International Journal of Communication Networks and Information Security
ISSN: 2073-607X, 2076-0930
Volume 15 Issue 04 Year 2023

Analysis of CMOS IC-based Hybrid Architecture for Edge Computing

Wen He*
Master, Faculty of Engineering and Built Environment,University Kebangsaan Malaysia, Bangi, 43600, Malaysia
hermen229@gmail.com

Article History

Received: 21 September 2023
Revised: 18 October 2023
Accepted: 20 December 2023

Abstract

With the rapid advancement of Internet of Things (IoT), mobile internet, and big data technologies, edge computing has emerged as a novel computing paradigm. In the hybrid architecture of edge computing,Complementary Metal-Oxide-Semiconductor (CMOS) integrated circuits play a pivotal role in empowering edge devices and servers with essential computing, storage, and communication capabilities. Despite their critical importance, CMOS integrated circuits in edge computing environments confront significant challenges in low-power electronics. These challenges include an increase in power density and a decrease in system stability and reliability. This paper delves into the key technologies of the hybrid architecture in edge computing and sheds light on the vital role of CMOS integrated circuits in edge devices. It introduces a novel approach for low-power electronics, which encompasses methods like optimization of double threshold voltage and refinement of algorithmic processes. These methods aim to tackle the power-efficiency issues while maintaining the performance of edge computing systems.Furthermore, the paper presents a detailed analysis of the proposed low-power techniques, focusing on how they can effectively reduce power consumption without compromising the functionality and efficiency of the edge computing systems. It concludes with a comprehensive discussion on the optimization results, highlighting the benefits and potential implications of implementing these low-power strategies in edge computing environments. This discussion not only underscores the importance of energy efficiency in edge computing but also opens new avenues for future research and development in this rapidly evolving field.

Keywords: Edge Computing, CMOS ICs, Hybrid Architecture, Low Power Design

1. Introduction

With the rapid development of technologies such as IoT, mobile Internet, and big data, more and more devices and sensors are connected to the Internet, generating massive amounts of data. The traditional cloud computing model suffers from high latency, bandwidth bottleneck, data privacy, and other problems in processing these data, which cannot meet the demand of real-time

and low latency [1], [2], [3]. To solve these problems, edge computing has emerged as an emerging computing model [4].

Edge computing is a distributed computing model that shifts data processing and computing power from the cloud to edge devices or edge servers close to the data source [5]. The edge computing architecture places computing and storage resources as close as possible to the data source, enabling faster data processing and decision-making, resulting in low-latency, real-time responses [6]. At the same time, edge computing can also reduce the load pressure on the cloud, reduce data transmission bandwidth requirements, and improve system reliability and security [7]. The development background of edge computing stems from the inadequacy of the traditional cloud computing model and the demand for real-time and reliability [8]. The cloud computing model has a centralized architecture, and data needs to be transmitted and processed over long distances to get results, which leads to higher latency [9]. Especially for some application scenarios that require high real-time performance, such as intelligent transportation, intelligent manufacturing, and IoT, cloud computing cannot meet the demand for fast response [10]. Convolutional neural network (CNN), as an emerging AI implementation, is usually deployed on network edge devices to achieve functions such as object detection and recognition [11]. However, traditional convolutional neural networks often require 1 to 1.5 orders of magnitude more computing power than ordinary edge devices and are difficult to work in edge nodes in the context of edge computing [12]. Therefore, one of the challenges is how to properly design a network architecture that is suitable for offloading to the edge nodes so that the edge and cloud computing power can play their respective advantages [13], [14], [15], [16]. At the same time, the hardware design of edge computing nodes needs to fit the specific characteristics of CNN edge implementations [17], [18], [19].

CMOS (Complementary Metal-Oxide-Semiconductor) integrated circuits are a common semiconductor device technology that plays a key role in edge computing and are widely used in edge devices and edge servers [20], [21]. First, the low power consumption of CMOS ICs enables edge devices to operate with a limited energy supply and extend the endurance of the devices [22]. Second, the high integration and small size of CMOS ICs enable edge devices to integrate more functions and processing capabilities for real-time data processing and analysis [23]. In addition, the lower cost of CMOS integrated circuits can reduce the manufacturing cost of edge devices and drive the popularity and adoption of edge computing [24].

In edge servers, CMOS integrated circuits provide powerful computing and storage capabilities that enable the processing and analysis of large-scale data [25]. The high integration and high performance of CMOS integrated circuits enable edge servers to handle complex computing tasks and large-scale data storage, meeting the requirements for high performance and large capacity in edge computing [26].

In summary, the application of CMOS integrated circuits in edge computing is of great significance. It not only provides the computational and storage capabilities required by edge devices and edge servers but also provides technical support for the implementation of hybrid architectures for edge computing. With the continuous development and innovation of CMOS integrated circuit technology, edge computing will further enhance its processing power and efficiency, bringing more applications and business opportunities to various industries [27].

However, CMOS ICs still face some challenges in edge computing. The development of the CMOS IC industry has enhanced the circuit operating frequency and integration but also made the circuit power density increase [28]. As the process feature size decreases, the power consumption of CMOS integrated circuit systems increases exponentially. Excessive power consumption reduces the stability and reliability of the system and makes packaging difficult. Therefore, low-power design is becoming an increasingly important part of current CMOS IC design. CMOS ICs continue to evolve towards deep submicron industrial nodes, and the static power consumption of transistors can no longer be ignored and must be taken into account in circuit design [29]. Meanwhile, [39] has illustrated such aspect also functions well in designing of systems.

Based on the above analysis, by studying the characteristics of convolutional neural network algorithms and FPGA features and then analyzing the characteristics of convolutional neural network models and the needs of edge computing context [30], we finally propose a target monitoring model suitable for working in the edge computing environment. At the same time, based on the low-power development direction of the chip, the low-power design and optimization

techniques of the embedded system are proposed. This paper realizes the dual improvement of CMOS integrated circuit and edge computing based on the low-power design of CMOS integrated circuit and the hybrid architecture acceleration algorithm of edge computing to achieve the purpose of improving the overall structure operation efficiency.

2. Related Works

With the advent of the Internet of Things (IoT) and the era of big data, edge computing has emerged as a promising computing paradigm that is gaining increasing attention. Edge computing involves the deployment of computing and data processing capabilities on edge devices or edge servers located closer to the data sources [31]. This enables real-time data processing and analysis, offering faster and more efficient computing and service capabilities.

The hybrid architecture of edge computing combines edge computing with traditional cloud computing, leveraging the collaborative functionality between edge devices, edge servers, and cloud servers [32]. This integration enables more flexible and reliable computing and data processing capabilities. The design of the hybrid architecture of edge computing encompasses various crucial technological domains, including sensor technology and data acquisition, network communication and data transmission, data processing and storage, as well as artificial intelligence and machine learning. These technologies synergistically contribute to providing comprehensive support and assurance for edge computing.

2.1 Edge Computing Overview

Edge computing is a highly virtualized platform that provides network services such as computation and storage between end devices and the cloud and is an extension of cloud computing. Edge computing operates on downlinking data from the cloud and uplinking data from smart devices, respectively. Edge devices have considerable computing power and idle resources. If the system can handle some simple edge devices with low latency, then the system can have low latency as it does for real-time tasks. The application of the edge computing model has the following 3 advantages.

(1) Processing huge amounts of temporary data at the edge of the network, only data with high computational complexity is uploaded to the cloud, which greatly reduces the pressure on network bandwidth and data centres.

(2) Providing data processing services near data production terminals eliminates the step of requesting responses from the cloud computing centre, achieving the purpose of reducing system latency and enhancing service response.

(3) Edge computing can prevent users from uploading private data, mainly by storing private data in network edge devices to reduce the risk of data leakage.

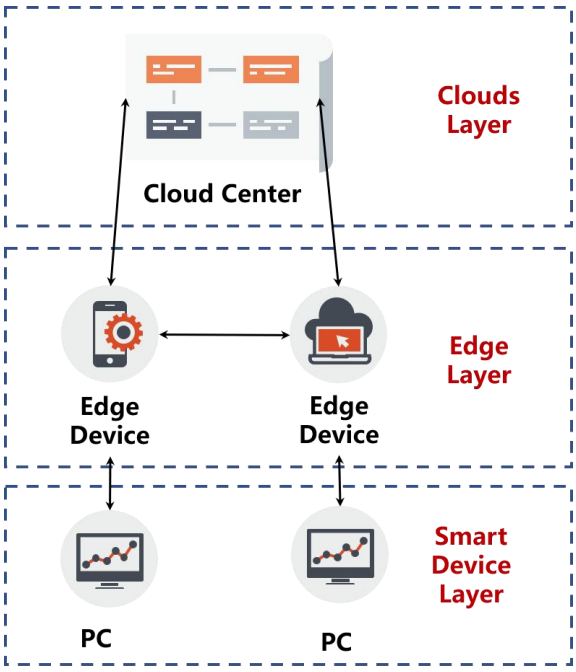

Figure 1. Generic Edge Computing Framework

2.2 CMOS ICs in Edge Devices

As an important part of edge computing architecture, edge servers need to have high computing power. CMOS integrated circuits in edge servers can provide high-performance computing power to support complex data processing and algorithm execution. By integrating multiple CMOS integrated circuits, the edge server can realize distributed computing and parallel processing to improve the overall computing efficiency and throughput.

Edge servers require fast data transfer and communication with edge devices to acquire real-time data and perform processing. CMOS integrated circuits enable fast data interaction between edge devices and edge servers through high-speed data transfer interfaces and communication protocols. This fast data transfer capability can reduce data transfer latency and improve real-time data processing and efficiency.

As a bridge between edge computing architecture and cloud computing, the edge server can work with cloud servers. CMOS ICs in the edge server can realize task offloading and distribution and send part of the computing tasks to cloud servers for processing to improve the overall computing efficiency and resource utilization. CMOS ICs, through high-speed network interfaces and protocols, realize edge server CMOS ICs enable fast data transmission and communication between edge servers and cloud servers through high-speed network interfaces and protocols to realize the collaborative work between edge computing and cloud computing.

3. Methodology

CMOS integrated circuit low-power design plays a key role in the edge computing hybrid architecture [33]. The goal of edge computing hybrid architecture is to enable efficient data processing and computation between edge devices and edge servers to provide low latency, high performance, and reliable services. The edge computing hybrid architecture requires collaborative data transfer and computation tasks between edge devices and edge servers. Since data transmission and computation tasks consume energy, low-power designs can reduce energy waste and improve energy efficiency. By designing low-power CMOS integrated circuits, the overall system energy consumption can be reduced, resulting in more energy-efficient and efficient edge computing.

The power consumption of CMOS integrated circuits consists of many factors and can be analyzed by type and composition [34]. According to the type of power consumption can be divided into dynamic power consumption, static power consumption, and surge power consumption, and the power consumption percentage depends on the application scenario requirements and system architecture algorithms. Low-power design is a top-down design, i.e., the design of low-power strategy starts at the system architecture level [35]. A proper system architecture design can save more than 60% of power consumption. The design at this level is often done by system and architecture designers, integrating multi-power and multi-voltage design, system clock design, software/hardware co-design, algorithm design, asynchronous design, IP selection, etc. It is demanding for designers and requires extensive design and practical experience. This paper implements a tool-based improvement to the dual-threshold algorithm to achieve a proven low-power design goal through simulation experiments.

3.1 Digital Threshold Voltage Optimization

The dual-threshold voltage technique is a widely used low-power design technique, and its role in reducing the static power consumption of the circuit is relatively significant. And it is a circuit optimization design technique that is widely used to reduce the static power consumption of integrated circuits. This design method is that the transistor devices of the circuit are designed with one threshold voltage, based on which the circuit is designed with transistor devices of multiple threshold voltages instead; that is, different threshold voltages are used for circuit units on different timing paths to ensure the timing requirements of the critical path, and the power consumption requirements of the non-critical path, thus reducing the static power consumption of the circuit.

In integrated circuit design, the delay of the off-key path determines the clock speed of the synchronous circuit. The timing delay of the critical path between the flip-flop and the synchronous circuit is compared with the delay of the non-critical path in Figure 2.

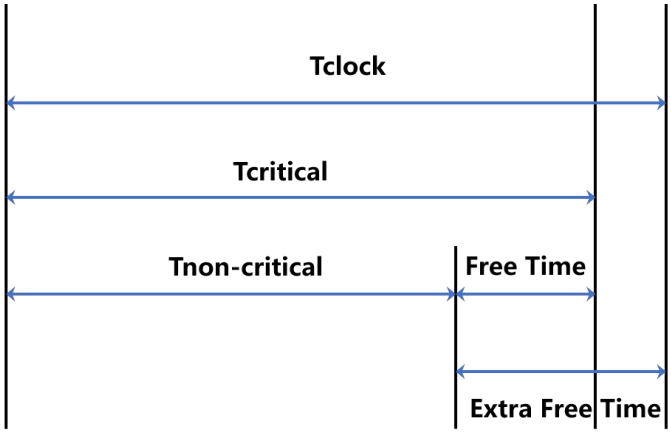

Figure 2. Idle Time of Different Paths

In a standard circuit design, the delay of the most critical paths determines the target clock frequency. However, since the number of these critical paths is only a small fraction of the total number of IC paths, the gate units on the non-critical paths work very slowly to a significant extent (their signals arrive earlier than required so that the signal arrival time at the receiver side and the usage time form a time interval). The performance of the circuit, in this case, is not improved by the early arrival of the signal at the receiving end of the non-critical path. The gate unit circuit on the non-critical path then consumes additional energy as a result, resulting in wasted power consumption.

Due to local timing constraints, all gate cells in the non-critical paths of the circuit cannot be replaced with high threshold cells at the same time. In order not to degrade the total system operation performance and stability, the gap between timing paths must be low and greater than zero. If the delay requirement cannot be met by replacing all gate cells on a path, then a combination of gates with both high and low threshold cells is required for that path.

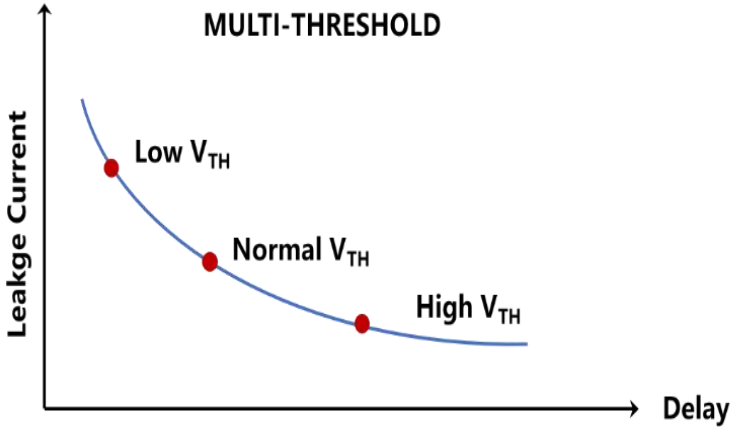

Figure 3. Leakage Current of a Dual Threshold Voltage Cell

3.2 Algorithm Process

To simplify the problem, some settings are made in this thesis for algorithm design as follows. In this paper, we assume that the high threshold voltage will not additionally increase the static power consumption of the boost circuit. Also, in the algorithm implementation, we will only consider the effect of the dual-threshold low-power design threshold voltage on the power consumption and delay of the entire timing circuit while the other conditions in the circuit remain unchanged. In addition, we assume that after this optimization of the threshold voltage circuit, the reduction in static power consumption is directly related to the proportion of high threshold voltage circuit components, i.e., the proportion of high threshold voltage devices in the circuit used in this paper to represent the effect of the algorithm in reducing power consumption. The algorithm procedure is as follows:

Step 1. Initialize, read into a netlist, and set a target time of T0.

Step 2. Calculate the slack margin for each timing path. By adjusting the timing constraints, the slack margin of the timing path P=(v1,v2.. .vk) is compared with our given target time T0, the relaxation margin minus the target time To is calculated, and all the timing paths P with relaxation margins less than 0 are marked into the path set R. The individual nodes on these paths are marked into set V1.

Step 3. The initial circuit optimization is performed by swapping the cells of each path in set V1 to low-threshold cells, calculating the relaxation margin difference of the cells on these paths before and after the replacement, marking them to that node cell, and then swapping the node cells back to high-threshold voltage cells. Then, the node cells with large intrinsic delay in set V1 are operated and replaced with low-threshold voltage cells, and these nodes are moved to the replacement set V1. The set VLVT is read into the circuit netlist.

Step 4. Recalculate the relaxation margin of each timing path and reset the set of paths R and the set of nodes V1.

Step 5. The high threshold voltage of the node of the timing path P in the timing path set R is replaced by the low threshold voltage, and the slack margin of that timing path is calculated after each replacement. Where the relaxation margin of each node is obtained by weighting both wns and tns, and the tns value of the node is calculated only for path values less than a specific value s0 in each round. This particular s0 is updated after one round of computation, and the size of the value is related to the size of the worst slack value after each re-update of the timing.

Step 6. After calculating the relaxation margin, the standard unit of the node with the largest negative relaxation margin on the path is replaced. If the relaxation margin is positive at this point, the temporal path P is removed from the set R, while these nodes are removed from V1 and moved to the replacement set VLVT. If not, the standard unit of the next node on the path is replaced. This is an iterative backtracking process that performs all the paths in the set R once.

Step 7. The node transformation in VLVT is read into the circuit, the circuit timing is updated, values such as s0 are updated, and at the same time, whether the path of the violation is eliminated or the number of iterations is reached is judged, and step 7 is executed; otherwise, step 4 is executed by backtracking.

Step 8 Statistical results, end of the algorithm.

The specific flowchart representation of the algorithm is shown in Figure 4:

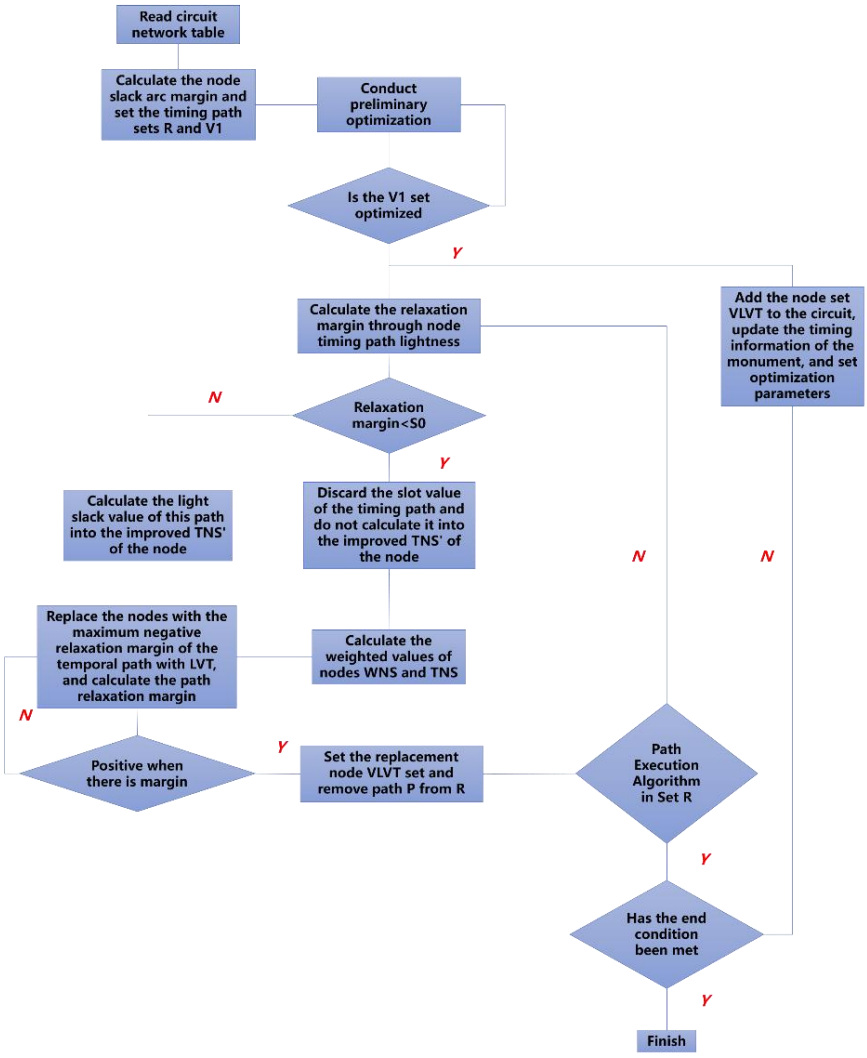

Figure 4. Algorithm Implementation Flow

3.3 Analysis of Results

Usually, the target period we set is a little tighter than the actual requirement, keeping some margin. After we optimize the circuit, the path of delay violation in the timing path of the circuit can achieve convergence of the timing. Since the library used can achieve a frequency of 200M, in order to verify the simulation results, we use a clock period constraint of 4.5ns in practice; then in the STA stage, the clock period is reset to 4.3ns, run the algorithm script, and simulate the calculations.

The optimization calculation is performed on the chip, and the slack margin of the timing paths of the violation is tracked. And these paths are marked in the algorithm to obtain their unoptimized timing violation.

During the optimization process of the algorithm, we record the slack margin of the circuit nodes and record the circuit timing slack margin as Slack A when it is not optimized, Slack B when it is initially optimized, and Slack C when it is finally optimized, then the slack margin of the violated paths in the timing paths changes as shown in Figure 5.

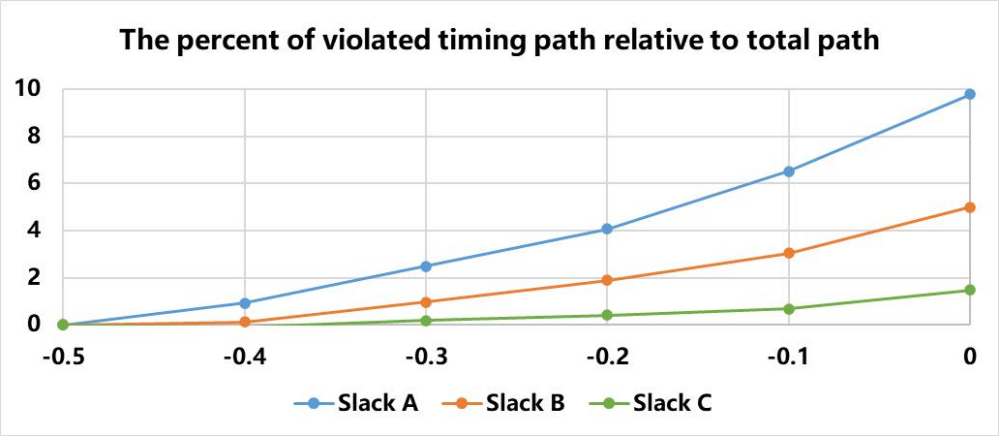

Figure 5. Chip Circuit Timing Path Relaxation Margin Variation

The dual-threshold low-power idea is to map each circuit cell in the gate-level netlist to a high-threshold voltage circuit cell and then to replace the high-threshold cell with a low-threshold voltage cell without violating the timing violation. Comparison between this algorithm and the algorithm in this paper Table 1:

Table 1. Comparison of the Two Algorithms

	Total Units	**Low Threshold Cell Count**	**Proportion Of Low Threshold Units**	**Timing**
Control Algorithm	8710	1567	0.18	4.35ns
Algorithms in this Paper	9120	1277	0.14	4.75ns

Compared with the control algorithm, this algorithm can make better use of the lose her margin in the path to maximize the use of high threshold voltage cells and thus reduce chip power consumption.

4. Convolutional Neural Network Acceleration Algorithm for Edge Computing Hybrid Architecture

In recent years, with the proliferation of computing power and data, neural network algorithms have developed significantly and have shown accuracy and efficiency beyond traditional algorithms in tasks such as image, speech, and video processing [36]. Especially after the AlexNet network model was proposed in 2012, intelligent image processing methods based on convolutional neural networks (CNNs) have gradually become mainstream methods and started to be applied in key areas of production life, such as aerospace, industrial inspection, and autonomous driving [37]. However, while convolutional neural networks provide powerful image feature extraction capability, they also bring huge computational parameters and billions of floating point computations, but their large number of parameters, computational complexity, and high inter-layer independence make them difficult to be efficiently deployed in edge scenarios with lower power consumption and fewer resources [38].

4.1 Hybrid Architecture Overall Design

First, to efficiently utilize the less storage resources inside the FPGA, the network model is compressed and optimized in this paper before deploying it. After that, a Digital Signal Processor (DSP) array computation acceleration architecture with data flow controlled by instructions is implemented on a Xilinx FPGA. Finally, the CPU processing speed was also optimized using multi-threading techniques to achieve a flowing processing effect.

This paper implements a Register Transfer Level RTL-level instruction-controlled data flow architecture on FPGAs that use high-speed DSP arrays to accelerate convolutional computation. In

addition, the architecture retains a certain degree of generality, using a dedicated set of software-generated instructions to control the data flow for the forward propagation of different backbone networks. A small amount of image pre-processing and feature map post-processing involving floating-point computation is designed in this paper to use CPUs for computation and to accelerate the inference computation of convolutional neural network algorithms using a hybrid architecture form. The data flow and control flow diagram of the overall design is shown in Figure 6. The CPU stores the pre-processed feature maps into the off-chip memory via the PCI-Express bus. The feature cache and weight cache read and cache the feature map and weight data from the off-chip memory according to the instruction opcode. These data are sent to the DSP array for multiplication and addition calculation under command control, and the results are computed by quantization, activation, pooling, upsampling, and other logic calculation modules to obtain the feature map under command control. The intermediate layer feature map is written back to the feature cache for subsequent propagation, and the output layer features are written back to the off-chip memory, which is read by the CPU through the PCI-Express bus and post-processed according to different algorithms to obtain the final results and display the output. Among them, instructions are pre-stored in an on-chip cache (ROM).

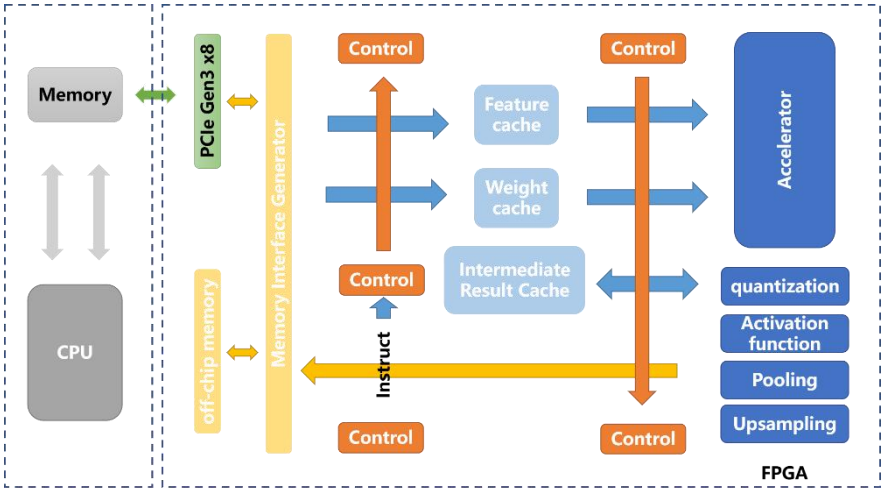

Figure 6. Overall Design Diagram of the Hybrid Architecture

4.2 Accelerated Solution Design

According to the analysis above, the architecture must have the ability to support multiple network structures and, maximize the efficiency of module operation and reduce the idle time of the computing core if it is to work in the context of edge computing where multiple target applications exist. Therefore, the architecture of the acceleration scheme under study is designed as a reconfigurable module multiplexing architecture, and an FPGA platform is chosen for its implementation: configurable basic network structures such as convolution and pooling are implemented on-chip, and various types of network structures can be composed according to the specific needs at the time of application. Such an architecture can significantly increase the module reuse rate and make the acceleration scheme designed in this section somewhat general. The acceleration scheme mainly consists of an interface module, a convolution module, a pooling module, and a parameter configuration module. The main workflow can be summarized as follows: First, the ARM Cortex-A9 CPU on the PS side sends a packet to the configuration module of the acceleration scheme to configure the internal computing registers via the AXI_lite bus. After the configuration is completed, the init_done flag bit is transmitted, and the configuration module sends instruction packets to the computing module according to the internal communication protocol. The specific structure is shown in Figure 7. Each module is sent a data request by the DMA module according to the configuration, and the request is converted into a communication protocol in AXI_full format by the interface module to take the corresponding weights and characteristics from the Double Rate Synchronous Dynamic Random Access Memory (DDR SDRAM) for arithmetic.

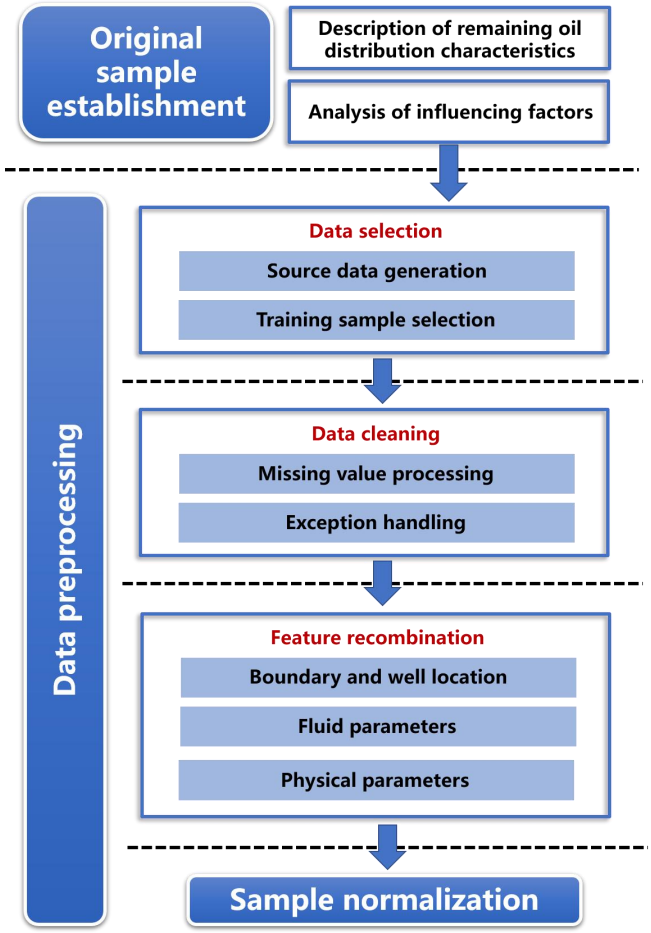

Figure 7. Overall Architecture and Development Environment of the Accelerated Solution

After the operation is completed, the operation module writes back and pulls up the completion flag. The configuration module will continuously read the completion flag register after the configuration is completed and feed the flag back to the driver function after successful reading to mark the completion of this operation, and the next configuration and calculation can be performed.

4.3 Module IP and System Packaging

In the implementation of the gas pedal, the system-level implementation is carried out by encapsulating various modules into IPs and then calling them in a unified way. The operational properties of each module are realized by configuring the corresponding parameters in the driver functions. Therefore, when the gas pedal handles the computing tasks of a multi-layer network, it only needs to call the same driver function to instantiate different parameters when it encounters the same type of network structure model. This approach increases the generality of the design and facilitates the gas pedal to work in resource-sensitive edge computing environments.

In this paper, two main operational modules for convolution and pooling, two CAS modules for convolution and pooling, and the ATIF interface module are implemented, and the packaging of the above IP is performed in the Vivado 18.02 tool. Figure 8 shows the encapsulated CAS_POOL module, CAS_CONV module, and their interfaces.

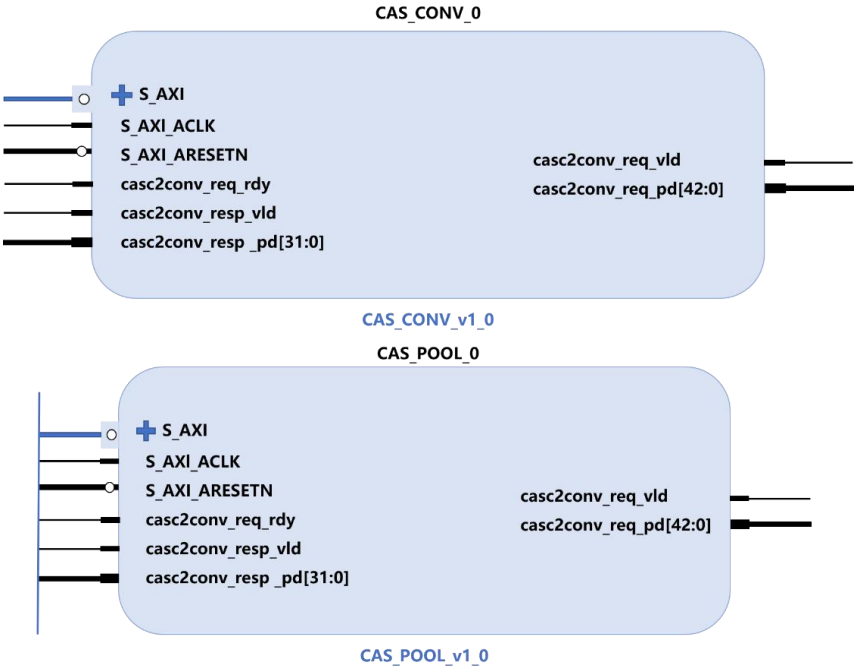

Figure 8. Encapsulated IP

After implementing the various IPs, the complete Block Design is shown in Figure 9.

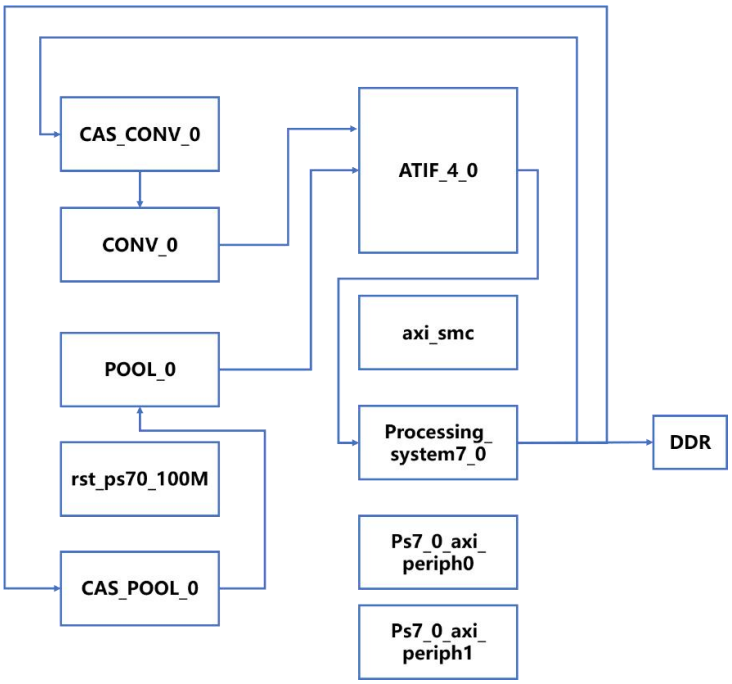

Figure 9. System-wide Block Design

The whole gas pedal works like a coprocessor. The main processor uses the AXI-lite bus to configure it, and the coprocessor receives the configuration information and, reads the data through its own AXI-full bus connected to the DDR controller, and then writes it back through the same path after computing. The main processor is the CPU hardcore on the PYNQ-z2 development board. Therefore, the whole system needs to be built in two packages, first assembling the gas pedal itself composed of IP through BlockDesign, then calling in the BlockDesign of the whole system and adding ZynqSystem, clock reset module, bus interconnect module, etc., to compose the final output file. The two modules near the middle of the left column are the convolution and pooling operation modules, which receive the configuration packets from the two CAS modules above and below the left column, parse them into operation instructions, and then send the data handling signals to the

ATIF module via its dma in the form of an internal protocol. After the ATIF module arbitrates the attribution of the current request, it turns to AXI-full protocol and sends it outward.

Since the DDR controller is integrated into the ZYNQ chip on the PYNQ board, the ATIF must first pass through the PS side when driving the DDR. After the data is read back from the above path, it is calculated by the corresponding arithmetic module and then written back to the corresponding physical address. The fourth module in the left column is the system-generated clock reset module when the clock is connected, and several modules in the right column, except ATIF and PS, are AXI bus connectors for bridging AXI interfaces that cannot be directly connected due to inconsistencies in bit width and timing. Because of the redundant connection lines, they are not drawn. After the whole system is packaged by BlockDesign, it needs to be packaged as an Overlay for system calls on the PYNQ board. Finally, it is exported as a bitstream file and the corresponding tcl file, which is copied to the PYNQ onboard SD card. At this point, the acceleration system implementation is complete.

4.4 Accelerated Solution Performance Analysis

Table 2 shows the resource consumption of the whole project. Due to the parallel computing characteristics of this design, the Digital Signal Processing (DSP) resources are relatively more consumed, with a total of 107, accounting for 48.64% of the overall resources, which are mainly used for multiplication and addition operations. 64 BRAM resources are consumed, accounting for 45.71% of the overall resources, which are mainly used as on-chip caches to pre-store weights or feature matrices. Look Up Table (LUT) resources consume 25.15% of the overall resources and are mainly used to perform calculations or participate in data transfer. Since the design of this topic involves a large amount of data multiplexing or moving, and there are also a large number of logical fits at the periphery of the arithmetic unit matrix, the LUT resource consumption is relatively high. Analysis of Table 2 shows that the multiplexing architecture adopted in this design keeps the overall gas pedal's dependence on resources at a low level and still does not consume all available resources on smaller FPGA boards, and there is some room for upward exploration of resources consumption indicators such as parallelism and bit width. Therefore, it is well suited to work in the resource-sensitive edge computing field.

Table 2. Resources Consumed by the Design

Resource	Utilization %	Available	Utilization
LUT	24.36	52700	12986
FF	12.54	105800	14217
DSP	47.86	240	125
BRAM	45.71	160	66
LUTRAM	11.8	16400	1798
BUFG	2.89	32	1

Since applications in the context of edge computing are particularly sensitive to power consumption, this gas pedal adopts a parallelized design and consumes fewer resources, so the overall power consumption of the gas pedal is at a fairly low level. In this paper, the power consumption information is analyzed using the Vivado tool, and the power consumption analysis of each specific module is shown in Table 3.

Table 3. Power Consumption of Each Module

Specific Modules	Power Consumption	Ratio
processing_system7_0	1.358W	73%
POOL_0	0.036W	3%
Leaf Cells	0.001W	<1%

axi_smc	0.048W	2%
ATIF_4	0.009w	1%
CONV	0.165W	12%
ps7_0_axi_perph	0.006W	1%
ps7_0_axi_perph_1	0.003W	1%
rst_ps7_0_100M	<0.001W	<1%
CAS_CONV	0.005W	<1%
CAS_POOL	0.002W	<1%

5. Conclusion

This paper describes the role of CMOS integrated circuits in edge computing hybrid architectures in edge devices and discusses the low-power design challenges faced in edge computing. To address these challenges, two low-power design approaches, dual-threshold voltage optimization and algorithmic process, are proposed.

Dual-threshold voltage optimization is achieved by using different operating voltages according to the requirements of different modules in the chip to optimize power consumption. In the experiment, by distinguishing the critical path and non-critical path of the circuit, a high threshold unit is used to design the non-critical path, and a low threshold circuit unit is used to design the critical path to achieve the improvement of circuit system performance and the reduction of circuit power consumption based on ensuring the timing convergence of the experimental circuit. For the algorithmic process, the power consumption can be reduced by algorithm optimization and improvement, such as using more efficient algorithms and data compression techniques.

After analyzing and discussing these approaches, it can be concluded that CMOS integrated circuits play an important role in edge computing but face the challenge of low-power design. With a dual-threshold voltage optimization and algorithmic process approach, power consumption can be effectively reduced, and the performance and reliability of edge devices can be improved.

References

[1] G. Aceto, V. Persico, and A. Pescapé, "Industry 4.0 and health: Internet of things, big data, and cloud computing for healthcare 4.0," *Journal of Industrial Information Integration*, vol. 18, p. 100129, 2020.

[2] Z. Lv, H. Song, P. Basanta-Val, A. Steed, and M. Jo, "Next-generation big data analytics: State of the art, challenges, and future research topics," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1891-1899, 2017.

[3] L.M. Dang, M.J. Piran, D. Han, K. Min, and H. Moon, "A survey on internet of things and cloud computing for healthcare," *Electronics*, vol. 8, no. 7, p. 768, 2019.

[4] H.R. Abdulqadir, S.R. Zeebaree, H.M. Shukur, M.M. Sadeeq, B.W. Salim, A.A. Salih, and S.F. Kak, "A study of moving from cloud computing to fog computing," *Qubahan Academic Journal*, vol. 1, no. 2, pp. 60-70, 2021.

[5] R. Kozik, M. Choraś, M. Ficco, and F. Palmieri, "A scalable distributed machine learning approach for attack detection in edge computing environments," *Journal of Parallel and Distributed Computing*, vol. 119, pp. 18-26, 2018.

[6] A.V. Dastjerdi, H. Gupta, R.N. Calheiros, S.K. Ghosh, and R. Buyya, "Fog computing: Principles, architectures, and applications," in *Internet of things,* Morgan Kaufmann, 2016, pp. 61-75.

[7] X. Zhang, Z. Cao, and W. Dong, "Overview of edge computing in the agricultural internet of things: key technologies, applications, challenges," *Ieee Access*, vol. 8, pp. 141748-141761, 2020.

[8] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, and M. Nemirovsky, "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing," in *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD),* IEEE, Dec. 2014, pp. 325-329.

[9] S. Garg, A. Singh, K. Kaur, G.S. Aujla, S. Batra, N. Kumar, and M. S. Obaidat, "Edge computing-based security framework for big data analytics in VANETs," *IEEE Network*, vol. 33, no. 2, pp. 72-81, 2019.

[10] J. Cheng, W. Chen, F. Tao, and C.L. Lin, "Industrial IoT in 5G environment towards smart manufacturing," *Journal of Industrial Information Integration*, vol. 10, pp. 10-19, 2018.

[11] W. Ullah, A. Ullah, T. Hussain, K. Muhammad, A. A. Heidari, J. Del Ser, S.W. Baik, and V.H.C. De Albuquerque, "Artificial Intelligence of Things-assisted two-stream neural network for anomaly detection in surveillance Big Video Data," *Future Generation Computer Systems*, vol. 129, pp. 286-297, 2022.

[12] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G.V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 11* , Springer International Publishing, 2020, pp. 128-144.

[13] W. Yu, F. Liang, X. He, W.G. Hatcher, C. Lu, J. Lin, and Yang, X., "A survey on the edge computing for the Internet of Things," *IEEE access*, vol. 6, pp. 6900-6919, 2017.

[14] F. Firouzi, B. Farahani, and A. Marinšek, "The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT)," *Information Systems*, vol. 107, p. 101840, 2022.

[15] P. Hu, S. Dhelim, H.Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of network and computer applications*, vol. 98, pp. 27-42, 2017.

[16] P. Mach, and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE communications surveys & tutorials*, vol. 19, no. 3, pp. 1628-1656, 2017.

[17] T. Zebin, P. J. Scully, N. Peek, A.J. Casson, and K. B. Ozanyan, "Design and implementation of a convolutional neural network on an edge computing smartphone for human activity recognition," *IEEE Access*, vol. 7, pp. 133509-133520, 2019.

[18] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, and T. Zhou, "A survey on edge computing systems and tools," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1537-1562, 2019.

[19] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A survey of recent advances in edge-computing-powered artificial intelligence of things," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13849-13875, 2021.

[20] E. Covi, E. Donati, X. Liang, D. Kappel, H. Heidari, M. Payvand, and W. Wang, "Adaptive extreme edge computing for wearable devices," *Frontiers in Neuroscience*, vol. 15, p. 611300, 2021.

[21] C. Shin, "Variation-aware advanced CMOS devices and SRAM", vol. 56, Dordrecht: Springer Netherlands, 2016.

[22] J. Zheng, Z. Fang, C. Wu, S. Zhu, P. Xu, J.K. Doylend, S. Deshmukh, E. Pop, S. Dunham, M. Li, and A. Majumdar, "Nonvolatile electrically reconfigurable integrated photonic switch enabled by a silicon PIN diode heater," *Advanced Materials*, vol. 32, no. 31, p. 2001218, 2020.

[23] S. M. Khan, A. Gumus, J. M. Nassar, and M. M. Hussain, "CMOS enabled microfluidic systems for healthcare based applications," *Advanced Materials*, vol. 30, no. 16, p. 1705759, 2018.

[24] M. Capra, R. Peloso, G. Masera, M. Ruo Roch, and M. Martina, "Edge computing: A survey on the hardware requirements in the internet of things world," *Future Internet*, vol. 11, no. 4, p. 100, 2019.

[25] O. Krestinskaya, A.P. James, and L.O. Chua, "Neuromemristive circuits for edge computing: A review," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 1, pp. 4-23, 2019.

[26] R.G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253-266, 2010.

[27] K. Kai, W. Cong, and L. Tao, "Fog computing for vehicular ad-hoc networks: paradigms, scenarios, and issues," *The Journal of China Universities of Posts and Telecommunications*, vol. 23, no. 2, pp. 56-96, 2016.

[28] J. M. Shalf, and R. Leland, "Computing beyond moore's law," *Computer*, vol. 48, no. 12, pp. 14-23, 2015.

[29] A. Pavlov, and M. Sachdev, "CMOS SRAM circuit design and parametric test in nano-scaled technologies: process-aware SRAM design and test" , vol. 40, Springer Science & Business Media, 2008.

[30] D. Gschwend, "Zynqnet: An fpga-accelerated embedded convolutional neural network," *arXiv preprint arXiv:2005.06892*, 2020.

[31] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran, "The role of edge computing in internet of things," *IEEE communications magazine*, vol. 56, no. 11, pp. 110-115, 2018.

[32] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1-36, 2019.

[33] C. Jiang, T. Fan, H. Gao, W. Shi, L. Liu, C. Cérin, and J. Wan, "Energy aware edge computing: A survey," *Computer Communications*, vol. 151, pp. 556-580, 2020.

[34] S.Y. Park, K. Na, M. Vöröslakos, H. Song, N. Slager, S. Oh, J.P. Seymour, G. Buzsáki, and E. Yoon, "A miniaturized 256-channel neural recording interface with area-efficient hybrid integration of flexible probes and CMOS integrated circuits," *IEEE Transactions on Biomedical Engineering*, vol. 69, no. 1, pp. 334-346, 2021.

[35] S. Pelley, D. Meisner, T.F. Wenisch, and J.W. VanGilder, "Understanding and abstracting total data center power," in *Workshop on Energy-Efficient Design*, vol. 11, Jun. 2009, pp. 1-6.

[36] V. Sze, Y.H. Chen, T.J. Yang, and J.S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295-2329, 2017.

[37] Y. Liu, H. Pu, and D.W. Sun, "Efficient extraction of deep image features using convolutional neural network (CNN) for applications in detecting and analysing complex food matrices," *Trends in Food Science & Technology*, vol. 113, pp. 193-204, 2021.

[38] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

[39] A. K. Yadav, M. B. Sharma, A. K . Bhagat, D. H . Shah, M . C R, and M. A . Awasthi, "Edge Computing in Centralized Data Server Deployment for Network Qos and Latency Improvement for Virtualization Environment," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 14, no. 3, pp. 214-225, Jan. 2023.