SystemC – Modules (01A)

SystemC

Young Won Lim
01/04/2014

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Based on the following original work

[1] Aleksandar Milenkovic, 2002
CPE 626 The SystemC Language – VHDL, Verilog Designer's Guide
http://www.ece.uah.edu/~milenka/ce626-02S/lectures/cpe626-SystemC-L2.ppt

[2] Alexander de Graaf, EEMCS/ME/CAS, 2010
SystemC: an overview ET 4351
ens.ewi.tudelft.nl/Education/courses/et4351/SystemC-2010v1.pdf

[3] Joachim Gerlach, 2001
System-on-Chip Design with Systent of Computer Engineering
http://www2.cs.uni-paderborn.de/cs/ag-hardt/Forschung/Data/SystemC-Tutorial.pdf

[4] Martino Ruggiero, 2008
SystemC
polimage.polito.it/~lavagno/codes/SystemC_Lezione.pdf

[5] Deepak Kumar Tal, 1998-2012
SystemC Tutorial
http://www.asic-world.com/systemc/index.html

Module Declaration

Basic building blocks of a SystemC design

Can contain processes (→ functionality)
Can contain sub-modules (→ hierarchy)

```
SC_MODULE( module_name ) {
    // Declaration of module ports
    // Declaration of module signals
    // Declaration of processes
    // Declaration of sub-modules

    SC_CTOR( module_name ) {
        // Module constructor
        // Specification of process type and sensitivity
        // Sub-module instantiation and port mapping
    }

    // Initialization of module signals
};
```

SC_MODULE Macro

A module correspond to a C++ class

class member data	↔	ports
class member functions	↔	processes S
class constructor	↔	process generation

```
class module_name : sc_module {
.....
};
```

```
SC_MODULE( module_name ) {
.....
};
```

SC_CTOR Macro

A shorthand of writing the constructor using a macro SC_CTOR.

The constructor does the following:

- Create hierarchy (sub-modules)
- Register functions as processes with the simulation kernel
- Declare sensitivity list for processes
- Accepts one argument only, which is the module name

Modules

- The basic building block
- Encapsulate HW / SW functionality
- Module functionality is achieved

  by means of processes
- Can contain sub-modules
- Can provide private variable/signals.
- Can interface to another modules via

  ports/interfaces/channels

Macro Examples (1)

```
SC_MODULE (Adder) {
    SC_CTOR (Adder) {
        . . .
    }
}
```

```
int sc_main (int argc, char* argv[]) {
    Adder Adder_1 ("Adder_1");
    sc_start();
    return (0);
}
```

instance name

```
class Adder : public sc_module {
    public:
    Adder (sc_module_name nm) : sc_module(nm)
    {
        . . .
    }
}
```

type of the argument nm

passed to the constructor of sc_module

constructor function

Macro Examples (2)

Constructor makes sure that all the parents used by the current class have been initialized.

The name nm of type **sc_module_name** needs to be initialized

```
class Adder : public sc_module {
    public:
    Adder (sc_module_name nm) : sc_module(nm)
    {
        . . .
    }
}
```

passed to the constructor of sc_module

like string class used for a hierarchical module name, e.g. “Mult.Adder”

Macro Definitions

```
#define SC_MODULE(user_module_name) \
      struct user_module_name : sc_module


#define SC_CTOR(user_module_name) \
      typedef user_module_name SC_CURRENT_USER_MODULE; \
      user_module_name( sc_module_name )
```

```
class Adder : public sc_module {
      public:
      Adder (sc_module_name nm) : sc_module(nm)
      {
            . . .
      }
}
```

```
SC_MODULE (Adder) {
      SC_CTOR (Adder) {
            . . .
      }
}
```

Custom Constructor

```
SC_MODULE (Adder) {
    SC_CTOR (Adder) {
        . . .
    }
}
```

```
SC_MODULE (Adder) {

    SC_HAS_PROCESS (Adder);

    Adder (sc_module_name nm, int other_para) : sc_module(nm) {

        . . . // custom constructor

    }

}
```

TMP

References

[1] Aleksandar Milenkovic, 2002
CPE 626 The SystemC Language – VHDL, Verilog Designer’s Guide
http://www.ece.uah.edu/~milenka/ce626-02S/lectures/cpe626-SystemC-L2.ppt

[2] Alexander de Graaf, EEMCS/ME/CAS, 2010
SystemC: an overview ET 4351
ens.ewi.tudelft.nl/Education/courses/et4351/SystemC-2010v1.pdf

[3] Joachim Gerlach, 2001
System-on-Chip Design with Systent of Computer Engineering
http://www2.cs.uni-paderborn.de/cs/ag-hardt/Forschung/Data/SystemC-Tutorial.pdf

[4] Martino Ruggiero, 2008
SystemC
polimage.polito.it/~lavagno/codes/SystemC_Lezione.pdf

[5] Deepak Kumar Tal, 1998-2012
SystemC Tutorial
http://www.asic-world.com/systemc/index.html