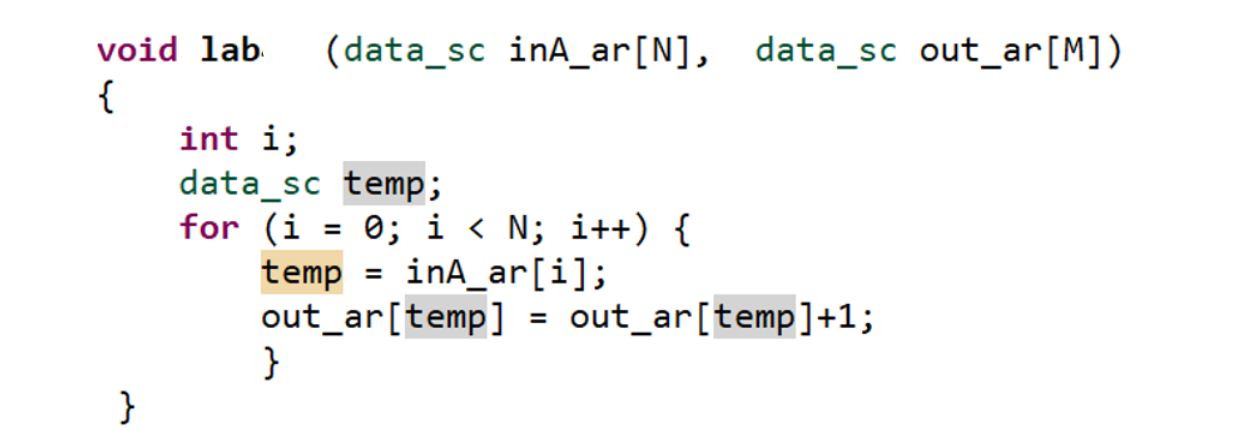
Lab3\_z2

* В папке C:\Xilinx\_trn\HLS2023\lab3\_z2 создайте папку source (для исходных файлов и теста)
* Создайте на языке С++ (и сохраните в папку C:\Xilinx\_trn\HLS2023\lab3\_z2\source)
  + Последовательно считывающую элементы массива, заполненного случайными целыми числами (тип short), имеющими значения от **0 до M-1**. Размер массива - **N** элементов.
  + Функция подсчитывает количество появлений чисел от 0 до **M-1** во входном массиве и записывает полученные значения в соответствующие элементы (с 0 до **M-1)**  выходного массива, имеющего **M** элементов типа short.

*Т.е. по входному массиву размером N, заполненного случайными целыми числами от* **0 до M-1** *строится гистограмма в выходном массиве размером M элементов.*

* + В файле lab3\_z2.h должны быть определены **M**  и **N** и тип данных data\_sc, имеющий тип short.

Например, можно примерно так (надо поменять название функции на lab3\_z2 и ….. – на ваше усмотрение):



* + тест lab3\_z2\_test.cpp для проверки работы функции. Тест должен обеспечивать
    - Запуск функции 3 раза
      * каждый раз с новым начальным значением от генератора случайных чисел (от 0 до M-1).
      * Перед следующим запуском функции не забудьте обнулить формируемый функцией выходной массив.
    - Формирование исходного входного массива, заполненного случайными числами.
      * N=16, M=8 (задать в h файле)
    - проверку правильности полученного функцией результата (сравнение полученного функцией массива и ожидаемого массива, вычисленного в тесте с использованием другого, не такого как в функции, алгоритма построения гистограммы ) и формирование признака успешного/неуспешного выполнения для каждого запуска функции.
* Отладьте функцию и тест (при неправильном результате в любом из запусков функции должен сообщать об ошибке).
* Создайте скрипт lab3\_z2.tcl автоматизирующий процесс:
  + Создания проекта lab3\_z2,
    - * Микросхема: xa7a12tcsg325-1Q
      * Период тактового сигнала: **10** нс, uncertainty **1**нс.
  + Подключения файла lab3\_z2.c (папка source ),
  + Подключения файла lab3\_z2\_test.c (папка source),
  + Си моделирование
  + Создания решений
    - sol1, для которого
      * Block Level I/O и Port Level I/O interface по умолчанию
      * Синтез
      * С/RTL cosimulation (с опцией полной трассировки)
    - sol2, для которого
      * Block Level I/O по умолчанию
      * Port Level I/O interface **ap\_fifo** ТОЛЬКО для входного массива
        + Для упрощения рекомендуется задать эти настройки в GUI пакета и

скопировать их из автоматически создаваемого пакетом файла

directives.tcl в файл lab3\_z2.tcl

* + - * Синтез
      * С/RTL cosimulation (с опцией полной трассировки)
* Отладьте и проверьте работу созданного скрипта.
* После выполнения скрипта откройте GUI
* Убедитесь, что созданы все решения
* Используя средства HLS сравните полученные решения
  + использованные интерфейсы
  + аппаратные затраты
  + результаты планирования (Schedule viewer)
  + С/RTL cosimulation
  + Заполненную xls таблицу и полученный график
  + объясните полученные результаты.
* Создайте решение sol3, для которого
  + Block Level I/O по умолчанию
  + Port Level I/O interface **ap\_fifo** ТОЛЬКО для выходного массива
    - Для упрощения рекомендуется задать эти настройки в GUI пакета и

скопировать их из автоматически создаваемого пакетом файла

directives.tcl в файл lab3\_z2.tcl

* + Осуществите синтез и объясните полученный результат

**Измерение времени выполнения на ПК**

* **Используются исходные коды функции lab3\_z2.cpp**
* На базе теста **lab3\_z2\_test.cpp следует** создать отдельный, модернизированный, тест **lab3\_z2\_testSW.cpp** (сохранить в папке C:\Xilinx\_trn\HLS2023\lab3\_z2\source) для проверки времени выполнения функции lab3\_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
  + M= N = 4096
  + M= N = 8192
  + M= 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 lab3\_z2.cpp
* Используются исходные коды теста lab3\_z2\_test.cpp (C:\Xilinx\_trn\HLS2023\lab3\_z2\source)
* Для решения sol2 следует осуществить синтез для случаев
  + M= N = 4096
  + M= N = 8192
  + M= 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
    - M= N = 4096
    - M= N = 8192
    - M= 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

ПРИМЕР КОДА ДЛЯ ИЗМЕРЕНИЯ ВРЕМЕНИ ПРОГРАММНОЙ РЕАЛИЗАЦИИ

