PipeFuser: Building Flexible Pipeline Architecture for DNN Accelerators via Layer Fusion

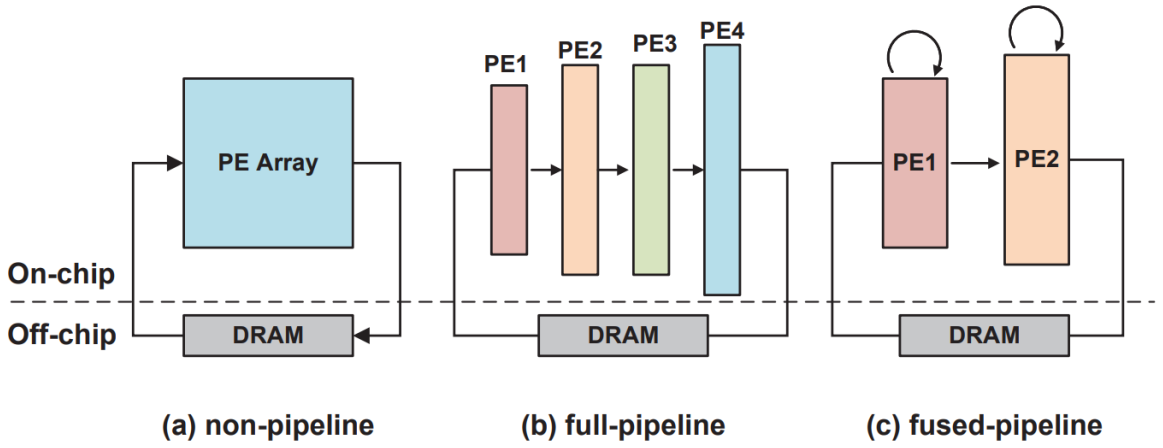
Xilang Zhou[1], Shuyang Li[1], Haodong Lu[2], Kun Wang[1,*]

[1]State Key Lab of ASIC & System, Fudan University, Shanghai, China
[2]Nanjing University of Posts and Telecommunications, Nanjing, China.
kun.wang@ieee.org

Presentation Outline

- **Introduction**
- **Preliminaries**
  - Background
  - Design Challenges
- **PipeFuser Design**
  - Microarchitecture and Computation Flow
  - Co-design Engine
- **Experiment**
- **Conclusion**

Introduction



Three architectural paradigms of DNN accelerators:

(a) non-pipeline: a unified accelerator processing all layers

(b) full-pipeline: Dedicated accelerators tailored for each individual DNN layer, fully pipelined

(c) fused-pipeline : Pipelined accelerators of fused layer groups

Introduction

Non-pipeline: need to accommodate diverse DNN layers, frequent off-chip memory access

Full-pipeline: scarce resource and large on-chip buffering demand with increasing model depth; pipeline imbalance

Fused-pipeline: improved pipeline balance; alleviate resource constraints; less on-chip storage demand

Introduction

Contribution:

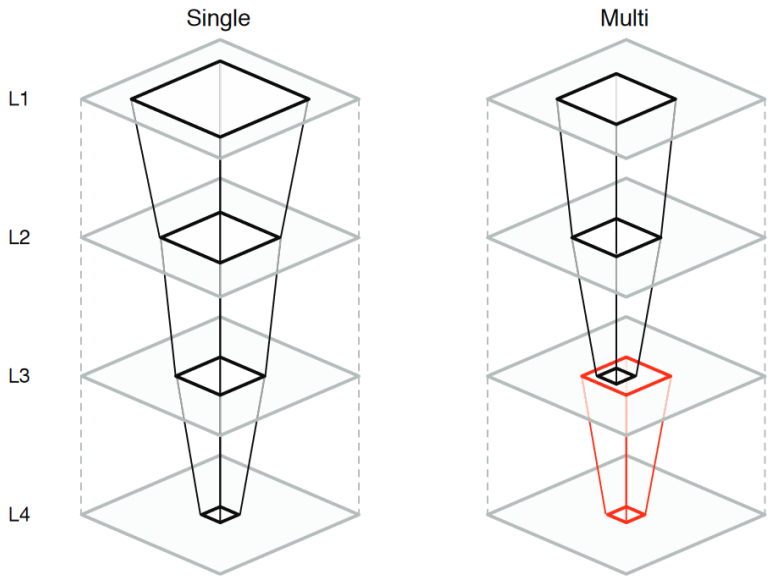
- We leverage the layer fusion technique and fused-pipeline architecture for DNN accelerator deployment.
- We develop an end-to-end automation framework, **PipeFuser,** for fused-pipeline deployment.
- A GA-based co-design engine for fast fusion strategy determination.
- Competitive performance with significant performance speedup and higher flexibility compared to non-pipeline and full-pipeline architectures.

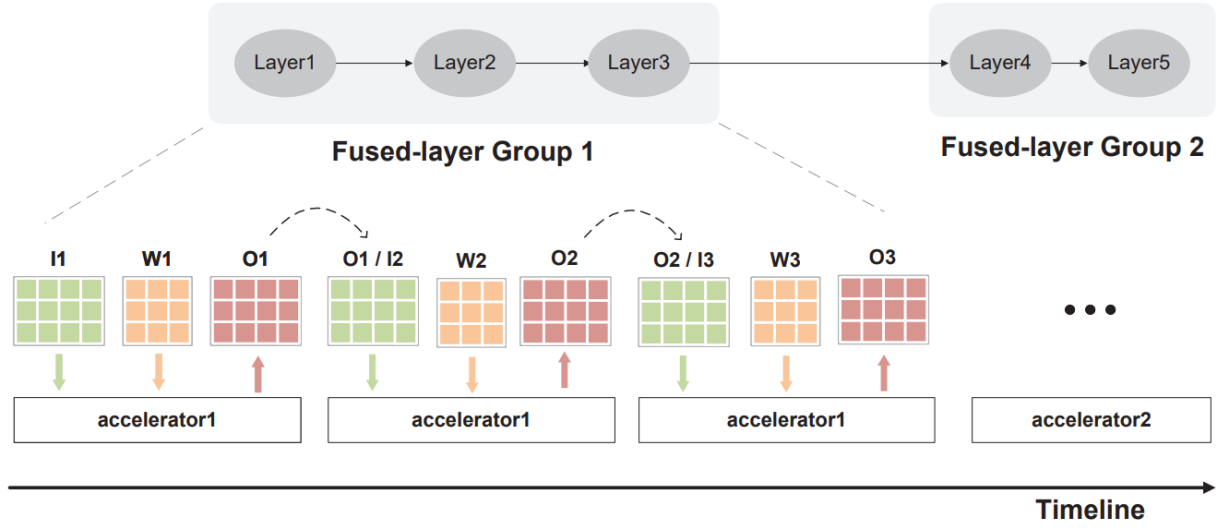
Presentation Outline

- **Introduction**
- **Preliminaries**
  - Background
  - Design Challenges
- **PipeFuser Design**
  - Microarchitecture and Computation Flow
  - Co-design Engine
- **Experiment**
- **Conclusion**

Background on Layer Fusion Technique

First proposed in [1]
Combined with **tiling** technique
Reduces intermediate data transfer between on-chip and off-chip within the fusion group

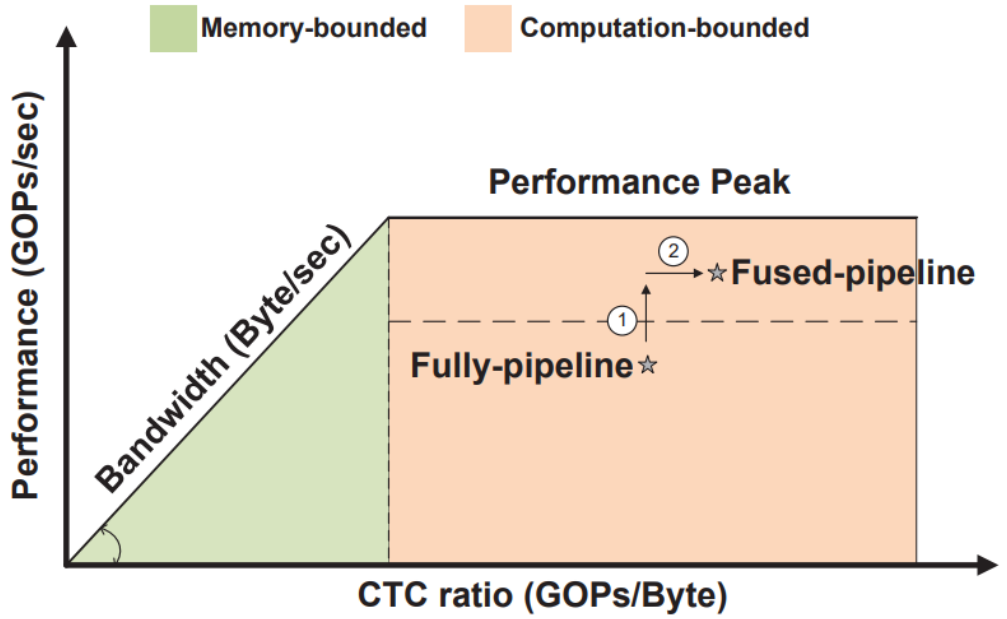




[1]M. Alwani, H. Chen, M. Ferdman, and P. Milder, “Fused-layer cnn accelerators,” in MICRO, 2016.

Performance analysis via Roofline Model

- Compute-To-Communication (CTC) ratio: # of MAC operations per byte
- Full-pipeline and fused-pipeline: computation-bounded
- Fused-pipeline arechitecture is more resource-efficient and performance-optimized



Design Challenges

- Challenge 1: How to formulate an efficient fusion strategy.
- Challenge 2: How to explore the expansive mapping space.

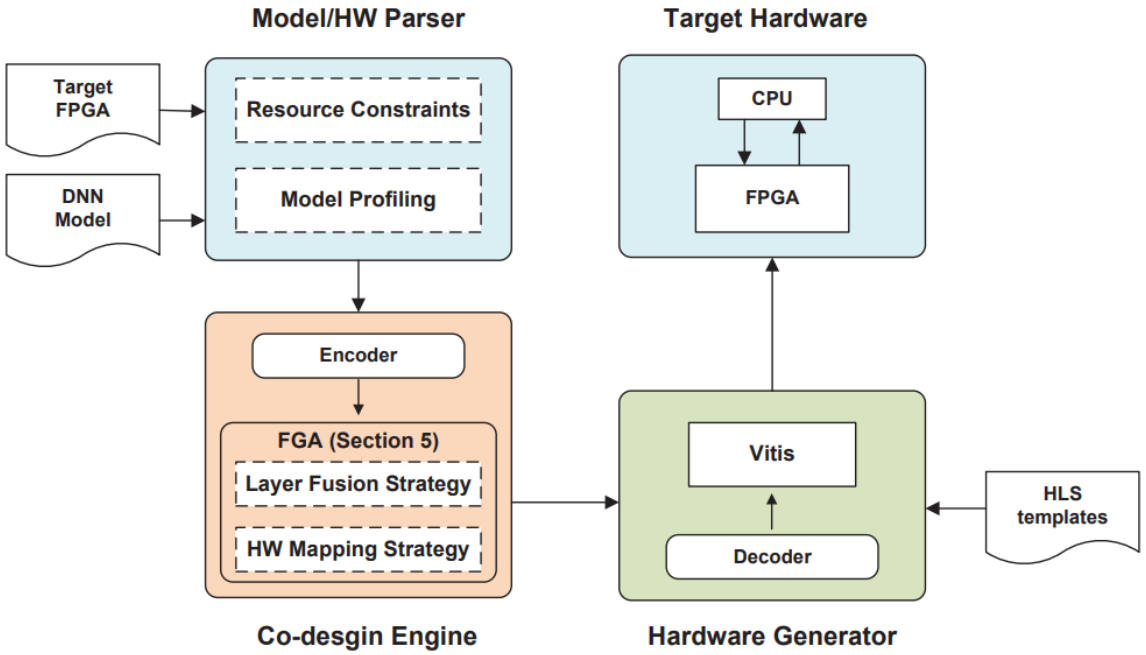
↓

Co-optimization problem

Presentation Outline

- **Introduction**
- **Preliminaries**
  - Background
  - Design Challenges
- **PipeFuser Design**
  - Microarchitecture and Computation Flow
  - Co-design Engine
- **Experiment**
- **Conclusion**

Framework Overview



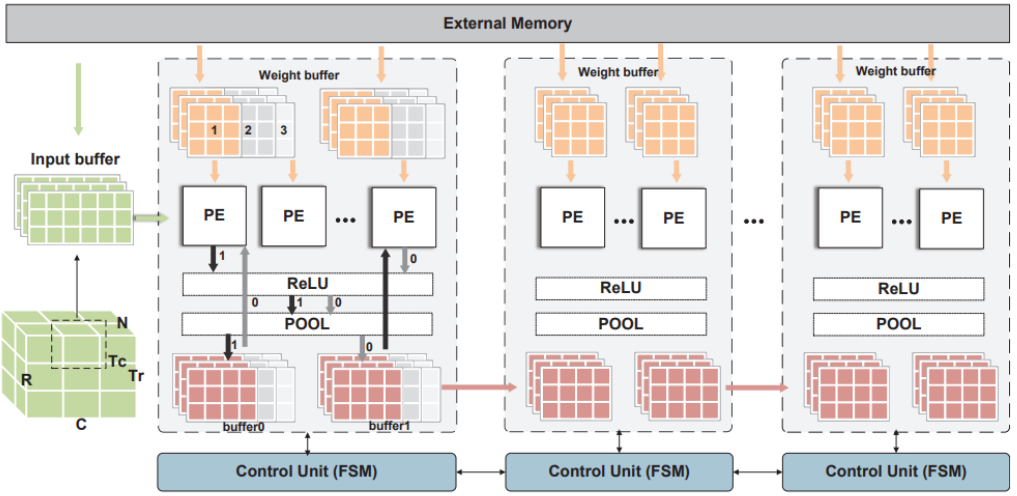
Three main components of PipeFuser:

Model/Hardware Parser

Co-design Engine

Hardware generator

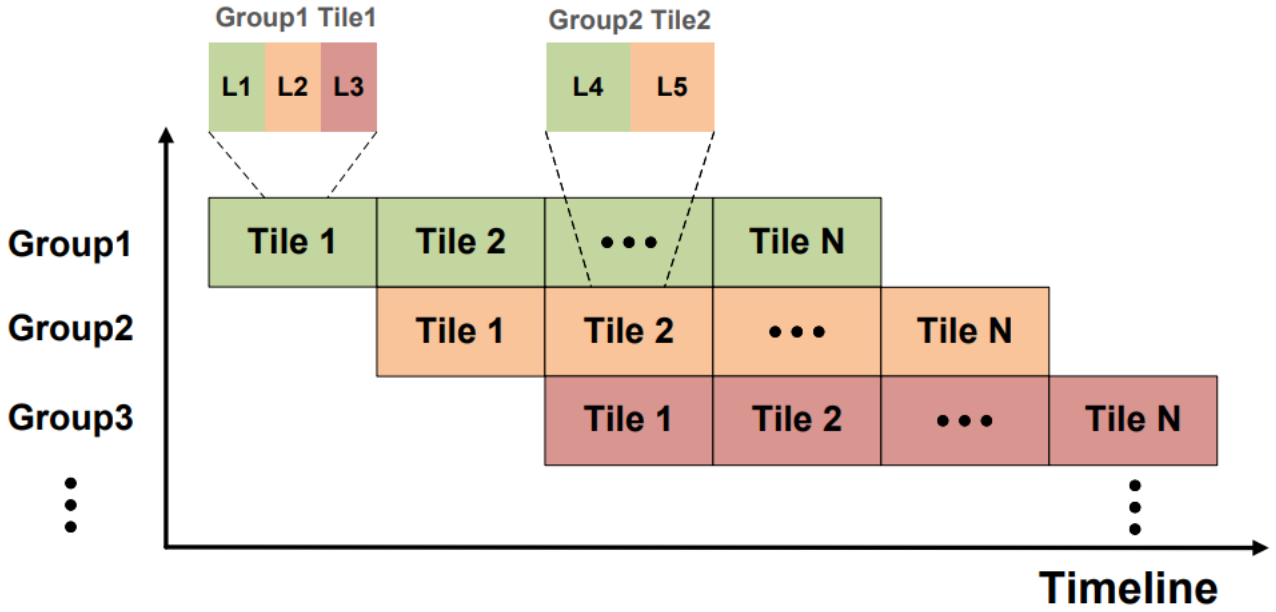
Microarchitecture and Computation Flow



Microarchitecture of Pipefuser

Modules:

- Computation
- Memory
- Control Unit



Computation Flow

Co-design Engine

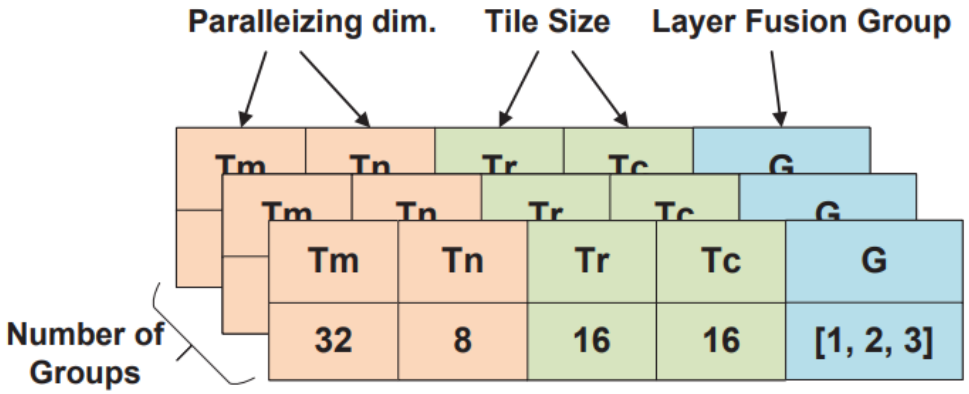
The co-design Engine is based on a Genetic Algorithm
GA is a search heuristic inspired by the process of natural selection

Four phases are considered in the GA:

- Initialization
- Crossover
- Mutation
- Evaluation and Selection

Co-design Engine

The fusion and mapping parameters (template parameters) need to be encoded into genome form first



Notation:
T_r and T_c: dimensions of the tile size (global)
T_m and T_n: parallelism dimensions
G: array of layers that are fused together

Total number of chromosomes = The number of fusion groups = the number of accelerators

Co-design Engine

Algorithm Flow

Algorithm 1: Explore Fusion and Mapping Spaces

Data: the probability of the crossover operation P_c, the probability of the mutation operation P_m, the number of the solutions within the population T, the number of generation G, the total layer of the net N_{layer}

Result: optimal number of accelerator n_{acc}, optimal solution S

```
while n_acc <= N_layer do
    initialize population X of T individuals;
    while g <= G do
        evaluate fitness of each individual;
        select parents for next generation;
        if random(0,1) < P_c then
            create offspring through crossover;
            exclude ineligible offspring;
        end
        if random(0,1) < P_m then
            create offspring through mutation;
            exclude ineligible offspring;
        end
    end
end
```

Initialization: initialize population X of T individuals

Crossover:

- directly swapping T_r and T_c between two sets of solutions.
- T_m and T_n are swapped within the same fusion group

Mutation:

- Mutation-Parallel-Dim
- Mutation-Tile Size
- Mutation-Group

Evaluation and Selection: tournament selection

Presentation Outline

- **Introduction**
- **Preliminaries**
  - Background
  - Design Challenges
- **PipeFuser Design**
  - Microarchitecture and Computation Flow
  - Co-design Engine
- **Experiment**
- **Conclusion**

Experiment Setup

- **Platform**

Xilinx Alveo U200 FPGA hosted on Intel Xeon Server CPU (Gold 5218R@2.10GHz) , synthesized with Vitis 2022.1

- **Hardware Baseline**

non-pipeline: HybridDNN[1], FlexCNN[2]
full-pipeline: DNNbuilder[3], TGPA[4]
hybrid architecture: DNNExplorer[5]
segment-grained pipeline: Deepburning-SEG[6]

- **DNN Workload**

VGGNet, ResNet
8-bit fixed-point data representation

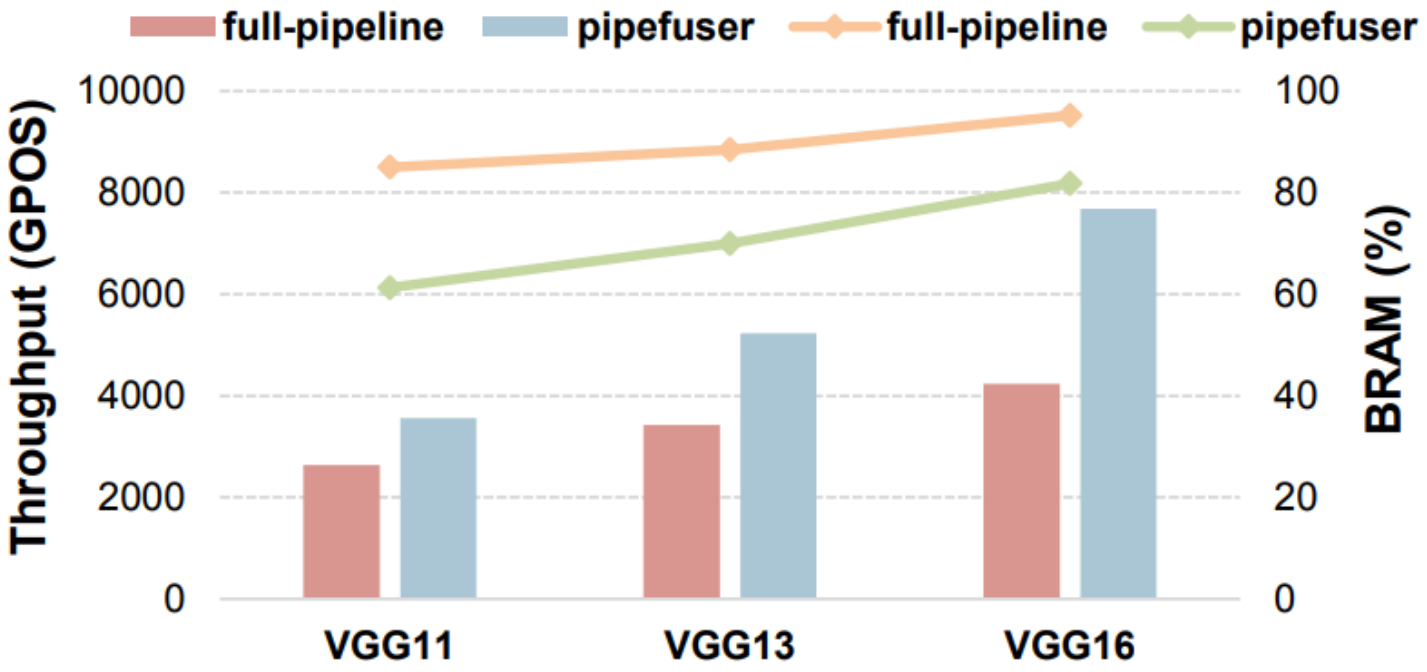
[1] H. Ye, X. Zhang, Z. Huang, G. Chen, and D. Chen, “Hybriddnn: A framework for high-performance hybrid dnn accelerator design and implementation,” in DAC, 2020.
[2] S. Basalama, A. Sohrabizadeh, J. Wang, L. Guo, and J. Cong, “Flexcnn: An end-to-end framework for composing cnn accelerators on fpga,” in TRETS, 2023.
[3] X. Zhang, J. Wang, C. Zhu, Y. Lin, J. Xiong, W.-m. Hwu, and D. Chen, “Dnnbuilder: An automated tool for building high-performance dnn hardware accelerators for fpgas,” in ICCAD, 2018.
[4] X. Wei, Y. Liang, X. Li, C. H. Yu, P. Zhang, and J. Cong, “Tgpa: Tile-grained pipeline architecture for low latency cnn inference,” in ICCAD, 2018.
[5] X. Zhang and D. Chen, “Dnnexplorer: a framework for modeling and exploring a novel paradigm of fpga-based dnn accelerator,” in ICCAD, 2020.
[6] X. Cai, Y. Wang, X. Ma, Y. Han, and L. Zhang, “Deepburning-seg: Generating dnn accelerators of segment-grained pipeline architecture,” in MICRO, 2022.

Experiment: Comparison with Previous Designs

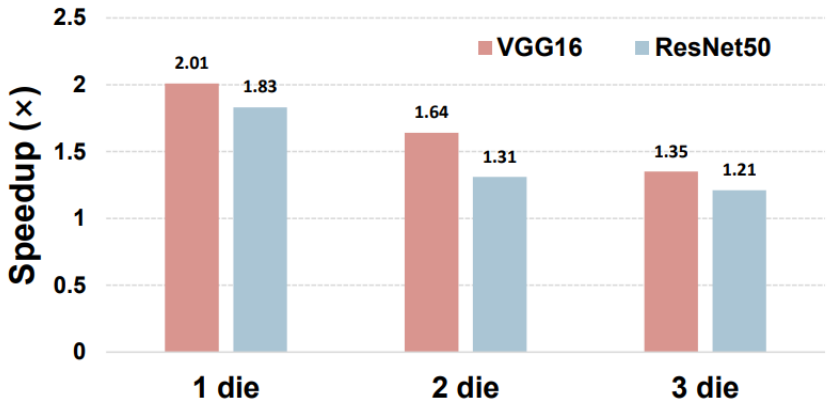
TABLE I
PERFORMANCE COMPARISON WITH PREVIOUS WORKS

Design	HybridDNN [15]	FlexCNN [3]	DNNBuilder [21]	TGPA [13]	DNNExplorer [20]	DeepBurning-SEG [4]	Ours
Device	VU9P	U250	KU115	VU9P	KU115	KU115	U200
Model	VGG16	VGG16	VGG16	VGG19	VGG16	VGG16	VGG16
Precision	12bit	16bit (8bit)	16bit (8bit)	16bit	16bit	8bit	8bit
Freq.(MHz)	167	241	235	210	200	235	250
DSPs	5136 (75.9%)	4736 (37.98%)	4318 (78%)	4096 (60%)	4444 (80.5%)	5128 (92.9%)	5026 (73.5%)
BRAM	NA	2692 (45.93%)	1578 (81%)	1690 (78%)	1648 (76.3%)	1486 (76.9%)	1310 (81.8%)
Throughput (GOPS)	3376	1543 (2329)	2011 (4022)	1510	1702	4778	**7684**
DSP Density (GOPS / DSPs)	0.66	0.33	0.93	0.37	0.38	0.93	**1.52**

Experiment: Full-pipeline vs Fused-pipeline



Experiment: Scalability Evaluation



- With Limited Resources
- With Increasing Layer Depth

TABLE II
SCALABILITY EVALUATION

Model	VGG11	VGG13	VGG16	VGG19	ResNet50	ResNet152
Group.number	3	3	4	5	4	6
Freq.(MHz)	250	250	250	235	220	220
DSPs	3982	4390	5026	5621	3560	4496
BRAM	982	1073	1310	1582	1244	1543
Throughput (GOPS)	3560	5231	7684	8431	2106	5415

Presentation Outline

- **Introduction**
- **Preliminaries**
  - Background
  - Design Challenges
- **PipeFuser Design**
  - Microarchitecture and Computation Flow
  - Co-design Engine
- **Experiment**
- **Conclusion**

Conclusion

PipeFuser

- Optimized Fused-Pipeline Architecture
- Automated End-to-End Framework

  GA-based Co-design Engine
- Evaluation
- Higher Throughput than Non-pipeline and Full-pipeline Architectures
- Higher DSP density
- Better Deployment Flexibility

Acknowlegdement

This work was financially supported in part by National Key Research and Development Program of China under Grant 2021YFA1003602, and in part by Shanghai Pujiang Program under Grant 22PJD003.

Thank you!

kun.wang@ieee.org