|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

Институт Информационных Технологий

Кафедра Вычислительной Техники (ВТ)

**ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №** **4**

«Основы языка команд Tcl»

по дисциплине

«Схемотехника устройств компьютерных систем»

|  |  |
| --- | --- |
| Выполнил студент группы  ИВБО-12-23 | Львов А. В. |
| Принял преподаватель кафедры ВТ | Дуксин Н. А. |
| Практическая работа выполнена | « \_\_ » \_\_\_\_\_\_\_ 2025 г. |
| «Зачтено» | « \_\_ » \_\_\_\_\_\_\_ 2025 г. |

Москва 2025

АННОТАЦИЯ

Данная работа включает в себя 7 рисунков, 13 листингов. Количество страниц в работе — 37.

СОДЕРЖАНИЕ

[1 ПОСТАНОВКА ЗАДАЧИ 4](#_Toc163689954)

[2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ 5](#_Toc163689955)

[2.1 Исходный код файлов проекта 5](#_Toc163689956)

[2.2 Файл с описанной программой на языке Tcl и результаты работы 26](#_Toc163689960)

[ЗАКЛЮЧЕНИЕ 36](#_Toc163689962)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 37](#_Toc163689963)

1. ПОСТАНОВКА ЗАДАЧИ

Сформировать набор файлов для создания проекта: Файлы на языке Verilog, содержащие модули для описания устройства, заданного вариантом, а также модули для верификации RTL-модели, файлы проектных ограничений для размещения проекта на ПЛИС, в набор файлов должны входить: конфигурация для работы устройства на частоте 100 МГц и для работы устройства на частоте 200 МГц.[1] Сформировать файл с описанной программой на языке TCL, в которой: создать новый проект для чипа «xc7a100tcsg324-1». создать в рамках проекта наборы файлов «designs» для каждого отдельно оформленного теста, создать два набора файлов «constrs» проектных ограничений, добавить в проект в набор файлов «source set» с именем «sources\_1» файлы, содержащие модули для описания устройства, заданного вариантом, добавить в проект в каждый набор файлов «simulation set» необходимые файлы для тестирования в рамках набора, добавить в проект в каждый набор файлов «constraints set» требуемые файлы проектных ограничений, запуск процесса симуляции последовательно для каждого из набора «simulation set», запустить синтез и имплементацию последовательно для каждого набора «constraints set», для каждого варианта имплементации получить отчёт о временных задержках («Timing Summary Report»), сохранить данные отчёта в соответствующие файлы. Запустить на исполнение разработанный скрипт [2-3]. Составить отчёт.

# ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

## 2.1 Исходный код файлов проекта

Исходный код проекта был взят из третьей практической работы, где и проведено его описание. Результат представлен на Листингах 2.1 – 2.10.

*Листинг 2.1 – Модуль basis*

module clk\_div

#(

DIV\_COUNT = 10000

)

(

input clk,

output reg clk\_div

);

reg [$clog2(DIV\_COUNT)-1:0] clk\_counter; // счётчик тактов для делителя

initial

begin

clk\_counter = {DIV\_COUNT{1'b0}};

clk\_div = 0;

end

always@ (posedge clk)

begin

if (clk\_counter == 0)

clk\_div <= 1;

else

clk\_div <= 0;

end

always@ (posedge clk)

begin

if (clk\_counter == (DIV\_COUNT-1))

clk\_counter <= 0;

else

clk\_counter <= clk\_counter + 1;

end

*Листинг 2.2 – Модуль fsm\_div*

`timescale 1ns / 1ps

module fsm\_div

(

input reset,

input clk,

input valid\_in,

input [3:0] d\_in,

output reg [3:0] d\_out,

output reg valid\_out,

output reg error\_out

);

// Константы ошибок

localparam NO\_ERROR = 0, DIV\_BY\_ZERO = 1;

// Регистры операндов

reg signed [3:0] a\_reg, b\_reg;

// Состояния конечного автомата

localparam S0 = 0, S1 = 1, S2 = 2, S3 = 3;

reg [1:0] state;

initial state = S0;

always@(posedge clk)

begin

if (reset)

state <= S0;

else

case(state)

// Сброс регистров

S0: begin

a\_reg <= 0;

b\_reg <= 0;

d\_out <= 0;

error\_out <= 0;

valid\_out <= 0;

state <= 1;

end

// Ввод первого операнда (делимого)

S1: if (valid\_in)

begin

a\_reg <= d\_in;

state <= S2;

end

// Ввод второго операнда (делителя)

S2: if (valid\_in)

begin

b\_reg <= d\_in;

state <= S3;

end

// Выполнение операции деления

S3: begin

if (b\_reg == 0)

begin

error\_out <= DIV\_BY\_ZERO;

valid\_out <= 1;

end

*Продолжение Листинга 2.2*

else if (a\_reg == 0)

begin

d\_out <= 0;

valid\_out <= 1;

end

else

begin

d\_out <= a\_reg / b\_reg;

valid\_out <= 1;

end

state <= S0;

end

endcase

end

endmodule

*Листинг 2.3 – Модуль FILTER*

module FILTER #(size = 3) (

input CLK, CLOCK\_ENABLE, IN\_SIGNAL,

output reg OUT\_SIGNAL, OUT\_SIGNAL\_ENABLE

);

reg [1:0] IN\_SIGNAL\_SYNC;

reg [size-1:0] counter;

initial

begin

IN\_SIGNAL\_SYNC = 0; counter = 0;

OUT\_SIGNAL = 0; OUT\_SIGNAL\_ENABLE = 0;

end

always @(posedge CLK)

begin

IN\_SIGNAL\_SYNC <= {IN\_SIGNAL\_SYNC[0], IN\_SIGNAL};

counter <= (IN\_SIGNAL\_SYNC[1] ~^ OUT\_SIGNAL) ?

{size{1'd0}} : (CLOCK\_ENABLE ? counter + 1 : counter);

if (&(counter) & CLOCK\_ENABLE)

OUT\_SIGNAL <= IN\_SIGNAL\_SYNC[1];

OUT\_SIGNAL\_ENABLE <= &(counter) & CLOCK\_ENABLE & IN\_SIGNAL\_SYNC[1];

end

endmodule

*Листинг 2.4 – Модуль top\_fsm\_div*

module top\_fsm\_div (

input clk,

input btn\_c\_in,

input btn\_reset\_in,

input [3:0] SW,

output [7:0] AN,

output [6:0] CATH,

output valid\_out\_LED

);

wire btn\_c\_out;

FILTER #(4) btn\_c\_filter(

.CLK(clk),

.CLOCK\_ENABLE(1),

.IN\_SIGNAL(btn\_c\_in),

.OUT\_SIGNAL\_ENABLE(btn\_c\_out)

);

wire btn\_reset\_out;

FILTER #(4) btn\_reset\_filter(

.CLK(clk),

.CLOCK\_ENABLE(1),

.IN\_SIGNAL(~btn\_reset\_in),

.OUT\_SIGNAL\_ENABLE(btn\_reset\_out)

);

wire fsm\_valid\_out, fsm\_error\_out;

wire [3:0] fsm\_d\_out;

reg [3:0] fsm\_d\_out\_reg;

reg fsm\_valid\_out\_reg, fsm\_error\_out\_reg;

assign valid\_out\_LED = fsm\_valid\_out\_reg;

initial

begin

fsm\_d\_out\_reg <= 0;

fsm\_valid\_out\_reg <= 0;

fsm\_error\_out\_reg <= 0;

end

always@(posedge clk)

begin

if (fsm\_valid\_out)

begin

fsm\_d\_out\_reg <= fsm\_d\_out;

fsm\_valid\_out\_reg <= fsm\_valid\_out;

fsm\_error\_out\_reg <= fsm\_error\_out;

end

else if (btn\_c\_out && fsm\_valid\_out\_reg)

begin

fsm\_d\_out\_reg <= 0;

fsm\_valid\_out\_reg <= 0;

fsm\_error\_out\_reg <= 0;

end

end

wire fsm\_valid\_in = btn\_c\_out && !fsm\_valid\_out\_reg;

fsm\_div fsm(

.clk(clk),

.valid\_in(fsm\_valid\_in),

*Продолжение Листинга 2.4*

.reset(btn\_reset\_out),

.valid\_out(fsm\_valid\_out),

.d\_in(SW),

.d\_out(fsm\_d\_out),

.error\_out(fsm\_error\_out)

);

wire clk\_div\_out;

clk\_div clk\_div1 (

.clk(clk),

.clk\_div(clk\_div\_out)

);

seven\_seg seg(

.clk(clk),

.CE(clk\_div\_out),

.RESET(btn\_reset\_out),

.NUMBER({fsm\_d\_out\_reg, 8'd0, 3'd0, fsm\_error\_out\_reg, 12'd0, SW}),

.AN\_MASK(8'b01101110),

.AN(AN),

.CATH(CATH)

);

endmodule

*Листинг 2.5 – Модуль* seven\_seg

module seven\_seg (

input clk,

input CE,

input RESET,

input [31:0] NUMBER,

input [7:0] AN\_MASK,

output [7:0] AN,

output reg [6:0] CATH

);

reg [7:0] AN\_REG;

initial AN\_REG = 0;

assign AN = AN\_REG | AN\_MASK;

reg [2:0] digit\_counter;

initial digit\_counter = 0;

wire [3:0] NUMBER\_SPLITTER [0:7];

genvar i;

generate

for (i = 0; i < 8; i = i + 1)

assign NUMBER\_SPLITTER[i] = NUMBER[((i+1)\*4-1)-:4];

endgenerate

always @(posedge clk)

if (RESET == 1)

digit\_counter <= 0;

else if (CE == 1)

digit\_counter <= digit\_counter + 3'b1;

wire [3:0] number = NUMBER\_SPLITTER[digit\_counter];

always @\*

begin

case (number)

*Продолжение Листинга 2.5*

4'h0: CATH <= 7'b1000000;

4'h1: CATH <= 7'b1111001;

4'h2: CATH <= 7'b0100100;

4'h3: CATH <= 7'b0110000;

4'h4: CATH <= 7'b0011001;

4'h5: CATH <= 7'b0010010;

4'h6: CATH <= 7'b0000010;

4'h7: CATH <= 7'b1111000;

4'h8: CATH <= 7'b0000000;

4'h9: CATH <= 7'b0010000;

4'ha: CATH <= 7'b0001000;

4'hb: CATH <= 7'b0000011;

4'hc: CATH <= 7'b1000110;

4'hd: CATH <= 7'b0100001;

4'he: CATH <= 7'b0000110;

4'hf: CATH <= 7'b0001110;

default: CATH <= 7'b1111111;

endcase

case (digit\_counter)

3'd0: AN\_REG <= 8'b11111110;

3'd1: AN\_REG <= 8'b11111101;

3'd2: AN\_REG <= 8'b11111011;

3'd3: AN\_REG <= 8'b11110111;

3'd4: AN\_REG <= 8'b11101111;

3'd5: AN\_REG <= 8'b11011111;

3'd6: AN\_REG <= 8'b10111111;

3'd7: AN\_REG <= 8'b01111111;

default: AN\_REG <= 8'b11111111;

endcase

end

endmodule

*Листинг 2.6 – Тестовый модуль test\_top\_fsm\_div*

`timescale 1ns / 1ps

module test\_top\_fsm\_div;

localparam CLK\_PERIOD = 10;

reg clk;

initial clk = 0;

always #(CLK\_PERIOD/2) clk <= ~clk;

localparam CLK\_DIV\_PERIOD = 100\_000;

reg clk\_div;

initial

begin

clk\_div = 0;

@(posedge clk);

forever begin

clk\_div <= 1;

#(CLK\_PERIOD);

clk\_div <= 0;

#(CLK\_DIV\_PERIOD - CLK\_PERIOD);

end

end

reg signed [3:0] SW;

initial SW = 0;

reg btn\_c\_in;

*Продолжение Листинга 2.6*

` initial btn\_c\_in = 0;

reg btn\_reset\_in;

initial btn\_reset\_in = 1;

wire [7:0] AN;

wire [6:0] CATH;

wire valid\_out\_LED;

top\_fsm\_div uut (

.clk(clk),

.btn\_c\_in(btn\_c\_in),

.btn\_reset\_in(btn\_reset\_in),

.SW(SW),

.AN(AN),

.CATH(CATH),

.valid\_out\_LED(valid\_out\_LED)

);

localparam TEST\_COUNT = 2;

integer i;

reg [7:0] test\_register [0:TEST\_COUNT-1];

initial

begin

for (i = 0; i < TEST\_COUNT; i = i + 1)

test\_register[i] = 8'b0;

@(posedge clk);

test\_top\_1(test\_register[0]);

test\_top\_2(test\_register[1]);

test\_stats();

end

task test\_stats;

integer i, j;

reg [1:0] test\_counter;

begin

test\_counter = 0;

$display("\n[%0t]: Результаты тестирования:", $time);

for (i = 0; i < TEST\_COUNT; i = i + 1)

begin

if (&(test\_register[i]))

begin

$display("Сценарий %0d пройден успешно.", i+1);

test\_counter = test\_counter + 1;

end

else begin

$display("Сценарий %0d НЕ пройден.", i+1);

for (j = 0; j < 8; j = j + 1)

if (!test\_register[i][j])

$display("Ошибка на шаге %0d", j + 1);

end

end

$display("Пройдено сценариев: %0d/%0d", test\_counter, TEST\_COUNT);

end

endtask

function [6:0] get\_cath\_mask;

input [3:0] number;

begin

case (number)

4'h0: get\_cath\_mask = 7'b1000000;

*Продолжение Листинга 2.6*

4'h1: get\_cath\_mask = 7'b1111001;

4'h2: get\_cath\_mask = 7'b0100100;

4'h3: get\_cath\_mask = 7'b0110000;

4'h4: get\_cath\_mask = 7'b0011001;

4'h5: get\_cath\_mask = 7'b0010010;

4'h6: get\_cath\_mask = 7'b0000010;

4'h7: get\_cath\_mask = 7'b1111000;

4'h8: get\_cath\_mask = 7'b0000000;

4'h9: get\_cath\_mask = 7'b0010000;

4'ha: get\_cath\_mask = 7'b0001000;

4'hb: get\_cath\_mask = 7'b0000011;

4'hc: get\_cath\_mask = 7'b1000110;

4'hd: get\_cath\_mask = 7'b0100001;

4'he: get\_cath\_mask = 7'b0000110;

4'hf: get\_cath\_mask = 7'b0001110;

default: get\_cath\_mask = 7'b1111111;

endcase

end

endfunction

function [7:0] get\_an\_mask;

input [2:0] an\_number;

begin

case (an\_number)

3'd0: get\_an\_mask = 8'b11111110;

3'd1: get\_an\_mask = 8'b11111101;

3'd2: get\_an\_mask = 8'b11111011;

3'd3: get\_an\_mask = 8'b11110111;

3'd4: get\_an\_mask = 8'b11101111;

3'd5: get\_an\_mask = 8'b11011111;

3'd6: get\_an\_mask = 8'b10111111;

3'd7: get\_an\_mask = 8'b01111111;

default: get\_an\_mask = 8'b11111111;

endcase

end

endfunction

task test\_top\_1;

output reg [7:0] test\_register;

begin

$display("\nСценарий 1. Деление на ноль (негативный сценарий)");

test\_script(

.a(4'h3), .b(4'h0),

.res\_expected(0),

.error\_expected(1),

.test\_register(test\_register)

);

end

endtask

task test\_top\_2;

output reg [7:0] test\_register;

begin

$display("\nСценарий 2. Деление двух чисел (позитивный сценарий)");

test\_script(

.a(4'h6), .b(4'h3),

.res\_expected(4'h2),

.error\_expected(0),

.test\_register(test\_register)

);

*Продолжение Листинга 2.6*

end

endtask

localparam PRESS = 1, RELEASE = 0;

localparam op\_an\_number = 0, op\_an\_count = 1;

localparam err\_an\_number = 4, err\_an\_count = 1;

localparam res\_an\_number = 7, res\_an\_count = 1;

task test\_script;

input [3:0] a, b, res\_expected;

input error\_expected;

output reg [7:0] test\_register;

reg test\_result;

reg [3:0] res\_real;

reg error\_real;

begin

// Ввод первого числа

@(posedge clk);

SW = a;

btn\_c(PRESS, 32);

btn\_c(RELEASE, 32);

$display("\n1) Проверка ввода первого числа.");

test\_segs(op\_an\_number, op\_an\_count, a, test\_result);

test\_register[0] = test\_result;

// Ввод второго числа

@(posedge clk);

SW = b;

btn\_c(PRESS, 32);

btn\_c(RELEASE, 32);

$display("\n2) Проверка ввода второго числа.");

test\_segs(op\_an\_number, op\_an\_count, b, test\_result);

test\_register[1] = test\_result;

$display("\n3) Проверка наличия сигнала готовности выходных данных на светодиоде.");

if (valid\_out\_LED)

$display("Сигнал готовности выходных данных присутствует на шине valid\_out\_LED");

else

$display("Сигнал готовности выходных данных отсутствует на шине valid\_out\_LED");

test\_register[2] = valid\_out\_LED;

$display("\n4) Проверка вывода результата на индикаторах.");

if (error\_expected) begin

$display("Результат не учитывается при ненулевой ошибке");

test\_register[3] = 1;

end

else begin

test\_segs(res\_an\_number, res\_an\_count, res\_expected, test\_result);

test\_register[3] = test\_result;

end

$display("\n5) Проверка вывода ошибки на индикаторах.");

test\_segs(err\_an\_number, err\_an\_count, error\_expected, test\_result);

test\_register[4] = test\_result;

// Подтверждение обработки вывода

btn\_c(PRESS, 32);

btn\_c(RELEASE, 32);

*Продолжение Листинга 2.6*

$display("\n6) Проверка сброса сигнала готовности выходных данных на светодиоде.");

if (valid\_out\_LED)

$display("Сигнал готовности выходных данных присутствует на шине valid\_out\_LED");

else

$display("Сигнал готовности выходных данных отсутствует на шине valid\_out\_LED");

test\_register[5] = valid\_out\_LED == 0;

$display("\n7) Проверка сброса результата на индикаторах.");

test\_segs(res\_an\_number, res\_an\_count, 0, test\_result);

test\_register[6] = test\_result;

$display("\n8) Проверка сброса ошибки на индикаторах.");

test\_segs(err\_an\_number, err\_an\_count, 0, test\_result);

test\_register[7] = test\_result;

end

endtask

task btn\_c;

input signal\_in;

input [6:0] ticks;

begin

@(posedge clk);

btn\_c\_in <= signal\_in;

$display("\n[%0t]: Сигнал %b подан на линию btn\_c\_in.", $time, signal\_in);

repeat(ticks + 2)

@(posedge clk);

btn\_c\_in <= 0;

$display("[%0t]: Сигнал %b убран с линии btn\_c\_in, подан сигнал 0", $time, signal\_in);

end

endtask

task btn\_reset;

input signal\_in;

input [6:0] ticks;

begin

@(posedge clk);

btn\_reset\_in <= signal\_in;

$display("\n[%0t]: Сигнал %b подан на линию btn\_reset\_in.", $time, signal\_in);

repeat(ticks + 2)

@(posedge clk);

btn\_reset\_in <= 0;

$display("[%0t]: Сигнал %b убран с линии btn\_reset\_in, подан сигнал 0", $time, signal\_in);

end

endtask

task test\_segs;

input [2:0] an\_number;

input [3:0] an\_count;

input [31:0] value;

*Продолжение Листинга 2.6*

output reg test\_result;

reg [3:0] i;

begin

test\_result = 1;

wait(AN == get\_an\_mask(an\_number));

$display("\n[%0t] Номер младшего индикатора: %d", $time, an\_number);

$display("Количество индикаторов: %d", an\_count);

$display("Значение: %h", value);

for (i = 0; i < an\_count; i = i + 1)

begin

@(posedge clk);

$display("[%0t] Номер индикатора: %0d", $time, an\_number + i);

$display("Ожидаемые сигналы на линии катодов (CATH): %b", get\_cath\_mask( value[(i+1)\*4-1 -: 4]));

$display("Фактические сигналы на линии катодов (CATH): %b", CATH);

if ( CATH != get\_cath\_mask( value[(i+1)\*4-1 -: 4] ) )

test\_result = 0;

@(posedge clk\_div);

@(posedge clk);

end

end

endtask

endmodule

*Листинг 2.7 – Тестовый модуль test\_filter*

`timescale 1ns / 1ps

module test\_filter;

reg clk;

initial clk = 0;

always #5 clk <= ~clk;

localparam PRESS = 1, RELEASE = 0;

reg IN\_SIGNAL; initial IN\_SIGNAL = 0;

wire OUT\_SIGNAL\_ENABLE;

FILTER #(5) btn\_c\_filter(

.CLK(clk),

.CLOCK\_ENABLE(1),

.IN\_SIGNAL(IN\_SIGNAL),

.OUT\_SIGNAL\_ENABLE(OUT\_SIGNAL\_ENABLE)

);

localparam TEST\_COUNT = 3;

reg [0:TEST\_COUNT-1] test\_register;

initial

begin

test\_register = {TEST\_COUNT{1'b0}};

test\_filter\_1();

test\_filter\_2();

test\_filter\_3();

test\_show\_stats();

end

task test\_filter\_1;

reg test\_result;

begin

$display("\n[%0t]: Тест 1. Реакция фильтра дребезга на сигнал высокого уровня на шине физ. манипулятора.", $time);

$display("[%0t]: (время удержания сигнала соответствует требуемому)", $time);

send\_signal\_to\_filter(PRESS, 32);

@(posedge clk) test\_result <= (OUT\_SIGNAL\_ENABLE == 1'b1);

send\_signal\_to\_filter(RELEASE, 32);

test\_info(1, test\_result);

end

endtask

task test\_filter\_2;

reg test\_result;

begin

$display("\n[%0t]: Тест 2. Реакция фильтра дребезга на сигнал высокого уровня на шине физ. манипулятора.", $time);

$display("[%0t]: (время удержания сигнала меньше требуемого)", $time);

send\_signal\_to\_filter(PRESS, 16);

@(posedge clk); test\_result = (OUT\_SIGNAL\_ENABLE == 1'b0);

test\_info(2, test\_result);

end

endtask

task test\_filter\_3;

reg test\_result;

begin

$display("\n[%0t]: Тест 3. Реакция фильтра дребезга на сигнал низкого уровня на шине физ. манипулятора.", $time);

*Продолжение Листинга 2.7*

send\_signal\_to\_filter(RELEASE, 32);

@(posedge clk); test\_result = (OUT\_SIGNAL\_ENABLE == 1'b0);

test\_info(3, test\_result);

end

endtask

task test\_info;

input integer test\_number;

input test\_result;

begin

test\_register[test\_number-1] = test\_result;

if (test\_result)

$display("[%0t]: Тест %0d пройден.", $time, test\_number);

else

$display("[%0t]: Тест %0d НЕ пройден.", $time, test\_number);

end

endtask

task test\_show\_stats;

integer i, test\_counter;

begin

$display("\nРезультаты тестирования:");

test\_counter = 0;

for (i = 0; i < TEST\_COUNT; i = i + 1)

begin

if (test\_register[i])

$display("Тест %2d пройден.", i+1);

else

$display("Тест %2d НЕ пройден.", i+1);

test\_counter = test\_counter + (test\_register[i] ? 1 : 0);

end

$display("Пройдено тестов: %0d/%0d", test\_counter, TEST\_COUNT);

end

endtask

task send\_signal\_to\_filter;

input signal\_in;

input [6:0] ticks;

begin

@(posedge clk);

IN\_SIGNAL <= signal\_in;

$display("[%0t]: Сигнал %b подан на линию.", $time, signal\_in);

repeat(ticks + 2)

@(posedge clk);

IN\_SIGNAL <= 0;

$display("[%0t]: Сигнал %b убран с линии, подан сигнал 0", $time, signal\_in);

end

endtask

endmodule

*Листинг 2.8 – Тестовый модуль test\_clk\_div*

`timescale 1ns / 1ns

module test\_clk\_div;

reg clk;

initial clk = 0;

always #5 clk <= ~clk;

wire clk\_div\_out;

clk\_div clk\_div1 (

.clk(clk),

.clk\_div(clk\_div\_out)

);

integer clk\_div\_period = 100\_000;

realtime t\_begin, t\_end;

initial

begin

@(posedge clk\_div\_out);

t\_begin = $realtime;

@(posedge clk\_div\_out);

t\_end = $realtime;

$display("Ожидаемый период сигнала: %0d", clk\_div\_period);

$write("Фактический период сигнала: ");

$write((t\_end - t\_begin));

if ( (t\_end - t\_begin) == clk\_div\_period )

$display("\nТест пройден.");

else

$display("\nТест НЕ пройден.");

end

endmodule

*Листинг 2.9 – Тестовый модуль test\_fsm\_div*

module test\_fsm\_div;

reg clk;

initial clk = 0;

always #5 clk <= ~clk;

reg valid\_in, reset;

reg [3:0] d\_in;

initial

begin

valid\_in = 0;

reset = 0;

d\_in = 4'd0;

end

wire [3:0] d\_out;

wire valid\_out, error\_out;

fsm\_div uut(

.clk(clk),

.valid\_in(valid\_in),

.reset(reset),

.valid\_out(valid\_out),

.d\_in(d\_in),

.d\_out(d\_out),

.error\_out(error\_out)

);

initial

begin

test\_fsm\_1();

test\_fsm\_2();

test\_fsm\_3();

test\_fsm\_4();

test\_fsm\_5();

test\_fsm\_6();

test\_fsm\_7();

test\_fsm\_8();

test\_fsm\_9();

test\_fsm\_10();

test\_fsm\_11();

test\_fsm\_12();

test\_show\_stats();

end

localparam NO\_ERROR = 0, DIV\_BY\_ZERO = 1;

task test\_fsm\_1;

reg test\_result;

begin

$display("\n[%0t]: Тест 1. Сброс внутренних регистров автомата.", $time);

test\_reset(test\_result);

test\_info(1, test\_result);

end

endtask

task test\_fsm\_2;

reg test\_result;

begin

$display("\n[%0t]: Тест 2. a / b при b == 0, a != 0", $time);

test\_error(4'd5, 4'd0, DIV\_BY\_ZERO, test\_result);

*Продолжение Листинга 2.9*

test\_info(2, test\_result);

end

endtask

task test\_fsm\_3;

reg test\_result;

begin

$display("\n[%0t]: Тест 3. a / b при b == 0, a == 0", $time);

test\_error(4'd0, 4'd0, DIV\_BY\_ZERO, test\_result);

test\_info(3, test\_result);

end

endtask

task test\_fsm\_4;

reg test\_result;

begin

$display("\n[%0t]: Тест 4. a / b при b != 0, a == 0", $time);

test\_res(4'd0, 4'd1, 4'd0, test\_result);

test\_info(4, test\_result);

end

endtask

task test\_fsm\_5;

reg test\_result;

begin

$display("\n[%0t]: Тест 5. a / b при |a| > |b|, a > 0, b > 0", $time);

test\_res(4'd6, 4'd3, 4'd2, test\_result);

test\_info(5, test\_result);

end

endtask

task test\_fsm\_6;

reg test\_result;

begin

$display("\n[%0t]: Тест 6. a / b при |a| < |b|, a > 0, b > 0", $time);

test\_res(4'd2, 4'd7, 4'd0, test\_result);

test\_info(6, test\_result);

end

endtask

task test\_fsm\_7;

reg test\_result;

begin

$display("\n[%0t]: Тест 7. a / b при |a| > |b|, a < 0, b < 0", $time);

test\_res(-4'd7, -4'd2, 4'd3, test\_result);

test\_info(7, test\_result);

end

endtask

task test\_fsm\_8;

reg test\_result;

begin

$display("\n[%0t]: Тест 8. a / b при |a| < |b|, a < 0, b < 0", $time);

test\_res(-4'd4, -4'd5, 4'd0, test\_result);

test\_info(8, test\_result);

end

endtask

*Продолжение Листинга 2.9*

task test\_fsm\_9;

reg test\_result;

begin

$display("\n[%0t]: Тест 9. a / b при |a| > |b|, a > 0, b < 0", $time);

test\_res(4'd5, -4'd2, -4'd2, test\_result);

test\_info(9, test\_result);

end

endtask

task test\_fsm\_10;

reg test\_result;

begin

$display("\n[%0t]: Тест 10. a / b при |a| < |b|, a > 0, b < 0", $time);

test\_res(4'd1, -4'd6, 4'd0, test\_result);

test\_info(10, test\_result);

end

endtask

task test\_fsm\_11;

reg test\_result;

begin

$display("\n[%0t]: Тест 11. a / b при |a| > |b|, a < 0, b > 0", $time);

test\_res(-4'd8, 4'd2, -4'd4, test\_result);

test\_info(11, test\_result);

end

endtask

task test\_fsm\_12;

reg test\_result;

begin

$display("\n[%0t]: Тест 12. a / b при |a| < |b|, a < 0, b > 0", $time);

test\_res(-4'd3, 4'd5, 4'd0, test\_result);

test\_info(12, test\_result);

end

endtask

localparam TEST\_COUNT = 12;

reg [0:TEST\_COUNT-1] test\_register;

initial test\_register = {TEST\_COUNT{1'b0}};

task test\_info;

input integer test\_number;

input test\_result;

begin

test\_register[test\_number-1] = test\_result;

if (test\_result)

$display("[%0t]: Тест %0d пройден.", $time, test\_number);

else

$display("[%0t]: Тест %0d НЕ пройден.", $time, test\_number);

end

endtask

task test\_show\_stats;

integer i, test\_counter;

begin

$display("\nРезультаты тестирования:");

test\_counter = 0;

for (i = 0; i < TEST\_COUNT; i = i + 1)

begin

if (test\_register[i])

*Продолжение Листинга 2.9*

$display("Тест %2d пройден.", i+1);

else

$display("Тест %2d НЕ пройден.", i+1);

test\_counter = test\_counter + (test\_register[i] ? 1 : 0);

end

$display("Пройдено тестов: %0d/%0d", test\_counter, TEST\_COUNT);

end

endtask

task test\_res;

input signed [3:0] a, b;

input signed [3:0] res\_expected;

output reg test\_result;

reg signed [3:0] res\_real;

reg error;

begin

divide\_a\_b(a, b, res\_real, error);

$display("Входные данные: a = %0d, b = %0d", a, b);

$display("Ожидаемый результат: %0d", res\_expected);

$display("Фактический результат: %0d", res\_real);

test\_result = (res\_expected == res\_real) && (error == NO\_ERROR);

end

endtask

task test\_error;

input signed [3:0] a, b;

input error\_num;

output reg test\_result;

reg signed [3:0] res;

reg error;

begin

divide\_a\_b(a, b, res, error);

$display("Входные данные: a = %0d, b = %0d", a, b);

$display("Ожидаемый номер ошибки: %0d", error\_num);

$display("Фактический номер ошибки: %0d", error);

test\_result = error == error\_num;

end

endtask

task test\_reset;

output reg test\_result;

begin

@(posedge clk) reset <= 1;

@(posedge clk) reset <= 0;

@(posedge clk);

$display("Ожидаемое состояние конечного автомата после сброса: %0d", 0);

$display("Фактическое состояние конечного автомата после сброса: %0d", uut.state);

test\_result = uut.state == 0;

end

*Продолжение Листинга 2.9*

endtask

task divide\_a\_b;

input [3:0] a, b;

output [3:0] res;

output error;

begin

// Ввод первого числа

@(posedge clk);

d\_in <= a; valid\_in <= 1;

@(posedge clk);

valid\_in <= 0;

// Ввод второго числа

@(posedge clk);

d\_in <= b; valid\_in <= 1;

@(posedge clk);

valid\_in <= 0;

// Ожидание результата

@(posedge valid\_out); @(posedge clk);

res = d\_out;

error = error\_out;

end

endtask

endmodule

*Листинг 2.10–Тестовый модуль test\_seven\_seg*

`timescale 1ns / 1ps

module test\_seven\_seg;

reg clk;

initial clk = 0;

always #5 clk <= ~clk;

localparam AN\_COUNT = 8;

localparam CATH\_COUNT = 7;

localparam DIGIT\_SIZE = 4;

localparam DIGIT\_COUNT = 16;

reg CE, RESET;

reg [AN\_COUNT\*DIGIT\_SIZE-1:0] NUMBER;

reg [AN\_COUNT-1:0] AN\_MASK;

initial

begin

CE = 1;

RESET = 0;

NUMBER = {(AN\_COUNT\*DIGIT\_SIZE){1'b0}};

end

wire [AN\_COUNT-1:0] AN;

wire [CATH\_COUNT-1:0] CATH;

seven\_seg uut (

.clk(clk),

.CE(1),

.RESET(RESET),

.NUMBER(NUMBER),

.AN\_MASK(AN\_MASK),

.AN(AN),

.CATH(CATH)

);

initial

begin

test\_seven\_segments(8'b00101100);

test\_show\_stats();

end

function [6:0] get\_cath\_mask;

input [3:0] number;

begin

case (number)

4'h0: get\_cath\_mask = 7'b1000000;

4'h1: get\_cath\_mask = 7'b1111001;

4'h2: get\_cath\_mask = 7'b0100100;

4'h3: get\_cath\_mask = 7'b0110000;

4'h4: get\_cath\_mask = 7'b0011001;

4'h5: get\_cath\_mask = 7'b0010010;

4'h6: get\_cath\_mask = 7'b0000010;

4'h7: get\_cath\_mask = 7'b1111000;

4'h8: get\_cath\_mask = 7'b0000000;

4'h9: get\_cath\_mask = 7'b0010000;

4'ha: get\_cath\_mask = 7'b0001000;

4'hb: get\_cath\_mask = 7'b0000011;

4'hc: get\_cath\_mask = 7'b1000110;

4'hd: get\_cath\_mask = 7'b0100001;

4'he: get\_cath\_mask = 7'b0000110;

*Продолжение Листинга 2.10*

4'hf: get\_cath\_mask = 7'b0001110;

default: get\_cath\_mask = 7'b1111111;

endcase

end

endfunction

function [7:0] get\_an\_mask;

input [2:0] an\_number;

begin

case (an\_number)

3'd0: get\_an\_mask = 8'b11111110;

3'd1: get\_an\_mask = 8'b11111101;

3'd2: get\_an\_mask = 8'b11111011;

3'd3: get\_an\_mask = 8'b11110111;

3'd4: get\_an\_mask = 8'b11101111;

3'd5: get\_an\_mask = 8'b11011111;

3'd6: get\_an\_mask = 8'b10111111;

3'd7: get\_an\_mask = 8'b01111111;

default: get\_an\_mask = 8'b11111111;

endcase

end

endfunction

reg [AN\_COUNT-1:0] test\_an\_register;

reg [DIGIT\_COUNT-1:0] test\_digit\_register;

reg test\_an\_mask\_register;

task test\_seven\_segments;

input [AN\_COUNT-1:0] mask\_value;

reg [3:0] i;

reg [3:0] number;

begin

$display("\n[%0t]: Тест отображения цифр на индикаторах, принципа работы динамической индикации и анодной маски.", $time);

test\_an\_register = {AN\_COUNT{1'b1}};

test\_digit\_register = {DIGIT\_COUNT{1'b1}};

test\_an\_mask\_register = 1'b1;

AN\_MASK = mask\_value;

$display("Битовая маска (AN\_MASK): %b", AN\_MASK);

wait(uut.digit\_counter == AN\_COUNT-1);

@(posedge clk);

number = 0;

repeat(DIGIT\_COUNT)

begin

// Подача числа на входную шину

for (i = 0; i < AN\_COUNT; i = i + 1)

NUMBER[ ((i+1)\*4)-1 -: 4 ] <= number;

@(posedge clk);

$display("\n[%0t]: Тест для цифры: %h", $time, number);

for (i = 0; i < AN\_COUNT; i = i + 1)

begin

$display("Текущий анод: %d", i);

test\_digit\_register[number] <= CATH == get\_cath\_mask(number);

$display("Ожидаемые сигналы на линии катодов (CATH): %b", get\_cath\_mask(number));

$display("Фактические сигналы на линии катодов (CATH): %b", CATH);

*Продолжение Листинга 2.10*

test\_an\_register[number] <= uut.AN\_REG == get\_an\_mask(i);

$display("Ожидаемые сигналы на линии анодов (ДО применения анодной маски): %b", get\_an\_mask(i));

$display("Фактические сигналы на линии анодов (ДО применения анодной маски): %b", uut.AN\_REG);

test\_an\_mask\_register <= AN == (get\_an\_mask(i) | AN\_MASK);

$display("Ожидаемые сигналы на линии анодов (ПОСЛЕ применения анодной маски): %b", get\_an\_mask(i) | AN\_MASK);

$display("Фактические сигналы на линии анодов (ПОСЛЕ применения анодной маски): %b", AN);

if (i != AN\_COUNT-1)

@(posedge clk);

end

number = number + 1;

end

end

endtask

task test\_show\_stats;

localparam TEST\_COUNT = 3;

integer test\_counter, i;

begin

test\_counter = 0;

$display("\n[%0t]: Результаты тестирования:", $time);

// Отображение цифры

if (&(test\_digit\_register))

begin

$display("1. Тест на отображение пройден успешно для всех возможных вариантов цифр.");

test\_counter = test\_counter + 1;

end

else begin

$display("1. Тест на отображение цифр НЕ пройден");

for (i = 0; i < DIGIT\_COUNT; i = i + 1)

if (!test\_digit\_register[i])

$display("Ошибка отображения цифры %d", i);

end

// Динамическая индикация

if (&(test\_an\_register))

begin

test\_counter = test\_counter + 1;

$display("2. Тест работы динамической индикации пройден успешно.");

end

else begin

$display("2. Тест работы динамической индикации НЕ пройден.", i);

for (i = 0; i < AN\_COUNT; i = i + 1)

if (!test\_an\_register[i])

$display("Ошибка на индикаторе %0d.", i);

end

// Анодная маска

if (test\_an\_mask\_register)

begin

$display("3. Тест анодной маски пройден успешно.");

test\_counter = test\_counter + 1;

end

else

$display("3. Тест анодной маски НЕ пройден.");

$display("Пройдено тестов: %0d/%0d.", test\_counter, TEST\_COUNT);

end

endtask

endmodule

*Далее будут представлены коды файлов проектных ограничений в Листингах 2.11 – 2.12*

*Листинг 2.11 – Файл проектных ограничений*

create\_clock -add -name clk\_pin -period 10.00 -waveform {0 5} [get\_ports { clk }]

set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { clk }]

set\_property -dict { PACKAGE\_PIN C12 IOSTANDARD LVCMOS33 } [get\_ports { btn\_reset\_in }]; #IO\_L3P\_T0\_DQS\_AD1P\_15 Sch=cpu\_resetn

set\_property -dict { PACKAGE\_PIN N17 IOSTANDARD LVCMOS33 } [get\_ports { btn\_c\_in }]; #IO\_L9P\_T1\_DQS\_14 Sch=btnc

set\_property -dict { PACKAGE\_PIN J15 IOSTANDARD LVCMOS33 } [get\_ports { SW[0] }]; #IO\_L24N\_T3\_RS0\_15 Sch=sw[0]

set\_property -dict { PACKAGE\_PIN L16 IOSTANDARD LVCMOS33 } [get\_ports { SW[1] }]; #IO\_L3N\_T0\_DQS\_EMCCLK\_14 Sch=sw[1]

set\_property -dict { PACKAGE\_PIN M13 IOSTANDARD LVCMOS33 } [get\_ports { SW[2] }]; #IO\_L6N\_T0\_D08\_VREF\_14 Sch=sw[2]

set\_property -dict { PACKAGE\_PIN R15 IOSTANDARD LVCMOS33 } [get\_ports { SW[3] }]; #IO\_L13N\_T2\_MRCC\_14 Sch=sw[3]

set\_property -dict { PACKAGE\_PIN H17 IOSTANDARD LVCMOS33 } [get\_ports { valid\_out\_LED }];

set\_property -dict { PACKAGE\_PIN T10 IOSTANDARD LVCMOS33 } [get\_ports { CATH[0] }]; #IO\_L24N\_T3\_A00\_D16\_14 Sch=ca

set\_property -dict { PACKAGE\_PIN R10 IOSTANDARD LVCMOS33 } [get\_ports { CATH[1] }]; #IO\_25\_14 Sch=cb

set\_property -dict { PACKAGE\_PIN K16 IOSTANDARD LVCMOS33 } [get\_ports { CATH[2] }]; #IO\_25\_15 Sch=cc

set\_property -dict { PACKAGE\_PIN K13 IOSTANDARD LVCMOS33 } [get\_ports { CATH[3] }]; #IO\_L17P\_T2\_A26\_15 Sch=cd

set\_property -dict { PACKAGE\_PIN P15 IOSTANDARD LVCMOS33 } [get\_ports { CATH[4] }]; #IO\_L13P\_T2\_MRCC\_14 Sch=ce

set\_property -dict { PACKAGE\_PIN T11 IOSTANDARD LVCMOS33 } [get\_ports { CATH[5] }]; #IO\_L19P\_T3\_A10\_D26\_14 Sch=cf

set\_property -dict { PACKAGE\_PIN L18 IOSTANDARD LVCMOS33 } [get\_ports { CATH[6] }]; #IO\_L4P\_T0\_D04\_14 Sch=cg

set\_property -dict { PACKAGE\_PIN J17 IOSTANDARD LVCMOS33 } [get\_ports { AN[0] }]; #IO\_L23P\_T3\_FOE\_B\_15 Sch=an[0]

set\_property -dict { PACKAGE\_PIN J18 IOSTANDARD LVCMOS33 } [get\_ports { AN[1] }]; #IO\_L23N\_T3\_FWE\_B\_15 Sch=an[1]

set\_property -dict { PACKAGE\_PIN T9 IOSTANDARD LVCMOS33 } [get\_ports { AN[2] }]; #IO\_L24P\_T3\_A01\_D17\_14 Sch=an[2]

set\_property -dict { PACKAGE\_PIN J14 IOSTANDARD LVCMOS33 } [get\_ports { AN[3] }]; #IO\_L19P\_T3\_A22\_15 Sch=an[3]

set\_property -dict { PACKAGE\_PIN P14 IOSTANDARD LVCMOS33 } [get\_ports { AN[4] }]; #IO\_L8N\_T1\_D12\_14 Sch=an[4]

set\_property -dict { PACKAGE\_PIN T14 IOSTANDARD LVCMOS33 } [get\_ports { AN[5] }]; #IO\_L14P\_T2\_SRCC\_14 Sch=an[5]

set\_property -dict { PACKAGE\_PIN K2 IOSTANDARD LVCMOS33 } [get\_ports { AN[6] }]; #IO\_L23P\_T3\_35 Sch=an[6]

set\_property -dict { PACKAGE\_PIN U13 IOSTANDARD LVCMOS33 } [get\_ports { AN[7] }]; #IO\_L23N\_T3\_A02\_D18\_14 Sch=an[7]

*Листинг 2.12 – Второй файл проектных ограничений*

create\_clock -add -name clk\_pin -period 10.00 -waveform {0 5} [get\_ports { clk }]

set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { clk }]

set\_property -dict { PACKAGE\_PIN H17 IOSTANDARD LVCMOS33 } [get\_ports { valid\_out\_LED }];

set\_property -dict { PACKAGE\_PIN T10 IOSTANDARD LVCMOS33 } [get\_ports { CATH[0] }]; #IO\_L24N\_T3\_A00\_D16\_14 Sch=ca

set\_property -dict { PACKAGE\_PIN R10 IOSTANDARD LVCMOS33 } [get\_ports { CATH[1] }]; #IO\_25\_14 Sch=cb

set\_property -dict { PACKAGE\_PIN K16 IOSTANDARD LVCMOS33 } [get\_ports { CATH[2] }]; #IO\_25\_15 Sch=cc

set\_property -dict { PACKAGE\_PIN K13 IOSTANDARD LVCMOS33 } [get\_ports { CATH[3] }]; #IO\_L17P\_T2\_A26\_15 Sch=cd

set\_property -dict { PACKAGE\_PIN P15 IOSTANDARD LVCMOS33 } [get\_ports { CATH[4] }]; #IO\_L13P\_T2\_MRCC\_14 Sch=ce

set\_property -dict { PACKAGE\_PIN T11 IOSTANDARD LVCMOS33 } [get\_ports { CATH[5] }]; #IO\_L19P\_T3\_A10\_D26\_14 Sch=cf

set\_property -dict { PACKAGE\_PIN L18 IOSTANDARD LVCMOS33 } [get\_ports { CATH[6] }]; #IO\_L4P\_T0\_D04\_14 Sch=cg

set\_property -dict { PACKAGE\_PIN J17 IOSTANDARD LVCMOS33 } [get\_ports { AN[0] }]; #IO\_L23P\_T3\_FOE\_B\_15 Sch=an[0]

set\_property -dict { PACKAGE\_PIN J18 IOSTANDARD LVCMOS33 } [get\_ports { AN[1] }]; #IO\_L23N\_T3\_FWE\_B\_15 Sch=an[1]

set\_property -dict { PACKAGE\_PIN T9 IOSTANDARD LVCMOS33 } [get\_ports { AN[2] }]; #IO\_L24P\_T3\_A01\_D17\_14 Sch=an[2]

set\_property -dict { PACKAGE\_PIN J14 IOSTANDARD LVCMOS33 } [get\_ports { AN[3] }]; #IO\_L19P\_T3\_A22\_15 Sch=an[3]

set\_property -dict { PACKAGE\_PIN P14 IOSTANDARD LVCMOS33 } [get\_ports { AN[4] }]; #IO\_L8N\_T1\_D12\_14 Sch=an[4]

set\_property -dict { PACKAGE\_PIN T14 IOSTANDARD LVCMOS33 } [get\_ports { AN[5] }]; #IO\_L14P\_T2\_SRCC\_14 Sch=an[5]

set\_property -dict { PACKAGE\_PIN K2 IOSTANDARD LVCMOS33 } [get\_ports { AN[6] }]; #IO\_L23P\_T3\_35 Sch=an[6]

set\_property -dict { PACKAGE\_PIN U13 IOSTANDARD LVCMOS33 } [get\_ports { AN[7] }]; #IO\_L23N\_T3\_A02\_D18\_14 Sch=an[7]

## 2.2 Файл с описанной программой на языке Tcl и результаты работы

Если проект с именем Praс\_4 открыт, то он закрывается. Далее создается новый проект с этим же именем, в который копируются все исходные модули, включая тестовые файлы и файлы проектных ограничений. Создание и настройка сета симуляции для каждого тестового модуля, которая включает создание сетов симуляций, добавление модулей дизайна и тестовых модулей в наборы. Удаление сета симуляции по умолчанию. Создание сетов для файлов проектных ограничений. Удаление сета проектных ограничений по умолчанию и добавление файлов проектных ограничений в проект. Создание IP-ядра VIO. Симуляция на разных наборах тестов и имплементация на разных наборах файлов проектных ограничений. Код на языке Tcl представлен на Листинге 2.13.[4]

*Листинг 2.13 – Код на языке Tcl*

set project\_name prac\_4

set project\_found [llength [get\_projects $project\_name] ]

if {$project\_found > 0} close\_project

set origin\_dir [file dirname [info script]]

cd $origin\_dir

set path $origin\_dir/$project\_name/$project\_name

# Создание проекта

create\_project $project\_name $project\_name -force -part xc7a100tcsg324-1

# Добавление файлов дизайна

file mkdir $path.srcs/designs

set design\_file\_list [list seven\_seg.v fsm\_div.v clk\_div.v top\_fsm\_div.v filter.v vio\_top\_fsm\_div.v ]

foreach s $design\_file\_list {

file copy $origin\_dir/${s} ${path}.srcs/designs/${s}

add\_files -fileset sources\_1 -norecurse ${path}.srcs/designs/${s}

}

# Добавление файлов тестового окружения

file mkdir $path.srcs/testbenches

set test\_name\_list [ list test\_seven\_seg test\_fsm\_div test\_clk\_div test\_filter test\_top\_fsm\_div ]

# Формирование ассоциативного массива формата (тестовый модуль - список файлов дизайнов для тестирования)

set test\_set\_designs(test\_seven\_seg) [ list seven\_seg.v ]

set test\_set\_designs(test\_fsm\_div) [ list fsm\_div.v ]

set test\_set\_designs(test\_clk\_div) [ list clk\_div.v ]

set test\_set\_designs(test\_filter) [ list filter.v ]

set test\_set\_designs(test\_top\_fsm\_div) [ list top\_fsm\_div.v fsm\_div.v clk\_div.v top\_fsm\_div.v filter.v seven\_seg.v ]

*Продолжение Листинга 2.13*

# Создание и настройка сета симуляции для каждого тестового модуля

foreach s $test\_name\_list {

# Создание сета симуляции

set set\_name ${s}\_set

file mkdir ${path}.srcs/testbenches/${set\_name}

create\_fileset -simset $set\_name

set\_property SOURCE\_SET {} [ get\_filesets $set\_name ]

# Добавления модулей дизайна в сет

set designs $test\_set\_designs(${s})

foreach design $designs {

add\_files -fileset $set\_name -norecurse ${path}.srcs/designs/$design

}

# Добавление тестового модуля в сет

file copy $origin\_dir/${s}.v ${path}.srcs/testbenches/${set\_name}/${s}.v

add\_files -fileset $set\_name ${path}.srcs/testbenches/${set\_name}/${s}.v

set\_property TOP ${s} [get\_filesets ${set\_name}]

}

# Удаление сета симуляции по умолчанию

current\_fileset -simset [ get\_filesets test\_seven\_seg\_set ]

delete\_fileset [ get\_filesets sim\_1 ]

file delete -force $path.srcs/sim\_1

# Создание сетов для файлов проектных ограничений

create\_fileset -constrset fsm\_with\_vio\_set

create\_fileset -constrset fsm\_set

# Ассоциативный массив с указанием модуля верхнего уровня

# для каждого сета проектных ограничений

array set constrset\_top\_module {

fsm\_with\_vio\_set vio\_top\_fsm\_div

fsm\_set top\_fsm\_div

}

# Удаление сета проектных ограничений по умолчанию

set\_property constrset fsm\_with\_vio\_set [get\_runs synth\_1]

set\_property constrset fsm\_with\_vio\_set [get\_runs impl\_1]

delete\_fileset [ get\_filesets constrs\_1 ]

file delete -force $path.srcs/constrs\_1

# Добавление файлов проектных ограничений в проект

file mkdir $path.srcs/constrs/fsm\_with\_vio

file mkdir $path.srcs/constrs/fsm

file copy $origin\_dir/c\_fsm.xdc $path.srcs/constrs/fsm/c\_fsm.xdc

file copy $origin\_dir/c\_vio.xdc $path.srcs/constrs/fsm\_with\_vio/c\_vio.xdc

add\_files -fileset fsm\_set -norecurse $path.srcs/constrs/fsm/c\_fsm.xdc

add\_files -fileset fsm\_with\_vio\_set -norecurse $path.srcs/constrs/fsm\_with\_vio/c\_vio.xdc

# Создание IP-ядра VIO

create\_ip -name vio -vendor xilinx.com -library ip -version 3.0 -module\_name vio\_0

set\_property -dict [list \

CONFIG.C\_NUM\_PROBE\_IN {3} \

CONFIG.C\_NUM\_PROBE\_OUT {3} \

CONFIG.C\_PROBE\_IN0\_WIDTH {8} \

CONFIG.C\_PROBE\_IN1\_WIDTH {7} \

*Продолжение Листинга 2.13*

CONFIG.C\_PROBE\_OUT2\_WIDTH {4} \

] [get\_ips vio\_0]

generate\_target {instantiation\_template} [get\_files "$path.srcs/sources\_1/ip/vio\_0/vio\_0.xci"]

update\_compile\_order -fileset sources\_1

generate\_target all [get\_files "$path.srcs/sources\_1/ip/vio\_0/vio\_0.xci"]

catch { config\_ip\_cache -export [get\_ips -all vio\_0] }

export\_ip\_user\_files -of\_objects [get\_files "$path.srcs/sources\_1/ip/vio\_0/vio\_0.xci"] -no\_script -sync -force -quiet

create\_ip\_run [get\_files -of\_objects [get\_fileset sources\_1] "$path.srcs/sources\_1/ip/vio\_0/vio\_0.xci"]

launch\_runs vio\_0\_synth\_1 -jobs 16

wait\_on\_runs vio\_0\_synth\_1

export\_simulation -of\_objects [get\_files "$path.srcs/sources\_1/ip/vio\_0/vio\_0.xci"] -directory "$path.ip\_user\_files/sim\_scripts" -ip\_user\_files\_dir "$path.ip\_user\_files" -ipstatic\_source\_dir "$path.ip\_user\_files/ipstatic" -lib\_map\_path [list {modelsim="$path.cache/compile\_simlib/modelsim"} {questa="$path.cache/compile\_simlib/questa"} {riviera="$path.cache/compile\_simlib/riviera"} {activehdl="$path.cache/compile\_simlib/activehdl"}] -use\_ip\_compiled\_libs -force -quiet

# Симуляция на разных наборах (simulation sets)

foreach t\_set [ get\_filesets test\* ] {

current\_fileset -simset $t\_set

file mkdir $origin\_dir/$project\_name/sim\_output/${t\_set}

# Команда сбрасывает время симуляции

set\_property -name xsim.simulate.runtime -value 0 -objects [get\_filesets ${t\_set}]

launch\_simulation

# Запуск симуляции на 7000us и перенаправление вывода из TCL-консоли в файл

restart

run 7000us > "$origin\_dir/$project\_name/sim\_output/${t\_set}/sim\_output.txt"

close\_sim

}

# Имплементация на разных наборах (constraints set)

foreach c\_set [ get\_filesets fsm\* ] {

set\_property constrset $c\_set [get\_runs synth\_1]

set\_property constrset $c\_set [get\_runs impl\_1]

set\_property TOP $constrset\_top\_module($c\_set) [get\_fileset sources\_1]

reset\_runs synth\_1

launch\_runs synth\_1 -jobs 16

wait\_on\_runs synth\_1

reset\_run impl\_1

launch\_runs impl\_1 -jobs 16

wait\_on\_runs impl\_1

open\_run impl\_1

report\_timing\_summary -file "$origin\_dir/$project\_name/timing\_summary\_${c\_set}"

write\_checkpoint -file "$origin\_dir/checkpoint.dcp"}

## Результатом работы данного кода стало создание проекта, также программой были проведены тесты, результаты которых были записаны в директории sim\_output (Рисунок 2.1). Программа провела все тесты и записала итоговый результат в соответствующие файлы (Рисунок 2.2 – 2.5). Были проведены синтез и имплементация на каждом наборе файлов проектных ограничений. Результаты были записаны во временную сводку (Рисунок 2.6 – 2.9).[5]

## 

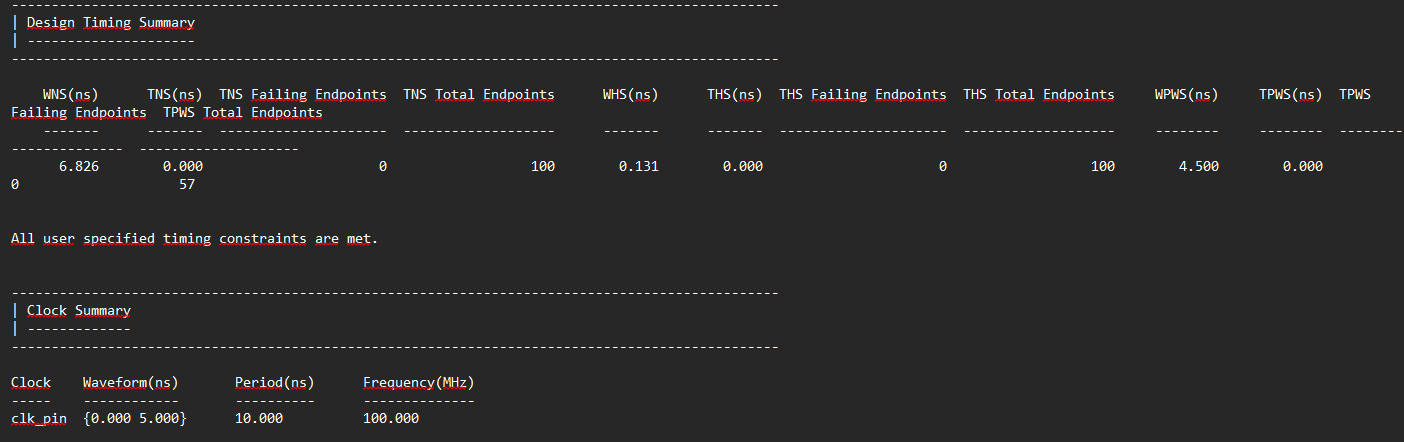
**Рисунок 2.1 – Директория sim\_output**

## 

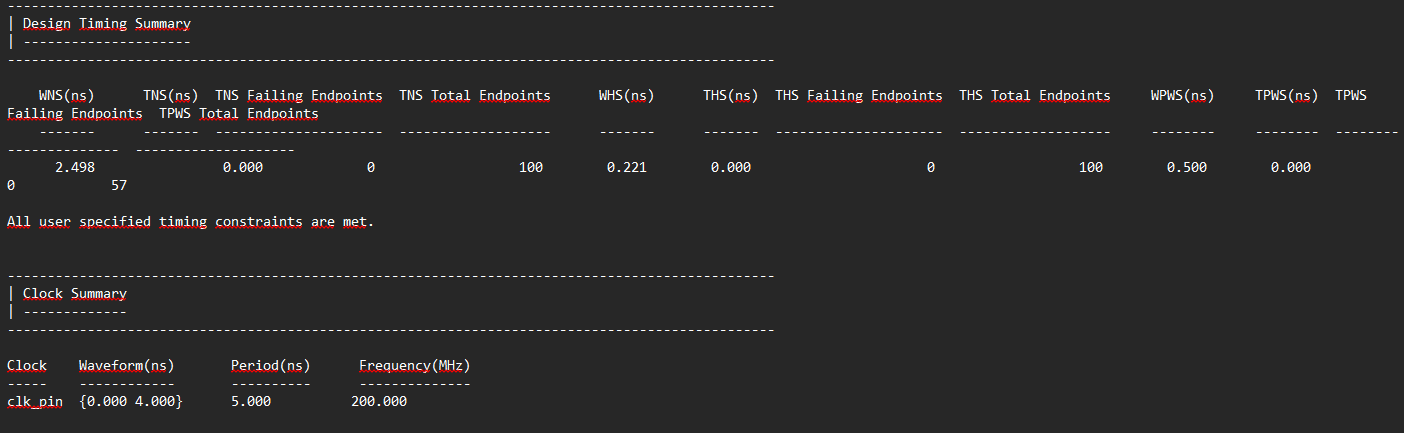
**Рисунок 2.2 – Файл результата симуляции**

## 

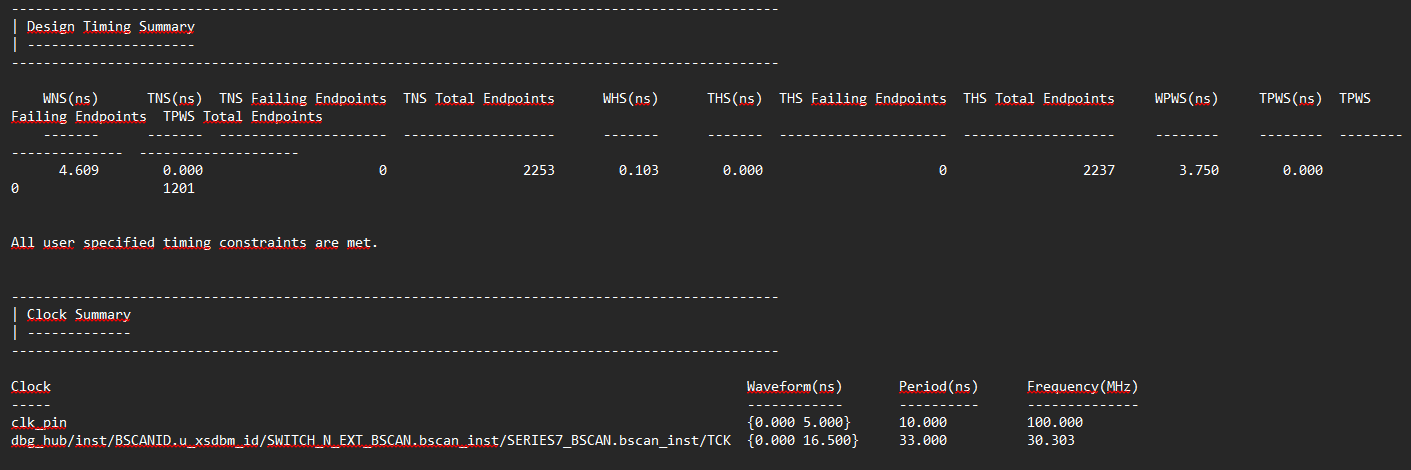
**Рисунок 2.3 – Файл результата симуляции модуля управления индикаторами**

****

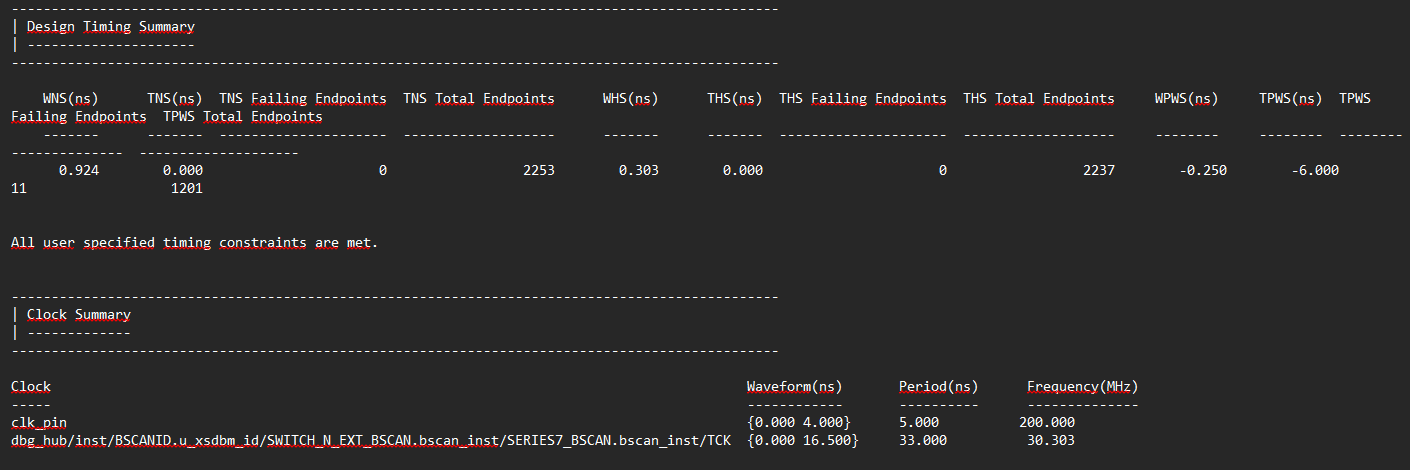
**Рисунок 2.4 – Значения задержек по Setup и Hold для набора проектных ограничений без vio с частотой 100 МГц**

****

**Рисунок 2.7 – Значения задержек по Setup и Hold для набора проектных ограничений без vio с частотой 200 МГц**

****

**Рисунок 2.8 – Значения задержек по Setup и Hold для набора проектных ограничений с vio с частотой 100 МГц**

****

**Рисунок 2.9 – Значения задержек по Setup и Hold для набора проектных ограничений с vio с частотой 200 МГц**

Программа успешно завершила свою работу, результаты временных задержек были выведены в соответствующие файлы.

ЗАКЛЮЧЕНИЕ

Таким образом, в данной практической работе был создан файл Tcl, в котором присутствует реализация создания нового проекта с исходными модулями, тестовыми модулями и файлом проектных ограничений, запуск симуляции, синтеза и имплементации на наборах тестов и файлов проектных ограничений. Синтез и имплементация была проведена без и с использованием IP-ядра VIO, с вариациями частот, равными 100 и 200 МГц. Полученный результаты были сохранены в соответствующие файлы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания по ПР № 4 — URL: https://online-edu.mirea.ru/mod/resource/view.php?id=405132.

2. Смирнов С.С. Информатика [Электронный ресурс]: Методические указания по выполнению практических и лабораторных работ / С.С. Смирнов — М., МИРЭА — Российский технологический университет, 2018. — 1 электрон. опт. диск (CD-ROM).

3. Тарасов И.Е. ПЛИС Xilinx. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования. — М.: Горячая линия — Телеком, 2021. — 538 с.: ил.

4. Антик М.И. Дискретная математика [Электронный ресурс]: Учебное пособие / Антик М.И., Казанцева Л.В. — М.: МИРЭА — Российский технологический университет, 2018 — 1 электрон. опт. диск (CD-ROM).

5. Практическая работа № 3 – URL: https://online-edu.mirea.ru/pluginfile.php?file=%2F1225652%2Fassignsubmission\_file%2Fsubmission\_files%2F3439663%2FПрактика-3.pdf&amp;forcedownload=1