Sustainability Assurance Modeling for SRAM-based FPGA Evolutionary Self-Repair

Rashad S. Oreifej, Rawad Al-Haddad, Rizwan A. Ashraf, and Ronald F. DeMara
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL, 32816-2362
demara@mail.ucf.edu

***Abstract*— A quantitative stochastic design technique is developed for evolvable hardware systems with self-repairing, replaceable, or *amorphous spare components*. The model develops a metric of *sustainability* which is defined in terms of residual functionality achieved from pools of *amorphous spares* of dynamically configurable logic elements, after repeated failure and recovery cycles. At design-time the quantity of additional resources needed to meet mission availability and lifetime requirements given the fault-susceptibility and recovery capabilities are assured within specified constraints. By applying this model to MCNC benchmark circuits mapped onto Xilinx Virtex-4 Field Programmable Gate Array (FPGA) with reconfigurable logic resources, we depict the effect of fault rates for aging-induced degradation under Time Dependent Dielectric Breakdown (TDDB) and interconnect failure under Electromigration (EM). The model considers a population-based genetic algorithm to refurbish hardware resources which realize repair policy parameters and decaying reparability as a complete case-study using published component failure rates.**

Keywords—Evolvable Hardware; Field Programmable Gate Arrays (FPGAs); Genetic Algorithms (GAs); Autonomous Fault-Refurbishment; Dynamic Resource Allocation; Reliability; Sustainability; Fault Modeling.

I. INTRODUCTION

Autonomous systems which employ dynamically reconfigurable devices, such as SRAM-based Field Programmable Gate Arrays (FPGAs) [1-2], aim to carry out complex tasks in dynamic and uncertain environments such as space missions, autonomous vehicles, and avionics. Their ability to provide fault handling and self-refurbishment has increasing significance as the mission criticality and duration is increased, and in the presence of continued technology-scaling to sub-90nm regimes utilized by embedded system components. SRAM-based FPGAs, like all semiconductor devices, are susceptible to a range of failure mechanisms under such conditions. In this work, the feasibility of evolvable hardware based approach in the presence of technology-driven faults is assessed.

These faults could be soft faults which are transient Single Event Upsets (SEUs), or hard permanent faults [3]. SEUs primarily affect storage elements and since FPGAs are comprised of large arrays of SRAM memory cells, historically, SEUs have received significant attention. However, hard failure considerations become more prominent as technology advances towards reduced device feature sizes which provide higher logic densities, lower energy operation, and a smaller form factor.

FPGA self-adaptation mechanisms have been demonstrated using a variety of approaches [4][5][6]. Recovery techniques range from static approaches involving simple spare replacement to model-free dynamic methods which consider metrics such as the degree of functionality restoration, recovery latency, area redundancy, complexity, and fault coverage. Being dynamically reconfigurable at runtime, FPGAs enable the spare granularity to be provided at variable levels. This can range from the module-level down to a gate-equivalent level, rather than a fixed datapath component. This relaxes fault-handling constraints from modular redundancy to adaptable, fine-grained, reconfigurable resources such as Lookup Tables (LUTs) which can realize a logic function of just a few Boolean inputs. In this regard, *Evolvable hardware systems* utilizing *Genetic Algorithms (GAs)* offer self-adaption capability at various granularities.

In this paper, the quantity of such uncommitted resources that the mission requires to survive a targeted deployment period is formulated as a constraint-satisfaction problem for reconfiguration enabled by GA-based refurbishment. Namely, a critical factor during autonomous refurbishment is the expected value of the Mean Time To Repair (MTTR). Conventional availability analysis dictates a threshold value of MTTR upon which the mission maintains an acceptable availability level. Meanwhile as the mission progresses, cumulative fault likelihood might best be anticipated to remain constant, yet in practice often increases. In other words, the time-to-refurbish may increase as more parts fail. Thus, dynamic reparation performance is seen as an optimization challenge that is amenable to GA-based fault recovery. Specifically, we address the fundamental technical objective:

- *How many uncommitted reconfigurable resources are needed for an FPGA-based system to sustain its functionality within planned mission availability and lifetime specifications when using online evolution?*

This paper formulates a quantitative stochastic sustainability model and applies it to FPGA-based self-configuring devices. This model estimates at design-time the resources required for refurbishment in order to meet mission availability and lifetime requirements in a given ecosystem of different fault types, rates, and impact. Hence, sustainability

This research was supported in part by NASA Intelligent Systems NRA Contract NNA04CL07A and DARPA contract #W31P4Q-08-C-0168.

analysis establishes a novel paradigm of analytical tools to refine such designs within constraints. This model is then applied to circuits from the MCNC circuit benchmark set with validated failure rates for a contemporary FPGA device as a case study.

II. Background: Faults Impacting SRAM-based FPGAs

FPGAs are subject to two main categories of faults: Soft and Hard faults. Soft faults are mainly due to SEU which are caused when a high-energy particle such as proton, neutron, alpha, or heavy ion strikes a storage element e.g. LUT, IOB, Flip-Flop, etc. This fault is manifested by a logical value inversion of that element. When the SEU occurs in the datapath flip-flops, it is transient in the sense that it only affects the data being processed at the time of the SEU and typically dissipates afterwards. On the other hand, if the SEU impacts memory elements and latches, it causes the design to become modified and malfunction and changes the stored value while it engenders transient glitches at the output of combinational circuits as a result of elastic and inelastic scattering [7], and hence referred to as a *Firm Soft Fault*. Firm soft faults can be readily recovered by reprogramming the device with the original configuration using the recovery process known as *scrubbing* [8]. However, firm soft faults in critical resources such as the reconfiguration control module could disrupt any further scrubbing attempts and hence require total system re-initialization which may not be feasible during a mission-critical deployment. We refer to such faults as *Persistent Soft Faults*. These are treated as permanent hard faults from reliability point of view [9], as described below.

Hard faults, on the other hand, entail permanent physical damage to the device. There are three main causes of hard faults: manufacturing defects due to process imperfections which manifest as Infant Mortality failures, Total Ionization Dose (TID) radiation-induced faults which accumulate over the mission lifetime, and aging-related faults due to component degradation during the mission [10]. Prominent aging-induced failure concerns in sub-90nm technology include *Electromigration (EM)* and *Time-Dependent Dielectric Breakdown (TDDB)*. EM is the phenomenon of electron depletion in narrow conductors with increased temperature. This creates a highly resistive path which entails significant interconnect delays that cause violation of timing constraints or can result in open circuit conditions, i.e. "stuck at open" faults [3]. TDDB occurs when electrons are trapped in the gate oxide which can create a low impedance or "short circuit" path which results in incorrect or sluggish transistor switching characteristics. The TDDB failure rate increases at high temperatures and thin oxide layers prevalent in current FPGA devices [11][12].

In this paper, soft faults will not be considered in our analysis due to their transient nature and straightforward resolution using scrubbing. Therefore, the analysis herein concentrates on aging-induced faults due to their importance [13][14]. The proposed approach invokes evolutionary refurbishment techniques that involve reconfiguring the logic fabric to avoid use of failed components. Hard faults may occur during the operational phase or constant-failure rate region of the "bathtub curve". Since we focus on sustainable mission applications in this work, we concentrate on the wear out period for FPGA devices.

III. Self-Repair With Genetic Algorithms

Genetic Algorithm (GA) based refurbishment proceeds by evolving distinct FPGA configurations which maintain throughput that is dictated by mission requirements despite the presence of failing resources. The role of GA in this work is during the repair phase, to apply genetic operators to search for alternative configurations which sustain functionality. The default configuration generated by the vendor's synthesis tool is mapped onto a GA chromosome. This chromosome is then manipulated by invoking genetic operators such as mutation or cell-swap as developed below. In this study, the operators are constructed to maintain the integrity of the functionality, as evolving a design from scratch incurs high computational complexity.

The mutation operator is designed to manipulate the inputs of the LUTs to search for segments based on their intrinsic operational performance. This permutation of the inputs may help to utilize some inherent redundancy in the LUT which was created at the synthesis stage. Similarly, the *cell-swap operator* seeks to swap distinct LUT blocks while maintaining the overall functionality as illustrated in Figure 1. The swap will exchange all the LUT input interconnections and LUT contents, as well as the physical location. After swapping, the two LUTs will implement different functionality and have different input lines as shaded below. Thus, some fully occupied LUT may be swapped with some partially occupied LUT and find alternative physical resource to recover from the fault impact.

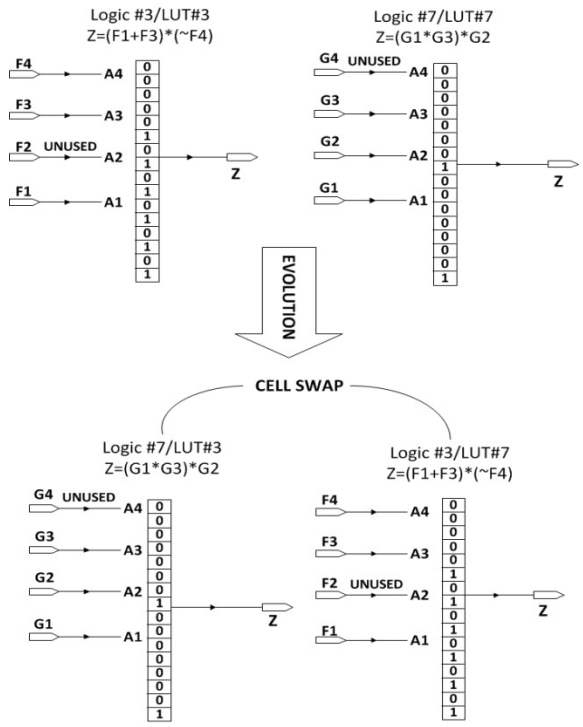

Fig. 1. Cell-swap: GA operator for refurbishment of FPGA configurations.

IV. Modeling Sustainability for Evolvable Hardware Systems

In biological systems, *sustainability* refers to an organism's ability to endure and remain viable by utilizing the resources within its environment. Likewise, a mission-critical electronic system is bounded within a finite domain of physical resources provided at design time. These resources cannot proliferate nor regenerate, yet can be reconfigured as demonstrated herein.

Figure 2 depicts the functional view of the *Sustainable Amorphous Resource Allocation (SARA)* model developed in this work. The first input is the *design resource information* which provides details of the design's FPGA resources which are subject to the faults considered in the analysis. The second input is the *probability distribution* of considered faults which impact the mission. The third input is the *repair policy* information. It includes parameters such as detection and refurbishment latencies and depreciation. The fourth input is the *availability threshold* which represents the minimum availability level below which the mission fails. The last input is the *mission duration*. The outcome from the model is the *quantity of amorphous reconfigurable resources* which must be budgeted in order for the subject design to survive throughout its mission lifetime. Alternatively, the model developed can output the *maximum mission lifetime* which an FPGA can sustain a desired availability threshold for a given provision of amorphous resources.

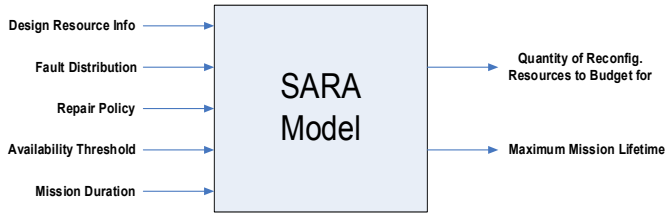

Fig. 2. Sustainability Model Functional Block Diagram.

The utility of the model is when planning a deployment in fault prone environment, the system can be designed to sustain operation despite the failures throughout its mission lifetime. In particular, the class of systems considered by the SARA Model is that of a finite resource system with reconfigurable yet non-regenerating resources.

A system can be survivable if and only if the number of resources available for fault refurbishment at time *n*, equals or exceeds the number of resources required for fault recovery throughout the mission lifetime **T** as shown in (1). Under the most favorable conditions, the refurbishment mechanism always achieves fault repair given sufficient resources. The fact that this capability to repair degrades over time as the system experiences faults is taken into account in the subsequent discussion:

$$R_{avail}(n) \geq \sum_{n}^{n+T} R_c(t) \tag{1}$$

Where, $R_{avail}(t)$ represents the number of resources available for repair as function of time and $R_c(t)$ represents the resources consumed as a function of time.

Equation (1) can be re-written in terms of a ratio as follows:

$$\frac{R_{avail}(n)}{\int_{t=n}^{n+T} R_c(t).dt} \geq 1 \tag{2}$$

Note, here the summation is transformed into an integral by assuming the repair process occurs in continuous time instead of discrete steps. Now, $R_c(t)$ can be approximated by using a probabilistic model. Let $\widetilde{R}_c(t)$ be the *estimated* number of resources to be consumed on system recovery at time *t*. Given the fault *probability density function (pdf)* and the cost C_i associated with the fault event we obtain an estimate of the number of resources consumed on system recovery over T:

$$\int_{t=n}^{n+T} \widetilde{R}_c(t).dt = T.\sum_{i=1}^{I} \lambda_i.C_i \tag{3}$$

Where λ_i is the rate of the fault of the type "*i*" and "*I*" is the number of failure modes. The fault rate is the reciprocal of the *Mean-Time-To-Failure (MTTF)*. Considering equation (2) and substituting for $\widetilde{R}_c(t)$ from (3):

$$\frac{R_{avail}(n)}{T.\sum_{i=1}^{I} \frac{C_i}{\int_{n}^{\infty} t.f_i(t).dt}} \geq 1 \tag{4}$$

Consider below R_d as total number of resources actually utilized by the design. If *f(t)* represents the resource fault *pdf* instead of the design fault *pdf*, equation (4) becomes:

$$\frac{R_{avail}(n)}{T.\sum_{i=1}^{I} \frac{C_i.R_d}{\int_{n}^{\infty} t.f_i(t).dt}} \geq 1 \tag{5}$$

Equation (5) does not take into account the fact that unutilized resources are fault prone too. Hence, it becomes:

$$\frac{R_{avail}(n)}{T.\sum_{i=1}^{I} \frac{C_i.[R_d + R_{avail}(n)]}{\int_{n}^{\infty} t.f_i(t).dt}} \geq 1 \tag{6}$$

Equation (6) hereafter is called the *Sustainability Test Ratio (STR)*. It holds true under the following assumptions that:

(a) Faults are independent,

(b) Successful reparability given sufficient unutilized resources,
(c) Constant fault arrival rate occurs, and
(d) MTTR < MTTF, otherwise the system becomes unavailable by definition.

Herein, the "Exponential Failure Law" is assumed in which the fault rate is assumed constant. Next, we demonstrate the utility of the SARA model through its application to determine availability and realize evolutionary repair of MCNC benchmark circuits implemented on a Xilinx Virtex-4 device.

V. REPAIRABILITY AND ITS EXTENTION TO SUSTAINABILITY

Repairability refers to the capability to recover at the incident of a fault. FPGA repairability degrades as the device undergoes faults during its operational lifetime. When system is placed under repair, throughput may be interrupted. Thus, besides repair, it is important for mission-critical systems to maintain availability. System availability in this context is the percentage of time that useful throughput is provided when exposed to the fault types as given below:

$$Availability = 1 - \left(\frac{MTTR_{TDDB}}{MTTF_{TDDB} + MTTR_{TDDB}} + \frac{MTTR_{EM}}{MTTF_{EM} + MTTR_{EM}} \right)$$

Herein, we will only consider TDDB and EM hard faults as discussed earlier. TDDB and EM faults are repaired using evolutionary techniques as described earlier in Section III. The impact of aging-induced TDDB and EM increases with time. The system undergoes hard faults due to these exposures as time elapses and that decreases the number of possible solutions for the repair mechanism to restore lost functionality. This leads to increasing MTTR and hence a decaying system Availability over time. The increase in MTTR depends on the repair mechanism. This behavior can be modeled as an exponential increase over time as shown in (8)

$$MTTR(t) = MTTR_0 e^{\eta \lambda t} \tag{8}$$

Where $MTTR_0$ is the initial mean time to repair at the beginning of the mission, η is the reparability depreciating coefficient and λt is the cumulative number of faults which have occurred up to time *t*.

Now, if the minimum acceptable availability for a given mission is denoted by $Avail_{thr}$, then $Availablity(t) \geq Avail_{thr}$, $t \in [0, T_{\max}]$ is desired. Thus, (7) can be re-written as:

$$Avail_{\min} = 1 - \left(\frac{MTTR_{TDDBo} e^{\eta_{TDDB} \lambda_{TDDB} T_{\max}}}{MTTF_{TDDB} + MTTR_{TDDBo} e^{\eta_{TDDB} \lambda_{TDDB} T_{\max}}} + \frac{MTTR_{EMo} e^{\eta_{EM} \lambda_{EM} T_{\max}}}{MTTF_{EM} + MTTR_{EMo} e^{\eta_{EM} \lambda_{EM} T_{\max}}} \right)$$

The above equation is then solved to obtain $T_{\max}$ which represents the maximum lifetime which maintains $Avail_{min}$. If only one fault type is considered for example, $T_{\max}$ can be simply calculated using:

$$T_{\max} \leq \frac{1}{\eta \lambda} \ln \left[\frac{1 - Avail_{\min}}{Avail_{\min}} \cdot \frac{MTTF}{MTTR_0} \right] \tag{10}$$

As a result, a system is anticipated to be survivable if and only if $T \leq T_{max}$ and STR ≥ 1. Once derived and instantiated with the failure rates of interest for the FPGA device on hand, the model can be applied routinely as demonstrated below.

VI. MCNC BENCHMARKS CASE STUDY

To illustrate the sustainability model, we consider the circuits in MCNC benchmark set. Table I lists the resource utilization numbers of the benchmark circuits implemented on Xilinx Virtex-4 XC4VSX35 FPGA device.

TABLE I. MCNC BENCHMARK CIRCUITS ON VIRTEX-4 FPGA

Circuit	Slices	LUTs	IOB
alu4	331	645	22
spex2	459	904	41
spex4	441	775	28
ex1010	452	754	20
misex3	357	672	28
seq	480	895	76
spla	482	890	62
pdc	338	616	56

As we are targeting harsh environment and stressful operating conditions, we obtained the *MTTF* numbers for *TDDB* and *EM* faults for a 90-nm technology node from [10, 13]. While the SARA model supports arbitrary failure rates, for the sake of analysis, we considered the following failure rate values for typical deployment environments: a conservative set of values for λ_{TDDB} at 1% per year and λ_{EM} at 0.2% per year.

Without the loss of generality, we will assume that all of the configuration bits are "essential bits" i.e. bits that make the design erroneous when flipped. If desired this conservative assumption that can be replaced by a de-rating factor using tools which identify essential bits such as *COSMIC, SEUPI,* or *Essential Bit Technology* from Xilinx. Moreover, we will assume that $C_i = 1$ for both EM and TDDB. This means when a fault occurs, it affects a single resource instance rather than multiple resources simultaneously.

In the GA used, the circuit is divided into *N* groups of contiguous resources called *Tiles*. Each tile has a *Concurrent Error Detection (CED)* mechanism to detect erroneous outputs. GA convergence time grows superlinearly with increased number of genomes in the chromosomal representation [15]. Hence, partitioning the design into multiple tiles, each evolved separately, substantially reduces the GA scalability issues. Redundant resources are sparse across the design in *Amorphous Resource Pool* (*ARP)* arrangement. The sparsity of the resources may lead to geographical limitations when reconfiguring and can limit the GA ability to find new designs. Resources in ARP do not have a designated functionality at design time. GA makes use of ARP resources to restore lost functionality by evolving a new functional bitstream from the FPGA fabric. A C++ simulator was built to evaluate the GA convergence time for a tile of 40-LUTs with 1 to 8 faults. The tuned GA parameters which were used for experiment are listed in Table II. They were selected based on preliminary runs to evaluate the optimal set of parameters.

TABLE II. ARP-BASED GA PARAMETERS

Parameter	Value
Population Size	50
Mutation Rate	0.5%
Crossover Rate	60%
Tournament Size	5
Elitism	2

The GA convergence time is translated from simulation generations into intrinsic evolution time using coefficients previously obtained in [16]. An *Arena* discrete simulation model was built for each MCNC benchmark to evaluate the reparability decay based on GA simulations. MTTF and MTTR results with considered failure rate are listed in Table III.

First, the model was applied to calculate T_{max} for several $Avail_{Thr}$ values [80% - 99.6%] where complete refurbishment was mandated given the conservative deployment parameters listed in Table III. The results are depicted in Figure 3. We also used the model to calculate R_{avail} lower bound values required to sustain function given the corresponding T_{max} values. The results are depicted in Figure 4. As can be seen from the results, as the $Avail_{Thr}$ is relaxed to lower values, the mission can endure longer durations. For instance, **spex2** benchmark deployed in such an environment and with the aforementioned GA technique, and $Avail_{Thr}$ of 99% is anticipated to survive for 5 years during which it will require around 50 un-utilized reconfigurable resources for repair. The same mission survivable duration drops down to 1 year: $Avail_{Thr}$ of 99.6%.

TABLE III. MCNC BENCHMARK CIRCUITS ARP-BASED GA REPARABILITY DECAY (CONSERVATIVE)

Circuit	Conservative: λ_{TDDB}=1%, λ_{EM}=0.2% *Time unit: years*			
	$MTTF_{TDDB}$	$MTTR_{TDDB}(t)$	$MTTF_{EM}$	$MTTR_{EM}(t)$
alu4	0.155	$1.97e^{0.2214t}$	0.7752	$1.97e^{0.0443t}$
spex2	0.1106	$2.95e^{0.1783t}$	0.5531	$2.95e^{0.0357t}$
spex4	0.129	$2.77e^{0.1904t}$	0.6452	$2.77e^{0.0381t}$
ex1010	0.1326	$2.76e^{0.1852t}$	0.6631	$2.76e^{0.037t}$
misex3	0.1488	$2.66e^{0.1709t}$	0.744	$2.66e^{0.0342t}$
seq	0.1117	$2.25e^{0.2307t}$	0.5587	$2.25e^{0.0461t}$
spla	0.1124	$2.24e^{0.2294t}$	0.5618	$2.24e^{0.0459t}$
pdc	0.1623	$2.86e^{0.1687t}$	0.8117	$2.86e^{0.0337t}$

It can also be inferred from Figure 3 and Figure 4 that in general the missions with smaller designs are survivable for longer periods. Yet, they require budgeting more resources which is advantageous to increase sustainability by supporting additional refurbishment episodes.

Availability threshold requirements vary from one mission to another. For example, if the mission involves a real-time live broadcast such as audio/video conversations or surveillance missions in which continuous coverage is sought after, high availability threshold is required. Whereas, if it is a data collection and transmission task in which there is little time sensitivity, a relatively lower availability threshold can be tolerated. Moreover, although high availability thresholds might appear sufficient, the implied downtime might be substantial when taking the mission duration into consideration. For example, an availability threshold of 99% implies a downtime of 15-minutes a day, or 876-hours in a 10-year mission, or a complete 1 year in a 100-year mission.

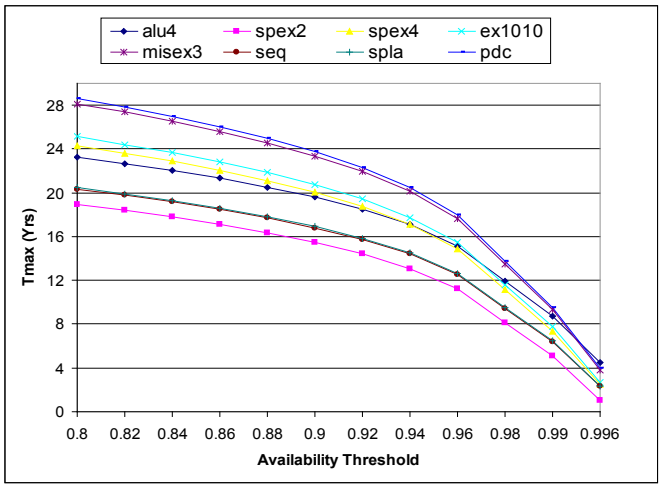

Fig. 3. MCNC Benchmarks Tmax vs. Availability (QOR: 100%).

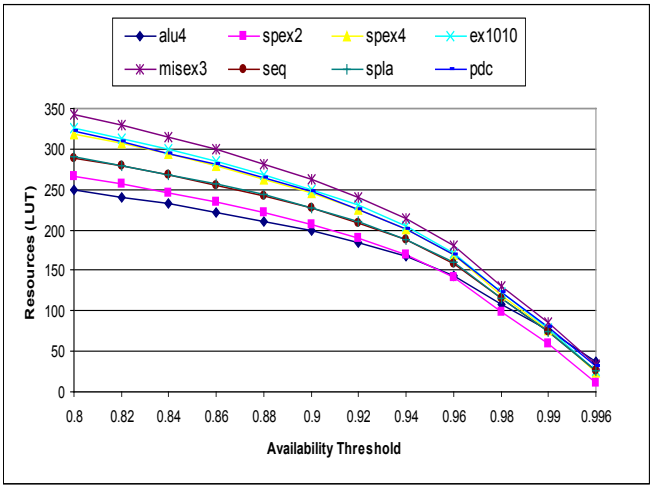

Fig. 4. Resource Required for Refurbishment (QOR: 100%).

Another important attribute to consider is the *Quality-Of-Refurbishment (QOR)*, which represents the fitness level at which refurbished design is qualified for functional operation. A QOR characteristic for FPGA self-repair is analogous to a Quality-of-Service metric in computer communication networks. In many FPGA repair cases, a mission can still make use of a partially-refurbished design [17]. For example, in video processing applications the system may be useful despite a few erroneous pixels within a video frame [18]. Since fitness is what guides the evolutionary search, the GA focuses on the genes of features that give the highest contribution to the fitness of individuals relatively quickly. This property can be observed from the results listed in the fourth column in Table IV. Figure 5 shows MCNC benchmark lifetimes are extended when repair process terminates once partial refurbished designs with QOR of 95% are evolved. Considering **spex2** for 99.6% $Avail_{thr}$, T_{max} increased from 1 to 12 years.

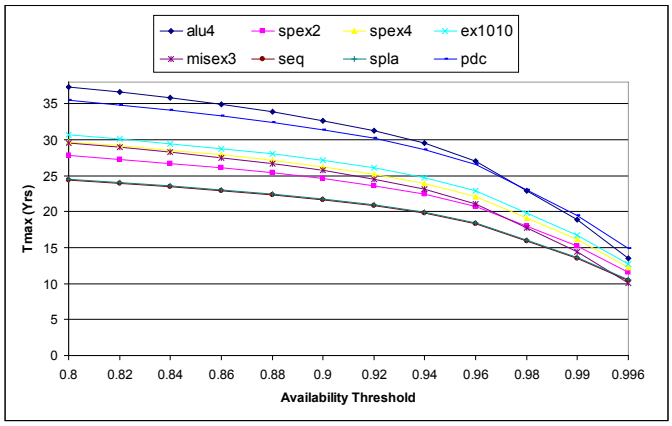

Fig 5. MCNC Benchmarks T_{max} versus Availability (QOR: 95%).

TABLE IV. ARP-BASED GA EVOLUTION RESULTS

# Faults	Ave. # Generations 95% Fitness	Ave. # Generations 100% Fitness	% of the GA Runtime to evolve 95% Fitness	# Runs
1	114	3962	2.88%	100
2	1230	31352	3.92%	50
3	3920	38601	10.16%	50
4	9238	63307	14.59%	30
5	11958	88746	13.47%	Interpolated (*Curve Fitting*)
6	19527	133248	14.65%	Interpolated (*Curve Fitting*)
7	31887	200066	15.94%	Interpolated (*Curve Fitting*)
8	51981	290643	17.88%	10

VII. CONCLUSION

In this paper, the SARA quantitative stochastic sustainability model for FPGA-based repairable systems is formulated to provide a novel mission-resource estimating approach and toolset. It estimates the quantity of resources at design-time required for refurbishment in order to meet mission availability, quality, and lifetime requirements in a given fault-prone ecosystem. This model is applied to circuits from the MCNC benchmark set with variations of parameters for illustration. Results show the estimated capability of these designs to sustain operation in realistic environments with the means of GA-based evolutionary repair. Should another repair mechanism be considered, similar experiments would be conducted to evaluate the reparability decay expression and so that their characteristics can then be substituted into the sustainable model.

REFERENCES

[1] I. A. Troxel, et al., "Flexible Fault Tolerance Using the ARTEMIS Reconfigurable Payload Processor," Military and Aerospace FPGA and Applications (MAFA), Palm Beach, FL, Nov. 27 - 29, 2007.

[2] G. Seagrave, "SpaceCube: A Reconfigurable Processing Platform for Space" *Military and Aerospace Programmable Logic Devices (MAPLD) Conference*, Annapolis, Maryland, September 15-18, 2008.

[3] JEDEC, "Failure Mechanisms and Models for Semiconductor Devices," JEDEC Publication JEP122-B. JEDEC Solid State Technology Association, Aug. 2003.

[4] E. Stott, P. Sedcole, P. Cheung, "Fault tolerance and reliability in field-programmable gate arrays," *IET Computers & Digital Techniques*, vol.4, no.3, pp.196-210, May 2010.

[5] J.A. Walker, M.A. Trefzer, S.J. Bale, A.M. Tyrrell, "PAnDA: A Reconfigurable Architecture that Adapts to Physical Substrate Variations," *IEEE Transactions on Computers*, vol. 62, no. 8, pp. 1584-1596, Aug. 2013.

[6] M. G. Parris, et al., "Progress in Autonomous Fault Recovery of Field Programmable Gate Arrays," *ACM Computing Surveys,* Vol. 43, Issue 4, pp 1 – 21, December, 2011.

[7] Ramanarayanan, Rajaraman, et al. "Modeling soft errors at the device and logic levels for combinational circuits." *IEEE Transactions on Dependable and Secure Computing,* pp. 202-216, 2009.

[8] M. Garvie and A. Thompson, "Scrubbing away transients and Jiggling around the permanent: Long survival of FPGA Systems through evolutionary self-repair," in *Proceedings of 10th IEEE International On-Line Testing Symposium*, pp. 155-160, July 12-14, 2004, Funchal, Madeira Island, Portugal.

[9] S. Straulino, "Results of a beam test at GSI on radiation damage for FPGAs Quick-Logic QL12x16BL and Actel 54SX32," Available at: http://ams.cern.ch/AMS/Beamtest/doc/tof.pdf, 2000.

[10] C. Bolchini and C. Sandionigi, "Fault Classification for SRAM-Based FPGAs in the Space Environment for Fault Mitigation," *IEEE Embedded Systems Letters,* vol. 2, pp. 107-110, 2010.

[11] J. R. Carter, et al., "Circuit-level modeling for concurrent testing of operational defects due to gate oxide breakdown," in Proceedings of Design, Automation and Test in Europe, pp. 300-305 Vol. 1, 7-11 March 2005, Munich, Germany.

[12] J. Srinivasan, et al., "The Impact of Technology Scaling on Lifetime Reliability," in *Proceedings of the International Conference on Dependable Systems and Networks*, Florence, Italy, June 28th - July 1, 2004.

[13] S. Srinivasan, et al., "Toward Increasing FPGA Lifetime," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, pp. 115-127, 2008.

[14] R. Wenjing, et al., "Toward Future Systems with Nanoscale Devices: Overcoming the Reliability Challenge," *IEEE Computer*, vol. 44, pp. 46-53, Feb. 2011.

[15] R. S. Oreifej, C. A. Sharma, R. F. DeMara., "Expediting GA-based evolution using group testing techniques for reconfigurable hardware," In *Proceedings of IEEE International Conference on Reconfigurable Computing and FPGA's,* pp. 1-8, Sept. 27 -29, 2006.

[16] R. S. Oreifej, et al., "Layered Approach to Intrinsic Evolvable Hardware using Direct Bitstream Manipulation of Virtex II Pro Devices," in *Proceedings of International Conference on Field Programmable Logic and Applications*, pp. 299-304, August 27 – 29, 2007.

[17] L. Sekanina and Z. Vasicek, "Approximate circuit design by means of evolvable hardware," in *Proceedings of IEEE International Conference on Evolvable Systems (ICES)*, pp. 21-28, April 16-19, 2013.

[18] R. Salvador, et al., "Self-Reconfigurable Evolvable Hardware System for Adaptive Image Processing," *IEEE Transactions on Computers*, vol. 62, no.8, pp.1481-1493, Aug. 2013.