Synthesis of Low-Sensitivity Second-Order Digital Filters Using Genetic Programming with Automatically Defined Functions

Kazuyoshi Uesaka and Masayuki Kawamata

Abstract—This letter proposes a synthesis method for low coefficient sensitivity second-order IIR digital filter structures using Genetic Programming with Automatically Defined Functions (GP-ADF). In this letter, digital filter structures are represented as S-expressions with subroutines. It is easy to generate syntactically valid S-expressions and perform the genetic operations because the representation is suitable for GP. Numerical examples use the fitness measure including the magnitude sensitivity, and demonstrates that the proposed method can synthesize efficiently very low coefficient sensitivity filter structures.

I. INTRODUCTION

The synthesis of a digital filter is the process of converting the transfer function into a digital filter structure. In the implementation of digital filters in special-purpose hardware or in software on a general-purpose computer, the filter structure corresponds to a hardware configuration or a specification of an algorithm. There are unlimited number of equivalent filter structures realizing a given transfer function [1], and they may have different characteristics such as coefficient sensitivity, round-off noise, and execution speed. Thus it is of interest to choose the filter structure that provides satisfactory performance, depending on the type of digital filter being implemented.

Genetic Programming (GP) has been applied to the synthesis of low coefficient sensitivity digital filters [2]. The GP-based synthesis can search low coefficient sensitivity digital filter structures by representing filter structures as S-expressions. The disadvantage of the GP-based synthesis in [2] is the complicated way of representing the filter structures. For this reason, for example, crossover requires the post-process to generate the syntactically valid S-expressions.

In this letter, we propose a simple way of representing the digital filter structures using the S-expressions with subroutines. The new representation is suitable for GP and requires no post-process of crossover.

We restrict ourselves to the synthesis of second-order IIR structures with the transfer function:

\[
H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}},
\]

which can be used in cascade or in parallel for the realization of high-order transfer functions of order more than two.

II. GENETIC PROGRAMMING WITH AUTOMATICALLY DEFINED FUNCTIONS

Genetic Programming is a global optimization technique based on the rules of natural selection and natural genetics [3]. GP is an iterative procedure which maintains a constant size population of computer programs representing candidate solutions to a problem. In GP, the computer programs are typically encoded as S-expressions which are the syntactic forms in Lisp programming language [4].

GP starts with the generation of an initial population which consists of a fixed number of computer programs. The initialization procedure creates the computer programs randomly. The next step, evaluation procedure, performs all the programs in the population and gives the fitness values to them. On the basis of the given fitness values, selection operation copies computer programs according to their fitness values. After selection, new individuals are generated by applying genetic operations to the population. The genetic operations consist of crossover, mutation, and inversion. These operations are applied to the computer programs in the population probabilistically. Again, evaluation procedure performs the computer programs changed by the genetic operation, and then gives the new fitness values to them.

The cycle of selection, genetic operations, and fitness evaluation is referred to as the generation, and the cycle is repeated until a predefined number of generation is reached. The program with the highest fitness in the final population is considered as a solution to a given problem.

GP with Automatically Defined Functions (GP-ADF) can search the S-expressions with subroutines and solve more complex problems than those solved by plain GP [5]. In GP-ADF, the S-expression consists of one result-producing branch and one or more function-defining branches.

III. SYNTHESIS OF DIGITAL FILTERS USING GP-ADF

A. Representation

The S-expression is composed of three basic functions: adders, multipliers, and unit delays to represent the filter structure, because the realization of a filter structure requires these basic elements. Table I summarizes the program architecture for the representation. This representation is suitable for GP compared with the representation used in the other GP-based synthesis [2] because the function and argument sets are clearly defined.

Consider a digital filter in Fig. 1(c). The S-expression of the digital filter is represented as Fig. 1(a), which has two function-defined branches and is identified with the rooted-tree shown in Fig. 1(b). The function \( b_1 \) represents the sub-circuit between node \( b_0 \) and node \( b_1 \), and the value of the function \( b_1 \) is the same as the output signal of the node \( b_1 \).

In this letter, the S-expression consists of one result-producing branch and eight function-producing branches. The result-producing branch is rooted at the output function \( y \). The function-defining branches are rooted at the node functions...
The function name $b_k$ corresponds to the $k$-th node in the filter structure. The number of the function-defining branches limits the number of nodes having one or more output branches in the filter structure synthesized. The bodies of the function-defining branches and the result-producing branch are constructed from three primitive functions representing the multipliers, two-input adders, and unit delays, whose names are $m$, $a$, and $d$, respectively. The multiplier and unit delay functions have only one argument; the adder function has two arguments. The arguments to the S-expression are the filter input $x$ or the automatically defined functions \{ $b_0, b_1, \cdots, b_7$ \}.

### B. Genetic Operations

The proposed method uses three genetic operations: crossover, inversion, and mutation.

**Crossover:** Crossover creates new S-expressions by exchanging sub S-expressions between two S-expressions. The sub S-expressions exchanged are selected randomly.

**Inversion:** Inversion creates a new S-expression by exchanging two sub S-expressions in one S-expression. The sub S-expressions exchanged are selected randomly.

**Mutation:** Mutation creates a new S-expression by replacing an existing argument symbol with the other possible symbol. The argument symbol replaced is selected randomly.

### C. Fitness Measure

The fitness measure for the synthesis of filter structures is concerned with the characteristics of the desired filter structure. In this letter, the fitness measure is defined as the sum of four evaluation values as follows:

$$ f = f_{sens} + f_{size} + f_{phys} + f_{tf}. $$  

(2)

The value $f_{sens}$ is defined as

$$ f_{sens} = S_{max} - S, $$  

(3)

where $S$ is the magnitude sensitivity [2] of the filter structure and the value $S_{max}$ is the maximum of the coefficient sensitivity values of all the filter structures in the population.

The value $f_{size}$, concerned with the number of functions in the S-expression, is defined as

$$ f_{size} = \begin{cases} 10, & \text{if } N_m \leq 5, N_a \leq 8, \text{ and } N_d \leq 4, \\ 0, & \text{otherwise}, \end{cases} $$  

(4)

where $N_m$, $N_a$, and $N_d$ are the number of $m$, $a$, and $d$ in the S-expression, respectively.

The value $f_{phys}$ is concerned with the physical realizability of the filter structures. The value $f_{phys}$ is defined as

$$ f_{phys} = \begin{cases} 10, & \text{if the filter structure is physically realizable,} \\ 0, & \text{otherwise,} \end{cases} $$  

(5)

where the filter structures with delay-free loops are regarded as being non-realizable.

The value $f_{tf}$, concerned with the realizability of a given transfer function, is defined as

$$ f_{tf} = \begin{cases} 10, & \text{if the S-expression realizes the transfer function } H(z), \\ 0, & \text{otherwise}. \end{cases} $$  

(6)

Note that all the multiplier coefficients in the S-expressions are regarded as unknown numbers. Thus, the multiplier coefficients should be calculated by the coefficient comparison between the transfer function $H(z)$ and $H'(z)$ of the S-expression with regard to the powers of $z^{-1}$ [2], where $H'(z)$ is the transfer function given by the S-expression.

### IV. DESIGN EXAMPLES

#### A. Example 1: Low-pass filter

To demonstrate the performance of the proposed synthesis method, we consider the synthesis of low coefficient sensitivity digital filter structures realizing the following transfer function:

$$ H(z) = \frac{0.098244 - 0.195065z^{-1} + 0.9586936z^{-2}}{1 - 1.957184z^{-1} + 0.9586936z^{-2}}. $$  

(7)
This transfer function has poles located close to the unit circle, and it is known to be highly sensitive to coefficient quantization.

Fig. 2 shows the filter structure synthesized by GP-ADF. The magnitude sensitivity $S$ of this filter structure is 15.370, while the coefficient sensitivity of the direct-form II realizing (7) is 2398.883, that of the filter structure shown in [2] is 22.451, and that of the balanced realization, which has the minimum coefficient sensitivity in some sensitivity measure [6], is 75.8174. The run for the synthesis has taken 22.1 minutes on a 400-MHz Intel Pentium II processor.

### B. Example 2: Band-pass filter

The next example is to synthesize the low coefficient sensitivity digital filter structures realizing the following transfer function:

$$H(z) = \frac{0.001347 - 0.001347z^{-2}}{1 - 0.434780z^{-1} + 0.997305z^{-2}}. \quad (8)$$

Fig. 3 shows the filter structure synthesized by GP-ADF. The magnitude sensitivity $S$ of this filter structure is 53.2182, while the coefficient sensitivity of the direct-form II realizing (8) is 722.6477 and that of the balanced realization is 498.7450.

### V. Conclusions

We have proposed the synthesis method of the second-order low coefficient sensitivity digital filter structures using Genetic Programming with Automatically Defined Functions. In this method, digital filter structures are represented as the S-expressions with subroutines. We have also defined the fitness measure including the value concerned with the coefficient sensitivity of the filter structures implemented. The numerical examples have shown that GP-ADF can synthesize the very low magnitude sensitivity filter structure efficiently.

### ACKNOWLEDGMENT

The authors thank anonymous reviewers and the editor for their helpful comments and suggestions.

### REFERENCES