

Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 74 (1983)

Heft: 5

Artikel: Interdisciplinary enhancement of VLSI design tools towards an integrated CAS system

Autor: Joerg, W.

DOI: <https://doi.org/10.5169/seals-904775>

Nutzungsbedingungen

Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften auf E-Periodica. Sie besitzt keine Urheberrechte an den Zeitschriften und ist nicht verantwortlich für deren Inhalte. Die Rechte liegen in der Regel bei den Herausgebern beziehungsweise den externen Rechteinhabern. Das Veröffentlichen von Bildern in Print- und Online-Publikationen sowie auf Social Media-Kanälen oder Webseiten ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. [Mehr erfahren](#)

Conditions d'utilisation

L'ETH Library est le fournisseur des revues numérisées. Elle ne détient aucun droit d'auteur sur les revues et n'est pas responsable de leur contenu. En règle générale, les droits sont détenus par les éditeurs ou les détenteurs de droits externes. La reproduction d'images dans des publications imprimées ou en ligne ainsi que sur des canaux de médias sociaux ou des sites web n'est autorisée qu'avec l'accord préalable des détenteurs des droits. [En savoir plus](#)

Terms of use

The ETH Library is the provider of the digitised journals. It does not own any copyrights to the journals and is not responsible for their content. The rights usually lie with the publishers or the external rights holders. Publishing images in print and online publications, as well as on social media channels or websites, is only permitted with the prior consent of the rights holders. [Find out more](#)

Download PDF: 14.08.2025

ETH-Bibliothek Zürich, E-Periodica, <https://www.e-periodica.ch>

Interdisciplinary enhancement of VLSI design tools towards an integrated CAD system

W. Joerg

The transformation process of traditional instruments or apparatus towards VLSI based products is outlined in the context of instrumentation industries. A brief analysis of the CAD market compared to current needs shows the necessity for interdisciplinary efforts to enhance the design tools. Some basic requirements for an integrated VLSI design system are explained and steps towards full integration are sketched.

Der Wandel von traditionellen Instrumenten und Apparaten zu VLSI-Produkten wird am Beispiel der Apparateindustrie dargestellt. Eine kurze Analyse des CAD-Marktes, verglichen mit den aktuellen Bedürfnissen, zeigt die Notwendigkeit interdisziplinärer Bemühungen, um die Hilfsmittel für den Entwurf zu fördern. Es werden einige grundsätzliche Anforderungen an integrierte VLSI-Entwurfssysteme erläutert und verschiedene Schritte in Richtung voller Integration skizziert.

On présente l'évolution des instruments et appareils traditionnels vers des produits VLSI à l'exemple de l'industrie de l'appareillage. Une analyse succincte du marché CAD comparé aux besoins actuels démontre la nécessité d'efforts interdisciplinaires pour faire progresser les outils de conception. Quelques-unes des exigences fondamentales des systèmes de conception intégrés pour VLSI sont expliquées et différentes étapes en direction de l'intégration complète sont esquissées.

This paper has been presented on the Fall 1982 Meeting on Computer Aided Design of the IEEE Swiss Section, Chapter on Solid State Devices and Circuits, on 19th October 1982 at Bern.

Author's address

W. Joerg, LGZ Landis & Gyr Zug Corp., Zentrallabor 3967, 6301 Zug.

1. Introduction

The current instrumentation market is undergoing significant changes: the manufacturers have to cope with demands for higher integration of more and more functions, higher reliability with increasing production costs and tougher competition conditions. A very promising solution to this optimisation problem lies in the use of custom ICs.

For several reasons the already widely accepted approach of connecting standard ICs on printed circuit boards proves to be insufficient for many applications, and the approach of using microprocessors may not always be adequate.

Some instrumentation companies have therefore started a product transformation process (fig. 1), which integrates functions of existing products and new requirements on dedicated VLSI chips or chip sets.

2. A need for design tools

Most of these companies cannot afford their own VLSI production line (which, in fact, could be hard to justify commercially since current trends rather indicate a saturation of the worldwide chip production capacity). So they prefer the "silicon foundry" approach which means that they have to struggle through the design steps of their chips, from the product definition down to the geometric layout. The "pattern generation" tape which is generated from the geometric layout, acts as interface between the designers and the chip manufacturer. At that point they partly lose control of the fabrication process till they get prototypes, and again some time later, after debugging the assembled production chips to be incorporated in the final product.

The facts that more and more products will become involved in this transformation process, that the complexity of design and verification is dramatically increased, and that several expensive steps of the overall production process can hardly be influenced, stress the need for efficient and reliable high quality design tools.

3. Current situation of design tools

A look at the current CAD market, shows an emphasis on particular tools ("application packages") aimed at solving specific sub-problems like schematic entry, logic simulation, circuit simulation, placement and routing, layout editing, design rule check, etc. Unfortunately incompatibility appears to be the only feature common to such packages.

The CAD activities of universities are mainly focused on the development of newer and better application packages; only little effort is put on integration problems.

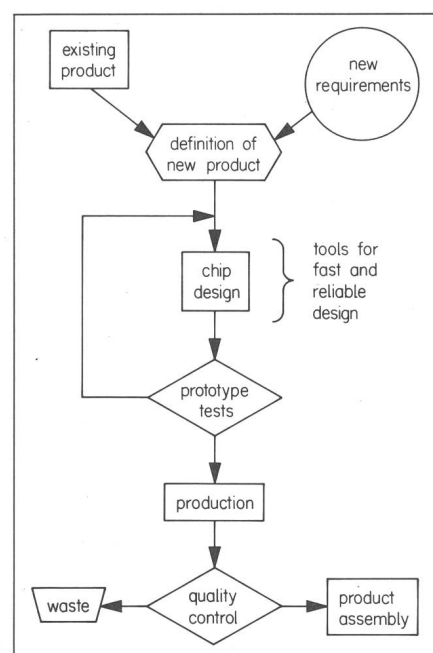


Fig. 1 Product transformation process

Such a situation is not acceptable to small and medium scaled industries entering the application field of microelectronics. Joint efforts of industries and universities have to focus on an integrated design system which puts together design tools in a safe and transparent way.

Distinction has to be made between "design with CAD-tools" and "design of CAD-tools". The latter will be of major interest in the following discussion.

The flowchart illustrates the development and application of VLSI design methodology. It begins with two sources of experience: "experience from software design methodologies" and "experience from VLSI design". These feed into the "VLSI design methodology" box. This methodology then informs the "software design technique" and the "VLSI design technique". The "software design technique" and "VLSI design technique" both contribute to the "frame for VLSI design system". This frame interacts bidirectionally with the "VLSI design tools". The "VLSI design tools" are supported by a stack of foundational disciplines: "systems theory", "mathematics", "electronics", "chemistry", and "physics". The "VLSI design tools" are used to develop the "integrated VLSI design system". Finally, the "integrated VLSI design system" leads to the "use of design tools", which in turn feeds back into the "VLSI design methodology" to refine the process.

ceptual base of an operational frame and the *improvement* of individual application packages with respect to integration requirements within that frame. Figure 2 shows some interdisciplinary connections in the design process of an integrated design system.

In VLSI environment a module would be described by e.g.: management parameters (name, creation date, updates, version, etc.), physical parameters (location, orientation, dimensions, etc.), interface parameters (type and location of interconnects, use of external entities, definition of internal entities to the outside, etc.), overall behaviour (required by hierarchical simulation) and the internal structure in terms of previously defined entities.

a module is described by:

- management parameters
- physical parameters
- interface parameters
- overall behaviour
- internal structure

The diagram shows a central module box with arrows pointing to five different representations:

- hierarchies**: A tree structure starting from a single box at the top, branching into three boxes, which then further branches into more boxes.
- abstract data types**: A box with multiple small squares on its top edge and a single line on its right edge.
- instances**: A stack of three overlapping rectangles.
- libraries**: A row of four boxes, with the middle two boxes replaced by an ellipsis (...).

The diagram illustrates the relationship between system and user libraries in a dynamic linking environment. It is divided into three main sections:

- Top Section:** Contains two boxes labeled "system library" and "user library". The "system library" box contains four icons: a small solid rectangle, a medium solid rectangle, a small solid rectangle, and a dashed rectangle. The "user library" box contains one dashed rectangle. Arrows from both boxes point down to the application section.
- Middle Section:** Labeled "application library" on the right. It contains five icons: a solid rectangle, a dashed rectangle, a dashed rectangle, a solid rectangle, and a solid rectangle.
- Bottom Section:** Labeled "application" at the bottom. It shows a complex internal structure with multiple nested boxes and a long horizontal bar at the bottom. A line connects the "application library" section to this internal structure, indicating the loading of the application's dependencies.

Parametrised modules (“module type” in PORTAL, or more restrictive: “class” in SIMULA) are used to describe parametrised solutions of classes of similar problems (e.g.: shift register with n elements, circular buffer with m elements of type t). Tested modules of “general” interest are collected in libraries on the appropriate level of generality (application level, user level, project level, etc.).

Achieving higher reliability on the one hand demands applying modern software design techniques and tools, incorporating accurate functional packages (e.g.: "interval arithmetic" which allows accurate numerical error estimation of simulations) and upgrading university built packages to industrial quality; on the other hand it requires appropriate user interfaces which guide the designer and restrict his freedom to what he really needs to do and to know, prevent many reliability and consistency problems; but

the most important contribution will be given by internal data manipulation mechanisms which guarantee data integrity and check their consistency at modification time.

Productivity increase can be achieved by reducing the number of feedback loops in the design process due to design errors. There are of course feedback loops in VLSI design which cannot really be eliminated: one typical example would be feeding interconnect capacitance, extracted from geometric layout, back to the timing simulation.

The basic idea here may be summarised by the slogan "correctness by design": once a program or design has proven correct on a particular abstraction level, it may become less efficient on lower abstraction levels and this may imply manipulations for optimisation, but the system should guarantee that the behaviour on the higher abstraction level is not changed. This requires reliable transition algorithms, guidance of the operator by the system and restriction of possible manipulation.

A substantial contribution to productivity increase can be given by increasing throughput: adapting application packages to the design methodology (e.g.: hierarchical simulation reusing results from earlier operations) and using new software technologies (e.g.: simulation with parallel processes).

Further productivity increases are implied by the design methodology itself: for instance reuse of already solved problems. But other requirements like reliability and data consistency cannot be implemented without serious effects on efficiency. The choice of how far all of the above mentioned requirements should be fulfilled is a very challenging optimising problem.

5. The VLSI design process

The VLSI design process may be viewed as a sequence of steps back and forth through several abstraction levels: defining objects on lower levels, characterising their behaviour and using them as building blocks or primitives on higher levels ("bottom up design") or subdividing objects on higher levels into sub-objects which have to be described by means of lower level objects ("top down design").

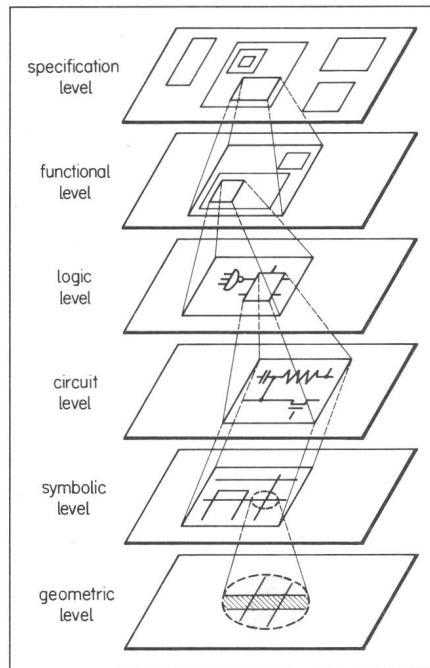


Fig. 5 Abstraction levels of VLSI design process

The "top down" versus "bottom up" war is rather an academic one, since both methods may be necessary for practical industrial designs: for instance device specialists have to construct efficient building blocks which require a high technology knowledge and make them accessible to circuit designers as "black boxes". On the other hand project leaders want to subdivide customers' specifications into sub-problems which they want to resolve by a functional description, which itself will be characterised by logic components, etc.

Figure 5 represents a possible classification of abstraction levels and shows an example of design structure. The aims of such a subdivision are an easier management of complexity by means of appropriate abstractions and the achievement of the best possible technology independence.

To make things clear: whatever the abstraction level is, the designers always track the same object; only the way of looking at it and representing it varies with the level. Therefore all application structuring capabilities (fig. 3 and 4), manipulation, debugging and simulation facilities asked for in the previous discussion, should be conceptually available at any abstraction level.

Computer scientists may find here an analogy to multi-pass compilation of high level languages: starting at the language level (functional level), a

program is translated down to an intermediate language (inherent program structure), from there down to an abstract machine level (e.g.: "stack machine") and finally down to the target machine.

The specification level which is aimed at formal program specification is still controversial (and this is particularly true for the VLSI design area). A major difference between language compilation and VLSI design is that the latter requires the possibility for designer interaction at virtually every abstraction level. For some less critical applications (e.g.: gate arrays) this need for interaction may be negligible, therefore programs for automatic translation from e.g. logic level down to geometric level are emerging. Such programs are given the rather mystified name of "silicon compilers".

6. Summary of requirements for an integrated CAD system

Supply the users with a system that

- guides them through "all" design steps;
- asks for their interaction only where design related decisions and actions are required;
- gives them access to the design at any abstraction level by means of homogeneous and consistent I/O tools;
- guarantees consistency of modified data throughout all abstraction levels (or at least records changes and directs users);
- supports principle of modular, hierarchical design and concept of libraries on every level;
- provides capabilities for design exchanges with other users at any level;
- makes the design as technology-independent as possible.

7. Steps towards an integrated CAD-system

The analysis of the current situation in the CAD field, in comparison with what is needed particularly by instrumentation industries shows that, first of all, experienced VLSI designers and software designers have to work out a VLSI design methodology.

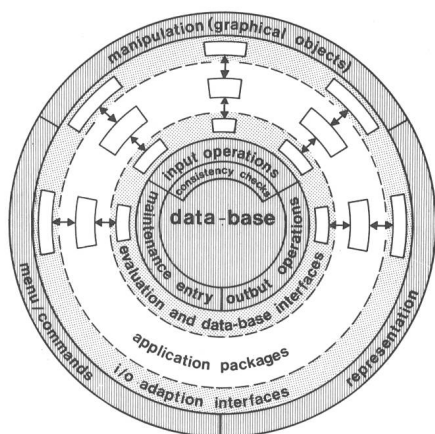


Fig. 6 Integrated CAD-system: integration concept

The next activities have to concentrate on definition and implementation of an operational frame (fig. 6) for successive integration of existing and most required application packages. These activities are split into two major topics:

- design and implementation of a human interface that fulfills the above mentioned requirements, particularly operation consistency and operator guidance;
- design and implementation of a fast and reliable design data base with its associated manipulation mechanisms.

Both activities have to be completed by conceptual work for interfacing application packages: I/O adaption interfaces and data base interfaces.

In a next step useful and accessible application packages are successively integrated in the operational frame. This step in fact is very critical because most existing packages will not easily fit into a new VLSI design methodology and cause inefficiencies within the new environment.

Finally new design tools have to be developed: they should be conceived for the new design methodology; they should be built with appropriate software techniques, be devised for the in-

tegration concept; hence they should fit better into the operational frame and provide a substantial increase in efficiency.

8. Conclusion

It is not the aim of this discussion to make things look easier than they are. We are aware that we have just pinpointed (and also omitted) some very difficult problems. Many of them are still pure research topics and no one can really make commitments with respect to their issues. We have also omitted any hardware considerations for possible implementation.

We just wanted to sketch some basic requirements, to show the need for interdisciplinary joint efforts and to outline a way to build an industrial tool which really deserves the qualification "integrated VLSI design system".