EHW Architecture for Design of FIR Filters for Adaptive Noise Cancellation

Uma Rajaram†, Raja Paul Perinbam†, Bharghava††

†Anna University, Chennai, India
††CVEST, International Institute of Information Technology, Hyderabad, India

Summary
This paper describes a new technique for the design of Finite Impulse Response (FIR) Filter within an Evolvable hardware framework, using genetic algorithm (GA), aimed at noise cancellation. This implementation aims at reducing the number of generations required to provide time bound optimal filter configuration and to improve the quality of the filter designed. The filter is designed to reconfigure itself and provide real-time noise cancellation. The filter logic is implemented on a novel reconfigurable fabric. The GA processing and the reconfigurable framework is synthesized on Xilinx XCV1000 hardware chip. The results obtained show the validity of the approach to adaptive noise cancellation using Evolvable Digital Filters.

Key words:
Evolvable Hardware, Noise Cancellation, Adaptive Filter, Reconfigurable Hardware

1. Introduction

Reconfigurable hardware devices make it possible to change the topology of electronic circuits at runtime. Using reconfigurable devices as a platform for Evolvable hardware (EHW) is well suited for real-time adaptive systems. EHW is a scheme inspired by natural evolution, for automatic design of hardware systems. It refers to hardware that can change its architecture and behavior dynamically and autonomously by interacting with its environment. It is built on top of reconfigurable logic devices, whose architecture can be reconfigured by using Evolutionary Algorithms (EA). The reconfigurable device acts as the design space for the EA, which then determines the optimum hardware configuration required for a particular design specification. EHW is best suited for cases where the design specification doesn't provide sufficient information to permit using conventional design methods [2]. For example, the specification may only state desired behavior of the target hardware. In other cases an existing circuit must adapt, i.e. modify its configuration, to compensate for faults or perhaps a changing operational environment. For instance, deep-space probes may encounter sudden high radiation environments and alter a circuit's performance; the circuit must self-adapt to restore as much of the original behavior as possible and quickly. Another example is in the abstraction and processing of the signal of fetus's rhythm of the heart close to the parent's during labor. This involves Adaptive Noise Cancellation (ANC) [6,7].

In this work, a reconfigurable Finite Impulse Response (FIR) filter constitutes the backbone of the Adaptive Noise Cancellation. A genetic algorithm based implementation of FIR filter to cancel out the interference from the varied noise sources and abstract the original signal is presented in this work. Both the filter as well as the hardware required for evolution is implemented in a single Field programmable gate array (FPGA). The circuit is based on context-switching in FPGA-devices and preliminary results indicate the use of a compact hardware as well as fast adaptation. The design is characterized by a multiplier-less architecture that employs Primitive Operator Filter (POF) technique [5] through which digital filters are realized using signal flow graphs comprising low complexity operations. POF is particularly advantageous for autonomous filter design using EHW, as it does not require any initial encoding scheme, such as canonic signed digit (CSD). Furthermore, [4] shows that FIR filters designed using POF are smaller in area than those designed using the CSD approach. The proposed evolvable architecture for ANC is very effective, resulting in significant improvement in terms of reproduced signal quality. The entire system is synthesized on a Xilinx Virtex XCV1000 FPGA. As the entire filter, including the MAC, and the delay elements, are realized on the reconfigurable fabric, a more optimum filter is realized (using fewer resources) for a given frequency response.

Manuscript received January 5, 2009
Manuscript revised January 20, 2009

2. Digital Filter

2.1 Basic Digital Filter

A digital filter consists of an interconnection of filter taps connected in a certain topology. Each tap holds a filter co-efficient. The major operations of a filter are multiplication at each filter tap and accumulation of their results. The number of taps decides the accuracy of the filter and the co-efficient describes the response required from the input signal. The interconnect topology of the taps determines the phase and magnitude of the output signal. FIR filter is chosen in this work as they are regarded as more stable and reliable and as such they can be used to study the effects of evolution on adaptability. A general FIR filter is described by equation (1)

$$Y(n)=k_1x(n) + k_2x(n-1) + k_3x(n-2) \ldots k_mx(n-m) \quad (1)$$

Where k_i is the i^{th} coefficient, *x* is the input signal, *y* is the output signal, 'm' is the number of filter coefficients (taps) and n is the input sample number. The topology of the FIR filter corresponding to equation (1) is shown in Figure 1.

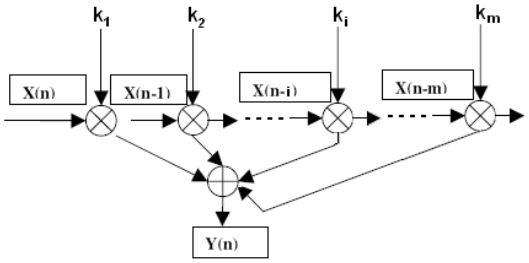

Figure 1 Generic FIR Filter Topology

Implementation of FIR-filters can be undertaken in either hardware or software. A software implementation will require sequential execution of the filter-functions. Hardware implementation allows the filter functions to be executed using parallel functional units and makes improved filter processing speed possible. It is possible to realize an FIR filter using primitive operations [5] such as addition, subtraction, and shifting, thereby reducing circuit complexity. This is of interest in this paper, as it provides a logical framework for the design of a reconfigurable fabric to implement the filter.

2.2 Adaptive Filter

Adaptive filters are self-designing using a recursive algorithm and are useful if complete priori knowledge of environment is not available. Adaptive filters are utilized in a variety of applications, both on stored data and on real time processing tasks. Areas for real time adaptive filter utilization includes room acoustic identification, echo cancellation, Adaptive Noise Cancellation (ANC), CDMA interference suppression etc. The Basic architecture of an adaptive filter is shown in Figure 2.

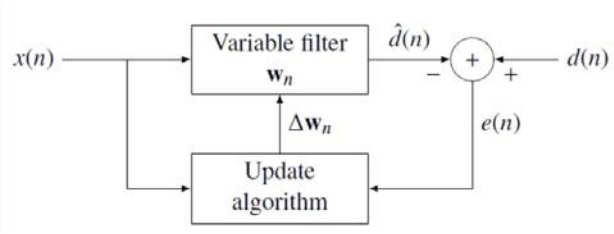

Fig. 2: Basic architecture of adaptive filter

2.3 Adaptive Noise Cancellation

ANC was first proposed by Widrow and Glow in 1975 [6], the objective of which is to filter out an interference component by identifying a linear model between measurable noise source and the corresponding immeasurable interference. Fig. 3 shows the schematic diagram of an ideal situation to which adaptive noise cancellation can be applied.

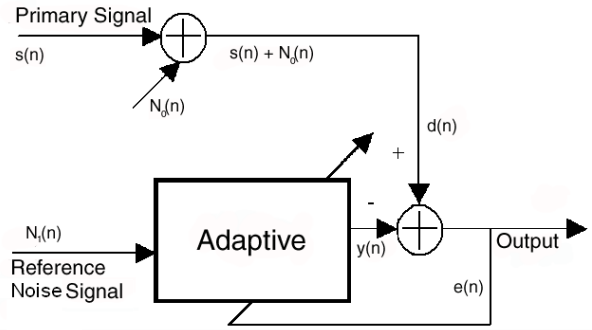

Figure 3 Adaptive Noise Cancellation Scheme

3. Evolvable Hardware

3.1 Evolvable Hardware Concepts

Evolvable hardware is based on the idea of combining reconfigurable devices with evolutionary algorithms such as Genetic Algorithms [3,4]. The basic concept in EHW is to regard the configuration bits for reconfigurable hardware devices as chromosomes for GA. By choosing an appropriate fitness function for the given task, GA can autonomously find the best hardware configuration in terms of chromosomes i.e. configuration

bits. The algorithm for evolving circuits on a reconfigurable fabric is shown in Fig. 4

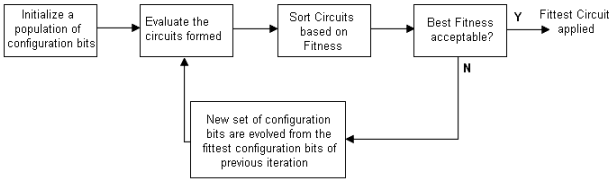

Figure 4 Block Diagram of EHW

Evolvable hardware problems fall into two categories: original design and adaptive systems. Original design uses evolutionary algorithms to design a system that meets a predefined specification. Adaptive systems reconfigure an existing design to counteract faults or changed operational environment. Original design of digital systems is not of much interest because industry already can synthesize enormously complex circuitry. For example, one can buy IP to synthesize USB port circuitry, Ethernet microcontrollers and even entire RISC processors. Some research into original design still yields useful results, for example genetic algorithms have been used to design logic systems with integrated fault detection that out perform hand designed equivalents. Original design of analog circuitry is still a wide-open research area. Indeed, the analog design industry is nowhere near as mature as is the digital design industry. Adaptive systems have been an area of intense interest in the recent past.

The fitness of an evolved circuit is a measure of how well the circuit matches the design specification. Fitness in evolvable hardware problems is determined via two methods:-

1) Extrinsic evolution: All circuits are simulated to see how they perform
2) Intrinsic evolution: Physical tests are run on actual hardware.

In off-line fitness computation (OFL) or Extrinsic evolution, the evolution is simulated in software, and only the elite chromosome is written to the hardware device. In online Fitness Computation (ONL), the hardware device gets configured for each chromosome for each generation (sometimes named intrinsic evolution) [1]. GA is the most commonly used evolutionary algorithm and uses biological operators like crossover and mutation as shown in figure 5.

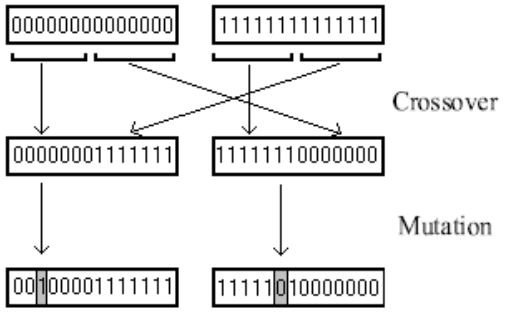

Figure 5 Genetic Algorithm operators

3.2 Related Work

The evolution of an 8-tap filter using a Xilinx Virtex XCV1000 FPGA has been demonstrated in [7], while [8] presents the complete hardware evolution of an adaptive filter (wherein the filter coefficients are evolved). In comparison with these research investigations the proposed design presents a fine-grain approach that offers better results in terms of power over general-purpose FPGAs or multiplier-based solutions. In [12], a fine-grained approach has been reported, which is used to implement image filters. Similarly, [11] proposes an adaptive median filter for image processing. But in these cases the input data is static. A similar design for FIR filters is presented in [9]. But the fitness function considered and the overall system architecture does not suit noise cancellation applications. The present approach differs from the above designs, in that, it tracks the noise accompanying the signal, and constantly evolves to give a better Signal-to-Noise ratio. Also, the implementation of a configuration cache speeds up future static evolutions.

4. System Architecture

4.1 Overall System Architecture

The approach taken in this paper deviates from generic EHW architecture in that it contains two working solutions at any given instant in time, and the more optimized of the two drives the output. The signal flow in the proposed architecture is shown in Fig. 6. This is done in order to dynamically track interference due to noise. Mutation is the more dominant genetic operator than Crossover, so as to provide subtle variations in the filter. Recent solutions of high fitness are stored in a configuration cache in order to reduce the number of generations required.

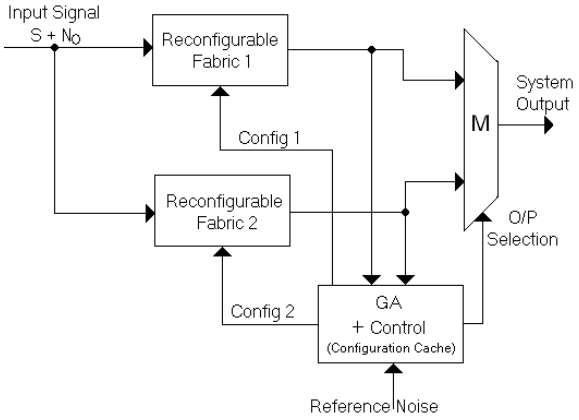

Figure 6 Proposed Overall System Architecture

Initially, both the Reconfigurable Fabrics (RF) are statically evolved to implement a filter of required specification. The respective configurations are then stored in the configuration cache. RF 1 is chosen to drive the output on reset. When RF 1 drives the output, RF 2 is constantly evolved predominantly by mutation, operated cyclically on the contents of the configuration cache. The fitness of RF 1 and RF 2 are compared in each generation and if RF2 is found fitter, then, the GA block selects the output of RF 2 as the system output and RF 1 and RF 2 exchange their previous roles. For the case of extrinsic evolution, the contents of the configuration cache can be used as the initial population. This can lead to reduced number of generations in producing an optimal result.

4.2 Reconfigurable Fabric

As digital filters can be implemented using repeated primitive operations such as addition and shifting as mentioned before, the reconfigurable fabric designed for this purpose consists of Programmable Processing Elements (PPE) with separate configurable shifting operators, and summing operators. Each line in the Reconfigurable fabric architecture is of 'n' bits. The Extended Cartesian Genetic Programming Reconfigurable Architecture [6,10] has been adopted adaptively, as shown in Fig. 7. The PPEs are arranged as a 4x6 matrix, with an extra PPE for the output. The input to this 4x6 matrix is the original input signal and its three delayed versions. In this implementation, 8-bit data is considered. The inputs to each PPE are connected to the outputs of the previous *l* columns, where *l* is the level-back parameter. The PPE operates on these inputs according to the configuration bits provided to it by the GA block. The Buffer Units (BU) are inserted between adjacent rows, to enable future pipelining and achieve higher speeds of operation.

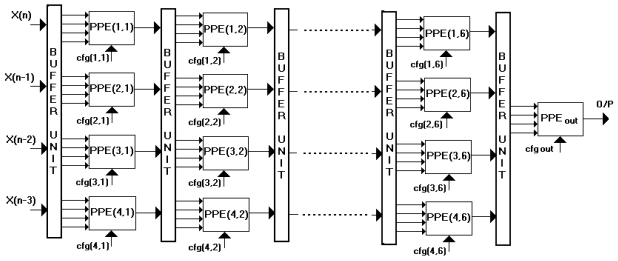

Figure 7 Extended Cartesian Genetic Programming Reconfigurable Architecture

For the current implementation on a Xilinx XCV1000 FPGA chip, the level-back has been chosen as 1 in order to realize the multiplexer blocks efficiently. Also, as the input to the 1st column consists of delayed versions of the input, a level-back value of 1 is justified. The chromosome for the reconfigurable fabric design is the set of all configuration bits representing 25 sets of 5 integers each.

4.3 Programmable Processing Element

The PPE used in the reconfigurable fabric has been designed to perform primitive operations in implementing an FIR filter. The architecture of the PPE is shown in Fig. 8.

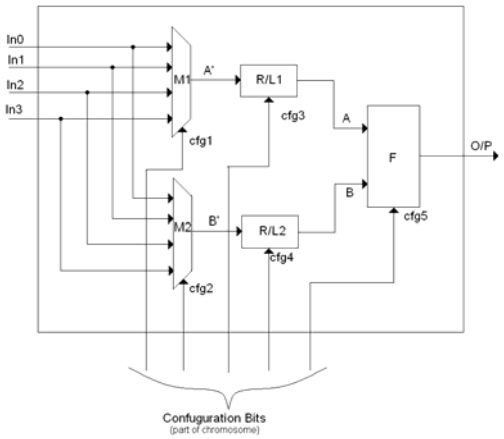

Figure 8 Architecture of the Programmable Processing element

The configuration of the PPE is determined by 5 sets of bits: *cfg1, cfg2, cfg3, cfg4 and cfg5*. *cfg1* and *cfg2* route the input signals to modules A' and *B'* and each measure 2 bits. The R/L blocks are shifter units. The configuration blocks c*fg3* and *cfg4* operate according to Table 1 and each are of size 2 bits.

Table 1: Configuration cfg3 and cfg4 of R/L blocks

Configuration Bits	Function
00	No shift
01	1 Left Shift
10	1 Right Shift
11	2 Right Shift

Table 2 gives the details of c*fg5* and is of 3 bits.

Table 2: Function code for *cfg5*

Configuration Bits	Function/Output
000	**A**
001	**B**
010	**A** XOR **B**
011	**A** XOR Inv(**B)**
100`	**A** OR **B**
101	**A** AND **B**
110	Inv(**B**)
111	Inv(**A**)

The total number of functions that can be performed by the designed PPE is decided by 7 bits, which implies that a total of 128 different functions can be performed by each of the PPE. The number of bits required to configure a single PPE is 11 and hence, 275 bits are required to configure the entire reconfigurable fabric (25 PPEs).

4.4 Implementation

The EHW CGP structure is implemented in a Xilinx Virtex XCV1000 FPGA. As shown in Figure 7, the phenotype layer contains the structural information for evolution and consists essentially of PPEs. Every PPE (which is the genotype) cell is made up of two input multiplexers, one Functional Block (FB), 2 shifter blocks and necessary interconnections as shown in Fig. 8. The FB contains a compact and possibly redundant representation of the functions, one of which is to be chosen as the active function for this PPE cell. The PPEs are initialized with the same functionality before they are reconfigured in the course of the evolution.

The evolutionary circuit takes in the input signal and reference noise signal as the input and produces a single output. For each PPE, the, multiplexer inputs will be chosen from the outputs of the previous l (here l=1) columns. The output of the FB is connected to the output of the PPE and is given by O as described by equation 2.

O={R/L[MUX(**cfg1**),**cfg3**],R/L[MUX(**cfg2**),**cfg4**],**cfg5**}

(2)

The fewer the functions, the faster is the evolution. There exists a trade-off between the functionality and the complexity of the hardware structure. In the current implementation the functionality is compromised in order to achieve lesser circuit complexity.

Each PPE array is provided with 4 delayed versions of the input signal every clock cycle. In each subsequent cycle, 3 inputs remain the same (but shifted), and the next signal value is loaded onto one of the inputs. Each PPE is directly supplied with data individually. The total delay exhibited by the EHW fabric equals the sum of the individual delays of the BU, the PPE, multiplied by (Number of Rows + 1), and also the initial delay of 3 units due to input format. Insertion of Buffer stages provides efficient filter tap control. The enable and clock signals provided to each PPE can be programmed to provide multiple delays. The PPE architecture requires 22 slices of a Xilinx Virtex xcv1000FPGA.

5. Genetic Algorithm Specification

As shown in section 4.3, the length of the chromosome for each RF is 275 bits. A GA with elitism and of a fixed population is used with selection, mutation and low probability crossover operators. An initial population of 24 individuals is generated constrained by the chromosome schemas. These individuals are tested for fitness and are evolved till terminal conditions are met. The first 2 individuals which satisfy terminal conditions are used to configure RF 1 and RF 2. The rate of mutation is taken at 8 % per gene and the crossover rate is fixed at 10%. The level of mutation is very high when compared to a generic EHW. An example chromosome for the designed RF is 12013 33121 12312 23212 11212 22312 22123 22121 …. Each of the single digit integers is represented by its 2-bit binary equivalent.

In the case of a static design, the evolution process drives the evolving design towards an optimal or near optimal solution. In the case of the adaptive filter the solution required varies with the input signal relative to the reference signal. The evolution involves the configuration

of the RF which realizes the filter. The 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.

$$Fitvalue = \sum_{i=1}^{n} \left| (Nr + Ri) - (Xi) \right| \qquad (3)$$

where,

Nr - Reference Noise signal
Ri - Response of the Evolved Filter
Xi - Input Signal (+ Noise)

This value is calculated by accumulating the difference between the sum of the reference noise signal and the filter output, and the noisy input signal, for each sample — 1 to n. The optimal solution sought is a fitness value of 0 i.e. 100% fitness. In the proposed architecture, after static evolution of a filter, the filters represented by RF 1 and RF 2 are put through the process of evolution alternately and the system output is selected depending on the fitness values of the 2 filters. The filter not being evolved can also be subjected to dynamic evolution as in [8] and could provide better results.

6. Simulation Results

In this section, varying noise conditions are considered and the evolved RF architecture is presented. The mean absolute error i.e. the mean absolute deviation between the true variations and the evolved RF output is computed and used as a measure to demonstrate the tracking ability of the evolved architecture.

Two signals were considered for the test. These were subjected to white noise of unit amplitude. Fig. 9 shows one form of the evolved architecture of the reconfigurable fabric during operation. The blank PPEs were unused for the particular filter configuration. Fig. 10 and Fig. 12 give the deviations of the output with respect to the original value of the transmitted signals. Fig. 11 and Fig.1 3 show the absolute error values of the filter for the 2 signals.

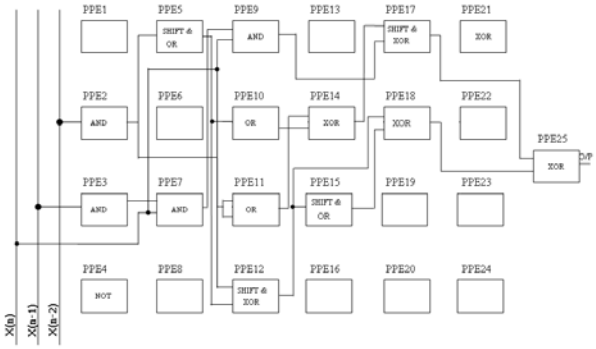

Figure 9 Evolved Architecture of the RF

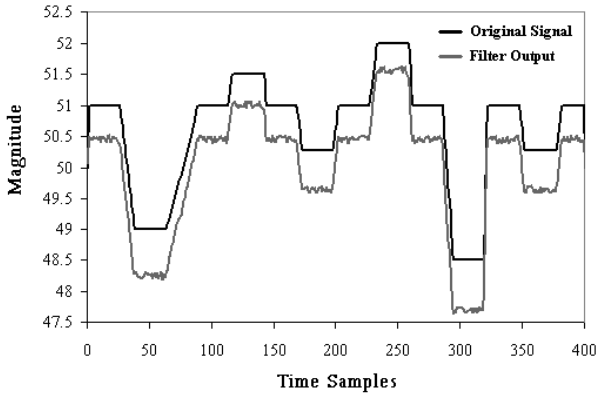

Figure 10 Variation of Filter Output for Signal 1

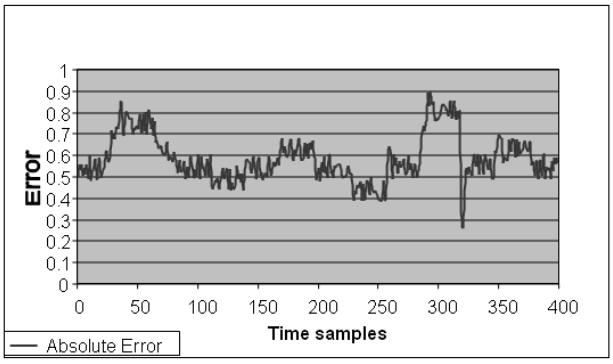

Figure 11 Error Curve of Filter Output for Signal 1

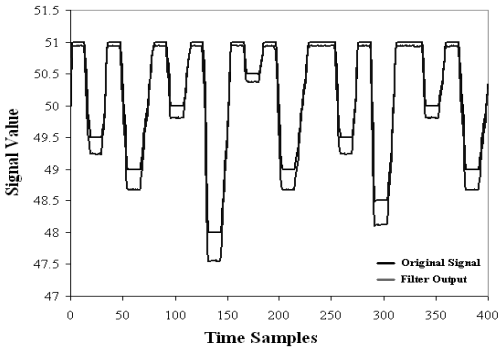

Figure 12 Variation of Filter Output with Signal 2

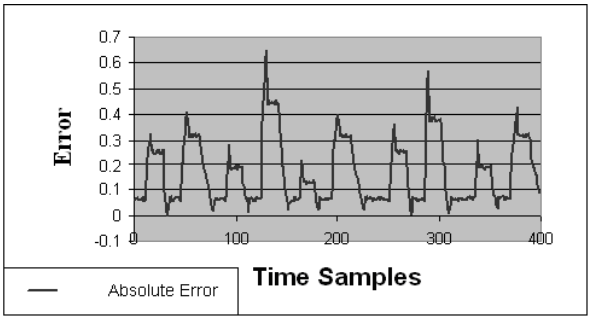

Figure 13 Error Curve of Filter Output for Signal 2

The waveform of the filter output closely conformed to the original signal waveform. The Mean Square Error value for signal 1 was calculated to be .587, and that of signal 2 to be .423. The above results validate the use of an EHW approach for noise cancellation.

8. Conclusion

An EHW based reconfigurable fabric is proposed in order to implement a noise canceling FIR filter. The filter is designed to reconfigure itself and provide real-time noise cancellation. The filter logic is implemented on a novel reconfigurable fabric designed for the specific purpose of implementing an FIR filter using primitive operators, and is synthesized on a Xilinx XCV1000 hardware chip. The results obtained show the validity of the approach to adaptive noise cancellation using Evolvable Digital Filters. The filter was able to track signal variations, and retrieved the signal data with minimal error. The architecture provides the capability of implementing the reconfigurable fabric in a pipelined fashion, but the current implementation does not make use of this feature. Future work includes the implementation of the pipelined version of the filter for improved speed, and to optimize the programmable processing element at circuit level for efficient ASIC implementation of the reconfigurable fabric.

References

[1]. Jim Torresen, “An Evolvable Hardware Tutorial”, FPL 2004, 821-830

[2]. L. Sekanina, “Evolvable Hardware Tutorial”, in GECCO 2007, New York

[3]. Bernard Widrow and Samuel D. Steavns, “Adaptive Signal Processing”, Pearson Edition, 2000.

[4]. Redmill, D. W., Bull, D. R., and Dagless, E., “Genetic synthesis of reduced complexity filters and filter banks using primitive operator directed graphs”. *IEE Proc. Circuits Devices Syst,* vol.147, pp. 303-310, 2000.

[5]. Bull, D. R. and Horrocks, D. H., “Primitive operator digital filters”, IEE Proc. Circuits, Devices and Systems, pp. 401-412, 1991.

[6]. L. Sekanina and P. Mikusek, “Analysis of Reconfigurable Logic Blocks for Evolvable Digital Architectures”, EvoWorkshops 2008, LNCS 4974, pp. 144–153, 2008.

[7]. Vinger, K. A. and Torresen, J., “Implementing evolution of FIR filters efficiently in an FPGA”, Proceedings of NASA/DoD Conference on Evolution Hardware (EH’03), pp. 26-29, 2003.

[8]. Tufte, G. and Haddow, P. C., “Evolving an adaptive digital filter”, Proceedings of the 2nd NASA/DoD Workshop on EH, 2000, pp. 143-150, 2000.

[9]. Evangelos F. Stefatos et al., “An EHW Architecture for the Design of Unconstrained Low-Power FIR Filters for Sensor Control Using Custom-Reconfigurable Technology”, Proceedings of the 2005 NASA/DoD Conference of Evolution Hardware (EH’05)

[10]. L. Sekanina, “Virtual Reconfigurable Circuits for Real-World Applications of Evolvable Hardware”, Evolvable Systems: From Biology to Hardware. Fifth International Conference, ICES 2003, 186-198

[11]. Vašíček Zdeněk, Sekanina Lukáš, "Novel Hardware Implementation of Adaptive Median Filters", In Proc. of 2008 IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop, Bratislava, SK, IEEE CS, 2008, p. 110-115, ISBN 978-1-4244-2276-0

[12]. Yang Zhang, Stephen L. Smith, and Andy M. Tyrell, “Digital Circuit Design using Intrinsic Evolvable Hardware”, Proceedings of NASA/DoD Conference on Evolution Hardware (EH’04), 2004.

[13]. Negoita Mircea, Sekanina Lukáš, Stoica Adrian, "Adaptive and Evolvable Hardware and Systems: The State of the Art and the Prospectus for Future Development", In Lecture Notes in Computer Science, Vol. 2008, No. 5179, DE, p. 310-318, ISSN 0302-9743

Author Biographies

1) Mrs. Uma Rajaram is a research scholar in the Electronics & Communications Engineering, Anna University, Chennai. Her areas of interest include evolvable hardware, genetic algorithms, adaptive signal processing etc.

2) Dr. Raja Paul Perinbam is a Professor in the faculty of the Electronics & Communications Engineering, Anna University, Chennai. His areas of interest include EHW, embedded systems, low power VLSI etc.

3) Bharghava is a graduate research scholar at the Center for VLSI & Embedded Systems, International Institute of Information Technology. His research interests include Processor Architecture, Multicore Processors, and Reconfigurable Computing.