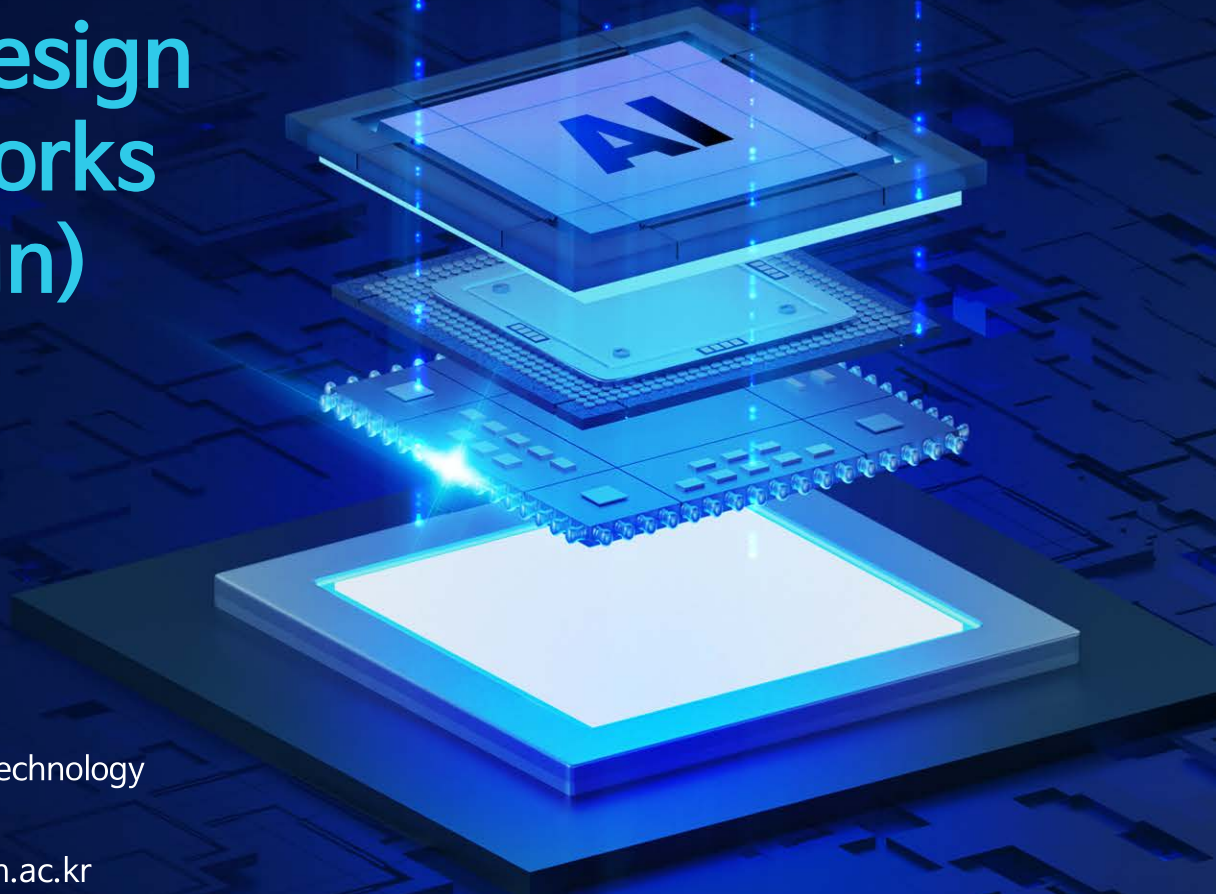
IDSL
Intelligent Digital Systems Design Lab.

SEOULTECH 서울과학기술대학교

Digital system (SoC) design for Deep Neural Networks (AI accelerator design)



Presenter Hyun Kim | Associate Professor

Affiliation Seoul National University of Science and Technology
Electrical and Information Engineering

Contact hyunkim@seoultech.ac.kr / idsl.seoultech.ac.kr

On-device AI

On-device AI is a technology that implements AI applications on edge/mobile devices by embedding its own acceleration platform without computation on a physically distant server

Advantages

01 Overcomes **Privacy/Security issues** of the traditional method where data is transmitted to the cloud for centralized processing

02 Enables **real-time processing of AI services** (low latency) without the need for wireless communication

03 Offers **Personalization**

04 **Solves the problem of limited training data**

05 **Reduces datacenter infrastructure costs/energies**

06 Supports **on-device training** at the fine-tuning level makes federated learning possible → Further reduces datacenter costs, enhances privacy protection, and optimizes user-specific performance

Target Markets for On-Device AI Inferencing

IoT	Mobile	AR/VR	Smart Surveillance	Autonomous Vehicles
<0.5 TMAC	0.5-2 TMAC	1-4 TMAC	2-10 TMAC	10s-100s TMAC

On-device AI Training

Demand for 'On-device AI Training' for optimal AI models in user's environments

01 Difficulty to cope with **different environments of each user** only with the global model created by the cloud

02 Re-training of AI models for each user in the cloud requires **large-scale computing/memory resources** on the cloud and may raise **personal privacy issues and labeling burdens**

On-device AI Training shares the burden of computation and labeling of the data center and delivers user-specific performance for each user without privacy issues

Challenging point

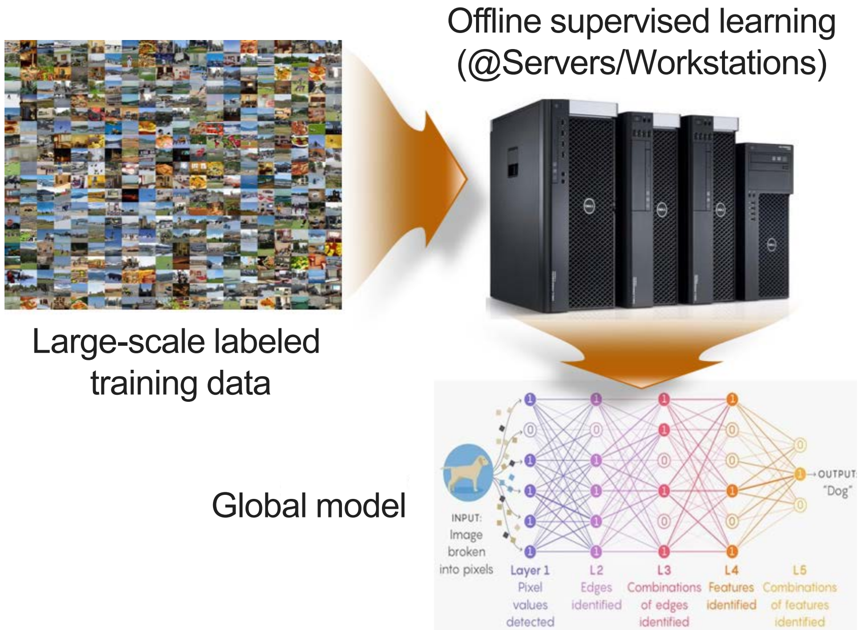
HW

Difficulty in implementing PEs for backpropagation + developing light weight schemes for DNN training

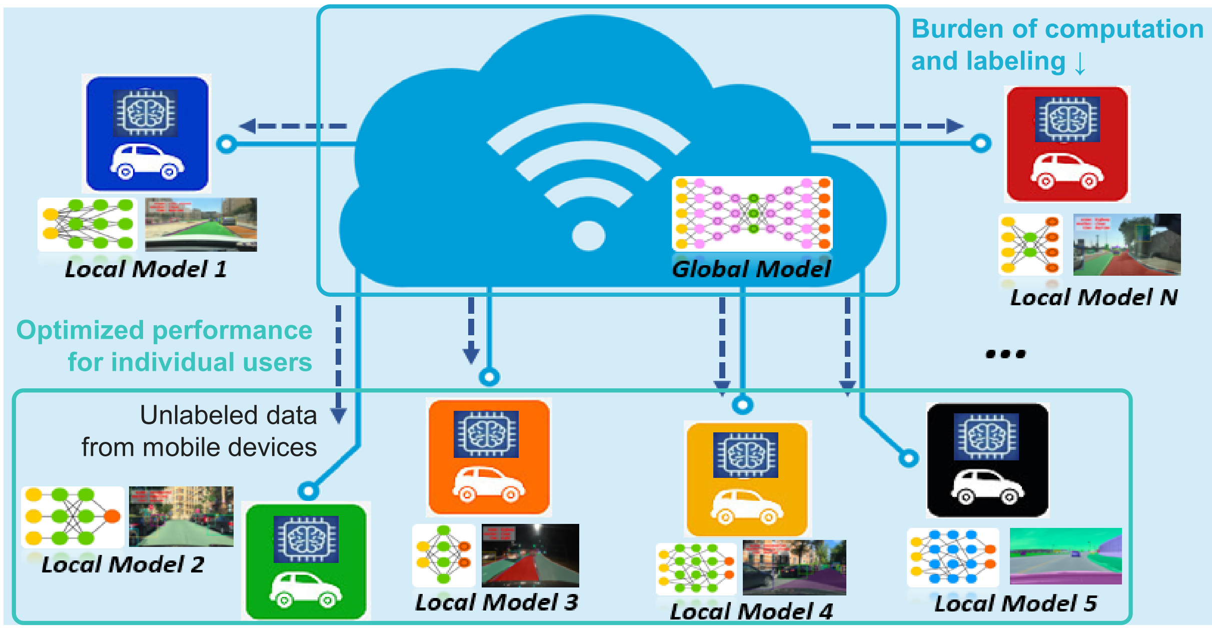
SW (Algorithm)

Difficulty in making pseudo labels of unlabeled data for self-learning + training only additional data

Example of on-server training of DNN model



On-device training of DNN model (smart mobility)

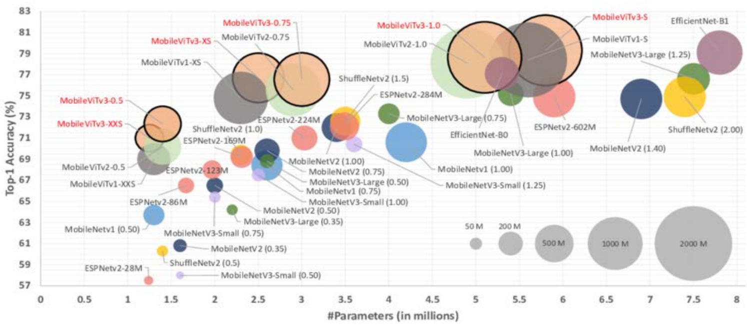


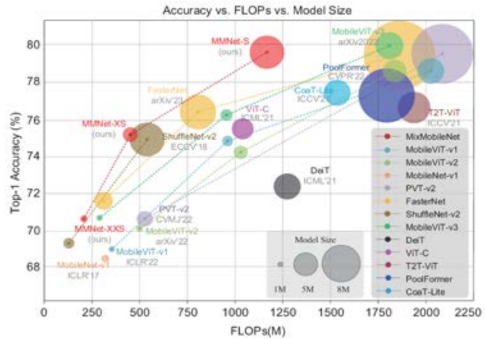
Challenges in On-device AI

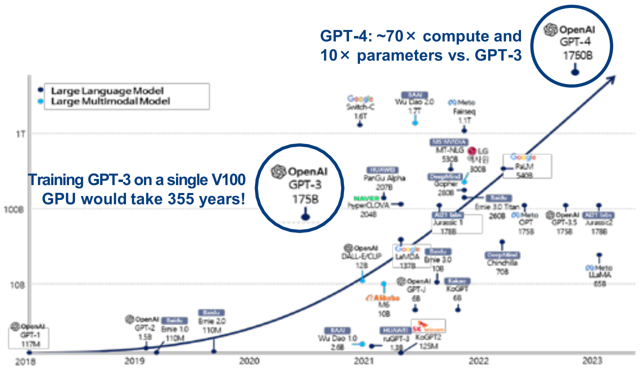
Large Computational Cost/Model Size

Layer depth and width are constantly increasing to **achieve higher accuracy**

Outstanding performance of AI has been sufficiently proven!





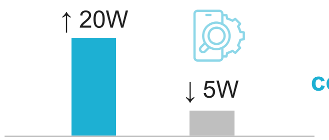


Large Power Consumption

Large computational costs lead to huge power consumption, and **AI's performance is limited** by power budget, especially on mobile devices

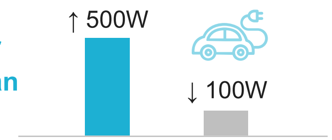
Limited Memory Capacity

On-device AI model sizes are constrained by the **main memory capacities of mobile devices (usually around 8GB) and laptops (typically about 16GB)**



Mobile Phone

4x-5x power consumption than power budget



Autonomous Driving

Go competition

Human vs AI (2016.03)

# of parameters (B)	GB of RAM (float32s)	GB of RAM (float16s)	GB of RAM (int8s)	GB of RAM (int4s)
7	28	14	7	3.5
13	52	26	13	6.5
32.5	130	65	32.5	16.25
65.2	260.8	130.4	65.2	32.6

Mobile limit

Laptop limit

※ **Energy-efficient AI accelerators** are expected as a key solution to these challenges in On-device AI

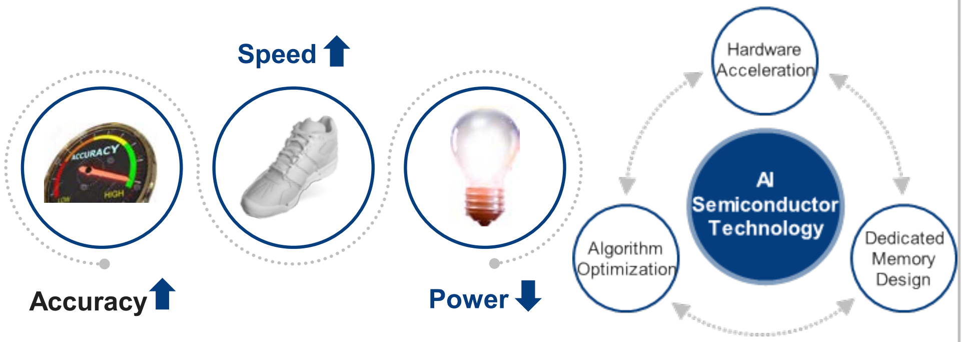
New AI Accelerators

	Global					
Company	intel	Qualcomm	Google	AXELERA	HAILO Empowering Intelligence	tenstorrent
Country	USA	USA	USA	Netherlands	Israel	Canada
Target	Cloud Edge (On-Premise) On-device	Cloud Edge (On-Premise) On-device	Cloud Edge (On-Premise) On-device	Edge (On-Premise) On-device	On-device	Edge (On-Premise) On-device
Representative products and performance	Lunar Lake NPU Edge 48 TOPS, 15W Mass production completed	Hexagon Edge 45 TOPS, 2W Mass production completed	Edge TPU Edge 4 TOPS, 2W Mass production completed	Metis Edge 214 TOPS, 14W Mass production completed	Hailo-8 Edge 2 TOPS, 2.5W Mass production completed	Eagle-N Edge (Vehicle) 250 TOPS, 5W Mass production scheduled
	Gaudi Server 1,835 TFLOPS 900W Mass production completed	Cloud AI 100 Ultra Server 288 TFLOPS 150W Mass production completed	Cloud TPU v4 Server 275 TFLOPS 250W Mass production completed	-	Hailo-8L Edge 13 TOPS 1.5W Mass production completed	Wormhole n150 Server 262 TLOPS (FP8) 160W Mass production completed

Key Issue of On-device AI Accelerators: Architecture

Three main goals of AI accelerators

High accuracy + High speed (Throughput) + Low power (Energy-Efficiency)



Necessity of architecture-level approach

Hardware acceleration of the optimized neural networks with parallelization and optimization enables **fast processing** with **low-power consumption**

Key Point

Optimal architecture design considering the model structure and the Roofline model

Most effective way to accelerate AI

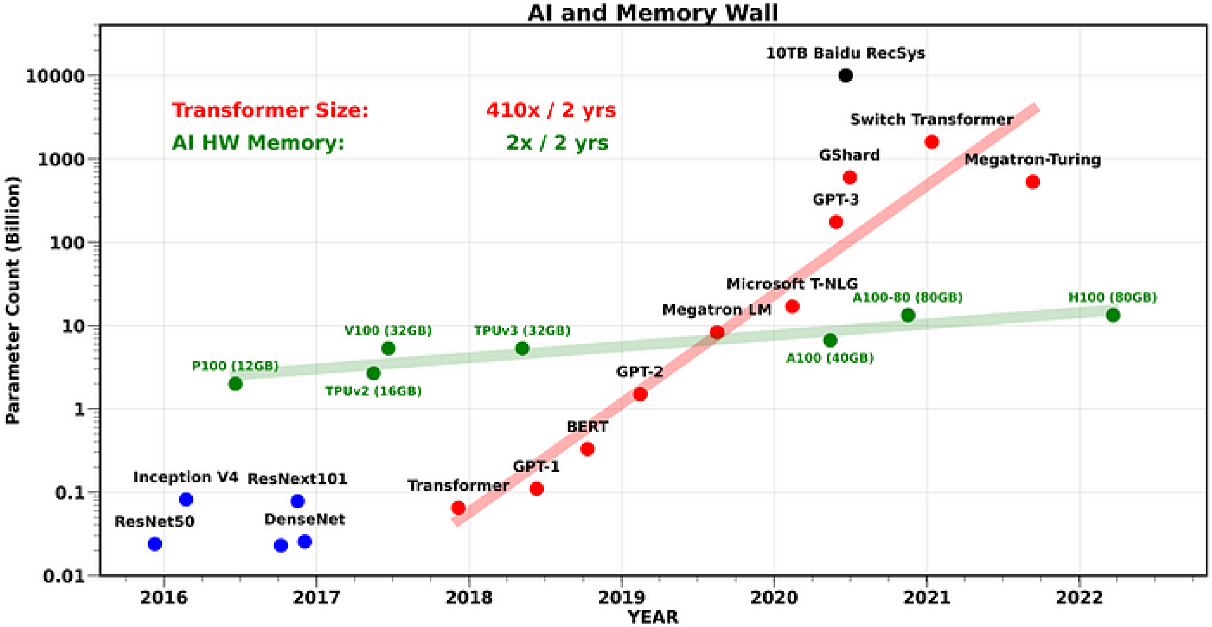
GPU — Various AI development frameworks are supported, but **GPU suffers from size & cost & power problems!**

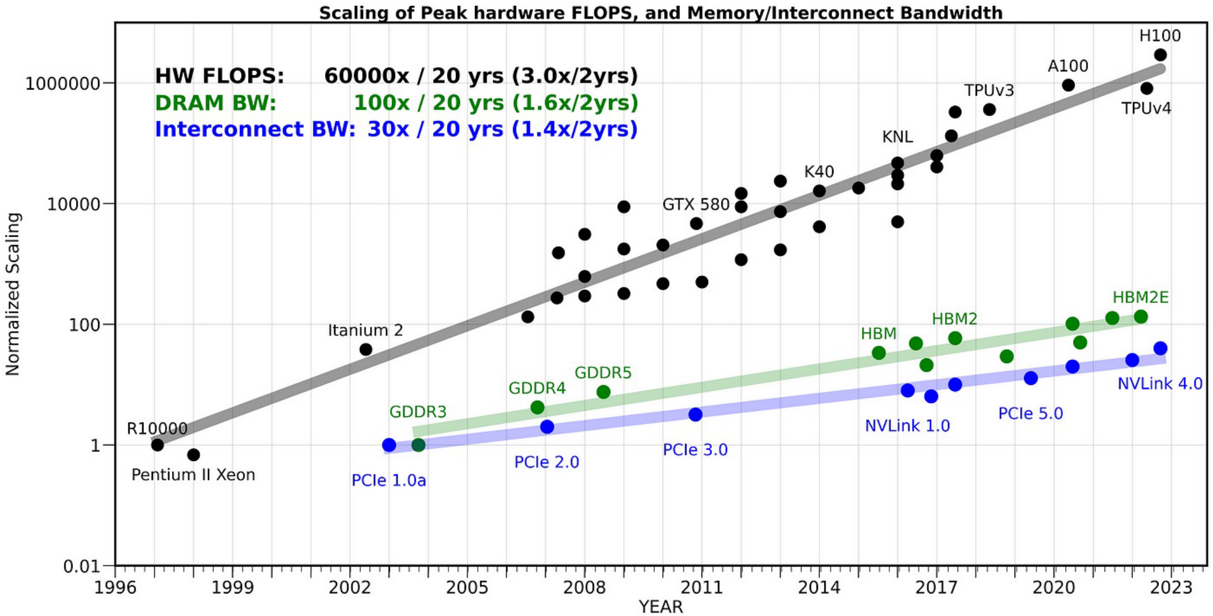
FPGA/ASIC — Expertise in implementation is required, but this approach has advantages of **small size, high cost-efficiency, high power-efficiency**, and is easy to apply techniques to increase **hardware utilization**

Power of processing the same network on FPGAs & GPUs

Model	CPU	GPU	FPGA
MobileNet	-	73W	5.9W
YOLOv2	-	170W	18.3W
YOLACT	57.2W	129W	7.1W
MobileViT	47.6W	106W	6.3W
RoBERTa	80W	126W	52.13W
Llama2	42.5W	130.6W	9W

Scaling of Model Size / PU / Memory / Interface



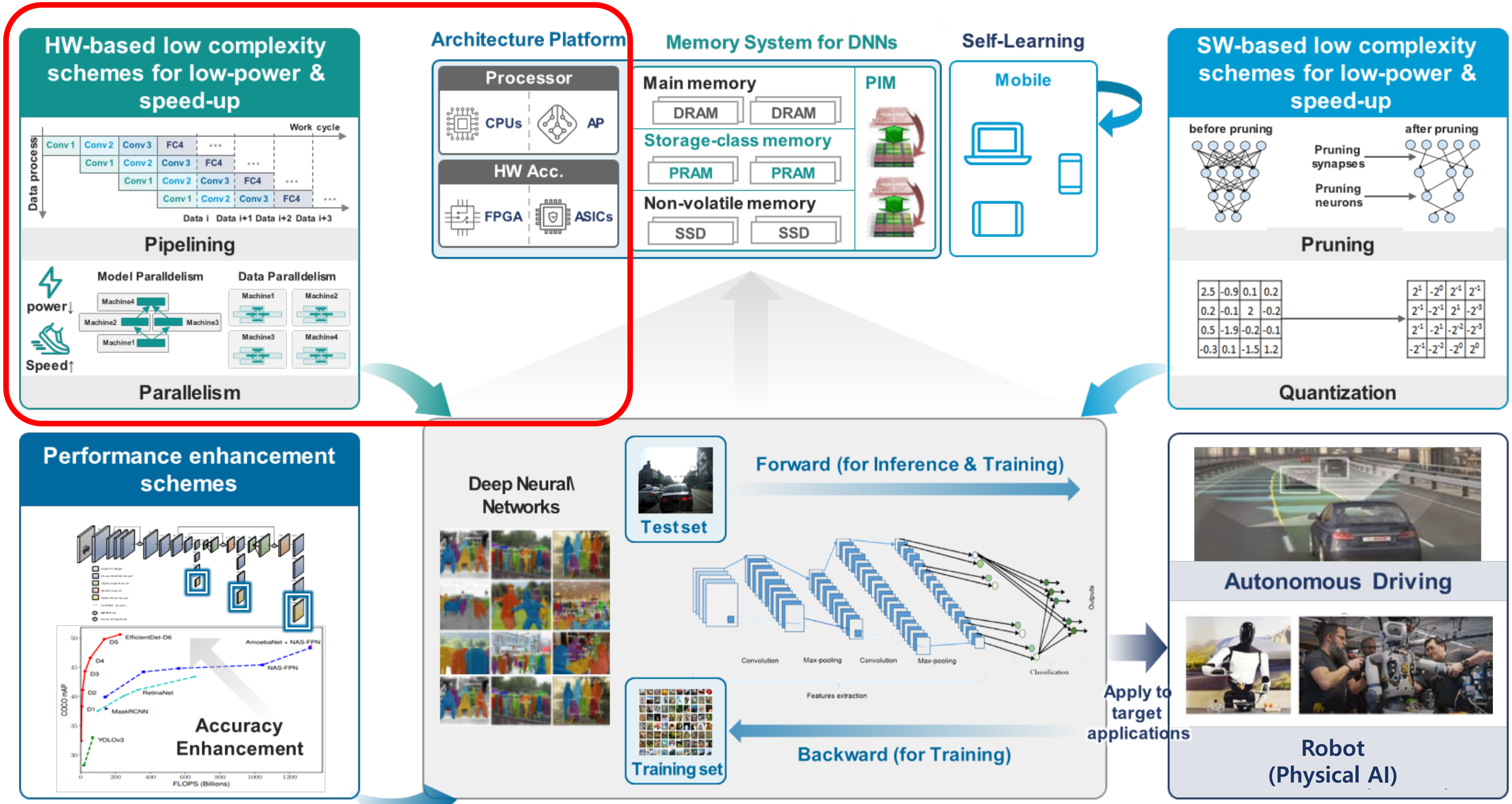


Overview of On-device AI Accelerators

IDSL Intelligent Digital Systems Design Lab.
SEOULTECH 서울과학기술대학교

Mobile Characteristic

- Both Inference & Training
- Low-Power FPGA/ASIC for Mobile
- Low Precision: 2b/4b/8b (INT)
- Sparse network
- Application-specific accelerator design



Fully-Pipelined Bottleneck FPGA Architecture for MobileNetv2

IDSL Intelligent Digital Systems Design Lab.
SEOULTECH 서울과학기술대학교

Goal

Design of an **energy-efficient MobileNetv2 accelerator** by **integrating structural features of the bottleneck block with batching techniques**

Motivation

- Existing designs have often overlooked the memory requirements of the bottleneck block (BB) and thus suffered from low HW utilization and poor support for batch processing, due to structural bottlenecks in MobileNet

Solution/ Contribution

1 Row-wise Pipelined Dataflow

Design a row-wise pipelined dataflow for the BB, considering the latency of each convolution stage, based on HW-oriented analysis of the MobileNet architecture

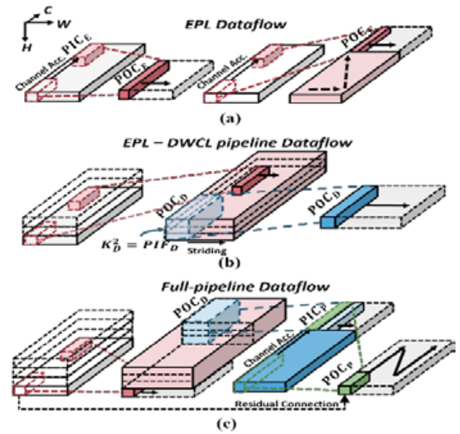
2 Fully-Pipelined Bottleneck Block (FPB)

Introduce an FPB architecture optimized for row-wise pipelined dataflow → Eliminate intermediate external memory accesses and reduce activation memory requirements by approximately 87% compared to a naïve pipelined implementation

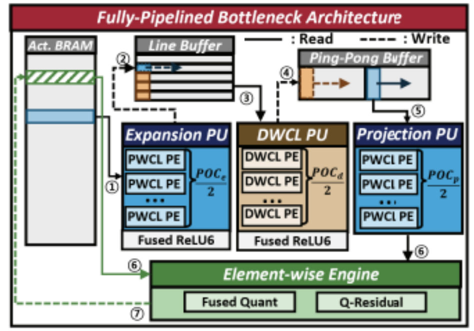
3 Utilize the FPB as the Computation Core for Batching

Deploy the FPB, capable of operating without external memory access, as the batch computation cores → Compensate for the drawbacks of batching while maximizing its benefits

Proposed row-wise pipelined dataflow



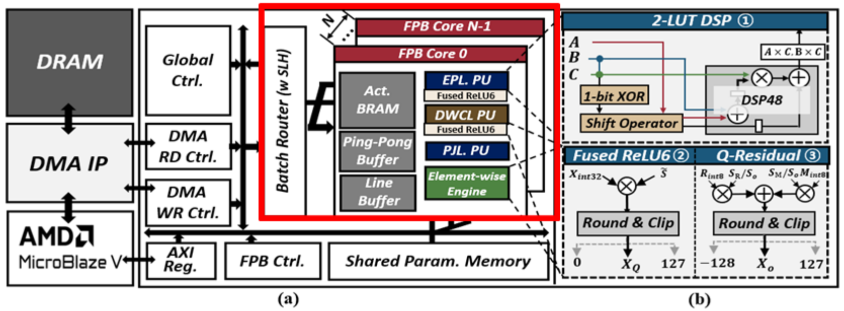
Block diagram of FPB



Performance evaluation with existing MobileNetv2 accelerators

	GPU	FPGA'22 (44)	FPGA'20 (45)	TRETS'24 (28)	TCAS-I'24 (31)	Batch 1	Ours Batch 2	Batch 4
Platform	RTX 2080ti	XC7Z020	XCK325T	XC7V690T	XC7V690T		XCVU9P	
Freq.(MHz)	-	100	200	150	200		200	
BRAMs	-	123	193.5	941.5	578	827.5	1005.5	1349.5
DSPs	-	208	704	2160	1024	517	1019	2041
LUTs (k) / FFs (k)	-	41.3/-	173.5/241.8	308.4/278.9	120.6/185	108.2/80.5	184.5/115.1	339.8/184.1
Bit-width / Data format	32/Float	Mixed/Fixed	8/Fixed	8/Int	Mixed/Int	-	8/Int	-
Speed (FPS)	1374.6	132.3	325.7	302.3	1496.6	325.2	645.8	1230.4
Throughput (GOPS)	942.9	78.7	98.56	181.8	939.8	204.2	405.6	772.7
Power (W)	102	3.5	8.57	11.35	11.6	4.98	5.52	6.4
Energy Efficiency (GOPS/W)	9.244	22.5	11.5	16.02	81	41.0	73.48	120.7
Top-1 Accuracy	72.93%	65.67%	-	70.8%	-		71.6%	

Overall architecture of proposed FAB accelerator



"FAB: FPGA-Accelerated Fully-Pipelined Bottleneck Architecture with Batching for High-Performance MobileNetv2 Inference," IEEE Transactions on Circuits and Systems I, 2025 (IF 5.2)

Demo: On-Device MobileNetv2 FPGA Accelerator

"FAB: FPGA-Accelerated Fully-Pipelined Bottleneck Architecture with Batching for High-Performance MobileNetv2 Inference," IEEE Transactions on Circuits and Systems I, 2025 (IF 5.2)

FPGA Design for CNN-based Object Detectors (1)

Goal

Achieve **optimal HW design for object detectors** by **reflecting the layer's characteristics & compensating for the accuracy loss**

Motivation

- Existing design uses a common HW organization scheme for all the layers (=not layer-specific), and causes significant accuracy drop

Solution/ Contribution

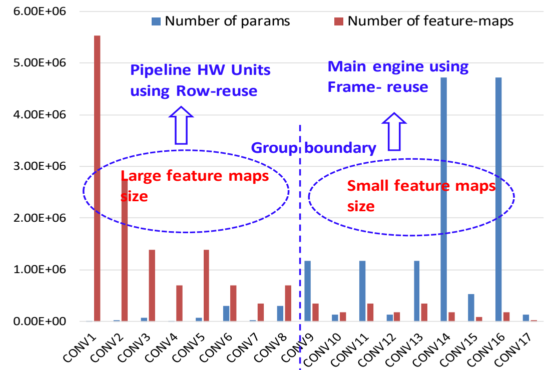
1 Mixed data flow

Row-based (pipelined HW structure) and frame-based (recursive HW structure) weight reuse schemes are applied to front (Group 1 w/ large feature-maps) and deep (Group 2 w/ large weights) layers, respectively

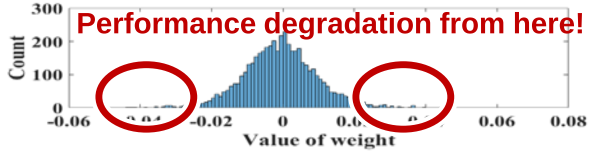
2 Mixed (Outlier-aware) precision quantization

Dense 1-bit for most small weights + Sparse 8-bit for only a few large weights (outliers)

Characteristics of each layer



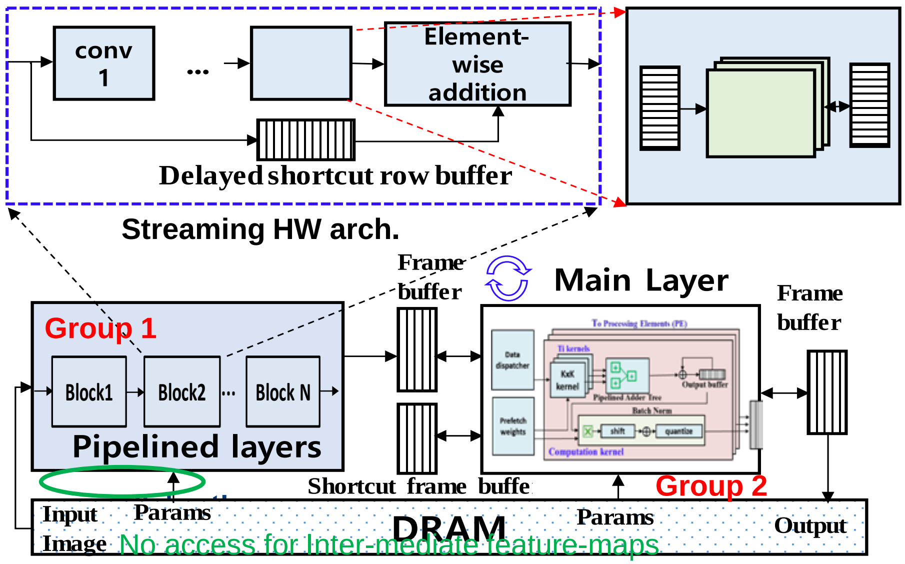
Weight distribution of YOLO



Performance Comparison with existing YOLO accelerators

	YOLO-GPU [1]	Tincy YOLO [2]	Lightweight YOLOv2 [3]	Proposed (Sim-YOLOv2)	Proposed (Layer Opt.)
Platform	GTX Titan X (16nm)	Zynq Ultrascale+ (16nm)	Zynq Ultrascale+ (16nm)	Virtex-7 VC707 (28nm)	Virtex-7 VC707 (28nm)
Freq.(MHz)	1 GHz	N/A	300 MHz	200 MHz	200 MHz
BRAMs (18 Kb)	N/A	N/A	1706	**1144**	**1245**
DSPs	N/A	N/A	377	272	829
LUTs – FFs	N/A	N/A	135K – 370K	155K – 115K	245K – 117K
CNN Size (GOP)	22.73	4.5	14.97	17.18	17.18
Precision (W,A)	(32, 32)	(1, 3)	(1-32, 1-32)	(1, 3-6)	(Mixed 1-8, 3-6)
Image Size	416 x 416	416 x 416	224 x 224	416 x 416	416 x 416
Frame Rate	88	16	40.81	109.3	109.3
Accuracy(%)	72.08	48.5	67.6	64.16	**71.13**
Throughput (GOPS)	1512	72	610.9	**1877**	**1877**
Efficiency (GOPS/kLUT)	N/A	N/A	4.52	**12.11**	**7.66**
Power(W)	170	6	N/A	18.29	N/A
Power Eff.(GOPS/W)	8.89	12	N/A	**102.62**	N/A

Mixed dataflow HW structure for YOLO



"A High-Throughput and Power-Efficient FPGA Implementation of YOLO CNN for Object Detection," IEEE TVLSI, 2019 (Citation: 473)

"Layer-specific Optimization for Mixed Data Flow with Mixed Precision in FPGA Design for CNN-based Object Detectors," IEEE TCSVT, 2021 (Citation: 91)

FPGA Design for CNN-based Object Detectors (2)

Goal

Design a low-power, high-throughput TinyYOLOv3 accelerator using HW/SW co-design solutions

Motivation

- There is a growing demand for dedicated dataflow and MAC operator for object detection on-device AI accelerators

Solution/ Contribution

1 Fully pipelined streamline architecture-based accelerator

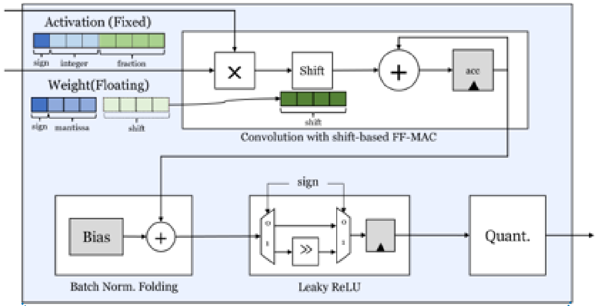
All operations with 8-bit W/A parameters are designed to be computed only with on-chip memory except for route layer operations

2 HW-friendly shift-based floating-fixed MAC

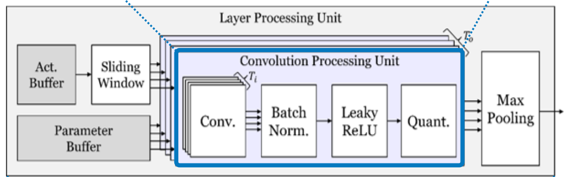
Proposed shift-based quantization performs adaptive linear quantization per layer, using a unified shift direction and shared shift value to reduce hardware resources and MAC complexity in shift-based FF-MACs

$\mathbf{shift} = \mathbf{AS} - (\mathbf{IS} + \boldsymbol{M}) - (\mathbf{E} - \boldsymbol{B})$ where AS: Accumulate Scale, IS: Input Scale, M: Mantissa bits, E/B: Weight/Bias's Exponent

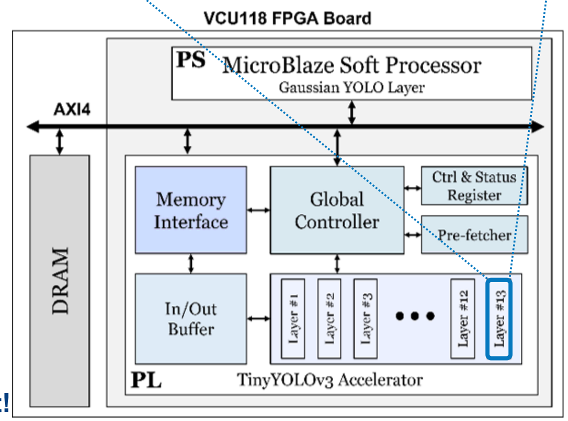
Convolution Processing Unit



Layer Processing Unit



Overall Architecture



Performance Comparison with existing TinyYOLOv3 accelerators

	[1]	[2]	[3]	[4]	[5]	Proposed
Year	2022	2020	2021	2021	2022	2023
Model	TinyYOLOv2	TinyYOLOv3	TinyYOLOv3	TinyYOLOv3	TinyYOLOv3	TinyYOLOv3
Platform	Xilinx XC7Z045	Xilinx XC7Z020	Xilinx XCKU040	Xilinx XCVU9P	Intel 10AX115	Xilinx XCVU9P
Freq.(MHz)	200	100	143	200	200	150
OCM(KB)	508.5	185	984	-	6,095	9,166
DSPs	448	160	839	2693	1122	96
LUTs(k)	99.4	25.9	139	17.7	146.1 (Altera ALMs)	132
FFs(k)	98.9	46.7	-	145.7	-	39.5
Image Size	416x416	416x416	416x416	416x416	416x416	416x416
Precision	16b	16b	16b	-	32b	8b
Accuracy(%)	-	30.9	-	-	33.1	**34.01**
FPS	-	1.88	32.4	32.1	36.3	**62.9**
Throughput(GOPS)	138.8	10.45	180	166.4	202	**351.1**
Power(W)	-	3.36	3.87	-	-	5.52
Power Eff.(GOPS/W)	-	3.11	46.51	-	-	**63.61**

Highest!

"Dedicated FPGA Implementation of the Gaussian TinyYOLOv3 Accelerator," IEEE TCAS-II, 2023 (IF 4.9, Citation: 37)

Demo: FPGA Design for CNN-based Object Detectors

"Dedicated FPGA Implementation of the Gaussian TinyYOLOv3 Accelerator," IEEE TCAS-II, 2023 (IF 4.9, Citation: 37)

Intelligent Digital Systems Design Lab.

SEOULTECH 서울과학기술대학교

On-Device Trainable Energy-Efficient FPGA Accelerator

Goal

On-device training accelerator for instance segmentation model

Motivation

Despite recent advances in CNN training accelerators, their application to instance segmentation remains largely unexplored

Solution/ Contribution

1 Separate architecture for CONV processing and auxiliary processing units

Two IPs acting as pipelines to achieve high hardware utilization

2 Reconfigurable Convolution Processing Unit (RC-PU)

Support various kernel sizes by dynamically switching between parallel and sequential modes, enabling efficient and flexible training for instance segmentation

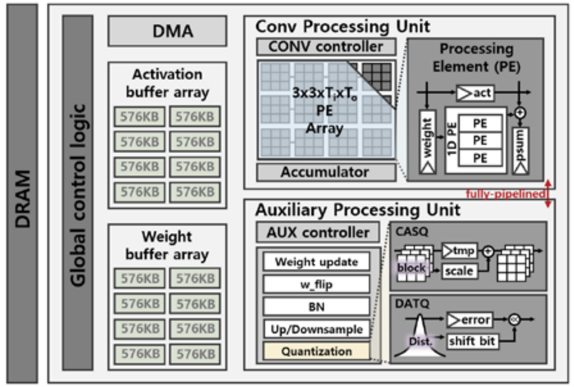
3 Dataflow that flexibly handles forward propagation and weight updates

Apply scale-smoothing and distribution -aware rounding to enable accurate INT8 training with minimal overhead

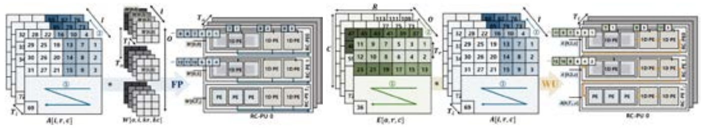
Comparison with Existing CNN Training Accelerators

	[32]	[33]	[34]	[21]	[27]	[22]	[35]	[23]	TETRIS
Device	XCVU9P	KCU1500	Stratix 10 MX	ZCU102	ZCU102	Alveo U50	ZC706	ZCU102	VCU118
Precision	FP16	INT8	FP16	BM(2,5)	FP32	PINT8	Fixed8	Fixed16	INT8
Frequency (MHz)	180	250	185	225	100	200	150	225	200
Dataset	ImageNet	CIFAR-10	CIFAR-10	CIFAR-10	ImageNet	ImageNet	CIFAR-10	CIFAR-10	Pascal SBD
Model	ResNet18	VGG-like	VGG-like	ResNet20	VGG-16	ResNet18	VGG-8	RepVGG-like	YOLACT
Resolution	224	32	32	32	224	224	32	32	550
Normalization	No	No	No	Yes	No	-	-	Yes	Yes
DSP	1216(18%)	1030(19%)	1046(26%)	502(20%)	1508(59.84%)	1024(17%)	130(14.44%)	864(34%)	1340(19.6%)
Logic Element[1]	432K(37%)	199K(30%)	221K(31%)	189K(69%)	-	273K(31%)	92.1K(42.1%)	158K(58%)	278K(11.7%)
BRAM[2]	-	1060(49%)	2998(44%)	1745(95.1%)	-	526(39%)	369.5(67.6%)	818(45%)	3136(31.9%)
Throughput (GOPS)	299.8	641.1	158.5	131.0	46.9	904.1	214.5	150.0	805.9
Power(W)	17.9	26.8	20.0	8.7	7.7	20.4	7.6	7.2	11.6
Energy eff. (GOPS/W)	16.7	24.0	7.9	15.1	6.1	44.3	28.2	20.8	69.5
Computational eff.[3]	-	-	-	10.6	5.3	7.9	-	-	344.5

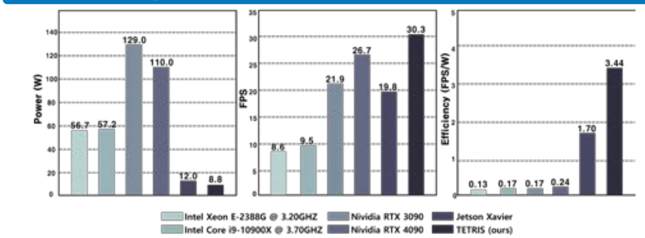
Overall architecture of TETRIS IP



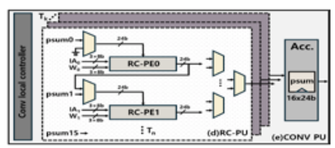
FW Propagation and Weight Update of CONV



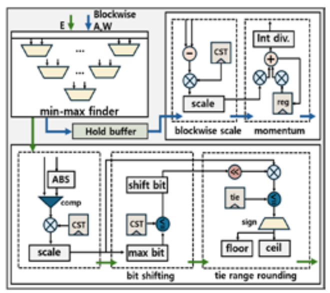
Comparison with Various Platforms



CONV PU Architecture



Quantizer in AUX PUs



"TETRIS: On-Device Trainable Energy-Efficient FPGA Accelerator for Trustworthy and Real-Time Instance Segmentation," ICCAD 2025 (BK21 Top-tier Conf.)

Demo: On-Device Trainable FPGA Accelerator

"TETRIS: On-Device Trainable Energy-Efficient FPGA Accelerator for Trustworthy and Real-Time Instance Segmentation," ICCAD 2025 (BK21 Top-tier Conf.)

Energy-Efficient FPGA Acceleration for HybridViTs

Intelligent Digital Systems Design Lab.

SEOULTECH 서울과학기술대학교

Goal

Energy-efficient FPGA accelerator for HybridViT with minimal accuracy degradation

Motivation 1

- HybridViTs combine CNNs and Transformers with different computational patterns, making optimization difficult

Motivation 2

- Existing accelerators lack dedicated solutions for HybridViTs and struggle with **quantization and nonlinear function** challenges

Solution/ Contribution

1 Integer-Only Inference

Fused quantization with single requantization step and fixed-point for residual connections

2 Linear Approximations

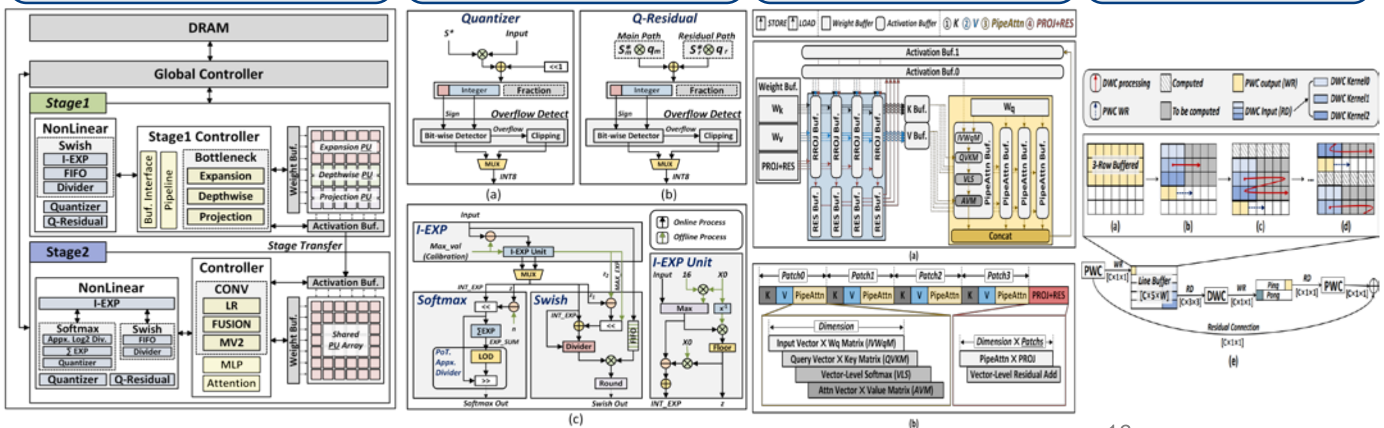
Hardware-friendly approximations for Swish and Softmax with shared units

3 Two-Stage Pipeline

Specialized architecture/dataflow with Stage 1 for CNN and Stage 2 for Transformer components

Proposed hardware architecture

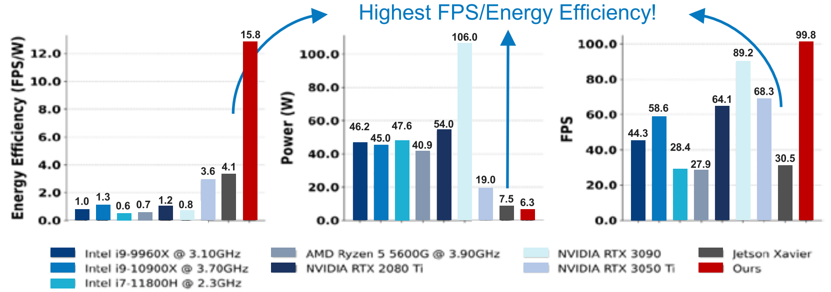
Overall HW architecture | Quantization & Approximation | Attention Dataflow | CNN Dataflow



Performance comparison with other accelerators

-	ME-ViT	VAQF	AutoVitAcc	VITA	HeatViT	Ours
Freq. (MHz)	150	150	150	150	150	100
FPS	13.20	31.6	25.9	2.75	109.2	99.83
Power (W)	6.5	7.8	9.4	0.88	10.7	6.31
FPS/W	2.03	4.06	2.76	3.12	10.2	15.83
FPS/DSP	0.013	0.047	0.012	-	0.056	0.085

Energy efficiency, power, fps comparison on CPU/GPU



HW resource utilization of approximation methods

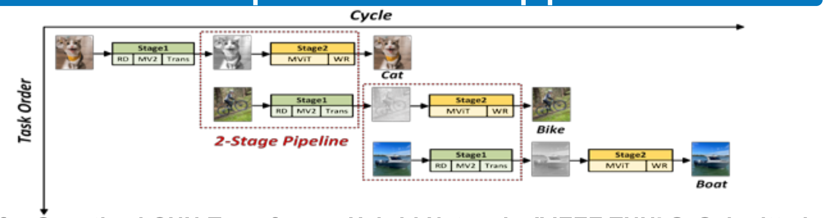
	Swish				Softmax			
	Base	I-ViT	HyQ	Ours	Base	I-ViT	HeatViT	Ours
FF	2321	1375	1549	267	3820	1091	1939	775
LUT	3887	2532	2900	1444	7183	1915	2364	1266
DSP	40	3	4	4	29	10	2	2

Lowest HW resource!

Accuracy comparison with SOTA ViT Quantization

Method	Bits (W/A)	Fully-Int?	Base Acc. (%)	Top-1 Acc. (%)	Acc. Drop (%)
HyQ	8/8	X	68.94	68.15	0.79
Q-HyViT	8/8	X	69.0	68.20	0.8
PTQ4ViT	8/8	X	69.0	37.75	31.25
Ours	8/8	✓	68.94	68.1	0.84

Example of the two-sate pipeline



"FLASH: Energy-Efficient FPGA Acceleration via Linear Approximation and Streamlined Two-Stage Pipeline Architectures for Quantized CNN-Transformer Hybrid Networks," IEEE TNNLS, Submitted

Real-Time ViT Inference on FPGA Platforms

IDSL Intelligent Digital Systems Design Lab.
SEOULTECH 서울과학기술대학교

Goal: Design an energy-efficient FPGA accelerator for ViT inference that achieves real-time performance on edge devices while maintaining high accuracy without fine-tuning

Motivation 1: MatMul dominates ViT computation and consumes excessive energy

Motivation 2: Power-of-Two quantization suffers severe accuracy degradation without fine-tuning

Motivation 3: Different ViT layers exhibit varying sensitivity to quantization Solution/Contribution

Solution/ Contribution

1 Adaptive Power-of-Two Rounding

Task-loss driven rounding optimization that minimizes layer-wise reconstruction error without fine-tuning

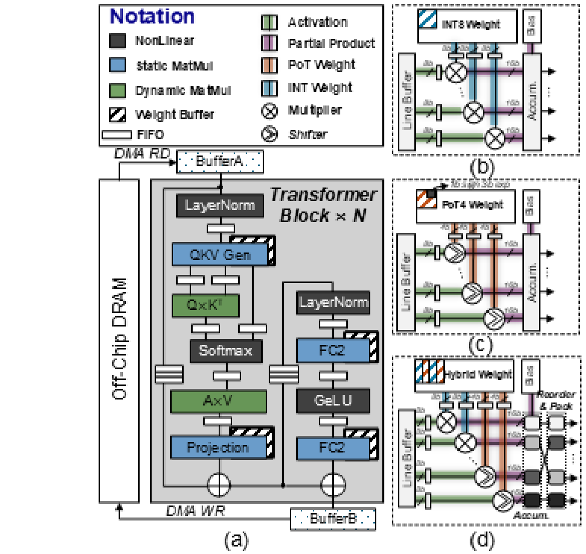
2 Sensitivity-Aware INT-PoT Mixed Quantization

Layer-wise mixed quantization scheme that selectively assigns optimal quantization (INT8/PoT4/ Hybrid) to each MatMul based on sensitivity analysis

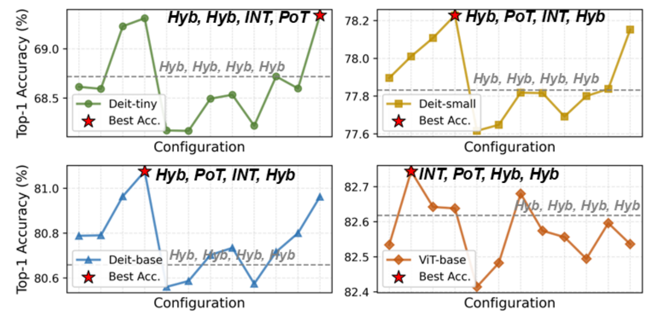
3 Fully-Pipelined Architecture with Dedicated Engines

Three specialized compute engines (MatINT, MatPoT, MathHybrid) integrated via FIFO-based pipeline to eliminate mode switching overhead, implemented using Vitis HLS with II=1 pipelining for sustained throughput

Overall pipelined architecture

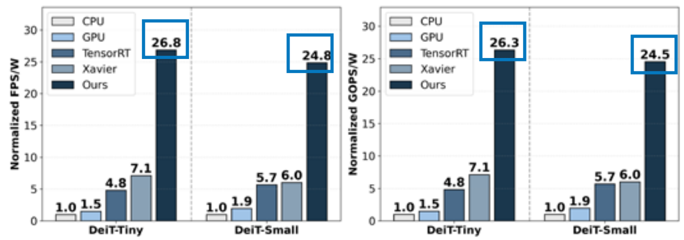


Accuracy comparison of mixed quantization config.



Comparison of FPS/W and GOPS/W on various HWs

The highest FPS/W & GOPS/W



HW performance comparison with SOTA ViT accelerators

	AVA	MoE-Sched		HeatViT		HEAT	ViA	GroupV	HDVIT	Ours	
Platform	ZCU102	ZCU102	AlveoU280	ZCU102	ZCU102	ZCU102	AlveoU50	ZCU102	AlveoU50	ZCU102	VCU128
Freq.(MHz)	150	300	250	150	150	300	300	300	300	300	300
Architec.	Temp.	Pipeline		Temporal		Temp.	Pipeline	Temp.	Pipeline	Pipeline	
Model	DeiT-S	ViT-T	ViT-S	DeiT-T	DeiT-S	Swin-T	Swin-T	ViT-S	DeiT-S	DeiT-T	DeiT-S
Prec.(W/A)	Mix/8	8/8	8/8	8/8	8/8	5.71/5.71	16/16	8/8	16/16	Mix/8	Mix/8
DSP	1552	1754	3352	1968	1955	768	2420	1268	3043	1421	5217
Power(W)	9.63	9.92	38.7	9.45	10.7	23.48	39	29.6	32.43	11.19	25.51
GOPS	907.8	455.3	1411	485.5	441.2	1023.3	309.6	762.7	1742.7	860.1	**2836.1**
FPS	99.7	182.1	153.4	183.4	109.2	118.9	-	89.76	189.4	**351.2**	314
FPS/W	10.35	18.36	3.96	19.41	10.21	5.06	-	3.03	5.84	**31.38**	12.30
GOPS/W	94.27	45.90	36.46	48.52	41.23	43.58	7.94	25.77	53.74	76.86	**111.18**

The highest FPS/W & GOPS/W

Comparison of accuracy and computation efficiency

Model	Method	Prec.(W/A)	#Mult(G)	#Shift(G)	Acc.(%)
DeiT-T	Baseline	32/32	1.22	0	72.13
	FQ-ViT	8/8	1.13	0.09	70.91
	P^2-ViT	8/8	1.13	0.09	69.60
	AVA	Mix/8	0.71	0.51	66.35
	Ours	Mix/8	**0.70**	**0.52**	**69.34**
DeiT-S	Baseline	32/32	4.52	0	79.83
	FQ-ViT	8/8	4.34	0.18	78.25
	P^2-ViT	8/8	4.34	0.18	77.56
	AVA	Mix/8	2.62	1.89	76.57
	Ours	Mix/8	**2.43**	**2.08**	**78.23**
ViT-B	Baseline	32/32	17.36	0	84.54
	FQ-ViT	8/8	17.00	0.35	82.50
	P^2-ViT	8/8	17.00	0.35	82.10
	AVA	Mix/8	10.08	7.28	78.56
	Ours	Mix/8	**9.03**	**8.32**	**82.74**

"SHARP: Shift-Aware Hybrid Power-of-Two Quantization-Assisted Real-Time Vision Transformer Inference on FPGA Platforms," DAC 2026, Submitted

LNS-MAC Systolic Array Design for HW-Efficient LLM Inference

Goal

A hardware-efficient, outlier-aware quantization framework combining the logarithmic number system (LNS) and Microscaling (MX)

Motivation

- MX quantization yields a homogeneous execution path across Transformer blocks—compact for on-device AI; However, low-precision MX quantization has the following limitations: (i) fixed bins, (ii) coarse shared exponents, and (iii) missing algorithm–hardware co-design

Solution/ Contribution

1 LNS base Dual-Bias

Use **Offline Block-Specific Dual-Bias for Weight Quantization (OBWQ)**, **Adaptive Dual-Bias for Activation Quantization (ADBQ)** to adaptively reshape bins and accommodate blocks skewed by outliers

2 LNS Scaling

Apply a **finer-grained LNS-based scaling factor** than shared-exponent scaling to better match per-block distributions

3 Systolic Array

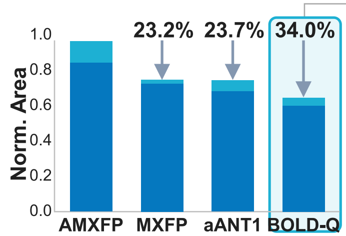
Deploy an **adder-based LNS–MAC systolic array**, improving compute and memory efficiency for on-device execution

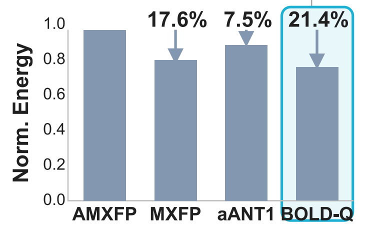
Area for BOLD-Q and baseline architectures

Arch.	Core: Component (µm²)	Core: Num.	Core: Area (mm²)	Others: Dequant. (mm²)	Others: Quant. (mm²)
AMXFP	PE (521.35)	1024	0.542	0.03	0.039
	Encoder (243.83)	32			
MXFP	PE (443.20)	1024	0.458	0.003	0.008
	Encoder (121.91)	32			
ANT	PE (423.77)	1024	0.432	0.017	0.017
	Decoder (12.64)	32			
	Encoder (121.91)	32			
BOLD-Q (Ours)	PE (372.41)	1024	0.383	0.003	0.017
	Preproc. (89.97)	32			
	Encoder (128.00)	32			

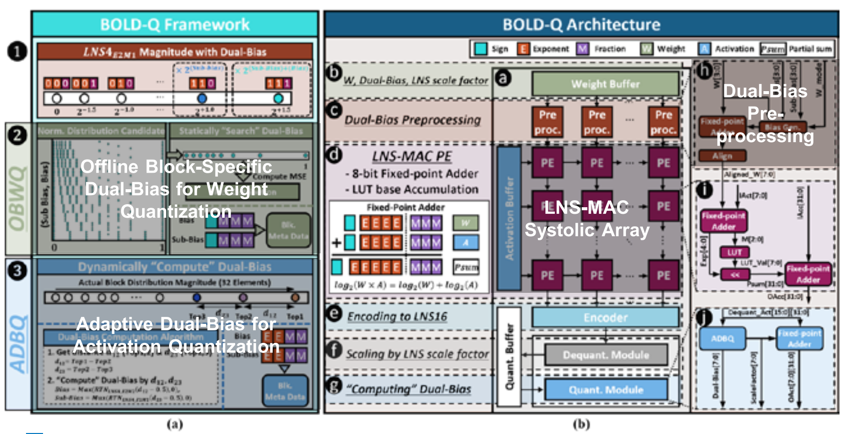
Lowest Area, energy!

Comparison of normalized area and energy





Framework & Architecture of BOLD-Q



Quantization Results for Various LLMs on WikiText-2 (perplexity ▼)

Method	Linear (W/A)	Atten. (W/A)	Scale Factor	Block Size	OPT 6.7B	OPT 13B	LLAMA1 7B	LLaMA1 13B	LLaMA2 7B	LLAMA2 13B	LLAMA3 8B	Mistral 7B
Baseline	16/16	16/16	-	-	10.86	10.13	5.68	5.09	5.47	4.88	6.14	5.25
GPTQ	4/16	16/16	FP16	128	10.93	10.17	5.83	5.20	5.63	4.99	6.56	5.39
AWQ	4/16	16/16	FP16	128	10.93	10.21	5.78	5.19	5.60	4.97	6.54	5.37
AMXFP	4/16	16/16	FPS	32	10.96	10.34	5.84	5.21	5.64	4.98	-	5.39
BOLD-Q	4/16	16/16	LNS8	32	10.89	10.16	5.76	5.15	5.55	4.95	6.40	5.32
Qserve	**4/8**	**4/16**	FP16	128	-	-	**5.89**	**5.25**	**5.70**	**5.08**	**6.76**	**5.42**
MXFP	4/8	4/8	POT8	32	12.27	11.64	6.81	5.91	6.75	6.01	-	5.88
AMXFP	4/8	4/8	FPS	32	11.40	10.99	5.99	5.35	5.85	5.20	-	5.48
BOLD-Q	**4/8**	**4/8**	LNS8	32	11.30	10.74	**5.97**	**5.28**	**5.79**	**5.14**	**6.70**	**5.41**
Quarot	**4/4**	**4/16**	FP16	Channel	-	-	**6.34**	**5.58**	**6.10**	**5.40**	**8.17**	**5.80**
ATOM	**4/4**	**4/16**	FP16	128	-	-	**6.16**	**5.46**	**6.12**	**5.31**	**7.76**	**5.76**
MXFP	4/4	4/4	POT8	32	22.51	12.88	10.20	7.80	11.18	6.98	11.17	9.43
AMXFP	4/4	4/4	FPS	32	13.06	11.90	6.25	5.52	6.22	5.47	7.72	5.71
BOLD-Q	**4/4**	**4/4**	LNS8	32	11.45	10.97	**6.18**	**5.45**	**6.07**	**5.37**	**7.34**	**5.56**

"BOLD-Q: Blockwise Outlier-aware Logarithmic Dual-Bias Quantization for Hardware-Efficient LLM Inference," DATE 2026 (BK21 Top-tier Conf.),

FP-INT GEMM Acceleration with Binary-Coding Quantization

Goal

To enable efficient LLM inference by reducing FP multiplications through SPoT-based binary-coding quantization (BCQ) and FP-INT HW acceleration

Motivation 1

- FP-INT format mismatches occurring from weight-only quantization cause hardware inefficiency due to costly FP multipliers and converters

Motivation 2

- BCQ minimizes FP multiplications but still suffers from limited representational capacity and residual FP multiplications caused by scaling factors

Solution/ Contribution

1 Multiplication-free BCQ algorithm

A mixed-precision quantization replacing FP multiplications with SPoT-based shift-add operations, achieving 1.13× lower perplexity with 1.1-bit precision

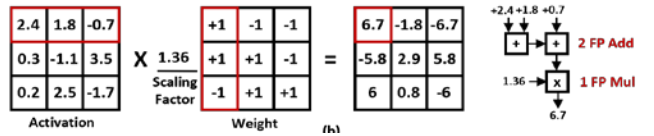
2 SS-BCQ supported HW-friendly module

An optimized HW design reducing activation-scale operations by 95.3% and efficiently managing salient weights in mixed-precision quantization

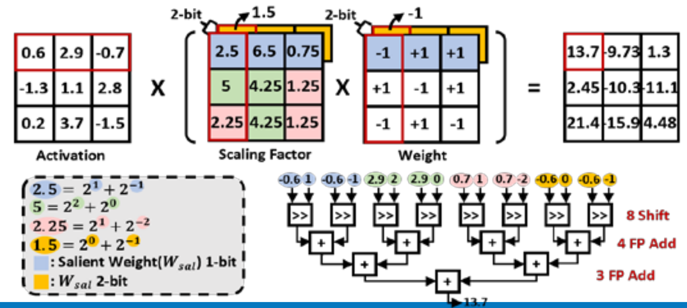
3 HW-SW co-optimization architecture

A fully pipelined FP–INT GEMM engine achieving 2.18× higher energy efficiency and 1.41× higher area efficiency over state-of-the-art accelerators

Operation Process of Naïve BCQ



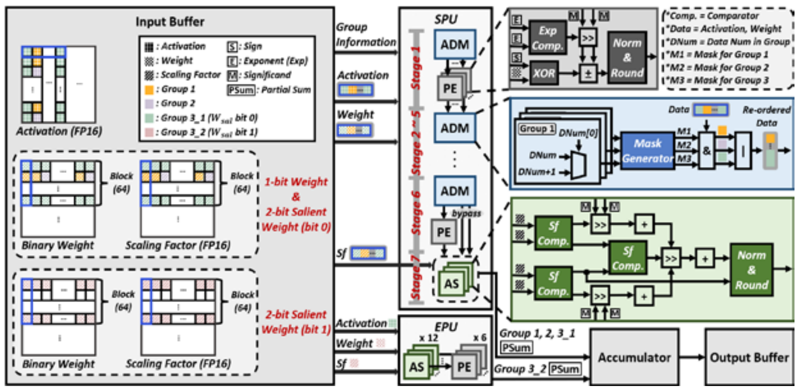
Operation Process of SS-BCQ



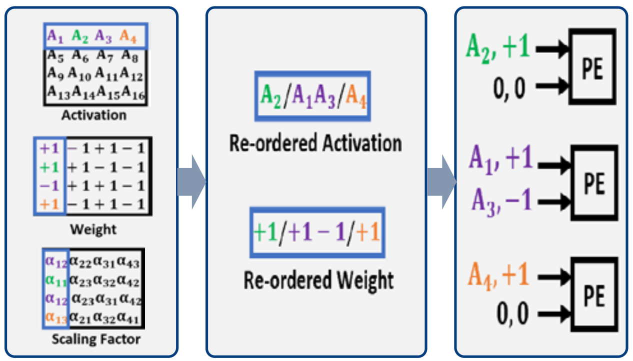
Comparison of FP GEMM Hardware design

	CORE	FPU	IFPU	FIGNA	FIGLUT
Quantization	BCQ 1.1-bit	-	BCQ 4-bit	PTQ 4/8-bit	BCQ 4-bit
Operation	FP Add	FP MAC	FP MAC	INT MAC	FP Add
Scaling Factor	SPOT	-	FP	-	POT
Normalized TOPS/W	**3.48**	1	0.7	1.12	1.6
Normalized TOPS/mm²	**4.17**	1	1.16	2.17	2.95

Overview of proposed CORE architecture



Processing flow of ADM module



Perplexity result of LLM Quantization Method

	Bits	OPT-1.3B	OPT-2.7B	OPT-6.7B	Llama-7B	Llama2-7B
CORE-S	1.1	**45.44**	**34.56**	**24.73**	**15.9**	**14.73**
CORE-P	1.1	72.68	50.43	35.53	25.29	14.82
BILLM	1.1	47.13	37.4	27.87	16.28	15.13
FIGLUT	4	67.95	35.46	24.13	-	-
OPTQ	2	4737.05	6294.68	442.63	68.60	19.92
AWQ	2	9472.81	2.29c + 4	8168.30	2.6c + 5	2.2c + 5
RTN	2	1.13e + 4	9505.76	2.84c + 4	1.07c + 5	1.78c + 4
	1	1.72c + 4	3.65c + 4	1.16c + 4	1.68c + 5	1.57c + 5
GPTQ	2	115.17	61.59	50.19	152.31	60.45
	1	1.49c + 4	1.41c + 4	1.06c + 4	2.67c + 5	1.16c + 5

"CORE: Compact, Optimized, and Resource-Efficient Compute Engine for FP-INT GEMM Acceleration with Binary-Coding Quantization," DAC 2026 (BK21 Top-tier Conf.), Submitted

Hardware-friendly Fully-Integer Approximation of Nonlinear Functions

Goal

Hardware-friendly Fully-Integer Approximation of Nonlinear Functions for Efficient Deployment of Quantized CLIP-ViTs

Motivation 1

Nonlinear functions become the dominant resource and power bottleneck on FPGA/ASIC accelerators

Motivation 2

High-precision approximation consumes excessive logic/DSP blocks, while low-precision approximation suffers significant accuracy loss and require costly fine-tuning

Solution/ Contribution

1 Optimal input clipping for PWL approximation

Derive optimal clipping bounds for PWL, improving LUT efficiency and approximation accuracy

2 Multi-PoT PWL slope conversion

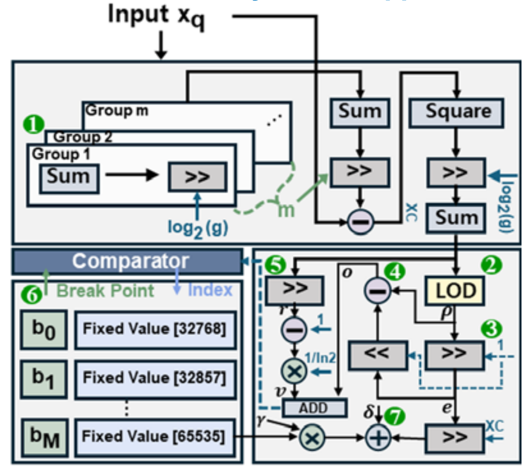
Convert PWL slopes to multi-PoT form, enabling shift-based operations without accuracy loss from single-PoT conversion

3 Linear interpolation for fractional parts

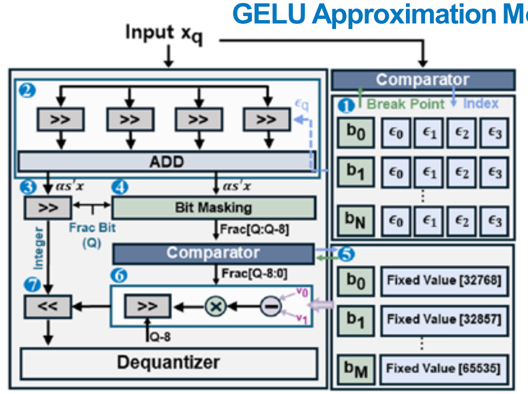
Incorporate linear interpolation to minimize LUT overhead in fixed-point PoT approximation

Overview of proposed HI-APP Architecture

LayerNorm Approximation Module



GELU Approximation Module



Accuracy comparison with SOTA Approximations

Model	Method	Approximations	Accuracy(%)
CLIP ViT-B/32	Baseline	-	62.35
	QUNF	-	55.93
	PEANO-ViT	ALL	44.31
	GQA-LUT	ALL	43.86
	HI-APP (Superior accuracy)	GELU+LayerNorm	**61.54**
		ALL	**61.42**
CLIP ViT-L/14	Baseline	-	74.57
	QUNF	-	69.92
	PEANO-ViT	ALL	66.06
	GQA-LUT	ALL	57.43
	HI-APP (Superior accuracy)	GELU+LayerNorm	**74.16**
		ALL	**74.03**

HW resource utilization of approximation methods

Functions	Method	DSP(Diff.)	LUT(Diff.)	FF(Diff.)
GELU	Baseline	247	17914	22368
	QUNF	12 (-95.1%)	6569 (-63.3%)	1924 (-91.4%)
	PEANO-ViT	16 (-93.5%)	**2940 (-83.5%)**	2951 (-86.1%)
	GQA-LUT	15 (-93.9%)	4290 (-76.05%)	**1313 (-94.1%)**
	HI-APP (Lowest HW resource Usage)	**0 (-100%)**	8722 (-51.3%)	2659 (-88.1%)
Layer Norm	Baseline	51	24609	29831
	QUNF	46 (-9.8%)	7661 (-68.9%)	5720 (-80.8%)
	PEANO-ViT	52 (+1.9%)	8157 (-66.8%)	8621 (-71.1%)
	GQA-LUT	46 (-9.8%)	13985 (-43.2%)	5371 (-81.9%)
	HI-APP (Lowest HW resource Usage)	**42 (-17.6%)**	**7440 (-69.8%)**	**1193 (-96.0%)**

Error comparison with SOTA Approximations

Method	GELU		Swish		LayerNorm	
	MSE	MAE	MSE	MAE	MSE	MAE
QUNF	1.68e-4	8.62e-3	1.65e-4	8.62e-3	5.96e-3	4.19e-2
PEANO-ViT	2.78e-4	1.38e-2	-		6.95e-3	4.07e-2
I-ViT	1.57e-3	1.91e-2	5.74e-3	3.47e-2	2.08e-2	5.16e-2
GQA-LUT	1.85e-4	1.12e-2	-		2.46e-3	2.49e-2
HI-APP (Lowest approximation error)	**5.46e-5**	**6.33e-3**	**8.58e-5**	**6.33e-3**	**1.54e-3**	**2.11e-2**

"HI-APP: Hardware-friendly Fully-Integer Approximation of Nonlinear Functions in Quantized CLIP-ViTs," DATE 2026

IDSL
Intelligent Digital Systems Design Lab.
서울과학기술대학교
SEOULTECH

Dynamic-Precision LUT-based Approximation Unifying Non-Linear Operations in Transformers

Goal

To achieve both high hardware efficiency and low error in approximating non-linear operations in transformers

Motivation 1

- Inefficient hardware implementations of non-linear approximation can become hardware bottleneck

Motivation 2

- Excessive focus on hardware efficiency can degrade model accuracy due to large approximation error

Solution/ Contribution

1 Dynamic Fixed-point Format (DFF)

A dynamically adjusts fraction bit-width based on data magnitude, leveraging 8-bit integer arithmetic

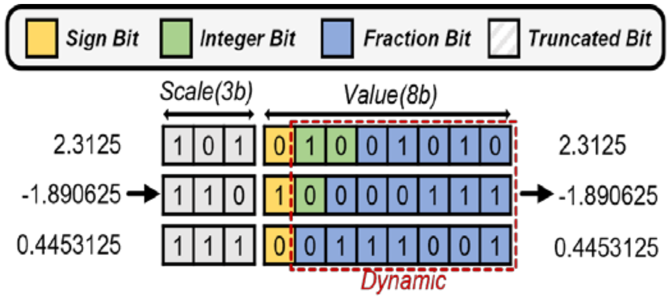
2 Genetic Adaptive Differential Evolution (GADE)

Optimize segments in piece-wise linear approximation using parallel mutants by progress, minimizing approximation error for a given LUT size

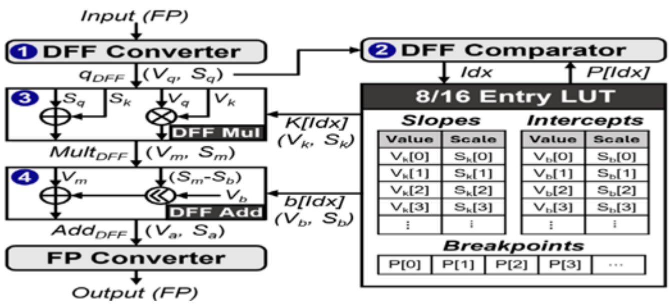
3 Non-linear approximation module using DFF

Enable a unified INT8 multiply-add datapath, reducing computational module size

Dynamic fixed-point format overview



Architecture of DFF approximation module



MSE comparison of LUT-based methods

Method	Setting	EXP (-9,0)	RECI (0.01,128)	RSQRT (0.01,128)	GeLU (-6,6)	SiLU (-6,6)	Avg.
RI-LUT	1 Entry 16 Entry	1.67e-5 5.56e-9	9.53e-4 7.49e-8	1.91e-4 2.23e-8	2.55e-3 2.54e-3	1.34e-2 1.33e-2	3.42e-3 3.17e-3
Q-HyViT	d=2, n=4 d=2, n=6	1.82e-5 1.19e-6	9.23e-5 1.45e-5	2.08e-6 1.55e-7	1.41e-3 1.31e-3	1.90e-3 1.80e-3	6.85e-4 6.25e-4
PTQ4ViT	8 Entry 16 Entry	2.24e-3 2.11e-3	8.05e-1 8.05e-1	4.24e-2 4.25e-2	5.02e-2 4.07e-2	4.41e-2 4.47e-2	1.89e-1 1.87e-1
Ours	8 Entry 16 Entry	1.35e-5 3.55e-6	7.94e-5 7.11e-5	9.94e-7 9.42e-7	2.90e-4 2.80e-4	1.62e-4 9.12e-5	**1.09e-4** **9.09e-5**

Inference performance in large language models

Model	Method	Setting	WIKI PPL ↓	Avg. Acc. ↑ (%)
LLaMA2-7B	Baseline	-	5.477	69.00
	RI-LUT [21]	1/16 Entry	6.628/6.307	64.49/65.32
	QUNF [23]	d=2, n=4/6	5.500/5.490	68.91/68.83
	Ours	8/16 Entry	**5.481/5.477**	**68.98/68.99**
Falcon-7B	Baseline	-	6.631	67.76
	RI-LUT [21]	1/16 Entry	9.042/8.331	63.09/64.09
	QUNF [23]	d=2, n=4/6	6.639/6.631	67.67/67.68
	Ours	8/16 Entry	**6.695/6.630**	**67.97/67.74**
Mistral-7B	Baseline	-	5.154	74.08
	RI-LUT [21]	1/16 Entry	6.391/6.062	68.95/70.04
	QUNF [23]	d=2, n=4/6	5.177/5.170	73.93/74.23
	Ours	8/16 Entry	**5.162/5.160**	**74.19/74.30**

Hardware performance in 28nm process

Method	Setting	Area (μm²)	Power (mW)	LUT size (Kb)
RI-LUT	1 Entry 16 Entry	1,360 2,337	0.48 0.84	0.096 2.256
Q-HyViT	d=2, n=4 d=2, n=6	2,579 3,433	0.65 0.88	5.802 5.802
PTQ4ViT	8 Entry 16 Entry	904 1,425	0.29 0.52	1.928 4.040
Ours	8 Entry 16 Entry	818 1,332	**0.27** **0.58**	**1.160** **2.360**

"LUT-APP: Dynamic-Precision LUT-based Approximation Unifying Non-Linear Operations in Transformers," DATE 2026

IDSL Intelligent Digital Systems Design Lab.

SEOULTECH 서울과학기술대학교

FPGA Acceleration of Sparsity-Tailored MoE-LLM

Goal

Design of a high-efficiency FPGA accelerator for MoE-LLMs addressing on-demand expert routing and huge expert parameter issues

Motivation 1

- High offloading latency caused by massive expert parameters results in severe hardware underutilization

Motivation 2

- The distribution of salient weights in MoEs necessitates unstructured pruning, requiring a dedicated hardware architecture

Solution/ Contribution

1 Unstructured Pruning with Pipeline Engines

Propose an importance-aware unstructured pruning method and a pipelined engine that processes MoE routing and pruning in parallel

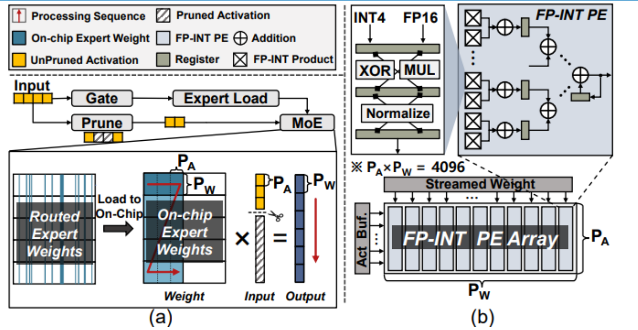
2 Dynamic HBM Address Generation

Design a dynamic address generation algorithm to handle the dynamic memory access patterns of MoE and pruning, achieving 78% HBM bandwidth utilization

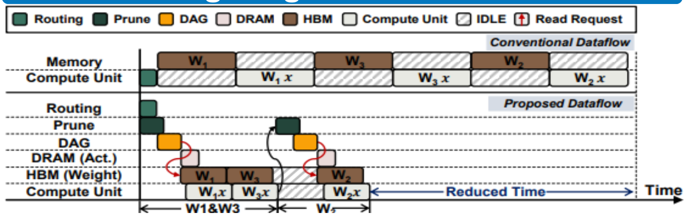
3 Asynchronous HBM interface and tightly coupled with PE arrays

Implement an asynchronous HBM interface operating at 2x system frequency with a bandwidth-optimized PE array, resulting in 69% hardware utilization

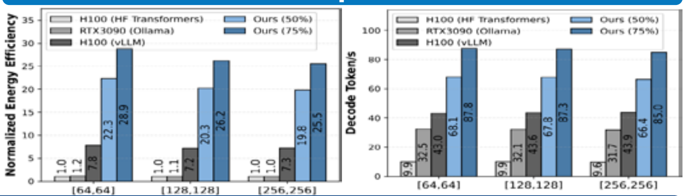
Proposed dataflow & FP-INT PE of FAST-MoE



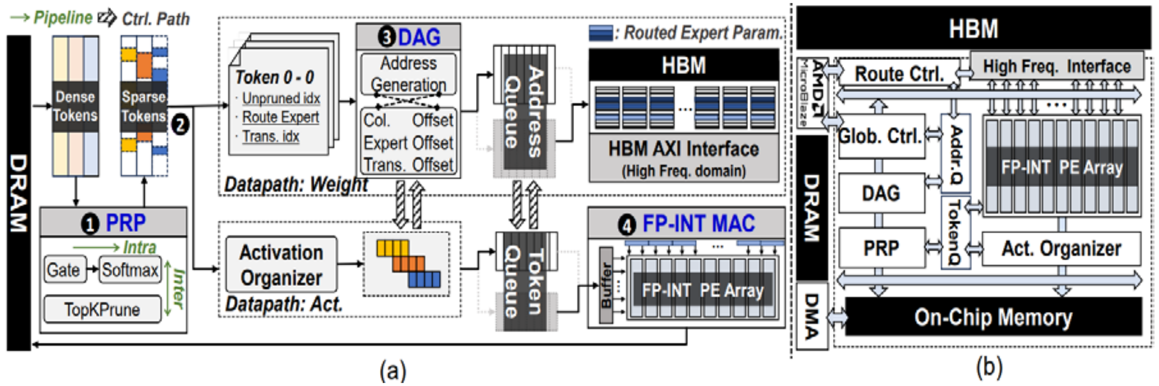
Timing Diagram of FAST-MoE



Performance Comparison with GPUs



Overall Architecture of FAST-MoE



Performance Comparison with FPGA-based Accelerations

	LLM Target		MoE-LLM Target	
	FlightLLM (FPGA'24)	Edgellm (TCAS-I'25)	FLAME (DAC'24)	FAST-MoE (Proposed)
Platform	Alveo U280	VCU128	Alveo U200	VCU128
Frequency(MHz)	225	140	-	150/300
Model (Size)	LLaMA2(7B)	GLM(7B)	Switch(7B)	Mixtral(7B)
LUTs(k)/FFs(k)	574/943	967/607	-	909/933
BRAMs	5634	7803	-	1966
DSP	6345	4563	-	4136
Throughput(Token/s)	55	69.4	86.78	**87.8**
Power(W)	45	56.8	-	**31.3**
Energy Eff.(Token/J)	1.22	1.23	-	**2.81**
Memory BW Util.(%)	65.9%	75%	-	**78%**

"FAST-MoE: FPGA-Acceleration of Sparsity Tailored MoE-LLM with Dynamic Unstructured Pruning-and-Address Scheduling," DAC 2026, Submitted