Using **Alliance** **Tutorials**

Abstract

*These tutorials introduce the design flow to be used in the **Alliance** CAD framework for the design and verification of a standard cells circuit, including the pads. Each step of the desgin flow is supported by one or more specific tools, whose use is briefly explained in the tutorials.*

These texts are meant to be simple and comprehensive, and are to be used to get into the system. Should something be unclear or wrong, please indicate this by sending an e-mail to `alliance-users@asim.lip6.fr`.

1   Introduction

In these tutorials, you will learn the practical use of some basic **Alliance** tools by building some very simple circuits from scratch. It is recommended that you read the `overview.pdf` file before proceeding, as it describes the main steps of the design conceptually.

2   Before starting

In those tutorials you will learn the practical use of the following **Alliance** tools :

In the first tutorial (simulation/ directory) :

- **asimut** : VHDL compiler and simulator.

- **genpat** : patterns generator.

- **genlib** : Netlist capture.

- **xpat** : Graphical pattern viewer.

In the second tutorial (place_route/ directory) :

- **ocp** : Standard cell placer.

- **ocr** : Over cell router (obsolete)

- **nero** : Over cell router

- **druc** : Design rule checker.

- **ring** : Core to pads router.

- **cougar** : Symbolic layout extractor.

- **lvx** : Netlist comparator.

- **graal** : Graphic layout editor.

In the third tutorial (synthesis/ directory) :

- **syf** : Finite state machine synthesizer.

- **boom** : Boolean optimization of a logic level behavioral description (VHDL data flow).

- **boog** : Mapping of a behavioral descriptiononto a standard cell library.

- **loon** : Fanout optimizer, global optimizer and timing analyser of

- **scapin** : Scan Path insertion

- **xsch** : Graphical schematic viewer.

If you run a `c-like` shell, like `csh` or `tcsh`, try to run the following command :

```
˜alp/addaccu %-) source /alliance/etc/alc_env.csh
```

Otherwise, if you run a `sh-like` shell, try to run the following command :

```
˜alp/addaccu %-) source /alliance/alc_env.sh
```

Before we proceed to the tutorial, you must make sure that the **Alliance** tools are readilly available when invoking them at the prompt. The prompt in represented in the following text by the symbol :

```
˜alp/addaccu %-)
```

In this system, `alp` is the user, `addaccu` is the current directory, and `%-)` is supposed to give us courage!

3   Execution environment set up

Later, before you will start examining alliance tools, you will probably want to know the environment variables setup. To see it, please enter the following command :

```
˜alp/addaccu %-) env | grep MBK
```

```
˜alp/addaccu %-) env | grep MBK
MBK_OUT_PH=ap
MBK_CATAL_NAME=CATAL
MBK_SCALE_X=100
MBK_VSS=vss
MBK_CATA_LIB=.:/alliance/cells/sxlib:/alliance/cells/padlib
MBK_WORK_LIB=.
MBK_VDD=vdd
MBK_C4_LIB=./cellsC4
MBK_IN_LO=vst
MBK_IN_PH=ap
MBK_TARGET_LIB=/alliance/cells/sxlib
MBK_OUT_LO=vst
```

Figure 1: MBK environment variables.

All these variables are documented at least with a manual page. However, some variables are documented in each tutorial.

4 File Formats

One of the interesting features of **Alliance** is that different file formats can be used for both netlist and layout view. However, in the design methodology we wish to promote, some formats are recommended. The vst, structural **VHDL**, is dedicated to netlist specification. The al format is dedicated to extracted layout representation. The ap format is the usual layout format.

So, prior to generate a specification netlist, you shall type:

```
˜alp/addaccu %-) setenv MBK_OUT_LO vst
```

Otherwise, if you are running a sh-like shell:

```
˜alp/addaccu %-) MBK_OUT_LO=vst; export MBK_OUT_LO
```

But if you wish to extract a netlist from the layout then you'll do:

```
˜alp/addaccu %-) setenv MBK_OUT_LO al
```

You are now ready to actually do all tutorials.