Design Reuse - Practical Problems and Solutions

Thomas Harriehausen

Siemens AG, Semiconductor Division
HL CAD SYS
P.O. Box 80 17 09, D-81617 Munich, Germany
Phone: +49 89 636-24762
Fax: +49 89 636-23650
Thomas.Harriehausen@HL.Siemens.DE

1st Workshop “Reuse Techniques for VLSI Design”
September 1, 1997
Karlsruhe, Germany

Design Reuse - Practical Problems and Solutions

Thomas Harriehausen
Siemens AG, Semiconductor Division
HL CAD SYS
P.O. Box 80 17 09, D-81617 Munich, Germany

Keywords

Design Reuse, ASCIA, ASIC, Hardware Description Language, Intellectual Property, Core Based Design, Macro Cell, Business Model, Security, Design Data Management, Standards

Abstract

Siemens Semiconductors is one of the world's biggest developers and producers of standard and application specific integrated circuits. Main areas of activity include memories, chip-cards, microcontrollers, automotive, telecommunication and consumer electronics ICs. Our unified “all-in-one” IC design system “ASCIA 4.1" supports most of the technologies, libraries and design methodologies used within the company worldwide.

It is almost generally accepted that design reuse may be a way to dramatically increase productivity and shorten design development time for certain - but not all - product groups. Before design reuse can become an integral part of design methodologies, a wide range of questions (technical, commercial, legal, social) has to be answered. Even of the technical issues, several are not addressed by the EDA tool suppliers. Some of them are so basic that many managers and researchers are not aware of their existence.

In this contribution, a short overview shall be given of the problems which designers and CAD support people face who want to push design reuse in a “real-life” environment. Considerations and solutions implemented in ASCIA 4.1 are presented where available.

1 Introduction

1.1 The Vision

Design reuse is currently being discussed as the only means to fill the dramatically growing gap between progress of deep sub micron semiconductor technology and electronic design automation (EDA) tool productivity. Around 8/96, the president and CEO of the EDA tool vendor Cadence Design Systems, Joe Costello, travelled the world spreading his vision of “**mix & match**” or “**plug & play**” of “**intellectual property (IP) blocks**” to build “**systems on a chip (SOC)**” which meet **price**, **time-to-market** and **performance** requirements.
The instrument to define the yet missing technical basis (data and interface standards) for this goal was initiated (and supplied with standardisation proposals) by him, too: the “**Virtual Socket Initiative Alliance (VSIA)**”. It remains to be seen whether VSIA will be able to reach goals that e.g. the “**CAD Framework Initiative (CFI)**” (now called “Silicon Integration Initiative”) and several other organisations missed to reach until today. The worst thing that could happen is that VSIA just declares a bunch of existing (preferably Cadence originating) formats like C, C++, VHDL, Verilog, SDF, LEF, GDSII to be the “new” standards for SOC designs.

1.2 The basic Problems

Some EDA tool vendors promise to deliver software which allows to build SOCs containing millions of logic gates. Many papers are published about reuse of VHDL design data. One might think: “Well, where is the problem, why doesn't everybody just do it?”

One problem is that most people talking/writing about design reuse are explicitly or implicitly only considering synthesis of **purely digital** - usually CMOS - **integrated circuits (ICs)** using a **hardware description language (HDL)**. But complete complex systems will scarcely ever be totally digital. Combining digital and **analog**, maybe even **radio frequency** (RF) functions on a chip is absolutely not straight-forward and scarcely supported by today's EDA tools.

Furthermore, IP blocks are not a new idea, just a new term for what we formerly called “**macro cells**”, “**cores**” or just “**building blocks**”. In all established IC developing organisations, lots of IP exists. Thus, the second problem is to make **“old” cores** which were not designed for reuse reusable in SOCs, at least within the company.

The third problem is that some facets of “design reuse” (e.g. commercial, legal, security, educational, social issues) can't be tackled with EDA software. People have to change their working style, their way to communicate or even their way of thinking and solving problems.

The fourth problem is that both “**design for reuse**” and “**design by reuse**” do not always make sense. Frequently, designers find that the effort to implement the requirements imposed by both approaches does not pay out. This is often the case for analog building blocks.

And last but not least: Although most of the existing EDA-related challenges are known for purely digital ASICs consisting of newly developed cores and glue logic, satisfactory commercial solutions do not yet exist for all of them. Some “success stories” are just marketing hype.

1.3 Analogies

The people who are developing methodologies for IC design reuse can learn from related domains where similar problems have to be or even have been solved.

The first domain is **software development**. Development, test and maintenance of complex software can only be managed by partitioning it. The price for partitioning software are data interfaces. An interesting approach is to build complex software packages from lean, specialised “**applets**” which conform to certain linking and data exchange standards. The analog terminology in chip design are “**chiplets**”. The problem is analog, too: design data exchange and signal interface standards must be defined and respected. In this domain, the EDA industry can probably learn from the software industry.

The second domain is **computer aided engineering** (CAE) of the electronic systems which are built from the components we are talking about, e.g. printed circuit boards (PCBs). Many of the problems which had to be solved for these systems will become increasingly important for us like easy IP block selection from catalogues, “pin compatibility”, second sources for cores, system simulation using “black box” models, “at speed” hybrid analog/digital manufacturing test, remote firmware maintenance/update, optimization of placement etc. I think that in this case, both domains can learn from each other.

1.4 Real Life

Like many other semiconductor companies, Siemens Semiconductors currently doesn't consider EDA software development as their core competency. Thus, we are primarily using software from highly specialised EDA tool vendors like Synopsys, Cadence, Mentor, Viewlogic etc. This is the reason why publication of brilliant theories and great algorithms for EDA tools or reuse workflows without a practical implementation doesn't help us at all.

On the other side, a bunch of commercial EDA tools and proprietary libraries doesn't make up a design system (although many people think so). None of our EDA tool suppliers can deliver an IC design system which works “out of the box”. The software we get on CDROM may be sufficient to execute some basic labs at a university's graduate IC design course. To create a leading-edge IC CAD system which permits SOC designs, software components from many sources have to be compared and integrated. The more versatile and comfortable this system shall be the more effort must be spent by the semiconductor company's CAD people or expensive consultants from the EDA vendors.
Most of the design libraries which EDA tool vendors deliver with their software are not suited for industrial use. Siemens Semiconductors’ central library department delivers a wide spectrum of libraries including several macro libraries to our user community to enhance productivity through reuse. But most of the cores which are potential candidates for **core based designs** are created by designers in our business units.

After a short overview of our unified IC design system ASCIA 4.1, several dimensions of the “design reuse” problem will be highlighted with the intention to start a controversial discussion.

2 ASCIA 4.1 Overview

The IC development system most widely used at Siemens Halbleiter (“HL”) is “**ASCIA 4.1**” (**A**dvanced **S**ilicon **C**omp**I**l**A**tion) [1]. This **unified, “all-in-one” IC design system** consists of a proprietary kernel, “best-in-class” design tools from all leading EDA tool vendors, technology data for all processes used by HL, and a variety of libraries.

Originally only intended for in-house use, today more than 1000 designers at almost 30 locations around the globe are using this system. This means that ASCIA 4.1 is one of the world's most widely spread productive IC development systems. The only way to promote this originally “monolithic” in-house system into an externally installable product was modularisation. Today, individual ASCIA 4.1 subsystems (“design kits”) can be installed at external locations.

ASCIA 4.1 supports fullcustom and cell based design methodologies, schematic, HDL and symbolic high level design entry. A wide range of device, standard cell and macro libraries are available for the ASCIA users. ASCIA offers a portfolio of more than 30 technologies like bipolar, CMOS (analog, digital), BiCMOS, smart power and several special purpose technologies.

ASCIA 4.1 was initially released in 12/94 as our “4th generation IC CAD system”. Since then, new system components are released monthly after having passed internal QA of the CAD and LIB departments. Meanwhile, ASCIA 4.1 has reached a high level of maturity. Recent areas of development are the migration of ASCIA from the SunOS 4.1.3 to the Solaris 2.5.1 operating system and the introduction of Cadence's data management tool “TDM” which is part of Cadence's DFII 4.4.X.

3 Problems and Solutions

Design reuse is a **multi-dimensional problem**. The following sections shall underline this thesis and thus also contain some aspects which are usually not considered in this context. Most items are highlighted from a system architect's point of view.

3.1 Design Representation Standards

EDA data standards are needed to pass the design data between the tools in a flow. But they are also necessary to efficiently exchange data between different companies which probably use different tools and data management systems.
Our industry lacks stable, fully supported standards. Most tools can only handle a subset of one version of a data representation standard ("view"). Thus, some information is usually lost when they are used as intermediate format.
Most EDA tools store their working data in a proprietary format and partly even in proprietary databases. Thus, we usually have a hierarchical data representation architecture: Between CAD systems or companies, data is exchanged using the "standard" formats. On top of this data bus, tool specific data representations allow effective processing of the data. Obvious setbacks are the need for data translation and the risk of inconsistent data which we face whenever data is duplicated. For timing information, a comprehensive standard from which all proprietary views can be derived is still missing.
To avoid the need for data conversion and the risk to loose data or work with inconsistent data, some people think it makes sense to focus on one EDA tool supplier. But even tools from the same supplier do not necessarily understand the same data "standard" (e.g. SDF).

External design contractors which are designing for Siemens Semiconductors do get an ASCIA subsystem adapted to their design's needs to avoid data exchange problems.

3.2 Entities of Reuse

Design reuse doesn't start with complex cores. Today, even a "full custom" IC will normally not be designed from scratch. Device libraries with parameterised layout cells ("pcells") can enhance productivity dramatically and contain valuable IP. Technology specific standard cell libraries are needed for synthesis of digital ICs. The limit between a complex standard cell and a macro/mega cell (i.e. "core") is not clearly defined.
Analog cores (like PLLs, A/D and D/A converters) usually contain less components than digital cores (like microcontrollers, DSPs). It is much more difficult to create a reusable analog macro than a digital one because analog macros almost always have to be parameterisable. A core based deign thus consists of one or more cores plus "glue" parts. The following terminology is often used to distinguish between different classes of cores:

- **Hard core**: Not customisable, usually optimised, technology dependent, layout and simulation models but no netlist available, good IP protection possible
- **Firm core**: Partially customisable to application specific needs, technology dependent
- **Soft core**: Highly parameterisable, technology and application independent

3.3 Design Reuse Levels

The scope of design representation levels on which design data is reused ranges from system level description via schematics and netlists down to physical layout.

For reuse on physical level, ASCIA 4.1 contains the proprietary tool “Mask Layout Reduction (MLR)” to map layout data between related technologies. Design rules are automatically obeyed and required chip area can be minimised through compaction.

3.4 Integration of non-digital Blocks

Integration of digital cores which were developed in the same HDL within the same environment is the easiest reuse scenario. Things get really challenging if analog blocks or full-custom digital blocks shall be integrated. In this case, the non-HDL blocks must be encapsulated to allow processing in the synthesis oriented design flow. Other problems have to be solved if old designs shall be reused of which only a netlist or the layout exists

3.5 Reproducibility of Design Environment

Yes, this is also a real-life reuse issue. If a design project shall be continued after a pause of several months or if a redesign of a chip which was designed several years ago is necessary, designers often face the problem that they can't reuse their own data. The reasons can be manifold: one of the tools formerly used is no longer in the market or the license has expired or a new version of the tool has been introduced which is not data compatible to the old version. The same may happen with computing infrastructure components like operating system, graphical user interface or file system used. Also, the contents of the libraries used (referenced) in the design may have changed. Even if only bugs were removed or enhancements implemented, old designs may no longer work.
In the CAD department, we decided not to keep at least one sample of all hardware, operating systems and EDA software tools in an archive because in the end, it would probably not help. After 5 years, almost nobody will be able to get these old components up and running again. Our tool vendors are usually not willing or able to generate license files for ancient software versions.

Within ASCIA 4.1, we try to provide a stable design environment at least for the duration of a design project. Our means to accomplish this are **modularisation** of the CAD system and **versioning** of these components. ASCIA consists of versioned “packages”. Packages are technologies, libraries, tool bundles (like a release of the Synopsys IC design software) and also the ASCIA kernel. Each design may define which version of which packages it wants to use. The problem for the CAD department is that the number of possible combinations of tool and library versions used can not be supported and we have to focus on certain “mainstream flows” consisting of predefined sets of component versions.

3.6 Modularity, Hierarchy, Interfaces

Design of SOCs is only possible using a “divide-and-conquer” strategy, i.e. partitioning the system into blocks, definition of the interfaces, concurrent design of the blocks or selection of suited cores (“make or buy”) and finally, system integration. If a block is too complex, the methodology has to be applied recursively. A key item are the interfaces, e.g. on-chip buses which must be well-defined but as general and versatile as possible (both with respect to function and timing) while providing the performance required.

Both aspects of reuse, “design for reuse” and “design by reuse”, can only be successfully applied if the blocks handled are of appropriate complexity. The more complex a core is, the more costly its creation will be. But its value may decrease because it becomes inflexible and creates additional restraints (e.g. w.r.t. place&route, timing) for the whole design.

In ASCIA 4.1, the design environment for a chip has a modular, hierarchical structure. Each design has its own UNIX account which contains all the design's data. Each project may consist of an arbitrary number of subprojects. Complex designs should be partitioned into several subprojects which can be designed at different sites and can use different design methodologies. Blocks which are reusable are stored in business unit specific, “internal reference libraries”. Design project accounts, internal reference libraries and all libraries supplied by the central library department have a **unified structure**. Through this ingenious feature, ASCIA 4.1 users can easily build up their design from different sources and reuse of design data among the worldwide ASCIA user community is facilitated.

3.7 Versions, Variants, Configurations of Design Data

Design for reuse only pays out if the IP blocks are used often and for a long time period. This implies that they have to be maintained, e.g. to remove bugs, increase flexibility and robustness, add new features or adapt them to new technologies or EDA tools. For this maintenance process, versions (sequential in time, reflecting design history) and variants (parallel in time, allowing for design alternatives) must be handled and it must be possible to answer the question “who did receive what data when?” Especially in case of generated macros like memories and pad cells where the end user operates (potentially error prone) generator software, this may be a real problem.
As soon as versions and variants must be handled, configuration management is necessary which allows to defines which components belong/fit together.

In ASCIA 4.1, both the CAD system (including all centrally developed reference libraries) and the design environments are versioned. A special version is the “current” version which points to the most recently released version. Of each ASCIA subproject, an arbitrary number of versions/variants may exist. In the subproject version containing the “top” cell of the design, an ASCIA specific, central configuration files exists which defines the subproject versions and ASCIA reference library versions which shall be used to create the design hierarchy below the top cell. Using “current” versions, **dynamic binding** is possible. Using a fixed version name, **static binding** can be implemented. All design tools in ASCIA extract the information about the location of design data to use from this configuration file.

Implementation of these basic design data management (DDM) tasks on UNIX level has the advantage of minimal administrative and performance overhead. The setback is that advanced DDM functions like multi-user access control and delta storage are not available. VHDL designers usually use emacs plus rcs or sccs to manage their source files.

3.8 Verification, Parasitics, Timing, Testing

Verification of the behaviour of integrated circuits is usually done using simulation. For cores, models for functional and timing simulation must be available. For a hard core which is delivered without netlist, a “black box” model must be provided to protect the IP contained in it while allowing verification of the complete system. For digital designs, more efficient alternatives are available in the form of formal verification and emulation. The views for both purposes must contain a complete description of the core's logic and thus cause an IP protection problem.

Physical verification runs (like DRC, ERC, LVS) should not be necessary for automatically created layouts. If however, manual changes were applied to the physical design or if any tool in the flow is unreliable, this last, very time consuming step becomes necessary, again.

Parasitic effects like crosstalking can only be calculated exactly when the physical design of the chip is known. Hard cores which work in many chips may fail in others due to parasitics caused by superimposed “external” circuitry. For big cores, it is unrealistic to prohibit any over-the-block routing. Routing channels for this purpose in hard cores make these even bigger.

Correct function of a soft or firm core in a customer's design depends on the quality and performance of the target technology and the cells offered by the library used by the synthesis tools. Thus, timing analysis of the synthesis results (e.g. using static timing analysis) is always necessary.

If a SOC is built using cores, the design for test (DFT) strategies implemented in the cores must fit together. This can be a problem with hard cores or if insertion of test functionality pushes a soft core's parameters off its specification.

3.9 Documentation, Design History

Documentation is a key issue for reusable components. But it seems to be a serious threat to many engineers. High time pressure is frequently used as an excuse when important know-how about the design stays in the heads of designers and therefore is lost when this expert changes the team. This problem requires a general change in working methodology and priorities.
A market for cores requires easy access to comprehensive data to select suitable cores for design-in and to identify application areas for which development of new cores makes sense. The WWW is probably the service to provide and search the information about available/required IP blocks. Siemens has built up one of the most sophisticated intranet information infrastructures in semiconductor industry. We provide all information about our CAD system via intranet WWW.
Our commercial EDA tools do not have a built-in facility which simplifies or forces documentation of the design process. For HDL based designs, information about the design history (“who did what when why?”) can be maintained with simple tools like sccs or rcs.
Most tools which handle graphical information (e.g. schematic entry tools) don't provide such functionality and don't know the meaning of a user's actions.
In ASCIA, we have implemented a utility for the Cadence DFII which asks for documentation input whenever a designer saves his work or checks it in.

3.10 Quality

A design block which shall be reusable must fulfil higher quality requirements than “normal” design data. These requirements include

- completeness and consistency of views
- precision of models w.r.t. topology and parameters
- robustness w.r.t. manufacturing process, timing budgets and parasitic effects caused by circuitry external to the block
- testability w.r.t. test coverage and support for certain test methodologies, e.g. IDDQ or ATPG
- possibility to verify the quality of a (new) block (version) automatically using e.g. regression testing

This is just a matter of effort in case a block is designed for reuse right from scratch. If blocks from an existing design shall be made re-usable later, things become more difficult.

3.11 Backup, Archiving

Another reason why a designer might not be able to re-use yesterday's results is that data is accidentally deleted or that due to a system malfunction, data is lost. In the UNIX world, the only help is to restore data from a - hopefully existing - backup. Unfortunately, a UNIX-level backup of databases which are currently being changed may cause problems.

For QA (ISO 9000), disaster recovery and product warranty reasons, we have to archive all design data which resulted in sold chips. The medium used must keep the data physically readable for at least 10 years. If this time frame is considered, it does not make any sense to store data in proprietary formats, e.g. Cadence or Synopsys or Avanti databases. The only formats which make sense are the few "standard" formats we have like EDIF, GDSII, SPICE, VHDL, Verilog and some other ASCII coded data.

3.12 Distribution of IP

IP blocks may be reused by the same design group, by different design groups within the same company or by design groups from other companies. Thus, processes to exchange design data between projects, sites and companies are required.
As a global player in the semiconductor market, Siemens HL has microelectronics development centres in Europe, Asia and the USA. Multi-site projects are usual since several years. Because all development centres use the unified ASCIA 4.1 environment, exchange of designs is very easy.

3.13 Commercial and Legal Issues

Exchange of IP requires a suitable business model. Fundamental questions like "Who has to pay whom how much for which kind of reuse of what?" have to be answered.
Probably, the semiconductor industry will split up into two distinct groups: IP providers and IP users. Today, Siemens Semiconductors plays several roles: ASIC design house, standard semiconductor producer, silicon foundry (to a limited extent), IP developer, IP provider (to a limited extent), IP user. We will have to decide whether we want to act as an IP vendor in the future.
Legal issues related to ownership and responsibility have to be clarified: "Who is responsible for quality, characterisation, documentation, synthesisability, testability, maintenance, defect analysis?" and "Who is responsible if a re-used component fails?" If we try to protect IP from becoming "public domain", we have to be able to answer questions like "How can a company prove that it is the originator of an IP block?" How can a core vendor track the trail of its IP, i.e. answer the question "Who did use which core in which design when?" "Vagabonding" macros of unclear origin which differ from their source in an unknown way are a threat both for the potential user and the original supplier.

Today, we have almost no answers to these questions in place. Hopefully, we will be able to adopt some solutions from the software domain.

3.14 Intellectual Property Protection

In the classical semicustom ASIC development model, the customer received only models of the library provider's IP. When IP blocks shall be sold to be fabricated elsewhere, it becomes difficult to retain IP.
In the past, reverse engineering of chip designs based on actual silicon was very expensive.

As soon as the design data is available in any machine readable format, life becomes much easier for IP pirates. No matter whether encrypted HDL code, endless flat netlists, gigantic layout data or a model in executable format is passed: it is possible to derive information from it which was not intended to be unveiled.
This problem has been recognised as one of the most important ones with respect to a global market for IP. One solution approach is to distribute “compiled” data instead of source data. But there is no real solution in sight. And I doubt that there will be one because of the analogy to the software domain where powerful reverse engineering tools exist. One approach I can think of is “security by obscurity” i.e. to make a design so complicated that it becomes difficult to understand it. Which applies to users and maintainers ...

3.15 Data Security

Today, most companies use UNIX as the main computing platform for IC design. UNIX is and will always be an inherently insecure operating system. Whoever wants to get superuser privileges (“root”) will be able to manage that through one of many security holes in the operating system and application software. A superuser can read and modify all data on all workstations s/he can access.

The internet is a very serious threat to system security. Using security holes in internet tools like news readers or WWW browsers, people from outside a company can gain illegal access to important IP. Therefore, design environments for critical products may have no direct or indirect connection to the inter- or intranet.
The internet and its protocols are often proposed as the ideal medium for IP exchange. Although this is probably right, two things have to be regarded. Firstly, HTTP is not suited for transfer of large amounts of data. Secondly, all data transferred in the Internet is systematically being snooped by diverse actors.

The only secure mode to store and transmit design data is in an encrypted form. Use of “strong” encryption methods must be permitted for this.

4 Outlook

Design entry is steadily moving to higher levels of abstraction. Graphical **high-level** (e.g. flow-chart and state-machine) **design entry tools** may be the next generation after today's HDLs. They can facilitate documentation and give a better overview of complex systems. The variety of design methodologies will increase. For those designers which hate GUIs, C, C++, Java or similar textual representations will be available.

Core based designs will normally contain programmable components. **Hardware/software co-design** will be the next challenge to tackle. Software development environments and our known hardware design systems will have to grow together. The overall development environment will become even more complex. We need good system architects and well educated/trained developers to work in such an environment.

Where will the “systems on chip” of the future be designed? Maybe, in 10 years from now, many EDA tools will be developed and used in **Asia**. Millions of well-educated hardware and software engineers will be available in countries like India, Korea and China soon. Then, “teleworking” will get a completely new meaning for Europe’s expensive engineers.

As a consequence of easy information exchange (e.g. WWW) and growing worldwide availability of engineering manpower, price for intellectual property of any kind will decrease. It can't be ruled out that some core providers will use “dumping” prices to push their products into the market hoping to establish built-in proprietary interfaces as “de facto” standards.

Let’s see what CFI, VSIA, ECSI and the like will have achieved till then.

5 Conclusion

Design reuse is a very complex challenge. Missing or insufficient standards for design data representation and exchange reduce productivity in the whole semiconductor industry.

In case a worldwide market for design data will emerge, the products handled will probably be complex entities of design data, the so called “cores”. Processes to create, model, test, document, maintain, offer, protect, search, transmit, verify, pay, integrate such cores have partly to be defined, yet.

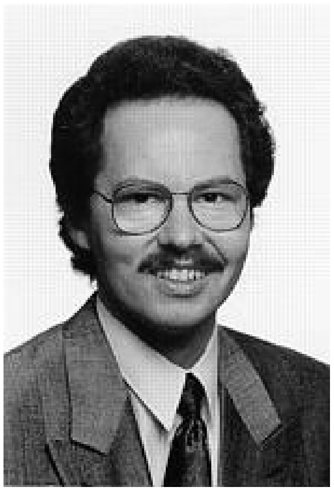
This overview had the intention to highlight that there is still a long way to go until we can build systems on a chip in a “plug & play” manner using analog and digital cores. In their marketing papers, our EDA tool suppliers usually describe what they want to be able to do. Not what they currently can provide. And which EDA tool vendor offers satisfying solutions for documentation, design data management, quality assurance, backup and encryption?

6 References

[1] Harriehausen, T.: “ASCIA - A Multi-Vendor IC Design System”
Proc. Int. Cadence User Group Conf., Sept. 8-12, 1996, Phoenix, AZ, pp. 207 - 215

About the Author

Born in 1958 in the north of Germany, Thomas Harriehausen finished his studies of Electrical Engineering and Computer Science at the University of Hannover with the diploma degree in 1987.



His primary interests were in microelectronics, microprocessors, telecommunication, operating systems and software engineering.

His diploma work was about new algorithms for automatic layout generation for bipolar ICs.
For his Ph.D. thesis, he developed a system for adaptive, real-time simulation of thermal networks.

Since 1992, he is with Siemens Semiconductors (“HL”) in Munich where he started as a support engineer in the central IC CAD department. He is one of the architects of HL's unified “all-in-one” IC design system ASCIA 4.1.

He became leader of the “CAD System Integration” group in 1995.
Since 1996, he is responsible for CAD “Systems and Training”.