|  |
| --- |
|  |
| 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 № 1**

«Управление семисегментными индикаторами»

по дисциплине

«Схемотехника устройств компьютерных систем»

|  |  |
| --- | --- |
| Выполнил студент группы  ИВБО-06-22 | Визитиу Д.В. |
| Принял ассистент кафедры ВТ | Люлява Д.В. |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_2024 г. |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2024 г. |

Москва 2024

# 

# АННОТАЦИЯ

//Данная работа включает в себя 5 рисунков и 5 листингов. Количество страниц в работе — 13.

# СОДЕРЖАНИЕ

[Федеральное государственное бюджетное образовательное учреждение 1](#_Toc159422868)

[высшего образования 1](#_Toc159422869)

[**«МИРЭА - Российский технологический университет»** 1](#_Toc159422870)

[**РТУ МИРЭА** 1](#_Toc159422871)

[АННОТАЦИЯ 2](#_Toc159422872)

[СОДЕРЖАНИЕ 3](#_Toc159422873)

[ВВЕДЕНИЕ 4](#_Toc159422874)

[1 СОЗДАНИЕ НЕОБХОДИМЫХ МОДУЛЕЙ НА VERILOG HDL 5](#_Toc159422875)

[1.1 Создание модуля управления семисегментными индикаторами 5](#_Toc159422876)

[1.2 Создание модуля верхнего уровня 7](#_Toc159422877)

[2 СОЗДАНИЕ ТЕСТОВОГО МОДУЛЯ И ЕГО ВЕРИФИКАЦИЯ 12](#_Toc159422878)

[3 СОЗДАНИЕ ФАЙЛА ПРОЕКТНЫХ ОГРАНИЧЕНИЙ 15](#_Toc159422879)

[ЗАКЛЮЧЕНИЕ 16](#_Toc159422880)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc159422881)

# ВВЕДЕНИЕ

В данной лабораторной работе рассматриваются вопросы индикации при помощи семисегментных индикаторов, объединённых в дисплей в составе отладочной платы серии Xilinx Nexys. Разрабатываемое устройство представляет собой устройство хранения истории ввода. 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.

# СОЗДАНИЕ НЕОБХОДИМЫХ МОДУЛЕЙ НА VERILOG HDL

## Создание модуля управления семисегментными индикаторами

Название модуля – «seven\_segment\_led». Модуль имеет следующие порты: входной порт «AN\_MASK» – маска для отключения части семисегментных индикаторов, входной порт «NUMBER» - значение, которое необходимо вывести на дисплей, входной порт «RESET» - сброс счетчика, входной порт «clk» – синхросигнал, выходной порт «AN» – шина разрешающих входов анодов для всех индикаторов, «SEG» - шина катодов для одного индикатора.

Объявляются и инициализируются нулем восьмибитный регистр «AN\_REG» и трехбитный регистр «digit\_pos». 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».

В блоке «always» регистру «digit\_pos» присваивается значение 0, если значение входного порта «RESET» равно 1, иначе значение «digit\_pos» увеличивается на 1. Данный блок выполняется всегда, когда значение синхросигнала и входного порта «RESET» меняется с 0 на 1. В следующем блоке «always» в блоке «case» по значению «NUMBER\_SPLITTER[digit\_pos]» выбирается значение для выходного порта «SEG», с помощью логического сдвига на значение «digit\_pos» выбирается значение для регистра «AN\_REG». Блок «always» выполняется всегда, когда «digit\_pos» меняет свое значение.

Код модуля синхронизатора представлен в Листинге 1.1, а его RTL-схема представлена на Рисунке 1.1.

*Листинг 1.1 – Реализация модуля управления семисегментными индикаторами*

module seven\_segment\_led(

input [7:0] AN\_MASK,

input [31:0] NUMBER,

input RESET,

input clk,

output [7:0] AN,

output reg [7:0] SEG);

reg [7:0] AN\_REG = 0;

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

reg [2:0] digit\_pos = 0;

genvar i;

generate

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

begin

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

end

endgenerate

assign AN = AN\_REG | AN\_MASK;

always@(posedge clk or posedge RESET)

if (RESET) digit\_pos = 0;

else digit\_pos = digit\_pos + 1;

always@(digit\_pos)

begin

case(NUMBER\_SPLITTER[digit\_pos])

4'h0: SEG <= 8'b11000000;

4'h1: SEG <= 8'b11111001;

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'd1 << digit\_pos);

End

endmodule

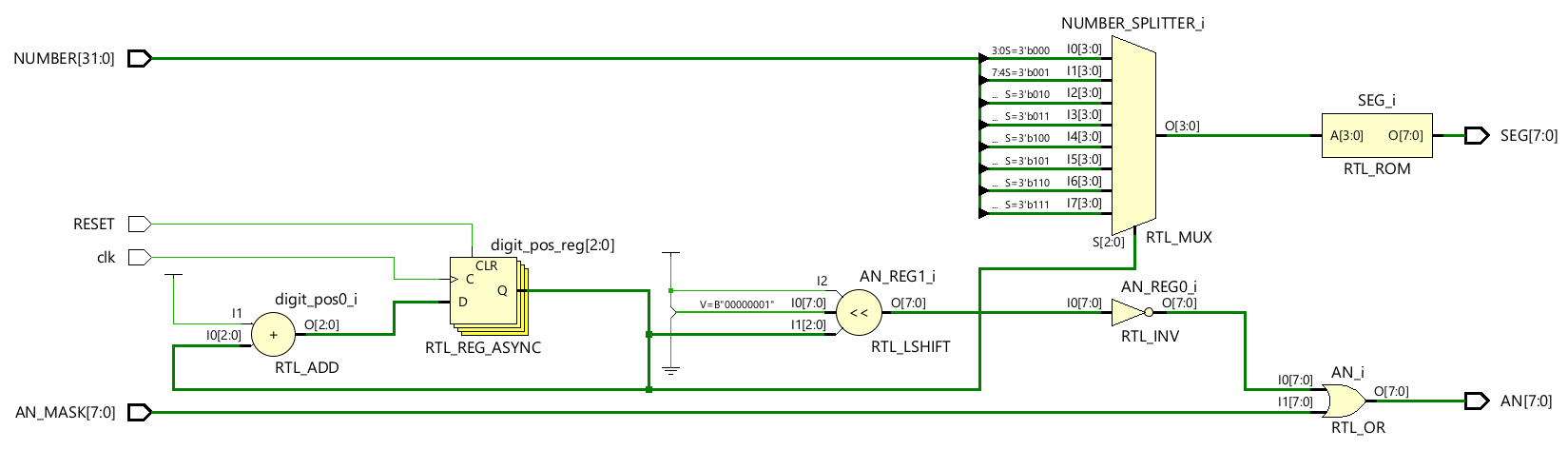


Рисунок 1.1 — RTL-схема модуля управления семисегментными индикаторами

## Создание модуля верхнего уровня

Перед созданием модуля верхнего уровня необходимо подключить некоторые дополнительные модули, а именно модули, описывающие синхронизатор, счетчик, делитель частоты и фильтр дребезга контактов.

Модуль, описывающий синхронизатор, используется в фильтре дребезга. Код модуля представлен в Листинге 1.2, а его RTL-схема представлена на Рисунке 1.2.

*Листинг 1.2 – Реализация модуля, описывающего синхронизатор*

module synchronizer(

input in, clk,

output out);

reg a, b;

always@(posedge clk)

begin

b <= a;

a <= in;

end

assign out = b;

endmodule

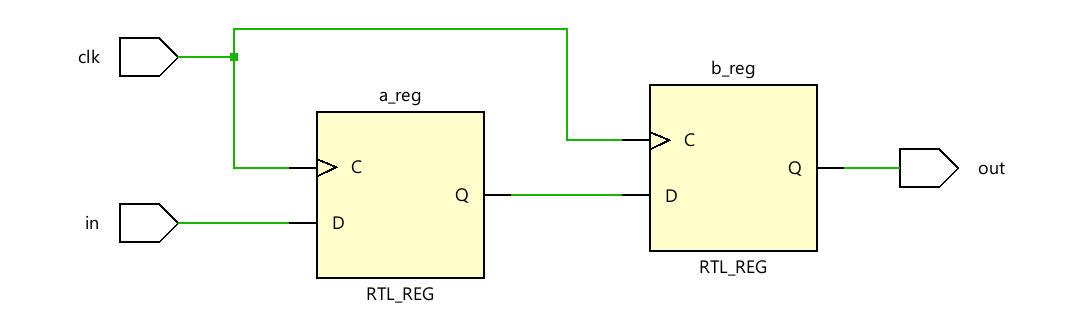


Рисунок 1.2 — RTL-схема модуля, описывающего синхронизатор

Для реализации делителя частоты и фильтра дребезга нужен счетчик. Код модуля, описывающего параметрический универсальный реверсивный счетчик, представлен в Листинге 1.3, а его RTL-схема представлена на Рисунке 1.3.

*Листинг 1.3 – Реализация модуля, описывающего параметрический универсальный реверсивный счетчик*

module counter #(MODULE = 4, STEP = 1) (

input clk,

input reset,

input enable,

input dir,

output reg [$clog2(MODULE)-1:0] cnt);

initial cnt = 0;

always@(posedge clk)

begin

if (enable)

begin

if (reset)

cnt = 0;

else if (dir)

cnt = (cnt + STEP) % MODULE;

else

cnt = (MODULE + (cnt - STEP)) % MODULE;

end

end

endmodule

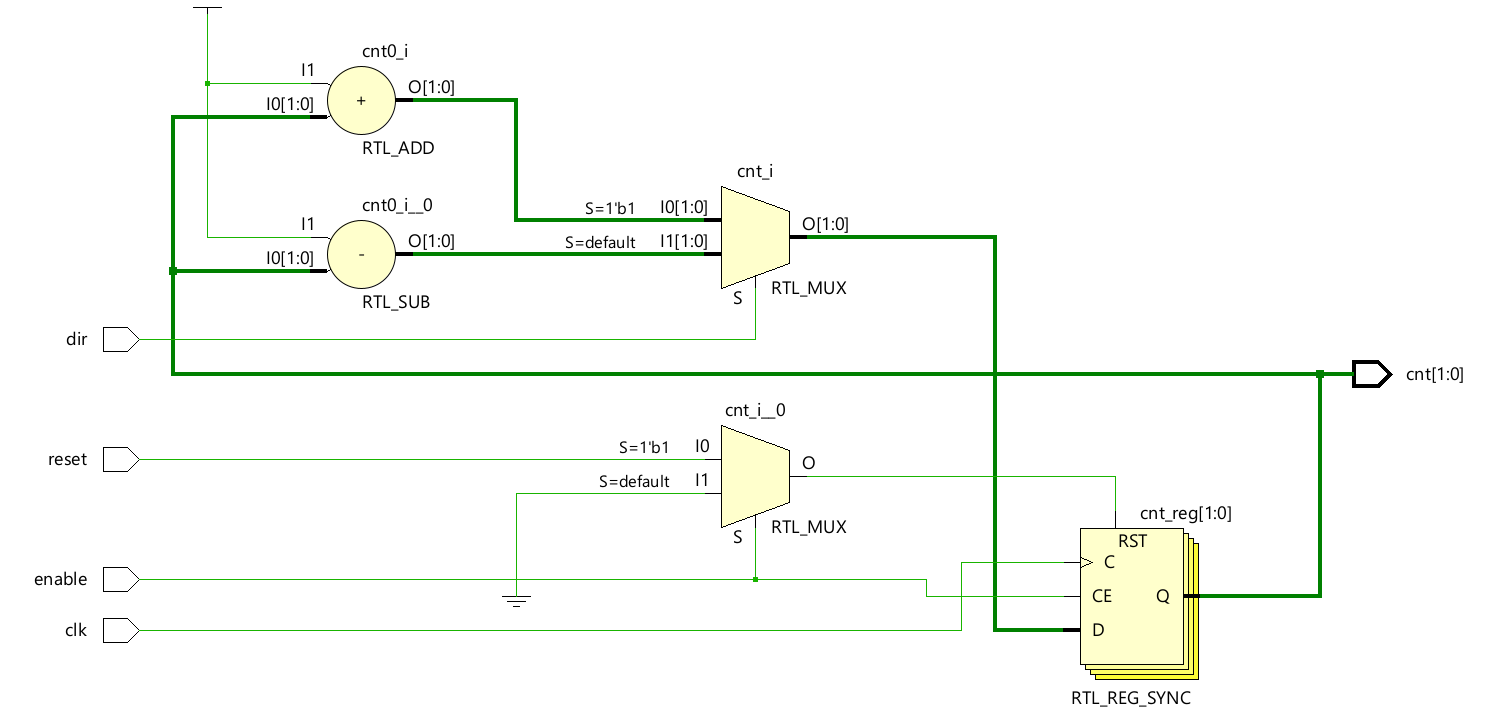


Рисунок 1.3 — RTL-схема модуля, описывающего параметрический универсальный реверсивный счетчик

Делитель частоты должен изменять частоту синхросигнала, чтобы информация на дисплее была хорошо видна человеческим глазом. Код модуля делителя частоты представлен в Листинге 1.4, а его RTL-схема представлена на Рисунке 1.4.

*Листинг 1.4 – Реализация модуля, описывающего делитель частоты*

module clk\_div #(DIV = 2) (

input clk,

output reg clk\_d);

wire [$clog2(DIV)-1:0] cnt;

counter #(.STEP(1), .MODULE(DIV)) cntr(

.clk(clk),

.reset(1'd0),

.dir(1'd1),

.cnt(cnt));

initial clk\_d = clk;

always@(clk)

if (cnt == 0)

clk\_d = ~clk\_d;

endmodule

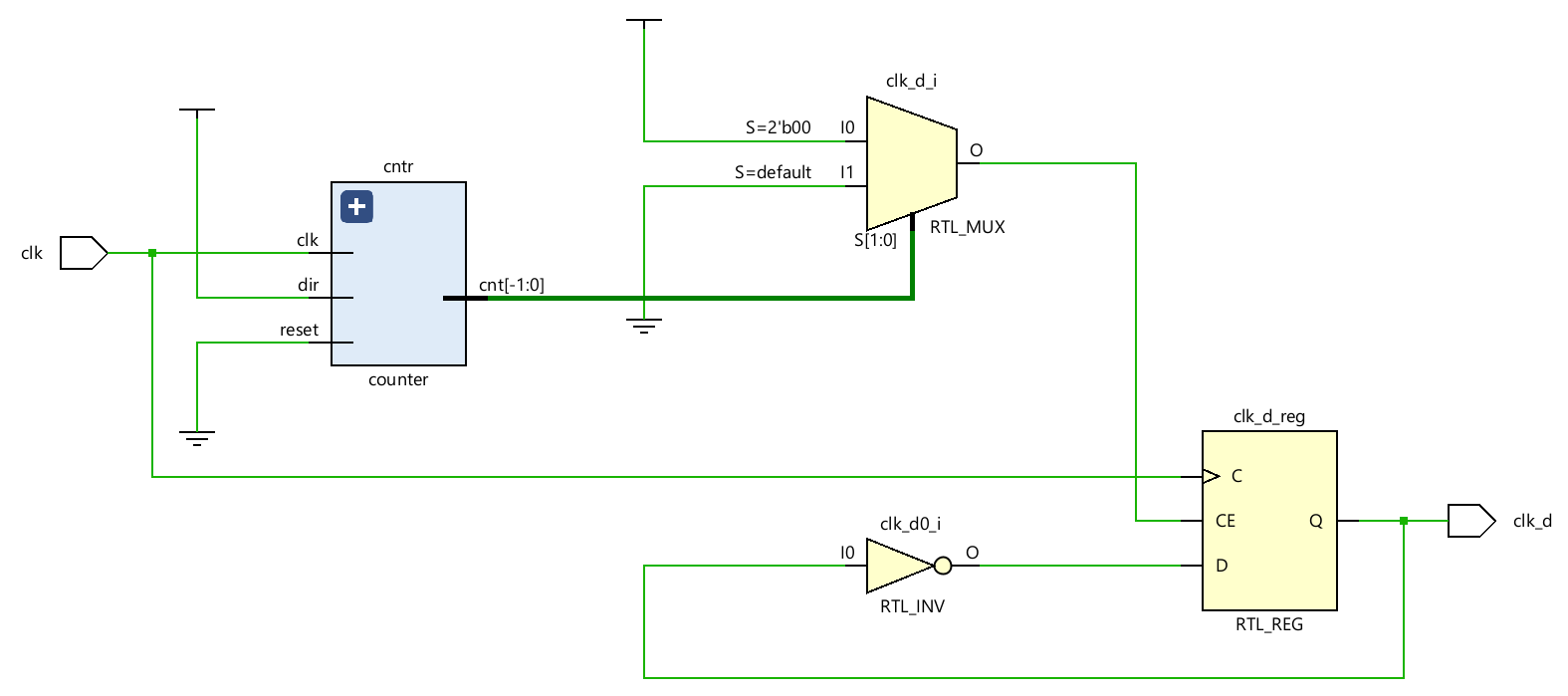


Рисунок 1.4 — RTL-схема модуля, описывающего делитель частоты

Так как разрешение записи осуществляется с помощью кнопки, необходимо использовать фильтр дребезга контактов. Код модуля, описывающего фильтр дребезга, представлен в Листинге 1.5, а его RTL-схема представлена на Рисунке 1.5.

*Листинг 1.5 – Реализация модуля, описывающего параметризованный фильтр дребезга контактов*

module debouncer #(MODULE = 8)(

input clk,

input in\_signal,

input CLOCK\_ENABLE,

output reg out\_signal,

output reg out\_signal\_enable);

wire sync\_signal;

*Продолжение Листинга 1.5*

wire [$clog2(MODULE)-1:0] counter\_res;

synchronizer sinc(.in(in\_signal), .clk(clk), .out(sync\_signal));

counter #(.MODULE(MODULE), .STEP(1)) cntr(

.clk(clk),

.reset(sync\_signal~^out\_signal),

.enable(CLOCK\_ENABLE),

.dir(1'b1),

.cnt(counter\_res));

always@(posedge clk)

if (& (counter\_res) & CLOCK\_ENABLE)

out\_signal <= sync\_signal;

always@(posedge clk)

out\_signal\_enable <= & (counter\_res) & sync\_signal & CLOCK\_ENABLE;

endmodule

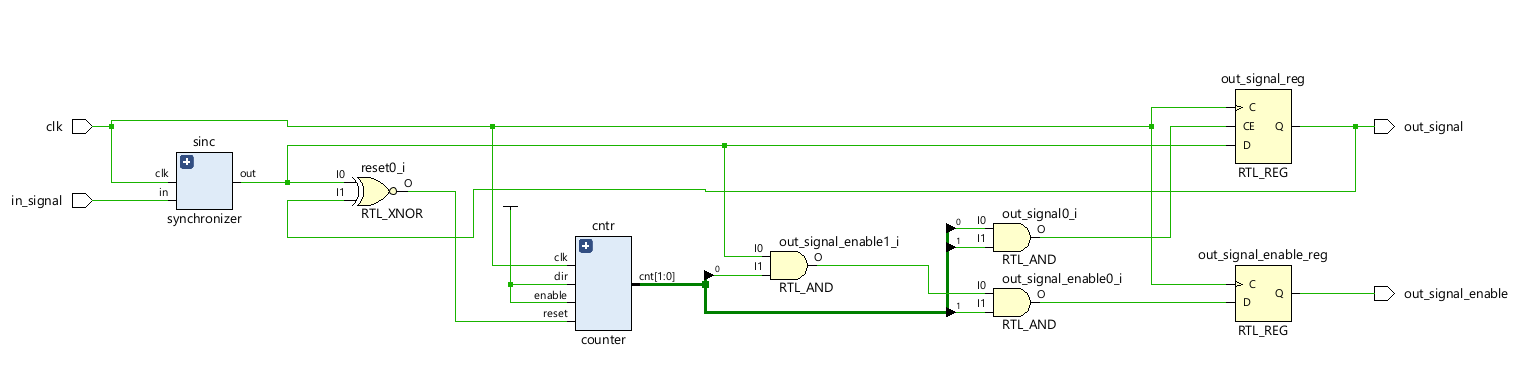


Рисунок 1.5 — RTL-схема модуля, описывающего параметризованный фильтр дребезга контактов

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

Объявляется регистр «mask» (маска, использующаяся для отключения части семисегментных индикаторов) и инициализируется значением «8'b11111111». Объявляется регистр «number» (используется для сохранения введенных значений) и инициализируется значением 0. Объявляется цепь «led\_clk» для вывода результата работы делителя частоты. Объявляются две цепи «deb\_en» и «deb\_reset» для вывода результатов работы первого и второго фильтра дребезга контактов соответственно.

Создается экземпляр модуля «seven\_segment\_led» с названием «led». К портам «AN\_MASK», «NUMBER», «RESET», «clk», «AN» и «SEG» подключаются «mask», «number», «1'b0», «led\_clk», «an» и «seg» соответственно.

Создается экземпляр модуля «debouncer» с названием «deb1». К портам «clk», «in\_signal» и «out\_signal\_enable» подключаются «clk», «en\_button» и «deb\_en» соответственно. Также создается экземпляр модуля «debouncer» с названием «deb2». К портам «clk», «in\_signal» и «out\_signal\_enable» подключаются «clk», «en\_button» и «deb\_reset» соответственно.

Создается экземпляр модуля «clk\_div» с названием «div», в единственный параметр которого передается 4. К портам «clk» и «clk\_d» подключаются «clk» » и «led\_clk» соответственно.

В первом блоке «always» регистрам «mask» и «number» присваиваются их начальные значения, если на цепи «deb\_reset» значение 1. Во втором блоке «always» регистр «mask» сдвигается на 1 влево, а «number» сдвигается на 4 влево и в конец записывается значение входного порта «digit», если на цепи «deb\_en» значение 1. Эти блоки работают по переднему фронту синхросигнала.

Код модуля, описывающего фильтр дребезга, представлен в Листинге 1.6, а его RTL-схема представлена на Рисунке 1.6.

*Листинг 1.6 – Реализация модуля верхнего уровня*

module main(

input [3:0] digit,

input en\_button,

input clk,

input reset,

output [7:0] an,

output [7:0] seg);

reg [7:0] mask = 8'b11111111;

reg [31:0] number = 0;

wire led\_clk;

wire deb\_en, deb\_reset;

seven\_segment\_led led(

.AN\_MASK(mask),

.NUMBER(number),

.RESET(1'b0),

.clk(led\_clk),

.AN(an),

.SEG(seg));

*Продолжение Листинга 1.6*

debouncer #(128) deb1(

.clk(clk),

.in\_signal(en\_button),

.CLOCK\_ENABLE(1'b1),

.out\_signal(),

.out\_signal\_enable(deb\_en));

debouncer #(128) deb2(

.clk(clk),

.in\_signal(reset),

.CLOCK\_ENABLE(1'b1),

.out\_signal(),

.out\_signal\_enable(deb\_reset));

clk\_div #(1024) div(

.clk(clk),

.clk\_d(led\_clk));

always@(posedge clk)

begin

if (deb\_reset)

begin

mask = 8'b11111111;

number = 0;

end

if (deb\_en)

begin

mask = mask << 1;

number = {number[27:0], digit};

end

end

endmodule

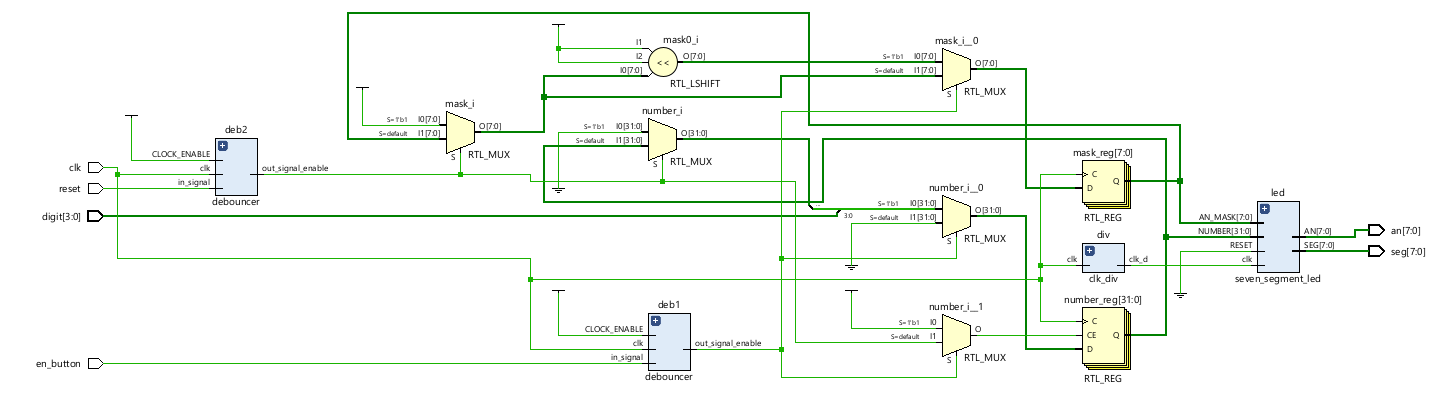


Рисунок 1.6 — RTL-схема модуля верхнего уровня

# СОЗДАНИЕ ТЕСТОВОГО МОДУЛЯ И ЕГО ВЕРИФИКАЦИЯ

Название тестового модуля – «test». Объявляется четырехбитный регистр «inp» (ввод цифры с помощью движковых переключателей) и инициализируется значением «4'b0000». Объявляются две восьмибитные цепи «seg» и «an». Также объявляются регистры «clk», «button» и «reset» и они инициализируются нулем.

Создается экземпляр модуля верхнего уровня «main» с названием «m». К портам «digit», «en\_button», «clk», «reset», «an» и «seg» подключаются «inp», «button», «clk», «reset», «an» и «seg» соответственно.

В блоке «always» каждые 10 наносекунд регистр «clk» меняет свое значение на противоположное.

В блоке «initial» симулируется работа с платой, 4 нажатия на кнопку, чтобы разрешить запись 4 цифр. В конце работы происходит нажатие на кнопку, отвечающую за восстановление изначальных значений. Чтобы сымитировать дребезг контактов, используется генерация псевдослучайных чисел.

Код тестового модуля представлен в Листинге 2.1 [3].

*Листинг 2.1 – Реализация тестового модуля*

`timescale 1ns / 1ps

module test;

reg [3:0] inp = 4'b0000;

reg reset = 0;

wire [7:0] seg;

wire [7:0] an;

reg clk = 0;

reg button = 0;

main m(.digit(inp),

.en\_button(button),

.clk(clk),

.reset(reset),

.an(an),

.seg(seg));

always #5 clk = ~clk;

initial

begin

#100

$srandom(33985);

repeat(4)

begin

inp = inp + 4'b0001;

repeat($urandom\_range(50,0))

begin

button = $random;

#3;

end

button = 1;

*Продолжение Листинга 2.1*

#200;

repeat($urandom\_range(50,0))

begin

button = $random;

#3;

end

button = 0;

#200;

end

#200

repeat($urandom\_range(50,0))

begin

reset = $random;

#3;

end

reset = 1;

#200;

repeat($urandom\_range(50,0))

begin

reset = $random;

#3;

end

reset = 0;

end

endmodule

Для удобства значения параметров «» модуля «» и «» модуля «» изменены на значения «» и «» соответственно.

Временная диаграмма работы симуляции тестового модуля представлена на Рисунке 2.1. Вводятся 4 значения с помощью нажатия на кнопку, в конце происходит нажатие на кнопку «reset» удаления истории ввода. Устройство выводит верные значения для отображения введенных значений на индикаторе, история ввода сохраняется в регистре. При следующем нажатии на кнопку светодиод гаснет.

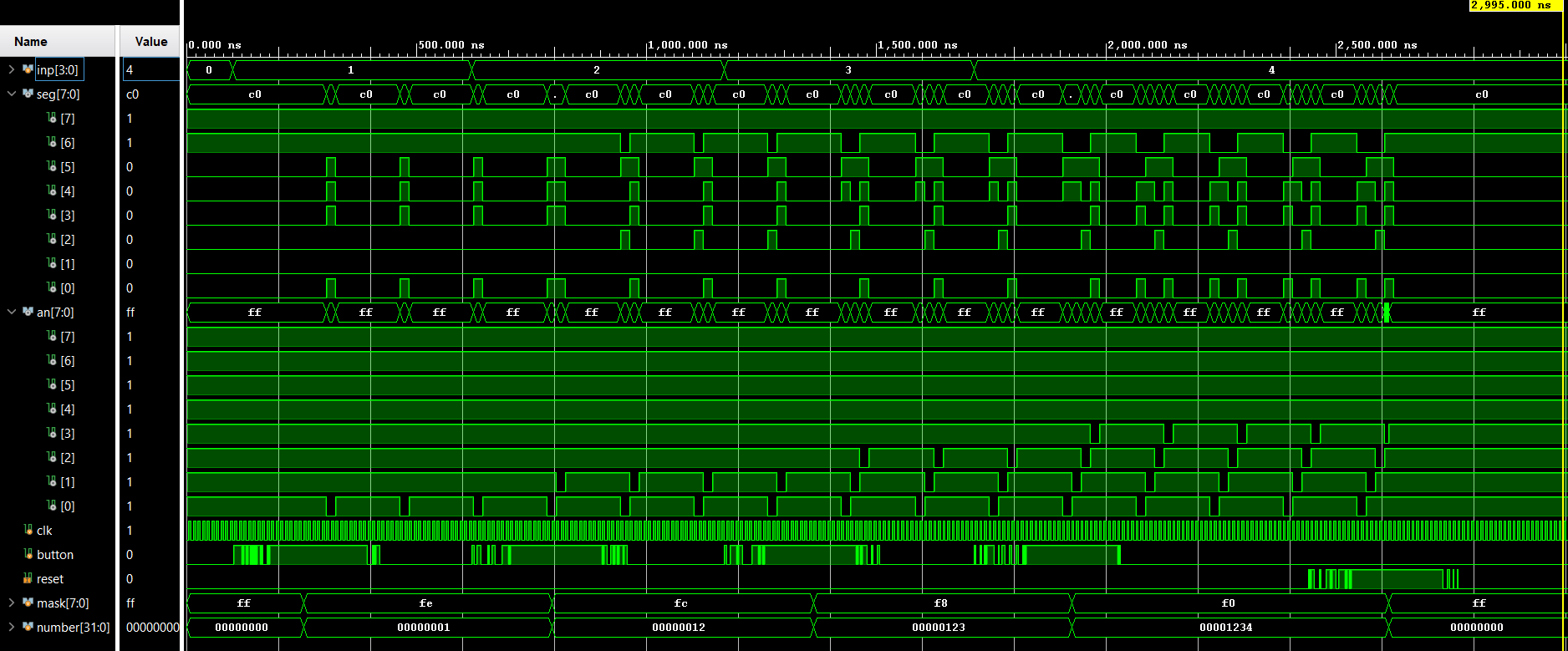


Рисунок 2.1 — Верификация тестового модуля

# СОЗДАНИЕ ФАЙЛА ПРОЕКТНЫХ ОГРАНИЧЕНИЙ

Содержание файла проектных ограничений представлено в Листинге 3.1 [3].

*Листинг 3.1 – Файл проектных ограничений*

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

set\_property IOSTANDARD LVCMOS33 [get\_ports {clk}]

set\_property PACKAGE\_PIN E3 [get\_ports {clk}]

set\_property PACKAGE\_PIN J15 [get\_ports {in[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {in[0]}]

set\_property PACKAGE\_PIN L16 [get\_ports {in[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {in[1]}]

set\_property PACKAGE\_PIN M13 [get\_ports {in[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {in[2]}]

set\_property PACKAGE\_PIN V10 [get\_ports {in[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {in[3]}]

set\_property PACKAGE\_PIN N17 [get\_ports {button}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {button}]

set\_property PACKAGE\_PIN M17 [get\_ports {reset}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {reset}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN J17} [get\_ports {an[0]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN J18} [get\_ports {an[1]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN T9} [get\_ports {an[2]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN J14} [get\_ports {an[3]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN P14} [get\_ports {an[4]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN T14} [get\_ports {an[5]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN K2} [get\_ports {an[6]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN U13} [get\_ports {an[7]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN T10} [get\_ports {seg[0]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN R10} [get\_ports {seg[1]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN K16} [get\_ports {seg[2]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN K13} [get\_ports {seg[3]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN P15} [get\_ports {seg[4]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN T11} [get\_ports {seg[5]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN L18} [get\_ports {seg[6]}]

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN H15} [get\_ports {seg[7]}]

# ЗАКЛЮЧЕНИЕ

//В ходе работы реализован параметризованный фильтр дребезга контактов, с помощью которого создан модуль для управления работой светодиода при помощи кнопки на Verilog HDL, созданы RTL-схемы всех модулей, создан тестовый модуль и произведена верификация тестового модуля.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. 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).

2. Методические указания по ПР № 8 — URL: [https://online-edu.mirea.ru/mod/resource/view.php?id=413209](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fonline-edu.mirea.ru%2Fmod%2Fresource%2Fview.php%3Fid%3D413209) (Дата обращения: 09.12.2023).

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