The HW/SW Design Language SystemC

Ossimitz Christoph
October 25th 2016
Hardware description languages

- VHDL
- Verilog
- SystemVerilog
- SystemC
SystemC

“SystemC is a C++ class library and a methodology that you can use to effectively create a cycle-accurate model of software algorithms, hardware architecture, and interfaces of your SoC (System On a Chip) and system-level designs.”
SystemC

“SystemC is a C++ class library and a methodology that you can use to effectively create a cycle-accurate model of software algorithms, hardware architecture, and interfaces of your SoC (System On a Chip) and system-level designs.”
SystemC

“The SystemC Class Library provides the necessary constructs to model system architecture including hardware timing, concurrency, and reactive behavior that are missing in standard C++.”
# Language Comparison

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Architecture</th>
<th>Hw/Sw</th>
<th>Behaviour</th>
<th>Functional Verification</th>
<th>Test-bench</th>
<th>RTL</th>
<th>Gates</th>
<th>Transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VHDL</th>
<th>System Verilog</th>
<th>VeraE</th>
<th>Sugar Jeda</th>
<th>Matlab</th>
</tr>
</thead>
</table>

http://www.esa.int/Our_Activities/Space_Engineering_Technology/Microelectronics/System-Level_Modeling_in_SystemC
SystemC & VHDL - Comparison

```systemc
SC_MODULE(half_adder) {
    sc_in<bool> a;
    sc_in<bool> b;
    sc_out<bool> sum;
    sc_out<bool> carry;

    void calc_output() {
        sum.write(a.read() xor b.read());
        carry.write(a.read() and b.read());
    }

    SCCTOR(half_adder) {
        SC_METHOD(calc_output);
        sensitive a;
        sensitive b;
    }
};
```

```vhdl
entity half_adder is
    port(
        a : in std_logic;
        b : in std_logic;
        sum : out std_logic;
        carry : out std_logic;
    );
end entity;

architecture beh of half_adder is
begin
    calc_output : process(a,b)
    begin
        sum <= a xor b;
        carry <= a and b;
    end process;
end architecture;
```
SystemC & VHDL - Comparison

SC_MODULE(full_adder) {
    sc_in<bool> a;
    sc_in<bool> b;
    sc_in<bool> c_in;

    sc_out<bool> sum;
    sc_out<bool> c_out;

    sc_signal<bool> c_int1;
    sc_signal<bool> c_int2;
    sc_signal<bool> sum_int1;

    void calc_carry() {
        c_out.write(c_int1.read() or c_int2.read());
    }

    SC_CTOR(full_adder) {
        half_adder* h_add1 = new half_adder("h_add1");
        h_add1->a(a);
        h_add1->b(b);
        h_add1->sum(sum_int1);
        h_add1->carry(c_int1);

        half_adder* h_add2 = new half_adder("h_add2");
        h_add2->a(c_in);
        h_add2->b(sum_int1);
        h_add2->sum(sum);
        h_add2->carry(c_int2);

        SC_METHOD(calc_carry);
        sensitive << c_int1;
        sensitive << c_int2;
    }
};

entity full_adder is

port(
    a : in std_logic;
    b : in std_logic;
    c_in : in std_logic;
    sum : out std_logic;
    c_out : out std_logic;
);

end entity;

architecture beh of full_adder is

signal c_int1, c_int2, sum_int1 : std_logic;

begin

    h_add1 : half_adder
    port map(a, b, sum_int1, c_int1);

    h_add2 : half_adder
    port map(c_in, sum_int1, sum, c_int2);

    c_out <= c_int1 or c_int2;

end architecture;
C++ functions in SystemC

```c
SC_MODULE(file_streamer)
{
    sc_in<sc_uint<8> > clk;
    sc_in<sc_uint<8> > clk_en;
    sc_out<sc_uint<8> > data_stream;
    FILE* stream_file;

    void calc_output() {
        if(clk_en.read()) {
            sc_uint<8> temp = fgetc(stream_file);
            data_stream.write((sc_uint<8>) temp);
        }
    }

    SCCTOR(file_streamer) {
        stream_file = fopen("streamdata.txt", "r");
        SC_METHOD(calc_output);
        sensitive << clk.pos();
    }
};
```
Transaction-Level Modeling (TLM)

- High-level modeling of a system
- Faster simulation speed compared to RTL
- SystemC TLM part of SystemC library
- For HW/SW Codesign, processor TLM models exist

- Typical use cases:
  - Early Software Development
  - Architecture Exploration
Transaction-Level Modeling (TLM)

- TLM abstracts communication between system components
- Components exchange data within *transactions*
- No implementation details like protocols, bus standards etc
- Different ways to model timing in a TLM model
SystemC-AMS

- Extension to allow modeling of analog and mixed-signal systems
- Continuous time and signals
- Additional library loaded together with SystemC library
- No limitations on the usage of SystemC features
- Different ways of modeling (block diagram, electrical network etc.)
Simulation Tools

- OSCI SystemC
- Modelsim/Questa (Mentor Graphics)
- Incisive (Cadence)
- VCS (Synopsys)
- Active-HDL (Aldec)
Synthesis Tools

• Catapult (Mentor Graphics)
• Stratus HLS (Cadence)
• Vivado HLS (Xilinx)
High-Level Synthesis

- Generates RTL code from higher level specification (usually algorithmic-level)
- Source commonly written in (a subset of) SystemC, C/C++ or MATLAB
- HLS still requires some user effort to work properly
High-Level Synthesis

- How many multipliers/dividers?
- How many cycles does the operation take?
- Pipelining?

double t_1 = x*x;
double t_2 = y*y;
double t_3 = t_1 + t_2;
double t_4 = sqrt(t3);
x = x/t_4;
y = y/t_4;
High-Level Synthesis

```plaintext
for(int i = 0; i < 4; i++)
    y = a[i] + y;
```
High-Level Synthesis

for(int i = 0; i < 4; i++)
    y = a[i] + y;
for(int i = 0; i < 4; i++)
    y = a[i] + y;
HLS with C/C++

- Good tool support
- Components modeled as functions or classes
- Easy for software engineers to get into
- No native notion of timing, concurrency and all the other nice SystemC features
Who uses SystemC?

- Intel
- Qualcomm
- Infineon
- NXP
Who uses SystemC?

Modeling of Automotive Wheel-Speed Sensor IC’s in SystemC

North American SystemC User’s Group meeting XVIII, San Francisco, CA

6th June 2012

Dr. Fritz Rasbornig
Wolfgang Granig
Dr. Simon Hainz

Infineon Technologies Austria AG
Sense & Control

email: Fritz.Rasbornig@infineon.com

http://www.nascug.org/events/18th/modeling_wheel-speed_sensor_6-6-2012.pdf
Who uses SystemC?

General Aspects

- **Overview of SystemC models today:**
  - Infineon (department Sense & Control) is using SystemC-models for mechanical, thermal and electrical investigations for automotive sensor products for several years.
  - SystemC models of sensor products (e.g. speed-, angle and linear Hall sensor) were delivered to several customers.
  - SystemC models are also used for internal co-simulation with VHDL → internal verification (“Executable Specification”)
  - Design-flow with the usage of SystemC-models:

Example Wheel Speed Sensor Project:

AE=Application Engineering  
CE=Concept Engineering  
PD=Product Design  
CV=Comonent Verification

http://www.nascug.org/events/18th/modeling_wheel-speed_sensor_6-6-2012.pdf
Who uses SystemC?

Motivation for SystemC/SystemC-AMS Models

- Advantages of using SystemC models for customers and Infineon:
  - Allow customer testing of chip features in a very early design step (before tape-out)
  - Reduction of development costs (e.g. avoiding redesigns etc.)
    - Time to market / First time right
  - Increase internal understanding of chip architecture and concept
  - Avoiding miss-interpretation of the specification
  - An accurate system model can deliver:
    - Understanding of the internal digital-algorithms, state-machines and output-protocols
    - Implementation of main signal processing flow (of analog- and digital-section)
Thank you for your attention!
References

- OSCI TLM-2.0 Language Reference Manual
- SystemC Verification with Modelsim
- Vivado Design Suite User Guide – High-Level Synthesis
- Modeling of Automotive Wheel-Speed Sensor IC’s in SystemC