INTERNATIONAL JOURNAL OF CIRCUIT THEORY AND APPLICATIONS, VOL. 16, 371–390 (1988)

# SIMULATED-ANNEALING-BASED OPTIMIZATION OF COEFFICIENT AND DATA WORD-LENGTHS IN DIGITAL FILTERS

FRANCKY CATTHOOR AND HUGO DE MAN
*IMEC, Kapeldreef 75, B-3030 Heverlee, Belgium*

AND

JOOS VANDEWALLE
*ESAT Lab., K.U. Leuven, Kard. Mercierlaan 94, B-3030 Heverlee, Belgium*

## SUMMARY

The accurate characterization of worst-case limit-cycle behaviour and other finite word-length effects in digital filters constitutes a challenging and important optimization application. Indeed, for a chip implementation this can lead to a significant reduction in the signal word-length and hence to considerable savings in terms of the final chip area. This paper describes a new and accurate method for this investigation which is based on a simulated annealing procedure with a general scope. Novel methods for dynamically and efficiently updating the essential parameters of the annealing schedule keep the required CPU times reasonable.

The quality of the results obtained with our optimization routine and the general applicability of our approach are further substantiated with some promising results for the quantization of coefficients in arbitrary digital filters.

## 1. INTRODUCTION

Kirkpatrick *et al.*[1] have introduced a useful analogy which exists between the process of solidifying a specific class of fluids (systems with many degrees of freedom as studied by statistical mechanics) up to its equilibrium, and the properties exhibited by the convergence of combinatorial optimization methods, especially for those solving NP-complete problems with a large number of variables (or parameters). This relation can be expressed in terms of a number of similar concepts which are common for the description of both processes.[1,2]

So what can be learned from this analogy? On the assumption that functional minimization is required, in most conventional optimization methods newly generated states are only accepted if the corresponding objective function ameliorates. But it is probably advantageous to allow also 'up hill climbing' under certain conditions. Hence, in the 'simulated' annealing approach, a control parameter is defined which plays the same role as the physical temperature; and while lowering this 'artificially' introduced $T$, the increments in the objective function which are still accepted have to become smaller and smaller (on the average at least).

In that way, the temperature defines a measure for the coarseness in the search for an optimal solution: as long as $T$ remains high, the detailed shape of the objective function as evaluated over the search-space is less relevant, similar to a photograph taken from a very high altitude (bird's eye view). Afterwards, the height is gradually lowered and a decision has to be made which 'valley' between the ridges is selected for a closer look. A practical example of an objective function, as a function of a single parameter value, is illustrated in Figure 1. It shows the 'hierarchy' is the minimization process: first the valley on the right will be selected, and then a finer search will end up with the global minimum.

Consequently, in the simulated annealing schedule, the *crucial decisions indicate the temperature where a lower level of hierarchy is reached during the search*. And it is very important to detect these branching points because overlooking a single one can lead to a far-from-optimal final result. As the main effort

0098–9886/88/040371–20$10.00 *Received 30 June 1988*
© 1988 by John Wiley & Sons, Ltd.

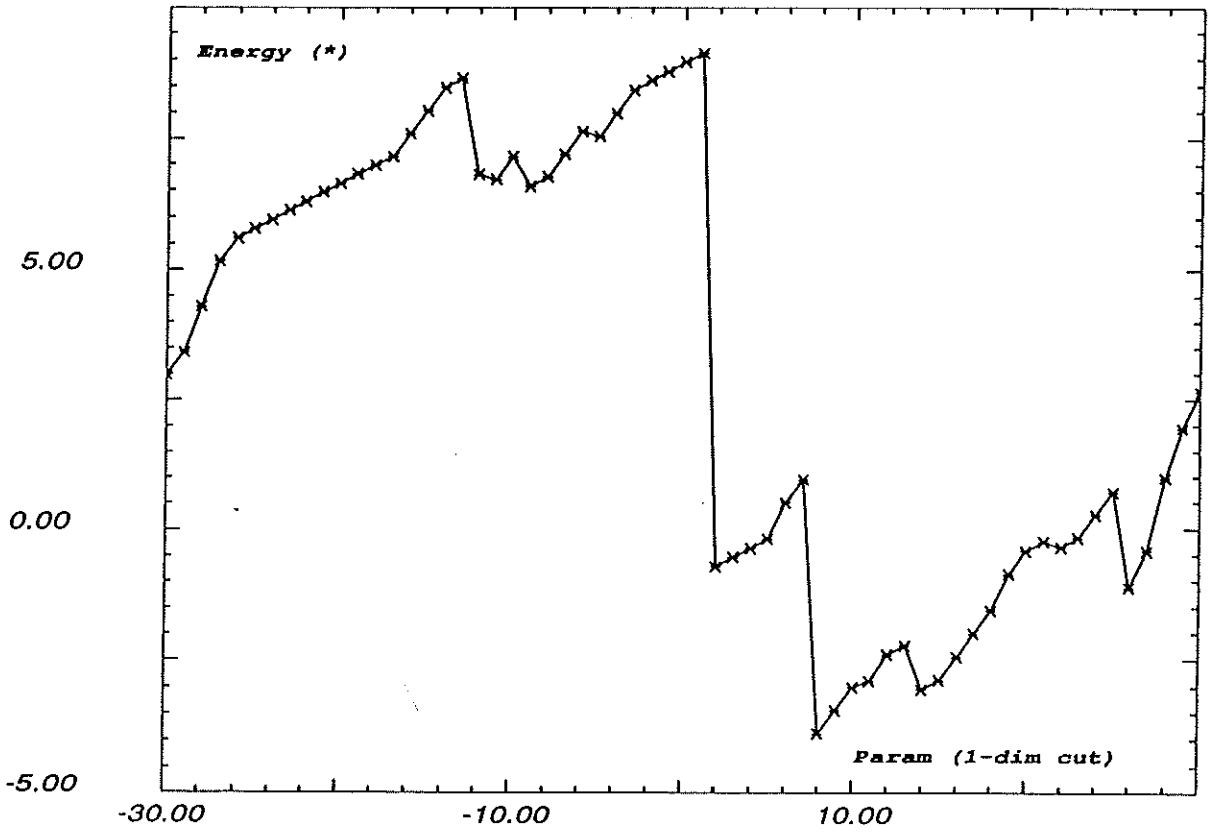



Figure 1. One-dimensional cut of typical objective function shape (sampled from a practical example for the investigation of finite word-length effects with SAILPLANE). The objective function (energy) is evaluated as a function of a single parameter value

should be spent at these critical temperatures, the efficiency of the applied schedule can be substantially increased by introducing adaptive annealing parameters[2] instead of fixed ones. Indeed, from the discussion above, it can be concluded that the annealing parameters should be controlled more 'strictly' in the neighbourhood of the branching points, resulting in the temporary expense of more objective function evaluations, but they can be relaxed otherwise. It is therefore very desirable to vary the number of evaluations spent at a specific temperature along with the decreasing temperature. Also an adaptive scale factor for the temperature (varying over a continuous range between a minimum and maximum) should be imposed instead of a selection between two fixed values. It should be stressed however that up to now, fixed schedules have been applied almost universally.

The concepts mentioned above have been integrated in an efficient simulation annealing schedule[2-4] constituting the core of the CAD tool SAMURAI which can handle very large, usually NP-hard or NP-complete, optimization problems. The acronym SAMURAI stands for *s*imulated *a*nnealing *m*odule with *u*pdated *r*ange of moves, adaptive *a*nnealing schedule and *i*nner-loop criterion. This Pascal routine is almost independent of the target application which is tackled. Initially, it has been employed for the investigation of finite word-length effects, such as limit-cycles, in arbitrary digital filters (Section 2). But the wider scope will also be demonstrated in Section 3 with some other (NP-complete) optimization applications such as clustered travelling salesman problems and the optimal quantization of fixed coefficients in digital filters by reducing the number of non-zero canonical-signed-digit (CSD) bits[5] which have to be implemented.

In this paper, it is *not the intention to comment on the efficiency of the simulated annealing technique in general or to analyse the improved characteristics of our annealing routine SAMURAI in particular.*

These topics are explored in a detailed way in another paper.[2] Instead, we will *present the new theoretical model which has been employed for the digital filter applications, the integration with the annealing routine and the very promising experimental results which have been obtained.* The latter will demonstrate the efficiency of our approach, but it sould be stressed that owing to the NP-hard nature of the optimization problems tackled it is not possible rigorously to prove that they are indeed 'optimal'. For larger test vehicles, only an indication of the effectiveness can be presented, based on the excellent convergence properties of the annealing.

## 2. THE INVESTIGATION OF FINITE-WORD-LENGTH EFFECTS IN DIGITAL FILTERS

Under finite-word-length conditions, digital filter systems are subject to non-linear operations. In many cases these will result in distortions, and eventually in a number of affected signal bits which have to be skipped at the output nodes because they do not contain useful information. As the data word-length constitutes an important parameter during the implementation phase, it is necessary to characterize these non-linear effects in an accurate way to avoid wasting hardware as much as possible. In the following subsections, the problem will be defined in detail and a formal model will be derived. For this mode, our new annealing module will be employed to provide an acceptable solution.

### *2.1. Problem definition*

Given a digital filter, which satisfies stability criteria, general filter theory guarantees that the output as well as the internal state variables will gradually converge to the zero state under zero-input conditions. However, this property only holds if all arithmetic operations are performed with an 'infinite' precision. Unfortunately, the latter condition is not generally realizable in practice, so in general it becomes necessary to quantize (e.g. round or truncate) the outputs of, for instance, power-of-2 scaling operators (indicated as scalers henceforward) or multipliers to a finite signal word-length. When the finite-word-length system is not excited any longer (zero-input condition) a steady-state behaviour will be exhibited. In many practical cases, the *quantization behaviour will result in* residual granularity oscillations or *zero-input limit cycles*,[6-10] usually periodic with a bounded period.

A simple example (also used further on) can be illustrated for the second-order filter in Figure 2(a) with Scal1 = 1/2 and Scal2 = 1/4. Starting from the initial state (−1, 1) in the two internal registers, the system is forever alternating between (−1, 1) and (1, −1).

But other types of limit cycles are also important in practice: under normal operation, a digital filter will be *continuously supplied with input signals* (e.g. constants or sinusoids), and the (worst-case) error between the ideal response and the observed response due to quantization effects is also a valuable

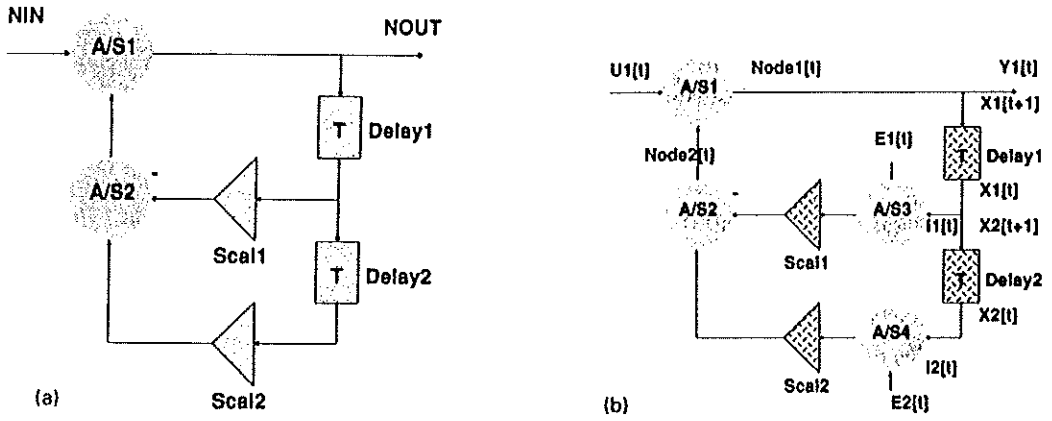



Figure 2. Simple second-order section: (a) block diagram; (b) modified flow graph with illustration of notations used in Theorem 1

property. The filter behaviour under overflow conditions is of interest too. The latter can result in the presence of *large-scale limit cycles*[10] which persist after the input has been reduced to zero again. A related property concerns the *forced-response stability*,[10] which verifies if the filter recovers after the occurrence of overflow, namely whether the internal nodes exhibit any type of granularity oscillations superposed on the 'ideal' response under arbitrary input excitations.

In order to reduce the effects of overflow under 'normal' behaviour as much as possible, a (minimal) number of guard-bits has to be included at the MSB-side (most significant bit). For this purpose, a *scaling analysis* has to indicate the maximal signal levels which can be allowed at the external inputs without giving rise to an internal overflow condition. Alternative methods such as the L1-norm analysis[11] provide a general upper bound which does not take the specific characteristics of the supplied excitation signals into account. The designer could, for instance, be interested in the largest amplitude which can be tolerated for a pure sinusoid source.

All these classes of 'finite-word-length effects' are undesirable in the eventual hardware implementation, especially when audio signals are filtered, as they will always be heard in the absence of any other significant signal. In order to deal with these effectively, two questions have to be answered:

(a) Can residual oscillations occur in the particular system realization?
(b) If so, find the *maximal amplitude* measured at the output or other important observation node(s) such as internal states; and determine its characteristics (period, DC-level).

When this worst-case signal and its amplitude have been determined, the designer can verify whether they still fall within the specifications on the distortion level (which usually differs from the allowed noise levels). If satisfactory, no additional actions have to be considered. Otherwise a number of additional LSB-bits have to be introduced which are skipped at the final output. In both cases, the bits which are (can be) affected by these parasitic effects, have to be implemented without actually contributing to the desired filter action, so they should be avoided for cost reasons. Indeed, it has to be taken into account that the *eventual hardware-cost*, e.g. measured as chip-area, *increases proportionally for every additional data-bit* in bit-slice orientated or bit-serial[12] (semi-)custom VLSI implementations. Note that it has been assumed *a priori* here that an accurate estimation of the number of wasted bits is available. The situation becomes even worse (and area is wasted) when some more pessimistic upper bound has to be taken into account.

In many cases, the designer can also try to eliminate the source of the oscillations by *selecting an appropriate quantization and/or overflow mechanism*, e.g. by changing a twos-complement value truncation into a rounding operation after the scalers. This will lead to different values for the quantization errors, and probably to another steady-state behaviour. The modified filter has to be analysed again to investigate the new estimates for the error amplitude. Usually, this process has to be reiterated until all undesirable effects have been removed or sufficiently reduced in magnitude.

For some classes of filters, such as the wave digital filters,[10] the absence of small- and large-scale limit cycles can be guaranteed, provided that quantization and overflow characteristics (namely magnitude truncation and saturation respectively) requiring an increased hardware cost are built in at some specific critical nodes. Hence, in these cases the designer can still save chip area by investigating the specific properties which are really needed in the filter design to meet the amplitude specifications for the parasitic oscillations with a minimal cost. He/she can, for instance, try to exchange some sign magnitude with the simpler value-truncation operation which is inherently built into the twos-complement arithmetic.

To the best of our knowledge, *no general strategies are available* systematically to determine an accurate estimate for the maximal amplitude of limit cycles or other effects present in the considered implementation. Only techniques to compute upper bounds on the limit-cycle behaviour were originally proposed for second-order sections[6-8,13,14] and for arbitrary filters.[9] The last approach also computes bounds on general error signals.[9] Moreover, in most techniques the nature of the implemented quantization characteristics is restricted. In order to change this situation, the CAD-tool *SAILPLANE* aims at providing the digital filter designer with a practical solution. It *combines an integer linear programming (ILP) formulation* of the problem (as described in the following subsection) *with the use of a general simulated annealing procedure*[2-4] efficiently to guide the search for worst-case limit cycles

and other finite-word-length effects. The acronym stands for *s*imulated *a*nnealing and *i*nteger *l*inear *p*rogramming for *l*imit-cycle *a*nalysis, *n*orms and *e*rror-bounds.

It should be stressed that, in this way, all analysis types which have been listed above and which are of major interest in a practical design environment (see also subsection 2.4) have been implemented in SAILPLANE. This tool-box allows for the designer to *investigate all aspects of the finite-word-length behaviour* (and not only zero-input limit cycles) and to make an *'optimal' decision concerning the signal word-lengths.*

### 2.2. *Integer linear programming formulation*

When fixed multipliers are expanded into shifts and add/subtracts, an arbitrary multi-rate digital signal flow graph is constructed out of adders, subtractors, power-of-2 scalers, registers and switches. Hence, it can be described with linear state equations. However, in the eventual finite-word-length implementation these equations are not sufficient to describe the influence of quantization effects. To model the exact behaviour of the actual hardware realization, a set of relations between bounded integer-valued variables has to be defined by extending the state model, as specified in the following theorem.

*Theorem 1*

The time-domain behaviour of an arbitrary digital filter constructed out of adders, subtractors, power-of-2 scalers, registers and switches, is fully defined in terms of finite-word-length effects by means of the following set of equations and inequalities (see Figure 3 for symbol definition).

$$X_n[t+1] = \sum_k a_{nk} X_k[t] + \sum_i b_{ni} U_i[t] + \sum_s c_{ns} E_s[t]$$
$$Y_l[t] = \sum_k d_{lk} X_k[t] + \sum_i e_{li} U_i[t] + \sum_s f_{ls} E_s[t] \tag{1}$$

$$E_j[t] = f(I_j[t], \mathrm{Val}_j) \tag{2a}$$

with $f$ expanded in a set of linear equations and inequalities, e.g. (5), and

$$I_j[t] = \sum_k g_{jk} X_k[t] + \sum_i h_{ji} U_i[t] + \sum_s r_{js} E_s[t] \tag{2b}$$

$$-2^{(\mathrm{WL}-1)} \leqslant \mathrm{Node}_m[t] < 2^{(\mathrm{WL}-1)} \tag{3}$$

An example of the applied notations is illustrated in Figure 2(b) for the original flow graph of Figure 2(a).

*Proof.* In a digital filter system (such as the general one of Figure 3 or the example in Figure 2(b)), the next states $X_n[t+1]$ (enumerated from 1 to $N$, i.e. the filter degree if canonical), and the external

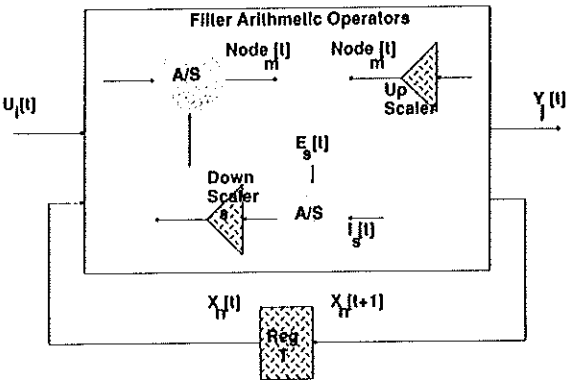



Figure 3. General symbol definitions for application of Theorem 1

output(s) $Y_l[t]$ (index $l$ ranging from 1 to 0) can be computed from the previous internal states $X_n[t]$, the external input-signal values $U_i[t]$ ($i$ ranging from 1 to $I$) and from the errors $E_s[t]$ occurring at the quantization nodes (namely the scalers $s$ from 1 to $S$). These commonly used relations are made explicit in (1). In their turn, the error values $E_s[t]$ can be modelled as integer-valued additive sources at the scalar inputs (see Figure 3). They are computed from the values of the scalar inputs $I_j[t]$ and the coefficients $\text{Val}_j$ as shown in (2a). For this purpose, the scalar inputs $I_j[t]$ also have to be computed as in (2b), with an expression similar to the one used in the set (1). If for instance value truncation is considered, the function $f(\ )$ is made explicit as

$$E_j[t] = I_j[t] \bmod \text{den}_j \tag{4a}$$

with $\text{den}_j$ representing the denominator (i.e. a power of 2) of the $j$th scaler. For the Scal2 coefficient in Figure 2(b), this equation would become

$$E_2[t] = I_2[t] \bmod 2^2 \tag{4b}$$

Indeed, when 5 is applied at the input of Scal2 the error value will be 1 LSB if considered from the input end.

In order to complete the description, the overflow mechanism has to be included. As a result of this, all node values are bounded due to the restricted word-length WL. Actually, if a twos-complement notation (and the corresponding wrap-around characteristic) are adopted, this will lead to a set of somewhat more complicated inequalities of the form (3) where the nodes $\text{Node}_m[t]$ are evaluated in the same way as the other left-hand-side variables in (1). In practice, however, it suffices to restrict the constraints in (3) to the output nodes of operators which can be subject to an increase in magnitude, i.e. the adders/subtractors and the up-scalers.

The last problem is located in the description of the function $f$ in (2). For the case of the value truncation, as shown in (5), the modulo operation can in general be rewritten in terms of a linear equation in combination with an inequality bounding the error value. The integer factor $m_j[t]$ has to be introduced in order to allow the error to fall within the appropriate range as imposed by the inequalities in (5).

$$E_j[t] = I_j[t] - m_j[t]\,\text{den}_j, \qquad \text{with} \qquad 0 \leqslant E_j[t] < \text{den}_j \tag{5}$$

In this way a *set of linear equations in terms of bounded integer variables* is obtained which have to be evaluated repetitively in a specific sequence. It should be noted that some equations in (1) and (2) can be simplified when the intermediate values of the scalar inputs are also introduced at the right-hand side. It can be easily verified also that, in the absence of delay-free loops, data-flow analysis on the block diagram of the filter will end up with a correct order to obtain unique values of all these variables.

For our goal, we have to compute the (maximal) signal amplitude which occurs at the output node(s). We will assume here that the evaluation starts from some (user-supplied or randomly chosen) initial filter state. When zero-input limit cycles are considered, the excitation nodes $U_i[t]$ remain all zero. Then the set of equations as defined above will be evaluated $p$ times, with $p$ large enough to enter the steady-state regime. Unfortunately, appropriate values for $p$ largely depend on the application, so the user has to enter some choice based on knowledge acquired during previous experiments.

Instead of systematically looking for the maximal observation values $V_k[t]$ encountered during all iterations, only the results $V_k[p]$ at time $p$ are examined. In the case of a periodical steady-state behaviour with period $P$, both options will arrive at an identical maximum when the search is considered over the set of all initial states, in other words if all of them can be selected with some non-zero probability. Indeed, if the maximum amplitude occurs for the $m$th time-instant in the oscillation, we can start from the $(m - p \bmod p)$th state and, after $p$ iterations, we will arrive at this maximum, as required.

The previous considerations directly lead to an integer linear programming (ILP) formulation. Indeed, the objective function can be define above as a weighted sum of a number of observation values $V_k[p]$, and is thus in principle dependent on a number of integer variables. However, these $V_k[p]$ are in their turn dependent on the many previous input and state variables and also errors (1)–(3)). By means of substitution, the dependent variables could be eliminated so that only the independent (unreducable) and

integer variables remain, namely the original states $X_n[0]$ and the input values $U_i[t]$ for $0 \leqslant t < p$. Unfortunately, this substitution is not always simple to perform symbolically but for our purposes (see subsection 2.3) it suffices repeatedly to evaluate these equations numerically for a given set of dependent variables $\{U_i[t], X_n[0]\}$. In addition, a number of inequalities have to be met as defined in (3) and (2) or (5) of the form $-B \leqslant x[t] < B$ (where $B$ represents the bound of the interval).

In summary, the goal is thus to *determine the set of variables* $\{U_i[t], X_n[0]\}$ *which optimizes the objective function within the constraints of the equations and inequalities*. This defines the ILP.

Up to this point, the objective function corresponding to our application is directly obtained from a weighted sum of the observation values at time $p$. In most cases, however, the difference between the bit-true and the infinite-precision simulations, namely the 'error-value' $E[p]$ [9] leads to more realistic and practical conclusions. This error function $E[p]$ has already been employed by Claesen *et al.*[9] and it can be expressed as (see Reference 9 for proof):

$$E[p] = \sum_j \sum_k E_j[t_k] h_j[p - t_k] \tag{6}$$

In this expression, the first summation index $j$ ranges over all scalers, and the second over the time-abscissae from 0 to $p$. The $h_j$ factor constitutes the time-domain values of the impulse response from the $j$th scaler up to the observed output node. If we adopt this objective function, the necessary equations for an instance of the ILP change but they can still be derived essentially from the set of equations as defined in (1)–(3).

It should also be mentioned that most of the objective functions which have been defined up to now, or others which can be defined by the users, can become both positive and negative. So in practice both the minimal and the maximum values should be determined to arrive at the worst-case situation. In the subsequent sections however, it will be assumed that the objective function for the optimization always has to be maximized (this can be simply obtained with a sign inversion).

Another remark concerns the presence of 'unfeasible' states. For the investigation of small-scale limit cycles, for instance, the evaluation should be stopped if an overflow condition occurs somewhere in the system. Indeed such a behaviour can give rise to large-scale oscillations which do not fall within the scope of the presently discussed application of zero-input limit cycles. So any initial state leading to an overflow condition is labelled as 'unfeasible' in the search region.

The ILP formulation as defined above is very flexible. It is, for instance, suitable to handle the description and investigation of both zero-input and constant-input limit cycles or even mixed cases. For this purpose, some of the inputs $U_i[t]$ can be put to 0, whereas others are retained at constant levels which are included as additional parameters in the optimization. Also, more general error bounds can be investigated by including the appropriate expressions for the external input-values $U_i[t]$ during some or all of the time-steps $t$. In this way, for instance, correlated 'noise' generated by (superimposed) sine waves can be investigated by adding the amplitude(s) of the sinusoids as a parameter and by precalculating the resulting input values for each time-step as a function of a set of particular sine frequencies. Restrictions on the time-domain characteristics of input signal sequences can be introduced too. These are useful when, for instance, a fixed relation exists between the excitations at two different nodes.

Moreover, other quantization or overflow characteristics can be modelled similarly to the value-truncation in (4) and (5). And if the objective function is modified, a scaling analysis can also be performed. So, within the frame of this very general description, a variety of distinct problems, as listed in subsection 2.1 and Reference 15, can be tackled. All of these are important in a practical design environment. The only restriction is the fact that the variables which occur in the objective function can be expressed in terms of the independent parameters (such as initial states and inputs) by means of linear expressions and that only constraints in the form of linear inequalities are present.

### *2.3 Motivation and implementation of the simulated annealing approach*

Several options are available to solve the ILP system in (1)–(3). In practice, as explained in the preceding subsection, the only freely assignable parameters in a digital filter realization normally result from the

initial states $X_n[0]$ together with the parameters in the expressions to derive the input excitations $U_i[t]$. So, in principle, an exhaustive search can be carried out by enumerating all sets to find the parameter(s) which are feasible and which maximize the objective function $E[p]$. This becomes rapidly impractical due to the exponential growth of the search space. For second-order systems where zero-input limit cycles are investigated, only the two internal states have to be taken into account. When the register states are represented with 16-bit values, already $(2^{16})^2$, or approximately 4 billion solutions have to be evaluated.

Unfortunately, general techniques to solve ILP systems (such as the Gomory cut-plane method[16]) are also prohibitively CPU-time consuming. This can be expected as the problem is NP-complete. Thus, severe restrictions are imposed on the feasible problem size when we require a rigorous, exact solution. Consequently, we have to look for more heuristically orientated solution strategies which do not guarantee to find the global optimum, but which arrive at a close estimate with a reasonably high probability and a 'less-than-exponential' increase (possibly polynomial) of the CPU-cost in terms of the problem size. In practice, an estimate within 10 per cent certainly suffices for this application.

To the best of our knowledge, there is *no efficient well-suited technique to identify or search for worst-case limit-cycles in arbitrary filter structures*. Therefore, only more generally applicable search strategies remain available based on, for instance, repetitive iteration improvement starting from a 'guided guess', which is described as 'local search strategy' in Chapter 19 of Reference 17. However, for the limit-cycle problem experiments have given evidence that the shape of the feasible region in the configuration space, i.e. the sets fulfilling the inequalities in (2.3), is not necessarily convex and sometimes even appears to be split up into several disconnected feasible subregions which are separated by parameter sets which lead to overflow. Furthermore, no clues whatsoever are available to the designer, in order to determine a seed for the gradient-based search which lies sufficiently close to the optimum. For this reason, we are likely to become trapped in a local optimum as we are only moving downhill and never uphill.[17] So, an extensive set of (randomly distributed) seeds has to be considered. An even in that case, nothing can be guaranteed concerning the quality of the obtained 'optimum'. Moreover, if the user is not satisfied with it, spending more CPU-time in terms of further iterations does not necessarily improve the results. In subsection 2.4 some experimental results to support these statements will be presented.

The *simulated annealing* (SA) method[1,2] constitutes a more *promising and efficient heuristic technique to obtain a 'near-optimal' solution* for (most practical) systems with more variables. This is primarily due to the fact that uphill moves are also accepted when certain conditions are fulfilled. Moreover, several theorems about the convergence behaviour of SA have been formulated.[18] In this way, even when the required assumptions are not always fully satisfied in practice, the theory presents some guidance concerning the general behaviour of the SA algorithm. Therefore, when more effort is spent in terms of CPU-time, usually the quality of the attained result will gradually improve. This permits the application of the following strategy: two runs with a different set of annealing parameters are started. The first set is chosen less 'stringent' (i.e. fewer objective function evaluations will be spent) than the second one. If both investigations arrive at roughly the same result within the user-supplied accuracy,[15] it can be concluded with a very high probability that the optimum is nearly reached. On the other hand, if these differ too much, a third run has to be scheduled with even more severe constraints for the convergence. To avoid this undesirable situation, it is important to enter a 'good' initial choice for the user-assignable parameters in the annealing schedule.

The ILP formulation of the previous subsection allows one to evaluate the necessary variables in the objective function expression in terms of the independent parameters in a very efficient way. As our improved simulated annealing routine still takes a lot of these evaluations (a few thousand at the least), each of these should indeed be restricted in CPU-time in order to keep the total CPU-consumption reasonable. Alternatively, a general simulator module such as BISICL[19] can perform the evaluation task in a more flexible way providing a larger number of primitive operators and quantization characteristics. As always in a trade-off, the cost measured by the CPU-consumption will be increased owing to the overhead involved to obtain the generality. Both approaches have been implemented and evaluated with our CAD tool SAILPLANE,[3,4] which employs the optimization kernel SAMURAI.

In the final version, we have eventually chosen the general simulation option because its flexibility will,

for instance, also allow the investigation of filters with weakly programmable or variable coefficients which would introduce intolerable non-linearities in the ILP equations of (1)–(3).

### 2.4. *Experimental results*

The SAILPLANE program has been evaluated for several practical industrial examples. It should be stressed that in each of these test vehicles, it has been *assumed* a priori *that the coefficients are already quantized* (see also section 3) in an optimal way. In order to take the actual finite-word-length effects into account, the *expansion of the multipliers in terms of primitive shift and add operations has to be accurately modelled in the final implementation*. This important aspect has been neglected in much previous work in this area,[6–8,13,14] probably because it has been assumed that (very expensive) floating point or full-scale parallel multipliers are available which do not require internal quantization. In practice, however, this area overhead will usually be avoided in an application-specific realization[12,20,21] where fixed coefficients can be implemented with a limited number of adders and shifters.

Although SAILPLANE does not impose any real restrictions on the topology, most filters implemented at present roughly belong to two distinct classes: cascade canonic structures and wave digital filters (WDFs).[10] In this subsection, we will primarily discuss the results which have been obtained for the zero-input limit-cycle investigation, as these disortions are the most widely studied in the literature,[6–10,13,14,22] and thus provide the best comparisons. In addition, the other effects will also be briefly mentioned.

For relatively small examples an exhaustive search (ES) has been carried out over the full or a part of the configuration space. Methods producing an upper bound on the limit cycle amplitude[6,7,13,14] or the complete error $E[p]$[9] (as defined in subsection 2.2), arrive at rather pessimistic results. Munson *et al.*[8] have investigated a restricted class of simple second-order sections. By means of an exhaustive search the maximum amplitude (zero-input) limit cycle is found. However, this technique remains exponential in terms of the problem size and it is not generally applicable. So, as the approach of Claesen *et al.*[9] constitutes the only completely general one we will compare most of our new results with it (the UB column in Table I).

The repeated iterative improvement (II) strategy with randomly selected seeds (see subsection 2.3) forms another reference to evaluate the quality of the investigation with SAILPLANE. In our experiments, the neighbourhood for the local search has been kept rather restricted, even though SAILPLANE also contains a mode to enlarge this 'move-range'.[15] In this approach, enough seeds have been generated for

Table I. Test-case results for the worst-case error value $E[p]$ (only the absolute value is indicated) between the high-level and bit-true evaluations of the mentioned filters. A full exhaustive search (ES), an upper bound (UB) technique,[9] an iterative improvement (II) strategy[17] and a new simulated annealing (SA) routine[2–4] are compared. In the first column, the size of the global search space for ES is also indicated, and in the last column the percentage savings for SA with respect to II are mentioned. The dashes indicate that no information is available owing to theoretical or CPU-time considerations. The parameters for the filters can be obtained in the references mentioned in the text or from the authors

| Example | ES (size) | UB | II | SA (percentage of iterations saved) |
|---|---|---|---|---|
| WDLF3 | 0·0773 ($2^{16}$) | 2·09 | glob | glob |
| VIEW-MR | — ($2^{168}$) | 205·6 | 30·45 | 35·35 (21) |
| VIEW-SR | — ($2^{90}$) | 21·7 | 12·4 | 12·7 (53) |
| SECORD | 1·02 ($2^{14}$) | 2·57 | 1·01 | 1·01 (84) |
| SECMUN | 16·734 ($2^{14}$) | 81·2 | 13·55 | glob (80) |
| KB123 | — ($2^{60}$) | 249·1 | 4·0006 | 4·0009 (30) |
| EQ8 | 2·416 ($2^{16}$) | 11·3 | glob | glob (52) |
| EQ8C | 0·4167 ($2^{16}$) | 9·3 | glob | 0·4155 (87) |
| EQV | — ($2^{39}$) | — | 509·675 | 511·13 (40) |
| EQVC | — ($2^{39}$) | — | 5·136 | 5·282 (81) |

each test case to arrive at a total of about 10,000 evaluations. This maximum has been specified in order to restrict the CPU-consumption.

The information for the respective test vehicles concerning the maximal error $E[p]$ between the theoretical double-precision and the integer bit-true calculations, for which only the absolute value is indicated, has been collected in Table I. In our experiments, the optimum obtained with simulated annealing (SA), almost never differs more than 1 per cent from the global one. However, for more complex systems the global optimal is impossible to obtain with ES or any other 'exact' method. Therefore, the results have been compared with those obtained for a set of extremely stringent annealing parameters, conforming to the strategy proposed at the end of subsection 2.3. Indeed, owing to the convergence properties of SA[18] it can be expected that when enough CPU-time is spent, a very close estimate for the optimum can be obtained in this way.

Instead of listing the CPU-times, which are very dependent on the filter complexity and on the computer system employed, we have found it more instructive to compare the simulated annealing with the iterative improvement based on the number of filter evaluations (iterations). In Table I, the percentage savings have been indicated for our simulated annealing routine compared to the maximum of 10,000 filter evaluations for the iterative improvement, except for EQV where a maximum of 25,000 has been applied both for II and SA. It should be noted that when the designer is not interested in the exact error but only in the integer limit-cycle magnitude, the CPU-consumption can be reduced even further.[15] Some examples for the required CPU-time will be presented further on in this subsection.

*2.4.1. Wave digital filters.* WDLF3 constitutes our first example. It is a simple wave digital lattice filter for which one branch consists of two cascaded two-port adaptors (Figure 4). In theory, magnitude-up truncation should be applied to avoid the occurrence of any limit cycles.[10] However, also for the simpler twos-complement value-truncation characteristic, it has been found with SAILPLANE that, in fact, no parasitic oscillations are present. The results obtained for $E[p]$ are shown in Table I. Although both SA and II arrive at the *global optimum* 0·077, the *iteration count* of the former remains *much lower* (only one third is required). It should be noted that the error remains less than one quantization unit, and this contribution is entirely due to the high-level evaluation which has not yet fully converged to 0 after a simulation period of 21 cycles.

VIEW-MR is the main module (12th order) of a multi-rate WDF used as a view-data filter with industrial specifications.[23] When simple twos-complement value truncation is applied, a limit cycle with period 16 oscillating between $-2$ and 5 quantization units has been discovered both with SA and II. But the maximal error amplitude after 150 time-cycles (namely 30·45; see Table I) is not worst-case for the II, even at the expense of more iterations, because SAILPLANE arrives at a larger error of 35·35. In the actual implementation,[23] magnitude-truncation is applied with slight modifications, as imposed for WDFs.[10] With SAILPLANE it has then been *verified that all limit-cycles are indeed suppressed.*

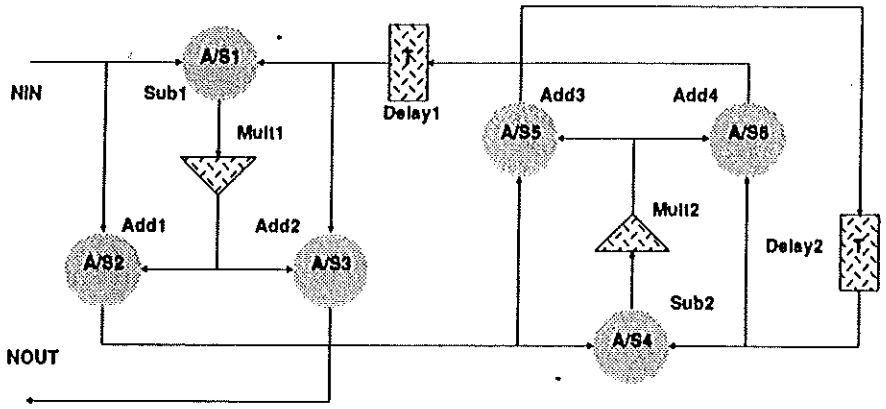



Figure 4. Block diagram of second-order branch in third-order wave digital lattice filter, where Mult1 = 1/2 and Mult2 = 1/8

The same view-data filter has also been redesigned with similar specifications, resulting in a 9th order signal-rate system with 3-port and 4-port adaptors. The structure is completely different but the same desirable results for the optimization with SAILPLANE have been be observed. The results for the WDF without the appropriate truncation characteristic, and thus not necessarily free of limit-cycles, are available in row VIEW-SR.

*2.4.2. Cascade canonic structures.* The circuit in the row labelled SECORD merely consists of a direct-form second-order section with two very simple scalar coefficients (powers of 2) (see Figure 2(a)). Neither SA nor II find the optimal solution, but the results are certainly close enough. Again, SAILPLANE requires many fewer evaluations.

Munson *et al.*[8] have made an extensive study for these second-order filter topologies by means of an (in principle) exhaustive method where the search range has been partly restricted based on theoretical considerations. For instance, a maximum amplitude limit cycle varying between $-13$ and $+13$ with an oscillation period of 34 cycles has been reported (see also Reference 24). It occurs when Scal1 = 1·111, Scal2 = 0·1111 (both in binary format) and when the minus sign for subtractor A/S2 (in Figure 2(a)) is located at the other input. For both coefficients, a sign-magnitude rounding was selected.[8,24] The required CPU-time a mounted to 20 s on a CYBER 175 host.[8] For this second order section, it can be verified in row SECMUN of table I that SAILPLANE detects a maximum error $E[p]$ of 16·73 q.u. (quantization units) for a simulation period of 50 cycles. The required CPU-time is approximately 710 s on a (much more modest) VAX780/VMS. The *same worst-case limit cycle as reported*[6,24] *is detected but, moreover, the worst-case global error is also supplied* in this case. Actually if the maximum integer amplitude has to be searched for, SAILPLANE takes many fewer iterations. In that case, about 150 s are more than sufficient to obtain a high probability for this global maximum.

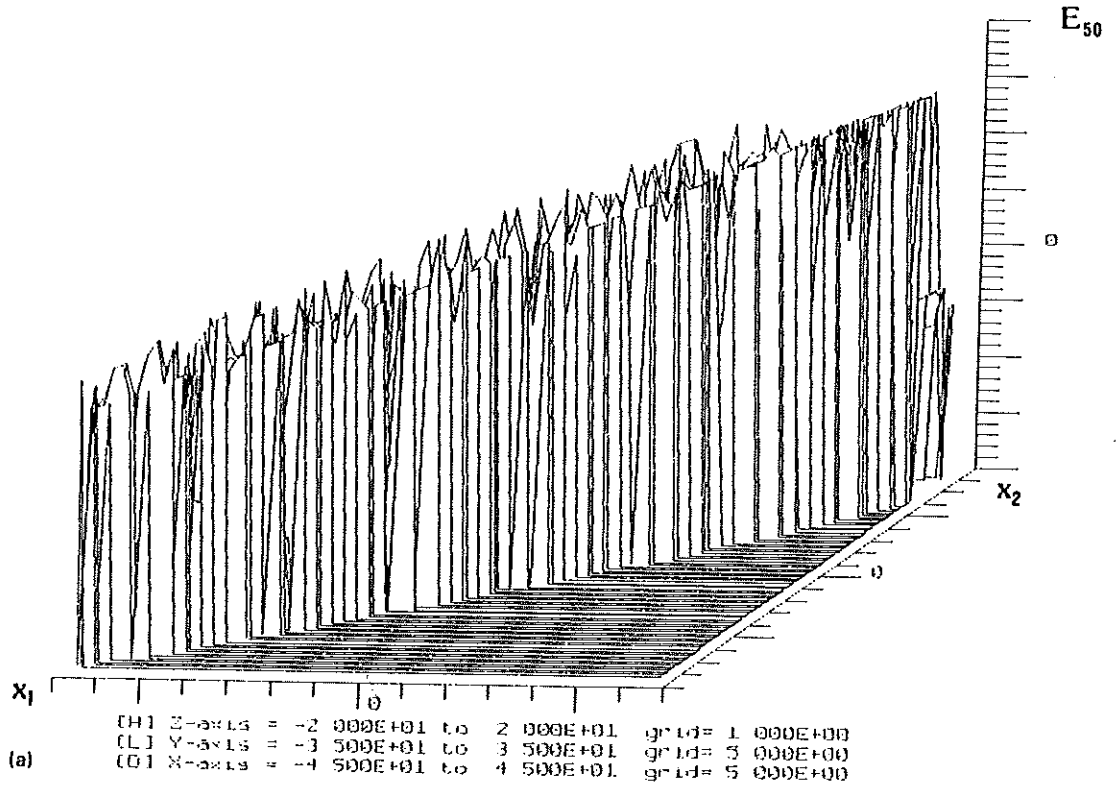
In Figure 5 the objective function has been plotted for the entire search space (as defined in subsection 2.2). The region with unfeasible initial states (leading to overflow for an internal signal word-length of 7 bits) has been represented as a 'flat' surface with value $-20$ in order to make the illustration more clear. It is clearly visible that the objective function $E[p]$ (with $p = 50$ cycles) changes in a very irregular way as a function of the initial states in the two registers. Hence, the task of the SA routine is not at all trivial. This is also demonstrated by the fact that the local search strategy with randomly selected seeds (the II entry in Table I) arrives at a very poor estimate (of only 13·55) for the maximum, even at the expense of five times as many filter evaluations.

Kieburtz[22] has presented an *experimental study* of a ten-order cascade canonic filter (five sections) with additional scalers. The results when only the first three sections are considered, are listed in the row labelled KB123. The predicted limit cycle of period $10^{11}$ has indeed been detected as the one with the largest amplitude. When a scaling factor of 2 is taken into account, both the minimum of $-8$ (see Table I) and the maximum of $-6$ listed in Figure 2 of Reference 22 are *confirmed*. For the entire filter with five sections, similarly a limit cycle of period 10 between 4 and $-8$ has been found. The experimental results in the original paper also indicate a 150-frame oscillation. But that one reaches only values between $-2$ and $-7$, so it will not be reported because the amplitude is not maximal.

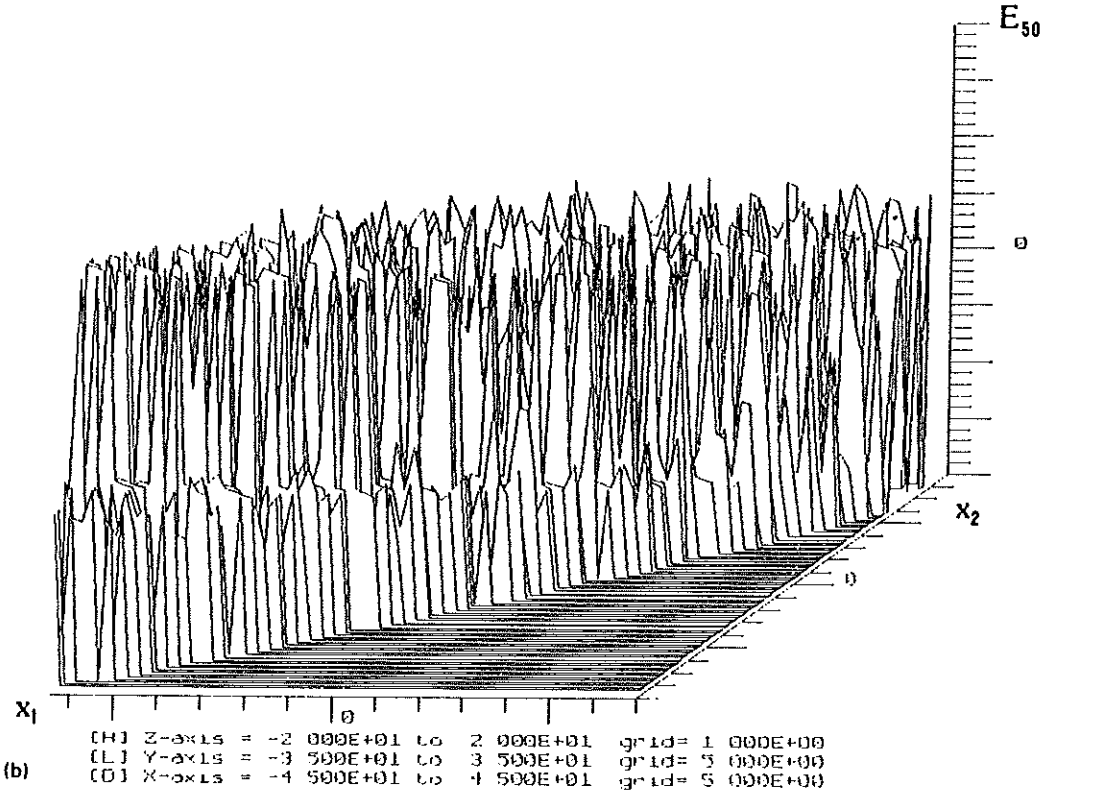
Our final example consists of a cascade of ten second-order sections modified with Agarwal–Burrus integrators (Figure 6), and contains three *weakly programmable coefficients* ScVar, ScBeta1 and ScBeta2. This system is to be used as the core of an audio-equalizer system[20] and has been integrated on chip.

EQ8 constitutes the eighth section separately with a fixed set of coefficient-values, and contains a limit cycle at the output with period 2, alternating between 0 and 2. A modified quantization, namely magnitude-up truncation instead of value truncation, has been suggested[20] to solve this problem, but it could not be ensured with a high certainty as only a few initial states had been simulated and checked. In the EQ8C row, it can be verified now that this approach is indeed successful for the eighth section.

Furthermore, as the coefficients are weakly programmable, this investigation should normally be performed for any set which can occur. The range of allowed coefficient values is usually bounded, and as already mentioned in subsection 2.2, a variable coefficient can be entered in SAILPLANE directly, *allowing the investigation of an entire class of filter sections (with a fixed topology) in one shot*. Actually,


E50
X2
X1
0
[H] Z-axis = -2 000E+01 to 2 000E+01 grid= 1 000E+00
[L] Y-axis = -3 500E+01 to 3 500E+01 grid= 5 000E+00
[O] X-axis = -4 500E+01 to 4 500E+01 grid= 5 000E+00

(a)


E50
X2
X1
0
[H] Z-axis = -2 000E+01 to 2 000E+01 grid= 1 000E+00
[L] Y-axis = -3 500E+01 to 3 500E+01 grid= 5 000E+00
[O] X-axis = -4 500E+01 to 4 500E+01 grid= 5 000E+00

(b)

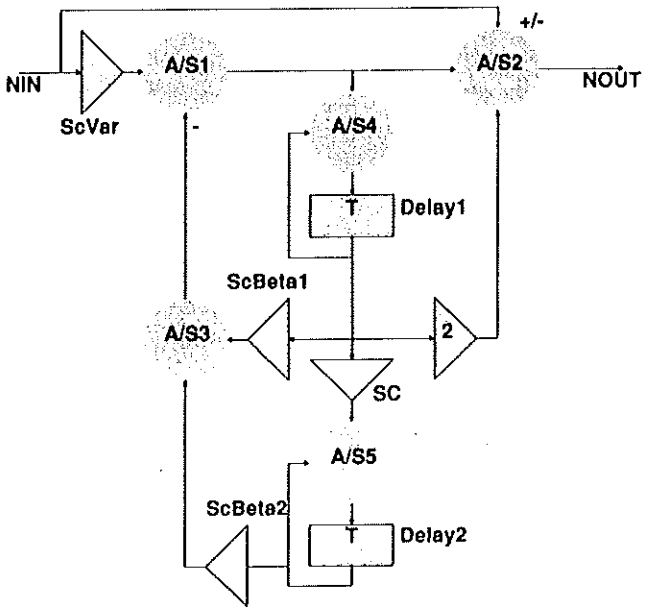



Figure 6. Block diagram of weakly programmable Agarwal–Burrus transformed second-order section as applied in an equalizer system which is part of an audio filter set. ScVar, ScBeta1 and ScBeta2 are variable nested multipliers; SC is a variable scaler

a number of alternative variable-multiplier realizations have been predefined in SAILPLANE,[15] such as a nested organization with binary or CSD-coded bits. Also, a variable scalar is provided. These primitives have been employed to generate an input description for the programmable equalizer which covers eight sections. For each of these sections all 24 sets of coefficient values[20] can be handled. The last two sections contain a coefficient ScBeta2 equal to 1 and do not exhibit any limit cycles at all, even with common value truncation. So they do not have to be taken into account. Hence, a *single SAILPLANE run suffices* to come up with the results in Table I for the equalizer version with and without the proposed change,[20] as listed under EQVC (section 5–8) and EQV. These figures confirm that the limit cycles present in the original scheme have been eliminated with the new quantization characteristic (see e.g. Figure 7). It has however been observed that when the coefficient range is a little bit extended, limit cycles can still occur for both realizations. Actually, an almost purely sinusoidal steady-state response with a reasonably large magnitude, oscillating between about $-400$ and $+400$ quantization units with a period of 50 cycles is illustrated in Figure 8. It is triggered from the initial state (10, $-365$) in the two internal registers, and with ScVar = 0, ScBeta1 = 1/64 and ScBeta2 = 1/16.

So the equalizer clearly demonstrates how SAILPLAIN can identify the subregions in the coefficient space which are free of limit cycles and those which are 'corrupted', in an accurate and very efficient way. This is in sharp contrast with more theoretical approaches for which the system in Figure 6 is already too complex.

*2.4.3. Evaluation and conclusions.* Except for EQ8 (with a minor difference compared with the II), the *SA always leads to better results* than the other methods, and with fewer iterations. Actually, the difference becomes more apparent for the two larger test cases, namely VIEW and KB123, which occur

Figure 5. Plot of objective function $E[p]$ (with $p = 50$ cycles) as a function of the set of initial states $X_1[0]$ and $X_2[0]$ in the two registers of a second order section with coefficients Scal1 = 1·111 and Scal2 = 0·1111.[8] Only a subset of the global search space has been illustrated. The vast region with unfeasible initial states (leading to overflow for an internal signal word-length of seven bits) has been represented as a 'flat' surface with value $-20$ in order to make the illustration more clear. Both the 'front' (a) and the 'back' (b) of the objective function shape are illustrated. The maximum (16·73) is located at the initial states $X_1[0] = 18$ ($Y$-axis) and $X_2[0] = 15$ ($X$-axis).

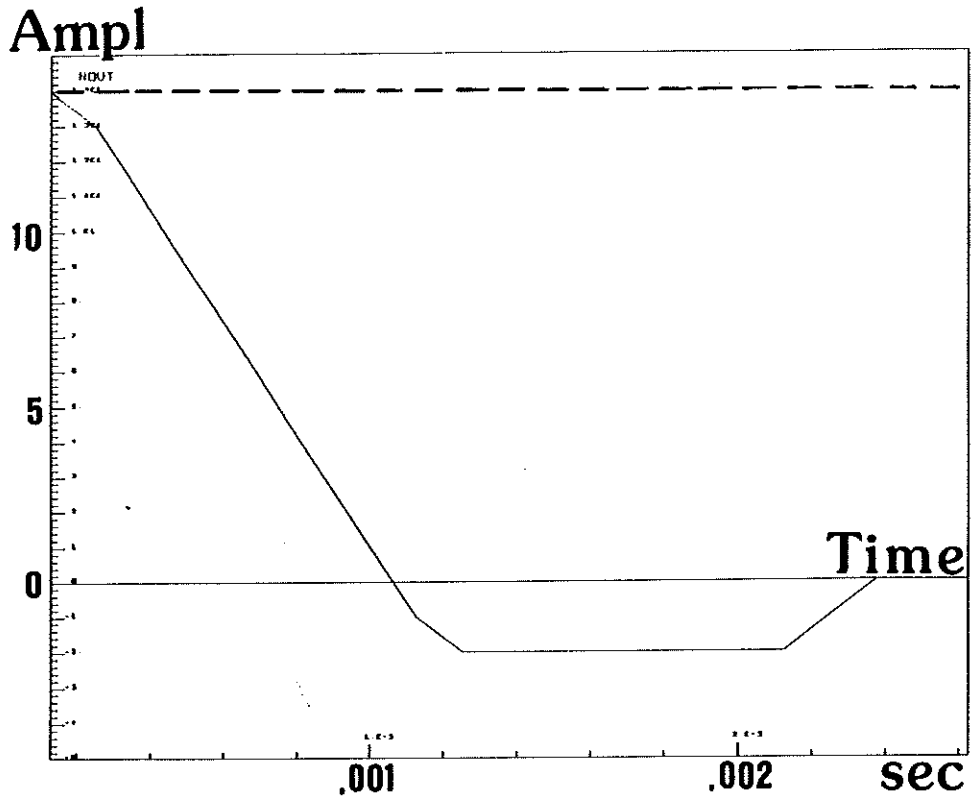



Figure 7. Constant offset of 14 q.u. (limit cycle with period 0) detected for a specific equalizer section as illustrated in Figure 6 (dashed line). If a modified quantization characteristic is applied, this limit cycle disappears again (solid line)

more frequently in practice. Furthermore, the actual CPU-time consumption will turn out to be even more in favour of SA, as a higher portion of the listed iterations lead to overflow conditions and, consequently, the simulation can be terminated long before the planned period $p$ is reached.

The actual CPU-time consumption usually lies between a few minutes (for WDLF3, SECMUN and SECORD) and a few hours (for VIEW, KB and EQV) on a VAX780. These figures remain reasonable, as this effort has to be performed only once in the design, namely when the final filter structure has been defined and when the finite-word-length effects have to be studied. So indeed, this investigation is not contained within a design loop which has to be iterated very frequently.

It is also important to note that *in the case of the repeated iterative improvement not much can be guaranteed about the quality of the obtained optimum* as, for instance, indicated by VIEW. So in general a vast number of iterations would have to be spent in order to be on the safe side. Owing to the convergence properties of simulated annealing,[18] this accuracy problem is much relieved. Actually, in the many practical examples which we have already investigated, the worst-case finite-word-length effect to be determined has always been confirmed with SAILPLANE. Moreover, within very reasonable iteration counts (usually a few thousand of them suffice) the obtained accuracy for the objective function has always been (much) better than 10 per cent.

Also the *comparison with methods resulting in an upper bound is very favourable for simulated annealing*, even though (much) more CPU time has to be spent for the latter. As shown in Table I, a lot of bits would have to be spent at the LSB side in order to take these finite-word-length effects into acount. For instance, in the first three sections of the Kieburtz filter (KB123), *eight bits would be needed* according to the upper bound technique, *whereas the actual worst-case limit-cycle affects only two bits*. The same situation occurs for the view-data filter (VIEW-MR) if magnitude truncation is applied. The upper bound,

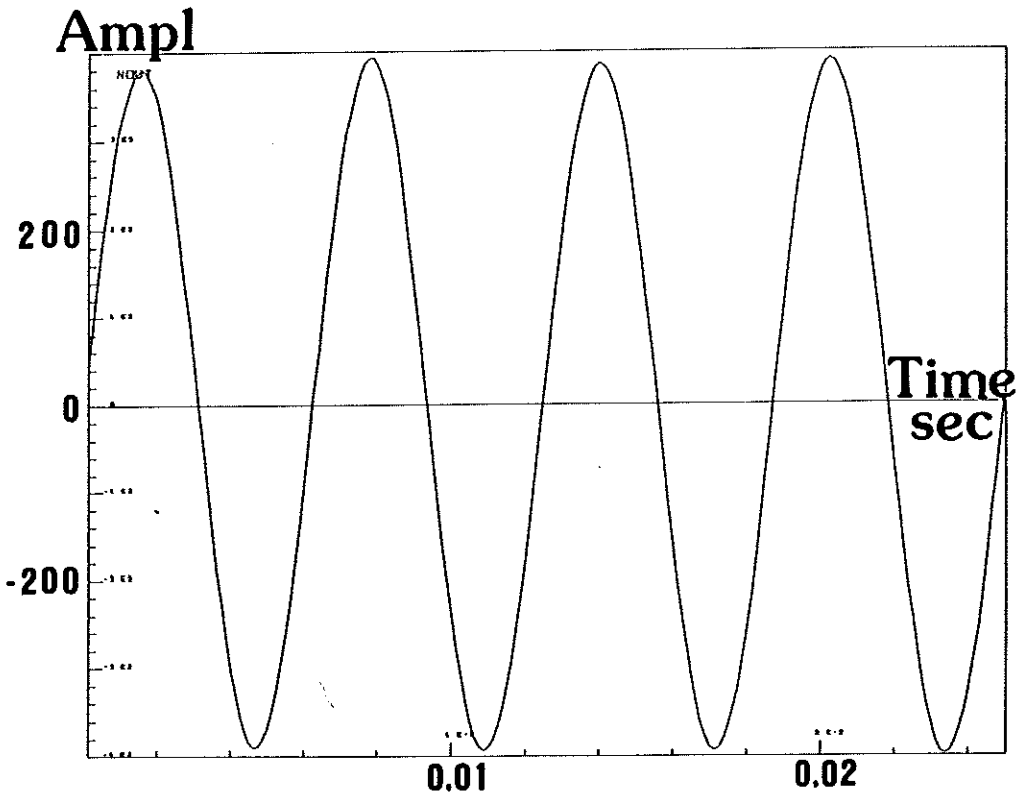



Figure 8. Zero-input limit cycle oscillating between about 400 and −400 q.u. (almost sinusoidal) which is triggered for a specific set of coefficients applied in the system of Figure 6

which is actually bounding the total error, remains very large even when the small-scale limit cycles are suppressed. Hence, a considerable saving in terms of area can be achieved in application-specific realizations (see subsection 2.1). It should also be noted that *for (weakly) programmable filters, no method exists at all to arrive at an upper bound.*

Finally, it should be emphasized that *all operators* implemented in SAILPLANE *behave in a bit-true manner.* Owing to the possibility of defining new elements in the bit-parallel simulator,[19] e.g. a nested multiplier with strictly binary or CSD-coded coefficients, variable multipliers based on these schemes and a variable scalar have already been provided.[15] Also a large set of different quantization and overflow characteristics is available.[19]

## 3. OTHER HARD SIMULATED-ANNEALING APPLICATIONS

In order to demonstrate the general applicability of our annealing routine SAMURAI,[2–4] some interesting results which have been obtained for other global optimization problems will be mentioned in this section. It has to be stressed that the approaches to modelling these applications are completely different from the ILP formulation for the finite-word-length effects.

### *3.1. The optimization of CSD-coefficients*

In an application-specific synthesis and implementation environment for digital filters,[12] several other hard optimization problems can be identified in addition to the finite-word-length investigation. Another important application of SAMURAI is related to the minimization of the number of non-zero bits in the quantized filter coefficients.

*3.1.1. Problem scope.* After the generation of an appropriate filter diagram with a synthesis tool, the *nominal set of coefficients has to be quantized while satisfying the frequency-domain specifications*, which are represented as upper and lower bounds. A novel method for computing the changes in the frequency response when the coefficient values are modified, has been presented by Jain *et al.*[25,26] This technique is based on a multi-parameter analysis, directly in the $z$-domain, of the filter flow graph and allows the evaluation of the feasibility of a particular quantization of the coefficient set in a very efficient and accurate way

In the CANDI program,[25,26] this approach has been applied to *minimize the number of non-zero bits of coefficients which are represented in the canonical-signed-digit (CSD) code.*[5] The optimization strategies included in this CAD tool are mainly heuristic and depend heavily on interactive user input in order to arrive at near-optimal solutions. Moreover, the definition of the goal function is too inflexible.

*3.1.2 CAUCASIAN: coefficient optimization with simulated annealing.* The annealing routine SAMURAI has been employed as an alternative for the heuristic optimization strategies available in the CANDI tool box.[26] Especially due to our modular approach, the *required effort to develop a new tool has been relatively small.* Indeed, in our general simulated annealing routine the parameters in the 'annealing schedule' have already been defined, namely the initial temperature, the temperature updating mechanism, the number of evaluations at each temperature, the stop criterion and so on.[2] Hence, for this new application only *three application-specific procedures* have to be added. Actually, this limited amount of effort to be spent also applies for coupling any other global optimization application with SAMURAI. This is again a very important advantage of using simulated annealing.

First of all, the *objective function to be optimized has to be evaluated* for an arbitrary state in the configuration space, i.e. a set of quantized CSD coefficients. In the most common case, simply the total number of CSD bits has to be counted. However, *if required, other optimization criteria can also be selected*,[27] such as the maximum word-length for the coefficient set. This flexibility is important for the designer and it constitutes a crucial advantage of the annealing approach that such changes are much more simple to include here than in the case when heuristic, tuned optimization techniques are applied (as in CANDI).

Next, the *feasibility of a certain state has to be verified*, i.e. whether all the constraints are fulfilled. In this case, it has to be checked whether a particular coefficient set leads to a frequency response within the requested frequency specifications (upper and lower bound). For this purpose, the CANDI procedures mentioned above have been employed. They perform this task in a very efficient way.

Finally, an appropriate and efficient *mechanism for making a move from the current to a new state in the search space* has to be specified. This choice is *extremely important* because it will *largely influence the effectiveness of the annealing.*

A move-set is needed for the generation of new states during the statistically controlled search.[2] The moves which have been selected in our approach are relatively simple, but they have proved to be very effective. Essentially, three types of move are considered, each with a different probability: non-zero CSD bits are omitted, shifted over one position, or inverted. For each of these classes, the change can be performed for a single coefficient or for a pair. Experiments have revealed that especially at the lower temperatures, when the annealing schedule is converging, the most effective improvements are due to the updating of two coefficients simultaneously but independently from each other.

In this way, the CAD tool CAUCASIAN (*c*oefficient *a*ssignment *u*nder *c*onstraints *a*pplying *si*mulated *an*nealing) has been constructed. In contrast to existing coefficient optimization strategies or techniques, this tool does not require a high-quality initial solution to be provided by the user. Actually, the optimization phase proceeds in a fully automated way (batch-orientated). In comparison with CANDI, the consumption of CPU-time is larger, but it remains quite reasonable as demonstrated with some examples further on.

*3.1.3. Experimental results obtained with CAUCASIAN.* Several practical filter systems[20] have been investigated, and the results with CAUCASIAN are tabulated in Table II. The comparison to the CANDI optima,[26] shows that *in most cases the total number of CSD bits has been further reduced.* Indeed, on

Table II. Results for the total number of CSD bits which have been obtained with CANDI and CAUCASIAN for a range of wave digital lattice filters (WDLF), for filters constructed out of cascaded direct-form second-order sections (Biquad); for a lossless discrete integrator (LDI) filter and for a transversal filter (FIR). The CPU-times for CAUCASIAN are mentioned in the last column and are evaluated for a VAX8600/VMS. These are not necessarily linearly proportional to the number of iterations and the filter complexity. Indeed, the ratio between the feasible and the less costly non-feasible states is also important

| Filter type | Order | Number of coefficient | CANDI result | CAUCASIAN result | CPU time in min (and number of iterations) |
|---|---|---|---|---|---|
| WDLF | 3 | 3 | | 3 | 0·5 (5·2K) |
| (PCM) | | | | 4(1 coefficient 0) | 0·2 (1·8K) |
| WDLF (audio) | 3 | 3 | 4 | 4 | 1·2 (6K) |
| Biquad (PCM) | 5 | 10 | 14 | 13 | |
| | | | | | 5·5 (15·6K) |
| WDLF (PCM) | 5 | 5 | 9 | 8 | 8·25 (20K) |
| Biquad (PCM) | 5 | 10 | 17 | 17 | 1 (1K) |
| WDLF (mob rad) | 6 | 6 | 12 | 7 | 5 (34K) |
| LDI (decim) | 9 | 9 | 17 | 14 | 58·3 (135K) |
| WDLF (decim) | 9 | 9 | 16 | 12 | 27·5 (194K) |
| WDLF (tmx) | 13 | 6 | 14 | 14 | 47 (190K) |
| WDLF (tmx) | 15 | 7 | 21 | 18 | 210 (1·3M) |
| WDLF (tmx) | 19 | 9 | 21 | 20 | 520 (2·5M) |
| WDLF (tmx) | 21 | 10 | 27 | 24 | 240 (580K) |
| FIR | 19 | 10 | — | 10 | 15 (45K) |

the average, the quantization step leads to less than two non-zero bits per coefficient, which saves many cycles in a shift-add based signal-processor realization, or a lot of silicon area when a chip implementation is preferred. This observation again demonstrates the power and generality of our annealing schedule.

For some filter topologies, such as for the wave digital lattice filters[10] (WDLF), a lot of hardware can be saved if one of the coefficients becomes zero, even at the expense of an increase in the total number of non-zero bits. Therefore, an option has been included in CAUCASIAN which allows stressing the occurrence of zero coefficients. This has resulted in a four non-zero bit solution with only two coefficients for the third order WDLF in the first row, which represents a low-pass PCM filter. So in that case the flow graph will contain only two adaptors instead of three. This mode has also proven to be very effective for the FIR example where four coefficients are eliminated.

It should be noted that the fact that the *filter topology can be selected in an arbitrary way* is due to the coupling with the general purpose evaluation tool *DIGEST*.[11] Moreover, it should also be stressed that the *efficiency of the CANDI procedures* for computing the changes in the frequency response, are essential for the annealing strategy, as many 'moves' have to be evaluated to obtained an accurate, near-optimal final result.

Recently,[28] a coefficient optimization tool for transversal filters, based on simulated annealing, has been reported. However, their annealing procedure appears to be less tuned towards minimizing the CPU time. Moreover, it should be noted that reasonably high order FIR systems (see e.g. Table II) can be handled in a very efficient way with SAILPLANE too, resulting in many zero coefficients (see above).

### 3.2. *The solution of large travelling salesman problems*

Travelling salesman problems (TSPs) exhibit an exponentially rising complexity as a function of the number of cities, and a rigorous proof for the NP-completeness has been supplied.[17] Therefore, methods for solving large TSPs have to rely on approximate, heuristic strategies. Even then, the reported results are usually restricted to a dimension of roughly a few hundred. Above that size, only methods leading to a very coarse approximation for the global optimum are available, because weeks of CPU time on

mainframes would have to be spent to obtain a better estimate. From the start however[1] annealing has appeared to provide reasonably good results for TSPs up to several hundred cities.

As reported,[2,4] also the SAMURAI module leads to *very satisfactory results* even for the *solution of very large TSP instances (up to a few thousand cities), especially if these are distributed in a clustered way.* The reason for this interesting property is explained in more detail in Reference 2.

In the context of VLSI design, solving TSPs is in itself not a very 'practical' task. But from the viewpoint of simulated annealing, this application is related to finding the optimal placement and routing of cells organized in a linear array,[1,29] which does indeed constitute a very important practical objective.

For instance, at our laboratory a set of silicon compilers is being developed which covers the design from a high-level functional specification up to a number of target architectural styles.[21,30] The CATHEDRAL-III target architecture consists of a set of bit-sliced co-operating data-paths.[31] The linear placement task for the functional building blocks in the data path during the lower level stages, which is related to the TSP problem, has proved to be an excellent candidate for applying our annealing routine.[32] Furthermore, the same applies for the two-dimensional placement of basic layout primitives in an analogue silicon compiler which is currently being implemented.

## 4. CONCLUSIONS

In this paper, an improved simulated annealing schedule[2-4] which has been implemented in the independent optimization module SAMURAI, is applied to the investigation of a broad range of finite word-length effects in digital filters.[4] *SAMURAI allows one to obtain sufficiently accurate estimates,* within between 1 and 10 per cent, dependent on the CPU-time spent, *for the global optima in hard (even NP-complete) optimization problems.*

Existing limit-cycle investigation programs for digital filters only generate upper bounds, and no dedicated heuristic techniques are available in the literature to arrive at a more accurate characterization. Therefore, the program SAILPLANE has been developed (Section 2) which is based on the simulated annealing routine SAMURAI. With this CAD tool, an exact bound can be found with a very high probability. As a result, *several bits can be saved on the required signal word length,* when compared to other methods (based on upper bounds) available in the literature (see subsection 2.4.3). Consequently a substantial amount of final chip area can be saved at the expense of a still reasonable amount of CPU-time: even for higher filter degrees (up to 10), a few hours on a VAX780/VMS suffice to obtain a solution which is in practice near-optimal (within a few per cent) with a very high probability (see subsection 2.4.3). The quality of these results is better than the ones arrived at with a repeated iterative improvement method starting from a set of randomly chosen seeds (see subsection 2.4 and Reference 3).

Moreover, SAILPLANE *handles arbitrary digital filters, even those with weakly programmable or variable coefficients.* This feature allows the characterization of an entire filter class in one shot, with specific bounds applied for the respective coefficient ranges.

The power and the general scope of SAMURAI have been further substantiated with results obtained for the quantization of fixed coefficients in digital filters by reducing the number of non-zero CSD bits which have to be implemented. For this purpose the CAD tool CAUCASIAN has been developed (section 3).

Our simulated annealing routine has also been evaluated for solving large travelling salesman problems. The latter have led to the observation that clustered TSPs, which exhibit an inherent hierarchical partitioning, are more suited for SAMURAI than the non-clustered variety. So although simulated annealing is of use for solving arbitrary global optimization problems, we believe that this *clustering property is in general highly desired for obtaining acceptable convergence rates towards near-optimal solutions for annealing procedures.*[2]

SAMURAI, SAILPLANE and CAUCASIAN are part of an integrated CAD tool-box for the silicon compilation of DSP applications[12,33] which has been implemented at IMEC as a research topic in the ESPRIT project no. 97.

ACKNOWLEDGEMENTS

We gratefully acknowledge S. Note, K. Mertens and L. Claesen for the use of the BISICL routines, and R. Jain for the use of the CANDI procedures. We also wish to thank all our colleagues in the project for the supply of practical filter examples and valuable Pascal utilities, and for the stimulating discussions. Finally, we are grateful for the very helpful suggestions of some anonymous reviewers for improving the readability of the text.

Joint research sponsored by the ESPRIT97 project of the EC.

REFERENCES

1. S. Kirkpatrick, C. D. Gelatt Jr. and M. Vecchi, 'Optimization by simulated annealing', *Science*, 220, 671–680 (1983).
2. F. Catthoor, J. Vandewalle and H. De Man, 'SAMURAI: a general and efficient simulated-annealing schedule, with fully adaptive annealing parameters', *Integration, the VLSI Journal*, 6, 147–178 (1988).
3. F. Catthoor, J. Vandewalle and H. De Man, 'SAILPLANE: a CAD-tool for the analysis of limit-cycle behaviour with the aid of simulated annealing', *Proc. IEEE Int. Conf. on Computer Design*, Port Chester, NY, October 1985, pp. 244–247.
4. F. Catthoor, J. Vandewalle and H. De Man, 'Investigation of finite word-length effects in arbitrary digital filters using simulated annealing', *Proc. IEEE Int. Symp. on Circuits and Systems*, San Jose, May 1986, pp. 1296–1297.
5. G. W. Reitwiesner, 'Binary arithmetic', in *Advances in computers, vol. 1*, Academic Press, 1966, pp. 231–308.
6. S. Parker and S. Hess, 'Limit-cycle oscillations in digital filters', *IEEE Trans. Circuit Theory*, CT-18, 687–697 (1971).
7. J. Long and T. Trick, 'An absolute bound on limit cycles due to roundoff errors in digital filters', *IEEE Trans. Audio Electroacoust.*, AU-21, 27–30 (1973).
8. D. Munson, J. Strickland and T. Walker, 'Maximum amplitude zero-input limit cycles in digital filters', *IEEE Trans. Circuits and Systems*, 31, 266–274 (1984).
9. L. Claesen, J. Vandewalle and H. De Man, 'General bounds on parasitic oscillations in arbitrary digital filters and their application in CAD', *Proc. IEEE Int. Symp. on Circuits and Systems*, Montreal, May 1984, pp. 747–750.
10. A. Fettweis and K. Meerkotter, 'Suppression of parasitic oscillations in wave digital filters', *IEEE Trans. Circuits and Systems*, 22, 239–246 (1975).
11. L. Claesen, J. Vandewalle and H. De Man, 'DIGEST: a digital filter evaluation and simulation tool for MOS-VLSI filter implementations', *IEEE J. Solid-state Circ.*, SC-19, 414–424 (1984).
12. H. De Man, J. Vandewalle *et al.*, 'A unified box for CAD tools for the design of dedication signal processing chips', *Proc. IEEE Int. Conf. on Computer Design*, Port Chester, NY, October 1984, pp. 838–844.
13. I. W. Sandberg and J. F. Kaiser, 'A bound on limit cycles in fixed-point implementations of digital filters', *IEEE Trans. Audio Electroacoust.*, AU-20, 110–112, (1972).
14. S Yakowitz and S. Parker, 'Computation of bounds for digital filters quantization errors', *IEEE Trans. Circuit Theory*, CT-20, 391–396 (1973).
15. F. Catthoor, J. Vandewalle and H. De Man, *SAILPLANE V1.1 Reference and User Manual*, IMEC, Heverlee, June 1986.
16. *NAG Fortran-library User Manual*, available from the Numerical Algorithms Group, Oxford, U.K.
17. C. H Papadimitriou and K. Steiglitz, *Combinatorial Optimization*, Prentice Hall Inc., 1982, pp. 366–371.
18. F. Romeo, A. Sangiovanni-Vincentelli and C. Sechen, 'Research on simulated annealing at Berkeley', *Proc. IEEE Int. Conf. on Computer Design*, Port Chester, NY, October 1984, pp. 652–657.
19. L. Claesen, S. Note, K. Mertens and H. De Man, 'Flexible and efficient bitparallel level simulation of hardware implemented digital signal processing algorithms', *Proc. IEEE Int. Symp. on Circuits and Systems*, San Jose, May 1986, pp. 871–874.
20. J. Vanginderdeuren, H. De Man, B. De Loore and G. Van den Audenaerde, 'Application specific integrated filter for HIFI digital audio signal processing', *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Tokyo, April 1986, pp. 1537–1540.
21. J. Rabaey, H. De Man, J. Vanhoof, G. Goossens and F. Catthoor, 'CATHEDRAL II: a synthesis system for multi-processor DSP systems', in D. Gajski (ed.), *Silicon Compilation*, 1988, pp. 311–360.
22. R. B. Kieburtz, 'An experimental study of roundoff effects in a tenth order recursive digital filter', *IEEE Trans. Comm.*, COM-21, 757–763 (1973).
23. L. Matterne, 'Bit-serial implementation of a view-data filter', *Int. ESPRIT97 Report*, Philips NATLAB, February 1985.
24. V. B. Lawrence and K. V. Mina, 'A new and interesting class of limit cycles in recursive digital filters', *Bell. Syst. Tech. J.*, 58, 379–408 (1979).
25. R. Jain, J. Vandewalle and H. De Man, 'Efficient and accurate multiparameter analysis of linear digital filters using a multivariable feedback representation', *IEEE Trans. on Circuits and Systems*, CAS-32, 225–235 (1985).
26. R. Jain, 'Computer-aided discrete coefficient optimisation for custom integrated digital filters', *Ph.D. Thesis*, Katholieke Universiteit Leuven, 1985.
27. F. Catthoor, J. Vandewalle and H. De. Man, *CAUCASIAN V1.0 Reference and User Manual*, IMEC, Heverlee, August 1986.
28. E. Diethorn and D. C. Munson, 'Finite word-length FIR digital filter design using simulated annealing', *Proc. IEEE Int. Symp. on Circuits and Systems*, San Jose, CA, May 1986, pp. 217–220.
29. E. Aarts and P. van Laarhoven, 'Statistical cooling: a unified approach to combinatorial optimisation problems', *Philips Journal of Research*, 40, (June), 193–226 (1985).
30. H. De Man, J. Rabaey, P. Six and L. Claesen, 'Cathedral II: a silicon compiler for digital signal processing', *IEEE Design and Test Magazine*, December 1986, pp. 13–25.

31. F. Catthoor and H. De Man, 'Customized architectural methodologies for high-speed image and video processing', *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, New York, April 1988, pp. 1985–1988.
32. S. Note, J. Van Meerbergen, F. Catthoor and H. De Man, 'Automated synthesis of a high-speed CORDIC algorithm with the cathedral-III compilation system', *Proc. IEEE Int. Symp. on Circuits and Systems*, Helsinki, Finland, June 1988.
33. L. Claesen, F. Catthoor, H. De Man, J. Vandewalle, S. Note and K. Mertens, 'A CAD environment for the thorough analysis, simulation and characterisation of VLSI implementable DSP systems', *Proc. IEEE Int. Conf. on Computer Design*, Port Chester, NY, October 1986, pp. 72–75.