Organic Embedded Architecture for Sustainable FPGA Soft-Core Processors

Kening Zhang, Navid Khoshavi, Jaafar M. Alghazo, and Ronald F. DeMara
Department of Engineering and Computer Science
University of Central Florida
Orlando, FL, USA 32816-2362
e-mail: demara@mail.ucf.edu

Keywords: Self-configuration, self-healing, self-adaptation, reliability, autonomous system, robustness, organic computing, observer/controller architecture, hardware agents, emergent behavior.

ABSTRACT

Mission-critical systems require increasing capability for fault handling and self-adaptation as their system complexities and inter-dependencies increase. *Organic Computing (OC)* architectures utilize biologically-inspired *self-x* properties which include self-configuration, self-reorganization, and self-healing which comprise the focus of this paper. To provide OC architectures with sufficient capability for exhibiting self-adaptive behavior, reconfigurable logic devices offer a suitable hardware platform. SRAM-based Field Programmable Gate Array (FPGA) logic devices can realize self-adaptation within their reconfigurable logic fabric using Evolvable Hardware techniques based on crossover, mutation, and iterative selection with intrinsic fitness assessment of the underlying hardware resources. In this paper, a dual-layer Organic Computing architecture called the *Organic Embedded System (OES)* is prototyped on a Xilinx FPGA reconfigurable fabric and assessed for maintainability metrics of *completeness of repair*, *repair time*, and *degraded throughput* during the repair phase. The approach used extends a widely known generic OC platform consisting of two layers: the *Functional Layer* and the *Autonomic Layer*. The Autonomic layer contains *Autonomic Elements (AEs)* that are responsible for correct operation of the corresponding *Functional Elements (FEs)* present on the Functional Layer.

Innovations include autonomously degraded online throughput during regeneration, spare configuration aging and outlier driven repair assessment, and a uniform design for AEs despite the fact that they monitor different types of FEs. Using the OES approach; a malfunctioning or faulty AE among the population can be distinguished by its discrepant performance. The OES approach is implemented using high-level Hardware Description Language (HDL) which directs a Supervisor Element (SE) to function as a fault management unit through the collection of AE information. Experimental results show that the OES Autonomic Layer demonstrates 100% faulty component isolation for both FEs and AEs with randomly injected single faults. Using logic circuits from the MCNC-91 benchmark test set, throughput during repair phases averaged 75.05%, 82.21%, and 65.21% for the `z4ml` (2-bit adder), `cm85a` (high fan-in combinational logic), and `cm138a` (balanced I/O combinational logic) circuits respectively under stated conditions.

1 ORGANIC EMBEDDED SYSTEM (OES) ARCHITECTURE

Trends in architecture and investigations for run-time adaptive systems have begun to explore the possibility of autonomous run-time reconfiguration for increased reliability and power awareness. The Organic Embedded System (OES) architecture developed herein utilizes Evolvable Hardware [1] approaches based on a variety of genetic techniques.

Requirements are partitioned into two logical layers. The functional layer houses the Intellectual Property (IP) core component or FEs. FEs can be any functional element from general purpose CPUs, memories, on-chip busses, special purpose processing units or network interfaces. The Autonomic layer consists of AEs and an interconnect structure among the AEs. The following properties are inherent:

1. FEs and AEs both reside on *two distinct layers* with an interconnection structure between them.
2. The AEs and FEs can either be realized in *hardware*, *software*, or through hardware/software *co-design*,
3. The AE layer should supervise the functionality of the FE elements in the FE layer while requiring *no application-specific algorithms on the AE layer* to be developed to realize this fault-tolerant functionality.
4. The Observer/Controller architecture includes an *Autonomic Supervisor (AS)* element which does not have a counterpart to evaluate if the AS is fault-free. Therefore, in the OES design we *address reducing the vulnerability of the AS* by emphasizing its simplicity as part of our approach.

As shown in Figure 1, the separate layers of the OC architecture implemented in the OES are mapped to alternating vertical columns of logic slices on the Xilinx Virtex II Pro FPGA device. This column-oriented structure permits the architecture to take advantage of Xilinx partial reconfiguration technology to manipulate the bitstreams of either the AEs or FEs configurations for the fault recovery.

Even a small size system composed of large numbers of various functionalities will need to occupy differing amounts of physical resources for each FE as well as require a different number of I/O resources. Thus as shown in Figure 1, each FE is placed in single column or multiple contiguous columns of the FPGA chip. The number of columns for each FE can be allocated as necessary according to the area requirements of the system being designed. Xilinx bus macros are used to provide re-locatable reconfigurable interfaces between FEs and AEs, AEs and the AS, and between FEs via a user-defined interconnection network module [2].

Furthermore, controllability and maintainability demands can become substantial because of the overhead associated with scheduling, coordinating, and communication among the large number of interacting components. In order to evenly distribute this burden, the decentralization of the Observer/Controller components is proposed. In OES, the AEs reduce the demand for centralized controllability as shown in Figure 2. It consists of a Concurrent Error Detection (CED) [3] unit to collect and evaluate outputs from 2 FEs, a Checksum for AE fault detection (which are checked against Stored Checksum values) and an Actuator. Each AE will monitor the operation of the corresponding FE component, evaluate the performance of the FE and render a local assessment on the failure status of the FE. An important architectural property of the OES is that all AE components are identical in structure despite the fact that they monitor different types of FEs. The homogeneous characteristics of the AE components deliver a uniform-behavior property which is leveraged to realize a consensus-based fault-detection methodology.

In addition to the AE and FE layers, the OES architecture also contains an AS. The AS implements the consensus mechanism to evaluate the behavior of all the AEs in the system and distinguish the abnormal individuals whose behavior may be distinguished from the rest of the members in the AE population. *Genetic Algorithm (GA)* operators are implemented here to achieve fault recovery. All other factors being equal, the likelihood of a local permanent fault of any component is proportional to the device area required for its realization. The AS is kept as simple as possible to reduce its complexity and reduce its likelihood of experiencing a fault proportionally.

2 SYSTEM OPERATION

The OES architecture supports several operational phases of interaction between the FEs, AEs, and AS. The initial state of all components is fault-free. Figure 2 shows a diagram of the flow of operations in the OES architecture as described below.

System initialization phase: *FE Initialization step:* Three functionally identical FE configurations labeled FE, FE, and S-FE are instantiated on different physical locations. Initially, only the two FEs are active and the S-FE acts a cold spare FE. The FEs supply the output for each set of inputs applied in parallel in a Concurrent Error Detection configuration to the AE for the fault detection.

Compute Checksum step: Each AE contains a Checksum Component which uses the stored outputs of the AE along with the small finite number of possible input combinations to the Evaluator and Actuator to populate the Check Sum Lookup Table (CS-LUT) in the AE. This feature in the AE will be utilized to detect if the current AE is faulty in a consensus-based approach [4]. For the benchmark circuit selected a carry and sum, the CS-LUT required a 16-entry x 4-bit memory.

FE fault detection/recovery and AE monitoring phase: As depicted in Figure 2, at runtime the inputs destined to the FE are applied to both active ones under a CED strategy. After allowing for FE inputs propagation time through the AE, the expected output will be supplied to AE-CED for the fault `CED module and any discrepancy between the two values will indicate that a fault has occurred in either of one the FEs or the AE-CED itself. Further detection will be required to distinguish which of the two is faulty.

If the AE component is identified as operational then the fault must of occurred in this output will be discarded. Next, the control will branch to a fault identification phase which will activate the cold standby FE and construct a temporary Triple Modular Redundancy (TMR) system which can articulate the faulty FE under the new supplied external input. Furthermore, the actuator will initiate a repair cycle which may require automatic evolutionary repair of the identified faulty FE. The faulty FE will be set as standby-under-repair and the AE-CED will return to receive the remaining two active FEs' inputs. The decision-making procedure causes at least one throughput-delay penalty.

The AE supports two exclusive modes: FE monitor mode and AE self-repair mode. Whenever the AS identifies that an AE is faulty, the AE will relinquish observation of its FE and focus on its own self-repair. Under FE monitor state, AE will keep observing the FE behavior and issue control instructions through the actuator.

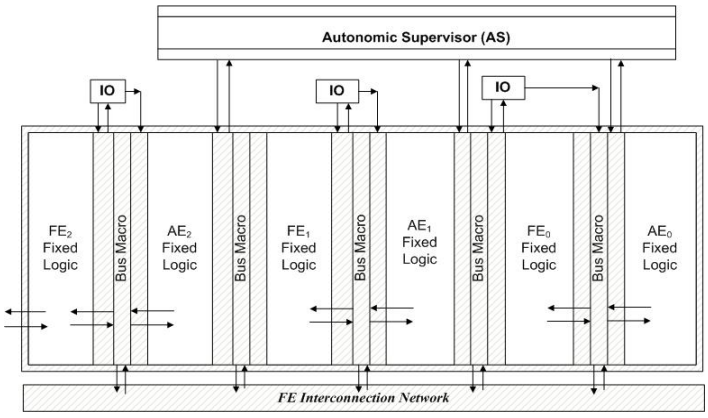

Figure 1- Column-oriented OES on Xilinx Virtex Series

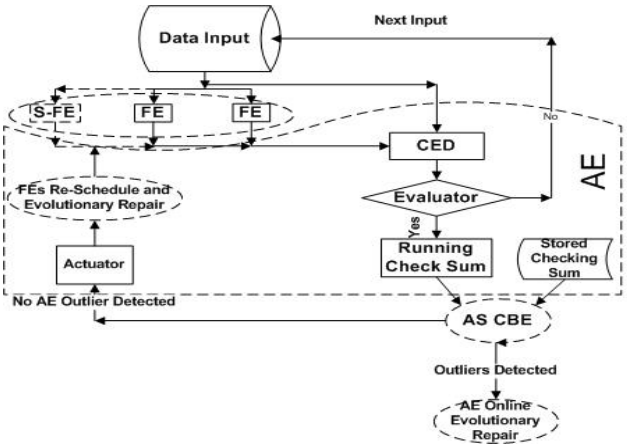

Figure 2- OES FE and AE Failure Detection Procedure

The recovery procedure entails the use of alternative designs for the AE that have identical functionality but distinct physical resources. GA operation will manipulate the representation of the AE bitstream and evaluate each new generated offspring until the fault is occluded. This off-line evolution may be time-consuming and halt the faulty FE operation, yet. However, it is an entirely autonomic repair without any human intervention and the faulty AE does not affect the current FE execution.

AE fault detection phase: Three possible faulty scenarios may occur inside the AE:

1. A fault may exist in the CED, Actuator, or Evaluator,
2. A fault may exist in Check Sum component, or
3. A fault may exist in the Stored CS-LUT.

All three scenarios are detected under the proposed approach. To detect if the CED, actuator, or evaluator are faulty, we apply the outputs of the three components to the checksum circuit while the inputs of the three components are simultaneously applied to a parallel search circuit that will locate the input combination and its corresponding output in the CS-LUT. By the time the inputs propagate through the checksum circuit, the output from CS-LUT will be available, the two values are then compared and any discrepancy will detect a fault. The second and third scenarios will also generate a discrepancy between the Checksum component and Stored Checksum component.

Furthermore, the approach reveals that the design would operate even under multiple faults as long as multiple faults generate the same faulty behaviors among different sub-components of the AE. This is impossible in this design because each sub-component is implemented with distinct logic/arithmetic functionality. Nonetheless, we have observed in experiments that GA mutation operator applied to AE unit and using cell swap can sometimes self-heal the AE unit even if more than one of its components is faulty.

CBE evaluation process and AE fault recovery Phase: A Consensus-Based Evaluation (CBE) approach is utilized for assessing the performance of individuals based on broad consensus of the AE population instead of a conventional fitness function defined for GAs. Adoption of CBE enables information contained in the population to not only enrich the evolutionary process, but also support fault detection and isolation. The AS component will collect all of AEs outputs and distinguish the abnormal individuals from the population instead of using traditional threshold, the population information will assist outlier identification and fault recovery.

The automatic fault recovery utilizes the homogeneous characteristic of the AE components; each fault impact on any AE can mirror the health of the AE configuration which may reveal some inherent fault immunity property. Even though each AE occupies different physical locations, they are implemented using identical logic functionality which can be used to overcome physical failure.

3 EVOLUTIONARY PROCESS OF FE AND AE

The basic principle of the evolutionary recovery approach advocated in this paper is to maintain the integrity of the configurations' functionality throughout the evolutionary process. Instead of exploring completely random search space, the proposed approach will move outwards from the original design space by trying permutations of the existing logic and interconnection for occluding the physical failure. The reason is that feasible repairs may be expected to require less computational complexity than realizing a completely new design. The evolutionary process generates improved genotype bitstrings which can be used to configure the logic fabric within a pre-defined genotype to phenotype mapping [5]. The phenotype is defined as the FE or AE circuit manifestation of a particular genotype. The physical genotype is based on the specific configuration bitstream which is generated by the Xilinx synthesis tools and is readable by the FPGAs in that device family. In order to reflect the identical logic functionality, the logical chromosome of the AE will be uniform despite the physical configuration.

The logical genotype in this paper is an LUT vector which contains logic and physical ordering information plus the configuration I/O information. As shown in Figure 3, the LUT is the basic building block of the genotype and contains both logic ordering numbers (Logic #) and physical ordering numbers (Col # and Row #) which identify both physical location and the functionality sequence of the LUT. Based on the genotype, three genetic operators are developed in this paper for manipulation, each of which emphasize a different aspect of information for the configuration and fault recovery process: 1) Mutation Operator: the objects mutated are input interconnections of LUTs; 2) Cell-Swap Operator: The Cell-Swap operator is swaps two distinct LUTs' blocks and meanwhile maintaining correct the logic order and functionalities in the genotype; 3) Partial Match Crossover Operator: Under *Partial Match Crossover (PMX)*, two configurations are aligned, and a crossover site is picked uniformly at random on the boundary of LUTs in genotype.

4 EXPERIMENTAL CONFIGURATION

FE and AE failure coverage: In the experiments, coverage and resolution of faults in the FE and faults in the AE are evaluated. The FE fault-handling experiments inject a stuck-at-zero or stuck-at-one fault at one of the FE's LUT input pins and the resolution process proceeds. The AE fault-handling experiment utilizes a CBE-based to approach detect the faulty AE in the population. Once the fault is detected, the AS generates a new population for identified AEs, reconfigures them on the logic fabric sequentially in order to evaluate their correctness. After all the configurations are evaluated, CBE keeps detecting faults in the AE under repair, until the number of newly created configuration evaluations reach the Evaluation Window, E_w. During the AE repair, the FE will reside on the chip and generate output even under fault impact conditions. The AE units are said to be functionally identical, but physically distinct due to the fact that they all contain the same functional elements with a constraint of identical number of I/O pins. This implies that as long as the AE is designed for the largest output word-width output by any FE, then all of the

FEs can differ in function and even differ in output word-width by just tying any unused input pins of the AE to ground without any loss of generality.

Single vs. multiple fault coverage: In order to determine the fault handling mechanism in the proposed system, two different fault models: *Single-Failure Model* and *Multiple-Failure Model* are introduced. If the *Single-Failure Model* is applied to the proposed system then the fault will be located either in a FE or an AE component, but not both simultaneously. Therefore the analysis of the evolutionary recovery operation will only focus on the faulty component without considering the other component's status. Whenever the AE component is undergoing an observable fault impact, the system will lose the monitoring functionality of the corresponding FE component. However, under *Single-Failure Model*, another FE component will be fault-free and maintain data throughput without error during that time period.

Alternatively, if the FE component is under the impact of the fault then the AE component notifies the AS that the incorrect output was generated from the output of the FE component. Even when the FE component is under fault impact, the cold spare can provide a ready replacement for reconfiguration. Under the FE fault case there is no unavailability once the switch to TMR to identify the failed resource is completed. The failed FE can be repaired in the background via the GA as a refurbished CED mode has been restored.

For a single FE fault, the system availability, A_{SF}, is given by Equation (1). Let the number of correct behaviors of the FE that have been observed during the evolutionary recovery phase be denoted by F_c while the number of errant or discrepant behaviors of the FE be denoted by F_e. The quantity *1* represents the number of faulty outputs, i.e. exactly one output required to detect the fault during the original CED configuration. The coefficient *2* is the number of the reconfigurations required, i.e. one from CED to TMR, and one back from TMR to CED. The quantity β represents reconfiguration time expressed in the same time units as the computation time units, yielding:

$$A_{SF} = \begin{cases} 100\% & \text{if AE under Single fault} \\ \dfrac{F_c}{F_c + 1 + 2\beta} & \text{if FE under Single fault} \end{cases} \quad (1)$$

The next scenario represents the *Multiple-Failure Model*. If multiple faults occur in only either the FE component or the AE component exclusively, it yields the same behavior as the

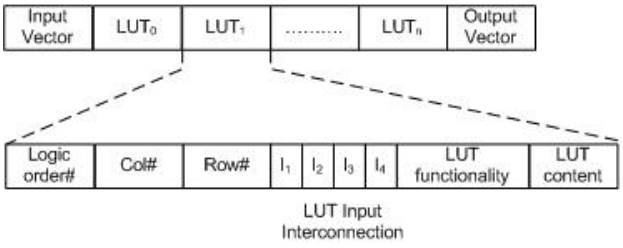

Figure 3- Genotype chromosomes of GA operation

Single-Failure Model case because no matter how many faults occur, the unit-under-test is the entire FE or AE itself during the fault detection step. On the other hand, if the AE and FE are under the impact of faults simultaneously, then the system will keep the FE online executing correct FE behavior if possible by introducing the S-FE.

If deployment of the S-FE is unsuccessful then group information will be used to repair the AE first. As long as the AE returns to normal, the FEs will be recruited into repair process. So under this strategy system still can maintain graceful degradation capability under the multiple fault impacts. The quantities F_c, F_e and β in the second line of equation (2) have the same definition as equation (1), and the quantities F_{c1} and F_{e1} stands for the correct and faulty output number of the FE during the AE repair period, the (F_{c2}, F_{e2}) stands for the correct and faulty output number during the FE repair period and the n is number of reconfigurations of the FE. Hence, the system availability under multiple faults, A_{MF}, is given by:

$$A_{MF} = \begin{cases} 100\% & \text{if AE under multiple faults} \\ \dfrac{F_c}{F_c + 1 + 2\cdot\beta} & \text{if FE under multiple faults} \\ \dfrac{F_{c1} + F_{c2}}{F_{c1} + F_{e1} + F_{c2} + F_{e2} + n\cdot\beta} & \text{if AE \& FE both under multiple faults} \end{cases} \quad (2)$$

5 HARDWARE PROTOTYPE

The case study was implemented on the Xilinx Virtex II Pro as proof of concept. Only a small number of resources are utilized for the AE and FE. The OES architecture in this case study consisting of a Full Adder FE unit with all of the elements in the AE Unit is realized using HDL implementation on the Xilinx Virtex II Pro FPGA using the GNAT library along with the MRRA framework and JTAG reconfiguration interface.

The Evaluator consists of XOR gates to check for any discrepancy between the FE units. There are three FE units of which only two are active during runtime, the third FE is a standby, i.e. S-FE, and will only become activated once a discrepancy is detected on the FE elements. Once a discrepancy is detected, the switching logic will be used to activate the standby FE. TMR will be used in this case for the Evaluation Window during which Genetic Operators will be used to repair the faulty FE individual. Once evolution achieves repair, the repaired FE will now become the S-FE. This process is invoked each time a FE discrepancy is detected.

6 RESULTS AND ANALYSIS

Three circuits were evaluated using the OES architecture, all of which are specified in Table 1 from the MCNC-91 benchmark [6]. The experiments implemented test the fault repair process on both the FE and AE components

Table 1- MCNC-91 Benchmark Results

Circuit Name	Circuit Function	Inputs	Outputs	Approximate Gates
z4ml	2-bit Add	7	4	20
cm85a	Logic	11	3	38
cm138a	Logic	6	8	17

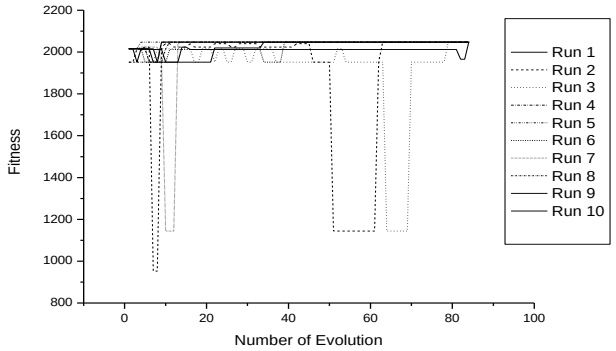

Figure 4- cm85a FE Evolutionary Recovery without CBE.

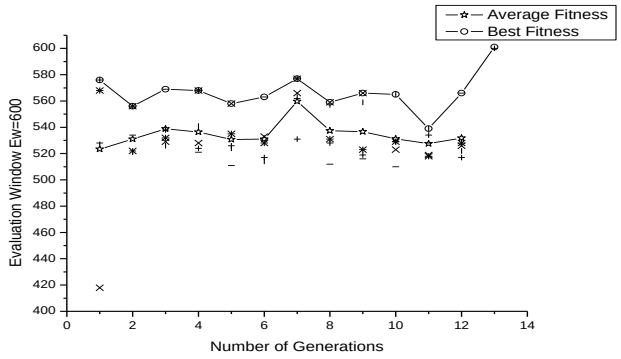

Figure 5- AE Evolutionary Repair for cm85a.

simultaneously. In the FE fault recovery process, only Mutation and Cell-Swap operators are applied and the unit evolved to the fault-free state without utilization of population information. A fault was also induced on an input of LUT within the AE unit. During the AE fault recovery process, all three genetic operators along with CBE were applied to evolve the AE unit to a refurbished status. The use of CBE along with GA operators proved to provide a large benefit in terms of number of repair iterations compared to conventional offline GA-based design-from-scratch-approaches. Each experiment was executed for 10 runs on each circuit. The GA parameters were set as Mutation Rate=0.5, Cell-Swap rate=0.5 and Crossover rate=0.5 in all of the runs. The population size for AE is five and FE there are 2 active and one spare. The GA tournament selection rate was selected to be 2.

The fitness obtained for the cm85a circuit when a stuck-at-zero fault was injected at 48 different locations. Specifically the circuit was synthesized using Xilinx ISE (Integrated Software Environment) to occupy 12 LUTs in which each LUT had 4 inputs, yielding 48 exclusive failure locations. The fitness of each `cm85a` circuit under each test scenario ranges from zero outputs correct up through a maximum of 2^{11}=2048 outputs correct because `cm85a` has 11 inputs. Fault location 25, 31, 41, 47 exhibited fitness less than 1250 while other locations had fitness near 2000.

The `z4ml` circuit experimental results are listed in Table 2 which lists the system Availability during the repair phase for R_{AE}=14.1% and R_{FE}=25%. It is important to note that the system Availability outside the repair phase is by definition 100%. The column *n* denotes the measured number of reconfigurations for either the AE or the FE during the repair process during each test run. In last three columns, we assume the β is equal to 100, 1000, and 10000 respectively, to anticipate the system performance under different reconfiguration to computation ratios. The result presented shows that even spanning 3 orders of magnitudes of difference, the system performance can still be acceptable for some certain circumstances. When β=1000, the average system availability is 75.05% for the `z4ml` circuit and 82.21% for the `cm85a` and 65.21% for `cm138a`. All, three may not degrade gracefully but they can maintain partial functionality under the fault impact. The values of the redundancy for both FE (R_{FE}) and AE (R_{AE}) are calculated based on the ratio of unused LUT inputs to the total number of LUTs inputs used by both the AE and FE designs respectively.

Figure 4 shows the evolutionary process for the `cm85a` circuit which has 11 inputs and 3 outputs and a maximum fitness of 2^{11}=2048. During the repair process, only mutation and cell-swap operators are implemented because there is only a single instance of FE under repair. To explain the concept behind Figure 4 and why the fitness and evolution behavior differs from conventional Genetic Algorithms(GA) is because the functional element is always predefined at design time. The GA in our system explores a limited search space to identify alternate physical resources to bypass a faulty input or faulty LUTs. Since the functionality of the FE is already

Table 2- `z4ml` Circuit Experiment Results.

	n	Fault-Free AE output		Fault-Impact AE		Avail. During Rep. $\beta=10^3$	Avail. During Rep. $\beta=10^4$	During Rep. $\beta=10^5$
		F_{c1}	F_{e1}	F_{c2}	F_{e2}			
AE	15	20856		2003		80.45%	87.41%	46.67%
FE	2	19979	2858	22	1			
AE	7	9403		917		72.99%	41.87%	29.52%
FE	2	8914	1302	10	1			
AE	17	24899		2215		81.48%	101.21%	50.31%
FE	2	23664	3380	8	1			
AE	11	14586		1702		78.10%	64.73%	39.31%
FE	2	14234	1992	8	1			
AE	11	15474		1375		78.53%	67.00%	40.12%
FE	2	14764	2036	2	1			
AE	3	3991		278		59.41%	17.98%	15.24%
FE	2	3685	521	6	1			
AE	7	9612		767		73.10%	41.95%	29.56%
FE	2	8929	1287	4	1			
AE	5	6880		444		68.78%	30.36%	23.30%
FE	2	6334	877	7	1			
AE	17	23201		2084		81.01%	95.23%	48.78%
FE	2	22068	3173	2	1			
AE	9	12622		1866		76.68%	57.69%	36.59%
FE	2	12592	1831	6	1			
Average System Availability During Repair						75.05%	60.54%	35.94%

predefined then the search space is limited to identifying distinct physical resources for repairing the existing fault to retain functionality. Hence, the GA is not exploring new designs, but maintaining functionality of the current design.

Also in Fig 7, the `cm138a` AE evolutionary process is shown. The population information helps repair the circuit. The difference of this evolutionary process with the traditional GA is that the configurations are all correct, but the physical fault in the hardware resources that they utilize produces the unexpected behavior of the circuit. Instead of generating new design, the evolution process only permutes the current design using different input line combination or different logic occupied physical resource. When comparing Figure 4 and Figure 5, the different benefit of the CBE-GA approach is seen relative to random GA operation.

7 CONCLUSION

In this paper, we developed Organic Embedded System (OES) architecture for sustainable performance of organic reconfigurable embedded applications. The Architecture was developed using SRAM-based Field Programmable Gate Arrays (FPGAs), an Organic Computing (OC) observer/controller organization, and regeneration with Genetic Operators. Other innovations in this paper included online availability during regeneration, aging and outlier driven repair assessment, and a uniform design for Autonomic Elements (AEs).

Using logic circuits from the MCNC-91 benchmark set, availability during repair phase averaged 75.05%, 82.21%, and 65.21% for the `z4ml`, `cm85a`, and `cm138a` circuits respectively under stated conditions. The developed CBE-GA approach is shown to outperform a Genetic Algorithm using a conventional absolute fitness function. Each experiment was executed for 10 runs on each benchmark circuit. The GA parameters were set as PMX rate=0.5 and LCS rate=0.5 with various mutation rates. The population size for AEs was selected to be five instances and there were 2 active FEs and one spare FE which were evaluated with tournament selection of size two per generation. The CBE-GA approach is shown to improve completeness of repair, repair time, and partial throughput during the repair phase through use of population information. System availability during the repair phase is shown to range from 35.9% to 80.4% under reconfiguration to computation ratios ranging from 10^3 to 10^5 clock cycles. These results demonstrate the feasibility of an OES architecture for aerospace missions with limited human intervention which is capable of recovery without taking the device out of service during the repair process.

REFERENCES

1. X. Yao, and T. Higuchi, “Promises and Challenges of Evolvable Hardware” *IEEE Transaction on Systems, Man, and Cybernetics- Part C: Applications and Reviews*, 1999.
2. K. Zhang, G. Bedette, R. F. DeMara, "Triple Modular Redundancy with Standby (TMRSB) Supporting Dynamic Resource Reconfiguration," *Proceedings of IEEE AUTOTESTCON*, September 18-21, 2006.
3. J. Khakbaz, and E. J. McCluskey, “Concurrent Error Detection and Testing for Large PLA's” *Trans. on Electron Devices, vol. ED-29, no. 4*, pp. 756-764, 1982.
4. K. Zhang, R. F. DeMara, and C. A. Sharma "Consensus-based evaluation for fault isolation and on-line evolutionary regeneration," *International Conference in Evolvable Systems (ICES'05)*, pp. 12 -24, Barcelona, Spain, September 12 - 14, 2005.
5. P. C. Haddow and V. K. Gautam, "DNA tiling for digital evolvable hardware", *IEEE International Conference on Evolvable Systems*, 16-19 April, 2013.
6. S. Yang, “Logic synthesis and optimization benchmarks user guide version 3.0”, Technical Report, Microelectronics Center of North Carolina, Jan. 1991.

BIOGRAPHIES

Kening Zhang, PhD
Baidu, Inc.
Building D, No. 1 Shangdi East Road, Haidian District
Beijing-10085 - China

e-mail: zhangkening@baidu.com

Kening Zhang received his Ph.D. degree in Computer Engineering from the University of Central Florida 2008. He is a Senior Manager at Baidu, Inc. in Beijing City, China.

Navid Khoshavi, PhD student
University of Central Florida
4000 Central Florida Blvd
Orlando, FL, 32826-2362 - USA

e-mail: navid.khoshavi@knights.ucf.edu

Navid Khoshavi is currently a Ph.D. student in Computer Engineering at the University of Central Florida. His research interests are in Fault-Tolerant Computer Systems, Evolvable Hardware, and Low Power Design.

Jaafar M. Alghazo, PhD
Dean of Continuing Education and Community Service
Prince Mohammad Bin Fahd University - Saudi Arabia

e-mail: jghazo@pmu.edu.sa

Jaafar M. Alghazo received his Ph.D. degree in Computer Engineering from the University of Illinois Carbondale in 2004. He is Dean of Dean of Continuing Education and Community Service at Prince Mohammad Bin Fahd University in Khobar, Saudi Arabia.

Ronald F. DeMara, PhD, Professor
University of Central Florida
4000 Central Florida Blvd
Orlando, FL, 32826-2362 - USA

e-mail: Ronald.Demara@ucf.edu

Ronald F. DeMara is a Professor of Electrical and Computer Engineering at the University of Central Florida. His research interests are in Adaptive Computer Architectures, Evolvable Hardware, and Reconfigurable Logic Devices.