Санкт-Петербургский государственный политехнический университет

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторным работам**

**Дисциплина**: Высокоуровневое моделирование средствами SystemC

Выполнил студент гр. 13541/2 Муравьев Ф.Э.

(подпись)

Руководитель Мамутова О.В.

(подпись)

“ ” 2017 г.

Санкт-Петербург

2017

**Лабораторная работа №2**

**Программа работы:**

1. Скопировать в локальную папку проект с примером регистра (с подсчетом четности).
2. Выполнить компиляцию проекта. Запустить созданное приложение, наблюдая результаты моделирования устройства в консоли. Проверить правильность работы устройства, открыв сгенерированный vcd файл в GTKWave.
3. Разработать конечный автомат Мура по индивидуальному заданию.
4. Создать тесты с самопроверкой для всех переходов конечного автомата.

**Вариант индивидуального задания**

Разработать конечный автомат:

Таблица переходов

|  |  |  |  |
| --- | --- | --- | --- |
| x2x1=00 | x2x1=01 | x2x1=10 | x2x1=11 |
| 30Н3 | 12Н3 | 110Н | НННН |

Таблица выходов

|  |  |  |  |
| --- | --- | --- | --- |
| r0 | r1 | r2 | r3 |
| 21 | 21 | 21 | 21 |
| 01 | 10 | 01 | 10 |

Граф переходов конечного автомата

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **00** | 3 | 0 | Н | 3 |
| **01** | 1 | 2 | Н | 2 |
| **10** | 1 | 1 | 0 | Н |
| **11** | Н | Н | Н | Н |

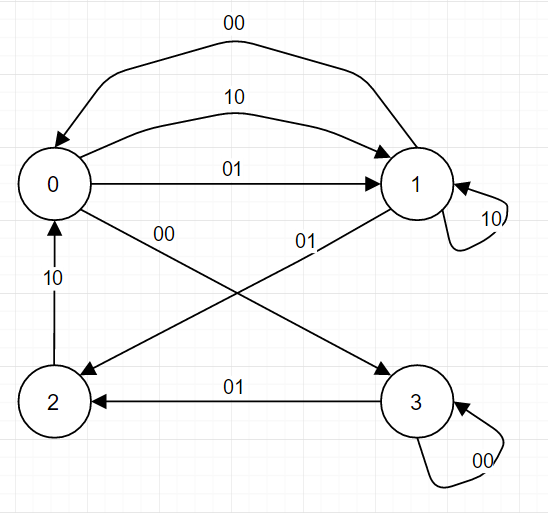


Рис 1.1. Граф переходов конечного автомата

**Выполнение работы.**

1. Скопировали проект с примером регистра
2. Скомпилировали проект. Запустили созданное приложение.
3. Проверили полученные результаты моделирования устройства.
4. Разработка конечного автомата согласно варианту индивидуального задания

Листинг 1.1. Файл statemashine.h

|  |
| --- |
| #include "systemc.h"  #ifndef DESIGN\_H  #define DESIGN\_H  SC\_MODULE(statemashine) {  sc\_in\_clk clock; // Clock input of the design  sc\_in<bool> reset; // active high, asynchronous Reset input  sc\_in<sc\_uint<2> > register\_in; // 8 bit vector input  sc\_out<sc\_uint<2> > register\_out; // 8 bit vector output    //------------Local Variables Here---------------------  sc\_signal<sc\_uint<2> > state;  //------------Code Starts Here-------------------------  // Below function implements actual register logic  void register\_store();    void data\_out\_method();  // Constructor for the register  // Since this register is a positive edge trigged one,  // We trigger the below block with respect to positive  // edge of the clock and also when ever reset changes state  SC\_CTOR(statemashine) :  clock("clock"),  reset("reset"),  register\_in("register\_in"),  register\_out("register\_out") {  cout << "Executing new" << endl;  SC\_CTHREAD(register\_store, clock.pos());  async\_reset\_signal\_is(reset, true);  SC\_METHOD(data\_out\_method);  sensitive << state;  } // End of Constructor  }; // End of Module  #endif /\* DESIGN\_H \*/ |

Листинг 1.2. Файл statemashine.cpp

|  |
| --- |
| #include "statemashine.h"  void statemashine::register\_store() {  state=0;  wait();  while (true) {  cout << "\*\*\*\*\*\*"<<endl;  cout << "Previous state == " << state<<" Input signal == "<<register\_in.read()<<endl;  if(state.read()==0)  {  if(register\_in.read()==0) {state=3; cout << " Now state == 3"<<endl;}  if((register\_in.read()==1)||(register\_in.read()==2)) {state=1; cout << " Now state == 1"<<endl;}  if(register\_in.read()==3) cout << " Now state == 0"<<endl;  }  if(state.read()==1)  {  if(register\_in.read()==0) {state=0; cout << " Now state == 0"<<endl;}  if(register\_in.read()==1) {state=2; cout << " Now state == 2"<<endl;}  if(register\_in.read()==2) {state=1; cout << " Now state == 1"<<endl;}  if(register\_in.read()==3) cout << " Now state == 1"<<endl;  }  if(state.read()==2)  {  if(register\_in.read()==2) {state=0; cout << " Now state == 0"<<endl;}  if(register\_in.read()!=2) cout << " Now state == 2"<<endl;  }  if(state.read()==3)  {  if(register\_in.read()==0) {state=3; cout << " Now state == 3"<<endl;}  if(register\_in.read()==1) {state=2; cout << " Now state == 2"<<endl;}  if((register\_in.read()!=1)&(register\_in.read()!=0)) cout << " Now state == 3"<<endl;  }  wait();  }  }  void statemashine::data\_out\_method() {  if((state.read()==0)||(state.read()==2)) register\_out = 1;  else register\_out = 2;  } |

Листинг 1.3. Файл testbench.cpp

|  |
| --- |
| //-----------------------------------------------------  // Testbench for the register ---------------->  //-----------------------------------------------------  #include "systemc.h"  #include "statemashine.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> > register\_in;  sc\_signal<sc\_uint<2> > register\_out;  int i = 0;  // Connect the DUT  statemashine test\_statemashine("test\_statemashine");  test\_statemashine.clock(clock);  test\_statemashine.reset(reset);  test\_statemashine.register\_in(register\_in);  test\_statemashine.register\_out(register\_out);  // Open VCD file  sc\_trace\_file \*wf = sc\_create\_vcd\_trace\_file("statemashine\_waveform");  // Dump the desired signals  sc\_trace(wf, clock, "clock");  sc\_trace(wf, reset, "reset");  sc\_trace(wf, register\_in, "din");  sc\_trace(wf, register\_out, "dout");  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;    assert(register\_out.read() == 1);    //from state 0 to state 1 signal 2  register\_in = 2;  sc\_start(4, SC\_NS);  assert(register\_out.read() == 2);  //from state 1 to state 1 signal 2  sc\_start(4, SC\_NS);  assert(register\_out.read() == 2);  //from state 1 to state 0 signal 0  register\_in = 0;  sc\_start(4, SC\_NS);  assert(register\_out.read() == 1);  //from state 0 to state 1 signal 1  register\_in = 1;  sc\_start(4, SC\_NS);  assert(register\_out.read() == 2);  //from state 1 to state 2 signal 1  //register\_in = 1;  sc\_start(4, SC\_NS);  assert(register\_out.read() == 1);  //from state 2 to state 0 signal 2  register\_in = 2;  sc\_start(4, SC\_NS);  assert(register\_out.read() == 1);  //from state 0 to state 3 signal 0  register\_in = 0;  sc\_start(4, SC\_NS);  assert(register\_out.read() == 2);  //from state 3 to state 3 signal 0  //register\_in = 0;  sc\_start(4, SC\_NS);  assert(register\_out.read() == 2);  //from state 3 to state 2 signal 1  register\_in = 1;  sc\_start(4, SC\_NS);  assert(register\_out.read() == 1);  reset = 1; // Assert the reset  cout << "@" << sc\_time\_stamp() << " Asserting reset\n" << endl;  sc\_start(5, SC\_NS);  assert(register\_out.read() == 1);  cout << "@" << sc\_time\_stamp() << " Terminating simulation\n" << endl;  sc\_close\_vcd\_trace\_file(wf);  return 0; // Terminate simulation  } |

Для проверки корректности работы конечного автомата, проверки соответствия работы графу переходов конечного автомата, был создан тест, включающий все возможные преходы из состояния в состояние. Тест выглядит следующим образом:

Reset – 0(10) – 1(10) – 1(00) – 0(01) – 1(01) – 2(10) – 0(00) – 3(00) – 3(01) – reset

**Полученные результаты моделирования:**

Сообщения из консоли:

|  |
| --- |
| SystemC 2.3.1-Accellera --- Feb 18 2017 01:17:28  Copyright (c) 1996-2014 by all Contributors,  ALL RIGHTS RESERVED  Executing new  @0 s Asserting reset  Info: (I702) default timescale unit used for tracing: 1 ps (statemashine\_waveform.vcd)  @6 ns De-Asserting reset  \*\*\*\*\*\*  Previous state == 0 Input signal == 2  Now state == 1  \*\*\*\*\*\*  Previous state == 1 Input signal == 2  Now state == 1  \*\*\*\*\*\*  Previous state == 1 Input signal == 0  Now state == 0  \*\*\*\*\*\*  Previous state == 0 Input signal == 1  Now state == 1  \*\*\*\*\*\*  Previous state == 1 Input signal == 1  Now state == 2  \*\*\*\*\*\*  Previous state == 2 Input signal == 2  Now state == 0  \*\*\*\*\*\*  Previous state == 0 Input signal == 0  Now state == 3  \*\*\*\*\*\*  Previous state == 3 Input signal == 0  Now state == 3  \*\*\*\*\*\*  Previous state == 3 Input signal == 1  Now state == 2  @42 ns Asserting reset  @47 ns Terminating simulation  Для продолжения нажмите любую клавишу . . . |

Результаты моделирования:

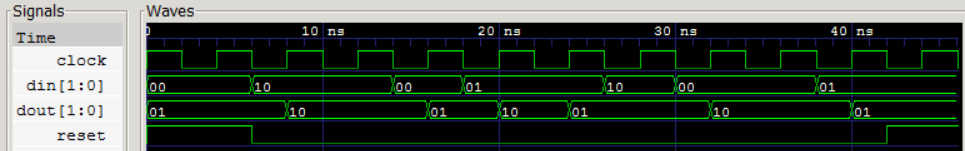


Рис 1.2. Результаты моделирования конечного автомата

Результаты моделирования соответствуют ожидаемым результатам, устройство работает корректно.