

An Efficient Fault-Tolerant VLSI Architecture Using Parallel Evolvable Hardware Technology

Evangelos F. Stefatos and Tughrul Arslan

School of Engineering & Electronics
The University of Edinburgh, King's Buildings, Mayfield Rd, Edinburgh
EH9 3JL

Evangelos.Stefatos@ee.ed.ac.uk

Tughrul.Arslan@ee.ed.ac.uk

Abstract

This paper proposes a novel, fault-tolerant, VLSI architecture, which utilizes an Evolvable Hardware (EHW) framework using a parallel Evolutionary Algorithm (EA). The architecture consists of two layers. The first layer considers the application in hand, whereas the second is used as a controller that monitors the performance of the first layer and reconfigures when appropriate, its computational elements. The demonstration of this architecture is done through a practical example of a Global-Positioning-System (GPS) Attitude Determination System. Firstly the structure and functionality of both layers is described. Subsequently, the paper provides results that demonstrate both the reliability and performance of the system, while various quantities of faults are simultaneously injected in both layers. According to these results, the first layer is capable to cope with higher amount of faults (worst-case scenario 35-40%) than the second (control) layer, which copes with faults that capture in worst-case scenario the 30% of its resources. Finally, an additional mechanism to this architecture is proposed that needs further investigation and promises further enhancing of the systems reliability.

1. Introduction

The concept of fault tolerance has a very important role in safety-critical applications such as aerospace, military missions and satellite communications, where human intervention in cases of emergency, is very difficult. Therefore, research over the last few years has focused on the need of exploring ageless, robust system designs for such applications. These systems should be able to operate for many decades in harsh environmental conditions, such as these prevailing in space. Major factors that mainly affect electronic devices in such environments are the cosmic radiation

and the variation in temperature conditions. Moreover, this problem is becoming more intensive as the applied technology shrinks and transistors become more sensitive due to issues like the *hot electron effect*.

Designing systems that are totally immune to faults is very difficult and maybe impossible. However, the terminology of fault tolerant systems expresses the ability to maintain their functionality in the presence of faults. Several techniques are found in literature presenting different approaches for designing robust electronic systems. Most of these approaches employ techniques such as *check-pointing*, *concurrent error detection* and *redundancy*, which are not practical because they reduce the operational speed and increase the area and cost of the chip.

Alternative approaches use evolutionary techniques [1], [2], [4] in order to design fault tolerant circuits. These techniques are motivated by the expanding field of Evolvable Hardware (EHW) that uses a Genetic Algorithm (GA), in order to monitor the performance of the system and reconfigure its electronic resources properly according to the dynamic environmental conditions. GAs are powerful, stochastic search methods based on principles of natural selection and population genetics [7] that are used to optimize problems in the field of reconfigurable, VLSI design.

The main target of this paper is to demonstrate a VLSI architecture that is capable to maintain its functionality within valid, tolerable boundaries despite the presence of numerous faults, which may cause malfunction to several hardware resources. The proposed platform uses an evolutionary process in order to reconfigure itself in real-time according to the faults that are detected at a given instance. This dynamic adaptation is based on the application and mapping of biological methods in hardware design. Consequently, the concept of EHW is introduced, which for the needs of this architecture is based on a Parallel Genetic Algorithm PGA [5], [6].

Our system consists of two distinct layers. The first layer performs the intended functionality and is called the Computational Layer (CPL). Its functionality can be differentiated based on the targeted application. For

the purpose of this paper the feasibility of the fault tolerant platform is demonstrated through a practical example of a GPS Attitude Determination System [8], [9]. The second layer controls the execution of the CPL and therefore is termed the Control Layer (CTL).

2. System Description

Due to the complexity of some systems especially those used in circumstances where the time constraints need to be in real-time [3], the execution of a GA in dynamic reconfiguration needs to be fast. PGAs have been shown superiority in such applications [8], [9]. This efficiency is achieved in terms of processing time because they search in parallel different ranges of potential solutions. Thus, complicated, multi-objective problems take advantage of this parallelism in order to cope with the bigger population that their complexity introduces.

In this architecture a fine-grained PGA is employed. This specific PGA requires a large number of Processing Elements (PEs) because the population is partitioned into a large number of subpopulations that ideally must consist of one individual (fine-grained PGA)¹. Furthermore, each PE has its own chromosome that provides a different optimized solution to the problem. Hence, each one of the two layers consists of N number of PEs. The exact number can differ according to the trade-off between the degree of robustness and the area occupation that a specific application requires. For the purpose of this paper both CPL and CTL layers consists of 64 PEs each. Their structure is almost the same, however their functionality is different. The taxonomy of the PEs in both layers is arranged in an 8x8 array that forms the fine-grained PGA that act on each member of the population in parallel. Consequently each member of the population performs crossover with its immediate (north, east, south, west) neighbours. Despite the fact that the PEs in each layer are different, there is still information that is exchanged between the two layers. Figure 1 illustrates the interface between the PEs of the two layers.

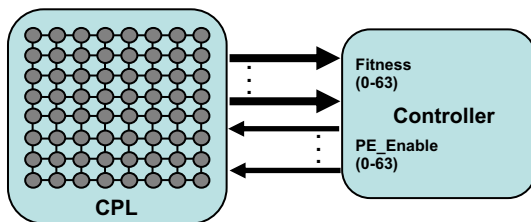


Figure 1: Interface between the two layers

¹ The "grain size in parallelism refers to the ratio of the time spent in computation and the time spent in communication. When this ratio is low the processing is called fine-grained.

Each one of the controllers in the CTL takes an input, which is the fitness value (5-bit) of each computational PE in the CPL. Hence, each controller in the CTL is able to monitor the fitness score of the PEs in the CPL. In addition, each controller has 64 output signals (PE_Enable – 1-bit), one for each PE in the CPL that drive the input signals of the PEs in the CPL and define which PEs should operate. The five bits that represent the fitness value of the GPS PEs are fractional and hence the minimum and maximum values that can be represented are 0 and 0.96875 respectively. In the next sections of this paper, it will be assumed that a malfunction occurred in a PE can be simulated by asserting its fitness score to logic "0" or alternatively to a value that is unacceptable (very poor) and does not converge to a predefined threshold value.

3. Computational Layer – CPL

As it was briefly mentioned in the introduction of this paper, the role of the CPL is to resolve the attitude determination of a vehicle, using Global Positioning Systems. This method is widely applied on the design of navigation systems that are employed in spacecrafts, aircrafts and other space applications. Further information concerning this method can be found in [8], [9]. As it was already mentioned, the 64 PEs that compose the CPL run a fine-grained PGA, which performs a parallel search to identify the following parameters:

- a. **Azimuth Angle ϕ** , of the baseline
- b. **Elevation Angle β** , of the baseline
- c. **Length b** , of the baseline

The first two parameters are used to determine the attitude of an object. The length of the baseline is defined as the distance between the two receiver antennas that are attached on the vehicle. The chromosome of each PE is illustrated in Table 1.

Table 1: Definition of chromosome in CPL

Parameters	ϕ	β	b
Bits	0-13	14-23	24-31

The fitness function that guides the searching procedure of the GA is defined in terms of baseline length and angles with both horizontal and vertical planes. The bigger the fitness value the better solution the GA resolves for the attitude of the vehicle. Figure 2 illustrates a flow diagram for the GA of each PE.

Moreover, it can be seen in [8], [9] that the fitness function is mainly expressed by trigonometric functions that are quite demanding in terms of

computational time. Therefore, in order to decrease the intensive computational processing that the Evaluation procedure of the GA, puts in each PE, efficient hardware algorithms have been employed, which are known as Coordinate Rotation Digital Computer (CORDIC) algorithms [10]. According to these, the trigonometric functions are calculated with quite high precision, based on iterative methods that utilize only “shifting” and “adding” operations. A significant advantage that this method presents in respect to the conventional techniques is the small amount of memory (ROM) that the algorithm needs, despite the nature and the demands of the applied application.

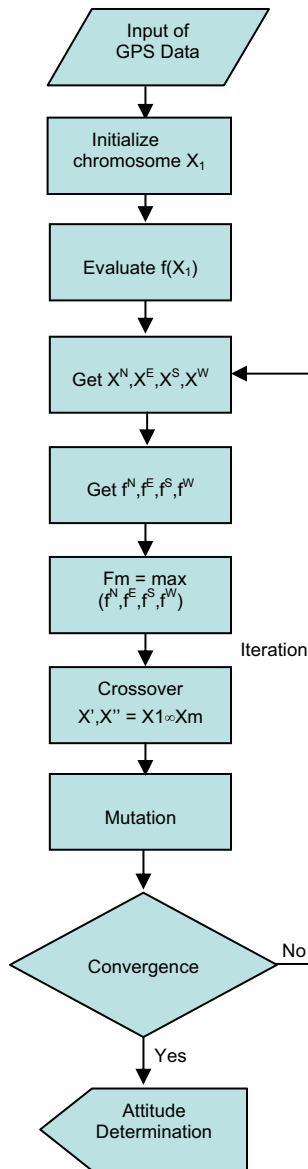


Figure 2: Flow Diagram of the GA in CPL

As it was mentioned before, the range step of the GA is fractional, which in turn implies the need of a very large look-up-table to store all possible angles. With CORDIC algorithms this waste of storage resources is eliminated because the trigonometric functions of the different angles are calculated rather than be stored in a huge amount of memory. Apart from the reduction in terms of the area, this method significantly reduces the chance of a fault occurring because memories seem to be more prone to failures.

4. Control Layer – CTL

The structure of this layer is identical to that of the CPL one. Hence it is organized as an 8x8 array structure. The 64 PEs are also termed *Controllers*, due to the nature of their functionality and run a fine-grained PGA. The scope of the PGA is to optimize the problem of the optimal and most efficient selection of a group of five “alive” PEs that comprise a cross in the CPL. This means that when a PE is surrounded by faulty ones in all the possible (north, east, south, west) directions, there is no point for it to operate. This is because it cannot disseminate its solution to the rest of the PEs connected to it. Therefore the PGA tries continuously to resolve 8 combinations of five “alive” PEs, which form a cross, as shown in Figure 3.

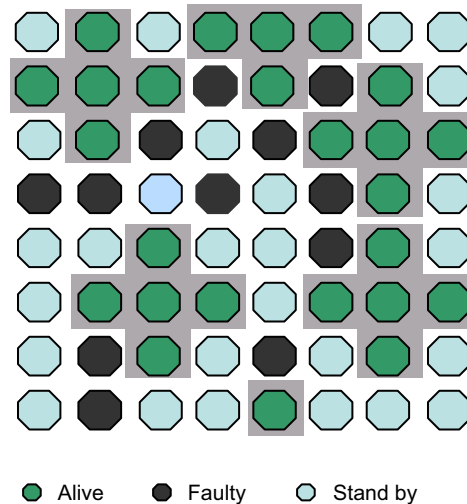


Figure 3: Potential combinations of valid solutions

Figure 3 illustrates five acceptable crosses. This information is given by one of the 64 chromosomes that compose the whole population of the CTL. The faulty PEs are depicted in a black color. It can be deduced that crosses, which contain even one faulty PE are not valid because the information that the faulty PE

transfers to its neighbours is invalid and hence the maximum search area that can be reached at an ideal case is decreased. The “alive” PEs, which are not used, will remain in *stand by* mode until the next iteration. In addition, they exchange information with the “alive” ones but it is not imperative for them to converge. As soon as they are “alive”, these can be selected in the next configuration to participate in potential crosses.

The flow diagram of the PGA in this layer consists of the same stages to the one depicted in Figure 2. However the fitness function and the definition of the 64 chromosomes that compose the overall population are different. Each chromosome contains the coordinates (x, y) of 8 PEs in the CPL that exist in the centre of the crosses selected by the chromosome. The coordinates are 8-bit long each and hence the total length of each individual is 64 bits.

The fitness function is expressed as the summation of the fitness values of 8 combination of “alive” PEs, which form a cross in the CPL.

$$\text{Fitness}_{(CTL)} = \sum_{i=0}^7 \text{fitness_i}_{(CPL)}$$

When the 8 selected crosses consist of “alive” PEs the fitness value should not be less than 38.75. This number comes out if we multiply the following parameters: 1) the number 8, which defines the crosses selected by a chromosome, 2) the minimum, acceptable fitness score (0.96875) for each PE in CPL and 3) the number 5, which defines the PEs that form each cross. Therefore, the number 38.75 is the threshold value of the fitness score that the GA must manage in order for the respective PE to converge.

Similarly for the CPL, faults can also occur in the CTL. In contrast to the CPL, the robustness of this layer is only based on the inherent parallelism of the PGA. Therefore, there is no any mechanism to evaluate the quality of the solutions that are given by the CTL in order to isolate the faulty PEs. In the presence of many faults (>40%) some of the PEs struggle to resolve an acceptable solution and the number of the iterations it takes these to converge, becomes prohibitive. However the reliability of the system is based on the fact that the majority of the PEs should converge.

5. Introducing Faults

It is a primary concern for a designer who tests a system for its ability to tolerate against faults induced into its hardware resources, to accurately specify the possible nature of faults that may occur and how these can be efficiently modeled into the system under test.

The fault-tolerant platform that is presented in this paper mainly targets *Space Applications*, where *Single-Event-Upset (SEU)* and *multiple-bit SEU* errors dominate. These errors occur when charged particles, which are usually originated from radiation belts or cosmic rays, lose energy by ionizing the medium through which they pass. As a result SEUs appear as a substantial deviation from the expected level value. In digital systems this is translated to single or multiple bit flips in memory or register elements.

Both the CPL and CTL were subjected to different levels of the above SEU errors in order to explore and demonstrate the capability of the overall architecture to resist failures.

In order for the system to meet all the constraints fault-tolerant applications required by, it is imperative for all the selected (alive) PEs in the CPL to converge. Therefore, it is obvious that the most destructive scenario for the functionality of the system is to cope with stuck-at zero faults that occur on the input/output registers of the PEs in both layers (CPL and CTL). Based on this scenario, it will be an unattainable objective for the “alive” PEs to converge because one or more bits of the register that keeps their fitness score may be stuck-at logic zero. As a conclusion, there will be a deadlock in the operation of the PGA preventing the “alive” PEs to converge.

On the contrary, when a stuck-at one fault occurs on memory cells that keep the fitness value, there are only few predefined positions in the bit-string (Most-Significant bit, MSb) that an alternation/flip in value will negatively affect the operation of the PGA. However, even this worst-case scenario, according to which a Controller erroneously believes that a PE in CPL has converged, is not destructive for the system because this fault will not be absorbed by the adjacent PEs. This deduction is shown in the flow-diagram of the PGA in Figure 2. It can be seen that after the *exchange* state at which each PE exchanges both the best chromosome and its respective fitness-score, it consequently performs another reproduction. Thus, it is impossible for an erroneous value affected by a stuck-at one fault to propagate to the whole array.

Therefore, the simulation patterns of this promising approach that still is in its infancy have been concentrated to the study of stuck-at zero faults. According to this approach each fault was individually injected into the system by pulling to logic zero the input/output registers of the faulty PEs.

The simulation of this robust platform was performed using Cadence Verilog XL simulation tool. The results that were obtained, aim at indicating the important role of the CTL when a significant number of faults occur in CPL.

6. Results Analysis

Initially we measured the average number of iterations it takes for the PGA in the CPL to converge at different failure levels (0%, 15%, 30% and 40%). Here is important to mention that the concept of iteration refers to a new generation (Figure 2). In addition, these results were obtained assuming that the CTL is out of operation. The ultimate scope of this simulation approach was to present the crucial role of the CTL, as the amount of faults gradually increases. Figure 4 depicts the results of the first simulation scenario (CTL out of operation).

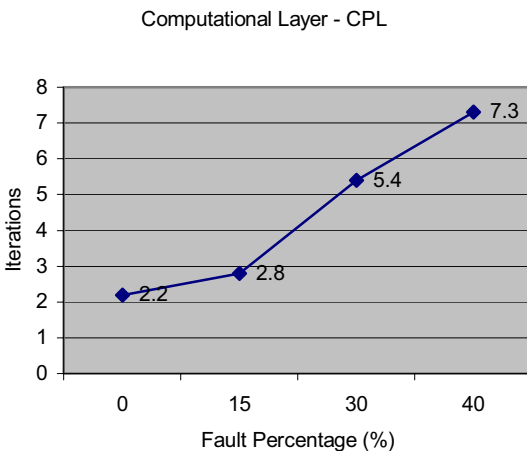


Figure 4: Mean number of iterations that PGA in CPL needs without the utilization of the CTL

The conclusion of the first simulation was that the mean number of the iterations was relative not only to the amount of faults but also to the way they emerge inside the array. Figure 5-6-7 depict the three distinctive configurations of the faults that were applied on the CPL. Based on the different configuration of faults that were simulated, it is apparent that the number of iterations increases significantly when “alive” PEs are surrounded by faulty ones. This is more obvious in the third scenario that is depicted in Figure 7, where five PEs are isolated in all the possible directions in which they can communicate with their neighbours. This collision occurs because the surrounded PEs cannot take advantage of the fine-grained parallelism, as they operate individually. The majority of the simulations showed that in these worst-case scenarios the number of iterations can increase dramatically and thus the performance of the system is inadequate for real-time applications.

Subsequently, we repeated the same simulations, incorporating this time the functionality of the CTL. During these simulations, it is assumed that the CTL is

immune of faults because our mainly scope is to present its beneficial effect over the CPL.

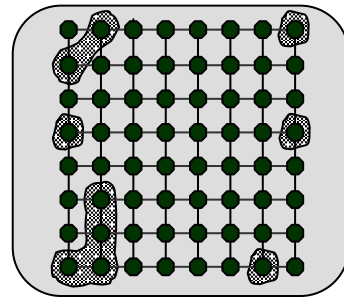


Figure 5: Percentage of faults 15%

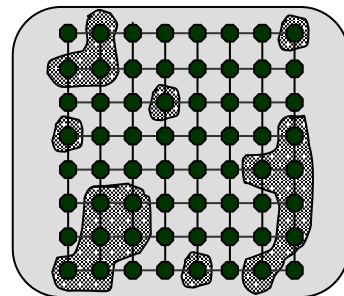


Figure 6: Percentage of faults 30%

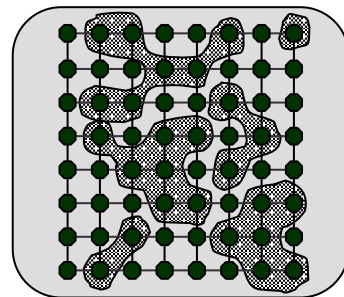


Figure 7: Percentage of faults 40%

In Figure 8 are depicted the results of the two different simulations patterns. The two graphs present respectively, the performance of the CPL when four different scenarios of faults are injected into the CPL. Comparing these two, it is apparent that the functionality of the CTL improves significantly the performance of the CPL. This enhancement becomes more evident when the number of faults increases. Apparently, this is the maximum boost that CTL can provide because we ideally assumed that there are not any faults in the CTL.

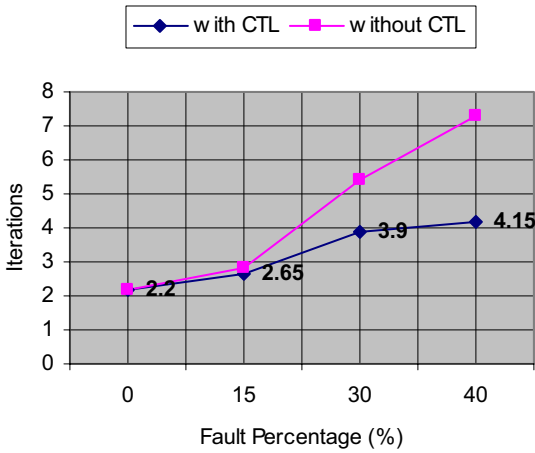


Figure 8: Mean number of iterations that PGA needs with and without CTL usage

Figure 9 illustrates the percentage reduction of the mean number of iterations for the following percentage of faults: 15, 30 and 40 respectively.

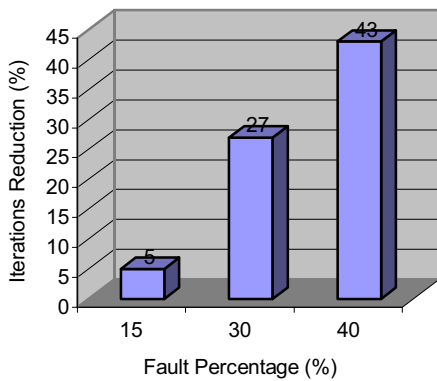


Figure 9: Percentage reduction of iterations

However, the significant reduction in the computational time that CPL needs, in order to achieve an optimization solution for the attitude determination problem, has some cost in terms of chip area. The design of a controller and a PE in the CPL has been synthesized with Synopsys Design Compiler, using UMC 0.18 micron standard cell technology library. The synthesis results indicate that the area overhead associated with the CTL is 0.387 mm² for each controller. More detailed results comparing the synthesized design are given in Table 2 and 3.

Table 2: Area Report (mm²)

	Controller	GPS – PE
Number of Ports	617	296

Number of Nets	13095	6789
Number of Cells	10640	6278
Combinational Area	0.258	0.104
Non-combinational Area	0.032	0.064
Net Area	0.097	0.061
Total Cell Area	0.290	0.168
Chip Area	0.387	0.229

Table 3: Timing Report (nsec)

	Controller	GPS – PE
Clock Period	50	50
Data Arrival Time	2.79	9.09
Data Required Time	48.91	48.82
Slack Time (MET)	46.12	39.73

Moreover, it is also very important to investigate how the performance of the CTL is affected when a growing number of failures occur in the controllers. Figure 10 demonstrates that the mean number of iterations that the controllers need to converge, increases with higher rate, for fault percentage larger than 20%. Therefore, after numerous simulations it was inferred that the CTL is constrained to cope with fewer faults than the CPL does. This is reasonable because CTL does not have a self-mechanism to isolate the PEs, which are surrounded by “dead” ones. However, the simulation results proved that CTL can compensate very efficiently for percentage of faults between 0-35%.

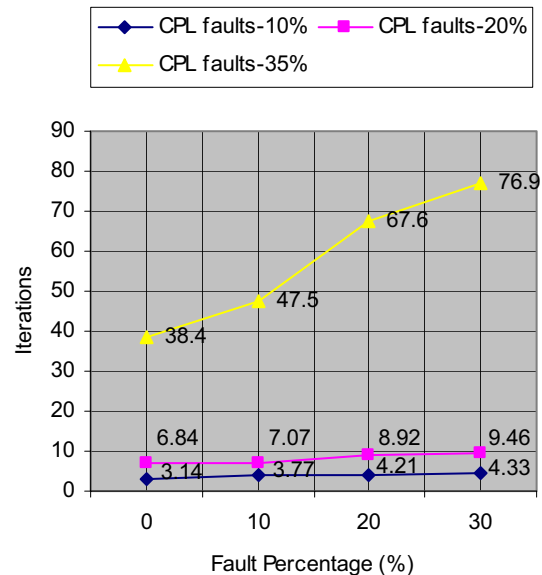


Figure 10: Mean number of iterations that CTL needs for different scenarios of faults in both layers

7. Conclusions

In this paper, an efficient fault tolerant, VLSI architecture has been presented. The analysis of its simulation results indicates that the platform is capable to maintain its functionality despite the presence of faults. According to our results, the percentage of the occurring faults can securely reach 35% and 30% for the CPL and CTL, respectively. This means that for these values, even for the worst case combinations of faults the system is able to respond within acceptable amount of time.

Future work can be done in this architecture, in order to explore alternative solutions to reduce the number of the non-operational PEs in both layers when these are surrounded by faulty ones. This is an alternative model of parallelism that introduces the concept of *migration*. According to this approach, each PE is able to communicate with every other attached in same row and column. Hence, even if a PE is isolated by its "dead" neighbours, it will have alternative paths to communicate (exchange its individual). It is believed that this approach will significantly increase the robustness of the system. However, it will introduce a significant increment in the number of the interconnections between the PEs. Moreover, some logical circuitry (multiplexers) will be needed to establish their routings. Consequently, there will be an increment in the area of the chip that has to be identified and compared with the respective increment that will be achieved in the overall robustness of the system. Finally, the computational time of the PGA has to be defined because its population will be increased, as the system will become more sophisticated.

8. References

- [1] A. Thompson. *Evolutionary Techniques for Fault Tolerance*. In Proc. UKACC Int. Conf. on Control 1996 (CONTROL '96), pages 693-698. IEE Conference Publication No. 427, 1996.
- [2] A. Thompson. *An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics*, 1st International Conference on Evolvable Systems 1996, Springer-Verlag, 1997.
- [3] B.C.H. Turton and T. Arslan. *A Parallel Genetic VLSI Architecture for Combinatorial Real-Time Applications*, 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, 12-14 September 1995, pp. 493-498.
- [4] B. Hounsell and T. Arslan, *Evolutionary Design and Adaptation of Digital Filters Within an Embedded Fault Tolerant Hardware Platform*, NASA/DoD Workshop on Evolvable Hardware, Long Beach, CA, 12-14 July 2001. pp. 127-135.
- [5] Cantu-Paz, E. 1995, *A Summary of Research on Parallel Genetic Algorithms*, IlliGAL Report No. 95007. University of Illinois at Urbana-Champaign.
- [6] Cantu-Paz, E. (1998). *A Survey of Parallel Genetic Algorithms*. *Calculateurs Paralleles, Reseaux et Systems Repartis*. Vol. 10, No. 2. pp. 141-171. Paris: Hermes.
- [7] D.E. Goldberg. *Genetic Algorithms in Search, Optimisation & Machine Learning*, Addison-Wesley, 1989.
- [8] J.Xu, T.Arslan, D.Wan and Q.Wang, *GPS Attitude Determination Using a Genetic Algorithm*, 2002 IEEE Congress on Evolutionary Computation (CEC'02), Volume 1, pp. 998-1002.
- [9] J.Xu, T.Arslan, Q.Wang and D.Wan, *An EHW architecture for real-time GPS attitude determination based on parallel genetic algorithm*, 2002 NASA/DoD Conference on Evolvable Hardware, pp. 133-141.
- [10] Raymond J. Andraka, *Building a High Performance Bit Serial Processor in FPGA*, 1996 On-Chip System Design Conference.