Design and Implementation of Max-Max Pooling Layer Through ASIC for CNN Applications

G. Tejaswini
Dept. of Electronics and Communication Engineering GMR Institute of Technology
Rajam, Vizianagaram, Andhra Pradesh, India - 532127
22341A0466@gmrit.edu.in

M. Akhila
Dept. of Electronics and Communication Engineering GMR Institute of Technology
Rajam, Vizianagaram, Andhra Pradesh, India - 532127
22341A04A1@gmrit.edu.in

K. Ravindra
Dept. of Electronics and Communication Engineering GMR Institute of Technology
Rajam, Vizianagaram, Andhra Pradesh, India - 532127
22341A0496@gmrit.edu.in

L. Sai Sri Vedavyas
Dept. of Electronics and Communication Engineering GMR Institute of Technology
Rajam, Vizianagaram, Andhra Pradesh, India - 532127
22341A0499@gmrit.edu.in

***Abstract*—Pooling layers, especially max pooling, are vital in deep learning architectures for reducing spatial dimensions and preserving salient features. This work proposes a custom ASIC-based hardware architecture for a max-max pooling layer, designed to improve the performance and efficiency of convolutional neural networks (CNNs) in resource-constrained environments. The design is implemented in SystemVerilog and validated through comprehensive testbench simulations. It employs a two-stage hierarchical max pooling operation using multiplexers, D flip-flops, and comparators. The core logic supports multiple operational modes via a 2-bit select line, enabling direct input loading, incremental/decremental updates, and default override functionality. A comparator module evaluates threshold conditions in real time, enhancing decision-making within the pooling logic. The design ensures minimal latency and synchronous operation with a clocked control system, optimizing hardware utilization. Simulation results confirm functional correctness and demonstrate the feasibility of deploying this architecture in ASIC-based real-time AI applications, such as edge computing and embedded vision systems.**

***Index Terms*—ASIC Design, CNN Acceleration, Comparator, Custom Pooling Layer, D Flip-Flop, Deep Learning Hardware, Digital Circuit Design, Hardware Architecture, Max-Max Pooling, Multiplexer, Edge AI, Real-Time Processing, SystemVerilog, Synchronous Logic, Embedded AI Systems**

I. Introduction

Convolutional Neural Networks (CNNs) are a class of deep learning models widely used for processing grid-like data such as images. They mimic the human visual cortex and are effective at learning hierarchical representations of features, making them invaluable for tasks like image classification and object detection. CNNs use convolutional layers to extract features and pooling layers to downsample feature maps. Pooling layers reduce spatial dimensions while retaining important information, which lowers computation and memory needs. Common pooling methods include max pooling (selecting the maximum value in each window) and average pooling (computing the average). Max pooling is especially popular because it highlights the strongest activations and is simple to implement in hardware using comparators and multiplexers.

While conventional max pooling processes one stage of selection, a multi-stage approach can further enhance feature retention. *Max-Max* pooling extends the traditional max pooling by applying the max operation iteratively across a hierarchy. For example, it may first apply max pooling within local patches and then apply another max pooling stage over the pooled results. This hierarchical procedure tends to preserve the most discriminative features and suppress noise, improving robustness in deep networks. However, it comes at the cost of added hardware and slightly increased latency. Figure 2 illustrates the conceptual block diagram of a two-stage max-max pooling process.

Implementing pooling in hardware such as an ASIC (Application-Specific Integrated Circuit) poses unique considerations. Max pooling is well-suited to hardware design since it relies on simple comparisons, avoiding complex arithmetic required by average pooling. This allows efficient use of logic gates and faster operation. The goal of this work is to design an optimized ASIC-friendly architecture for max-max pooling that achieves high efficiency while maintaining correct functionality. The proposed design provides configurable modes and supports real-time operation, making it suitable for embedded and edge AI applications.

II. Literature Survey

Recent research has explored various advanced pooling strategies and hardware accelerators for CNNs. More *et al.* proposed a *Horizontal Max Pooling* scheme to reduce noise in feature maps by applying max pooling across extended regions, improving feature detection accuracy [3]. In another study, an area-efficient CNN accelerator was introduced that supports global average pooling with arbitrary shapes, significantly reducing gate count in the pooling module for IoT applications [4]. These works highlight the importance of

custom pooling operations and hardware co-design for efficient inference.

Other efforts have focused on novel hardware-friendly pooling layers. For example, researchers have designed new pooling methods such as *Binary Weighted Absolute Deviation (BWAD)* and *Absolute Maximum Deviation (AMD)*. These methods use simple logic (no multipliers or dividers) to achieve low area and power in hardware while maintaining accuracy comparable to standard max pooling [5]. The proposed BWAD and AMD pooling schemes were validated on FPGA and ASIC platforms, showing substantial area-delay-product and power-delay-product improvements. Such studies demonstrate that alternative pooling architectures can yield better hardware efficiency without sacrificing neural network performance.

In summary, these works suggest that customizing pooling operations and optimizing them for hardware can greatly benefit CNN accelerators. However, the specific concept of cascading multiple max pooling stages in a compact ASIC design has not been extensively addressed. This motivates the present methodology: a hybrid max-max pooling layer designed in SystemVerilog, intended to combine analog comparator efficiency with digital control for low-latency, low-power performance.

III. Methodology

The proposed hardware architecture implements a two-stage max-max pooling operation. The design reuses basic max pooling elements (comparators, multiplexers, and D flip-flops) in a cascaded fashion. Figure 1 shows the block diagram of the basic max pooling unit for four inputs. This unit compares pairs of input values using comparators and selects the larger value through a multiplexer, storing it in an output register on each clock cycle. Extending this, the complete design (Fig. 2) consists of two such pooling stages in series. In the first stage, a comparator tree finds the maximum of the raw input values. The second stage then compares this intermediate maximum with a value stored in a register, effectively performing a second max pooling operation.

To control the operation, a 2-bit mode select signal is provided. Depending on the mode, the circuit can operate as follows:

- **Load Mode:** New input values are loaded into registers directly.
- **Increment/Decrement Modes:** The stored maximum value can be incrementally updated (e.g., if inputs change slightly) or decremented, allowing the layer to adapt over time without clearing internal state.
- **Default Mode:** A default or override value can be output regardless of inputs.

A separate comparator monitors the final maximum against a programmable threshold. If the pooled value exceeds this threshold, an output flag is asserted. This allows dynamic decision logic (for example, triggering activation) based on the pooling result.

All operations are synchronous with the system clock. D flip-flops latch intermediate results so that on each positive clock edge, the next stage of pooling can occur. This pipelined, clocked approach ensures minimal combinational delay and allows continuous data flow.

The architecture was coded in SystemVerilog. The core components include comparator trees for pairwise maximum selection, 4-to-1 multiplexers to route the correct maximum, D flip-flops for state retention, and control logic for the 2-bit select. Figure 1 illustrates the basic max-pooling unit, and Figure 2 illustrates the complete two-stage max-max pooling architecture. The design was synthesized under a 90 nm CMOS technology with a target clock period of 4 ns (250 MHz). Extensive simulation testbenches were developed to verify functionality under various input patterns and to evaluate timing and power.

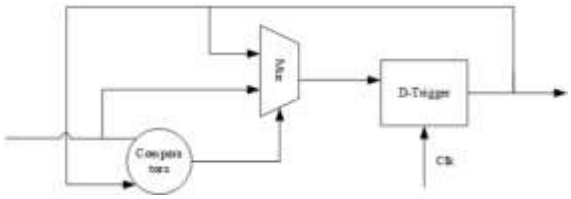

Fig. 1. Hardware architecture of the basic max pooling layer.

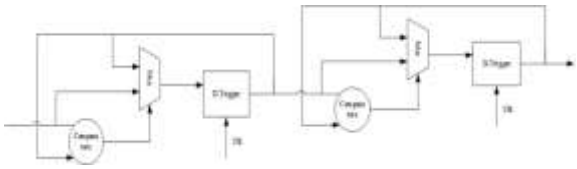

Fig. 2. Hardware architecture of the proposed two-stage max-max pooling layer.

IV. Results and Discussion

The design was synthesized and simulated using Cadence tools on a 90 nm CMOS process. Timing reports show that the design meets the 4 ns clock constraint with a slack of approximately +0.23 ns, indicating timing closure. Simulation waveforms (Figure 3) confirm correct max-max pooling behavior: in each cycle, the maximum of the inputs (and subsequently the maximum of the intermediate result with its stored value) is output. Notably, when the enable signal is high, the pooling operation proceeds; when low, outputs hold their reset values.

The following table summarizes key implementation metrics obtained from synthesis reports:

For comparison, Table II contrasts the traditional single-stage max pooling with the proposed two-stage max-max pooling. The two-stage approach yields higher feature quality and

TABLE I
POWER, AREA, AND TIMING METRICS OF THE MAX-MAX POOLING DESIGN

Metric	Value
Cell count	4820
Total area	35400 μm^2
Dynamic power	980 μW
Leakage power	37.2 μW
Total power	1.0172 mW
Timing slack (4 ns)	+0.23 ns

robustness at the cost of increased complexity and hardware resources.

TABLE II
COMPARISON OF MAX POOLING AND MAX-MAX POOLING

Aspect	Max Pooling	Max-Max Pooling
Stages	Single-stage	Two-stage (cascaded)
Operation	Selects max of local inputs	max pooling twice
Feature Quality	Good	Better (more refined)
Robustness	Moderate (noise sensitive)	Higher (filters noise)
Complexity	Lower (less hardware)	Higher (slightly latency)
Use Case	Low-power or early layers	Accuracy-deep layers

Overall, the results demonstrate that the proposed architecture correctly implements max-max pooling with minimal delay overhead. The additional comparator stage increases selectivity, which can improve accuracy in noisy environments, while remaining feasible in terms of area and power for ASIC deployment. Further experiments with larger pooling windows or multi-channel inputs would be needed to fully characterize performance in various CNN models.

Fig. 3. Simulation waveform illustrating the output of the max-max pooling layer over time.

V. CONCLUSION

This work presents a mixed-signal ASIC implementation of a two-stage max-max pooling layer for CNN acceleration. The design successfully combines digital control elements (flip-flops and multiplexers) with comparator circuits to achieve efficient downsampling. Cadence simulations in 90 nm CMOS confirm that the pooling operation functions as intended: the comparator outputs consistently identify the maximum values, and the flip-flops maintain state correctly across clock cycles. The 90 nm process provides adequate precision (outputs stable to several decimal places) and a good power-performance balance for embedded vision tasks. The dual-comparator architecture allows concurrent comparisons of multiple inputs, enabling the multi-stage pooling essential for this approach, while the clocked control logic ensures proper timing.

Key areas for further work include expanding the set of test vectors (varying input patterns, noise, and edge cases) to fully verify corner-case behavior, as well as performing post-layout simulations to account for interconnect effects. Additionally, analyzing power consumption under different operating modes and comparing to a pure digital implementation would provide insight into efficiency gains. The success of this design demonstrates a viable analog–digital hybrid approach for on-chip CNN accelerators, potentially mitigating the von Neumann bottleneck by leveraging analog comparisons for data-intensive operations.

VI. FUTURE SCOPE

Future work could integrate this max-max pooling module more tightly with convolutional layers on the same chip, creating a unified CNN accelerator. Supporting programmable pooling window sizes would make the design adaptable to different network architectures (e.g., 3x3, 5x5 kernels). Exploring higher precision comparators or configurable thresholds could improve applicability to networks requiring finer granularity. Finally, porting the design to newer process nodes (e.g., 28 nm or below) and exploring its behavior in a full CNN pipeline would further validate its utility in real-world embedded AI systems.

REFERENCES

[1] F. Yan, Z. Zhang, Y. Liu, and J. Liu, "Design of convolutional neural network processor based on FPGA resource multiplexing architecture," *Sensors*, vol. 22, p. 5967, 2022, doi:10.3390/s22165967.

[2] Y.-H. Huang, P.-H. Kuo, and J.-D. Huang, "Hardware-friendly activation function designs and its efficient VLSI implementations for transformer-based applications," in *Proc. 2023 IEEE 5th Int. Conf. on Artificial Intelligence Circuits and Systems (AICAS)*, Hangzhou, China, 2023, pp. 1–5, doi:10.1109/AICAS57966.2023.10168591.

[3] Y. More, K. Dumbre, and B. Shiragapur, "Horizontal max pooling: A novel approach for noise reduction in max pooling for better feature detection," in *Proc. 2023 IEEE East-West Design, Systems & IoT Conf. (ESCI)*, Dindigul, India, 2023, pp. 1–5, doi:10.1109/ESCI56872.2023.10099648.

[4] Y. Bai *et al.*, "An area-efficient CNN accelerator supporting global average pooling with arbitrary shapes," in *Proc. 2024 IEEE 5th Int. Conf. on AICAS*, Abu Dhabi, UAE, 2024, pp. 413–416, doi:10.1109/AICAS59952.2024.10595877.

[5] V. Rayapati, M. Basavaraju, and M. Rao, "High performance and energy efficient AMD and BWAD pooling schemes characterized for CNN accelerators," in *Proc. 2023 IEEE 26th Int. Conf. on Digital System Design (DSD)*, 2023, pp. 470–477, doi:10.1109/DSD60849.2023.00071.

[6] Y.-L. Xie, X.-R. Lin, C.-Y. Lee, and C.-W. Lin, "Design and implementation of a 4096Hz-63MHz bandpass sigma-delta ADC for [title incomplete]," *IEEE Sensors J.*, vol. 24, no. 11, pp. 19834–19844, Jun. 2024, doi:10.1109/JSEN.2024.3393469.

[7] D. Ene´riz *et al.*, "Low-cost FPGA implementation of deep learning-based heart sound segmentation for real-time CVDs screening," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–16, 2024, Art. no. 2003616, doi:10.1109/TIM.2024.3392271.