

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.04 «Программная инженерия»

Дисциплина Архитектура электронно-вычислительных машин

подпись, дата

Цель работы

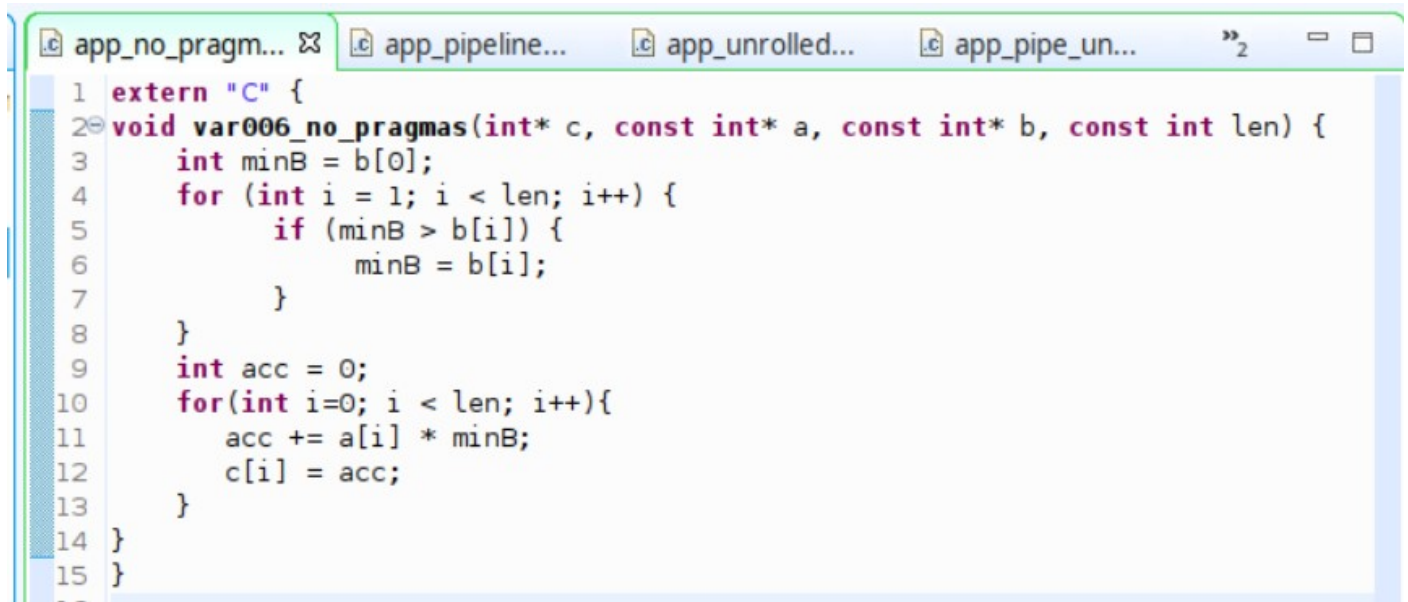
Изучение методики и технологии синтеза аппаратных устройств ускорения вычислений по описаниям на языках высокого уровня.

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 высоко-уровневого синтеза, выполнить его отладку.

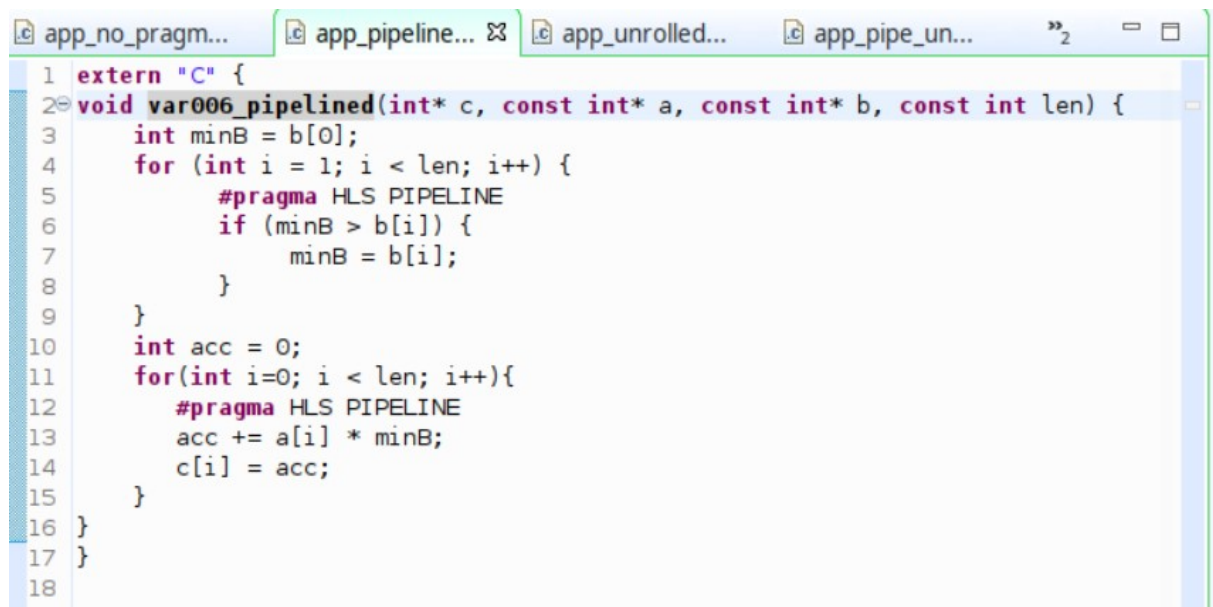
1 Практическая часть

На рисунках 1.1-1.4 приведены файлы функций ядра на основе индивидуального задания.



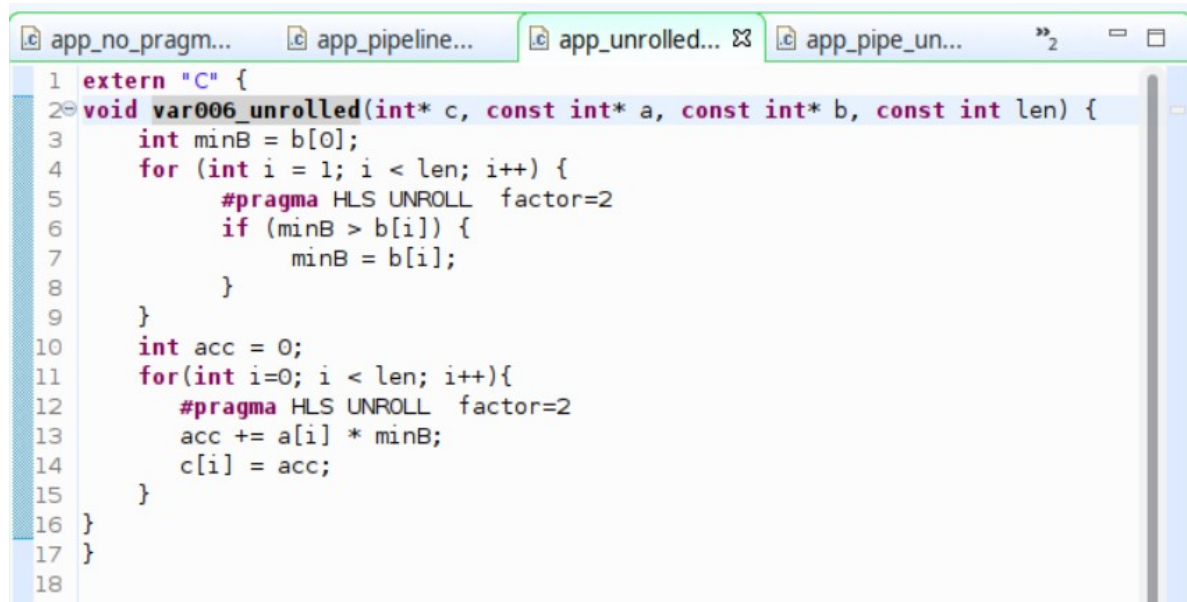
```
1 extern "C" {
2 void var006_no_pragmas(int* c, const int* a, const int* b, const int len) {
3     int minB = b[0];
4     for (int i = 1; i < len; i++) {
5         if (minB > b[i]) {
6             minB = b[i];
7         }
8     }
9     int acc = 0;
10    for(int i=0; i < len; i++){
11        acc += a[i] * minB;
12        c[i] = acc;
13    }
14 }
15 }
```

Рисунок 1.1 – Не оптимизированный цикл



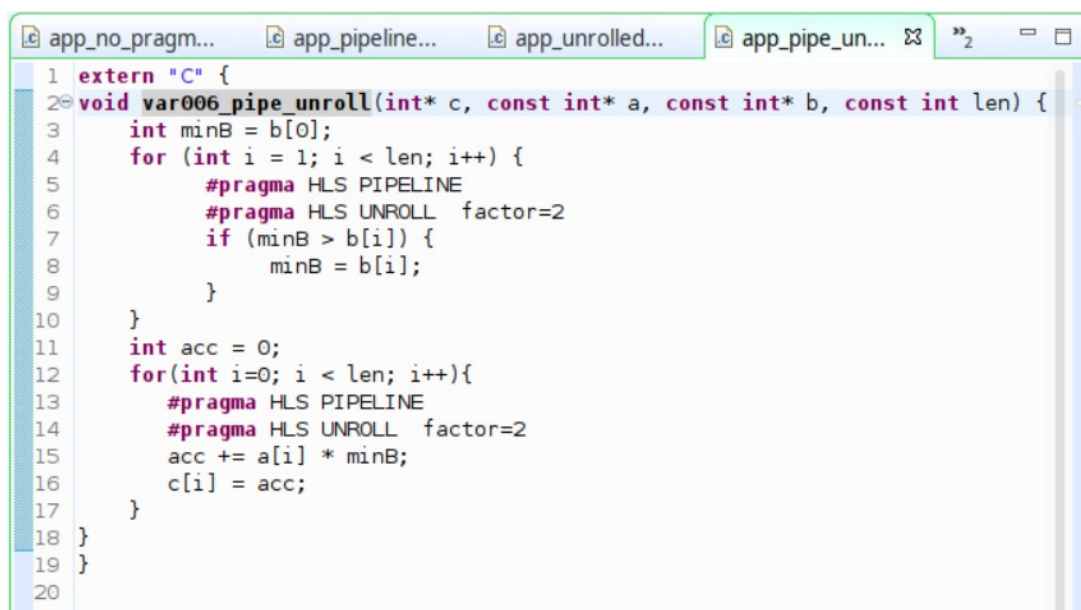
```
1 extern "C" {
2 void var006_pipelined(int* c, const int* a, const int* b, const int len) {
3     int minB = b[0];
4     for (int i = 1; i < len; i++) {
5         #pragma HLS PIPELINE
6         if (minB > b[i]) {
7             minB = b[i];
8         }
9     }
10    int acc = 0;
11    for(int i=0; i < len; i++){
12        #pragma HLS PIPELINE
13        acc += a[i] * minB;
14        c[i] = acc;
15    }
16 }
17 }
18 }
```

Рисунок 1.2 – Конвейерная организация цикла



```
1 extern "C" {
2 void var006_unrolled(int* c, const int* a, const int* b, const int len) {
3     int minB = b[0];
4     for (int i = 1; i < len; i++) {
5         #pragma HLS UNROLL factor=2
6         if (minB > b[i]) {
7             minB = b[i];
8         }
9     }
10    int acc = 0;
11    for(int i=0; i < len; i++){
12        #pragma HLS UNROLL factor=2
13        acc += a[i] * minB;
14        c[i] = acc;
15    }
16 }
17 }
18 }
```

Рисунок 1.3 – Частично развернутый цикл



```
1 extern "C" {
2 void var006_pipe_unroll(int* c, const int* a, const int* b, const int len) {
3     int minB = b[0];
4     for (int i = 1; i < len; i++) {
5         #pragma HLS PIPELINE
6         #pragma HLS UNROLL factor=2
7         if (minB > b[i]) {
8             minB = b[i];
9         }
10    }
11    int acc = 0;
12    for(int i=0; i < len; i++){
13        #pragma HLS PIPELINE
14        #pragma HLS UNROLL factor=2
15        acc += a[i] * minB;
16        c[i] = acc;
17    }
18 }
19 }
20 }
```

Рисунок 1.4 – Конвейерный и частично развернутый цикл

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

```
<terminated> (exit value: 0) SystemDebugger_hls_acc_lab_system_hls_acc_lab [OpenCL] /iu_home/iu7036/workspace/hls_acc_lab
[Console output redirected to file:/iu_home/iu7036/workspace/hls_acc_lab/Emulation-SW/SystemDebugger_hls_acc_lab_system_hls_acc_lab.log]
Found Platform
Platform Name: Xilinx
INFO: Reading /iu_home/iu7036/workspace/hls_acc_lab_system/Emulation-SW/binary_container_1.xclbin
Loading: '/iu_home/iu7036/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) |
|-----|-----|
| var006_no_pragmas | 3059780 |
|-----|-----|
| var006_unrolled | 1677155 |
|-----|-----|
| var006_pipelined | 3192033 |
|-----|-----|
| var006_pipe_unroll | 609804 |
|-----|-----|
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 1.5 – Результаты работы приложения в режиме Emulation-SW

На рисунке 1.6 приведена копия экрана Assistant View для сборки Emulation-HW.

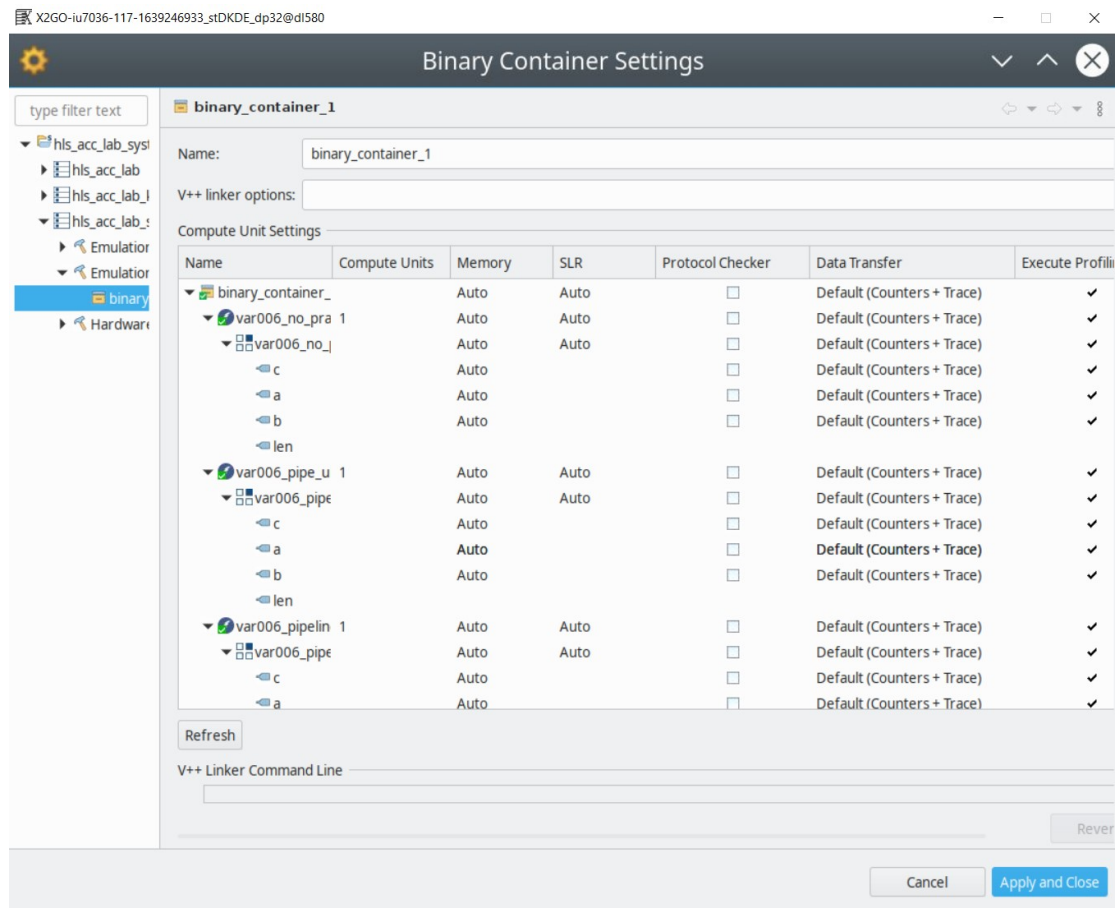


Рисунок 1.6 – Копия экрана Assistant View для сборки Emulation-HW

На рисунках 1.7-1.8 приведены результаты работы приложения в режиме Emulation-HW.

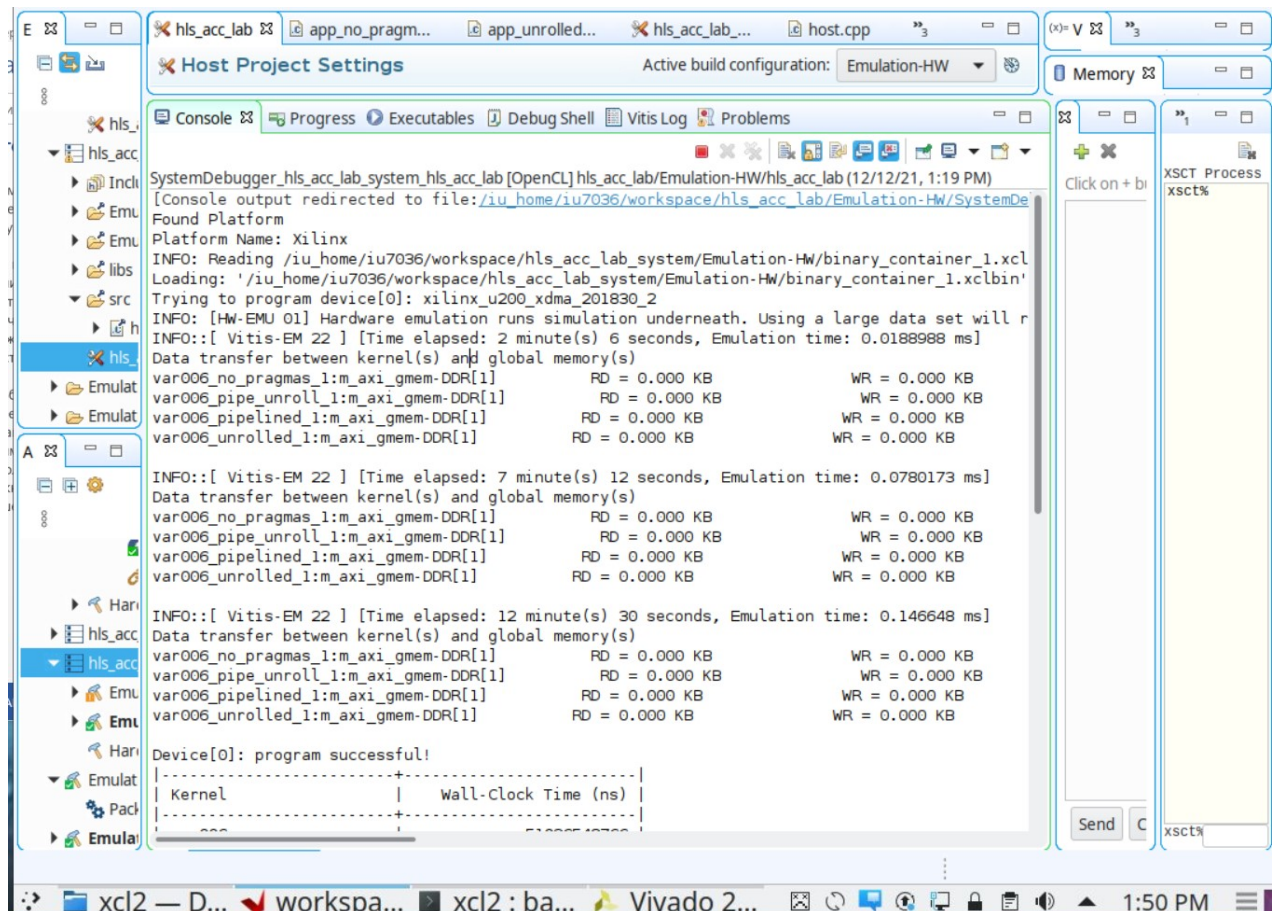


Рисунок 1.7 – Результаты работы приложения в режиме Emulation-HW (Начало)

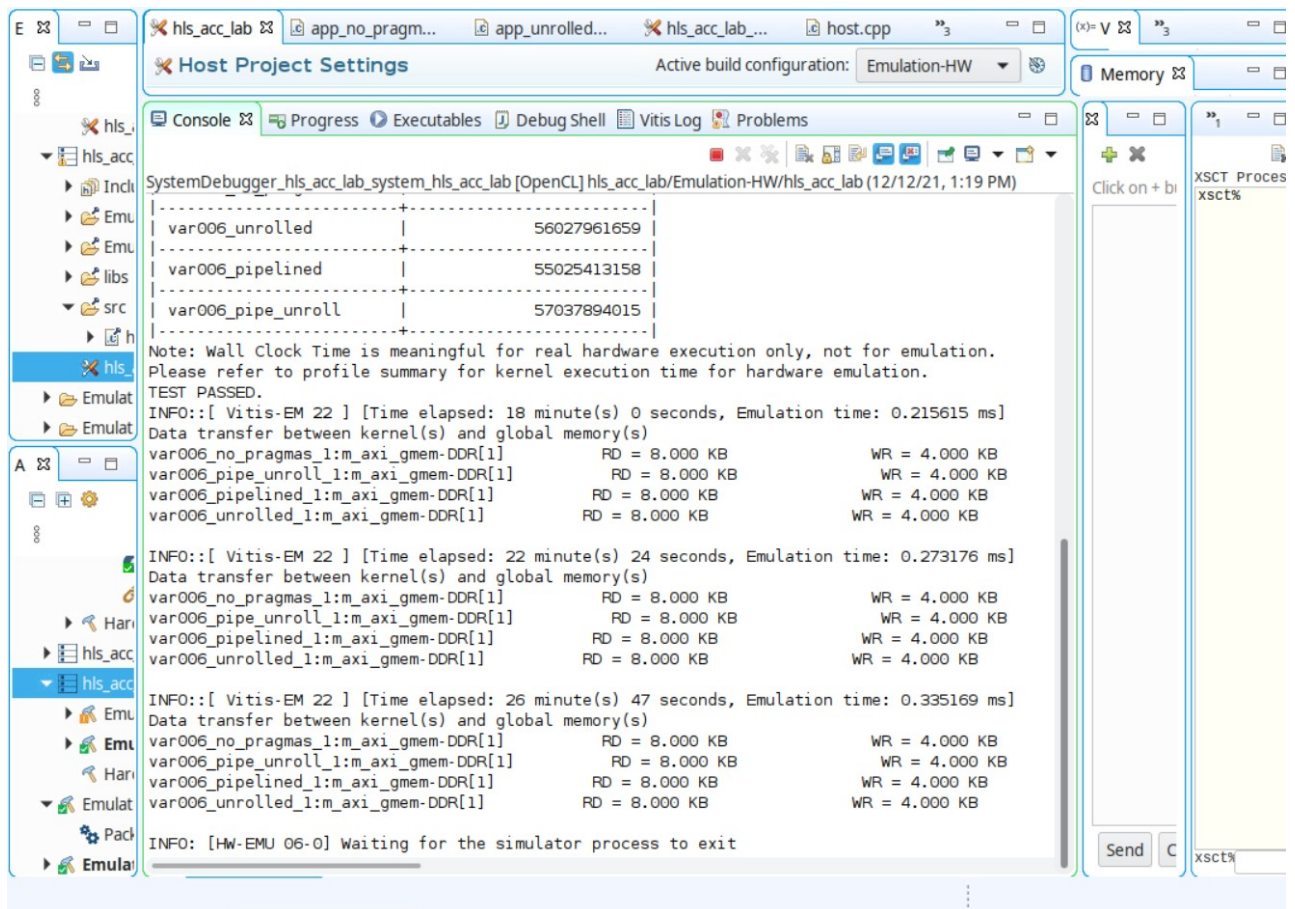


Рисунок 1.8 – Результаты работы приложения в режиме Emulation-HW
(Продолжение)

На рисунке 1.9 приведено окно внутрисхемного отладчика Vivado для сборки в режиме Emulation-HW.

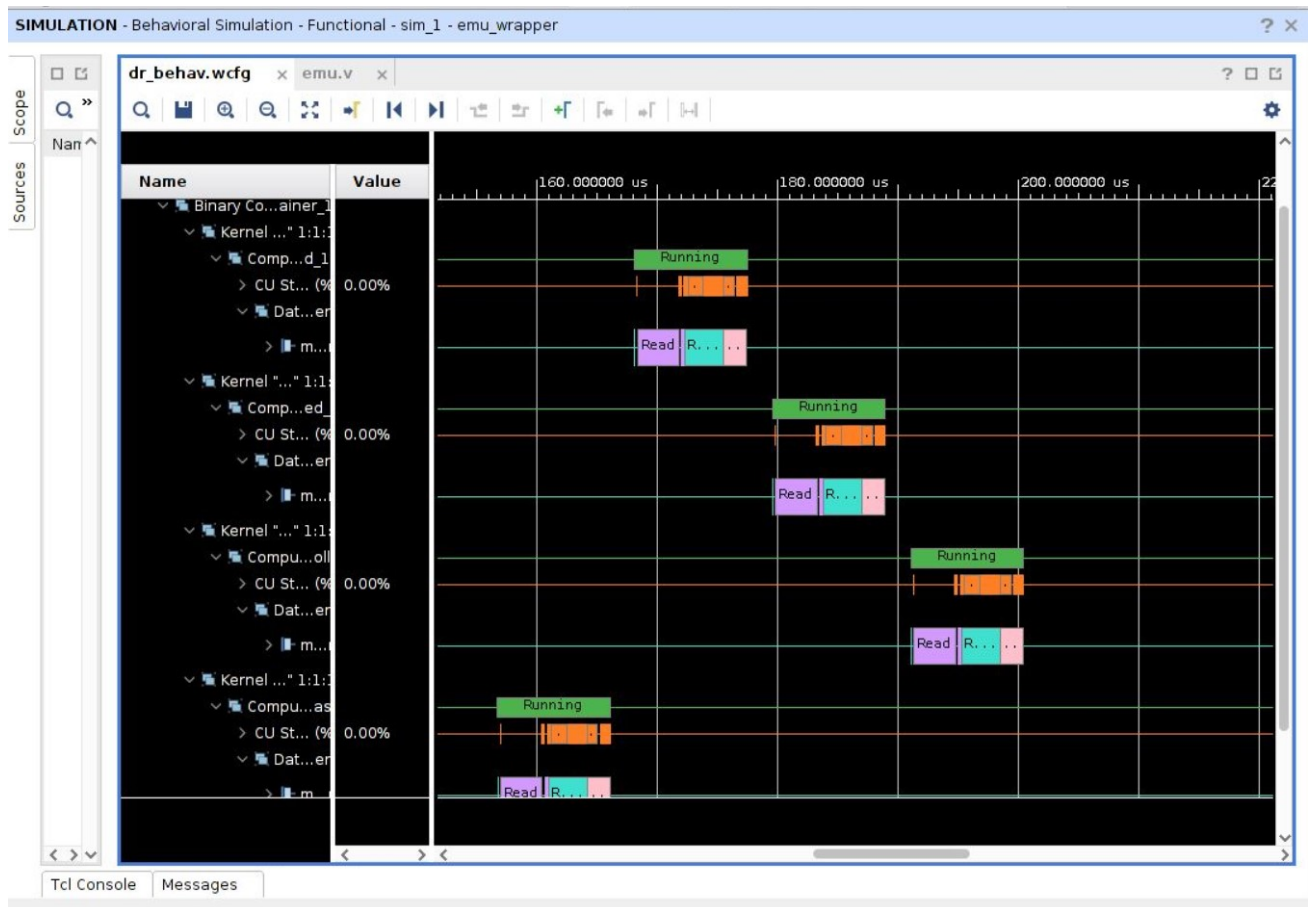
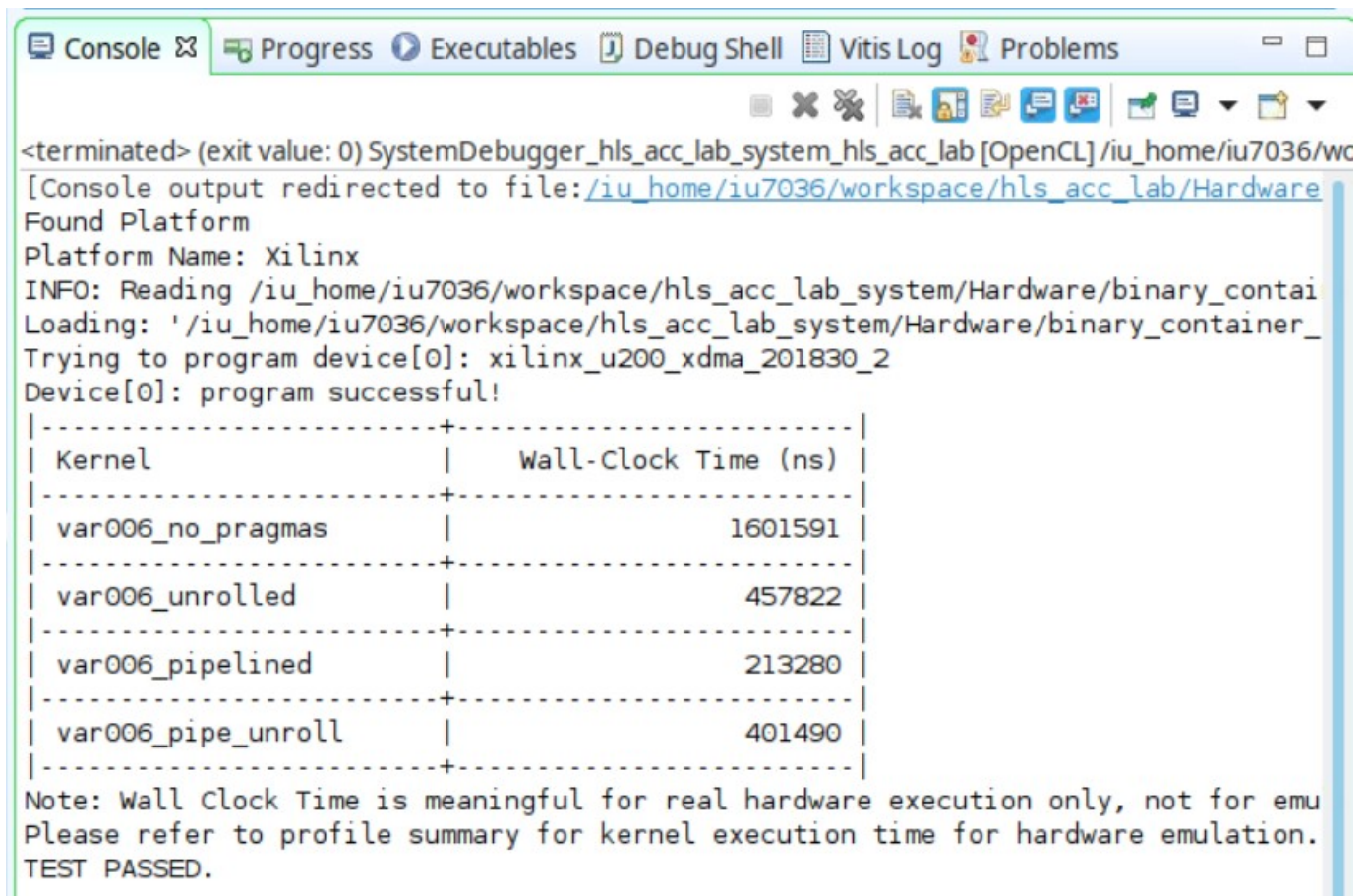


Рисунок 1.9 – Окно внутрисхемного отладчика Vivado для сборки в режиме Emulation-HW

На рисунке 1.10 приведены результаты работы приложения в режиме Hardware.



The screenshot shows the Vitis IDE console window with the following content:

```
<terminated> (exit value: 0) SystemDebugger_hls_acc_lab_system_hls_acc_lab [OpenCL] /iu_home/iu7036/wc
[Console output redirected to file:/iu_home/iu7036/workspace/hls_acc_lab/Hardware
Found Platform
Platform Name: Xilinx
INFO: Reading /iu_home/iu7036/workspace/hls_acc_lab_system/Hardware/binary_contai
Loading: '/iu_home/iu7036/workspace/hls_acc_lab_system/Hardware/binary_container_
Trying to program device[0]: xilinx_u200_xdma_201830_2
Device[0]: program successful!
|-----+-----|
| Kernel                | Wall-Clock Time (ns) |
|-----+-----|
| var006_no_pragmas     |          1601591     |
|-----+-----|
| var006_unrolled       |          457822      |
|-----+-----|
| var006_pipelined      |          213280      |
|-----+-----|
| var006_pipe_unroll    |          401490      |
|-----+-----|
Note: Wall Clock Time is meaningful for real hardware execution only, not for emu
Please refer to profile summary for kernel execution time for hardware emulation.
TEST PASSED.
```

Kernel	Wall-Clock Time (ns)
var006_no_pragmas	1601591
var006_unrolled	457822
var006_pipelined	213280
var006_pipe_unroll	401490

Рисунок 1.10 – Результаты работы приложения в режиме Hardware

На рисунке 1.11 приведена копия экрана для вкладки «Summary».

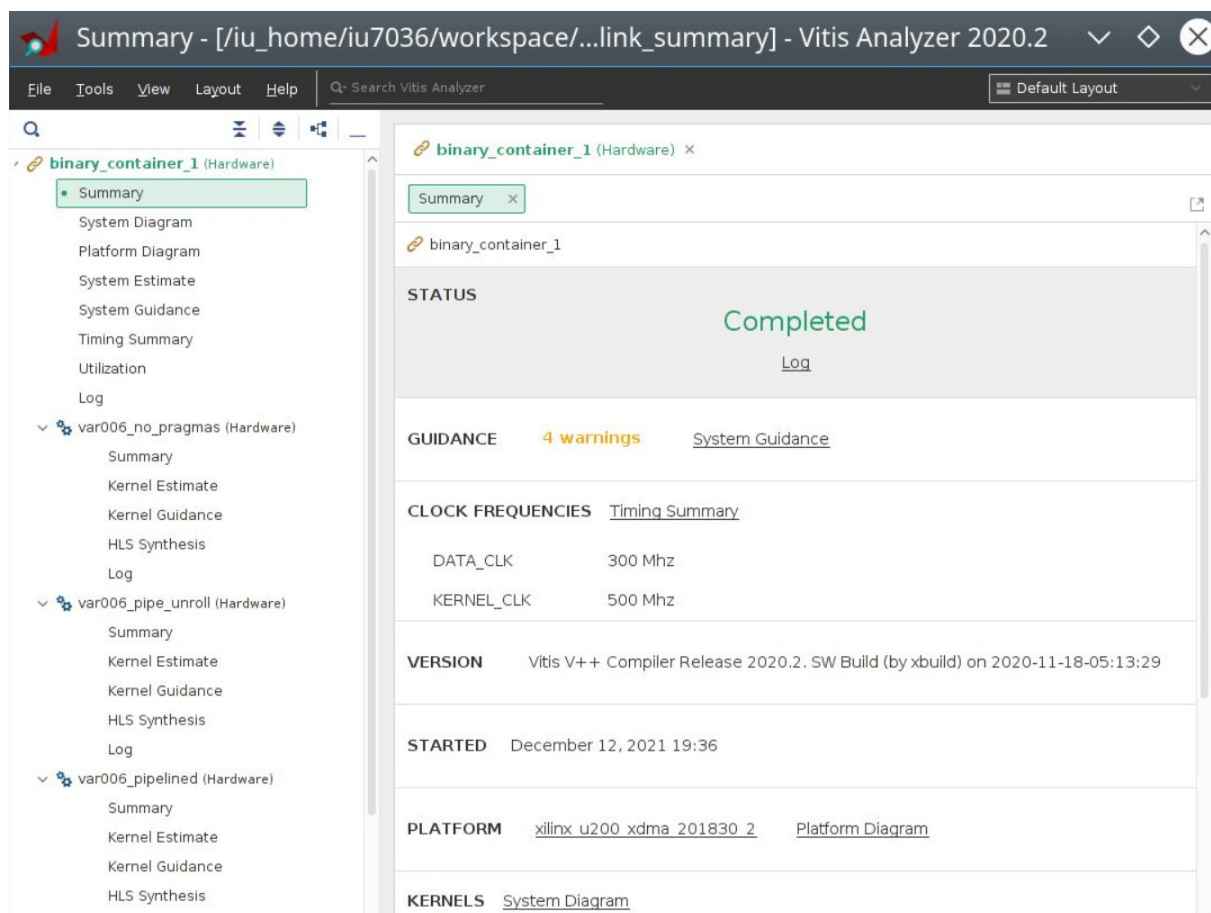


Рисунок 1.11 – Копия экрана для вкладки «Summary»

На рисунке 1.12 приведена копия экрана для вкладки «System Diagram».

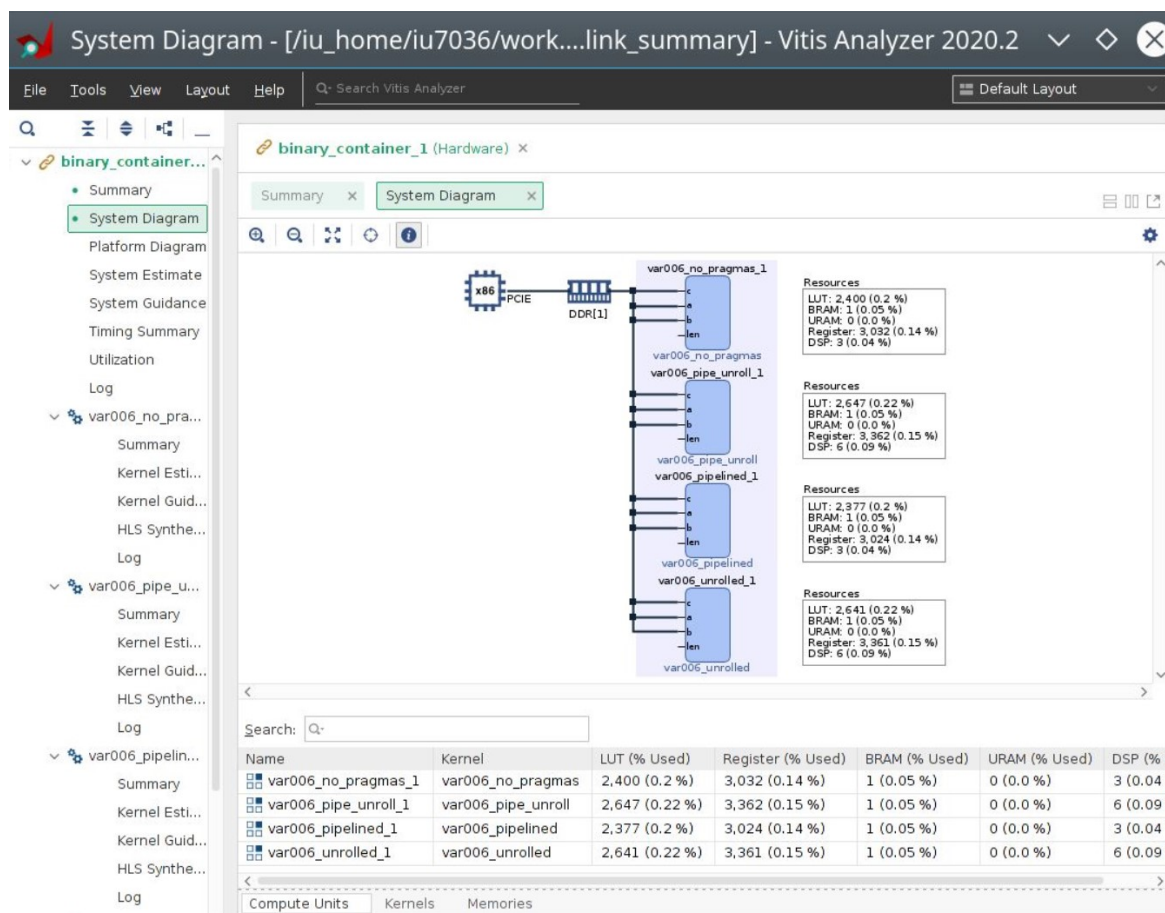


Рисунок 1.12 – Копия экрана для вкладки «System Diagram»

На рисунке 1.13 приведена копия экрана для вкладки «Platform Diagram».

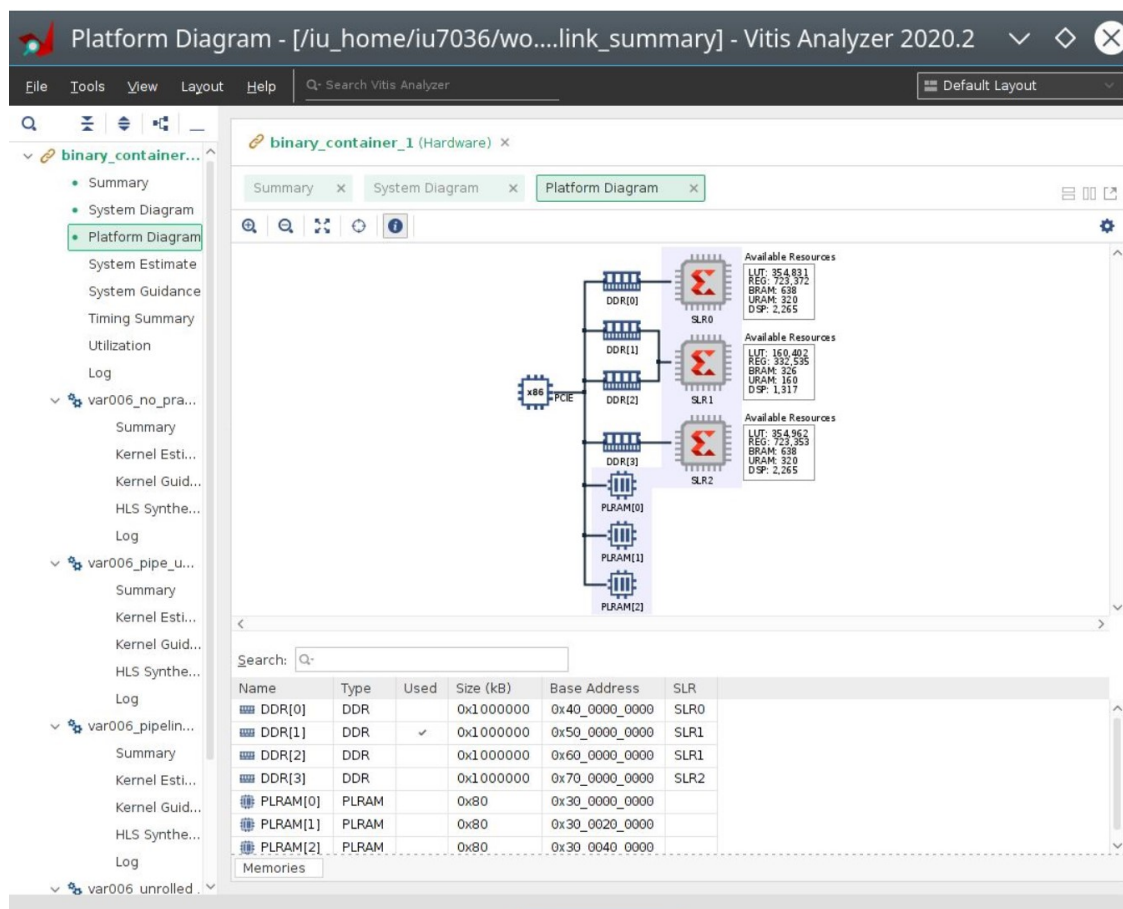
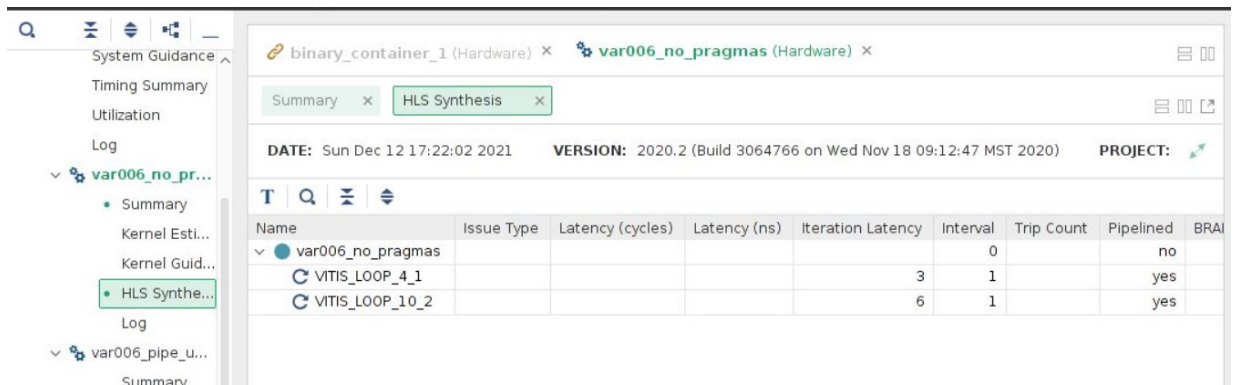


Рисунок 1.13 – Копия экрана для вкладки «Platform Diagram»

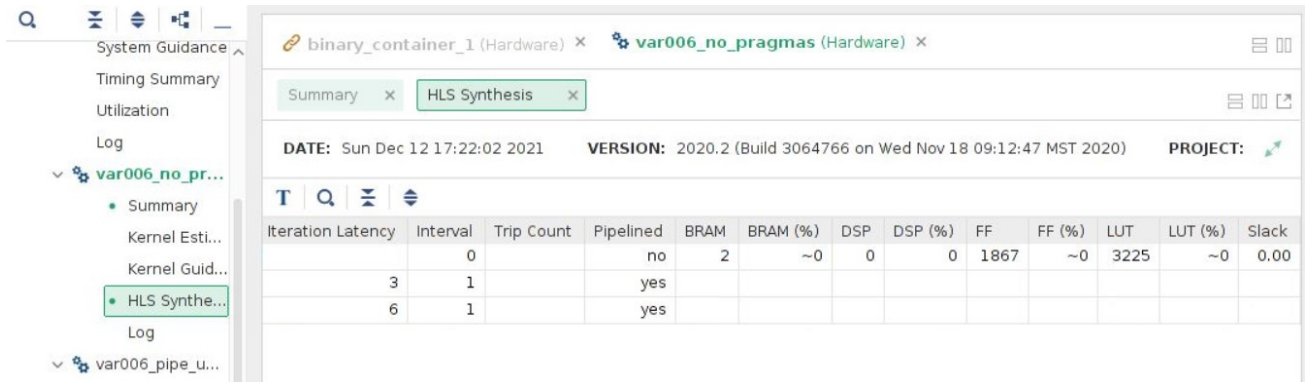
На рисунках 1.14-1.15 приведены копии экрана для вкладки «HLS Synthesis» для ядра сборки Hardware app_no_pragmas.



The screenshot shows the 'HLS Synthesis' tab selected in the left-hand navigation pane. The main window displays the synthesis report for the 'var006_no_pragmas' kernel. The report includes a table with the following data:

Name	Issue Type	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM
var006_no_pragmas					0		no	
VITIS_LOOP_4_1				3	1		yes	
VITIS_LOOP_10_2				6	1		yes	

Рисунок 1.14 – Копии экрана для вкладки «HLS Synthesis» для ядра сборки Hardware app_no_pragmas (Начало)



The screenshot shows the 'HLS Synthesis' tab selected in the left-hand navigation pane. The main window displays the synthesis report for the 'var006_no_pragmas' kernel. The report includes a table with the following data:

Iteration Latency	Interval	Trip Count	Pipelined	BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)	Slack
	0		no	2	~0	0	0	1867	~0	3225	~0	0.00
3	1		yes									
6	1		yes									

Рисунок 1.15 – Копии экрана для вкладки «HLS Synthesis» для ядра сборки Hardware app_no_pragmas (Продолжение)

На рисунках 1.16-1.17 приведены копии экрана для вкладки «HLS Synthesis» для ядра сборки Hardware app_pipelined.

Name	Issue Type	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM
var006_pipe_unroll	II Violation				0		no	2
VITIS_LOOP_4_1	II Violation			4	2		yes	
VITIS_LOOP_12_2	II Violation			7	2		yes	

Рисунок 1.16 – Копии экрана для вкладки «HLS Synthesis» для ядра сборки Hardware app_pipelined (Начало)

Iteration Latency	Interval	Trip Count	Pipelined	BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)	Slack
	0		no	2	~0	0	0	2453	~0	3744	~0	0.00
4	2		yes									
7	2		yes									

Рисунок 1.17 – Копии экрана для вкладки «HLS Synthesis» для ядра сборки Hardware app_pipelined (Продолжение)

На рисунках 1.20-1.21 приведены копии экрана для вкладки «HLS Synthesis» для ядра сборки Hardware app_unrolled.

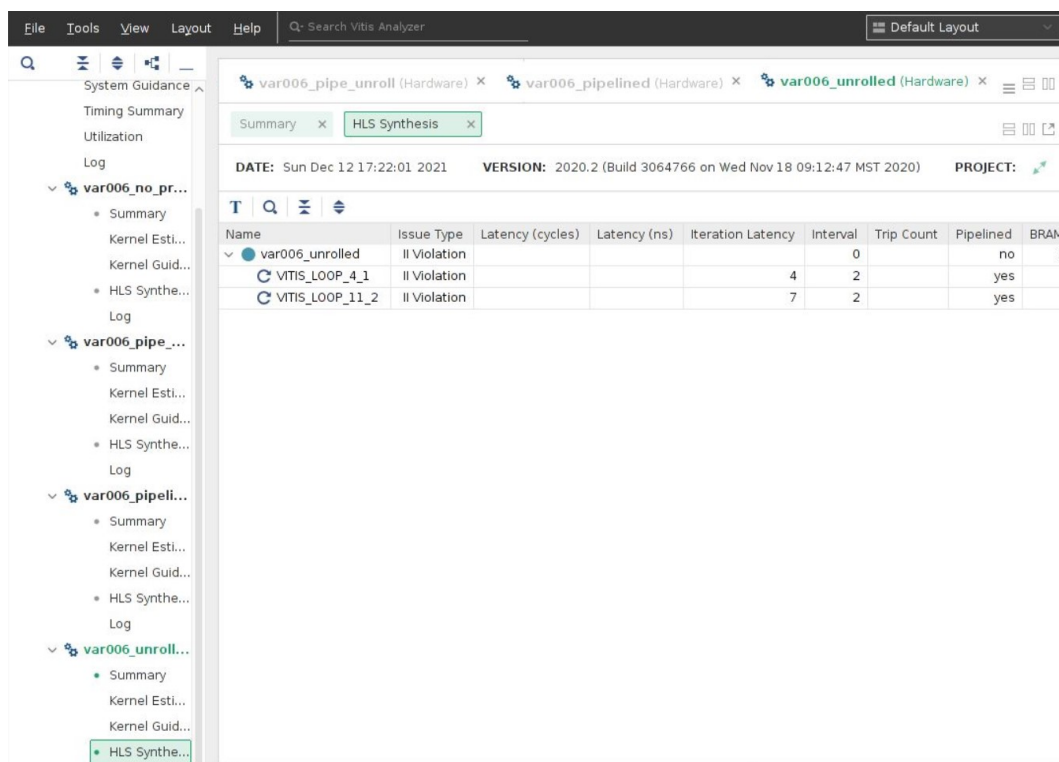


Рисунок 1.18 – Копии экрана для вкладки «HLS Synthesis» для ядра сборки Hardware app_unrolled (Начало)

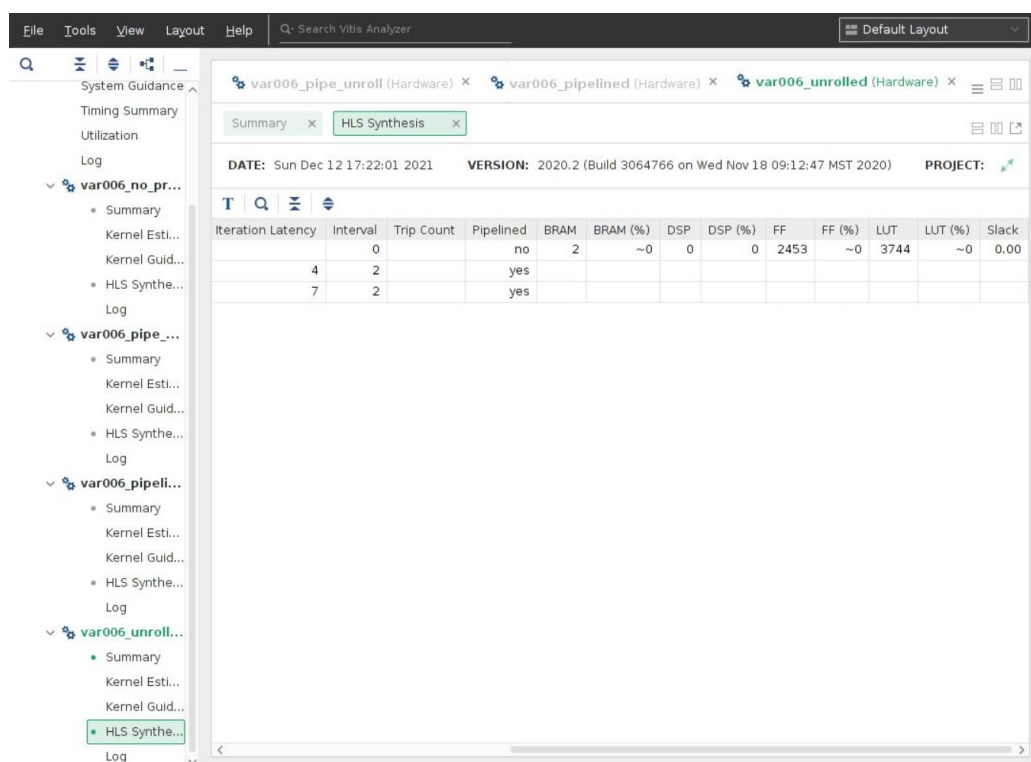


Рисунок 1.19 – Копии экрана для вкладки «HLS Synthesis» для ядра сборки Hardware app_unrolled (Продолжение)

На рисунках 1.18-1.19 приведены копии экрана для вкладки «HLS Synthesis» для ядра сборки Hardware app_pipe_unroll.

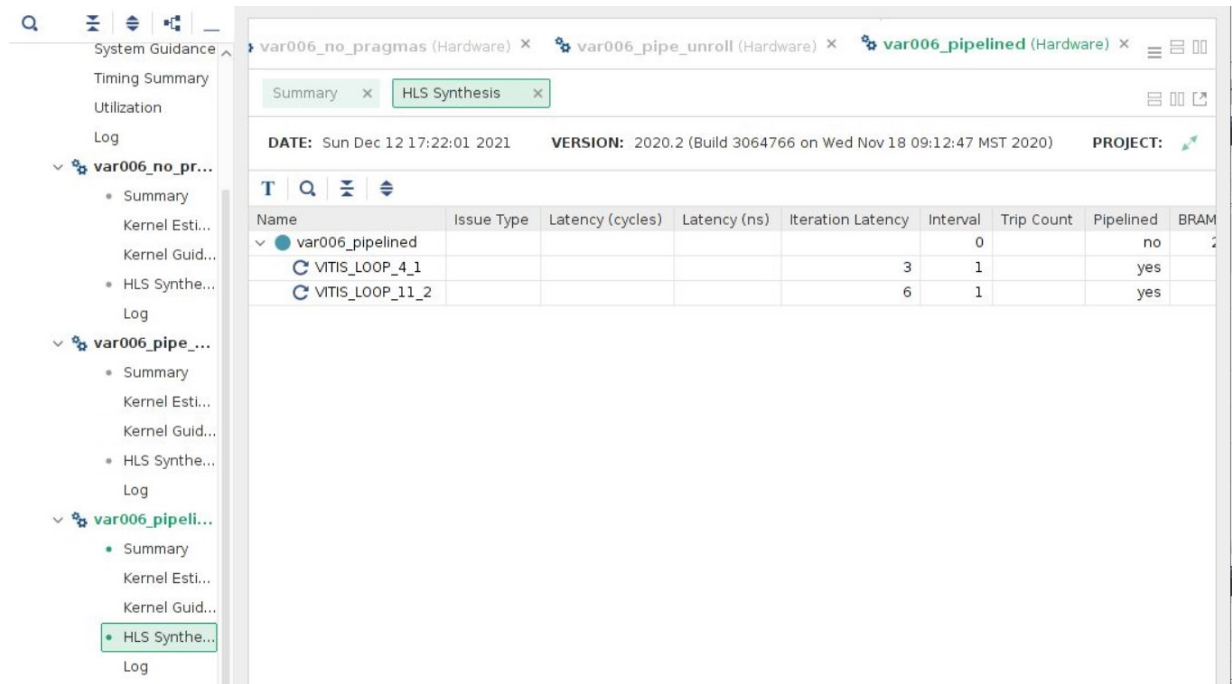


Рисунок 1.20 – Копии экрана для вкладки «HLS Synthesis» для ядра сборки Hardware app_pipe_unroll (Начало)

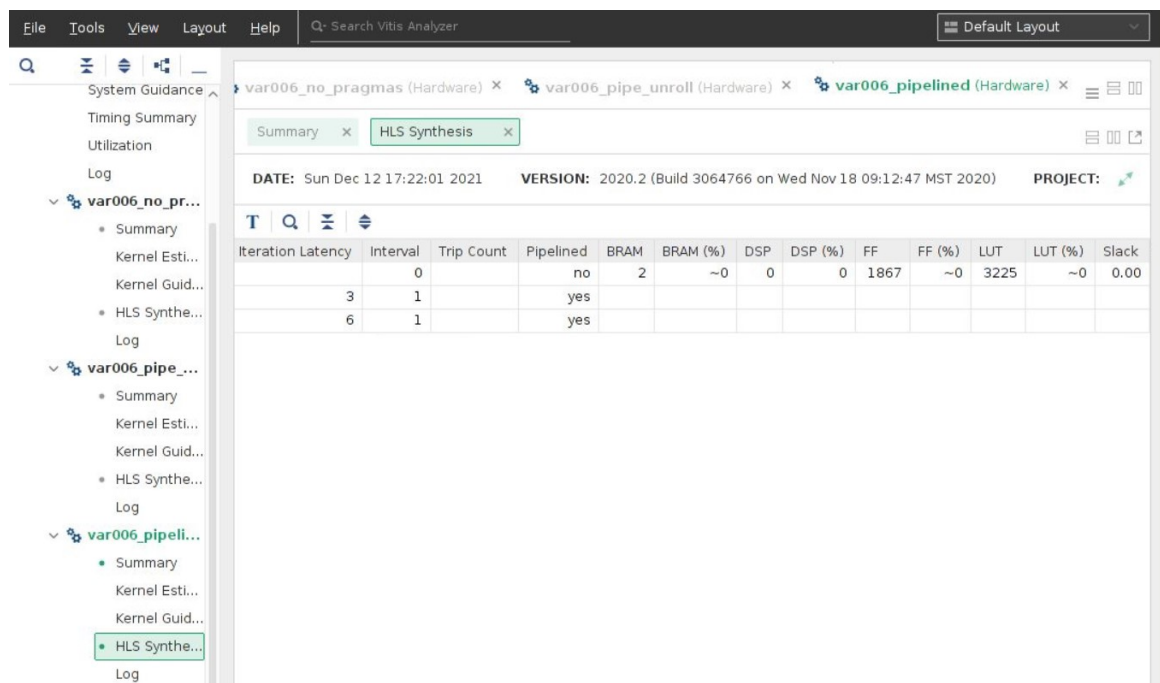


Рисунок 1.21 – Копии экрана для вкладки «HLS Synthesis» для ядра сборки Hardware app_pipe_unroll (Продолжение)

Из рисунка 1.10 можно сделать вывод о том, что наибольшее время выполнения, как и ожидалось, у цикла без оптимизаций.

Далее идет частично развернутый цикл, так как в нем все развернутые итерации выполняются параллельно и количество итераций уменьшается.

Наименьшее время выполнения было достигнуто при конвейерной обработке цикла, так как внутри цикла не оказалось зависимости по данным и цикл имел возможность начинать последующие итерации менее чем за три такта.

Одновременное применение конвейеризации и частичного развертывания тела цикла позволило ускорить обработку по сравнению с применением только развертывания тела, однако уступило по времени организации, при которой использовалась только конвейерная организация. Это, вероятно, вызвано неудачно подобранными параметрами развертывания, что вызвало замедление загрузки данных из памяти.

Заключение

В результате выполнения работы были изучены методики и технологии синтеза аппаратных устройств ускорения вычислений по описаниям на языках высокого уровня.

Был рассмотрен маршрут проектирования устройств, представленных в виде синтаксических конструкций ЯВУ C/C++, изучены принципы работы IDE Xilinx Vitis HLS и методика анализа и отладки устройств.

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