Микротрубочки

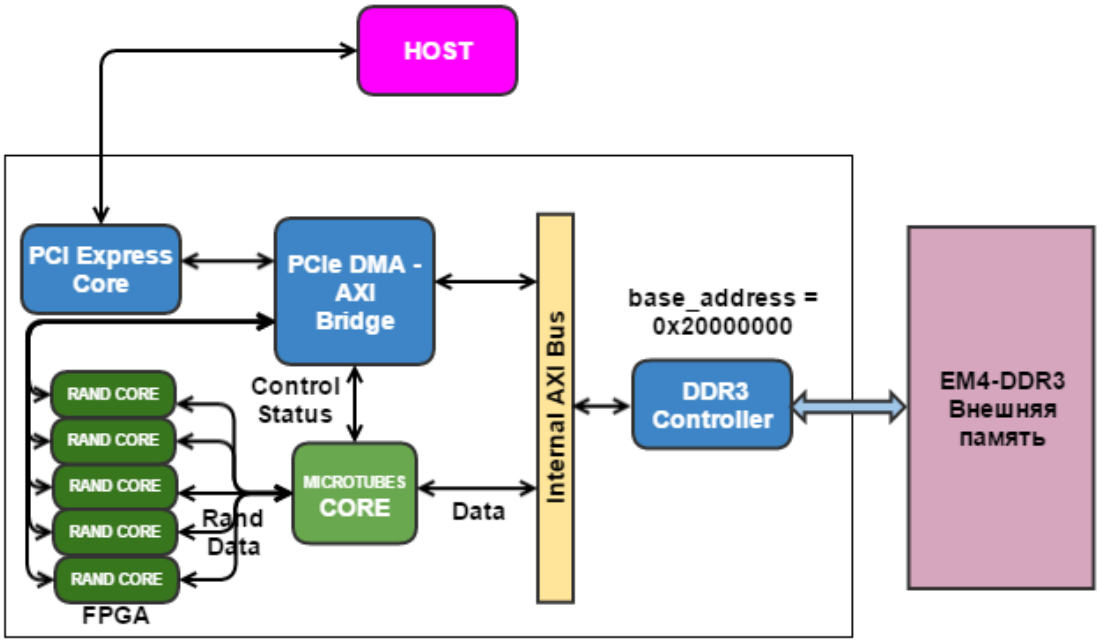
Исходные данные

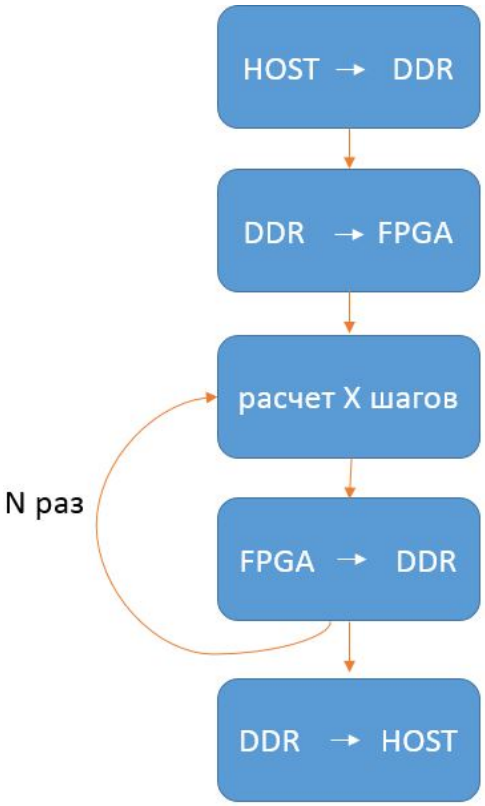
1. Есть два компьютера
    a. Сервер VM-XILINX (Windows 7). К нему подключаемся через RDP. Тут работаем в Vivado HLS и собираем битстрим в Vivado.
        i. Пароль для всех студентов 123456
    b. Сервер Kraft (Ubuntu). К нему подключен FPGA блок. Заходим на неге через Putty SSH через Windows машину VM-XILINX. Тут запускаем реальное прикладное приложение (хост-программу).
2. Исходные файлы
    a. На сервере Kraft - установленная Rosta SDK. Расположение - /opt/RostaSDK
        i. Готовая исходная хост-программа (общая) /opt/RostaSDK/lin/examples/mt
        ii. Утилиты (реконфигурация FPGA) /opt/RostaSDK/lin/utilities
    b. На VM-XILINX в вашей домашней директории две копии одного проекта rc_mt_2000_12 и готовый собранный битстрим

Порядок действий

1. Запустить готовое приложение (готовый битстрим и хост-программу)
    a. Скопировать готовый битстрим с VM-XILINX на сервер Kraft
    b. Запустить утилиту конфигурации с сконфигурировать FPGA скопированным на Kraft битстримом
    c. Запустить хост-программу rc47_mt_12
2. Самостоятельно собрать проект FPGA без изменения HLS ядра
    a. Проекты лежат в папке xilprojects (есть ярлык на рабочем столе)
3. Изучение и внесение изменений в приложение

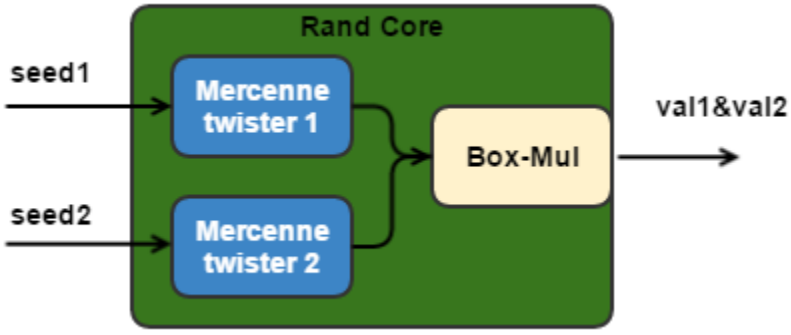
Обзор





Описание вычислительных ядер

Генератор случайных чисел RAND CORE.



Для наших целей необходим генератор случайных нормально распределенных, независимых случайных чисел, который мог бы выдавать результат каждый такт.

Этого удалось достичь с помощью

- преобразования Бокса-Мюллера wiki link (первый вариант)
- генератора равномерно распределенных случайных чисел "Вихрь Мерсенна" wiki link

С коды находятся в свободном доступе, они были некоторым образом модифицированы для того, чтобы случайные числа генерировались каждый такт.

При старте каждого генератора он инициализируется с помощью сидов, поступивших ему на вход. После некоторой начальной задержки он способен выдавать случайные числа каждый такт.

В проекте использовано пять одинаковых генераторов случайных чисел, каждый из которых выдает по два случайных нормально распределенных числа за такт.

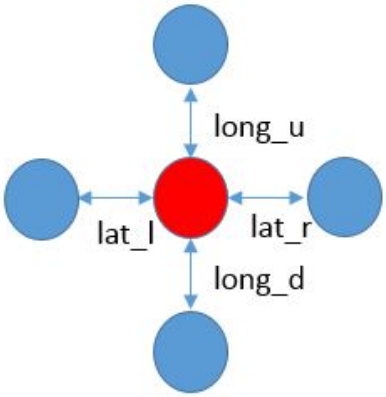
Вычислитель Microtubes CORE.

Задача Microtubes-core - вычислить координаты всех молекул через n шагов. Алгоритм работы ядра можно описать следующим образом;

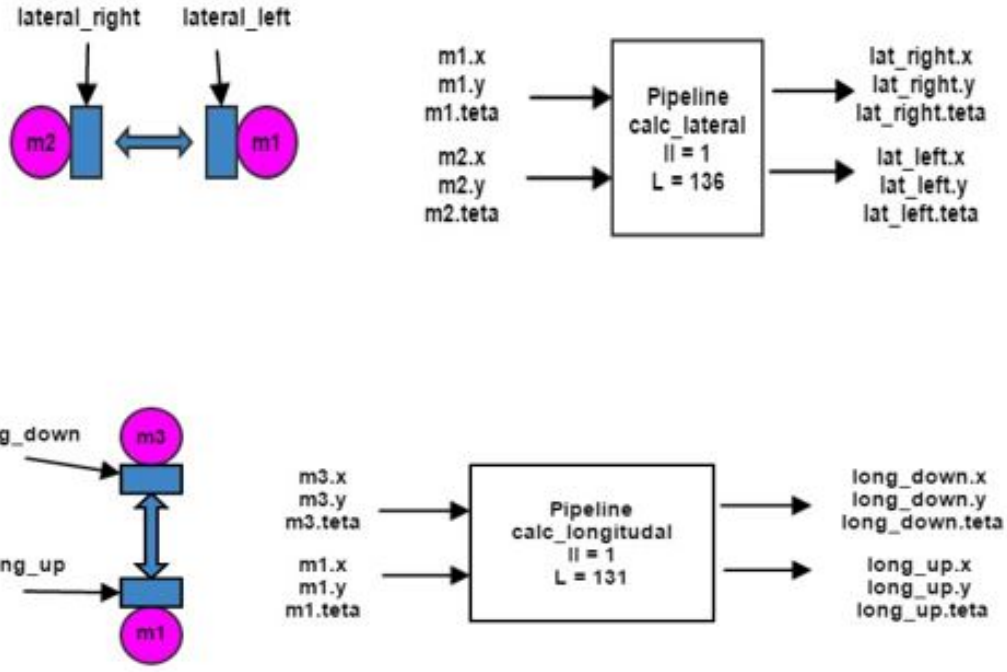
- чтение начальных координат из DDR памяти
- вычисления в течение n-шагов
- запись посчитанных координат в DDR память

Вычисления на одном шаге состоят из двух этапов:

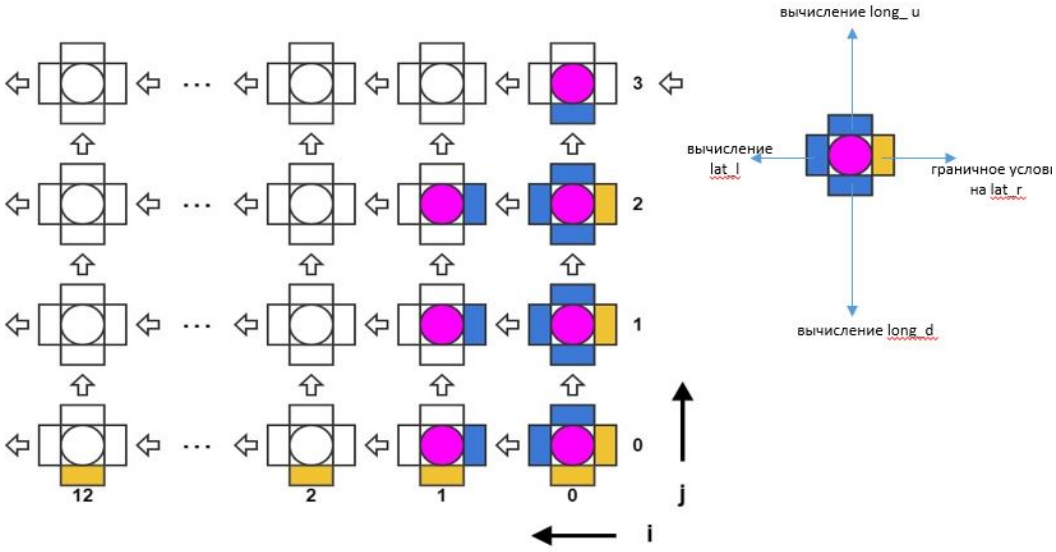
- вычисление сил взаимодействия как функций от координат молекул, посчитанных на предыдущем шаге
- вычисление новых координат молекул как функций вычисленных на данном шаге сил взаимодействия + случайной добавки



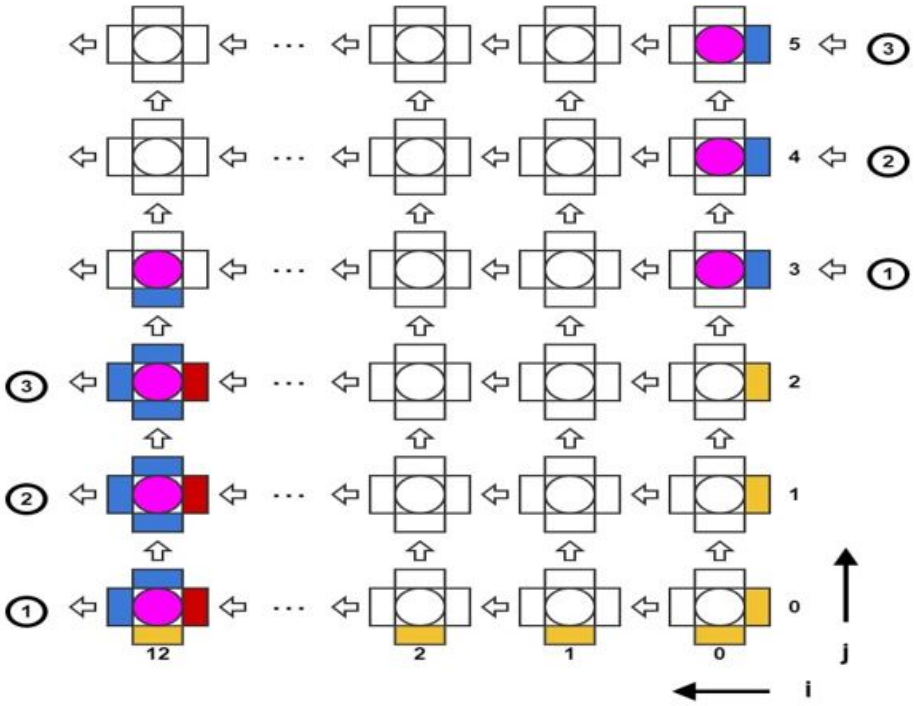
У каждой молекулы может быть до четырех молекул-соседей. Силы взаимодействия с соседями условно обозначены lat_l, lat_r, long_u, long_d. В случае отсутствия соседа соответствующая сила заменяется граничным условием.



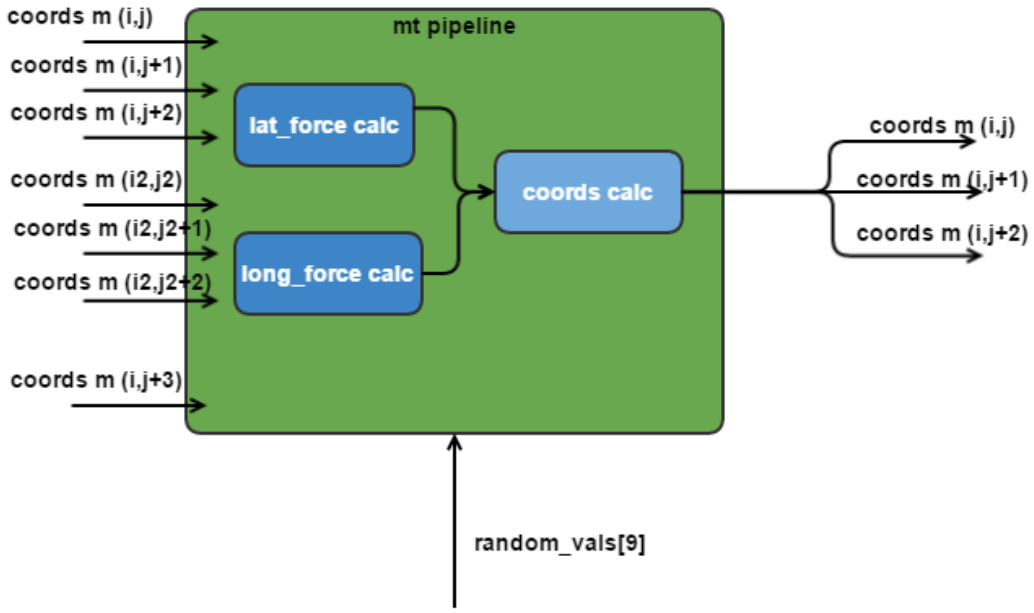
Все вычисления организованы в конвейерную схему. В конвейерной схеме можно выделить два больших вычислительных блока - функции, вычисляющие силы longitudinal and lateral взаимодействий.


вычисление long_ u
вычисление
lat_l
граничное услови
на lat_r
вычисление long_d
3
2
1
0
12
2
1
0
j
i

На одном шаге одновременно вычисляются силы и новые координаты для 3 текущих молекул. Стартовая молекула отмечена красным. Заметим, что lat_r(i+1,j)=lat l(i,j); long_u(i,j)=long_d(i,j+1). (i !=12) Таким образом, на данном шаге для данной молекулы необходимо посчитать только силы lat_l и long_u.



Важно обратить внимание на то, каким образом расположена трубочка. Из рисунка видно, что конечная точка спирали (отмечена оранжевым) граничит с молекулой, расположенной через три молекулы по вертикали. Таким образом lat_r(0,j+3)=lat_l(12,j). Ядро



Данный конвейер способен выдавать новый результат (координаты трех молекул на следующем шаге) каждый такт.

Интерфейс между Microtubes CORE и RAND CORE.

Так как Microtubes rand_core после этапа инициализации может гарантированно выставлять новые случайные значения на выходе каждый такт, то между ядрами реализован интерфейс ap_ack см. UG902 guide, стр. 470 и далее. Ядро Microtubes_core выставляет ap_ack как только прочитает значение из входного порта. Rand_core останавливается до того момента, как получит сигнал ap_ack, в случае получения сигнала генерирует новые выходные значения.

Доступ к DDR памяти со стороны HLS ядра

Аналогично лабораторной работе с HLS ядром.

Сборка HLS проектов, генерация IP ядер

1. Перейти в папку проекта mt_hls_2000_12
2. Перейти в папку hls
3. Запустить командный файл run.bat. Дождаться его завершения.
4. Открыть сгенерированный проект в Vivado HLS 2014.4
5. Открыть файл mt_fpga.cpp
6. Постараться разобраться в работе функций, опираясь на обзор

1. Перейти в папку проекта mt_hls_2000_12
2. Перейти в папку rand
3. Запустить командный файл run.bat. Дождаться его завершения.
4. Открыть сгенерированный проект в Vivado HLS 2014.4
5. Открыть файл rand_fpga.cpp
6. Постараться разобраться в работе функций, опираясь на обзор

Сборка FPGA проекта, генерация файла конфигурации ПЛИС

1. Перейти в папку проекта mt_hls_2000_12

2. Перейти в папку project
3. Запустить командный файл create_project.bat. Дождаться его завершения. Должно появиться приглашение командной строки vivado.
4. В командной строке Vivado набрать start_gui
5. Запустить генерацию файла конфигурации *.bit
6. После завершения генерации файла прошивки, скопировать его на Kraft

Запуск теста

Подготовка

1. Убедиться, что в системе обнаруживается хотя бы одна плата RC-47

```
$ cd /opt/RostaSDK/lin/utilities
$ ./rc47_find_devices
```

Пример вывода информации о найденной плате

```
/opt/RostaSDK/lin/utilities$  ./rc47_find_devices
Searching for RC47 boards
Found 1 board
Bus Address Chip ID
RC47 #0:::
System Chip 11:00.0 b007
Virtex-7 C0 0F:00.0 2000
Virtex-7 C1 0E:00.0 2000
Virtex-7 C2 12:00.0 2000
Virtex-7 C3 13:00.0 2000
PLX 8732 0C:00.0 8732
RC47 # 0 SN = 220003
```

Реконфигурация ПЛИС

1. Для того, чтобы реконфигурировать ПЛИС Virtex-7, с хост-компьютера, необходимо набрать в командной строке

```
$ cd /opt/RostaSDK/lin/utilities
$ ./rc47_config -b [board number] [chip_select] -f [path_to_file]
```

где board number - номер платы в системе, chip_select - любая комбинация C0, C1, C2, C3 или all, path_to_file - путь к файлу прошивки *.bit

Пример запуска утилиты:

```
/opt/RostaSDK/lin/utilities$   ./rc47_config -b 0 -v all -f $ROSTA_SDK_DIR/bit-files/mt_hls_2000_12.bit
Searching for RC47 boards
Found 1 board
                Bus Address     Chip ID
RC47 #0:::
      System Chip     0A:00.0       b007
      Virtex-7 C0     0C:00.0       2000
      Virtex-7 C1     0D:00.0       2000
      Virtex-7 C2     09:00.0       2000
      Virtex-7 C3     08:00.0       2000
Board selected: 0

-------------
Entered rc47_config_v7
Will now configure: C0 C1 C2 C3
C3 device diabled      08:00.0
C2 device diabled      09:00.0
C1 device diabled      0d:00.0
C0 device diabled      0c:00.0

load_bitstream: bit_file_path = $ROSTA_SDK_DIR/bit-files/mt_hls_2000_12.bit
file length = 4994915
Start configuration...
Config End OK
Wait for link
Rescan PCI Express bus after reconfig...
Searching for RC47 boards
                Bus Address     Chip ID
RC47 #0:::
      System Chip     0A:00.0       b007
      Virtex-7 C0     0C:00.0       2000
      Virtex-7 C1     0D:00.0       2000
      Virtex-7 C2     09:00.0       2000
      Virtex-7 C3     08:00.0       2000
setpci -s c:00.0 68.L=00003820
setpci -s d:00.0 68.L=00003820
setpci -s 9:00.0 68.L=00003820
setpci -s 8:00.0 68.L=00003820
```

Запуск HLS теста

1. Скопировать хост-программу /opt/RostaSDK/lin/examples/mt к себе в домашнюю директорию
2. Если тест не скомпилирован, скомпилировать его

```
$ cd $HOME_DIR/mt
$ make -B
```

3. Запустить тест, убедиться, что тест не выдает ошибок.

```
$ cd $HOME_DIR/mt
$   ./rc47_mt_12 -b [board] -v [chip_select] -N [N_d] -compare - rand
chip_select: C0 or C1 or C2 or C3
N_d: number of molecules in one filament
-compare to compare CPU and FPGA results
-rand to run CPU test with randomness
```

где board - номер платы в системе, chip_select - C0 или C1, N_d=36,

Пример запуска теста:

```
$   ./rc47_mt_12 -b 0 -v C0 -N 36 -compare -rand
================================================================================
=================
Searching for RC47 boards
Found 1 board
                Bus Address     Chip ID
RC47 #0:::
      System Chip     0C:00.0       b007
      Virtex-7 C0     0E:00.0       2000
      Virtex-7 C1     0F:00.0       2000
      Virtex-7 C2     0B:00.0       2000
      Virtex-7 C3     0A:00.0       2000
      PLX 8732        08:00.0       8732

Selected:
Board number    :::     0
Virtex Chip     :::     C0
Init DDR OK!
N_d is 36
N_d is 36
size is 1024
TOTAL_STEPS = 10000
STEPS_TO_WRITE = 1000
Flag rand is SET, 1

Flag rand is SET
hereerereerere
load_coords 1 flag_rand 1 flag_seed 1
hls done cnt = 2, reg_val = 0xd8
Step 0
      CPU Time = 0.608593
      FPGA Time = 0.002540
load_coords 1 flag_rand 1 flag_seed 0
hls done cnt = 2, reg_val = 0xd8
Step 1
      CPU Time = 0.618781
      FPGA Time = 0.002434
load_coords 1 flag_rand 1 flag_seed 0
hls done cnt = 2, reg_val = 0xd8
Step 2
      CPU Time = 0.582409
      FPGA Time = 0.002436
load_coords 1 flag_rand 1 flag_seed 0
```

hls done cnt = 2, reg_val = 0xd8
Step 3
    CPU Time = 0.582417
    FPGA Time = 0.002441
load_coords 1 flag_rand 1 flag_seed 0
hls done cnt = 2, reg_val = 0xd8
Step 4
    CPU Time = 0.582271
    FPGA Time = 0.002441
load_coords 1 flag_rand 1 flag_seed 0
hls done cnt = 2, reg_val = 0xd8
Step 5
    CPU Time = 0.582292
    FPGA Time = 0.002445
load_coords 1 flag_rand 1 flag_seed 0
hls done cnt = 2, reg_val = 0xd8
Step 6
    CPU Time = 0.582196
    FPGA Time = 0.002428
load_coords 1 flag_rand 1 flag_seed 0
hls done cnt = 2, reg_val = 0xd8
Step 7
    CPU Time = 0.582433
    FPGA Time = 0.002448
load_coords 1 flag_rand 1 flag_seed 0
hls done cnt = 2, reg_val = 0xd8
Step 8
    CPU Time = 0.582523
    FPGA Time = 0.002419
load_coords 1 flag_rand 1 flag_seed 0
hls done cnt = 2, reg_val = 0xd8
Step 9
    CPU Time = 0.581987
    FPGA Time = 0.002467

```
Test OK!
```

Внесение изменений в HLS проект

В данном пункте необходимо добавить новую функциональность в hls ядро. Необходимо реализовать выбор длины микротрубочки N_d в зависимости от входного параметра. В качестве входного параметра использовать p2.

p2	N_d
0	12
1	24
2	36
3	48
4	60
5	72
6	84
7	108
8	156
9	216

Генерация IP ядра, добавление нового IP ядра в проект.

1. Для генерации IP ядра необходимо на верхней панели выбрать пункт Export RTL
2. В появившемся окне выбираем
   Format Selection - IP catalog
   Нажать на кнопку Configutation.. , указать vendor - urock, version - 0.
   В качестве языка экспортируемого ядра выбираем Verilog
3. Жмем OK.
4. В FPGA проекте в меню выбрать:
   Project Settings -> IP.
   Нажать кнопку Refresh Repository.
   Нажать OK
5. В tcl консоли набираем
   report_ip_status -name ip_status
6. В появившейся вкладке нажать кнопку upgrade Selected
7. Запустить генерацию файла конфигурации *.bit
8. После завершения генерации файла прошивки, скопировать его на Kraft

Генерация файла займет довольно много времени (~5! часов), поэтому можно, не дожидаясь завершения, переходить к пункту "Внесение изменений в хост-программу".

Внесение изменений в хост-программу.

Открыть файл rc47_mt_12.cpp найти в файле определение функции mt_fpga

Алгоритм работы хост-программы заключается в следующем:

- Запись seeds для всех генераторов случайных чисел в регистры
- Инициирование начала работы rand hls ядра.
- В течении N итераций
    - Запуск хост-функции (программная имплементация алгоритма)
    - Запуск вычислителя
        - Запись данных по определенному адресу в DDR память
        - Конфигурирование HLS ядра (задание параметров STEPS_TO_WRITE, rand и.т.д.)
        - Инициирование начала работы HLS ядра
        - Ожидание завершения работы HLS ядра
        - Чтение данных по определенному адресу из DDR памяти
    - Проверка на ошибки

Для того, чтобы версия хост-программы соответствовала новому HLS ядру , необходимо внести соответствующие изменения:

- Добавить чтение данных по еще одному адресу HLS_B

После внесения изменений необходимо скомпилировать тест

Запуск теста с новым HLS ядром

Выполнить подпункты "Реконфигурация ПЛИС" и "Запуск HLS теста" из пункта "Запуск теста"

Убедиться в отсутствии ошибок.

Результатом данной работы являются

rc47_mt_12.cpp

mt_fpga.cpp

Необходимо показать работу хост-программы преподавателю.