СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является изучение архитектуры гетерогенных вычислительных систем и технологии разработки ускорителей вычислений на базе ПЛИС фирмы Xilinx.

Для достижения данной цели необходимо выполнить следующие задачи:

- изучить основные сведения о платформе Xilinx Alveo U200;
- разработать RTL 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.

1.1 Функциональная схема разрабатываемой аппаратной системы

1.2 Изучение работа шины AXI

По моему варианту требовалось реализовать функцию в соответствии с формулой 1.1.

$$R[i] = A[i]/16 - 11 \tag{1.1}$$

Листинг 1.1 — Изменный код модуля rtl_kernel_wizard_0_example_adder.v

```
// 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]/16 - 11;
  end
end
```

Далее приведены диаграммы, иллюстрирующие процесс рукопожатия и пакетного чтения.

Чтобы указать завершение пакетного чтения и записи, устройство использует сигнал RLAST. На диаграмме этого нельзя увидеть, так как изначально было указано малое время симуляции.

Ниже на рисунке 1.1 приведена транзакция чтения данных вектора на шине AXI4 MM из DDR памяти.

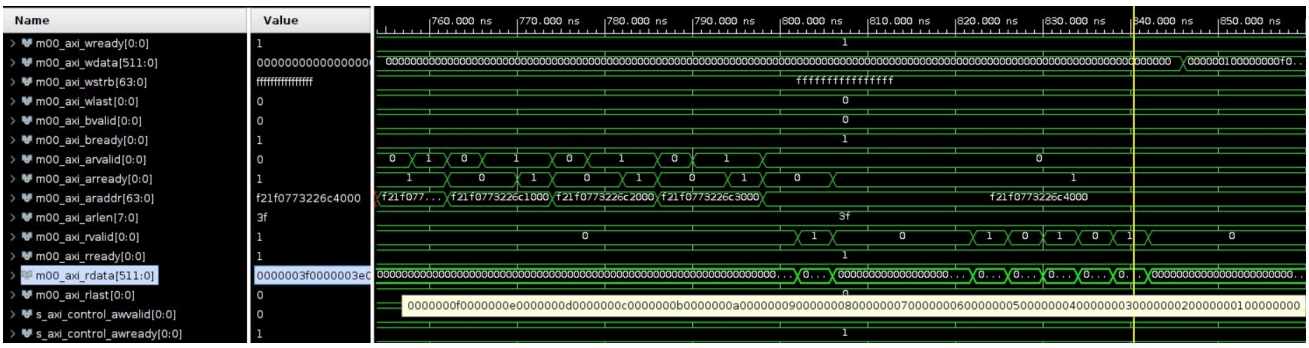

Рисунок 1.1 — Транзакция чтения данных вектора на шине AXI4 MM из DDR памяти.

Ниже на рисунке 1.2 приведена транзакция записи результата инкремента данных на шине AXI4 MM. Видно, что каждое прочитанное значение было инкрементировано.

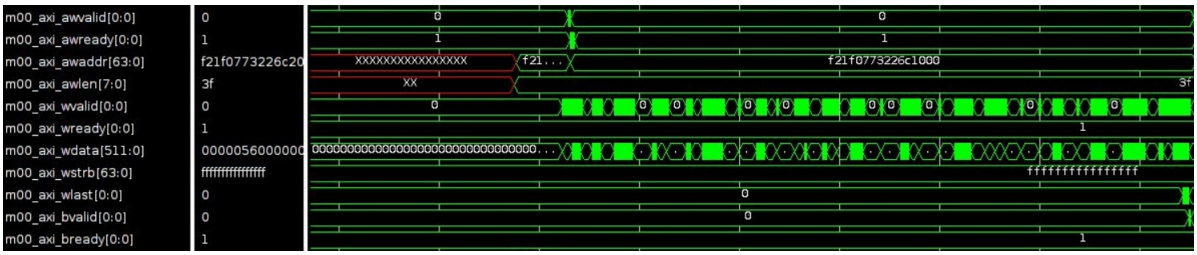

Рисунок 1.2 — Транзакция записи результата инкремента данных на шине AXI4 MM

Ниже на рисунке 1.3 приведен инкремент данных на шине AXI4 MM.

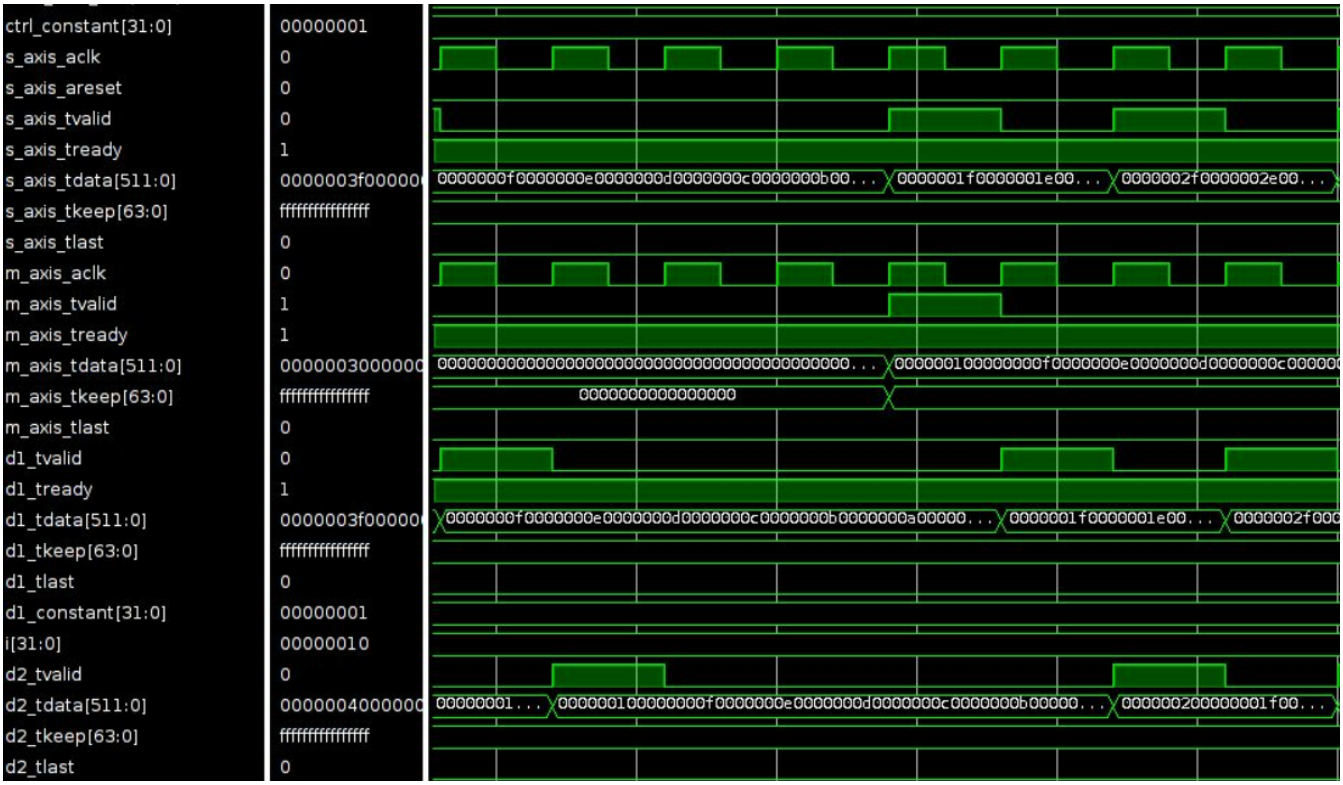

Рисунок 1.3 — Инкремент данных на шине AXI4 MM

Ниже на рисунке 1.4 приведен фрагмент кода модуля tl_kernel_wizard_0_example_adder.v (до изменения) с выполнением инкрементирования данных.

```
  dl_constant <= ctrl_constant;
end

// 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

// Register inputs to fifo
```

Рисунок 1.4 — Код модуля tl_kernel_wizard_0_example_adder.v с выполнением инкрементирования данных

Теперь изменим модуль rtl_kernel_wizard_0_example_adder.v, чтобы ускоритель выполнял предложенную функцию.

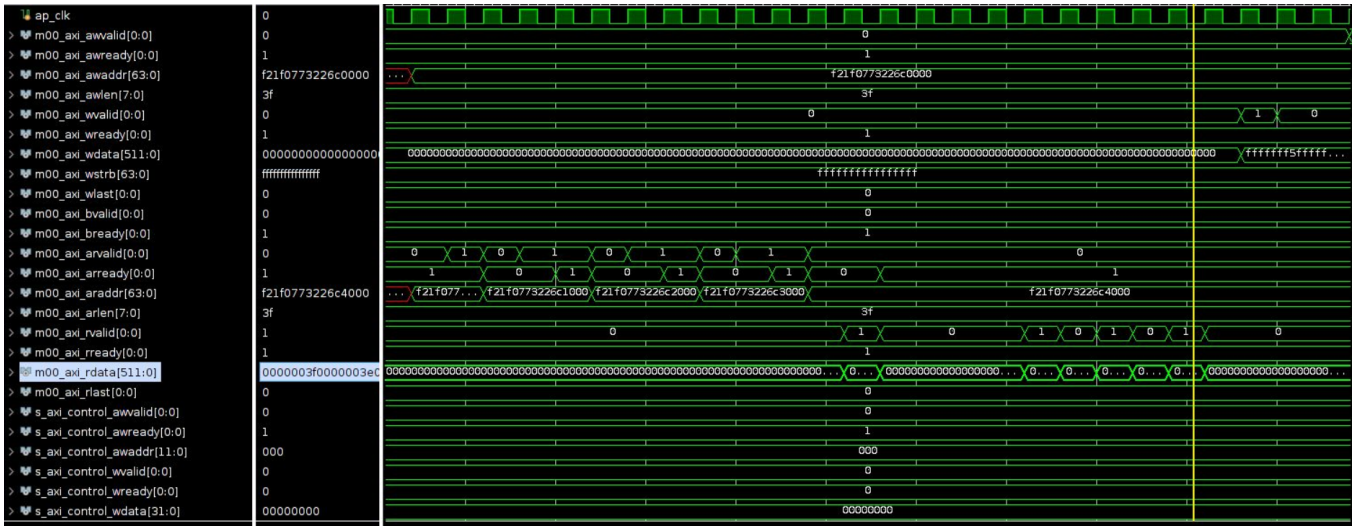

Рисунок 1.5 — Транзакция чтения данных вектора на шине AXI4 MM из DDR памяти.

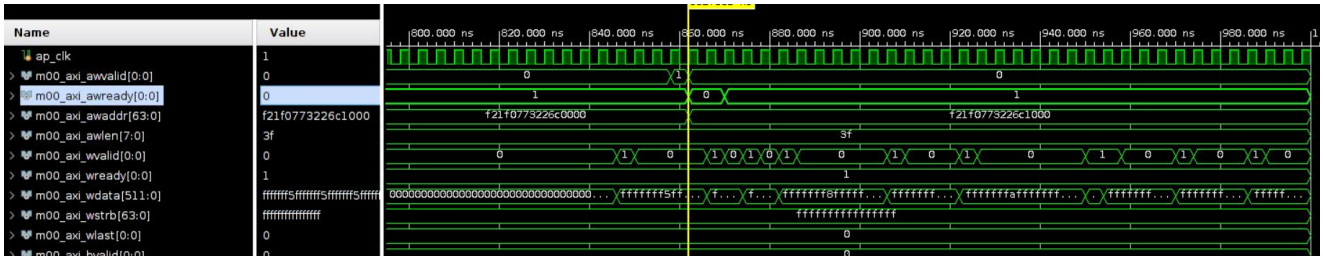

Рисунок 1.6 — Транзакция записи результата инкремента данных на шине AXI4 MM

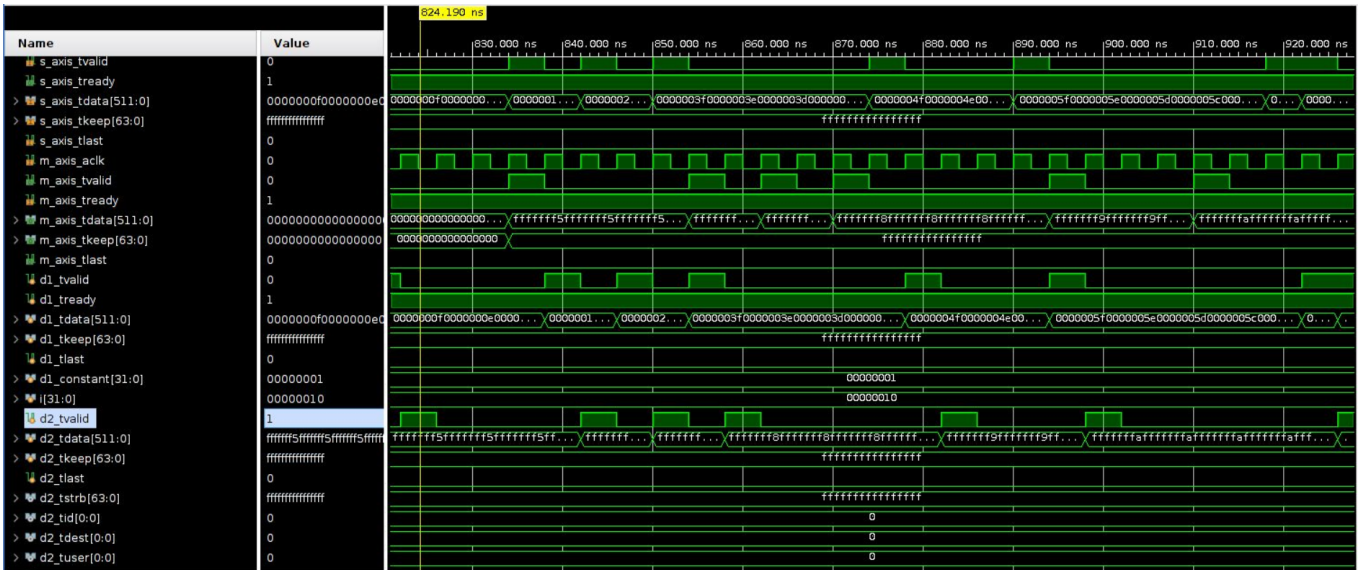

Рисунок 1.7 — Инкремент данных на шине AXI4 MM

1.3 Сборка проекта

Ниже в листинге 1.2 приведено содержимое конфигурационного файла. В соответствии с вариантом требовалось использовать регионы SLR1 и DDR[1].

Листинг 1.2 — Содержимое конфигурационного файла

```
[connectivity]
 nk=rtl_kernel_vinc:2:vinc0.vinc1
 slr=vinc0:SLR1
 sp=vinc0.axi4_0:DDR[1]

[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
```

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

1.4 Запуск программного обеспечения на хост-системе

Ниже, в листинге 1.3 приведены измененные части кода модифицированного модуля host_example.cpp.

Листинг 1.3 — Модуль host_example.cpp

```
1    ...
2     for (cl_uint i = 0; i < number_of_words; i++) {
3         if ((h_data[i]/16 - 11) != h_axi00_ptr0_output[i]) {
4             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]);
5             check_status = 1;
6         }
7         //  printf("i=%d, input=%d, output=%d\n", i,
             h_axi00_ptr0_input[i], h_axi00_ptr0_output[i]);
8     }
9     ...
10 } // end of main
```

Ниже на рисунке 1.8 приведены результаты тестирования.

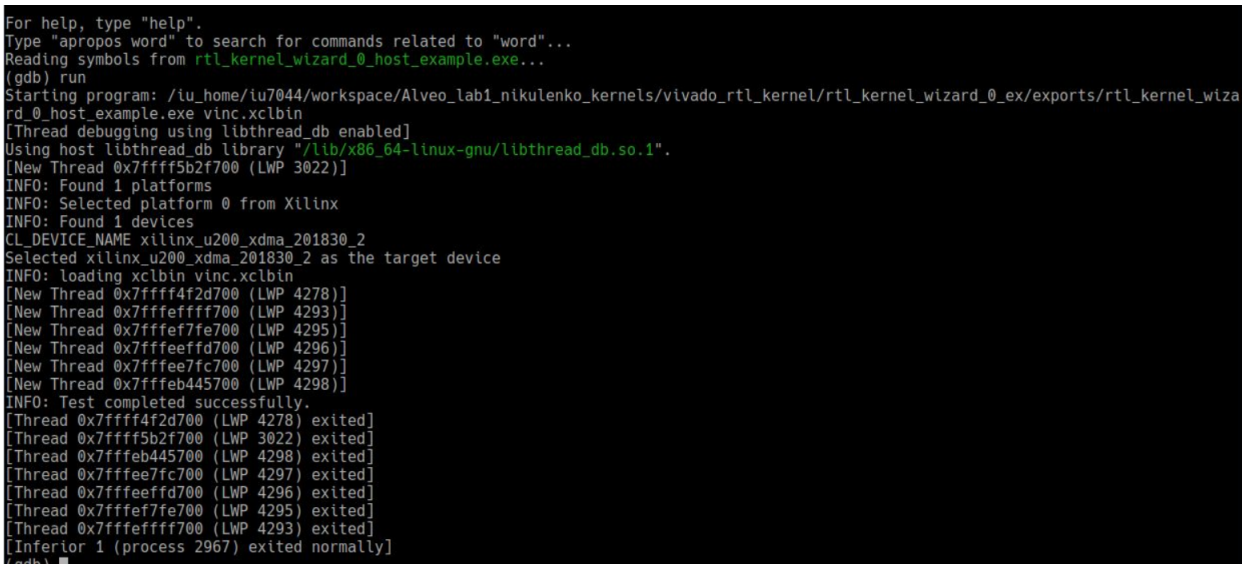

Рисунок 1.8 — Результаты тестирования

Ответы на контрольные вопросы

1. Преимущества и недостатки XDMA и QDMA платформ

Недостатки использовани XDMA:

- бОльшая латентность и меньшая пропускная способность за счет того, что данные сначало должны быть перемещены в память ускорителя.

Преимущества использования QDMA:

- предоставляет прямое потоковое соединение с низкой задержкой и большой пропускной способностью между хостом и ядрами;

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

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

1. С помощью вызова clGetPlatformIDs хост получает все платформы.

2. С помощью вызова clGetPlatformInfo хост получает имя платформы и затем выбирает платформу Xilinx.

3. С помощью вызова clGetDeviceIDs хост получает ID устройства.

4. С помощью вызова clGetDeviceInfo хост получает информацию об устройстве.

5. С помощью вызова clCreateContext создается контекст для переменных.

6. С помощью вызова clCreateCommandQueue создается команда для устройтво-ускорителя.

3. Процедура запуска задания на исполнения в ускорительном ядре VINC

1. С помощью вызова load_file_to_memory (см. приложение А) данные, бинарный поток (данные из *.xclbin), копируются из ОЗУ в локальную память ускорителя посредством DMA.

2. По итогу выполнения clCreateProgramWithBinary, clBuildProgram и clCreateKernel создается исполняемый файл (уже в памяти устройства-ускорителя).

3. С помощью clCreateBuffer и clEnqueueWriteBuffer данные, подлежащие обработке, копируются из ОЗУ в локальную память ускорителя посредством DMA (с помощью второй команды осуществляется передача указателей на начало буферов исходных операндов).

4. С помощью двух вызовов clSetKernelArg указываются параметры (в данном случае это d_scalar00 и d_axi00_ptr0).

5. С помощью команды clEnqueueNDRangeKernel запускается исполнение ядра (программы на ускорителе).

6. С помощью команды clEnqueueReadBuffer выполняется чтение готовых данных.

4. Процесс линковки на основании содержимого log-файла

Процесс сборки состоит из шести этапов (фаз).

1. Анализ конфигурационного файла, анализ профиля устройства, поиск необходимых аппаратных компонентов, интерфейсов;

2. FPGA linking synthesized kernels to platform (Block-level synthesis, Top-level synthesis);

3. FPGA logic optimization (минимизация логики (булевой) для оптимизации площади, минимизации задержек);

4. FPGA logic placement (Преобразование булевых уравнений в схему логики ПЛИС. Выбор конкретного места для каждого логического блока в ПЛИС);

5. FPGA routing (разводка [создание соединений между логическими блоками]);

6. FPGA bitstream generation (генерирование файла с программной информацией для отправки его на ПЛИС [*.xclbin файл]);

ПРИЛОЖЕНИЕ А

СОДЕРЖИМОЕ ФАЙЛА HOST_EXAMPLE.CPP

Листинг А.1 — Содержимое файла host_example.cpp

```
// This is a generated file. Use and modify at your own risk.
////////////////////////////////////////////////////////////////////////////////

/*******************************************************************************
Vendor: Xilinx
Associated Filename: main.c
#Purpose: This example shows a basic vector add +1 (constant) by manipulating
#         memory inplace.
*******************************************************************************/

#include <fcntl.h>
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#ifdef _WINDOWS
#include <io.h>
#else
#include <unistd.h>
#include <sys/time.h>
#endif
#include <assert.h>
#include <stdbool.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <CL/opencl.h>
#include <CL/cl_ext.h>
#include "xclhal2.h"

////////////////////////////////////////////////////////////////////////////////

#define NUM_WORKGROUPS (1)
#define WORKGROUP_SIZE (256)
#define MAX_LENGTH 8192
#define MEM_ALIGNMENT 4096
#if defined(VITIS_PLATFORM) && !defined(TARGET_DEVICE)
#define STR_VALUE(arg)      #arg
#define GET_STRING(name) STR_VALUE(name)
#define TARGET_DEVICE GET_STRING(VITIS_PLATFORM)
```

```
#endif

////////////////////////////////////////////////////////////////////////////////

cl_uint load_file_to_memory(const char *filename, char **result)
{
    cl_uint size = 0;
    FILE *f = fopen(filename, "rb");
    if (f == NULL) {
        *result = NULL;
        return -1; // -1 means file opening fail
    }
    fseek(f, 0, SEEK_END);
    size = ftell(f);
    fseek(f, 0, SEEK_SET);
    *result = (char *)malloc(size+1);
    if (size != fread(*result, sizeof(char), size, f)) {
        free(*result);
        return -2; // -2 means file reading fail
    }
    fclose(f);
    (*result)[size] = 0;
    return size;
}

int main(int argc, char** argv)
{

    cl_int err;                             // error code returned from api calls
    cl_uint check_status = 0;
    const cl_uint number_of_words = 4096; // 16KB of data


    cl_platform_id platform_id;          // platform id
    cl_device_id device_id;              // compute device id
    cl_context context;                  // compute context
    cl_command_queue commands;           // compute command queue
    cl_program program;                  // compute programs
    cl_kernel kernel;                    // compute kernel

    cl_uint* h_data;                                // host memory for input vector
    char cl_platform_vendor[1001];
    char target_device_name[1001] = TARGET_DEVICE;

```

```
    cl_uint* h_axi00_ptr0_output = (cl_uint*)aligned_alloc(MEM_ALIGNMENT,MAX_LENGTH * sizeof(cl_uint*)); // host memory for output vector
    cl_mem d_axi00_ptr0;                          // device memory used for a vector

    if (argc != 2) {
        printf("Usage: %s xclbin\n", argv[0]);
        return EXIT_FAILURE;
    }

    // Fill our data sets with pattern
    h_data = (cl_uint*)aligned_alloc(MEM_ALIGNMENT,MAX_LENGTH * sizeof(cl_uint*));
    for(cl_uint i = 0; i < MAX_LENGTH; i++) {
        h_data[i]  = i;
        h_axi00_ptr0_output[i] = 0;

    }

    // Get all platforms and then select Xilinx platform
    cl_platform_id platforms[16];       // platform id
    cl_uint platform_count;
    cl_uint platform_found = 0;
    err = clGetPlatformIDs(16, platforms, &platform_count);
    if (err != CL_SUCCESS) {
        printf("ERROR: Failed to find an OpenCL platform!\n");
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    }
    printf("INFO: Found %d platforms\n", platform_count);

    // Find Xilinx Plaftorm
    for (cl_uint iplat=0; iplat<platform_count; iplat++) {
        err = clGetPlatformInfo(platforms[iplat], CL_PLATFORM_VENDOR, 1000, (void *)cl_platform_vendor,NULL);
        if (err != CL_SUCCESS) {
            printf("ERROR: clGetPlatformInfo(CL_PLATFORM_VENDOR) failed!\n");
            printf("ERROR: Test failed\n");
            return EXIT_FAILURE;
        }
        if (strcmp(cl_platform_vendor, "Xilinx") == 0) {
            printf("INFO: Selected platform %d from %s\n", iplat, cl_platform_vendor);
            platform_id = platforms[iplat];
```

```
            platform_found = 1;
        }
    }
    if (!platform_found) {
        printf("ERROR: Platform Xilinx not found. Exit.\n");
        return EXIT_FAILURE;
    }

    // Get Accelerator compute device
    cl_uint num_devices;
    cl_uint device_found = 0;
    cl_device_id devices[16];  // compute device id
    char cl_device_name[1001];
    err = clGetDeviceIDs(platform_id, CL_DEVICE_TYPE_ACCELERATOR, 16,
        devices, &num_devices);
    printf("INFO: Found %d devices\n", num_devices);
    if (err != CL_SUCCESS) {
        printf("ERROR: Failed to create a device group!\n");
        printf("ERROR: Test failed\n");
        return -1;
    }

    //iterate all devices to select the target device.
    for (cl_uint i=0; i<num_devices; i++) {
        err = clGetDeviceInfo(devices[i], CL_DEVICE_NAME, 1024,
            cl_device_name, 0);
        if (err != CL_SUCCESS) {
            printf("ERROR: Failed to get device name for device %d!\n", i);
            printf("ERROR: Test failed\n");
            return EXIT_FAILURE;
        }
        printf("CL_DEVICE_NAME %s\n", cl_device_name);
        if(strcmp(cl_device_name, target_device_name) == 0) {
            device_id = devices[i];
            device_found = 1;
            printf("Selected %s as the target device\n", cl_device_name);
        }
    }

    if (!device_found) {
        printf("ERROR:Target device %s not found. Exit.\n",
            target_device_name);
        return EXIT_FAILURE;
    }

    // Create a compute context
```

```
    //
    context = clCreateContext(0, 1, &device_id, NULL, NULL, &err);
    if (!context) {
        printf("ERROR: Failed to create a compute context!\n");
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    }

    // Create a command commands
    commands = clCreateCommandQueue(context, device_id,
        CL_QUEUE_PROFILING_ENABLE | CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE,
        &err);
    if (!commands) {
        printf("ERROR: Failed to create a command commands!\n");
        printf("ERROR: code %i\n",err);
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    }

    cl_int status;

    // Create Program Objects
    // Load binary from disk
    unsigned char *kernelbinary;
    char *xclbin = argv[1];

    //----------------------------------------------------------------------------
    // xclbin
    //----------------------------------------------------------------------------
    printf("INFO: loading xclbin %s\n", xclbin);
    cl_uint n_i0 = load_file_to_memory(xclbin, (char **) &kernelbinary);
    if (n_i0 < 0) {
        printf("ERROR: failed to load kernel from xclbin: %s\n", xclbin);
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    }

    size_t n0 = n_i0;

    // Create the compute program from offline
    program = clCreateProgramWithBinary(context, 1, &device_id, &n0,
                                        (const unsigned char **)
                                            &kernelbinary, &status, &err);
    free(kernelbinary);

    if ((!program) || (err!=CL_SUCCESS)) {
```

```
        printf("ERROR: Failed to create compute program from binary %d!\n", err);
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    }


    // Build the program executable
    //
    err = clBuildProgram(program, 0, NULL, NULL, NULL, NULL);
    if (err != CL_SUCCESS) {
        size_t len;
        char buffer[2048];

        printf("ERROR: Failed to build program executable!\n");
        clGetProgramBuildInfo(program, device_id, CL_PROGRAM_BUILD_LOG, sizeof(buffer), buffer, &len);
        printf("%s\n", buffer);
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    }

    // Create the compute kernel in the program we wish to run
    //
    kernel = clCreateKernel(program, "rtl_kernel_wizard_0", &err);
    if (!kernel || err != CL_SUCCESS) {
        printf("ERROR: Failed to create compute kernel!\n");
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    }

    // Create structs to define memory bank mapping
    cl_mem_ext_ptr_t mem_ext;
    mem_ext.obj = NULL;
    mem_ext.param = kernel;


    mem_ext.flags = 1;
    d_axi00_ptr0 = clCreateBuffer(context, CL_MEM_READ_WRITE | CL_MEM_EXT_PTR_XILINX, sizeof(cl_uint) * number_of_words, &mem_ext, &err);
    if (err != CL_SUCCESS) {
      std::cout << "Return code for clCreateBuffer flags=" << mem_ext.flags << ": " << err << std::endl;
    }

```

```
    if (!(d_axi00_ptr0)) {
        printf("ERROR: Failed to allocate device memory!\n");
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    }


    err = clEnqueueWriteBuffer(commands, d_axi00_ptr0, CL_TRUE, 0, sizeof(cl_uint) * number_of_words, h_data, 0, NULL, NULL);
    if (err != CL_SUCCESS) {
        printf("ERROR: Failed to write to source array h_data: d_axi00_ptr0: %d!\n", err);
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    }


    // Set the arguments to our compute kernel
    // cl_uint vector_length = MAX_LENGTH;
    err = 0;
    cl_uint d_scalar00 = 0;
    err |= clSetKernelArg(kernel, 0, sizeof(cl_uint), &d_scalar00); // Not used in example RTL logic.
    err |= clSetKernelArg(kernel, 1, sizeof(cl_mem), &d_axi00_ptr0);

    if (err != CL_SUCCESS) {
        printf("ERROR: Failed to set kernel arguments! %d\n", err);
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    }

    size_t global[1];
    size_t local[1];
    // Execute the kernel over the entire range of our 1d input data set
    // using the maximum number of work group items for this device

    global[0] = 1;
    local[0] = 1;
    err = clEnqueueNDRangeKernel(commands, kernel, 1, NULL, (size_t*)&global, (size_t*)&local, 0, NULL, NULL);
    if (err) {
        printf("ERROR: Failed to execute kernel! %d\n", err);
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    }
```

```
    clFinish(commands);


    // Read back the results from the device to verify the output
    //
    cl_event readevent;

    err = 0;
    err |= clEnqueueReadBuffer( commands, d_axi00_ptr0, CL_TRUE, 0,
        sizeof(cl_uint) * number_of_words, h_axi00_ptr0_output, 0, NULL,
        &readevent );


    if (err != CL_SUCCESS) {
        printf("ERROR: Failed to read output array! %d\n", err);
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    }
    clWaitForEvents(1, &readevent);
    // Check Results

    for (cl_uint i = 0; i < number_of_words; i++) {
        if ((h_data[i]/16 - 11) != 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]);
    }


    //------------------------------------------------------------------------
    // Shutdown and cleanup
    //------------------------------------------------------------------------
    clReleaseMemObject(d_axi00_ptr0);
    free(h_axi00_ptr0_output);



    free(h_data);
    clReleaseProgram(program);
    clReleaseKernel(kernel);
```

```
    clReleaseCommandQueue(commands);
    clReleaseContext(context);

    if (check_status) {
        printf("ERROR: Test failed\n");
        return EXIT_FAILURE;
    } else {
        printf("INFO: Test completed successfully.\n");
        return EXIT_SUCCESS;
    }


} // end of main
```

ПРИЛОЖЕНИЕ Б

СОДЕРЖИМОЕ LOG-ФАЙЛА

Листинг Б.1 — Содержимое log-файла

```
INFO: [v++ 60-1306] Additional information associated with this v++ link can be found at:
    Reports: /iu_home/iu7044/_x/reports/link
    Log files: /iu_home/iu7044/_x/logs/link
INFO: [v++ 60-1548] Creating build summary session with primary output /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/vivado_rtl_kernel/rtl_kernel at Thu Dec  9 01:02:14 2021
INFO: [v++ 60-1316] Initiating connection to rulecheck server, at Thu Dec 9 01:02:14 2021
INFO: [v++ 60-1315] Creating rulecheck session with output '/iu_home/iu7044/_x/reports/link/v++_link_vinc_guidance.html', at Thu Dec  9 01:02:33 2021
INFO: [v++ 60-895]   Target platform: /opt/xilinx/platforms/xilinx_u200_xdma_201830_2/xilinx_u200_xdma_201830_2.xpfm
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'
INFO: [v++ 74-74] Compiler Version string: 2020.2
INFO: [v++ 60-1302] Platform 'xilinx_u200_xdma_201830_2.xpfm' has been explicitly enabled for this release.
INFO: [v++ 60-629] Linking for hardware target
INFO: [v++ 60-423]   Target device: xilinx_u200_xdma_201830_2
INFO: [v++ 60-1332] Run 'run_link' status: Not started
INFO: [v++ 60-1443] [01:03:28] Run run_link: Step system_link: Started
INFO: [v++ 60-1453] Command Line: system_link --xo /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/src/vitis_rtl_kernel/rtl_ker --config /iu_home/iu7044/_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/iu7044/_x/link/int --temp_dir /iu_home/iu7044/_x/link/sys_link
INFO: [v++ 60-1454] Run Directory: /iu_home/iu7044/_x/link/run_link
INFO: [SYSTEM_LINK 60-1316] Initiating connection to rulecheck server, at Thu Dec  9 01:03:43 2021
INFO: [SYSTEM_LINK 82-70] Extracting xo v3 file /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/src/vitis_rtl_kernel/rtl_ker
INFO: [SYSTEM_LINK 82-53] Creating IP database /iu_home/iu7044/_x/link/sys_link/_sysl/.cdb/xd_ip_db.xml
INFO: [SYSTEM_LINK 82-38] [01:03:45] build_xd_ip_db started: /data/Xilinx/Vitis/2020.2/bin/build_xd_ip_db -ip_search 0  -sds-pf /iu_home/iu7044/_x/link/sys_link/xilinx_u200_xdma_201830_2.hpfm -clkid 0 -ip
```

```
/iu_home/iu7044/_x/link/sys_link/iprepo/mycompany_com_kernel_rtl_kernel_wizard_0_1_
-o /iu_home/iu7044/_x/link/sys_link/_sysl/.cdb/xd_ip_db.xml
INFO: [SYSTEM_LINK 82-37] [01:04:19] build_xd_ip_db finished successfully
Time (s): cpu = 00:00:35 ; elapsed = 00:00:34 . Memory (MB): peak =
1557.891 ; gain = 0.000 ; free physical = 54402 ; free virtual = 215693
INFO: [SYSTEM_LINK 82-51] Create system connectivity graph
INFO: [SYSTEM_LINK 82-102] Applying explicit connections to the system
connectivity graph:
/iu_home/iu7044/_x/link/sys_link/cfgraph/cfgen_cfgraph.xml
INFO: [SYSTEM_LINK 82-38] [01:04:19] cfgen started:
/data/Xilinx/Vitis/2020.2/bin/cfgen -nk rtl_kernel_wizard_0:1:vinc0
-slr vinc0:SLR1 -sp vinc0.m00_axi:DDR[1] -dmclkid 0 -r
/iu_home/iu7044/_x/link/sys_link/_sysl/.cdb/xd_ip_db.xml -o
/iu_home/iu7044/_x/link/sys_link/cfgraph/cfgen_cfgraph.xml
INFO: [CFGEN 83-0] Kernel Specs:
INFO: [CFGEN 83-0]   kernel: rtl_kernel_wizard_0, num: 1  {vinc0}
INFO: [CFGEN 83-0] Port Specs:
INFO: [CFGEN 83-0]   kernel: vinc0, k_port: m00_axi, sptag: DDR[1]
INFO: [CFGEN 83-0] SLR Specs:
INFO: [CFGEN 83-0]   instance: vinc0, SLR: SLR1
INFO: [CFGEN 83-2228] Creating mapping for argument vinc0.axi00_ptr0 to
DDR[1] for directive vinc0.m00_axi:DDR[1]
INFO: [SYSTEM_LINK 82-37] [01:04:49] cfgen finished successfully
Time (s): cpu = 00:00:29 ; elapsed = 00:00:30 . Memory (MB): peak =
1557.891 ; gain = 0.000 ; free physical = 57689 ; free virtual = 217477
INFO: [SYSTEM_LINK 82-52] Create top-level block diagram
INFO: [SYSTEM_LINK 82-38] [01:04:49] cf2bd started:
/data/Xilinx/Vitis/2020.2/bin/cf2bd --linux --trace_buffer 1024
--input_file /iu_home/iu7044/_x/link/sys_link/cfgraph/cfgen_cfgraph.xml
--ip_db /iu_home/iu7044/_x/link/sys_link/_sysl/.cdb/xd_ip_db.xml
--cf_name dr --working_dir /iu_home/iu7044/_x/link/sys_link/_sysl/.xsd
--temp_dir /iu_home/iu7044/_x/link/sys_link --output_dir
/iu_home/iu7044/_x/link/int --target_bd pfm_dynamic.bd
INFO: [CF2BD 82-31] Launching cf2xd: cf2xd -linux -trace-buffer 1024 -i
/iu_home/iu7044/_x/link/sys_link/cfgraph/cfgen_cfgraph.xml -r
/iu_home/iu7044/_x/link/sys_link/_sysl/.cdb/xd_ip_db.xml -o dr.xml
INFO: [CF2BD 82-28] cf2xd finished successfully
INFO: [CF2BD 82-31] Launching cf_xsd: cf_xsd -disable-address-gen -bd
pfm_dynamic.bd -dn dr -dp /iu_home/iu7044/_x/link/sys_link/_sysl/.xsd
INFO: [CF2BD 82-28] cf_xsd finished successfully
INFO: [SYSTEM_LINK 82-37] [01:05:06] cf2bd finished successfully
Time (s): cpu = 00:00:14 ; elapsed = 00:00:17 . Memory (MB): peak =
1557.891 ; gain = 0.000 ; free physical = 57560 ; free virtual = 217396
INFO: [v++ 60-1441] [01:05:07] Run run_link: Step system_link: Completed
Time (s): cpu = 00:01:34 ; elapsed = 00:01:38 . Memory (MB): peak =
1585.129 ; gain = 0.000 ; free physical = 57576 ; free virtual = 217411
```

```
INFO: [v++ 60-1443] [01:05:07] Run run_link: Step cf2sw: Started
INFO: [v++ 60-1453] Command Line: cf2sw -sdsl
    /iu_home/iu7044/_x/link/int/sdsl.dat -rtd
    /iu_home/iu7044/_x/link/int/cf2sw.rtd -nofilter
    /iu_home/iu7044/_x/link/int/cf2sw_full.rtd -xclbin
    /iu_home/iu7044/_x/link/int/xclbin_orig.xml -o
    /iu_home/iu7044/_x/link/int/xclbin_orig.1.xml
INFO: [v++ 60-1454] Run Directory: /iu_home/iu7044/_x/link/run_link
INFO: [v++ 60-1441] [01:05:25] Run run_link: Step cf2sw: Completed
Time (s): cpu = 00:00:17 ; elapsed = 00:00:18 . Memory (MB): peak =
    1585.129 ; gain = 0.000 ; free physical = 58137 ; free virtual = 217404
INFO: [v++ 60-1443] [01:05:25] Run run_link: Step rtd2_system_diagram:
    Started
INFO: [v++ 60-1453] Command Line: rtd2SystemDiagram
INFO: [v++ 60-1454] Run Directory: /iu_home/iu7044/_x/link/run_link
INFO: [v++ 60-1441] [01:05:36] Run run_link: Step rtd2_system_diagram:
    Completed
Time (s): cpu = 00:00:00.02 ; elapsed = 00:00:11 . Memory (MB): peak =
    1585.129 ; gain = 0.000 ; free physical = 57544 ; free virtual = 216815
INFO: [v++ 60-1443] [01:05:36] Run run_link: Step vpl: Started
INFO: [v++ 60-1453] Command Line: vpl -t hw -f xilinx_u200_xdma_201830_2
    --remote_ip_cache /iu_home/iu7044/.ipcache --output_dir
    /iu_home/iu7044/_x/link/int --log_dir /iu_home/iu7044/_x/logs/link
    --report_dir /iu_home/iu7044/_x/reports/link --config
    /iu_home/iu7044/_x/link/int/vplConfig.ini -k
    /iu_home/iu7044/_x/link/int/kernel_info.dat --webtalk_flag Vitis
    --temp_dir /iu_home/iu7044/_x/link --no-info --iprepo
    /iu_home/iu7044/_x/link/int/xo/ip_repo/mycompany_com_kernel_rtl_kernel_wizard_0_1_
    --messageDb /iu_home/iu7044/_x/link/run_link/vpl.pb
    /iu_home/iu7044/_x/link/int/dr.bd.tcl
INFO: [v++ 60-1454] Run Directory: /iu_home/iu7044/_x/link/run_link

****** vpl v2020.2 (64-bit)
  **** SW Build (by xbuild) on 2020-11-18-05:13:29
    ** Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.

INFO: [VPL 60-839] Read in kernel information from file
    '/iu_home/iu7044/_x/link/int/kernel_info.dat'.
INFO: [VPL 74-74] Compiler Version string: 2020.2
INFO: [VPL 60-423]   Target device: xilinx_u200_xdma_201830_2
INFO: [VPL 60-1032] Extracting hardware platform to
    /iu_home/iu7044/_x/link/vivado/vpl/.local/hw_platform
WARNING: /data/Xilinx/Vitis/2020.2/tps/lnx64/jre9.0.4 does not exist.
[01:11:18] Run vpl: Step create_project: RUNNING...
[01:11:18] Run vpl: Step create_project: Started
Creating Vivado project.
```

```
[01:11:52] Run vpl: Step create_project: Completed
[01:11:52] Run vpl: Step create_bd: Started
[01:13:35] Run vpl: Step create_bd: RUNNING...
[01:15:15] Run vpl: Step create_bd: RUNNING...
[01:16:50] Run vpl: Step create_bd: RUNNING...
[01:18:48] Run vpl: Step create_bd: RUNNING...
[01:20:25] Run vpl: Step create_bd: RUNNING...
[01:22:00] Run vpl: Step create_bd: Completed
[01:22:00] Run vpl: Step update_bd: Started
[01:22:02] Run vpl: Step create_bd: RUNNING...
[01:22:04] Run vpl: Step update_bd: Completed
[01:22:04] Run vpl: Step generate_target: Started
[01:23:40] Run vpl: Step generate_target: RUNNING...
[01:25:17] Run vpl: Step generate_target: RUNNING...
[01:26:46] Run vpl: Step generate_target: RUNNING...
[01:28:33] Run vpl: Step generate_target: RUNNING...
[01:30:05] Run vpl: Step generate_target: RUNNING...
[01:31:20] Run vpl: Step generate_target: Completed
[01:31:20] Run vpl: Step config_hw_runs: Started
[01:31:39] Run vpl: Step config_hw_runs: Completed
[01:31:39] Run vpl: Step synth: Started
[01:33:53] Top-level synthesis in progress.
[01:34:30] Top-level synthesis in progress.
[01:35:08] Top-level synthesis in progress.
[01:35:43] Top-level synthesis in progress.
[01:36:21] Top-level synthesis in progress.
[01:37:02] Top-level synthesis in progress.
[01:37:42] Top-level synthesis in progress.
[01:38:20] Top-level synthesis in progress.
[01:38:58] Top-level synthesis in progress.
[01:39:34] Top-level synthesis in progress.
[01:40:11] Top-level synthesis in progress.
[01:40:48] Top-level synthesis in progress.
[01:41:29] Top-level synthesis in progress.
[01:42:06] Top-level synthesis in progress.
[01:42:44] Top-level synthesis in progress.
[01:43:24] Top-level synthesis in progress.
[01:44:04] Top-level synthesis in progress.
[01:44:41] Run vpl: Step synth: Completed
[01:44:41] Run vpl: Step impl: Started
[02:44:32] Finished 2nd of 6 tasks (FPGA linking synthesized kernels to
    platform). Elapsed time: 01h 38m 41s

[02:44:32] Starting logic optimization..
[02:54:58] Phase 1 Retarget
[02:57:35] Phase 2 Constant propagation
```

```
[02:59:26] Phase 3 Sweep
[03:04:26] Phase 4 BUFG optimization
[03:06:18] Phase 5 Shift Register Optimization
[03:06:54] Phase 6 Post Processing Netlist
[03:22:09] Finished 3rd of 6 tasks (FPGA logic optimization). Elapsed time: 00h 37m 37s

[03:22:09] Starting logic placement..
[03:27:13] Phase 1 Placer Initialization
[03:27:13] Phase 1.1 Placer Initialization Netlist Sorting
[03:42:49] Phase 1.2 IO Placement/ Clock Placement/ Build Placer Device
[03:51:42] Phase 1.3 Build Placer Netlist Model
[04:05:09] Phase 1.4 Constrain Clocks/Macros
[04:06:24] Phase 2 Global Placement
[04:06:24] Phase 2.1 Floorplanning
[04:09:28] Phase 2.1.1 Partition Driven Placement
[04:09:28] Phase 2.1.1.1 PBP: Partition Driven Placement
[04:10:04] Phase 2.1.1.2 PBP: Clock Region Placement
[04:14:23] Phase 2.1.1.3 PBP: Compute Congestion
[04:15:02] Phase 2.1.1.4 PBP: UpdateTiming
[04:17:33] Phase 2.1.1.5 PBP: Add part constraints
[04:18:10] Phase 2.2 Update Timing before SLR Path Opt
[04:18:48] Phase 2.3 Global Placement Core
[04:43:32] Phase 2.3.1 Physical Synthesis In Placer
[04:55:06] Phase 3 Detail Placement
[04:55:06] Phase 3.1 Commit Multi Column Macros
[04:55:43] Phase 3.2 Commit Most Macros & LUTRAMs
[05:00:44] Phase 3.3 Small Shape DP
[05:00:44] Phase 3.3.1 Small Shape Clustering
[05:02:02] Phase 3.3.2 Flow Legalize Slice Clusters
[05:02:42] Phase 3.3.3 Slice Area Swap
[05:07:57] Phase 3.4 Place Remaining
[05:08:36] Phase 3.5 Re-assign LUT pins
[05:09:56] Phase 3.6 Pipeline Register Optimization
[05:09:56] Phase 3.7 Fast Optimization
[05:14:23] Phase 4 Post Placement Optimization and Clean-Up
[05:14:23] Phase 4.1 Post Commit Optimization
[05:23:51] Phase 4.1.1 Post Placement Optimization
[05:24:33] Phase 4.1.1.1 BUFG Insertion
[05:24:33] Phase 1 Physical Synthesis Initialization
[05:27:06] Phase 4.1.1.2 BUFG Replication
[05:29:42] Phase 4.1.1.3 Replication
[05:35:54] Phase 4.2 Post Placement Cleanup
[05:36:30] Phase 4.3 Placer Reporting
[05:36:30] Phase 4.3.1 Print Estimated Congestion
[05:38:21] Phase 4.4 Final Placement Cleanup
```

```
[06:49:48] Finished 4th of 6 tasks (FPGA logic placement). Elapsed time: 03h 27m 39s

[06:49:48] Starting logic routing..
[06:56:54] Phase 1 Build RT Design
[07:08:39] Phase 2 Router Initialization
[07:08:39] Phase 2.1 Fix Topology Constraints
[07:08:39] Phase 2.2 Pre Route Cleanup
[07:09:19] Phase 2.3 Global Clock Net Routing
[07:12:32] Phase 2.4 Update Timing
[07:26:39] Phase 2.5 Update Timing for Bus Skew
[07:26:39] Phase 2.5.1 Update Timing
[07:31:49] Phase 3 Initial Routing
[07:31:49] Phase 3.1 Global Routing
[07:36:54] Phase 4 Rip-up And Reroute
[07:36:54] Phase 4.1 Global Iteration 0
[07:56:15] Phase 4.2 Global Iteration 1
[08:02:06] Phase 4.3 Global Iteration 2
[08:06:35] Phase 4.4 Global Iteration 3
[08:09:43] Phase 4.5 Global Iteration 4
[08:12:18] Phase 5 Delay and Skew Optimization
[08:12:18] Phase 5.1 Delay CleanUp
[08:12:18] Phase 5.2 Clock Skew Optimization
[08:12:58] Phase 6 Post Hold Fix
[08:12:58] Phase 6.1 Hold Fix Iter
[08:12:58] Phase 6.1.1 Update Timing
[08:20:01] Phase 7 Route finalize
[08:20:38] Phase 8 Verifying routed nets
[08:21:54] Phase 9 Depositing Routes
[08:25:44] Phase 10 Route finalize
[08:26:22] Phase 11 Post Router Timing
[08:33:16] Finished 5th of 6 tasks (FPGA routing). Elapsed time: 01h 43m 28s

[08:33:16] Starting bitstream generation..
[10:36:50] Creating bitmap...
[11:22:30] Writing bitstream ./pfm_top_i_dynamic_region_my_rm_partial.bit...
[11:22:30] Finished 6th of 6 tasks (FPGA bitstream generation). Elapsed time: 02h 49m 13s
[11:25:41] Run vpl: Step impl: Completed
[11:25:51] Run vpl: FINISHED. Run Status: impl Complete!
INFO: [v++ 60-1441] [11:26:20] Run run_link: Step vpl: Completed
Time (s): cpu = 00:19:54 ; elapsed = 10:20:43 . Memory (MB): peak = 1585.129 ; gain = 0.000 ; free physical = 69387 ; free virtual = 228013
INFO: [v++ 60-1443] [11:26:20] Run run_link: Step rtdgen: Started
```

```
INFO: [v++ 60-1453] Command Line: rtdgen
INFO: [v++ 60-1454] Run Directory: /iu_home/iu7044/_x/link/run_link
INFO: [v++ 60-991] clock name 'clkwiz_kernel_clk_out1' (clock ID '0') is being mapped to clock name 'DATA_CLK' in the xclbin
INFO: [v++ 60-991] clock name 'clkwiz_kernel2_clk_out1' (clock ID '1') is being mapped to clock name 'KERNEL_CLK' in the xclbin
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
INFO: [v++ 60-1453] Command Line: cf2sw -a /iu_home/iu7044/_x/link/int/address_map.xml -sdsl /iu_home/iu7044/_x/link/int/sdsl.dat -xclbin /iu_home/iu7044/_x/link/int/xclbin_orig.xml -rtd /iu_home/iu7044/_x/link/int/vinc.rtd -o /iu_home/iu7044/_x/link/int/vinc.xml
INFO: [v++ 60-1652] Cf2sw returned exit code: 0
INFO: [v++ 60-2311] HPISystemDiagram::writeSystemDiagramAfterRunningVivado, rtdInputFilePath: /iu_home/iu7044/_x/link/int/vinc.rtd
INFO: [v++ 60-2312] HPISystemDiagram::writeSystemDiagramAfterRunningVivado, systemDiagramOutputFilePath: /iu_home/iu7044/_x/link/int/systemDiagramModelSlrBaseAddress.json
INFO: [v++ 60-1618] Launching
INFO: [v++ 60-1441] [11:26:35] Run run_link: Step rtdgen: Completed
Time (s): cpu = 00:00:14 ; elapsed = 00:00:15 . Memory (MB): peak = 1585.129 ; gain = 0.000 ; free physical = 69376 ; free virtual = 228002
INFO: [v++ 60-1443] [11:26:35] Run run_link: Step xclbinutil: Started
INFO: [v++ 60-1453] Command Line: xclbinutil --add-section DEBUG_IP_LAYOUT:JSON:/iu_home/iu7044/_x/link/int/debug_ip_layout.rtd --add-section BITSTREAM:RAW:/iu_home/iu7044/_x/link/int/partial.bit --force --target hw --key-value SYS:dfx_enable:true --add-section :JSON:/iu_home/iu7044/_x/link/int/vinc.rtd --append-section :JSON:/iu_home/iu7044/_x/link/int/appendSection.rtd --add-section CLOCK_FREQ_TOPOLOGY:JSON:/iu_home/iu7044/_x/link/int/vinc_xml.rtd --add-section BUILD_METADATA:JSON:/iu_home/iu7044/_x/link/int/vinc_build.rtd --add-section EMBEDDED_METADATA:RAW:/iu_home/iu7044/_x/link/int/vinc.xml --add-section SYSTEM_METADATA:RAW:/iu_home/iu7044/_x/link/int/systemDiagramModelSlrBaseAddress.js --output /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/vivado_rtl_kernel/rtl_kernel
INFO: [v++ 60-1454] Run Directory: /iu_home/iu7044/_x/link/run_link
XRT Build Version: 2.8.743 (2020.2)
```

```
        Build Date: 2020-11-16 00:19:11
           Hash ID: 77d5484b5c4daa691a7f78235053fb036829b1e9
Creating a default 'in-memory' xclbin image.

Section: 'DEBUG_IP_LAYOUT'(9) was successfully added.
Size   : 440 bytes
Format : JSON
File   : '/iu_home/iu7044/_x/link/int/debug_ip_layout.rtd'

Section: 'BITSTREAM'(0) was successfully added.
Size   : 39676274 bytes
Format : RAW
File   : '/iu_home/iu7044/_x/link/int/partial.bit'

Section: 'MEM_TOPOLOGY'(6) was successfully added.
Format : JSON
File   : 'mem_topology'

Section: 'IP_LAYOUT'(8) was successfully added.
Format : JSON
File   : 'ip_layout'

Section: 'CONNECTIVITY'(7) was successfully added.
Format : JSON
File   : 'connectivity'

Section: 'CLOCK_FREQ_TOPOLOGY'(11) was successfully added.
Size   : 274 bytes
Format : JSON
File   : '/iu_home/iu7044/_x/link/int/vinc_xml.rtd'

Section: 'BUILD_METADATA'(14) was successfully added.
Size   : 3152 bytes
Format : JSON
File   : '/iu_home/iu7044/_x/link/int/vinc_build.rtd'

Section: 'EMBEDDED_METADATA'(2) was successfully added.
Size   : 2759 bytes
Format : RAW
File   : '/iu_home/iu7044/_x/link/int/vinc.xml'

Section: 'SYSTEM_METADATA'(22) was successfully added.
Size   : 5853 bytes
Format : RAW
File   :
    '/iu_home/iu7044/_x/link/int/systemDiagramModelSlrBaseAddress.json'
```

```

Section: 'IP_LAYOUT'(8) was successfully appended to.
Format : JSON
File   : 'ip_layout'
Successfully wrote (39698905 bytes) to the output file: /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/vivado_rtl_kernel/rtl_kernel
Leaving xclbinutil.
INFO: [v++ 60-1441] [11:26:37] Run run_link: Step xclbinutil: Completed
Time (s): cpu = 00:00:00.60 ; elapsed = 00:00:02 . Memory (MB): peak = 1585.129 ; gain = 0.000 ; free physical = 69334 ; free virtual = 227998
INFO: [v++ 60-1443] [11:26:37] Run run_link: Step xclbinutilinfo: Started
INFO: [v++ 60-1453] Command Line: xclbinutil --quiet --force --info /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/vivado_rtl_kernel/rtl_kernel --input /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/vivado_rtl_kernel/rtl_kernel
INFO: [v++ 60-1454] Run Directory: /iu_home/iu7044/_x/link/run_link
INFO: [v++ 60-1441] [11:26:41] Run run_link: Step xclbinutilinfo: Completed
Time (s): cpu = 00:00:03 ; elapsed = 00:00:03 . Memory (MB): peak = 1585.129 ; gain = 0.000 ; free physical = 69282 ; free virtual = 227946
INFO: [v++ 60-1443] [11:26:41] Run run_link: Step generate_sc_driver: Started
INFO: [v++ 60-1453] Command Line:
INFO: [v++ 60-1454] Run Directory: /iu_home/iu7044/_x/link/run_link
INFO: [v++ 60-1441] [11:26:41] Run run_link: Step generate_sc_driver: Completed
Time (s): cpu = 00:00:00.01 ; elapsed = 00:00:00.06 . Memory (MB): peak = 1585.129 ; gain = 0.000 ; free physical = 69271 ; free virtual = 227935
INFO: [v++ 60-244] Generating system estimate report...
INFO: [v++ 60-1092] Generated system estimate report: /iu_home/iu7044/_x/reports/link/system_estimate_vinc.xtxt
INFO: [v++ 60-586] Created /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/vivado_rtl_kernel/rtl_kernel
INFO: [v++ 60-586] Created /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/vivado_rtl_kernel/rtl_kernel
INFO: [v++ 60-1307] Run completed. Additional information can be found in:
    Guidance: /iu_home/iu7044/_x/reports/link/v++_link_vinc_guidance.html
    Timing Report: /iu_home/iu7044/_x/reports/link/imp/impl_1_xilinx_u200_xdma_201830_2_bb_locked
    Vivado Log: /iu_home/iu7044/_x/logs/link/vivado.log
    Steps Log File: /iu_home/iu7044/_x/logs/link/link.steps.log

INFO: [v++ 60-2343] Use the vitis_analyzer tool to visualize and navigate the relevant reports. Run the following command.
    vitis_analyzer /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/vivado_rtl_kernel/rtl_ke
INFO: [v++ 60-791] Total elapsed time: 10h 24m 48s
```

```
INFO: [v++ 60-1653] Closing dispatch client.
```

ПРИЛОЖЕНИЕ В

СОДЕРЖИМОЕ XCLBIN.INFO-ФАЙЛА

Листинг В.1 — Содержимое xclbin.info-файла

```

==========================================================================
XRT Build Version: 2.8.743 (2020.2)
        Build Date: 2020-11-16 00:19:11
           Hash ID: 77d5484b5c4daa691a7f78235053fb036829b1e9
==========================================================================
xclbin Information
------------------
    Generated by:           v++ (2020.2) on 2020-11-18-05:13:29
    Version:                2.8.743
    Kernels:                rtl_kernel_wizard_0
    Signature:
    Content:                Bitstream
    UUID (xclbin):          99fff744-14b1-4e1f-8911-c9b4849b3467
    Sections:               DEBUG_IP_LAYOUT, BITSTREAM, MEM_TOPOLOGY, IP_LAYOUT,
                            CONNECTIVITY, CLOCK_FREQ_TOPOLOGY, BUILD_METADATA,
                            EMBEDDED_METADATA, SYSTEM_METADATA,
                            GROUP_CONNECTIVITY, GROUP_TOPOLOGY
==========================================================================
Hardware Platform (Shell) Information
-------------------------------------
    Vendor:                 xilinx
    Board:                  u200
    Name:                   xdma
    Version:                201830.2
    Generated Version:      Vivado 2018.3 (SW Build: 2568420)
    Created:                Tue Jun 25 06:55:20 2019
    FPGA Device:            xcu200
    Board Vendor:           xilinx.com
    Board Name:             xilinx.com:au200:1.0
    Board Part:             xilinx.com:au200:part0:1.0
    Platform VBNV:          xilinx_u200_xdma_201830_2
    Static UUID:            c102e7af-b2b8-4381-992b-9a00cc3863eb
    Feature ROM TimeStamp:  1561465320

Clocks
------
    Name:       DATA_CLK
    Index:      0
```

```
    Type:        DATA
    Frequency:  300 MHz

    Name:        KERNEL_CLK
    Index:       1
    Type:        KERNEL
    Frequency:  500 MHz

Memory Configuration
--------------------
    Name:           bank0
    Index:          0
    Type:           MEM_DDR4
    Base Address:  0x4000000000
    Address Size:  0x400000000
    Bank Used:      No

    Name:           bank1
    Index:          1
    Type:           MEM_DDR4
    Base Address:  0x5000000000
    Address Size:  0x400000000
    Bank Used:      Yes

    Name:           bank2
    Index:          2
    Type:           MEM_DDR4
    Base Address:  0x6000000000
    Address Size:  0x400000000
    Bank Used:      No

    Name:           bank3
    Index:          3
    Type:           MEM_DDR4
    Base Address:  0x7000000000
    Address Size:  0x400000000
    Bank Used:      No

    Name:           PLRAM[0]
    Index:          4
    Type:           MEM_DRAM
    Base Address:  0x3000000000
    Address Size:  0x20000
    Bank Used:      No

    Name:           PLRAM[1]
```

```
    Index:          5
    Type:           MEM_DRAM
    Base Address:   0x3000200000
    Address Size:   0x20000
    Bank Used:      No

    Name:           PLRAM[2]
    Index:          6
    Type:           MEM_DRAM
    Base Address:   0x3000400000
    Address Size:   0x20000
    Bank Used:      No
==========================================================================
Kernel: rtl_kernel_wizard_0

Definition
----------
    Signature: rtl_kernel_wizard_0 (uint scalar00, int* axi00_ptr0)

Ports
-----
    Port:           s_axi_control
    Mode:           slave
    Range (bytes):  0x1000
    Data Width:     32 bits
    Port Type:      addressable

    Port:           m00_axi
    Mode:           master
    Range (bytes):  0xFFFFFFFFFFFFFFFF
    Data Width:     512 bits
    Port Type:      addressable

--------------------------
Instance:           vinc0
    Base Address:   0x1800000

    Argument:           scalar00
    Register Offset:    0x010
    Port:               s_axi_control
    Memory:             <not applicable>

    Argument:           axi00_ptr0
    Register Offset:    0x018
    Port:               m00_axi
    Memory:             bank1 (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/iu7044/workspace/Alveo_lab1_nikulenko.cfg --connectivity.nk
            rtl_kernel_wizard_0:1:vinc0 --connectivity.slr vinc0:SLR1
            --connectivity.sp vinc0.m00_axi:DDR[1] --input_files
            /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/src/vitis_rtl_kernel/rtl_
            --link --optimize 0 --output
            /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/vivado_rtl_kernel/rtl_ker
            --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/iu7044/workspace/Alveo_lab1_nikulenko.cfg
139                       --connectivity.nk rtl_kernel_wizard_0:1:vinc0
140                       --connectivity.slr vinc0:SLR1
141                       --connectivity.sp vinc0.m00_axi:DDR[1]
142                       --input_files
                              /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/src/vitis_
143                       --link
144                       --optimize 0
145                       --output
                              /iu_home/iu7044/workspace/Alveo_lab1_nikulenko_kernels/vivado_rt
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=AggressiveExplor
153                       --vivado.prop
                              run.impl_1.STEPS.ROUTE_DESIGN.ARGS.DIRECTIVE=Explore
154 ==================================================================================
155 User Added Key Value Pairs
156 ------------------------------
```

```
    <empty>
==========================================================================
```