# Kaiserslautern's CAD Activities within the CVT Project

**Reiner W. Hartenstein**

Nr. 125/85

Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049,
D-675 Kaiserslautern, F.R.G., phone xx49 (631) 205-2606 / (7251) 3575

# Kaiserslauterer CAD-Aktivitäten im Rahmen des CVT-Projekt

Reiner W. Hartenstein

Zusammenfassung.

Das CVT-Projekt ist ein von der Europäischen Gemeinschaft geförderetes Deutsch/Französisch/Italienisches Projekt mit dem Ziel der Entwicklung eines experimentellen integrierten CAD-Systemes. CVT steht für "CAD for VLSI in Telecommunications". Kaiserslautern bearbeitet drei subtasks aus diesem Projekt, insbesondere in Zusammenarbeit mit dem Forschungszentrum CSELT (Centro Studi et Laboratori Telecommunicazioni) in Turin, und der Universität Grenoble. Nach einem kurzen Überblick über das CVT-Projekt wird ein Einblick gegeben in die in Kaiserslautern durchgeführten Arbeiten, sowohl direkt im CVT-Rahmen, als auch indirekt durch dieses angeregt.

# Kaiserslautern's Activities within the CVT Project

Kaiserslautern,
January 1985

Contents

1. The CVT Project

2. Kaiserslautern's activities within the CVT project

2.1 KARL
2.1.1 A language for the "long thin man"
2.2 ABL

3. other KARL applications within CVT

3.1 KARENE
3.2 KARL fault simulator
3.3 Test pattern generation from KARL sources
3.4 Miscellaneous

# 1. The CVT Project

CVT stands for "CAD for VLSI in Telecommunications". The German/French/Italian CVT Project, having a total budget of about 20 Million ECU, is 50% funded by the European Community. The main contractors are:

- FI im FTZ Darmstadt (Research Institute of the German Bundespost)
- CNET in Grenoble (Centre National d'Etudes de Telecommunications)
- CSELT in Torino (Centro Studi et Laboratori Telecommunicazioni)

Subcontractors are for instance the companies:

- CIT-ALCATEL
- EFCIS
- Olivetti
- SGS-ATES
- Telefunken

and, for instance, the following universities:

- Aachen
- Darmstadt
- Dortmund

- Karlsruhe
- Kaiserslautern
- Grenoble
- Bologna
- Genova
- Torino (Politecnico)

Subject of CVT research is the development of an experimental integrated CAD system for VLSI. The current official standard software environment of the CVT project is VMS on VAX-11/750. The official CVT software guidelines request a portability of all software implemented within CVT to be portable onto UNIX 4.2. The goal of this is, to achieve portability from VAXes onto work stations of the future. The CVT project is subdivided into 4 tasks:

- Architecture
- Languages and Data Structures
- Testing
- Device Modelling

# 2. Kaiserslautern's activities

In close cooperation with Aachen Technical University and CSELT (see above) Kaiserslautern is leading the following subtasks of the CVT project:

- A graphic RT (register transfer) language (ABL)
- RT Language with floor plan modularity (KARL-3)
- KARL maintenance and application support

The graphics editor, being part of the "deliverables" of the first subtask is being implemented at CSELT.

## 2.1 KARL

KARL stands for "Kaiserslautern RT Language", where RT stands for "Register Transfer Level". KARL is the official RT level language of the CVT project. It is a non-procedural language for the description of digital data path networks, mainly above the gate level (logioc design level). Non-procedural means, that KARL does not have primitives for sequence description, such as for the sequential behaviour of controllers. The only way to describe controllers in KARL is to describe a model or an implementation of it at data path network level.

*The following section, explaining motivations of the KARL-3 language design, is a version of an extended abstract having been submitted somewhere else.*

## 2.1.1 A Language for the "Long Thin Man"

This paper describes a CAD tool-box-based methodology for direct implementation of algorithms and other digital design problems onto silicon. This methodology gives an optimum support for structured design approaches, such as e. g. advocated by the Mead-&-Conway scene. Including all steps from design problem capture, design for testability, down to physical layout design and test geneneration the methodology minimizes the number of languages needed throughout the CAD system. It also provides two formal design exchange interfaces, both using the same language: one for traditional customer/ designer cooperation, (in case the manufacturer's design facilities are used), and one for customer/ manufacturer cooperation (in case of silicon foundry operation).

One important ingredient of the methodology is a hardware description language, especially taylored for efficient use by the type of designer mentioned above. This "long thin man" has much more innovative power, since he is competent in both, the application field, as well as in physical layout design (also see [1,2]). He has a feeling, wether an algorithm is a good candidate for being "cast into silicon".

In many cases, such as e. g. the silicon implementation of the inner loop of an algorithm (an example is found in [3]), the most important conceptual step is the development of a "key cell" with array capabilities. This means to plunge down locally from the register transfer (RT) level down under the surface of circuit design.

Only a few transistors have to be handled, and only clocked digital behaviour is of interest, to find out wether the circuit idea meets the more global design problem logically. A typical circuit simulator means an overkill, since timing and analogue performance data are not needed at this stage of the design process. What is needed is a very simple form of mixed mode simulation, where one or a few small circuit level descriptions are needed, locally embedded into a larger RT level description.

So we extended the KARL-2 language in use for years [4, 5 ] to KARL-3 [6, 7] by adding only a very few primitives (to keep KARL simple) for:

- declaration of (hierarchies of) rectangular cells
- instantiation of cells including mirror and rotate transforms
- horizontal and vertical abutment to connect those cells
- a volative memory element for dynamic logic

All other primitives needed for this short "plunge down" have been available already within the old KARL-III, such as e. g. the technology-independant bus concept including a very simple "enables" operator useful to model:

- pass transistors (NMOS),
- transfer gates (CMOS),
- three-state output stages (bipolar),
- bus drivers (logic or RT level)

and others. For illustration see [4-7]. Also three types of ports had been available before: inputs, outputs, and bidirectional ports. So an important second stage within the described methodology is relatively simple.

The first phase of the design process which we call "design problem capture" uses the same language to provide a verified formal specification of the design problem. In this case one only uses the functional description capabilities of KARL. A simulation in this phase may verify, wether the specification is feasible, bug-free, and, wether it meets the design problem. The debugged KARL specification provided by this first stage of the deign

process may be also used as an exchange format to convey the design problem over to a design team. The HAFO IC manufacturer (Sweden) used this exchange interface to accept design problems from customers running a simulator, such as, for instance, to design a stack-oriented microprocessor for LM Ericson (Sweden).

The result of the second stage of the system, where the conceptual phase of structural design is carried out, may also be used as an interchange format to pass it to a layout designer. This physical design may be done on a CAD system purchased from somewhere else. The best use of the methodology described here, however, in our opinion is, when a "long thin man" or a "long thin person" uses the entire system by himself, or herself.

Another part of the methodology is a strategy of "accompanying test patterns" described in a language suitable for both, simulator activation and testing. Since the methodology uses no silicon compilation nor other methods providing some sort of "correctness by construction", the consitency of descriptions should be checked throughout all levels having been explained above. The "accompanying test patterns" may be used to find out, wether descriptions of different levels exhibit the same behaviour. Finally these patterns may be preserved for its use in testing. The test pattern generation algorithm to be used here is described somewhere else [ 8 ].

## Literature:


[1] R. Hartenstein: Die "Neue Mikroelektronik", GI annual conference, Kaiserslautern, 1981, Springer-Verlag, 1981

[2] R. Hartenstein Shared Cultures: CIF Library, Starting Frames and Scalabl Design Rules; NATO Advanced Study Institute on Design Methodologies for VLSI, Louvain-La-Neuve, Belgium, 1980, Stijt & Noordhoff, Publishers, 1981

[3] R. Hartenstein — The evolution of a planar algorithm: a history of design decisions; report, Kaiserslautern Univ., 1983

[4] R. Hartenstein, P. Liell — KARL-II Reference Manual, third edition, Kaiserslautern, 1983

[5] N.N. — KARL-II Instant, version no. 2, Kaiserslautern, 1983

[6] R. Hartenstein, K. Lemmert — KARL-III Reference Manual, Kaiserslautern, 1984

[7] N.N. — KARL-III Instant, Kaiserslautern 1984

[8] R. Hartenstein, A. Wodtko, — Automatic Generation of Functional Test Patterns from RT Language Source; (submitted for this conference)

## 2.2 ABL

*Preface: This section is a version of an extended abstract having been submitted somewhere else, entitled "ABLED a RT Level Schematics Editor and Simulator User Interface".*

At logical design level it is useful to have both, a symbolic notation, and a graphic diagram notation. This would also be useful at RT level. Especially for design methodologies of structured VLSI design would be efficiently supported by such a notation. However, since a graphic RT level notation is not yet widely available, the task of its development and its implementation has been adopted by our group under partial funding by the European Community within the CVT project..

The paper describes a schematics editor for the graphic RT language ABL (A Blockdiagram Language), having been designed in the seventies [11], and having been specified for computerization in 1983 [12]. Now the ABL editor (ABL = ABL EDitor [13 - 15]) is almost completed, running under VMS on VAX-11/750. The paper gives a survey on the computerized version [2] of ABL (called ABL-III), using a few illustrative application examples. It also gives insight into the structure of the editor.

A second part of the paper describes the integration of the ABL editor into the KARL software system. By means of an ABL-to-KARL translator [16] the editor is used as an interactive graphics user interface to the KARL system (a paper about KARL has been submitted separately for this symposium). So the ABLED is used to supply a RT level hardware description input to the simulator from schematics data having been entered to the system interactively .

This integration of ABLED into the system also includes a feedback of simulation results back into the schematics drawing on the screen. Also simulator commands and stimuli, which are

not part of the hardware description, may be entered via the ABLED user's screen. Some sort of "mini windows" concept has been incorporated into the ABLED to place little communication text "labels" at proper locations within the schematics on the screen.

Since ABL (as well as its counterpart KARL-III [17]), has floor plan expression capabilities along with the usual cell instantiation transforms, as well as simple circuit level behavioural primitives primitives, we believe, that it is ang excellent CAD tool to support the "Long Thin Man's" [17] typical way of structured VLSI design.

## *Literature*


[11] R. Hartenstein — Fundamentals of Structured Hardware Design, North Holland Publ. Co., Amsterdam/New York, 1977

[12] G. Girardi, R. Hartenstein — ABL Specification (draft), report, Kaiserslautern/Torino, 1983

[13] G. Girardi — ABL editor: cell definition, CVT report, Torino, Italy, Jan. 1984

[14] G. Girardi — ABL data structure, CVT report, Torino, March 1984

[16] U. Welters — ABL2KARL translator: algorithm description, CVT report, Kaiserslautern, 1984

[17] R. Hartenstein, K. Lemmert — A Design Language for the Long Thin Man; (submitted for this symposium)

# 3. Other KARL applications within CVT

KARL as the official RT language of the CVT project has a number of interfaces to other parts of the CVT software package being implemented. The following sections give a few remarks only on some of these KARL applications.

## 3.1 KARENE

KARENE is a KARL extension designed by a research group at the Univerity at Grenoble, France. KARENE adds to KARL such language primitives, which allow to express sequential behaviour of controllers, such as e. g. sequential machines, or, microprogrammed controllers. KARENE is being implemented in Grenoble in such a way, that a KARENE precompiler produces KARL source code. That's why the KARL simulator can be used also to simulate hardware described in KARENE. So the algorithmic levels of hardware descriptions, above the non-procedural RT level, may be reached.

## 3.2 KARL fault simulator

By modification of the KARL compiler/simulator (having been implemented in Kaiserslautern) the Olivetti Research Center at Ivrea, Italy (in cooperation with Aachen Technical University), produced a KARL fault simulator. This may be used to check the quality of test patterns. A fault extractor, being implemented at CSELT, will produce a table of possible faults from layout descriptions expressed in CIF format (Caltech Intermediate Form). This way a realistic fault catalogue may be produced automatically for a particular technology.

## 3.3 Test pattern generation from KARL sources

Within the "Testing" task of CVT, and, in Kaiserslautern, also an automatic test pattern generator is being designed using KARL descriptions as an input source. The goal is, that for each digital circuit being described in KARL a go/no go test may be generated automatically. The research environment, described in section 3.2 is a basis for this activity. Since KARL is the core of a couple of other software packages within the CVT package (see above and below), the power of this test generation facility will go far beyond the range of hardware to be described in KARL.

## 3.4 Miscellaneous

Other applications of KARL within CVT are: simulation of microprogrammed computer structures (implementation: a retargetable microprogram compiler is generating KARL source code), simulation of hardware from symbolic layout formats (implementation: a KARL superset "hyperKARL" also expressing personality matrixes of PLAs, PALs, Kolte Arrays, and other symbolic gate array formats, is translated into KARL by a filter program), simulation of highly parallel hardware (a Petri net-like notation, expressed in a pseudo symbolic layout format to be easily derived from tabular Petri net notations, is translated from hyperKARL into KARL source code).

This by far is no complete list of KARL applications within CVT. All this work is mainly done by CVT partners outside Kaiserslautern. One exception from this is the hyperKARL filter program, which has been implemented in Kaiserslautern. Another task of Kaiserslautern is the KARL maintenance, as well as application support provided to other CVT partners by courses, consulting, and installation support.

## 3.5 Literature

[18] S. Morpurgo, A. Hunger, M. Melgara, C. Segre: RTL Validation of VLSI: An Integrated Set of Tools for KARL; submitted for IFIP Int'l. Symp. on Computer Hardware Description Languages and their Applications, Tokyo, Japan, Aug. 1985

# EEC microelectronics research

On February 1st of this year the EEC gave its approval to a community research project for the development of advanced design techniques for large scale integrated circuits, specific to the telecommunications field.

The project, named CVT Project (CAD-VLSI for Telecommunications) should cost, within the next three years (February 1983–1986) 24 000 000 ECU, half of which will be provided by the EEC. Twenty-eight organizations made up of research institutes, universities and industrial concerns in Italy, France and Germany will participate in research.

The German research contribution, which amounts to about 8 000 000 ECU, is coordinated by the FI/DBP (Forschungsinstitut der Deutschen Bundespost beim FTZ) of Darmstadt.

The research will be carried out jointly by AEG-TELEFUNKEN, FHG (Fraunhofer Gesellschaft), FI/DBP, GMD (Gesellschaft für Mathematik und Datenverarbeitung), Standard Elektrik Lorenz AG, and the Universities of Aachen I, Aachen II, Bremen, Darmstadt, Dortmund I, Dortmund II, Karlsruhe and Kaiserslautern.

The French research contribution, which amounts to about 7 000 000 ECU, is coordinated by CNET (Centre National d'Etudes des Télécommunications). The research will be carried out jointly by CENG-LETI, CII Honeywell Bull, CIT-ALCATEL, CNET, IMAG, INRIA, THOMSON-EFCIS.

The Italian research contribution, which amounts to about 9 000 000 ECU, is coordinated by the Centro Studi e Laboratori Telecomunicazioni S.p.A. (CSELT) of the IRI-STET Group.

The research will be carried out jointly by CSELT, Italtel, Olivetti and SGS-Ates, the Polytechnics of Milan and Turin and the Universities of Bologna and Genova.

The main aim of the project is to give to European industry the capability to design very advanced and complex electronic circuitry, such as very large scale integrated circuits (VLSI), using the enormous potential of microelectronic technology (about 1 million transistors in a single component in the coming years). In particular, the research programme aims to develop automated design aids integrated into a system.

The research is therefore of strategic importance for the development and future of telecommunications inasmuch as VLSI circuits will be the basis of new telecommunications systems. This EEC initiative, which plays an important role in the collaboration of Common Market public and private research organizations in the European telecommunications industry, has interesting possibilities of development in other industrial sectors.