Synthesis of Low Coefficient Sensitivity Digital Filters Using Genetic Programming

Kazuyoshi Uesaka and Masayuki Kawamata

Department of Electronic Engineering, Graduate School of Engineering, Tohoku University
Aoba-yama 05, Sendai 980-8579, JAPAN
Phone: +81-22-217-7094, Fax: +81-22-263-9169, E-mail:uesaka@ecei.tohoku.ac.jp

Abstract— **This paper proposes a new approach to the synthesis of low coefficient sensitivity digital filters using Genetic Programming (GP). GP is applied to the synthesis problem by establishing a mapping between the S-expressions and the filters. Genetic operators can then be applied to change the connections between the elements in the filters, and consequently to change the coefficient sensitivities of those filters. Fitness measure that includes the coefficient sensitivities enables the selection operator to choose low sensitivity filters. A numerical example is presented to illustrate that the sensitivity of the filter, synthesized by GP, is lower than that of the minimum coefficient sensitivity state-space realization.**

I. Introduction

Digital filters implemented either in software on a general-purpose computer, or in special-purpose hardware, are always realized by using binary words of finite length. The behavior of digital filters with coefficients represented by finite wordlength may differ considerably from its initial design. Hence, it is desired to search filters that are less sensitive to the coefficient changes due to wordlength limitations, or what is called low coefficient sensitivity digital filters.

Many synthesis methods have been proposed which lead to low coefficient sensitivity digital filters. One of the effective synthesis methods is the state-space approach. This approach searches optimal state-space realizations that minimize the output error variance due to arithmetic roundoff errors using similarity transformation [1,2]. Although this method was developed for the synthesis of minimum roundoff digital filters, it is also available for low coefficient sensitivity digital filters [3]. Another effective method is coefficient conversion, which realizes low-sensitive filters by replacing each of the multipliers by two multipliers in parallel [4].

The above methods synthesize low coefficient sensitivity filters systematically. However, they search low coefficient sensitivity filters within only a set of limited filters, that is, it is expected that there exist filters whose coefficient sensitivities are much lower than the low coefficient sensitivity filters proposed so far.

In this paper, we present an evolutionary approach to the synthesis of low coefficient sensitivity digital filters using Genetic Programming (GP). Section II outlines GP. Section III outlines the synthesis of digital filters. Section IV describes our implementation of GP for the synthesis of low coefficient sensitivity digital filters. A numerical example and its result are given in Section V. Section VI gives some concluding remarks.

II. Genetic Programming

GP is a technique which evolves computer programs representing possible solutions to a problem [5]. GP starts with a population of individual computer programs created randomly. Each individual in the population is executed and then measured in terms of how well it performs in the population. This measure is called the fitness. The fitness values are assigned to all the individuals in the population. They are used as a basis for selection. Selection increases high-fitness individuals in the population and eliminates low-fitness individuals from the population. After selection, new individuals are generated by applying genetic operations to the population. Each program in the new population is then measured for fitness. The cycle of selection, genetic operations, and fitness evaluation, is repeated until a predefined number of generation N_{gen} is reached. The program with the highest fitness in the final population is considered as the desired program, i.e. the solution to the problem.

III. Synthesis of Digital Filters

The synthesis of a digital filter is the process of converting the transfer function into a structure. The synthesis requires some criteria for selecting the desired filter because digital filters synthesized can differ quite significantly with respect

to complexity, number of elements, and their property.

In this paper, we use three criteria: the physical realizability f_{phys}, the realizability of a given transfer function f_{tf}, and the coefficient sensitivity S. The first criterion f_{phys} is a criterion whether the filter has delay free loops or not. The second criterion f_{tf} is a criterion whether the order of the synthesized filter is equal to the one obtained after approximation or not. The third one is concerning the value of coefficient sensitivity S defined as

$$S = \max_{\omega} \sum_{i}^{M} \left| \frac{\partial |H(e^{j\omega})|}{\partial m_i} \right|, \tag{1}$$

where $H(e^{j\omega})$ is the transfer function of the filter, m_i is a multiplier coefficient, and M is the number of multipliers in the filter.

IV. GP-based Synthesis Method

There are three preparatory steps in applying GP to a problem. These three steps involve determining the representation scheme, the fitness measure, and the method of genetic operations.

A. Representation

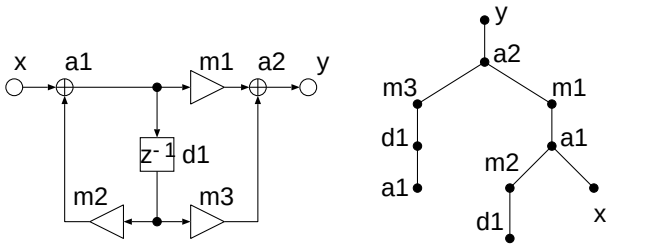
GP can be applied to the synthesis of digital filters if a mapping between the filters and computer programs can be established. For this reason, digital filters are represented as S-expressions in this paper. The S-expression is the syntactic form in Lisp programming language [6].

The S-expression of a digital filter in Fig. 1(a) is

```
(y (a2 (m1 (a1 (x)
               (m2 (d1))))
       (m3 (d1 (a1))))),
```

which is identified with the tree shown in Fig. 1(b). In this tree, the terminal node x is the filter input; the root of the tree, labeled y, is the filter output; the non-terminal nodes such as $a_1, a_2, \cdots, d_1$ represent the basic elements of the filter. The symbols a_i, m_i, and d_i correspond to adders, multipliers, and delays, respectively. The connections between the node a_2 and the two nodes m_3 and m_1 show that m_3 and m_1 supply their output signals to the input of a_2.

In the terminal nodes, we place the filter input x and some nodes selected from the set of non-terminal nodes like a_1 and d_1 in Fig. 1. The number of the nodes of each terminal is equal to the value obtained by subtracting unity from the sum of all output branches of the corresponding basic element; the number of the terminal node x is equal to the number of the



(a) an example of a filter (b) the corresponding S-expression tree

Fig. 1. Representation of a digital filter.

branches from the filter input x. For example, the element a_1 in Fig. 1(a) has two output branches to m_1 and to d_1, and thus there is only one terminal node a_1 in Fig. 1(b).

B. Fitness Measure

The fitness measure in this paper is the sum of the values obtained by three criteria shown in Section III. The physical realizability f_{phys} is set to 10 if the filter does not have delay free loops, and if not, 0. The realizability of a given transfer function f_{tf} is set to 50 if the order of the synthesized filter is equal to the one obtained after approximation, and if not, 0. The criterion concerning the coefficient sensitivity is set to $S_{max} - S$ in order to assign a high fitness value to low sensitivity filters. The value S_{max} is the maximum of the coefficient sensitivity values of all the digital filters in the population.

A calculation of the coefficient sensitivity in (1) requires the multiplier coefficients in the filter. In this paper, all the multiplier coefficients are regarded as unknown quantities. Hence, we must obtain the multiplier coefficients by the following procedure: first, the transfer function in terms of the unknown multiplier coefficients is obtained by applying the Mason's theorem to the tree, and second, a comparison between the obtained transfer function and the desired transfer function forms a set of simultaneous equations. The set of the solutions to these equations gives the multiplier coefficients in the filter.

C. Selection and Genetic Operations

Selection operation in this paper is a combination of two selection strategies: elitist selection and tournament selection. The elitist selection copies N_{sur} best individuals in the next generation. The number N_{sur} is called the survival size. The rest of individuals is selected by the tournament selection, where a predefined number of individuals are selected randomly from the population, and a selective competition takes

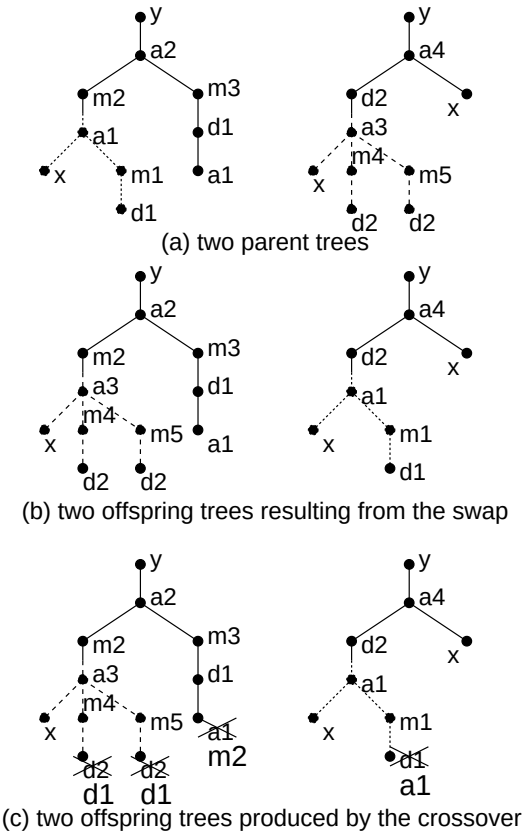


Fig. 2. Crossover between two filters at the tree level.

TABLE 1

CONTROL PARAMETERS FOR GP.

Population size	N_{size}	400
Survival size	N_{sur}	2
Tournament size	N_{tour}	5
Generations	N_{gen}	20
Crossover rate	P_{cross}	0.6
Mutation rate	P_{mut}	0.01
Inversion rate	P_{inv}	0.3

place. This number is called the tournament size N_{tour}. Only the highest-fitness individual in each tournament population is allowed to copy in the next generation. The competition is repeated until the sum of individuals selected by the elitist selection and by the tournament election reaches the population size N_{size}.

The genetic operators such as crossover, mutation and inversion are applied to the individuals selected by the tournament selection. They are applied to each tree with probabilities, P_{cross} (crossover rate), P_{mut} (mutation rate), and P_{inv}

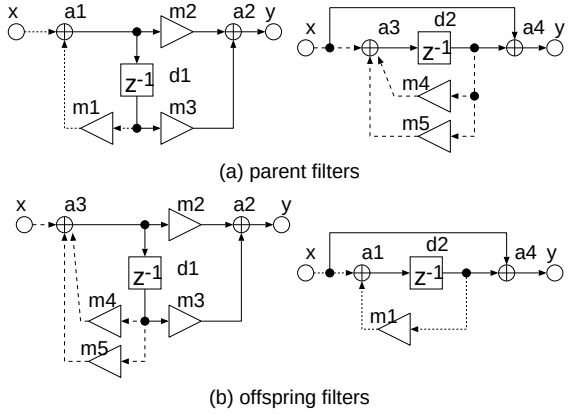


Fig. 3. Crossover between two filters at the structural level.

(inversion rate), respectively.

Crossover creates new offspring by swapping sub-trees between the two parents. The sub-trees to be swapped are chosen randomly. The trees resulting from the swap, however, cannot represent filters because they have some terminal nodes that are not members of the set of non-terminal nodes. For this reason, all such terminal nodes are replaced with nodes selected randomly from the set of non-terminal nodes.

Fig. 2 shows a diagram of crossover. In Fig. 2(a), the nodes a_1 and a_3 are randomly selected as the crossover point of the left parent and the right parent respectively. Fig. 2(b) shows the offspring trees resulting from the swap. These trees cannot represent the filters because the left offspring has the terminal nodes d_2's and a_1, and the right offspring has the terminal node d_1. In Fig. 2(c), d_2's, a_1, and d_1 are replaced with d_1's, m_2, and a_1 which do not appear terminal nodes in 2(b). The filters shown in Figs. 3(a) and (b) correspond to the trees shown in Figs. 2(a) and (c) respectively.

Mutation selects a node in an offspring tree randomly, and then replaces the existing sub-tree at the node with a new randomly generated sub-tree. In this paper, the new randomly generated sub-tree is composed of a set of the terminal and non-terminal nodes of the sub-tree selected for the mutation. Inversion selects two random nodes in an offspring tree and then swaps the sub-trees at the selected nodes.

V. NUMERICAL EXAMPLE

To illustrate the performance of the synthesis method presented in this paper, we consider a second-order low-pass narrow bandwidth filter having the following transfer function:

$$H(z) = \frac{0.098244 - 0.195065z^{-1} + 0.098244z^{-2}}{1 - 1.957184z^{-1} + 0.958693z^{-2}}. \quad (2)$$

TABLE 2

COEFFICIENT SENSITIVITY AND THE NUMBER OF MULTIPLIERS, TWO-INPUT ADDERS AND BRANCHES.

	sensitivity S	multipliers	two-input adders	branches
filter synthesized by GP	22.451	4	9	20
optimal state-space realization [7]	31.049	8	5	19
direct form II	2393.883	5	5	16

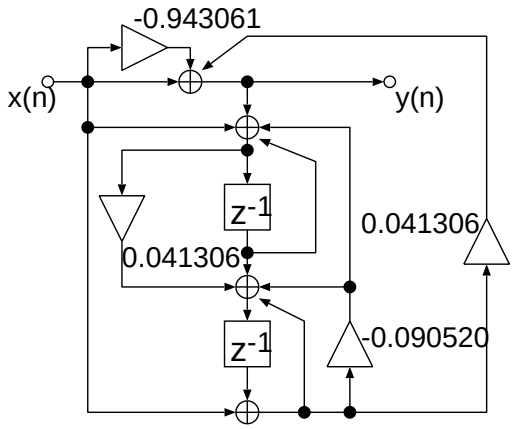


Fig. 4. Filter synthesized by GP.

The GP parameters used in our experiment are shown in Table 1. The GP-based synthesis software has been developed in Common Lisp. The run for the synthesis has taken 18.11 hours on the SUN SPARCserver 1000E consisting of four 85-MHz SuperSPARC-II processors.

Fig. 4 shows the new filter synthesized by our method. Table 2 compares the filter synthesized by GP with the minimum coefficient sensitivity state-space realization [7] and the direct form II in terms of coefficient sensitivity S. Table 2 also shows the numbers of multipliers, two-input adders and branches of these realizations. The number of filter branches corresponds to the number of branches in the corresponding tree. From this table, we can observe that the coefficient sensitivity of the filter synthesized by GP is lower than that of the minimum coefficient sensitivity state-space realization. Moreover, the filter synthesized by GP requires much less multipliers — only the half of the number necessary for the state-space realization.

VI. CONCLUDING REMARKS

We have proposed the GP-based method for synthesizing low coefficient sensitivity digital filters. The feature of our synthesis method is to search desired filters by changing the topology of filters. In particular, the key techniques we have developed for the implementation in this paper are as follows: the representation of digital filters as programs (S-expressions), the fitness measure including the coefficient sensitivity, and the genetic operators changing the topology of filters. The numerical example has illustrated that the coefficient sensitivity of the filter synthesized by GP is lower than that of the minimum coefficient sensitivity state-space realization. Moreover, the filter synthesized by GP requires only the half of multipliers necessary for the state-space realizations.

REFERENCES

[1] C. T. Mullis and R. A. Roberts, "Synthesis of minimum roundoff noise fixed point digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-23, pp. 551–562, September 1976.

[2] S. Y. Hwang, "Minimum uncorrelated unit noise in state-space digital filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 273–281, August 1977.

[3] M. Kawamata and T. Higuchi, "A unified approach to the optimal synthesis of fixed-point state-space digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 911–920, August 1985.

[4] R. C. Agarwal and C. S. Burrus, "New recursive digital filter structures having very low sensitivity and roundoff noise," *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 921–927, December 1975.

[5] J. R. Koza, *Genetic Programming*. The MIT Press, 1992.

[6] P. H. Winston and B. K. P. Horn, *LISP*. Addison-Wesley, 1989.

[7] C. W. Barnes, "Computationally efficient second-order digital filter sections with low roundoff noise gain," *IEEE Trans. Circuits Syst.*, vol. CAS-31, pp. 841–847, October 1984.