SE303 : Co-simulation

Simuler conjointement HDL et SystemC

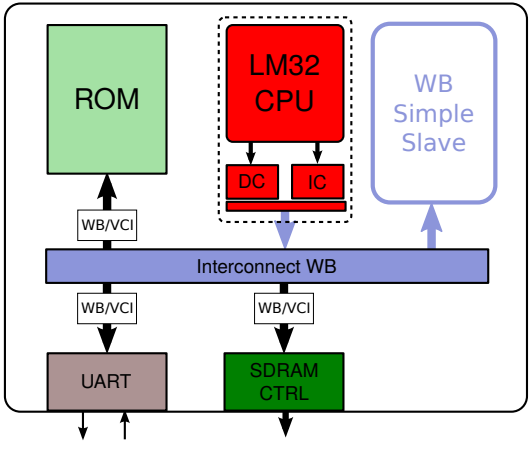
Tarik Graba
<tarik.graba@telecom-paris.fr>
Septembre 2020

Co-Simulation
Raffinement vers le RTL

- Remplacer un module SystemC par un module RTL
- Utiliser la plateforme comme modèle de référence
- Aller vers un modèle plus précis
- Modèle RTL/Synthétisable
- Besoin d'un simulateur RTL

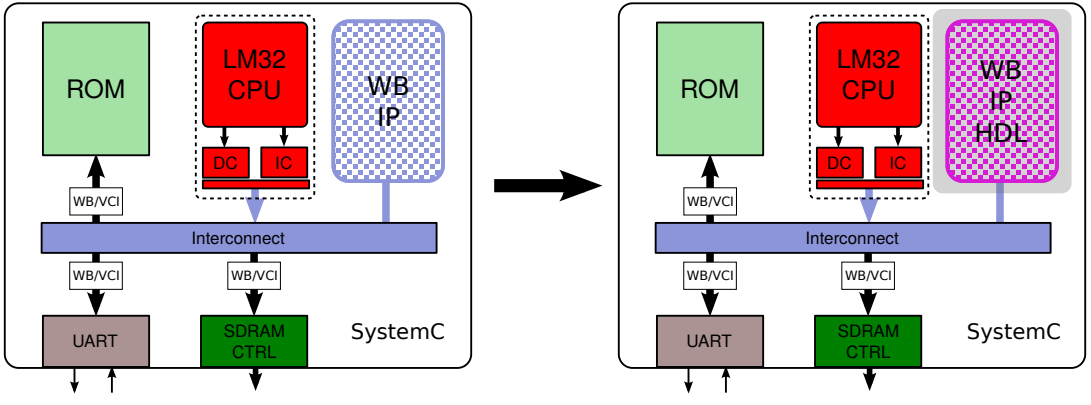
Co-Simulation

Raffinement vers le RTL



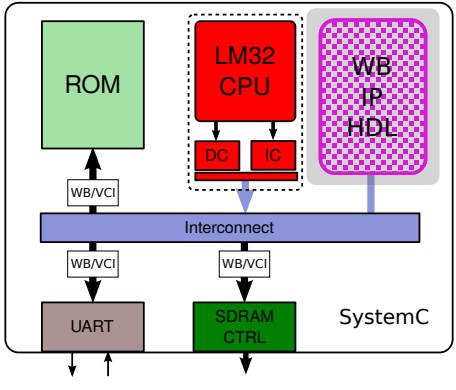
Co-Simulation

Raffinement vers le RTL



Co-Simulation

Raffinement vers le RTL



Co-Simulation

Intégrer un mudule Verilog dans une simulation SystemC

- Aller vers un modèle plus précis
- L'outil doit le permettre:
  - de simuler du HDL (Verilog/SystemVerilog, VHDL) et du SystemC
  - de faire les deux en même temps
- Modelsim sait le faire (NCSim de Cadence sait aussi le faire)

Co-Simulation

Utiliser un simulateur qui comprend les 2 langages

- `Modelsim`
  - `vlib` bibliothèque de travail
  - `vlog` pour compiler du Verilog/SystemVerilog
  - `vcom` pour compiler vu VHDL
  - `vsim` pour simuler
- Pour le SystemC
  - `sccom` pour la compilation
  - `sccom -link` pour l'édition de liens

Co-Simulation

Intégrer un mudule Verilog dans une simulation SystemC

- Garder la simulation actuelle:
  - l'infrastructure de simulation
  - le `sc_main`
- Remplacer un module SystemC par un équivalent HDL
- Modelsim simulera le tout
- Problème, quand on compile du C++, il faut que tous les objets soient définis!
  - Obligation d'avoir un wrapper

Co-Simulation avec Modelsim

Un wrapper!

```
module wb_simple_slave (
                        input  p_clk,
                        input  p_resetn,
                        //  WB signals
                        input  [31:0] p_wb_DAT_I,
                        output [31:0] p_wb_DAT_O,
                        input  [31:0] p_wb_ADR_I,
                        output p_wb_ACK_O,
                        input  p_wb_CYC_I,
                        output p_wb_ERR_O,
                        input  p_wb_LOCK_I,
                        output p_wb_RTY_O,
                        input  [3:0]p_wb_SEL_I,
                        input  p_wb_STB_I,
                        input  p_wb_WE_I
                        );

                        ....

endmodule
```

Co-Simulation avec Modelsim

Un wrapper!

```
#ifndef _SCGENMOD_wb_simple_slave_
#define _SCGENMOD_wb_simple_slave_

#include "systemc.h"

class wb_simple_slave : public sc_foreign_module
{
public:
    sc_in<bool> p_clk;
    sc_in<bool> p_resetn;
    sc_in<sc_uint<32> > p_wb_DAT_I;
    sc_out<sc_uint<32> > p_wb_DAT_O;
    sc_in<sc_uint<32> > p_wb_ADR_I;
    sc_out<bool> p_wb_ACK_O;
    sc_in<bool> p_wb_CYC_I;
    sc_out<bool> p_wb_ERR_O;
    sc_in<bool> p_wb_LOCK_I;
    sc_out<bool> p_wb_RTY_O;
    sc_in<sc_uint<4> > p_wb_SEL_I;
    sc_in<bool> p_wb_STB_I;
    sc_in<bool> p_wb_WE_I;

    ......
```

Co-Simulation avec Modelsim

Un wrapper!

```
    ......

    wb_simple_slave(sc_module_name nm, const char* hdl_name)
     : sc_foreign_module(nm),
       p_clk("p_clk"),
       p_resetn("p_resetn"),
       p_wb_DAT_I("p_wb_DAT_I"),
       p_wb_DAT_O("p_wb_DAT_O"),
       p_wb_ADR_I("p_wb_ADR_I"),
       p_wb_ACK_O("p_wb_ACK_O"),
       p_wb_CYC_I("p_wb_CYC_I"),
       p_wb_ERR_O("p_wb_ERR_O"),
       p_wb_LOCK_I("p_wb_LOCK_I"),
       p_wb_RTY_O("p_wb_RTY_O"),
       p_wb_SEL_I("p_wb_SEL_I"),
       p_wb_STB_I("p_wb_STB_I"),
       p_wb_WE_I("p_wb_WE_I")
    {
        elaborate_foreign_module(hdl_name);
    }
    ~wb_simple_slave()
    {}

};

#endif
```

Co-Simulation avec Modelsim

Un wrapper!

- On redéfinit un module avec une interface compatible
- On hérite plus d'un `sc_module` mais d'un `sc_foreign_module`
- Dans le constructeur on appelle `elaborate_foreign_module` pour attacher le module SystemC à un module HDL de nom `hdl_name`

Co-Simulation avec Modelsim

Un wrapper!

- Si le module HDL est déjà compilé, mentor fourni un outil pour générer automatiquement ce wrapper
  - `scgenmod`
- Permet de préciser la conversion de types entre les deux modes
  - `scgenmod --help` pour plus d'informations
- L'outil n'est pas parfait et le wrapper généré n'est parfois pas bon:
  - Attention aux paramètres
  - Attention aux types

Co-Simulation avec Modelsim

Résumons

1. Créer la bibliothèque de travail
   - `vlib work`
2. Compiler le HDL
   - `vlog foo.v`
3. Généré le fichier d'en-tête du wrapper
   - `scgenmod toto > foo.h`
4. Remplacer dans le `sc_main` le module SystemC par le wrapper

Co-Simulation avec Modelsim

Modifier le textttsc_main

Remplacer l'instanciation du module SystemC…

```
// Déclaration du module SystemC
#include "wb_simple_slave.h"

...

soclib::caba::WbSimpleSlave<wb_param> simple_slave ("WB_simple_slave");
simple_slave.p_clk              (signal_clk);
simple_slave.p_resetn           (signal_resetn);
simple_slave.p_wb               (signal_wb_slave);

...
```

Co-Simulation avec Modelsim

Modifier le textttsc_main

… par le wrapper.

```
// Définition du wrapper vers le module Verilog
#include "hdl/include/wb_simple_slave.h"

...

wb_simple_slave simple_slave("simple_slave","wb_simple_slave");
simple_slave.p_clk        (signal_clk);
simple_slave.p_resetn     (signal_resetn);
simple_slave.p_wb_DAT_I   (signal_wb_slave.MWDAT );
simple_slave.p_wb_DAT_O   (signal_wb_slave.MRDAT );
simple_slave.p_wb_ADR_I   (signal_wb_slave.ADR );
simple_slave.p_wb_ACK_O   (signal_wb_slave.ACK );
simple_slave.p_wb_CYC_I   (signal_wb_slave.CYC );
simple_slave.p_wb_ERR_O   (signal_wb_slave.ERR );
simple_slave.p_wb_LOCK_I  (signal_wb_slave.LOCK);
simple_slave.p_wb_RTY_O   (signal_wb_slave.RTY );
simple_slave.p_wb_SEL_I   (signal_wb_slave.SEL );
simple_slave.p_wb_STB_I   (signal_wb_slave.STB );
simple_slave.p_wb_WE_I    (signal_wb_slave.WE  );
...
```

Co-Simulation avec Modelsim

Et en suite

- Compiler les modules SystemC
  - `sccom`
- L'édition de liens
  - `sccom -link`

Co-Simulation avec Modelsim

Avec SocLib?

- La compilation avec `sccom` est supportée par SocLib
- Il suffit de configurer `soclib-cc`

`http://www.soclib.fr/trac/dev/wiki/SoclibCc/AndModelsim`

Co-Simulation avec Modelsim

Où commencer?

- L'exemple accessible ici dans la branche **cosim** du dépôt *soclib_hello_world*.
- Avec:
  - Une version RTL de l'esclave simple de départ
- Aussi, ce qu'il faut pour automatiser:
  - la compilation du SystemC,
  - la compilation du RTL,
  - la génération des wrappers.