Applications of interval analysis for circuit design

Leenaerts, D.M.W.

Published in:
IEEE Transactions on Circuits and Systems

DOI:
10.1109/31.55038

Published: 01/01/1990

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author’s version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 17. Dec. 2018
conversions. Output conversions for more than eight moduli can be handled with more than one chip and external binary adders. The same technique may be applied to handle output conversion when a larger output resolution is desired. Two (or more) chips can be used to form partial sums, which are then added off-chip.

The architecture can be slightly modified if one wishes to do QRNS processing, although the details of the modification are beyond the scope of the present discussion. The modifications are minimal (and are necessary only to input conversion) but they do increase the I/O burden on the chip. The converter without this modification can be used with QRNS if one is willing to provide a modulo multiply-and-accumulate for each RNS channel, and if one is willing to accommodate the necessary input connections on the RNS processors performing the RNS-to-QRNS conversion.

IV. CONCLUSION

A simple systolic architecture has been developed which permits both input and output conversion to be performed using the same hardware. The architecture is extremely powerful, as both the number of moduli and their values can be chosen arbitrarily (up to the capacity of the hardware design) and programmed into the converter. Another advantage of the design is that it is not a bottleneck for the RNS system, since its throughput is the same as the individual RNS channels. A chip designed in 1.25μm CMOS allows up to eight 6-bit moduli in the RNS. This design should provide an off-the-shelf solution for most RNS conversion requirements.

REFERENCES


APPLICATION OF INTERVAL ANALYSIS FOR CIRCUIT DESIGN

D. M. W. LEENAERTS

Abstract—In top-down circuit design, a principle task is to partition and map design constraints on a normal operating range of a collection of sub-blocks. This problem is propagated through each hierarchical level until the solutions for all levels are found. This top-down parameter assignment and instantiation of sub-blocks may eventually break down at some level due to an unrealizable circuit. Then the process has to be restarted a number of times before a realizable partition can be produced. In this paper, an application of interval analysis in the design environment is presented to assure in advance that this process will always yield a solution. The presented methodology and corresponding algorithm can be used in such hierarchical design strategies. At each hierarchical level, the solution space, if nonempty, is valid for all lower levels and is in agreement with decisions taken earlier in the hierarchy.

Application of Interval Analysis for Circuit Design

D. M. W. Leenaerts

Abstract—In top-down circuit design, a principle task is to partition and map design constraints on a normal operating range of a collection of sub-blocks. This problem is propagated through each hierarchical level until the solutions for all levels are found. This top-down parameter assignment and instantiation of sub-blocks may eventually break down at some level due to an unrealizable circuit. Then the process has to be restarted a number of times before a realizable partition can be produced. In this paper, an application of interval analysis in the design environment is presented to assure in advance that this process will always yield a solution. The presented methodology and corresponding algorithm can be used in such hierarchical design strategies. At each hierarchical level, the solution space, if nonempty, is valid for all lower levels and is in agreement with decisions taken earlier in the hierarchy.

I. INTRODUCTION

In complex engineering design tasks, such as analog IC design, the circuit can be decomposed into smaller sub-blocks, which are considered to behave almost independently of each other. Each sub-block can be described by a set of underdetermined equations in some of the circuit parameters and circuit voltages and currents. Due to design constraints for normal circuit operation, some or all of these unknowns may only be chosen within specific domains. The problem is then to solve the equations of each sub-block in such a way that a solution exists when the sub-blocks are interconnected, leaving a maximum amount of freedom for the set of design parameters. A methodology to find such a solution is to use interval arithmetic [1]–[6].

For each parameter of a sub-block, a default and an actual domain—i.e., an interval—is defined. The default domain is based on designer’s experience or design constraints. One can, for instance, think of minimum and maximum dimension of transistors in a certain technology or minimum and maximum allowable power dissipation of a sub-block, etc. A valid parameter is a value within the range of the actual domain. A valid parameter may be determined by a partitioning of the design constraints together with a partitioning of the circuit into smaller blocks at each hierarchical level.

Now, at each level the description of the domains of the parameters must be used to find the solution space for matching the design constraints with the chosen sub-blocks. The solution space actually yields new parameter domains, i.e., the new actual domains for each parameter. The advantage of this technique is that the solution space at each level does not conflict with the solution space at higher levels. Decisions made earlier at higher levels remain valid. The design process goes hierarchical downwards until the solution space for the lowest level, the basic building blocks, is found.

Then, bottom-up, the solution space at higher levels is assembled using the description of the solution at the lowest levels. At the top design level, actual values can be chosen in the produced solution space, automatically resulting in valid parameter values for the lower blocks in agreement with the design constraints at all levels of the hierarchy.

A design problem for which interval techniques can be used is the design of an op-amp. To simplify the problem consider only the gain of a three state op-amp as a design constraint. At the highest design level the gain could be partitioned over these stages in several ways. One level lower, there is a possibility that the decision made earlier is unrealizable. For instance, to design the first stage for the given gain may be impossible, whereas a higher gain in the design of the last stage could be easy. So, at the top level a wrong partition of the design constraint has taken place. To circumvent this problem, consider the interval in which the gain may exist. Now, at the top level, try to find the solution space of partitioning the gain over the three stages, given the domain of the amplification used by each stage. If a solution exists, one level lower a maximum of freedom for the design parameters can be used to design the stages, in agreement with the top level. Above, the interval methodology is used to find the solution space using a decomposition (top-down) strategy with the information for partitioning collected by a bottom-up construction.

At each hierarchical level, using the description of the solution space of the level above and the model description of this level, a set of equations, with domains for the variables, can be
formulated. So, all levels can use the same strategy to solve this set of equations, described in an algorithm.

Here a methodology will be presented to find the parametric solution of a set of linear equations with bounded variables. In Section II, an explanation of the solution strategy will be given. In Section III, some examples to demonstrate the application of the algorithm in circuit design are given. Finally, some conclusions are drawn in Section IV.

II. Theory
To use interval techniques, model descriptions for the blocks have to be defined at each hierarchical level, in which the circuit can be partitioned.

In the case of linear sub-blocks, the problem is to find all the solutions \( \mathbf{x} \) of \( n \) equations with \( m \) variables \( (m > n) \),

\[
\mathbf{A}\mathbf{x} = \mathbf{b} \quad \text{with } \mathbf{x} \in \mathbb{S}_x
\]

where \( \mathbb{S}_x \) represents the domains for all elements of \( \mathbf{x} \). The solution for this problem can be derived from an algorithm from Tschernikow [7], who presented an algorithm to find the solution space of a set of \( k \) linear equations of \( p \) variables,

\[
\mathbf{C}\mathbf{u} = \mathbf{0} \quad \text{with } \forall_j u_j \geq 0.
\]

The solution space describes all non-negative solutions of (2). Because the presented interval algorithm is based on the Tschernikow technique, a detailed explanation to solve (2) is outlined below. The algorithm starts to define a start tableau for (2)

\[
T^1 = (T_1^1 | T_2^1) = \begin{bmatrix}
1 & c_{11} & \cdots & c_{1k} \\
0 & \vdots & \ddots & \vdots \\
& c_{1p} & \cdots & c_{kp}
\end{bmatrix}
\]

where \( T_2^1 \) is a unit matrix and \( T_1^1 \) is composed by placing a row of (2) as a column in (3). For every row \( i \) \((i = 1, \ldots, p)\), define \( S(i) \) as the collection of columns of \( T_1^1 \) with zeros in row \( i \). Define \( S(i_1,i_2) \) for every combination \((i_1,i_2)\) \((i_1, i_2 = 1, \ldots, p)\) as the collection of columns of \( T_1^1 \) with zeros in both \( i_1 \) and \( i_2 \). A column of \( T_1^1 \), say column \( j \), is chosen at random with at least one nonzero element. From the collection \( S(i_1,i_2) \), only the subset with opposite sign in column \( j \), called \( S(i_1,i_2,j) \), is important. A new table, \( T^2 \), can now be composed by first placing the rows from \( T^1 \) with a zero in column \( j \) into \( T^2 \). Next, search for the pairs \((i_1,i_2)\), for which \( S(i_1,i_2,j) \neq \emptyset \), and place any linear combination of rows \( i_1 \) and \( i_2 \) such that a zero in column \( j \) is created in \( T^2 \).

In the same way, a new table \( T^t \) can be composed from table \( T^{t-1} \).

This process ends when there are one or more strict positive or strict negative columns in \( T^t \) (in which case there is no solution) or there are only zeros in \( T^t \). In the latter case the table is given by

\[
T^{end} = (T_1^{end} | T_2^{end}) = \begin{bmatrix}
b_{11} & \cdots & b_{1p} \\
\vdots & \ddots & \vdots \\
b_{1t} & \cdots & b_{tp}
\end{bmatrix} 0
\]

with the non-negative solution

\[
\mathbf{u} = \sum_{i=1}^t p_i \mathbf{b}_i
\]

where \( \mathbf{b}_i = (b_{i1}, \ldots, b_{ip}) \) \((i = 1, \ldots, t)\), and \( p_i \) is a non-negative parameter. The set of \( (b_{11}, \ldots, b_{1p})^T \) describes the corners of the convex solution space.

The procedure, written in pseudopascal, is given below

To use the algorithm of Tschernikow, (1) must be transformed into an equivalent representation of (2). In (1), only the parameter \( x_i \), which is not in agreement with \( x_i \geq 0 \), according to \( u_i \geq 0 \) in (2), must be redefined. When \( x_i \) is negative, only the substitution \( u_i = -x_i \) is necessary.

However, if \( x_i \) is bounded, then use a substitution variable, say \( u_i \), such that \( u_i \) is bounded between zero and a positive value, \( u_i \in [0,U_i] \). To get the restriction \( u_i \geq 0 \), use a new variable to describe the bound of \( u_i \) in an extra equation,

\[
u_i + u_i - U_i = 0 \quad \text{with } \forall i, u_i \geq 0.
\]

Define a slack-variable, say \( u_i \) \((u_i = 1)\), and multiply it with the constant \( U_i \) in (6). This gives the linear equation

\[
u_i + u_i - U_i u_i = 0 \quad \text{with } \forall i, u_i \geq 0.
\]

The total algorithm to solve problems according to (1) can now be defined. First transform the problem into a problem like (2), using the substitution in (6) and (7). Then solve the set of equations using Tschernikow and redescribe the solution space in the variables in which the problem was defined.

To demonstrate the technique to solve a given problem like (1), define

\[
x_1 - x_2 + 2x_3 + x_4 = -4
\]

\[
-x_1 - 2x_3 + x_5 = -10
\]

\[
x_1 \in [0,2], x_2 \in [1,4], x_3 \in [-3,-1], x_4 \in [-8,5].
\]

To reformulate (8) to a form equal to (2), first define

\[
\begin{align*}
u_1 &= x_1 \\
u_2 &= x_2 - 1 \\
u_3 &= x_3 - 1 \\
u_4 &= x_4 + 8
\end{align*}
\]

to get lower bounds for \( u_1, u_2, u_3, \) and \( u_4 \) equal to zero,

\[
\begin{align*}
u_1 - u_2 - 2u_3 + u_4 - 7 &= 0 \\
-2u_1 - u_2 - u_3 - u_4 + 16 &= 0
\end{align*}
\]

\[
u_1 \in [0,2], u_2 \in [0,3], u_3 \in [0,2], u_4 \in [0,13].
\]

Now, to get the restriction \( \forall_i u_i \geq 0 \), use new variables to describe the upper bounds of \( u_1, u_2, u_3, \) and \( u_4 \) in some extra equations. For \( u_3 \) this equation is

\[
u_1 + u_3 = 2 = 0 \quad \text{with } \forall_i u_i \geq 0.
\]
Define a slack-variable, say \( u_9 \), with \( u_9 = 1 \), and multiply it with all the constants in (10) and (11). This gives the set of equations

\[
\begin{align*}
7u_9 &= 0 \\
16u_9 &= 0 \\
-2u_9 &= 0 \\
-3u_9 &= 0 \\
-2u_9 &= 0 \\
-13u_9 &= 0
\end{align*}
\]

with \( v, u_i \geq 0 \). This set of equations is in agreement with the Tschernikow set and can be solved.

Tableau \( T^1 \), according to (3), is now described by

\[
\begin{align*}
1 & 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 -2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
III. APPLICATIONS OF THE INTERVAL ALGORITHM

To demonstrate that the presented technique can be used to solve different design problems, two applications are given.

In the first example, an impedance \( Z = Z_1 + Z_2 + Z_3 \) is considered, representing a series connection of three impedances

\[
Z_k = \frac{x_k + s}{(k + s)(3 + k + s)}, \quad k \in \{1, 2, 3\}. \tag{20}
\]

Each of the impedances \( Z_k \) is composed of resistors and capacitors only (see Fig. 1). From circuit theory it is known that for non-negative \( C_k \) and \( R_k \), it is required that \( k \leq x_k \leq 3 + k \).

The goal is to find solutions for \( x_k \) such that

\[
\arg\{Z(s)\}_{Z_1, Z_2} = -45^\circ \quad \text{and} \quad \arg\{Z(s)\}_{Z_1, Z_2} = -60^\circ. \tag{21}
\]

This results in the following set of equations:

\[
0.2018x_1 + 0.0696x_2 + 0.0155x_3 = 1
\]

\[
0.1330x_1 + 0.0971x_2 + 0.0636x_3 = 1
\]

\[
\text{with } 1 \leq x_1 \leq 4
\]

\[
2 \leq x_2 \leq 5
\]

\[
3 \leq x_3 \leq 6. \tag{22}
\]

Calculation of the solution space, using the above outlined procedure, yields

\[
\begin{bmatrix}
    x_1 \\
    x_2 \\
    x_3
\end{bmatrix} =
\begin{bmatrix}
    3.058 \\
    3.528 \\
    3.922
\end{bmatrix} +
\begin{bmatrix}
    3.992 \\
    2 \\
    4.468
\end{bmatrix} p_1
\]

\[
\text{with } p_1 + p_2 = 1 \text{ and } \forall p_i \geq 0. \tag{23}
\]

According to (23), several configurations are possible. Choosing \( p_1 = p_2 = 0.5 \) gives the configuration of Fig. 2(a), and \( p_1 = 1, p_2 = 0 \) leads to the realization of Fig. 2(b).

The second application is the design of a two-stage op-amp for dc biasing. To calculate the dc behavior linear equations appear to be sufficiently accurate.

Using a bottom-up strategy, the design process is to decompose the op-amp in a first and a second stage, to design them independently, and to connect the stages together to find an overall solution for the dc operation point of the op-amp. Using the interval algorithm to solve this problem, the designer is free to make a parameter choice at the highest design level, in agreement with the design of the two stages. The parameters for each stage are the bias current, the dc input/output voltage, and the dissipation. A possible linear description of the first stage, according to Fig. 3(a), could be given by

\[
V_{dd} = V_{dd} - |V_{dd}| - R_1 I_1
\]

\[
V_{dd} \in \{2.5, 3.8\}
\]

\[
R_1 = \left(\sqrt{B_m} S_1 I_1\right)^{-1} = 100 \text{ k\$}
\]

\[
I_1 \in \{2, 20\}
\]

\[
P_1 = (V_{dd} + |V_{dd}|) I_1
\]

\[
P_1 \in \{0, 0.1\} \tag{24}
\]

and for the second stage, a simple source-follower,

\[
V_{dd} = V_{dd} - R_2 I_2
\]

\[
V_{dd} \in \{-1, 1\}
\]

\[
R_2 = \left(\sqrt{2/B_m} S_2 I_2\right)^{-1} = 20 \text{ k\$}
\]

\[
I_2 \in \{10, 100\}
\]

\[
P_2 = (V_{dd} + |V_{dd}|) I_2
\]

\[
P_2 \in \{0, 2\} \tag{25}
\]

according to Fig. 3(b).

From the highest design level, the domains for the parameters are given. The solution space for the unknowns in the first stage as produced by the interval algorithm is given by

\[
\begin{bmatrix}
    V_{dd} \\
    I_1 \text{ (\$A)} \\
    P_1 \text{ (\$W)}
\end{bmatrix} = k_1 \begin{bmatrix}
    3.8 \\
    2 \\
    10
\end{bmatrix} + k_2 \begin{bmatrix}
    3 \\
    20 \\
    100
\end{bmatrix}
\]

\[
\text{with } \sum_{i=1}^{2} k_i = 1, \quad \forall k_i \geq 0. \tag{26}
\]
At this stage the full solution space is described. From these values using optimization criteria for $V_o$ and $P_{\text{diss}}$, the sizes of the transistors could be determined. For instance, if the designer wants $V_o = 0$, $r_2$ must be equal to 1. The sub-blocks are defined using (26), (27), (29) and (30). The first stage is designed with $V_{dss} = 3$, $I_o = 10$ $\mu A$, and $P_{1} = 0.1$ mW, and the second stage with $V_{dss} = 3$, $I_o = 100$ $\mu A$ and $P_{2} = 1$ mW. All the parameters take values within their design constraints. Using (24) and (25), the values for $S_3$ and $S_4$ are found. Simulations with SPICE prove the usefulness of linear equations for such a design problem.

Normally the equations describing the dc behavior of an op-amp are nonlinear. To deal with the nonlinear problem solution techniques for interval arithmetic in nonlinear equations are under development.

IV. CONCLUSION

The presented interval technique can be used in the design environment to partition and map design constraints on a normal operating range of a collection of sub-blocks. This technique is valid only for linear equations.

Using a top-down strategy, the first advantage is that the solution space, if it exists, does not conflict with the solution space at higher levels and so decisions made earlier remain valid. At each level there is a maximum amount of freedom for the set of design parameters. The second advantage of this technique is the implementation in an algorithm. At each hierarchical design level the same problem exists, which means that the same algorithm can be used at each level. This makes the implementation of the interval technique within the design process easier.

ACKNOWLEDGMENT

The author wishes to thank W. M. G. van Bokhoven and J. A. Hegt for the valuable discussions and suggestions.

REFERENCES