ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»(ИУ7)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.04 «Программная инженерия»

О Т Ч Е Т

по лабораторной работе № 4

Название Методология разработки и верификации ускорителей вычислений

на платформе Xilinx Alveo

Дисциплина Архитектура элекронно-вычислительных машин

Студент:	_____	Козлова И. В.
	подпись, дата	Фамилия, И.О.
Преподаватель:	_____	Попов А. Ю.
	подпись, дата	Фамилия, И. О.

Москва — 2021 г.

Оглавление

Цель работы

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

Для достижения поставленной цели необходимо выполнить следующие задачи:

- изучить основные сведения о платформе Xilinx Alveo U200;

- разработать RTL (Register Transfer Language, язык регистровых передач) описание ускорителя вычислений по индивидуальному варианту;

- выполнить генерацию ядра ускорителя;

- выполнить синтез и сборку бинарного модуля ускорителя;

- разработать и отладить тестирующее программное обеспечение на серверной хост-платформе;

- провести тесты работы ускорителя вычислений.

Основные теоретические сведения

В ходе лабораторной работы будет использован базовый шаблон так называемого RTL проекта VINC, который может быть создан в IDE Xilinx Vitis и САПР Xilinx Vivado. Шаблон VINC выполняет попарное сложение чисел исходного массива и сохраняет результаты во втором массиве. Проект VINC включает:

Проект ПО хоста, выполняющий инициализацию аппаратного ядра и его тестирование через OpenCL вызовы.

Синтезируемый RTL проект ядра ускорителя на языках Verilog и SystemVerilog.

Функциональный тест ускорителя VINC на языке SystemVerilog.

Все перечисленные проекты создаются автоматически посредством запуска мастера RTL проектов в IDE Xilinx Vitis, и могут далее модифицироваться как через тот же мастер, так и в ручном режиме в САПР Xilinx Vivado, или обычном текстовом редакторе. Ряд проектных процедур необходимо запустить из консоли ОС Linux.

Проект VINC представляет собой аппаратное устройство, связанное шиной AXI4 MM (Memory mapped) с DDR[i] памятью, и получающее настроечные параметры по интерфейсу AXI4 Lite от программного обеспечения хоста (см. рисунок 1). В рамках всей системы используется единое 64-х разрядное адресное пространство, в котором формируются адреса на всех AXI4 шинах.
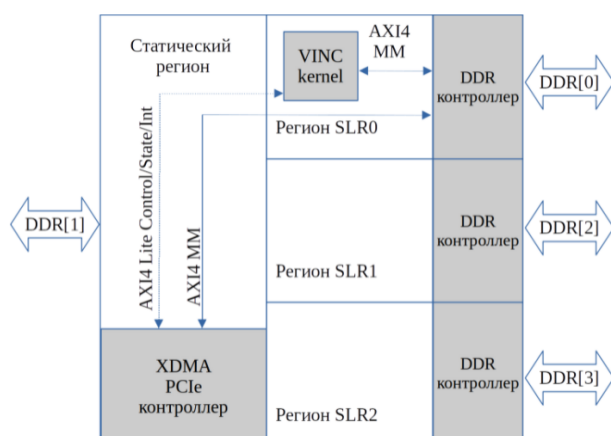
Рисунок 1 – Размещение проекта на ПЛИС xcu200-fsgd2104-2-е карты Alveo U200

В каждой карте U200 имеется возможность подключить ускоритель к

любому DDR[i] контроллеру в том регионе, где будет размещен проект. Всего для пользователя доступны 3 динамических региона: SLR0,1,2, для которых выделены каналы локальной памяти DDR[0], DDR[2], DDR[3] соответственно. Вся подключенная память DDR[0..3] доступна со стороны статического региона, в котором размещена аппаратная часть XRT.

Память DDR[1] доступна для использования как в статическом регионе, так и в динамическом регионе SLR1.

Предполагается, что эта память может служить для организации эффективной подсистемы памяти ускорительной карты: буферизации данных, передаваемых между хост-системой и ускорителем.

Копии экранов моделирования исходного проекта VINC (исходная программа)

По умолчанию, в диаграмму (которую необходимо получить в приложении Vivado) добавлены сигналы шины AXI4 MM, представляющие собой 5 независимых каналов передачи сообщений, которые представлены в таблице 1.

Таблица 1 – Результаты замеров времени.

Канал передачи	Группы сигналов
Канал чтения адреса от ведущего к ведомому	m00_axi_ar*
Канал чтения данных от ведомого к ведущему	m00_axi_r*
Канал записи адреса записи от ведущего к ведомому	m00_axi_aw*
Канал запись данных от ведущего к ведомому	m00_axi_w*
Канал записи ответа от ведомого к ведущему	m00_axi_b*

Каналы позволяют сформировать конвейерные транзакции чтения и записи. Последовательность событий транзакции чтения можно представить следующим образом: ARVALID→ ARREADY→ RVALID→ RREADY.

Последовательность событий транзакции записи: AWVALID→ AWREADY → WVALID → WREADY → BVALID → BREADY.

На рисунке 2 приведена транзакция чтения данных вектора на шине AXI4 MM из DDR памяти.

Рисунок 2 – Транзакция чтения данных вектора на шине AXI4 MM из DDR памяти

На рисунке 3 приведена транзакция записи результата инкремента данных на шине AXI4 MM.

Рисунок 3 – Транзакция записи результата инкремента данных на шине AXI4 MM

На рисунке 4 приведен фрагмент кода модуля rtl_kernel_wizard_0_example_a с выполнением инкрементирования данных.

```
// Adder function
always @(posedge s_axis_aclk) begin
  for (i = 0; i < LP_NUM_LOOPS; i = i + 1) begin
    d2_tdata[i*C_ADDER_BIT_WIDTH+:C_ADDER_BIT_WIDTH] <= d1_tdata[C_ADDER_BIT_WIDTH*i+:C_ADDER_BIT_WIDTH] + d1_constant;
  end
end
```

Рисунок 4 – Код модуля rtl_kernel_wizard_0_example_adder.v с выполнением инкрементирования данных

Копии экранов моделирования исходного проекта VINC (измененная программа)

В соответствии с вариантом 9 необходимо было изменить код проекта. Реализовать функцию 1

$$R[i] = min(A[i] - 4, 5) \tag{1}$$

На рисунке 5 приведен код измененнной программы.


```
// Adder function
always @(posedge s_axis_aclk) begin
  for (i = 0; i < LP_NUM_LOOPS; i = i + 1) begin
    if (d1_tdata[C_ADDER_BIT_WIDTH*i+:C_ADDER_BIT_WIDTH] - USER_C1 < USER_C2)
        d2_tdata[i*C_ADDER_BIT_WIDTH+:C_ADDER_BIT_WIDTH] <= d1_tdata[C_ADDER_BIT_WIDTH*i+:C_ADDER_BIT_WIDTH] - USER_C1;
    else
        d2_tdata[i*C_ADDER_BIT_WIDTH+:C_ADDER_BIT_WIDTH] <= USER_C2;
  end
end
```

Рисунок 5 – Измененный код модуля
rtl_kernel_wizard_0_example_adder.v

Константы, которые используются в данном коде, представлены на рисунке 6.


```
7  // default_nettype of none prevents implicit wire declarati
8  `default_nettype none
9  `timescale 1ps / 1ps
10
11 module rtl_kernel_wizard_0_example_adder #(
12   parameter integer C_AXIS_TDATA_WIDTH = 512, // Data width
13   parameter integer C_ADDER_BIT_WIDTH  = 32,
14   parameter integer C_NUM_CLOCKS       = 1,
15   parameter integer USER_C1            = 4,
16   parameter integer USER_C2            = 5
17 )
18 (
19
```

Рисунок 6 – Константы

На рисунке 7 приведена транзакция чтения данных вектора на шине AXI4 MM из DDR памяти.
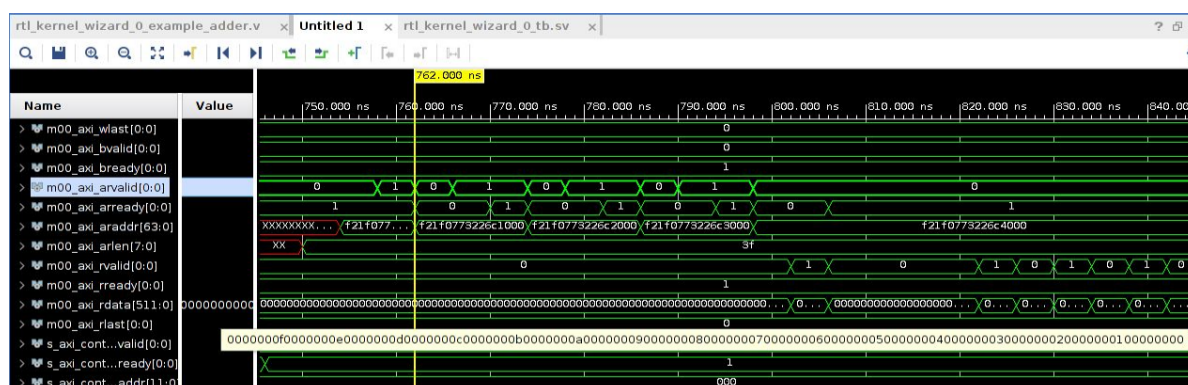
Рисунок 7 – Транзакция чтения данных вектора на шине AXI4 MM из
DDR памяти

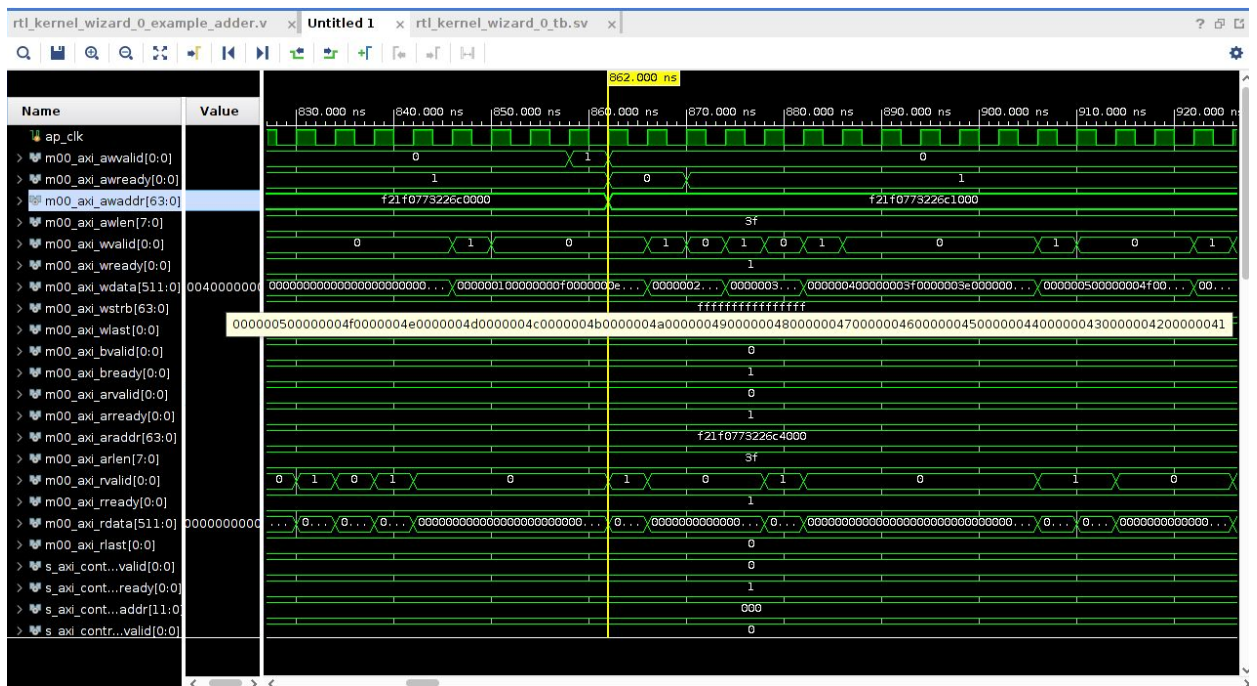На рисунке 8 приведена транзакция записи результата инкремента данных на шине AXI4 MM.

Рисунок 8 – Транзакция записи результата инкремента данных на шине
AXI4 MM

Сборка проекта

Для сборки проекта необходимо было написать конфигурационный файл.

В конфигурационом файле указывается основная информация для работы компилятора v++:

1. Количество и условные имена экземпляров ядер.

2. Тактовая частота работы ядра.

3. Для каждого ядра: выбор области SLR (SLR[0..2]), выбор DDR (DDR[0..3]) памяти, выбор высокопроизводительной памяти PLRAM(PLRAM[0,1,2]).

4. Параметры синтеза и оптимизации проекта.

На рисунке 9 представлен конфигурационный файл для сборки проекта.

```
[connectivity]
nk=rtl_kernel_wizard_0:1:vinc0
slr=vinc0:SLR0
sp=vinc0.m00_axi:DDR[0]

[vivado]
prop=run.impl_1.STEPS.OPT_DESIGN.ARGS.DIRECTIVE=Explore
prop=run.impl_1.STEPS.PLACE_DESIGN.ARGS.DIRECTIVE=Explore
prop=run.impl_1.STEPS.PHYS_OPT_DESIGN.IS_ENABLED=true
prop=run.impl_1.STEPS.PHYS_OPT_DESIGN.ARGS.DIRECTIVE=AggressiveExplore
prop=run.impl_1.STEPS.ROUTE_DESIGN.ARGS.DIRECTIVE=Explore
```

Рисунок 9 – Конфигурационный файл

Содержимое файлов v++*.log и *.xclbin.info. приведено в приложениях.

Тестирование

Для того, чтобы запустить тесты, необходимо изменить условие проверки в автоматически созданном программном модуле host_example.cpp.

Часть кода модуля host_example.cpp приведена на рисунке 10. Было изменено условие проверки, на проверку, соответствующую моему варианту.

```cpp
for (cl_uint i = 0; i < number_of_words; i++) {
    //printf("h_data[i] %d, h_output[i] %d", h_data[i], h_axi00_ptr0_output[i]);
    if (h_data[i] - 4 > 5) {
        h_data[i] = 5;
    } else {
        h_data[i] -= 4;
    }
    //printf("h_data[i] %d, h_output[i] %d", h_data[i], h_axi00_ptr0_output[i]);
    if (h_data[i] != h_axi00_ptr0_output[i]) {
        printf("ERROR in rtl_kernel_wizard_0::m00_axi - array index %d (host addr 0x%03x)"
            " - input=%d (0x%x), output=%d (0x%x)\n", i, i*4, h_data[i], h_data[i],
        h_axi00_ptr0_output[i], h_axi00_ptr0_output[i]);
        check_status = 1;
    }
    //  printf("i=%d, input=%d, output=%d\n", i,  h_axi00_ptr0_input[i],
    h_axi00_ptr0_output[i]);
}
```

Рисунок 10 – host_example.cpp

Тесты запускались с помощью утилиты xgdb. Результаты тестирования приведены на рисунке 11.

Рисунок 11 – Тестирование

По результатам можно увидеть, что все тесты выполнены успешно.

Контрольные вопросы

1. Назовите преимущества и недостатки XDMA и QDMA платформ.

Преимущества QDMA:

- позволяет передавать поток данных непосредственно в логику FPGA параллельно с их обработкой.

- предоставляет разработчикам прямое потоковое соединение с низкой задержкой между хостом и ядрами.

- включает высокопроизводительный DMA, который использует несколько очередей, оптимизированных как для передачи данных с высокой пропускной способностью, так и для передачи данных с большим количеством пакетов.

Недостатки XDMA:

- требует, чтобы данные сначала были полностью перемещены из памяти хоста в память FPGA (DDRx4 DIMM или PLRAM), прежде чем логика FPGA сможет начать обработку данных, что влияет на задержку на запуска задачи.

2. Назовите последовательность действий, необходимых для инициализации ускорителя со стороны хост-системы.

1. Хост получает все платформы.

2. Хост выбирает имя платформы Xilinx.

3. Хост получает Id устройства.

4. Хост получает информацию об устройстве.

5. Создается контекст для переменных.

6. Создается команда для устройста-ускорителя.

3. Какова процедура запуска задания на исполнения в ускорительном ядре VINC.

1. Данные из .xclbin копируются из ОЗУ в локальную память ускорителя посредством DMA.

2. В памяти устройства-ускорителя создается исполняемый файл.

3. Те данные, которые подлежат обработке, копируются из ОЗУ в локальную память усокрителя посредством DMA.

4. Указываются необходимые параметры и запускается программа на ускорителе.

5. В конце выполняется чтение готовых данных.

4. Опишите процесс линковки на основании содержимого файла v++_*.log.

1. Анализ профиля устройства. Анализ конфигурационного файла. Поиск необходимых интерфейсов.

2. FPGA linking synthesized kernels to platform

3. FPGA logic optimization (оптимизация логики ПЛИС) для минимизации задержки.

4. FPGA logic placement (размещение логики ПЛИС, то есть выбор конкретного мета для определенного логического блока).

5. FPGA routing (маршрутизация ПЛИС)

6. FPGA bitstream generation (генерация битового потока ПЛИС, то есть генерация файла [*.xclbin]).

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.

Приложение 1

Листинг 1 – Содержимое файла host_example.cpp

```
 1  // This is a generated file. Use and modify at your own risk.
 2  ////////////////////////////////////////////////////////////////////////////////
 3
 4  /********************************************************************************
 5  Vendor: Xilinx
 6  Associated Filename: main.c
 7  #Purpose: This example shows a basic vector add +1 (constant) by manipulating
 8  #         memory inplace.
 9  ********************************************************************************/
10
11  #include <fcntl.h>
12  #include <stdio.h>
13  #include <iostream>
14  #include <stdlib.h>
15  #include <string.h>
16  #include <math.h>
17  #ifdef _WINDOWS
18  #include <io.h>
19  #else
20  #include <unistd.h>
21  #include <sys/time.h>
22  #endif
23  #include <assert.h>
24  #include <stdbool.h>
25  #include <sys/types.h>
26  #include <sys/stat.h>
27  #include <CL/opencl.h>
28  #include <CL/cl_ext.h>
29  #include "xclhal2.h"
30
31  ////////////////////////////////////////////////////////////////////////////////
32
33  #define NUM_WORKGROUPS (1)
34  #define WORKGROUP_SIZE (256)
35  #define MAX_LENGTH 8192
36  #define MEM_ALIGNMENT 4096
37  #if defined(VITIS_PLATFORM) && !defined(TARGET_DEVICE)
38  #define STR_VALUE(arg)      #arg
39  #define GET_STRING(name) STR_VALUE(name)
40  #define TARGET_DEVICE GET_STRING(VITIS_PLATFORM)
41  #endif
42
43  ////////////////////////////////////////////////////////////////////////////////
44
45  cl_uint load_file_to_memory(const char *filename, char **result)
46  {
47      cl_uint size = 0;
48      FILE *f = fopen(filename, "rb");
49      if (f == NULL) {
50          *result = NULL;
51          return -1; // -1 means file opening fail
52      }
53      fseek(f, 0, SEEK_END);
54      size = ftell(f);
55      fseek(f, 0, SEEK_SET);
56      *result = (char *)malloc(size+1);
57      if (size != fread(*result, sizeof(char), size, f)) {
58          free(*result);
59          return -2; // -2 means file reading fail
60      }
61      fclose(f);
62      (*result)[size] = 0;
63      return size;
64  }
65
66  int main(int argc, char** argv)
67  {
68
69      cl_int err;                             // error code returned from api calls
70      cl_uint check_status = 0;
71      const cl_uint number_of_words = 4096; // 16KB of data
72
73
74      cl_platform_id platform_id;         // platform id
75      cl_device_id device_id;             // compute device id
```

```
76        cl_context context;                  // compute context
77        cl_command_queue commands;           // compute command queue
78        cl_program program;                  // compute programs
79        cl_kernel kernel;                    // compute kernel
80
81        cl_uint* h_data;                                 // host memory for input vector
82        char cl_platform_vendor[1001];
83        char target_device_name[1001] = TARGET_DEVICE;
84
85        cl_uint* h_axi00_ptr0_output = (cl_uint*)aligned_alloc(MEM_ALIGNMENT,MAX_LENGTH * sizeof(cl_uint*));
                 // host memory for output vector
86        cl_mem d_axi00_ptr0;                             // device memory used for a vector
87
88        if (argc != 2) {
89            printf("Usage: %s xclbin\n", argv[0]);
90            return EXIT_FAILURE;
91        }
92
93        // Fill our data sets with pattern
94        h_data = (cl_uint*)aligned_alloc(MEM_ALIGNMENT,MAX_LENGTH * sizeof(cl_uint*));
95        for(cl_uint i = 0; i < MAX_LENGTH; i++) {
96            h_data[i]    = i;
97            h_axi00_ptr0_output[i] = 0;
98
99        }
100
101        // Get all platforms and then select Xilinx platform
102        cl_platform_id platforms[16];        // platform id
103        cl_uint platform_count;
104        cl_uint platform_found = 0;
105        err = clGetPlatformIDs(16, platforms, &platform_count);
106        if (err != CL_SUCCESS) {
107            printf("ERROR: Failed to find an OpenCL platform!\n");
108            printf("ERROR: Test failed\n");
109            return EXIT_FAILURE;
110        }
111        printf("INFO: Found %d platforms\n", platform_count);
112
113        // Find Xilinx Plaftorm
114        for (cl_uint iplat=0; iplat<platform_count; iplat++) {
115            err = clGetPlatformInfo(platforms[iplat], CL_PLATFORM_VENDOR, 1000, (void *)cl_platform_vendor,
                     NULL);
116            if (err != CL_SUCCESS) {
117                printf("ERROR: clGetPlatformInfo(CL_PLATFORM_VENDOR) failed!\n");
118                printf("ERROR: Test failed\n");
119                return EXIT_FAILURE;
120            }
121            if (strcmp(cl_platform_vendor, "Xilinx") == 0) {
122                printf("INFO: Selected platform %d from %s\n", iplat, cl_platform_vendor);
123                platform_id = platforms[iplat];
124                platform_found = 1;
125            }
126        }
127        if (!platform_found) {
128            printf("ERROR: Platform Xilinx not found. Exit.\n");
129            return EXIT_FAILURE;
130        }
131
132        // Get Accelerator compute device
133        cl_uint num_devices;
134        cl_uint device_found = 0;
135        cl_device_id devices[16];   // compute device id
136        char cl_device_name[1001];
137        err = clGetDeviceIDs(platform_id, CL_DEVICE_TYPE_ACCELERATOR, 16, devices, &num_devices);
138        printf("INFO: Found %d devices\n", num_devices);
139        if (err != CL_SUCCESS) {
140            printf("ERROR: Failed to create a device group!\n");
141            printf("ERROR: Test failed\n");
142            return -1;
143        }
144
145        //iterate all devices to select the target device.
146        for (cl_uint i=0; i<num_devices; i++) {
147            err = clGetDeviceInfo(devices[i], CL_DEVICE_NAME, 1024, cl_device_name, 0);
148            if (err != CL_SUCCESS) {
149                printf("ERROR: Failed to get device name for device %d!\n", i);
150                printf("ERROR: Test failed\n");
151                return EXIT_FAILURE;
152            }
153            printf("CL_DEVICE_NAME %s\n", cl_device_name);
154            if(strcmp(cl_device_name, target_device_name) == 0) {
155                device_id = devices[i];
```

```
156              device_found = 1;
157              printf("Selected_%s_as_the_target_device\n", cl_device_name);
158          }
159      }
160
161      if (!device_found) {
162          printf("ERROR:Target_device_%s_not_found._Exit.\n", target_device_name);
163          return EXIT_FAILURE;
164      }
165
166      // Create a compute context
167      //
168      context = clCreateContext(0, 1, &device_id, NULL, NULL, &err);
169      if (!context) {
170          printf("ERROR:_Failed_to_create_a_compute_context!\n");
171          printf("ERROR:_Test_failed\n");
172          return EXIT_FAILURE;
173      }
174
175      // Create a command commands
176      commands = clCreateCommandQueue(context, device_id, CL_QUEUE_PROFILING_ENABLE |
              CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE, &err);
177      if (!commands) {
178          printf("ERROR:_Failed_to_create_a_command_commands!\n");
179          printf("ERROR:_code_%i\n",err);
180          printf("ERROR:_Test_failed\n");
181          return EXIT_FAILURE;
182      }
183
184      cl_int status;
185
186      // Create Program Objects
187      // Load binary from disk
188      unsigned char *kernelbinary;
189      char *xclbin = argv[1];
190
191      //————————————————————————————————————————————————————————
192      // xclbin
193      //————————————————————————————————————————————————————————
194      printf("INFO:_loading_xclbin_%s\n", xclbin);
195      cl_uint n_i0 = load_file_to_memory(xclbin, (char **) &kernelbinary);
196      if (n_i0 < 0) {
197          printf("ERROR:_failed_to_load_kernel_from_xclbin:_%s\n", xclbin);
198          printf("ERROR:_Test_failed\n");
199          return EXIT_FAILURE;
200      }
201
202      size_t n0 = n_i0;
203
204      // Create the compute program from offline
205      program = clCreateProgramWithBinary(context, 1, &device_id, &n0,
206      (const unsigned char **) &kernelbinary, &status, &err);
207      free(kernelbinary);
208
209      if ((!program) || (err!=CL_SUCCESS)) {
210          printf("ERROR:_Failed_to_create_compute_program_from_binary_%d!\n", err);
211          printf("ERROR:_Test_failed\n");
212          return EXIT_FAILURE;
213      }
214
215
216      // Build the program executable
217      //
218      err = clBuildProgram(program, 0, NULL, NULL, NULL, NULL);
219      if (err != CL_SUCCESS) {
220          size_t len;
221          char buffer[2048];
222
223          printf("ERROR:_Failed_to_build_program_executable!\n");
224          clGetProgramBuildInfo(program, device_id, CL_PROGRAM_BUILD_LOG, sizeof(buffer), buffer, &len);
225          printf("%s\n", buffer);
226          printf("ERROR:_Test_failed\n");
227          return EXIT_FAILURE;
228      }
229
230      // Create the compute kernel in the program we wish to run
231      //
232      kernel = clCreateKernel(program, "rtl_kernel_wizard_0", &err);
233      if (!kernel || err != CL_SUCCESS) {
234          printf("ERROR:_Failed_to_create_compute_kernel!\n");
235          printf("ERROR:_Test_failed\n");
236          return EXIT_FAILURE;
```

```
237          }
238
239          // Create structs to define memory bank mapping
240          cl_mem_ext_ptr_t mem_ext;
241          mem_ext.obj = NULL;
242          mem_ext.param = kernel;
243
244
245          mem_ext.flags = 1;
246          d_axi00_ptr0 = clCreateBuffer(context,  CL_MEM_READ_WRITE | CL_MEM_EXT_PTR_XILINX,  sizeof(cl_uint) *
                  number_of_words, &mem_ext, &err);
247          if (err != CL_SUCCESS) {
248              std::cout << "Return_code_for_clCreateBuffer_flags=" << mem_ext.flags << ":_" << err << std::endl;
249          }
250
251
252          if (!(d_axi00_ptr0)) {
253              printf("ERROR:_Failed_to_allocate_device_memory!\n");
254              printf("ERROR:_Test_failed\n");
255              return EXIT_FAILURE;
256          }
257
258
259          err = clEnqueueWriteBuffer(commands, d_axi00_ptr0, CL_TRUE, 0, sizeof(cl_uint) * number_of_words,
                  h_data, 0, NULL, NULL);
260          if (err != CL_SUCCESS) {
261              printf("ERROR:_Failed_to_write_to_source_array_h_data:_d_axi00_ptr0:_%d!\n", err);
262              printf("ERROR:_Test_failed\n");
263              return EXIT_FAILURE;
264          }
265
266
267          // Set the arguments to our compute kernel
268          // cl_uint vector_length = MAX_LENGTH;
269          err = 0;
270          cl_uint d_num = 0;
271          err |= clSetKernelArg(kernel, 0, sizeof(cl_uint), &d_num); // Not used in example RTL logic.
272          err |= clSetKernelArg(kernel, 1, sizeof(cl_mem), &d_axi00_ptr0);
273
274          if (err != CL_SUCCESS) {
275              printf("ERROR:_Failed_to_set_kernel_arguments!_%d\n", err);
276              printf("ERROR:_Test_failed\n");
277              return EXIT_FAILURE;
278          }
279
280          size_t global[1];
281          size_t local[1];
282          // Execute the kernel over the entire range of our 1d input data set
283          // using the maximum number of work group items for this device
284
285          global[0] = 1;
286          local[0] = 1;
287          err = clEnqueueNDRangeKernel(commands, kernel, 1, NULL, (size_t*)&global, (size_t*)&local, 0, NULL,
                  NULL);
288          if (err) {
289              printf("ERROR:_Failed_to_execute_kernel!_%d\n", err);
290              printf("ERROR:_Test_failed\n");
291              return EXIT_FAILURE;
292          }
293
294          clFinish(commands);
295
296
297          // Read back the results from the device to verify the output
298          //
299          cl_event readevent;
300
301          err = 0;
302          err |= clEnqueueReadBuffer( commands, d_axi00_ptr0, CL_TRUE, 0, sizeof(cl_uint) * number_of_words,
                  h_axi00_ptr0_output, 0, NULL, &readevent );
303
304
305          if (err != CL_SUCCESS) {
306              printf("ERROR:_Failed_to_read_output_array!_%d\n", err);
307              printf("ERROR:_Test_failed\n");
308              return EXIT_FAILURE;
309          }
310          clWaitForEvents(1, &readevent);
311          // Check Results
312
313          for (cl_uint i = 0; i < number_of_words; i++) {
314              //printf("h_data[i] %d, h_output[i] %d", h_data[i], h_axi00_ptr0_output[i]);
```

```
315                if (h_data[i] - 4 > 5) {
316                    h_data[i] = 5;
317                } else {
318                    h_data[i] -= 4;
319                }
320                //printf("h_data[i] %d, h_output[i] %d", h_data[i], h_axi00_ptr0_output[i]);
321                if (h_data[i] != h_axi00_ptr0_output[i]) {
322                    printf("ERROR in rtl_kernel_wizard_0::m00_axi - array index %d (host_addr 0x%03x) - input=%d
323                        (0x%x), output=%d (0x%x)\n", i, i*4, h_data[i], h_data[i], h_axi00_ptr0_output[i],
                            h_axi00_ptr0_output[i]);
324                    check_status = 1;
325                }
326                //  printf("i=%d, input=%d, output=%d\n", i,  h_axi00_ptr0_input[i], h_axi00_ptr0_output[i]);
327            }
328
329
330
331        //------------------------------------------------------------------------
332        // Shutdown and cleanup
333        //------------------------------------------------------------------------
334        clReleaseMemObject(d_axi00_ptr0);
335        free(h_axi00_ptr0_output);
336
337
338
339        free(h_data);
340        clReleaseProgram(program);
341        clReleaseKernel(kernel);
342        clReleaseCommandQueue(commands);
343        clReleaseContext(context);
344
345        if (check_status) {
346            printf("ERROR: Test failed\n");
347            return EXIT_FAILURE;
348        } else {
349            printf("INFO: Test completed successfully.\n");
350            return EXIT_SUCCESS;
351        }
352
353
354    } // end of main
```

Приложение 2

Листинг 2 – Содержимое log-файла

```
 1  INFO: [v++ 60-1306] Additional information associated with this v++ link can be found at:
 2  Reports: /iu_home/iu7039/workspace/p1/_x/reports/link
 3  Log files: /iu_home/iu7039/workspace/p1/_x/logs/link
 4  INFO: [v++ 60-1548] Creating build summary session with primary output /iu_home/iu7039/workspace/p1/vinc.
       xclbin.link_summary, at Sat Nov 20 11:42:41 2021
 5  INFO: [v++ 60-1316] Initiating connection to rulecheck server, at Sat Nov 20 11:42:42 2021
 6  INFO: [v++ 60-1315] Creating rulecheck session with output '/iu_home/iu7039/workspace/p1/_x/reports/link/v
       ++_link_vinc_guidance.html', at Sat Nov 20 11:43:00 2021
 7  INFO: [v++ 60-895]   Target platform: /opt/xilinx/platforms/xilinx_u200_xdma_201830_2/
       xilinx_u200_xdma_201830_2.xpfm
 8  INFO: [v++ 60-1578]   This platform contains Device Support Archive '/opt/xilinx/platforms/
       xilinx_u200_xdma_201830_2/hw/xilinx_u200_xdma_201830_2.dsa'
 9  INFO: [v++ 74-74] Compiler Version string: 2020.2
10  INFO: [v++ 60-1302] Platform 'xilinx_u200_xdma_201830_2.xpfm' has been explicitly enabled for this release
       .
11  INFO: [v++ 60-629] Linking for hardware target
12  INFO: [v++ 60-423]   Target device: xilinx_u200_xdma_201830_2
13  INFO: [v++ 60-1332] Run 'run_link' status: Not started
14  INFO: [v++ 60-1443] [11:43:51] Run run_link: Step system_link: Started
15  INFO: [v++ 60-1453] Command Line: system_link --xo /iu_home/iu7039/workspace/p1/Alveo_lab01_kernels/src/
       vitis_rtl_kernel/rtl_kernel_wizard_0/rtl_kernel_wizard_0.xo --config /iu_home/iu7039/workspace/p1/_x/
       link/int/syslinkConfig.ini --xpfm /opt/xilinx/platforms/xilinx_u200_xdma_201830_2/
       xilinx_u200_xdma_201830_2.xpfm --target hw --output_dir /iu_home/iu7039/workspace/p1/_x/link/int --
       temp_dir /iu_home/iu7039/workspace/p1/_x/link/sys_link
16  INFO: [v++ 60-1454] Run Directory: /iu_home/iu7039/workspace/p1/_x/link/run_link
17  INFO: [SYSTEM_LINK 60-1316] Initiating connection to rulecheck server, at Sat Nov 20 11:44:04 2021
18  INFO: [SYSTEM_LINK 82-70] Extracting xo v3 file /iu_home/iu7039/workspace/p1/Alveo_lab01_kernels/src/
       vitis_rtl_kernel/rtl_kernel_wizard_0/rtl_kernel_wizard_0.xo
19  INFO: [SYSTEM_LINK 82-53] Creating IP database /iu_home/iu7039/workspace/p1/_x/link/sys_link/_sysl/.cdb/
       xd_ip_db.xml
20  INFO: [SYSTEM_LINK 82-38] [11:44:06] build_xd_ip_db started: /data/Xilinx/Vitis/2020.2/bin/build_xd_ip_db
       -ip_search 0  -sds-pf /iu_home/iu7039/workspace/p1/_x/link/sys_link/xilinx_u200_xdma_201830_2.hpfm -
       clkid 0 -ip /iu_home/iu7039/workspace/p1/_x/link/sys_link/iprepo/
       mycompany_com_kernel_rtl_kernel_wizard_0_1_0,rtl_kernel_wizard_0 -o /iu_home/iu7039/workspace/p1/_x/
       link/sys_link/_sysl/.cdb/xd_ip_db.xml
21  INFO: [SYSTEM_LINK 82-37] [11:44:36] build_xd_ip_db finished successfully
22  Time (s): cpu = 00:00:31 ; elapsed = 00:00:30 . Memory (MB): peak = 1557.891 ; gain = 0.000 ; free
       physical = 264561 ; free virtual = 290855
23  INFO: [SYSTEM_LINK 82-51] Create system connectivity graph
24  INFO: [SYSTEM_LINK 82-102] Applying explicit connections to the system connectivity graph: /iu_home/iu7039
       /workspace/p1/_x/link/sys_link/cfgraph/cfgen_cfgraph.xml
25  INFO: [SYSTEM_LINK 82-38] [11:44:36] cfgen started: /data/Xilinx/Vitis/2020.2/bin/cfgen  -nk
       rtl_kernel_wizard_0:1:vinc0 -slr vinc0:SLR0 -sp vinc0.m00_axi:DDR[0] -dmclkid 0 -r /iu_home/iu7039/
       workspace/p1/_x/link/sys_link/_sysl/.cdb/xd_ip_db.xml -o /iu_home/iu7039/workspace/p1/_x/link/sys_link
       /cfgraph/cfgen_cfgraph.xml
26  INFO: [CFGEN 83-0] Kernel Specs:
27  INFO: [CFGEN 83-0]    kernel: rtl_kernel_wizard_0, num: 1  {vinc0}
28  INFO: [CFGEN 83-0] Port Specs:
29  INFO: [CFGEN 83-0]    kernel: vinc0, k_port: m00_axi, sptag: DDR[0]
30  INFO: [CFGEN 83-0] SLR Specs:
31  INFO: [CFGEN 83-0]    instance: vinc0, SLR: SLR0
32  INFO: [CFGEN 83-2228] Creating mapping for argument vinc0.axi00_ptr0 to DDR[0] for directive vinc0.m00_axi
       :DDR[0]
33  INFO: [SYSTEM_LINK 82-37] [11:45:01] cfgen finished successfully
34  Time (s): cpu = 00:00:25 ; elapsed = 00:00:25 . Memory (MB): peak = 1557.891 ; gain = 0.000 ; free
       physical = 264560 ; free virtual = 290846
35  INFO: [SYSTEM_LINK 82-52] Create top-level block diagram
36  INFO: [SYSTEM_LINK 82-38] [11:45:01] cf2bd started: /data/Xilinx/Vitis/2020.2/bin/cf2bd  --linux --
       trace_buffer 1024 --input_file /iu_home/iu7039/workspace/p1/_x/link/sys_link/cfgraph/cfgen_cfgraph.xml
       --ip_db /iu_home/iu7039/workspace/p1/_x/link/sys_link/_sysl/.cdb/xd_ip_db.xml --cf_name dr --
       working_dir /iu_home/iu7039/workspace/p1/_x/link/sys_link/_sysl/.xsd --temp_dir /iu_home/iu7039/
       workspace/p1/_x/link/sys_link --output_dir /iu_home/iu7039/workspace/p1/_x/link/int --target_bd
       pfm_dynamic.bd
37  INFO: [CF2BD 82-31] Launching cf2xd: cf2xd -linux -trace-buffer 1024 -i /iu_home/iu7039/workspace/p1/_x/
       link/sys_link/cfgraph/cfgen_cfgraph.xml -r /iu_home/iu7039/workspace/p1/_x/link/sys_link/_sysl/.cdb/
       xd_ip_db.xml -o dr.xml
38  INFO: [CF2BD 82-28] cf2xd finished successfully
39  INFO: [CF2BD 82-31] Launching cf_xsd: cf_xsd -disable-address-gen -bd pfm_dynamic.bd -dn dr -dp /iu_home/
       iu7039/workspace/p1/_x/link/sys_link/_sysl/.xsd
40  INFO: [CF2BD 82-28] cf_xsd finished successfully
41  INFO: [SYSTEM_LINK 82-37] [11:45:16] cf2bd finished successfully
42  Time (s): cpu = 00:00:13 ; elapsed = 00:00:14 . Memory (MB): peak = 1557.891 ; gain = 0.000 ; free
       physical = 264525 ; free virtual = 290816
43  INFO: [v++ 60-1441] [11:45:16] Run run_link: Step system_link: Completed
```

```
44  Time (s): cpu = 00:01:24 ; elapsed = 00:01:25 . Memory (MB): peak = 1553.914 ; gain = 0.000 ; free
       physical = 264582 ; free virtual = 290868
45  INFO: [v++ 60−1443] [11:45:16] Run run_link: Step cf2sw: Started
46  INFO: [v++ 60−1453] Command Line: cf2sw −sdsl /iu_home/iu7039/workspace/p1/_x/link/int/sdsl.dat −rtd /
       iu_home/iu7039/workspace/p1/_x/link/int/cf2sw.rtd −nofilter /iu_home/iu7039/workspace/p1/_x/link/int/
       cf2sw_full.rtd −xclbin /iu_home/iu7039/workspace/p1/_x/link/int/xclbin_orig.xml −o /iu_home/iu7039/
       workspace/p1/_x/link/int/xclbin_orig.1.xml
47  INFO: [v++ 60−1454] Run Directory: /iu_home/iu7039/workspace/p1/_x/link/run_link
48  INFO: [v++ 60−1441] [11:45:33] Run run_link: Step cf2sw: Completed
49  Time (s): cpu = 00:00:16 ; elapsed = 00:00:17 . Memory (MB): peak = 1553.914 ; gain = 0.000 ; free
       physical = 264551 ; free virtual = 290847
50  INFO: [v++ 60−1443] [11:45:33] Run run_link: Step rtd2_system_diagram: Started
51  INFO: [v++ 60−1453] Command Line: rtd2SystemDiagram
52  INFO: [v++ 60−1454] Run Directory: /iu_home/iu7039/workspace/p1/_x/link/run_link
53  INFO: [v++ 60−1441] [11:45:40] Run run_link: Step rtd2_system_diagram: Completed
54  Time (s): cpu = 00:00:00.01 ; elapsed = 00:00:08 . Memory (MB): peak = 1553.914 ; gain = 0.000 ; free
       physical = 263964 ; free virtual = 290259
55  INFO: [v++ 60−1443] [11:45:40] Run run_link: Step vpl: Started
56  INFO: [v++ 60−1453] Command Line: vpl −t hw −f xilinx_u200_xdma_201830_2 −−remote_ip_cache /iu_home/iu7039
       /workspace/p1/.ipcache −−output_dir /iu_home/iu7039/workspace/p1/_x/link/int −−log_dir /iu_home/iu7039
       /workspace/p1/_x/logs/link −−report_dir /iu_home/iu7039/workspace/p1/_x/reports/link −−config /iu_home
       /iu7039/workspace/p1/_x/link/int/vplConfig.ini −k /iu_home/iu7039/workspace/p1/_x/link/int/kernel_info
       .dat −−webtalk_flag Vitis −−temp_dir /iu_home/iu7039/workspace/p1/_x/link −−no−info −−iprepo /iu_home/
       iu7039/workspace/p1/_x/link/int/xo/ip_repo/mycompany_com_kernel_rtl_kernel_wizard_0_1_0 −−messageDb /
       iu_home/iu7039/workspace/p1/_x/link/run_link/vpl.pb /iu_home/iu7039/workspace/p1/_x/link/int/dr.bd.tcl
57  INFO: [v++ 60−1454] Run Directory: /iu_home/iu7039/workspace/p1/_x/link/run_link
58
59  ****** vpl v2020.2 (64−bit)
60  **** SW Build (by xbuild) on 2020−11−18−05:13:29
61  ** Copyright 1986−2020 Xilinx, Inc. All Rights Reserved.
62
63  INFO: [VPL 60−839] Read in kernel information from file '/iu_home/iu7039/workspace/p1/_x/link/int/
       kernel_info.dat'.
64  INFO: [VPL 74−74] Compiler Version string: 2020.2
65  INFO: [VPL 60−423]    Target device: xilinx_u200_xdma_201830_2
66  INFO: [VPL 60−1032] Extracting hardware platform to /iu_home/iu7039/workspace/p1/_x/link/vivado/vpl/.local
       /hw_platform
67  WARNING: /data/Xilinx/Vitis/2020.2/tps/lnx64/jre9.0.4 does not exist.
68  [11:50:52] Run vpl: Step create_project: Started
69  Creating Vivado project.
70  [11:51:19] Run vpl: Step create_project: Completed
71  [11:51:19] Run vpl: Step create_bd: Started
72  [11:53:03] Run vpl: Step create_bd: RUNNING...
73  [11:54:41] Run vpl: Step create_bd: RUNNING...
74  [11:56:18] Run vpl: Step create_bd: RUNNING...
75  [11:58:08] Run vpl: Step create_bd: RUNNING...
76  [11:59:44] Run vpl: Step create_bd: RUNNING...
77  [12:00:20] Run vpl: Step create_bd: Completed
78  [12:00:20] Run vpl: Step update_bd: Started
79  [12:00:22] Run vpl: Step update_bd: Completed
80  [12:00:22] Run vpl: Step generate_target: Started
81  [12:01:58] Run vpl: Step generate_target: RUNNING...
82  [12:03:27] Run vpl: Step generate_target: RUNNING...
83  [12:04:52] Run vpl: Step generate_target: RUNNING...
84  [12:06:24] Run vpl: Step generate_target: RUNNING...
85  [12:07:51] Run vpl: Step generate_target: RUNNING...
86  [12:09:26] Run vpl: Step generate_target: RUNNING...
87  [12:10:56] Run vpl: Step generate_target: RUNNING...
88  [12:11:30] Run vpl: Step generate_target: Completed
89  [12:11:30] Run vpl: Step config_hw_runs: Started
90  [12:13:02] Run vpl: Step config_hw_runs: Completed
91  [12:13:02] Run vpl: Step synth: Started
92  [12:15:38] Block−level synthesis in progress, 0 of 66 jobs complete, 8 jobs running.
93  [12:16:13] Block−level synthesis in progress, 0 of 66 jobs complete, 8 jobs running.
94  [12:16:51] Block−level synthesis in progress, 0 of 66 jobs complete, 8 jobs running.
95  [12:17:29] Block−level synthesis in progress, 0 of 66 jobs complete, 8 jobs running.
96  [12:18:07] Block−level synthesis in progress, 0 of 66 jobs complete, 8 jobs running.
97  [12:18:43] Block−level synthesis in progress, 0 of 66 jobs complete, 8 jobs running.
98  [12:19:22] Block−level synthesis in progress, 0 of 66 jobs complete, 8 jobs running.
99  [12:19:57] Block−level synthesis in progress, 0 of 66 jobs complete, 8 jobs running.
100 [12:20:36] Block−level synthesis in progress, 0 of 66 jobs complete, 8 jobs running.
101 [12:21:12] Block−level synthesis in progress, 0 of 66 jobs complete, 8 jobs running.
102 [12:21:52] Block−level synthesis in progress, 0 of 66 jobs complete, 8 jobs running.
103 [12:22:27] Block−level synthesis in progress, 0 of 66 jobs complete, 8 jobs running.
104 [12:23:06] Block−level synthesis in progress, 5 of 66 jobs complete, 3 jobs running.
105 [12:23:42] Block−level synthesis in progress, 6 of 66 jobs complete, 2 jobs running.
106 [12:24:21] Block−level synthesis in progress, 6 of 66 jobs complete, 7 jobs running.
107 [12:24:56] Block−level synthesis in progress, 6 of 66 jobs complete, 8 jobs running.
108 [12:25:35] Block−level synthesis in progress, 6 of 66 jobs complete, 8 jobs running.
109 [12:26:13] Block−level synthesis in progress, 8 of 66 jobs complete, 6 jobs running.
110 [12:26:51] Block−level synthesis in progress, 8 of 66 jobs complete, 6 jobs running.
111 [12:27:27] Block−level synthesis in progress, 8 of 66 jobs complete, 8 jobs running.
```

```
112  [12:28:05]  Block−level  synthesis  in  progress ,  8  of  66  jobs  complete ,  8  jobs  running .
113  [12:28:41]  Block−level  synthesis  in  progress ,  8  of  66  jobs  complete ,  8  jobs  running .
114  [12:29:19]  Block−level  synthesis  in  progress ,  8  of  66  jobs  complete ,  8  jobs  running .
115  [12:29:55]  Block−level  synthesis  in  progress ,  8  of  66  jobs  complete ,  8  jobs  running .
116  [12:30:33]  Block−level  synthesis  in  progress ,  8  of  66  jobs  complete ,  8  jobs  running .
117  [12:31:10]  Block−level  synthesis  in  progress ,  8  of  66  jobs  complete ,  8  jobs  running .
118  [12:31:49]  Block−level  synthesis  in  progress ,  12  of  66  jobs  complete ,  4  jobs  running .
119  [12:32:25]  Block−level  synthesis  in  progress ,  13  of  66  jobs  complete ,  3  jobs  running .
120  [12:33:04]  Block−level  synthesis  in  progress ,  13  of  66  jobs  complete ,  8  jobs  running .
121  [12:33:40]  Block−level  synthesis  in  progress ,  13  of  66  jobs  complete ,  8  jobs  running .
122  [12:34:18]  Block−level  synthesis  in  progress ,  14  of  66  jobs  complete ,  7  jobs  running .
123  [12:34:54]  Block−level  synthesis  in  progress ,  15  of  66  jobs  complete ,  6  jobs  running .
124  [12:35:33]  Block−level  synthesis  in  progress ,  15  of  66  jobs  complete ,  8  jobs  running .
125  [12:36:09]  Block−level  synthesis  in  progress ,  16  of  66  jobs  complete ,  7  jobs  running .
126  [12:36:47]  Block−level  synthesis  in  progress ,  16  of  66  jobs  complete ,  7  jobs  running .
127  [12:37:23]  Block−level  synthesis  in  progress ,  16  of  66  jobs  complete ,  8  jobs  running .
128  [12:38:02]  Block−level  synthesis  in  progress ,  16  of  66  jobs  complete ,  8  jobs  running .
129  [12:38:38]  Block−level  synthesis  in  progress ,  16  of  66  jobs  complete ,  8  jobs  running .
130  [12:39:17]  Block−level  synthesis  in  progress ,  16  of  66  jobs  complete ,  8  jobs  running .
131  [12:39:53]  Block−level  synthesis  in  progress ,  16  of  66  jobs  complete ,  8  jobs  running .
132  [12:40:31]  Block−level  synthesis  in  progress ,  19  of  66  jobs  complete ,  5  jobs  running .
133  [12:41:08]  Block−level  synthesis  in  progress ,  19  of  66  jobs  complete ,  5  jobs  running .
134  [12:41:47]  Block−level  synthesis  in  progress ,  19  of  66  jobs  complete ,  8  jobs  running .
135  [12:42:23]  Block−level  synthesis  in  progress ,  19  of  66  jobs  complete ,  8  jobs  running .
136  [12:43:02]  Block−level  synthesis  in  progress ,  20  of  66  jobs  complete ,  7  jobs  running .
137  [12:43:40]  Block−level  synthesis  in  progress ,  22  of  66  jobs  complete ,  5  jobs  running .
138  [12:44:18]  Block−level  synthesis  in  progress ,  22  of  66  jobs  complete ,  6  jobs  running .
139  [12:44:54]  Block−level  synthesis  in  progress ,  23  of  66  jobs  complete ,  7  jobs  running .
140  [12:45:32]  Block−level  synthesis  in  progress ,  23  of  66  jobs  complete ,  7  jobs  running .
141  [12:46:09]  Block−level  synthesis  in  progress ,  23  of  66  jobs  complete ,  8  jobs  running .
142  [12:46:47]  Block−level  synthesis  in  progress ,  23  of  66  jobs  complete ,  8  jobs  running .
143  [12:47:23]  Block−level  synthesis  in  progress ,  23  of  66  jobs  complete ,  8  jobs  running .
144  [12:48:01]  Block−level  synthesis  in  progress ,  23  of  66  jobs  complete ,  8  jobs  running .
145  [12:48:38]  Block−level  synthesis  in  progress ,  23  of  66  jobs  complete ,  8  jobs  running .
146  [12:49:17]  Block−level  synthesis  in  progress ,  23  of  66  jobs  complete ,  8  jobs  running .
147  [12:49:54]  Block−level  synthesis  in  progress ,  25  of  66  jobs  complete ,  6  jobs  running .
148  [12:50:32]  Block−level  synthesis  in  progress ,  25  of  66  jobs  complete ,  6  jobs  running .
149  [12:51:10]  Block−level  synthesis  in  progress ,  25  of  66  jobs  complete ,  8  jobs  running .
150  [12:51:49]  Block−level  synthesis  in  progress ,  28  of  66  jobs  complete ,  5  jobs  running .
151  [12:52:26]  Block−level  synthesis  in  progress ,  28  of  66  jobs  complete ,  5  jobs  running .
152  [12:53:06]  Block−level  synthesis  in  progress ,  29  of  66  jobs  complete ,  7  jobs  running .
153  [12:53:42]  Block−level  synthesis  in  progress ,  30  of  66  jobs  complete ,  6  jobs  running .
154  [12:54:20]  Block−level  synthesis  in  progress ,  31  of  66  jobs  complete ,  6  jobs  running .
155  [12:54:58]  Block−level  synthesis  in  progress ,  31  of  66  jobs  complete ,  7  jobs  running .
156  [12:55:34]  Block−level  synthesis  in  progress ,  31  of  66  jobs  complete ,  8  jobs  running .
157  [12:56:10]  Block−level  synthesis  in  progress ,  31  of  66  jobs  complete ,  8  jobs  running .
158  [12:56:48]  Block−level  synthesis  in  progress ,  31  of  66  jobs  complete ,  8  jobs  running .
159  [12:57:25]  Block−level  synthesis  in  progress ,  31  of  66  jobs  complete ,  8  jobs  running .
160  [12:58:04]  Block−level  synthesis  in  progress ,  31  of  66  jobs  complete ,  8  jobs  running .
161  [12:58:40]  Block−level  synthesis  in  progress ,  33  of  66  jobs  complete ,  6  jobs  running .
162  [12:59:19]  Block−level  synthesis  in  progress ,  33  of  66  jobs  complete ,  6  jobs  running .
163  [12:59:56]  Block−level  synthesis  in  progress ,  33  of  66  jobs  complete ,  8  jobs  running .
164  [13:00:33]  Block−level  synthesis  in  progress ,  35  of  66  jobs  complete ,  6  jobs  running .
165  [13:01:10]  Block−level  synthesis  in  progress ,  37  of  66  jobs  complete ,  4  jobs  running .
166  [13:01:50]  Block−level  synthesis  in  progress ,  37  of  66  jobs  complete ,  8  jobs  running .
167  [13:02:26]  Block−level  synthesis  in  progress ,  40  of  66  jobs  complete ,  5  jobs  running .
168  [13:03:05]  Block−level  synthesis  in  progress ,  42  of  66  jobs  complete ,  3  jobs  running .
169  [13:03:42]  Block−level  synthesis  in  progress ,  42  of  66  jobs  complete ,  6  jobs  running .
170  [13:04:20]  Block−level  synthesis  in  progress ,  42  of  66  jobs  complete ,  8  jobs  running .
171  [13:04:57]  Block−level  synthesis  in  progress ,  44  of  66  jobs  complete ,  6  jobs  running .
172  [13:05:36]  Block−level  synthesis  in  progress ,  44  of  66  jobs  complete ,  6  jobs  running .
173  [13:06:13]  Block−level  synthesis  in  progress ,  44  of  66  jobs  complete ,  8  jobs  running .
174  [13:06:52]  Block−level  synthesis  in  progress ,  45  of  66  jobs  complete ,  7  jobs  running .
175  [13:07:29]  Block−level  synthesis  in  progress ,  46  of  66  jobs  complete ,  6  jobs  running .
176  [13:08:08]  Block−level  synthesis  in  progress ,  46  of  66  jobs  complete ,  7  jobs  running .
177  [13:08:45]  Block−level  synthesis  in  progress ,  47  of  66  jobs  complete ,  7  jobs  running .
178  [13:09:24]  Block−level  synthesis  in  progress ,  48  of  66  jobs  complete ,  6  jobs  running .
179  [13:10:01]  Block−level  synthesis  in  progress ,  48  of  66  jobs  complete ,  7  jobs  running .
180  [13:10:38]  Block−level  synthesis  in  progress ,  48  of  66  jobs  complete ,  8  jobs  running .
181  [13:11:15]  Block−level  synthesis  in  progress ,  51  of  66  jobs  complete ,  5  jobs  running .
182  [13:11:54]  Block−level  synthesis  in  progress ,  53  of  66  jobs  complete ,  3  jobs  running .
183  [13:12:32]  Block−level  synthesis  in  progress ,  53  of  66  jobs  complete ,  7  jobs  running .
184  [13:13:10]  Block−level  synthesis  in  progress ,  54  of  66  jobs  complete ,  7  jobs  running .
185  [13:13:46]  Block−level  synthesis  in  progress ,  58  of  66  jobs  complete ,  3  jobs  running .
186  [13:14:26]  Block−level  synthesis  in  progress ,  58  of  66  jobs  complete ,  6  jobs  running .
187  [13:15:03]  Block−level  synthesis  in  progress ,  58  of  66  jobs  complete ,  8  jobs  running .
188  [13:15:42]  Block−level  synthesis  in  progress ,  60  of  66  jobs  complete ,  6  jobs  running .
189  [13:16:21]  Block−level  synthesis  in  progress ,  60  of  66  jobs  complete ,  6  jobs  running .
190  [13:16:59]  Block−level  synthesis  in  progress ,  60  of  66  jobs  complete ,  6  jobs  running .
191  [13:17:37]  Block−level  synthesis  in  progress ,  60  of  66  jobs  complete ,  6  jobs  running .
192  [13:18:17]  Block−level  synthesis  in  progress ,  60  of  66  jobs  complete ,  6  jobs  running .
193  [13:18:55]  Block−level  synthesis  in  progress ,  61  of  66  jobs  complete ,  5  jobs  running .
```

```
194 [13:19:35] Block−level synthesis in progress, 61 of 66 jobs complete, 5 jobs running.
195 [13:20:13] Block−level synthesis in progress, 62 of 66 jobs complete, 4 jobs running.
196 [13:20:53] Block−level synthesis in progress, 63 of 66 jobs complete, 3 jobs running.
197 [13:21:31] Block−level synthesis in progress, 63 of 66 jobs complete, 3 jobs running.
198 [13:22:11] Block−level synthesis in progress, 64 of 66 jobs complete, 2 jobs running.
199 [13:22:48] Block−level synthesis in progress, 64 of 66 jobs complete, 2 jobs running.
200 [13:23:29] Block−level synthesis in progress, 64 of 66 jobs complete, 2 jobs running.
201 [13:24:08] Block−level synthesis in progress, 65 of 66 jobs complete, 1 job running.
202 [13:24:48] Block−level synthesis in progress, 66 of 66 jobs complete, 0 jobs running.
203 [13:25:26] Block−level synthesis in progress, 66 of 66 jobs complete, 0 jobs running.
204 [13:26:06] Top−level synthesis in progress.
205 [13:26:44] Top−level synthesis in progress.
206 [13:27:25] Top−level synthesis in progress.
207 [13:28:03] Top−level synthesis in progress.
208 [13:28:43] Top−level synthesis in progress.
209 [13:29:21] Top−level synthesis in progress.
210 [13:30:01] Top−level synthesis in progress.
211 [13:30:38] Top−level synthesis in progress.
212 [13:31:19] Top−level synthesis in progress.
213 [13:31:57] Top−level synthesis in progress.
214 [13:32:38] Top−level synthesis in progress.
215 [13:33:16] Top−level synthesis in progress.
216 [13:33:56] Top−level synthesis in progress.
217 [13:34:33] Top−level synthesis in progress.
218 [13:35:14] Top−level synthesis in progress.
219 [13:35:51] Top−level synthesis in progress.
220 [13:36:31] Top−level synthesis in progress.
221 [13:36:55] Run vpl: Step synth: Completed
222 [13:36:55] Run vpl: Step impl: Started
223 [14:39:18] Finished 2nd of 6 tasks (FPGA linking synthesized kernels to platform). Elapsed time: 02h 53m
         26s
224
225 [14:39:18] Starting logic optimization..
226 [14:45:53] Phase 1 Generate And Synthesize MIG Cores
227 [15:23:12] Phase 2 Generate And Synthesize Debug Cores
228 [15:49:15] Phase 3 Retarget
229 [15:52:00] Phase 4 Constant propagation
230 [15:53:22] Phase 5 Sweep
231 [15:58:08] Phase 6 BUFG optimization
232 [16:00:12] Phase 7 Shift Register Optimization
233 [16:00:55] Phase 8 Post Processing Netlist
234 [16:14:55] Finished 3rd of 6 tasks (FPGA logic optimization). Elapsed time: 01h 35m 37s
235
236 [16:14:55] Starting logic placement..
237 [16:19:40] Phase 1 Placer Initialization
238 [16:19:40] Phase 1.1 Placer Initialization Netlist Sorting
239 [16:31:40] Phase 1.2 IO Placement/ Clock Placement/ Build Placer Device
240 [16:39:35] Phase 1.3 Build Placer Netlist Model
241 [16:52:12] Phase 1.4 Constrain Clocks/Macros
242 [16:53:37] Phase 2 Global Placement
243 [16:53:37] Phase 2.1 Floorplanning
244 [16:57:14] Phase 2.1.1 Partition Driven Placement
245 [16:57:14] Phase 2.1.1.1 PBP: Partition Driven Placement
246 [16:59:27] Phase 2.1.1.2 PBP: Clock Region Placement
247 [17:04:04] Phase 2.1.1.3 PBP: Compute Congestion
248 [17:05:23] Phase 2.1.1.4 PBP: UpdateTiming
249 [17:06:52] Phase 2.1.1.5 PBP: Add part constraints
250 [17:06:52] Phase 2.2 Update Timing before SLR Path Opt
251 [17:07:34] Phase 2.3 Global Placement Core
252 [17:39:57] Phase 2.3.1 Physical Synthesis In Placer
253 [17:53:24] Phase 3 Detail Placement
254 [17:53:24] Phase 3.1 Commit Multi Column Macros
255 [17:54:12] Phase 3.2 Commit Most Macros & LUTRAMs
256 [17:59:15] Phase 3.3 Small Shape DP
257 [17:59:15] Phase 3.3.1 Small Shape Clustering
258 [18:01:34] Phase 3.3.2 Flow Legalize Slice Clusters
259 [18:02:18] Phase 3.3.3 Slice Area Swap
260 [18:07:36] Phase 3.4 Place Remaining
261 [18:08:19] Phase 3.5 Re−assign LUT pins
262 [18:09:49] Phase 3.6 Pipeline Register Optimization
263 [18:09:49] Phase 3.7 Fast Optimization
264 [18:14:21] Phase 4 Post Placement Optimization and Clean−Up
265 [18:14:21] Phase 4.1 Post Commit Optimization
266 [18:23:07] Phase 4.1.1 Post Placement Optimization
267 [18:23:53] Phase 4.1.1.1 BUFG Insertion
268 [18:23:53] Phase 1 Physical Synthesis Initialization
269 [18:26:48] Phase 4.1.1.2 BUFG Replication
270 [18:29:44] Phase 4.1.1.3 Replication
271 [18:36:11] Phase 4.2 Post Placement Cleanup
272 [18:36:56] Phase 4.3 Placer Reporting
273 [18:36:56] Phase 4.3.1 Print Estimated Congestion
274 [18:38:24] Phase 4.4 Final Placement Cleanup
```

```
275 [19:46:20] Finished 4th of 6 tasks (FPGA logic placement). Elapsed time: 03h 31m 24s
276
277 [19:46:20] Starting logic routing..
278 [19:51:30] Phase 1 Build RT Design
279 [20:01:57] Phase 2 Router Initialization
280 [20:02:44] Phase 2.1 Fix Topology Constraints
281 [20:02:44] Phase 2.2 Pre Route Cleanup
282 [20:03:26] Phase 2.3 Global Clock Net Routing
283 [20:06:31] Phase 2.4 Update Timing
284 [20:19:59] Phase 2.5 Update Timing for Bus Skew
285 [20:19:59] Phase 2.5.1 Update Timing
286 [20:25:04] Phase 3 Initial Routing
287 [20:25:04] Phase 3.1 Global Routing
288 [20:29:31] Phase 4 Rip-up And Reroute
289 [20:29:31] Phase 4.1 Global Iteration 0
290 [20:55:43] Phase 4.2 Global Iteration 1
291 [21:08:17] Phase 4.3 Global Iteration 2
292 [21:12:46] Phase 5 Delay and Skew Optimization
293 [21:12:46] Phase 5.1 Delay CleanUp
294 [21:12:46] Phase 5.1.1 Update Timing
295 [21:19:22] Phase 5.2 Clock Skew Optimization
296 [21:20:08] Phase 6 Post Hold Fix
297 [21:20:08] Phase 6.1 Hold Fix Iter
298 [21:20:51] Phase 6.1.1 Update Timing
299 [21:25:57] Phase 7 Route finalize
300 [21:26:38] Phase 8 Verifying routed nets
301 [21:28:07] Phase 9 Depositing Routes
302 [21:31:51] Phase 10 Route finalize
303 [21:32:33] Phase 11 Post Router Timing
304 [21:39:06] Finished 5th of 6 tasks (FPGA routing). Elapsed time: 01h 52m 46s
305
306 [21:39:06] Starting bitstream generation..
307 [23:33:58] Creating bitmap...
308 [00:21:31] Writing bitstream ./pfm_top_i_dynamic_region_my_rm_partial.bit...
309 [00:22:18] Finished 6th of 6 tasks (FPGA bitstream generation). Elapsed time: 02h 43m 11s
310 [00:26:12] Run vpl: Step impl: Completed
311 [00:26:30] Run vpl: FINISHED. Run Status: impl Complete!
312 INFO: [v++ 60-1441] [00:27:12] Run run_link: Step vpl: Completed
313 Time (s): cpu = 00:41:43 ; elapsed = 12:41:31 . Memory (MB): peak = 1553.914 ; gain = 0.000 ; free
        physical = 230667 ; free virtual = 258886
314 INFO: [v++ 60-1443] [00:27:12] Run run_link: Step rtdgen: Started
315 INFO: [v++ 60-1453] Command Line: rtdgen
316 INFO: [v++ 60-1454] Run Directory: /iu_home/iu7039/workspace/p1/_x/link/run_link
317 INFO: [v++ 60-991] clock name 'clkwiz_kernel_clk_out1' (clock ID '0') is being mapped to clock name '
        DATA_CLK' in the xclbin
318 INFO: [v++ 60-991] clock name 'clkwiz_kernel2_clk_out1' (clock ID '1') is being mapped to clock name '
        KERNEL_CLK' in the xclbin
319 INFO: [v++ 60-1230] The compiler selected the following frequencies for the runtime controllable kernel
        clock(s) and scalable system clock(s): Kernel (DATA) clock: clkwiz_kernel_clk_out1 = 300, Kernel (
        KERNEL) clock: clkwiz_kernel2_clk_out1 = 500
320 INFO: [v++ 60-1453] Command Line: cf2sw -a /iu_home/iu7039/workspace/p1/_x/link/int/address_map.xml -sdsl
        /iu_home/iu7039/workspace/p1/_x/link/int/sdsl.dat -xclbin /iu_home/iu7039/workspace/p1/_x/link/int/
        xclbin_orig.xml -rtd /iu_home/iu7039/workspace/p1/_x/link/int/vinc.rtd -o /iu_home/iu7039/workspace/p1
        /_x/link/int/vinc.xml
321 INFO: [v++ 60-1652] Cf2sw returned exit code: 0
322 INFO: [v++ 60-2311] HPISystemDiagram::writeSystemDiagramAfterRunningVivado, rtdInputFilePath: /iu_home/
        iu7039/workspace/p1/_x/link/int/vinc.rtd
323 INFO: [v++ 60-2312] HPISystemDiagram::writeSystemDiagramAfterRunningVivado, systemDiagramOutputFilePath: /
        iu_home/iu7039/workspace/p1/_x/link/int/systemDiagramModelSlrBaseAddress.json
324 INFO: [v++ 60-1618] Launching
325 INFO: [v++ 60-1441] [00:27:27] Run run_link: Step rtdgen: Completed
326 Time (s): cpu = 00:00:14 ; elapsed = 00:00:15 . Memory (MB): peak = 1553.914 ; gain = 0.000 ; free
        physical = 230695 ; free virtual = 258914
327 INFO: [v++ 60-1443] [00:27:27] Run run_link: Step xclbinutil: Started
328 INFO: [v++ 60-1453] Command Line: xclbinutil --add-section DEBUG_IP_LAYOUT:JSON:/iu_home/iu7039/workspace/
        p1/_x/link/int/debug_ip_layout.rtd --add-section BITSTREAM:RAW:/iu_home/iu7039/workspace/p1/_x/link/
        int/partial.bit --force --target hw --key-value SYS:dfx_enable:true --add-section :JSON:/iu_home/
        iu7039/workspace/p1/_x/link/int/vinc.rtd --append-section :JSON:/iu_home/iu7039/workspace/p1/_x/link/
        int/appendSection.rtd --add-section CLOCK_FREQ_TOPOLOGY:JSON:/iu_home/iu7039/workspace/p1/_x/link/int/
        vinc_xml.rtd --add-section BUILD_METADATA:JSON:/iu_home/iu7039/workspace/p1/_x/link/int/vinc_build.rtd
         --add-section EMBEDDED_METADATA:RAW:/iu_home/iu7039/workspace/p1/_x/link/int/vinc.xml --add-section
        SYSTEM_METADATA:RAW:/iu_home/iu7039/workspace/p1/_x/link/int/systemDiagramModelSlrBaseAddress.json --
        output /iu_home/iu7039/workspace/p1/vinc.xclbin
329 INFO: [v++ 60-1454] Run Directory: /iu_home/iu7039/workspace/p1/_x/link/run_link
330 XRT Build Version: 2.8.743 (2020.2)
331 Build Date: 2020-11-16 00:19:11
332 Hash ID: 77d5484b5c4daa691a7f78235053fb036829b1e9
333 Creating a default 'in-memory' xclbin image.
334
335 Section: 'DEBUG_IP_LAYOUT'(9) was successfully added.
336 Size   : 440 bytes
337 Format : JSON
```

```
338  File    : '/iu_home/iu7039/workspace/p1/_x/link/int/debug_ip_layout.rtd'
339
340  Section: 'BITSTREAM'(0) was successfully added.
341  Size    : 43430742 bytes
342  Format : RAW
343  File    : '/iu_home/iu7039/workspace/p1/_x/link/int/partial.bit'
344
345  Section: 'MEM_TOPOLOGY'(6) was successfully added.
346  Format : JSON
347  File    : 'mem_topology'
348
349  Section: 'IP_LAYOUT'(8) was successfully added.
350  Format : JSON
351  File    : 'ip_layout'
352
353  Section: 'CONNECTIVITY'(7) was successfully added.
354  Format : JSON
355  File    : 'connectivity'
356  Section: 'CLOCK_FREQ_TOPOLOGY'(11) was successfully added.
357  Size    : 274 bytes
358  Format : JSON
359  File    : '/iu_home/iu7039/workspace/p1/_x/link/int/vinc_xml.rtd'
360  Section: 'BUILD_METADATA'(14) was successfully added.
361  Size    : 3025 bytes
362  Format : JSON
363  File    : '/iu_home/iu7039/workspace/p1/_x/link/int/vinc_build.rtd'
364  Section: 'EMBEDDED_METADATA'(2) was successfully added.
365  Size    : 2754 bytes
366  Format : RAW
367  File    : '/iu_home/iu7039/workspace/p1/_x/link/int/vinc.xml'
368  Section: 'SYSTEM_METADATA'(22) was successfully added.
369  Size    : 6254 bytes
370  Format : RAW
371  File    : '/iu_home/iu7039/workspace/p1/_x/link/int/systemDiagramModelSlrBaseAddress.json'
372  Section: 'IP_LAYOUT'(8) was successfully appended to.
373  Format : JSON
374  File    : 'ip_layout'
375  Successfully wrote (43453522 bytes) to the output file: /iu_home/iu7039/workspace/p1/vinc.xclbin
376  Leaving xclbinutil.
377  INFO: [v++ 60-1441] [00:27:30] Run run_link: Step xclbinutil: Completed
378  Time (s): cpu = 00:00:00.53 ; elapsed = 00:00:03 . Memory (MB): peak = 1553.914 ; gain = 0.000 ; free
         physical = 230646 ; free virtual = 258906
379  INFO: [v++ 60-1443] [00:27:30] Run run_link: Step xclbinutilinfo: Started
380  INFO: [v++ 60-1453] Command Line: xclbinutil --quiet --force --info /iu_home/iu7039/workspace/p1/vinc.
         xclbin.info --input /iu_home/iu7039/workspace/p1/vinc.xclbin
381  INFO: [v++ 60-1454] Run Directory: /iu_home/iu7039/workspace/p1/_x/link/run_link
382  INFO: [v++ 60-1441] [00:27:34] Run run_link: Step xclbinutilinfo: Completed
383  Time (s): cpu = 00:00:03 ; elapsed = 00:00:04 . Memory (MB): peak = 1553.914 ; gain = 0.000 ; free
         physical = 230622 ; free virtual = 258883
384  INFO: [v++ 60-1443] [00:27:34] Run run_link: Step generate_sc_driver: Started
385  INFO: [v++ 60-1453] Command Line:
386  INFO: [v++ 60-1454] Run Directory: /iu_home/iu7039/workspace/p1/_x/link/run_link
387  INFO: [v++ 60-1441] [00:27:34] Run run_link: Step generate_sc_driver: Completed
388  Time (s): cpu = 00:00:00 ; elapsed = 00:00:00.06 . Memory (MB): peak = 1553.914 ; gain = 0.000 ; free
         physical = 230618 ; free virtual = 258879
389  INFO: [v++ 60-244] Generating system estimate report...
390  INFO: [v++ 60-1092] Generated system estimate report: /iu_home/iu7039/workspace/p1/_x/reports/link/
         system_estimate_vinc.xtxt
391  INFO: [v++ 60-586] Created /iu_home/iu7039/workspace/p1/vinc.ltx
392  INFO: [v++ 60-586] Created vinc.xclbin
393  INFO: [v++ 60-1307] Run completed. Additional information can be found in:
394  Guidance: /iu_home/iu7039/workspace/p1/_x/reports/link/v++_link_vinc_guidance.html
395  Timing Report: /iu_home/iu7039/workspace/p1/_x/reports/link/imp/
         impl_1_xilinx_u200_xdma_201830_2_bb_locked_timing_summary_routed.rpt
396  Vivado Log: /iu_home/iu7039/workspace/p1/_x/logs/link/vivado.log
397  Steps Log File: /iu_home/iu7039/workspace/p1/_x/logs/link/link.steps.log
398  INFO: [v++ 60-2343] Use the vitis_analyzer tool to visualize and navigate the relevant reports. Run the
         following command.
399  vitis_analyzer /iu_home/iu7039/workspace/p1/vinc.xclbin.link_summary
400  INFO: [v++ 60-791] Total elapsed time: 12h 46m 1s
401  INFO: [v++ 60-1653] Closing dispatch client.
```

Приложение 3

Листинг 3 – Содержимое xclbin.info-файла

```
1 
2 ═══════════════════════════════════════════════════════════════════════════
3 XRT Build Version: 2.8.743 (2020.2)
4 Build Date: 2020−11−16 00:19:11
5 Hash ID: 77d5484b5c4daa691a7f78235053fb036829b1e9
6 ═══════════════════════════════════════════════════════════════════════════
7 xclbin Information
8 ──────────────────
9 Generated by:          v++ (2020.2) on 2020−11−18−05:13:29
10 Version:               2.8.743
11 Kernels:               rtl_kernel_wizard_0
12 Signature:
13 Content:               Bitstream
14 UUID (xclbin):         f48ab827−3033−4124−9be2−3e2b03b06a1f
15 Sections:              DEBUG_IP_LAYOUT, BITSTREAM, MEM_TOPOLOGY, IP_LAYOUT,
16 CONNECTIVITY, CLOCK_FREQ_TOPOLOGY, BUILD_METADATA,
17 EMBEDDED_METADATA, SYSTEM_METADATA,
18 GROUP_CONNECTIVITY, GROUP_TOPOLOGY
19 ═══════════════════════════════════════════════════════════════════════════
20 Hardware Platform (Shell) Information
21 ────────────────────────────────────────────
22 Vendor:                xilinx
23 Board:                 u200
24 Name:                  xdma
25 Version:               201830.2
26 Generated Version:     Vivado 2018.3 (SW Build: 2568420)
27 Created:               Tue Jun 25 06:55:20 2019
28 FPGA Device:           xcu200
29 Board Vendor:          xilinx.com
30 Board Name:            xilinx.com:au200:1.0
31 Board Part:            xilinx.com:au200:part0:1.0
32 Platform VBNV:         xilinx_u200_xdma_201830_2
33 Static UUID:           c102e7af−b2b8−4381−992b−9a00cc3863eb
34 Feature ROM TimeStamp: 1561465320
35 
36 Clocks
37 ──────
38 Name:       DATA_CLK
39 Index:      0
40 Type:       DATA
41 Frequency:  300 MHz
42 
43 Name:       KERNEL_CLK
44 Index:      1
45 Type:       KERNEL
46 Frequency:  500 MHz
47 
48 Memory Configuration
49 ────────────────────
50 Name:          bank0
51 Index:         0
52 Type:          MEM_DDR4
53 Base Address:  0x4000000000
54 Address Size:  0x400000000
55 Bank Used:     Yes
56 
57 Name:          bank1
58 Index:         1
59 Type:          MEM_DDR4
60 Base Address:  0x5000000000
61 Address Size:  0x400000000
62 Bank Used:     No
63 
64 Name:          bank2
65 Index:         2
66 Type:          MEM_DDR4
67 Base Address:  0x6000000000
68 Address Size:  0x400000000
69 Bank Used:     No
70 
71 Name:          bank3
72 Index:         3
73 Type:          MEM_DDR4
74 Base Address:  0x7000000000
75 Address Size:  0x400000000
```

```
76  Bank Used:      No
77
78  Name:            PLRAM[0]
79  Index:           4
80  Type:            MEM_DRAM
81  Base Address: 0x3000000000
82  Address Size: 0x20000
83  Bank Used:      No
84
85  Name:            PLRAM[1]
86  Index:           5
87  Type:            MEM_DRAM
88  Base Address: 0x3000200000
89  Address Size: 0x20000
90  Bank Used:      No
91
92  Name:            PLRAM[2]
93  Index:           6
94  Type:            MEM_DRAM
95  Base Address: 0x3000400000
96  Address Size: 0x20000
97  Bank Used:      No
98  ===================================================================================
99  Kernel: rtl_kernel_wizard_0
100
101  Definition
102  ----------
103  Signature: rtl_kernel_wizard_0 (uint num, int* axi00_ptr0)
104
105  Ports
106  -----
107  Port:             s_axi_control
108  Mode:             slave
109  Range (bytes): 0x1000
110  Data Width:    32 bits
111  Port Type:       addressable
112
113  Port:             m00_axi
114  Mode:             master
115  Range (bytes): 0xFFFFFFFFFFFFFFFF
116  Data Width:    512 bits
117  Port Type:       addressable
118
119  -------------------------------
120  Instance:        vinc0
121  Base Address: 0x1c00000
122
123  Argument:              num
124  Register Offset:    0x010
125  Port:             s_axi_control
126  Memory:                <not applicable>
127
128  Argument:              axi00_ptr0
129  Register Offset:    0x018
130  Port:             m00_axi
131  Memory:                bank0 (MEM_DDR4)
132  ===================================================================================
133  Generated By
134  ------------
135  Command:         v++
136  Version:         2020.2 - 2020-11-18-05:13:29 (SW BUILD: 0)
137  Command Line:  v++ —config /iu_home/iu7039/workspace/p1/Alveo_lab01.cfg —connectivity.nk
         rtl_kernel_wizard_0:1:vinc0 —connectivity.slr vinc0:SLR0 —connectivity.sp vinc0.m00_axi:DDR[0] —
         input_files /iu_home/iu7039/workspace/p1/Alveo_lab01_kernels/src/vitis_rtl_kernel/rtl_kernel_wizard_0/
         rtl_kernel_wizard_0.xo —link —optimize 0 —output vinc.xclbin —platform xilinx_u200_xdma_201830_2
         —report_level 0 —target hw —vivado.prop run.impl_1.STEPS.OPT_DESIGN.ARGS.DIRECTIVE=Explore —vivado
         .prop run.impl_1.STEPS.PLACE_DESIGN.ARGS.DIRECTIVE=Explore —vivado.prop run.impl_1.STEPS.
         PHYS_OPT_DESIGN.IS_ENABLED=true —vivado.prop run.impl_1.STEPS.PHYS_OPT_DESIGN.ARGS.DIRECTIVE=
         AggressiveExplore —vivado.prop run.impl_1.STEPS.ROUTE_DESIGN.ARGS.DIRECTIVE=Explore
138  Options:          —config /iu_home/iu7039/workspace/p1/Alveo_lab01.cfg
139  —connectivity.nk rtl_kernel_wizard_0:1:vinc0
140  —connectivity.slr vinc0:SLR0
141  —connectivity.sp vinc0.m00_axi:DDR[0]
142  —input_files /iu_home/iu7039/workspace/p1/Alveo_lab01_kernels/src/vitis_rtl_kernel/rtl_kernel_wizard_0/
         rtl_kernel_wizard_0.xo
143  —link
144  —optimize 0
145  —output vinc.xclbin
146  —platform xilinx_u200_xdma_201830_2
147  —report_level 0
148  —target hw
149  —vivado.prop run.impl_1.STEPS.OPT_DESIGN.ARGS.DIRECTIVE=Explore
```

```
150 ——vivado.prop run.impl_1.STEPS.PLACE_DESIGN.ARGS.DIRECTIVE=Explore
151 ——vivado.prop run.impl_1.STEPS.PHYS_OPT_DESIGN.IS_ENABLED=true
152 ——vivado.prop run.impl_1.STEPS.PHYS_OPT_DESIGN.ARGS.DIRECTIVE=AggressiveExplore
153 ——vivado.prop run.impl_1.STEPS.ROUTE_DESIGN.ARGS.DIRECTIVE=Explore
154 ================================================================================
155 User Added Key Value Pairs
156 ————————————————————
157 <empty>
```