ALLIANCE & CORIOLIS Checker Toolkit

Contents

Alliance & Coriolis Checker Toolkit	1
Toolkit Purpose	1
Toolkit Contents	1
Benchmark Makefiles	2
Coriolis Configuration Files	2
Coriolis and Clock Tree Generation	3
rhel6 and Clones	3
Benchmarks Special Notes	3
alliance-run	3
Tools & Scripts	3
Pad Layout Converter px2mpx.py	3

Toolkit Purpose

This toolkit has been created to allow developpers to share through git a set of benchmarks to validate their changes in Alliance & Coriolis before committing and pushing them in their central repositories. A change will be considered as validated when all the developpers can run successfully all the benchs in their respective environments.

As a consequence, this repository is likely to be *very* unstable and the commits not well documenteds as they will be quick corrections made by the developpers.

Toolkit Contents

The toolkit provides:

• Three benchmark designs:

Design	Technology	Cell Libraries
adder	MOSIS	msxlib, mpxlib, msplib
AM2901	ALLIANCE dummy	sxlib, pxlib
alliance-run (AM2901)	Alliance dummy	sxlib, dp_sxlib, padlib
SNX	MOSIS	msxlib, mpxlib, msplib

• Three cell libraries.

All thoses libraries are for use with the Mosis technology. We provides them as part of the toolkit as we are still in the process of validating that technology, and we may have to perform quick fixes on them. The design are configured to use them instead of those supplied by the Alliance installation.

- msxlib: Standard Cell library.
- mpxlib: Pad library, compliant with CORIOLIS.
- msplib: Pad library, compliant with ALLIANCE / ring. Cells in this library are wrappers around their counterpart in mpxlib, they provides an outer layout shell that is usable by ring.
- The RDS file for the MOSIS technology scn6m_deep_09.rds, for the same reason as the cell libraries.
- Miscellenous helper scripts.

UPMC/LIP6

Benchmark Makefiles

The main body of the Makefile has been put into benchs/etc/rules.mk. It provides the following targets:

	core.ap	The placement of the design's core
ALLIANCE	\$(CHIP)_alc.ap	The complete layout of the design (P&R).
	druc	Symbolic layout checking
	lvx	Perform LVS.
	graal	Launch graal in the Makefile 's environement
	dreal	Launch dreal in the Makefile's environement, and load
		the gds file of the design.
	\$(CHIP)_crl.ap	The complete layout of the design (P&R).
Coriolis	druc-crl	Symbolic layout checking
	lvx-crl	Perform LVS.
	cgt-interactive	Launch cgt and prep it to perform P&R
	cgt	Launch cgt in the Makefile 's environement

A top Makefile in a bench directory must define at least the following variables:

```
CORE = adder
CHIP = chip
MARGIN = 2
GENERATE_CORE_VST = Yes
USE_MOSIS = Yes
```

include ../etc/rules.mk

export	$MBK_IN_LO = vst$
export	$MBK_OUT_LO = vst$
export	$RDS_IN = gds$
export	$RDS_OUT = gds$

Where variables have the following meaning:

Variable	Usage
CORE	The name of the <i>core</i> model
CHIP	The stem of the <i>chip</i> model. It is declined in two versions, one for Alliance (suffix _alc) and one for Coriolis (suffix _crl). This is needed because the two core uses different sets of pads.
GENERATE_CORE_VST	Tells if the rules to generate the core has to be included. If set to No, then the core <i>must</i> be present and will be considered as a primary file.
USE_MOSIS	Tells whether or not use the MOSIS technology.

CORIOLIS Configuration Files

Unlike Alliance which is entirely configured through environement variables or system-wide configuration file, Coriolis uses configuration files in the current directory. They are present for each bench:

• <cwd>/.coriolis_techno.conf : Select which symbolic and real technology to use.

UPMC/LIP6 2

<cwd>/.coriolis.conf : Override for any system configuration, except for the technology.

CORIOLIS and Clock Tree Generation

When CORIOLIS is used, it create a clock tree which modificate the original netlist. The new netlist, with a clock tree, has a postfix of _clocked.

Note



Trans-hierarchical Clock-Tree. As CORIOLIS do not flatten the designs it creates, not only the top-level netlist is modificated. All the sub-blocks connected to the master clock are also duplicateds, whith the relevant part of the clock-tree included

RHEL6 and Clones

Under RHEL6 the developpement version of CORIOLIS needs the devtoolset-2. rules.mk tries, based on uname to switch it on or off.

Benchmarks Special Notes

alliance-run

This benchmark comes mostly with it's own rules and do not uses the ones supplieds by rules.mk. It uses only the top-level configuration variables.

It a sligtly modified copy of the alliance-run found in the ALLIANCE package (modification are all in the Makefile). It build an AM2901, but it is splitted in a control and an operative part (data-path). This is to also check the data-path features of ALLIANCE.

And lastly, it provides a check for the CORIOLIS encapsulation of ALLIANCE through PYTHON wrappers. The support is still incomplete and should be used only by very experienced users. See the demo* rules.

Tools & Scripts

Pad Layout Converter px2mpx.py

The px2mpx.py script convert pad layout from the pxlib (ALLIANCE dummy technology) into mpxlib (MOSIS compliant symbolic technology).

Basically it multiplies all the coordinate by two as the source technology is 1 μ type and the target one a 2 μ . In addition it performs some adjustement on the wire extension and minimal width and the blockage sizes.

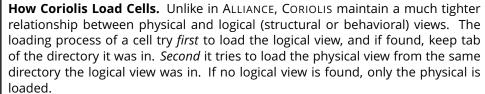
As it is a one time script, it is heavily hardwired, so before using it do not forget to edit it to suit your needs.

The whole conversion process is quite tricky as we are cheating with the normal use of the software. The steps are as follow:

- 1. Using the Alliance dummy technology and in an empty directory, run the script. The layouts of the converted pads (*_mpx.ap) will be created.
- 2. In a second directory, this time configured for the MOSIS technology (see .coriolis2_techno.conf) copy the converted layouts. In addition to the layouts, this directory must also contain the behavioral description of the pads (.vbe). Otherwise, you will not be able to see the proper layout.
- 3. When you are satisfied with the new layout of the pads, you can copy them back in the official pad cell library.

UPMC/LIP6

Note





Conversely, when saving a cell, the directory it was loaded from is kept, so that the cell will be overwritten, and not duplicated in the working directory as it was in ALLIANCE.

This explains why the behavioral view of the pad is needed in the directory the layouts are put into. Otherwise you would only see the pads of the system library (if any).

UPMC/LIP6 4