Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчет по лабораторной работе №1**

по дисциплине «SystemC»

Проектирование сдвигающего регистра и счетчика

Выполнил

студент гр. 13541/2

Никитенко А.П.

Преподаватели

Петров М.А.

Мамутова О.В.

**Цель работы**: разработка счетчика и сдвигающего регистра на языке SystemC согласно варианту и проведение тестирования.

**Ход работы.**

Основные входы/выходы счетчика:

* clk
* areset
* sreset\_n
* dout

Основные входы/выходы сдвигающего регистра:

* clk
* areset
* sreset
* cin
* cout
* dout

Таблица дополнительных функций устройств (Вариант 4).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | counter | | | shift\_register | | |
| ena | load | up/down | ena | load | left/right |
| **4** |  | **+** |  |  |  | **+** |

Реализация сдвигающего регистра и счетчика представлена в файлах shift\_reg.h, shift\_reg.cpp(Приложение 1) и counter.h, counter.cpp соответственно(Приложение 2).

Результаты тестирования для обоих устройств были сгенерированы в .vcd файл и совпадают с ожидаемыми результатами.

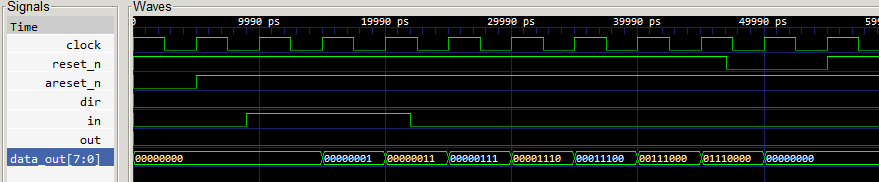
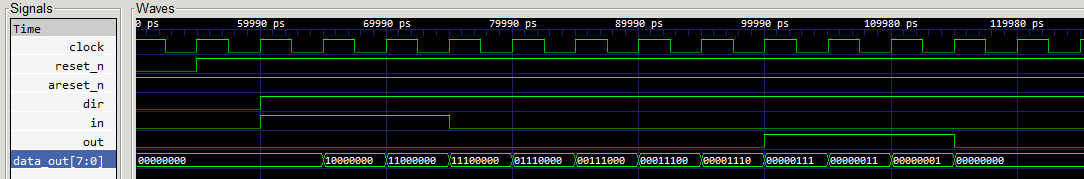
 

Рис. 2. Результат тестирования сдвигающего регистра

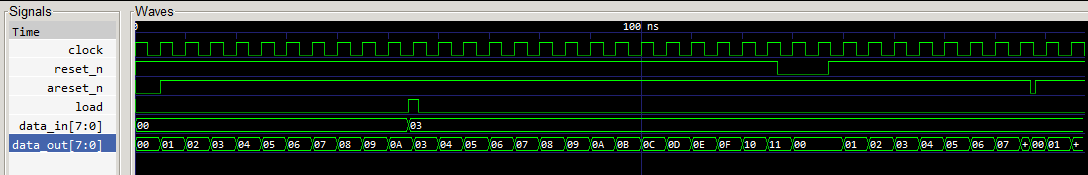


Рис.3. Результат тестирования счетчика

Приложение 1

**shift\_reg.h**

#ifndef SHIFT\_REG\_H

#define SHIFT\_REG\_H

#include <systemc.h>

SC\_MODULE(shift\_reg) {

sc\_in\_clk clock; // Clock input of the design

sc\_in<bool> reset\_n; // negative high, synchronous Reset input

sc\_in<bool> areset\_n; //asynchronous reser, negative high

sc\_in<bool> dir;

sc\_in<bool> in; //bit input

sc\_out<sc\_uint<8> > data\_out; // 8 bit vector input

sc\_out<bool> out; // output

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

sc\_signal<sc\_uint<8>> data;

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

// Below function implements actual register logic

void shift();

//Constructor

SC\_CTOR(shift\_reg) :

clock("clock"),

reset\_n("reset\_n"),

areset\_n("areset\_n"),

dir("dir"),

data\_out("data\_out"),

out("out") {

SC\_CTHREAD(shift, clock.pos());

reset\_signal\_is(reset\_n, false);

async\_reset\_signal\_is(areset\_n, false);

} // End of Constructor

}; // End of Module

#endif // SHIFT\_REG\_H

**shift\_reg.cpp**

#include "shift\_reg.h"

void shift\_reg::shift() {

// We check if reset is active

data = 0;

data\_out = 0;

out = 0;

wait();

while (true) {

if(dir==false){

data.write((data.read().range(6, 0), in.read()));

data\_out.write(data.read());

out = (bool)data.read().bit(7);

}

else {

data.write((in.read(),data.read().range(7, 1)));

data\_out.write(data.read());

out = (bool)data.read().bit(0);

}

cout << "@" << sc\_time\_stamp() << " :: Have stored " << data\_out.read() << endl;

wait();

}

}

**testbench.cpp**

#include "systemc.h"

#include "shift\_reg.h"

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

{

sc\_clock clock("clock", 5, SC\_NS);

sc\_signal<bool> reset\_n;

sc\_signal<bool> areset\_n;

sc\_signal<bool> dir;

sc\_signal<sc\_uint<8>> data;

sc\_signal<bool> in;

sc\_signal<sc\_uint<8>> data\_out;

sc\_signal<bool> out;

// Connect the DUT

shift\_reg shift\_reg\_test("shift\_reg");

shift\_reg\_test.clock(clock);

shift\_reg\_test.reset\_n(reset\_n);

shift\_reg\_test.areset\_n(areset\_n);

shift\_reg\_test.dir(dir);

shift\_reg\_test.in(in);

shift\_reg\_test.data\_out(data\_out);

shift\_reg\_test.out(out);

// Open VCD file

sc\_trace\_file \*wf = sc\_create\_vcd\_trace\_file("shift\_reg\_waveform");

// Dump the desired signals

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

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

sc\_trace(wf, areset\_n, "areset\_n");

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

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

sc\_trace(wf, data\_out, "data\_out");

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

dir = 0;

in = 0;

reset\_n = 1;

areset\_n = 0;

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

sc\_start(5, SC\_NS);

areset\_n = 1;

sc\_start(4, SC\_NS);

in = 1;

sc\_start(1, SC\_NS);

sc\_start(5, SC\_NS);

assert(data\_out.read() == 0);

sc\_start(5, SC\_NS);

assert(data\_out.read() == 1);

sc\_start(2, SC\_NS);

in = 0;

sc\_start(3, SC\_NS);

assert(data\_out.read() == 3);

sc\_start(5, SC\_NS);

assert(data\_out.read() == 7);

sc\_start(5, SC\_NS);

assert(data\_out.read() == 0x0e);

sc\_start(5, SC\_NS);

assert(data\_out.read() == 0x1c);

sc\_start(5, SC\_NS);

assert(data\_out.read() == 0x38);

sc\_start(2, SC\_NS);

int tmp = data\_out.read();

cout << endl << "tmp = " << tmp << endl;

reset\_n = 0;

sc\_start(1, SC\_NS);

assert(data\_out.read() == tmp);

sc\_start(4, SC\_NS);

assert(data\_out.read() == 0);

sc\_start(3, SC\_NS);

reset\_n = 1;

sc\_start(5,SC\_NS);

dir=1;

in=1;

sc\_start(5,SC\_NS);

assert(data\_out.read()==0x00);

sc\_start(5,SC\_NS);

assert(data\_out.read()==0x80);

sc\_start(5,SC\_NS);

assert(data\_out.read()==0xc0);

in=0;

sc\_start(5,SC\_NS);

assert(data\_out.read()==0xe0);

sc\_start(5,SC\_NS);

assert(data\_out.read()==0x70);

for(int i = 0; i<15; i++) sc\_start(5, SC\_NS);

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

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

return 0;

}

Приложение 2

**counter.h**

#include "systemc.h"

#ifndef COUNTER\_H

#define COUNTER\_H

SC\_MODULE(counter) {

sc\_in\_clk clock; // Clock input of the design

sc\_in<bool> reset\_n; // negative high, synchronous Reset input

sc\_in<bool> areset\_n; //asynchronous reser, negative high

sc\_in<bool> load;

sc\_in<sc\_uint<8> > data\_in; // 8 bit vector input

sc\_out<sc\_uint<8> > data\_out; // 8 bit vector output

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

sc\_uint<8> reg;

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

// Below function implements actual register logic

void count();

//Constructor

SC\_CTOR(counter) :

clock("clock"),

reset\_n("reset\_n"),

areset\_n("areset\_n"),

load("load"),

data\_in("data\_in"),

data\_out("data\_out") {

cout << "Start Executing" << endl;

SC\_CTHREAD(count, clock.pos());

reset\_signal\_is(reset\_n, false);

async\_reset\_signal\_is(areset\_n, false);

} // End of Constructor

}; // End of Module

#endif

#include "counter.h"

void counter::count() {

// We check if reset is active

reg = 0;

data\_out.write(reg);

wait();

while (true) {

if(load) reg = data\_in.read();

else reg +=1;

cout << "@" << sc\_time\_stamp() << " :: Have stored " << reg << endl;

data\_out.write(reg);

wait();

}

}

**testbench.cpp**

#include "systemc.h"

#include "counter.h"

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

{

sc\_clock clock("clock", 5, SC\_NS);

sc\_signal<bool> reset\_n;

sc\_signal<bool> areset\_n;

sc\_signal<bool> load;

sc\_signal<sc\_uint<8>> data\_in;

sc\_signal<sc\_uint<8>> data\_out;

// Connect the DUT

counter test\_counter("test\_counter");

test\_counter.clock(clock);

test\_counter.reset\_n(reset\_n);

test\_counter.areset\_n(areset\_n);

test\_counter.load(load);

test\_counter.data\_in(data\_in);

test\_counter.data\_out(data\_out);

// Open VCD file

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

// Dump the desired signals

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

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

sc\_trace(wf, areset\_n, "areset\_n");

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

sc\_trace(wf, data\_in, "data\_in");

sc\_trace(wf, data\_out, "data\_out");

data\_in = 0;

load = 0;

reset\_n = 1;

areset\_n = 0;

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

sc\_start(5, SC\_NS);

areset\_n = 1;

for (int i=0; i<9; i++){

sc\_start(5, SC\_NS);

assert(data\_out.read() == i+1);

}

sc\_start(4, SC\_NS);

data\_in = 3;

load = 1;

sc\_start(2, SC\_NS);

load = 0;

sc\_start(2, SC\_NS);

assert(data\_out.read() == 3);

sc\_start(2, SC\_NS);

for (int i=0; i<13; i++){

sc\_start(5, SC\_NS);

assert(data\_out.read() == i+4);

}

sc\_start(2, SC\_NS);

int tmp = data\_out.read();

cout << endl << "tmp = " << tmp << endl;

reset\_n = 0;

sc\_start(5, SC\_NS);

assert(data\_out.read() == 0);

sc\_start(5, SC\_NS);

reset\_n = 1;

for (int i=0; i<7; i++){

sc\_start(5, SC\_NS);

}

sc\_start(5, SC\_NS);

areset\_n = 0;

sc\_start(1, SC\_NS);

assert(data\_out.read() == 0);

areset\_n = 1;

sc\_start(10, SC\_NS);

cout << endl;

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

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

return 0;

}