FPGA-based Keyword Spotting Accelerator with Data Reuse Optimization

Yutong Zhu*

School of Integrated Circuits, Southeast University, Nanjing, 211100, China

*Corresponding author: 213232494@seu.edu.com

Abstract. Keyword Spotting (KWS), also known as wake-word detection, has become a basic component of voice systems such as smart speakers, mobile assistants and Internet of Things devices. It enables the device to continuously listen to specific commands while maintaining minimum energy consumption. With the development of neural networks, how to achieve efficient reasoning in a resource-limited environment is still a major challenge. Field Programmable Gate Array (FPGA) - a reconfigurable logic device that allows customization to be manufactured after production - provides a good balance between flexibility, power consumption and performance. This paper proposes an FPGA-based KWS accelerator optimization framework, which uses the data reuse mechanism to improve computing efficiency and reduce off-chip memory access. The system integrates the convolutional layer, activation layer and fully connected (FC) layer under a unified finite state control logic, and is modeled and simulated in Vivado 2017. The proposed row buffer-based reuse scheme minimizes redundant memory operations. Compared with the baseline design, the number of memory reads is reduced by 88.9% and the latency is reduced by 50%. The research results show that memory-centered architecture optimization can improve the reasoning performance of FPGA-based micro-machine learning (TinyML). This work provides a scalable and energy-saving framework for future intelligent edge systems.

Keywords: FPGA, Keyword Spotting, Data Reuse, TinyML, Hardware Optimization.

1. Introduction

KWS has become an indispensable part of modern voice interaction technology and the primary entrance to human-computer communication. Its application scope covers smartphones, smart speakers, autonomous robots and Internet of Things edge devices, etc. The effectiveness of the system depends on its ability to continuously detect predefined commands under a strict energy budget [1].

Although Central Processing Units (CPUs) and Graphics Processing Units (GPUs) provide computing flexibility, they are less energy efficient for KWS, which is always online. In contrast, FPGA has large-scale parallelism, configurability and high energy efficiency. FPGA can integrate digital signal processing (DSP) units and memory blocks customized for workloads, making them ideal for neural reasoning tasks [2].

However, keyword recognition accelerators based on FPGA usually face memory access bottlenecks, especially frequent access to chip DRAM will lead to a significant increase in energy consumption. In order to alleviate this problem, researchers have proposed a variety of optimization strategies, such as data reuse, on-chip buffering and memory blocking, so as to minimize access to external storage under limited on-chip storage resources [3-5]. For example, Shortcut Fusion and other studies show that the on-chip memory allocation of intermediate data can significantly reduce the number of DRAM accesses and improve the energy efficiency of FPGA accelerators [4]. This mechanism provides a reference optimization idea for the keyword recognition accelerator when processing voice or other edge computing tasks. These optimization technologies are not only applicable to CNN, but also the key to various computing-intensive acceleration tasks (such as sparse matrix multiplication) on FPGA [5]. These methods aim to reuse intermediate results locally to avoid redundant reading.

This paper introduces an FPGA optimized keyword recognition accelerator using a structured data reuse scheme. The system achieves lower memory access and lower computing latency while

maintaining accuracy. This architecture provides valuable reference for TinyML design for resource-limited environments.

2. Fundamental Principles and Background

2.1. Definition of KWS

KWS refers to the detection of predefined trigger words in a continuous voice stream. The classic method uses the hidden Markov model (HMM) and the Gaussian hybrid model (GMM) to identify the acoustic pattern [6]. With the progress of deep learning, the robustness of convolutional neural networks (CNN) and circular neural networks (RNN) to noise and speaker differences has been significantly improved [7].

Typical KWS processes include: using Mell frequency inverse spectral coefficient (MFCC) for feature extraction; convolutional layer for local feature detection; and full connection layer for classification output. The KWS model based on deep learning has a higher accuracy rate, but it needs to be carefully optimized for edge deployment.

2.2. FPGA Architecture and Operation Principle

FPGA is a reconfigurable semiconductor device, including logic blocks, lookup tables (LUTs), triggers and interconnection networks. It supports post-manufacturing hardware design through Verilog or VHDL. The reconfigurability of FPGA enables it to perform parallel computing efficiently, which is suitable for accelerating neural network workloads [8].

In the proposed FPGA-based architecture, the computing framework is mainly composed of logical units, which manage the control flow and sequence between functional modules. The computing core of the convolutional layer and the full connection layer – multiply accumulate (MAC) - is implemented with a special DSP slice to achieve high parallelism and arithmetic accuracy. On-chip random access memory (BRAM) efficiently processes local data cache and temporary feature storage, significantly reducing dependence on external memory. In addition, the reconfigurable interconnection network provides flexible routing between processing modules, thus optimizing data paths and supporting the reuse of intermediate results at different computing stages.

In the KWS accelerator, FPGA can be optimized for the parallelism of fine-grained calculations, so as to optimize throughput and energy efficiency at the same time [9].

2.3. Data Reuse and Memory Optimization

Memory bandwidth is still the main bottleneck of FPGA accelerators. The convolutional layer requires multiple overlapping data readings, resulting in an increase in energy consumption and delay. Data reuse alleviates this problem by storing intermediate calculation results in the local buffer. When the convolution window overlaps, the previously loaded data will be reused instead of being read from the DRAM again [10].

The row buffer (a structure based on the shift register) achieves this optimization by sliding data between registers, thus minimizing new data loading. Therefore, the accelerator can reuse up to 90% of the characteristic data while maintaining the integrity of the calculation.

2.4. Historical Development of FPGA-Based Accelerators

The early FPGA accelerator realized the basic matrix multiplication. With the introduction of Advanced Synthesis (HLS), the threshold of FPGA programming has been lowered [11]. Recent progress includes hybrid precision operation [12]. These developments make FPGA a feasible platform for deep learning reasoning in edge AI systems.

It is worth noting that the integration of data reuse and assembly line scheduling has brought about a leap in performance. For example, the research of Chen et al. shows that memory-centered

optimization can improve energy efficiency by up to 3 times compared with simple implementations [13].

In short, the KWS system relies on efficient convolutional computing, and FPGA-based hardware provides the flexibility and parallelism required to perform these calculations efficiently. The next section will introduce in detail the proposed FPGA-based accelerator architecture, its data reuse design, and the simulation results to verify its performance improvement.

3. System Design and Simulation

This paper proposes an FPGA-based KWS accelerator, which adopts a modular and hierarchical structure to run three main computing engines: convolution, activation and FC under central control, which is shown in Fig. 1. The system is developed in the Vivado 2017 emulator and implemented in Verilog language. Each module is parameterized for testing and optimization at different model scales and bit widths.

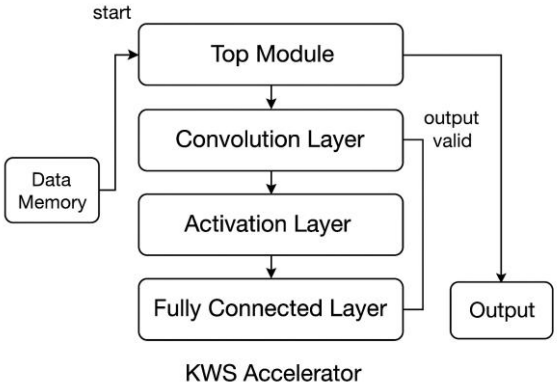

Fig. 1 FPGA-based KWS Accelerator System Architecture (Picture credit: Original)

3.1. System Architecture and Design Flow

The accelerator uses a pipeline-style architecture that enables convolution, activation, and FC layers to passage data consecutively. The Top Module provides the system interface for simulation and potential hardware deployment, coordinating and linking these submodules.

Convolution Layer (conv2d.sv). The convolution module is an operation that carries out 3 MAC operations in the input feature maps. In the non-cached variant, the nine feature elements are continuously accessed in memory every time a convolution window moves, causing unnecessary access to data. The optimized implementation adds a line buffer (in conv2dreuse.sv) of intermediate feature data stored on FPGA on-chip registers. The buffer moves instead of reading the same data again to overlap the windows, and the values within the buffer are reused instead of reading them out of the external memory, which brings massive improvements in the number of reads of the same data. The main innovation of this design is this optimization.

Activation Layer (relu.sv). The ReLU activation layer filters out negative values by integrating element-wise nonlinearity to the convolution results. It lies pipelined to keep synchronization with the convolution module, allowing steady data flow, despite staying programmatically straightforward.

Fully Connected Layer (fc.sv). The FC layer performs dense matrix-vector multiplication after processing the feature data, calculating the final classification logits for all potential keywords. To preserve numerical integrity, this layer is implemented to utilize customized arithmetic logic and nested loops, which are controlled solely by the system clock.

Control and Timing Logic. Employing signals like start, output_valid, and internal states, a finite-state machine (FSM) administers the computation flow between modules. The FSM supports handshake synchronization for multi-cycle execution and ensures correct operation sequencing. Timing signals coming from the testbench (tb) For guarantee end-to-end performance, sv) are utilized to assess the latency between start and valid pulses.

3.2. Dataflow and Memory Subsystem Design

Memory bandwidth underlies the energy and latency spectrum of a typical deep neural network. Local storage and reuse are key components of the dataflow strategy to addresses this. A 3×3 rotating window line buffer, which acts integrated as a range of registers that change data per clock cycle, is used for the envisioned data reuse mechanism. Only one new pixel is fetched per clock cycle, while the remaining eight values are reused.

This structure benefits power efficiency because on-chip accesses have substantially fewer energy-consuming than DRAM transactions, lowering the number of external reads by roughly 88.9%, as ascertained by simulation. The design reduces the level of LUTs and flip-flops required for buffer storage while preventing raising the complexity of the hardware.

The dataflow path may become summarized in the following points: Local Registers are utilized to loads and record the feature map and weight data. Convolution MAC actions are carried out over buffered data during the computation stage. Convolution results are digested by the ReLU and FC modules during the activation and classification stages. Output Stage: The output class inputs the final classification score, which signifies completion via output_valid.

3.3. Simulation and Verification Setup

The testbench (tb.sv) acts as a verification environment to generate clock and reset signals, initialize feature and weight data, and capture timing information. Two parallel instances of the accelerator - the baseline version (u_base) and the reuse optimized version (u_reuse) - are simulated at the same time under the same input. Performance indicators (including memory reads, execution delay and signal synchronization) will be automatically printed into the console. The simulation results are as follows:

Baseline: 9 memory reads per cycle, latency = 100,000 ns; Reuse: 1 memory read per cycle, latency = 50,000 ns; Read reduction = 88.9%, latency reduction = 50%

The waveform analysis further verified the correctness of the function: the starting signal starts the convolution at 60,000 ns. The corresponding output_valid pulse appears at 160,000 ns in the baseline model and 110,000 ns in the multiplex model. No signal overlap or competitive conditions were observed, confirming timing stability and data integrity.

Performance Summary is shown in Table 1.

Table 1. Performance Summary

	Baseline	Reuse	Reduction
reads	9	1	88.89%
latency	100,000ns	50,000ns	50.00%

4. Typical Application Scenarios and Analysis

4.1. TinyML FPGA Speech Recognition for IoT Devices

Kokkinis and Siozios analyzed a TinyML-based keyword search accelerator for low-power Internet of Things devices [13]. They proposed a fast resource estimation method for multilayer sensor (MLP) accelerators based on FPGA. This research focuses on accurately predicting the usage of on-chip resources and power consumption at the early stage of design, which is crucial for resource-limited edge devices. The study emphasizes that achieving efficient hardware utilization through accurate resource estimation is the key to ensuring that TinyML FPGA applications achieve low power consumption and real-time reasoning performance [13]. Their research results can provide a reference for FPGA-based accelerators that use memory-centered optimization methods for constant online keyword search.

4.2. Edge-AI Voice Detection in Smart Homes

Bae and his colleagues have developed a low-power FPGA engine specifically for keyword recognition in smart home environments [14]. They used the deeply separable binary/ternary neural network to implement the wake-up word (WUW) detector and multi-command classifier on the FPGA board. Their hardware system includes on-chip memory buffering to maximize data reusability, thus reducing external DRAM access and achieving a real-time delay of less than 100 milliseconds. This application scenario aims to meet the needs of voice interaction that is always on in household appliances, and takes into account low power consumption and high response speed. This work provides practical insights for the FPGA-based hardware-level, memory-optimized keyword recognition accelerator, which can balance accuracy, latency and power consumption.

4.3. Wearable and IoT Voice Activation for Edge Devices

Smith et al. proposed an FPGA-based configurable keyword recognition accelerator designed to serve resource-limited edge/Internet of Things devices, such as wearable devices or smart sensors [15]. They use a small, time-efficient neural network (TENet), which is mapped to FPGA and uses a distributed on-chip buffer to manage memory access and reduce latency. The target scenario is the voice activation that is always on in devices with tight power consumption and memory budgets, in which the voice command triggers the downstream function. Their analysis shows that by blocking the data and activating it in the buffer on the chip, the accelerator can reduce the bandwidth of off-chip memory, thus improve energy efficiency while maintain the accuracy of the identifier at about 95%. These findings provide guidance for the practical deployment of FPGA-based keyword recognition accelerators in the Internet of Things ecosystem.

4.4. FPGA Inference Benchmarking

Reddi and his colleagues used the MLPerf benchmark test to evaluate the reasoning performance of the FPGA-based small neural network model in keyword recognition [9]. Their research emphasizes the importance of memory and computing optimization, and shows that energy efficiency is improved by 3% to 5% compared with CPU implementation. These research results show that through the FPGA keyword search accelerator, effective on-chip memory scheduling and data reuse strategies can be used to reduce latency and power consumption. This research provides indirect but crucial suggestions for the construction of FPGA accelerators, which can achieve real-time, low-power keyword recognition by understanding the performance bottlenecks in memory access and computing.

5. Conclusion

This study proposes a keyword search accelerator based on FPGA, which is optimized by the data reuse mechanism. Convolution layer was created to reduced redundant data reads and enhance throughput without relinquishing accuracy by featuring a reusable line buffer. The planned optimization's effectiveness appears proven by simulation results that demonstrated an 88.9% reduction in memory reads and a 50% reduction in latency.

For deeper models and additional convolutional layers, scalability is rendered possible by the modular architecture. Future extensions may look at multi-channel convolution, normalized arithmetic, and adaptive precision techniques despite the recent work concentrating on three-and three-channel operations. Furthermore, power efficiency and flexibility may be enhanced by including dynamic voltage scaling and partial reconfiguration.

For real-time testing, the design may come shipped to Xilinx Zynq or Intel Cyclone platforms in future research. Its application in edge AI environments would stand augmented by including multilingual KWS models and adaptive noise filtering. The planned framework supplies a solid foundation for the introduction of TinyML accelerators for the upcoming FPGA that work memory-efficient and low-latency-optimized.

References

[1] Zhang S Y, Chen Y, Yang H. Tiny Speech Recognition: A Microcontroller Benchmark. Proceedings of MLSys. 2021.

[2] Chen G, Parada C, Heigold G. Small-footprint keyword spotting using deep neural networks. 2014 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2014: 4087-4091.

[3] Mittal S. A survey of FPGA-based accelerators for convolutional neural networks. Neural computing and applications, 2020, 32(4): 1109-1139.

[4] Putra R V W, Hanif M A, Shafique M. ROMANet: Fine-grained reuse-driven off-chip memory access management and data organization for deep neural network accelerators. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2021, 29(4): 702-715.

[5] Liu Y, Chen R, Li S, et al. FPGA-based sparse matrix multiplication accelerators: From state-of-the-art to future opportunities. ACM Transactions on Reconfigurable Technology and Systems, 2024, 17(4): 1-37.

[6] Povey D, Ghoshal A, Boulianne G, et al. The Kaldi speech recognition toolkit. IEEE 2011 workshop on automatic speech recognition and understanding. 2011, 1: 5.1.

[7] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436-444.

[8] Xilinx Inc. Vivado Design Suite User Guide: High-Level Synthesis. UG902, 2021.

[9] Reddi V J, Cheng C, Kanter D, et al. Mlperf inference benchmark. 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2020: 446-459.

[10] Wan Y, Chen J, Yang X, et al. DSA-CNN: an fpga-integrated deformable systolic array for convolutional neural network acceleration. Applied Intelligence, 2025, 55(1): 65.

[11] Du C, Yamaguchi Y. High-level synthesis design for stencil computations on FPGA with high bandwidth memory. Electronics, 2020, 9(8): 1275.

[12] Wang J, He Z, Zhao H, et al. Low-Bit Mixed-Precision Quantization and Acceleration of CNN for FPGA Deployment. IEEE Transactions on Emerging Topics in Computational Intelligence, 2024.

[13] Kokkinis A, Siozios K. Fast Resource Estimation of FPGA-Based MLP Accelerators for TinyML Applications. Electronics, 2025, 14(2): 247.

[14] Bae S, Kim H, Lee S, et al. FPGA implementation of keyword spotting system using depthwise separable binarized and ternarized neural networks. Sensors, 2023, 23(12): 5701.

[15] He K, Chen D, Su T. A configurable accelerator for keyword spotting based on small-footprint temporal efficient neural network. Electronics, 2022, 11(16): 2571.