



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"

Тема Разработка ускорителей вычислений средствами САПР Xilinx Vitis HLS

Студент Прянишников А. Н.

Группа ИУ7-55Б

Оценка (баллы) _____

Преподаватели Попов А.Ю.

Цель работы

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

В ходе лабораторной работы рассматривается маршрут проектирования устройств, представленных в виде синтаксических конструкций ЯВУ C/C++, изучаются принципы работы IDE Xilinx Vitis HLS и методика анализа и отладки устройств. В ходе работы необходимо разработать ускоритель вычислений по индивидуальному заданию, разработать код для тестирования ускорителя, реализовать ускоритель с помощью средств высокоуровневого синтеза, выполнить его отладку.

Ход работы

Вариант индивидуального задания

Мой вариант – 6.

На листинге 1 представлен исходный код индивидуального задания.

Листинг 1: Исходный код индивидуального задания

```
extern "C"
{
void var006(int* c, const int* a, const int* b, const int len)
{
    int minB = b[0];
    for (int i = 1; i < len; i++)
    {
        if (minB > b[i])
        {
            minB = b[i];
        }
    }
    int acc = 0;
    for(int i=0; i < len; i++)
    {
        acc += a[i] * minB;
        c[i] = acc;
    }
}
}
```

Файлы функций ядра на основе индивидуального задания

На листингах 2-5 представлен код функций ядра на основе индивидуального задания:

Листинг 2: Код без изменений

```
extern "C"
{
void var006(int* c, const int* a, const int* b, const int len)
{
```

```

int minB = b[0];
for (int i = 1; i < len; i++)
{
    if (minB > b[i])
    {
        minB = b[i];
    }
}
int acc = 0;
for(int i=0; i < len; i++)
{
    acc += a[i] * minB;
    c[i] = acc;
}
}
}

```

Листинг 3: Развернутый цикл

```

extern "C" {
    void var006_unrolled(int* c, const int* a, const int* b, const int
        len)
    {
        int minB = b[0];
        for (int i = 1; i < len; i++)
        {
            #pragma HLS UNROLL factor=2
            if (minB > b[i])
            {
                minB = b[i];
            }
        }
        int acc = 0;
        for(int i=0; i < len; i++)
        {
            #pragma HLS UNROLL factor=2
            acc += a[i] * minB;
            c[i] = acc;
        }
    }
}

```

Листинг 4: Конвейерное исполнение

```

extern "C" {
    void var006_pipelined(int* c, const int* a, const int* b, const
        int len)
    {
        int minB = b[0];

```

```

        for (int i = 1; i < len; i++)
        {
            #pragma HLS PIPELINE
            if (minB > b[i])
            {
                minB = b[i];
            }
        }
        int acc = 0;
        for(int i=0; i < len; i++)
        {
            #pragma HLS PIPELINE
            acc += a[i] * minB;
            c[i] = acc;
        }
    }
}

```

Листинг 5: Развернутый цикл и конвейерное исполнение

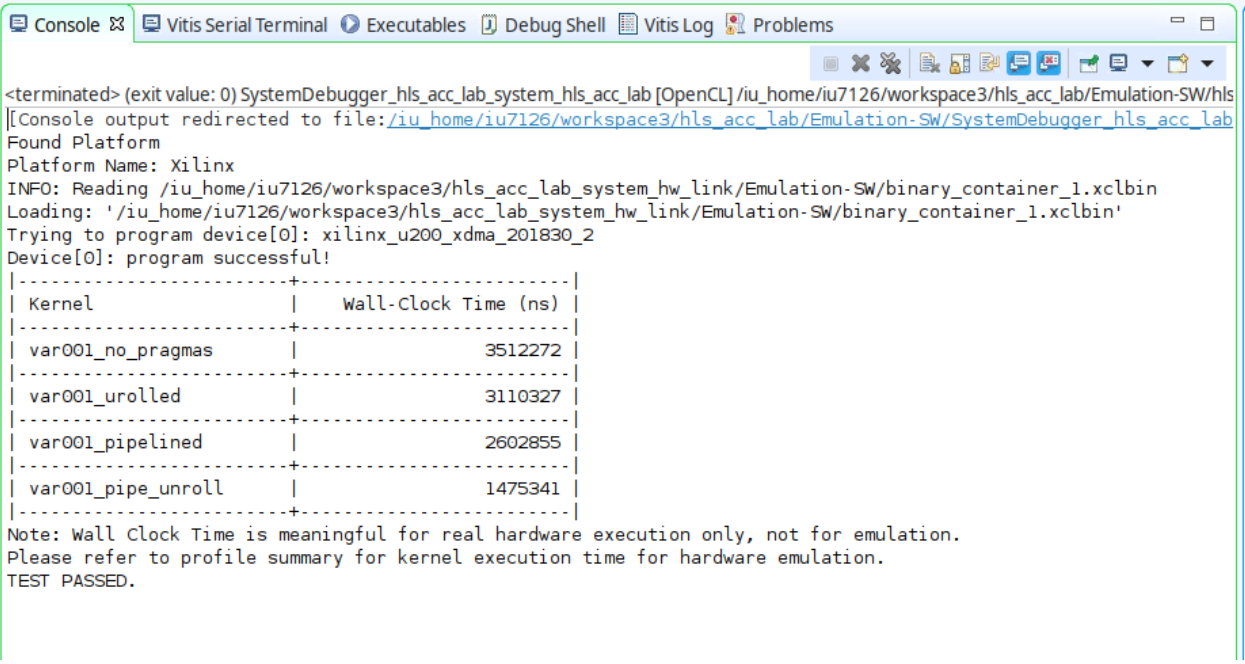
```

extern "C" {
    void var006_pipe_unroll(int* c, const int* a, const int* b, const
        int len)
    {
        int minB = b[0];
        for (int i = 1; i < len; i++)
        {
            #pragma HLS PIPELINE
            #pragma HLS UNROLL factor=2
            if (minB > b[i])
            {
                minB = b[i];
            }
        }
        int acc = 0;
        for(int i=0; i < len; i++)
        {
            #pragma HLS PIPELINE
            #pragma HLS UNROLL factor=2
            acc += a[i] * minB;
            c[i] = acc;
        }
    }
}

```

Результаты работы приложения в режиме Emulation-SW

На рисунке 1 представлен результат работы программы приложения в режиме Emulation-SW.



```
<terminated> (exit value: 0) SystemDebugger_hls_acc_lab_system_hls_acc_lab [OpenCL] /iu_home/iu7126/workspace3/hls_acc_lab/Emulation-SW/hls
[[Console output redirected to file:/iu_home/iu7126/workspace3/hls_acc_lab/Emulation-SW/SystemDebugger_hls_acc_lab
Found Platform
Platform Name: Xilinx
INFO: Reading /iu_home/iu7126/workspace3/hls_acc_lab_system_hw_link/Emulation-SW/binary_container_1.xclbin
Loading: '/iu_home/iu7126/workspace3/hls_acc_lab_system_hw_link/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) |
|-----+-----|
| var001_no_pragmas | 3512272 |
|-----+-----|
| var001_unrolled | 3110327 |
|-----+-----|
| var001_pipelined | 2602855 |
|-----+-----|
| var001_pipe_unroll | 1475341 |
|-----+-----|
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 1: Результат работы программы приложения в режиме Emulation-SW

Видно, что последующие оптимизации ускоряли работу программы. Оптимизация с помощью развёрнутого цикла и конвейерного выполнения показала лучшие результаты. Это связано с тем, что итерации второго цикла могут быть выполнены параллельно, так как не зависят друг от друга.

Копия экрана Assistant View для сборки Emulation-HW

На рисунке 2 представлена копия экрана Assistant View для сборки Emulation-HW

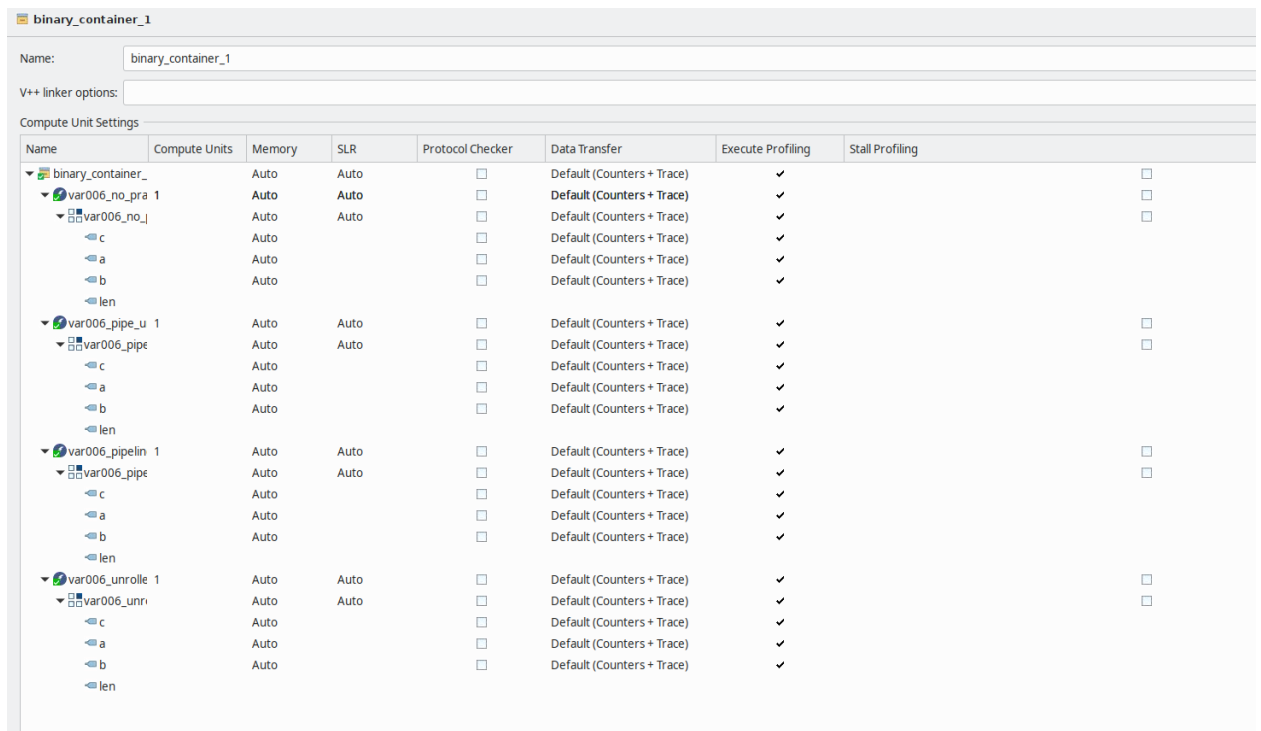


Рисунок 2: Копия экрана Assistant View для сборки Emulation-HW

Результаты работы приложения в режиме Emulation-HW

Так как вывод оказался слишком большим, и на экран не поместился, результат выложен в виде листинга:

На листинге 6 представлен результат работы приложения в режиме Emulation-HW

Листинг 6: Результат работы приложения в режиме Emulation-HW

```
Found Platform
Platform Name: Xilinx
INFO: Reading /iu_home/iu7126/workspace3/hls_acc_lab_system_hw_link/
      Emulation-HW/binary_container_1.xclbin
Loading: '/iu_home/iu7126/workspace3/hls_acc_lab_system_hw_link/Emulation-
      HW/binary_container_1.xclbin'
Trying to program device[0]: xilinx_u200_xdma_201830_2
INFO: [HW-EMU 01] Hardware emulation runs simulation underneath. Using a
      large data set will result in long simulation times. It is recommended
      that a small dataset is used for faster execution. The flow uses
      approximate models for DDR memory and interconnect and hence the
      performance data generated is approximate.
INFO::[ Vitis-EM 22 ] [Time elapsed: 2 minute(s) 11 seconds, Emulation
```

```

    time: 0.0271371 ms]
Data transfer between kernel(s) and global memory(s)
var006_no_pragmas_1:m_axi_gmem-DDR[1]          RD = 0.000 KB
          WR = 0.000 KB
var006_pipe_unroll_1:m_axi_gmem-DDR[1]          RD = 0.000 KB
          WR = 0.000 KB
var006_pipelined_1:m_axi_gmem-DDR[1]            RD = 0.000 KB
          WR = 0.000 KB
var006_unrolled_1:m_axi_gmem-DDR[1]             RD = 0.000 KB
          WR = 0.000 KB

INFO::[ Vitis-EM 22 ] [Time elapsed: 7 minute(s) 13 seconds, Emulation
    time: 0.114581 ms]
Data transfer between kernel(s) and global memory(s)
var006_no_pragmas_1:m_axi_gmem-DDR[1]          RD = 0.000 KB
          WR = 0.000 KB
var006_pipe_unroll_1:m_axi_gmem-DDR[1]          RD = 0.000 KB
          WR = 0.000 KB
var006_pipelined_1:m_axi_gmem-DDR[1]            RD = 0.000 KB
          WR = 0.000 KB
var006_unrolled_1:m_axi_gmem-DDR[1]             RD = 0.000 KB
          WR = 0.000 KB

Device[0]: program successful!
|-----+-----|
| Kernel                | Wall-Clock Time (ns) |
|-----+-----|
| var001_no_pragmas      |          42016715918 |
|-----+-----|
| var001_urolled         |          40017638958 |
|-----+-----|
| var001_pipelined       |          36011311502 |
|-----+-----|
| var001_pipe_unroll     |          34015147242 |
|-----+-----|

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.

INFO::[ Vitis-EM 22 ] [Time elapsed: 12 minute(s) 15 seconds, Emulation
    time: 0.204923 ms]
Data transfer between kernel(s) and global memory(s)
var006_no_pragmas_1:m_axi_gmem-DDR[1]          RD = 8.000 KB
          WR = 4.000 KB
var006_pipe_unroll_1:m_axi_gmem-DDR[1]          RD = 8.000 KB
          WR = 4.000 KB
var006_pipelined_1:m_axi_gmem-DDR[1]            RD = 8.000 KB

```



```

    WR = 4.000 KB
var006_unrolled_1:m_axi_gmem-DDR[1]          RD = 8.000 KB
    WR = 4.000 KB

INFO::[ Vitis-EM 22 ] [Time elapsed: 17 minute(s) 18 seconds, Emulation
    time: 0.292367 ms]
Data transfer between kernel(s) and global memory(s)
var006_no_pragmas_1:m_axi_gmem-DDR[1]          RD = 8.000 KB
    WR = 4.000 KB
var006_pipe_unroll_1:m_axi_gmem-DDR[1]          RD = 8.000 KB
    WR = 4.000 KB
var006_pipelined_1:m_axi_gmem-DDR[1]          RD = 8.000 KB
    WR = 4.000 KB
var006_unrolled_1:m_axi_gmem-DDR[1]          RD = 8.000 KB
    WR = 4.000 KB

INFO::[ Vitis-EM 22 ] [Time elapsed: 19 minute(s) 46 seconds, Emulation
    time: 0.335542 ms]
Data transfer between kernel(s) and global memory(s)
var006_no_pragmas_1:m_axi_gmem-DDR[1]          RD = 8.000 KB
    WR = 4.000 KB
var006_pipe_unroll_1:m_axi_gmem-DDR[1]          RD = 8.000 KB
    WR = 4.000 KB
var006_pipelined_1:m_axi_gmem-DDR[1]          RD = 8.000 KB
    WR = 4.000 KB
var006_unrolled_1:m_axi_gmem-DDR[1]          RD = 8.000 KB
    WR = 4.000 KB

```

Видно, что последующие оптимизации ускоряли работу программы. Оптимизация с помощью развёрнутого цикла и конвейерного выполнения показала лучшие результаты. Это связано с тем, что итерации второго цикла могут быть выполнены параллельно, так как не зависят друг от друга.

Окно внутрисхемного отладчика Vivado для сборки в режиме Emulation-HW

На рисунке 3 представлено окно внутрисхемного отладчика Vivado для сборки в режиме Emulation-HW

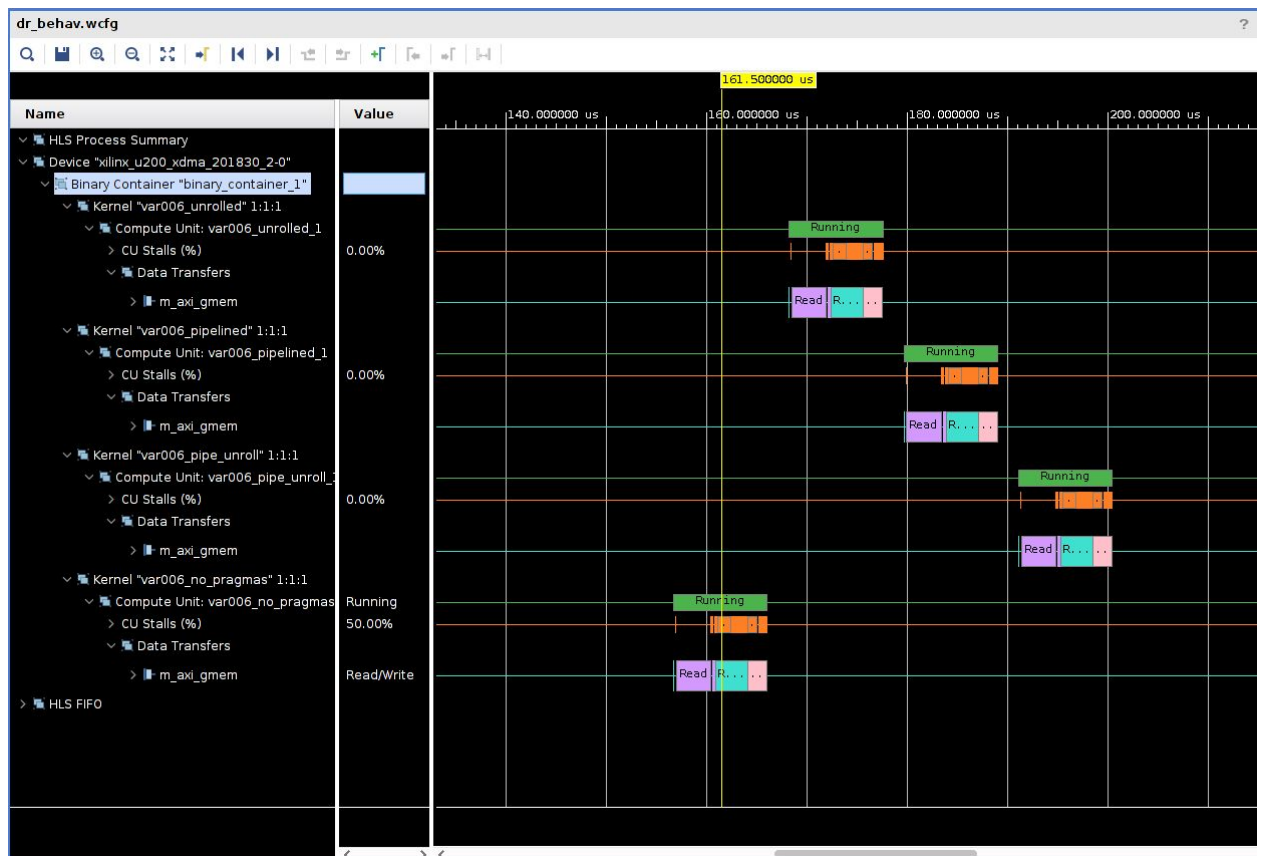
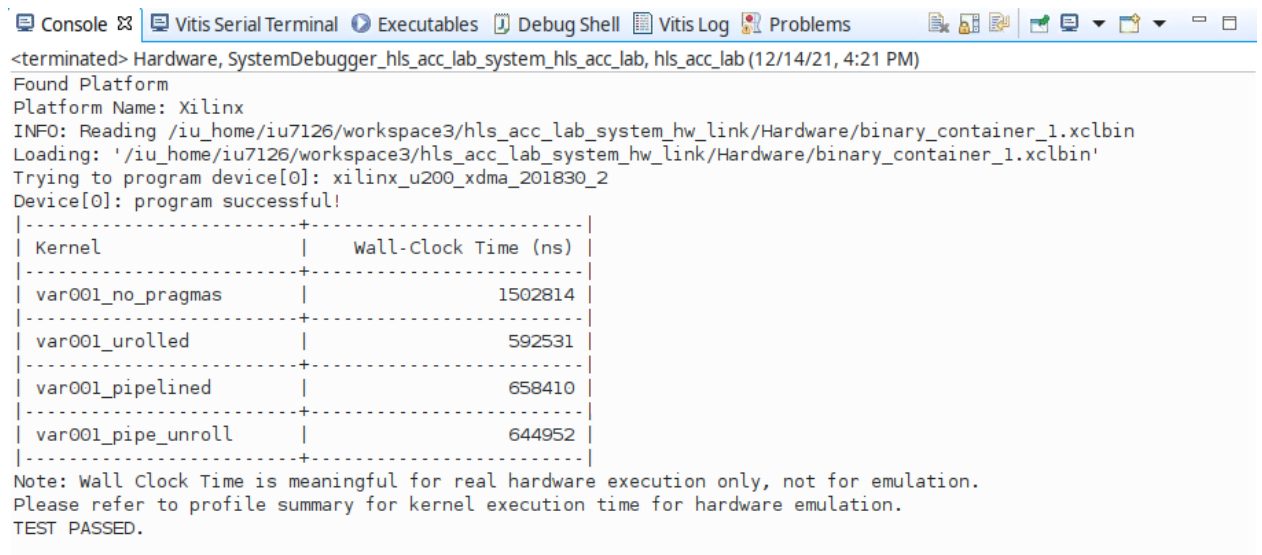


Рисунок 3: Окно внутрисхемного отладчика Vivado для сборки в режиме Emulation-HW

Результаты работы приложения в режиме Hardware

На рисунке 4 представлены результаты работы приложения в режиме Hardware

The screenshot shows the Vitis Serial Terminal interface. At the top, there are tabs for 'Console', 'Vitis Serial Terminal', 'Executables', 'Debug Shell', 'Vitis Log', and 'Problems'. The main window displays the following text: '<terminated> Hardware, SystemDebugger_hls_acc_lab_system_hls_acc_lab, hls_acc_lab (12/14/21, 4:21 PM)'. Below this, it says 'Found Platform', 'Platform Name: Xilinx', 'INFO: Reading /iu_home/iu7126/workspace3/hls_acc_lab_system_hw_link/Hardware/binary_container_1.xclbin', 'Loading: '/iu_home/iu7126/workspace3/hls_acc_lab_system_hw_link/Hardware/binary_container_1.xclbin'', 'Trying to program device[0]: xilinx_u200_xdma_201830_2', and 'Device[0]: program successful!'. A table follows, showing kernel execution times in nanoseconds. The table has two columns: 'Kernel' and 'Wall-Clock Time (ns)'. The data rows are: 'var001_no_pragmas' (1502814), 'var001_unrolled' (592531), 'var001_pipelined' (658410), and 'var001_pipe_unroll' (644952). Below the table, a note states: '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.' The final line is 'TEST PASSED.'

```
<terminated> Hardware, SystemDebugger_hls_acc_lab_system_hls_acc_lab, hls_acc_lab (12/14/21, 4:21 PM)
Found Platform
Platform Name: Xilinx
INFO: Reading /iu_home/iu7126/workspace3/hls_acc_lab_system_hw_link/Hardware/binary_container_1.xclbin
Loading: '/iu_home/iu7126/workspace3/hls_acc_lab_system_hw_link/Hardware/binary_container_1.xclbin'
Trying to program device[0]: xilinx_u200_xdma_201830_2
Device[0]: program successful!
+-----+-----+
| Kernel | Wall-Clock Time (ns) |
+-----+-----+
| var001_no_pragmas | 1502814 |
+-----+-----+
| var001_unrolled | 592531 |
+-----+-----+
| var001_pipelined | 658410 |
+-----+-----+
| var001_pipe_unroll | 644952 |
+-----+-----+
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 4: Результаты работы приложения в режиме Hardware

Самым быстрым оказался вариант только с развёрнутым циклом, но все оптимизированные варианты тратят в 2.5 раза меньше тактов, чем стандартная программа.

Копия экрана для вкладок «Summary», «System Diagram», «Platform Diagram» и четыре вкладки «HLS Synthesis» для каждого ядра сборки Hardware.

На рисунке 5 представлена копия экрана для вкладки Summary:

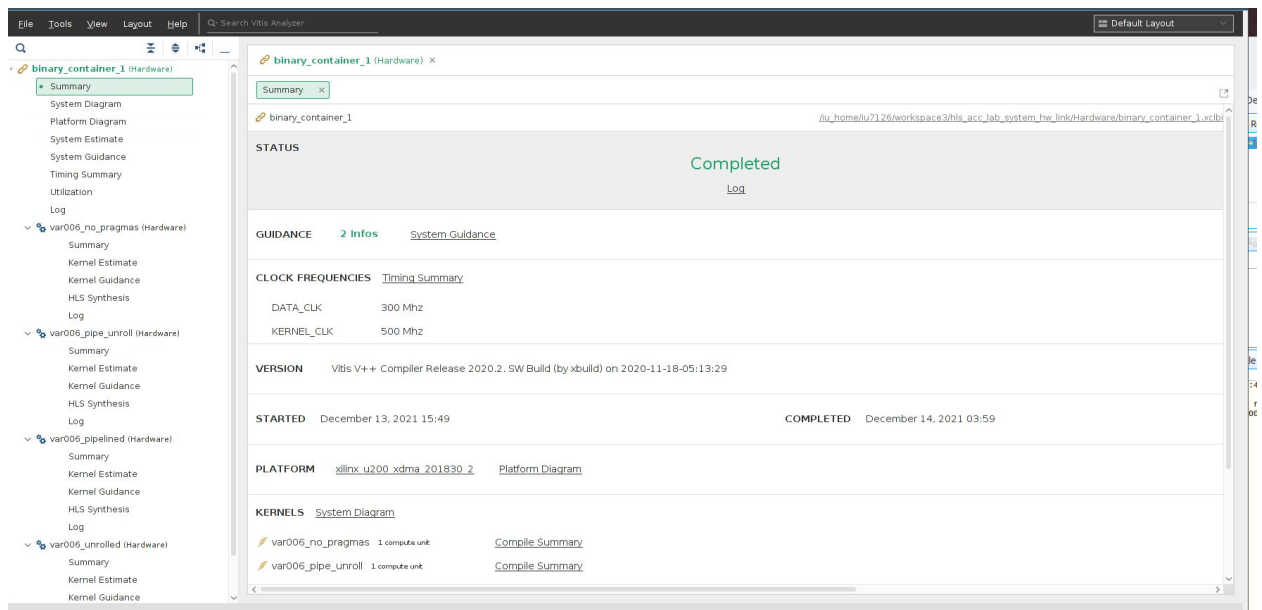


Рисунок 5: Копия экрана для вкладки Summary

На рисунке 6 представлена копия экрана для вкладки System Diagram:

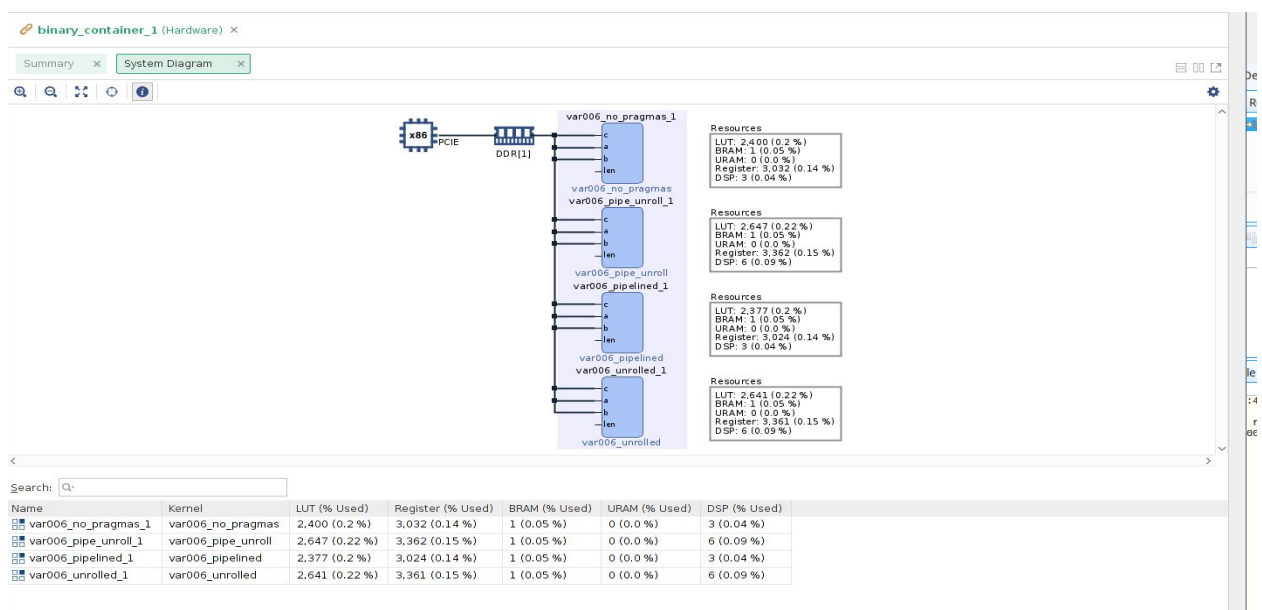


Рисунок 6: Копия экрана для вкладки System Diagram

На рисунке 7 представлена копия экрана для вкладки Platform Diagram:

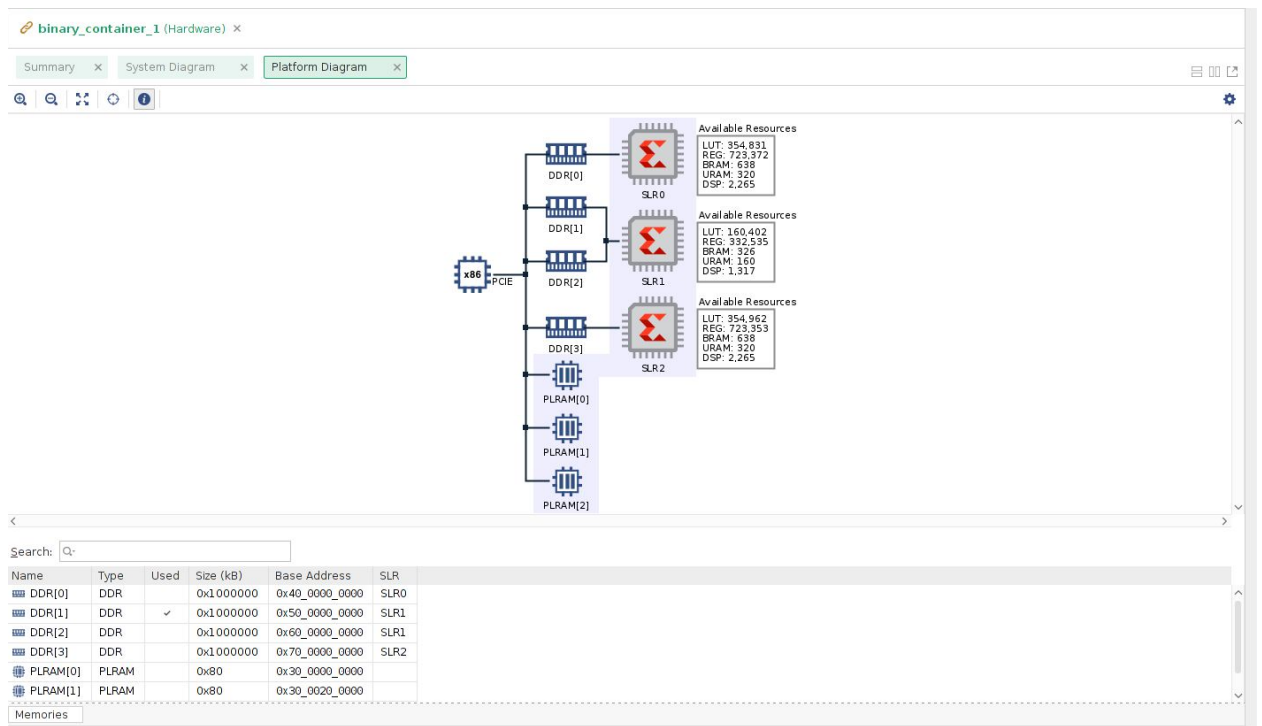


Рисунок 7: Копия экрана для вкладки Platform Diagram:

На рисунке 8 представлена копия экрана для вкладки HLS Synthesis для ядра без оптимизаций

Name	Issue Type	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)	Slack
var006_no_pragmas					0		no	2	~0	0	0	1867	~0	3225	~0	0.00
VITIS_LOOP_5_1				3	1		yes									
VITIS_LOOP_13_2				6	1		yes									

Рисунок 8: Копия экрана для вкладки HLS Synthesis для ядра без оптимизаций

На рисунке 9 представлена копия экрана для вкладки HLS Synthesis для ядра с развёрнутым циклом:

<div> binary_container_1 (Hardware) × var006_no_pragmas (Hardware) × var006_pipe_unroll (Hardware) × var006_pipelined (Hardware) × var006_unrolled (Hardware) × </div>																
<div> Summary × HLS Synthesis × </div>																
DATE: Sun Dec 12 22:46:27 2021 VERSION: 2020.2 (Build 3064766 on Wed Nov 18 09:12:47 MST 2020) PROJECT: var006_unrolled SOLUTION: solution (Vitis Kernel Flow Target)																
<div> T Q Z </div>																
Name	Issue Type	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)	Slack
var006_unrolled	II Violation					0	no	2	~0	0	0	2453	~0	3744	~0	0.00
VITIS_LOOP_5_1	II Violation					4	yes									
VITIS_LOOP_14_2	II Violation					7	yes									

Рисунок 9: Копия экрана для вкладки HLS Synthesis для ядра с развёрнутым циклом

На рисунке 10 представлена копия экрана для вкладки HLS Synthesis для ядра с конвейерным выполнением:

<div> binary_container_1 (Hardware) × var006_no_pragmas (Hardware) × var006_pipe_unroll (Hardware) × var006_pipelined (Hardware) × </div>																
<div> Summary × HLS Synthesis × </div>																
DATE: Sun Dec 12 22:46:26 2021 VERSION: 2020.2 (Build 3064766 on Wed Nov 18 09:12:47 MST 2020) PROJECT: var006_pipelined SOLUTION: solution (Vitis Kernel Flow Target)																
<div> T Q Z </div>																
Name	Issue Type	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)	Slack
var006_pipelined						0	no	2	~0	0	0	1867	~0	3225	~0	0.00
VITIS_LOOP_5_1						3	yes									
VITIS_LOOP_14_2						6	yes									

Рисунок 10: Копия экрана для вкладки HLS Synthesis для ядра с конвейерным выполнением

На рисунке 11 представлена копия экрана для вкладки HLS Synthesis для ядра с развёрнутым циклом и конвейерным выполнением:

<div> binary_container_1 (Hardware) × var006_no_pragmas (Hardware) × var006_pipe_unroll (Hardware) × </div>																
<div> Summary × HLS Synthesis × </div>																
DATE: Sun Dec 12 22:46:27 2021 VERSION: 2020.2 (Build 3064766 on Wed Nov 18 09:12:47 MST 2020) PROJECT: var006_pipe_unroll SOLUTION: solution (Vitis Kernel Flow Target)																
<div> T Q Z </div>																
Name	Issue Type	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)	Slack
var006_pipe_unroll						0	no	2	~0	0	0	2453	~0	3744	~0	0.00
VITIS_LOOP_5_1	II Violation					4	yes									
VITIS_LOOP_15_2	II Violation					7	yes									

Рисунок 11: Копия экрана для вкладки HLS Synthesis для ядра с развёрнутым циклом и конвейерным выполнением

Выводы

В ходе лабораторной работы были изучены принципы работы IDE Xilinx Vitis HLS и методика анализа и отладки устройств. В ходе работы был

разработан ускоритель вычислений по индивидуальному заданию, разработан код для тестирования ускорителя, реализован ускоритель с помощью средств высоко-уровневого синтеза, выполнена его отладка.

Удалось запустить приложение в трёх режимах: Emulation-SW, Emulation-HW и Hardware. С помощью оптимизаций удалось значительно ускорить работу программы. Для Emulation-SW и Emulation-HW быстрее всего отработала оптимизация с помощью развёрнутого цикла и конвейрного выполнения. Для Hardware самым быстрым оказался вариант только с развёрнутым циклом, но все оптимизированные варианты тратят в 2.5 раза меньше тактов, чем стандартная программа.

Оптимизация стала возможна, потому что итерации второго цикла исходного кода могут быть выполнены параллельно, так как не зависят друг от друга.