**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»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5**

**По курсу: "Архитектура ЭВМ"**

Студент Чыонг Нгуен Вьет Уи Группа ИУ7-52Б Название предприятия МГТУ им. Н. Э. Баумана, каф. ИУ7 Тема Методология разработки и верификации ускорителей вычислений на платформе

Xilinx Alveo

|  |  |  |
| --- | --- | --- |
| Студент: |  | Чыонг Н.В.У. |
| Преподаватель: | подпись, дата | Фамилия, И.О.  Попов A.Ю. |
|  | подпись, дата | Фамилия, И. О. |

Москва — 2022 г.

|  |  |  |
| --- | --- | --- |
| [**Цель работы**](#_bookmark1) |  | **2** |
| [**1 Основные теоретические сведения**](#_bookmark2) |  | **3** |
| [**Основные теоретические сведения**](#_bookmark2) |  | **3** |
| [1.1 Методология ускорения вычислений на основе ПЛИС](#_bookmark3) | . . . . . . . . . . . | 3 |

[Файлы функций ядра на основе индивидуального задания](#_bookmark5) 5

[Результаты рабты приложения в режиме Emulation-SW](#_bookmark10) 9

[Копия экрана Assistant View для сборки Emulation-HW](#_bookmark11) 10

[Результаты работы приложения в режиме Emulation-HW](#_bookmark12) 11

[Окно внутрисхемного отладчика Vivado для сборки в режиме Emulation-](#_bookmark14) [HW](#_bookmark14) 12

[Результаты работы приложения в режиме Hardware](#_bookmark16) 13

[Копии вкладок для сборки Hardware](#_bookmark17) 14

[Контрольные вопросы](#_bookmark23) 19

[Заключение](#_bookmark0) 20

Изучение методики и технологии синтеза аппаратных устройств ускорения вычис- лений по описаниям на языках высокого уровня.

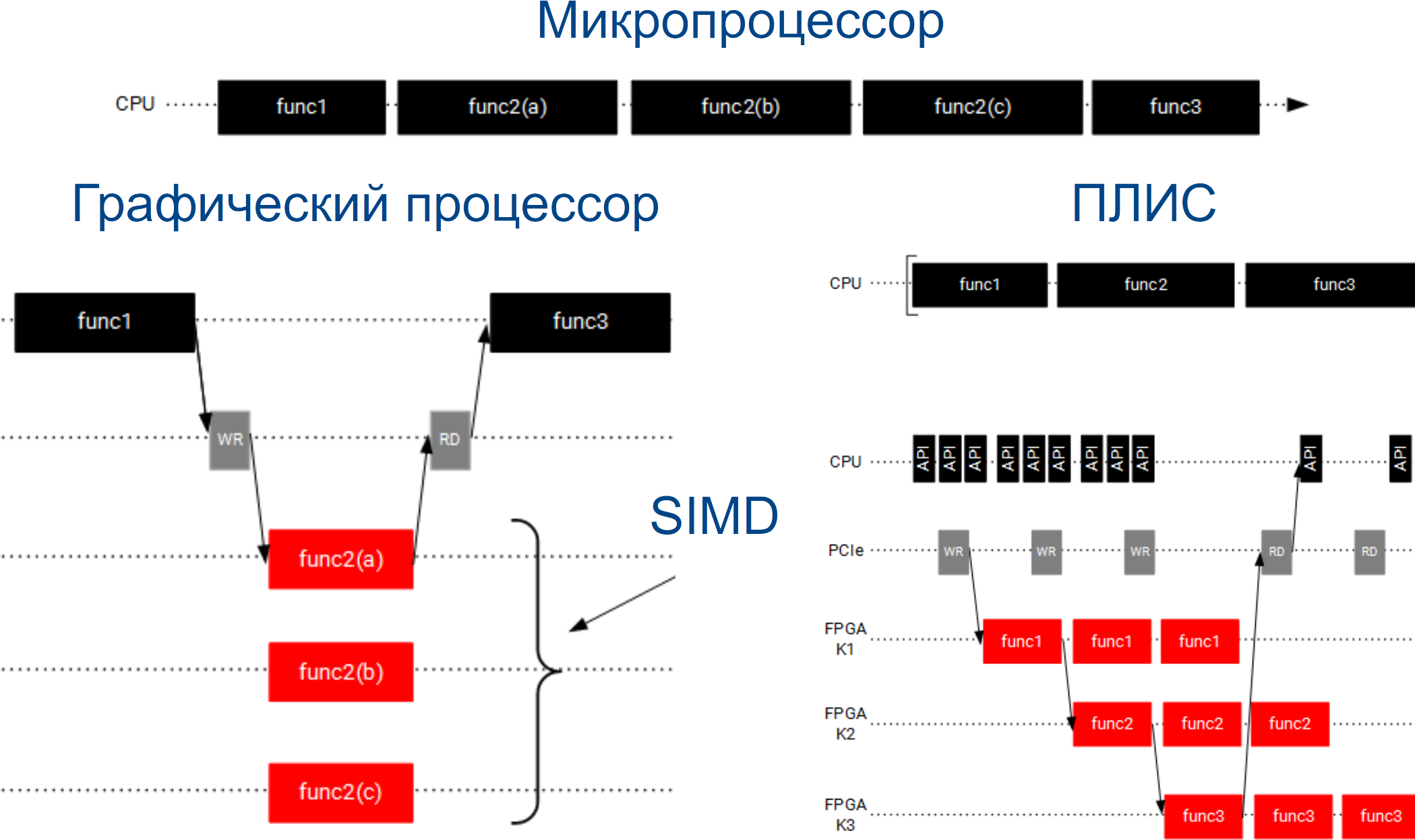
Для достижения поставленной цели необходимо выполнить следующие задачи:

* рассматреть маршрут проектирования устройств, представленных в виде синтак- сических 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, выпол- нить его отладку.

# Основные теоретические сведения

* 1. **Методология ускорения вычислений на основе ПЛИС**

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

На рисунке [1.1](#_bookmark4) 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 вы- сокого 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.

# Файлы функций ядра на основе инди- видуального задания

В листинге [1.1](#_bookmark6) представлен не оптимизированный цикл на основе индивидуального задания.

В исходном коде были ошибки, которые я исправила для того, чтобы код стал ра- бочим. Были объявлены массивы tmp\_add и tmp\_mul, хотя далее используются tmpA, tmpB.Также во втором вложенном цикле было неверное присваивание и сравнение.

Листинг 1.1 – Индивидуальный вариант 20

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | **extern** "C" { |  | | | |
| 2 | #d e f i n e N 64 |
| 3 | **void** var 020 ( **int** ∗ c , **const int** ∗ a , **const int** ∗ | b , | **const** | **int** | l e n ) |
|  | { |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 4 |  | **int** | i t e r a t i o n s = l e n / N; |
| 5 |  | **int** | tmpA[N ] ; |
| 6 |  | **int** | tmpB [N ] ; |
| 7 |  | **for** | ( **int** i = 0 ; i < i t e r a t i o n s ; i++) { |
| 8 |  |  | **for** ( **int** j = 0 ; j < N; j++) { |
| 9 |  |  | tmpA[ j ] = a [ i ∗N+j ] + b [ i ∗N+j ] ; |
| 10 |  |  | tmpB [ j ] = a [ i ∗N+j ] ∗ b [ i ∗N+j ] ; |
| 11 |  |  | } |
| 12 |  |  | **for** ( **int** j = 0 ; j < N; j++) { |
| 13 |  |  | **i f** (tmpA[ j ]>tmpB[ j ] ) { |
| 14 |  |  | c [ i ∗N+j ] = tmpA[ j ] ; |
| 15 |  |  | } |
| 16 |  |  | **else** { |
| 17 |  |  | c [ i ∗N+j ] = tmpB [ j ] ; |
| 18 |  |  | } |
| 19 |  |  | } |
| 20 |  | } |  |
| 21 | } |  |  |
| 22 } |  |  |  |

В листинге [1.2](#_bookmark7) представлен конвейерная организация цикла на основе индивиду- ального задания.

Листинг 1.2 – Индивидуальный вариант 20

**extern** "C" {

1

2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | #d e f i n e N 64 | | |  | | | |
| 4 | **void** var 020 \_pipelined ( **int** ∗ c , **const**  **const int** l e n ) { | | | **int** ∗ a , | **const** | **int** ∗ | b , |
| 5 |  | #pra | gma HLS PIPELINE | | | | |
| 6 |  | **int** | i t e r a t i o n s = l e n / N; | | | | |
| 7 |  | **int** | tmpA[N ] ; | | | | |
| 8 |  | **int** | tmpB [N ] ; | | | | |
| 9 |  | **for** | ( **int** i = 0 ; i < i t e r a t i o n s ; i++) { | | | | |
| 10 |  |  | **for** ( **int** j = 0 ; j < N; j++) { | | | | |
| 11 |  |  | tmpA[ j ] = a [ i ∗N+j ] + b [ i ∗N+j ] ; | | | | |
| 12 |  |  | tmpB [ j ] = a [ i ∗N+j ] ∗ b [ i ∗N+j ] ; | | | | |
| 13 |  |  | } | | | | |
| 14 |  |  | **for** ( **int** j = 0 ; j < N; j++) { | | | | |
| 15 |  |  | **i f** (tmpA[ j ]>tmpB[ j ] ) { | | | | |
| 16 |  |  | c [ i ∗N+j ] = tmpA[ j ] ; | | | | |
| 17 |  |  | } | | | | |
| 18 |  |  | **else** { | | | | |
| 19 |  |  | c [ i ∗N+j ] = tmpB [ j ] ; | | | | |
| 20 |  |  | } | | | | |
| 21 |  |  | } | | | | |
| 22 |  | } |  | | | | |
| 23  24 } | } |  |  | | | | |

В листинге [1.3](#_bookmark8) представлен частично развернутый цикл на основе индивидуального задания.

Листинг 1.3 – Индивидуальный вариант 20

**const int** ∗ b , **const**

|  |  |  |
| --- | --- | --- |
| 1  2  3 | **extern** "C" {  #d e f i n e N 64  **void** var 020 \_urolled ( **int** ∗ c , **const int** ∗ a ,  **int** l e n ) { | |
| 4 | #pra | gma HLS UNROLL |
| 5 | **int** | i t e r a t i o n s = l e n / N; |
| 6 | **int** | tmpA[N ] ; |
| 7 | **int** | tmpB [N ] ; |
| 8 | **for** | ( **int** i = 0 ; i < i t e r a t i o n s ; i++) { |
| 9 |  | **for** ( **int** j = 0 ; j < N; j++) { |
| 10 |  | tmpA[ j ] = a [ i ∗N+j ] + b [ i ∗N+j ] ; |
| 11 |  | tmpB [ j ] = a [ i ∗N+j ] ∗ b [ i ∗N+j ] ; |
| 12 |  | } |
| 13 |  | **for** ( **int** j = 0 ; j < N; j++) { |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 14 |  |  |  | **i f** | (tmpA[ j ]>tmpB[ j ] ) { |
| 15 |  |  |  |  | c [ i ∗N+j ] = tmpA[ j ] ; |
| 16 |  |  |  | } | **else** { |
| 17 |  |  |  |  | c [ i ∗N+j ] = tmpB [ j ] ; |
| 18 |  |  |  | } |  |
| 19 |  |  | } |  |  |
| 20 |  | } |  |  |  |
| 21 | } |  |  |  |  |
| 22 } |  |  |  |  |  |

В листинге [1.4](#_bookmark9) представлен конвейерный и частично развернутый цикл на основе индивидуального задания.

Листинг 1.4 – Индивидуальный вариант 20

**extern** "C" {

#d e f i n e N 64

**void** var 020 \_pipe\_unroll ( **int** ∗ c , **const int** ∗ a ,

**const int** l e n ) { #pragma HLS DATAFLOW

**int** i t e r a t i o n s = l e n / N;

**int** tmpA[N ] ;

**int** tmpB [N ] ;

**for** ( **int** i = 0 ; i < i t e r a t i o n s ; i++) {

**for** ( **int** j = 0 ; j < N; j++) {

tmpA[ j ] = a [ i ∗N+j ] + b [ i ∗N+j ] ; tmpB [ j ] = a [ i ∗N+j ] ∗ b [ i ∗N+j ] ;

}

**for** ( **int** j = 0 ; j < N; j++) {

**i f** (tmpA[ j ]>tmpB[ j ] ) { c [ i ∗N+j ] = tmpA[ j ] ;

} **else** {

c [ i ∗N+j ] = tmpB [ j ] ;

}

}

}

}

}

**const**

**int** ∗ b ,

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

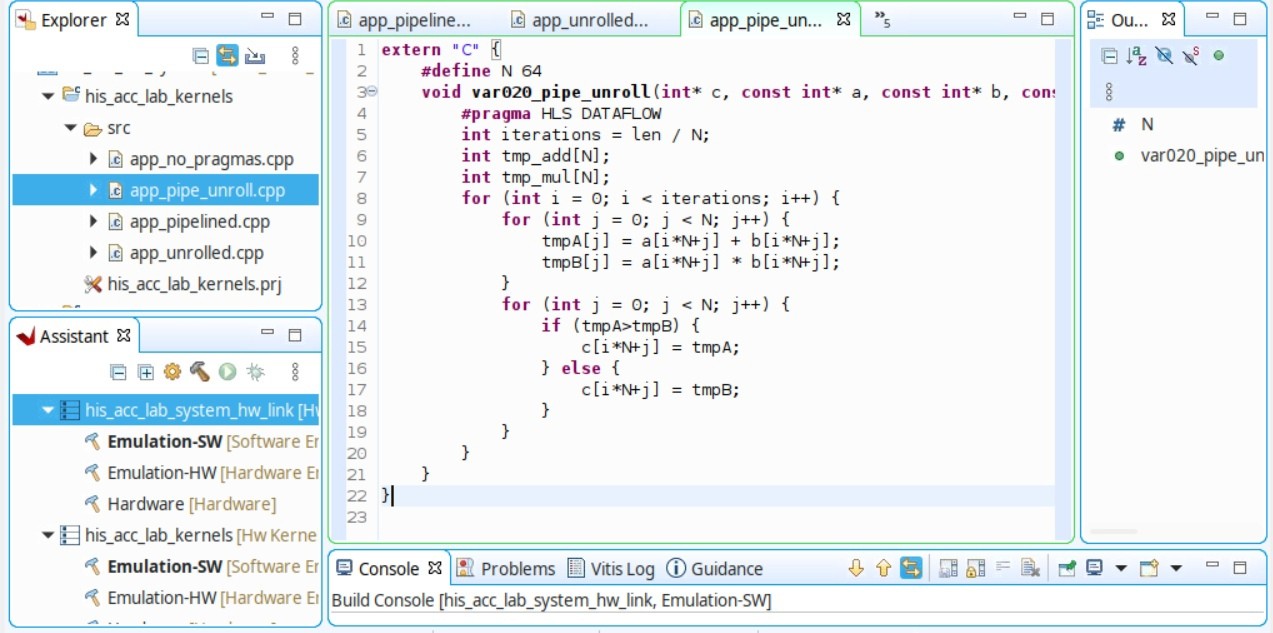
18

19

20

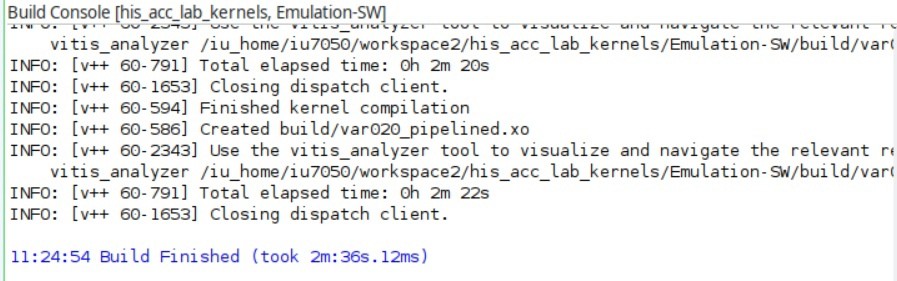
21

22



## Рисунок 1.2 – Файлы в системе

Сборка hls\_acc\_lab\_kernel и hls\_acc\_lab\_system\_hw\_link.



## Рисунок 1.3 – Результат сборки hls\_acc\_lab\_kernel.

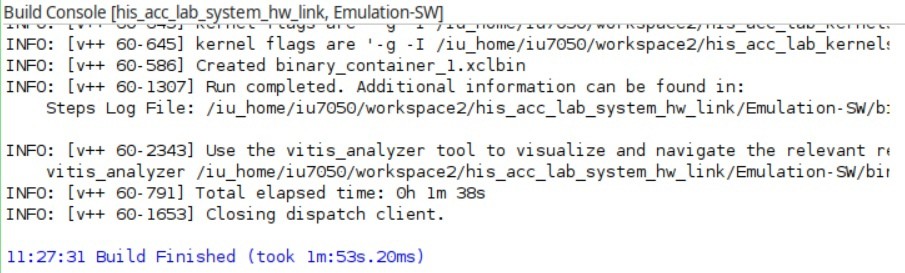
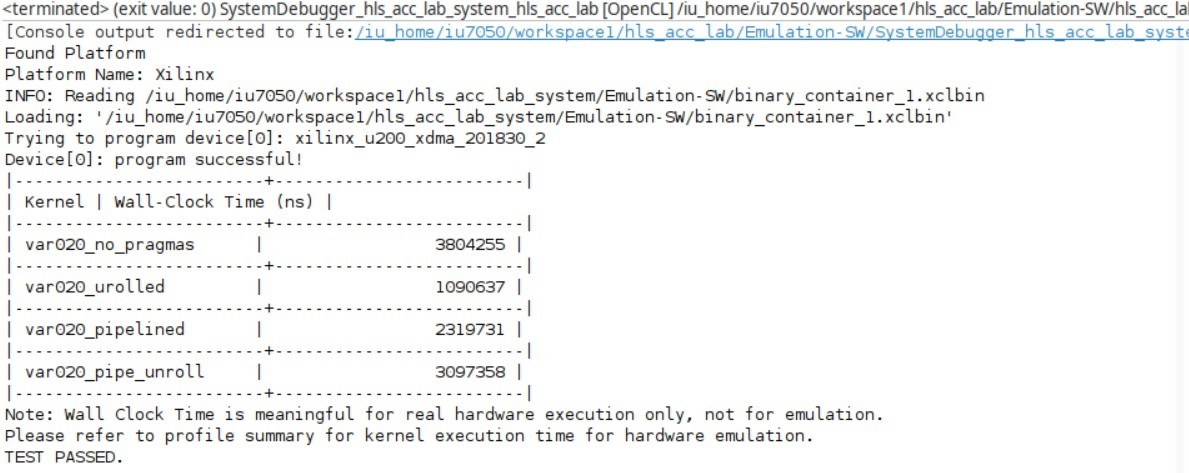
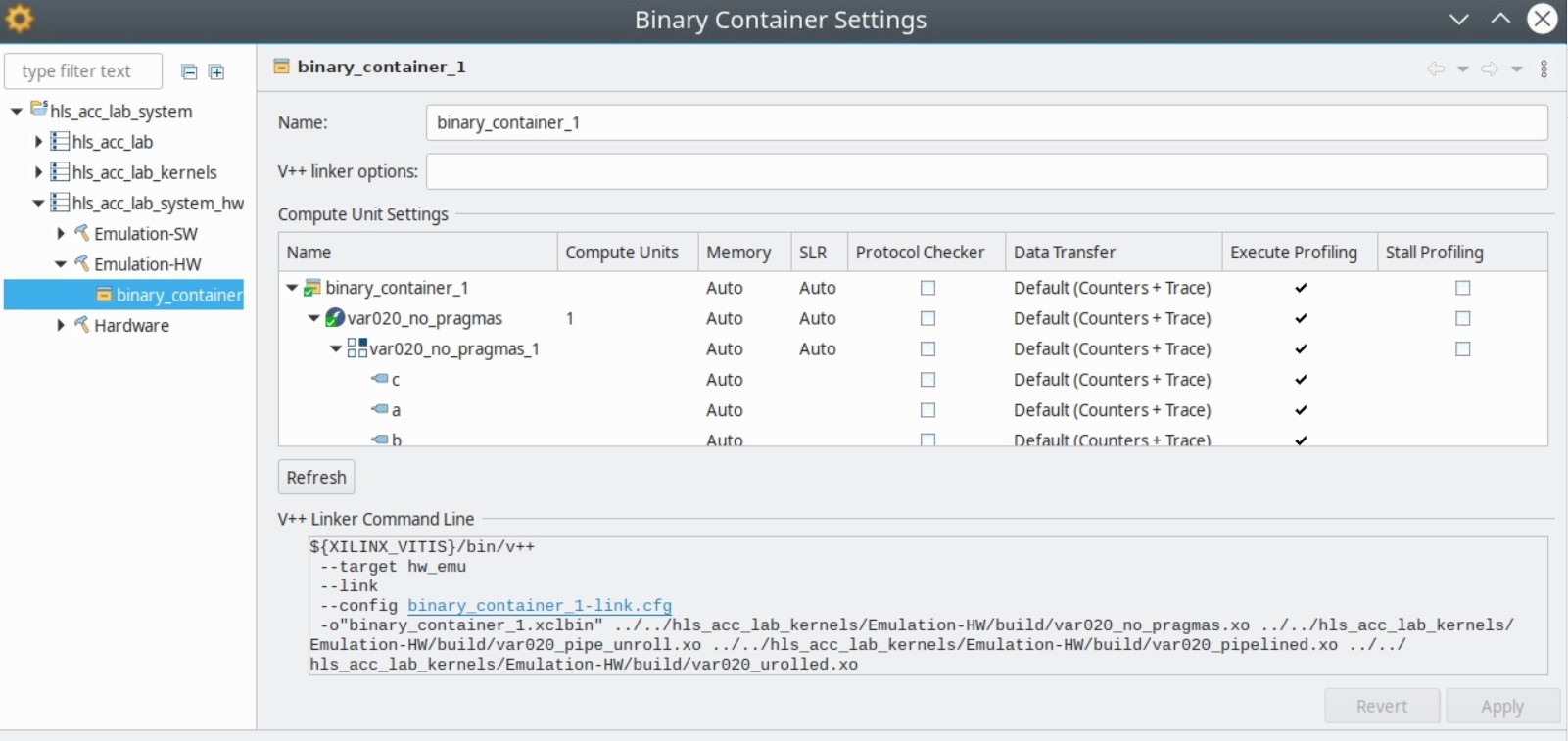


Рисунок 1.4 – Результат сборки hls\_acc\_lab\_system\_hw\_link.

На рисунке 1.5 представлены результаты работы приложения в режиме Emulation- SW.

## Рисунок 1.5 – Результаты рабты приложения в режиме Emulation-SW

На рисунках 1.6- 1.7 представлена копия экрана Assistant View для сборки Emulation- HW.

## Рисунок 1.6 – Копия экрана Assistant View для сборки Emulation-HW

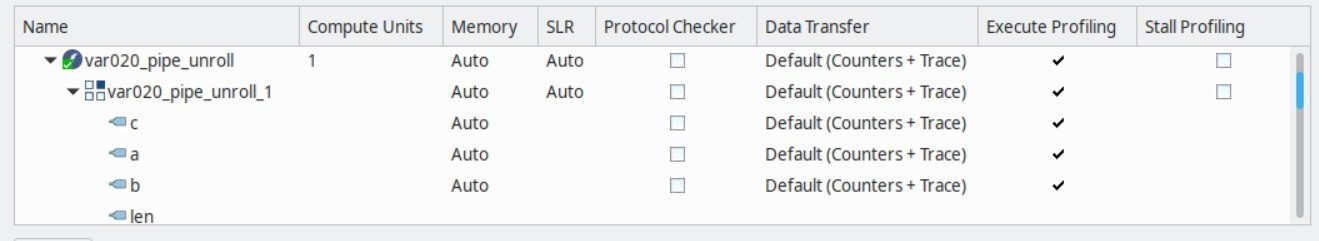


Рисунок 1.7 – Копия экрана Assistant View для сборки Emulation-HW

# жиме Emulation-HW

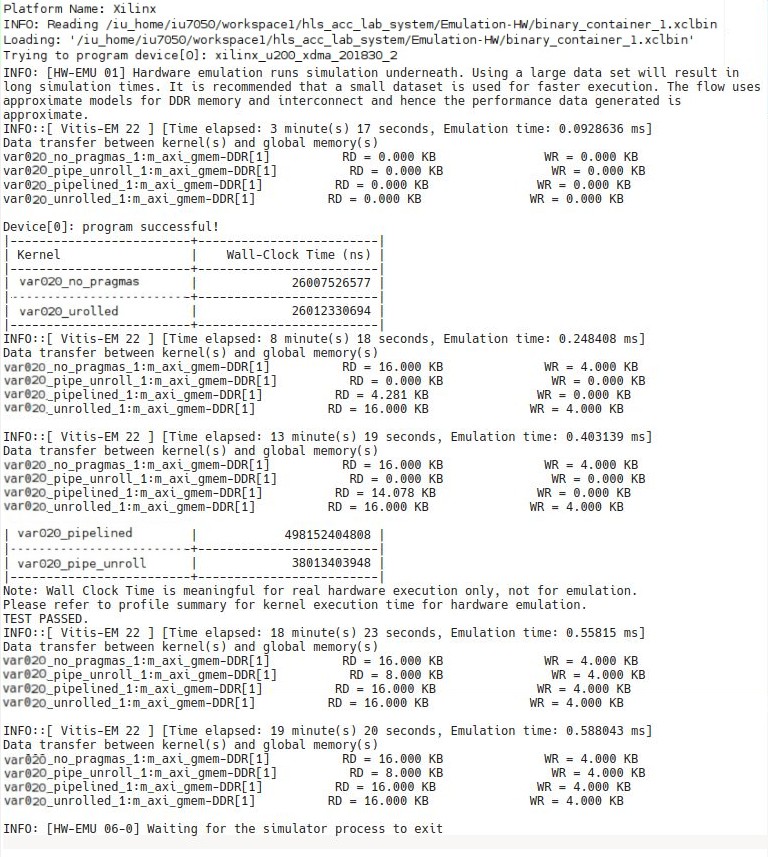
На рисунке [1.8](#_bookmark13) представлены результаты работы приложения в режиме Emulation- HW.

Рисунок 1.8 – Результаты рабты приложения в режиме Emulation-HW

# для сборки в режиме Emulation-HW

На рисунке [1.9](#_bookmark15) представлено окно внутрисхемного отладчика Vivado для сборки в режиме Emulation-HW.

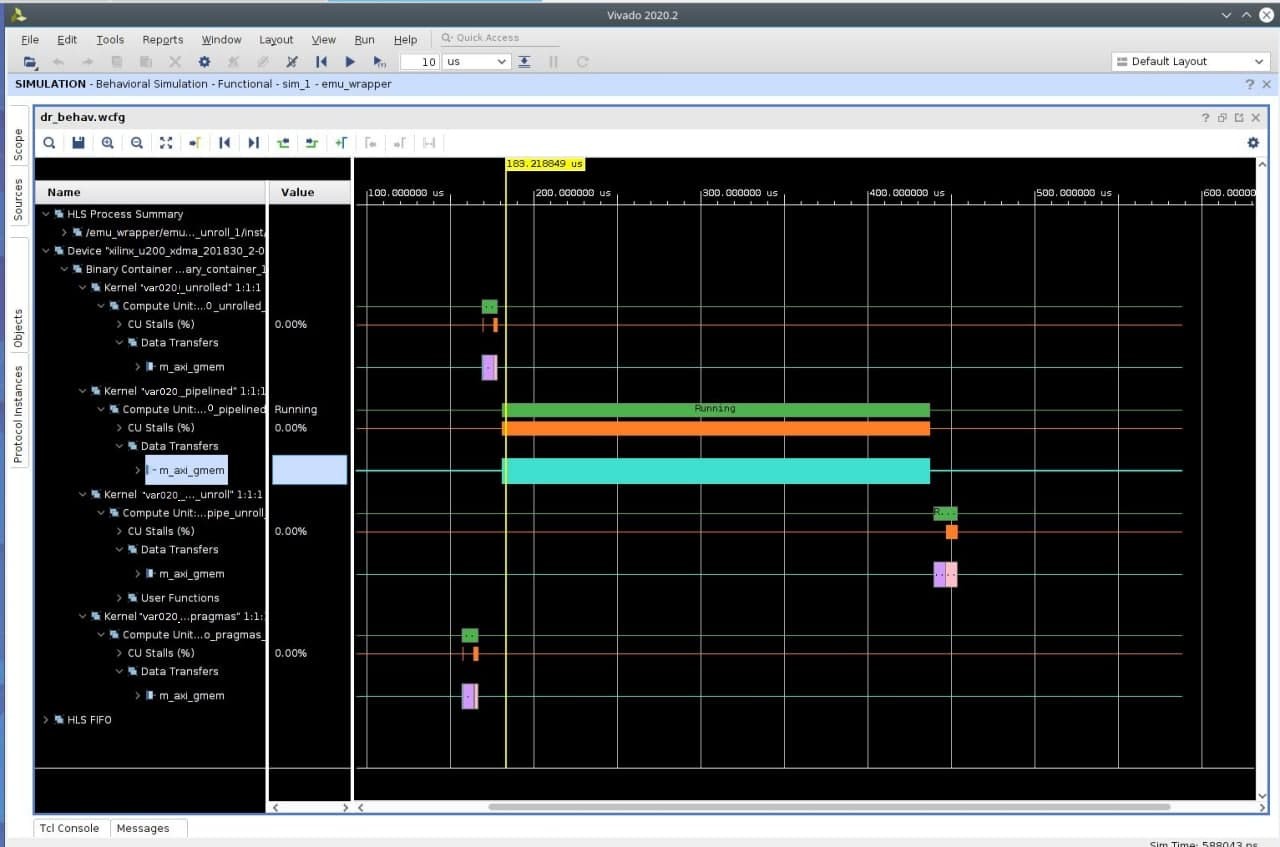
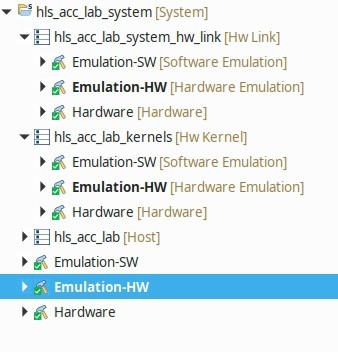


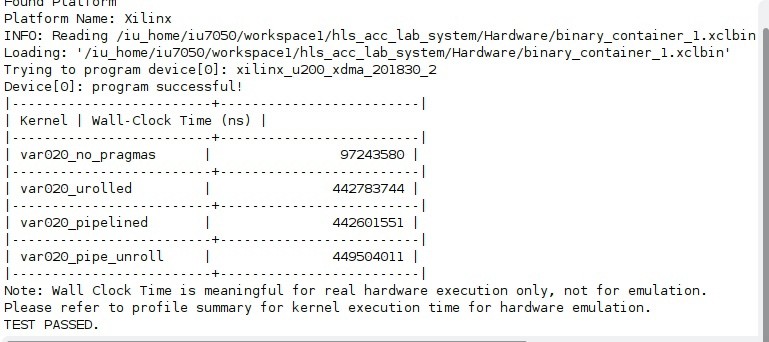
Рисунок 1.9 – Окно внутрисхемного отладчика Vivado для сборки в режиме Emulation-HW

# жиме Hardware



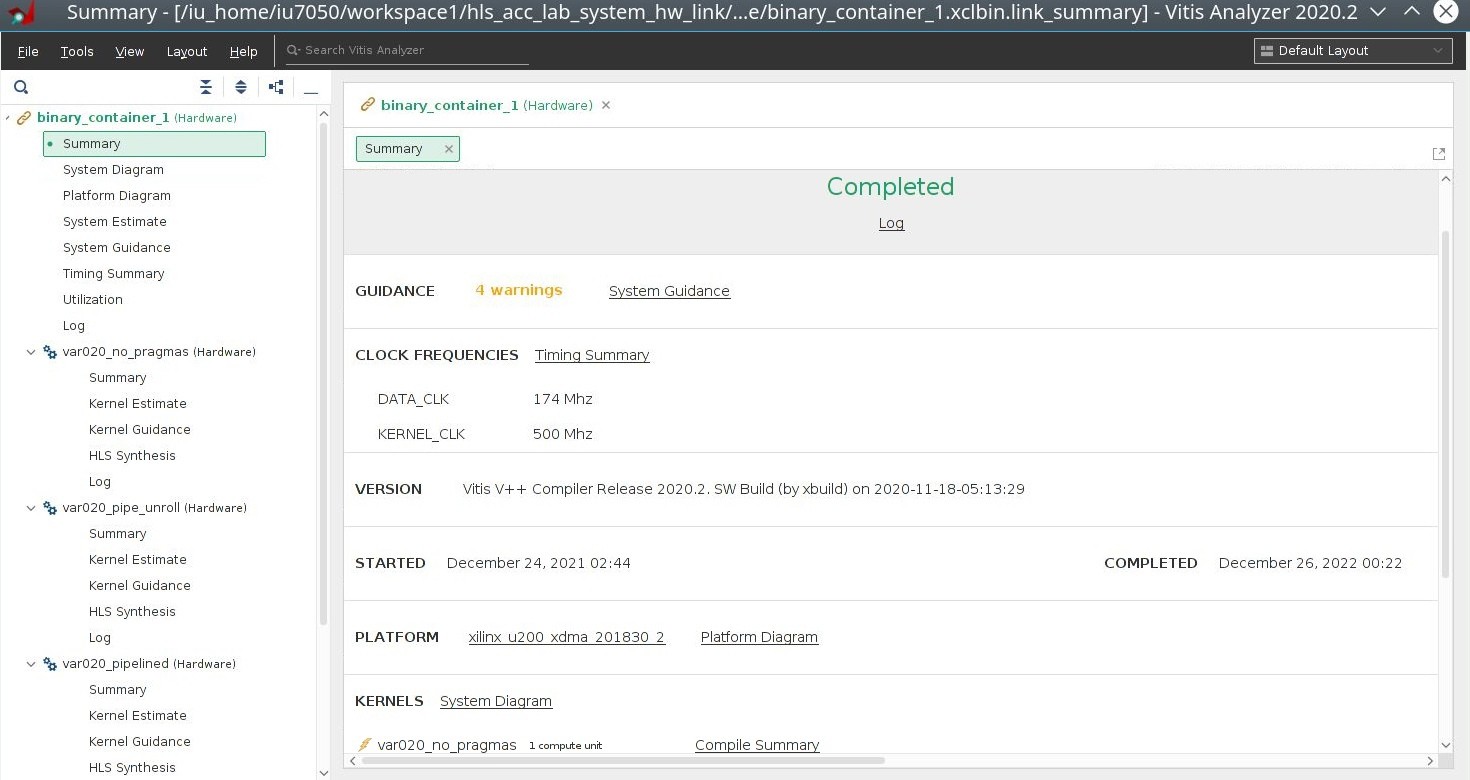
## Рисунок 1.10 – Результаты сборки Hardware

На рисунке 1.11 представлены результаты работы приложения в режиме Hardware.



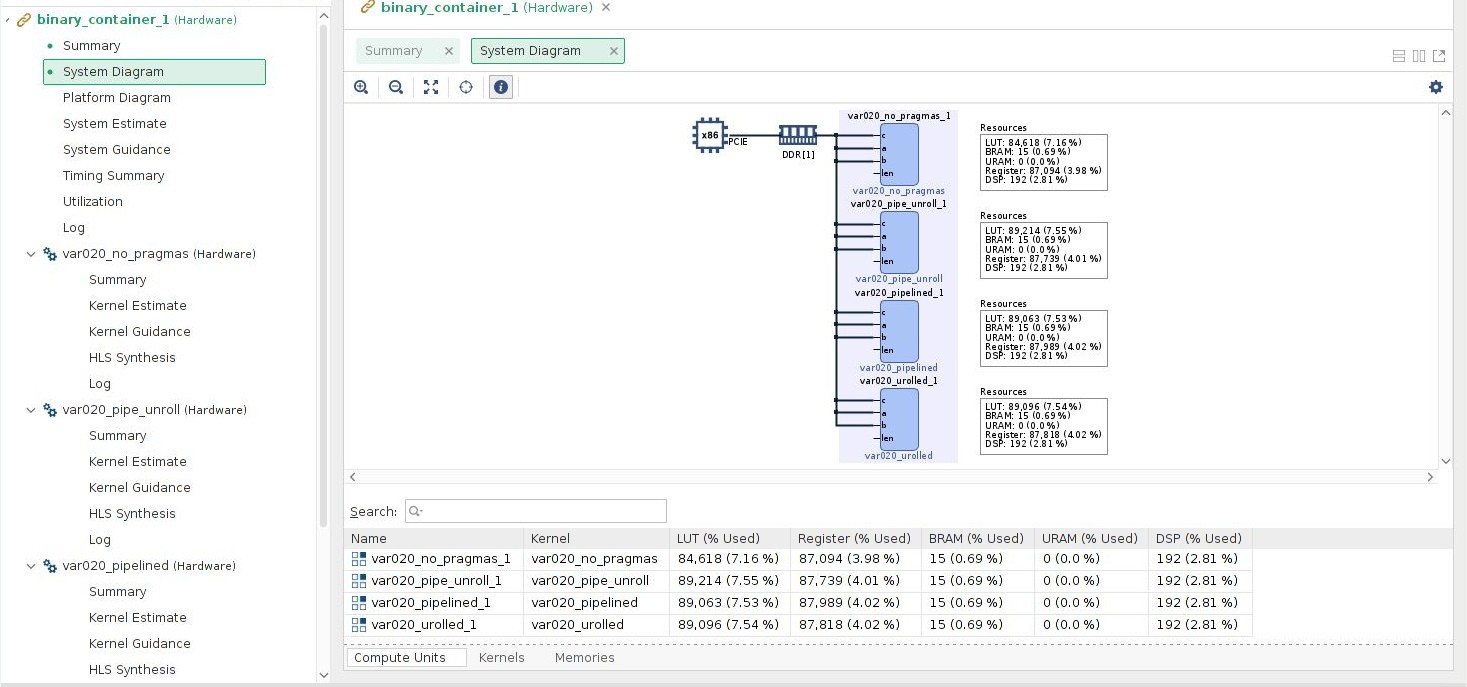
## Рисунок 1.11 – Результаты рабты приложения в режиме Hardware

На рисунке 1.12 представлены результаты работы приложения в режиме Hardware.



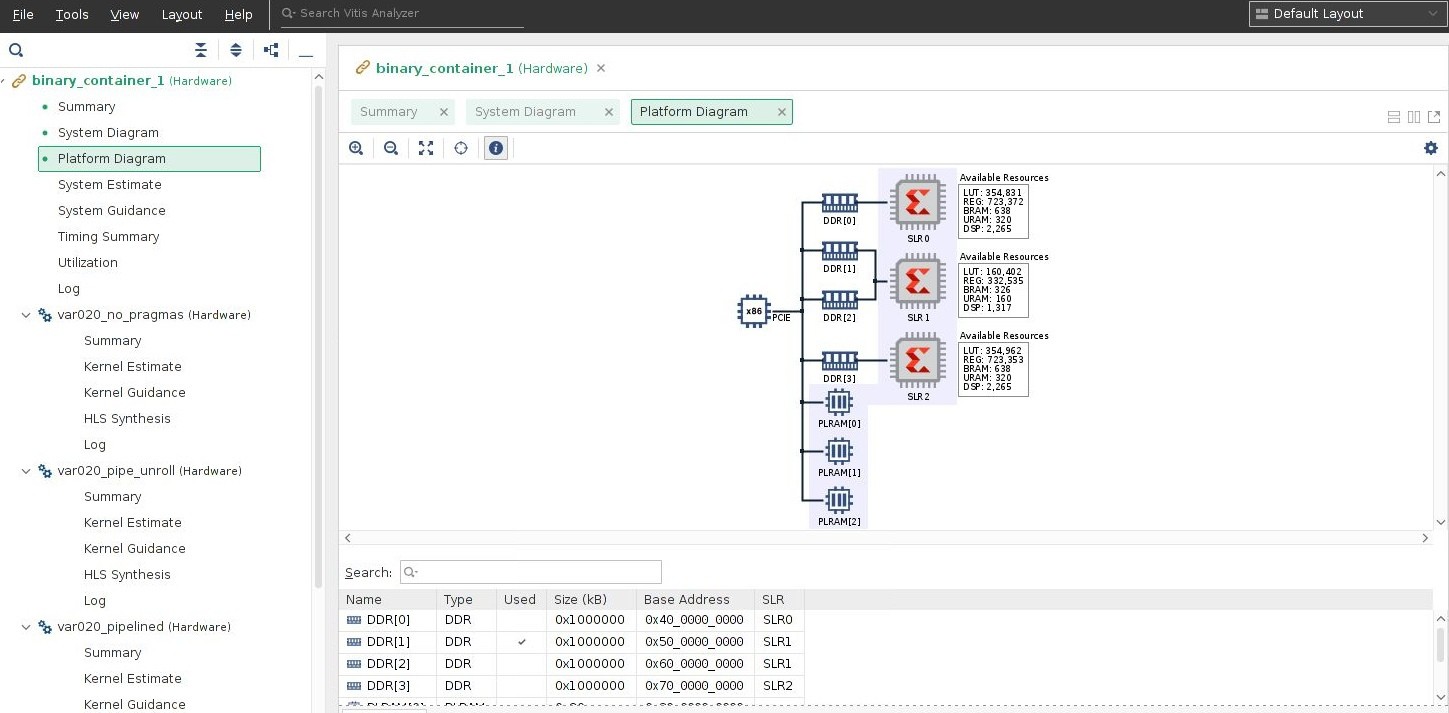
## Рисунок 1.12 – Результаты рабты приложения в режиме Hardware

На рисунке 1.13 представлены результаты работы приложения в режиме Hardware.



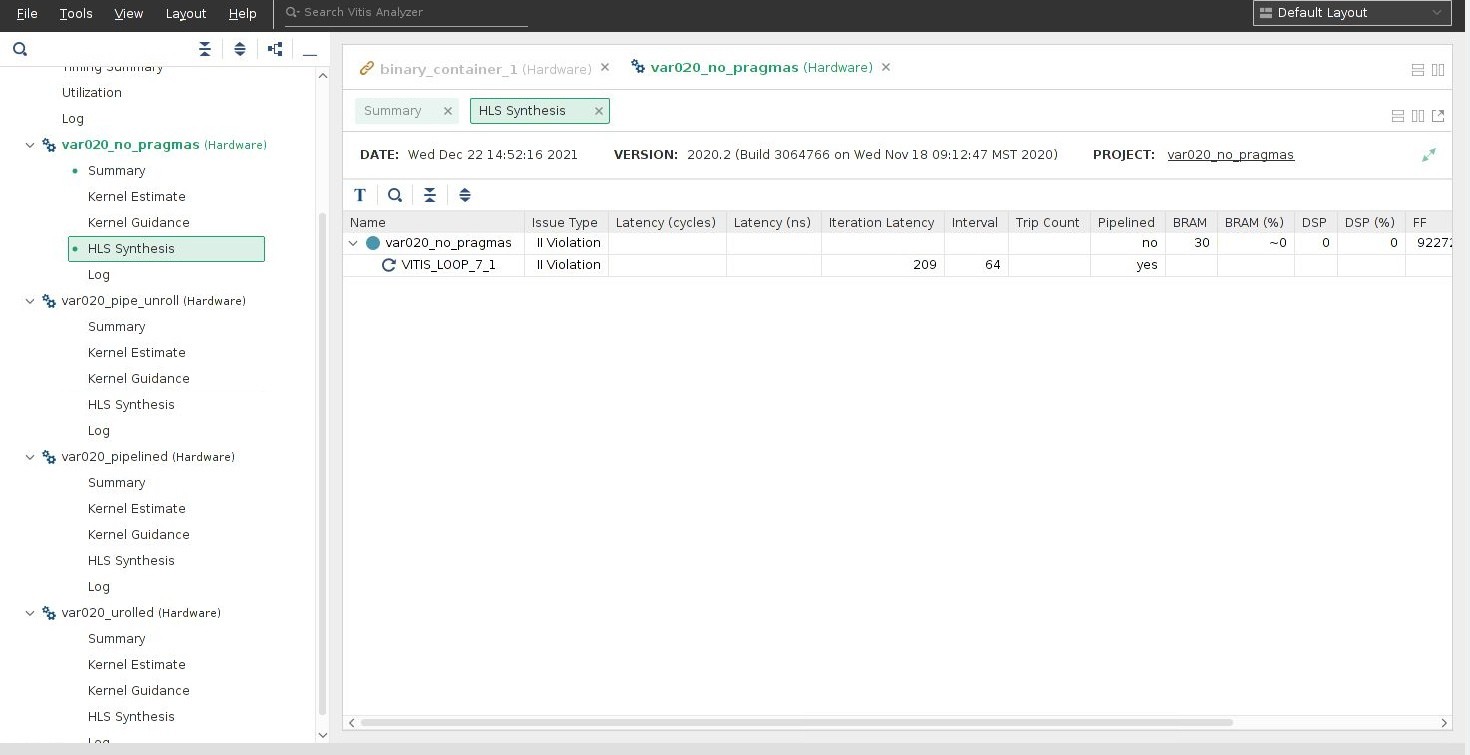
## Рисунок 1.13 – Результаты рабты приложения в режиме Hardware

На рисунке [1.14](#_bookmark18) представлены результаты работы приложения в режиме Hardware.



## Рисунок 1.14 – Результаты рабты приложения в режиме Hardware

На рисунке [1.15](#_bookmark19) представлены результаты работы приложения в режиме Hardware.



## Рисунок 1.15 – Результаты рабты приложения в режиме Hardware

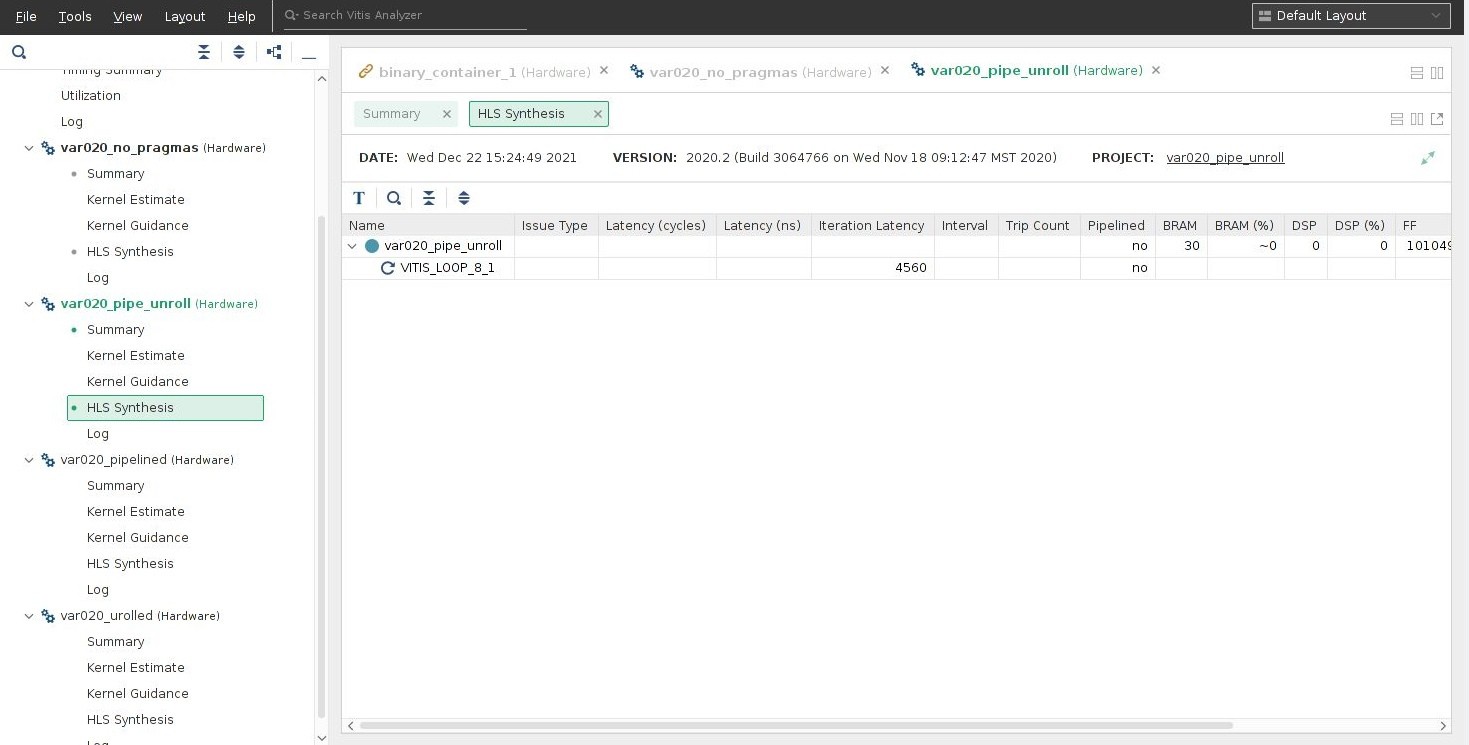
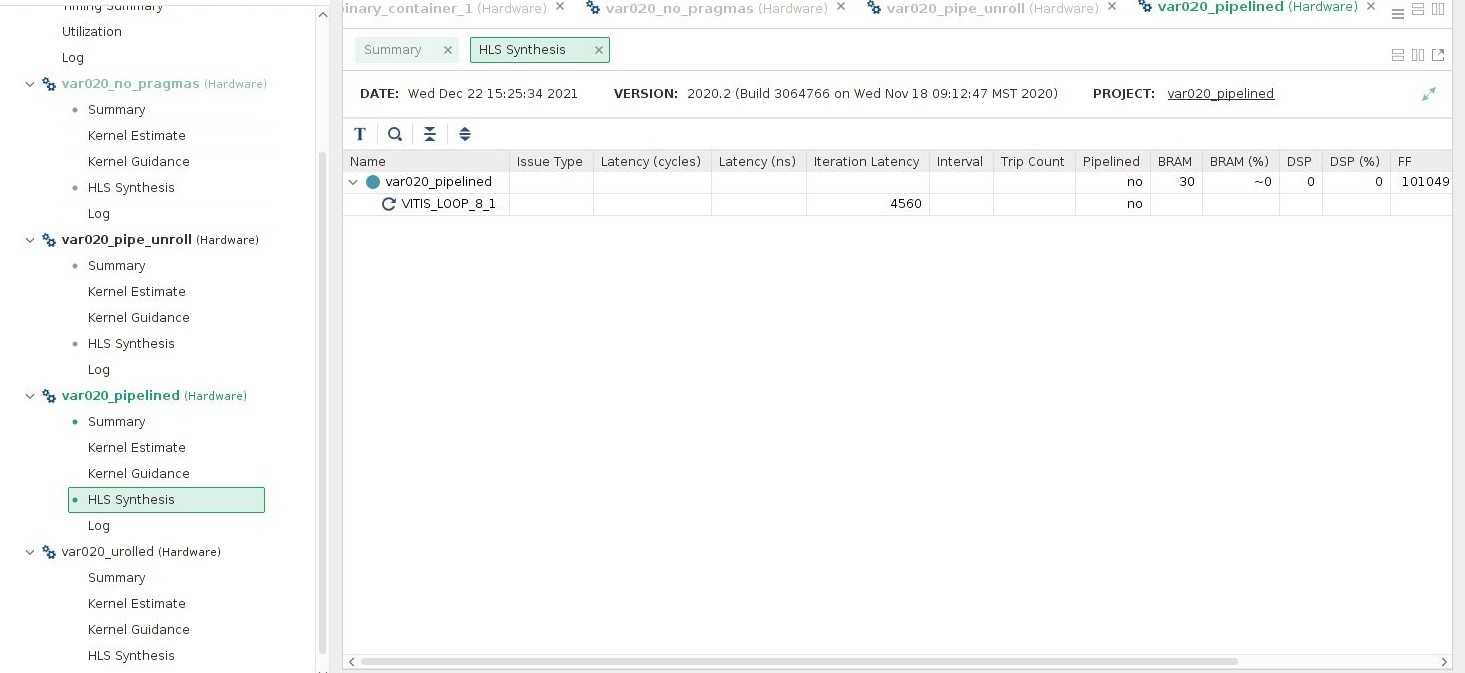


Рисунок 1.16 – Результаты рабты приложения в режиме Hardware



## Рисунок 1.17 – Результаты рабты приложения в режиме Hardware

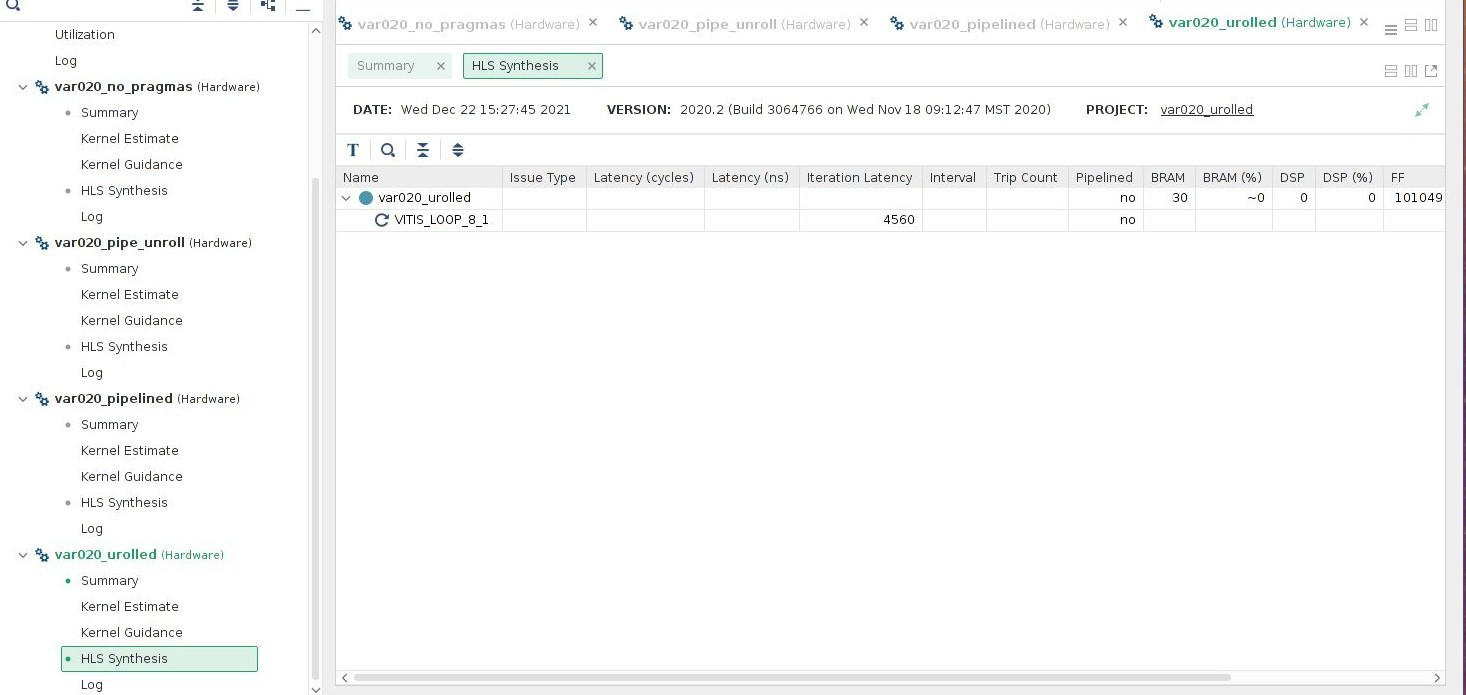


Рисунок 1.18 – Результаты рабты приложения в режиме Hardware

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.

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.

Изучены методики и технологии синтеза аппаратных устройств ускорения вычис- лений по описаниям на языках высокого уровня.

Были выполнены следующие задачи:

* + рассматреть маршрут проектирования устройств, представленных в виде синтак- сических конструкций ЯВУ C/C++;
  + изучены принципы работы IDE Xilinx Vitis HLS;
  + изучина методика анализа и отладки устройств;
  + разработан ускоритель вычислений по индивидуальному заданию;
  + разработан код для тестирования ускорителя;
  + реализован ускоритель с помощью средств высоко-уровненного синтеза, выпол- нить его отладку.

Поставленная цель достигнута.