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 №3**

**Знакомство с описанием параметризованных устройств на языке SystemC**

**Программа работы:**

1. Создать потактовое описание FIFO, работающего с 8-разрядными словами. Глубина FIFO должна задаваться в конструкторе.

Обязательные входы:

- clk

- sreset\_n

- data\_in

- push

- pop

Обязательные выходы:

- data\_out

- empty

- full

2. Создать на основе класса FIFO шаблон класса FIFOParam, в котором параметром выступает тип данных.

3. Унаследовать от созданного шаблона класса FIFOParam, шаблон класса FIFOParamExtended с дополнительными выходами:

- almost\_empty

- almost\_full

Для всех трёх описаний необходимо написать соответствующие тесты.

**Выполнение работы.**

1. Создать потактовое описание FIFO, работающего с 8-разрядными словами. Глубина FIFO должна задаваться в конструкторе.

Листинг 1.1. Файл FIFO.h

|  |
| --- |
| #include "systemc.h"  #ifndef DESIGN\_H  #define DESIGN\_H  SC\_MODULE(FIFO) {  sc\_in\_clk clk; // Clock input of the design  sc\_in<bool> sreset\_n; // active high, synchronous Reset input  sc\_in<sc\_uint<8>> data\_in; // 8 bit vector input  sc\_in<bool> push;  sc\_in<bool> pop;  sc\_out<sc\_uint<8>> data\_out; // 8 bit vector output  sc\_out<bool> empty;  sc\_out<bool> full;  //------------Local Variables Here---------------------  //sc\_signal<sc\_uint<2> > state;  int \*data;  int size;  int pop\_pointer;  int push\_pointer;  bool flag\_push\_on\_pop;  void write\_FIFO();  void read\_FIFO();  void reset();  // Constructor  SC\_HAS\_PROCESS(FIFO);  FIFO(sc\_module\_name name, int max) :  clk("clk"),  sreset\_n("sreset\_n"),  data\_in("data\_in"),  data\_out("data\_out") {  data=new int [max];  size=max;  flag\_push\_on\_pop=false;  push\_pointer=0;  pop\_pointer=0;  //reset  SC\_CTHREAD(reset, clk.pos());  reset\_signal\_is(sreset\_n, false);  //write\_FIFO  SC\_CTHREAD(write\_FIFO, clk.pos());  reset\_signal\_is(sreset\_n, false);  //read\_FIFO  SC\_CTHREAD(read\_FIFO, clk.pos());  reset\_signal\_is(sreset\_n, false);  } // End of Constructor  }; // End of Module  #endif /\* DESIGN\_H \*/ |

Листинг 1.2. Файл FIFO.cpp

|  |
| --- |
| #include "FIFO.h"  void FIFO::write\_FIFO() {  wait();  while (true) {  if (push==1) {  if(flag\_push\_on\_pop==false)  {  data[push\_pointer]=data\_in.read();  cout<<sc\_time\_stamp()<<" Write data = "<<data\_in.read()<<"; push\_pointer change from "<<push\_pointer;  push\_pointer+=1;  empty=0;  if(push\_pointer==size) push\_pointer=0;  cout<< " to "<< push\_pointer<<endl;  if(push\_pointer==pop\_pointer)  {  flag\_push\_on\_pop=true;  full=1;  }  }  }  wait();  }  }  void FIFO::read\_FIFO() {  wait();  while (true) {  if (pop==1){  if (flag\_push\_on\_pop==false)  {  if(pop\_pointer!=push\_pointer)  {  data\_out=data[pop\_pointer];  cout<<sc\_time\_stamp()<<" Read data = "<<data[pop\_pointer]<<"; pop\_pointer change from "<<pop\_pointer;  pop\_pointer+=1;  if (pop\_pointer==size) pop\_pointer=0;  cout<< " to "<< pop\_pointer<<endl;  if (pop\_pointer==push\_pointer) empty=1;  }  }  //flag\_push\_on\_pop=true  else  {  data\_out=data[pop\_pointer];  pop\_pointer+=1;  if (pop\_pointer==size) pop\_pointer=0;  flag\_push\_on\_pop=false;  full=0;  }  }  wait();  }  }  void FIFO::reset(){  flag\_push\_on\_pop=false;  push\_pointer=0;  pop\_pointer=0;  data\_out=0;  empty=1;  full=0;  wait();  while (true){  wait();  }  } |

Листинг 1.3. Файл testbench.cpp

|  |
| --- |
| //-----------------------------------------------------  // Testbench for the FIFO ---------------->  //-----------------------------------------------------  #include "systemc.h"  #include "FIFO.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\_core::sc\_report\_handler::set\_actions( "/IEEE\_Std\_1666/deprecated", sc\_core::SC\_DO\_NOTHING );  sc\_clock clk("clk", 4, SC\_NS);  sc\_signal<bool> sreset\_n;  sc\_signal<sc\_uint<8> > data\_in;  sc\_signal<bool> push;  sc\_signal<bool> pop;  sc\_signal<sc\_uint<8> > data\_out;  sc\_signal<bool> empty;  sc\_signal<bool> full;  // Connect the DUT  FIFO test\_FIFO("test\_FIFO", 2);  test\_FIFO.clk(clk);  test\_FIFO.sreset\_n(sreset\_n);  test\_FIFO.data\_in(data\_in);  test\_FIFO.push(push);  test\_FIFO.pop(pop);  test\_FIFO.data\_out(data\_out);  test\_FIFO.empty(empty);  test\_FIFO.full(full);  // Open VCD file  sc\_trace\_file \*wf = sc\_create\_vcd\_trace\_file("FIFO\_waveform");  // Dump the desired signals  sc\_trace(wf, clk, "clk");  sc\_trace(wf, sreset\_n, "sreset\_n");  sc\_trace(wf, data\_in, "data\_in");  sc\_trace(wf, push, "push");  sc\_trace(wf, pop, "pop");  sc\_trace(wf, data\_out, "data\_out");  sc\_trace(wf, empty, "empty");  sc\_trace(wf, full, "full");  sreset\_n = 0; // Assert the reset  push=0;  pop=0;  cout << "@" << sc\_time\_stamp() << " Asserting reset\n" << endl;  sc\_start(6, SC\_NS);  //6ns  assert(empty.read() == 1);  assert(full.read() == 0);  sreset\_n = 1; // De-assert the reset  cout << "@" << sc\_time\_stamp() << " De-Asserting reset\n" << endl;  data\_in=3;  push=1;  sc\_start(4, SC\_NS);  //10ns  assert(empty.read() == 0);  push=0;  data\_in=0;  pop=1;  sc\_start(4, SC\_NS);  //14ns  assert(empty.read() == 1);  assert(data\_out.read() == 3);  data\_in=4;  pop=0;  push=1;  sc\_start(4, SC\_NS);  //18ns  assert(empty.read() == 0);  data\_in=5;  sc\_start(4, SC\_NS);  //22ns  assert(full.read() == 1);    data\_in=6;  sc\_start(4, SC\_NS);  //26ns  assert(full.read() == 1);  push=0;  pop=1;  sc\_start(4, SC\_NS);  //30ns  assert(full.read() == 0);  assert(data\_out.read() == 4);  sc\_start(4, SC\_NS);  //34ns  assert(full.read() == 0);  assert(data\_out.read() == 5);  sc\_start(4, SC\_NS);  //38ns  assert(full.read() == 0);  assert(data\_out.read() == 5);  sreset\_n = 0;  sc\_start(17, SC\_NS);  cout << "@" << sc\_time\_stamp() << " Asserting reset\n" << endl;  cout << "@" << sc\_time\_stamp() << " Terminating simulation\n" << endl;  sc\_close\_vcd\_trace\_file(wf);  return 0; // Terminate simulation  } |

Для проверки корректности работы FIFO был создан тест.

**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  @0 s Asserting reset  Info: (I702) default timescale unit used for tracing: 1 ps (FIFO\_waveform.vcd)  @6 ns De-Asserting reset  8 ns Write data = 3; push\_pointer change from 0 to 1  12 ns Read data = 3; pop\_pointer change from 0 to 1  16 ns Write data = 4; push\_pointer change from 1 to 0  20 ns Write data = 5; push\_pointer change from 0 to 1  32 ns Read data = 5; pop\_pointer change from 0 to 1  @55 ns Asserting reset  @55 ns Terminating simulation  Для продолжения нажмите любую клавишу . . . |

Результаты моделирования:

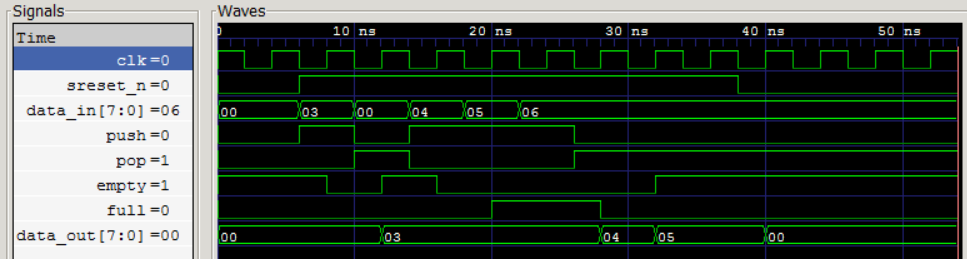


Рис 1.1. Результаты моделирования конечного автомата

Результаты моделирования соответствуют ожидаемым результатам, устройство работает корректно.

1. Создать на основе класса FIFO шаблон класса FIFOParam, в котором параметром выступает тип данных

Листинг 2.1. Файл FIFOParam.h

|  |
| --- |
| #include "systemc.h"  #ifndef DESIGN\_H  #define DESIGN\_H  template <class T>  SC\_MODULE(FIFOParam) {  sc\_in\_clk clk; // Clock input of the design  sc\_in<bool> sreset\_n; // active high, synchronous Reset input  sc\_in<sc\_uint<8>> data\_in; // 8 bit vector input  sc\_in<bool> push;  sc\_in<bool> pop;  sc\_out<sc\_uint<8>> data\_out; // 8 bit vector output  sc\_out<bool> empty;  sc\_out<bool> full;  //------------Local Variables Here---------------------  //sc\_signal<sc\_uint<2> > state;  T \*data;  int size;  int pop\_pointer;  int push\_pointer;  bool flag\_push\_on\_pop;  void write\_FIFO();  void read\_FIFO();  void reset();  // Constructor  SC\_HAS\_PROCESS(FIFOParam);  FIFOParam(sc\_module\_name name, int max) :  clk("clk"),  sreset\_n("sreset\_n"),  data\_in("data\_in"),  data\_out("data\_out") {  data=new T [max];  size=max;  flag\_push\_on\_pop=false;  push\_pointer=0;  pop\_pointer=0;  //reset  SC\_CTHREAD(reset, clk.pos());  reset\_signal\_is(sreset\_n, false);  //write\_FIFO  SC\_CTHREAD(write\_FIFO, clk.pos());  reset\_signal\_is(sreset\_n, false);  //read\_FIFO  SC\_CTHREAD(read\_FIFO, clk.pos());  reset\_signal\_is(sreset\_n, false);  } // End of Constructor  }; // End of Module  template <class T>  void FIFOParam<T>::write\_FIFO() {  wait();  while (true) {  if (push==1) {  if(flag\_push\_on\_pop==false)  {  data[push\_pointer]=data\_in.read();  cout<<sc\_time\_stamp()<<" Write data = "<<(T) data\_in.read()<<"; push\_pointer change from "<<push\_pointer;  push\_pointer+=1;  empty=0;  if(push\_pointer==size) push\_pointer=0;  cout<< " to "<< push\_pointer<<endl;  if(push\_pointer==pop\_pointer)  {  flag\_push\_on\_pop=true;  full=1;  }  }  }  wait();  }  }  template <class T>  void FIFOParam<T>::read\_FIFO() {  wait();  while (true) {  if (pop==1){  if (flag\_push\_on\_pop==false)  {  if(pop\_pointer!=push\_pointer)  {  data\_out=data[pop\_pointer];  cout<<sc\_time\_stamp()<<" Read data = "<<(T) data[pop\_pointer]<<"; pop\_pointer change from "<<pop\_pointer;  pop\_pointer+=1;  if (pop\_pointer==size) pop\_pointer=0;  cout<< " to "<< pop\_pointer<<endl;  if (pop\_pointer==push\_pointer) empty=1;  }  }  //flag\_push\_on\_pop=true  else  {  data\_out=data[pop\_pointer];  pop\_pointer+=1;  if (pop\_pointer==size) pop\_pointer=0;  flag\_push\_on\_pop=false;  full=0;  }  }  wait();  }  }  template <class T>  void FIFOParam<T>::reset(){  flag\_push\_on\_pop=false;  push\_pointer=0;  pop\_pointer=0;  data\_out=0;  empty=1;  full=0;  wait();  while (true){  wait();  }  }  #endif /\* DESIGN\_H \*/ |

Листинг 2.2. Файл testbench.cpp

|  |
| --- |
| //-----------------------------------------------------  // Testbench for the FIFO ---------------->  //-----------------------------------------------------  #include "systemc.h"  #include "FIFOParam.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\_core::sc\_report\_handler::set\_actions( "/IEEE\_Std\_1666/deprecated", sc\_core::SC\_DO\_NOTHING );  sc\_clock clk("clk", 4, SC\_NS);  sc\_signal<bool> sreset\_n;  sc\_signal<sc\_uint<8> > data\_in;  sc\_signal<bool> push;  sc\_signal<bool> pop;  sc\_signal<sc\_uint<8> > data\_out;  sc\_signal<bool> empty;  sc\_signal<bool> full;  // Connect the DUT  FIFOParam<int> test\_FIFO("test\_FIFO", 2);  test\_FIFO.clk(clk);  test\_FIFO.sreset\_n(sreset\_n);  test\_FIFO.data\_in(data\_in);  test\_FIFO.push(push);  test\_FIFO.pop(pop);  test\_FIFO.data\_out(data\_out);  test\_FIFO.empty(empty);  test\_FIFO.full(full);  // Open VCD file  sc\_trace\_file \*wf = sc\_create\_vcd\_trace\_file("FIFO\_waveform");  // Dump the desired signals  sc\_trace(wf, clk, "clk");  sc\_trace(wf, sreset\_n, "sreset\_n");  sc\_trace(wf, data\_in, "data\_in");  sc\_trace(wf, push, "push");  sc\_trace(wf, pop, "pop");  sc\_trace(wf, data\_out, "data\_out");  sc\_trace(wf, empty, "empty");  sc\_trace(wf, full, "full");  sreset\_n = 0; // Assert the reset  push=0;  pop=0;  cout << "@" << sc\_time\_stamp() << " Asserting reset\n" << endl;  sc\_start(6, SC\_NS);  //6ns  assert(empty.read() == 1);  assert(full.read() == 0);  sreset\_n = 1; // De-assert the reset  cout << "@" << sc\_time\_stamp() << " De-Asserting reset\n" << endl;  data\_in=3;  push=1;  sc\_start(4, SC\_NS);  //10ns  assert(empty.read() == 0);  push=0;  data\_in=0;  pop=1;  sc\_start(4, SC\_NS);  //14ns  assert(empty.read() == 1);  assert(data\_out.read() == 3);  data\_in=4;  pop=0;  push=1;  sc\_start(4, SC\_NS);  //18ns  assert(empty.read() == 0);  data\_in=5;  sc\_start(4, SC\_NS);  //22ns  assert(full.read() == 1);    data\_in=6;  sc\_start(4, SC\_NS);  //26ns  assert(full.read() == 1);  push=0;  pop=1;  sc\_start(4, SC\_NS);  //30ns  assert(full.read() == 0);  assert(data\_out.read() == 4);  sc\_start(4, SC\_NS);  //34ns  assert(full.read() == 0);  assert(data\_out.read() == 5);  sc\_start(4, SC\_NS);  //38ns  assert(full.read() == 0);  assert(data\_out.read() == 5);  sreset\_n = 0;  sc\_start(17, SC\_NS);  cout << "@" << sc\_time\_stamp() << " Asserting reset\n" << endl;  cout << "@" << sc\_time\_stamp() << " Terminating simulation\n" << endl;  sc\_close\_vcd\_trace\_file(wf);  return 0; // Terminate simulation  } |

Для проверки корректности работы FIFO был создан тест.

**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  @0 s Asserting reset  Info: (I702) default timescale unit used for tracing: 1 ps (FIFO\_waveform.vcd)  @6 ns De-Asserting reset  8 ns Write data = 3; push\_pointer change from 0 to 1  12 ns Read data = 3; pop\_pointer change from 0 to 1  16 ns Write data = 4; push\_pointer change from 1 to 0  20 ns Write data = 5; push\_pointer change from 0 to 1  32 ns Read data = 5; pop\_pointer change from 0 to 1  @55 ns Asserting reset  @55 ns Terminating simulation  Для продолжения нажмите любую клавишу . . . |

Результаты моделирования:

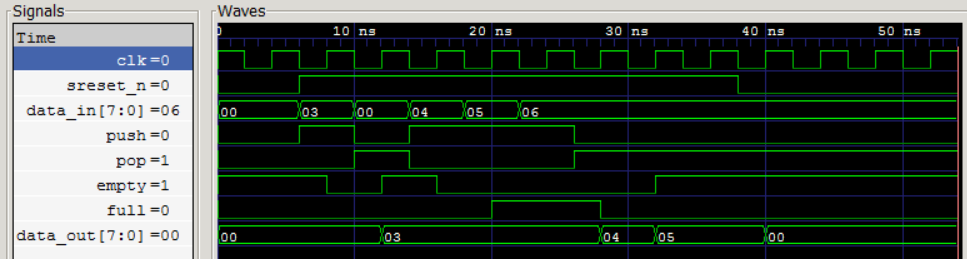


Рис 2.1. Результаты моделирования конечного автомата

Результаты моделирования соответствуют ожидаемым результатам, устройство работает корректно.

3. Унаследовать от созданного шаблона класса FIFOParam, шаблон класса FIFOParamExtended с дополнительными выходами:

- almost\_empty

- almost\_full