2023 IEEE 41st International Conference on Computer Design (ICCD) | 979-8-3503-4291-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCD58817.2023.00033

Polyform: A Versatile Architecture for Multi-DNN Execution via Spatial and Temporal Acceleration

Lingxiang Yin[1], Amir Ghazizadeh[1], Shilin Tian[1], Ahmed Louri[2], Hao Zheng[1]
[1]*Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, USA*
[2]*Department of Electrical and Computer Engineering, George Washington University, Washington, D.C., USA*
[1]{lingxiang.yin, amir.g, shilin.tian, hao.zheng}@ucf.edu, [2]louri@gwu.edu

***Abstract*—Contemporary applications and cloud workloads often comprise multiple Deep Neural Network (Multi-DNN) models. These models exhibit significant variations in computation, memory, and communication characteristics. For such heterogeneous workloads, a static and rigid hardware accelerator can no longer provide efficient and high-performance execution. To this end, we propose a versatile accelerator, called *Polyform*, to support the concurrent execution of different DNN models with the goal of improving energy and performance efficiency. Specifically, *Polyform* features two unique designs from both hardware and scheduling standpoints. On the hardware level, we have designed a flexible interconnection network that facilitates the formation of multiple sub-accelerators. Our design allows for spatial resource partitioning, including bandwidth and computation, while also providing effective communication support for various parallelism choices. On the scheduling level, *Polyform* employs a novel two-stage Genetic Algorithm (GA) to explore and identify the optimal configurations such as task orders, partition size, dataflow styles (e.g., weight or output stationary), and bandwidth. Our simulation shows that *Polyform* achieves remarkable results compared to prior work, including up to 77.8% energy reduction and a 2.79× improvement in throughput as compared to prior work [1]–[3].**

I. Introduction

The success of Deep Neural Networks (DNNs) is tightly coupled to the increased computing power enabled by the underlying hardware. Specialized hardware accelerators [4]–[7] have been designed for specific DNN models. However, the diverse nature of DNN models, serving various applications, has resulted in varying computation and communication characteristics. Static accelerator designs can no longer efficiently meet the demands of executing multiple DNNs.

This challenge involves a spatio-temporal perspective in executing multi-DNNs. From the temporal point of view, efficient management of shared resources in a monolithic accelerator across different time steps is crucial for optimal performance and resource utilization. The simultaneous execution of multiple DNNs could lead to resource contention in shared resources like memory bandwidth and computational units, resulting in performance degradation or resource underutilization [8]. Moreover, effective task scheduling, including order of execution, dependency management, and workload balancing, is also important.

From the spatial standpoint, an accelerator needs to be partitioned spatially, simultaneously accommodating multi-DNN with different computation and communication requirements. For example, Planaria [2] proposed a reconfigurable architecture that can concurrently execute various DNN tasks, where compute/memory resources are dynamically allocated. MAGMA [3] proposed a design running multi-DNN with a couple of dataflow choices. However, none of the current accelerators can provide all the desired spatial and temporal functionalities in one unified architecture as shown in Table I.

Table I: A comparison of the state-of-the-art designs for Multi-DNN Execution.

Design	Spatial Partitioning	Temporal Scheduling	Partition Granularity	Flexible Dataflow	Flexible Bandwidth
PREMA [1]	✗	✓	Low	✗	✗
Planaria [2]	✓	✓	High	✗	✗
MAGMA [3]	◐	✓	Medium	◐	✓
Polyform	✓	✓	**High**	✓	✓

✓ Full support ✗ No support ◐ Limited support

In this paper, we introduce *Polyform*, a versatile and flexible accelerator capable of executing multi-DNN on one accelerator. *Polyform* consists of optimized designs at both the hardware and scheduling levels. On the hardware level, *Polyform* supports a flexible interconnection network that can facilitate the formation of multiple sub-accelerators. Our design allows for spatial resource partitioning, including bandwidth and computation, while also providing effective communication support for various dataflows. *Polyform* facilitates various communications, specifically aligning the traffic pattern requirements (e.g., broadcast and unicast communications) and the dataflow in each sub-accelerator. On the scheduling level, *Polyform* employs a novel two-stage Genetic Algorithm (GA). In the first stage, the algorithm prioritizes finding an optimal task allocation policy and determining the suitable number of sub-accelerators. Once the optimal partitioning scheme for each workload is identified, the second stage configures the architectural settings within those sub-accelerators, including processing elements, bandwidth, and different dataflows (such as weight or output stationary). Through our simulation, we observe that *Polyform* achieves up to 77.8% energy reduction and 2.79× throughput improvement as compared to prior work [1]–[3].

II. Polyform Architecture Design

Our proposed accelerator design consists of two unique designs: (1) a versatile and flexible DNN architecture with the ability to be spatially partitioned for multiple DNN tasks with any desired size and dataflow, and (2) a holistic scheduling framework that is capable of managing task allocation, resource allocation, and dataflow selection at the same time.

2576-6996/23/$31.00 ©2023 IEEE
DOI 10.1109/ICCD58817.2023.00033

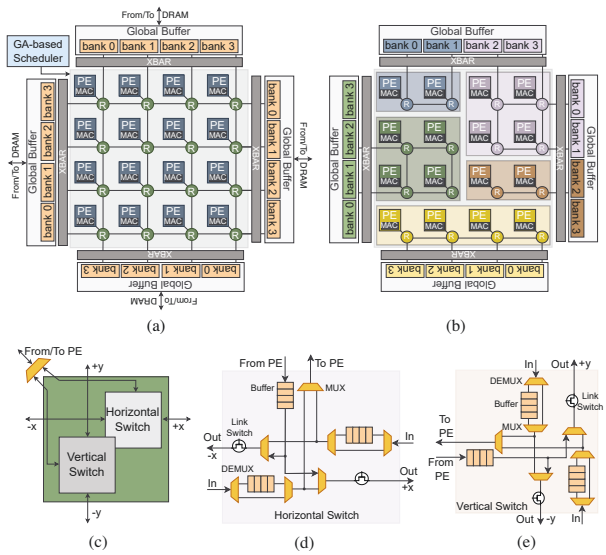


Figure 1: (a) The proposed overall architecture, (b) an example of accelerator partitioning with five running DNN tasks, and (c) the flexible router. (d) and (e) are detailed designs of a horizontal and a vertical switch inside the router, respectively.

A. Proposed Accelerator Micro-architecture

In this work, we use a 4×4 design to illustrate the functionalities of the proposed accelerator, as shown in Figure 1(a). The accelerator is equipped with a genetic algorithm (GA)-based scheduler, distributed global buffers (banked buffers), an array of PEs, and a flexible interconnect. Consequently, it can be dynamically partitioned into multiple individually running sub-accelerators, and each sub-accelerator is sized and configured to support its running DNN tasks, as shown in Figure 1(b). All the PEs are interconnected by a flexible NoC, and PEs and banked global buffers are connected by a crossbar (XBAR) to provide all-to-all connections. Each PE consists of 16 MAC units for matrix multiplication, buffers for weight filters, input activations, and output activations.

B. Flexible Interconnection Network Design

Conventional routers with complex virtual channels and crossbars are designed to address the unpredictable CPU cache and memory traffic. However, multi-DNN execution offers optimization opportunities due to its predictable, rhythmic traffic, in which data propagates in a row-wise or column-wise manner. Given this, complex crossbars and virtual channels are no longer needed.

In this work, we propose a flexible router and link design to leverage the mentioned opportunities. Specifically, we design low-cost vertical and horizontal switches supporting column- and row-wise communications as shown in Figure 1(c). Both vertical and horizontal switches can support either bus-based or packet-based communications. Figure 1(d) and Figure 1(e) show the detailed design for each switch. For both switches, the input ports can be used to continue the data propagation through a wire like a bus or a buffer. Multiple transistors

Table II: Supported Dataflows in Ployform and Their Communication Patterns (Router Modes).

Parallelism Dimension	Weight Filter		Input Activation		Output Activation		Temporal Stationary
	Multi	Uni	Multi	Uni	Multi	Uni	
C		✓		✓		✓	WS/IS/OS
K		✓	✓			✓	WS/IS/OS
(R, S)		✓	✓			✓	WS/IS/OS
(X', Y')	✓			✓		✓	IS/OS
(C, R, S)		✓	✓			✓	WS/IS/OS
(K, X', Y')	✓			✓		✓	IS/OS
(C, R, S, K, X', Y')		✓		✓		✓	N/A

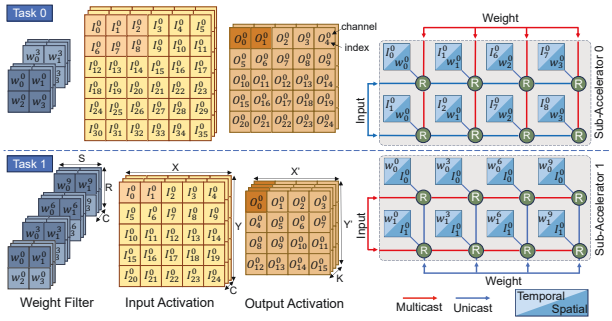


Figure 2: An example of supporting different dataflows for different tasks in different sub-accelerators.

are implemented, acting as a link switch to turn off the data transmission when the router is placed at the edge of the partitioned network to avoid signal interference. Alternatively, transistors can be replaced by tri-state link repeaters [9] but with a much higher cost.

C. Flexible Partitioning and Dataflow Support

The flexible router and links enable the sub-accelerator partitioning by turning on the link switch. The signal propagation is separated among partitions. Each partitioned accelerator can be configured according to its desired dataflow, including spatial parallelism (C, R, S, K, X, Y, X', Y') and temporal reuse strategy (weight stationary (WS), input stationary (IS), and output stationary (OS)). Table II summarizes the characterizations and router mode for different dataflows. Since the partial sums are accumulated hop by hop, it requires unicast support at either horizontal or vertical direction. As such, at least one of the weight filter and input activation matrices should be distributed in a unicast manner.

To illustrate *Polyform*'s mapping strategy, we present two convolutional layer examples as $Task\ 0$ and $Task\ 1$ in Figure 2. $Task\ 0$ and $Task\ 1$ are mapped to partitioned sub-accelerators 0 and 1, respectively. For $Task\ 0$, the weights (w_i^j) are supplied to their corresponding PEs in a multicast mode, while the inputs (I_i^j) are supplied in a unicast mode. In such a case, the vertical switches are configured as a bus-based network. The banked global buffer acts as a master node, whereas all the PEs at each row or column are slave nodes. The horizontal switches are configured as a packet-based network, where the packets are transmitted hop by hop, reaching their destination. The large dimensions of the input activation favor the input stationary dataflow, as the reuse of the input elements avoids redundant memory access for the large matrix. The following equation shows how different kernels (w_i^j) are multicasted to the first and second row (row_0

and row_1) of sub-accelerator 0, while the inputs ($\underline{I_i^j}$) are pinned to the PEs and reused for two consecutive timesteps, t_0 and t_1 (underscores denote stationary elements).

$$Task\,0 \begin{cases} t_0 \begin{cases} row_0 : (\underline{I_0^0} \times w_0^0) + (\underline{I_1^0} \times w_1^0) + (\underline{I_6^0} \times w_2^0) + (\underline{I_7^0} \times w_3^0) \\ row_1 : (\underline{I_1^0} \times w_0^0) + (\underline{I_2^0} \times w_1^0) + (\underline{I_7^0} \times w_2^0) + (\underline{I_8^0} \times w_3^0) \end{cases} \\ t_1 \begin{cases} row_0 : (\underline{I_0^0} \times w_0^3) + (\underline{I_1^0} \times w_1^3) + (\underline{I_6^0} \times w_2^3) + (\underline{I_7^0} \times w_3^3) \\ row_1 : (\underline{I_1^0} \times w_0^3) + (\underline{I_2^0} \times w_1^3) + (\underline{I_7^0} \times w_2^3) + (\underline{I_8^0} \times w_3^3) \end{cases} \end{cases}$$

For $Task$ 1, however, the large channel size of the filter, together with the small size of the input channel necessitates a weight stationary dataflow. The weights are stored in the PE buffers while the inputs are iterated through different time steps. *Polyform*'s flexible router links multicast the same input element ($\underline{I_i^j}$) across each row of the sub-accelerator, in which NoC is configured as a bus. Meanwhile, the weight elements ($\underline{w_i^j}$) are unicasted vertically to each PEs. For $Task$ 1 in sub-accelerator 1, we compute the following in t_0 and t_1 timesteps:

$$Task\,1 \begin{cases} t_0 \begin{cases} row_0 : (I_0^0 \times \underline{w_0^0}) + (I_0^0 \times \underline{w_0^3}) + (I_0^0 \times \underline{w_0^6}) + (I_0^0 \times \underline{w_0^9}) \\ row_1 : (I_1^0 \times \underline{w_1^0}) + (I_1^0 \times \underline{w_1^3}) + (I_1^0 \times \underline{w_1^6}) + (I_1^0 \times \underline{w_1^9}) \end{cases} \\ t_1 \begin{cases} row_0 : (I_1^0 \times \underline{w_0^0}) + (I_1^0 \times \underline{w_0^3}) + (I_1^0 \times \underline{w_0^6}) + (I_1^0 \times \underline{w_0^9}) \\ row_1 : (I_2^0 \times \underline{w_1^0}) + (I_2^0 \times \underline{w_1^3}) + (I_2^0 \times \underline{w_1^6}) + (I_2^0 \times \underline{w_1^9}) \end{cases} \end{cases}$$

Such adaptive interconnects within each individual sub-accelerator enable an opportunistic choice of dataflow, introducing a new parameter when exploring the architectural design space for *Polyform*.

D. Routing Deadlock Avoidance

Routing deadlock can arise as a result of circular channel dependency. To resolve the circular channel dependency, turn-restricted routing algorithms or deadlock recovery techniques are implemented [10], [11] to remove the circular dependency. However, the data movement flows at each row/column in *Polyform*, and thus the decoupled horizontal and vertical data paths are exempt from the prerequisites forming any dependency.

III. Polyform Two-Stage GA-Based Search Model

GA, inspired by biological evolution, excels in addressing complex, multi-objective problems [12]. However, its applicability is limited when dealing with concurrent task allocation, resource allocation, and dataflows due to vast design possibilities and problem variations. Our approach involves treating the design space exploration as two distinct optimization problems: task allocation, referred to as 'stage 1,' and resource allocation and dataflow selection, referred to as 'stage 2'. These problems are addressed using a unified GA, as illustrated in Figure 3.

1) Stage 1 - Task Allocation: During the execution of multiple DNN models, each model can be divided into layers, referred to as tasks. The inherent independence between different DNNs allows for concurrent, non-blocking execution of these tasks, offering spatial parallelization opportunities. In stage 1, our aim is to achieve optimal task allocation and determine the appropriate number of sub-accelerators.

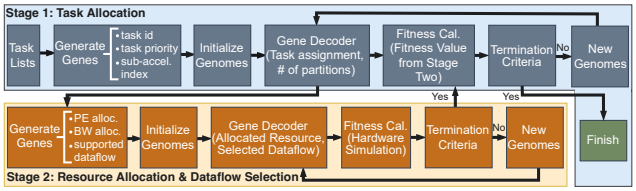


Figure 3: Illustration of our proposed two-stage genetic algorithm for task allocation (stage 1) and resource allocation and dataflow selection (stage 2).

2) Stage 2 - Resource Allocation and Dataflow Selection: Similarly, this stage starts with gene generation, but each gene represents the number of PEs, bandwidth, and dataflow selected for each sub-accelerator. Gene decoder feeds task allocation, dataflow, and resource allocation information into MAESTRO [13] to calculate fitness.

Fitness calculation is used to evaluate the configuration with specific objectives (e.g., latency, energy, or energy-delay product). Here, the fitness calculation is divided into two stages but correlated, as the maximum fitness value obtained in the second stage is returned to the first stage. Our two-stage GA terminates after 10,000 iterations, as determined by our empirical study, or when the fitness converges. The entire search duration ranges from 20 to 30 seconds, and the overhead can be overlapped with the execution time of multiple DNN workloads.

IV. Evaluation

A. ASIC Synthesis Tools

To evaluate the area and power consumption, we use the Synopsys Design Compiler NXT with the Synopsys SAED-PDK $32nm$ technology for the synthesis of *Polyform*'s architecture and estimate the power using Synopsys PrimeTime PX with worse case PVT corner.

B. Simulation Setup

We compare *Polyform* with several state-of-the-art designs: PREMA [1], which utilizes a single accelerator with sequential task scheduling based on priorities; Planaria [2], which supports dynamic spatial partitioning with a fixed dataflow and compute-memory resource ratio; and MAGMA-HO and MAGMA-HE [3], which consist of four homogeneous and heterogeneous sub-accelerators, respectively. Each sub-accelerator in MAGMA is designed with different numbers of PEs and dataflows. All designs have identical specifications: a 16×16 PE array, a memory bandwidth of 64 GB/s, and a global buffer size of 16 MB. Each PE includes 16 KB weight buffers, 8 KB input buffers, 8 KB output buffers, and 16 16-bit MAC units. We select a set of benchmark applications to examine different DNN applications with unique computation characteristics shown in Table III. For multi-DNN execution, each dataset consists of 20 randomly selected tasks from different application domains.

C. Performance Analysis

Figure 4 shows that *Polyform* outperforms PREMA, Planaria, MAGMA-HO, and MAGMA-HE by factors of 2.79×,

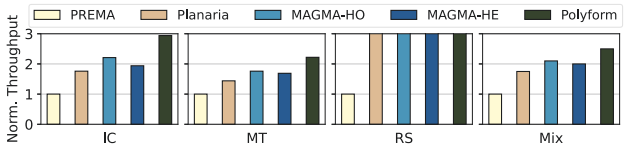


Figure 4: Throughput comparison with different DNN models, normalized to PREMA.

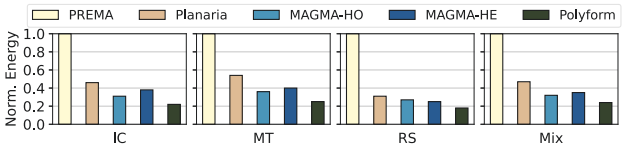


Figure 5: Energy comparison with different DNN models, normalized to PREMA.

Table III: Benchmarks Description

Domain	DNN Model
Image Classification (IC)	ResNet50, Mobilenets
Machine Translation (MT)	GNMT, Transformer-XL
Recommendation System (RS)	WnD, DIN
Mixed Workload	Random tasks from IC, MT, and RS

1.44×, 1.22×, and 1.28× on average. In the RS benchmark, *Polyform* achieves a comparable throughput to MAGMA-HO and MAGMA-HE because it is partitioned into four sub-accelerators with similar dataflow choices. In the IC benchmark, *Polyform* demonstrates more significant throughput enhancements, improving by factors of 2.94×, 1.76×, 1.30×, and 1.52× compared to PREMA, Planaria, MAGMA-HO, and MAGMA-HE, respectively. This improvement indicates *Polyform*'s ability to handle various sizes of DNN models in the IC benchmark.

D. Energy Analysis

Figure 5 illustrates that *Polyform* reduces the overall energy consumption by 77.8%, 49.0%, 29.3%, and 34.3% on average when compared to PREMA, Planaria, MAGMA-HO, and MAGMA-HE, respectively. This reduction is attributed to the increased throughput (resulting in a shorter makespan) and reduced off-chip memory access.

E. Sensitivity Study for Different Number of Tasks

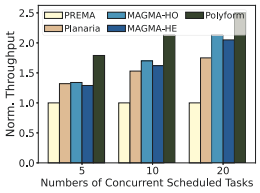
We investigate the impact of varying the number of Mixed tasks from 5 to 20 on the effectiveness of *Polyform*. Figure 6(a) demonstrates that *Polyform* improves the overall throughput by up to 2.50×, 1.43×, 1.33×, and 1.39× compared to PREMA, Planaria, MAGMA-HO, and MAGMA-HE, respectively. The improvement becomes more significant with a larger number of tasks. This observation highlights the crucial role of spatial parallelism in effectively handling a large set of tasks.

F. Area Analysis

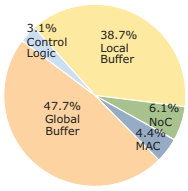
The area breakdown is shown in Figure 6(b). The overall accelerator requires an area of 39.89 mm^2, where the global buffers, NoC, MACs, local buffer, and control logic consume 19.03 mm^2, 2.43 mm^2, 1.76 mm^2, 15.43 mm^2 and 1.24 mm^2, respectively. Given its architectural simplicity, our proposed flexible router achieves a 56% reduction in area compared to a traditional 256-bit router.

V. Conclusion

In this paper, we propose an energy and performance-efficient DNN accelerator in pursuit of multi-DNN execution. The proposed accelerator employs two unique designs for the exploitation of DNN parallelism. First, we propose a flexible



(a)



(b)

Figure 6: (a) Throughput analysis for the various number of concurrent scheduled Mix tasks, and (b) the area breakdown of our proposed accelerator.

accelerator architecture that can be spatially partitioned into multiple smaller accelerators called sub-accelerators. Each sub-accelerator can be optimized in accordance with different parallelization and data reuse strategies. Second, a GA-based framework is proposed to simultaneously manage resource allocation, dataflow selection, and task scheduling. Our proposed design can achieve significant energy and throughput improvements as compared to prior work.

References

[1] Yujeong Choi et al. PREMA: A predictive multi-task scheduling algorithm for preemptible neural processing units. In *Proc. of HPCA*, pages 220–233. IEEE, 2020.

[2] Soroush Ghodrati et al. Planaria: Dynamic architecture fission for spatial multi-tenant acceleration of deep neural networks. In *Proc. of MICRO*, pages 681–697. IEEE, 2020.

[3] Sheng-Chun Kao et al. MAGMA: An optimization framework for mapping multiple dnns on multiple accelerator cores. In *Proc. of HPCA*, pages 814–830. IEEE, 2022.

[4] Yu-Hsin Chen et al. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *Proc. of ISCA*, pages 367–379. IEEE, 2016.

[5] Zidong Du et al. ShiDianNao: Shifting vision processing closer to the sensor. In *Proc. of ISCA*, pages 92–104. IEEE, 2015.

[6] Jiaqi Yang et al. Venus: A versatile deep neural network accelerator architecture design for multiple applications. In *Proc. of DAC*. IEEE, 2023.

[7] Lingxiang Yin et al. Aries: Accelerating distributed training in chiplet-based systems via flexible interconnects. In *Proc. of ICCAD*, 2023.

[8] Seah Kim et al. MoCA: Memory-centric, adaptive execution for multi-tenant deep neural networks. In *Proc. of HPCA*, pages 828–841, 2023.

[9] Avinash Kodi et al. iDEAL: Inter-router dual-function energy and area-efficient links for network-on-chip (NoC) architectures. In *Proc. of ISCA*, pages 241–250. IEEE, 2008.

[10] William James Dally et al. *Principles and practices of interconnection networks*. Elsevier, 2004.

[11] Hao Zheng et al. Adapt-noc: A flexible network-on-chip design for heterogeneous manycore architectures. In *Proc. of HPCA*, pages 723–735. IEEE, 2021.

[12] Lingxiang Yin et al. Exploring architecture, dataflow, and sparsity for gcn accelerators: A holistic framework. In *Proc. of GLSVLSI*, pages 489–495. ACM, 2023.

[13] Hyoukjun Kwon et al. MAESTRO: A data-centric approach to understand reuse, performance, and hardware cost of dnn mappings. In *IEEE Micro*, pages 20–29. IEEE, 2020.