* Создать на языке С++ функцию,
  + Умножающую вектор А (N элементов) на вектор В (N элементов) по следующему алгоритму

A[a1 a2 a3 a4]

B[b1 b2 b3 b4]

=

res[

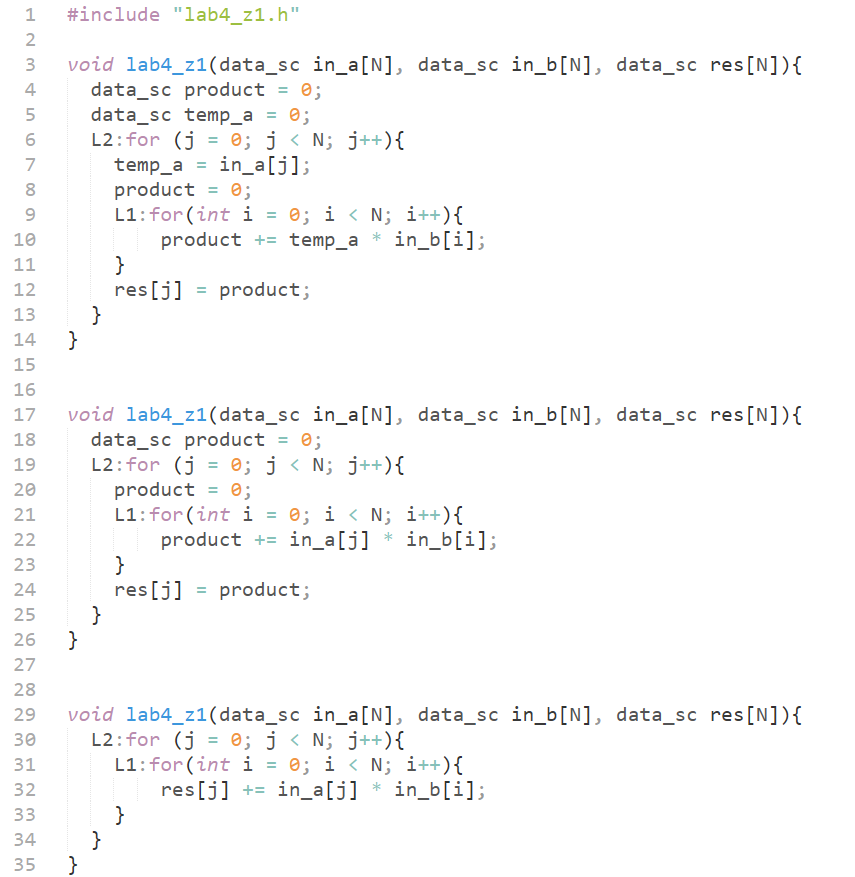
a1\*b1 + a1\*b2 + a1\*b3 +a1\* b4

a2\*b1 + a2\*b2 + a2\*b3 +a2\* b4

a3\*b1 + a3\*b2 + a3\*b3 +a3\* b4

a4\*b1 + a4\*b2 + a4\*b3 +a4\* b4]

* + Результат – вектор res ( **N** элементов).
  + Аргументы функции: два входных вектора (in\_a[N], in\_b[N]), выходной вектор (res[N])
  + В файле lab4\_1.h должны быть определены: **N = 4**, тип данных data\_sc, имеющий тип int
  + Примеры реализации функции приведены ниже. 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 три варианта реализации и на основе полученных данных сделайте выбор и объясните его)
  + Дать самому внутреннему циклу, реализованному в функции, имя L1, остальным L2…



* Создать на языке С++ тест для проверки работы функции. Тест должен обеспечивать
  + Запуск функции 3 раза
    - Заполнение векторов случайными значениями.
    - Перед следующим запуском функции не забудьте обнулить ожидаемое значение.
  + проверку правильности полученного функцией результата.
* Отладить функцию и тест (при неправильном результате в любом из запусков функции тест должен сообщать об ошибке).
* Создать скрипт автоматизирующий процесс:
  + Создания проекта lab4\_1,
  + Назначения функции lab4\_1
  + Подключения файла lab4\_1.c (папка source ),
  + Подключения файла lab4\_1\_test.c (папка source),
  + Создания решения
    - sol1, для которого
      * Микросхема: xa7a12tcsg325-1Q
      * Период тактового сигнала: 10нс, uncertainty 1нс.
      * Си моделирование
      * ОТКЛЮЧИТЬ конвейеризацию для всех циклов
        + **Pipeline off – для каждого цикла**
      * Синтез
      * С/RTL cosimulation (с опцией Dump Trace = Port)
    - sol2, для которого
      * для цикла L1 задать директиву **Pipeline II=1**
      * Синтез
      * С/RTL cosimulation (с опцией Dump Trace = Port)
    - Sol3, для которого
      * для цикла L1 задать директиву **Pipeline off**
      * **для внешнего цикла задайте Pipeline II=1 и rewind**
      * Синтез
      * С/RTL cosimulation (с опцией Dump Trace = Port)
* Отладить и проверить работу созданного скрипта.
* После выполнения скрипта открыть GUI
* Убедиться, что созданы все решения
* Используя средства HLS проверить, сравнить и зафиксировать использованные интерфейсы для каждого из трех решений
* Используя средства HLS сравнить полученные решения (привести Report Comparison)
  + Timing
  + Latency
  + Utilization Estimation
* Используя средства HLS сравнить и привести результаты планирования (Schedule viewer) для каждого из трех решений

Для каждого из трех решений привести временные диаграммы

* + Решение 1
    - Надо показать
      * II, Latency для первого запуска функции
  + Решение 2
    - Надо показать
      * II, Latency для первого запуска функции
      * Два запуска функции
  + Решение 3
    - Надо показать
      * Три (два) запуска функции

**Измерение времени выполнения на ПК**

* **Используются исходные коды функции lab4\_z1.cpp (лучшее по быстродействию решение)**
* На базе теста **lab4\_z1\_test.cpp следует** создать отдельный, модернизированный, тест **lab4\_z1\_testSW.cpp** (сохранить в папке C:\Xilinx\_trn\HLS2023\lab4\_z1\source) для проверки времени выполнения функции lab4\_z1 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\_z1.cpp
* Используются исходные коды теста lab4\_z1\_test.cpp (C:\Xilinx\_trn\HLS2023\lab4\_z1\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