Федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский политехнический университет
Петра Великого»
Институт компьютерных наук и технологий
Программная инженерия

ПОЛИТЕХ
Санкт-Петербургский
политехнический университет
Петра Великого

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

Санкт-Петербург
2025

Оглавление

1. Техническое задание....3
2. Описание алгоритма....3
3. Программная реализация....4
   3.1 Блок-схема....4
   3.2 SystemC код....5
   3.3 Симуляция (тестирование)....7
4. RTL SystemC....8
5. Verilog код....10
   5.1 код....10
   5.2 Тестирование симуляцией....12
   5.3 Отчёт в среде Quartus II....12
   5.4 RTL схема....12
   5.5 Технологическая схема....12

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

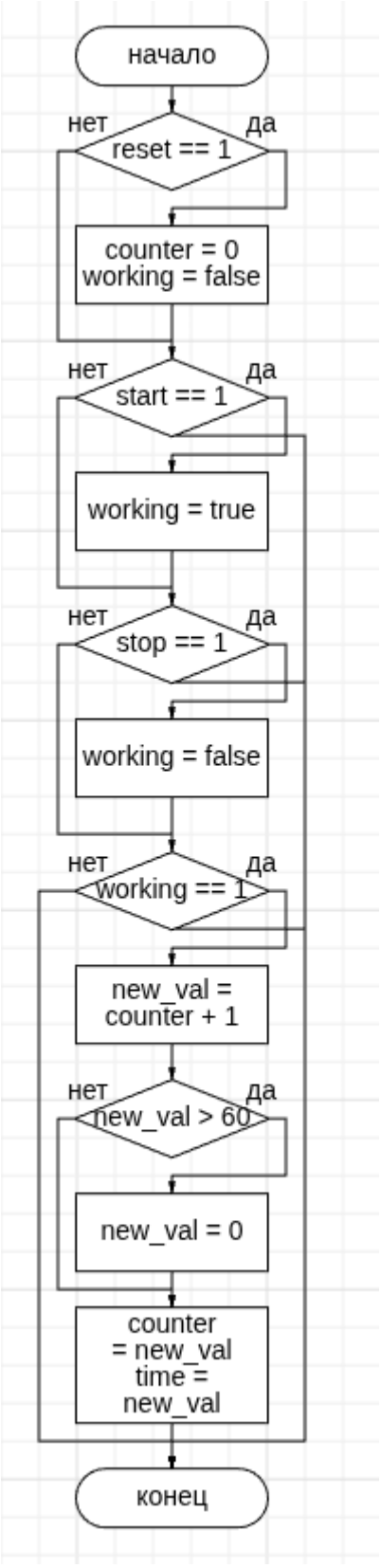
6) Произвести симуляцию Verilog кода

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

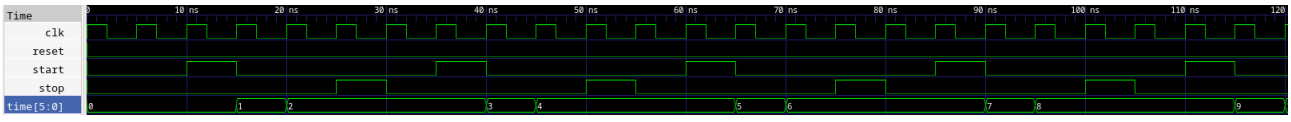
```
#ifndef STOPWATCHCONTROLLERTEST_H
#define STOPWATCHCONTROLLERTEST_H

#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

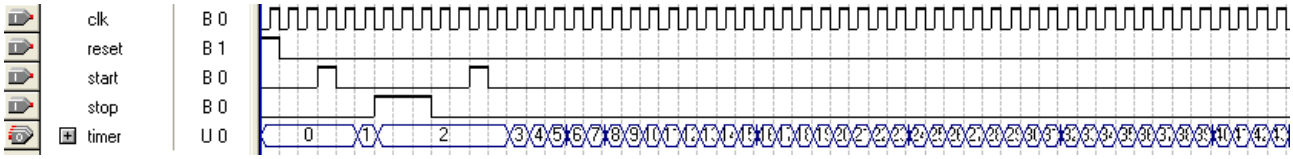
5.1 код

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
```

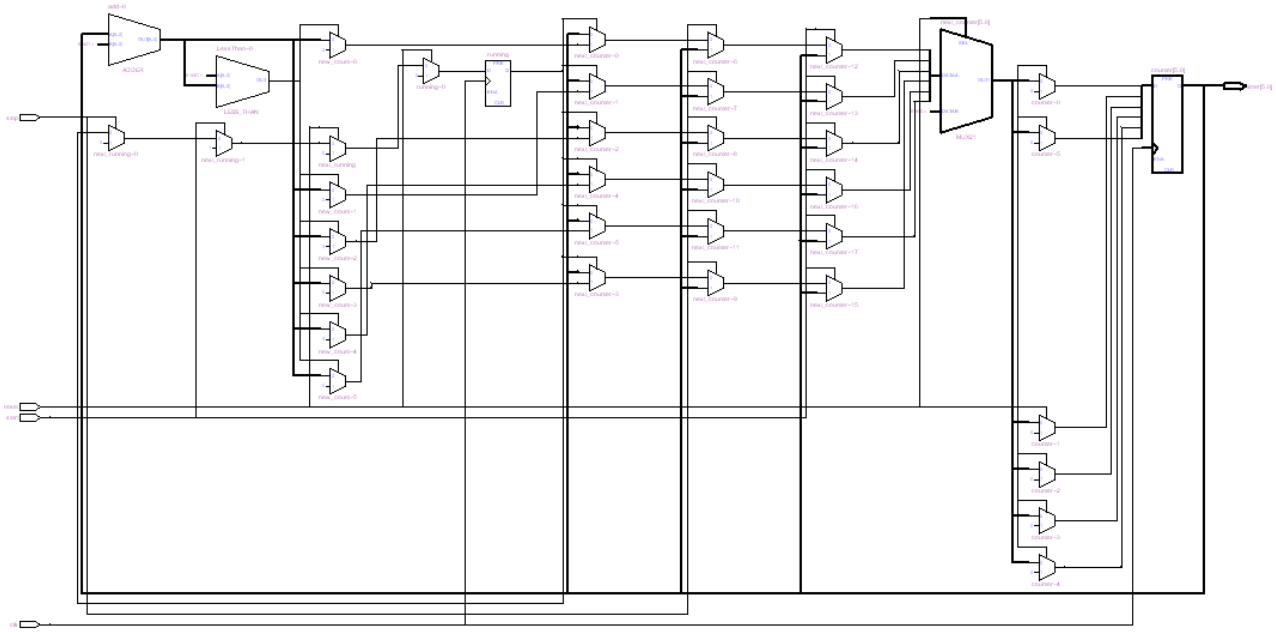
5.2 Тестирование симуляцией



5.3 Отчёт в среде Quartus II

Flow Status	Successful - Fri Jun 04 16:44:30 2021
Quartus II Version	5.0 Build 148 04/26/2005 SJ Full Version
Revision Name	src
Top-level Entity Name	stopwatchControllerRTL
Family	Stratix
Met timing requirements	Yes
Total logic elements	13 / 10,570 (< 1 %)
Total pins	10 / 336 (2 %)
Total virtual pins	0
Total memory bits	0 / 920,448 (0 %)
DSP block 9-bit elements	0 / 48 (0 %)
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 2 (0 %)
Device	EP1S10F484C5
Timing Models	Final

5.4 RTL схема



5.5 Технологическая схема