Interdisciplinary enhancement of VLSI design tools towards an integrated CAS system

Autor(en): **Joerg, W.**

Objekttyp: **Article**

Zeitschrift: **Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association Suisse des Electriciens, de l'Association des Entreprises électriques suisses**

Band (Jahr): **74 (1983)**

Heft 5

PDF erstellt am: **28.05.2024**

Persistenter Link: https://doi.org/10.5169/seals-904775

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.

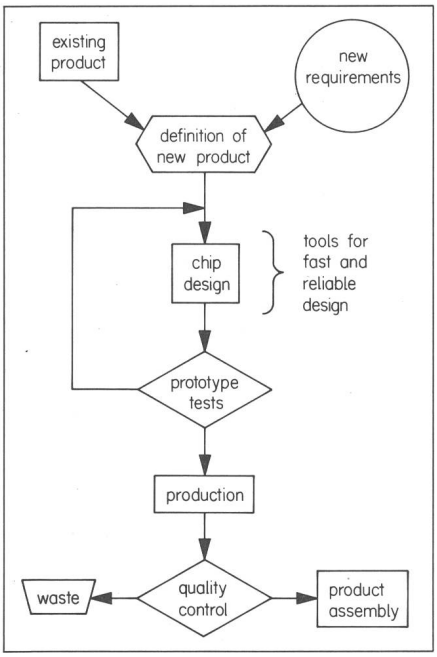

Fig. 1 Product transformation process

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.

Some (more or less lucky) attempts at connecting several application packages are being carried out by semiconductor industries and software houses. But none of the commercially available systems deserve the denomination "integrated CAD system" and so the user is still faced with major problems like lack of data exchange possibilities, lack of checks for data consistency, need for "manual" data conversion from one package to another, inconsistent (and clumsy) user interface, spreading of conceptually related data over large amounts of files (paraphrased as "data bases"), questionable software quality, etc.

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.

4. Interdisciplinary efforts to enhance design systems

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 motivations for design system enhancement are found in the needs for management of higher *complexity*, higher *reliability* and increased *productivity*. This involves both the definition of a VLSI *design methodology* as a conceptual 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.

Managing higher complexity suggests the use of structuring tools which emphasise partitioning into subproblems, classifying into similar problems, construction of hierarchies and reuse of already solved problems. Computer science in this field supplies good experience with a basic structuring concept called "module" which should find its counterpart in VLSI design methodology. A module is an abstract entity which allows structuring of problems, hiding their solution ("semantics") within fences ("black boxes") and describing the associated interface (fig. 3).

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.

When a module is embedded in a larger environment or hierarchy, the interface is expected to be checked ("context-sensitive syntax") by processing programs ("compile-time checks", "run-time checks").

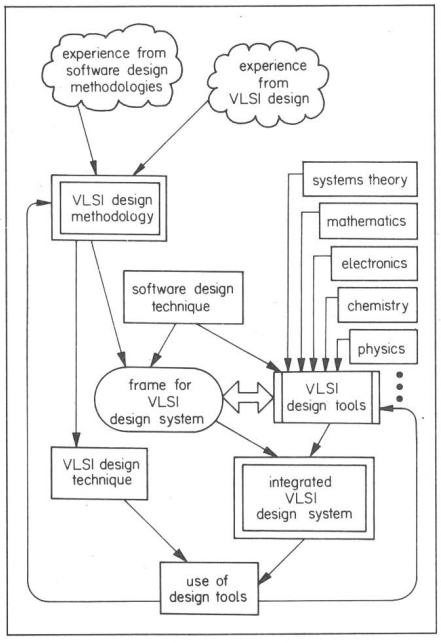

Fig. 2 Design process of a design system

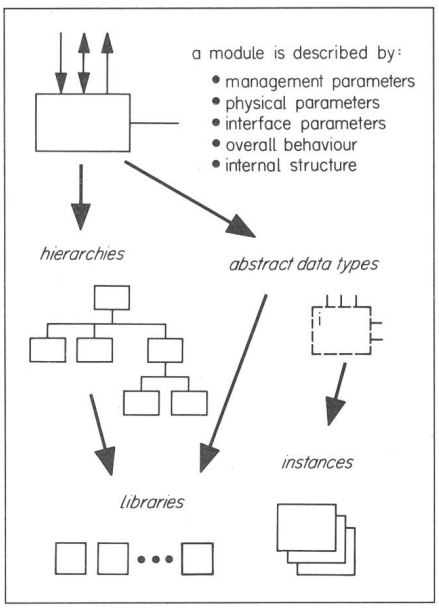

Fig. 3 Basic structuring element: module

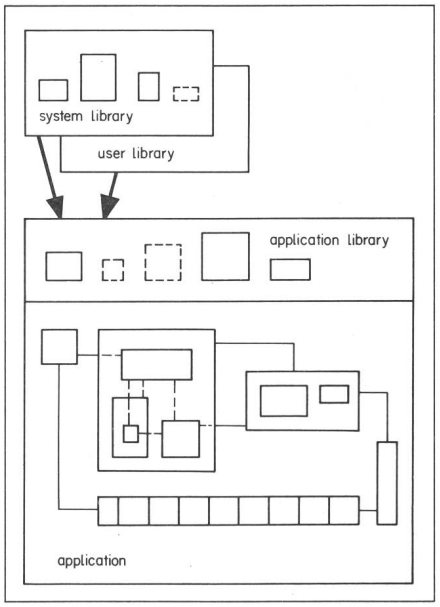

Fig. 4 Structuring an application

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.).

Figure 4 shows an example of application structuring. It is important to recognise that this structuring is performed on a given abstraction level. An efficient design system is expected to provide facilities for object-oriented manipulation ("syntax-driven editing"), debugging ("program-tuning step by step") and simulation ("execution") which, from the users sight, only deal with concepts, primitives and entities of that particular abstraction level and hide all non-related, often frustrating activities required by the underlying system (e.g.: file manipulations, program sequencing, 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").

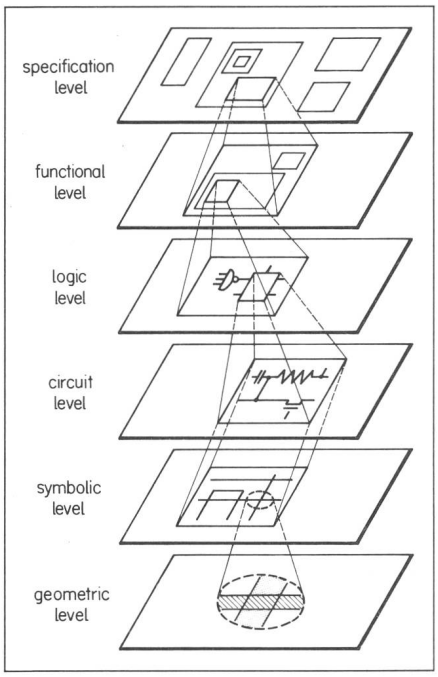

Fig. 5 Abstraction levels of VLSI design process

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.

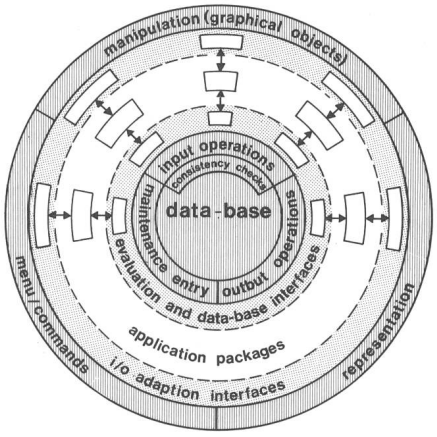

Fig. 6 Integrated CAD-system: integration concept

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 integration 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 committments 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".