Digital Filter Design Through Simulated Evolution

Massimiliano Erba*, Roberto Rossi*, Valentino Liberali†, and Andrea G. B. Tettamanzi†

Abstract — **An evolutionary algorithm is used to design a finite impulse response digital filter with reduced power consumption. The proposed design approach combines genetic optimization and simulation methodology, to evaluate a multi-objective fitness function which includes both the suitability of the filter transfer function and the transition activity of digital blocks. The best filter obtained is automatically synthesized in VHDL. A design example is presented and compared with the conventional solution, demonstrating that genetic optimization can help in reducing digital switching power.**

1 Introduction

Although digital filter design methodology is well established, the technological trend in silicon integration is now demanding for new CAD tools, to increase designer productivity and to cope with increased integration density. The larger and larger number of devices, integrated onto a single silicon chip, not only leads to increased computational capability and frequency performance, but also increases power consumption, which is expected to be a major problem for integrated circuit designers in the next decade. Since power consumption is non-linear and input pattern dependent, its minimization is a difficult task.

This paper illustrates an evolutionary approach to the design of digital filters with reduced power consumption. Evolutionary algorithms [1] are a broad class of optimization methods inspired by biology, that build on the key concept of Darwinian evolution [2]. After being successfully applied to physical design (partitioning, placement and routing), now bio-inspired electronic design methods are being considered in a variety of circumstances [3, 4].

In this work, circuits are evolved to compute a finite impulse response (FIR) filtering function. As explained in Section 2, the main objective is the reduction of power consumption through the minimization of transition activity of digital logic. Based on the representation illustrated in Section 3, the evolutionary algorithm in Section 4 is devised to evolve the FIR filter, which is described in VHDL and evaluated through the Synopsys synthesis and simulation tools, in order to obtain an accurate estimate of its power consumption. Section 5 presents a design case, comparing the filter obtained from simulated evolution with a conventional design.

*Department of Electronics, University of Pavia, Via Ferrata 1, 27100 Pavia, Italy.

†Department of Information Technologies, University of Milan, Via Bramante 65, 26013 Crema, Italy.

2 Motivation and Problem Statement

In a digital system, most of the power consumption is due to logic transitions of nodes. It can be expressed as:

$$P = \frac{1}{2} C V^2 f \alpha \tag{1}$$

where C is the total capacitance, V is the power supply voltage, f is the clock frequency, and α is the transition activity, i.e. the average number of logic transitions in a clock period [5]. From (1), it is apparent that a low power design approach must account not only for minimization of area and interconnections (to reduce C), but also for reduction of the transition activity of digital gates. The latter task is not a straightforward one, because digital activity is a non-linear function of input patterns.

For this reason, an evolutionary algorithm based on genetic optimization has been developed, with a twofold target. First of all, the digital filters must be evaluated to check if their frequency response meets the mask requirements (band edges, pass-band ripple, stop-band attenuation). Then, filters meeting the mask constraints are evaluated to estimate their transition activity. The selection is done according to a global fitness function, combining both the mask fitness and the activity fitness.

The best result obtained with simulated evolution is eventually encoded in VHDL and synthesized with Synopsys; the synthesis report provides information on the number of equivalent gates used for the design.

3 Genetic Representation of FIR Filters

The response of a FIR filter in the z-domain is:

$$H(z) = \sum_{k=0}^{N-1} h(k) z^{-k} \tag{2}$$

Digital filters are usually designed by minimizing the number of coefficients, i.e. the impulse response length N, and then by approximating the real coefficients $h(k)$ with finite arithmetic numbers. By replacing multiplication with shifts and additions, a digital filter can be described using a very small number of elementary operations: an example is the set of primitives listed in Table 1. For genetic encoding purposes, each elementary operation is uniquely defined by its own code (one character) and by two integer numbers, which represent the relative offset (calculated backwards from the current position) of the two operands. All primitives include a delay z^{-1}, to avoid possible problems due to timing violations during the synthesis process.

As an example, the following sequence is made of 5 primitives (5 genes):
`(D 1 3) (L 2 2) (A 2 1) (D 1 0) (S 1 4)`
By denoting with x the filter input and by y_i the output of the i-th block, the genetic sequence is interpreted as follows:

$$\begin{aligned} y_0 &= xz^{-1} \\ y_1 &= 2^2 xz^{-1} \\ y_2 &= (y_0 + y_1)z^{-1} \\ y_3 &= y_2 z^{-1} \\ y_4 &= (y_3 - y_0)z^{-1} \end{aligned} \tag{3}$$

The last value is the output of the filter. By merging the equations (3), we obtain the input-output relationship:

$$y = x(5z^{-4} - z^{-2}) \tag{4}$$

Such representation of a filter is equivalent to a computer program in a simple imperative programming language. Therefore we can apply genetic programming [6] to the task of designing FIR filters.

It is worth remarking that the approach presented in this paper differs from other evolutionary methods for digital filter design, where coefficients are found through a genetic optimization [7, 8]. Our approach aims at finding a set of elementary operations which performs the desired filtering function, without imposing any pre-defined structure. Thus, the evolutionary algorithm can perform a better exploration of the design space, finding non-conventional solutions which can be (in some sense) better than conventional ones [9].

4 The Evolutionary Algorithm

To design filters through genetic optimization, an evolutionary algorithm with variable population size has been implemented. The initial population is seeded with two short random individuals. At every generation, selection determines a set of n surviving individuals, which will be used to produce m offspring by crossover and mutation. The selection strategy uses linear ranking with elitism, whereby the best two individuals survive with probability $P_s = 1$, and the individual of rank i has a probability $P_s = 1 - \frac{i}{2\bar{n}}$ of surviving, where $\bar{n}$ is a parameter of the algorithm affecting the average population size.

Reproduction is carried out by a multi-point crossover: two parents are randomly divided into the same number of segments, and a new individual is created by concatenating segments taken with equal probability from either parent. This reproduction strategy produces individuals with variable length. Mutation is applied to all new individuals, and consists in randomly modifying, removing or inserting a gene with a small probability p_{mut}.

Fitness is assigned to individuals as follows: the frequency range is sampled at N equally spaced frequencies ω_i, and the filter response $H(\omega_i)$ is calculated for every ω_i in the pass-band and in the stop-band. The partial error ϵ_i is set to zero, if $|H(\omega_i)|$ lies within the specifications; otherwise it is proportional to the overshoot or undershoot with respect to the given tolerance. The total error E_{tot} is simply the sum of all partial errors. Fitness consists of two parts:

1. mask fitness, defined as $f_M = \frac{1}{1+E_{\text{tot}}}$, measures the extent to which the filter complies with frequency specifications; $f_M = 1$ when specifications are completely met;
2. activity fitness, defined as $f_A = \frac{a}{N_T}$, where N_T is the number of weighted digital transitions per input sample and a is a constant; its contribution is higher for filters with low transition activity, which is responsible for power consumption. The relative weight for digital activity of primitive operators has been determined through gate level simulations, to account also for possible glitches due to carry in additions and subtractions.

Finally, the global fitness is defined as:

$$f = \begin{cases} f_M & \text{if } E_{\text{tot}} > 0 \\ f_M + f_A & \text{otherwise} \end{cases} \tag{5}$$

Table 1: Primitives of the genetic algorithm

Name	Code	Description
Input	`I` u u	$y_i = x$
Delay	`D` n_1 u	$y_i = y_{i-n_1} z^{-1}$
Left shift	`L` n_1 p	$y_i = 2^p y_{i-n_1} z^{-1}$
Right shift	`R` n_1 p	$y_i = 2^{-p} y_{i-n_1} z^{-1}$
Add	`A` n_1 n_2	$y_i = (y_{i-n_1} + y_{i-n_2})z^{-1}$
Subtract	`S` n_1 n_2	$y_i = (y_{i-n_1} - y_{i-n_2})z^{-1}$
Change sign	`C` n_1 u	$y_i = -y_{i-n_1} z^{-1}$

u = unused operand

5 Simulation Results

The proposed design method has been applied to the design of a digital filter to be used as a decimation stage in a $\Sigma\Delta$ analog-to-digital converter. The filter specifications are:

- normalized pass-band: $0 \div 0.25$
- max pass-band ripple: 0.2 dB
- normalized stop-band: $0.75 \div 1$
- min stop-band attenuation: 60 dB
- normalized "don't care" band: $0.25 \div 0.75$

Several simulation were run. Experimental evidence showed that in about 50 % of runs the evolutionary algorithm converges towards a valid solution (i.e. a solution satisfying all mask constraints) within 100,000 generations (approximately 2 h of CPU time on a 550-MHz Pentium III processor).

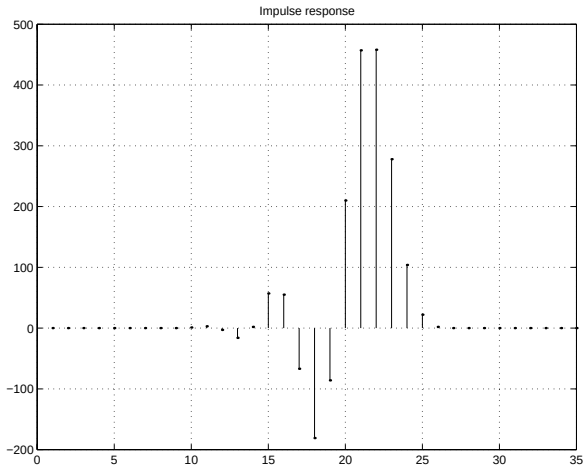

Figure 1: Impulse response of the evolved filter

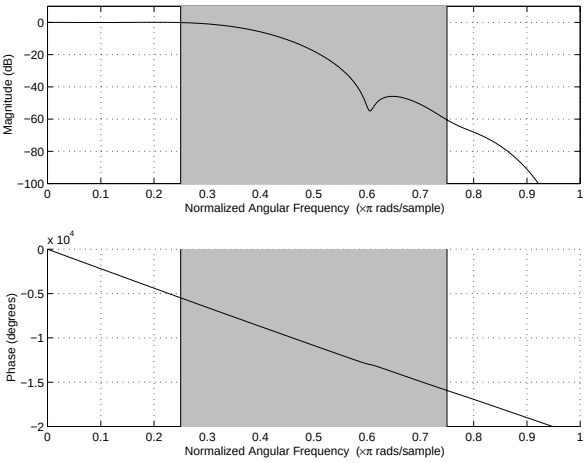

Figure 2: Frequency response of the evolved filter; the grey area represents the "don't care" band

As an example, the genome of one of the filters satisfying all constraints has 46 primitives and corresponds to the impulse response illustrated in Fig. 1. The resulting frequency response is shown in Fig. 2; Fig. 3 contains a detail of the pass-band ripple. It is apparent that the filter meets all design specifications; moreover, the phase is approximately linear, although no phase requirement was considered during the evolutionary optimization.

For comparison, Fig. 4 shows the frequency response of a filter designed using the `remez` function available in Matlab and its Signal Processing Toolbox [10].

Fig. 5 illustrates the VHDL code generated by the algorithm. The RTL architecture is translated by Synopsys into the schematic diagram shown in Fig. 6. After the logic minimization, the evolved filter is implemented with 10,000 equivalent logic gates; while the conventional filter implementa-

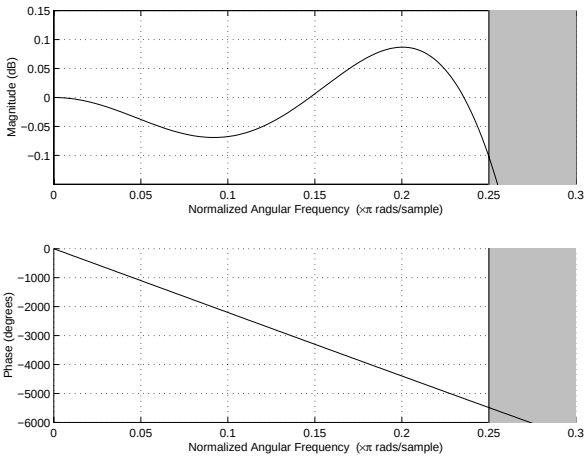

Figure 3: Detail of the pass-band ripple

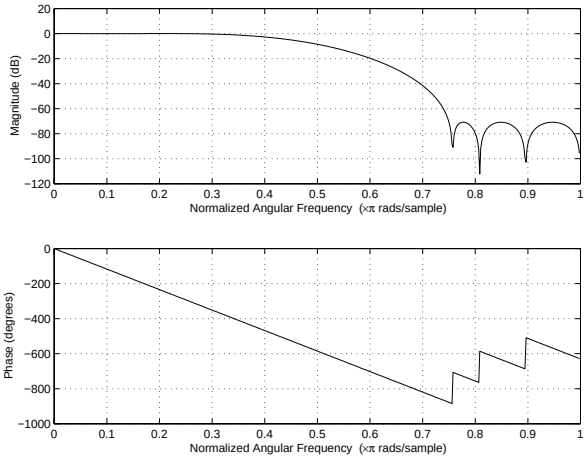

Figure 4: Frequency response of the filter designed with conventional methodology

tion required about 40,000 gates. By comparing these two figures, we can conclude that the effort of the evolutionary algorithm towards optimization of transition activity has led also to a dramatic reduction of the required hardware, thus reducing the silicon area (and hence the cost) by a factor of 4. From simulation with a pseudorandom input pattern, the evolved filter has a power consumption of 14 mW, while the previously designed filter dissipates 40 mW. Table 2 compares the filter obtained through simulated evolution with the conventional design.

6 Conclusion

This paper has described an evolutionary approach to the design of digital filters. Genetic encoding of filter primitives has a fine granularity which is exploited by the evolutionary algorithm during its random search. The encoding is also a straight-

```
entity filter is
port(
  y:out std_logic_vector(29 downto 0);
  clk:in std_logic;
  rst:in std_logic;
  x:in std_logic_vector(15 downto 0)
);
end entity;

architecture RTL of filter is
--filter signals
signal y0: std_logic_vector(16 downto 0);
signal y1: std_logic_vector(16 downto 0);
signal y2: std_logic_vector(15 downto 0);
...
signal y45: std_logic_vector(29 downto 0);

begin
y <= y45;
gene0: adder generic map(Bits_x1=>x'length,
   Bits_x2=>x'length, Bits_y=>y0'length)
   port map(x1=>x, x2=>x, y=>y0,
   clk=>clk, rst=>rst);
gene1: adder generic map(Bits_x1=>x'length,
   Bits_x2=>x'length, Bits_y=>y1'length)
   port map(x1=>x, x2=>x, y=>y1,
   clk=>clk, rst=>rst);
gene2: change_s generic map(Bits=>y2'length)
   port map(x=>x, y=>y2, clk=>clk,
   rst=>rst);
...
gene45: adder generic map(Bits_x1=>y43'length,
   Bits_x2=>y42'length, Bits_y=>y45'length)
   port map(x1=>y43, x2=>y42, y=>y45,
   clk=>clk, rst=>rst);
end RTL;
```

Figure 5: VHDL synthesizable code generated by the algorithm

forward representation of the impulse response of the filter, thus allowing a direct evaluation of cost and transition activity of digital blocks. A fitness function has been devised to allow multi-objective evolution. The "best" result produced by the algorithm is automatically translated into VHDL code, which can be synthesized into a circuit without any additional operation, according to the standard digital design methodology.

The results obtained with the simulated evolution show that minimization of transition activity leads to a dramatic reduction of the hardware with respect to the conventional design methodology, while maintaining the same filter performance.

Acknowledgment

This work was supported by ESPRIT Project 29261 – MIXMODEST.

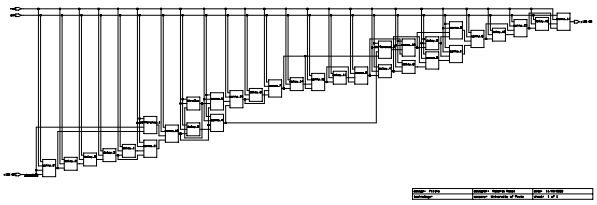

Figure 6: Schematic diagram of the filter synthesized with Synopsys

References

[1] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, Oxford, UK, 1996.

[2] C. Darwin. *On the Origin of Species by Means of Natural Selection*. John Murray, London, UK, 1859.

[3] R. Drechsler. *Evolutionary Algorithms for VLSI CAD*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[4] M. Sipper, D. Mange, and E. Sanchez, "Quo vadis evolvable hardware?", *Communications of the ACM*, vol. 42, no. 4, pp. 50–56, Apr. 1999.

[5] M. Pedram, "Power minimization in IC design: Principles and applications", *ACM Trans. on Design Automation of Electronic Systems*, vol. 1, pp. 3–56, Jan. 1996.

[6] J. R. Koza. *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, MA, USA, 1993.

[7] D. Suckley, "Genetic algorithm in the design of FIR filters", *IEE Proceedings – Part G*, vol. 138, no. 2, pp. 234–238, Apr. 1991.

[8] K. Uesaka and M. Kawamata, "Synthesis of low-sensitivity second-order digital filters using genetic programming with automatically defined functions", *IEEE Signal Processing Lett.*, vol. 7, no. 4, pp. 83–85, Apr. 2000.

[9] A. Thompson, P. Layzell, and R. S. Zebulum, "Explorations in design space: Unconventional electronics design through artificial evolution", *IEEE Trans. Evolutionary Computation*, vol. 3, no. 3, pp. 167–196, Sept. 1999.

[10] The Mathworks, Inc. *Signal Processing Toolbox*. Natick, MA, USA, 1983.

Table 2: Filter characteristics (after synthesis)

	evolutionary	conventional
No. of coefficients	17	13
No. of primitives	32	60
No. of logic gates	10,000	40,000
Power consumption	14 mW	40 mW