Genetic Programming Approach for Fault Modeling of Electronic Hardware

Ajith Abraham
School of Computer Science and Engineering
Chung-Ang University, Seoul, South Korea
ajith.abraham@ieee.org

Crina Grosan
Department of Computer Science
Babes-Bolyai University, Cluj-Napoca, 3400, Romania
cgrosan@cs.ubbcluj.ro

Abstract- **This paper presents two variants of Genetic Programming (GP) approaches for intelligent online performance monitoring of electronic circuits and systems. Reliability modeling of electronic circuits can be best performed by the stressor – susceptibility interaction model. A circuit or a system is deemed to be failed once the stressor has exceeded the susceptibility limits. For on-line prediction, validated stressor vectors may be obtained by direct measurements or sensors, which after preprocessing and standardization are fed into the GP models. Empirical results are compared with artificial neural networks trained using bakpropagation algorithm. The performance of the proposed method is evaluated by comparing the experiment results with the actual failure model values. The developed model reveals that GP could play an important role for future fault monitoring systems.**

1 Introduction

Real time monitoring of the healthiness of complex electronic systems/circuits/hardware is a difficult challenge to both human operators and expert systems. When the electronic circuit or system is controlling a critical task fault prediction will be very important. This paper proposes a stressor-susceptibility interaction model for analyzing the hardware and two variants of genetic programming methods for approximating the various complex functions to monitor the performance of the system.

Stressor is a physical entity influencing the lifetime of a component or circuit. A stressor, indicating a physical entity x will be denoted as ψ_X. Stressors can be broadly classified into three main groups. First group contains the electrical stressors, parameters related to the electrical behavior of the circuit. Second group of stressors is the mechanical stressors, which are related to the mechanical environment of the component. Third group of parameters influencing the lifetime of components is related to the thermal environment of the component. Susceptibility of a component to a certain failure mechanism is defined as the probability function indicating the probability that a component will not remain operational for a certain time under a given combination of stressors. The susceptibility related to the failure mechanism y is usually defined as S_y $(t, \psi_p, \psi_q, \psi_r)$.

The new technique of electronic system failure prediction using stressor- susceptibility interaction [1][2][6] is briefly discussed in Section 2. This technique can be extended to simple electronic components and for complicated electronic circuits and equipment. Section 3 presents some of the common failure mechanisms in practical situations. The derivation of stressor sets using Monte Carlo Analysis is given in Section 4 followed by Section 5 where we had derived a stressor-susceptibility model for a circuit. Section 6 gives some theoretical background about genetic programming models used and artificial neural networks. In section 7 we have reported the experiment results and finally conclusions are provided in Section 8.

2 Stressor-Susceptibility Interaction

Failure probabilities require detailed analysis of both stressors and susceptibility. Most components tend to have more than one failure mechanism, resulting in more than one "failure probability". It can be shown that there is a strong correlation between the various failure mechanisms existing within a component.

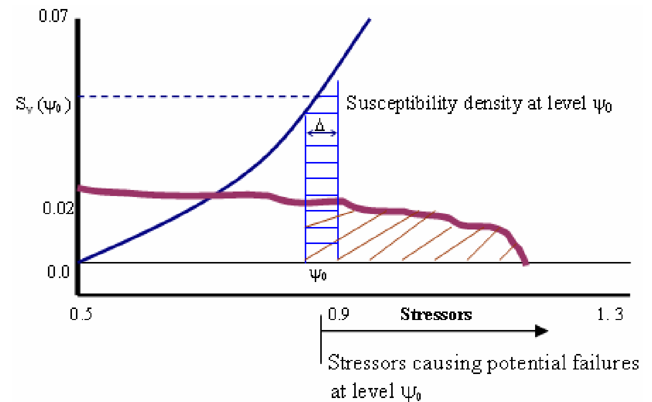

Figure 1. Stressor-Susceptibility interaction for single failure mechanism.

Figure 1 illustrates the stressor - susceptibility interaction for a single failure mechanism. The main source of problem is the overlap between stressor and susceptibility density. The first step is to calculate the failure probability for this stressor distribution on a failure mechanism with a single, one variable, time independent catastrophic susceptibility model. This results in the probability function given by (1)

$$f_{fail,\,y,\,\Psi}(\Psi_0) = \int_{\Psi_0}^{\infty} f_y(\Psi)\,d\Psi \quad (1)$$

To calculate the failure probability as a function of more complex susceptibility model, it will be necessary to calculate the failure probability of a part of the susceptibility model, for a certain stressor interval Δ, characterized by its mean value ψ_o and the corresponding susceptibility density function at that point $S_y(\psi_o)$. Considering the probability that a part has failed at a lower susceptibility level, result in the possibility to predict the failure probability per time interval of a certain failure at stressor level ψ_0 using (2).

$$f_{fail,y,\Psi}(\Psi_0) = \Delta(S_Y(\Psi_0)\int_{\Psi_0}^{\infty} f_y(\Psi)d\Psi)\left(1 - \int_{0}^{\psi_0-\Delta} f_{fail,y}(\psi)d\psi\right) \quad (2)$$

The last term is introduced to subtract failures caused by stressors at a lower susceptibility level. As, most often, failure probabilities are very small, in many cases the previous expression will simplify to (3).

$$f_{fail,y,\psi}(\psi_0) = \Delta(S_y(\Psi_0)\int_{\Psi_0}^{\infty} f_y(\Psi)d\Psi) \quad (3)$$

$$\text{when } (1 - \int_{0}^{\psi_0-\Delta} f_{fail,y}(\psi)d\psi) = 1 \quad (4)$$

Since the susceptibility is defined as the probability that a component will not remain operational during a certain time, it is therefore possible to calculate the failure probability during a certain observation time t_{obs}.

$$f_{fail,y,\psi}(\psi_0) = t_{obs} * \Delta * (S_y(\Psi_0)\int_{\Psi_0}^{\infty} f_y(\Psi)d\Psi) \quad (5)$$

The important requirement for using (5) is that the observation time t_{obs} must be larger than the total elapsed sampling time to obtain an ergodic description of the associated stressors $t_{total\ sample}$ ($t_{obs} > t_{total\ sample}$); $f_{ail,\ y\ ,\ \psi}$, (t, ψ) is assumed to be constant during the time interval t_{obs}.
From (5) it is possible to calculate the failure probability of a part per fail mechanism per time interval using (6).

$$f_{fail,y} = \int_{0}^{\infty} f_{fail,y,\psi}(t,\psi)d\psi \quad (6)$$

Equation (6) can now be used to calculate the part failure probability per time interval

$$f_{fail} = \sum_{i=1}^{n} f_{fail,i} \quad (7)$$

Using the previous assumptions it is also possible to calculate the probability that a component survives from time t to $t+dt$. Equation (8) can be used to calculate the failure probability for one single failure mechanism within one single device.

$$R(t\ ....\ t+\Delta t) = \frac{\sum_{i=1}^{k} Devices\ operational\ at\ time\ (t+\Delta t)}{\sum_{i=1}^{n} Devices\ operational\ at\ (t)} = R(t)F(\Delta t) = R(t)\Delta t f(t) \quad (8)$$

As for large series of components, the physical structures of the individual components will be different for every component, the survival probability of such a series of components will also show individual differences. The stress on a component may vary with time due to circuit behavior and circuit use. The circuit behavior will differ amongst a series of circuits due to physical differences in the individual circuit components, the physical structure of a circuit, the use of a circuit and the environment (electrical, thermal, etc.) of the circuit. To summarize the variety of effects it is useful to describe stressors as stochastic signals with properties depending on the influencing factors mentioned above. These assumptions make it possible to derive the failure probability and reliability of a component using a Markov approach.

For Markov approach the following requirements should be fulfilled

- Susceptibility of all failure mechanisms in a component is known and is constant in the time interval (t, $t+\Delta t$).
- All stressors ψ_a (t), ψ_b (t), … are known as stochastic signals for the time interval (t, $t+\Delta t$).
- The failure probability (or reliability) is known at a certain (initial) time t.

Using these properties it is possible to calculate the reliability and failure probability for components, derived from internal failure mechanisms for time $t+\Delta t$. For this purpose the following relationships are used

$$P(t+\Delta t) = P(t) ✪ (\Delta t)$$

$$= \vec{P}(t)\begin{bmatrix} P_{x\to x} & \cdots & \cdots & P_{x\to y} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ P_{y\to x} & \cdots & \cdots & P_{y\to y} \end{bmatrix} \quad (9)$$

where P(t) is the state probability vector of a component. This state probability vector is defined as

$\vec{P}(t) =$

$P_{operational}(t)$: probability that part is operational at time t
$P_a(t)$: probability that part fails due to failure mechanism a at time t
$P_b(t)$: probability that part fails due to failure mechanism b at time t
$P_n(t)$: probability that part fails due to failure mechanism n at time t

Table 1. Some common Failure mechanisms with associated causes and stressors

No.	Failure Mechanism	Influencing aspect or associated stressors
1	Thermal Failure (general)	• Dissipated power • Environmental Temperature • Thermal resistance • Thermal capacitance
2	Current Breakdown (Hot spot melting)	• Resistivity of the material • Impurities/ mechanical distortions in the material causing increase in current density. • Thermal resistivity coefficient.
3	Power breakdown (Thermal cracks)	• Thermal expansion coefficient of the materials. • Thermal resistivity coefficients of the materials.
4	Impact Ionization	• Electric field
5	Avalanche breakdown	• Electric field (positive temperature coefficient)
6	Zener breakdown	• Electric field (negative temperature coefficient)
7	Corrosion	• Environmental temperature (negative influence on susceptibility) • Dissipated power • D C Voltage
8	Electromigration	• Current density • Environmental temperature
9	Secondary diffusion	• Temperature
10	Switch on pulse power dissipation (for bipolar junctions)	• Voltage slope dV/dt • Current slope dI/dt
11	Switch off pulse power dissipation (for bipolar junctions)	• Voltage slope dV/dt • Maximum reverse junction current • Applied reverse voltage • Storage charge Q's in the diode at the moment of polarity reversal.
12	Forward bias second breakdown (for power transistors)	• Collector emitter voltage • Slope of the base current during switching on dI_b/dt • Slope of the collector current during switching on dI_c/dt • Environmental temperature
13	Reverse bias second breakdown (for power transistors)	• Collector emitter voltage • Discharge speed dI_b/dt (optimum value) • Stored charge at the moment of transistor switch off (closely related to collector current at the moment of switch off). • Environmental temperature

$$\sum_{j=1}^{n} P_j(t) = 1$$

$$P_1(t) = P_{operational}(t) = R(t)$$

$$P_{2\ldots n}(t) = P_{fail,\,2\ldots n}(t) = F_{fail,\,2\ldots n}(t)$$

$$P_{x\prod y} = P_{(y\,(t+\Delta t)\,|\,x\,(t))} = f_y(t)\,\Delta t\;P_x(t)$$

It is possible to replicate this calculation process for a whole batch of circuits. In this case, for every circuit the individual stressor/ susceptibility interaction is calculated thus simulating batch behavior. Using this method, it is possible to derive the failure probability for many parts in many practical situations, also in cases where considerable differences (in stressors and susceptibility) exist within a batch.

3. Modeling Stressor Sets and Susceptibility

There are two different categories of failure mechanisms applicable to electronic components [3][7][8][9]. First, the failure mechanisms that are related to the electrical stress in a circuit. Second there are failure mechanisms related to the intrinsic aspects of a component [4][6]. Table 1 shows some of the typical failure mechanisms and their causes with associated stressors. There are two possible ways to obtain stressor sets for practical circuits. The first possibility involves usage of computer simulation models to derive all circuit signals using one single

simulation. Second possibility is to derive stressor sets from practical measurements. In those cases where sufficient systems are available it is possible to do a statistical evaluation of the individual stressor functions existing in individual systems. As the stressor sets are dependent on the conditions of use and the operation modes of a system it is important that the measured stressor is based on all the possible operation modes of a circuit and all the possible transitions between the various operation modes. This can become a quite tedious job as the entire operation is to be repeated for a number of systems to obtain an accurate statistical mean stressor model. Accurate description of a stressor set needs a sampling frequency of at least twice the highest frequency in the stressor frequency spectrum. Accurate description of a stressor set will require a number of samples sufficient to cover all the different states of the system. As a signal has often more than one quasi-stationary states, each characterized by their stressor set, it is possible to derive the overall stressor set function from the individual state stressor sets using (10)

$$f_{str,y(x)} = \sum_{i=1}^{n} \frac{T_i}{T_{total}} f_{str,y,i(x)} \qquad (10)$$

$f_{str,y\,(x)}$ is the stressor probability density function of quasi-stationary state i. T_i / T_{total} is the fraction of time that the stressor is in quasi-stationary state i.

4. Monte Carlo Analysis for Stressor Sets

In a Monte Carlo Analysis (MCA), a logical model of the system being analyzed is repeatedly evaluated, each run using different values of the distributed parameters. The selection of parameter values is made randomly, but with probabilities governed by the relevant distribution functions. Statistical exploration covers the tolerance space by means of the generation of sets of random parameters within this tolerance space. Each set of random parameters represents one circuit. Multiple circuit simulations, each with a new set of random parameters, explore the tolerance space. Statistically the distribution of all random selections of one parameter represents the parameter distribution.

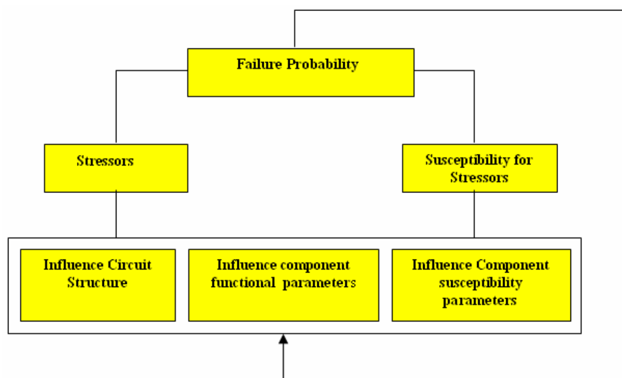

Figure 2. Monte Carlo analysis

Although the number of simulations required for MCA is quite large, this analysis method is useful, especially because the number of parameters in the failure prediction of circuits is often too large to allow the use of other techniques. Figure 2 illustrates the MCA. With MCA it is possible to simulate the behavior of a large batch of circuits and derive stressor sets. The next phase will be the combination of the derived stressor sets with the component susceptibilities in order to decide whether a component will fail or not. As for the failure prediction, the most important aspect is to prevent failures; susceptibility will be expressed using the susceptibility limit. To distinguish circuits where failures are possible any circuit in the MCA causing to exceed a susceptibility limit are marked as fail. Circuits where no stressors exceed susceptibility limit are marked as pass.

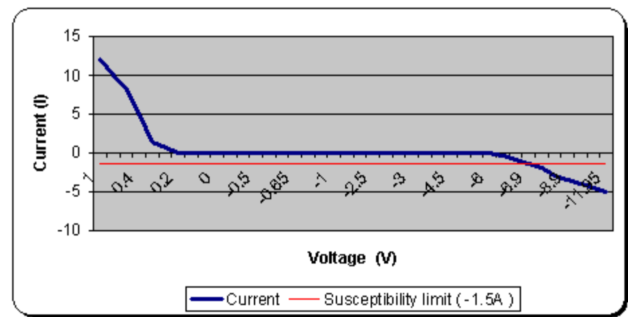

Figure 3. Stressor - susceptibility interaction model

5. Modeling Stressor Sets and Susceptibility

The analysis was carried out on a power circuit and the main cause of the failure of the circuit was a Schottky diode. Analysis shows that the main failure mechanisms are leakage current and excess crystal temperature. Using the procedure described earlier, it was possible to derive a complete individual stressor set for the failure mechanism of this diode. Figure 3 illustrates the joint stressor – susceptibility interaction model in terms of voltage and current. The susceptibility limit for leakage current is set at -1.5A.

6. Intelligent Paradigms

Linear Genetic Programming (LGP)

Linear genetic programming is a variant of the GP technique that acts on linear genomes [5]. Its main characteristics in comparison to tree-based GP lies in that the evolvable units are not the expressions of a functional programming language (like LISP), but the programs of an imperative language (like c/c ++). An alternate approach is to evolve a computer program at the machine code level, using lower level representations for the individuals.

The basic unit of evolution here is a native machine code instruction that runs on the floating-point processor unit (FPU). Since different instructions may have different sizes, here instructions are clubbed up together to form instruction blocks of 32 bits each. The instruction blocks hold one or more native machine code instructions, depending on the sizes of the instructions. A crossover point can occur only between instructions and is prohibited from occurring within an instruction. However the mutation operation does not have any such restriction.

The settings of various linear genetic programming system parameters are of utmost importance for successful performance of the system. The population space has been subdivided into multiple subpopulation or demes. Migration of individuals among the subpopulations causes evolution of the entire population. It helps to maintain diversity in the population, as migration is restricted among the demes. Moreover, the tendency towards a bad local minimum in one deme can be countered by other demes with better search directions. The various LGP search parameters are the mutation frequency, crossover frequency and the reproduction frequency: The crossover operator acts by exchanging sequences of instructions between two tournament winners. Steady state genetic programming approach was used to manage the memory more effectively.

Multi Expression Programming (MEP)

A GP chromosome generally encodes a single expression (computer program). By contrast, a MEP chromosome encodes several expressions [10]. MEP genes are represented by substrings of a variable length. The number of genes per chromosome is constant. This number defines the length of the chromosome. Each gene encodes a terminal or a function symbol. A gene that encodes a function includes pointers towards the function arguments. Function arguments always have indices of lower values than the position of the function itself in the chromosome [11].

Each MEP chromosome is allowed to encode a number of expressions equal to the chromosome length (number of genes). The value of these expressions may be computed by reading the chromosome top down. Partial results are computed by dynamic programming and are stored in a conventional manner. Due to its multi expression representation, each MEP chromosome may be viewed as a forest of trees rather than as a single tree, which is the case of Genetic Programming. As MEP chromosome encodes more than one problem solution, it is interesting to see how the fitness is assigned. The chromosome fitness is usually defined as the fitness of the best expression encoded by that chromosome.

Artificial Neural Networks (ANN)

Artificial Neural Networks have been developed as generalizations of mathematical models of biological nervous systems. A neural network is characterised by the network architecture, the connection strength between pairs of neurons (weights), node properties, and updating rules. The updating or learning rules control weights and/or states of the processing elements (neurons). Normally, an objective function is defined that represents the complete status of the network, and its set of minima corresponds to different stable states of the network. It can learn by adapting its weights to changes in the surrounding environment, can handle imprecise information, and generalise from known tasks to unknown ones. Each neuron is an elementary processor with primitive operations, like summing the weighted inputs coming to it and then amplifying or thresholding the sum. Learning typically occurs by example through training, where the training algorithm iteratively adjusts the connection weights (synapses). Backpropagation (BP) is one of the most famous training algorithms for multilayer perceptrons. BP is a gradient descent technique to minimize the error E for a particular training pattern. For adjusting the weight (w_{ij}) from the i-th input unit to the j-th output, in the batched mode variant the descent is based on the gradient ∇E ($\frac{\delta E}{\delta w_{ij}}$) for the total training set:

$$\Delta w_{ij}(n) = -\varepsilon * \frac{\delta E}{\delta w_{ij}} + \alpha * \Delta w_{ij}(n-1) \qquad (11)$$

The gradient gives the direction of error E. The parameters ε and α are the learning rate and momentum respectively.

7. Experiment Results

The experiment system consists of two stages: model construction (training) and performance evaluation. The stressor – susceptibility interaction model was analyzed in detail (as illustrated in Figure 3) and the main causes of failures were identified. Analysis showed that the main cause of the failure was excess junction temperature and leakage current. A mathematical model was built relating the failure probability, leakage current and junction temperature. A failure simulation was carried out and the data set was generated. We attempted to predict the component temperature and leakage current for a given voltage and current. Data was generated by simulating circuit failure. 80% of the randomly selected data was used for training and remaining for testing and validation purposes. All the training data were standardized before training. The input parameters considered are the Voltage (V) and Current (I). Predicted outputs are the junction temperature and leakage current.

- **LGP Training**

After a trial and error approach, the following parameter settings were used for the experiments:
Population size: 500
Number of generations: 100
Mutation frequency: 90%
Crossover frequency: 60%
Number of demes: 10

Maximum program size: 256
Target subset size: 100

- **MEP Training**

The following parameter values were used by MEP:
Population size: 500
Number of generations: 300
Chromosome length: 40
Number of mutations per chromosome: 4
Crossover probability: 0.9

- **ANN Training**

We used a feedforward neural network with 2 hidden layers in parallel, 2 input neurons corresponding to the input variables and 3 output neurons. Initial weights, learning rate and momentum used were 0.3, 0.1 and 0.1, respectively. The training was terminated after 3500 epochs.

Performance and Results Achieved

Figures 4 and 6 illustrate the evolved models using LGP for junction temperature and leakage current prediction using the evolved models. Figures 5 and 7 depict the average code length and best code length (program size) for the two prediction models using LGP. Table 2 summarizes the comparative performance of LGP, MEP and ANN. As illustrated in Figure 8, the MEP models converged after 300 generations.

Table 2. Performance comparison

	Root Mean Squared Error (RMSE)		
	LGP	**MEP**	**ANN**
	Training		
Junction temperature	0.00948	0.00593	0.00697
Leakage current	**0.00493**	0.00829	0.00589
	Testing		
Junction temperature	0.00911	0.01034	0.01278
Leakage current	**0.00493**	0.010032	0.00359

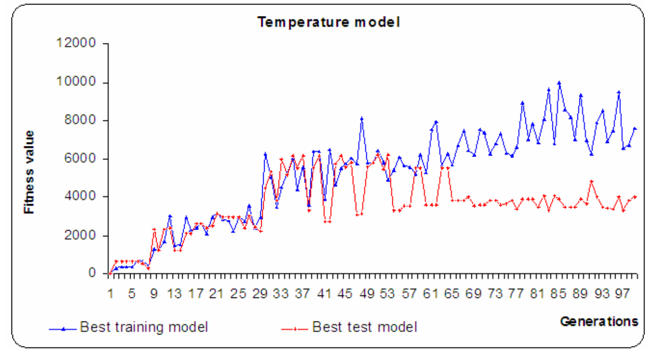

Figure 4. Evolved models for temperature

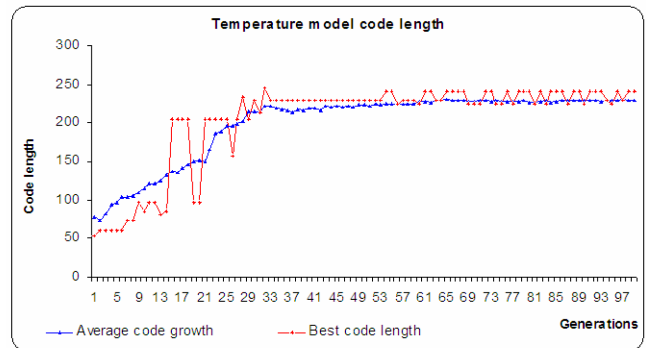

Figure 5. Program size growth for the temperature model

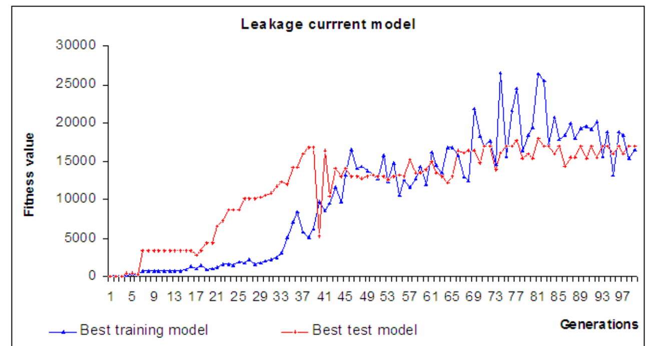

Figure 6. Evolved models for leakage current

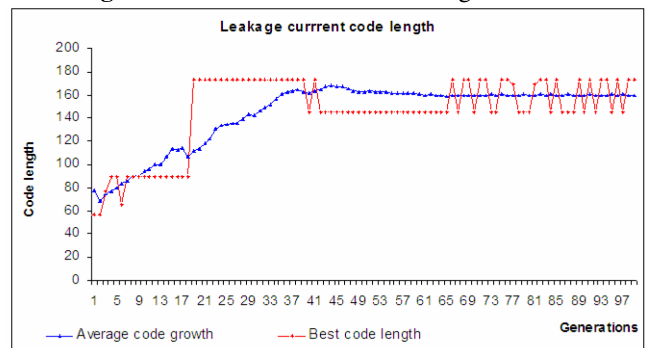

Figure 7. Program size growth for leakage current

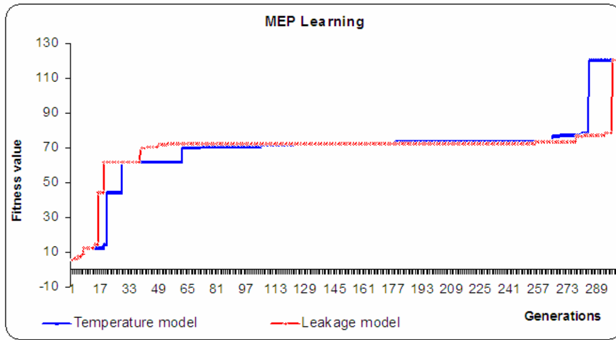

Figure 8. MEP training for the two models

Following are the evolved functions using MEP.

Leakage current model:

(((cos(x[1] - (x[0] * x[0]) - (x[1] > (x[0] * x[0]) ? x[1] : (x[0] * x[0]) - (x[1] - (x[0] * x[0]))))) > x[1] ? (cos(x[1] - (x[0] * x[0]) - (x[1] > (x[0] * x[0]) ? x[1] : (x[0] * x[0]) - (x[1] - (x[0] * x[0]))))) : x[1]) / (sin(x[0] * x[0] + x[0]))) < ((fabs(0.0605182042909024)) / 0.0605182042909024) ? (((cos(x[1] - (x[0] * x[0]) - (x[1] > (x[0] * x[0]) ? x[1] : (x[0] * x[0]) - (x[1] - (x[0] * x[0]))))) > x[1] ? (cos(x[1] - (x[0] * x[0]) - (x[1] > (x[0] * x[0]) ? x[1] : (x[0] * x[0]) - (x[1] - (x[0] * x[0]))))) : x[1])/(sin(x[0]*x[0]+ x[0]))):((fabs(0.0605182042909024))/ 0.0605182042909024)

Temperature model:

0.80123294778283 > (fabs((x[0] + Log2(x[0])) / ((Lg(x[0])) > x[0] ? (Lg(x[0])) : x[0]))) ? 0.80123294778283 : (fabs((x[0] + Log2(x[0])) / ((Lg(x[0])) > x[0] ? (Lg(x[0])) : x[0])))

8. Conclusions

In this paper, we attempted to predict the failures of electronic circuits and systems using two variants of genetic programming and the performance were compared using artificial neural networks. The proposed GP models seems to work very well with LGP giving the optimal performance for modeling leakage current and junction temperature. Compared to neural network, an important advantage of the GP models is its simplicity in implementing directly in the hardware itself. As depicted in Section 7 (MEP evolved functions), the massive neural network could be replaced by simple functions using hardware or light software.

The developed models should be also reliable during worst conditions. Our future research will be targeted in evaluating the developed GP models for robustness and handling of noisy and approximate data that are typical in circuits.

The problem modeling using stressor– susceptibility interaction method can be widely applied to a wide range of electronic circuits or systems. However, it requires intense knowledge on the circuit behavior to model the various dependent input parameters to predict the results accurately.

References

[1] Abraham A. and Nath B., Failure Prediction of Critical Electronic Systems in Power Plants Using Artificial Neural Networks, First International Power and Energy Conference, INTPEC99, Australia, November 1999.

[2] Abraham A., A Soft Computing Approach for Fault Prediction of Electronic Systems, In Proceedings of The Second International Conference on Computers in Industry, Published by The Bahrain Society of Engineers, Majeed A Karim et al (Eds.), pp. 83-91, 2000.

[3] Chan A.H., A formulation of environmental stress testing and screening, Proceedings of the Annual Reliability and Maintainability Symposium (IEEE Reliability Society), pp 99-104, January 1994.

[4] Arsenault J.E. and Roberts J.A., Reliability and maintainability of electronic systems, Computer Science press, Maryland, 1980.

[5] Banzhaf. W., Nordin. P., Keller. E. R., Francone F. D., Genetic Programming: An Introduction on The Automatic Evolution of Computer Programs and its Applications, Morgan Kaufmann Publishers, Inc., 1998.

[6] Brombacher A.C, Reliability by Design, Wiley, 1995

[7] Jenson F., Electronic Component Reliability, Wiley, 1995.

[8] Klion J., Practical Electronic Reliability Engineering, VNR Edition, 1992.

[9] Fuqua N.B., Reliability Engineering for Electronic Design, Marcel Dekker, 1987.

[10] Oltean M. and Grosan C., A Comparison of Several Linear GP Techniques, Complex Systems, Vol. 14, Nr. 4, pp. 285-313, 2004.

[11] Oltean M. and Grosan C., Evolving Evolutionary Algorithms using Multi Expression Programming. Proceedings of The 7th European Conference on Artificial Life, Dortmund, Germany, pp. 651-658, 2003.