МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт Информационных Технологий
Кафедра Вычислительной Техники (ВТ)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

«Управление семисегментными индикаторами»

по дисциплине

«Схемотехника устройств компьютерных систем»

Выполнил студент группы ИВБО-12-23 — Туктаров Т.А.

Принял преподаватель кафедры ВТ — Дуксина И.И.

Лабораторная работа выполнена «__»_____2025 г.

«Зачтено» «__»_____2025 г.

Москва 2025

АННОТАЦИЯ

Данная работа включает в себя 8 рисунков и 9 листингов. Количество страниц в работе — 24.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной лабораторной работе рассматриваются вопросы индикации при помощи семисегментных индикаторов, объединённых в дисплей в составе отладочной платы серии Xilinx Nexys[1-2]. Разрабатываемое устройство представляет собой устройство хранения истории ввода[3]. Ввод очередного значения осуществляется при помощи движковых переключателей платы Xilinx Nexys. Для подтверждения ввода используется кнопка BTN_C платы Xilinx Nexys. Хранимая история ввода должна отображаться при помощи семисегментных индикаторов в составе платы Xilinx Nexys. Последнее введённое значение должно отображаться на крайнем правом индикаторе правого дисплея. Далее, справа налево должны располагаться ранее введённые значения в порядке их прихода (чем раньше пришло значение, тем левее оно располагается). Количество хранимых значений прямо пропорционально количеству индикаторов. Для отладочной платы Nexys A7 число хранимых значений
равно 8.

1 СОЗДАНИЕ НЕОБХОДИМЫХ МОДУЛЕЙ НА VERILOG HDL

1.1 Описание дополнительных модулей

В работе буду использованы модули счётчика, делителя частоты, синхронизатора и фильтра дребезга контактов. Рассмотрим каждый модуль.

Начнём с модуля счётчика. Название модуля – «count». Модуль имеет следующие параметры: «MODULE», отвечающий за модуль счёта, по умолчанию равный 2; «STEP», отвечающий за шаг счёта, по умолчанию равный 1. Модуль обладает следующими входными портами: «clk» - синхросигнал, «reset» - сброс, «enable» – разрешающий сигнал, «direction» - направление счёта; выходной регистр «cnt». С помощью оператора «initial» происходит инициализация значения «cnt» значением 0. Далее в блоке «always» при поступлении переднего фронта синхросигнала «clk» происходит изменение выходного регистра «cnt». При единице на «reset» значение счётчика «cnt» будет сброшено в ноль, иначе при единице на «enable» будет изменено значение «cnt» в соответствии с направлением, модулем и шагом счёта.

Реализация модуля представлена в Листинге 1.1. RTL-схема представлена на Рисунке 1.1.

Листинг 1.1 – Реализация модуля параметризированного универсального реверсивного счётчика

```
`timescale 1ns / 1ps
module count# (step = 1, mod = 16) (
    input clk,
    input dir,
    input RE, //Reset
    input CE, //Clock enable
    output reg [$clog2(mod) -1:0] out
);
    initial out  = 0;

    always@(posedge clk or posedge RE)
    begin
        if (RE)
            out <= 0; // Reset
        else if (CE) begin
            if (dir == 0)
                out <= (out + step) % mod;
            else
                out <= (mod + out - step) % mod;
        end
    end
endmodule
```

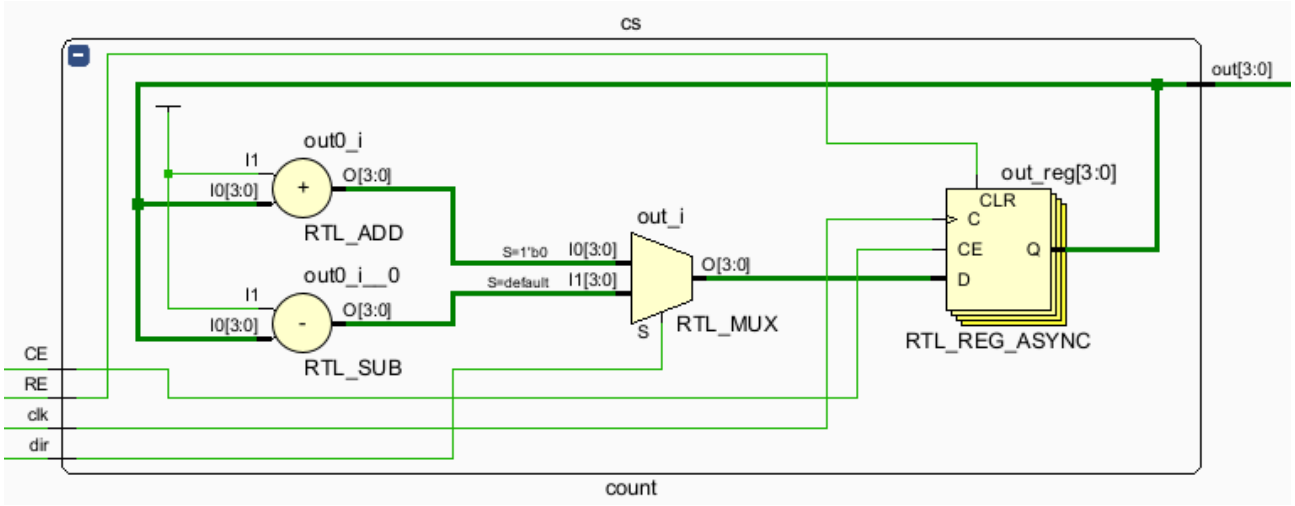

Рисунок 1.1 - RTL-схема модуля универсального параметризированного реверсивного счётчика

Следующий модуль делителя частоты. Название модуля – «clk_divider». Модуль имеет параметр «div», отвечающий за значение, на которое будет изменяться частота синхросигнала, по умолчанию равный 2. Модуль обладает входным портом «clk» - синхросигнал; и выходной регистр «clk_div».

Далее создаётся экземпляр счётчика «cntr» с параметрами «STEP» со значением единицы и «MODULE» со значением «DIV/2». После происходит присваивание портам соответствующих значений. На вход «reset» подаётся 0, «direction» подаётся 0, так как направлению счёта у нас постоянное на суммирование, на порт «enable» подаётся 1. С помощью оператора «initial» происходит инициализация значения «clk_div» значением 0. Далее с помощью блока «always» отслеживается поступление переднего фронта «clk». При равенстве значение «cnt», полученного с выхода счётчика, значению 0, будет произведена инвертация значения «clk_div».

Реализация модуля представлена в Листинге 1.2. RTL-схема представлена на Рисунке 1.2.

Листинг 1.2 – Реализация модуля параметризированного делителя частоты

```
`timescale 1ns / 1ps
module clk_divider#(div = 2) (
    input clk,
    output reg clk_div
);
    reg l;
    wire [8:0] out;
    count #(.step(1), .mod(div/2)) cntr(
        .clk(clk),
        .RE(1'b0),
        .CE(1'b1),
        .dir(1'b0),
        .out(out)
    );
    initial clk_div = 0;
    always@(posedge clk)begin
        if (out ==0 && l==1)
        clk_div = ~clk_div;
        l<=1;
    end
endmodule
```

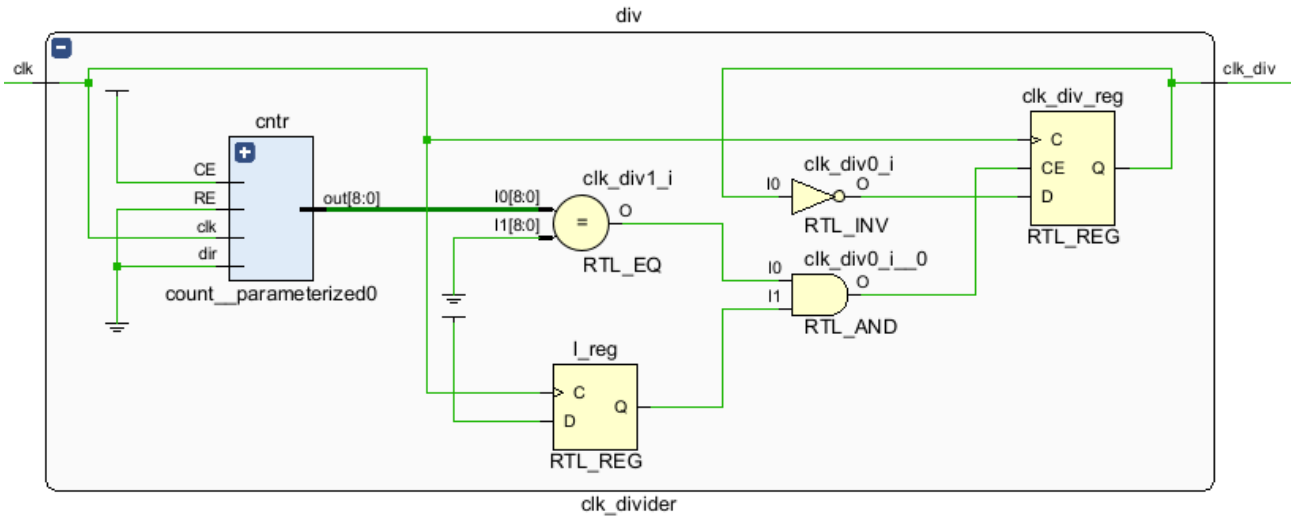

Рисунок 1.2 - RTL-схема модуля параметризированного делителя частоты

Следующий модуль синхронизатора. Название модуля – «synchro». Модуль обладает следующими входными портами: «clk» - синхросигнал, «in» - сигнал с кнопки; и выходом «out». Объявляются два регистра – «a» и «b». В блоке «always», работающем по переднему фронту синхросигнала, регистру «b» присваивается значение на регистре «a», а регистру «a» – значение на входном порте «in». Далее с помощью оператора непрерывного присваивания «assign» выходному порту присваивается значение на регистре «b».

Реализация модуля представлена в Листинге 1.3. RTL-схема представлена на Рисунке 1.3.

Листинг 1.3 – Реализация модуля синхронизатора

```
`timescale 1ns / 1ps
module synchro(
    input in_signal,
    input clk,
    output q,
    output notq
);
    reg new_signal = 1'bx;
    reg last_signal = 1'bx;
    wire lq;
    dtrigger tr1(
        .C(clk),
        .D(new_signal),
        .en(1'b1),
        .q(lq)
    );
    dtrigger tr2(
        .C(clk),
        .D(last_signal),
        .en(1'b1),
        .q(q)
    );
    always @(posedge clk) begin
        new_signal=in_signal;
        last_signal=lq;
    end
endmodule
```

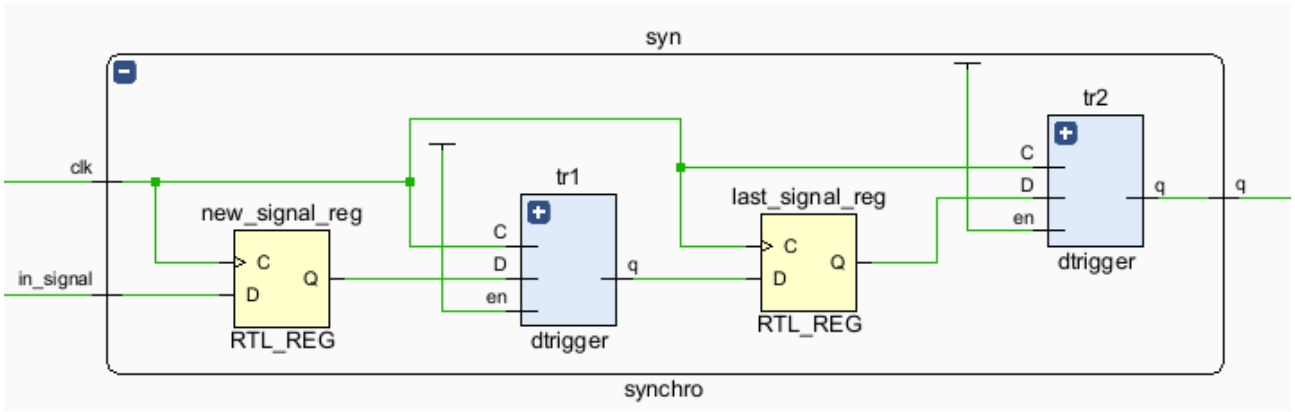

Рисунок 1.3 - RTL-схема модуля синхронизатора

Следующий модуль фильтра дребезга контактов. Название модуля – «filtercon». Модуль обладает параметром «MODULE», по умолчанию равный 8. Модуль обладает следующими входными портами: «clk» - синхросигнал, «in_signal» - сигнал с кнопки; и выходными регистрами: «out_signal» - значение кнопки без дребезга, «out_signal_enable» - сигнал о нажатии кнопки.

Создается экземпляр модуля «synchronizer» с названием «sync». К порту «in» подключается цепь «in_signal», к порту «clk» – синхросигнал «clk», к выходному порту «out» подключается цепь «sync_signal». Далее создается экземпляр модуля «counter» с названием «cntr», в параметр «MODULE» подается

«MODULE» фильтра, а в параметр «STEP» подается 1. К порту «clk» подключается синхросигнал «clk», в порт «reset» подается результат выражения «sync_signal~^out_signal», к порту «enable» подключается параметр «CLOCK_ENABLE», в порт «direction» подается 0, а к выходному порту «cnt» подключается цепь «counter_res».

В блоке «always», работающему по переднему фронту, регистру «out_signal» присваивается значение на цепи «sync_signal», если выражение «&(counter_res) & CLOCK_ENABLE» равняется 1, также «out_signal_enable» присваивается результат выражения «&(counter_res) & sync_signal & CLOCK_ENABLE».

Реализация модуля представлена в Листинге 1.4. RTL-схема представлена на Рисунке 1.4.

Листинг 1.4 – Реализация модуля устранения дребезга контактов

```
`timescale 1ns / 1ps
module filtercon #(mode = 2)(
    input in_signal,
    input clock_enable,
    input clk,
    output wire out_signal,
    output out_signal_enable,
    output wire [1:0] q_count
);

    wire out_sync;
    wire out_xnor;
    wire out_first_and;
    wire out_second_and;
    wire out_third_and;
    synchro syn(
        .in_signal(in_signal),
        .clk(clk),
        .q(out_sync)
    );

    count cs(
        .clk(clk),
        .RE(out_sync~^out_signal),
        .dir(0),
        .CE(clock_enable),
        .out(q_count)
    );

    dtrigger dt1(
        .C(clk),
        .D(out_sync),
        .en(out_second_and),
        .q(out_signal)
    );
    dtrigger dt2(
        .C(clk),
        .D(out_third_and),
        .en(1'b1),
        .q(out_signal_enable)
    );

    assign out_first_and = &q_count;
    assign out_second_and = out_first_and & clock_enable;
    assign out_third_and = out_second_and & out_sync;

endmoduleend


endmodule
```

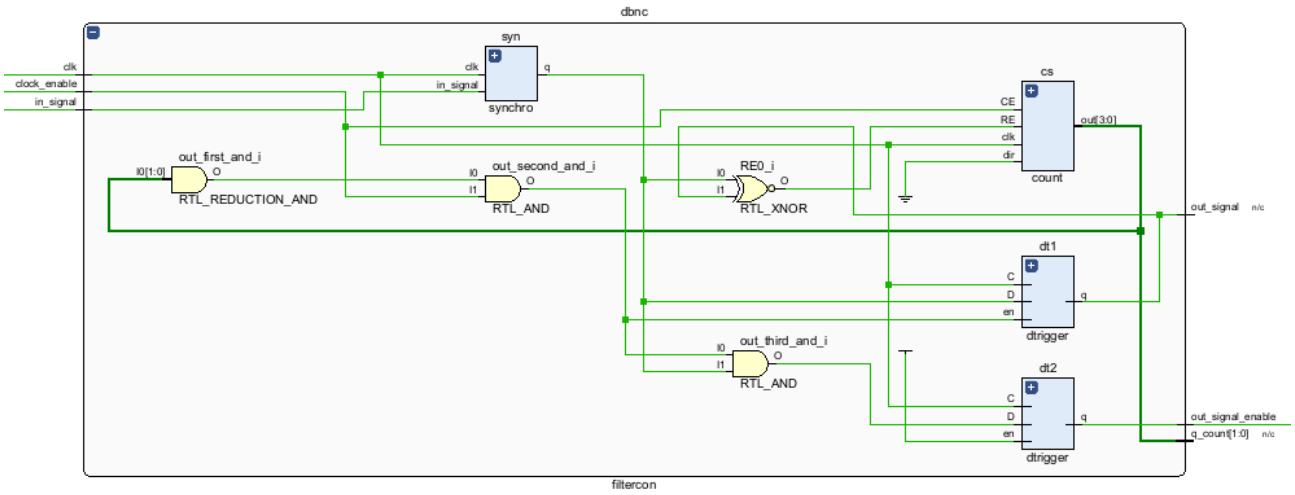

Рисунок 1.4 - RTL-схема модуля синхронизатора

1.2 Создание модуля управления семисегментными индикаторами

Название модуля – «SevenSegmentLED». Модуль имеет следующие порты: входная восьмибитная шина «AN_MASK» – маска для отключения части семисегментных индикаторов, входная 32-битная шина «NUMBER» - значение, выводимое на дисплей, входной порт «RESET» - сброс, входной порт «clk» – синхросигнал, выходная восьмибитная шина «AN» – значения анодов для всех индикаторов, выходной восьмибитный регистр «SEG» - значения катодов.

Объявляется и инициализируется нулем восьмибитный регистр «AN_REG» для управления анодами семисегментного индикатора. Объявляется трехбитный регистр «counter_res» для определения позиции горящего символа на дисплее. Объявляется четырехбитная шина «NUMBER_SPLITTER» из восьмибитных проводов. Далее с помощью ключевого слова «genvar» объявляется переменная «i», которая используется только в блоке «generate». В блоке «generate» находится цикл «for», с помощью которого шина «NUMBER» разделяется на 8 частей, объединенных в шине «NUMBER_SPLITTER».

С помощью оператора непрерывного присваивания «assign» к выходному порту «AN» подключается результат выражения «AN_REG | AN_MASK».

Создаётся модуль счётчика cntr с параметрами «MODULE», равным 8, и «STEP», равным 1. К порту «clk» подключен «clk», к порту «reset» - подключен «RESET», на порт «enable» подана 1, на порт «direction» подан 0, к порту «cnt» подключена трёхбитная шина «counter_res»

В блоке «always», работающему по переднему фронту «clk», «SEG», присваивается значение «8'b11111111», если «RESET» равно 1, иначе в блоке «case» по значению «NUMBER_SPLITTER[counter_res]» выбирается значение для выходного порта «SEG», с помощью логического сдвига на значение «counter_res» выбирается значение для регистра «AN_REG».

Реализация модуля представлена в Листинге 1.5. RTL-схема представлена на Рисунке 1.5.

Листинг 1.5 – Реализация модуля управления семисегментными индикаторами

```
module SevenSegmentLED(
    input [7:0] AN_MASK,
    input [31:0] NUMBER,
    input clk,
    input RESET,
    output [7:0] AN,
    output reg[7:0] SEG
);

    wire[2:0] counter_res;

    count #(.mod(8), .step(1)) cntr(
        .clk(clk),
        .RE(RESET),
        .CE(1'b1),
        .dir(1'b0),
        .out(counter_res)
    );

    reg [7:0] AN_REG = 0;
    assign AN = AN_REG | AN_MASK;
    wire [3:0] NUMBER_SPLITTER[0:7];
    genvar i;
    generate
        for (i = 0; i < 8; i = i + 1)
        begin
            assign NUMBER_SPLITTER[i] = NUMBER[((i+1)*4-1)-:4];
        end
    endgenerate
    always @(posedge clk)
        begin
        if (RESET)
            SEG <= 8'b11111111;
        else
        begin
            case (NUMBER_SPLITTER[counter_res])
                4'h0: SEG <= 8'b11000000;
                4'h1: SEG <= 8'b11111001;
```

Продолжение Листинга 1.5

```
                4'h2: SEG <= 8'b10100100;
                4'h3: SEG <= 8'b10110000;
                4'h4: SEG <= 8'b10011001;
                4'h5: SEG <= 8'b10010010;
                4'h6: SEG <= 8'b10000010;
                4'h7: SEG <= 8'b11111000;
                4'h8: SEG <= 8'b10000000;
                4'h9: SEG <= 8'b10010000;
                4'ha: SEG <= 8'b10001000;
                4'hb: SEG <= 8'b10000011;
                4'hc: SEG <= 8'b11000110;
                4'hd: SEG <= 8'b10100001;
                4'he: SEG <= 8'b10000110;
                4'hf: SEG <= 8'b10001110;
                default: SEG <= 8'b11111111;
            endcase
            AN_REG = ~(8'b1 << counter_res);
        end
    end
endmodule
```

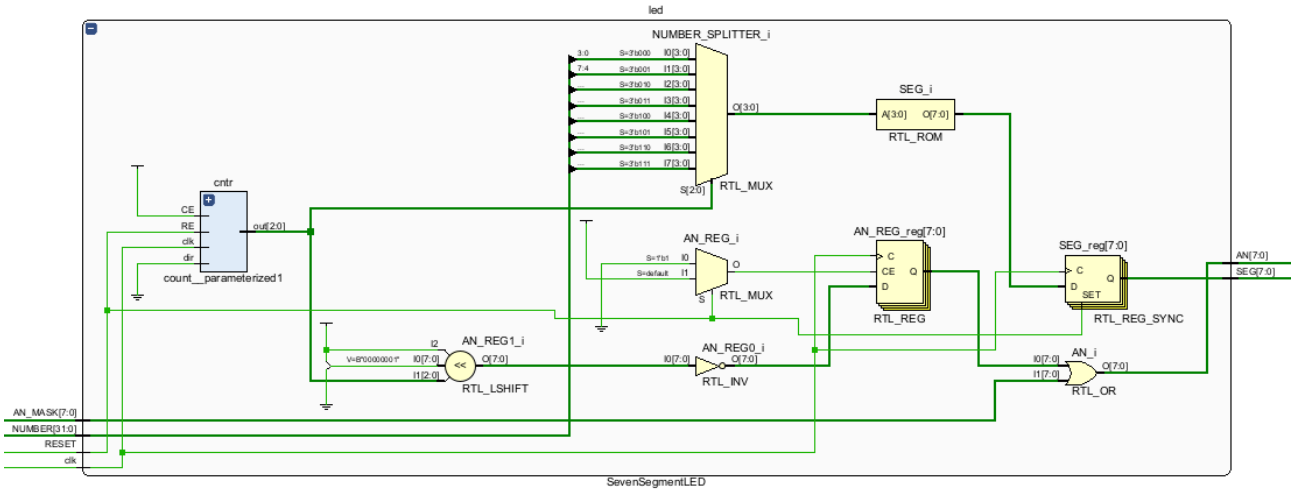

Рисунок 1.5 - RTL-схема модуля управления семисегментными индикаторами

Для реализации данных модулей использовался отдельный модуль, реализующий D–триггер, представленный в Листинге 1.6.

Листинг 1.6 – Реализация модуля D–триггера

```
`timescale 1ns / 1ps
module dtrigger(
    input wire C,
    input wire D,
    input wire en,
    output reg q);
    initial begin
        q<=0;
    end

    always @(posedge C) begin
        if(en) begin
            q <= D;
        end
    end
endmodule
```

1.3 Создание модуля верхнего уровня

Модуль верхнего уровня имеет название «main». Он обладает следующими портами: четырехбитный входной порт «SWITHCES» - цифра, которую необходимо отобразить на дисплее, входной порт «button_in» - кнопка для разрешения записи, синхросигнал «clk», входной порт «button_reset_in» для сброса значений маски и регистра, выходной порт «an» – шина разрешающих входов анодов для всех индикаторов, «seg» - шина катодов для одного индикатора.

Объявляется регистр «AN_MASK» (маска, использующаяся для отключения части семисегментных индикаторов) и инициализируется значением «8'b11111111». Объявляется регистр «NUMBER» (используется для хранения введенных значений) и инициализируется значением 0. Объявляется цепь «clk_div» для вывода результата работы делителя частоты. Объявляются 4 цепи «button_signal», «button_signal_en», «reset_signal_en» и «reset_signal» для вывода результатов работы первого и второго фильтра дребезга контактов соответственно.

Создается экземпляр модуля «debouncer» с названием «dbnc», в единственный параметр которого передается 128. К портам «clk», «in_signal», «CLOCK_ENABLE», «out_signal» и «out_signal_enable» подключаются «clk», «button_in», «1'b1», «button_signal» и «button_signal_en» соответственно. Также создается экземпляр модуля «debouncer» с названием «dbnc_reset», в единственный параметр которого передается 128. К портам «clk», «in_signal», «CLOCK_ENABLE», «out_signal» и «out_signal_enable» подключаются «clk», «button_reset_in», «1'b1», «reset_signal» и «reset_signal_en» соответственно.

Создается экземпляр модуля «clk_div» с названием «div», в единственный параметр которого передается 1024. К портам «clk» и «clk_div» подключаются «clk» и «clk_div» соответственно.

Создается экземпляр модуля «SevenSegmentLED» с названием «led». К портам «AN_MASK», «NUMBER», «RESET», «clk», «AN» и «SEG»

подключаются «mask», «number», «reset_signal», «clk_div», «AN» и «SEG» соответственно.

В блоке «always» обновляется последняя цифра регистра, регистрам «AN_MASK» и «NUMBER» присваиваются их начальные значения, если на цепи «reset_signal» значение 1. Если на цепи «button_signal_en» значение 1, регистр «AN_MASK» сдвигается на 1 бит влево, а «NUMBER» сдвигается на 4 бита влево. Блок работает по переднему фронту синхросигнала.

Код модуля верхнего уровня представлен в Листинге 1.6, а его RTL-схема представлена на Рисунке 1.6.

Листинг 1.6 – Реализация модуля контроллера

```
`timescale 1ns / 1ps
module main(
    input [3:0] SWITCHES,
    input button_in, clk, button_reset_in,
    output [7:0] AN,
    output [6:0] SEG,
    output reg [31:0] NUMBER,
    output wire clk_div
    // output reg [3:0] LEDS
);
    wire button_signal, button_signal_en, reset_signal_en, reset_signal;
    reg[7:0] AN_MASK = 8'b11111111;
    initial begin
        NUMBER <= 0;
    end
    filtercon #(128) dbnc(
        .clk(clk),
        .in_signal(button_in),
        .clock_enable(1'b1),
        .out_signal(button_signal),
        .out_signal_enable(button_signal_en)
    );

    filtercon #(128) dbnc_reset(
        .clk(clk),
        .in_signal(button_reset_in),
        .clock_enable(1'b1),
        .out_signal(reset_signal),
        .out_signal_enable(reset_signal_en)
    );

    clk_divider #(1024) div(
        .clk(clk),
        .clk_div(clk_div)
    );
```

Продолжение Листинга 1.6

```
    SevenSegmentLED led(
        .AN_MASK(AN_MASK),
        .NUMBER(NUMBER),
        .clk(clk_div),
        .RESET(reset_signal),
        .AN(AN),
        .SEG(SEG)
    );

    always@(posedge clk)
    begin
        if (reset_signal)
        begin
            NUMBER <= 0;
            AN_MASK <= 8'b11111111;
        end
        else if (button_signal_en)
            begin
                NUMBER <= {NUMBER[27:0], SWITCHES};
                AN_MASK <= {AN_MASK[6:0], 1'b0};
            end
    end
endmodule
```

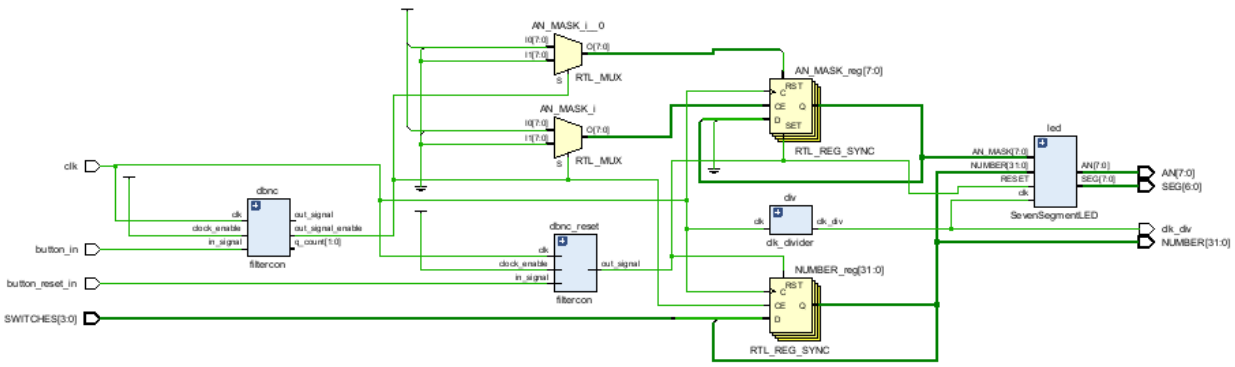

Рисунок 1.6 - RTL-схема модуля верхнего уровня

2 СОЗДАНИЕ ТЕСТОВОГО МОДУЛЯ И ЕГО ВЕРИФИКАЦИЯ

Название тестового модуля – «testbench». Объявляется четырехбитный регистр «SWITCHES» (ввод цифры с помощью движковых переключателей) и инициализируется значением 0. Объявляются две восьмибитные цепи «SEG» и «AN». Также объявляются регистры «clk», «button» и «button_reset» и они инициализируются нулем.

Создается экземпляр модуля верхнего уровня «Controller» с названием «cntlr». К портам «SWITCHES», «button_in», «clk», «button_reset_in», «AN» и «SEG» подключаются «SWITCHES», «button», «clk», «button_reset», «AN» и «SEG» соответственно.

В блоке «always» каждые 5 наносекунд регистр «clk» меняет свое значение на противоположное.

В блоке «initial» симулируется работа с платой: два нажатия на кнопку, чтобы разрешить записи двух цифр. В конце работы происходит нажатие на кнопку, отвечающую за восстановление изначальных значений. Чтобы сымитировать дребезг контактов, используется генерация псевдослучайных чисел.

Реализация тестового модуля представлена в Листинге 2.1. На Рисунке 2.1 представлена верификация тестового модуля.

Листинг 2.1 – Реализация тестового модуля

```
`timescale 1ns / 1ps
module testbench;
reg[3:0] SWITCHES = 0;
reg clk = 0;
reg button = 0;
reg button_reset = 0;
wire[7:0] AN;
wire[6:0] SEG;
wire[31:0] NUMBER;
wire clk_div;
main cntlr(
 .SWITCHES(SWITCHES),
 .button_in(button),
 .clk(clk),
 .button_reset_in(button_reset),
 .AN(AN),
```

Продолжение Листинга 2.1

```
.NUMBER(NUMBER),
 .SEG(SEG),
 .clk_div(clk_div)
 );
always #5 clk = ~clk;
initial
begin
 #4000
 $srandom(35000);
 SWITCHES = 4'b1111;
 repeat($urandom_range(50,0))
 begin
 button = $random;
 #3;
 end
 button = 1;
 #1000;

 repeat($urandom_range(50,0))
 begin
 button = $random;
 #3;
 end
 button = 0;
 #8000;

 SWITCHES = 4'b0110;
 repeat($urandom_range(50,0))
 begin
 button = $random;
 #3;
 end
 button = 1;
 #1000;

 repeat($urandom_range(50,0))
 begin
 button = $random;
 #3;
 end
 button = 0;
 #8000;
 repeat($urandom_range(50,0))
 begin
 button_reset = $random;
 #3;
 end
 button_reset = 1;
 #800;

 repeat($urandom_range(50,0))
 begin
 button_reset = $random;
 #3;
 end
 button_reset = 0;
end
endmodule
```

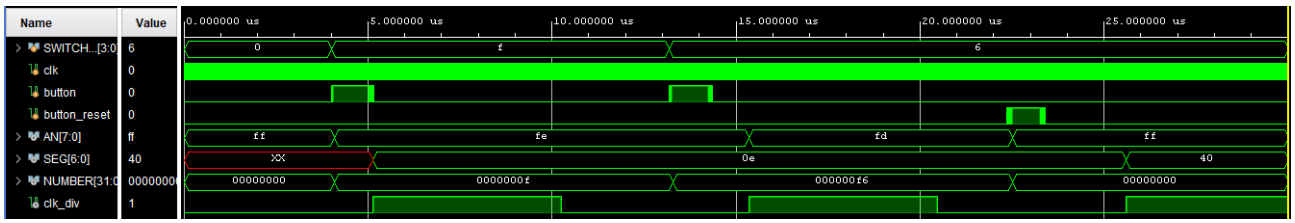

Рисунок 2.1 – Результат верификации тестового модуля.

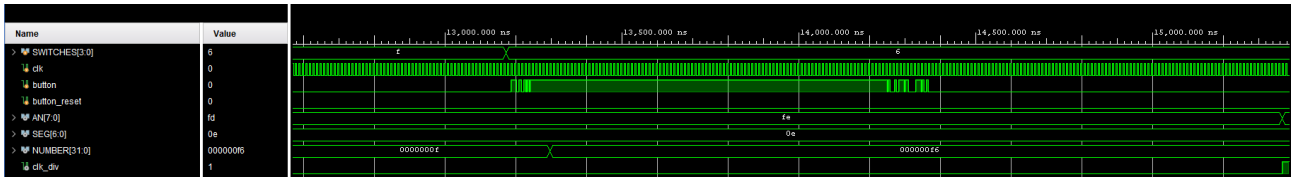

Рисунок 2.2 – Фильтрация дребезга.

3 СОЗДАНИЕ ФАЙЛА ПРОЕКТНЫХ ОГРАНИЧЕНИЙ И ВЕРИФИКАЦИЯ НА ПЛАТЕ

Содержание файла проектных ограничений представлено в Листинге 3.1.

Листинг 3.1 – Реализация проектных ограничений

```
create_clock -add -name sys_clk -period 10.00 -waveform {0 5} [ get_ports
{clk}]

set_property IOSTANDARD LVCMOS33 [get_ports {clk}]
set_property PACKAGE_PIN E3 [get_ports {clk}]

#кнопки

set_property -dict {PACKAGE_PIN N17 IOSTANDARD LVCMOS33} [get_ports
{button_in}]
set_property -dict {PACKAGE_PIN C12 IOSTANDARD LVCMOS33} [get_ports
{button_reset_in}]

#свичи

set_property -dict {PACKAGE_PIN J15 IOSTANDARD LVCMOS33} [get_ports
{SWITCHES[0]}]
set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33} [get_ports
{SWITCHES[1]}]
set_property -dict {PACKAGE_PIN M13 IOSTANDARD LVCMOS33} [get_ports
{SWITCHES[2]}]
set_property -dict {PACKAGE_PIN R15 IOSTANDARD LVCMOS33} [get_ports
{SWITCHES[3]}]

#аноды
set_property IOSTANDARD LVCMOS33 [ get_ports { AN[0] } ]
set_property PACKAGE_PIN J17 [ get_ports { AN[0] } ]
set_property IOSTANDARD LVCMOS33 [ get_ports { AN[1] } ]
set_property PACKAGE_PIN J18 [ get_ports { AN[1] } ]
set_property IOSTANDARD LVCMOS33 [ get_ports { AN[2] } ]
set_property PACKAGE_PIN T9 [ get_ports { AN[2] } ]
set_property IOSTANDARD LVCMOS33 [ get_ports { AN[3] } ]
set_property PACKAGE_PIN J14 [ get_ports { AN[3] } ]
set_property IOSTANDARD LVCMOS33 [ get_ports { AN[4] } ]
set_property PACKAGE_PIN P14 [ get_ports { AN[4] } ]
set_property IOSTANDARD LVCMOS33 [ get_ports { AN[5] } ]
set_property PACKAGE_PIN T14 [ get_ports { AN[5] } ]
set_property IOSTANDARD LVCMOS33 [ get_ports { AN[6] } ]
set_property PACKAGE_PIN K2 [ get_ports { AN[6] } ]
set_property IOSTANDARD LVCMOS33 [ get_ports { AN[7] } ]
set_property PACKAGE_PIN U13 [ get_ports { AN[7] } ]

#катоды
set_property IOSTANDARD LVCMOS33 [ get_ports { SEG[0] } ]
set_property PACKAGE_PIN T10 [ get_ports { SEG[0] } ]
set_property IOSTANDARD LVCMOS33 [ get_ports { SEG[1] } ]
set_property PACKAGE_PIN R10 [ get_ports { SEG[1] } ]
set_property IOSTANDARD LVCMOS33 [ get_ports { SEG[2] } ]
set_property PACKAGE_PIN K16 [ get_ports { SEG[2] } ]
set_property IOSTANDARD LVCMOS33 [ get_ports { SEG[3] } ]
set_property PACKAGE_PIN K13 [ get_ports { SEG[3] } ]
set_property IOSTANDARD LVCMOS33 [ get_ports { SEG[4] } ]
set_property PACKAGE_PIN P15 [ get_ports { SEG[4] } ]
```

Продолжение Листинга 3.1

```
set_property IOSTANDARD LVCMOS33 [ get_ports { SEG[5] } ]
set_property PACKAGE_PIN T11 [ get_ports { SEG[5] } ]
set_property IOSTANDARD LVCMOS33 [ get_ports { SEG[6] }
```

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы приобретён навык построения управляющего устройства для визуального отображения информации при помощи набора семисегментных индикаторов, изучены основные особенности считывания сигнала с физического устройства ввода — кнопки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания по ЛР № 1 — URL: https://online-edu.mirea.ru/mod/resource/view.php?id=413209 (Дата обращения: 18.02.2024).

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. Рабан, Жан.М., Чандракасан, А., Николич, Б. Цифровые интегральные схемы. Методология проектирования. 2-е изд.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2016. – 912 с.: ил. – Паралтит. англ. ISBN 978-5-8459- 1116-2 (рус.).

5. Шафер Д., Фатрелл Р., Шафер Л. Управление программными проектами: достижение оптимального качества при минимуме затрат: Пер. с англ. – М.: Издательский дом «Вильямс», 2004. – 1136 с.: ил. – Парал.тит.англ.