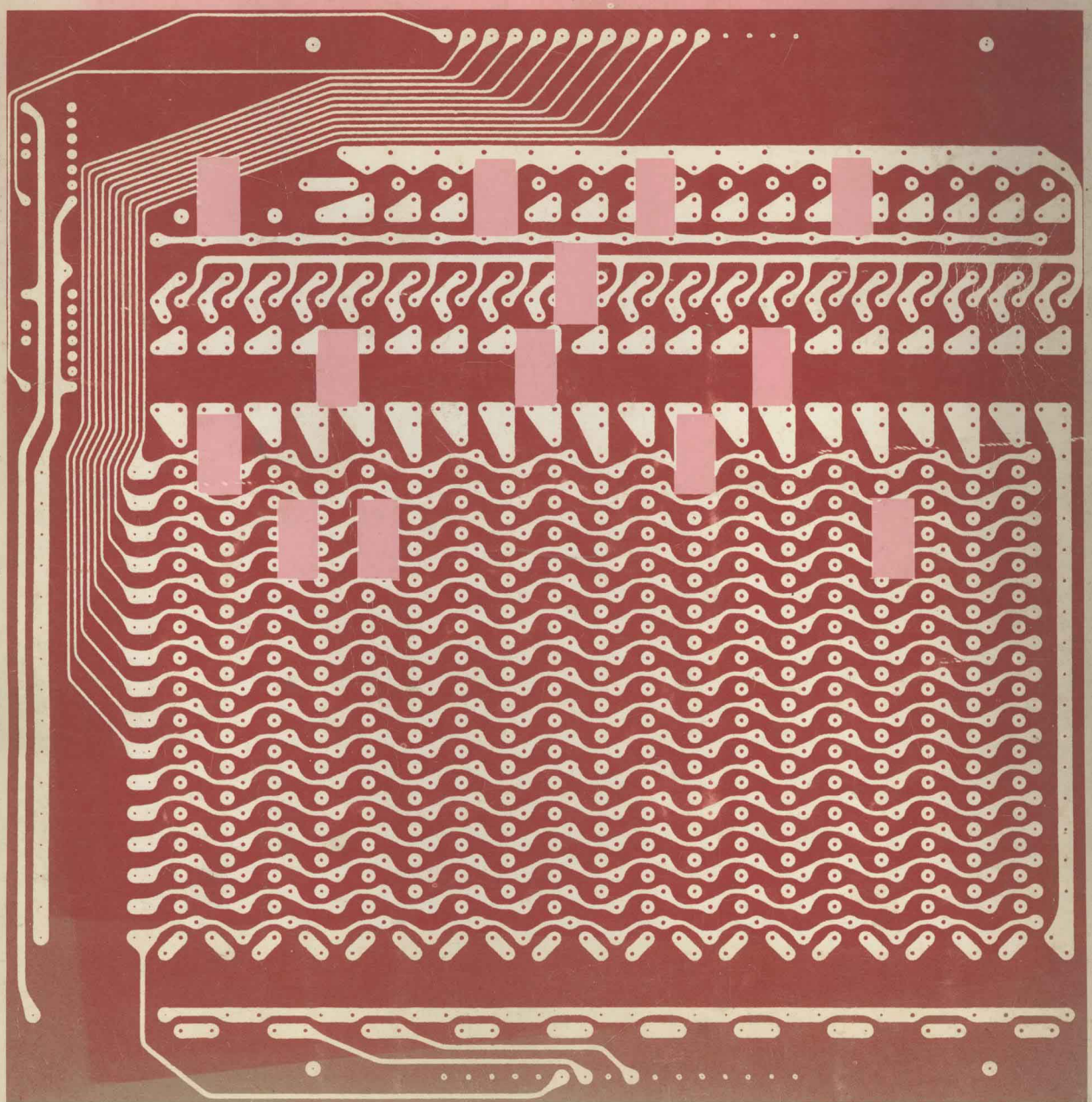
Computer aided design and manufacture of electronic components, circuits and systems



International Conference on

Computer Aided Design and Manufacture of Electronic Components, Circuits and Systems

3-6 July 1979

Organised by the
Electronics Division of the
Institution of Electrical Engineers

in association with the
British Computer Society
Institute of Electrical and Electronics Engineers
(Circuits and Systems Society)
Institute of Electrical and Electronics Engineers
(Region 8)
Institute of Mathematics and its Applications
Institution of Electronic and Radio Engineers

with the support of the
Convention of National Societies of Electrical Engineers of Western Europe (EUREL)

Venue
University of Sussex

Organising Committee

Dr R Spence *(Chairman)*
H G Adshead
Prof K Antreich, Federal Republic of Germany +
S Bird
Prof M Breuer, USA +
Prof L Chua *
Dr O P D Cutteridge * *
H De Man, Belgium +
R G Dixon § §
Dr S J Hewitt
Dr E Laksburg, USSR +
B Magnhagen, Sweden +
Dr R W McGuffin
G Musgrave
Prof K G Nichols * * *
Prof J V Oldfield
J F Pinel, Canada +
Dr A P Race §
Dr P E Radley
J C Rault, France +
J K Skwirzynski
M Tomljanovich, Italy +

§ representing the British Computer Society
* representing the Institute of Electrical and Electronics Engineers — Circuits and Systems Society
* * representing the Institute of Electrical and Electronics Engineers (Region 8)
* * * representing the Institution of Electronic and Radio Engineers
§ § representing the Southern Centre of the Institution of Electrical Engineers
\+ corresponding member

Published by the Institution of Electrical Engineers, London and New York, ISBN: 0 85296205 3
The cover was designed by GWA Design Consultants.

List of Authors

Contents

The Institution of Electrical Engineers is not, as a body, responsible for the opinions expressed by individual authors or speakers

Contents

Contents

Contents

Contents

VLSI - A CHALLENGE TO CAD

S. A. Szygenda

The University of Texas at Austin, Austin, Texas USA

INTRODUCTION

As we rapidly approach the era of Very Large Scale Integrated (VLSI) circuitry, Computer Aided Design (CAD) has acquired a new prominence and urgency. In the past, CAD was often considered a luxury or, at best, a tool to decrease the design and fabrication cycle time. However, as VLSI becomes a reality, CAD becomes an absolute necessity. The complexity and the sizes being considered in the realm of VLSI gives rise to this mandatory position for CAD. Therefore, those of us in the area of CAD are now faced with new and exciting challenges; to meet the needs of VLSI. These challenges are even more important due to the very large cost involved in designing CAD systems for the future.

It is not often that a particular area of the computing discipline, namely CAD, has the opportunities that presently exist. If in fact CAD does not rise to the challenge, and provide a set of tools to permit efficient and cost effective use of VLSI technologies, then, it is certainly questionable whether these new technologies can be effectively used. Therefore, the importance of the CAD challenges are evident.

CAD systems of the future will, in fact, be mandatory and essential, but it should be kept in mind that they simply constitute a set of tools, for the designer to more efficiently and cost effectively perform his function. CAD alone cannot be considered as a savior. CAD provides the various tools that are essential ingredients to the design process. The way the tools are used, and the effectiveness and cost of their use, are directly related to the total design philosophy and policies, implemented by the users. Therefore, those of us involved in CAD can only attempt to provide a repertoire of CAD tools, in the configuration of a total CAD system, to assist the designers in doing their job. Even though CAD is, in fact, essential, it is not and cannot be considered as the ultimate design tool.

In this paper we will first begin by considering an overview of a total CAD system for VLSI, and indicate those areas which would represent significant divergences from the traditional CAD systems being used presently. One of the particularly significant areas, is the testing of the VLSI devices and systems. This testing problem will then be considered in detail, in a later section.

OVERVIEW

The philosophy for a VLSI-CAD system is, to a large extent, different than CAD systems of the past and present. The cost of repeated low level iterations through; layout, detailed simulation, timing analysis, fault simulation, automatic test generation, etc. will be prohibitive for VLSI. Therefore, it is essential that more thorough testing and evaluation be done in the earlier phases of the system, where repeated iterations for more detailed accuracy and verification are cost effective.

Design Input

Traditional CAD systems have allowed for manual, equation, and/or graphical input, as major inputs to the DA systems. Inputs provided in this manner are normally manually translated or entered directly into the CAD data base. While this approach has been feasible in the past, it is rapidly becoming less feasible, and will be highly undesirable in the future; for a number of reasons.

First of all, the quantity of data for VLSI would make this approach prohibitive. Secondly, the data being entered into the data base will require considerably more validation than these input media allow. If erroneous data appears in the data base, via direct input, it is likely that these errors will not be detected until such time as layout, or detailed low level analysis, is performed. As indicated earlier, this is the most costly manner for performing this function. Therefore, an additional means for entering data must be provided by new CAD systems. This is usually termed behavioral, or structural, design language input. It should be remembered, that initial specification, design rules, built in tests, etc. should be considered concurrently with the behavioral or structural design language description.

Design Language

Using a high level design language input would require that the design specification be translated into a high level design language. The design language would then interface directly with a design verification system, both at the behavioral and structural level. The necessity for having both behavioral and structural design verification is that, initially, it is likely that simple behavioral characteristics would be all that would be available. Hence, design verification can begin at this level. As the design matures, structural detail would be specified and, to some extent, automatically generated from the behavioral specification. The design verification process would then continue, until the behavioral and structural descriptions are verified to whatever degree deemed necessary. This process could involve the elimination of numerous design errors and description errors. Furthermore, it could aid in the addition of testing hardware in the design; to test those areas of logic that were found to be insufficiently tested, by the design verification process.

Test Generation

One of the functions, of this design verification process, would be the generation of tests to validate the design. These tests could be generated manually, by the designer or other individuals knowledgeable of the design. The number and sophistication of tests would be dictated by the level of design verification required. The same tests would be supplemented and applied continually, as more structure is provided for the design and as lower level analysis takes place. It is suggested that this approach could vastly limit the amount of automatic test generation that would be necessary for the testing of VLSI devices.

Data Base

After the design has been adequately verified at this design level, data could be synthesized, in an automatic or semiautomatic fashion, into the data base. This could avoid numerous errors caused due to manual generation of the data base. It would also be essential that the data base have the ability to protect and coordinate all accesses to data. A possible method of accomplishing this function, would be to have automatic synthesis capability and a data base monitor. These would basically be the only two means for entering or retrieving data. In addition, the data base monitor could provide flexibility for expansion, in the future, as well as transparency of data structures. After the design has been validated at a high level, and the data base created, all further low level functions should be minimized; if the CAD system is to be cost effective. Ideally this number should be 1. This objective necessitates a thorough design verification at the higher levels, where continual iterations are considerably more cost effective.

After the data base is created, other major functions remain to be performed. This includes; physical layout and modeling, low level simulation, timing analysis, load analysis, fault simulation, etc. Considerable interaction could take place between the physical layout and low level analysis functions. For example, element level simulation, for design verification, should occur before physical layout. However, much of the detailed timing and load analysis, which would be part of the simulation process, may not be possible until after the layout has been performed. This interaction is, again, somewhat of a divergence from the existing uses of CAD systems.

Low Level Design Verification

In the low level design verification process, the tests, that were created in the high level design verification process, could be retrieved from a test file and applied at the lower levels. These tests should in fact be quite adequate for the design verification at the lower level. If it is determined that they are not adequate, then, it is questionable that the high level design verification was in fact sufficient. If the design had been verified extensively at the higher level, it should be expected that these tests could be used effectively throughout the remainder of the CAD system. This would eliminate, to a large extent, the need for automatic test generation, which would be extremely costly for VLSI systems.

Physical Layout

As physical layout proceeds, one would have to be concerned with engineering changes, or design errors, encountered at this level of the overall CAD function. These would necessitate making changes to the description provided in the data base, and could result in the necessity of going back to higher levels of the design description for reverification of the design. For example; if a design error, detected at a low level, results in the necessity of the design being reverified, it may be necessary to go back to the higher level design verification languages, change the description, validate further through high level simulation, and provide additional tests. This process in itself could require additional testing modifications to be made to the design, which could result in description errors and/or design errors, which could consequently require iterations through the entire design process. Therefore, the overall objective must be to minimize the number of these iterations.

Fault Simulation and Diagnosis

After the low level design analysis has been completed and the layout is in fact underway, fault simulation and diagnostic test production could be undertaken. Again, the test file, that had been used for design verification, would be the primary source of tests for diagnostics and would be used in conjunction with fault simulation. If thorough design analysis and design verification has been performed, the existing test set should result in a high percentage of fault detection, reducing or eliminating the need for automatic test generation. However, it is to be expected that automatic test generation capabilities should be provided, for very selective uses, in cases where the test sets are not adequate.

Outputs

The output of this type of CAD system would be two primary files. One containing layout data that could then be used for mask production, and the other being a test file which would contain tests that could be used at all levels of the testing process.

It should be obvious that a recurring theme, throughout this discussion, has been testing at all levels. The remainder of this paper will specifically consider these testing problems.

TESTING

Testing is an integral part of the entire design automation process, and may, also, be thought of as encompassing areas beyond those that would be considered part of CAD. We could think of the testing functions as including:

- design testing
- layout and fabrication testing
- field testing.

The objectives of these three areas are basically the same; namely, attempting to accomplish design, fabrication, and maintenance, with a minimum number of possible

errors and rapid detection and location of failures, for immediate repair. While they philosophically are considering the same design, with the same objectives, the approaches are considerably different. The major difference being the type of errors that could occur.

The objective, in the design testing category, would be to detect design errors, before they are propagated to other design automation and fabrication processes. This propagation would necessitate costly re-iterations through the design process or could result in designs with errors finding their way into the field.

The layout and fabrication testing area basically assumes a valid design and performs many different levels of testing, to determine errors that could have been introduced during the layout and fabrication processes. These include errors ranging from interconnection inadequacies, to errors that could have been introduced by the design automation process, to fabrication errors.

The field testing area assumes that one has a valid design that has at one time, after the checkout phase of the fabrication testing process, operated correctly. Consequently, failures that are likely to occur here would not normally include design and fabrication errors. Although, in some cases, due to the inadequacies and the non-comprehensive nature of these other areas of testing, these types of errors do, in fact, get into the field. However, the assumption is that field failures are those induced by environmental condition, aging, marginal tolerances, etc.

Design Testing

In the design area, one starts with the basic design and attempts to test this design to whatever extent is possible. For VLSI, this would involve testing from the highest to lowest level of design. A possible approach to this problem is a hierarchical simulation-testing concept. This procedure would provide complete design testing, from the top, down to the most basic device, before such time as fabrication testing begins. The more design testing that can be accomplished the better, but practical considerations must also be part of the total design philosophy. The highest level in the hierarchical structure would involve behavioral testing.

Behavioral Testing-Level 1. This level of testing would utilize the behavioral description of the system. The behavioral description could be provided directly by the designer, who would also provide the tests for this level of testing. The procedure would be one of providing the design and tests. Then, testing, with the aid of a high level simulator, to whatever level required for confidence in the operation of design, at this behavioral level. At this point the designer would proceed to the next level of testing, which would be the structural level.

Structural Testing-Level 2. Structural specifications of the system are basically created from the behavioral description, by adding design parameters which would begin to specify the structure of the system. The structural specification, at this level could be that required to provide input to a register transfer level simulator, to validate the structural design. Both the behavioral description, as specified in level 1, and the structural description, as specified in level 2, could be validated by user tests, through the use of simulation.

At level 2 the interactive nature of the testing philosophy begins to come into play. The tests developed at level 1 are saved and used for structural testing at level 2. Response comparisons, where possible, are then made to cross-validate the behavioral and structural descriptions. Additional tests may be required and these are then provided at level 2. This may in fact necessitate going back to the behavioral descriptions at level 1 for revalidation at that level. This particular loop would continue until such time as the user is satisfied with this level of testing.

It should be pointed out that it is also possible for a description to be provided directly in a structural form, instead of starting with a behavioral description. This would be true throughout the total testing hierarchy, that is, one can begin testing at any level. In fact, for the simplest VLSI building blocks, it is likely that one would begin with a bottom-up approach; beginning at the lowest level, instead of at the highest level.

The structural specification and testing at level 2 would not consider accurate timing information, or the specific hardware implementation to be used. It, instead, considers the design at an operational level. For example, if register A were to be added to register B, level 1 and level 2 would consider the add operation but not the specific adder implementation that would be used. This implementation would be added in the lower levels of the design.

Functional Testing-Level 3. Functional testing would be the first level of testing that operates on specific designs, to be implemented. Those operations that were specified in the behavioral and structural descriptions are now replaced with functional modules to perform the operations. In other words, an added level of detail has been provided, to the design, and more complete testing is possible and required.

A similar philosophy of testing occurs here as was described in level 2. Testing will be initiated by the use of tests developed at levels 1 and 2. Supplemental tests can be added, as necessary, and response comparisons evaluated until an acceptable level of testing has been achieved. It must be remembered that this may necessitate going back to higher levels of the description, for revalidation. The necessity to go back to previous levels of verification should be avoided, if at all possible, because these create costly feedback loops. This is particularly true the further down the design hierarchy one proceeds.

At the functional level of testing considerable amounts of timing and detailed analysis can take place, however, a significant level of detail is still assumed and not specifically tested. This level of detail would be provided in the modular and hybrid levels of testing.

Modular Testing-Level 4. Modules, as used in this context, are structural specifications for particular parts of the design. These would normally represent some subdivision of the description provided at the functional level.

In the modular testing block, one could consider detailed internal timing of the module. Whereas, at the functional level, nominal timing through the module would be used. In other words, more critical timing analysis and cost-performance tradeoffs can be accomplished at this level. As in all the previous levels, tests from the earlier levels could be applied for validation and response comparisons. Additional tests could then be added at the discretion of the user.

Hybrid Element Testing-Level 5. The hybrid element testing could be considered as the lowest level of design testing that is normally performed. At this level the primary element building blocks may be simulated. These could range from transistors to functional/modular elements, depending on the level of design verification testing that one desires.

For VLSI, ultimate testing would occur at the lowest level of description. It is envisioned that this level of testing could take place on primitive subblocks that would be considered in the physical layout of the chip. It is not envisioned that the entire chip would be tested at this particular level. The philosophy of a hierarchical testing scheme permits this approach to testing at the lower levels. It would be extremely cost prohibitive to attempt to test the total design at the lowest levels of description. Furthermore, the amount of data that would be generated, if one attempted detailed testing of an entire VLSI design as one entity, would be monumental. Therefore, it is likely that the building blocks selected by layout could be tested thoroughly as separate entities; avoiding the necessity of having to do detailed low level testing of the design as one complete entity.

At this level, one could proceed as described at the previous levels; cross-checking the responses from the higher levels of testing with this lower level testing and supplementing tests as required.

Circuit Testing-Level 6. The lowest level of the testing hierarchy would be the actual circuit level testing of the system. It is envisioned that, for VLSI systems, this level of testing would be used extremely selectively and primarily for thorough validation of the primitive building blocks, or cells, used in chip design. This could be the starting point for bottom up design of the primitives. Once the primitives have been thoroughly tested and evaluated, they could be considered as building blocks that could be used at higher levels of design and, consequently, at higher levels of testing. This particular approach could reduce the total testing required for custom designs.

At the completion of this hierarchical procedure, the user developed set of tests should thoroughly validate the design of the system. The advantage of this approach, in the testing philosophy, is that these tests could be used successively at each lower level. In a bottom-up approach this is extremely impractical. The task of translating tests from lower levels to higher levels present major difficulties and, in many instances, this could be impossible. In going from the top levels to the more detailed levels, the translation becomes much simpler,

Another distinct advantage exists in adopting this type of testing philosophy. That is, these tests can be used for testing in the areas of fabrication and field testing, i.e. diagnostics for fault detection and isolation. In fact, if an adequate level of testing has been performed for design verification, these tests should represent, by far, the major tests required for the other areas of testing. This could reduce the amount of automatic test generation that would be required for VLSI systems.

Layout and Fabrication Testing

Layout. The objective of layout testing is to insure that the physical design, represented by a design file conforms to certain types of design rules and specifications.

Mask testing begins with decomposed geometric mask entities (points, lines, shapes, figures, etc.). Configurations that realize functional modules and their interconnections are recognized, with the data developed here used in many of the remaining analysis phases.

One of the major problems in mask analysis for VLSI is that the sheer volume of data involved renders classical approaches ineffective. Instead, computational complexity must be controlled by exploiting the hierarchical nature of the design technique. This method could be based on thorough mask layout analysis of lower level constituent modules, and thorough characterization of layout attributes based on such analysis. In effect, we could carry over, into layout testing, the same hierarchical approach used for design testing.

For analysis purposes, each module external connection could have a set of information associated with it, including; source/sink strength, connectivity, timing, and functional description. At any level of the hierarchy, mask testing could combine module descriptors with the features used to realize interconnect, in order to develop the information necessary to accomplish testing.

Physical design rule checking deals with checking features and their neighbors for prohibited or out-of-tolerance configurations. One- or two-dimensional scans could be used to check pseudo-mask data.

Connectivity verification could trace all interconnect paths to check on missing or incorrect wiring between terminals of constituent modules. Path characterization data developed here could be used later for timing traces based on capacitance analysis and other information. Connectivity verification could, on a net-by-net basis, confirm that wiring required by the logical description of the module has actually been incorporated into the mask level layout.

Timing traces are signal propagation

procedures that develop the external capacitance and timing data for a module. These values could then be compared with nominal design values earlier specified for the module.

Loading analysis is closely related to timing analysis, and the two functions could be combined.

Fabrication. The final vehicle for validation would likely be a tester, capable of applying tests and observing responses. The necessity for this type of testing is twofold. Firstly, it may be that total built-in-testing was not accomplished. This would require outside testing for the detection of some fabrication defects. If, in fact, comprehensive built-in-testing had been accomplished, stand alone testers would not be required; i.e. the tester itself would be built into the design. This could be envisioned as the ultimate goal for testing and validation of fabrication defects and, in fact, other areas of testing also. On the other hand, if one assumes the built-in-tests are not sufficiently comprehensive, then testing for fabrication errors is a necessity.

Furthermore, design and diagnostic testing performed previously in the design cycle, using simulation models, would not be adequate as a replacement for stand alone testers. Conversely, stand alone testers would not be adequate for testing those things that were tested via simulation. Justification for these statements is quite simple, since testers test primarily for fabrication defects. These fabrication defects are faults other than those that would be modeled in a fault simulation testing environment and other than those that may be considered to be design errors.

A major limitation and cost of stand-alone testers is the acquisition of the tests to be applied through the tester. The procedure described in this paper alleviates, to a large extent, this problem, by adopting the hierarchical test philosophy that was described. In other words, the tests generated in the design verification cycle would represent the major portion of tests used in the failure testing area. Supplemental tests added in the failure test area would then constitute the primary test set that would be used for fabrication testing. Therefore, costly automatic test generation would be minimized.

It should be remembered that fabrication testing would constitute the final testing performed on the chip before being released. Therefore, this level of testing must be comprehensive, utilizing either the approach of stand-alone testers or built-in-test philosophies. It is my feeling that a built-in-test philosophy represents the trend of the future and could be the most comprehensive and cost effective approach to be ultimately used.

Field Testing

The topic of fabrication testing can realistically be considered to be part of a CAD system capability. Field testing, on the other hand, is usually not considered part of this function. However, it is felt that the total testing of a design must be considered in its entirety and not separated by artificial constraints. This paper divides field testing into four major areas of detection; tester detection, self test detection, error report detection, and non reproducible error detection.

Tester Detection. Detection of an error by a tester is a periodic detection technique and the assumption is made that the error has occurred, is solid, and now is being replaced by a tester. The maintenance procedure could be one of replacing the unit in the field and sending it back to the factory. At the factory it would be necessary to determine the fault characteristics that caused the failure. This usually would be done through the use of fault simulation and diagnosis. Performing this function is essential to determine what fault characteristics caused the particular unit to fail. Therefore, it is not realistic to state that one need only detect failures to particular units, since the units are going to be replaced. Isolation of the failure is also required to determine the failure mechanism causing the error, to prevent them from occurring in the future, or to anticipate failures of other similar units. It may be that isolation of the failure internal to the unit is not necessary during the field maintenance procedure, but isolation would be required later in the failure analysis process.

Self Test Detection. Detection by self test would be the most desirable form of detection because it is a concurrent form of detection, as opposed to a periodic type of detection. In this concurrent form, the testing is being performed simultaneously with the operation of the unit. If an error occurs, it is indicated immediately and need for detection by tester is essentially avoided. Once the error has been detected by this self test circuitry, the error indication could be given and replacement of the unit could take place. As in all other cases, the actual isolation of the failure must be accomplished to prevent future errors, or to take other corrective action, where necessary. So all techniques will in fact eventually end up in detailed isolation requirements for the failures.

Error Report Detection

Another method of detecting field errors could be error reports, by a user of the particular unit. Once the error report is received, field tests could be undertaken; usually with the use of a stand-alone tester, to attempt to duplicate the condition that was reported. If, in fact, duplication is successful, the same procedure could be followed as described under the section entitled detection by tester.

If an error cannot be duplicated with the diagnostic tests (but is reproducible with the user test), it would be necessary to gather all possible fault detection data and make the assumption that the periodic testing accomplished by the tester was incomplete. This implies that the test set which was, run through the tester, was incomplete and not capable of detecting the error. Under this condition, the unit could be replaced and sent back to the factory, to identify the particular fault, and to derive additional tests to detect occurrences of

these types of failures, in the future.

Non Reproducible Errors. Non reproducible errors, detected by any technique, are by far the most difficult to handle. These are usually termed transient or intermittent errors, but often they can be solid errors, that are not detectable by available tests. Particular sequences required to produce a error may be very complex, dependent on initial state conditions of the device, and, virtually, impossible to identically duplicate. Therefore, a error may actually be solid, but extremely difficult to reproduce. The approach normally taken in this situation is that a count is incremented, for this particular unit, saying that a non-reproducible error has occurred. After the count reaches some maximum number the unit could be taken out of service and sent back to the factory for detailed analysis. If the count is not at a maximum, the unit could be put back into service and used until an error indication was again received.

It again should be pointed out that built-in-self-tests can in many cases detect and isolate intermittent and transient type errors. This again substantiates the need for an evolution to this form of testing.

SUMMARY

This paper has attempted to briefly consider the impact that VLSI may have upon CAD. The major theme is that VLSI represents new and difficult challenges to CAD, as we know it today. This is largely due to the added complexity and size of devices to be considered.

CAD has advanced greatly in the past fifteen years. Today it is used quite effectively at the chip and board level. However, the size of VLSI introduces a computational complexity far beyond today's CAD capabilities. Consequently, existing systems fall short of requirements at the lowest level of VLSI design, as well as at higher levels. This situation is further complicated since the same size complexity that makes existing CAD systems impractical, prevents extensive manual design, validation, and layout. Therefore, it is this author's opinion that CAD systems for VLSI are mandatory, and the development of such systems will be very expensive. These new CAD systems will be extremely large software systems, that are fraught with the well known problems and costs of such software development efforts. However, the challenges exist, and if we are to meet these challenges we must proceed with these development efforts, now; or realize that we can fall years behind technology.

This paper has also suggested that testing represents one of the most severe problems and costs in the VLSI area. A discussion of various testing techniques and procedures were described. A hierarchical approach to the testing problem was proposed as a possible approach to handling this problem, in a cost effective fashion.

This research was supported in part by the DoD Joint Services Electronics Program through the Air Force Office of Scientific Research (AFSC) contract F49620-77-C-0101 and Comprehensive Computing Systems and Services, Inc., Austin, Texas.

OPTIMISATION AND DESIGN OF LINEAR NETWORKS, INCLUDING NESTED SUBNETWORKS, TO SOME FREQUENCY-DOMAIN SPECIFICATIONS

A.A.-F.Nabawi and K.G. Nichols

University of Southampton, U K

INTRODUCTION

Network optimisation by adjustment of network-element values (Director and Rohrer (1), Wing and Behar (2)) normally requires the evaluation of a penalty function P(p) and its gradient (1st-order sensitivity) ($\partial P/\partial p$) for each iteration of the P-function minimisation algorithm (Fletcher and Powell (3), Fletcher and Reeves (4), and Gill and Murray (5)). Here p is a vector of design parameters, e.g. element values to be adjusted and P is a measure of the difference between the actual and desired performance of the network.

In this contribution we describe a computationally-efficient method for finding the penalty function and its sensitivity for a network which includes embedded subnetworks. The method is used in a network optimization package and the strategy needed to progress an optimization exercise is also discussed with reference to a practical example.

For linear two-port networks in the frequency domain, the penalty function P can be quite simply related to the two-port admittance parameters Y11, Y12, Y21 and Y22 over the range of frequencies used in the specification of P. The major computational problem is then the evaluation of the parameters Y11, Y12, Y21 and Y22 and their 1st-order sensitivities ($\partial Y11/\partial p$), ($\partial Y12/\partial p$), ($\partial Y21/\partial p$) and ($\partial Y22/\partial p$) at a set of suitable frequencies. Since the problem is similar, but distinct, at each frequency, it is sufficient to consider just one such frequency. In general, the main network contains embedded subnetworks which in turn may contain further subnetworks embedded to any practical depth. This hierarchical structure is not only a convenience for network description but is also an essential requirement for computationally-efficient analysis which proceeds recursively through the embedded levels commencing at the deepest.

THEORY

Consider an (n+1) node subnetwork with input node 1, output node 2 and reference node (n+1). Representing the definite nodal-admittance matrix of this subnetwork by Y = {yrs}, where yrs is the matrix entry for the r-th row and s-th column, that of its adjoint subnetwork is Y transposed (Director (6)). Here we will denote a voltage or a current in the adjoint subnetwork by a circumflex over the symbol for the quantity: thus, for example, vr and $\hat{v}r$ are the voltages of, and ir and $\hat{i}r$ are the external currents into, node r, in the subnetwork and its adjoint respectively. By definition, the two-port parameters are given by,

Y11=i1 and Y21=i2 subject to v1=1, v2=0
Y12=i1 and Y22=i2 subject to v1=0, v2=1

The 1st order sensitivities of the two-port parameters with respect to an element of Y can be shown to be given by (Nichols and Nabawi (7))

$$\frac{\partial Y11}{\partial yrs} = vs*\hat{v}r \text{ subject to } \begin{matrix} v1=\hat{v}1=1 \\ v2=\hat{v}2=0 \end{matrix}$$

$$\frac{\partial Y12}{\partial yrs} = vs*\hat{v}r \text{ subject to } \begin{matrix} v1=\hat{v}2=1 \\ v2=\hat{v}1=0 \end{matrix}$$

$$\frac{\partial Y21}{\partial yrs} = vs*\hat{v}r \text{ subject to } \begin{matrix} v2=\hat{v}1=1 \\ v1=\hat{v}2=0 \end{matrix}$$

$$\frac{\partial Y22}{\partial yrs} = vs*\hat{v}r \text{ subject to } \begin{matrix} v2=\hat{v}2=1 \\ v1=\hat{v}1=0 \end{matrix}$$

To evaluate the two-port parameters, solutions of the subnetwork are required to find the input and output currents for two different excitations (1 volt input and 0 volt output, 0 volt input and 1 volt output). For the sensitivities of the two-port parameters, all node voltages of the subnetwork, and its adjoint, are also required for the same two excitations. All these solutions can be obtained by a method (7) which is computationally only a little more expensive than a single solution of the subnetwork. Just one LU factorisation of the admittance matrix, with first and second rows and columns deleted, is required involving of the order of (n-2)**3 arithmetic operations. This factorization is appropriate also to the adjoint subnetwork. Four forward and four backward substitutions, involving of the order of (n-2)**2 arithmetic operations, then suffice to find all node voltages for both the subnetwork and its adjoint for the two different excitations. From the node voltages for the subnetwork, the input and output currents are found by scalar multiplication involving of the order of (n-2) arithmetic operations.

In general, the vector p of design parameters influences more than one element of Y. The sensitivities of the two-port parameters with respect to p is therefore obtained from the chain rule as

$$\frac{\partial Y\ell m}{\partial p} = \sum_{rs} \frac{\partial Y\ell m}{\partial yrs} * \frac{\partial yrs}{\partial p}$$

where the summation is performed only over those values of r and s for which it is known that ($\partial yrs/\partial p$) is nonzero; see below. The first factor in each term is available in the form $vs*\hat{v}r$ from the analysis of the network as explained above.

In the analysis of a hierarchically embedded system of subnetworks, at any level of embedding, the calculated two-port parameters Yℓm are to be included in the admittance matrix at the next shallowest level. In addition, the sensitivities ($\partial Y\ell m/\partial p$) evaluated by the chain rule are available to simply determine the ($\partial yrs/\partial p$) at the next shallowest level. In some instances, a nonzero

$(\partial yrs/\partial p)$ is not carried up from a deeper level but results from an element of p defined as a design parameter in the current level.

The topological connection data for a subnetwork embedded in another determines precisely to which nodal-admittance matrix entries (yrs) the two-port parameters $Y\ell m$ of the embedded subnetwork will contribute. This information, together with a knowledge of in which subnetworks the elements of p are located, provides a highly efficient means not only for the recursive building of the nodal-admittance matrix, but also, for the recursive evaluation of the chain rule for $(\partial Y\ell m/\partial p)$ through the embedded levels. Note that it is topological connection information which selects the terms to be included in the chain-rule sum; i.e. the terms for which $(\partial yrs/\partial p)$ is nonzero.

In a typical circuit-design optimization exercise, the least-squares penalty function

$$P = \sum_{\omega\varepsilon\Omega} \frac{1}{2}\,\lambda(\omega) * \left[A(\omega,p) - S(\omega) \right]^2$$

might be used. Here $A(\omega,p)$ is the actual response (real, imaginary, magnitude or phase), $S(\omega)$ is the desired response (design specification), and $\lambda(\omega)$ is a weighting function. The frequency range is Ω and ω is a point within this range. The summation is performed for all such points ω for which an analysis is available (log- or linear-scaled equal intervals). The sensitivity vector is given by

$$\frac{\partial P}{\partial p} = \sum_{\omega\varepsilon\Omega} \lambda(\omega) * \left[A(\omega,p)-S(\omega)\right]^2 * \frac{\partial A(\omega,p)}{\partial p}$$

We consider, for example, the case when $A(\omega,p)$ is the magnitude of the voltage gain $G = -Y21/Y22$. Then

$$\frac{\partial G}{\partial p} = \left[Y21 * \frac{\partial Y22}{\partial p} - Y22 * \frac{\partial Y21}{\partial p} \right] / Y22^2$$

These quantities can be evaluated since the two-port parameters and their sensitivities have been calculated by analysing the network as already described. If G and $(\partial G/\partial p)$ are split into real and imaginary parts $(X+jY)$ and $(X'+jY')$ respectively, then

$$A(\omega,p) = \left[X^2 + Y^2\right]^{\frac{1}{2}}$$

$$\frac{\partial A(\omega,p)}{\partial p} = \left[X*X' + Y*Y' \right] / A(\omega,p)$$

and so P and $(\partial P/\partial p)$ can be evaluated.

A similar procedure is available for other response functions $A(\omega,p)$.

IMPLEMENTATION

The algorithm of the preceding section is implemented in an interactive program system suitable for use on small machines. The system is written entirely in FORTRAN (almost exclusively ANSI) and the current version typically requires about 48-kbytes of memory in a 'chained' or 'overlayed' environment. Any subnetwork may have up to 20 nodes but, because of the subnetwork embedding strategy, quite large networks can be analysed. Up to 15 levels of subnetwork embedding are currently allowed. Analyses may be performed for the two-port parameters (Y,Z,H and S; the latter being developed from Y) and the voltage gain of a network, and also for the first-order sensitivities of any of these quantities, at a single frequency or over a range of frequencies. The design parameter vector p has a maximum of eight elements which can be identified as subnetwork components at any depth of embedding.

At present only the voltage gain can be used in a least squares penalty function of the type discussed in the last section but the specification can be in terms of either magnitude or phase. The desired response can be specified by the user in terms of upper and lower bounds with no penalty incurred when the response falls within the bounds. The bounds may coincide or may otherwise be as close or as far apart as the user desires. The upper and lower bound values are described to the system at a number of discrete frequency points together with the weighting function to be used in the interval immediately above the point concerned. The system uses linear interpolation to obtain bound values for frequencies between the discrete points.

In a network-design optimisation exercise, the system attempts to minimise P; firstly, by analysing the network for P and $(\partial P/\partial p)$ and, secondly, by using an optimization package ((5), Lança (8)) for predicting a new value of p for which P is reduced. Note, realistic lower and upper limit values for the elements of p can be specified by the user. The two steps are repeated until a satisfactorily low value of P can be achieved or until no further reduction in the value of P can be effected.

RESULTS AND DISCUSSION

We present results of a design-optimization exercise on a frequency-selective amplifier (Rigby and Lampard (9), Idleman et al (10). The circuit of the amplfier is shown in figures 1, 2, 3 and 4 defined in terms of embedded subnetworks but is otherwise as in reference (10). An initial analysis yields a voltage-gain response with a centre frequency of 645 kHz and a Q of about 50 as shown in figure 5 and is in agreement with the results given in reference (10). At the centre frequency the calculated gain is 35dB.

The aim of the exercise was to move the centre frequency to 800 kHz and to seek a gain at this frequency of between 45 and 50 dB. The result of this exercise is shown in figure 6. The centre frequency is 801 kHz and the gain is 48dB.

Despite early expectations, it is our experience that fully automated network optimization is not a practical proposition. The reason is that, excepting very trivial examples, specification of the desired function and associated weighting functions, and the choice of the design-parameter vector, require adjustments as the total optimization exercise proceeds. The exercise must be split into a number of stages with refinements to the specification and parameter set at each successive stage based on the results of the previous stage. For any given exercise, the user follows a learning curve in which he rapidly improves his rate of convergence to the required design. Also, the user needs to be skilled and experienced in the 'ways' of the circuit he is designing; this is no task for the naive user. Often a number of