Hardware Based Genetic Evolution of Self-Adaptive Arbitrary Response FIR Filters

Abstract

This work presents a hardware implementation of an FIR Filter that is self-adaptive; that responds to arbitrary frequency response landscapes; that has built-in coefficient error tolerance capabilities; and that has a minimal adaptation latency. This hardware design is based on a heuristic genetic algorithm. Experimental results show that the proposed design is more efficient than non-evolutionary designs even for arbitrary response filters. As a byproduct, the paper also presents a novel flow for the complete hardware design of what is termed as an Evolutionary System on Chip (ESoC). With the inclusion of an evolutionary process, the ESoC is a new paradigm in modern System-on-Chip (SoC) designs. The ESoC methodology could be a very useful structured FPGA/ASIC implementation alternative in many practical applications of FIR Filters.

Index Terms

Digital Adaptive Filters, Genetic Algorithms, Finite Impulse Response (FIR) Filters, Arbitrary Frequency Response Filters, Intrinsic Evolution, System-on-Chip (SoC)

I. Introduction

Important signal processing functions include removal or enhancement of signal components. Filters are extensively employed for the same. Specifically they are used in many cutting edge electronic applications within which they form critical elements. Filters could be analog or digital and may posses an Infinite Impulse Response (IIR type) or a Finite Impulse Response (FIR type). Finite Impulse Response (FIR) digital filters have many applications in a wide range of Digital Signal Processing (DSP) algorithms. The lack of feedback in the design makes them more reliable and robust than their IIR counterparts. In practice, *Digital Finite Impulse Response (FIR) Filters* implement Eq.(1) where, p is the order of the filter, h_i are the filter coefficients, $x[n]$ and $y[n]$ are the n^{th} input and output signal samples respectively.

$$y[n] = \sum_{i=1}^{p} h_i x[n-i] \tag{1}$$

A. Adaptive Filters

Adaptive filters are those which self-adjust their filter coefficients according to an optimizing algorithm. This could be due to changing requirements on field as well as due to application specific demands. By way of contrast, *non-adaptive filters* have static filter coefficients. Because of the complexity of the optimizing algorithms most adaptive filters are digital. They are routinely used in a wide range of DSP applications. Equalization of data transmission channels in high-speed MODEMS, noise cancellation in speaker phones, interface removal in medical

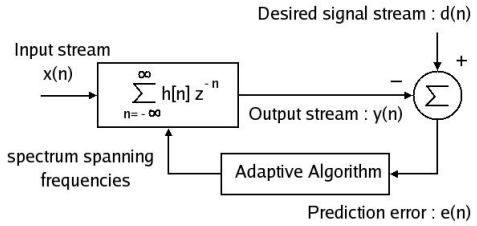

Fig. 1. A general adaptive FIR filter topology using the output error formulation as a feed. The prediction error ($e(n)$) could be different from a simple difference between the output and the desired signal stream.

imaging, canceling narrow-band interference in direct sequence spread spectrum systems and beam reforming in radio astronomy are some practical applications of adaptive systems. Conventional first generation systems typically embody basic *FIR Linear Filter* structures along with simple gradient descent or least squares algorithms for coefficient adaptation. As such, they have certain performance limitations [1]. A more accurate adaptive structure is needed to realize a robust filter that adapts to the changing requirements on filed, quickly and correctly. Fig.1 shows a general *Adaptive Digital FIR Filter* topology. It shows an input sample stream ($x[n]$) filtered into an output stream ($y[n]$) which is compared with the desired sample stream ($d(n)$). The resulting deviation in the output is used as the prediction error ($e(n)$) to guide the adaptive algorithm which reconfigures the filter coefficients ($h[n]$) dynamically.

B. Some Approaches to FIR Filter Design

Signal processing architectures find *adaptive filter* fabrics indispensable and often need them to be capable of responding to various malfunctions caused by endogenous and exogenous factors. Consequently, *error tolerance* in adaptive systems demands a need for resilient designs. We can realize this using hardware fault tolerance which can be achieved by means of redundancy and other supplementary hardware modulation techniques [2]. However the most critical component of the design - the filter coefficient values, need to be corrected in alternate, quicker ways for reduced circuit off-times. Also, for error tolerance to be practical and effective, reduced overall on-chip *adaptation latency* is also essential. In reconfigurable systems, minimization of *adaptive complexity* is another critical objective.

In [3], Adams and Wilson sought *adaptation efficiency* by dividing the filter design into two stages - a computationally efficient pre-filter followed by an amplitude equalizer. For the same objective, Nevo, et.al in [4] described FIR cascaded structures with an interpolated filter; In [5], Wade, Van-Eetvelt and Darwen have described a non-interactive filter design method; And Suckley [6] has furthered on it to propose band-fit filter designs using cascaded filter primitives. However, these *complex filters* exhibit little or *no error resilience* besides having a *limited adaptation range*. Their hardware implementation is plainly impractical. Realistic hardware platforms impose restrictions based on fixed register widths leading to a loss of precision in the storage of filter coefficients. The resulting filters known as Finite Word Length (FWL) filters would however accept this trade-off to benefit from faster computations.

Several *simple configuration* methods have been proposed for effective FWL design of FIR filters or for filters with Canonical Signed Digit (CSD) coefficients [7] [8] [9]. Rounding-off the coefficients to the nearest integers has been another common practice. Coefficients were also sought to be represented as the sum of a few Signed Power-of-Two (SPT) terms. Further improvements were proposed using Primitive Operator Directed Graph (PODG) representations in [10]. However, these methods have issues of *poor response fidelity and high latency*.

Conventional filter designs hence involve multiple, often conflicting design criteria and finding an optimal solution is therefore not a simple task. Analytic or simple iterative methods usually lead to sub-optimal designs. This necessitates optimization based methods for filter design [11] [12] [13]. However, the such methods formulated for digital filters are often complex, highly non-linear and multi-modal in nature. However, in recent years, a variety of optimization algorithms have been proposed based on the mechanics of natural selection and genetics, which have come to be known collectively as genetic algorithms (GAs) [14]. GAs are stochastic search methods that can be used to search for an optimal solution to the evolution function of an optimization problem [15]. Holland proposed GAs in the early seventies [16]. De Jong extended GAs to functional optimization [17] based on the detailed mathematical model for the GA presented by Goldberg in [14]. In the past, GAs have been applied to a number of optimization designs including digital and analog filters [19]. Genetic filter designs have also been of considerable interest in recent years. Genetic evolution (application of several instances of a given GA) of FIR Filters has been of considerable interest in the recent past [1] [6] [18] [20] and [21]. However, the focus in each of these has been discrete and diverse. In [20], Tufte and Haddow demonstrate a model for GA operations on FIR filters. But their focus is not on design practicality (hardware implementation). They consider non-integer (floating point) coefficients which makes their design process all the more complex. In addition, the fitness function of the GA they use does not scan the entire frequency spectrum but considers only specific frequencies for evaluation. This limits the accuracy coefficient estimates. The trade-off not considered here is that between integer coefficients (avoiding floating point operations) and excess spectrum samples (accuracy).

Genetic Algorithms use simple operations over a large number of iterations to optimize system goals making their implementation easy and effective. Errors in filter coefficients lead to an egress in the prediction error which could be used to correct the faulty coefficients on-field. GAs also have the flexibility to determine coefficient values for quite an arbitrary set of frequency response characteristics. Such nasty responses are a common premise of many real life applications. However the major challenge in the genetic design for FIR filters is the adaptation latency which can upshoot quite significantly owing to a poor search methodlogy. A GA involves a large number of heuristics based on the problem dynamics which when exploited appropriately alleviates the latency issue to a significant extent.

C. Genetic Operators - Principles of Natural Selection

Genetic Algorithms (GAs) manipulate a population of individuals in each generation (iteration), where each individual, termed as a chromosome or genotype, represents one candidate solution to the problem. Within the population, individuals with better fitness survive to reproduce. Their genetic material is varied to produce new

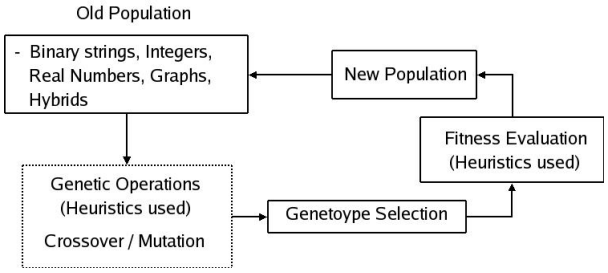

Fig. 2. A Generic GA Cycle - Involves four major steps Initial Population Generation, Variation, Evaluation and Selection

individuals as offsprings which form the seeds for the following generations. The genetic material is modeled using some data structure with finite attributes. As in nature, selection provides the necessary driving mechanism for better solutions to survive. Each solution is associated with a fitness value that re- flects how good it is compared to the other solutions in the population. The variation process comprises of crossover and mutation, which concoct material by partial exchange among genotypes and by random alterations of data strings respectively. The frequency of these operations is controlled by certain pre-set probabilities which require heuristics appropriate for the particular problem at hand. The representation, variation, evaluation and selection operations constitute the basic GA cycle or generation, Fig.2, which is repeated until some pre-determined criteria are satisfied which also require a comprehension of the problem heuristics. With increased computing power and hardware, simulation of evolutionary systems is becoming more and more tractable and GAs are being applied to many real world problems including the design of digital filters. The crucial need in such designs is the use of appropriate heuristics for probabilistic variation and selection. In this paper, we present analytical insights into the selection of the Initial Population (IP) and the Genetic Operators (GO) used by the GA. This exploits the problem heuristics to the maximum and unlike several instances of GA reported in the literature, the IP for the GA proposed in this paper is not derived at random but is derived using a deterministic procedure based on the properties of the impulse response of FIR filters. This makes the entire adaptation process fast and robust. As a byproduct, the paper also proposes a comprehensive methodology for the design of what is termed as the Evolutionary System on Chip (ESoC).

The rest of the paper is organized as follows. In Sec.II, we explore existing filter architectures and their advantages. Specifically, we dwell on the comparison between the spatial design and the frequency domain design of FIR Filters. In Sec. III, we present the functions used by the evolutionary algorithm with analytical justifications for choosing them. In Sec.IV, we propose and describe the complete Evolutionary System-on-Chip (ESoC) hardware design and its parallelism, scalability and Built in Error Tolerance (BiET), . Sec.V provides experimental results for the proposed design and hardware architecture where we present simulation cases for exemplary linear phase FWL test cases. Finally we conclude in sec.VI.

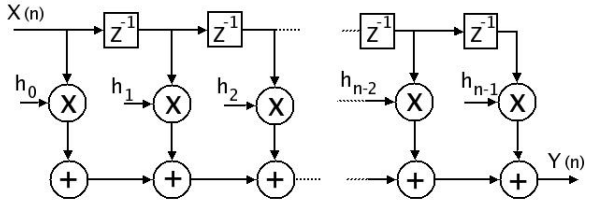

Fig. 3. Direct Form Implementation of Digital FIR Filters

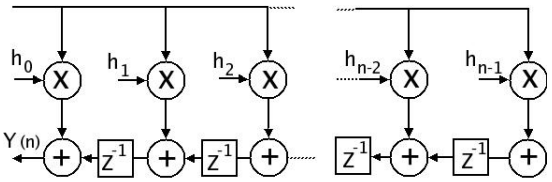

Fig. 4. Transposed Direct Form Implementation

II. FIR Architectures and Design

A. FIR filter architectures

Digital FIR filter outputs are related to their inputs by the following difference equation (Expanded from Eq.(1))

$$y[n] = h_0 x[n] + h_1 x[n-1] + \ldots + h_p x[n-p] \tag{2}$$

where p is the filter order, $x[n]$ is the input signal, $y[n]$ is the output signal and h_i are the filter coefficients. Hardware implementation of the filters in the time domain can have the canonical form translating equation (2) directly or the *broadcast(transposed direct) architecture* obtained using the inversion property of the resulting Signal Flow Graph (SFG) - Fig.3 and Fig.4. We use the broadcast architecture in our hardware design. FWL constraints require truncation of coefficient lengths besides their magnitudes. The minimum length of linear phase low-pass FIR filters to meet the frequency domain specifications is approximately

$$N = \frac{-20log_{10}\sqrt{\delta_p \delta_s} - 13}{14.6\Delta F} + 1 \tag{3}$$

where δ_p and δ_s are the passband and stop band ripples and ΔF is the transition bandwidth [4]. Eq.(3) forms the initial coefficient length constraint on our filter specification which is done offline.

B. Spatial and frequency domain design

The filter coefficients used in the FIR architecture form a $sinc$ function, the Fourier transform of which is a window, Fig. 5. Frequency selective filters could be designed in the *Frequency Domain (FD)* for which a multitude of complex algorithms exist. They focus on the process of cumulatively windowing the uniform frequency span to approximately reach the desired frequency response. This is then inverse transformed to translate to a $sinc$ in the time domain. Some popular FD domain design algorithms are outlined in [22]. *Spatial Design (SD)* is an alternative to the Frequency Domain Design (FDD) of FIR filters. It is advantageous over the FDD as it eliminates the need for the complex operations of transformation and inverse transformation of the response to determine its fitness in an iterative design. This has tremendous implications in hardware implementability simplifying the design flow to basic operations and hence speeds up the compuational steps substantially. However a large number of samples may still be needed for an accurate determination of the fitness function for the GA. This trade-off is a design choice and we stick to simpler operations in the spatial domain for the filter operation. In the next few sections, we

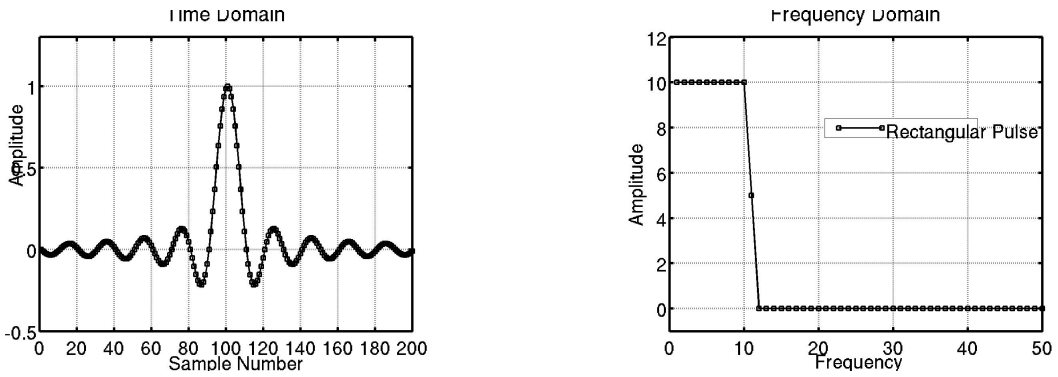

Fig. 5. A Sinc Function expressed as f(x)=sin(x)/x and its transform, a window function. The FIR filter coefficients form a sinc function.

propose the genetic evolution of filter coefficients in the time domain including appropriate heuristics with analytical justifications and describe its complete implementation in hardware.

III. Evolutionary System Design

The objective of spatial design of FIR filters is to accurately estimate the coefficients h_i, $1 \leq i \leq p$, in Eq.(1) at any point of time. This section explains how a GA can be employed for the same. The proposed algorithm has the following steps.

Step 1 *Limiting the search space for the coefficients*

(A) Determination of the largest coefficient h^{max}

Given the order of a filter p, the first step in the FIR design process is to estimate the largest value, h^{max}, that could be assigned to the coefficients h_i, $1 \leq i \leq p$ in Eq.(2). This is done as shown in Eq.(4) [22].

$$h^{max} = \int_0^1 (G_{id}) \mathrm{d}f_n \tag{4}$$

where G_{id} is the expected frequency response normalized w.r.t f_n, $f_n = \frac{f}{f_s}$, f_s is the sampling frequency and f is the range of frequencies considered in the spectrum. The summation shown in Eq.(5) is a trapezoidal approximation to the integral in Eq.(4). This is easily implementable in hardware as an alternative evaluation of h^{max}.

$$h^{max} = \sum_{j=0}^{N_j} \sum_{i=1}^{9} \frac{10^{j+1}}{2f_{Nyq}} \times G_{id}(10j + i) \tag{5}$$

The limits in Eq.(5) for i and j are chosen so as to span the frequency spectrum containing the frequency components of the input signal. f_{Nyq} in the equation is the Nyquist Frequency obtained as an input parameter and N_j is the highest decade order of the expected response. This is a tunable factor which dictates the accuracy of initial evaluation.

(B) Determination of coefficient search range h_i^{max}

The odd number (p) of filter coefficients, h_i in Eq.(1), are symmetric about the mean value h_q, where $q = \frac{(p+1)}{2}$. The FIR filter coefficients lie on a $sinc$ function. In general, a $sinc$ function is given by $sinc(x) = \frac{sin(x)}{x}$ as

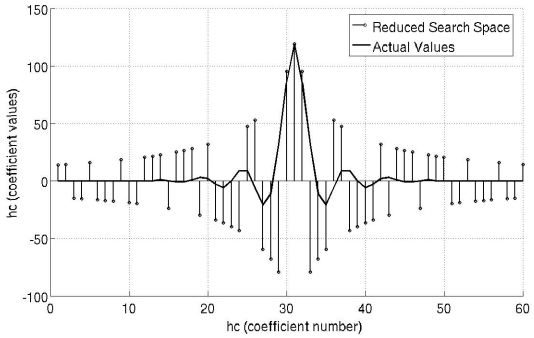

Fig. 6. Reduced GA search space with the Gaussian Switch. f_c=0.25 and p = 60

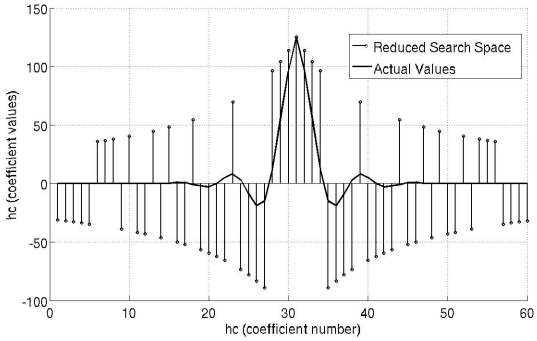

Fig. 7. Sample case with f_c=0.1 and p = 60

explained in Sec.II-B. From Eq.(5) the mean value h_q of the filter coefficients is determined which is the largest value among the coefficient set of cardinality p. To estimate an upper bound (h_i^{lim}) for coefficients other than h_q, we use a $\frac{1}{x}$ functional model. The upper bound is used to limit the search space for the forthcoming genetic algorithm. This model starts with h_q as the highest value of the function with a gradient modulated by the lowest cut-off frequency, f_{cutoff}, of the filter which is obtained as an input parameter. This is shown in the computation of the interpolating step in Eq.(7). The modulation is justified by the fact that the gradient fall for the $\frac{1}{x}$ model close to the mean coefficient for steeper cut-off filters is lesser than that for filters with wider transition bands. Fig.6 and Fig.7 show comparative effects of incorporating f_{cutoff} in the range estimation. In Fig.7, the normalized cut-off frequency is lower and hence the gradient fall from the mean value is slower compared to that in Fig.6. This model captures the fact that a large number of coefficients are close to the mean value for filters with lower cut-offs. Hence, based on the maximum coefficient value, h^{max}, determined in Step 1(A) we can compute the maximum range for each of h_i in Eq.(1) by modeling a $\frac{1}{x}$ behaviour. This can be done by linear interpolation in the inverse coefficient space, Eq.(8). As a starting point for this, we compute Δ^{mean} and δ_{step} as shown in Eq.(6) and Eq.(7). h_i^{lim}, the coefficient search range for each of the i^{th} coefficient is hence computed using δ_{step} linear increments from the mean value Δ^{mean}.

$$\Delta^{mean} = \frac{1}{h^{max}} \tag{6}$$

$$\delta_{step} = \frac{f_{cutoff}}{h^{max}} \tag{7}$$

$$\Delta_i^{lim} = \Delta^{mean} + i.\delta_{step} \tag{8}$$

$$h_i^{lim} = \left\lceil \left(\frac{1}{\Delta_{n-i}^{lim}} \right) \right\rceil \tag{9}$$

Based on the description of functional model in this section, we can determine the coefficient limit envelope h_i^{lim}, Eq.(9), which determines the range within which each of the filter coefficients, h_i, $1 \leq h_i \leq q$, in Eq.(1) lie. This methodology helps narrow down the search domain from infinity to h_i^{lim} within which the possible coefficients

may lie. To translate the Δ_i^{lim} values into the coefficient limits, we take upper integer ceils of the reciprocals of Eq.(8) and flip their order to get the coefficient limits.

Step 2 *User Configuration*

The filter hardware is designed to accomodate a maximum of IP_{max} Initial Populations (IP) in the parallel process such that a given population set can hold upto p_{max} coefficients. The user can configure the number of coefficients and the size of the IP such that they lie within the above limits. The user also inputs the ideal filter response G_{id}, the filter cut-off frequencies and the sampling rate f_s. The evolutionary hardware estimates h_{max} and h_i^{lim}, $1 \leq i \leq p$, as described in Step 1.

Step 3 *IP Generation*

As mentioned earlier, the GA generates a user defined number of chromosomes. Each chromosome is a vector $(h_1, h_2 \ldots h_p)$ of p integers such that $h_i = \pm h_i^{lim}$ (based on the binary switch) in the first iteration and are restrained within $-h_i^{lim} \leq h_i \leq h_i^{lim}$ in subsequent iterations. The h_i's start with h_i^{lim} as initial estimates and slowly evolve towards the required coefficients. It is easy to see that, each chromosome describes a filter albeit with a different frequency response. The following steps describe the GA used in the design.

Step 4 *Genetic Evolution*

There are two evolution stages in the design. At the parallel macro-evolution stage, a *random normal binary switch* toggles the sign of the coefficient limit h_i^{lim} randomly for each of the i^{th} coefficients in the GA search. This introduces more variation in the population. The GA search range is $[0, \pm h_i^{lim}]$ at this stage. At the second micro-evolution stage, we follow a similar methodology but the GA search space in this case is $[-h_i^{lim}, h_i^{lim}]$. A matrix of switched h_i^{max} coefficient sets is formed again. The number of elements of the matrix are user defined to be IP. Each of the randomly switched initial population vectors are used to independently and parallely evolve towards the best fit solution. The following section explains the evolution strategy for each independent vector in the matrix.

(A) Mutation

The variation strategy of the filter involves *random normal perturbation* of the object parameter, h_i, between the preset limits decided by h_i^{lim}. The mutation incorporates progressive reduction and occasional bursts in standard deviations of the perturbation, Eq.(10). This can be seen as analogous to the temperature surges in simulated annealing (SA) and has shown emperically better results than other variation methodlogies.

$$h_i^{'} = h_i + N\left(0, \frac{h_i}{mod(G_{id}^f, i)}\right) \tag{10}$$

where G_{id}^f is the G_{id} gain at frequency f, $h_i^{'}$ is the mutated coefficient value of h_i and $N(\mu, \sigma)$ represents a normal distribution function with mean μ and variance σ as shown in Eq.(11)

$$N(\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{11}$$

Fig.8 shows the standard deviations for the mutation of the coefficients for a symmetric test filter. The test case is of length $p = 21$. Note that the values decrease with the coefficient number. The heuristic for this being that the

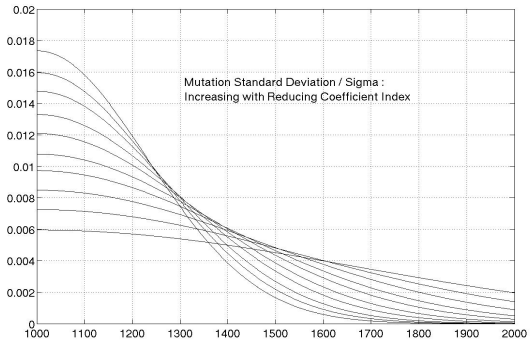

Fig. 8. Decreasing σ with increasing coefficient numbers j in the normal random mutation. The tallest peak stands for the mean coefficient whose deviation is the least and the deviation increases with reducing coefficient index

upper bound of h_{lim}^{j} forms a closer estimate of the ideal filter coefficients for near mean coefficient numbers than the farther ones the deviation between which can be modeled close to (11).

(B) Evaluation

The evaluation of the coefficient set is based on the fitness value computed by summing up the absolute deviations of the evolved gain G_{ev} from the expected gain, G_{id}^{f} over the entire frequency spectrum using the isolated filter hardware.

$$\xi_{curr}(s) = \sum_{i=1}^{M} [G_{ev}(i) - G_{id}(i)] \tag{12}$$

This evaluation shown in Eq.(12) *is unlike in [20] where rather than evaluating the entire frequency spectrum, specific sinusoids are used for fitness evaluation.*

(C) Selection and Termination

The selection of the chromosome set for every, parallel macro-evolution stage is based on the overall fitness criterion which is terminated after a pre-determined best fitness has been reached or when the fitness improvements over the previous generations are not substantial. Eq.(13) is used to estimate δ_{step} for the GA termination.

$$\delta_{step} = \begin{cases} 0 & \text{if } \xi_{curr} < \xi_{best} \\ 0 & \text{if } (\xi_{curr} - \xi_{past}) < \delta_{min} \\ \delta_{step}^{norm} & \text{Otherwise} \end{cases} \tag{13}$$

If the difference between the current fitness estimate, ξ_{curr}, and the best estimate over a pre-determined set of iterations, ξ_{past}, does not exceed δ_{min}, a minimal distinction criterion, the genetic algorithm is terminated. Hence obtaining the required filter coefficients.

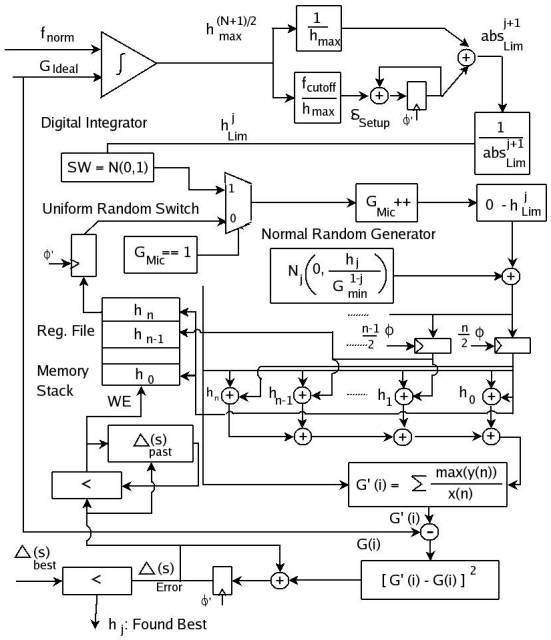

Fig. 9. Macro-Evolutionary System on Chip

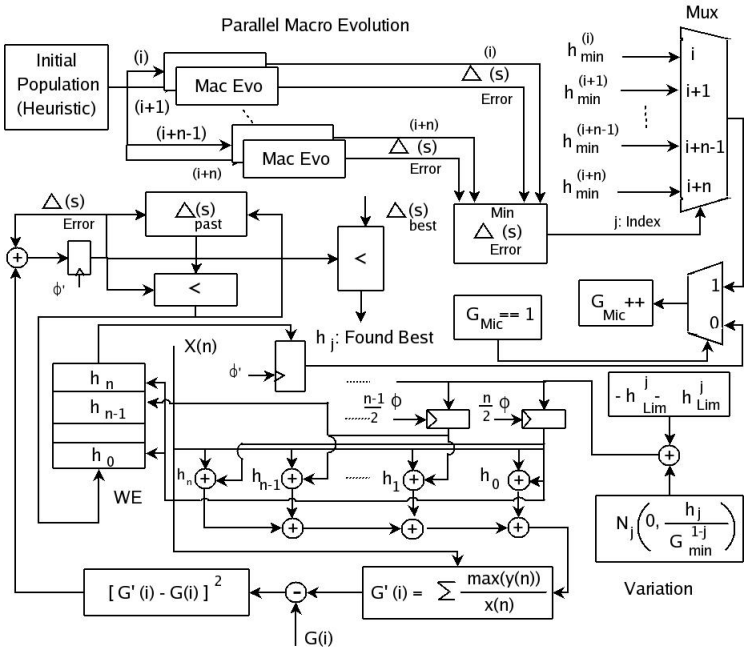

Fig. 10. Parallel and Scalable Micro-Evolution Stage

IV. Intrinsic Design

Intrinsic (or hardware) design of evolvable systems has been an issue of great interest in recent times. However the implementation complexity of traditional GAs make it *very difficult to translate the algorithmic design into hardware*. Insights into approximations and simplified heuristics are essential for effective hardware translation. In [21], Redmill and Bull use the Primitive Operator Directed Graph (PODG) representation to simplify the design. The GA is used to provide a Non Dominated Set (NDS) of solutions. However the entire FIR structure is evolved from scratch. In [23], Miller follows a similar approach using gate arrays. He points out the difficulties in hardware implementation and falls back to extrinsic evolution. *Tufte and Haddow [20], propose the Complete Hardware Evolution (CHE) implementation of the GA pipeline. But their implementation description is only a traditional GA.* The Evolutionary System on Chip (ESoC) serves to address this lacking balance between convergence speeds and implementation practicality. In the following sections, we highlight the implementation mechanism at module level hardware logic for the previously described two stage heuristic evolution of the FIR filter. We demonstrate that the filter is self sustaining and independently evolving. The reconfiguration and adaptation times for the filter are quite practical and the hardware architecture described can form a backbone for evolution of FIR filters in many real life applications. The degree of scalability and the level of parallelism in the design are customizable and provide a flexibility for fast convergence of results keeping the evolutionary genetics practical.

A. The Evolutionary System on Chip (ESoC):

In the following sections, the paper proceeds to present details of the hardware implementation of the two stage evolution mechanism described in the previous section. In a practical Evolutionary System on Chip (ESoC), we have a macro-evolution stage for a faster but coarser convergence and a micro-evolution stage for more accurate coefficient estimates with slower convergence rates.

Macro-Evolution Stage Fig.9 shows the Evolutionary System on Chip (ESoC) architecture that implements the GA. In other words, all the heuristics proposed in the previous sections are translated into hardware and shown implementable on a structured FPGA/ASIC platform.

The population generation block The population generator integrates the ideal gain contour specified by the user $w.r.t$ the normalized frequency to determine the maximum coefficient value. This is then randomly switched using the Normal Random, $N(0,1)$, switch. This is multiplexed through to the filter in the first macro-generation.

The evolution and filtering block The isolated filter, which comes offline temporarily, filters out a stream of spectrum spanning sinusoidal inputs to determine the corresponding outputs. The filter coefficients are the coefficient limits during the first iteration and the best fit coefficients, h_i^{best}, progressively. The mutation of the coefficients occur as in Eq.(10). The normal random generator is again modulated with an intrinsically self-adaptive strategy parameter. The deviation space is $[0, |h_i^{lim}|]$ in every macro-iteration. The filtering block used in the design is the broadcast architecture which provides a pipelined and fast filter implementation to evaluate the fitness span over the entire spectrum of frequencies in realistic time frames.

Evaluation and Selection block The fitness evaluation for multiple frequencies is done using the multiplier and accumulator. This evaluated fitness $\xi_{curr}(s)$ is compared against the best available fitness value $\xi_{best}(s)$ or with the previous fitness value $\xi_{past}(s)$ to either terminate the macro-evolution or to update the coefficient sets h_i^{past} with the current (more) fit coefficients, h_i^{curr}, respectively, Eq.(13). This comparison records the best coefficient set by write enabling (WE) the Coefficient Register File (CRF) from which the coefficients are read out to be used in successive generations.

Micro-Evolution Stage The Micro-evolution stage of the adaptive filter is shown in Fig.10. The parallel ESoC design is a set of Macro-Evolution stages running in parallel over a customized number of parallel processes. These arrive at their own respective fitness values. The minimum fitness, ξ_{min}^j, among them is chosen to be the best fitness and the micro-stage runs another GA very similar to the one in the macro-stage using the $h_i^{best,mac}$ as the initial coefficient estimate. The sole difference between the macro and the micro stages is that the search space for the micro stage is $[-h_i^{lim}, h_i^{lim}]$. This makes this stage a little diverse in search albeit very precise. The convergence of the microstage of evolution is a little slow compared to the macro stage. However this is faster when run in unison with the macrostage than when run alone.

a) Parallelism and Scalability in the design: The proposed design has a high degree of customizable parallelism and scalability. This is reflected in the number of macro-stages which could be used for the GA. The trade-offs in this choice are speed and accuracy. This customizable similarity among the macro and micro stages is advantageous

to the effect that the same filter and hence the GA can be run over multiple cycles providing good hardware re-usability.

b) Built in Error Tolerance (BiET): Isolated self repair is a built in feature of the design. The advantage of the design is that any error occurring in the system coefficients after it has evolved and is running on field can be easily corrected by re-evolving the filter . This provides a robust architecture immune to externalities. This BiET has important applications in noise cancellation and adaptive channel equalization. The filter architecture is summarized

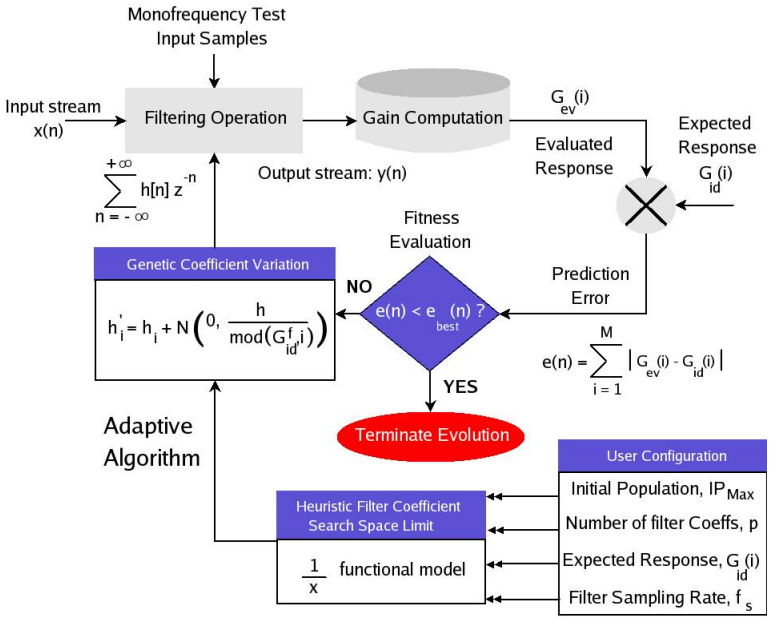

Fig. 11. The evolutionary adaptive filter design including the UI

as a flow diagram in Fig.11. The user inputs the parameters as described in Step 2. The initial coefficient estimation occurs as explained in Step 1. This limits the search space. Step 3 is the IP generation which is controlled by the UI as shown. The genetic evolution step is shown as a separate block. Monofrequency sinusoidal samples spanning the spectrum are input to the filter with the initial estimate of the coeffients (tap weights). The FIR filter operation yeilds an output sample stream $y[n]$ which is used to estimate the gain as shown in Eq.12. The gain is evaluated based on the maximum average output of the stream $y[n]$ when the input stream, $x[n]$ has unit amplitude, mono-frequency, spectrum spanning sinusoids. This evaluated gain is used to estimate the fitness and finally decide the termination as shown in Eq.(13).

V. Experimental results

The results presented here are from the design experiments done using the architecture in Sec.IV. The entire ESoC architecture shown in Fig.9 and Fig.10 was emulated on MathWorks Inc. Matlab Version 7.0.0.19901(R14) and was run on a RHEL Linux workstation with a 3.0 GHz Pentium IV processor. Fig.13 and Fig.14 show the macro and micro fitness progress of some sample filter cases. An initial choice for a test filter was with cut-off

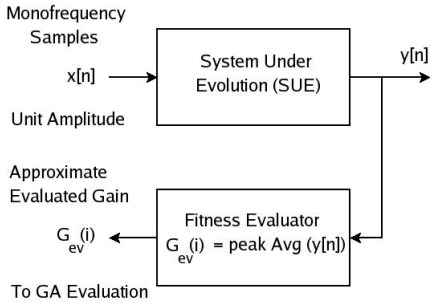

Fig. 12. The fitness evaluator for the system under evolution.

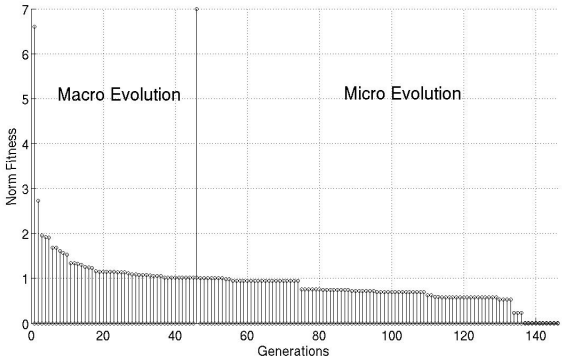

Fig. 13. Macro and Micro fitness evolution progress. Testcase with $p = 21$, $\delta_{s/p} = [15, 250, 15]$ and $f_{s/p} = [0.11, 0.2]$

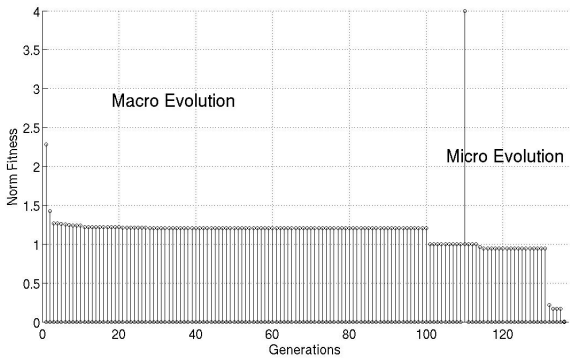

Fig. 14. coarser case with $p = 21$, $\delta_{s/p} = [320, 90, 100, 320]$ and $f_{s/p} = [0.1, 0.25, 0.43, 0.5]$

frequencies, $f_{cutoff} = 0.22, 0.33, 0.44, 0.66$, band gains $A_v = 10.0193, 106.667, 203.314, 300$, length $p = 21$ and sampling frequency $f_s = 18 \times 10^3 Hz$. Fig.15 [1] and Table.I and Table.II show the response characteristics. The algorithm converged within less than 100 generations (compare with 1171 generations in [20]). Fig.15 shows the expected frequency response of the actual filter, the response obtained from the Parks-McClellan algorithm (remez exchange in Matlab) and the evolved filter response using our genetic methodlogy. The response characteristics are within reasonable practical specifications and with an acceptable adaptation latency. Fig.16 shows the macro-fitness evolution where the fitness values evaluated using the algorithm progress *w.r.t* time.

Fig.17 shows the frequency response characteristics and Fig.18 shows the macro-fitness evolution of a band-pass filter case with cut-off frequencies, $f_{cutoff} = 0.1, 0.11, 0.22, 0.33$, band gains $A_v = 10, 300, 300, 10$, length $p = 21$ and sampling frequency $f_s = 18 \times 10^3 Hz$. Fig.21 shows the micro-fitness evolution of the filter. The micro-evolution stage fine tunes the convergence of the macro stage. The refinement is systematic and typically stops in fewer iterations than the macro-evolution stage. Fig.19 shows a low pass filter characteristic, Fig.18 shows the Macro fitness evolution and Fig.22 shows the Micro fitness evolution progress. The filter was designed using

[1] scaled/f_s normalized samples. The fitness is scaled *w.r.t* the best fit value based on the selection criterion in Eq.(13)

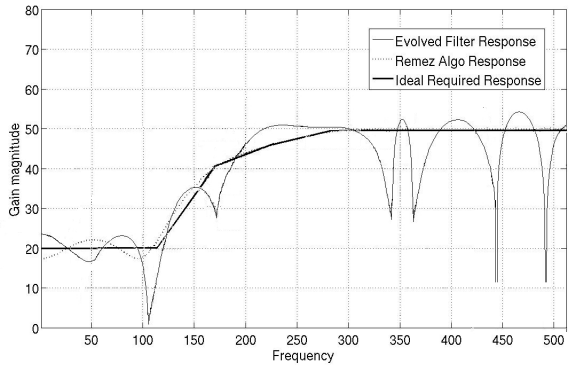

Fig. 15. Frequency response evolution progress. Testcase with coarse spec $p = 21$, $\delta_{s/p} = [10.0193, 106.667, 203.314, 300]$ and $f_{s/p} = [0.22, 0.33, 0.44, 0.66]$

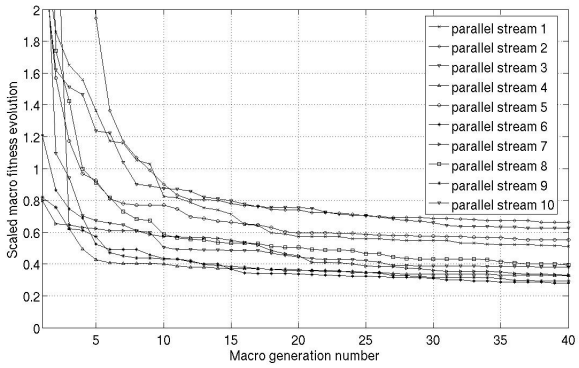

Fig. 16. Macro fitness evolution progress. Testcase with coarse spec $p = 21$, $\delta_{s/p} = [10.0193, 106.667, 203.314, 300]$ and $f_{s/p} = [0.1, 0.11, 0.22, 0.33]$

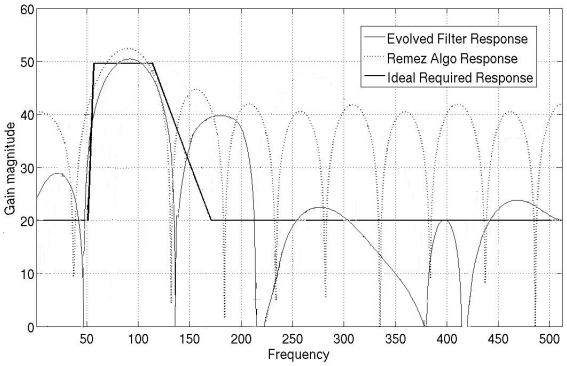

Fig. 17. Frequency response evolution progress. Testcase with $p = 21$, $\delta_{s/p} = [10, 300, 300, 10]$ and $f_{s/p} = [0.1, 0.11, 0.22, 0.33]$

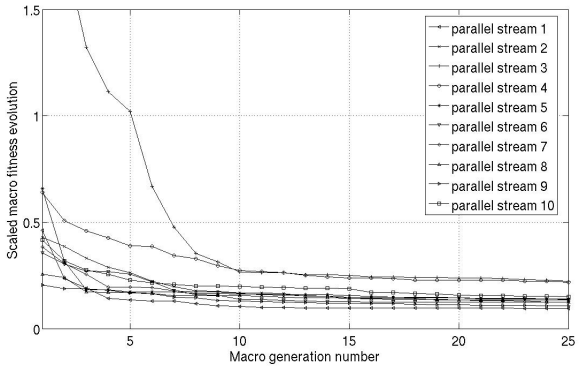

Fig. 18. Macro fitness evolution progress. Testcase with $p = 21$, $\delta_{s/p} = [10, 300, 300, 10]$ and $f_{s/p} = [0.1, 0.11, 0.22, 0.33]$

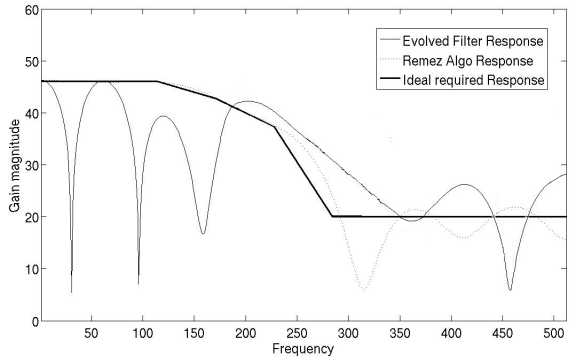

Fig. 19. Frequency response evolution progress. Testcase with $p = 21$, $\delta_{s/p} = [199.9873, 136.6667, 73.3460, 10]$ and $f_{s/p} = [0.22, 0.33, 0.44, 0.67]$

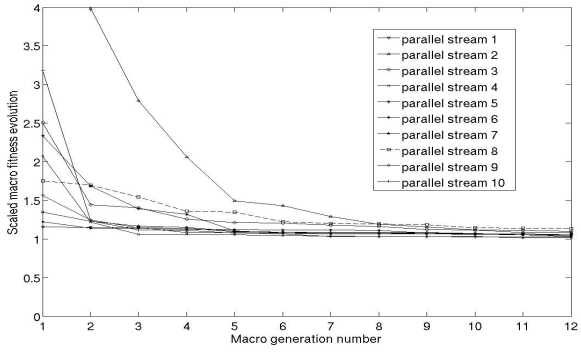

Fig. 20. Macro fitness evolution progress. Testcase with $p = 21$, $\delta_{s/p} = [199.9873, 136.6667, 73.3460, 10]$ and $f_{s/p} = [0.22, 0.33, 0.44, 0.67]$

$f_{cutoff} = 0.22, 0.33, 0.44, 0.67$, band gains $A_v = 199.9873, 136.6667, 73.3460, 10$, length $p = 21$ and sampling frequency $f_s = 18 \times 10^3 Hz$. Implementation of the adaptable genetic filter on a dedicated ASIC would require very short reconfiguration times with an acceptable convergence precision.

Table.II presents the coefficient sets for varying filter lengths and specifications before and after evolution. The table also gives comparisons with the software evaluated coefficients using the approximate remez-exchange algorithm in Matlab. The Parks-McClellan algorithm uses the Remez exchange algorithm and Chebyshev approximation theory to design filters with an optimal fit between the desired and actual frequency responses [27]. The algorithm maximizes the error between the desired frequency response and the actual frequency response characteristic. Filters designed this way exhibit an equiripple behavior in their frequency responses and are sometimes called equiripple filters. [27]. Fig.13 and Fig.14 show the evolution of the filter fitness over the genetic iterations for some sample filters. They demonstrate the evolution speed of the filter coefficients. From the results obtained, we conclude that the filter evolution performs best for sharp cut-off filters. Arbitrary response landscapes are evolved to a far better precision than other approximate algorithmic designs. The major strength in the entire ESoC heuristic however lies in the practicality and robustness of the design which can provide on-line self adaptation within the hardware with reduced iterative effort. The implementation of the genetic method described in this paper is practically feasible since it has simpler operations. This is easily implementable when compared to complex transformations and trigonometric function evaluations in other approximate algorithms like the Parks-McClellan. The complex operations pose a major challenge to the implementation feasibility of these other approximate algorithms. An extension of this work may

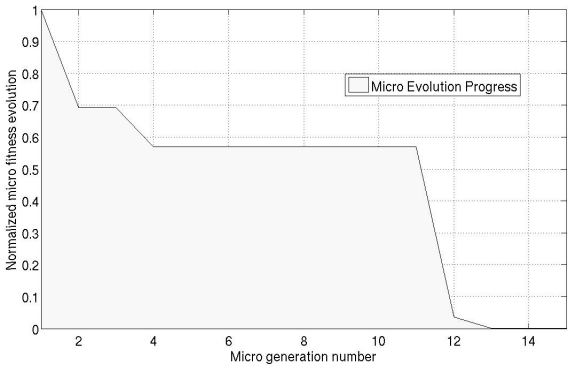

Fig. 21. Macro fitness evolution progress. Testcase with $p = 21$, $\delta_{s/p} = [10, 300, 300, 10]$ and $f_{s/p} = [0.1, 0.11, 0.22, 0.33]$

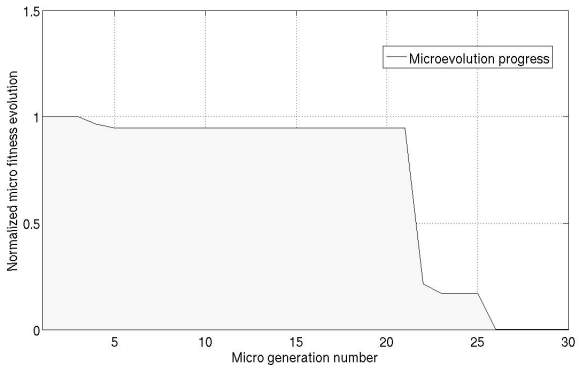

Fig. 22. Micro fitness evolution progress. Testcase with $p = 21$, $\delta_{s/p} = [199.9873, 136.6667, 73.3460, 10]$ and $f_{s/p} = [0.22, 0.33, 0.44, 0.67]$

involve an evident method of reducing the hardware costs by restricting the filter coefficients to integers. This simplifies the filter design without mitigating its performance. This design approach has been explored in many recent research works [1] [6] [21]. This has also involved exploration of the design using evolutionary algorithms. The current work has furthered this design simplification and approach. *The ingenuity of this work lies in the*

deterministic methods presented for generation of the genomic population and mutative variations while keeping faster convergence speeds for filter coefficients making them practical design choices with low implementation complexities and reduced latencies. Also the hardware architecture (ESoC) presented in Sec.IV has a high degree of parallelism and scalability. In [26], a reconfigurable switching architecture for filters is presented which could be effectively used in tandem with the ESoC design of Sec.IV to make the evolutionary switching more effective. In [25], Haseyama and Matsuura present a novel method for tuning the filter coefficients to greater floating point precision. The method incorporates Simulated Annealing (SA) with the GA for coefficient quantization. The results demonstrated in Sec.V from the ESoC design could be bettered by cascading a second stage of the GA-SA tuning routine presented in [25] if the ESoC filter needs to be extended to floating point precision. This may be seen as a design enhancement to the ESoC methodlogy.

TABLE I

COEFFICIENT SETS FOR FILTER TEST CASES

$filt_{testcase}^{(1-5)}$	$h_j, halfset$	p
h_{evo}^j	[26,28,31,0,-1,25,38,-16,-59,-11,84]	21
h_{rem}^j	[-1,2,2,0,0,0,-6,-7,16,55,84]	21
h_{evo}^j	[14,-1,17,18,20,22,25,0,-1,85,105]	21
h_{rem}^j	[-126,62,57,58,62,68,74,79,84,87,108]	21
h_{evo}^j	[61,-41,-1,-36,-76,47,84,26,-130,-20,187]	21
h_{rem}^j	[1,-2,-2,1,0,1,10,11,-24,-84,186]	21
h_{evo}^j	[0,22,33,21,-24,-39,0,-10,-26,0,39]	21
h_{rem}^j	[55,-11,-25,-39,-43,-35,-18,1,17,26,38]	21
h_{evo}^j	[140,-50,157,-77,-33,-90,200,-92,0,-134,282]	21
h_{rem}^j	[-19,4,9,13,15,12,6,0,-6,-9,290]	21

VI. CONCLUSION

The designs and results set out in this paper clearly describe the implementation techniques for self-adaptive FIR filters with an arbitrary response. We use a modified genetic algorithm for the same. Deterministic heuristics in filter evolution at various stages show a praticality of implementation with fast convergence rates. We demonstrate a heuristic based limitation of the initial search space for the filter coefficients(tap weights) reduces the convergence latency. The maximum (mean) coefficient value, the filter cut-off and sampling frequencies and a $\frac{1}{x}$ functional model is used to restrict this search domain. The Genetic Algorithm (GA) used in this work incorporates a mutation of the strategy parameter based on a random normal perturbation which is modulated based on the coeffficient number with a sign selecting switch. The Evolutionary System on Chip (ESoC) design presented in Sec.IV provides a clear,

TABLE II

COEFFICIENT EVOLUTION RESULTS OVER MAC/MIC GENERATIONS FOR TEST FILTER CASES - f_s/p = STOP/PASS BAND CUTOFFS AND $\delta_{s/p}$ = STOP/PASS BAND GAINS

$filt_{testcase}^{(1-5)}$	$f_{s/p}^{(norm)}$	$\delta_{s/p}$	$N_{Mac/Mic}$
h_{lim}^{j}	0,0.22,0.33,0.44,0.67,1.0	200,199.99,136.67,73.4,10,10	-
h_{evo}^{j}	0,0.22,0.33,0.44,0.67,1.0	200,199.99,136.67,73.4,10,10	65/26
h_{rem}^{j}	0,0.22,0.33,0.44,0.67,1.0	200,199.99,136.67,73.4,10,10	-
h_{lim}^{j}	0,0.1,0.1111,1.0	400,400,10,10	-
h_{evo}^{j}	0,0.1,0.1111,1.0	400,400,10,10	46/7
h_{rem}^{j}	0,0.1,0.1111,1.0	400,400,10,10	-
h_{lim}^{j}	0,0.22,0.33,0.44,0.66,1.0	10,10,106.67,203.3,300,300	-
h_{evo}^{j}	0,0.22,0.33,0.44,0.66,1.0	10,10,106.67,203.3,300,300	84/5
h_{rem}^{j}	0,0.22,0.33,0.44,0.66,1.0	10,10,106.67,203.3,300,300	-
h_{lim}^{j}	0,0.1,0.11,0.22,0.33,1.0	10,10,300,300,10,10	-
h_{evo}^{j}	0,0.1,0.11,0.22,0.33,1.0	10,10,300,300,10,10	70/13
h_{rem}^{j}	0,0.1,0.11,0.22,0.33,1.0	10,10,300,300,10,10	-
h_{lim}^{j}	0,0.1,0.11,0.22,0.33,1	300,300,200,200,300,300	-
h_{evo}^{j}	0,0.1,0.11,0.22,0.33,1	300,300,200,200,300,300	126/98
h_{rem}^{j}	0,0.1,0.11,0.22,0.33,1	300,300,200,200,300,300	-

detailed and structural architecture for the implementation of the evolutionary architecture on a standalone hardware platform. The ESoC architecture provides a high level of parallelism and scalability which typically lack in the design of evolutionary systems. The need for practical fault-tolerant adaptive filters is also addressed in this work. The solution presented here is a complete hardware evolution model which is robust and speedily self-adaptive. With the advent of modern safety critical systems, ESoC might be the right direction to take. In this context, the proposed technique has significantly large practical relevance in many real life applications.

REFERENCES

[1] S. Sundaralingam and K. Sharman, "Genetic Evolution of Adaptive Filters,"*Proc. of DSP UK 97*, pp. 47–53, Dec 1997.

[2] Miron Abramovici, John M. Emmert, Charles E. Stroud, Roving Stars, *An Integrated Approach To On-Line Testing, Diagnosis, And Fault Tolerance For Fpgas In Adaptive Computing Systems* Proceedings of the The 3rd NASA/DoD Workshop on Evolvable Hardware, p.73, July 12-14, 2001.

[3] Adams J.W and Wilson A.N.Jr,, "Some efficient digital prefilter structures,"*IEEE Trans.* CAS 31, pp 260–265, 1984.

[4] Saramaki T, Neuvo Y & Mitra S.K, "Design of computationally efficient interpolated FIR filters,"*IEEE Transactions on Circuits & Systems* vol. 35, issue 1, pp. 70–87, Jan 1988.

[5] Wade G, Van-Eetvelt P & Darwen H, "Synthesis of efficient low-order FIR filters from primitive sections,"*IEE Proc* vol.137, pp.367–372, 1990.

[6] Suckley D, "Genetic Algorithms in the Design of FIR Filters,"*IEEE Proceedings-G* vol. 138, pp. 234–238, June 1998.

[7] N.I. Cho & S.U. Lee, "Optimal Design of Finite Precision FIR Filters using linear progr. with reduced constr.," *IEEE Trans. Signal Proc.* vol. 46, pp 195–199, Jan 1998.

[8] D.M. Kodel, "Design of Optimal Finite Precision FIR filters using linear programming techniques," *IEEE Trans. Acoust. Speech, Signal Proc.* vol. ASSp-28, no. 3, pp. 304308, June 1980.

[9] Y.C. Lim, S.R. Parker & A.G. Constantinides, "Finite Word Length FIR Filter Design Using Integer Programming Over a Discrete Coefficient Space," *IEEE Trans. Acoust Speech, Signal Proc.* pp.661-664, Aug 1982.

[10] Bull D.R. and Horrocks D.H, "Primitive Operator Digital Filters," *IEE Proc-G* vol. 138, no. 3, pp. 401412, June 1991.

[11] A. Antoniou, "Digital Signal Processing: Signals, Systems and Filters,"*McGraw-Hill*, 2005.

[12] T.W. Parks & C.S. Burrus, "Digital Filter Design,"*John Wiley* NY, 1987.

[13] W.S. Lu and A. Antoniou, "Two-Dimensional Digital Filters,"*Marcel Dekker* New York, 1992.

[14] D.E. Godlberg, "Genetic Algorithms in Search Optimization and Machine Learning,"*Addison-Wesley,* San Francisco, 1989.

[15] J. Pittman and C.A. Murthy, "Fitting optimal piecewise linear functions using genetic algorithms,"*IEEE Trans.Pattern Analysis and Machine Intel.* vol. 22, no.7, pp. 701–718, July 2000.

[16] J.H. Holland, "Adaptation of Natural and Artificial Systems,"*University of Michigan Press* Ann Arbor, MI, 1975.

[17] K.A. De Jong, "An Analysis of the Behaviour of a Class of Genetic Adaptive Systems,"*PhD Dissertation Dissertation Abstracts International, 36, 10, 5140B*, University Microfilms No. 76-9381, 1975.

[18] Sabbir U.A and A. Antoniou "Design of Digital Filters Using Genetic Algorithms,"*6th IEEE International Symposium on Signal Processing and Information Technology* August 2006.

[19] Horrocks D.H and Khaliffa Y.M, "Genetically derived filter circuits using preferred value components,"*IEE Colloq on Analogue Signal Proc.* Oxford, UK, pp. 4/1–4/5, 1994.

[20] G Tufte and P.C Haddow, "Evolving an adaptive digital filter,"*Proc. of* 2^{nd} *NASA/DoD Workshop on Evolvable HW* pp. 143–150, July 2000.

[21] Redmill D.W. and Bull D.R., "Automated design of low complexity FIR filters ,"*Proc.of the IEEE International Symposium on Circuits and Systems, 1998. ISCAS '98* vol.5, pp 429–432, May-June 1998.

[22] Oppenheim A.V, Schafer R.W and Buck J.R "Discrete time signal Processing,"*Second Edn* Pearson Ed. 2004.

[23] Miller J.F, "An evolvable hardware approach to digital filter design,"*IEE Half-day Colloq. on Evolutionary HW Syst.* pp. 5/1–5/4, June 1999.

[24] Stefatos E.F, Keymeulen D et.al, "An An EHW Arch. for the Design of Unconstrained Low-Power FIR Filters for Sensor Control Using Custom-Reconfig. Tech. ,"*Proc.of the 2005 NASA/DoD Conf. on EHW* pp. 147–153, June-July 2005.

[25] M. Haseyama and D. Matsuura, "A Filter Coefficient. Quantization Method With Genetic Algorithm, Including. Simulated Annealing,"*IEEE Signal Processing Letters* Volume 13, Issue 4, pp. 189–192, April 2006.

[26] J. Torresen, K.A. Vinger, "High Performance Computing by Context Switching Reconfigurable Logic,"*Proc. the 16th European Simulation Multiconference on Modeling and Simulation*, pp. 207–210, 2002.

[27] Rabiner, L.R., J.H. McClellan, and T.W. Parks, "FIR Digital Filter Design Techniques Using Weighted Chebyshev Approximations,"*Proc. IEEE 63 (1975)*