An Evolutionary Approach to Automatic Generation of VHDL Code for Low-Power Digital Filters

Massimiliano Erba[1], Roberto Rossi[1], Valentino Liberali[2], and Andrea G.B. Tettamanzi[2]

[1] Università degli Studi di Pavia
Dipartimento di Elettronica
Via Ferrata 1, 27100 Pavia, Italy
erba@ele.unipv.it, roberto@ele.unipv.it
[2] Università degli Studi di Milano
Dipartimento di Tecnologie dell'Informazione
Via Bramante 65, 26013 Crema, Italy
vliberali@crema.unimi.it, tettaman@dsi.unimi.it

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 proper choice of fitness function and selection criteria allows the genetic algorithm to perform a better search within the design space, thus exploring possible solutions which are not considered in the conventional structured design methodology. Although the evolutionary process is not guaranteed to generate a filter fully compliant to specifications in every run, experimental evidence shows that, when specifications are met, evolved filters are much better than classical designs both in terms of power consumption and in terms of area, while maintaining the same performance.

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 power and increased 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 minimisation 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

J. Miller et al. (Eds.): EuroGP 2001, LNCS 2038, pp. 36–50, 2001.

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]. Evolutionary algorithms for circuit synthesis are a powerful technique, which could provide innovative solutions to hard design problems, also when classical decomposition methods may fail [5].

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 consuption through the minimization of transition activity of digital logic. Based on the behavioral description illustrated in Section 3, the evolutionary algorithm described 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.

2 Motivation and Problem Statement

Microintegrated technology enables the development of deep submicron CMOS circuits for digital signal processing (DSP) in a broad variety of applications. Low-cost and high-performance circuits are required to be competitive in a more and more demanding consumer market. Single chip solutions reduce costs, increase reliability, and can help to reduce weight and size of hand-held electronic products.

Being the real world steadily analog, interfaces between the digital processor and external variables must be used. This leads to mixed analog-digital integrates systems, including the DSP, the analog-to-digital and digital-to-analog conversion, and, in some cases, analog pre-processing and/or signal conditioning. In mixed systems, performance limitations come mainly from the analog section which interfaces the digital processing core with the external world. In such ICs, analog device performance can be severely affected by disturbances coming from switching digital nodes [6]. Therefore, low power digital design is a must, since it also helps to reduce the disturbance generated by the digital section of the circuit.

The power consumption P in a digital system is mainly due to the logic transitions of circuit nodes, and can be expressed as:

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

where C is the node 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 [7].

From (1), it is apparent that a low power design approach must account not only for minimisation of area and interconnections (to reduce C), but also for reduction of the transition activity of digital gates. The latter task is not

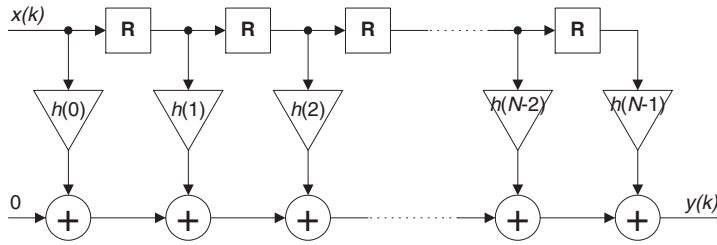

Fig. 1. Canonical direct form of a FIR filter

a straightforward one, because digital activity is a non-linear function of input patterns.

For this reason, an evolutionary algorithm based on genetic optimisation has been developed, with a twofold target. First of all, the digital filters randomly designed 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 the 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

A FIR filter implements a time-domain convolution between the input signal $x(k)$ and the filter impulse response, represented by the set of constant filter coefficients $h(k)$:

$$y(k) = \sum_{i=0}^{N-1} h(i)x(k-i) \tag{2}$$

The "canonical direct form" shown in Fig. 1 is a straightforward implementation of (2) [8]. It requires a set of $(N-1)$ registers (indicated with R in the figure) to store the previous samples of the input signal $x(k)$, a set of multiplier blocks to perform the multiplications of the (delayed) input samples and the filter coefficients, and an adder tree to obtain the weighted sum at the output $y(k)$.

Taking the $\mathcal{Z}$-transform of both input and output signals, we can write the frequency response of the filter in the z-domain:

$$H(z) = \frac{y(z)}{x(z)} = \sum_{i=0}^{N-1} h(i)z^{-i} \tag{3}$$

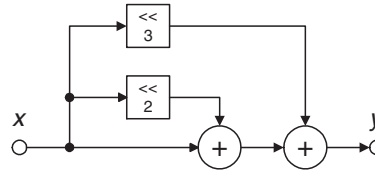

Fig. 2. Multiplication by 13 implemented with shifters and adders

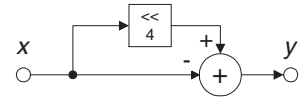

Fig. 3. Multiplication by 15 implemented with one shifter and one subtractor

which is still represented in the graphical form shown in Fig. 1, as the z-domain response of a register is z^{-1} [9].

To save area and to reduce power consumption, multiplier blocks are often replaced with shifters and adders [10]. As an example, multiplication by 13 can be implemented with two shifts and two additions, as illustrated in Fig. 2, where the block "$<< n$" means "left shift by n positions".

The canonical signed digit (CSD) representation [11] assigns a separate sign to each digit: 0, 1 and $\bar{1}$ (= –1). Its goal is to minimise the number of non-zero digits: by encoding the filter coefficients with CSD, the filter ouput can be computed using a reduced amount of hardware, since multiplications by zero are simply not implemented. As an example, consider the multiplication by 15: since $15 = 2^3 + 2^2 + 2^1 + 2^0 = (001111)_2$, this operation in binary arithmetics would require three shifts and three additions; while using CSD we can write $15 = 2^4 - 2^0 = (0100\bar{0}\bar{1})_2$, and we implement the same operation using only one shifter and one subtractor (Fig. 3).

Starting from these considerations, a digital filter can be described using a very small number of elementary operations, or *primitives*. All the operations can be arranged into a linear sequence, with the input signal $x(k)$ entering into the leftmost primitive ant the output signal $y(k)$ being the output of the rightmost primitive. Since a FIR filter does not have any feedback loop, the signal flow is always from left to right.

The primitives selected for FIR filters are listed in Table 1. Each elementary operation is encoded 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 at the input. All primitives include a delay z^{-1}, to avoid possible problems due to timing violations during the synthesis process. The sequence of the encoded elementary operations is the genotype of the digital filter.

Table 1. Primitives of the genetic algorithm

Name	Code	Op 1	Op 2	Description
Input	`I`	not used	not used	Copy input: $y_i = x$
Delay	`D`	n_1	not used	Store value: $y_i = y_{i-n_1} z^{-1}$
Left shift	`L`	n_1	p	Multiply by 2^p: $y_i = 2^p y_{i-n_1} z^{-1}$
Right shift	`R`	n_1	p	Divide by 2^p: $y_i = 2^{-p} y_{i-n_1} z^{-1}$
Adder	`A`	n_1	n_2	Sum: $y_i = (y_{i-n_1} + y_{i-n_2}) z^{-1}$
Subtractor	`S`	n_1	n_2	Difference: $y_i = (y_{i-n_1} - y_{i-n_2}) z^{-1}$
Complement	`C`	n_1	not used	Multiply by -1: $y_i = -y_{i-n_1} z^{-1}$

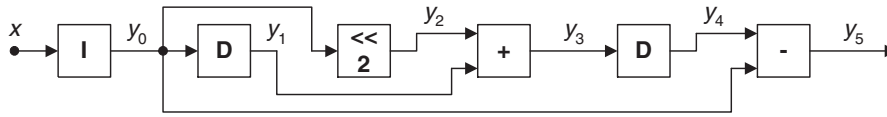

Fig. 4. Example of a signal flow graph in linear form

As an example, the following sequence is made of 6 primitives (6 genes):
`(I 0 2) (D 1 3) (L 2 2) (A 2 1) (D 1 0) (S 1 5)`
It can be represented in the graphical linear form illustrated in Fig. 4 and it is interpreted as follows:

$$\begin{aligned} y_0 &= x \\ y_1 &= y_0 z^{-1} \\ y_2 &= 2^2 y_0 z^{-1} \\ y_3 &= (y_1 + y_2) z^{-1} \\ y_4 &= y_3 z^{-1} \\ y_5 &= (y_4 - y_0) z^{-1} \end{aligned} \tag{4}$$

The last value is the output of the filter. By merging the equations (4), we obtain the input-output relationship:

$$y = x(5z^{-4} - z^{-1}) \tag{5}$$

We note that such representation of a filter has the same essence as a program in a simple imperative programming language, and we can apply genetic programming [12] or, more precisely, Cartesian genetic programming [13] to the task of designing FIR filters.

The design method proposed is this paper is completely different from the conventional design approach.

Classical filter designs are based on the minimization of a norm in the design space, e.g. on the determination of the minimum number of filter coefficients that allow to obtain a filter response within the given specifications. Such an appoach reduces the number of filter coefficients (the "taps" in Fig. 1), but it does not guarantee that the digital network is "optimal" in some sense at the end of the design process.

On the other hand, the evolutionary design approach uses a fine granularity of the primitives and, hence, of the filter structure. From the structure shown in Fig. 4), filter coefficients cannot be directly seen. The advantages of this approach is the simple genetic encoding, that allows the evolutionary algorithm to perform a fine search within the design space, exploring also possible solutions which are simply not considered in the classical method [14]. By taking into consideration a larger design space, we have a higher chance of finding a solution which is (in some sense) "better" than the conventional one.

Since the main target of this work is the reduction of power consumption, the evolutionary algorithm will have to estimate the transition activity of the filter. To this end, primitives have been characterized to evaluate their average power consumption. According to gate level simulations, we assigned a relative weight $W = 30$ to sum, difference and complement.

4 The Evolutionary Algorithm

In order to make filters evolve, a variable population size evolutionary algorithm using an $(n + m)$ strategy and linear ranking selection with elitism 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. Four alternative selection strategies have been considered:

1. an "aging" rule, which assigns a life expectation to every individual at its birth, according to its fitness: life expectation linearly decreases from 100 to 1 generation as the fitness goes from best (f_{best}) to worst (f_{worst});
2. a selection based on normalized fitness and ranking, whereby each individual has a probability $P_s = k\sqrt{f'_i r'_i}$ of surviving, where f'_i is the normalized fitness of the individual of rank i, defined as $f'_i = \frac{f_i - f_{\text{best}}}{f_{\text{best}} - f_{\text{worst}}}$, r'_i is the normalized rank, defined as $r'_i = 1 - \frac{i}{N}$, N is the actual population size, and k is a parameter dynamically calculated to adjust the population size;
3. a sort of linear ranking selection 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 the average population size;
4. a histogram based selection, whereby the fitness histogram is calculated by dividing the fitness interval $[0, f_{\text{best}}]$ into B bins (with $2 \leq B \leq 50$, B depending on the population size), and the survival probability of each individual is $P_s = 1$ if the individual lies into a bin less populated than the

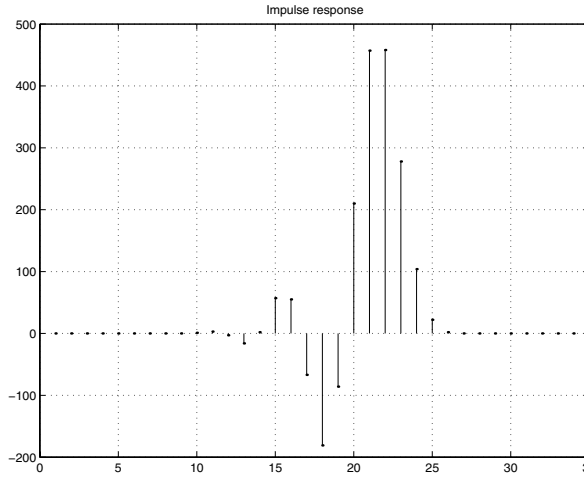

Fig. 5. Impulse response of the best filter generated (pass band: 0 ... 0.25)

average, otherwise $P_s < 1$ if it lies into a bin exceeding the average bin population density.

The four selection strategies greatly differ in promoting genetic diversity: from experimental evidence, strategy 1 tends to produce a population of individuals identical to the best one, while strategy 4 favours the highest diversity. Strategies 2 and 3 have an intermediate behaviour.

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 catenating 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 $f_A = 1 + \frac{a}{N_T}$, 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 a low transition activity, which is responsible for power consumption.

Table 2. Success of different selection strategies

		faster result within specifications	
Selection strategy	% of success	generation number	CPU time
1. (aging)	45 %	11109	51 s
2. (norm. fitness and ranking)	50 %	46372	210 s
3. (linear ranking)	33 %	4269	63 s
4. (histogram)	25 %	3646	445 s

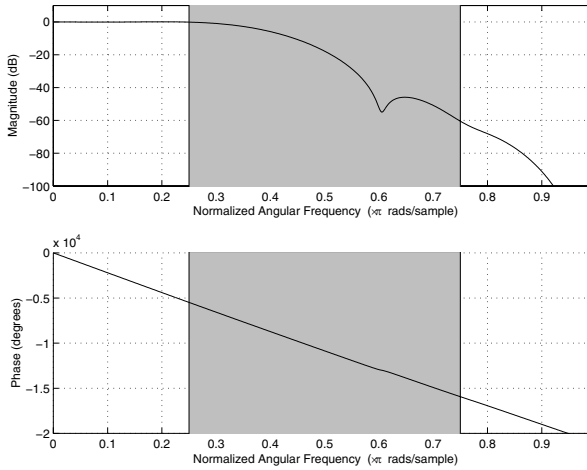

Fig. 6. Frequency response of the best filter generated (pass band: 0 ... 0.25); the gray area represents the "don't care" band

Finally, the global fitness is defined as:

$$f = \begin{cases} f_M & \text{if } E_{\text{tot}} > 0 \\ f_M + f_A & \text{otherwise} \end{cases} \tag{6}$$

Such definition introduces a gap between individuals meeting the specifications and individuals not meeting them. The main function of this gap is just to avoid numerical errors during the ranking for selection or ambiguity during histogram calculation: the gap ensures that the individuals complying with mask specifications are not in the same bin with individuals not meeting them.

5 Experiments and 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:

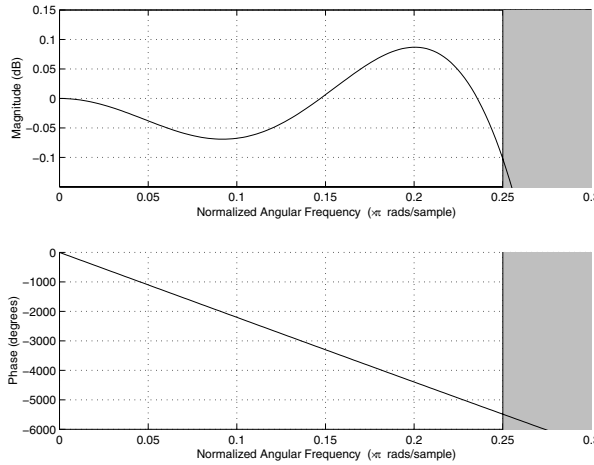

Fig. 7. Detail of the pass-band ripple (pass band: 0 ... 0.25)

- normalized pass-band: 0 ... 0.25
- pass-band maximum ripple: 0.2 dB
- normalized stop-band: 0.75 ... 1
- pass-band minimum attenuation: 60 dB
- normalized "don't care" band: 0.25 ... 0.75

The parameters of the evolutionary algorithm were set as:

- minimum population size: $n_{\min} = 10$
- maximum population size: $n_{\max} = 1000$
- crossover probability: $p_{\text{cross}} = 0.05$
- mutation probability: $p_{\text{mut}} = 0.01$

From these parameters, the average population size is $\bar{n} = 100$ for selection strategies 1, 2, and 3, while selection stategy 4 always maximizes the population size to achieve the maximum genetic diversity.

Several runs have been done with the same parameter set and different selection strategies. Runs are considered successful if they produce a filter design complying with specifications within 100,000 generations (10,000 generations for strategy 4, as the population size is ten times higher). Table 2 summarizes the results, indicating the percentage of successful runs and the number of generations and the CPU time of the first result obtained (which is not necessarily the best one among all the runs. Each line summarizes at least ten different runs.

The successful designs produced by the evolutionary algorithm are automatically encoded in VHDL. In the VHDL description, the algorithm calculates automatically the length of digital words, in order to avoid arithmetic overflow while saving area and power. Filters described in VHDL can be synthesized and simulated according to the conventional digital design methodology.

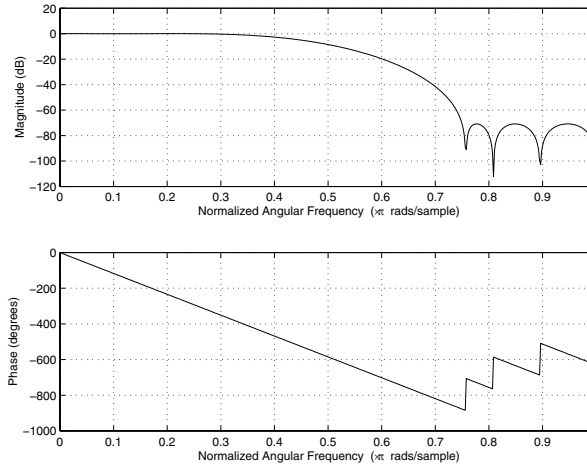

Fig. 8. Frequency response of the filter designed with conventional methodology (pass band: 0 ... 0.25)

In the following part of this section, we consider two design cases. Both are low-pass filters: the first one with normalized pass band (0, 0.25) and normalized stop band (0.75, 1), and the second one with normalized pass band (0, 0.35) and normalized stop band (0.65, 1).

5.1 Low-Pass Filter with Normalized Pass Band (0, 0.25)

The filter specifications are listed at the beginning of Sect. 5.

The genome of the evolved filter has 46 primitives and corresponds to the impulse response illustrated in Fig. 5. The resulting frequency response is shown in Fig. 6; Fig. 7 contains a detail of the pass-band ripple. It is apparent that the filter meets all design specifications; also the phase is linear within the pass band, although the linear phase requirement was not considered during the evolutionary process.

For comparison, Fig. 8 shows the frequency response of a filter designed using the `remez` function available in Matlab and its Signal Processing Toolbox [15].

Fig. 9 illustrates the VHDL code generated by the algorithm. The RTL architecture is translated by Synopsys into the schematic diagram shown in Fig. 10. The evolved filter is implemented with 10,000 equivalent logic gates; while the conventional filter implementation required about 40,000 gates. By comparing these two figures, we can conclude that the effort of the evolutionary algorithm towards optimisation 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 pseudorandom input pattern, the evolved filter has a power consumption of 14 mW, while the previously designed filter dissi-

```
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 y3: std_logic_vector(17 downto 0);
signal y4: std_logic_vector(15 downto 0);
signal y5: std_logic_vector(18 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);
gene3: adder generic map(Bits_x1 => y1'length, Bits_x2 => y0'length,
       Bits_y => y3'length)
       port map(x1 => y1, x2 => y0, y => y3, clk => clk, rst => rst);
gene4: init generic map(Bits => x'length)
       port map(x => x, y => y4, clk => clk, rst => rst);
gene5: adder generic map(Bits_x1 => y3'length, Bits_x2 => y2'length,
       Bits_y => y5'length)
       port map(x1 => y3, x2 => y2, y => y5, 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;
```

Fig. 9. Synthesizable VHDL code generated by the algorithm

Table 3. Filter characteristics (from post-synthesis simulation)

	evolutionary	conventional
No. of coefficients	17	15
No. of primitives	32	60
No. of logic gates	10,000	40,000
Power consumption	14 mW	40 mW

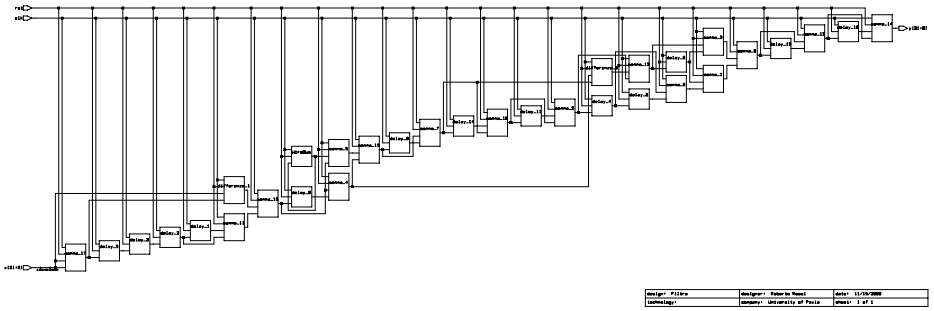

Fig. 10. Schematic diagram of the filter synthesized with Synopsys (pass band: 0 ... 0.25)

pates 40 mW. Table 3 compares the filter obtained through simulated evolution with the conventional design.

The CPU time required for the evolution was about 300 s.

Fig. 11 illustrates the evolution of the fitness of the best individual, for which the selection strategy 1 (aging) has been used. From the figure, one can observe the typical punctuated equilibria where abrupt changes are interleaved with long periods of stasis.

5.2 Low-Pass Filter with Normalized Pass Band (0, 0.35)

The filter specifications for the second design case are:

- normalized pass-band: 0 ... 0.35
- pass-band maximum ripple: 0.2 dB
- normalized stop-band: 0.65 ... 1
- pass-band minimum attenuation: 60 dB
- normalized "don't care" band: 0.35 ... 0.65

The genome of the evolved filter has 254 primitives and corresponds to the impulse response illustrated in Fig. 12. The resulting frequency response is shown in Fig. 13. It is apparent that also this design example meets all specifications. The phase is linear within the pass band.

The CPU time required for this design was about 3000 s.

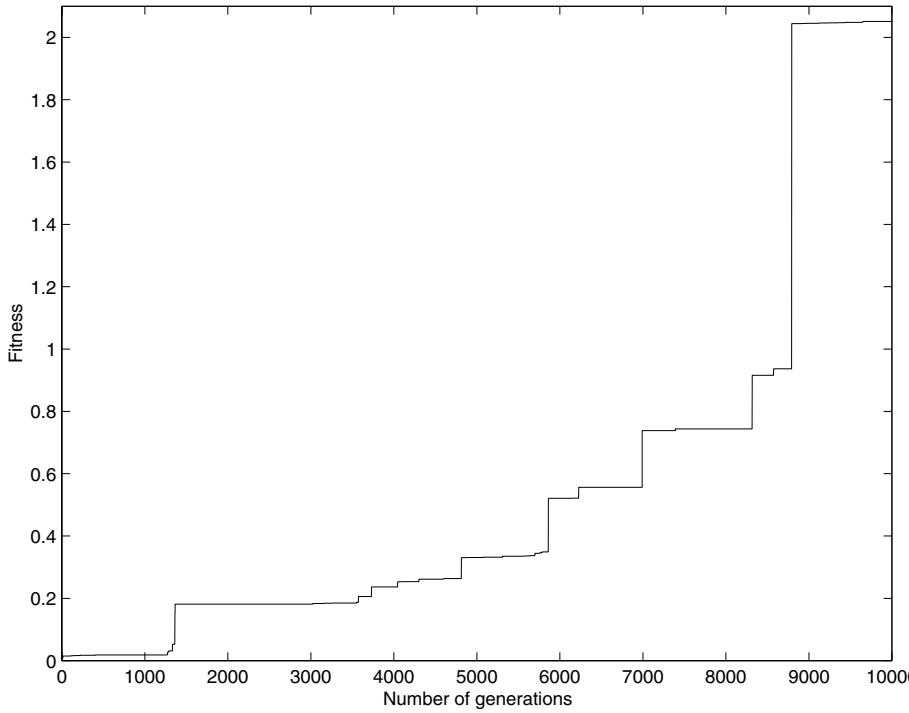

Fig. 11. Evolution of the fitness of the best filter (pass band: 0 ... 0.25)

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 search. From the genetic encoding, the impulse response of the filter can be easily derived, thus allowing a direct evaluation of cost and transition activity of digital blocks. A fitness function has been devised to allow multi-objective evolution. Four selection criteria have been considered: none of them can be considered optimal, since experimental evidence shows that the evolution process may remain stuck for long time into a local maximum, without reaching the design target within a defined number of iterations. Therefore, multiple runs are required to obtain a high probability of success.

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 minimisation of transition activity leads to a dramatic reduction of the hardware with re-

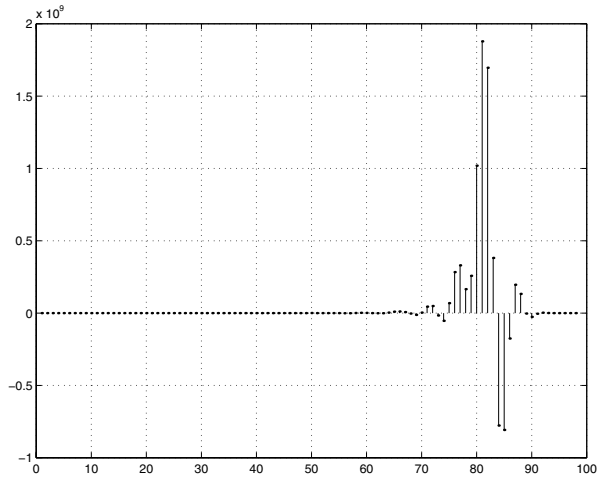

Fig. 12. Impulse response of the best filter generated (pass band: 0 ... 0.35)

spect to the conventional design methodology, while maintaining the same filter performance.

Acknowledgments

This work was supported by ESPRIT Project 29261 – MIXMODEST. The authors wish to thank the anonymous reviewers for their valuable comments.

References

1. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, Oxford, UK (1996)
2. Darwin, C.: On the Origin of Species by Means of Natural Selection. John Murray, London, UK (1859)
3. Drechsler, R.: Evolutionary Algorithms for VLSI CAD. Kluwer Academic Publishers, Dordrecht, The Netherlands (1998)
4. Sipper, M., Mange, D., Sanchez, E.: Quo vadis evolvable hardware? Communications of the ACM **42** (1999) 50–56
5. Thompson, A., Layzell, P.: Analysis of unconventional evolved electronics. Communications of the ACM **42** (1999) 71–79
6. Liberali, V., Rossi, R., Torelli, G.: Crosstalk effects in mixed-signal ICs in deep submicron digital CMOS technology. Microelectronics Journal **31** (2000) 893–904
7. Pedram, M.: Power minimization in IC design: Principles and applications. ACM Trans. on Design Automation of Electronic Systems **1** (1996) 3–56
8. Jackson, L. B.: Digital Filters and Signal Processing. Kluwer Academic Publishers, Dordrecht, The Netherlands (1986)

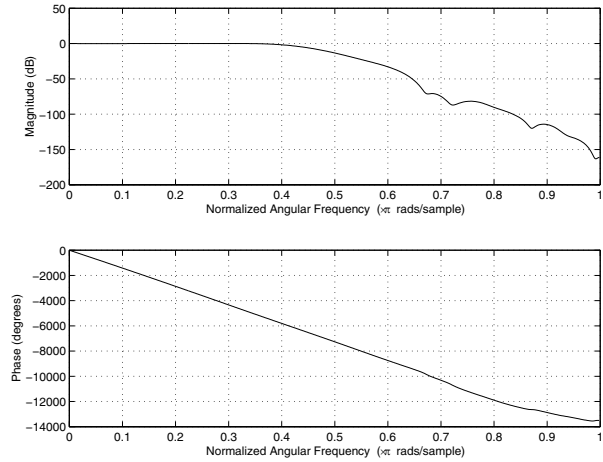

Fig. 13. Frequency response of the best filter generated (pass band: 0 ... 0.35)

9. Jackson, L. B.: Signals, Systems, and Transforms. Addison-Wesley, Reading, MA, USA (1991)
10. Zhao, Q., Tadokoro, Y.: A simple design of FIR filters with power-of-two coefficients. IEEE Trans. Circ. and Syst. **35** (1988) 556–570
11. Pirsch, P.: Architectures for Digital Signal Processing. John Wiley & Sons, Chichester, UK (1998)
12. Koza, J. R.: Genetic Programming: on the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge, MA, USA (1993)
13. Miller, J. F., Thomson, P.: Cartesian genetic programming. In Poli, R. et al. (Eds.), Genetic Programming European Conference (EuroGP 2000), Springer-Verlag, Berlin, Germany (2000) 121–132
14. Thompson, A., Layzell, P., Zebulum, R. S.: Explorations in design space: Unconventional electronics design through artificial evolution. IEEE Trans. Evolutionary Computation **3** (1999) 167–196
15. The Mathworks, Inc.: , Signal Processing Toolbox. Natick, MA, USA (1983)