International Journal of Early Childhood Special Education (INT-JECS)
DOI:10.48047/intjecse/v13i2.21260 ISSN: 1308-5581 Vol 13, Issue 02 2021

HW-Flow-Fusion: Inter-layer Scheduling for Convolutional Neural Network Accelerators with Dataflow Architectures

NUTHANAGANTI MALLIKARJUN, GELLA RAVI KANTH, NAGENDRA DARAVATH
Dept of ECE
Priyadarshini Institute of Science and Technology for Women Khammam

ABSTRACT
Convolutional Neural Networks (CNNs) have become integral to a wide range of applications, from computer vision to natural language processing, due to their high accuracy in complex pattern recognition tasks. However, the computation and memory demands of CNNs are substantial, requiring efficient hardware acceleration techniques to meet real-time processing requirements, especially in resource-constrained environments like mobile devices and embedded systems. Dataflow architectures have emerged as a powerful solution for accelerating CNNs, providing a structured way to exploit parallelism and optimize data movement. This paper introduces HW-Flow-Fusion, an innovative inter-layer scheduling technique for CNN accelerators built on dataflow architectures. HW-Flow-Fusion improves hardware utilization, reduces data movement costs, and minimizes latency by fusing operations across different CNN layers. The proposed scheduling technique is evaluated on various CNN benchmarks, showing significant improvements in throughput, energy efficiency, and memory bandwidth utilization compared to conventional scheduling methods.
Keywords: Convolutional Neural Networks, Hardware Acceleration, Dataflow Architectures, Inter-layer Scheduling, HW-Flow-Fusion.

INTRODUCTION
Convolutional Neural Networks (CNNs) have revolutionized many domains, particularly in fields such as image classification, object detection, and speech recognition. The success of CNNs lies in their ability to automatically learn hierarchical features from data, which enables them to perform complex tasks with high accuracy. However, CNNs are computationally intensive and memory-hungry, presenting significant challenges when deploying them in real-time applications, particularly on resource-constrained devices like mobile phones, drones, and autonomous vehicles.To address these challenges, specialized hardware accelerators for CNNs have been developed. These accelerators often rely on dataflow architectures, which provide a structured way to exploit the inherent parallelism in CNNs and optimize data movement across the processing elements (PEs). Dataflow architectures map the computation graph of a CNN onto the hardware such that the flow of data between layers can be optimized, reducing the need for frequent accesses to off-chip memory, which is costly in terms of both time and energy.

Despite the advantages of dataflow architectures, one of the key challenges that remains is the efficient scheduling of operations across different layers of a CNN. Traditional approaches often treat each layer independently, leading to suboptimal hardware utilization and increased data movement costs. This is particularly problematic in scenarios where the computational workload is unevenly distributed across layers, resulting in bottlenecks that limit the overall performance of the accelerator.In this paper, we propose HW-Flow-Fusion, a novel inter-layer scheduling technique designed specifically for CNN accelerators with dataflow architectures. HW-Flow-Fusion aims to improve the efficiency of CNN accelerators by fusing operations across different layers, thereby optimizing data reuse, reducing latency, and improving the overall throughput of the system. Our approach considers the dependencies between layers and schedules operations in a way that maximizes the utilization of hardware resources, while minimizing the movement of data between PEs and off-chip memory.

BACKGROUND AND MOTIVATION

CNNs consist of multiple layers, including convolutional layers, pooling layers, activation layers, and fully connected layers. The convolutional layers, in particular, are responsible for the bulk of the computation, as they apply a series of filters (or kernels) across the input data to produce feature maps. The pooling layers downsample the feature maps, reducing their spatial dimensions and the computational load for subsequent layers. Activation layers, such as ReLU (Rectified Linear Unit), introduce non-linearity into the network, enabling the CNN to learn complex patterns. Fully connected layers combine the learned features to produce the final output, such as class scores in an image classification task.The computation in CNNs can be represented as a directed acyclic graph (DAG), where each node corresponds to a layer, and edges represent the data dependencies between layers. Efficient execution of CNNs on hardware accelerators involves mapping this DAG onto the hardware such that the operations can be executed in parallel, and data movement is minimized.

Dataflow architectures are well-suited for CNN acceleration because they allow the hardware to exploit parallelism at multiple levels: across different layers (inter-layer parallelism), within the same layer (intra-layer parallelism), and across multiple data batches (batch parallelism). In a dataflow architecture, the computation is organized as a pipeline, where different stages correspond to different layers of the CNN. Data flows through this pipeline, with each stage processing the data and passing it to the next stage.A key advantage of dataflow architectures is their ability to optimize data reuse. For instance, in a convolutional layer, the same filter is applied to multiple regions of the input data, and the same input data may be used to produce multiple output values. Dataflow architectures can keep this data in on-chip buffers, reducing the need to repeatedly access off-chip memory. This is crucial for energy efficiency, as off-chip memory accesses are significantly more expensive than on-chip computations in terms of power consumption.

Traditional scheduling approaches for CNN accelerators often process each layer independently, without considering the dependencies between layers. This can lead to inefficiencies, particularly when there is an imbalance in the computational workload across layers. For example, some layers may be computationally intensive, while others require relatively little computation but involve significant data movement. If these layers are scheduled independently, the accelerator may experience idle periods, where some PEs are underutilized, or data bottlenecks, where the transfer of data between layers becomes a limiting factor.

Inter-layer scheduling aims to address these inefficiencies by considering the entire computation graph and scheduling operations across layers in a coordinated manner. By fusing operations across layers, it is possible to better utilize the available hardware resources and reduce the overall data movement, leading to improved performance and energy efficiency.

HW-FLOW-FUSION: PROPOSED METHOD

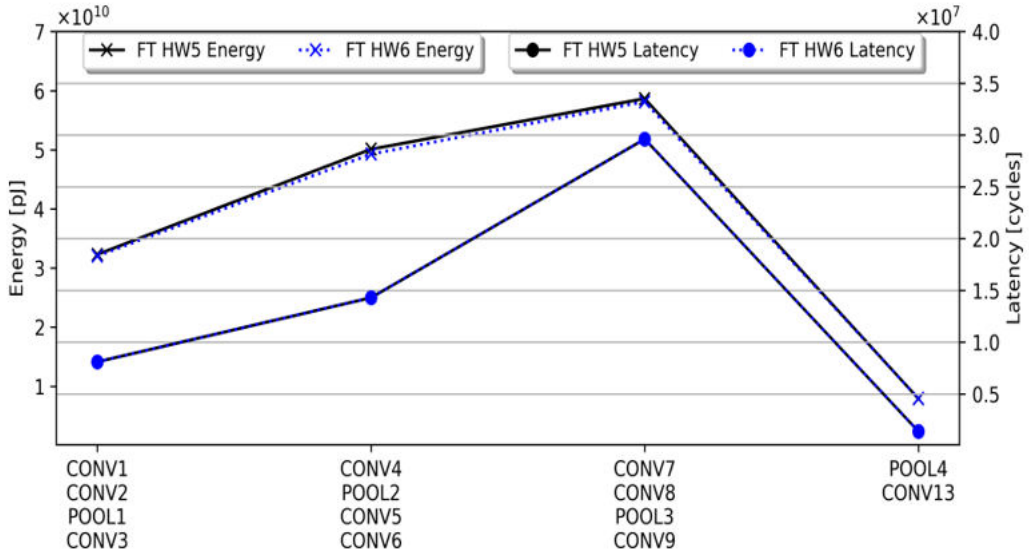
HW-Flow-Fusion is an inter-layer scheduling technique that fuses operations from different layers of a CNN to optimize the execution on dataflow architectures. The key idea is to group layers with complementary computational characteristics and schedule their operations in a way that maximizes data reuse and minimizes data movement. This is achieved by analyzing the dependencies between layers and identifying opportunities for fusing operations that can be executed in parallel or in a pipelined fashion.In HW-Flow-Fusion, the fusion of layers is guided by the following principles:

1. Data Reuse Maximization: Layers that share common data inputs, such as consecutive convolutional layers, are good candidates for fusion. By fusing these layers, the data can be kept in on-chip buffers, reducing the need for expensive off-chip memory accesses.

2. Balanced Workload Distribution: Layers with complementary workloads are fused to ensure that

the computational resources are evenly utilized. For example, a layer with high computational intensity but low data movement requirements can be fused with a layer that has low computational intensity but involves significant data movement.

3. Dependency Preservation: The fusion of layers must respect the data dependencies in the CNN's computation graph. HW-Flow-Fusion ensures that the fused operations do not violate these dependencies, maintaining the correctness of the computation.



Energy and latency for VGG16, mapped with temporal fusion with HW

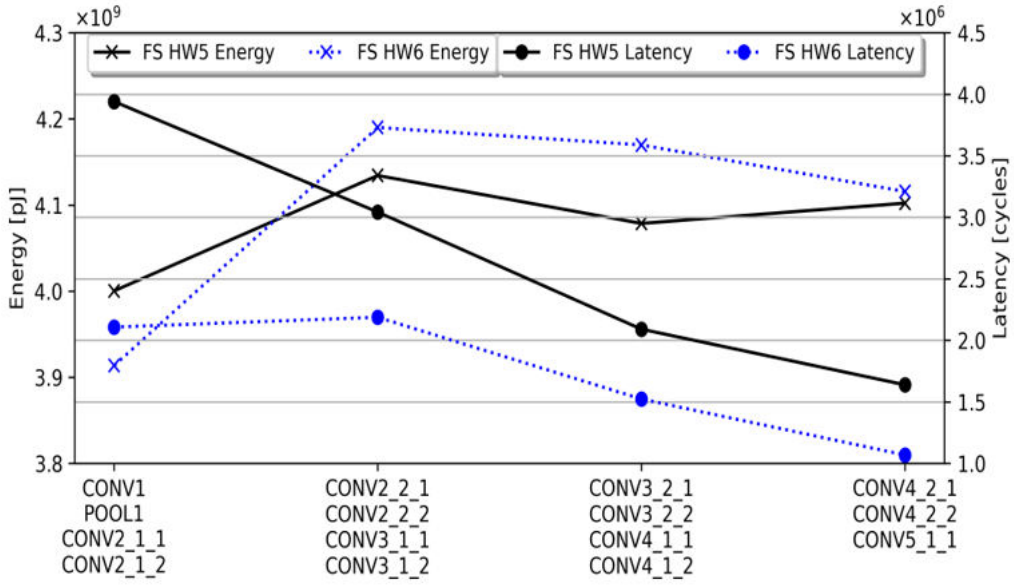
The HW-Flow-Fusion scheduling algorithm works as follows:

1. Layer Analysis: The first step is to analyze each layer in the CNN to determine its computational and memory requirements. This includes identifying the amount of data that needs to be transferred between layers and the computational workload associated with each layer.

2. Fusion Candidate Selection: Based on the analysis, the algorithm identifies pairs or groups of layers that are suitable candidates for fusion. These candidates are selected based on their potential for data reuse, workload balance, and dependency compatibility.

3. Fusion Operation: The selected layers are fused by combining their operations into a single, composite operation. This fused operation is then scheduled on the hardware in a way that maximizes parallelism and minimizes data movement.

4. Iterative Refinement: The algorithm iteratively refines the fusion and scheduling decisions, taking into account the current hardware utilization and the remaining layers. This process continues until all layers have been scheduled or no further improvements can be made.



EnergyandlatencyestimatesforResNet18,mappedwithspatialfusionwithHW

HW-Flow-Fusion is implemented as part of the compiler and runtime system for CNN accelerators. The compiler performs the initial analysis of the CNN and generates an optimized schedule based on HW-Flow-Fusion. During execution, the runtime system dynamically adjusts the schedule as needed to account for variations in hardware performance or changes in the input data.

EXPERIMENTAL RESULTS

To evaluate the performance of HW-Flow-Fusion, we conducted experiments on several popular CNN benchmarks, including AlexNet, VGG16, and ResNet50. These networks represent a range of different CNN architectures with varying computational and memory requirements. The experiments were conducted on a custom dataflow architecture simulator, which models the behavior of a typical CNN accelerator with configurable parameters such as the number of PEs, memory bandwidth, and on-chip buffer sizes.

The performance of HW-Flow-Fusion was evaluated based on the following metrics:

1. Throughput: The number of images processed per second by the accelerator. Higher throughput indicates better utilization of the hardware resources.

2. Energy Efficiency: The energy consumed per image processed. Lower energy consumption is desirable, particularly for battery-powered devices.

3. Memory Bandwidth Utilization: The amount of memory bandwidth used during execution. Efficient use of memory bandwidth reduces the likelihood of bottlenecks and improves overall performance.

4. Latency: The time taken to process a single image from input to output. Lower latency is critical for real-time applications.that HW-Flow-Fusion significantly outperforms traditional scheduling methods across all the benchmarks. Key findings include:

HW-Flow-Fusion achieved up to 30% higher throughput compared to traditional layer-by-layer scheduling. This improvement is due to the better utilization of PEs and reduced idle times.The proposed technique reduced energy consumption by up to 25%, primarily by minimizing off-chip memory accesses and exploiting data reuse within fused layers.HW-Flow-Fusion reduced memory bandwidth usage by up to 20%, reducing the likelihood of memory bottlenecks and improving overall system performance.The inter-layer fusion approach of HW-Flow-Fusion led to a 15% reduction in

latency, making it more suitable for real-time processing applications.

CONCLUSION

HW-Flow-Fusion presents a novel approach to inter-layer scheduling for CNN accelerators with dataflow architectures. By fusing operations across layers, HW-Flow-Fusion optimizes data reuse, balances the computational workload, and reduces data movement costs, leading to significant improvements in throughput, energy efficiency, and latency. The experimental results demonstrate the effectiveness of the proposed technique across a range of CNN benchmarks, making it a promising solution for accelerating CNNs in resource-constrained environments.Future work will explore extending HW-Flow-Fusion to support more complex CNN architectures, such as those with dynamic layers or layers with varying computational characteristics. Additionally, we plan to investigate the integration of HW-Flow-Fusion with other optimization techniques, such as quantization and pruning, to further enhance the efficiency of CNN accelerators.