UNIVERSITY OF CALIFORNIA SAN DIEGO

CNNs on Multi-channels 2-D Systolic Array with Versatile Pruning

TEAM: STARLIGHT

Author	*Student ID*
Liangyuan Wang	A69041633
Zhuoting Yang	A69041696
Jingbin Lin	A69041649
Zihao Yang	A123
Haotian Ye	A456

Contents

Background **3**

1 CNNs Model **3**

2 2-D Systolic Array **3**

Implementation **3**

1 CNNs Model Training **3**

2 Weight Stationary 2-D Systolic Array **4**
2.1 Vanilla Version (Part 1) 4
2.2 2-bit/4-bit Reconfigurable SIMD Systolic Array (Part 2) 4
2.3 Multi-cores Tiling Architecture (Alpha 4) 5

3 Output Stationary 2-D Systolic Array **5**
3.1 WS/OS Reconfigurable Architecture (Part 3) 5
3.2 Output-Stationary Skip Optimization (Alpha 3) 5

Conclusions **6**

Reference **6**

Background

1 CNNs Model

Convolutional Neural Networks (CNNs) represent a fundamental class of deep learning architectures widely used for image classification tasks. These architectures leverage spatial locality and parameter sharing, allowing models to efficiently extract hierarchical features from input images. A typical CNN consists of convolutional layers for feature extraction, nonlinear activation functions, and pooling layers for spatial down-sampling, followed by fully connected layers for classification. Because convolution dominates the computational workload-often accounting more than 90% of multiply-accumulate(MAC) operations-CNNs are of particular interest in hardware acceleration and VLSI implementation.

In this course (ECE284: VLSI Implementation for Machine Learning), we studied several widely adopted CNNs Models such as VGGNet and ResNet and trained them on the CIFAR-10 dataset. In addition to studying classical CNNs architectures, the course also introduced model compression techniques, including quantization and pruning, which are essential for deploying deep networks on hardware accelerators.

2 2-D Systolic Array

The 2-D systolic array is a widely adopted architecture for accelerating dense linear-algebra, particularly the multiply-accumulate(MAC) patterns that dominate convolutional neural networks. In this course, we studied how a convolutional layer can be transformed into a a series of matrix operations that naturally map onto a 2-D grid of processing elements(PEs). Each PE perform local computations while passing partial sums, activations, or weights to neighboring cells in pipelined fashion. This spatial data movement enable high throughput, predictable timing, and efficient reuse of data as it propagates across the array.

The course introduced two major hardware dataflow strategies for mapping CNN workloads onto systolic arrays:

- **Weight-stationary (WS):** In this dataflow, the weights remain fixed within the PEs while inputs stream across the array. This maximizes weight reuse because each stored filter value participates in many MAC operations without needing to be reloaded. WS is particularly attractive when weight bandwidth is the bottleneck, and it aligns well with compact, deeply pipelined grids where filters are reused across multiple activation tiles.
- **Output-stationary (OS):** Here, each PE accumulates a specific output activation (or partial sum) locally until the final result is completed. Inputs and weights flow through the array, but partial sums are retained within the PE instead of being passed downstream. OS is effective for maximizing accumulation locality and minimizing write traffic to external memory, especially for layers with large output feature maps.

Implementation

1 CNNs Model Training

Item	Value
Dataset	CIFAR-10
4-bit Activation Model Accuracy	91.2%
2-bit Activation Model Accuracy	90.0%
4-bit Activation Quantization Error	3.2444×10^{-7}
2-bit Activation Quantization Error	1.9340×10^{-6}

Table 1: Quantization-aware training results

We train models with 4-bit activations and 4-bit weights, as well as 2-bit activations with 4-bit weights, to match the supported precision modes of our hardware accelerator. To improve robustness under aggressive quantization, we adopt an optimized training strategy that combines Adam optimization, label smoothing, and a cosine learning rate scheduler (Alpha 1). Compared to a baseline SGD with momentum, this approach improves final accuracy, helping us overcome a training bottleneck.

As a result, the 4-bit activation model achieves an accuracy of 91.2%, while the 2-bit activation model reaches 90%. Quantization errors remain extremely small (on the order of 10^{-6}), indicating that the trained models are well-aligned with the numerical behavior of the target hardware.

Method	4-bit Acc (%)	2-bit Acc (%)
Baseline (SGD + Momentum)	89.0	89.0
Adam + Label Smoothing + Cosine Scheduler	91.2	90.0

Table 2: Comparison of training strategies under quantization.

In addition, we explore mixed-granularity pruning as part of the training pipeline (Alpha 2). Our proposed C2F pruning method combines structured pruning with fine-grained unstructured pruning for higher sparsity. This approach preserves model accuracy while maintaining compatibility with systolic array execution, outperforming purely structured pruning in both accuracy and sparsity.

Method	VGG16_Quant (4-bit) Acc (%)	ResNet20_Quant (4-bit) Acc (%)
Unstructured Pruning	90.31	89.94
Structured Pruning	83.48	78.53
Our Method (Coarse-to-Fine)	90.22	87.74

Table 3: Accuracy comparison of pruning methods on quantized models.

Method	VGG16_Quant (4-bit) Sparsity (%)	ResNet20_Quant (4-bit) Sparsity (%)
Unstructured Pruning	87.90	69.82
Structured Pruning	96.24	61.92
Our Method (Coarse-to-Fine)	87.92	81.64

Table 4: Sparsity comparison of pruning methods on quantized models.

2 Weight Stationary 2-D Systolic Array

2.1 Vanilla Version (Part 1)

The vanilla accelerator is implemented as an 8×8 weight-stationary systolic array using a modular RTL hierarchy. Processing elements are defined in `mac.v` and composed into rows and arrays through `mac_row.v` and `mac_array.v`. Input activations and weights are buffered by FIFO-based modules, while partial sums are collected via `ofifo.v` and accumulated using `sfu.v`. The integrated design in `corelet.v` is functionally verified using a controller-based testbench with file-driven stimuli. The design is synthesized on a Cyclone IV GX FPGA, achieving a maximum frequency of 128.72 MHz with approximately 11% logic utilization.

2.2 2-bit/4-bit Reconfigurable SIMD Systolic Array (Part 2)

We extend the vanilla weight-stationary systolic array with a lane-reconfigurable SIMD PE to support both 2-bit and 4-bit activation execution using the same datapath.

In 2-bit mode, each PE processes two activation lanes in parallel using two 2-bit×4-bit multipliers and accumulates partial sums on two PSUM lanes. In 4-bit mode, the activation is decomposed into two 2-bit segments, and the resulting partial sums are bit-shifted and merged to form a 4-bit×4-bit result, with a single 4-bit weight broadcast to both weight registers.

A lightweight control logic switches between the two modes without introducing additional multipliers or lanes. To support 16 output channels in 2-bit mode, channel tiling is implemented in the testbench controller, which schedules multiple passes over the 8×8 array and aggregates partial sums without modifying the hardware datapath.

2.3 Multi-cores Tiling Architecture (Alpha 4)

We implements a multi-core tiling architecture under the weight-stationary dataflow by replicating the vanilla 8×8 systolic array at the tile level. Each core reuses the same PE, row, and array hierarchy defined, without modifying the internal datapath.

Input activations are partitioned into tiles and distributed to multiple WS cores by a lightweight controller, while weights remain stationary within each core. Partial sums generated by each core are collected and combined after execution. This approach exploits parallelism across replicated WS arrays to improve throughput and scalability.

Metric	Value
Target FPGA	Cyclone IV GX
Ops	128
Frequency	128.72 MHz
Dynamic Power	340.72 mW
Throughput	16.5 GOPs/s
Energy Efficiency	48.4 GOPs/W
Logic Elements	17,112 / 149,760 (11%)

Table 5: FPGA implementation results of the 8×8 systolic array.

3 Output Stationary 2-D Systolic Array

3.1 WS/OS Reconfigurable Architecture (Part 3)

We extend the vanilla systolic array to support output-stationary execution in addition to the weight stationary mode. The reconfiguration is implemented at the tile and array levels by reusing the same processing elements and datapaths, with lightweight multiplexing controlled by a single dataflow select signal.

In OS mode, partial sums are locally accumulated within each processing element, while activations and weights stream through the array. An additional input FIFO is introduced to support the required weight streaming behavior. Functional verification is performed using the first convolutional layer with partial channel and spatial mapping, confirming correct WS/OS switching and OS execution.

3.2 Output-Stationary Skip Optimization (Alpha 3)

Alpha 3 is built upon WS/OS reconfigurable systolic array architecture, where we select the output-stationary execution mode as the baseline. Leveraging the local accumulation property of OS, partial sums remain within each processing element, enabling accurate detection of idle and skip cycles without affecting computational correctness.

On top of this OS mode, we introduce fine-grained clock gating for output-stationary gate skipping across the PE, row, and array hierarchy. Local activity signals are used to selectively disable the MAC datapath, FIFOs,

and related registers when no valid accumulation is required, effectively reducing unnecessary switching activity while preserving the correctness and flexibility of the original WS/OS reconfigurable design.

Conclusions

On the software side, we apply quantization-aware training to support both 4-bit and 2-bit execution and propose a coarse-to-fine pruning strategy that balances sparsity and accuracy while remaining compatible with systolic-array-based computation. These techniques enable efficient low-precision inference and provide software models that align well with the target hardware constraints.

On the hardware side, we implement an 8×8 weight-stationary systolic array as the vanilla design (Part 1) and extend it with a lane-reconfigurable 2-bit/4-bit SIMD architecture (Part 2), enabling flexible precision support with minimal hardware overhead. We further realize a WS/OS reconfigurable architecture (Part 3) and explore output-stationary skip optimization (Alpha 3) and multi-core tiling (Alpha 4), respectively highlighting tradeoffs between energy efficiency and scalability.

Overall, this project demonstrates that careful hardware–software co-design and dataflow-aware optimization can effectively balance accuracy, efficiency, and extensibility for CNN acceleration on resource-constrained FPGA platforms.

Reference

[1] C. Ogbogu, *et al.*, "Energy-Efficient ReRAM-Based ML Training via Mixed Pruning and Reconfigurable ADC," in *Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, Vienna, Austria, 2023, pp. 1–6.