information

Article

Efficient Dynamic Reconfigurable CNN Accelerator for Edge Intelligence Computing on FPGA

Kaisheng Shi [1], Mingwei Wang [1,*], Xin Tan [1], Qianghua Li [2] and Tao Lei [1,3]

[1] College of Electronic Information and Artificial Intelligence, Shaanxi University of Science and Technology, Xi'an 710021, China
[2] College of Electrical and Control Engineering, Shaanxi University of Science and Technology, Xi'an 710021, China
[3] Shaanxi Joint Laboratory of Artificial Intelligence, Shaanxi University of Science and Technology, Xi'an 710021, China
* Correspondence: wangmingwei@sust.edu.cn

Abstract: This paper proposes an efficient dynamic reconfigurable CNN accelerator (EDRCA) for FPGAs to tackle the issues of limited hardware resources and low energy efficiency in the deployment of convolutional neural networks on embedded edge computing devices. First, a configuration layer sequence optimization method is proposed to minimize the configuration time overhead and improve performance. Second, accelerator templates for dynamic regions are designed to create a unified high-speed interface and enhance operational performance. The dynamic reconfigurable technology is applied on the Xilinx KV260 FPGA platform to design the EDRCA accelerator, resolving the hardware resource constraints in traditional accelerator design. The YOLOV2-TINY object detection network is used to test the EDRCA accelerator on the Xilinx KV260 platform using floating point data. Results at 250 MHz show a computing performance of 75.1929 GOPS, peak power consumption of 5.25 W, and power efficiency of 13.6219 GOPS/W, indicating the potential of the EDRCA accelerator for edge intelligence computing.

Keywords: FPGA; CNN; dynamic reconfiguration; hardware accelerator; target detection

Citation: Shi, K.; Wang, M.; Tan, X.; Li, Q.; Lei, T. Efficient Dynamic Reconfigurable CNN Accelerator for Edge Intelligence Computing on FPGA. *Information* **2023**, *14*, 194. https://doi.org/10.3390/info14030194

Academic Editors: Lorenzo Carnevale and Massimo Villari

Received: 3 February 2023
Revised: 8 March 2023
Accepted: 13 March 2023
Published: 20 March 2023

1. Introduction

The field of machine vision has seen the growth of convolutional neural network (CNN) algorithms, leading to the development of new models like LeNet, AlexNet, R-CNN series, YOLO series, and SSD series [1–5], which are widely used in tasks such as image classification, target detection, and biological feature recognition [6–8]. As big data drives the need for advanced algorithmic functions, CNN networks have become larger and more complex, posing a challenge to current hardware systems in terms of computational performance. The traditional approach of improving performance through additional hardware devices has reached a bottleneck due to hardware limitations [9]. To meet this challenge, researchers have turned to dynamically reconfigurable FPGA systems which offer improved resource utilization, performance, and the ability to dynamically combine different hardware processing architectures during operation [10].

In recent years, the research of FPGA-based convolutional neural network accelerators has been split into two main areas: model compression and efficient hardware architecture design. Model compression aims to maintain accuracy while reducing the hardware requirements for computation and storage. Techniques used for model compression include model pruning [11], knowledge distillation [12], and model quantization [13]. However, these methods only provide limited improvement for deploying algorithmic models on FPGAs. On the other hand, efficient hardware architecture design is divided into general-purpose and dedicated architectures. For general purposes, the Caffine framework proposed by

ZHANG et al. [14] accelerates CNN model inference using shrinkage variable processing elements, but requires significant hardware resources, making FPGA deployment expensive. SHARMA et al. [15], proposed DnnWeaver architecture that uses a set of coarse processing units to perform convolutional operations and communication with DRAM through a standard AXI bus. The architecture was verified with a 16-bit fixed point precision VGG-16 model. WANG et al. [16] proposed a DPU-based YOLOv3 acceleration architecture using the Vitis AI development tool, which has a rich deep learning development foundation, supports fast development processes, and provides convenient development for accelerating the YOLOv3 inference network. The architecture was deployed and evaluated on the ZCU104 evaluation board, achieving good detection frame rates, but with an energy efficiency of only 2.47 GOPS/W, and it lacks flexibility in handling diverse tasks. Dedicated architectures include the reconfigurable ARM+FPGA-based CNN accelerator proposed by ZHANG et al. for the YOLOV2-TINY model [17], which reduces data moves and off-chip storage access through time-configured computing layers and 16-bit quantization. WANG et al. [18], proposed a Sparse-YOLO architecture that optimizes the YOLOV2 model using a sparse convolution method, generating parallel computation soft cores with OpenCL and deploying the model through the CPU. This architecture achieves 61.9 fps detection speed with 26 W operating power but sacrifices energy efficiency.

A high-performance dynamic reconfigurable CNN accelerator (EDRCA) architecture based on FPGAs is proposed to address the problems of limited hardware resources and low energy efficiency in existing studies. The EDRCA uses dynamic reconfiguration to speed up convolutional neural network inference and optimizes layer sequences for better system design. The accelerator also has a reconfigurable module interface for improved performance and is tested using the YOLOV2-TINY target detection model on a Xilinx KV260 FPGA platform, demonstrating its high engineering value.

The rest of this paper is organized as follows: In Section 2, the convolutional neural network model YOLOV2-Tiny and dynamic reconfigurable techniques are presented. In Section 3, the dynamically reconfigurable CNN accelerator (EDRCA) architecture is proposed, and its hardware design and design methodology are described in detail. In Section 4, the EDRCA architecture is deployed on a Xilinx KV260 FPGA hardware platform and the YOLOV2-Tiny algorithm is used as the deployment algorithm to obtain experimental results and compare the hardware performance with different accelerator synthesized from related studies. In Section 5, it concludes with a short summary containing conclusions and discussion of possible future research work.

2. CNN Model and Dynamic Reconfigurable Technology

2.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of deep-structured feedforward artificial neural network that can process large amounts of pixel data from 3D images and demonstrate superior scalability and feature extraction compared to traditional artificial neural networks [19]. As a result, they are widely used in machine vision. CNNs typically consist of several operational layers, including pooling layers, convolutional layers, fully connected layers, and activation layers [20]. This paper evaluates the feasibility of the EDRCA architecture by using YOLOV2-Tiny, a classic CNN architecture, and the VOC2012 public dataset [21] as the training and validation data.

YOLOV2-Tiny is a one-stage target detection network based on CNN that employs the anchor boxes method of Faster R-CNN to optimally predict the classification and localization of candidate boxes by generating a series of predefined boxes at different scales and proportions to serve as candidate target boxes, with a 7% improvement in recall compared to YOLOV1. To maintain spatial and location information, the fully connected layer in YOLOV1 has been removed. The last pooling layer was also removed to improve the resolution of the output feature map, and the input image size of 448 $\times$ 448 in YOLOv1 was changed to 416 $\times$ 416, ensuring that the pixels of the feature map width and height is odd, resulting in a centralized network. The network structure of YOLOV2-

Tiny, which consists of 9 convolution layers and 6 max-pooling layers, is illustrated in Table 1. Compared to YOLOV2, which has 23 convolutional layers and 5 max-pooling layers, YOLOV2-Tiny simplifies the network structure and significantly reduces the number of network parameters, making it ideal for deployment on FPGA platforms.

Table 1. The network structure of Yolov2-Tiny.

Name	Number of Convolution Kernels	Convolution Kernel Size/Step/Fill	Input Feature Maps	Input Feature Maps
Conv1	16	3 × 3/1/1	416 × 416 × 3	416 × 416 × 16
Pool1		2 × 2/2/0	416 × 416 × 16	208 × 228 × 16
Conv2	32	3 × 3/1/1	208 × 208 × 16	208 × 208 × 32
Pool2		2 × 2/2/0	208 × 208 × 32	104 × 104 × 32
Conv3	64	3 × 3/1/1	104 × 104 × 32	104 × 104 × 64
Pool3		2 × 2/2/0	52 × 52 × 64	52 × 52 × 128
Conv4	128	3 × 3/1/1	52 × 52 × 128	26 × 26 × 128
Pool4		2 × 2/2/0	26 × 26 × 128	26 × 26 × 256
Conv5	256	3 × 3/1/1	26 × 26 × 256	13 × 13 × 256
Pool5		2 × 2/2/0	13 × 13 × 256	13 × 13 × 256
Conv6	512	3 × 3/1/1	13 × 13 × 256	13 × 13 × 512
Pool6		2 × 2/2/0	13 × 13 × 512	13 × 13 × 512
Conv7	1024	3 × 3/1/1	13 × 13 × 512	13 × 13 × 1024
Conv8	1024	3 × 3/1/1	13 × 13 × 1024	13 × 13 × 1024
Conv9	125	1 × 1/1/0	13 × 13 × 1024	13 × 13 × 125

2.2. Dynamic Reconfigurable Technology for FPGAs

The concept of dynamic reconfigurable technology [22] was first introduced to enhance the logic capacity and reduce reconfiguration time. With the advent of FPGAs by XILINX in 1984, the main challenge faced in implementing large designs was the limited resources available on FPGAs. Developers solved this issue by creating a prototype architecture that utilized dynamic reconfigurable FPGAs, which increased the number of profiles and thus, the logic capacity [23]. Over time, hardware advancements improved the availability of resources and researchers shifted their focus to other aspects such as scalability of dynamically reconfigurable systems, maintaining communication during reconfiguration, and the system's ability to adapt to various computational tasks.

Dynamic reconfiguration in FPGAs can be categorized into full and partial reconfiguration [24]. Full dynamic reconfiguration involves altering the functionality of an FPGA by reconfiguring all its hardware resources, while partial dynamic reconfiguration (DPR) divides the FPGA into a static logic area and a dynamic logic area. The dynamic logic area, also referred to as the partially reconfigurable partition, can have its circuit structure modified by loading and unloading different hardware profiles. During real-world implementation, DPR enables modifications to running FPGA applications through loading dynamic configuration files, without disrupting or affecting the running applications in the static logic area. Concept of design for dynamic partial reconfiguration is depicted in Figure 1, which separates the FPGA into two reconfigurable regions, A and B, each with adequate resource space. Reconfigurable modules A1, A2, B1 and B2 can be dynamically configured from the bitstream library during system operation [25], enabling the dynamic combination of hardware systems to form various hardware architectures without interrupting the system. This also effectively resolves the challenge of limited FPGA hardware resources.

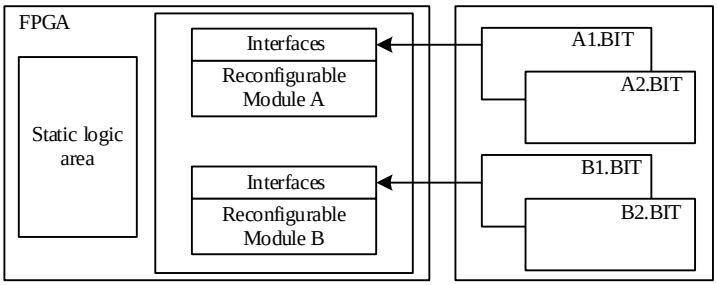

Figure 1. Concept of design for dynamic partial reconfiguration.

3. Design of Dynamic Reconfigurable CNN Accelerator

3.1. Hardware Design

The EDRCA architecture is a dynamic reconfigurable accelerator for convolutional neural networks, as shown in Figure 2. The entire hardware system is deployed on the Xilinx KV260 FPGA hardware platform. The EDRCA accelerator is divided into FPGA and ARM ends. Its ARM controller uses an ARM Cortex-A7 architecture processor, specifically the Allwinner A20 model, which is responsible for controlling, configuring, and completing data transmission with the FPGA through the AXI communication bus. The memory device used is 4 GB of DDR4, which provides ample cache space for the entire hardware system. Communication between the FPGA side and the ARM side is done through the 128-bit AXI4 bus protocol, using the M_AXI_HPM0_FPD, M_AXI_HPM0_LPD, and S_AXI_HPC0_FPD high-performance interfaces. As for the FPGA end, it is divided into three parts: dynamic area 1, dynamic area 2, and static area. The static region mainly handles the management of interrupt, clock, and reset signals, as well as logic decoupling for the dynamic regions. In signal management, the system utilizes the Clocking Wizard IP for clock signal management and the AXI Interrupt Controller IP for interrupt signal management. When an interrupt signal is triggered, the system saves the current hardware system state and transfers control to the interrupt handler, which is responsible for handling the interrupt event. The system then restores the previous program state so that the program can continue executing before the interrupt occurred. On the DDR4, 256 MB of address space is allocated to each dynamic region to meet the data dumping needs. Three 250 MHz clock signals are established in the EDRCA to drive the static region, dynamic region 1, and dynamic region 2, respectively. The reset signal is divided into static and dynamic areas, with the static area's clock reset signal synchronized by the processor reset module and the dynamic area's clock reset signal managed by the Xilinx's SIHA Manager IP core. To ensure that the accelerator operates smoothly, a Decoupler module is used for logic decoupling between reconfigurable and static modules, and the DFX_Shutdown Manager module manages the AXI4-Lite and AXI4 bus interfaces to provide a safe environment for the accelerator.

In order to ensure stability and high bandwidth in the EDRCA accelerator, the reconfigurable module in the dynamic region is designed with a fixed signal interface and register address. A dedicated accelerator template, created using High-Level Synthesis (HLS) on the VIVADO HLS development tool, provides a uniform interface for each computation layer. To achieve efficient sequential data transfer, the hardware interface specification employs the m_AXI communication protocol with a 32-bit address width and burst transfer mode, achieving a maximum operational bandwidth of 17GB/s. The YOLOV2-TINY model consists of pooling and convolutional activation layers, with the latter accounting for 76.31% of the model. Figure 3 illustrates the calculation method of the convolutional layer. First, two input feature maps are used as inputs and fed into the convolution activation accelerator through a buffer. Each input feature map is convolved with a $H \times W$ convolutional kernel moving window. The parallel computation of the convolution activation layer is designed in Algorithm 1, aiming to improve its performance through pipelining and full parallelization operations. The hardware design of the layer was created using High-Level synthesis (HLS), then converted to Verilog by the VIVADO HLS development tool, and

finally, the operation layer accelerator IP core was generated. The optimal sequence of layers with minimum time overhead was determined through the optimal configuration sequence method, and the reconfigurable module in the dynamic region was constructed with a focus on minimizing configuration times.

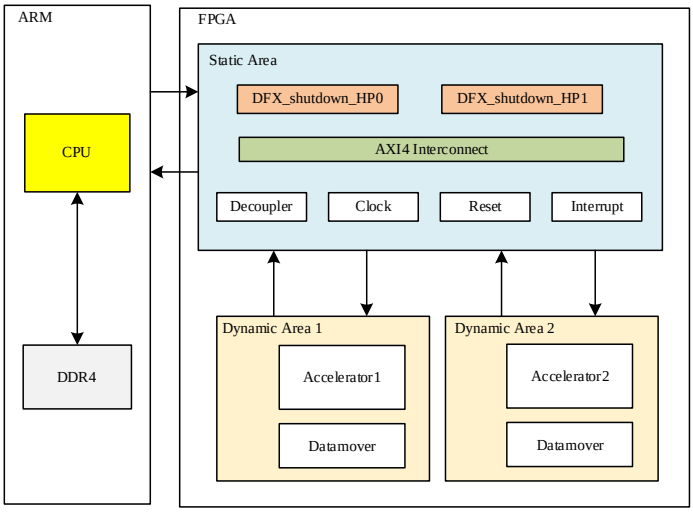

Figure 2. Overall architecture of CNN accelerator.

Algorithm 1. Convolutional activation unit parallel acceleration algorithm pseudocode

```
INPUT InputFeature[too][trr][tcc],
Weighs[Tcout][Tcin][H][W];
OUTPUT OutFeature[Tcout][Tcout][Tyout];
1.FOR(i = 0; i < H; i++){
2. FOR(j = 0; j < W; j++){
3. FOR(trr = row; trr < min(row + Tr,R); trr++){
4. FOR(tcc = col; tcc < min(col + Tm,C); tcc++){
5.#PRAGMA HLS PIPELINE // Pipeline
6. FOR(too = to; too < min(to + Tn,M); tii++){
7.#PRAGMA HLS UNROLL // Full Parallelization Operations
8. FOR(tii = ti; tii < min(ti + Tn,N); tii++){
9.#PRAGMA HLS UNROLL // Full Parallelization Operations
10.LOOP: OutFeature[too][trr][tcc] +=\
11. Weights[too][tii][i][j] * InputFeature [tii][S*trr+i][S*tcc+j];
13.}}}}}}
```

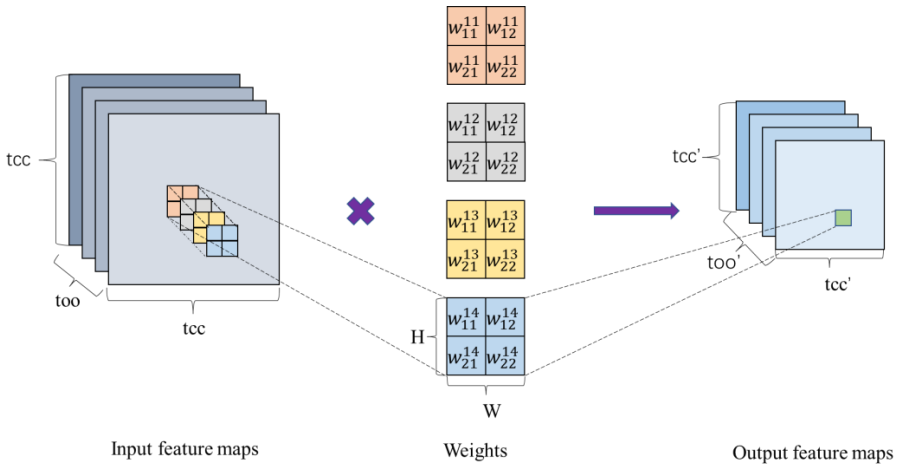

Figure 3. The calculation method of a convolutional layer.

3.2. Reconfigurable Modules

The reconfigurable module (RM) in Figure 4 has two buses for communication—a 128-bit M_AXI data bus and a 32-bit S_AXI_Control command bus. Both buses connect to DDR4 through the 128-bit system bus, allowing the internal accelerator group of the RM to be accessed and activated via address register. The RM operates at a reference clock frequency of 250 MHz and is supplied with the control clock from the hardware platform's clock through the internal PLL. The system's reset signal is triggered by the reset button outside the FPGA. Upon power-on, the hardware platform initializes the DDR4 controller and sends a reset signal to the reset module, putting the RM and the FPGA's peripherals in a ready state.

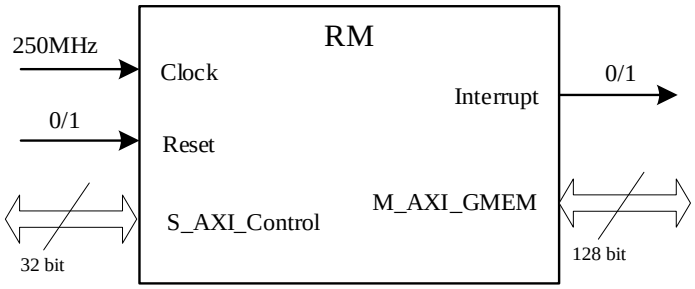

Figure 4. Top-level design interface specification of the single reconfigurable module group.

The EDRCA accelerator's reconfigurable module can be equipped with varying specifications of convolutional and pooling operation accelerators based on the computational task at hand. To improve integration of the RM, the convolutional operation and activation function layers are combined into a single convolutional activation layer. When assigning reconfigurable modules to different reconfiguration regions, the number of hardware resources they occupy should be as close as possible to optimize their computational performance and minimize hardware waste. Figure 4 shows the top-level design interface specification of the reconfigurable module template. The accelerator system loads a configuration file to trigger the execution process after grouping the computing layer accelerators, and a Decoupler module is used in the static region to prevent logic overlap between the dynamic logic region and the reconfigurable module. The Decoupler module decouples the logic of the reconfigurable module in the static area.

3.3. Optimized Configuration Sequence Method

In designing the EDRCA accelerator, the time overhead of reconfiguration must also be taken into account. When implementing a large-scale target detection algorithm, the system may need to undergo multiple reconfigurations, which can severely impact its operational efficiency. To mitigate this, the design of the dynamic reconfiguration region must be optimized. This can be achieved by minimizing the number of reconfigurations through optimizing the configuration sequence of the reconfigurable accelerator in the dynamic region to produce a faster accelerator.

The dynamically reconfigurable system has a considerable optimizable execution time overhead that is caused by the latency in using accelerators with limited computational resources and slower operation speed, as well as the overhead from multiple reconfiguration files. The reconfiguration file's execution time overhead depends on the size of the bitstream file and the bandwidth of the configuration port. Hence, optimizing the configuration sequence involves reducing the number of reconfigurations and assigning more computational resources. This approach optimizes the configuration sequence by identifying the optimal sequence of layer operator profiles with the lowest execution time overhead, allocating more computational bandwidth to each layer operator, and reducing the reload time overhead in the full sequence, thus improving the operation speed of dynamic reconfigurable accelerators.

The optimization of the configuration sequence is broken down into two parts: layer clustering and layer sequence. As shown in Algorithm 2, layer clustering involves using a clustering algorithm to classify layer accelerators with low execution time overhead. The computing accelerators of pre-defined CNN layers are divided into several sets (Dj) based on their types, and the execution time for each accelerator (X) in the set is measured. The fastest accelerator with the least execution time overhead (Tx) is identified, and all other layers are grouped into the same set (Ci) based on the same configuration as X. This process is repeated until all layers are clustered into their respective sets. The EDRCA accelerator can simultaneously reconfigure multiple computing layer accelerators in each of the two dynamic reconfiguration regions. The layer sequence involves analyzing the YOLOV2-Tiny CNN model and combining the accelerators in each set (C) into one or more profiles, creating an optimal layer sequence with minimal time overhead and minimum reconfiguration frequency based on the CNN architecture.

Algorithm 2: Layer Clustering Pseudocode

Input: Set of layers, D = {X1, X2, X3, ..., Xm}
Output: Set of classes, C = {C1, C2, C3, ..., Ci}
1. Group the layers of the same type into set D
2. Set class count, I = 1
3. REPEAT
4. Find the fastest layer X in set D
5. Place X into a new set, Ci
6. Calculate the execution time overhead of other layers using the same configuration as X
7. IF the redundant time overhead is less than the reconfiguration time overhead THEN
8. Add the layer to set Ci
9. END IF
10. IF all other layers have been processed THEN
11. Remove set Ci from the total set C
12. END IF
13. Increment class count, i++
14. UNTIL the current set D is empty
15. Return set Ci

4. Experiments and Results

4.1. Experimental Settings

The performance of the EDRCA accelerator is evaluated using the Xilinx KV260 FPGA platform (Figure 5) as the experimental hardware. The YOLOV2-TINY model is deployed and runs on the platform. The XCK26-SFVC784 chip model, with 4 GB DDR4 SDRAM memory, operates the accelerator at a frequency of 250 MHZ. The experiment utilizes AMD-XILINX's development tools VIVADO (2022.1) and VIVADO HLS (2022.1) for engineering design and hardware module design, with comprehensive verification performed on the FPGA side. The PETALINUX operating platform is deployed, using the PYNQ-Kira framework for soft and hard co-design. The generated configuration file is loaded onto the Xilinx KV260 development board, and YOLOV2-TINY uses the VOC2012 public dataset as the experimental data.

4.2. Experimental Flow

To implement the EDRCA accelerator application YOLOV2-Tiny, the steps are as follows, as shown in Figure 6: first, the YOLOV2-TINY model is built using the Tensorflow framework and the model parameters are obtained by training on the VOC2012 dataset with an input image resolution of 416 × 416. The model parameters are saved as binary files with floating-point data. Next, the optimal configuration sequence method is used to generate the optimal layer sequence with minimum time overhead, resulting in a reconfigurable module configuration file with the minimum number of reconfigurations. Finally, the

EDRCA accelerator is deployed on the Xilinx KV260 FPGA platform using hardware-software co-design to perform the target detection task of YOLOV2-TINY.

Figure 5. Xilinx KV260 FPGA hardware platform.

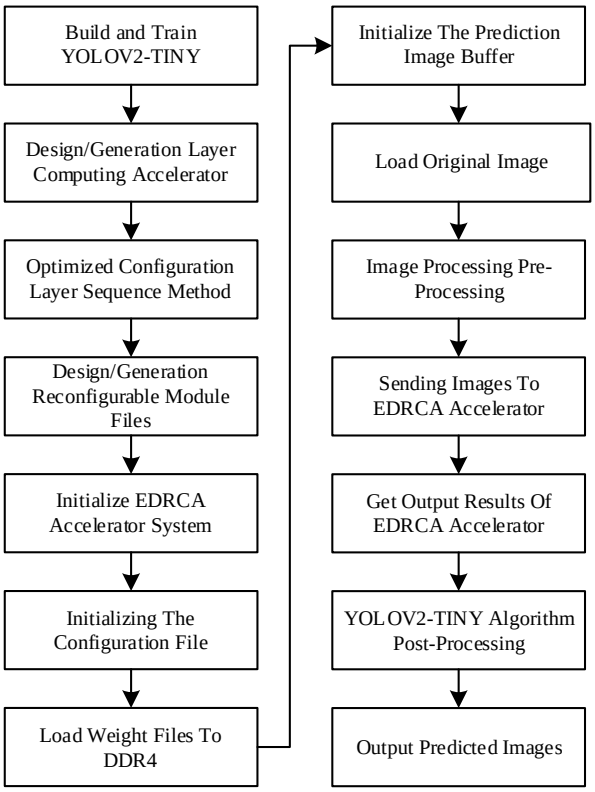

Figure 6. Implementation flow for hardware deployment of YOLOV2-Tiny.

4.3. Results

The final result of the design flow in Figure 5 was the construction of YOLOV2-TINY on the EDRCA accelerator system. The FPGA was synthesized and verified using VIVADO. Table 2 shows the actual hardware resource consumption of the EDRCA accelerator on the FPGA for the static region, dynamic region1, and dynamic region2, with 100% utilization for FF, 99.4% for LUT, 100% for BRAM, 100% for DSP, and 100% for URAM. These resources are essential for digital circuits, Flip-Flop (FF) is used to store the changes in state or signals of digital circuits; Lookup table (LUT) is used to implement logical functions; Block Random Access Memory (BRAM) is used to store large data structures; Digital Signal Processor (DSP) is used for high-speed digital signal processing and computing tasks; Ultra-Random Access Memory (URAM) is used to store large amounts of data. The dynamic reconfigurable design with reconfigurable modules that can be dynamically

configured effectively addressed the resource constraint problem, enabling full utilization of the hardware resources of the Xilinx KV260 FPGA.

Table 2. Hardware resource consumption of EDRCA accelerator in Xilinx KV260 FPGA.

Resource Type	FF	LUT	BRAM	DSP	URAM
Static Area	65,280	32,640	0	384	0
Dynamic Area 1	84,480	41,888	72	432	32
Dynamic Area 2	84,480	41,888	72	432	32
Resource Utilization	234,240	116,416	144	1248	64
Total Resource Volume	234,240	117,120	144	1248	64
Resource Utilization Rate (%)	100	99.4	100	100	100

The optimal layer sequence is generated using the optimal configuration layer sequence method discussed in Section 3.3. The general accelerator typically refers to a multi-operation layer accelerator that speeds up the entire CNN network, and connects the operation layer accelerators in the order of the CNN network to form a complete CNN network accelerator in an FPGA. The general accelerator performs the computational task by reconfiguring the operation kernels of the CNN network architecture in the order of the operational layers. As demonstrated by YOLOV2-TINY, the EDRCA accelerator reduces the number of configurations to 22.22% compared to the general accelerator, as shown in Figure 7. By using the optimal layer sequence method, the reconfiguration execution time is also reduced to 12.08% compared to the general accelerator, thus significantly reducing the reconfiguration-induced time overhead.

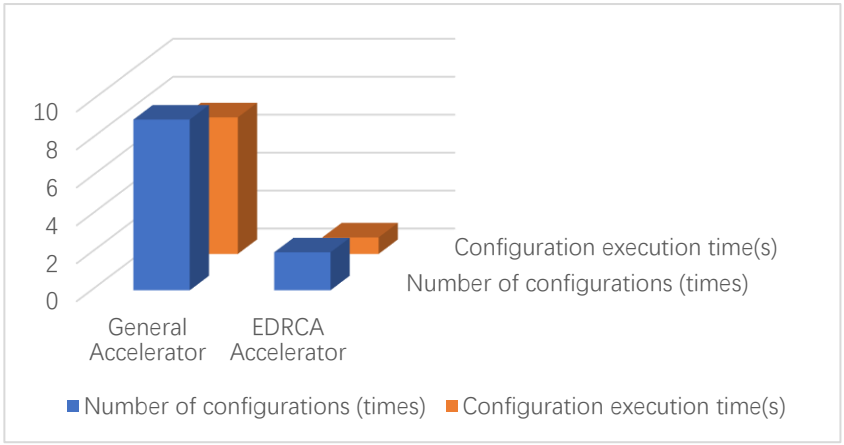

Figure 7. Comparison of EDRCA and general accelerator.

After conducting the experimental steps, the performance of the YOLOV2-TINY target detection application was evaluated on the EDRCA accelerator. The prediction results of YOLOV2-TINY were compared with the original graph and the prediction graph when implemented on the AMD Ryzen7 CPU and NVIDIA GeForce RTX2060 hardware platform, as shown in Figure 8. Table 3 shows the results of separate tests conducted on the FPGA, CPU, and GPU. At a clock frequency of 250 MHz, the EDRCA accelerator on the Xilinx KV260 platform achieved a computing performance of 75.1929 GOPS, which was 10.17 times higher than the AMD Ryzen7 CPU. The peak power consumption was 5.25 W, which was only 3.08% of the NVIDIA GeForce RTX2060 GPU, while consuming only 3.08% of the GPU's power.

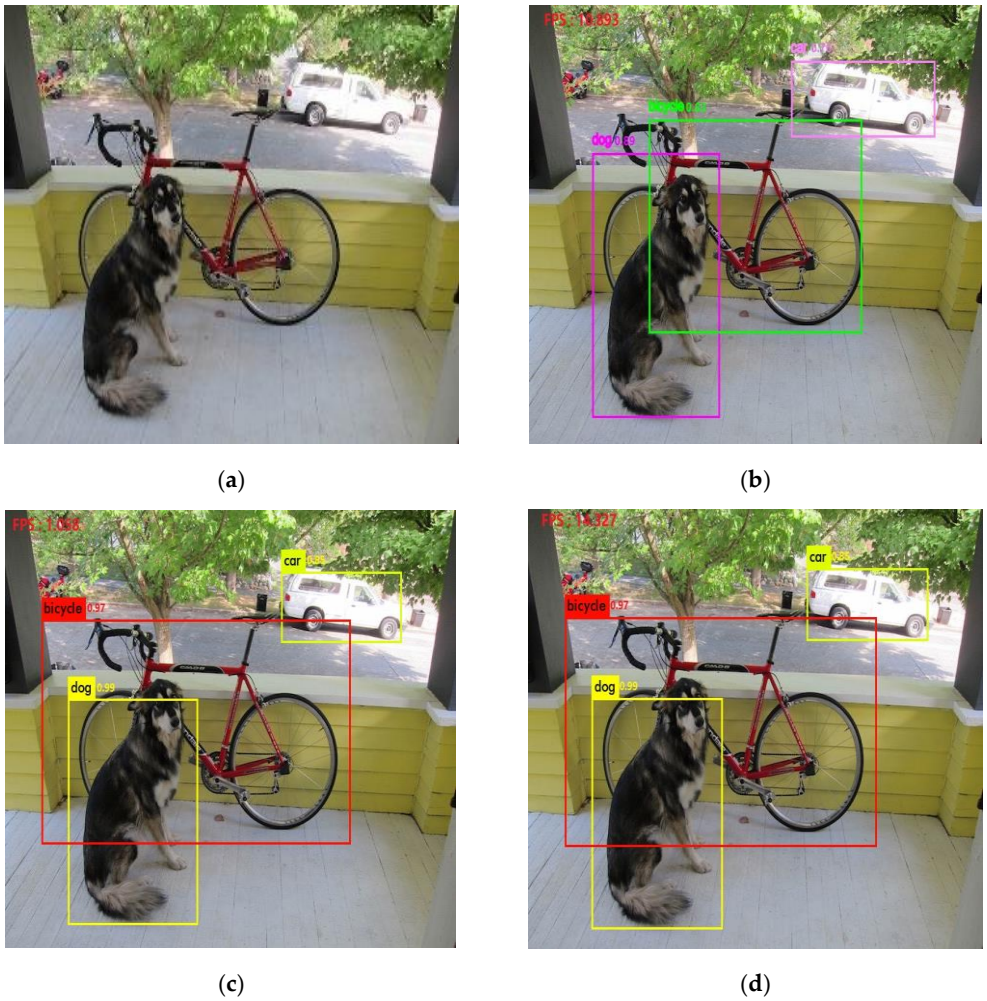

Figure 8. Comparison of original graph and predicted results on Xilinx KV260 FPGA, AMD Ryzen7 CPU, and NVIDIA GeForce RTX2060; (**a**) Original image; (**b**) Xilinx KV260 FPGA Predicted results; (**c**) AMD Ryzen7 CPU Predicted results; (**d**) NVIDIA GeForce RTX2060 Predicted results.

Table 3. Performance comparison results of different platforms.

Category	EDRCA	CPU	GPU
Hardware Platform	XCK26-SFVC784	AMD Ryzen7 4800 H	NVIDIA GeForce RTX2060
Operating Frequency (MHz)	250	2900	7010
Throughput (GOPS)	75.1928	7.3926	100.1748
Power Consumption (W)	5.520	45	175
Single Picture Inference time (s)	0.0918	0.9455	0.0698
Frames Per Second (fps)	10.893	1.058	14.327

The EDRCA accelerator performance experiment verifies its performance using the Xilinx KV260 FPGA platform management tool. The tool monitors the running frequency (250 MHz) and power consumption (5.520 W) of the FPGA. The power consumption of AMD Ryzen7 4800 H and NVIDIA GeForce RTX2060 was monitored by the power management tool in the computer server. The flux indicator measures the computational capability of the CNN algorithm model on each hardware platform, while the energy efficiency indicator measures the overall performance of the algorithm on each hardware platform. The hardware performance of EDRCA is compared to other accelerators in a comprehensive manner and the results are shown in Table 4. One study [16] designed a general-purpose architecture using PE array design, which performs better in terms of computation but

consumes 2.14 times more power than EDRCA. Another study [26] used 16-bit fixed-point quantization and a pipelined design to improve computing throughput and low-power performance, but it underutilizes resources and has less computing performance than EDRCA, which is 1.46 times more energy efficient. A third study [27] designed an accelerator at the RTL level with a method to find optimal convolutional chunking parameters, improving computational performance and data transfer bandwidth but consuming 2.15 times more resources than EDRCA and being 2.15 times less energy efficient.

Table 4. Results of EDRCA versus related work.

Category	Ref. [16]	Ref. [26]	Ref. [27]	EDRCA
Algorithm Model	YOLOV2-TINY	YOLOV2-TINY	YOLOV2-TINY	YOLOV2-TINY
Hardware Platform	ZCU102	MZ7035	XC7Z045	XCK26-SFVC784
Operating Frequency (MHz)	300	142	200	250
Fixed Data Precision (bit)	16	16	16	32
Giga Operations Per Second (GOPS)	102	25.64	121.2	75.1928
Single picture inference time (s)	11.8	2.754	19.2	5.520
Power consumption (W)	11.8	2.754	19.2	13.6219
Peak Energy Efficiency (GOPS/W)	8.6441	9.310	6.3125	13.6219

5. Discussion

This paper proposes the EDRCA, a high-performance, dynamically reconfigurable FPGA-based CNN accelerator architecture, to address the limitations of hardware resources and energy efficiency in embedded edge computing devices. Using the YOLOV2-TINY target detection algorithm for verification, the EDRCA leverages dynamic reconfiguration techniques on Xilinx KV260 FPGAs to overcome hardware constraints and optimize resource utilization. The paper also proposes an optimal configuration layer sequence method for CNNs and a unified high-speed interface template for the EDRCA accelerator to improve performance and bandwidth. The experiments show that the EDRCA can significantly enhance computing performance and resolve resource limitations through dynamic reconfiguration, achieving a peak energy efficiency of 13.6219 GOPs/W at 250 MHz.

Future research can focus on two areas. Firstly, we will investigate ways to optimize the EDRCA for target detection algorithms, aiming to enhance the performance of the dynamically reconfigurable hardware accelerator and minimize the initialization phase time overhead, to meet the demands of more complex industrial and military applications. Secondly, we plan to add encryption capability to EDRCA through dynamically reconfigurable technology, thereby improving the data security in edge computing systems.

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

Funding: This research was funded by various grants from the Shaanxi Provincial Department of Science and Technology, including the Key R&D Project (No. 2023-YBGY-215, No.2020-GY091, 21JK0548), as well as the Shaanxi Provincial Department of Education Service Local Special Program Project (No. 21JC002). Additionally, support was provided by Xi'an City Science and Technology Plan Project (No. 21XJZZ0006, No. 21XJZZ0005), Xianyang City Science and Technology Bureau Unveiling hanging major special project (L2022-JBGS-GY-01), Xianyang City Science and Technology Bureau plan project (No. 2020K02-64), and Xi'an City Weiyang District Science and Technology Plan Project (No. 202115).

Data Availability Statement: All data are contained within this article.

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

References

1. Nisar, A.; Nehete, H.; Verma, G.; Kaush, B.K. Hybrid Multilevel STT/DSHE Memory for Efficient CNN Training. *IEEE Trans. Electron. Devices* **2023**, *70*, 1006–1013. [CrossRef]
2. Ding, L.; Li, H.; Hu, C.; Zhang, W.; Wang, S. Alexnet feature extraction and multi-kernel learning for object-oriented classification. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2018**, *42*, 277–281. [CrossRef]
3. Ren, S.Q.; He, K.M.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1137–1149. [CrossRef] [PubMed]
4. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
5. Cheng, C. Real-Time Mask Detection Based on SSD-MobileNetV2. In Proceedings of the 2022 IEEE 5th International Conference on Automation, Electronics and Electrical Engineering, Shenyang, China, 18–20 November 2022; pp. 761–767.
6. Zhang, J.H.; Zhang, F.; Xie, M.; Liu, X.Y.; Feng, T.Y. Design and Implementation of CNN Traffic Lights Classification Based on FPGA. In Proceedings of the 2021 IEEE 4th International Conference on Electronic Information and Communication Technology, Xi'an, China, 18–20 August 2021; pp. 445–449.
7. Pestana, D.; Mirand, P.R.; Lopes, J.D.; Duarte, R.P.; Vestias, M.P.; Neto, H.C.; Sousa, J.T. A Full Featured Configurable Accelerator for Object Detection With YOLO. *IEEE Access* **2021**, *9*, 75864–75877. [CrossRef]
8. Chen, Y.H.; Fan, C.P.; Chang, R.C. Prototype of Low Complexity CNN Hardware Accelerator with FPGA-based PYNQ Platform for Dual-Mode Biometrics Recognition. In Proceedings of the 2020 International SoC Design Conference, Yeosu, Republic of Korea, 21–24 October 2020; pp. 189–190.
9. Xiao, Q.; Liang, Y. Fune: An FPGA Tuning Framework for CNN Acceleration. *IEEE Des. Test* **2019**, *37*, 46–55. [CrossRef]
10. Zeng, T.H.; Li, Y.; Song, M.; Zhong, F.L.; Wei, X. Lightweight tomato real-time detection method based on improved YOLO and mobile deployment. *Comput. Electron. Agric.* **2023**, *205*, 107625. [CrossRef]
11. Anwar, S.; Hwang, K.; Sung, W. Structured pruning of deep convolutional neural networks. *ACM J. Emerg. Technol. Comput. Syst. (JETC)* **2017**, *13*, 1–18. [CrossRef]
12. Gou, J.; Yu, B.; Maybank, S.J.; Tao, D. Knowledge distillation: A survey. *Int. J. Comput. Vis.* **2021**, *129*, 1789–1819. [CrossRef]
13. Fan, A.; Stock, P.; Graham, B.; Grave, E.; Gribonval, R.; Jegou, H.; Joulin, A. Training with Quantization Noise for Extreme Model Compression. *arXiv* **2020**, arXiv:2004.07320.
14. Zhang, C.; Sun, G.; Fang, Z.; Zhou, P.P.; Pan, P.C.; Cong, J.S. Caffeine: Towards Uniformed Representation and Acceleration for Deep Convolutional Neural Networks. *Trans. Comput. Aided Des. Integr. Circuits Syst.* **2019**, *38*, 2072–2085. [CrossRef]
15. Sharma, H.; Park, J.; Mahajan, D.; Amaro, E.; Kim, J.K.; Shao, C.; Mishra, A.; Esmaeilzadeh, H. From High-Level Deep Neural Models to FPGAs. In Proceedings of the ACM International Symposium on Microarchitecture, Taipei, Taiwan, 15–19 October 2016; pp. 1–16.
16. Wang, J.; Gu, S. FPGA Implementation of Object Detection Accelerator Based on Vitis-ai. In Proceedings of the 2021 11th International Conference on Information Science and Technology (ICIST), IEEE, Chengdu, China, 21–23 May 2021; pp. 571–577.
17. Zhang, S.; Cao, J.; Zhang, Q.; Zhang, Q.; Zhang, Y.; Wang, Y. An FPGA-Based Reconfigurable CNN Accelerator for YOLO. In Proceedings of the 2020 IEEE 3rd International Conference on Electronics Technology, Chengdu, China, 8–12 May 2020; pp. 74–78.
18. Wang, Z.; Xu, K.; Wu, S.X.; Liu, L.Z.; Wang, D. Sparse-YOLO: Hardware/Software Co-Design of an FPGA Accelerator for YOLOv2. *IEEE Access* **2020**, *8*, 116569–116585. [CrossRef]
19. Aloysius, N.; Geetha, M. A Review on Deep Convolutional Neural Networks. In Proceedings of the 2017 International Conference on Communication and Signal Processing, Chennai, India, 6–8 April 2017; pp. 588–592.
20. Wang, X.; Deng, J.Y.; Xie, X.Y. Design and implementation of reconfigurable CNN accelerator. *Transducer Microsyst. Technol.* **2022**, *41*, 82–85, 89.
21. Everingham, M.; Gool, L.V.; Williams, C.K.I.; Winn, J.; Zisserman, A. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. 2, 5. Available online: http://www.pascalnetwork.org/challenges/VOC/voc2012/workshop/index.html (accessed on 2 February 2023).
22. Yuan, F.L.; Gong, L.; Lou, W.Q. Performance Cost Modeling in Dynamic Reconfiguration Hardware Acceleration. *Comput. Eng. Appl.* **2022**, *58*, 69–79.
23. Chong, W.; Ogata, S.; Hariyama, M.; Kameyama, M. Architecture of a Multi-Context FPGA Using Reconfigurable Context Memory. In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Denver, CO, USA, 4–8 April 2005; p. 7.
24. Sunkavilli, S.; Chennagouni, N.; Yu, Q. DPReDO: Dynamic Partial Reconfiguration enabled Design Obfuscation for FPGA Security. In Proceedings of the 2022 IEEE 35th International System-on-Chip Conference, Belfast, UK, 5–8 September 2022; pp. 1–6.
25. Yuan, F.L. *Convolutional Neural Network Accelerator Based on Dynamic Hardware Reconfiguration*; University of Science and Technology of China: Hefei, China, 2021.

26. Xu, H.D. *Research and Implementation of Target Detection Algorithm Based on Zynq Platform*; School of Information and Communication Engineering: Chengdu, China, 2021.
27. Chen, T.S. *Design and Implementation of a Reconfigurable Convolutional Neural Network Accelerator Based on FPGA*; Guangdong University of Technology: Guangzhou, China, 2021.