SCiPX: a SystemC to IP-XACT converter (v1.0)

http://www-sop.inria.fr/aoste/index.php?page=software/scipx

Jean-François Le Tallec, Benoît Ferrero

January 28, 2011

Contents

1 Introduction

1.1 Tool overview

SCiPX is a translator from SystemC code to IP-XACT interface description. Its goal is to allow to make any SystemC component description become IP-XACT compliant, and thus available for insertion and plug-and-play assembly into IP-XACT based virtual platform building environments. SCiPX and its documentation are available from
`http://www-sop.inria.fr/aoste/index.php?page=software/scipx`. SCiPX and its environment are presented in a report paper [1].

SystemC is a language for Electronic System-Level (ESL) design of digital circuits, and Systems-on-Chip (SoC), available at `http://www.systemc.org/` and formalized as IEEE 1666 standard. SystemC design representations may be provided at several well-defined levels. Widely used are the RTL level (for synthesis) and the TLM level (for high-level simulation). SystemC program runtime consists of two clearly distinct phases: first, an elaboration phase which consists in actually building the various objects and the global structure of teh system; a simulation phase, which animates the system built in the previous phase, whose structure then remains static and unchanged.

IP-XACT is an Architecture Description Language (ADL) meant for easy assembly of IP-component based Hardware Virtual Platforms (VPF). The elementary component bahavioral content may eventually be provided in HDL form, for instance as SystemC source code (but not necessarily so). IP-XACT is available from `http://www.accellera.org/activities/ip-xact`, and documented as IEEE 1685 standard. IP-XACT allows for various types of proprietary annotations as so-called "vendor extensions".

SCiPX extracts structural information from SystemC programs, both by running their elaboration phase (in C++, using the dedicated C++ libraries of SystemC), and by performing static analysis (mostly for naming informations). Results from both types of extractions are then reassembled into IP-XACT compliant syntax.
For this, SCiPX builds up on top of existing softwares:

- the analysis and extraction during elaboration phase borrows partly from **PinaVM**, an academic tool developed at VERIMAG laboratory in Grenoble, available at `http://gitorious.org/pinavm` and described in [2]; PinaVM in turn was itself based on the LLVM C++ compiler infrastructure (`http://llvm.org/`); It provides translation into dedicated format for formal analysis tools mostly. A survey of similar tools for analysis of SystemC elaboration phase is conducted by the authors of PinaVM in [3]
- SCiPX also internally relies on the Doxygen environment (`http://www.doxygen.org/`) for abstract syntax analysis of C++ and SystemC code.

The current version of SCiPX uses PinaVM, LLVM version 2.7 , and Doxygen version 1.6 . The translator can be applied as a stand-alone command process.

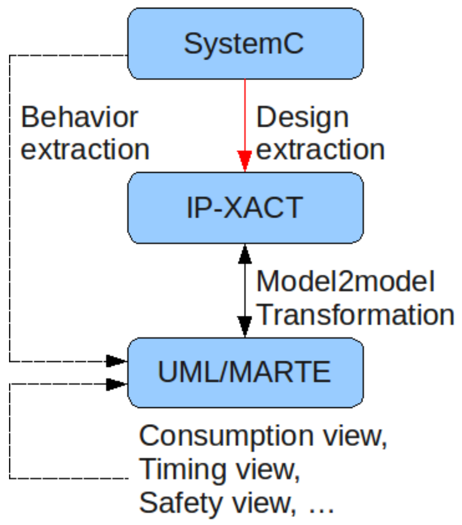

Figure 1: Global Flow

SCiPX is connected to further model transformations, this time between IP-XACT and the OMG standard profile for Modeling and Analysis of Real-Time Embedded systems (**MARTE**), available at `http://www.omg.org/omgmarte/`. These companion tools, named **Marte2IPXACT** and **IPXACT2Marte** respectively, are also availlable from our software download site (`http://www-sop.inria.fr/aoste/index.php?page=software`). They require the use of a MARTE-compliant UML editor to display and interact with the MARTE description, such as Papyrus by CEA-LIST, `http://www.papyrusuml.org/`.
The purpose of such translations is to link the IP-XACT standard for hardware IP-component based virtual platform construction to professional tools for UML/SysML-based Model Driven Engineering (MDE), thereby allowing sophisticated handling of property annotations (such as SWaP: Size/Weight and Power info) in a much more fancy and standardized way than current vendor extensions. The global transformation approach is pictured in 1.

SCiPX modifies PinaVM back-end to produce IP-XACT compliant architecture descriptions. Because some of the naming information is lost across the underlying LLVM (together with several other minor things), another processing is run in parallel to recover the needed parts by static analysis on the original SystemC programs using Doxygen. Collaboration betwen the two processings ensures that the results are then pasted at the proper place.

So far SCiPX translates from RTL code to the equivalent level of interface and structure declaration in IP-XACT. Extension towards TLM level is well under way, currently tested in beta-release. The various steps of SCiPX are displayed in 2.

Future steps in our development will include the extraction of component behavioral parts from SystemC directly into model form expressed as UML/MARTE descriptions, or conversely the production of SystemC code from abstract Models of Computation and Communications (MoCCs) provided in MARTE. This is made possible because of the possibility offered by MARTE to annotate precisely the UML bahavioral diagrams (state and activity diagrams mostly) to endow them with the precise operational semantics of hierarchical Mealy FSMs, SDF or Kahn Proces Networks, and other such modeling framework recognized in the SystemC literature as a goal for design.

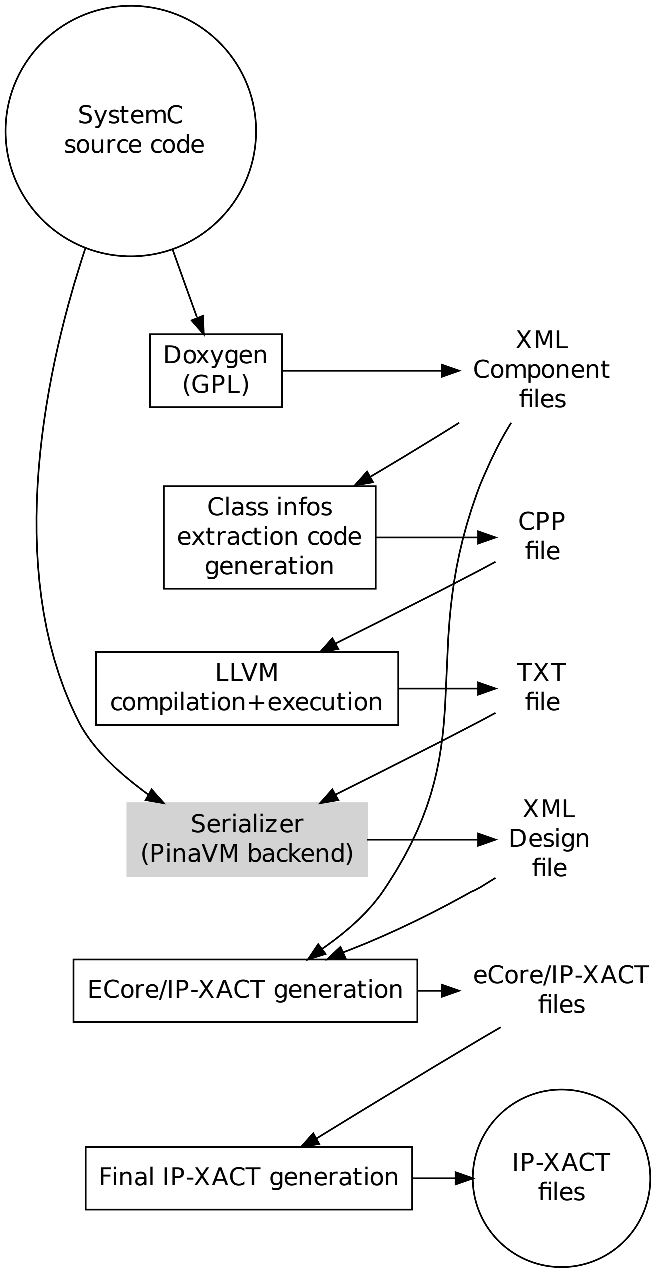

Figure 2: Transformation steps in SCiPX

2 Getting started: installation notice

Warning : Currently the prototype SCiPX version runs only under Debian Linux 32 bits (check the actual distribution page for new versions). Otherwise people should install a Linux virtual machine on their computer.

Step 1 Prior to SCiPX installation one needs to install several interdependent Debian package. This is done by the command

```
$sudo apt−get install autoconf graphviz bison flex
libxerces−c−dev xsdcxx
```

Step 2 Then one can proceed to download the SCiPX.tgz file from the download site
`http://www-sop.inria.fr/aoste/software/scipx/download/`.

Step 3 Decompresing this file (typing `tar xvzf SCiPX.tgz` for instance) will create files under the root `SCiPX` subdirectory. This hierarchy contains a modified version of PinaVM. It consists of source files which should then be compiled.

Step 4 Next, one should provide the absiolute path-name of the tool's installation directory by replacing teh value of the Unix variable `$TOOLSDIR`

```
$cd SCiPX
$autoconf
$export TOOLSDIR=${PWD}INSTALLDIR
$./configure −−with−tools−dir=$TOOLSDIR
```

Step 5 Actually run the installation script (which should take a couple of minutes)

```
$./install−pinavm.sh
```

Step 6 The tool location should then be added to the general `$PATH` variable. Then PinaVM requires to rerun its installation (with all UNix variables set).

```
$export PATH=$PATH:$TOOLSDIR/lib/llvm−gcc/bin:$TOOLSDIR
/lib/llvm−2.6/bin
$./install−pinavm.sh
```

Now SCiPX is successfully installed. It can be tested by running a small toy example (simple_example) :

```
$cd systemc−examples/simple_example;
$make
```

The result of the .make command will produce a hierarchy of IP-XACT .xml files from the .cpp files of the SystemC design, stored in a `spirit` folder.

2.1 Implementation features

We describe now some of the internals of SCiPX. This should be read carefully by anyone who wishes to enter or modify the source code of SCiPX.

Ecore meta-models of both SystemC and IP-XACT models have been developed. They can be provided on demand. Please contact us (see the download page).
From the SystemC Ecore meta-model an .xsd schema for XMI is automatically generated, thanks to the integrated genmodel of Eclipse. This .xsd has been modified to handle the addition of references, converted into c++ classes and structures thanks to the `xsd` tool. Possible modifications should not alter this.

Our reference .xsd file can be provided on demand. The `xsd` tool can be retrieved by the command:

```
$sudo apt-get install xsdcxx
```

If ever the .xsd files get modified, the new corresponding `c++` files are built by running :

```
$xsd cxx-tree --generate-serialization --namespace-map
'http://fr.inria.aoste.systemc'='pinavmv2_ns'
systemCXMI.xsd
```

Then the .cxx file should be renamed into .cpp, and copied in `SCiPX/backends/XMLBackend` folder. This folder actually contains most of the files which could be modified. In particular, it contains all methods and classes to genrate the IP-XACT objects. Note that all fields are mandatory in these classes.

Recompilation of `SCiPX` (after modification) can be done by runing `make` under the `SCiPX/toplevel` subfolder.

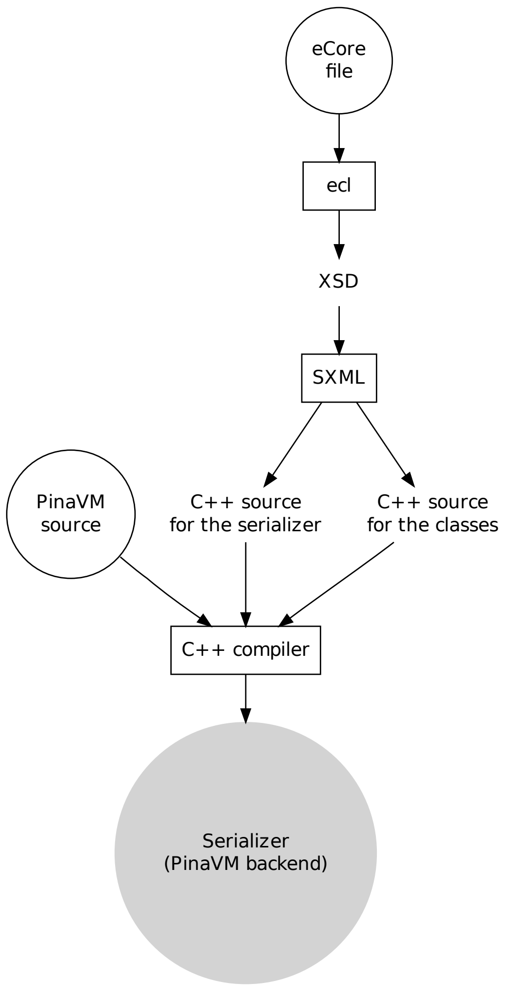

Figure 3: Production of the serializer

References

[1] Jean-Francois Le Tallec, Julien DeAntoni, Robert de Simone, Benoit Ferrero, Frédéric Mallet, and Laurent Maillet-Contoz. Combining systemc, ip-xact and uml-marte in model-based soc design. In *2nd Workshop on Model Based Engineering for Embedded Systems Design (M-BED 2011)*, Grenoble, France, 2011.

[2] Kevin Marquet and Matthieu Moy. PinaVM: a SystemC Front-End Based on an Executable Intermediate Representation. Technical Report TR-2010-8, Verimag, 2010.

[3] Kevin Marquet, Matthieu Moy, and Bageshri Karkare. A Theoretical and Experimental Review of SystemC Front-Ends. Technical Report TR-2010-4, Verimag, 2010.