САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ПЕТРА ВЕЛИКОГО

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

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчет

по заданию lab3

Дисциплина

«Средства проектирования аппаратуры цифровой обработки сигналов»

выполнил:

Курякин Д. А

группа:

преподаватель:

Антонов А. П.

Санкт-Петербург

2022

Оглавление

[1 Задание lab3\_2 3](#_Toc103572720)

[1.1 Задание 3](#_Toc103572721)

[1.2 Описание на языке Verilog 4](#_Toc103572722)

[1.3 Результат синтеза (RTL) 10](#_Toc103572723)

[1.4 Моделирование 10](#_Toc103572724)

[1.5 Назначение выводов СБИС 11](#_Toc103572725)

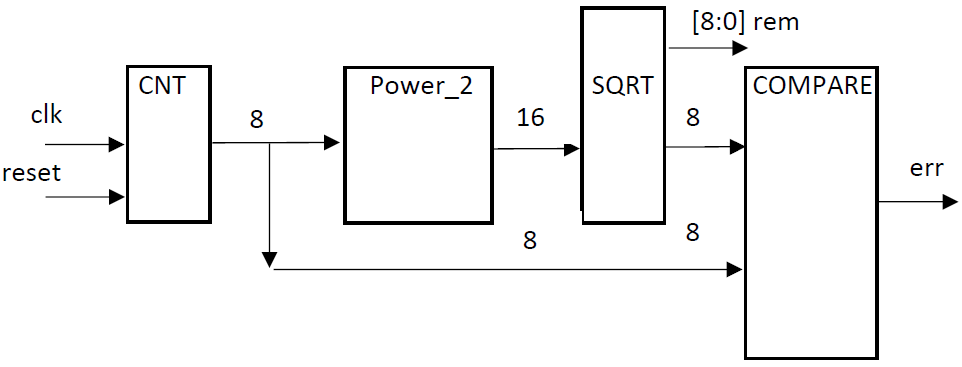
[1.6 Отладка 11](#_Toc103572726)

[1.7 Выводы 11](#_Toc103572727)

# Задание lab3\_2

## Задание

1. Структура проекта



2. Входы

1. clk – тактовый сигнал (50МГц)
2. Reset – вход асинхронного сброса счетчика

3. Выходы

1. err – выход ошибки обработки (если исходные данные не равны целой части данных, полученных в результате обработки, то на выходе будет 1; если равны – то 0).
2. [8:0] rem – остаток после вычисления квадратного корня.

4. CNT - счетчик, создаваемый с помощью IP модуля LPM\_COUNTER

1. Разрядность 8 бит
2. Двоичный счетчик на сложение
3. Вход асинхронного сброса.

5. Power\_2 – модуль возведения в степень 2. Описать на VerilogHDL.

1. Входы 8 бит и выходы 16 бит.

6. SQRT – модуль извлечения квадратного корня, создаваемый с помощью IP модуля ALTSQRT

1. Вход 16 бит
2. Выход целой части 8 бит
3. Остаток 9 бит

7. COMPARE – модуль сравнения. Если входы не равны, то на выходе будет 1; если равны – то 0. Описать на VerilogHDL.

8. Что надо сделать:

1. Создать проект в пакете Quartus
2. Рабочая папка C:\Intel\_trn\Quartus\_MSim\_Deb\Lab2\_3
3. Модуль верхнего уровня Lab2\_3
4. Файл верхнего уровня Lab2\_3.v
5. Микросхема EP4CE6E22C8
6. Создать модули на Verilog (имена указаны на структуре)
7. Создать модули на основе IP (имена указаны на структуре)
8. Создать модуль верхнего уровня иерархии на Verilog.
9. Осуществить компиляцию проекта в пакете Quartus, исправить ошибки и проверить с помощью RTL Viewer, что проект собран правильно.
10. Разработать тест - простейший тест (за основу можно взять тест из LAB2\_1).

## Описание на языке Verilog

Описание разрабатываемого устройства на языке Verilog приведено ниже листингах 1-1 – 1-7.

Листинг 1-1. lab3\_2.v

|  |
| --- |
| `timescale 1ns / 1ps  module lab3\_2 #(  parameter div = 100000000)(  input clk,  input dir,  input ena,  input rst,  output reg [6:0] digit,  output reg [7:0] an  );  wire [6:0] out0, out1;  reg [15:0] num\_indicate = 32'b0;  wire int\_rst, int\_ena, int\_dir;  CYNCin SYNCin\_RST(.clk(clk), .in(rst), .out(int\_rst));  CYNCin SYNCin\_ENA(.clk(clk), .in(ena), .out(int\_ena));  CYNCin SYNCin\_DIR(.clk(clk), .in(dir), .out(int\_dir));  wire cout;  CNT\_DIV #(div) CNT\_DIV\_inst(.clk(clk), .rst(int\_rst), .ena(int\_ena), .ena\_out(cout));  wire [5:0] cnt\_val;  CNT CNT\_inst(.clk(clk), .rst(int\_rst), .ena(cout & int\_ena), .dir(int\_dir), .out(cnt\_val));    wire [6:0] hex\_int [1:0];  CODER CODER\_inst(.clk(clk), .addr(cnt\_val), .hex\_data0(hex\_int[0]), .hex\_data1(hex\_int[1]));  CYNCout CYNCout\_inst(.clk(clk), .reset(int\_rst), .ena(int\_ena),  .in0(hex\_int[0]), .in1(hex\_int[1]),  .out0(out0), .out1(out1));  always @(posedge clk) begin  if (num\_indicate > 16'h8000) begin  digit <= out0;  an <= ~8'h01;  end else begin  digit <= out1;  an <= ~8'h02;  end  num\_indicate = num\_indicate + 16'h01;  end  endmodule |

Листинг 1-2. CNT.v

|  |
| --- |
| module CNT (  input clk, rst, ena, dir,  output [5:0] out  );  integer MAX\_VALUE = 17;  reg [5:0] count;  initial begin  count = 6'b0;  end  assign out = count;  always @(posedge clk)  begin  if(ena) begin  if(rst) begin  if(dir)  if(count != MAX\_VALUE)  count = count + 1'b1;  else  count = 6'b0;  else  if(count != 6'b0)  count = count - 1'b1;  else  count = MAX\_VALUE;  end else begin  count = 6'b0;  end  end  end  endmodule |

Листинг 1-3. CNT\_DIV.v

|  |
| --- |
| `timescale 1ns / 1ps  module CNT\_DIV #(  parameter div\_cnt = 50000000)(  input clk, rst, ena,  output reg ena\_out  );    reg [26:0] count;  initial begin  count = 10'b0;  ena\_out = 0'b0;  end  always @(posedge clk)  begin  if(ena) begin  if(rst) begin  if(count == div\_cnt-1) begin  count = 26'b0;  ena\_out = 1'b1;  end else begin  count = count + 1'b1;  ena\_out = 1'b0;  end  end else begin  count = 26'b0;  ena\_out = 1'b0;  end  end  end  endmodule |

Листинг 1-4. CODER.v

|  |
| --- |
| `timescale 1ns / 1ps  module CODER (  input clk,  input [5:0] addr,  output [6:0] hex\_data0, hex\_data1  );  wire [3:0] addr0, addr1;  blk\_mem\_gen\_0 ROM(.clka(clk), .clkb(clk), .addra(addr0), .douta(hex\_data0), .addrb(addr1), .doutb(hex\_data1));  assign addr1 = (addr==0) ? 0: addr/8'ha;  assign addr0 = (addr==0) ? 0: addr%8'ha;  endmodule |

Листинг 1-5. CYNCin.v

|  |
| --- |
| module CYNCin(  input clk, in,  output reg out  );  reg data;  always @(posedge clk) begin  data <= in;  out <= data;  end  endmodule |

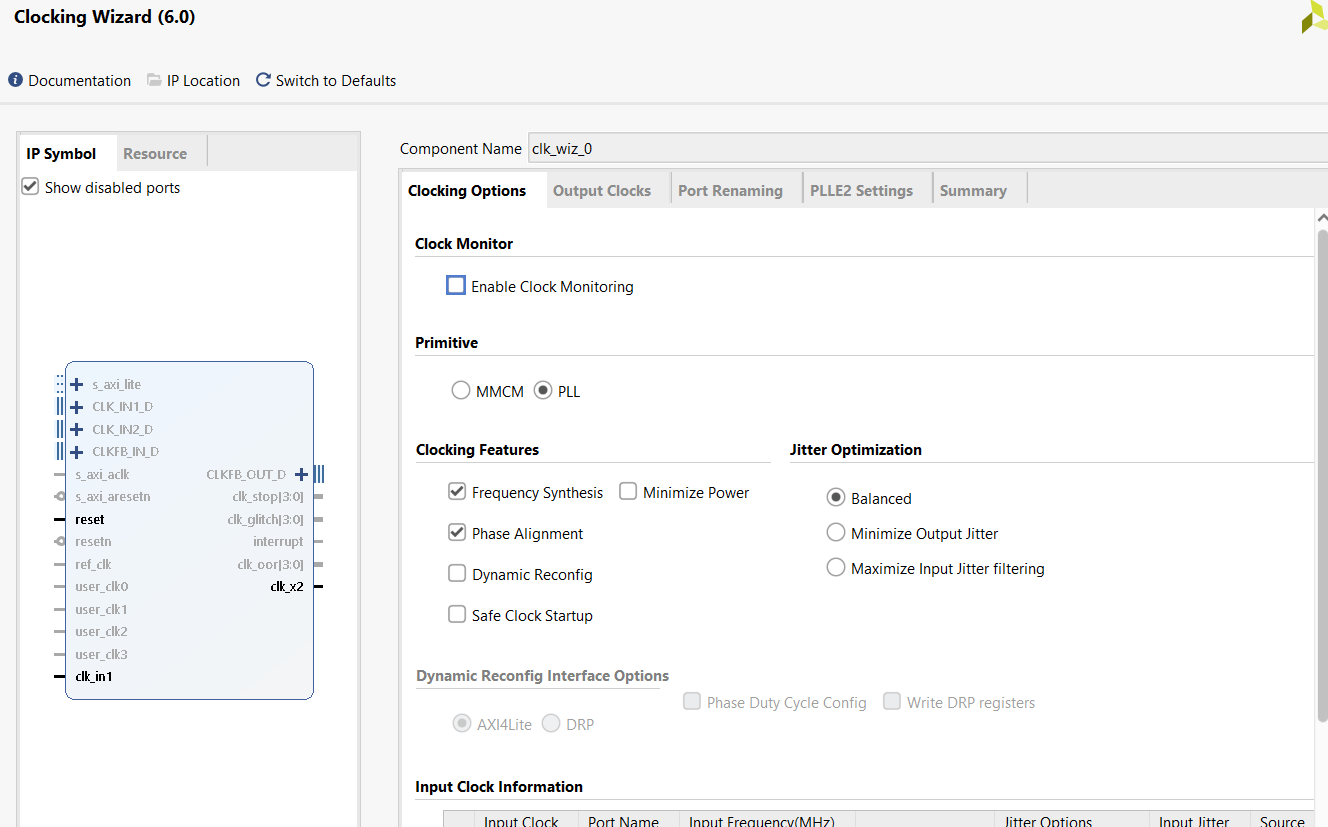
Листинг 1-6. CYNCout.v

|  |
| --- |
| module CYNCout (  input clk, reset, ena,  input [6:0] in0, in1,  output reg [6:0] out0, out1  );  always @(posedge clk, posedge reset) begin  if(reset)  if(ena) begin  out0 = in0;  out1 = in1;  end  else begin  out0 = 7'b0;  out1 = 7'b0;  end  end  endmodule |

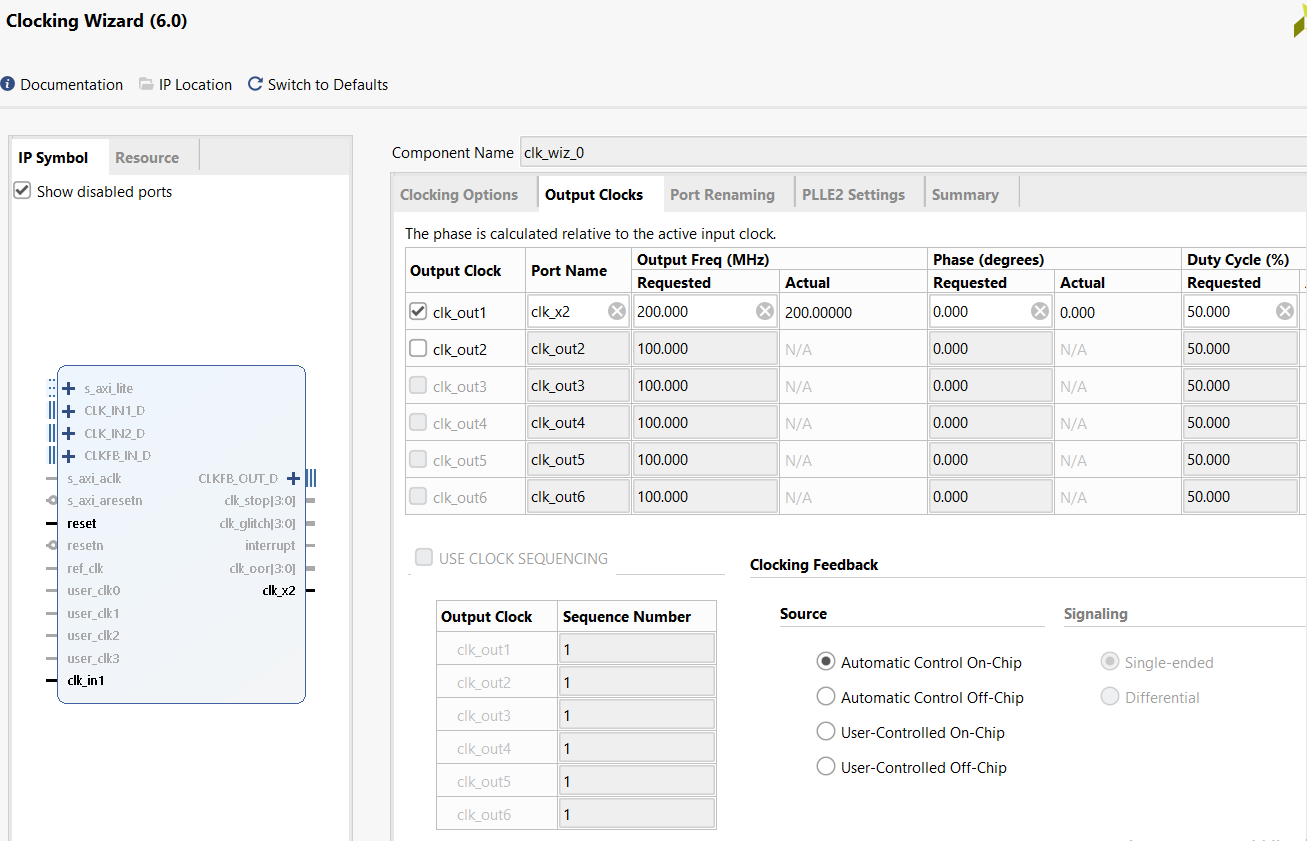
Листинг 1-6. db\_Lab3\_2.v

|  |
| --- |
| `timescale 1ns / 1ps  module db\_Lab3\_2(input clk);  localparam div = 4;  wire clk\_2x, dir, ena, rst;  wire [6:0] digit;  wire [7:0] an;  clk\_wiz\_0 wiz\_inst(.clk\_in1(clk), .clk\_2x(clk\_2x));  lab3\_2 #(div) lab\_inst(.clk(clk\_2x), .dir(dir), .ena(ena), .rst(rst), .digit(digit), .an(an));  vio\_0 vio\_inst(.clk(clk\_2x), .probe\_in0(digit), .probe\_in1(an), .probe\_in2(clk), .probe\_out0(dir), .probe\_out1(ena), .probe\_out2(rst));  ila\_0 ila\_inst(.clk(clk\_2x), .probe0(digit), .probe1(an), .probe2(clk));  endmodule |

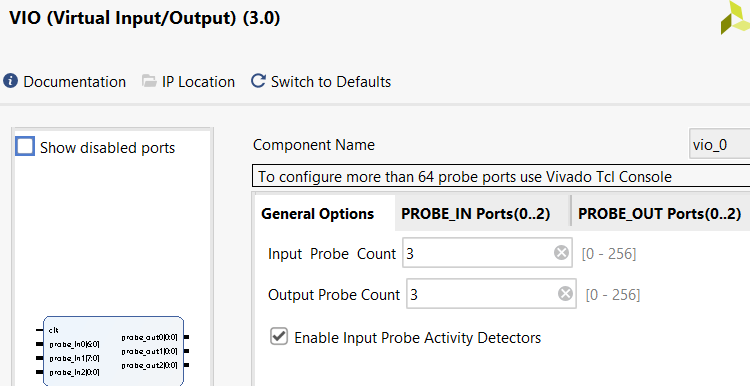
На рис. 1-1 – 1-6 приведены настройки ip ядер.



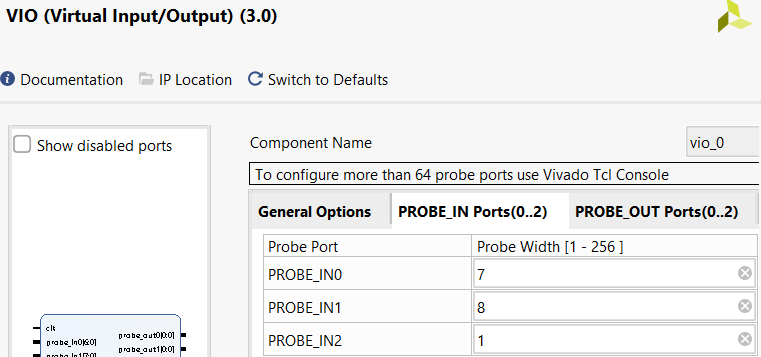
*Рис. 1-1. Настройка clocking wizard. Clocking Options*



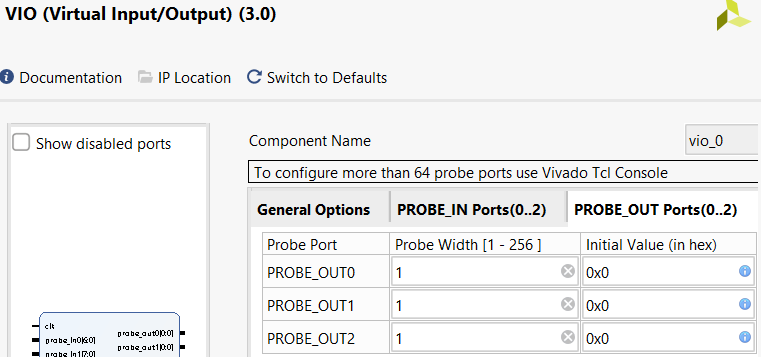
*Рис. 1-2. Настройка clocking wizard. Output Clocks*



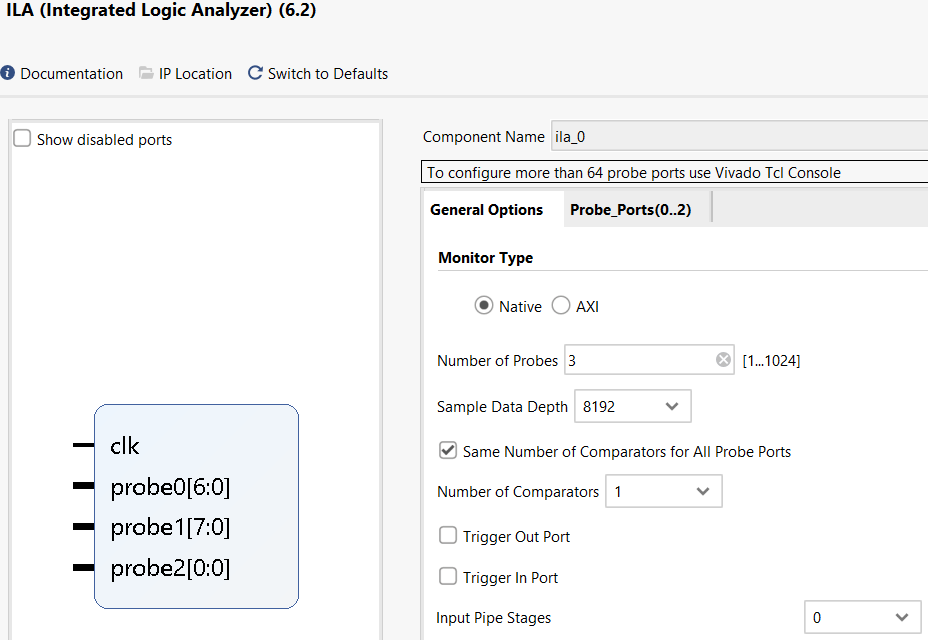
*Рис. 1-3. Настройка VIO. General*



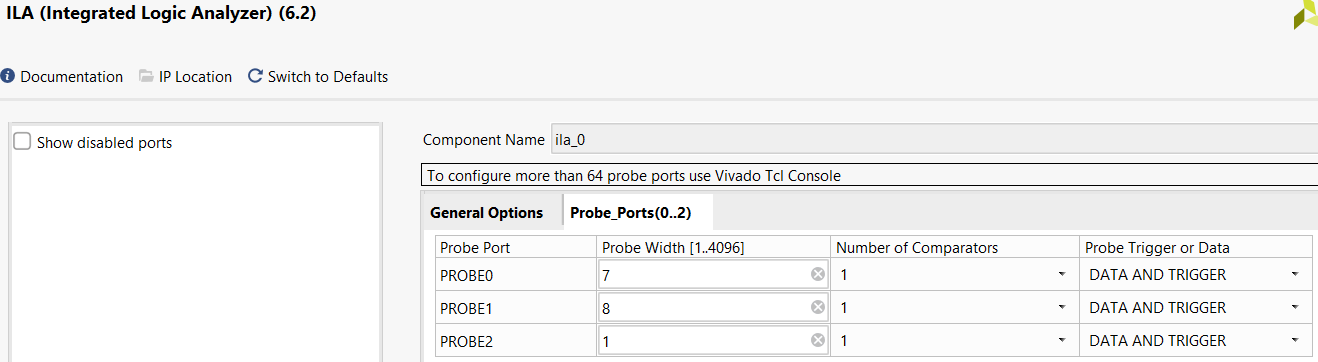
*Рис. 1-4. Настройка VIO. Inputs*



*Рис. 1-5. Настройка VIO. Output*



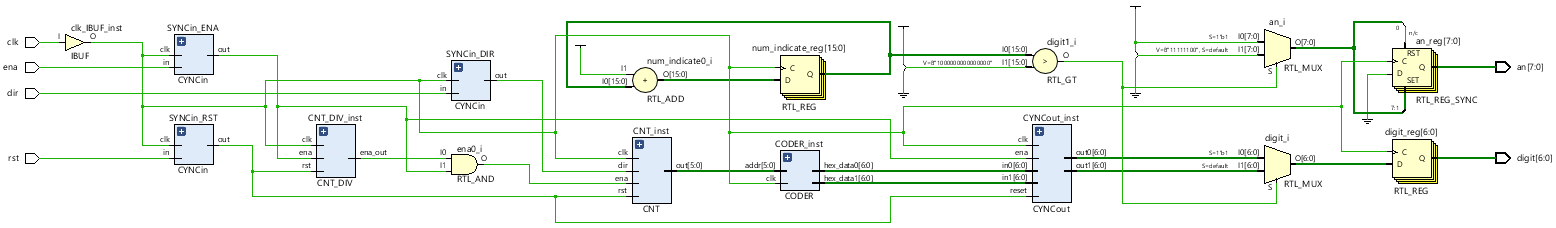
*Рис. 1-6. Настройка ILA. General*



*Рис. 1-6. Настройка ILA. Inputs*

## Результат синтеза (RTL)

Результат синтеза описания на языке Verilog приведен ниже, на Рис. 1‑7.



*Рис. 1-7. RTL схема*

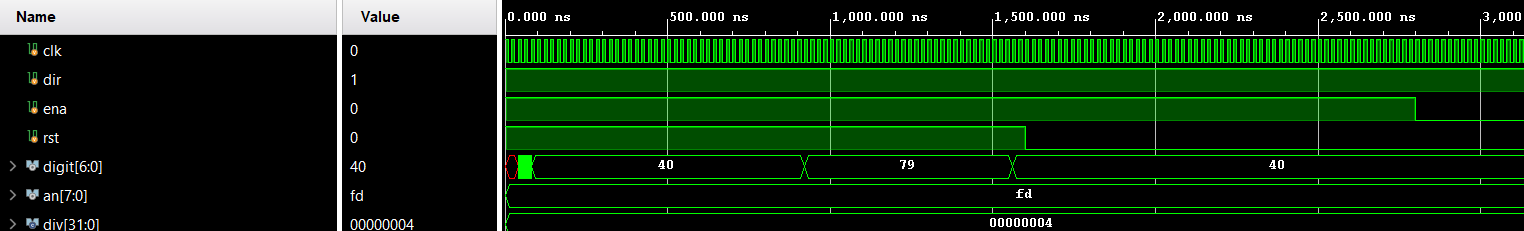
## Моделирование

Для проверки правильности работы созданного Verilog описания использовались тесты представленные в листингах 1-8. Результаты моделирования представленны на рис. 1-8.

Листинг 1-8. tb\_divider.v

|  |
| --- |
| `timescale 1ns / 1ps  module tb\_lab3\_2();  localparam div = 4;  reg clk = 1'b1, dir = 1'b1, ena = 1'b1, rst = 1'b1;  wire [6:0] digit;  wire [7:0] an;  lab3\_2 #(div) lab(.clk(clk), .dir(dir), .ena(ena), .rst(rst), .digit(digit), .an(an));  initial forever #10 clk = ~clk;  initial begin  #1600  rst = 1'b0;  #200  rst = 1'b0;  #1000  ena = 1'b0;  #400  $stop;  end  endmodule |

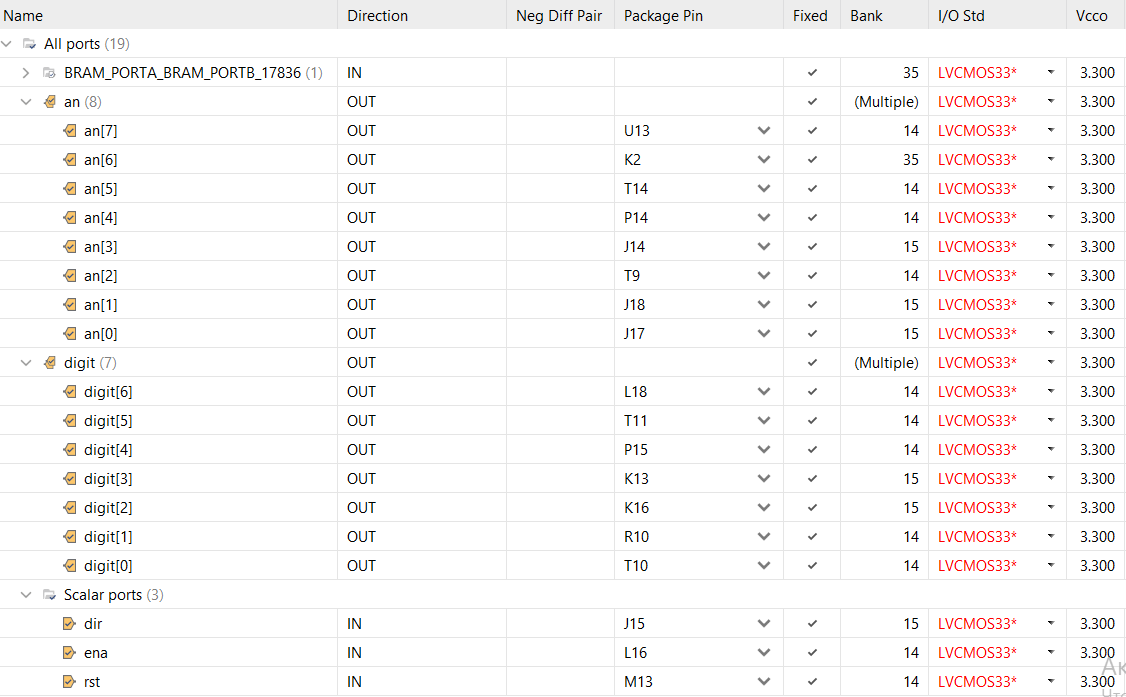
Результаты моделирования:



*Рис. 1-8. Результаты моделирования tb\_divider*

## Назначение выводов СБИС

Назначение выводов СБИС и стандартов ввода вывода приведено на Рис. 1‑10



*Рис. 1‑10 Назначение выводов*

## Отладка

Была проведена отладка проекта. Все результаты соответствуют ожиданиям.

## Выводы

В результате был создан проект по требуемому заданию.