

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: Знакомство с описанием синхронных устройств на RTL уровне на языке SystemC.

Выполнил студент гр. 13541/2

(подпись)

В.Е. Бушин

Руководитель

(подпись)

О.В. Мамутова

“ _ ” _____ 2017 г.

Санкт-Петербург

2017

Программа работы:

- 1) Скопировать в локальную папку проект с примером регистра.
- 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 собственные устройства: счетчик и сдвигающий регистр, - с дополнительными функциями по индивидуальному заданию.
- 4) Создать тесты с самопроверкой для всех основных и дополнительных функций разработанных устройств.

Основные входы/выходы счетчика:

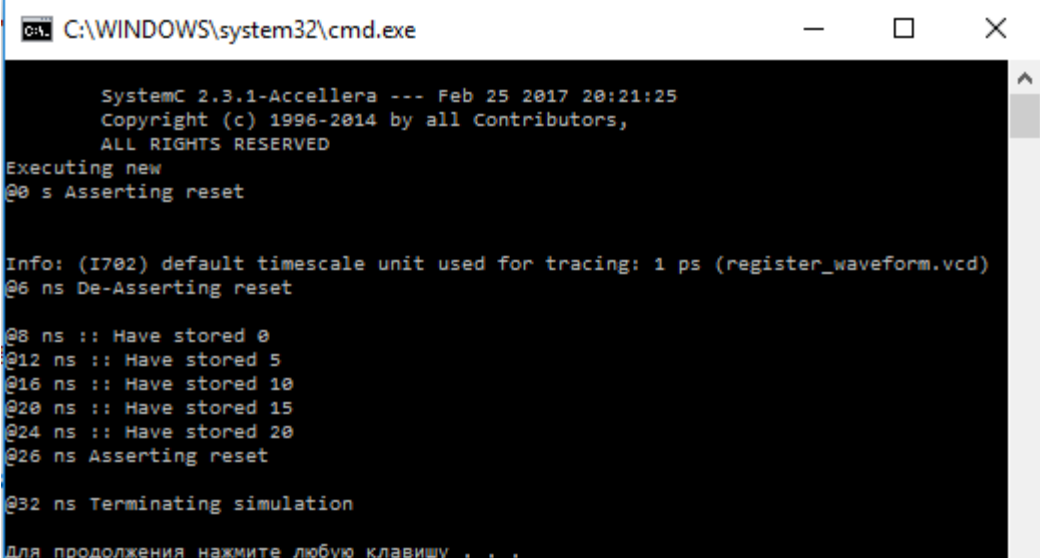
- clk
- areset
- sreset_n
- dout

Основные входы/выходы сдвигающего регистра:

- clk
- areset
- sreset
- cin
- cout
- dout

Выполнение работы

- 1) Создан новый проект на основе примера.
- 2) Успешно выполнена компиляция проекта. Созданное приложение запущено и получены следующие результаты:



```
C:\WINDOWS\system32\cmd.exe

SystemC 2.3.1-Accellera --- Feb 25 2017 20:21:25
Copyright (c) 1996-2014 by all Contributors,
ALL RIGHTS RESERVED
Executing new
@0 s Asserting reset

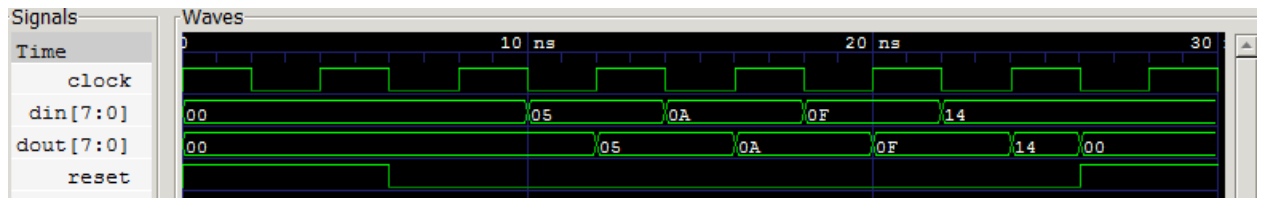
Info: (I702) default timescale unit used for tracing: 1 ps (register_waveform.vcd)
@6 ns De-Asserting reset

@8 ns :: Have stored 0
@12 ns :: Have stored 5
@16 ns :: Have stored 10
@20 ns :: Have stored 15
@24 ns :: Have stored 20
@26 ns Asserting reset

@32 ns Terminating simulation

Для продолжения нажмите любую клавишу . . .
```

Открыв сгенерированный vcd файл в GTKWave, мы увидим корректные результаты работы устройства:



- 3) Был разработан счётчик с сигналом разрешения счёта. Реализация счётчика находится в файлах counter.h и counter.cpp.

Содержимое файла counter.h:

```
#ifndef COUNTER_H
#define COUNTER_H

#include "systemc.h"

SC_MODULE(counter) {
    sc_in_clk clock; // Clock input of the design
    sc_in<bool> reset;
    sc_in<bool> sreset; // active high, synchronous Reset input
    sc_in<bool> ena; // count enable signal
    sc_out<sc_uint<8> > counter_out; // 8 bit vector output

    //-----Local Variables Here-----
    sc_uint<8> count;

    void do_count();

    //constructor
    SC_CTOR(counter) :
        clock("clock"),
        reset("reset"),
        sreset("sreset"),
        ena("ena"),
        counter_out("counter_out")
    {
        cout << "Executing new" << endl;
        SC_CTHREAD(do_count, clock.pos());
        async_reset_signal_is(reset, true);
        reset_signal_is(sreset, true);
    }
};

#endif /* COUNTER_H */
```

Содержимое файла counter.cpp:

```
#include "counter.h"

void counter::do_count() {
    count = 0;
    counter_out.write(count);
    wait();
    while (true) {
        if (ena.read() == 1) {
            count = count + 1;
            counter_out.write(count);
        }
        wait();
    }
}
```

Также был разработан сдвигающий регистр с параллельной загрузкой. Его реализация находится в файлах shift.h и shift.cpp.

Содержимое файла shift.h:

```
#ifndef SHIFT_H
#define      SHIFT_H

#include "systemc.h"

SC_MODULE(shift_register) {
    sc_in_clk clock; // Clock input of the design
    sc_in<bool> reset;
    sc_in<bool> sreset; // active high, synchronous Reset input
    sc_in<bool> load; //load signal
    sc_in<sc_uint<8> > data_in;
    sc_in<bool> c_in;
    sc_out<bool> c_out;
    sc_out<sc_uint<8> > register_out;

    sc_uint<8> reg;

    void shifting();

    SC_CTOR(shift_register) :
        clock("clock"),
        reset("reset"),
        sreset("sreset"),
        load("load"),
        data_in("data_in"),
        register_out("register_out"),
        c_out("c_out"),
        c_in("c_in")
    {
        cout << "Executing new" << endl;
        SC_CTHREAD(shifting, clock.pos());
        async_reset_signal_is(reset, true);
        reset_signal_is(sreset, true);
    }
};

#endif /* SHIFT_H */
```

Содержимое файла shift.cpp:

```
#include "shift.h"

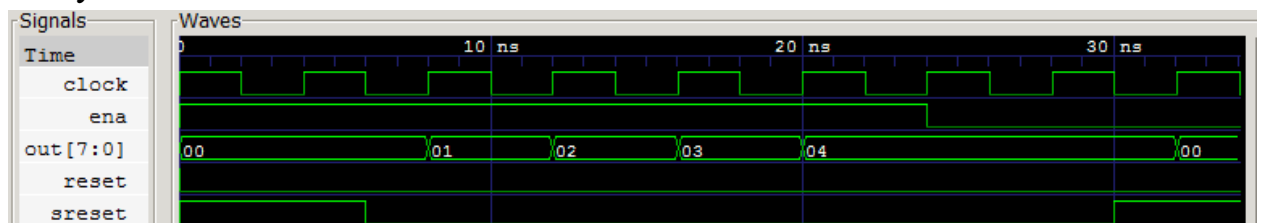
void shift_register::shifting()
{
    reg = 0;
    register_out.write(reg);
    c_out = reg.bit(0);
    wait();
    while(true)
    {
        if(load.read()==1)
            reg = data_in.read();
        else {
            c_out = reg.bit(0);
            reg = (c_in, reg(7,1));
        }
        register_out.write(reg);
        wait();
    }
}
```

```
}  
}
```

4) Для счётчика был создан следующий тест:

```
int sc_main(int argc, char* argv[]) {  
    sc_clock clock("clock", 4, SC_NS);  
    sc_signal<bool> reset;  
    sc_signal<bool> sreset;  
    sc_signal<bool> ena;  
    sc_signal<sc_uint<8> > counter_out;  
    int i = 0;  
  
    counter test_count("test_count");  
    test_count.clock(clock);  
    test_count.reset(reset);  
    test_count.sreset(sreset);  
    test_count.ena(ena);  
    test_count.counter_out(counter_out);  
  
    // Open VCD file  
    sc_trace_file *wf = sc_create_vcd_trace_file("counter");  
    // Dump the desired signals  
    sc_trace(wf, clock, "clock");  
    sc_trace(wf, reset, "reset");  
    sc_trace(wf, sreset, "sreset");  
    sc_trace(wf, ena, "ena");  
    sc_trace(wf, counter_out, "out");  
  
    sreset = 1;  
    ena = 1;  
    sc_start(6, SC_NS);  
  
    sreset = 0;  
    sc_start(18, SC_NS);  
  
    ena = 0;  
    sc_start(6, SC_NS);  
  
    sreset = 1;  
    sc_start(6, SC_NS);  
    return 0;  
}
```

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



Для сдвигающего регистра также был создан тест:

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

```

sc_signal<sc_uint<8> > data_in;
sc_signal<sc_uint<8> > register_out;
sc_signal<bool> c_out;
sc_signal<bool> c_in;

shift_register test_shift("test_shift");
test_shift.clock(clock);
test_shift.reset(reset);
test_shift.sreset(sreset);
test_shift.load(load);
test_shift.data_in(data_in);
test_shift.register_out(register_out);
test_shift.c_out(c_out);
test_shift.c_in(c_in);

// Open VCD file
sc_trace_file *wf = sc_create_vcd_trace_file("shift");
// Dump the desired signals
sc_trace(wf, clock, "clock");
sc_trace(wf, reset, "reset");
sc_trace(wf, sreset, "sreset");
sc_trace(wf, load, "load");
sc_trace(wf, data_in, "din");
sc_trace(wf, register_out, "out");
sc_trace(wf, c_out, "c_out");
sc_trace(wf, c_in, "c_in");

reset = 0;
sreset = 1;
load = 0;
c_in = 0;
sc_start(6, SC_NS);

sreset = 0;
load = 1;
data_in = 77;
sc_start(12, SC_NS);

load = 0;
sc_start(10, SC_NS);

c_in = 1;
sc_start(20, SC_NS);

sreset = 1;
sc_start(10, SC_NS);
sc_close_vcd_trace_file(wf);
return 0;
}

```

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

