jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2023, VOL. 69, NO. 2, PP. 211–217
Manuscript received December 4, 2022; revised April, 2023. DOI: 10.24425/ijet.2023.144352

Embryonic Architecture with Built-in Self-test and GA Evolved Configuration Data

Gayatri Malhotra, Punithavathi Duraiswamy, and J.K. Kishore

***Abstract*—The embryonic architecture, which draws inspiration from the biological process of ontogeny, has built-in mechanisms for self-repair. The entire genome is stored in the embryonic cells, allowing the data to be replicated in healthy cells in the event of a single cell failure in the embryonic fabric. A specially designed genetic algorithm (GA) is used to evolve the configuration information for embryonic cells. Any failed embryonic cell must be indicated via the proposed Built-in Self-test (BIST) the module of the embryonic fabric. This paper recommends an effective centralized BIST design for a novel embryonic fabric. Every embryonic cell is scanned by the proposed BIST in case the self-test mode is activated. The centralized BIST design uses less hardware than if it were integrated into each embryonic cell. To reduce the size of the data, the genome or configuration data of each embryonic cell is decoded using Cartesian Genetic Programming (CGP). The GA is tested for the 1-bit adder and 2-bit comparator circuits that are implemented in the embryonic cell. Fault detection is possible at every function of the cell due to the BIST module's design. The CGP format can also offer gate-level fault detection. Customized GA and BIST are combined with the novel embryonic architecture. In the embryonic cell, self-repair is accomplished via data scrubbing for transient errors.**

***Keywords*—Embryonic; BIST; Self-test; Genetic Algorithm; Cartesian Genetic Programming**

I. Introduction

THE deep space systems require a different approach for fault tolerance due to communication delay and limited hardware resources. These systems should have the ability to re-configurable itself to counter new challenges. The embryonic fabric-based cellular architecture provides self-repair using additional spare cells instead of triplicating all the cells. Thus, this approach is suitable for smaller space system designs in deep space missions. The embryonic circuit's growth and its operation are decided by the stored genome configuration data. The embryonic cellular structure has inbuilt fault tolerance due to its structure. Each cell contains the total genome data of the fabric. The novel embryonic architecture, which evolved from the concept of cloning, is used by the biological cell to grow into a bigger system with BIST, as proposed in this paper. The reconfigurable self-healing embryonic cell mentioned in [1] contains self-diagnostic within each cell. This calls for multiple resources and increases with the number of cells. The BIST proposed in this work is centralized to the fabric and requires fewer resources. It is based on a comparison of the reference golden signature (no fault condition) with the working circuit response signature. The signature register size only will increase for the bigger fabric with more outputs.

Gayatri Malhotra is with U R Rao Satellite Centre and M S Ramaiah University of Applied Science, India (e-mail: gayatri_t76@yahoo.com).
Punithavathi Duraiswamy is with M S Ramaiah University of Applied Science, India (e-mail: punithavathi.ec.et@msruas.ac.in).
J.K. Kishore is with U R Rao Satellite Centre, India (e-mail: jkk@ursc.gov.in).

In the reference [1], the BIST module uses the EXOR gate to compare the function unit output to its duplicate. The number of EXOR gates and function unit duplication will increase with the number of outcomes [2]. Also, the BIST design must be modified for different cell functions. The proposed BIST in this work needs the change of output signal allocation to the linear feedback shift register used for generating the signature response. Entire function duplication is not required for the BIST module.

Many approaches to self-healing in the embryonic fabric act after detecting faulty output, while the proposed BIST is for self-check-in at regular intervals during run-time. After fault detection, self-repair is initiated automatically. This is an attempt to minimize the use of duplicating function resources and to standardize the BIST design for different circuits. It will develop an intelligent adaptive and reconfigurable system that can automatically initiate self-reconfigurable after the fault is detected. The detection is possible at the cell level which contains multiple molecules or nodes. Fault detection is also possible at the molecule level, equivalent to a node in the CGP format. Node level fault identification is possible due to the CGP format of data in the design. Initially, the fault is detected at the cell level, then the corresponding node or molecule can be identified by verifying each node's output.

Some approaches to self-healing in the embryonic fabric call for the embryonic cell's row or column elimination [3] [4]. As for the transient faults, the entire embryonic cell's row or column elimination is not the scrubbing of configuration memory, which is planned for self-repair [5]. A transient error can be repaired after data scrubbing [6]. In the proposed embryonic fabric, additional spare cells are to be utilized only in the event of faulty cell isolation.

The design is coded using Verilog and simulated for 1-bit adder and 2-bit comparator cell design. The genome data is generated through a GA and encoded as Cartesian Genetic Programming (CGP). Section 2 describes the novel embryonic architecture with CGP data. Section 3 proposes an embryonic fabric design for combinatorial digital circuits. It describes the design of embryonic cells and switch box interconnection.

Section 4 presents the Built-in Self-Test design methodology for Embryonic Cells. The sub-modules of the design like the controller, random pattern generator (RPG), response analyzer, and fault detection method are underlined. Section 5 is about fault detection process for adder and comparator cells. Section 6 contains the simulation results. Section 7 is about GA design for configuration data generation in CGP format. Section 8 concludes the results, and Section 9 is the scope for future work.

II. Embryonic Architecture With CGP Configuration Data

The embryonic design approach is considered to design the fault-tolerant electronics system [7] [8]. Embryonic, a homogeneous array of embryonic cells, possesses self-healing due to its structure. The proposed embryonic fabric has a multi-cellular system, where each cell contains a CGP decoder, Memory, and self-repair unit. The hardware module for the BIST controller is common to the fabric, while the self-repair module is part of each cell. This way, the hardware resources for the BIST controller do not need to be duplicated in each cell. The fault detection through BIST enables the identification of the faulty embryonic cell; further, the CGP data format can identify the node level fault in the defective cell. In this work, though the fault detection is limited to cell level, the node/gate level detection is possible as a principle. The self-repair module implements memory scrubbing for transient faults, thus incorporating registers only.

The CGP format represents the digital circuit as a rectangular array of nodes. Each node is an operation on the node inputs. Integers sequentially index all node inputs, node operations, and node outputs. The configuration data (Genome data) for the embryonic cell is a linear string of these integers. Applying GA for the digital circuit design explores a more extensive search space. The GA achieves design optimization better than traditional approaches [9] [10]. The configuration genome data generation for embryonic cell is through GA, while the genome data format is Cartesian Genetic Programming (CGP) [11] [12]. The CGP data generation is through HsClone and OIMGA algorithms, while other GA are also planned to be tested [13].

The embryonic is also known as 'Electronic Stem Cells' [14]. It implements a digital system with fault-tolerant capabilities inspired by the ontogeny process. The embryonic cellular structure can induce fault tolerance in the design by self-repairing [3] [15]. The natural methods of growing, reproducing, and healing are adopted in electronic design. The 'Stem Cell' has the feature of becoming anything they want to be, inspiring the electronics design for using embryonic cellular structure. The division mechanism is the main driving force behind the development of the entire organism. The same is applied as a cloning mechanism in this embryonic fabric design.

There are several approaches to implement self-healing, self-repair, and self-replicate in the embryonic cellular architecture. The electronic tissue called POEtic design [16] is inspired by three life axes: Phylogenesis, Ontogenesis, and Epigenesis. The Ontogenetic axis refers to the cellular growth that helps in self-repair. The structural principles of living organisms like multi-cellular architecture, cellular division, and cellular differentiation are utilized to enable systems to grow, self-replicate, and self-repair [17]. The embryonic cell architecture is designed for self-diagnostic, and the fault recovery methods embedded in it. The approach described in [1] is to kill the faulty cell and make it transparent while transferring its functions to a neighboring cell. The self-healing is achieved through cell elimination and further by row elimination. In eDNA approach [18], the electronic cell 'eCell' reads the electronic DNA 'eDNA' to interpret the function it must do and, in case of one eCell failure, to move the function to another eCell. Thus, to create self-organization and self-healing of electronic cells.

In this paper, a Built-in Self-test (BIST) methodology [19] is integrated with the embryonic fabric to initiate self-repair at the embryonic cell level. The BIST controller controls the different types of embryonic cells per the selected gene. The fault detection information at the embryonic cell level is available to the embryonic fabric controller. Based on this, the controller initiates the self-repair of the faulty cell. This approach deals with transient errors and provides self-healing [6].

III. CGP Decoding and Built-in Self-Test

The proposed novel embryonic fabric consists of embryonic cells, switch boxes, and the fabric controller module. The fabric controller must control the data transfer between the cells, switch boxes, and input-output units. The novel embryonic fabric architecture is shown in Fig. 1. It consists of ten cells, four each for a 1-bit adder and a 2-bit comparator with two spare cells. Four 1-bit adder cells in cascading can build a 4-bit adder; similarly, an 8-bit comparator from four 2-bit comparators is built using switch boxes for cascading the signals.

In the FPGA structure, a Fixed configuration bits are required to configure its logic blocks and establish interconnections. The embryonic cell fabric achieves this with fewer configuration bits as CGP is the data format. The cloning process in the fabric implements the data copy to all the cells. The first cell's genome data is loaded externally, and cloning takes care of copying the genome data bits to the following cells during run-time. The cloning process is discussed earlier, where data is in Look Up Table (LUT) format [20]. In this work, the data type is changed from LUT format to CGP format. The circuit size is flexible and contains clone data bits and CGP data.

The CGP data format over the LUT data format dominates in the case of modular design. The 4-bit adder needs 29 bits, while the CGP format needs (45 + 4) bits (1-bit adder and clone count for four cells). The genome data contains two genes, adder, and comparator functions. The CGP decoder decodes the gene data within the embryonic cell. The genome data is 161 bits with CGP data for a 1-bit adder and 2-bit comparator. The fabric controller integrates the BIST controller to enable self-test. The self-test is established at the cell level to control self-repair also at the cell level.

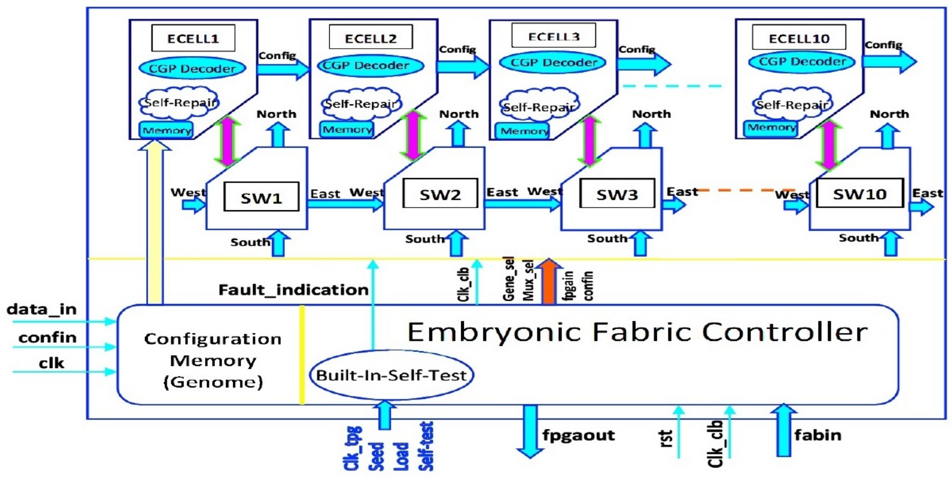

Fig. 1. Embryonic Fabric Architecture

A. *Embryonic Adder and Comparator Cell Design*

The embryonic cell contains genome data memory, cloning mechanism, CGP decoder, cell configuration controller, and self-repair module. The 'conn' is the trigger signal to initiate genome data loading into First cell. The cascading of cells is possible to have a scalable circuit design. The 4-bit adder is designed from 1-bit adder configuration data, and the 8-bit comparator is designed from 2-bit comparator configuration data.

In CGP, the representation of data is in the form of nodes. Each node is expressed as in1; in2; logical function. The K-map derived data for 1-bit adder is 0 1 0; 0 1 1; 3 2 0; 3 2 1; 6 4 2; 5; 7. This is the circuit designed through K-map and represented using CGP format. The last two nibbles are output nodes. Similarly, the K-map derived data for 2-bit comparator is 132; 13f; 31f; 02b;048; 47c; 28b; 699; a59; b; c. The CGP data length for adder, comparator, and counter data is 161 bits (45 + 4 bits: adder + 108 + 4 bits: comparator). The adder has five nodes (5 x 3octets x 3bits = 45 bits), comparator has nine nodes (9 x 3octets x 4bits = 108 bits). The clone count for each adder, comparator, and counter cell is expressed using four bits within configuration data.

B. *Embryonic Switch Box Design*

The data transfer between the cells is done through switch boxes. Each switch box has four directional buses and one dedicated bus with the neighbouring cell. The adder and comparator functions are carried out using east and west data buses. Buses are used to route carry for adder cells and compare signals for lower bit comparator cells. The signals between embryonic cells and switch boxes are also used in CGP data decoding.

C. *BIST Controller for Embryonic Cells*

BIST is a design for testability (DFT) that includes the testing features within the circuit under test (CUT). The basic BIST architecture requires the following modules within the embryonic fabric design controller- 1. A Test Controller 2. A Test Pattern Generator 3. A Response Analyzer

The Hardware pattern generator or test pattern generator (TPG) is to generate the test patterns for CUT. The selected type of TPG for embryonic fabric is a Linear feedback shift register (LFSR). A response analyzer (RA) is required to compare the CUT response with a stored response. It is designed using an another LFSR, which is used as a signature analyzer. It compacts and analyzes the CUT test responses to determine any fault. During normal operation mode, CUT (embryonic fabric) receives its inputs from the test bench/simulator and performs the function for which it is designed. During the Self-test mode, a TPG circuit applies a sequence of test patterns to the CUT. The output response compactor evaluates the test responses. The responses are compacted to form signatures. The golden reference signatures are already saved for no fault condition. The response signature is compared with the stored reference signature to find if CUT is good or faulty. The signature register saved for adder and comparator cells is depicted in simulation results.

As part of the embryonic fabric controller, the BIST controller initiates the Self-test mode whenever triggered. The 'fpgain' are inputs generated through the test bench as 'fabin' in normal mode and as 'testin' in Self-test mode. In case the Self-test is planned, the BIST controller initiates the test pattern generator module and response analysis module. The clock for BIST modules is 'clk-tpg,' simulated in Self-test mode only. The 'fpgaout' from the embryonic cell corresponding to the 'testin' are routed to the controller to the response analyzer. The 'fault-indication' from the RA is transferred through the controller to the faulty cell for further action.

D. *Test Pattern Generator Module*

The TPG is implemented using LFSR, a 40-stage shift register formed from Inputs-outputs (FFs), with the outputs of selected FFs being fed back to the shift register's inputs. When LFSR is used for TPG, it must cycle rapidly through many states. These states are designed by the design parameters of the LFSR and generate the test patterns. The implemented LFSR is 40-stage with the characteristic polynomial Equation (1) is an example.

$$P(X) = X^{39} + X^{38} + 1 \tag{1}$$

The final LFSR configuration is to generate 40-bit test data (4 bits each for ten embryonic cells). LFSR is initially loaded with seed data when the load is '1'. The LFSR function is synchronous with 'clk-tpg'.

E. Output Response Analysis Module

The RA module must check the circuit's responses for the random test pattern applied as input. In the embryonic fabric of ten cells, the response from all the cells is to be verified. Each adder embryonic cell has two outputs - sum and carry. Each comparator embryonic cell also has two outputs - AsmlB and AlargB. As per the gene selected for a cell, the function is identified as an adder or comparator. The total response is enormous, so it is required to compact this response to a manageable size. The RA compresses large test responses into a single word called a signature. The signature is compared with the pre-stored golden signature obtained from the fault-free circuit responses using the same compression mechanism. If the signature matches the golden signature, the CUT is fault-free. Otherwise, it is faulty. The response analyser technique used for embryonic fabric is signature analysis. The response compaction is done using LFSR. The data bits from POs (Primary Outputs) are compacted by dividing the PO polynomial by the characteristic polynomial of LFSR. The remainder of the polynomial division is the signature. The seed value is usually zero before testing.

F. MISR for Response Compaction- Signature Analysis

In ordinary LFSR response compacter, one of these must be put for each PO; this will lead to hardware overhead. Multiple-Input Signature Register (MISR) is the type that compacts all the cell outputs into one LFSR. All responses of adder cells are superimposed into one 'signature-reg-adder'. Similarly, all responses of comparator cells are superimposed into 'signature-reg-cmprtr'. The design of adder MISR and comparator MISR are shown in Fig. 2 and Fig. 3.

Both MISR are 10-stage LFSR as there are five adder cells (four cascaded and one spare) with two outputs from each and five comparator cells with two outputs from each. The implemented LFSR is an internal feedback type and has characteristic polynomial Equation (2) is an example.

$$P(X) = X^9 + X^8 + 1 \tag{2}$$

IV. FAULT DETECTION OF EMBRYONIC ADDER AND COMPARATOR CELLS

For fault detection, the degree of data polynomial (output response from cells) should be less than $2^{10} - 1$, where 10 is the degree of the characteristic polynomial of the LFSR. The number of 'clk-tpg' cycles is calculated (about 1000), and after

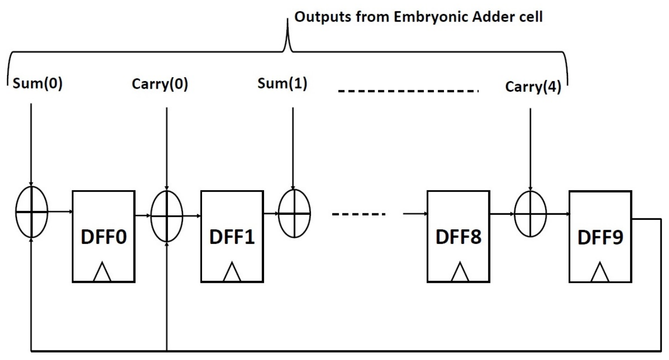

Fig. 2. MISR for Embryonic Adder Cells Outputs

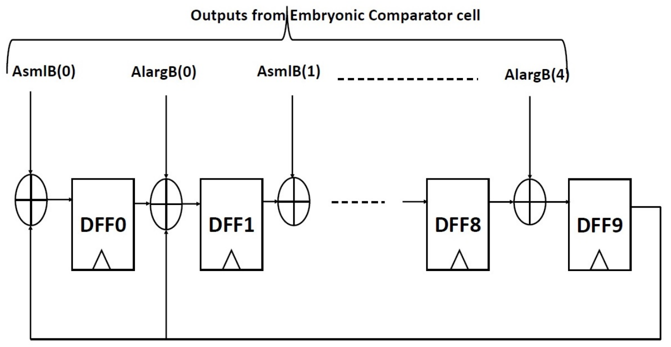

Fig. 3. MISR for Embryonic Comparator Cells Outputs

that fault-free signature analysis register (SAR) is saved. After the same clock cycles, the self-test signal is made low, and the 'signature-reg-adder' and 'signature-reg-cmprtr' are saved.

The 'signature' value for adder and comparator is compared with the saved fault-free signature value. The fault is simulated by making one adder cell outputs 'stuck at 0'. This fault type is considered for the initial test, while other fault types of multiple bit upsets are planned to be verified later with the design. The RA module successfully detects the 'stuck at 0' fault. The fault detection signal is routed to the fabric controller to further route to faulty cells. As the fault is identified, Self-repair is initiated by re-loading the configuration data to all defective cells (Scrubbing). The complete process of fault simulation to fault detection is depicted in Fig. 4. The reliability analysis of Self-repair is to be included [21].

Once the fault is detected at the adder cell level, the spare adder cell can replace the faulty cell for permanent fault. After the fault detection at the cell level, the fault is further localized to the sum or carry function level. For example, in the adder cell, the function sum uses two XOR gates (two nodes), so the fault at the sum function is due to one or both gates. Once the faulty nodes are identified, the respective gates can be replaced with available spare gates. The detailed design of gate level replacement is planned as the future scope of this work.

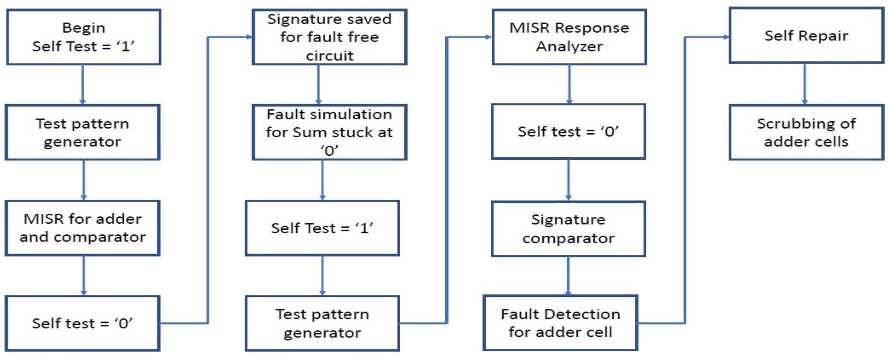

Fig. 4. Fault Simulation to Fault Detection Process Flow

V. Simulation Results

The embryonic fabric with ten cells is created through configuration data cloning. The data cloning process copies the configuration data to all the cells. The Self-test mode of the embryonic fabric is triggered with a 'Self-test' signal. The 'fpgain' is the input simulation from the test bench, while the 'testin' is the input simulation from the BIST module. Once the 'Self-test' is high, the 'seed' is loaded into LFSR for random test pattern generation. The 'testin' is applied to embryonic fabric cells as inputs, the corresponding 'fpgaout', the outputs from cells are saved. The 'gene-sel' for ten cells is defined as '0' for the adder cell and '1' for the comparator cell. The switch box buses 'dt-east' and 'dt-west' are assigned with the signals that need to be transferred between cells. The 'fault-indication' is the one-bit signal from each cell. After the fixed clock cycles, the signatures are saved. In the RA module, golden signature is saved for adder and comparator functions.

After the adder and comparator MISR signatures ('signature-reg-adder' and 'signature-reg-cmprtr') are saved, the 'sig-save' is made high to indicate the fabric controller. With the fault simulation, again BIST is re triggered. After fixed clock cycles, 'fault-indication' is analyzed by comparing 'state-out-adder' with 'signature-reg-adder' and 'state-out-cmprtr' with 'signature-reg-cmprtr'. The sum output from all adder cells is simulated for 'stuck at '0"; thus, fault indication is high ('0100001111') for adder cells only. This indication is routed through the fabric controller to each cell.

The cell initiates 'scrubbing' by re-loading genome data to cell memory. This is the Self-repair process for transient faults. The 'confin-test' initiates the faulty cell's scrubbing. After scrubbing, 'scrub-complete' is set high to indicate the completion of the process. To verify the fabric functionality, the RA is executed for the same fixed clock cycles. As 'signatures' are saved earlier, only a comparison with MISR 'state-out' is carried out. To check it, the simulation for 'stuck at '0" is removed now. Thus, the simulation results indicate no 'fault-indication'. This approach is applicable for transient errors, while the cell replacement is to be considered for the permanent error.

VI. PARALLEL GA DESIGN FOR CGP CONFIGURATION DATA GENERATION

Genetic algorithms are useful in solving search and optimization problems. The time required to get a converged solution can be high due to the computational complexity of GA. The parallel implementation of a GA on FPGA brings optimization in processing time [22]. The Fitness, crossover, mutation, and selection modules are replicated to form parallel pipelines. The novel parallel pipeline is applied to modify the HsClone algorithm [23] to achieve faster convergence. Parallel processing enables the algorithm to run concurrently on more sets of data. The fine-grained parallel GA is advantageous over sequential GA as well as the FPGA possesses the advantage as a parallel platform [24].

Another OIMGA algorithm is also modified and applied to generate the CGP data for adder, comparator, and counter circuits. The performance of the two algorithms is compared for convergence time.

A. *Parallel HsClone GA*

The 'PHsClone' GA is developed as an approach based on the 'half-siblings-and-a-clone' crossover technique. The pseudo-code for the modified parallel algorithm for adder and comparator is-

- Initially, the four parallel valid random patterns are generated
- Validity of the pattern is as per CGP constraint with no feedback
- The fitness for sum and carry/AsmlB and AlargB is checked for all node outputs
- If Fitness is best fit from any node, a new pattern is evolved
- If Fitness $<$ M percent; Crossover and mutation with a random pattern
- If Fitness $>$ M percent; Crossover and mutation with the current pattern
- Repeat from step 2 till the new evolved pattern is obtained

The value of M is tuned for 40% for adder and 70% for comparator. There are four parallel processes of population generation and testing. The configuration data size for the 1-bit adder is 45 bits, while for the 2-bit comparator, it is 120 bits (includes one spare node). The limitation of the HsClone is that it needs memory to store its population.

B. Optimum Individual Monogenetic GA-OIMGA

Another algorithm that is used for the generation of configuration data is OIMGA. This does not use much memory for storing the population. The different parameters of OIMGA are shown in Table I.

TABLE I
OIMGA PARAMETERS

GA Parameters	Description
l	Individual Length
n	Population Size
m	Size of miniature space around LOI
t-gens	Max no. of consecutive global generations
k-gens	Max no. of consecutive local generations
d-adjustor	Range of mutation of an individual
m-rate	Probability of mutation

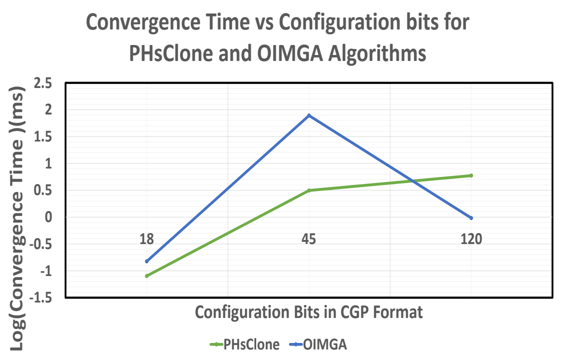

Fig. 5. Convergence Time for PHsClone and OIMGA

The convergence time vs. configuration data length vs. PHsClone and OIMGA algorithm performance is shown in Fig. 5. It is compared for 1-bit adder (45bits), 2-bit comparator (120 bits), and 1-bit counter (18 bits). Following are the inferences from the data, - The PHsClone performs better for a 1-bit counter than OIMGA. The reason is the parallel execution of four random searches in the PHsClone and the use of both crossover and mutation as genetic operators. - In the case of the 1-bit adder, the PHsClone performs better than OIMGA. Here the convergence time difference with OIMGA is huge. The reason is that the two functions of sum and carry are to be evolved in the same 45-bit data. This calls for data optimization along with enlarged search space. This is achieved in PHsClone due to four parallel processes and more genetic operators. - For 2-bit comparator, the OIMGA performs better as the only one function ‘larger’ got evolved out of two. It needs five nodes (60 bits) out of ten nodes (120 bits). The rest node data bits are available as spare. The OIMGA performs better when there are many spare nodes available, so the invalid nodes can be rejected.

The proposed customized algorithm is where the parallel processes are executing, and more genetic operators are available. For small data sizes, Parallel HsClone performs better, while for large data sizes, OIMGA with spare nodes can give faster convergence. The limitation with PHsClone is the more hardware extensive due to parallel processes and associated registers. Further analysis is required to determine an optimum number of parallel processes that balance the resource usage to faster convergence.

VII. Conclusion

The novel embryonic fabric architecture designed with two types of genes, adder, and comparator, is proposed for deep space systems. The fabric includes CGP decoder and self-repair module within each embryonic cell. The centralized BIST is implemented for the embryonic fabric architecture. Within BIST, LFSR for random number generator is designed and run for fixed clock cycles. The response analyser is implemented using MISR and designed separately for adder and comparator cells. The transient error is simulated for ’stuck at zero’ in adder cells. The fault detection is carried out through BIST, and memory scrubbing got executed. After the memory scrubbing, the BIST module is re-initiated to verify the effect. The simulation results are verified for the implementation of Self-repair through scrubbing. The configuration data for adder and comparator in CGP format is evolved using customized GAs. The performance of two GAs; PHsClone and OIMGA is compared for the implemented circuits.

VIII. Scope for Future Work

The fabric design must be modified to include sequential circuits also. The BIST approach must be upgraded in case of a permanent error in the cells. In the case of permanent error, the faulty cell will become transparent, so the routing is to be modified. The signal routing will be updated to use a spare cell in place of the faulty cell. The case of fault that occurs within the BIST module is also to be considered for fault reliability.

References

[1] X. Zhang, G. Dragffy, A. G. Pipe, N. Gunton and Q. M. Zhu, ”A Reconfigurable Self-healing Embryonic Cell Architecture”, in Proc. of the International Conference on Engineering of Reconfigurable Systems and Algorithms, 2003, pp. 134–140.
[2] G. Martinović and I. Novak, ”A combined architecture of biologically inspired approaches to self-healing in embedded systems”, in Proc. of International Conference on Smart Systems and Technologies, 2017, pp. 17–22, Paper identifier https://doi.org/10.1109/SST.2017.8188663
[3] Y. Shanshan, W. Youren, ”A new self-repairing digital circuit based on embryonic cellular array”, *8th International Conference on Solid-State and Integrated Circuit Technology*, 2006, pp. 1997–1999, Paper identifier https://doi.org/10.1109/ICSICT.2006.306573
[4] Z. Zhang, Y. Wang, ”Method to self-repairing reconfiguration strategy selection of embryonic cellular array on reliability analysis”, in Proc. of the 2014 NASA/ESA Conference on Adaptive Hardware and Systems, 2014, pp. 225–232, Paper identifier https://doi.org/10.1109/AHS.2014.6880181
[5] Z. Zhai, Q. Yao, Y. Xiaoliang, Y. Rui and W. Youren, ”Self-healing strategy for transient fault cell reutilization of embryonic array circuit”, NASA/ESA Conference on Adaptive Hardware and Systems, 2018, pp. 225–232, Paper identifier https://doi.org/10.1109/AHS.2018.8541472

[6] R. Salvador, A. Otero, J. Mora, E. D. La Torre, L. Sekanina and T. Riesgo, ”Fault tolerance analysis and self-healing strategy of autonomous, evolvable hardware systems”, International Conference on Reconfigurable Computing and FPGAs, 2011, pp. 164–169, Paper identifier https://doi.org/10.1109/ReConFig.2011.37

[7] E. Benkhelifa, A. Pipe and A. Tiwari, ”Evolvable embryonics: 2-in-1 approach to self-healing systems”, *Procedia CIRP*, 11, 2013, pp. 394–399, Paper identifier https://doi.org/10.1016/j.procir.2013.07.029

[8] V. Sahni and V. P. Pyara, ”An Embryonic Approach to Reliable Digital Instrumentation Based on Evolvable Hardware”, *IEEE Transactions on Instrumentation and Measurement*, 52(6), 2003, pp. 1696–1702, Paper identifier https://doi.org/10.1109/TIM.2003.818737

[9] K.H. Chong, I.B. Aris, M.A. Sinan and B.M. Hamiruce, ”Digital Circuit Structure Design via Evolutionary Algorithm Method”, *Journal of Applied Sciences*, 7, 2007, pp. 380-385.

[10] E. Benkhelifa, A. Pipe, G. Dragffy and M. Nibouche, ”Towards evolving fault tolerant biologically inspired hardware using evolutionary algorithms”, *IEEE Congress on Evolutionary Computation, Singapore*, 2007, pp. 1548-1554, Paper identifier https://doi.org/10.1109/CEC.2007.4424657

[11] J. F. Miller, ”Cartesian Genetic Programming. Natural Computing Series”, 43, 2011, Paper identifier https://doi.org/10.1007/978-3-642-17310-3

[12] G. Malhotra, V. Lekshmi, S. Sudhakar and S. Udupa, ”Implementation of threshold comparator using Cartesian genetic programming on embryonic fabric”, *Advances in Intelligent Systems and Computing*, 939, 2019, pp. 93–102.

[13] E. Stomeo, T. Kalganova and C. Lambert, ”A novel genetic algorithm for evolvable hardware”, *IEEE Congress on Evolutionary Computation*, 2006, pp. 134–141, Paper identifier https://doi.org/10.1109/CEC.2006.1688300

[14] Lucian Prodan, Gianluca Tempesti, Daniel Mange and André Stauffer, ”Embryonics: electronic stem cells”, In Proc. of the eighth international conference on Artificial life, 2003, pp. 101–105.

[15] D. Mange, A. Stauffer and G. Tempesti, ”Embryonics: A macroscopic view of the cellular architecture”, *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1478, 1998, pp. 174–184, Paper identifier https://doi.org/10.1007/BFb0057619

[16] Yann Thoma, Gianluca Tempesti and Eduardo Sanchez, ”POEtic: An Electronic Tissue for Bio-Inspired Cellular Applications”, *Biosystems*, vol. 76, 1-3 (2004).

[17] A. Stauffer, Daniel Mange, and Joel Rossier, ”Design of Self-organizing Bio-inspired Systems”, Second NASA/ESA Conference on Adaptive Hardware and Systems, 2007.

[18] M. R. Boesen and J. Madsen, ”eDNA: A bio-inspired reconfigurable hardware cell architecture supporting self-organisation and self-healing”, NASA/ESA Conference on Adaptive Hardware and Systems, 2009, pp. 147–154, Paper identifier https://doi.org/10.1109/AHS.2009.22

[19] C.E. Stroud, ”A Designer’s Guide to Built-in Self-Test”, *Springer*, 2002.

[20] Gayatri Malhotra, Joachim Becker and Maurits Ortmanns, ”Novel Field Programmable Embryonic Cell for Adder and Multiplier”, 9th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME-2013), June 2013.

[21] Z. Zhang and Y. Wang, ”Method to self-repairing reconfiguration strategy selection of embryonic cellular array on reliability analysis”, In Proc. of the 2014 NASA/ESA Conference on Adaptive Hardware and Systems, 2014, pp. 225–232, Paper identifier https://doi.org/10.1109/AHS.2014.6880181

[22] M. F. Torquato and M. A. C. Fernandes, ”High-Performance Parallel Implementation of Genetic Algorithm on FPGA”, Circuits, Systems, and Signal Processing, 38(9), 2019, pp. 4014–4039, Paper identifier https://doi.org/10.1007/s00034-019-01037-w

[23] Z. Zhu, D. J. Mulvaney and V. A. Chouliaras, ”Hardware implementation of a novel genetic algorithm. Neurocomputing”, 71(1–3), 2007, pp. 95–106, Paper identifier https://doi.org/10.1016/j.neucom.2006.11.031

[24] A. AL-Marakeby, ”FPGA on FPGA: Implementation of Fine-grained Parallel Genetic Algorithm on Field Programmable Gate Array”, International Journal of Computer Applications, 80(6), 2013, pp. 29–32, Paper identifier https://doi.org/10.5120/13867-1725