HW/SW Entwurfssprachen am Beispiel System-C

Florian Zaruba, Thomas Weber

November 11, 2014

Table of contents

Motivation

- Rapidly increasing number of embedded systems.

Motivation

- Rapidly increasing number of embedded systems.

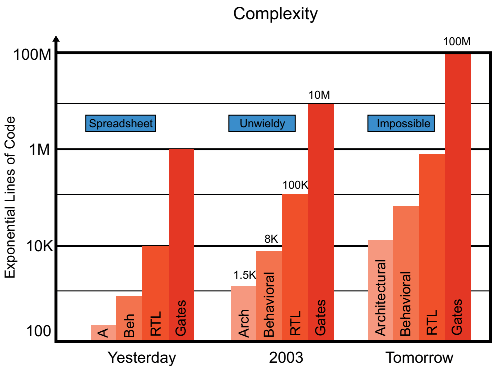

Figure: Increasing complexity [1]

System-Level Description Language (SDL)

- Specification and design at various abstraction levels
- Faster design cycles
- Seperation between communication and functionality (TLM)
- e.g.: SystemC (by accellera systems initiative), System Verilog, SpecC

What is SystemC

- Set of C++ classes and macros (Concurrency and notion of time)
- Common language for SW and HW, C++
- Enhanced productivity, support higher abstraction level (TLM)
- Many C++ libraries useful

SystemC Architecture

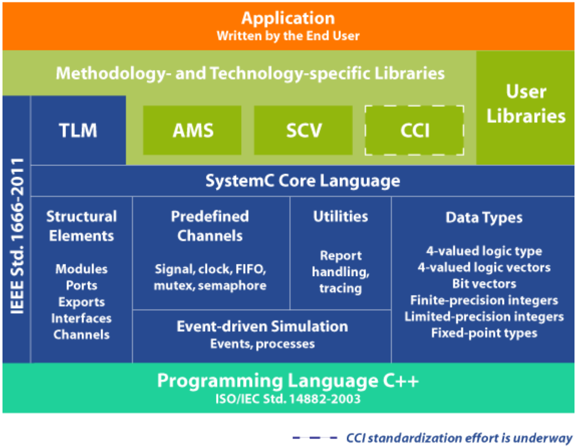

Figure: SystemC architecture [1]

[1] `http://www.accellera.org/community/systemc/about_systemc/`

History

- 27/09/1999 Open SystemC Initiative announced
- 01/03/2000 V0.91 released
- 01/02/2001 V2.0 released
- 06/06/2005 SystemC 2.1 LRM and TLM 1.0 released
- 12/07/2006 SystemC Verification Library 1.0p2
- 09/06/2008 TLM-2.0.0 library released
- 08/03/2010 SystemC AMS extension 1.0 released
- 10/11/2011 IEEE pproves the IEEE 16662011 standard for SystemC
- Currently SystemC V2.3.1, AMS 2.0, SystemC Verification 2.0

Where does it fit in?

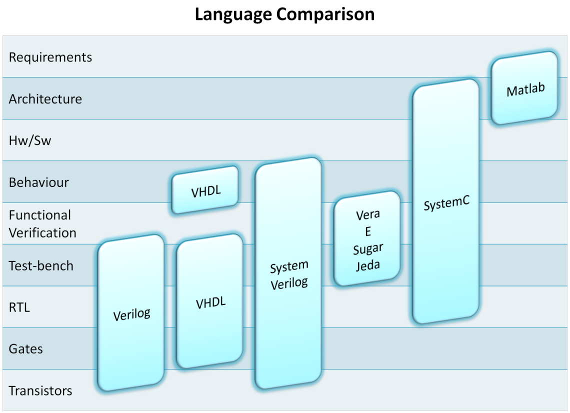

Figure: Hardware Description Languages and Abstraction Levels[2]

[2] http://www.esa.int/Our_Activities/Space_Engineering/Microelectronics/System-Level_Modeling_in_SystemC

Benefits/Drawbacks

- Benefits
  - Lots of C++ libraries available and useful
  - IEEE standard
  - Virtual prototyping
  - Verification
- Drawbacks
  - Still a C++ compiler/runtime errors
  - Debug C++, not System-C

Features of SystemC

- C
  - notion of time
  - hardware datatypes
- VHDL
  - verification
  - complexity
- SystemC Features
  - Modules
  - Processes/Threads
  - Ports and Signals
  - Time

C vs. SysC vs. VHDL Datatypes

C

```
bool
char
uintk_t
int
long
...
```

SystemC

```
sc_bit
sc_logic
sc_lv[k]
sc_uint<k>
sc_signal<...>
```

VHDL

```
bit
std_logic
std_logic_vector (k
    dowto 0)
signal
```

C vs. SysC vs. VHDL Architecture

C

```
/* c - header*/

bool sample_module(bool
        clk, bool reset,
        bool enable)
{
  //timefunction?
  if(reset == false){
    a = true;
  }
}
```

SystemC

```
SC_MODULE(sample_module){
   sc_in_clk    clk;//clock
   sc_in<sc_bit> reset;//
          signals
   sc_in<sc_bit> enable;
   sc_out<sc_bit> a; //
          output

   void a_function(){
     if(reset.read() != 0)
           {
       a.write('1');
     }

    SC_CTOR(){
      SC_METHOD(a_function
             );
      sensitive_pos << clk
             ;
      //sensitive << clk.
             pos();
    }

   }
}
```

VHDL

```
architecture sample_arch of
      sample_module is

   clk  : in std_logic;
   reset : in std_logic;
   enable: in std_logic;
   a : out std_logic;

  begin

  a_function : process(clk)
  begin
  if reset = '1' then
    a <= '1';
  end if;

end architecture ;
```

SystemC Quizz

SystemC Codesample.

```
#include "systemc.h"
SC_MODULE (first_prog) {
  sc_in_clk     clock ;      // Clock input of the design
  sc_in<bool>   reset ;      // active high, synchronous Reset input
  sc_in<bool>   enable;
  sc_out<sc_uint<4> > output;
  sc_uint<4> something;
  void incr_something () {
    if (reset.read() == 1) {
      something =  0;
      output.write(something);
    } else if (enable.read() == 1) {
      something = something + 1;
      output.write(something);
      cout<<"@" << sc_time_stamp() <<"␣::␣Incremented␣Something␣"
        <<output.read()<<endl;
    }
  }
  SC_CTOR(first_prog) {
    cout<<"Executing␣new"<<endl;
    SC_METHOD(incr_something);
    sensitive << reset;
    sensitive << clock.pos();
  }
};
```

AND-Gatter? OR-Gatter? Counter? FlipFlop?

Partitioning

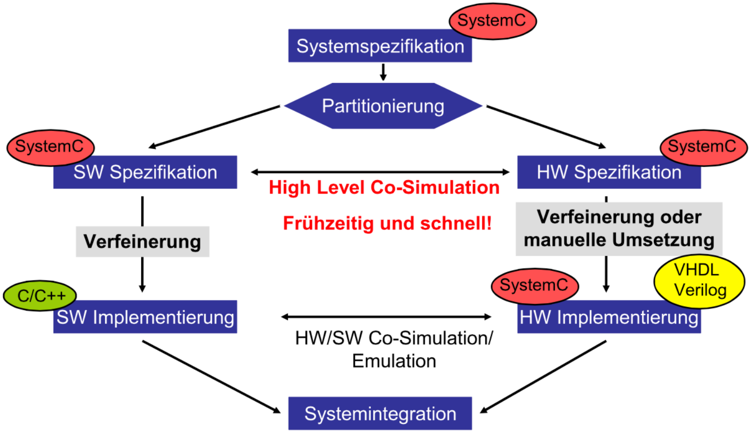

Figure: Partitioning of HW and SW **??**

Automated Partitioning

- Techniques
  - Partitioning Algorithms (Graph Theory)
    - Kernighan-Lin-Algorithm
    - Simulated annealing
  - Design Space Exploration
- Tools
  - LegUP - Open Source

SystemC Tools

SystemC to VHDL

- gcc/gdb for simualtion/debugging
- Catapult SL/LP
- Cynthesizer 5
- Modelsim - Verification
- Bambu - Open Source (C behavioral desc.)
- Mentor Graphics Catapult C/Mentor Graphics System Vision
- Cadence C-to-Silicon Compiler
- Xilinx Vivado Design Suite

Users

- ARM Ltd.
- AMD
- Intel
- Futjitsu VSLI
- STMicroelectronics
- Realtek
- ...

Literature I

David C Black, Jack Donovan, Bill Bunton, and Anna Keist.
SystemC: From the ground up.
Springer, 2004.

Andrew Canis, Jongsok Choi, Mark Aldham, Victor Zhang, Ahmed Kammoona, Jason H Anderson, Stephen Brown, and Tomasz Czajkowski.
Legup: high-level synthesis for fpga-based processor/accelerator systems.
In *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, pages 33–36. ACM, 2011.

Christian Haubelt.
Digitale Hardware/Software-Systeme.
Springer-Verlag, 2010.

Literature II

Open SystemC Initiative.
Functional specification for systemc 2.3.1.
URL http://www.systemc.org, http://www.systemc.org, 2014.

Thorsten Grötkerand Stan Liao, Grant Martin, Stuart Swan, and Thorsten Grötker.
System design with SystemC.
Springer, 2002.