САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ПЕТРА ВЕЛИКОГО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчет

по заданию lab8

Дисциплина

«Технологии проектирования аппаратных средств компьютерных систем»

выполнил:

Курякин Д. А

группа:

преподаватель:

Антонов А. П.

Санкт-Петербург

2022

Оглавление

[1 Задание lab1\_1 4](#_Toc98192533)

[1.1 Задание 4](#_Toc98192534)

[1.2 Описание на языке Verilog 4](#_Toc98192535)

[1.3 Результат синтеза (RTL) 4](#_Toc98192536)

[1.4 Моделирование 4](#_Toc98192537)

[1.5 Назначение выводов СБИС 5](#_Toc98192538)

[1.6 Тестирование на плате Nexys4 DDR 6](#_Toc98192539)

[1.7 Выводы 6](#_Toc98192540)

# Задание lab8

## Задание

Изображение выглядит как текст

Автоматически созданное описание

## Описание на языке System Verilog

Описание разрабатываемого устройства на языке Verilog приведено ниже листингах 1–2.

Листинг 1. db\_Lab8\_1.sv

|  |
| --- |
| `timescale 1ns / 1ps  module db\_Lab8\_1(input logic clk);  logic rst;  logic ena;  logic arith\_logic\_sel;  logic [7:0] ip\_data1, ip\_data2;  logic [1:0] operation;  logic [15:0] data\_out;  arith\_logic A(.\*);  vio\_0 VIO(.clk(),  .probe\_in0(data\_out),  .probe\_out0(rst),  .probe\_out1(ena),  .probe\_out2(arith\_logic\_sel),  .probe\_out3(operation),  .probe\_out4(ip\_data1),  .probe\_out5(ip\_data2));  ila\_0 ILA(.clk(),  .probe0(data\_out));  endmodule |

Листинг 2. arith\_logic.sv

|  |
| --- |
| `timescale 1ns/1ps  // User defined enumerated data type declaration  typedef enum logic[1:0] {add=0, sub, mul, div} arith\_operation;  typedef enum logic[1:0] {nand\_op=0, nor\_op, not\_op, xor\_op} logic\_operation;  // Creating one Packed and one Unpacked Structure here  // User defined Packed Structure declaration  typedef struct packed { arith\_operation arithmetic\_op;  logic\_operation logical\_op;  logic [7:0] data1;  byte data2;  logic [15:0] arith\_result;  logic [15:0] logic\_result;  } arith\_logic\_info;    module arith\_logic(  input logic clk, rst, ena,  input logic arith\_logic\_sel, // Selects between arith or logic operations  input logic [1:0] operation, // Selects which arith or logic operations is to be done  input logic [7:0] ip\_data1, ip\_data2, // Two data inputs  output logic [15:0] data\_out // Output of the selected operation  );  bit arith\_logic\_sel\_rg;  bit [1:0] operation\_rg;  bit [7:0] ip\_data1\_rg, ip\_data2\_rg;  always @ (posedge clk) begin  if (ena)  if (!rst) begin  arith\_logic\_sel\_rg <= arith\_logic\_sel;  operation\_rg <= operation;  ip\_data1\_rg <= ip\_data1;  ip\_data2\_rg <= ip\_data2;  end else begin  arith\_logic\_sel\_rg <= 0;  operation\_rg <= 0;  ip\_data1\_rg <= 0;  ip\_data2\_rg <= 0;  end  end  always @ (posedge clk) begin  if (ena)  if (!rst) begin  arith\_logic\_info arith\_data, logic\_data; // Creating two objects arith\_data and logic\_data  // Selecting Arithmetic operation  if (arith\_logic\_sel\_rg==0) begin  //Read the input data  arith\_data.arithmetic\_op = arith\_operation'(operation\_rg);  arith\_data.logical\_op = logic\_operation'(0);  arith\_data.data1 = ip\_data1\_rg;  arith\_data.data2 = ip\_data2\_rg;    case (arith\_data.arithmetic\_op) // Accessing enum data type through arith\_data input  // As per the value of enum, do the arithmatic operations  // Accesing the Packed Structure for two datas and storing the result in an Packed Union  add : arith\_data.arith\_result = arith\_data.data1 + arith\_data.data2;  sub : arith\_data.arith\_result = arith\_data.data1 - arith\_data.data2;  mul : arith\_data.arith\_result = arith\_data.data1 \* arith\_data.data2;  div : arith\_data.arith\_result = arith\_data.data1 / arith\_data.data2;  endcase  data\_out = arith\_data.arith\_result; //Write the result to the output  end else begin // Selecting Logical operation    logic\_data.arithmetic\_op = arith\_operation'(0);  logic\_data.logical\_op = logic\_operation'(operation\_rg);    logic\_data.data1 = ip\_data1\_rg;  logic\_data.data2 = ip\_data2\_rg;    case (logic\_data.logical\_op) // Accessing enum data type through logic\_data input  // As per the value of enum, do the logic operations  // Accesing the Packed Structure for two datas and storing the result in an Packed Union  nand\_op : logic\_data.logic\_result = ~((logic\_data.data1) & (logic\_data.data2));  nor\_op : logic\_data.logic\_result = ~((logic\_data.data1) | (logic\_data.data2));  not\_op : logic\_data.logic\_result = ~ (logic\_data.data1);  xor\_op : logic\_data.logic\_result = ((logic\_data.data1) ^ (logic\_data.data2));  endcase  data\_out = logic\_data.logic\_result;  end  end else begin  data\_out = '0;  end  end  endmodule |

## Результат синтеза (RTL)

Результат синтеза описания на языке Verilog в пакете Quartus приведен ниже, на рис 1-1.

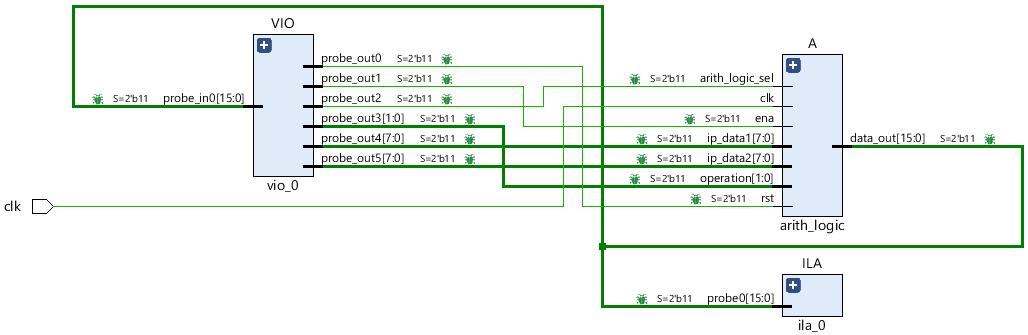


Рис. 1‑1 Синтезированная схема

## Моделирование

Для проверки правильности работы созданного Verilog описания использовались тесты представленные в листингt 3:

Листинг 3. tb\_arith\_logic.sv

|  |
| --- |
| `timescale 1ns / 1ps  module tb\_arith\_logic;  logic clk = 0;  logic rst = 0;  logic ena = 1;  logic arith\_logic\_sel;  logic [7:0] ip\_data1, ip\_data2;  logic [1:0] operation;  logic [15:0] data\_out;  always #10 clk=~clk;  // instantiate the unit under test (uut)  arith\_logic inst(.\*);  initial begin  // initialize inputs  arith\_logic\_sel = 0;  operation = 2'b00;  ip\_data1 = 0;  ip\_data2 = 0;  end  initial begin  arith\_logic\_sel = 0;  ip\_data1 = 8'd20;  ip\_data2 = 8'd10;  operation = 2'b00;  #20;  operation = 2'b01;  #20;  operation = 2'b10;  #20;  operation = 2'b11;  #20;  arith\_logic\_sel = 1;  ip\_data1 = 8'd60;  ip\_data2 = 8'd15;  operation = 2'b00;  #20;  operation = 2'b01;  #20;  operation = 2'b10;  #20;  operation = 2'b11;  end  endmodule |

Результаты моделирования приведены на Рис. 1‑2

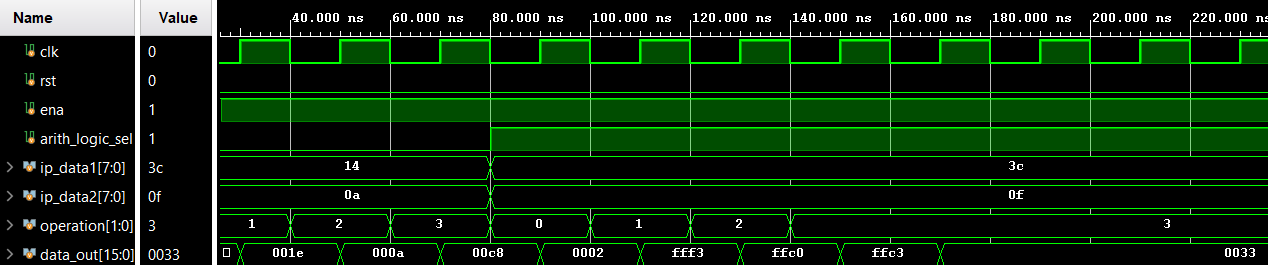


Рис. 1‑2 Результат моделирования tb\_lfsr

## Назначение выводов СБИС

Назначение выводов СБИС и стандартов ввода вывода приведено на Рис. 1‑3

Изображение выглядит как стол

Автоматически созданное описание

Рис. 1‑3 Назначение выводов

## Тестирование на плате Nexys4 DDR

Для тестирования проекта на плате использовались тесты, описанные в разделе 1.4. Проведенное на плате Nexys4 DDR тестирование разработанного устройства показало: результаты совпадают с ожидаемыми, устройство работает в соответствии с заданием.

## Выводы

В результате было реализовано устройство, описанное в лабораторной работе.