Comparisons of Self-Healing Fault-Tolerant Computing Schemes

Huan-yu Tu

***ABSTRACT* – Fault tolerance has always been a critical feature for reliable electronic systems. As the systems are becoming more and more complex, novel approaches inspired from bio-system have become an interested research field. Biological systems possessed characteristics such as self-healing and self-reproduction that are similar to the requirement of fault tolerance design. This paper investigates and compares three major domains in bio-inspired self-healing fault tolerance techniques. The results are presented and discussed here.**

Keywords – self-healing, fault tolerance, simulations

1. INTRODUCTION

As electronics and computing systems are becoming more and more complex, comprehensive fault testing becomes more and more difficult as well, if not impossible at all. Hence, faults can remain in a system, or be introduced into a system anytime during the operation. These faults can ultimately cause a system failure. Research is continuously looking for efficient ways to improve the ability of a system to function correctly in the presence of faults, to become fault tolerant.

The basic principle in fault tolerant system design is redundancy. Traditional fault tolerant techniques include two types of redundancy: time and information. The scheme behind these techniques is pretty simple and straightforward, but the extra equipment and design time required bring significant cost to the system as well.

One alternative way to achieve fault tolerance comes from biological systems. In recent years, research has taken an interest in how biological organisms manage to survive through self-healing and self-reproduction. In general, the emerging bio-inspired systems can be classified into three distinct domains [1]:

1) *Epigenesis:* It relates to learning within a species. The field of artificial neural networks (ANNs) is the largest research field within this area. From the viewpoint of fault tolerance, *immunotronics* is a rapidly rising topic.

2) *Ontogeny:* It is concerned with the development of an individual from a single mother or *zygote* cell through to a multicellular system. The field of *embryonics* has developed fault-tolerant electronic systems based upon a cellular structure, whereby each cell can take on the role of any other cell within the system. Recent work has also mathematically shown that embryonic systems can provide better reliability [2].

3) *Phylogeny:* It is related to the evolution through the generations of a species. Bio-inspired fault tolerance research exists in this domain in the form of evolvable hardware (EHW). Research has investigated the evolution of devices through the evolution of populations of circuits.

The three domains have developed largely along their own individual paths, although there have been proposals to combine the concepts from more than one domain [3].

Huan-Yu Tu is with the Department of Mathematics and Computer Science, Eastern Connecticut State University, Willimantic CT 06226. USA. (e-mail: tuh@easternct.edu).

1.1 Human immune system

After birth, every living creature needs some way to protect itself from hazardous external influences. This ability to respond to changes in the environment is not only inherited, but also obtained through learning. In order to achieve this adaptation, every individual is born with several different levels of systems, which are defined to undertake modification corresponding to the interaction with the outside world. One of these systems is the immune system. Human immune system works as a multi-layered defense system, and is capable of preventing infections from approximately 10^{16} foreign sources.

From a genetic point of view, the immune system has two fundamental characteristics. First, it must recognize 'self' so that an individual's immune system won't attack his own cells and tissues. Second, it must be able to detect 'non-self' so that intruders would be recognized and then destroyed. This ability to distinguish self/non-self elements is also crucial to a fault tolerant electronic system since error detection is the prerequisite to fault removal. The similarity between the requirements of fault tolerance and the defense mechanism of human body imply a mapping from the immune system to the design of reliable systems.

The immune system provides a series of highly specific defensive response to the entry of foreign substances. Some remarkable properties demonstrated by biological immune systems include:

1. Detection of known/unknown intruders.
2. Elimination/neutralization of intruders.
3. Remember these previously unknown intruders.
4. Selective proliferation and self-replication for future quick recognition.

In its simplest form, the underlying process is self/nonself differentiation, to determine what is a cell of the body (a valid state/transition in the hardware) and what is not (an invalid state/transition in the hardware). Humoral immunity uses B cells to generate antibodies and helper T cells to activate the

ISBN: 978-988-17012-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

production of antibodies. A brief description of the procedure is introduced as below and shown in Figure 1.

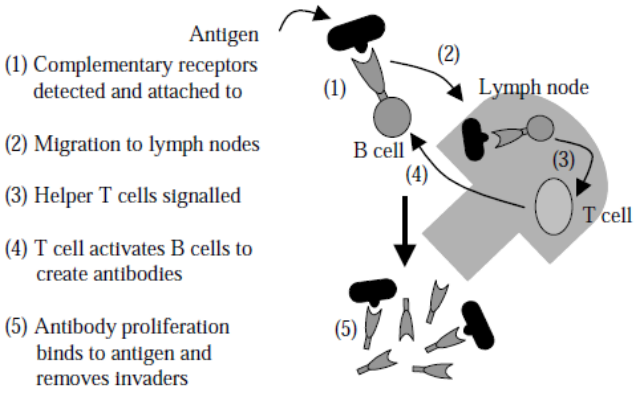

Figure 1: The humoral immune response, from [5]

Immature helper T cells are sent to thymus directly from the bone marrow, where they are exposed to all the self cells or proteins. This is to make sure that an immune response can only be initiated against cells not belonging to the body. If any biding between receptors on a helper T cell and a self-cell occur, then the immature helper T cell is destroyed. At the end of this 'learning' stage, all the left T cells do not bind to any of those self cells or proteins. They are called matured T cells (only 1%-5% of those immature T cells will survive), and then migrate into lymph nodes throughout the body to create a distributed detection network. B cells exist in the whole circulatory system and tissue looking for possible foreign intruders. Those cells that have complementary receptors will be picked up and get bound to B cells. Once a pair of B cell and foreign cell is created, both of them will be passed to lymph nodes. The helper T cells there will determine if the collected cell should be attacked according to the protein fragments provided. If a match occurs, the B cell will be signaled to manufacture and proliferate antibodies, and the collected cell will be destroyed.

1.2 Software fault tolerance

Immunological approaches are used in computer security, virus protection, and software tolerance. Negative algorithm developed by Forrest and Perelson is one major implementation. Figure 2 adopted from [6] demonstrates the algorithm.

In contrast with existing fault tolerance architectures such as NMR, which checks for the presence of valid operation constantly, negative selection algorithm works by checking for the presence of invalid operation. From a randomly generated original set of data *R* (immature T cells) selects a set of strings R_o (the matured helper T cells) of length *l*, which fail to match any self-string $s_i \in S$ (cells in the body), also of length *l*, in at least *c* contiguous positions. The probability of a match between two random strings (here between a self string $s \in S$ and a randomly generated immature tolerance condition r_o) in at least *c* contiguous positions is given by equation 1.

$$P_M \approx m^{-c}[(l-c)(m-1)/m+1] \quad (1)$$

where $m^{-c} << 1$ and *m* is the number of alphabet symbols (2 for a binary FSM).

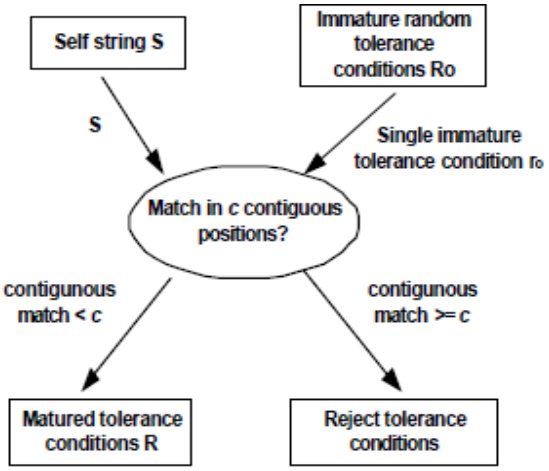

Figure 2: Negative selection algorithm is used to create a set of tolerance conditions R that do not match any self strings in c contiguous positions. Adopted from [6].

Any string that match in at least *c* contiguous positions are deleted. In order to create *R*, a set of tolerance conditions, with a sufficiently low failure probability, the number of valid self strings must be much less than the number of invalid strings. If the number of tolerance conditions Nr is limited, the theoretical probability that the system fails to detect non-self is given by

$$P_{f_t} = (1-P_m)^{N_r} \quad (2)$$

A trade off exists between the number of tolerance conditions and the probability of failure in fault detection.

Table 1: Mapping of immune system to hardware fault tolerance. Adopted from [4]

Immune system	**Hardware fault tolerance**
Self cells, proteins	Valid states, transitions
Non-self intruders	Invalid states, transitions
B cells	System state condition comparison
T cells	Stored tolerance conditions
Learning during gestation	Generation of tolerance conditions

1.3 Hardware fault tolerance

Similarly, the features and operations of immune system could be translated into hardware fault tolerant system. Table 1 shows the mapping and summarizes the analogies; it also implies that an immunologically inspired approach based upon the use of Finite State Machine (FSM) is feasible.

The hardware immune system uses an FSM to represent the system to be immunized, which is shown in figure 3.

Transitions t_{qx} stands for defined transitions between valid states, while t_{ex} represents invalid transitions either between invalid state and valid state or between valid states.

ISBN: 978-988-17012-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

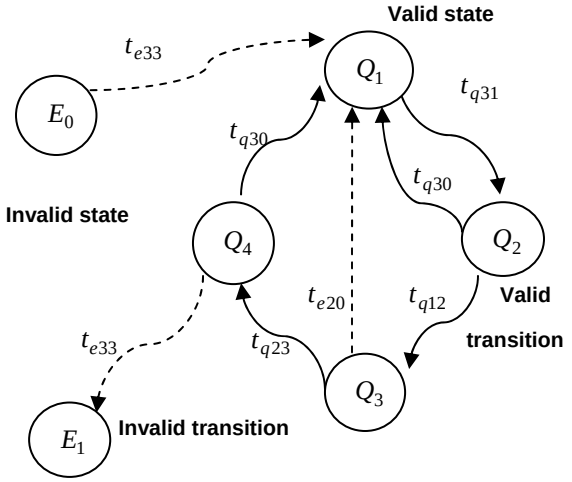

Figure 3: Valid and invalid state transitions in FSM representation of the system to be immunized

To perform error detection, the immunization cycle is divided into three stages:

1). Collecting data correspond to valid states and transitions: The goal of the data collection stage is to create a data set S that covers a complete or substantial percentage of all possible valid state transitions. It can be achieved through different approaches, including exhaustive random generation, and using a set of predefined test sequences. Gathered data is stored as the concatenation of user inputs, current state of the FSM and next state of the FSM. With an example of a 4-bit 0-9 binary coded decimal counter, gathered data will be stored in format of 10-bit strings, including 2 bits inputs (Enable, Reset), 4 bits previous state and 4 bits current state. The complete set of valid state transition have a total of $(2^2 \times 10) = 40$ self strings. This leaves $(2^{10} \times 40) = 984$ non-self strings to be detected.

2). Generating tolerance conditions: Generation of tolerance conditions requires the generation of the tolerance conditions to monitor for change in the self data. The random generation of tolerance conditions used in negative selection algorithm, which we introduced earlier, has a major shortfall, in terms of storage space, of the overlap in detectors. D'haeselee developed another method of improving the coverage of the string space through the greedy detector generating algorithm [7]. This seems to be a more efficient way to provide optimal coverage of the non-self search with minimum number of tolerance conditions. This method is realized by not generating detectors randomly, but instead extracting them according to the probabilities of non-match. Those tolerance conditions that match the most non-self strings are extracted before the others and are placed as far apart as possible. This method has the benefits of either reducing the probability of failing to detect a non-self string for a given number of tolerance conditions or reducing the number of tolerance conditions for a fixed failure probability.

3). Detecting operational fault and reconfiguration: The hardware immune system is incorporated into the system that is being protected, acting as a "wrapper" to the FSM. Figure 4 shows the simplified architecture. Under normal operation, only self strings are present, and the hardware immune system does nothing but monitoring. Once a fault creates a non-self state, the state recognition component will recognize a non-self string and pass the string to tolerance conditions component. A positive result will be generated if *c* contiguous bits match being found between any tolerance condition and the search string. The hardware immune system possesses several analogies to the natural immune system:

– It is separate to the FSM, which means that additional techniques for improving reliability could be added with no extrinsic effect to the hardware immune system.

– It monitors the state of the FSM, and only intervenes as necessary. A 'wait' state will be injected and the presence of a fault will be flagged.

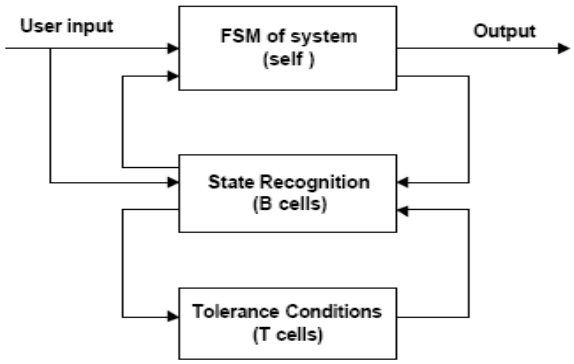

Figure 4: The hardware immune system attached to the FSM representation of system to be immunized. Adopted from [4]

2. EMBRYONICS

It is well known that there are around 60 trillion cells in a human being. In each of these 60 trillion cells, there is a stripe consisting of 2 billion characters called genome. It contains the information needed for both the construction and the operation of the organism. In this way, the organism can be viewed as a distributed system with 60 trillion cells operating in parallel. This system is highly fault tolerant in general.

Embryonics is a new fault tolerant strategy for cellular arrays by employing a similar scheme to the embryonic development of living being [8-11]. Through incorporating three important features in the cellular development of living being with the cellular structure in nature, improved fault tolerance can be achieved. These features are cellular division, cellular differentiation, and multicellular organization.

At the beginning of the embryonic development of a biological multicellular organism, a new individual is formed out of a single cell (the fertilized egg). Some time after the conception, the egg divides itself by a mechanism called mitosis, i.e. cellular division. Cellular division results in two cells with identical genetic material (DNA). Then the generated cells continuously repeat cellular division, passing to every offspring a complete copy of DNA. During this reproductive process, cells in certain position will take specific task through decoding the information stored in the DNA according to their positions within the embryo, i.e. cellular differentiation. Cellular differentiation results in the formation of different tissues, organs, and limbs that are essential for an organism to survive. At the same time, cells are arranged in such a way to achieve an optimal behavior for

ISBN: 978-988-17012-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

the organism, i.e. multicellular organization [2].

Inspired by the nature of the development of the multicellular development, embryonic arrays can be constructed in such way that the intersection of a row and a column defines a cell and at the same time each cell has an identical physical structure, i.e. an identical network of connections (wires) and an identical set of operations (combinational and sequential logical operators) [9]. Figure 5 shows an embryonic array and the architecture of a cell. The embryonic arrays are connected to its neighbors in the way of North-East-West-South (NEWS).

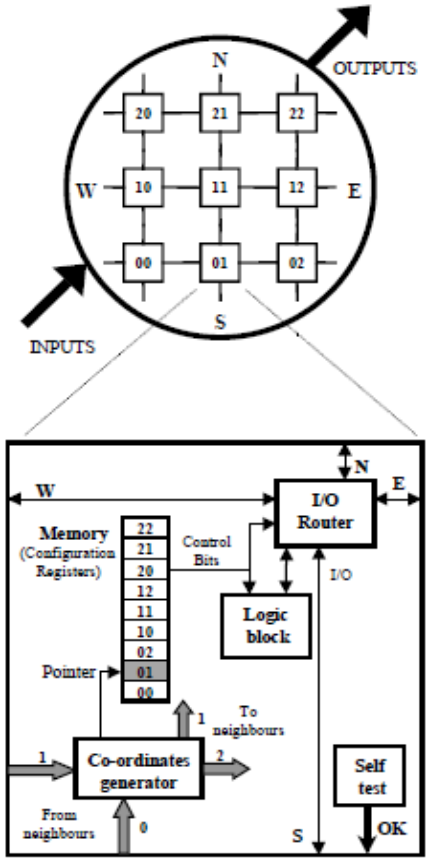

Figure 5: Architecture of Embryonic Cell. Adapted from [2]

Analogue to the genome in the biological multicellular organism [2], configuration registers are used to store the information need to specific the function of each embryonic cell. Strictly speaking, every embryonic cell should contain a full set of configuration registers. Before the process of cellular division, a full set of reconfiguration registers will be stored in the (0,0) cell. When the cellular division starts, information stored in the reconfiguration registers is send to the north and east neighbors cells. At the same time, the co-ordinations generator in each cell calculates their coordinates. During the cellular differentiation, the position, which depends on the coordinate of each cell define its task through generating a pointer to the reconfiguration registers. Combination of embryonic cells can take any desirable task constructed in this way. However, it is very difficult, if not impossible, to implement such a scheme in hardware for a large system. Therefore, a more practical approach incorporating the beneficial structure of biological system is to store only the reconfiguration register of an embryonic cell and those of its close neighbors.

Three features can be concluded for the embryonic cellular structured constructed above [14-15]: simplicity, vast parallelism, and locality.

– Simplicity: All the cells are identical except the stored configuration information.
– Vast parallelism: Cellular system can be implemented with millions of cells.
– Locality: A cell communicates only with four or less neighbors; therefore, the connection only carries small amount information.

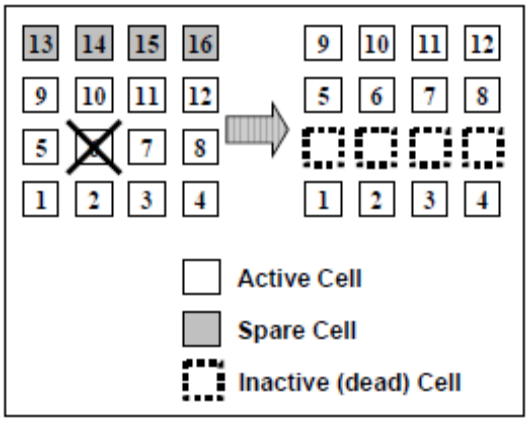

Figure 6: Row-elimination fault-tolerance strategy. Adapted from [2]

2.1 Fault-tolerance Strategy

In the systems comprised of bio-inspired cellular arrays, the fault-tolerance is improved by employing a distributing scheme, i.e. every cell stores not only information about its own function but also functions of its nearest neighbor. Therefore, once a failure of a cell is detected, the reconfigure signal will take control of the system and transmit the information of the failed cell to its neighbor cell, assuming that only one cell failure can occur at a time. With this fault-tolerant scheme, two approaches can be employed to achieve fault-tolerance while still making the cell simple enough to have high reliability: row-elimination and cell-elimination strategy [2, 12-13].

In the row-elimination strategy, one row of cells will be force to be transparent (inactive) once a cell failure is detected in that row, as show in the Figure 6.

In the row-elimination fault-tolerance strategy, the hardware for distributed diagnosis and fast reconfigure will be very simple to implement. However, it will be costly since the whole row containing fault cell is forced to be inactive, therefore requiring large amount of spares to achieve proper fault-tolerance capability for arrays with many cells in a row.

Cell-elimination fault-tolerance strategy makes efficient usage of spare cells with the cost of more complex hardware to realize the distributed diagnosis and fast reconfigure. In the cell-elimination fault-tolerance strategy, only the fault cell is inactive once it is found to be faulty. This scheme is illustrated in Figure 7. If all the spare cells in one row has been used to mask the faults in the same row, additional fault

ISBN: 978-988-17012-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

occurring in that row will activate a row-elimination fault-tolerance scheme.

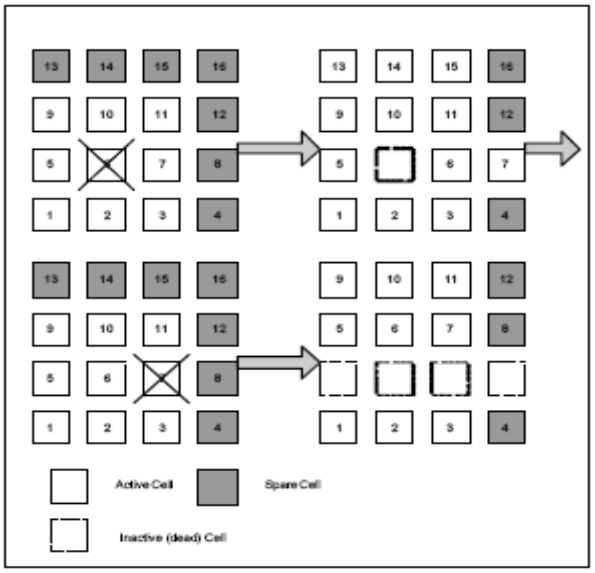

Figure 7: cell-elimination fault-tolerance strategy. Adapted from [2]

2.2 Reliability Analysis of the two fault-tolerance strategy.

In the discussion below, we assume that all the cells in the system are identical, i.e. they have the same reliability behavior. For simplicity, constant failure rate ? is assumed for each cell, i.e., the probability of a cell that still functioning at time t is:

$$p(t) = e^{-\lambda t} \quad (3)$$

In the following analysis, a system with mxn cells will be considered as operational if an rxk sub-array functions properly, which is illustrated in figure 8.

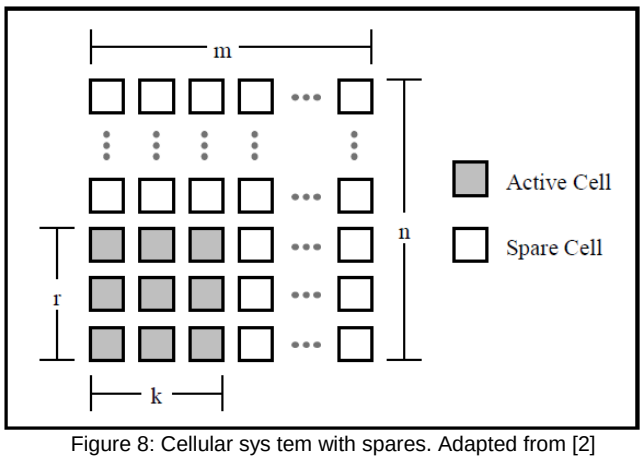

Figure 8: Cellular sys tem with spares. Adapted from [2]

2.2.1 The k-out-of-m reliability model:

Assume a system with m units is operational if any k units remain functional. Designate p as the probability of a unit that functions properly, we can write the probability of an operational system with k functioning units as:

$$P(k, m, p) = \binom{m}{k} p^k (1-p)^{m-k} \quad (4)$$

Therefore, the reliability of this system can be obtained as:

$$R_{sys} = \sum_{i=k}^{m} P(i, m, p) = \sum_{i=k}^{m} \binom{m}{i} p^i (1-p)^{m-i} \quad (5)$$

Assume each unit also has a constant failure rate ?, and substitute (1) into (3):

$$R_{sys}(t) = \sum_{i=k}^{m} \binom{m}{i} e^{-i\lambda t} (1-e^{-\lambda t})^{m-i} \quad (6)$$

It is well know the mean time to failure (MTTF) of a system is defined as:

$$MTTF = \int_0^{\infty} t R_{sys}(t) dt \quad (7)$$

The mean time to failure for this system will be obtained by substituting (4) into (5) and solving the integral as (6)

$$MTTF = \int_0^{\infty} \sum_{i=k}^{m} \binom{m}{i} e^{-i\lambda t} (1-e^{-\lambda t})^{m-i} dt = \frac{1}{\lambda} \sum_{i=k}^{m} \frac{1}{i} \quad (8)$$

2.2.2 Reliability model for row-elimination strategy

In the row-elimination fault-tolerance strategy, row with faulty cells will be inactive after reconfiguration. With this scheme, the system with m rows can employ the k out of m model if the reliability of each row is known.

Each row can be model as cells connected in series. With the constant rate assumption, the reliability of a row with m cells can be written as:

$$R_r = \prod_{i=1}^{m} p_i(t) = \prod_{i=1}^{m} e^{-\lambda t} = e^{-m\lambda t} \quad (9)$$

Therefore, the reliability of this system can be obtained by employing r-out-of-n scheme for this system and substituting (7) into (4):

$$R_{sysr}(t) = \sum_{i=k}^{n} \binom{n}{i} e^{-im\lambda t} (1-e^{-m\lambda t})^{n-i} \quad (10)$$

Similarly, the mean time to failure for the system with cell-elimination fault-tolerance is:

$$MTTF = \int_0^{\infty} t R_{sysr}(t) dt = \frac{1}{m\lambda} \sum_{i=r}^{n} \frac{1}{i} \quad (11)$$

2.2.3 Reliability model for the cell-elimination strategy

In the cell-elimination fault -tolerance strategy, two steps are involved: 1) the fault cells will be masked by the spare in the same row if a spare is still available in that row and 2) row-elimination scheme is activated due to the absence of spare in the row where fault occurs. Since at least k cells in a row are required to function properly before row-elimination is activated, the reliability of a row can be written as:

$$R_{rc} = \sum_{i=k}^{m} \binom{m}{i} e^{-i\lambda t} (1-e^{-\lambda t})^{m-i} \quad (12)$$

Similarly, at least r rows are required to function properly if a system remains operational, therefore, the reliability of

system can be obtain by employing the r out of n scheme as:

$$R_{sysc} = \sum_{j=r}^{n} R_{rc}(t)^{j}(1-R_{rc}(t))^{n-j} \qquad (13)$$

2.2.4 Comparison of row- and cell- elimination faulttolerance strategy:

Assume the constant failure rate is same for rowelimination and cell-elimination strategies and is 10^{-6}/hour. In [8] and [9], the authors compared these two strategies with two systems using 75 out of 100 scheme and a working size of 75×25. The size of the system for rowelimination is 100×25 while that of the system for cellelimination is 100×50. The results showed that cellelimination has a better reliability than row-elimination with the above assumption. It may be not obvious since the systems have different size, therefore different spare cells. However, the calculation of two systems with the same working size of 75×50 and the same number of spare cells will give the same result with the 75 out of 100 scheme. The size of the row-elimination system is 200×50 while that of the cell-elimination system is 100×100. The comparison is shown in Figure 9.

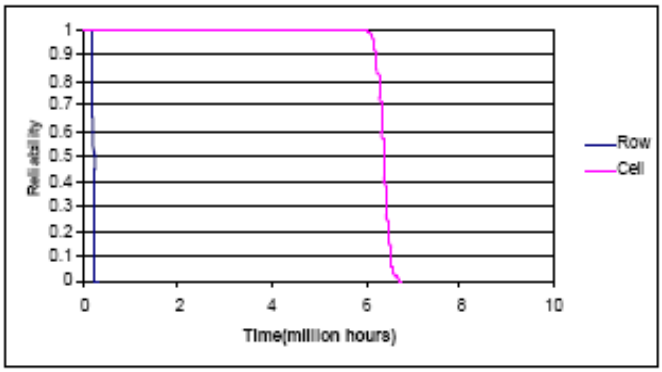

Figure 9: Reliability comparisons between row- and cellelimination.

As discussed earlier, the cell-elimination strategy has more complex hardware implementation than rowelimination strategy, therefore having a larger failure rate than the row-elimination strategy and making it a little undesirable.

3. CONCLUSION

Electronic systems are designed to provide stable, reliable, and long-term services. As the cost of design and implement extra fault tolerant components using conventional techniques becoming extraordinarily overwhelming, novel bio-inspired techniques are believed to be an alternative way.

This paper has presented and compared currently available bio-inspired techniques in three major domains: Epigenesis , ontogeny, and phylogeny. The ideas behind these schemes come from the self/nonself differentiation, development and self-reproduction, and evolution among generations of species, respectively. The three domains have been developed largely along their own individual paths, although future combinations between different domains are currently being worked on.

REFERENCES

1. M.Sipper, E.Sanchez, D.Mange, M.Tomassini, A.Perez-Uribe, and A.Stauffer, “A phylogenetic, ontogenetic and epigenetic view of bio-inspired hardware system,” *IEEE Trans. Evol.Comput.*, vol.1, pp.83-97, Apr.1997
2. C.Ortega and A.C.Tyrrell, “Reliability Analysis in Self-Repairing Embryonic Systems”, in *Proceeding of 2nd NASA/DoD Workshop on Evolvable Hardware*, 1999, pp.120 -128
3. D.W.Bradley, C.Ortega-Sanchez, and A.M.Tyrrell. “Embryonics + Immunotronics: A Bio-Inspired Approach to Fault Tolerance”. in *Proceeding of 2nd NASA/DoD Workshop on Evolvable Hardware*, July 2000.
4. D.W.Bradley and A.M.Tyrrell, “Immunotronics –Novel Finite-State-Machine Architectures With Built-In Self-Test Using Self Nonself Differentiation”, *IEEE Trans. Evol. Comput.*, vol.6, No.3, pp.227-238, Jun. 2002
5. D.W.Bradley and A.M.Tyrrell, “The architecture for a Hardware Immune System”, in *Proceedings of the 3rNASA/DoD Workshop on Evolvable Hardware*, 2001. pp.193-200
6. S.Forrest, A.S.Perelson, L.Allen, and R.Cherukuri, “Selfnonself discrimination in a computer,” in *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1994, pp.202-212.
7. P.D’haeseleer, S.Forrset, and P.Helman, “An immunological approach to change detection: Algorithms, analysis and implications,” in *Proceeding of the 1996 IEEE Symposium on Computer Security and Privacy*. Los Alamitos, CA:IEEE Comput. Soc. Press, 1996, pp.110-119
8. D. Mange, A. Stauffer, “Introduction to Embryonics: Towards New Self-repairing and Self-reproducing Hardware Based on Biological-like Properties”, *Artificial Life and Virtual Reallty, John Wiley*, 1994.
9. D. Manger, M. Goeke, D. Madon, A. Stauffer and G. Tempesti. “ Embryonics: A New Family of Coarse-Grained Field-Programmable Gate Array with Self-Repair and Self- Reproducing Properties,” *Circuits and Systems*, ISCAS '96.,vol.4 , 1996,pp. 25 -28
10. D. Mange, M. Sipper, A. Stauffer and G. Tempesti, “Toward Robust Integrated Circuits: The Embryonics Approach”, *Proceedings of the IEEE*, vol. 88, no. 4, 2000, pp. 516-541
11. D. Mange, E. Sanchez, A. Stauffer, G. Tempesti, P. Marchal, and C. Piguet, “Embryonics: A New Methodology for Designing Field-Programmable Gate Arrays with Self-Repair and Self-Replicating Properties”, *IEEE transactions on very large scale integration (VLSI) system*, vol. 6, no.3, 1998, pp.387-399
12. C.Ortega and A. Tyrrell, “Reliability Analysis of Self- Repairing Bio-inspired Cellular Hardware”, *Evolutionary Hardware Systems, IEE Half-day Colloquium on 1999*, pp. 2/1 -2/5
13. C. Ortega-Sanchez, A. Tyrrell, D. Mange, A. Stauffer, and G. Tempesti, Reliability analysis of a self-repairing embryonic machine”, *Euromicro Conference, 2000. Proceedings of the 26th* , Volume1 , 2000, pp.356 -361
14. L. Sekanina and V. Drábek, “Relation between Fault Tolerance and Reconfiguration in Cellular Systems,” *On-Line Testing Workshop. Proceedings. 6th IEEE International* , 2000, pp.25 -30
15. M. Sipper. “The Emergence of Cellular Computing”, *IEEE Computer*, 32(7), 1999, pp.18-26

ISBN: 978-988-17012-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)