Characterizing and Improving Resilience of Accelerators to Memory Errors in Autonomous Robots

DEVAL SHAH, University of British Columbia, Canada
ZI YU XUE*, Massachusetts Institute of Technology, USA
KARTHIK PATTABIRAMAN, University of British Columbia, Canada
TOR M. AAMODT, University of British Columbia, Canada

Motion planning is a computationally intensive and well-studied problem in autonomous robots. However, motion planning hardware accelerators (MPA) must be soft-error resilient for deployment in safety-critical applications, and blanket application of traditional mitigation techniques is ill-suited due to cost, power, and performance overheads. We propose Collision Exposure Factor (CEF), a *novel metric* to assess the failure vulnerability of circuits processing spatial relationships, including motion planning. CEF is based on the insight that the safety violation probability increases with the surface area of the physical space exposed by a bit-flip. We evaluate CEF on four MPAs. We demonstrate empirically that CEF is correlated with safety violation probability, and that CEF-aware selective error mitigation provides 12.3×, 9.6×, and 4.2× lower dangerous Failures-In-Time rate on average for the same amount of protected memory compared to uniform, bit-position, and access-frequency-aware selection of critical data. Furthermore, we show how to employ CEF to enable fault characterization using 23,000× fewer fault injection (FI) experiments than exhaustive FI, and evaluate our FI approach on different robots and MPAs. We demonstrate that CEF-aware FI can provide insights on vulnerable bits in an MPA while taking the same amount of time as uniform statistical FI. Finally, we use the CEF to formulate guidelines for designing soft-error resilient MPAs.

CCS Concepts: • **Computer systems organization** → **Reliability**; **Robotics**.

Additional Key Words and Phrases: Reliability, Autonomous robots, Motion planning, Collision detection.

1 INTRODUCTION

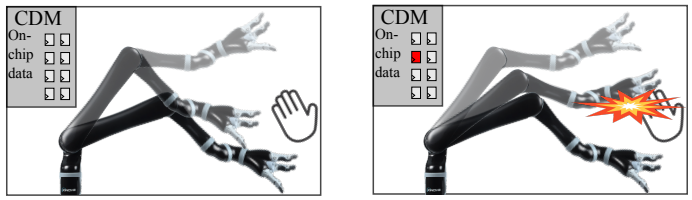
Autonomous robots are increasingly used for real-time and safety-critical tasks, including medical care [26, 42], autonomous driving [122, 126], and home assistance [67, 136]. The size of autonomous robots market is expected to grow by 4× from 2022 to 2030 [14, 58]. As autonomous robots are becoming an integral part of our lives, it is crucial to ensure that an autonomous robot does not collide with other objects, and thereby harm the safety of its surroundings.

Motion planning allows an autonomous robot to navigate and reach its end goal safely without collisions. Therefore, motion planning is key to the many tasks performed by autonomous robots, including navigation, object manipulation, footstep planning, and full-body movement. It has been an area of study since the 1970s [66, 70], and is today one of the key research topics in robotics. For example, motion planning constitutes about ∼10% of the total publications in top robotics conferences [2, 3] (ICRA-2022 had 931 papers, 115 of which were in motion planning). Motion planning has also received significant attention from industry, with an increasing number of patents. For example, the number of patents granted every year by the United States Patent Office (USPTO) on collision detection and motion planning has increased by 8× from 2015 to 2022 [39, 127].

The computational and real-time demands of motion planning exceed those provided by typical CPUs. Recently, several approaches have been proposed to accelerate motion planning on different

*Work performed while at UBC

This research has been funded in part by the National Sciences and Engineering Research Council of Canada (NSERC) through a NSERC Strategic Project Grant.
Authors' addresses: Deval Shah, University of British Columbia, Canada, devalshah@ece.ubc.ca; Zi Yu Xue, Massachusetts Institute of Technology, USA, fzyxue@mit.edu; Karthik Pattabiraman, University of British Columbia, Canada, karthikp@ece.ubc.ca; Tor M. Aamodt, University of British Columbia, Canada, aamodt@ece.ubc.ca.



(a) Error-free scenario (b) Faulty scenario

Fig. 1. The effect of soft errors on safety in autonomous robots.

hardware platforms, including GPUs [12, 35], FPGAs [7, 91, 113], and ASICs [10, 77, 89, 112, 114, 139]. Motion Planning Accelerators (MPAs) have achieved impressive performance gains and are being adopted in industry [28, 102, 106]. However, the use of MPAs in robotics applications has significant safety implications. For example, the IEC 61508 [57] provides functional safety standards for electronic systems used in applications such as robotics in terms of allowable dangerous failure rate per hour (Section 2.3). In modern systems, errors induced by particle strikes and radiation, or soft errors, make up the majority of SRAM and register-level faults [84, 116]. Hence, soft errors are a major threat to safety standards compliance in MPAs used in robotics. Furthermore, due to their transient nature, soft errors cannot be eliminated during the design and test phases of a chip, and hence need runtime mitigation.

There have been many studies on the effects of soft errors on CPUs, GPUs, and FPGAs [31, 32, 38, 47, 49, 64, 75, 76, 88, 94, 96–98, 110, 123, 124, 128, 131], deep learning accelerators [53, 54, 74, 78, 101, 108, 125, 134], and autonomous vehicle systems [11, 56, 61–63, 135]. However, their impact on accelerators for robotics has not been studied. Application-agnostic blanket error mitigation techniques such as error-correcting codes (ECC) and dual/triple modular redundancy (DMR/TMR) can increase the area, cost, and power by 12%-125% (Table 5), and degrade the performance of these hardware accelerators. With consumer applications driving growth of robotics, the electronics controlling these systems will become increasingly cost-sensitive [23, 37, 93]. Furthermore, any increase in the MPA's power consumption significantly reduces a mobile robot's operation time as MPAs suitable for real-time motion planning can contribute to 15%-50% of its total power consumption [69, 79, 91]. While there has been work on sensor and actuator faults in robotics [22], there has been no study of the reliability of MPAs in the presence of soft errors. *To the best of our knowledge, we are the first paper on characterizing and improving the reliability of MPAs.*

In this paper, we study soft errors in the collision detection module (CDM), which is the largest, most energy-consuming, and safety-critical component in MPAs [7, 77, 89, 114]. We find that CDMs in these accelerators consist of on-chip storage elements to store the information about space that the robot passes through for possible motions. These storage elements account for more than 97% of the sequential elements (based on our RTL synthesis experiments), and so we focus on it in this paper. Figure 1 shows the effect of soft errors in the CDM. In the error-free scenario (Figure 1a), the robot navigates safely to the end goal. However, in Figure 1b, a soft error causes the MPA to misidentify a path taken by the robot as safe, thereby resulting in a collision. This mis-identification is due to an error modifying the geometric representation of the space that the robot passes through.

Prior work has explored memory and register file designs that allow flexible partition into protected and non-protected regions for incorporating selective error mitigation in systems using CPUs and GPUs [16, 86, 133, 140, 141]. These techniques protect only the most vulnerable data and do so by placing it in protected memory. A challenge with applying such an approach to error mitigation is that it requires accurate and fast identification of the most critical data. One approach is to use exhaustive fault injection (FI) to identify storage bits that exhibit the highest resilience improvement

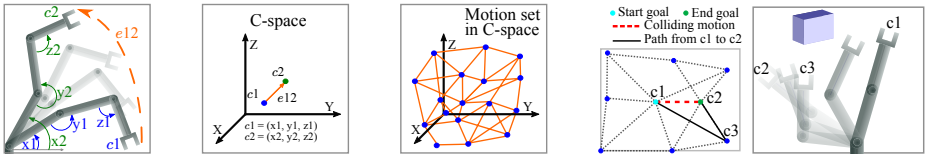
when protected from soft errors. Unfortunately, exhaustive FI can take up to 24,000 CPU hours for a typical MPA (Section 6.1.1). Such high FI time overhead for error mitigation each time the MPA is reconfigured for a different task or robot introduces practical deployment challenges. For fully autonomous robots, the MPA's storage data can be generated or modified at runtime [34, 52, 60], requiring runtime characterization of vulnerable data for selective error mitigation. Also, as noted in earlier work [88] exhaustive FI is an inefficient way to gain insight during architecture design.

Prior work on FI techniques for CPUs and GPUs [31, 32, 49, 76, 92, 124, 131] has obtained significant reductions in the FI time. These FI tools and techniques are targeted towards specific languages, Instruction Set Architectures (ISA), or CPU/GPU system simulators and often exploit the microarchitecture or ISA to reduce the FI time and estimate the failure probability (Section 8). However, these techniques cannot be directly applied to robotic accelerators that use specialized microarchitectures and instruction sets. Thus, there is a need for techniques that efficiently characterize the effect of soft errors in robotics applications. Architectural Vulnerability Factor (AVF) has been widely used to define the vulnerability of a structure and can be measured using an analytical method such as Architecturally Correct Execution (ACE) analysis [88] or FI [73]. Directly applying AVF methodology such as ACE analysis to MPAs requires considering the positions of obstacles in the environment, thus leading to the need to run a large number of simulations to accurately estimate the fraction of time a hardware storage element contains an ACE bit.

To overcome these challenges, we introduce a novel heuristic, *Collision Exposure Factor (CEF)* that depends only upon the accelerator and robot, not on the environment. Other heuristics, such as bit position [74] and access-frequency [68, 83] have been proposed to find critical bits for deep learning accelerators and embedded applications, respectively. However, our analysis shows that considerable variation exists in the failure probabilities of bits with the same access-frequency or bit position in MPAs. In contrast, our approach provides a higher reduction in the overall failure probability as our proposed heuristic is more accurate at finding critical bits in the MPAs (Section 6.2).

The CEF estimates the vulnerability factor of memory bits storing spatial information. The 3D model data of the swept spaces of a robot's possible motions play a key role in real-time collision detection and motion planning, and is stored in the on-chip memory. Each bit in the on-chip storage helps specify the bounds of some motion's swept space. We define a bit's *critical space* as the region excluded from the swept space if that bit changes value due to a fault. A bit-error can lead to erroneous collision detection and safety violations if an obstacle overlaps with this bit's critical space. To account for this violation, CEF of a bit is defined as the surface area of that bit's critical space exposed to obstacles due to a soft error. The CEF is defined for a memory bit, and the probability of a fault in a bit resulting in a safety violation monotonically increases with its CEF value. Thus CEF values of memory bits can be used to estimate their relative vulnerability factors. In contrast, AVF estimates the failure probability due to soft error for a structure, and not in individual bits.

The probability of a fault in a bit resulting in a safety violation depends upon the environmental characteristics (e.g., average size and number of obstacles). By considering the entire swept space of the robot's motion, the CEF decouples the effects of the robot model and the environment on safety violations, and the safety violation probability of a bit monotonically increases with its CEF value. CEF can be calculated once for a given MPA and robot *without* needing to consider obstacles in the environment (which might be highly dynamic). The underlying reason this separation is possible is that, in the most widely used approaches for real-time motion planning, a robot has a fixed set of possible short motions that are precomputed independently of obstacle positions [72]. During operation a subset of these motions is selected to navigate to a given goal depending upon obstacle positions. For purposes of fault analysis we decouple these steps by assuming a conservative distribution (e.g., uniform) on where objects will appear. We show empirically that the CEF computed this way *independent of the environment* is positively correlated with the average failure probability.



(a) (b) (c) (d)

Fig. 2. (a) shows the spatial poses and motion of a 2D robot with three degrees of freedom (x, y, z) (figure reproduced from [112]), (b) represents these poses and motion in the robot's C-space, and (c) shows a motion set in the C-space. (d) shows a motion set for a robot with two DOFs. The figure highlights a path between c1 and c2 in the presence of an obstacle in c-space (left) and physical space (right).

Further, we propose a CEF-aware error mitigation technique to selectively protect values with higher CEF in an MPA's on-chip storage. Finally, we propose a two-phase FI methodology: Phase 1 FI to find the CEF of all the bits (for a given robot and MPA), and Phase 2 FI to find the relation between the CEF and failure probability with fault site pruning. Uniform statistical FI-based characterization of the CDM provides similar speedup over exhaustive FI as CEF-aware FI (Section 6.1.1). However, it does not find the safety-violation probability of an individual bit nor does it find relative vulnerability of different bits, which is needed for error mitigation. In CEF-aware FI, on the other hand, decoupling the two FI phases allows efficient calculation of the safety-violation probability for every bit in the CDM. In summary, we make the following contributions in this paper:

- Establish the necessary conditions for safety violations (i.e., collisions), and propose Collision Exposure Factor (CEF), a reliability metric for CDM storage elements.
- Propose an efficient *CEF-aware error mitigation* technique that selectively protects values with higher CEF, and compare it to uniform, exhaustive FI, bit position-aware, and access-frequency-aware application of DMR, TMR, and ECC techniques for four CDM designs.
- Propose a two-phase FI methodology using the CEF of storage elements for fault site pruning to reduce FI time by orders of magnitude.

Our results show that CEF-aware selective error mitigation results in 12.3×, 9.6×, and 4.2× lower failure rate for the same amount of protected memory compared to uniform, bit position, and access-frequency-aware selection of critical data. Further, CEF-aware FI reduces the FI time by 23,000× with minimal loss of accuracy, and finds the failure rate of the MPA and individual bits. Finally, we study the impact of architectural design parameters on resilience and error mitigation overheads, and demonstrate the potential for designing resilient MPAs architectures using the CEF.

2 BACKGROUND AND MOTIVATION

This section briefly summarizes relevant background information on motion planning, general CDM architecture, and safety in robotics. Then, we summarize the limitations of current FI methodologies, and discuss the need for application-specific reliability metric and FI methodology for MPAs.

2.1 Motion Planning

The objective of motion planning is to compute a collision-free path for a robot. Motion planning is performed in the robot's configuration space (C-space), which has the same number of dimensions as the robot's degrees of freedom (DOFs). Figure 2a shows a planar robot with three DOFs, while Figure 2b represents its C-space. Each dimension of the C-space represents the range of angles of a joint of the robot. The point $c1$ with coordinates (x1, y1, z1) in the C-space corresponds to the pose $c1$ in Figure 2a. Similarly, edge $e12$ represents the robot's motion from $c1$ to $c2$. The complexity of motion planning increases exponentially with the robot's DOFs [109]. Thus, approximate methods

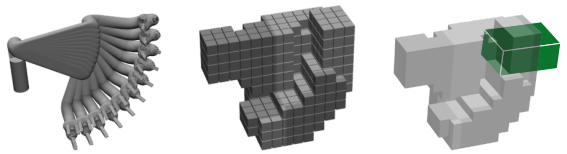


Fig. 3. (a) shows the swept space of a robot motion, (b) represents the voxelization of the swept space (explained in Section 2.2), and (c) shows an obstacle that overlaps with the swept space of this motion.

such as probabilistic roadmaps [65] are widely used [71, 109] over the last 20 years. Many hardware accelerators [7, 77, 89, 90, 113, 139] have been proposed for these algorithms.

2.1.1 Probabilistic roadmaps. Probabilistic motion planning consists of two phases. In the pre-processing stage, a graph, called a *motion set*, is constructed in the robot's C-space. Figure 2c and Figure 2d give examples of motion sets for a robot with three and two DOFs, respectively. The nodes in the graph correspond to the robot's spatial poses, and an edge between close-by nodes represent motion generated by a local planner (e.g., linear interpolation between two poses) from one pose to another. Motion set consists of collision-free poses and motions of the robot for given environment. In the query phase, a path search algorithm (e.g., Dijkstra's algorithm) is used to find a path between given start and end poses using the precomputed motion set. A path consists of one or more motions from the motion set. The probabilistic roadmap is a multi-query method, i.e., the same motion set is used to perform motion planning for multiple start and goal poses.

Leven and Hutchinson [72] proposed a real-time motion planning approach based on probabilistic roadmaps for a dynamic environment and is also used by MPAs [46, 77, 89, 90, 113, 139]. In this approach, the motion set is generated for an obstacle-free environment. At runtime, collision detection is performed to find collision-free motions in this set for a given placement of obstacles. The path search module considers collision-free motions to find a feasible path between the start and end pose in the resulting *collision-free* motion set graph. For example, in Figure 2d, an obstacle in the robot's environment makes some of the motions in the motion set flagged as "colliding motion", which are avoided by the path search module to find a path between c1 and c2.

The information about the swept space of each motion in the motion set is stored in the memory to facilitate real-time collision detection. Swept space of a motion is the space occupied by a robot as the given motion is followed. For example, as shown in Figure 3a and Figure 3b, a motion's swept space is discretized, and its 3D model is stored. As shown in Figure 3c, the motion's swept space is checked for overlap with obstacles for collision detection at runtime. Several motion planning approaches propose to modify this motion set at runtime [34, 52, 60]. For these approaches, the swept spaces of new motions are generated at runtime outside of the critical path.

2.2 Collision Detection Module (CDM)

The collision detection step is the most time- and energy-consuming in motion planning and takes up to 99% of the total runtime on a CPU [12]. As collision detection must be performed in real-time with low latency to ensure safety in an environment with dynamic obstacles, MPAs typically dedicate more than 85% of their total area to accelerate collision detection [77, 89, 90], making the CDM the most vulnerable component in an MPA. An erroneous collision detection can potentially lead to a collision between the robot and an obstacle in its surroundings, making collision detection safety-critical. Many approaches have been proposed to accelerate collision detection on different hardware platforms. The architecture proposed by Murray et al. [91] uses specialized combinational circuits for a given motion set, but it is not reconfigurable to different motion sets at runtime. In

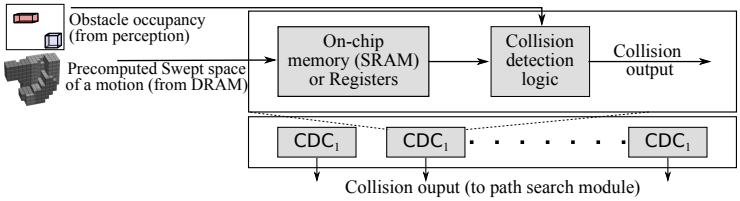


Fig. 4. Architecture of a Collision Detection Module (CDM).

contrast, GPU-based collision detection acceleration [12, 35] provides high flexibility, but it is not energy-efficient. Configurable collision detection hardware accelerators [7, 10, 77, 89, 90, 113, 139] provide a balance between flexibility and performance/energy-efficiency. We focus on the fault characterization of such reconfigurable CDMs of four MPAs, which we refer to as A1 [91], A2 [89], A3 [77], and A4 [139] (Section 5.2) throughout this paper.

Figure 4 illustrates the architecture of a generic configurable CDM. The CDM consists of multiple collision detection circuits (CDCs) to check multiple motions for collision in parallel. Each CDC consists of on-chip memory/registers to store a motion's swept space, and exploits inter- and intra-query data reuse. The on-chip storage can be configured for different motions at runtime. Based on our synthesis experiments, the on-chip memory to store swept spaces constitutes more than 97% of the sequential elements and $\sim$ 50% of the total area in CDMs. Earlier work on FPGA-based accelerators [89] uses combinational logic to encode swept-space data. However, most programmable CDMs have a high fraction of sequential elements dedicated to on-chip storage for spatial data [77, 89, 90, 139]. We also verify this using RTL synthesis of these programmable CDMs (architecture explained in Section 5.2). Therefore, we focus on the on-chip memory for fault characterization, and do not consider the effect of soft errors in combinational logic in this work.

The key design consideration of CDMs is the geometric representation used to store swept spaces in the memory. Several representations for storing the 3D model have been proposed for motion planning, including polygonal meshes [5], bounding box hierarchies [40, 130], voxels [114], and octrees [59]. These approaches differ in storage requirements, representation accuracy, and computational complexity. All CDM hardware accelerators studied in this work use either grid-based or octree-based representations (explained in Section 3.1) as these representations are less compute- and memory-intensive. For both representations, the swept space of a motion is discretized into fixed-size cubes known as *voxels*. Voxelized swept space is then stored using a set of structures specific to the representation used. At runtime, a perception sensor module senses obstacle occupancy information, converts it to voxels, and sends it to the CDCs. The CDCs perform collision detection for stored motions and send the output to the path search module. Note that the proposed metric CEF is applicable regardless of the geometric representation used.

2.3 Functional Safety in Robotics

Safety is a crucial consideration in robotics. Hence, the failure rate of circuits used in robotics applications, including MPAs, is an important factor. The FIT rate of a circuit (where 1 FIT is one failure/billion hours) consisting of multiple components can be computed using Equation (1) [74, 88].

$$\text{FIT} = \sum_{i \in \text{components}} \text{S}_i \times \text{SDC}_i \times \text{FIT}_{\text{Raw}} \tag{1}$$

FIT_{Raw} is the raw FIT rate defined in FIT/Mb and depends upon multiple factors including technology node, ambient conditions, and elevation [105]. S_i is the number (in Mb) of sequential elements/latches in component i. SDC_i is the probability of a fault in component i affecting the application output.

Table 1. Comparison of complete and selective ECC. The die cost calculation is based on the equation provided in [55] (Chapter 1.6). The wafer cost, yield, and impurity factors are for 16nm technology node [18, 25, 119].

	No ECC	Full ECC	Selective ECC (SIL 2)	Selective ECC (SIL 3)
Total area (mm^2)	450	502.5	454.5	469.8
Cost/die ($)	59.9	70.0	60.8	63.6

IEC 61508 [57] defines an international safety standard for safety-critical electronic systems. This standard is based on the risks of failure and defines four Safety Integrity Levels (SIL). Each SIL expresses the upper bound on the average frequency of dangerous failures per hour (PFH) [82]. SIL 1 is the least stringent, while SIL 4 is the most stringent. The maximum allowable PFH rate decreases by three orders of magnitude from SIL 1 (10^{-5}) to SIL 4 (10^{-8}). Note that the IEC 61508 standard considers the entire electronic system, not only the MPA.

One approach to make circuits soft error-resilient for certifiable safety is to use hardware error mitigation techniques on storage, which incurs high cost/power/performance overheads. Autonomous vehicles and robotics industries typically have shallow profit margins. For example, the profit margin per unit is under $1000 for several automobile industries [137]. Electronic systems contribute up to 40% to the total cost of a car [24, 100] (at the time of writing). Hence cost-effective solutions to make MPAs more reliable are imperative [23]. The overheads for complete protection of storage structures increase with their size. In such a case, the protection may be sacrificed entirely if the area/power overheads are over budget. In comparison, selective protection is flexible and provides safety with less overhead than complete protection of storage structures by protecting only the most vulnerable data. Table 1 compares the die area and cost for complete and selective ECC for the A3 [77]. As shown in Table 1, selective ECC can reduce the cost by 10% for SIL 3 ($\sim$ 1% increase in the profit margin). While the table compares only the die cost, an increase in the die area has a cumulative effect on the total cost of an electronic system, amplifying the need for selective error mitigation.

2.4 Fault Injection (FI)

In a circuit, a soft error can occur at any location and time. Assuming a single-bit error model, where only one bit is affected by a soft error in the component, exhaustive fault characterization typically requires A $\times$ B FI runs. In this equation, A is the number of fault sites in space, and B is the number of fault sites in time. A is determined by the number of bits, and B is determined by the application's total execution cycles and the number of possible inputs to the application. Unfortunately, this requires very high numbers of FI runs. In MPAs, different combinations of obstacle positions (input to collision detection) add to the number of FI runs, making the space even larger.

Accelerators' fault characterization is typically carried out by statistical FI experiments [19, 33, 48, 49, 73, 74]. Statistical FI performs random sampling in the fault space and allows to tune the number of FI experiments as per the required confidence of the estimated failure rate for a system [73]. Often, statistical FI is performed per component to find the safety-critical components in a circuit. However, the CDM component occupies more than 85% of the area in an MPA and mainly consists of sizable on-chip storage structures to store swept spaces. Hence, selective error mitigation of CDM requires a strategic approach to determine the safety-critical data in the on-chip storage. Further, the number of FI runs needed for statistical FI increases with a decrease in the probability of a bit error resulting in an output error, i.e., the failure probability of a component [73]. Typical CDMs have a low overall failure probability ($\sim$ 0.3%) which increases the number of FI runs required when using uniform random statistical FI (Section 6.1.1). Therefore, it would be beneficial to develop an application-aware FI strategy in which the sites for FI are strategically selected.

Table 2. Summary of frequently used acronyms

CEF	Collision Exposure Factor
MPA	Motion Planning Accelerator
CDM	Collision Detection Module
CDC	Collision Detection Circuits- A CDM consists of multiple CDCs
SIL	Safety Integrity Level (Section 2.3)
SDC-C	Silent Data Corruption leading to False-negative collision detection (Section 3)
CDF	Collision Detection Failure rate per billion hours (Section 4.3)
A1-A4	CDMs studied in this work

3 RELIABILITY METRIC FOR MOTION PLANNING

In this section we explain how erroneous collision-detection outcomes can lead to safety-critical events, then describe *Collision Exposure Factor* (CEF) in detail, and finally discuss how to apply CEF to building resilient MPAs. Table 2 summarizes the frequently used acronyms in this work.

Specialized accelerators, such as a CDM, obtain efficiency by replacing long sequences of software instructions with specialized hardware. Such accelerators may perform computations under the supervision of a command processor via an ISA interface (e.g., Google's TPU [29]), and/or may start computation triggered by an event such as the arrival of a new frame of data in a buffer [21]. While an error originating in the accelerator could potentially propagate to the command processor and thereby result in erroneous operation (e.g., hang or system software crash) this paper focuses on Silent Data Corruption (SDC) within the CDM that can result in the robot colliding with an object.

Given a motion set and the current positions of obstacles, collision detection finds motions that may lead to collisions. A bit-flip can lead to erroneous collision detection. A false-positive outcome can lead to a poor (longer) path or the motion planner's inability to find a path, in which case the robot may get stuck. We perform fault injection experiments to measure the probability of a soft error resulting in a false-positive outcome; this gives an upper bound on the probability of the motion planner's inability to find a path due to soft errors. We find that the motion planning failure rate increases by up to 0.001% for 1 second/motion planning query [99] and 45nm technology node due to soft error induced false positive outcomes, which is significantly lower than the failure rate of probabilistic motion planning algorithms ($5-10\%$ [99]). Further, a simple mechanism such as using a watchdog timer to refresh the corrupted on-chip data can be employed to avoid such cases. We do not focus on false-positive outcomes in this work, as their impact on the motion planning failure rate is very low, and safety standards do not consider false-positives but only safety violations.

We define an SDC-C as an event involving a false-negative result when performing collision detection for a proposed motion and an object (i.e., a colliding motion is misidentified to be collision-free even though sensors detected an object the proposed motion would collide with if chosen during path search). Similar to AVF, SDC-C probability gives a probability of raw error in a memory bit becoming an error in motion planning (i.e., false-negative collision detection). Since, in general, an obstacle might appear anywhere in the environment, a naive (but expensive) approach to estimate SDC-C probability is to simulate many sample environments, each with randomly placed obstacles. Below we consider a more efficient approach suitable, for example, during CDM architecture design.

3.1 Collision Exposure Factor (CEF)

To analyze the errors that can occur in a CDM circuit and their effect on SDC-C probability, we focus on bit changes that can lead to false-negative collision detection. Specifically, we consider the impact of a change in a single bit used to represent a portion of a motion's swept space. As mentioned in Section 2.2, swept spaces of the motions are stored in CDM memory or registers and used to find possible collisions with the obstacles. Each bit in the on-chip storage helps specify the bounds of some motion's swept space. We define the *critical space* of a bit as the region excluded

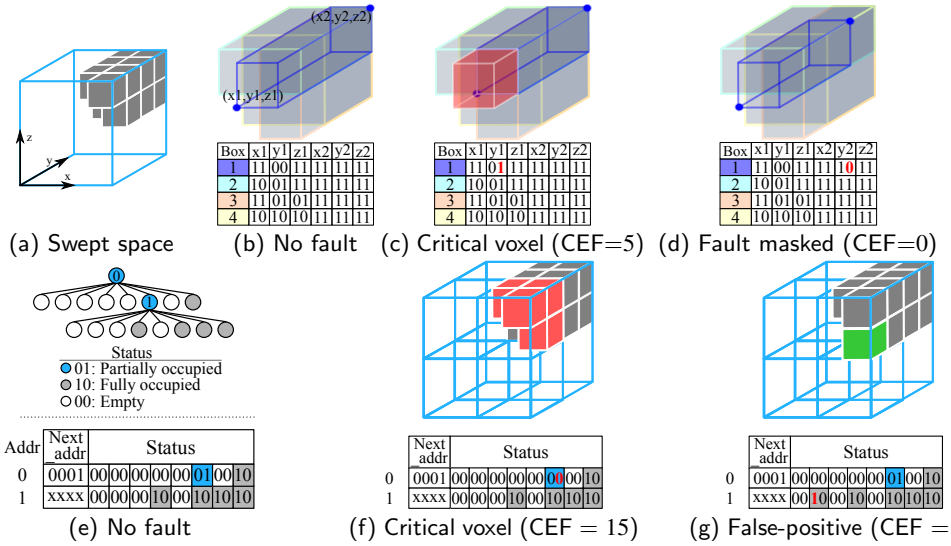


Fig. 5. Analyzing impact of bit flips in a CDM. (a) represents the swept space of a motion stored in CDM, (b)-(d) represent box-based CDM (A2), and (e)-(g) represent octree-based CDM (A3). CEF values are normalized to the surface area of one face of a voxel.

from the swept space if that bit changes value due to a fault. This region represents a part of the swept space that can potentially lead to false-negative collision detection if an obstacle overlaps with the critical space and does not overlap with the remaining swept space. We then define the *collision exposure factor* (CEF) of a bit as the surface area of that bit's critical space exposed to obstacles. If the geometric representation uses voxelized space, the CEF can be normalized to the surface area of one face of a voxel. In our evaluation (Section 6.1), reported CEF values are unitless. The CEF values for different accelerators (A1-A4) are on the same scale as the same normalization factor is used.

Figure 5 illustrates the critical space of specific bits due to single-bit errors for different geometric representations. Figure 5a represents the swept space of a motion. Geometric representation methods convert the swept space to a set of *structures*, such as voxels (used in A1 described in Section 5.2), boxes (used in A2), or octree nodes (used in A3 and A4). Each of these structures is encoded into bits and stored in the on-chip storage of the CDM. For example, a voxel structure is stored using its coordinates, a box structure is stored using the coordinates of the diagonal voxels, and an octree node structure is stored using a custom data structure described below. A bit flip caused by a soft error would modify the space represented by the structure in different ways depending on the location of the bit in the structure and the geometric representation.

Figure 5b shows a box-based representation of the swept space, where four boxes are required to cover the swept space. Since space is divided into four voxels in x, y, and z directions, a total of six ($\log_2 4 \times 3$) bits are used to store a coordinate. The highlighted box is represented with coordinates $(x1, y1, z1)$ and $(x2, y2, z2)$ of the diagonal voxels, before any error is introduced. In Figure 5c, flipping a specific bit causes the value of $y1$ to increase by one voxel. As the voxel highlighted in red is now excluded from the box, and no other box covers the voxel, it becomes part of the bit's critical space. A soft error in this bit results in the exclusion of the critical space from swept space; hence the collision detection circuit fails to detect a collision if an obstacle occupies this critical space, leading to an SDC-C. In contrast, in Figure 5d, a bit flip causes $y2$ to decrease, but this bit flip is masked and does not impact the CDM's output as other boxes cover the voxels exposed.

Figure 5e shows the octree representation of the same swept space. In this representation, the root node and all partially occupied nodes are stored in memory using a tree data structure. Each node in the tree contains two fields: "`status`" and "`next_addr`". The `status` field contains an entry for each octant within the node indicating whether that octant is empty, partially or fully occupied. Only partially occupied octants are further divided. The `next_addr` field contains the start address of an array holding the resulting children nodes. For example, in Figure 5e, the node stored in memory at address 0 contains only one partially occupied octant, and the node containing information about it is stored at address 1. In Figure 5f, a bit flip in the node at address 0 modifies the status of a particular octant to be "fully empty", thereby adding all the voxels in that octant to the critical space. In Figure 5g, a bit flip in the node at address 1 modifies the status of an octant to "fully occupied", resulting in a false-positive rather than a false-negative. In this case, motions that would not lead to a collision may be disallowed. However, this false-positive outcome does not result in potential for a collision (an SDC-C). Therefore, this voxel (highlighted in green) is not a part of the critical space.

For a given erroneous bit, it is more likely that an obstacle occupies its critical space and results in an SDC-C as the exposed surface area of this critical space increases. Hence, intuitively, the SDC-C probability of a bit increases with its CEF value (Section 6.1.2). Note that the CEF of a bit is *independent of the position of the obstacles in the environment*. The CEF captures the probability of obstacles appearing in critical space and decouples the effect of a soft-error and exact position of obstacles on SDC-C probability. While the CEF definition assumes a uniform distribution of obstacles, it can also be extended to nonuniform distributions. For example, the CEF value can be scaled by the estimated probability of obstacles occupying the critical space for a nonuniform distribution of obstacles in the environment.

One approach is to use the volume of the critical space (CS_volume) as a reliability metric. However, the exposed surface area gives a measure of the area of critical space through which obstacles can overlap only with the critical space and not the rest of the swept space, leading to an SDC-C. In contrast, CS_volume does not differentiate between the surface area exposed to obstacles and the surface area touching the remaining swept space. Hence CEF performs better than CS_volume as a reliability metric (Section 6.2). For a given motion set of the robot and accelerator, the CEF of

Algorithm 1 CEF measurement (Phase 1 FI)

Input: Motion_set, Swept_data, Swept_voxels, CDM;
Output: bit_info = {bitID: (CEF, Critical_space)};

```
1:  for Motion ∈ Motion_set do
2:      bits = Swept_data(Motion)
3:      voxels = Swept_voxels(Motion)
4:      for b ∈ bits do
5:          Critical_space = ∅
6:          Collision_vector = FI(CDM, bits, voxels, b)
7:          for (v, collision) ∈ (voxels, Collision_vector) do
8:              if ¬ collision then
9:                  Critical_space = Critical_space ∪ v
10:             end if
11:         end for
12:         CEF = Calculate_CEF (Critical_space, voxels)
13:         bit_info[b]= (CEF, Critical_space)
14:     end for
15: end for
```

all the bits can be calculated using Algorithm 1. The algorithm works by considering each possible motion in turn (Line 1). For a given motion, `Swept_data` returns the storage bits used to represent its swept space (Line 2), and `Swept_voxel` returns the voxels in the swept space of that motion (Line 3). The loop between Line 4 and 14 considers each bit in the `Swept_data`. The CDM takes precomputed swept space (i.e., `bits`) and obstacle occupancy voxels as inputs and performs collision

detection (Figure 4). The storage elements of the CDM are initialized with `bits`. To find the critical space an FI run is performed setting `voxels` as the obstacle occupancy voxels input to the CDM (Line 6). Specifically, a fault is injected into a low-level (e.g., RTL model or microarchitectural simulator) model of the CDM at bit `b` (Line 6). The CDM outputs a `Collision_vector` containing collision detection output for each voxel in `voxels`. To find the critical space, each bit of the `Collision_vector` is checked (Lines 7-11). For an error-free run, the CDM detects collision for all the voxels in the swept space (i.e., `voxels`). However, for an FI run, a voxel `v` is added to the `Critical_space` if the CDM does not detect a collision (Line 8 and 9). `CEF` is then calculated by measuring the exposed surface area of the `Critical_space` (Line 12). Note that the CEF is obtained without the need to consider a potentially unbounded set object placements. Thus the number of FI runs to measure CEF using Algorithm 1 is orders of magnitude lower than exhaustive FI in which for each bit, multiple FI runs need to be performed with a large number of environment scenarios.

As presented above, Algorithm 1 assumes an RTL or architecture model for the CDM. We note that the resulting reliance upon fault injection to determine CEF could be avoided provided an analytical model is available to compute the critical space on Lines 5 to 12. Such a model could be used to analyze CEF without performing FI prior to the development of RTL model or architecture simulators. However, we use FI to measure CEF, so the proposed Algorithm 1 can be used across different accelerators. Algorithm 1 is proposed for CDMs that use a voxelized representation of spatial data and output collision decisions (A1-A4). The majority of the collision detection acceleration approaches for robotics use voxelized representation as it consumes less memory and computational resources compared to polygonal mesh-based representation [27]. However, the concept behind CEF is applicable regardless of the underlying design parameters, and Algorithm 1 to measure the CEF needs can be modified for a different CDM. For example, for a triangle meshes-based representation of swept space, Line 5-12 can be replaced by a geometry-based calculation of the exposed surface.

MPA is a part of complex systems such as autonomous vehicles. Fault Tree Analysis (FTA) is often used to analyze the failure probability of the overall system by constructing a logic tree of the failure events of subsystems [107]. The proposed metric CEF, being close to the application domain, can easily be integrated with the FTA of the overall system. CEF is useful for CDMs that dedicate a significant area to storage structures for spatial data. Earlier work on FPGA-based accelerators [89] uses combinational logic to encode spatial data. Combinational logic is known to be less prone to soft errors compared to sequential elements [36]. However, the definition of CEF applies to erroneous combinational logic gates if needed.

3.2 CEF-aware Error Mitigation

For a fixed budget of area/power overhead, selective protection of the most vulnerable bits provides the optimal reduction in the failure rate. Selective error mitigation in storage structures can be implemented by reliability-aware data placement to partially protected memory. However, this requires efficient ways to identify vulnerable data as the data can be generated/modified at runtime. We find that the CEF of a bit gives a measure of its impact on the SDC-C probability of the CDM (Section 6.1.2). Therefore, we can use information about the CEF of each bit in an input design to selectively apply error mitigation techniques such as ECC, DMR/TMR, and strike suppression. This reduces the cost of providing resilience compared to blanket protection of all the bits.

Different structures such as voxel (A1), box (A2), and octree node (A3 and A4) are stored in the on-chip storage of the CDM depending upon the geometric representation method used. CEF-aware error mitigation is performed by placing structures with higher CEF in the protected storage regions. In our evaluation, the sum of CEFs over all the bits in a structure is used as its CEF. Because CEF of a bit gives a measure of its SDC-C probability, CEF-aware data placement results in a higher reduction of the overall failure rate for a given fraction of protected storage compared to other heuristics

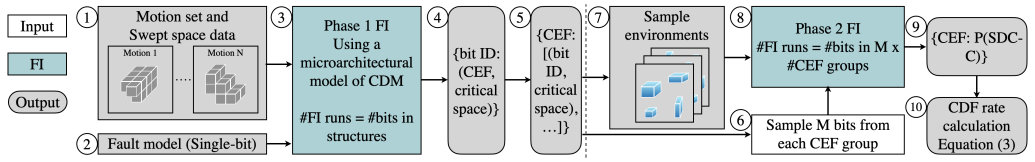


Fig. 6. CEF-aware FI

(Section 6.2). The CEF measurement is done offline or outside of the critical path using Algorithm 1 after motion set generation.

Selective error mitigation can also be used to guide the accelerator design process. At the accelerator design stage, the designer can perform CEF-aware FI (Section 4.2) for a set of target robots, motion sets, and the expected number of obstacles in the environment to find the distribution of CEF, SDC-C probability of bits, and the failure rate of the system without any error mitigation. Based on this information, a designer can determine the required fraction of protected memory for a given failure rate requirement or achievable failure rate for a given area/power budget for error mitigation.

4 CEF-AWARE RELIABILITY CHARACTERIZATION

SDC-C probability and the failure rate measurements are important to find the error mitigation requirements for certifiable safety. Exhaustive FI to find the SDC-C probability of all the bits takes a long time (Section 2.4). In this section, we first explain the fault model and demonstrate how to use the CEF to enable a hierarchical fault analysis methodology that reduces the number of FI runs.

4.1 Fault Model

In the CDMs studied in this work, on-chip SRAM/latches that store the swept space of motions account for 97% of the on-chip sequential elements (based on our synthesis results). Our fault model assumes transient single-bit faults in this SRAM/latches. The faults are uniformly distributed in time and space. Only 3% of on-chip sequential elements are used for meta-data (e.g., counters, address register, and finite state machine). These microarchitecture-specific registers are not subjected to CEF in our experiments as they can be characterized and protected with low overheads using latch hardening techniques such as strike suppression or redundant node [111].

4.2 CEF-aware FI

We propose a two-phase CEF-aware FI (shown in Figure 6), a technique to speed up FI for MPAs.

Phase 1: CEF measurement (Environment independent). For a given motion set (1) and fault model (2), Phase 1 performs microarchitecture- or RTL-level FI (3) to find the CEF and critical space of each bit in the swept space data. Algorithm 1 is used for this phase. The number of FI runs for this phase is limited to the number of bits in swept space data of all the motions in the robot's motion set. The CEF and critical space of all bits are stored to be used in the next phase (4). Note that environmental information, such as the position of obstacles or the robot, is not needed for this phase, as CEF does not depend upon the environment.

Phase 2: SDC-C probability measurement. A bit flip might lead to an SDC-C depending upon the position of obstacles as shown in Section 3. Since for a dynamic environment, obstacles can appear anywhere in the space, a large number of FI runs with random environmental scenarios are required to measure the SDC-C probability of a bit with statistical significance. In our experiments, the representative environment scenarios are generated using apriori information about the environment, such as the distribution and average number/size of the obstacles. In Phase 2, we use the CEF

information gathered in Phase 1 to speed up the fault analysis. As mentioned, there is a strong (positive) correlation between the CEF and SDC-C probability (Section 6.1.2). Thus, we can speed up the FI experiments many-fold by performing FI for only a subset of bits with a given CEF value to measure the approximate SDC-C probability for all bits with the same CEF. The CEF information gathered in Phase 1 is used to group bits with equal CEF values together ⑤. Then, for each CEF value, M bits are selected at random ⑥. Finally, using multiple sample environment scenarios ⑦ and FI simulations ⑧, the SDC-C probability for each CEF value is measured ⑨. Note that M is a tunable parameter in the above heuristic. We use the analytical model in Leveugle et al. [73] to determine the value of M to measure SDC-C probability with the required confidence level and error margin. The CEF-aware sampling, while being faster than exhaustive FI, may introduce inaccuracies due to the approximations it makes. However, the accuracy can be increased by increasing the value of M - we evaluate this trade-off in Section 6.1.1.

To speed up the FI simulation, we further exploit the fault propagation in the studied CDMs. All the accelerators studied in this work use a geometric representation that divides the space into voxels. For a given motion and environment scenario, a non-empty subset of swept space and obstacle occupancy voxels signify potential collision. Thus the effect of a bit flip can be captured by storing the erroneous swept space voxels of the corresponding motion, instead of performing slow microarchitectural and RTL simulations for multiple environmental scenarios. Simple set operations can be used to find the SDC-C probability. For a given bit flip, an FI simulation is used to find the erroneous swept space (i.e., set of voxels) using Algorithm 1. Line 3 of the Algorithm 1 is modified to use the set of all environment voxels as `voxels`. Similarly, Lines 8 and 9 are modified to measure the erroneous swept space (i.e., colliding voxels). For each environment scenario, the subsets of obstacle occupancy voxels and erroneous and error-free swept space are measured. If the erroneous subset is empty, but the error-free subset is non-empty, the bit flip will result in an SDC-C. The same strategy is used for exhaustive, statistical, and CEF-aware FI. While we focus on FI to measure CEF and SDC-C probability, we believe that an analytical model can be used to replace the multiple runs and find the relation between critical space and SDC-C probability. We defer this to future work.

Equation 2 is used to calculate the SDC-C probability of a CDM. Here bits_{CDM} is the set of all bits stored in the CDM's on-chip memory. $\text{CEF}(x)$ is the CEF value of bit x, and $\text{P}(\text{SDC-C}_{\text{CEF}(x)})$ is the SDC-C probability for the CEF value $\text{CEF}(x)$.

$$\text{P}(\text{SDC-C}_{\text{CDM}}) \approx \frac{1}{|\text{bits}_{\text{CDM}}|} \times \left[\sum_{x \in \text{bits}_{\text{CDM}}} \text{P}(\text{SDC-C}_{\text{CEF}(x)}) \right] \tag{2}$$

Table 3 compares the different phases of CEF-aware FI with exhaustive and uniform random statistical FI approaches for CDM fault characterization and error mitigation. We also use CEF and CEF-aware FI to analyze and compare the effects of microarchitectural design parameters of the CDM, and to derive the principles of *resilience-aware* MPA design (Section 7).

4.3 Collision Detection Failure (CDF) Rate Calculation

As mentioned in Section 2.3, safety standards provide an upper bound on the frequency of dangerous failures (e.g., collision in motion planning). We combine Equation (1) and Equation (2) to measure the Collision Detection Failure rate per billion hours (CDF) for CDMs as below:

$$\text{CDF}_{\text{CDM}} = |\text{bits}_{\text{CDM}}| \times \text{P}(\text{SDC-C}_{\text{CDM}}) \times \text{FIT}_{\text{Raw}} \times \alpha \tag{3}$$

Here, the parameter α is set to $N/2$ for CDMs with inter-query data reuse, else it is set to 1, as explained below. In the MPAs we study, the on-chip data is typically reused across multiple executions of the same application [45, 89, 114, 139] to reduce the DRAM-bandwidth requirement

Table 3. Comparison of different fault injection (FI) approaches.

	Exhaustive FI	Uniform random statistical FI	CEF-aware FI	
			Phase 1 (CEF measurement)	Phase 1 + Phase 2 (SDC-C measurement)
Description	Performs FI on all combinations of bits and environment scenarios	Samples a few combinations of bits and environment scenarios to perform FI	Performs environment-agnostic FI to measure CEF of all the bits (Algorithm 1)	Samples a few combinations of environmental scenarios and bits for each CEF group and performs FI
Number of FI runs	10^{10}	7×10^6	10^6	7×10^6
FI time	24,000 hours	2-4 hours	1-2 hours	2-4 hours
Measures overall failure rate	Y	Y	N	Y(2.5% error)
Finds vulnerable bits for selective error mitigation	Y	N	Y(Approx.)	Y(Approx.)
Measures SDC probability of different bits	Y	N	N	Y(Approx.)

and data movement. In such a case, a bit-flip due to a soft error will persist in the buffer and affect multiple executions, until the buffers are reloaded, and the overall CDF rate of the system increases due to data reuse. This is similar to bit-flips in configuration memory of FPGA [94], where a bit-flip due to single-event upset persists and affects the application output until the configuration memory is refreshed. A1, A2, $A3_{scaled}$, and A4 accelerators reuse on-chip data across N (where $N >> 1$) collision detection queries. A soft error can occur in any cycle during a collision detection query; however, the bit will remain erroneous from the start (i.e., cycle 0) for subsequent collision detection queries until the on-chip data is refreshed. Thus the average value of the number of collision detection queries for which a bit will be erroneous from starting is approximately $N/2$, assuming all collision queries take the same time. Hence, to measure the CDF rate for such CDMs, faults are injected at cycle 0 in Phase 2, and the measured failure rate is scaled by $\alpha = N/2$ to account for data reuse. Whereas in A3, the on-chip data is refreshed for every collision detection query. For CDMs without inter-query data reuse, α is set to 1, and the fault sites are uniformly distributed in time in Phase 2.

5 EXPERIMENTAL METHODOLOGY

5.1 Experimental Setup

Table 4 summarizes the robots used in our experiments. These robotic arms are representative of widely used industrial robots [87, 104], and are also included in larger humanoid robots [4]. Because we did not have access to the real robots, we use the Klampt [51] software simulator to simulate the robot's movements and the environment- this has also been used in prior work [50, 91, 114].

The environment size for a robot is determined by its reach, and the environment is discretized into a grid of $32 \times 32 \times 32$ voxels. For each robot, we generate a motion set with 16,384 poses and 32,768 motions, using Leven and Hutchinson's strategy [72], similar to prior work [91]. We use uniform motion sets in all our experiments, i.e., poses are distributed uniformly in the C-space. The uniform distribution of poses is assumed in the absence of any prior information about the robot's task and fixed obstacles in the environment. However, this information can be used to build a motion set with a nonuniform distribution of poses, where more poses are concentrated in the region of interest [129]. We also evaluate CEF on a nonuniform motion set in Section 6.1.2.

Table 4. Robots used for fault characterization.

Robot	Degrees of freedom	Reach in one direction	Mechanical power (W)
Kinova Jaco2 [69]	7	90 cm	25
Programmable Universal Machine for Assembly (PUMA) 761 [138]	6	150 cm	30
AL5D [79]	4	27 cm	12

Table 5. Accelerators studied. We list power and area for each accelerator from the paper and report suitable error mitigation and accelerator area overhead to protect on-chip memory. We use information gathered from our synthesis of RTL models about the fraction of CDM area and power consumed by the on-chip memory to calculate the overall CDM area and power overheads. ECC overheads do not include the area /power of decoder and encoder circuits.

Accelerator	Representation	Data reuse	#CDCs	Power (W)	Area (mm^2)	Storage elements	ECC area/power overhead	TMR area/power overhead
A1 [89, 114]	Voxel	Yes	32,768	N/A	N/A	Registers	20/16%	125/100%
A2 [89, 114]	Box	Yes	32,768	35	400	Registers	20/16%	125/100%
A3 [77]	Octree	No	128	0.47	1.7	SRAM	12/9%	100/75%
$A3_{scaled}$ [77]	Octree	Yes	32,768	121	450	SRAM	12/9%	100/75%
A4 [139]	Flattened Octree	Yes	32,768	20	-	SRAM/DRAM	12/12%	200/200%

To measure the SDC-C probability, we perform FI with a set of 10,000 random environment scenarios. Each sample environment contains $3-12$ cuboid-shaped randomly placed obstacles, and the length/height/width of each obstacle is $5-20$cm, which is consistent with other work on motion planning and collision detection [77, 91]. We generate four sets of environments to study the effect of the number of obstacles on the SDC-C probability (Section 6.1.2). We use the label Dx ($x \in [1,2,3,4]$) to represent a set of 10,000 environment scenarios, where obstacles occupy an average x% volume of the environment. The average number of obstacles increases with the value x.

5.2 Collision Detection Modules (CDM)

Table 5 summarizes the four accelerators studied in this work and overall CDM area/power overheads for complete protection of on-chip storage using ECC and TMR error mitigation techniques. *These constitute the only published work on ASIC-based programmable, real-time accelerators for probabilistic roadmaps-based motion planning and collision detection, to the best of our knowledge.*

We built microarchitectural simulators and RTL models of these CDMs ourselves as there was no existing simulator. We use Verilog to build the RTL models and implement the combinational logic, control logic, and on-chip memory as described in the prior work on the accelerators we studied in this work [77, 89, 114, 139]. CDMs studied in this work consist of multiple (128-32768) CDUs. Hence, though the overall area of a CDM is high, the design of the basic building block, CDU, is simple and can be implemented accurately ($100-200$ lines of Verilog code). Each CDU consists of storage elements for the swept space data and a collision detection circuit (CDC). Figure 7 represents the architecture of CDCs for different accelerators studied in this work. Further, we implement testbenches to verify the functionality and timing of the RTL models. We use a sample motion set, and perform collision detection for 10,000 environmental scenarios for functional verification. We further measure the timing of collision detection operation in terms of the number of cycles and validate with the reported data in the prior work [77, 89, 114, 139].

We synthesized our RTL models using the Synopsys Design Compiler [1] and the OpenRAM Memory Compiler [44] to estimate the area and power of storage elements in CDMs at 45nm technology (FreePDK45 design library [118]). Because we are interested in the relative area and power consumption of different storage elements and combinational circuits, the technology node's choice should not significantly impact the results. We build the microarchitectural simulators with Python. We model all sequential elements accurately in our microarchitectural model, and verify the collision detection output of these models for 10,000 environmental scenarios.

A1 (Base accelerator): This architecture was proposed by Murray et al. [89], and is based on the earlier proposed accelerator for FPGAs [91]. A motion's swept space is stored in registers using the 3D Cartesian coordinates of each voxel in the swept space. Figure 8a and 8b show how a swept space

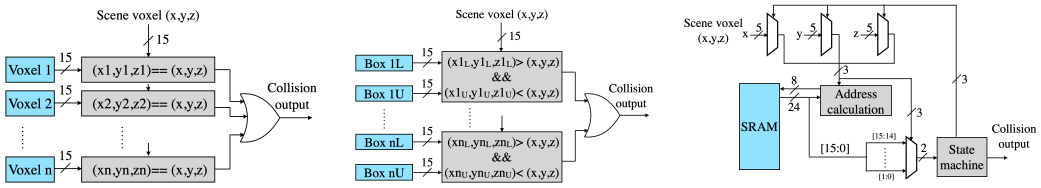


Fig. 7. CDU architecture block diagram for different CDMs.

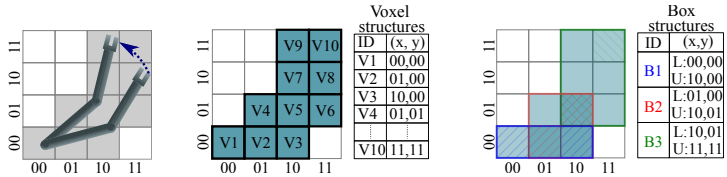


Fig. 8. Voxelization and voxel/box based representation example in 2D. (a) Represents a robot and its motion in 2D. Shaded voxels represent the voxelized swept space. (b) Swept space is stored using coordinates (x,y) of individual voxels. (c) Contiguous voxels are combined into boxes and stored using diagonal voxels' coordinates (striped voxels). L and U represent lower and upper diagonal voxels; darker regions represent overlapping of multiple boxes.

is converted to voxels. The collision detection circuit compares the obstacle occupancy voxels with each voxel in the swept space to find if the motion is in collision with obstacles (Figure 7a).

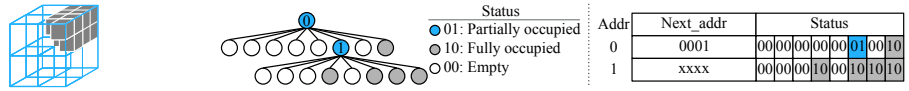
A2 (Spatial locality-aware accelerator): This architecture, proposed by Murray et al. [89] and Sorin et al. [114], is an optimization of A1. There is a significant degree of spatial locality in the voxels in a swept space; hence, contiguous voxels are merged into a larger box. A box can be represented by the coordinates of two diagonal voxels. Figure 8c gives an example. The CDC checks if an obstacle occupancy voxel is inside any of these boxes for collision detection (Figure 7b).

In A1 and A2, all on-chip registers are read in parallel by the collision detection logic instead of using a memory array with limited read ports. This design choice provides significant speedup. Thus all storage elements in A1 and A2 are treated as individual registers that can be protected using a latch protection technique (e.g., DMR or TMR).

A3 (Octree-based accelerator): This architecture was proposed by Lian et al. (2018) [77], and uses the octree structure to store the motion's swept space (shown in Figure 9 and explained in Section 3). Collision detection is performed by traversing the tree to find if obstacle occupancy voxels overlap with the swept space. Figure 7c represents the architecture of CDC for A3.

The proposed design of A3 uses 128 CDCs, where motions in the motion set are processed for collision detection in batches. Hence, there is no inter-query on-chip data reuse, which results in significant DRAM bandwidth requirement [139]. For comparison, we also study a scaled-up version of A3, called $A3_{scaled}$, where the number of CDCs is equal to A1 and A2 (32,768), and on-chip data is reused across multiple collision queries, reducing DRAM memory traffic. While the SDC-C probabilities for both A3 and $A3_{scaled}$ are equal, their failure rates (CDF) are different due to differences in the sizes of their components and the value of N (Equation 3).

A4 (Flattened octree-based accelerator): This architecture was proposed by Yang et al. [139], and proposes processing-in-memory for collision detection with a flattened octree-based representation of the swept space. In the flattened octree-based representation, multiple levels of the trees can be flattened in a single level. For example, if all the levels of a 5-level tree are flattened, the resultant tree consists of a single root node with 32,768 children nodes, where each child node is 1 or



(a) (b)

Fig. 9. Octree representation (figure reproduced from [77]). (a) represents the spatial division of the swept space, and (b) represents the corresponding Octree structure (A3).

0 specifying occupancy of a single node. Such representation consumes more storage but facilitates efficient processing-in-memory, reducing data-movement overhead significantly.

5.3 Fault Injection (FI)

We use microarchitectural simulators for FI, as RTL-level FI was very slow. We use Dell EMC R440 CPU nodes. However, to validate the FI accuracy of our microarchitectural simulators, we perform 10,000 FI runs (uniformly distributed in space and time) on the microarchitectural and RTL models. For RTL-level FI, we use Cadence Incisive Functional Safety Simulator [15]. For microarchitectural FI, a fault is injected, and collision detection is performed. Figure 10 shows that the CEF values measured with microarchitectural FI and RTL-level FI have a 100% correlation accelerators A1-A4.

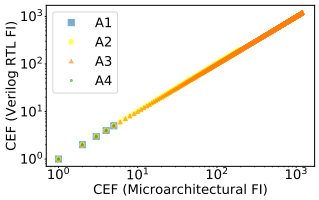


Fig. 10. CEF values measured by microarchitectural FI and Verilog RTL-level FI for different CDMs.

For CEF-aware FI, we determine the value of M (number of samples) per CEF group required to measure the SDC-C probability with 95% confidence level and 2.5% error margin [73]. The SDC-C probability and population size of the highest CEF group are used to determine the value of M, as these bits contribute the most to the overall SDC-C probability. In our experiments, the value of M is 2×10^6, 2×10^5, 15×10^3, and 2×10^6 for A1, A2, A3 and A4 respectively. We represent soft errors as single-bit flips in hardware registers, which is consistent with most other papers studying the effects of soft errors [8, 20, 32, 132]. While we focus on single-bit flips, Equation (2) can be extended to accommodate a multi-bit fault model. We defer this to future work.

5.4 CDF Rate Calculation

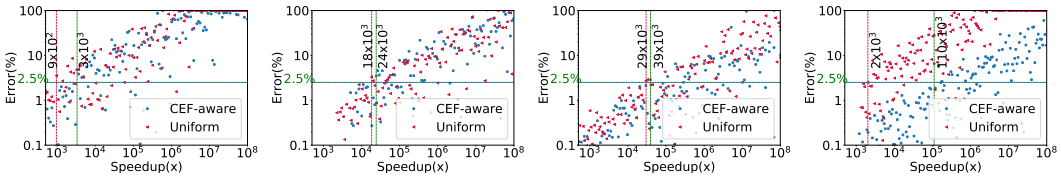
To calculate the CDF rate of the accelerators, we use Equation (3), where N is the expected number of executions of the application before the bits are reloaded in on-chip storage, and its value depends on the accelerator's architecture and deployment. Reloading of data can be overlapped with collision detection to hide the DRAM accesses latency. We set the value of $N = 3600$ for the accelerators with inter-query data reuse (A1, A2, $A3_{scaled}$, and A4), as the power overhead of DRAM accesses for refreshing data after 3600 collision detection queries is within 1%. We use $FIT_{Raw} = 177$ FIT/Mb (for 45nm CMOS [6]) in Equation (3). Note that while the choice of N and FIT_{Raw} affect the absolute values of the CDF rate, they affect neither our fault characterization nor CEF-aware error mitigation.

6 EVALUATION

In this section we present our results for fault characterization (Section 6.1) and error mitigation (Section 6.2) using CEF.

6.1 Fault Characterization

6.1.1 CEF-aware FI. In Section 4, we propose two-phase CEF-aware FI to reduce the number of FI runs. In CEF-aware FI, instead of performing FI for all the bits for multiple environment scenarios, we sample a subset from a group of bits with the same CEF value. This sampling introduces inaccuracies



(a) A1 (Voxel-based) (b) A2 (Box-based) (c) A3 (Octree-based) (d) A4 (Flattened octree-based)

Fig. 11. Error versus speedup for different FI approaches.

in the measured SDC-C probability and CDF rates compared to exhaustive FI. Figure 11 shows the speedup versus error of the calculated CDF rate for CEF-aware FI and uniform statistical FI. The speedup is the ratio of the number of exhaustive FI runs to that of CEF-aware FI or uniform FI.

As can be seen from the figure, on average, CEF-aware FI achieves $23,000\times$ speedup over exhaustive FI (geometric mean) with 2.5% error margin. The vertical line represents the speedup for error less than 2.5%. The error margin can be reduced further at the cost of more FI trials (Section 4). Uniform statistical FI exhibits a similar speedup as CEF-aware FI over exhaustive FI. Note however that uniform statistical FI cannot be used to find vulnerable bits for selective error mitigation (Table 3). The speedup of CEF-aware FI over exhaustive FI is due to two reasons: (1) a significant fraction of bits have CEF equal to 0, and hence have very low or 0 SDC-C probability. CEF-aware FI ignores these bits for FI, (2) Only a few bits have high CEF and SDC-C probability, and hence contribute the most to overall SDC-C probability. CEF-aware FI segregates such bits and requires fewer samples to measure SDC-C. Phase 1 of CEF-aware FI consumes 1, 1, 2, and 1 CPU hours for A1, A2, A3, and A4, respectively. Phase 2 of CEF-aware FI consumes less than 2 CPU hours for all accelerators. In contrast, as per our experiments, exhaustive FI takes 24,000, 18,000, 22,000, and 20,000 CPU hours for A1, A2, A3, and A4, respectively.

6.1.2 Evaluation of reliability metric. As discussed in Section 4, we propose the CEF as a reliability metric for the MPA's bits. We study the relationship between the CEF and the SDC-C probability of different bits to demonstrate the validity of CEF metric. The SDC-C probability of entire CDM is calculated using SDC-C probability of different CEF groups in Equation (2). We use the benchmarks [D1-D4] (Section 5.1). Figure 12 shows the SDC-C probability of bits with different CEF values for different accelerators, robots, and benchmarks. The CEF values are on the same scale for different accelerators for a given robot. A3 has a higher range of CEF values than other accelerators, which is also reflected in higher SDC-C values. Note that for A3, there is dispersion in SDC-C probability for bits with high CEF values. This occurs as a fault in a bit in the `next_addr` field adds a large number of voxels in the represented swept space (green voxels in Figure 5g), which leads to positive collision detection between obstacles and erroneous expanded swept space, and reduces SDC-C probability for some bits with high CEF value. For all the benchmarks, the SDC-C probability of a bit increases as its CEF increases, except for a few bits in A3. Note that as the obstacle occupancy density increases, the SDC-C probability also increases, as there are higher chances that the soft error in a CDM will result in a collision. Thus, even though the relation between SDC-C probability and CEF is not linear, a monotonically non-decreasing relation demonstrates the validity of the CEF as a reliability factor. *The SDC-C probability of bits is thus strongly positively correlated with their CEF values across benchmarks, which signifies that the CEF can be used as a reliability metric for MPAs.*

Nonuniform motion set. We also validate the use of CEF as a reliability metric for a nonuniform motion set. In such a motion set, the motions are nonuniformly sampled in space. Figure 13 shows SDC-C probability versus CEF of bits for nonuniform motion sets. As shown in the figure, there is

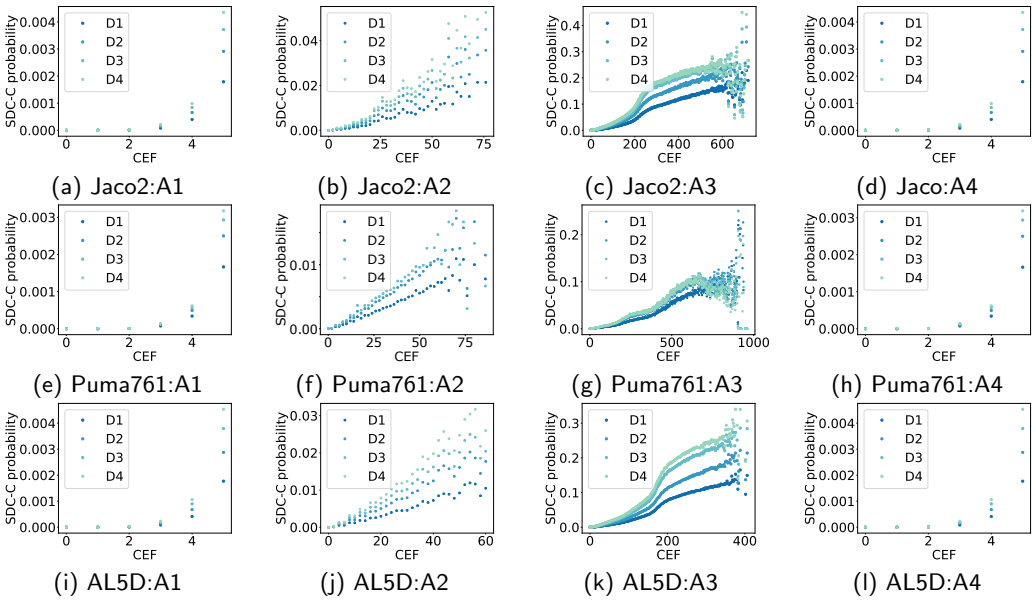


Fig. 12. SDC-C probability versus CEF of accelerator A1-A4 for Jaco2, AL5D, and Puma761 robot on benchmarks D1-D4. Note that different plots have significantly varying ranges for the vertical axes.

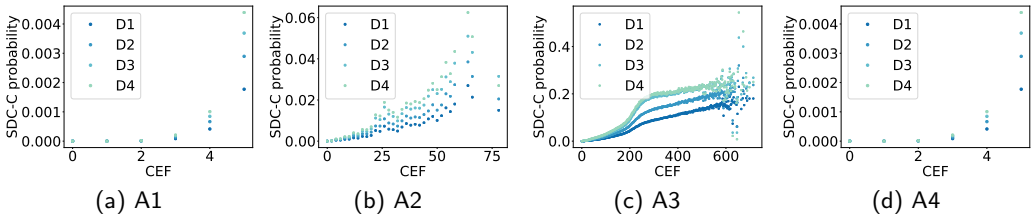


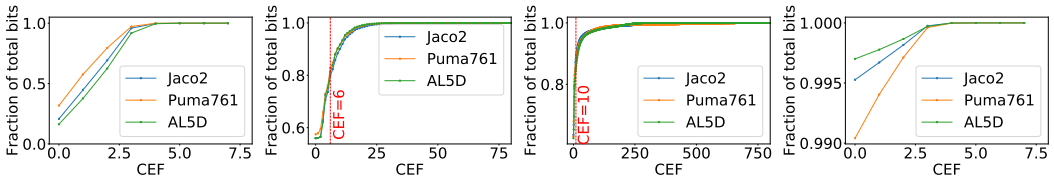
Fig. 13. SDC-C probability versus CEF of accelerator A1, A2, A3, and A4 for Jaco2 for a nonuniform motion set. Note that different plots have significantly differing ranges for the vertical axes.

strong correlation between the SDC-C probability and CEF of bits, showing the validity of CEF for nonuniform motion sets.

6.1.3 Characterization of CEF. To further understand the contribution of individual bits to the overall SDC-C probability, we group the bits according to their CEFs, and plot the cumulative distribution of bits in the CDM. Figure 14 shows that for all four CDMs, there is a high degree of asymmetry in the distribution of bits according to the CEF. For A1 and A4, 20% and 99% of the total bits have CEF equal to 0, respectively, and do not need to be protected. On the other hand, for A2, only 20% of the total bits have CEF greater than 6 and significantly contribute to the overall SDC-C probability. Thus, protecting only 20% bits can reduce the CDF rate by 60%, as we show in Section 6.2. Similarly, in A3, only 15% of the total bits have a CEF greater than 10. We further examine the CEF distribution asymmetry for each accelerator.

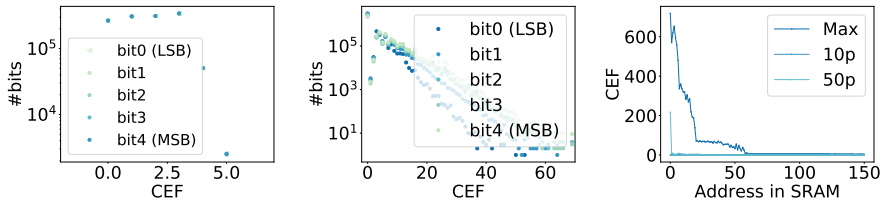
A1: In A1, each structure/variable stores a voxel using its coordinates to represent a part of the swept space. A soft error in any bit of the coordinate will result in misrepresentation of that voxel, and so bit position does not affect CEF range (Figure 15a).

A2: The effect of faults in different bits of coordinates is shown in Figure 15b. The CEF range of bits decreases from the most significant bit (MSB) to the least significant bit (LSB), as a fault in the



(a) A1 (Voxel-based) (b) A2 (Box-based) (c) A3 (Octree-based) (d) A4 (Flattened octree-based)

Fig. 14. Cumulative distribution of CEF of all bits. The vertical axis starts from 0.99 for A4 as more than 99% of the bits have CEF equal to 0.



(a) A1 (b) A2 (c) A3

Fig. 15. CEF characterization of bit position/SRAM address.

MSB is likely to result in a higher change in the box size represented by a pair of coordinates than LSB. Thus, the number of bits with high CEF values increases from LSB to MSB. For example, the ratio between the number of MSBs ($bit4$) and the number of LSBs ($bit0$) with CEF $= 20$ is 1000.

A3: We calculate the range of CEF for bits for different SRAM addresses, as shown in Figure 15c. All nodes in the octree for a motion's swept space are stored in the contiguous address space of the SRAM; the nodes closer to the octree's root node that divide the 3D space at a coarser granularity are stored in the lower address range of SRAM. As can be seen, the CEFs of bits with lower SRAM addresses are much higher than those with higher SRAM addresses. For example, the average CEF of bits in SRAM address 0 is 9× that of the average CEF of bits in SRAM address 20.

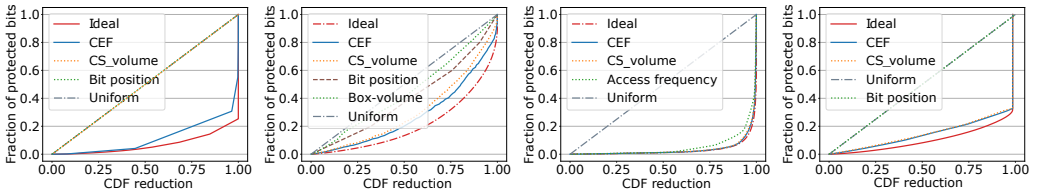
A4: A flattened-octree consists of a node for each voxel in the environment. The swept space of a motion in the motion set is a small fraction of the entire environment. Thus, most of the nodes in the flattened octree are empty, and only a small fraction of nodes are occupied. An error in an empty node representation results in false-positive and hence has CEF equal to zero. Due to this, more than 99% of the bits have CEF equal to 0 for A4.

In summary, for all four accelerators, the distribution of the bits as per CEF is highly asymmetric. Hence, the CEF metric facilitates finding the most SDC-C-prone bits in the circuit.

6.2 Error Mitigation Techniques

As described in Section 3.2, we propose a CEF-aware selective error mitigation technique. We first evaluate the CEF-aware selection of structures for reliability-aware data placement at deployment time. We further evaluate the area savings achieved by the proposed techniques for accelerators A1, A2, $A3_{scaled}$, and A4 for different SIL targets at design time. Note that A3's CDF rate (0.1) is well below the highest SIL requirement, and hence it does not need error mitigation.

6.2.1 CEF-aware selective error mitigation. Figure 16 compares the CDF rate reduction achieved by different selection criteria for the same fraction of protected memory. We assume that protecting a bit reduces its SDC-C probability to 0 as our aim is to compare different heuristics for selecting the bits to be protected, which is independent of the underlying error mitigation technique. We also



(a) A1 (b) A2 (c) A3$_{scaled}$ (d) A4

Fig. 16. Comparison of different selection criteria for selective error mitigation in A1, A2, A3$_{scaled}$, and A4. (a)-(d) compare the fraction of protected bits versus CDF rate reduction for different CDMs. Overall, CEF-aware selection results in the highest CDF rate reduction for the same amount of protected bits. There is a significant overlap between `Bit position`, `Uniform`, and `CS_Volume` for A1. There is a significant overlap between `Bit position` and `Uniform`, and `CS_Volume` and `CEF` for A4.

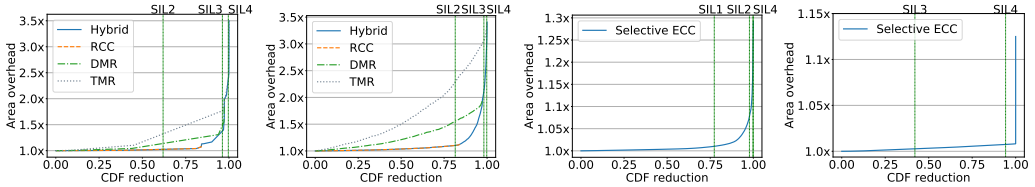
compare with another intuitive approach to define the reliability metric that uses the volume of the critical space (CS_volume) instead of the exposed surface area. We further discuss the results for A1, A2, A3$_{scaled}$, and A4.

A1: We compare our approach with ideal, uniform, bit position, and CS_volume-aware error mitigation. Bit position does not require FI and can be used as a proxy for the vulnerability factor, where the vulnerability of the bits decreases from the MSB to the LSB. Bit position has been previously proposed for selective error mitigation in DNN accelerators [74]. Figure 16a shows the fraction of protected bits versus the CDF rate reduction curve for CEF-aware selection and the other heuristics for A1. The CEF-aware selection of bits results in 52.35× reduction in the CDF rate on average (geometric mean) for the same amount of protected bits compared to CS_volume-aware, bit position-aware, and uniform selection. The CDF rate reduction for CEF-aware error mitigation is only 1.60× lower than the ideal CDF rate reduction that can be achieved by exhaustive FI.

A2: We compare CEF with ideal, uniform, CS_volume, bit position, and box-volume. For A2, a structure represents a box; the volume of the box can be used as the vulnerability factor of the structure. *Box-volume* is an application-specific heuristic that does not require fault characterization. Figure 16b shows the fraction of protected bits versus the CDF rate reduction for different selection heuristics. The CEF-aware selection of bits results in average 1.25×, 1.76×, 2.10×, and 2.46× lower CDF rate than CS_volume, bit position, box volume, and uniform selection for the same amount of protected bits. The CDF rate reduction for CEF-aware error mitigation is only 1.66× lower than the ideal CDF rate reduction achieved by exhaustive FI.

Intuitively, one may expect bit position to provide higher benefits than CEF, as typically MSBs are more critical. However, we find that CEF provides a higher reduction in the CDF rate than the bit position, due to two reasons. First, though the bit position captures the failure probability of bits within a structure, it does not capture the relative failure probabilities across different structures. For example, in A2, different boxes cover different numbers of voxels x in the swept space that is not covered by other boxes; these voxels contribute to the critical space. The CEF increases as the x value increases, and so structures with higher x have higher failure probabilities, which is not captured by bit position. Second, the bit position does not consider whether the error will lead to a false-negative or a false-positive. Similarly, box-volume does not necessarily capture the number of critical voxels x. In contrast, the CEF captures the relative failure impact of all the structures, and hence provides a higher CDF rate reduction.

A3$_{scaled}$: Figure 16c shows the fraction of protected bits versus the CDF rate reduction for different selection criteria in A3$_{scaled}$. We compare CEF with two heuristics: uniform and access-frequency-based selection. Access-frequency-based selection has been proposed for embedded applications [68, 83]. We find that the CEF-aware selection of bits results in an average 1.07× and



(a) A1 (b) A2 (c) A3$_{scaled}$ (d) A4

Fig. 17. Area overhead versus CDF rate reduction for selective error mitigation in A1, A2, A3$_{scaled}$, and A4. (a)-(d) compare the area overhead for different CDF reduction values for all CDMs. Table 7 summarizes the area/power overhead for CEF-aware error mitigation for different safety levels and blanket protection of the entire on-chip memory.

1.90× lower CDF rate for the same amount of protected bits compared to CS_volume and access-frequency. This is because the access-frequency-based heuristic captures the failure probability of structures within a single motion, but not the relative failure probabilities across different motions. Further, CEF-aware selection achieves 18.89× and 0.79× reduction in CDF rate than uniform and ideal error mitigation, respectively.

A4: Figure 16d compares different selection criteria for A4. We find that the CEF-aware selection of bits results in an average 1.02× reduction in CDF rate for the same amount of protected bits compared to CS_volume. In the proposed accelerator, the access-frequency for all the bits of a flattened octree is the same, and hence uniform and access-frequency-based selection given the same reduction in the CDF rate. Further, CEF-aware selection achieves 9.41× and 0.83× reduction in CDF rate than uniform/access-frequency and ideal error mitigation respectively.

6.2.2 Area savings using CEF-aware error mitigation. Further, we measure the area/power overheads to achieve different SILs using the CEF-aware error mitigation approach. In A1 and A2, on-chip storage elements consist of 15-bit registers that are accessed independently. We assume the use of latch hardening technique summarized in Table 6 in Sullivan et al. [120] to measure the overheads for A1 and A2. . Latch hardening is more suitable than ECC for error mitigation in registers that are accessed in parallel. In contrast, we use ECC (SEC-DED code) for A3$_{scaled}$ and A4 as these accelerators use SRAM for on-chip storage or DRAM (Table 5). For A3$_{scaled}$, each entry in SRAM consists of 24 bits. We assume an overhead of 7 bits (30%) to store the ECC bits. For A4, each entry in DRAM consists of 64 bits, with 8 bits overhead for ECC bits. We ignore the area overhead of the error detection/correction circuits themselves. Note that other approaches for ECC [41] can also be combined with CEF-aware selection. Figure 17 shows the storage area overhead versus CDF rate reduction for CEF-aware error mitigation for A1, A2, A3$_{scaled}$, and A4. SIL 1-SIL 4 markers are shown for CDF rate achievement for the Jaco2 robot and different safety standard levels. Table 7 compares the area/power overheads for different safety levels for all accelerators with the blanket protection of CDMs. CEF-aware selection results in significantly lower overhead for all levels.

6.2.3 CEF-aware error mitigation implementation. As mentioned in Section 3.2, CEF-aware selective error mitigation can be implemented by reliability-aware data placement in partially protected memory. We discuss possible implementations of selective error mitigation for CDMs studied in this work.

Table 6. Area overhead and CDF rate reduction for different latch hardening techniques.

Latch type	Area Overhead	failure rate reduction
Strike Suppression (RCC [111])	1.15x	6.3x
Redundant Node (SEUT [111])	2x	37x
Triple Module Redundancy (TMR [80])	3.5x	1,000,000x

Table 7. Area/Power overhead for blanket and CEF-aware error mitigation for different SILs. In A1, A2, $A3_{scaled}$, and A4, on-chip storage elements consume $\sim 50\%/\sim 30\%$, $\sim 50\%/\sim 30\%$, $\sim 40\%/\sim 30\%$, and $\sim 98\%/\sim 98\%$ of the total area/power of the CDM, respectively.

Accel.	Blanket Error Mitigation	CEF-aware Error Mitigation			
		SIL1	SIL2	SIL3	SIL4
A1	125%/75%	-	1.2%/0.7%	18.8%/11.3%	68.9%/41.4%
A2	125%/75%	-	5.1%/3.1%	55.7%/33.4%	101.9%/61.1%
$A3_{scaled}$	12%/9%	0.4%/0.3%	3.1%/2.3%	8.8%/6.6%	11.4%/8.5%
A4	12%/12%	-	-	0.3%/0.3	0.7%/0.7%

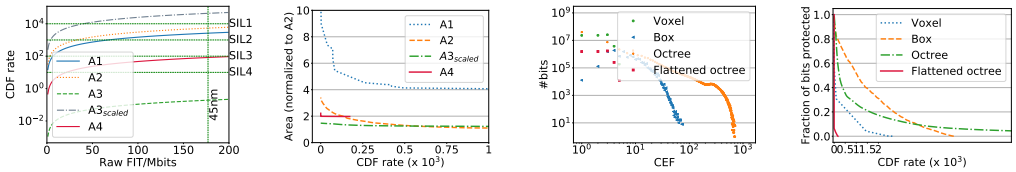


Fig. 18. Comparison of different CDM architectures geometric representations. (a) compares the CDF rate for different Raw FIT values and technology nodes, (b) compares the CDM area versus CDF rate, (c) compares the CEF distribution, and (d) compares the fraction of protected bits versus CDF rate for different accelerators.

For A1 and A2, all on-chip registers in a CDC are accessed in parallel, and there are no address registers. In this case, each CDC contains a fixed number of protected registers. Selective error mitigation can be achieved by CEF-aware placement of data in CDC's protected and unprotected registers. For A3, each CDC consists of an SRAM. Instead of a single SRAM, protected and unprotected SRAMs with a fixed address range can be used in this case. Thus, selective error mitigation can be achieved by CEF-aware address assignment and placement, with minimal performance/energy overheads [85]. A4 uses a DRAM to store the data. In this case, ECC bits can be embedded into each data row with low overhead, as shown in [17]. Several papers have focused on a flexible partition of the register file and DRAM with low overheads incorporating selective error mitigation in systems using CPUs and GPUs [17, 86, 133, 140, 141]. The implementation and overheads of selective error-mitigation depend upon the accelerator's microarchitecture, with scope for co-designing microarchitecture and selective error-mitigation implementation.

7 ARCHITECTURAL IMPLICATIONS

In this section, we compare the reliabilities of the four CDMs to draw lessons for resilience-aware CDM design. Error mitigation can incur significant performance, area, and energy overheads (e.g., 3.5× area and energy for TMR), depending upon the CDM architecture. Hence, we need to consider the overheads of error mitigation for making architectural decisions.

Figure 18a compares the CDF rates of accelerators A1, A2, A3, $A3_{scaled}$, and A4. Figure 18b compares the error mitigation overhead for these accelerators for different CDF rates. Even though $A3_{scaled}$ has the highest CDF rate (Figure 18a), ECC can be used for low overhead error mitigation in $A3_{scaled}$ as it uses SRAM.

Geometric representation of swept space: Figure 18c compares the CEF distribution of different geometric representations used in the CDMs studied for a motion set of the Jaco2 robot. We find that two aspects of the geometric representation mainly affect the CEF distribution and range: (1) redundancy, and the (2) volume covered by a structure. In the box-based representation, to take advantage of spatial locality, the optimization process converts the swept space into a set of boxes. Each box covers the maximum possible number of voxels in the swept space. This adds redundancy

as some voxels are covered by multiple boxes, and are hence not included in critical space. In the octree-based representation, nodes near the root node divide the space at a coarser level and represent a much larger volume. Also, there is no redundancy in the representation. These factors result in an overall higher CEF for octree. Conversely, in voxel- and flattened-octree-based representation, each structure/bit represents a single voxel, and so their CEF is less than 6 (maximum 6 surfaces of a voxel). Further, geometric representation determines the suitable storage structures and the error mitigation approach. For example, using SRAM/DRAM with ECC results in lower costs of error mitigation for stringent CDF rate requirements as shown in Figure 18d ($A3_{scaled}$ and A4).

Data reuse: Many accelerators use on-chip buffers to store frequently used data, and exploit data reuse to reduce memory accesses. As a result, a soft error-induced bit flip persists until it is overwritten by reloading the swept space data as explained in Section 4.3. Therefore, if the erroneous data is used for N collision queries, the CDF rate increases by $N/2$ times (Equation 3). However, frequent reloading of on-chip data increases DRAM accesses and incurs performance and energy overheads. Thus there is a trade-off between the overhead of error mitigation and DRAM accesses.

8 RELATED WORK

Fault Characterization: Several FI tools focus on CPUs and GPUs [31, 32, 49, 75, 76, 92, 110, 123, 124, 131]. TRIDENT [76] uses compiler information to estimate SDC probability without performing FI for CPU. gem5-Approxilyzer [131] uses gem5 simulator [13] for FI, and proposes a methodology for fault site pruning. GPU-Qin [31] proposed an FI methodology that balances representativeness and efficiency and performs FI on real GPU hardware. GPU FI tools [32, 92] use GPGPU-Sim [9] for FI, and propose fault site pruning for the GPU SIMT execution model. gpuFI-4 [110] proposes a detailed microarchitecture-level FI tool to characterize the cross-layer vulnerability for single and multiple-bit faults. SASSIFI [49] and NVBitFI [123] propose instrumentation-based FI tools for NVIDIA GPUs. These tools are useful for fault characterization of general purpose applications, but they cannot be easily applied to robotics accelerators that use specialized microarchitectures and instruction sets. More recent papers have studied the resilience of DNN accelerators [53, 54, 74, 78, 101, 108, 125, 134], and autonomous vehicle systems [11, 56, 61–63, 135]. We focus, instead, on robotics accelerators processing spatial information. Due to the differences between the datapath and information processed by the DNN accelerators and motion planning accelerators, the fault propagation to the output is significantly different between them. Another body of work has proposed metrics to characterize the vulnerability of structures to faults [30, 88, 115, 117]. Mukherjee et al. [88] defined the Architectural Vulnerability Factor (AVF), and proposed Architectural Correct Execution analysis (ACE-analysis) to approximate AVF for microprocessor structures [88]. Sridharan et al. [115] proposed the Program Vulnerability Factor (PVF) to decouple the program's fault-masking effect from that of the microarchitecture, and quantify the vulnerability of a program. Fang et al. extended this work in ePVF [30] to consider only SDC-causing bits. Although useful for CPUs and GPUs, architectural or microarchitectural ACE-analysis is challenging to apply to accelerators due to differences in the ISA and workload. For MPAs in particular, whether a bit is ACE heavily depends upon the position of obstacles, and requires multiple simulations to estimate. The CEF gives a measure of the fraction of runtime for which a bit is ACE without such simulations.

Error Mitigation Techniques: There has been significant work on selective error mitigation techniques for CPUs [103], GPUs [81, 85, 95], and FPGAs [78, 98]. Mittal et al. [85] proposed compressing similar values in GPUs. Palframan et al. [95] analyzed GPGPU applications and proposed architectural modifications to reduce the magnitude of errors. Unfortunately, these methods are difficult to apply to MPAs due to differences in the ISA and microarchitecture. Reis et al. [103] and Mahmoud et al. [81] proposed software-level instruction replication to improve the resilience of CPUs and GPUs respectively. However, accelerators typically use complex custom instructions

and perform more computations per instruction [21, 77, 91], and hence software-level duplication would result in significant overheads. Li et al. [76] examined the resilience properties of DNN accelerators and proposed a method to protect vulnerable bits. Guan et al. [43] proposed leveraging application-specific data properties in CNNs to minimize the error correction overhead. In contrast, we focus on accelerators processing spatial information.

Motion Planning: Many techniques have been proposed to accelerate motion planning on CPUs and GPUs [7, 12, 35, 113]. However, these do not meet the energy and performance requirements of autonomous robots [77, 90]. ASIC-based and FPGA-based MPAs [10, 77, 90, 114, 139] focus on performance and energy optimizations, rather than resilience. Other work [22, 121] has studied resilient motion planning under sensor and communication faults, but not soft errors. Note that many of these techniques use the term *roadmap* instead of *motion set* to denote the same idea.

9 CONCLUSIONS AND FUTURE WORK

Motion planning is a critical task in autonomous robots, and motion planning accelerators (MPAs) have been proposed to speed it up significantly. Collision detection is the most resource-consuming and safety-critical module in MPAs. In this work, we propose a spatially-aware reliability metric (CEF) for MPAs, based on the exposed surface area of critical space. We propose a CEF-aware mitigation strategy and Fault Injection (FI) method based on this metric. We also find that CEF-aware error mitigation achieves significant collision detection failure rate reduction, even while incurring low area and energy overheads. We find that CEF-aware FI results in $\sim 23,000\times$ speedup over exhaustive FI to identify the critical bits. Finally, we identify the architectural design parameters affecting the resilience and error mitigation overheads in MPAs. The FI tool and simulator implementation is available at https://github.com/ubc-aamodt-group/MPA_resilience.

There are several possible directions for future work. First, while we focus on single bit flips, both the reliability metric CEF, and the proposed error mitigation and FI methods can be extended to a multi-bit fault model. In particular, Phase 1 of FI, CEF measurement needs to be modified to include a multi-bit fault model, where CEF of a bit can be calculated as an average of CEF for different multi-bit fault combinations. Second, while we focused on probabilistic roadmap-based MPAs, the underlying ideas can be applied to other robotics accelerators that process spatial information. Finally, our observations on the effect of the different design parameters on the resilience and error mitigation overhead open up the direction of "resilience-aware" algorithm-hardware co-design.

REFERENCES

[1] "Design compiler," https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html, 2020.
[2] *2022 International Conference on Robotics and Automation, ICRA 2022, Philadelphia, PA, USA, May 23-27, 2022*. IEEE, 2022. [Online]. Available: https://doi.org/10.1109/ICRA46639.2022
[3] *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*. IEEE, 2022. [Online]. Available: https://doi.org/10.1109/IROS47612.2022
[4] E. Ackerman, "Jaco is a low-power robot arm that hooks to your wheelchair," https://spectrum.ieee.org/the-human-os/robotics/medical-robots/robot-arm-helps-disabled-11-year-old-girl-show-horse-in-competition, 2019.
[5] J. Amanatides and K. Choi, "Ray tracing triangular meshes," 1995.
[6] AMD Xilinx, "Device reliability report second half 2021 (ug116)," https://docs.xilinx.com/r/en-US/ug116/Failure-Rate-Summary, 2022.
[7] N. Atay and B. Bayazit, "A motion planning processor on reconfigurable hardware," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2006, pp. 125–132.
[8] F. Ayatolahi, B. Sangchoolie, R. Johansson, and J. Karlsson, "A study of the impact of single bit-flip and double bit-flip errors on program execution," in *Proceedings of the International Conference on Computer Safety, Reliability, and Security*, ser. SAFECOMP. Springer-Verlag, 2013, pp. 265–276.
[9] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing cuda workloads using a detailed gpu simulator," in *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, 2009.

[10] M. Bakhshalipour, S. B. Ehsani, M. Qadri, D. Guri, M. Likhachev, and P. B. Gibbons, "Racod: Algorithm/hardware co-design for mobile robot path planning," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022.
[11] S. S. Banerjee, S. Jha, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data," ser. DSN. IEEE, 2018, pp. 586–597.
[12] J. Bialkowski, S. Karaman, and E. Frazzoli, "Massively parallelizing the RRT and the RRT," ser. IROS, 2011.
[13] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, p. 1–7, 2011.
[14] D. Bridge, "Autonomous robot market to exhibit a remarkable growth of usd 11607.13 million by 2030," https://www.globenewswire.com/news-release/2023/02/02/2600592/0/en/Autonomous-Robot-Market-to-Exhibit-a-Remarkable-Growth-of-USD-11607-13-Million-by-2030-Size-Share-Trends-Key-Drivers-Growth-and-Opportunity-Analysis.html, 2023.
[15] Cadence Design Systems, "Incisive functional safety simulator," https://www.cadence.com/en_US/home/tools/system-design-and-verification/simulation-and-testbench-verification/incisive-functional-safety-simulator.html, 2020.
[16] Y. Cao, L. Chen, and Z. Zhang, "Memory design for selective error protection," in *Proceedings of the IEEE International Conference on Computer Design*, ser. ICCD, 2015, pp. 133–140.
[17] Y. Cao, L. Chen, and Z. Zhang, "Memory design for selective error protection," in *Proceedings of the IEEE International Conference on Computer Design*, ser. ICCD. IEEE Computer Society, 2015, pp. 133–140.
[18] R. Cappel and C. Perry-Sullivan, "Yield and cost challenges at 16nm and beyond," https://sst.semiconductor-digest.com/2016/02/yield-and-cost-challenges-at-16nm-and-beyond/, 2016.
[19] C.-K. Chang, S. Lym, N. Kelly, M. B. Sullivan, and M. Erez, "Evaluating and accelerating high-fidelity error injection for hpc," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, ser. SC '18. IEEE Press, 2018.
[20] A. Chatzidimitriou, P. Bodmann, G. Papadimitriou, D. Gizopoulos, and P. Rech, "Demystifying soft error assessment strategies on arm cpus: Microarchitectural fault injection vs. neutron beam experiments," in *Proceedings of the International Conference on Dependable Systems and Networks*, ser. DSN, 2019, pp. 26–38.
[21] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2016.
[22] A. L. Christensen, R. O'Grady, M. Birattari, and M. Dorigo, "Fault detection in autonomous robots based on fault injection and learning," *Autonomous Robots*, pp. 49–67, 2008.
[23] Daniel Sorin, "Reliable robotics in academia and industry," Keynote 2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS), 2021.
[24] Deloitte, "Semiconductors - the next wave," https://www2.deloitte.com/content/dam/Deloitte/cn/Documents/technology-media-telecommunications/deloitte-cn-tmt-semiconductors-the-next-wave-en-190422.pdf, 2019.
[25] S. Designline, "Fd soi benefits rise at 14nm," https://www.eetimes.com/fd-soi-benefits-rise-at-14nm/, 2016.
[26] Diligent Robotics, "Moxi helps clinical staff do more," https://diligentrobots.com/moxi, 2020.
[27] C. Ericson, *Real-Time Collision Detection*. CRC Press, Inc., 2004.
[28] H. et al., "2.4 a distributed autonomous and collaborative multi-robot system featuring a low-power robot soc in 22nm cmos for integrated battery-powered minibots," in *2019 IEEE International Solid- State Circuits Conference*, 2019.
[29] J. et al., "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, June 2017, pp. 1–12.
[30] B. Fang, Q. Lu, K. Pattabiraman, M. Ripeanu, and S. Gurumurthi, "epvf: An enhanced program vulnerability factor methodology for cross-layer resilience analysis," in *Proceedings of the International Conference on Dependable Systems and Networks*, ser. DSN, 2016, pp. 168–179.
[31] B. Fang, K. Pattabiraman, M. Ripeanu, and S. Gurumurthi, "Gpu-qin: A methodology for evaluating the error resilience of gpgpu applications," in *IEEE International Symposium on Performance Analysis of Systems and Software*, 2014.
[32] B. Fang, K. Pattabiraman, M. Ripeanu, and S. Gurumurthi, "A systematic methodology for evaluating the error resilience of gpgpu applications." IEEE Press, 2016, pp. 3397–3411.
[33] N. Farazmand, R. Ubal, and D. Kaeli, "Statistical fault injection-based avf analysis of a gpu architecure," *IEEE Workshop on Silicon Errors in Logic*, 01 2012.
[34] A. Faust, O. Ramirez, M. Fiser, K. Oslund, A. Francis, J. Davidson, and L. Tapia, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, 2018, pp. 5113–5120.
[35] R. Gayle, P. Segars, M. Lin, and D. Manocha, "Path planning for deformable robots in complex environments," in *Robotics: Science and Systems*, 2005, pp. 225–232.

[36] B. Gill, N. Seifert, and V. Zia, "Comparison of alpha-particle and neutron-induced combinational and sequential logic error rates at the 32nm technology node," in *IEEE International Reliability Physics Symposium*, 2009, pp. 199–205.
[37] T. Globe and Mail, "Turning autonomous cars into profit won't be easy," https://www.theglobeandmail.com/opinion/article-turning-autonomous-cars-into-profit-wont-be-easy/, 2019.
[38] D. A. G. Gonçalves de Oliveira, L. L. Pilla, T. Santini, and P. Rech, "Evaluation and mitigation of radiation-induced soft errors in graphics processing units," *IEEE Transactions on Computers*, vol. 65, pp. 791–804, 2016.
[39] Google, "Google patents," https://patents.google.com/?q=%22collision+detection%22,%22motion+planning%22,%22collision+avoidance%22,%22path+planning%22&q=autonomous,%22self+driving%22,robotics&country=US&status=GRANT&language=ENGLISH&type=PATENT, 2020.
[40] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtree: A hierarchical structure for rapid interference detection," in *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, 1996.
[41] J. Gracia-Moran, L. Saiz-Adalid, J. Baraza-Calvo, and P. Gil, "Correction of adjacent errors with low redundant matrix error correction codes," in *2018 Eighth Latin-American Symposium on Dependable Computing (LADC)*, 2018.
[42] J. Groopman, "Robots that care," https://www.newyorker.com/magazine/2009/11/02/robots-that-care, 2009.
[43] H. Guan, L. Ning, Z. Lin, X. Shen, H. Zhou, and S.-H. Lim, "In-Place Zero-Space Memory Protection for CNN," in *Advances in Neural Information Processing Systems*, 2019.
[44] M. R. Guthaus, J. E. Stine, S. Ataei, B. Chen, B. Wu, and M. Sarwar, "OpenRAM: An open-source memory compiler," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2016, pp. 1–6.
[45] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, vol. 16, 2016, pp. 243–254.
[46] Y. Han, Y. Yang, X. Chen, and S. Lian, "Dadu series - fast and efficient robot accelerators," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–8.
[47] I. S. Haque and V. S. Pande, "Hard data on soft errors: A large-scale assessment of real-world error rates in gpgpu," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 691–696.
[48] S. K. S. Hari, M. Sullivan, T. Tsai, and S. W. Keckler, "Making convolutions resilient via algorithm-based error detection techniques," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2021.
[49] S. K. S. Hari, T. Tsai, M. Stephenson, S. W. Keckler, and J. Emer, "SASSIFI: An architecture-level fault injection tool for GPU application resilience evaluation," in *IEEE International Symposium on Performance Analysis of Systems and Software*, ser. ISPASS, 2017, pp. 249–258.
[50] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, ser. ICRA, 2015, pp. 2951–2957.
[51] K. Hauser, *Robust Contact Generation for Robot Simulation with Unstructured Meshes*. Springer, Cham, 2016, pp. 357–373.
[52] K. K. Hauser and J. Latombe, "Integrating task and prm motion planning : Dealing with many infeasible motion planning queries," 2009.
[53] Y. He, P. Balaprakash, and Y. Li, "Fidelity: Efficient resilience analysis framework for deep learning accelerators," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020, pp. 270–281.
[54] Y. He, M. Hutton, S. Chan, R. De Gruijl, R. Govindaraju, N. Patil, and Y. Li, "Understanding and mitigating hardware failures in deep learning training systems," in *ISCA*, 2023.
[55] J. L. Hennessy and D. A. Patterson, *Computer Architecture, a quantitative approach*. Morgan Kaufmann, 2012.
[56] Y.-S. Hsiao, Z. Wan, T. Jia, R. Ghosal, A. Mahmoud, A. Raychowdhury, D. Brooks, G.-Y. Wei, and V. J. Reddi, "Mavfi: An end-to-end fault analysis framework with anomaly detection and recovery for micro aerial vehicles," in *DATE*, 2023.
[57] IEC, "Functional safety and iec 61508," http://www.iec.ch/functionalsafety/, 2020.
[58] F. B. Insights, "The global autonomous mobile robots market..." https://www.fortunebusinessinsights.com/autonomous-mobile-robots-market-105055, 2022.
[59] C. L. Jackins and S. L. Tanimoto, "Oct-trees and their use in representing three-dimensional objects," *Computer Graphics and Image Processing*, pp. 249–270, 1980.
[60] L. Jaillet and T. Siméon, "A prm-based motion planner for dynamically changing environments," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, pp. 1606–1611 vol.2, 2004.
[61] S. Jha, S. Banerjee, T. Tsai, S. K. S. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, and R. K. Iyer, "ML-Based Fault Injection for Autonomous Vehicles: A Case for Bayesian Fault Injection," in *DSN*, 2019, pp. 112–124.
[62] S. Jha, S. S. Banerjee, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "AVFI: Fault Injection for Autonomous Vehicles," ser. DSN-W. IEEE, 2018, pp. 55–56.
[63] S. Jha, S. Cui, T. Tsai, S. K. S. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, and R. K. Iyer, "Exploiting temporal data diversity for detecting safety-critical faults in av compute systems," in *DSN*, 2022.

[64] M. Kaliorakis, D. Gizopoulos, R. Canal, and A. Gonzalez, "Merlin: Exploiting dynamic instruction behavior for fast and accurate microarchitecture level reliability assessment," in *ISCA*, 2017.
[65] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, pp. 566 – 580, 1996.
[66] L. E. Kavraki and S. M. LaValle, *Motion Planning*. Springer Berlin Heidelberg, 2008.
[67] T. Kevan, "Will robots find a place in the smart home?" https://www.digitalengineering247.com/article/will-robots-find-a-place-in-the-smart-home, 2019.
[68] S. Kim and A. Somani, "Area efficient architectures for information integrity in cache memories," in *Proceedings of the 26th International Symposium on Computer Architecture*, 1999, pp. 246–255.
[69] KINOVA, "Kinova jaco assistive robot," https://www.kinovarobotics.com/sites/default/files/KINO-2018-Bro-Assistive-ZH_YUL-06-R-Web.pdf, 2018.
[70] S. M. LaValle, "Motion planning," *IEEE Robotics Automation Magazine*, pp. 79–89, 2011.
[71] S. M. LaValle, *Planning Algorithms*. USA: Cambridge University Press, 2006.
[72] P. Leven and S. Hutchinson, "A framework for real-time path planning in changing environments," *The International Journal of Robotics Research*, pp. 999–1030, 2002.
[73] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE, 2009, pp. 502–506.
[74] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (DNN) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC, 2017, pp. 1–12.
[75] G. Li, K. Pattabiraman, C.-Y. Cher, and P. Bose, "Understanding error propagation in gpgpu applications," in *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016, pp. 240–251.
[76] G. Li, K. Pattabiraman, S. K. S. Hari, M. Sullivan, and T. Tsai, "Modeling Soft-Error propagation in programs," ser. DSN. IEEE, 2018, pp. 27–38.
[77] S. Lian, Y. Han, X. Chen, Y. Wang, and H. Xiao, "Dadu-p: A scalable accelerator for robot motion planning in a dynamic environment," in *Proceedings of the Annual Design Automation Conference*, ser. DAC, 2018.
[78] F. Libano, B. Wilson, J. Anderson, M. J. Wirthlin, C. Cazzaniga, C. Frost, and P. Rech, "Selective hardening for neural networks in fpgas," *IEEE Transactions on Nuclear Science*, pp. 216–222, 2019.
[79] Lynxmotion, "Al5d robot arm specs," http://www.lynxmotion.com/driver.aspx?Topic=specs04, 2020.
[80] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, 1962.
[81] A. Mahmoud, S. K. S. Hari, M. B. Sullivan, T. Tsai, and S. W. Keckler, "Optimizing software-directed instruction replication for GPU error detection," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC, Dallas, Texas, 2018, pp. 1–12.
[82] T. Meany, "Functional Safety for Integrated Circuits," Technical Article, Analog Devices https://www.analog.com/media/en/technical-documentation/tech-articles/a54121%20functional-safety-for-integrated-circuits.pdf, 2012.
[83] M. Mehrara and T. Austin, "Exploiting selective placement for low-cost memory protection," *Transactions on Architecture and Code Optimization*, pp. 1–24, 2008.
[84] S. Mitra, T. Karnik, N. Seifert, and M. Zhang, "Logic soft errors in sub-65nm technologies design and CAD challenges," in *Proceedings of the Annual Design Automation Conference*, 2005, pp. 2–4, iSSN: 0738-100X.
[85] S. Mittal, H. Wang, A. Jog, and J. S. Vetter, "Design and Analysis of Soft-Error Resilience Mechanisms for GPU Register File," in *2017 30th International Conference on VLSI Design (VLSID)*, 2017, pp. 409–414.
[86] P. Montesinos, W. Liu, and J. Torrellas, "Using register lifetime predictions to protect register files against soft errors," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks*, ser. DSN, 2007.
[87] H. P. Moravec, "Puma," https://www.britannica.com/technology/PUMA-robot, 2021.
[88] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proceedings of the ACM/IEEE International Symposium on Microarchitecture*, ser. MICRO. IEEE Computer Society, 2003, pp. 29–40.
[89] S. Murray, W. Floyd-jones, G. Konidaris, and D. J. Sorin, "A Programmable Architecture for Robot Motion Planning Acceleration," in *International Conference on Application-specific Systems, Architectures and Processors*, 2019.
[90] S. Murray, W. Floyd-Jones, Y. Qi, D. J. Sorin, and G. Konidaris, "Robot motion planning on a chip," in *Robotics: Science and Systems*, 2016.
[91] S. Murray, W. Floyd-Jones, Y. Qi, G. Konidaris, and D. J. Sorin, "The microarchitecture of a real-time robot motion planning accelerator," in *Proceedings of the ACM/IEEE International Symposium on Microarchitecture*, 2016.
[92] B. Nie, L. Yang, A. Jog, and E. Smirni, "Fault site pruning for practical reliability analysis of GPGPU applications," in *Proceedings of the ACM/IEEE International Symposium on Microarchitecture (MICRO)*, 2018.

[93] OHMNILABS, "The importance of low-cost robotics," https://ohmnilabs.com/content/the-importance-of-low-cost-robotics/, 2019.
[94] P. S. Ostler, M. P. Caffrey, D. S. Gibelyou, P. S. Graham, K. S. Morgan, B. H. Pratt, H. M. Quinn, and M. J. Wirthlin, "Sram fpga reliability analysis for harsh radiation environments," *IEEE Transactions on Nuclear Science*, 2009.
[95] D. J. Palframan, N. S. Kim, and M. H. Lipasti, "Precision-aware soft error protection for GPUs," in *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)*, 2014, pp. 49–59.
[96] G. Papadimitriou and D. Gizopoulos, "Demystifying the system vulnerability stack: Transient fault effects across the layers," in *48th Annual International Symposium on Computer Architecture (ISCA)*, 2021.
[97] G. Papadimitriou and D. Gizopoulos, "Avgi: Microarchitecture-driven, fast and accurate vulnerability assessment," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 935–948.
[98] B. Pratt, M. Caffrey, P. Graham, K. Morgan, and M. Wirthlin, "Improving fpga design robustness with partial tmr," in *2006 IEEE International Reliability Physics Symposium Proceedings*, 2006, pp. 226–232.
[99] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, vol. 37, pp. 48–66, 2021.
[100] J. Ramsey, "40https://www.autoblog.com/2020/05/11/car-electronics-cost-semiconductor-chips/, 2020.
[101] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G.-Y. Wei, "Ares: A framework for quantifying the resilience of deep neural networks," in *DAC*, 2018.
[102] Realtime Robotics, Inc., "Realtime robotics," https://rtr.ai/, 2020.
[103] G. Reis, J. Chang, N. Vachharajani, R. Rangan, and D. August, "SWIFT: software implemented fault tolerance," in *International Symposium on Code Generation and Optimization*, 2005, pp. 243–254.
[104] R. B. Review, "Kinova honored for jaco robotic arm," https://www.roboticsbusinessreview.com/health-medical/kinova_honored_for_jaco_robotic_arm/, 2014.
[105] M. Riera, A. Canal, Ramon anf Gonzalez, J. Abella, M. Anglada, and M. Torrents, "Soft-error vulnerability evolution:a 4d study (bulk/soi, planar/finfet)," in *International Workshop on Reliability and Aging in Forthcoming Electronic Systems*, 2015.
[106] M. Rouman, "Siemens and realtime robotics accelerate the integration of industrial robotic workcells," https://blogs.sw.siemens.com/tecnomatix/siemens-and-realtime-robotics-accelerate-the-integration-of-industrial-robotic-workcells/, 2020.
[107] E. Ruijters and M. Stoelinga, "Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools," *Computer Science Review*, vol. 15-16, pp. 29–62, 2015.
[108] A. Ruospo, G. Gavarini, C. de Sio, J. Guerrero, L. Sterpone, M. S. Reorda, E. Sanchez, R. Mariani, J. Aribido, and J. Athavale, "Assessing convolutional neural networks reliability through statistical fault injections," in *DATE*, 2023.
[109] O. Salzman, "Sampling-based robot motion planning," *Communications of the ACM*, pp. 54–63, 2019.
[110] D. Sartzetakis, G. Papadimitriou, and D. Gizopoulos, "gpufi-4: A microarchitecture-level framework for assessing the cross-layer resilience of nvidia gpus," in *IEEE ISPASS*, 2022.
[111] N. Seifert, V. Ambrose, B. Gill, Q. Shi, R. Allmon, C. Recchia, S. Mukherjee, N. Nassif, J. Krause, J. Pickholtz, and A. Balasubramanian, "On the radiation-induced soft error performance of hardened sequential elements in advanced bulk cmos technologies," in *IEEE International Reliability Physics Symposium*, 2010, pp. 188–197.
[112] D. Shah, N. Yang, and T. M. Aamodt, "Energy-efficient realtime motion planning," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ser. ISCA '23, 2023.
[113] X. Shi, L. Cao, D. Wang, L. Liu, G. You, S. Liu, and C. Wang, "HERO: Accelerating Autonomous Robotic Tasks with FPGA," ser. IROS, 2018, pp. 7766–7772.
[114] D. J. Sorin, G. Konidaris, W. Floyd-Jones, and S. Murray, "Motion Planning for Aotonomous Vehicles and Reconfigurable Motion Planning Processor," U.S. Patent 20 190 163 191, 2019.
[115] V. Sridharan and D. R. Kaeli, "Eliminating microarchitectural dependency from architectural vulnerability," in *Proceedings of the International Symposium on High-Performance Computer Architecture*, ser. HPCA, 2009.
[116] V. Sridharan, N. DeBardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi, "Memory Errors in Modern Systems: The Good, The Bad, and The Ugly," *ACM SIGPLAN Notices*, vol. 50, pp. 297–310, 2015.
[117] V. Sridharan and D. R. Kaeli, "Using hardware vulnerability factors to enhance AVF analysis," in *Proceedings of the IEEE international symposium on Computer architecture*, ser. ISCA, Jun. 2010, pp. 461–472.
[118] J. E. Stine, I. D. Castellanos, M. H. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "FreePDK: An open-source variation-aware design kit," in *IEEE International Conference on Microelectronic Systems Education (MSE)*, 2007, pp. 173–174.
[119] D. Stow, Y. Xie, T. Siddiqua, and G. H. Loh, "Cost-effective design of scalable high-performance systems using active and passive interposers," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017.
[120] M. Sullivan, B. Zimmer, S. Hari, T. Tsai, and S. W. Keckler, "An Analytical Model for Hardened Latch Selection and Exploration," ser. SELSE, 2016.

[121] X. Sun, R. Nambiar, M. Melhorn, Y. Shoukry, and P. Nuzzo, "DoS-Resilient Multi-Robot Temporal Logic Motion Planning," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6051–6057.
[122] Tesla, "Future of driving," https://techxplore.com/news/2020-04-robots-heroes-war-coronavirus.html, 2020.
[123] T. Tsai, S. K. S. Hari, M. Sullivan, O. Villa, and S. W. Keckler, "Nvbitfi: Dynamic fault injection for gpus," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021, pp. 284–291.
[124] S. Tselonis and D. Gizopoulos, "GUFI: A framework for GPUs reliability assessment," ser. ISPASS, 2016.
[125] A. Tyagi, Y. Gan, S. Liu, B. Yu, P. Whatmough, and Y. Zhu, "Thales: Formulating and estimating architectural vulnerability factors for dnn accelerators," *HPCA*, 2023. [Online]. Available: https://par.nsf.gov/biblio/10411383
[126] Uber Advanced Technologies Group, "We believe in the power of technology," https://www.uber.com/ca/en/atg/technology/, 2020.
[127] USPTO, "Uspto patent full-text and image database," http://patft.uspto.gov/netahtml/PTO/search-adv.htm, 2020.
[128] A. Vallero, S. Tselonis, D. Gizopoulos, and S. Di Carlo, "Multi-faceted microarchitecture level reliability characterization for NVIDIA and AMD GPUs," ser. VTS. IEEE, 2018, pp. 1–6.
[129] J. van den Berg and M. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2004.
[130] G. van den Bergen, "Efficient collision detection of complex deformable models using aabb trees," *Journal of Graphics Tools*, pp. 1–13, 1998.
[131] R. Venkatagiri, K. Ahmed, A. Mahmoud, S. Misailovic, D. Marinov, C. W. Fletcher, and S. V. Adve, "gem5-approxilyzer: An open-source tool for application-level soft error analysis," in *DSN*, 2019.
[132] R. Venkatagiri, K. Swaminathan, C. Lin, L. Wang, A. Buyuktosunoglu, P. Bose, and S. Adve, "Impact of software approximations on the resiliency of a video summarization system," in *Proceedings of the International Conference on Dependable Systems and Networks*, ser. DSN, 2018, pp. 598–609.
[133] X. Vera, J. Abella, J.-A. Pineiro, A. Gonzalez, and R. Ronen, "Selectively protecting a register file," U.S. Patent 20 080 155 375, 2008.
[134] A. Veronesi, F. Dall'Occo, D. Bertozzi, M. Favalli, and M. Krstic, "Exploring software models for the resilience analysis of deep learning accelerators: the nvdla case study," in *25th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 2022.
[135] Z. Wan, K. Swaminathan, P.-Y. Chen, N. Chandramoorthy, and A. Raychowdhury, "Analyzing and improving resilience and robustness of autonomous systems," in *ICCAD*, 2022.
[136] Washington State University, "Smart home tests first elder care robot," https://www.sciencedaily.com/releases/2019/01/190114130913.htm, 2019.
[137] M. Wayland, "Toyota's per-car profits lap detroit's big 3 automakers," https://www.detroitnews.com/story/business/autos/2015/02/22/toyota-per-car-profits-beat-ford-gm-chrysler/23852189/, 2015.
[138] Wikipedia, "Programmable universal machine for assembly," https://en.wikipedia.org/wiki/Programmable_Universal_Machine_for_Assembly/, 2020.
[139] Y. Yang, X. Chen, and Y. Han, "Dadu-cd: Fast and efficient processing-in-memory accelerator for collision detection," in *57th ACM/IEEE Design Automation Conference (DAC)*, 2020.
[140] D. H. Yoon and M. Erez, "Virtualized ecc: Flexible reliability in main memory," *IEEE Micro*, pp. 11–19, 2011.
[141] R. Zheng and M. C. Huang, "Redundant memory array architecture for efficient selective protection," in *Proceedings of the International Symposium on Computer Architecture*, ser. ISCA '17, 2017.