САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ПЕТРА ВЕЛИКОГО

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

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по курсовому проекту

на тему «Процедура создания тестов и моделирования»

Дисциплина

«Технологии проектирования аппаратных средств компьютерных систем»

выполнил: Шаменов А.А.

группа: 13541/1

преподаватель: Антонов А. П.

Санкт-Петербург

2019

Оглавление

[Введение 3](#_Toc12214754)

[**Тесты класса I.** 3](#_Toc12214755)

[**Тесты класса II.** 4](#_Toc12214756)

[**Инструментарий** 4](#_Toc12214757)

[Задание 1 5](#_Toc12214758)

[1.1 Задание 5](#_Toc12214759)

[1.2 Описание на языке Verilog 5](#_Toc12214760)

[1.3 Тестирование на плате 5CSEMA5F31C6 11](#_Toc12214761)

[2 Задание 2 12](#_Toc12214762)

[2.1 Задание 12](#_Toc12214763)

[2.2 Описание на языке Verilog 12](#_Toc12214764)

[2.3 Тестирование на плате 5CSEMA5F31C6 18](#_Toc12214765)

[3 Задание 3 19](#_Toc12214766)

[3.1 Задание 19](#_Toc12214767)

[3.2 Описание на языке Verilog 19](#_Toc12214768)

[3.3 Тестирование на плате 5CSEMA5F31C6 26](#_Toc12214769)

[4 Задание 4 27](#_Toc12214770)

[4.1 Задание 27](#_Toc12214771)

[4.2 Описание на языке Verilog 27](#_Toc12214772)

[4.3 Тестирование на плате 5CSEMA5F31C6 34](#_Toc12214773)

[5 Задание 5 35](#_Toc12214774)

[5.1 Задание 35](#_Toc12214775)

[5.2 Описание на языке Verilog 35](#_Toc12214776)

[5.3 Тестирование на плате 5CSEMA5F31C6 43](#_Toc12214777)

[СПИСОК ИСТОЧНИКОВ 44](#_Toc12214778)

# Введение

Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом [1].

По степени автоматизации тестирование принято разделять на ручное и автоматизированное.

Ручное тестирование проводится тестировщиком (разработчиком) без использования программных средств, для проверки устройства или программного продукта путем моделирования действий пользователя. Очевидным недостатком ручного тестирования является сложность покрытия всех возможных действий.

Автоматизированное тестирование использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс.

В данной работе использованы два вида тестовых воздействий:

**Тесты класса I.**

Такие тесты просты для написания и быстрой проверки небольших проектов. Тест класса I состоит из собственно тестируемого модуля, набора тестовых воздействий и средств визуализации результатов моделирования. Проверка результатов – чаще всего, визуальная – осуществляется вручную. Схема модуля тестирования первого класса представлена на рисунке 1.

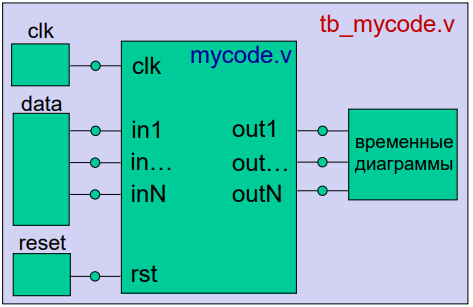


Рисунок 1 – Модуль тестирования класса I.

**Тесты класса II.**

Отличие тестов второго класса заключается в том, что данные, полученные с выходов тестируемого модуля, сравниваются с заранее описанными ожидаемыми значениями (рисунок 2).

Такие тесты сложнее в написании, однако, имеют ряд преимуществ, таких как возможность повторного использования инженерами, не знакомыми с тестируемым кодом, а также минимизируют человеческий фактор, т.к. анализ осуществляется автоматически.

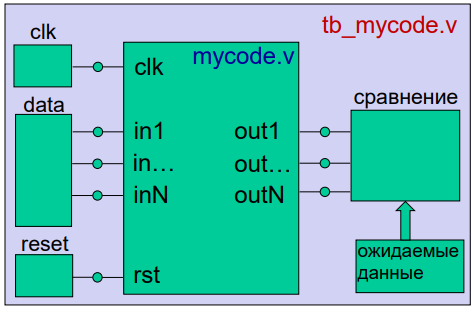


Рисунок 2 – Модуль тестирования класса II.

**Инструментарий**

В данной работе рассматривается тестирование программ, написанных для ПЛИС Intel Cyclone V [2], реализованных на плате DE-1 SoC [3]. Тестирование осуществляется при помощи среды для симуляции и отладки программ на HDL языках ModelSim [4], входящей в состав пакета Quartus Prime [5].

# Задание 1

## Задание

На языке Verilog описать параметризованный преобразователь N-разрядного двоичного кода в позиционный (Переключатели платы задают двоичный код, светодиоды отображают позиционный).

- Модуль – b2oh; файл b2oh.v; проект в Quartus – b2oh. Рабочая папка – labs\_1

1) Параметр – N – разрядность двоичного кода.

2) Стандарты и номера выводов СБИС для платы задать с помощью атрибутов.

На языке Verilog создать описание тестов:

1) Тест класса 1 (имя теста – tb1\_b2oh.v)

2) Тест класса 2 с вычислением результата (имя теста – tb2\_b2oh.v)

3) Тест класса 2 с чтением файлов (имя теста – tb2\_b2oh.v)

А) Файл с тестовыми воздействиями – input\_b2oh.dat

Б) Файл с ожидаемыми результатами – exp\_b2oh.dat

Осуществить моделирование и отладку модулей и проверку тестов класса 2.

Реализовать проект на плате.

## Описание на языке Verilog

Описание разрабатываемого устройства на языке Verilog приведено ниже на следующем листинге:

|  |
| --- |
| 1. module b2oh #(parameter W = 8)( 2. input clk, 3. input [W-1:0] binary, 4. output reg [2\*\*W-1:0] pos 5. ); 6. always @ (posedge clk) 7. pos <=(1'b01 << binary); 8. endmodule |

**Тест 1 класса**

Программа для теста первого класса приведена в нижеследующем листинге:

|  |
| --- |
| 1. timescale 1 ns / 1ns 2. module tb1\_b2oh(); 3. localparam CLKPERIOD = 20; 4. localparam N = 2; 5. reg[N-1:0] bin; 6. wire [2\*\*N-1:0] pos; 7. reg clk; 8. b2oh #N DUT(clk, bin, pos); 9. initial begin 10. clk = 1'b0; 11. forever #(CLKPERIOD/2) clk = ~clk; 12. end 13. initial begin 14. bin = {N{1'b0}} 15. forever begin 16. @(negedge clk); 17. bin = bin + 1'b01; 18. end 19. end 20. initial #80 $stop; 21. endmodule; |

Приведем на рис. 3 временную диаграмму теста. Результат работы устройства полностью соответствует ожиданиям.

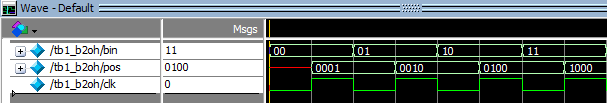


Рисунок 3 – Временная диаграмма теста 1 класса

**Тест 2 класса с вычислением результата**

Код теста второго класса с вычислением результата приведен в следующем листинге

|  |
| --- |
| 1. timescale 1 ns / 1ns 2. module tb2\_b2oh(); 3. localparam CLKPERIOD = 20; 4. localparam N = 2; 5. reg[N-1:0] bin; 6. wire[2\*\*N-1:0] pos; 7. reg clk; 8. b2oh #N DUT (clk, bin, pos); 9. initial begin 10. clk = 1'b0; 11. forever #(CLKPERIOD/2) clk = ~clk; 12. end 13. initial begin 14. bin = {N{1'b0}}; 15. @(posedge clk); #1; 16. if(pos != 4'd01) begin 17. $display("wrong: expected %d, but got %d", 4'd01, pos); 18. $stop; 19. end 20. @(negedge clk); 21. bin = bin + 1'b01; 22. @(posedge clk); #1; 23. if(pos != 4'd05) begin 24. $display("wrong: expected %d, but got %d", 4'd05, pos); 25. $stop; 26. end 27. @(negedge clk); 28. bin = bin + 1'b01; 29. @(posedge clk); #1; 30. if(pos != 4'd04) begin 31. $display("wrong: expected %d, but got %d", 4'd04, pos); 32. $stop; 33. end 34. @(negedge clk); 35. bin = bin + 1'b01; 36. @(posedge clk); #1; 37. if(pos != 4'd08) begin 38. $display("wrong: expected %d, but got %d", 4'd08, pos); 39. $stop; 40. end 41. $display("position=$d", pos); 42. end 43. initial #240 $stop; 44. endmodule; |

Временная диаграмма данного теста отображена на рисунке 4 и совпадает с временной диаграммой из предыдущего пункта, поскольку ошибок во время тестирования не возникло. Попробуем смоделировать ошибку теста, подставив ожидаемое значение 4’d05 вместо 4’d02 на втором фронте тактового сигнала. Результат работы теста с ошибкой приведен на рисунке 5.



Рисунок 4 – Временная диаграмма теста 2 класса c вычислением результата

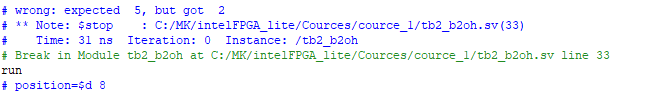


Рисунок 5 – Результаты выполнения теста с ошибкой

**Тест 2 класса с чтением файлов**

Код теста второго класса с чтением файлов тестовых воздействий и ожидаемых результатов в следующем листинге:

|  |
| --- |
| 1. timescale 1 ns / 1ns 2. module tb2f\_b2oh(); 3. localparam CLKPERIOD = 20; 4. localparam N = 2; 5. reg[N-1:0] bin; 6. wire[2\*\*N-1:0] pos; 7. reg clk; 8. integer i; 9. reg [N-1:0] input\_mem [0:3]; 10. reg[2\*\*N-1:0] exp\_mem[0:3]; 11. b2oh #N DUT(clk, bin, pos); 12. initial begin : clk\_init 13. clk = 1'b0; 14. forever #(CLKPERIOD/2) clk = ~clk; 15. end 16. initial begin : testbench\_as\_it\_is 17. bin = {N{1'b0}}; 18. for(i =0; i < 4; i++) begin 19. @(negedge clk); 20. bin = input\_mem[i]; 21. @(posedge clk); #1; 22. if(pos != exp\_mem[i]) begin 23. $display("wrong: expected %d, but got %d", 24. exp\_mem[i], pos); 25. $stop; 26. end 27. #(CLKPERIOD/2); 28. end 29. $display("DECODER TESTED"); 30. end 31. initial begin : set\_data 32. $readmemb("input\_b2oh.dat", input\_mem); 33. $readmemb("input\_b2oh.dat", exp\_mem); 34. end 35. initial #400 $stop; 36. endmodule |

Приведем в следующем листинге тестовые воздействия и ожидаемые результаты:

|  |
| --- |
| Файл input\_b2oh.dat  00  01  10  11  Файл exp\_b2oh.dat  0001  0010  0100  1000 |

Временная диаграмма данного теста отображена на рисунке 6 и совпадает с временной диаграммой из предыдущего пункта, поскольку ошибок во время тестирования не возникло. Попробуем смоделировать ошибку теста, подставив ожидаемое значение 0111 в файле exp\_b2oh.dat вместо 0100 на втором фронте тактового сигнала. Результат работы теста с ошибкой приведен на рисунке 7.

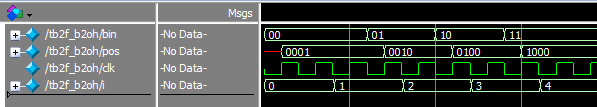
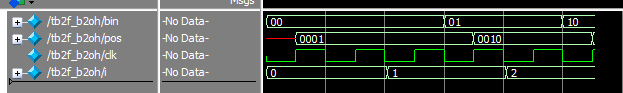


Рисунок 6 – Временная диаграмма теста 2 класса с чтением файлов



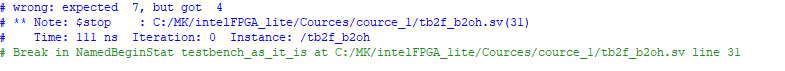


Рисунок 7 – Результаты выполнения теста с ошибкой

## Тестирование на плате 5CSEMA5F31C6

Проведенное на плате 5CSEMA5F31C6 тестирование разработанного устройства показало: результаты тестирования совпадают с ожидаемыми результатами, устройство работает в соответствии с заданием.

# Задание 2

## Задание

На языке Verilog описать на структурном уровне параметризованный N-разрядный мультиплексор 2 в 1.

- Модуль – mux; файл mux.v; проект в Quartus – mux. Рабочая папка – labs\_2

1) Параметр – N – разрядность данных.

2) Стандарты и номера выводов СБИС для платы задать с помощью атрибутов.

На языке Verilog создать описание тестов:

1) Тест класса 1 (имя теста – tb1\_mux.v)

2) Тест класса 2 с вычислением результата (имя теста – tb2\_mux.v)

3) Тест класса 2 с чтением файлов (имя теста – tb2f\_mux.v)

А) Файл с тестовыми воздействиями – input\_mux.dat

Б) Файл с ожидаемыми результатами – exp\_mux.dat

Осуществить моделирование и отладку модулей и проверку тестов класса 2.

Реализовать проект на плате.

## Описание на языке Verilog

Описание разрабатываемого устройства на языке Verilog приведено ниже на следующем листинге:

|  |
| --- |
| 1. `timescale 1ns / 100ps 2. module mux(a\_in, b\_in, pba, d\_out); 3. localparam N = 4; 4. mux2\_1 ul (a\_in[0], b\_in[0], pba, d\_out[0]); 5. mux2\_1 u2 (a\_in[1], b\_in[1], pba, d\_out[1]); 6. mux2\_1 u3 (a\_in[2], b\_in[2], pba, d\_out[2]); 7. mux2\_1 u4 (a\_in[3], b\_in[3], pba, d\_out[3]); 8. endmodule |

**Тест 1 класса**

Код теста первого класса приведен в следующем листинге:

|  |
| --- |
| 1. `timescale 1 ns/ 100 ps 2. module tb1\_mux; 3. localparam N = 4; 4. localparam nums = (1<<N); 5. reg pba; 6. reg [N - 1:0] a\_in, b\_in; 7. wire [N - 1:0] d\_out; 8. integer i, j, k; 9. mux 10. mux\_test(.a\_in(a\_in),.b\_in(b\_in),.pba(pba),.d\_out(d\_out)); 11. initial 12. begin 13. for (i = 0; i < nums; i = i + 1) 14. begin 15. for (j = 0; j < nums; j = j + 1) 16. begin 17. for (k = 0; k < 2; k = k + 1) 18. begin 19. a\_in = i; 20. b\_in = j; 21. pba = k; 22. #10; 23. end 24. end 25. end 26. $stop; 27. end 28. endmodule |

Временная диаграмма данного теста отображена на рисунке 3. Как видно, результат работы устройства полностью соответствует ожиданиям.

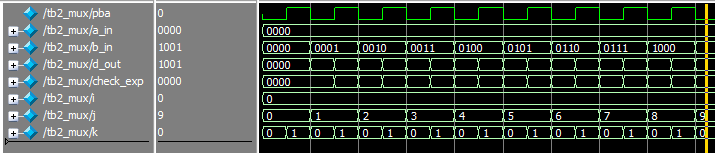


Рисунок 3 – Временная диаграмма теста 1 класса

**Тест 2 класса с вычислением результата**

Код теста второго класса с вычислением результата приведен в следующем листинге:

|  |
| --- |
| 1. `timescale 1 ns/ 100 ps 2. module tb2\_mux; 3. localparam N = 4; 4. reg pba; 5. reg [N - 1:0] a\_in, b\_in; 6. wire [N - 1:0] d\_out; 7. reg [N - 1:0] check\_exp; 8. integer i, j, k; 9. mux 10. mux\_test(.a\_in(a\_in),.b\_in(b\_in),.pba(pba),.d\_out(d\_out)); 11. initial 12. begin 13. for (i = 0; i < (1 << N); i = i + 1) 14. begin 15. for (j = 0; j < (1 << N); j = j + 1) 16. begin 17. for (k = 0; k < 2; k = k + 1) 18. begin 19. a\_in = i; 20. b\_in = j; 21. pba = k; 22. check\_exp = (pba) ? b\_in : a\_in; 23. #10; 24. if(check\_exp != d\_out) err\_task; 25. end 26. end 27. end 28. $display("\n\*\*SIMULATION PASSED SUCCESSFULLY\*\*\n"); 29. $stop; 30. end 31. task err\_task; 32. begin 33. $display("Time:%t ERROR: expected=%0d result=%0d \n", 34. $realtime, check\_exp, d\_out); 35. $stop; 36. end 37. endtask 38. endmodule |

Временная диаграмма данного теста отображена на рисунке 4 и совпадает с временной диаграммой из предыдущего пункта, поскольку ошибок во время тестирования не возникло. Попробуем смоделировать ошибку теста, изменив строку вычисления сигнала. Результат работы теста с ошибкой приведен на рисунке 5.

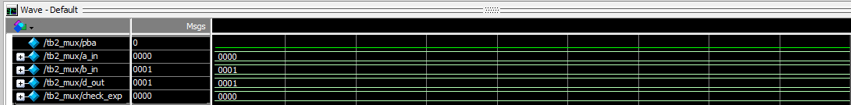


Рисунок 4 – Временная диаграмма теста 2 класса c вычислением результата

**Тест 2 класса с чтением файлов**

Код теста второго класса с чтением файлов тестовых воздействий и ожидаемых результатов приведен в листинге 4. Содержимое файлов приведено в листинге 5

|  |
| --- |
| 1. `timescale 1 ns/ 100 ps 2. module tb2f\_mux; 3. localparam N = 4; 4. reg pba; 5. reg [N - 1:0] a\_in, b\_in; 6. wire [N - 1:0] d\_out; 7. reg [N \* 2:0] input\_mem [9:0]; 8. reg [N - 1:0] exp\_mem [9:0]; 9. integer i; 10. mux mux\_test(.a\_in(a\_in),.b\_in(b\_in),.pba(pba),.d\_out(d\_out)); 11. initial begin 12. $readmemb("input\_mux.dat", input\_mem); 13. $readmemb("exp\_mux.dat", exp\_mem); 14. end 15. initial 16. begin 17. for (i = 0; i < 10; i = i + 1) 18. begin 19. pba = input\_mem[i][0]; 20. a\_in = input\_mem[i][N \* 2:N + 1]; 21. b\_in = input\_mem[i][N:1]; 22. #10; 23. if(d\_out != exp\_mem[i]) err\_task; 24. end 25. $display("\n\*\*SIMULATION PASSED SUCCESSFULLY\*\*\n"); 26. $stop; 27. end 28. task err\_task; 29. begin 30. $display("Time:%t ERROR: expected=%0d result=%0d \n", $realtime, exp\_mem[i], d\_out); 31. $stop; 32. end 33. endtask 34. endmodule |

Приведем в следующем листинге тестовые воздействия и ожидаемые результаты:

|  |
| --- |
| Файл input\_b2oh.dat  000000100  001101011  010001100  010101111  011010000  011110011  100010100  100110111  111000000  111100011  Файл exp\_b2oh.dat  0000  0101  0100  0111  0110  1001  1000  1011  1110  0001 |

Временная диаграмма данного теста отображена на рисунке 6 и совпадает с временной диаграммой из предыдущего пункта, поскольку ошибок во время тестирования не возникло. Попробуем смоделировать ошибку теста, изменив содержимое файла exp\_mux.dat. Результат работы теста с ошибкой приведен на рисунке 7.

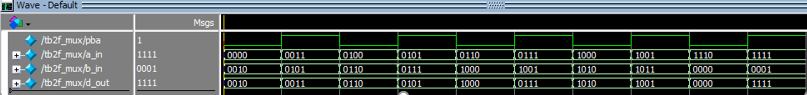


Рисунок 6 – Временная диаграмма теста 2 класса с чтением файлов

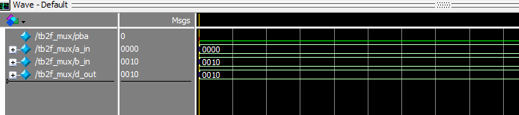


Рисунок 7 – Результаты выполнения теста с ошибкой

## Тестирование на плате 5CSEMA5F31C6

Проведенное на плате 5CSEMA5F31C6 тестирование разработанного устройства показало: результаты тестирования совпадают с ожидаемыми результатами, устройство работает в соответствии с заданием.

# Задание 3

## Задание

На языке Verilog описать на структурном уровне параметризованный N-разрядный де-мультиплексор 1 в 2.

- Модуль – dmux; файл dmux.v; проект в Quartus – dmux. Рабочая папка – labs\_3

1) Параметр – N – разрядность данных.

2) Стандарты и номера выводов СБИС для платы задать с помощью атрибутов.

На языке Verilog создать описание тестов:

1) Тест класса 1 (имя теста – tb1\_dmux.v)

2) Тест класса 2 с вычислением результата (имя теста – tb2\_dmux.v)

3) Тест класса 2 с чтением файлов (имя теста – tb2f\_dmux.v)

А) Файл с тестовыми воздействиями – input\_dmux.dat

Б) Файл с ожидаемыми результатами – exp\_dmux.dat

Осуществить моделирование и отладку модулей и проверку тестов класса 2.

Реализовать проект на плате.

## Описание на языке Verilog

Описание разрабатываемого устройства на языке Verilog приведено ниже на следующем листинге:

|  |
| --- |
| 1. `timescale 1 ns/ 100 ps 2. module dmux(sw, pba, led0, led1); 3. localparam N = 4; 4. (\* altera\_attribute = "-name IO\_STANDARD \"3.3-V LVCMOS\"", 5. chip\_pin = "AE12" \*) 6. input pba; 7. (\* altera\_attribute = "-name IO\_STANDARD \"3.3-V LVCMOS\"", 8. chip\_pin = "AF10, AF9, AC12, AB12" \*) 9. input [N - 1: 0] sw; 10. (\* altera\_attribute = "-name IO\_STANDARD \"3.3-V 11. LVCMOS\"", chip\_pin = "V18, V17, W16, V16" \*) 12. output [N - 1:0] led0; 13. (\* altera\_attribute = "-name IO\_STANDARD \"3.3-V 14. LVCMOS\"", chip\_pin = "W20, Y19, W19, W17" \*) 15. output [N - 1:0] led1; 16. dmux1\_2 u1 (sw[0], pba, led0[0], led1[0]); 17. dmux1\_2 u2 (sw[1], pba, led0[1], led1[1]); 18. dmux1\_2 u3 (sw[2], pba, led0[2], led1[2]); 19. dmux1\_2 u4 (sw[3], pba, led0[3], led1[3]); 20. endmodule |

**Тест 1 класса**

Код теста первого класса ниже на следующем листинге:

|  |
| --- |
| 1. `timescale 1 ns/ 100 ps 2. module tb1\_dmux; 3. localparam N = 4; 4. localparam chanel = 2; 5. reg pba; 6. reg [N - 1:0] sw; 7. wire [N - 1:0] led0, led1; 8. integer i, j; 9. dmux dmux\_test(.sw(sw), .pba(pba), .led0(led0), 10. .led1(led1)); 11. initial 12. begin 13. for (i = 0; i < chanel; i = i + 1) 14. begin 15. pba = i; 16. for (j = 0; j < (1 << N); j = j + 1) 17. begin 18. sw = j; 19. #5; 20. end 21. end 22. $stop; 23. end 24. endmodule |

Временная диаграмма данного теста отображена на рисунке 3. Как видно, результат работы устройства полностью соответствует ожиданиям.



Рисунок 3 – Временная диаграмма теста 1 класса

**Тест 2 класса с вычислением результата**

Код теста второго класса с вычислением результата приведен ниже на следующем листинге:

|  |
| --- |
| `timescale 1 ns/ 100 ps  module tb2\_dmux;  localparam N = 4;  localparam chanel = 2;  reg pba;  reg [N - 1:0] sw;  wire [N - 1:0] led0, led1;  reg [2\*N - 1:0] led\_exp;  integer i, j;  dmux dmux\_test(.sw(sw), .pba(pba), .led0(led0), .led1(led1));  initial begin  for (i = 0; i < chanel; i = i + 1)  begin  pba = i;  for (j = 0; j < (1 << N); j = j + 1)  begin  sw = j;  led\_exp = (pba) ? {sw, {N{1'b0}}} : {{N{1'b0}}, sw};  #5;  if (led\_exp != {led1, led0}) err\_task;  end  end  $display("\n\*\*SIMULATION PASSED SUCCESSFULLY\*\*\n");  $stop;  end  task err\_task;  begin  $display("Time:%t ERROR: expected=%0d result=%0d \n", $realtime, led\_exp, {led1, led0});  $stop;  end  endtask  endmodule |

Фрагмент временной диаграммы данного теста отображена на рисунке 14 и совпадает с временной диаграммой из предыдущего пункта, за исключением сигнала *led\_exp*, отображающего ожидаемое значение работы устройства. Попробуем смоделировать ошибку теста, изменив строку вычисления сигнала *led\_exp* c {led1, led0} на {led0, led1}. Результат работы теста с ошибкой приведен на рисунке 5.

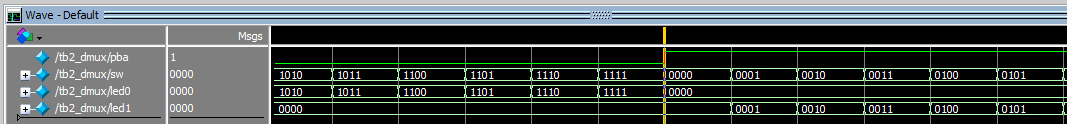


Рисунок 4 – Временная диаграмма теста 2 класса c вычислением результата

**Тест 2 класса с чтением файлов**

Код теста второго класса с чтением файлов тестовых воздействий и ожидаемых результатов приведен в следующем листинге:

|  |
| --- |
| `timescale 1 ns/ 100 ps  module tb2f\_dmux;  localparam N = 4;  reg pba;  reg [N - 1:0] sw;  wire [N - 1:0] led0, led1;  reg [N:0] input\_mem [2\*(1 << N) - 1:0];  reg [2\*N - 1:0] exp\_mem [2\*(1 << N) - 1:0];  integer i;  dmux dmux\_test(.sw(sw), .pba(pba), .led0(led0), .led1(led1));  initial  begin  $readmemb("input\_dmux.dat", input\_mem);  $readmemb("exp\_dmux.dat", exp\_mem);  end  initial  begin  for (i = 0; i < 2\*(1 << N); i = i + 1)  begin  sw = input\_mem[i][N:1];  pba = input\_mem[i][0];  #5;  if (exp\_mem[i] != {led1, led0}) err\_task;  end  $display("\n\*\*SIMULATION PASSED SUCCESSFULLY\*\*\n");  $stop;  end  task err\_task;  begin  $display("Time:%t ERROR: expected=%0d result=%0d \n", $realtime, exp\_mem[i], {led1, led0});  $stop;  end  endtask  endmodule |

Приведем в следующем листинге тестовые воздействия и ожидаемые результаты:

|  |
| --- |
| Файл input\_dmux.dat  00000  00010  00101  00111  01000  01010  01101  01111  10000  10010  10101  10111  11000  11010  11101  11111  Файл exp\_dmux.dat  00000000  00000001  00100000  00110000  00000100  00000101  01100000  01110000  00001000  00001001  10100000  10110000  00001100  00001101  11100000  11110000 |

Временная диаграмма данного теста отображена на рисунке 16 и совпадает с временной диаграммой из предыдущего пункта, поскольку ошибок во время тестирования не возникло. Попробуем смоделировать ошибку теста, изменив содержимое файла exp\_dmux.dat. Результат работы теста с ошибкой приведен на рисунке 7.

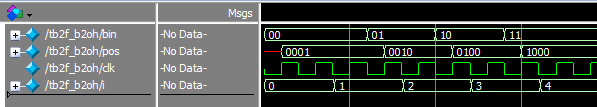


Рисунок 6 – Временная диаграмма теста 2 класса с чтением файлов

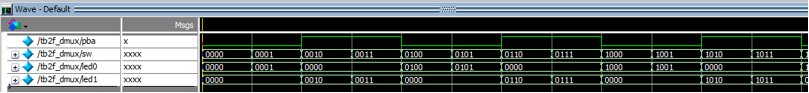


Рисунок 7 – Результаты выполнения теста с ошибкой

## Тестирование на плате 5CSEMA5F31C6

Проведенное на плате 5CSEMA5F31C6 тестирование разработанного устройства показало: результаты тестирования совпадают с ожидаемыми результатами, устройство работает в соответствии с заданием.

# Задание 4

## Задание

На языке Verilog описать на преобразователь 4 разрядного двоичного кода в 2-10 (Переключатели платы зажают двоичный код данных; светодиоды отображают 2-10).

- Модуль – b2bd; файл b2bd.v; проект в Quartus – b2bd. Рабочая папка – labs\_4

1) Параметр – N – разрядность данных.

2) Стандарты и номера выводов СБИС для платы задать с помощью атрибутов.

На языке Verilog создать описание тестов:

1) Тест класса 1 (имя теста – tb1\_b2bd.v)

2) Тест класса 2 с вычислением результата (имя теста – tb2\_b2bd.v)

3) Тест класса 2 с чтением файлов (имя теста – tb2f\_b2bd.v)

А) Файл с тестовыми воздействиями – input\_b2bd.dat

Б) Файл с ожидаемыми результатами – exp\_b2bd.dat

Осуществить моделирование и отладку модулей и проверку тестов класса 2.

Реализовать проект на плате.

## Описание на языке Verilog

Описание разрабатываемого устройства на языке Verilog приведено ниже на следующем листинге:

|  |
| --- |
| `timescale 1 ns/ 100 ps  module b2bd(sw, led);  localparam N = 4;  (\* altera\_attribute = "-name IO\_STANDARD \"3.3-V LVCMOS\"", chip\_pin = "AF10, AF9, AC12, AB12" \*)  input [N-1:0] sw;  (\* altera\_attribute = "-name IO\_STANDARD \"3.3-V LVCMOS\"", chip\_pin = "W20, Y19, W19, W17, V18, V17, W16, V16" \*)  output reg [2\*N-1:0] led;  always @(sw)  if (sw > {4'b1001})  led = sw + (N\*N - 10);  else  led = sw;  endmodule |

**Тест 1 класса**

Код теста первого класса приведен на следующем листинге:

|  |
| --- |
| `timescale 1 ns/ 100 ps  module tb1\_b2bd;  localparam N = 4;  reg [N - 1:0] sw;  wire [2\*N - 1:0] led;  integer i;  b2bd b2bd\_test(.sw(sw), .led(led));  initial  begin  for (i = 0; i < N\*N; i = i + 1)  begin  sw = i;  #5;  end    $stop;  end  endmodule |

Временная диаграмма данного теста отображена на рисунке 1. Как видно, результат работы устройства полностью соответствует ожиданиям.

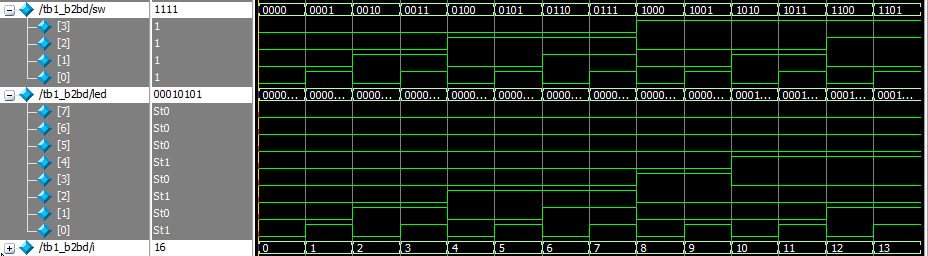


Рисунок 1 – Временная диаграмма теста 1 класса

**Тест 2 класса с вычислением результата**

Код теста второго класса с вычислением результата приведен на следующем листинге:

|  |
| --- |
| `timescale 1 ns/ 100 ps  module tb2\_b2bd;  localparam N = 4;  reg [N - 1:0] sw;  wire [2\*N - 1:0] led;  reg [2\*N - 1:0] led\_exp;  integer i, tmp;  b2bd b2bd\_test(.sw(sw), .led(led));  initial  begin  for (i = 0; i < N\*N; i = i + 1)  begin  sw = i;  tmp = i - 10;  convert;  #5;  if (led\_exp != led) err\_task;  end    $display("\n\*\*SIMULATION PASSED SUCCESSFULLY\*\*\n");  $stop;  end  task convert;  begin  if( i < 10)  led\_exp = {4'b0000,4'b1111 & i};  if( i > 9)  led\_exp = tmp + 12;  end  endtask  task err\_task;  begin  $display("Time:%t ERROR: expected=%0d result=%0d \n", $realtime, led\_exp, led);  $stop;  end  endtask  endmodule |

Временная диаграмма данного теста отображена на рисунке 4.6 и совпадает с временной диаграммой из предыдущего пункта, поскольку ошибок во время тестирования не возникло. Попробуем смоделировать ошибку теста, подставив ожидаемое значение 4’d12 вместо 4’d16 на втором фронте тактового сигнала. Результат работы теста с ошибкой приведен на рисунке 3.

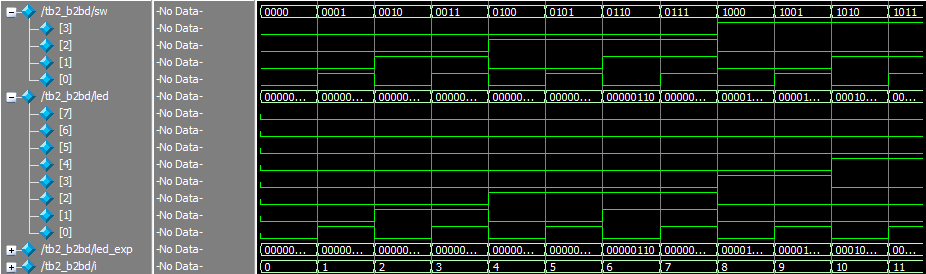


Рисунок 2 – Временная диаграмма теста 2 класса



Рисунок 3 – результат выполнения теста с ошибкой

**Тест 2 класса с чтением файлов**

Код теста второго класса с чтением файлов тестовых воздействий и ожидаемых результатов приведен на рисунке 4. Содержимое файлов приведено ниже на листингах:

|  |
| --- |
| ‘timesclale 1 ns / 1ns  module tb2f\_b2oh();  localparam CLKPERIOD = 20;  localparam N = 2;  reg[N-1:0] bin;  wire[2\*\*N-1:0] pos;  reg clk;  integer i;  reg[N-1:0] input\_mem [0:3];  reg[2N\*\*-1:0] exp\_mem[0:3];  b2oh #N DUT(clk, bin, pos);  initial begin : clk\_init  clk = 1’b0;  forever #(CLKPERIOD/2) clk = ~ clk;  end  initial begin : testbench\_as\_it\_is  bin = {N{1’b0}};  for(i = 0; i < 4; i ++) begin  @(negedge clk);  Bin = input\_mem[i];  @(posedge clk); #1;  if(pos != exp\_mem[i]) begin  $display(“wrong: expected %d, but got %d”, exp\_mem[i], pos);  $stop;  end  #(CLKPERIOD/2)  end  $display(“DECODER TESTED”);  end  initial begin : set\_data  $readmemb(“input\_b2oh.dat”, input\_mem);  $readmemb(“exp\_b2oh.dat”, exp\_mem);  end  initial #400 $stop;  endmodule |

Приведем в следующем листинге тестовые воздействия и ожидаемые результаты:

|  |
| --- |
| Файл input\_b2oh.dat  0000  0001  0010  0011  0100  0101  0110  0111  1000  1001  1010  1011  1100  1101  1110  1111  Файл exp\_b2oh.dat  00000000  00000001  00000010  00000011  00000100  00000101  00000110  00000111  00001000  00001001  00010000  00010001  00010010  00010011  00010100  00010101 |

Временная диаграмма данного теста отображена на рисунке 4 и совпадает с временной диаграммой из предыдущего пункта, поскольку ошибок во время тестирования не возникло. Попробуем смоделировать ошибку теста, подставив ожидаемое значение 00000011 в файле exp\_b2oh.dat вместо 00000001 на первом фронте тактового сигнала. Результат работы теста с ошибкой приведен на рисунках 5 и 6.

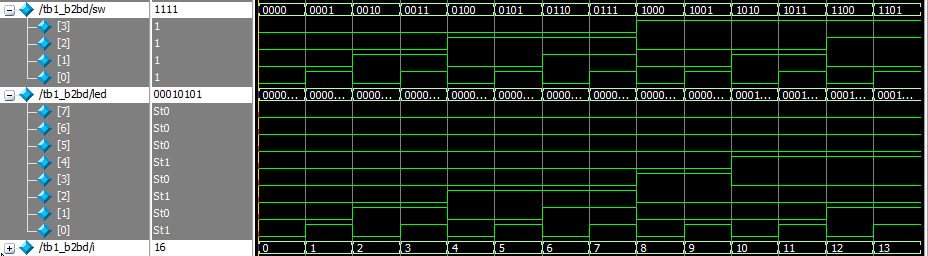


Рисунок 4 – Временная диаграмма теста 2 класса с чтением файла

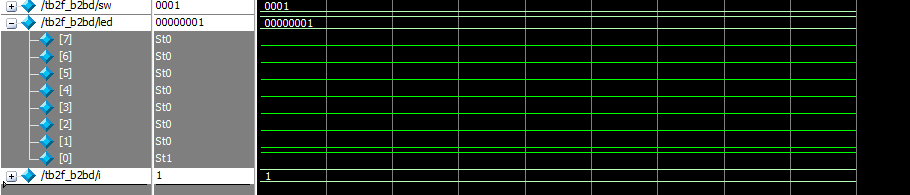


Рисунок 5 – Результаты выполнения теста с ошибкой

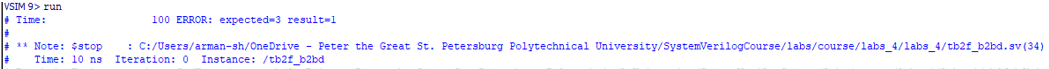


Рисунок 6 – Результаты выполнения теста с ошибкой

## Тестирование на плате 5CSEMA5F31C6

Проведенное на плате 5CSEMA5F31C6 тестирование разработанного устройства показало: результаты тестирования совпадают с ожидаемыми результатами, устройство работает в соответствии с заданием.

# Задание 5

## Задание

На языке Verilog описать на преобразователь 4 разрядного двоичного кода в 7 - сегементный (Переключатели платы зажают двоичный код данных; 7-сегментный индикатор отображает значение 4-разрядного двочного кода в hex).

- Модуль – b27s; файл b27s.v; проект в Quartus – b27s. Рабочая папка – labs\_5

1) Параметр – N – разрядность данных.

2) Стандарты и номера выводов СБИС для платы задать с помощью атрибутов.

На языке Verilog создать описание тестов:

1) Тест класса 1 (имя теста – tb1\_b27s.v)

2) Тест класса 2 с вычислением результата (имя теста – tb2\_b27s.v)

3) Тест класса 2 с чтением файлов (имя теста – tb2f\_b27s.v)

А) Файл с тестовыми воздействиями – input\_b27s.dat

Б) Файл с ожидаемыми результатами – exp\_b27s.dat

Осуществить моделирование и отладку модулей и проверку тестов класса 2.

Реализовать проект на плате.

## Описание на языке Verilog

Описание разрабатываемого устройства на языке Verilog приведено ниже на следующем листинге:

|  |
| --- |
| `timescale 1 ns/ 100 ps  module b27s(sw, ss\_a);  localparam N = 4;  (\* altera\_attribute = "-name IO\_STANDARD \"3.3-V LVCMOS\"", chip\_pin = "AF10, AF9, AC12, AB12" \*)  input [N - 1:0] sw;  (\* altera\_attribute = "-name IO\_STANDARD \"3.3-V LVCMOS\"", chip\_pin = "AH28, AG28, AF28, AG27, AE28, AE27, AE26" \*)  output reg [2\*N - 2:0] ss\_a;  reg [6:0] ss\_arr [15:0];  initial  begin  ss\_arr[0] = 7'h40; // 0  ss\_arr[1] = 7'h79; // 1  ss\_arr[2] = 7'h24; // 2  ss\_arr[3] = 7'h30; // 3  ss\_arr[4] = 7'h19; // 4  ss\_arr[5] = 7'h12; // 5  ss\_arr[6] = 7'h02; // 6  ss\_arr[7] = 7'h78; // 7  ss\_arr[8] = 7'h00; // 8  ss\_arr[9] = 7'h10; // 9  ss\_arr[10] = 7'h08; // 10  ss\_arr[11] = 7'h03; // A  ss\_arr[12] = 7'h46; // B  ss\_arr[13] = 7'h21; // C  ss\_arr[14] = 7'h06; // D  ss\_arr[15] = 7'h0e; // F  end    always @\*  begin  ss\_a = ss\_arr[sw[N - 1:0]];  end  endmodule |

**Тест 1 класса**

Код теста первого класса приведен ниже на следующем листинге:

|  |
| --- |
| `timescale 1 ns/ 100 ps  module tb1\_b27s;  localparam N = 4;  reg [N - 1:0] sw;  wire [2\*N - 2:0] ss\_a;  integer i;  b27s b27s\_test(.sw(sw),.ss\_a(ss\_a));  initial  begin  for (i = 0; i < N\*N; i = i + 1)  begin  sw = i;  #5;  end    $stop;  end  endmodule |

Временная диаграмма данного теста отображена на рисунке 1. Как видно, результат работы устройства полностью соответствует ожиданиям.

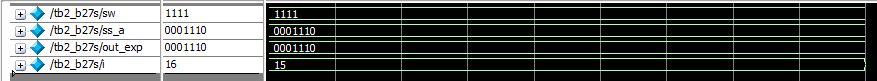


Рисунок 1 – Временная диаграмма теста 1 класса

**Тест 2 класса с вычислением результата**

Код теста второго класса с вычислением результата приведен на следующем листинге:

|  |
| --- |
| `timescale 1 ns/ 100 ps  module tb2\_b27s;  localparam N = 4;  reg [N - 1:0] sw;  wire [2\*N - 2:0] ss\_a;  reg [2\*N - 2:0] out\_exp;  integer i;  b27s b27s\_test(.sw(sw),.ss\_a(ss\_a));  initial  begin  for (i = 0; i < N\*N; i = i + 1)  begin  sw = i;  convert;  #5;  if (out\_exp != ss\_a) err\_task;  end    $display("\n\*\*SIMULATION PASSED SUCCESSFULLY\*\*\n");  $stop;  end  task convert;  begin  case (i)  4'd0: out\_exp = 7'h40; // 0  4'd1: out\_exp = 7'h79; // 1  4'd2: out\_exp = 7'h24; // 2  4'd3: out\_exp = 7'h30; // 3  4'd4: out\_exp = 7'h19; // 4  4'd5: out\_exp = 7'h12; // 5  4'd6: out\_exp = 7'h02; // 6  4'd7: out\_exp = 7'h78; // 7  4'd8: out\_exp = 7'h00; // 8  4'd9: out\_exp = 7'h10; // 9  4'd10: out\_exp = 7'h08; // 10  4'd11: out\_exp = 7'h03; // A  4'd12: out\_exp = 7'h46; // B  4'd13: out\_exp = 7'h21; // C  4'd14: out\_exp = 7'h06; // D  4'd15: out\_exp = 7'h0e; // F  endcase  end  endtask  task err\_task;  begin  $display("Time:%t ERROR: expected=%0b result=%0b \n", $realtime, out\_exp, ss\_a);  $stop;  end  endtask  endmodule |

Временная диаграмма данного теста отображена на рисунке 2 и совпадает с временной диаграммой из предыдущего пункта, поскольку ошибок во время тестирования не возникло. Попробуем смоделировать ошибку теста, подставив ожидаемое значение 1110 вместо 1111001 на втором фронте тактового сигнала. Результат работы теста с ошибкой приведен на рисунке 3.

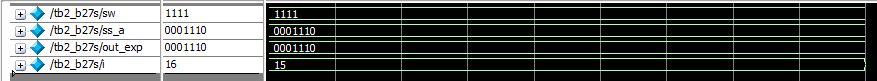


Рисунок 2 – Временная диаграмма теста 2 класса



Рисунок 3 – Результаты выполнения теста с ошибкой

**Тест 2 класса с чтением файлов**

Код теста второго класса с чтением файлов тестовых воздействий и ожидаемых результатов приведен ниже на следующем листинге:

|  |
| --- |
| `timescale 1 ns/ 100 ps  module tb2f\_b27s;  localparam N = 4;  reg [N - 1:0] sw;  wire [2\*N - 2:0] ss\_a;  reg [N - 1:0] input\_mem [0:N\*N - 1];  reg [2\*N - 2:0] exp\_mem [0:N\*N - 1];  integer i;  b27s b27s\_test(.sw(sw),.ss\_a(ss\_a));  initial  begin  $readmemb("input\_b27s.dat", input\_mem);  $readmemb("exp\_b27s.dat", exp\_mem);  end  initial  begin  for (i = 0; i < N\*N; i = i + 1)  begin  sw = input\_mem[i];  #5;  if (exp\_mem[i] != ss\_a) err\_task;  end    $display("\n\*\*SIMULATION PASSED SUCCESSFULLY\*\*\n");  $stop;  end  task err\_task;  begin  $display("Time:%t ERROR: expected=%0b result=%0b \n", $realtime, exp\_mem[i], ss\_a);  $stop;  end  endtask  endmodule |

Приведем в следующем листинге тестовые воздействия и ожидаемые результаты:

|  |
| --- |
| Файл input\_b2oh.dat  0000  0001  0010  0011  0100  0101  0110  0111  1000  1001  1010  1011  1100  1101  1110  1111  Файл exp\_b2oh.dat  1000000  1111001  0100100  0110000  0011001  0010010  0000010  1111000  0000000  0010000  0001000  0000011  1000110  0100001  0000110  0001110 |

Временная диаграмма данного теста отображена на рисунке 4 и совпадает с временной диаграммой из предыдущего пункта, поскольку ошибок во время тестирования не возникло. Попробуем смоделировать ошибку теста, подставив ожидаемое значение 1000001 в файле exp\_b2oh.dat вместо 10000000 на первом фронте тактового сигнала. Результат работы теста с ошибкой приведен на рисунках 5 и 6

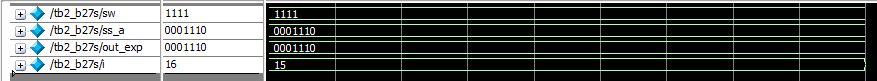


Рисунок 4 – Временная диаграмма

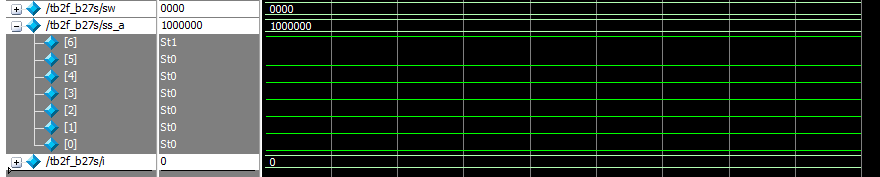


Рисунок 5 – Результаты выполнения теста с ошибкой

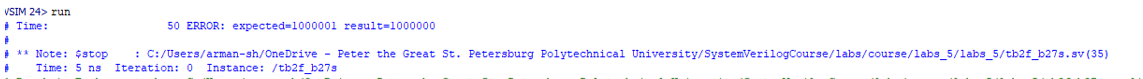


Рисунок 6 – Результаты выполнения теста с ошибкой

## Тестирование на плате 5CSEMA5F31C6

Проведенное на плате 5CSEMA5F31C6 тестирование разработанного устройства показало: результаты тестирования совпадают с ожидаемыми результатами, устройство работает в соответствии с заданием.

# СПИСОК ИСТОЧНИКОВ

1. ISO/IEC TR 19759:2015 Software Engineering — Guide to the software engineering body of knowledge (SWEBOK)
2. FPGA Design Software - Intel® Quartus® Prime : [Электронный ресурс] // Intel | Data Center Solutions, IoT, and PC Innovation. URL: https://www.intel.com/content/www/us/en/programmable/products/fpga/cyclone-series/cyclone-v/support.html . (Дата обращения: 15.06.2019 )
3. Terasic - SoC Platform - Cyclone - DE1-SoC Board: [Электронный ресурс] // :: Terasic Inc. - Expertise in FPGA/ASIC Design :: URL: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=836&PartNo=4 (Дата обращения: 15.06.2019 )
4. Intel® FPGA Simulation - ModelSim\*-Intel® FPGA: [Электронный ресурс] // Intel | Data Center Solutions, IoT, and PC Innovation. URL: https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/model-sim.html (Дата обращения: 15.06.2019 )
5. FPGA Design Software - Intel® Quartus® Prime: [Электронный ресурс] // Intel | Data Center Solutions, IoT, and PC Innovation. URL: https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/overview.html (Дата обращения: 15.06.2019 )