Deliverable – D19

Software tools released under GPL

Grant Agreement No:	**214364**
Project acronym:	**GALAXY**
Project title:	**GALS InterfAce for CompleX Digital System Integration**
Funding Scheme:	**STREP**
Date of latest version of Annex I against which the assessment will be made:	**23.10.2007.**
Contractual Date of Delivery to the EC:	**28 February 2010**
Actual Date of Delivery to the EC:	**28 February 2010**
Author(s):	**Lilian Janin (UNIMAN)**
Participant(s):	**UNIMAN**
Work Package:	**WP5**
Security:	**Public**
Nature:	**Software tools + Report**
Version:	1
Total number of pages:	22

Abstract:

This document introduces the first public release of the GALAXY tools.

Manual and tutorials being planned as a future activity, this report is intended to serve, in the meantime, as a user guide for software installation and first steps.

The main components described in this report are:

- graphical user interface for design entry,
- code generators that produce Makefiles, SystemC and HDL architecture code
- graphical frontend to control external tool flows
- simulation and co-simulation backends

Keyword list: GALS, asynchronous, software, tools, co-simulation

Function	*Responsibility*	*Date*	*Signature*
Written by:	Lilian Janin	28 Feb 2010	
Checked by:	Members of GALAXY Consortium		
Approved by:	-		

Reserved to EC

Approved by:			

CHANGE RECORDS

ISSUE	*DATE*	*§ : CHANGE RECORD*	*AUTHOR*
1	28-Feb-10	1st version	Lilian Janin

BIBLIOGRAPHIC RECORD

Project Number:	214364 GALAXY
Project Title:	GALAXY
Deliverable Type:	Software tools + Report
Deliverable Number:	D19
Contractual Date of Delivery:	28 Feb 2010
Actual Date of Delivery:	28 Feb 2010
Title of Deliverable:	Software tools released under GPL
Work package contributing to the Deliverable:	WP5
Authors:	Lilian Janin
Abstract	This document introduces the first public release of the GALAXY tools. Manual and tutorials being planned as a future activity, this report is intended to serve, in the meantime, as a user guide for software installation and first steps. The main components described in this report are: - graphical user interface for design entry, - code generators that produce Makefiles, SystemC and HDL architecture code - graphical frontend to control external tool flows - simulation and co-simulation backends
Keywords	GALS, asynchronous, software, tools, co-simulation
Confidentiality Level	Public
Name of Client:	EC
Distribution List:	GALAXY, EC, internet
Authorised by:	
Issue:	1
Document ID:	D19
Total Number of Pages:	22
Contact Details:	Lilian.janin@manchester.ac.uk

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

This document introduces the first public release of the GALAXY tools.

Manual and tutorials being planned as a future activity, this report is intended to serve, in the meantime, as a user guide for software installation and first steps.

The main components described in this report are:

- graphical user interface for design entry;
- code generators that produce Makefiles, SystemC and HDL architecture code;
- graphical frontend to control external tool flows;
- simulation and co-simulation backends.

2 REFERENCES

2.1 ACRONYMS

ASIP	Asynchronous-Synchronous IPs packaging format
ELF	Executable and Linkable Format
FPGA	Field-Programmable Gate Array
GALS	Globally Asynchronous Locally Synchronous
IP	Intellectual Property
VHDL	VHSIC Hardware Description Language
XML	eXtendible Markup Language

3 LIST OF TOOLS

Here is an exhaustive list of the tools provided within the GALAXY framework, and a short description for each of them.

3.1 GALAXY-IDE

Galaxy-ide is the main graphical user interface.

It can be used for design entry by allowing the user to create, open, transform and save ASIP design files. For this, the main view shows a graphical representation of the design.

It also serves as a front-end to launch all the other tools. For this, a tool flow window allows the user to control external tools, and an execution window provides means of interaction with these tools.

3.2 ASIP TRANSFORMATIONS

The ASIP file format is Galaxy's Asynchronous-Synchronous IPs packaging format. It is used by most of the tools in the framework to convey the description of the current design. Design entry tools (*galaxy-ide*) create or modify ASIP files with user components, while back-end tools use the same format to add lower level components such as co-simulation components at the interfaces.

Tools: *asip-add-cosim-comps*, *asip-split-sim*

Asip-add-cosim-comps: Adds the required co-simulation components at the interfaces of each group of components targeting the same simulator.

Asip-split-sim: Splits one ASIP file into multiple ones, each of them containing only the components targeting the same simulators (1 ASIP file per simulator).

The resulting files are usually sent directly to the code generators.

3.3 CODE GENERATORS (OR ASIP TRANSLATORS)

Tools: *asip2systemc*, *asip2v*, *asip2asm*, *asip2bash*.

These code generators read one ASIP file and generate the corresponding source code (SystemC, Verilog, ARM assembly language, bash script).

In the same way as the ASIP file, the generated source code is a top-level code instantiating and linking together the various IP source codes.

However, even though ASIP descriptions usually contain a hierarchy of components, the generated source codes are always flat (the hierarchy is actually flattened by *asip-split-sim*).

3.4 CODE COMPILERS (OR TRANSLATORS)

Tools: *galaxy-wrapper-bitgen*

galaxy-wrapper-map

galaxy-wrapper-ngdbuild

galaxy-wrapper-par

galaxy-wrapper-xst

galaxy-wrapper-gcc-systemc

galaxy-wrapper-arm-as

galaxy-wrapper-arm-ld

These tools transform a file from one file format to another. This includes compilers (e.g. g++: SystemC->object file or executable), synthesisers (e.g. Xilinx XST: Verilog->netlist) and many others (e.g. netlist->Xilinx FPGA bit file).

Combined in appropriate flows, they can be used to translate any HDL source code into formats that can be sent to simulators (including executable binaries, “simulated” by just executing them; e.g. SystemC simulations).

The correct execution sequence of these translators is controlled by the Tool Flow System (see Section 4).

3.5 SIMULATORS (OR SIMULATION TARGETS)

Tools: *galaxy-launcher-precompiled-executable*

galaxy-launcher-systemc-i386

galaxy-launcher-vcs

galaxy-launcher-icarus

galaxy-loader-arm-cpu-on-jim-s-board

galaxy-loader-virtexE-on-jim-s-board

In the ASIP format, simulators are referred to as “Simulation Targets”, and include any tool able to execute their input code (and optionally generating a trace file). The input code can be provided in many different formats: Verilog, SystemC, but also FPGA bit files, ELF executables.

In this context, simulators are therefore including software simulators, emulators, external FPGA boards, launchers of pre-compiled executable binaries, etc.

4 TOOL FLOW SYSTEM

The Tool Flow System is the gateway to link external (vendor or open-source) tools to the GALAXY framework.

It has a well-defined interface to facilitate the integration of new tools and tool flows.

Its aim is to generate, from the knowledge of available tool flows, an execution sequence of tools to go from the ASIP and IP source codes to the simulators.

To achieve this, the tool flow database is made of three main sections:

- available file formats (including how to recognise them, e.g. from their extension);
- available simulators, with which input file formats they require;
- available translators, with their required input and produced output file formats.

From this information, a graph of tools and file formats is created (Figure 1), and appropriate execution sequences are generated when the user desires to “simulate file X with simulator Y”.

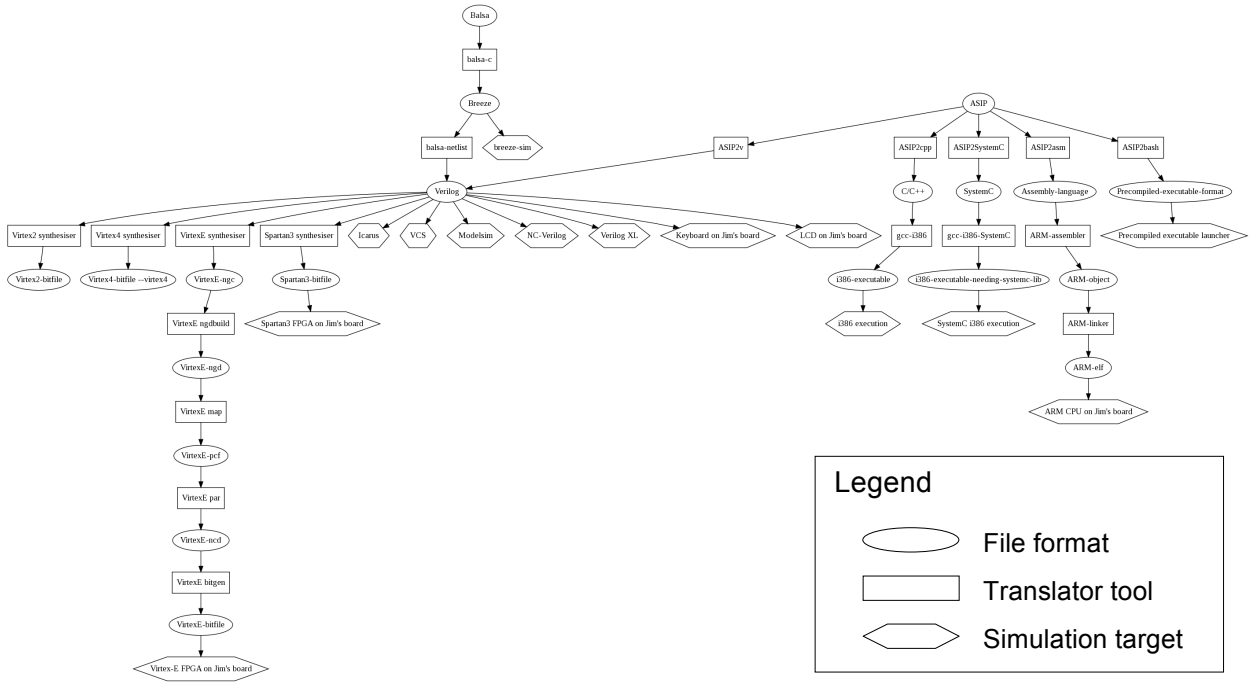

Figure 1: Example of tool flow

4.1 PLATFORMS AND REMOTE EXECUTION

Different users/computers have access to different sets of simulators and translators.

To take this into account, we introduced the notion of “platforms”, which are defined as “a processing unit with the ability to launch simulators and with its own disk storage (to store

trace files)" (so, FPGA boards are not usually seen as platforms. They are seen as simulators).

Tools available on a remote computer (remote platform) can be accessed, and additional data is stored in the database to know how to launch tools remotely (does the process need to go via ssh? etc.).

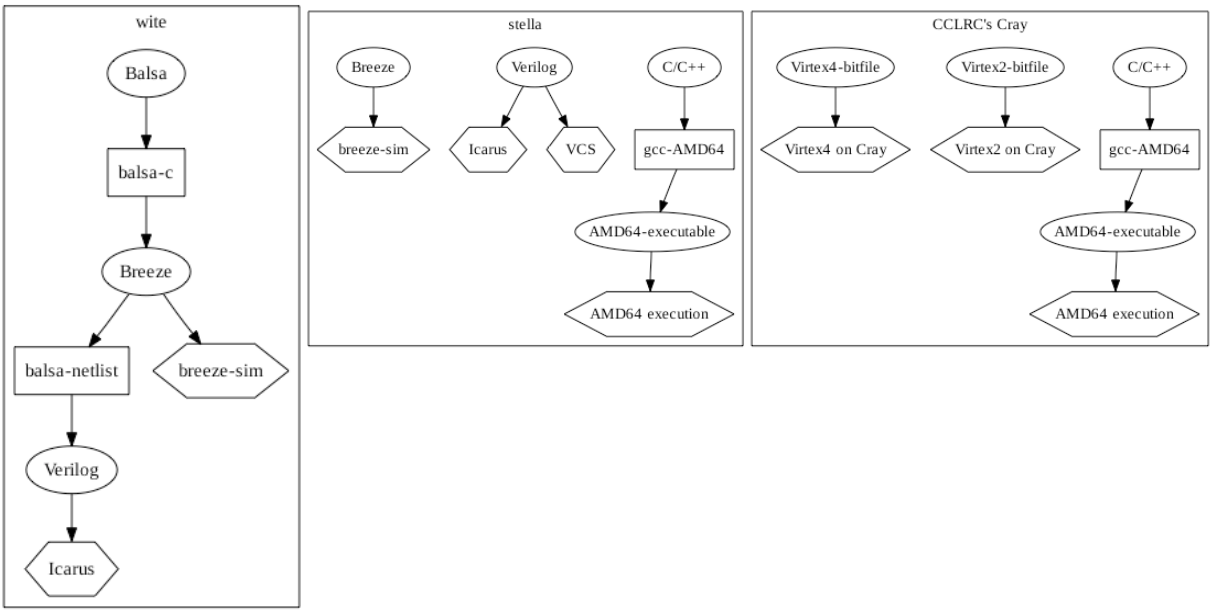

Figure 2: Different tool flows for different platforms

4.2 CO-SIMULATION: SPECIAL HARDWARE (FPGA) TARGETS

Hardware targets such as FPGA boards require some additional description.

Where co-simulation links between software simulators could be instantiated dynamically, and in unlimited numbers – it's all software - , co-simulation links between hardware entities are real pre-routed wires in fixed places and in limited numbers.

Therefore, the tools generating the co-simulation interfaces need to know how to refer to these wires, and need to know about the associated constraints.

In order to deal with this, we introduced a section to describe hardware target interfaces and hardware links between targets.

4.3 ASIP ROUTERS

When two hardware targets are not physically connected (or more often: when a hardware target is not directly connected to the host to access a software simulator), but the user targets these two targets with two interconnected components, it is impossible to route the signals between the two components.

To solve this problem, the Tool Flow System finds the possible hops through intermediate hardware targets, and automatically inserts "ASIP routers", components able to transfer the

signals on each hop, following a specific protocol. In the Galaxy-ide GUI, the user is given the choice of the different paths between targets, and the ASIP router components are transparently added to his design.

In Figure 3 below, three simulators are targeted in the design. The green link indicates that a direct connection is possible. The blue link reports a possible connection through intermediate hops. A red link would indicate that there is no way to connect two targets.

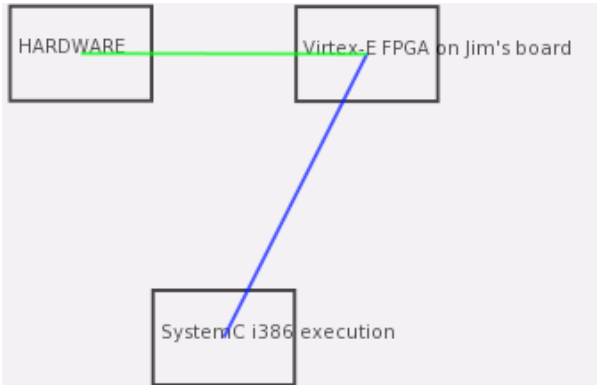

Figure 3: Analysis of links between simulation targets

Figure 4 shows the list of hops suggested in a drop-down list. In this case there was only one possible path to connect the two targets.

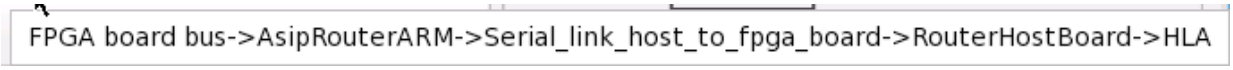

Figure 4: List of suggestions of routing hops to connect two targets

Figure 5 shows the new path between targets and through routers. Two routers were added. Black links are drawn between simulators and routers, as this is not completely implemented yet, but the backend tools are able to use this information correctly.

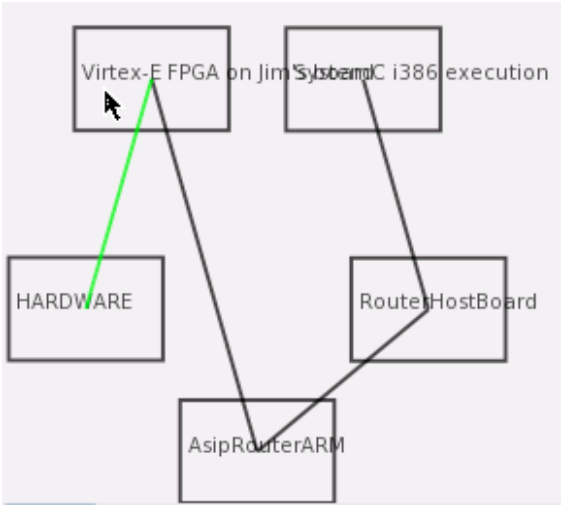

Figure 5: Final routing between targets and through routers

5 SIMULATION / CO-SIMULATION

Simulation is easily handled by the Tool Flow System: a group of files in a single file format is compiled/synthesised/placed/routed in a sequential flow, and finally one simulator is called, optionally generating a trace file.

Co-simulation is a lot harder. First it involves parallel execution of the different simulators, with the difficulty to show all the outputs concurrently and to provide control over all the simulators somehow.

But more importantly, before the execution, the co-simulation interfaces need to be prepared and communication protocols defined.

5.1 CO-SIMULATION INTERFACES

Section 4 mentioned the information stored in the Tool Flow System to describe the links between simulation targets. Those links are the basis of how the co-simulation interfaces are generated.

Rather than generating specific co-simulation interfaces between each set of two (connectable) simulators, the co-simulation interfaces are specific to the sets simulator + communication link. As most of the time a simulator is connected to only a single communication link, this reduces greatly the number of possible co-simulation interfaces. This simplified view allows us to prepare in advance co-simulation components in limited number, and instantiate them at the interfaces when preparing the co-simulation files (done by the tool *asip-add-cosim-comps*).

5.2 THE SIMULATION TARGET VIEW

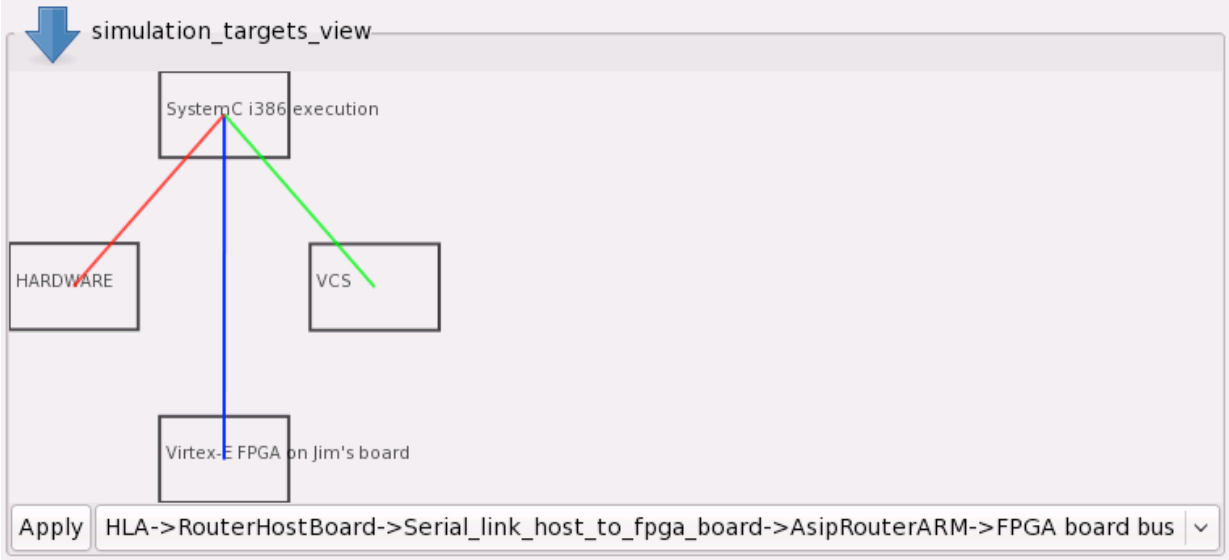

Figure 6: Galaxy-IDE Simulation Target View

As described and illustrated in Section 4.3 and Figure 6, the Simulation Target View colour-codes the links between simulation targets in order to help the user prepare his design for co-simulation:

- Green links represent two simulators directly connected to each other and ready for co-simulation;
- Blue links represent two simulators not connected to each other, but able to communicate through ASIP routers – selecting the link fills in the drop down list below to help the user insert the ASIP routers in the design;
- Red links represent two simulators that do not have any means to communicate – the user needs to select different simulators for interconnected components.

6 INTEGRATED DESIGN ENVIRONMENT

Galaxy-ide is the main graphical user interface.

It can be used for design entry by allowing the user to create, open, transform and save ASIP design files. For this, the main view shows a graphical representation of the design (Figure 7 shows a dual graph view, where the full design is visible in the top view and where the lower view is zoomed in on a particular component).

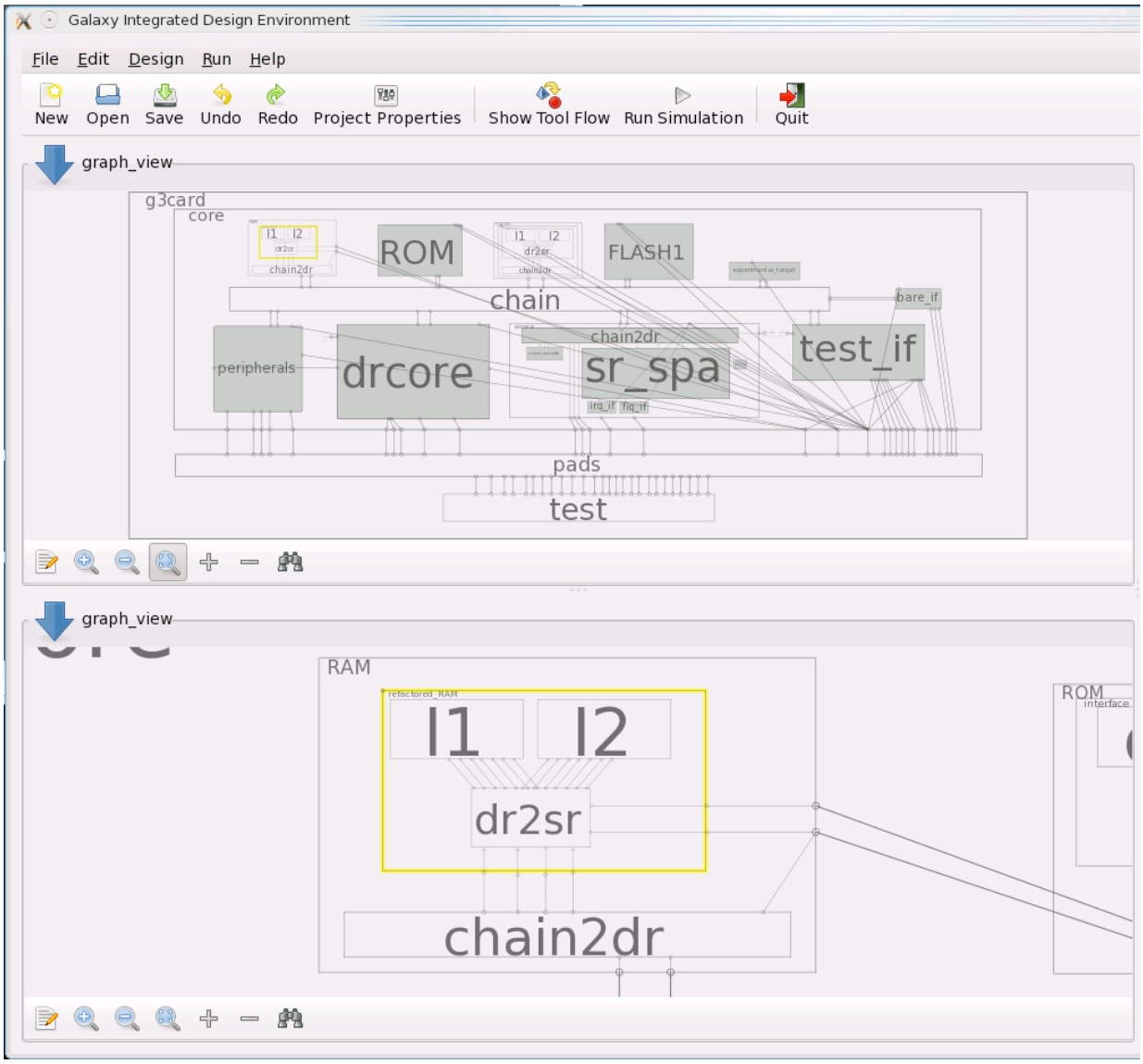

Figure 7: Galaxy-IDE Design Environment

The Galaxy IDE also serves as a front-end to launch all the other tools. For this, a tool flow window allows the user to control external tools, and an execution window provides means of interaction with these tools (Figure 8, Figure 9).

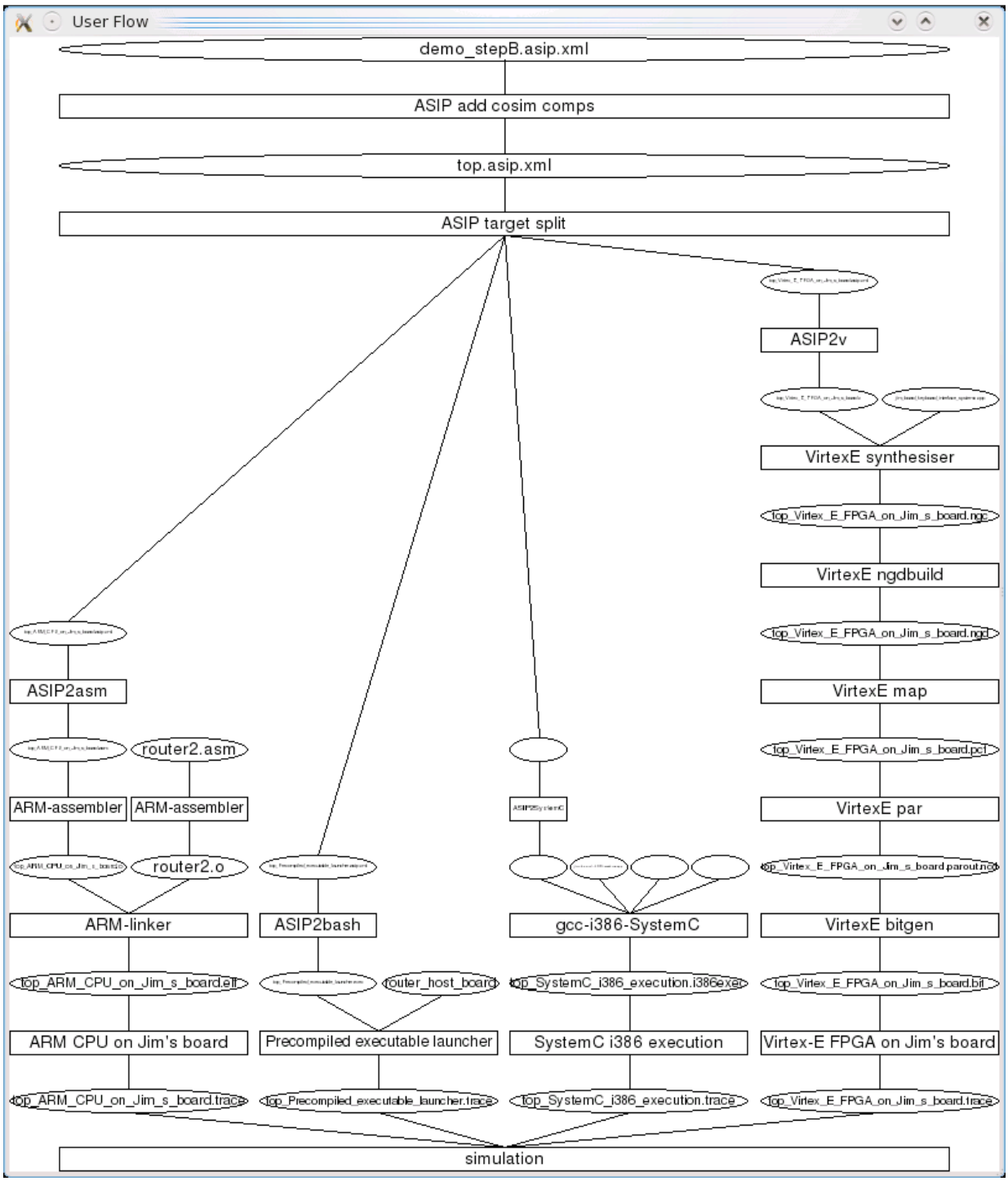

Figure 8: Galaxy-IDE Tool Flow View

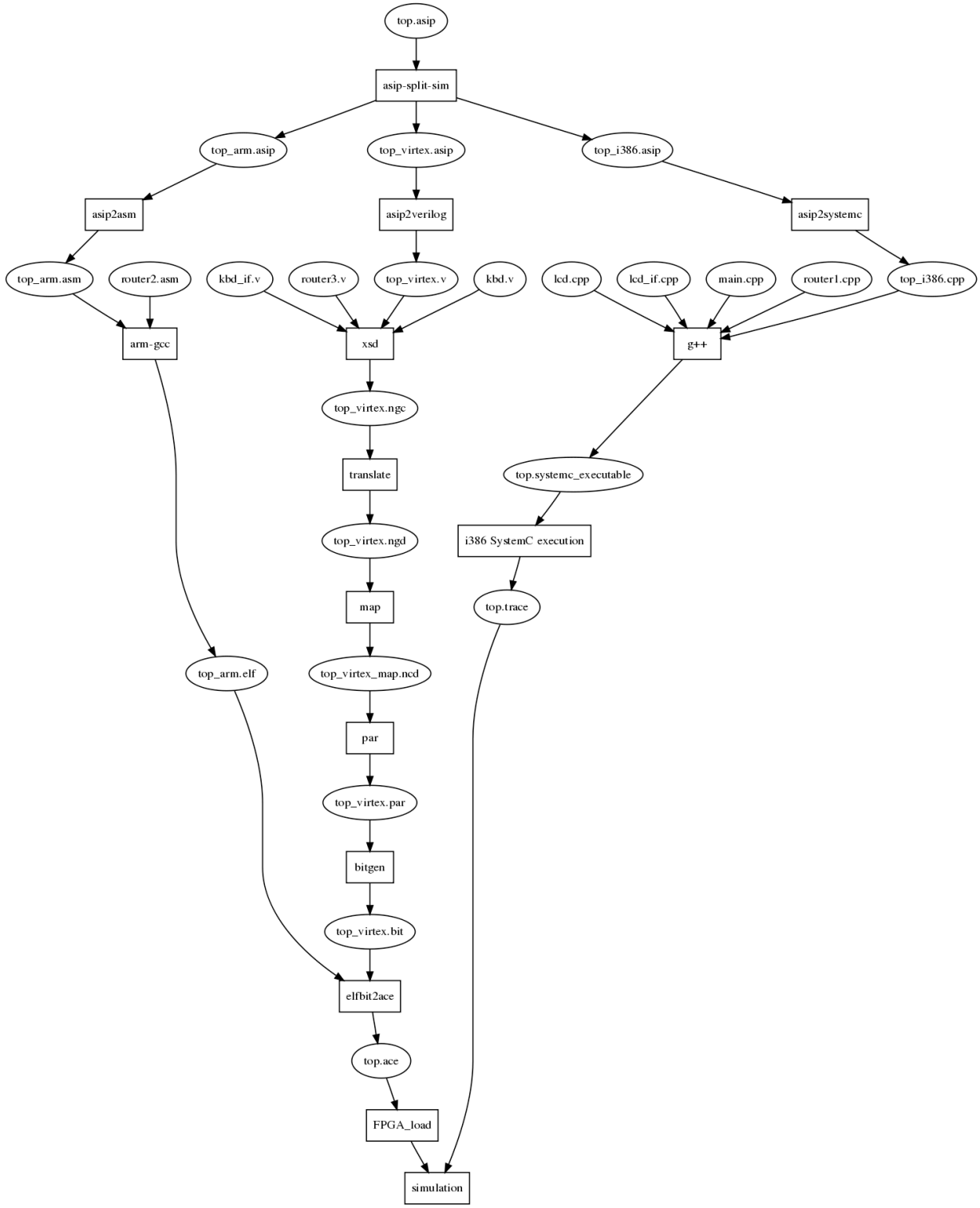

Figure 9: Galaxy Tool flow with Graphviz layout

7 TUTORIALS / DEMOS

This section is intended as a guide for our GALAXY partners to try out and discover these new tools.

The two original test cases are provided:

- G3card, a very large Verilog project with two asynchronous processors;

- The calculator example, where five components are available at various levels of abstraction, and can be mixed and sent to different simulators or FPGA targets in any combination.

7.1 TOOLS INSTALLATION AND SETUP

For this internal version of the tools, the installation process has not yet been polished, and is not tested.

The tools are provided as a source code package, which can be compiled on any UNIX system.

7.1.1 Dependencies

On recent linux/unix systems, most dependencies can be installed by using installation tools such as yum or apt.

```
yum install gtkmm24-devel xerces-c-devel expat-devel graphviz-devel
```

Only one dependency, XSD, needs to be installed manually.

The download page is: http://codesynthesis.com/products/xsd/download.xhtml

This package is only needed temporarily for the compilation of the Galaxy tools:

```
cd /tmp
wget http://codesynthesis.com/download/xsd/3.2/linux-gnu/i686/xsd-3.2.0-i686-linux-gnu.tar.bz2
tar xzf xsd-3.2.0-i686-linux-gnu.tar.bz2
cd xsd-3.2.0-i686-linux-gnu
make
```

If another directory than /tmp is chosen, the *--with-xsd* parameter in the next section will need to be adjusted.

7.1.2 Compilation and Installation

The *galaxy-tools-1.0a1.tar.bz2* file can be decompressed anywhere. It creates a *galaxy-tools-1.0a1* directory, from which one can run the usual configure, make, make install, possibly as root for the last step.

Two arguments should be passed to the *configure* script:

--with-xsd=<path-to-xsd> : indicates where XSD was (temporarily) installed

--prefix=<path> : indicates where to install the tools after compilation (/usr/local/galaxy recommended)

The final step is to adjust the PATH environment variable:

```
tar xjf galaxy-tools-1.0a1.tar.gz
cd galaxy-tools-1.0a1
./configure --with-xsd=/tmp/xsd-3.2.0-i686-linux-gnu \
            --prefix=/usr/local/galaxy
make
sudo make install
export PATH=$PATH:/usr/local/galaxy/bin
```

7.1.3 Testing

Correct operation can be checked by running *galaxy-ide* on an example:

```
cd test/g3card
galaxy-ide –i g3card.asip.xml
```

7.2 CALCULATOR EXAMPLE

```
cd test/calculator
galaxy-ide –i calculator.asip.xml
```

Five components are available at various levels of abstraction, and can be mixed and sent to different simulators or FPGA targets in any combination.

7.3 G3CARD EXAMPLE

```
cd test/g3card
galaxy-ide –i g3card.asip.xml
```

Figure 10 (and Figure 7, page 16) are screenshots of the G3card project as implemented in the Galaxy tools. This example is only mentioned here as it has been the main development test case for the IDE, and is a good demonstrator of the design navigation features due to its large number of components. Simulation files are not distributed for this design.

Use the mouse wheel to zoom in/out and expand/collapse sub-components using Control or Alt + mouse wheel. Hours of fun!

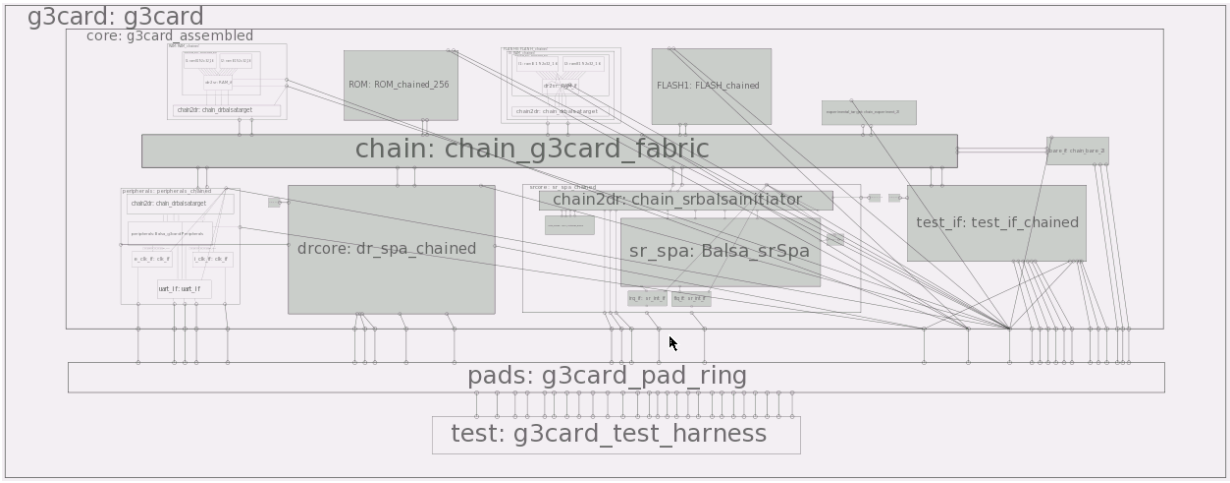

Figure 10: G3card example

7.4 INTEGRATED TUTORIAL

```
cd test/tutorial
galaxy-ide
```

In the *Help* menu, starting "*Feature 1: Introduction to ASIP components*" will start the demo, automatically going through all the features visible in the *Help* menu.

The tutorial automatically stops whenever the user moves the mouse.

To resume the tutorial after interrupting it, restart the current feature by selecting it in the *Help* menu.

8 SUMMARY

These new GALS design tools are released to the public under the GPL v2 license.

We are very interested to hear about your experience.