electronics

MDPI

Article

Implementation of Highly Reliable Convolutional Neural Network with Low Overhead on Field-Programmable Gate Array

Xin Chen [1,*], Yudong Xie [1], Liangzhou Huo [1], Kai Chen [1], Changhao Gao [1], Zhiqiang Xiang [1], Hanying Yang [1], Xiaofeng Wang [2], Yifan Ge [2] and Ying Zhang [1]

[1] College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China; wxnxybwja@nuaa.edu.cn (Y.X.); hlz1031@nuaa.edu.cn (L.H.); chenkai2436@nuaa.edu.cn (K.C.); gaochanghao@nuaa.edu.cn (C.G.); xiangzq@nuaa.edu.cn (Z.X.); hanying_yang@nuaa.edu.cn (H.Y.); tracy403@nuaa.edu.cn (Y.Z.)

[2] Beijing Aerospace Automatic Control Institute, Beijing 100854, China; wenyd@muc.edu.cn (X.W.); 16051139@buaa.edu.cn (Y.G.)

* Correspondence: xin_chen@nuaa.edu.cn

Abstract: Due to the advantages of parallel architecture and low power consumption, a field-programmable gate array (FPGA) is typically utilized as the hardware for convolutional neural network (CNN) accelerators. However, SRAM-based FPGA devices are extremely susceptible to single-event upsets (SEUs) induced by space radiation. In this paper, a fault tolerance analysis and fault injection experiments are applied to a CNN accelerator, and the overall results show that SEUs occurring in a control unit (CTRL) lead to the highest system error rate, which is over 70%. After that, a hybrid hardening strategy consisting of a finite state machine error-correcting circuit (FSM-ECC) and a triple modular redundancy automatic hardening technique (TMR-AHT) is proposed in this paper to achieve a tradeoff between radiation reliability and design overhead. Moreover, the proposed methodology has very small workload and good migration ability. Finally, by full exploiting the fault tolerance property of CNNs, a highly reliable CNN accelerator with the proposed hybrid hardening strategy is implemented with Xilinx Zynq-7035. When BER is 2×10^{-6}, the proposed hybrid hardening strategy reduces the whole system error rate by 78.95% with the overhead of an extra 20.7% of look-up tables (LUTs) and 20.9% of flip-flops (FFs).

Keywords: convolutional neural network; single-event upsets; field-programmable gate array

check for updates

Citation: Chen, X.; Xie, Y.; Huo, L.; Chen, K.; Gao, C.; Xiang, Z.; Yang, H.; Wang, X.; Ge, Y.; Zhang, Y. Implementation of Highly Reliable Convolutional Neural Network with Low Overhead on Field-Programmable Gate Array. *Electronics* **2024**, *13*, 879. https://doi.org/10.3390/electronics13050879

Academic Editor: Alexander Barkalov

Received: 7 November 2023
Revised: 29 January 2024
Accepted: 5 February 2024
Published: 25 February 2024

1. Introduction

Convolutional neural networks (CNNs) have emerged as the most successful algorithms in the field of artificial intelligence (AI) due to their superior performance in tasks such as image classification [1], object detection [2], and real-time analysis [3]. Compared to CPU or GPU solutions, CNN accelerators implemented on SRAM-based field-programmable gate arrays (FPGAs) have the distinct advantages of high flexibility and low power consumption. However, when an SRAM-based FPGA is hit by high-energy particles, the logic level of the affected region is easily flipped, which is known as the single-event upset (SEU). After that, the normal operation of the CNN will be disrupted, and operation errors may be induced, such as disorders in program execution, incorrect calculation results, and even system crashes. Thereby, several studies have recently investigated the impact of SEUs on neural networks, algorithm-based fault-tolerance (ABFT) strategies, such as the quantization technique, and redundancy techniques, which are represented by redundant PEs, and so on.

With the aid of fault injection technology, ref. [4] evaluated the resilience of a CNN to SEUs and found that convolutional and pooling layers are more resilient than fully connected layers. Ref. [5] analyzed the susceptibility of a deep neural network (DNN) and artificial neural network (ANN) to SEUs through partial reconfiguration technology.

The analyses showed that the effects of SEUs on these neural networks are inversely proportional to the data flow distance from the hit network layer to the last layer. SEUs in processing elements (PEs) are more critical than those occurring in RAMs. If one PE out of 6000 PEs crashes, the accuracy of GoogLeNet and MobileNetV2 would drop by more than 20% [6].

The principle of the quantization technique is converting the data format from a floating point to a fixed point. Ref. [7] applied binary quantization to convolutional layers and found that CNNs with quantized convolutional layers can effectively reduce sensitivity to radiation. Compared to a 32-bit floating-point implementation, an 8-bit integer design can provide more than six times the fault-free execution [8]. A dynamic fixed-point quantization was presented to reduce the bit width of the data from 32-bit to 8-bit. The symbol error rate (SER) of quantized ZynqNet was reduced by 71.36%, and the circuit area was reduced by 44.76% compared with standard ZynqNet [9]. Ref. [10] performed a comprehensive layer-by-layer fault analysis of isomorphic and heteromorphic quantized DNNs. The research results indicated that quantizing the DNN model to fewer bits can help improve the resilience of the model. However, quantization bits that are too small may sacrifice the resilience and accuracy of the model. Similarly, through modifying the configuration memory, ref. [11] performed fault injection experiments on a binary neural network (BNN). Compared with a sequential circuit, including registers and BRAMs, the reliability of a BNN was obviously degraded when the SEUs occurred in the configuration memory.

The redundancy technique is another popular solution for hardening the CNN. The error-detecting mechanism of a PE array was adopted in [6]. During the idle status of the CNN accelerator, the algorithm checks the function of all LUTs and DSPs, locates incorrect PEs, and informs the controller to avoid using the disrupted PEs. Ref. [12] proposed a hybrid computing architecture (HyCA) and a dot-production processing unit (DPPU) to recalculate all operations mapped to faulty PEs in a two-dimensional computing array. Multiply-and-accumulate units (MACs) and memory cells (MEMs) are hardened by the triple modular redundancy (TMR) technique [9]. Ref. [13] adopted TMR technology to selectively harden the last network layer, which has been proven to be the most susceptible layer to SEUs in CNN accelerators. However, the approach of hardening an entire network layer level does not achieve a good trade-off between radiation reliability and design overhead. Ref. [14] developed an ensemble of weak CNNs to build a low-cost robust classifier. The system reliability was improved when suffering from soft errors, and the overhead was much lower than that when using TMR.

To minimize the design cost, the ABFT strategy was also studied recently. Aiming to operate a multiply matrix in convolutional kernels of the YOLO network, the error checking and correction technique was adopted in [15], and it was found that 50% to 60% of multiply faults induced by SEUs can be detected and corrected. Selective multiply accumulate zero-optimization (SMART) checks whether the input value provided to the neural network neuron is zero [16]. If the value is zero, the corresponding multiply accumulate operation is bypassed, and the fault tolerance of the tested neural network is increased by 1.78 times. Ref. [17] proposed a retraining-based solution, which utilizes fault results induced by SEUs in the CNN as samples and introduces a penalty mechanism to retrain the CNN. However, the retraining procedure needs many samples and resources, and it is time-consuming. Therefore, the retraining-based solution is not fit for the mobile hardware platform. During training, ref. [18] added the fault resilience principle to pretrained networks of DNNs with the cost of a few fine-tuning steps. The results showed that the failure rate of neuronal activation can be significantly reduced for single- and multiple-bit flip failures.

In summary, it is hard to improve the radiation reliability performance with a small design overhead using the existing hardening techniques. There are two main reasons. Firstly, the scale of CNNs is huge, and hardening a large number of circuits will consume significant resources. Secondly, the CNN algorithm is complex, and the computations in CNNs are very intensive, so it is very difficult to efficiently correct or mask the SEU errors of CNNs.

In this paper, the SEU tolerance of CNN accelerators is analyzed in detail. Based on the analyzed results, SEU fault experiments are carried out to search the most sensitive components (MSCs) against SEU errors. Thereafter, a hybrid hardening strategy that combines a finite-state machine error-correcting circuit (FSM-ECC) and the triple modular redundancy automatic hardening technique (TMR-AHT) is proposed to obtain a good balance between radiation reliability and design overhead. Meanwhile, the workload of the proposed methodology is very small, and the migration ability is also very good. The experimental results show that the proposed hybrid hardening strategy applied to the control unit (CTRL) of all layers reduces the whole system error rate by 62.5% with the cost of 20.7% look-up tables (LUTs) and 20.9% flip-flops (FFs).

The remainder of this paper is organized as follows. Section 2 outlines the basics of CNN accelerator and SEU fault simulation. SEU tolerance analysis of CNN and identifications of MSCs are presented in Section 3. Section 4 proposes a hybrid hardening strategy for a CNN accelerator. In Section 5, the hardening performance and the design overhead of a CNN accelerator are evaluated and analyzed. Finally, conclusions and future works are given in Section 6.

2. Basics of CNN Accelerator and SEU Fault Simulation

2.1. CNN Accelerator

As one of the most typical CNN networks, Lenet-5 is composed of three convolutional layers, two pooling layers, and two fully connected layers. Figure 1a shows the model of CNN networks and the dimensions of each layer. Convolutional layers (Conv1, Conv2, Conv3) utilize a lot of MACs, and extract key features from input images or features. The dimensions of key features are reduced further by pooling layers (Pool1, Pool2), which include many comparators. Convolutional and pooling layers are connected alternately, followed by fully connected layers (Fc1, Fc2). The fully connected layers then complete the classification or prediction task based on these features.

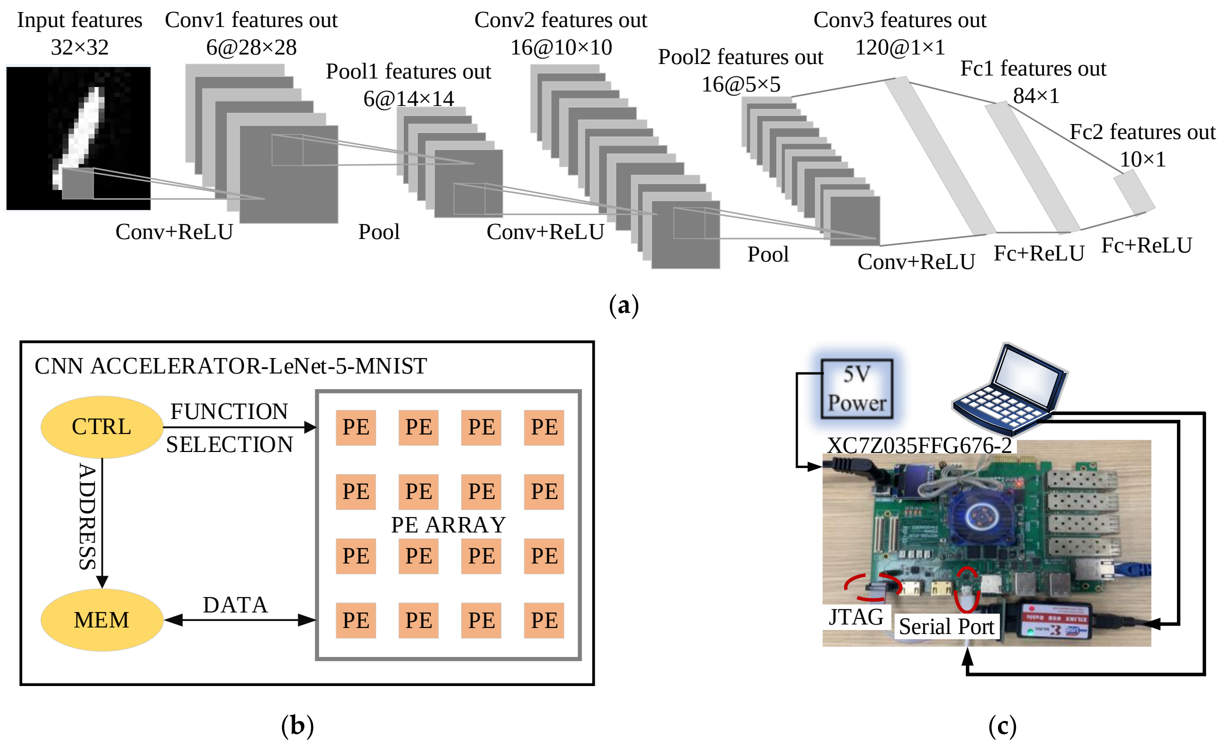


Figure 1. Model, architecture, and implementation platform of CNN: (**a**) Lenet-5 model; (**b**) architecture, (**c**) hardware platform.

As seen from Figure 1b, the CNN accelerator consists of the CTRL, the MEM, and the PE array to implement the function of the CNN model shown in Figure 1a. The CTRL transfers reading/writing addresses to the MEM, and sends specific instructions to PEs, such as convolution or pooling calculations. The MEM stores CNN parameters which usually are input/output feature maps, weights, and biases of each layer, and so on. The PE array contains lots of PEs to perform parallel operations, and each PE consists of numerous MACs, adders, comparators, and so on. In other words, for the convolutional layer, pooling layers, and fully connected layers shown in Figure 1a, the operations of these layers are controlled by the CTRL, the related data are read from MEM, computed by the PE array, and then written back to the MEM.

The design of the CNN accelerator shown in Figure 1b is written in Verilog HDL first. The Xilinx EDA tool is then utilized to synthesis and implement the CNN design, and generate the configuration file for SRAM-FPGA. When the hardware platform shown in Figure 1c is powered on, the corresponding FPGA, which is Xilinx XC7Z035FFG676-2, is configured based on the configuration file.

2.2. SEU Fault Model

SEUs will occur in any component of a CNN accelerator. Figure 2 shows an example of SEU fault injection and propagation. As seen from Figure 2, an SEU occurs in a storage cell of weight BRAM. Based on control data from the CTRL, the flipped data in weight are transferred to PEs and involved in convolutional calculations. Finally, the SEU error propagates to activation function rectified linear unit (ReLU) and results in incorrect convolutional results. To simulate these SEUs, the SEU fault model SFM described by Equation (1) is utilized for fault injection.

$$SFM =< ST, FN, FD, BER > \tag{1}$$

where ST denotes the start time of one fault injection, FN represents the flipped node identity, FD indicates the fault duration of injected SEU, and BER is short for bit error rate, which is determined by the intensity of radiation sources.

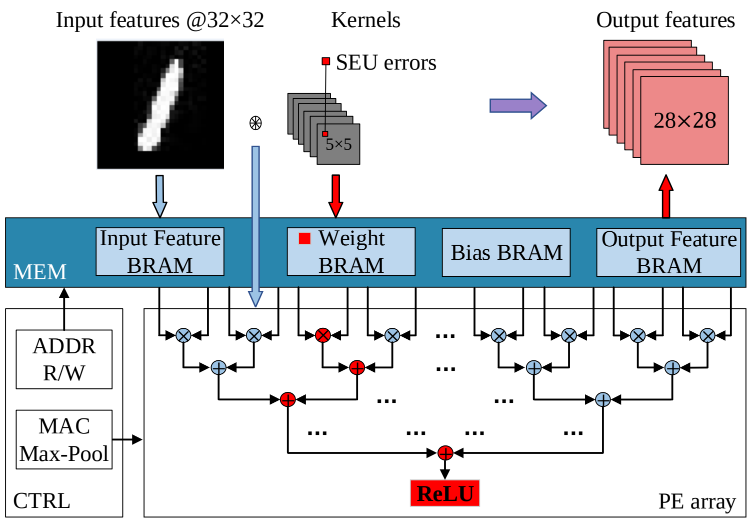


Figure 2. Convolution calculations with SEUs in CNN accelerator.

The value of BER is defined by Equation (2). A higher BER value indicates that more SEUs occur in the CNN within the same simulation time.

$$BER = N_{FN}/N_{TN} \tag{2}$$

where N_{FN} is the number of flipped nodes and N_{TN} is the number of total nodes.

It is worthy to note that the units of ST and FD are both the cycle of the system clock. Assuming that T is the total operating cycles of CNN accelerator for a task, the value of ST is a random integer which ranges from 0 to $T-1$. When FD is 1, it represents that an SEU error lasts only one cycle of the system clock.

2.3. SEU Evaluation Platform

Compared to the typical SEE evaluation scheme [19,20], the proposed scheme provides a rapid solution to analyze the vulnerable modules in the logic designs before implementation [21].

As described in Figure 3, the SEU evaluation platform performs fault injection simulation into the evaluated CNN design, and analyzes the simulation results. The evaluated CNN design is instantiated twice, named c1 and c2. The original stimuli are used for the normal operation of the evaluated circuit. The fault injection stimuli are sent only to the evaluated circuit c2. The fault monitor monitors the outputs of evaluated circuits c1 and c2 simultaneously. If the outputs are different, it indicates that an error is induced by the injected fault. The analyzed results are then stored for analysis.

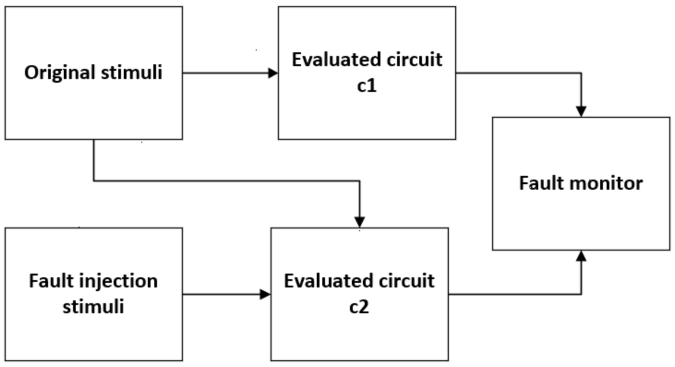


Figure 3. Operation of SEU evaluation platform.

The whole operation is performed automatically by Python script; the workload of SEU/SET evaluation is not large. In addition, the evaluated circuit does not need to be modified for SEU/SET evaluation, and then making mistakes in modification work is avoided. The reason is that the operation of fault injection is implemented by the assignment statements "force" and "release" provided by Verilog HDL. The assignment statement "force" can force the value of a circuit node to be the designated value when "force" is effective. The assignment statement "release" can release the force assignment and restore the circuit to normal operation. Complex fault injection simulations are also able to be performed through the combination of "force" and "release".

2.4. CNN Fault Type

The classification accuracy (CA) is the average probability of correct classification. The definition of CA is given by Equation (3), and *LF(x,y)* is a simple logic function which is shown in Equation (4).

$$CA = \frac{1}{N}\sum_{i=1}^{N} LF(argmax(v_i), Lbl_i) \tag{3}$$

$$LF(x,y) = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases} \tag{4}$$

where v_i is one-dimensional vector of probability elements output by the Fc2 layer, $argmax(v_i)$ returns the index of the maximum value of v_i, and Lbl_i is the label index of the i-th image.

However, the classification accuracy is not accurate enough to reflect the effect of SEUs on the operation of the CNN accelerator. In this paper, the failure status of the CNN accelerator caused by SEUs is divided into four types, which are system crash, serious error, tolerable error, and benign error, as shown in Table 1. Under the case of benign error, the classification category represented by $argmax(v_i)$ and classification accuracy of Lentet-5 network represented by v_i are both correct, and the system runs correctly. Under the case of tolerable error, although the classification accuracy is wrong, the probability of the corresponding category is still the largest, so this image can be correctly classified. Serious error means that both the classification category and the classification accuracy are both wrong. System crash indicates that no result is given on the desired time.

Table 1. Fault types induced by SEU.

Fault Types	Notes
System crash	No values are given on time.
Serious error	Both $argmax(v_i)$ and v_i are wrong.
Tolerable error	v_i is wrong, but $argmax(v_i)$ is correct.
Benign error	Both $argmax(v_i)$ and v_i are correct.

3. SEU Tolerance Analysis of CNN

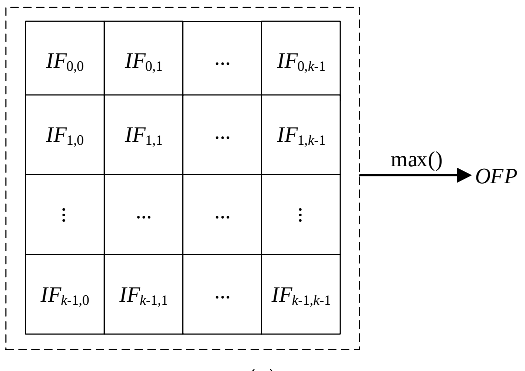
The input/output features, weights, and biases are stored in the MEM. Based on control data from the CTRL, the corresponding data in the MEM is transferred to all PEs and participates in the calculations. Fortunately, many memory hardening techniques such as TMR and error correcting code are reported and work very well. Therefore, only the adverse effects of SEU errors on the PE array and the CTRL are analyzed here.

3.1. SEU Tolerance Analysis of PE Array

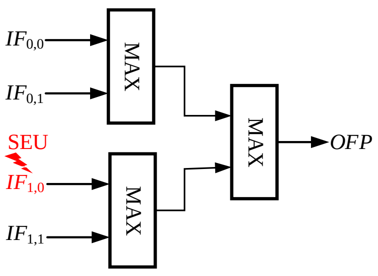
3.1.1. Analysis of Calculations in Pooling Layer

The function of pooling layer is a kind of data downsampling. Taking max-pooling shown in Figure 4a as an example, n is the size of pooling kernel and coordinate $(0,0)$ represents the top-left corner of feature map; $IF_{i,j}$ is a input feature whose coordinate is $(i,\ j)$ in the pooling kernel. The output feature of the pooling kernel OFP is equal to the maximum value in the $n \times n$ pooling kernel, which is described by Equation (5).

$$OFP = \max_{\forall i,j \in \{0,1\ldots,n-1\}} IF_{i,j} \tag{5}$$



(**a**)



(**b**)

Figure 4. Two-dimensional max-pooling operation: (**a**) $n \times n$ kernel size; (**b**) 2×2 kernel size.

A max-pooling with a kernel size of $n \times n$ and a stride of n can compress the feature size by a factor of n^2. Therefore, after the execution of a pooling layer, the mask probability of SEU error that exists in a certain input feature is $\left(n^2-1\right)/n^2$. Figure 4b shows an example of max-pooling with a kernel size of 2×2. If an SEU occurs in $IF_{i,j}$, the mask probability is $3/4$. Obviously, the mask probability will approach 1 with the increase in n.

3.1.2. Analysis of Calculations in Convolutional Layer

Based on Figure 2, the l-th channel output feature of convolutional calculation OFC_l can be described by Equation (6). When the nodes of the convolutional layer, especially the nodes of weights, bias, and input features, are affected by SEUs, erroneous convolutional values are generated in the convolutional layer.

$$OFC_l = \mathrm{ReLU}\left(\sum_{k=0}^{m-1}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} IF_{i,j,k} \times W_{i,j,k,l} + b_l\right) \tag{6}$$

where m denotes the channel size of the input feature map, $IF_{i,j,k}$ represents the feature value located at coordinate $(i,\ j)$ of the k-th channel of the input feature map, $W_{i,j,k,l}$ represents the weight located at coordinates (i, j) determined by the k-th input channel and the l-th output channel, and b_l is the bias of the l-th output channel.

However, it is worth noting that the last operation of convolutional calculation in Equation (6) is ReLU. In actuality, there are three commonly adopted activation functions in the convolutional layer, which are sigmoid, tanh and ReLU. Illustrated by Figure 5, sigmoid and tanh effectively confine the erroneous convolutional values to ranges of [0, 1] and [−1, 1], respectively, regardless of their magnitude. Meanwhile, ReLU restricts negative inputs to 0, while preserving positive inputs in their original scale. Therefore, after the operations of sigmoid and tanh, the errors induced by SEUs will be limited to the ranges of [0, 1] and [−1, 1], respectively. In contrast, the SEU tolerance ability of ReLU against SEUs depends on the sign of input value. When the input value is negative, ReLU is completely immune to SEUs' adverse effect. However, the induced SEU error in positive input will propagate to the next stage without any reduction.

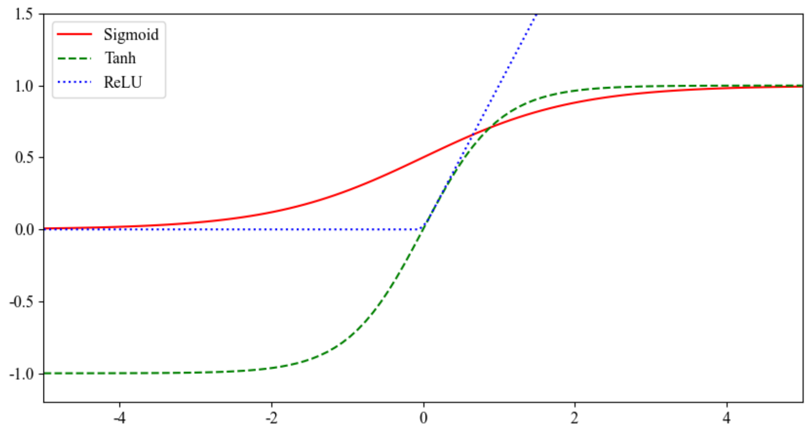


Figure 5. Three common activation functions in CNNs.

3.1.3. Analysis of Calculations across Multiple Layers

Based on the above analysis, it can be deduced easily that activation and max-pooling functions are able to suppress the effect of SEU errors. Since multiple convolutional layers and pooling layers are utilized in CNNs, the inhibitory effect of a CNN on SEU errors should be more and more obvious with the propagation of SEU errors in the CNN, which matches the opinions presented by ref. [5] very well.

However, experimental results show that, under the same BER, SEU errors in the first layer will incur a worse classification accuracy performance of a CNN accelerator compared with SEU errors occurring in later layers of a CNN. This seems to be in conflict with the analysis described in Sections 3.1.1 and 3.1.2, but can be explained from the aspects of circuit scale and execution time.

First, the occurrence probability of SEU errors is proportional to circuit scale. As seen in Figure 1a, the circuit scale of the first layer is much larger than that of the other layers. Therefore, more SEU errors will occur in the first layer.

Second, SEU errors are a recoverable type of soft errors. Therefore, SEU errors succeed in disrupting the operation of CNN only when the hit layer is active. It indicates that for a specific layer, the probability disrupted by SEU errors is proportional to the execution time. As shown in Figure 6, the Conv1 layer, which is the first layer, consumes the longest execution time, accounting for about 80% of total execution time. The execution time of various layers tends to decrease with the decrease in feature map size, without regard to the parallelism mechanism of the CNN accelerator. Therefore, despite the circuit scale, the effective probability of SEU errors occurring during the operation of Conv1 is much higher than that occurring during the operation of other layers.

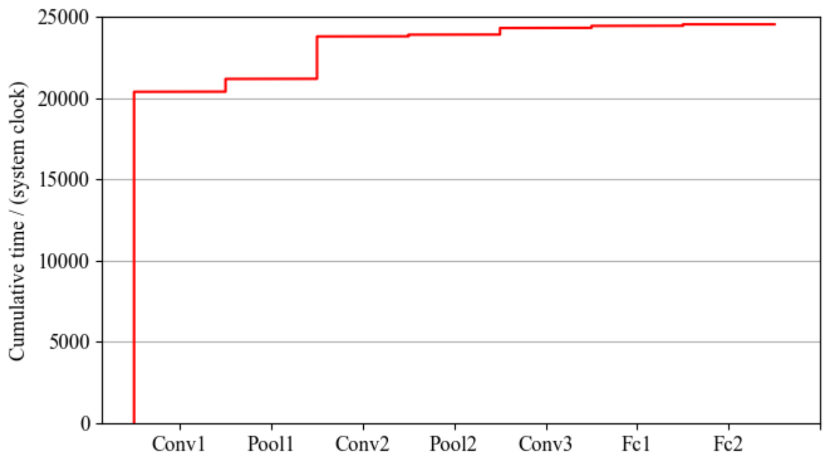


Figure 6. Cumulative time of CNN operation.

3.2. *SEU Tolerance Analysis of CTRL*

The storage cells in the CTRL are mainly divided into two groups, which are state machine registers and data registers. State machine registers are utilized to store state transitions between and within network layers, providing necessary control information for PEs. Most data registers are adopted to generate addresses and control flags for reading/writing MEMs.

Figure 7a describes an example of an FSM operation which is affected by SEU. When the status code becomes corrupted, the state of related PE jumps from Conv1 to Conv3, and the mismatched features and weights are then transferred to the corresponding PE. In addition, based on the current operating status, CTRL will send corresponding flag signals to instruct the execution of corresponding PEs. As shown in Figure 7b, after the pooling operation is carried out, the state should perform pooling operation again. But since the flag of "Pool done" is flipped, the state turns back to the main state directly. After that, the operation of the corresponding PE is disrupted.

3.3. *MSC Identification*

To locate the MSC, the CNN accelerator shown in Figure 1 is taken as the evaluated design. The radiation performance of the CNN accelerator is evaluated through the SEU evaluation platform shown in Section 2.3. As seen in Figure 1, the size of input features is 32×32, the output of Conv1 features includes six channels, the size of each channel is 28×28, and is short for $6@28 \times 28$. The sizes of Pool1, Conv2, Pool2, Conv3, Fc1, and Fc2 are $6@14 \times 14$, $16@10 \times 10$, $16@5 \times 5$, $120@1 \times 1$, 84×1, and 10×1, respectively.

Based on the low probability of SEUs shown in refs. [4,9], we set BER to 10^{-6} and performed independent experiments over 1000 times to the CNN accelerator. The clock frequency was 100 MHz.

Figure 8a shows the fault type statistics of the CNN model view when SEU errors are injected into different layers of the CNN separately. It was found that SEU errors in convolutional layers have a greater impact on classification accuracy than those in pooling and fully connected layers. For instance, SEU errors in Conv1 layer caused a

40.7% reduction in classification accuracy, and 35.3% of these errors accounted for the flips in control logic. In contrast, SEU errors in Fc2 had almost no effect on the classification accuracy of the CNN accelerator. The reason is that to improve the reliability of the CNN accelerator, a local reset operation is triggered before each layer is activated. After then, only SEU errors occurring during Fc2 activation status are able to affect classification accuracy. Meanwhile, the activation time of Fc2 layer is very short, and then the operations of Fc2 layer are seldom disrupted.

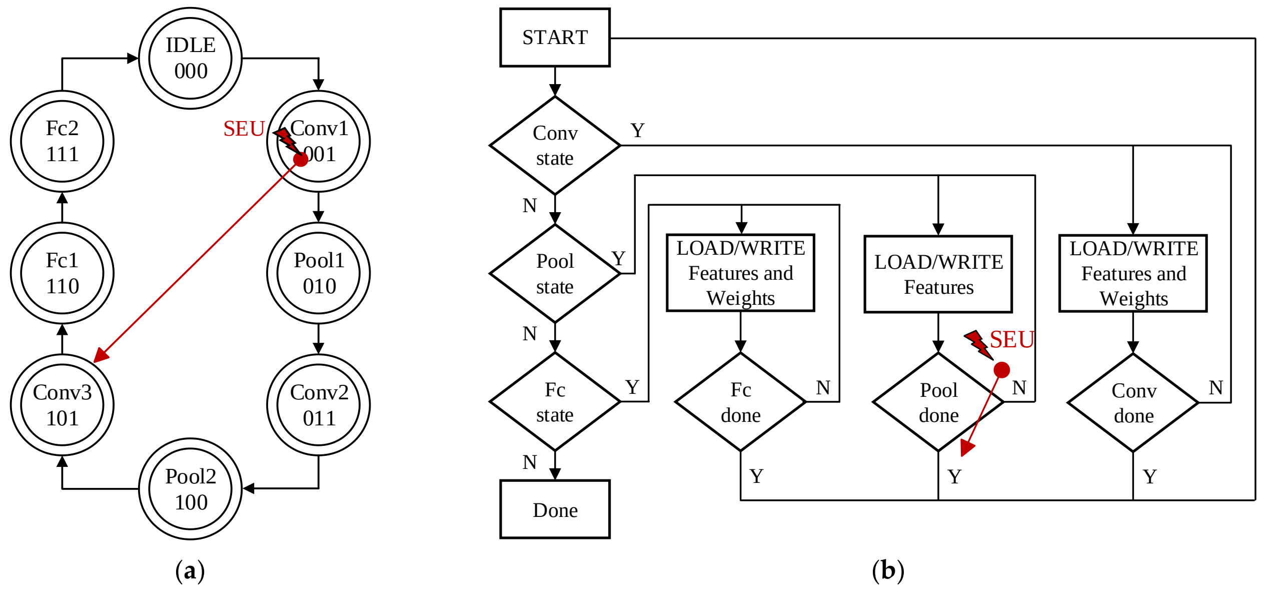


Figure 7. Examples of SEU effects on CTRL: (**a**) Status code error; (**b**) critical bit error.

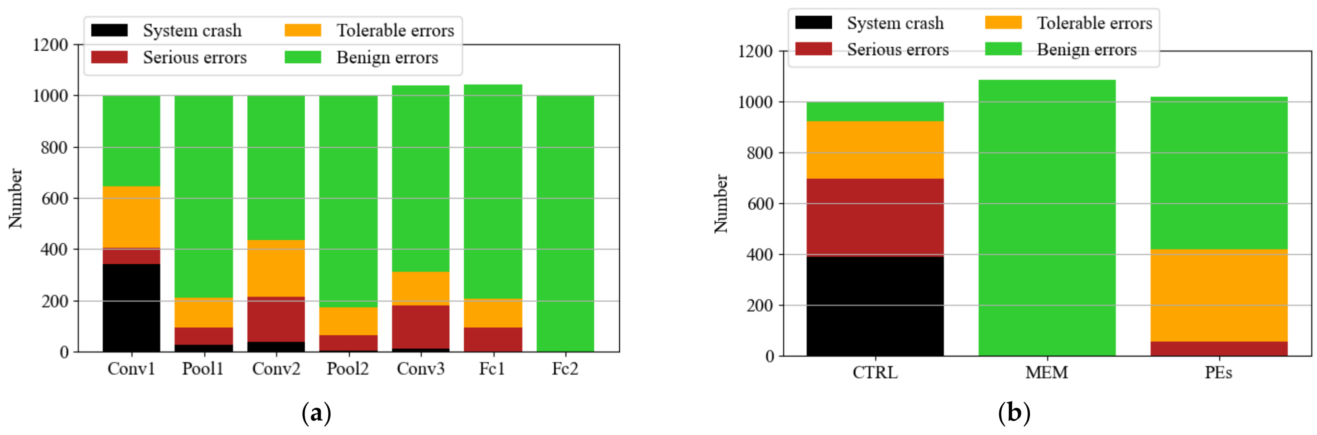


Figure 8. Fault type statistics at BER = 10^{-6}: (**a**) CNN model view; (**b**) architecture view.

The experimental results shown in Figure 8b describe the impact of SEU errors from the architecture view. When the SEU errors occur in the CTRL, the classification accuracy drops sharply, up to 69.5%. However, when SEU errors occur in the MEM and the PE array, the decrease in classification accuracy is much smaller and does not exceed 5.4%.

In conclusion, the SEU effect on convolutional pooling and fully connected layers is inversely proportional to the data flow distance. Moreover, SEU events occurring in the PE array, such as activation functions and pooling functions, incur tolerable errors and benign errors with a high possibility. Therefore, based on the above analysis, it is found that the CTRL is the MSC in the CNN accelerator.

4. Hardening Strategies

To enhance the resilience of the CNN accelerator against SEU errors with low overhead, the FSM error-correcting circuit (FSM-ECC) and TMR automatic-hardening technique (TMR-AHT) are proposed to harden state machine registers and data registers in the CTRL, respectively.

4.1. FSM Error-Correcting Circuit

To detect and correct SEU errors in FSM unit, the FSM-ECC was designed, and the schematic is shown in Figure 9. As seen from Figure 9, the FSM-ECC consists of one FSM, three DFFs, one XOR array, two multiplexers (MUXs), and one ROL-1bit, where the function of the ROL-1bit is shifting the input left by 1 bit cyclically.

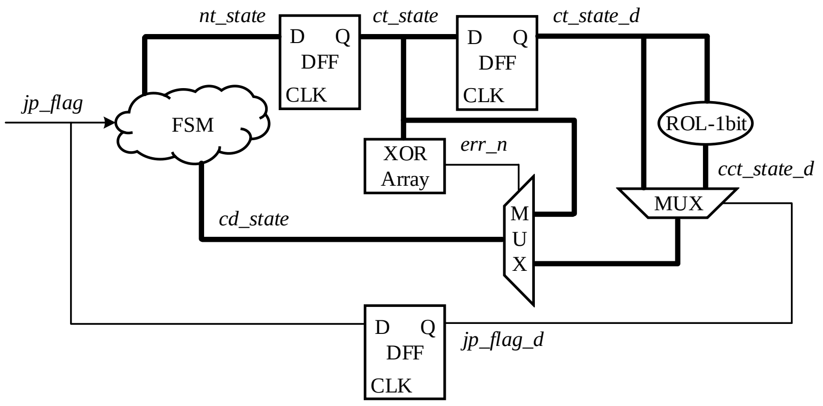


Figure 9. Schematic of FSM-ECC.

One-hot coding is adopted for the state codes, which include next state *nt_state*, current state *ct_state*, delayed current state *ct_state_d*, corrected delayed current state *cct_state_d*, and corrected state *cd_state*. We assume that an FSM comprises n states, collectively forming a set S. Each state is represented by n-bit S_i ($S_i \in S$), as shown in Equation (7).

$$S_i = \{s_i^{n-1}, \cdots s_i^{k+1}, s_i^k, s_i^{k-1}, \cdots s_i^0\}, i, k \in [0, n-1] \tag{7}$$

where s_i^k represents the k-th bit in S_i.

Furthermore, the value of s_i^k is given by Equation (8), where $LF()$ is described by Equation (4).

$$s_i^k = LF(i, k), i, k \in [0, n-1] \tag{8}$$

Assuming that *cd_state* is S_i and jump signal *jp_flag* is active, FSM sends S_{i+1} to *nt_state* if $i \neq n-1$. If $i = n-1$, it indicates that FSM reaches the last state, so S_0 will be assigned to *nt_state*. If *jp_flag* is invalid, *nt_state* keeps the state S_i. With the trigger of system clock, *ct_state* is then updated by *nt_state*.

If one bit of *ct_state* is flipped by SEU, the flipped *ct_state* is no longer any value in the set S. Therefore, there are two logic-1s or no logic-1 in the flipped *ct_state*. According to the operation of XOR array, whose function is given by Equation (9), *err_n* will be active. When the delayed jump signal *jp_flag_d* is active, *cd_state* is assigned to be *cct_state_d*.

$$err\_n = \hat{}ct\_state \tag{9}$$

The timing waveforms in Figure 10 illustrate the operation of FSM-ECC. At $t1$, no registers are affected by SEU, so *err_n* is not asserted. Meanwhile, since *jp_flag* is invalid, the signals of *ct_state*, *ct_state_d*, and *nt_state* are the same. At $t2$, *ct_state* [2] is affected by SEUs and jumps from 0 to 1. Based on the operation of XOR array, *err_n* is asserted. When *jp_flag* is invalid, it means that there is no state transition in the previous clock cycle. Since *ct_state_d* is the delayed *cd_state*, the value of *ct_state_d* is directly reloaded to *cd_state*.

At *t*3, *ct_state* [1] is flipped by SEU, and then *err_n* is asserted. Considering *jp_flag_d* is valid, *ct_state_d* cannot be sent to *cd_state* directly. Instead, *cct_state_d* is transmitted to *cd_state* through two MUXs. At *t*4, an SEU error occurs in *ct_state_d*. However, since *err_n* is inactive, *ct_state* is sent to *cd_state* directly. Therefore, the SEU error does not have any impact on the FSM operation.

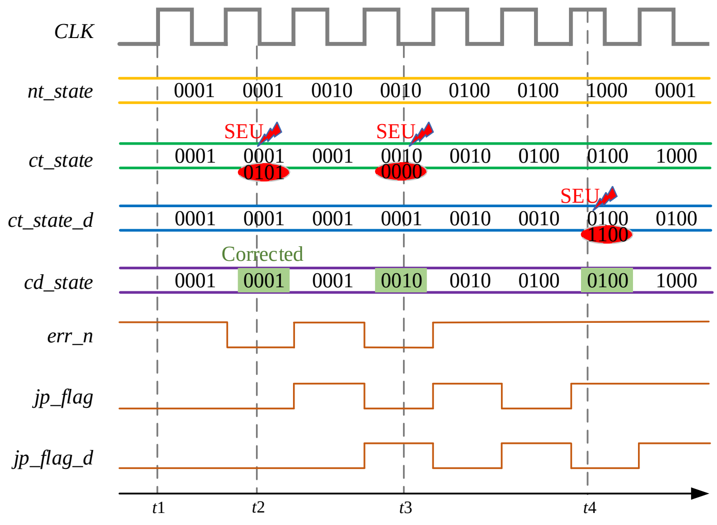


Figure 10. Waveforms of FSM-ECC.

4.2. TMR Automatic-Hardening Technique

In terms of circuit reliability, TMR is a successful solution that sacrifices area to obtain better radiation-reliability performance. Except for the registers of FSM, the CTRL still includes many other registers, where most of them are utilized as counters and control flags. These registers occupy very few hardware resources but play a crucial role in the CNN accelerator. Therefore, hardening these registers with TMR can obviously raise the radiation reliability with minimal hardware overhead.

Because the scale of the CTRL is huge, a hardening script was developed to locate and harden the rest unhardened registers of the CTRL automatically. The flowchart of the proposed script is presented in Figure 11. At first, the script searches all registers with register transfer level (RTL) with the keyword “reg”. However, if there is no clock signal included in the sensitive list of the registers, the corresponding variables with the keyword “reg” will be implemented by combination circuit. These kind of register-type variables will be ignored by the script. After then, the registers in one-hot coding state logic are also ignored, since the registers of FSM are hardened by FSM-ECC. The left register-type variables located in the sequential logic are recorded in a register list. Finally, these recorded registers are hardened automatically by the proposed script.

As one example, shown in Figure 12, the unhardened register is named as DFF, the input of the register is D, and the output of the register is Q. During the hardening operation, the register DFF is copied three times. The three registers are named as DFF0, DFF1, and DFF2, respectively. The inputs of the three registers are all D. The outputs of the three registers are Q0, Q1, and Q2, respectively. After that, a three-input majority voter circuit, which is combinational logic, is added. The three inputs of the majority voter circuit are connected with the outputs of the three registers separately. The output of the majority voter circuit is Q. Since SEU errors are highly unlikely to occur in any two circuits simultaneously, a correct output result can be guaranteed by TMR-AHT.

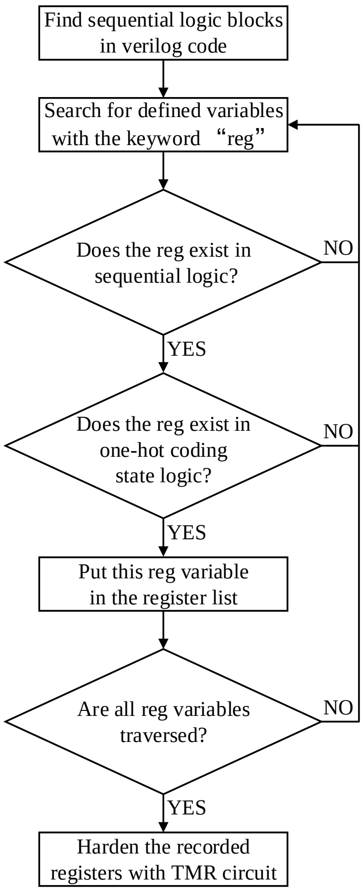

Find sequential logic blocks in verilog code
Search for defined variables with the keyword "reg"
Does the reg exist in sequential logic?
NO
YES
Does the reg exist in one-hot coding state logic?
NO
YES
Put this reg variable in the register list
Are all reg variables traversed?
NO
YES
Harden the recorded registers with TMR circuit

Figure 11. The flowchart of TMR-AHT.

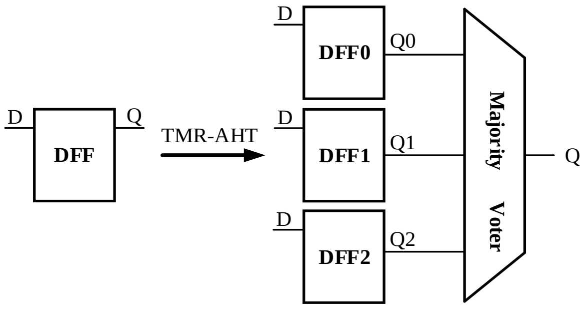


Figure 12. A hardening example with TMR-AHT.

5. Evaluations and Verifications

5.1. Verifications of FSM-ECC and TMR-AHT

To evaluate the reliability performance of our proposed hardening strategies, the evaluation platform presented in Section 2.3 is utilized to perform a large number of experiments on the CNN accelerator shown in Figure 1 with different BER. As shown in Figure 13, when BER is 10^{-7}, only a few registers are disrupted. Due to the layer-by-layer computational architecture and inherent fault-tolerance ability of CNN, both the unhardened and the hardened CNN accelerator have high classification accuracy, and the latter has a classification error rate of less than 0.1%. When BER is 10^{-6}, the classification accuracy of unhardened CNN decreases to 30.36%. However, the hardened CNN has only two classification errors among 1032 independent experiments. When BER is 10^{-5}, the unhardened CNN fails to work completely. However, the hardened CNN still achieves 92.67% classification accuracy.

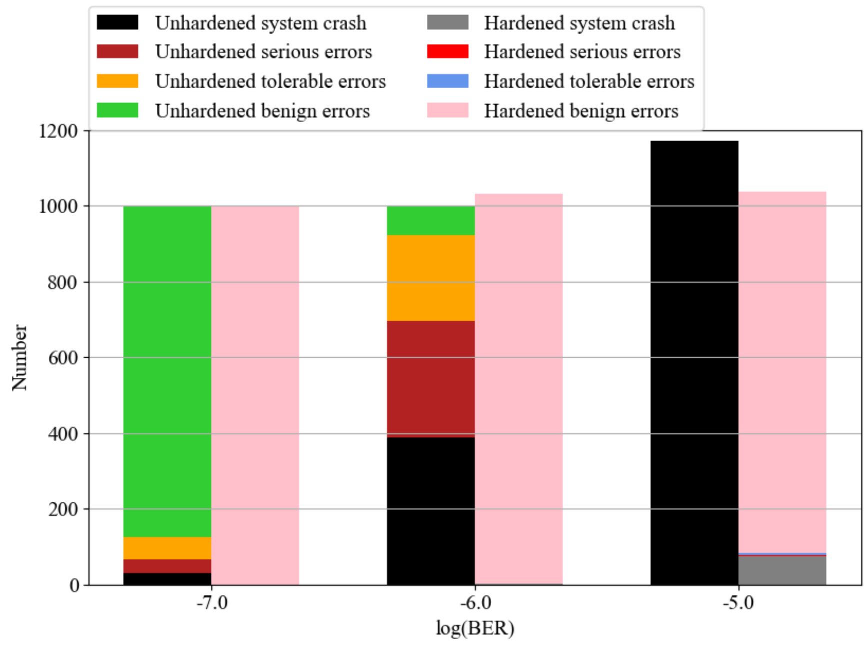


Figure 13. Fault type statistics for hardened and unhardened CNN with different BERs in the CTRL.

5.2. Comparisons of Design Overhead and Fault Tolerance

To make a fair comparison with the proposed hardening strategies, the method proposed in ref. [13] is implemented with our CNN accelerator design. Table 2 shows the utilized resources of unhardened and hardened CNN accelerators on Xilinx Zynq-7035. The clock frequency is 100 MHz.

Table 2. Utilized resources.

	Unhardened	**Ref. [13]**	**Proposed**
LUTs	8040 (100%)	15,475 (192.5%)	9705 (120.7%)
FFs	11,932 (100%)	22,379 (187.6%)	14,427 (120.9%)
DSPs	120 (100%)	120 (100%)	120 (100%)
BRAMs	33 (100%)	33 (100%)	33 (100%)

As seen in Figure 9, FSM-ECC has two long paths. One path includes XOR array and MUX. Another path includes ROL-1bit and two MUXs. The TMR-AHT technique also increases the delay of critical path and reduces the slack.

But it is worth noting that the critical path of the network layer is much longer than the critical path of the CTRL unit. Therefore, the increase in the critical path of the CTRL unit due to the proposed methodology usually has little effect on the maximum frequency of the CNN accelerator.

To compare and verify the effectiveness of the hardened CNN accelerator, fault injection simulations with different BERs were applied to three CNNs, which are unhardened CNN, the CNN hardened by ref. [13], and the proposed hardened CNN. For each BER, the fault injection simulations were performed at least 1000 times, and every simulation time was 24,550 clock cycles, which are the cycles needed by a complete classification task. Table 3 shows the corresponding error probabilities including the total number of serious errors and system crashes, and energy efficiency, which is defined by Equation (10).

$$\mathrm{HF} = \frac{\mathrm{ERR}}{\mathrm{RIR}} \times 100\% \tag{10}$$

where HF is short for hardening efficiency. ERR is the error reduction rate, which is the difference between the error probability of the unhardened CNN and the hardened CNN.

RIR is the resource increase rate, which is the percentage of extra resource consumed by the hardened CNN.

Table 3. Error probability and hardening efficiency at different BERs.

BER	Error Probability (%)			HF (%)	
	Unhardened	**[13]**	**Proposed**	**[13]**	**Proposed**
10^{-8}	0.6 (100%)	0.30 (50%)	0.00 (0%)	0.17 (11.81%)	1.44 (100%)
10^{-7}	10.57 (100%)	10.49 (99.24%)	0.36 (3.41%)	0.04 (0.16%)	24.54 (100%)
2×10^{-7}	20.56 (100%)	19.26 (93.68%)	1.08 (5.25%)	0.72 (1.54%)	46.80 (100%)
10^{-6}	68.89 (100%)	70.51 (102.35%)	5.56 (8.07%)	−0.90 (−0.59%)	152.17 (100%)
2×10^{-6}	86.36 (100%)	82.05 (95.01%)	7.41 (8.58%)	2.40 (1.27%)	189.71 (100%)
10^{-5}	100.0 (100%)	100.0 (100%)	39.13 (39.13%)	0.00 (0%)	146.25 (100%)
2×10^{-5}	100.0 (100%)	100.0 (100%)	72.18 (72.18%)	0.00 (0%)	66.84 (100%)

When BER is 10^{-8}, the error probabilities of the compared three CNNs are all very small, but only the error probability of our hardening CNN is zero. With the increase in BER, error probabilities of the three CNNs all increase. When BER is larger than 10^{-7}, the error probabilities of the unhardened and hardened CNN by ref. [13] rise obviously. Furthermore, there is no significant difference in error probability between the unhardened and hardened CNN by ref. [13], regardless of BER level. It indicates that the hardening strategy presented in ref. [13] is not applicable to typical CNN accelerators. Meanwhile, when BER is 2×10^{-6}, our hardening strategy still works well and achieves an accuracy rate of 92.59%. In comparison, the other two CNNs only achieve an accuracy rate of 13.64% and 17.95%, respectively. The error probability of our hardening CNN increases sharply only when BER is larger than 2×10^{-6}.

In addition, the HF values of CNN hardened by ref. [13] are always nearly zero, and the maximum value of HF for the CNN hardened by ref. [13] is 2.40%. When BER is lower than 10^{-7}, the HF value of our hardened CNN is less than 25%. The reason is that the error probability among this range of BER is much smaller, and the corresponding ERR is also small. When BER rises monotonically in the range of $[10^{-7}, 2 \times 10^{-6}]$, the HF value of our hardening CNN also rises, and reaches the peak point when BER is 2×10^{-6}. It means that our hardening CNN achieves a good tradeoff between radiation reliability and design overhead. The HF value of all hardened methods falls when BER approaches 10^{-5}. Fortunately, research has demonstrated that an SEU is a low-probability event and BER usually does not exceed 10^{-5} [4,9].

The comparison results of five hardened accelerator designs and our design are shown in Table 4, which compares the utilized resources and error probability of each scheme, and also gives the implementation conditions of each scheme, such as fault injection method and harden method. Ref. [7] implemented a CNN accelerator design using the hybrid of binary and FP32 quantization accuracy. Compared to the FP32 implementation, the error probability of this implementation is increased by 12% and the LUT resource consumption is reduced by 31.46%.

Ref. [8] hardened the FP32 CNN accelerator design with FP16 and INT8 quantization, resulting in 47.9% and 76.3% LUTs reduction, 28.6% and 33.3% FFs reduction, and 22% and 72% ERR, respectively.

Ref. [9] reinforced the unhardened FP32 CNN accelerator with full TMR and 8-bit fixed quantization, and the error probabilities were reduced by 33.59% and 40.30%, respectively. Meanwhile, the LUT and FFs used by the full TMR harden method increased by 119.3% and 95.1%, while the LUT and FFs used by the Quant Fix8 harden method decreased by 17.6% and 31.0%, respectively. It can be seen that the quantization method can better harden the CNN accelerator while reducing resource consumption, but too much quantization will lead to obvious precision loss.

Table 4. Utilized resources and error probability.

	Proposed		[7]	[8]		[9]		[10]	[13]	[16]
CNN Model	Lenet-5		Lenet-4	Lenet-4		ZynqNet		MLP-3	Lenet-4	MLP-3
Platform	XC7Z035		XCZU9EG	ZYNQ		XC7Z020		Software	XCZU9EG	Software
Baseline	Unhardened Fix W8A16		FP32	FP32		Unhardened FP32		Unhardened Fix 32	Unhardened FP32	Unhardened FP32
Precision	Fix W8A16		Binary and FP32	FP16	INT8	-		Fix 4	-	-
Fault Injection Method	Software		Hardware	Hardware		Hardware		Software	Hardware	Hardware
Harden Method	FSM-ECC & TMR-AHT		Quant	Quant		Full TMR	Quant Fix 8	Quant	Selective TMR	SMART+ TMR
LUTs RIR	20.7%		−31.46%	−47.9%	−76.3%	119.3%	−17.6%	-	6.74%	-
FFs RIR	20.9%		0%	−28.6%	−33.3%	95.1%	−31.0%	-	0%	-
BER	10^{-7}	10^{-6}	-	-		-		10^{-1}	-	-
ERR	96.59%	91.93%	−12%	22%	72%	33.59%	40.30%	39.96%	14%	55.52%

Ref. [13] used the selective TMR technique to harden the MNIST CNN accelerator. Compared to the unreinforced model, selective TMR reduced the probability of critical error by 14% with a marginal overhead of only 8%.

Refs. [10,16] utilized Quant, SMART, TMR, and SMART+TMR, respectively, to harden the MLP, and the error probability was decreased by 39.96%, 43.84%, 53.97%, and 55.52%, respectively.

Compared with the above designs, the harden method proposed in this paper can better harden the CNN accelerator while increasing fewer resources, and can ensure that the error probability is reduced by more than 90%.

6. Conclusions and Future Works

This paper presents a hybrid hardening strategy consisting of a finite state machine error-correcting circuit (FSM-ECC) and a triple modular redundancy automatic-hardening technique (TMR-AHT). The proposed methodology can be applied to generic SRAM-FPGA, and even can be extended to the hardened application of ASIC digital design. Moreover, the workload of the proposed methodology is very small. FSM-ECC can harden any kind of FSM, and the workload only involves checking the width of the state register. The other registers in the CTRL unit are replaced by TMR registers with TMR-AHT automatically.

To validate the effect of the proposed hybrid hardening strategy, a CNN accelerator was hardened and implemented with an SRAM-based FPGA. Our experiments reveal that CTRL is the least resilient to SEUs compared to other layers and components. Our FSM-ECC and TMR-AHT mapping to CTRL is an optimal scheme for CNN accelerator design. When BER is 2×10^{-6}, our hardening strategy reduces the CNN error rate by 78.95%, and the hardening efficiency is as high as 189.71%.

In the future, we will continue to research the low-overhead hardened techniques of CNN accelerators, such as double modular redundancy (DMR) or selective TMR.

Author Contributions: Conceptualization, X.C. and K.C.; methodology, X.C. and K.C.; software, Y.X., L.H. and K.C.; validation, Y.X. and K.C.; formal analysis, Y.X. and K.C.; investigation, Y.X. and K.C.; resources, Y.X., L.H. and K.C.; data curation, Y.X. and K.C.; writing—original draft preparation, X.C., Y.X., K.C., C.G., Z.X. and H.Y.; writing—review and editing, X.C., Y.Z., X.W. and Y.G.; visualization, Y.X., L.H. and C.G.; supervision, X.C., X.W. and Y.G.; project administration, X.C.; funding acquisition, X.C., X.W. and Y.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research and the APC were funded by the National Defense Science and Technology Key Laboratory, grant number 61425010302.

Data Availability Statement: Dataset available on request from the authors.

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

References

1. Li, Q.; Cai, W.; Wang, X.; Zhou, Y.; Feng, D.D.; Chen, M. Medical image classification with convolutional neural network. In Proceedings of the 2014 13th International Conference on Control Automation Robotics & Vision (ICARCV), Singapore, 10–12 December 2014; pp. 844–848. [CrossRef]
2. Yang, X.; Wu, T.; Wang, N.; Huang, Y.; Song, B.; Gao, X. HCNN-PSI: A hybrid CNN with partial semantic information for space target recognition. *Pattern Recognit.* **2020**, *108*, 107531. [CrossRef]
3. Priyadarshini, I.; Puri, V. Mars weather data analysis using machine learning techniques. *Earth Sci. Inform.* **2021**, *14*, 1885–1898. [CrossRef]
4. Kain, E.T.; Lovelly, T.M.; George, A.D. Evaluating SEU Resilience of CNNs with Fault Injection. In Proceedings of the 2020 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 22–24 September 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–5.
5. Lopes, I.C.; Kastensmidt, F.L.; Susin, A.A. SEU susceptibility analysis of a feedforward neural network implemented in a SRAM-based FPGA. In Proceedings of the 2017 18th IEEE Latin American Test Symposium (LATS), Bogota, Colombia, 13–15 March 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6.
6. Li, W.; Ge, G.; Guo, K.; Chen, X.; Wei, Q.; Gao, Z.; Wang, Y.; Yang, H. Soft error mitigation for deep convolution neural network on FPGA accelerators. In Proceedings of the 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), Genova, Italy, 31 August–2 September 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–5.
7. Libano, F.; Wilson, B.; Wirthlin, M.; Rech, P.; Brunhaver, J. Understanding the impact of quantization, accuracy, and radiation on the reliability of convolutional neural networks on FPGAs. *IEEE Trans. Nucl. Sci.* **2020**, *67*, 1478–1484. [CrossRef]
8. Libano, F.; Rech, P.; Neuman, B.; Leavitt, J.; Wirthlin, M.; Brunhaver, J. How reduced data precision and degree of parallelism impact the reliability of convolutional neural networks on FPGAs. *IEEE Trans. Nucl. Sci.* **2021**, *68*, 865–872. [CrossRef]
9. Wang, H.B.; Wang, Y.S.; Xiao, J.H.; Wang, S.L.; Liang, T.J. Impact of single-event upsets on convolutional neural networks in Xilinx Zynq FPGAs. *IEEE Trans. Nucl. Sci.* **2021**, *68*, 394–401. [CrossRef]
10. Syed, R.T.; Ulbricht, M.; Piotrowski, K.; Krstic, M. Fault resilience analysis of quantized deep neural networks. In Proceedings of the 2021 IEEE 32nd International Conference on Microelectronics (MIEL), Nis, Serbia, 12–14 September 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 275–279.
11. Du, B.; Azimi, S.; De Sio, C.; Bozzoli, L.; Sterpone, L. On the reliability of convolutional neural network implementation on SRAM-based FPGA. In Proceedings of the 2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Noordwijk, The Netherlands, 2–4 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
12. Liu, C.; Chu, C.; Xu, D.; Wang, Y.; Wang, Q.; Li, H.; Li, X.; Cheng, K.T. HyCA: A hybrid computing architecture for fault-tolerant deep learning. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2021**, *41*, 3400–3413. [CrossRef]
13. Libano, F.; Wilson, B.; Anderson, J.; Wirthlin, M.J.; Cazzaniga, C.; Frost, C.; Rech, P. Selective hardening for neural networks in FPGAs. *IEEE Trans. Nucl. Sci.* **2018**, *66*, 216–222. [CrossRef]
14. Gao, Z.; Zhang, H.; Yao, Y.; Xiao, J.; Zeng, S.; Ge, G.; Wang, Y.; Ullah, A.; Reviriego, P. Soft error tolerant convolutional neural networks on FPGAs with ensemble learning. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2022**, *30*, 291–302. [CrossRef]
15. Dos Santos, F.F.; Draghetti, L.; Weigel, L.; Carro, L.; Navaux, P.; Rech, P. Evaluation and mitigation of soft-errors in neural network-based object detection in three GPU architectures. In Proceedings of the 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), Denver, CO, USA, 26–29 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 169–176.
16. Rajappa, A.J.; Reiter, P.; Sartori, T.K.S.; Laurini, L.H.; Fourati, H.; Mercelis, S.; Hellinckx, P.; Bastos, R.P. SMART: Selective MAC zero-optimzation for neural network reliability under radiation. In Proceedings of the 34th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis (ESREF), Toulouse, France, 2–5 October 2023.
17. Xia, L.; Liu, M.; Ning, X.; Chakrabarty, K.; Wang, Y. Fault-tolerant training enabled by on-line fault detection for RRAM-based neural computing systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *38*, 1611–1624. [CrossRef]
18. Schorn, C.; Guntoro, A.; Ascheid, G. An efficient bit-flip resilience optimization method for deep neural networks. In Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1507–1512.
19. Chen, K.; Chen, X.; Zhang, Y.; Zhang, Z. A rapid evaluation technology for SEU in convolutional neural network circuits. In Proceedings of the 2021 IEEE 3rd International Conference on Circuits and Systems (ICCS), Chengdu, China, 29–31 October 2021; pp. 19–23.
20. Chen, X.; Huo, L.; Xie, Y.; Shen, Z.; Xiang, Z.; Gao, C.; Zhang, Y. FPGA-Based Cross-Hardware MBU Emulation Platform for Layout-Level Digital VLSI. In Proceedings of the 2023 IEEE 32nd Asian Test Symposium (ATS), Beijing, China, 14–17 October 2023; pp. 1–6.
21. Lu, Y.; Chen, X.; Zhai, X.; Saha, S.; Ehsan, S.; Su, J.; McDonald-Maier, K. A fast simulation method for analysis of SEE in VLSI. *Microelectron. Reliab.* **2021**, *120*, 114110. [CrossRef]