

AN EMBEDDED PROGRAMMABLE CORE FOR THE IMPLEMENTATION OF HIGH PERFORMANCE DIGITAL FILTERS

Ben I. Hounsell, Tughrul Arslan, *

Department of Electrical Engineering,
The University of Edinburgh,
The King's Buildings, Mayfield Road,
Edinburgh EH9 3JL, Scotland, UK
Phone: +44 131 650 5619 FAX: +44 131 650 6554
Email: biho@ee.ed.ac.uk

ABSTRACT

This paper presents a novel, embedded *programmable* logic array (PLA) for implementing high performance filter functions. A *custom-built* configurable architecture together with a *fast* and *local* interconnect hierarchy, provides a high degree of flexibility for realisation of a given filter specification. The PLA is designed to be reconfigured with external hardware or software all in a system-on-chip platform. A stochastic algorithm is used in this paper to demonstrate the automated configuration of the PLA architecture for a practical filter example. The highly parallel nature of the PLA architecture ensures scalability for complex filter tasks, and provides a highly fault tolerant platform for embedded filter applications. Investigations show that even when 20% of the PLA architecture is damaged the same filter example can still be successfully implemented. Further results demonstrate the scalability, speed, and array utilisation of the architecture using a typical SoC bus specification.

1 INTRODUCTION

With the advent of system-on-chip (SoC) devices, there is an increasing demand for high performance, dedicated, programmable Digital Signal Processor (DSP) cores. These cores must be suitable for embedding in SoC platforms, for a wide range of applications. Finite impulse response (FIR) filters are used in many of these applications, ranging from audio data manipulation, to channel equalisation. General purpose DSPs, such as the TMS320 series from Texas Instruments, do not provide sufficient throughput to implement high speed FIR filters, due to their single multiplier architecture. General purpose FPGAs, such as those from Xilinx [1], are suitable for implementing dedicated filter architectures. However, the general functionality of FPGAs result in complex configurable logic blocks

*Tughrul Arslan is also affiliated with the Institute for System Level Integration (ISLI), Livingston, Scotland UK

(CLBs) which require a high degree of interconnect. This restricts circuit throughput and increases the embedded filter core size within the total SoC platform area.

Multiplierless filter design techniques have been shown to produce high performance architectures beyond those attainable through implementation on general purpose DSPs [2, 3]. Performance increases are achieved by reducing multiplication to a series of bit-shifts and additions/subtractions, for a specific set of filter coefficients. Bull et.al. introduced the concept of primitive operator filters (POFs), which utilise directed graphs to optimise resources for a desired set of filter coefficients, using the multiplierless filter approach [4, 5, 6]. This technique represents coefficients in standard 2's complement, making it widely suitable for many DSP applications. A configurable architecture has been proposed for the implementation of POF in hardware [6]. However, this architecture uses a single bus which therefore produces a bottle-neck and restricts throughput. In addition, the size and complexity of the CLBs used severely limits the scalability of the architecture, as CLBs do not efficiently map into filter coefficients. Additional, novel VLSI implementations of multiplierless FIR filters range from circuits capable of sampling frequencies from 313kHz to 120 MHz, and filter orders from 32 to 64 taps [7, 8, 9, 10]. However, these architectures are tailored to a specific set of fixed coefficients and are therefore not programmable or easily reusable.

This paper presents a novel programmable logic array (PLA) dedicated for implementing high performance multiplierless filters. A custom CLB architecture together with a *fast* and *local* interconnect hierarchy, provides a high degree of flexibility for realisation of a given filter specification. The highly parallel nature of the PLA ensures that the architecture scales linearly with filter complexity. These characteristics make it ideal for implementation as an embedded core for adaptive applications where the architecture can be reconfigured with external hardware or

software. The PLAs flexibility and suitability for implementing filters is demonstrated through the utilisation of a stochastic algorithm to autonomously configure the PLA core for an example filter application. The ability of the PLA to maintain filter functionality even when 20% of the architecture is damaged has also been investigated; and further distinguishes our PLA from other more general purpose DSP architectures.

The paper is organised as follows: Section 2 presents an overview of the programmable core architecture, and details the interconnect hierarchy. Section 3 explains the CLB architecture and presents an example PLA configuration using the POF design approach. The stochastic algorithm is introduced in section 4, and the automated configuration of the PLA for a practical filter examined. Faults are then introduced onto the PLA architecture and its robustness investigated for the same filter example. Comparative results are then presented. Section 5 describes both the physical constraints and performance of the PLA core when synthesised to operate within a typical SoC bus architecture. Conclusions are drawn in section 6.

2 PLA ARCHITECTURE

A PLA core comprises 5 columns and 6 rows of programmable arithmetic logic units (PALUs). These dimensions were empirically derived so as to produce a flexible architecture with low signal latency. The final PALU column is used to output coefficient taps. The PLA therefore implements filters in the transposed direct form [11]. Figure 1 presents both the PLA core architecture and an external stochastic control algorithm, designed to automate the configuration of the PLA for a given filter application. The control algorithm is discussed in greater detail in section 3 of this paper.

Input ' $X(n)$ ', output ' $Y(n)$ ' and coefficient data-widths are specified by the designer. Multiple PLA cores can be instantiated during parameterisation in order to produce a PLA architecture capable of implementing filters with a maximum of $5 \times N$ taps. Where N denotes the number of PLA cores to be connected. PALUs are arranged in columns, each connecting to an array of programmable interconnect logic, which in turn connects to the next column of PALUs. A second level of *fast interconnect* provides greater connectivity between non-adjacent PALU columns. Interconnect arrays also provide each PALU with a direct connection to ' $X(n)$ ', and to logic '0'. An example of the hierarchical interconnect strategy can be seen in Figure 2. 3-bits of control logic are required to programme the interconnect for each PALU input. The entire PLA core is configured serially via a serial-in-parallel-out shift register of bit-length S .

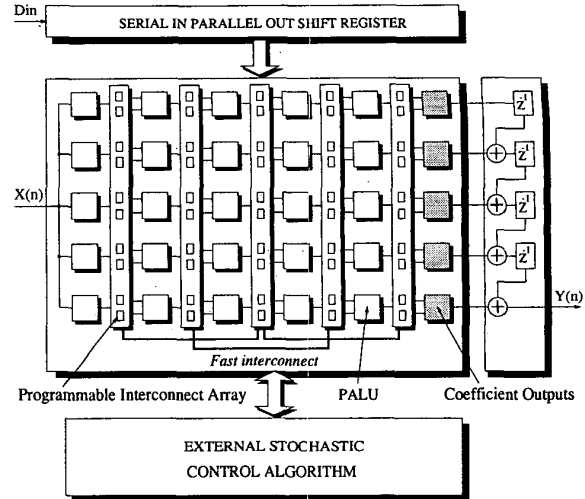


Figure 1: System overview of programmable filter architecture.

3 PALU ARCHITECTURE

Figure 2 also displays the structure of our PALU element. Each PALU is able to implement either a parallel left-shift (SH), addition or subtraction (AoS). Bit-width ' L ' is dependent on the input data width of ' $X(n)$ '. 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 therefore acts as a through-connect. The first column of PALUs are left-shift only and are capable of shifting between 0 and $L - 2$ bits. Each PALU output connects to a synchronous register (R), creating a pipelined architecture to enable fast data throughput. A total of $C + 1$ clock cycles are therefore required before filtered data will be present on ' $Y(n)$ ', where $C = 6$ and is the number of PALU columns. As a result, the throughput of the PLA is *not* effected by filter length. This is a considerable advantage over single multiplier filter architectures.

An example PLA configuration for a 5-tap FIR filter using the POF design methodology can be seen in Figure 3. PALUs and interconnects which have not been used are shaded out with diagonal lines. It can 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 [12, 13]. Also note that a number of PALUs simply act as through connect (shift-by-zero) from previous PALU elements.

4 AUTOMATED DESIGN EXAMPLE

A number of practical filters have been implemented on our PLA. This paper presents the implementation of a 24-tap low-pass filter, designed using the Remez exchange al-

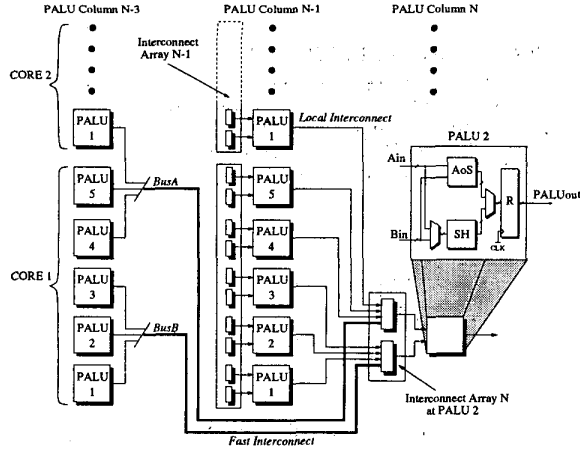


Figure 2: Programmable routing between PALU elements showing local and fast connectivity

gorithm developed by McClellan et.al. and benchmarked in [14]. The filter was implemented on the PLA in folded form and in 16-bit 2's complement. Three PLA cores were connected together in order to provide the 12 taps needed to implement the low-pass filter.

A genetic algorithm (GA) was incorporated into the external control algorithm to autonomously configure the PLA for the desired filter specification. GAs are stochastic optimisation techniques which perform an iterative, non-heuristic search through a space of possible solutions (in this case the correct configuration string to implement a 24-tap low-pass filter on our PLA) [15, 16]. A number of solutions are investigated concurrently and comprise a *population* of competing individuals, or *chromosomes*. Each chromosome is evaluated and a *fitness* assigned which is representative of the quality of the solution it describes. The fitness of each filter was calculated by comparing the desired coefficient of each tap with that produced on the relevant output of the PLA, formalised as follows:

$$Q_x = \sum_{i=1}^T \begin{cases} f_i/F_i & \text{if } f_i \leq F_i \\ F_i/f_i & \text{otherwise} \end{cases} \quad (1)$$

Where Q is the final "fitness score", T is the total number of taps, f_i is the PLA output of the current tap and F_i is the desired current coefficient. The GA was given no heuristic knowledge of the PLA architecture or multiplierless filter design techniques, so as to provide an independent measure of the cores suitability for implementing FIR filters. The following GA parameters were used: Mutation, $m = 2/S$, two-way tournament selection, population size 100 and no crossover. The low-pass filter was configured ten times on the PLA using the genetic algorithm to obtain a per-

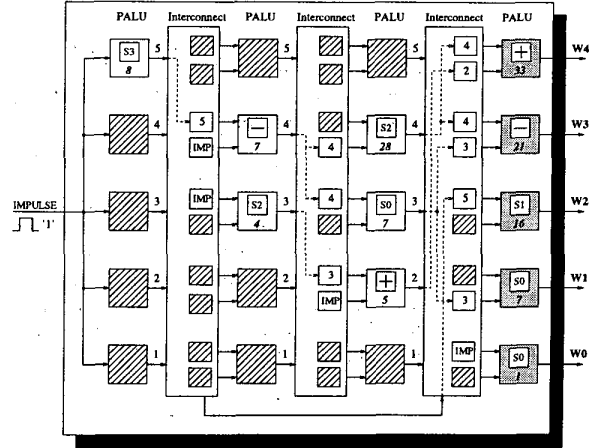


Figure 3: Example PLA configuration of 5-tap primitive operator filter.

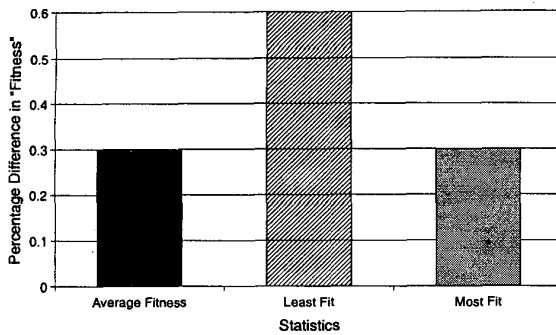
formance average. Ten randomly generated "populations" of configuration-strings were created for each automated configuration. A total of 5500 iterations were performed by the GA for each of the ten filters implemented. The entire platform was simulated using Cadence's *Leapfrog* VHDL simulation tool. An example PLA configuration can be seen in Figure 4.

4.1 Investigation of PLA Robustness

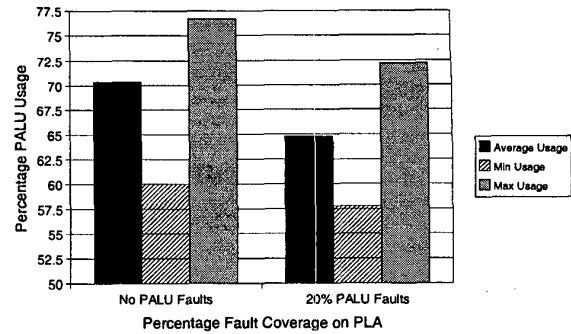
On average, approximately 70% of the PALUs were required to implement the low-pass filter. This provides the PLA with sufficient free elements to be reconfigured should part of the architecture become damaged. To investigate the effectiveness of the PLA to maintain filter functionality despite damage, 20% of the architectures PALUs were randomly made faulty. Each individual fault was obtained by pulling both inputs of a given PALU to zero and setting it to shift-by-zero, effectively simulating a "Stuck a zero" fault. The PLA itself was again implemented using 3 instantiated cores. The same fault locations were then maintained over the ten times the low-pass filter was configured, to obtain an average of the PLAs effectiveness as a fault tolerant architecture. Figure 5 displays the topology of the PALU faults covering 20% of the PLA architecture.

4.2 Results

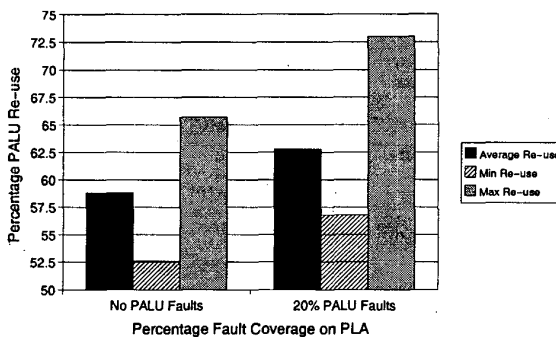
Figure 6 displays the performance of the PLA when implementing the low-pass filter; resulting from both its configuration by the GA, and the inclusion of faulty PALUs. It can be seen from Figure 6(a) that the quality of filters implemented on the faulty PLA is on average only 0.3% lower than when no faults are present. This degradation in *fitness* has a negligible effect on filter performance. Fig-



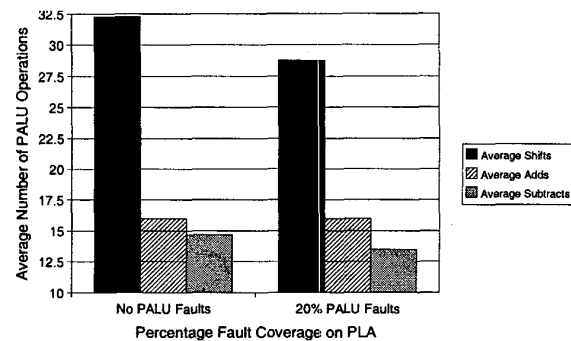
(a) Comparative difference of success of implementing FIR filter on PLA with 20% fault coverage against PLA with no faults.



(b) PALU usage (coverage) required by FIR filter.



(c) PALU re-use exploited by FIR filter.



(d) Operations performed by PALU to implement FIR filter.

Figure 6: Performance of PLA to autonomously generate a 24-tap low-pass FIR filter with an without faults.

ure 6(b) shows that approximately 5% fewer PALUs were, on average, used to implemented the low-pass filter when the PLA was faulty. This is considerably smaller than the 20% loss of functionality experienced on the PLA, and demonstrates the potential benefits resulting from the high degree of parallelism inherent in the architecture.

Figure 6(c) details the number of PALUs which are re-used (to generate partial product terms) with and without faults on the architecture. Between 4-7% more PALUs are re-used when the PLA is damaged. This figure is relatively small, again when compared to the high number of PALUs which were dysfunctional, and demonstrates the flexibility of the architecture to provide possible configurations which maintain filter performance.

Approximately the same number of addition and subtraction operations were performed on the PLA to produce the required filter response, before and after faults were introduced. Fewer shifts were implemented after 20% of the PALUs were made faulty. This accounts for the primary difference in the number of PALUs required to implement

the filter when compared to the non-faulty PLA. In both examples the number of shift operations is approximately double that of additions and subtractions. This is particularly desirable as shifts consume little power and area, and are the primary means of product generation in multiplier-less filter design techniques such as POF [4].

5 SYNTHESIS OF PLA CORE

The PLA core has been synthesised from behavioural VHDL using *Synopsys Design Analyzer* and Alcatel's 0.35 μ m MTC45000 technology library. In order to determine the scalability of our PLA core, the architecture was synthesised at clock frequencies of 10, 25, 50, 80, 90 and 100 MHz, with all data widths set at 16-bits. Theses frequencies were chosen to reflect typical timing constraints required on high speed SoC bus architectures, which range from 60 to 100MHz. The resulting logic area (in NAND gates) can be seen in Figure 7. Area remains relatively constant up to 50MHz. Between 50 and 100MHz area increase is approximately linear. For technologies

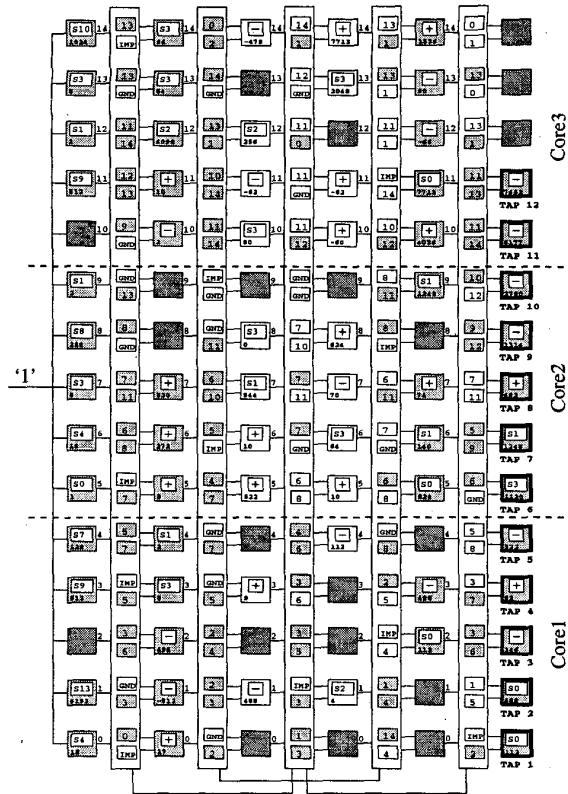


Figure 4: Example PLA configuration for implementation of 24-tap low-pass filter. Blank PALUs show that the logic is not used.

smaller than $0.35\mu\text{m}$, faster throughput could be achieved.

The critical timing path was found to lie between the *BusA/BusB* input of any given set of interconnect logic, and the output register of the PALU which is associated with this interconnect. This is explained in more detail in Figure 8. The largest delay is incurred through the adder/subtractor unit of the PALU. One way to reduce this would be to customise the adder/subtractor block for a specific silicon technology. This would limit the general portability of the PLA core, but further increase its performance.

6 CONCLUSIONS

This paper presents a novel, embedded programmable core for implementing high performance filter functions. Multiplication is replaced with bit-shifts, additions and subtractions, using custom programmable arithmetic logic units (PALUs). A *fast* and *local* interconnect hierarchy, provides a high degree of flexibility for realisation of a given filter specification. Coefficients are generated within

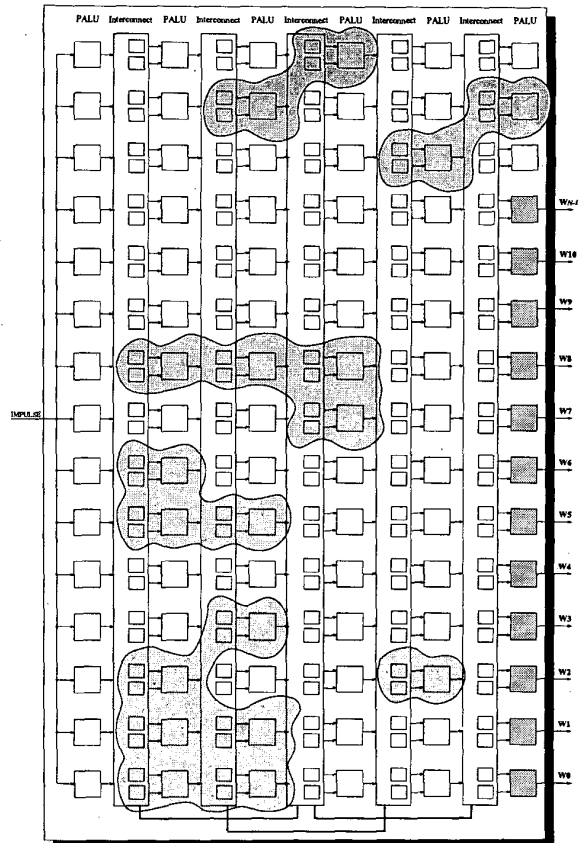


Figure 5: Fault coverage of PLA with 20% damage.

a distributed, highly parallel architecture providing component redundancy, desirable in fault tolerant applications. A 24-tap practical low-pass filter was autonomously implemented across 3 PLA cores, using a stochastic control algorithm to demonstrate configuration of the architecture. The robustness of the PLA was then examined by damaging 20% of the PALUs within the array and re-configuring the PLA for the same filter example. Results show that the PLA provided sufficient flexibility for the stochastic algorithm to implement the low-pass filter with only a 0.3% reduction in functionality. PALU utilisation and re-use remained comparable to that required by the undamaged PLA. Synthesis of the core demonstrated that fast data throughput can be achieved at typical SoC bus speeds of up to 100 MHz, and with a latency of 7 clock cycles, irrespective of filter length and the number of cores instantiated. Smaller, more modern silicon technologies would be expected to provide further increases in the operational speed of the embedded programmable core.

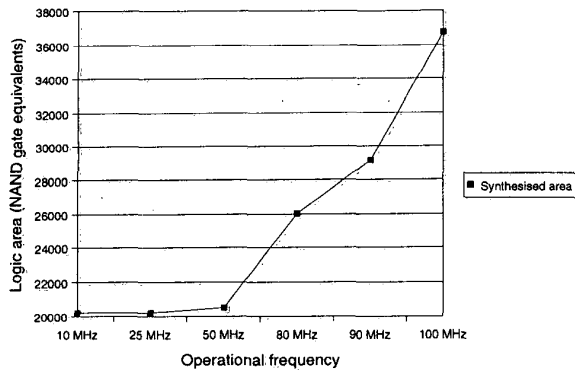


Figure 7: Logic area of PLA core as a result of synthesis for increasing operational speeds.

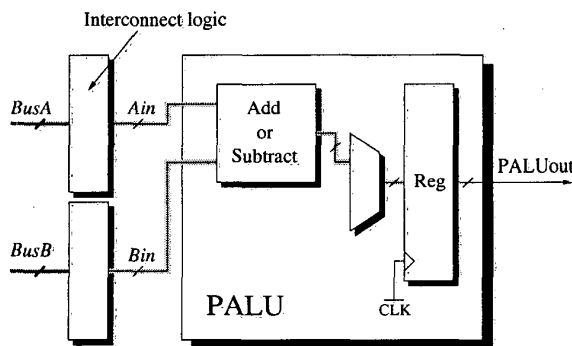


Figure 8: Critical delay path through PLA architecture.

Acknowledgements

This work has been kindly sponsored by Applied Materials.

References

- [1] Xilinx Data Book 2000, www.xilinx.com
- [2] R. A. Hawley, B. C. Wong, T.-J. Lin, J. Laskowski, and H. Samueli, Design Techniques for Silicon Compiler Implementations of High-Speed FIR Digital Filters, *IEEE Journal of Solid-State Circuits*, vol. 31, no. 5, May 1996, pp. 656-667.
- [3] H. Samueli, An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients, *IEEE Trans. Circuits Syst.*, vol. 36, no. 7, pp. 1044-1047, July 1989.
- [4] D. R. Bull, D. H. Horrocks, Primitive operator digital filters, *IEE Proc. -G*, pp.401-412, 1991.
- [5] G. Wacey, D. R. Bull, Architectural synthesis of digital filters for ASIC implementation, Digital and Analogue Filter and Filtering Systems, IEE Colloquium on, pp. 6/1-6/5, 1991.
- [6] T. Arslan, H. I. Eskikurt, and D. H. Horrocks, Configurable Structures for a Primitive Operator Digital Filter FPGA, *IEEE Workshop on Signal Processing Systems. SIPS 97 - Design and Implementation*, 1997, pp. 532-540.
- [7] J. B. Evans, An efficient FIR filter architecture, in Proc. May 1993 Int. Symposium. Acoust., Speech, Signal Processing, vol. 1, pp. 627-630.
- [8] I. Enis Ungan, and M. Askar, A gate array chip for high frequency DSP applications, in Proc. 1994 Int. Conf. 7th Mediterranean Electrotechnical, pp. 549-552.
- [9] S. Nooshabadi, J. A. Montiel-Nelson, and G. S. Visweswarain, Micropipeline architecture for multiplier-less FIR filters, in Proc. Jan 1997 Int. Conf. 10th Conference on VLSI Design
- [10] SungHun Yoon, and Myung H. Sunwoo, An efficient multiplierless FIR filter chip with variable-length taps, Signal Processing Systems, 1997. SIPS 97 - Design and Implementation., 1997 IEEE Workshop on, pp. 412-420
- [11] E. C. Ifeachor, and B. W. Jervis Digital Signal Processing, Addison-Wesley, 1993.
- [12] S. Levi, and A. K. Agrawala, Fault tolerant system design, McGraw-Hill 1994.
- [13] R. Karri, K. Hogstedt, and A. Orailoglu, Rapid prototyping of fault tolerant VLSI systems, High-Level Synthesis, 1994., Proceedings of the 7th International Symposium on, pp. 126-131.
- [14] J. McClellan, T. W. Parks, and L. R. Rabiner, A Computer Program for Designing Optimum FIR Linear Phase Digital Filters, *IEEE Transactions on Audio and Electroacoustics*, vol. AU-21, no. 6, pp. 506-526, Dec 1973.
- [15] D. E. Goldberg, Genetic algorithms in search optimization and machine learning, Addison-Wesley, Reading, Mass., 1989.
- [16] T. Bäck, D. B. Fogel, and T. Michalewicz, Evolutionary Computation I, INSTITUTE OF PHYSICS PUBLISHING, Bristol and Philadelphia., 2000.