

## ALLIANCE & CORIOLIS Checker Toolkit

### Contents

<b>ALLIANCE &amp; CORIOLIS Checker Toolkit</b>	<b>1</b>
Toolkit Purpose	1
Toolkit Contents	1
Benchmark Makefiles	2
CORIOLIS Configuration Files	3
CORIOLIS and Clock Tree Generation	3
RHEL6 and Clones	3
Benchmarks Special Notes	3
alliance-run	3
Tools & Scripts	4
One script to run them all: go.sh	4
Command Line cgt: doChip.py	4
Pad Layout Converter px2mpx.py	4

### 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 benches in their respective environments.

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

### Toolkit Contents

The toolkit provides:

- Six benchmark designs:

Design	Technology	Cell Libraries
adder	MOSIS	msxlib, mpxlib, msplib
AM2901 (standard cells)	ALLIANCE dummy	sxlib, pxlib
AM2901 (datapath)	ALLIANCE dummy	sxlib, dp_sxlib, pxlib
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.

## Benchmark Makefiles

The main body of the Makefile has been put into `benchs/etc/rules.mk`.

The Makefile in the various bench directories provides some or all this targets, according to the fact they can be run with CORIOLIS, ALLIANCE or both.

ALLIANCE	<code>layout-alc</code>	The complete layout of the design (P&R).
	<code>druc-alc</code>	Symbolic layout checking
	<code>lvx-alc</code>	Perform LVS.
	<code>graal</code>	Launch <code>graal</code> in the Makefile 's environnement
	<code>dreal</code>	Launch <code>dreal</code> in the Makefile 's environnement, and load the <code>gds</code> file of the design.
CORIOLIS	<code>layout</code>	The complete layout of the design (P&R).
	<code>druc</code>	Symbolic layout checking
	<code>lvx</code>	Perform LVS.
	<code>view</code>	Launch <code>cgt</code> and load the design (chip)
	<code>cgt</code>	Launch <code>cgt</code> in the Makefile 's environnement

A top Makefile in a bench directory must looks like:

```

        CORE = adder
        CHIP = chip
        MARGIN = 2
GENERATE_CORE_VST = Yes
USE_CLOCKTREE = No
USE_MOSIS = Yes
USE_DEBUG = No

include ../etc/rules.mk

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

check:      lvx

layout:      chip_crl_kite.ap
lvx:         lvx-chip_crl_kite
druc:        druc-chip_crl_kite
gds:         chip_crl_kite.gds
view:        cgt-view-chip_crl_kite

lvx-alc:     lvx-chip_alc
druc-alc:    druc-chip_alc
```

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 <code>_alc</code> ) and one for CORIOLIS (suffix <code>_crl</code> ). 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 <code>No</code> , then the core <i>must</i> be present and will be considered as a primary file.
USE_CLOCKTREE	Adds a clock-tree to the design (CORIOLIS).
USE_MOSIS	Tells whether or not use the MOSIS technology.
USE_DEBUG	Activate debug support on <code>cgt</code> .

### CORIOLIS Configuration Files

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

- `<cwd>/coriolis2/techno.py`: Select which symbolic and real technology to use.
- `<cwd>/coriolis2/settings.py`: 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 modifcated. 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 slightly 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

### One script to run them all: `go.sh`

To call all the bench's `Makefile` sequentially and execute one or more rules on each, the small script utility `go.sh` is available. Here are some examples:

```
dummy@lepka:bench$ ./bin/go.sh clean
dummy@lepka:bench$ ./bin/go.sh lvx
```

### Command Line `cgt`: `doChip.py`

As a alternative to `cgt`, the small helper script `doChip.py` allows to perform all the P&R tasks, on an stand-alone block or a whole chip.

### 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 $\mu$  type and the target one a 2 $\mu$ . In addition it performs some ajustement 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.

#### Note

**How Coriolis Load Cells.** Unlike in ALLIANCE, CORIOLIS maintain a much tighter relationship between physical and logical (structural or behavioral) views. The loading process of a cell try *first* to load the logical view, and if found, keep tab of the directory it was in. *Second* it tries to load the physical view from the same directory the logical view was in. If no logical view is found, only the physical is loaded.

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).

