**Отчёт по лабораторной работе**

по дисциплине

«Языки моделирования и описания цифровой аппаратуры»

**Программная на RTL SystemC и аппаратная на Verilog HDL реализации алгоритма «Контроллер секундомера»**

Студент гр. 5130904/30008 Ребдев П.А

Проверил: Амосов В.В

Оглавление

[1. Техническое задание 3](#__RefHeading___Toc163_1041169204)

[2. Описание алгоритма 3](#__RefHeading___Toc175_1041169204)

[3. Программная реализация 4](#__RefHeading___Toc177_1041169204)

[3.1 Блок-схема 4](#__RefHeading___Toc179_1041169204)

[3.2 SystemC код 5](#__RefHeading___Toc181_1041169204)

[3.3 Симуляция (тестирование) 7](#__RefHeading___Toc183_1041169204)

[4. RTL SystemC 8](#__RefHeading___Toc4287_127534864)

[5. Verilog код 10](#__RefHeading___Toc4289_127534864)

[5.1 код 10](#__RefHeading___Toc4291_127534864)

[5.2 Тестирование симуляцией 12](#__RefHeading___Toc4293_127534864)

[5.3 Отчёт в среде Quartus II 12](#__RefHeading___Toc4295_127534864)

[5.4 RTL схема 12](#__RefHeading___Toc4297_127534864)

[5.5 Технологическая схема 12](#__RefHeading___Toc4299_127534864)

## 1. Техническое задание

1) Программная реализация на SystemC алгоритма работы контроллера секундомера

2) Произвести симуляцию SystemC кода

3) Программная реализация на RTL SystemC алгоритма работы контроллера секундомера

4) Произвести симуляцию RTL SystemC кода

5) Транслировать RTL SystemC в Verilog

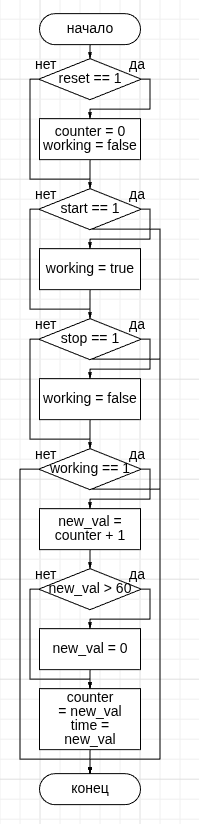
6) Произвести симуляцию Verilog кода

## 2. Описание алгоритма

Секундомер выдаёт число тактов clk с момента получения сигнала start. По сигналу stop секундомер останавливается. Сигналы start и stop выдаются на один такт. Счёт ведётся до 60.

## 3. **Программная реализация**

### 3.1 Блок-схема



### 3.2 SystemC код

main.cpp

#include <systemc.h>

#include "stopwatchController.h"

#include "stopwatchControllerTest.h"

int sc\_main(int argc,char \*argv[])

{

stopwatchController igen("igen");

stopwatchControllerTest igentest("igen\_test");

sc\_clock s\_clk("clk", 5 ,SC\_NS);

sc\_signal< bool > s\_reset("reset");

sc\_signal< bool > s\_start("start");

sc\_signal< bool > s\_stop("stop");

sc\_signal< sc\_uint< 6 > > s\_time("time");

igentest.clk(s\_clk);

igentest.reset(s\_reset);

igentest.start(s\_start);

igentest.stop(s\_stop);

igentest.time(s\_time);

igen.clk(s\_clk);

igen.reset(s\_reset);

igen.start(s\_start);

igen.stop(s\_stop);

igen.time(s\_time);

sc\_trace\_file \* tf = sc\_create\_vcd\_trace\_file("lab13");

tf->set\_time\_unit(1, SC\_NS);

sc\_trace(tf,s\_clk, "clk");

sc\_trace(tf,s\_reset, "reset");

sc\_trace(tf,s\_start, "start");

sc\_trace(tf,s\_stop," stop");

sc\_trace(tf,s\_time, "time");

sc\_start(20000, SC\_NS);

sc\_close\_vcd\_trace\_file(tf);

return 0;

}

stopwatchController.h

#ifndef STOPWATCHCONTROLLER\_H

#define STOPWATCHCONTROLLER\_H

#include <systemc.h>

SC\_MODULE(stopwatchController)

{

sc\_in< bool > clk;

sc\_in< bool > reset;

sc\_in< bool > start;

sc\_in< bool > stop;

sc\_out< sc\_uint< 6 > > time;

sc\_signal< bool > working;

sc\_signal< sc\_uint< 6 > > counter;

void update\_state();

SC\_CTOR(stopwatchController)

{

SC\_METHOD(update\_state);

sensitive << clk.pos();

dont\_initialize();

}

};

#endif

stopwatchController.cpp

#include "stopwatchController.h"

void stopwatchController::update\_state()

{

if(reset.read())

{

counter.write(0);

working.write(false);

}

else if (start.read())

{

working.write(true);

}

else if (stop.read())

{

working.write(false);

}

else if (working.read())

{

sc\_uint< 6 > new\_val = counter.read() + 1;

if (new\_val > 60)

{

new\_val = 0;

}

counter.write(new\_val);

time.write(new\_val);

}

}

stopwatchControllerTest.h

#ifndef STOPWATCHCONTROLLERTEST\_H

#define STOPWATCHCONTROLLERTEST\_H

#include <iostream>

#include <systemc.h>

SC\_MODULE(stopwatchControllerTest)

{

sc\_in< bool > clk;

sc\_out< bool > reset;

sc\_out< bool > start;

sc\_out< bool > stop;

sc\_in< sc\_uint< 6 > > time;

unsigned int controllerTime;

void generate\_signals();

SC\_CTOR(stopwatchControllerTest):

controllerTime(0)

{

SC\_THREAD(generate\_signals);

sensitive << clk.pos();

}

};

#endif

stopwatchControllerTest.cpp

#include "stopwatchControllerTest.h"

void stopwatchControllerTest::generate\_signals()

{

reset.write(1);

wait();

reset.write(0);

wait(10, SC\_NS);

while(true)

{

start.write(1);

wait(5, SC\_NS);

start.write(0);

wait(10, SC\_NS);

stop.write(1);

wait(5, SC\_NS);

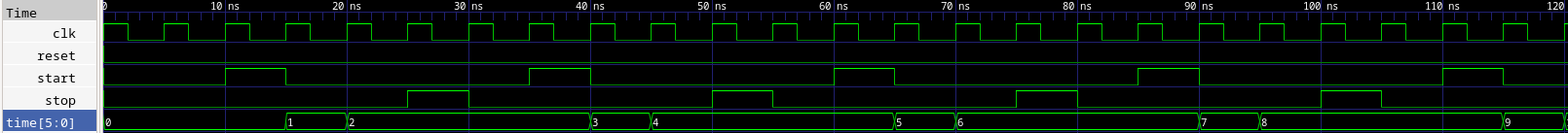
stop.write(0);

wait(5, SC\_NS);

}

}

### 3.3 Симуляция (тестирование)



## 4. RTL SystemC

stopwatchControllerRTL.h

#ifndef STOPWATCHCONTROLLERRTL\_H

#define STOPWATCHCONTROLLERRTL\_H

#include <systemc.h>

SC\_MODULE(stopwatchControllerRTL)

{

sc\_in< bool > clk;

sc\_in< bool > reset;

sc\_in< bool > start;

sc\_in< bool > stop;

sc\_out< sc\_uint< 6 > > time;

sc\_signal< bool > running;

sc\_signal< bool > next\_running;

sc\_signal< sc\_uint< 6 > > counter;

sc\_signal< sc\_uint< 6 > > next\_counter;

void comb\_logic();

void seq\_logic();

SC\_CTOR(stopwatchControllerRTL)

{

SC\_METHOD(seq\_logic);

sensitive << clk.pos();

SC\_METHOD(comb\_logic);

sensitive << reset << start << stop << running << counter;

}

};

#endif

stopwatchControllerRTL.cpp

#include "stopwatchController.h"

void stopwatchController::update\_state()

{

if(reset.read())

{

counter.write(0);

working.write(false);

}

else if (start.read())

{

working.write(true);

}

else if (stop.read())

{

working.write(false);

}

else if (working.read())

{

sc\_uint< 6 > new\_val = counter.read() + 1;

if (new\_val > 60)

{

new\_val = 0;

}

counter.write(new\_val);

time.write(new\_val);

}

}

## 5. Verilog код

### 5.1 код

stopwatchControllerRTL.v

module stopwatchControllerRTL(timer,stop,start,reset,clk);

output [5:0] timer;

input stop;

input start;

input reset;

input clk;

reg [5:0] timer;

reg [5:0] new\_count;

reg [5:0] next\_counter;

reg [5:0] counter;

reg next\_running;

reg running;

//comb\_logic:

always @(counter or running or stop or start or reset )

begin

next\_running =(running );

next\_counter =(counter );

timer =(counter );

if (reset )

begin

next\_running =(0);

next\_counter =(0);

end

else

begin

if (start )

begin

next\_running =(1);

end

else if (stop )

begin

next\_running =(0);

end

else if (running )

begin

new\_count =counter +1;

if (new\_count >60)

begin

new\_count =(0);

end

next\_counter =(new\_count );

end

end

end

//seq\_logic:

always @(posedge clk )

begin

if (reset )

begin

running <=(0);

counter <=(0);

end

else

begin

running <=(next\_running );

counter <=(next\_counter );

end

end

endmodule

### 5.2 Тестирование симуляцией

### 5.3 Отчёт в среде Quartus II

### **5.4 RTL схема**

### **5.5 Технологическая схема**

