Reliability Engineering and System Safety 205 (2021) 107223

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

Exact and efficient reliability and performance optimization of synchronous task graphs

Reza Ramezani[a,*], Abolfazl Ghavidel[b], Yasser Sedaghat[b]

[a] Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran
[b] Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

ARTICLE INFO

Keywords:
Synchronous task graphs
Optimization
Reliability
Performance
Makespan
FPGA

ABSTRACT

SRAM-based FPGAs have found many applications in modern computer systems. In these systems, high-performance computing applications are executed as task graphs in which reliability and performance are crucial constraints. In this paper, an exact method is presented to efficiently optimize the reliability and performance of synchronous task graphs running on SRAM-based FPGAs in harsh environments. Solving this optimization problem leads to the generation of a true Pareto set of Fault Tolerance (FT) techniques. Each solution of this set determines FT techniques of the tasks and leads to specific reliability and makespan. Thus, this solution set trades off between reliability and makespan, and one of the solutions that best meets system requirements can be applied to the running tasks to optimize the reliability and performance. The proposed technique is novel as it obtains the true Pareto set of FT techniques of the whole task graph by partitioning the task graph into its segments, optimizing different segments separately, and joining the obtained solutions. This partitioning strategy leads to reduce the computation time significantly. In this paper, it is mathematically proved that the proposed partitioning strategy generates global optima from the local ones without losing any optimal solutions. The experiments show that the proposed technique improves the MTTF of real-world and random task graphs by 46.30% on average without any negative effects on the performance. Then, the efficiency of the computation time of the proposed technique is demonstrated by conducting several experiments on small-, medium-, and large-size synchronous task graphs and comparing the results with other exact and evolutionary optimization methods. Finally, supplementary experiments in dynamic environments show that the proposed technique outperforms adaptive state-of-the-art FT techniques in terms of reliability and makespan improvement.

1. Introduction

Field Programmable Gate Arrays (FPGAs)-based reconfigurable computers have gained more and more importance in recent years. Modern SRAM-based FPGAs include plenty of configurable logic and routing resources to implement hardware tasks composed of millions of gates. Such FPGAs are very attractive for a wide variety of applications as they feature the ability to integrate a complex Systems-on-Chip (SoC) on a single device, combining the speedup of hardware with the flexibility of software, unlimited reconfiguration at runtime, and lower development cost than ASICs [1]. In particular, SRAM-based FPGAs are very popular in space mission systems as they are inherently flexible to meet different functional requirements [2]. In these systems, the applications are usually modeled as task graphs to manage dependencies among different tasks [3].

Compared to ground-level systems, space computing ones experience more radiations during their mission [4]. SRAM-based FPGAs suffer from susceptibility to radiations induced by soft errors [5]. The soft error rate (SER) of the environment highly affects the reliability of such FPGAs. Consequently, some Fault Tolerance (FT) techniques are required to alleviate the negative impacts of soft errors. However, FT techniques appear with different overheads [6]. Therefore, they should be applied optimally in such a way that not only the system reliability increases but also the imposed overheads affect the system performance as least as possible.

Several researchers have tried to improve the reliability of FPGA-based designs by using conventional and modern FT techniques [7]. Although significant reliability improvements have been obtained by them, they have not paid enough attention to overheads imposed by the employed FT techniques. The overheads of these techniques lead to performance degradation. Therefore, some researchers have focused on improving system reliability while a given performance is guaranteed.

* Corresponding author.
E-mail addresses: r.ramezani@eng.ui.ac.ir (R. Ramezani), ghavidel@mail.um.ac.ir (A. Ghavidel), y_sedaghat@um.ac.ir (Y. Sedaghat).

https://doi.org/10.1016/j.ress.2020.107223
Received 17 February 2020; Received in revised form 13 July 2020; Accepted 26 August 2020
Available online 29 August 2020

Notations

n	Number of tasks
m	Number of segments
τ_i	Task τ_i
SG_i	Segment i
X	Decision variables set of the whole task graph
X_i	Decision variables set of SG_i
x	Solution
$f_{o_k}(X)$	k_{th} objective function
$f_{o_k}(x)$	k_{th} objective function value
$f_{o_k}(x_{ij})$	k_{th} objective function values of the j_{th} solution of SG_i
r_i	Redundancy level of task τ_i
N_{max}	Maximum redundancy level of tasks
(x_a, x_b)	Concatenation of solutions x_a and x_b
$f_{o_k}(x_a, x_b)$	k_{th} objective function value of the solution (x_a, x_b)
R_{τ_i}	Initial reliability of task τ_i
R'_{τ_i}	Reliability of fault-tolerant task τ_i
R^*_{TG}	Reliability objective function
IR^*_{TG}	Inverse reliability objective function
IR^+_{TG}	Additive inverse reliability objective function
MS_{TG}	Makespan objective function

However, reliability and performance are two contradicting features which should be improved simultaneously. Thus, it can be seen as a multi-objective optimization problem with reliability and performance as objective functions. Solving this optimization problem leads to the generation of a *true* Pareto set of FT techniques that trades off reliability and performance.

A recent study made by the authors [8] has shown that, by applying selective FT techniques to the tasks, it is possible to improve the system reliability without degrading its performance. Their proposed technique uses an approximation approach to optimize the reliability and performance simultaneously. Although this technique obtains the solutions efficiently, it does not guarantee to generate optimal solutions. Therefore, in another study, the authors have presented an exact method that guarantees the generation of optimal solutions for the problem of reliability and makespan optimization of task graphs [9]. Although this technique can find global optimal solutions, it exhibits exponential time complexity, where the problem size exponentially increases with the number of tasks, which makes it unsuitable for runtime systems.

In order to come up with these challenges, in this paper, a technique named *segment-technique* has been presented to exactly and efficiently optimize the reliability and performance of synchronous task graphs running on SRAM-based FPGAs in harsh environments. The proposed technique generates an optimal solution set of FT techniques for a synchronous task graph in such a way that its solutions are non-dominated and lead to trade off between reliability and makespan. The synchronous task graphs consist of multiple serially connected subgraphs known as segments. The presented approach obtains the *true* Pareto set of FT techniques of the whole task graph by exactly optimizing the reliability and makespan of different segments of the task graph separately, combining the non-dominated solutions of the segments in order to generate a preliminary set of solutions, and finally filtering out the dominated solutions. In this regard, it has been proved that such a partitioning strategy leads to the generation of the *true* Pareto set of the whole task graph with low time complexity.

The efficiency of the proposed technique has been evaluated by conducting several experiments on real-world and randomly-generated task graphs. In this regard, the positive impacts of applying optimal FT techniques on improving the reliability of task graphs without affecting their makespan has been first demonstrated. Then, it has been shown that the proposed technique outperforms the time complexity of the recent *enumeration-based Pareto technique* by examining medium-size task graphs. The third experiment compares the proposed solution with three well-known evolutionary algorithms in terms of precision, recall, and time complexity. The obtained results show that the proposed technique requires much less computation time, yet it yields much better precision and recall. The last experiment demonstrates the positive impacts of the proposed technique in reliability and makespan improvement of task graphs running in harsh environments with dynamic and unpredicted SREs.

The rest of the paper is organized as follows. Section 2 introduces related studies and Section 3 presents illustrative examples and the system model. Section 4 elaborates basic concepts and contains the general methodology and foundations of the presented technique. In Section 5, it is proved that the proposed technique can generate the *true* Pareto set of the whole task graph. The experimental evaluations and the obtained results are discussed in Section 6. Finally, Section 7 concludes the paper.

2. Related work

Reliability assessment is one of the most important testing goals during the system validation and verification process [10]. In this regard, designers should assess the reliability of their designs and improve it if it does not comply with system requirements [11]. This issue is especially important in safety-critical applications that require high levels of reliability [12].

Since SRAM-based FPGAs are vulnerable to soft errors, the problem of improving the reliability of FPGA-based designs has been addressed by many researchers. However, most of them have not paid enough attention to the performance degradation problem. Preliminary FT techniques were applied within designs by developers themselves. The efficiency of these techniques should be assessed during the validation and verification phase. RcB and NSCP are examples of these techniques which require acceptance tests to validate the outputs [13]. These techniques are not widely used as they are application-dependent and are very hard to apply [14]. Therefore, modern FT techniques are used which are applied to the designs that have been developed completely [15]. These techniques consist of two main FT strategies: design-based methods and recovery-based methods.

Design-based methods are based on replication and are applied to different abstractions regardless of the final application [16]. For example, the application of dual and triple redundancies in increasing the reliability of FPGA-based designs has been addressed in [17] and [18]. Similarly, the *Adaptive FT technique* proposed by [19] is a dynamic redundancy-based FT technique that employs different task replications for different orbits during space missions. Although these FT techniques impose different overheads on the system, their overheads can be reduced by protecting more critical parts of the design [20].

Recovery-based methods are the second group of mitigation techniques that avoid soft error accumulation in configuration memory and can be used by SRAM-based reconfigurable computers at runtime [21]. In this approach, the configuration memory of designs is sporadically refreshed to recover faulty cells using scrubbing or task reallocation tasks [22]. The efficiency of the scrubbing techniques can be improved by customizing the recovery process for essential configuration bits [23].

Modern SRAM-based reconfigurable computers are widely used in safety-critical real-time systems. In these systems, performance is as important as reliability. Although conventional FT techniques improve the system reliability, they impose unaffordable overheads on runtime systems which leads to the performance degradation. Very few studies have paid attention to performance constraints when applying FT techniques. These studies try to increase the system's reliability while a

given performance is guaranteed as well. For instance, the Primary/Backup scheme presented in [24] supports FT while reducing the performance degradation. The real-time FT scheduling algorithm presented in [25] uses a sufficient schedulability test to guarantee the schedulability of hybrid hardware/software tasks while tolerating f_i faults during their execution. Similarly, the authors have examined the impacts of different FT strategies for different real-time scheduling algorithms on the performance of reconfigurable computers [26]. In another study, a reconfigurable FT framework has been presented by [27] for dynamic environments which determines the best FT techniques of the tasks by taking both task execution time and system fault rate into account. Another approach in this context is the co-employment of CPUs-FPGAs to make tradeoffs between the reliability of CPUs and the performance of FPGAs [28]. All these studies try to increase the system reliability while degrading its performance as least as possible.

Although the mentioned studies try to not degrade the system performance when applying FT techniques, they do not guarantee to optimize it. Performance is a crucial factor for modern computer systems as future applications demand high-performance computing (HPC). In this regard, as the granularity of FT techniques affects the system reliability and performance, the problem of selecting the optimal granularity level of FT techniques has been addressed in [29]. Fine-grain granularities yield better reliability at the cost of performance losses. In a previous study [8], the authors have presented an approximation technique to address the problem of reliability and performance optimization of general task graphs executing on SRAM-based reconfigurable computers. This technique generates a Pareto set of FT techniques for the task graphs in which each solution of the generated set determines the FT technique of each hardware task of the task graph. Thus, due to the competition of the objective functions, this set features different reliability and makespan trade-offs. Thus, a solution that best meets system requirements can be applied to the running tasks. Since this study has used an approximation strategy, it does not guarantee to generate global optima of reliability and makespan. Therefore, in another study, a technique named *enumeration-based Pareto technique* has been presented to exactly optimizing the reliability and makespan of general task graphs [9]. This technique generates the *true* Pareto set of FT techniques by using an exact optimization method. Although the *enumeration-based Pareto technique* guarantees the generation of the *true* Pareto set, it suffers from the exponential time complexity since the size of the exploration space increases drastically with the number of tasks. Therefore, it cannot be used at modern computer systems whose characteristics and requirements change dynamically at runtime, especially the space-computing ones that experience different SERs during their mission.

Therefore, a new approach is required to exactly and efficiently optimize the reliability and makespan simultaneously in such a way that it can be used at runtime systems. In this paper, a new technique is presented for the problem of reliability and makespan optimization of synchronous task graphs running on SRAM-based FPGAs in harsh environments. It guarantees the generation of optimal solutions with low time complexity.

3. System model and illustrative examples

In this section, the basic concepts are explained first. Then, the model of tasks and task graphs has been presented, followed by discussing the reliability model used for calculating the system reliability. Finally, an illustrative example has been provided to better understand the basic ideas of the presented reliability and makespan optimization problem.

3.1. Preliminary concepts

A technical story is presented in this subsection to make the concepts and ideas of this study clearer.

CPUs are general-purpose processors that provide flexibility at the cost of performance losses. ASICs are contradicting processing devices which lack flexibility but provide acceptable levels of performance. SRAM-based FPGAs are modern computing devices that trade off CPUs and ASICs. These FPGAs provide both flexibility and performance which make them appropriate solutions for modern computer systems.

In SRAM-based FPGAs, the designs (known as hardware tasks) are configured on the FPGA by programming bitstreams of the configuration memory. Since SRAM memories can be modified at runtime, the hardware tasks can be reconfigured at runtime to change the system functionality. As the FPGAs have limited resources, scheduling strategies are required to steer the reconfiguration and execution of hardware tasks on the FPGA. This issue is especially important for task graphs, as the underlying scheduler not only should manage the device resources but also it should meet the precedence constraints of the tasks. The employed scheduler highly affects the execution length of the task graph (known as makespan). Makespan is a representative of the system performance.

Despite being programmable, SRAMs are vulnerable to soft errors induced by the environment. Thus, reliability is the main concern of SRAM-based reconfigurable computers which is usually compensated for by using FT techniques (e.g., by duplicating or triplicating hardware tasks). Although FT techniques improve the system reliability, they impose different overheads on the system and cause an increase to its makespan (which leads to degrade the system performance).

As reliability and makespan are contradicting features, the FT techniques should be applied in such a way that the system reliability is improved, but their imposed overheads do not affect the makespan significantly. Therefore, it can be seen as a multi-objective optimization problem with reliability maximization and makespan minimization as objective functions, and the FT technique of individual hardware tasks as decision variables. As the objective functions are contradicting, instead of just one solution, solving the corresponding optimization problem leads to the generation of multiple solutions, known as Pareto set. Each solution of this set determines the FT technique of individual hardware tasks of the task graph. If the optimization problem is solved by an exact method (which guarantees to find global optimal solutions) the obtained solution set is referred to as *true* Pareto set. In a Pareto set, all solutions are non-dominated. In other words, no solution is completely better than the other ones.

For example, assume the reliability and makespan of a task graph without applying any FT techniques is [0.9868524, 2593]. After optimizing the reliability and makespan of this task graph, a Pareto set with three solutions [0.9888453, 2605], [0.9923657, 2647], and [0.9975587, 2689] are obtained. All these solutions are non-dominated, and the system designer can select the best one based on system requirements.

Obtaining the *true* Pareto set with exact methods is an NP-Hard problem, and it takes exponential computation time with the number of tasks. Therefore, many researchers have used inexact optimization techniques (such as approximation ones and evolutionary ones) to solve the optimization problem in lower time at the cost of losing global optima. In this study, an exact technique is presented which guarantees to obtain the *true* Pareto set of FT techniques of synchronous task graph efficiently with little computation times. The proposed technique is based on the task graph decomposition idea whose ability in obtaining global optima is proved in this study. The efficiency of the proposed technique is also demonstrated by conduction several experiments over state-of-the-art exact and inexact optimization methods.

3.2. Task and task graph model

The synchronous task graph is an intra-task parallelism task model in which the initial task graph is forked into several independent segments (a.k.a. subgraphs, levels), each of which contains one or more tasks that can execute simultaneously in parallel. Upon completion of a

segment, its parallel tasks join into the next segment and this behavior may repeat again up until the completion of the task graph.

We consider a synchronous task graph $TG = \{\Gamma, S, \delta\}$ where Γ is the set of n hardware tasks as $\Gamma = \{\tau_1, ..., \tau_n\}$. Each hardware task τ_i is attributed with $\tau_i = \{CT_i, CB_i, TS_i, R_i\}$ where CT_i is task computation time which includes task execution time plus its configuration delay, CB_i is the number of configurable logic blocks (CLB) required by the task, TS_i is task size in the configuration memory, and R_i is the task reliability which can be obtained using different reliability models [30-32]. S is the set of m segments in TG as $S = \{SG_1, ..., SG_m\}$, where each segment SG_i contains one or more hardware tasks as $SG_i \subseteq \Gamma$ and $\cap_{SG_i \in S} SG_i = \varnothing$. Finally, δ denotes the set of dependencies among tasks in which δ_{ij} indicates the dependency between hardware tasks τ_i and τ_j and means that τ_j cannot start its execution until the execution of τ_i finishes. Thus, if $\delta_{ij} \in \delta$, τ_i and τ_j cannot be placed at the same segment.

In this model, the tasks of segment SG_i cannot start their execution until the execution of all tasks of segment SG_{i-1} is finished [33]. However, if there are enough resources, all tasks of a segment can execute in parallel. Fig. 1 illustrates a sample synchronous task graph with 5 segments and 11 hardware tasks.

3.3. Reliability model

Since the SRAM-based FPGAs are susceptible to radiation-induced upsets [34], the soft errors are the object of concern. Each hardware task τ_i is attributed to the reliability R_i. With the assumption that the task graph TG execute without any failure if all its tasks execute correctly, the reliability of TG is obtained as:

$$R_{TG} = \prod_{\tau_i \in \Gamma} R_i \tag{1}$$

Once the reliability of task graph TG is obtained, the Mean Time to Failure (MTTF) of TG is calculated as [35]:

$$MTTF_{TG} = \frac{MS_{TG}}{1 - R_{TG}} \tag{2}$$

where MS_{TG} is the makespan of TG and equals to the total execution time of the task graph. In this paper, the configurations and executions of task graphs are managed using an *As Soon As Possible* (ASAP) scheduling strategy [9]. This strategy has been selected as it is simple and yields suboptimal schedules with very little runtime overheads. However, other scheduling strategies can be used instead as the proposed technique is orthogonal to all of them.

As it will be discussed soon, in this paper, the reliability R_i of hardware task τ_i is increased by applying replication-based FT techniques [29]. Thus, following Eq. (1), improving the reliability of hardware tasks leads to an increase in the reliability of the whole task graph. Nevertheless, such FT techniques have time (in terms of execution time and configuration time) and space (in terms of configuration memory) overheads. Thus, making a hardware task τ_i of segment s_j redundant might lead to increase the makespan of s_j and as a result, increase the makespan of the whole task graph. Thus, the FT technique of a task should be selected optimally in such a way that not only the reliability of the task is improved but also its imposed overheads are negligible. In this paper, an efficient yet exact technique has been presented to generate such optimal FT techniques for synchronous task graphs.

3.4. Illustrative example

In this section, an illustrative example has been provided to better clarify the ideas presented in this study. For this purpose, task τ_1 of SG_1 and tasks τ_2, τ_3, and τ_4 of SG_2 of Fig. 1 have been attributed with different characteristics as presented in Table 1. In this case, the computation time and size of the tasks as well as their resource occupancy on the target FPGA are shown.

A simple schedule of the tasks of Table 1 is shown in Fig. 2. In this case, the *ASAP* scheduling strategy without applying any FT techniques has been used to execute the tasks on a given FPGA which comprises 14 CLBs. As the schedule of Fig. 2 shows, the execution of SG_1 finishes at time 17. Then, the execution of SG_2 starts. However, the tasks of SG_2 cannot be executed concurrently as their cumulative size exceeds available resources. Therefore, in this example, the execution of SG_2 finishes at time 71. For simplicity, the execution of other segments of the task graph is not shown.

Next, the reliability of the task graph is increased by applying some replication-based FT techniques to tasks τ_1 and τ_2. For this purpose, the *Duplication With Comparison* (DWC) technique has been applied to task τ_1, and the *Triple Modular Redundancy* (TMR) technique has been applied to task τ_2 whose obtained schedule is presented in Fig. 3. As this figure shows, although applying FT techniques has led to improve the task graph reliability, its makespan has not been prolonged compared to the simple schedule of Fig. 2. In this case, the schedule of Fig. 3 dominates that of Fig. 2 as it yields better reliability without increasing the makespan. In this example, for simplicity, it has been assumed that the computation time of the decider is negligible with respect to the computation time of the tasks.

In another schedule, the DWC FT technique has been applied only to task τ_3 whose scheduling result is shown in Fig. 4. As this figure shows, since there are not enough resources, task τ_3 has been replicated temporally. Therefore, compared to the schedule of Fig. 2, this schedule yields better reliability but at the cost of prolonging the makespan. In other words, these two schedules are non-dominated, and none of them is better than the other one. However, the schedule of Fig. 3 dominates that of Fig. 4 as it yields better reliability and makespan.

The goal of this paper is to exactly and efficiently generate the non-dominated solution set of FT techniques for a synchronous task graph each of which leads to specific reliability and makespan in such a way that none of the corresponding schedules is dominated by other schedules.

4. Problem formulation

4.1. Problem definition

Since FT techniques impose different overheads on the system, the reliability of a system is usually improved at the cost of performance degradation. Thus, there exist some sort of contradictions between reliability and makespan objective functions. Therefore, the optimization methods that address the problem of reliability and makespan optimization of a task graph instead of generating just one FT solution, generate a solution set of FT techniques for the whole task graph [9]. Each solution contains multiple FT techniques, each of which corresponds to a hardware task in the task graph. Therefore, different solution sets yield different reliabilities and makespans for the task graph. Thus, it can be seen as a multi-objective optimization problem in which reliability and makespan of the task graph are the objective functions, and the FT techniques of the tasks (in terms of redundancy level) are

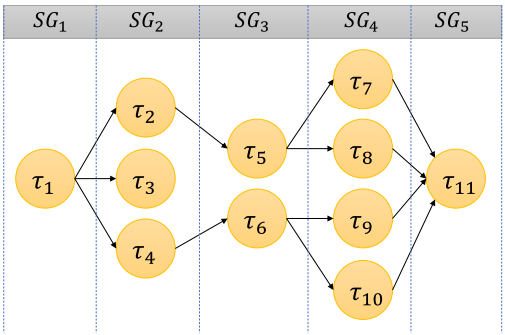

Fig. 1. A sample synchronous task graph.

Table 1
The characteristics of tasks τ_1, τ_2, τ_3, and τ_4 of Fig. 1.

Task	Computation Time	Task Size (CLB)	Resource Usage (%)
τ_1	17	5	35.71%
τ_2	19	3	21.43%
τ_3	20	12	85.71%
τ_4	15	13	92.86%

decision variables.

Without losing generality, assuming n decision variables and d objective functions, let us define the following terms from the optimization problem point of view:

- $X = \{r_1, r_2, ..., r_n\}$ is the set of decision variables. Any set of decision variables whose variables have been assigned a value is referred to as a solution x.
- $f_o(X) = \{f_{o_1}(X), f_{o_2}(X), ..., f_{o_d}(X)\}$ is a set of objective functions. These functions return a set of specific values for each solution x. Given solution x, each one of these values is denoted as $f_{o_k}(x)$, $1 \le k \le d$.

In this paper, each decision variable r_i indicates the redundancy level of hardware task τ_i. In this regard, with n hardware tasks in the task graph, the set of decision variables is represented by $X = \{r_1, r_2, ..., r_n\}$, in which r_i denotes the redundancy level of Task τ_i, $1 \le r_i \le N_{max}$. For example, $r_i = 1$ indicates no FT technique for τ_i, $r_i = 2$ stands for the DWC technique, and $r_i = 3$ denotes the TMR technique. The value of N_{max} is the maximum redundancy level of the tasks. Besides, in this paper, the reliability maximization and makespan minimization are the objective functions of the presented optimization problem.

There are multiple Pareto optimality definitions in multi-objective optimization problems in terms of minimization, as [36]:

Definition 1. (Pareto dominance): A solution x dominates y (denoted as $x \succ y$) *iff* $f_{o_t}(x) \le f_{o_t}(y)$ and $\exists\, q$: $f_{o_q}(x) < f_{o_q}(y)$; $t, q \in \{1, 2, ..., d\}$; otherwise, x and y are not better than each other. If there are no solutions that dominate x, then x is a *non-dominated solution.*

Definition 2. (Pareto set): A set of all non-dominated solutions $\{x | \exists y: y \succ x\}$ is called a Pareto set.

Definition 3. (True Pareto set): If all possible non-dominated solutions are generated using an exhaustive optimization method, the obtained solution set is called *true Pareto set.*

The presented optimization problem is a nonlinear integer one as the decision variables (redundancy level of the tasks) take integer values. It has been proved that optimization problems with integer variables cannot be optimally solved in polynomial time [37]. Thus, in this paper, a partitioning-based technique has been presented to overcome this complexity for synchronous task graphs. With this strategy, by analyzing the task graph by partitioning it into a number of segments, the computation will take a lower time with respect to the analysis of the whole task graph. The main goal of this study is to obtain the *true* Pareto set of FT techniques of the whole task graph at runtime with low time complexity.

4.2. Outline of segment-technique

The presented technique generates the *true* Pareto set of FT techniques in three steps:

1) Task Graph Segmentation: First of all, the synchronous task graph (TG) is partitioned into m disjoint segments SG_i in such a way that $|SG_i| \ge 1$ and $\sum_{i=1}^{m} |SG_i| = n$, where $|SG_i|$ is the number of tasks of SG_i. In this partitioning $\{SG_1, SG_2, ..., SG_m\}$ have disjoint tasks. In other words, each segment SG_i will have a set of disjoint decision variables X_i where $X_i \subseteq X$ and $\cap_{i \in m} X_i = \emptyset$, X being that of the whole task graph. Different strategies, such as level-based scheduling, can be used to partition the task graph into multiple segments [38-41].
2) True Pareto set generation of the segments: Then, an exact multi-objective optimization method is used to generate the true Pareto set of FT techniques of the segments separately. In this paper, the enumeration-based Pareto technique presented in [9] has been used to generate the true Pareto set of each segment separately in which reliability and makespan are objective functions. This technique is based on the definitions given at the beginning of this section. Nevertheless, any other exact multi-objective optimization method presented in the literature can be used instead for the generation of the true Pareto set, as they are completely orthogonal to the solution that this paper presents.
3) Combination of solutions: Finally, the obtained non-dominated solutions of the segments are combined sequentially to produce the preliminary solution set of FT techniques. The Cartesian product operation has been used for this purpose. The Cartesian product operates on two sets A and B and yields a set of ordered pairs (x_a, x_b) in which $x_a \in A$ and $x_b \in B$. In the presented problem, each combined solution (x_a, x_b) is obtained as X_A **CAT** X_B, where **CAT** is the concatenation operator and concatenates the decision variables of the two solutions x_a and x_b to generate a new solution. In addition, since the objective functions return a set of specific values for each solution, the objective function values of the combined solutions are calculated as well and are denoted by $f_{o_k}(x_a, x_b)$. In other words, in this paper, the Cartesian product of two solution sets A and B results in a new solution set as:

$$\{(x_a, x_b), f_{o_k}(x_a, x_b) | x_a \in A, x_b \in B, \ k = \{1, 2\}\} \quad (3)$$

where $f_{o_k}(x_a, x_b)$ is the k_{th} objective function value of the combined solution (x_a, x_b). For example, assume two solution sets A and B, each of which contains three solutions where A has two decision variables and B has one decision variable. The Cartesian product of these two solution sets leads to the generation of a solution set with 9 solutions as shown in Table 2. As this table shows, in each combination, the values of the

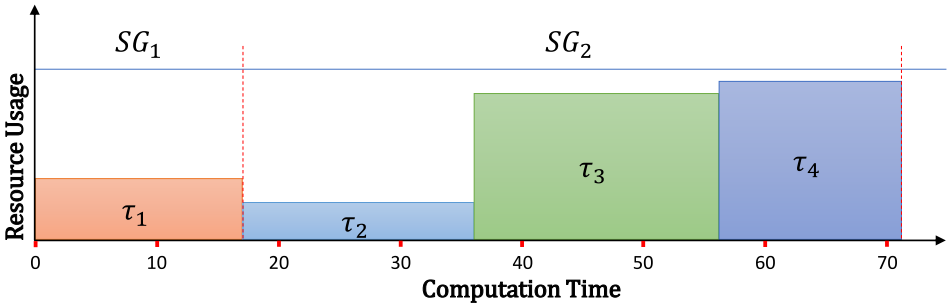

Fig. 2. A simple schedule of the tasks of Table 1 with no FT techniques.

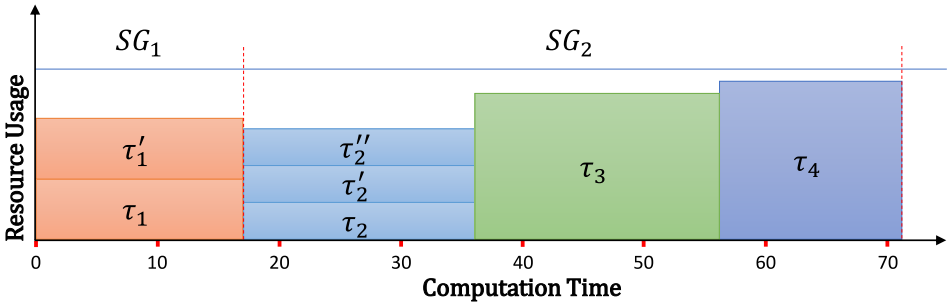

Fig. 3. A schedule of the tasks of Table 1 with applying DWC technique to task τ_1 and TMR technique to task τ_2.

decision variables of the solutions are concatenated. In addition, each combined solution leads to a specific objective function value that should be calculated.

Finally, in the next section, it will be shown that the Cartesian product of the solutions would generate some dominated solutions. Hence, at this step, a filtering function is applied to remove the dominated solutions after each combination. Due to simplicity, this filtering is not shown in Table 2. After generating the solution set of FT techniques, a solution that best meets system requirements is selected to be applied to the running tasks by the underlying system middleware.

As it will be proved in the next sections, such a partitioning strategy yields the same results as an exact optimization method is applied to the whole task graph without partitioning it. Last but not least, it should be noted that Step 1 (task graph segmentation) is done at design time. Nevertheless, as the *true* Pareto set of FT techniques depends on the reliability of tasks, both Step 2 and Step 3 are performed at runtime, once the reliability of the tasks changes due to changes in the SER of the environment.

The workflow of the proposed technique is shown in the flowchart of Fig. 5. As this figure shows, a task graph application is developed first at runtime followed by the task graph segmentation and the task graph validation and verification. Next, at runtime, it is checked if the task graph reliability and makespan meets system requirements. If so, the task graph continues its normal execution. Otherwise, the proposed technique is applied to select a set of FT techniques that best meets system requirements for the running task graph. The modified task graph is then rescheduled to be executed by the FPGA. Since the FPGA might be operated in different orbital conditions, the SER of the environment might be changed which results in changing the task graph reliability. Thus, the compliance of the task graphs is examined repeatedly.

4.3. Additive reliability objective function

In this study, the reliability (R_{TG}) and makespan (MS_{TG}) of task graph TG are the objective functions of the optimization problem. Since in the proposed solution each segment has its own decision variables, the objective functions of the whole task graph should be obtained from those of the different segments. Since Step 3 is performed at runtime, rescheduling the whole task graph to obtain its reliability and makespan leads to an increase in the time complexity of the proposed solution. In order to cope with this complexity, a new strategy has been proposed which is based on additive objective functions. With this strategy, the values of the objective functions of the task graph are obtained from summing the objective function values of the different segments. The decomposability of reliability and makespan objective functions are discussed in the current and the next subsections. In this subsection, an additive reliability objective function is derived by which the reliability of the task graph can be obtained from the reliability of the segments, regardless of the redundancy level of the tasks.

In this work, it has been assumed that the initial reliability of a hardware task τ_i (denoted as R_i) depends on the characteristics of the task and the SER of the environment, and it is obtained from the reliability model presented in [42]. It is noteworthy to state that any other reliability estimation methods can be used instead to estimate R_i [31,32], since they are orthogonal to the solution presented in this paper. In this work, the reliability of tasks is increased by applying replication-based FT techniques. As mentioned before, each decision variable r_i indicates the redundancy level of task τ_i. Thus, by replicating task τ_i for r_i times, using the $1-out-of-r_i$ FT scheme, the reliability of task τ_i after applying FT techniques (denoted as R_i') is given by [43]:

$$R_i' = \sum_{k=1}^{r_i} \binom{r_i}{k} (R_i)^k (1-R_i)^{r_i-k} \tag{4}$$

Without losing generality, suppose each hardware task is a segment. Therefore, assuming the task graph is executed successfully if all the tasks are executed without any failure, the reliability of the task graph TG after applying FT techniques is obtained from the product of reliability of its segments, regardless of the redundancy level of the tasks [44], as:

$$R_{TG}^* = \prod_{\tau_i \in TG} R_i' = \prod_{\tau_i \in TG} \left(\sum_{k=1}^{r_i} \binom{r_i}{k} {R_i}^k (1-R_i)^{r_i-k} \right) \tag{5}$$

On the one hand, the objective functions should be additive in the proposed technique. On the other hand, since the definitions of optimality presented in the previous section are in terms of minimization, the objective functions should be defined in terms of minimization as well. It is clear that the reliability objective function of Eq. (5) is not additive and also it is used in the reliability maximization optimization

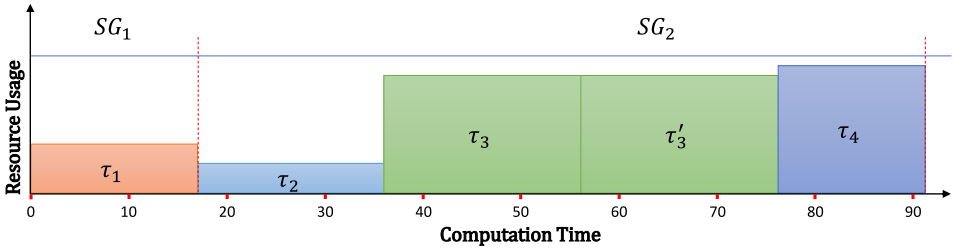

Fig. 4. A schedule of the tasks of Table 1 with applying DWC technique to task τ_3.

Table 2
An example of the Cartesian product of two solution sets.

		Solution Set B		
		$(SM), f_{o_k}(B_1)$	$(DWC), f_{o_k}(B_2)$	$(TMR), f_{o_k}(B_3)$
Solution set A	$(SM, TMR), f_{o_k}(A_1)$	$(SM, TMR, SM), f_{o_k}(A_1, B_1)$	$(SM, TMR, DWC), f_{o_k}(A_1, B_2)$	$(SM, TMR, TMR), f_{o_k}(A_1, B_3)$
	$(DWC, DWC), f_{o_k}(A_2)$	$(DWC, DWC, SM), f_{o_k}(A_2, B_1)$	$(DWC, DWC, DWC), f_{o_k}(A_2, B_2)$	$(DWC, DWC, TMR), f_{o_k}(A_2, B_3)$
	$(TMR, TMR), f_{o_k}(A_3)$	$(TMR, TMR, SM), f_{o_k}(A_3, B_1)$	$(TMR, TMR, DWC), f_{o_k}(A_3, B_2)$	$(TMR, TMR, TMR), f_{o_k}(A_3, B_3)$

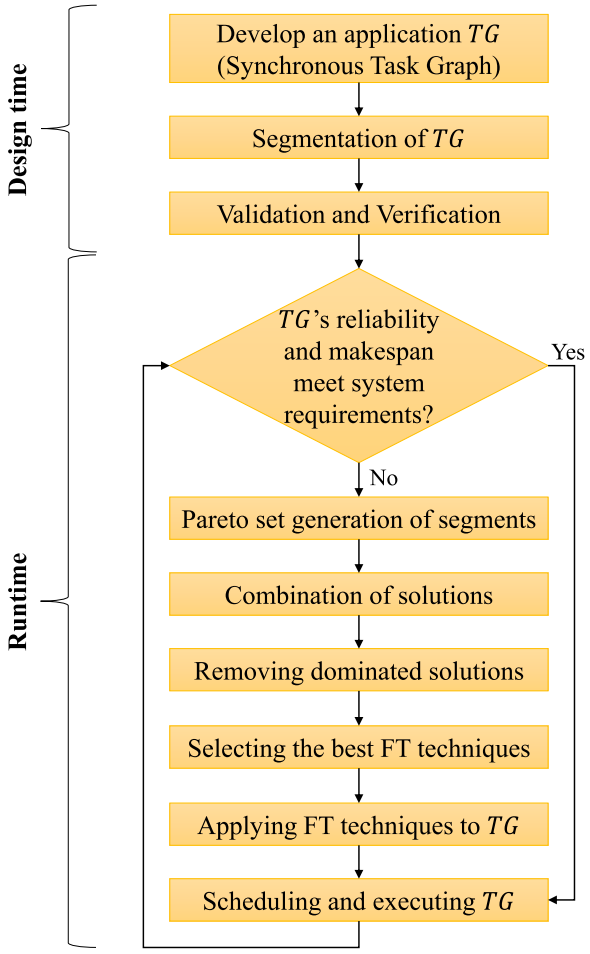

Fig. 5. Flowchart of the proposed technique workflow.

problems. The inverse of reliability is a good alternative objective function that can be used in minimization optimization problems. Thus, an inverse reliability objective function (denoted by IR^{+}_{TG}) has been presented in Proposition 1 which is additive and its minimization generates the same optimization solutions that the maximization of the original reliability objective function of Eq. (5) generates.

Proposition 1. *The inverse reliability objective function IR^{+}_{TG} is additive and its minimization generates the same optimization solutions as the maximization of the reliability objective function R^{*}_{TG}.*

Proof. : Since R^{*}_{TG} is in terms of maximization, $1/R^{*}_{TG}$ can be used in minimization optimization problems. In other words, the minimization of $1/R^{*}_{TG}$ is the same as the maximization of R^{*}_{TG}. Thus, let define the inverse reliability of task graph TG as: $IR^{*}_{TG} = 1/R^{*}_{TG}$. Therefore, we have:

$$IR^{*}_{TG} = 1/R^{*}_{TG} = 1/\prod_{\tau_i \in TG} R'_i \tag{6}$$

Assume IR^{+}_{TG} is an additive inverse reliability objective function that is monotonically equivalent to IR^{*}_{TG}. Monotonic equivalency means that if $IR^{*}_{TG1} < IR^{*}_{TG2}$ then $IR^{+}_{TG1} < IR^{+}_{TG2}$. Therefore, a set of decision variables minimizing IR^{*}_{TG} leads to the minimization of IR^{+}_{TG} as well. In order to transform the inverse reliability objective function (IR^{*}_{TG}) to a monotonically equivalent additive one (IR^{+}_{TG}), a logarithmic transformation has been applied to the objective function of Eq. (6), as:

$$IR^{+}_{TG} = \log IR^{*}_{TG} = \log\left(1/\prod_{\tau_i \in TG} R'_i\right) = \log 1 - \log \prod_{\tau_i \in TG} R'_i = -\sum_{\tau_i \in TG} \log R'_i \tag{7}$$

By this transformation, IR^{+}_{TG} is obtained from the negative of summation of the logarithm of the segments' reliability. IR^{+}_{TG} uses a logarithmic transformation which is a monotonic transformation and conserves the rank order of the solutions. Thus, the minimization of IR^{+}_{TG} generates the same optimization solutions as the maximization of R^{*}_{TG}. Since IR^{+}_{TG} is additive and it has been presented in terms of minimization, it can be easily used by the presented approach. Finally, by using the assumption used in Eq. (5), the inverse reliability of a segment ($IR^{+}_{SG_j}$) can be obtained as $IR^{+}_{SG_j} = \sum_{\tau_i \in SG_j} - \log R'_i$. Thus, regardless of the redundancy level of the tasks, we have $IR^{+}_{TG} = \sum_{i=1}^{m} IR^{+}_{SG_i}$, where m is the number of segments. □

4.4. Additive makespan objective function

Makespan is another objective function of the presented problem. Proposition 2 establishes that the makespan of the whole task graph can be always obtained as the summation of the segments' makespan, regardless of the redundancy level of the tasks.

Proposition 2. *In the case of synchronous task graphs, $MS_{TG} = \sum_{i=1}^{m} MS_{SG_i}$, regardless of the redundancy level of the tasks.*

Proof. : The proof is easy and straightforward. Let MS_{SG_i} denote the makespan of SG_i. In synchronous task graphs, the hardware tasks of a segment should be joined before starting the next segment. In other words, the segments are executed serially and a segment does not start its execution until the execution of all tasks of the previous segment and their redundancies finishes. Thus, since the segments cannot execute in parallel, we will have $MS_{TG} = \sum_{i=1}^{m} MS_{SG_i}$. □

As it will be shown, both the additive reliability and makespan objective functions are used by the proposed *segment-technique* to generate the *true* Pareto set of FT techniques. Next, some propositions have been presented to address this question: "*Is it possible to generate the true Pareto set of the whole task graph by partitioning the task graph into multiple segments, optimizing each segment separately, and then combining and filtering out the obtained solutions?*"

5. True Pareto set generation for synchronous task graphs

As it was shown, there are two additive objective functions to be minimized in the presented problem: the inverse reliability of the task

graph (IR^{+}_{TG}) and makespan of the task graph (MS_{TG}). For simplicity, in the rest of the paper, $f_{o_1}(X)$ denotes one of the objective functions and $f_{o_2}(X)$ denotes the other one, where X is a decision variable set of TG. In other words, the general optimization problem is $\{Min\ f_{o_1}(X),\ Min\ f_{o_2}(X)\}$.

When the objective functions are denoted as additive functions, there exist several properties of the non-dominated solutions obtained from the presented *segment-technique*. Propositions 3–5 will discuss these properties in detail. In these propositions, it is assumed that $f_{o_k}(x) = \sum_{i=1}^{m} f_{o_k}(x_i),\ k = \{1,\ 2\}$, where $x_i \in X_i$ and $X = \cup_{i \in m} X_i$.

The first property relates to the *true* Pareto set coverage. In this regard, Proposition 3 proves that the proposed technique guarantees the generation of the *true* Pareto set.

Proposition 3. *Any non-dominated solution of the true Pareto set of the whole task graph can be obtained from the Cartesian combination of non-dominated solutions of the true Pareto set of the segments, given objective functions* $f_{o_k}(x) = \sum_{i=1}^{m} f_{o_k}(x_i),\ k = \{1,\ 2\}$.

Proof. : As mentioned before, in the presented technique, the *true* Pareto set of the segments is generated separately using an exact optimization method. Thus, the optimization problem of the segment SG_i is formulated as $\{Min\ f_{o_1}(X_i),\ Min\ f_{o_2}(X_i) \mid X_i \subset X\}$. Suppose NDS is the set of non-dominated solutions of the *true* Pareto set of the original task graph and NDS_{SG_i} is the set of non-dominated solutions of the *true* Pareto set of segment SG_i. In addition, suppose NDS_C is the non-dominated solutions set generated from applying the Cartesian operator to non-dominated solutions of the segments. In other words

$$NDS_C = NDS_{SG_1} \times \ \ldots \times NDS_{SG_m} \tag{8}$$

where $\times$ is the Cartesian product operator. Now, assuming there exists a solution $x' \in NDS$ but $x' \notin NDS_C$, the proof is by contradiction. Since $x' \in NDS$, then $f_{o_1}(x') < f_{o_1}(x)$ and $f_{o_2}(x') \leq f_{o_2}(x)$, $\forall x \notin NDS$. In addition, without losing generality, suppose there exist two segments, e.g., $SG = \{SG_1,\ SG_2\}$. In this case, $x' = (x_1',\ x_2')$. Since the objective functions are additive, we have $f_{o_1}(x_1') + f_{o_1}(x_2') < f_{o_1}(x_1) + f_{o_1}(x_2)$. There exist four different scenarios to satisfy this expression:

1) $f_{o_1}(x_1') < f_{o_1}(x_1)$ and $f_{o_1}(x_2') \leq f_{o_1}(x_2)$
2) $f_{o_1}(x_1') \leq f_{o_1}(x_1)$ and $f_{o_1}(x_2') < f_{o_1}(x_2)$
3) $f_{o_1}(x_1') \geq f_{o_1}(x_1)$ and $f_{o_1}(x_2') < f_{o_1}(x_2)$
4) $f_{o_1}(x_1') < f_{o_1}(x_1)$ and $f_{o_1}(x_2') \geq f_{o_1}(x_2)$

In order to prove the proposition, it has to be shown that there cannot exist an $x' \in NDS$ but $x' \notin NDS_C$. The first two scenarios indicate that NDS_{SG_1} and NDS_{SG_2} include x_1' and x_2' or better solutions with respect to the objective function f_{o_1}. The former circumstance is a contradiction to the assumption that $x' \notin NDS_C$ because, as indicated by Eq. (8), NDS_C is obtained from the Cartesian combination of non-dominated solutions of the segments. The latter circumstance is a contradiction to the assumption that $x' \in NDS$ because a better solution implies that x' has been dominated. In the last two scenarios, there is a solution which dominates x', i.e., $(x_1,\ x_2')$ and (x_1', x_2) are both better than x', which is a contradiction to the original assumption that $x' \in NDS$. These four scenarios can also be followed for $f_{o_2}(x') < f_{o_2}(x)$ and the last two ones for $f_{o_2}(x') = f_{o_2}(x)$ to complete the proof. □

Although Proposition 3 guarantees the generation of the *true* Pareto set, some of the solutions achieved by the Cartesian combination might be dominated, as shown by Proposition 4.

Proposition 4. *Some solutions obtained from the Cartesian combination of non-dominated solutions of the segments* (NDS_C) *might be dominated.*

Proof. : The proposition can be proved by contradiction. Let $(f_{o_1}(x_{ij}), f_{o_2}(x_{ij}))$ denote the objective function values of the jth non-dominated solution of segment SG_i. Assume there exist two segments, each of which contains two non-dominated solutions, with the objective function values $(f_{o_1}(x_{11}), f_{o_2}(x_{11}))$ and $(f_{o_1}(x_{12}), f_{o_2}(x_{12}))$ for the first segment, and the objective function values $(f_{o_1}(x_{21}), f_{o_2}(x_{21}))$ and $(f_{o_1}(x_{22}), f_{o_2}(x_{22}))$ for the second one. Since each segment has two non-dominated solutions, the solutions of the first segment should meet $f_{o_1}(x_{11}) < f_{o_1}(x_{12})$ and $f_{o_2}(x_{11}) > f_{o_2}(x_{12})$, and the solutions of the second segment should meet $f_{o_1}(x_{21}) > f_{o_1}(x_{22})$ and $f_{o_2}(x_{21}) < f_{o_2}(x_{22})$. Four different objective function values can be achieved from the Cartesian combination of these non-dominated solutions as:

1) $(f_{o_1}(x_{11}, x_{21}), f_{o_2}(x_{11}, x_{21})) = (f_{o_1}(x_{11}) + f_{o_1}(x_{21}), f_{o_2}(x_{11}) + f_{o_2}(x_{21}))$
2) $(f_{o_1}(x_{11}, x_{22}), f_{o_2}(x_{11}, x_{22})) = (f_{o_1}(x_{11}) + f_{o_1}(x_{22}), f_{o_2}(x_{11}) + f_{o_2}(x_{22}))$
3) $(f_{o_1}(x_{12}, x_{21}), f_{o_2}(x_{12}, x_{21})) = (f_{o_1}(x_{12}) + f_{o_1}(x_{21}), f_{o_2}(x_{12}) + f_{o_2}(x_{21}))$
4) $(f_{o_1}(x_{12}, x_{22}), f_{o_2}(x_{12}, x_{22})) = (f_{o_1}(x_{12}) + f_{o_1}(x_{22}), f_{o_2}(x_{12}) + f_{o_2}(x_{22}))$

Now, assume that all these combined solutions are non-dominated. It means that in the comparison of these combined solutions, for example, if $f_{o_1}(x) < f_{o_1}(x')$, then $f_{o_2}(x) > f_{o_2}(x')$ and vice versa. Thus, let us violate the assumption of the proposition through a numerical example. Suppose, the objective function values of non-dominated solutions of the segments are as follow:

1) $(f_{o_1}(x_{11}), f_{o_2}(x_{11})) = (1,\ 2)$
2) $(f_{o_1}(x_{12}), f_{o_2}(x_{12})) = (3,\ 1)$
3) $(f_{o_1}(x_{21}), f_{o_2}(x_{21})) = (3,\ 5)$
4) $(f_{o_1}(x_{22}), f_{o_2}(x_{22})) = (4,\ 3)$

These values indicate that the solutions of the segments are non-dominated. Nevertheless, the combination of x_{11} with x_{22}, $(f_{o_1}(x_{11}, x_{22}), f_{o_2}(x_{11}, x_{22}))$, dominates the combination of x_{12} with x_{21}, $(f_{o_1}(x_{12}, x_{21}), f_{o_2}(x_{12}, x_{21}))$, which is in contradiction to the assumption of the proposition. In other words, $(1 + 4,\ 2 + 3)$ dominates $(3 + 3,\ 1 + 5)$. □

As the Cartesian combination of non-dominated solutions would generate some dominated solutions, a filtering operation is required to remove any dominated solutions from the obtained solution set. Initial experiments revealed that sequentially removing dominated solutions after the Cartesian combination of two solution sets performs much faster than removing dominated solutions at once, after Cartesian combinations of all the solution sets. Proposition 5 shows that removing dominated solutions after each Cartesian combination of two solution sets does not miss any non-dominated solution of the *true* Pareto set.

Proposition 5. *Removing dominated solutions after each Cartesian combination of non-dominated solutions of segments, does not lead to any failure in identifying the non-dominated solutions of the true Pareto set.*

Proof. : Assume $x' \in NDS$. Therefore, $f_{o_1}(x') < f_{o_1}(x)$ and $f_{o_2}(x') \leq f_{o_2}(x)$, $\forall x \notin NDS$. The proof is by contradiction. Let us consider a case consisting of three segments, where $x_i,\ x_i',\ and\ x_i'' \in NDS_{SG_i}$. Following Proposition 3, we have:

$$f_{o_1}(x_1') + f_{o_1}(x_2') + f_{o_1}(x_3') < f_{o_1}(x_1) + f_{o_1}(x_2) + f_{o_1}(x_3)$$

Now assume there exists a solution $x'' = (x_1'',\ x_2'')$ in the Cartesian combination of segments SG_1 and SG_2 which dominates $f_{o_1}(x_1') + f_{o_1}(x_2')$, i.e., $f_{o_1}(x_1'') + f_{o_1}(x_2'') < f_{o_1}(x_1') + f_{o_1}(x_2')$. Therefore $x' = (x_1',\ x_2')$ is removed from the solution set. Nevertheless, the combination of $x'' = (x_1'',\ x_2'')$ and x_3' will dominate x', i.e.

$$f_{o_1}(x_1'') + f_{o_1}(x_2'') + f_{o_1}(x_3') < f_{o_1}(x_1') + f_{o_1}(x_2') + f_{o_1}(x_3')$$

Therefore, x' cannot be within the NDS which is a contradiction. □

In summary, the proposed *segment-technique* can generate all the non-dominated solutions of the *true* Pareto set *iff* the objective functions are additive, i.e., $f_{o_k}(x) = \sum_{i=1}^{m} f_{o_k}(x_i),\ k = \{1,\ 2\}$. In this case, Proposition 3 guarantees that the *true* Pareto set of the whole task graph can be obtained from the Cartesian combination of non-dominated solutions of the *true* Pareto set of the segments. Proposition 4 shows that this Cartesian combination would generate dominated solutions that need to be removed. Finally, Proposition 5 states that removing

dominated solutions after each combination does not result in missing any non-dominated solutions of the *true* Pareto set.

6. Experiments

6.1. Experimental setup

The proposed *segment-technique* has been evaluated by conducting several simulations on real-world and synthetic synchronous task graphs. In the experiments, Montage, Epigenomic, LIGO, and Cybershake applications have been used as real-world task graphs [45]. The Montage has an application in astronomy and the Cybershake is used in characterizing the earthquake hazards in an area. The Epigenomic has an application in biology and the LIGO analyzes the unified data obtained from compact binary systems. These task graphs differ in their core structure, as shown in Fig. 6. In addition, some real-world-inspired task graphs have been generated randomly for the experiments. In this regard, the standard P-Method [46] has been used to generate realistic task graphs. P-Method is based upon a task graph adjacency matrix which is constructed by a probabilistic process. In P-Method, a Bernoulli process with connectivity parameter c ($0 \leq c \leq 1$) has been used to determine the connectivity among tasks.

In the experiments, the Xilinx™ Virtex-5 XUPV5LX110T FPGA [47] has been used to simulate a realistic reconfigurable computer. This device has 160 rows and 54 columns of CLBs in which each CLB group has 20 CLBs. As a result, there exist 8 CLB groups in each column. The experiments made by our research group shows that the time required to reconfigure a CLB group is 3.53 *ms* [9].

Different simulations have been carried out on both real-world and randomly-generated task graphs to evaluate the effectiveness of the proposed technique. In this regard, the scheduling of a task graph on the underlying FPGA without applying any FT techniques has been simulated on a PC to obtain the initial reliability and makespan of the task graph. Reliability has been obtained using Eq. (1) and makespan has been obtained using the *ASAP* scheduling strategy. The proposed technique has then been applied to the task graph to obtain its *true* Pareto set of FT techniques with reliability (Eq. (4)) and makespan as objective functions. For random task graphs, in each experiment, 100 different task graphs have been examined and the average of the obtained results has been presented as the final result.

These experiments correspond to the system testing life cycle which is done at both design time and runtime. Thus, the steps of the system testing life cycle [48] and its corresponding operations in the experiments are shown in Table 3.

In the first experiment, the positive effects of the proposed technique on the reliability of task graphs without makespan degradation have been examined. Then, the *segment-technique* has been compared with the *enumeration-based Pareto technique* to demonstrate the efficiency of the proposed technique (in terms of time complexity) in generating the *true* Pareto set of FT techniques. The proposed solution has then been applied to medium- and large-size task graphs and its results have been compared with three well-known evolutionary algorithms. Finally, the proposed *segment-technique* has been evaluated in harsh environments with dynamic SERs to demonstrate how it yields better reliability and makespan compared to state-of-the-art adaptive FT techniques. For the experiments, the SERs reported in [9] have been used. However, since the first three experiments require just one SER, the average of the minimum SER ($1.0E-7$) and the maximum SER ($5.0E-5$) is used in them.

All the aforementioned techniques have been implemented using C# 5.0 and have been experimented on Windows 10 operating system on a computer equipped with 4 GB RAM and Intel Core i5 processor (2.4 GHz, 3 MB cache).

6.2. Comparison with no FT technique (MTTF improvement)

The positive effects of the proposed *segment-technique* on improving the MTTF of task graphs without degrading their makespan have been examined in the first experiment. Thus, the task graphs have been first scheduled without applying any FT techniques to obtain their initial MTTF and makespan as the point of reference. Then, the *true* Pareto set of FT techniques of the task graphs have been generated using the proposed *segment-technique*. These FT techniques lead to different reliability and makespan tradeoffs. Thus, in this experiment, a solution that has the same makespan of the initial schedule but yields better reliability has been chosen from the solutions of the *true* Pareto set.

In this experiment, the real-world task graphs Montage, Epigenomic, LIGO, and Cybershake have been first examined individually. The obtained results are presented in Fig. 7. The obtained results demonstrate the ability of the proposed technique in improving the reliability of the task graphs without adversely affecting their makespan. In other words, by choosing appropriate FT techniques for each hardware task, the MTTF of these task graphs has been improved up to 68.73% but their makespan has not been prolonged.

In SRAM-based FPGAs, the reliability of task graphs is affected by different characteristics of hardware tasks [42]. Thus, more experiments have been conducted to disclose how the task CLB count and the task computation time affect the reliability improvement obtained by the proposed technique. It is worth noting to remark that the impacts of the task size and the task configuration delay have not been experimented separately since they depend on the task CLB count.

In this experiment, hardware tasks with 50 to 1500 CLBs have been used for the task graph generation. The results of this experiment are depicted in Fig. 8. The results show that the proposed technique improves the MTTF of the task graphs from 28.24% to 92.28% without prolonging their makespan. The results demonstrate that by using optimal FT techniques generated by the proposed technique, it is possible to hold the makespan of task graphs constant while their MTTF is improved substantially.

In the next experiment, hardware tasks with computation times from 50 *ms* to 500 *ms* have been examined whose results are illustrated in Fig. 9. The obtained results show moderate improvements for task

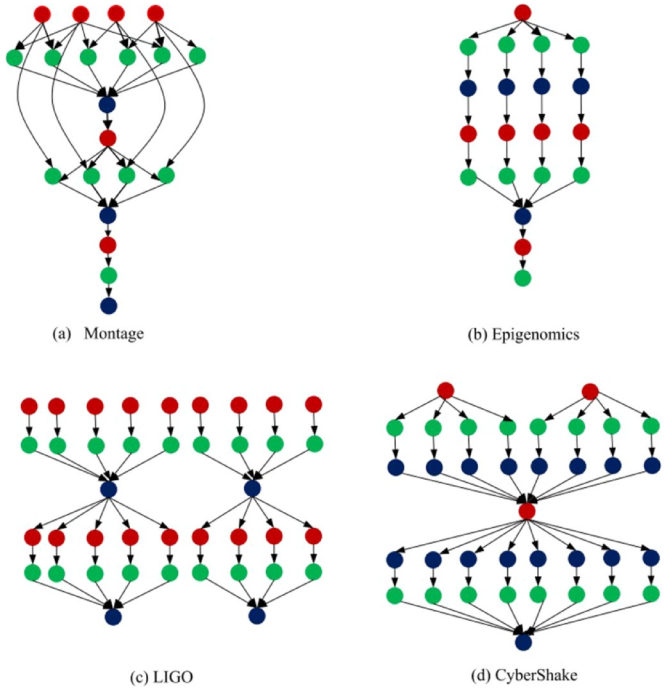

Fig. 6. Structure of four real-world task graphs [41].

Table 3
The system testing life cycle of the experiments.

Step	Operations	Stage
Requirement Analysis	Goal: testing task graphs on SRAM-based FPGAs in environments with dynamic SERs	Design time
Test Planning	Determining reliability and makespan as objective functions (testing goals)	Design time
Test Case Development	Determining the underlying FPGA and preparing target task graphs	Design time
Test Environment Setup	Measuring the current SER of the environment.	Runtime
Test Execution	Applying the proposed technique over task graphs and the underlying FPGA by considering the SER	Runtime
Test Closure	Obtaining the solution set of FT techniques and selecting the best one	Runtime

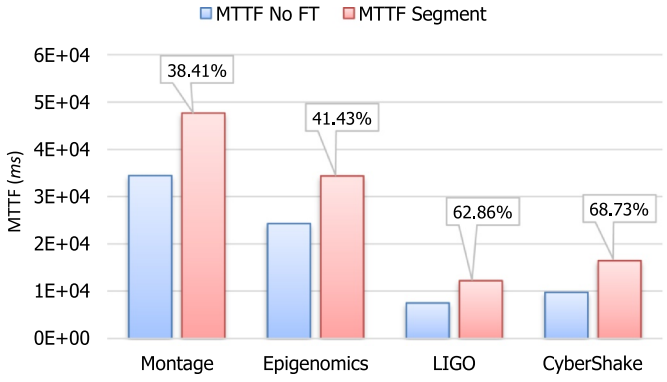

Fig. 7. The MTTF improvement achieved for real-world task graphs.

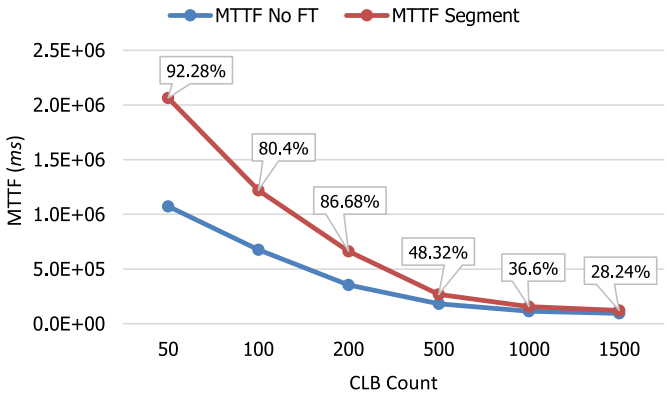

Fig. 8. The impacts of task size (CLB count) on the MTTF improvement.

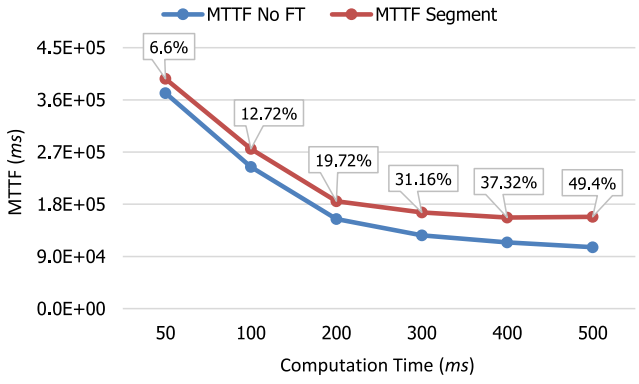

Fig. 9. The impacts of task computation time on the MTTF improvement.

graphs and these improvements increase by increasing the tasks' computation time. Similar to the previous experiments, in this experiment too, the MTTFs have been improved without prolonging the makespan. Nevertheless, supplementary experiments have shown that the configuration delay of tasks is a serious obstacle to the reliability and makespan optimization problem. In other words, the MTTF improvement of task graphs without degrading their makespan will be more significant when such configuration delays are alleviated [49]. The reason is that when the configuration delay of tasks is low, more tasks can be replicated with the lowest impact (in some cases no impact) on the task graph makespan. It gives more opportunity for replicating tasks that leads to more MTTF improvements.

6.3. Comparison with optimal FT technique (Time complexity)

As mentioned before, the *enumeration-based Pareto technique* presented in [9] is an exact method for the reliability and makespan optimization problem of general task graphs. It uses an exhaustive search strategy and analyzes the task graph as a whole to obtain the *true* Pareto set of FT techniques. In contrast, in our proposed technique, all non-dominated solutions of the segments of a synchronous task graph are first obtained separately and then sequentially combined to obtain the final *true* Pareto set of FT techniques. It also has been proved that such a strategy yields the same results as an exact method yields for the whole synchronous task graph. The *enumeration-based Pareto technique* suffers from the time complexity and its computation time increases exponentially by increasing the number of tasks. Thus, in this experiment, both real-world task graphs and the randomly-generated ones have been examined to compare the computation time of the proposed *segment-technique* with that of the *enumeration-based Pareto technique*.

In this experiment, several random task graphs which consist of 4 to 30 tasks have been first generated and examined whose obtained results are shown in Table 4. In the results, for each task count, the average number of non-dominated solutions and the computation times of both the proposed *segment-technique* and the *enumeration-based Pareto technique* [9] have been presented. The obtained results demonstrate that our proposed technique has much lower computation time and this improvement is more significant when the number of tasks increases. By increasing the number of tasks, the computation time of the *enumeration-based Pareto technique* increases drastically (on average, 5.23X for each step). The reason is that this technique enumerates all possible FT techniques for all the tasks and then selects the non-dominated ones

Table 4
Comparison of the computation time of the presented technique with the *enumeration-based Pareto technique* presented in [9] over random task graphs with different number of tasks.

#Tasks	#Solutions	Exact	Segment
4	18.12	13 *ms*	21 *ms*
6	38.85	40 *ms*	26 *ms*
8	62.65	88 *ms*	33 *ms*
10	107.7	355 *ms*	54 *ms*
12	141.47	1.43 s	78 *ms*
14	187.63	5.68 s	103 *ms*
16	239.3	22.72 s	170 *ms*
18	346.08	1.51 min	209 *ms*
20	495.57	6.10 min	264 *ms*
22	748.56	24.23 min	303 *ms*
24	1018.31	1.61 *hrs*	352 *ms*
26	1452.45	6.47 *hrs*	436 *ms*
28	2136.94	25.87 *hrs*	512 *ms*
30	2893.73	4.41 *days*	584 *ms*

for the solution set. In contrast, our proposed technique uses an exact optimization method for each segment separately which as a result, due to the lower number of tasks in the segments, leads to better computation times.

Similar experiments have been conducted on real-world task graphs whose results are presented in Table 5. As the results show, Montage and Epigenomic both have 20 tasks (see Fig. 6). In this case, as the *enumeration-based Pareto technique* does not pay attention to the shape of task graphs and it just enumerates all possible FT techniques for all tasks, its computation time is almost the same for them. In contrast, although the number of tasks of these two task graphs is equal, our presented technique takes almost different times for Montage and Epigenomic task graphs (16% discrepancy). The reason is that, instead of the whole number of tasks, the number of tasks in each segment most contributes to the computation time of the proposed *segment-technique*. The results also show that for larger task graphs (LIGO and Cybershake), the *enumeration-based Pareto technique* cannot find all non-dominated solutions in an acceptable time (we limited the computation time to 7 days), whereas our presented technique finishes the computation in 961 *ms* on average.

6.4. Comparison with evolutionary algorithms (GA, PSO, ant colony)

Evolutionary algorithms do not guarantee the generation of exact solutions for optimization problems but they take reasonable computation times. Thus, in the third set of experiments, the performance of the proposed technique has been compared with that of three well-known evolutionary and stochastic optimization techniques *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) [50], *Particle Swarm Optimization* (PSO) [51], and *Ant Colony Optimization* (ACO) [52]. These evolutionary algorithms have been selected as their performance in the reliability optimization problems has been demonstrated [53].

In this experiment, the *true* Pareto set of FT techniques has been first generated by applying the proposed *segment-technique* to medium- and large-size task graphs with 25–100 tasks. Then, *NSGA-II, PSO*, and *ACO* techniques have been applied to the same task graphs to obtain non-dominated solution sets. Since these evolutionary algorithms are stochastic and do not guarantee to yield optimal results, their obtained solutions might be different from those of the *true* Pareto set. Thus, two metrics have been used for the evaluation: *Recall* which denotes what percent of solutions of the *true* Pareto set is generated by the evolutionary algorithms and *Precision* which denotes what percent of solutions generated by the evolutionary algorithms belongs to the *true* Pareto set.

As mentioned before, the exploration space of the presented problem increases drastically by increasing the number of tasks. Thus, as the evolutionary algorithms search the exploration space stochastically, in this experiment, the computation time, as well as the recall and precision of the proposed technique, have been compared with those of the evolutionary algorithms.

The obtained results are tabulated in Table 6. In this table, first, the time required to generate the *true* Pareto set using our proposed technique has been presented. Then, the results obtained from *NSGA-II, PSO*, and *ACO* techniques have been shown. For this purpose, these techniques have been run for different time spans from 30 s to 1 hour to see how their evolution affects the recall and precision of the solutions.

As the proposed *segment-technique* generates the *true* Pareto set, it has been used as the point of reference (with recall and precision equal to 1). The evolutionary algorithms have been continuously run for 30 s, 1 min, 5 min, 10 min, 30 min, and 1 hour to generate non-dominated solutions. As the results of Table 6 show, the presented *segment-technique*, on average, takes 57 *ms* for task graphs with 10 tasks and takes 9.65 s for task graphs with 100 tasks. In contrast, not only the evolutionary algorithms take much more time for generating non-dominated solutions but also their obtained solutions are sub-optimal. For example, in the case of task graphs with 10 tasks, the best obtained result devotes to PSO with Recall = 93.83% and Precision = 97.51%, which has been obtained after 1 hour of computation. In addition, by increasing the number of tasks, the recall and precision of the evolutionary algorithms have been decreased. For example, by having 50 tasks in task graphs, after 1 hour of computation, the best recall and precision are 28.63% and 83.14%, respectively, whereas in this case, our proposed technique generates the *true* Pareto set in 1.28 s on average. With 75 tasks, PSO and ACO have not found any solutions and the NSGA-II in the best case has found only 0.16% of the solutions. Besides, in the case of 100 tasks, no solutions have been found by the evolutionary algorithms. The reason is that, as presented in [9], with n tasks and three redundancy levels (no FT, DWC, and TMR), the search space will contain 3^n different cases. Thus, it is very unlikely, if not impossible, to find *true* solutions among $3^{100} \approx 5.15E47$ cases using a stochastic approach. Finally, the average and maximum recall and precision values obtained by the evolutionary algorithms have been presented at the bottom of the table. The obtained results show that the proposed technique features efficient computation time and yields exact solutions. Thus, it is suitable to be used at runtime in systems operating in environments with dynamic and unpredicted SERs, as it will be shown in the next subsection.

6.5. Comparison with adaptive technique

As mentioned before, the SER of the environment affects the reliability of hardware tasks. An adaptive FT technique named *three-mode adaptive strategy* has been presented in [19] for space mission computer systems operating in environments with dynamic SERs. This technique divides the range of SERs of the environment into three intervals and in each interval applies a fixed FT technique to the tasks. For this purpose, it first estimates the minimum and maximum SER of the environment during the mission. Then, no FT technique is applied at runtime when the current SER is less than 10% of the expected SER range. It applies TMR when the current SER is more than 50% of the expected SER range. Otherwise, the tasks are duplicated.

In the fourth set of experiments, the *true* Pareto set of FT techniques generated by the proposed technique has been compared with the *three-mode adaptive strategy*. For this purpose, the SER of four solar conditions (Solar Max, Worst Week, Worst Day, and Peak 5-Min) of five orbits (GEO, GPS, MOL, Polar, and LEO) are used [49]. However, for simplicity, a representative subset of them are presented in this section. In this experiment, for different SERs, the adaptive strategy has been applied to task graphs and their reliability and makespan have been obtained. Then, the proposed *segment-technique* has been applied to the task graphs to generate the *true* Pareto set of FT techniques from which two solutions are chosen: 1) The solution that increases the reliability without prolonging the makespan, and 2) The solution that decreases the makespan without degrading the reliability. The results of the experiment are illustrated in Fig. 10.

As Fig. 10 shows, in the first SER range (0% – 10%), the adaptive strategy has not applied any FT techniques to the tasks. Therefore, it is not possible to further improve the makespan using the proposed *segment-technique*. However, the proposed technique yields significant MTTF improvements without degrading makespan. In the next SER ranges (10 – 50% and 50 – 100%), both makespan and MTTF of the task

Table 5
Comparison of the computation time of the presented technique with the *enumeration-based Pareto technique* presented in [9] over real-world task graphs.

Task Graph	#Tasks	#Solutions	Exact	Segment
Montage	20	386	6.71 min	141 *ms*
Epigenomic	20	465	6.66 min	168 *ms*
LIGO	40	8755	NaN	894 *ms*
Cybershake	36	5642	NaN	402 *ms*

Table 6
Comparison of the proposed technique with evolutionary algorithms NSGA-II, PSO, and ACO.

Iteration EAs ↓	#Tasks → / Metric → / Proposed Technique	10 Recall 100% 57 *ms*	10 Precision 100%	25 Recall 100% 373 *ms*	25 Precision 100%	50 Recall 100% 1.28 s	50 Precision 100%	75 Recall 100% 3.32 s	75 Precision 100%	100 Recall 100% 9.65 s	100 Precision 100%
30 s	NSGA-II	**55.10%**	78.45%	**50.12%**	73.66%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	PSO	51.44%	76.54%	47.13%	**75.98%**	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	ACO	49.14%	**80.32%**	31.24%	66.14%	**0.91%**	**12.50%**	0.00%	0.00%	0.00%	0.00%
1 min	NSGA-II	**88.24%**	**96.09%**	**69.44%**	86.49%	1.45%	18.12%	0.00%	0.00%	0.00%	0.00%
	PSO	82.63%	87.14%	65.63%	82.16%	0.98%	23.12%	0.00%	0.00%	0.00%	0.00%
	ACO	78.92%	83.56%	66.14%	**89.14%**	**4.52%**	**53.16%**	0.00%	0.00%	0.00%	0.00%
5 min	NSGA-II	**92.41%**	**96.88%**	**73.65%**	86.76%	19.89%	72.63%	**0.02%**	**0.15%**	0.00%	0.00%
	PSO	90.93%	94.16%	72.55%	83.14%	21.12%	69.12%	0.00%	0.00%	0.00%	0.00%
	ACO	88.56%	93.77%	71.44%	**90.32%**	**24.16%**	**83.14%**	0.00%	0.00%	0.00%	0.00%
10 min	NSGA-II	92.65%	96.89%	75.29%	85.38%	22.90%	73.85%	**0.09%**	**0.26%**	0.00%	0.00%
	PSO	**92.78%**	**97.42%**	**75.66%**	84.13%	24.16%	74.16%	0.00%	0.00%	0.00%	0.00%
	ACO	91.42%	94.86%	72.12%	**89.86%**	**25.85%**	**82.96%**	0.00%	0.00%	0.00%	0.00%
30 min	NSGA-II	92.69%	96.87%	75.35%	85.39%	24.12%	76.32%	**0.13%**	**0.31%**	0.00%	0.00%
	PSO	**93.75%**	**97.16%**	**76.14%**	84.52%	**27.16%**	81.96%	0.00%	0.00%	0.00%	0.00%
	ACO	91.43%	95.14%	73.06%	**90.16%**	26.74%	**83.14%**	0.00%	0.00%	0.00%	0.00%
1 h	NSGA-II	92.71%	96.87%	75.35%	85.39%	25.95%	77.66%	**0.16%**	**0.39%**	0.00%	0.00%
	PSO	**93.83%**	**97.51%**	**76.14%**	84.52%	**28.63%**	84.12%	0.00%	0.00%	0.00%	0.00%
	ACO	91.49%	95.26%	73.13%	**90.52%**	26.74%	**83.14%**	0.00%	0.00%	0.00%	0.00%
Total EAs	Average	82.39%	90.85%	69.27%	85.18%	16.96%	58.28%	0.02%	0.06%	0.00%	0.00%
	Max	93.83%	97.51%	76.14%	90.52%	28.63%	84.12%	0.16%	0.39%	0.00%	0.00%

graphs have been improved without deteriorating the other metric. Besides, within a given SER range, the amount of MTTF and makespan improvements are increased by increasing the SER. The results show that even though the adaptive strategy has positive effects on the system reliability in environments with dynamic SERs, the proposed *segment-technique* yields better reliability and makespan, and, due to its efficient computation time, it is suitable to be used at runtime in systems operating in SER-varying environments.

7. Conclusions

In this paper, a *segment-technique* has been presented to address the problem of reliability and makespan optimization of synchronous task graphs running on SRAM-based FPGAs in environments with dynamic and unpredicted SERs. In these systems, once the SER changes a new set of FT techniques should be generated at runtime to be applied to the running tasks in order to adjust the system reliability and performance.

The presented approach generates the *true* Pareto set of FT techniques in three steps including the task graph segmentation, generating the *true* Pareto set of the segments separately, and finally combining solutions of the segments using the Cartesian product operator and removing dominated ones from the solution set. In this regard, it has been proved that such a strategy generates the *true* Pareto set as an exact method generates for the whole task graph with low time complexity.

Different experiments have been conducted to evaluate the efficiency of the proposed technique. Thus, the first set of experiments have been performed to demonstrate the positive impacts of the proposed solution on the system reliability. The obtained results show that by using optimal FT techniques generated by the proposed technique, it is possible to improve the MTTF of task graphs by 46.30% on average without deteriorating their makespan. The second and the third set of

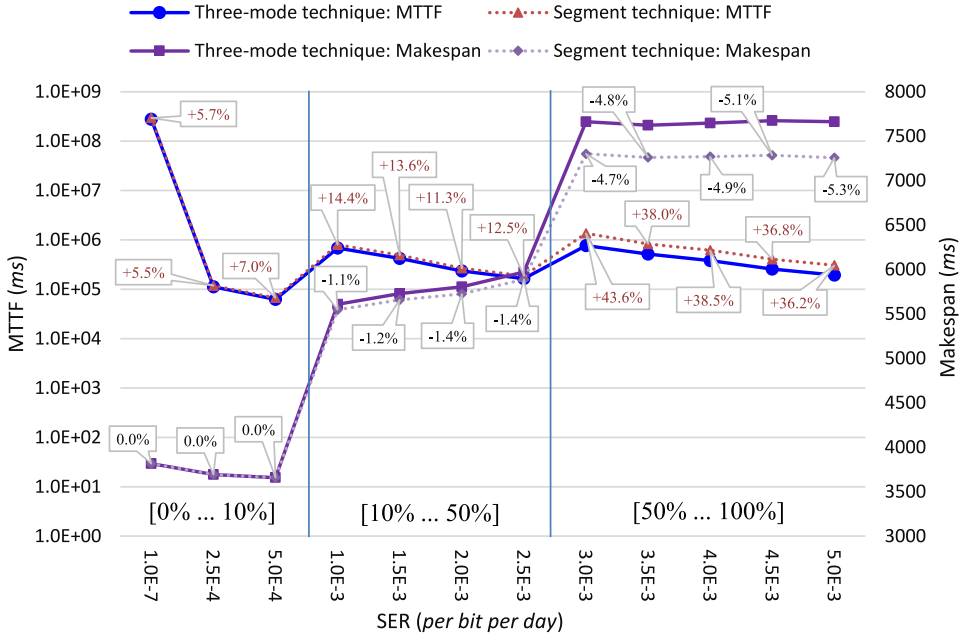

Fig. 10. MTTF and makespan improvements of the proposed technique over *three-mode adaptive strategy* [19] (left y-axis scale is logarithmic).

experiments have demonstrated the efficient computation time of the proposed solution for small-, medium-, and large-size task graphs with respect to other exact and evolutionary optimization methods. Finally, supplementary experiments have been performed to reveal the positive impacts of the proposed technique on the runtime systems that are employed in environments with dynamic and unpredicted SERs. The obtained results have demonstrated that the presented approach gains better reliability and makespan over state-of-the-art adaptive FT techniques with a reasonable time in dynamic environments.

The presented approach is limited to synchronous task graphs. Thus, proposing efficient solutions for generating the *true* Pareto set of FT techniques for general task graphs is recommended as a direction for future work.

Declaration of Competing Interest

None.

References

[1] Ramezani R. Dynamic scheduling of task graphs in multi-FPGA systems using critical path. J Supercomput 2020:1–22. https://doi.org/10.1007/s11227-020-03281-3.
[2] Engelen S, Gill E, Verhoeven C. On the reliability, availability, and throughput of satellite swarms. IEEE Trans Aerosp Electron Syst 2014;50(2):1027–37.
[3] Ramezani R. A prefetch-aware scheduling for FPGA-based multi-task graph systems. J Supercomput 2020;76:7140–60. https://doi.org/10.1007/s11227-020-03153-w.
[4] Jung S, Choi JP. Predicting system failure rates of SRAM-based FPGA on-board processors in space radiation environments. Reliab Eng Syst Safety 2019;183:374–86.
[5] Gao Z, Zhang L, Han R, Reviriego P, Li Z. Reliability Evaluation of Turbo Decoders Implemented on SRAM-FPGAs. 38th VLSI Test Symposium (VTS). IEEE; 2020. p. 1–6.
[6] Wei W, Namba K, Kim Y-B, Lombardi F. A Novel Scheme for Tolerating Single Event/Multiple Bit Upsets (SEU/MBU) in Non-Volatile Memories. IEEE Trans Comput 2016;65(3):781–90.
[7] Ghavidel A, Sedaghat Y, Naghibzadeh M. Hybrid scheduling to enhance reliability of real-time tasks running on reconfigurable devices. J Supercomput 2020;76:4701–30.
[8] Ramezani R, Sedaghat Y, Naghibzadeh M, Clemente JA. A decomposition-based reliability and makespan optimization technique for hardware task graphs. Reliab Eng Syst Safety 2018;180:13–24.
[9] Ramezani R, Sedaghat Y, Naghibzadeh M, Clemente JA. Reliability and Makespan Optimization of Hardware Task Graphs in Partially Reconfigurable Platforms. IEEE Trans Aerosp Electron Syst 2017;53(2):983–94.
[10] *System, Software, and Hardware Verification and Validation (Standard 1012-2016)*, IEEE, 2016.
[11] *Systems and Software Engineering - Life Cycle Processes - Requirements Engineering (Standard 29148-2018)*, ISO/IEC/IEEE, 2018.
[12] *General Principles of Reliability Analysis of Nuclear Power Generating Station Systems and Other Nuclear (Standard 352-2016)*, IEEE, 2016.
[13] Pullum LL. Software fault tolerance techniques and implementation. Artech House; 2001.
[14] Chiang M-C, Huang C-Y, Wu C-Y, Tsai C-Y. Analysis of a Fault-Tolerant Framework for Reliability Prediction of Service-Oriented Architecture Systems. IEEE Trans Reliab 2020:1–36. https://doi.org/10.1109/TR.2020.2968884.
[15] Kastensmidt FL, Reis R. Fault-tolerance techniques for SRAM-based FPGAs. Springer; 2007.
[16] Li T, Liu H, Yang H. Design and Characterization of SEU Hardened Circuits for SRAM-Based FPGA. IEEE Trans Very Large Scale Integr (VLSI) Syst 2019.
[17] Zhao Z, Nguyen NT, Agiakatsikas D, Lee G, Diessel O. Fine-grained module-based error recovery in FPGA-based TMR systems. ACM Trans Reconfig Technol Syst (TRETS) 2018;11(1):4.
[18] Clark LT, Patterson DW, Ramamurthy C, Holbert KE. An embedded microprocessor radiation hardened by microarchitecture and circuits. IEEE Trans Comput 2016;65(2):382–95.
[19] Jacobs A, Cieslewski G, George AD, Gordon-Ross A, Lam H. Reconfigurable fault tolerance: a comprehensive framework for reliable and adaptive FPGA-based space computing. ACM Trans Reconfig Technol Syst (TRETS) 2012;5(4):232–63.
[20] Mandal S, Sarkar S, Ming WM, Chattopadhyay A, Chakrabarti A. Criticality aware soft error mitigation in the configuration memory of SRAM based FPGA. 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID). IEEE; 2019. p. 257–62.
[21] *In-system configuration of programmable devices (standard 1532-2002)*, IEEE, 2002.
[22] Morgan KS, et al. 10 Fault Tolerance Techniques and Reliability Modeling for SRAM-Based FPGAs. Radiat Effects Semicond 2018:140.
[23] Zhang R, Xiao L, Li J, Cao X, Li L. An adjustable and fast error repair scrubbing method based on xilinx essential bits technology for SRAM-based FPGA. IEEE Trans Reliabil 2019;69(2):430–9.
[24] Yin J-Y, Guo G-C, Wu Y-X. A hybrid fault-tolerant scheduling algorithm of periodic and aperiodic real-time tasks to partially reconfigurable FPGAs. International Workshop on Intelligent Systems and Applications (ISA). IEEE; 2009. p. 1–5.
[25] Yin J, Zheng B, Sun Z. A hybrid real-time fault-tolerant scheduling algorithm for partial reconfigurable system. J Comput (Taipei) 2012;7(11):2773–80.
[26] Ramezani R, Sedaghat Y. Scheduling periodic real-time hardware tasks on dynamic partial reconfigurable devices subject to fault tolerance. 4th International eConference on Computer and Knowledge Engineering (ICCKE). IEEE; 2014. p. 1–6.
[27] Jacobs A, Wulf N, George AD. Task scheduling for reconfigurable systems in dynamic fault-rate environments. High Performance Extreme Computing Conference (HPEC). IEEE; 2013. p. 1–6.
[28] Vallero A, Carelli A, Di Carlo S. Trading-off reliability and performance in FPGA-based reconfigurable heterogeneous systems. 13th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS). IEEE; 2018. p. 1–6.
[29] Hoque KA, Mohamed OA, Savaria Y. Dependability modeling and optimization of triple modular redundancy partitioning for SRAM-based FPGAs. Reliabil Eng Syst Safety 2019;182:107–19.
[30] Ramezani R, Clemente JA, Sedaghat Y, Mecha H. Estimation of hardware task reliability on partially reconfigurable FPGAs. 16th European Conference on Radiation and Its Effects on Components and Systems (RADECS). IEEE; 2016. p. 1–4.
[31] Héron O, Arnaout T, Wunderlich H-J. On the reliability evaluation of SRAM-based FPGA designs. International Conference on Field Programmable Logic and Applications (FPL). IEEE; 2005. p. 403–8.
[32] Ostler PS, et al. SRAM FPGA reliability analysis for harsh radiation environments. IEEE Trans Nucl Sci 2009;56(6):3519–26.
[33] Kretzschmar U, Gomez-Cornejo J, Astarloa A, Bidarte U, Del Ser J. Synchronization of faulty processors in coarse-grained TMR protected partially reconfigurable FPGA designs. Reliab Eng Syst Safety 2016;151:1–9.
[34] Villalta I, Bidarte U, Gómez-Cornejo J, Jiménez J, Lázaro J. SEU emulation in industrial SoCs combining microprocessor and FPGA. Reliab Eng Syst Safety 2018;170:53–63.
[35] Lee J-Y, Feng Z, He L. In-place decomposition for robustness in FPGA. International Conference on Computer-Aided Design (ICCAD). IEEE/ACM; 2010. p. 143–8.
[36] Garg H, Sharma S. Multi-objective reliability-redundancy allocation problem using particle swarm optimization. Comput Ind Eng 2013;64(1):247–55.
[37] Kleinberg J, Tardos É. Algorithm design. India: Pearson Education; 2006.
[38] Omranian-Khorasani S, Naghibzadeh M. Deadline constrained load balancing level based workflow scheduling for cost optimization. 2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA). IEEE; 2017. p. 113–8.
[39] Kanagaraj K, Swamynathan S. Structure aware resource estimation for effective scheduling and execution of data intensive workflows in cloud. Future Gener Comput Syst 2018;79:878–91.
[40] Arabnejad V, Bubendorfer K, Ng B. Budget and deadline aware e-science workflow scheduling in clouds. IEEE Trans Parallel Distrib Syst 2018;30(1):29–44.
[41] Adhikari M, Amgoth T. An intelligent water drops-based workflow scheduling for IaaS cloud. Appl Soft Comput 2019;77:547–66.
[42] Ramezani R, Clemente JA, Franco FJ. Analytical Reliability Estimation of SRAM-based FPGA Designs against Single-bit and Multiple-cell Upsets. Reliab Eng Syst Safety 2020;202. https://doi.org/10.1016/j.ress.2020.107036.
[43] Shooman ML. Reliability of computer systems and networks: fault tolerance, analysis, and design. John Wiley & Sons; 2003.
[44] Liu G, Zeng Y, Li D, Chen Y. Schedule length and reliability-oriented multi-objective scheduling for distributed computing. Soft Comput 2014:1–11.
[45] Singh V, Gupta I, Jana PK. A novel cost-efficient approach for deadline-constrained workflow scheduling by dynamic provisioning of resources. Future Gener Comput Syst 2018;79:95–110.
[46] Al-Sharaeh S, Wells BE. A comparison of heuristics for list schedules using the Box-method and P-method for random digraph generation. 28th Southeastern Symposium on System Theory. IEEE; 1996. p. 467–71.
[47] XilinxCorporation, "Virtex-5 FPGA Configuration User Guide, UG191 (v 3.11)," online at:www.xilinx.com/support/documentation/user_guides/ug191.pdf, 2012.
[48] O'Regan G. Concise guide to software testing. Springer; 2019.
[49] Ramezani R, Sedaghat Y, Clemente JA. Reliability Improvement of Hardware Task Graphs via Configuration Early Fetch. IEEE Trans Very Large Scale Integr (VLSI) Syst 2017;25(4):1408–20.
[50] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002;6(2):182–97.
[51] Khalili-Damghani K, Abtahi A-R, Tavana M. A new multi-objective particle swarm optimization method for solving reliability redundancy allocation problems. Reliab Eng Syst Safety 2013;111:58–75.
[52] Zhao J-H, Liu Z, Dao M-T. Reliability optimization using multiobjective ant colony system approaches. Reliab Eng Syst Safety 2007;92(1):109–20.
[53] Jahromi AE, Feizabadi M. Optimization of multi-objective redundancy allocation problem with non-homogeneous components. Comput Ind Eng 2017;108:111–23.