Autonomous FPGA Fault Handling through Competitive Runtime Reconfiguration

Ronald F. DeMara, *Senior Member, IEEE* and Kening Zhang, *Student Member, IEEE*
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450
demara@mail.ucf.edu

Abstract

An autonomous self-repair approach for SRAM-based FPGAs is developed based on Competitive Runtime Reconfiguration (CRR). Under the CRR technique, an initial population of functionally identical (same input-output behavior), yet physically distinct (alternative design or place-and-route realization) FPGA configurations is produced at design time. At run-time, these individuals compete for selection based on a fitness function favoring fault-free behavior. Hence, any physical resource exhibiting an operationally-significant fault decreases the fitness of those configurations which use it. Through runtime competition, the presence of the fault becomes occluded from the visibility of subsequent FPGA operations. Meanwhile, the offspring formed through crossover and mutation of faulty and viable configurations are reintroduced into the population. This enables evolution of a customized fault-specific repair, realized directly as new configurations using the FPGA's normal throughput processing operations. Multiple phases of the fault handling process including Detection, Isolation, Diagnosis, and Recovery are integrated into a single cohesive approach. FPGA-based multipliers are examined as a case study demonstrating evolution of a complete repair for a 3-bit x 3-bit multiplier from several stuck-at-faults within a few thousand iterations. Repairs are evolved in-situ, in real-time, without test vectors, while allowing the FPGA to remain partially online.

1 Introduction

Evolutionary mechanisms can actively restore mission-critical functionality in SRAM-based reprogrammable devices. They provide an alternative to device redundancy for dealing with permanent degradation due to radiation-induced stuck-at-faults, thermal fatigue, oxide breakdown, electromigration, and other local permanent damage. Potential benefits include recovery without the increased weight and size normally associated with spares. Also, failures need not be precisely diagnosed due to intrinsic evaluation of the FPGA's residual functionality through assessment of the Genetic Algorithm (GA) fitness function.

Autonomous repair of FPGAs is of particular interest in aerospace applications for both in-flight and Ground Support Equipment devices. Some deep space satellites utilize over 100 FPGA devices [Actel99], and more recently, space-qualified versions of high-density SRAM-based FPGAs have become commercially-available [Xilinx04]. Meanwhile NASA terrestrial applications routinely employ FPGAs for tasks ranging from launch control to signal processing. SRAM-based FPGAs are of significant importance due to their high density and increasing use in mission-critical/safety-impacting applications. Meanwhile, they offer unlimited reprogrammability that can enable autonomous repair.

For in-flight applications, FPGA devices encounter harsh environments of mechanical/acoustical stress during launch, high doses of ionizing radiation, and thermal stress. Simultaneously, they are required to operate reliably for long mission durations with limited or absent capabilities for diagnosis/replacement and little onboard capacity for spares. Hence, recent research has focused on employing the reconfigurability inherent in field programmable devices to increase reliability and autonomy [Keymeulen00] [Vigander01] [Abramovici01] [Lohn03].

This paper develops a novel autonomous fault handling approach to these problems based on *Competitive Runtime Reconfiguration (CRR)*. Some distinguishing features of CRR are:

1. integration of an adaptive computing approach throughout the device lifecycle,
2. device refurbishment without additional function or resource test vectors, and

3. a novel fitness assessment approach via pairwise discrepancy detection without a pre-conceived oracle for correctness.

Fault handling lifecycle support is realized using a pairwise comparison to *detect* faults against a diverse population of competing configuration alternatives. The genetic operator of crossover *isolates* the failed physical resource, and alone or with mutation, realizes a failure-specific *repair* during normal operations to make detailed physical failure mode *diagnosis* unnecessary. CRR integrates competition and evolution wholly within the FPGA's normal data throughput processing flow to eliminate the need for additional test vectors. Because a fitness function is used that favors fault-free behavior, the FPGA's normal input data stream can be used to evaluate fitness. This also ranks competing alternatives with regards to their relative performance to provide graceful degradation even in the presence of multiple faults.

2 Related Work

Over the last few years, *adaptive regeneration* has been investigated as an alternative to pre-determined spares. As listed in Table I, Vigander's [Vigander01] regeneration approach extends Triple Modular Redundancy (TMR) to utilize faulty FPGAs that have been partially regenerated using evolutionary algorithms. He demonstrates that FPGA-based implementations of 4-bit x 4-bit multipliers can be automatically reconfigured to realize partial refurbishment. Since each partially refurbished multiplier is deficient with respect to only selected input pairs, a voting arrangement of partially refurbished devices exhibits complete functionality. TMR, Vigander's, and other *q*-plex *spatial voting* approaches deliver real-time fault resolution, but increase power consumption *q*-fold during fault-free operation.

Lohn, Larchev, and DeMara [Lohn03b] develop a FPGA bit-string representation along with mutation and two-point crossover operators for actively refurbishing interconnect as well as logic resources. Their approach regenerated a Quadrature Decoder circuit on a Xilinx SRAM-based Virtex XCV1000 FPGA. It shows that a stuck-at-fault on the input to a Configurable Logic Block (CLB) can be resolved through GA-based repair. The GA synthesizes a new alternative configuration within a population of 40 competing configurations after a few hundred generations. Their GA was shown to be capable of recycling faulty components as partially-damaged CLBs were sometimes reassigned to alternative functions based on their residual functionality. While Lohn, Larchev, and DeMara's approach demonstrated complete regeneration for a modestly-sized sequential circuit, the refurbishment was performed offline and required exhaustive fitness test vectors.

Lach's deterministic approach [Lach98] segments the FPGA into static tiles at design time with a known functionality, some redundant resources, and a pre-designed alternate configuration. Spare tiles can be selected when needed, but their functionality is predetermined and thus limited.

On the other hand, STARS [Abramovici01] is a resource-oriented diagnostic test approach that performs Built-in Self-Tests (BISTs) on roving sub-sections of the FPGA. Portions are continually taken offline in succession and tested while the functionality is moved to a new location within the reprogrammable fabric. Detection latency can be large since tests must sweep through all intervening resources before a fault is detected. Potential throughput unavailability due to diagnostic reconfigurations when no faults have yet occurred is also consideration. Nonetheless, it provides a useful approach for deterministic exhaustive online constructive repair.

An alternative approach is taken by Keymeulen, Stoica, and Zebulem [Keymeulen00] using a design-time emphasis. They develop evolutionary techniques so that a circuit is initially designed to remain functional even in presence of various faults. Their *population-based* fault tolerant design method evolves diverse circuits and then selects the most fault-insensitive individual. This method enables passive runtime operation. It is also applicable for constructing a diverse initial population under our proposed CRR approach.

The CRR technique will be presented in detail in Section 3 and some selected characteristics are overviewed in Table I. CRR utilizes a *temporal voting* approach whereby the outputs of just two competing instances are compared at any instant and alternative pairings are considered over time. The presence or absence of a discrepancy is used to adjust the fitness of both individuals without rendering any judgment at that instant on which individual is actually faulty. The faulty, or later exonerated, configuration is determined over time through other pairings of competing configurations. The competitive process is applied repeatedly to form a strong consensus across a diverse pool of alternatives. Under CRR, the FPGA's outputs are compared before they leave the reconfigurable device so fault detection occurs on the first erroneous output and detection latency is negligible. Fault isolation in the TMR, Vigander, and Lach approaches is restricted to coarse predefined functional granularities. Meanwhile, STARS attempts to isolate faults at only the very finest resource granularity. Alternatively, CRR does not impose a predetermined fault isolation granularity in order to achieve refurbishment.

Table I: Characteristics of Related FPGA Fault-Handling Schemes

Approach	Fault Handling Method	Fault Detection		Resource Coverage			Fault Isolation
		Latency	Distinguish Transients	Logic	Inter-connect	Comparator	Granularity
TMR	Spatial voting	Negligible	No	Yes	Yes	No	Voting element
[Vigander01]	Spatial voting & offline evolutionary regeneration	Negligible	No	Yes	No	No	Voting element
[Lohn, Larchev, DeMara03]	Offline evolutionary regeneration	Negligible	No	Yes	Yes	No	Unnecessary
[Lach98]	Static-capability tile reconfiguration	Relies on independent fault detection mechanism					
STARS [Abramovici01]	Online BIST	Up to 8.5M outputson ORCA FPGA	Test pattern transients	Yes	Yes	No	LUT function
[Keymeulen, Stoica, Zebulum00]	Population-based fault insensitive design	Design-time prevention emphasis	No	Yes	Yes	No	Not addressed at runtime
CRR	*Competing configurations with temporal voting and online regeneration*	*Negligible*	*Transients are attenuated automatically*	*Yes*	*Yes*	*Yes*	*Variable*

3 CRR Technique for Autonomous Regeneration

Figure 1 depicts the relative health states in a dynamic population of competing FPGA configurations under CRR. Initially, a population of `Pristine` individuals is created. These primordial configurations are functionally-identical (same input-output behavior), yet utilize physically-distinct resources (alternative design or place-and-route implementation). When output discrepancies occur in operations performed by the FPGA, transitions in the health state of the exhibiting configurations occur. These transitions are labeled by the numbered arcs in Figure 1.

To illustrate the process, consider the FPGA physical arrangement shown in Figure 2. It depicts two competing *half-configurations* labeled Functional Logic Left ("*L*") and Functional Logic Right ("*R*") loaded in tandem. These competing configurations can be stored in the same EEPROM device required to boot the SRAM FPGA to keep device count low or can be maintained in an external RAM.

The state transition arrows labeled "$L=R$" and "$L\neq R$" indicate whether the two resident half-configurations produce either *matching* or *discrepant* outputs, respectively. For example, when $L=R$ occurs under the transition event "*1*" then both individuals retain their `Pristine` state. However when their outputs disagree then transition "*2*" occurs. Both *L* and *R* configurations are demoted to the `Suspect` pool and the fitness of both individuals is decreased. Whenever transition "*2*" occurs, a *Fault Alert* indicator is issued because two functionally-identical circuits disagree. Hence at least one resource fault must have occurred.

Whenever two half-configurations agree, the fitness values of both half-configurations are raised. By repeated pairing over a period of time, only those half-configurations which do not use faulty resources will eventually become preferred. This is because the fitness of a faulty half-configuration is always decreased regardless of its pairing, yet fitness of fault-free half-configurations which are paired together are increased.

Any physical resource exhibiting an operationally-significant fault automatically decreases the fitness of those configurations which use it. This process occurs as part of the normal processing throughput of the FPGA without requiring any supplemental test vectors or other diagnostics. The determination of a configuration's health state is based on its cumulative correctness performance relative to threshold values over a period of random samples called the *Evaluation Window*, denoted by E^W.

More formally, the *i*-th half configuration remains in the `Suspect` pool until its fitness f_i evaluated over the preceding E^W pairings the drops below the *Repair Threshold* ($f_i < f_{RT}$). The *i*-th half-configuration is then marked as `Under Repair` until its fitness rises above the *Operational Threshold* ($f_i \geq f_{OT}$) through the application of genetic operators. Normally, f_{OT} is selected such that $f_{OT} > f_{RT}$ which provides dithering immunity such that the configuration is indeed `Refurbished` as indicated by transition "*8*."

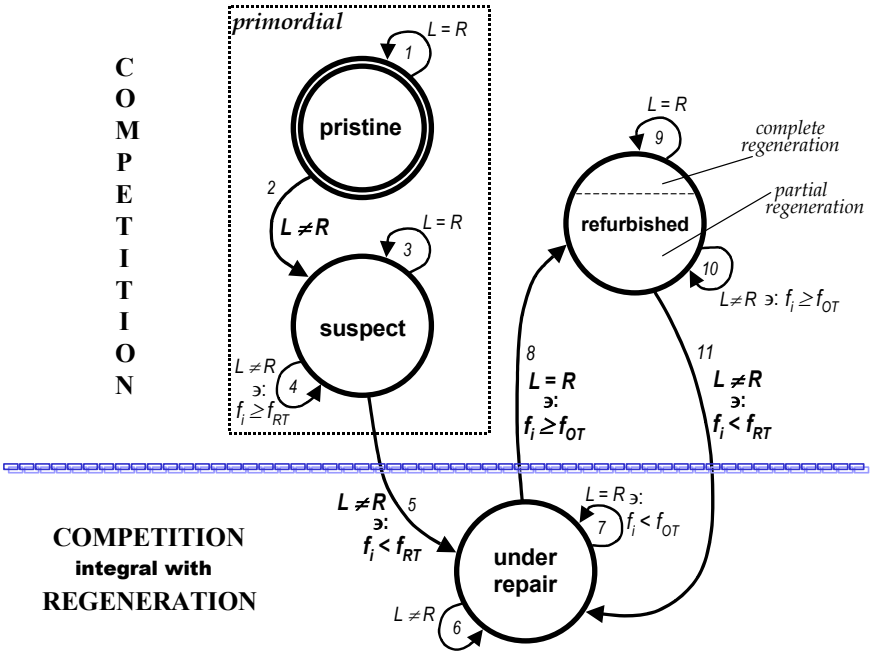

Figure 1: States in the Lifetime of the i^{th} Half-Configuration

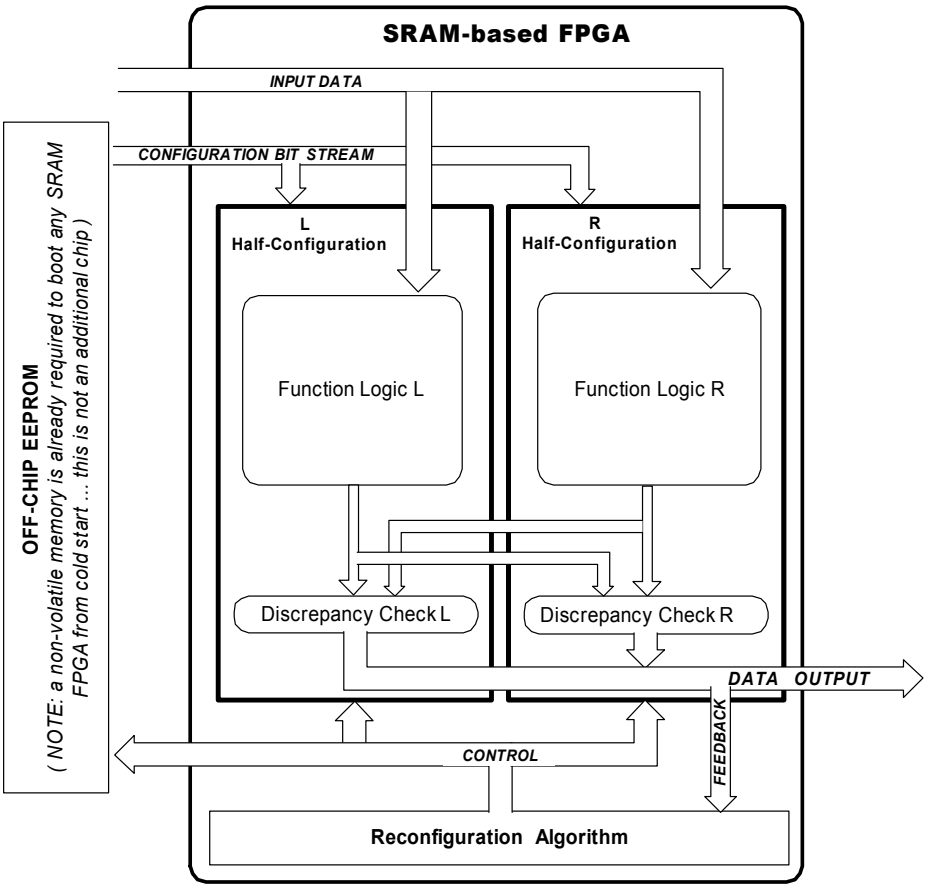

Figure 2: Tandem CRR arrangement in SRAM-based Reconfigurable FPGA

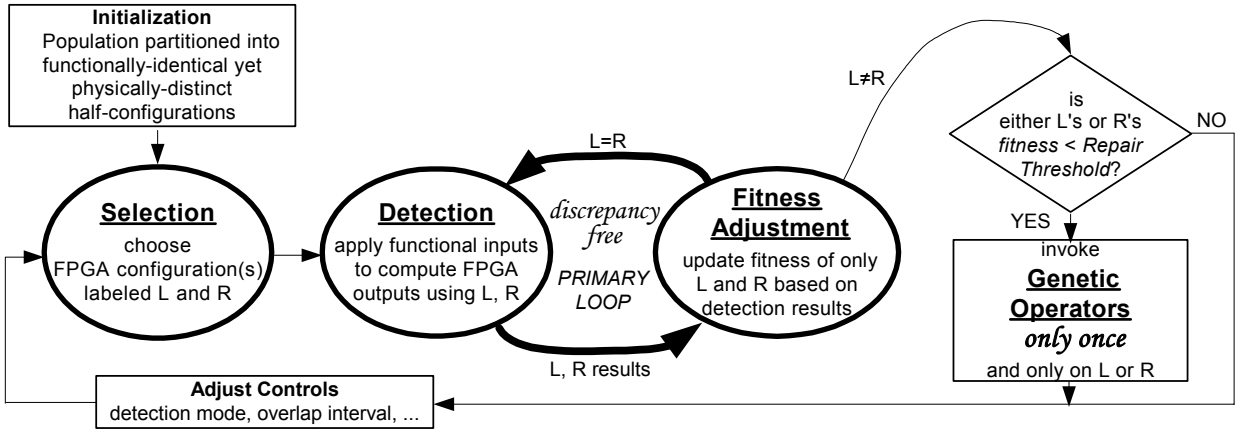

Figure 3: Procedural Flow in the CRR Technique

Whenever instances of transition "*9*" occur then f_i is further increased and *complete regeneration* becomes possible though not necessarily externally distinguishable from *partial regeneration*. Competing half-configurations remain `Refurbished` unless $f_i < f_{RT}$ at which time they again demoted to the `Under Repair` state due subsequent discrepant behavior.

The procedural flow of the CRR algorithm that calculates the health state transitions is depicted in Figure 3. After initialization, *Selection* of the *L* and *R* half-configurations occurs which are then loaded into the FPGA. The *Detection* process is conducted when the normal data processing inputs are applied to the FPGA. Based on agreement or disagreement among the outputs of the two competing *L* and *R* half-configurations, *Fitness Adjustment* for both individuals occurs. The central *PRIMARY LOOP* representing *discrepancy-free* behavior, can repeat indefinitely without any reconfiguration of the FPGA. Only when outputs disagree, as indicated by the "*L≠R*" path in Figure 3, do alternate configurations need to be loaded. If $f_i < f_{RT}$ then Genetic Operators are invoked only once on the resident *i*-th half-configuration(s). The modified half-configuration is then immediately returned to the pool of competing configurations and the Selection step is resumed under normal FPGA throughput processing operations. Genetic operators used include two-point crossover to replace functional units with designs from other viable configurations, mutation which reconfigures suspect CLBs with random alternatives, cell swapping which moves CLBs within a configuration.

3.1 Selection Process

The *Selection* process is shown in Figure 4. Its usual flow is for `Pristine`, `Suspect`, and then `Refurbished` individuals to be preferred in that order for one half-configuration. On the other hand, the other half-configuration is selected based on a stochastic process determined by the *Re-introduction Rate* (λ_R). In particular, `Under Repair` individuals are selected as one of the competing half-configurations on average at a rate equal to λ_R. Henceforth, this now genetically-modified configuration will be *re-introduced* into the operational throughput flow as a new competitor to potentially exhibit fault-free behavior against the larger pool of configurations not currently undergoing repair.

As listed near the center of Figure 4, three selection biases are employed to favor either *inventory rotation*, *correctness*, or *correctness augmented with operational performance.* In the latter case, inclusion of a second order term within the fitness function allows not only repair, but also simultaneous consideration of which competing equally-correct configurations exhibit better *throughput* or perhaps *power consumption* performance.

An additional innovation is that λ_R is not only a continuous variable, but can be adapted under autonomous control. In particular, we strive for Mean-Time-To-Repair (*MTTR)* < Mean-Time-Between-Failures *(MTBF)* by monitoring the ratio of the number of computations elapsed between transitions {"*5*" and "*8*"} (for MTTR), as compared to the sum of iterations between transitions [{"*8*" and "*10*"} + {"*2*" and "*5*"}] (for MTBF) as numbered in Figure 1 and adjusting λ_R accordingly.

3.2 Detection Process

The *Detection* process in Figure 5 depicts a two formulations of *Tandem* and *Bounding* operational modes for fault-identification through FPGA runtime reconfiguration. In addition to the *Tandem* mode presented thus far whereby both the *L* and *R* half-configurations are resident simultaneously, an intriguing alternative is available.

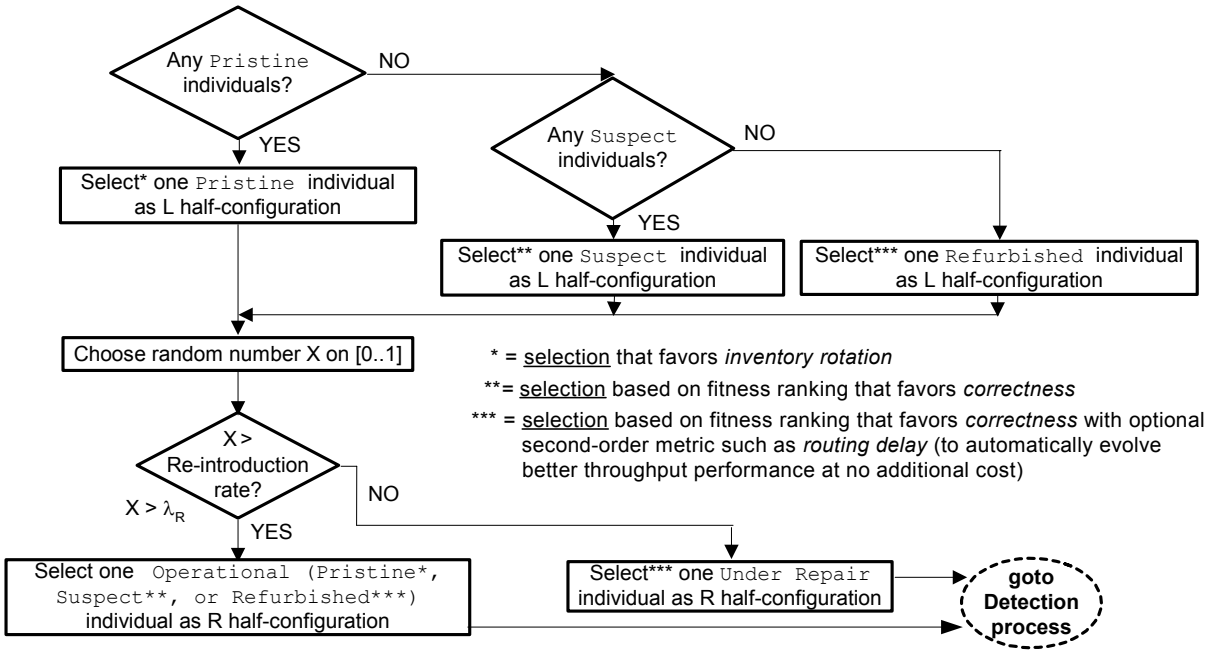

Figure 4: Selection in the CRR Technique

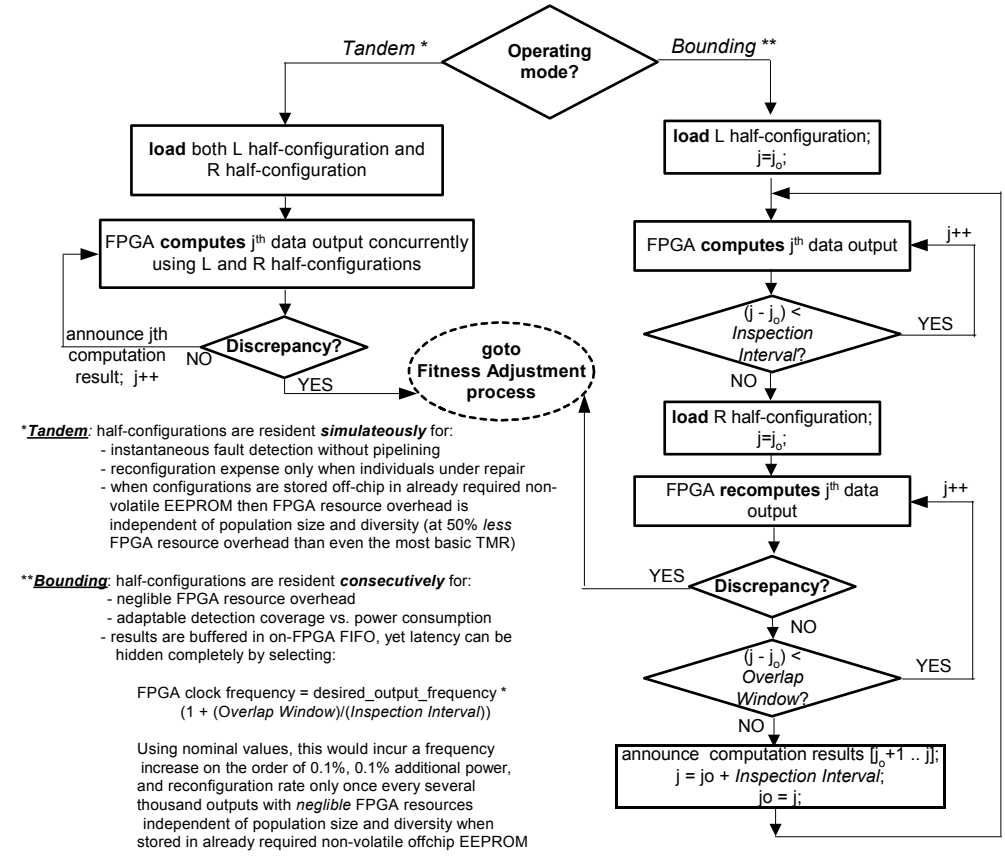

Figure 5: Detection in the CRR Technique

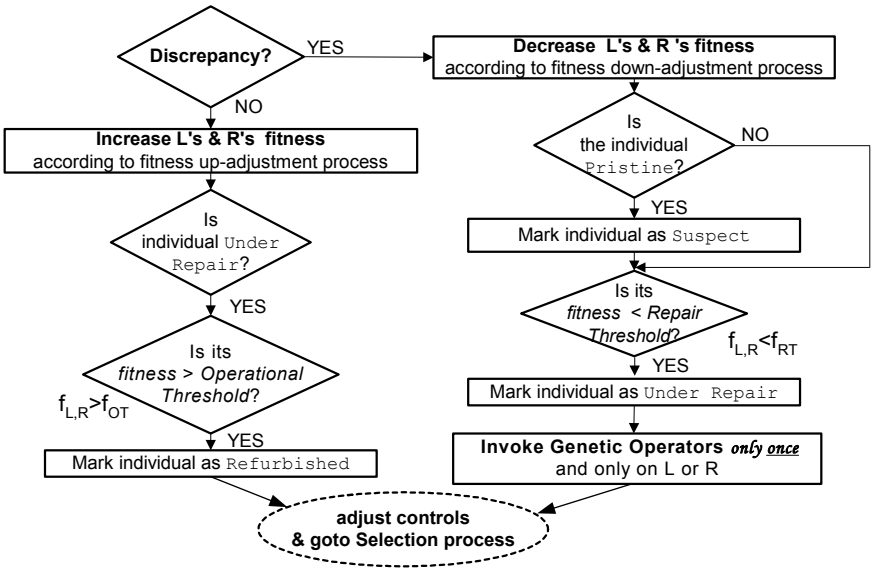

Figure 6: Fitness Adjustment and Evolutionary Processes in the CRR Technique

Under *Bounding* detection mode, only a single configuration is resident in the FPGA at any time allowing FPGA resource utilization close to 100%. Bounding mode computes a contiguous run of outputs as bounded by the *Inspection Interval.* Next, the FPGA is reconfigured to a different single competing configuration and outputs within the *Overlap Window* are redundantly computed. These repeated computations are used to adjust fitness only if a discrepancy occurs.

3.3 Fitness Adjustment and Evolution

Figure 6 depicts *Fitness Adjustment and Evolutionary Processes.* Under CRR fitness of both *L* and *R* half-configurations are increased by the *fitness up-adjustment procedure* when their outputs agree. Alternatively, a *fitness down-adjustment procedure* with a steeper gradient occurs in the event of a discrepancy. Defining f_{RT} as dynamic value proportional to the instantaneous maximum population fitness (f_{Elite}) such that $f_{RT}=0.001f_{Elite}$ has performed well in the experiments described below.

When ($f_i < f_{RT}$) then two-point crossover with a randomly selected `pristine` individual and then mutation are invoked a single time. For crossover to occur such that offspring are guaranteed to utilize only mutually-exclusive physical resources with other resident half-configurations, the larger population can be partitioned into mutually exclusive sub-populations *L* and *R*. By enforcing speciation so breeding occurs exclusively in *L* or exclusively in *R*, non-interfering resource use is maintained.

Finally, since an instance of the XNOR checking logic is embedded within each individual as shown in Figure 2, the error checking circuitry itself also competes to exhibit fault-free-behavior. Hence, an error in any checker decreases that configuration's fitness. Thus the use of the faulty checker is eventually avoided during subsequent selections.

4 Experimental Results

Table IV shows that CRR-based regeneration is feasible for an FPGA-based 3x3 multiplier. Starting with an initial population of 20 viable configurations, random stuck-at faults were injected. For each experimental run, one fault was injected randomly into one of five different CLBs out of the 36 CLBs that were utilized to implement the multiplier on a simulated FPGA architecture. Each fault reduced the number of correct outputs from 64 out-of-64, to an average of 32.6 out-of-64. Regeneration was performed under CRR using a fitness adjustment process based on the Hamming Distance of competing outputs, gradated fitness adjustment, repair threshold at 98% of current maximum fitness, and a 10% re-introduction rate. While the simulated FPGA remained partially online, regeneration improved correctness to an average of 61.6 out-of-64 possible outputs. In roughly half of the experimental runs, complete repair was achieved in a few thousand evaluations of the normal functional input data to the simulated FPGA when starting with a highly diverse initial population. During the regeneration period, data throughput averaged 87.4%. Hence, only 13.6% of the total computations needed to be redundant in order to preclude propagation of any discrepant outputs, even when candidate repairs were being re-introduced to refurbish the impacted FPGA configuration without additional test vectors.

Table IV: CRR Performance during Regeneration of a 3x3 Multiplier

3x3 Multiplier Run #	Fault Location	Failure Type	Correctness after Fault	Total Iterations	Discrepant Iterations	Repair Iterations	Final Correctness	Throughput
1	CLB3,LUT0,Input1	Stuck-at-1	52 / 64	623601	94095	1953	64 / 64	84.9
2	CLB6,LUT0,Input1	Stuck-at-0	33 / 64	304125	29721	525	54 / 64	90.2
3	CLB5,LUT2,Input0	Stuck-at-1	22 / 64	913149	139505	2849	64 / 64	84.7
4	CLB7,LUT2,Input0	Stuck-at-0	38 / 64	531115	64351	1311	62 / 64	87.9
5	CLB9,LUT0,Input1	Stuck-at-0	40 / 64	347129	38380	767	64 / 64	88.9
		Average	*32.6 / 64*	*543823*	*73210*	*1409*	*61.6 / 64*	*87.4*

5 Conclusions

Faults in the FPGA fabric, configuration-storing memory, and error detection circuitry are addressed by CRR. These are handled via the CRR process because faulty competing functional outputs will indicate discrepancy, as do configurations loaded with incorrect bit streams from configuration storing memory, and faulty XNOR comparators disagree, respectively. If any of these faults happens to be a transient, CRR's alternate pairing process will scrub the SEU(s) automatically. Fitness will be decreased initially due to transient faults, but later rise so that transients are resolved instantly and attenuated automatically over time.

Nonetheless, some failure exposures inevitably remain. If the reconfigurability of the FPGA is impaired then any approach based on reconfiguration would by definition become infeasible. Likewise a catastrophic failure impacting a vast majority of the half-configurations could make correct consensus impossible. Vulnerabilities in the CRR control process, such as the fitness of individuals can be mitigated by maintaining a local fitness table within each half-configuration; other control functions in the GA can be made to be competitive using alternate physical resources as well. We are currently studying these extensions, and also broadening the CRR paradigm and its application to other self-healing systems.

Using adaptive competition, the vast majority of resources in the FPGA can be covered. Faults are detected, isolated, and resolved without additional test vectors while allowing the device to remain partially online. Furthermore, this approach has the potential to provide these benefits with no increase in chip count in the fielded system if the GA is embedded within the FPGA fabric.

Acknowledgments

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

References

[Abramovici01] M. Abramovici, J. M. Emmert, and C. E. Stroud, "Roving STARs: An Integrated Approach To On-Line Testing, Diagnosis, and Fault Tolerance For FPGAs in Adaptive Computing Systems," *NASA/DoD Workshop on Evolvable Hardware*, 2001.

[Actel99] Actel Corporation, "Actel FPGAs Make Significant Contribution to Global Space Exploration," available at http://www.actel.com/company/press/1999pr/SpaceContribution.html

[Keymeulen00] D. Keymeulen, A. Stoica, and R. Zebulum, "Fault-Tolerant Evolvable Hardware using Field Programmable Transistor Arrays," *IEEE Transactions on Reliability*, Vol.49, No. 3, Sept. 2000.

[Lach98] J. Lach, W.H. Mangione-Smith, and M. Potkonjak, "Low Overhead Fault-Tolerant FPGA Systems," *IEEE Transactions on VLSI Systems*, Vol. 6, No. 2, June 1998, pp 212-321.

[Lohn03a] J. D. Lohn, G. Larchev, and R. F. DeMara, "A Genetic Representation for Evolutionary Fault Recovery in Virtex FPGAs," In *Proceedings of the 5th International Conference on Evolvable Systems (ICES)*, Trondheim, Norway, March 17-20, 2003.

[Lohn03b] J. D. Lohn, G. Larchev, and R. F. DeMara, "Evolutionary Fault Recovery in a Virtex FPGA Using a Representation That Incorporates Routing," In *Proceedings of 17th International Parallel and Distributed Processing Symposium*, Nice, France, April 22-26, 2003.

[Vigander01] S. Vigander, *Evolutionary Fault Repair of Electronics in Space Applications*, Dissertation, Norwegian University Sci. Tech., Trondheim, Norway, February 28, 2001.

[Xilinx04] Xilinx Inc., "Xilinx Launches New Era Of Digital Design In Aerospace And Defense With Introduction of QPRO Virtex-II Family," May 2004, available at: http://www.xilinx.com/prs_rls/end_markets/0458qprovii.htm