Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №4 по курсу "Архитектура ЭВМ"

Тема Разработка ускорителей вычислений на платформе Xilinx Alveo

Студент Цветков И.А.

Группа ИУ7-53Б

Оценка (баллы)

Преподаватель Дубровин Е.Н.

Москва — 2021 г.

Содержание

Введение

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

В ходе лабораторной работы предлагается 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.

1 Теоритические основы

При выполнении лабораторной работы будет использоваться усоритель вычислений на **Xilinx Alveo**.

1.1 Технология разработки ускорителей вычислений на Xilinx Alveo

Ускорителями вычислений принято называть специальные аппаратные устройства, способные выполнять ограниченный ряд задач с большей параллельностью и за меньшее время в сравнении с универсальными микропроцессорными ЭВМ . Как правило, ускоритель представляет собой структуру, включающую большое количество примитивных микропроцессорных устройств, объединенных шинами связей.

Создание ускорителей вычислений является трудоемким процессом, так как охватывает не только аппаратную разработку самого устройства, но и предполагает оптимизацию архитектуры ЭВМ для обеспечения наибольшей пропускной способности каналов передачи операндов и результатов, а также минимизации задержек и вычислительных затрат при ожидании работы ускорителей. Можно условно разделить ускорители на два класса: ускорители на основе СБИС и на основе ПЛИС.

В данной лабораторной работе мы изучим технологию создания ускорителей вычислений на основе ПЛИС. Основной плат ускорителя **Xilinx Alveo U200** является ПЛИС xcu200-fsgd2104-2-e архитектуры Xilinx UltraScale, выполненная по 16-нанометровой технологии. Плата обеспечивает взаимодействие с хост-системой через интерфейс PCIe gen3 x16, и помимо ПЛИС содержит 4 планки памяти DIMM DDR4 по 16 ГБ, и два QSFP разъема для подключения 100ГБ Ethernet сети.

Для работы с ускорительной платой разработано специальное окружение **XRT** (Xilinx Runtime), включающее компоненты пользовательского пространства и драйвера ядра. XRT поддерживает как карты ускорителей на основе PCIe, так и встроенную архитектуру на основе MPSoC (для встраиваемых плат с ПЛИС Xilinx), обеспечивающую стандартизованный программ-

ный интерфейс для **Xilinx FPGA**.

1.2 Описание архитектуры разрабатываемого ускорителя

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 SystemVerilo
- функциональный тест ускорителя VINC на языке SystemVerilog.

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 (на рисунке 1.1). В рамках всей системы используется единое 64-х разрядное адресное пространство, в котором формируются адреса на всех AXI4 шинах.

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.

Выбор одного из регионов для размещения проектов осуществляется на этапе так называемой линковки конфигурационного файла при помощи компилятора v++(фактически: компоновки, размещение и трассировки нескольких проектов в единый конфигурационный файл).

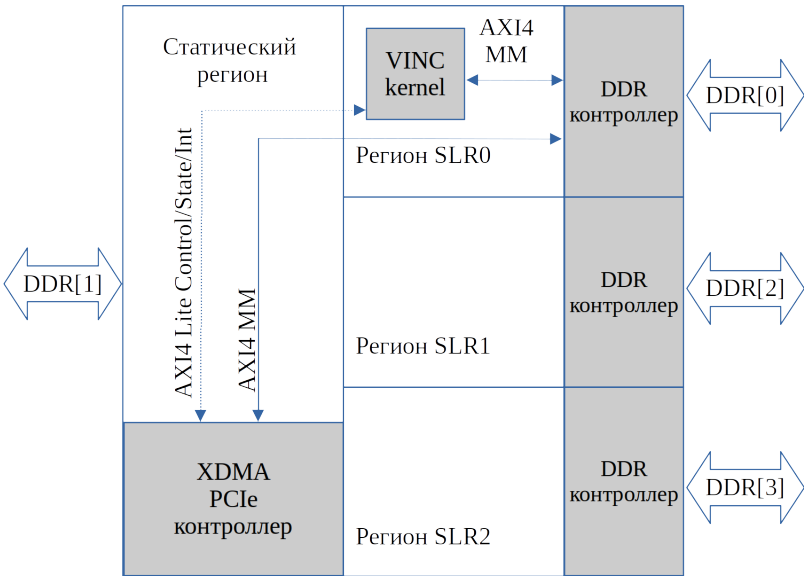

Рисунок 1.1 – Размещение проекта на ПЛИС xcu200-fsgd2104-2-e карты Alveo U200

2 Выполнение лабораторной работы

Для изучения технологии будут выполнены следующие задания.

2.1 Моделирование исходного проекта VINC

На рисунке 2.1 представлена транзакция чтения данных вектора на шине AXI4 MM из DDR памяти. Также на рисунке 2.2 – транзакция записи результата инкремента данных на шине AXI4 MM, а на рисунке ?? инкремент данных в модуле.

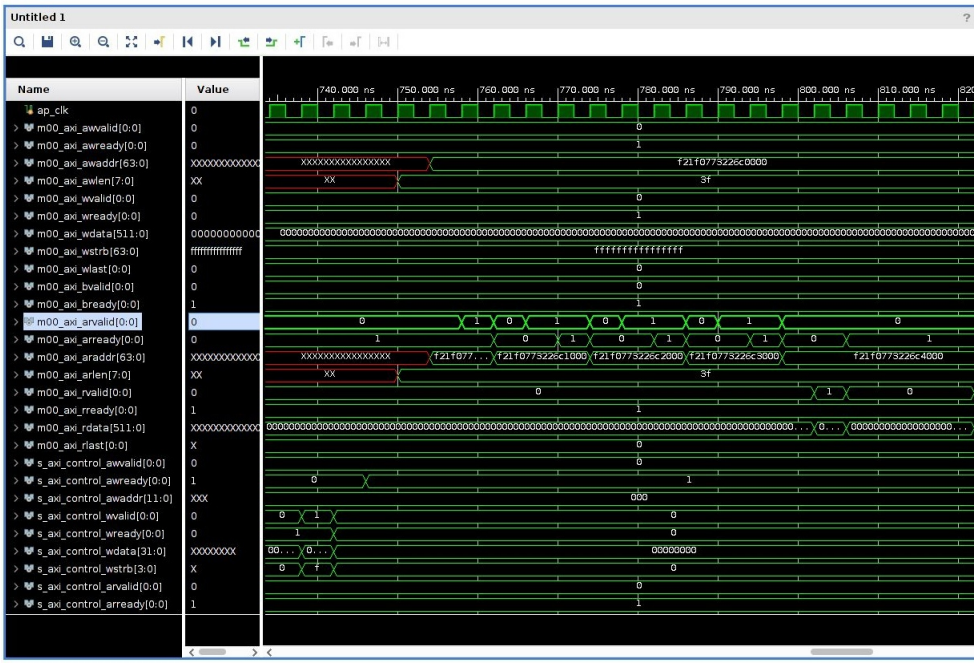

Рисунок 2.1 – Транзакция чтения данных вектора на шине AXI4 MM из DDR памяти

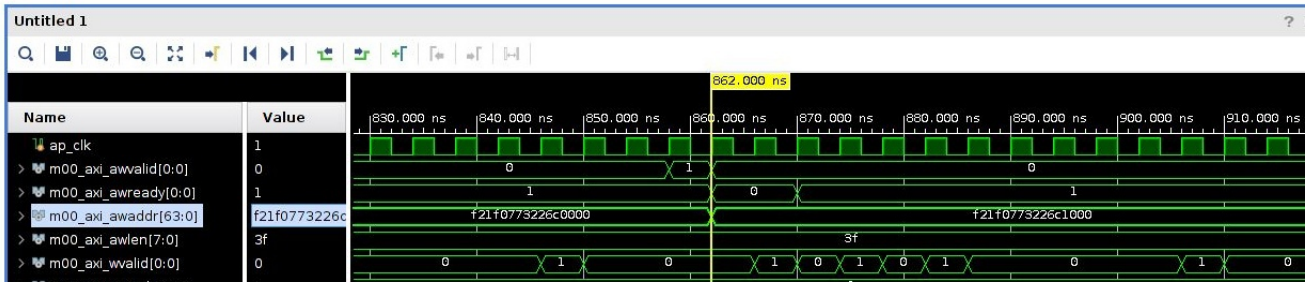

Рисунок 2.2 – Транзакция записи результата инкремента данных на шине AXI4 MM

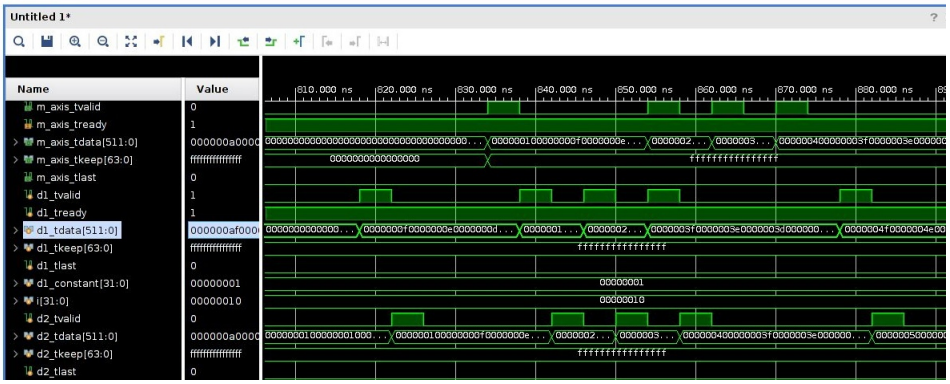

Рисунок 2.3 – Инкремент данных в модуле

2.2 Моделирование проекта VINC, изменен-ного по индивидульному варианту

В соответствии с индивидуальным вариантом (Вариант 19) нужно было реализовать в коде следующую функцию:

$$R[i] = max(A[i], 3000) \tag{2.1}$$

На рисунке 2.4 представлена реализация функции на языке Verilog, которая была вставлена в код проект. При этом использовалась константа $CONST = 3000$.

```
// 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] > CONST)
      d2_tdata[i*C_ADDER_BIT_WIDTH+:C_ADDER_BIT_WIDTH] <= d1_tdata[C_ADDER_BIT_WIDTH*i+:C_ADDER_BIT_WIDTH];
    else
      d2_tdata[i*C_ADDER_BIT_WIDTH+:C_ADDER_BIT_WIDTH] <= CONST;
  end
end
```

Рисунок 2.4 – Функция индивидуального варианта

При этом на рисунке 2.5 представлена транзакция чтения данных вектора на шине AXI4 MM из DDR памяти. Также на рисунке 2.6 – транзакция записи результата инкремента данных на шине AXI4 MM, а на рисунке **??** инкремент данных в модуле.

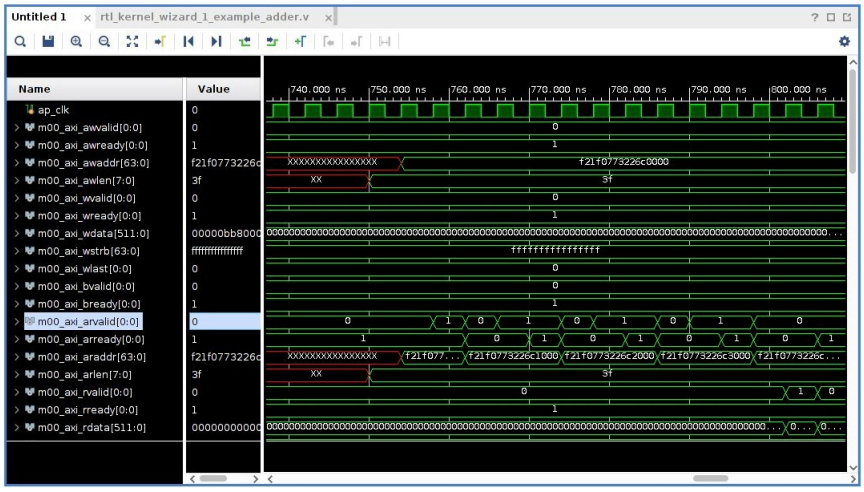

Рисунок 2.5 – Транзакция чтения данных вектора на шине AXI4 MM из DDR памяти

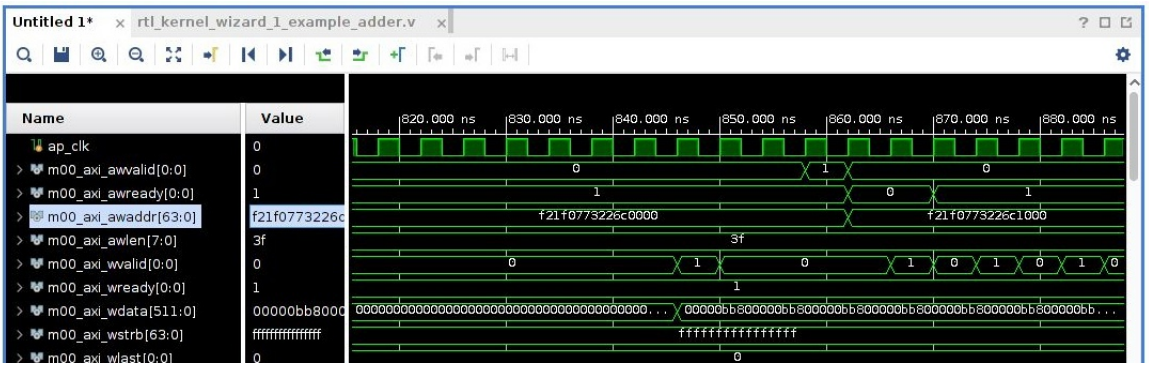

Рисунок 2.6 – Транзакция записи результата инкремента данных на шине AXI4 MM

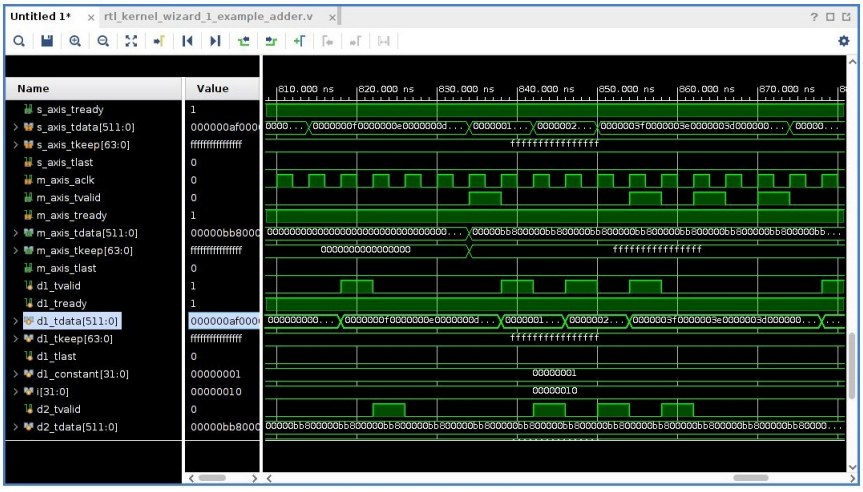

Рисунок 2.7 – Инкремент данных в модуле

2.3 Линковка проекта

Для линковки проекта компилятором **v++** используется конфигурационный файл **config.cfg**, который содержит основную информацию для работы компилятора, такую, как:

- количество и условные имена экземпляров ядер;
- тактовая частота работы ядра;
- для каждого ядра: выбор области SLR (SLR[0..2]), выбор DDR (DDR[0..3]) памяти, выбор высокопроизводительной памяти PLRAM(PLRAM[0,1,2]).
- параметры синтеза и оптимизации проекта.

На рисунке 2.8 представлен конфигурационный файл для данного проекта, в котором **SLR1** и **DDR[2]**, что соответствует индивидуальному варианту.

```
[connectivity]
nk=rtl_kernel_wizard_1:1:vinc0
slr=vinc0:SLR1
sp=vinc0.m00_axi:DDR[2]

[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 2.8 – Конфигурационный файл

Примечание: листинги файлов **v++*.log** и ***.xclbin.info** приведены в приложении.

2.4 Тестирование

После успешной линковки проекта получается файл ***.xclbin**. Также нужно получить *exe* файл **host_example.cpp**, который будет использован при тестировании.

Но прежде в **host_example.cpp** необоходимо изменить условие проверки. На рисунке 2.9 представлена изменная проверка, которая соответствует функции для индивидуального варианта.

```
// Check Results

for (cl_uint i = 0; i < number_of_words; i++) {
    if (h_data[i] < 3000)
        h_data[i] = 3000;

    if ((h_data[i]) != h_axi00_ptr0_output[i]) {
        printf("ERROR in rtl_kernel_wizard_1::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 2.9 – Измененная проверка при тестировании

В итоге, на рисунке 2.10 приведены результаты тестированияы утилитой **xgdb**. Все тесты пройдены успешно.

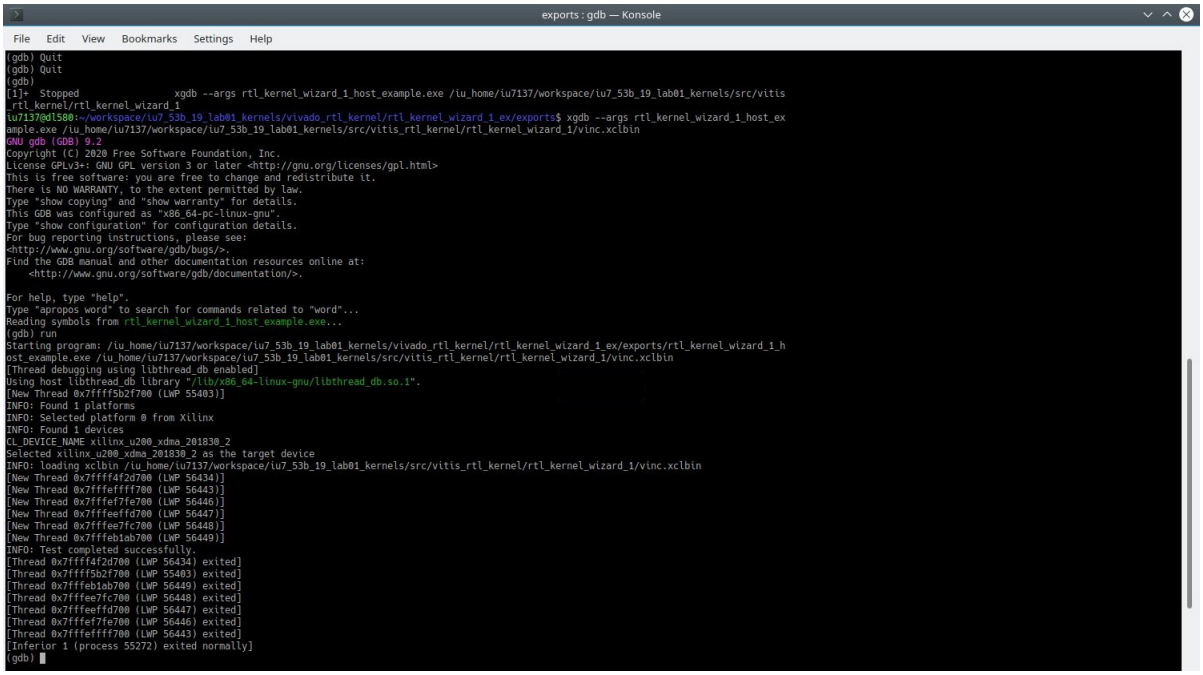

Рисунок 2.10 – Результаты тестирования

Заключение

В данной лабораторной работе были рассмотрены и иузчены ускорители вычислений на примере Alveo от фирмы Xilinx.

Приложение

Coming soon...