

# Evolutionary Design and Adaptation of Digital Filters Within an Embedded Fault Tolerant Hardware Platform

Ben I. Hounsell and Tughrul Arslan  
Department of Electronics and Electrical Engineering  
The University of Edinburgh  
King's Buildings  
Mayfield Rd  
Edinburgh EH9 3JL  
(Int +44) 131 650 5665  
ben.hounsell@ee.ed.ac.uk

## Abstract

*Finite impulse response filters (FIRs) are crucial devices for robust data communication and manipulation. Multiplierless filters have been shown to produce high performance systems with fast signal processing and reduced area. Furthermore, the distributed architecture inherent in multiplierless filters makes it a suitable candidate for fault tolerant design. Alternative approaches to the design of fault tolerant systems have been proposed using evolutionary algorithms (EAs) and the concept of evolvable hardware (EHW). This paper presents an evolvable hardware platform for the automated design and adaptation of multiplierless digital filters. Filters are realised within a dedicated programmable logic array (PLA). The platform employs a genetic algorithm to autonomously configure the PLA for a given set of coefficients. The ability of the platform to adapt to increasing numbers of faults was investigated through the "evolution" of a 31-tap low-pass FIR filter. Results show that the functionality of filters evolved on the PLA was maintained despite an increasing number of faults covering up to 25% of the PLA area. Additionally, three PLA initialisation methods were investigated to ascertain which produced the fastest fault recovery times. It was shown that seeding a population of random configuration-strings with the best configuration currently obtained resulted in a 6 fold increase in fault recovery speed over other methods investigated.*

## 1 INTRODUCTION

Finite impulse response filters (FIRs) constitute the back-bone of most digital signal processing (DSP) systems,

and are crucial for robust data communication and manipulation. Such devices are frequently employed in environments where issues such as high processing speed, low physical area, and device reliability are highly critical, such as in space applications. For many such applications DSPs must maintain functionality over prolonged periods in harsh environments. Built in reliability of FIR filter devices is therefore required. As a result considerable design resources are poured into innovative realisations of the FIR filter algorithm in hardware. The performance and sustained reliability of hardwired FIR filters are therefore of great importance.

The multiplier is the primary performance constraint when implementing FIR filters structures in hardware as it is costly in terms of area, power and signal delay. However, by fixing the coefficient of each tap, dedicated multipliers can be used which reduce multiplication to a series of bit-shifts and additions/subtractions. As a result the filter remains fixed to a specific transfer function. A number of multiplierless programmable filter architectures have been developed which counter-act this problem. [5, 27]. Multiplierless FIR filter design techniques such as canonic signed digit (CSD) coefficient encoding [17, 11], and primitive operator filter (POF) design [3, 24, 1] have been shown to produce high performance filters with fast signal processing and reduced area. Both rely on the generation of partial products which can then be re-used to generate one or more coefficient terms. However, these design algorithms have not yet been integrated with the programmable filter architectures which would enable optimal configuration of the filter architecture for a given filter specification. Furthermore, the distributed manner of partial product generation inherent in multiplierless FIR filter architectures makes it a suitable candidate for fault tolerant design.

Fault tolerance systems are widely used in space applic-

ations such as commercial satellite communication where hardware deteriorates due to damaged caused by cosmic rays, and in other inhospitable environments where human intervention is difficult or impossible. Systems must therefore maintain functionality despite factors such as severe temperature variation, radiation and operational wear. Conventional fault tolerant VLSI systems employ techniques such as *check-pointing* [21], *concurrent error detection* [14] and *redundancy*. Their purpose is to maintain system operation, or prevent further successive faults by minimising the damage sustained [10]. However, fault tolerant systems are costly as they reduce operational speed and increase physical area.

Alternative approaches to the design of fault tolerant systems have recently been proposed using evolutionary algorithms [6, 2] in the expanding field of evolvable hardware (EHW) [20, 12, 13, 22]. Such approaches provide novel techniques for fault recovery and prevention without the need of additional redundancy, fault detection or diagnosis. Instead a genetic algorithm (GA) is used to monitor system performance and reconfigure aspects of a circuit to counter-act any deleterious faults. Fault tolerant systems which employ EHW must therefore be able to adapt online when required. Hardwired GA's are capable of running considerably faster than those implemented in software on general purpose micro-computers, and are therefore suitable for applications which require online adaptation. As a result, GAs are frequently mapped onto Programmable logic devices (PLDs) so that the fitness function can later be modified for different design criteria [23, 15, 18, 4]. Custom EAs have also been implemented on ASICs [7, 26]. In such cases the fitness algorithm is then set for a specific application.

For EHW to provide a competitive solution to conventional fault tolerant design, EHW resources must be smaller than those required by conventional fault tolerant architectures. The benefits of using a single fixed EHW resource become more apparent as circuit size or complexity increases. Inversely, hardware requirements for conventional fault tolerant designs will continue to grow.

Whilst a number of very novel circuits have been generated using EHW, their complexity and size remain relatively low when evolved using *gate-level components* such as 2-input logic gates (AND, OR etc.). Components of such low granularity create a very large search space, which inhibits the design of large or complex systems. The search space can be reduced by using larger *functional* logic blocks and/or restricting the configuration of components for a specific design task [7, 8]. Although it is argued that reducing the search space in this manner can limit the potential of EHW to generate completely new and novel circuit structures, a compromise must be made between component granularity, flexibility, and the need for EHW to realise

larger, more complex systems if it is to compete with conventional design practices, such as with fault-tolerance.

## 1.1 EHW for Multiplierless FIR Filter Design

This paper presents an evolvable hardware platform for the automated design and adaptation of multiplierless FIR digital filters. Filters are realised within a highly parallel, custom programmable logic array (PLA), designed with in-built redundancy. The platform employs a genetic algorithm to autonomously configure the PLA for a given set of coefficients, whereby the search space is greatly reduced by using dedicated interconnect and functional programmable logic blocks. This provides the flexibility of an adaptive programmable filter, with the performance benefits inherent in a fixed coefficient architecture. The result is a high performance programmable platform dedicated for rapid implementation of fault tolerant filter algorithms. THE PLA architecture itself is suitable for the implementation of multiplierless filter design algorithms including CSD encoding and POF.

The paper is organised as follows: Section 2 presents the embedded fault tolerant PLA platform and the genetic algorithm used to automate filter design and adaptation. Section 3 details a 31-tap low-pass filter used as a design example and for investigation of the PLAs suitability for fault recovery and prevention. Results are obtained through circuit simulation. Evolution is therefore extrinsic. The performance of the filter platform to adapt to an increasingly high number of faults is presented in section 4. Three methods for population initialisation of PLA configuration-strings after a fault is detected are discussed in section 5. Conclusions are then drawn in section 6.

## 2 EMBEDDED FAULT TOLERANT PLATFORM

The embedded architecture presented in this paper has been developed in VHDL (Very High Speed Integrated Circuits, VHSIC, Hardware Description language) such that global parameters can be characterised, providing a scalable processing core which can be ported into larger DSP applications. The designer must therefore decide the data input width, data output width, and the maximum number of taps the platform can support. The latter will determine the dimensions of the PLA.

The filter platform comprises the *PLA*, *genetic algorithm*, and a *system controller*. The system controller interfaces with whatever external hardware or software is required to provide the current filter specification. The current tap-length and coefficient variables are stored in the system controller, which can be updated online for various filter

tasks. Both the coefficients, and the tap-length of the current filter are relayed to the genetic algorithm unit which must then determine how best to configure the PLA. The genetic algorithm also verifies that the PLA core is outputting the desired coefficient set. Each tap within the PLA core is therefore output back to the GA unit.

### 2.1 PLA for FIR Filter Evolution

THE PLA consist of a number of identical programmable arithmetic logic units (PALUs) to create a *coarse-grained* programmable architecture. Each “granular” PALU is able to implement either a parallel n-bit left-shift, addition or subtraction as shown in Figure 1, with a bit-width dependent on the input data width. Five control-bits are required to configure each PALU. The programmable shifter may left-shift from 0 to 3-bits, and requires a 2-bit control. A PALU implementing a shift-by-zero acts as a through-connect. Each PALU output then feeds into a synchronous register to create a pipelined architecture which increases data throughput.

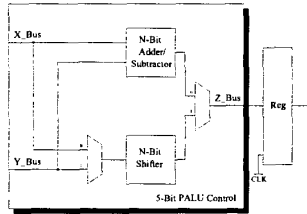


Figure 1. Programmable ALU for Multiplierless FIR Filtering.

An example PLA configuration for a 5-tap FIR filter using the POF design methodology can be seen in Figure 2. PALUs and interconnects which have not been used are shaded out with diagonal lines. It can already be seen that a great deal of redundancy is inherent in the PLA architecture. Redundancy has been shown to be an efficient means of fault tolerant system design [9]. Also note that a number of PALUs simply act as through connect (shift-by-zero) from previous PALU elements.

PALUs are arranged in columns, each connecting to an array of interconnect logic, which in turn connects to the next column of PALUs. A second level of interconnect provides greater connectivity between non-adjacent PALU columns through additional routing of alternate interconnect arrays. This can again be seen in Figure 2. In addition to PALU routing, the interconnect array provides each PALU with a direct connection to the data input, and

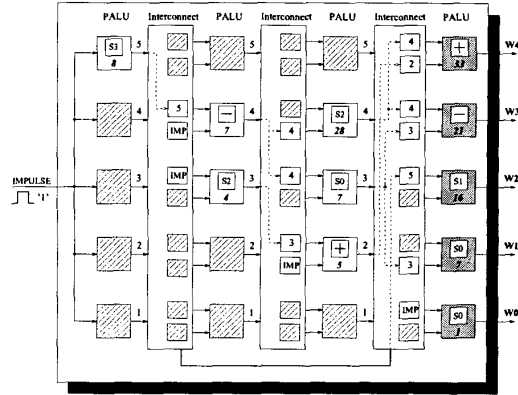


Figure 2. Example PLA configuration of 5-tap primitive operator filter.

to logic '0'. This helps to reduce the number of PALUs utilised, which frees up connectivity. Note that the first column of PALUs are left-shift-only as they directly receive *Data Input* (impulse). These PALUs are capable of shifting between 0 and  $L - 2$  bits, where  $L$  is the input data width.

So as to utilise all PALUs, filter taps are fixed to the final column of the PLA. This configuration is most suited to a transposed direct form filter implementation. A total of  $N + T$  clock cycles is therefore required to process input data, where  $T$  denotes tap length and  $N$  is the number of columns used by the PLA. PLA dimensions are therefore governed by the maximum number of taps required for a filter application.

### 2.2 Evaluating Filter Configurations

Evolutionary algorithms have already been used to optimise filter coefficients along with add and shift operations for multiplierless filter designs[16, 25, 19]. The GA presented in this paper extends this principle to the optimal configuration of hardwired PALUs in order to obtain a set of desired filter coefficients within our dedicated PLA architecture.

The GA was provided with no heuristic knowledge of either the PLA architecture nor any multiplierless FIR filter design methodologies such as POF. The search space is therefore naturally constrained by both the connectivity and dimensions of the PLA architecture, and the add/subtract and shift operations provided by each PALU. The configuration-string required to programme the PLA core is thus manipulated by the genetic algorithm. A pop-

ulation of configuration-strings are then used to produce an optimal PLA configuration for a given set of filter coefficients.

Suitable configuration of the PLA topology was assessed by the GA based on the quality of the filter coefficients presented to it from the PLA core. The *fitness* of each coefficient was calculated by comparing it with the corresponding desired coefficient. The fitness scores of each coefficient were then summed to provide an absolute fitness of the current configuration-string. The success of a given filter was then measured on the ability of the GA to successfully modify the configuration-string over a number of iterations, such that a set of coefficients were obtained which most closely matched those stored in the system controller. One benefit of employing this comparative fitness measure was that it would be simple to implement in VHDL at the RTL level, and would translate easily into hardware.

A generational genetic algorithm was employed, with the following parameters: *population size 100*, *no crossover*, *2-way tournament selection*, and *mutation rate 2/L*; where L is configuration-string length for programming the PLA. This is a derivation of the generalised 1/L mutation rule presented by Mühlenbein. The higher mutation rate was empirically derived as it yielded better results

### 3 FILTER DESIGN EXAMPLE

The suitability of the fault tolerant PLA core for the implementation of filters using EHW was investigated through the development of a 31-tap low pass FIR filter.

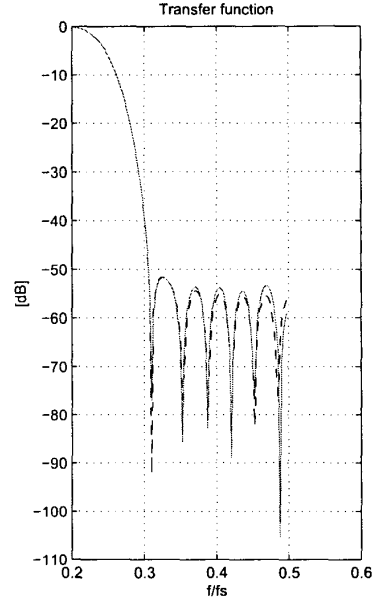
The low-pass filter specification was taken from a suite of components used in the industrial design of a low-power filter core for hearing aids; in joint development with Bernafon LTD and the university of Edinburgh [28]. Filter coefficients are shown in Table 1. All other coefficients are zero.

Coefficient Taps	Dec
$W_{-15}, W_{15}$	-59
$W_{-13}, W_{13}$	96
$W_{-11}, W_{11}$	-220
$W_{-9}, W_9$	461
$W_{-7}, W_7$	-876
$W_{-5}, W_5$	1606
$W_{-3}, W_3$	-3171
$W_{-1}, W_1$	10326
$W_0$	16384

**Table 1. Non-zero coefficients required for response of 31-tap low-pass filter.**

As a result 9 distinct taps are required for a folded form implementation using a 16-bit data-width and 2's complement

encoding. The corresponding filter response is shown by the solid-line in Figure 3.



**Figure 3. Transfer function for 31-tap low-pass FIR Filter**

## 4 INTRODUCING FAULTS

### 4.1 Experimental Setup

The PLA was subjected to four increasingly large numbers of faulty PALUs. These faults covered 0%, 5%, 13% and 25% of the PLA architecture. Each individual fault was obtained by pulling both inputs of a given PLA to zero and setting it to shift-by-zero, effectively simulating a “Stuck a zero” fault. For each increasing percentage of faults the low-pass filter was evolved five times on the PLA using the genetic algorithm. Five randomly generated “populations” of configuration-strings were created for each of the five filters evolved. A total of 6700 iterations were performed by the GA for each of the 5 filters evolved. Few iterations were chosen so that adaptation could occur in “simulated real time”. The fault tolerant hardware platform was simulated using Cadence’s *Leapfrog* VHDL simulation tool. The PLA itself was implemented using 6 columns of PALU and

16 rows. These dimensions were chosen to provide limited redundancy for when faults were introduced into the PLA.

Faults were placed at random for each level of coverage. The same faults were then maintained over the five times the low pass filter was evolved so to obtain an average. Figure 4 displays the topology of PALU faults for each level of fault coverage.

#### 4.2 Analysis

The ability of the fault tolerant hardware platform to adapt to or sustain increasing faults was investigated through four criteria: The fitness of the filter evolved, the number of PALUs used, the degree of PALU re-use (generation of partial products), and the total number of shift, add and subtract operations required to implement the desired coefficients. Results are shown in Figure 5.

Figure 5 reveals that at PALU faults from 0 to 13%, the genetic algorithm is in each case able to evolve a filter with a maximal fitness of 99.9%. When the PLA is 25% faulty a 0.5% decrease in the fittest filter solution is incurred. The average fitness of filter coefficients evolved remain above 98.5% until 25% of the PLA experiences faults. A fitness  $\geq 98.5\%$  was required to produce a transfer function with acceptable low-pass characteristics and a gain no less than -52 dB. An example of a typically acceptable response for the low-pass filter is shown by the dashed-line in Figure 3. Variation of the least fit filters evolved by the GA is more marked as the percentage of faults in the PLA increases.

The average number of PALUs used to implement the low pass filter reduces as the percentage of faults found in the PLA increases. However, only a 10% reduction in PALU usage is experienced on the PLA despite a 25% decrease in the number of functional PALUs available. This suggests that the PLA provides the GA with a means of counter-acting the deleterious effects of PALUs which are "stuck at zero". Comparisons between figures 5(a) and 5(b) reveal that as fewer PALU resources are made available to the GA through faults, a reduction in filter performance is experienced. The number of PALUs required to implement the filter therefore relates directly to the fitness of the evolved solution.

Accounting for variations of faults at 5 and 13% of the PLA area, the average number of PALUs reused within the PLA (to generate partial product terms) remains relatively constant, with an overall reduction of less than 5%. Again this supports the notion of a robust PLA architecture capable of providing an adaptable environment for fault tolerant design using EHW.

Regardless the degree of faulty PALUs, the number of shifter operations selected by the GA is double that of either additions or subtractions. This is encouraging as shifts consume little power and are considered to be the primary

means of product generation when using multiplierless filter techniques such as POF design.

Of course the placement of faulty PALUs will greatly effect the ability of the GA to evolve high quality filters. Faults close to, or directly on PALUs which are connected to coefficient taps will have a more detrimental effect than those distributed in the centre of the PLA. This could account for the poorer average fitness performance of filters generated on the PLA with a faulty area of 5% compared to those generate by the GA on a PLA with 13% faults. Given the added number of redundant rows present in the PLA architecture it would be possible to adapt the platform such that taps could be moved to other PALUs which are not near faults, or are themselves faulty, but still located on the final column.

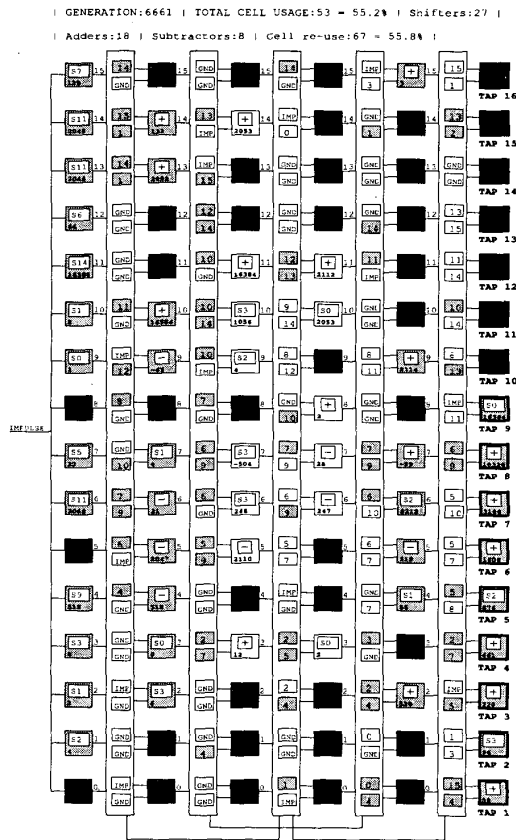
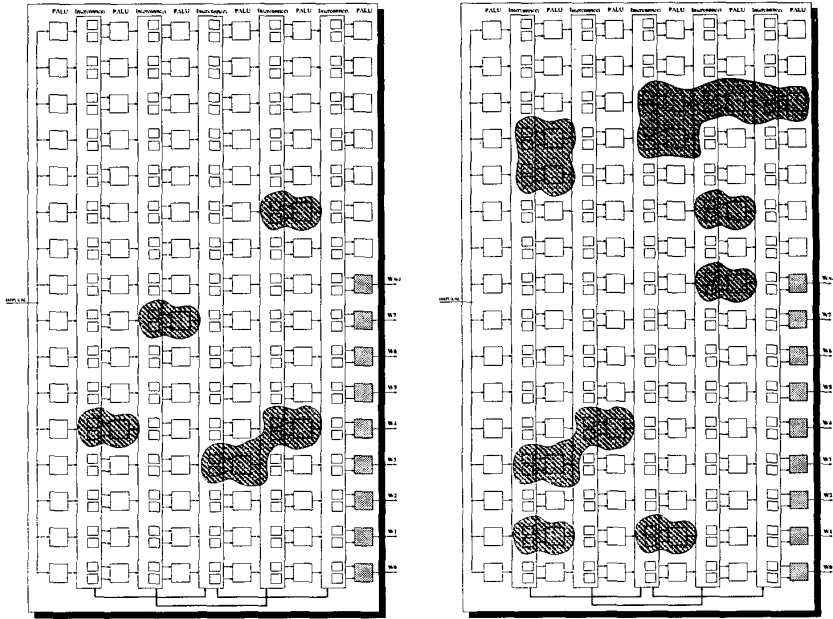
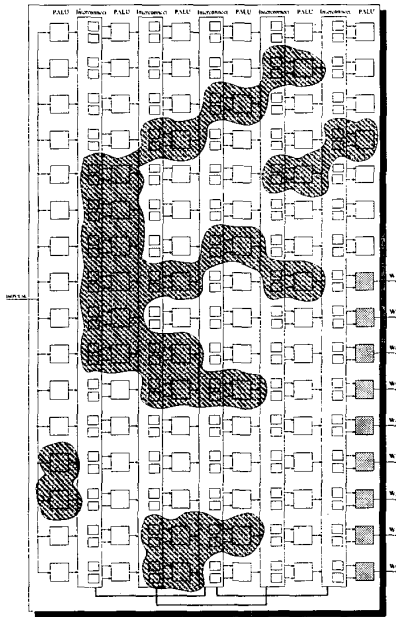


Figure 6. Schematic showing configuration of low-pass FIR filter on PLA with 13% faults.



(a) 5% PALU faults.

(b) 13% PALU faults.



(c) 25% PALU faults.

Figure 4. "Stuck-at-Zero" fault topologies covering PLA

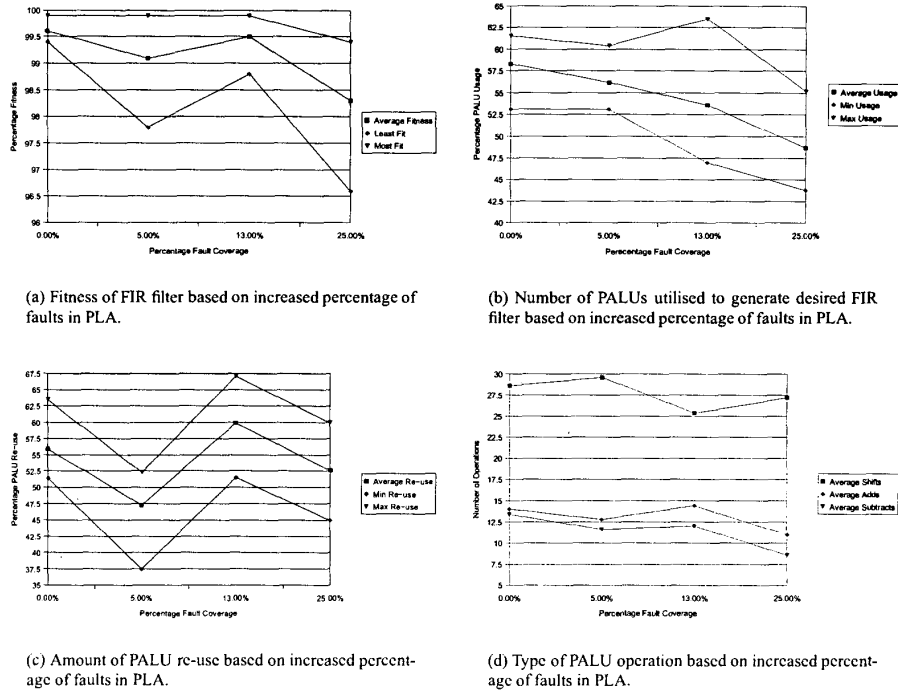


Figure 5. PLA architectures with increasing percentages of faulty PALUs

An example PLA configuration of the 31-tap low pass filter evolved with 99.9% correctness and 13% of its PALUs faulty is illustrated in Figure 6. Additional models of faults besides “stuck-at-zero” are currently under investigation, and will focus specifically on faults commonly associated with PLDs. The ability of the PLA to adapt to these faults will then be investigated.

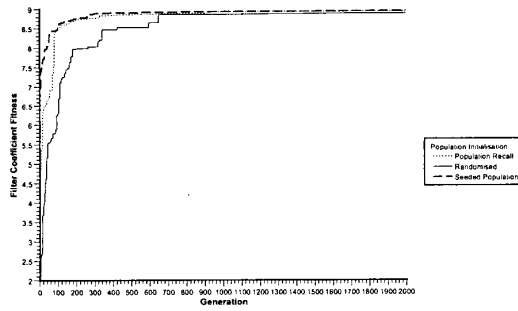
## 5 POPULATION INITIALISATION AFTER FAULT DETECTION

In each of the scenarios studied in section 4, populations of configuration-strings were randomly generated. However, two other approaches to population initialisation are available. In this paper they are termed *Population seeding*, and *Population recall*. Both of these approaches were investigated to see if fault recovery times after detection could be decreased when compared to random population initialisation. The method of population seeding applied in this paper involved taking the fittest solution stored from the previous evolutionary run (and currently in operation

on the PLA), and placing it into a population of 99 randomly generated configuration-strings. Population recall simply involves re-introducing the most recent population of configuration-strings evolved, and using this as the initial start point.

The effectiveness of both approaches was examined using the same PLA topology with 13% of its PALUs “stuck-at-zero”, as shown in Figure 4(b). So as to obtain an averaged performance the same low-pass filter was evolved five times using the population seeding approach. In each case the PLA configuration shown in Figure 6 was used as the seed. All other configuration-strings were randomly generated. A randomly selected final population, evolved for the low-pass filter with no faults in the PLA, was used as the initial population for the recall approach. As no configuration-string needed to be randomly generated this scenario was run only once. The performance of each initialisation approach was determined to be the number of generations required by the GA to produce a filter response with a fitness  $\geq 98.5\%$ . Figure 7 displays the results obtained. The averaged evolution of filter fitness using random population

initialisation has also been shown.



**Figure 7. Fitness performance of filter evolved on PLA based on various methods of generating the initial population of configuration-strings.**

Both population seeding and recall enable the GA to adapt to the faulty PLA architecture and produce filters with target fitness considerably faster than when a population of configuration-strings are randomly generated. Population seeding produces filters of fitness  $\geq 98.5\%$  after 272 generations, population recall required 513 generations, whilst random initialisation took an average of 1521 generations to reach target fitness. This translates to a 6 fold and 3 fold increase in fault recovery over that of random initialisation for population seeding and recall respectively.

## 6 CONCLUSION

An embedded fault tolerant hardware platform for the automated design of multiplierless digital filters has been presented. The platform uses the principle of evolvable hardware (EHW) to automate the configuration of programmable arithmetic logic units (PALUs) capable of implementing either left-shifts, additions or subtractions. These PALUs form a programmable logic array (PLA) dedicated to the implementation of multiplierless filters. The ability of the platform to adapt to increasing numbers of faults was investigated through the “evolution” of a 31-tap low-pass FIR filter. Four increasingly large numbers of faulty PALUs were introduced to the PLA from 0 to 25% of the total architecture. Results show that the functionality of filters evolved on the PLA was maintained despite the increasing number of faults present. This was attributed to redundant PALUs (inherent in the PLA) exploited through the use of EHW. Two additional methods of population initialisation were examined to see if fault recovery times after

detection could be decreased compared with random population initialisation. It was shown that seeding a population of random configuration-strings with the best configuration currently obtained produced filters of acceptable fitness 6 times faster than with purely randomised population initialisation.

Further work towards including heuristic knowledge about multiplierless filter design, such as POF and CSD, and about the dedicated PLA architecture is currently underway. The aim is to produce “optimally” designed filters on the PLA and further increase the speed in which the platform can adapt to system faults for real time applications.

## References

- [1] T. Arslan, H. I. Eskikurt, and D. H. Horrocks. Configurable structures for a primitive operator digital filter fpga. In *IEEE Workshop on Signal Processing Systems, SIPS 97 - Design and Implementation*, pages 532–540, 1997.
- [2] T. Bäck, D. B. Fogel, and T. Michalewicz, editors. *Evolutionary Computation 1*. INSTITUTE OF PHYSICS PUBLISHING, Bristol and Philadelphia, 2000.
- [3] D. R. Bull and D. H. Horrocks. Primitive operator digital filters. *IEE Proc -G*, pages 401–412, 1991.
- [4] Y.-H. Choi and D. J. Chung. Vlsi processor of parallel genetic algorithm. In *Proceedings of the Second IEEE Asia Pacific Conference on ASICs, AP-ASIC 2000*, pages 143–146, 2000.
- [5] J. B. Evans. An efficient fir filter architecture. In *Int. Symposium. Acoust., Speech, Signal Processing*, volume 1, pages 627–630, 1993.
- [6] D. E. Goldberg. *Genetic algorithms in search optimization and machine learning*. Addison-Wesley, Reading, Mass, 1989.
- [7] T. Higuchi, M. Masahiro, M. Iwata, I. Kajitani, W. Liu, and M. Salami. Evolvable hardware at functional level. In *IEEE International Conference on Evolutionary Computation*, pages 187–192, 1997.
- [8] B. I. Hounsell and T. Arslan. A novel genetic algorithm for the automated design of performance driven digital circuits. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 1, pages 601–608, July 2000.
- [9] R. Karri, K. Hogstedt, and A. Orailoglu. Rapid prototyping of fault tolerant vlsi systems. In *Proceedings of the 7th International Symposium on High-Level Synthesis*, pages High-Level Synthesis, 1994.
- [10] S. Levi and A. K. Agrawala. *Fault tolerant system design*. McGraw-Hill, 1994.
- [11] Y. C. Lim and S. R. Parker. Fir filter design over a discrete powers-of-two coefficient space. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-31:583–591, June 1993.
- [12] J. D. Lohn, G. L. Haith, S. P. Colombano, and D. Stassinopoulos. Towards evolving circuits for autonomous space applications. In *IEEE Aerospace Conference Proceedings*, volume 5, pages 473–486, 2000.
- [13] T.-S. Park, C.-H. Lee, and D.-J. Chung. Intrinsic evolution for synthesis of fault recoverable circuit. *IEICE Trans. Fundamentals*, E83-A(12):2488–2497, Dec 2000.

- [14] J. H. Patel and L. Y. Fung. Concurrent error detection in alu's by recomputing with shifted operands. *IEEE Trans. Computers*, 31(7):589–595, July 1982.
- [15] R. Porter, k. McCabe, and N. Bergmann. An applications approach to evolvable hardware. In *Proceedings of the First NASA/DoD Workshop on Evolvable Hardware*, pages 170–174, 1999.
- [16] D. W. Redmill, D. R. Bull, and E. Dagless. Genetic synthesis of reduced complexity filters and filter banks using primitive operator directed graphs. *IEE Proc. -Circuits Devices Syst*, 147(5):303–310, October 2000.
- [17] H. Samueli. An improved search algorithm for the design of multiplierless fir filters with powers-of-two coefficients. *EEE Trans. Circuits Syst*, 36(7):1044–1047, July 1989.
- [18] M. Sipper, M. Goeke, D. Mange, A. Stauffer, E. Sanchez, and M. Tomassini. The firefly machine: online evolware. In *IEEE International Conference on Evolutionary Computation*, pages 181–186, 1997.
- [19] S. Sriranganathan, D. R. Bull, and D. W. Redmill. Design of 2-d multiplierless fir filters using genetic algorithms. In *GALESIA. First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 282–286, 1995.
- [20] A. Stocia, D. Keymeulen, V. Duong, and C. Salazar-Lazaro. Automatic synthesis and fault-tolerant experiments on an evolvable hardware platform. In *IEEE Aerospace Conference Proceedings*, volume 5, pages 465–471, 2000. I'll write something here.
- [21] Y. Tamir and M. Tremblay. High performance fault-tolerant vlsi systems using micro rollback. *IEEE Trans. Computers*, 39(4):548–554, April 1990.
- [22] A. Thompson. Evolving fault tolerant systems. In *GALESIA. First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 524–529, 1995.
- [23] G. Tufte and P. C. Haddow. Evolving and adaptive filter. In *Proceedings of the Second NASA/DoD Workshop on Evolvable Hardware*, pages 143–150, 2000.
- [24] G. Wacey and D. R. Bull. Architectural synthesis of digital filters for asic implementation. In *IEE Colloquium on Digital and Analogue Filter and Filtering Systems*, pages 6/1–6/7, 1991.
- [25] G. Wade, A. Roberts, and G. Williams. Multiplier-less fir filter design using a genetic algorithm. *IEE Proceedings on Vision, Image and Signal Processing*, 141(3):175–180, 1994.
- [26] S. Wakabayashi, T. Koide, N. Toshine, M. Goto, Y. Nakayama, and K. Hayya. An lsi implementation of an adaptive genetic algorithm with on-the-fly crossover operator selection. In *Proceedings of the ASP-DAC '99. Asia and South Pacific Design Automation Conference*, volume 1, pages 37–40, 1999.
- [27] S. Yoon and M. H. Sunwoo. An efficient multiplierless fir filter chip with variable-length taps. In *IEEE Workshop on Signal Processing Systems. SIPS 97 - Design and Implementation*, volume 1, pages 412–420, 1997.
- [28] E. Zwyssig. Low power digital filter design for hearing aid applications. Master's thesis, The University of Edinburgh UK, 2000.