A Design Language for the "Long Thin Man": *Proposal of a Methodology for Problem Capture and Conceptual Design*

R. Hartenstein, K. Lemmert,
Kaiserslautern University

CVT report,
Kaiserslautern, Febr. 15, 1985,

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

Preface

The development of, and, user support for KARL also includes the development of "programming style" for KARL users. To teach the use of KARL not only includes the development of a set of good examples, but also to give an answer to the question, when and where to use KARL. Of course, the same questions may be raised fot KARENE users, since KARL is a subset of KARENE. This is an attempt to stimulate a discussion on this.

This is an extended abstract having been submitted for a conference. It describes a methodology proposed by us for the use of KARL and KARENE. However, this proposal has not yet been discussed with subtask 1.6 (KARENE, Grenoble), nor with the task leader of task 1.

Extended Abstract

This paper describes a CAD tool-box-based methodology for direct implementation of algoritms 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. Fig. 1 illustrates his differences from specialized designers found in design centers organized more conventionally. 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 for example in 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-II language in use for years [4, 5] to KARL-III [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 (see fig. 3) is relatively simple.

The first phase of the design process (see fig. 2) 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 out 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)

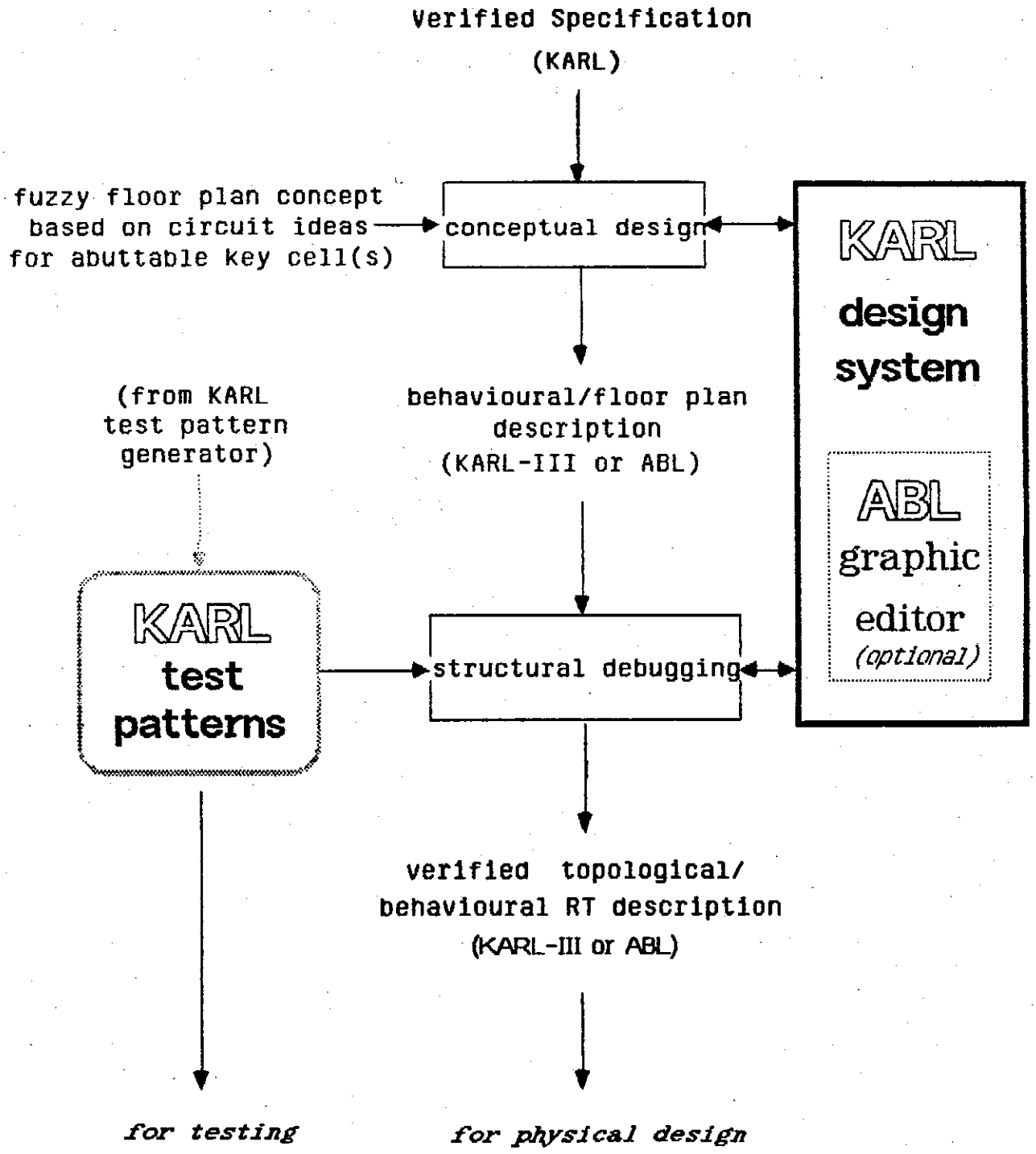

Fig. KARL use in the conceptual phase of structural VLSI design

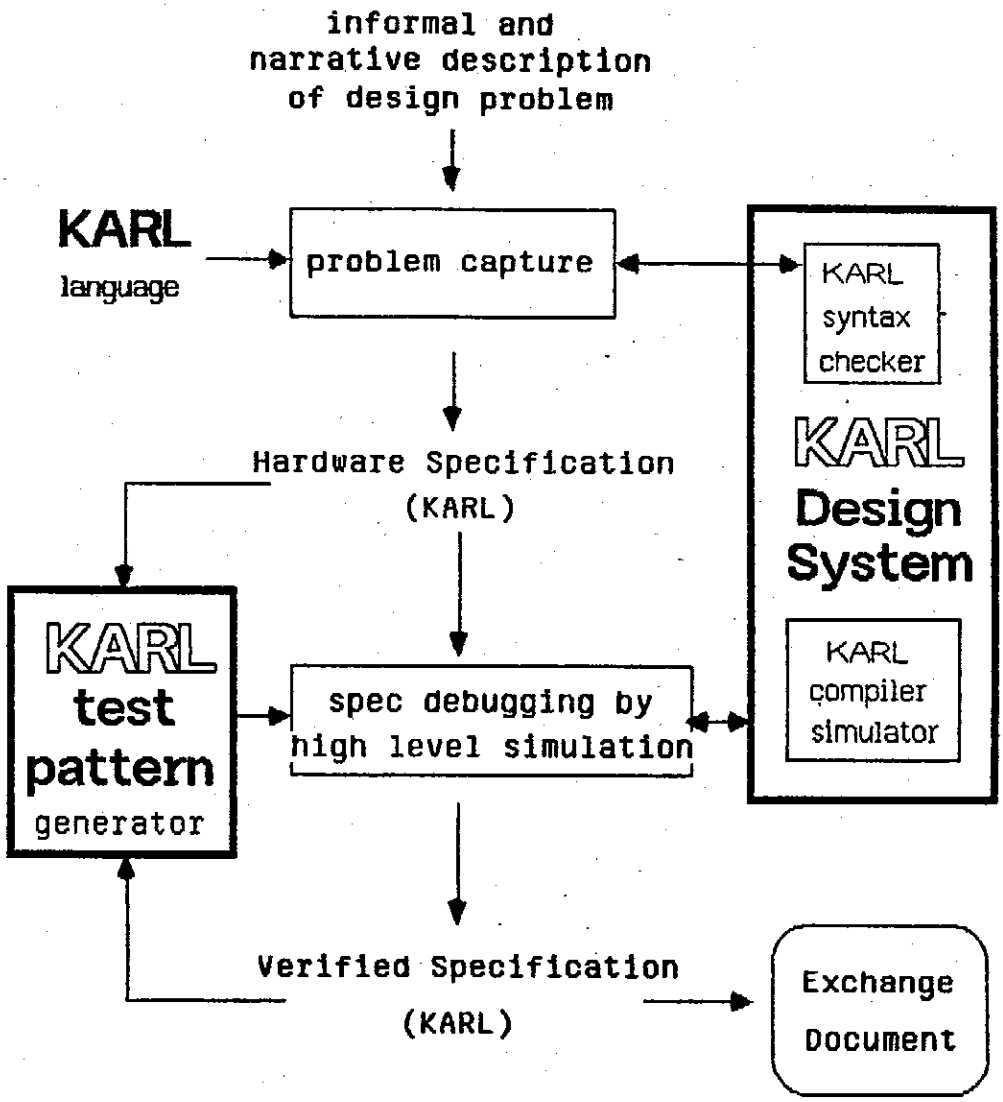

Fig. . KARL use in design problem capture and specification verification

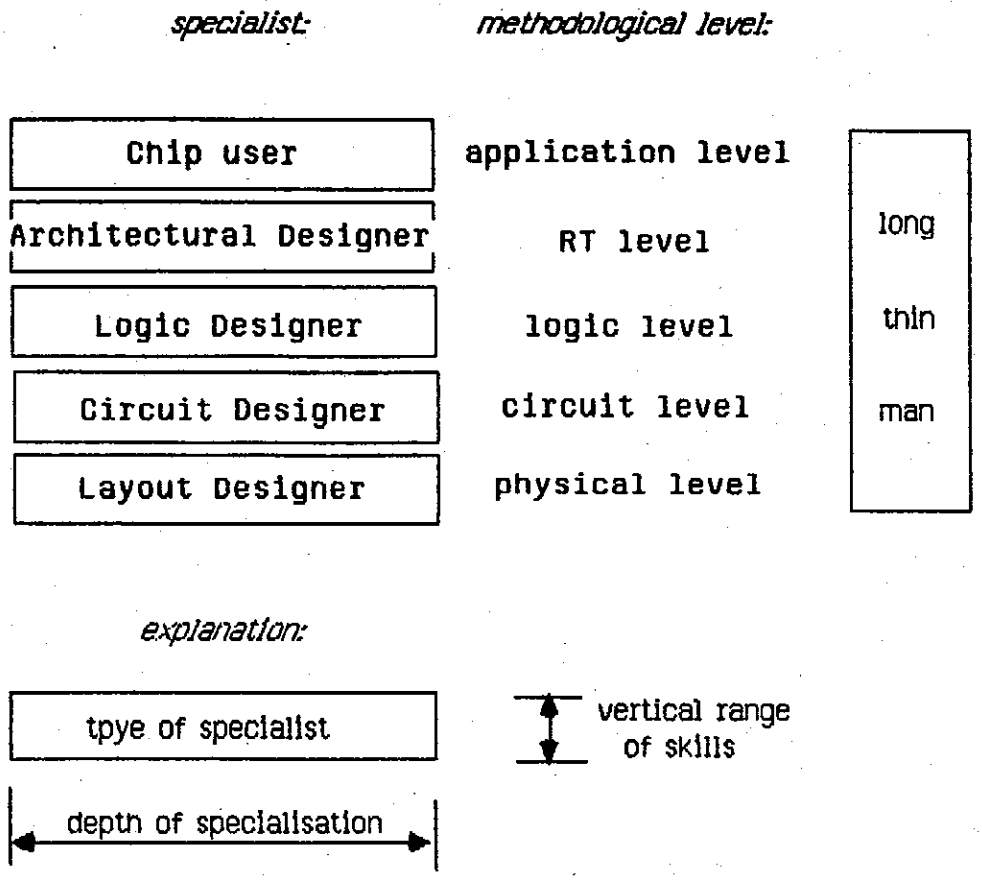

Fig. . Areas of responsibility in VLSI design: conventional (example) versus "structured design" by the "long thin man".