A Hardware-based Evolutionary Algorithm with Multi-Objective Optimization Operators for On-Chip Transient Fault Detection

Marcel Merten* Sebastian Huhn*† Rolf Drechsler*†

*University of Bremen, Germany
{huhn,mar_mer,drechsle}
@informatik.uni-bremen.de

†Cyber-Physical Systems, DFKI GmbH
28359 Bremen, Germany

***Abstract*—Over the last years, the structure sizes of integrated circuits have significantly been decreased. This allows for the development of small, powerful, and energy-efficient circuits, as required for the challenging application scenarios like given in automotive or avionic systems. Nanometer scaled technology nodes are more vulnerable against transient faults, for instance, as induced by high radiation beams, potentially causing an erroneous behavior of the system. Different types of approaches have been proposed to increase the robustness of circuits against these faults, particularly for safety-critical applications. Such a countermeasure calculates, for instance, application-specific knowledge yielding a highly efficient fault detection mechanism that enhances the robustness significantly. Since these approaches invoke formal techniques for an advanced state analysis, a high computational effort is required, limiting the applicability for large circuit designs. This work addresses these shortcomings by combining an evolutionary algorithm with newly developed multi-objective optimization operators, deliberately designed for the state analysis of sequential circuits. The developed measures are all seamlessly integrated into one dedicated hardware module. By this, prototyping devices like field programmable gate arrays can be orchestrated during the regular circuit design flow to execute the proposed module to, in the end, benefit from an enormous hardware-acceleration. The experimental evaluation clearly proves that the presented method allows calculating application-specific knowledge effectively. More precisely, the run-time is reduced by more than 1,200X while retaining (or even improving) the efficacy of the resulting on-chip fault detection mechanism compared to state-of-the-art in terms of robustness enhancement and introduced hardware overhead.**

I. Introduction

During the last decade, a lot of research has been carried out in the field of electronic design automation and the development of nanometer-scaled technology nodes. The achieved progress allows for the design of highly complex *Integrated Circuits* (ICs) that are pervasive in nearly all aspects of daily life. It can be noted that the number of sequential elements, i.e., *Flip Flops* (FFs), in integrated circuits has been steadily increasing. However, the vulnerability of FFs against transient faults grows due to the nanometer-scaled feature sizes. A transient fault at a FF potentially flips the stored logic value for a short period of time if it is not logically, electrically, or temporarily masked. This may affect the output values and corrupts the IC's functional behavior. In safety-critical systems, these violations potentially lead to severe malfunctions and, hence, can cause large collateral damage. Application fields like aeronautic systems are influenced by the environment and exposed to high-energetic radiation, which implies that the system is even more prone to transient faults [1], [2].

978-1-6654-1060-1/22/$31.00

New mechanisms are required to decrease the vulnerability of ICs against transient faults, i.e., by detecting and reacting to an occurring fault. The *robustness* serves as an important metric for this, derived from the number of non-robust FFs being vulnerable to transient faults over all FF of the IC. For increasing the robustness of a sequential circuit, the number of vulnerable (non-robust) FFs needs to be decreased. Typically, the circuit is extended by introducing redundant hardware [3]–[6] or redundant time-delayed computation [7], [8] to *harden* the corresponding FFs. Even though these techniques work in principle, a significant hardware overhead or a measurable effect on the resulting latency is introduced, leading to high costs and reduced performance.

Due to the drawbacks of the existing approaches, an application-specific approach has been proposed in [9], [10] that determines application-specific knowledge to increase the robustness of the circuit while introducing only a slight hardware overhead compared to space-based approaches like *Triple Modular Redundancy* (TMR) and impacting the timing only negligibly. The application-specific knowledge is either calculated by the random approach [9], resulting in very low run-time, or by incorporating formal techniques to improve the results further, as done in [10]. However, using formal techniques yield an enormous run-time, which is infeasible for large circuits.

This work proposes an *Evolutionary Algorithm* (EA) with novel *Multi-Objective Optimization* (MOO) operators, deliberately designed for the state analysis of sequential circuits and seamlessly integrated into a hardware module. The proposed MOO-EA is meant to be executed for determining the application-specific knowledge using prototyping devices like *Field Programmable Gate Arrays* (FPGAs) during the regular circuit design flow. This knowledge is then fed back to the software flow of [10] to, finally, emit the improved gate-level netlist. MOO-EA increases the robustness without sacrificing the scalability to larger circuits. Thus, the shortcomings of existing techniques [9], [10] are mitigated. In particular, MOO-EA significantly reduces the run-time up to 1,200X while achieving comparable robustness enhancements.

The structure of this paper is as follows: Section II gives a brief introduction to required fundamentals and related works.

The proposed MOO-EA is presented in Section III and the experimental evaluation is presented in Section IV. Finally, a conclusion and outlook to future works are given in Section V.

II. Preliminaries

This section introduces EAs, sequential circuits, and the transient fault model. Furthermore, the technique [9] is shortly described that forms the basis of the proposed approach and, hence, is strictly required for the comprehension.

A. Evolutionary Algorithms

The general idea of EAs is about simulating the natural evolution process to solve an optimization problem. An EA encodes a population of individuals containing solutions for the optimization problem. An individual i can be represented as a triple $i = (G, Z, F)$ with G_i equals the genotype encoding the solution of the optimization problem, Z_i holds the additional information, for example, specific configurations for the individual, and F_i represents the fitness of i. The genotype is reflected by a set of genes, which are the substitutable atomic parts concerning the chosen encoding.

Typically, EAs implement *Genetic Operators* (GOs) to reproduce individuals, such as recombination and mutation, to optimize the resulting genotype. The recombination operator (cf. Equation (1)) recombines two or more genotypes to form a new individual and the mutation operator (cf. Equation (2)) randomly changes the genotypes yielding new genes that improve the genetic diversity. Hereby, genetic diversity is defined by the diversity of the genotypes inside the population and increases the probability of reaching a global optimum [11].

$$Recombination^{\Phi} : (G \times Z)^r \rightarrow (G \times Z), \text{ with } r \geq 2 \text{ and rand. state } \Phi \quad (1)$$

$$Mutation^{\Phi} : (G \times Z) \rightarrow (G \times Z), \text{ with rand. state } \Phi \quad (2)$$

EAs can incorporate additional side information about the addressed domain to improve the behavior of the GOs. Every individual i is evaluated by a fitness function $\psi(i)$ to assess the quality of the given solution. The individuals are passed to a selection operator that removes individuals. The truncation selection is a frequently used selection operator removing individuals with the worst fitness values. Finally, the remaining population is reproduced again to replace the removed individuals if the termination condition, e.g., a certain fitness value, is not met. Note that every reproduction cycle of the EA is called generation.

B. Sequential Circuits and Transient Faults

A sequential circuit ϕ can be represented as a gate-level representation and consists of *Primary Inputs* (PIs), *Primary Outputs* (POs), combinational gates G, and sequential elements, such as FFs, i.e., $\phi = (PI, PO, G, SE)$. Each FF stores a logical value for one clock cycle and, therefore, holds a specific FF state. All FF states considering one clock cycle define the related circuit state. Due to the increasing complexity of circuits, the feature size must be shrunk to meet the requirements. The shrinking feature size leads to an increased vulnerability of circuits against transient faults.

A transient fault is typically caused by single event upsets, e.g., electrical noise, particle strikes, or other environmental effects [1], [2]. Generally, a transient fault at a FF is modeled as an unintended toggled FF state, which potentially leads to an unspecified behavior of the circuit ϕ. Particularly for safety-critical applications, such a faulty behavior may lead to large collateral damage. The robustness is formally given in Definition 1.

Definition 1. *Robustness*
A FF ff_1 is called robust if an occurring (single) transient fault in ff_1 does not affect the input/output behavior of the circuit ϕ. The robustness of ϕ considers the average robustness of all FFs in ϕ and acts as a metric to evaluate the overall vulnerability against the considered fault model.

Different approaches have been proposed to improve the robustness – also known as *hardening* – of sequential circuits. The robustness increasing methods applied to the *Circuit-under-Hardening* (CuH) can roughly be categorized into space-based [3]–[6], timing-based [7], [8], and application-specific [12] approaches. Typically, such a hardening process involves newly inserted on-chip *Fault Detection Mechanisms* (FDMs) or *Fault Correction Mechanism* (FEM) that are meant to observe the circuit's behavior to, in the end, detect (or correct) transient faults. In particular, when orchestrating FDMs, a certain fault signal is raised to execute certain precautions in case of transient faults.

C. Enhancing Robustness of Sequential Circuits Using Application-specific Knowledge and Formal Methods

The method proposed in [9] is an application-specific approach specialized for single transient faults while introducing only a relatively small hardware overhead. The approach [9] determines application-specific knowledge to improve the circuit's robustness by inserting highly effective FDMs by heavily orchestrating formal techniques combined with binary decision diagrams. Basically, the (functional) equivalence between groups of FFs is being evaluated and takes advantage of redundancies that prevail in certain states of the circuit during its functional operation. Thus, functional equivalent FFs – in terms of their logical values – in specific states of the circuit are determined and used to deduce *Equivalence Properties* (EPs), as formally given in Definition 2.

Definition 2. *Equivalence Property*
Let $\mathcal{F}$ be the set of all FFs and $F_{tm} \subseteq \mathcal{F}$ an arbitrarily subset. Furthermore, let S be the set of all reachable states of the circuit and $S_{tm} \subseteq S$ an arbitrarily subset. The value of a FF $ff \in \mathcal{F}$ in a state $s \in S$ is noted by $ff(s)$. Then EP(S_{tm}, F_{tm}) is satisfied, iff.
$\forall ff_n, ff_m \in F_{tm} : ff_n(s) == ff_m(s), \forall s \in s_{tm}$.

The EP specifies the behavior of the circuit partially with respect to a set of covered states. Given an arbitrary set of states S_{tm} and a set of FFs $F_{tm} \subseteq \mathcal{F}$ satisfying the EP(S_{tm}, F_{tm}), the equivalence of all $ff \in F_{tm}$ for all $s \in S_{tm}$ can be evaluated on-chip during the functional

operation. The evaluation is performed by a dedicated logic block that consists of an activator and a comparator unit; for more details, please consider [9].

Both units form the holistic FDM, which is seamlessly integrated into the gate-level netlist of the design-under-hardening. The circuit can be extended with multiple FDMs, which are commonly contributing to the overall fault detection of arbitrary transient faults in FFs. Internally, the activator detects whether a circuit state currently prevails that satisfies an EP. Note that all these states are pre-calculated during the hardening phase for different sets of FFs using the state collector concept, as described in detail in [9]. If such an EP state is observed during the functional operation, the activator is enabled. In parallel, the comparator checks whether the set of FFs, for which a logical equivalence of their internal values can be assumed, is the same or not. The fault signal can then be derived as follows: If an EP state prevails and the set of FFs is not equivalent, a transient error must have been occurred in one FF of the considered set.

The effectiveness of the approach highly depends on the sets $F_{tm} \subseteq \mathcal{F}$. In [9], the FF sets are selected randomly, i.e., random FF sets $F^1_{tm} \subseteq \mathcal{F}$ of a predefined cardinality (partition size) p_s are calculated. If all FF sets of a specific cardinality are checked, the cardinality of F^1_{tm} is decremented. Definition 3 describes the decrement of cardinality in more detail. Note that each set F^1_{tm}, that holds at least one valid state $s \in S$ satisfying an EP, can be used to generate a FDM.

Definition 3. *Random approach cardinality decrement*
Let $F^{x+1}_{tm} \subset F^x_{tm}$, $\forall x \in \mathbb{N}\backslash\{0\}$ then all $F^x_{tm} \subset F^1_{tm}$, $\forall x \in \mathbb{N}\backslash\{0,1\}$ are excluded from the search. However, in case of $\forall s \in S : \neg EP(\{s\}, F^1_{tm})$, the EP is calculated on a subset $F^2_{tm} \subset F^1_{tm}$ only. The actual FFs to be removed from F^1_{tm} are calculated by a greedy-wise algorithm, which is done iteratively until a subset satisfies the EP on at least one state $s \in S$ or $|F^x_{tm}| < 2$ or no further FFs remain.

This approach leads to a low run-time, but the resulting robustness can be improved. Thus, an SAT-based metric for evaluation has been proposed in [10], allowing for calculating more effective FFs. However, the orchestration of formal technique results in high run-time, strictly limiting the scalability.

III. Evolutionary Approach

This work proposes a specialized hardware-based EA to determine application-specific knowledge of a sequential circuit allow for synthesizing highly effective FDMs, which enable the detection of transient faults. More precisely, the proposed approach significantly increases the robustness while consuming only negligible run-time and introducing a slight hardware-overhead. By this, the speed of the state-of-the-art greedy-like technique [9] and the effectiveness of the SAT-based technique [10] are combined. In the end, this allows enhancing the robustness of larger circuit designs, which was not possible earlier.

The overall data flow of the proposed system is given in Figure 1. The system stimulates the PIs of the CuH with pseudo-random inputs by an on-chip *Random Pattern Generator* (RPG). Each time a new pattern has been assigned to the PIs, the EA extracts the internal FF states of the circuit in the following clock cycle. The state is then internally used to evaluate the application-specific knowledge, which is encoded in the individuals. Afterwards, the EA signalizes the RPG to provide a new input pattern. This process is meant to be executed on a prototyping device to benefit from a massive hardware acceleration, which is substituted by using a logic simulator in this work. Internally, the EA encodes individuals $i \in I$ as a population. The genotype G_i of an individual i consists of the application-specific knowledge, being represented as a set of genes encoded as integer FF-IDs. Therefore, each FF of the CuH is numbered with its ID. The fitness F_i acts as a metric considering the number of states that satisfy the EP in one generation (for G_i) and, hence, approximates the robustness increase. Since there is no limited generation size due to the RPG, the number of observed states per generation is an additional parameter. Finally, the additional information Z_i encodes various information to optimize the genetic operators. In the following, the fitness function, GOs, and the additional information of this specific EA are explained in detail. The fitness function is essential for any EA to assess the individuals' quality. However, calculating the robustness in each generation leads to an infeasible run-time. If no exact fitness function can be performed with respect to the time constraints, an approximative function can be applied, for instance, by considering prediction intervals, as proposed in [13]. An appropriate metric for the application-specific knowledge is calculated by considering the EP. Consequently, if an observed state s, $EP(\{s\}, i.G)$ is satisfied, an individual i increases its fitness. Since genotypes of high cardinality cover relatively more FF-IDs, the corresponding FDMs detect more transient faults yielding higher robustness. More precisely, each individual i is evaluated with respect to $|G_i|$. The resulting fitness function $\psi(i, s)$ for an individual i in a state s, as stated in Equation 3.

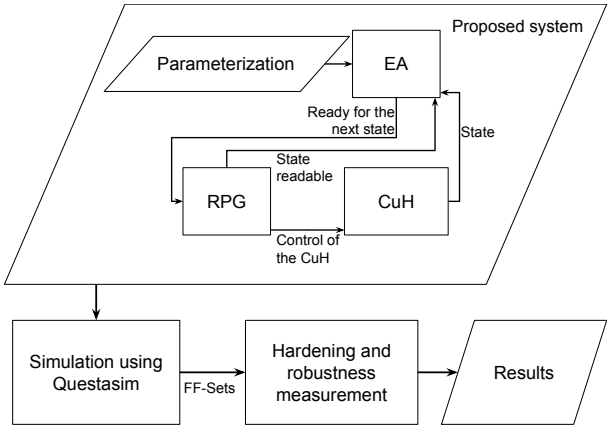

Figure 1: Dataflow of the proposed approach

$$\psi(i,s) = \begin{cases} |G_i|, & \text{if } |G_i| \geq 1 \,\&\, EP(\{s\}, G_i) \\ 1, & |G_i| == 1 \\ 0, & else \end{cases} \quad (3)$$

The fitness of an individual i is calculated by $\psi(i) = \sum_{x=1}^{|S|}(\psi(i, s_x)), s_x \in S$ with S the observed states in a generation. The genotype's maximum size needs to be predefined since the population is saved in (hardware) registers. Addi-

tionally, the GOs are adapted to the problem. The mutation performs operations that remove, add or replace one gene per generation. If the maximal size of the genotype is reached, remove or replace are applied. The recombination chooses two parents, whereby individuals with higher fitness have a higher probability of becoming a parent. The parents get recombined by either inherit a gene from the child or not. If the number of inherited genes reaches the genotype's maximum size, further inheritance is applied as a replacement. The selection operator is realized as a truncation selection. Therefore, the population is sorted by a hardware-based merge sort like [14] concerning the fitness values. Afterward, half of the population holding smaller fitness values is removed. The resulting steady-state behavior converges significantly faster in average [15].

For further improving the GO's performance, multiple extensions have been deliberately designed and implemented. This allows for determining additional information $i.F$ for every individual i that is being calculated and stored as follows:

Collision information – describes how often FF-IDs of a genotype violate the EP regarding observed states. A *Dominant Logical Value* (DLV) is calculated to determine the genes violating the EP. The DLV is the logical value that prevails in most FFs represented by a genotype concerning a specific state. For all genes representing FFs that do not hold the current DLV, the collision value is increased. This information is used to remove or replace genes with higher collision values more likely during the mutation and recombination.

State coverage information – reflects the number of FFs f in a genotype G_i – considering an observed state s – that satisfy EP({s},G_i), whereby Equation 4 holds.

$$\nexists j : f \in G_j, EP(\{s\}, G_j) \text{ and } (|G_j| > |G_i| \parallel (|G_j| == |G_i| \;\&\&\; F_j > F_i)) \quad (4)$$

Individuals without such genotypes are removed if the coverage information is used. Thus, the state coverage information influences the selection operator while being collected over a predefined number of generations (*state coverage memory*). This ensures reflecting state coverage information over multiple generations while removing deprecated information that is no longer valid.

Gene coverage information – describes if an individual i covers a gene g essentially, i.e., $(g \notin G_j || F_j < F_i), \forall j \in P$ holds. The gene coverage information is used to protect the individuals that have essentially covered genes such that they are not removed from the population. This method maximizes the number of covered FFs by the corresponding FDMs.

Subset fit-factor – is used to avoid multiple genotypes representing redundant application-specific knowledge. A genotype G_i is redundant if it equals an existing genotype, or $\nexists k$ with $G_i \subset G_k \subset G_j$ such that $F_k > F_j$. The *Subset Fit Factor* (SFF) indicates whether the subset G_i potentially becomes a subset G_k with higher fitness than F_j. An optimal G_{k-opt} holds the same number of EP satisfying states as G_i, and the maximum possible size equals $(|G_j| - 1)$. G_{k-opt} approximates the maximum possible fitness of any k with $G_i \subset G_k \subset G_j$. The maximum possible fitness is related with F_j, as follows:

$$SFF(G_i) = \frac{|\{s \in S | EP(\{s\}, G_i)\}| * (|G_j| - 1)}{F_j} \quad (5)$$

This value i is greater than 1 if the maximum possible fitness of one or more superset G_k – derived from G_i with $G_i \subset G_k \subset G_j$ – is higher than F_j. Consequently, if the subset fit-factor is used and the fit-factor is ≤ 1, the individual is removed from the population.

Validity – protects a subset G_i from being removed if it satisfies the EP in a state that was not covered by any superset in previous generations before. This preserves the knowledge of states satisfied by the individual G_i but not by any other superset. For the same reasons as the state coverage information, the validity is collected over a predefined number *validity memory* of generations.

During the execution of an EA, the population may be stuck in a local optimum, which prevents any further optimizations. Thus, a mechanism to ensure the exploration of new search space is required. Typically, complex methods like speciation technologies are used to evolve and protect different solutions in parallel, as proposed in [16]. Due to the hardware-based implementation, this work implemented a rather straight childhood principle. Childhood constantly evolves into new individuals with empty genotypes. These individuals are solely affected by the mutation operation and remain protected from any selection mechanisms of the population since, otherwise, they would be removed due to the already developed existing individuals. If an individual fails to improve its fitness for a predefined number of consecutive generations (*stagnation-border*), the individual is removed from childhood and added to the population.

Since the FDM's comparator unit observes the covered FFs output value by introducing a new fanout, the capacity of the corresponding signal gets slightly increases. This changes the latency and may require deviating cell types used during the layout synthesis, which is considered because each FF is meant to be covered only once by a genotype. Consequently, the genotypes of the calculated individuals have to be disjunctive.

Two different methods were implemented for controlling the individuals' number that covers a gene during the EA's execution, which are both described in the remainder.

Gene coverage without overlap – checks the gene coverage for each gene of an individual. If for all genes $g \in i.G$, i.e., no individual i_x with $g \in i_x.G$ exists, and $i_x.F > i.F$, the individual i covers it genes without overlaps and is therefore essential. Only essential individuals i are considered for the FDM synthesis if this mechanism is enabled.

Sequential calculation of FF-sets – sequentially calculates FF-sets by dividing the overall evolution into sequences of a user-defined number of generations. The best individual i_{best} is stored for each sequence and the genes $i_{best}.G$ are removed from the EA for all successive sequences.

Subsequently, all covered genes are removed from each genotype of the population while beginning with the geno-

type of the best individual. This principle ensures that all application-specific knowledge is extracted from the population. As the final step, the determined most beneficial sets of FF are then fed back into the software hardening flow, yielding the enhanced gate-level netlist.

IV. Experimental Evaluation

This section describes the experimental evaluation of the proposed MOO-EA and discusses the obtained results concerning the robustness increase and run-time compared to the random [9] and SAT-based [10] approach.

All experiments have been conducted on an *Intel i7700k* with *32 GB* system memory within a *QuestaSim* simulation environment. The proposed technique is implemented as a Simulink model, which is transformed to a register transfer level using *MATLAB 9.3* in combination *MATLAB HDL-Coder 3.11*. The same ITC'99 benchmark circuits of [9], [10] and the identical simulation-based robustness assessment tool-flow have been considered for a fair comparison. This tool-flow determines the robustness by injecting a transient fault in the CuH while comparing the signals at the POs only.

The proposed EA hardware is emitted as synthesizable hardware description language and meant to be implemented directly in hardware by using fast prototyping hardware like FPGAs. After the EA's execution, the calculated data are then fed back into the hardening (software) flow that generates the FDMs and, finally, emits the enhanced gate-level netlist of the CuH. This netlist can then be directly passed through, like the place & route.

The considered maximum of genes per genotype is set to 16 FFs, reflecting the chosen partition sizes of [9], [10]. The population's size is chosen concerning the number of FFs in the CuH. The proposed approach can easily be applied to an arbitrary circuit with any amount of FFs. The considered circuits have up to 245 FFs, whereas the EA can produce overlaps during execution and, hence, the population size is set to 200 individuals. Further parameters to either determine additional information or control the childhood operation exist, which have been evaluated by various runs with stepwise adjusted parameters, such as the sequence length. In each generation, 128 states are observed from 128 randomly generated stimuli. This parameter influences the resulting size of the evolved genotypes and, hence, is chosen based on the following observations. Smaller genotypes have fewer genes to collide resulting in more EP satisfying states. However, larger genotypes have higher fitness increases per EP satisfying state. Thus, the results with fewer observed patterns tend to calculate larger genotypes, and the genotypes' size is shrinking when considering more observed states per generation. Both variants lead to FDMs yielding lower robustness values. Since the recombination impacts the resulting child individual stronger than the mutation, five recombined children per generation are considered. The number of individuals developed in childhood is set to five yielding a sufficiently sized population to optimize the application-specific knowledge while probably overcoming local optima. The stagnation-border of the childhood is set to three, which has proven itself as an appropriate number of generations to protect the children. Higher stagnation-borders usually protect the children too long and, therefore, prevent the childhood from evolving new children. The state coverage information and validity memory are both set to ten generations.

TABLE I: Stage-I results

Approach \ Values	R [%]	HWO [%]	HT [s]
EA	93.8	19.47	2.71
random	85.22	8.05	9.31
SAT-based	95.86	6.46	2,985.90

Evaluation: The following values been obtained: the robustness of the original circuit (*orig. R*), the robustness of the enhanced circuit (*R*), and the *Hardening Time* (HT) in terms of the EA execution time (*HT[EA]*). Note that the hardening run-time includes all steps from the analysis to the generation of the enhanced gate-level netlist of the CuH, orchestration the newly inserting FDMs. The EA execution time has been normalized from the simulation environment with respect to the usual FPGAs' functional clock frequency of approx. 200Mhz.

Table I presents the average results of all conducted experiments, including ten different configurations to perfectly adjust the EA concerning different circuit's characteristics. Considering the random- and SAT-based approach, the conducted experiments consider partition sizes of 8 and 16. The results clearly demonstrate that the achieved *R* values outperform the results of the random approach [9]. In particular, the proposed technique achieves outstanding results for circuit *b14* - the *Hardware Overhead* (HWO) of the EA is 19.47% higher than any other approach.

In order to determine a baseline for the following evaluation, a *standard* EA without any MOO-EA technique, has been implemented. The considered configurations are as follows:

standard – the configuration for the baseline comparison; all additional information, the childhood and both integrated methods to ensure disjoint genotypes are deactivated.

mod-1 – this setting enables all additional information and techniques except state coverage information and childhood. Furthermore, the gene coverage (without overlaps) is activated to ensure the best individuals are disjoint.

mod-2 – this run equals mod-1, but instead of the gene coverage without overlap, the sequential calculation of FF-Sets is activated and set to 20 generations.

The benchmark circuits strongly vary in their structural characteristics and, hence, it is unlikely to determine one global configuration for the EA to cover all circuits best at once. Thus, for a fair comparison against the previous works, only the two most beneficial configurations are considered. Consequently, the configuration with the most beneficial *R* have been chosen for each circuit.

Figure 2 visualizes the *R* of the two most effective configurations of the MOO-EA, the random and SAT-based approaches, and the standard-EA (baseline). The standard-EA achieves significantly lower robustness values than the random approach for all circuits except b06. In contrast to this, the newly proposed MOO-EA achieves better results than the random approach in any circuit except the b07, in

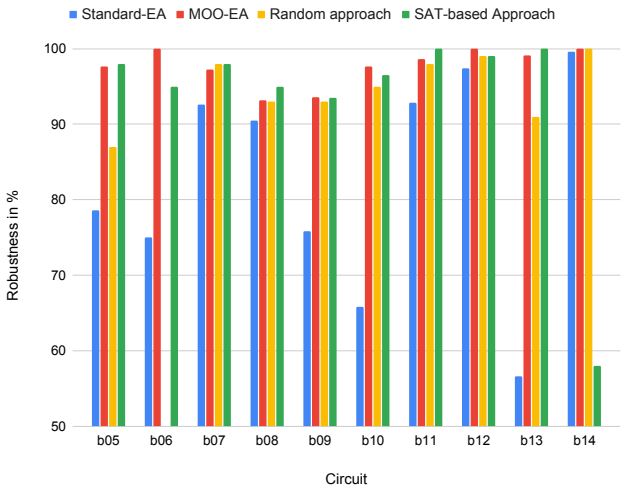

Figure 2: Robustness comparison

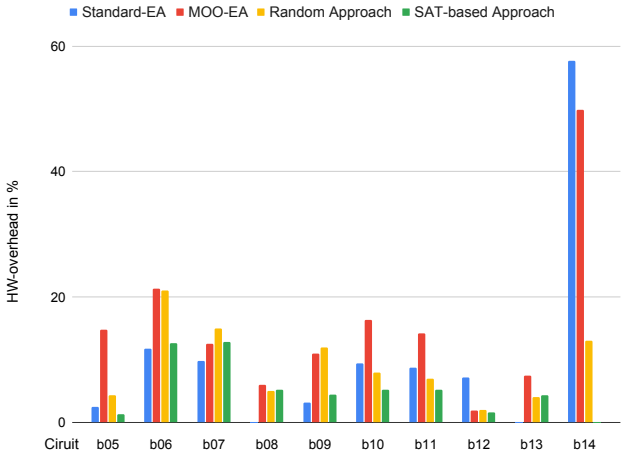

Figure 3: Hardware overhead comparison

which a slight decrease occurs. *b07* implements a counter and, hence, has a deviating behavior compared to the others. Thus, different configurations for the EA would be required. Compared to the SAT-based approach, the MOO-EA achieves better results for three of the circuits and equal results for b09.

The standard-EA generally introduces a small HWO comparable to the HWO of the random- and SAT-based approach in most circuits, as visualized in Figure 3. The HWO of the MOO-EA is slightly higher than the values of the standard-EA. The main difference to the existing approaches can be observed at the *b14*, which is not processible by the SAT-based approach due to infeasible run-time. The MOO-EA allows determining of more partitions as done by the random approach. In turn, the additionally observed partitions do not further improve robustness since approx. 100% robustness – concerning the robustness assessment technique – is already achieved but measurably increases the resulting HWO.

Finally, Figure 4 presents the *HT*. The EA benefits from its HW-based character allowing for a massive hardware acceleration during the state space exploration of the CuH.

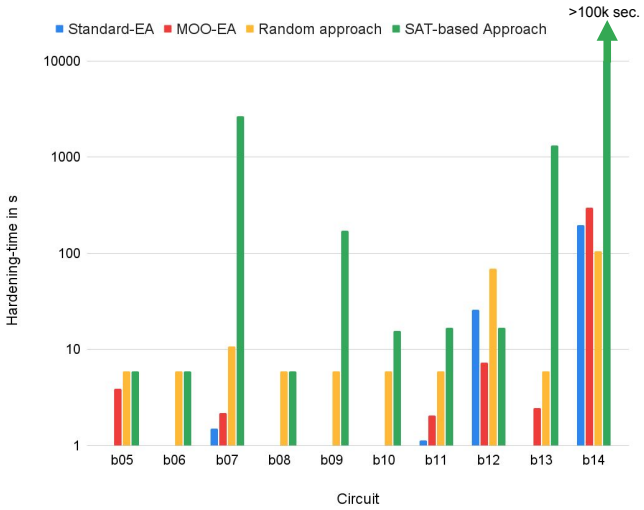

Figure 4: Hardening time comparison

Multiple hundreds of generations are passed in milliseconds, outperforming a similar offline EA-based approach by far. The proposed approach only needs up to 0.05 seconds for all of the considered circuits. The rest of the *HT* is caused by the simulation-based robustness assessment tool-flow. The smallest average *HT* is achieved by the standard-EA (except for b12), followed by the optimized-EA. As indicated earlier, the SAT-based approach exceeds the *HT* threshold of 100k seconds. However, the newly proposed optimized-EA allows processing *b14* within minutes. More precisely, a speed-up by 1,215X has been achieved for *b13* while an average factor 251X is accomplished considering all benchmarks. This circumstance clearly proves the superiority of the developed optimized-EA over existing approaches.

V. Conclusions

This paper presented a hardware-based EA orchestrating novel multi-objective optimization operators to significantly increase the robustness of sequential circuits against transient faults. In the end, the proposed approach achieved a robustness increase nearly as good as the SAT-based technique's result while achieving a massive speed-up of up to 1,200X. The proposed technique provides an effective way to conduct application-specific knowledge and, hence, enabled for the first time an increase of robustness comparable to the SAT-based approach for larger ICs. Future work will focus on a deeper analysis of the observed states to further optimize the EA. Additionally, more pessimistic robustness measures will be considered, for instance, regarding silent data corruptions, and, hence, the benefit when using the proposed approach will be even more significant.

VI. Acknowledgements

This work was financially supported by the German Federal Ministry of Education and Research BMBF under the framework of VE-CirroStrato and the AI initiative of the Free Hanseatic City of Bremen. We would like to thank Verific Design Automation Inc. for providing the SystemVerilog front-end used for the implementation of our technique.

REFERENCES

[1] T. Heijmen and A. Nieuwland, "Soft-error rate testing of deep-submicron integrated circuits," in *IEEE European Test Symp.*, 2006, pp. 247–252.

[2] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, 2005.

[3] C. E. Stroud and A. E. Barbour, "Design for testability and test generation for static redundancy system level fault-tolerant circuits," *International Test Conference*, pp. 812–818, 1989.

[4] Z. Luo, "ECC, an extended calculus of constructions," *Symposium on Logic in Computer Science*, pp. 386–395, 1989.

[5] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[6] A. Sanchez-Macian, P. Reviriego, and J. A. Maestro, "Hamming SEC-DAED and extended hamming SEC-DED-TAED codes through selective shortening and bit placement," *IEEE Transactions on Device and Materials Reliability*, vol. 14, no. 1, pp. 574–576, 2006.

[7] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," *Microarchitecture*, pp. 7–18, 2003.

[8] D. Blaauw, S. Kalaiselvan, K. Lai, W.-H. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull, "Razor II: in situ error detection and correction for PVT and SER tolerance," in *IEEE International Solid-State Circuits Conference*, 2008, pp. 400–622.

[9] S. Huhn, S. Frehse, R. Wille, and R. Drechsler, "Enhancing robustness of sequential circuits using application-specific knowledge and formal methods," *ASP Design Automation Conference*, pp. 182–187, 2017.

[10] ——, "Determining application-specific knowledge for improving robustness of sequential circuits," *IEEE Transaction on VLSI Systems*, vol. 27, no. 4, pp. 875–887, 2019.

[11] J. Morrison and F. Oppacher, "Maintaining genetic diversity in genetic algorithms through co-evolution," *Advances in Artificial Intelligence*, pp. 128–138, 1998.

[12] S. Hertz, D. Sheridan, and S. Vasudevan, "Mining hardware assertions with guidance from static analysis," *IEEE Transaction on CAD of Integrated Circuits and Systems*, vol. 32, no. 6, pp. 952–965, 2013.

[13] C. Plump, B. J. Berger, and R. Drechsler, "Improving evolutionary algorithms by enhancing an approximative fitness function through prediction intervals," in *IEEE Congress on Evolutionary Computation*, 2021, pp. 127–135.

[14] Gayahathri K and Harshini V S, and Dr Senthil Kumar K K, "Hardware implementation of sorting algorithm using fpga," *International Journal of Advance Research and Innovative Ideas in Education*, vol. 4, no. 2, pp. 719–725, 2018.

[15] D. Corus, D.-C. Dang, A. V. Eremeev, and P. K. Lehre, "Level-based analysis of genetic algorithms and other search processes," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 707–719, 2017.

[16] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 19–46, 2007.