FPGA Self-Repair using an Organic Embedded System Architecture

Kening Zhang, *Student Member, IEEE*, Jaafar Alghazo, and Ronald F. DeMara, *Senior Member, IEEE*
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816-2362
demara@mail.ucf.edu

Abstract

An Organic Embedded System (OES) architecture is developed for sustainable performance using SRAM-based Field Programmable Gate Arrays (FPGAs), an Organic Computing (OC) observer/controller organization, and regeneration with Genetic Operators. Innovations include availability during online regeneration, aging and outlier driven repair assessment, and a uniform design for Autonomic Elements (AEs) despite the fact that they monitor different types of Functional Elements (FEs). Using logic circuits from the MCNC-91 benchmark set, availability during repair phases averaged 75.05%, 82.21%, and 65.21% for the z4ml, cm85a, and cm138a circuits respectively under stated conditions. In addition to simulation, the proposed OES architecture synthesized from HDL was prototyped on Xilinx Virtex II Pro FPGA device supporting partial reconfiguration to demonstrate the feasibility of OC approaches for intrinsic regeneration of the selected circuit.

This research was supported in part by NASA Intelligent Systems NRA Contract NNA04CL07A.

1. INTRODUCTION

Electronic systems require increasing capability for fault tolerance and self-adaptation as their system complexities and inter-dependencies continue to increase. The realizations of systems that are capable of exhibiting such adaptive behaviors constitute the vision sought by *Organic Computing (OC)* [SCHMECK 2005]. OC *self-x* properties include self-configuration, self-reorganization, and self-healing [LIPSA et. al 2005; SCHMECK 2005] which comprise the focus of this paper. Ideally, these objectives are maintained in an autonomous fashion, yet sufficiently constrained to avoid undesirable emergent behaviors. In particular, OC systems rely on self-organization to respond to internal imbalances and changing environmental conditions using *Observer/Controller* architecture [AVAZIENIS 1997].

To provide OC architectures with sufficient capability for exhibiting self-adaptive behavior, reconfigurable logic devices offer an attractive hardware platform [SCHMECK 2005]. SRAM-based Field Programmable Gate Arrays (FPGAs) logic devices can realize self-adaptation within their reconfigurable logic fabric using Evolvable Hardware techniques. Since evolution is employed, the *Observer/Controller* has the task of detecting internal/external errors and well as initiating reconfiguration when necessary.

A widely known generic OC platform called the Autonomous System-on-a-Chip (ASoC) architecture proposed in [LIPSA et. al 2005] is depicted in Fig. 1. The ASoC platform consists 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. Every FE such as CPU, RAM, and Network Interface has a counterpart Monitor, Evaluator, and Actuator component within the Autonomic Layer. The Autonomic Layer also consists of an *Autonomic Supervisor (AS)* that has no counterpart on the Functional Layer. The AS is responsible for the correct functionality of all AEs on the Autonomic Layer.

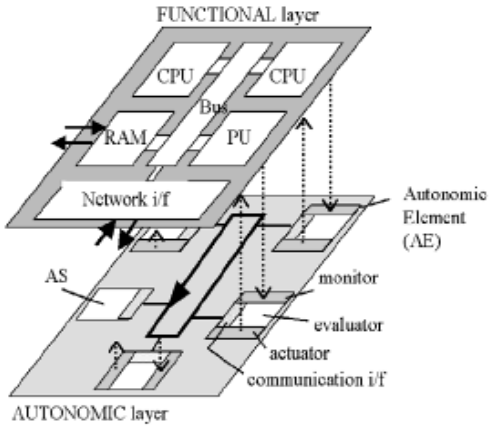

Fig. 1. Dual-Layer ASoC proposed by Lipsa [LIPSA et al 2005]

The Dual-Layer ASoC design approach in Fig. 1 is extended herein to provide fault coverage at both the Functional Layer and Autonomic Layer. This is achieved by assessing consensus among elements in a two-fold approach. Consensus is used first to realize failure detection. Once identified, consensus provides an organic method for fitness evaluation of competing alternatives during evolution providing a self-regulating approach to fault resolution. The measured performance is analyzed as an integrated OC system for self-configuration and self-healing. This demonstrates a generic OC architecture that can detect faults and refurbish itself while still providing a

degraded level of valid throughput even during the refurbishment period.

2. PREVIOUS WORK

Related works in the literature have explored techniques useful for development of an OC system from various theoretical and practical perspectives. For example, in [BERNAUER et. al 2006] a runtime reliability evaluation of ASoC architectures was addressed. The objective was to design SoCs that can tolerate faults by introducing dynamic reliability, power management, and security tradeoffs, as well as adaptation to environmental changes and unpredictable failure scenarios. Under these conditions, a theoretical model for calculating error probability during run-time is presented. A related fault model in [BOUAJILA et. al 2006] concentrated on transient and timing faults caused by ionizing radiation or variations at the technology or device level. The C-program simulations executed on Leon-2 processor code resulted in a penalty of two cycles for the detection and correction of an error in the processor's pipeline. Work has also been conducted on prototyping platforms capable of support OC architectures. For instance, the *Egret* system provides a platform for reconfigurable SoC's supporting applications such as OC [BERGMANN and WILLIAMS 2003]. The design objectives of Egret were to provide a platform to rapidly prototype embedded reconfigurable applications and a straightforward path to commercialization of prototyped designs. The platform consisted of modular functional elements that can be interconnected to design an embedded application for reconfigurable logic.

In more a general study, identification of SoC system requirements for detecting faults and handling the faulty components is addressed in [BOUAJILA et. al 2006]. Fault tolerant error detection techniques are classified into three groups: hardware redundancy, information redundancy, and time redundancy. The three techniques and their combination were surveyed on an Autonomous SoC design consisting of the two layers: the Functional Layer and Autonomic Layer. In this theoretical framework, it was suggested that the Autonomic SoC would need a well-tailored AE layer which would cope with malfunctioning subcomponents. The simulation consisted of a paradigm with priori knowledge about the system's behavior when an error occurs and examines setting a threshold for errors that can occur before the system goes into self-organizing mode [BOUAJILA et. al 2006].

In order for an autonomous system to invoke its self-healing mode, it must be able on its own to detect errors during run-time [BECKER and HÜBNER 2006]. Reconfiguration and detection techniques explored include *scrubbing* which is the continuous reconfiguration of the bitstream to refresh the stored configuration [CARMICHAEL et. al 2000], Built-In-Self-Test (BIST) techniques [STROUD et. al 2004], on-chip hardware test benches [SEKAR et. al 2000] and Triple Modular Redundancy (TMR) extensions [ZHANG et. al 2006]. Decentralized approaches to Observer/Controller units can be preferable in the design of fault-detection and self-healing systems due to the fact that the observer/controller system itself might be faulty [BECKER and HÜBNER 2006], and this is one focus of our OES Architecture described in Section 3.

For realization of the recovery phase, Genetic Algorithms (GAs) have been applied to FPGA devices in various approaches. In the cases of intrinsic hardware evolution, the GA is invoked to apply crossover and mutation on the FPGA bitstream to evolve a fault-specific repair in-situ on the device. A software-simulation study of this approach was presented in [MILLIORD et al 2005]. It also explored the use of voting systems that operate in parallel despite imperfect GA solutions to refurbishment of local permanent damage in the FPGA fabric. Results showed improvement in aggregate repair performance from several different incomplete repairs obtained by the GAs. In [BRANKE et. al 2006], an autonomous self-repair approach for SRAM-based FPGAs is developed based on Competitive Runtime Reconfigurability. This approach was applied to a FPGA-based multiplier design which demonstrated evolution of a complete repair for 3x3 multiplier from several stuck-at-faults within a few thousand iterations. Using conventional offline population based approaches, GAs were also explored in [LOHN et. al 2003] for evolutionary fault recovery in Virtex FPGAs using an external controller and an offline repair process.

3. ORGANIC EMBEDDED SYSTEM (OES) ARCHITECTURE

As shown in Fig. 2, 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.

In OES, the AEs reduce the demand for centralized controllability as shown in Fig. 3. It consists of a *Concurrent Error Detection (CED)* [KHAKBAZ and McCLUSKEY 1982] 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 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 evaluation fault-detection methodology.

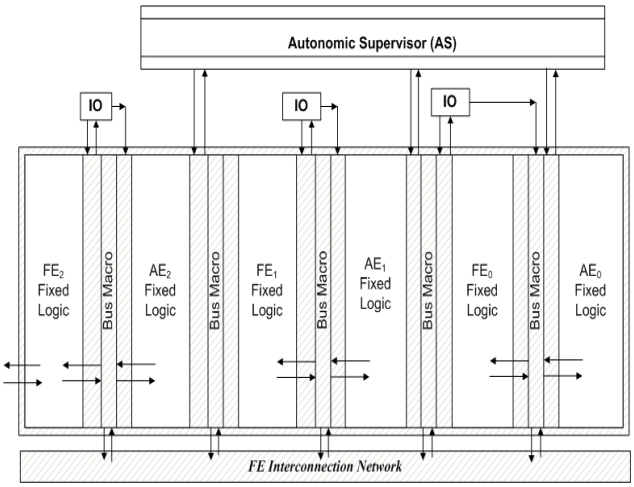

Fig. 2: Column-oriented OES on Xilinx Virtex II Pro FPGA platform.

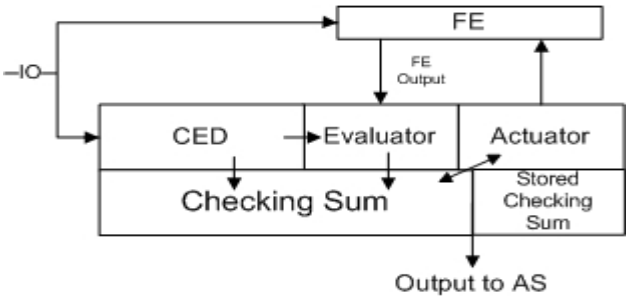

Fig. 3: AE architecture in OES.

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. Autonomic Evolutionary repair operators in [LOHN et. al 2003] are implemented here to achieve fault recovery. All other factors being equal, the likelihood of local permanent fault of any component is proportional to the device area required for its realization. Thus, the AS is as simple as possible to reduce its complexity and corresponding likelihood of experiencing a resource fault.

3.1 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. Fig. 4 shows a diagram of the flow of operations in the OES architecture as described below.

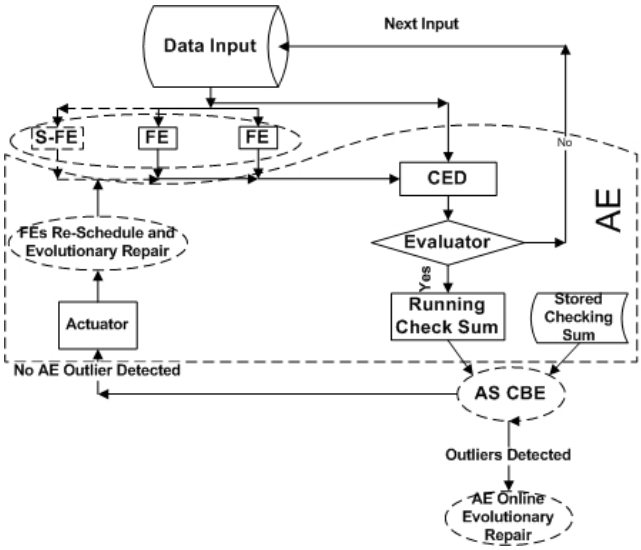

Fig. 4: OES Integrated FE and AE Failure Detection Procedure.

3.2 System Initialization

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. For the benchmark circuit selected a carry and sum, the CS-LUT required a 16-entry x 4-bit memory.

3.3 FE Fault Detection/Recovery

As depicted in Fig 4, 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 detection. The output of the FE is then compared in the AE-CED module and any discrepancy between the two values will indicate that a fault has occurred either of one the FE 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 innocent then the fault which occurred in this output will be discarded and

control will branch to a fault identification phase which will wakeup the cold standby FE and construct a temporary TMR system which can articulate the faulty FE under the new supplied external input. Furthermore, as described in Section 4, the actuator will initiate a repair cycle which may require automatic evolutionary repair in [LOHN et. al 2003] of the identified faulty FE which 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 as described above and AE self-repair mode. Whenever the AS identifies that an AE is faulty then 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.

The recovery procedure entails the use of alternative designs for the AE that have identical functionality but distinct physical resources. Evolutionary repair operation will manipulate the representation of the AE bitstream and evaluate each new generated offspring until the fault is occluded. This evolution may be time-consuming and halt the faulty FE operation, yet it is entirely automatic repair without any human intervention.

3.4 AE fault detection Phase

Three possible fault 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 simultaneously the inputs of the three components are 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.

4 EXPERIMENTAL RESULTS

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 *Consensus Based Evaluation (CBE)* [Zhang et al 2005] to 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 that AE under repair, until the number of newly created configuration evaluations reach predefined 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 yet 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 loss of generality.

The case study example shown in Fig. 5 was implemented on the Xilinx Virtex II Pro as proof of concept to accompany other software simulations performed and presented in this section. 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.

In Fig. 5, the FE and AE units are shown in dashed boxes. The CS-LUT is shown in the dotted box. 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 shown within the red box (contents not shown) will be used to activate the standby FE. TMR will be used in this case for the during various evaluation times which Evolutionary Operators will be used to repair the faulty FE individual. Once evolution achieves repair, the repaired FE then becomes the S-FE. This process is repeated each time a FE discrepancy is detected.

Notice that the inputs of the FE unit are connected to all FE units including the standby FE. Discrepancy in the two FE elements is detected using XOR gates fed to an OR gate. The output of the evaluator is fed to the Actuator that

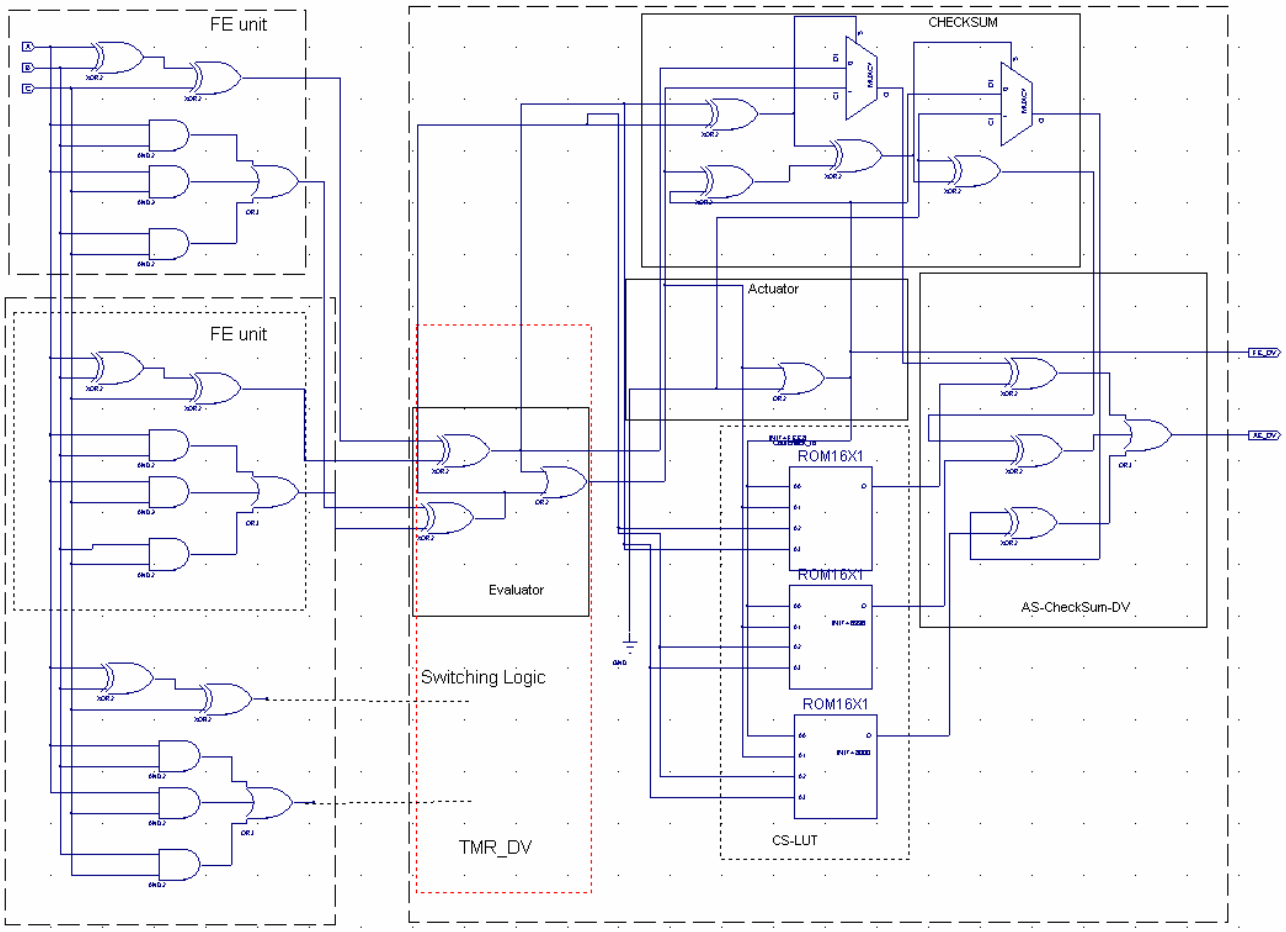

Fig.5: Gate Level Design of OES (Case study)

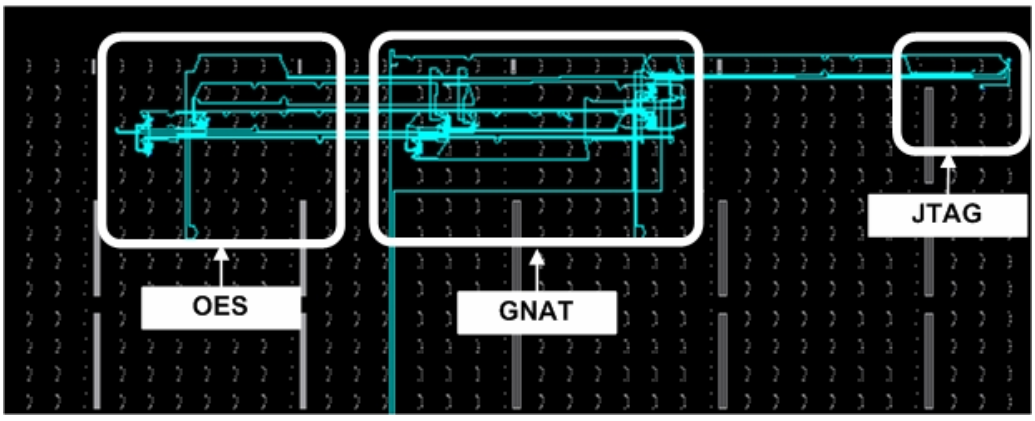

Fig. 6: Physical Layout of OES system on FPGA with GNAT/JTAG shown

uses an XOR gate to send a signal labeled FE Discrepancy Value (FE_DV) that will initiate GA operators on the FE unit once a discrepancy is detected. The outputs of the two XORs checking the two outputs of the two FE elements along with the Evaluator output and Actuator output are all fed to a checksum unit consisting of 4-to-2 compressor tree. In this particular case study only one 4-to-2 compressor is needed. To check for any discrepancy between the Checksum element and the CS-LUT during runtime, a circuit similar to the evaluator circuit is used. The outputs of the Checksum element and CS-LUT are fed to XORs and the output of the XORs are fed to an OR gate. The output of the OR gate named AE Discrepancy Value (AE_DV) will determine if a discrepancy is found between the two elements at runtime. The AE_DV is fed to the AS unit (not shown in figure) where it will be used along with CBE to confirm that the AE is in fact faulty and will cause the AS unit to initiate GA operators on the faulty AE element. Fig. 6 produced by Xilinx ISE shows the physical layout of the design shown in Fig. 5 on a Xilinx FPGA Virtex-II Pro.

Fig. 7 shows the OES evolution-based repair of the hardware prototype implemented with VHDL on the Xilinx Virtex-II Pro FPGA using the GNAT library along with the MRRA framework and JTAG reconfiguration interface. The design is that of the case study example shown in Fig. 6. The maximum fitness for case study example is 2 x 2^3=16 since the design has two outputs AE_DV and FE_DV. The genetic operators used perform Mutation and Crossover during each generation. The Crossover operator realizes a random single-point genetic crossover on the two parent chromosomes; the Mutation operator realizes a random single bit genetic mutation on the logical representation of the logical genotype representation. The Genetic operators perform only these two operations because re-routing is not supported under any Xilinx-II Pro FPGA dynamic reconfiguration flow.

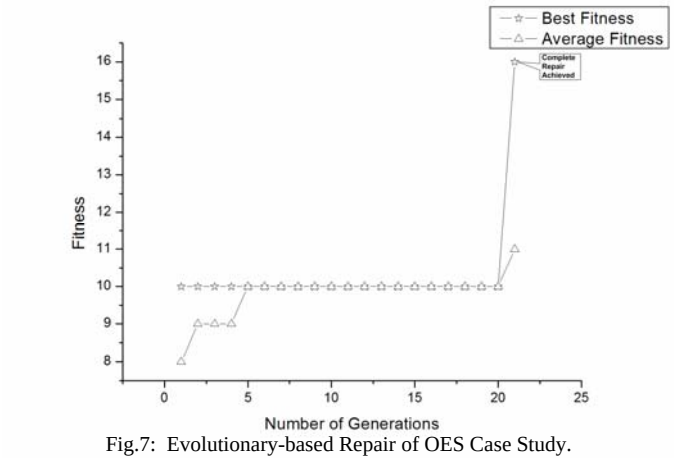

Fig.7: Evolutionary-based Repair of OES Case Study.

It should be noted here that redundancy was manually added to the design at a percentage of 15.4% of total LUT count to allow for routing changes by virtue of remapping permutation possible among redundant logic elements. The population size was set to 10 AEs/FEs. Fig. 7 shows the generation process for the OES system under stuck-at-one fault applied to one of the inputs of the AE. The system was fully repaired within 22 generations with one configuration from the population reaching the maximum fitness of 16 while the average fitness of the population reached 11 out of 16. Of course as soon as this one individual is completely repaired then it can be designated as the S-FE on the reconfigurable fabric.

Table 1: MCNC-91 Benchmark Circuits Evaluated on OES Architecture

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

5 CONCLUSION

In this paper, we developed Organic Embedded System (OES) architecture for sustainable performance of reconfigurable FPGA soft cores. The architecture was developed using an OC observer/controller organization and regeneration with Evolutionary Operators. Other innovations included provision of availability during regeneration, outlier-driven repair assessment, and a uniform design for the AEs. The design objective of developing an architecture that exhibits self-adaptation and self-healing properties can be attained using such techniques for completely autonomic operation without human intervention. The OES architecture is capable of handling many single fault scenarios and several multiple fault scenarios.

Experimental results strongly supported our design objectives were met. 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 synthesized OES architecture was prototyped on Xilinx Virtex-II Pro FPGA device supporting partial reconfiguration to demonstrate the feasibility of the OES architecture for intrinsic regeneration of the selected circuit. Through application of genetic operators for mutation and crossover, the OES architecture successfully achieved full repair of faulty element in the presence of stuck-at-zero and stuck-at-one faults. This integrated the use of redundant LUTs inherited in the FPGAs design. This integrated approach provides an innovation in fault-handling capability not only for the FEs, but also for the AEs as well.

For future work, we plan to evaluate extensions of the OES architecture for space-based embedded architectures. We are developing a working OES prototype on Xilinx Virtex-4 FPGA platform to supports more advanced online reconfiguration. We are also exploring new GA operators for fault-isolation and fault-correction.

6 REFERENCES

[BECKER 2006] BECKER J., HÜBNER M. 2006. Run-time Reconfigurability and other Future Trends. In *Proceedings of the 19th Annual Symposium on Integrated Circuits and Systems, Design Ouro* Preto, MG, Brazil, August 2006, ACM Press, New York, NY, 9-11.

[BERGMANN and WILLIAMS 2003]BERGMANN N., & WILLIAMS J. 2003. Egret: a platform for reconfigurable system-on-chip. In *Proceedings of 2003 IEEE International Conference on Field-Programmable Technology (FPT)*, December 2003, Tokyo, 340-343.

[BERNAUER et al 2006] BERNAUER A., BRINGMANN O., ROSENSTIEL W., BOUAJILA A., STECHELE W., HERKERSDORF A. 2006. An Architecture for Runtime Evaluation of SoC Reliability. In *INFORMATIK 2006 - Informatik für Menschen*, Lecture Notes in Informatics, Köllen Verlag, vol. P-93 of GI-Edition, 177-185.

[BOUAJILA et al 2006] BOUAJILA A., BERNAUER A., HERKERSDORF A., ROSENSTIEL W., BRINGMANN O., & STECHELE W. 2006. Error Detection Techniques Applicable in an Architecture Framework and Design Methodology for Autonomic SoCs. In Yi Pan, Franz J. Rammig, Hartmut Schmeck, and Mauricio Solar, editors, *1st IFIP International Conference on Biologically Inspired Cooperative Computing (BICC 2006)*, vol. 216, August 2006, Springer, Boston, MA, USA, 107-113.

[BRANKE et al 2006] BRANKE J., MNIF M., MÜLLER-SCHLOER C., PROTHMANN H., RICHTER U., ROCHNER F., SCHMECK H. 2006. Organic Computing - Addressing Complexity by Controlled Self-organization. In Tiziana Margaria, Anna Philippou, and Bernhard Steffen, *Proceedings of ISoLA 2006*, November 2006, Paphos, Cyprus, 200-206.

[CARMICHAEL et al 2000] CARMICHAEL C., CAFFREY M., AND SALAZAR A. 2000. Correcting single-event upsets through Virtex partial configuration. *Technical Report, Xilinx Corporation, XAPP216 (v1.0)*. Retrieved April 26th 2007 from: http://direct.xilinx.com/bvdocs/appnotes/xapp216.pdf

[Zhang et al 2005] ZHANG KAND DEMARA R. F. 2005. "Consensus-based Evaluation for Fault Isolation and On-line Evolutionary Regeneration," in *Proceedings of the International Conference in Evolvable Systems (ICES'05)*, pp. 12 - 24, Barcelona, Spain, September 12 - 14, 2005.

[KHAKBAZ and Mccluskey 1982] KHAKBAZ, J., & E.J. MCCLUSKEY 1982.Concurrent Error Detection and Testingfor Large PLA's. *Joint Special Issue on VLSI, IEEE Transaction on Electron Devices*, 756-764 and *IEEE Journal of Solid-State Circuits*,386-394

[LIPSA et al 2005] LIPSA G., HERKERSDORF A., ROSENTIEL W., BRINGMANN O., & STECHELE W. 2005. Towards a Framework and a Design Methodology for Autonomic SoC. In *Proceedings of the Second International Conference on Autonomic Computing (ICAC'05)*, June 2005 IEEE Computer Society, Washington, DC, USA, 391-392

[LOHN et al 2003]LOHN J. D., LARCHEV G., AND DEMARA R. F. 2003. A Genetic Representation for Evolutionary Fault Recovery in Virtex FPGAs. In *Proceedings of the Fifth International Conference on Evolvable Systems (ICES'03)*, Trondheim, Norway, March 2003, A. Tyrrell, P. Haddow, and J. Torresen (eds), Springer LNCS 2606, Berlin, 47 – 56.

[MILLIORD 2005] MILLIORD C. J., SHARMA C. A., DEMARA R. F. 2005. Dynamic Voting Schemes to Enhance Evolutionary Repair in Reconfigurable Logic Devices. in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig'05)*, Puebla City, Mexico, 2005.

[SCHMECK 2005] SCHMECK H. 2005. Organic Computing- A New vision for distributed Embedded Systems, In *Proceedings of Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, IEEE Computer Society, Washington, DC, USA, 201-203.

[SEKAR et al 2000]SEKAR K., SANCHEZ P., DEY S., CHENG Y. and CHEN L. 2000. Embedded Hardware and Software Self-Testing Methodologies for Processor Cores. *In Proceeding of the 37th Conference on Design Automation (DAC'00)*. Los Angeles, Ca, USA, June 2000, IEEE Computer Society, Washington, DC, USA 625-630.

[STROUD et al 2004]STROUD C., SUNWOO J., GARIMELLA S., and HARRIS J. 2004. Built-In Self-Test for System-on-Chip: A Case Study. *In Proceeding of the International Test Conference (ITC'04)*, October 2004, Charlotte, NC, USA, IEEE Computer Society, Washington, DC, USA 837-846.

[YANG 1991] YANG S., 1991. Logic synthesis and optimization benchmarks user guide version 3.0. *Technical Report, Microelectronics Center of North Carolina*, Research Triangle Park, NC, Jan. 1991.