electronics

MDPI

Article

Fault Classification and Diagnosis Approach Using FFT-CNN for FPGA-Based CORDIC Processor

Yu Xie, He Chen *, Yin Zhuang and Yizhuang Xie

Beijing Key Laboratory of Embedded Real-Time Information Processing Technology, Beijing Institute of Technology, Beijing 100081, China; xieyu1993@bit.edu.cn (Y.X.); zhuangyin640829@163.com (Y.Z.); xyz551_bit@bit.edu.cn (Y.X.)

* Correspondence: chenhe@bit.edu.cn; Tel.: +86-138-1187-1870

Abstract: Within the realm of digital signal processing and communication systems, FPGA-based CORDIC (Coordinate Rotation Digital Computer) processors play pivotal roles, applied in trigonometric calculations and vector operations. However, soft errors have become one of the major threats in high-reliability FPGA-based applications, potentially degrading performance and causing system failures. This paper proposes a fault classification and diagnosis method for FPGA-based CORDIC processors, leveraging Fast Fourier Transform (FFT) and Convolutional Neural Networks (CNNs). The approach involves constructing fault classification datasets, optimizing features extraction through FFT to shorten the time of diagnosis and improve the diagnostic accuracy, and employing CNNs for training and testing of faults diagnosis. Different CNN architectures are tested to explore and construct the optimal fault classifier. Experimental results encompassing simulation and implementation demonstrate the improved accuracy and efficiency in fault classification and diagnosis. The proposed method provides fault prediction with an accuracy of more than 98.6% and holds the potential to enhance the reliability and performance of FPGA-based CORDIC circuit systems, surpassing traditional fault diagnosis methods such as Sum of Square (SoS).

Keywords: fault diagnosis; Field-Programmable Gate Arrays (FPGAs); Coordinate Rotation Digital Computer (CORDIC); Fast Fourier Transform (FFT); Convolutional Neural Networks (CNNs)

check for updates

Citation: Xie, Y.; Chen, H.; Zhuang, Y.; Xie, Y. Fault Classification and Diagnosis Approach Using FFT-CNN for FPGA-Based CORDIC Processor. *Electronics* **2024**, *13*, 72. https://doi.org/10.3390/electronics13010072

Academic Editor: Alexander Barkalov

Received: 20 November 2023
Revised: 19 December 2023
Accepted: 21 December 2023
Published: 22 December 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the advancement of digital signal processing and communication systems, FPGA-based CORDIC (Coordinate Rotation Digital Computer) processors assume a crucial role. The foundational principle of CORDIC arithmetic is rooted in the simple yet ancient principles of two-dimensional geometry [1]. Initially introduced by Jack E. Volder in 1959, the iterative formulation of this computational algorithm aimed at trigonometric function computation, multiplication, and division [2]. CORDIC-based computing garnered increased attention in 1971 [3], demonstrating its versatility as a unified algorithm capable of implementing a broad spectrum of elementary transcendental functions, including logarithms, exponentials, and square roots, through the adjustment of a few simple parameters. Over the past decade, the CORDIC algorithm has found extensive applications in Digital Signal Processors (DSP), radar signal processing, image processing, software-defined radio, neural networks, and MIMO systems [4].

However, FPGA-based CORDIC processors are susceptible to various factors such as aging, environmental conditions, and manufacturing defects, leading to the occurrence of faults [5]. Especially as the features shrink in size, the microelectronic devices and circuits become more sensitive to space radiation [6]. Faults can result in performance degradation, computational errors, and system failures [7–9]. Therefore, accurate and efficient methods for fault classification and diagnosis are crucial for ensuring the integrity and reliability of the system [10].

To solve the problem of the FPGA sensitivity, scientists and engineers invented many fault-tolerant methods to enhance the reliability of FPGA-based systems. Traditional ways to address fault tolerance (FT) or self-healing are often based on redundancy or algorithm. Triple modular redundancy (TMR) [11], Reduced Precision Redundancy (RPR) [12], Partial TMR [13], Algorithm-Based Fault-Tolerance (ABFT), and concurrent error detection (CED) are commonly used as information redundancy methods. The method for ABFT-FFT takes advantage of convolution theorem is proposed in [14]. In addition, the approach using error correction codes (ECCs) and Sum of Square (SoS) becomes feasible for linear operations such as FFT and adaptive filters [15].

Recent studies have collectively emphasized the potential of deep learning techniques in enhancing fault diagnosis in industrial processes [16]. These techniques also address challenges related to imbalanced datasets, cyber-attacks, and achieving high levels of prediction accuracy and recall. In [17], the challenging problem of fault prediction in industrial processes was investigated using cyber-physical systems. A novel deep learning-based approach called CURNet was proposed, which surpasses existing techniques in terms of fault prediction recall and fault type classification accuracy. The authors of [18] focused on the application of machine learning algorithms for detecting faults in power distribution and transmission systems. The authors of [19] presented a case study on using different machine learning techniques to create heterogeneous ensembles for online failure prediction in software systems. The authors of [20] proposed a semi-supervised label consistent dictionary learning framework for machine fault classification, incorporating both labeled and unlabeled data to enhance representation and classification performance. The authors of [21] proposed the WR-ELM model, a weighted regularization extreme learning machine, for effectively predicting the number of software faults in different scenarios. The authors of [22] provided a comprehensive literature review on the use of software metrics and machine learning algorithms for fault prediction, with a focus on the PROMISE repository dataset. The authors of [23] presented the application of machine learning algorithms for fault prediction in transformers using voltage and current data. Research spans applications in power distribution, transmission systems, and software failure prediction, utilizing diverse machine learning techniques, including semi-supervised learning and weighted regularization models. These advancements underscore the importance of leveraging machine learning for effective fault prediction and classification across different domains, contributing to improved system reliability and performance.

This paper aims to propose a fault classification and diagnosis method for FPGA-based CORDIC processors, leveraging Fast Fourier Transform (FFT) and Convolutional Neural Networks (CNNs). The main features extraction of the fault characteristic signals is optimized by FFT, and CNN performs excellent performance in fault classification and diagnosis task. Using the CNN approach for fault classification and diagnosis, we can enhance the accuracy and efficiency of fault classification and diagnosis, thereby ensuring the reliability and performance of CORDIC processors. The main contributions are summarized as follows.

1. A hardware fault prediction method is proposed. This method provides fault prediction in the module level of CORDIC, and it is based on FFT-CNN.
2. To shorten the time of diagnosis and improve the diagnostic accuracy, the main features extraction of the fault characteristic signals is optimized by FFT. Different CNN architectures are tested to explore and construct the optimal fault classifier.
3. The proposed method provides fault prediction with an accuracy of more than 98.6%, which is higher than the traditional methods such as SoS. The proposed fault prediction approach is tested in an FPGA-based fault injection platform, and it has scalability to work for different circuits.

The structure of this paper is as follows: Section 2 introduces the faults overview, including the circuit architecture of floating-point CORDIC, faults signal selection, and traditional fault diagnosis method. Section 3 elaborates on our proposed FFT-CNN fault classification and diagnosis method, including the construction of fault classification dataset,

feature extraction optimization using FFT, and the training and testing procedures of CNNs. Subsequently, in Section 4, the experimental results are analyzed and discussed, including the implementation and performance analysis. Finally, Section 5 provides a summary of the paper and outlines future work.

2. Faults Overview

A fault refers to an unusual physical condition within a hardware system that leads to an error. An error, in turn, indicates a manifestation of a fault within the hardware system. The output deviates from the anticipated value due to inconsistencies between a component's logical state and its intended state. Moreover, failure denotes the system's incapacity to perform its intended functionality or behavior. Failure can result from the propagation of chain errors to the system level. However, it is worth noting that not all faults in the hardware system will lead to significant errors or failures, as some might remain inactive. Single-Event Upsets (SEUs) that may cause soft errors have emerged as a problematic phenomenon in FPGA processors.

In this paper, the fault-tolerant system mainly consists of Design Under Test (DUT), Fault Injector, and Fault Diagnosis Controller, as shown in Figure 1. We take an FPGA-based floating-point CORDIC processor as the DUT, which is extensively applied in radar signal processing, image processing, remote sensing, and neural networks like SNN [24]. This kind of processors are often used to calculate nonlinear transcendental operations such as sin/cos functions in our programs of spaceborne real-time synthetic aperture radar (SAR) imaging [25]. Then, the fault injector for this DUT is briefly introduced in Section 2.2. This fault injection system can simulate faults by change the configuration data of FPGA; hence, it is necessary for the construction of faults classification dataset. Furthermore, we propose an FFT-CNN fault classification and diagnosis approach, which is considered as the central innovation point and is elaborated in detail in Section 3. Compared with traditional fault diagnosis methods such as SoS, the proposed approach provides fault prediction with higher accuracy to enhance the reliability and performance of FPGA-based CORDIC circuit systems.

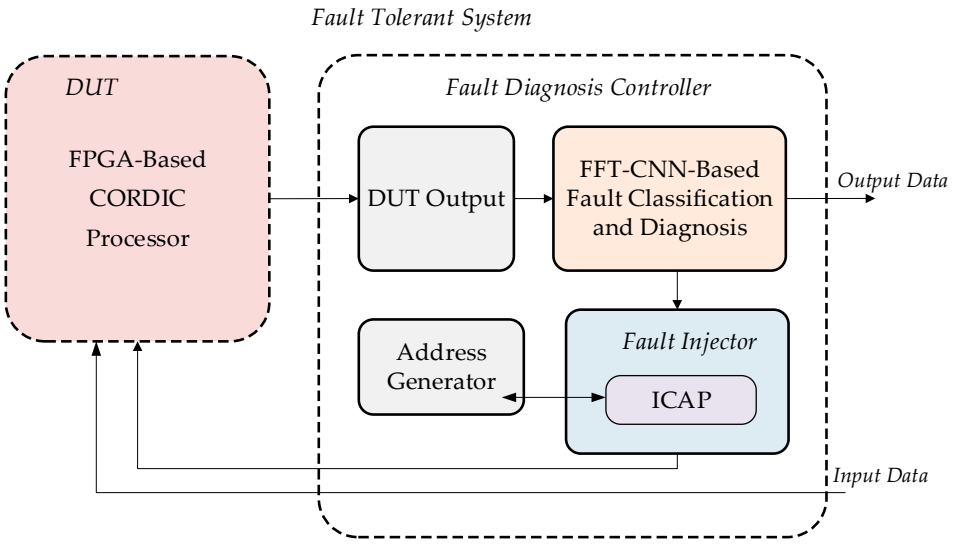


Figure 1. Architecture of the fault-tolerant system.

2.1. DUT for Faults Diagnosis

2.1.1. Architecture of Floating-Point CORDIC

This paper introduces an FPGA-based floating-point CORDIC processor as the DUT, composed of pre-processing units, rotation-processing units, and post-processing units. The top-level structure of the floating-point CORDIC processor is depicted in Figure 2. Internal circuits of each unit in the design achieve maximum reuse, with top-level configu-

ration handling input signals, coordinate systems, and rotation modes to enable different operations. In this design, 2-bit signals T_0 and T_1 are employed to represent the coordinate system, and a 1-bit signal P represents the rotation mode. In the practical implementation, T_0, T_1, and P can be combined into a single 3-bit signal, which is input into various units of the floating-point CORDIC processor. Specifically, pre-processing mainly includes input registers, mathematical transformations, and data formatting. Rotation-processing mainly involves shift operations, iteration processes, and corresponding control logic. Post-processing includes quadrant recovery, output rounding, and output registers.

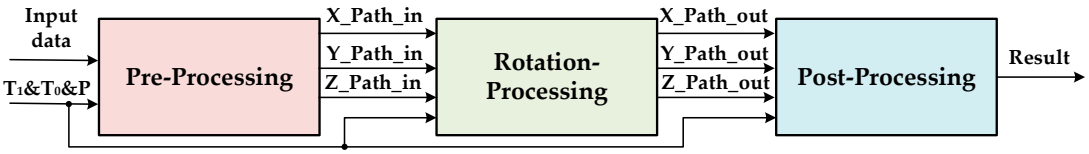


Figure 2. The topological structure of the floating-point CORDIC processor.

2.1.2. Pre-Processing

The pre-processing unit primarily performs the conversion of input data from floating-point to fixed-point through the Range of Convergence (ROC) conversion process, and the hardware structure is shown in Figure 3. This paper adopts the IEEE-754 standard single-precision floating-point data format. The input data can be represented as:

$$X = (-1)^{S_X} \cdot 2^{E_X - Bias} \cdot (1.M_X). \quad (1)$$

where the bias value *Bias* is set to 127, 1 bit represents the sign (*Sx*), 8 bits represent the exponent (*Ex*), and the remaining 23 bits represent the mantissa (*Mx*) or fractional part.

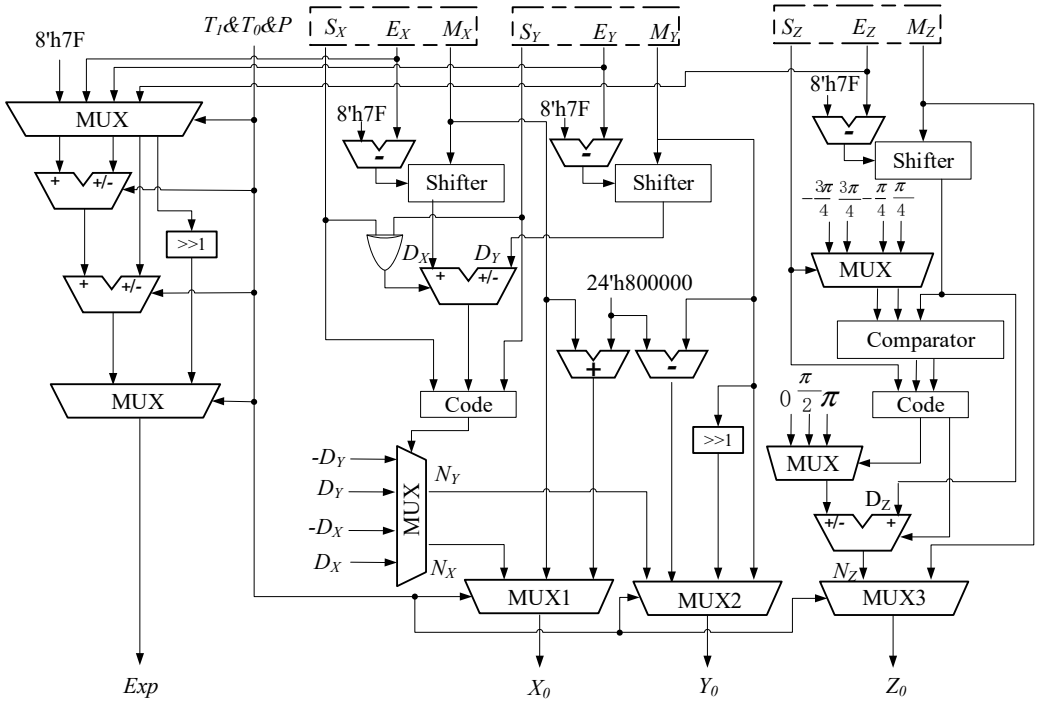


Figure 3. Hardware structure of the pre-processing unit.

Depending on the selected operation by the user, the T_0, T_1, and P signals will choose the appropriate data path. The outputs of the pre-processing modules X_0, Y_0, and Z_0 under different data paths are shown in Table 1. From the table, it can be observed that the pre-processing unit has three outputs to the reconfigurable unit, namely X_0, Y_0, and Z_0. The

processed exponent bits are directly output to the post-processing section for floating-point data normalization.

Table 1. Signal configuration of different operations of pre-processing.

T_1&T_0&P	Operation	X_0	Y_0	Z_0	*Exp*
000	Sine/Cosine	M_X	M_Y	M_Z	/
001	Arctan	N_X	N_Y	M_Z	/
010	Multiplication	M_X	M_Y	M_Z	$E_X + E_Z - 127$
011	Division	M_X	$M_Y/2$	M_Z	$E_Y - E_X + 127$
101	Root	$M_X + 1$	$M_Y - 1$	M_Z	$E_X/2$ or $E_Y/2$

The five operation modes can be categorized into three groups.

1. Trigonometric Functions

In this module, fixed-point numbers are represented as D_X, D_Y, and D_Z. The entire circular data are then mapped to the range covered by the CORDIC algorithm using mathematical transformations.

2. Multiplication and Division

The mantissa is represented as a fixed point, consisting of 1 sign bit, 1 integer bit, and 23 decimal bits, serving as input to the CORDIC rotation unit. The multiplication is given by $E_Y = E_X + E_Z - 127$. Division is represented as $E_Z = E_Y - E_X + 127$. To ensure $|Y/X| < 1$, a right shift unit is added to the Y data path.

3. Square Root

Firstly, the parity of the exponent is determined, and then the mantissa part is normalized based on the exponent part. The initial values for the X and Y paths are taken as M_X+1 and M_Y-1, where $M_X = M_Y$.

2.1.3. Rotation-Processing

The rotation unit adopts a pipelined iterative structure, and a single-layer iterative structure is shown in Figure 4. Each layer requires two 1-bit shifters, three selectors, and three 25-bit add/subtract operators. The rotation direction *selx* and *selz* are defined by the selector MUX3.

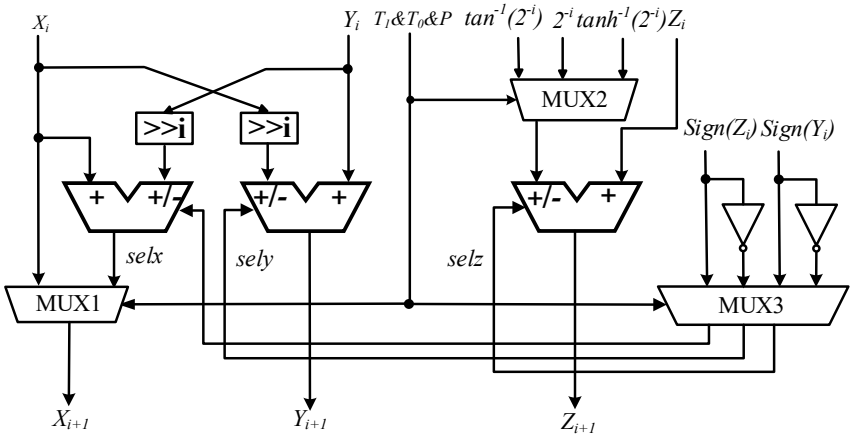


Figure 4. Hardware structure of the rotation-processing unit.

Table 2 provide five different signal configuration modes of multiplexers for computing the corresponding five different operation modes in the pre-processing module.

Table 2. Signal configuration of different operations of rotation-processing.

$T_1\&T_0\&P$	MUX1 X-Path	MUX2 θ_i	*selx*	MUX3 *sely*	*selz*
000	$x_{i+1} = x_i + selx(y_i 2^{-i})$	$\theta_i = tan^{-1}(2^{-i})$	$\sim sign(Z_i)$	$sign(Z_i)$	$\sim sign(Z_i)$
001	$x_{i+1} = x_i + selx(y_i 2^{-i})$	$\theta_i = tan^{-1}(2^{-i})$	$sign(Y_i)$	$\sim sign(Y_i)$	$sign(Y_i)$
010	$x_{i+1} = x_i$	$\theta_i = (2^{-i})$	/	$sign(Z_i)$	$\sim sign(Z_i)$
011	$x_{i+1} = x_i$	$\theta_i = (2^{-i})$	/	$\sim sign(Y_i)$	$sign(Y_i)$
101	$x_{i+1} = x_i + selx(y_i 2^{-i})$	$\theta_i = tanh^{-1}(2^{-i})$	$\sim sign(Y_i)$	$\sim sign(Y_i)$	$sign(Y_i)$

2.1.4. Post-Processing

This module implements quadrant recovery for trigonometric functions and other operations for fixed-point to floating-point conversion. We set a 1-bit signal, where D = 0 represents trigonometric operations and D = 1 represents the other three operations. The quadrant recovery is performed based on the encoding signals S_2, S_1, and S_0 collected by the pre-processing module, as shown in Table 3. Then, the exponent section is normalized, and the mantissa section is determined. After concatenating them, the result is output as a floating-point number. The final output results are represented as *X_result*, *Y_result*, and *Z_result*. The hardware structure is shown in Figure 5.

Table 3. Signal configuration of different operations of post-processing.

$D\&S_2\&S_1\&S_0$	X-Path	Y-Path	Z-Path
0000	X	Y	Z
0001	−Y	X	$Z + \pi/2$
0010	−X	−Y	$Z - \pi$
0011	−X	−Y	$Z + \pi$
0100	Y	−X	$Z - \pi/2$
1XXX	X	Y	Z

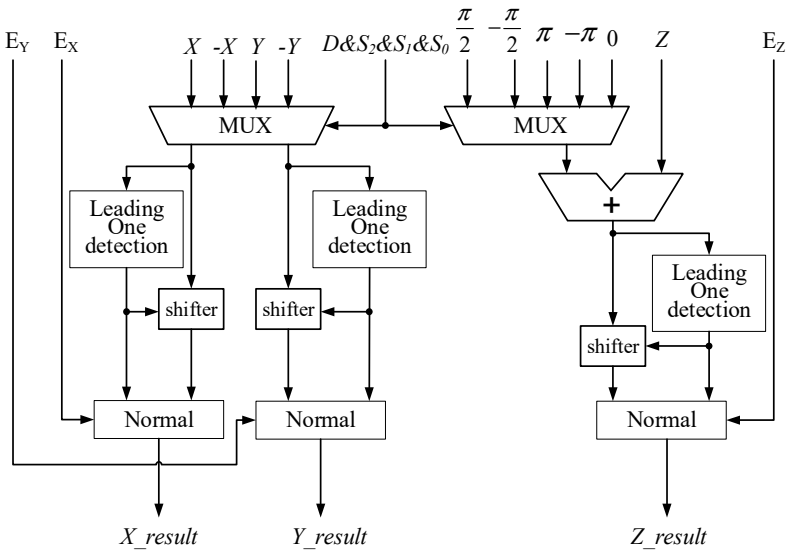


Figure 5. Hardware structure of the post-processing unit.

2.2. Fault Injector

The fault injection system is necessary for the whole fault classification and diagnosis procedure [26]. Fault injection in real conditions of execution [27] is ideal, but it has unavoidable disadvantages such as uncontrollability, high time consumption, and high cost. More recently, fault injection methods based on the Internal Configuration Access Port (ICAP) of SRAM-based FPGAs have drawn much attention [28,29], which are implemented in Commercial Off-the-Shelf (COTS) FPGAs.

The architecture of the proposed fault injection system is shown in Figure 6. It mainly consists of a Fault Injector, Memory Controller, ICAP, and external memory. By utilizing ICAP module, the Fault Injector can be realized. The fault injection or error repair is a simple read-modify-write process. When we finish the procedure of injecting fault, the FPGA operates the data that includes errors. Then, the FFT-CNN fault classification and diagnosis approach come into force. ICAP is a configuration interface which makes it possible to access configuration memory by the finite-state machine (FSM). External memory is used to store the original configuration data.

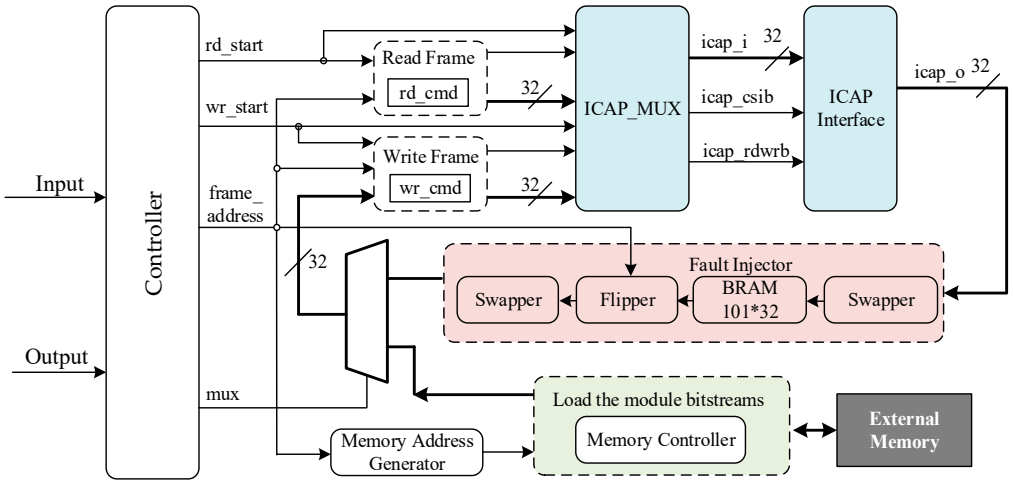


Figure 6. Architecture of the fault injection system.

2.3. Faults Signal Selection

Figure 2 illustrates faults that can occur in the three types of processing units, including the error-free state categorized as a special fault. In the CORDIC DUT circuit, there are generally four fault categories. Category labels are utilized to represent each fault category, as presented in Table 4. If a fault signal belongs to a particular fault category, the corresponding classification category is labeled as 1, while the other categories are labeled as 0, enabling easy distinction among all faults [30]. The output vector index set is recorded as F = {error-free, pre-processing, rotation-processing, post-processing}.

Table 4. Fault classification and labels.

No	Fault Position	Label
1	Error-Free	[1,0,0,0]
2	Pre-Processing	[0,1,0,0]
3	Rotation-Processing	[0,0,1,0]
4	Post-Processing	[0,0,0,1]

2.4. Traditional Fault Diagnosis Method

The conventional approach for managing fault tolerance or self-healing in literature typically relies on redundancy or replication [13]. These methods require significant resource consumption, which may be deemed unacceptable given certain constraints. ABFT methods such as SoS are often used in DSP circuits [15]. For the CORDIC processor, Figure 7 presents a feasible architecture of the SoS check.

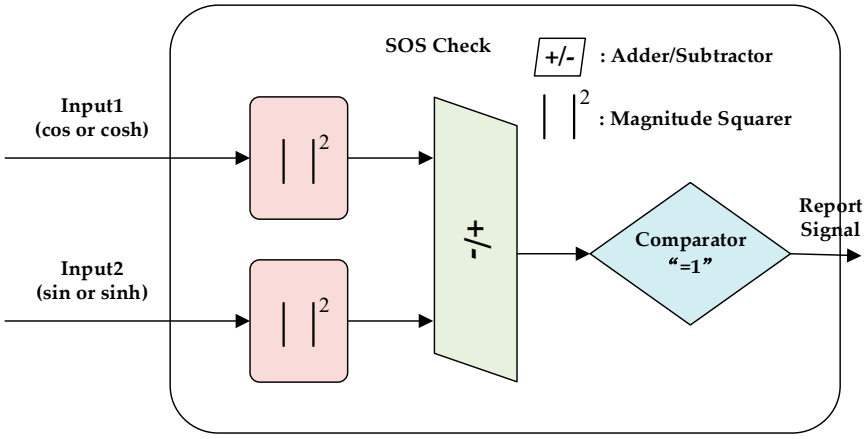


Figure 7. Architecture of the SoS check.

The inputs of the SOS check consist of the outputs from the original component. In the rotation mode of CORDIC circular systems, both the cosine (cos) and sine (sin) results are generated simultaneously. Similarly, in the rotation mode of CORDIC hyperbolic systems, the hyperbolic cosine (cosh) and hyperbolic sine (sinh) results perform the same processing flow. Consequently, the characteristic of trigonometric functions can be leveraged to determine the presence of errors. We employed Equation (2) for the detection process. Input1 and input2 were squared first and then added or subtracted according to the type of CORDIC systems. The resulting value was compared with "1" to obtain the error report signal, assuming a 24-bit fixed-point data format was employed in the FPGA implementation. The data consisted of 22 bits for decimal representation, resulting in a least significant digit (LSB) value of approximately $2^{-22} \approx 2.4 \times 10^{-7}$. Therefore, the confidence interval used in the comparison process was set as $[1 - 10^{-6}, 1 + 10^{-6}]$. More specifically, in the comparator, it was labeled as "=1".

$$\begin{aligned} cos^2\theta + sin^2\theta &= 1 \\ cosh^2\theta - sinh^2\theta &= 1. \end{aligned} \tag{2}$$

For SoS check, the errors caused by the last few stages of rotation processing result in minimal accuracy loss. If the confidence interval is too large, it will affect the speed, while if it is too small, it will lead to a decrease in accuracy. Different confidence intervals were set for the three processing modules in this paper, and the SoS method was used for fault classification and diagnosis experiments with an accuracy of no more than 95%.

3. FFT-CNN Fault Classification and Diagnosis Approach

Figure 8 shows the processing flow of the FFT-CNN fault classification and diagnosis approach. The CORDIC DUT circuit was implemented on the Xilinx Virtex7 XC617V 690T FPGA, which was equipped with two sets of large-capacity 8 GB DDR, allowing for data interaction with the IO interface boards through high-speed transceivers. The IO interface boards and multiple remote sensing data processing boards adhered to the VPX protocol standard and were collectively inserted into an interconnected backplane. The Xilinx Virtex7 FPGA and Jetson TX2 GPU SoC modules were interconnected and engaged in data exchange through a PCI Express Genx2 interface.

The FFT-CNN fault classification and diagnosis approach was implemented on the TX2 GPU SoC modules. This approach mainly consists of three parts:

- Constructing fault classification datasets.
- Optimizing features extraction through FFT to shorten the time of diagnosis and improve the diagnostic accuracy.
- Employing CNNs for the training and testing of fault diagnosis. Different CNN architectures were tested to explore and construct the optimal fault classifier.

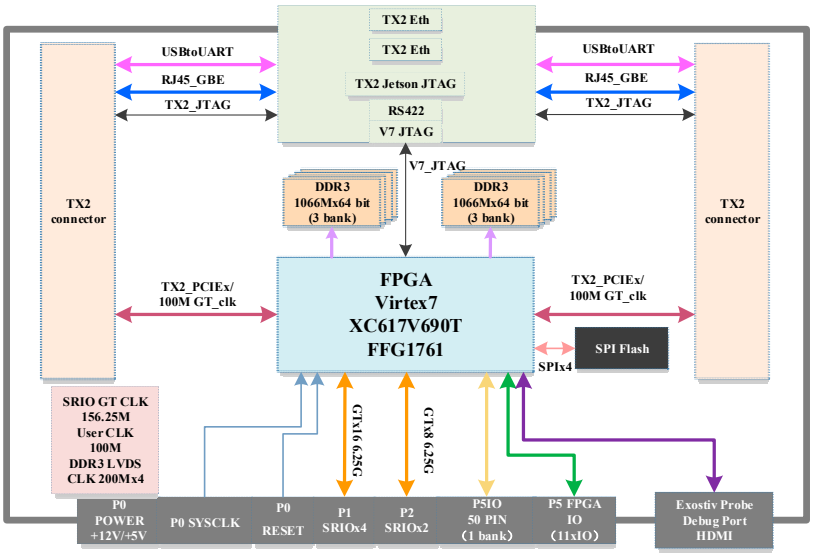


Figure 8. Processing flow of the FFT-CNN fault classification and diagnosis approach.

3.1. Faults Calssification Dataset

To perform fault classification and diagnosis of the FPGA-based CORDIC processor based on CNNs, the first requirement is to construct a fault classification dataset. We selected 18 input phase signals, as shown in Table 5, for the correlative tests. Using the fault injector in Section 2.1, we simulated faults in various modules of the CORDIC processor and obtained corresponding sample datasets based on the labels shown in Table 4. The entire CORDIC processor had a total of 1870 injectable fault positions, including 660 error-free class samples, 231 pre-processing class samples, 559 rotation-processing class samples, and 420 post-processing class samples. After conducting injection tests on all signals, the final size of our dataset was 18 × 1870 = 33,660.

Table 5. Input phase signals for faults classification dataset.

Phase(Π)/24-Bit Format	cos	sin	Phase (Π)/24-Bit Format	cos	sin
−1.0000/'0xC00000'	−1.0000	0.0000	−0.9810/'0xC1374C'	−0.9982	−0.0597
−0.7500/'0xD00000'	−0.7071	−0.7071	−0.7290/'0xD15810'	−0.6590	−0.7522
−0.5000/'0xE00000'	0.0000	−1.0000	−0.4780/'0xE16873'	0.0691	−0.9976
−0.2500/'0xF00000'	0.7071	−0.7071	−0.2270/'0xF178D5'	0.7563	−0.6542
0.0000/'0x000000'	1.0000	0.0000	0.0240/'0x018937'	0.9972	0.0753
0.2500/'0x100000'	0.7071	0.7071	0.2750/'0x11999A'	0.6494	0.7604
0.5000/'0x200000'	0.0000	1.0000	0.5260/'0x21A9FC'	−0.0816	0.9967
0.7500/'0x300000'	−0.7071	0.7071	0.7570/'0x3072B0'	−0.7225	0.6914
1.0000/'0x400000'	−1.0000	0.0000	0.9990/'0x3FEF9E'	−1.0000	0.0031

Figure 9 shows one set of test data. (A) represents the result of the pre-processing fault state, (B) represents the error-free state, (C) represents the post-processing fault state, and (D) represents the rotation-processing fault state. We can observe that compared to the error-free class, all three faulty modules had significant differences in their output results (cos and sin function values). This also preliminarily validates the feasibility of using CNNs for fault classification and diagnosis.

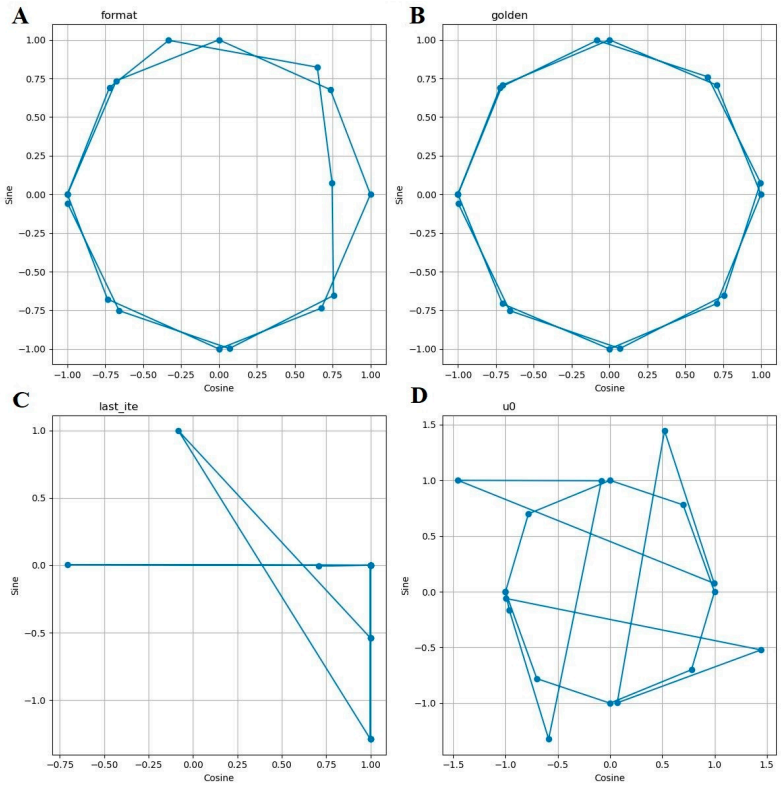


Figure 9. Output data of the CORDIC processor with different fault states. (**A**) Pre-processing fault state; (**B**) error-free state; (**C**) post-processing fault state; (**D**) rotation-processing fault state.

3.2. Feature Extraction Optimization by FFT

The Fourier Transform (FT) is a fundamental mathematical tool used to analyze and decompose signals into their constituent frequency components. It is named after Joseph Fourier, a French mathematician who introduced the concept in the 19th century. The FT converts a time-domain signal into its frequency-domain representation, providing valuable insights into the signal's spectral content and enabling various applications in signal processing, image processing, communications, and more.

The FT decomposes a signal by expressing it as a sum of sinusoidal functions, each characterized by its amplitude and phase at a particular frequency. This transformation allows us to examine the relative contributions of different frequency components present in the signal. By revealing the frequency composition of a signal, the FT enables tasks such as filtering, compression, noise removal, and modulation analysis.

Mathematically, the FT is defined as an integral equation that computes the complex-valued spectrum of a continuous-time signal. The discrete-time counterpart, known as the Discrete Fourier Transform (DFT), is used for analyzing discrete-time signals sampled at uniform intervals. The Fast Fourier Transform (FFT) algorithm is a fast computational method for efficiently computing the DFT, enabling rapid processing of signals in real-time applications.

The Discrete Fourier Transform (DFT) is an algorithm used to convert a digital signal sampled in the time domain into its representation in the frequency domain [31]. The N-point discrete Fourier transform (DFT) of an input sequence is defined as follows.

$$X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n) W_N^{kn}. \quad (3)$$

The Cooley–Tukey algorithm utilizes the divide-and-conquer technique to recursively divide a DFT of size N = M*L into multiple smaller DFTs of sizes M and L. For N = M*L and M= 2^k, let

$$\begin{cases} n = L \cdot m + l; m = 0, 1, ..., M-1; l = 0, 1, ..., L-1 \\ k = i + M \cdot j; i = 0, 1, ..., M-1; j = 0, 1, ..., L-1 \end{cases}. \quad (4)$$

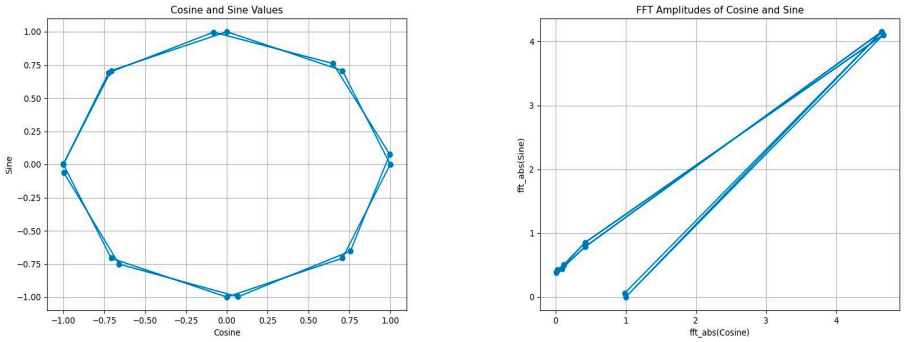
The N-point FFT can be represented in a two-dimensional form as follows:

$$X(i + M \cdot j) = X_0(i) + W_N^i \cdot \underline{W_L^j} \cdot X_1(i) + ... + W_N^{(L-1)i} \cdot \underline{W_L^{(L-1)j}} \cdot X_{L-1}(i). \quad (5)$$

where $X_l(i)$ is the M-point FFT and is described as follows:

$$X_l(i) = \sum_{m=0}^{M-1} x(L \cdot m + l) W_M^{im}. \quad (6)$$

Each hardware fault is represented by its unique frequency characteristics. Therefore, the FFT is used to represent faults in the frequency domain for easier fault classification. In addition, the FFT aids in preprocessing the original signals and contributes to data compression and feature extraction. The FFT is a faster version of the Discrete Fourier Transform (DFT). While computing N-point DFT using the definition requires O(N^2) arithmetic operations, the FFT computation of the same result only requires O(N*$\log_2$N) operations. Hence, the advantages of using FFT can be described as follows. It provides the primary features of fault characteristic signals, making it easier to distinguish faults using the frequency spectrum, as each fault has a unique frequency signature. Figure 10 shows an example comparison of error-free state output between origin data and data after feature extraction optimization by FFT.



(**A**) (**B**)

Figure 10. Comparison of error-free state output data. (**A**) Original data; (**B**) feature extraction optimization by FFT.

Figure 11 shows a set of output data after FFT with different fault states. (A) represents the result of the pre-processing fault state, (B) represents the error-free state, (C) represents the post-processing fault state, and (D) represents the rotation-processing fault state. Compared with Figure 9, we can observe and draw the following conclusion: For the output sine/cosine values of CORDIC processing, the main features of fault characteristic signals can be effectively extracted through FFT. This method increases the inter-class difference and reduces the intra-class difference, which can contribute to CNNs training.

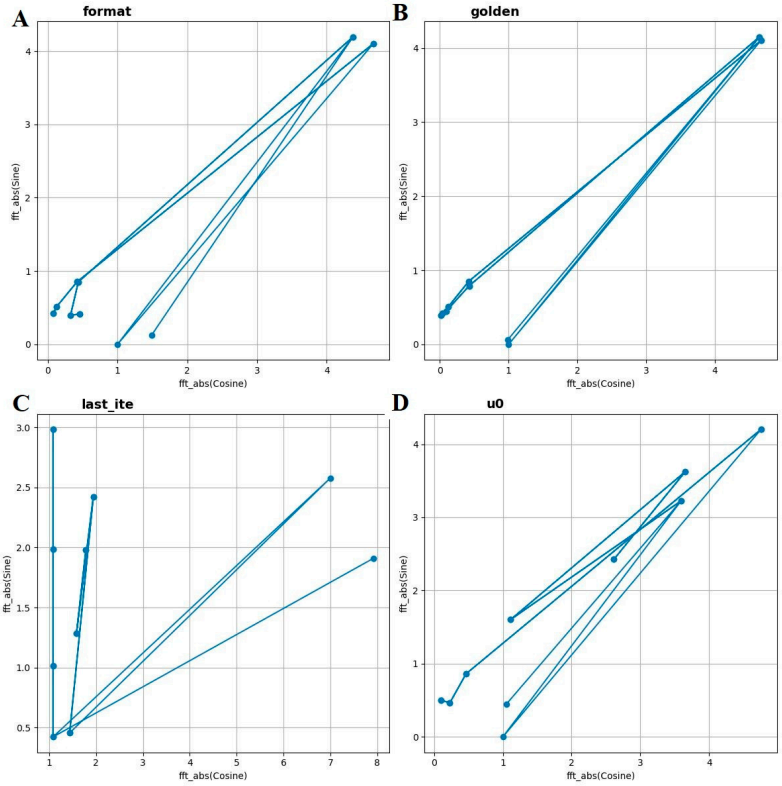


Figure 11. Output data after FFT with different fault states. (**A**) Pre-processing fault state; (**B**) error-free state; (**C**) post-processing fault state; (**D**) rotation-processing fault state.

3.3. CNN-Based Fault Classification and Diagnosis

3.3.1. Review of Different CNNs

In CNNs, convolutional layers are used to extract features from input images [32,33]. The operation in a convolutional layer is a three-dimensional calculation by input feature maps and convolution kernels [34,35]. The calculation of standard convolution is defined as:

$$O_{x,y} = \sum_{c=0}^{C_{in}-1} \sum_{n=0}^{K-1} \sum_{m=0}^{K-1} I_{x+s\times m, y+s\times n} \times w_{m,n} + b_{x,y}. \quad (7)$$

where $I_{x+s\times m,\, y+s\times n}$ is the value of the input feature map at the point of $(x + s \times m, y + s \times n)$, C_{in} is the number of input channels, s represents the stride of the convolutional layer, $w_{m,n}$ is the corresponding weight in the kernels, K is the size of kernels, $b_{x,y}$ is the corresponding bias, and $O_{x,y}$ is the value of the output feature map at the point of (x, y). Several CNN-based neural networks, LeNet-5 [36], ReSNet [37], and VGGNet [38], have been proposed for classification.

LeNet-5 is a convolutional neural network (CNN) architecture developed by Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner in 1998. It was originally designed for recognizing handwritten digits, making it one of the pioneering models for image classification tasks. LeNet-5 consists of seven layers, including three convolutional layers followed by max-pooling layers and then fully connected layers. It uses the tanh activation function and primarily relies on convolutional layers to extract features from input images. LeNet-5 has a relatively small number of parameters and provides impressive results on simple image recognition tasks.

ResNet (short for "Residual Network") is a deep learning architecture introduced by Kaiming He et al. in 2015. It addresses the problem of vanishing gradients in deep networks and enables the training of very deep neural networks. ResNet18 and ResNet34 are two variants of the ResNet architecture. They consist of multiple residual blocks, where

each block has multiple convolutions and shortcut connections. The main idea behind ResNet is the introduction of identity mappings, which enables the information to skip one or more layers within the block. This helps to alleviate the degradation problem of network performance when increasing the depth. By adding skip connections, ResNet enables the training of deeper networks without worrying about the vanishing gradient problem. ResNet18 has 18 convolutional layers, while ResNet34 has 34 layers. Generally, as the depth increases, ResNet models tend to perform better on challenging image recognition tasks.

VGG (Visual Geometry Group) is a deep convolutional neural network architecture developed by the Visual Geometry Group at the University of Oxford. It was introduced by Karen Simonyan and Andrew Zisserman in 2014. VGGNet follows a simple structure consisting of multiple stacked convolutional layers and max-pooling layers, followed by fully connected layers. VGGNet has different variations named after the number of layers it has. VGG11 and VGG13 refer to configurations with 11 and 13 convolutional layers, respectively. VGGNet focuses on increasing the depth rather than the width (number of filters) of the network. This allows it to learn complex hierarchical features and capture fine-grained details. However, the large number of parameters in VGGNet leads to increased memory requirements and longer training times compared to other models.

When comparing these networks, it is important to consider factors such as model performance, computational requirements, and the specific task at hand. Different networks may excel in different scenarios, and it is often necessary to experiment and fine-tune the model to achieve the best results for a specific problem.

In this paper, different CNN architectures were adopted to explore and construct the optimal fault classifier for CORDIC. We used LeNet-5, VGGNet11/13, and ResNet18 /34 for comparative experiments at the same level. Specifically, for the dataset in Section 3.1, we made optimizations to the original network and redesigned a network structure suitable for CORDIC faults classification. The final network structure used for training and testing is shown in Figures 12 and 13 and Table 6.

In order to match the network architectures of LeNet-5, VGGNet11/13, and ResNet18/34, we encapsulated the one-dimensional data shown in Table 5 into a 32 × 32 size. This was done as an optimization in the network design process. By resizing the one-dimensional data to a 32 × 32 matrix, we were able to adhere to the input requirements of the chosen network architectures. This enabled us to utilize the existing network structures and operations designed for image-based tasks.

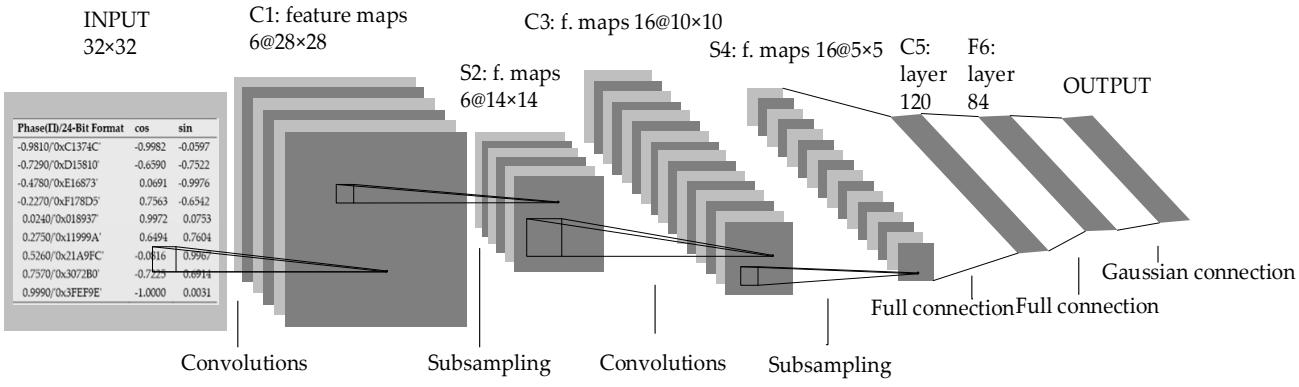


Figure 12. Architecture of LeNet-5 neural network for CORDIC fault diagnosis.

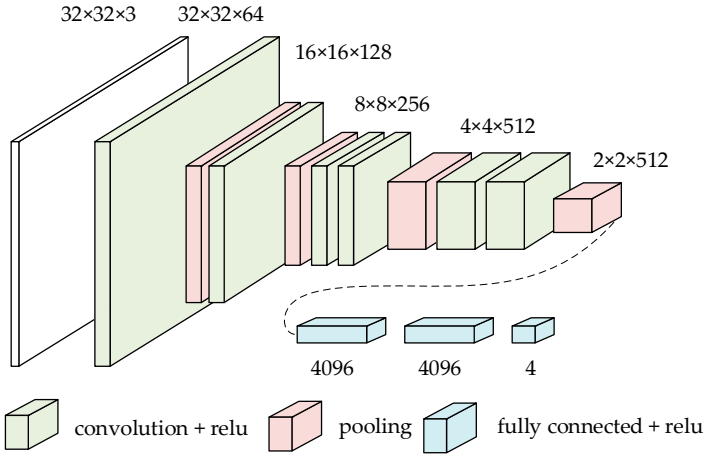


Figure 13. Diagram of the VGG11 neural network.

Table 6. Architecture of the ResNet18 and ResNet34 neural networks.

Layer Name	Output Size	18-Layer	34-Layer
conv1	32 × 32	7 × 7, 64, stride 2	
conv2_x	16 × 16	3 × 3, max pool, stride 2	
		$\begin{bmatrix} 3\times3 & 64 \\ 3\times3 & 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3 & 64 \\ 3\times3 & 64 \end{bmatrix} \times 3$
conv3_x	8 × 8	$\begin{bmatrix} 3\times3 & 128 \\ 3\times3 & 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3 & 128 \\ 3\times3 & 128 \end{bmatrix} \times 4$
conv4_x	4 × 4	$\begin{bmatrix} 3\times3 & 256 \\ 3\times3 & 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3 & 256 \\ 3\times3 & 256 \end{bmatrix} \times 6$
conv5_x	2 × 2	$\begin{bmatrix} 3\times3 & 512 \\ 3\times3 & 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3 & 512 \\ 3\times3 & 512 \end{bmatrix} \times 3$
	1 × 1	average pool 1000-d fc, SoftMax.	
FLOPs		1.8×10^9	3.6×10^9

Therefore, the modifications made to the network architectures involved encapsulating the one-dimensional data into the appropriate 32 × 32 size. This enabled us to effectively utilize the convolutional layers, pooling layers, and other relevant network operations within the chosen architectures.

3.3.2. Training and Testing

Figure 14 shows the training steps for the proposed method. A total of 33,660 samples were used for training, validation, and testing. In the training mode, 60% of the samples (20,196 samples) were used for training, 20% (6732 samples) were used for validation, and the remaining 20% (6732 samples) were used for testing.

After the first step of fault injection and CORDIC processing, the data samples go through FFT preprocessing to obtain data features in the frequency domain. In the second step, the output of the FFT is passed to the CNN, which performs feature extraction and learning steps after setting appropriate parameters. Then, there is the classification process, which is used to determine the fault state. Finally, in the third step, the predicted fault state is compared with the actual fault state. If the predicted fault matches the actual fault, the training stops. Otherwise, feedback on the results is sent back to the FFT and preprocessing stage to update the relevant parameters for more accurate learning. This iterative process continues until the predicted fault state matches the actual fault state. In the testing stage, new data are subjected to FFT transformation, and the FFT result is sent to the CNN-based

classification model to perform its role, completing the entire fault diagnosis process and providing the final classification decision for the fault.

The training and testing model are shown in Figure 15. The process is divided into two stages: training and testing. In the training stage, the input cosine and sine values, along with their corresponding fault labels, are applied to train the neural network, to establish a model Function f(cos, sin) for the input data. This model is then used to test future inputs during the testing stage. In the testing stage, new inputs are used to test the model and validate its operation. The model provides corresponding outputs, i.e., fault categories, for the inputs. The FFT preprocessing is applied to the neural network for training. In the training cases, the neural network learns patterns of different fault shapes and presents this training process as a model. A model consists of assumptions formed based on the input data to classify the outputs during the testing stage. Testing is the next step after completing the training, where new inputs are tested. In the testing cases, the inputs undergo FFT preprocessing first and then the network is tested using the results. The fault classification dataset is applied to the neural network using the model constructed in the training stage to obtain classification and diagnosis results.

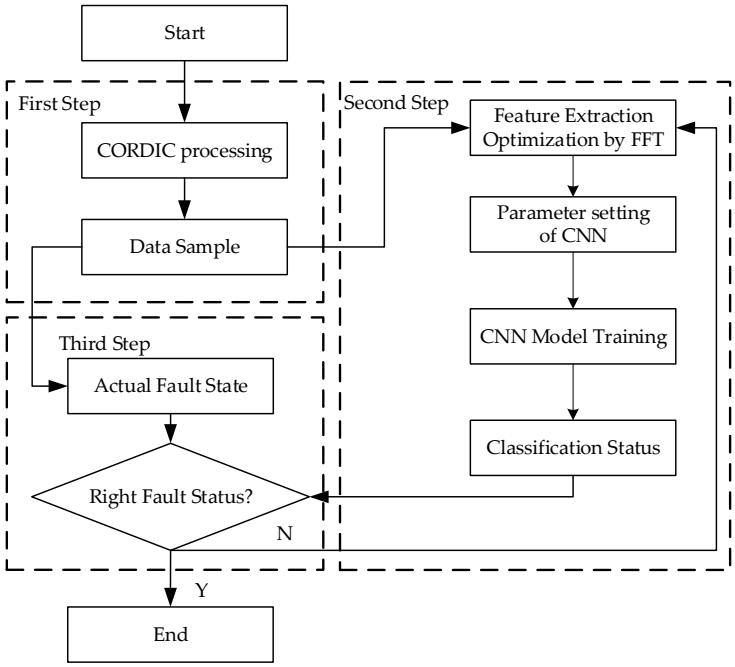


Figure 14. The training flow of the proposed FFT-CNN fault diagnosis approach.

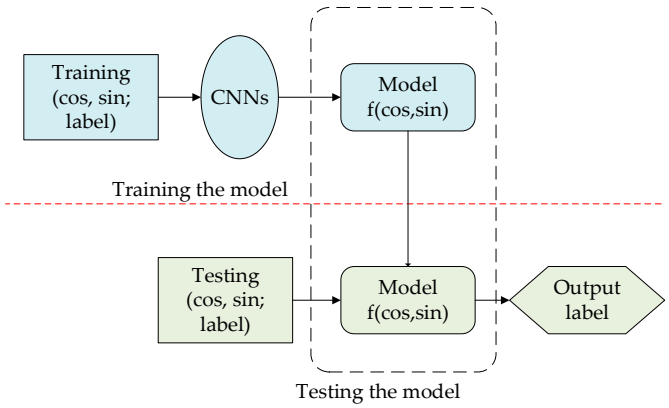


Figure 15. Fault diagnosis model of the proposed FFT-CNN approach.

3.3.3. Evaluation

The accuracy of a CNN network can be computed using the following equation, using the terms TP (True Positive), TN (True Negative), FP (False Positive), and FN (False Negative):

$$Accuracy = (TP + TN) \ / \ (TP + TN + FP + FN). \quad (8)$$

In this equation, TP represents the number of true positive predictions, which are instances where the network correctly predicts a positive class. TN represents the number of true negative predictions, which are instances where the network correctly predicts a negative class. FP represents the number of false positive predictions, which are instances where the network incorrectly predicts a positive class. FN represents the number of false negative predictions, which are instances where the network incorrectly predicts a negative class. To calculate accuracy, we sum TP and TN (the correct predictions) and divide it by the sum of TP, TN, FP, and FN (the total number of predictions). This gives us the overall accuracy of the CNN network's predictions.

4. Experimental Results and Analysis

4.1. Simulation and Implementation

Xilinx Virtex7 FPGA XC7VX690T-1FFG176 and NVIDIA Jetson TX2 (Tegra X2 SoC module) were employed on the hardware platform [39,40]. Xilinx Virtex7 was equipped with high-speed transceivers. Jetson TX2 was equipped with Denver 2 and ARM Cortex-A57 CPUs in addition to a GPU. Figure 16 shows the hardware architecture of the whole fault-tolerant system implemented in this paper.

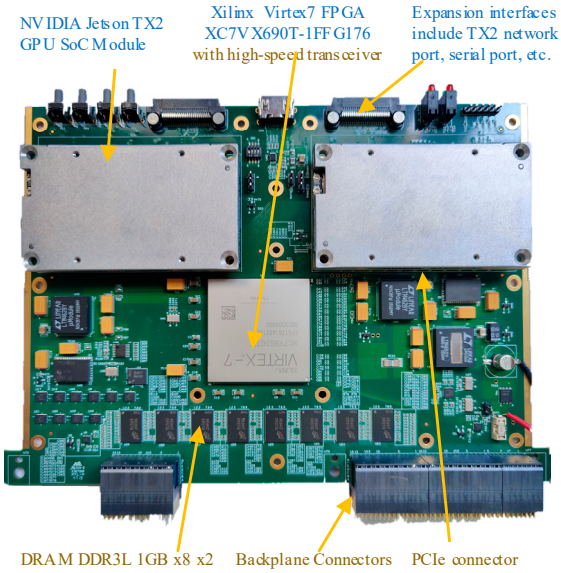


Figure 16. Hardware architecture of the fault-tolerant system.

The hardware implementation of FPGA mainly included the CORDIC processor introduced in Section 2.1 (which is selected as the DUT) and the fault injector introduced in Section 2.2 (for the construction of the fault classification dataset). In Section 2.2 and as illustrated in Figure 6, the architecture of the proposed fault injection system is explained and shown. This architecture mainly consisted of the Fault Injector, Memory Controller, ICAP (Internal Configuration Access Port), and external memory. Each of these components plays a crucial role in the fault injection process. We designed an optimized memory controller to reduce the hardware resource of the fault injector. As shown in Table 7, our fault injector occupied 385 LUTs, 56 FFs, and 1.5 RAMs, less than 1% of the total resources in Virtex7 FPGA 690T. Compared to the Soft Error Mitigation (SEM) IP [41] created by

Xilinx, our approach has less hardware overheads and therefore has fewer sensitive areas for higher reliability [42].

Table 7. Hardware resource comparison of fault injectors.

	Hardware Overhead			
	LUTs	**FFs**	**BRAM (RAM36)**	**I/O**
Our approach	385	56	1.5	1
SEM	559	139	1.5	56

Hardware Resource of the CORDIC DUT circuit and total is presented in Table 8. The proposed FFT-CNN fault classification and diagnosis approach is implemented on NVIDIA Jetson TX2, including ten different methods as shown in Table 9.

Table 8. Hardware resource of the DUT and total.

	CORDIC(DUT)	**Fault Injector**	**Total**
Slice Registers (as FF)	2110	385	2495
Slice LUTs	2238	56	2294

Table 9. Performance comparison of different classification methods.

Different Methods	**Accuracy/%**	**Training Epoch Number of Max Accuracy**
LeNet-5	98.21	192
FFT-LeNet-5	98.65	188
VGG11	97.31	177
FFT-VGG11	98.66	174
VGG13	96.87	148
FFT-VGG13	98.66	146
ResNet18	96.43	146
FFT- ResNet18	99.55	144
ResNet34	86.55	78
FFT- ResNet34	96.41	68
SoS (traditional)	95%	\

4.2. Comparison and Analysis

The performance comparison of different classification methods is shown in Table 9. For the CORDIC module, performing FFT on the results to achieve clustering and then deep learning yielded better results compared to directly learning from the original results. The original results exhibited scattered data distribution, with similar within-class and between-class variances, making it difficult to classify. However, applying FFT to the CORDIC signals effectively increased the between-class variance while reducing the within-class variance, thus improving the classification performance. By comparing and analyzing these data, we can draw the following three conclusions:

(1) Different methods' accuracy: From the Accuracy column, it can be found that accuracy varied when using different CNN models and different FFT-CNN methods for training and testing the dataset. The FFT-LeNet-5 and FFT-VGG13 methods had higher accuracy, with 98.65% and 98.66%, respectively. Compared with VGG11/13 and ResNet18/34, LeNet-5 is a network with the fewest layers, which means it had comparable accuracy to other methods while utilizing resources more optimally. It is notable that ResNet34 had a relatively lower accuracy of 86.55%. This may have been due to the network's excessive depth. Deep networks can experience a decrease in classification and diagnostic performance in certain situations [37].

(2) Training epoch number corresponding to the maximum accuracy: From the Training Epoch Number of Max Accuracy column, it can be observed that different methods require different training epoch numbers to achieve the maximum accuracy. For example, the FFT-ResNet18 method reached the highest accuracy of 99.55% at the 144th training epoch. From data in the table, it can be observed that as the number of network layers increased, the convergence speed of the entire learning process became faster.

(3) Impact of FFT preprocessing: It should be noted that after FFT preprocessing, the required learning time was shorter, and the accuracy was higher. For example, compared to directly training on the fault classification dataset, the FFT-LeNet-5 and FFT-VGG13 methods achieved better accuracy in less time.

In order to better illustrate the performance improvement of the proposed FFT-CNN fault classification and diagnosis approach, we selected relevant data for visual comparison, including the traditional SoS check, Lenet-5, FFT-Lenet-5, and FFT-Resnet18, as shown in Figure 17. The results presented in the figure align with the conclusions mentioned above. The FFT-Resnet18 method, which demonstrated the best performance, can achieve an accuracy improvement of 4.55% over the traditional SoS method. To be more specific, the accuracy of the FFT-Resnet18 method was 99.55%, while the accuracy of the traditional SoS method was 95%.

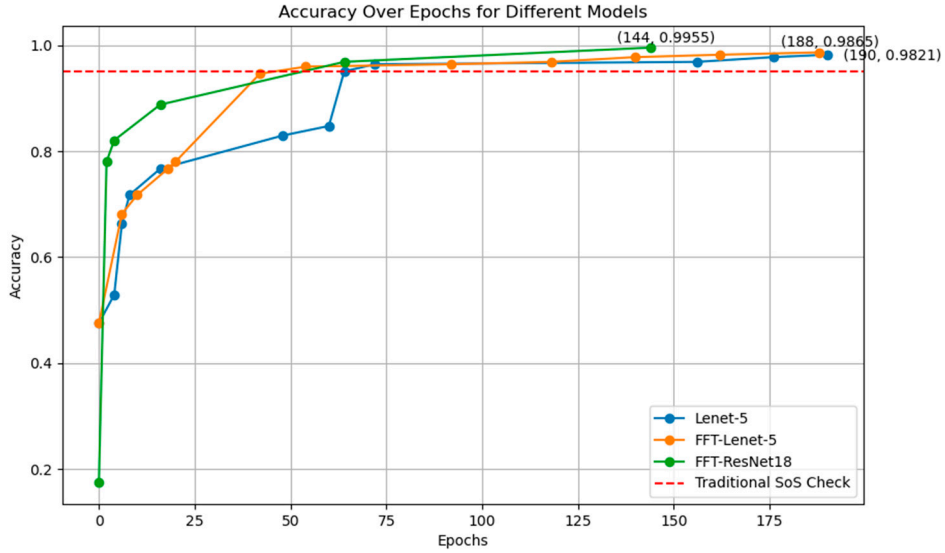


Figure 17. Accuracy comparison of the different methods.

5. Conclusions

This study proposes a fault classification and diagnosis method based on FFT-CNN for the FPGA-based CORDIC processor. By utilizing FFT and CNN, we successfully constructed a fault classification dataset and perform feature extraction, fault classification, and diagnosis. Experimental results demonstrate excellent accuracy and reliability of the method, achieving over 98.6% accuracy. This indicates the potential of this method in improving the performance and reliability of FPGA-based CORDIC circuit systems.

Despite the encouraging results achieved in this study, there are several aspects that can be further improved and explored. The following are potential directions for future work: (1) Expanding the scope of application: The current method primarily applies to CORDIC processors. Future research could consider extending it to other types of processors or devices to increase the algorithm's generality. (2) Considering more fault types: The current method focuses mainly on soft errors, but there exist other types of faults in practical applications, such as hardware faults and power supply noise. Further research could consider extending the fault classification and diagnosis models to cover a wider range of fault types. (3) Optimizing algorithm performance: Although the current method

has achieved good results, there is still room for further optimization. Future research could explore more efficient feature extraction methods, more precise designs of classification models, and strategies for handling large-scale datasets.

In conclusion, the fault classification and diagnosis method based on FFT-CNN provides an efficient and accurate means of fault diagnosis for FPGA-based CORDIC processors. Future research can further expand the application scope, optimize algorithm performance, and apply it to more systems.

Author Contributions: H.C. and Y.X. (Yu Xie) conceived and designed the framework. Y.X. (Yizhuang Xie) and Y.X. (Yu Xie) performed the mapping technology and analyzed the data. Y.X. (Yu Xie) and Y.Z. debugged the system hardware. Y.X. (Yu Xie) wrote the paper. Y.Z. and H.C. reviewed papers and submitted comments. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by program under Grant JCKY2021602B037 and the National Natural Science Foundation of China (General Program) under Grant 62371048.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Meher, P.K.; Valls, J.; Juang, T.-B.; Sridharan, K.; Maharatna, K. 50 Years of CORDIC: Algorithms, Architectures, and Applications. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2009**, *56*, 1893–1907. [CrossRef]
2. Volder, J.E. The Birth of Cordic. *J. Signal Process. Syst.* **2000**, *25*, 101–105. [CrossRef]
3. Walther, J.S. The Story of Unified Cordic. *J. Signal Process. Syst.* **2000**, *25*, 107–112. [CrossRef]
4. Lakshmi, B.; Dhar, A.S. CORDIC Architectures: A Survey. *VLSI Des.* **2010**, *2010*, 794891. [CrossRef]
5. Kwak, J.-H.; Piuri, V.; Swartzlander, E. Fault-tolerant high-performance CORDIC processors. In Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, Yamanashi, Japan, 25–27 October 2000. [CrossRef]
6. Wang, S.; Wen, Z.; Yu, L. High-Performance Fault-Tolerant CORDIC Processor for Space Applications. In Proceedings of the 2006 1st International Symposium on Systems and Control in Aerospace and Astronautics, Harbin, China, 19–21 January 2006. [CrossRef]
7. Ibrahim, M.M.; Asami, K.; Cho, M. Evaluation of SRAM based FPGA performance by simulating SEU through fault injection. In Proceedings of the 2013 6th International Conference on Recent Advances in Space Technologies (RAST), Istanbul, Turkey, 12–14 June 2013; pp. 649–654. [CrossRef]
8. Xilinx. *UltraScale Devices Maximize Design Integrity with Industry-Leading SEU Resilience and Mitigation*; White Paper—WP462 (v1.0); Xilinx: San Jose, CA, USA, 2015.
9. Souari, A.; Thibeault, C.; Blaquiere, Y.; Velazco, R. An automated fault injection for evaluation of LUTs robustness in SRAM-based FPGAs. In Proceedings of the 2015 IEEE East-West Design & Test Symposium (EWDTS), Batumi, Georgia, 26–29 September 2015; pp. 1–4. [CrossRef]
10. Cheatham, J.A.; Emmert, J.M.; Baumgart, S. A survey of fault tolerant methodologies for FPGAs. *ACM Trans. Des. Autom. Electron. Syst.* **2006**, *11*, 501–533. [CrossRef]
11. Kim, E.P.; Shanbhag, N.R. Soft N-Modular Redundancy. *IEEE Trans. Comput.* **2010**, *61*, 323–336. [CrossRef]
12. Snodgrass, J. Low-Power Fault Tolerance for Spacecraft FPGA-Based Numerical Computing. Ph.D. Thesis, Naval Post-graduate School, Monterey, CA, USA, 2006.
13. Xie, Y.; Yu, W.-Y.; Zhang, N.; Chen, H.; Xie, Y.-Z. Partial Triple Modular Redundancy Method for Fault-Tolerant Circuit based on HITS Algorithm. In Proceedings of the 2023 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Juan-Les-Pins, France, 3–5 October 2023; pp. 1–4. [CrossRef]
14. Reviriego, P.; Bleakley, C.; Maestro, J. A novel concurrent error detection technique for the fast Fourier transform. In Proceedings of the IET Irish Signals and Systems Conference (ISSC 2012), Maynooth, Ireland, 28–29 June 2012; pp. 1–5. [CrossRef]
15. Gao, Z.; Reviriego, P.; Xu, Z.; Su, X.; Zhao, M.; Wang, J.; Maestro, J.A. Fault Tolerant Parallel FFTs Using Error Correction Codes and Parseval Checks. *IEEE Trans. Very Large Scale Integr. Syst.* **2015**, *24*, 769–773. [CrossRef]
16. Khalil, K.; Eldash, O.; Kumar, A.; Bayoumi, M. Machine Learning-Based Approach for Hardware Faults Prediction. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**, *67*, 3880–3892. [CrossRef]
17. Ruan, H.; Dorneanu, B.; Arellano-Garcia, H.; Xiao, P.; Zhang, L. Deep Learning-Based Fault Prediction in Wireless Sensor Network Embedded Cyber-Physical Systems for Industrial Processes. *IEEE Access* **2022**, *10*, 10867–10879. [CrossRef]
18. Mandal, S.; Bohre, A.K. Fault Classification in Electrical Systems Using Machine Learning Algorithms. In Proceedings of the 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT), Kannur, India, 11–12 August 2022; pp. 1412–1416. [CrossRef]
19. Campos, J.R.; Costa, E.; Vieira, M. Improving Failure Prediction by Ensembling the Decisions of Machine Learning Models: A Case Study. *IEEE Access* **2019**, *7*, 177661–177674. [CrossRef]

20. Jiang, W.; Zhang, Z.; Li, F.; Zhang, L.; Zhao, M.; Jin, X. Joint Label Consistent Dictionary Learning and Adaptive Label Prediction for Semisupervised Machine Fault Classification. *IEEE Trans. Ind. Inform.* **2015**, *12*, 248–256. [CrossRef]
21. Bal, P.R.; Kumar, S. WR-ELM: Weighted Regularization Extreme Learning Machine for Imbalance Learning in Software Fault Prediction. *IEEE Trans. Reliab.* **2020**, *69*, 1355–1375. [CrossRef]
22. Meiliana; Karim, S.; Warnars, H.L.H.S.; Gaol, F.L.; Abdurachman, E.; Soewito, B. Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset. In Proceedings of the 2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), Phuket, Thailand, 20–22 November 2017; pp. 19–23. [CrossRef]
23. Hadiki, H.; Hasnaoui, F.S.; Georges, S. Transformers Faults Prediction Using Machine Learning Approach. In Proceedings of the 2023 Fifth International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), Zouk Mosbeh, Lebanon, 5–7 July 2023; pp. 30–35. [CrossRef]
24. Wu, J.; Zhan, Y.; Peng, Z.; Ji, X.; Yu, G.; Zhao, R.; Wang, C. Efficient Design of Spiking Neural Network with STDP Learning Based on Fast CORDIC. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2021**, *68*, 2522–2534. [CrossRef]
25. Xie, Y.; Xie, Y.; Li, B.; Chen, H. Advancements in Spaceborne Synthetic Aperture Radar Imaging with System-on-Chip Architecture and System Fault-Tolerant Technology. *Remote Sens.* **2023**, *15*, 4739. [CrossRef]
26. Mohammadi, A.; Ebrahimi, M.; Ejlali, A.; Miremadi, S.G. SCFIT: A FPGA-based fault injection technique for SEU fault model. In Proceedings of the 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE 2012), Dresden, Germany, 12–16 March 2012; pp. 586–589. [CrossRef]
27. Quinn, H.; Graham, P.; Morgan, K.; Baker, Z.; Caffrey, M.; Smith, D.; Wirthlin, M.; Bell, R. Flight Experience of the Xilinx Virtex-4. *IEEE Trans. Nucl. Sci.* **2013**, *60*, 2682–2690. [CrossRef]
28. Xie, Y.; Qiao, T.; Xie, Y.; Chen, H. Soft error mitigation and recovery of SRAM-based FPGAs using brain-inspired hybrid-grained scrubbing mechanism. *Front. Comput. Neurosci.* **2023**, *17*, 1268374. [CrossRef] [PubMed]
29. Xie, Y.; Chen, H.; Xie, Y.-Z.; Mao, C.-A.; Li, B.-Y. An Automated FPGA-Based Fault Injection Platform for Granularly-Pipelined Fault Tolerant CORDIC. In Proceedings of the 2018 International Conference on Field-Programmable Technology (FPT), Naha, Japan, 10–14 December 2018; pp. 370–373. [CrossRef]
30. Wang, T.; Qi, J.; Xu, H.; Wang, Y.; Liu, L.; Gao, D. Fault diagnosis method based on FFT-RPCA-SVM for Cascaded-Multilevel Inverter. *ISA Trans.* **2016**, *60*, 156–163. [CrossRef]
31. Xie, Y.; Wei, X.; Chen, L.; Xie, Y.-Z.; Chen, H. Variable-length and high-precision FFT processors based on configurable constant factor multipliers and memory reallocations. *IEICE Electron. Express* **2018**, *15*, 20180610. [CrossRef]
32. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems 28 (NIPS 2015), Montreal, QC, Canada, 7–12 December 2015.
33. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man Cybern.* **1979**, *9*, 62–66. [CrossRef]
34. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
36. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
37. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef] [PubMed]
38. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**. [CrossRef]
39. Nvidia, JETSON TX. Available online: https://www.nvidia.cn/autonomous-machines/embedded-systems/jetson-tx2 (accessed on 2 March 2023).
40. Xu, X.; Dong, X.; Xie, Y. On-Board Wind Scatterometry. *Remote Sens.* **2020**, *12*, 1216. [CrossRef]
41. Xilinx IP, Soft Error Mitigation Controller LogiCORE (S.E.M.C.L). *Soft Error Mitigation Controller LogiCORE IP Product Guide (PG036)*; Xilinx: San Jose, CA, USA, 2015.
42. Stoddard, A.; Gruwell, A.; Zabriskie, P.; Wirthlin, M.J. A Hybrid Approach to FPGA Configuration Scrubbing. *IEEE Trans. Nucl. Sci.* **2016**, *64*, 497–503. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.