# Evolving an Adaptive Digital Filter

Gunnar Tufte and Pauline C. Haddow
The Norwegian University of Science and Technology
Department of Computer and Information Science
O. S. Bragstadsplass 2E, 7491 Trondheim, Norway
gunnart@idi.ntnu.no, pauline@idi.ntnu.no
(47)73594452, (47)73594458

## Abstract

*One important feature of signal processing is coping with noise. In a non-adaptive filter, characteristics of the filter may be refined to remove noise. One method of achieving this is to use evolution to decide the filter characteristics. However, if the noise level is sufficient or the input signal is not of the required type for the output signal required, then a satisfactory output signal may not be achievable. To be able to achieve the required output signal for a wide range of input signals and noise, it is desirable to be able to adjust both the characteristics and the type of the filter. In this way the resulting filter may be said to be an adaptive filter. In this paper we propose an on-chip solution for an adaptive digital filter using an on-chip evolvable hardware method. We highlight a challenge within evolvable hardware for adaptive designs and that is to find efficient ways in which sufficient genetic material will be available to the evolution process. This problem appears when the evolution process is automatically restarted so as to adapt to a change in the environment.*

## 1. Introduction

One of the aims of evolutionary hardware is to be able to offer adaptive hardware. In this work we focus on the design through evolution of an adaptive digital filter. The adaptive filter provides a good case study for adaptive design due to it's requirement for real time fitness evaluation. In addition, we consider an on-chip solution where the genetic algorithm and the evolving filter design itself are implemented on the same chip.

Historically, filtering of audio signals is an analogue problem requiring an analogue solution. Today, especially in applications requiring very fast solutions, analogue solutions are still the answer [8]. However, in many other areas, digital signal processing is the solution used. The reason for the growth in digital solutions is the ease and reduced cost of digital design in comparison to analogue design.

A non-adaptive filter has a predefined function i.e. it may be a low pass, high pass, band pass or band stop filter. However, features such as noise in the environment may effect the filtered result. Adaptive filters, on the other hand, may be designed to adjust to their environment such that the filter may adapt to noise so as to produce the desired result. The desired result itself, is given in the filter's specification. This tells us the type of filter and the type of input signals that the filter can handle. However, if we wish a more flexible filter design then the filter type itself should be adjustable. This may be adjusted to cope with different input signals so as to produce the desired output signal or the specification of the filter itself may be changed to change the desired output signal. In the results presented herein we focus on adjusting the filter coefficients representing the filter characteristics and the filter type to produce the filter response required for the specification given. In addition, we test out the effect of noise on the input signal and the filter's ability to filter out the noise and still retain the required response.

Digital solutions often involve a Digital Signal Processor (DSP). However, in this paper we propose using a field programmable gate array (FPGA) where both a genetic algorithm (GA) [3] and the filter design itself are implemented on the same FPGA chip. The GA is used to generate the filter coefficients thus providing for an adaptive digital filter design.

## 2. Background and Motivation

A digital filter consists of an interconnection of filter taps — the topology. This topology determines the phase and magnitude of the output signal. Each tap holds a filter coefficient. The number of taps decides the accuracy of the filter and the coefficients describe the response required from the input signal. The main operations of a filter are multiplication at each filter tap and accumulation of their results.

As stated, digital signal processing today usually involves a DSP — a processor with an architecture and instruction set specialised for signal processing tasks. One key feature of a DSP is the ability to perform multiply operations at high speed. In the case of a filter, where a constant stream of data is sampled, the time used to process each sample most be less than the sample cycle. Since a DSP processes instructions sequentially, processing of the filter taps is sequential.

An FPGA implementation, on the other hand, provides the opportunity to perform the operations required for the filter taps in parallel by implementing multiple filter taps. Other tasks which may be performed on the incoming data stream, such as generating new filter coefficients, may be processed simultaneously. The ability to generate coefficients at run time is a requirement of adaptive filters. Adaptive filters may be used to suppress noise or for active signal enhancement in environments where noise or signal conditions vary.

One approach to filter design automation is to use evolution to aid the design process. Most work in this area has focused on analogue filter designs, particularly non-adaptive filters which may be classified as passive or active. Genetic Programming (GP) [4] has been used to evolve active filter designs through the choice of topologies, component types and their respective values [5]. Evaluation of the evolved circuits was conducted using SPICE simulations. Zebulum et.al. have also evolved filter designs but in their case passive filters were the focus and a genetic algorithm was used instead of a genetic program [13]. Both of these works may be classified under extrinsic evolution [1] where evaluation of the evolved circuit is conducted in software. Miller has also evolved filter designs using extrinsic evolution [7, 6] where the focus is on reducing the hardware resources required — in these cases using a gate array with only few components.

Other approaches have used intrinsic evolution [13] where the evolutionary algorithm is still implemented in software but evaluation is conducted in hardware. One such approach used a field programmable analogue array FPAA from Zetex [2]. In this case an active filter was implemented using traditional design techniques but the characteristics of the filter i.e. amplification and frequency bands were under the control of the evolution process. A similar approach may be seen by Zebulum et. al [14] but a different device was used — Motorola's MPAA020 FPAA. It should be noted, however, that this device is no longer available.

At ETL, evolutionary techniques have been used in the design of an adaptive filter. An LSI analogue EHW chip for active filters uses a GA to tune the filter to compensate for production variations [8]. This means that unlike the other filter solutions presented in this section, there is no single filter design solution. Production variations in the chips themselves are catered for by allowing the genetic algorithm which evolves the filter characteristics to be implemented on a neighbouring micro-controller. Such a solution, allows evolution to tune the analogue filter chip after production so as to meet the required specification.

At Glasgow, evolutionary techniques are used to design adaptive filters based on IIR filters. IIR filters are less reliable than FIR filters and therefore the aim of their work is not only to increase filter performance but to increase reliability [9, 10].

In this work, an adaptive filter solution is proposed where the complete design — the filter and genetic algorithm are implemented on the same chip. In our case an FPGA is the chosen technology resulting in a digital solution to the adaptive filter problem. The evolutionary process evolves coefficients for the filter. These include the filter characteristics and type.

## 3. Digital Filters

Digital filters may be categorised as Infinite (IIR) or Finite (FIR) Impulse Response Filters. FIR filters are chosen for this work as they may be regarded as a more stable design due to the lack of feedback in the design and, as such, provide a more reliable design in which to focus our study on the effects of evolution on adaptivity.

The topology of a FIR Filter consists of a delay line followed by an accumulator, as shown in figure 1. The incoming signal to the filter (So) is

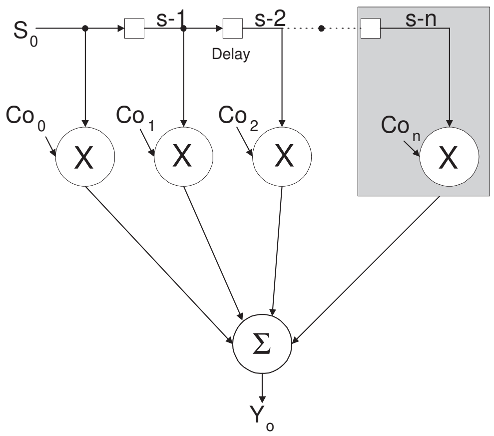


**Figure 1. Architecture of a FIR filter**

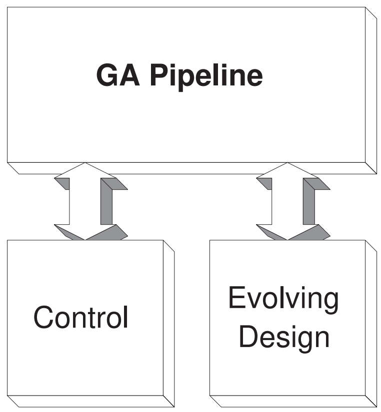


**Figure 2. Data Flow in CHE**

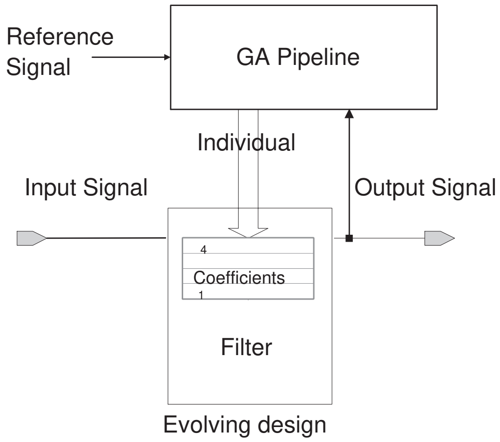


**Figure 3. Adaptive FIR Filter Structure using CHE**

applied to the delay line resulting in samples (s-1) to (s-n). Each tap (illustrated as a shaded box) consists of an incoming sample and a coefficient (represented by $(Co_o)$ to $(Co_n)$. The coefficient is applied to the incoming sample to shape the output sample from the tap. These output samples are accumulated to produce the filter output (Yo).

In non-adaptive filters, coefficients are pre-calculated along with the filter type required and are built into the filter design. In adaptive filters these coefficients need to be calculated at run time.

An adaptive filter consists of a digital filter along with a generator. The generator is an adaptive algorithm that adjusts filter coefficients to meet changing conditions in the input signal — in our case a genetic algorithm is used. The generator uses a reference signal which specifies the desired filter output i.e. the required pass-band and stop-band.

## 4. Implementation

The implementation presented herein may be termed as complete hardware evolution (CHE) [12] which is illustrated in figure 2. The main feature of a CHE implementation is the Genetic Algorithm pipeline (GA-pipeline) which, as the name suggests, is a pipeline of modules representing the various functions or parts of the evolution process required to process a given generation, together with a feedback to handle the creation of further generations. In addition, control hardware allows for communication with the outside world. The third main feature is the evolving design itself — in this case an adaptive FIR filter.

As shown in figure 3, the GA-pipeline and the evolving filter are connected together by a two way communication. Individuals are transferred to the filter and the filter response (current output signal) for the current individual are fed back to the GA-pipeline for fitness evaluation. A given individual provides a set of coefficients for the filter. The reference signal within the generator provides the filter specification i.e. the required pass-band and stop-band, to the GA-pipeline. Control logic is not shown in this figure.

Since the GA-pipeline and the evolving filter are implemented on the same FPGA and never disconnected from each other, a continuous evaluation of filter performance is possible. The monitoring of filter performance is used to restart evolution when the current filter design no longer solves it's

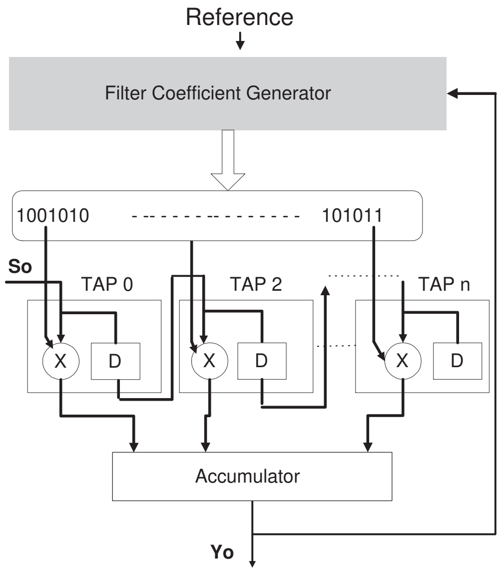


**Figure 4. A Multiply Accumulate FIR Filter Implementation**

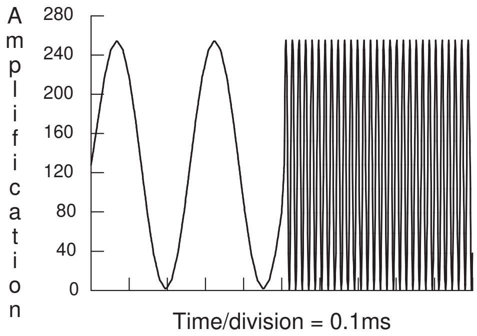


**Figure 5. Input Signal**

task satisfactorily.

The adaptive FIR filter design is based on the Multiply Accumulate structure shown in Figure 4. This may be compared to figure 3 where the GA-pipeline is the generator and the filter of the evolving design represents the delay line and accumulator shown in the figure 4. The coefficients of the evolving design are those coefficients applied to the delay line in figure 4.

## 5 Fitness Evaluation

In the case of a static design, the evolution process drives the evolving design towards an optimal or near optimal solution e.g. a tone discriminator [11] or a multiplexor design [12]. In the case of the adaptive filter there is one single sought solution i.e. the filter most respond to the input signal in accordance with the reference signal. However, the evolution process is not evolving this signal but the coefficients that will enable such a signal to appear at the output. The coefficient solution will vary with varying input signals to the filter and, as such, this type of design may be termed a dynamic design i.e. solutions vary over time.

The fitness function uses the response from the current filter i.e. the output of the design represented by the current individual, to calculate a fitness value (*Fitvalue*) for the proposed solution. This value is calculated by accumulating the difference between the reference signal itself (*refsig*) and the response from the filter (*response*) i.e. the output signal, for each sample — *i to n*. The optimal solution sought is a fitness value of 0 i.e. 100% fitness.

$$Fitvalue = \sum_{i=1}^{n} |refsig - response_i|$$

## 6. Experimentation and Results

In these experiments, the population size is 32 and the genotype length is 32 bits. These 32 bits are split into four eight bit sections representing 8 bit coefficients. The selection mechanism is linear rank selection without elitism. Crossover rate is set to 0.75 and mutation rate is set to 0.27 per bit. These rates are similar to those used at ETL [8]. All experiments were repeated 5 times.

### 6.1. Test 1: Steady Input, Reference Signal A

The purpose of this experiment is to check that the evolution process can find the correct coefficients for a given input signal so as to achieve the desired output value. As such, no type of filter is assumed and a randomly chosen steady input signal, as shown in figure 5, is chosen. The evolution process is thus required not only to find the required characteristics but also the filter type.

Figure 6 illustrates the reference signal i.e. the filter specification, used for this experiment. As shown, the lower frequency component of the input signal ($f_{low}$) lies below the break point in the frequency spectrum and the higher frequency component ($f_{high}$) lies after the break point. In figure 7,

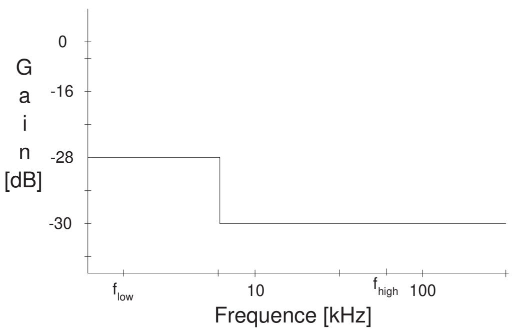


**Figure 6. Reference Signal A**

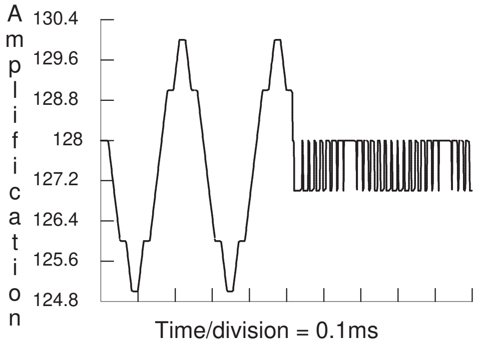


**Figure 7. Test 1: Filter Response**

the response of the filter for the input signal given in figure 5 may be seen. As expected the low frequency element of the input signal is damped but the high frequency element is damped to a much greater extent as required by the reference signal. Figure 8 illustrates fitness of the filter under evolution. As shown, the desired response was achieved after 1171 generations.

## 6.2. Test 2: Steady Input, Reference Signal B

In this experiment, the same input signal is fed to the filter but a new reference signal — reference signal B, shown in figure 9 is used. Again, the lower component of the input signal is below the break point and the higher component is after the break point.

The filter response shown in figure 10 illustrates, as expected from the reference signal, that the lower component of the input signal is damped the most. The design was evolved in 1072 generations, as shown in figure 11.

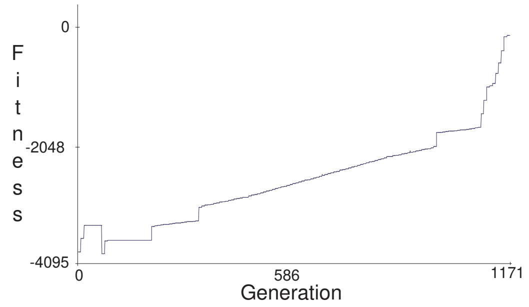


**Figure 8. Test 1: Fitness**

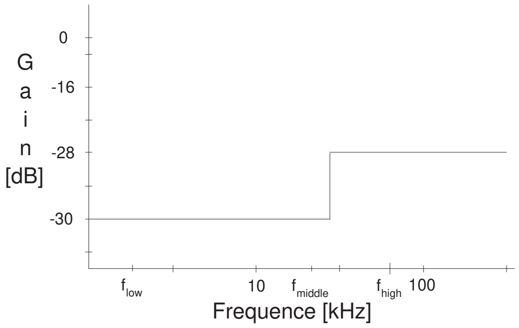


**Figure 9. Reference Signal B**

## 6.3. Test 3: Noisy Input Signal, Reference B

Tests 1 and 2 illustrate different response requirements based on different reference signals. However, the choice of a steady input signal is a simplification of the adaptability available in the adaptive filter. To further test the filter, the filter is tested under environmental variations i.e. noise. A similar test to test 2 was conducted using the same reference signal — see figure 9. To begin with, the same input signal, shown in figure 5 was sent to the filter but after a period noise was introduced. The introduction of noise forces the evolution process to evolve a new set of coefficients to meet the requirements specified in the reference signal. Noise at different frequencies is simulated by altering the frequency components of the input signal. In this case, as shown in figure 12 an extra frequency component between the two existing components is added to the input signal.

The steady input signal was held for a period allowing evolution to reach an optimal solution before noise was introduced. To ensure that the effect of noise on the input signal effected fitness of the best individual, rank selection without elitism was

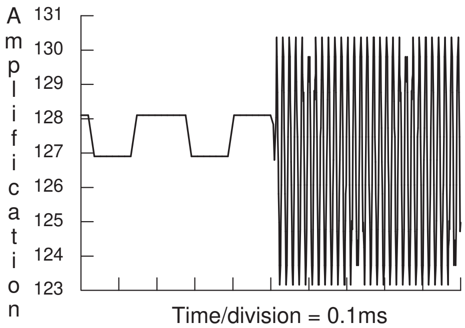


**Figure 10. Test 2: Filter Response**

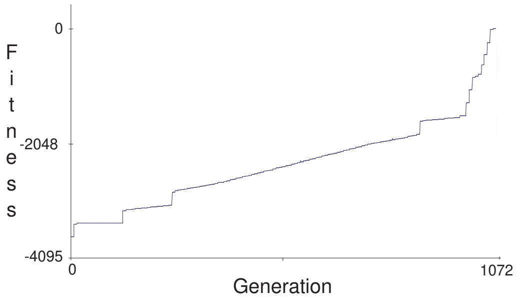


**Figure 11. Test 2: Fitness**

chosen. An obvious fall in fitness is thus expected as soon as noise appears on the input signal.

The reference signal is that used in test 2, however, as shown in figure 12, when noise is introduced to the input signal we have three components instead of two. This new component ($f_{middle}$) lies below the breakpoint, as illustrated in figure 9.

### Test 3a: No Change in the Population

The filter response, before noise is introduced to the input signal, is identical to that illustrated in figure 10. As shown, the filter's response meets the requirements before noise is introduced to the input signal. However, the response to noise on the input signal is quite different as shown in figure 13. At the point where noise is introduced the response suddenly becomes very poor, as expected. However, as illustrated in figure 14, although fitness falls thus triggering the evolution process to evolve a new solution, very little improvement in fitness is seen. Thus there is very little improvement in response after the initial fall.

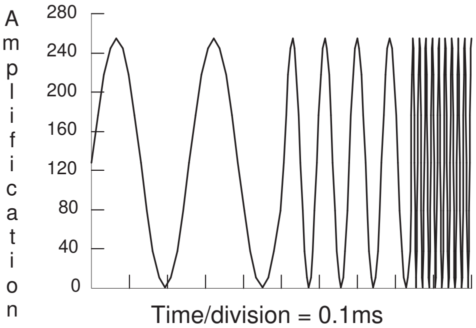


**Figure 12. Noisy Input Signal**

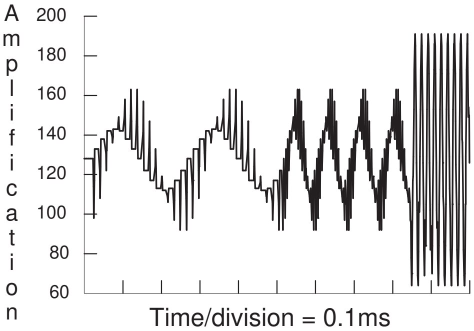


**Figure 13. Test 3a: Filter Response**

Investigation into this problem highlighted no errors in the GA-pipeline or in the filter itself. Thus it was concluded that there was not sufficient variety in the genetic material of the population to allow evolution to evolve to a new solution. One possible solution was to increase mutation to such an extent so as to force a change in the population. However, this method was deemed to be a slow solution and therefore it was decided to introduce new random individuals to the population which would replace some of the existing individuals.

### Test 3b: Introducing Genetic Material to the Population

Test 3 was rerun but this time when fitness and response fell, 50% of the population was swapped out for random individuals. These individuals were obtained by retaining 50% of the original population at generation 0 in memory ready to be swapped in when required.

The response of the input signal before noise is

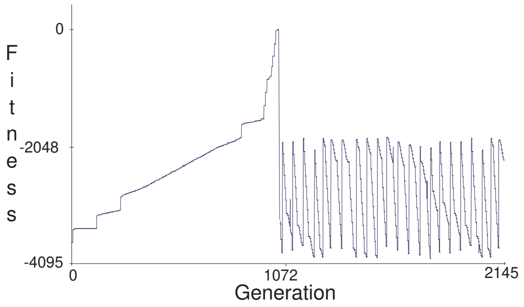


**Figure 14. Test 3a: Fitness**

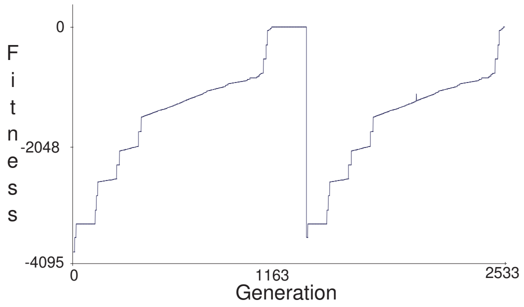


**Figure 16. Test 3b: Fitness**

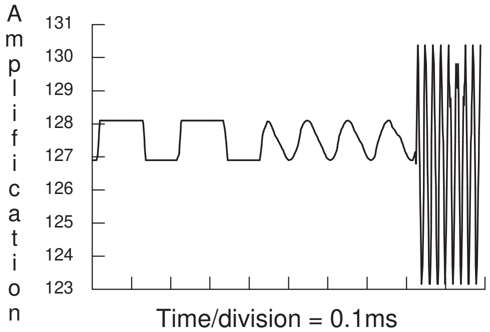


**Figure 15. Test 3b: Filter Response for a Noisy Input Signal**

again identical to that shown in figure 10. However, unlike in test 3a, the required response is achieved after noise is introduced — see figure 15. Figure 16 illustrates fitness throughout the evolution process. The expected fall in fitness is shown as well as a steady rise to optimal fitness as the design evolves towards the optimal solution. Now that sufficient genetic material is available to the evolution process, a new set of coefficients that match the input signal and the reference signal are found and a good response is achieved, as shown.

## 7. Ongoing work

The cost of the solution in test 3b is a little extra glue logic as well as extra memory to store 50% of the population. Off course for designs with large individuals and/or large populations, this solution may involve a large amount of storage. Further investigation is thus required to identify the amount of the population that needs to be introduced to ensure evolution to an optimal solution. This value is off course expected to vary with different applications and, as such investigation into different adaptive solutions is required. In addition, other solutions such as changes to the mutation rate which perhaps together with insertion of a smaller population may achieve similar results. As such, these solutions are worth investigation .

Another feature of this work which promotes further investigation is the choice between a solution similar to that at ETL where a microprocessor was used to run the genetic algorithm and this solution where the genetic algorithm was implemented on-chip. Performance and cost evaluation of these two methods for adaptive hardware with respect to different applications needs further investigation.

## 8. Evaluation & Conclusion

This work provides a case study of an adaptive digital filter design, to study evolution of adaptive designs through the use of CHE. Since the evaluation of how well the proposed design solves the specified task is conducted by the fitness evaluation in the GA-pipeline, exploration of a new design solution is started up when the current design no longer solves it's task satisfactorily. Using CHE, where the evolution process is on the same chip, adaption does not need to be decided at design time but the required change in the circuit is free to evolve dynamically with changing external forces. As such CHE provides a method for adaptive hardware.

We have presented a design approach and implementation of an adaptive digital FIR filter. Under testing we have shown how adaptivity to noise may be achieved. When noise appears at generation X, the current solution no longer solves the problem i.e. fitness falls. The evolution process automatically restarts to create new generations which adapt to the change in the environment — in this case noise. To ensure enough variation in the genetic material at generation X, 50% of the population was replaced with random individuals.

## References


[1] H. de Garis. Artifical life: Growing an artificial brain with a million neural net modules inside a trillion cell cellular automata machine. In *4th International Symposium on Micro Machine & Human Science*, 1993.

[2] S. J. Flockton and K. Sheehan. Intrinsic circuit evolution using programmable analogue arrays. In *2nd International Conference on Evolvable Systems (ICES98)*, Lecture Notes in Computer Science, pages 144–153. Springer, 1998.

[3] D. Goldberg. *GENETIC ALGORITHMS in search optimization & machine learning*. Addison Wesley, 1989.

[4] J. Koza. Genetically breeding populations of computer programs to solve problems in artificial intelligence. In *Proceedings of the Second Int. Conf.on Tools for AI*, pages 819–827. IEEE Computer Society Press, 1990.

[5] J. Koza et al. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation*, 1(2):109–128, July 1997.

[6] J. Miller. Evolution of digital filters using a gate array model. In *Proceedings of the First EvolASP'99 Workshop on Image Analysis and Signal Processing*, pages 17–30, 1999.

[7] J. Miller. On the filtering properties of evolved gate arrays. In *The first NASA/DoD Workshop on Evolvable Hardware*, pages 2–11, 1999.

[8] M. Murakawa, S. Yoshizawa, T. Adachi, K. Suzuki, K. Takasuka, M. Iwata, and T. Higuchi. Analogue EHW chip for intermediate frequency filters. In *2nd International Conference on Evolvable Systems (ICES98)*, Lecture Notes in Computer Science, pages 134–143. Springer, 1998.

[9] S. Sundaralingam and K. Sharman. Genetic evolution of adaptive filters. In *Proceedings of DSP UK*, pages 47–53, 1997.

[10] S. Sundaralingam and K. Sharman. Evolving complex adaptive IIR structures. In *Proceedings of EUSIPCO 98*, 1998.

[11] A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In *1st International Conference on Evolvable Systems (ICES96)*, Lecture Notes in Computer Science, pages 390–405. Springer, 1996.

[12] G. Tufte and P. C. Haddow. Prototyping a GA pipeline for complete hardware evolution. In *The 1st NASA/DoD Workshop on Evolvable Hardware*, pages 18–25, 1999.

[13] R. Zebulum, M. Pacheco, and M. Vellasco. Evolvable system in hardware design: Taxonomy, survey and applications. In *1st International Conference on Evolvable Systems (ICES96)*, Lecture Notes in Computer Science, pages 344–358. Springer, 1996.

[14] R. Zebulum et al. Analog circuits evolution in extrinsic and intrinsic modes. In *2nd International Conference on Evolvable Systems (ICES98)*, Lecture Notes in Computer Science, pages 154–165. Springer, 1998.