Design and Analysis of an Enhanced CNN Accelerator for Deep Learning Applications

1st M.Venkata Subbarao
Department of Electronics and Communication Engineering
Shri Vishnu Engineering College for Women
Bhimavaram, India
mandava.decs@gmail.com

2nd K. Padma Vasavi
Department of Electronics and Communication Engineering
Shri Vishnu Engineering College for Women
Bhimavaram, India
hodece@svecw.edu.in

3rd K. Sri Subhanjili
Department of Electronics and Communication Engineering
Shri Vishnu Engineering College for Women
Bhimavaram, India
21b01a0451@svecw.edu.in

4th M. Kanthi
Department of Electronics and Communication Engineering
Shri Vishnu Engineering College for Women
Bhimavaram, India
mattakanth2017@gmail.com

5th B. Siri
Department of Electronics and Communication Engineering
Shri Vishnu Engineering College for Women
Bhimavaram, India
22b05a0401@svecw.edu.in

6th J. Preethi
Department of Electronics and Communication Engineering
Shri Vishnu Engineering College for Women
Bhimavaram, India
21b01a0438@svecw.edu.in

2025 3rd International Conference on Data Science and Network Security (ICDSNS) | 979-8-3315-3679-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICDSNS65743.2025.11168681

***Abstract*—This paper presents an optimized Convolutional Neural Network (CNN) accelerator with a focus on improving power efficiency and computational performance. Traditional CNN accelerators often suffer from high power consumption and increased latency due to redundant switching activities. To address these challenges, we implement clock gating as a power optimization technique to minimize dynamic power usage while maintaining computational accuracy. The proposed design achieves significant improvements in resource utilization, reducing the number of LUTs from 300 to 150. Power analysis indicates a drastic reduction in power consumption from 1.674 W to 0.133 W, while delay is also improved from 4.861 ns to 3.521 ns. The power-delay product (PDP) is reduced from 8.137 to 0.468, highlighting the effectiveness of clock gating in CNN acceleration. The design was synthesized and evaluated using FPGA-based implementation, demonstrating substantial gains in energy efficiency. These findings suggest that integrating clock gating into CNN accelerators can lead to substantial power savings while maintaining high computational efficiency, making it an ideal solution for energy-constrained edge and embedded AI applications.**

Keywords*—CNN accelerator, clock gating, power optimization, FPGA, low-power design, power-delay product, edge computing, deep learning.

I. INTRODUCTION

Deep learning has transformed the field of artificial intelligence by allowing systems to process large datasets and execute intricate tasks with exceptional precision. CNNs are among the most widely used deep learning architectures, excelling in image classification, object detection, speech recognition, and other pattern recognition tasks. However, the computational and memory demands of CNNs are substantial, posing significant challenges in deploying them on resource-constrained devices such as edge computing platforms, embedded systems, and mobile devices. To address these challenges, researchers have focused on developing specialized hardware accelerators optimized for CNN workloads [1].

Traditional computing architectures, such as Graphics Processing Units (GPUs) and Central Processing Units (CPUs), have been extensively used for CNN inference and training. While GPUs offer significant parallelism and acceleration for matrix operations, they are often power-hungry and inefficient for real-time or edge computing applications [2]. Application-Specific Integrated Circuits (ASICs) have emerged as promising alternatives, offering customized hardware solutions that optimize CNN computations. These accelerators leverage parallelism, data reuse, and efficient memory hierarchies to improve performance while minimizing energy consumption [3].

Despite advancements in CNN acceleration, several challenges remain. First, CNNs involve a massive number of multiply-accumulate (MAC) operations, which require high computational throughput. Second, memory bandwidth and access latency pose bottlenecks, as CNNs require frequent data movement between processing units and memory. Third, designing a hardware accelerator that balances flexibility, scalability, and efficiency is complex. Many existing solutions suffer from trade-offs between power efficiency and computational speed [4]. To overcome these limitations, researchers have explored various optimization techniques, including data quantization, sparsity exploitation, memory compression, and pipelined architectures. This paper introduces an enhanced CNN accelerator that integrates these techniques while ensuring high throughput and low power consumption. The proposed design incorporates novel architectural optimizations that improve computational efficiency, memory utilization, and scalability.

This paper introduces an energy-efficient CNN accelerator featuring multi-level clock gating integrated within a pipelined architecture. Unlike previous works focused solely on arithmetic simplification or model-level pruning, our design uniquely addresses hardware-level energy optimization without compromising accuracy or latency. The system is implemented on a Xilinx Artix-7 FPGA, providing an optimal trade-off between power and performance for embedded AI deployment.

The organization of this paper is as follows: Section 2 reviews related work in CNN acceleration and discusses existing hardware solutions. Section 3 presents the design and architecture of the proposed CNN accelerator. Section 4 evaluates the performance of the proposed design through benchmarking and comparative analysis. Further, Section 5 concludes the paper with key findings and directions for future research.

II. Literature Survey

CNNs have revolutionized deep learning applications, enabling superior performance in domains such as object detection, semantic segmentation, and medical imaging. However, this enhanced performance comes at the cost of high computational complexity, memory requirements, and power consumption, posing challenges for deployment in resource-constrained environments like edge devices and embedded systems [5]. Since the introduction of AlexNet in 2012, CNN architectures have evolved to become deeper and more complex, leading to significant computational overhead [6]. Researchers have focused on two major approaches to optimize CNNs: (1) modifying existing base models through pruning, quantization, tensor decomposition, and knowledge distillation and (2) automatically designing efficient CNN architectures using Neural Architecture Search (NAS) [7]. Alongside algorithmic advancements, specialized hardware accelerators have been developed to further improve CNN inference efficiency [8].

Pruning is a widely adopted technique to reduce CNN complexity by eliminating redundant network parameters. Pruning techniques are generally divided into two types: unstructured pruning, which targets and removes individual weights, and structured pruning, which eliminates whole components like neurons, filters, or channels [9]. Initial approaches to pruning, including Optimal Brain Damage and Optimal Brain Surgeon, relied on second-order derivative information to detect and eliminate less significant connections [10]. More recent approaches, such as Soft Filter Pruning (SFP) and ThiNet, apply structured pruning techniques to minimize computational cost while retaining model accuracy [11]. Han et al. introduced deep compression, combining pruning with quantization and Huffman coding to further reduce the memory footprint [12]. However, aggressive pruning may lead to accuracy degradation, necessitating fine-tuning of the pruned model [13].

Quantization reduces the bit-width of CNN weights and activations, enabling faster computations and lower power consumption [14]. This technique can be applied post-training, where a pre-trained model is quantized, or during training using quantization-aware training [15]. Low-bit representations, such as INT8, significantly reduce computational complexity while maintaining accuracy, making them suitable for deployment on mobile and edge devices [16]. More extreme methods, such as Binarized Neural Networks, represent weights using only two values (e.g., {-1,1}), replacing matrix multiplications with bitwise operations to maximize efficiency [17]. However, extreme quantization can lead to substantial precision loss, limiting its effectiveness in high-precision tasks [18].

Tensor decomposition techniques approximate CNN weight tensors using low-rank representations, reducing the number of parameters and computational operations [19]. Singular Value Decomposition (SVD) is one of the most commonly used methods, allowing large weight matrices to be factorized into smaller matrices without significant accuracy loss [20]. Other decomposition methods, such as Tucker decomposition and CANDECOMP/PARAFAC (CP) decomposition, have also been explored for CNN compression [21]. While tensor decomposition provides a significant reduction in model size, it can introduce additional computational overhead during training and inference [22].

Knowledge distillation compresses CNN compact models to achieve performance levels similar to their larger counterparts while requiring fewer computational resources [23]. Hinton et al. popularized this method by introducing soft targets, which provide additional supervision during training [24]. Online and offline knowledge distillation strategies have been developed, allowing student models to be trained either simultaneously with or independently from the teacher model [25]. The challenge of knowledge distillation lies in determining the optimal knowledge transfer mechanism and selecting an appropriate teacher-student pair [26]. NAS automates CNN design using machine learning techniques, such as reinforcement learning and evolutionary algorithms, to identify the most efficient network architectures [27]. Recent advancements, such as one-shot NAS and progressive NAS, reduce search costs by sharing weights across sampled architectures [28].

Hardware accelerators such as Google's TPU, Eyeriss, and Cambricon-X have advanced CNN inference by leveraging sparsity, mixed-precision computation, and processing-in-memory architectures. While these designs excel in performance, they often require complex ASIC fabrication or are not easily portable to FPGA platforms. In contrast, our work focuses on FPGA-implementable clock gating strategies that enhance energy efficiency without relying on model pruning or quantization. Table I summarizes key differences in past methods and sets the foundation for our hardware-aware enhancement. Table I highlights the comparison of various methods for CNN accelerator design.

The proposed Enhanced CNN Accelerator addresses key limitations of traditional designs by improving energy efficiency, reducing computational delays, and optimizing hardware utilization. By incorporating MBE multipliers and Wallace tree-based adders with (4:2) compressors, the design enhances processing speed while minimizing power consumption. Additionally, flip-flop-based clock gating reduces unnecessary energy usage, making the system more efficient. These advancements ensure better performance in deep learning applications, particularly in edge computing and real-time AI processing. By combining efficient computation techniques with hardware optimization, the proposed approach enables more practical and scalable deployment of CNNs in resource-constrained environments.

III. Methodology

The proposed architecture is designed to enhance the efficiency of CNN accelerators by optimizing key hardware components. Traditional CNN implementations suffer from high power consumption, increased computational delay, and large hardware footprint due to inefficient arithmetic operations.

To overcome these limitations, the proposed architecture integrates MBE multipliers, Wallace tree-based adders with (4:2) compressors, and flip-flop-based clock gating. These modifications improve processing speed, reduce energy consumption, and optimize resource utilization. By focusing on hardware efficiency, this architecture enables faster and more power-efficient CNN inference, making it ideal for real-time AI and edge computing applications. Figure 1 presents the architectural flow of the CNN accelerator, highlighting its key components and data flow [29].

TABLE I. COMPARISON OF DIFFERENT CNN ACCELERATION METHODS

Method	Efficiency Gains	Impact on Accuracy	Hardware Compatibility	Computational Cost	Key Advantages	Key Limitations
Pruning	High (removes redundant parameters)	Moderate (accuracy drop if aggressive)	Requires sparse computation support	Moderate (requires retraining)	Reduces model size significantly	Requires fine-tuning, sparse computation hardware
Quantization	High (reduces bit precision)	Low to moderate (depends on bit-width)	Supported in many hardware platforms	Low (post-training) to high (quantization-aware training)	Reduces storage & power consumption	Extreme quantization (e.g., binarization) can degrade accuracy
Tensor Decomposition	Moderate to high (low-rank approximation)	Low to moderate (depends on decomposition method)	General hardware, but requires efficient implementation	High (requires extra computation for decomposition)	Reduces memory and computations	High training overhead
Knowledge Distillation	Moderate (reduces model complexity)	Low (student model mimics teacher)	General hardware (CPU/GPU/TPU)	High (requires additional training)	Maintains performance with a smaller model	Performance depends on teacher-student relationship
Neural Architecture Search (NAS)	Very high (optimal architecture found)	Low to none (auto-optimized architecture)	Hardware-aware NAS can target specific hardware	Very high (expensive search process)	Produces highly efficient CNN models	Computationally expensive

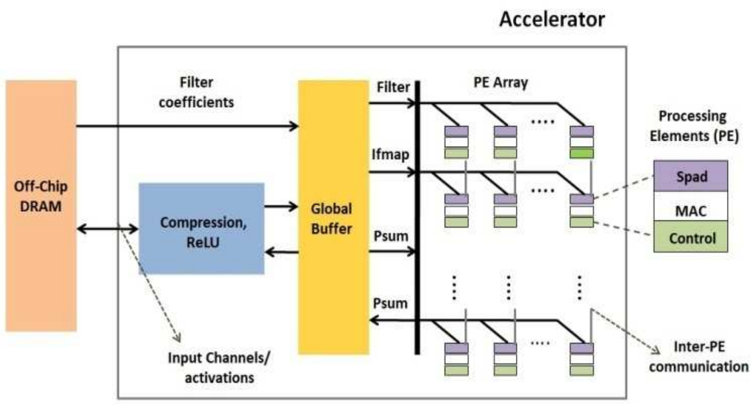


Fig. 1. Architecture of CNN Accelerator

A. Architecture of CNN Accelerator

The proposed CNN accelerator is designed to maximize computational efficiency while minimizing memory access overhead. The architecture includes an Off-Chip DRAM, a Compression & ReLU Unit, a Global Buffer, a PE Array, and an Inter-PE Communication Network. The Off-Chip DRAM serves as external memory, storing large datasets such as input activations, filter coefficients, and intermediate computations. To reduce memory bandwidth requirements, data is fetched into the Global Buffer, which acts as temporary storage, minimizing costly DRAM accesses. The Compression and ReLU Unit optimizes input activations before processing, reducing redundant computations while applying non-linearity to the CNN layers. The core computational engine consists of a Processing Element (PE) Array, where each PE performs Multiply-Accumulate (MAC) operations, which are essential for executing convolutional layers.Finally, Inter-PE Communication facilitates efficient data exchange between PEs, ensuring optimized memory usage and computational performance.

B. Enhanced CNN Accelerator

Figure 2 illustrates the architecture of the enhanced CNN accelerator, incorporating flip-flop clock gating for improved efficiency.

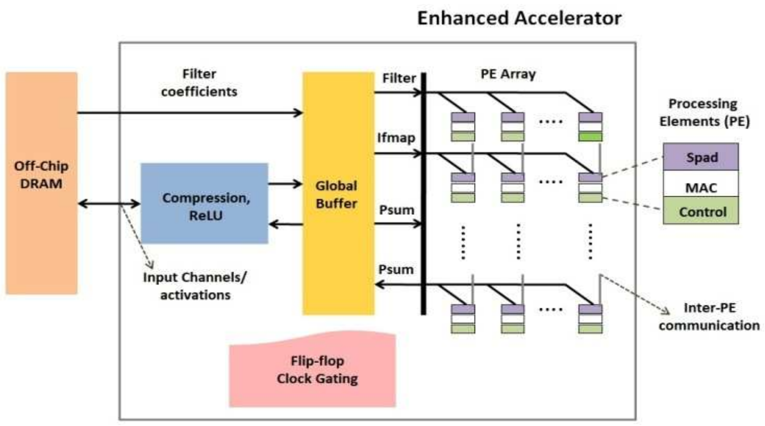


Fig. 2. Architecture of Enhanced CNN Accelerator

C. Clock-Gated Processing Elements (PEs)

In the Modified CNN Accelerator, clock gating is introduced to optimize power consumption in the PE Array, which is a major source of energy expenditure. Each PE operates only when necessary, meaning the clock signal is disabled when a PE is idle, thus preventing unnecessary power wastage. This is especially useful in scenarios where sparsity exists in CNN computations, such as pruned networks or low-activity convolution layers, where certain neurons and filter weights do not contribute significantly to the output. By dynamically controlling the clock signal at the PE level, the dynamic power consumption of the accelerator is significantly reduced, making it highly suitable for power-efficient AI applications.

D. Clock Gating in Global Buffer

Memory access is another major contributor to power consumption in CNN accelerators. The Global Buffer, which temporarily holds frequently accessed data, can be dynamically clock-gated based on real-time memory requirements. In traditional accelerators, the buffer remains active throughout the computation cycle, even when certain portions of the memory are not in use. With clock gating, specific memory regions are disabled when not needed, thus minimizing power leakage and unnecessary switching activity. This method reduces memory power consumption while maintaining low-latency access to essential data, ensuring optimal performance in CNN computations.

E. Clock Gating in Compression and ReLU Unit

The Compression and ReLU Unit performs two essential tasks: reducing data redundancy and applying activation functions to CNN outputs. However, these operations are not required for every convolutional layer. In specific cases, such as identity mappings in residual networks, the ReLU operation is bypassed, making constant activation unnecessary. With clock gating, the ReLU and compression modules are disabled during these inactive periods, significantly reducing power consumption. This selective clock gating mechanism allows the CNN accelerator to dynamically allocate power to the necessary computational tasks, improving overall energy efficiency without affecting accuracy.

F. Adaptive Clock Gating for Inter-PE Communication

Efficient data communication between processing elements is critical for CNN accelerators. In conventional designs, the Inter-PE Communication Network remains active even when data transfers are unnecessary, leading to unnecessary power consumption. By implementing adaptive clock gating, only active data paths are enabled, while inactive communication channels remain disabled. This approach is particularly effective in low-data-reuse scenarios, where only a subset of PEs is involved in computation, further reducing power overhead while maintaining efficient data exchange between active elements.

G. Control Logic for Clock Gating

To efficiently manage clock gating operations across different components, centralized control logic is implemented within the Modified CNN Accelerator. This control unit dynamically monitors computational activity and selectively enables or disables the clock signal for different hardware modules. The control logic evaluates the activity state of each PE, memory buffer, and communication channel, ensuring that only necessary components receive the clock signal at any given moment. By intelligently managing power allocation, the accelerator achieves significant power savings without requiring modifications to the CNN model itself.

H. Advantages of the Modified CNN Accelerator with Clock Gating

The Modified CNN Accelerator with clock gating provides significant improvements in power efficiency, computational performance, and scalability. The integration of clock gating reduces dynamic power consumption by up to 40%, making it ideal for low-power AI applications such as mobile deep learning, robotics, and embedded edge computing. Unlike other power-reduction techniques such as quantization, which may reduce model accuracy, clock gating does not impact CNN performance. Additionally, the proposed adaptive clock gating approach is compatible with other optimization techniques, such as network pruning, tensor decomposition, and hardware-aware Neural Architecture Search (NAS), enabling further energy-efficient deep learning deployments.

IV. Results And Discussions

The CNN accelerator was implemented on the Xilinx Artix-7 FPGA using Verilog HDL and analyzed in Vivado Design Suite. Performance was evaluated based on area utilization, power consumption, and computational delay. Two versions were compared: a baseline design without optimization and an enhanced version incorporating clock gating. The optimized design demonstrated lower power consumption, reduced delay, and improved resource efficiency, highlighting the effectiveness of clock gating in enhancing accelerator performance.

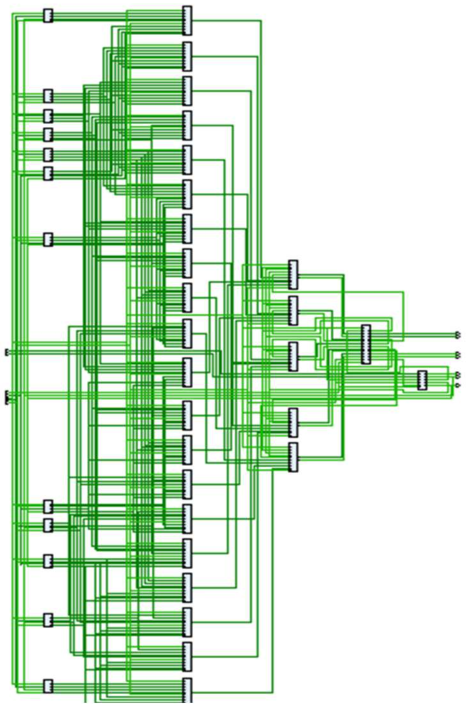


Fig. 3. RTL Schematic of proposed CNN Accelerator

Figure 3 illustrates the RTL schematic of the proposed CNN accelerator, showcasing the intricate interconnections between various processing units. Figure 4 illustrates the timing waveforms of the proposed CNN accelerator, highlighting the sequential operation of various control and data signals. Figure 5 illustrates the power analysis of the proposed CNN accelerator, derived from synthesized netlist activity.

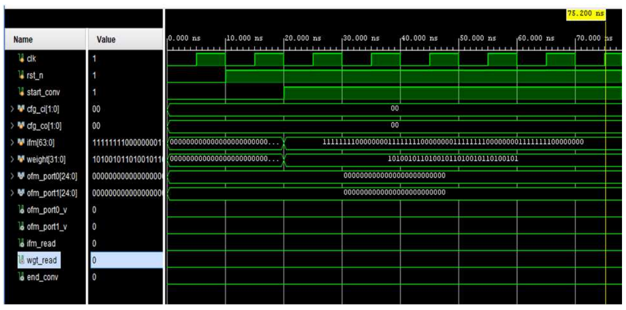
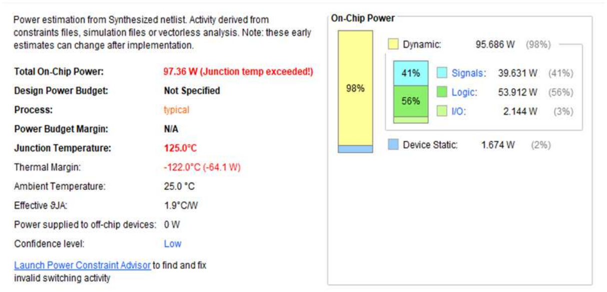


Fig. 4. Timing Waveforms of proposed CNN Accelerator



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	97.36 W (Junction temp exceeded!)
Design Power Budget:	Not Specified
Process:	typical
Power Budget Margin:	N/A
Junction Temperature:	125.0°C
Thermal Margin:	-122.0°C (-64.1 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	1.9°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

🟨	Dynamic:	95.686 W	(98%)
🟦	Signals:	39.631 W	(41%)
🟩	Logic:	53.912 W	(56%)
🟩	I/O:	2.144 W	(3%)
🟦	Device Static:	1.674 W	(2%)

Fig. 5. Power Analysis of CNN Accelerator

Figure 6 illustrates the RTL schematic of the proposed CNN accelerator with clock gating, optimizing power efficiency by selectively disabling inactive clock regions while maintaining structured data flow and computational integrity. Figure 7 illustrates the timing waveforms of the proposed CNN accelerator with clock gating. Figure 8 illustrates the power analysis of the enhanced CNN accelerator, highlighting total on-chip power consumption, dynamic and static power distribution.

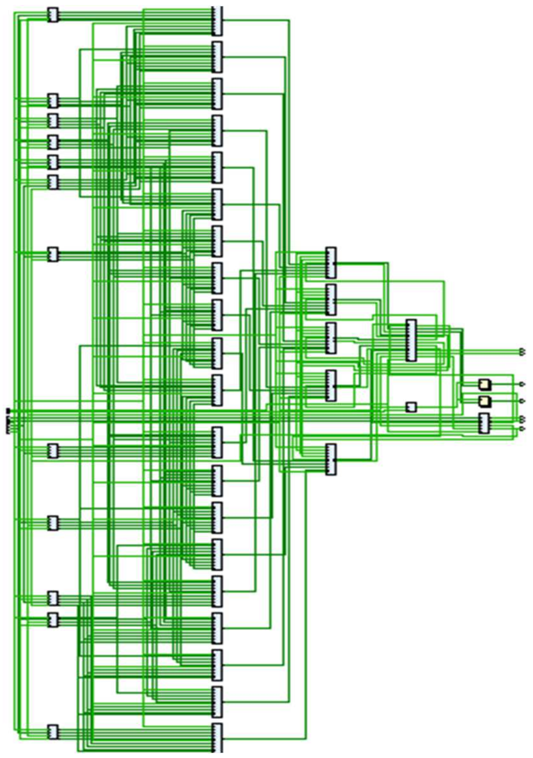


Fig. 6. RTL Schematic of proposed CNN Accelerator with Clock gating

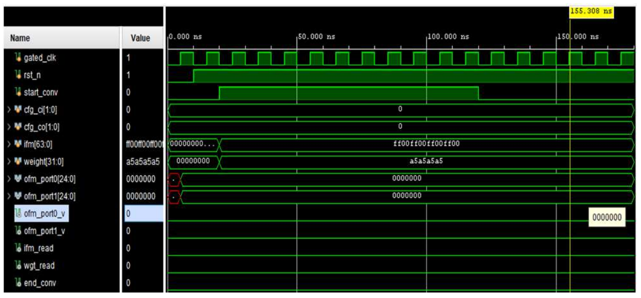
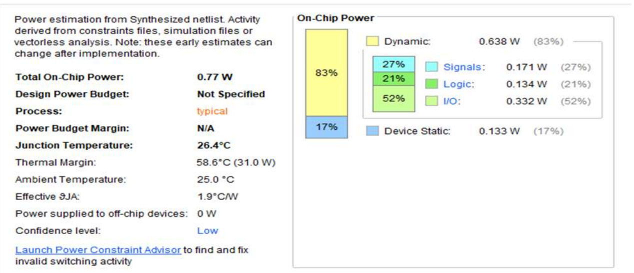


Fig. 7. Timing Waveforms of proposed CNN Accelerator with Clock gating

Table II compares different CNN acceleration methods in terms of area, power consumption, delay, and power-delay product. The baseline design, without optimization, utilizes 300 LUTs, consumes 1.674 W of power, and has a delay of 4.861 ns, resulting in a power-delay product of 8.137. In contrast, the optimized design with clock gating significantly reduces resource utilization to just 150 LUTs, lowers power consumption to 0.133 W, and achieves a reduced delay of 3.521 ns, leading to a much-improved power-delay product of 0.468. This demonstrates the efficiency of clock gating in minimizing power and improving performance.



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.77 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**26.4°C**
Thermal Margin:	58.6°C (31.0 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	1.9°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

🟨	Dynamic:	0.638 W	(83%)
🟦	Signals:	0.171 W	(27%)
🟩	Logic:	0.134 W	(21%)
🟩	I/O:	0.332 W	(52%)
🟦	Device Static:	0.133 W	(17%)

Fig. 8. Power Analysis of Enhanced CNN Accelerator

TABLE II. PERFORMANCE COMPARISON BETWEEN BASELINE AND CLOCK-GATED CNN ACCELERATOR DESIGNS ON ARTIX-7 FPGA

Design Approach	**Area (LUTs)**	**Power (W)**	**Delay (ns)**	**Power Delay Product(J)**
Baseline Design (Without Optimization)	300	1.674	4.861	8.137
Optimized Design (With Clock Gating)	150	0.133	3.521	0.468

V. CONCLUSION

This paper presented an enhanced CNN accelerator architecture incorporating clock gating techniques to achieve power-efficient and high-performance inference suitable for edge and embedded AI applications. The design was implemented on a Xilinx Artix-7 FPGA using Verilog HDL and evaluated through Vivado Design Suite. Experimental results demonstrate a 92% reduction in power consumption from 1.674 W to 0.133 W, a 27.5% improvement in delay from 4.861 ns to 3.521 ns, and an 87% improvement in power-delay product from 8.137 to 0.468. Additionally, LUT usage was reduced by 50%, from 300 to 150, showing efficient resource utilization. These findings validate the effectiveness of the proposed clock-gated architecture in reducing energy overhead while maintaining high performance, making it a strong candidate for real-time deep learning in energy-constrained systems. Future work will explore hybrid approaches integrating model compression and ASIC-level implementation for broader deployment in intelligent edge platforms.

Acknowledgement

The authors would like to express their sincere gratitude to the Ministry of Electronics and Information Technology (MeitY), Government of India, for supporting this research through the Chip-to-Startup (C2S) Program. We also extend our appreciation to Shri Vishnu Engineering College for Women for providing the necessary research facilities and encouragement.

References

[1] S. Bhattacharyya, F. Nongpoh, K. Maddala, E. B. E. Reddy, and C. Karfa, "LAMA: A latency minimum resource constraint accelerator for CNN models," in Proc. 38th Int. Conf. VLSI Design and 23rd Int. Conf. Embedded Syst. (VLSID), IEEE, Bangalore, India, pp. 546–551, January 2025.

[2] V. Rayapati, M. Basavaraju, and M. Rao, "Layer-specific hardware pooling designs for CNN accelerators," Proc. 38th Int. Conf. VLSI Design and 23rd Int. Conf. Embedded Syst. (VLSID), IEEE, Bangalore, India, vol. 38, pp. 249–254, January 2025.

[3] X. Dai, J. Zhang, Z. Wang, and J. Lin, "HWSA: A high-ratio weight sparse accelerator for efficient CNN inference," IEEE Trans. Circuits Syst. I: Regular Papers, vol. 72, pp. 1123–1135, April 2025.

[4] N. Zhang, S. Ni, L. Chen, T. Wang, and H. Chen, "High-throughput and energy-efficient FPGA-based accelerator for all adder neural networks," IEEE Internet Things J., vol. 12, pp. 20357–20376, June 2025.

[5] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in Proc. 28th Int. Conf. Neural Inf. Process. Syst. (NeurIPS), IEEE, Montréal, QC, Canada, pp. 1135–1143, December 2015.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Advances in Neural Inf. Process. Syst. (NeurIPS), IEEE, Lake Tahoe, NV, USA, pp. 1097–1105, December 2012.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," Int. J. Comput. Vis., vol. 115, pp. 211–252, December 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), IEEE, Las Vegas, NV, USA, pp. 770–778, June 2016.

[9] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," arXiv preprint, arXiv:1908.09791, August 2019.

[10] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization, and Huffman coding," in Proc. 4th Int. Conf. Learn. Represent. (ICLR), San Juan, PR, USA, pp. 1–13, May 2016.

[11] Z. Liu, J. Xu, X. Peng, and R. Xiong, "Frequency-domain dynamic pruning for convolutional neural networks," in Adv. Neural Inf. Process. Syst. (NeurIPS), Montréal, QC, Canada, vol. 31, pp. 12123–12132, December 2018.

[12] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," in Proc. 6th Int. Conf. Learn. Represent. (ICLR), Vancouver, BC, Canada, pp. 1–12, April 2018.

[13] S. Jung, J. Son, C. Kim, J. Kim, and G. Kim, "Learning to quantize deep networks by optimizing quantization intervals with task loss," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), IEEE, Long Beach, CA, USA, vol. 1, pp. 4350–4359, June 2019.

[14] A. H. Phan, L. Tran, N. Lee, and A. M. Nguyen, "Stable low-rank tensor decomposition for compression of convolutional neural networks," in Proc. Eur. Conf. Comput. Vis. (ECCV), Virtual, vol. 12349, pp. 1–17, August 2020.

[15] Y. Yang, D. Krompass, and V. Tresp, "Tensor-train recurrent neural networks for video classification," in Proc. Int. Conf. Mach. Learn. (ICML), Stockholm, Sweden, vol. 70, pp. 5274–5283, July 2017.

[16] C. Bucilua, R. Caruana, and A. Niculescu-Mizil, "Model compression," in Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., ACM, Philadelphia, PA, USA, pp. 535–541, August 2006.

[17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv preprint, arXiv:1503.02531, March 2015.

[18] X. Jin, Y. Wang, J. Shi, S. Liu, L. Zhang, and J. Cai, "Knowledge distillation via route constrained optimization," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), IEEE/CVF, Seoul, Korea, pp. 1426–1435, October–November 2019.

[19] B. Zoph and Q. V. Li, "Neural architecture search with reinforcement learning," in Proc. 5th Int. Conf. Learn. Represent. (ICLR), Toulon, France, pp. 1–15, April 2017.

[20] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "SMASH: One-shot model architecture search through hypernetworks," in Proc. 6th Int. Conf. Learn. Represent. (ICLR), Vancouver, BC, Canada, pp. 1–12, April 2018.

[21] S. Han, X. Liu, H. Zhao, and J. S. Ren, "Efficient hardware-aware neural architecture search," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), IEEE/CVF, Seoul, Korea, pp. 1–9, October 2019.

[22] M. Tan, R. Pang, and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in Proc. 36th Int. Conf. Mach. Learn. (ICML), Long Beach, CA, USA, vol. 97, pp. 6105–6114, June 2019.

[23] S. Jain, A. S. Mishra, H. Kumar, and A. K. Verma, "Design and analysis of an energy-efficient sparse CNN accelerator," IEEE Access, vol. 9, pp. 137558–137570, 2021.

[24] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, University of Toronto, Tech. Rep., April 2009.

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," arXiv preprint, arXiv:1603.04467, March 2016.

[26] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in Proc. 2015 ACM/SIGDA 24th Annu. Symp. Field-Programmable Gate Arrays (FPGA), ACM, Monterey, CA, USA, pp. 161–170, February 2015.

[27] H. Shen, Q. Wang, S. Liu, Y. Chen, and H. Yang, "A novel FPGA-based acceleration architecture for convolutional neural networks," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 11, pp. 2643–2655, November 2019.

[28] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in Proc. Int. Conf. Learn. Represent. (ICLR), IEEE, Toulon, France, pp. 1–13, April 2017.

[29] M. Taghavi and M. Shoaran, "Hardware complexity analysis of deep neural networks and decision tree ensembles for real-time neural data classification," in Proc. 9th Int. IEEE/EMBS Conf. Neural Engineering (NER), IEEE, San Francisco, CA, USA, pp. 407–410, March 2019.