Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчет по лабораторной работе №2**

по дисциплине «SystemC»

Проектирование конечного автомата

Выполнил

студент гр. 13541/2

Никитенко А.П.

Преподаватели

Петров М.А.

Мамутова О.В.

**Цель работы**: разработка конечного автомата на языке SystemC согласно варианту и проведение тестирования.

**Ход работы.**

Таблица переходов согласно варианту представлена в Таблице 1.

Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x2x1** | **r0** | **r1** | **r2** | **r3** |
| **00** | 2 | 2 | 2 | 1 |
| **01** | 1 | Н | 1 | H |
| **10** | Н | Н | Н | Н |
| **11** | 3 | 3 | H | 0 |

Таблица выходов конечного автомата представлена в Таблице 2.

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| **R0** | **R1** | **R2** | **R3** |
| 10 | 10 | 01 | 01 |

По данным таблицам был составлен граф переходов конечного автомата (рис. 1).

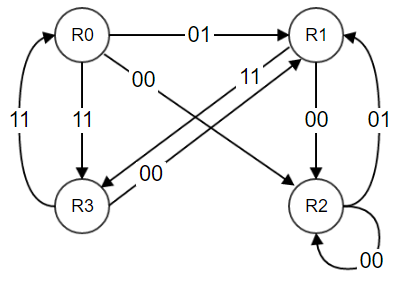


Рис. 1. Граф переходов конечного автомата

Реализация конечного автомата представлена в файлах state-machine.h и state\_machine.cpp(Приложение 1). Реализация тестирования представлена в файле testbench.cpp(Приложение 2).

Результаты тестирования сгенерированы в .vcd файл и совпадают с ожидаемыми результатами.

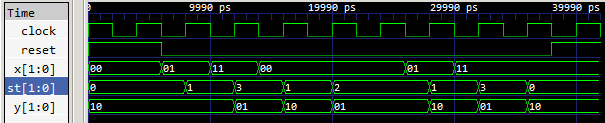


Рис. 2. Содержимое .vcd файла

Приложение 1

**state\_machine.h**

#include "systemc.h"

#ifndef DESIGN\_H

#define DESIGN\_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> > x; //input

sc\_out<sc\_uint<2> > y; //output

sc\_out<sc\_uint<2> > st;

//------------Local Variables Here---------------------

sc\_signal<sc\_uint<2> > state;

//------------Code Starts Here-------------------------

// Below function implements actual register logic

void change\_st();

void data\_out();

// Constructor

SC\_CTOR(state\_machine) :

clock("clock"),

reset("reset"),

x("x"),

y("y"),

st("st"){

cout << "Executing new" << endl;

SC\_CTHREAD(change\_st, clock.pos());

async\_reset\_signal\_is(reset, true);

SC\_METHOD(data\_out);

sensitive << state;

} // End of Constructor

}; // End of Module

#endif /\* DESIGN\_H \*/

**state\_machine.cpp**

#include "state\_machine.h"

void state\_machine::change\_st() {

state = 0;

wait();

while (true) {

switch(state.read()){

case 0:

if (x.read() == 0) state = 2;

else if (x.read() == 1) state = 1;

else if (x.read() == 3) state = 3;

else state = 0;

break;

case 1:

if (x.read() == 0) state = 2;

else if (x.read() == 3) state = 3;

else state = 0;

break;

case 2:

if (x.read() == 0) state = 2;

else if (x.read() == 1) state = 1;

else state = 0;

break;

case 3:

if (x.read() == 0) state = 1;

else if (x.read() == 3) state = 0;

else state = 0;

break;

default:

state = 0;

}

wait();

}

}

void state\_machine::data\_out() {

switch(state.read()){

case 0: st=0; y = 2; break;

case 1: st=1; y = 2; break;

case 2: st=2; y = 1; break;

default: st=3; y = 1;

}

}

Приложение 2

**testbench.cpp**

#include "systemc.h"

#include "state\_machine.h"

#define soft\_assert(signal, expected) \

if (signal.read() != expected) { \

cerr << "@" << sc\_time\_stamp() << " Check failed. Expected: " << expected << ". Actual: " << signal.read() << ".\n" << endl; \

}

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

sc\_clock clock("clock", 4, SC\_NS);

sc\_signal<bool> reset;

sc\_signal<sc\_uint<2> > x;

sc\_signal<sc\_uint<2> > y;

sc\_signal<sc\_uint<2> > st;

int i = 0;

// Connect the DUT

state\_machine test\_state\_machine("test\_state\_machine");

test\_state\_machine.clock(clock);

test\_state\_machine.reset(reset);

test\_state\_machine.x(x);

test\_state\_machine.y(y);

test\_state\_machine.st(st);

// Open VCD file

sc\_trace\_file \*wf = sc\_create\_vcd\_trace\_file("st\_machine\_waveform");

// Dump the desired signals

sc\_trace(wf, clock, "clock");

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

sc\_trace(wf, x, "x");

sc\_trace(wf, y, "y");

sc\_trace(wf, st, "st");

reset = 1; // Assert the reset

cout << "@" << sc\_time\_stamp() << " Asserting reset\n" << endl;

sc\_start(6, SC\_NS);

reset = 0; // De-assert the reset

cout << "@" << sc\_time\_stamp() << " De-Asserting reset\n" << endl;

x = 1;

sc\_start(4, SC\_NS);

assert(y.read() == 2);

x = 3;

sc\_start(4, SC\_NS);

assert(y.read() == 1);

x = 0;

sc\_start(4, SC\_NS);

assert(y.read() == 2);

x = 0;

sc\_start(4, SC\_NS);

assert(y.read() == 1);

x = 0;

sc\_start(4, SC\_NS);

assert(y.read() == 1);

x = 1;

sc\_start(4, SC\_NS);

assert(y.read() == 2);

x = 3;

sc\_start(4, SC\_NS);

assert(y.read() == 1);

x = 3;

sc\_start(4, SC\_NS);

assert(y.read() == 2);

reset = 1; // Assert the reset

cout << "@" << sc\_time\_stamp() << " Asserting reset\n" << endl;

sc\_start(1, SC\_NS);

assert(y.read() == 2);

sc\_start(5, SC\_NS);

cout << "@" << sc\_time\_stamp() << " Terminating simulation\n" << endl;

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

return 0; // Terminate simulation

}