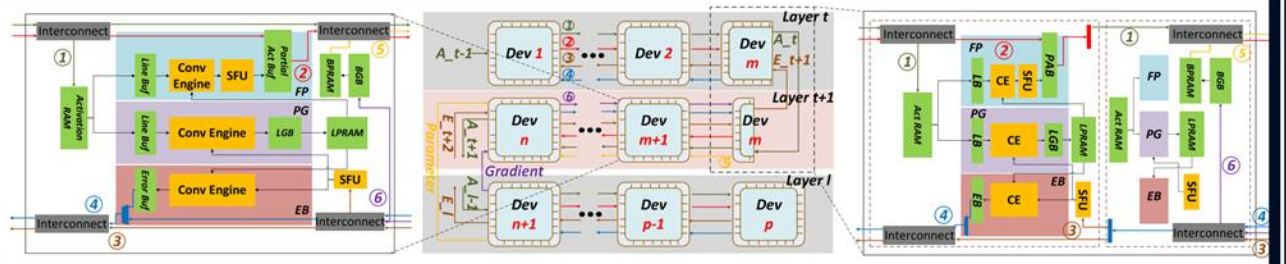
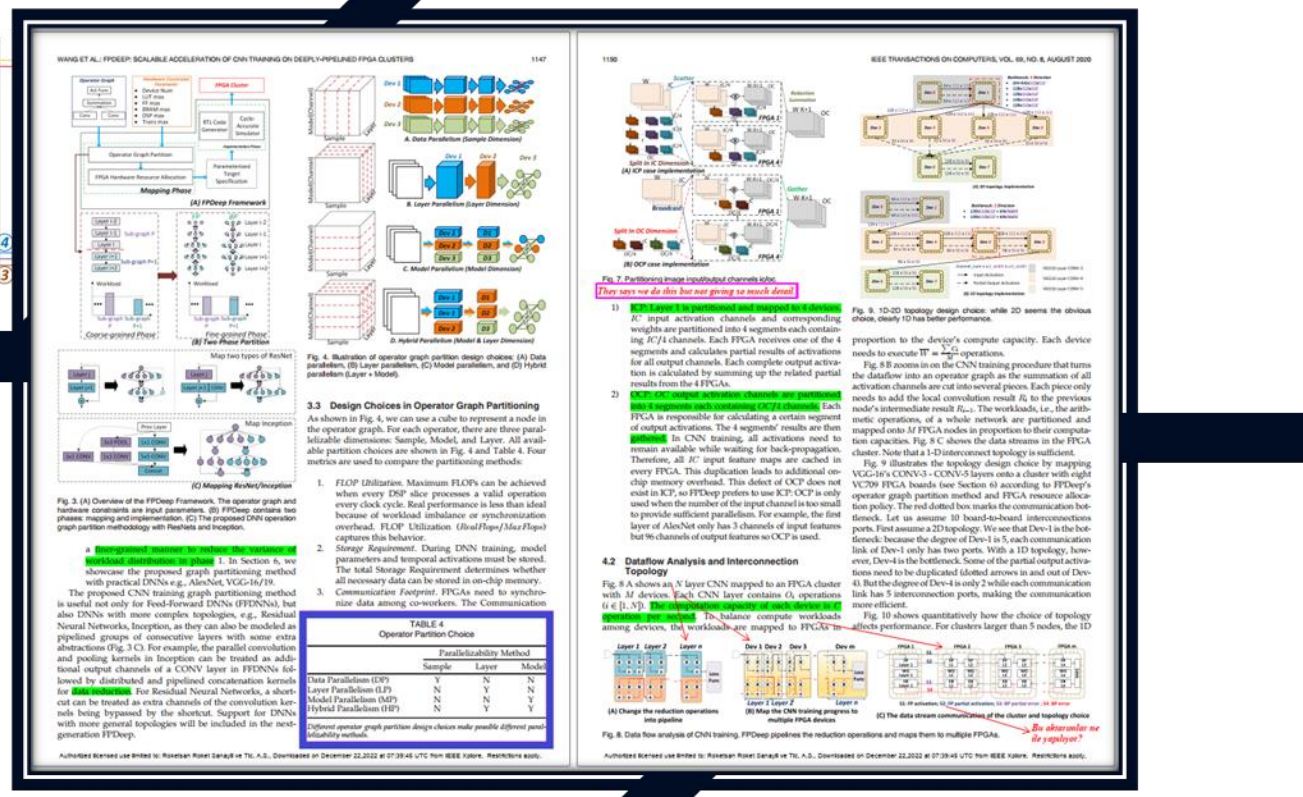
FPDeep: Scalable Acceleration of CNN Training on Deeply-Pipelined FPGA Clusters

-IEEE TRANSACTIONS ON COMPUTERS, VOL. 69, NO. 8, AUGUST 2020-





- United States Department of Energy
- Pacific Northwest National Laboratory
- The Center for Advanced Technology Evaluation
- National Science Foundation
- Microsoft
- Intel

CNN

Parallelism

Distributed Systems

FPGA

Parallel Computing

CMP641 Software Development for Parallel Computers

Kürşat Çakal – N21138461

kursatckl@gmail.com

MILESTONES

01 Abstract

- Current Industry Demands
- CNN with Lightweight Environments

02 Introduction of FP-Deep

- Parallelism Approaches
- Start Point
- Existing Problems of Literature Solutions
- Short Match Background of CNN

03 FP-Deep Architecture

- FP-Deep Framework
  - Flow Diagram
  - Operator Graph Partitioning Method
  - Resource Allocation Policy
- FP-Deep Imlementation
  - ICP Implementation
  - OCP Implementation
  - MPI Simulation Supported Physical Implementation

04 Results

- Performance Metrics
- Resource Utilization
- Convergence Rate
- Cluster Level Experimental Result by Comparing Literature

05 Conclusion

- Literature Contribution
- Strengths
- Short Recap

ABSTRACT

- Deep Convolutional Neural Networks
  - Revolutionized Numerous Application
    - Aviation
    - Space
    - Military
    - Wearable Technologies
    - Health Sciences
  - Lightweight Environment Requirements
    - High Mobility
    - Low Power Consumption
    - High Performance
- Open-Ended Demands for Deep Learning Applications in the Industry
  - More Complex Networks
  - Larger Datasets
  - Lower Training Times
  - Less Power Consumption

- **New Computing Solutions are Critical !!!**

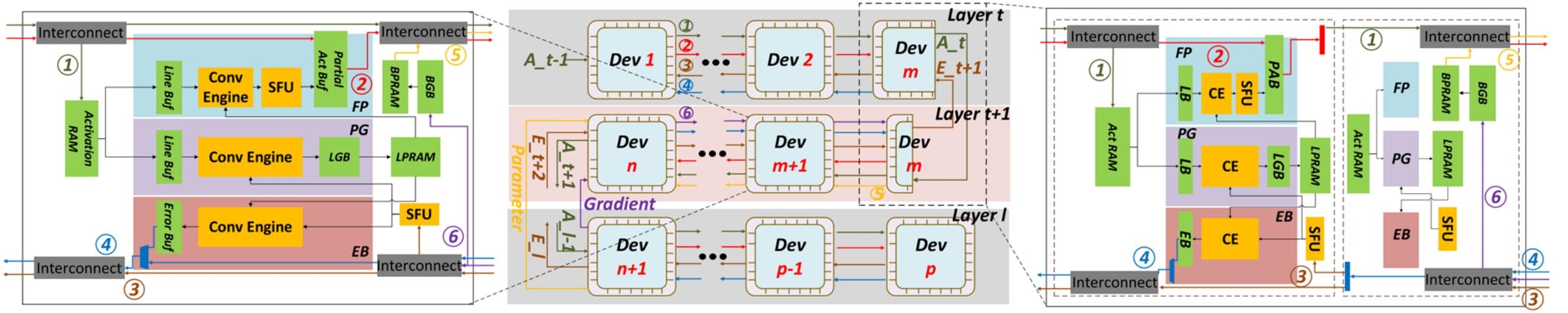
- **Performing** and **Scaling** CNN Computations with Lightweight Systems !

- FPDeep Framework is proposed in this study.

- **Parallelism Methods** proposed **for** configuring **distributed** and **reconfigurable Devices Clusters**
  - Fast Training and Implementation Processes
  - CNN Architectures in Lightweight Environments

INTRODUCTION

- The FPDeep, a novel **FPGA-Cluster-Based Training Framework for CNNs** that solves these problems.
- FPDeep Framework **Mechanism** does this by using
  - Data Parallelism
  - Layer Parallelism
  - Model Parallelism
  - Hybrid Parallelism
- It uses novel **workload mapping** and **weight balancing** strategies
- Each device computes only certain layers or a part of a single layer
- Each device is **optimized independently** with respect to its **own computation volume**.
- The cluster is now a single **fine-grained** pipeline so the batch size can be arbitrarily small.



(B) Implementation when entire device work for one layer *(A) Overall architecture of mapping (l-t+1) layers to p devices* *(C) Implementation when one device work for two adjacent layers*

① Input activation for current layer ② Partial output activation for current layer ③ Error from next layer ④ Partial error for current layer ⑤ Parameter stored remotely
⑥ Gradient transferred to remote parameter *A_t* Output activation from layer t *E_t* Error from layer t ***PAB*** Partial activation buffer ***SFU*** Special function unit

Activation RAM – Balanced Parameter RAM – Local Parameter RAM – Local Gradient RAM – Balanced Gradient Buffer – Local Gradient Buffer – Balanced Gradient Buffer

INTRODUCTION

Name	Network Type	Data Set	Training Accuracy		Test Accuracy	
			SB	LB	SB	LB
F_i	FC	MNIST	$99.66\% \pm 0.05\%$	$99.92\% \pm 0.01\%$	$98.03\% \pm 0.07\%$	$97.81\% \pm 0.07\%$
F_2	FC	TIMIT	$99.99\% \pm 0.03\%$	$98.35\% \pm 2.08\%$	$64.02\% \pm 0.20\%$	$59.45\% \pm 1.05\%$
C_1	Shallow Conv	CIFAR-10	$99.89\% \pm 0.02\%$	$99.66\% \pm 0.20\%$	$80.04\% \pm 0.12\%$	$77.26\% \pm 0.42\%$
C_2	Deep Conv	CIFAR-10	$99.99\% \pm 0.04\%$	$99.99\% \pm 0.01\%$	$89.24\% \pm 0.12\%$	$87.26\% \pm 0.07\%$
C_3	Shallow Conv	CIFAR-100	$99.56\% \pm 0.44\%$	$99.88\% \pm 0.30\%$	$49.58\% \pm 0.39\%$	$46.45\% \pm 0.43\%$
C_4	Deep Conv	CIFAR-100	$99.10\% \pm 1.23\%$	$99.57\% \pm 1.84\%$	$63.03\% \pm 0.50\%$	$57.81\% \pm 0.17\%$

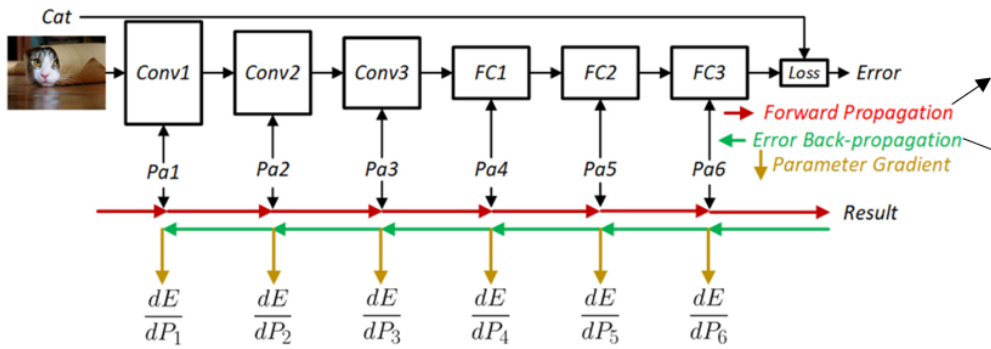
Importance of Batch Size

- Firstly Analysing and Comparing Large Batch – Small Batch;
  - It is observed that LB does not decrease the accuracy derived from the training set
  - The LB does substantially reduce the testing accuracy.
  - Batch Parallelism is a problem in the literature.

- Authors applies **new parallelism paradigm** called **Pipeline Parallelism**.

- Pipeline parallelism means **multiple steps depend on each other**, but **the execution can overlap** and the output of one step is streamed as input to the next step **without break** the **computation chain flow**.

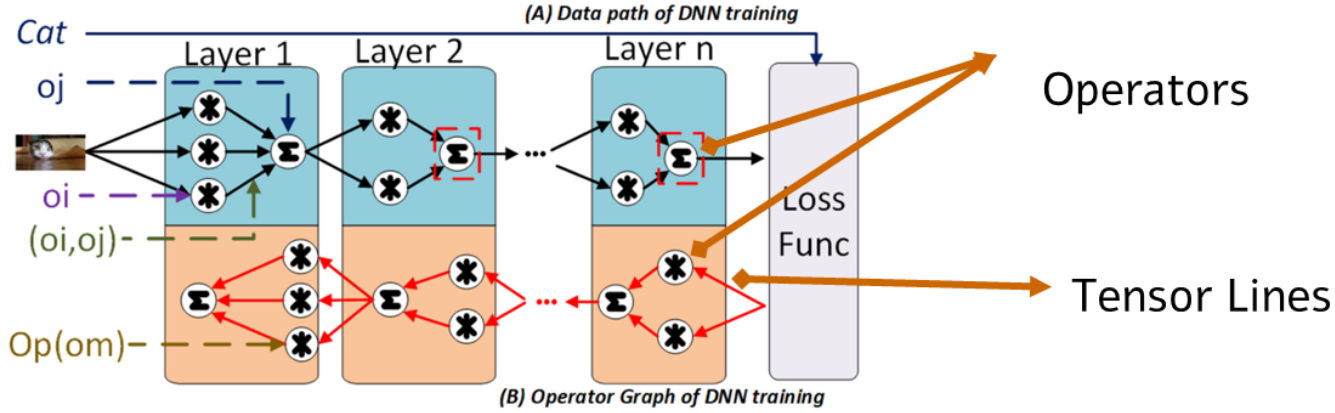
INTRODUCTION



(A) Data path of DNN training

- Error of output features.
- It is calculated:
  - Compare(inference, label)

Errors back propagated through CNN.



(B) Operator Graph of DNN training

- Literature approaches have two problems.

  - **First**, connected flow; **a particular layer waits until the previous layer finishes**. Therefore, **all** hidden layer **features must be cached until** the last feature of a layer is obtained by **back propagation phase**. This **requires** large storage requirements which means **usage of other memories** apart from OCM.

  - **Second**, the **computational workloads varies** greatly among layers. A naive workload distribution can result in a **large number of idle cycles** due to **inter-layer dependencies**.

MATH BACKGROUND

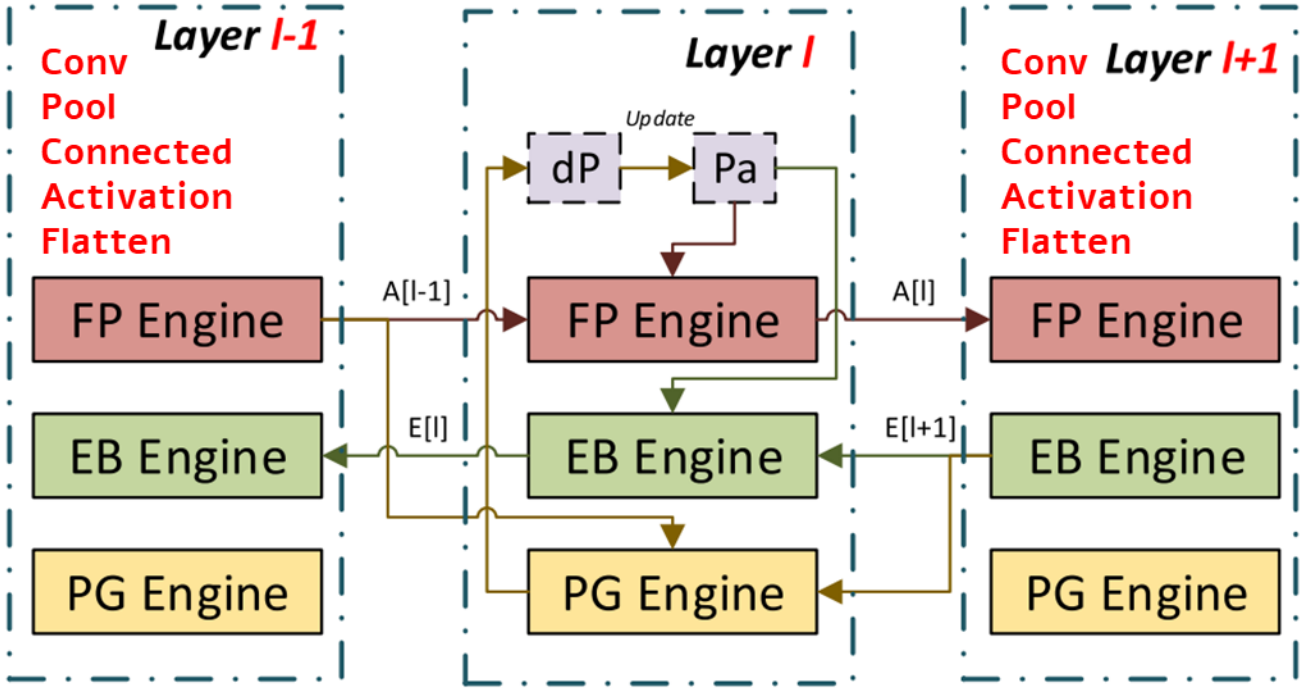


TABLE 2
Definitions of Symbols Used in the Equations

Notation	Description
A[b][l][c]	Activation[batch-id][layer-id][channel-id]
Pa[b][l][o][i]	Parameter[batch-id][layer-id][output-channel-id] [input-channel-id]
dP[b][l][o][i]	Differential of the parameter[batch-id][layer-id] [output-channel-id][input-channel-id]
E[b][l][c]	Error of the layer[batch-id][layer-id][channel-id]
W	Size of the activation
K	Size of the convolution kernel
IC	Number of the input channels
OC	Number of the output channels

- ❑ Matrix Multiplications
- ❑ Activation Functions
- ❑ Loss Functions
- ❑ Differential Operations
- ❑ Error Calculations
- ❑ Data Exchange

Too much work to parallelize !!

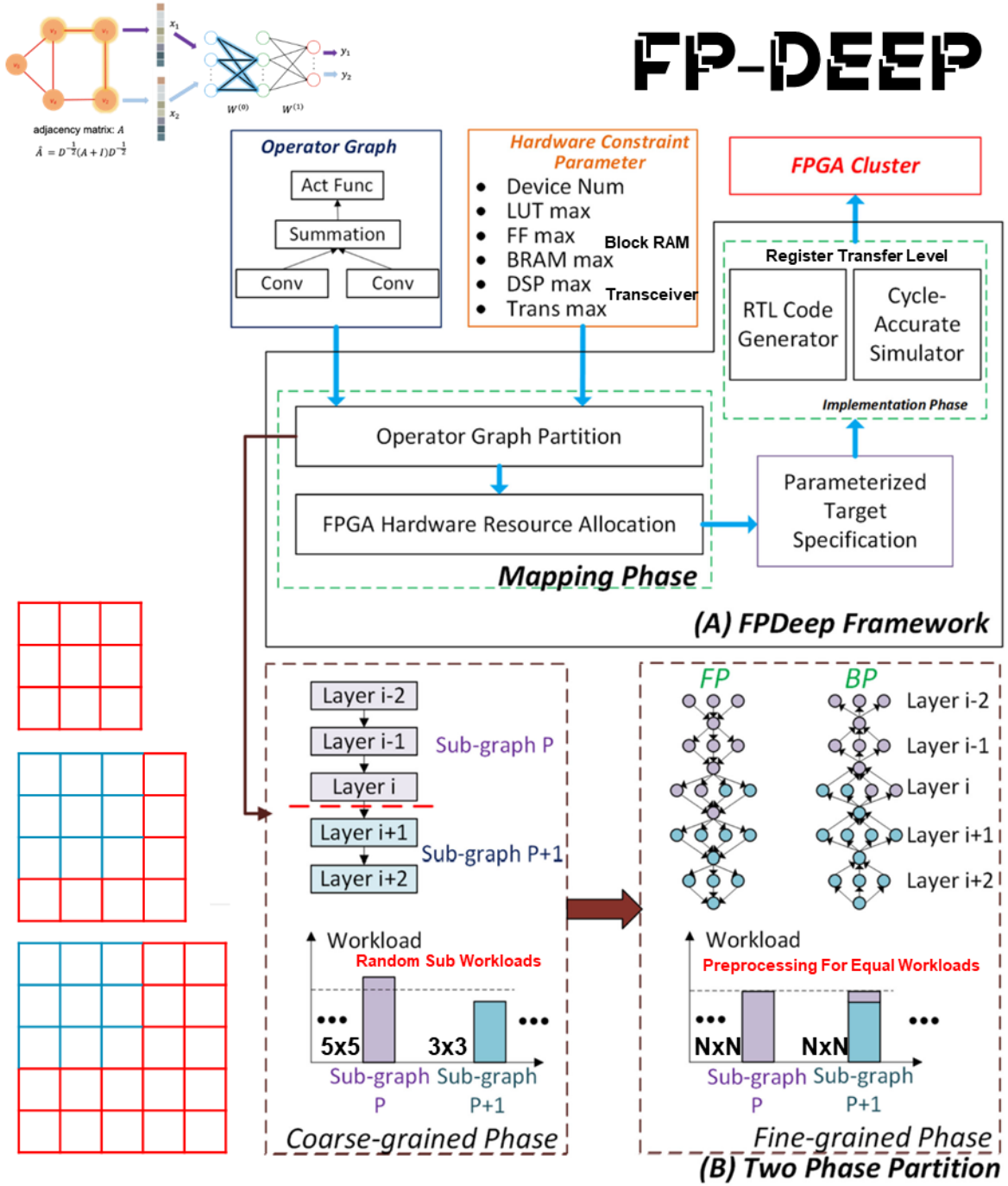
$$A[b][l][c] = \sum_p A[b][l-1][p] * Pa[b][l][c][p] \quad (1)$$

$$E[b][l][c] = \sum_p E[b][l+1][p] * Pa[b][l][p][c] \quad (2)$$

$$dP[b][l][p][q] = A[b][l-1][q] * E[b][l][p] \quad (3)$$

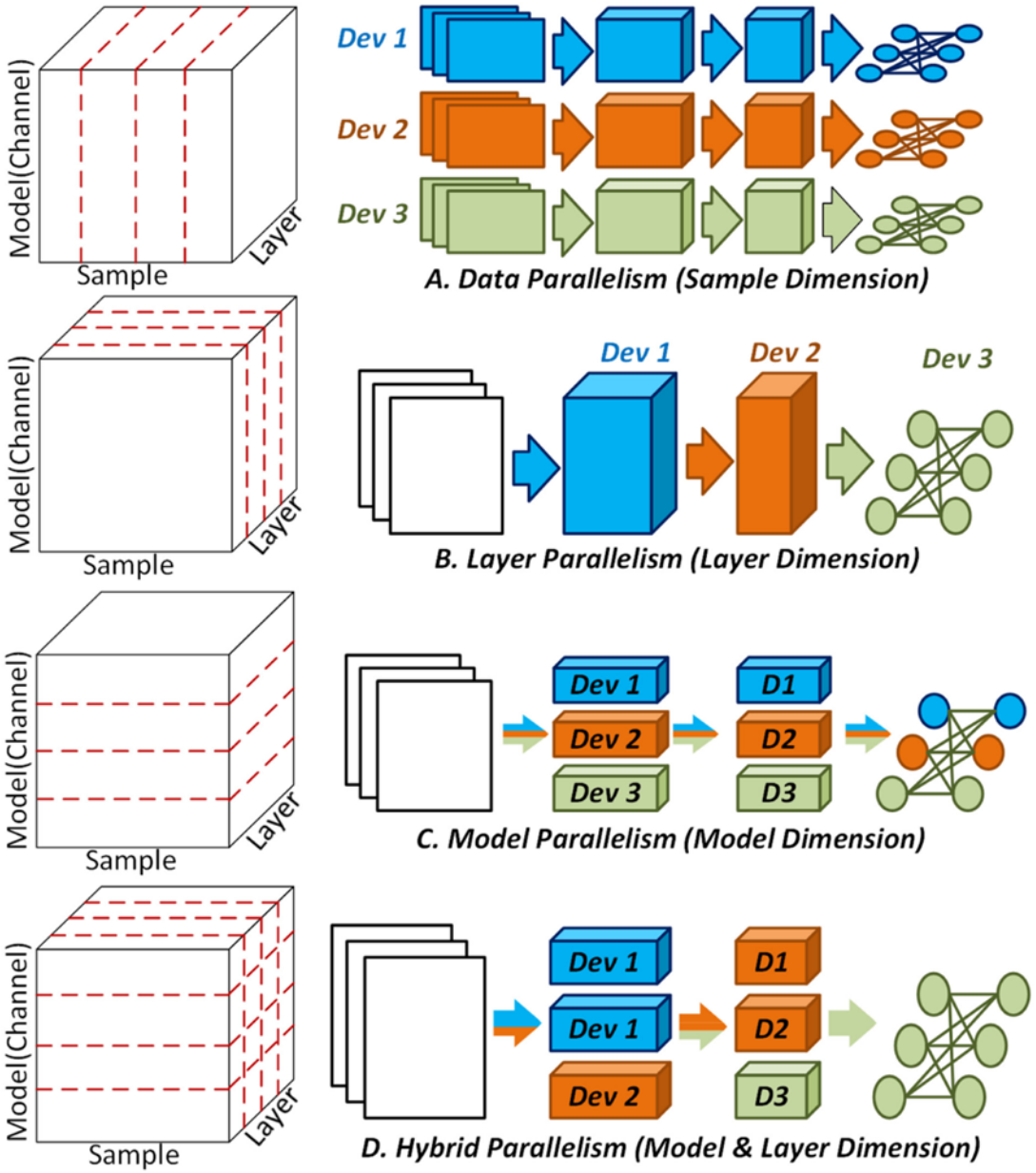
$$Pa[b+1][l][p][q] = dP[b][l][p][q] + Pa[b][l][p][q]. \quad (4)$$

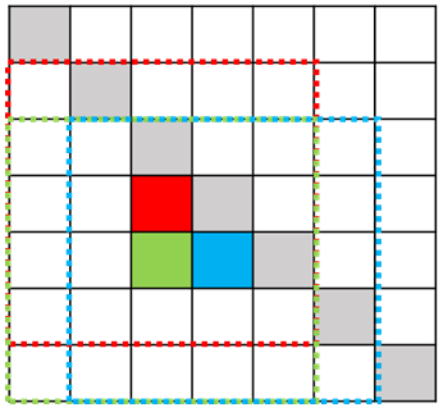
FP-DEEP FRAMEWORK



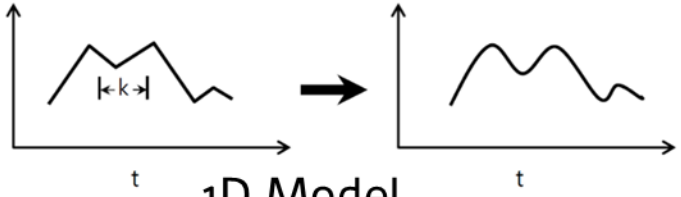
- Finding the global optimal solution of graph partitioning is **NP-hard** problem.
- FPDeep **takes advantage of** the fact that DNN training logic can be modeled as a **computational pipeline** consisting of **groups** of **consecutive layers**
- FPDeep graph partitioning works in two phases:
  - Course Grained
    - The coarse-grained **graph** *G* is **partitioned** into **multiple** (number of FPGAs) **subgraphs** with similar sizes of workloads.
    - **Subgraphs created with a variance factor**.
  - Fine Grained
    - FPDeep performs **reallocation** of FPGA **resources** in a finer-grained manner to **reduce** the **variance** of **workload distribution** in phase 1.
    - It **requires pre-processing** but comes with **efficient resource consumption**.

FP-DEEP FRAMEWORK

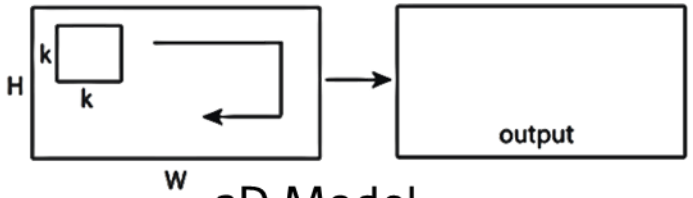




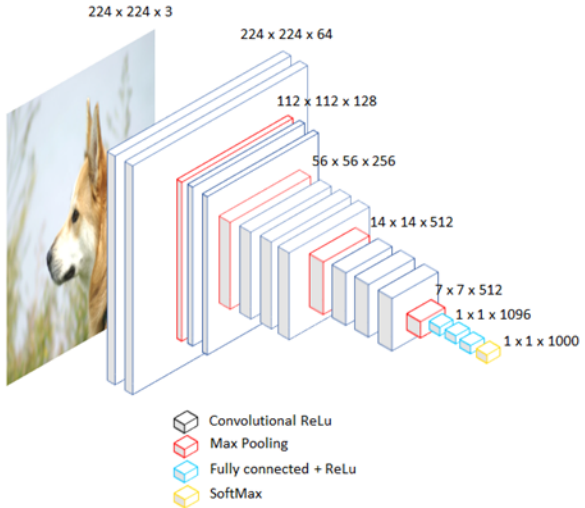
Data Parallelism



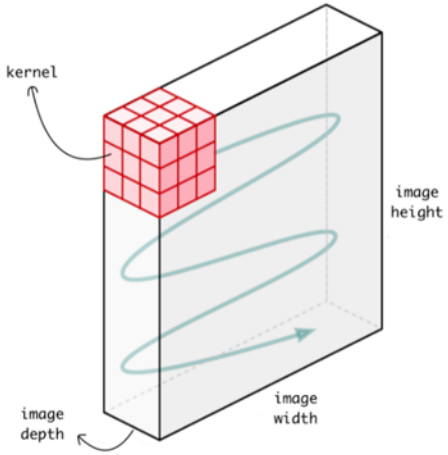
1D Model



2D Model



Layer Parallelism



3D Model

Model Parallelism

FP-DEEP FRAMEWORK

Operator Graph Partitioning

FPDeep Framework uses four metrics to select parallelizing method:

1-) FLOP Utilization

- If DSP Slice (speed up the execution of signal processing functions) able to process a valid operation in every clock cycle. Maximum FLOPs can be achieved.
- Synchronization overhead effects Cumulative FLOPs.
- (CumulativeFLOPs/MaximumFLOPs) determine FLOPUtilization.

2-) Storage Requirement

Model Parameters and Temporal Activation Operations are storage factors.
Total storage factor determine whether all necessary data can be stored in OCM.

3-) Communication Footprint

FPGA Data Synchronization among co-workers.
It determine entire communication data throughput.

4-) Communication Average Bandwidth

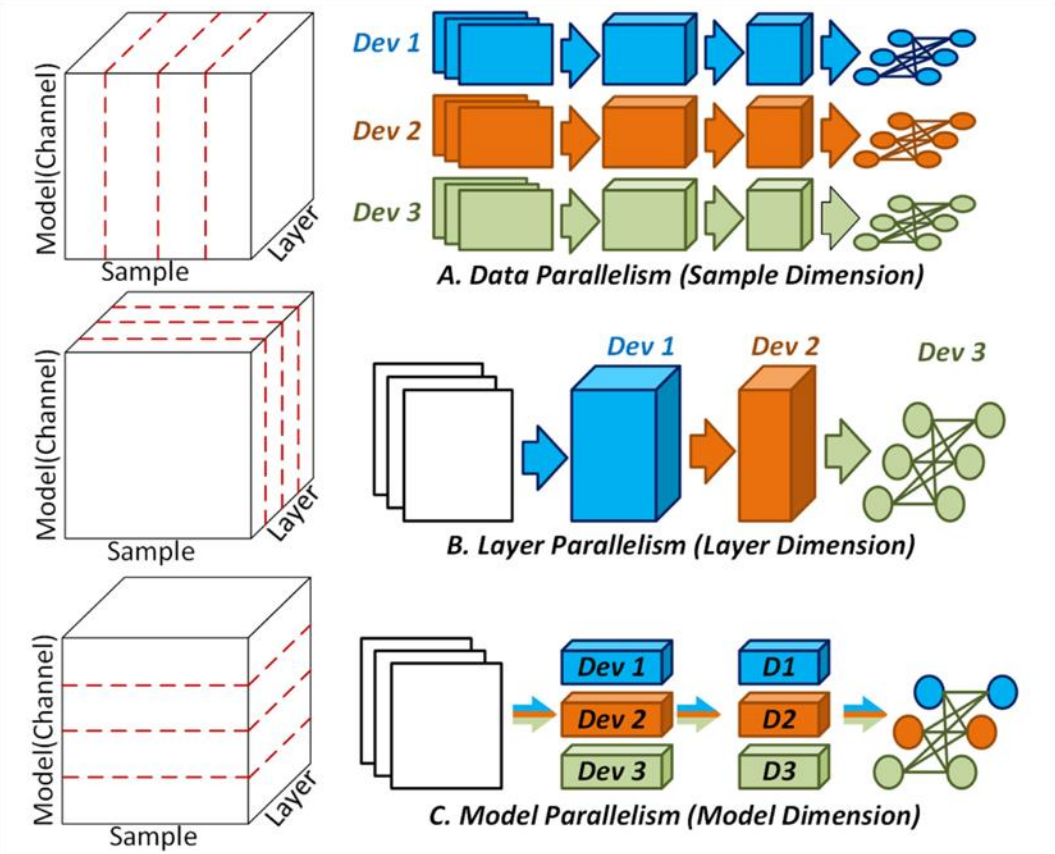
Data Throughput/Iteration Time

FP-DEEP FRAMEWORK

Operator Graph Partitioning

1-) Analysis of FLOP Utilization

- Layer Parallelism is best choice. (all FPGAs works in pipeline manner in FPDeep)
- Data Parallelism is second choice. (cause serious communication overhead)
- Model Parallelism is worst choice. (needs additional layer for synchronization)

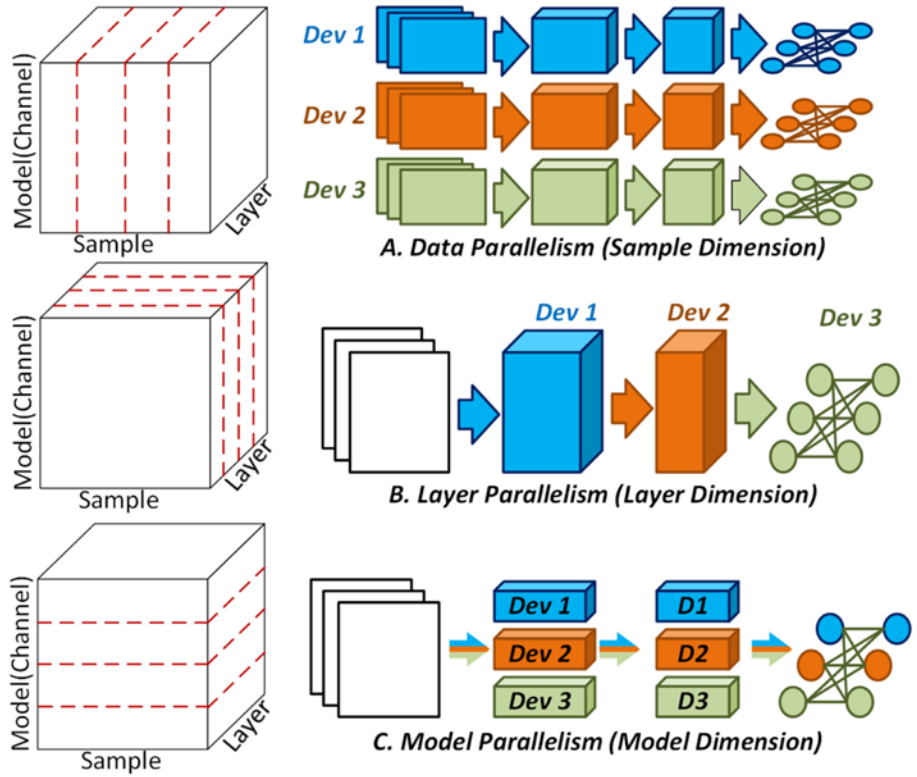


FP-DEEP FRAMEWORK

Operator Graph Partitioning

2-) Analysis of Storage Requirement

- Layer Parallelism is best choice.
  - Clusters does not need to store temporal activations.
- Data Parallelism and Model Parallelism is worst choices.
  - For DP and MP, each FPGA must keep its own copy of the DNN model's parameters. Also, all temporal activations must be maintained in local memory.

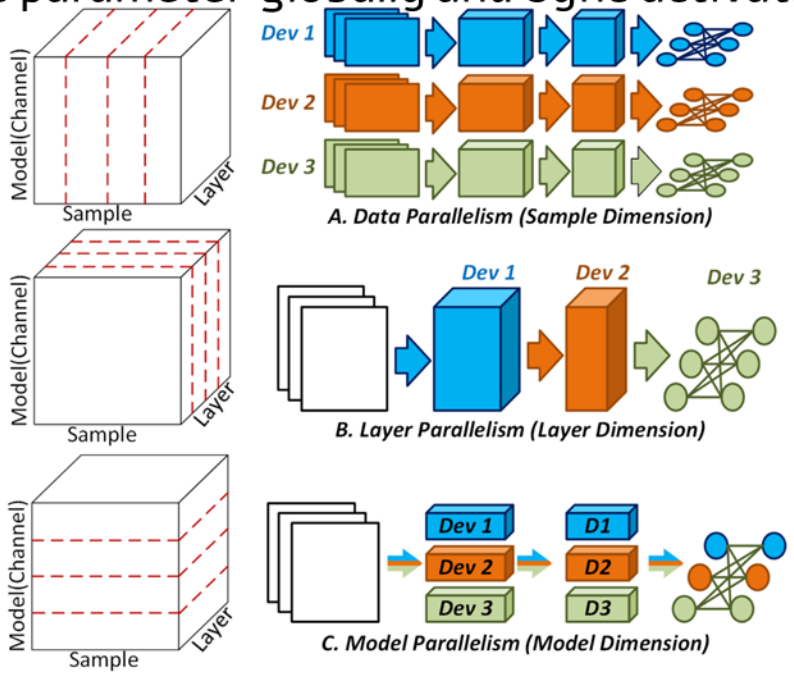


FP-DEEP FRAMEWORK

Operator Graph Partitioning

3-) Analysis of Communication Footprint (Data Sync)

- Data Parallelism is best choice.
  - Temporal Activation Operations stored locally.
  - Only DNN model parameters need dev-to-dev communication.
- Layer Parallelism is second choice.
  - Each device needs to sync activation operations.
- Model Parallelism is worst choice.
  - It needs both sync parameter globally and sync activations among channel. (High Bottleneck)

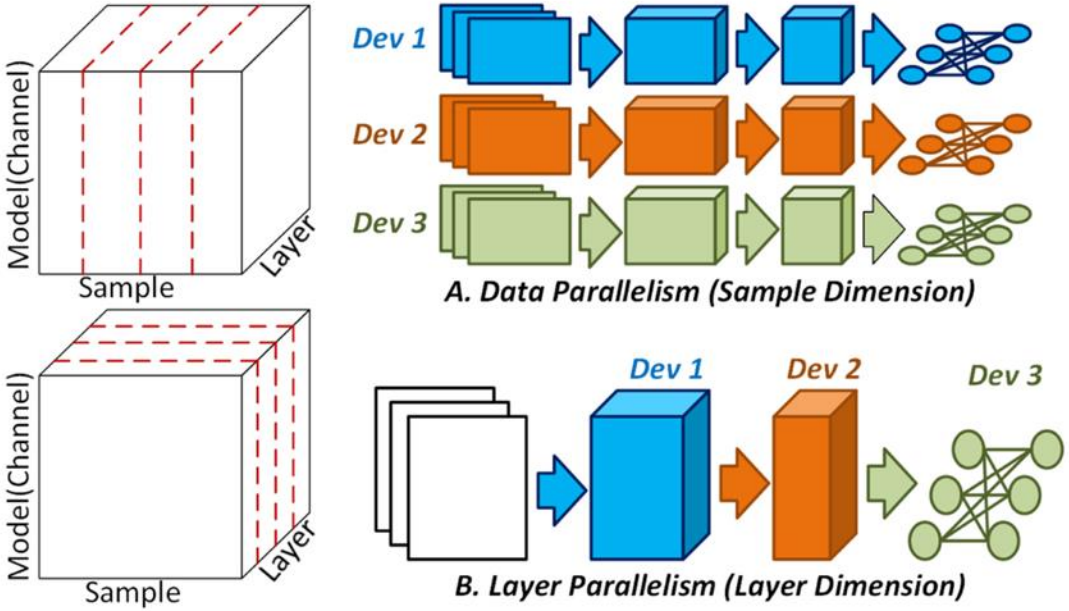


FP-DEEP FRAMEWORK

Operator Graph Partitioning

4-) Analysis of Communication Average Bandwidth

- Data Parallelism is best choice.
  - Bandwidth is lowest due to sync parameters after all workers finished.
- Layer Parallelism is worst choice.
  - Layers need to sync activation operations with adjacent node.

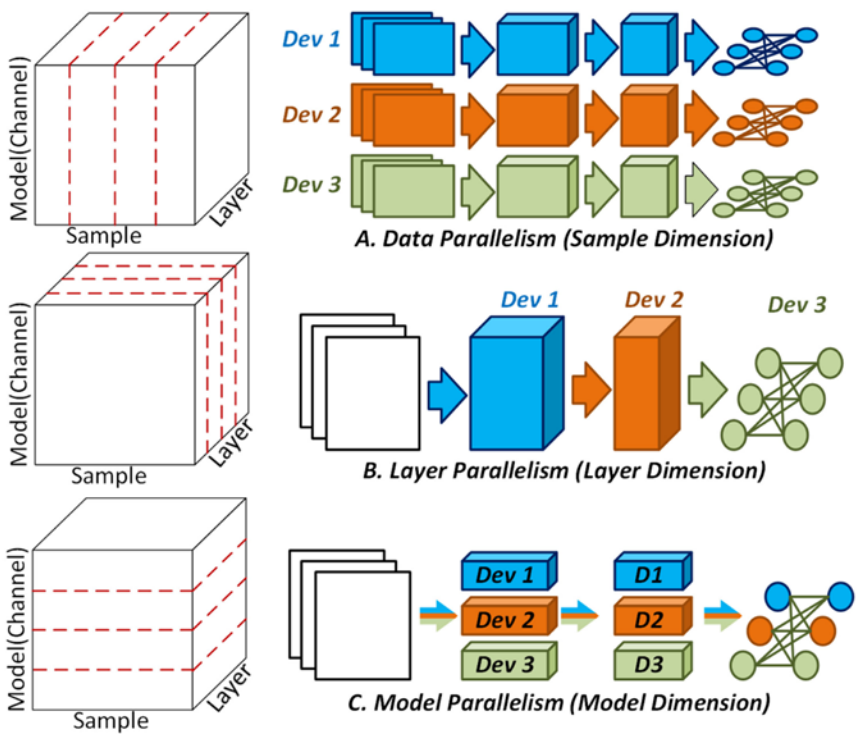


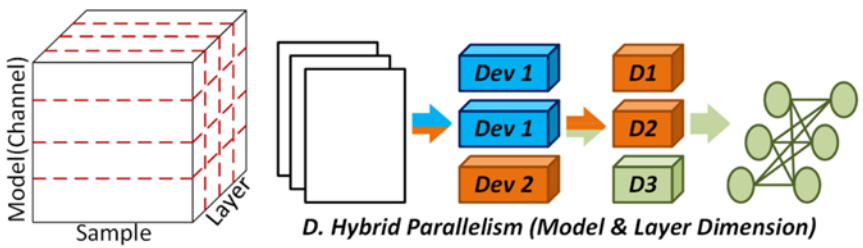
FP-DEEP FRAMEWORK

OGP Summary

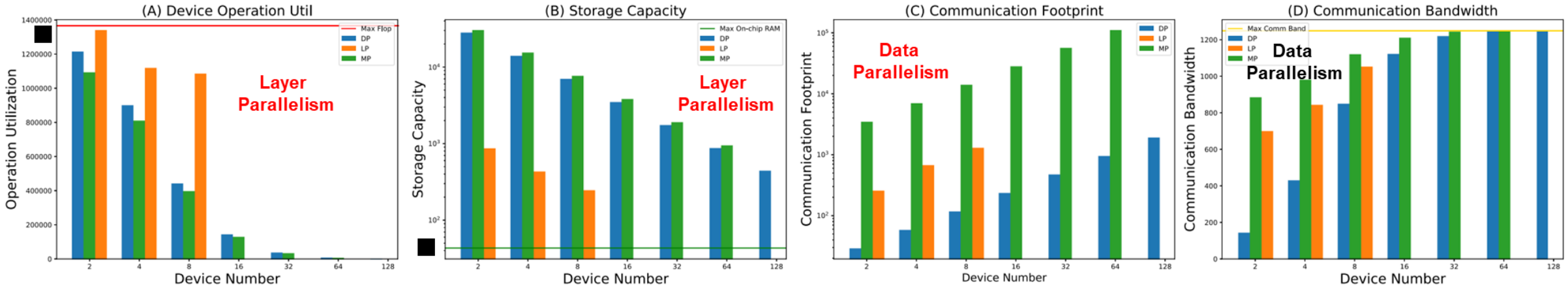
- Rather than DP, LP, or MP, FPDeep uses a hybrid parallel method.
- It works in a deeply pipelined manner with workload balanced among devices
  - This improves FLOP Utilization.
- The balanced allocation policy also reduces the Storage Requirement of the device-to-device activation buffer.

The only bottleneck for large scale clusters is the communication bandwidth.





FP-DEEP FRAMEWORK



Comparison Of Different Operator Graph Partition Methods With Defined Metrics

FPDeep Example of Selecting Parallelization Method

Evaluation of defined metrics for different parallelization methods and scales of FPGA clusters tested.

- ❑ VGG-16 (13 Convolution + 3 Fully Connected)
- ❑ Batch Size 128
- ❑ Device 128
- ❑ LP → Clusters with less than 16.
- ❑ MP → Minimum Channel Count 64

FP-DEEP FRAMEWORK

Resource Allocation

- Optimization Problem
  - Pipeline is constructed by Convolution Engines in FPGA
    - 2-D Systolic Arrays
    - LUT
    - FF
    - BRAM (Block-RAM)
    - DSP
  - Large FPGA Clusters
    - Goal is maximizing Throughput
  - Small FPGA Clusters
    - Goal is minimizing Storage Requirement

$$Buf^i = f_1(CE^i)$$

$$T = min\left(\frac{Op(G_i)}{Perf(CE^i)}\right)$$

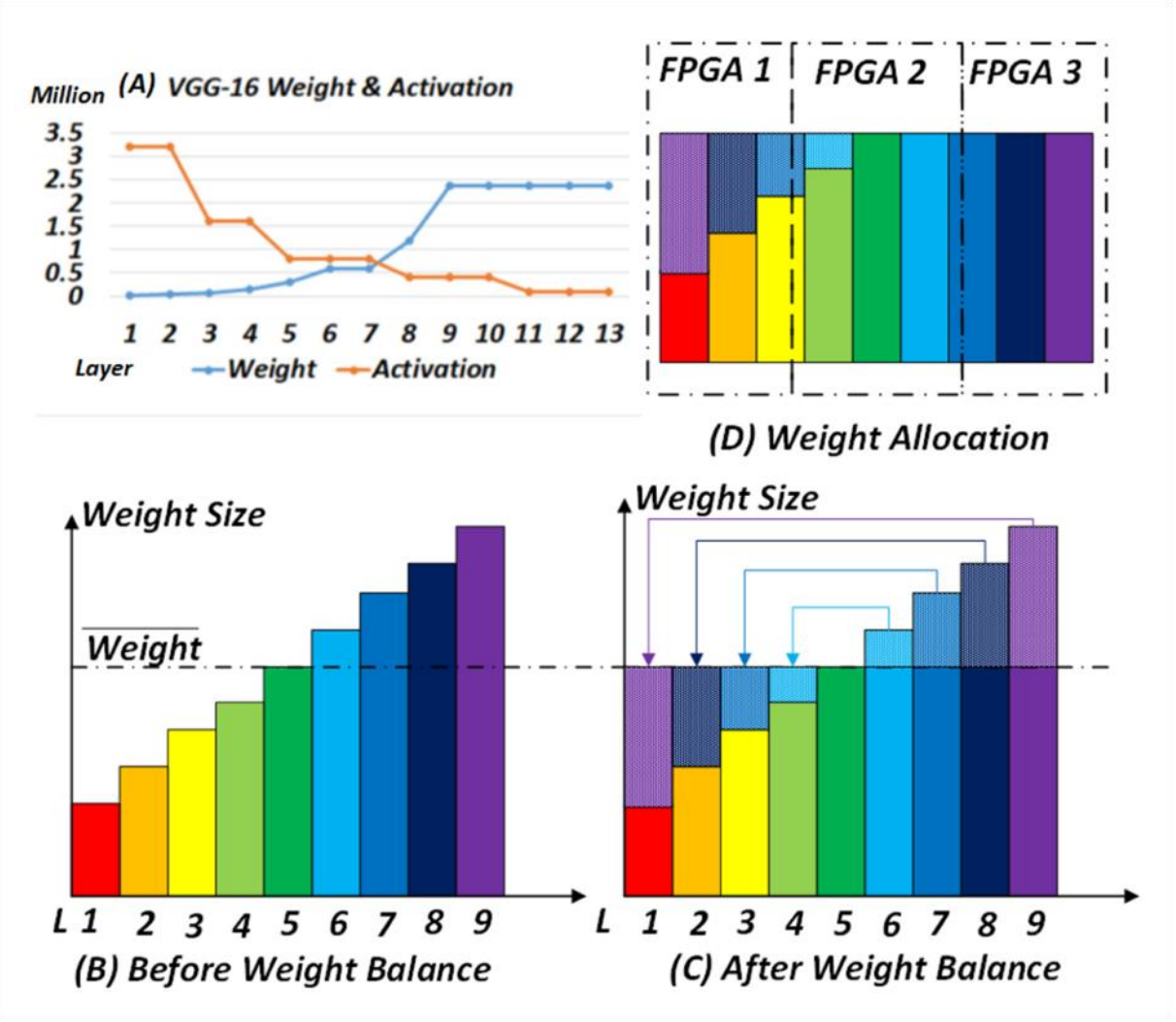
$$LUT^i = f_2(\alpha CE^i, \beta Buf^i, (1-\alpha)CE^i, (1-\beta)Buf^i) \leq LUT_{max}$$

$$FF^i = f_3(\alpha CE^i, \beta Buf^i, (1-\alpha)CE^i, (1-\beta)Buf^i) \leq FF_{max}$$

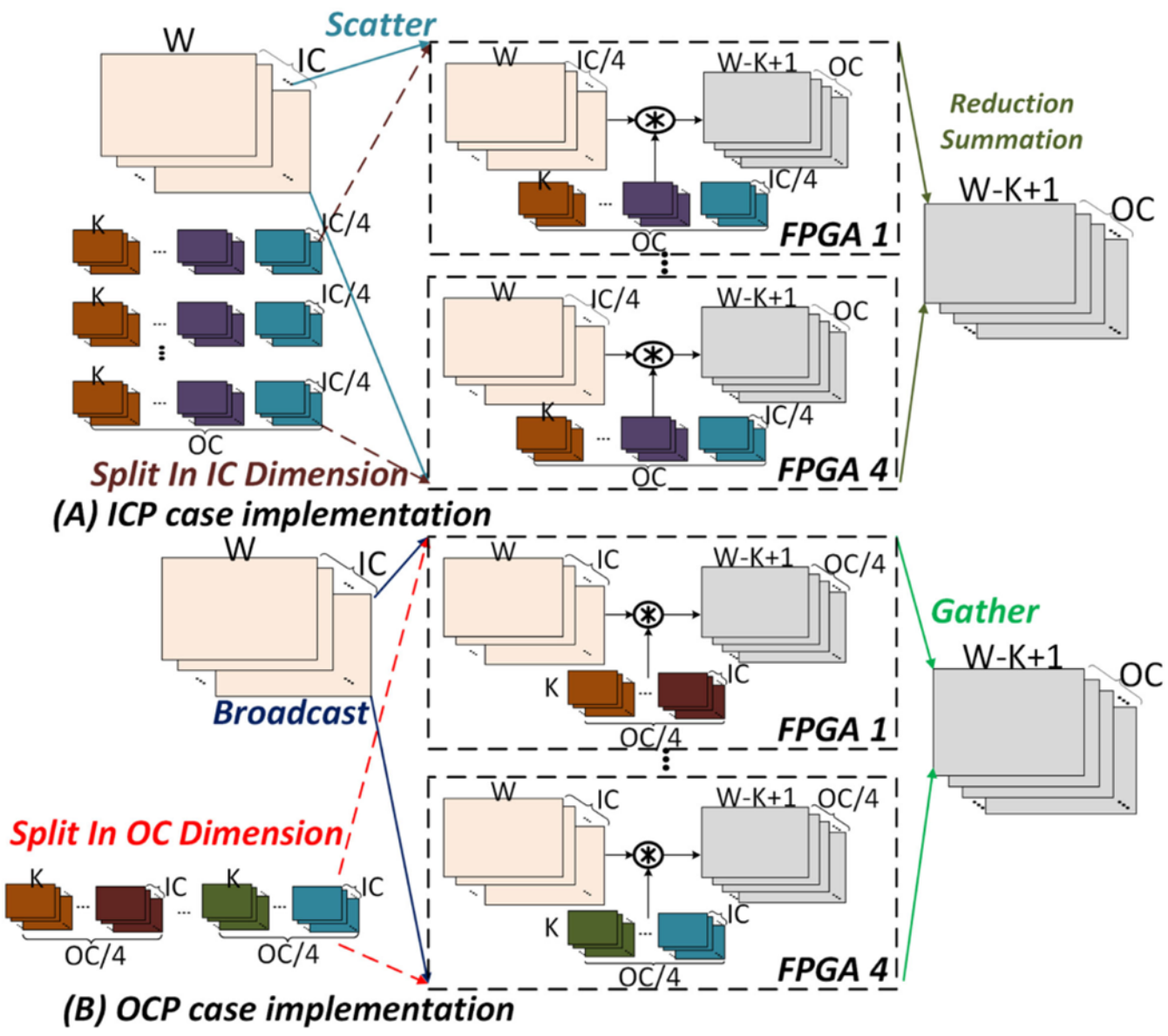
$$BRAM^i = f_4(\beta Buf^i) \leq BRAM_{max}$$

$$DSP^i = f_5(\alpha CE^i) \leq DSP_{max}$$

$$S = max(f_6(\beta Buf^i) + f_7(\beta Buf^i, (1-\beta)Buf^i))$$



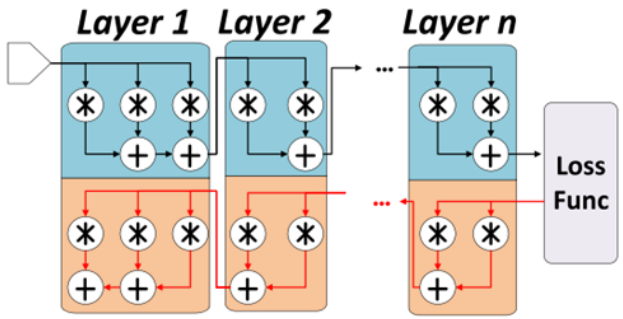
FP-DEEP IMPLEMENTATION



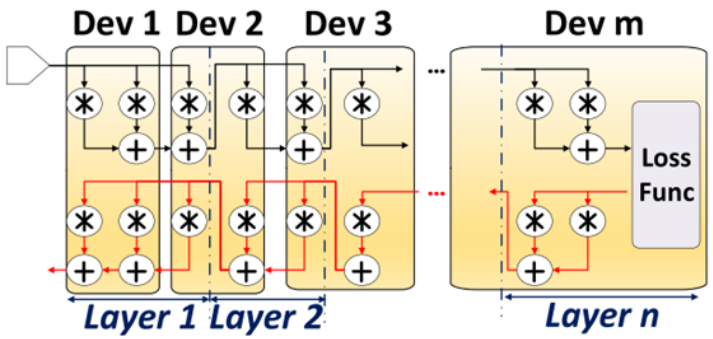
- Case 1 : ICP Implementation
  - Input Activation Channels
    - Weights
    - Kernels
      - Input Channel / 4
- Case 2 : OCP Implementation
  - Input Activation Channels
    - Weights
    - Kernels
      - Output Channel / 4

- In CNN training, all activations need to remain available while waiting for back propagation.
- For Case 1 : IC feature maps are **cached in every FPGA**. This **duplication leads** to additional **On-Chip Memory Overhead**.
- **This defect of OCP does not exist in ICP**.
- FP-Deep prefers to use ICP.
- OCP is only used when the number of the input channel is too small to provide sufficient parallelism.
- For example, the first layer of AlexNet only has 3 channels of input features but 96 channels of output features so OCP is used.

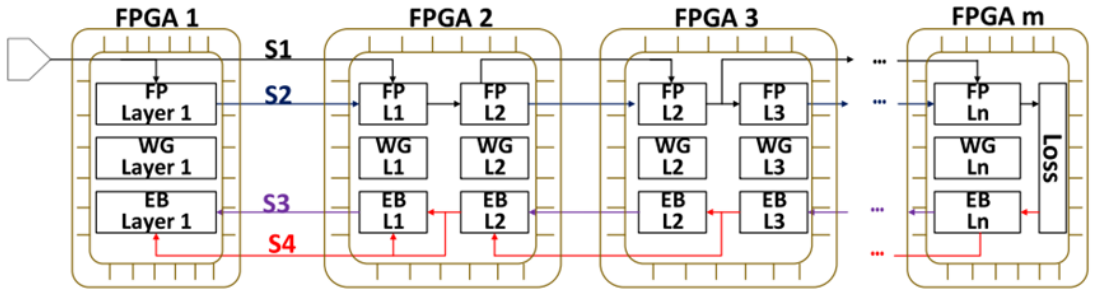
FP-DEEP IMPLEMENTATION



(A) Change the reduction operations into pipeline

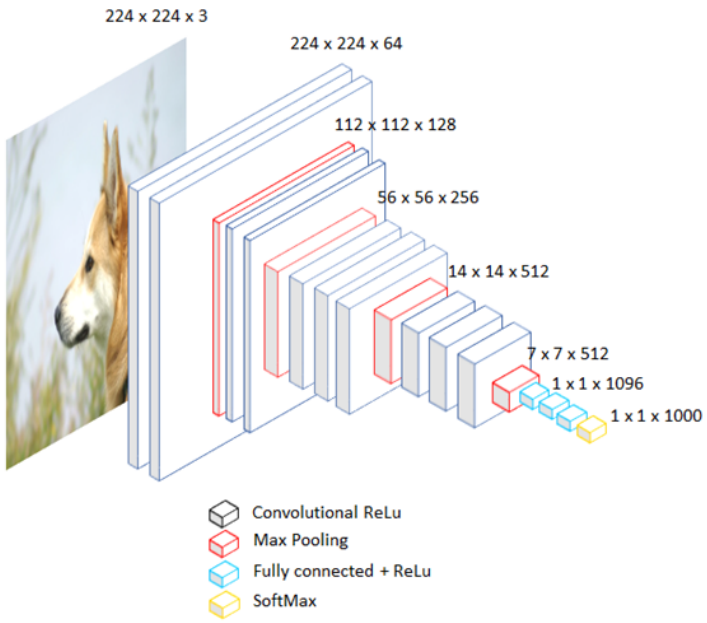


(B) Map the CNN training progress to multiple FPGA devices

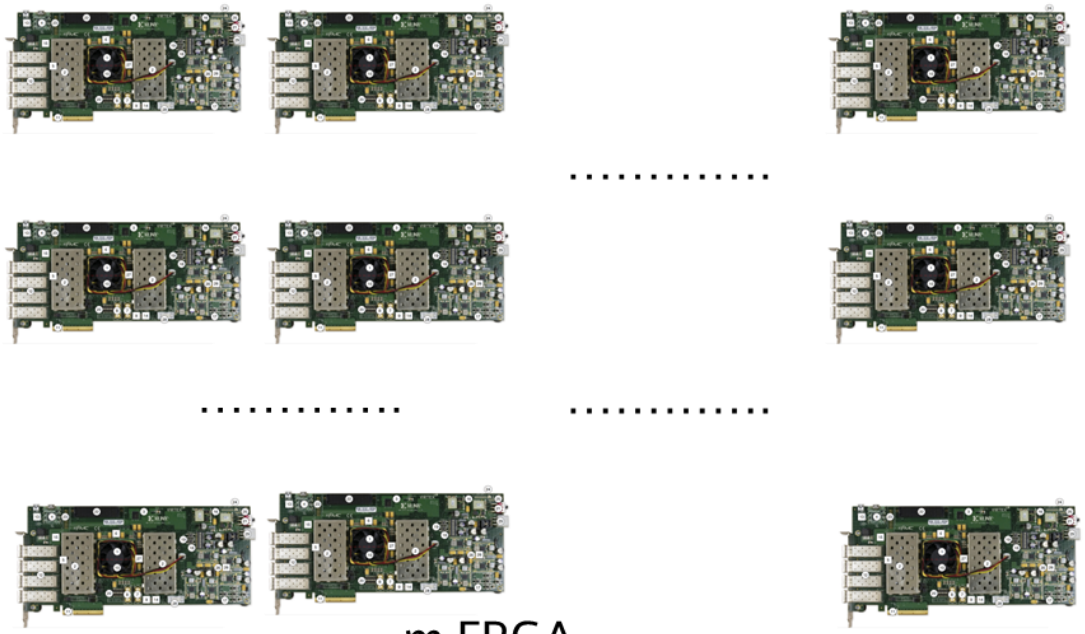


S1: FP activation; S2: FP partial activation; S3: BP partial error ; S4: BP error

(C) The data stream communication of the cluster and topology choice

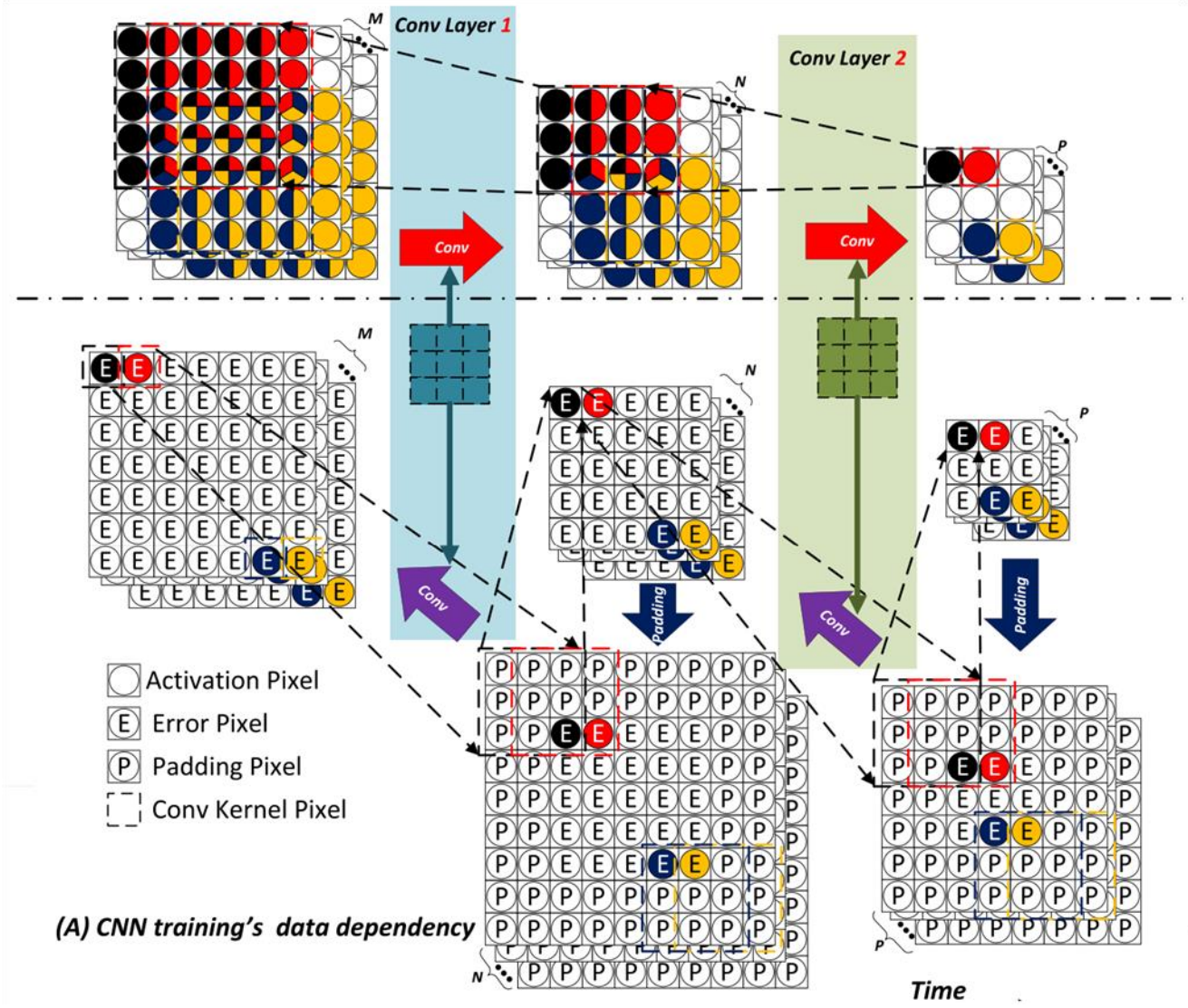


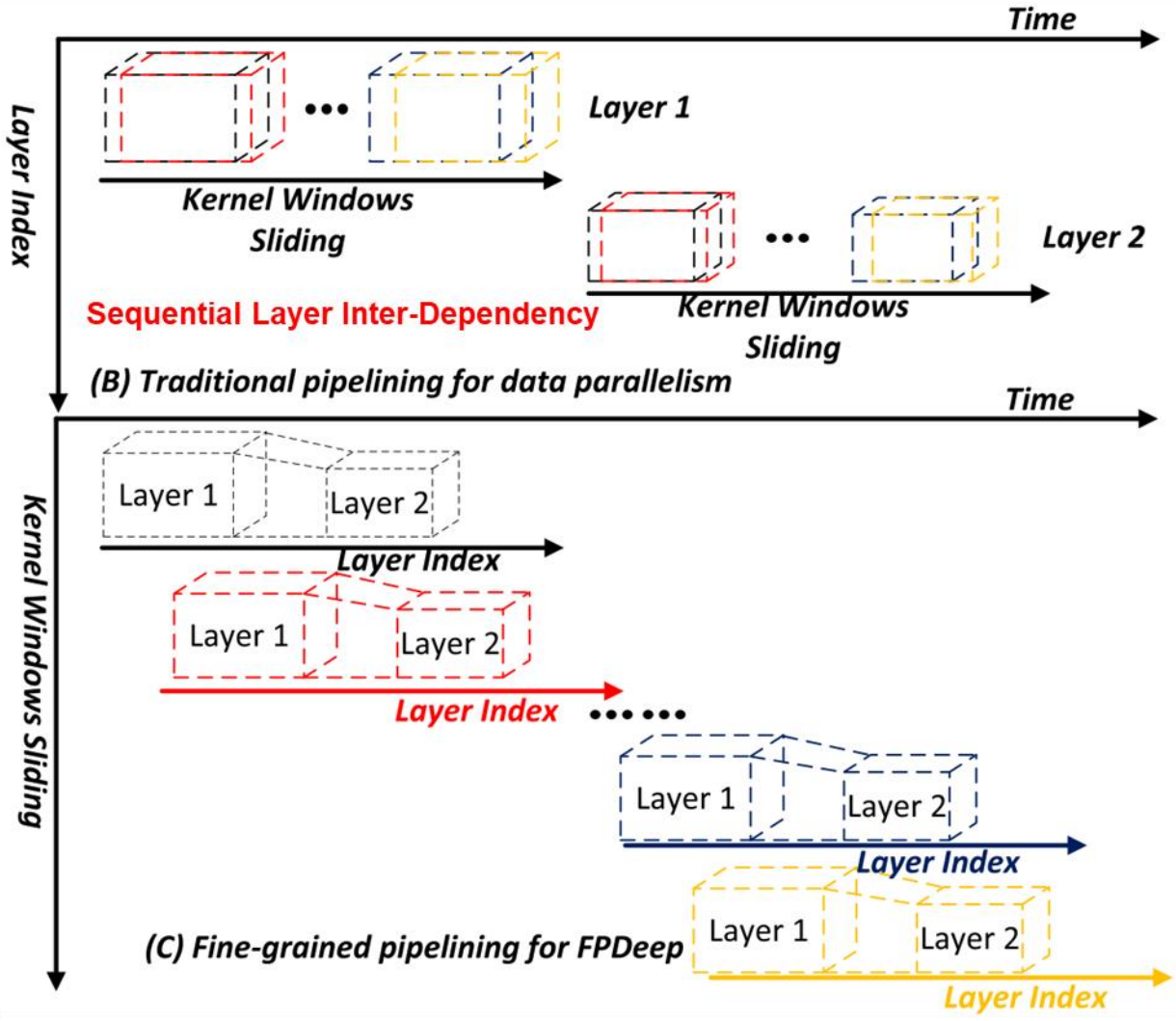
n Layer CNN



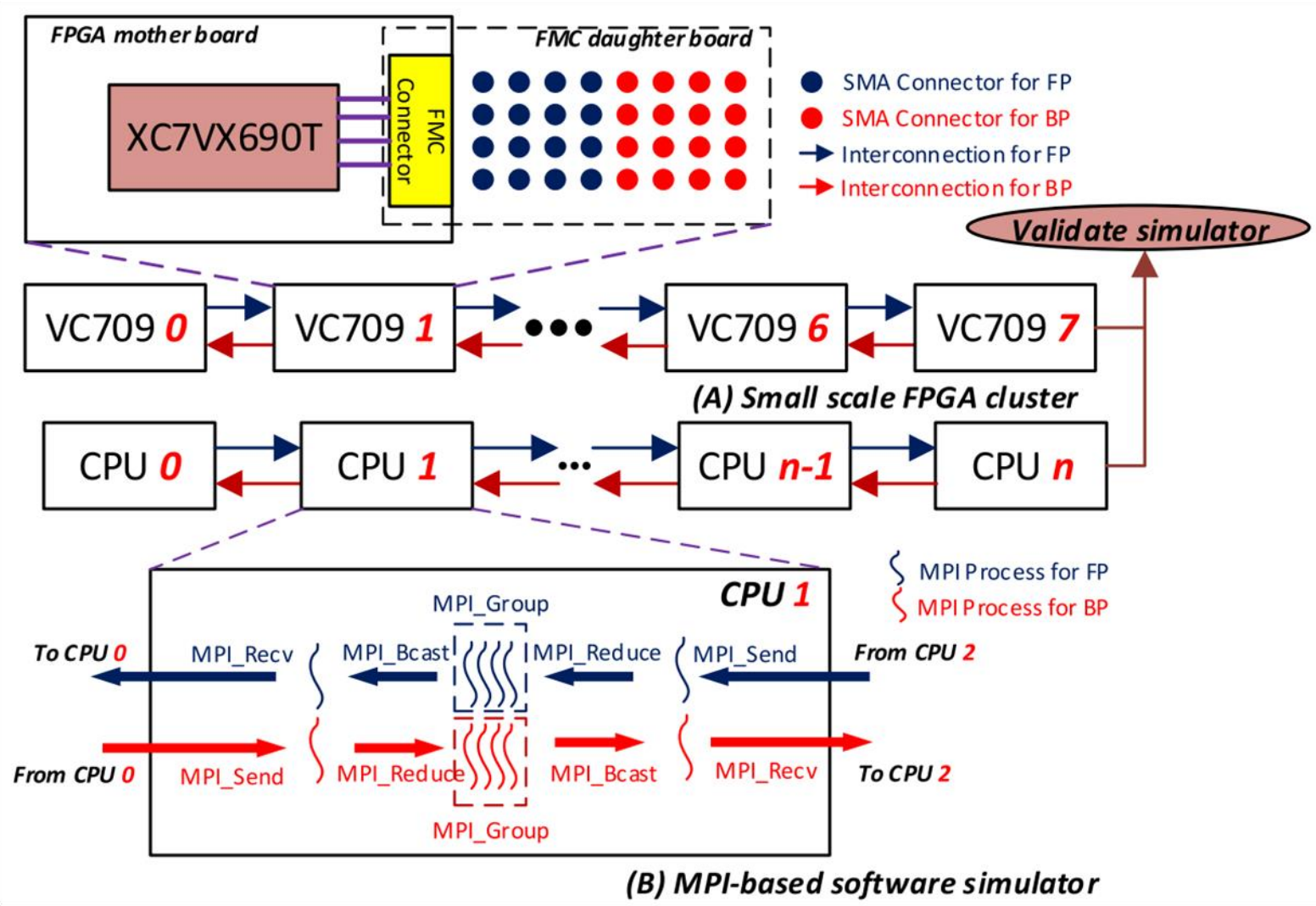
m FPGA

FP-DEEP IMPLEMENTATION



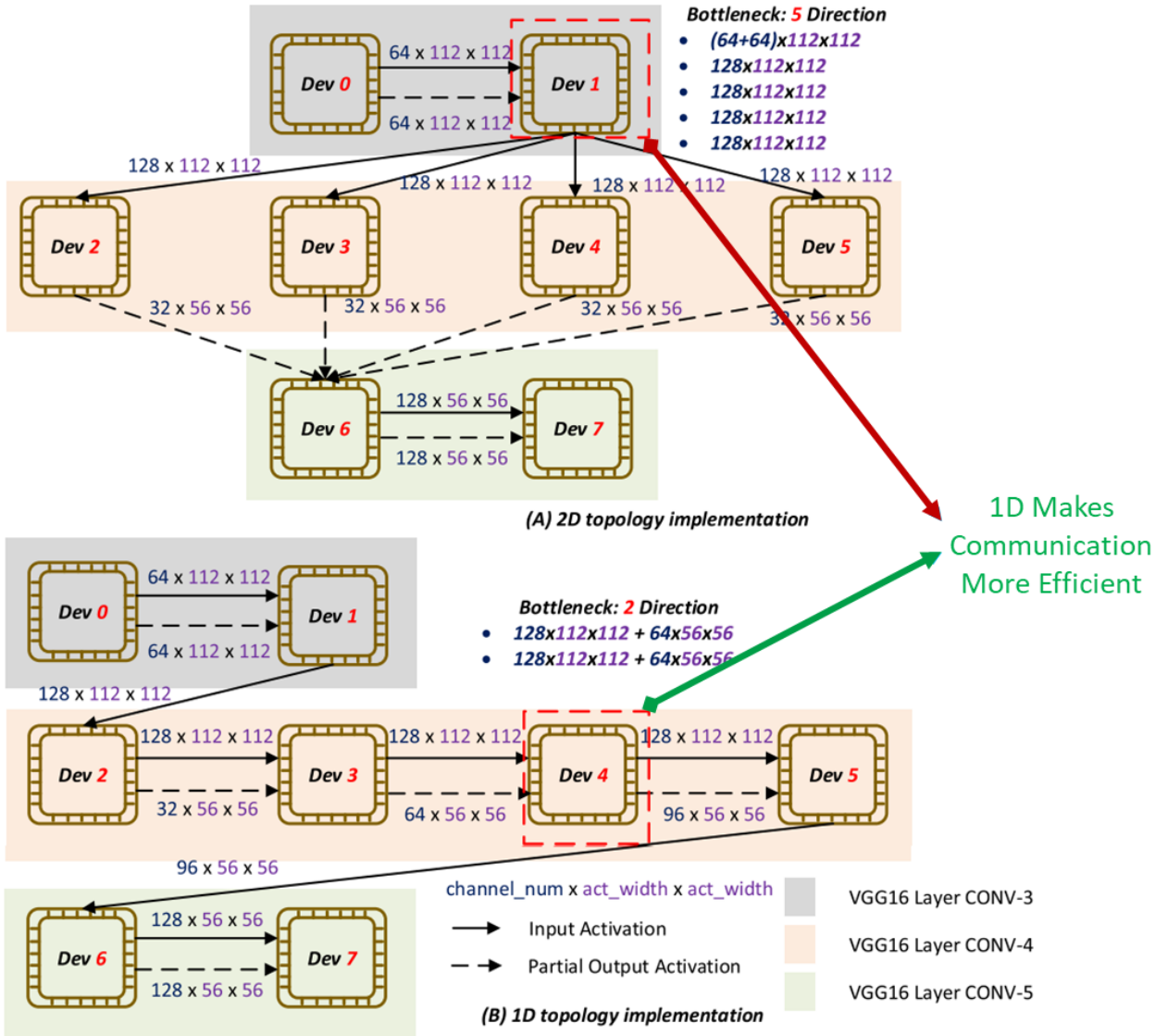


FP-DEEP IMPLEMENTATION

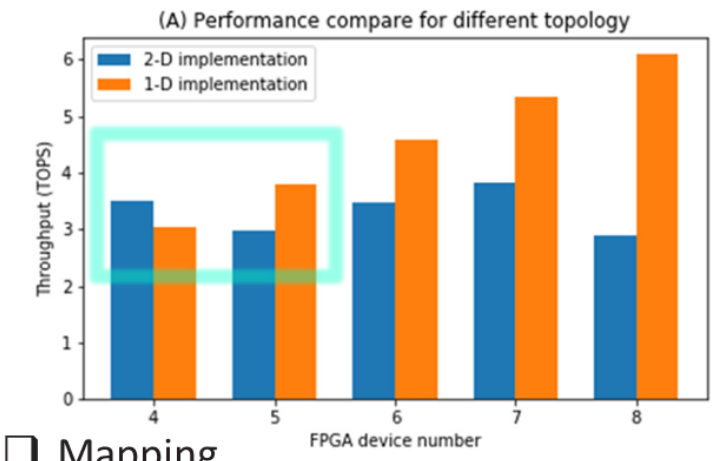


The FPDeep software simulator is compiled with **gcc 7.1** and **OpenMPI 2.1.1.**
It is currently using for large-scale network test such as AlexNET, VGG-16, VGG-19.

FP-DEEP RESULTS

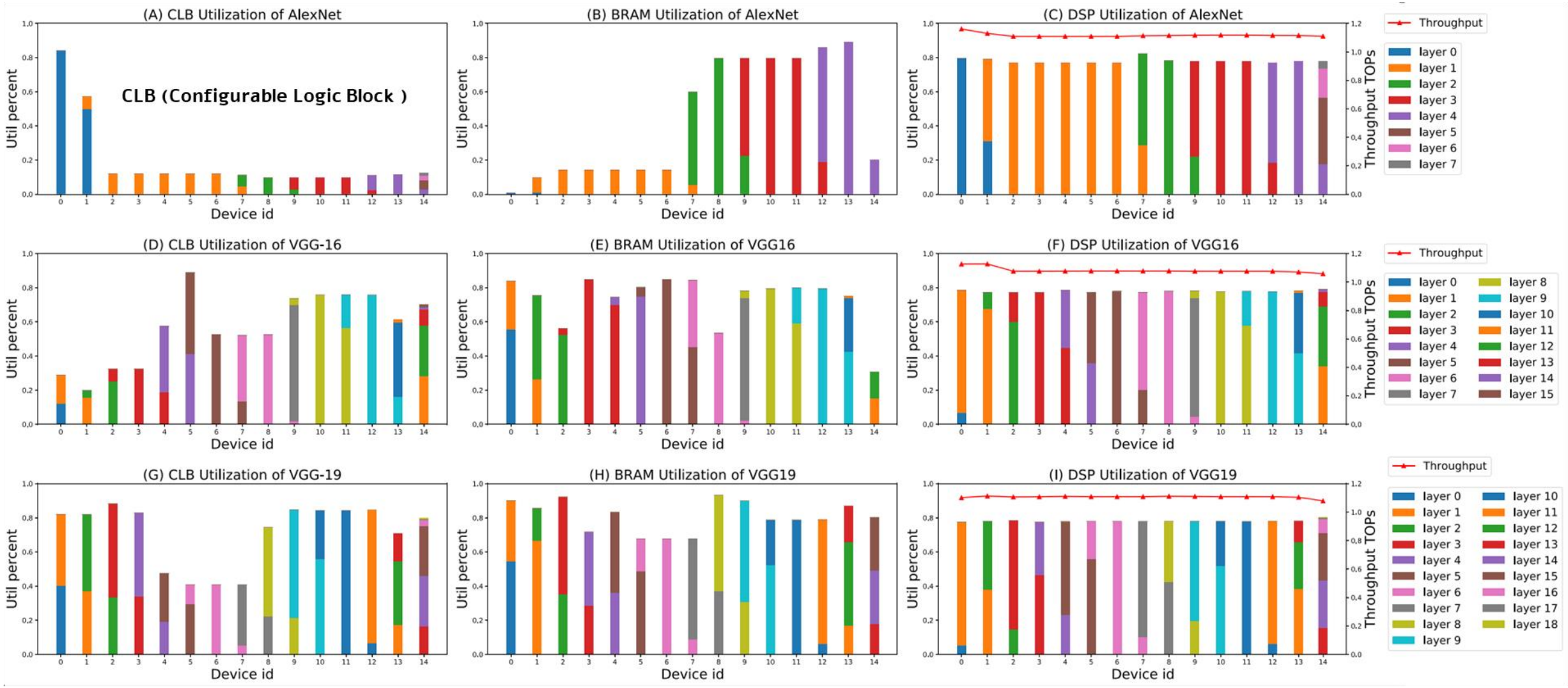


1D Makes Communication More Efficient



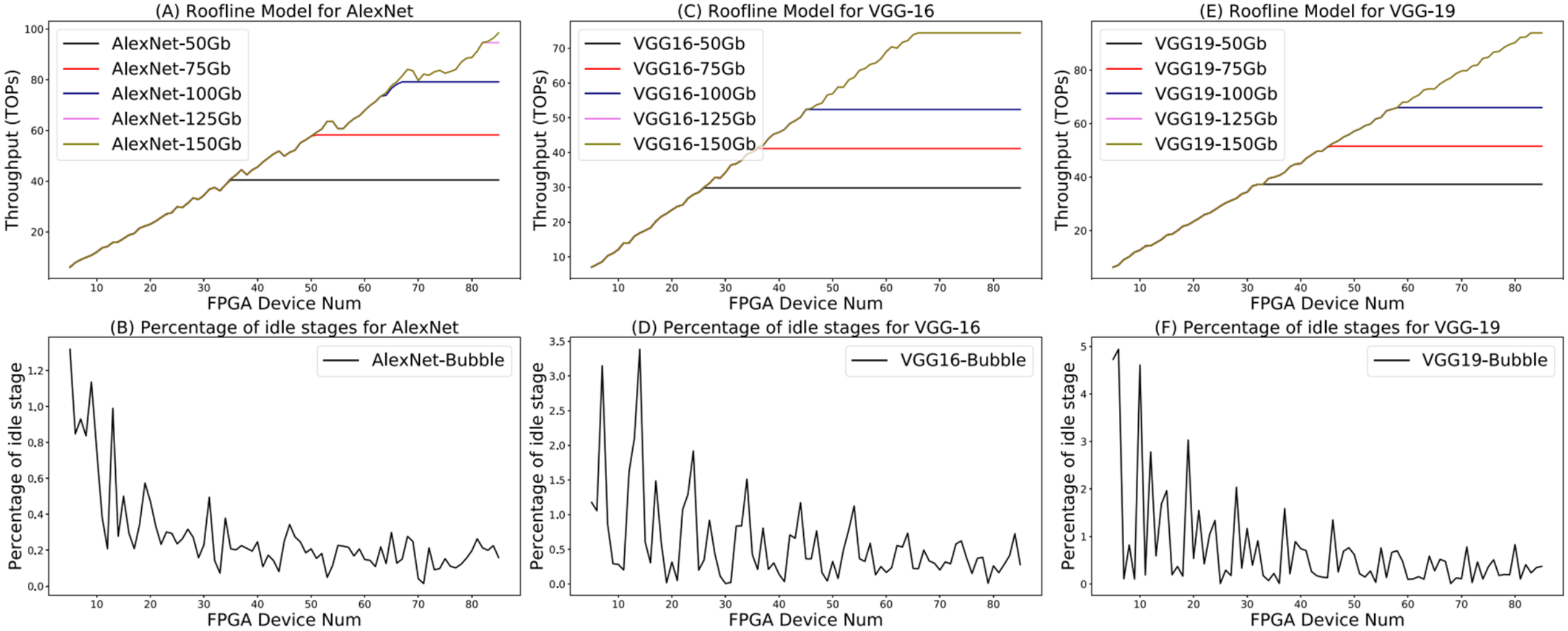
- ❑ Mapping
  - VGG-16's [Convolution 3 → 5]
  - VC709 FPGA
- ❑ FPDeep's
  - Graph Partition Method
  - FPGA Resource Allocation Policy
- ❑ Performance Comparison
  - 2D : Overhead and Data Movement
    - **1 EVK with n FPGA** : 2D Implementation
    - Intra-Node Connectivity is Limited
  - 1D : Simplicity
    - **4 EVK with 1 FPGA** : 1D Implementation
    - Inter-Node Connectivity is Limited

FP-DEEP RESULTS

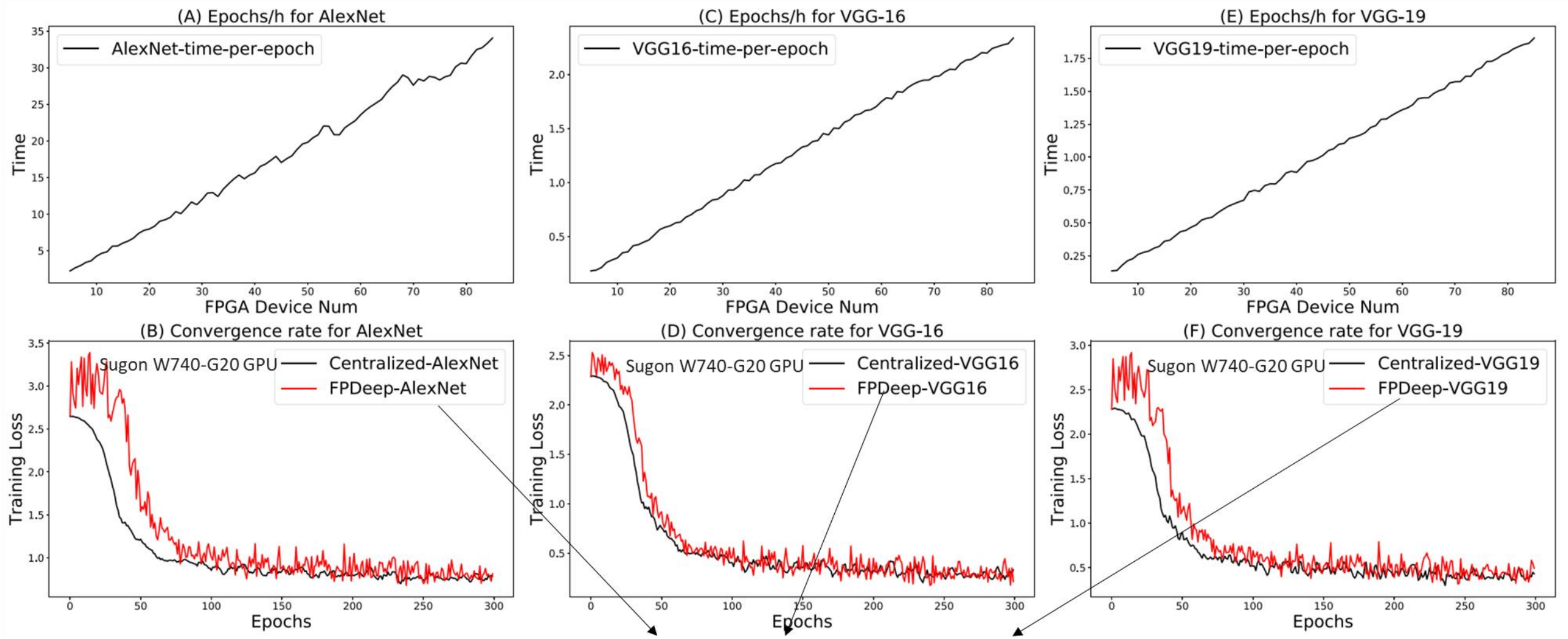


- ❑ No requirement for parameter balancing to achieve the best performance so → CLB and BRAM utilization unbalanced in AlexNet
- ❑ DSP slices utilization is roughly 80% (each FPGA throughput about 1 Gb/s)
- ❑ On-chip BRAM is only used for Convolutional Layers (utilization is roughly 80%)

FP-DEEP RESULTS



FP-DEEP RESULTS



Sugon CX50-G20 CPU & Intel Xeon E5-2680 v3 CPU Cluster Implementation

Produced Model Error → Minimum (Convergence)

FP-DEEP RESULTS

Cluster Level Experimental Results with GPUs and CPUs

	CPU [43]	GPU [43]	GPU		FPGA [43]		FPDeep		
Device	AMD A10	Titan X	Tesla K80		Xilinx XC7VX690T				
CNN Model	AlexNet	AlexNet	AlexNet [23]	VGG-16 [24]	AlexNet	VGG-16	AlexNet	VGG-16	VGG-19
Config	1 CPU	1 GPU	1 GPU	1 GPU	4 FPGAs	1 FPGA	15 FPGAs	15 FPGAs	15 FPGAs
Perf (GOPS)	34.23	1385	2330	2018	207 (Per FPGA)	290 (Per FPGA)	1157(Per FPGA)	1197(Per FPGA)	1220(Per FPGA)
Training time (H)/ Epoch	NA	NA	NA	NA	NA	NA	0.17	2.19	2.76
Power efficiency (GOPS/J)	0.39	4.22	7.87	6.86	6.55	8.28	37.09	37.88	38.13

All (CPU/GPU/FPGA) implementations use single precision floating point. [43], [25], and [26] do not give experiment results of training time per epoch.

FPDeep provides **5x** higher **performance** than previous FPGA work and comparable to the Titan X GPU.
FPDeep provides **8-13x** better **energy efficiency** than the Tesla K80 : Titan X
FPDeep provides **5.6x** better **energy efficiency** than the previous FPGA work.
FPDeep provides **5.7x** better **energy efficiency** than K80.

CONCLUSION

❑ Main Contributions of Study

- First
  - The **breaking down** the well-known **scalability wall** of CNN training. (Up to 128 Devices)
  - It **demonstrates FPGA Clusters** to be a **competitive technology** for **CNN** Training
- Second
  - A Novel Framework for **mapping CNN** Training logic to **Distributed FPGA Clusters**
    - No waiting for backpropagation
    - Reducing Storage Demand
    - Only OCM is required for Convolution Layers
- Third
  - Fine-grained **pipeline design** that **minimizes** the **time** while **waiting** for back-propagation. **CNN features remains available** and parallelism can be possible **without breaking** the architecture **integrity**
- Fourth
  - Fine-grained **partitioning** and **mapping methodologies**
    - It provides almost perfect **workload distribution** and **weight balancing among FPGAs**
      - Multiple FPGAs Cooperatively → Same Layer
      - Multiple Layers Mapped → Same FPGA
- Fifth
  - **RTL code generator** that **automatically creates** RTL **implementations** based on the **mapping scheme** generated **by FPDeep**

CONCLUSION

❑ Main Benefits of Study

- First
  - → No performance loss by **batch size growths**
- Second
  - → The **computation** and **storage requirements balanced** among node
  - → A Novel Workload and Weight Partitioning Schemes are proposed
- Third
  - → **Fine-Grained** Pipeline architecture leads **high parallelism** and **utilization**
  - → **Fine-Grained** Pipeline reduces **time for caching CNN features** during back-propagation
  - → Storage demands dramatically reduced for Convolutional LayerS
    - ○ Only On-Chip Memory !!
- Fourth
  - → **Simplest Topology** that is preferred for **interconnecting** the **FPGAs for CNN**
  - → It enables **widespread applications** for Deep Learning (Aviation, Space, Military, Wearable Technologies and Health Sciences)
- Fifth
  - → **High Scalability** with large number of FPGA. The **only limiting factor** is **FPGA-to-FPGA bandwidth**

THANK YOU