Санкт-Петербургский государственный политехнический университет
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе

Дисциплина: Высокоуровневое моделирование средствами SystemC

Тема: Конечный автомат на языке SystemC.

Выполнил студент гр. 13541/2 _____ В.Е. Бушин
(подпись)
Руководитель _____ О.В. Мамутова
(подпись)

“_” _____2017 г.

Санкт-Петербург
2017

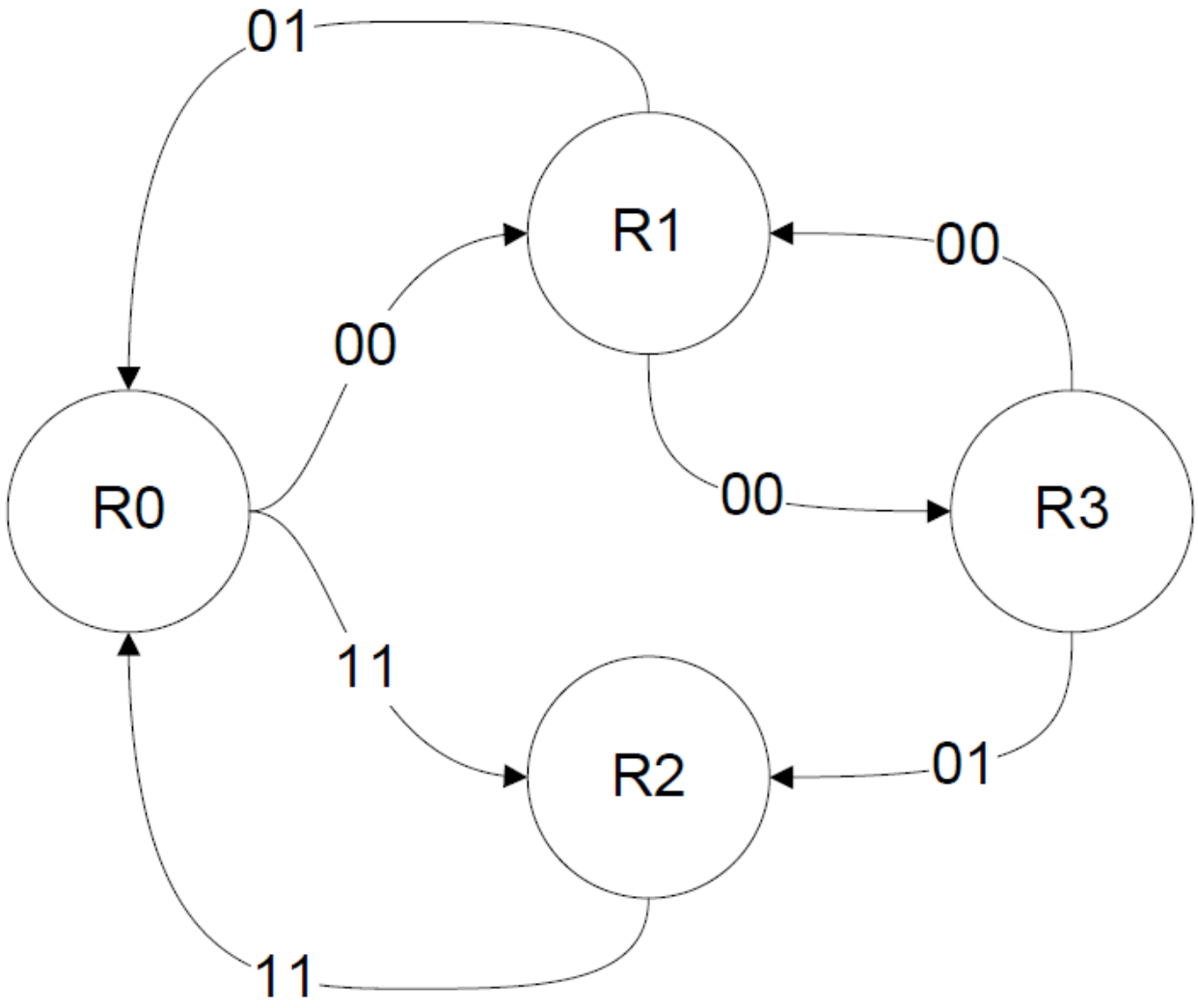
Задана таблица переходов конечного автомата:

Тип триггера	Таблица переходов				Таблица выходов			
	$x_2x_1 = 00$	$x_2x_1 = 01$	$x_2x_1 = 10$	$x_2x_1 = 11$	r_0	r_1	r_2	r_3
D	1321	H022	HHHH	2H0H	11	01	00	10

Таким образом, получаем следующую таблицу:

x_2x_1	r_0	r_1	r_2	r_3
00	1	3	2	1
01	H	0	2	2
10	H	H	H	H
11	2	H	0	H

Данной таблице соответствует следующий граф переходов:



Был реализован конечный автомат по заданной таблице переходов.
Реализация конечного автомата находится в файлах state.h и state.cpp.

Содержимое файла state.h:

```
#include "systemc.h"

#ifndef STATE_H
#define      STATE_H

SC_MODULE(state_machine) {
    sc_in_clk clock; // Clock input of the design
    sc_in<bool> reset; // active high, asynchronous Reset input
    sc_in<sc_uint<2> > in;
    sc_out<sc_uint<2> > out;

    sc_signal<sc_uint<2> > state;

    void state_change();

    void data_out_method();

    SC_CTOR(state_machine) :
    clock("clock"),
            reset("reset"),
            in("in"),
            out("out")
    {
        cout << "Executing new" << endl;
        SC_CTHREAD(state_change, clock.pos());
        async_reset_signal_is(reset, true);
        SC_METHOD(data_out_method);
        sensitive << state;
    }
};

#endif /* STATE_H */
```

Содержимое файла state.cpp:

```
#include "state.h"

void state_machine::state_change()
{
    state = 0;
    wait();
    while(true){
        switch(state.read()){
            case 0:{
                if(in.read() == 0)
                    state = 1;
                else if(in.read() == 3)
                    state = 2;
                break;
            }
            case 1:{
                if(in.read() == 0)
                    state = 3;
                else if(in.read() == 1)
                    state = 0;
                break;
            }
            case 2:{
                if(in.read() == 3)
                    state = 0;
                break;
            }
```

```
            case 3:{
                if(in.read() == 0)
                    state = 1;
                else if(in.read() == 1)
                    state = 2;
                break;
            }
        }
        wait();
    }
}

void state_machine::data_out_method()
{
    switch(state.read())
    {
        case 0:{
            out = 3;
            break;
        }
        case 1:{
            out = 1;
            break;
        }
        case 2:{
            out = 0;
            break;
        }
        case 3:{
            out = 2;
            break;
        }
    }
}
```

Для конечного автомата был создан следующий тест с самопроверкой:

```
int sc_main(int argc, char* argv[]) {
    sc_clock clock("clock", 4, SC_NS);
    sc_signal<bool> reset;
    sc_signal<sc_uint<2> > in;
    sc_signal<sc_uint<2> > out;

    state_machine test("test");
    test.clock(clock);
    test.reset(reset);
    test.in(in);
    test.out(out);

    // Open VCD file
    sc_trace_file *wf = sc_create_vcd_trace_file("state_waveform");
    // Dump the desired signals
    sc_trace(wf, clock, "clock");
    sc_trace(wf, reset, "reset");
    sc_trace(wf, in, "in");
    sc_trace(wf, out, "out");
    sc_trace(wf, test.state, "state");

    reset = 1; // Assert the reset
    cout << "@" << sc_time_stamp() << " Asserting reset\n" << endl;

    sc_start(6, SC_NS);
    assert(out.read() == 3);
```

```
    reset = 0; // De-assert the reset
    cout << "@" << sc_time_stamp() << " De-Asserting reset\n" << endl;
    in = 0;
    sc_start(17, SC_NS);
    assert(out.read() == 2);

    in = 1;
    sc_start(6, SC_NS);
    assert(out.read() == 0);

    in = 3;
    sc_start(14, SC_NS);
       assert(out.read() == 3);

    in = 0;
    sc_start(5, SC_NS);
       assert(out.read() == 1);

    in = 1;
    sc_start(12, SC_NS);
       assert(out.read() == 3);
    return 0;
}
```

Открыв сгенерированный файл state_waveform.vcd в GTKWave, мы увидим следующее: