- Создать проект lab4_z4
- Микросхема: xa7a12tcsg325-1q
- ***N = 16384, scale = { два случайных числа).***
- Создать две функции (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 temp1 and temp2. However, the next task, Loop2, only uses the value of temp1.
The value of temp2 is not consumed until after Loop2. Therefore, temp2 bypasses the next task
in the sequence, which limits the performance of the DATAFLOW optimization

```
void foo_b(int data_in[N], int scale[3], 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[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 Loop2 consumes temp2 and produces temp4 as follows. This ensures that the data flow from one task to the next.

```
void foo_b(int data_in[N], int scale[3], int data_out1[N], int data_out2[N]) {
int temp1[N], temp2[N], temp3[N], temp4[N];

Loop1: for(int i = 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.c для 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 (выполнить Zoom 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 (выполнить Zoom to Fit)
                - На скриншоте показать Latency
                - На скриншоте показать Initiation Interval
            - **Dataflow viewer**
    - осуществить синтез для случая **ping-pong buffers**:
        - 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 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