TOWARD BIOLOGICALLY INSPIRED SELF HEALING DIGITAL EMBEDDED DEVICES: BIO-SYMPLE

Shawkat S. Khairullah, Tim Bakker, and Carl R. Elks
Department of Electrical and Computer Engineering, Virginia Commonwealth University (VCU)
601 West Main Street, Richmond, Virginia, USA
khairullahss@vcu.edu; bakkert@vcu.edu; crelks@vcu.edu

ABSTRACT

One of the more interesting concepts being postulated for next gen power plants, is the notion of highly resilient and survivable I&C systems. While resilience and survivability are "cross system attributes", resilience and robustness will rely on highly robust and fault resilient devices as a foundation. This paper presents a new self-healing digital I&C architecture inspired by the operation of two biological concepts: human immune system and embryonic stem cells. The purpose of this research is look at different bio-inspired paradigms for organizing system functions so that they become inherently resilient due to device organization and behaviors. Our current proposed approach (called BIO-SymPle) is organized in two levels: critical service layer, which is providing the intended service of critical application, and healing layer, that supervises the operation of all repair mechanisms. To facilitate coordination, organized behavior among cells we employ network-based topology structures which is comprised of four active cells, four pre-generated redundant cells, and one embryonic stem cell. Collectively, the ensemble represents the building block that executes the local functionality based on the expression for DNA genetic codes stored inside each cell. The four bio-operational active cells are protected by their neighboring spare cells that imitate how white blood cells are distributed around the antigens in their locations. Ultimately, the proposed multi-layer architecture is able to cover a significant number of coincident faults and we believe architectures like BIO-SymPLe can positively impact the next generation digital I&C system.

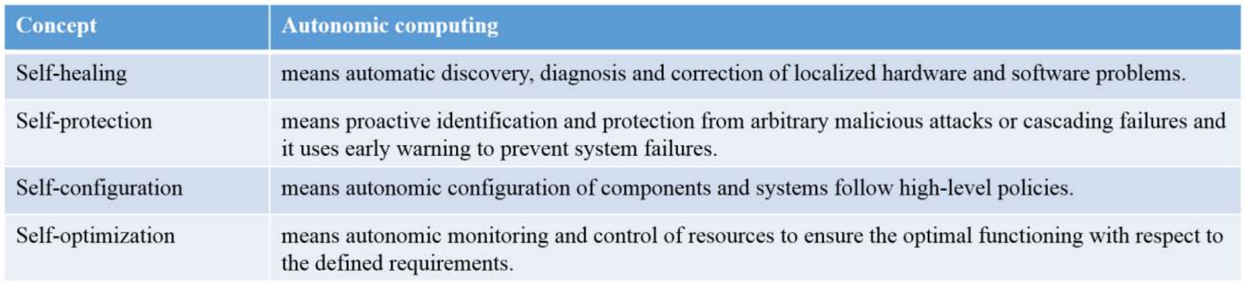
. *Key Words*: safety-critical, embryonic stem cells, human immune system, white blood cells, and antigens.

1 INTRODUCTION

Computing and communication capabilities are rapidly being embedded in all types of objects and structures in the physical environment. Applications with enormous societal impact and economic benefit are often called *critical infrastructure systems*, such as, power and water, industrial systems, transportation systems, medical devices, security systems, building automation, emergency management, and many other systems vital to our well-being[1][2][3]. As we bridge computing, communication and human machine interactions into the physical world of critical systems, we create a new class of systems – *Cyber Physical Systems*. Cyber-physical systems (CPS) are physical and engineered systems whose operations are monitored, coordinated, controlled and integrated by a computing and communication core. For Nuclear Power Plant (NPP) critical systems, the occurrence of unknown or unforeseen interactions due to failures, design flaws or oversights can lead to unacceptable system behaviors. When these malfunction or fail, the operation of corresponding systems in the real world can impair physical safety or trigger loss of life [1].

Currently, in the academic and industrial arenas, there is significant research and development activity devoted to multidisciplinary theories, leveraging new technologies to harness the promise of CPSs [1][3]. Of the many facets of current CPS research, *autonomic computing* is seen as a promising way to manage the complexity of CPS systems, while enhancing the resilience of these systems to a variety of threats, failures, and faults. Autonomic computing systems are systems that can manage themselves given high-level objectives from human actors or operators. Autonomic computing is not a single discipline, but a collection of disciplines organized around a concept of self-governance – much in the way biological systems have evolved. From the broadest stance, autonomic computing can be seen as four interwoven characteristics: *Self-configuration, Self-Optimization, Self-healing, and Self-protection*. Table 1 provides an overview of these features [15]

Table 1. Four aspects of self-management with autonomic computing



Concept	Autonomic computing
Self-healing	means automatic discovery, diagnosis and correction of localized hardware and software problems.
Self-protection	means proactive identification and protection from arbitrary malicious attacks or cascading failures and it uses early warning to prevent system failures.
Self-configuration	means autonomic configuration of components and systems follow high-level policies.
Self-optimization	means autonomic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements.

Of these four attributes, *self-healing and self-protection* capabilities are desirable requirements in much of the Instrumentation & Control (I&C) digital equipment in the US and International NPP. The attributes of self-healing and self-protection provide innate abilities to withstand disturbances, faults, failures and recovery quickly from these distributions to a guaranteed level of service as these critical systems interacts with the physical environment. While there are many strategic approaches to realizing self-healing systems reported in the literature [6][7][9], in this paper we are concerned with approaches inspired from two biological concepts: human immune system and embryonic stem cells.

In recent years, self-healing concepts based on biological physiology have received attention for the possibility of building systems that are more robust than traditional fault tolerance methods [7]. Self-healing models based on immune systems, embryonics, and dendritic cells, have been reported in the literature with varying degrees of success and adoption. However, many of these models are not fully developed for safety critical applications - that is, they lack provenance and verifiability of integrity attributes like real-time behavior, determinism, fault tolerance, and testability.

1.1 Biologically Inspired Autonomic Concepts

To understand the possibilities of biological inspired self-diagnostic and self-healing systems, and to set the context for the reader, we discuss briefly the main techniques used by most highly complex biological organisms to achieve self-healing and self-protection. Self-diagnostic and self-healing properties are intrinsic to biological organisms and these mechanisms are automatically activated by the body and without any external intervention. There are three fundamental self-healing techniques used in biological organisms to protect it from pathogens, antigens, and internal diseases, and these will be used as source of inspiration. These techniques are *cell cycle*, *stem cells*, and *immune system (innate and adaptive)*. Within each of these techniques, there are diverse mechanisms used by the organism to provide self-healing at the cellular level – collectively providing overall organism resilience [4].

The first technique called cell life cycle shown in Fig. 1(a), is considered as a self-diagnostic mechanism in cellular biology. This mechanism enables the cell to continuously monitor its internal state and the external environment. It tests whether the cell has correctly duplicated its DNA contents in the chromosomes and segregated them into two copies such that each daughter cell can receive a complete genome after the division process. This mechanism usually occurs in four phases and there is a checking point at the end of each phase to test integrity [4]. Second mechanism called stem cells shown in Fig. 1(b), includes two types of cells: adult stem cells and embryonic stem cells. The first type are ready either to be divided to produce more stem cells or to be differentiated into any terminally differentiated cell that has a specific function. However, the second type, are available to be continuously divided and differentiated into any type of cells. These cells can become any type of specific functional cell based on the gene expression that determines which gene is active or not on the DNA molecule [4]. The third mechanism called the immune system is often used by the human body to protect it against different types of pathogens and antigens like the viruses and bacteria. This system has a distributed defense and self-healing mechanisms which can be an inspiration for the design of a distributed fault-tolerant digital systems. Inside the human body, there are millions of the white blood cells called lymphocytes which are distributed throughout the body and can swim through the blood vessels through the extracellular array matrix going to the place of the infection. These cells can detect the infection using special types of cells called B-Cells and fix it or kill it using another type of cell called T-Cell. This system doesn't have any kind of centralized recovery mechanisms, and thus single point of failure that may be vulnerable for being damaged [4] [9].

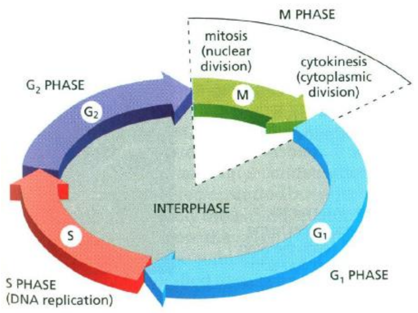

M PHASE
mitosis (nuclear division)
cytokinesis (cytoplasmic division)
G_2 PHASE
G_2
M
INTERPHASE
S
G_1
S PHASE (DNA replication)
G_1 PHASE

Figure 1(a). Biological cell life cycle

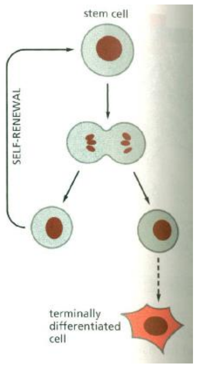


Figure 1(b). Stem cell differentiation

1.2 BIO-SymPLe Architectural Concepts

Self-healing systems are often comprised of interactive collections of *autonomic elements*—individual programmable elements (PE's) that contain resources and deliver services to the external world. Programmable elements manage their internal behavior and their relationships with other elements in accordance with overall system safety policy. Each element is responsible for managing its own internal behavior and for managing its interactions with an environment that consists largely of signals and messages from other elements and the external world. An element's internal behavior and its relationships with other elements is driven by safety and functional goals that its designer has embedded in it, by other elements that have authority over it, or by agreement to peer elements with its explicit consent. The element may require assistance from other elements to achieve its computational or self-healing goals. If so, it will be responsible for obtaining necessary resources from other elements and for dealing with real-time failures, such as the failure of a required resource to be acquired.

The organization and function of the proposed self-healing architecture is built on a model of self-healing that is a hybrid of biological models and dependability models. Nonetheless, attributes of concern for embedded safety criticality remain unchanged in the context of biological self-healing systems. In this paper, a new self-healing digital I&C system called Bio-SymPLe architecture is proposed and this system has its foundations in PLC architecture to design reactive real time I&C digital systems that can be used in safety critical embedded devices.

2 RELATED WORK

Different self-healing digital systems have been designed in the literature to mask, detect, and recover from the effects of faults. However, these techniques have led to inefficient designs and low-level of fault coverage due to the hardware redundancy for the entire systems and the limited number of fault tolerant techniques [5][13]. In [6], a basic cell architecture for a self-repairing FPGA has been proposed by Mange et al. The design of their cell was based on the concept of time redundancy technique as a fault tolerant mechanism to detect the fault occurrence inside the cell within the array at decentralized level. The internal architecture of that cell includes two types of multiplexers and two flip flops whose outputs are compared to detect the fault occurrence. Ortega & Tyrrell in [7] and [8] have proposed a different approach in embryonics by building a two-level hierarchical architecture consisting of a cellular level and organism level. Two modes of operation have been proposed to control the operation of the organism in a fault tolerant manner. The first mode is called operation mode in which the cells of the array are working together to perform the designed function of the application. The second mode is called reconfiguration mode in which, once a fault is detected in any cell of the array, the cells on the right hand of this cell recalculate their own addresses and download their new configuration genetic codes to heal the array against the failure. The embryonic cell architecture that has been proposed at the university of York is developed by (Zhang, Dragffy, & Pipe, 2006) at the university of west of England in [10]. They have increased the level of fault coverage by adding more levels of fault tolerance to detect the transient faults in the configuration memory besides the detection of permanent faults in the function unit. Their embryonic architecture also works at two levels: cellular and organism. The internal architecture of each cell includes eight functional units: function unit, I/O router, DNA segment memory. DNA repair unit, diagnostic logic, co-ordinate generator, reconfiguration unit, and a core register. In [9], the authors have designed a self-healing architecture inspired by the biological process of the human immune system. Their design approach is using several distributed spare cells embedded among the functional cells in the array to heal the system at a decentralized level against the faults. Three types of cells are included in the design: functional cells, spare cells, and router cells. In [12], Wang et al. have designed a different approach to realize the embryonic cell. Their design is based on using the Look Up Table (LUT) as a building block for the functional unit and the column elimination is used as a self-healing mechanism by making all the cells within the same column of the faulty cell as transparent cells that perform none function. Ultimately, a novel self-repairing hardware architecture inspired by paralogous gene regulatory circuits was designed as a fault tolerant feature in biological living organisms to achieve fast fault recovery with an efficient use of hardware resources [14].

3 OVERVIEW OF THE BIO-SYMPLE ARCHITECTURE

Our Bio-SymPLe architecture is based in part on the way biological organisms achieve resiliency: separation of concerns (transient faults, permanent faults, mutations, CCFs). We partition our architecture into two principle divisions; (1) the *Critical Service layer*, which is responsible for providing the intended functionality of the critical application, and (2) the *Healing layer*, which is responsible for monitoring the behavior of the functions and triggering the required repair mechanisms to heal the critical layer. The critical layer, which includes two embedded sub-layers: functional cell and tissue (see Fig. 2), contains four bio operational B cells, four pre-generated redundant T cells, four transfer inputs units, and four 2to1 multiplexers. However, the healing layer consists of all other components outside the critical layer and these components are four local healing layers, one global healing layer, select PF signals unit, an input transfer unit, and one spare stem cell.

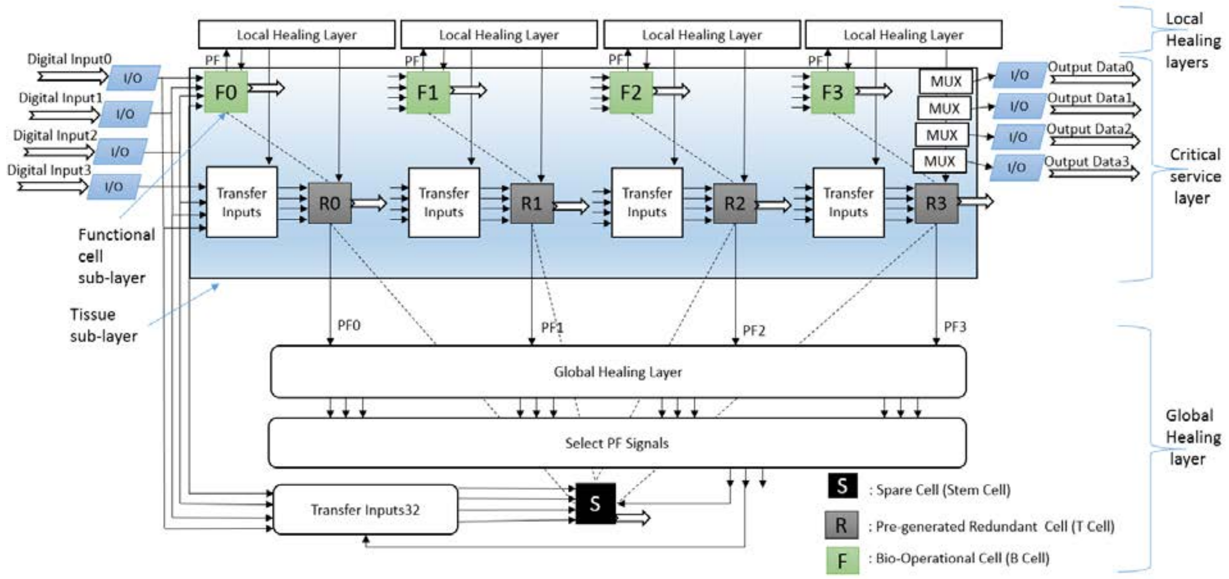


Figure 2. Bio SymPLe Architecture Concept

The four active B cells are assumed to execute four different functions on digital input data based on both the addresses of the cells in the system and the activated genes stored inside the configuration memories. However, the correct execution of the function in each cell is being monitored continuously by its neighboring local healing layer and once a fault is detected and determined to be transient, it will be tolerated using a hybrid redundancy unit embedded inside the cell. The hybrid unit represents a first line of defense against the discovered faults. However, if the fault is diagnosed as a permanent fault using a self-checking unit, the faulty cell will be killed and replaced with its neighboring T cell using three control signals generated by the local healing layer. One signal is used to close the output of the faulty cell by activating a network of tri state buffers. Second signal is used to activate the same genetic code stored in the configuration memory of the T cell. The last signal is activating the selection lines of four 2to1 multiplexers inside the transfer inputs unit to reroute the input data around the faulty cell and make it access the healthy T cell. This reconfiguration process, which represents the second line of defense against the first permanent fault, can be activated for the other three functional B cells: F1, F2, and F3 concurrently or sequentially based on the fault occurrence in any one of these cells. Third line of defense, which includes differentiating the spare stem cell, can be activated either in case of the failure one of the four local healing layers to repair the B cell with its neighboring T cell or in case of the occurrence of two permanent faults in B cell and T cell sequentially.

At this time, the global healing should be notified through one of the four fault signals generated by the four T cells: R0, R1, R2, and R3. The global healing layer can produce three control signals for the recovery of each faulty T cell and the select PF signals unit selects which one of these three signals used to configure the stem cell. This unit is generating only three control signals: signal to activate one of the four genetic codes stored in the configuration memory of the stem cell and another signal goes to the selection lines of four 2to1 multiplexers to make the faulty T cell reroute its four input signals into the stem cell. The last signal is used to activate the enable signals for a network of tristate buffers connected to the output ports of the faulty BT pair. In addition, the two outputs of each pair of neighboring cells: B cell and T cell and the output of the stem cell are driven into one output port through a 3to1 multiplexer. Since we have four pairs of BT cells in the tissue sub-layer, four 3to1 multiplexers are connected to four digital output ports. Hardware implementation of this proposed architecture is realized on Field Programmable Gate Arrays

(FPGAs), which provides a substantial underlying fabric of programmable elements that contain resources and deliver services to the external world.

The basic structure of the proposed biological B cell shown in Fig. 3 consists of two divisions: data flow and control flow. The data flow includes: direct I/O routing unit, diagonal I/O routing unit, and the data path for the Adaptive Functional Block (AFA) unit and the control flow includes: address reservation station, gene configuration memory, and the control path of the FAFU. Furthermore, the proposed cell is basically working in two operational modes: configuration mode and run mode. In the configuration mode, the values of the genetic control registers, which are used to configure the FAFU and I/O routing units, are being stored inside the gene memory. However, in run mode, the cell works as an event driven component being used as reactive real time control system in which, once the cell receives the address of its neighboring cell, it performs a couple of responses until the resulted output is available on one output port of the two I/O routing units. In addition, the input registers located inside the functional execution unit are being enabled to read the input digital data through the I/O routing units, executing the functionality of the task, writing the resulted data into the output registers, and sending a 'done' signal to a control unit. Since the global function of the critical application requires different types of functions (NOT, OR, AND,…), the configuration genetic code is stored inside the configuration memory of each cell in the system, but which logical operation can be executed by the cell, that is based on the expression for the genome memory. To synchronize the operation of the cells, each functional cell needs to have a simple finite state machine (FSM) that controls the flow of data inside the cell.

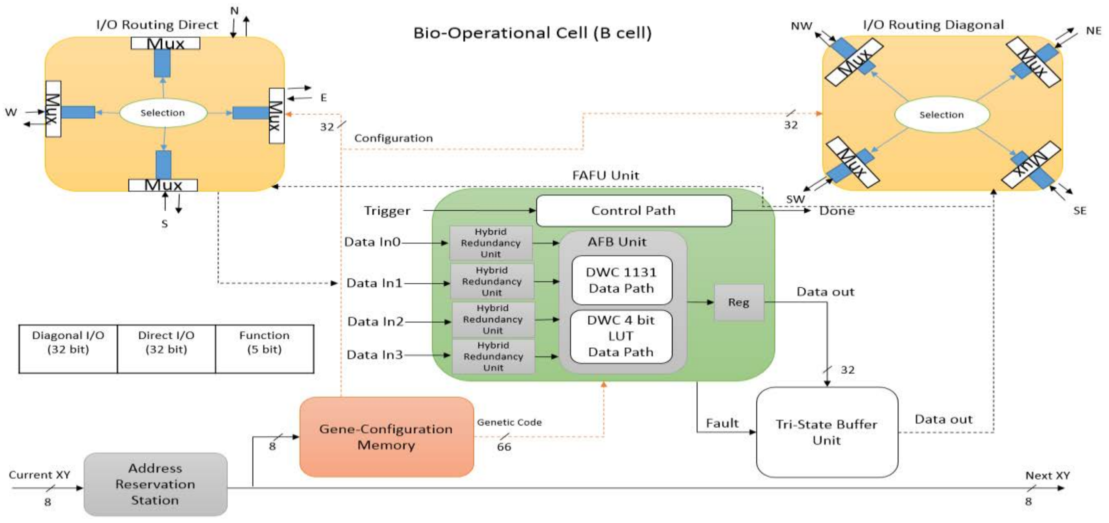


Figure 3. The internal structure of the biological inspired cell

3.1 IEC 1131 and 61499 standards

Programmable Logic Controllers (PLCs) vendors provide the industrial control system designer to use different languages to write a PLC program and some of these languages are ladder diagram(LD), structure text(ST), instruction list(IL) and functional block diagram(FBD). The FBD language was designed based on some international standards (e.g. IEC 1131 and 61499 standards) and these standards define the operational semantics for each functional block in the program. 1131 standard was proposed in 1993 to be used as a design standard in the field of industrial control systems and this standard was developed in 2003 to be used in distributed reactive control systems [17].

3.2 1131-based Adaptive Functional Block Operational Semantics

A fundamental decision in developing operational semantics for function blocks is to determine what model of computation is best suited to guarantee the deterministic behavior. In proposing a synchronous model for function blocks, it is desirable that the proposed semantics are defined according to standards like IEC1131 and 61499. The proposed AFB shown in Fig. 4, is build based on the concept of PLCs and divided into three divisions: control flow, 1131-based data flow, and LUT-based data flow. According to the standards, the AFB can only be activated with the occurrence of "Trigger" input event signal. After that, the control flow activates either the 1131-based data flow or the LUT-based data flow based on the value of the received control signals and the output data is selected through a 2to1 multiplexer. The 1131-based data flow is comprised of four input registers, 1131-based functional unit, and one output register and the LUT-based data flow consists of four input registers, 4-input Look-Up Table (LUT), and one output register. This AFB takes three clock cycles as an execution time to produce the output value and make it available at the "Data_out" output port after the trigger signal is activated. The functional unit is capable of implementing one function of up to four variables and of 22 functions at one time. These functions are logical and bitwise AND, OR, NAND, NOR, XOR, XNOR, and NOT operators. It also includes selection operations MAX, MIN, relational operators, LT, LE, GT, GE, EQ, and Product functionality). Each local function implemented in each cell is assumed to be a part of a larger function (digital I&C system).

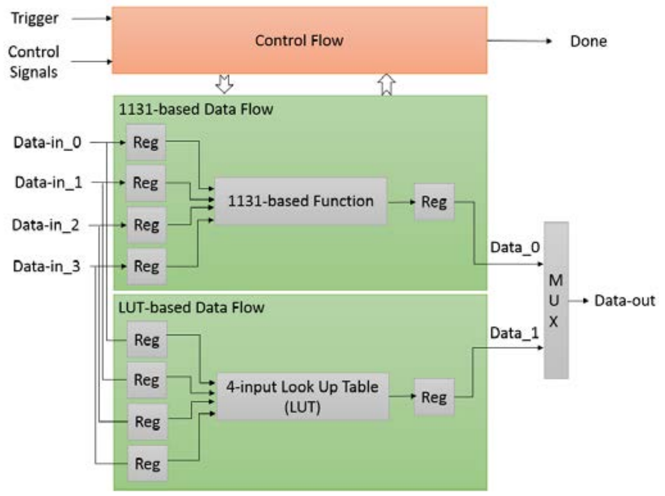


Figure 4. 1131-based Adaptive Functional Block

3.3 Fault-Tolerance Techniques for 1131-based AFB

Fault tolerance techniques based on redundancy are typically used to detect the fault occurrence in digital systems. The redundancy can either be in time, which is called active redundancy and used to detect the transient faults that may occur in sequential logic circuits, or in hardware, which is called passive redundancy and used to identify the permanent faults occurring in combinational logic circuits [16]. Fault tolerance involves design of a system or a subsystem in such a way that, even if certain types of faults occur, a fault will not propagate through the system and induce a failure in the system. In our proposed architecture, two fault tolerance mechanisms are embedded in the design of the B cell to increase the reliability level of the whole system. Firstly, a duplication with a comparison (DWC) unit shown in Fig. 5, which imitates the operation of B cells in the immune system that attack the antigens in their local locations, was designed as a passive redundancy technique to detect the permanent faults that may attack the 1131 functional unit which is working as a logical combinational unit. This unit is comprised of two duplicated functional units, a comparator, a register and tri state buffer unit.

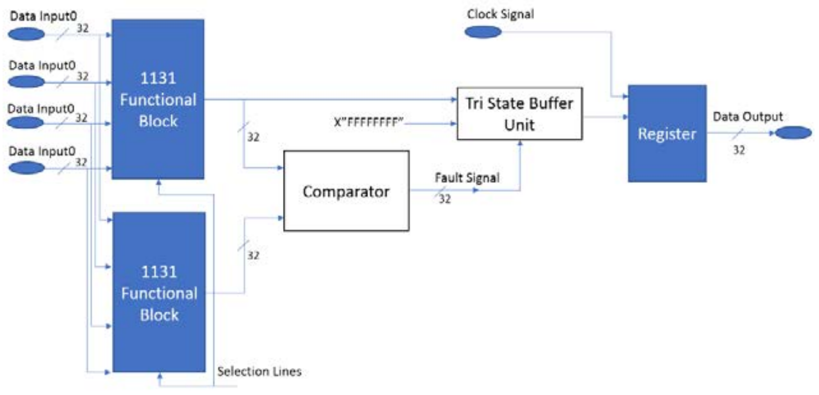


Figure 5.   DWC 1131 functional unit

Secondly, as it can be seen in Fig. 6, a hybrid redundancy unit was designed as an active redundancy technique to tolerate the effects of transient faults that may defect the input registers for the AFA unit. This unit basically consists of eight hardware components: three registers, three error detection units, a monitoring switch, and a comparator. The outputs of three registers are connected to three inputs of the switch in such a way that the two outputs of the switch are equaled to the values of two inputs. Once a transient fault is injected into one of the three registers, the monitoring unit is triggered and use the value of the spare register that doesn't need any initialization process. As a result, the data out of the hybrid redundancy unit is producing an erroneous value until the rising edge of the next clock. At the rising edge of the clock, the switching unit eliminates the faulty Reg unit and replaces it with the spare Reg unit. This reconfiguration process is designed to be fast and thus produces an erroneous value only for half a clock cycle.

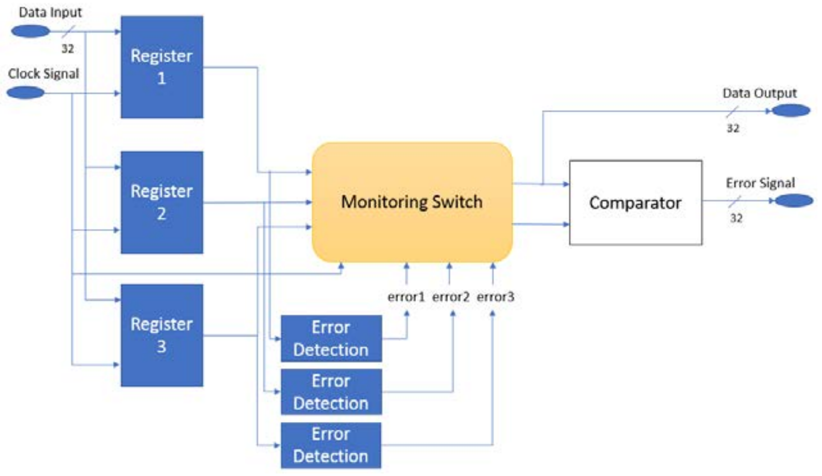


Figure 6. Hybrid redundancy unit

3.4 Fault Aware 1131-based Adaptive Functional Unit

The Fault Aware 1131-based Functional Unit (FAFU) shown in Fig. 7, combines the two fault tolerance techniques mentioned previously. As a consequence, this unit has become immune against the effects of two types of faults: transient and permanent. The DWC 1131 generic unit and the hybrid redundancy unit are embedded in the self-checking unit. A transient fault was simultaneously injected into four input registers of the four hybrid redundancy units (HRU) embedded inside the fault aware functional unit (FAFU) in which transient faults can be tolerated sequentially for an unlimited number of times using a self-monitoring switching unit. This unit continuously reads the status of the error signals produced by three triplicated duplication with comparison (DWC) register units. The concept of HRU is based on the operation of active redundancy in which a hot spare unit is taking over the functionality of the defective unit. Normally, the combinational logical circuits are vulnerable to be defected by permanent failure modes and the sequential logical circuits are susceptible to transient faults (soft errors). Once the trigger signal is activated, the FAFU goes into a "start state" state which takes one clock cycle. After that, one clock cycle is taken to activate the four parallel input registers, and then the "wait-execution" state takes one cycle for execution. In "Done" state, the output is produced at the output registers. That means three cycles are required for producing the output and raising the done signal after the trigger is activated.

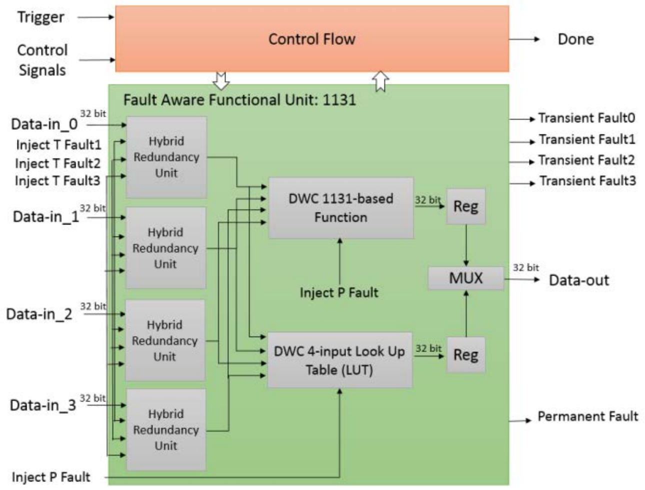


Figure 7. Fault Aware Functional Unit (FAFU) operational schematic

3.5 Fault Tolerance Hypercube Network Topology

The hypercube network architecture shown in Fig. 8, consists of nine functional cells: four bio active cells, four pre-generated redundant cells, and one bio stem cell. These cells are connected in such a way that each active cell that imitates the operation of the biological B cell is protected by its own neighboring T cell. Once one of the four T cells fails to repair the faulty B cell, a stem cell can be differentiated to perform the same functionality as a second line of defense.

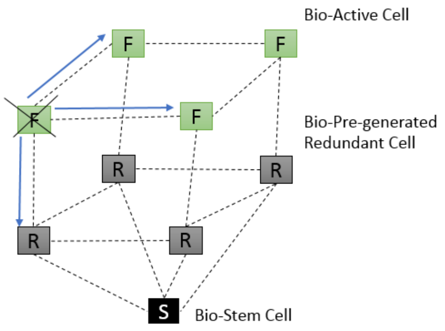


Figure 8. Hypercube network topology

Furthermore, the fault detection and self-healing mechanisms occurring inside all functional cells of the system are working concurrently to determine the type of the detected fault whether it is transient or permanent and based on the results of the diagnosis, the faulty cell will either be killed or used to increase the system efficiency. If the fault is determined to be transient, its effect is tolerated quickly using a hybrid redundancy unit and the same faulty cell is used to execute the local functionality. On the other hand, the faulty cell defected by a permanent fault triggers a hypercube elimination strategy, in which the output of the faulty cell is placed in a high impedance state to be protected as a Fault Containment Region (FCR). The FCR is designed in such a way that prevents the faulty functional cell from producing erroneous values either to its neighboring cells or to the external world through connecting a network of tri state buffers before the output port of the cell. After that, the address of the faulty cell is being sent to its three neighboring cells: F, F, and R (see Fig. 8). The neighboring two F cells will use the received address to reroute around the faulty cell and the pre-generated cell will use the same address to activate the same genetic code. This process is imitating how the immune system is generating the T cells to surround and attack the antigen in its location and kill it. Furthermore, another self-diagnosis mechanism is working inside the pre-generated cell to detect any new fault and the same steps will be repeated to perform the hypercube elimination strategy if the fault is diagnosed as permanent. However, a spare stem cell is required to store the configuration control register for the four B cells connected in the hyper cube. This process is imitating how the embryonic stem cell is differentiated in case of the failure of the immune system in generating the T cells as a first line of defense.

4 CASE STUDIES AND RESULTS DISCUSSION

In this section of the paper, two case studies are presented to verify and demonstrate the correct operation of the self-healing mechanisms occurring at different layers of the proposed Bio SymPLe architecture. Quartus Prime 15.1 Lite Edition software from Altera was used as a design tool that supports several FPGA device families and the system was embedded in a digital platform the Altera Cyclone V (5CGXFC7C7F23C8) FPGA.

4.1 First case study

This case study presents the hardware implementation and simulation results for the high-level perspective of Bio SymPLe architecture. Fig. 9(a) shows a block diagram for eight hardware components: four pairs of BT cells, one global healing layer, select PF unit, global transfer inputs unit, and one spare stem cell, that are connected to realize the architecture and Fig. 9(b) shows the actual hardware implementation.

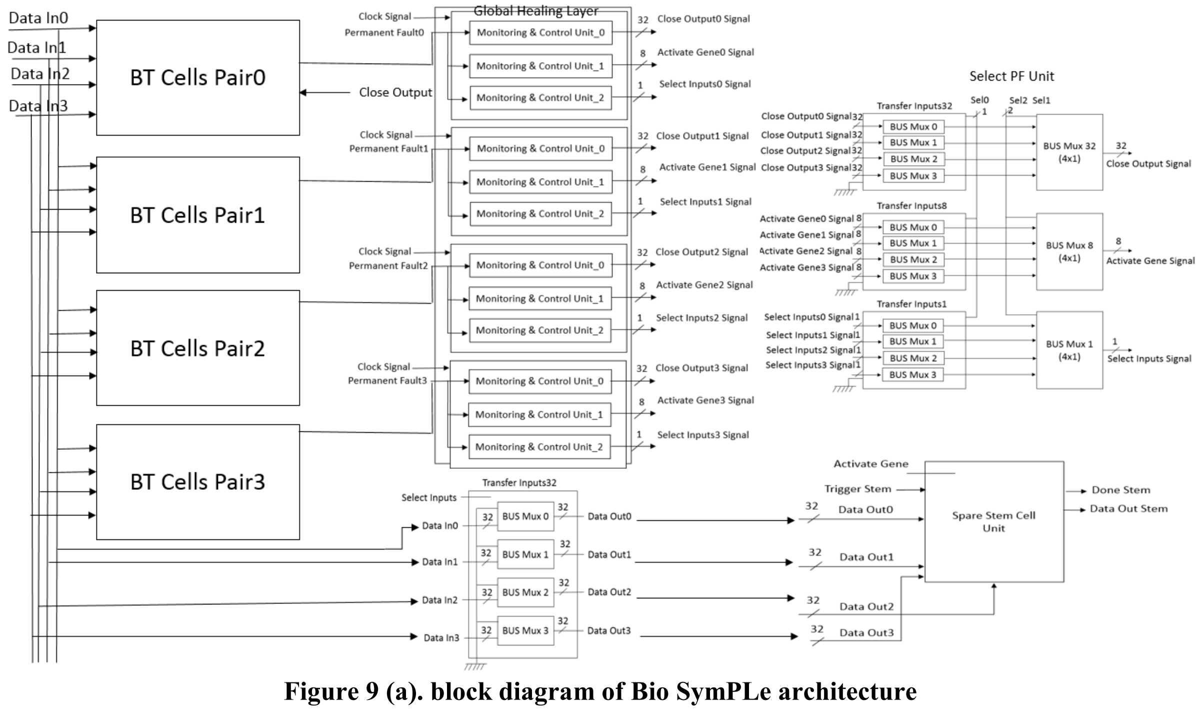


Figure 9 (a). block diagram of Bio SymPLe architecture

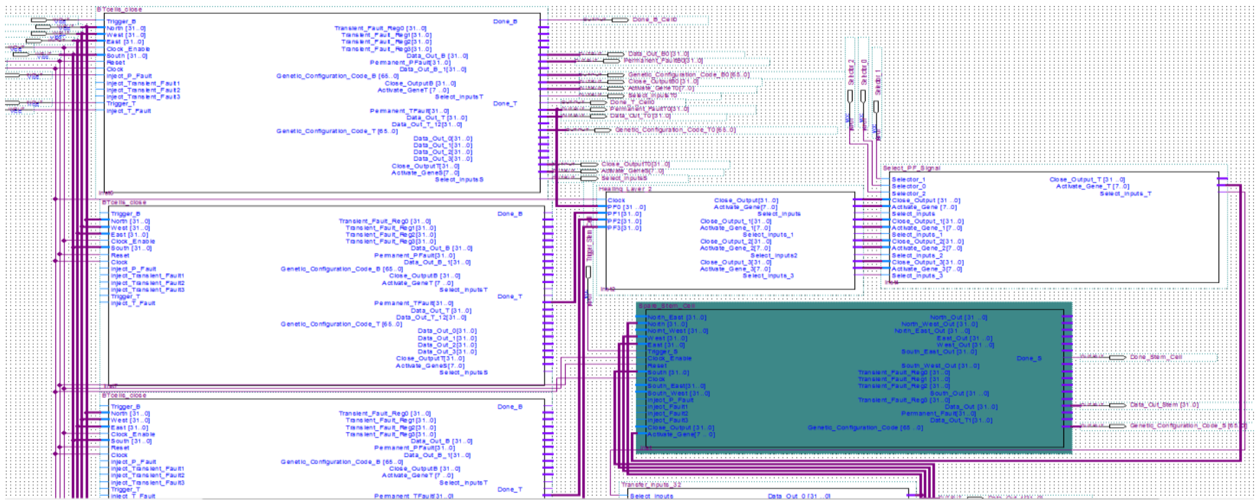


Figure 9 (b). Low level schematic diagram of Bio SymPLe architecture

Fig. 9(c) shows that two sequential permanent faults have been injected into the 1131-based FAFU units of B cell and T cell located in BT cells pair0, faults are identified by embedded self-checking units, self-healing is activated, and the system is repaired.

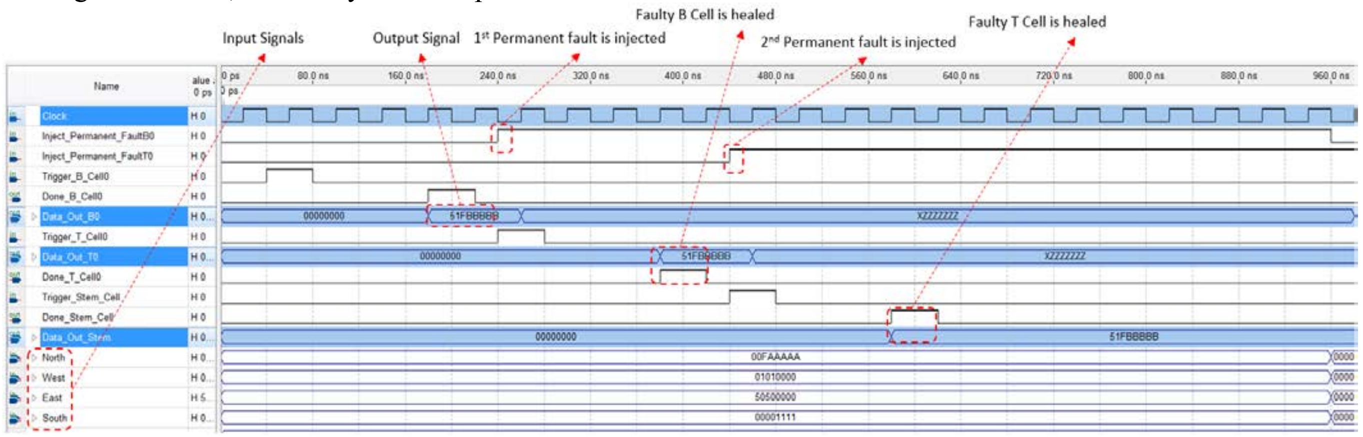


Figure 9 (c). Time diagram simulation results

Regarding the B cell recovery, once the trigger signal of the B cell0 is activated at time 40ns, it will take this cell four clock cycles to produce the 32bit hexadecimal output value "51FBBBBB" with the rising edge of the clock at time 180ns. This value represents the resulted output after executing a bitwise OR function on the four input signals values:"00FAAAAA","01010000", "50500000", and "00001111". Whenever a permanent fault is injected into the FAFU of a cell, it will be detected immediately by a self-checking unit embedded in the same cell and the three healing mechanisms start working. For example, at time 240ns, a permanent fault is injected into the FAFU of the B cell0 and the self-checking unit detects that fault and activates the tri state buffer unit embedded in the same cell. The buffer unit is working as a fault confinement region (FCR) for the erroneous value produced by the faulty cell and it causes the value of the "Data_Out_B0" to become high impedance with the rising edge of the next clock. Second mechanism is transferring the four input signals of the faulty B cell to the inputs of the pre-generated T cell as a rerouting process and third mechanism is activating the same 66-bit genetic code stored in the neighboring pre-generated T cell. These are the three basic strategies that were proposed as a self-healing mechanism for the faulty B cell. However, the T cell cannot produce the correct output value until it receives its own trigger signal and generates the done signal. For instance, at the rising edge before time 400ns, the done signal of the T cell is generated and the cell produces the output value which represents the end of the healing mechanism for the faulty B cell. Additionally, regarding the T cell recovery, once the trigger signal of the T cell is activated, it produces the output value with the rising edge of the clock. Also, the self-checking unit can detect the second sequential fault being injected into the FAFU of this cell and the healing mechanism starts working. Ultimately, regarding the S cell recovery, there is no benefit from injecting a fault into the S cell and observing the healing mechanisms because the limit of one hypercube network topology is either to tolerate two sequential permanent faults or to tolerate four permanent faults concurrently.

4.2 Second case study

This case study presents the hardware implementation and simulation results for the internal architecture of one BT cells pair. Fig. 10(a) shows a block diagram for the five hardware components required inside each pair of the Bio SymPLe architecture: one bio operational B cell, one pre-generated T cell, two local healing layers, and a local transfer unit, and Fig. 10(b) shows the actual hardware implementation.

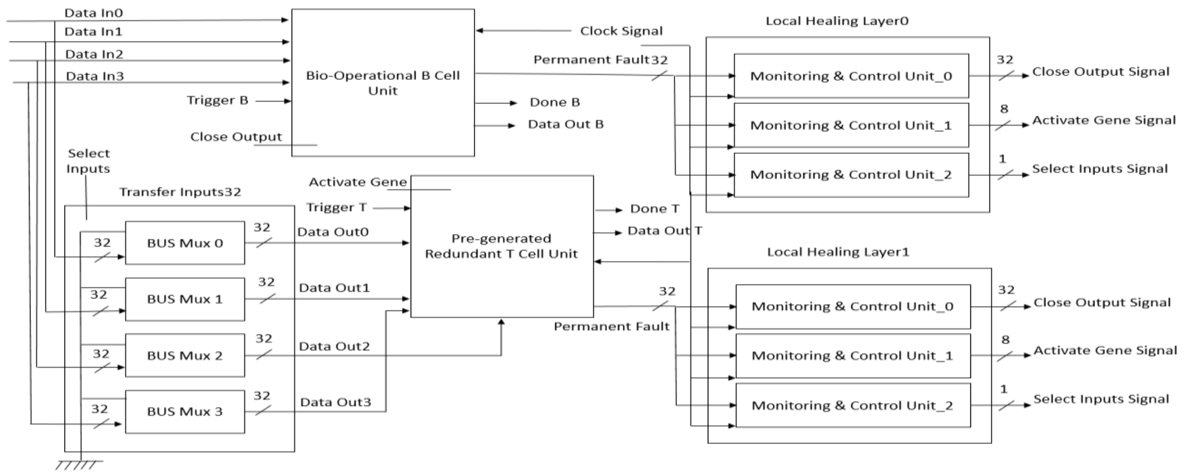


Figure 10 (a). Block diagram of one B cell, one T cell, and two local healing layers

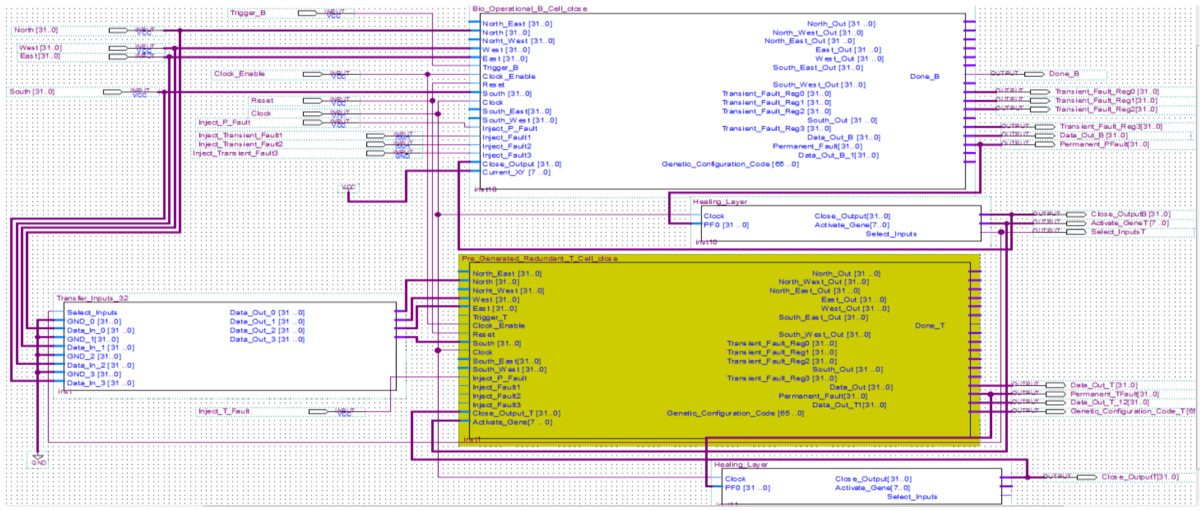


Figure 10 (b). Low level schematic diagram of one B cell, one T cell, and two local healing layers

Fig. 10(c) shows that three sequential transient faults have been injected into the input registers of FAFU unit for the bio operational cell located in pair0, hybrid redundancy units are tolerating their impacts quickly, and the output signal is generated without producing any erroneous value.

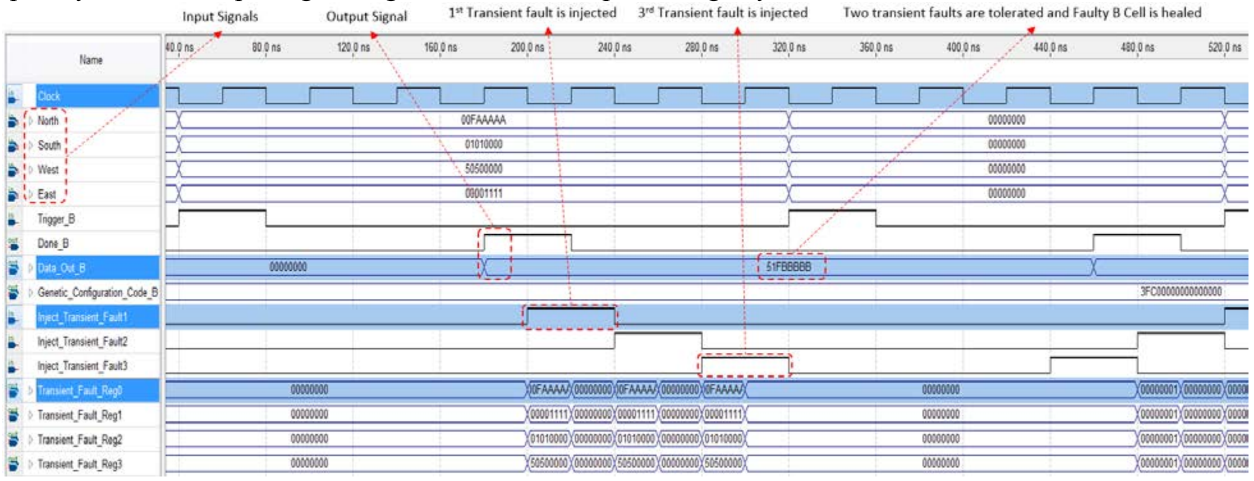


Figure 10 (c). Time diagram simulation results

Regarding the fault tolerance inside the bio B cell using the hybrid redundancy unit, once the trigger signal of the B cell0 is activated at time 40ns, it will take this cell four clock cycles to generate the 32bit hexadecimal output value "51FBBBBB" with the rising edge of the clock at time 180ns. This value represents the resulted output after executing a bitwise OR function on the four input signals values:"00FAAAAA","01010000", "50500000", and "00001111". Three sequential transient faults were injected into three input registers for the FAFU of the bio B cell at times:200ns, 240ns, and 280ns. As a consequence, the four output tag registers:" Transient_Fault_Reg0"," Transient_Fault_Reg1"," Transient_Fault_Reg2", and" Transient_Fault_Reg3", shown in Fig. 10(c) are generating an erroneous value only for half a clock cycle. However, the output signal "Data_Out_B" is not affected by these transient faults and continue to produce the correct value.

5 CONCLUSION AND FUTURE WORK

This paper presents the ongoing work of biologically inspired self-healing SymPLe architecture, which is aimed to be used in nuclear power plant (NPP) digital applications. These applications require high levels of resilience against several kinds of failure modes. The proposed architecture aims at both decreasing the complexity level of the bio inspired functional cell and increasing the reliability and safety levels of the self-healing digital I&C system by embedding different and diverse fault tolerant techniques in the system design. The design of three biologically inspired cells: B cell, T cell, and stem cell was based on combining concepts from biology and PLC architecture. The biological concepts provide the architecture with a high resilience against different types of faults with recovery mechanisms at different layers in the system. However, the PLC architecture has led to a concept of separation between the control flow and the data flow which is not available in other self-healing systems and that will enhance the process of verification and validation required in critical systems. Future work will include embedding more diverse biological concepts and different fault tolerance techniques inside the cell to increase the reliability level of the whole digital I&C system. Also, a concept of verification and validation will be used based on formal methods to verify both the data flow and the control flow at runtime inside the FAFU unit. Furthermore, different network topologies will be used to make the functional cells work cooperatively to perform the functionality for a complex safety critical NPP application.

6 REFERENCES

1. Storey, N, "*Safety-Critical Computer Systems*". New York, NY: Addison Wesley Longman, 1996.
2. Knight, J; Randell, Brian, "*Fundamentals of Dependable Computing for Software Engineers*". FL: Taylor & Francis Group, 2012.
3. Lee, E., & Seshia, S., "Introduction to Embedded Systems- A Cyber-Physical Systems Approach" Second Edition, LeeSeshia.org, 2015.
4. Alberts, B.; Johnson, A.; Lewis, J.; Morgan, D., "*Molecular Biology of the Cell*". New York, NY: Garland Science, 2015.
5. Johnson, B, "*Design and Analysis of Fault-Tolerant Digital Systems*". New York, NY: Addison-Wesley Publishing Company, Inc. 1989.
6. Mange, D. (1992). Microprogrammed systems: an introduction to firmware theory. London: Chapman & Hall.
7. Ortega, C., & Tyrrell, A. (1997). Biologically inspired reconfigurable hardware for dependable applications. In *Hardware Systems for Dependable Applications (Digest No: 1997/335), IEE Half-Day Colloquium on* (pp. 3–1). IET. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=660654
8. Ortega, C., & Tyrrell, A. (1999). Reliability analysis in self-repairing embryonic systems. In *Evolvable Hardware, 1999. Proceedings of the First NASA/DoD Workshop on* (pp. 120–128). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=785443
9. Lala, P. K., & Kumar, B. K. (2003). An architecture for self-healing digital systems. *Journal of Electronic Testing*, *19*(5), 523–535.
10. Zhang, X., Dragffy, G., & Pipe, A. G. (2006). Embryonics: A Path to Artificial Life? *Artificial Life*, *12*(3), 313–332. http://doi.org/10.1162/artl.2006.12.3.313
11. Tempesti, G., Mange, D., Mudry, P.-A., Rossier, J., & Stauffer, A. (2007). Self-replicating hardware for reliability: The embryonics project. *ACM Journal on Emerging Technologies in Computing Systems*, *3*(2), 9–es. http://doi.org/10.1145/1265949.1265955
12. Zhang, Z., Wang, Y., Yang, S., Yao, R., & Cui, J. (2008). The research of self-repairing digital circuit based on embryonic cellular array. *Neural Computing and Applications*, *17*(2), 145–151. http://doi.org/10.1007/s00521-007-0095-9
13. Butler, R, "A Primer on Architectural Level Fault Tolerance". National Aeronautics and Space Administration: Langley Research Center, Hampton, Virginia. 2008.
14. Kim, S., Chu, H., Yang, I., Hong, S., Jung, S. H., & Cho, K.-H. (2012). A Hierarchical Self-Repairing Architecture for Fast Fault Recovery of Digital Systems Inspired From Paralogous Gene Regulatory Circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *20*(12), 2315–2328. http://doi.org/10.1109/TVLSI.2011.2176544
15. IBM, "*Autonomic computing toolkit*," Tech. Rep. http://www- 106.ibm.com/developer works/autonomic/probdet.html, IBM, 2004.
16. Kastensmidt, F; Carro, L; Reis, R, "*Fault-Tolerance Techniques for SRAM-based FPGAs*". Dordrecht, Netherlands: Springer. 2006.

17. R. Wieringa, "*Design methods for reactive systems*": Yourdan, Statemate, and the UML. Morgan Kaufmann, 2003.