A Novel Hybrid Voter Using Genetic Algorithm and Performance History

Tarang Agarwal *agarwaltarang07@gmail.com*
Department of Electronics Engineering
Institute of Technology, Banaras Hindu University
Varanasi, 221005, India

Akhilesh Pathak *pathak.akhilesh@gmail.com*
Department of Electronics Engineering
Institute of Technology, Banaras Hindu University
Varanasi, 221005, India

Anand Mohan *amohan@bhu.ac.in*
Department of Electronics Engineering
Institute of Technology, Banaras Hindu University
Varanasi, 221005, India

Abstract

Triple Modular Redundancy (TMR) is generally used to increase the reliability of real time systems where three similar modules are used in parallel and the final output is arrived at using voting methods. Numerous majority voting techniques have been proposed in literature however their performances are compromised for some typical set of module output value. Here we propose a new voting scheme for analog systems retaining the advantages of previous reported schemes and reduce the disadvantages associated with them. The scheme utilizes a genetic algorithm and previous performances history of the modules to calculate the final output. The scheme has been simulated using MATLAB and the performance of the voter has been compared with that of fuzzy voter proposed by Shabgahi et al [4]. The performance of the voter proposed here is better than the existing voters.

Keywords : TMR, Soft Threshold, Genetic Algorithm, Weighted Average Voting

1. INTRODUCTION

The fault tolerant techniques are adopted to increase reliability of critical systems like aerospace, telecommunication, healthcare etc. Triple Modular Redundancy (TMR) is the most commonly adopted scheme due to its ease of implementation and considerable increase in the reliability. Here three modules work in parallel and there outputs are evaluated by a voter which gives an output based on the voting scheme implemented in the voter. In digital voting scheme, the determination of exact majority vote is straight forward; however, while in case of analog systems, finding the majority consensus amongst the outputs of redundant modules of a TMR requires determination of majority considering two closest matching analog outputs of the redundant modules. For example, it is very difficult to obtain an exact match between the outputs of replicated analog sensors of a fault tolerant data acquisition system or if the output is generated by diversely implemented software using floating point arithmetic. Therefore the design and performance of analog fault tolerant systems has been greatly focusing on development of improved voter design which can use '*hard threshold*' or '*soft threshold*' based voter implementation [9,4] The hard threshold based voter implementations determine the majority consensus among the outputs of redundant analog modules using mid-value, fixed threshold [5,6]. The mid-value selector voting algorithm [5] generates voted output by considering the mid values of the outputs of redundant modules, fixed threshold voting [6] scheme generates majority output if the absolute difference between the outputs of different pairs of redundant modules are less than a prefixed value. In fixed threshold voting, the output from different modules are in agreement if the absolute difference between them is less than a fixed value otherwise they will be in disagreement. But the use of inexact voter with a fixed threshold value encounters many problems as (i) the value of threshold is application specific so selection of the threshold is critical. (ii) Different operational modes in applications require may require different value of

threshold (iii) some acceptable results may get ignored by using fixed threshold. Alternatively, soft threshold based fuzzy voters [4] has been reported to tackle the voter's accuracy problem. These voters convert the mod value of relative differences into fuzzy membership and apply fuzzy rules to generate consensus output from the voter. The fuzzy voters perform better than hard threshold voting schemes having fixed threshold but it proves ineffective beyond moderate differences between outputs of redundant modules. Therefore a need is felt to evolve a new voting logic adopting best of all method depending on the input.

In this paper a new hybrid method of voting using genetic algorithm and history based record is proposed which overcomes the disadvantages of previous voting methods. The availability and safety performance of the proposed genetic voter has been evaluated through MATLAB simulation studies and it is shown that our proposed design gives higher number of correct results without compromising with safety even in presence of larger mod differences. Section II describes the basic concepts of genetic algorithm followed by the proposed genetic and history based voter in section III. Section IV presents the MATLAB simulation of the proposed voter along with the variation in outputs of redundant modules. The simulation results indicating the error minimization in the voted output in terms of availability and safety as compared to the fuzzy voter have been discussed in section V. Section VI contains the conclusion indicating its benefit in design of fault tolerant systems.

2. GENETIC ALGORITHM

Genetic algorithm introduced by John Holland works on principle of natural evolution [11]. Further the efficiency of Genetic algorithm has been mathematically established [2].

Solution to problems using genetic approach involves encoding and evaluation. The parameters of interest are converted into codes and combined together to form a chromosome. Further the solution is evaluated for a fitness function in an iterative manner. Figure 2.1 shows the basic flow chart of the genetic algorithm where an initial population of solutions of a given problem is generated and the value of fitness function (which has to be minimized/maximized) for each solution is calculated, the solution with better fitness function has a higher probability of survival. A simple genetic algorithm that yields good results is composed of three operations:

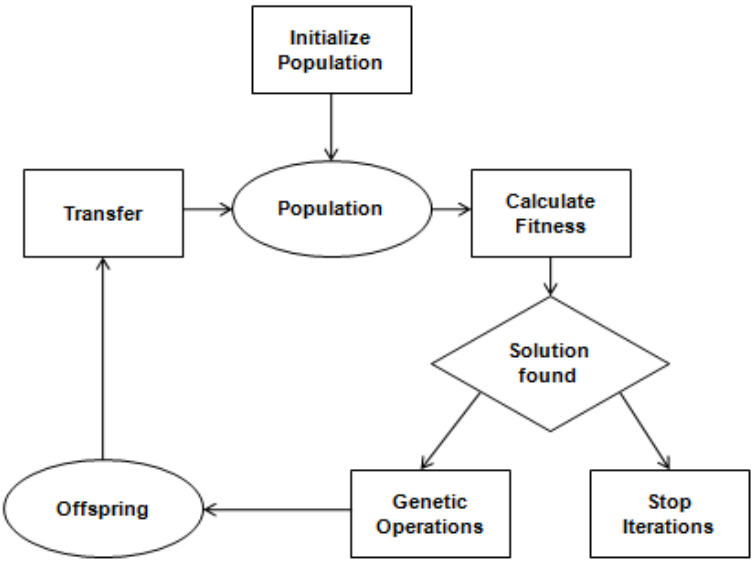

FIGURE 2.1: Flow Chart of Genetic Algorithm Process

1. Reproduction
2. Crossover

3. Mutation

Reproduction is a process in which individual solutions are selected according to their objective function values, F (fitness function). Selecting solutions according to their fitness values ensures that they have a higher probability of contributing one or more offspring in the next generation.
After reproduction, simple **crossover** may proceed in two steps. First, members of the newly reproduced solutions are selected randomly. Second, each pair of solutions undergoes swapping of information at certain position/s.

Mutation is a genetic operator that alters one or more gene value in a chromosome (solution) from its initial state. This can result in an entirely new chromosome added in next generation. With these new gene values, the genetic algorithm would be able to arrive at better solution compared to the previous cycle.

3. THE PROPOSED VOTING METHOD

The proposed voter has been implemented as a Triple Modular Redundancy (TMR); however it can be extended to N-Modular Redundancy. It has been observed during our study of different majority voting schemes reported in the literature that a common voting strategy is generally considered for all categories of module outputs across the board. It is felt that if the voter can be made intelligent to adopt voting logic depending on the modules' agreeability with each other, it can give better results. Therefore here we propose a hybrid voting scheme which adapt itself depending on the agreeability between different modules. The agreeability between two modules is defined in terms of their mod differences with each other. If the mod difference is less than a predefined threshold value (∈); the two modules are said to be in agreement. Let d_{ij} represents the mod difference between outputs of modules *i* and *j* i.e. $d_{ij} = |X_i - X_j|$

In all for a TMR, three cases can occur namely:

Case-1 is the case where all the modules are in agreement with each other i.e. the max difference d_{ij} is within threshold limit.

Cae-2 is the case where all three modules are not in agreement with each other but atleast one module-output pair is in agreement with each other so that it follows TMR principle. So in this case min difference d_{ij} should be less than threshold value.

Case-3 is the case where no module output-pair is in agreement with each other, so here principle of TMR violates.

The output **y** will be calculated using weighted average method in all three cases but the strategy to calculate weight of each module will be different for the three cases.
The procedure to calculate voter output y for (all these three cases) is shown in figure (3.1). The max value of difference d_{ij} less than the threshold value (∈) indicates that all module outputs are in agreement with each other**.** Therefore the output y is taken as mean of all three inputs.

i.e**.** $W_1 = W_2 = W_3 = 1$ and

$$y = \frac{(X_1+X_2+X_3)}{3} \quad (1)$$

If the max value of difference d_{ij} is not less than threshold value but min value of difference d_{ij} is less than threshold (**ε**)**.** It implies that atleast one pair is in agreement with each other, the weights are calculated using the fitness function defined in eq. (2) which should be minimized to get the weights for modules:

$$F = \sum_{i=1}^{3} \frac{\epsilon}{\min(d_{ij}, d_{ik})} * (y - X_i)^{\frac{(d_{ij}+d_{ik}+d_{jk})}{(d_{ij}+d_{ik})}} \; ; i \neq j \neq k \quad (2)$$

The value of output y corresponding to minimum fitness value is considered as the voter output.
The value of weight W_i in genetic algorithm is represented in binary number b and the real value of this weight is calculated by eq. (3)

$$W_i = W_{min_i} + \frac{b}{2^L - 1}(W_{max_i} - W_{min_i}) \quad (3)$$

As the values of weights are in [0-1] range so W_{min}= 0 and W_{max}= 1, where b is the binary value of weight W_i and L is no. of bits representing W_i.

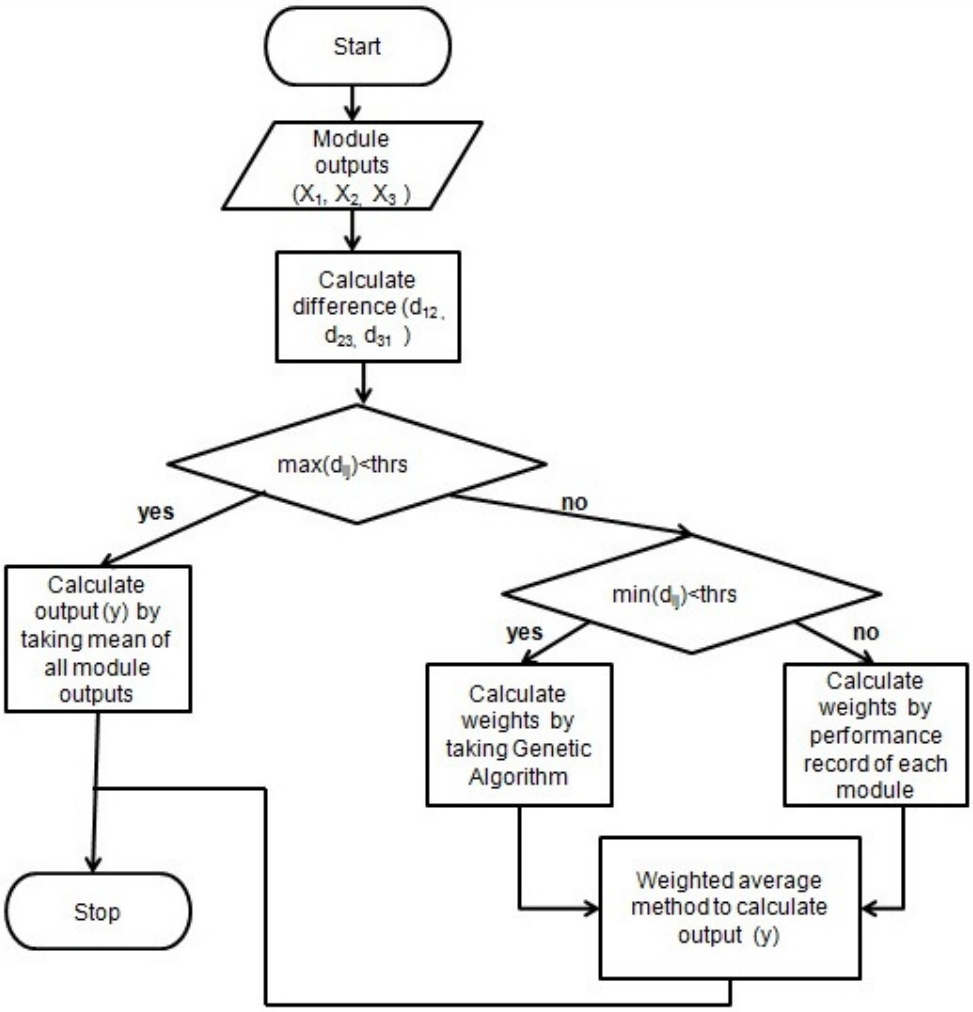

FIGURE 3.1: Flow chart of proposed voter

When there is no agreeability even between a single module pair (case 3) i.e. min (d_{ij}) > ε, the output y is calculated with the help of previous performance history of each module. A performance parameter P_i for each module is defined and its value depends upon the participation history record of each module. The algorithm below describes how the performance of each module is saved when at least one module pair is in agreement, and when there is no agreeability even between a single module pair, the performance parameter P_i is used as weight for each module.

Initialize all P_i's to 0 and j=0
For each cycle
{
If min (d_{ij}) < threshold
Calculate weights W_i's with the help of genetic algorithm;
Increment P_i by 1 corresponding to the module with max weight;
j = j+1;
When min (d_{ij}) > threshold

For each module, weight $W_i = {}^{P_i}/_{j}$ ***;***

}

The performance parameter corresponding to the module whose weight is highest will get incremented by 1 and in each cycle this procedure will be repeated so in totally disagreement condition these P_i's will be used as weights for the modules. j represents total number of cycles.

IV. MATLAB Simulation:

The proposed genetic voter has been designed in MATLAB R2009a which uses genetic toolbox. The schematic depicting the experiment conducted using the proposed genetic voter based TMR system is shown in figure (4.1) below.

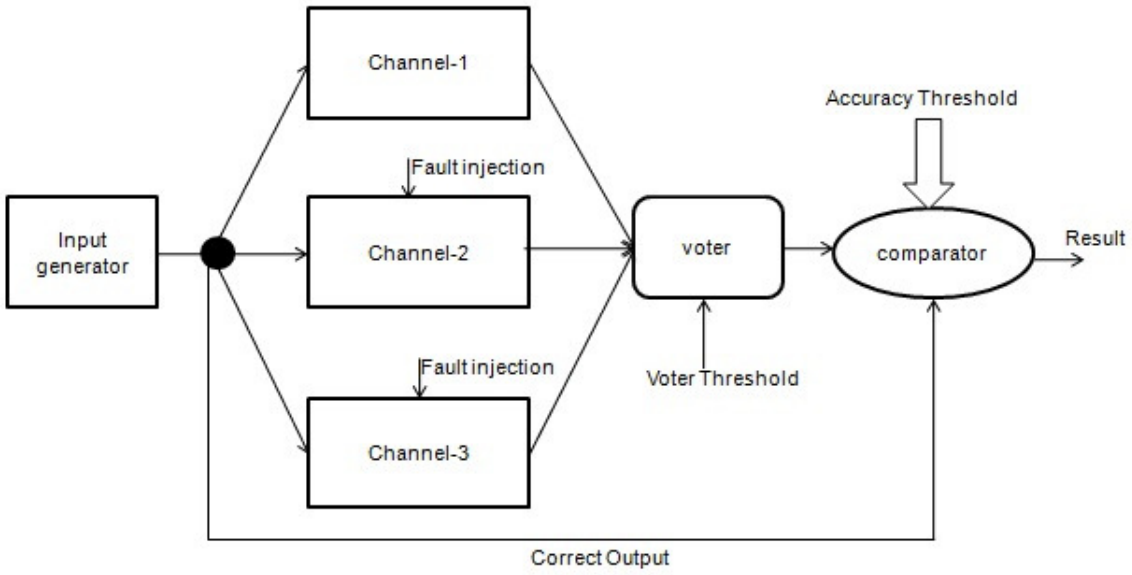

FIGURE 4.1: Simulation Model

An input data generator is used to generate data and error in two channels is injected. The input data generator produces correct output in each cycle and uniformly distributed random errors are injected in this notional correct output. Now these faulty outputs are supplied to the genetic voter and voter result is compared with the notional correct output. If the voter output is within an accuracy threshold value from the notional correct output, the voter output is considered to be correct while if the difference is larger than threshold value, output is considered to be incorrect. The method adopted is same as used by shabgahi et al.[4] where a fuzzy based voter has been proposed.

The following parameters are used to do experiment on proposed voter:

- Input data: u (t) = 100+100*sin (t) sampled at 0.1 sec.
- Accuracy Threshold value=0.5
- One module is fault free, while the error in other two has been injected randomly by uniform distribution in interval [$-e_{max}+e_{max}$].
- The simulation experiment is performed for 10^4 cycles and no. of correct results n_c, no. of incorrect results n_{ic}, and no. of benign results n_d are calculated. Benign output means that voter is not able to produce any output.
- The availability A of the system is defined as: $A - \frac{n_c}{n}$
- The safety S of the system is defined as: $S = (1 - \frac{n_{ic}}{n})$

  Where n is total no. of outputs i.e. $n = n_c + n_{ic} + n_d$

5. SIMULATION EXPERIMENTS

Experiments compare the performance of proposed genetic voter with the fuzzy voter [4] in terms of safety and availability parameter defined above. Figure 5.1(a) and 5.1(b) shows the availability and safety performance respectively of two voters when one module is fault free and the fault in other two modules has been injected randomly with uniform distribution in range [$-e_{max}$ $+e_{max}$]. The x-axis in graph shows the value of maximum amplitude e_{max}. It is observed that our proposed voter gives [0-32]% better availability and [0-5]% better safety results as compared to fuzzy voter proposed in [4]. For example when the fault injected in two modules are of magnitude [-5 +5], the following table shows the results for both the voters:

Voter	No. of correct outputs (n_c)	No. of incorrect outputs (n_{ic})	No. of benign outputs (n_d)
Genetic Voter	5515	2390	2095
Fuzzy Voter	2503	2391	5106

TABLE 1: comparison of results for max error amplitude of 5

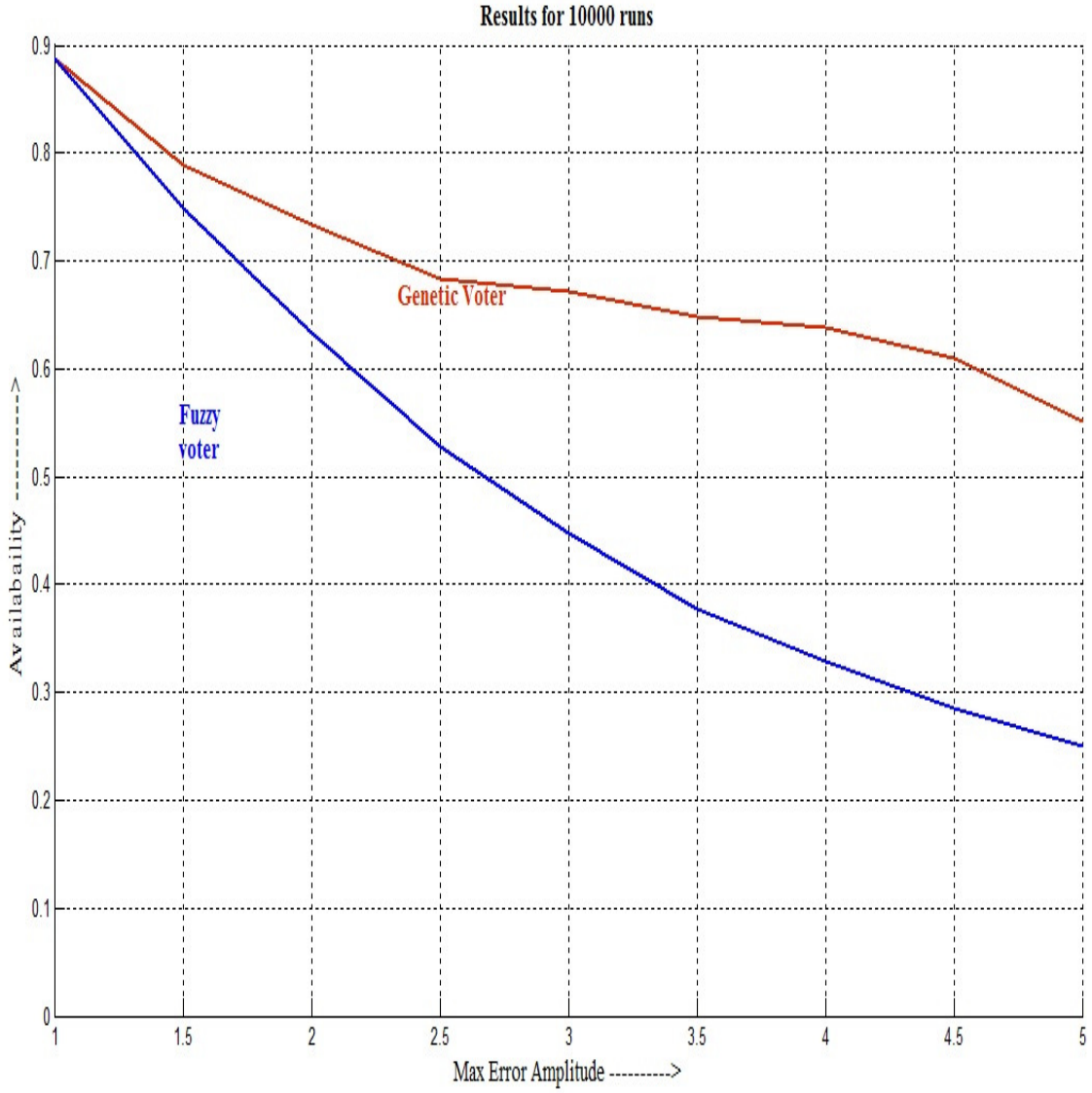

FIGURE 5.1(a): Performance comparison of voters in terms of availability when fault injection in two modules is within the same range

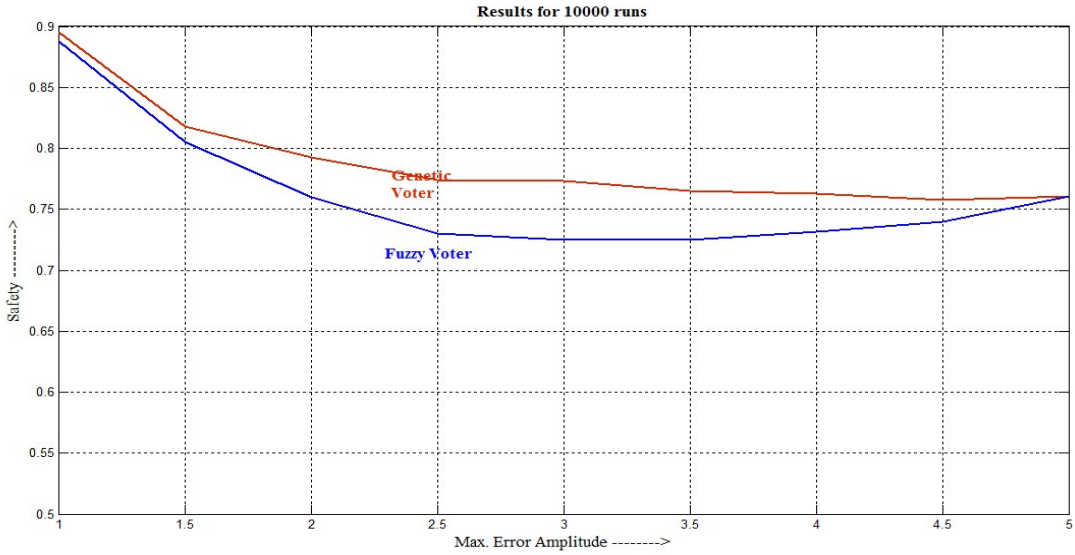

FIGURE 5.1(b): Performance comparison of voters in terms of safety when fault injection in two modules is within the same range

Figure 5.2(a) and 5.2(b) shows the availability and safety performance respectively of two voters when one module is fault free, second module is more faulty where the fault has been injected randomly with uniform distribution in range [-5 +5], and fault in third module is varied from [$-e_{max}$ $+e_{max}$]. The x-axis in graph shows the value of maximum amplitude e_{max}. It is observed that without compromising with safety our proposed voter gives [1-35] % better availability in this case as compared to fuzzy voter.

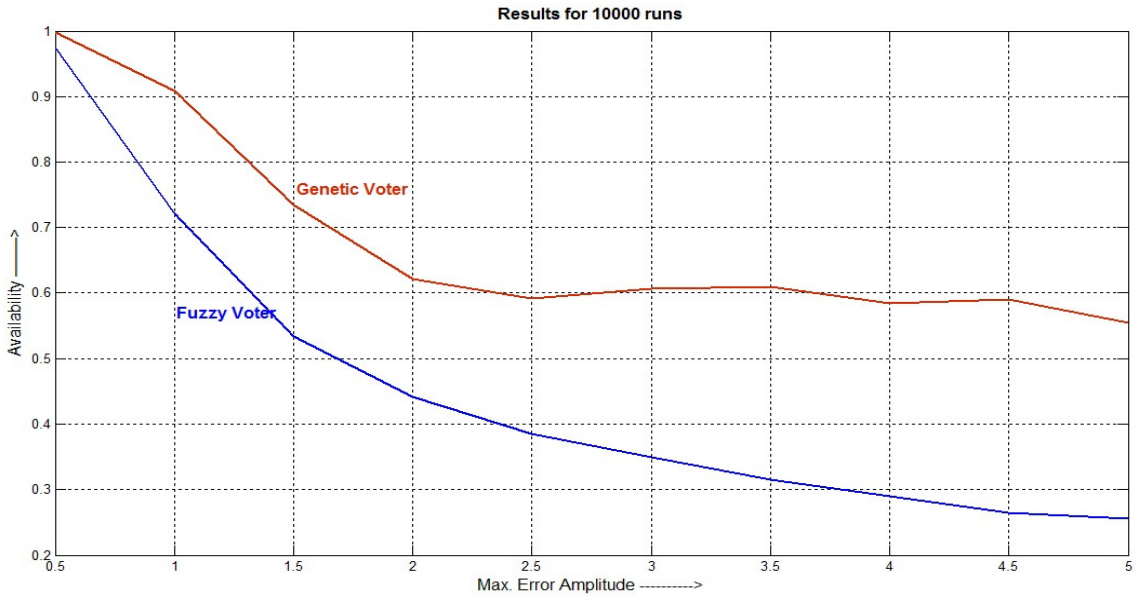

FIGURE 5.2 (a): Performance comparison of voters in terms of availability when fault injection in two modules is in the different ranges.

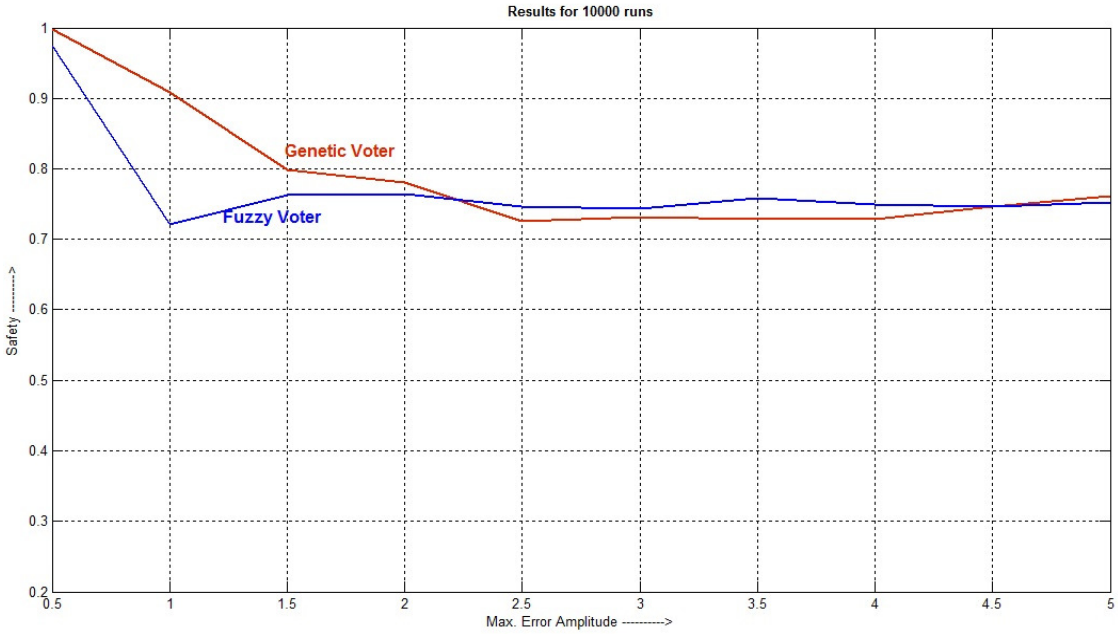

FIGURE 5.2 (b): Performance comparison of voters in terms of safety when fault injection in two modules is in the different ranges.

The majority voting techniques for analog inputs work towards an approximation for approval of agreeability between the modules. It becomes a challenging task to evolve a method for the approximation which is a compromise between safety and availability of the system output. The aim is to suggest a scheme to maximise both these parameters so that these schemes can be used as a generic algorithm for critical fault tolerant systems. Among the existing algorithms reviewed in the literature it has been seen that the fuzzy voter proposed by Shabgahi provides a better fault tolerant solution both in terms of the safety and availability parameters. However the voter falls short of expectations for larger errors. The new hybrid scheme based on genetic algorithm and performance history has been validated through simulation and found to be better in terms of availability [0-32]% for all the cases with higher magnitude and the safety parameter has been ensured to be better to some extent [0-8]%.

6. CONCLUSION

In this paper we propose a novel hybrid voter using genetic method and performance history of the modules. The algorithm has been simulated using MATLAB and its performance with respect to safety and availability has been found to be better than the existing voting techniques. This voter can be implemented for systems requiring high availability without compromising with the safety; however for time critical applications better genetic search techniques would be required to ensure time bound response of the system.

REFERENCES:

Books:

[1] Parag K. Lala, “Fault Tolerant and Fault Testable Hardware Design”, BS Publications, ISBN: 81-7800-038-5

[2] David E. Goldberg, “Genetic Algorithms in Search, Optimization & Machine Learning”, Addison-Wesley Publications, ISBN : 981-405-394-5

Journals:

[3] Von Neumann, J., “Probabilistic logics and synthesis of reliable organisms from unreliable components”, Automata Studies, in Annals of Mathematical Studies, No. 34, 43-98(ED.: C. E. Shannon and J. McCarthy), Princeton University Press (1956).

[4] G. Latif-Shabgahi, A.J. Hirst, “A Fuzzy Voting Scheme for Hardware and Software Fault Tolerant Systems”, Fuzzy Sets and Systems 150 (2005) 579–598, Elsevier Publication.

[5] M.D. Krstic, M.K. Stojcev G. Lj. Djordjevic and I.D. Andrejic," A mid-value select voter ",Microelectronics Reliability, Volume 45, Issues 3-4, March-April 2005, Pages 733-738, Elsevier Publication.

[6] Behrooz Parhami, "Voting algorithms", IEEE Trans. Reliability, Vol. 43, No. 4, pp. 617-629, December 1994.

Conference Proceedings:

[7] G. Latif-Shabgahi, J. M. Bass, S. Bennett, “History-based weighted average voter: a novel software voting algorithm for fault-tolerant computer systems,” in 9th workshop Parallel and Distributed Processing, 2001, pp. 402-409

[8] Milos Manic, Deborah Frincke, "Towards the Fault Tolerant Software: Fuzzy Extension of Crisp Equivalence Voters", IECON’01: The 27th Annual conference of the IEEE Industrial electronics Society, 2001.

[9] Behrooz Parhami, "Optimal Algorithm for Exact, Inexact and Approval voting", FTCS- 22: Twenty second International symposium on fault tolerant computing, Boston, July 1992, pp. 404-411.

Electronic References:

[10] Darrel Whitley, “A genetic algorithm tutorial”, Statistics and Computing (1994) 4, 65-85.

[11] John H Holland, “Genetic Algorithms”, Internet: http://www2.econ.iastate.edu/tesfatsi/holland.gaintro.htm