Automatic Floating-Point to Fixed-Point Transformations

Kyungtae Han, Alex G. Olson, and Brian L. Evans
Dept. of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712–1084 USA
Email: han@ieee.org, aolson@ece.utexas.edu, bevans@ece.utexas.edu

***Abstract*— Many digital signal processing and communication algorithms are first simulated using floating-point arithmetic and later transformed into fixed-point arithmetic to reduce implementation complexity. For the floating-point to fixed-point transformation, this paper describes two methods within an automated transformation environment. The first method, a gradient-based search for single-objective optimization with sensitivity information, provides a single solution, and can become trapped in local optima. The second method, a genetic algorithm for multi-objective optimization, provides a family of solutions that form a tradeoff curve for signal quality vs. implementation complexity. We provide case studies for an infinite impulse response filter. In the case study, implementation complexity is lookup table area for a field programmable gate array (FPGA) realization. We have made the transformation methods available in a software release on the Web.**

I. Introduction

Realization of digital signal processing algorithms with the fixed-point data type provides many benefits, such as savings in power and area. Typically, the algorithms are developed in floating-point arithmetic and later transformed to the fixed-point arithmetic with manually evaluated tradeoffs in signal quality and implementation complexity. Since the transformation process is time consuming and error prone, many methods have been proposed and developed to automate the fixed-point transformation [1]–[6].

Fixed-point transformation consists of fixed-point conversion and wordlength optimization. Fixed-point conversion is a process of converting floating-point programs to fixed-point programs. During the conversion, floating-point data type is changed into the fixed-point data type, and floating-point arithmetic operations are modified to fixed-point arithmetic operations.

Digital signal processors, in which wordlengths have already been given, require fixed-point conversion followed by scaling to prevent the overflow and underflow of signals. Wordlength optimization is required only when the given wordlength is too short to satisfy the desired performance or to reduce power consumption [7].

For targeting a customized integrated circuit (IC) or a FPGA, wordlengths can be chosen to be any width with tradeoffs in objectives. Shorter width usually achieves savings in area and power, while signal distortion is higher. Wordlength can be optimized by optimization algorithms [3]–[6], [8], [9] and those algorithms can also automate the optimization process.

Most of the algorithms minimize hardware area by satisfying error specifications. Sometimes, designers make tradeoffs between error specifications and hardware area instead of fixing one objective. This paper proposes multi-objective wordlength optimization, which optimizes more than one objective at the same time. Furthermore, an environment for automated fixed-point transformation is proposed and demonstrated with a case study.

II. Related Work

A. Fixed-Point Simulation Environment

Many methods have been developed to model fixed-point systems. TI developed a fixed-point C++ class to facilitate implementation of fixed-point DSP algorithms [10]. In [1], [11], a fixed-point simulation environment is implemented for C and C++ at Seoul National University. By modifying variable declarations of the floating-point code and overloading operators in the *gFix* class, the floating-point data type is converted to the fixed-point data type. In [12], annotation and interpolation techniques to convert the floating-point data type to fixed-point data type with analytical range estimation are employed.

B. Wordlength Optimization

Sung and Kum [8] developed a simulation-based wordlength optimization algorithm. Optimum wordlengths are searched from a minimum wordlength state by increasing wordlengths with priority on a hardware block having the lowest hardware cost. In [9], an area model and a noise model are proposed. For an objective in optimization, area-based objective functions and error models are used. A mixed integer linear programming (MILP) model and heuristic search methods are employed to solve the wordlength optimization problem. Shi and Brodersen [5] use simulation-based methods to evaluate various sensitivities. Simulation results are used to develop a model of the system that is used for wordlength optimization. The Mosek optimizer, which handles single objective optimization, is used as a search engine.

C. Gradient Search

Gradient-based search algorithms utilize gradient information of wordlength to find better solutions. The gradient information can be obtained from sensitivity measurements of complexity (CM) or distortion (DM) according to the wordlength set. A complexity-and-distortion measurement method (CDM) utilizes all sensitivity information simultaneously [13]. The CDM seems to be a multi-objective algorithm. However, this is not a multi-objective algorithm, as its objective is a single weighted sum of the complexity and the distortion metrics.

D. Genetic Algorithm

In 1975, Holland introduced an optimization procedure that mimics the process observed in natural evolution [14] and is known as the genetic algorithm, or GA [15], [16]. Because of its simple implementation procedure, the GA can be used as an optimization tool for designing AI-hybrid systems for real-world applications.

Most problems in nature have several objectives (normally conflicting with each other) that need to be achieved at the same time. These problems, called "multi-objective" optimization problems, were originally studied in the context of economics [17]. Because of the conflicting nature of their objectives, multi-objective optimization problems do not normally have a single solution, and, in fact, they even require the definition of a new notion of "optimum." The most commonly adopted notion of optimality in multi-objective optimization is that originally proposed by Edgeworth [18] and later generalized by Pareto [19]. Such a notion is called *Edgeworth-Pareto optimality* or, more commonly, *Pareto optimality*.

E. Wordlength Optimization with Multi-Objective Evolutionary Algorithms

When implementing a digital filter in hardware, filter coefficients have to be represented with finite wordlength. Several methods have been proposed to effectively design finite impulse response (FIR) filters with linear programming [20]. Xu and Daley [21] show that GA is superior to integer programming techniques in fixed-point filter design.

Genetic algorithms have been applied to wordlength design in digital signal processing. Wordlengths in digital signal processing are analogous to genes, and each set of wordlengths is analogous to a chromosome. The GA is used to determine wordlength in filter coefficients [22] and to optimize the wordlength of input data and coefficients in an FFT processor [23] with a single objective.

Some papers have employed multiple objectives for wordlength optimization. Leban and Tasic [24] used mean square error, delay, and area as objectives. Signal-to-noise ratio and power are used as objectives by Sulaiman and Arslan [25]. These approaches employed a weighted sum as a fitness function. As in the case of the weighted sums methods, the relative importance of objectives should be specified using weights (quantitatively). Furthermore, a simple weighted-sum technique only finds a single solution of the many possible optimal solutions in the objective space. Thus, the single solution does not provide the ability to understand the various tradeoffs that are possible in objective space [26].

```
Function c=adder(a,b)
c=0;
c=a+b;
```

(a) Floating point program for adder

```
Function [c]=adder_fx(a,b,numtype,mathtype)
c=0;
a=fi (a,numtype.a,mathtype.a);
b=fi (b,numtype.b,mathtype.b);
c=fi (c,numtype.c,mathtype.c);
c(:)=a+b;
```

(b) Converted fixed-point program for adder (only the core code is shown)

Fig. 1. Conversion to fixed point by a code generator

In this paper, Pareto ranking approaches are used for multiple objective evolutionary algorithms to optimize wordlength, and the results are shown in Section IV.

III. Automated Transformation from Floating Point to Fixed Point

The proposed transformation from floating point to fixed point has three phases: code generation, range estimation, and wordlength optimization. A code generator converts floating-point programs to fixed-point programs that handle fixed-point data types and arithmetic. The code generator also creates other auxiliary programs for an automatic transformation environment. A range estimator finds range information in the fixed-point system to prevent overflow and underflow. Wordlength optimization finds the optimum wordlength according to objectives such as signal distortion and hardware complexity.

A. Code Generation

The first process in the automated transformation to fixed point is code generation. A given floating-point program shown in Fig 1 (a) is converted to a fixed-point program that can handle variable wordlengths by a parameterized input, as shown in Fig. 1 (b).

The *fi* is one of the functions in the Fixed-Point Toolbox in MATLAB to define a fixed-point data type [27]. Each fixed-point variable is defined via an input parameter and number type instead of constants. This input parameter is controlled by wordlength optimization programs.

The code generator also creates several programs for a fixed-point transformation environment, as shown in Fig. 2. The top program generated by a code generator plays a role as headquarters in the transformation to fixed-point. It establishes an environment for a range estimation and optimum wordlength search according to configurations that can be modified by designers. The top program mainly executes range estimations and wordlength optimizations.

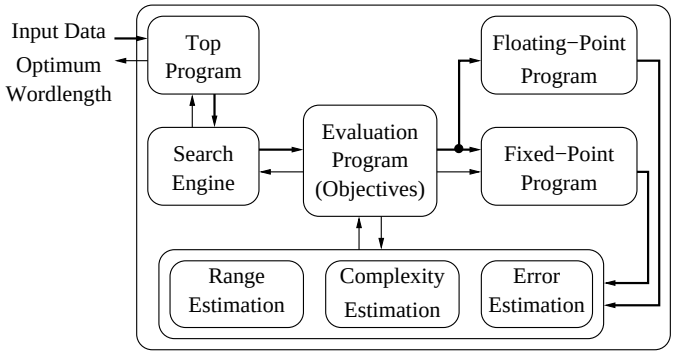


Fig. 2. Automated transformation environment

The optimum wordlength depends on the input signal properties. The input signal is passed by the top code, search engine, and objective code. The top code calls a search engine, which explores the wordlength space to find the optimum wordlength. The genetic algorithm can be used as a search engine for multi-objective optimization.

The evaluation code collects objective values according to wordlength states and input signal. One of objectives can be a signal error, which is the difference between the floating-point output and fixed-point output. Complexity, power consumption, or timing information can be used as an objective.

B. Range Estimation

Range information is used to determine integer wordlength in order to prevent overflow and underflow. A signal range can be estimated by two methods. One is a simulation-based method and the other is an analytical method. A simulation-based method monitors the signal range of variables and finds a maximum value and a minimum value. An analytical method calculates signal ranges by using a range-propagation property through operations. Simulation is not necessary in the analytical method. However, the calculated result from an analytical method is conservative, and wordlength could grow infinitely in feedback systems. A simulation-based method is useful for complicated systems, including loops; however, it needs time for the simulation. Both methods can be used selectively. The simulation-based method can be used in feedback parts, and the analytical method can be used in other parts.

C. Optimum Wordlength Search

Optimum wordlength can be found with wordlength optimization algorithms that have search algorithms, such as a complete search, gradient-based search, and genetic and evolutionary algorithms. One of the more powerful search engines is the genetic and evolutionary search engine. The genetic search engine handles multiple objectives and finds a Pareto front, although the computational complexity of this algorithm is very high.

The search engine generates wordlength candidates, which the evaluation function then evaluates. The information of the objective values could be used to generate the next candidates. For the evaluation, the error value or difference value between the floating-point programs and fixed-point programs can be modeled by an analytical or statistical approach. Analytical approaches estimate the error at each system output. In the statistical approach, simulation is used to estimate the error.

Cost value can be obtained by modeling the fixed-point systems. Modeling the exact implementation scheme used would be specific to the vendor. Area models in [9] are used for complexity estimation.

TABLE I

SIMULATION RESULTS IN IIR FILTER OF SEVERAL SEARCH METHODS

Search Method	*Gradient Measure*	*Number of Simulations*	*Complexity Estimate (LUT)*	*Distortion (RMS)*
Gradient	DM	316	51.05	0.0981
Gradient	CDM	145	49.85	0.0992
Gradient	CM	417	51.95	0.0986
Complete	-	16^7	-	-

IV. CASE STUDY

A. Infinite impulse response filter

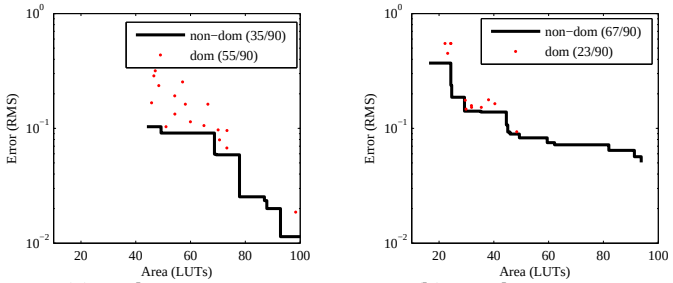
To illustrate the methods presented in this paper, the infinite impulse response (IIR) filter that has 7 wordlengths is simulated. There are various methods for deriving the error function and cost function. For simplifying the simulation, the root mean squared (RMS) error is measured for the error function, and a linear cost function of wordlength is assumed. The required performance of the IIR filter is assumed to be a maximum error of 0.1 RMS.

Simulation results of gradient-based search algorithms in IIR filter are shown in Table I. Each algorithm obtains a single solution for a simulation. CDM methods show smaller number of simulations and complexity satisfying minimum distortion requirement of 0.1. However, the complete search requires huge number of simulations to search entire search space.

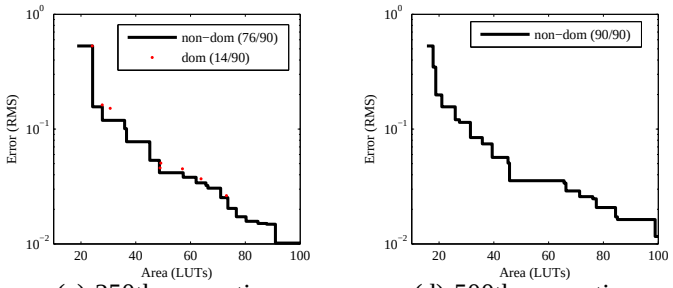
Simulation results utilizing multiple objective genetic and evolutionary algorithms on wordlength design in an IIR filter are shown in Fig. 3. The total hardware area, one of the objectives, is evaluated by the area model of the arithmetic unit in [9]. The error between the floating-point output and fixed-point output is measured by simulations.

Since the genetic algorithm mimics the evolutionary process of plants and animals, each generation shows different results. Fig. 3 shows nondominated and dominated (inferior) solutions at each generation. The plot of the objective functions whose non-dominated vectors are in the Pareto optimal set is called the Pareto front. After many generations, the Pareto front tends to move toward the left and downward. The number of dominated solutions decreases as the number of generations increases. The 500th generation has only non-dominated solutions.

Designers have a choice of wordlength solutions according to the Pareto front. The Pareto front gives the tradeoff curve in hardware area and finite precision output error. A smaller area requires a larger error, and a larger area needs a smaller error. Thus, the obtained Pareto front gives designers flexibility in a system design.

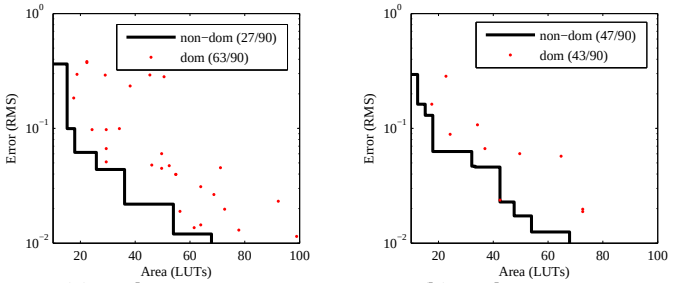


(a) 50th generation (b) 100th generation

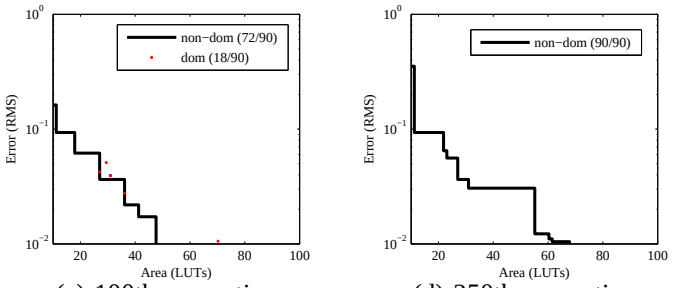


(c) 250th generation (d) 500th generation

Fig. 3. Results of multiple objective genetic and evolutionary algorithms in the IIR filter case study with seven wordlength variables (Population for one generation is 90)



(a) 25th generation (b) 50th generation



(c) 100th generation (d) 250th generation

Fig. 4. Result of multiple objective genetic and evolutionary algorithms in the IIR filter case study with three wordlength variables (Population for one generation is 90)

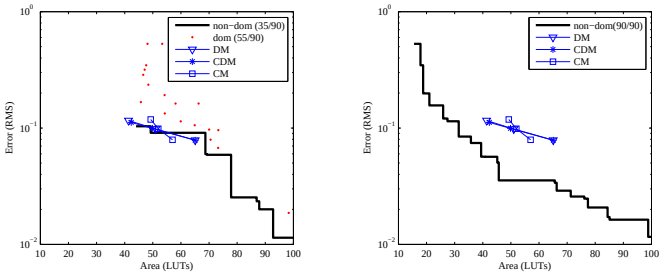
Note that the Pareto front in a descendant is not always better than that in ancestor. The solution for an error of 10^{-2} at the 250th generation required 90 lookup tables (LUTs). However, the solution for the same error at the 500th generation needs at least 100 LUTs. Thus, the 250th generation has a better solution for the error of 10^{-2} than the 500th generation. The offspring could be worse than their ancestors because the genetic and evolutionary algorithm utilizes a random process throughout selection, mating, and mutation.

The genetic and evolutionary algorithm is computationally intensive. It requires many simulations for errors in each population as well as the genetic operations of selection, mating, and mutation for each generation. Considering only the number of simulations for errors, the 500th generation requires 45,000 ($= 500$ generations $* 90$ populations) simulations.

A reduced number of variables can reduce the number of simulations. Fig. 4 shows results in the IIR filter study with three wordlength variables. The wordlength at the output of the multipliers are selected for three variables. The results show the same trends as with seven wordlength variables. However, all solutions at the 250th generation are non-dominated. Thus, 22500 ($= 250$ generations $* 90$ populations) are sufficient for three variables in this case study.

B. Comparison

The gradient-based search algorithms and genetic algorithms are compared. Results from gradient-based search algorithms with the FPGA area models are superposed on the results from a genetic algorithm in Fig. 5. Three desired RMS values of 0.08, 0.1, and 0.12 are given, since the gradient-based search algorithms generate one solution each. DM, CDM, and CM are used as gradient-based search algorithms.



(a) 50th generation (b) 500th generation

Fig. 5. Overlap genetic evolutionary algorithm results in 50th and 500th generation with gradient-based search results for the IIR filter case study with seven wordlength variables (Required RMS for gradient-based search $\leq \{0.12, 0.1, 0.08\}$)

The solutions from the gradient-based search algorithms are similar to the Pareto front in the genetic algorithm at the 50th generation. The RMS error of 0.1 needs approximately 50 LUTs in both methods. However, at the 500th generation the genetic algorithm finds better solutions than the gradient-based search algorithms, as shown in Fig. 5 (b). This difference demonstrates that the gradient-based search methods are trapped by local optima. However, the genetic algorithm can avoid local optima.

With the respect to computational complexity, gradient-based search algorithms need less simulation time compared to genetic algorithms. The gradient-based search algorithms require 145 simulations for CDM and 417 for CM, whereas the genetic algorithm needs 4,500 simulations to obtain a similar result and 45,000 simulations for the 500th generation. Furthermore, the genetic algorithm requires computations to execute genetic operations at every generation.

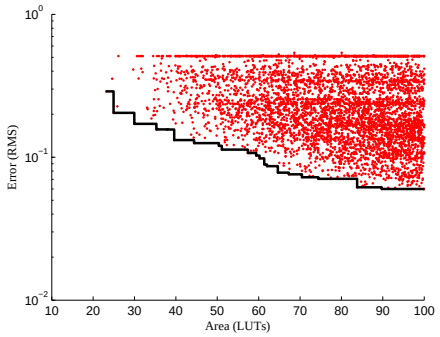


Fig. 6. Result of random search algorithm in the IIR filter study with seven wordlength variables (45,000 samples)

For comparison with a genetic algorithm approach using 500 generations and a population of 90 per generation, we generate 45,000 randomly selected wordlengths and compute the Pareto front. Fig. 6 shows the result. The Pareto front of the genetic algorithm as shown in Fig. 3 (d) is better than that of the random search algorithm as shown in Fig. 6. This simulation result shows that the genetic algorithm outperforms the random search algorithm with the same number of samples.

Parallel processing can decrease the running time. The genetic algorithm can be parallelized up to population size since individuals can be evaluated independently. However, gradient-based search algorithms are limited in their use of parallelism because gradient-based search algorithms evaluate the next neighbors and move a current point to one of the neighbors. Thus, the gradient-based search algorithm can be parallelized only up to the number of neighbors or wordlength variables.

V. Conclusion

Techniques for automating the transformation from floating point to fixed point in software are presented. This software provides an environment to transform floating-point programs to fixed-point programs for digital signal processing algorithms. Fixed-point conversion and wordlength optimization are executed in this environment. Wordlength optimization algorithms utilizing genetic and evolutionary algorithms can optimize the tradeoff between signal quality and implementation complexity. Alternatively, wordlength search algorithms utilizing gradient information can provide faster ways to find data wordlengths, but they get caught in local optima. The automated transformation software is available at

http://www.ece.utexas.edu/˜bevans/projects/wordlength/

References

[1] S. Kim, K. Kum, and W. Sung, "Fixed-point optimization utility for C and C++ based digital signal processing programs," *IEEE Trans. Circuits Syst.*, vol. 45, no. 11, pp. 1455–1464, Nov. 1998.

[2] K. Kum, J. Kang, and W. Sung, "AUTOSCALER for C: An optimizing floating-point to integer C program converter for fixed-point digital signal processors," *IEEE Trans. Circuits Syst.*, vol. 47, pp. 840–848, Sept. 2000.

[3] M. Cantin, Y. Savaria, and P. Lavoie, "A comparison of automatic word length optimization procedures," in *Proc. IEEE Int. Sym. on Circuits and Systems*, Phoenix-Scottsdale, Ariz, USA, May 2002, pp. 612–615.

[4] K. Han, I. Eo, K. Kim, and H. Cho, "Numerical word-length optimization for CDMA demodulator," in *Proc. IEEE Int. Sym. on Circuits and Systems*, vol. 4, Sydney, NSW, Australia, May 2001, pp. 290–293.

[5] C. Shi and R. W. Brodersen, "Automated fixed-point data-type optimization tool for signal processing and communication systems," in *Proc. Design Automation Conf.*, San Diego, CA, June 2004, pp. 478–483.

[6] K. Han and B. L. Evans, "Wordlength optimization with complexity-and-distortion measure and its applications to broadband wireless demodulator design," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Proc.*, vol. 5, Montreal, Quebec, Canada, May 2004, pp. 37–40.

[7] K. Han, B. L. Evans, and E. E. Swartzlander, "Low-power multipliers with data wordlength reduction," in *Proc. IEEE Asilomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, USA, Oct. 2005, pp. 1615–1619.

[8] W. Sung and K. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Trans. Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.

[9] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Trans. Computer-Aided Design*, vol. 22, no. 10, pp. 1432–1442, Oct. 2003.

[10] W. Cammack and M. Paley, "Fixpt: a C++ method for development of fixed point digital signal processing algorithms," in *Proc. Hawaii Int. Conf. on System Sciences*, Wailea HI, Jan. 1994, pp. 87–95.

[11] K. Kum, J. Kang, and W. Sung, "A floating-point to fixed-point C converter for fixed-point digital signal processors," in *Proc. SUIF Compiler Workshop*, Aug. 1997.

[12] H. Keding, M. Willems, M. Coors, and H. Meyr, "FRIDGE: A fixed-point design and simulation environment," in *Proc. IEEE Design Automation and Test in Europe*, France, France, Feb. 1998, pp. 429–435.

[13] K. Han and B. L. Evans, "Optimum wordlength search using a complexity-and-distortion measure," *EURASIP Journal on Applied Signal Processing*, vol. 2006, no. 5, pp. 103–116, 2006.

[14] C. Darwin, *The Origin of Species by Means of Natural Selection*. Blatimore, Maryland: Penguin Books, 1859.

[15] D. E. Goldberg, "Genetic and evolutionary algorithms come of age," *Communications of the ACM*, vol. 37, no. 3, pp. 113–119, Mar. 1994.

[16] J. H. Holland, *Adaptation in Natural and Artifical Systems: An Introductory Analysis with Application to Biology, Control, and Artifical Intelligence*. Ann Arbor, MI: University of Michigan Press, 1975.

[17] C. A. C. Coello, "Guest editorial: Special issue on evolutionary multiobjective optimization," *IEEE Trans. on Evolutionary Computation*, vol. 7, pp. 97–98, April 2003.

[18] F. Y. Edgeworth, *Mathematical Physics*, London, U.K., 1881.

[19] V. Pareto, *Cours D'Economie Politique*, Lausanne, Switzerland, 1896.

[20] N. I. Cho and S. U. Lee, "Optimal design of finite precision FIR filters using linear programming with reduced constraints," *IEEE Trans. Signal Process.*, vol. 46, no. 1, pp. 195–199, Jan 1998.

[21] D. J. Xu and M. L. Daley, "Design of optimal digital filter using a parallel genetic algorithm," *IEEE Trans. Circuits and Systems*, vol. 42, no. 10, pp. 673–675, Oct 1995.

[22] M. Haseyama and D. Matsuura, "A filter coefficient quantization method with genetic algorithm, including simulated annealing," *IEEE Signal Processing Letters*, vol. 13, pp. 189–192, April 2006.

[23] N. Sulaiman and T. Arslan, "A genetic algorithm for the optimisation of a reconfigurable pipelined FFT processor," in *Proc. Evolvalbe Hardware*, June 2004, pp. 104–108.

[24] M. Leban and J. F. Tasic, "Word-length optimization of LMS adaptive FIR filters," in *Proc. IEEE Mediterranean Electrotechnical Conference*, May 2000, pp. 774–777.

[25] N. Sulaiman and T. Arslan, "A multi-objective genetic algorithm for on-chip real-time optimisation of word length and power consumption in a pipelined FFT processor targeting a MC-CDMA receiver," in *Proc. Evolvalbe Hardware*, June 2005, pp. 154–159.

[26] G. Rohling, "Multiple objective evolutionary algorithms for independent, computationally expensive objective evaluations," Ph.D. dissertation, Georgia Institute of Technology, Atlanta, GA, Nov. 2004.

[27] *Fixed-Point Toolbox User's Guide*, The MathWorks, Inc., Natick, MA, USA. [Online]. Available: http://www.mathworks.com