МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт (филиал) КБ направление 12.03.01

Кафедра КБ6 “Приборы и информационно-измерительные системы”

Дисциплина “РУнБПЛИС”

ОТЧЕТ
по лабораторной работе на тему:

Создание IP-ядра в Vivado

Студент А.Д. Глухов
подпись, дата инициалы и фамилия

Группа БПБО-02-20 шифр 20Б0924

МОСКВА 2022 г.

Содержание

ВВЕДЕНИЕ

IP-ядра – это готовые блоки для проектирования микросхем (например, для построения систем на кристалле).

Различают три основных класса блоков:

программные IP-блоки— блоки, специфицированные на языке описания аппаратуры;

схемотехнические блоки — блоки, специфицированные на схемотехническом уровне, без привязки к конкретной топологической реализации;

физические (топологические) блоки— блоки, специфицированные на физическом уровне реализации СБИС (например, GDSII для ASIC).

Программируемая логическая интегральная схема (ПЛИС) - электронный компонент (интегральная микросхема), используемый для создания конфигурируемых цифровых электронных схем.

Программируемые логические интегральные схемы - ПЛИС являются одними из самых перспективных элементов цифровой схемотехники. ПЛИС представляет собой кристалл, на котором расположено большое количество простых логических элементов. Изначально эти элементы не соединены между собой. Соединение элементов (превращение разрозненных элементов в электрическую схему) осуществляется с помощью электронных ключей, расположенных в этом же кристалле. Электронные ключи управляются специальной памятью, в ячейки которой заносится код конфигурации цифровой схемы. Таким образом, записав в память ПЛИС определенные коды, можнособрать цифровое устройство любой степени сложности (это зависит от количества элементов на кристалле и параметров ПЛИС). В отличие от микропроцессоров, в ПЛИС можно организовать алгоритмы цифровой обработки на аппаратном (схемном) уровне. При этом быстродействие цифровой обработки резко возрастает.

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Создать IP-ядро мультиплексора в среде разработки Xilinx Vivado, выполнить тестирование, получить электрическую схему и провести моделирование, доказывающее работоспособность устройства.

1 КРАТКОЕ СОЗДАНИЕ МУЛЬТИПЛЕКСОРА

Как уже было показано в отчете первой лабораторной работы, создается мультиплексор, по тем же шагам и этапам.

В данном случае мы будем использовать код, показанный на рисунке 1.

```
module mult(
    input [3:0] X,
    input [1:0] S,
    input EN,
    output Y
    );

     reg Y;

     always @*
     if(EN)
     begin
        Y = 0;
        Y = X[S];
     end
endmodule
```

Рисунок 1 – Код мультиплексора

2 ПОДКЛЮЧЕНИЕ IP

Для начала необходимо создать IP-ядро. Заходим в меню, выбираем Tools, Create and Package New IP. На рисунке 2 демонстрируется создание нового IP-ядра.

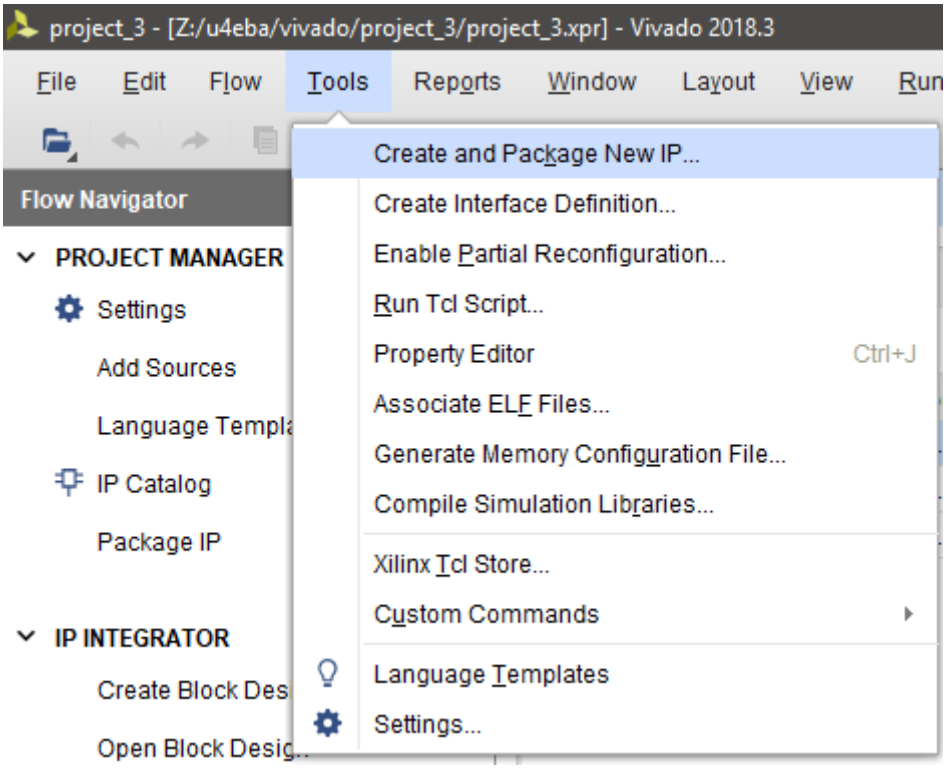

Рисунок 2 – Создание IP-ядра

В появившемся окне нажимаем Next. В следующем выбираем Package your current project для создания ядра из данного проекта, жмем Next. На рисунке 3 демонстрируется окно Create and Package.

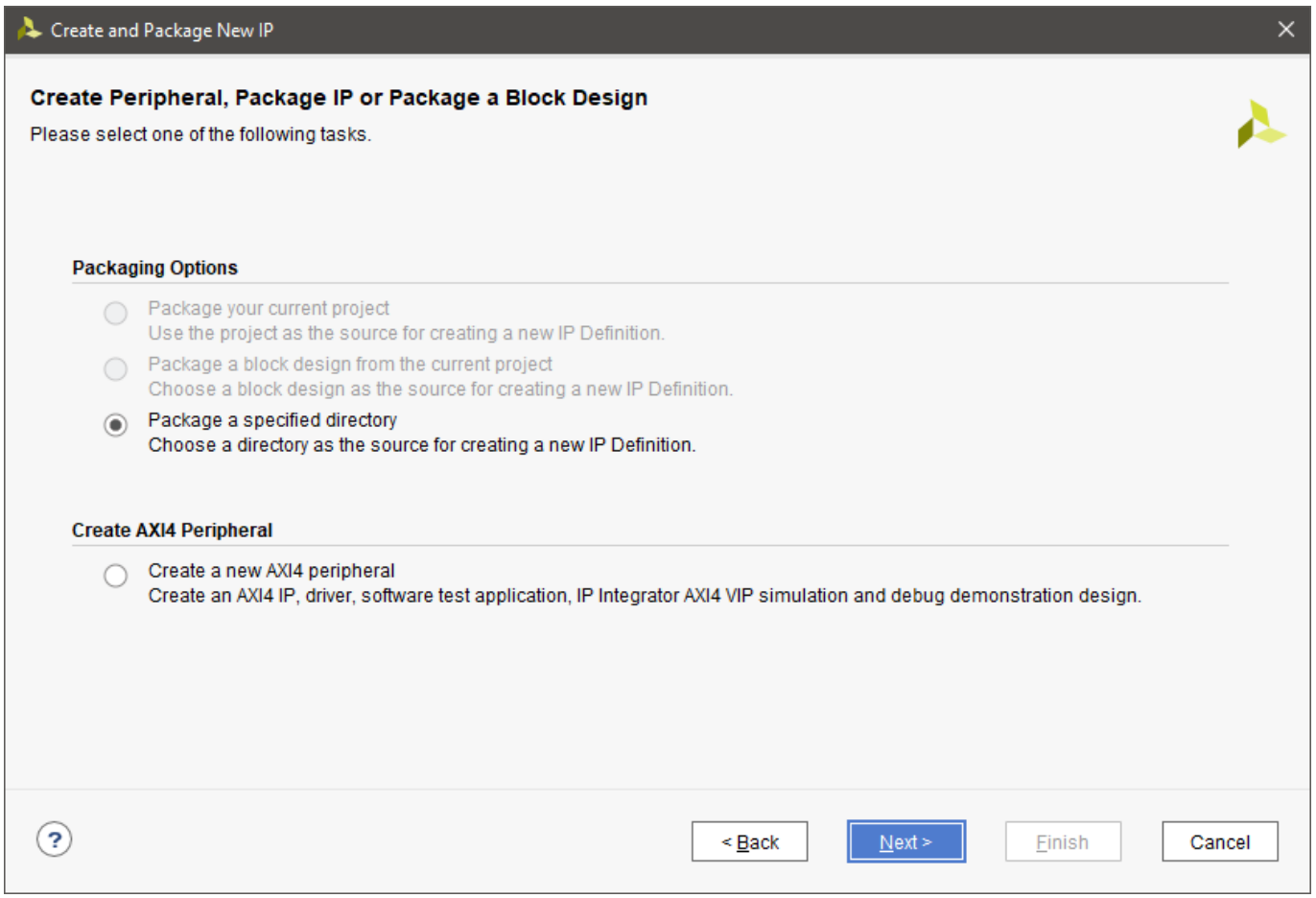

Рисунок 3 – Окно Create and Package

В следующем меню можно выбрать путь к директории с ядром, путь к ядру будет по умолчанию - в директории проекта. Жмем Next и затем Finish. На рисунке 4 демонстрируется возможность изменение пути к директории с ядром.

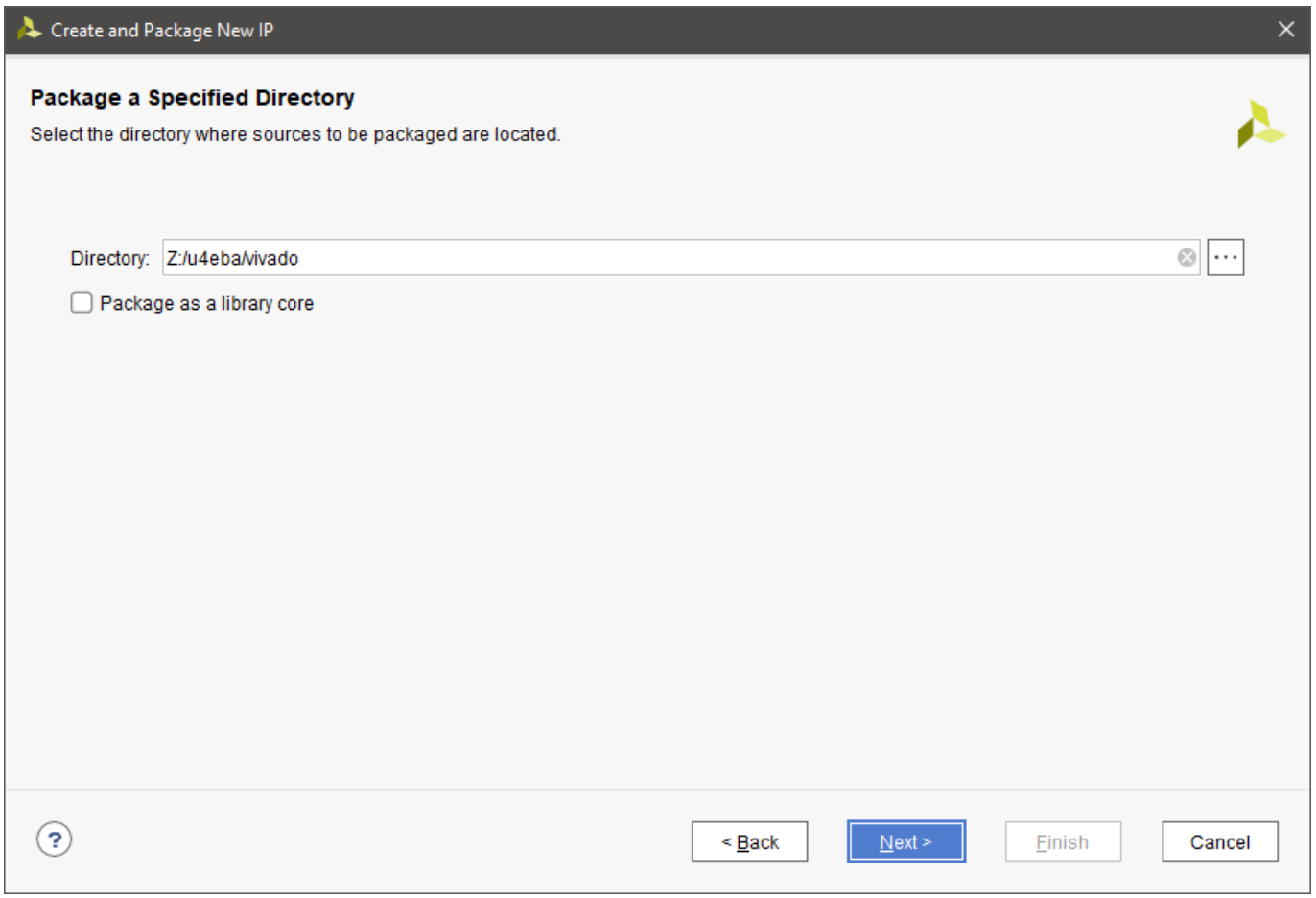

Рисунок 4 – Путь к директории с ядром

В менеджере проектов должна появится новая вкладка Package IP.

Следующим действием необходимо упаковать IP-ядро. В меню Identification можно указать информацию о ядре (о создателе, версии и т.д.). В меню Compatibility можно добавить поддерживаемые устройства. В Menu Groups специальная настройка групп для проектов с большим количеством модулей. В остальных меню производится настройка портов и работа с памятью. Сейчас нас интересует пункт Review and Package, зайдя в который, для упаковки, нажимаем Package IP. На рисунке 3 демонстрируется окно Create and Package.

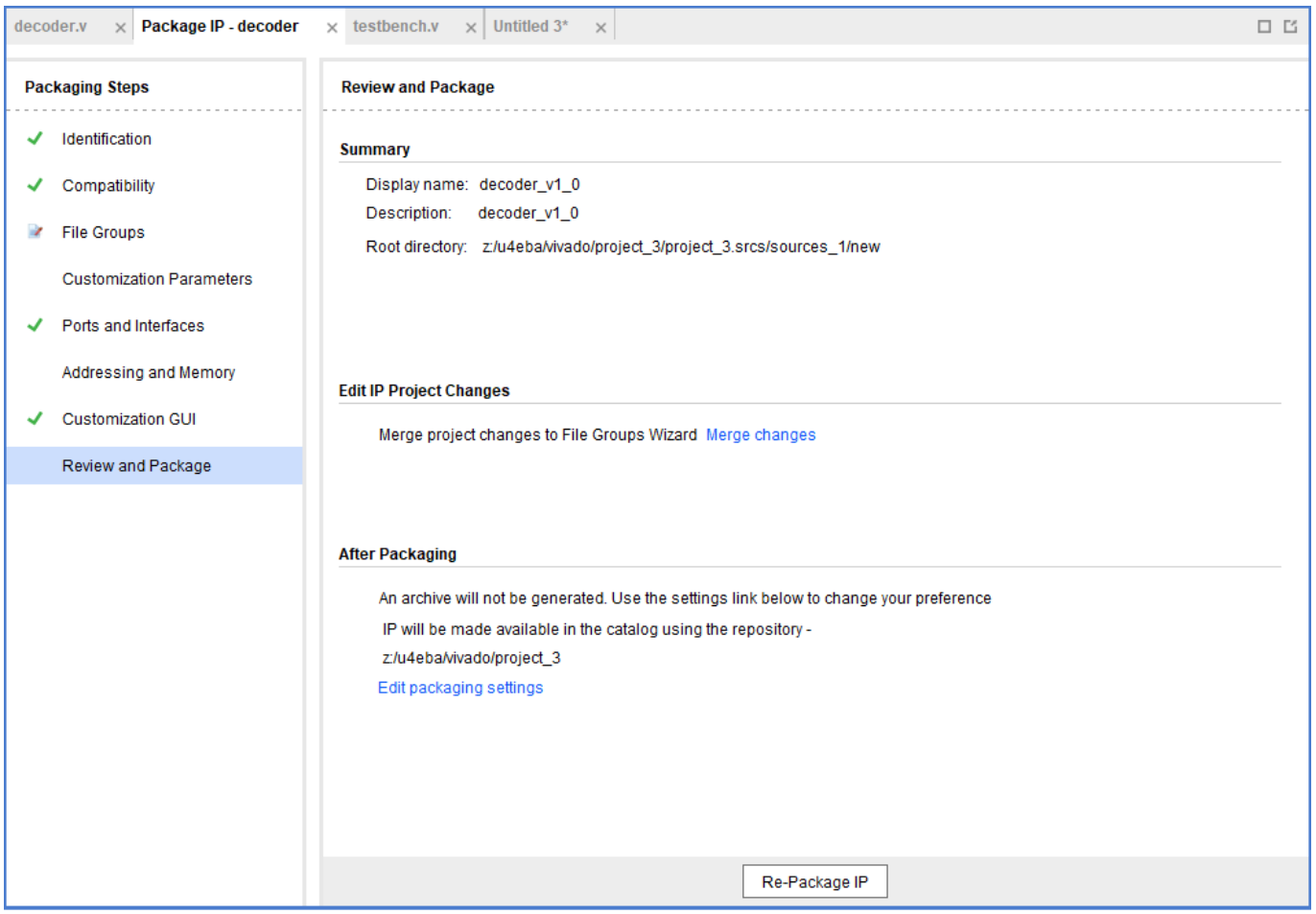

Рисунок 5 – Настройки Package IP

На этом этапе ядро готово, и в него занесена информация о симуляциях, синтезах, имплементациях (если таковые имелись). При изменении чего-либо в ядре достаточно снова зайти в меню Review and Package и нажать Re-package IP.

Далее необходимо использовать созданное ядро в проекте. Требуется создать расширенный мультиплексор из имеющихся. Пример представлен на рисунке 6.

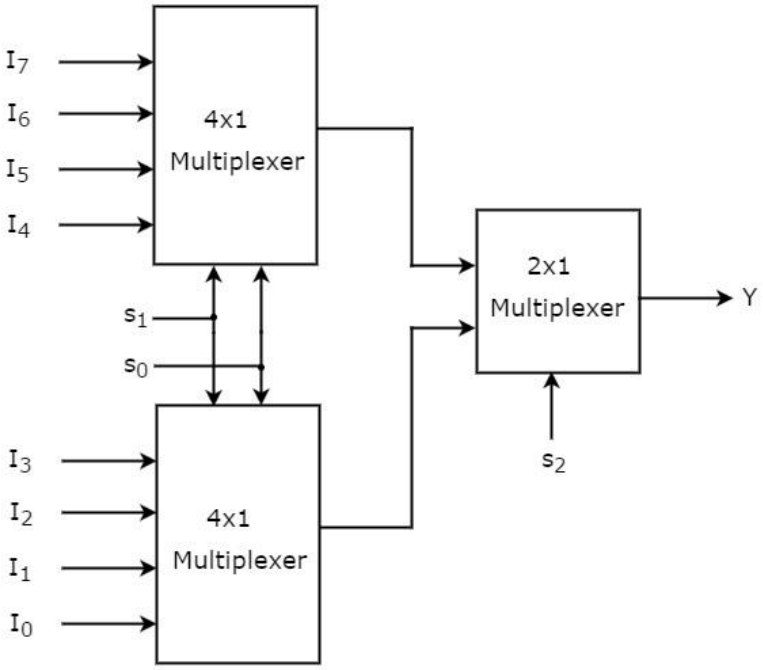

Рисунок 6 – Пример расширенного мультиплексора

Вместо мультиплексора на 2 входа мы будем использовать мультиплексор на 4 входа, а не используемые входы подключим к точке нулевого потенциала.

Создаем новый проект. В менеджере проектов выбираем IP Catalog. В появившемся каталоге нашимаем правой кнопкой мыши в любом месте и выбираем “Add Repository...” в отобразившемся меню.

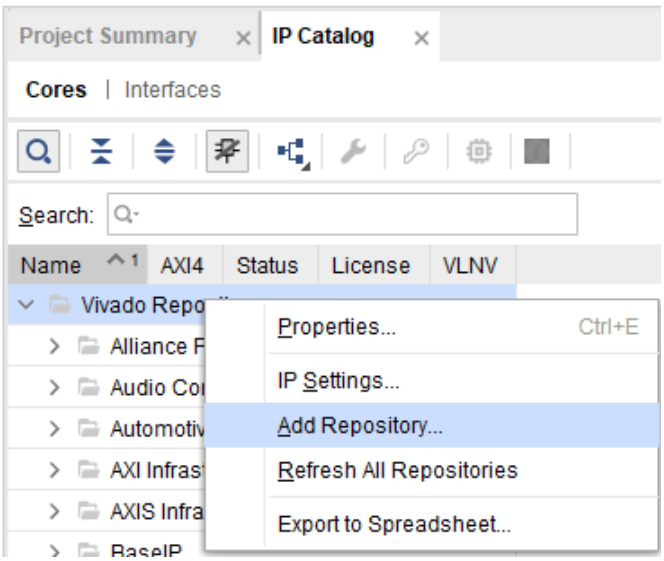

Рисунок 7 – Контекстное меню Vivado Repository

В появившемся меню указываем путь к папке с нашим IP-ядром. Теперь оно доступно для использования в новом проекте. Создаем блочный дизайн.

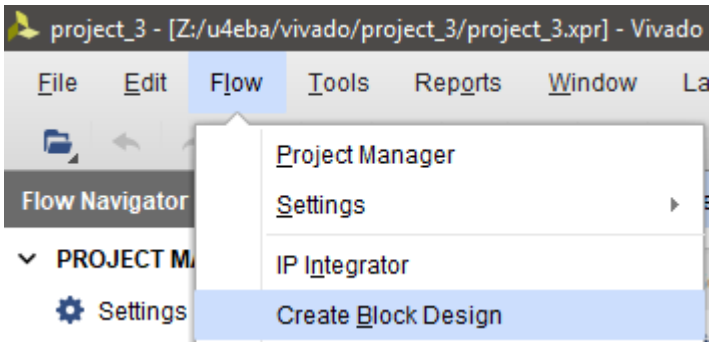

Рисунок 8 – контекстное меню Flow

Нажимаем Add IP.

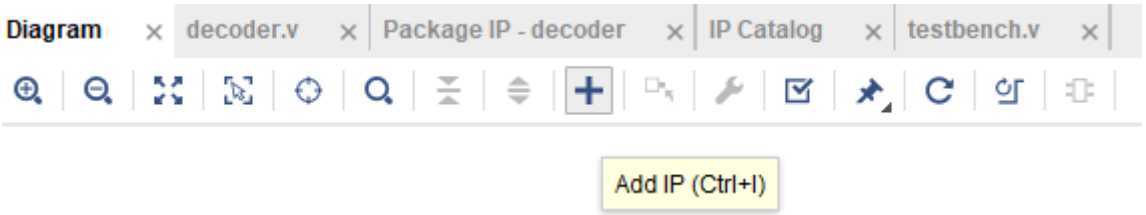

Рисунок 9 – Add IP

Находим в каталоге наше ядро.

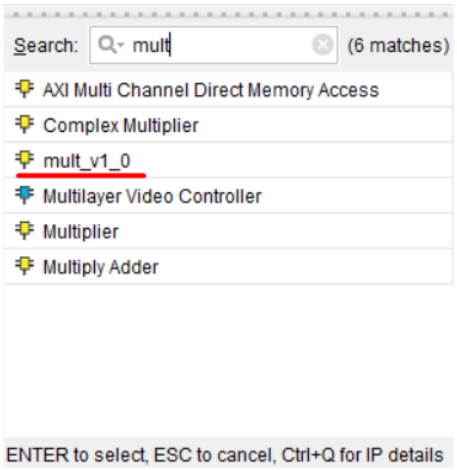

Рисунок 10 – Ядро в каталоге

Добавляем его трижды. Для объединения выходов с двух мультиплексоров в один добавляем встроенное IP-ядро Concat.

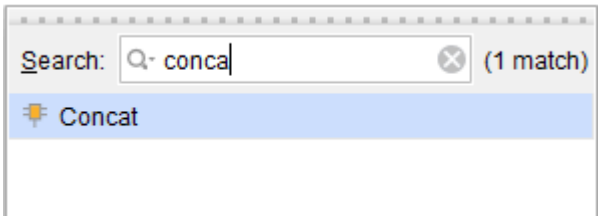

Рисунок 11 – Добавление элемента Concat

Рисунок 12 - Блочный элемент Concat

Дважды щелкаем по нему левой кнопкой мыши и настраиваем его как показано на рисунке 13.

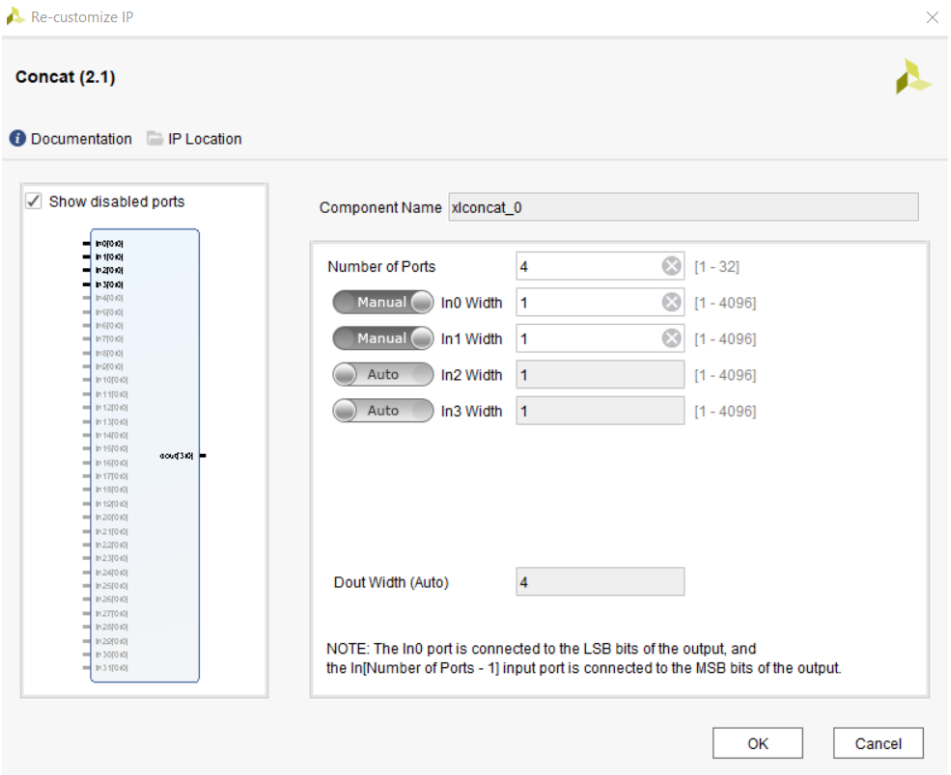

Рисунок 13 – Настройки Concat

Нажимаем по пустому месту правой кнопкой мыши и выбираем Create Port. Создаем порты для входов DATA_IN[7:0], EN_0, S[2:0], GND, и для выхода DATA_OUT. Пример создания порта представлен на рисунке 14.

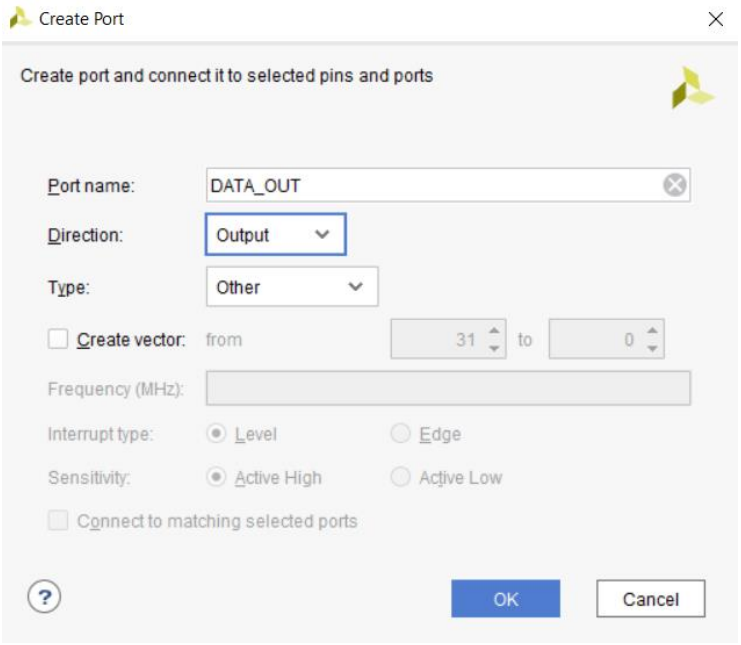

Рисунок 14 – Создание порта выхода

Далее подключаем порт данных к входам мультиплексоров. Для разделения на старшие и младшие биты данных используется IP-ядро Slice. Соединение представлено на рисунке 15.

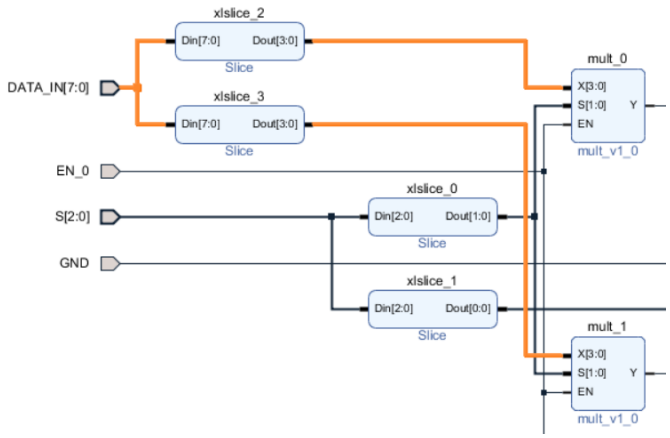

Рисунок 15 – Подключение порта данных

Таким же способом подключаем управляющий порт S. В этом случае на управляющие входы первых двух мультиплексоров пойдут 2 младших бита, а на

входы другого пойдет один старший. Чтобы не оставлять один из управляющих входов неподключенным, мы его подключим ко входу GND, который является точкой нулевого потенциала. Подключение входа S представлено на рисунке 16.

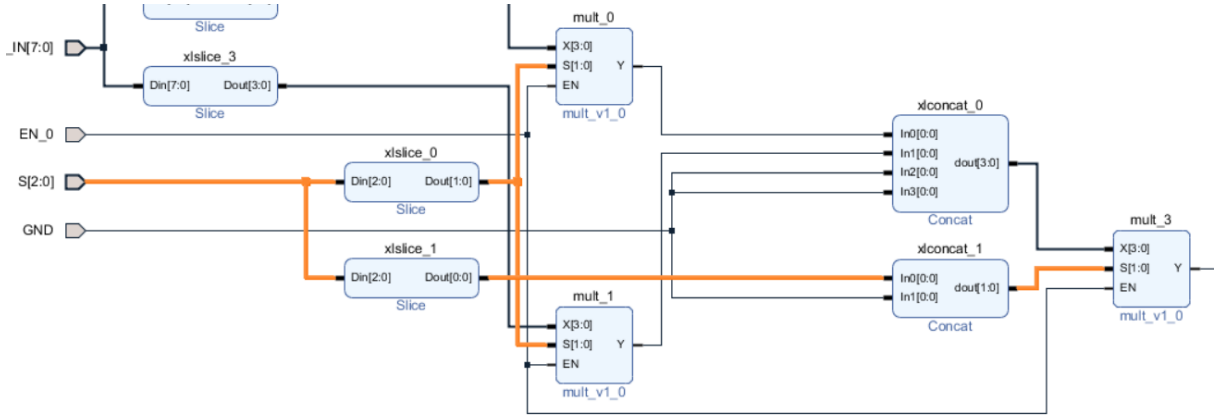

Рисунок 16 – Подключение порта S

Итоговый вид блок схемы представлен на рисунке 17.

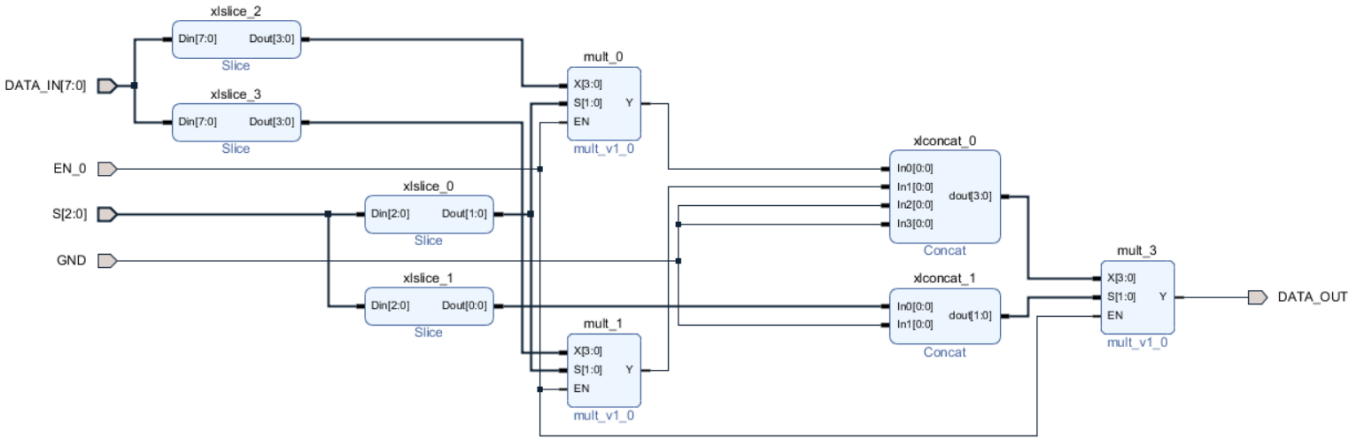

Рисунок 17 – Блок схема

Далее необходимо запустить симуляцию проекта. Для написания testbench, добавим файл симуляции Add Sources, Add or create simulation sources, Create File, указать имя testbench. Код testbench представлен на рисунке 18.

```
module testbench();


   reg [2:0] S;
   reg EN_0;
   reg [7:0] DATA_IN;
   wire DATA_OUT;
   reg GND;

   design_1 uut(.DATA_IN(DATA_IN), .S(S), .EN_0(EN_0), .GND(GND),
.DATA_OUT(DATA_OUT));
   initial begin
      GND = 0;
      EN_0 = 1;
      S = 0;
      DATA_IN= 1;
      #200;

      S = 0;
      DATA_IN= 2;
      #200;

       S = 0;
       DATA_IN= 3;
       #200;
      $stop;
   end
endmodule
```

Рисунок 18 – Код testbench

Далее проводим моделирование. Результаты моделирования представлены на рисунке 19.

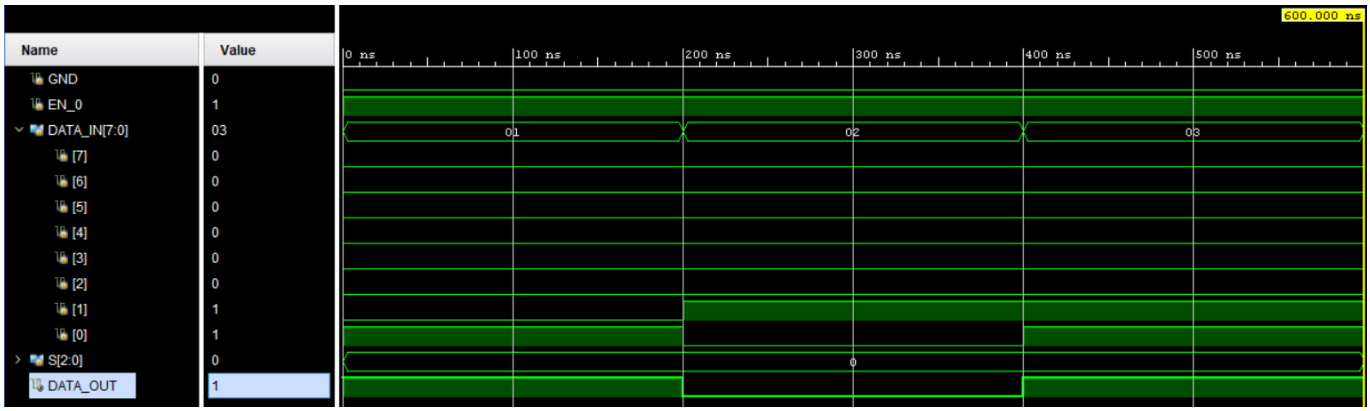

Рисунок 19 – Результаты моделирования

По ее результатам убеждаемся в том, что устройство работает корректно.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной лабораторной работы было осуществлено проектирование IP-ядра мультиплексора и использование его в среде разработки Vivado, написан код мультиплексора на языке Verilog, выполнено тестирование, создана электрическая схема модуля и проведено моделирование, доказывающее работоспособность устройства.

СПИСОК ИСТОЧНИКОВ

1 Грушвицкий Р.И., Мурсаев А.Х., Угрюмов Е.П. Проектирование систем намикросхемах программируемой логики. СПб.: БХВ-Петербург, 2002. – 608 с.

2 Максфилд К. Проектирование на ПЛИС. Курс молодого бойца.М.: Издательский дом «Додэка-XXI», 2007. – 408 с. (перевод с английского).

3 Тарасов И. Маршрут проектирования ПЛИС Xilinx в САПР Vivado // Компоненты и технологии. – 2012. – № 12.

ПРИЛОЖЕНИЕ А

Исходный код модуля мультиплексора

```
module mult(
    input [3:0] X,
    input [1:0] S,
    input EN,
    output Y
    );

     reg Y;

     always @*
     if(EN)
     begin
        Y = 0;
        Y = X[S];
     end
endmodule
```

ПРИЛОЖЕНИЕ Б

Исходный код testbench.

```
module testbench();
    reg [2:0] S;
    reg EN_0;
    reg [7:0] DATA_IN;
    wire DATA_OUT;
    reg GND;

    design_1 uut(.DATA_IN(DATA_IN), .S(S), .EN_0(EN_0), .GND(GND), .DATA_OUT(DATA_OUT));
    initial begin
        GND = 0;
        EN_0 = 1;
        S = 0;
        DATA_IN= 1;
        #200;

        S = 0;
        DATA_IN= 2;
        #200;

         S = 0;
         DATA_IN= 3;
         #200;
        $stop;
    end
endmodule
```