The Architecture for a Hardware Immune System

D.W. Bradley and A. M. Tyrrell
Department of Electronics, University of York
Heslington, York, England
www.bioinspired.com

Abstract

Since the advent of fault tolerance in the 1960s, numerous techniques have been developed to increase the reliability of safety critical and space borne missions. In the last decade novel approaches to this field have sought inspiration from nature in the form of evolutionary and developmental forms of fault tolerance. In nature an additional inspiration axis exists in the form of learning. The body's own immune system uses a form of learning to maintain reliable operation in the body even in the presence of invaders. This has only recently been applied as a computational technique in the form of artificial immune systems (AIS). This paper demonstrates a new application of AIS with an immunologically inspired approach to fault tolerance. It is shown a finite state machine can be provided with a hardware immune system to provide a novel form of fault detection giving the ability to detect every faulty state during a normal operating cycle. We call this immunotronics.

1 Introduction

Nature demonstrates radically different approaches to complex problem solving that are now being used within computing and electronic fields to improve upon many of the classical techniques. One such area - reliable system design has experimented with both evolutionary methods through evolvable hardware [1] and the multi-cellular development in embryonics [2] [3]. Many different species possess defence mechanisms that can be referred to as an immune system. The zoologist Elie Metchnikoff first observed the presence of an immune system in starfish, with cells covering and engulfing a rose thorn that had pierced the creature [4]. He again observed the effects after much research with the Daphnia parasite [5]. Plants, fish, insects have also developed immune systems. Vertebrates have evolved a highly complex, multi-layered defense mechanism in the form of the immune system to provide protection from potentially hazardous external influences such as bacterial and viral infections (antigens). The process of antigen detection and self tolerance is essentially one of self/non-self differentiation performed with an almost limitless accuracy. The similarities between the requirements of fault tolerance and the operation of the body's defence mechanisms have highlighted the relevance of the immune system as a conceptual model for the design of future reliable systems [6]. This paper demonstrates one such approach.

Section 2 introduces the human immune system, discussing feature that have inspired the development of a hardware immune system (HIS). Section 3 reviews the field of artificial immune systems and demonstrates some of the many applications that this new area is already finding uses in. Section 4 introduces the hardware immune system by developing analogies between the immune system and hardware fault tolerance into a framework for a hardware immune system for finite state machine immunisation. Section 5 presents the results of a sample counter and its ability to detect the presence of faults. This is followed in section 6 with an analysis discussing the future of the work and improved fault detection capabilities. The paper is concluded in section 7.

2 The reliable human body

The body relies on the human immune system as a multi-layered defence mechanism. It is capable of preventing the onset of infection from approximately 10^{16} different foreign molecules [7]. Detection is implemented by several layers, each differing in complexity and protection method [8]. Table 1 compares the layers of biological reliability to those in hardware fault tolerance.

The acquired immune system, or more specifically the area of humoral immunity is the major source of inspiration for this work as the detection methods, as will be shown, are attractive for the task of hardware fault detection. In its simplest form, the underlying process is one of self/non-self differentiation, i.e. to determine what is a cell of the body (a

Defence mechanism	Human immune system	Hardware protection
Atomic barrier (physical)	Skin (mechanical) Mucous membranes (trap foreign organisms)	Hardware enclosure (physical/EM protection)
Physiological	Temperature (inhibit growth of pathogens) Acidity (destroy ingested microorganisms)	Environmental settings (temperature control)
Innate immunity	Phagocytes (macrophages) (Kill and digest foreign cells)	N-modular redundancy [9] Embryonics [10]
Acquired immunity	Humoral immunity (bacterial infections) Cell mediated immunity (viral infections)	Immunotronics [11]

Table 1. Layers of protection in the human body and hardware

valid state in the hardware) and what is not (an invalid state in the hardware). Humoral immunity, also known as antibody mediated immunity, protects the body from bacterial infections using B cells to generate antibodies and helper T cells to activate the production of antibodies. Centralised learning occurs in the thymus - the initial destination of immature helper T cells that have developed from stem cells in the bone marrow. The learning process ensures that an immune response can only be initiated against cells not belonging to the body. Self cells, or proteins circulate through the thymus and are exposed to the immature helper T cells. If any binding between receptors on a helper T cell and a self cell occur then the immature helper T cell is destroyed - a process known as programmed cell death. An estimated 1-5% of the immature helper T cells survive [8]. The process is essentially one of *negative selection*. The matured T cells are then distributed throughout the body into lymph nodes by the lymphatic system to take part in a distributed detection process (figure 1).

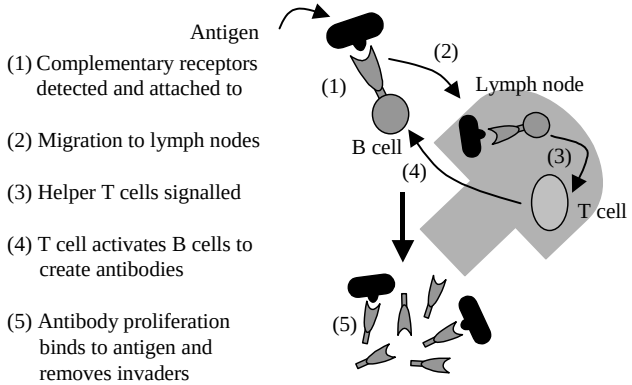

Figure 1. The humoral immune response

The humoral immunity cycle of figure 1 begins with the B cells which patrol the circulatory system and tissues for complementary receptors to bind to. B cells (and any cell they have become attached to) are picked up by the lymphatic system (which drains all the tissues of the body) and passed to lymph nodes. Protein fragments are presented to the helper T cells which determine if the collected cell should be attacked. If a match occurs with a certain affinity, estimated to be approximately 15 continuous receptors [12] the B cells are signalled to begin manufacture and proliferation of antibodies. The invading antigen is then destroyed.

3 Artificial immune systems

Advancements in our understanding of the fundamental elements of the human immune system over the last few decades has given rise to the field of artificial immune systems and immunological computation [13]. Immunological approaches are currently used for computer security [14], virus protection [15] [16], anomaly detection [17], process monitoring [18], robot control [19] and software fault tolerance [20]. Computer security, virus protection and anomaly detection applications have developed from the negative selection algorithm developed by Forrest and Perelson [15] from theoretical analyses of the matching and binding properties of the immune system [21]. Figure 2, adopted from [22] demonstrates the algorithm.

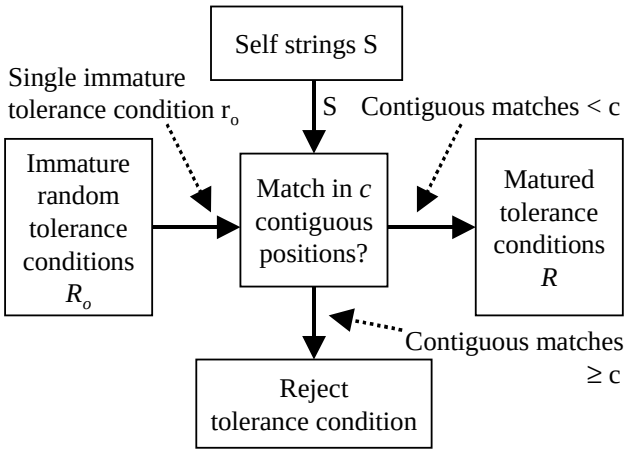

Figure 2. The negative selection algorithm

The negative selection algorithm works by selecting a set of strings R (the matured helper T cells), of length l, from a randomly generated original set of data R_o (immature T cells), that 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 strings is given equation 1.

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

where $m^{-c} \ll 1$ and m is the number of alphabet symbols (2 for a binary FSM). Any strings that match in at least c contiguous positions are deleted. In order to create R, referred to as the set of *tolerance conditions*, with a sufficiently low failure probability, the number of valid self strings N_s must be a lot less than the number of invalid strings N_e. A detailed analysis of the matching probabilities specific to this algorithm can be found in [15]. Using a probabilistic detection method means that a trade off can be made between the number of tolerance conditions, N_R and therefore memory requirements and the probability of failing to detect a fault, P_f (detection probability, P_M is therefore $1-P_f$). For theoretical predictions this is defined by equation 2.

$$P_{f_t} = (1-P_M)^{N_R} \quad (2)$$

During the operational stage of the process data is extracted from the protected system and compared against the list of stored tolerance conditions R. If any match occurs then the system is alerted to a potential anomaly/fault. The operational stage is computationally much simpler, making implementation for a hardware system viable.

4 The hardware immune system

From a classical perspective, three stages need to be addressed in order to create a fault tolerant system:

- The *detection* of an error or output deviating from the norm. For the hardware immune system this dictates the differentiation between self and non-self.
- The *minimisation* or *eradication* of the consequential effects of the fault.
- *Activation* of a suitable recovery procedure.

This paper concentrates on the first of the three stages for the hardware immune system. In mapping the immune system over to a hardware representation, the analogies in table 2 are used.

The hardware immune system uses a finite state machine (FSM) representation of the system to be immunised. In principle, any hardware system can be represented by an individual or interconnected set of FSMs, furthermore this allows both self and nonself to be explicitly defined, as shown in figure 3.

Immune system	**Hardware fault tolerance**
Self	Normal operation
Nonself (antigen)	Faulty operation
Antibody (B cell)	System state/tolerance condition comparison
Memory T cells	Set of stored tolerance conditions
Learning during gestation	Generation of tolerance conditions
Inactivation of antigen	Return to normal operation
Lifetime of organism	Operational lifetime of the hardware

Table 2. Immune system to hardware fault tolerance mapping

Under normal conditions (self) only transitions t_{qx} can occur. The advent of a fault causing an undefined transition is shown by t_{ex}. Concentrating on the complete transition rather than specific states is ideal as it then permits an undefined transition between two individually valid states to be detected. Although the hardware is represented by a state machine any representation is feasible for immunisation, given the condition that it permits extraction of a fixed length representation of the state, output, or elements that are to be protected at any point in time.

To permit fault detection, the immunisation cycle is split into three stages:

- Collection of data that corresponds to self.
- Tolerance condition generation.
- Immune system configuration and operational fault detection.

Data gathering is performed using the hardware/software testbench of figure 4 using a Virtex XCV300 FPGA [23] as the hardware. This currently operates by randomly generating input sequences to the FSM until all the valid transitions, defined by the state diagram have been reached. The alternative, if possible, is to use a set of predefined test sequences designed to extract all data as quickly and efficiently as possible. The data gathered is stored as the concatenation of user inputs, current state of the FSM and next state of the FSM.

The next stage requires the generation of the tolerance conditions to monitor for change in the self data. Two ap-

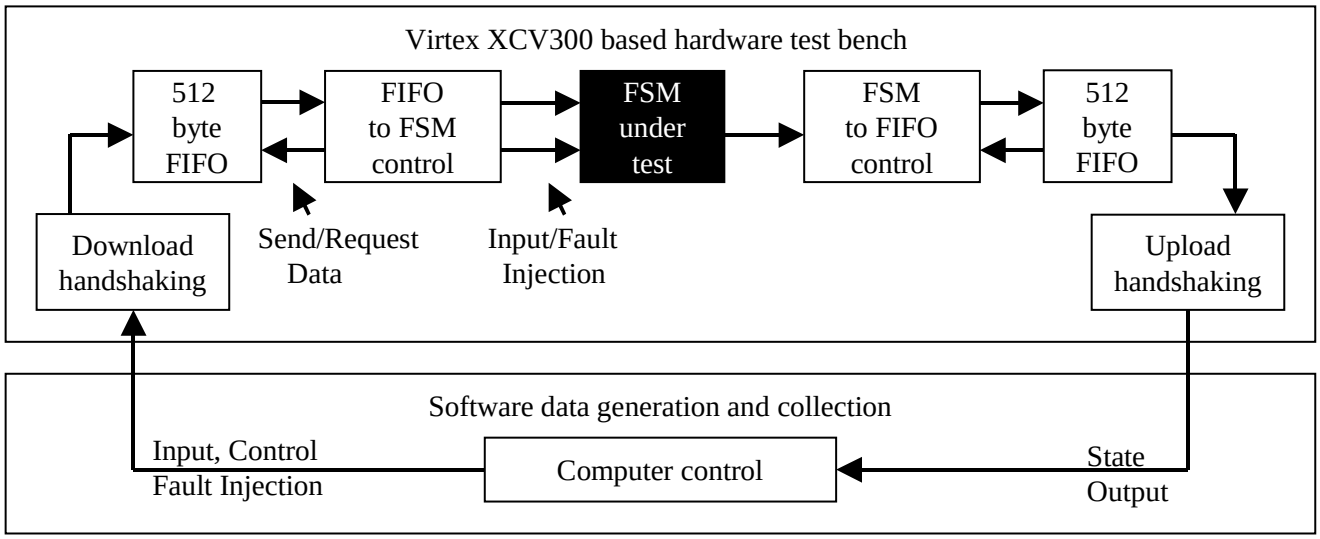

Figure 4. Hardware/software testbench for data generation and gathering

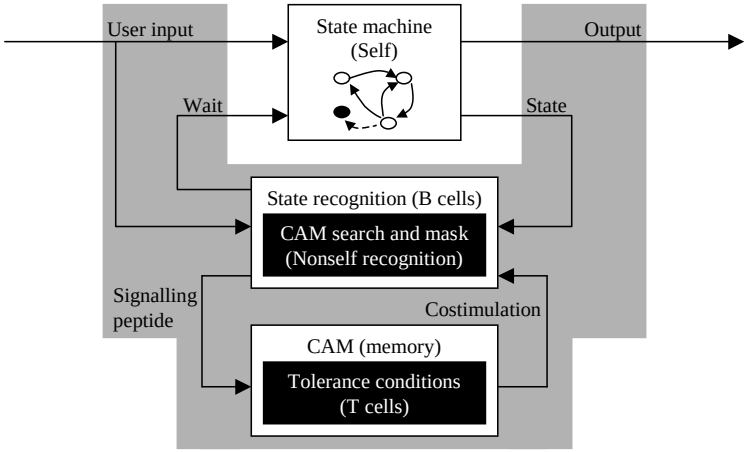

Figure 5. The hardware immune system attached to the finite state machine

proaches are shown, the first uses the random generation of tolerance conditions from an initial repertoire R_o previously discussed. This approach - the most immunologically representative of the two approaches has several weaknesses for use within a hardware environment that can be improved upon. The random generation of detectors can result in many tolerance conditions matching many of the same nonself strings which while this is perhaps beneficial redundancy wise (a fault causing a bit to flip state in one tolerance condition still leaves other tolerance conditions detecting the same faults) in very inefficient resource wise if the number of tolerance conditions is to be limited. Section 5 demonstrates this effect. D'haeseleer [24] developed the greedy detector generating algorithm to provide optimal coverage of the nonself search space with the minimum number of strings, or tolerance conditions in this case. The algorithm differs somewhat in operation to an immunological approach but produces tolerance conditions distributed as far apart as possible over the complete range of nonself strings, furthermore, generation of tolerance conditions is significantly quicker. Using the greedy detector algorithm requires the match length c to be defined. Those tolerance conditions matching the most nonself strings are 'extracted' first. Storage constraints can then be traded off against the matching probability. The variation in these parameters are shown for a 4-bit counter in the results in section 5.

With the tolerance conditions generated they must then be downloaded to the host hardware immune system (figure 5). The hardware immune system possesses several analogies to the natural immune system:

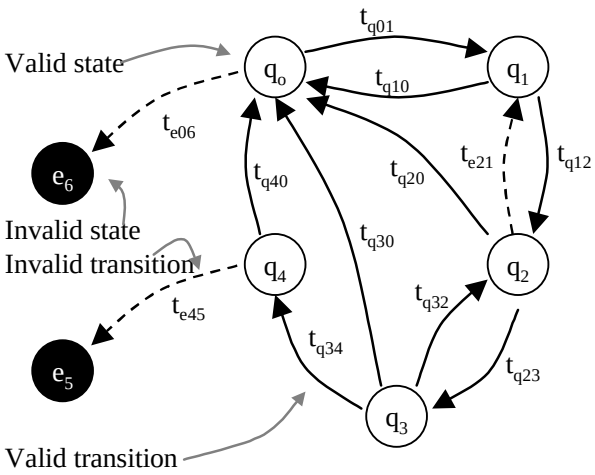

Figure 3. FSM hardware representation of the system to be immunised

- The hardware immune system is separate to the FSM, i.e. self that is being protected, just like the natural immune system. In hardware, this means that additional techniques for improving reliability can be included with no detrimental effect to the hardware immune system.

- The hardware immune system monitors the state of the FSM and only intervenes as necessary. At present this results in the injection of a ‘wait’ state and flagging the presence of a fault.

To permit total searching of the tolerance condition memory storage element in a single cycle prior to the next

cally derived theoretical probability of failure (for random tolerance condition generation) then the plot shows that for $c < 7$, $P_{fe}(Ex)$ does not perform as well as theoretical predictions, due to overlap in the nonself strings matched by the tolerance conditions. At $c = 4$ a 40% difference is observed as it is only possible to only generate a very small number of tolerance conditions that fail to match the self data. The greedy generator improves upon this significantly and actually performs better than theoretical predictions for $c > 5$. This is due to the greedy generator specifically selecting optimal values rather than completely randomly which the theoretical prediction relies upon.

If the number of tolerance conditions is extended from $10 \leq N_r \leq 100$ as shown in figure 7, the optimal match length can be determined for the required failure probability and storage requirements. For a requirement of 10 or 20 tolerance conditions $c = 5$ provides the lowest failure probability. For 30-60 $c = 6$ provides the lowest failure probability. For 70-100 $c = 7$ is best suited. If the plot of figure 7 is extended to include more tolerance conditions the optimal match length progresses from 8 through to 10.

6 Analysis

The paper has demonstrated the immunisation of only a small system, with the immunisation hardware appearing more complex that the system under analysis itself. Fault detection hardware is relatively simple - hardware to extract the data from the system under test, and a memory device with simple control logic. The example system is used to demonstrate that an immune inspired technique is viable and we are currently in the process of applying the hardware immune system to much larger systems.

The architecture permits a trade off between failure probability and storage requirements. The existing architecture permits 100% fault detection if the match length c equals the length of the self strings to be protected. This approach can be significantly improved if a variable match length approach is considered, as originally suggested in [15]. Before such a solution could be justified it would be necessary to determine if the reduction in number of tolerance conditions outweighed the increased complexity of the string matching hardware.

The work has assumed that it is possible to gather all the self data from the finite state machine to be immunised. In reality with a state machine of increased size and complexity this is likely not to be possible. The advantage with this technique over monitoring and searching for self is that if a complete repertoire of self strings is not obtained prior to the immunisation process, then the resulting set of tolerance conditions can still detect all faults. The side effect of this is that it can result in incorrectly interpreting the unknown self conditions as nonself.

The preceding sections have dealt with the detection of a fault by the hardware immune system. The next challenge is to determine the consequential action to be performed to limit the propagation of an error. A form of forward or backward error recovery could provide a solution or alternatively, a more novel solution would be to selectively correct the faulty state until a self condition is achieved again. This could be either completely autonomously or with initial human intervention to determine the correct self condition given a specific fault. One proposal is to implement a form of forward error recovery that first analyses the state of the system and corrects the output and next state logic if a previously encountered fault occurs again.

In nature, the process completes with destruction of the foreign invader. The phenomenal cellular redundancy in the body makes the death of a cell insignificant. In an hardware system the same does not generally apply, at least with a typical architecture. The deactivation of a complete subsystem in an n-modular redundant configuration for a single fault it effective, but not resource efficient. Unfortunately technology does not allow new components to be regrown on silicon (as yet) so replacing a defective item is not possible. However, implementation in an embryonic array does go some way to providing such a solution. The addition of a hardware immune system using similar techniques discussed has been considered for this [27].

7 Conclusion

This paper has demonstrated a immunologically inspired approach to hardware fault detection and discussed the architecture for a hardware immune system to detect faults in systems. The work has used a finite state machine representation of the system to be protected although there is no reason why this approach could not be applied to any number of techniques.

The hardware immune system, even in its current state permits 100% detection of faults if storage requirements to not impose any restrictions. The future of this work looks towards improved matching capabilities and immunisation of larger and more complex state machine based systems to assess the detection abilities of the hardware immune system.

Acknowledgments

This work has been supported by the Engineering and Physical Sciences Research Council, UK and Xilinx, Inc.

References

[1] A. Thomspon. Evolutionary Techniques for Fault Tolerance. In *Proceedings of the UKACC International*

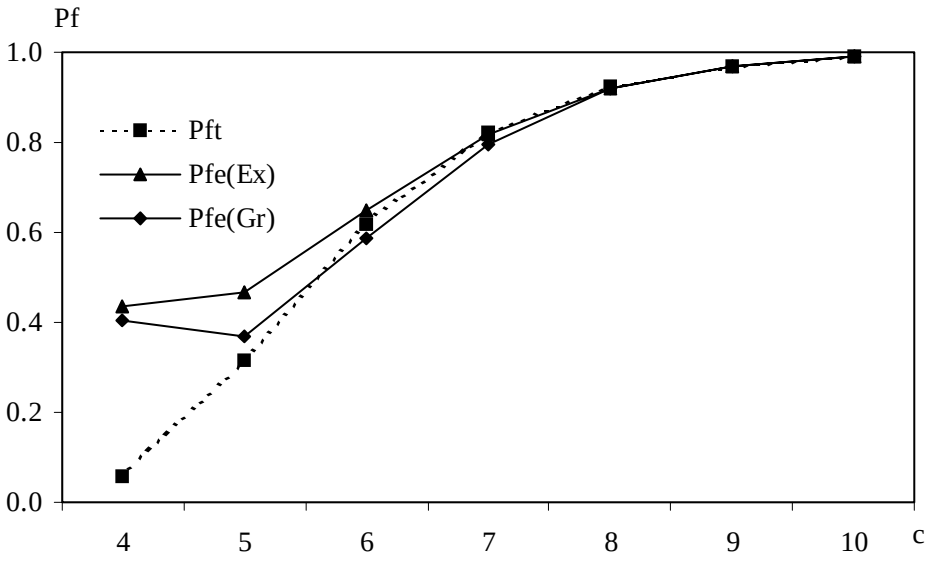

Figure 6. Probability of failure (Pf) against match length (c) for 10 tolerance conditions (Nr). The plot shows theoretical, exhaustive generator and greedy generator failure rates

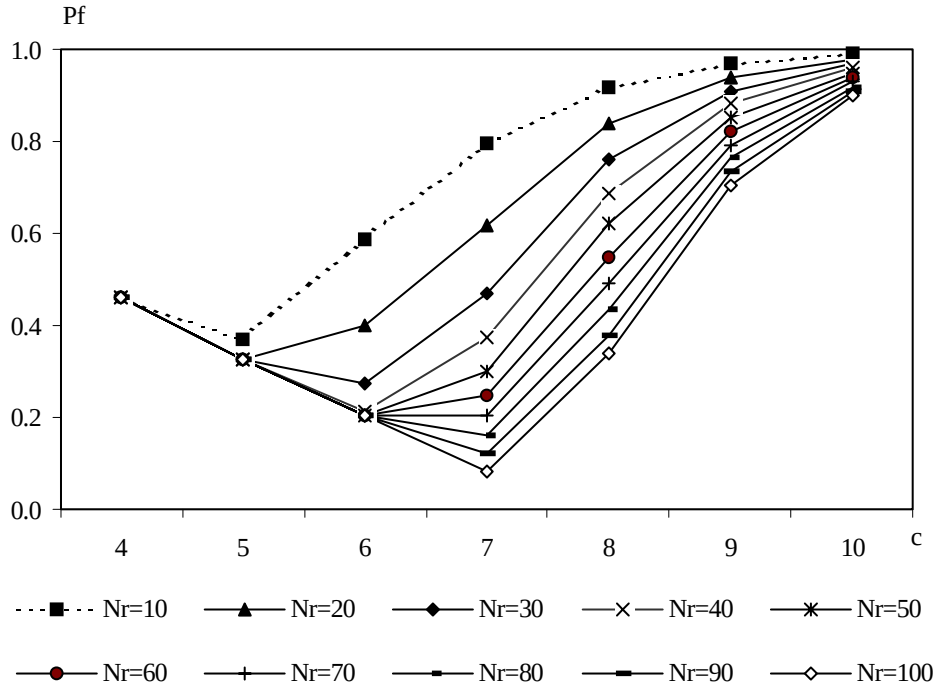

Figure 7. Probability of failure (Pf) against match length (c) for 10 - 100 tolerance conditions (Nr), for greedy generator failure rates

Conference on Control, pages 693–698. IEE Conference Publication No. 427, 1996.

[2] C. Ortega-Sánchez, D. Mange, S. Smith, and A. Tyrrell. Embryonics: A Bio-Inspired Cellular Architecture with Fault-Tolerant Properties. *Genetic Programming and Evolvable Machines*, 1(3):187–215, July 2000.

[3] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti. Toward Robust Integrated Circuits: The Embryonics Approach. *Proceedings of the IEEE*, 88(4):516–541, April 2000.

[4] G. Beck and G.S. Habicht. Immunity and the Invertebrates. *Scientific American*, pages 42–46, November 1996.

[5] I. Roitt. *Essential Immunology*. Blackwell Science, 9th edition, 1997.

[6] A. Avizienis. Towards Systematic Design of Fault-Tolerant Systems. *IEEE Computer*, 30(4):51–58, April 1997.

[7] J. Inman. The Antibody Combining Region: Speculations on the hypothesis of general multispecificity. In G. Bell, A. Perelson, and G. Pimbley, editors, *Theoretical Immunology*, pages 234–278. Marcel Dekker, 1978.

[8] J. Kuby. *Immunology*. W.H. Freeman and Company, 3rd edition, 1997.

[9] A. Avizienis. Fault-Tolerance: The Survival Attribute of Digital Systems. *Proceedings of the IEEE*, 66(10):1109–1125, October 1978.

[10] C. Ortega-Sánchez and Tyrrell A.M. MUXTREE Revisited : Embryonics as a Reconfiguration Strategy in Fault-Tolerant Processor Arrays. In M. Sipper, D. Mange, and A. Pérez-Uribe, editors, *Evolvable Systems : From Biology to Hardware*, volume 1478 of *Lecture Notes in Computer Science*. Springer-Verlag, September 1998.

[11] D.W. Bradley and A.M. Tyrrell. Immunotronics: Hardware Fault Tolerance Inspired by the Immune System. In J. Miller, A. Thompson, P. Thomson, and T.C. Fogarty, editors, *Third Internation Conference on Evolvable Systems (ICES2000)*, volume 1801 of *Lecture Notes in Computer Science*, pages 11–20. Springer-Verlag, April 2000.

[12] J.K. Percus, O.E. Percus, and A.S. Perelson. Predicting the size of the T-cell receptor and antibody combining region from consideration of efficient self-nonself discrimination. In *Proceedings of the National Academy of Science*, volume 90, pages 1691–1695. Oxford University Press, 1993.

[13] D. Dasgupta and N. Attoh-Okine. Immunity-Based Systems: A Survey. In *IEEE International Conference on Systems, Man and Cybernetics*, Orlando, 1997.

[14] S. Forrest, S.A. Hofmeyr, A. Somayaji, and T.A. Longstaff. A sense of self for Unix processes. In *Proceedings of 1996 IEEE Symposium on Computer Security and Privacy*, 1996.

[15] S. Forrest, A.S. Perelson, L. Allen, and R. Cherukuri. Self-Nonself Discrimination in a Computer. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pages 202–212, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[16] J.O. Kephart. A Biologically Inspired Immune System for Computers. In R.A. Brooks and P. Maes, editors, *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 130–139. MIT Press, 1994.

[17] D. Dasgupta and S. Forrest. An Anomaly Detection Algorithm Inspired by the Immune System. In D. Dasgupta, editor, *Artificial Immune Systems and Their Applications*, pages 262–277. Springer-Verlag, 1998.

[18] A. Ishiguro, Y. Watanabe, and Y. Uchikawa. Fault Diagnosis of Plant Systems using Immune Networks. In *Proceedings of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 34–42. IEEE, 1994.

[19] A. Ishiguro, T. Kondo, Y. Watanabe, and Y. Uchikawa. Immunoid: An Immunological Approach to Decentralized Behaviour Arbitration of Autonomous Mobile Robots. In H.M. Voight et al., editors, *Parallel Problem Solving from Nature IV*, volume 1141 of *Lecture Notes in Computer Science*, pages 666–675. Springer-Verlag, September 1996.

[20] S. Xanthakis, S. Karapoulios, R. Pajot, and A. Rozz. Immune System and Fault Tolerant Computing. In J.M. Alliot, editor, *Artificial Evolution*, volume 1063 of *Lecture Notes in Computer Science*, pages 181–197. Springer-Verlag, 1996.

[21] J.K. Percus, O.E. Percus, and A.S. Perelson. Probability of Self-Nonself Discrimination. In A.S. Perelson and G. Weisbuch, editors, *Theoretical and Experimental Insights into Immunology*, pages 63–70. Springer-Verlag, 1992.

[22] P. D'haeseleer. An Immunological Approach to Change Detection: Theoretical Results. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, County Kerry, Ireland, June 1996.

[23] Xilinx Inc. Virtex data sheet, 1999. http://www.xilinx.com/partinfo/virtex.pdf.

[24] P. D'haeseleer. Further Efficient Algorithms for Generating Antibody Strings. Technical Report CS95-3, Department of Computer Science, University of New Mexico, 1995.

[25] J.E. Hunt and D.E. Cooke. Learning using an artificial immune system. *Journal of Network and Computer Applications*, 19:189–212, 1996.

[26] C.J. Gibert and T.W. Routen. Associative Memory in an Immune-Based System. In *Proceedings of the 12th International Conference on Artificial Intelligence AAAI-94*, pages 852–857, 1994.

[27] D.W. Bradley, C. Ortega-Sánchez, and A.M. Tyrrell. Embryonics + Imunotronics: A Bio-Inspired Approach to Fault Tolerance. In *Proceedings of 2nd NASA/DoD Workshop on Evolvable Hardware*, July 2000.