Evolvable Hardware Architecture Using Genetic Algorithm for Distributed Arithmetic FIR Filter

K. Krishnaveni, C. Ranjith and S.P. Joy Vasantha Rani

Abstract The aim of the paper is to design evolvable hardware (EHW) architecture for Finite Impulse Response Filter using Genetic Algorithm. Evolvable hardware refers to hardware that can change its behaviour (parameters such as coefficients) according to the changes in its environment. To update the filter coefficients adaptively, genetic algorithm was used. The proposed filter architecture was implemented with Xilinx Spartan 6 FPGA (XC6SLX45-CSG324) Trainer Kit. Hardware design was synthesized using the EDK (Embedded Development Kit) platform and the genetic algorithm was implemented in SDK (Software Development Kit) of Xilinx Platform Studio tool (XPS) 14.6.

Keywords Evolvable hardware · Genetic algorithm · FIR filter · Distributed arithmetic · FPGA · VLSI

1 Introduction

The evolvable hardware is a method of designing circuits based on the reconfigurable structures. Evolvable hardware uses the evolutionary algorithms (EA) to design the circuits without manual engineering [1]. Evolvable hardware can change its architecture and behaviour dynamically and autonomously according to its changes in environment [2]. The evolvable hardware design deals with the design of circuit based on the evolutionary techniques. The circuit design using this techniques use the mathematical model either of reconfigurable structure or of system with variable parameters. These techniques are used to find new and innovative solution to the circuits. Then the resulting circuit can be implemented as real hardware [3].

K. Krishnaveni (✉) · C. Ranjith · S.P. Joy Vasantha Rani
Electronics Engineering Department, MIT Campus, Anna University, Chennai, India
e-mail: krishnaveni.k.2012@gmail.com

S.S. Dash et al. (eds.), *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, Advances in Intelligent Systems and Computing 517,
DOI 10.1007/978-981-10-3174-8_26

The search for the suitable configuration of the evolvable structures is entirely guaranteed, by using the EA [4]. In this paper, genetic algorithm (GA) is used as the evolutionary algorithm and the programmable devices are preferably field programmable gate array (FPGA) [5]. The configuration bits for FPGA are the chromosomes of the GA. The evolvable finite impulse response (FIR) filter acts as the backbone of adaptive noise cancellation.

Genetic Algorithm is the adaptive algorithm used to update the FIR filter coefficients. Thus implementing evolvable FIR filters using GA for noise cancellation and thereby extracting the original signal. The FIR filter was designed using distributed arithmetic (DA) method using VHDL [6]. For this filter design the optimal coefficients are generated using GA.

In this paper, the GA operations are performed on the soft core Micro Blaze processor of the Spartan 6 FPGA for finding the fittest chromosome by configuring the FPGA to design an optimized system. The Micro Blaze is a 32 bit soft core processor provided as an intellectual property (IP) core by the Xilinx vendor [7].

The rest of the paper is organized as follows. Section 2 discusses the proposed system architecture of the evolvable hardware architecture of FIR filter. Section 3 describes the implementation of the GA for EHW of FIR filter. Results are discussed in the Sect. 4. Then the conclusions are given in Sect. 5.

2 Proposed System Architecture

The block diagram of the evolvable system is shown in Fig. 1. The FIR filter implemented in VHDL is added as an IP core and is integrated with GA written in C which is operated on Micro Blaze processor of FPGA. The FIR filter is designed using DA method. The evolvable system is used for changing the behaviour of the filter. The GA finds the fittest configuration bits (coefficients) for the FIR filter [6].

This GA program is imported into Micro Blaze soft core processor and the generations with their fitness values are displayed on the console window of the software development kit(SDK) using the UART peripheral [5].

2.1 FIR Filter

Digital filter consists of interconnection of filter taps connected in certain topology which operates on discrete-time signals. Each tap holds a filter co-efficient. FIR filter output is computed as a weighted sum (finite) of the past, present and perhaps future values of the filter input. FIR filter is chosen for this work because it has more stability and reliability. It can be used for the study of the effects of evolution on adaptability.

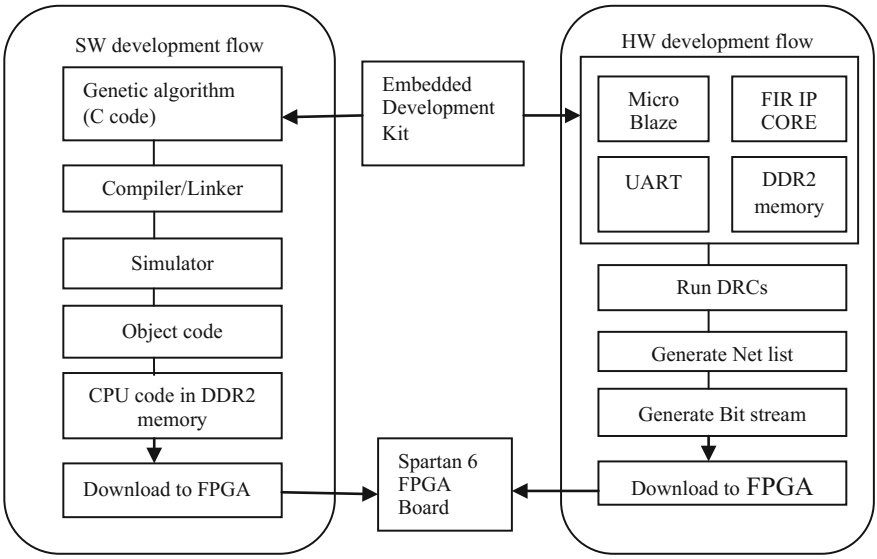

Fig. 1 Block diagram of evolvable hardware architecture. *DRC* Design Rule Check; *SW* Software; *HW* Hardware

FIR filter is described by equation,

$$y = \sum_{k=0}^{n-1} h_x x_{n-k}, \tag{1}$$

where x is the input signal, y is filter output, k is the number of co-efficients of the filter, h is filter coefficient, n is number of taps.

The transpose form of FIR filter topology is shown in Fig. 2. This filter structure is designed using DA method. Distributed arithmetic is the extension of multiply and accumulate unit (MAC). It is the efficient method for calculating the inner product or sum of products and accumulates to the filter output [8].

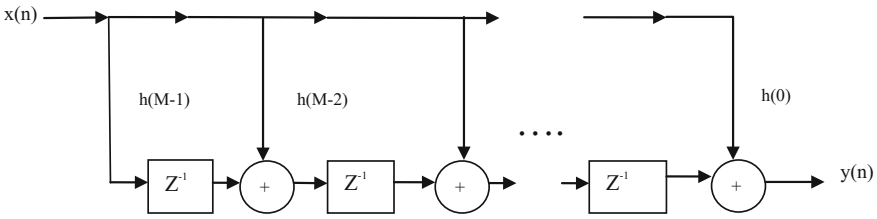

Fig. 2 Transpose form FIR Filter

The x_k value in the Eq. (1) be a N-bits scaled two's complement number

$$|x_k| < 1$$
$$x_k : \{b_{k0}, b_{k1}, b_{k2} \ldots, b_{k(N-1)}\},$$

where b_{k0} is the *sign bit*

We can express x_k as

$$x_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} \tag{2}$$

by substituting x_k in Eq (1),

Therefore,

$$y = -\sum_{k=1}^{K} (b_{k0} \cdot h_k) + \sum_{k=1}^{K} \sum_{n=1}^{N-1} (h_k \cdot b_{kn}) 2^{-n} \tag{3}$$

$$y = -\sum_{k=1}^{K} ((b_{k0}) \cdot h_k) + \sum_{n=1}^{N-1} [h_k \cdot b_{1n} + h_k \cdot b_{2n} + \cdots + h_k \cdot b_{2n}] 2^{-n} \tag{4}$$

Equation (4) is the final equation of the DA.

$$\sum_{k=1}^{K} h_k b_{kn} \tag{5}$$

Equation (5), can be pre-calculated for all possible values of b_{1n}, b_{2n} … b_{Kn}. These values are stored using a look-up table of 2^K words addressed by *K-bits.*

The Distributed arithmetic FIR filter structure is shown in Fig. 3. Carry save accumulation is used for shift accumulation process [9].

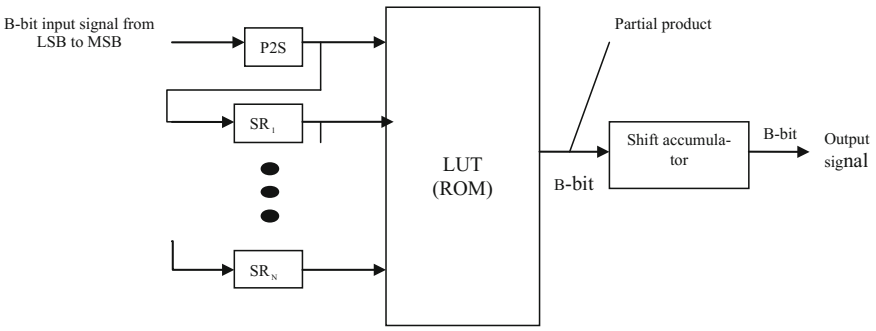

Fig. 3 Distributed arithmetic FIR filter structure. *P2S* parallel to serial converter, *SR* shift register, *LUT* look-up table

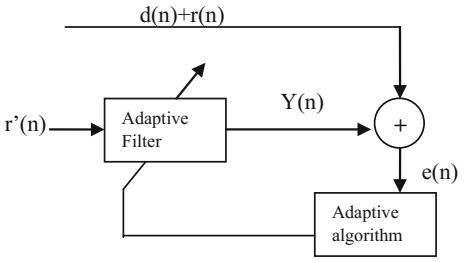

Fig. 4 Adaptive noise cancellation. Where d(n) + r (n) = noisy input; r (n) = noise; d(n) = original signal; $r'(n)$ = reference noise; $y(n)$ = filter output; e (n) = error signal

Adaptive Filter. Adaptive filters are self-designing using recursive algorithm and used where knowledge of environment is not available. Adaptive filters are used for noise cancellation application. Adaptive noise cancellation, is used to filter out an interference component by identifying a linear model between measurable noise source and the corresponding immeasurable. Figure 4 shows the adaptive noise cancellation system. The GA used as the adaptive algorithm [10].

2.2 Genetic Algorithm

Genetic algorithm (GA) is a particular class of EA, categorized as global search heuristics. GAs is heuristic search algorithms based on the mechanism of natural selection and genetics. The general flow of GA is shown in Fig. 5. Each individual in the population is called a chromosome (or individual), representing a solution to the problem. Chromosome is a string of symbols either it can be a binary or real-valued bit string [11].

The chromosomes evolve through successive iterations called generations. During each generation, the chromosomes are evaluated, using some measures of fitness. The next generation is created by, new chromosomes, called offspring, are formed by crossover, mutation operators [3]. Then new generation is formed by using selection operation, based on the fitness values. Fittest chromosomes have the highest probabilities of being selected. GAs converges to the best chromosome which represents the optimal solution to the problem after several generations. GA requires the following, a genetic representation of the solution and a fitness function

Fig. 5 Pseudo code for GA

```
Choose initial population
Evaluate each individual's fitness
Repeat
Select best-ranking individuals to reproduce
Mate the pairs at random
Apply crossover operator
Apply mutation operator
Determine the population's average fitness until termination condition
(until one individual has the optimal desired fitness or number of gen-
erations has passed)
```

to evaluate the solution. The genetic operators are selection, crossover, Mutation are used for creating new generation of solutions according to the fitness function.

Selection. Selection method is used for finding two (or more) individuals for crossover. The selection is the degree to which the better individuals are favoured: the higher the selection pressure, the more the better individuals are favoured. The selection strategies are Roulette wheel selection, Tournament selection, Truncation selection, Ranking and Scaling, Sharing selection.

Roulette wheel selection determines the selection probability or survival probability for each chromosome proportional to the fitness value. The Selection probability P_{select} of each chromosome is calculated by:

$$p(x)_{\text{select}} = \frac{J(x)}{\sum J}, \tag{6}$$

where $J(X)$ is the fitness score of the chromosome $X1$, $X2$, $X3$ … Xn and J is the sum of all fitness scores in the population.

Crossover. Crossover is an artificial mating in which chromosomes from two individuals are combined to form chromosome for next generation. Crossover is performed by splicing two chromosomes from two different individuals (solutions) at a crossover point and swapping the corresponding spliced parts. This process will provide a better solution represented by the new chromosome.

Crossover types are single point crossover, two point crossover, and uniform crossover. Single point crossover is shown in Table 1. In single point crossover on both parents' organism strings is selected. All bits after that point in either organism string is swapped between the two parent organisms. Crossover probability decides how often crossover will be performed. The crossover probability Pc has to be always high which will guarantee rapid search for the solution.

Mutation. Mutation is a random adjustment in the genetic composition. The mutation operator changes the current value of a gene to a different one. For bit string chromosome this change amounts to flipping a 0 bit to a1 or vice versa. Table 2 shows an example for random mutation. Mutation probability refers how often parts of chromosome will be mutated. Mutation generally prevents the GA from falling into local extremes.

Table 1 Single-point crossover

Parent A	11001|001
Parent B	11011|101
Offspring A	11001|101
Offspring B	11011|001

Table 2 Random Mutation

Chromosome after crossover	11011001
Chromosome after mutation	11**1**110**1**1

3 Implementation

This section explains about the implementation of evolvable system design. A hardware platform was developed using Xilinx platform Studio tool with the available library components, such as Micro Blaze processor [12], UART (Universal Asynchronous Receiver Transmitter), DDR2 (Double data rate synchronous dynamic random access memory). The UART and DDR2 components are interface with Micro Blaze processor using a common clock module. This hardware design is done using embedded development kit (EDK) platform [9].

Now, the FIR filter structure using DA method implemented in VHDL is imported as an IP core in the above hardware platform and is interfaced with the Micro Blaze processor. The GA is implemented in the SDK platform. Then an embedded system is formed by integrating the SDK platform with the hardware platform implemented in EDK. The GA is coded using C language. The specifications of GA:

Selection method: Roulette wheel selection in which each individual gets a slice of the wheel, but more fit ones get larger slices than less fit ones.
Crossover: Single point crossover.
Probability of crossover: 70%
Mutation: Random mutation
Probability of Mutation: 0.5%
Population size: 4(number of coefficients)
No. of iteration: 1000(maximum)

The FIR filter specifications: The cut-off frequency of the filter was set to 0.5 kHz and the filter order 3 with 4 filter coefficients. The reason for the number of coefficients was chosen small is mainly reduces the computation of the code.

4 Results and Discussions

The performance results of the proposed evolvable hardware architecture were compared with Matlab results. The hardware platform was designed using XPS tool and the GA was implemented using the SDK platform of the Xilinx Spartan 6 FPGA. Then this design was fused to the FPGA Kit and the terminal of the SDK platform was connected to PC through serial communication port UART.

The Fig. 6 shows the implementation of GA configuration in SDK platform in FPGA Spartan 6 Trainer Kit. The final set of normalized coefficients was displayed on the console window of the SDK platform through the use of UART peripheral.

The Fig. 7 shows the original signal with frequency of 100 kHz and the sampling frequency is set to 300 kHz. 100 number of samples was taken for simulation (Iterations). The original signal is corrupted by white noise with variance $\sigma^2 = 0.5$. The sinusoidal signal with additive white noise is illustrated in Fig. 8.

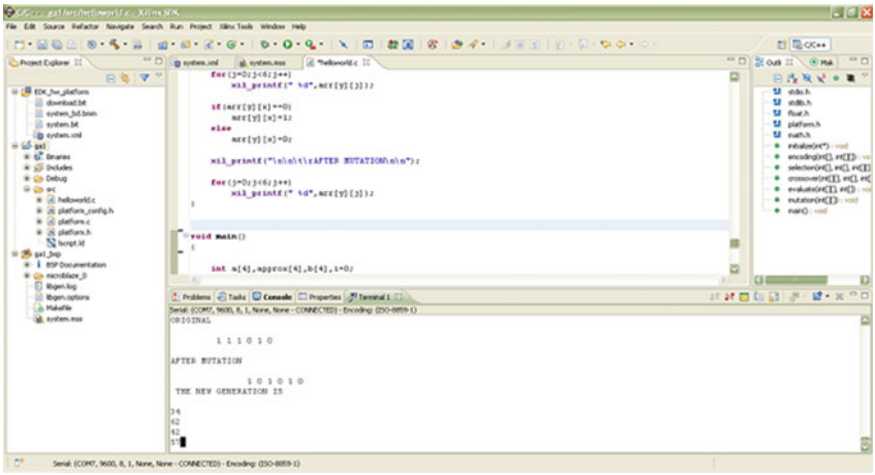

Fig. 6 SDK Platform for GA configuration

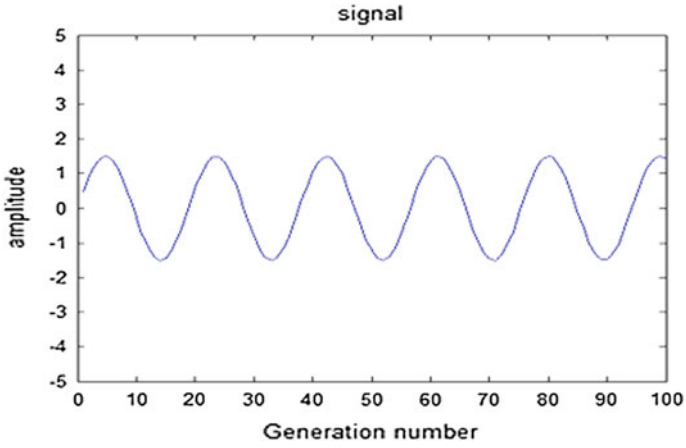

Fig. 7 Original signal

The aim of the simulation was the mean square error minimisation. Figure 9 shows the enhanced signal for the crossover probability of 0.7 and mutation probability of 0.005 of GA [13]. The better results are achieved by a crossover probability of range between 0.65 and 0.9. So crossover probability of 0.7 was considered. Similarly for mutation rate the range was 0.5–1%. So Mutation probability of 0.5% was considered.

The error was reduced significantly and reaches the global optimum value after 80th generation. The final set coefficients from GA were (0.0075, 0.0021, 0.0021

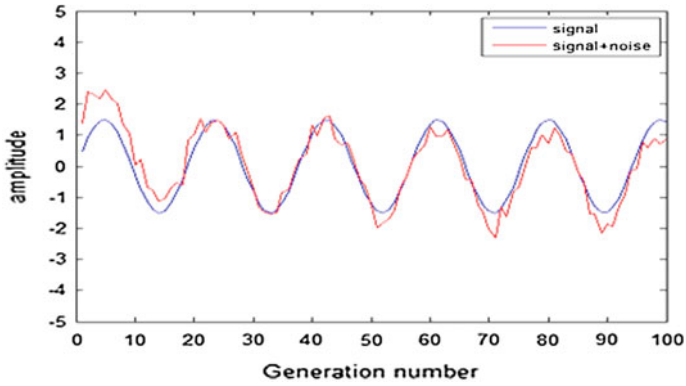

Fig. 8 Original signal and original signal + noise

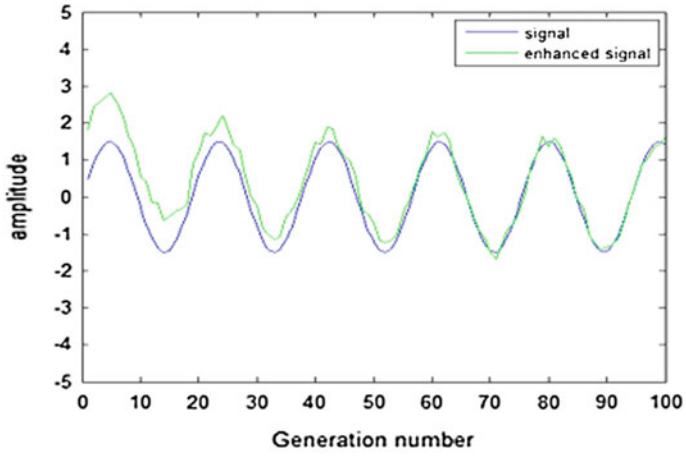

Fig. 9 Signal and enhanced signal by genetic algorithm

and 0.0071). It can be observed that the filter output gradually gets closer to the original signal after 80th generation.

5 Conclusion

The use of adaptive algorithm for noise cancellation has been presented in this paper by using the method of updating the coefficients of FIR Filter (Evolvable Filter) using GA. The design of the FIR IP Core in VHDL was done and the coefficients were updated using GA (implemented in C) in FPGA. This paper also

describes the importance of crossover probability, mutation probability of GA program. Thus, it may be concluded that GAs are feasible and better practical approach for adaptive filtering.

References

1. L. Sekanina.: Evolvable Hardware Tutorial. GECCO 2007, New York.
2. Jim Torresen.: An Evolvable Hardware Tutorial. Department of Informatics. University of Oslo. (2004).
3. Aifeng Ren, Wei Zhao, Shuo Tang, Xin Tong, Ming Luo.: Implementation of Evolvable Hardware based Improved Genetic Algorithm. IEEE conference (2011) 2112–2115.
4. Ruben Salvador, Lukas Sekanina et. al.: Self-Reconfigurable Evolvable Hardware system for Adaptive Image Processing. IEEE transactions on computers vol. 62, No. 8, (Aug. 2013) 1481–1493.
5. Ranjith, C., Joy Vasantha Rani, S.P., Priyadharsheni, B., Medhuna Suresh and Madhusudhanan, M.: Optimizing GA operators for System Evolution Of Evolvable Embedded Hardware On Virtex 6 FPGA. ARPN Journal of Engineering and Applied Sciences vol. 10, No. 11, (June 2015) 4908–4914.
6. Zdenek Vasicek, Lukas Sekanina.: An evolvable hardware system in Xilinx Virtex II Pro FPGA. International Journal on Innovative Computing and Applications, Vol. 1, No. 1, (2009) 63–73.
7. Rod Jesman, Fernando Martinez Vallina and Jafar Saniie.: Micro Blaze Tutorial Creating a Simple Embedded System and Adding Custom Peripherals Using Xilinx EDK Software Tools. Embedded Computing and Signal Processing Laboratory, Illinois Institute of Technology.
8. Syed Shahzad Shah, Saquib Yaqub and Faisal Suleman.: Distributed arithmetic for the Design of High Speed FIR Filter using FPGAs. UET, Taxila, (1999).
9. Stanley A. White.: Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review. IEEE ASSP Magazine, (July 1989).
10. Uma Raja ram, Raja Paul Perinbam, Bharghava.: EHW Architecture for Design of FIR Filters for Adaptive Noise cancellation. IJCSNS, vol. 9, No. 1, (Jan. 2009) 41–48.
11. Ajoy Kumar Dey.: A Method of Genetic Algorithm (GA) for FIR Filter Construction. Vol. 1, (Dec. 2010) 87–90.
12. Yang Zhang, Stephen L. Smith, and Andy M. Tyrell.: Digital Circuit Design using Intrinsic Evolvable Hardware. Proceedings of NASA/DoD Conference on Evolution Hardware (EH'04), (2004).
13. Pradeep kaur, Simatpreet Kaur.: Optimization of FIR Filters using Genetic Algorithm. IJETICS vol. 1, No. 3 (2012) 228–232.