datapath Compiler, a timing analyzer, an interactive generator debugger, full-custom generators, and a test coverage tool. Ongoing researches are also experienced in the design methodology management domain.

8. Acknowledgements

The authors would like to thank all the teachers, permanent researchers, and students more or less involved in the building of the ALLIANCE CAD system. Special thanks are granted to P. Allègre, P. Bazargan, L. Burgun, A. Derieux, G.E. Descamps, M. Hirech, L. Lucas, H. Mehrez , F. Pétrot, V. Pouilley, H. Rejouan, H-N. Vuong, and F. Wajsbürt. F. Pêcheux also thanks E. Carletti and C. Gherardi for their efficiency in the making of the distribution tape.

9. References

[1] A. Greiner, M. Laurentin, R. Marbot, "DESB", Proc. EuroDAC, Hamburg, September 1992.
[2] A. Greiner and J.P. Leroy, "A symbolic Layout View in EDIF for Process Independent Design," 4th European EDIF Forum, November 1990.
[3] A. Compan, A. Greiner, F. Pêcheux, F. Pétrot, "GENVIEW: A Portable Source-Level Debugger For MacroCell Generators", Proc. European Design Automation Conference, February 1991.
[4] J. Médou Zengue Ze, "Vérification automatique des règles de dessin des circuits VLSI: règles formelles, approche hiérarchique," UPMC Phd thesis, defended on the 21st october 1991.
[5] A. Amara, A. Derieux, A. Greiner, "Conception d'un microprocesseur de quatre bits: de la spécification à la réalisation," Annales de télécommunication 46, n° 9-10, 1991.
[6] B. Zerouk, "Etude et Réalisation d'un processeur SIMD," Rennes 1 Phd thesis, defended on the 5th may 1992.
[7] A. Compan, P. Debaud, V. Delorme, H. Mehrez, J.A. François, F. Pêcheux, "GAF: A Floating-point adder generator using CXgen function library," Proc. Euromicro, Vienna, 1991.
[8] C. Grosjean, L. Lucas, F. Wajsbürt, L. Winckel, "STACS Architecture Internal Report", 1992.
[9] Cadence, Edge Design Framework Reference Manual, 1989.
[10] Compass, Chip Compiler Reference Manual, 1989.

6. Experience and results

The ALLIANCE System has been extensively used during the past three academic years (89-90, 90-91 and 91-92) as a practical support of an undergraduate course on CMOS VLSI design. This initiation course lasts 13 weeks with a 2 hours lecture and 4 hours spent using the ALLIANCE system per week, and involves 60 students and 3 teachers. Students have no previous knowledge on VLSI design and mainly come from two distinct channels: "computer science" and "electrical engineering" masters of sciences teached at the UPMC. During this course, students are required to design and implement an AMD 2901 compatible processor, starting from a commercial data-sheet. The chip, which complexity is about 2000 transistors, is designed by groups of 2 or 3 students [5].

Academic year	Number of AMD2901 Projects	Number of fully achieved projects	Technology
89-90	20	9	2 mic. ES2
90-91	22	13	1.5 mic ES2
91-92	32	23	1.2 mic ES2

Figure 4: Undergraduate projects and results.

Figure 4 shows that, with the passing years, the number of projects increases, as well as the fully achieved (designed and validated) AMD2901 projects, while the technology lambda still decreases. The VHDL language was prefered to GenRAD hilo during 91-92, when ASIMUT became available. The ALLIANCE system is also used for post-graduate projects. These projects range from medium complexity ASICs developed in 6 months by a couple of designers (DATA-SAFE, TNT, SMALL [6], etc...) to high complexity microprocessors (FRISC [7], STACS [8]) developed by a pool of PHD students. Figure 5 describes the various complexities of the chips built at UPMC.

Project	# of transistors	Functionality
SMAL [6]	17 000	One bit processor for SIMD architectures
DATA-SAFE	35 000	Dynamic data encryption chip
TNT	60 000	Switch-router for T800 transputers
FRISC [7]	200 000	Floating-point RISC microprocessor
STACS [8]	800 000	Static control superscalar microprocessor

Figure 5: Various complex chips designed with ALLIANCE.

The three largest circuits described in Figure 5 use not only standard-cells but also parameterized generators for regular blocks like RAMs, data-paths, or floating-point operators. The FRISC and TNT projects successfully used the CADENCE [9] and COMPASS [10] place and route tools, and therefore prove the interoperability of the ALLIANCE system.

7. Conclusion

We are very happy to use a proprietary set of tools for teaching CMOS VLSI design for two good reasons. First, we simply can't afford 50 high end workstations to run commercial CAD systems like Mentor Graphics or Cadence. Second, the Cadence system has been used in research project at MASI. It is a powerful and sophisticated environment but is much too complex for novice undergraduate students. The great advantage of the ALLIANCE CAD system is that we have done our best to stick to the basic concepts of VLSI design. We tried very hard not to propose highly parameterized tools. At last, we experienced that the technology migration and process independence are key issues. Hence, it is crucial to rely on a portable standard-cell library and on portable generators. The next release of ALLIANCE will probably contain a

format. The resulting netlist can either be hierarchical or flattened (transistor netlist). CIF and GDS2 input formats are also supported.

LVX is a logical versus extracted net-compare tool. Supported formats for both input files are EDIF, SPICE, VHDL and ALLIANCE internal format. The resulting boolean indicates if an error has occured or if the two netlist match together.

DESB is a functional asbtractor/disassembler for CMOS circuits [1]. It provides a VHDL Data-Flow behavioral description from the transistor netlist of a circuit, by extracting a pseudo-gate netlist of "cones". The input file is a - possibly extracted - flattened transistor netlist. The output is a simulable behavioral VHDL model (data-flow without timing informations). DESB can be distinguished from commercial CAD abstractors by the fact that it does not need a predefined cell library or transistor patterns.

PROOF performs a formal comparison between two behavioral VHDL descriptions. PROOF supports the same subset of VHDL as ASIMUT, LOGIC and DESB. It is based on ordered binary decision diagrams (Bdds).

ALC is an hierarchical symbolic layout editor. It requires a X-Window graphical environment (X11R5). ALC is used for cell layout design or hierarchical block construction. It provides on-line DRC and automatic display of equipotential nets.

MBK2PS creates a Postscript file from a symbolic layout cell.

LOGIC is a logic synthesis tool. The input file is a behavioral description of the circuit using the same VHDL subset as the logic simulator. The output is netlist of gates (EDIF, VHDL, ALLIANCE). LOGIC can be mapped to any Standard-Cell library, as long as a VHDL is provided with each cell. LOGIC can also run without a predefined standard-cell library, thanks to an internal cell compiler.

4. Supported exchange formats

Figure 3 describes all the VLSI description formats the ALLIANCE CAD system handles.

	Behavioral view	Structural view	Physical view
Parsers	• VHDL	• ALLIANCE • EDIF • COMPASS • SPICE	• ALLIANCE • COMPASS
Drivers	• VHDL	• ALLIANCE • EDIF • COMPASS • SPICE • HILO	• ALLIANCE • COMPASS • GDSII • CIF

Figure 3: different supported VLSI description formats.

5. ALLIANCE internal architecture

The complete ALLIANCE CAD system is about 500 000 lines of C. For now, it runs on a SparcStation with SunOs 4.0 or later, and requires the basic X-Window library X11. The distribution tape shows that there are three kinds of programs:

- common data structures and manipulation primitives
- parsers/drivers for read/write external file formats
- actual tools

The ALLIANCE package has been distributed in several french and foreign universities and is available through anonymous FTP (including on-line documentation and examples) at **ftp-masi.ibp.fr** . For further information, please look in the **pub/cao-vlsi** directory.

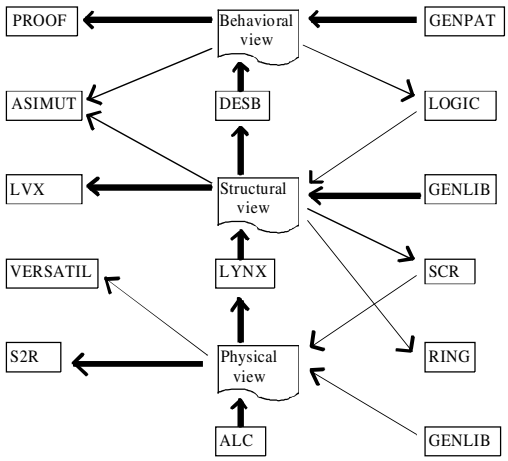

Figure 2: the three basic views and tool interoperability.

ASIMUT is a VHDL logic simulator. The supported VHDL subset allows both structural and behavioral data-flow description (without timing information). Complex microprocessors, including INTEL 8086 and MIPS R3000 have been successfully simulated with ASIMUT. ASIMUT is based on an event-driven algorithm and powerful representation of boolean functions.

GENPAT is a language interpreter dedicated to efficient simulation pattern descriptions. It generates an ASCII file that can act as an input of ASIMUT. A GENPAT file format to MSA translator allows the generation of appropriate simulation patterns for the Tektronix LV500 tester.

SCLIB is a CMOS Standard-Cell library that contains about 60 different cells. To each cell correspond a VHDL model, a transistor-level schematic (SPICE or EDIF format) and the related symbolic layout file (ALLIANCE format).

GENLIB is a procedural language for netlist capture and placement description (there is no schematic editor in the ALLIANCE system). GENLIB provides a consistent set of C primitives, giving the designers the ability to describe VLSI circuit netlists in terms of terminals, signals and instances, or circuit topologies in terms of placement of abutment boxes. These primitives work in a coordinated manner: first, the view is opened ; second, design actions are performed ; third, the view is closed. GENLIB is actually used to build parameterized generators [3].

SCR is a place and route tool for Standard-Cells. The placement system is based on simulated annealing. Feed-throughs and power routing wires are automatically inserted where needed. The input netlist can be any of the following: structural VHDL, EDIF, or ALLIANCE internal format. The output is either an hierarchical (channels are instanciated) or flattened (channels are inserted) chip core layout without external pads.

RING is a specific router dedicated to the final routing of chip core and input/output pads. RING takes into account the various problems of pad placement optimization, power and ground distribution.

S2R is the ultimate tool used in our design flow to perform process mapping. S2R stands for "symbolic to real", and translates the hierarchical symbolic layout description into physical layout required by a given silicon supplier. The translation process involves complex operations such as denotching, oversizing, gap-filling and layer adaptation. Output formats are either CIF or GDSII. S2R relies on a specific parameterized technology file.

VERSATIL is a fast Design Rule Checker. The input file is a - possibly hierarchical - symbolic layout. The VERSATIL DRC makes great use of a windowing algorithm that divides the circuit into independent partitions to enhance performances [4].

LYNX is a layout extractor. The input is a - possibly hierarchical - symbolic layout. The output is an extracted netlist with parasitic capacitances. Output formats are EDIF, SPICE, VHDL or ALLIANCE internal

and structural views. The structural description is checked against the behavioral description by using the same set of patterns.

2.3 Physical design

Once the circuit netlist has been captured and validated, the designer can opt for two solutions. Either he decides to optimize his layout and uses procedural placement functions (Reference 6), or he prefers automatic placing and routing (Reference 7). GENLIB contains procedural placement functions that allow students to design any kind of bit-sliced macro-cell topology. SCR is a Standard-Cell Placer and Router. From an input netlist and topology directives, it generates the routed chip core. The output of SCR is either flattened or hierarchical. As stated in References 9 and 10, the circuit core is now ready to be connected to external pads. The Core-to-pads router, RING, aims at doing this operation automatically, provided the user has given an appropriate netlist.

2.4 Verification

In our VLSI course, we intend to show that VLSI verification is as important as VLSI physical design. For that reason, we have introduced in the design flow powerful tools to perform behavior, netlist and layout verifications. Figure 1 shows that several operations can be achieved on the circuit layout. The core layout can be validated thanks to an hierarchical Design Rule Checker, VERSATIL, in a reasonable time. An extracted netlist can be obtained from the resulting layout (References 12 and 13). LYNX, the layout extractor operates on both hierarchical and flattened layout and can output both flattened netlist (transistor netlist) and hierarchical netlist. The transistor netlist is the input of the efficient DESB functional abstractor. DESB (Reference 14) provides a VHDL Data-Flow behavioral description from the transistor netlist of a circuit. The resulting behavior can be compared to the initial specification, thanks to the formal proof analyzer PROOF (References 15 and 16) . When extracted hierarchically, the resulting netlist (Reference 14) can be compared with the original netlist (Reference 4) by using the LVX tool (Reference 19). LVX, that stands for Logical Versus Extracted, is a netlist comparator that matches every design object found in both netlists.

2.5 Test and coverage evaluation

For now, the fault coverage provided by the functional patterns is evaluated thanks to GenRAD Hifault fault simulator.

2.6 Logic synthesis

Parallel to the proposed design flow, ALLIANCE also provides a complete and efficient Logic Synthesis system, called LOGIC. This sub-package aims at discarding the second part of the previous methodology, the design and capture of the structural view. From a VHDL behavioral description of the circuit, LOGIC is able to produce a complete gate netlist.

3. Tools of the ALLIANCE package

Every ALLIANCE tool has been designed to simply interface with each other, in order to support the proposed design flow. Nevertheless, each tool can also be used independently, thanks to the multiple standard formats used for input and output files. The process independence goal is achieved thanks to the symbolic layout approach [2]: the same Standard-Cell library SCLIB has been successfully ported to different processes (2µ 1.5µ and 1.2µ ES2, and 1.0µ Philips). One of the most important characteristics of the ALLIANCE system is that it provides a common internal data structure to represent the three basic views of a chip:

- the behavioral view
- the structural view
- the physical view

Figure 2 details how all the ALLIANCE tools are linked together around the basic behavioral, structural and physical databases.

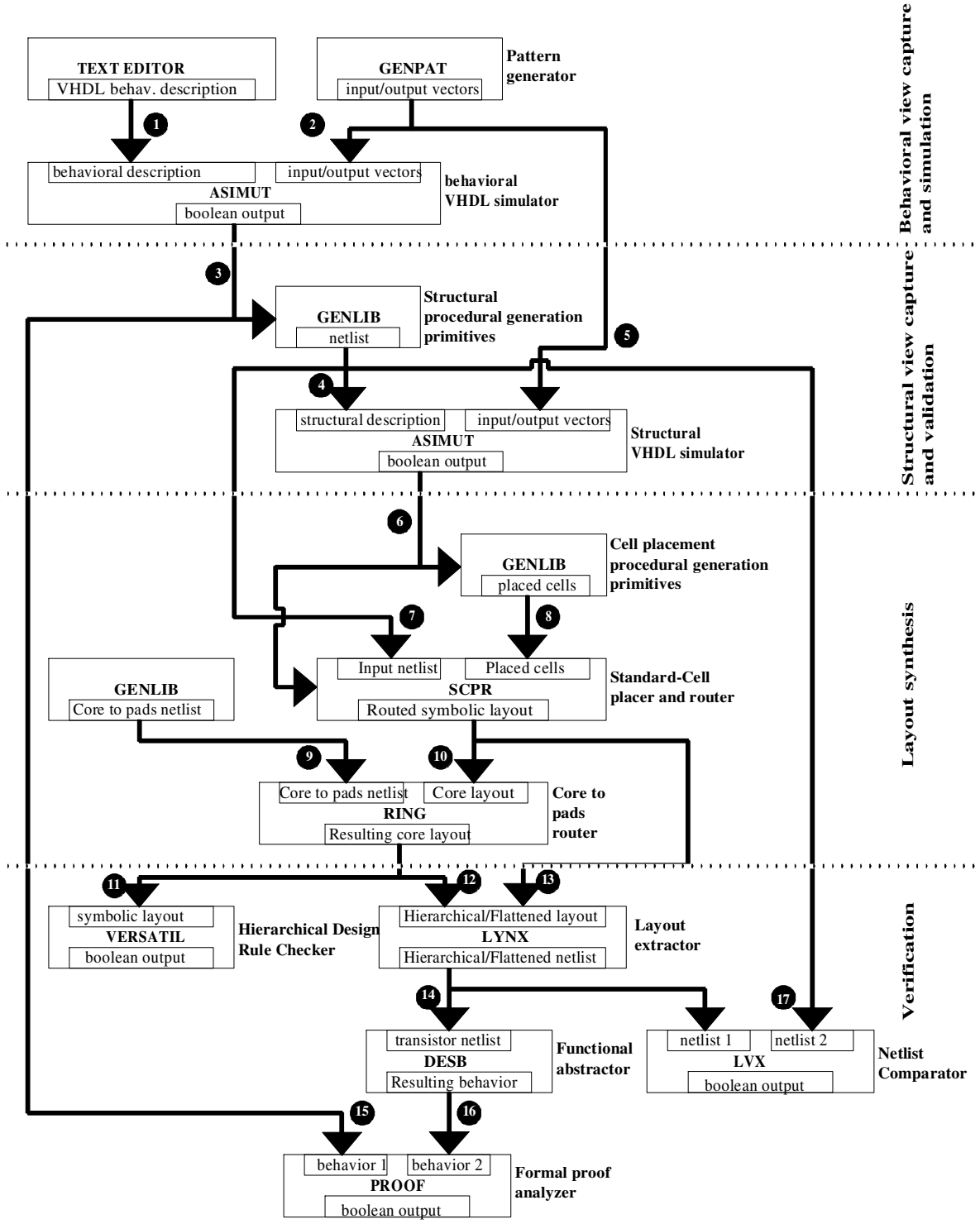

Figure 1: the ALLIANCE design flow

2.2 Capture and validation of the structural view

Reference 3 indicates that the structural view can be captured when the behavior is validated. The actual capture of the netlist relies on a specific description language, GENLIB. GENLIB is a netlist-oriented library of C functions. In the design methodology, it is essential for the students to get acquainted with the C language basics. The advantage of such an approach is that designers do not have to learn several language with specific syntax and semantics. According to References 4 and 5, ASIMUT can operate on both behavioral

According to the interoperability constraints, each ALLIANCE tool can operate as a standalone program as well as a part of the complete ALLIANCE design framework. Each ALLIANCE tool therefore supports several standard VLSI description formats (SPICE, EDIF, VHDL, CIF, GDS2). The ALLIANCE tools support a zero-default top-down design flow with not only construction tools (layout editor, automatic placer/router) but also validation tools, from design rule checker to functional abstraction and formal proof.

1.4 Compactness

Unlike commercial CAD systems, the ALLIANCE CAD Framework suits the limited ressources of low-cost workstations. The only requirements are a UNIX system with at least 8 Mbytes of memory, appropriate disk storage (30 Mbytes per user), and graphic capabilities (X-Window).

1.5 Easiness

All tools and the proposed design flow are simple to teach and to learn. In any situation, easiness and simplicity have been prefered to sophisticated approaches. From a pratical point of view, both on-line documentation (UNIX "man") and paper are provided with each tool of the ALLIANCE package.

2. Proposed Design Flow

We refer the term "design flow" as a sequenced set of operations employed in designing a complete VLSI Chip. Although the ALLIANCE system contains logic synthesis tools for education purposes, the logic synthesis tools are not part of the proposed designed flow. Logic synthesis tools can be used after the students have experienced themselves with schematic design and capture, topological problems, etc...

In the design flow, we rely on a strict definition of all the objects and design functions found in VLSI design. The design flow is based on the Mead-Conway model and is characterized by its top-down aspect. We use a library of symbolic Standard-Cells and rely on structured hierarchical approach. Figure 1 presents the basic design methodology. This picture emphasizes the top-down aspect of the design flow, and points out that our methodology is breaked up into 5 distinct parts, the latter being not shown:

- Capture and simulation of the behavioral view
- Capture and validation of the structural view
- Physical design
- Verification
- Test and coverage evaluation

On Figure 1, every tool is presented as a box and stands for a single executable program capable of performing a specific design function. A tool is characterized by its input data (upper side of the box), its output data (lower side of the box).

The design flow also includes miscellaneous tools like layout editor and Postscript plotter. Numbers on Figure 1 indicate references used in the following sections.

2.1 Capture and simulation of the behavioral view

Reference 1 details that the capture of the behavioral view is the very first step of our design flow. Any VLSI design begins with a timing independent description of the circuit with a subset of VHDL behavior primitives. Reference 2 indicates that we also provide GENPAT, a language used to describe simulation patterns.

Once a VHDL behavioral description and vectors have been determined, a functional simulation can be started, as shown in Figure 1. The behavioral VHDL simulator is ASIMUT. It validates the input behavior, according to the input/output vectors.

ALLIANCE: A complete Set of CAD Tools for teaching VLSI Design

Alain Greiner, François Pêcheux

Laboratoire MASI/CAO-VLSI, Institut de Programmation
Université Pierre et Marie Curie (PARIS VI), Tour 55-65, 2ème étage, Porte 13
4, Place Jussieu 75252 PARIS Cedex 05 FRANCE

Abstract

The ALLIANCE package is a complete set of CAD tools for teaching VLSI design. ALLIANCE aims at allowing universities to start and develop VLSI design activities without too much time and money investments. Each ALLIANCE tool can operate as a standalone program as well as a part of the complete design framework. The basic design flow is first presented, then each tool is reviewed. Experience and results conclude the paper.

1. Introduction

The ALLIANCE package is the result of ten years effort spent at the MASI Laboratory of the Pierre et Marie Curie University (UPMC), in Paris. During these years, our major goal was to provide our undergraduate and graduate students with a complete CAD framework, designed to assist them in VLSI CMOS course. The CAO-VLSI team focuses its activity on two key issues: parallel architectures using high complexity ASICs, and innovative CAD tools for VLSI design. Strong interaction exists between the Hardware Architecture team (15 PHD students and 6 permanent researchers) and the CAD team (17 PHD students and 4 permanent researchers). The main CAD action aims at fulfilling both the needs of experienced designers by providing practical answers to state-of-the-art problems (logic synthesis, procedural generation, layout verification, test and interoperability), and novice designers, by providing a simple and consistent set of tools. Our VLSI design flow is therefore based on both advanced CAD tools that are not available in commercial CAD systems, such as functional abstraction [1], and standard construction/validation tools.

1.1 Technology independence

The target is digital CMOS design. The ALLIANCE package provides some process independence in order to allow the designers to easily port their design from one silicon supplier to another. Technology independence relies on a symbolic layout approach.

1.2 Portability

The ALLIANCE package has been designed to run simultaneously on a large number of low-cost UNIX workstations under X11R5 (more than 50). Several hardware platforms, from i386 based microcomputers to SparcStations and DEC Stations, are supported.

1.3 Modularity