Evolutionary FIR filter

Petr Burian

University of West Bohemia in Pilsen, Department of Applied Electronics and Telecommunications
Univerzitni 26, 30614 Pilsen, Czech Republic
burianp@kae.zcu.cz

Abstract – **This paper deals with the design of the FIR filter whose parameters are obtained using an evolutionary algorithm. It explores usage problems of evolutionary algorithms in adaptive devices domain, then discusses implementation of the adaptive FIR filter prototype based on the standard genetic algorithm realized by FPGA circuit, and continues by discussion of advantages and disadvantages of such an implementation.**

I. INTRODUCTION

Evolutionary computational techniques (evolutionary algorithms) are profitable tools for solution of optimization tasks. They are based on the Darwin's theory of evolution and survival of the fittest, and make possible to solve the optimization tasks that can be defined as a search for global/local maximum/minimum functions.

The basic representative of this group of algorithms is the Standard Genetic Algorithm. It is a relatively simple algorithm whose base is an individual. A group of individuals forms a population. An individual represents just one solution of the optimization problem. Various representations can be used for individuals, for example the classical binary format is used very often. Note that a suitable version of representation is very important for correct function of the evolution. At first, the algorithm has to initiate individuals of a population, which is realized by filling them with random values. Afterwards, recombination operators are applied. Generally, only two operators – the crossover and the mutation – are used. These operators change genetic information of the individuals, and thereby the very solution of the task which is represented by the individuals. After recombination the algorithm has to evaluate the solutions that are represented by the individuals. The fitness function provides this evaluation: it tells us how quality the solution contained in an individual is. If the required solution is found, the algorithm can be terminated. If not, the algorithm will select individuals for a new population and the whole process repeats. We can say that a new generation begins. The algorithm can be terminated if a certain number of generations is achieved. [1]

These algorithms are also used in the design of digital or analog circuits. They look for topology or circuit parameters. In case of digital filters, the algorithm can search for suitable parameters of the filter.

II. FIR FILTER

The FIR (finite impulse response) filter is one of the basic types of digital filters. Its function is based on convolution. The FIR filters excel in simple structure and stability. They consist of a shift register, multipliers and adders. The shift register accumulates input data samples that are multiplied by parameters (the impulse response) of the filter. Afterwards, these multiples are counted up and the consequent result creates the filter output. [2]

Filter parameters determine its kind and its amplitude characteristic. If the impulse response is changed by the evolution, the features of the filter are changed too. The FIR filter is always stable. For that reason, this type of filter is suitable to determine the impulse response by means of evolutionary techniques.

In this project the symmetric FIR filter with 29 taps is used. The taps of odd numbers are preferable if we want to generate various kinds of filters. Each parameter is represented by 8 bits in the two's complement. In addition, structures of the FIR filter can be very simply implemented by FPGA circuit. The FIR filter also has to be able of quick reconfiguration if we want to use it for cooperation with the evolutionary algorithm.

III. AIM OF THE PROJECT

The main goal of this project is the design of the evolutionary FIR filter. Parameters (the impulse response) of this filter are obtained by the evolution. It means that no operator intervention is needed for the correct function of the filter. The standard FIR filter has one data input and one data output. In addition, the evolutionary filter will have a special data input – the input for ideal output samples. The evolution will have to find a suitable impulse response that ensures a correspondence between the output signal and ideal output signal. That is why the filter will have an adaptive character.

IV. EVOLUTIONARY ALGORITHM

In this project, the Standard Genetic Algorithm was used. It is clear that standard version of this algorithm might not be optimal for this type of application; however, only a detailed testing and analyses can show its suitability.

The algorithm uses 16 individuals. Each individual consists of 15 parameters of the FIR filter. It is: 15 x 8 bits = 120 bits, then 4 bits for the control of the filter output and 4 bits for the reserve are used.

On the whole, the individual consists of 128 bits (16 bytes). If an offspring (max. 16), mutants (max. 16) and a former generation (16) are taken into account, a memory for 64 (16 x 4) individuals (1024 bytes) is needed. Only 15 parameters are in individuals. However, the FIR filter can be created by 30 (only 29 are used in this project) parameters, because the filter is symmetric. The parameters are stored in two's complement. The initialization of a population is made so that individuals are filled by random values.

This algorithm uses only a primitive one-point crossover. The crossover depends on the probability of the crossover – P_c. Two parents participate in the crossover and two children result from it.

The situation with the mutation is more difficult as the mutation can be implemented in many ways. In the project the following method of mutation is used: At first, an individual is divided into single bytes (parameter of the FIR filter). Afterwards, the generator of random numbers generates 11-bits random number. First 8 bits determine (according to the probability of the mutation - P_m) whether a certain byte will undergo the mutation. Remaining 3 bits determine position of the bit for the mutation. The mutation of the bit means its negation. All individuals of the population undergo the mutation duly. It is very profitable if the probability of mutation can change value within a run of the algorithm.

For the selection of a new population, the algorithm utilizes a principle of tournament. Members are selected from the former population, offspring and mutants. The selection process picks out two random individuals, and the individual with a higher value of fitness is chosen for the new population. However, this principle does not guarantee that the individual added to the new generation will be the best one; hence, this selection is extended by elitism. Elitism is a technique which ensures moving of the best individual (the leader) into the new population.

The most difficult part of the algorithm is the fitness function.

V. ISSUES OF THE DYNAMIC FITNESS FUNCTION

As it has already been noted, the fitness function is a key element of the evolutionary algorithm. This function expresses quality of a found solution of the optimization task. The fitness function can be classified into two categories: it can be static or dynamic. The kind of the function depends on a concrete application. For example, in case of the evolvable combinational logic circuit, we use the static fitness function. The combinational logic circuit can be described by a truth table. The aim of the optimization is therefore evident already at the beginning of the evolution. For that reason, the fitness function is invariant throughout the evolution. However, if adaptive behaviour is required, the fitness function must be time variable because the application has to react dynamically on the environment variable. In case of the FIR filter, the function has to react on the actual input data of the filter. This fact affects the implementation of the evolvable component. Evolution in the environment variable runs de facto continually; in contrast to the static environment, where the evolution can be terminated if a sufficient solution is found. [3]

VI. FITNESS

It is necessary to modify the algorithm and its fitness function so that the algorithm can work with the environment variable. The dynamic fitness function has to transform into the static function; then the algorithm works with the static fitness function, and the adaptive character is conserved at the same time. Suitable transformation into the static function ensures that all individuals within one generation have the same evaluative criteria.

Work with dynamic fitness function is solved by means of a special sample memory – the samples unit. Principle of this unit is based on shift register. The samples unit accumulates samples of the input and ideal output signal. If the fitness function is started, last 200 samples (200 input samples and 200 ideal output samples) from the sampler are stored in work registers in a fitness module. The valuation of the whole population within one generation proceeds with the same samples of signals; this way equal conditions for all members of the population are ensured. The dynamic fitness function and adaptive character of the application require other changes in the algorithm. The evaluation of individuals of the former population (generation) is needed. It is a significant difference against the static fitness function. The individuals which were not changed will have to be evaluated too, because parameters of the fitness function can be different in comparison with last evaluation.

The concrete fitness function of the project is based on the method of least squares. The filter in the fitness module processes the input signal (from the samples unit). Afterwards, the differences between the filter output and ideal output signal are calculated. The sum of these differences creates resultant values of fitness. The best individual has the smallest value of fitness.

The required adaptive character of the application also demands changes in the structure (fig. 1) of evolvable system. There have to be minimally two FIR filters in the system: the first used for calculation of the fitness function, the second one serving to the processing of the input signal. The latter is configured by the best individual (the leader). Two filters in the system allow the correct function of the evolutionary algorithm and processing of the input signal at the same time.

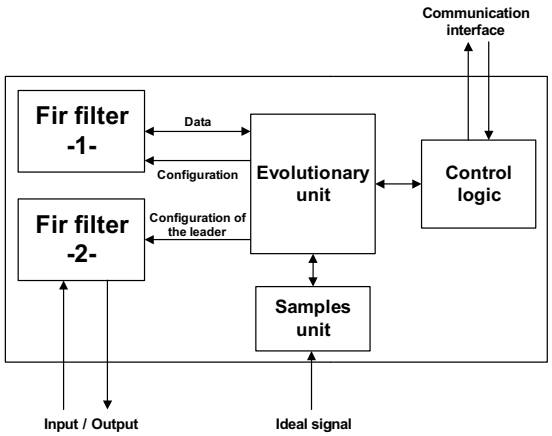

Figure 1: The structure of the evolvable system

VII. IMPLEMENTATION

The whole system was implemented by the FPGA Cyclone II circuit. Audio codec Wolfson WM8731 was used for an input and an output of the analog signals. The system is created by means of the VHDL language, and drawn as a periphery for the Avalon bus. This conception is very profitable for the debugging and the testing. We can very easily observe and control the process of the algorithm by the NIOS II processor and its J-TAG console. However, in the created logic the algorithm runs very quickly.

For storage of the individuals, offspring and mutants an embedded memory is exploited. This memory allows very fast data operations. As noted previously, 1024 bytes of this memory are needed for all individuals. The memory for the values of the fitness is needed too. Every value of fitness needs 4 bytes (32 bits). In sum, we need 256 bytes of the memory for the fitness.

Several modules (fig. 2) perform the main operations of the standard genetic algorithm. They are called: crossover, mutation, selection, fitness and elitism. These modules are controlled by means of the control register. The modules are connected with the memory by the multiplexer. This multiplexer is controlled by the control register, too. Therefore, this register controls the whole process of the evolution. It is possible to read the actual fitness value of the leader.

All modules are optimized so that access into the memory is exploited effectively. The pipeline structures are used in the modules. For example: the process of the crossover (the creation maximum number of the offspring - 16) needs only 21 cycles of system clock, the process of the mutation (the mutation of 256 parameters) needs only 275 cycles of system clock. However, the calculation of the fitness function is very time-consuming. This function needs 11 182 cycles (the valuation of 48 individuals – former population, mutants and offspring). Naturally, it is possible to reduce this time-consumption: we can use fewer samples for fitness function or exploit some parallel structures. One generational cycle needs roughly 12 000 system cycles. If system clock is 100 MHz, the performance of the evolution is approximately 8 300 generations per second.

The system also includes the generator of pseudo random numbers. The generator is created by the 32-bits linear feedback shift register with the polynomial $x^{32} + x^7 + x^6 + x^2 + x^0$. This generator is designed so that a random number is generated in one cycle. [4]

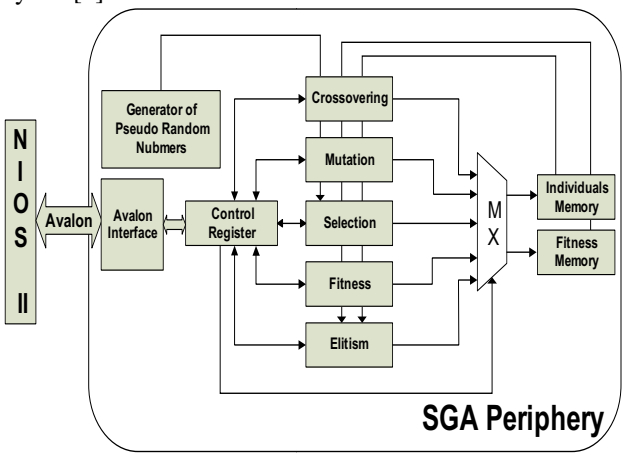

Figure 2: The Standard Genetic Algorithm Periphery

VII. TESTING

For testing and verification of the system function the simulation of the perturbing influence on useful signal was used. The interference signal was superimposed to the useful signal. The useful signal is used as ideal output signal. The task of the evolutionary FIR filter is to eliminate the interference signal. It is surprising that the filter provides good results after only few generational cycles. It is important to note that this implementation is the first prototype of the system, and after a future development better results might be expected.

On the next figures you can see an example of the performance of the filter. One signal with frequency 1 kHz and another with frequency 10 kHz were regarded as the useful signal. Signals with frequency 5 kHz and 15 kHz were superimposed on this signal. The useful signal was termed as an ideal filter output. The sample frequency 48 kHz was used for the testing. The good working of the filter is detectable already in the 6th generation, and in the next generations results goes on to improve. In the 1000th generation we can see that attenuation of "the interference signals" is around 42dB (5 kHz) and 43 dB (15 kHz). When interpreting the results let´s take into account possibilities of the FIR filter with 29 taps.

The progression of the evolution:

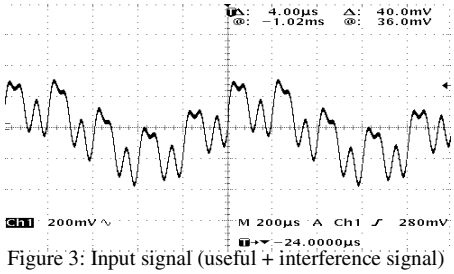

Figure 3: Input signal (useful + interference signal)

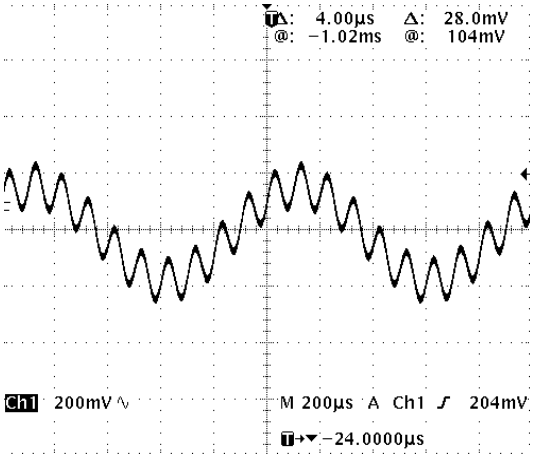

Figure 4: Ideal signal

Generation 1:

Fitness = 11028

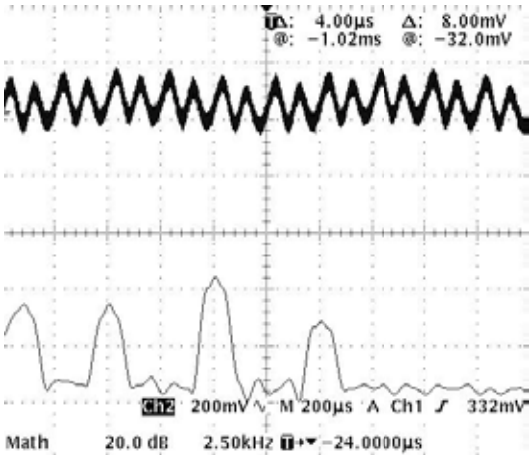

Figure 5: Output signal and its frequency spectrum (gen. 1)

Generation 6:

Fitness = 8625

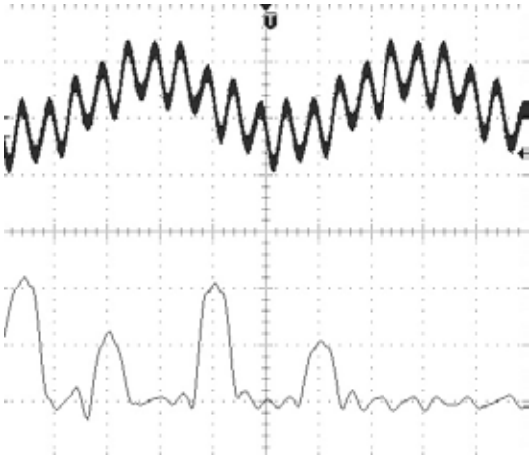

Figure 6: Output signal and its frequency spectrum (gen. 6; 200mV/div, 20dB/div)

Generation 1000:

Fitness = 1005

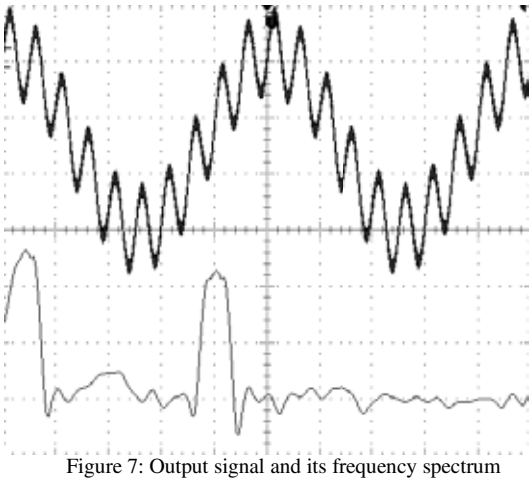

Figure 7: Output signal and its frequency spectrum (gen. 1000; 100mV/div, 20dB/div)

VIII. CONCLUSION

The evolutionary FIR filter was successfully implemented by the FPGA Cyclone II device, and its function is surprisingly good. The system contains two FIR filters. The first one is used for the fitness function, the second one for the processing of the input signal. The evolutionary FIR filter is able to change its impulse response and the type of filter. Performance of the implementation of the standard genetic algorithm is around 8 300 generations per second. The filter uses the fitness function based on the shape of the signal in the time domain. Tests show that the one-point crossover is not suitable for the evolutionary filter. It is evident that another type of the crossover should be found.

REFERENCES

[1] Lažanský, Mařík, Štěpánková: *Umělá inteligence 3.* Academie, 2003. ISBN 80-200-472-6.

[2] Alan V. Oppenheim, Ronald W. Schafer, John R. Buck: Discrete-time Signal Processing. Prentice Hall, 1999. ISBN 0137549202, 9780137549207

[3] Sekanina L.: Evolvable components. Springer, Natural Computing series, 2004. ISBN 3540403779.

[4] Martin, P.: A Hardware Implementation of a Genetic Programming System Using FPGAs and Handel-C, In: Genetic Programming and Evolvable Machines, 2008, p. 317 – 343, ISSN 1389-2576.