**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра Вычислительной Техники**

**ОТЧЕТ**

**по лабораторной работе №3**

**по дисциплине «Верификация и тестирование встраиваемых систем»**

**ТЕМА: Модульное и иерархическое проектирование**

|  |  |  |  |
| --- | --- | --- | --- |
| Студенты гр. 6309 | |  | Васин А. М. |
| Преподаватель | |  | Мурсаев А. Х. |
|  |
|  |  |  |  |

Санкт-Петербург

2021

# Цель работы

На первом этапе необходимо разработать описание устройства в едином модуле с генератором тестового воздействия на базе синтаксиса языка VerilogHDL. На втором этапе необходимо разработать testbench универсального типа с явно выделенными модулями тестируемого блока, генератора теста и блока контроля результата.

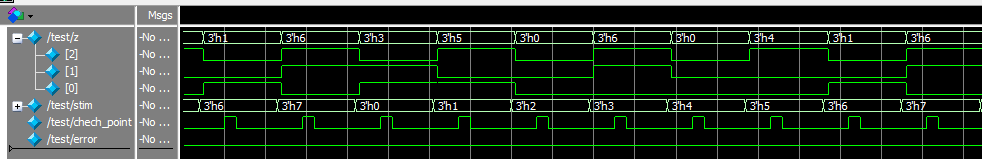
Функции, которые необходимо реализовать представлены ниже в табличной форме.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x2 | x1 | x0 | F1 | F2 | F3 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |

# Ход работы

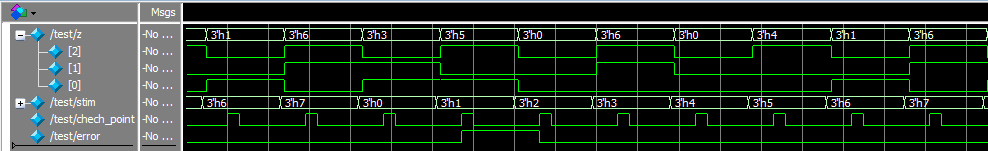
Часть 1

1. Создали новый проект, добавили необходимые файлы и изучили содержимое.
2. Скорректировали файлы так, чтобы воспроизводились три логические функции в соответствии с условием. Провели симуляцию и запуск проекта, результаты на рисунках ниже.



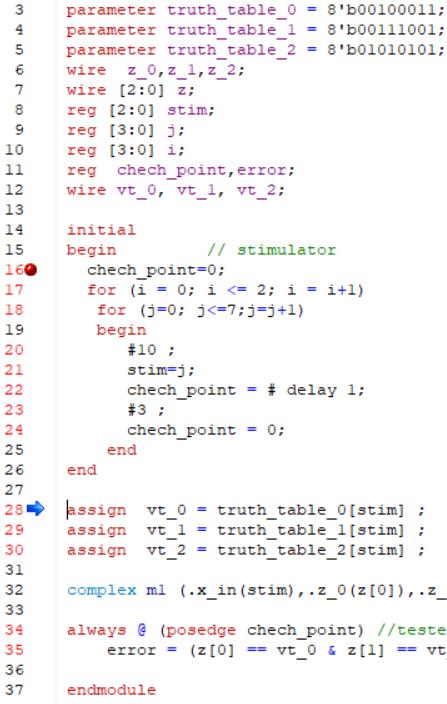
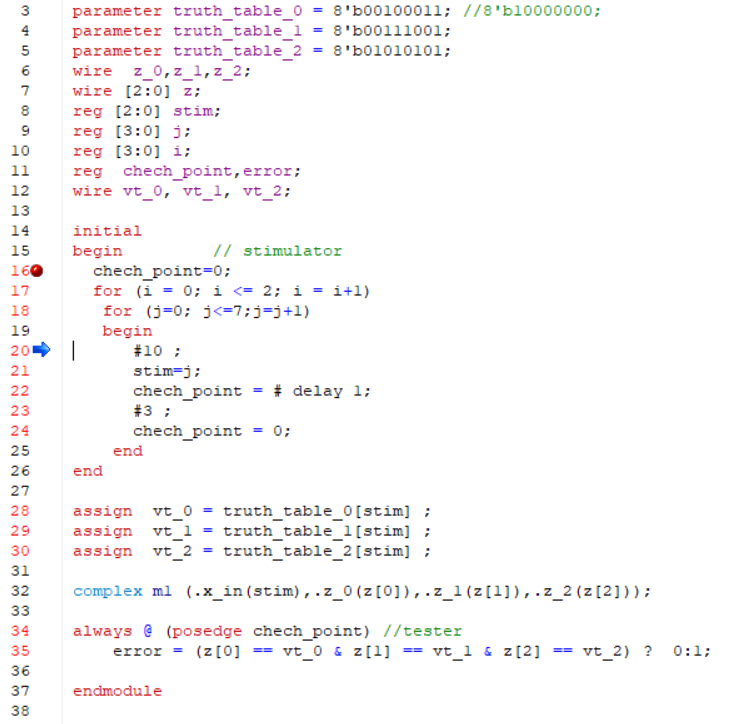
Как мы видим, функции вычисляются верно, ошибок не возникает.

1. Изменили одну из тестовых последовательностей для того, чтобы проверить правильность работы оператора tester.

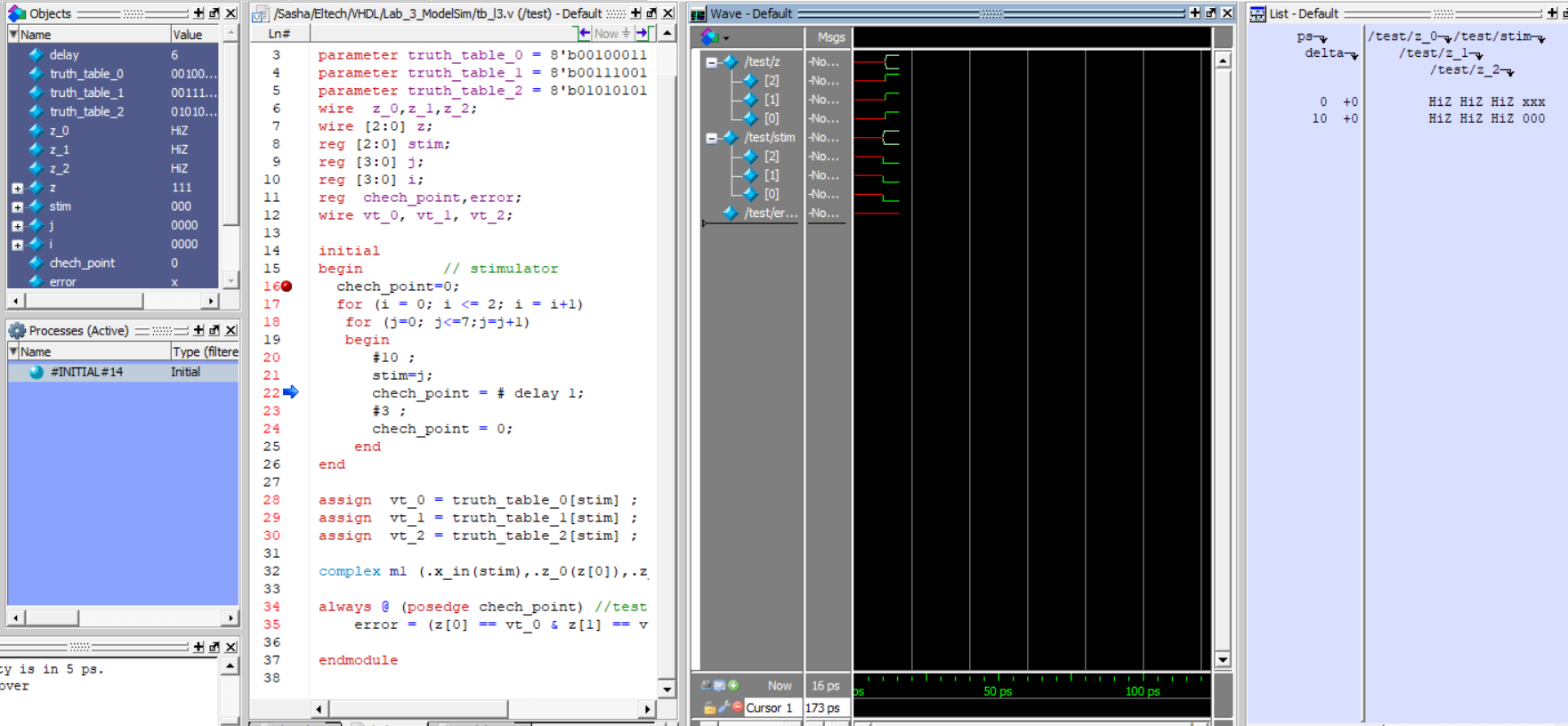


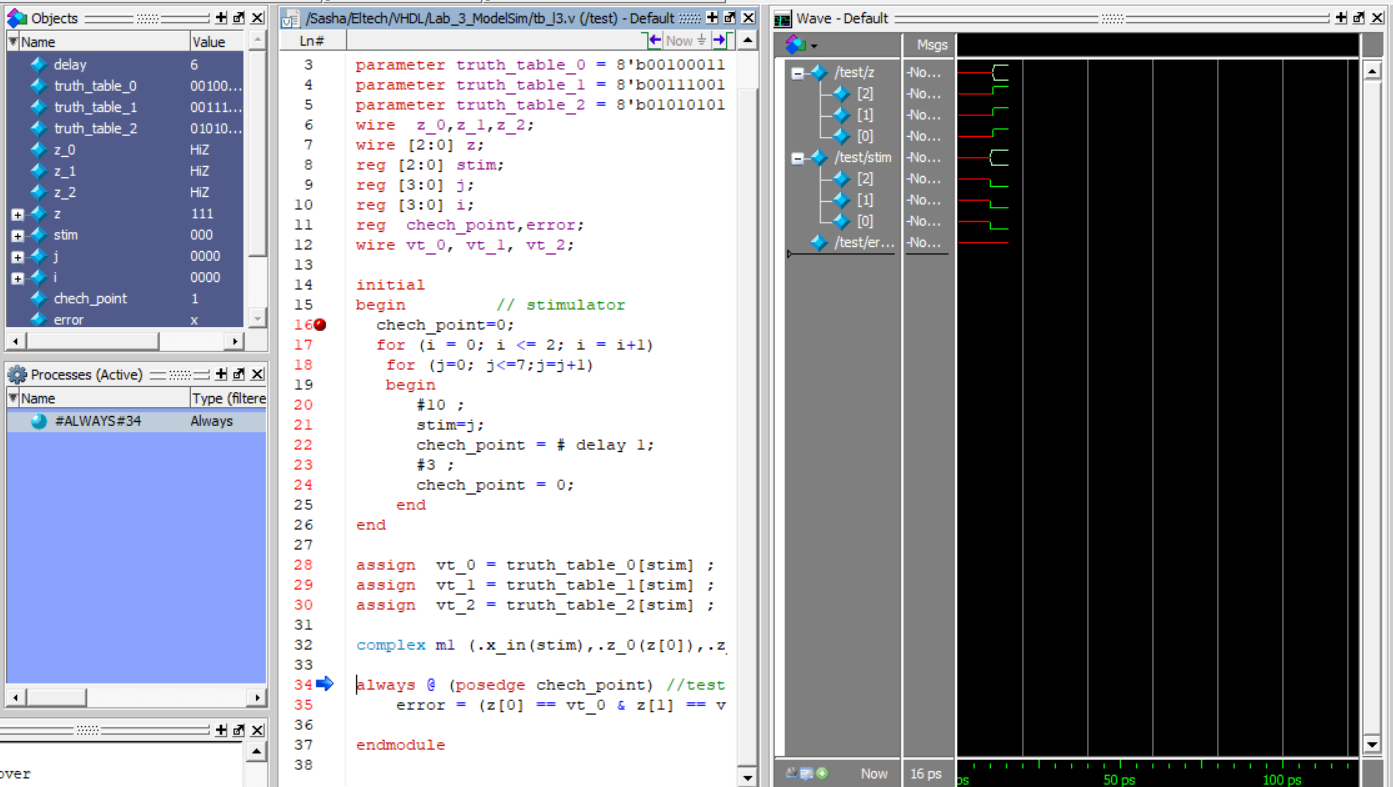
Получили ошибки, что говорит нам о том, что оператор tester работает верно.

1. Провели отладку в пошаговом режиме с наблюдением происходящего в окне List.

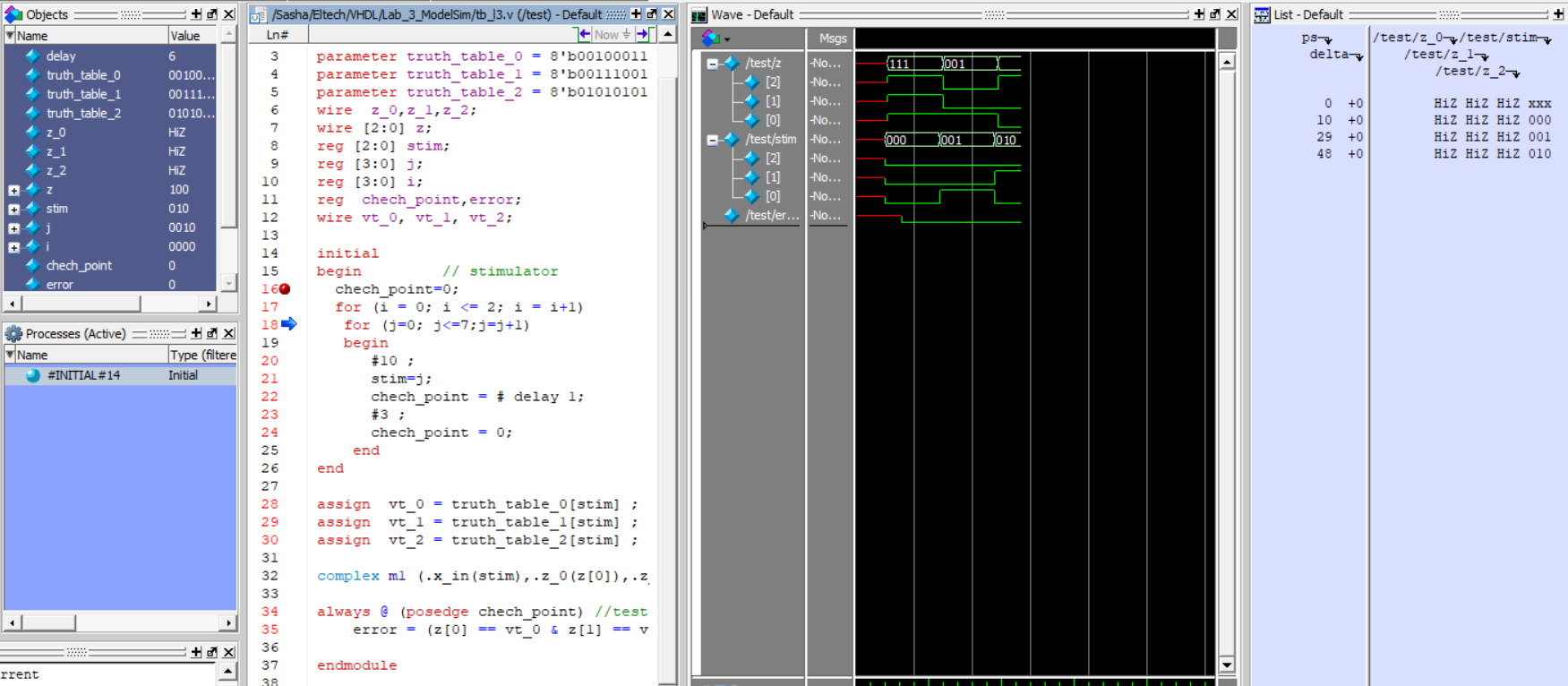


Как мы видим, так как в блоке цикла есть задержка, то следующим шагом идёт переход на строку 28.





После изменения chech\_point мы переходим в блок always, что можно наблюдать в окне Processes.

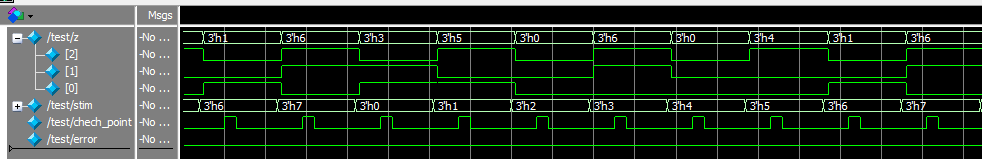


Как можно наблюдать в окне List – переменные инициализируются.

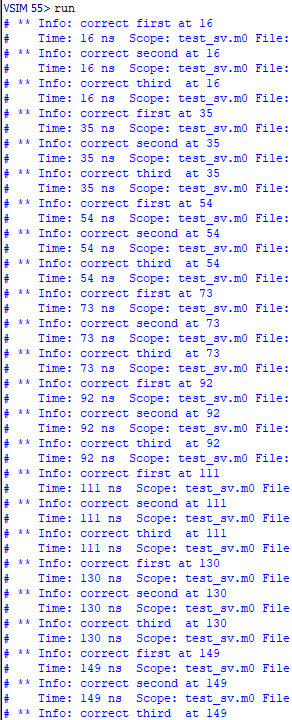
Часть 2

1. Создали новый проект на основе проекта из первой части, добавили файлы prog&interf.sv и Tb\_l3.sv.
2. Скорректировали файлы так, чтобы воспроизводились три логические функции в соответствии с условием. Провели симуляцию и запуск проекта, результаты на рисунках ниже.

Окно Wave:

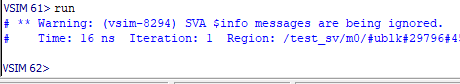


Окно получаемых сообщений:



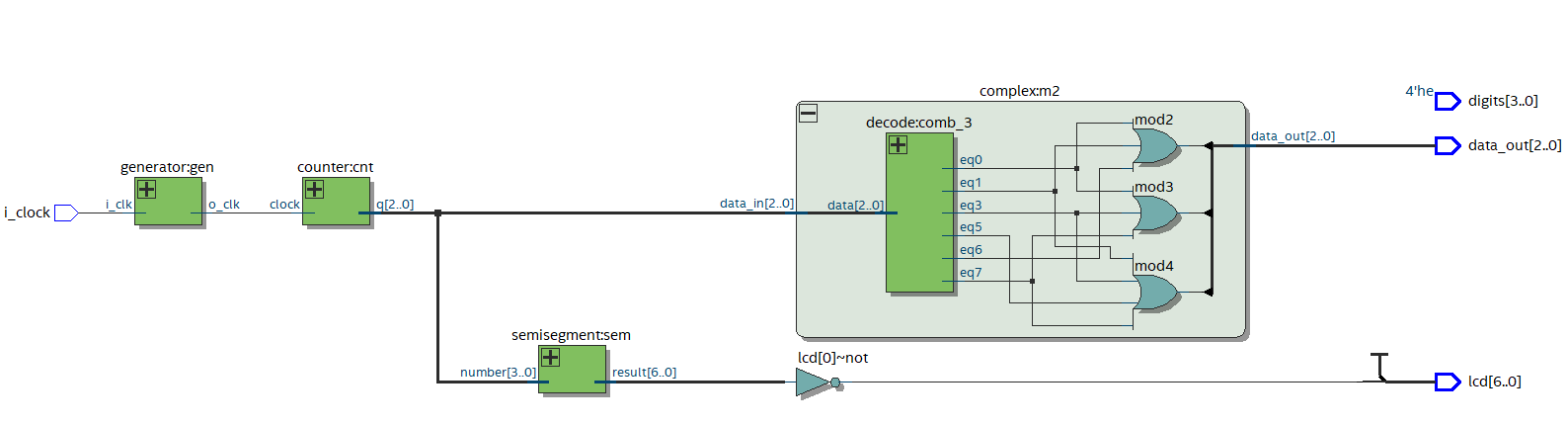
Как мы видим, функции вычисляются верно, ошибок не возникает.

1. Отключили показывание информационных сообщений



1. Подготовили проект для загрузки в макетную плату. Для этого перешли в среду Quartus, добавили мега-функцию счётчик и заменили наш дешифратор на мега-функцию дешифратор. Также подключили модуль вывода тестируемых данных (которые генерируются счётчиком) на циферблат и подключили модуль generator, который в зависимости от параметра понижает входной clock, чтобы данные были различимы. Три выходных сигнала будем выводить на светодиоды.

После компиляции проекта получили такое синтезированное устройство, как показано ниже на рисунке.



1. Произвели макетирование: назначили входные и выходные контакты и загрузили проект на плату и проследили за результатом. Результат макетирования можно наблюдать на рисунках ниже.

(рисунки-фото макетирования)

**Вывод**: в ходе выполнения данной работы мы научились пользоваться модулями, как встроенными, так и своими из других файлов. Познакомились с понятием testbench и провели тестирование модулей, без «влезания» внутрь, что позволит быстро использовать их для проектов в реальных платах ПЛИС. Конечные модули в приложении А.

# Приложение А

## Часть 1

**module test;**

parameter delay = 6;

parameter truth\_table\_0 = 8'b01000011; //For testing tester 8'b01000001;

parameter truth\_table\_1 = 8'b10001001;

parameter truth\_table\_2 = 8'b10101010;

wire [2:0] z;

reg [2:0] stim;

reg [3:0] j;

reg [3:0] i;

reg chech\_point,error;

wire vt\_0, vt\_1, vt\_2;

initial

begin // stimulator

chech\_point=0;

for (i = 0; i <= 2; i = i+1)

for (j=0; j<=7;j=j+1)

begin

#10 ;

stim=j;

chech\_point = # delay 1;

#3 ;

chech\_point = 0;

end

end

assign vt\_0 = truth\_table\_0[stim] ;

assign vt\_1 = truth\_table\_1[stim] ;

assign vt\_2 = truth\_table\_2[stim] ;

complex m1 (.x\_in(stim),.z\_0(z[0]),.z\_1(z[1]),.z\_2(z[2]));

always @ (posedge chech\_point) //tester

error = (z[0] == vt\_0 & z[1] == vt\_1 & z[2] == vt\_2) ? 0:1;

**endmodule**

**module decod(x\_in,y\_out);**

parameter delay=3; //delay

parameter n=4; // number of inputs

parameter u=8; //number of outputs

input x\_in; //ports mode

output y\_out; //ports mode

wire [n-1:0] x\_in; //ports type

reg [u-1:0] y\_out; //ports type

reg [n:0] i; //counter for loop

always @ (x\_in) //operator is executed whenever x\_in changes

begin

# delay;

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

y\_out[i] = x\_in==i ? 1:0;

end

**endmodule**

**module complex(x\_in,z\_0,z\_1,z\_2);**

input x\_in;

output z\_0,z\_1,z\_2 ;

wire [2:0] x\_in;

wire z\_0,z\_1,z\_2;

wire [7:0]y; //internal conections in module

decod # (1,3,8) mod1(x\_in,y);

// Change according to your variant

or mod2(z\_0, y[0], y[1], y[6]);

or mod3(z\_1, y[0], y[3], y[7]);

or mod4(z\_2, y[1], y[3], y[5], y[7]);

**endmodule**

## Часть 2

**interface bus();**

logic check\_point;

logic [2:0] data\_in;

logic [2:0] data\_out;

modport stim (output data\_in, data\_out, check\_point);

modport monit (input data\_in, data\_out, check\_point);

**endinterface**

**program stimul (bus.stim u2);**

parameter delay = 6;

integer j, i;

initial

begin

uuu.check\_point = 0;

for (i = 0; i <= 2; i = i+1)

for(j = 0; j <= 7; j = j + 1)

begin

#10;

u2.data\_in = j;

u2.check\_point = #delay 1;

#3;

u2.check\_point = 0;

end

end

**endprogram**

**module mot (bus.monit u1);**

parameter truth\_table\_1 = 8'b01000011;

parameter truth\_table\_2 = 8'b10001001;

parameter truth\_table\_3 = 8'b10101010;

logic vt\_1, vt\_2, vt\_3;

time c\_time;

assign vt\_1 = truth\_table\_1[u1.data\_in];

assign vt\_2 = truth\_table\_2[u1.data\_in];

assign vt\_3 = truth\_table\_3[u1.data\_in];

always @(posedge u1.check\_point)

begin

c\_time = $time;

assert (u1.data\_out[0] == vt\_1) $info("correct first at %0t", c\_time);

else $error("error first at %0t", c\_time);

assert (u1.data\_out[1] == vt\_2) $info("correct second at %0t", c\_time);

else $error("error second at %0t", c\_time);

assert (u1.data\_out[2] == vt\_3) $info("correct third at %0t", c\_time);

else $error("error third at %0t", c\_time);

end

**endmodule**

**module test\_sv;**

bus uuu ();

mot m0 (.u1(uuu));

stimul m1 (.u2(uuu));

complex m2 (uuu.data\_in, uuu.data\_out[0], uuu.data\_out[1], uuu.data\_out[2]);

**endmodule**

## Часть Quartus

**module Tb\_\_l3(data\_out, i\_clock, lcd, digits);**

input logic i\_clock; // input clock

output logic [6:0] lcd; // lcd semisegment pins

output logic [3:0] digits; //control semisegment digits

output logic [2:0] data\_out; //connect to the leds

logic [2:0] data\_in;

logic clock;

logic lcd\_inv;

assign lcd = ~lcd\_inv;

//Turn on digigtal 1

assign digits[0] = 1'b0;

assign digits[1] = 1'b1;

assign digits[2] = 1'b1;

assign digits[3] = 1'b1;

generator #(25000000) gen (i\_clock, clock); //For working with board: 25000000;

semisegment sem (.number(data\_in), .result(lcd\_inv)); //lcd\_inv

counter cnt (.clock(clock), .q(data\_in));

complex m2 (.data\_in(data\_in), .data\_out(data\_out));

**endmodule**

**module complex(data\_in, data\_out);**

input wire [2:0] data\_in;

output wire [2:0] data\_out;

wire [7:0] y; //internal conections in module

decode (

.data(data\_in),

.eq0(y[0]),

.eq1(y[1]),

.eq2(y[2]),

.eq3(y[3]),

.eq4(y[4]),

.eq5(y[5]),

.eq6(y[6]),

.eq7(y[7])

);

// Change according to your variant

or mod2(data\_out[0], y[0], y[1], y[6]);

or mod3(data\_out[1], y[0], y[3], y[7]);

or mod4(data\_out[2], y[1], y[3], y[5], y[7]);

**endmodule**