



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-11-23

Туктаров Т.А

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, 20 листингов. Количество страниц в работе — 41.

# СОДЕРЖАНИЕ

1	ПОСТАНОВКА ЗАДАЧИ .....	4
2	ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ .....	5
2.1	Исходный код файлов проекта .....	5
2.2	Файл с описанной программой на языке Tc1 и результаты работы .....	26
	ЗАКЛЮЧЕНИЕ .....	40
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	41

# 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. Составить отчёт.

## 2 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
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
            end
        endcase
    end
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;
    endcase
end
```

### 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)
    );
end
endtask
```

### 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);
end
endtask
```

### 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
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'b10000000;
        4'h1: get_cath_mask = 7'b11111001;
        4'h2: get_cath_mask = 7'b0100100;
        4'h3: get_cath_mask = 7'b0110000;
        4'h4: get_cath_mask = 7'b00111001;
        4'h5: get_cath_mask = 7'b0010010;
        4'h6: get_cath_mask = 7'b0000010;
        4'h7: get_cath_mask = 7'b11111000;
        4'h8: get_cath_mask = 7'b00000000;
        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'b00011110;
        default: get_cath_mask = 7'b11111111;
    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
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_resetrn
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

На листинге 2.13 представлен полный код tcl файла. Далее идет разбор его работы и объяснение команд.

*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 xc7a100tcsq324-1

# Добавление файлов дизайна
file mkdir $path.srscs/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} ${path}.srscs/designs/${s}
    add_files -fileset sources_1 -norecurse ${path}.srscs/designs/${s}
}
# Добавление файлов тестового окружения
file mkdir $path.srscs/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 ]

# Создание и настройка сета симуляции для каждого тестового модуля
foreach s $test_name_list {

    # Создание сета симуляции
    set set_name ${s}_set
    file mkdir ${path}.srscs/testbenches/${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 ${path}.srscs/designs/$design
    }
    # Добавление тестового модуля в сет
    file copy $origin_dir/${s}.v ${path}.srscs/testbenches/${set_name}/${s}.v
    add_files -fileset $set_name ${path}.srscs/testbenches/${set_name}/${s}.v
    set_property TOP ${s} [get_filesets $set_name]
}
}
```

### *Продолжение Листинга 2.13*

```
# 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

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} \
    CONFIG.C_PROBE_OUT2_WIDTH {4} \
] [get_ips vio_0]

generate_target{instantiation_template}[get_files$path.srcs/sources_1/ip/vio_0/v
io_0.xci"]
update_compile_order -fileset sources_1
generate_target all [get_files "$path.srcs/sources_1/ip/vio_0/vio_0.xci"]
```

### Продолжение Листинга 2.13

```
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"
}
```

#### Листинг 2.14

```
# устанавливаем название проекта
set project_name prac_4

# Если есть открытые проекты, закрыть проект
set project_found [llength [get_projects $project_name] ]
if {$project_found > 0} close_project

# Получение и переход в нужные директории проекта
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 xc7a100tcs324-1
```

В первых 14-и строчках(листинг 2.14) кода устанавливается имя проекта, проводится проверка на присутствие открытых проектов (если такие присутствуют, проект закрывается), переход в рабочую директорию и создание самого проекта.

#### Листинг 2.15

```
# Добавление файлов тестового окружения
file mkdir $path.srscs/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 ]
```

Далее в листинге 2.15 производится добавление тестовых файлов, а также создание ассоциативного массива формата «тестовый модуль - список файлов дизайнов для тестирования»



### Листинг 2.16

```
# 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}]
}
```

В листинге 2.16 создаются сет симуляций для каждого тестового модуля с помощью цикла `foreach`, который берет имена модулей из списка `test_name_list`.

### Листинг 2.17

```
# 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
```

В листинге 2.17 происходит удаление сетов симуляции и проектных ограничений по умолчанию, и добавление вместо них новых. Также создается массив с указанием модулей верхнего уровня

### Листинг 2.18

```
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} \
    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
```

В листинге 2.18 создается создание IP-ядра VIO.

### Листинг 2.19

```
# 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
}
```

На листинге 2.19 проводится симуляция. С помощью цикла foreach для каждого сета создается папка для результатов симуляций, запускается симуляция и производится перенаправление вывода из TCL-консоли в файл.

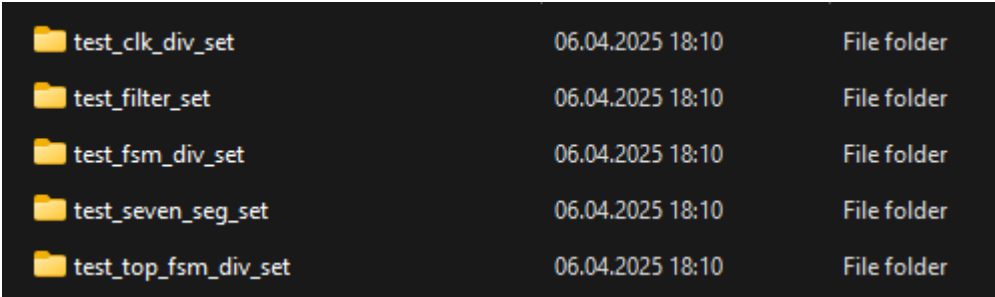
### Листинг 2.20

```
# 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"
}
```

На листинге 2.20 происходит аналогичный процесс, только с имплементацией.

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]



test_clk_div_set	06.04.2025 18:10	File folder
test_filter_set	06.04.2025 18:10	File folder
test_fsm_div_set	06.04.2025 18:10	File folder
test_seven_seg_set	06.04.2025 18:10	File folder
test_top_fsm_div_set	06.04.2025 18:10	File folder

Рисунок 2.1 – Директория `sim_output`

```
Ожидаемый результат: 0
Фактический результат: 0
[445000]: Тест 8 пройден.

[445000]: Тест 9.  $a / b$  при  $|a| > |b|$ ,  $a > 0$ ,  $b < 0$ 
Входные данные:  $a = 5$ ,  $b = -2$ 
Ожидаемый результат: -2
Фактический результат: -2
[505000]: Тест 9 пройден.

[505000]: Тест 10.  $a / b$  при  $|a| < |b|$ ,  $a > 0$ ,  $b < 0$ 
Входные данные:  $a = 1$ ,  $b = -6$ 
Ожидаемый результат: 0
Фактический результат: 0
[565000]: Тест 10 пройден.

[565000]: Тест 11.  $a / b$  при  $|a| > |b|$ ,  $a < 0$ ,  $b > 0$ 
Входные данные:  $a = -8$ ,  $b = 2$ 
Ожидаемый результат: -4
Фактический результат: -4
[625000]: Тест 11 пройден.

[625000]: Тест 12.  $a / b$  при  $|a| < |b|$ ,  $a < 0$ ,  $b > 0$ 
Входные данные:  $a = -3$ ,  $b = 5$ 
Ожидаемый результат: 0
Фактический результат: 0
[685000]: Тест 12 пройден.

Результаты тестирования:
Тест 1 пройден.
Тест 2 пройден.
Тест 3 пройден.
Тест 4 пройден.
Тест 5 пройден.
Тест 6 пройден.
Тест 7 пройден.
Тест 8 пройден.
Тест 9 пройден.
Тест 10 пройден.
Тест 11 пройден.
Тест 12 пройден.
Пройдено тестов: 12/12
```

Рисунок 2.2 – Файл результата симуляции

```

Фактические сигналы на линии анодов (ПОСЛЕ применения анодной маски): 11111111
Текущий анод: 3
Ожидаемые сигналы на линии катодов (CATH): 0001110
Фактические сигналы на линии катодов (CATH): 0001110
Ожидаемые сигналы на линии анодов (ДО применения анодной маски): 11110111
Фактические сигналы на линии анодов (ДО применения анодной маски): 11110111
Ожидаемые сигналы на линии анодов (ПОСЛЕ применения анодной маски): 11111111
Фактические сигналы на линии анодов (ПОСЛЕ применения анодной маски): 11111111
Текущий анод: 4
Ожидаемые сигналы на линии катодов (CATH): 0001110
Фактические сигналы на линии катодов (CATH): 0001110
Ожидаемые сигналы на линии анодов (ДО применения анодной маски): 11101111
Фактические сигналы на линии анодов (ДО применения анодной маски): 11101111
Ожидаемые сигналы на линии анодов (ПОСЛЕ применения анодной маски): 11101111
Фактические сигналы на линии анодов (ПОСЛЕ применения анодной маски): 11101111
Текущий анод: 5
Ожидаемые сигналы на линии катодов (CATH): 0001110
Фактические сигналы на линии катодов (CATH): 0001110
Ожидаемые сигналы на линии анодов (ДО применения анодной маски): 11011111
Фактические сигналы на линии анодов (ДО применения анодной маски): 11011111
Ожидаемые сигналы на линии анодов (ПОСЛЕ применения анодной маски): 11111111
Фактические сигналы на линии анодов (ПОСЛЕ применения анодной маски): 11111111
Текущий анод: 6
Ожидаемые сигналы на линии катодов (CATH): 0001110
Фактические сигналы на линии катодов (CATH): 0001110
Ожидаемые сигналы на линии анодов (ДО применения анодной маски): 10111111
Фактические сигналы на линии анодов (ДО применения анодной маски): 10111111
Ожидаемые сигналы на линии анодов (ПОСЛЕ применения анодной маски): 10111111
Фактические сигналы на линии анодов (ПОСЛЕ применения анодной маски): 10111111
Текущий анод: 7
Ожидаемые сигналы на линии катодов (CATH): 0001110
Фактические сигналы на линии катодов (CATH): 0001110
Ожидаемые сигналы на линии анодов (ДО применения анодной маски): 01111111
Фактические сигналы на линии анодов (ДО применения анодной маски): 01111111
Ожидаемые сигналы на линии анодов (ПОСЛЕ применения анодной маски): 01111111
Фактические сигналы на линии анодов (ПОСЛЕ применения анодной маски): 01111111

[1355000]: Результаты тестирования:
1. Тест на отображение пройден успешно для всех возможных вариантов цифр.
2. Тест работы динамической индикации пройден успешно.
3. Тест анодной маски пройден успешно.
Пройдено тестов: 3/3.

```

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

Design Timing Summary

WNS(ns)	TNS(ns)	TNS Failing Endpoints	TNS Total Endpoints	WHS(ns)	THS(ns)	THS Failing Endpoints	THS Total Endpoints	WPWS(ns)	TPWS(ns)	TPWS
Failing Endpoints	TPWS	Total Endpoints								
0	6.826	0.000	0	100	0.131	0.000	0	100	4.500	0.000
		57								

All user specified timing constraints are met.

Clock Summary

Clock	Waveform(ns)	Period(ns)	Frequency(MHz)
clk_pin	{0.000 5.000}	10.000	100.000

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

Design Timing Summary

WNS(ns)

TNS(ns)

TNS Failing Endpoints

TNS Total Endpoints

WHS(ns)

THS(ns)

THS Failing Endpoints

THS Total Endpoints

WPWS(ns)

TPWS(ns)

TPWS

Failing Endpoints

TPWS

Total Endpoints

0

2.498

57

0.000

0

100

0.221

0.000

0

100

0.500

0.000

All user specified timing constraints are met.

Clock Summary

Clock

Waveform(ns)

Period(ns)

Frequency(MHz)

clk\_pin

{0.000 4.000}

5.000

200.000

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

Design Timing Summary

WNS(ns)	TNS(ns)	TNS Failing Endpoints	TNS Total Endpoints	WHS(ns)	THS(ns)	THS Failing Endpoints	THS Total Endpoints	WPWS(ns)	TPWS(ns)	TPWS
Failing Endpoints	TPWS	Total Endpoints								
0	4.609	0.000	0	2253	0.103	0.000	0	2237	3.750	0.000
		1201								

All user specified timing constraints are met.

Clock Summary

Clock	Waveform(ns)	Period(ns)	Frequency(MHz)
clk_pin	{0.000 5.000}	10.000	100.000
dbg_hub/inst/BSCANID.u_xsdbrm_id/SWITCH_N_EXT_BSCAN.bscan_inst/SERIES7_BSCAN.bscan_inst/TCK	{0.000 16.500}	33.000	30.303

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

Design Timing Summary

**Рисунок 2.9 – Значения задержек по Setup и Hold для набора проектных ограничений с v10 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 Tc1, 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. Полученные 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&forcedownload=1](https://online-edu.mirea.ru/pluginfile.php?file=%2F1225652%2Fassignsubmission_file%2Fsubmission_files%2F3439663%2FПрактика-3.pdf&forcedownload=1)