САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ПЕТРА ВЕЛИКОГО

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

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчет

по заданию labs7-10

Дисциплина

«Языки описания аппаратных средств вычислительных систем»

выполнил:

Курякин Д. А

группа: 3540901/12001

преподаватель:

Антонов А. П.

Санкт-Петербург

2021

Оглавление

[1 Задание labs\_7 3](#_Toc90517316)

[1.1 Задание 3](#_Toc90517317)

[1.2 Описание на языке Verilog 3](#_Toc90517318)

[1.3 Результат синтеза 4](#_Toc90517319)

[1.4 Моделирование 4](#_Toc90517320)

[1.5 Назначение выводов СБИС 10](#_Toc90517321)

[1.6 Тестирование на плате Nexys4 DDR 10](#_Toc90517322)

[1.7 Выводы 10](#_Toc90517323)

[2 Задание labs\_8 11](#_Toc90517324)

[2.1 Задание 11](#_Toc90517325)

[2.2 Описание на языке Verilog 11](#_Toc90517326)

[2.3 Результат синтеза 12](#_Toc90517327)

[2.4 Моделирование 12](#_Toc90517328)

[2.5 Назначение выводов СБИС 16](#_Toc90517329)

[2.6 Тестирование на плате Nexys4 DDR 16](#_Toc90517330)

[2.7 Выводы 16](#_Toc90517331)

[3 Задание labs\_9 17](#_Toc90517332)

[3.1 Задание 17](#_Toc90517333)

[3.2 Описание на языке Verilog 17](#_Toc90517334)

[3.3 Результат синтеза 18](#_Toc90517335)

[3.4 Моделирование 19](#_Toc90517336)

[3.5 Назначение выводов СБИС 21](#_Toc90517337)

[3.6 Тестирование на плате Nexys4 DDR 22](#_Toc90517338)

[3.7 Выводы 22](#_Toc90517339)

[4 Задание labs\_10 23](#_Toc90517340)

[4.1 Задание 23](#_Toc90517341)

[4.2 Описание на языке Verilog 23](#_Toc90517342)

[4.3 Результат синтеза 24](#_Toc90517343)

[4.4 Моделирование 24](#_Toc90517344)

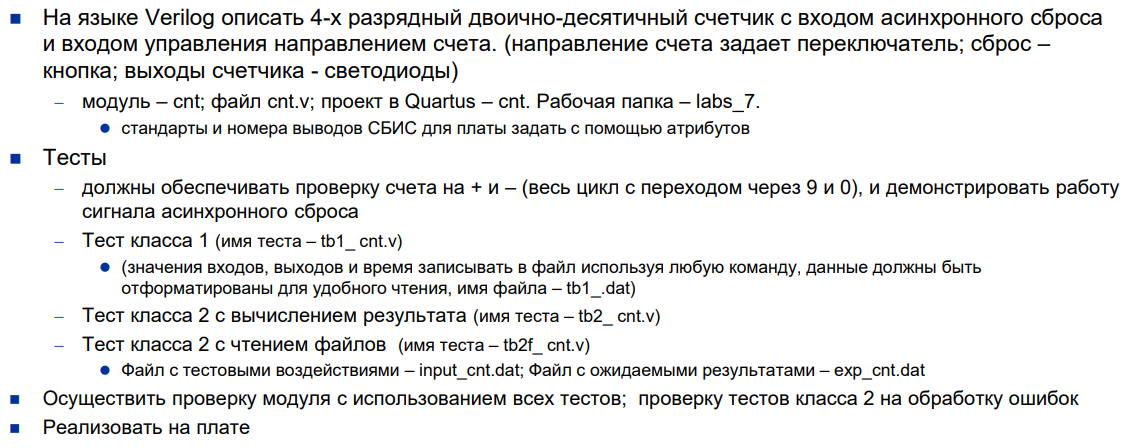
[4.5 Назначение выводов СБИС 28](#_Toc90517345)

[4.6 Тестирование на плате Nexys4 DDR 29](#_Toc90517346)

[4.7 Выводы 29](#_Toc90517347)

# Задание labs\_7

## Задание



## Описание на языке Verilog

Описание разрабатываемого устройства на языке Verilog приведено ниже в листинге 1-1.

Листинг 1-1. Labs\_7.v

|  |
| --- |
| module cnt(  input clk, dir, aclr,  output reg [3:0] result  );  // integer c = 2;  integer c = 50000000;  reg [27:0] count;  reg ena;    initial begin  result = 4'b0000;  count = 0;  ena = 1'b0;  end    always @(posedge ena, posedge aclr) begin  if (aclr)  result = 1'b0;  else begin  result = (dir) ? result + 1'b1 : result - 1'b1;    if (result == 4'b1010 & dir)  result = 1'b0;  if (result == 4'b1111 & ~dir)  result = 4'b1001;  end  end    always @ (posedge clk) begin  count = count + 1;  if(count >= c) begin  ena = 1'b1;  count = 0;  end else  ena = 1'b0;  end  endmodule |

## Результат синтеза

Результат синтеза описания на языке Verilog в пакете Quartus приведен ниже, на Рис. 1‑1.

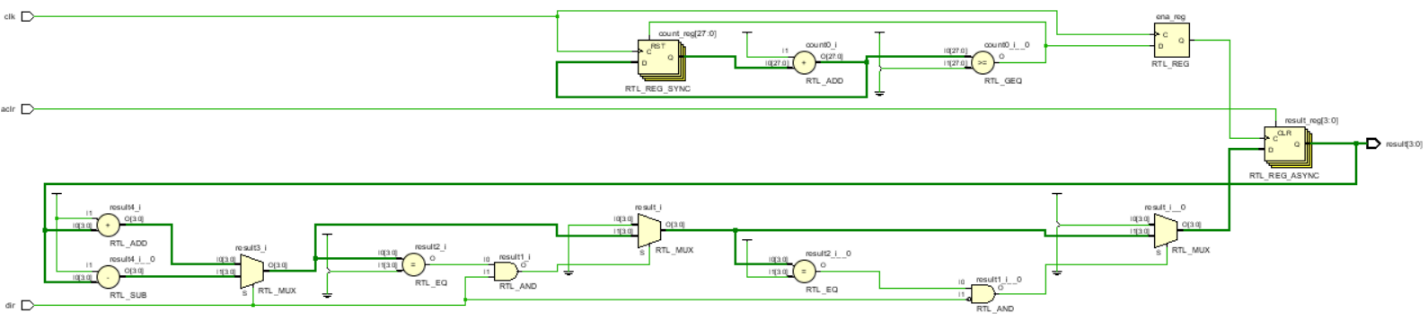


Рис. 1‑1 Синтезированная схема

## Моделирование

Для проверки правильности работы созданного Verilog описания использовались тесты:

Листинг 1-2. tb1\_cnt.v

|  |
| --- |
| `timescale 1ns / 1ps  module tb1\_cnt();  reg clk, dir, aclr;  wire [3:0] q;  integer i, j, file;  cnt fun(clk, dir, aclr, q);    initial begin: file\_init  file = $fopen("tb1\_.dat");  if (!file) $stop(2);  end  initial begin: generate\_clock  clk <= 1'b0;  forever #20 clk = !clk;  end  initial begin: init\_data  dir <= 1'b0;  aclr <= 1'b0;  end    initial begin:assert\_  forever begin  for (j = 0; j < 2; j = j + 1) begin  for (i = 0; i < 32; i = i + 1) begin  @ (posedge clk) begin  $fdisplay(file, "time: %t res: dir: %b aclr: %b data: %b",  $realtime, dir, aclr, q);  $display("time: %t res: dir: %b aclr: %b data: %b",  $realtime, dir, aclr, q);  end  end  aclr = ~aclr; #20; aclr = ~aclr;  dir = ~dir;  if(j != 1) begin  $fdisplay(file, "\*\*\*CHANGE DIRECTION\*\*\*");  $display("\*\*\*CHANGE DIRECTION\*\*\*");  end  end  $fdisplay(file, "\*\*\*TEST SUCESSFULL PASSED\*\*\*");  $display("\*\*\*TEST SUCESSFULL PASSED\*\*\*");  $fclose(file);  $stop;  end  end  endmodule |

Результаты моделирования приведены на Рис. 1‑2

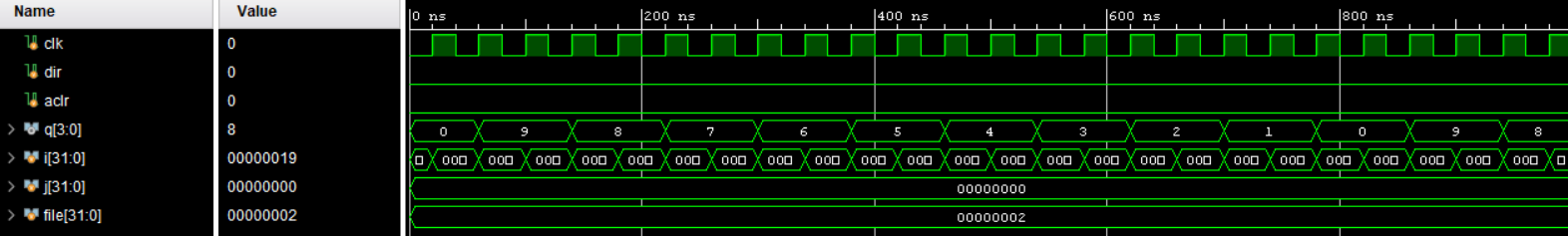


Рис. 1‑2 Результат моделирования

Листинг 1-3. tb2\_cnt.v

|  |
| --- |
| `timescale 1 ns/1 ps  module tb2\_cnt();  reg clk, dir, aclr;  wire [3:0] q;  reg [3:0] exp;  integer i, j, file;    integer c = 2;  reg [27:0] count;  cnt cnt2 ( clk, dir, aclr, q);  initial begin: file\_init  file = $fopen("tb2\_.dat");  if (!file) $stop(2);  end  initial begin: generate\_clock  clk <= 1'b0;  count = 28'b0;  forever #10 clk = !clk;  end  initial begin: init\_data  exp <= 1'b0;  dir <= 1'b0;  aclr <= 1'b0;  end  initial begin: compute\_exp    forever @ (posedge clk, posedge aclr) begin  if (aclr)  exp = 4'b0;  else begin    count = count + 1;  if(count >= c) begin  count = 0;  exp = (dir) ? exp + 1'b1 : exp - 1'b1;  if (exp == 4'b1010 & dir)  exp = 1'b0;  if (exp == 4'b1111 & ~dir)  exp = 4'b1001;  end  end  end  end    initial begin: assert\_  forever begin  for (j = 0; j < 2; j = j + 1) begin  for (i = 0; i < 32; i = i + 1) begin  @ (posedge clk) begin #20; assert\_equals; end  end  aclr = ~aclr; #40; aclr = ~aclr;  dir = ~dir;  if(j != 1)  $fdisplay(file, "\*\*\*CHANGE DIRECTION\*\*\*");  end  $fdisplay(file, "\*\*\*TEST SUCESSFULL PASSED\*\*\*");  $fclose(file);  $stop;  end  end  task assert\_equals;  if (q == exp) begin  $fdisplay(file, "time: %t res: dir: %b aclr: %b data: %b",  $realtime, dir, aclr, q);  end else begin  $fdisplay(file, "error on step %d time: %t expected: %b actual %b (aclr: %b ; dir: %b)",  j , $realtime, exp, q, aclr, dir);  $fclose(file);  $stop;  end  endtask  endmodule |

Результаты моделирования приведены на Рис. 1‑3

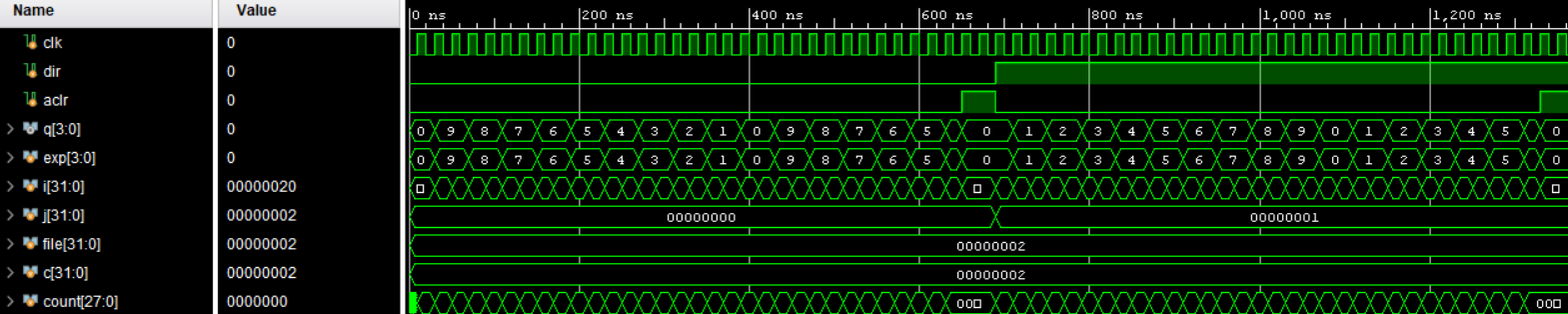


Рис. 1‑3 Результат моделирования

Листинг 1-4. tb2f\_cnt.v

|  |
| --- |
| `timescale 1 ns/1 ps  module tb2f\_cnt();  reg clk, dir, aclr;  wire [3:0] q;  reg [3:0] exp [0:31];  reg [3:0] c;  integer i, j, file, count;  cnt fun ( clk, dir, aclr, q);  initial begin: file\_init  file = $fopen("tb2f\_.dat");  $readmemh("exp\_cnt.mem", exp);  if (!file) $stop(2);  end  initial begin: generate\_clock  clk <= 1'b0;  count = 0;  c = exp[count];  forever #10 clk = !clk;  end  initial begin: init\_data  dir <= 1'b0;  aclr <= 1'b0;  end    initial begin: assert\_scope  forever begin  for (j = 0; j < 2; j = j + 1) begin  for (i = 0; i < 32; i = i + 1) begin  @ (posedge clk) begin  assert\_equals;  if(i[0])  count = count + 1;  c = exp[count];  end    end  aclr = ~aclr; #20; aclr = ~aclr;  dir = ~dir;  if(j != 1)  $fdisplay(file, "\*\*\*CHANGE DIRECTION\*\*\*");  end  $fdisplay(file, "\*\*\*TEST SUCESSFULL PASSED\*\*\*");  $fclose(file);  $stop;  end  end  task assert\_equals;  if (q == c)  success\_task;  else  err\_task;  endtask  task err\_task;  begin  $fdisplay(file, "error on step %d time: %t expected: %b actual %b (aclr: %b ; dir: %b)",  j , $realtime, exp[count], q, aclr, dir);  $fclose(file);  $stop;  end  endtask  task success\_task;  begin  $fdisplay(file, "time: %t res: dir: %b aclr: %b data: %b",  $realtime, dir, aclr, q);  end  endtask  endmodule |

Результаты моделирования приведены на Рис. 1‑4

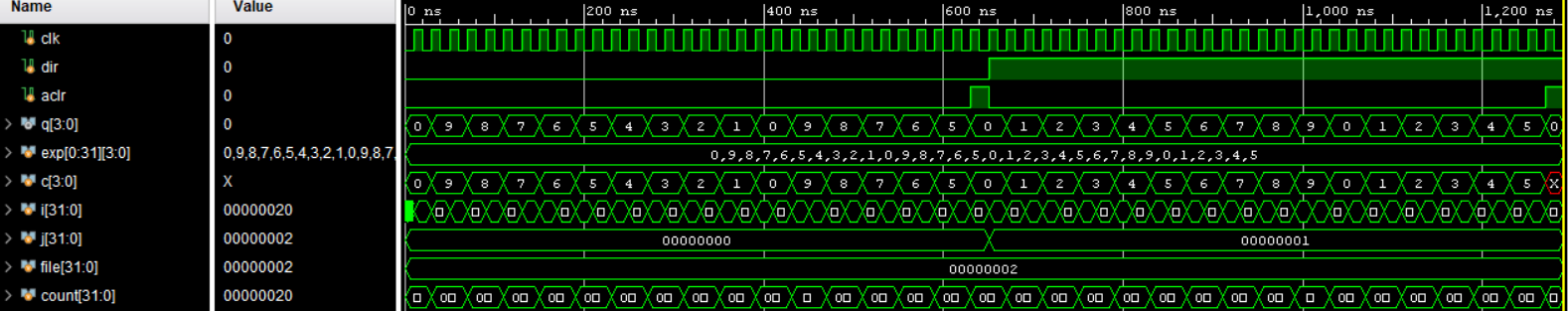


Рис. 1‑4 Результат моделирования

## Назначение выводов СБИС

Назначение выводов СБИС и стандартов ввода вывода приведено на Рис. 1‑5

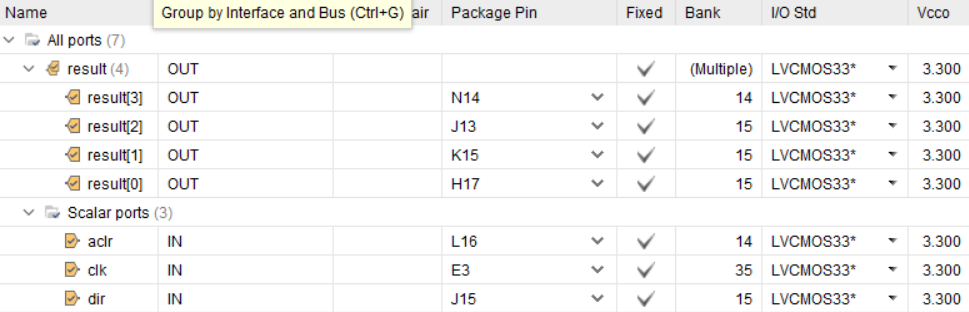


Рис. 1‑5 Назначение выводов

## Тестирование на плате Nexys4 DDR

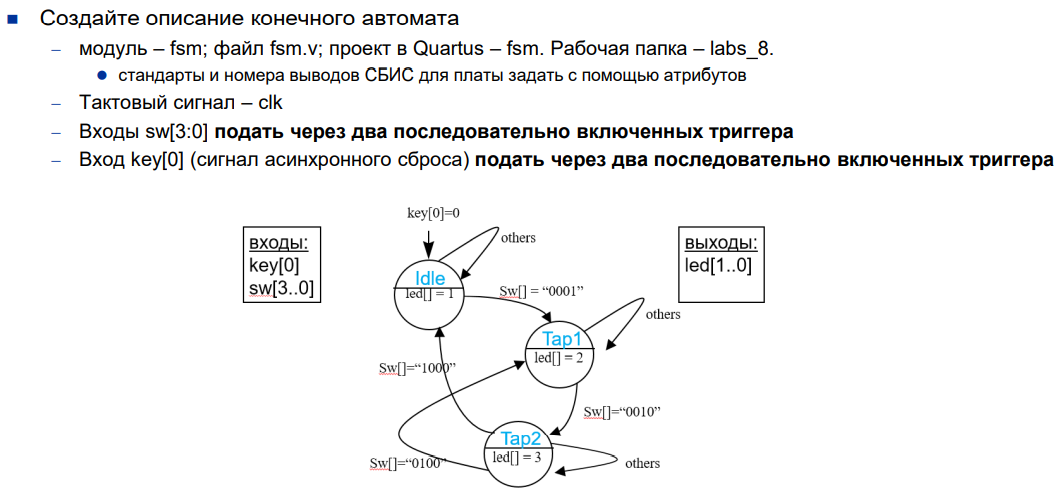
Для тестирования проекта на плате использовались тесты, описанные в разделе 1.4. Проведенное на плате Nexys4 DDR тестирование разработанного устройства показало: результаты совпадают с ожидаемыми, устройство работает в соответствии с заданием.

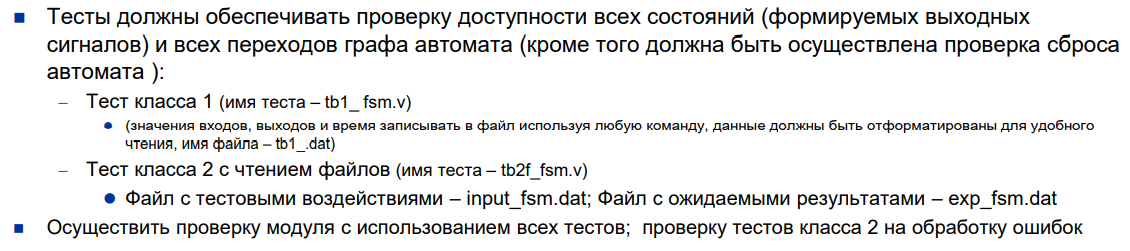
## Выводы

В результате было реализовано устройство, описанное в задании, и были написаны тесты, требуемые в задании.

# Задание labs\_8

## Задание





## Описание на языке Verilog

Описание разрабатываемого устройства на языке Verilog приведено ниже в листинге 2-1.

Листинг 2-1. fsm.v

|  |
| --- |
| module fsm (  input clock,  input aclr,  input [3:0] sw,  output reg [1:0] led  );  reg [1:0] state;  parameter s0 = 0, s1 = 1, s2 = 2;  initial begin  state = s0;  end  always @ (posedge clock, posedge aclr) begin  if (aclr)  state <= s0;  else  case (state)  s0 : if (sw == 4'h1) state <= s1;  s1 : if (sw == 4'h2) state <= s2;  s2 : begin  if (sw == 4'h8) state <= s0;  else  if (sw == 4'h4) state <= s1;  end  endcase  end    always @ (state) begin  case (state)  s0: led = 2'b01;  s1: led = 2'b10;  s2: led = 2'b11;  endcase  end  endmodule |

## Результат синтеза

Результат синтеза описания на языке Verilog приведен ниже на Рис. 2‑1.

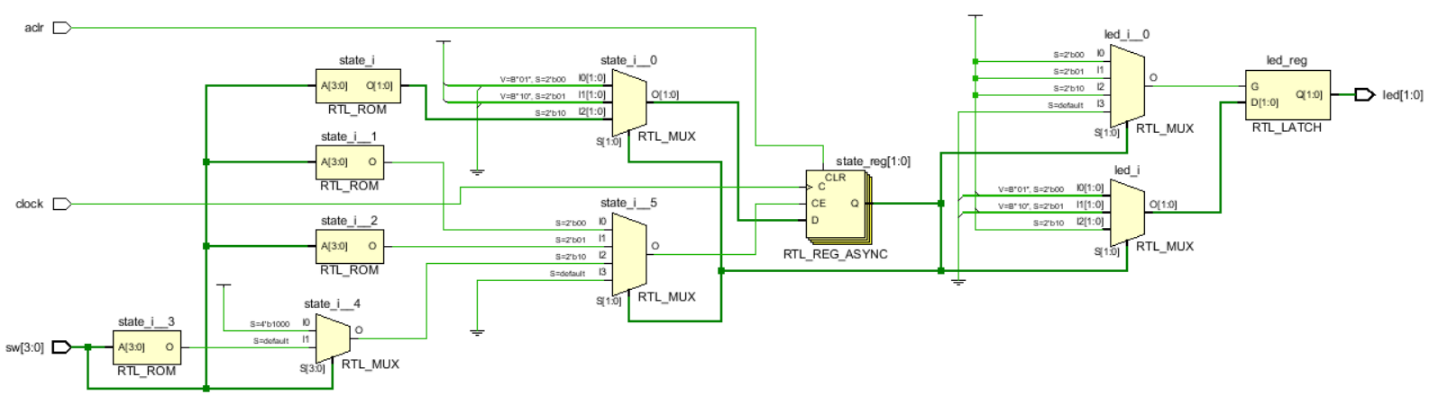


Рис. 2‑1 Синтезированная схема

## Моделирование

Для проверки правильности работы созданного Verilog описания использовались тесты:

Листинг 2-2. tb1\_fsm.v

|  |
| --- |
| module tb1\_fsm();  reg clk, aclr;  reg [3:0] sw;  wire [1:0] led;  integer file;  fsm tst (clk, aclr, sw, led);    initial begin: file\_init  file = $fopen("tb1\_.dat");  $fmonitor(file, "time %t aclr: %b sw: %b led %b", $realtime, aclr, sw, led);  if (!file) $stop(2);  end    initial begin: generate\_clock  $timeformat(-9, 0, "ns", 4);  clk <= 1'b0;  forever #5 clk = !clk;  end  initial begin: init\_data  aclr <= 1'b0;  sw <= 4'b0000;  end    initial begin:assert\_scope  #10; sw = 4'b0001;  #10; sw = 4'b0010;  #10; sw = 4'b0100;  #10; sw = 4'b1000;  #10; sw = 4'b0010;  #10; sw = 4'b1000;  #10; sw = 4'b0001;    #10; aclr = 1'b1;  #20; aclr = 1'b0;  #10; sw = 4'b0001;  #10; sw = 4'b0010;  #10; sw = 4'b0100;  #10; sw = 4'b1000;  #10; sw = 4'b0010;  #10; sw = 4'b1000;  #10; sw = 4'b0010;  $fdisplay(file, "\*\*\*TEST SUCESSFULL PASSED\*\*\*");  $fclose(file);  $stop;  end  endmodule |

Результаты моделирования приведены на Рис. 2-2.

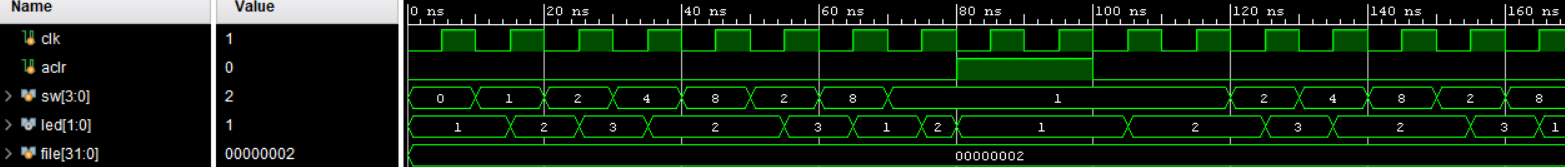


Рис. 2‑2 Результат моделирования

Листинг 2-3. tb2f\_fsm.v

|  |
| --- |
| `timescale 1ns / 1ps  module tb2f\_fsm();  reg clk, aclr;  reg [3:0] sw;  wire [1:0] led;  integer file;    reg [3:0] in[33:0];  reg [1:0] exp [16:0];  integer i;  fsm tst (clk, aclr, sw, led);    initial begin: file\_init  file = $fopen("tb2f\_.dat");  $readmemb("input\_fsm.mem", in);  $readmemb("exp\_fsm.mem", exp);  if (!file) $stop(2);  end    initial begin: generate\_clock  $timeformat(-9, 0, "ns", 4);  clk <= 1'b0;  forever #5 clk = !clk;  end  initial begin: init\_data  aclr <= 1'b0;  sw <= 4'b0000;  end    initial begin:assert\_scope    for(i=0;i<17;i=i+1) begin  #10; aclr = in[i\*2]; sw = in[i\*2+1]; assert\_equals;  end  $fdisplay(file, "\*\*\*TEST SUCESSFULL PASSED\*\*\*");  $fclose(file);  $stop;  end    task assert\_equals;  if (led == exp[i])  success\_task;  else  err\_task;  endtask    task err\_task;  begin  $fdisplay(file, "eror time %t aclr: %b sw: %b led %b exp %b", $realtime, aclr, sw, led, exp[i]);  $fclose(file);  $stop;  end  endtask    task success\_task;  begin  $fdisplay(file, "time %t aclr: %b sw: %b led %b", $realtime, aclr, sw, led);  end  endtask  endmodule |

Результаты моделирования приведены на Рис. 2-3.

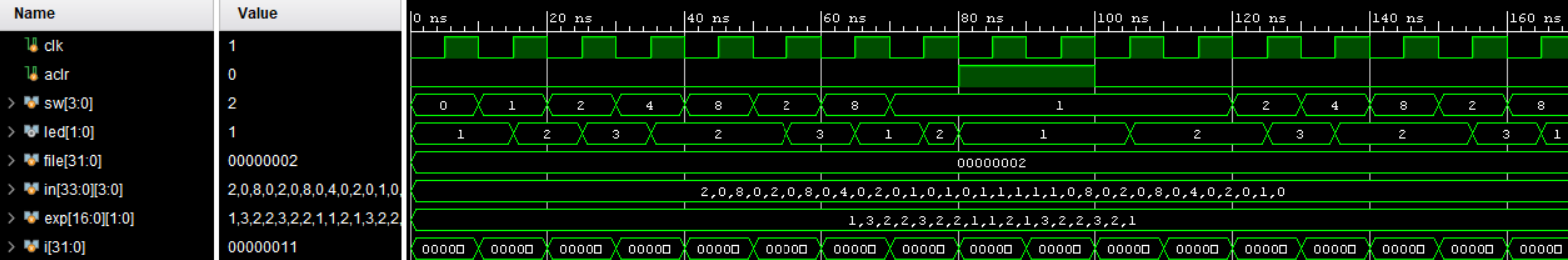


Рис. 2‑3 Результат моделирования

## Назначение выводов СБИС

Назначение выводов СБИС и стандартов ввода вывода приведено на Рис. 2‑4

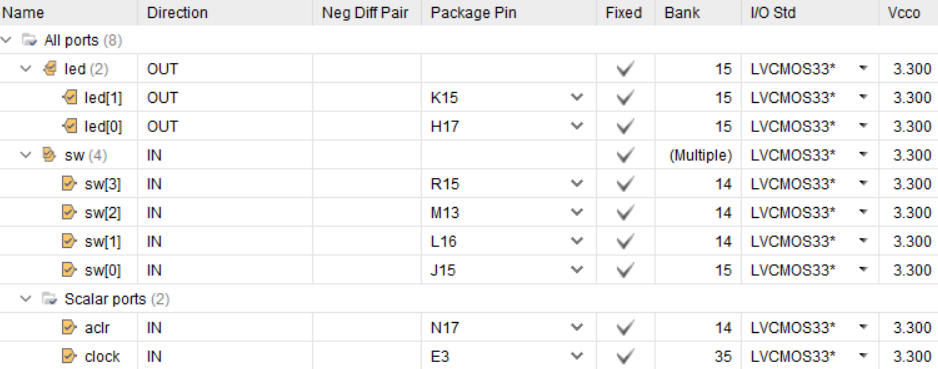


Рис. 2‑4 Назначение выводов

## Тестирование на плате Nexys4 DDR

Для тестирования проекта на плате использовались тесты, описанные в разделе 2.4

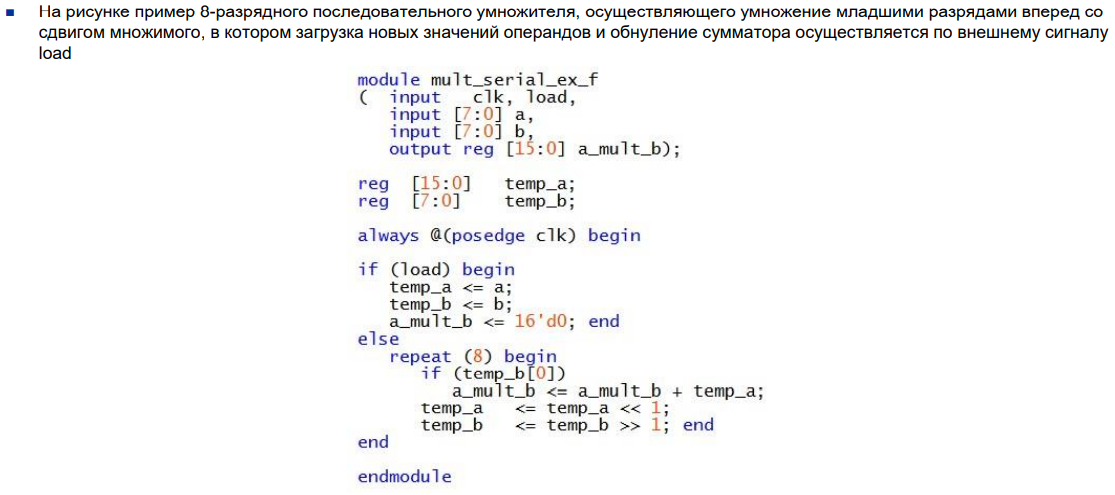
Проведенное на плате Nexys4 DDR тестирование разработанного устройства показало: результаты совпадают с ожидаемыми, устройство работает в соответствии с заданием.

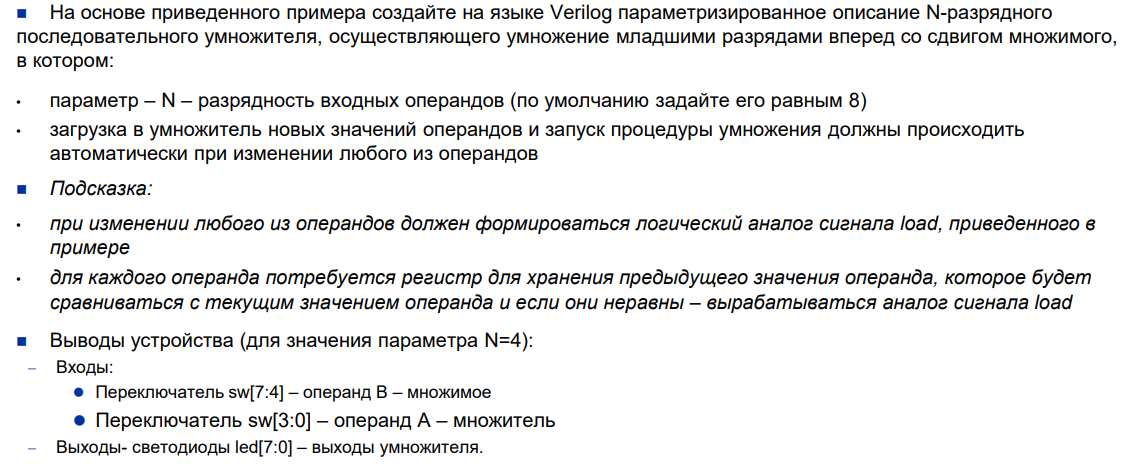
## Выводы

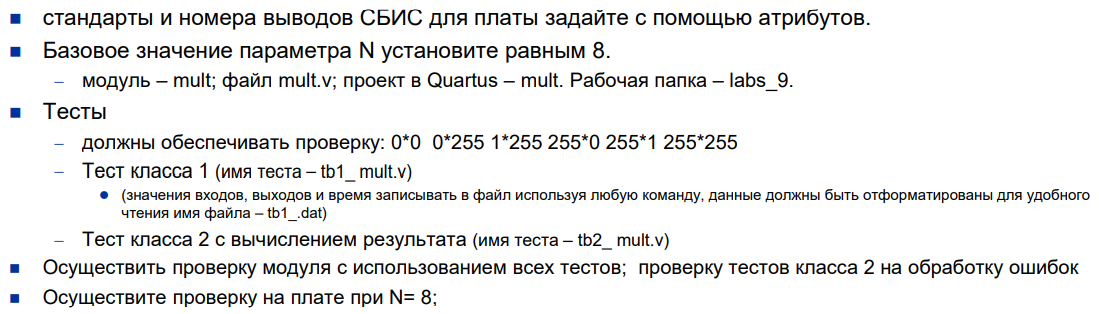
В результате было реализовано устройство, описанное в задании, и были написаны тесты, требуемые в задании.

# Задание labs\_9

## Задание







## Описание на языке Verilog

Описание разрабатываемого устройства на языке Verilog приведено ниже листинге 3-1.

Листинг 3-1. mult.v

|  |
| --- |
| module mult #(  parameter N = 8  )(  input clk,  input [N - 1:0] a, b,  output reg [N\*2 - 1:0] a\_mult\_b  );    reg [N - 1:0] a\_, b\_;  reg [N\*2 - 1:0] temp\_a;  reg [N - 1:0] temp\_b;  initial begin  a\_ <= {N{1'b0}}; b\_ <= {N{1'b0}};  end  always @(posedge clk)  begin  temp\_a = a;  temp\_b = b;  if (a\_ != a || b\_ != b) begin  a\_mult\_b = {N\*2{1'b0}};  repeat (N) begin  if (temp\_b[0])  a\_mult\_b = a\_mult\_b + temp\_a;  temp\_a = temp\_a << 1;  temp\_b = temp\_b >> 1;  end  a\_ <= a; b\_ <= b;  end  end  endmodule |

## Результат синтеза

Результат синтеза описания на языке Verilog приведен ниже на Рис. 3‑1.

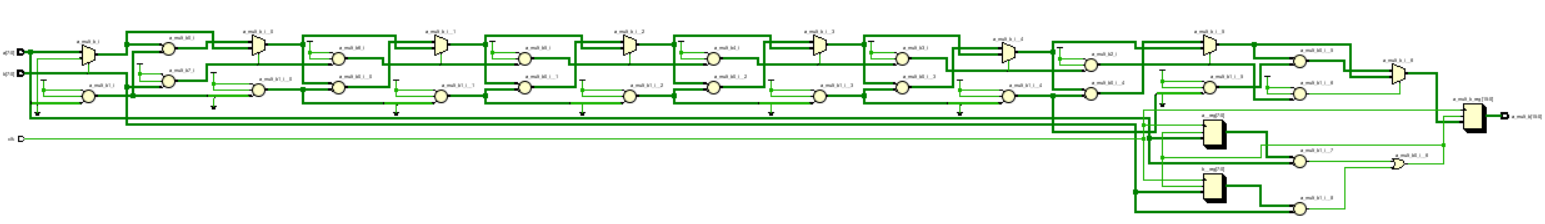


Рис. 3‑1 Синтезированная схема

## Моделирование

Для проверки правильности работы созданного Verilog описания использовались тесты:

Листинг 3-2. tb1\_mult.v

|  |
| --- |
| `timescale 1ns / 1ps  module tb1\_mult ();  localparam N = 8;  reg clk;  reg [7:0] a, b;  wire [15:0] a\_mult\_b;  integer i;  integer file;  initial begin  file = $fopen("tb1\_.dat");  end  mult #(N) L(clk, a, b, a\_mult\_b);    initial begin : enerate\_clk  clk <= 1'b0;  forever #10 clk = !clk;  end    initial begin  a = 8'h01; b = 8'hff; #40; assert\_equals;  a = 8'hff; b = 8'h01; #40; assert\_equals;  a = 8'h0; b = 8'h0; #40; assert\_equals;  a = 8'h0; b = 8'hff; #40; assert\_equals;  a = 8'hff; b = 8'h0; #40; assert\_equals;  a = 8'hff; b = 8'hff; #40; assert\_equals;  for(i = 1024;i<1044;i=i+1) begin  {a, b} = i;  #40;  assert\_equals;  end  $fclose(file);  $stop;  end  task assert\_equals;  begin  $fdisplay(file, "%d \* %d = %d", a, b, a\_mult\_b);  $display("%h", a\_mult\_b);    end  endtask  endmodule |

Результаты моделирования приведены на Рис. 3-2.

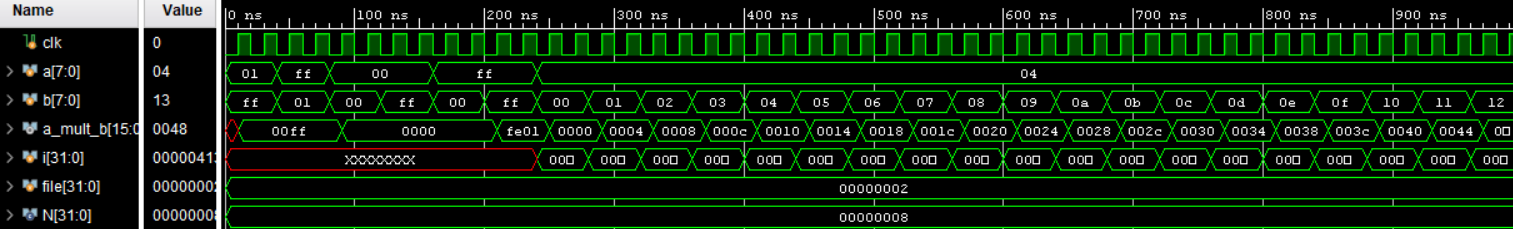


Рис. 3‑2 Результат моделирования

Фрагмент вывода в файл:

1 \* 255 = 255

255 \* 1 = 255

0 \* 0 = 0

0 \* 255 = 0

255 \* 0 = 0

255 \* 255 = 65025

4 \* 0 = 0

4 \* 1 = 4

…..

4 \* 18 = 72

4 \* 19 = 76

\*\*\*TESTS PASSED, YES IT'S TRUE\*\*\*

Листинг 3-3. tb2\_mult.v

|  |
| --- |
| `timescale 1 ns/ 1 ps  module tb2\_mult();  parameter N = 8;  reg clk;  reg [N - 1:0] a, b;  reg [N\*2 - 1:0] exp;  wire [N\*2 - 1:0] a\_mult\_b;  integer file;  integer i;    mult #(N) fun (clk, a, b, a\_mult\_b);    initial begin  file = $fopen("tb2\_.dat");  end    initial begin: generate\_clock  clk <= 1'b0;  forever #20 clk = !clk;  end    initial begin:assert\_scope  a = 8'h01; b = 8'hff; #40; assert\_equals;  a = 8'hff; b = 8'h01; #40; assert\_equals;  a = 8'h0; b = 8'h0; #40; assert\_equals;  a = 8'h0; b = 8'hff; #40; assert\_equals;  a = 8'hff; b = 8'h0; #40; assert\_equals;  a = 8'hff; b = 8'hff; #40; assert\_equals;  for(i = 1024;i<1044;i=i+1) begin  {a, b} = i;  #40;  assert\_equals;  end  $fdisplay(file, "\*\*\*TESTS PASSED, YES IT'S TRUE\*\*\*");  $fclose(file);  $stop;  end  task assert\_equals;  begin    exp = a \* b;    if (exp == a\_mult\_b)  $fdisplay(file, "%d \* %d = %d", a, b, a\_mult\_b);  else  begin  $fdisplay(file, "\*\*\*OOPS, THERE IS ERRROR\*\*\*");  $fdisplay(file, "%d \* %d = %d", a, b, a\_mult\_b);  $fdisplay(file, "Expected result is %d, but actual is %d", exp, a\_mult\_b);  $fclose(file);  $stop;  end  end  endtask  endmodule |

Результаты моделирования приведены на Рис. 3-3.

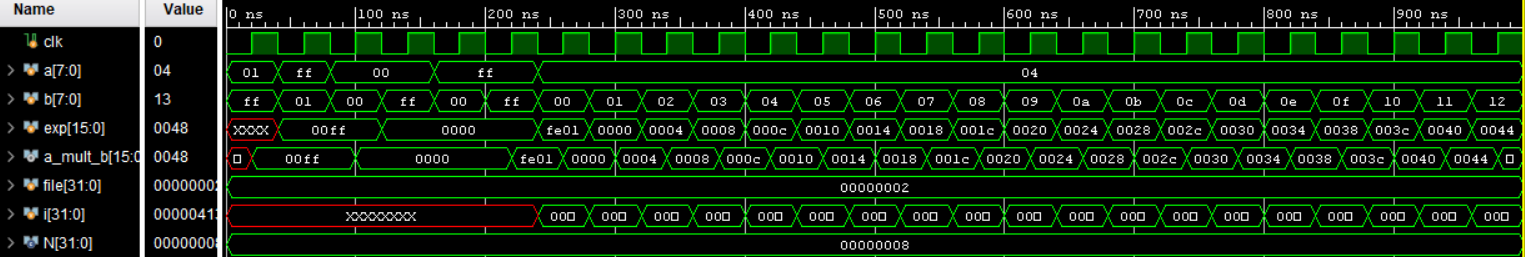


Рис. 3‑3 Результат моделирования

Фрагмент вывода в файл:

1 \* 255 = 255

255 \* 1 = 255

0 \* 0 = 0

0 \* 255 = 0

255 \* 0 = 0

255 \* 255 = 65025

4 \* 0 = 0

4 \* 1 = 4

…..

4 \* 18 = 72

4 \* 19 = 76

\*\*\*TESTS PASSED, YES IT'S TRUE\*\*\*

## Назначение выводов СБИС

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



Рис. 3‑3 Назначение выводов

## Тестирование на плате Nexys4 DDR

Для тестирования проекта на плате использовались тесты, описанные в разделе 3.4

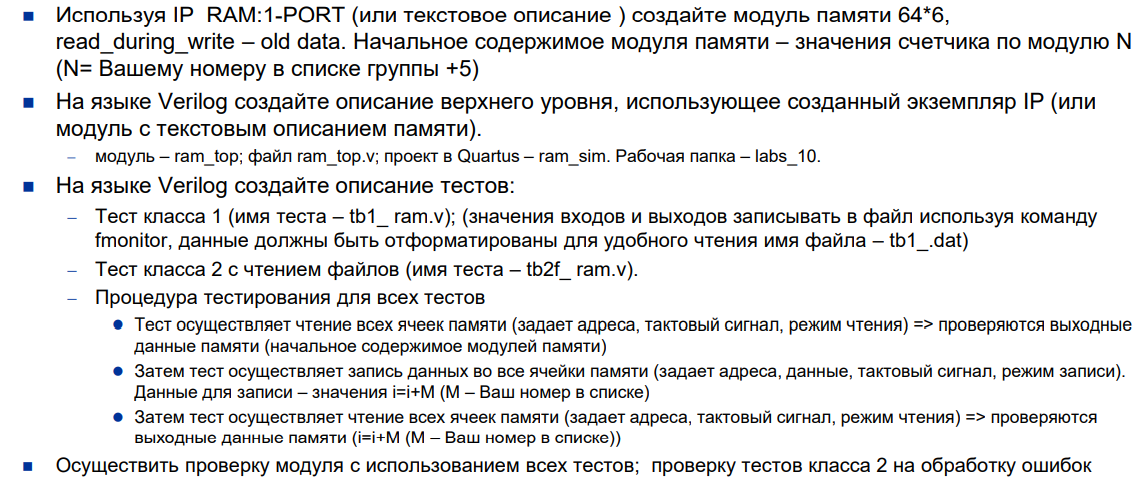
Проведенное на плате Nexys4 DDR тестирование разработанного устройства показало: результаты совпадают с ожидаемыми, устройство работает в соответствии с заданием.

## Выводы

В результате было реализовано устройство, описанное в задании, и были написаны тесты, требуемые в задании. Так же был получен опыт предотвращения дребезга контактов.

# Задание labs\_10

## Задание



## Описание на языке Verilog

Описание разрабатываемого устройства на языке Verilog приведено ниже листинге 4-1.

Листинг 4-1. ram\_top.v

|  |
| --- |
| module ram\_top #(  parameter N = 14  )(  input clk, read, write,  input [5:0] adr,  input [5:0] data,  output [5:0] led  );  reg [5:0] mem [63:0];  integer i;  initial begin  for(i = 0; i < 64; i = i + 1)  mem[i] = i+N;  end  always @(posedge clk) begin  if(write)  mem[adr] = data;  end  assign led = (read)? mem[adr]: 0;  endmodule |

## Результат синтеза

Результат синтеза описания на языке Verilog приведен ниже на Рис. 3‑1.

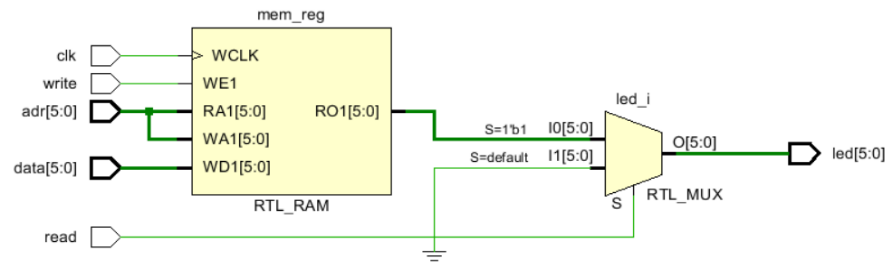


Рис. 4‑1 Синтезированная схема

## Моделирование

Для проверки правильности работы созданного Verilog описания использовались тесты:

Листинг 4-2. tb1\_ram.v

|  |
| --- |
| module tb1\_ram();  localparam N = 4;  reg clk, read, write;  reg [5:0] adr;  reg [5:0] data;  wire [5:0] led;  integer file;  integer i;  ram\_top #(N) LAB(clk, read, write, adr, data, led);  initial begin  file = $fopen("tb1\_.dat");  $fmonitor(file, "time=%t read = %b write = %b adr = %b data = %b, led = %b", $time, read, write, adr, data,led);  $monitor("time=%t read = %b write = %b adr = %b data = %b led = %b", $time, read, write, adr, data,led);  end    initial begin: generate\_clock  clk <= 1'b0;  forever #10 clk = !clk;  end    initial begin: init  write <= 1'b0;  adr = 1'b0;  end    initial begin:assert\_scope  read = 1'b1;  for(i=0;i<10;i=i+1) begin  @ (posedge clk)  adr = i;  end  read = 1'b0;  write = 1'b1;  for(i=0;i<10;i=i+1) begin  @ (posedge clk) begin  adr = i;  data = i;  end  end  read = 1'b1;  write = 1'b0;  adr = 0;  for(i=0;i<10;i=i+1) begin  @ (posedge clk)  adr = i;  end  $fdisplay(file, "\*\*\*TESTS PASSED, YES IT'S TRUE\*\*\*");  $fclose(file);  $stop;  end  endmodule |

Результаты моделирования приведены на Рис. 4-2.

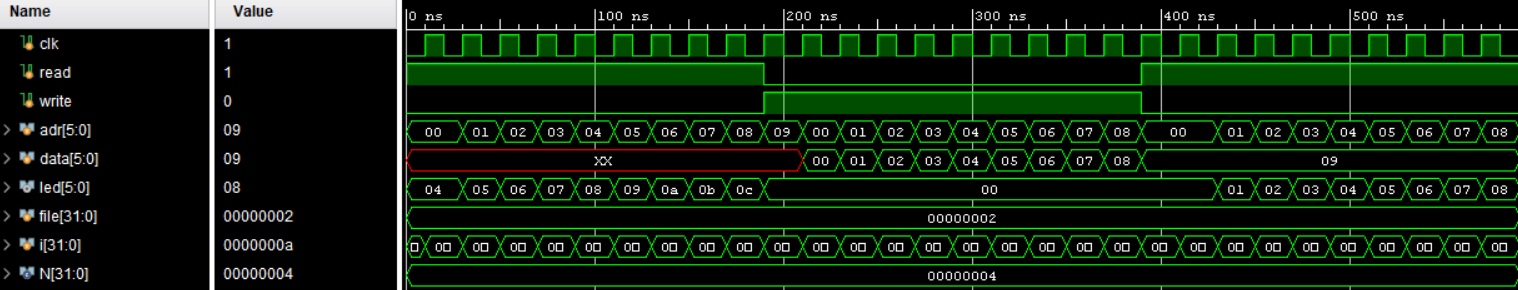


Рис. 4‑2 Результат моделирования

Часть вывода в файл:

time= 0 read = 1 write = 0 adr = 000000 data = xxxxxx, led = 000100

time= 30000 read = 1 write = 0 adr = 000001 data = xxxxxx, led = 000101

…..

time= 550000 read = 1 write = 0 adr = 000111 data = 001001, led = 000111

time= 570000 read = 1 write = 0 adr = 001000 data = 001001, led = 001000

\*\*\*TESTS PASSED, YES IT'S TRUE\*\*\*

Листинг 4-3. tb2f\_ram.v

|  |
| --- |
| `timescale 1ns / 1ps  module tb2f\_ram();  localparam N = 4;  reg clk, read, write;  reg [5:0] adr;  reg [5:0] data;  wire [5:0] led;  integer file;  reg [5:0] exp [30:0];  reg [5:0] e;  integer i, count;  ram\_top #(N) LAB(clk, read, write, adr, data, led);  initial begin: init\_file  file = $fopen("tb2f\_.dat", "w");  $readmemh("exp\_ram.mem", exp);  end    initial begin: generate\_clock  clk <= 1'b0;  forever #10 clk = !clk;  end    initial begin: init  write <= 1'b0;  adr = 1'b0;  count = 0;  end    initial begin:assert\_scope  read = 1'b1;  for(i=0;i<10;i=i+1) begin  @ (posedge clk) begin  adr = i;  count = count + 1;  e = exp[count];  assert\_equals;  end  end  read = 1'b0;  write = 1'b1;  for(i=0;i<10;i=i+1) begin  @ (posedge clk) begin  adr = i;  data = i;  count = count + 1;  e = exp[count];  assert\_equals;  end  end  read = 1'b1;  write = 1'b0;  adr = 0;  for(i=0;i<10;i=i+1) begin  @ (posedge clk) begin  adr = i;  count = count + 1;  e = exp[count];  assert\_equals;  end  end  $fdisplay(file, "\*\*\*TESTS PASSED, YES IT'S TRUE\*\*\*");  $display("\*\*\*TESTS PASSED, YES IT'S TRUE\*\*\*");  $fclose(file);  $stop;  end    task assert\_equals;  if (led == e)  success\_task;  else  err\_task;  endtask    task err\_task;  begin  $fdisplay(file, "error on time: %t expected: %h actual %h (read: %b; write: %b; adr: %h, data: %h)",  $realtime, e, led, read, write, adr, data);    $display("error on time: %t expected: %h actual %h (read: %b; write: %b; adr: %h, data: %h)",  $realtime, e, led, read, write, adr, data);  $fclose(file);  $stop;  end  endtask    task success\_task;  begin  $fdisplay(file, "time: %t read: %b; write: %b; adr: %h; data: %h; led: %h",  $realtime, read, write, adr, data, led);  $display("time: %t read: %b; write: %b; adr: %h; data: %h; led: %h",  $realtime, read, write, adr, data, led);  end  endtask  endmodule |

Результаты моделирования приведены на Рис. 4-3.

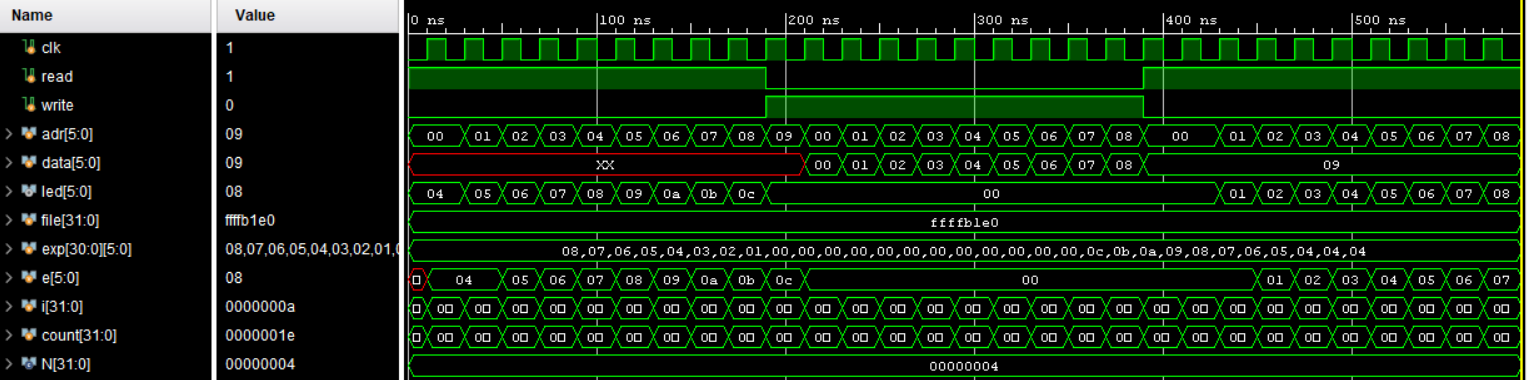


Рис. 4‑3 Результат моделирования

## Назначение выводов СБИС

Назначение выводов СБИС и стандартов ввода вывода приведено на Рис. 4‑4.

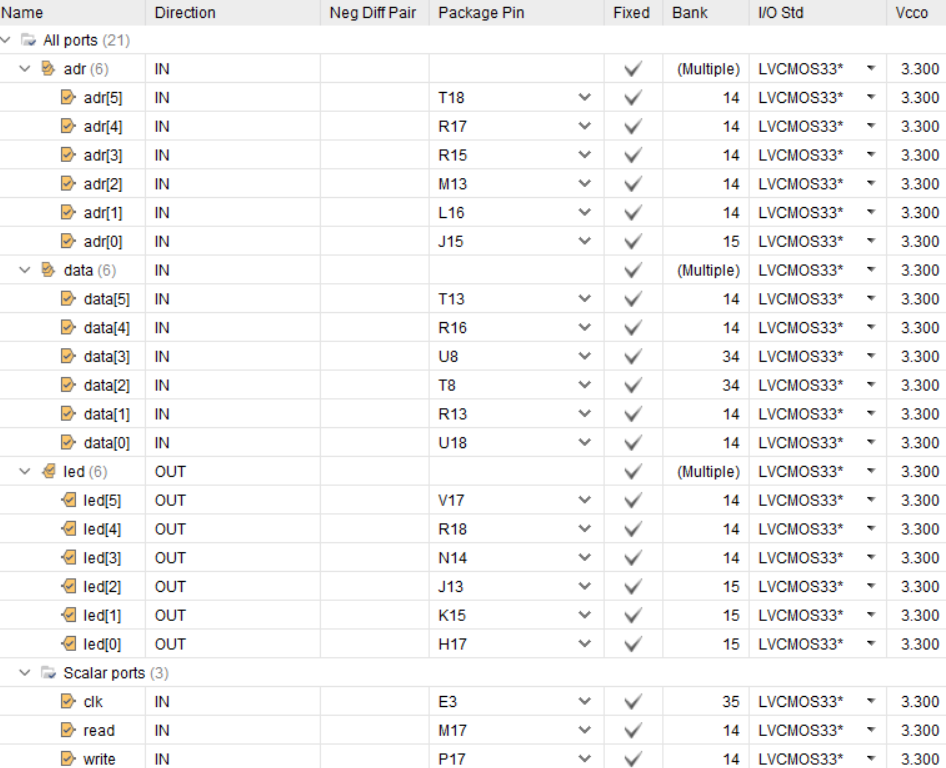


Рис. 4‑4 Назначение выводов

## Тестирование на плате Nexys4 DDR

Для тестирования проекта на плате использовались тесты, описанные в разделе 4.4

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 задании, и были написаны тесты, требуемые в задании.