задержка интервал инициализации.

# Решение 3

Добавление директивы

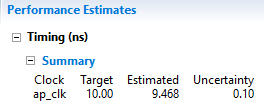


Моделирование

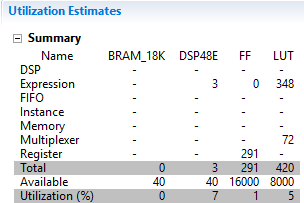


Синтез

Производительность



Использование ресурсов



Данное решение потребует на микросхеме 3 элемента DSP48E, 291 регистр и 420 элементов LUT.

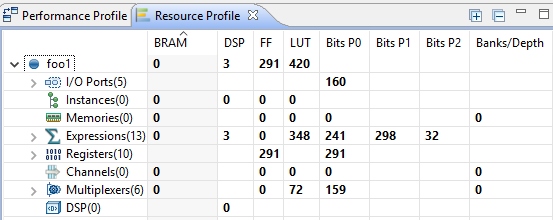
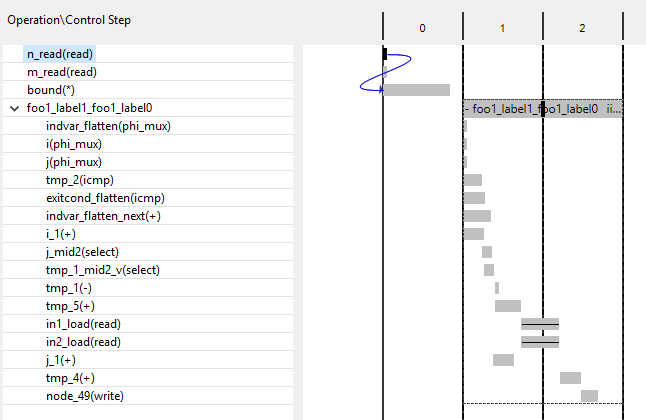
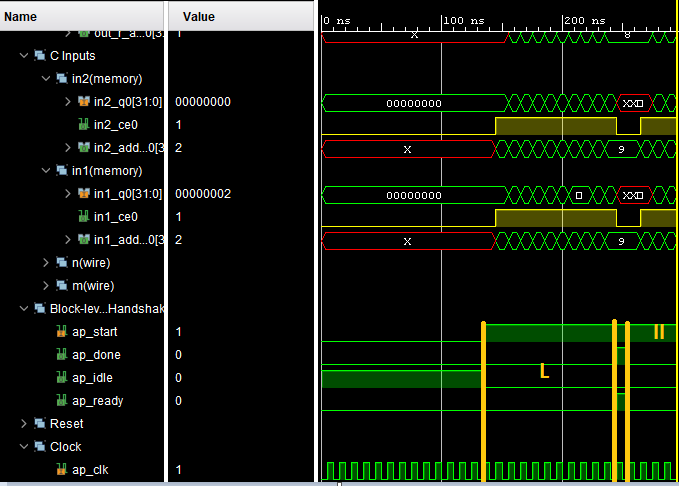


Диаграмма выполнения решения

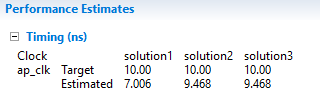


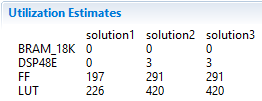
C\RTL моделирование



# Вывод

Обобщенная таблица результатов





Как конвейерная обработка циклов, так и развертывание циклов улучшают производительность аппаратной функции, используя параллелизм между итерациями цикла. В последовательных языках, таких как C / C ++, операции в цикле выполняются последовательно, и следующая итерация цикла может начаться только после завершения последней операции в текущей итерации цикла. Контурная конвейерная обработка позволяет выполнять операции в цикле одновременно, как показано на следующем рисунке.

