Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт

по лабораторной работе №5

Название: Разработка ускорителей вычислений средствами САПР высокоуровневого синтеза Xilinx Vitis HLS

Дисциплина: Архитектура ЭВМ

Студент ИУ7-54Б _____ Л.Е.Тартыков

(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ А.Ю.Попов

(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Цель работы

Изучение методики и технологии синтеза аппаратных устройств ускорения вычислений по описаниям на языках высокого уровня C/C++.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- рассмотреть маршрут проектирования устройств, представленных в виде синтаксических конструкций ЯВУ C/C++;
- изучить принципы работы IDE Xilinx Vitis HLS;
- изучить методику анализа и отладки устройств;
- разработать ускоритель вычислений по индивидуальному заданию;
- разработать код для тестирования ускорителя;
- реализовать ускоритель с помощью средств высоко-уровненного синтеза, выполнить его отладку.

1 Основные теоретические сведения

1.1 Методология ускорения вычислений на основе ПЛИС

Ускорение вычислительных алгоритмов с использованием программируемых логических интегральных схем (ПЛИС) имеет ряд преимуществ по сравнению с их реализацией на универсальных микропроцессорах, или графических процессорах. В то время, как традиционная разработка программного обеспечения связана с программированием на заранее определенном наборе машинных команд, разработка программируемых устройств - это создание специализированной вычислительной структуры для реализации желаемой функциональности.

На рисунке 1.1 представлены принципы организации вычислений на различных платформах.

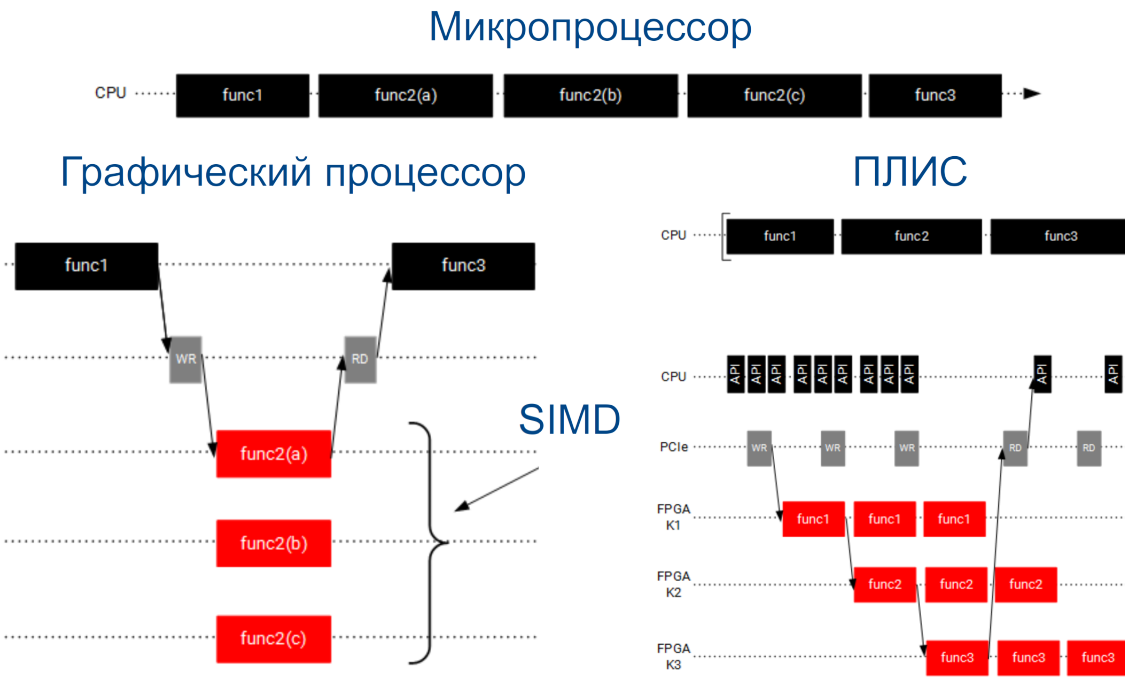

Рисунок 1.1 – Принципы организации вычислений на различных платформах

Методологию создания ускорителей на ПЛИС с применением средств синтеза высокого уровня (High Level Synthesis, HLS) можно представить в виде трех этапов:

1. Создание архитектуры приложения.

2. Разработка ядра аппаратного ускорителя на языках C/C++.

3. Анализ производительности и выявление способов ее повышения.

2 Вариант индивидуального задания

Вариант индивидуального задания - 16.

На листинге 2.1 представлен программный код неоптимизированного цикла.

Листинг 2.1 – Программный код неоптимизированного цикла

```
extern "C" {
    void var016(int* c, const int* a, const int* b, const int
       len) {
        int minB = a[len-1];
        for (int i = len-1; i >=0 ; i--) {
            if (minB > b[i]) minB = b[i];
        }
        for (int i = 0; i < len; i++) {
            if (a[i] < minB) {
                c[i] = minB;
            } else {
                c[i] = a[i];
            }
        }
    }
}
```

На листинге 2.2 представлен программный код конвейерной организации цикла.

Листинг 2.2 – Программный код конвейерной организации цикла

```
extern "C" {
    void var016_pipelined(int* c, const int* a, const int* b,
       const int len) {
        int minB = a[len-1];
        #pragma HLS PIPELINE
        for (int i = len-1; i >=0 ; i--) {
            if (minB > b[i]) minB = b[i];
        }
        for (int i = 0; i < len; i++) {
            if (a[i] < minB) {
                c[i] = minB;
            } else {
                c[i] = a[i];
            }
```

```
14          }
15      }
16  }
```

На листинге 2.3 представлен программный код частично развернутого цикла.

Листинг 2.3 – Программный код частично развернутного цикла

```
1  extern "C" {
2      void var016_unrolled(int* c, const int* a, const int* b,
           const int len) {
3
4          int minB = a[len-1];
5          for (int i = len-1; i >=0 ; i--) {
6              #pragma HLS UNROLL
7              if (minB > b[i]) minB = b[i];
8          }
9          for (int i = 0; i < len; i++) {
10             if (a[i] < minB) {
11                 c[i] = minB;
12             } else {
13                 c[i] = a[i];
14             }
15         }
16     }
17 }
```

На листинге 2.4 представлен программный код конвейерного и частично развернутого цикла.

Листинг 2.4 – Программный код конвейерного и частично развернутого цикла

```
1  extern "C" {
2      void var016_pipe_unroll(int* c, const int* a, const int* b,
           const int len) {
3          int minB = a[len-1];
4          #pragma HLS PIPELINE
5          for (int i = len-1; i >=0 ; i--) {
6              #pragma HLS UNROLL factor=2
7              if (minB > b[i]) minB = b[i];
8          }
9          for (int i = 0; i < len; i++) {
10             #pragma HLS UNROLL factor=2
11             if (a[i] < minB) {
12                 c[i] = minB;
13             } else {
14                 c[i] = a[i];
15             }
16         }
17     }
18 }
```

3 Результаты работы приложения в режиме Emulation-SW

На рисунке 3.1 представлен результат работы приложения в режиме Emulation-SW.

```
Found Platform
Platform Name: Xilinx
INFO: Reading /iu_home/iu7136/workspace/hls_acc_lab_system/Emulation-SW/binary_container_1.xclbin
Loading: '/iu_home/iu7136/workspace/hls_acc_lab_system/Emulation-SW/binary_container_1.xclbin'
Trying to program device[0]: xilinx_u200_xdma_201830_2
Device[0]: program successful!
|-------------------------+-------------------------|
| Kernel                  |    Wall-Clock Time (ns) |
|-------------------------+-------------------------|
| var016_no_pragmas       |                 1560122 |
|-------------------------+-------------------------|
| var016_unrolled         |                  655607 |
|-------------------------+-------------------------|
| var016_pipelined        |                  587078 |
|-------------------------+-------------------------|
| var016_pipe_unroll      |                 1121007 |
|-------------------------+-------------------------|
Note: Wall Clock Time is meaningful for real hardware execution only, not for emulation.
Please refer to profile summary for kernel execution time for hardware emulation.
TEST PASSED.
```

Рисунок 3.1 – Результаты работы приложения в режиме Emulation-SW

4 Результаты работы приложения в режиме Emulation-HW

На рисунке 4.1 представлен результат работы приложения в режиме Emulation-HW.

Рисунок 4.1 – Результаты работы приложения в режиме Emulation-HW

5 Результаты работы приложения в режиме Hardware

На рисунке 5.1 копия экрана вкладки "Summary".

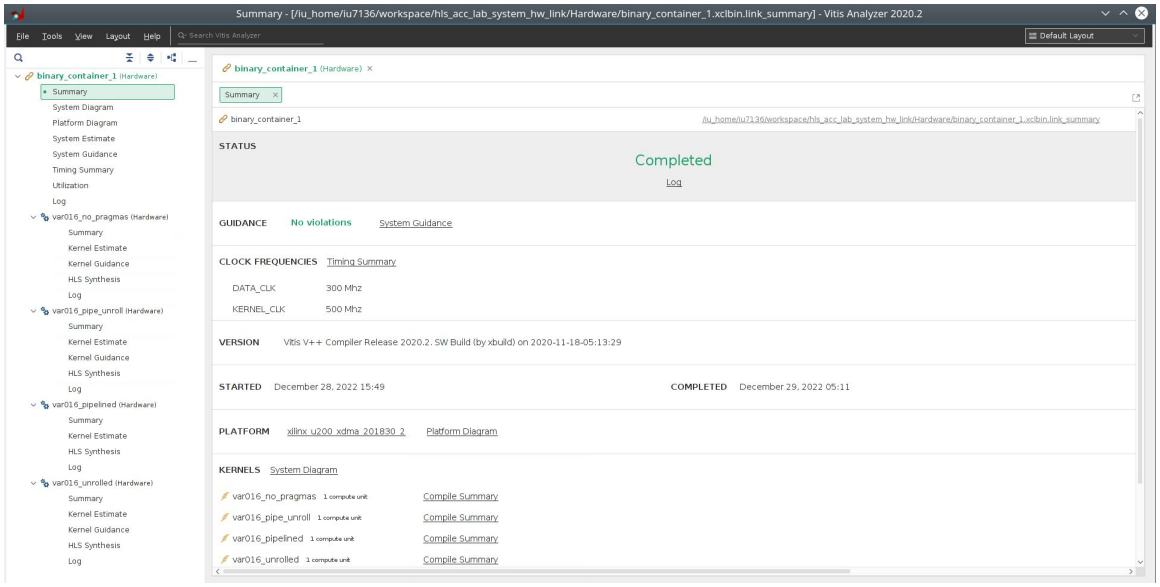

Рисунок 5.1 – Копия экрана вкладки "Summary"

На рисунке 5.2 копия экрана вкладки "System Diagram".

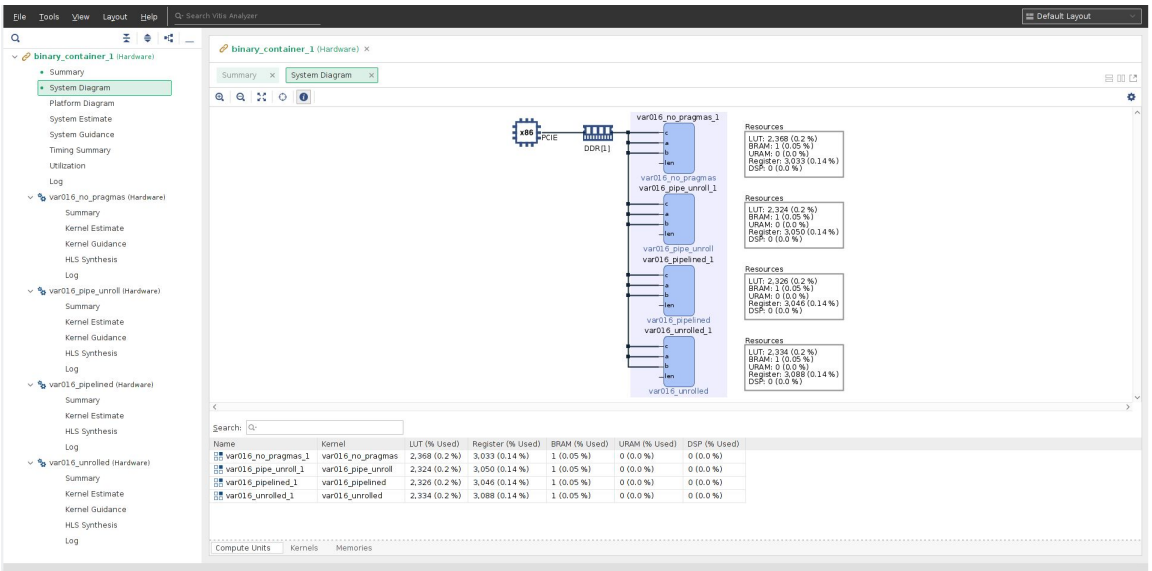

Рисунок 5.2 – Копия экрана вкладки "System Diagram"

На рисунке 5.3 копия экрана вкладки "Platform Diagram".

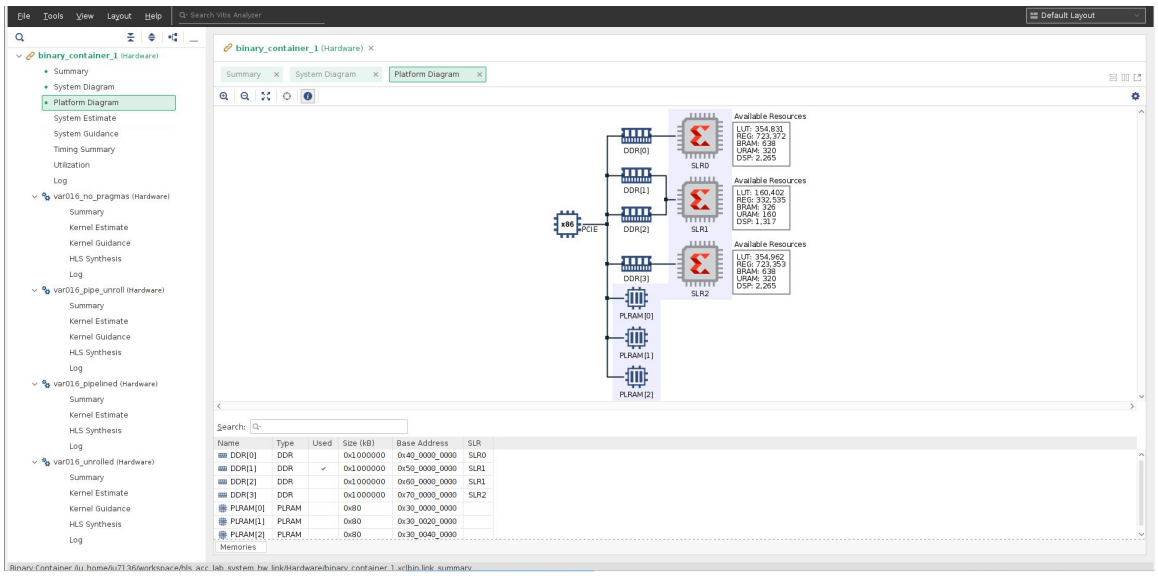

Рисунок 5.3 – Копия экрана вкладки "Platform Diagram"

На рисунке 5.4 копия экрана вкладки "HLS Synthesis"для ядра "no_pragmas".

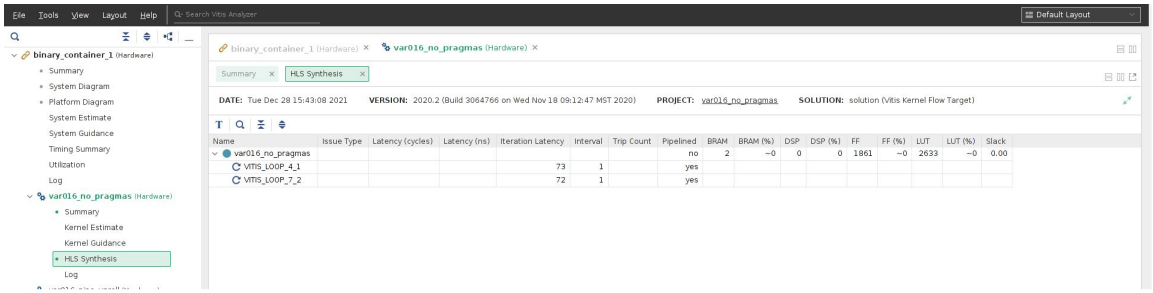

Рисунок 5.4 – Копия экрана вкладки "HLS Synthesis"для ядра "no_pragmas"

На рисунке 5.5 копия экрана вкладки "HLS Synthesis"для ядра "pipe_unroll".

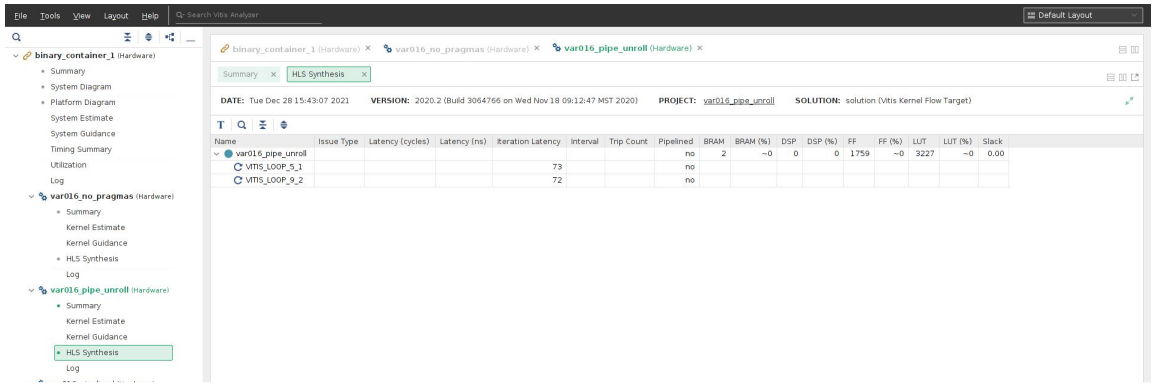

Рисунок 5.5 – Копия экрана вкладки "HLS Synthesis"для ядра "pipe_unroll"

На рисунке 5.6 копия экрана вкладки "HLS Synthesis"для ядра "pipelined".

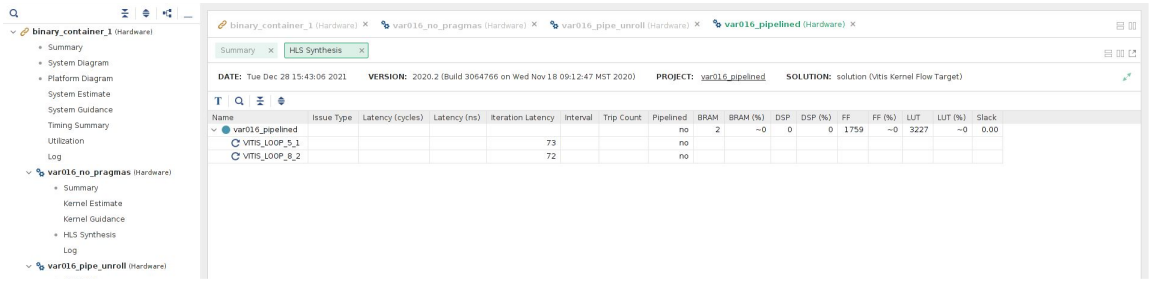

Рисунок 5.6 – Копия экрана вкладки "HLS Synthesis"для ядра "pipelined"

На рисунке 5.7 копия экрана вкладки "HLS Synthesis"для ядра "unrolled".

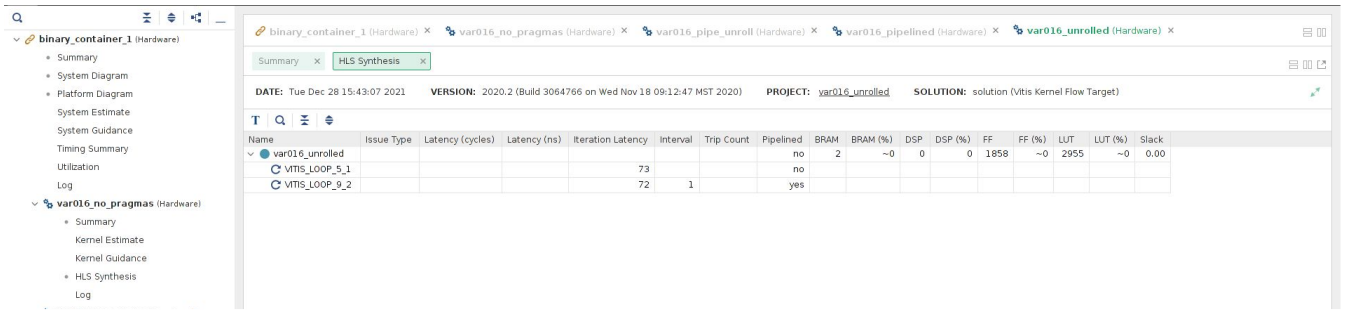

Рисунок 5.7 – Копия экрана вкладки "HLS Synthesis"для ядра "unrolled"

На рисунке 5.8 представлен результат работы приложения в режиме Hardware.

```
Found Platform
Platform Name: Xilinx
INFO: Reading /iu_home/iu7136/workspace/hls_acc_lab_system/Hardware/binary_container_1.xclbin
Loading: '/iu_home/iu7136/workspace/hls_acc_lab_system/Hardware/binary_container_1.xclbin'
Trying to program device[0]: xilinx_u200_xdma_201830_2
Device[0]: program successful!
|-------------------------+-------------------------|
| Kernel                  |    Wall-Clock Time (ns) |
|-------------------------+-------------------------|
| var016_no_pragmas       |               128210801 |
|-------------------------+-------------------------|
| var016_unrolled         |               363226413 |
|-------------------------+-------------------------|
| var016_pipelined        |               528863849 |
|-------------------------+-------------------------|
| var016_pipe_unroll      |               528824484 |
|-------------------------+-------------------------|
Note: Wall Clock Time is meaningful for real hardware execution only, not for emulation.
Please refer to profile summary for kernel execution time for hardware emulation.
TEST PASSED.
```

Рисунок 5.8 – Результаты работы приложения в режиме Hardware

Контрольные вопросы

1. Назовите преимущества и недостатки аппаратных ускорителей на ПЛИС по сравнению с CPU и графическими ускорителями?

Достоинствами данной системы являются:

- низкая стоимость в сравнению с аппаратными ускорителями;
- большая частота эмуляции;
- компактность.

2. Назовите основные способы оптимизации циклических конструкций ЯВУ, реализуемых в виде аппаратных ускорителей?

Основными недостатками аппаратных эмуляторов на ПЛИС являются:

- необходимость перекомпиляции проекта и переконфигурации ПЛИС при любом исправлении содержимого проекта;
- наличие специализированного программного обеспечения для разделения модели микросхемы на части для загрузки в отдельные ПЛИС.

3. Назовите этапы работы программной части ускорителя в хост системе?

Способы оптимизаций циклов:

- конвейерная обработка циклов;
- разворачивание циклов;
- потоковая обработка;

4. В чем заключается процесс отладки для вариантов сборки Emulation-SW, Emulation-HW и Hardware?

1. Этап 1: Инициализируется среда OpenCL.
2. Этап 2. Приложение создает три буфера, необходимых для обмена данными с ядром: два буфера для передачи исходных данных и один для вывода результата (память должна быть выделена с выравниванием 4 КБ).
3. Этап 3. Запуск задачи на исполнение.
4. Этап 4: После завершения работы всех команд выходной буфер R_buf содержит результаты работы ядра.

5. Какие инструменты и средства анализа результатов синтеза возможно использовать в Vitis HLS для оптимизации ускорителей?

- Программная эмуляция (Emulation-SW) - код ядра компилируется для работы на CPU хост-системы. Этот вариант сборки служит для верификации совместного исполнения кода хост-системы и кода ядра, для выявления синтаксических ошибок, выполнения отладки на уровне исходного кода ядра, проверки поведения системы.
- Аппаратная эмуляция (Emulation-HW) - код ядра компилируется в аппаратную модель (RTL), которая запускается в специальном симуляторе на CPU. Этот вариант сборки и запуска занимает больше времени, но обеспечивает подробное и точное представление активности ядра. Данный вариант сборки полезен для тестирования функциональности ускорителя и получения начальных оценок производительности.
- Аппаратное обеспечение (Hardware) - код ядра компилируется в аппаратную модель (RTL), а затем реализуется на FPGA. В результате формируется двоичный файл xclbin, который будет работать на реальной FPGA.

Компилятор Xilinx Vitis v++ является одним из наиболее удачных проектов в этой области, и позволяет генерировать из описаний на языке C/C++ синтезируемые низкоуровневые RTL проекты, которые затем отображаются на структуру ПЛИС.

Заключение

Изучены архитектуры гетерогенных вычислительных систем и технологии разработки ускорителей вычислений на базе ПЛИС фирмы Xilinx.

Были выполнены следующие задачи:

- изучены основные сведения о платформе Xilinx Alveo U200;
- разработано RTL (Register Transfer Language, язык регистровых передач) описание ускорителя вычислений по индивидуальному варианту;
- выполнена генерация ядра ускорителя;
- выполнены синтез и сборка бинарного модуля ускорителя;
- разработано и отлажено тестирующее программное обеспечение на серверной хост-платформе;
- проведены тесты работы ускорителя вычислений.

Поставленная цель достигнута.