Министерство образования и науки РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа программной инженерии

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

по дисциплине « Языки моделирования и описания цифровой аппаратуры»

**Программная на** **RTL** **SystemC и аппаратная на Verilog HDL реализации алгоритма Дифференциатора**

**Вариант №22**

Выполнил:

Студент гр. 5130904/30007 Голиков В.С.

Проверил: Амосов В.В.

Санкт-Петербург

2025 г.

**Оглавление**

[1. Формулировка задачи 3](#_Toc197981062)

[2. Блок схема системы дифференциатора 4](#_Toc197981063)

[3. Текст программной реализации на RTL SystemC 5](#_Toc197981064)

[4. Тестирование симуляцией (временная диаграмма) 10](#_Toc197981065)

[5. Блок-схема аппаратной реализации дифференциатора 11](#_Toc197981066)

[6. Сгенерированный Verilog код (через sc2v) 12](#_Toc197981067)

[7. Тестирование симуляцией 13](#_Toc197981068)

[8. Отчет в среде Quartus 13](#_Toc197981069)

[9. RTL схема 14](#_Toc197981070)

[10. Технологическая схема 14](#_Toc197981071)

# Формулировка задачи

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

# Изображение выглядит как текст, диаграмма, План, Шрифт Контент, сгенерированный ИИ, может содержать ошибки.Блок схема системы дифференциатора

# Текст программной реализации на RTL SystemC

**differentiator.h**

#ifndef DIFFERENTIATOR\_H

#define DIFFERENTIATOR\_H

#include <systemc.h>

SC\_MODULE(Differentiator) {

sc\_in<bool> clk;

sc\_in<bool> reset;

sc\_in<sc\_int<16>> in;

sc\_out<sc\_int<16>> out;

sc\_int<16> prev\_value;

bool first\_sample;

void compute\_diff();

SC\_CTOR(Differentiator);

};

#endif

**differentiator.cpp**

#include "differentiator.h"

Differentiator::Differentiator(sc\_module\_name name) : sc\_module(name), first\_sample(true) {

SC\_METHOD(compute\_diff);

sensitive << clk.pos() << reset;

}

void Differentiator::compute\_diff() {

if (reset.read()) {

prev\_value = 0;

out.write(0);

first\_sample = true;

} else if (clk.posedge()) {

sc\_int<16> current = in.read();

if (first\_sample) {

out.write(0);

first\_sample = false;

} else {

out.write(current - prev\_value);

}

prev\_value = current;

}

}

**testbench.h**

#ifndef TESTBENCH\_H

#define TESTBENCH\_H

#include <systemc.h>

SC\_MODULE(Testbench) {

sc\_out<bool> clk;

sc\_out<bool> reset;

sc\_out<sc\_int<16>> in;

sc\_in<sc\_int<16>> out;

void test\_sequence();

void clock\_gen();

SC\_CTOR(Testbench);

};

#endif

**testbench.cpp**

#include "testbench.h"

#include <iostream>

Testbench::Testbench(sc\_module\_name name) : sc\_module(name) {

SC\_THREAD(test\_sequence);

SC\_THREAD(clock\_gen);

}

void Testbench::test\_sequence() {

reset.write(true);

wait(10, SC\_NS);

reset.write(false);

sc\_int<16> test\_data[8] = {1, 3, 5, 6, 7, 3, 0, 1};

for (int i = 0; i < 8; i++) {

in.write(test\_data[i]);

wait(10, SC\_NS);

cout << "At time " << sc\_time\_stamp() << ": ";

cout << "In = " << in.read() << ", Out = " << out.read() << endl;

}

wait(10, SC\_NS);

sc\_stop();

}

void Testbench::clock\_gen() {

clk.write(true);

wait(5, SC\_NS);

while(true) {

clk.write(false);

wait(5, SC\_NS);

clk.write(true);

wait(5, SC\_NS);

}

}

**differentiator\_RTL.h**

#ifndef DIFFERENTIATOR\_RTL\_H

#define DIFFERENTIATOR\_RTL\_H

#include <systemc.h>

SC\_MODULE(Differentiator\_RTL) {

// Портовая карта (как в задании)

sc\_in<bool> clk; // Тактовый сигнал

sc\_in<bool> reset; // Сброс (активный высокий уровень)

sc\_in<sc\_int<16>> in; // Входные данные

sc\_out<sc\_int<16>> out; // Выходные данные (разность)

// Внутренние регистры (аналогично ImpulseGenRTL)

sc\_signal<sc\_int<16>> prev\_in\_reg; // Хранение предыдущего значения

sc\_signal<bool> valid\_reg; // Флаг валидности данных

// Методы

void comb\_logic(); // Комбинационная логика

void seq\_logic(); // Последовательная логика

// Конструктор

SC\_CTOR(Differentiator\_RTL) {

SC\_METHOD(seq\_logic);

sensitive << clk.pos() << reset;

SC\_METHOD(comb\_logic);

sensitive << in << prev\_in\_reg << valid\_reg;

}

};

#endif

**differentiator\_RTL.cpp**

#include "Differentiator\_RTL.h"

void Differentiator\_RTL::comb\_logic() {

if (reset.read()) {

out.write(0); // На сбросе выход = 0

} else {

if (valid\_reg.read()) {

out.write(in.read() - prev\_in\_reg.read()); // Вычисляем разность

} else {

out.write(0); // Первый отсчёт - разность 0

}

}

}

void Differentiator\_RTL::seq\_logic() {

if (reset.read()) {

prev\_in\_reg.write(0);

valid\_reg.write(false);

} else {

prev\_in\_reg.write(in.read()); // Запоминаем текущее значение

valid\_reg.write(true); // Данные стали валидными

}

}

**main.cpp**

#include <systemc.h>

#include "differentiator.h"

#include "testbench.h"

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

sc\_set\_time\_resolution(1, SC\_NS);

sc\_signal<bool> clk, reset;

sc\_signal<sc\_int<16>> input, output;

Differentiator diff("Differentiator");

diff.clk(clk);

diff.reset(reset);

diff.in(input);

diff.out(output);

Testbench tb("Testbench");

tb.clk(clk);

tb.reset(reset);

tb.in(input);

tb.out(output);

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

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

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

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

sc\_trace(tf, input, "input");

sc\_trace(tf, output, "output");

sc\_trace(tf, diff.prev\_value, "prev\_value");

cout << "Starting simulation..." << endl;

sc\_start();

cout << "Simulation finished." << endl;

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

return 0;

}

# Тестирование симуляцией (временная диаграмма)

# Блок-схема аппаратной реализации дифференциатора

Изображение выглядит как текст, диаграмма, План, зарисовка

Контент, сгенерированный ИИ, может содержать ошибки.

# Сгенерированный Verilog код (через sc2v)

module Differentiator\_RTL (

input wire clk,

input wire reset,

input wire signed [15:0] data\_in,

output reg signed [15:0] diff\_out

);

reg signed [15:0] prev\_data;

reg first\_sample;

wire signed [15:0] comb\_diff;

//комбинационная

assign comb\_diff = first\_sample ? 16'b0 : (data\_in - prev\_data);

//последовательная

always @(posedge clk or posedge reset) begin

if (reset) begin

prev\_data <= 16'b0;

diff\_out <= 16'b0;

first\_sample <= 1'b1;

end else begin

diff\_out <= comb\_diff;

prev\_data <= data\_in;

first\_sample <= 1'b0;

end

end

endmodule

# Тестирование симуляцией

Изображение выглядит как текст, линия, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

# Отчет в среде Quartus

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

# RTL схема

Изображение выглядит как зарисовка, Параллельный, линия

Контент, сгенерированный ИИ, может содержать ошибки.

# 10. Технологическая схема

Изображение выглядит как текст, линия, диаграмма, Параллельный

Контент, сгенерированный ИИ, может содержать ошибки.