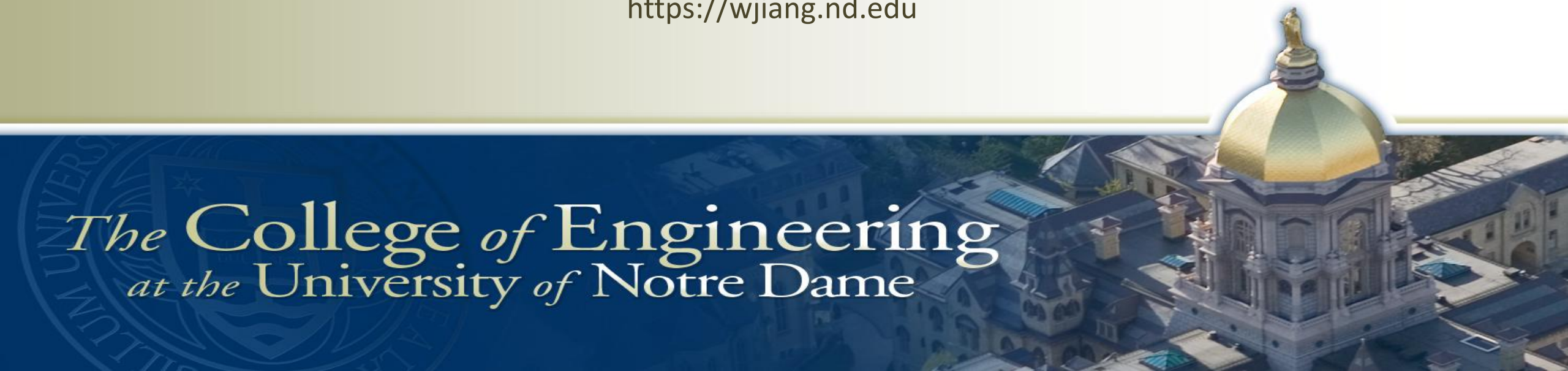
Hardware/Software Co-Exploration of Neural Architectures on FPGAs

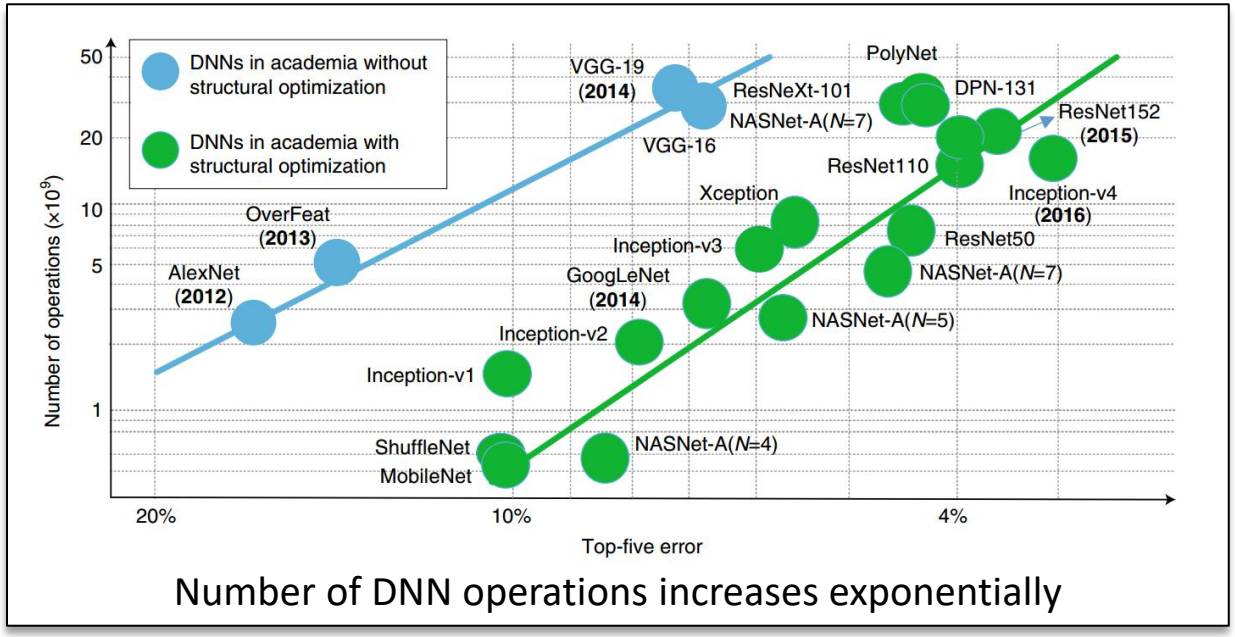
Weiwen Jiang
Postdoc Scholar
wjiang2@nd.edu
https://wjiang.nd.edu



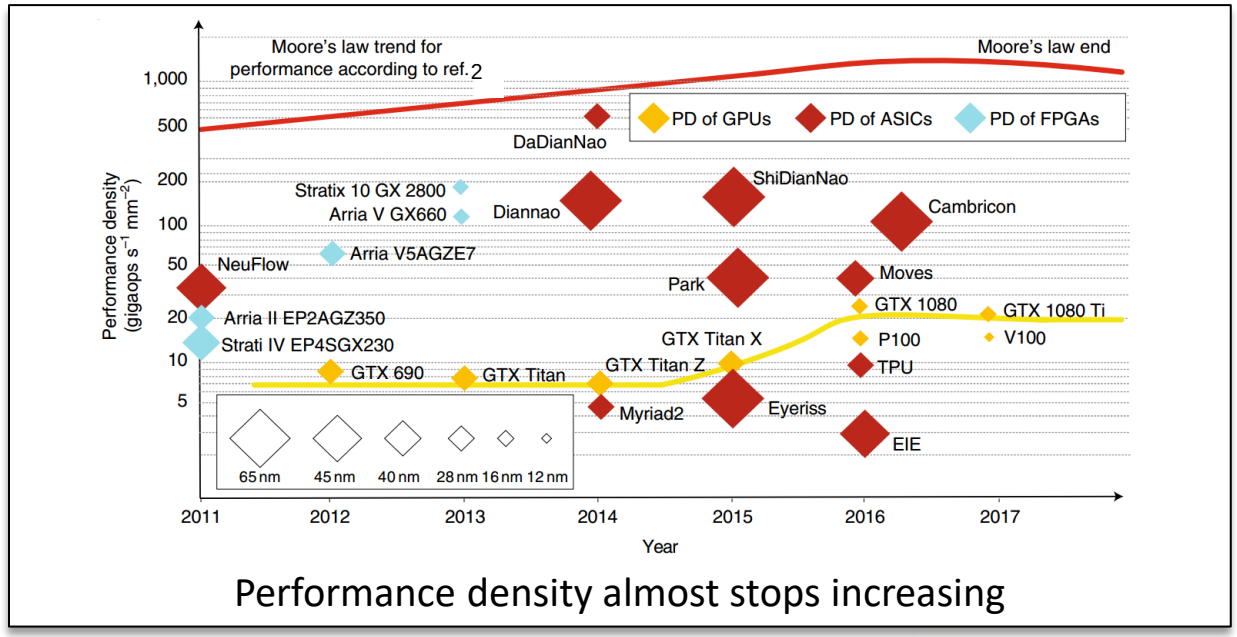
Outline

- **Introduction**
- **XFER: Achieving Super-Linear Speedup across Multi-FPGA**
  - CODES+ISSS'19 (Best Paper Finalist)
- **Co-Exploration of Neural Architecture and FPGA Implementation**
  - DAC'19 (Best Paper Finalist)
- **Other Research Directions**
- **Conclusion**

Computation Demand > Supply



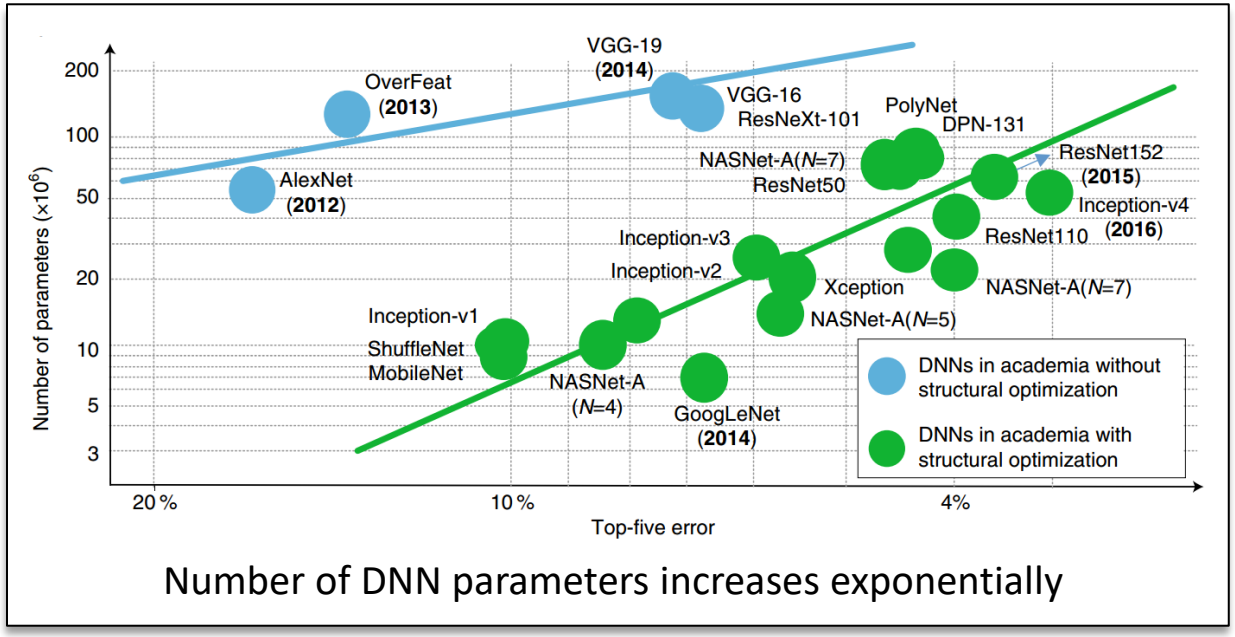
Number of DNN operations increases exponentially



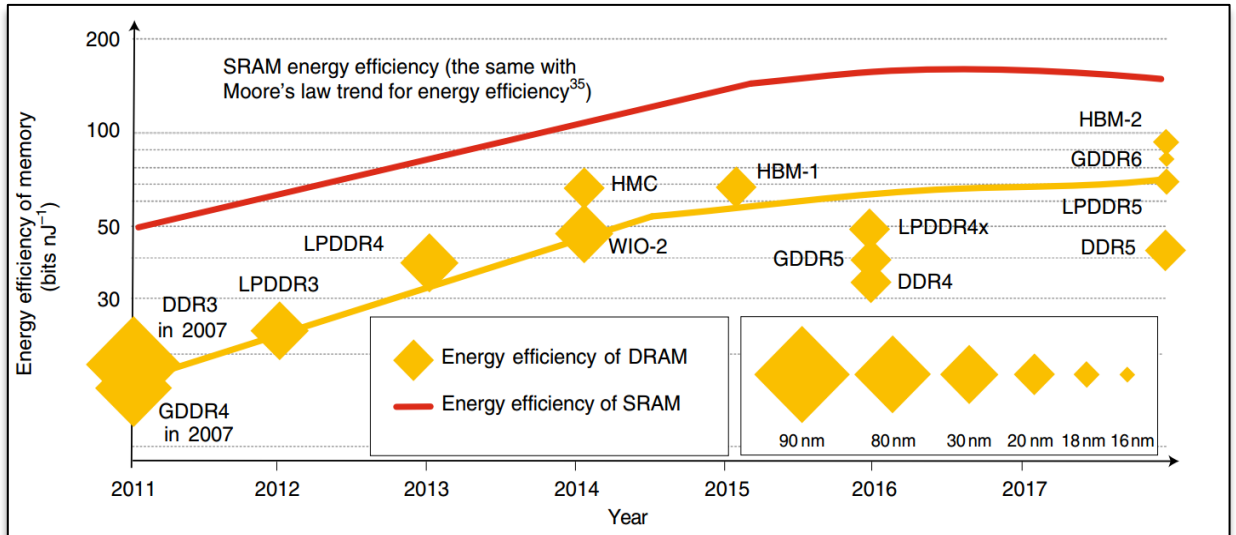
Performance density almost stops increasing

[ref.1] Xu, Xiaowei, et al. "Scaling for edge inference of deep neural networks." Nature Electronics 1.4 (2018): 216.
[ref.2] Sutter, H. The free lunch is over: A fundamental turn toward concurrency in software. Dr Dobb's J. 30, 202–210.

Storage Demand > Supply



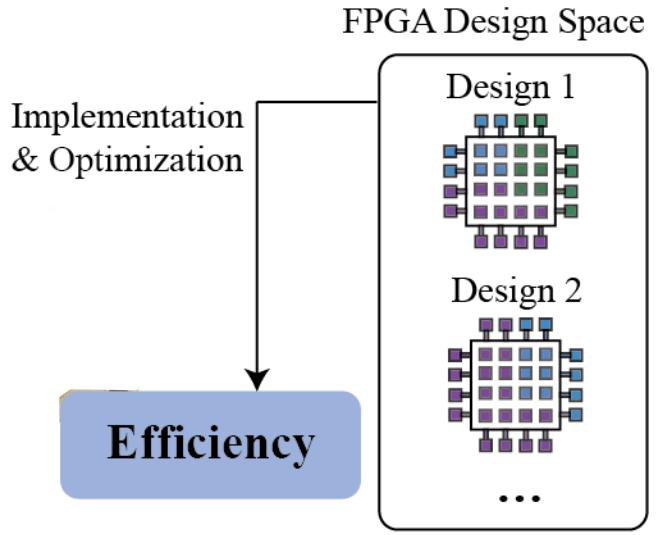
Number of DNN parameters increases exponentially



Energy efficiency of memory almost stops increasing

[ref.1] Xu, Xiaowei, et al. "Scaling for edge inference of deep neural networks." Nature Electronics 1.4 (2018): 216.
[ref.2] Sutter, H. The free lunch is over: A fundamental turn toward concurrency in software. Dr Dobb's J. 30, 202–210.

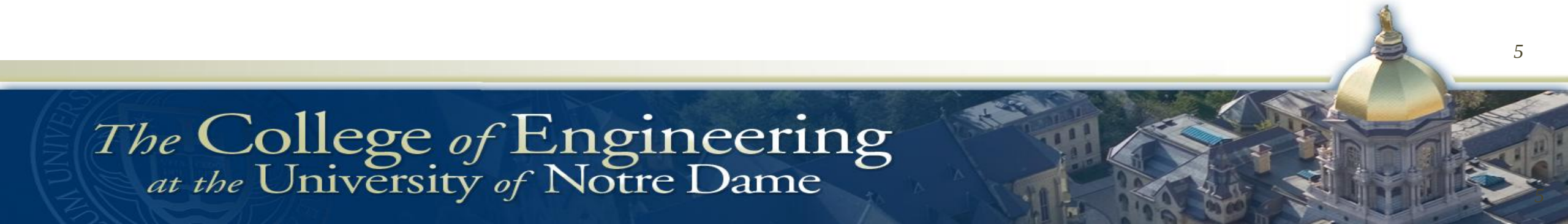
Efforts on Both Software and Hardware Sides are Required



Neural Architecture Implementation on FPGAs

Outline

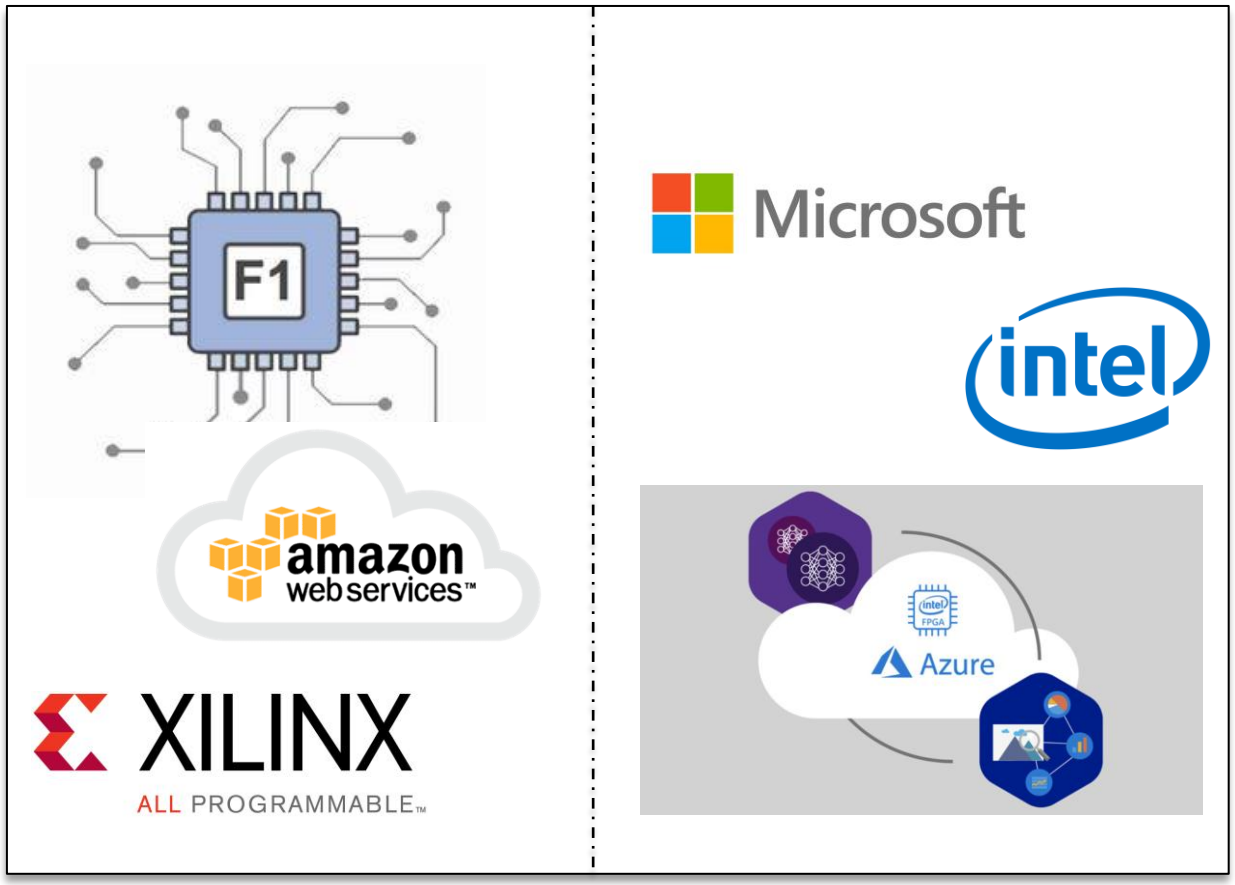
- Introduction
- **XFER: Achieving Super-Linear Speedup across Multi-FPGA**
  - CODES+ISSS'19 (Best Paper Finalist)
- Co-Exploration of Neural Architecture and FPGA Implementation
  - DAC'19 (Best Paper Finalist)
- Other Research Directions
- Conclusion



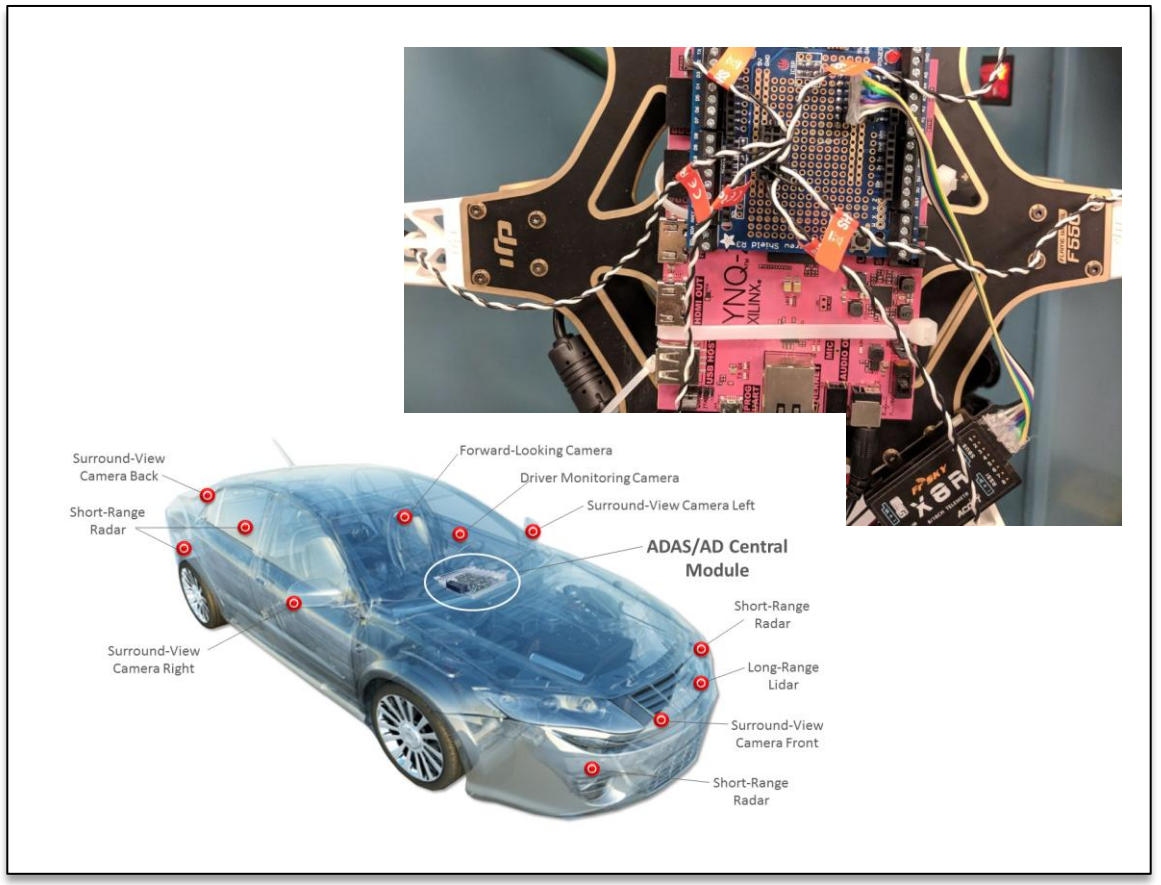
Background

FPGAs in DNN Applications

FPGA in Cloud Computing



FPGA in Edge Computing

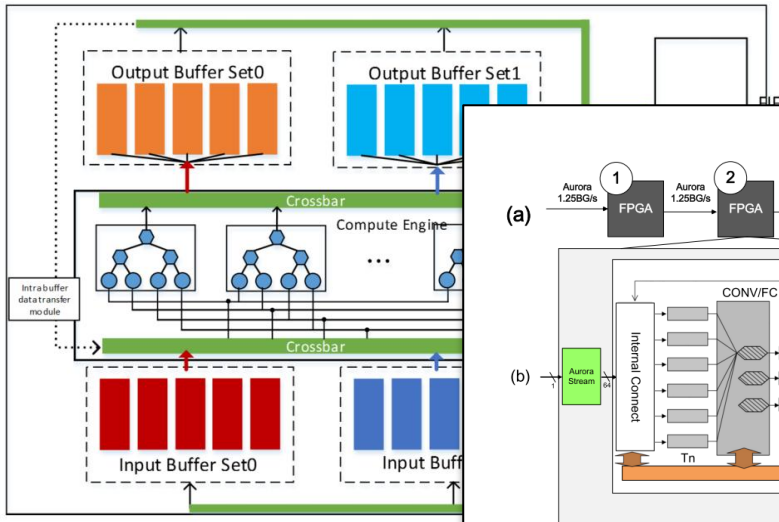


[ref] Where Do FPGAs Stand in Auto IC Race? https://www.eetimes.com/document.asp?doc_id=1333419#

[ref] PYNQ on UAV http://brennancain.com/pynqcopter-an-open-source-fpga-overlay-for-uavs/

Response to Limited Resource: Pipelining Multi-FPGA

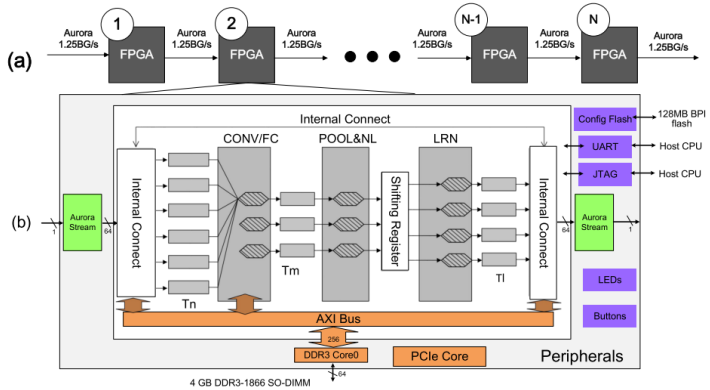
Single-FPGA

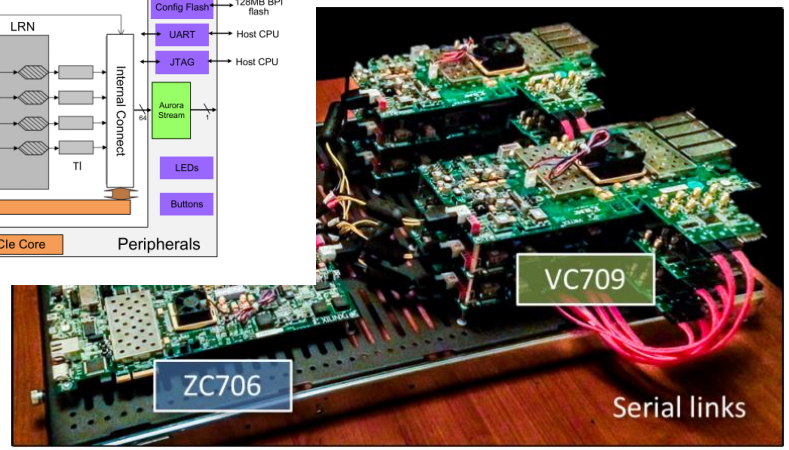


Limited Resource → Li

[Ref] C. Zhang et al. Optimizing fpga-
Proc. of FPGA, pages 161–170. ACM

Pipelined FPGAs



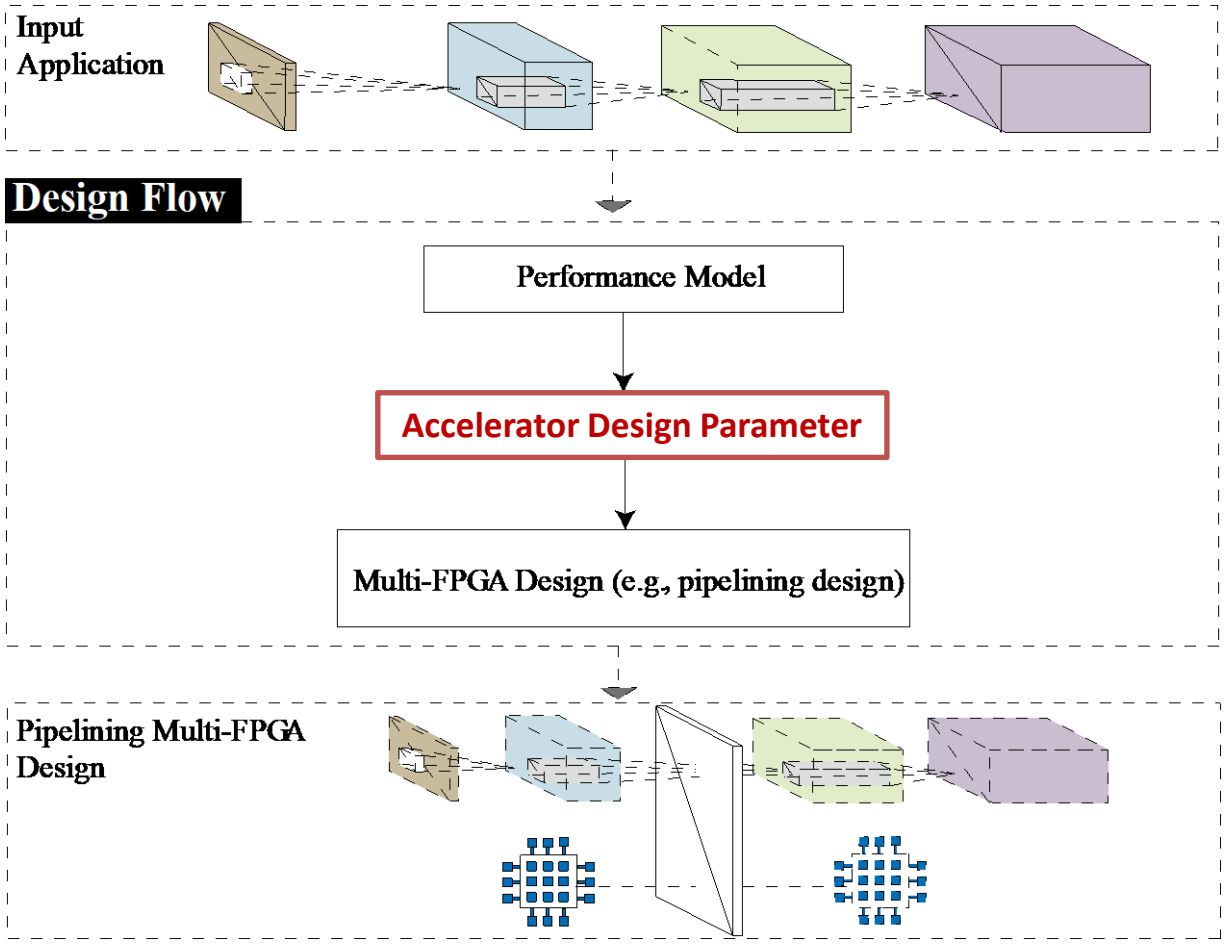


Pipeline → High Throughput

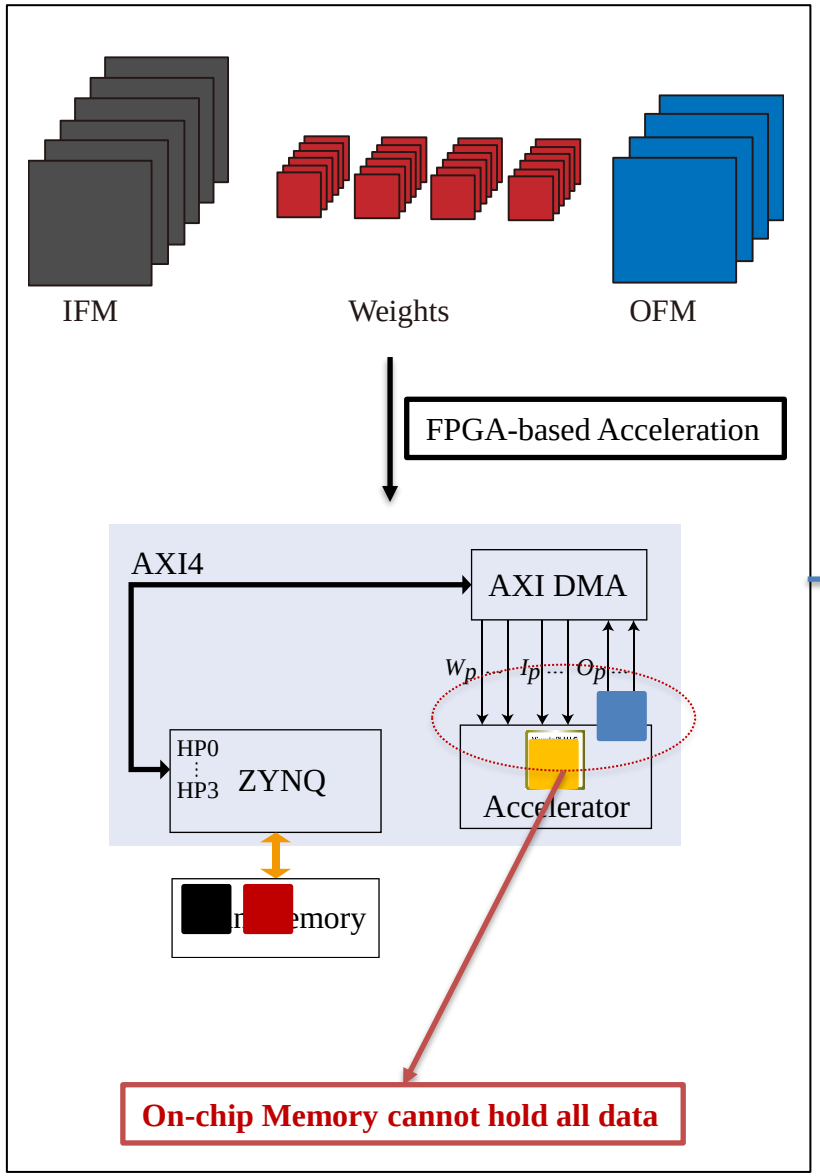
Latency is not improved

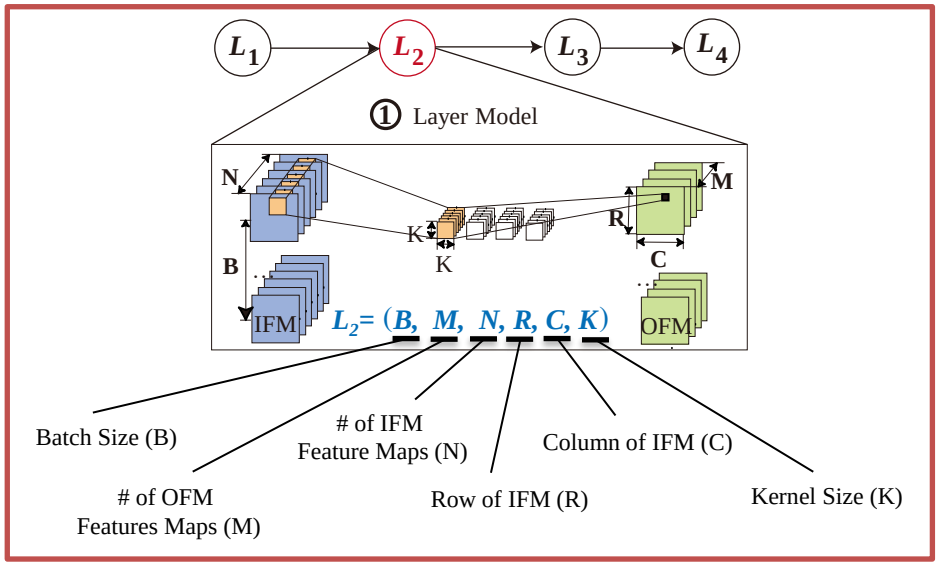
[Ref] C. Zhang et al. Energy-efficient cnn implementation on a deeply pipelined fpga cluster. In Proc. of ISLPED, pages 326–331, 2016.

Pipelining Multi-FPGA: Design Flow

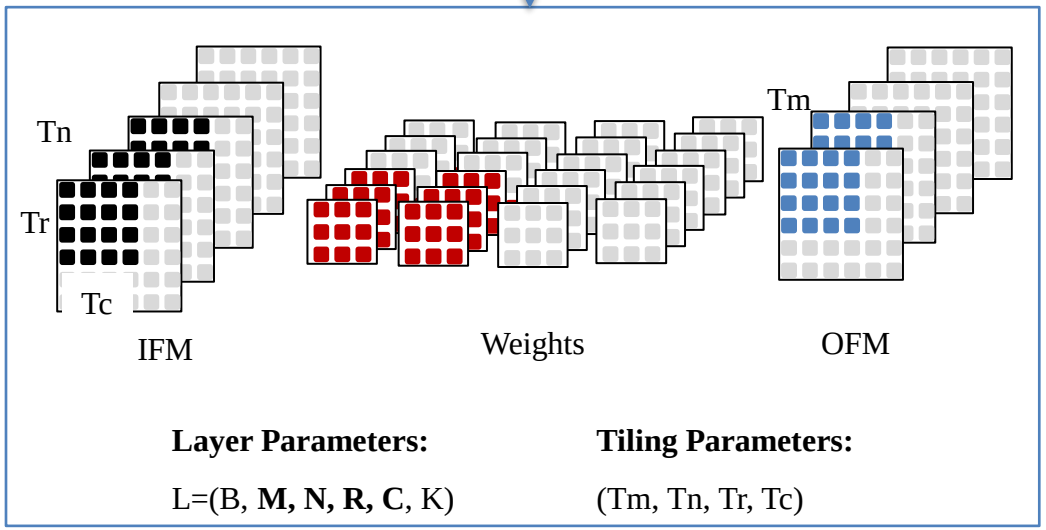


Accelerator Design Parameters





Loop Tiling



Layer Parameters:

L=(B, **M, N, R, C**, K)

Tiling Parameters:

(Tm, Tn, Tr, Tc)

Design Parameters Will Significantly Affect the Performance

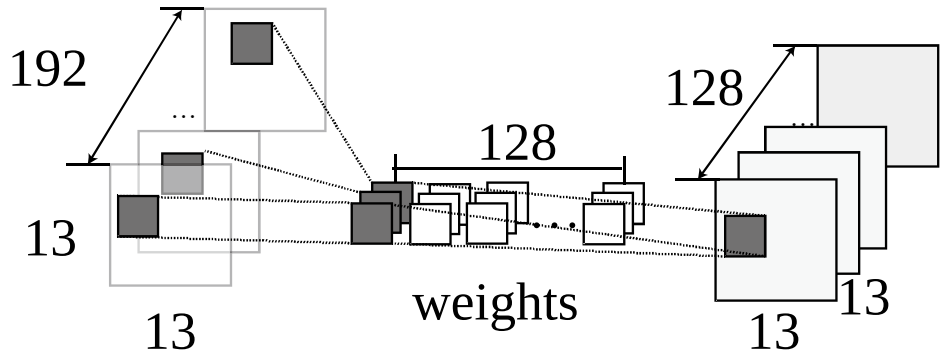
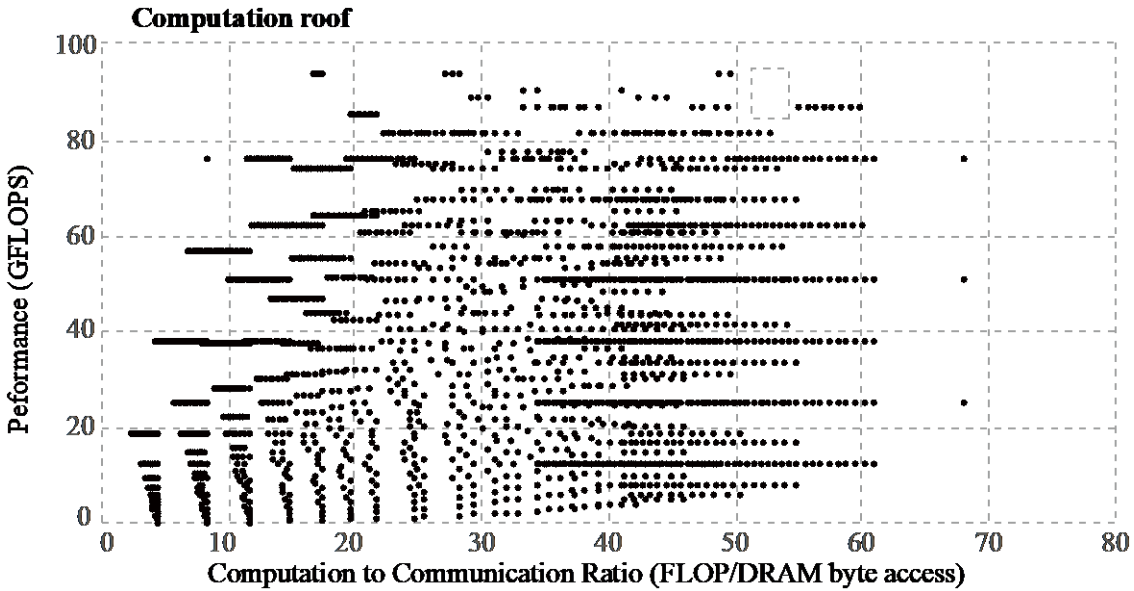


Fig. Convolution of Layer 5 in AlexNet

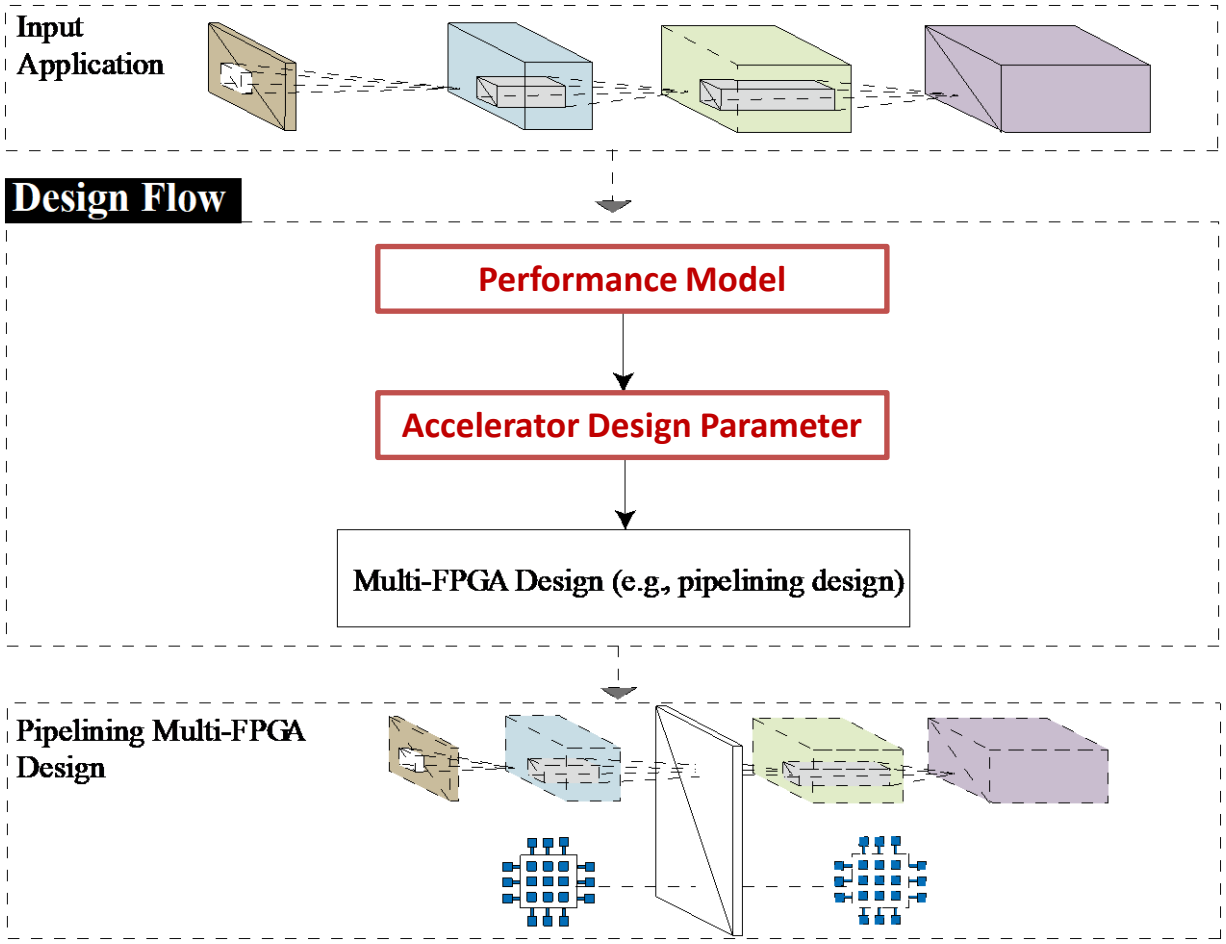


Observation:

Different design parameters lead to different performance

Optimizations on design parameters

Pipelining Multi-FPGA Design Flow



Inaccurate Performance Model Leads to Sub-Optimal Design

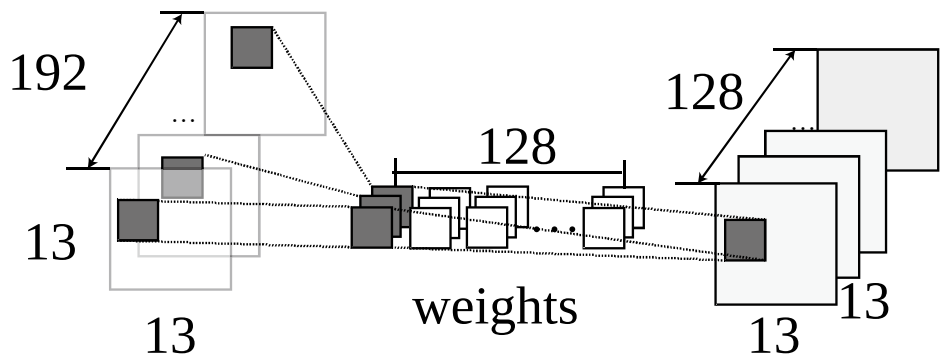


Fig. Convolution of Layer 5 in AlexNet

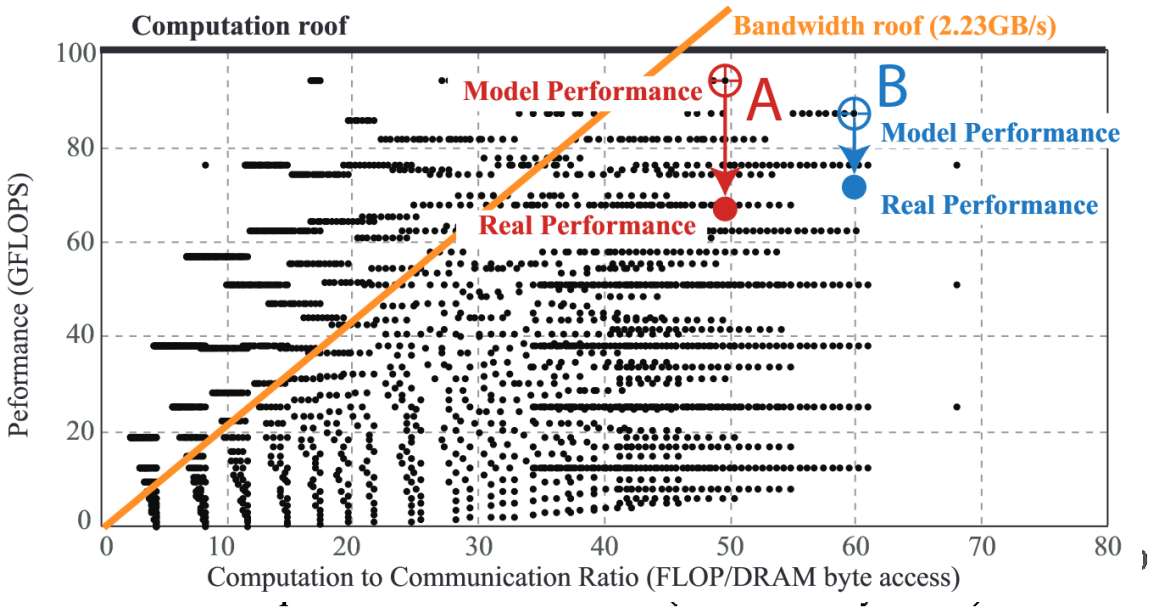


Fig. Real Performance of Layer 5 in Alexnet on ZCU 102

Observation:

Without fine-grained memory access model and inter-FPGA communication model

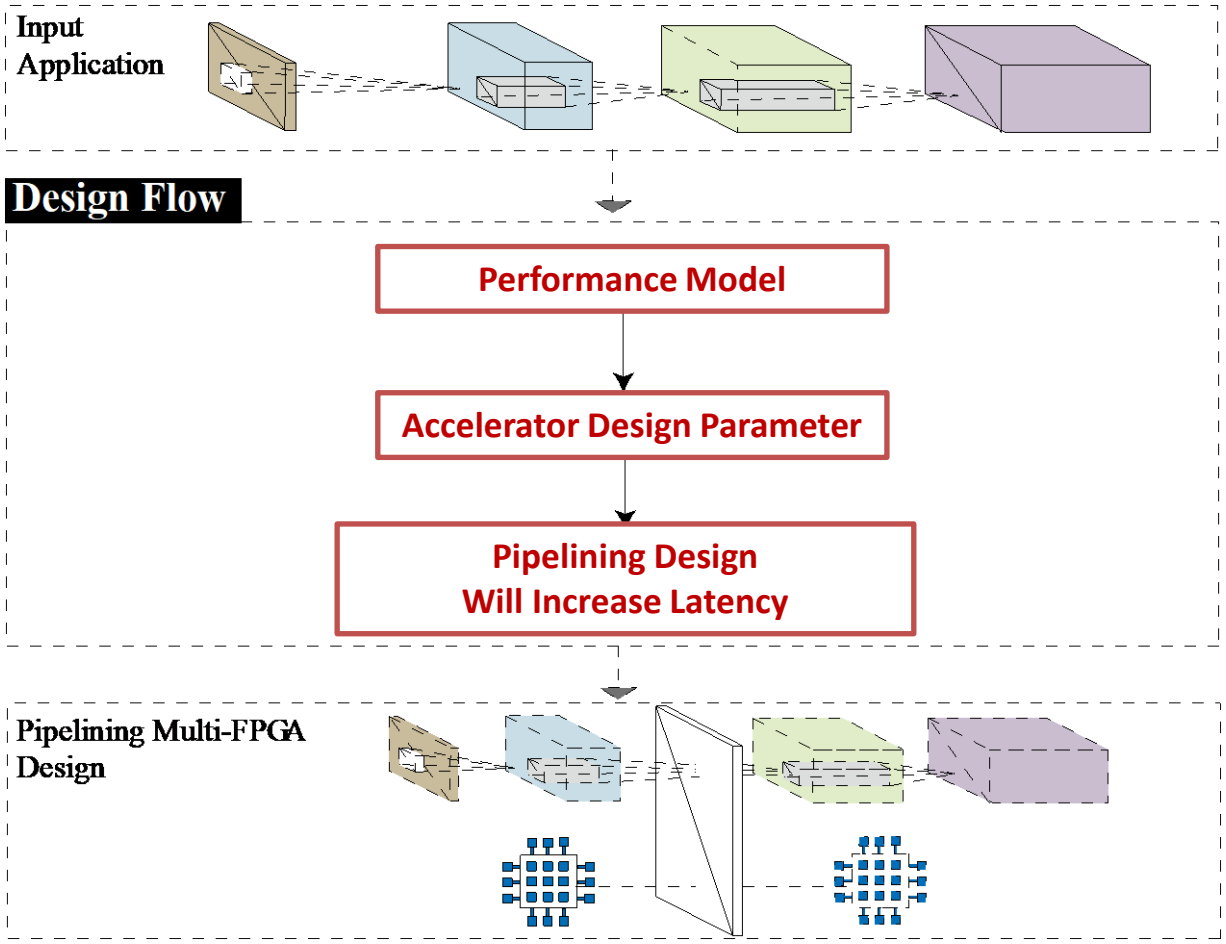
↓

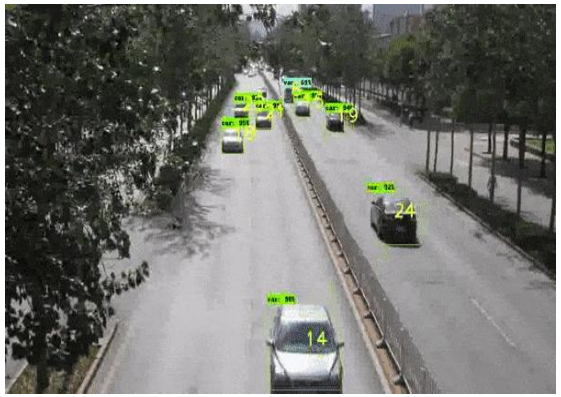
Inaccurate performance model for multi-FPGA

↓

Best model performance
≠
Best real performance
(leads to sub-optimal solution)

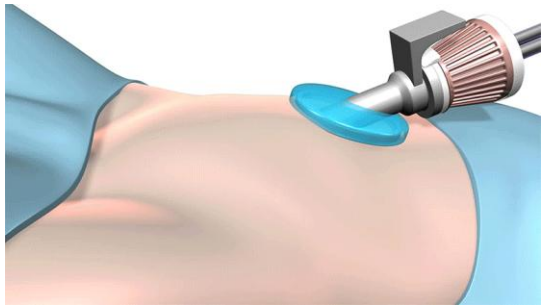
Pipelining Multi-FPGA Design Flow





Traffic Surveillance

Self-Driving Vehicle

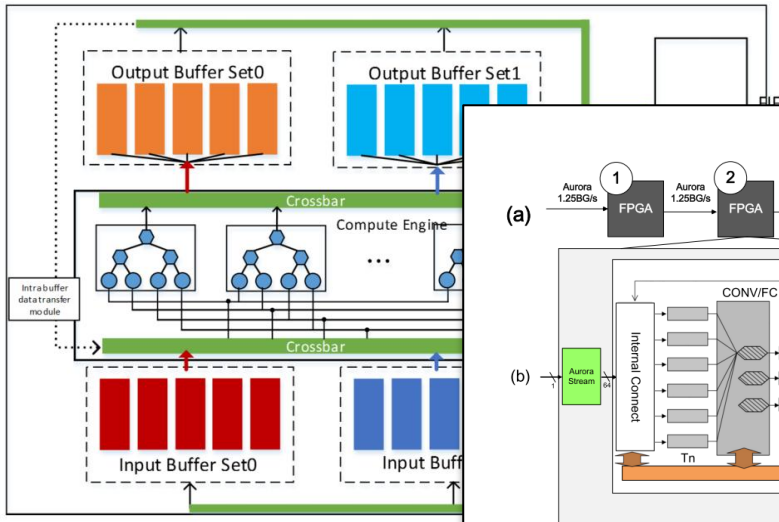


Robots in Surgery

[ref.1] Google's Waymo
[ref.2] da Vinci Surgical Robot at St. Francis Health Center - Robotic Gizmos

Response to Limited Resource: Pipelining Multi-FPGA

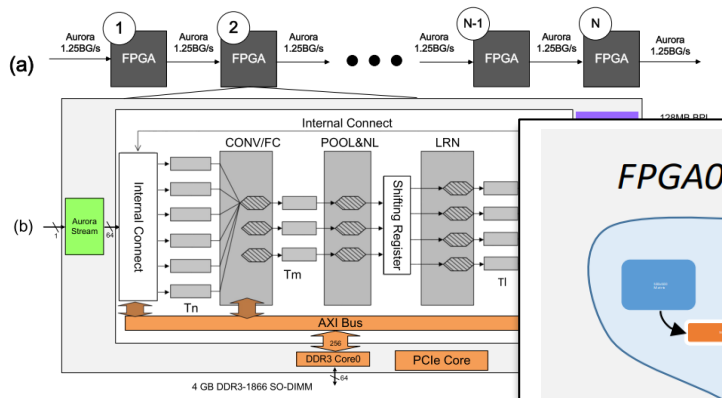
Single-FPGA



Limited Resource → Li

[Ref] C. Zhang et al. Optimizing fpga-
Proc. of FPGA, pages 161–170. ACM

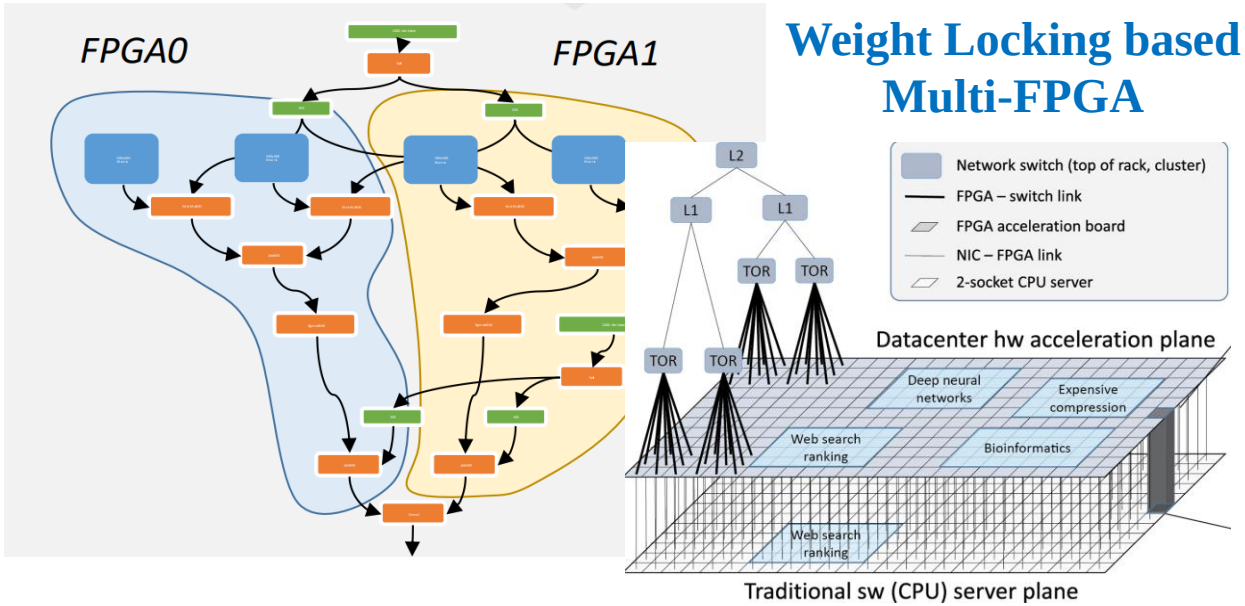
Pipelined FPGAs



Pipeline → High Throu
La

[Ref] C. Zhang et al. Energy-efficient cnn im
ISLPED, pages 326–331, 2016.

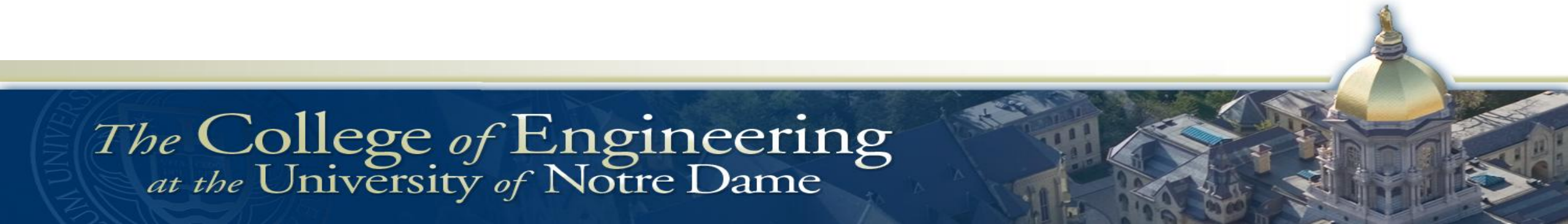
Weight Locking based Multi-FPGA



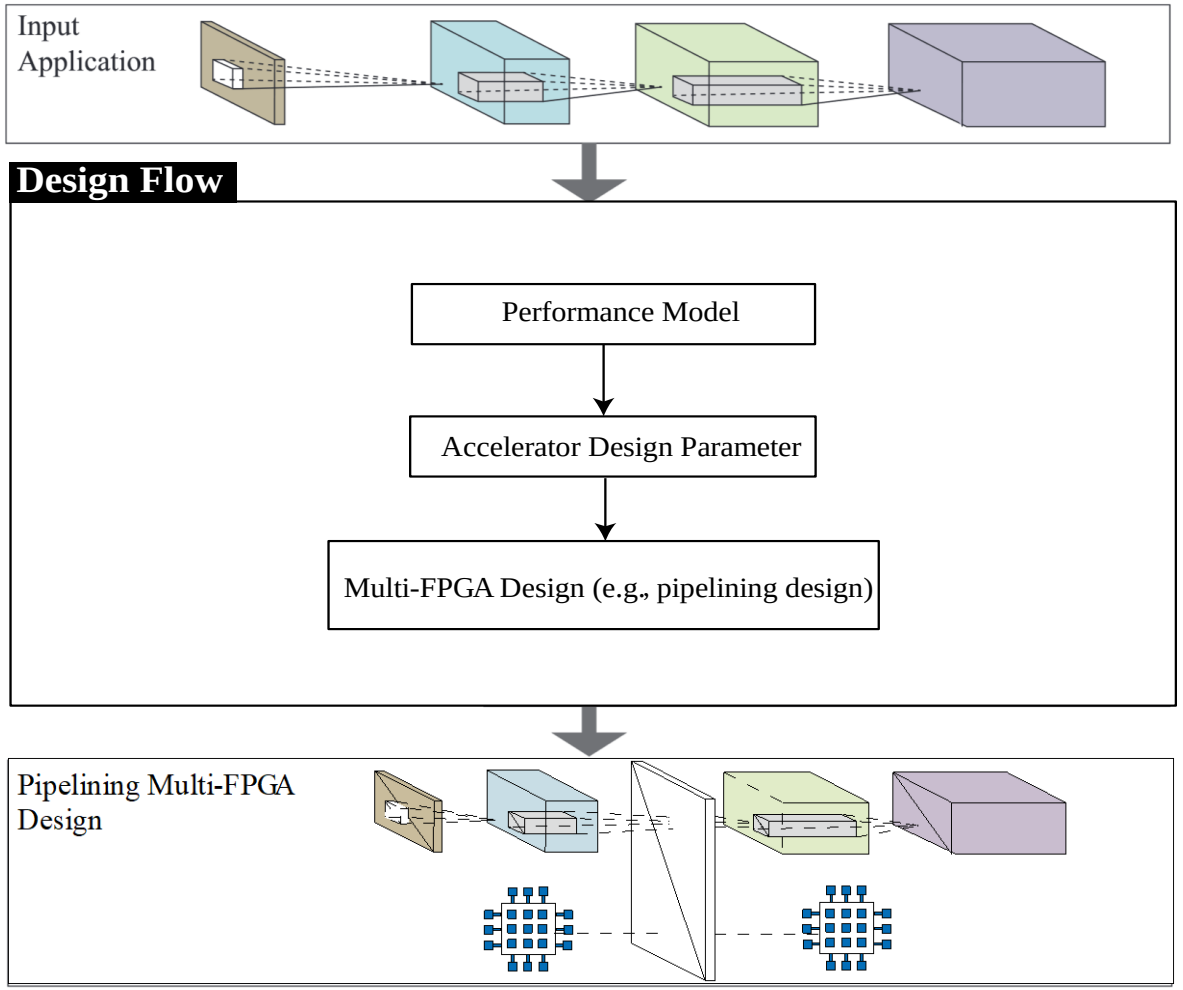
Lock weights to multiple FPGAs → Specific for RNN
Not applicable for sharing weights (e.g. CNN)

[Ref] Eric Chung, et al. Serving DNNs in Real Time at Datacenter Scale with Project Brainwave, In IEEE Micro, 2018

Super-LIP Framework



Super-LIP Framework



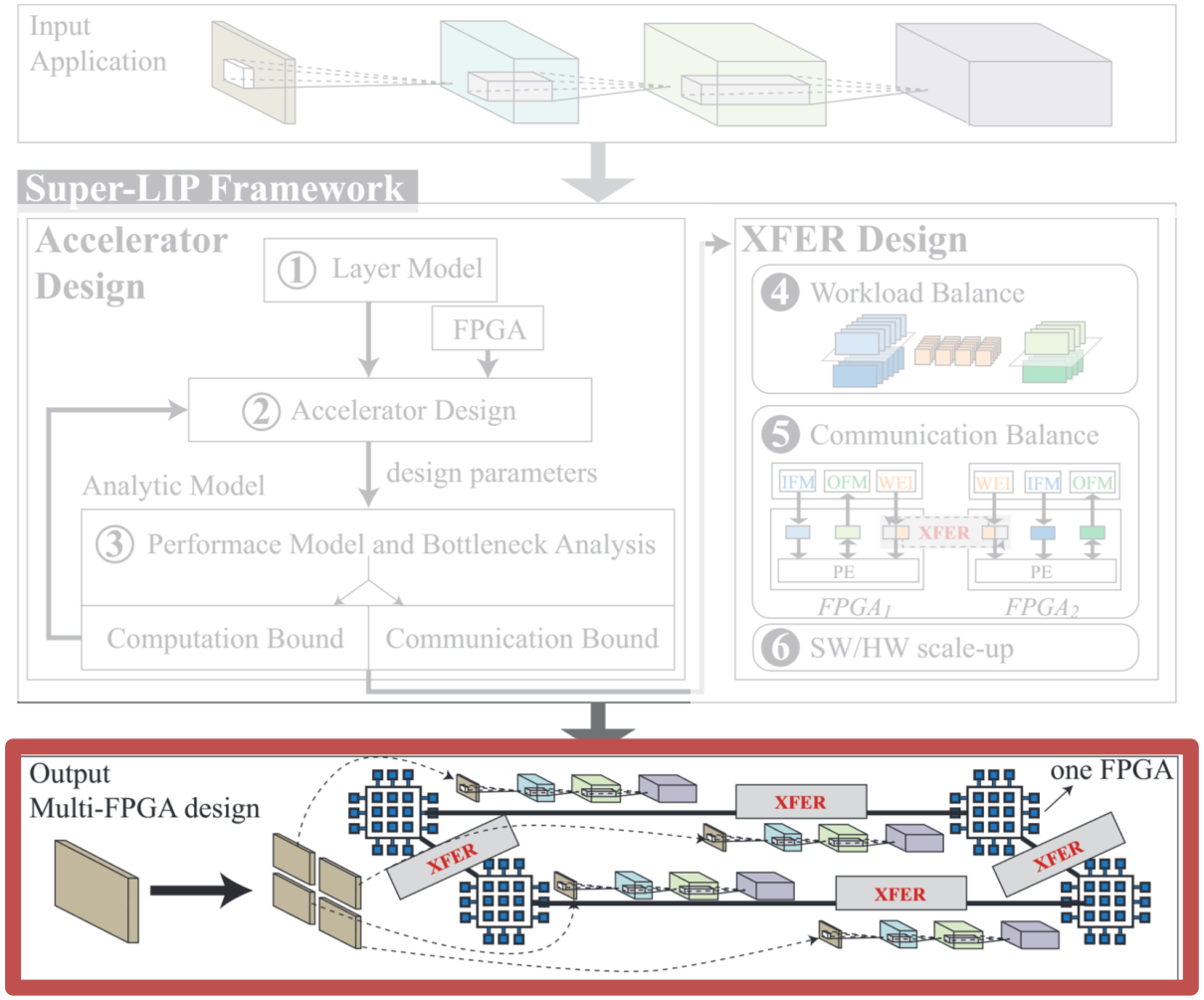
Optimizing Design Parameters with a more Accurate Model

- Accelerator Design
- XFER Design

XFER Design

Targeting on Minimizing Latency

Super-LIP Framework



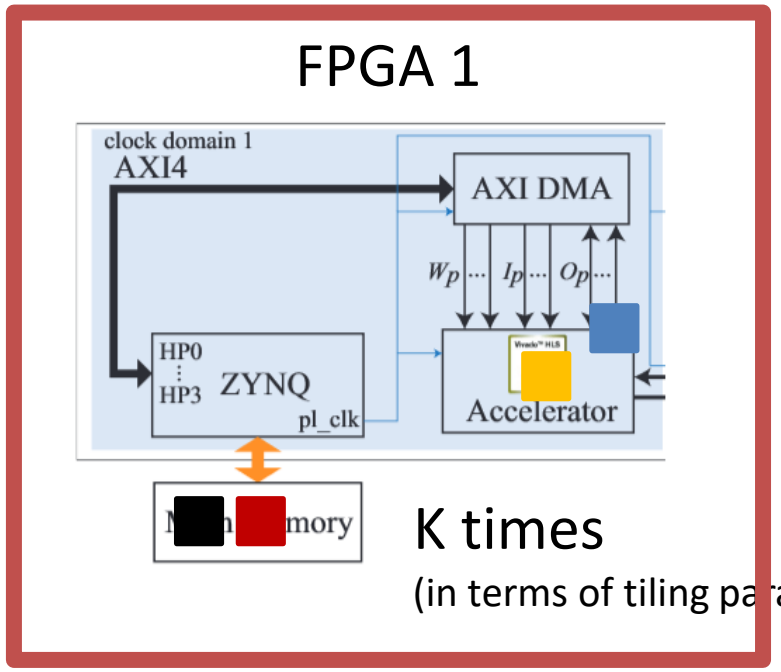
Optimizing Design Parameters with a more Accurate Model

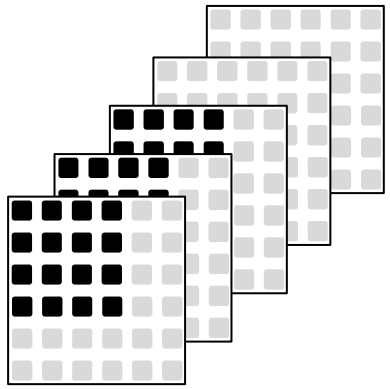
- Accelerator Design
- XFER Design

XFER Design
Targeting on Minimizing Latency

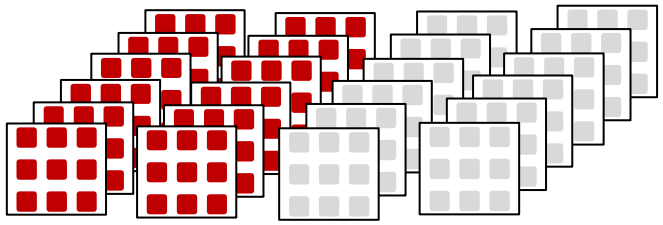
XFER Design

Performance on Single FPGA bounded by Limited Resource

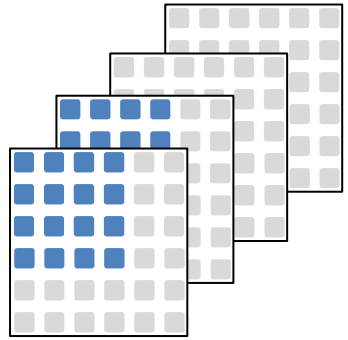




IFM



Weights

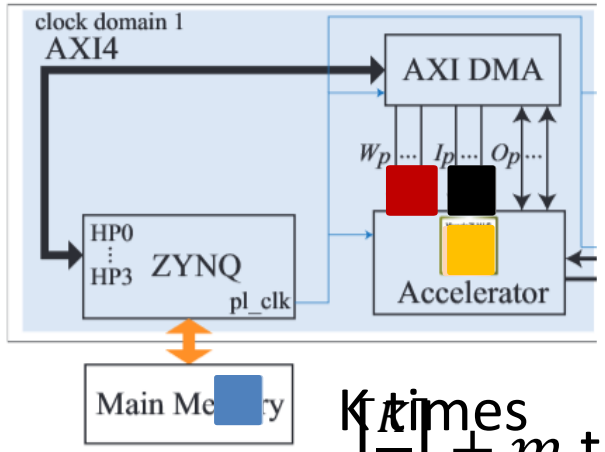


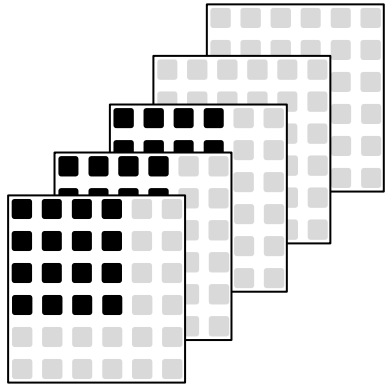
OFM

NN	Operation	Cycles	Note
AlexNet Layer 5	Comm_IFM	2,612	Performance Dominated by **Comm_Weights** **Latency is 5,658**
	Comm_Weights	**5,658**	
	Comm_OFM	368	
	Computation	3,326	

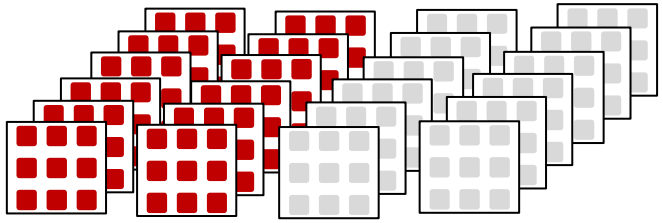
Double Computation Resource cannot Double Performance

FPGA 1

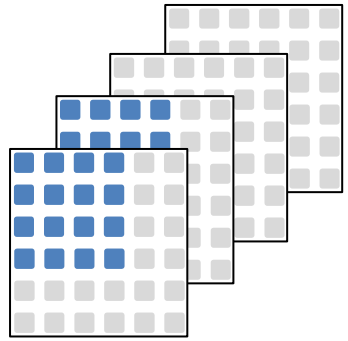




IFM



Weights



OFM

K times

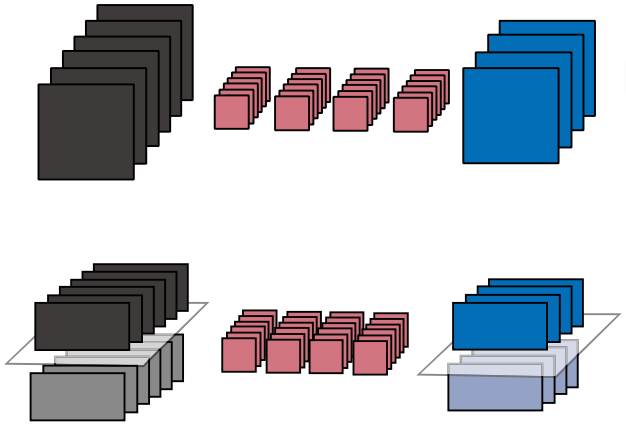
$\left\lfloor \frac{K}{2} \right\rfloor + m$ times

(in terms of tiling parameters)

(K is in terms of tiling parameters)

(m is in terms of stride and padding)

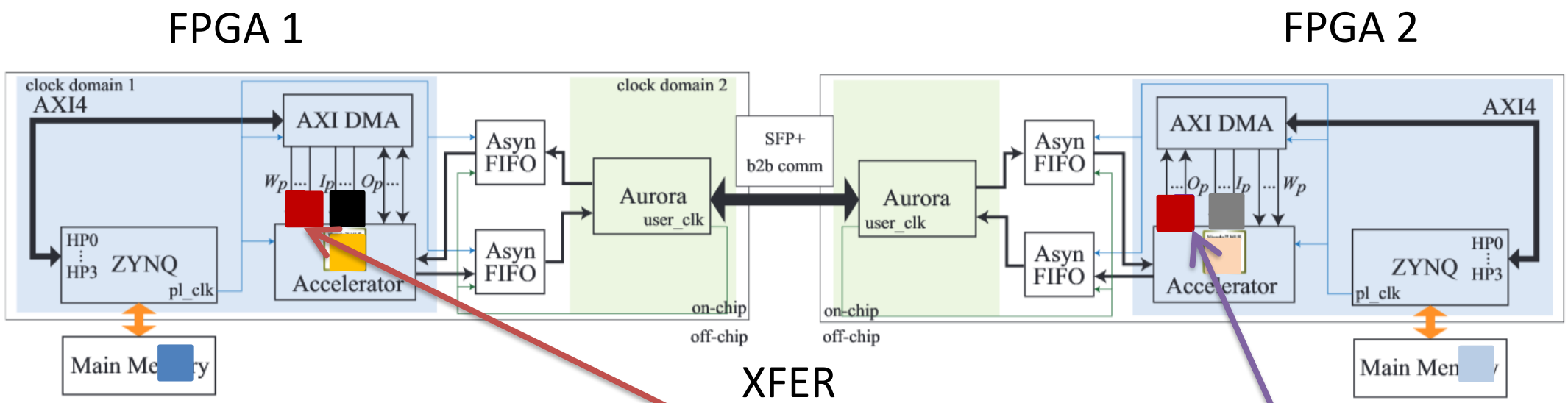
NN	Operation	Single	w/o XFER	Note
AlexNet Layer 5	Comm_IFM	2,612	1,412	**1.91X Speedup**
	Comm_Weights	**5,658**	**2,953**	
	Comm_OFM	368	234	
	Computation	3,326	1,782	



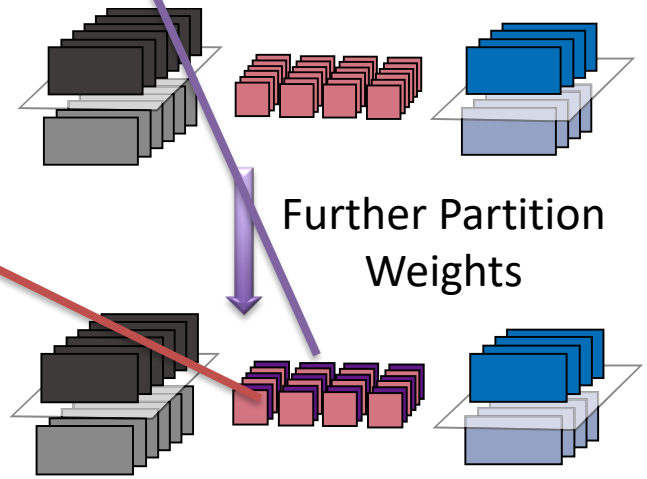
Partition for Parallel

XFER: Achieve Super-Linear Speedup by Transferring Accesses

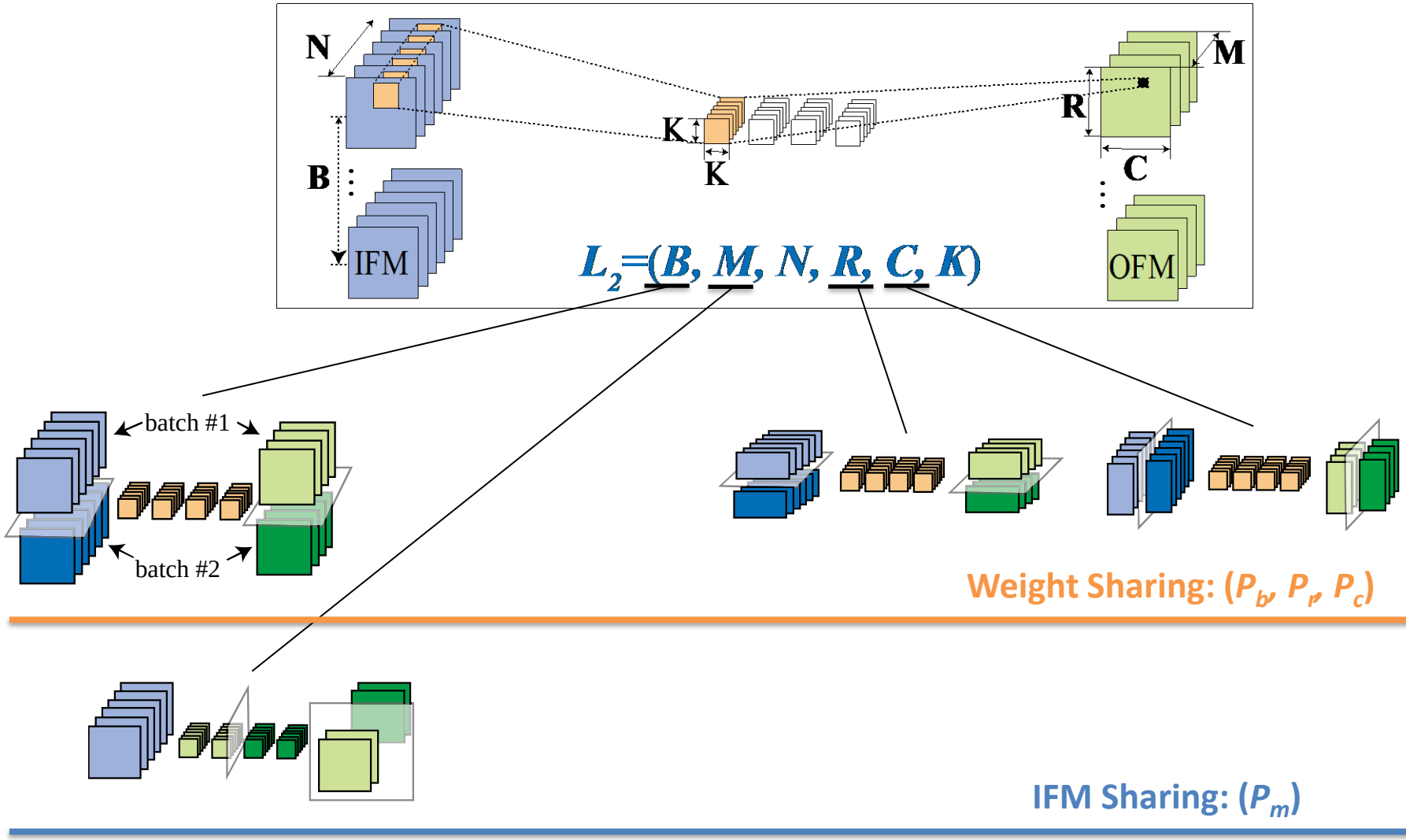
—— from Off-Chip Memory to Inter-FPGA Links



NN	Operation	Single	w/o XFER	w/ XFER	Note
AlexNet Layer 5	Comm_IFM	2,612	1,412	1,412	Bottleneck Moves to Comp. 3.18X Speedup
	Comm_Weights	**5,658**	**2,953**	1,695	
	Comm_OFM	368	234	234	
	Computation	3,326	1,782	**1,782**	

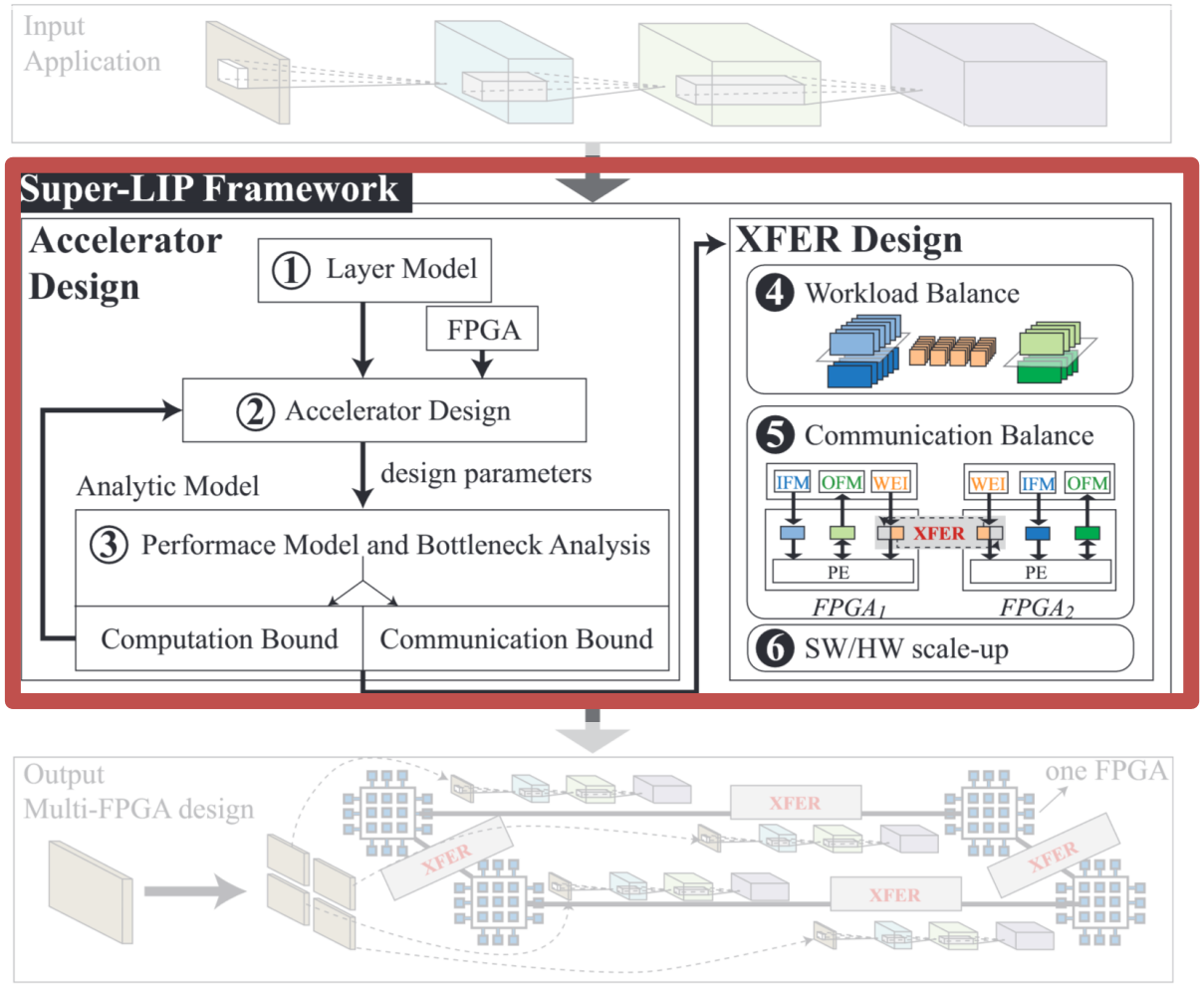


Application Optimization —— Partitioning



Performance Model

Super-LIP Framework

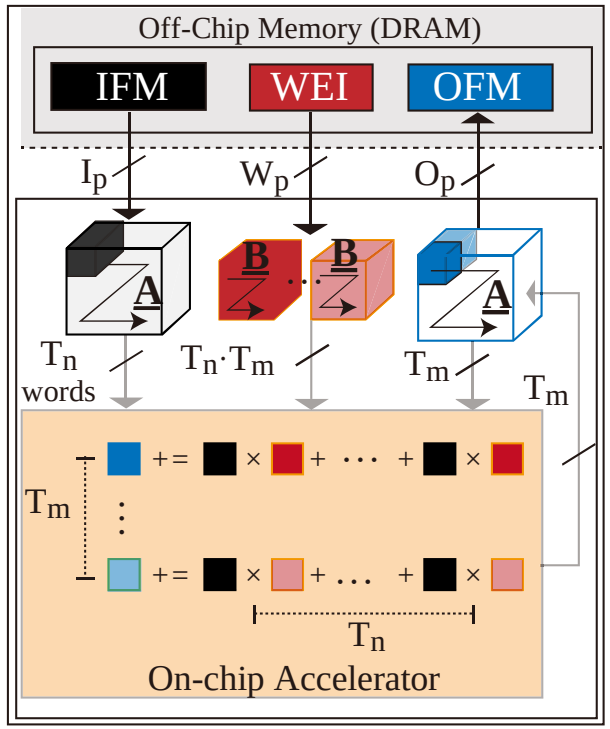


Optimizing Design Parameters with a more Accurate Model

- Accelerator Design
- XFER Design

XFER Design
Targeting on Minimizing Latency

Accelerator Design —— On-Chip Design



Communication Parameters (bit/cycle):

Ip: bandwidth for loading IFM

Wp: bandwidth for loading weights

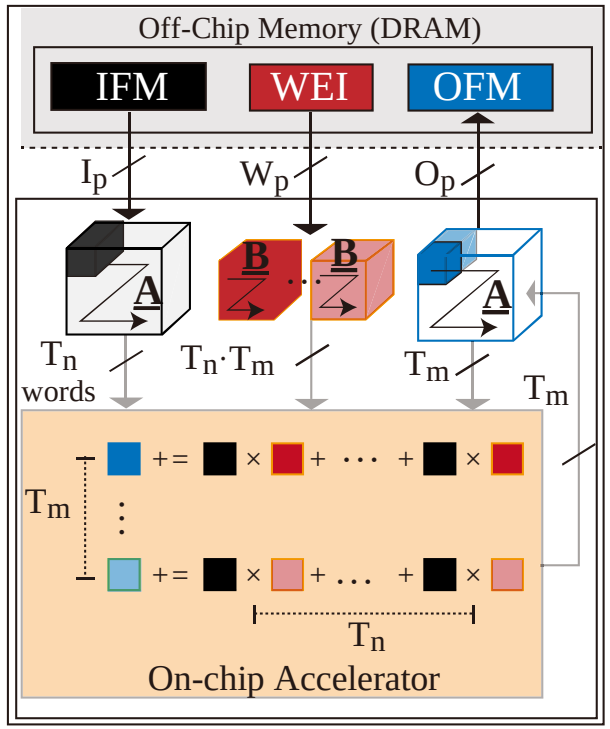
Op: bandwidth for storing OFM

Computation (MACs): $T_m \cdot T_n$

OFM Stationary Data Flow

Accelerator Model

Accelerator Design —— Optimization under Resource Constraints



Communication Parameters (bit/cycle):

Ip: bandwidth for loading IFM

Wp: bandwidth for loading weights

Op: bandwidth for storing OFM

Computation (MACs): $T_m \cdot T_n$

OFM Stationary Data Flow

Problem Definition

Given:

- Layer **(B, M, N, R, C, K)**
- Computation resource — DSP **(𝔻)**
- On-chip memory size **(𝔹)**
- Off-chip memory bandwidth (𝕨)

Determine:

- Tiling: (Tm, Tn, Tr, Tc)
- Bandwidth allocation: (Ip, Wp, Op)
- Partition: (Pb, Pr, Pc, Pm)

Objective:

Minimizing latency under resource constraints

Computation Resource Constraint

Computation(32-bit): $\boldsymbol{T_m \cdot T_n} \leq \mathbb{D}$; *or* (16-bit) $5 \cdot \boldsymbol{T_m \cdot T_n} \leq \mathbb{D}$;

Memory Resource Constraints

IFM buffer: $\mathbf{bI} = \mathbf{2} \times (\boldsymbol{T_n \cdot T_r \cdot T_c \cdot BITs}/\mathbf{18K})$

bitwidth(32b or 16b)

Weight buffer: $\mathbf{bW} = \mathbf{2} \times (\boldsymbol{T_m \cdot T_n \cdot k \cdot k \cdot BITs}/\mathbf{18K})$

OFM buffer: $\mathbf{bO} = \mathbf{2} \times (\boldsymbol{T_m \cdot T_r \cdot T_c \cdot BITs}/\mathbf{18K})$

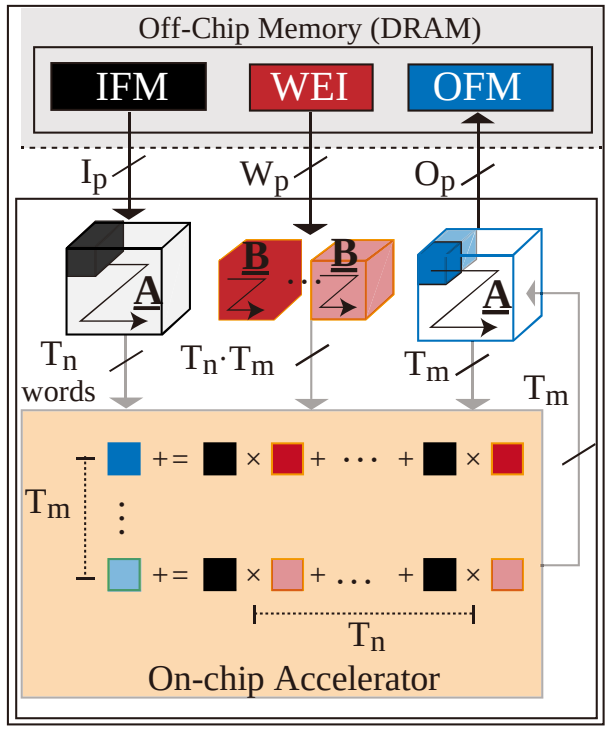
BRAM blokc size

$\mathbf{bI} + \mathbf{bW} + \mathbf{bO} \leq \mathbb{B}$

Communication Resource Constraint

$\boldsymbol{BITs} \times (\boldsymbol{I_p} + \boldsymbol{W_p} + \boldsymbol{O_p}) \leq \mathbb{w}$

Accelerator Design —— Optimization under Resource Constraints



Communication Parameters (bit/cycle):

Ip: bandwidth for loading IFM

Wp: bandwidth for loading weights

Op: bandwidth for storing OFM

Computation (MACs): $T_m \cdot T_n$

OFM Stationary Data Flow

Problem Definition

Given:

- Layer **(B, M, N, R, C, K)**
- Computation resource — DSP **(𝔻)**
- On-chip memory size **(𝔹)**
- Off-chip memory bandwidth (𝕨)

Determine:

- Tiling: (Tm, Tn, Tr, Tc)
- Bandwidth allocation: (Ip, Wp, Op)
- Partition: (Pb, Pr, Pc, Pm)

Objective:

Minimizing latency under resource constraints

Memory Access and Computation Latency

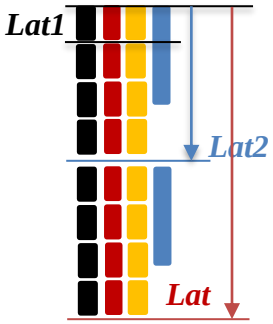
Load IFM: $tI_{mem} = T_n \cdot T_r \cdot T_c / I_p$

Load Weight: $tW_{mem} = T_m \cdot T_n \cdot K \cdot K / W_p$

Store OFM: $tO_{mem} = T_m \cdot T_r \cdot T_c / O_p$

Computation: $tComp = K \cdot K \cdot T_r \cdot T_c$

Latency under Parallelism: OFM stationary



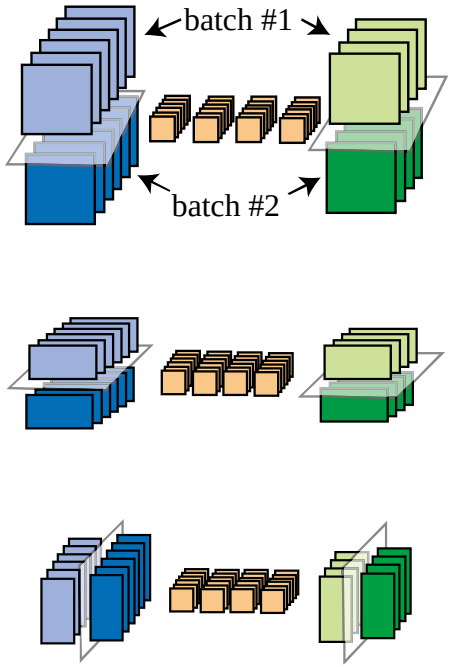
Inner most para: $Lat_1 = max\{tComp, tI_{mem}, tW_{mem}\}$

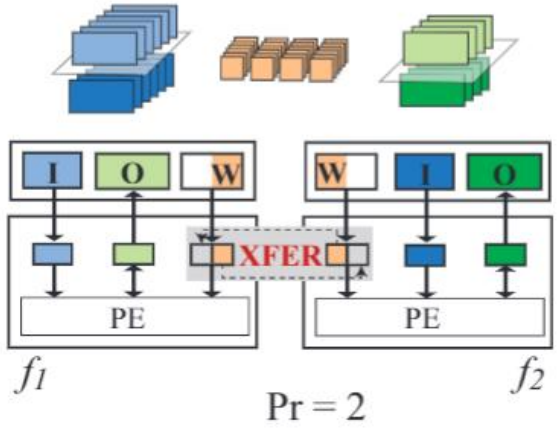
Second layer para: $Lat_2 = max\{\frac{N}{T_n} Lat_1, tO_{mem}\}$

Total Latency: *# of loops* *Pipeline loading time*

$$Lat = B \times \frac{R}{T_r} \times \frac{C}{T_c} \times \frac{M}{T_m} \times Lat_2 + (tO_{mem} + Lat_1)$$

XFER Design —— Weight Sharing






```
void XFER_load_weight(float bW[SIZE],
                      stream<float> &mem_in,
                      stream<float> &b2b_in,
                      stream<float> &b2b_out){
  for(int i=0; i<SIZE/2; i++){
#pragma HLS PIPELINE II=1
#pragma HLS dependence variable=bW intra false
    bW[i] = mem_in.read();
    bW[i+SIZE/2] = b2b_in.read();
    b2b_out.write(bW[i]);
}}// end of for loop and function
```

HLS codes for FPGA f_2

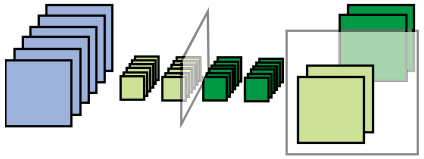
Model Modifications:

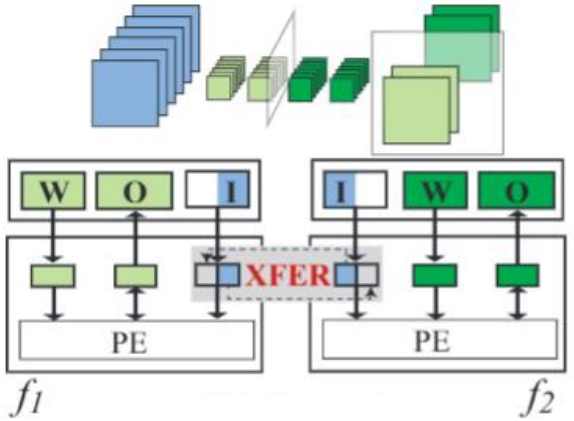
Load Weight: $tW_{mem} = T_m \cdot T_n \cdot K \cdot K/(W_p \cdot P_b \cdot P_r \cdot P_c)$

Inter-FPGA latency: $tW^i_{b2b} = T_m \cdot T_n \cdot K \cdot K/(W^{b2b}_p \cdot P_b \cdot P_r \cdot P_c)$

Inner most para: $Lat_1 = max\{tComp, tI_{mem}, tW_{mem}, \max_{i\in[1,\ldots,P-1]}\{tW^i_{b2b}\}\}$

XFER Design —— IFM Sharing






```
void XFER_load_ifm(float bI[SIZE],
                   stream<float> &mem_in,
                   stream<float> &b2b_in,
                   stream<float> &b2b_out){
  for(int i=0; i<SIZE/2; i++){
#pragma HLS PIPELINE II=1
#pragma HLS dependence variable=bI intra false
    bI[i] = mem_in.read();
    bI[i+SIZE/2] = b2b_in.read();
    b2b_out.write(bW[i]);
  }}// end of for loop and function
```

HLS codes for FPGA f_2

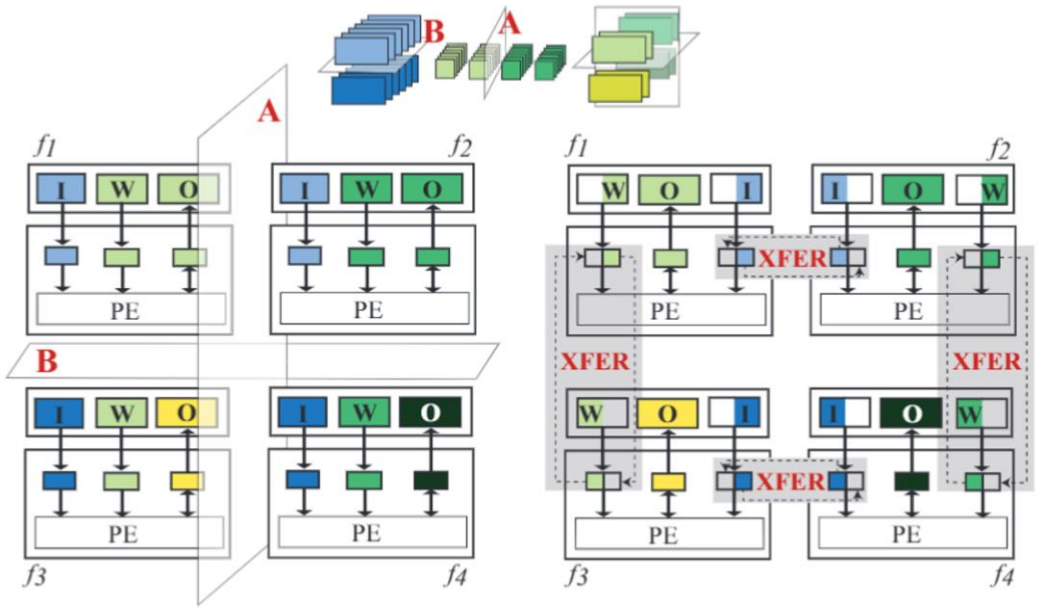
Model Modifications:

Load IFM: $tI_{mem} = T_n \cdot T_r \cdot T_c/(I_p \cdot P_m)$

Inter-FPGA latency: $tI^i_{b2b} = T_m \cdot T_n \cdot K \cdot K/(I^{b2b}_p \cdot P_m)$

Inner most para: $Lat_1 = max\{tComp, tI_{mem}, tW_{mem}, \max_{i\in[1,\ldots,Q-1]}\{tI^i_{b2b}\}\}$

XFER Design —— Scale-Up



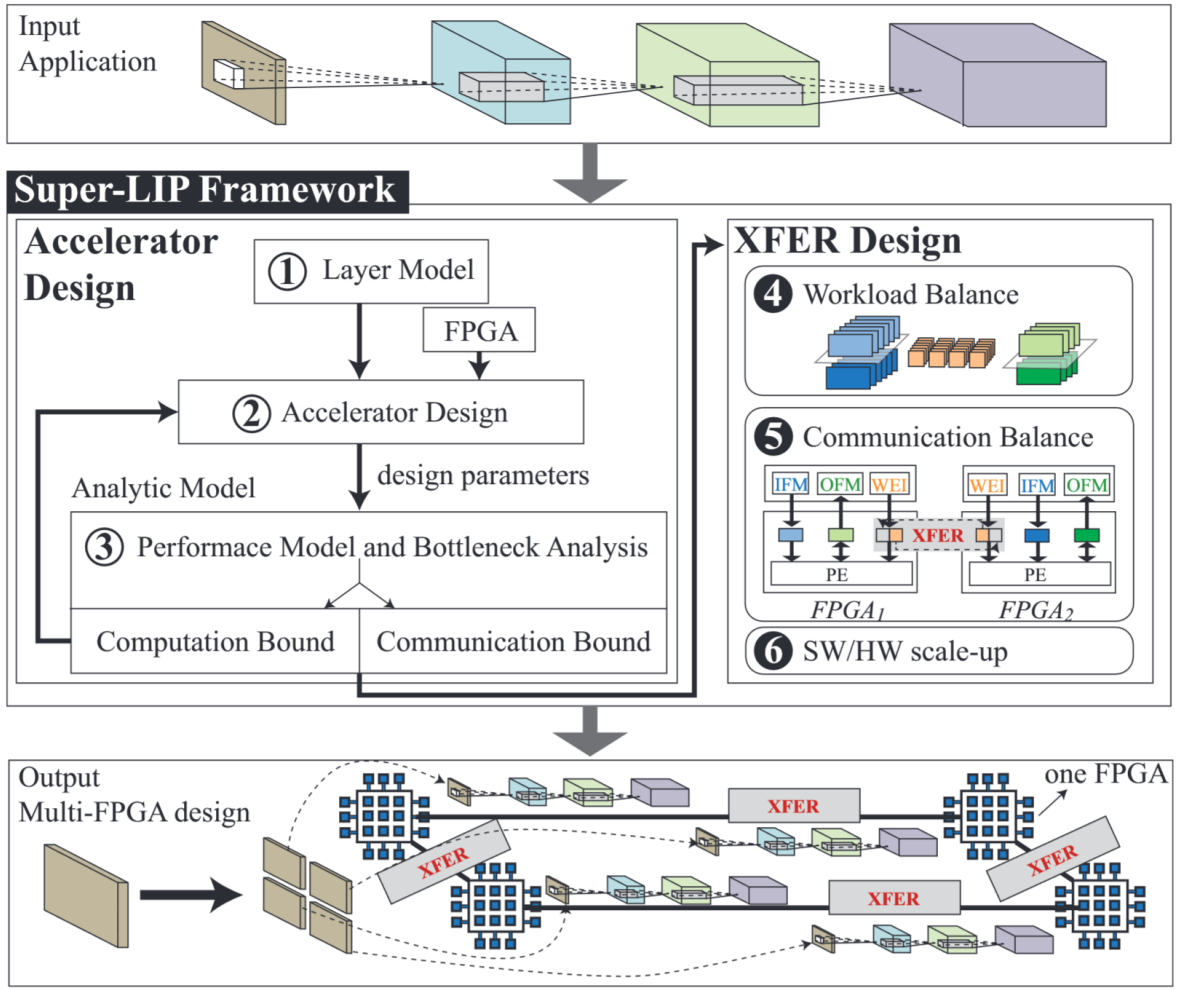
Example:

$P_b \cdot P_r \cdot P_c = 2$ & $P_m = 2$

Topology: 2D-torus

Data Movement: Pipelining

Recap —— Super-LIP Framework



Optimizing Design Parameters with a more Accurate Model

- Accelerator Design
- XFER Design

XFER Design
Targeting on Minimizing Latency

Model Overview

1. Resource Constraints

Computation(32-bit): $\boldsymbol{T_m \cdot T_n \le \mathbb{D}}$; *or* (16-bit) $\boldsymbol{5 \cdot T_m \cdot T_n \le \mathbb{D}}$;

IFM buffer: $\mathbf{bI = 2 \times (T_n \cdot T_r \cdot T_c \cdot BITs/18K)}$

Weight buffer: $\mathbf{bW = 2 \times (T_m \cdot T_n \cdot k \cdot k \cdot BITs/18K)}$

OFM buffer: $\mathbf{bO = 2 \times (T_m \cdot T_r \cdot T_c \cdot BITs/18K)}$

$\mathbf{bI + bW + bO \le \mathbb{B}}$

$\boldsymbol{BITs \times (I_p + W_p + O_p) \le \mathbb{w}}$

2. Memory Access and Computation Latency

Load IFM: $\boldsymbol{tI_{mem} = T_n \cdot T_r \cdot T_c/(I_p \cdot P_m)}$

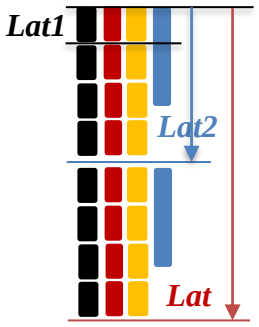
Load Weight:

$\boldsymbol{tW_{mem} = T_m \cdot T_n \cdot K \cdot K/(W_p \cdot P_b \cdot P_r \cdot P_c)}$

Store OFM: $\boldsymbol{tO_{mem} = T_m \cdot T_r \cdot T_c/O_p}$

Computation: $\boldsymbol{tComp = K \cdot K \cdot T_r \cdot T_c}$

3. Parallelism (Pipelining and Double Buffer)



Inner most para:

$$\boldsymbol{Lat_1 = max\{tComp, tI_{mem}, tW_{mem}, \max_{i\in[1,\dots,P-1]}\{tW^i_{b2b}\}, \max_{i\in[1,\dots,Q-1]}\{tI^i_{b2b}\}\}}$$

Second layer para: $\boldsymbol{Lat_2 = max\{\frac{N}{T_n} Lat_1, tO_{mem}\}}$

Total Latency:

$$\boldsymbol{Lat = B \times \frac{R}{T_r} \times \frac{C}{T_c} \times \frac{M}{T_m} \times Lat_2 + (tO_{mem} + Lat_1)}$$

4. Communication among Multiple FPGAs (XFER)

Transfer IFM:

$\boldsymbol{tI^i_{b2b} = T_m \cdot T_n \cdot K \cdot K/(I^{b2b}_p \cdot P_m)}$

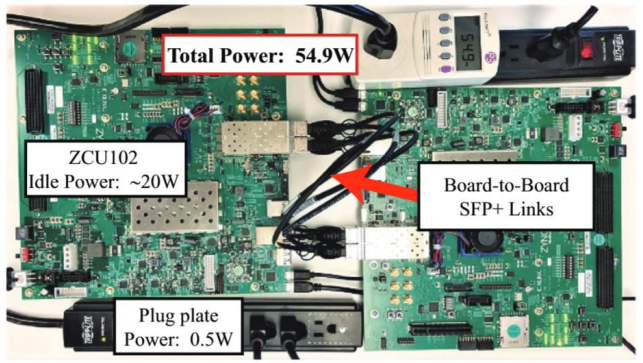
Transfer Weights:

$\boldsymbol{tW^i_{b2b} = T_m \cdot T_n \cdot K \cdot K/(W^{b2b}_p \cdot P_b \cdot P_r \cdot P_c)}$

Objective: $\boldsymbol{min = Lat}$

Results

Comparison with Existing Designs and GPU Platforms



Testbed:
CNN: AlexNet

FPGA: Xilinx ZUC102 FPGAs connected by SFP+ Links

Comparison results of **XFER** with comparisons to **GPUs** and the **existing FPGA designs**

Design	mGPU		GPU		*FPGA15*		*ISCA17*		*ISLPED16*		**XFER**			
Precision	32bits float		32bits float		32bits float		32bits float		16bits fixed		32bits float		16bits fixed	
Device	Jetson TX2		Titan X		VX485T		VX485T		4×VX690t		2×ZCU102		2×ZCU102	
Freq (MHz)	1300MHz		1139MHz		100MHz		100MHz		150MHz		100MHz		200MHz	
Power (Watt)	16.00		162.00		18.61		-		126.00		52.40		54.40	
DSP Uti.	-		-		80%		80%		-		90.79%		55.87%	
BRAM Uti.	-		-		49.71%		43.25%		-		72.92%		92.43%	
Overall Perf.	Lat. *ms*	Thr. *GOPS*	Lat. *ms*	Thr. *GOPS*	Lat. *ms*	Thr. *GOPS*	Lat. *ms*	Thr. *GOPS*	Lat. *ms*	Thr. *GOPS*	Lat. *ms*	Thr. *GOPS*	Lat. *ms*	Thr. *GOPS*
	11.1 - 13.2	110.75	5.1 - 6.4	235.55	21.62	69.09	60.13	85.47	30.6	128.8	10.13	149.54	**2.27**	**679.04**
E.-E. (GOPS/W)	6.88		1.45		3.71		-		1.02		2.85		**12.48**	

Lowest Latency among all competitors

Comparison with Single-FPGA Design on ZCU102 Boards

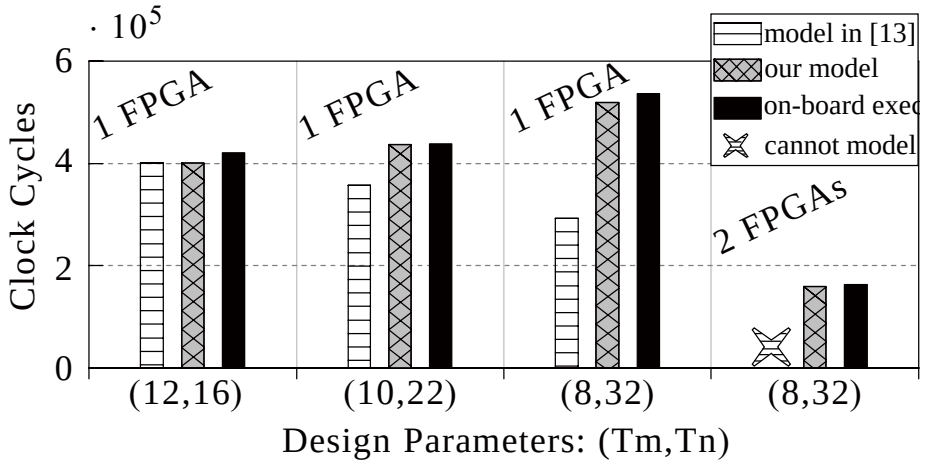
Comparison results of **XFER** with SOTA single FPGA design

Design	32bits float				16bits fixed			
	FPGA15		**Super-LIP**		*FPGA15*		**Super-LIP**	
$\langle T_m, T_n \rangle$	⟨64, 7⟩		⟨64, 7⟩		⟨64, 24⟩		⟨128, 10⟩	
Power (W)	25.70 (1 FPGA)		**52.40 (2 FPGAs)**		26.00 (1 FPGA)		**54.40 (2 FPGAs)**	
Perf.	Lat. ms	Thr. GOPS	**Lat. ms**	**Thr. GOPS**	Lat. ms	Thr. GOPS	**Lat. ms**	**Thr. GOPS**
conv1	7.36	28.6	**3.66**	**57.6**	3.74	56.5	**0.94**	**224.5**
conv2	5.20	86.1	**2.55**	**175.5**	1.48	302.6	**0.48**	**933.1**
conv3	4.50	66.4	**1.73**	**172.7**	1.20	249.6	**0.33**	**906.2**
conv4	3.41	65.7	**1.31**	**171.0**	0.89	252.6	**0.35**	**640.8**
conv5	2.28	66.0	**0.88**	**170.9**	0.59	251.7	**0.17**	**879.5**
overall	22.75	66.6	**10.13**	**149.5**	7.90	195.1	**2.27**	**679.0**
Perf. Impr.	1.00×		**2.25×**		1.00×		**3.48×**	
E.-E. (GOPS/W)	2.59		**2.85**		7.51		**12.48**	
E.-E. Impr.	-		**9.21%**		-		**39.86%**	

Achieve Super-Linear Speedup

for both 32-bits and 16-bits implementations

Model Accuracy Evaluation



- Ours: average deviation is **only 2.53%**
- Existing model: deviation is **up to 45.47%**
- We can accurately estimate performance of **multiple FPGAs**

Performance Bottleneck Detection and Alleviation using the proposed model

Design	Precision	Tm ,Tn	Partition	Our Model				On-Board			Deviation			Speedup
				Cycles	BRAM	DSPs	Bound	Cycles	BRAM	DSPs	Cycles	BRAM	DSPs	
A (Single)	32b float	8, 32	-	519168	592	1280	IFM	535530	624	1326	3.06%	5.13%	3.47%	baseline
B (XFER)		8, 32	Pm=2	158880	592	1280	Comp.	162114	640	1331	1.99%	7.50%	3.83%	3.30X
C (Single)	16b fixed	64, 20	-	115200	1448	1280	Weight	118688	1516	1324	2.94%	4.49%	3.32%	baseline
D (XFER)		64, 20	Pr=2	32760	1448	1280	Comp.	34622	1530	1330	5.38%	5.36%	3.76%	3.43X

Open Source

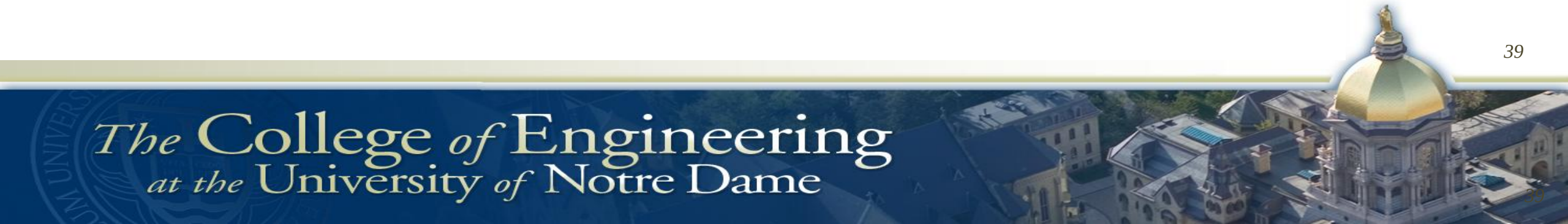
https://github.com/PITT-JZ-COOP/XFER_FPGA

SCAN ME

- **Performance Model**
- **XFER Design**
  - HLS
  - SDK
  - Vivado_project

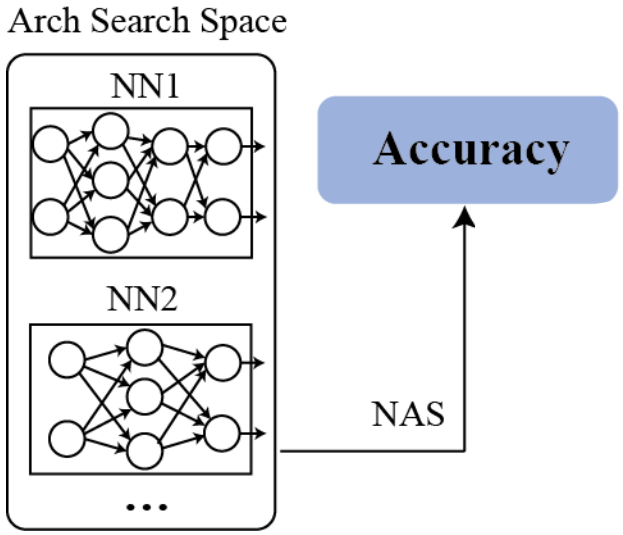
Outline

- Introduction
- XFER: Achieving Super-Linear Speedup across Multi-FPGA
  - CODES+ISSS'19 (Best Paper Finalist)
- **Co-Exploration of Neural Architecture and FPGA Implementation**
  - DAC'19 (Best Paper Finalist)
- Other Research Directions
- Conclusion



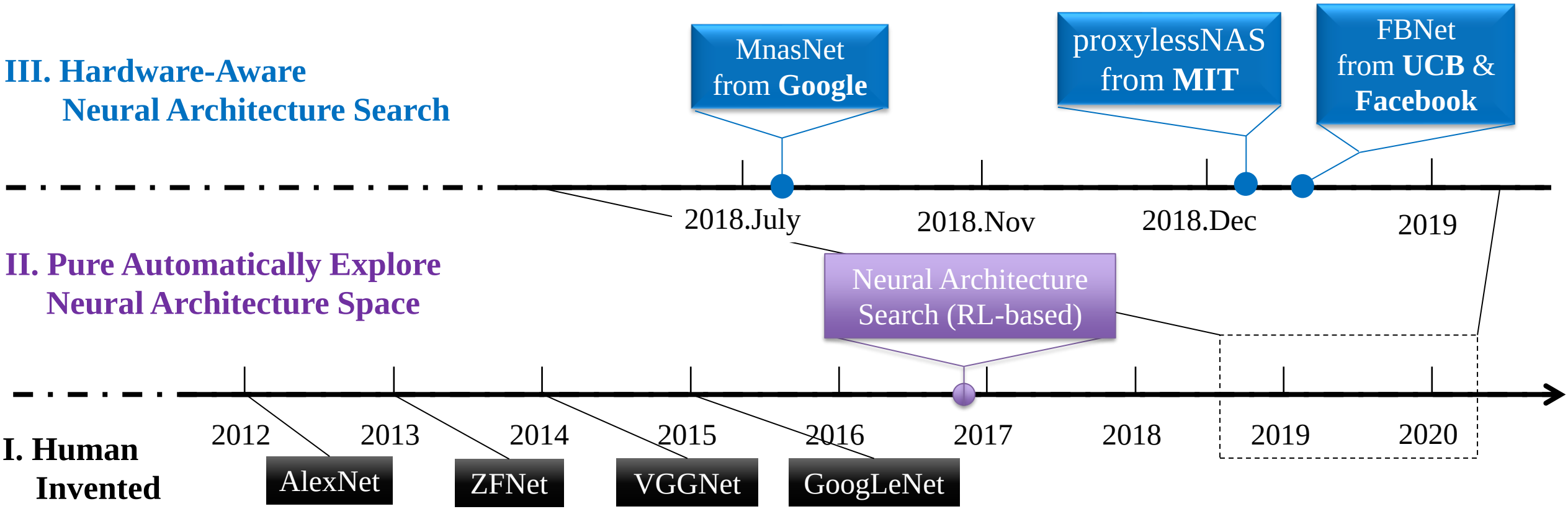
39

Efforts on Both Software and Hardware Sides are Required

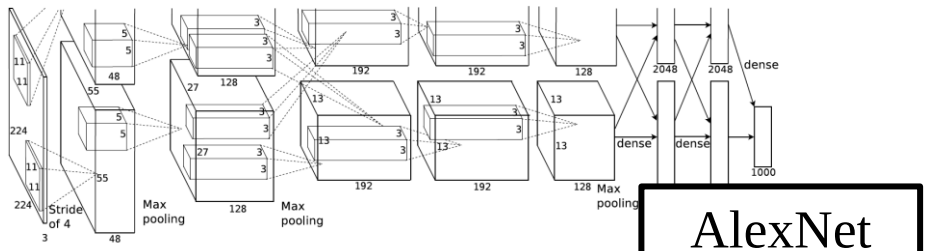


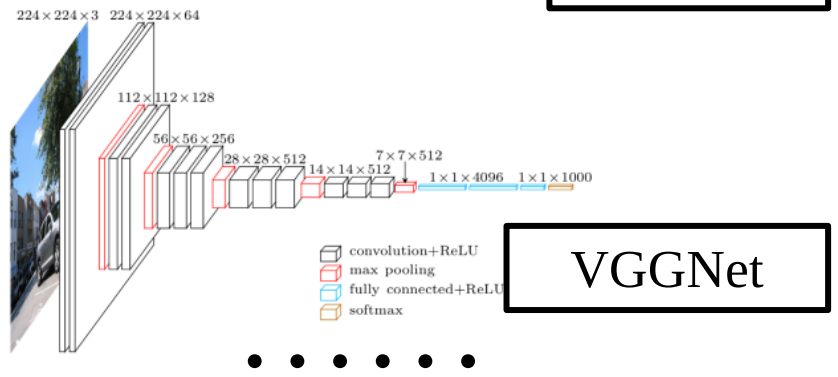
Neural Architecture Search

Evolution of Exploring Deep Neural Architectures

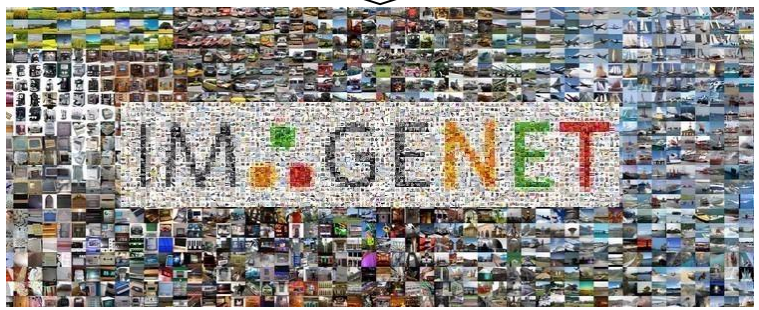


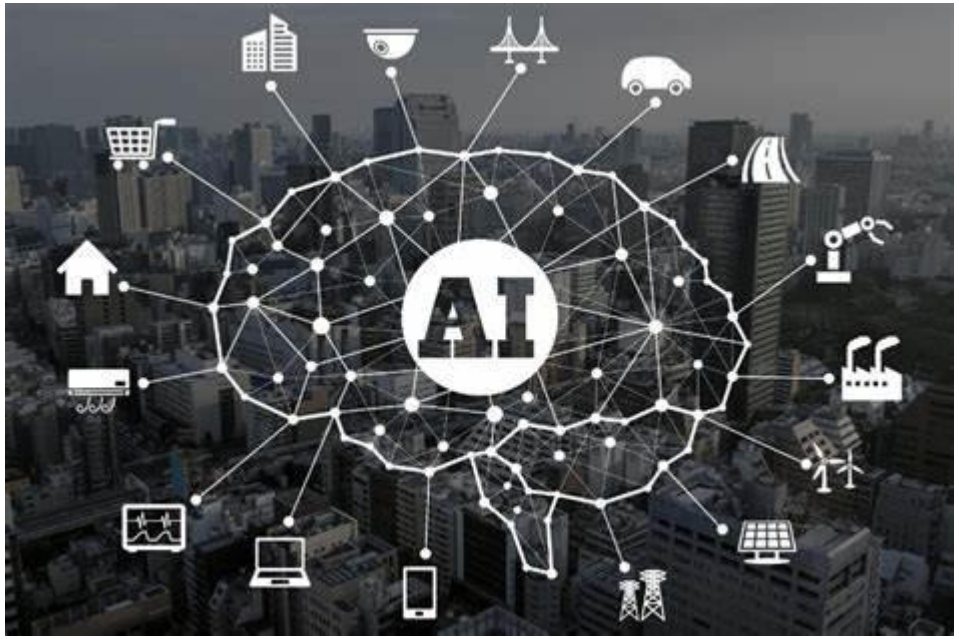
Phase I: Human Invented





Design arch. for dedicated application



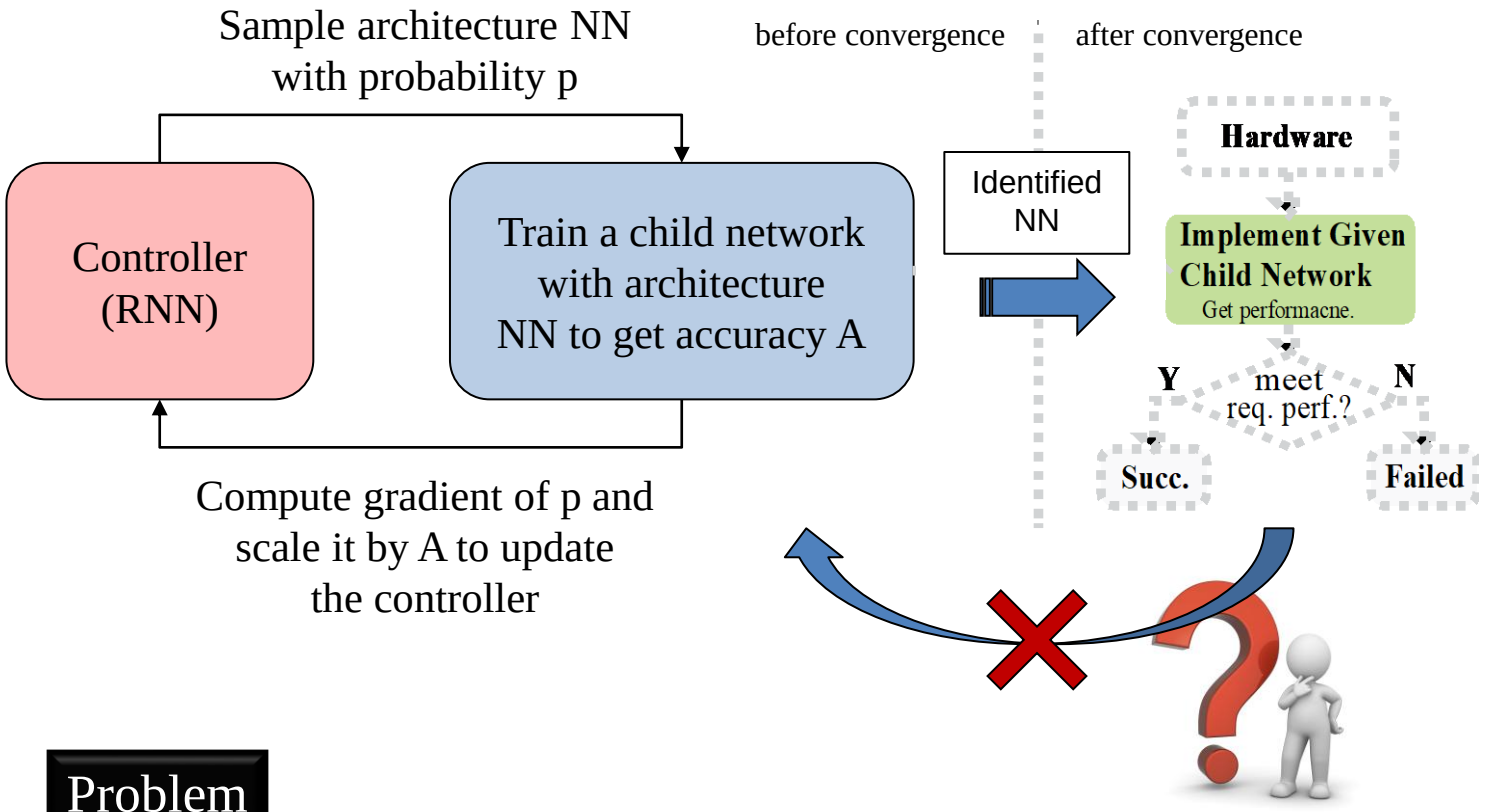


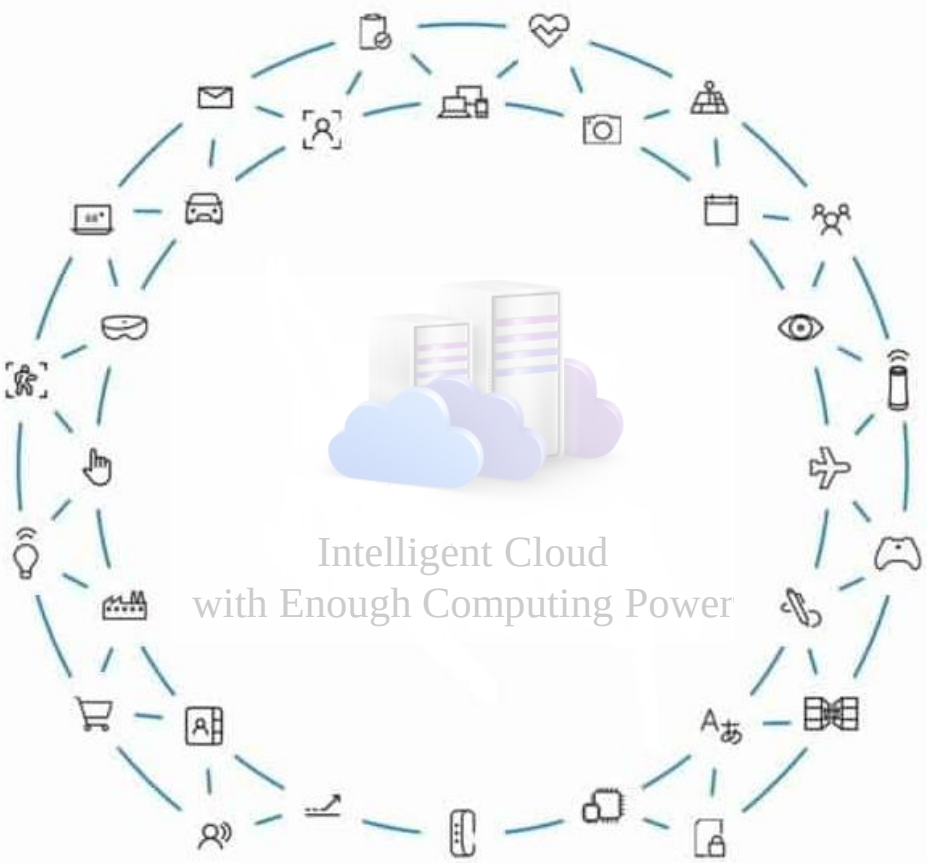
Era of AI Democratization

Problem

- **Domain knowledge and excessive labor**
- **It is impossible to manually design specific arch. for dedicated application in the era of AI democratization**

Phase II: Neural Architecture Search (NAS)



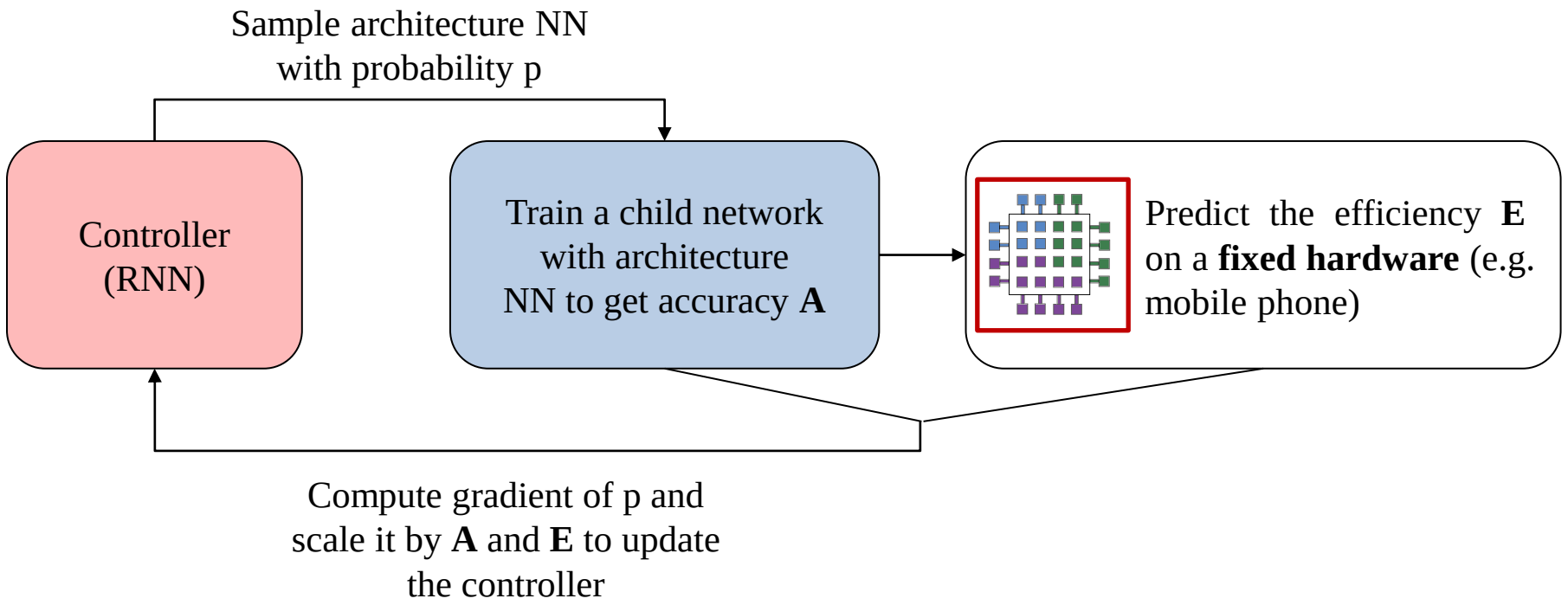


Problem

- **No constraint on hardware resource consumed**

REF: Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *ICLR 2017*

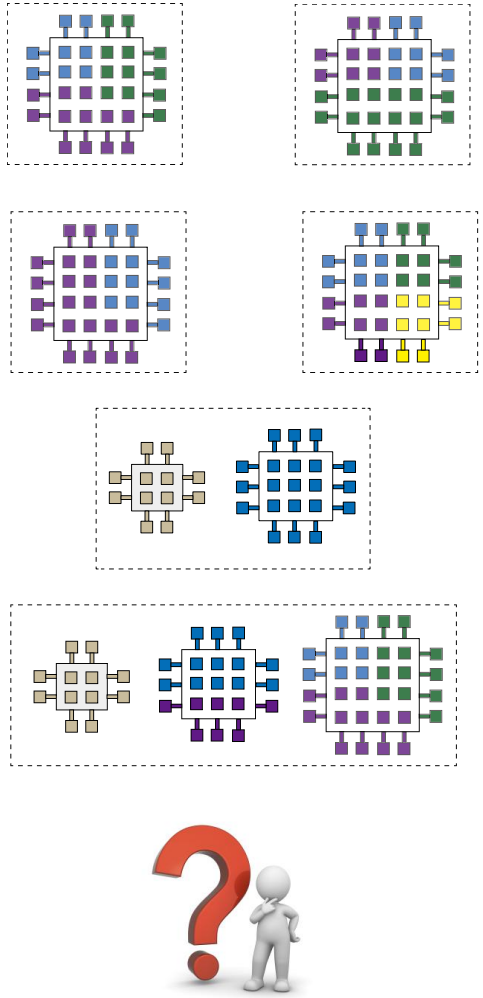
Phase III. Hardware-Aware NAS



Problem

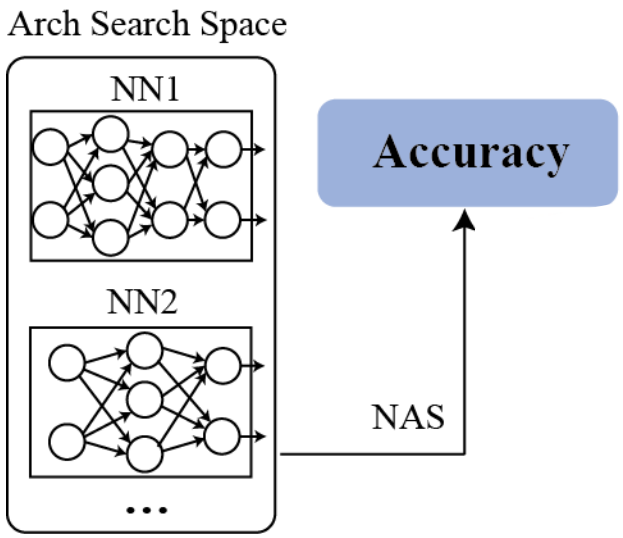
- **It works for particular fixed hardware, but not suitable for programmable hardware**

Different Hardware Designs

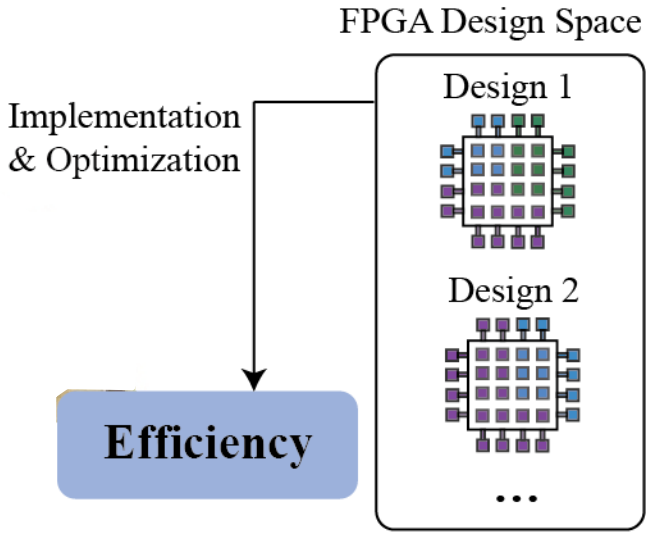


REF: Mingxing Tan, et al. "MnasNet: Platform-Aware Neural Architecture Search for Mobile." arXiv 2018

A Missing Link between Two Design Spaces

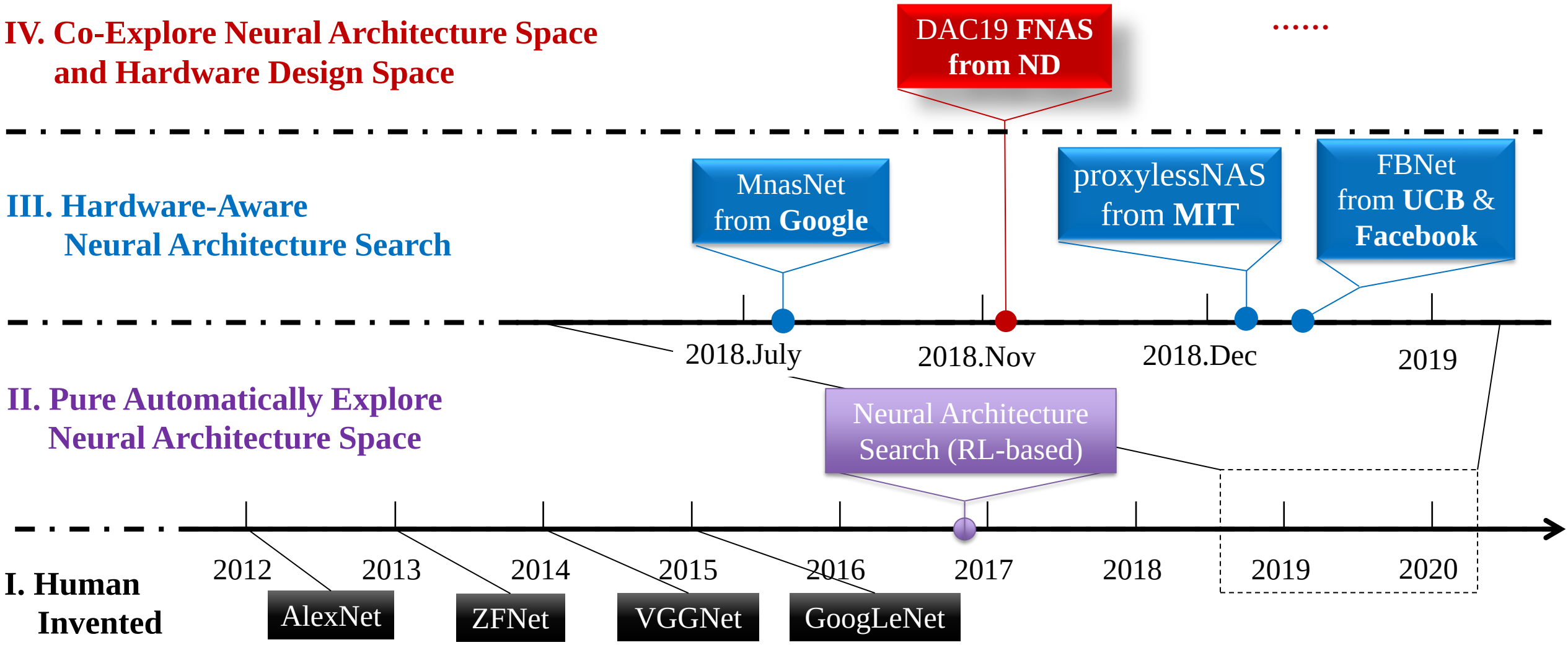


Neural Architecture Search



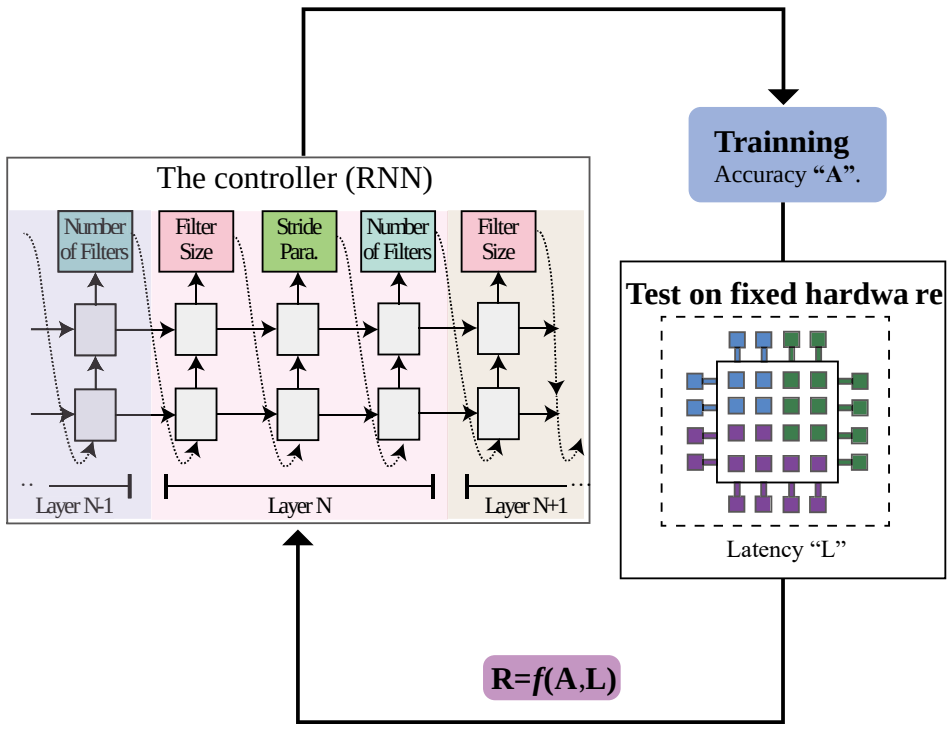
Neural Architecture Implementation on Hardware

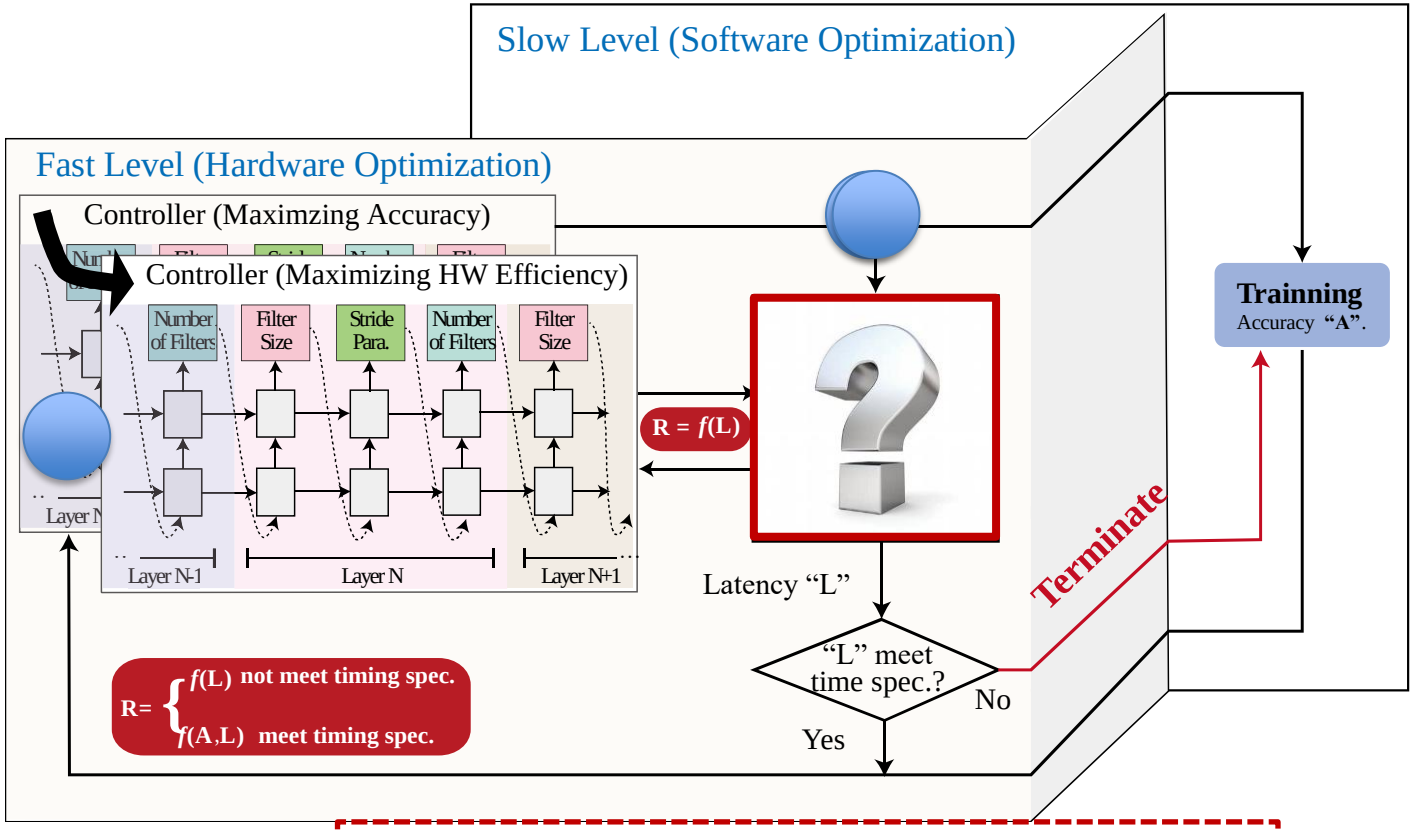
Evolution of Exploring Deep Neural Architectures



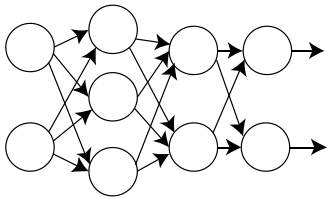
FNAS Framework

HW-Aware NAS vs. FPGA-Implementation Aware NAS（FNAS）



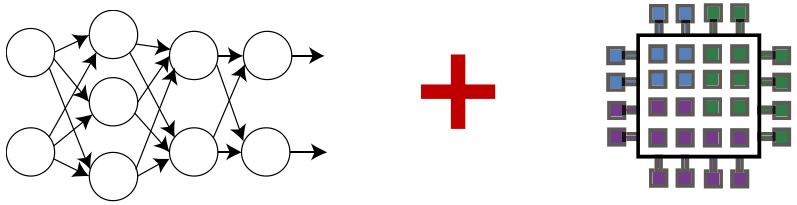


Output: A neural architecture



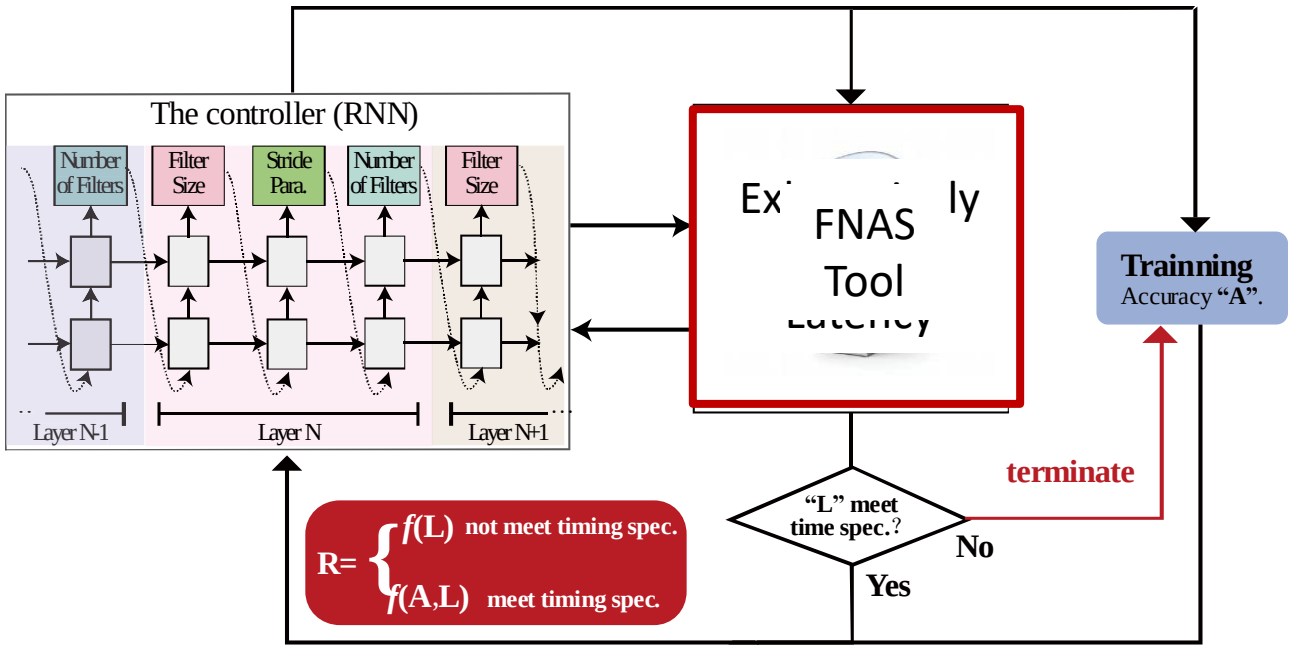
HW-Aware NAS

Output: A pair of neural architecture and hardware design



FNAS

Solutions & Challenges



Our Solution: FNAS tools to response to challenges

FNAS-Design "Design on Program Logic" **C1**	**FNAS-GG** "Tile-based Task Graph Generator" **C2**
FNAS-Sched "Scheduler on Processing System" **C2**	**FNAS-Analyzer** Estimate Performance " **L**" **C3**

Naïve Solution: HW-Aware + Exhaustively Evaluate Lat.

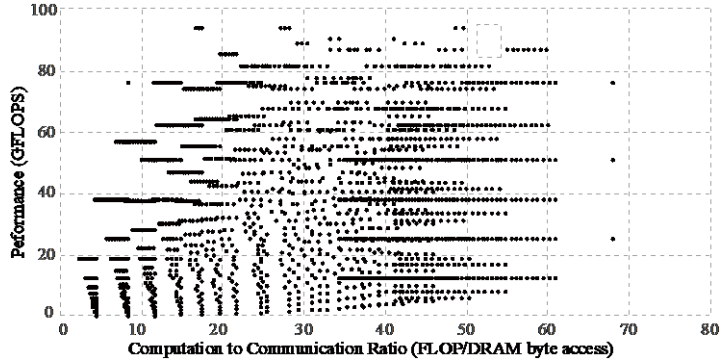


Fig1. Possible designs for Layer 5 of AlexNet on ZCU102

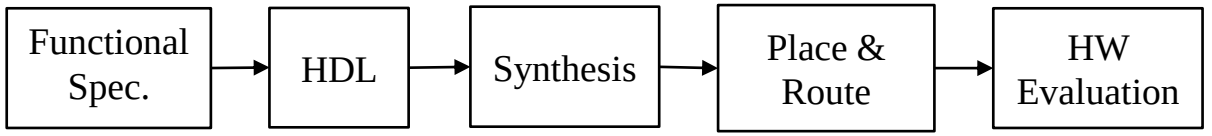


Fig2. Procedure of performance evaluation

Challenges:

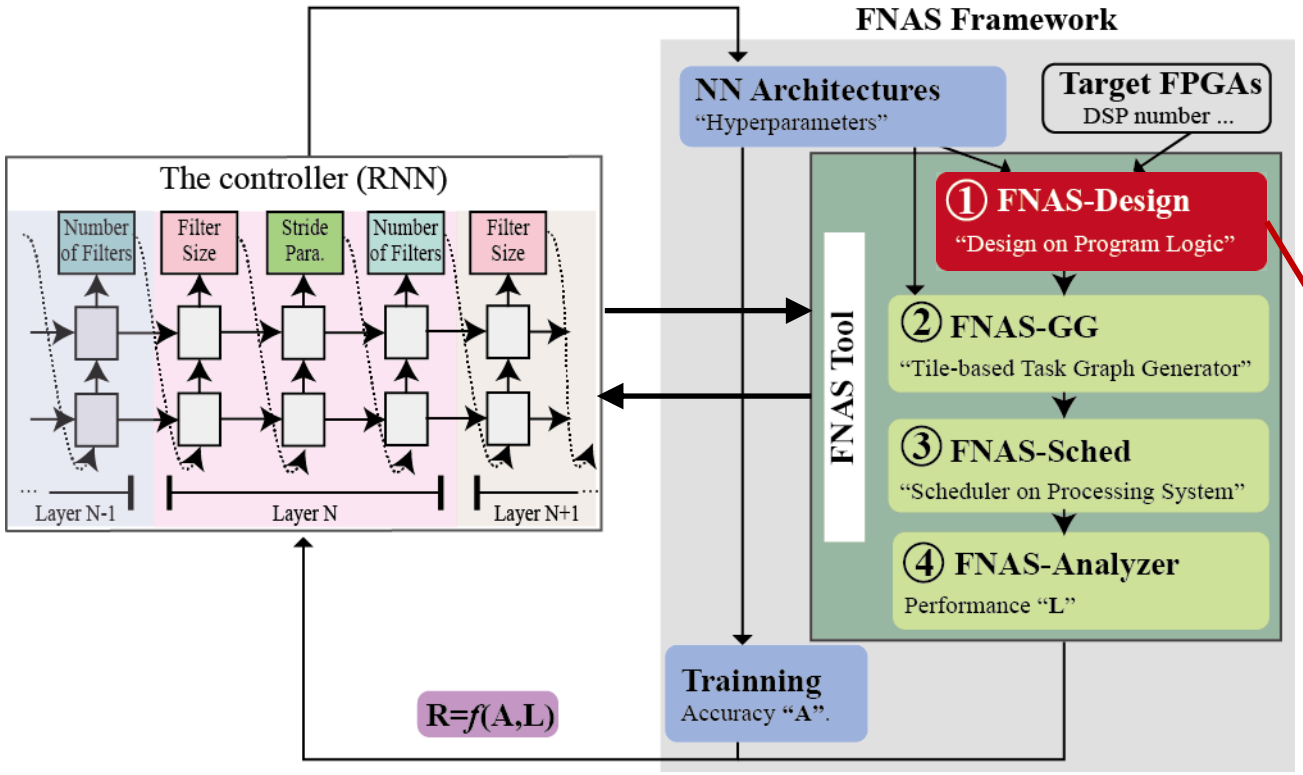
C1: Huge design space!

C2: Multi-FPGA design!

C3: Time-consuming evaluation!

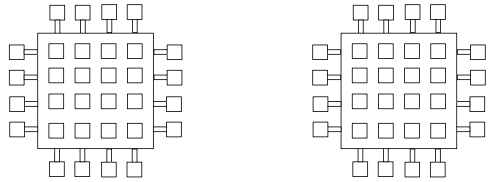
Infeasible

FNAS: Design Optimization (on-chip design)



Given:

1. FPGAs with attributes including DSPs, BRAM, etc.



2. A neural architecture with determined hyperparameters

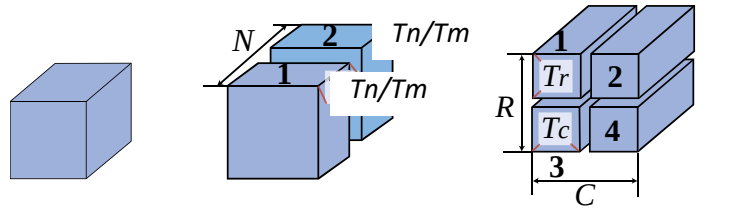
On-chip accelerator design:

Determine:

1. On-chip buffer allocation; 2. Accelerator size for computing

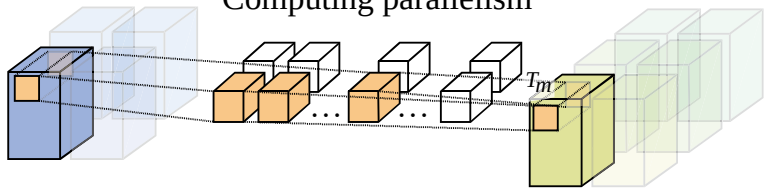
(note: both are determined by tiling parameters, Tm, Tn, Tr, Tc)

One layer:



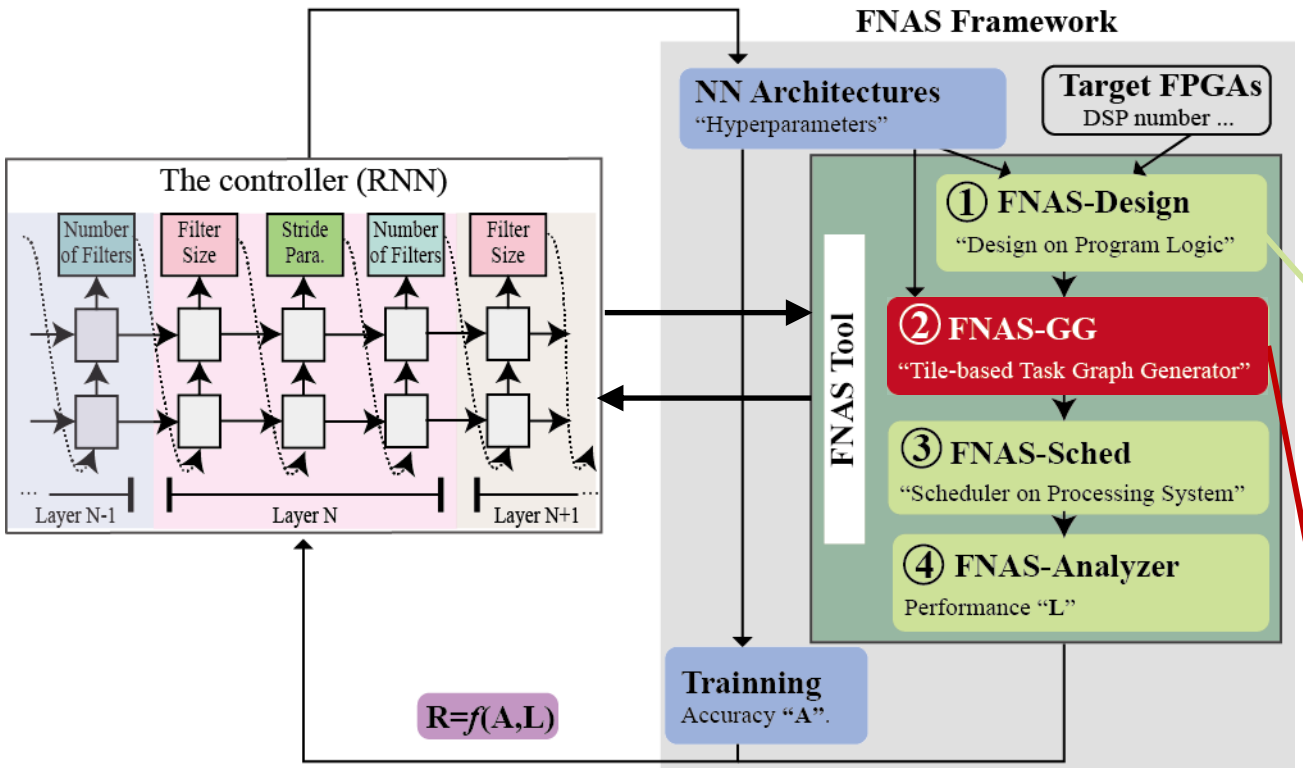
Computing parallelism

Multiple layers:



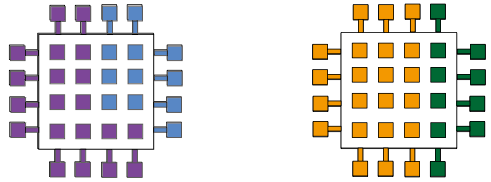
REF: Chen Zhang et al. 2015. Optimizing fpga-based accelerator design for deep convolutional neural networks. In Proc. of FPGA.

FNAS: Graph Generator



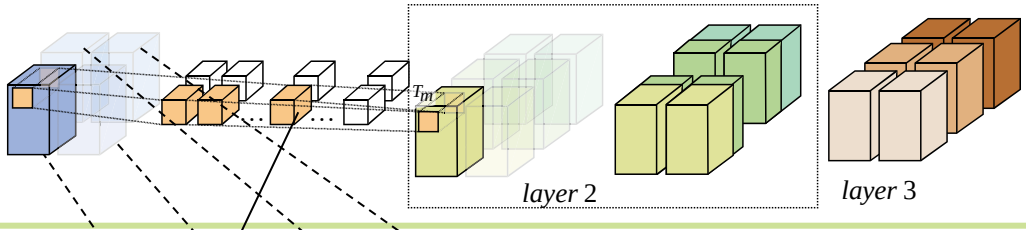
Given:

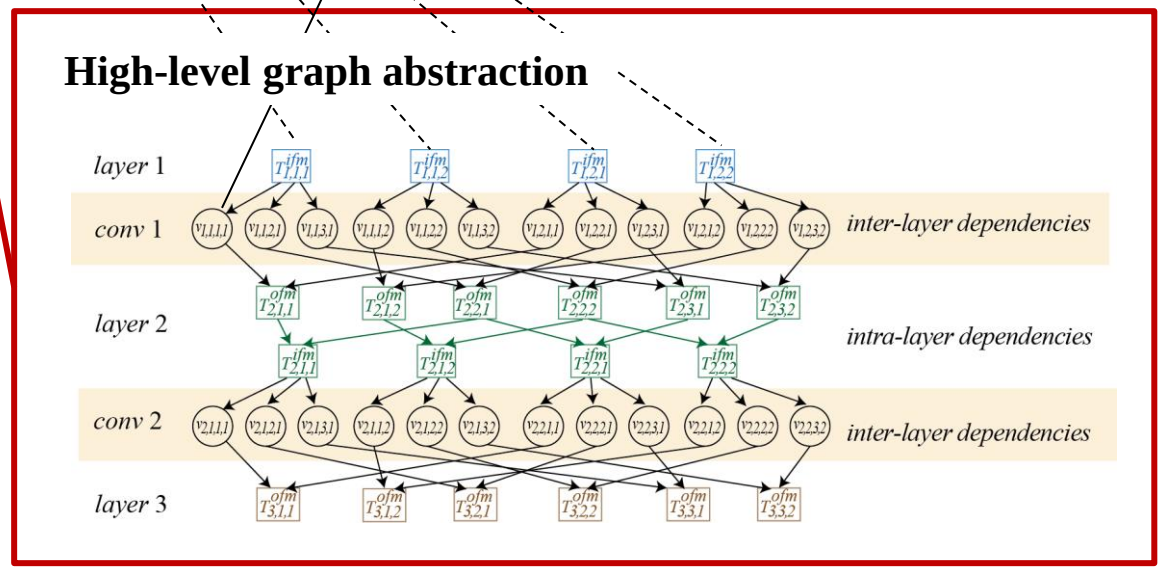
1. FPGAs with attributes including DSPs, BRAM, etc.



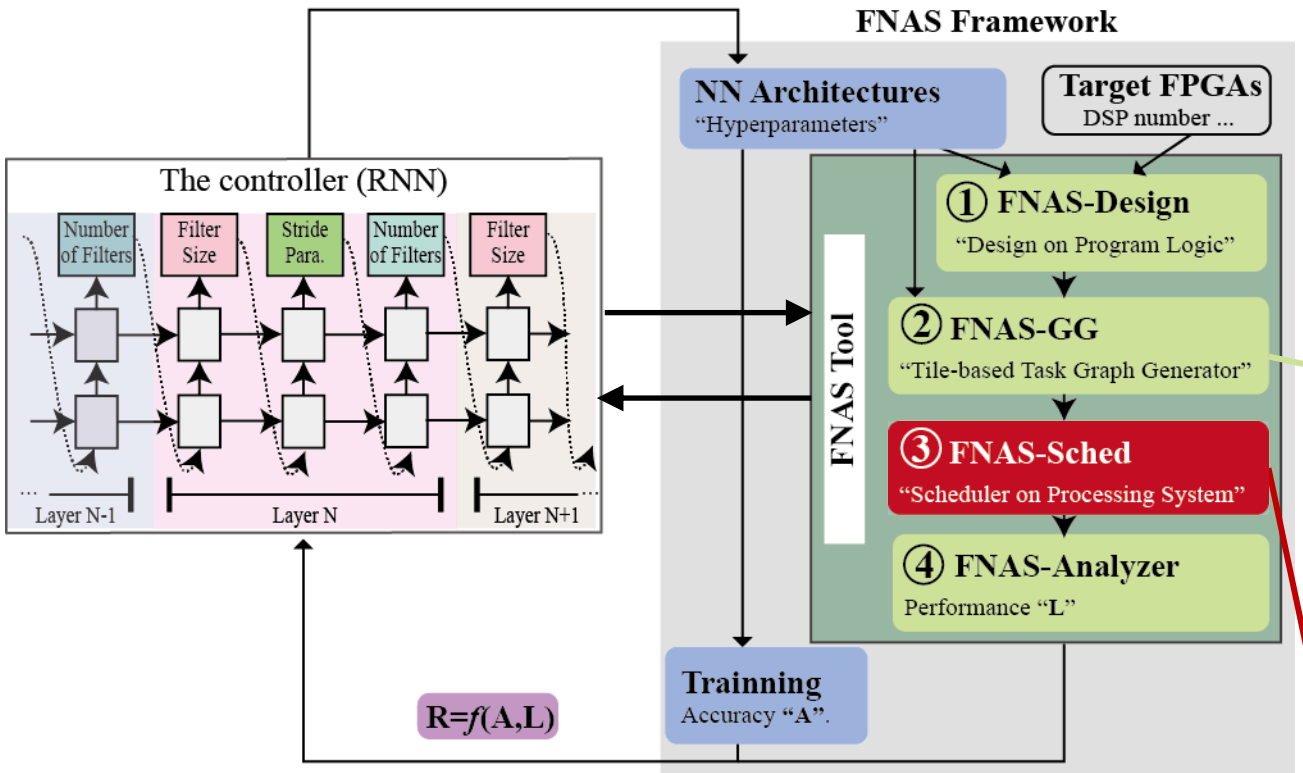
.....

2. A neural architecture with determined hyperparameters



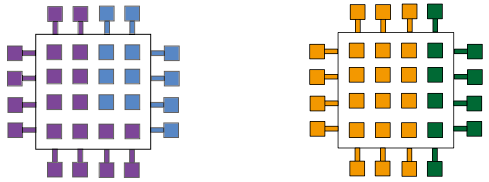


FNAS: Schedule (off-chip design)

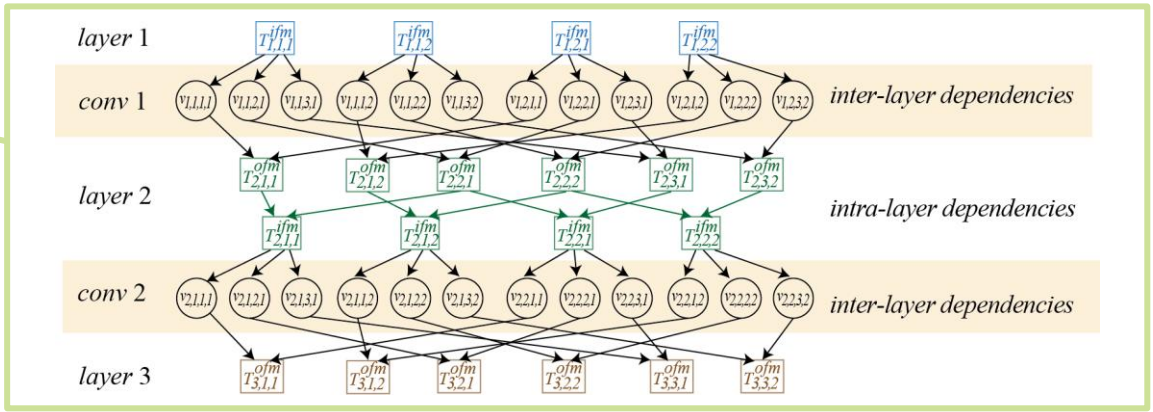


Given:

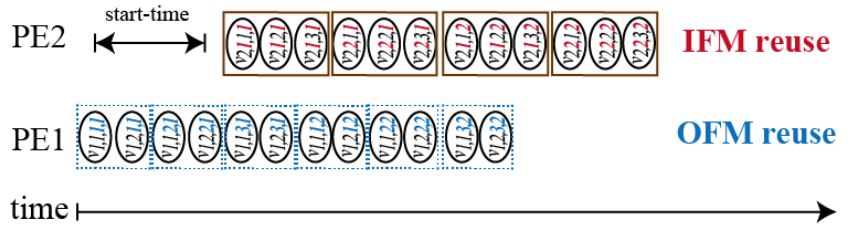
1. FPGAs with attributes including DSPs, BRAM, etc.



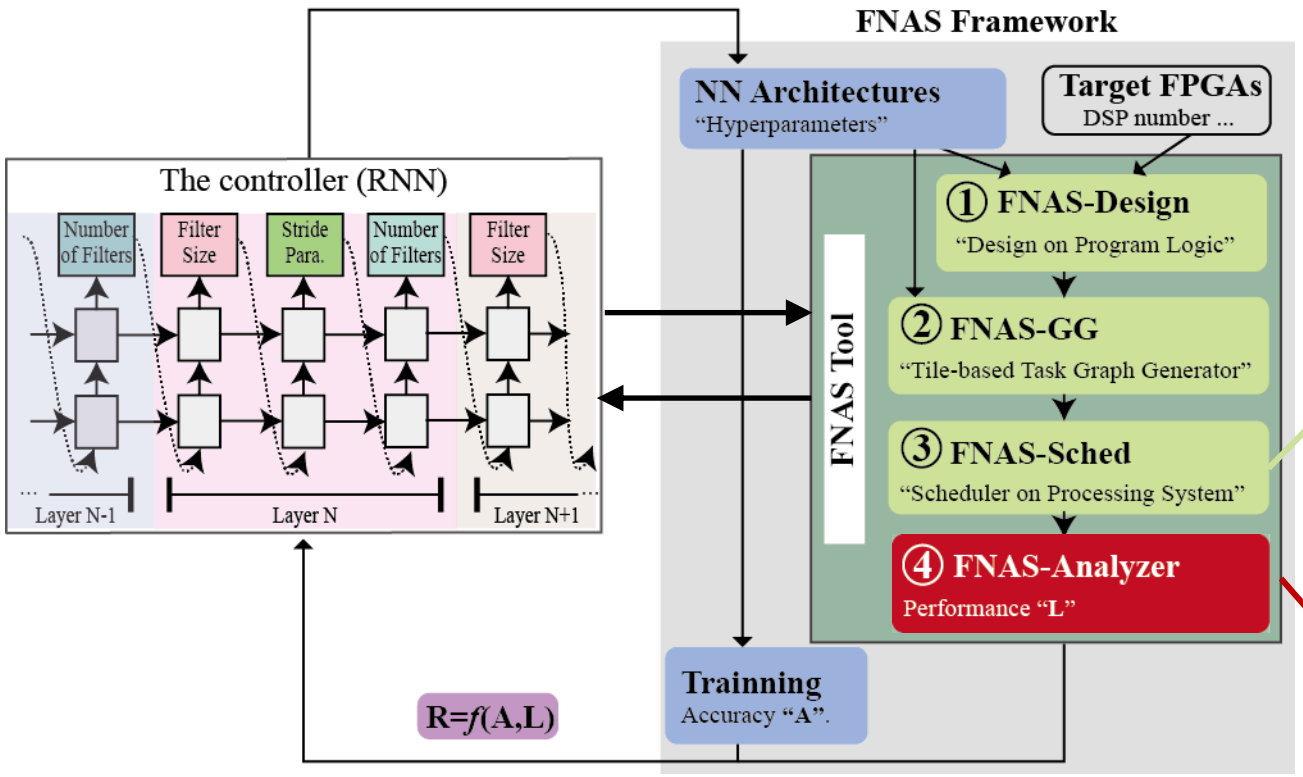
2. A neural architecture with determined hyperparameters



Schedule of tasks in graph on multiple FPGAs

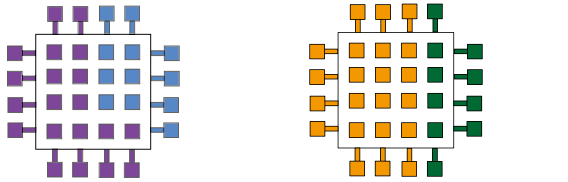


FNAS: Analyzer

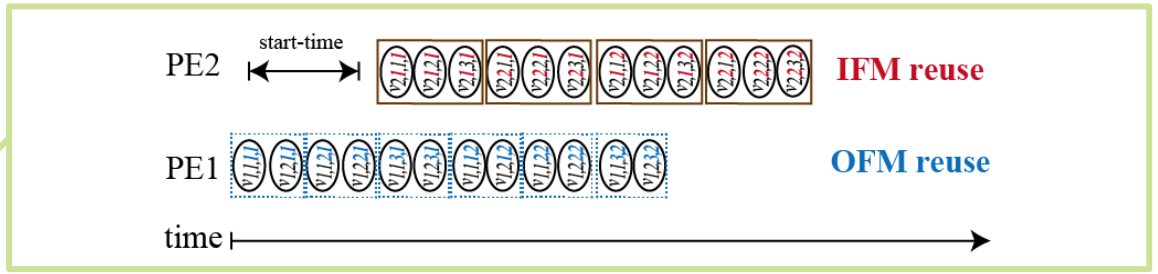


***Given*:**

1. FPGAs with attributes including DSPs, BRAM, etc.



2. A neural architecture with determined hyperparameters



Latency = pipeline start time + processing time

***Output*:**

1. A tailored FPGA Design
2. The system latency

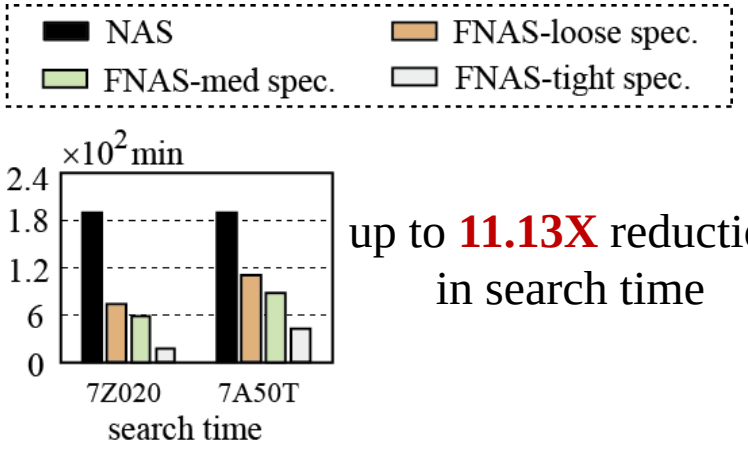
Experimental Results

Experimental Setting

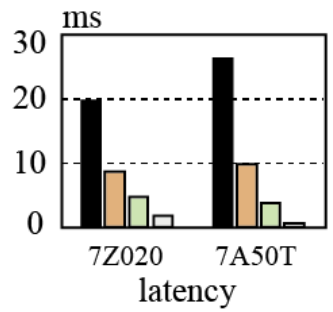
FPGAs		Xilinx 7A50T		Xilinx 7Z020
Datasets	 MNIST		 CIFAR-10	 ImageNet
NAS Search Space	Layer Num.	up to 5	up to 10	up to 15
	Filter Size	[5, 7, 14]	[1, 3, 5, 7]	[1, 3, 5, 7]
	Filter Num.	[9, 18, 36]	[24, 36, 48, 64]	[16, 32, 64, 128]
HW Search Space	Channel Tiling Para. (Tm,Tn); Row Tiling Para. (Tr); Col Tiling Para. (Tc); Schedule			
Timing Spec. (ms)		[2, 5, 10, 20]	[1.5, 2, 2.5, 10]	[2.5, 5, 7.5, 10]

Experimental Results

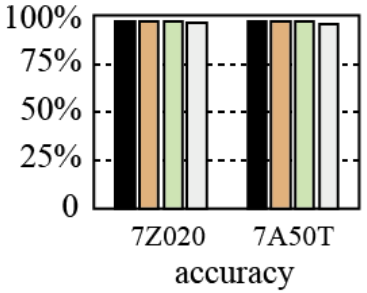
Different Hardware (MNIST)



up to **11.13X** reduction in search time

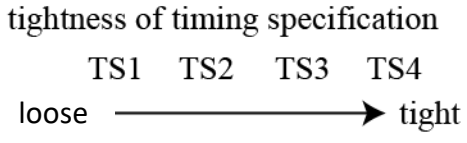


up to **7.81X** reduction in inference latency

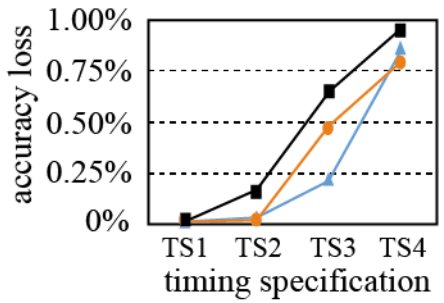


below **0.9%** loss in accuracy

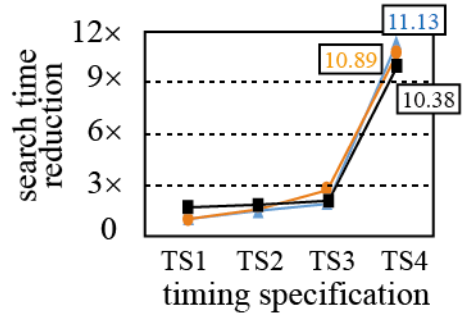
Different Datasets (7Z020)



Baseline: NAS

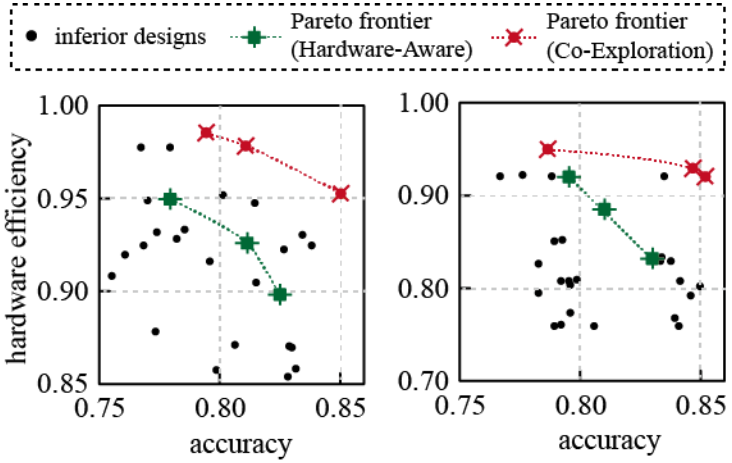


below **1%** loss in accuracy



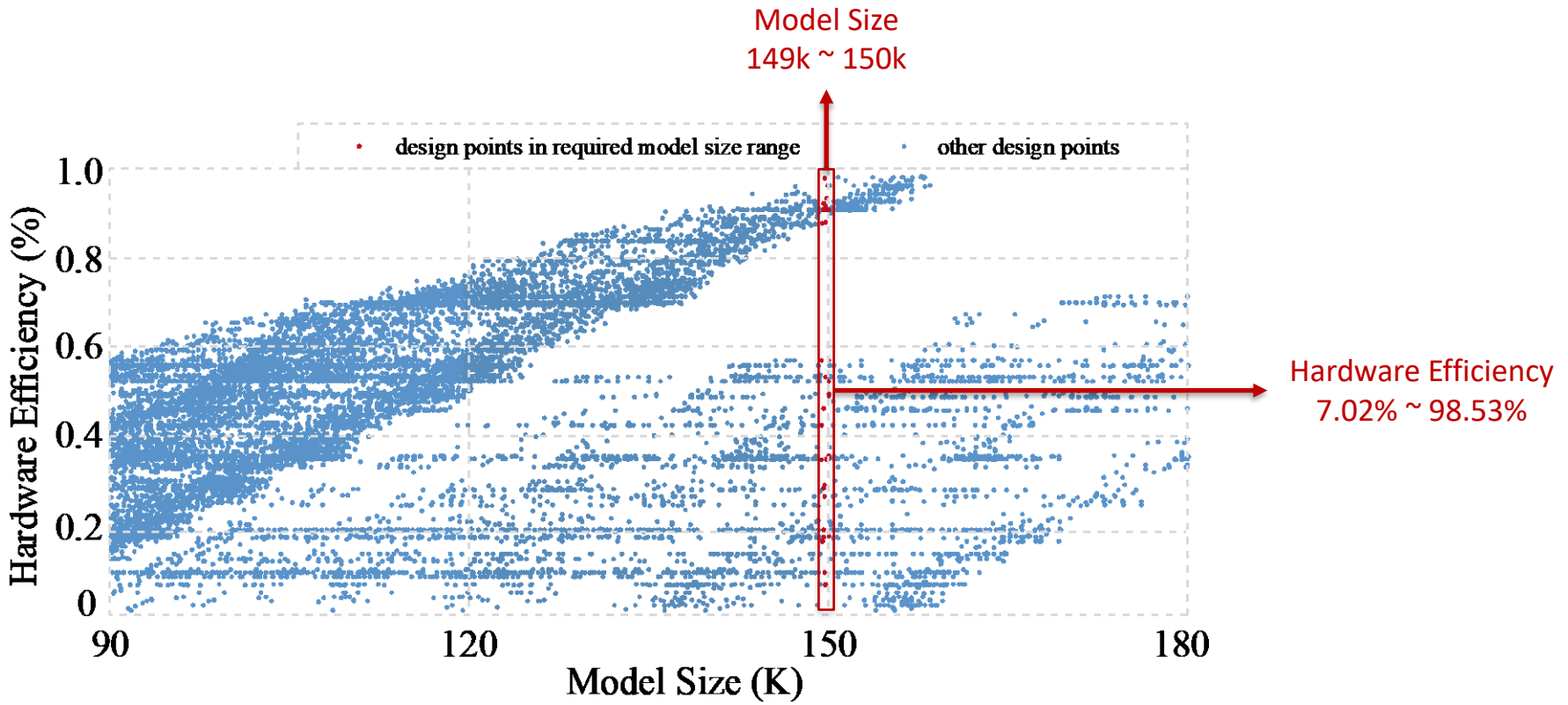
up to **10X** reduction in inference latency

Compare to HW-Aware NAS (CIFAR-10 + 7Z020)



FNAS can significantly **push forward** the Pareto frontiers between **accuracy and efficiency** tradeoff

Experimental Results: Importance of Co-Exploration



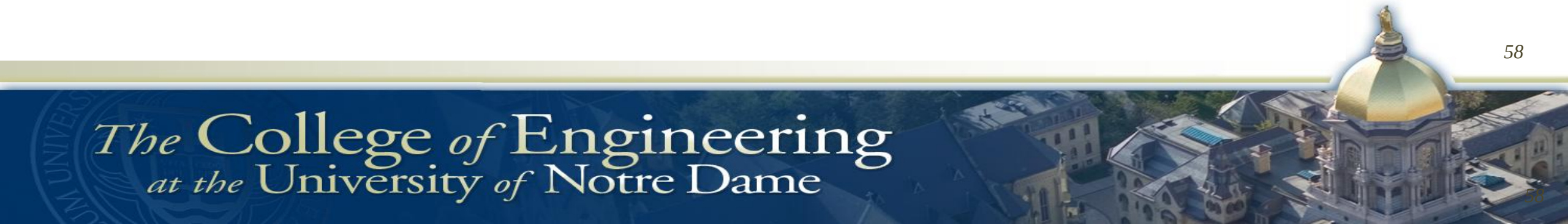
In the design space:

Models with similar model sizes may have distinct hardware efficiency

=> **Cannot restrict model size** to guarantee **hardware efficiency**

Outline

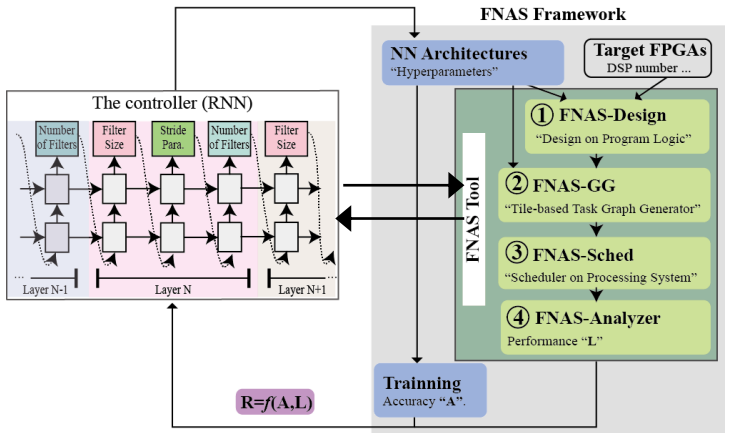
- Introduction
- XFER: Achieving Super-Linear Speedup across Multi-FPGA
  - CODES+ISSS'19 (Best Paper Finalist)
- Co-Exploration of Neural Architecture and FPGA Implementation
  - DAC'19 (Best Paper Finalist)
- **Other Research Directions**
- Conclusion



58

Our Work on Co-Exploring Neural Architecture and Hardware

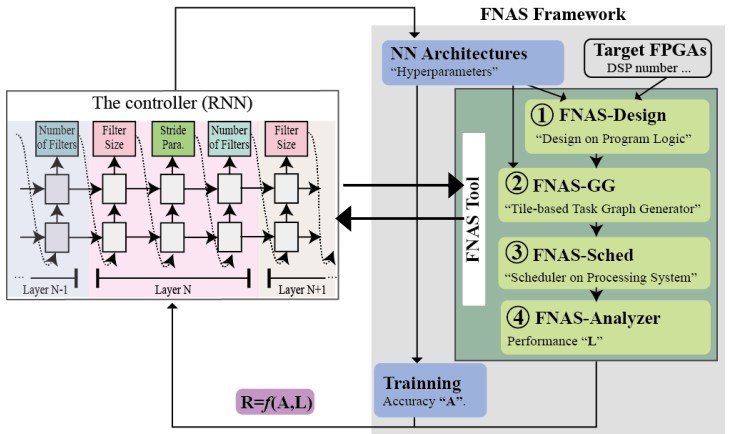
FPGA Implementation-Aware NAS (FNAS)



DAC'19 Best Paper Nomination

Our Work on Co-Exploring Neural Architecture and Hardware

FPGA Implementation-Aware NAS (FNAS)

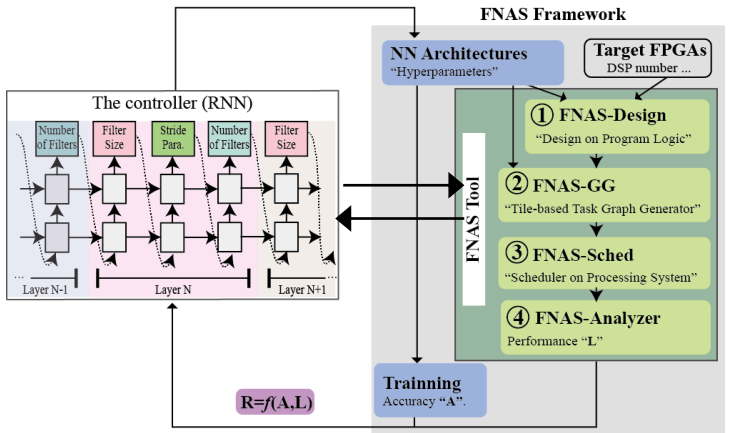


DAC'19 Best Paper Nomination

- **Scalability** (multiple-FPGAs, NoCs, etc.)
- **Limited Resource** (quantization, etc.)
- **Novel Computing Architectures** (in-memory Computing, NVM, etc.)
- ……

Design Neural Networks on Scalable Architecture

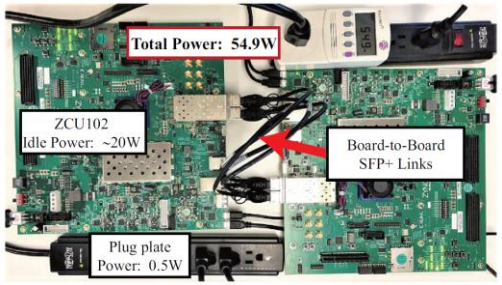
FPGA Implementation-Aware NAS (FNAS)



DAC'19 Best Paper Nomination

Scalability in Cloud Computing (multiple-FPGAs, NoCs, etc.)

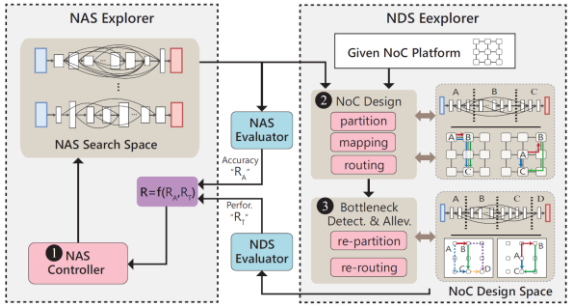
Super-Linear Speedup across Multiple FPGAs (XFER)



CODES+ISSS'19

Best Paper Nomination

Co-Explore Network-on-Chip Design and Neural Architectures (NANDS)

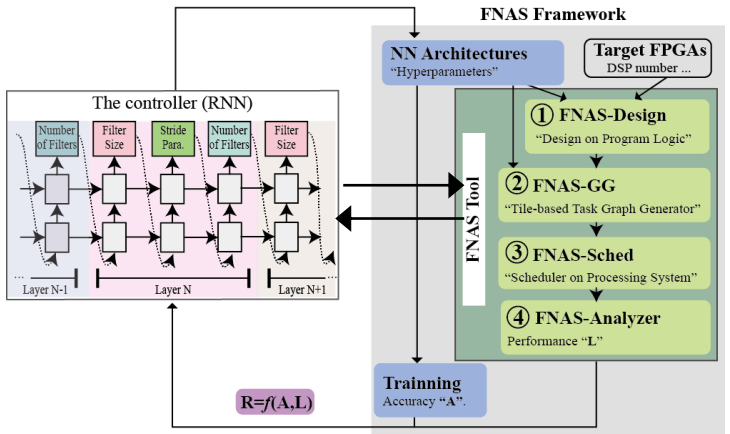


ASP-DAC'20

Best Paper Nomination

Design Neural Networks for Resource-Limited Device

FPGA Implementation-Aware NAS (FNAS)



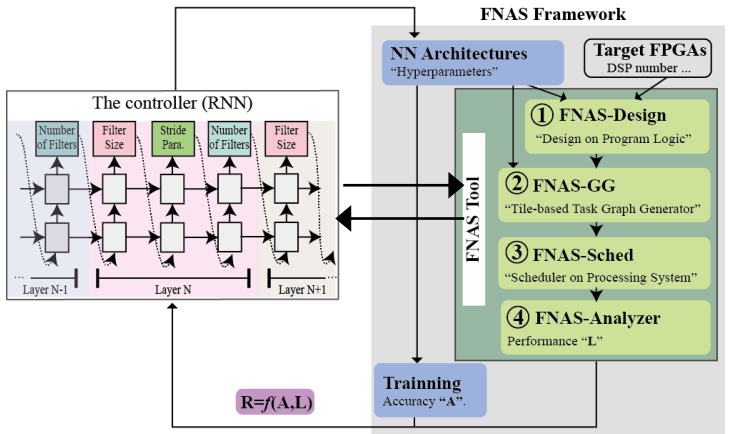
DAC'19 Best Paper Nomination

- **Scalability** in Cloud Computing (multiple-FPGAs, NoCs, etc.)
- **Limited Resource** in Edge Computing (quantization, etc.)
- **Novel Computing Architectures** (in-memory Computing, NVM, etc.)
- ……

62

Design Neural Networks for Resource-Limited Device

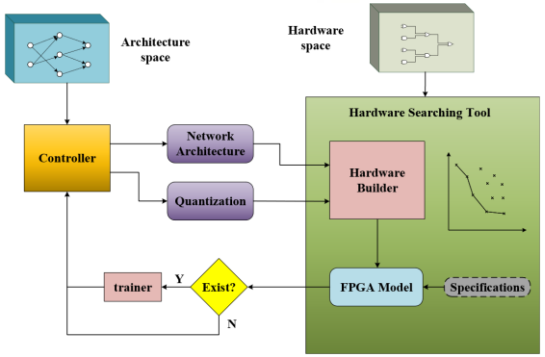
FPGA Implementation-Aware NAS (FNAS)



DAC'19 Best Paper Nomination

Limited Resource in Edge Computing (quantization, etc.)

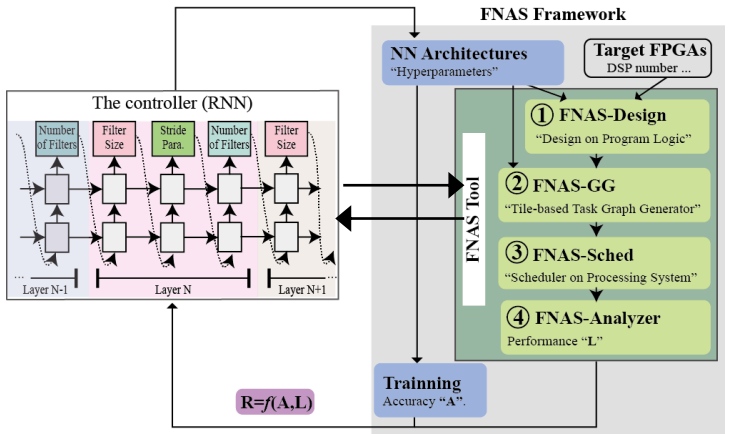
Co-Explore Quantization and Neural Architectures (QuanNAS)



ICCAD'19

Design Neural Networks for Novel Computing Architectures

FPGA Implementation-Aware NAS (FNAS)

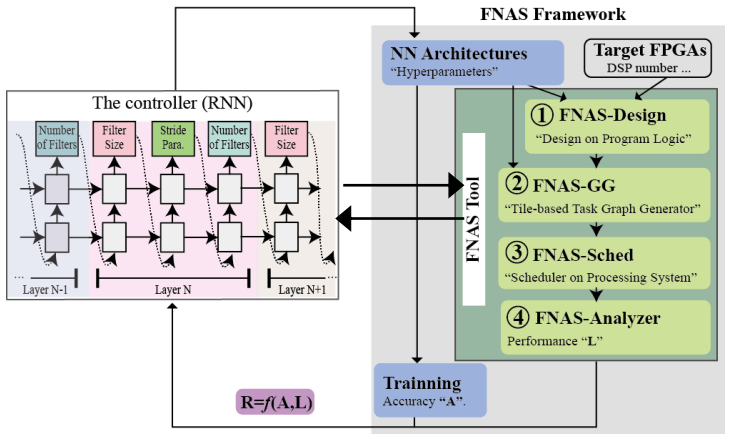


DAC'19 Best Paper Nomination

- **Scalability** in Cloud Computing (multiple-FPGAs, NoCs, etc.)
- **Limited Resource** in Edge Computing (quantization, etc.)
- **Novel Computing Architectures** (in-memory Computing, NVM, etc.)
- ……

Design Neural Networks for Novel Computing Architectures

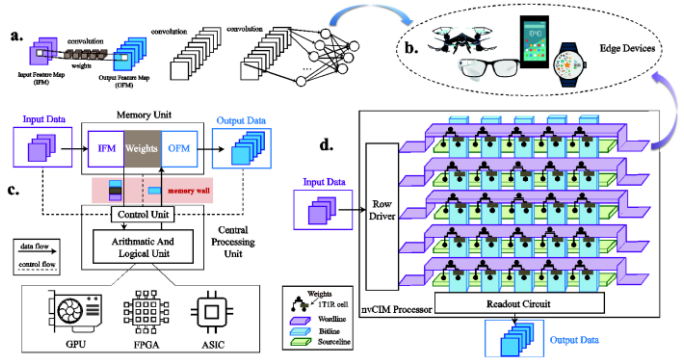
FPGA Implementation-Aware NAS (FNAS)



DAC'19 Best Paper Nomination

Novel Computing Architectures (in-memory Computing, NVM, etc.)

Integrating Memristors and CMOS for Better AI



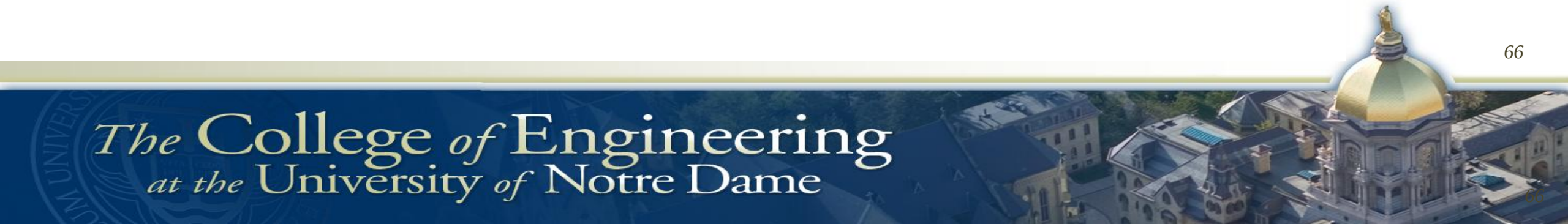
Nature Electronics'19

Co-Explore Neural Architectures and Devices

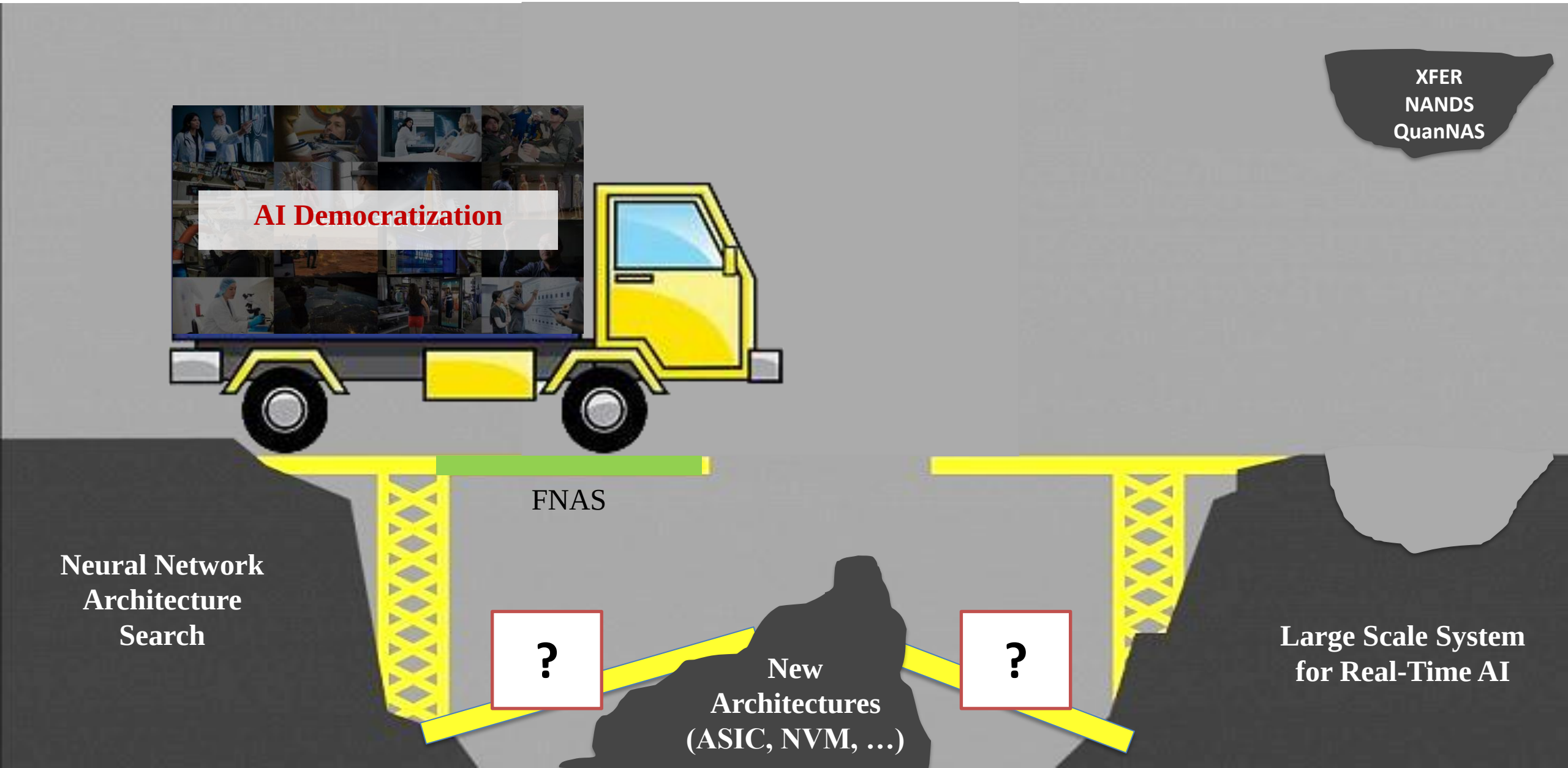
Ongoing Project

Outline

- Introduction
- XFER: Achieving Super-Linear Speedup across Multi-FPGA
  - CODES+ISSS'19 (Best Paper Finalist)
- Co-Exploration of Neural Architecture and FPGA Implementation
  - DAC'19 (Best Paper Finalist)
- Other Research Directions
- **Conclusion**



Conclusion and Future Work



Reference

[1] Weiwen Jiang, Xinyi Zhang, Edwin H.-M. Sha, Qingfeng Zhuge, Lei Yang, Yiyu Shi and Jingtong Hu, "Accuracy vs. Efficiency: Achieving Both through FPGA-Implementation Aware Neural Architecture Search," in Proc. of **DAC 2019**. **(Best Paper Finalist)**

[2] Weiwen Jiang, Edwin Sha, Xinyi Zhang, Lei Yang, Qingfeng Zhuge, Yiyu Shi and Jingtong Hu, "Achieving Super-Linear Speedup across Multi-FPGA for Real-Time DNN Inference," **CODES+ISSS 2019** and **ACM TECS** **(Best Paper Finalist)**

[3] Lei Yang, Weiwen Jiang, Weichen Liu, Edwin Sha, Yiyu Shi and Jingtong Hu, "Co-Exploring Neural Architecture and Network-on-Chip Design for Real-Time Artificial Intelligence," **ASP-DAC 2020** **(Best Paper Finalist)**

[5] Qing Lu, Weiwen Jiang, Xiaowei Xu, Yiyu Shi and Jingtong Hu, "On Neural Architecture Search for Resource-Constrained Hardware Platforms," in Proc. of **ICCAD 2019** (Invited Paper)

[6] Weiwen Jiang, Bike Xie, Chun-Chen Liu and Yiyu Shi, "Integrating Memristors and CMOS for Better AI," **Nature Electronics**, September 2019

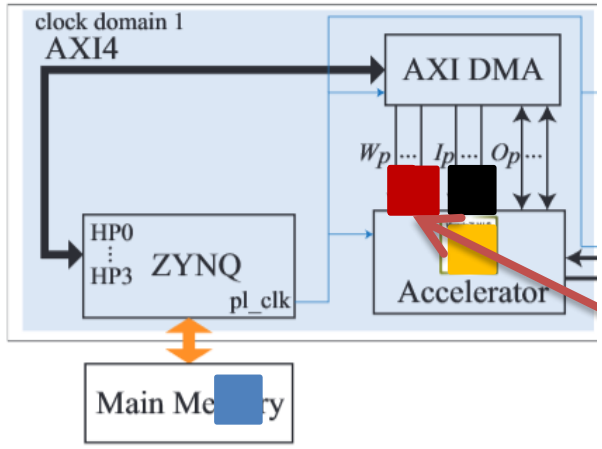
Thank You!

Weiwen Jiang
wjiang2@nd.edu
https://wjiang.nd.edu

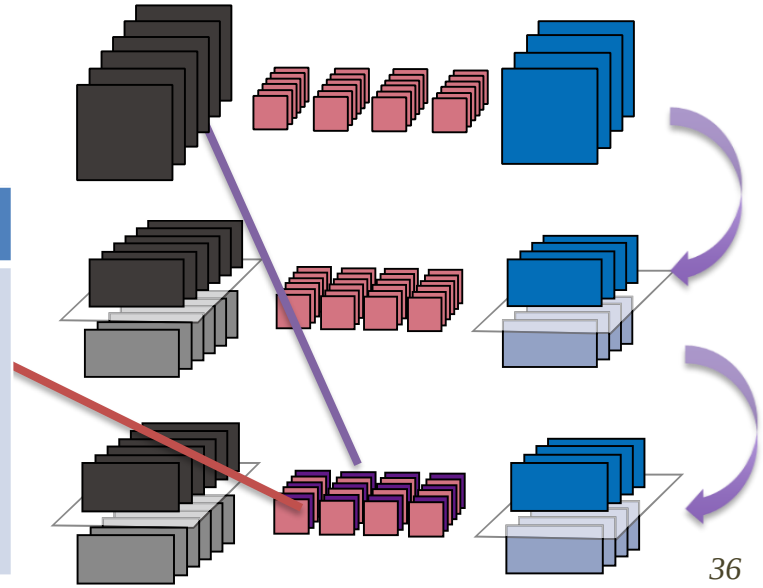
XFER: Achieve Super-Linear Speedup by Transferring Accesses

—— from Off-Chip Memory to Inter-FPGA Links

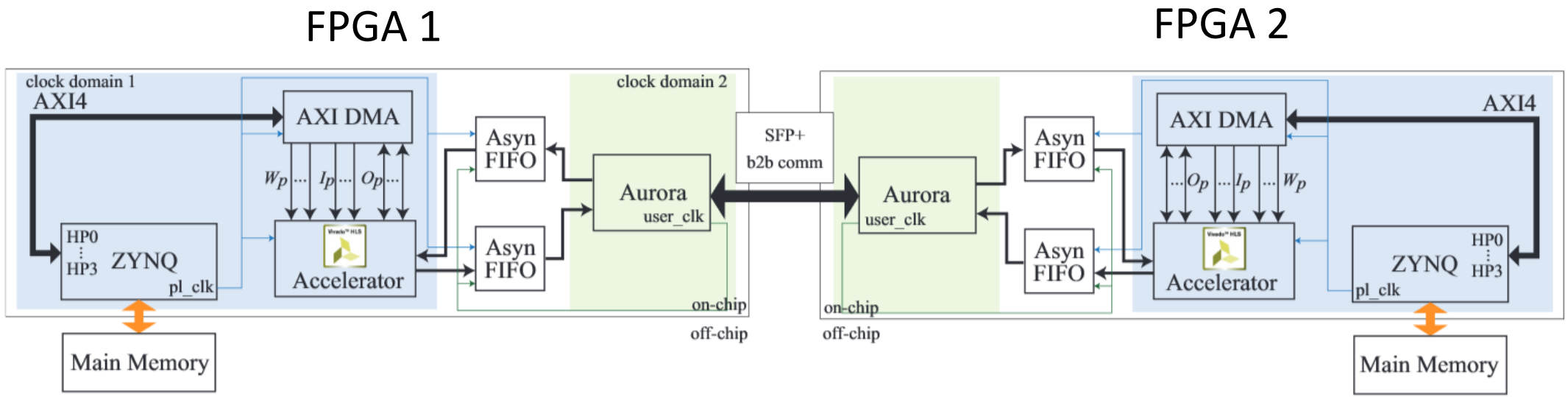
FPGA 1



NN	Operation	Single	w/o XFER	w/ XFER	Note
AlexNet Layer 5	Comm_IFM	2,612	1,412	1,412	Bottleneck Moves to Comp. 3.18X Speedup
	Comm_Weights	5,658	2,953	1,695	
	Comm_OFM	368	234	234	
	Computation	3,326	1,782	1,782	



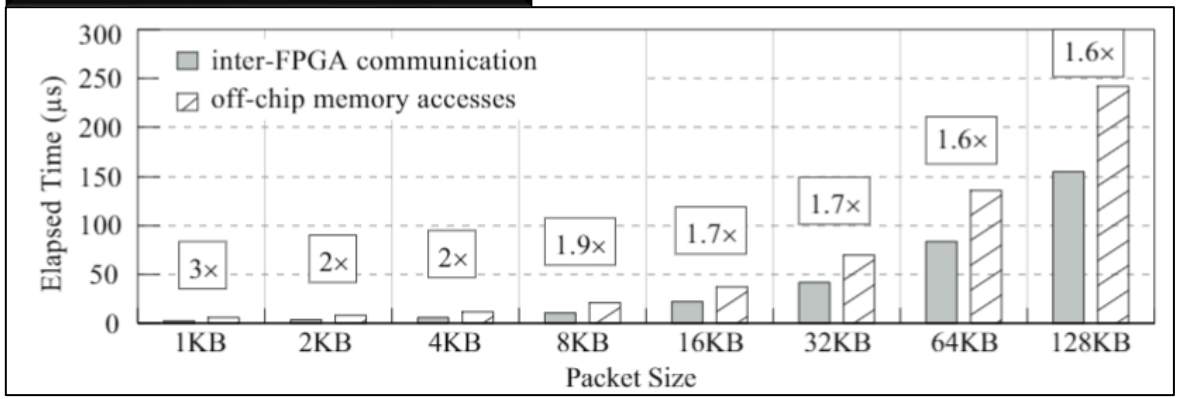
XFER: Architecture and Motivation



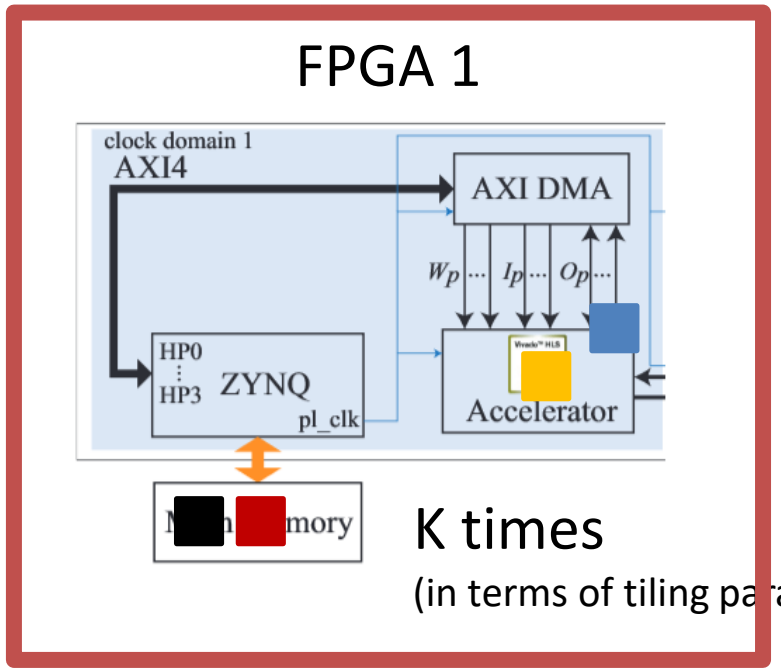
Feature

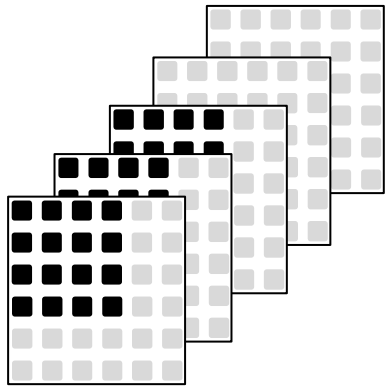
- Different **clock domains** for computation (**low**) and communication (**high**)
- Communication **not go through Off-Chip Memory**, but **directly switch between on-chip buffers**

Results & Motivations

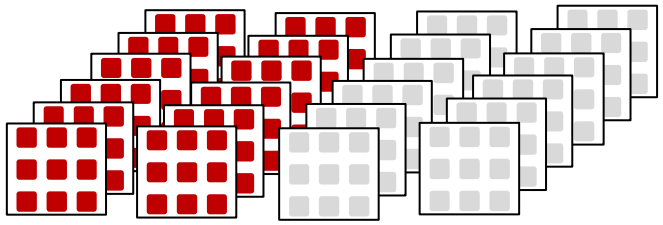


Performance on Single FPGA bounded by Limited Resource

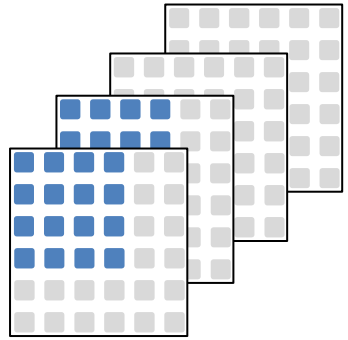




IFM



Weights

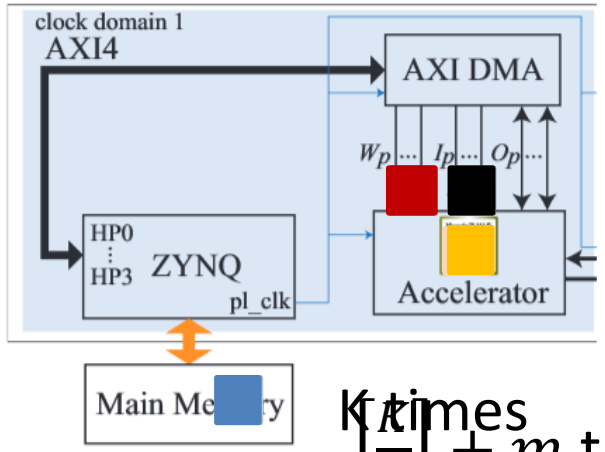


OFM

NN	Operation	Cycles	Note
AlexNet Layer 5	Comm_IFM	2,612	Performance Dominated by **Comm_Weights** **Latency is 5,658**
	Comm_Weights	**5,658**	
	Comm_OFM	368	
	Computation	3,326	

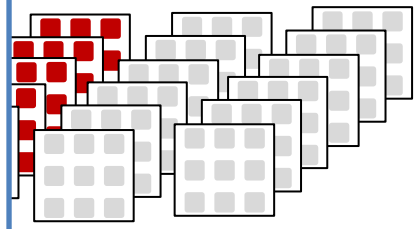
Double Computation Resource cannot Double Performance

FPGA 1

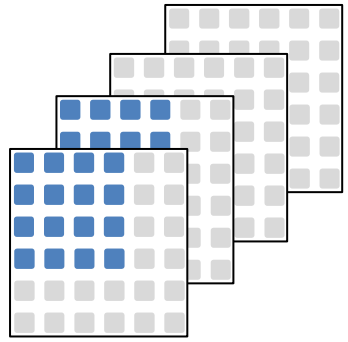


Observation:

1. Loading IFM, OFM, Computation are conducted **independently** on two FPGAs
2. Performance is still dominated by **loading weights**, and weights are **duplicated** on two FPGAs



Weights



OFM

IFM

K times

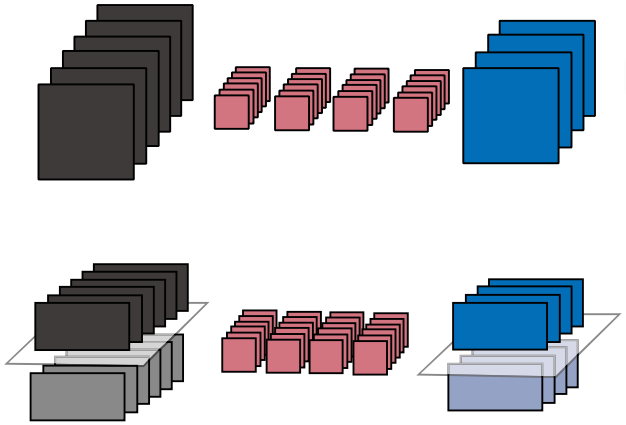
$\left\lfloor \frac{K}{2} \right\rfloor + m$ times

(in terms of tiling parameters)

(K is in terms of tiling parameters)

(m is in terms of stride and padding)

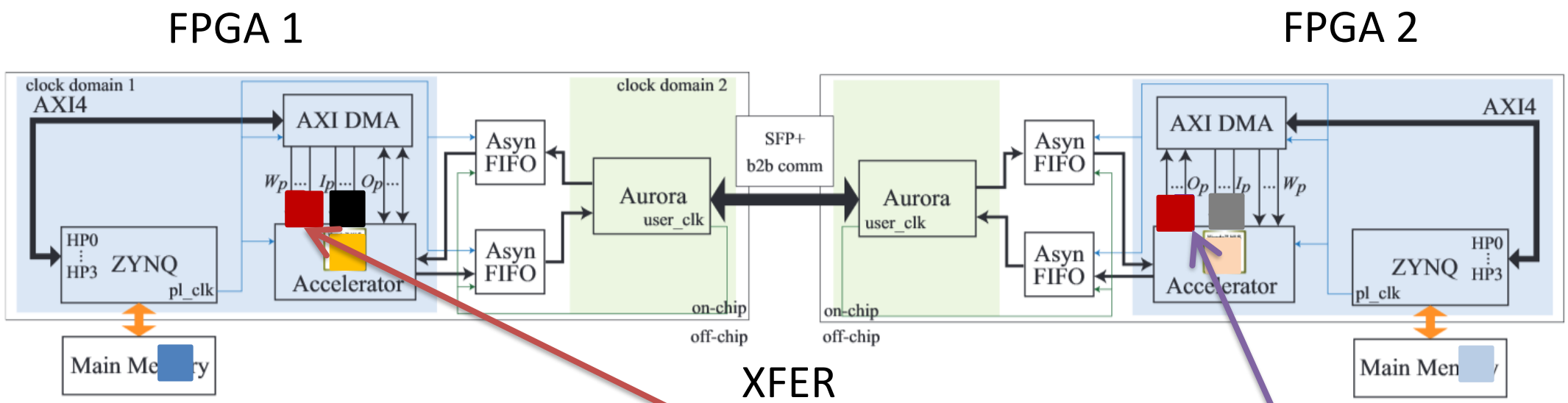
NN	Operation	Single	w/o XFER	Note
AlexNet Layer 5	Comm_IFM	2,612	1,412	**1.91X Speedup**
	Comm_Weights	**5,658**	**2,953**	
	Comm_OFM	368	234	
	Computation	3,326	1,782	



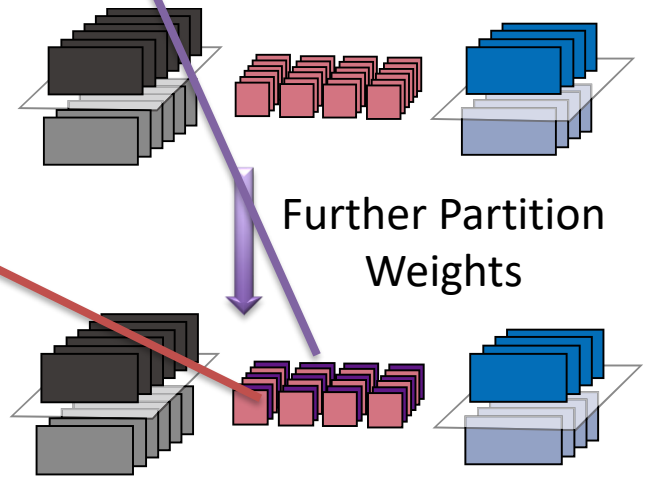
Partition for Parallel

XFER: Achieve Super-Linear Speedup by Transferring Accesses

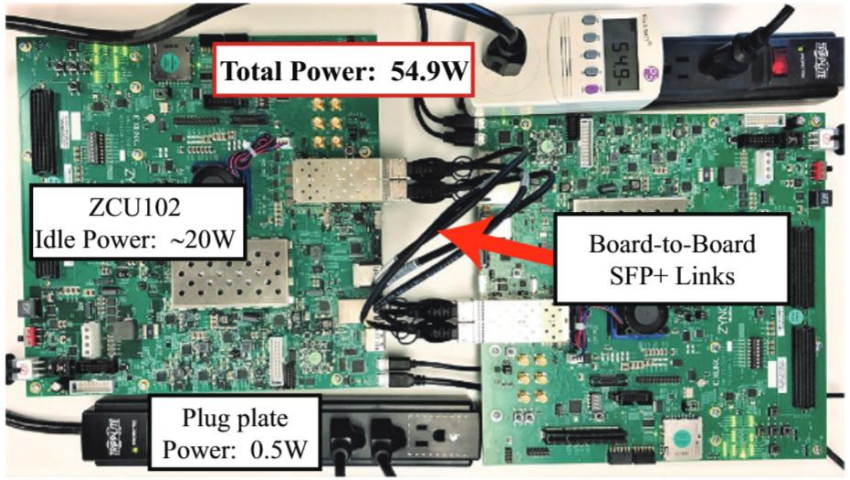
—— from Off-Chip Memory to Inter-FPGA Links



NN	Operation	Single	w/o XFER	w/ XFER	Note
AlexNet Layer 5	Comm_IFM	2,612	1,412	1,412	Bottleneck Moves to Comp. 3.18X Speedup
	Comm_Weights	**5,658**	**2,953**	1,695	
	Comm_OFM	368	234	234	
	Computation	3,326	1,782	**1,782**	



Experimental Results



Testbed:
Xilinx ZUC102 FPGAs connected by SFP+ Links

Comparison results of **XFER** with comparisons to **GPUs** and the **existing FPGA designs**

Design	mGPU		GPU		*FPGA15*		*ISCA17*		*ISLPED16*		**XFER**			
Precision	32bits float		32bits float		32bits float		32bits float		16bits fixed		32bits float		16bits fixed	
Device	Jetson TX2		Titan X		VX485T		VX485T		4×VX690t		2×ZCU102		2×ZCU102	
Freq (MHz)	1300MHz		1139MHz		100MHz		100MHz		150MHz		100MHz		200MHz	
Power (Watt)	16.00		162.00		18.61		-		126.00		52.40		54.40	
DSP Uti.	-		-		80%		80%		-		90.79%		55.87%	
BRAM Uti.	-		-		49.71%		43.25%		-		72.92%		92.43%	
Overall Perf.	Lat. *ms*	Thr. *GOPS*	Lat. *ms*	Thr. *GOPS*	Lat. *ms*	Thr. *GOPS*	Lat. *ms*	Thr. *GOPS*	Lat. *ms*	Thr. *GOPS*	Lat. *ms*	Thr. *GOPS*	Lat. *ms*	Thr. *GOPS*
	11.1 - 13.2	110.75	5.1 - 6.4	235.55	21.62	69.09	60.13	85.47	30.6	128.8	10.13	149.54	**2.27**	**679.04**
E.-E. (GOPS/W)	6.88		1.45		3.71		-		1.02		2.85		**12.48**	

Lowest Latency among all competitors

Experimental Results

Comparison results of **XFER** with SOTA single FPGA design

Design	32bits float				16bits fixed			
	FPGA15		**Super-LIP**		*FPGA15*		**Super-LIP**	
$\langle T_m, T_n \rangle$	⟨64, 7⟩		⟨64, 7⟩		⟨64, 24⟩		⟨128, 10⟩	
Power (W)	25.70 (1 FPGA)		**52.40 (2 FPGAs)**		26.00 (1 FPGA)		**54.40 (2 FPGAs)**	
Perf.	Lat. ms	Thr. GOPS	**Lat. ms**	**Thr. GOPS**	Lat. ms	Thr. GOPS	**Lat. ms**	**Thr. GOPS**
conv1	7.36	28.6	**3.66**	**57.6**	3.74	56.5	**0.94**	**224.5**
conv2	5.20	86.1	**2.55**	**175.5**	1.48	302.6	**0.48**	**933.1**
conv3	4.50	66.4	**1.73**	**172.7**	1.20	249.6	**0.33**	**906.2**
conv4	3.41	65.7	**1.31**	**171.0**	0.89	252.6	**0.35**	**640.8**
conv5	2.28	66.0	**0.88**	**170.9**	0.59	251.7	**0.17**	**879.5**
overall	22.75	66.6	**10.13**	**149.5**	7.90	195.1	**2.27**	**679.0**
Perf. Impr.	1.00×		**2.25×**		1.00×		**3.48×**	
E.-E. (GOPS/W)	2.59		**2.85**		7.51		**12.48**	
E.-E. Impr.	-		**9.21%**		-		**39.86%**	

Achieve Super-Linear Speedup

for both 32-bits and 16-bits implementations

QuanNAS: Co-Explore Quantization and Neural Architectures

Search Space

Parameter	Symbol	Value
# filters	N	(24, 36, 48, 64)
filter height	Fh	(1, 3, 5, 7)
filter width	Fw	(1, 3, 5, 7)
stride height	Sh	(1, 2, 3)
stride width	Sw	(1, 2, 3)
pooling size	Ps	(1, 2)
activation integer bits	Ai	(0, 1, 2, 3)
activation fractional bits	Af	(0, 1, 2, 3, 4, 5, 6)
weight integer bits	Wi	(0, 1, 2, 3)
weight fractional bits	Wf	(0, 1, 2, 3, 4, 5, 6)

Besides filter size and shape, **we co-explore bit width of weights and activation data, for FPGAs and ASICs.**

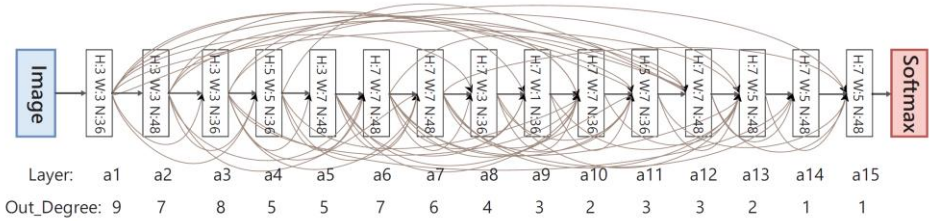
Results

Design	rL	rT	Acc w/o Quan	Acc w/ Quan	#LUTs	FPS	para size (kbits)
A_1-d_1	100,000	500	87.76%	80.23%	99,871	556	1,867
A_1-d_2	100,000	1000	87.76%	25.79%	99,848	1157	1,189
B_1-d_1	100,000	500	89.71%	87.64%	96,904	512	3,463
B_1-d_2	100,000	1000	89.71%	64.35%	98,752	1020	2,784
B_1-d_3	300,000	2000	89.71%	50.93%	285,441	2083	2,835
D	30,000	1000	83.65%	82.98%	29,904	1293	457
E	100,000	1000	86.99%	82.76%	94,496	1042	1,923
F	300,000	2000	87.03%	84.92%	299,860	2089	1,217

- A,B are first explored by NAS, then to be quantized. In this way, the **accuracy loss** by quantization is huge.
- D,E,F are explored by our Co-Exploration framework, which output **better accuracy** for quantized network with **smaller parameter size**

NANDS: Co-Explore NoC Design and Neural Architectures

Why NoC? Complicated Structure from NAS

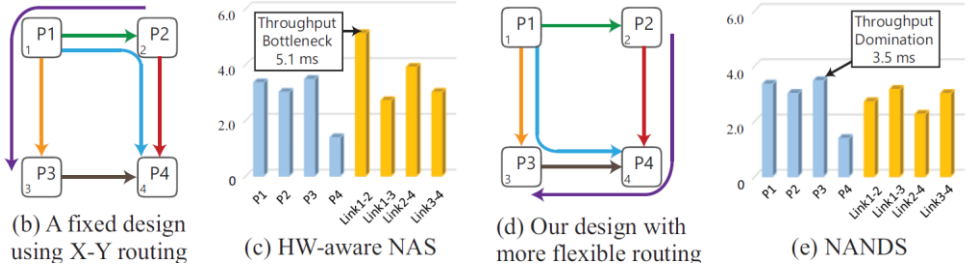


(a) The neural network architecture found by NAS

Operation Time \ Platforms	Single Processing Element	4 Processing Elements	
		Bus Interconnection	2-D Mesh NoC
Computation (ms)	12.4	3.4	3.4
Data transmission (ms)	—	14.7	6.2

(b) The timing performance of network implementations on different platforms

Fixed design (X-Y routing) leads lower performance



(b) A fixed design using X-Y routing

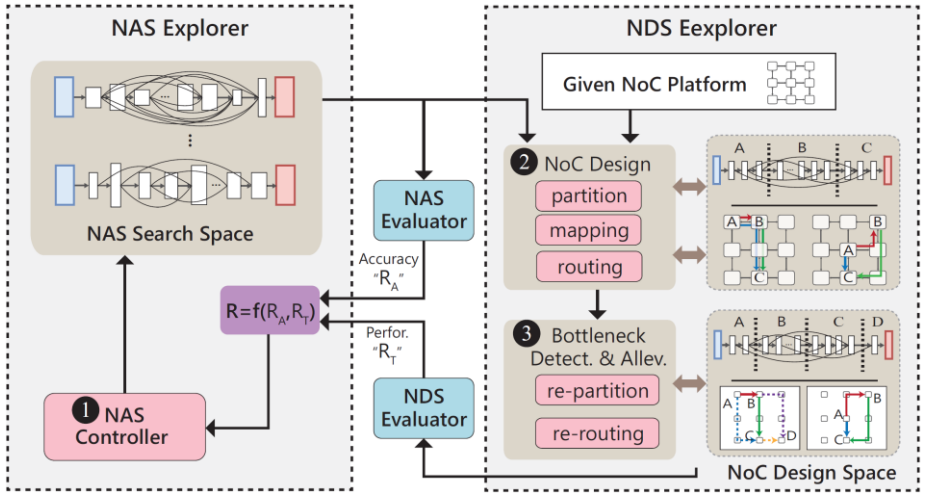
(c) HW-aware NAS

(d) Our design with more flexible routing

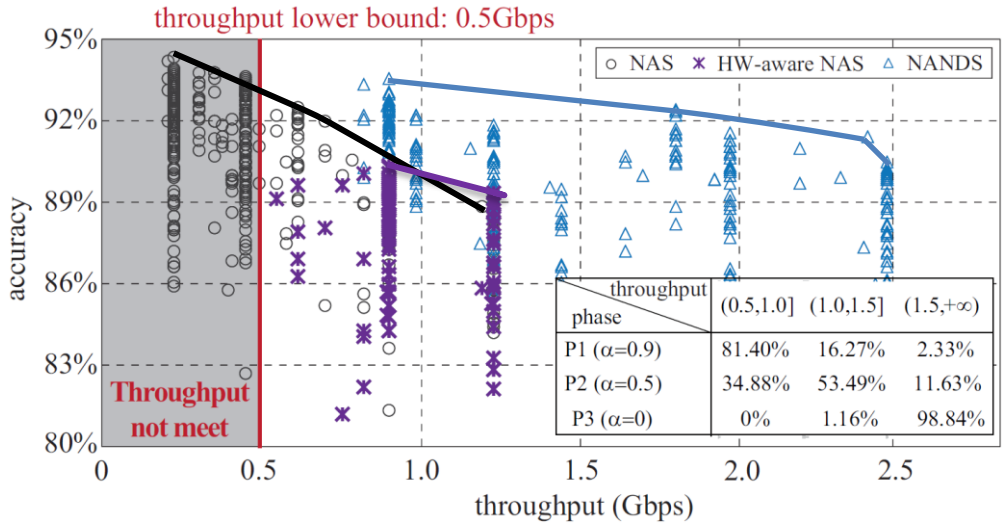
(e) NANDS

31.37% Improvement

Co-Exploration Framework



Results



phase \ throughput	(0.5,1.0]	(1.0,1.5]	(1.5,+∞)
P1 (α=0.9)	81.40%	16.27%	2.33%
P2 (α=0.5)	34.88%	53.49%	11.63%
P3 (α=0)	0%	1.16%	98.84%