- SystemC and VHDL -

Martin Radetzki
consulting@edacentrum.de
www.edacentrum.de
Mechanisms for Mixing Languages

› **Foreign Language Interface (FLI)**
  – defined in VHDL, limited use, allows calling C functions
  – not a mechanism for coupling VHDL & SystemC simulation

› **Open Model Interface (OMI) – IEEE Std. 1499**
  – hardly available, and if available, doesn’t work

› **Programming Language Interface (PLI)**
  – gives access to simulator internals; difficult to use

› **Proprietary mechanisms, e.g. from verification add-ons**
  – usually works fine, but ties you to that vendor

› **Mechanism to connect modules from different languages**
  – as easy (?) as simulating VHDL and Verilog together
  – known as language co-simulation or mixed language simulation
Definition of Terms

Co-Simulation

- Multiple simulator kernels
- Communication overhead

Mixed Language Kernel

- Single simulator kernel
- Significantly smaller overhead

SystemC / VHDL Mixed Language Simulation
Mixing SystemC and VHDL

VHDL in SystemC

- SystemC Module
  - VHDL
    - Entity / Architecture
    - Configuration

SystemC in VHDL

- VHDL Entity / Architecture
  - SystemC Module

Typical case
- SystemC testbench
- VHDL design under test

SystemC model
- VHDL RTL not yet avail.
- Higher level in VHDL
Mixed-Language Fundamentals

› **In principle, mixing VHDL and SystemC is easy**
  – Both languages have modules with ports
  – Simply instantiate a module from one language using the other language’s instantiation syntax

› **The devil is in the details**
  – The instantiated module is missing when compiling higher level
  – Languages provide different ports and data types
  – Parameter (generics) mechanisms are quite different
  – SystemC modules may use higher-level communication

› **This presentation shows how it works**
  – Sample simulator: Modelsim 6.0
SystemC in VHDL

VHDL Entity / Architecture

SystemC Module

in ports

Parameters

Generic map

(in)out ports

VHDL signals
SystemC Module To Be Instantiated

comparator.h

SC_MODULE (Comparator) {

  // ports
  sc_in<int>   in1, in2;
  sc_out<bool> out1;

  // process declaration
  void action();

  SC_CTOR (Comparator) {
    SC_METHOD (action);
    sensitive << in1
               << in2;
  }

};

comparator.cpp

#include "comparator.h"

void Comparator::action()
{
  out1 = false;
  if (in1 > in2) {
    out1 = true;
  }
}

SC_MODULE_EXPORT (Comparator);

need to „export“ every top module of a SystemC module hierarchy
Instantiation in VHDL

vgencomp Comparator generates a VHDL component

component Comparator
    port(
        in1 : in  std_logic_vector(31 downto 0);
        in2 : in  std_logic_vector(31 downto 0);
        out1 : out std_logic
    );
end component;
... -- instantiation to be written by user
  c1 : comparator port map(a, b, a_gt_b);
  c2 : comparator port map(
    in1  => b,
    in2  => c,
    out1 => b_gt_c
  );
  c3 : comparator port map(c, a, out1 => c_gt_a);
Binding by Name

› No configuration for instances of component Comparator
› Instances are connected to a module
   – that has the same name as the configuration
   – that has port names and types matching the component's ports

› SystemC is case-sensitive
   – “Comparator“ != “comparator“
› VHDL is case-insensitive
   – “Comparator“ = “comparator“

› User manual: all names must match exactly (that’s safe)
› Practice: names match regardless of case
   (but maybe not when using other tools)
Passing Parameters from VHDL to SystemC

- Currently requires use of a workaround flow
- Generation of stub entity for the SystemC module
  - Option 1: Automatic generation
  - Option 2: Manual generation
- SystemC module constructor must obtain generic values through specific function calls, to be applied in the same order as the generics appear in VHDL:

```c
int generic_int = sc_get_int_param("int_generic");
float generic_real = sc_get_real_param("real_generic");
bool generic_bool = sc_get_bool_param("bool_generic");
```

 functions exist for further types

 generic names as in VHDL stub entity
VHDL in SystemC

SystemC Module

VHDL Entity

- in ports
- (in)out ports
- Generics
- Constructor call

SystemC signals
Foreign Module Declaration

entity sorter is
  port(
      a : in std_logic_vector(31 downto 0);
      b : in std_logic_vector(31 downto 0);
      c : in std_logic_vector(31 downto 0);
      biggest : out std_logic_vector(31 downto 0) );
end;

class Sorter : public sc_foreign_module
{
  public:
    sc_in<sc_lv<32> > a;
    sc_in<sc_lv<32> > b;
    sc_in<sc_lv<32> > c;
    sc_out<sc_lv<32> > biggest;

    Sorter(sc_module_name nm, const char* hdl_name)
      : sc_foreign_module(nm, hdl_name), a("a"), b("b"), c("c"), ...
    {
      ...
    }
};
# VHDL in SystemC – Instantiation

```cpp
#include "sorter.h"
...
SC_MODULE( testbench )
{
    // signals to wire up the device under test
    sc_signal<sc_lv<32> > asig, bsig, csig, fsig;
    ...
    Sorter dut;
    ...
    SC_CTOR( testbench ) : ...,
    dut( "dut", "sorter" ) { ... 
    ...
    dut.a(asig);
    dut.b(bsig);
    dut.c(csig);
    dut.biggest(fsig);
    ...
```

**SystemC stub**

**instantiation**

**initialization with additional parameter**

**port / signal connection**
Passing Parameters from SystemC to VHDL

class Sorter : public sc_foreign_module
{
public:
    ...
    Sorter(sc_module_name nm,
            const char* hdl_name,
            int num_generics,
            const char* generic_list)
        : sc_foreign_module(nm, hdl_name,
                            num_generics, generic_list),
          a("a"), b("b"), c("c"), ...
    {}
}

Sorter dut; // class member
    ...
    // in class constructor
    const char *generic_list[3];
    generic_list[0] = strdup("int_generic=7");
    generic_list[1] = strdup("real_generic=1.5");
    generic_list[2] = strdup("bool_generic=false");
    dut = new Sorter("dut", "sorter", 3, generic_list);
## Compatible SystemC and VHDL Ports

<table>
<thead>
<tr>
<th>SystemC port</th>
<th>VHDL port</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sc_in&lt;T&gt;</code></td>
<td><code>in</code></td>
</tr>
<tr>
<td><code>sc_out&lt;T&gt;</code></td>
<td><code>out / buffer</code></td>
</tr>
<tr>
<td><code>sc_inout&lt;T&gt;</code></td>
<td><code>inout</code></td>
</tr>
<tr>
<td><code>sc_in_resolved</code></td>
<td><code>in std_logic</code></td>
</tr>
<tr>
<td><code>sc_out_resolved</code></td>
<td><code>out std_logic</code></td>
</tr>
<tr>
<td><code>sc_inout_resolved</code></td>
<td><code>inout std_logic</code></td>
</tr>
<tr>
<td><code>sc_in_rv&lt;N&gt;</code></td>
<td><code>in std_logic_vector(N-1 downto 0)</code></td>
</tr>
<tr>
<td><code>sc_out_rv&lt;N&gt;</code></td>
<td><code>out std_logic_vector(N-1 downto 0)</code></td>
</tr>
<tr>
<td><code>sc_inout_rv&lt;N&gt;</code></td>
<td><code>inout std_logic_vector( ... )</code></td>
</tr>
<tr>
<td><code>sc_in_clk</code></td>
<td><code>in boolean / bit / std_[u]logic</code></td>
</tr>
<tr>
<td><code>sc_out_clk</code></td>
<td><code>out boolean / bit / std_[u]logic</code></td>
</tr>
<tr>
<td><code>sc_inout_clk</code></td>
<td><code>inout boolean / bit / std_[u]logic</code></td>
</tr>
<tr>
<td>other port types</td>
<td>other port types are not supported on the language boundary</td>
</tr>
</tbody>
</table>
## Compatible SystemC and VHDL Data Types

<table>
<thead>
<tr>
<th>SystemC type T</th>
<th>VHDL type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool / sc_bit</td>
<td>boolean / bit / std_logic</td>
</tr>
<tr>
<td>sc_logic</td>
<td>std_logic</td>
</tr>
<tr>
<td>sc_bv&lt;N&gt;</td>
<td>bit_vector(N-1 downto 0)</td>
</tr>
<tr>
<td>sc_lv&lt;N&gt;</td>
<td>std_logic_vector(N-1 downto 0)</td>
</tr>
<tr>
<td>sc_[u]int&lt;N&gt;</td>
<td>bit_vector(N-1 downto 0)</td>
</tr>
<tr>
<td>[unsigned] char</td>
<td>std_logic_vector(N-1 downto 0)</td>
</tr>
<tr>
<td>[unsigned] int</td>
<td></td>
</tr>
<tr>
<td>[unsigned] long</td>
<td></td>
</tr>
<tr>
<td>other data types are not supported on the language boundary</td>
<td></td>
</tr>
</tbody>
</table>
### VHDL to SystemC Logic Value Mapping

<table>
<thead>
<tr>
<th>std_logic</th>
<th>sc_logic</th>
<th>sc_bit</th>
<th>bool</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>Z</td>
<td>Z</td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>true</td>
</tr>
<tr>
<td>U</td>
<td>X</td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>W</td>
<td>X</td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>–</td>
<td>X</td>
<td>0</td>
<td>false</td>
</tr>
</tbody>
</table>

No logic values are lost when mapping from SystemC to VHDL.
Conclusion

Mixed-language simulation for SystemC and VHDL ...

› works
› uses mechanisms similar to mixed VHDL / Verilog simulation
› is fairly simple as long as no generics / parameters involved
› employs some mechanisms that are not standardized

If you need consulting on whatever aspects of SystemC ...

› talk to us today
› contact consulting@edacentrum.de later