|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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 **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 5 |

**Название:**

Разработка ускорителей вычислений средствами САПР высокоуровневого синтеза Xilinx Vitis HLS

**Дисциплина:** Архитектура ЭВМ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-52Б |  |  | И.С. Климов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | А.Ю. Попов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2021

**Содержание**

[**1.** **Исходные данные** 4](#_Toc92492436)

[**2.** **Режим программной эмуляции (Emulation-SW)** 6](#_Toc92492437)

[**3.** **Режим аппаратной эмуляции (Emulation-HW)** 7](#_Toc92492438)

[**4.** **Режим аппаратного исполнения (Hardware)** 9](#_Toc92492439)

[**Вывод** 13](#_Toc92492440)

[**Контрольные вопросы** 14](#_Toc92492441)

[**1.** **Вопрос 1** 14](#_Toc92492442)

[**2.** **Вопрос 2** 14](#_Toc92492443)

[**3.** **Вопрос 3** 15](#_Toc92492444)

[**4.** **Вопрос 4** 16](#_Toc92492445)

[**5.** **Вопрос 5** 16](#_Toc92492446)

# **Введение**

**Цель работы –** изучить методики и технологии синтеза аппаратных устройств ускорения вычислений по описаниям на языках высокого уровня. Для достижения цели были поставлены и решены следующие **задачи**:

* изучить маршрут проектирования устройств, представленных в виде синтаксических конструкций языков высокого уровня C/C++;
* изучить основные возможности, средства отладки и анализа, которые предоставляет IDE Xilinx Vitis HLS для разработчиков ускорителей;
* разработать программу для ускорителя вычислений по индивидуальному заданию и протестировать ее.

# **Исходные данные**

На рисунке 1.1 представлен изначальный код программы в соответствии с **вариантом 8**.

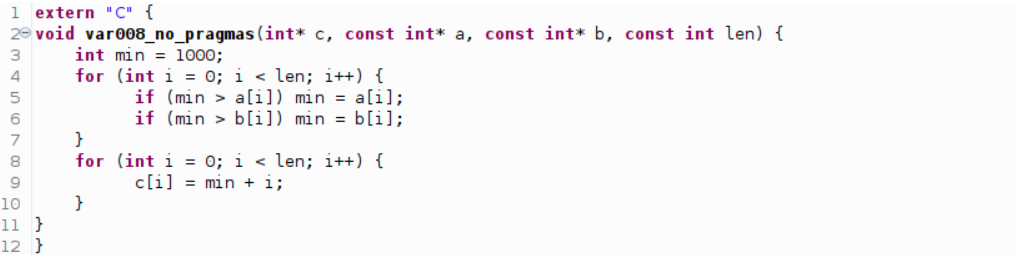


Рисунок 1.1 – изначальный код программы

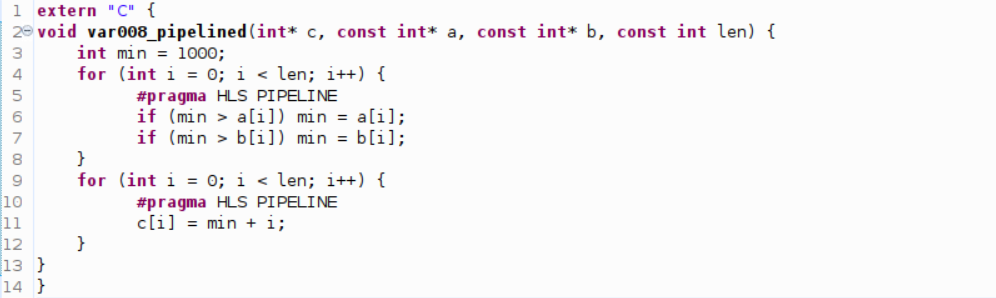
На рисунках 1.2 – 1.4 приведены коды программ, в которых используются директивы, указывающие компилятору, как оптимизировать программу.

Рисунок 1.2 – Оптимизированный код программы с директивой, указывающей препроцессору на конвейерную обработку циклов

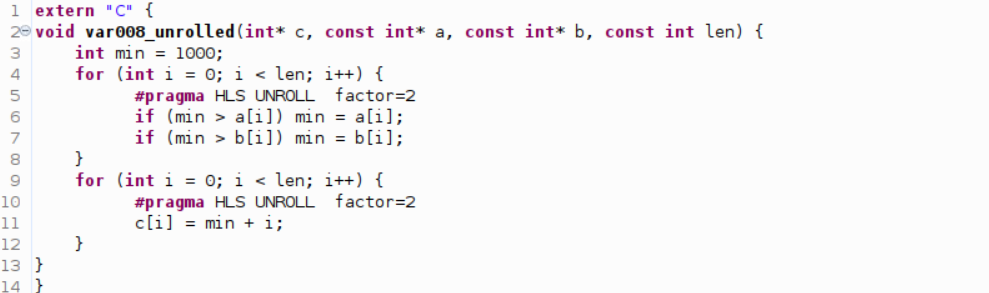


Рисунок 1.3 – Оптимизированный код программы с директивой, указывающей препроцессору на разворачивание циклов

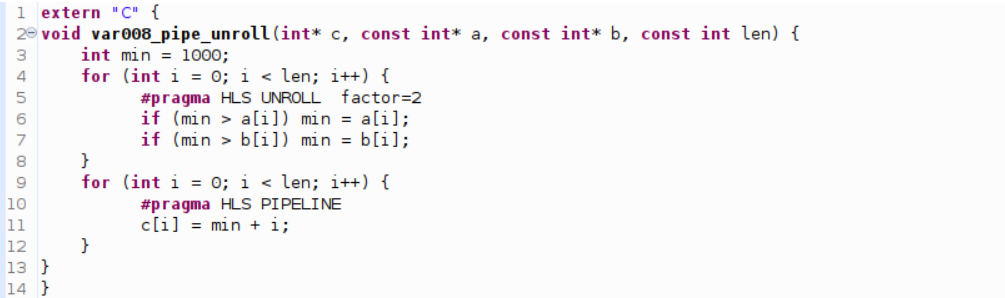


Рисунок 1.4 – Оптимизированный код программы с директивами, указывающими препроцессору на конвейерную обработку одного цикла и разворачивание второго

# **Режим программной эмуляции (Emulation-SW)**

На рисунке 2.1 представлен результат работы в режиме программной эмуляции.

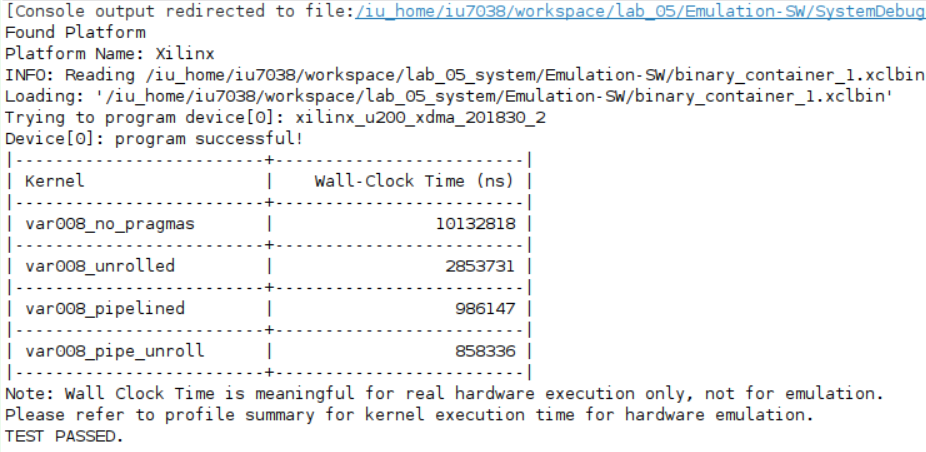


Рисунок 2.1 – Результат работы в режиме Emulation-SW (сравнительная таблица со временем выполнения)

# **Режим аппаратной эмуляции (Emulation-HW)**

На рисунках 3.1 – 3.2 представлены копии экрана с открытым Assistant View.

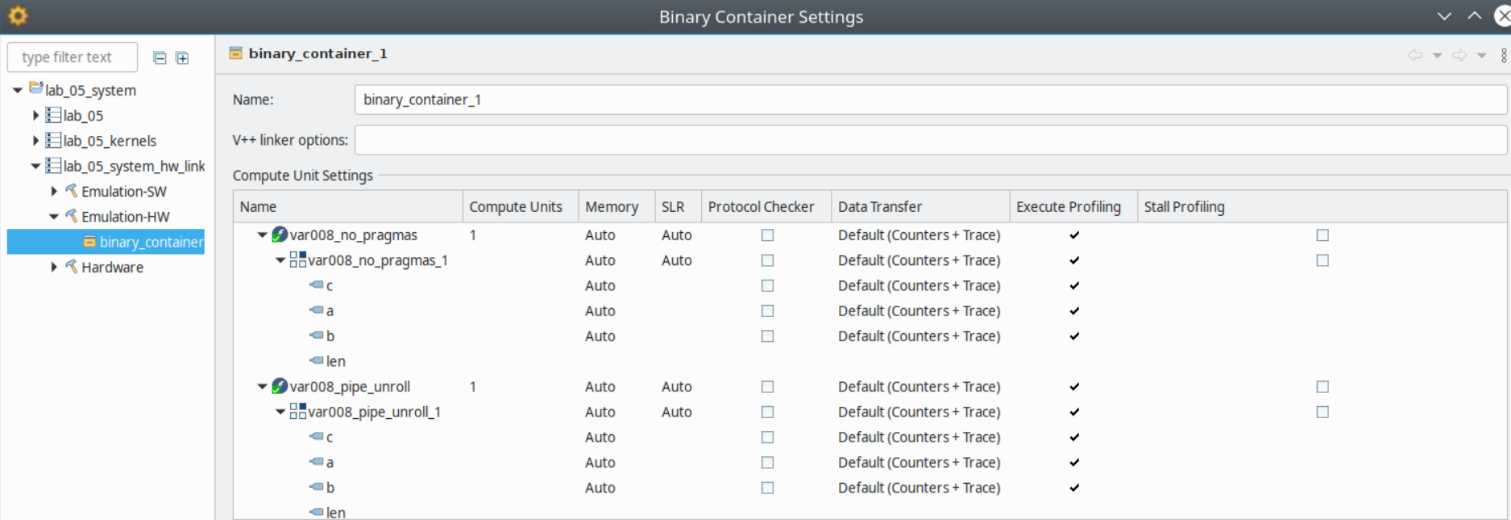


Рисунок 3.1 – Копия экрана Assistant View (часть 1)

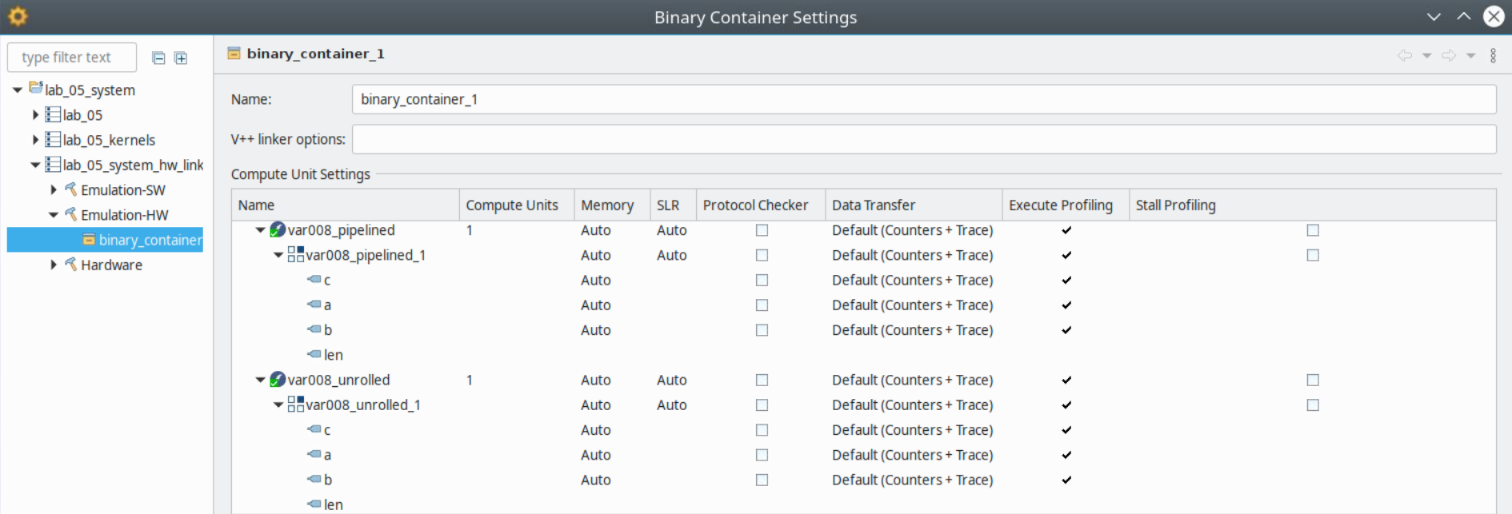


Рисунок 3.2 – Копия экрана Assistant View (часть 2)

На рисунке 3.3 представлен результат работы приложения в режиме аппаратной эмуляции (окно внутрисхемного отладчика Vivado с диаграммами работы четырех ядер).

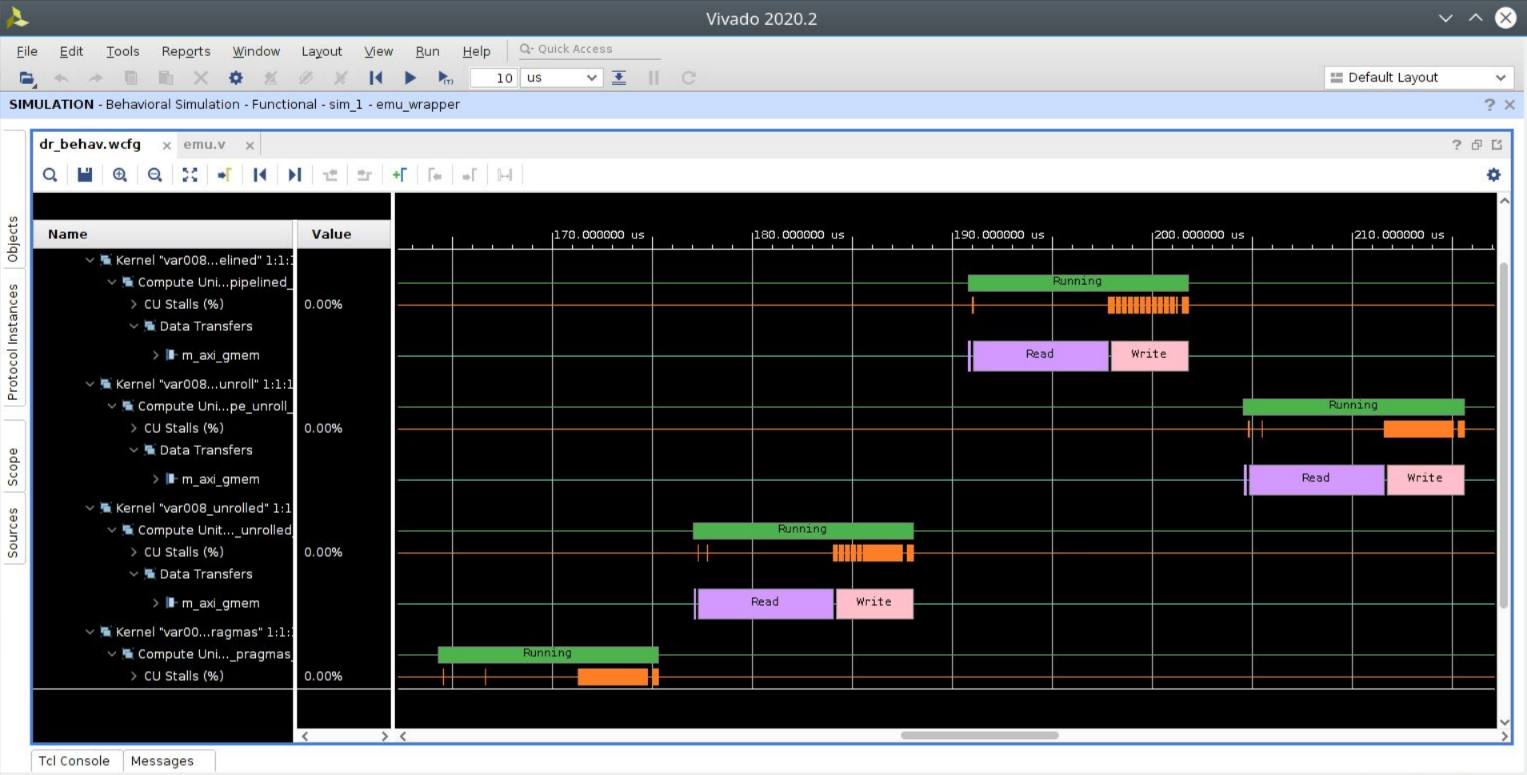


Рисунок 3.3 – Результат работы в режиме Emulation-SW (диаграммы работы четырех ядер)

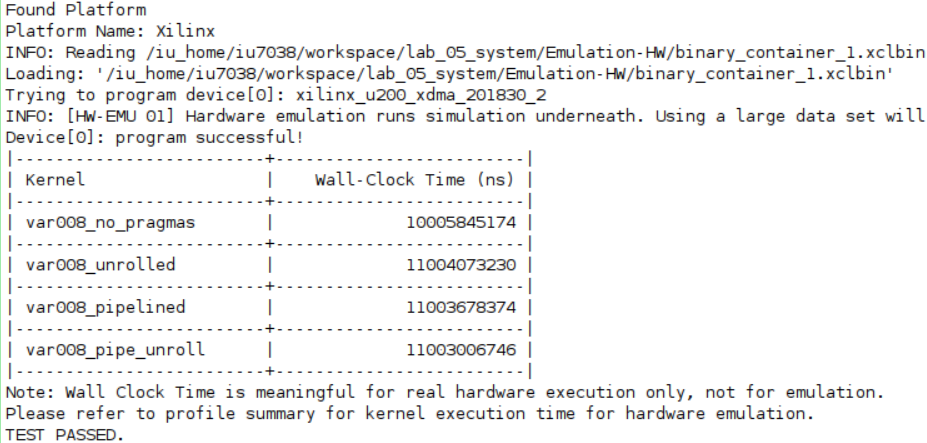
На рисунке 3.4 представлена сравнительная таблица со временем выполнения.

Рисунок 3.4 – Результат работы в режиме Emulation-HW (сравнительная таблица со временем выполнения)

# **Режим аппаратного исполнения (Hardware)**

На рисунке 4.1 представлена сравнительная таблица со временем выполнения.

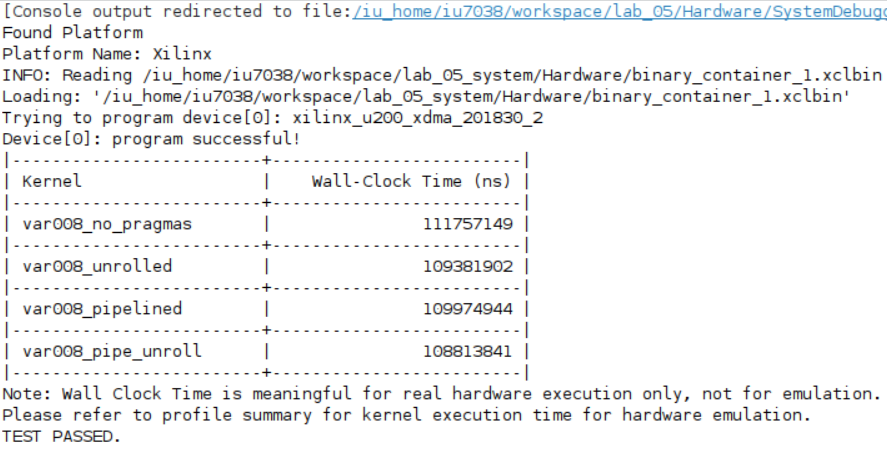


Рисунок 4.1 – Результат работы в режиме Hardware (сравнительная таблица со временем выполнения)

На рисунках 4.2 – 4. представлены копии экранов со вкладок из Link Summary.

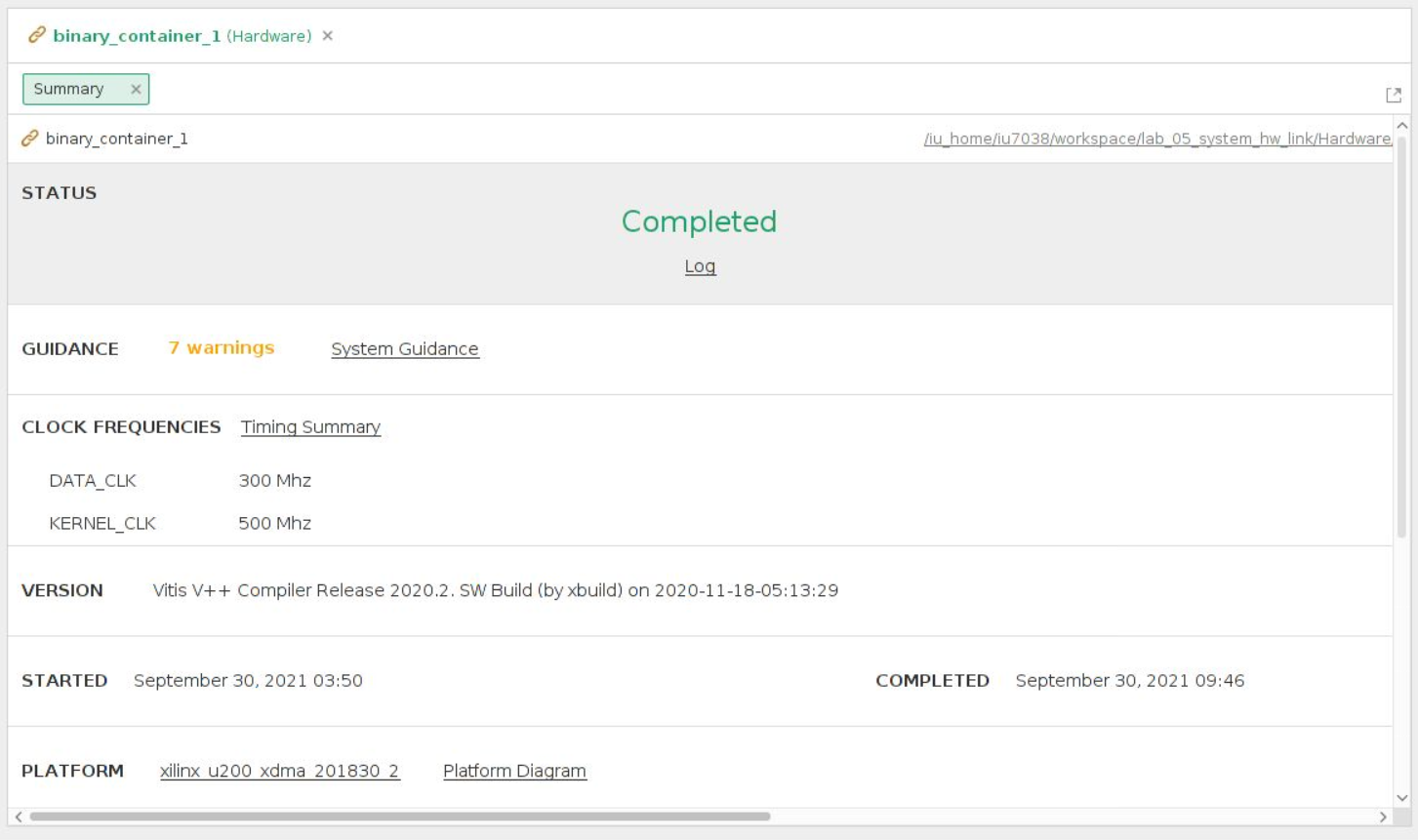


Рисунок 4.2 – Копия экрана со вкладкой Summary

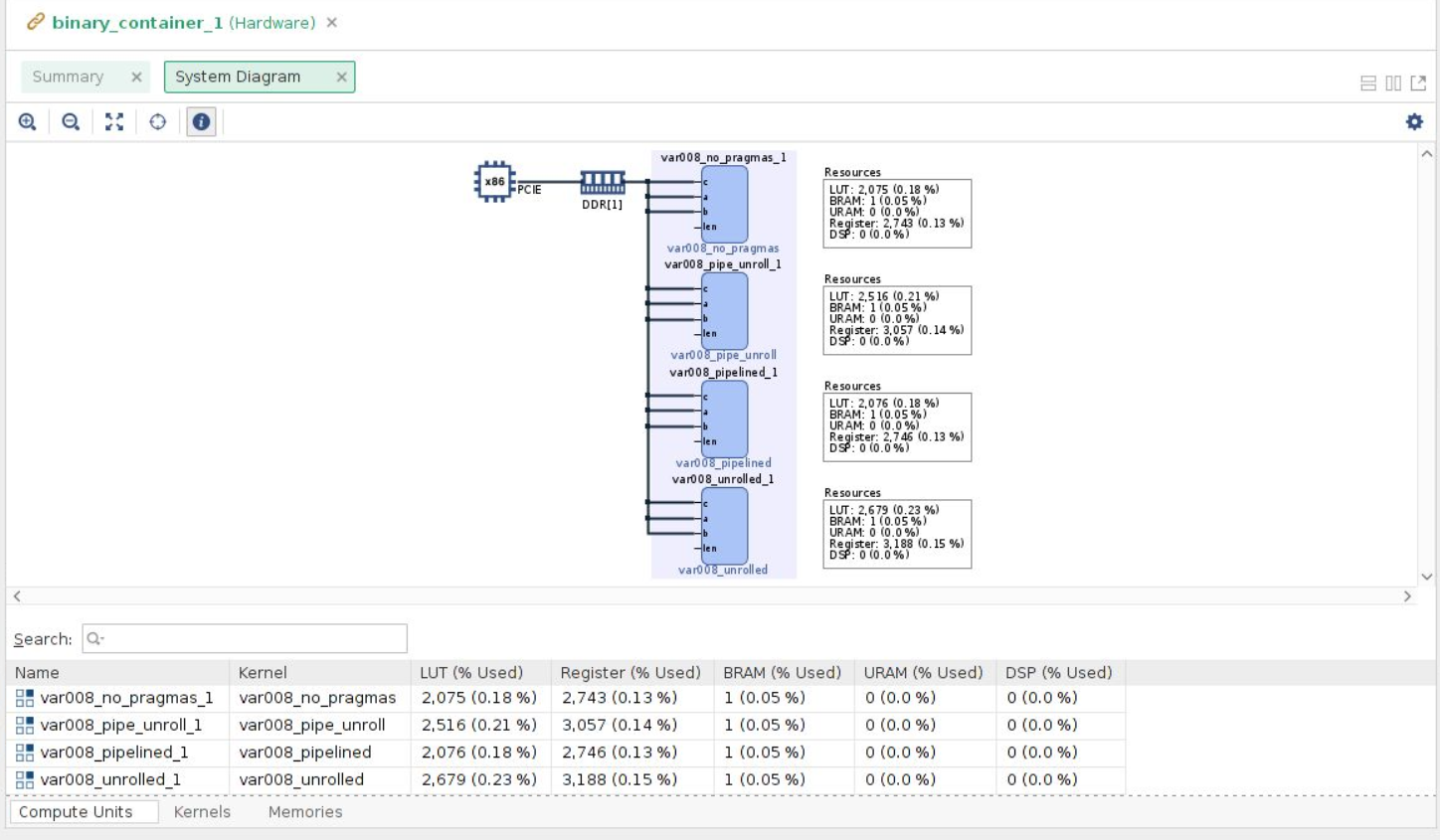


Рисунок 4.3 – Копия экрана со вкладки System Diagram

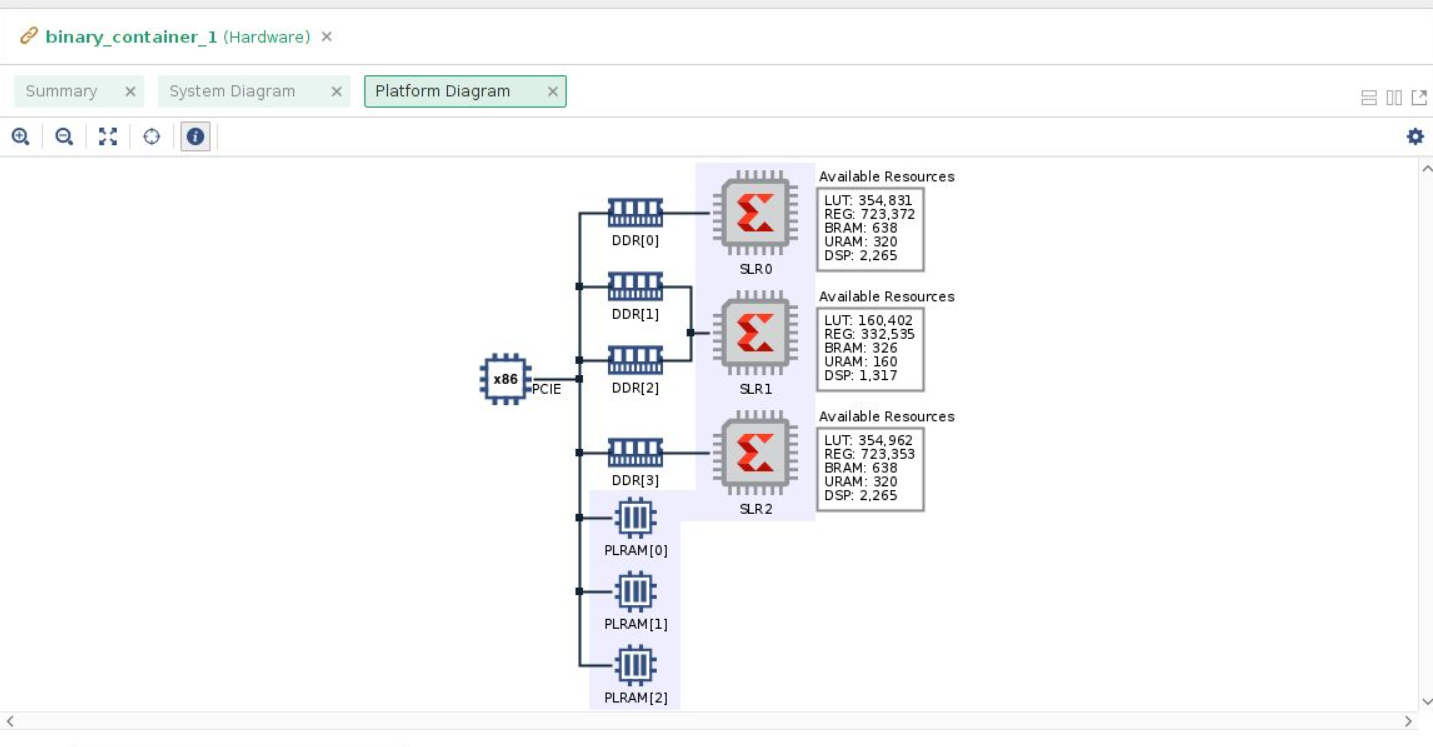


Рисунок 4.4 – Копия экрана со вкладки Platform Diagram (часть 1)

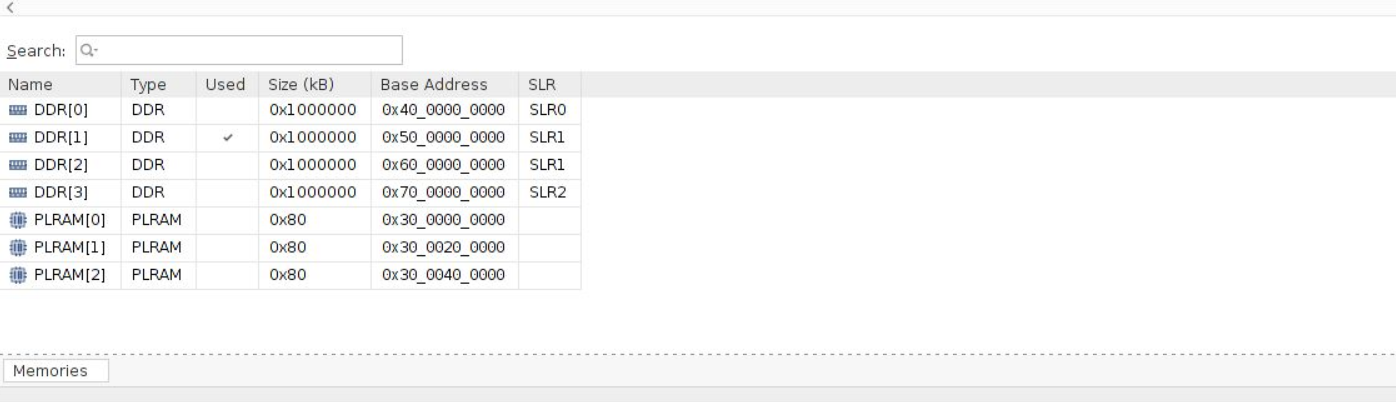


Рисунок 4.5 – Копия экрана со вкладки Platform Diagram (часть 2)

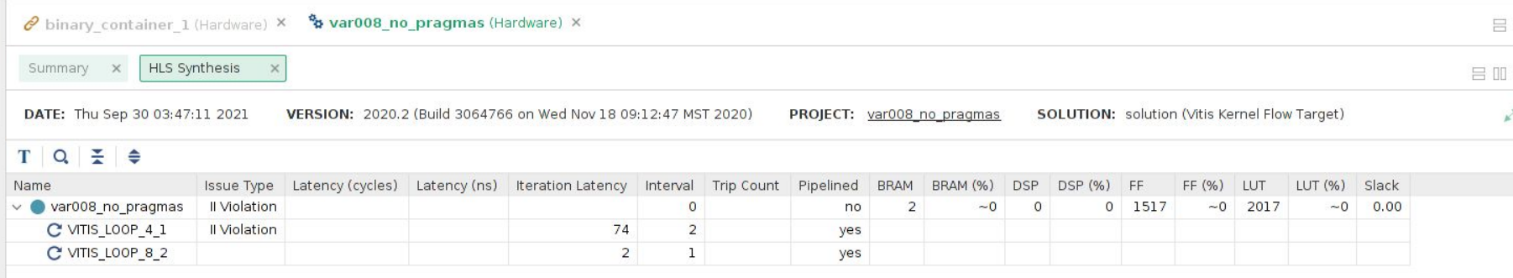


Рисунок 4.6 – Копия экрана со вкладки HLS Synthesis (без оптимизаций)

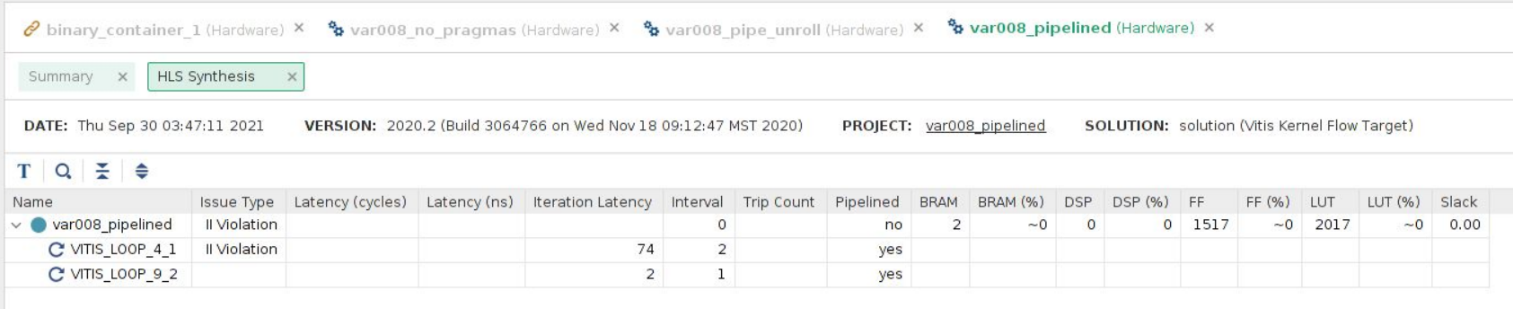


Рисунок 4.6 – Копия экрана со вкладки HLS Synthesis (конвейерная обработка циклов)

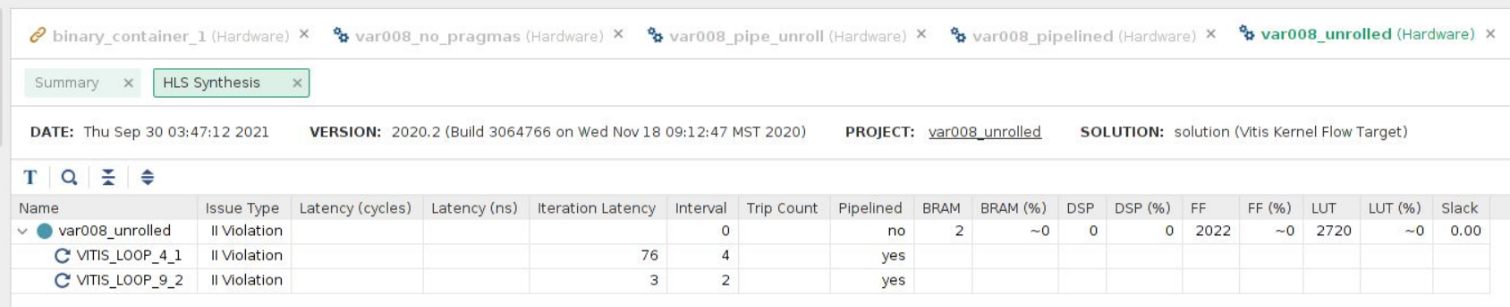


Рисунок 4.6 – Копия экрана со вкладки HLS Synthesis (развертывание циклов)

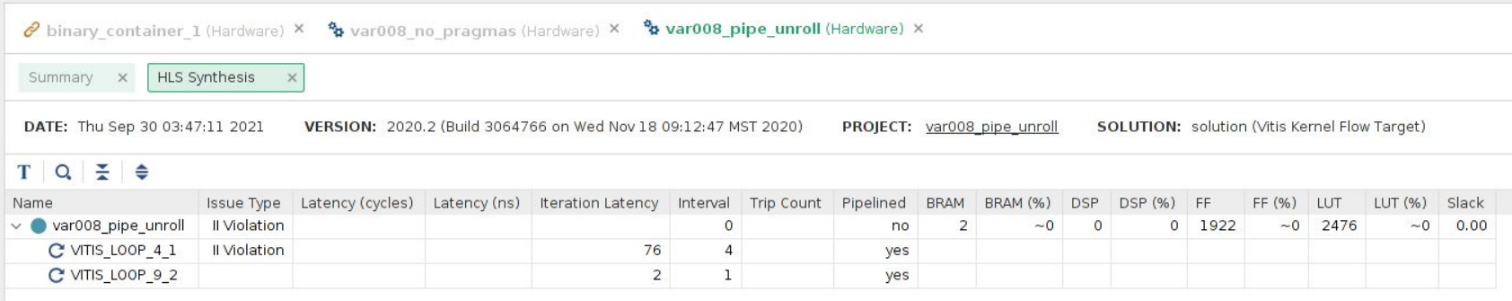


Рисунок 4.6 – Копия экрана со вкладки HLS Synthesis (конвейеризация и развертывание циклов)

# **Вывод**

Таким образом, были изучены методики и технологии синтеза аппаратных устройств ускорения вычислений по описаниям на языках высокого уровня.

Конвейерная обработка цикла приводит к разворачиванию любых циклов, вложенных внутрь конвейерного цикла. Если внутри цикла существуют зависимости по данным, может оказаться невозможным достичь запуска новой итерации в каждом такте, и результатом может быть больший интервал инициации.

При разворачивании циклов все развернутые итераций цикла будут выполняться параллельно, поэтому для реализации такого варианта оптимизации потребуется больший объем программируемых логических ресурсов. В результате компилятор может столкнуться с проблемами, связанными с таким большим количеством ресурсов, и с проблемами емкости, которые замедляют процесс компиляции ядра.

Оба варианта оптимизации привели к уменьшению количества тактов, и как следствие, к уменьшению времени выполнения. Комбинирование двух методов также привело к более эффективному решению.

# **Контрольные вопросы**

Ниже представлены ответ на контрольные вопросы.

## **Вопрос 1**

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

Аппаратные ускорители в отличие от CPU и графических ускорителей представляют собой полностью настраиваемую архитектуру, которую разработчик может использовать для размещения вычислительных блоков с требуемой функциональностью. Возможность настроить аппаратное обеспечение под специализированную задачу позволяет достичь высокой производительности. Также ПЛИС позволяет достичь лучшего показателя энергоэффективности. Графические процессоры масштабируют производительность за счет большого количества ядер и использования параллелизма SIMD/SIMT.

## **Вопрос 2**

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

К основным способам оптимизации циклических конструкций относится:

1. **Конвейеризация** – позволяет повысить пропускную способность за счет увеличения времени синтеза программы и требований к ресурсам ПЛИС. При конвейеризации вместо комплексного преобразования входных данных в одной сложной схеме используются последовательные простые операции, каждая из которых выполняется в своем цифровом узле, а промежуточные результаты запоминаются в триггерах. Это упрощение преобразований позволяет уменьшить число последовательных ячеек от триггера до триггера. Для указания компилятору о необходимости конвейеризировать циклическую обработку используется директива PIPELINE;
2. **Разворачивание циклов –** итерации начинают выполняться параллельно на собственном наборе оборудования. Количество тактов на выполнение всего цикла уменьшается за счет роста размера схемы. Для указания компилятору о необходимости развернуть цикл используется директива UNROLL.

## **Вопрос 3**

**Назовите этапы работы программной части ускорителя в хост системе.**

Этапы работы программной части ускорителя в хост системе:

1. вычисление размера массива;
2. объявление и инициализация исходных массивов;
3. получение списка устройств и инициализация контекста;
4. создание контекста и очередей команд к устройствам;
5. получение необходимой информации об устройстве;
6. создание программного объекта OpenCL и загрузка программы в двоичном формате на ускоритель;
7. выделение памяти под буферы устройства;
8. для запуска каждой реализации
   1. установка необходимых для тестирования значения;
   2. копирование содержимое буферов в DDR память ускорительной карты;
   3. запуск задачи на исполнение и ожидание готовности по прерыванию;
   4. чтение метки времени исполнения задачи;
   5. чтение данных из DDR памяти устройства в буфер результатов.

## **Вопрос 4**

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

1. При отладке в режиме программной эмуляции (Emulation-SW) код ядра компилируется для работы на ЦПУ 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, понимания или проверки поведения системы.
2. Для отладки в режиме аппаратной эмуляции (Emulation-HW) код ядра компилируется в аппаратную модель (RTL), которая запускается в специальном симуляторе на ЦПУ. Этот вариант сборки занимает больше времени, но обеспечивает подробное и точное представление активности ядра. Данный вариант сборки полезен для тестирования функциональности ускорителя и получения начальных оценок производительности.
3. Для отладки в режиме аппаратного обеспечения (Hardware) код ядра компилируется в RTL, а затем реализуется на FPGA. В результате чего формируется двоичный файл xclbin, который будет работать на реальной FPGA.

## **Вопрос 5**

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

Для оптимизации ускорителей можно использовать:

1. отладчик, имеющий графический интерфейс;
2. конструкции, указывающие компилятору путь оптимизации (прагмы и директивы (𝑠𝑒𝑡\_𝑑𝑖𝑟𝑒𝑐𝑡𝑖𝑣𝑒\_\*));
3. средство анализа Vivado IDE, позволяющее также оценивать время и затраты после синтеза или размещения, выполнять симуляцию выполнения программы на ускорителе, после высокоуровневого синтеза оптимизировать проекты на уровне межрегистровых передач.