**Single-producer-consumer**

***Для использования DATAFLOW оптимизации, все элементы, передаваемые между циклами должны соответствовать модели single-producer-consumer model.***

***Каждая переменная должны получать значение только из одного цикла и должна передаваться только в один цикл. В примере ниже, temp1 используется в двух циклах Loop2 и Loop3. Что нарушает указанную модель.***

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

*int temp1[N];*

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

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

*}*

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

*data\_out1[j] = temp1[j] \* 22;*

*}*

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

*data\_out2[k] = temp1[k] \* 33;*

*}*

*}*

***Модифицированная версия кода использует функцию Split для того, что бы удовлетворять модели single-producer-consumer:***

***Вариант1:***

***данные из Loop1 поступаю в функцию Split и затем в циклы Loop2 и Loop3.***

***Для данного кода может быть применена DATAFLOW Optimization***

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

*// Duplicated data*

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

*out1[i] = in[i];*

*out2[i] = in[i];*

*}*

*}*

*void lab4\_z2(int data\_in[N], int scale, int data\_out1[N], int data\_out2[N]) {*

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

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

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

*}*

*Split(temp1, temp2, temp3);*

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

*data\_out1[j] = temp2[j] \* 22;*

*}*

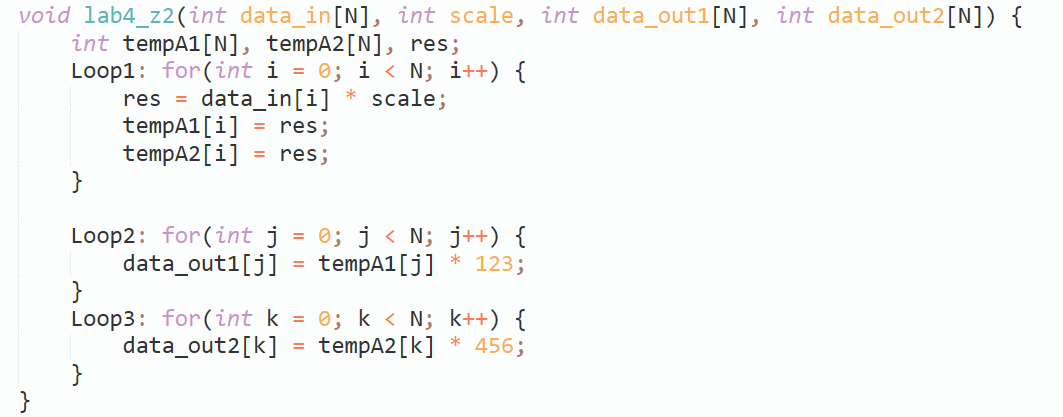
*Loop3: for(int k = 0; k < N; k++) {*

*data\_out2[k] = temp3[k] \* 33;*

*}*

*}*

Вариант 2



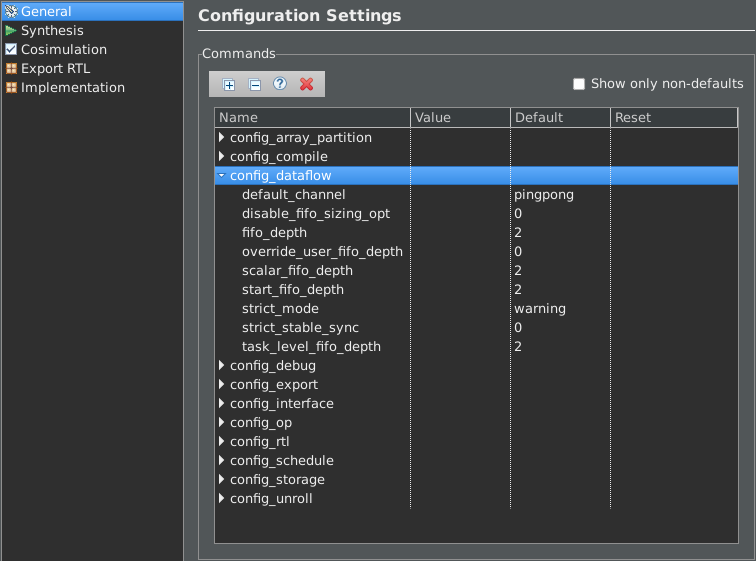
***Выберите из них ту реализацию, которая Вам кажется более производительной и объясните Ваш выбор (если не понятно, какую реализацию выбрать и как объяснить свой выбор, синтезируйте все два варианта реализации и на основе полученных данных сделайте выбор и объясните его)***

* Создать тест lab4\_z2\_test.c для проверки функции
  + Запуск функции 3 раза
    - Заполнение векторов случайными значениями.
    - Перед следующим запуском функции не забудьте обнулить ожидаемое значение.
  + проверку правильности полученного функцией результата.
* Тип данных (для всех аргументов он – int), значение N (***N = 16***) следует задать в файле lab4\_z2.h
* Создать скрипт автоматизирующий процесс:
  + Создания проекта lab4\_z2
    - Микросхема: xa7a12tcsg325-1q
  + Создания решении **sol1, sol2, sol3**
    - задать: clock period 6, 10, 14 **(решения sol1, sol2, sol3)**; clock\_uncertainty 1
    - осуществить моделирование
    - 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 (выполнить Zoom to Fit)

На скриншоте показать Latency

На скриншоте показать Initiation Interval

* + Решения sol4
    - осуществить синтез с DATAFLOW и параметром **FIFO для default\_channel**:



* + - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнить Zoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * **Dataflow viewer**
    - Осуществить C|RTL моделирование
      * 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

* + Решения sol5
  + осуществить синтез с DATAFLOW и параметром **ping-pong для default\_channel** :
    - привести в отчете:
      * performance estimates=>summary
      * utilization estimates=>summary
      * scheduler viewer (выполнить Zoom to Fit)
        + На скриншоте показать Latency
        + На скриншоте показать Initiation Interval
      * **Dataflow viewer**
  + Осуществить C|RTL моделирование
    - 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
  + Объяснить отличия в синтезе двух вариантов sol4 и sol5 между собой

**Измерение времени выполнения на ПК**

* **Используются исходные коды функции lab4\_z2.cpp (лучшее по быстродействию решение)**
* На базе теста **lab4\_z2\_test.cpp следует** создать отдельный, модернизированный, тест **lab4\_z2\_testSW.cpp** (сохранить в папке C:\Xilinx\_trn\HLS2023\lab4\_z2\source) для проверки времени выполнения функции lab4\_z2 на ПК:
  + добавьте 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/> ).
  + Увеличьте количество запусков синтезируемой функции до 32. Для каждого запуска нужно измерить время выполнения (Точность измерения времени – наносекунды);
* Следует осуществить компиляцию модернизированного теста и запускать его как отдельное приложение
* Следует провести измерение времени выполнения синтезируемой функции на Вашем ПК **для каждого** из случаев
  + N = 4096
  + N = 8192
  + N = 16384
* среди 32 запусков необходимо найти и зафиксировать максимальное, минимальное значения времени выполнения и медиану.

**Измерение времени выполнения на аппаратной реализации**

* Используются исходные коды функции lab4\_z2.cpp
* Используются исходные коды теста lab4\_z2\_test.cpp (C:\Xilinx\_trn\HLS2023\lab4\_z2\source)
* Для решения лучшего по производительности следует осуществить синтез для случаев
  + N = 4096
  + N = 8192
  + N = 16384

и для каждого случая зафиксировать: II, Estimated period, время выполнения = II \* Estimated period

**Сравнительный анализ**

* Составить xls таблицу и построить два графика (
  + по оси Х – случаи
    - N = 4096
    - N = 8192
    - N = 16384
  + по У – время выполнения функции на ПК и аппаратного
* Оформить отчет, который должен включать
  + Задание
  + Раздел с описанием исходного кода функции
  + Раздел с описанием теста
  + Раздел с описание созданного командного файла
  + Раздел с описанием результатов сравнения решений (со снимками экрана)
  + Раздел с анализом результатов
    - Анализ и выбор оптимального (критерий максимальная производительность) решения
  + Раздел с описанием модернизированного теста
    - Следует указать компилятор, используемый для компиляции.
  + Результаты измерения **времени выполнения на ПК**
    - Следует указать: тип процессора, базовую частоту работы, максимальную частоту работы, объем ОЗУ.
  + Результаты измерения времени выполнения на аппаратной реализации
  + Раздел с анализом результатов
  + Выводы

Архив должен включать всю рабочую папку проекта (включая модернизированный тест и **скомпилированные приложения – папка ..\source**), отчет