**# Лаба 2**

clock - целевая длительность работы одного такта синтезируемой функции

unsertainty - допустимый разброс времени выполнения одного такта синтезируемой функции

latency - задержка получения данных от синтезируемой функции/подфункции

interval (II) - полное время выполнения синтезируемой функции/подфункции

dsp - цифровой сигнальный процессор

ff - триггеры

lut (look-up table) - таблица перекодировок

**# Лаба 3**

set\_directive\_interface -mode <ap\_ctrl\_chain/ap\_ctrl\_hs/ap\_ctrl\_none> <<interface>>

\* директива, определяющая режим, в котором будет работать интерфейс (входной или выходной параметр функции)

\* -mode <ap\_ctrl\_chain/ap\_ctrl\_hs/ap\_ctrl\_none> - аргумент выбора режима, в данном случае перечисление протоколов блочного уровня (block-level protocols)

\* ap\_ctrl\_hs - режим работы интерфейса по умолчанию - интегрирует порты протокола блочного уровня в начало операций взаимодействия с интерфейсом - обеспечивает индикацию простоя, готовности и выполнения операций ввода/вывода

\* ap\_ctrl\_chain - интегрирует порты протокола блочного уровня в начало операций взаимодействия с интерфейсом - обеспечивает возможность приостановки и продолжения чтения с порта - обеспечивает индикацию простоя, готовности и выполнения операций ввода/вывода

\* ap\_ctrl\_none - отключение всех портов протоколов блочного уровня

set\_directive\_interface -mode <ap\_fifo> <<interface>>

\* ap\_fifo - буфер чтения (обычно равный двум элементам) входных данных по принципу "первый вошёл - первый вышел"

**# Лаба 4**

set\_directive\_pipeline [-II <num> / -rewind / -II <num> -rewind] <<cycle\_label>>

\* директива, уменьшающая количество периодов инициации функции или цикла путём изменения синтезируемой программы таким образом, чтобы некоторые операции, в том числе ввода/вывода, выполнялись параллельно.

\* -II <num> - initiation interval - количество периодов инициации, после которых начинается выполнение следующего цикла выполнения цикличного кода

\* -rewind - директива, позволяющая запускать исполнение цикла до окончания работы предыдущего экземпляра самого себя (допустим, если это цикл А в цикле Б, то А\_2 может быть запущен до окончания исполнения итерации цикла А\_1].

set\_directive\_dataflow

\* данная директива делает попытку параллельного выполнения следующего по коду цикла без ожидания конца предыдущего

\* это возможно лишь тогда, когда оба таких цикла независимы друг от друга, т.е. не конкурируют за разделяемые ресурсы

config\_dataflow -default\_channel <fifo/pingpong>

\* конфигурация директивы set\_directive\_dataflow

\* -default\_channel <fifo/pingpong> - конфигурация работы с оперативной памятью

\* pingpong - реализация буфера обмена такого, что в нём хранится некоторое множество данных, передаваемых в одно направление

\* fifo - реализация буфера обмена таким образом, что операции чтения и записи могут чередоваться

**Лаба 5**

set\_directive\_interface -mode <ap\_memory/...> -storage\_type <ram\_1p/ram\_2p> <<interface>>

\* ap\_memory - реализация портов ввода/вывода таким образом, чтобы аргументы взаимодействия с циклами влияли на работу интерфейса ввода/вывода

\* -storage\_type <ram\_1p/ram\_2p/...> - тип оперативной памяти - одно- или двух-канальная (есть и другие виды)

\* ram\_1p - одноканальная память

\* ram\_2p - двухканальная память

set\_directive\_array\_reshape -dim <dimension> -factor <factor> -type block <<interface>>

\* директива, выполняющая операцию преобразования входного потока данных таким образом, чтобы объединить последовательные множества данных при чтении/записи в один бОльший элемент данных. Таким образом количество одномоментно поступающих данных в синтезируемый код (и потенциально обрабатываемых) кратно увеличивается, а общее количество циклов передачи данных кратно уменьшается

set\_directive\_array\_partition -dim <dimension> -factor <factor> -type block <<interface>>

\* директива, выполняющая операцию разбиения множества входных/выходных данных на несколько равных по размеру (с некоторыми исключениями) меньших множеств данных. Такие изменения могут привести к увеличению используемого количества блоков памяти, но и увеличить пропускную способность памяти.

set\_directive\_unroll -factor <factor> <<cycle\_label>>

\* «развёртывание» циклов

\* -factor <factor> - множитель, указывающий, сколько раз должен быть "развёрнут" цикл

**Лаба 6**

[set\_directive\_array\_reshape/set\_directive\_array\_partition] -dim <dimension> -factor <factor> -type <block/cyclic/complete> <<interface>>

\* -dim <dimension> - измерение, в котором осуществляются действия над циклами

\* -factor <factor> - множитель, указывающий, во сколько раз должна быть изменена размерность входящего массива

\* -type <block/cyclic/complete> - представление в памяти массивов после их разбиения/соединения

\* block - разбивает массив на последовательные блоки - (factor:2) [1, 2, 3, 4, 5, 6, 7, 8] -> [1, 2, 3, 4] и [5, 6, 7, 8]

\* cyclic - разбивает массив так, чтобы элементы поочерёдно попадали в разные массивы - (factor:2) [1, 2, 3, 4, 5, 6, 7, 8] -> [1, 3, 5, 7] и [2, 4, 6, 8]

\* complete - по умолчанию - переформирует массивы массивов так, чтобы был образован один единый массив из множества элементов изначальных - [[1, 2, 3, 4], [5, 6, 7, 8]] -> [1, 2, 3, 4, 5, 6, 7, 8]

\* в случае с array\_reshape происходит не разбиение, а создание больших "слов", в остальном идея та же

**Лаба 7**

set\_directive\_bind\_op -op <mul/add/sub/...> -impl <dsp/fabric> -latency <latency> <<variable>>

\* директива, сопоставляющая выполняемые над выбранной переменной выбранные операции с выбранным вычислительным или логическим блоком.

\* -op <mul/add/sub/...> тип операции над переменной

\* -impl <dsp/fabric> - выбор блока, на котором будут выполняться операции

\* dsp – использование dsp ресурсов платы

\* fabric – не использовать dsp

\* -latency <latency> – желаемая задержка при использовании выбранного ресурса.