Agile Hardware and Software Co-design for RISC-V-based Multi-precision Deep Learning Microprocessor

Zicheng He[1,2], Ao Shen[1], Qiufeng Li[1], Quan Cheng[1] and Hao Yu[1]

SUSTech[1]
Speaker: Qiufeng Li[1]

UCLA[2]
Supervisor: Hao Yu[1]

Southern University of Science and Technology

Outline

1

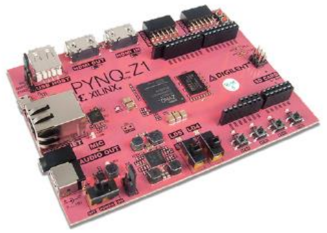
Introduction

Edge computing and future

Introduction: Edge Computing

1. Edge computing vs cloud computing？

2. How to deploy?

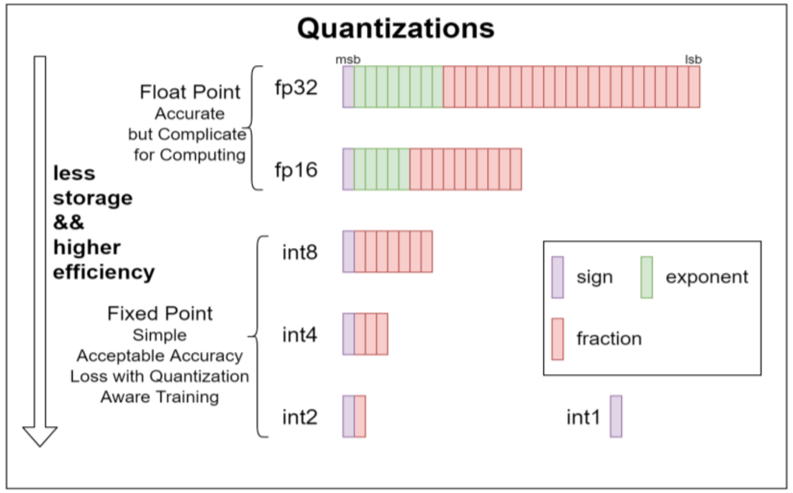


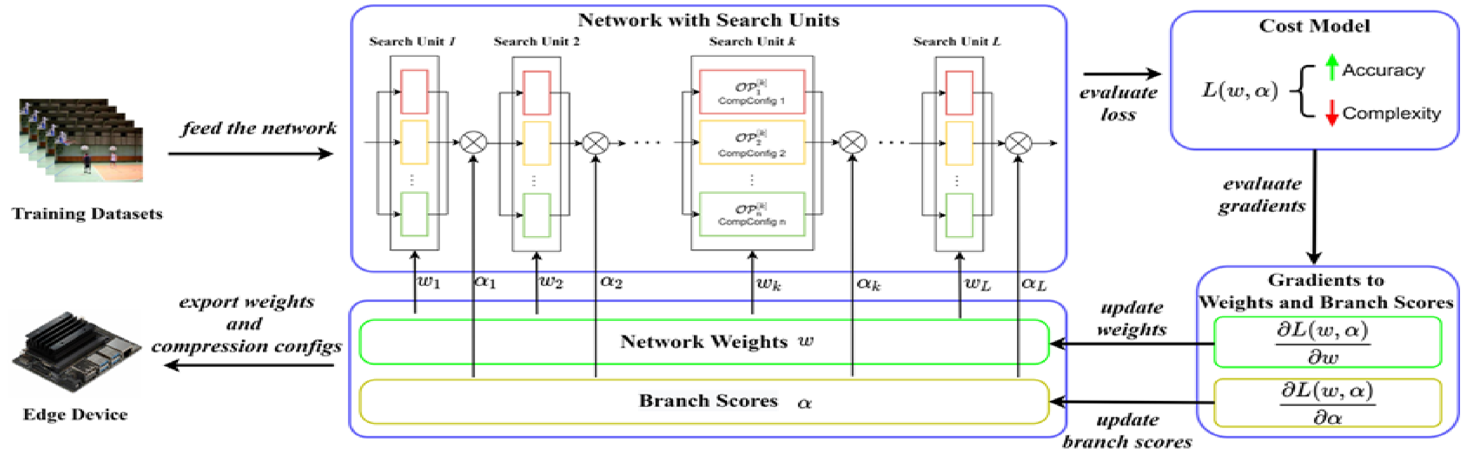
?

Deep learning framework

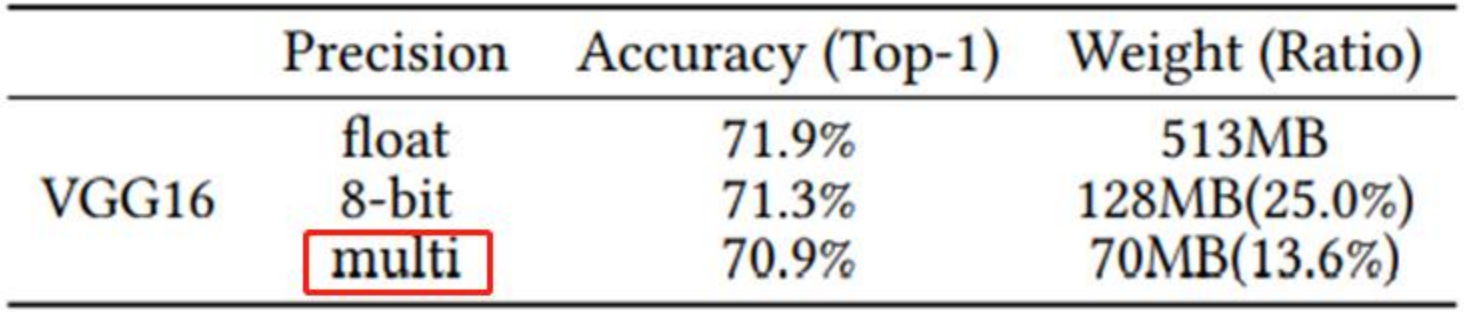
Hardware devices

Introduction: NAS Quantization





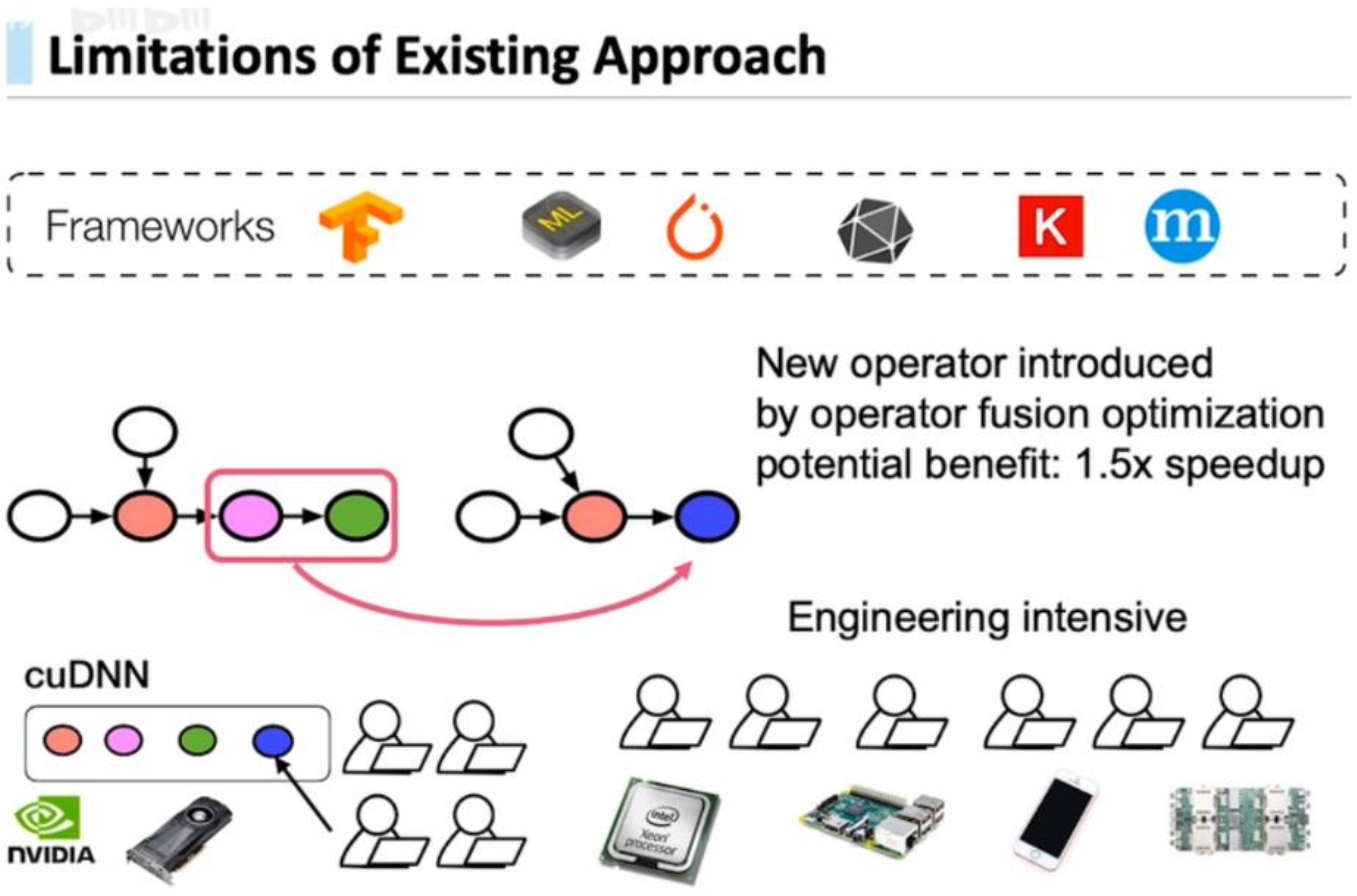
Branch scores turn discrete selection into differential



	Precision	Accuracy (Top-1)	Weight (Ratio)
VGG16	float	71.9%	513MB
	8-bit	71.3%	128MB(25.0%)
	multi	70.9%	70MB(13.6%)

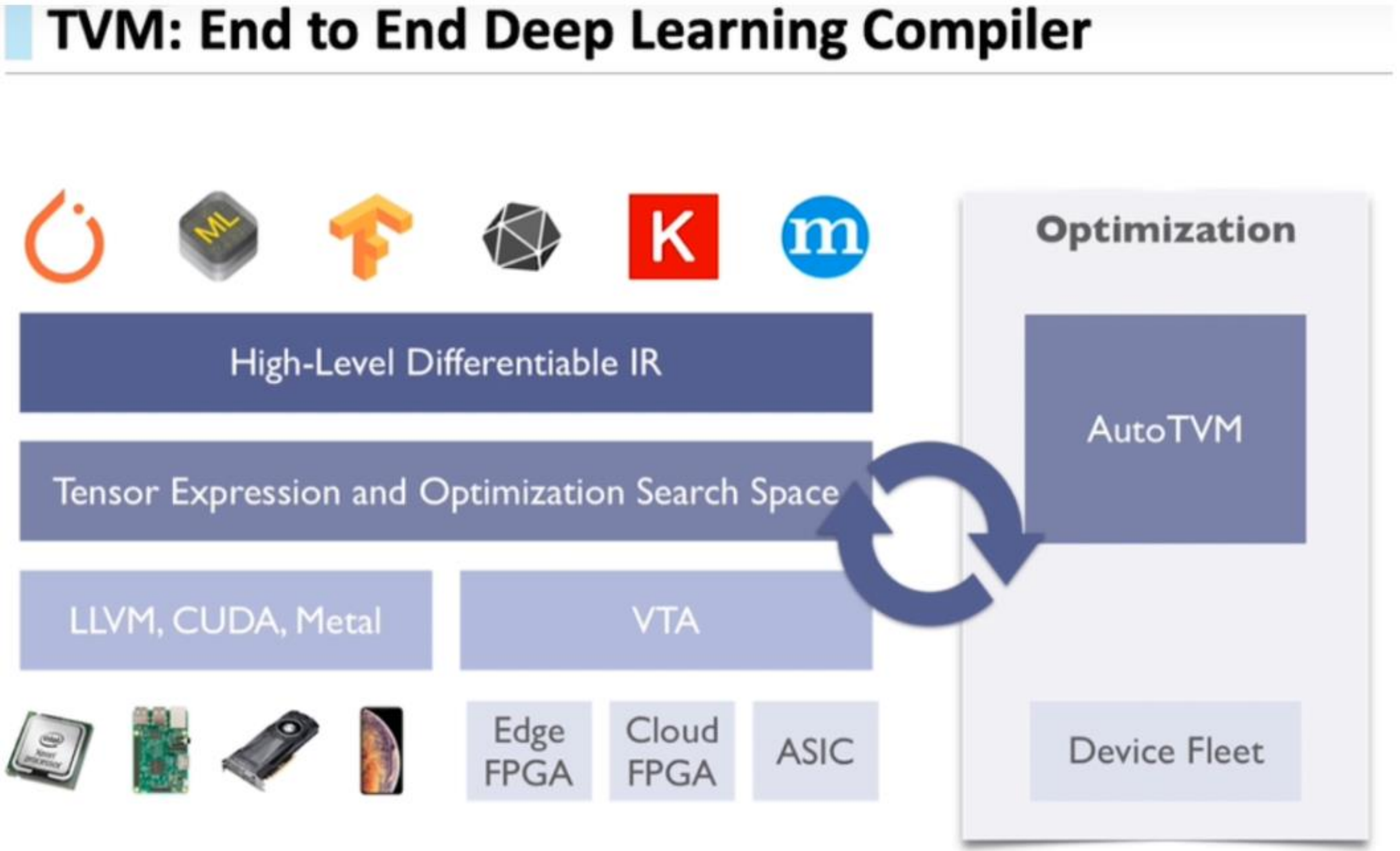
Advantages of NAS quantization: achieve a well trade-off between network performance and hardware efficiency

Introduction: Main Challenge


Limitations of Existing Approach
Frameworks
New operator introduced
by operator fusion optimization
potential benefit: 1.5x speedup
Engineering intensive
cuDNN
NVIDIA

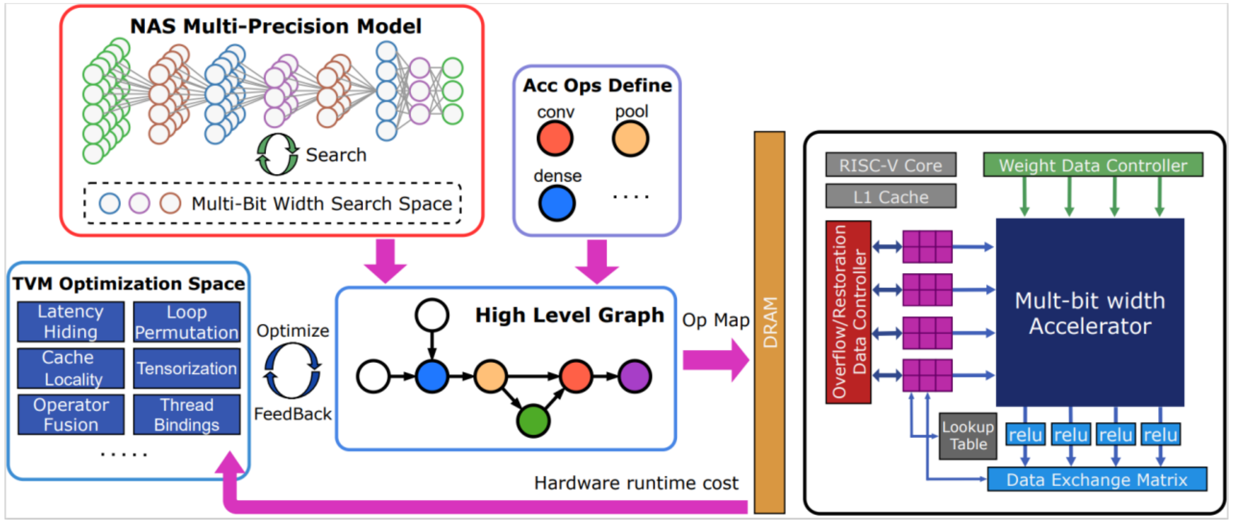
Require significant **manual tuning**

Introduction: TVM


TVM: End to End Deep Learning Compiler
ML
K
m
High-Level Differentiable IR
Tensor Expression and Optimization Search Space
LLVM, CUDA, Metal
VTA
Edge FPGA
Cloud FPGA
ASIC
Optimization
AutoTVM
Device Fleet

TVM provide performance portability to deep learning workloads across diverse hardware back-ends

Introduction: Main Contribution



Co-design and deep learning framework for RISC-V-based multi-precision deep learning microprocessor

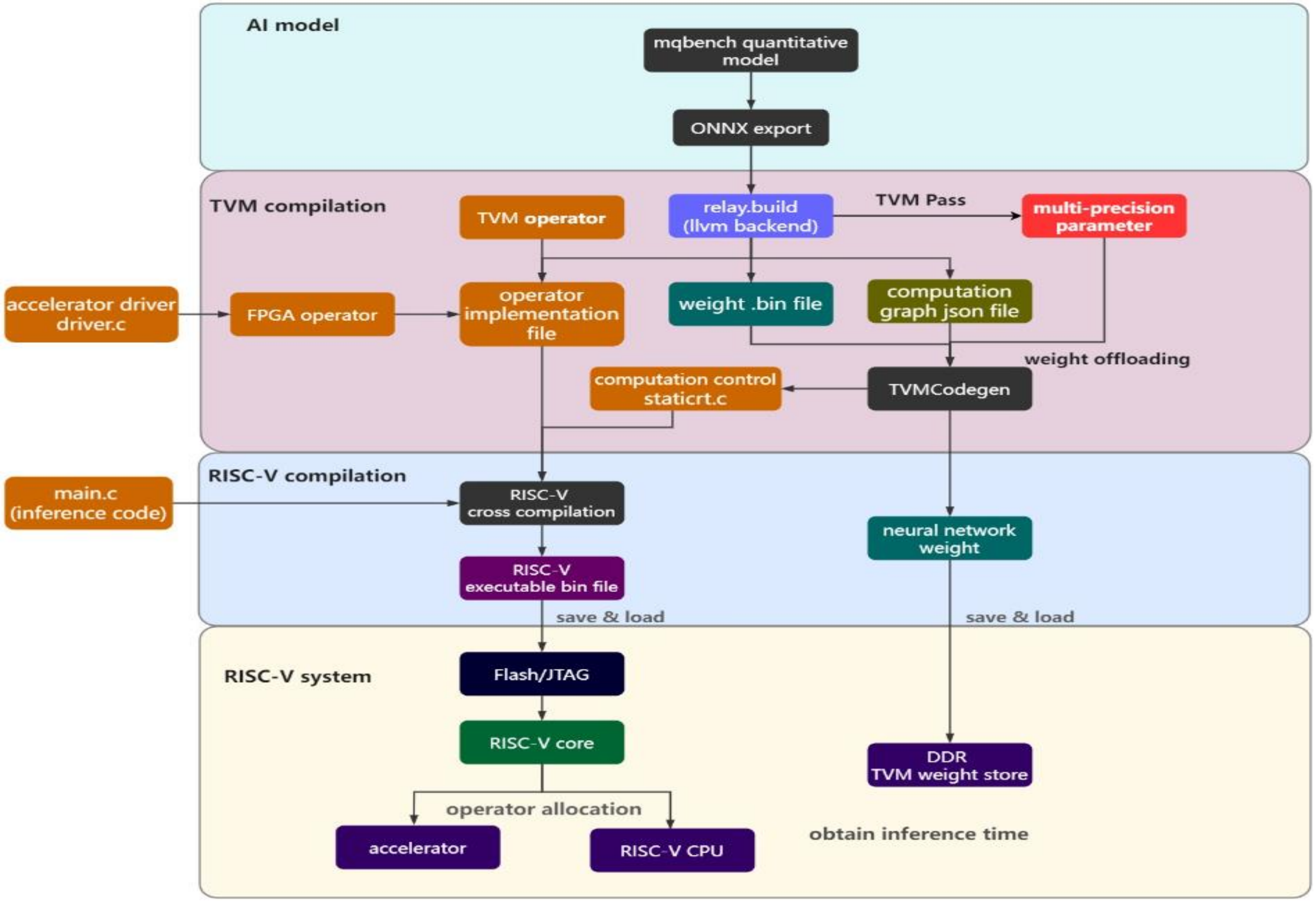
Engineering intensive: The model inference still has to be completed with manual deployment to achieve the full utilization of hardware

Few frameworks: There are however few frameworks developed supporting model deployment with mixed precision quantization on embedded devices

The main contribution of the proposed work:

- Agile hardware/software co-design topology
- Multi-precision accelerator
- TVM stack with multi-precision quantization support and custom RISC-V instructions

Introduction: Overall Flowchart

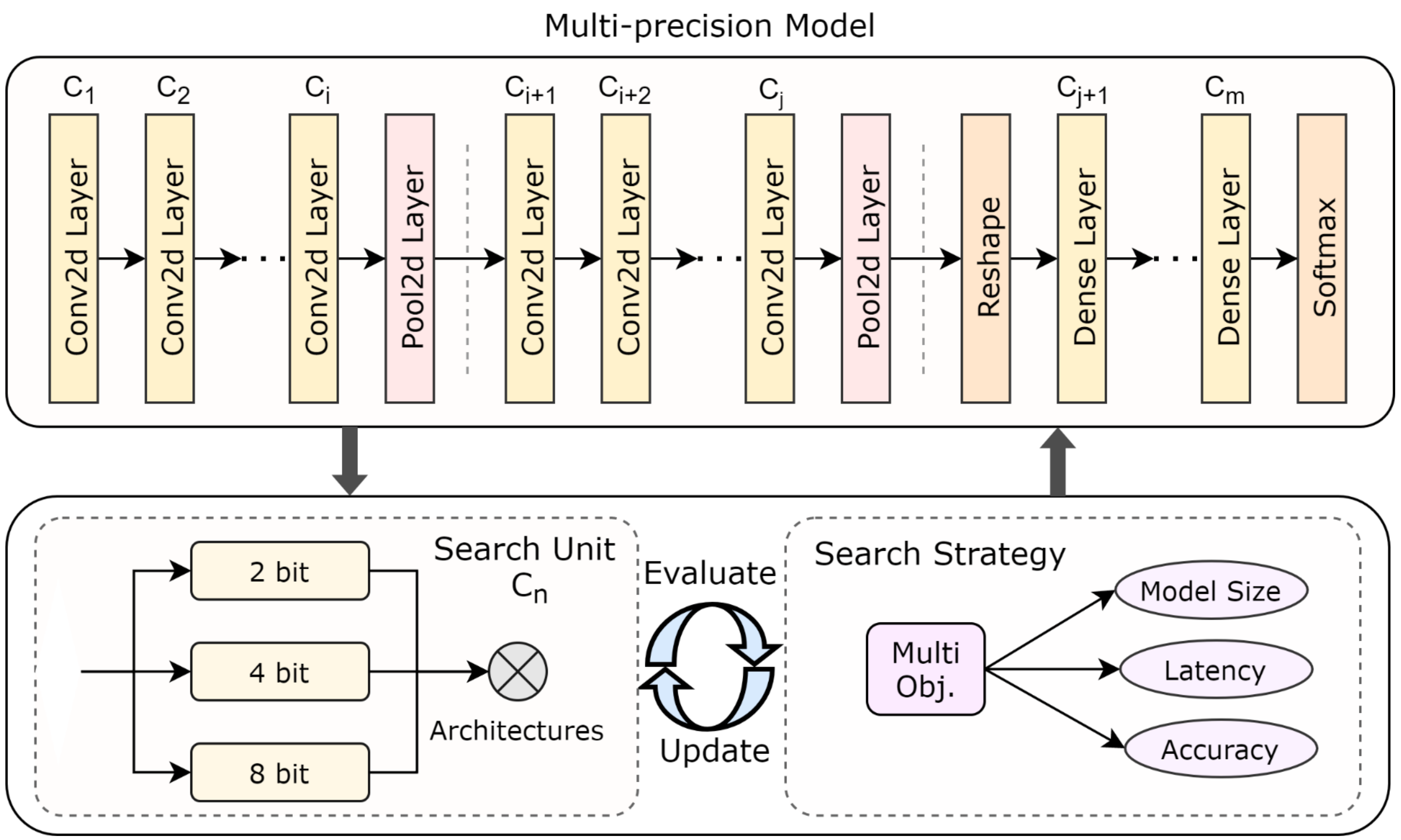


2

Multi-precision Accelerator

Multi-precision hardware and software implementation methods

Multi-precision Accelerator: Quantization



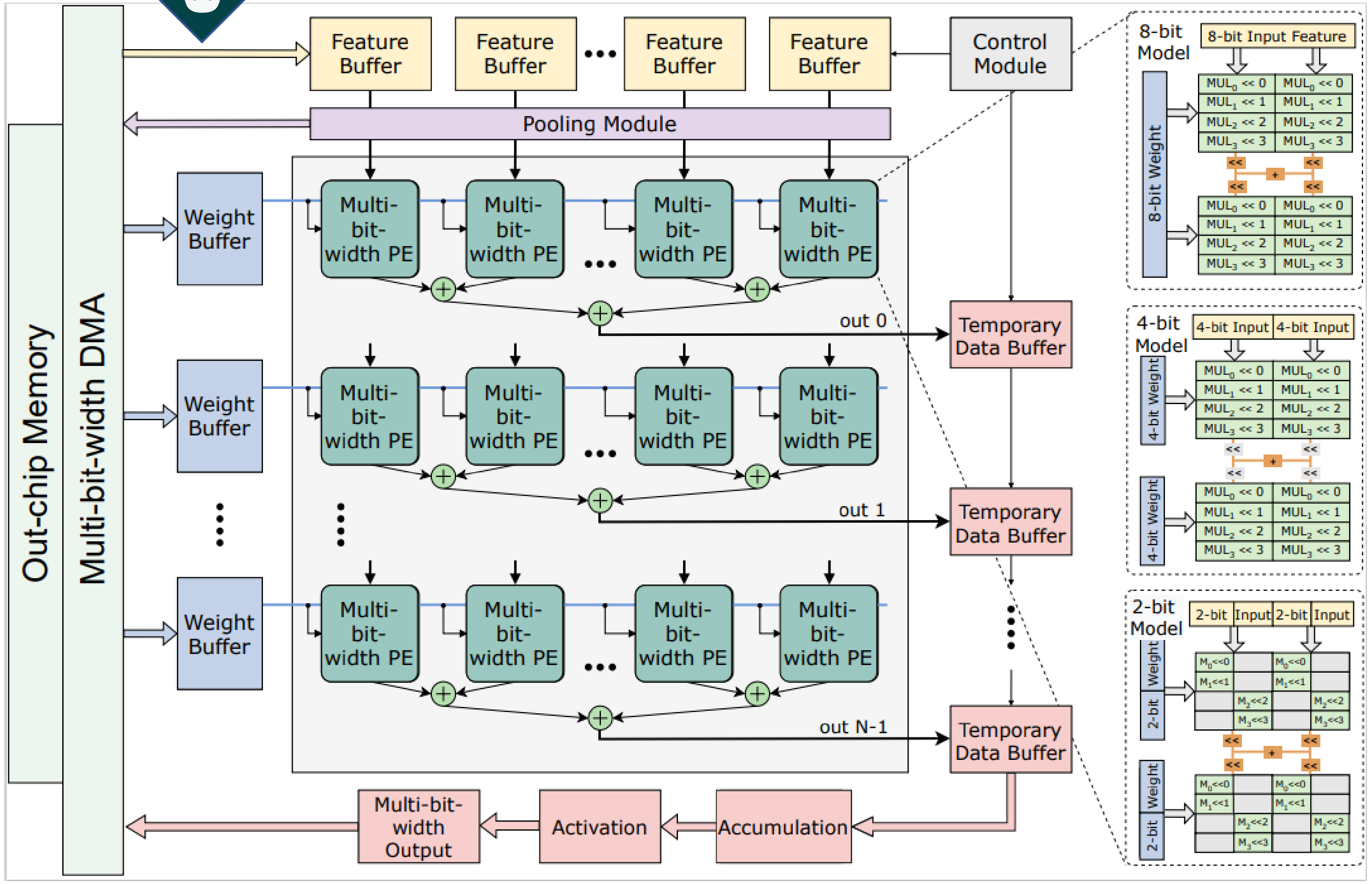
Model quantization

$$x_{int} = round(\frac{x}{\Delta})$$
$$x_Q = clamp(-N_{range}, N_{range} - 1, x_{int})$$

Multi-precision bit-width search

$$\min_{A \in \mathcal{A}} \min_{W,\alpha} L(w, \alpha)$$

Multi-precision Accelerator: Architecture



Hardware architecture of multi-precision accelerator

Multi-bit-width PE use **BSC method** that supports 8, 4, 2-bit multiplication

$$f_{calc}(x,y,z)=\sum_{i=0}^{K-1}\sum_{j=0}^{K-1}\sum_{m=1}^{C_{in}}F_{i,a}(x+i,y+j,m)K_{M,b}(i,j,m,z)$$

$$F_{out}(x,y,z)=\sum_{a=0}^{U-1}(2^a\cdot\sum_{b=0}^{U-1}(2^b\cdot f_{calc}(x,y,z)))$$

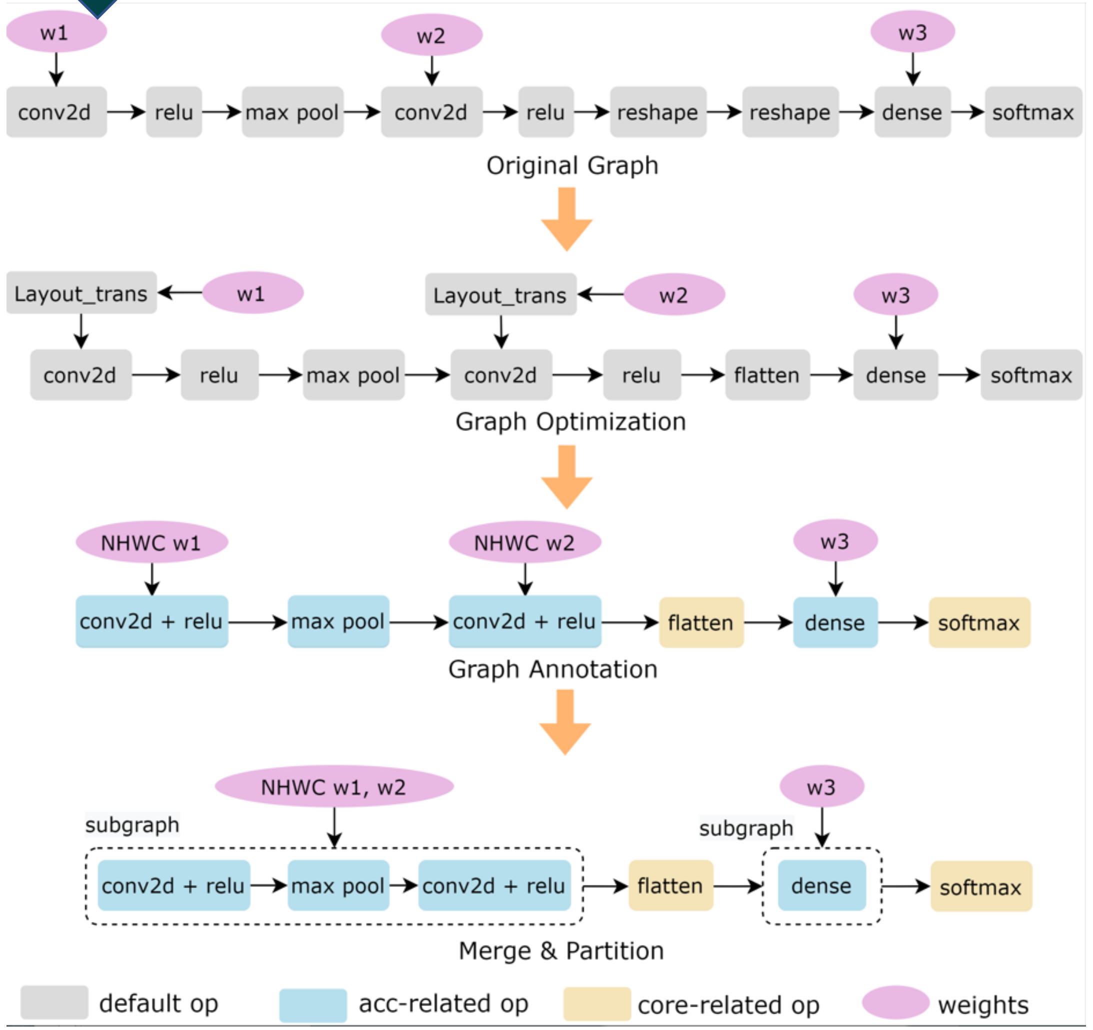
3

Multi-precision Compiler

Bridge between multi-precision models and hardware

13

Multi-precision Compiler: Graph Transform



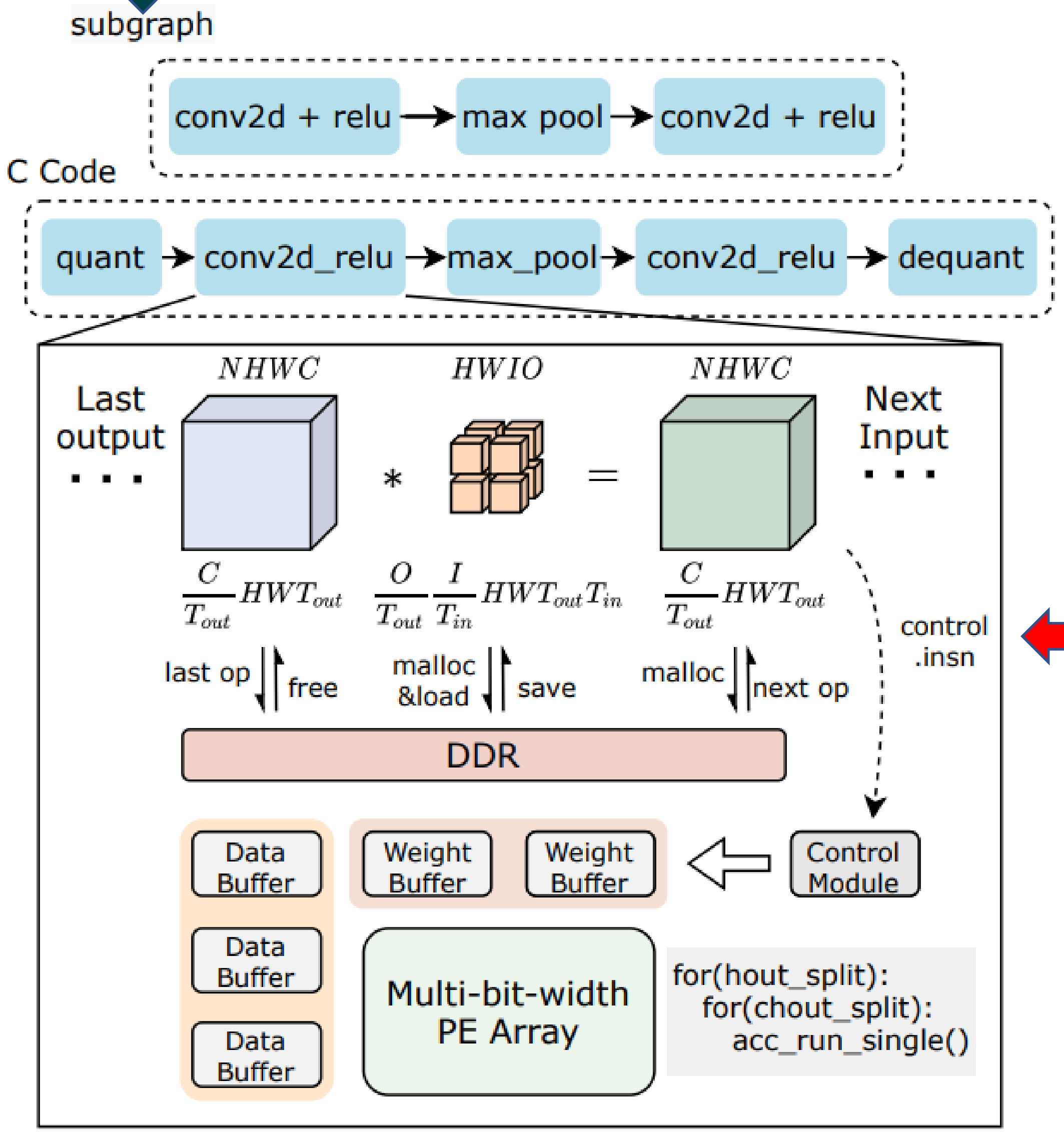
Graph transformation stages

Optimize the original graph: Constant folding, data layout transformation, etc

Annotate each operator: Annotate the backend that the operator needs to map

Merge adjacent identical annotation operators and partition: Form a computational subgraph of the same backend

Multi-precision Compiler: Mapping



Multi-Precision Operator Mapping Accelerator

The subgraph is embedded in the graph, and the quantization and accuracy recovery operations only need to be placed at the beginning and end of each subgraph

```
// asm(".insn r opcode func3 func7 rd rx1 rx2")
// Set the base address of the accelerator
void Init ( uint32_t acc_addr );
// Write the data into the name - register
void Write_Reg ( reg_t name , const int data );
// Read the data from the name - register
int Read_Reg ( reg_t name );
```

Inline assembly functions read and write to accelerator registers

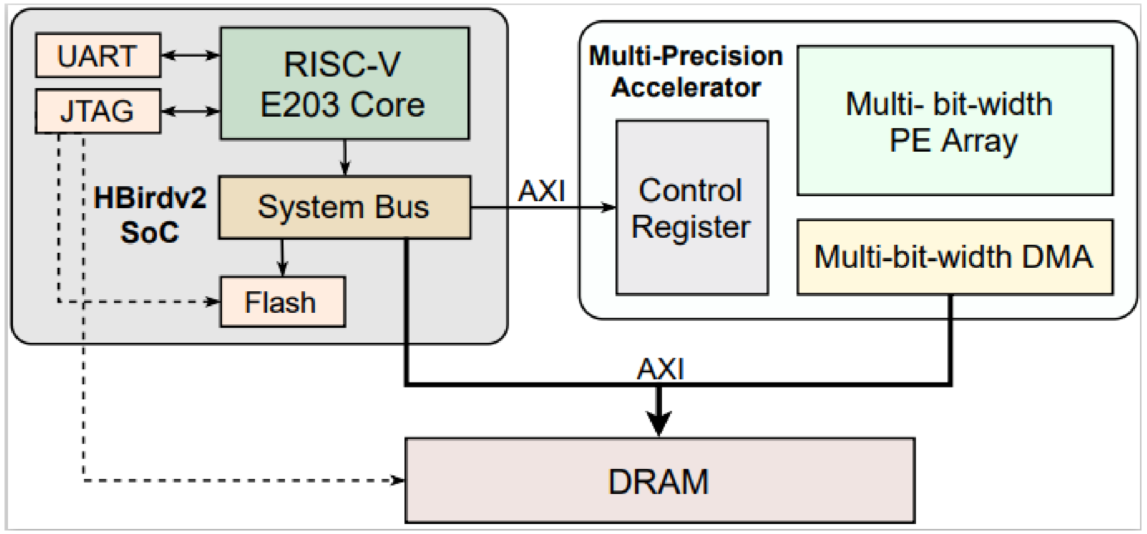
4

Multi-precision System

Complete operation system construction

Multi-precision System: Communication

RISC-V based Multi-precision Microprocessor



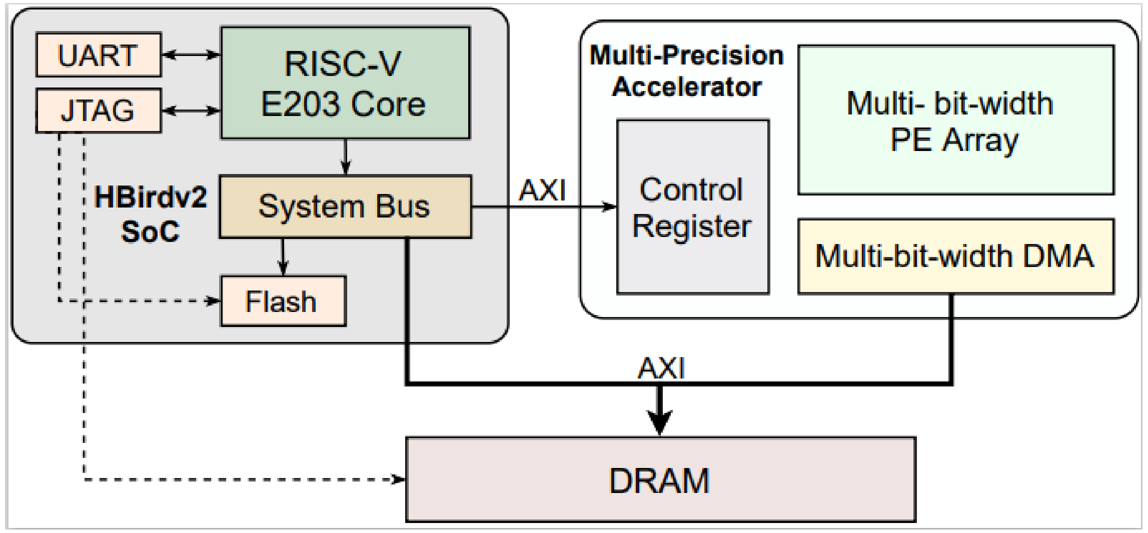
Communication: The accelerator is connected to the RISC-V core through the AXI-Lite bus, and the accelerator can also read and write data directly to the DRAM through AXI-Full

Device Module	Start Address	End Address
Debug Module	0x0000_0000	0x0000_0FFF
UART Serial Port	0x1001_3000	0x1001_3FFF
DRAM Module	0x4000_0000	0x4FFF_FFFF
Acc Control Register	0x5000_0000	0x5000_FFFF

The physical addresses of the microprocessor peripherals are as above

Multi-precision System: Instruction

RISC-V based Multi-precision Microprocessor



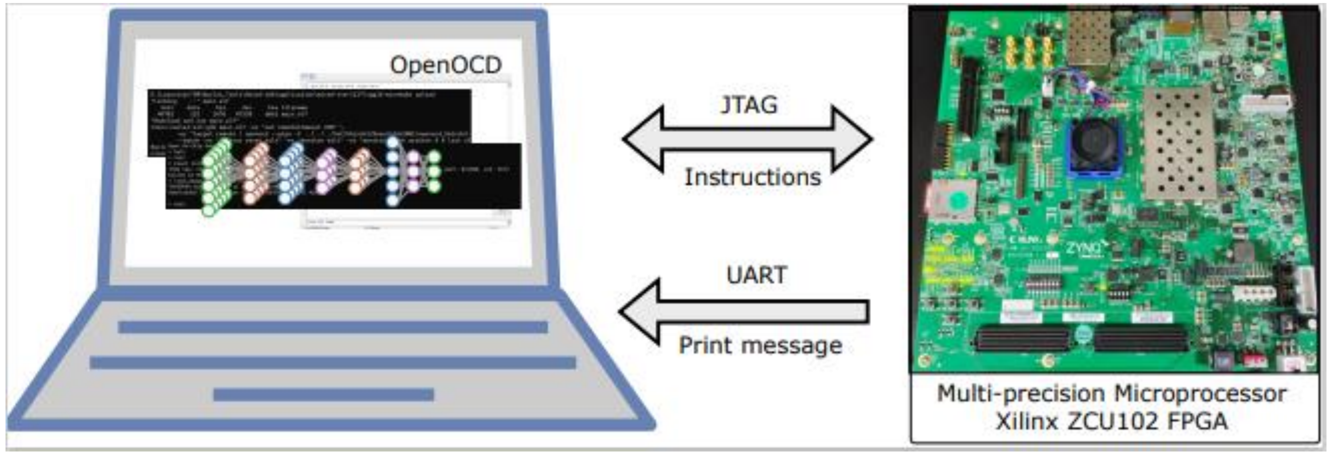
Custom Instruction: CL_RG, LD_RG, and ST_RG represent cleaning, loading, and storing data, respectively

31:25	24:20	19:15	14	13	12	11:7	6:0
func7	rs2	rs1	xd	xs1	xs2	rd	opcode
CL_RG	00000	00000	0	0	0	00000	CUSTOM-0
LD_RG	00000	rs1	1	1	0	rd	CUSTOM-0
ST_RG	rs2	rs1	0	1	1	00000	CUSTOM-0

5

Experiment Results

Experiment Results: Experiment Setup

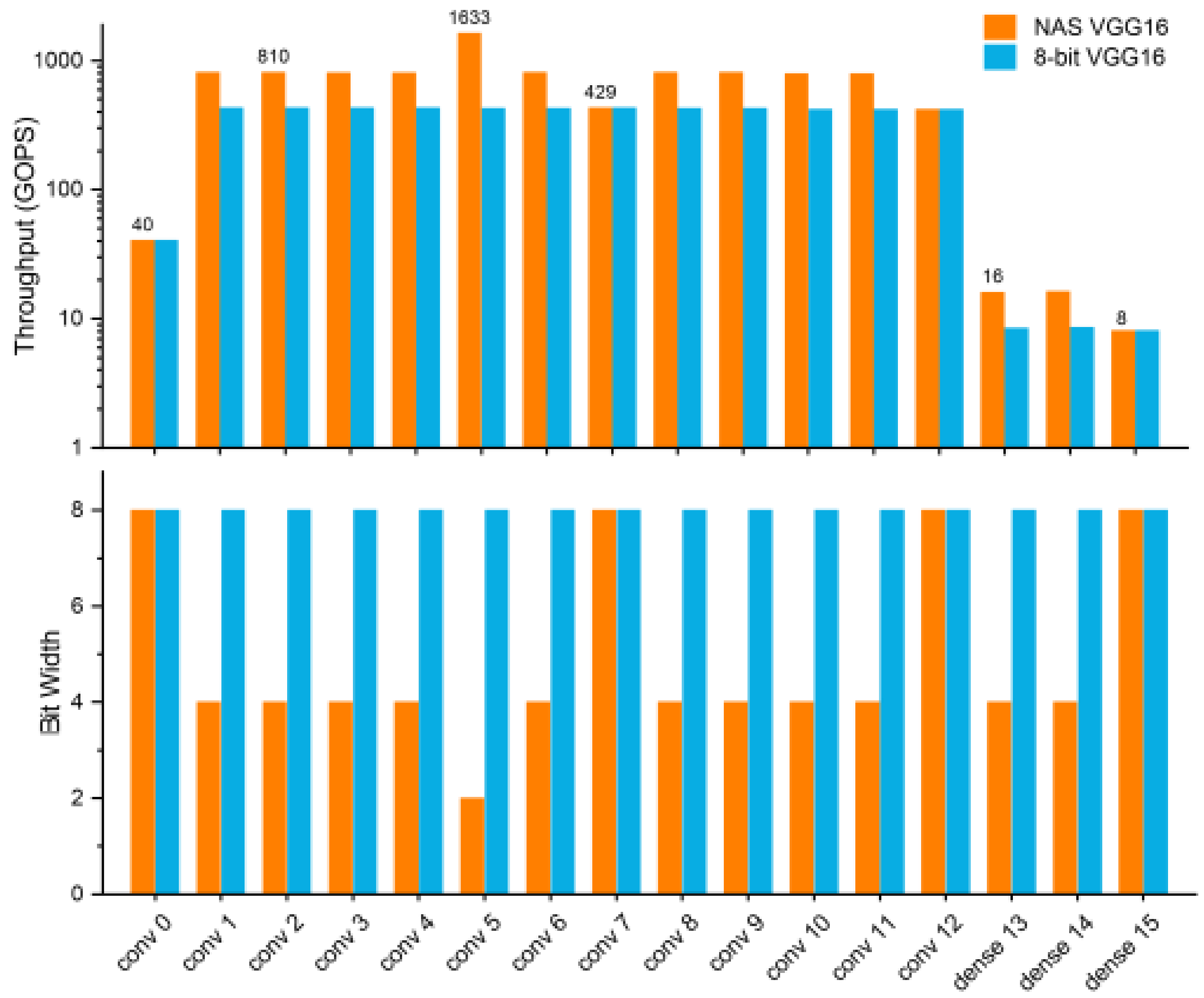


PC **FPGA**

	Precision	Accuracy (Top-1)	Weight (Ratio)
VGG16	float	71.9%	513MB
	8-bit	71.3%	128MB(25.0%)
	multi	70.9%	70MB(13.6%)

The loss of NAS-VGG16 accuracy is only within 1.0%

Experiment Results: Throughput



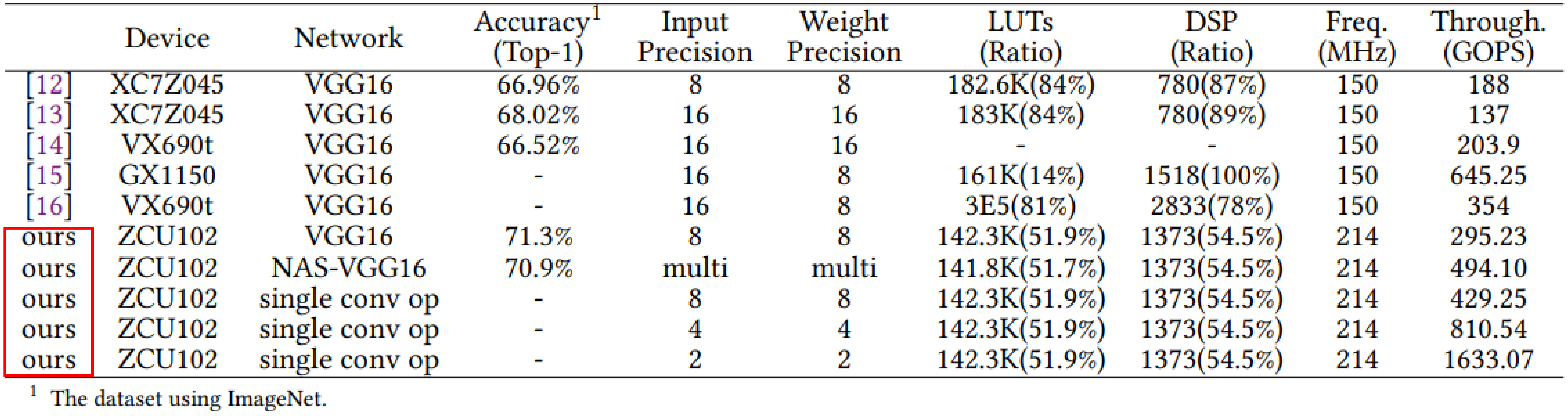
Throughput and bit-width of the NAS-based VGG16 and 8-bit VGG16 model at conv2d and dense layers

NAS VGG16 VS 8-bit VGG16

Lower bit-width: Compared with 8-bit model, **NAS-VGG16** allows lower bit-width

Higher throughput: For the multi-precision operators, the accelerator could reach **1633 GOPS** for 2-bit in conv5, **810 GOPS** for 4-bit in conv2 and **429 GOPS** for 8-bit in conv7

Experiment Results: Throughput

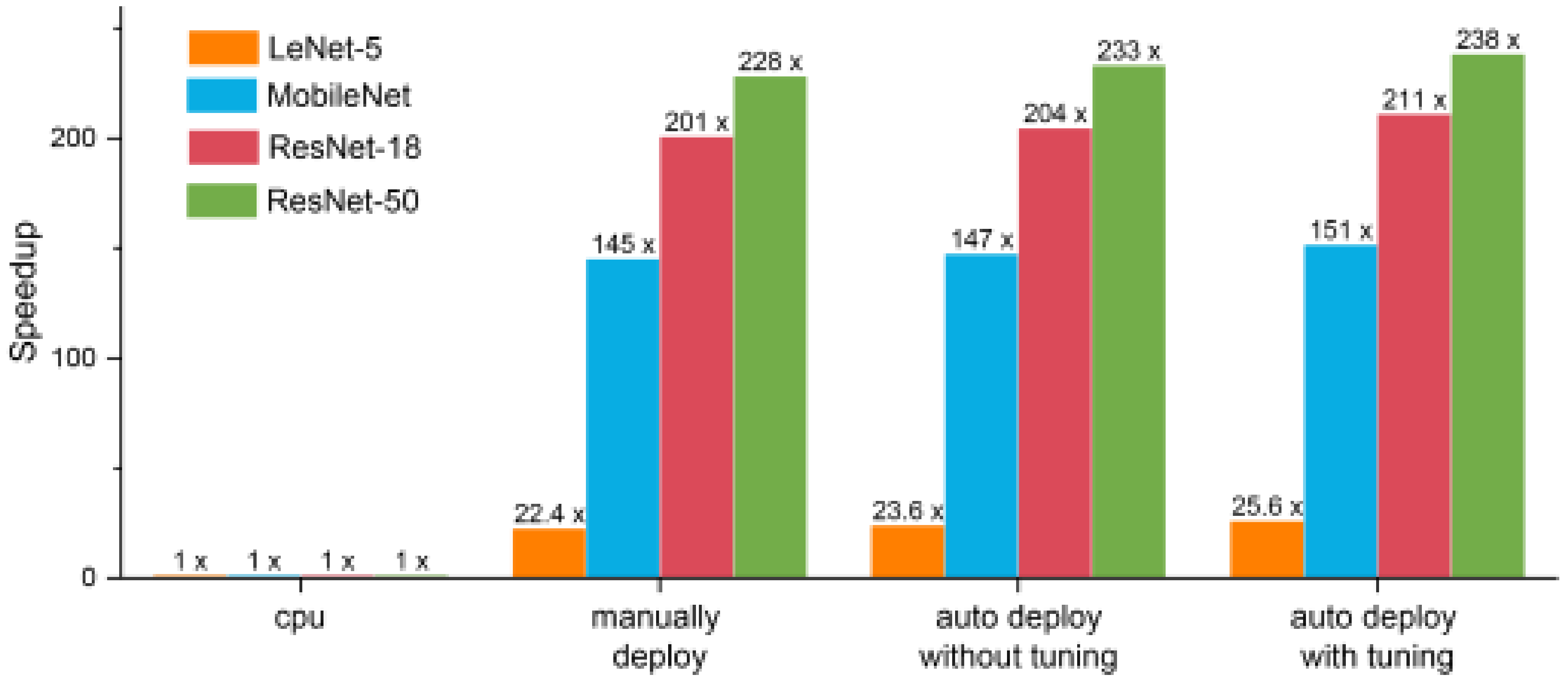


	Device	Network	Accuracy[1] (Top-1)	Input Precision	Weight Precision	LUTs (Ratio)	DSP (Ratio)	Freq. (MHz)	Through. (GOPS)
[12]	XC7Z045	VGG16	66.96%	8	8	182.6K(84%)	780(87%)	150	188
[13]	XC7Z045	VGG16	68.02%	16	16	183K(84%)	780(89%)	150	137
[14]	VX690t	VGG16	66.52%	16	16	-	-	150	203.9
[15]	GX1150	VGG16	-	16	8	161K(14%)	1518(100%)	150	645.25
[16]	VX690t	VGG16	-	16	8	3E5(81%)	2833(78%)	150	354
ours	ZCU102	VGG16	71.3%	8	8	142.3K(51.9%)	1373(54.5%)	214	295.23
ours	ZCU102	NAS-VGG16	70.9%	multi	multi	141.8K(51.7%)	1373(54.5%)	214	494.10
ours	ZCU102	single conv op	-	8	8	142.3K(51.9%)	1373(54.5%)	214	429.25
ours	ZCU102	single conv op	-	4	4	142.3K(51.9%)	1373(54.5%)	214	810.54
ours	ZCU102	single conv op	-	2	2	142.3K(51.9%)	1373(54.5%)	214	1633.07

[1] The dataset using ImageNet.

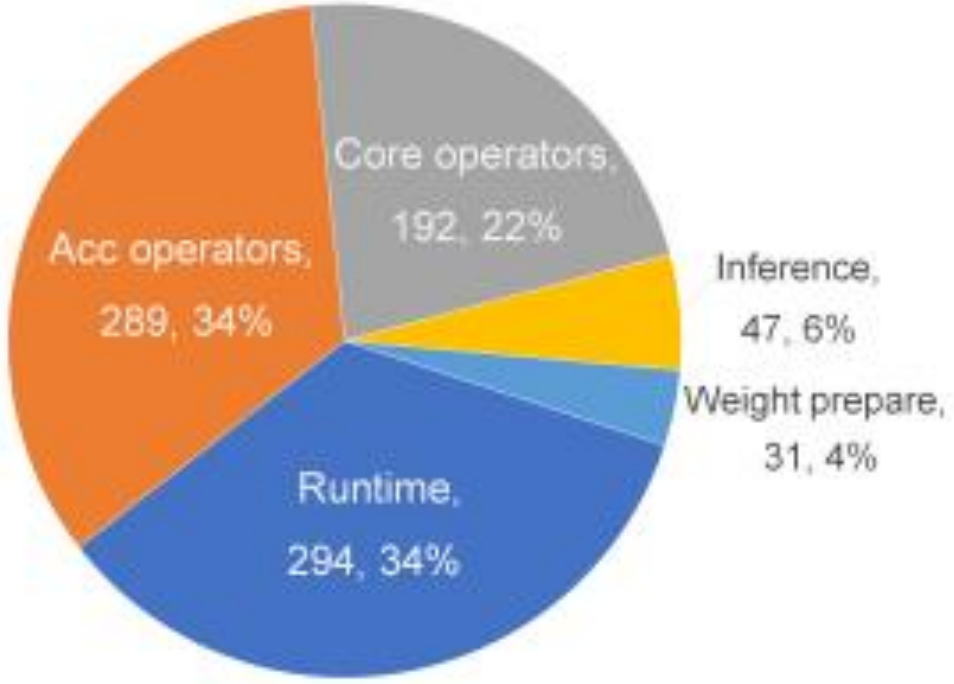
We also compare our results with prior specialized FPGA accelerators and the comparison table is shown in table

Experiment Results: Latency

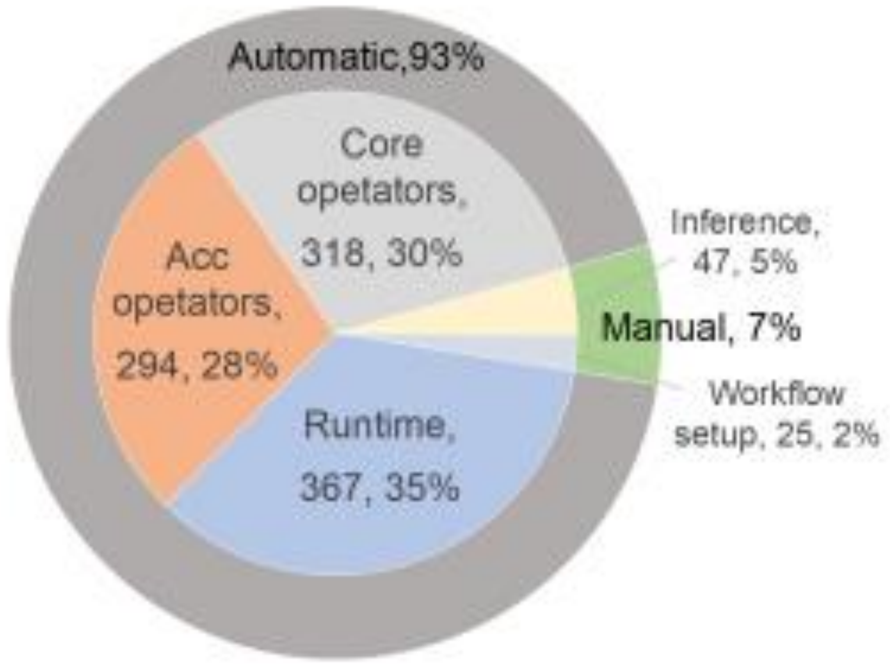


Deployment of NAS multi-precision optimization models for some models with non-accelerator operators: LeNet-5, MobileNetv1, ResNet-18 and ResNet-50

Experiment Results: Agility



(a) manual deployment



(b) auto deployment

Line-of code comparison for MobileNet

6 Conclusion

Conclusion

1 We proposed an **agile hardware/software co-design** microprocessor featured with automatic NAS-based optimized models deployment.

2 We implemented the heterogeneous architecture using **BSC accelerator** controlled by **custom RISC-V instructions**, which exhibits high flexibility and decent performance for various CNN models.

3 We also developed our compiler with custom FPGA backend integration to automate DL workloads mapping with both graph-level and operator-level optimizations.

Thank You!

SUSTech Southern University of Science and Technology