Reprinted from Gero, J. S., (Ed.), Artificial Intelligence in Engineering: Design. Elsevier Science Publishers, Amsterdam, Netherlands, pages 101-118.

An electronic design CAD system combining knowledge-based techniques with optimization and simulation

T. Ketonen, P. Lounamaa, J. K. Nurminen
Nokia Research Center, PO Box 780, SF-00101 Helsinki, Finland

ABSTRACT Recently intelligent CAD systems have been subject to considerable interest. The emphasis of the research has been on symbolic computing techniques and advanced mathematical techniques such as optimization and simulation have attracted less attention. We believe that the coupling of these symbolic and advanced numerical techniques is superior to the use of each one separately. Such an integrated CAD system has been implemented for analog electronic design. The system is used in the design of mobile telephones and radio links. Initial experiences of the users suggest clear increases both in the productivity of the design engineers and in the quality of the designs.

1. INTRODUCTION

Recently intelligent CAD systems have been subject to considerable interest (for an overview see (Sriram [11])). The application areas as well as the goals of the studies have been varied but most of them have been centered around the use of knowledge-based techniques (Kowalsky [7], Mitchell [9], Mittal [10]).

The use of design knowledge and symbolic manipulation is not adequate in applications which require extensive use of numerical values, such as in the design of analog radio circuits. Mathematical optimization, simulation and other advanced numerical techniques are needed, for instance, to find the best combination of parameter values or to verify the behaviour of a design. Since these numerical tools and techniques are of great help to a design engineer it should be clear that incorporating them in a knowledge-based CAD-system is beneficial. Forcing

different tasks to a unified framework, e.g., rule-based reasoning, is a common human mistake (Einhorn [3]) which should be avoided by selecting for each task the method or tool that is best suited to it.

The use of numerical techniques is often complicated since the user has to define the problem in a special way for each tool. Moreover, the selection of a proper tool for a particular task can be difficult. These decisions require the user to have considerable expertise. Experimental, innovative design is difficult in an environment which constantly forces the user to perform routine tasks.

Coupling symbolic and numeric computing (Kitzmiller [5]) offers new solutions to the above problems. Symbolic computing techniques can be used in problem formulation and in the interpretation of results but the actual calculations are performed using numerical routines. Combined with a graphical user-interface this technique allows the user to work with familiar concepts, such as design diagrams, and easily perform necessary analysis and synthesis operations. The mathematical details of the design tools are hidden from the user allowing him or her to concentrate on important design decisions.

The use of a multitude of tools is particularly important in analog CAD, since no single tool can adequately analyze the behaviour of a design. Radio frequency (RF) design is even more complicated because of the high signal frequencies. Therefore it has been a challenging testbed of our ideas in a project developing a CAD system called RFT (Radio Frequency Tools) for RF-design.

In the next section we represent an overall view of the RFT system. The roles of optimization and design knowledge in design synthesis are examined in section 3. An example of the functional level design of a mobile telephone is given in section 4. In sections 5 and 6 the use of simulators is discussed and the final section summarises our experiences and discusses the status of the work.

2. THE STRUCTURE AND IMPLEMENTATION OF THE RFT DESIGN SYSTEM

Most of the work on intelligent electronic CAD-systems has been concerned primarily with digital design (e.g. Brewer[2], Mitchell [9], Subrahmanyam [12]). Compared with digital design, the number of

components in a typical analog design is small but the functions and dependencies of analog components are very complicated. Therefore complex simulators and even laboratory prototyping are required to verify the behaviour of a design.

One of the main goals of the RFT-system is to help the engineer to use these different tools in an easy way. This is accomplished by a common graphical user-interface to all of the design tools. Whenever a certain task has to be done, the proper tool is selected and the design diagram is converted to the format required by the tool. In addition, a number of tool specific checking and correction rules are used to guarantee the validity of the tool input.

Besides tool-specific knowledge the system contains design knowledge which is used in cooperation with simulation and optimization tools to synthesize and evaluate designs. The design engineer has an integral role in the evaluation and modification of suggested designs, bringing flexibility and creativity to the system. The design tools perform routine tasks leaving only the most important design decision to the engineer.

The RFT system is implemented in the HP-9000 series workstation using its extended Common Lisp -environment. The most important non-standard extensions used are function calls to graphics and windows libraries, foreign function interface to C- and FORTRAN-routines, operating system calls and the object oriented programming system.

The object oriented programming paradigm has been used to represent the design entities. Since class definitions using the object system on the HP computer require considerable amounts of memory, the representation had to be simplified to reduce the number of different object classes.

Display windows, design diagrams and design elements, that is components and blocks, are represented as objects. Display windows take care of the actual display of design diagrams. Each diagram has a list of element-objects belonging to the diagram. Diagram-objects also have methods for manipulating the elements in the diagram, performing various analysis etc. The most important data in element-objects are attributes and connections to other elements. Elements also have methods which specify their behaviour in different conversions and

operations during the design process.

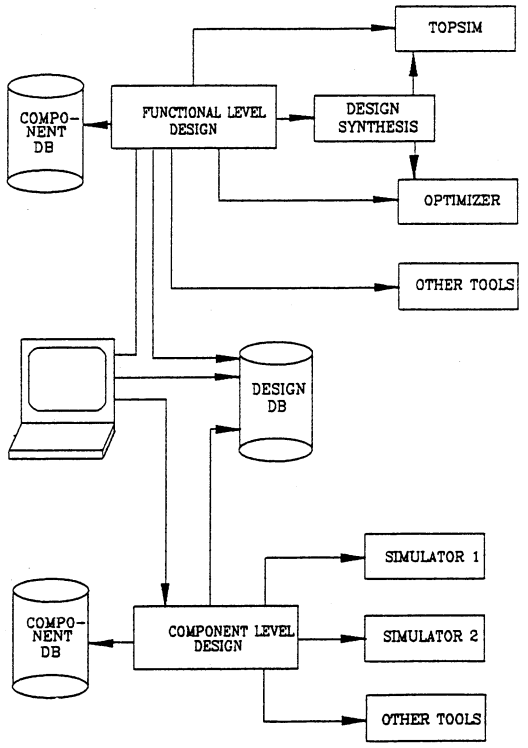


Figure 1. An overview of the system architecture.

The design objects are manipulated via a graphical user interface. Using the mouse and pop-up menus the user can build design diagrams, edit attribute values, control the use of different tools and the representation of their results. Although the system automates many phases of the design process a user-friendly interface is very important encouraging the user to explore new alternative solutions and in visualizing design diagrams and results of analysis.

A relational database is used to store the design diagrams, their components and other design data. The interface between the database (in our case Oracle) and Lisp is implemented using a layer of C-code which passes the database queries from Lisp to the database and returns the found rows. In the database, object instances are mapped to relations, each relation corresponding to an object type. The relations are automatically created in the database when new component types are added. If existing components

types are modified incremental changes to the database are made.

Most of the design and component type knowledge is stored in textual definition files so that they can be updated by well-informed design engineers. The knowledge is represented by macro-calls. The macros convert the knowledge to a suitable form for the system to use. For instance, the following functional block definition is a call to the Lisp macro def-func-element. It defines the corresponding object class and routines which, for instance, check that a proper number of input and output signal are connected to the block. This definition also contains data which is used to perform analysis such as the transformation functions and attribute conversion rules.

```
(def-func-element VOLT_CONTR_OSC
   :class sub
   :inputs 1
   :outputs 1
   :frequency_transformation "VCO_CONSTANT/S"
   :phase_transformation "VCO_CONSTANT"
   :attributes
      ((block_type nil string)
       (center_frequency nil frequency)
       (vco_constant nil vco-gain))
   :attribute_conversions
      ((vco_constant "vco_constant*2.0*PI"))
   )
```

As a side-effect the definition macros gather useful information to be used in other parts of the system. For instance, a file with the statements defining the database relations is created when the knowledge definition files are evaluated.

Structuring the system as discussed above allows the division of development work between the domain experts and the system developers. Using the knowledge definition files the users enter the design and component knowledge by themselves. This means that also the longer run maintenance of knowledge in the system can be carried out by the domain experts themselves. The system developers create general tools which utilize this user-specified knowledge.

3. DESIGN SYNTHESIS USING DESIGN KNOWLEDGE TOGETHER WITH OPTIMIZATION

The goal of product design is to develop a product

fulfilling a given specification. Moreover the design should be optimal in some respect when, for instance, cost, size and power consumption is concerned. These are normally conflicting criteria, and therefore pareto-optimal solutions are common in design tasks. Pareto-optimality (Zeleny, [14]) means that when one criteria is improved some other must be relaxed. The selection among these pareto-optimal designs is largely a matter of taste which depends on the business prospects and practices in the field.

The above description suggests that a design task can be represented as a multi-objective optimization problem. Although this is theoretically possible and feasible in some fields, few practical electronic design problems can be completely solved using mathematical optimization techniques. Most real design tasks are so complex that they cannot be given a precise mathematical formulation, or if the problem can be formulated, it is computationally too difficult to be solved in a reasonable time (Klein [6]).

Partitioning a design to subproblems reduces the complexity of the problem. The subproblems are solved and their solutions combined to obtain a complete design. The division can correspond to the natural abstraction levels of the design process, such as the functional, component and layout design levels, but each level may also be further divided into simpler subtasks. Such a decomposition seldom leads directly into a consistent solution. Instead a user-controlled iterative process is needed to ensure that the interactions between subsystems are properly handled.

Depending on their nature, the subtasks can be solved either by using design knowledge or by optimization. (By optimization we mean the use of well-known numerical mathematical optimization algorithms. Heuristic rules which are used to manipulate some features of a design, such as complexity, are considered to be design knowledge).

When feasible, the use of optimization instead of design knowledge is a preferable alternative since the optimization algorithm finds a mathematical optimum to a given problem. Using heuristic rules the optimal solutions are practically never found. In optimization the user can control the effect of various factors by weighing those variables that he or she considers important and experiment with emphasis on different features of the design.

Not all problems are suitable to optimization and for these heuristic techniques are needed. These problems can be solved either by using formalized design knowledge or by asking the design engineer for a solution. Whenever the design engineer is making design decisions, verification rules can be used in order to eliminate designs which violate known design constraints.

As an example of the above concept we analyze in the next section in more detail the functional level design of a receiver as implemented in RFT.

4. FUNCTIONAL LEVEL RF-DESIGN

RF-design consists of various abstraction levels such as functional, component and layout design. Starting from the functional level the design proceeds hierarchically to more detailed design levels. Since the design decisions at the functional level have the greatest effect on the performance and cost of the final device, efficient design tools are particularly important at this level.

The parameter values as well as the structure of the design are defined in a stagewise manner using a number of tools. Some design stages may have to be repeated if drastic modifications are later needed. The user can control the flow of the design process and modify any design decision made by the system. The main design steps of the functional level design are represented in figure 2.

The functional level of RF-design can be divided into structural design and parameter selection tasks. Structural design rules are used to suggest modification to the structure of a design diagram. The parameter values are fixed using both numerical and knowledge-based techniques.

Some parameters have fairly simple mathematical dependencies and their values can be solved using optimization. For instance, the gain parameters of consecutive blocks can be summed up to obtain total gain. In the case of mobile telephone design about one half of the parameters can be solved in this way but they are fortunately the most important ones.

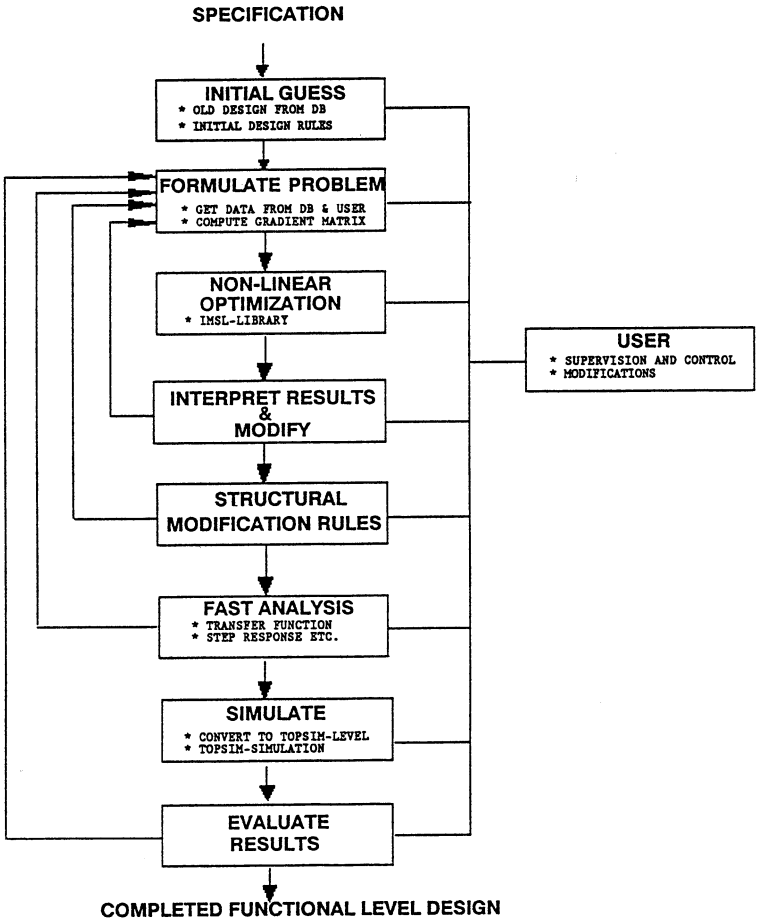


Figure 2. The flow of design synthesis.

In our case the original optimization task is quite complicated but by variable substitutions simpler non-linear optimization problems can be formulated. The number of variables in the model depends on the number of blocks in the design, but the number of constraint equations depends only on the design task. The forms of the constraint equations are easily derived from the specifications of the designed device.

The formulation of the optimization criteria is more complicated. Since the parameters selected in the functional level design form the specification of the component level design, the goal is to select them so that an acceptable component level design for each block can easily be found.

Each parameter has a range of possible values. One end of the interval represents the best possible value of the parameter, which is also the most difficult one to achieve in component level design. The other end-point (if any) is a parameter value which is easily obtained in the component level design. For instance, the gain of an amplifier may vary in the range [0.0; 15.0], the lower bound 0.0 meaning that no amplification is needed. Naturally, the component level design of an amplifier is very easy if the required gain is near 0.0. For some parameters, such as noise figure, the higher values are easier to obtain. A good heuristic objective function is thus to minimize the sum of the distances of parameter values from their easy-to-design end-points.

Because of the differences in the component level design, the difficulty to design a block varies from one block to another. Some blocks are fairly easy to design although their parameters have tight specifications. To take this kind of factors into account, the differences in the object function can be given weights according to their importance.

The optimization problem is thus the following:

$$\text{minimize} \sum_i w_i(x_i - l_i) + \sum_i w_i(u_i - x_i) \qquad (1)$$

subject to
the specification and bounds of x_i

where
x_i is the unknown parameter value,
l_i is its lower bound,
u_i is its upper bound and
w_i is the weight of importance.

The objective function is divided into two parts since some of the parameters are favorable small and some large. The weights w_i depend on the block type and are larger for blocks that are difficult to design at the component level. Generally the selection of wi is fairly easy and minor changes in their values have only little effect on the optimization results.

The non-linear optimization problem is solved using the successive quadratic programming algorithm implemented in subroutine NCONF/NCONG of IMSL-library (IMSL [4]). To be able to use the FORTRAN subroutines

of IMSL a FORTRAN-routine defining the optimization problem has to be generated. It consists of a main program and a subroutine defining the constraint and objective functions.

The efficiency of the optimization routine can be enhanced if a gradient matrix of partial derivatives is used. It is computed in symbolic form using a symbolic derivation routine (Abelson [1]). The gradient matrix can also be used if the designer wants to gain some insight of how each variable contributes to the value of a particular function. The entries of the gradient matrix in a particular point are approximations of changes in the functions values as the variable values are slightly altered.

The results obtained using the optimization routine are optimal with repect to the used objective function. However, since the objective function is defined heuristically there is no guarantee that the solutions are indeed optimal. To define optimality in this kind of situations is difficult since there is no well-defined numerically measurable goal. Using the heuristic objective to minimize the difficulty of component level design seems, however, to work in practice. Parameter values found by experienced engineers and automatically generated ones were very near to each other in a number of test cases.

Some blocks, for instance filters, have parameters with discrete values, the requirements of some of the parameters are different on higher and lower frequencies etc. To take these kind of factors into account a number of optimizations with minor changes in the problem statement are needed. Often the changes are based on the results of the previous optimizations and lead to fixing parameter values at various levels to test the behaviour of the design with, for instance, different signal frequencies.

The optimization routines can only be used to select some of the parameters. To fix the values of the rest of the parameters is more difficult. The dependencies of parameter values are so complicated that their effect on the performance of the overall design is very difficult or even impossible to compute. For instance, the behaviour of filters is characterized by their response curves. To define the form of the curve is numerically extremely difficult or even impossible in practice. In these cases simulations can be used to study the effects of these parameters. Heuristic situation specific design

rules, such as

```
(if (and (type-of block 'filter)
         (> (rf-frequency block) 400)
         (= (if-number block) 1))
    (setf (i-f-frequency block) 45))
```

are used to suggest values for some parameters and to propagate the effects of possible modifications.

Structural design is needed to add or remove some block in the design or change the order of the blocks. The need to change the structural design arises in two cases: when it seems likely to fill the specification with a smaller number of blocks or when it has been noticed that the specification cannot be fulfilled with the current structure. The former case leads to the reduction of blocks and thus to more economical designs. The latter is aiming at finding a feasible solution to the problem by rearranging or substituting some of the current blocks or by adding new blocks.

The decisions to modify the structure are governed by rules which are based on the optimization or simulation results. A typical rule like

```
(if (and (type-of block 'amplifier)
         (< (gain block) 2.0))
   (remove-block block))
```

specifies that if the computed gain of an amplifier is very small then it is likely to be unnecessary and can be removed.

The modification rules are often straightforward and their number is relatively low, less than 100 per design type. As a starting point a similar or resembling design can be read from the database or the user can supply one. Another alternative is to start with a maximal number of blocks which is known to satisfy the specification and then drop unnecessary blocks away as the structural design proceeds.

In summary, symbolic computing and heuristics are used in three ways in the design process: the former to formulate the optimization problem, the latter as a way to define performance criteria and a combination as rulebases that suggest structural and parametric modifications of designs.

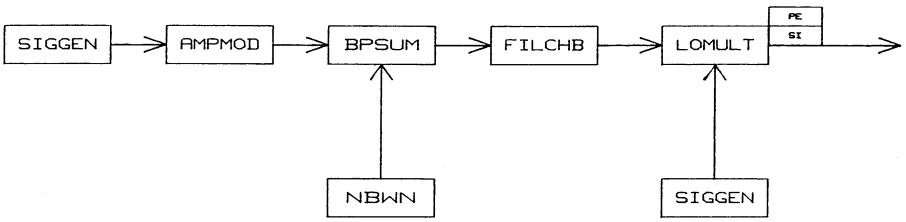
5. SIMULATOR INTERFACE

RF-design is a demanding task because of the complex interdependencies among the components. Prototyping is often the only way to validate designs. The use of simulators, however, has reduced the need for prototyping considerably. To simulate a design a number of simulators are needed, each with different capabilities. For instance, at the component level, separate simulators are used for linear and non-linear circuits. At the functional level a block-level simulator, TOPSIM (TOPSIM [13]), is used.

Although the use of simulations reduces the amount of time needed for prototyping it causes some additional complications which are especially evident when a number of simulators are used. The designer has to be familiar with the specific input and output formats of the simulators. Moreover, most simulators have some special "tricks" which the user has to be familiar with to get meaningful results and to avoid inconvenient run-time errors.

A unified graphical user-interface is a partial solution to the above problem. With it the user can work with familiar design diagrams. The design diagram is created using the mouse and menus and the values needed are supplied as parameters for individual blocks. Once the design has been created an automatic conversion to the input format of the simulator takes place, the conversion routine taking care of the syntax and naming conventions of the simulator. Figure 3. represents the graphical simulator diagram representation together with the generated input file.

This, however, is not enough to help the user to operate the simulators with ease. In most cases the user has to pay attention to simulator specific constraints. Some constraints are even undocumented and may result in misleading simulation results. Expertise to avoid errors like this is valuable and should be distributed to end-users.




```
INITIAL
DELT=5.0E-4
FINTIM=0.1
F1=10000.0
PARAMETER F2=20.0,40.0
DYNAMIC
X7:=SIGGEN(50.0,1.0,0.0)
X6:=AMPMOD(F1,1.0,0.0,0.5,1.0,X7)
X2:=SIGGEN(F1+100.0,1.0,0.0)
X1:=NBWN(1,F2,F1)
X5:=BPSUM(X6,X1,F1)
X4:=FILCHB(2,8,F1,100.0,1.0,0.2,0,X5)
X3:=LOMULT(X4,X2)
MEASURE PERSPE(0.0,100,0.0,200.0,-1,1,10.0,2,X3)
MEASURE SIGNOI(0.0,1/DELT,10,X3)
END
```

Figure 3. A simulation diagram and a corresponding simulator input file

In our design environment we use simulation knowledge to verify simulator input and warn users of illegal value or component combinations. There are two types of verifications, those done when the diagram editor is used and those which can be used only after the diagram is ready for simulation. At editing time fairly simple verifications are performed to ensure that a proper number of input and output signals are connected to each block and that these signal are of compatible type. The checks done when the whole diagram is ready are more complicated. They concentrate on the dependencies between the parameter values and the signal types. For instance, TOPSIM requires some signals to have a particular type. The type of a signal is determined at run-time by a threshold value which depends on the simulation parameters. An illegal signal type leads to an error which is noticed only after a considerable run-time. To find this error before simulation, the values of signal types are estimated and propagated through the

diagram to make sure that proper signals are used.

The above discussed simulation knowledge as well as the definition of the blocks known to the simulator are defined using Lisp-macros. The domain experts write the definitions and the tools for their use are embedded in the system. The following definition defines a simulator block ampmod.

```
(def-top-element ampmod
   :document "AMPMOD simulates an analog amplitude
              modulator."
   :class modem
   :inputs baseband
   :outputs analytic
   :attributes
    ((f0 nil frequency "carrier frequency")
     (amp 1 amplitude "carrier amplitude")
     (pha 0 phase "carrier phase in radians")
     (sens 1 am_sensitivity "modulator sensitivity")
     (amean 1 voltage "DC component"))
   :constraints
   ("f0 > 1.0/(5.0*delt)"))
```

The block has 5 attributes: f0, amp, pha, sens and amean. Each attribute has a name, default value, attribute type and help text. The attribute types are defined in a separate file and consist of value ranges, default units and alternative unit fields.

Input and output type information are used to check the consistency of connected ports. The constraints field is used for error checking to find illegal parameter value combinations. Some components have additional keywords describing exceptions to the normal use of the block, such as signal type modification rules or reversed parameter order in the simulator input. Since the definition of new components is fairly easy using this approach, the domain experts can perform the maintenance of the system.

The use of conventional simulators based on traditional programming languages is dictated by their availability and hard coded features. An alternative approach is to use Lisp-based simulators which facilitate the modelling work by allowing easy changes to the model and support experimental simulations by eliminating compilation, linking and data transfer times consumed when using a separate simulator (Lounamaa [8]). Because of the high quality

of modern Lisp compilers (Abelson [1]) and the possibility to declare types of variables the execution speed of Lisp-based simulators is comparable to, for instance, FORTRAN-based simulators.

6. CONVERSION TO SIMULATOR LEVEL

At the functional design level the simulators (e.g., TOPSIM) operate with ideal components. Since the real components have many non-ideal features they have to be modelled using a number of ideal simulator blocks. For instance, an amplifier can be modelled using an ideal amplifier block and a white noise random generator. Because of ideal blocks, diagrams describing simulation input are different from original design diagrams.

A transformational approach is used to convert a design diagram to a corresponding simulator diagram. However, the simulator diagrams can also be edited in the same way as the design diagrams and can therefore be modified if the user is not satisfied with the ones generated by the system.

The transformation rules are defined by relating each functional block to a corresponding combination of TOPSIM blocks, defining their connections and rules specifying their parameters. The following definition defines the conversion rule for mixer. The results of the conversion strongly depend on the parameter values of converted blocks. Conversion results with different parameter combinations are represented in figure 4.

```
;;; Conversion rule for Mixer
;;; The TOPSIM blocks used in simulation are listed
;;; in elements
;;; The connections of the blocks are given in
;;; connections
;;; Their relative locations are given in locations
(defconversion Mixer
   (if block_type
     ;; simulate mixer using a table of measured
     ;; values
     (progn (elements (bpnonl
                        :a block_type
                        :b block_type
                        :c block_type
                        :bkoff 0.0)
                      (lomult))
       (connections (input bpnonl)
```

```
                    (bpnonl lomult)
                    (input lomult))
      (locations (bpnonl 0 0)
                 (lomult 1 0)))
    (if noise_figure
      ;; add noise block to simulate noisy behaviour
      (progn (elements (lomult)
                       (nbwn :ix 1
                             :snr noise_figure
                             :f0 upper_rf_freq)
                       (bpsum :f0 upper_rf_freq))
        (connections (input bpsum)
                     (nbwn  bpsum)
                     (input lomult)
                     (bpsum lomult)
                     (lomult output))
        (locations (bpsum 0 0)
                   (nbwn  0 1)
                   (lomult 1 0)))
      ;; ideal mixer
      (elements (lomult)))))
```

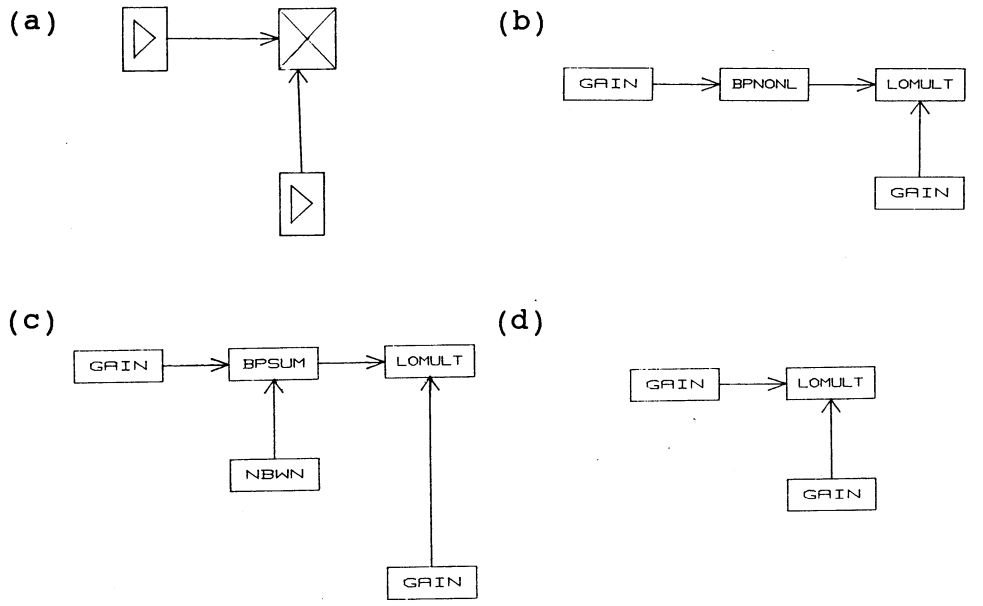


Figure 4. The use of a conversion rule to convert mixer to TOPSIM-level. (a) original functional level diagram (b) TOPSIM-diagram of a mixer whose behaviour has been measured (c) TOPSIM-diagram of a noisy mixer (d) TOPSIM-diagram of an ideal mixer.

For some components the transformations depend on the type of the simulated measurement and therefore a 8number of transformation rules are needed for each block. After the transformation is performed verification rules are used to ensure that the input is valid, after which the simulation is started.

7. CONCLUSIONS AND STATUS OF WORK

In this paper we have described how knowledge-based techniques have been integrated with simulation and optimization techniques in an analog CAD system. The resulting RFT-system offers easy access to different design tools as well as design synthesis options. The graphical interface of the system allows users to work with familiar design concepts and the tool-specific knowledge of the system takes care of necessary conversions.

Although the individual techniques used are already established, little work, at least to our knowledge, has been done to integrate them in a design system of industrial scale. Our experience indicates that AI- and OR-techniques can be successfully combined in a CAD-system. Although our application, RF-design, is a highly specialized field we believe that similar solutions could also be successfully used in other design tasks.

An early version of the system for functional design was delivered in December 1987 to a RF-design group. Objective evaluation of the system performance is difficult since the quality of designs depends on many factors which are difficult to measure and compare. The same applies to design times since no studies were conducted before the system was delivered. The subjective experiences of the design engineers suggest major speed-ups in design times and an increase in the quality of the designs. The improvements result largely from elimination of routine tasks, from the detection of design errors in early stages of the design and from the increased use of analysis and simulation tools.

Currently the system is being extended to support component level design.

8. ACKNOWLEDGEMENTS

The authors gratefully acknowledge Nokia-Mobira for design expertise and funding.

REFERENCES

1. Abelson, H. and Sussmann G. J.(1985). Structure and Interpretation of Computer Programs, The MIT Press, Cambridge, Massachusetts.
2. Brewer, F. and Gajski, D.(1986). An Expert-System Paradigm for Design, in Proceedings of the 23rd Annual Design Automation Conference.
3. Einhorn, Hillel J.(1982). Learning from Experience and Suboptimal Rules in Decision Making, in Kahnemann, D., Slovic, P. and Tversky A. (Ed.), Judgement under uncertainty: Heuristics and Biases, Cambridge University Press.
4. IMSL Math/Library User's manual(1987). IMSL, pp. 895-908
5. Kitzmiller, C. T. and Kowalik, J. S.(1987). Workshop report: Coupling Symbolic and Numeric Computing in Knowledge-Based Systems,AI Magazine, Summer 1987.
6. Klein, M. F. and Hodges, D. A.(1986). A Modular Framework for an Interactive Design Synthesis System, in Digest of Technical Papers of the IEEE International Conference on Computer-Aided Design, Santa Clara, 11-13 November.
7. Kowalsky, T. J.(1985). Knowledge Representation, Learning and Expert Systems, An Artificial Intelligence Approach to VLSI Design, Kluwer Academic Press, Hingham, MA.
8. Lounamaa, Pertti(1988). Advanced Modeling Environment for Dynamic Systems, to appear in Simulation and Modelling in Artificial Intelligence, John Wiley.
9. Mitchell, T. M., Steinberg, L. I. and Shulman, J.(1985). A Knowledge-Based Approach to Design, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-7(5), pp. 502-510.
10. Mittal, S., Dym, C. L. and Morjaria M.(1986). PRIDE: An Expert System for the Design of Paper Handling Systems, IEEE Computer, July 1986.
11. Sriram, D. and Joobbani, R. (Guest Editors)(1985). Special Section on AI in Engineering, SIGART Newsletter(92);38-127.
12. Subrahmanyam, P. A.(1986). Synapse: An Expert System for VLSI Design, IEEE Computer, July 1986.
13. TOPSIM III User's manual(1983), Torino Polytechnico.
14. Zeleny, M.(1982). Multiple Criteria Decision Making, McGraw-Hill, New York.