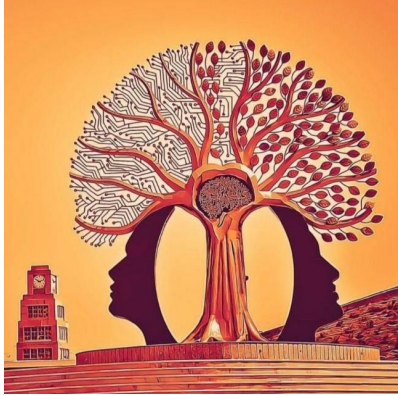
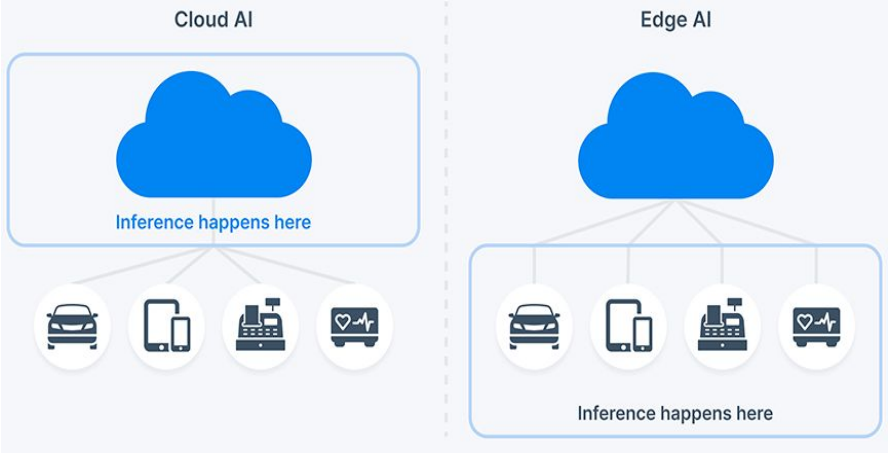


Hardware Software Co-design for Edge AI Acceleration

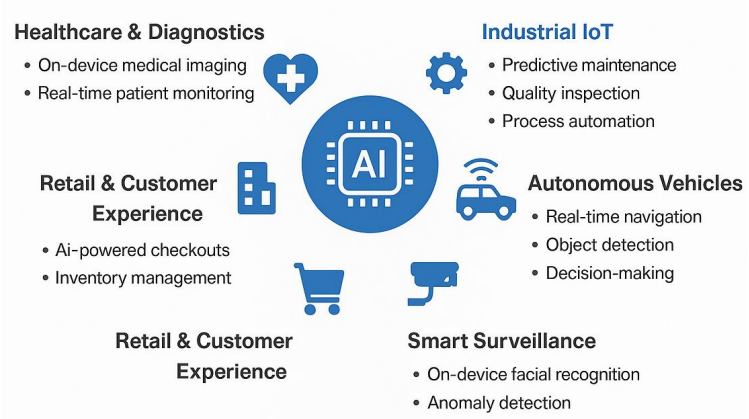
Binod Kumar
Electrical Engineering, IIT Jodhpur

SPARC SwHwCoD Workshop (17–23 December 2025)

Introduction

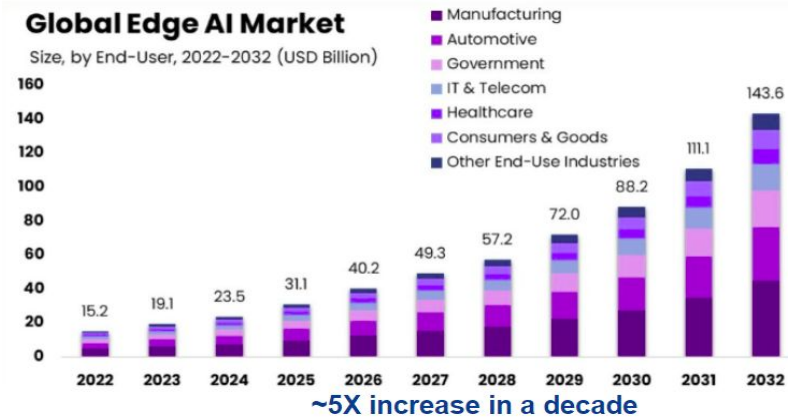


Cloud vs Edge Computing Paradigm

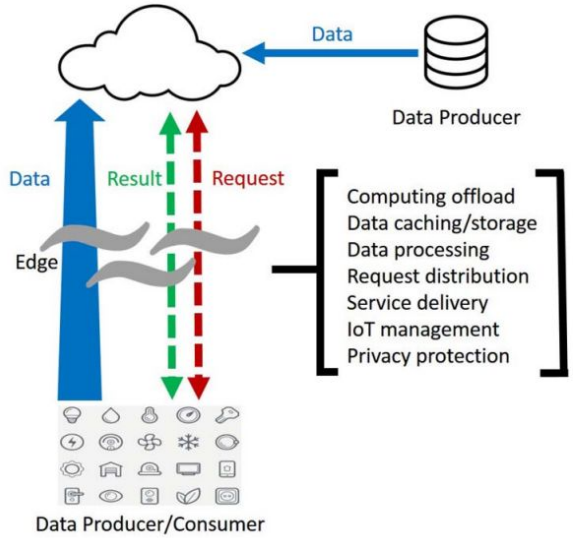


Applications of Edge-AI

- **Real-Time Processing:** AI inference on-device, eliminating cloud delays.
- **Enhanced Privacy:** Local data handling for better security.
- **Key Uses:** Autonomous vehicles, smart surveillance, predictive IoT, healthcare, and industrial automation



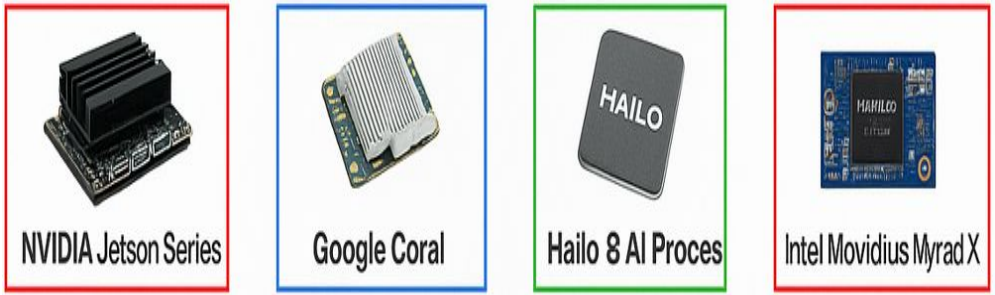
Introduction to Edge-AI



Edge Computing Paradigm

Edge Inference:

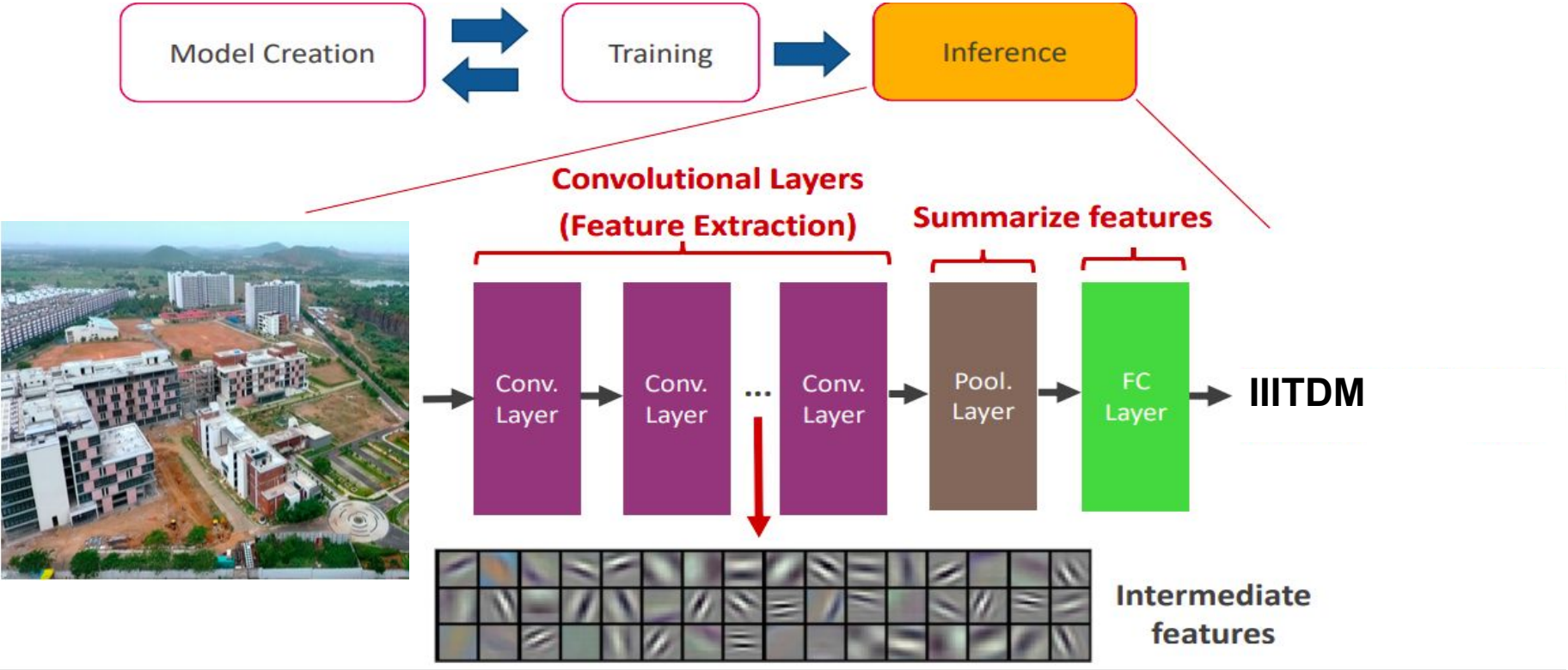
- **Edge inference**: Execution of AI algorithm at edges.
- **On-Device AI Processing:** Executes AI models locally on edge devices without relying on cloud servers.
- **Low Latency & Real-Time Response:** Minimizes data transfer delays for time-critical applications.
- **Enhanced Privacy & Security:** Sensitive data remains on-device, reducing exposure to external networks.



Commercial edge devices.

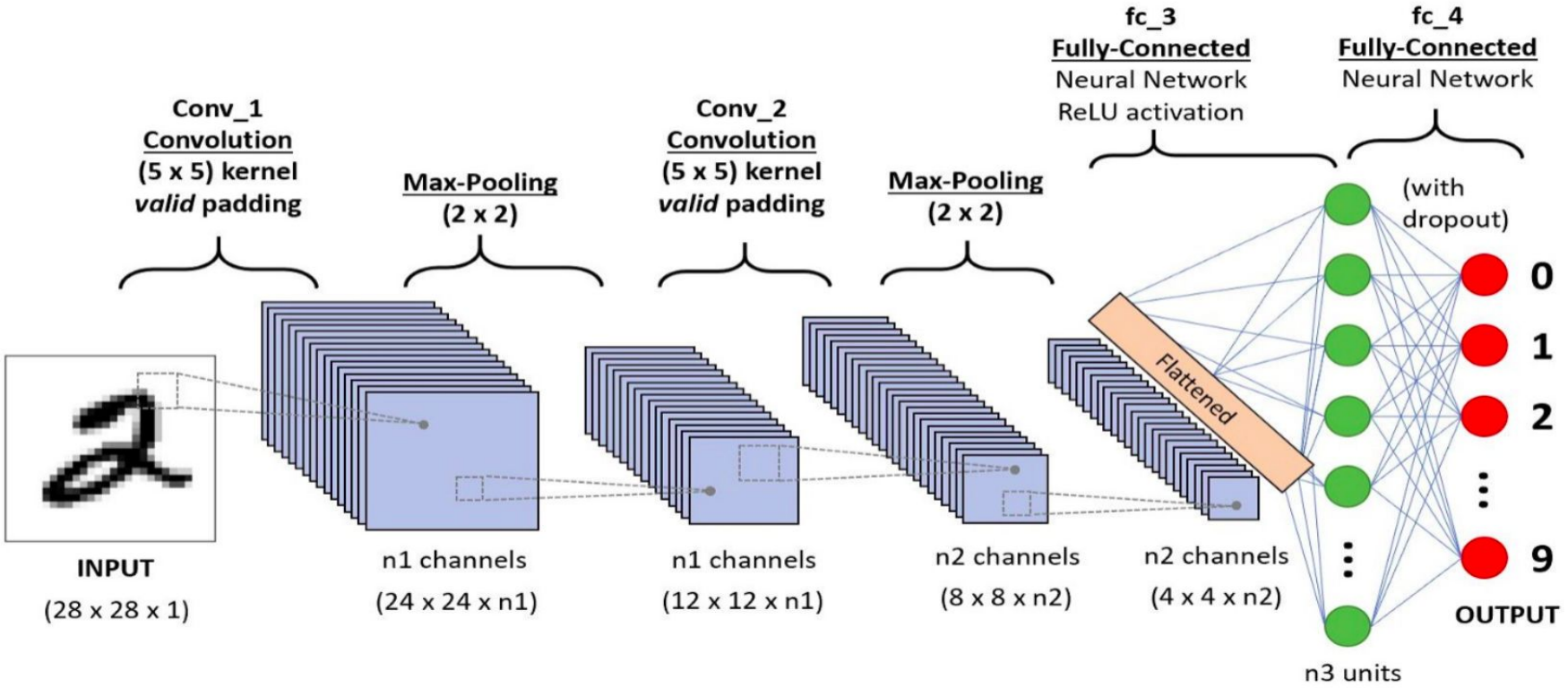
Adapted from W. Shi, J.Cao, Q Zhang et al. Edge Computing: Vision & Challenges. In IEEE Internet of Things Journal, 2018.

AI Task Example: Image Recognition



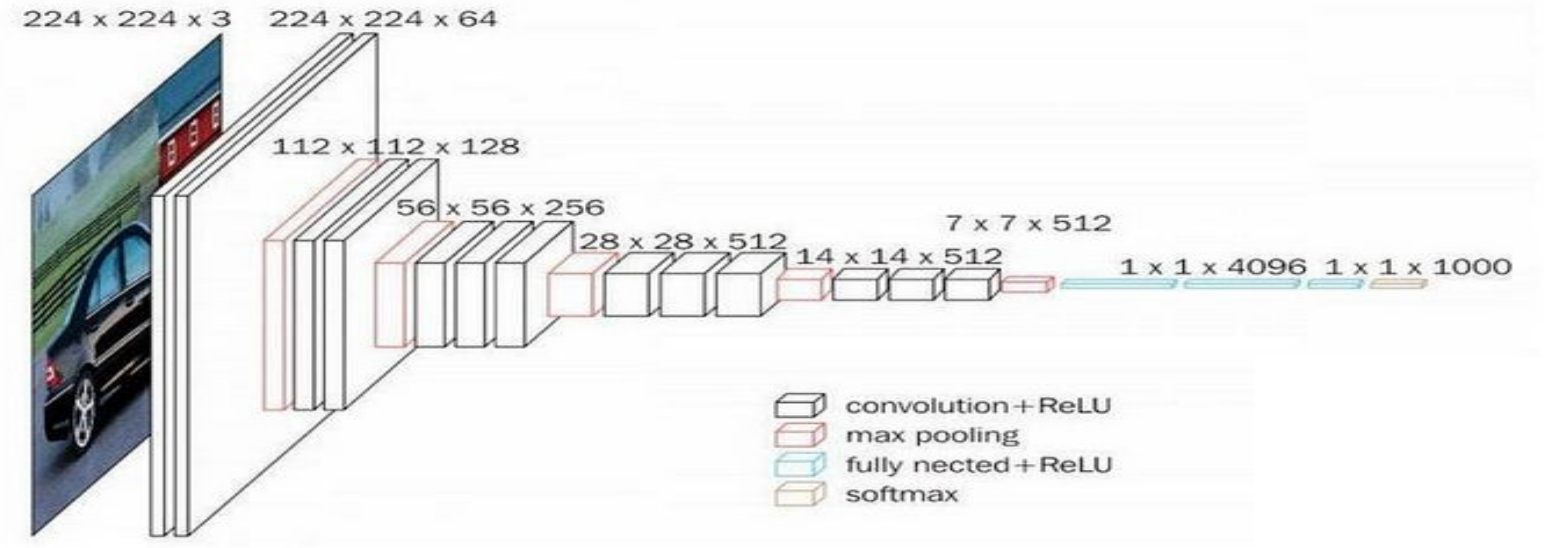
Adapted from Scale-sim Tutorial slides (ASPLOS 2021)

Convolution Neural Network (CNN) Example



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Another Example: VGG-16 Architecture



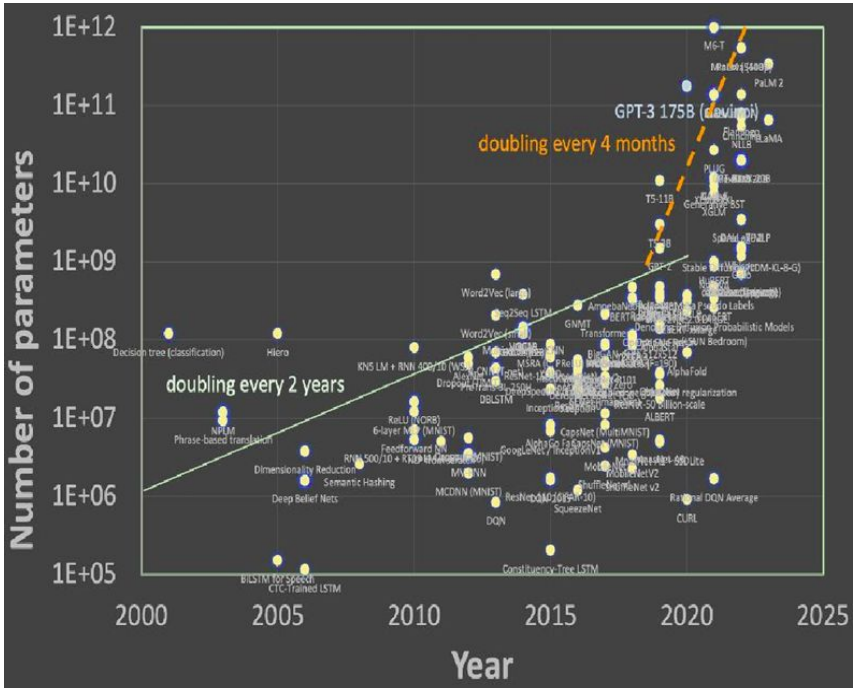
Motivation for Edge AI Accelertion

- **Rising Model Complexity:** Deep learning workloads demand new high-performance and low-power architectures.
- **Efficient Accelerator Design:** Need for reconfigurable, energy-efficient AI cores with hardware-software co-design.
- While it's acceptable for Deep Neural Network (DNN) to utilize high-end GPUs for training,requiring such powerful processors for inference,is highly limiting.
- Applications to various new and battery constrained technologies necessitate low-compute environments:

  a) Mobile phones

  b) Unmanned Aerial Vehicles(UAVs)

  c) IoT devices etc.



Increase in Complexity of Deep Learning Models

Adapted from https://ourworldindata.org/scaling-up-ai' [Online Resource].

Challenge-1: In AI (neural network) Computations

- **Millions of Parameters (i.e., weights)**
  - Billions of computations → **Need lots of parallel computations**

DNN Topology	No. of Weights
EfficientNet B0 (2019)	5.3M
GPT-3 (2020)	175B
Swin Transformer (2021)	3B
Palm (2022)	540B
Llama 2 (2023)	7B, 13B, 70B
Llama 4 (2024)	17B~109B

This makes CPUs inefficient

PE: Processing Element

RF: Register File

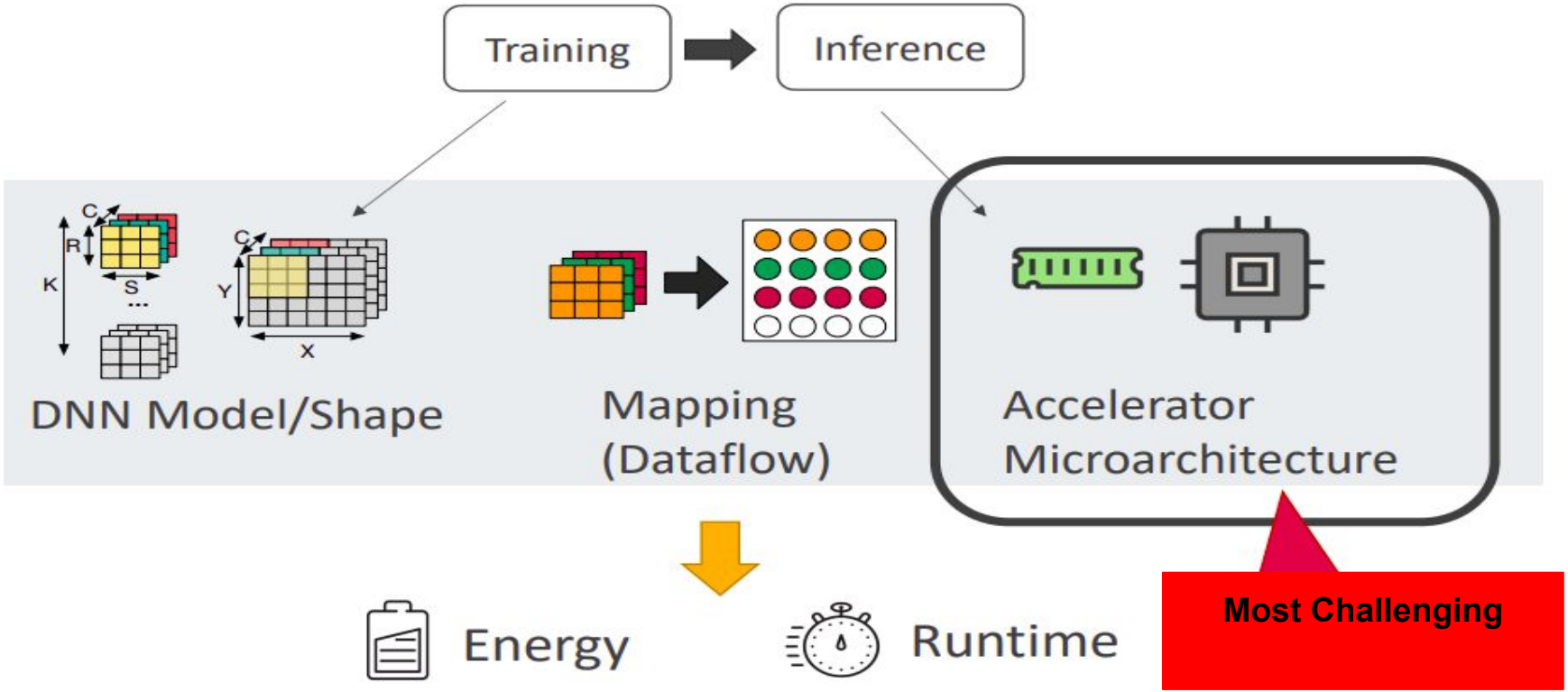
DRAM: Dynamic Random Access Memory

  - Heavy data movement → **Need to reduce energy**

This makes GPUs inefficient

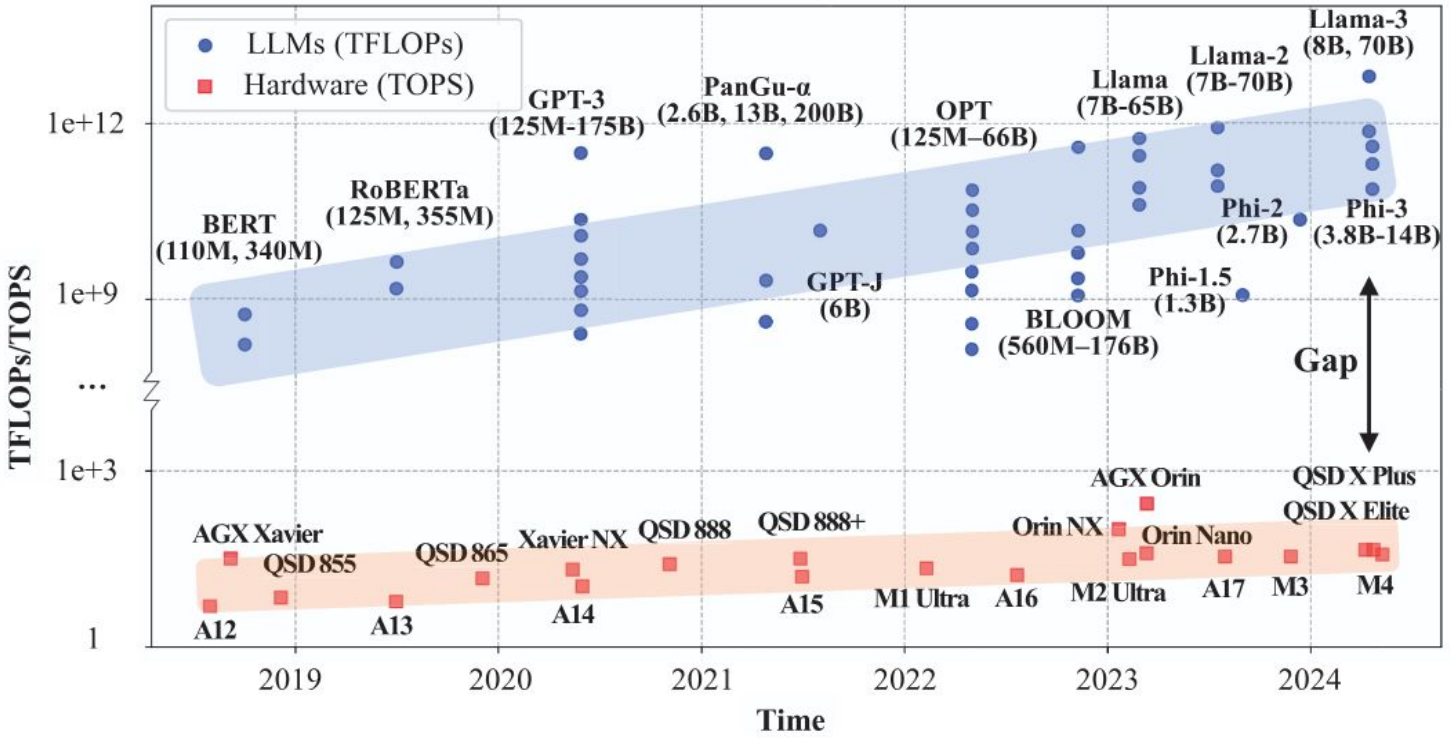
Adapted from Scale-sim Tutorial slides (ASPLOS 2021)

Challenge-2: In Edge AI Hardware Design & Deployment



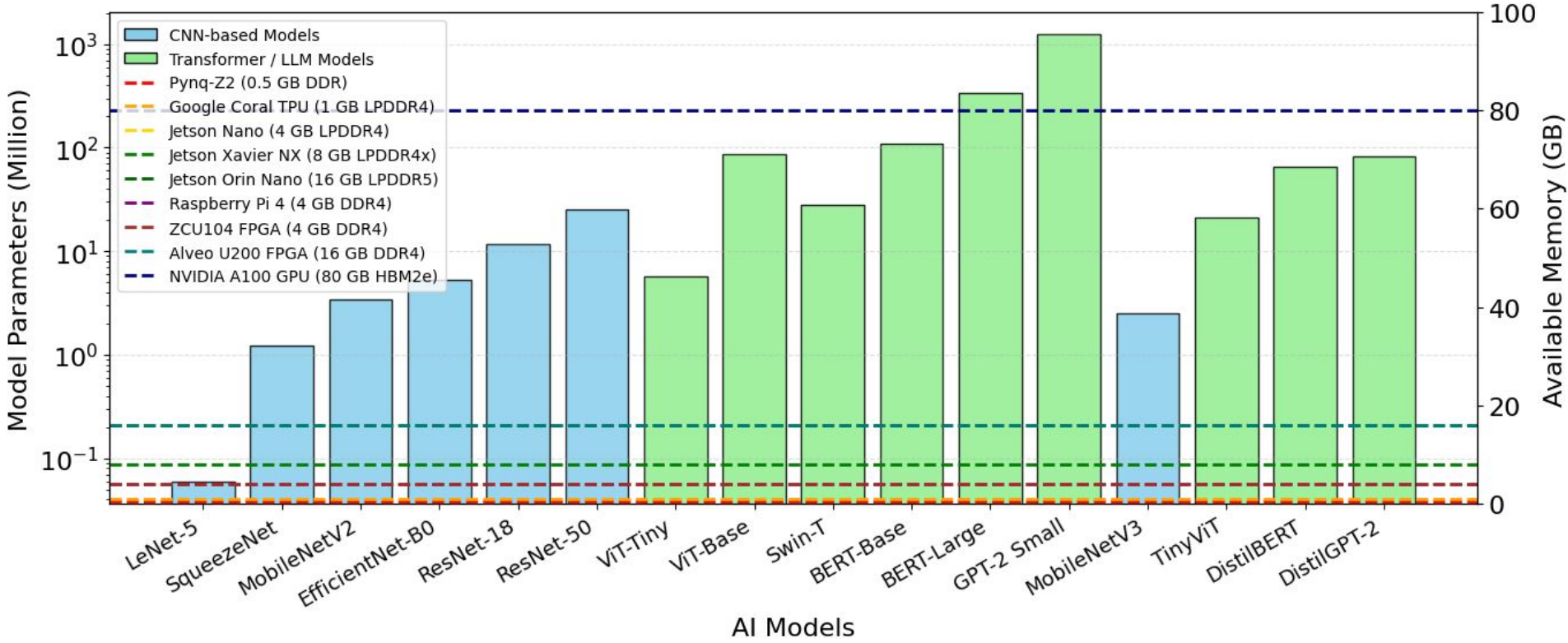
Adapted from Scale-sim Tutorial slides (ASPLOS 2021)

LLM (Large Language Model) Acceleration



LLMs (TFLOPs) vs. Edge Devices (TOPS) Over Time. (QSD: Qualcomm Snapdragon.)

Edge AI Devices Comparison



Comparative Evaluation of Different Devices

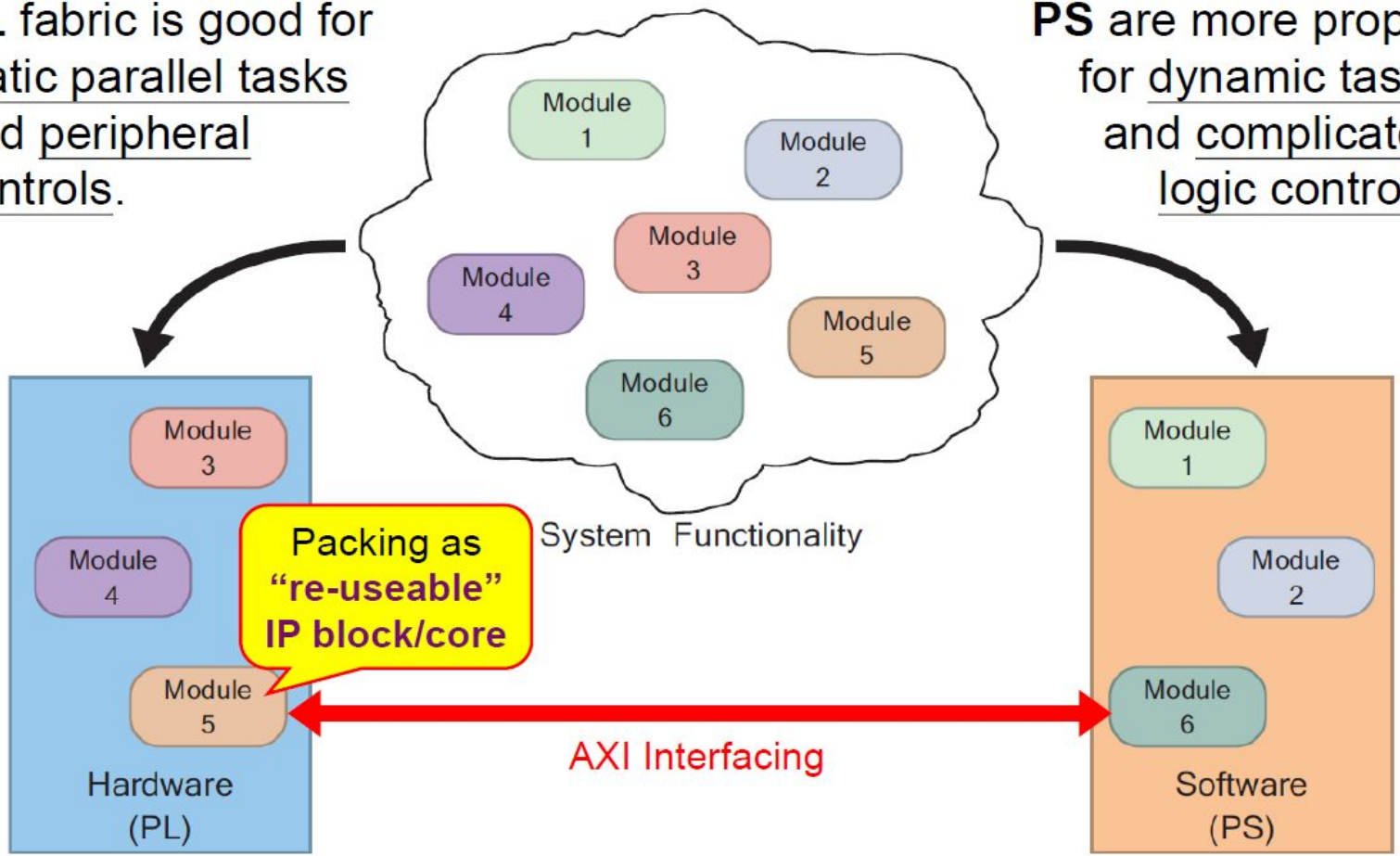
	CPU-based	GPU-based	FPGA-based	ASIC-based
Advantages	Good versatility Lowest price Multitasking High programmability	Medium versatility Massive parallelism Moderate programmability	Customized designs Low latency High performance/watt	Extremely low power Highest performance
Limitations	Limited parallelism	Power hungry	Limited on-chip memory Requires design expertise	High development cost Long Time-To-Market Low flexibility
Example Devices	Arm Cortex-M Series Raspberry Pi Series NanoPi Series Sipeed MAIX Series	Nvidia Jetson Series AMD Ryzen Family Arm Mali GPUs	Xilinx Zynq FPGAs Intel Arria 10 FPGAs Lattice iCE40 FPGAs	Google Edge TPU Ascend 310 processor In-memory chips Neuromorphic chips
Development Tools	Arm NN TensorFlow Lite	TensorRT Intel OpenVino	Intel OpenVino Xilinx Edge AI platform	Apache TVM

Adapted from Hao et al., “Enabling Design Methodologies and Future Trends for Edge AI: Specialization and Co-design”, IEEE Design & Test, 2021

Hardware/Software Partitioning

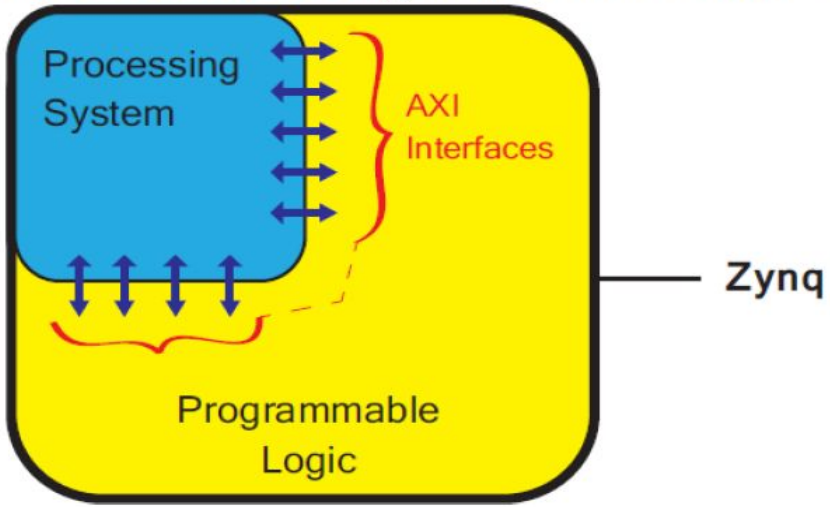
PL fabric is good for static parallel tasks and peripheral controls.

PS are more proper for dynamic tasks and complicated logic controls.



Zynq: A Programmable SoC (from Xilinx) Features

- **Processing System (PS)**: Dual-core ARM Cortex-A9 CPU
- **Programmable Logic (PL)**: Equivalent traditional FPGA
- **Advanced eXtensible Interface (AXI)**: High bandwidth, low latency connections between PS and PL.
  - *PS and PL can each be used for what they do best*, ***without*** the overhead of interfacing between PS and PL.

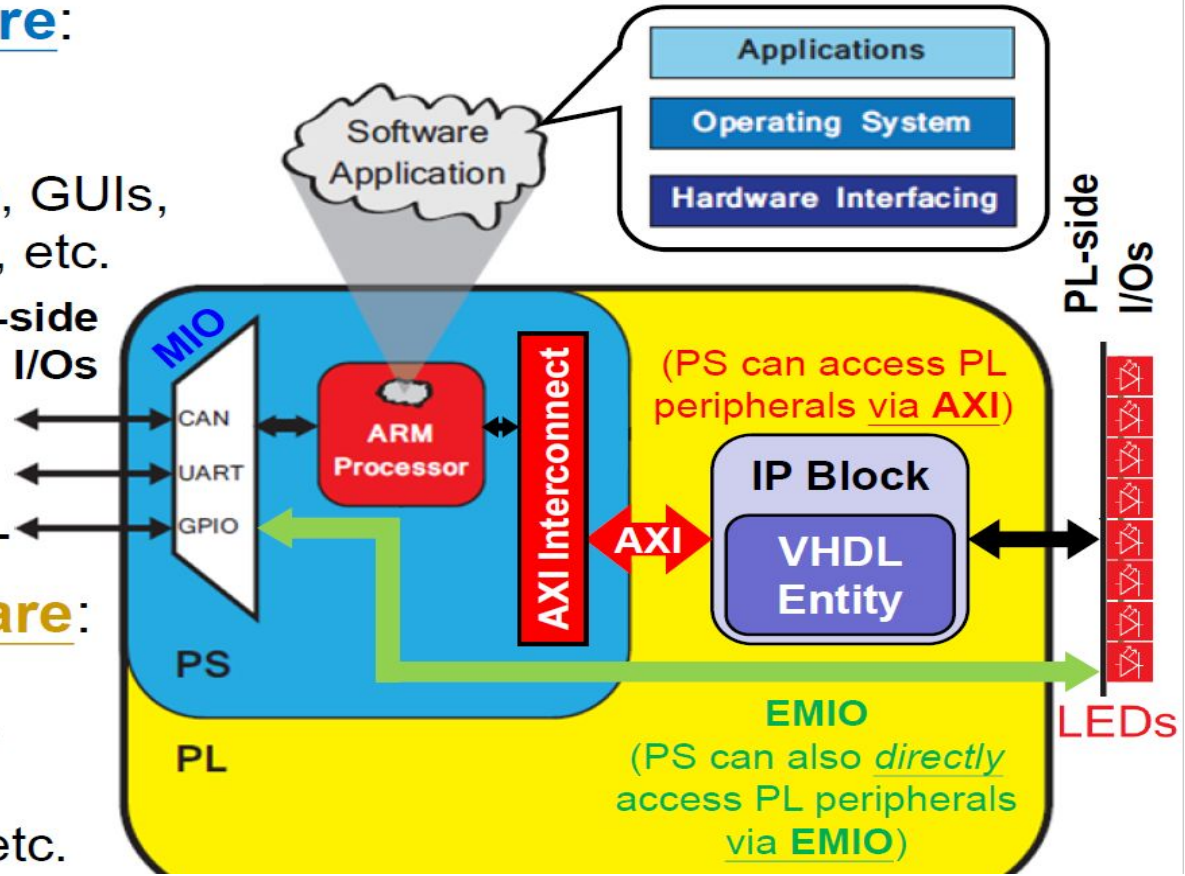


Rapid Prototyping with Zynq : PS + PL

PS for Software:
general purpose sequential tasks, operating system, GUIs, user applications, etc.

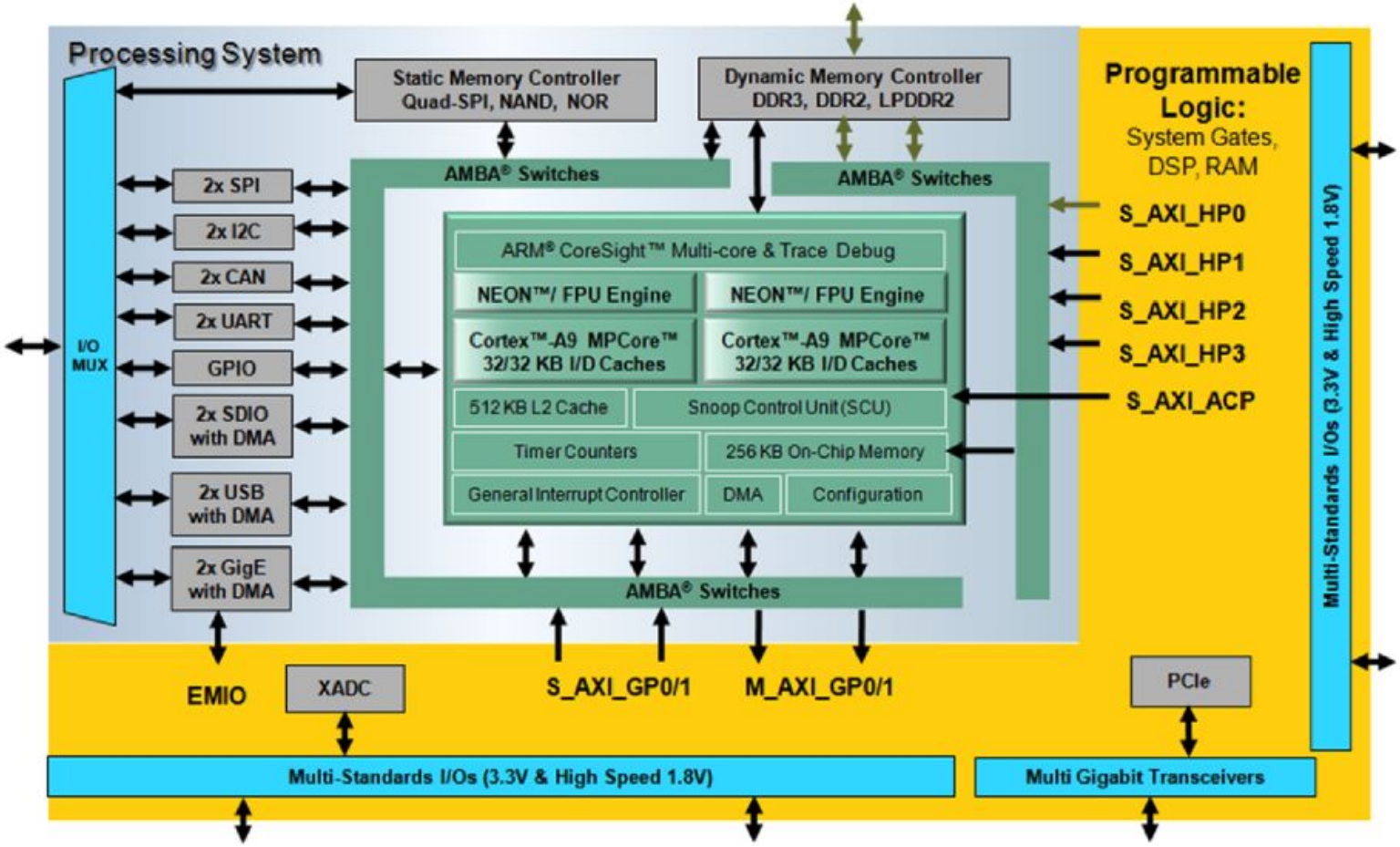
AXI:
hardware interfacing between PS & PL

PL for Hardware:
intensive data computation, PL-side peripheral communication, etc.



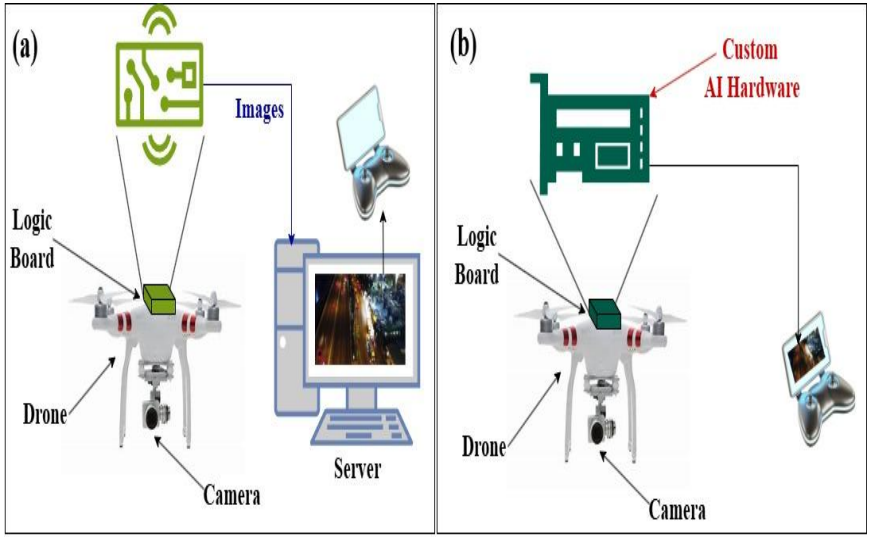
VHDL or Verilog -> any HDL can be used.

Zynq SoC Block Diagram



Lightweight Surveillance Image Classification through Hardware-Software Co-design

- A lightweight and well-generalized attention CNN model is developed and evaluated on a VisDrone-19 [1] and custom-made dataset.
- A buffer-enabled IP of attention-CNN is designed using high-level synthesis (HLS) that significantly reduces the data movement between off-chip (PS) to on-chip (PL) memory.
- Optimization methodology of IP core (PL) using multiple compiler directives.

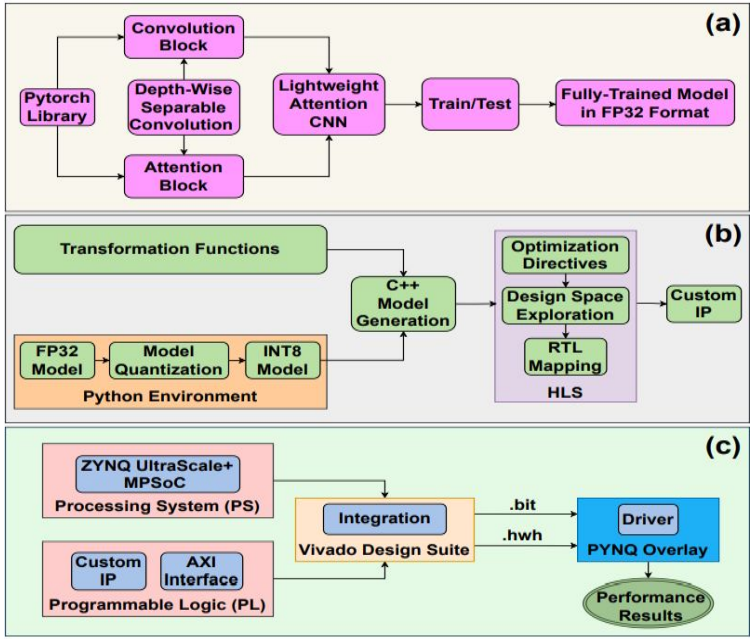


Cloud-based v/s Edge-based Surveillance

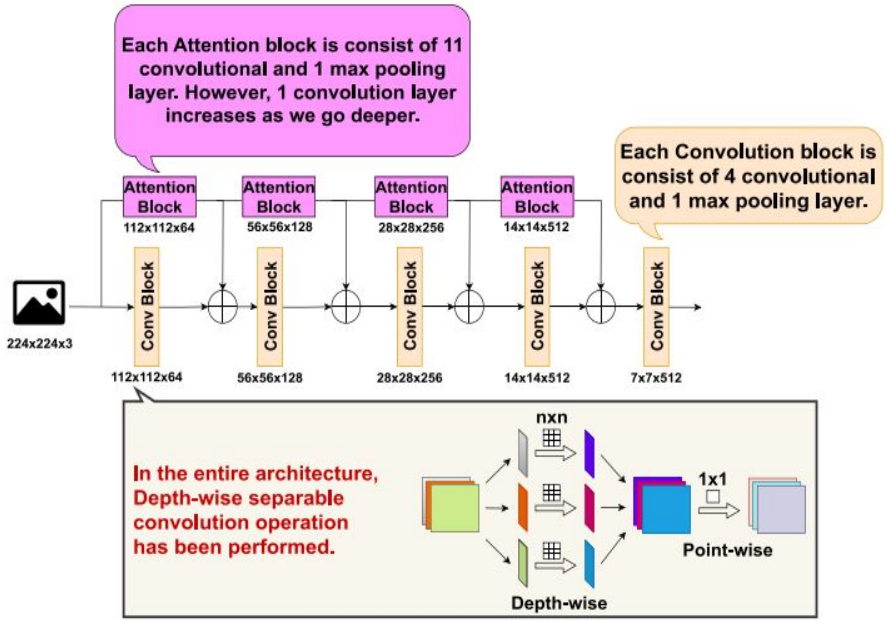
[1] "Visdrone-19 dataset," https://github.com/VisDrone/VisDrone-Dataset, accessed: 2023-06-06.

A. Yadav et al., IEEE Embedded Systems Letters. Vol. 17, No. 4, pp.222-225, August 2025.

Lightweight Surveillance Image Classification through Hardware-Software Co-design



(a) Model building (b) Model Transformation (c) H/S co-design.



Lightweight Attention-CNN Model

A. Yadav et al., IEEE Embedded Systems Letters. Vol. 17, No. 4, pp.222-225, August 2025.

Lightweight Surveillance Image Classification through Hardware-Software Co-design

- **Computation for normal convolution**:

  H*W*n*k^2*m

- **Computation for depth-wise:**

  H*W*n*k^2

- **Computation for point-wise:**

  H*W*n*m

Here: **H** is height of the input feature map,
W is width of the input feature map
n is Number of input channels (feature maps)
k is Kernel size (e.g., 3 for a 3×3 kernel)
m is Number of output channels (filters)

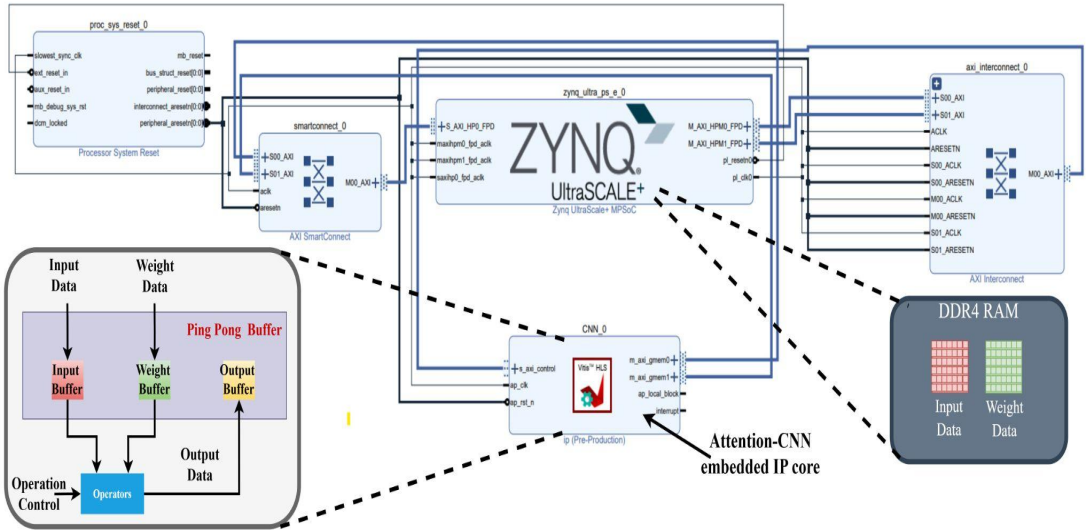


Fig. 12: Co-designed Hardware Accelerator targeted on ZCU104 & Zynq UltraScale+ MPSoC (PS)

A. Yadav et al., IEEE Embedded Systems Letters. Vol. 17, No. 4, pp.222-225, August 2025.

A) Lightweight Surveillance Image Classification through Hardware-Software Co-design

Results:

S. No.	Cases	Model	1	2	3	4	5	Avg
1	Case1	M1	81.47	97.30	100.00	87.14	92.28	91.64
		M2	80.96	94.68	98.42	90.55	92.91	91.10
		M3	79.15	90.18	96.24	79.07	89.27	86.78
2	Case2	M1	85.71	97.68	98.46	82.12	93.05	91.40
		M2	83.93	96.15	95.05	80.26	92.98	89.67
		M3	82.15	92.32	94.08	78.80	90.02	87.07
3	Case3	M1	82.28	97.15	99.33	83.16	91.57	90.90
		M2	79.61	97.38	99.19	82.65	91.50	90.06
		M3	75.20	91.08	96.34	79.62	82.94	85.83
4	Case4	M1	84.27	97.06	98.00	79.28	92.25	90.17
		M2	84.15	92.41	90.65	79.33	89.23	87.03
		M3	83.92	89.03	89.44	77.58	87.68	85.33

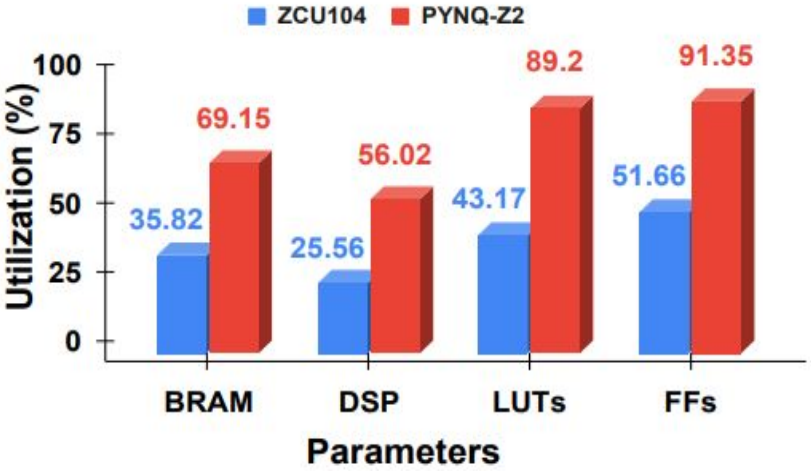
Validation Accuracy Comparison of Different Models For 5 Classes

- **Dataset 1 (DS1)**: 8.6k images containing pedestrians, cars, bicycles, tricycles, etc.
- **Dataset 2 (DS2)**: In-house dataset (1,301 images) created from drone-shot videos through:
  - Frame extraction using CV2 & Tkinter (Python)
  - Object detection with VGG16 model
  - Manual labeling with colored bounding boxes for class annotation
- **M1: Attention-CNN model** having 6.81 MB model size and has 0.88 million parameters.
- **M2: VGG16 model** having 528 MB model size and has 138 million parameters.
- **M3: MobileNetV2** having 14 MB model size and has 2.6 million parameters.
- **Epoch**=20. **In case 1**, training & validation are done with DS1 only. **In case 2**, training & validation are done with DS2 only. **In case 3**, training & validation are done with DS1 & DS2, respectively. **In case 4**, training & validation are done with DS2 & DS1,respectively.
- **1:** Pedestrian, **2:** People, **3:** Bicycle, **4:** Car, **5:** Van.

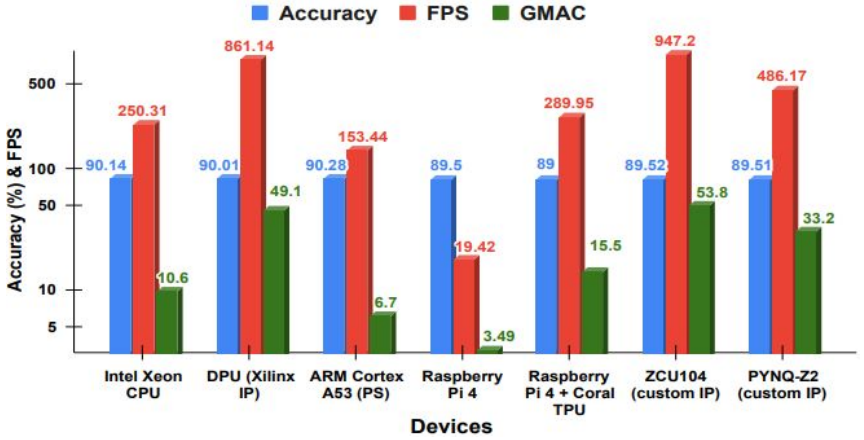
A. Yadav et al., IEEE Embedded Systems Letters. Vol. 17, No. 4, pp.222-225, August 2025.

A) Lightweight Surveillance Image Classification through Hardware-Software Co-design

Results:



Hardware utilization result on different embedded FPGA boards.



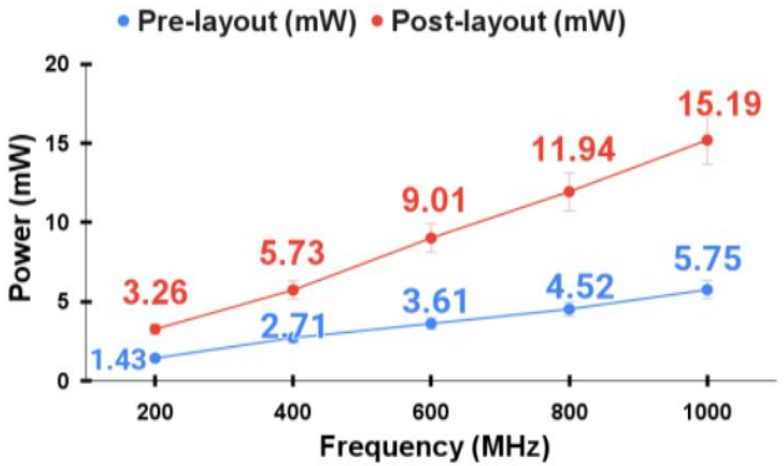
Comparison of performance consideration across different platforms.

- FPS: Frames-per-second, GMAC: Giga-Multiply-Accumulate operations.
- Model Size: 6.81MB, 0.88M Parameters, Epochs=20.

A. Yadav et al. in IEEE Embedded Systems Letters. Vol. 17, No. 4, pp.222-225, August 2025.

Lightweight Surveillance Image Classification through Hardware-Software Co-design

Results:



Pre & Post Layout Power of Co-designed Accelerator at 90nm GPDK

S.No.	Hardware Platform	Power (Watt)	Energy (J/img)	Speed (FPS)	Efficiency (FPS/W)	Buffer (MB)
1	Pynq-Z2 [8]	2.59	0.52	5.00	1.93	0.53
2	Ult. 96-V2 [8]	4.69	0.31	15.00	3.20	0.53
3	Pynq-Z1 [8]	1.97	0.22	9.00	4.57	-
4	ZCU104	4.01	0.21	28.67	4.65	0.24

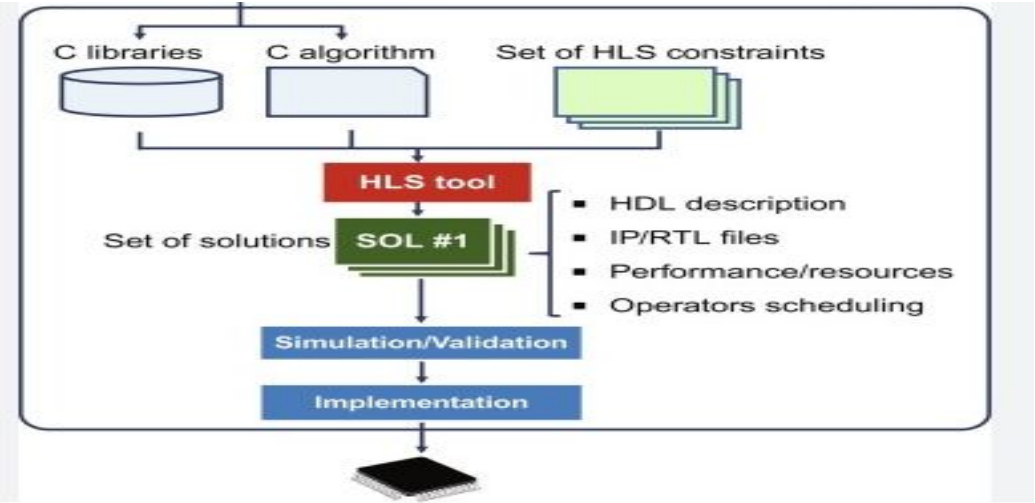
Performance Comparison with state-of-the-art

[8] Z. Zhang, M. A. P. Mahmud, and A. Z. Kouzani, “Fitnn: A low-resource fpga-based cnn accelerator for drones,” IEEE Internet of Things Journal, vol. 9, no. 21, pp. 21 357–21 369, 2022.

A. Yadav et al. in IEEE Embedded Systems Letters. Vol. 17, No. 4, pp.222-225, August 2025.

Design Synthesis

High-level synthesis (HLS) is a design process that automatically creates hardware implementations from high-level descriptions. It's also known as behavioral or algorithmic synthesis.



Without Pragma (Latency=545,LUT's=171)

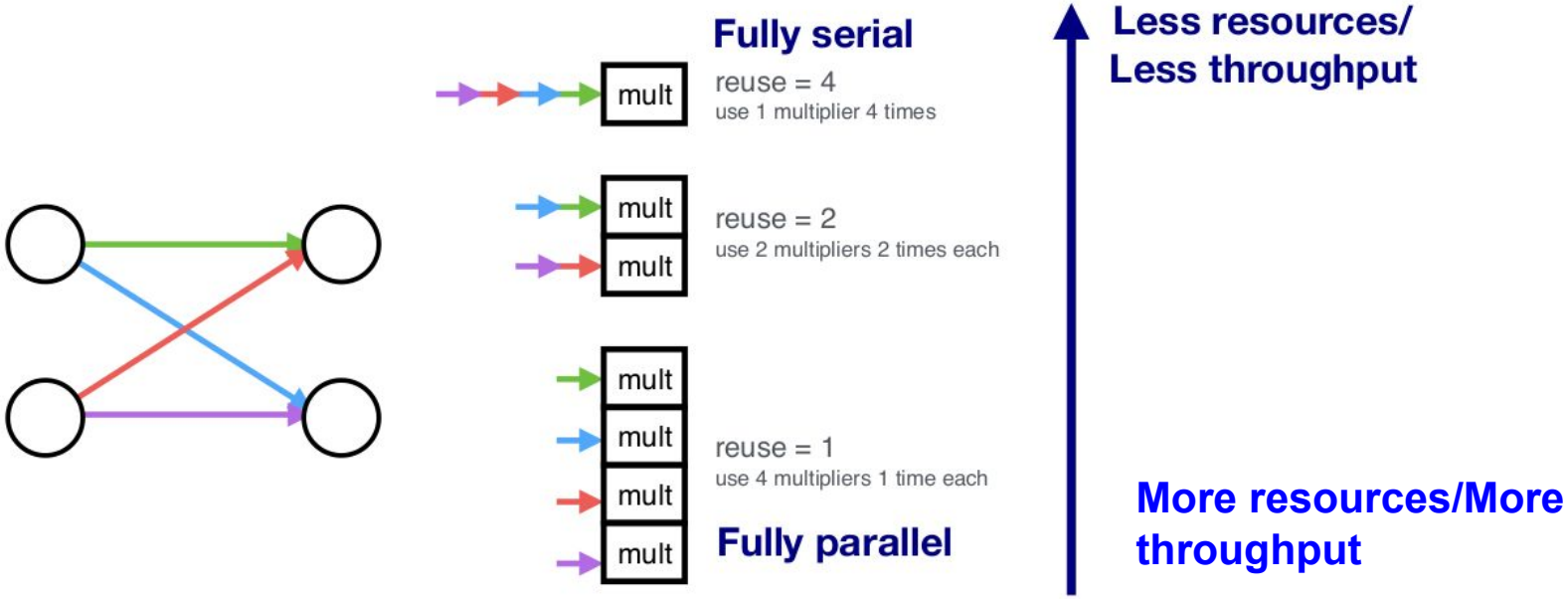
```
1# define N 16
2void pragma (int A[N][N],int x[N],int y[N]){
3for(int i=0;i<N;i++){
4        # pragma HLS pipeline off
5 int acc=0;
6 for (int j=0;j<N;j++){
7 acc += A[i][j]*x[j];
8 }
9 y[i]=acc;
10 }
11}
```

Pipeline outer loop (Latency =139,LUT's=1352)

```
/# define N 16
void pragma (int A[N][N],int x[N],int y[N]){
 for(int i=0;i<N;i++){
        # pragma HLS pipeline II=1
 int acc=0;
 for (int j=0;j<N;j++){
            # pragma HLS unroll
 acc += A[i][j]*x[j];
 }
 y[i]=acc;
 }
}
// Inner loop automatically unrolled when pipeline the outer loop.
```

Efficient NN design: parallelization

- Trade-off between latency and FPGA resource usage determined by the parallelization of the calculations in each layer
- Configure the **"reuse factor"** = number of times a multiplier is used to do a computation



Reuse factor: how much to parallelize operations in a hidden layer

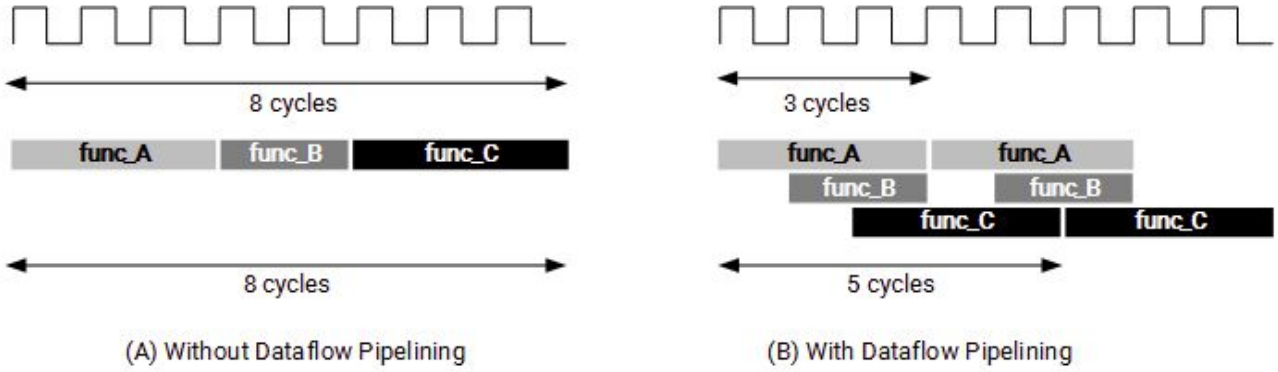
Design Optimization Techniques: `pragma HLS pipeline`

To make a design efficient, the DATAFLOW pragma enables task-level pipelining, allowing functions and loops to overlap in their operation, increasing the concurrency of the register transfer level (RTL) implementation, and increasing the overall throughput of the design.

```
void top (a,b,c,d) {
  ...
  func_A(a,b,i1);
  func_B(c,i1,i2);
  func_C(i2,d)

  return d;
}
```

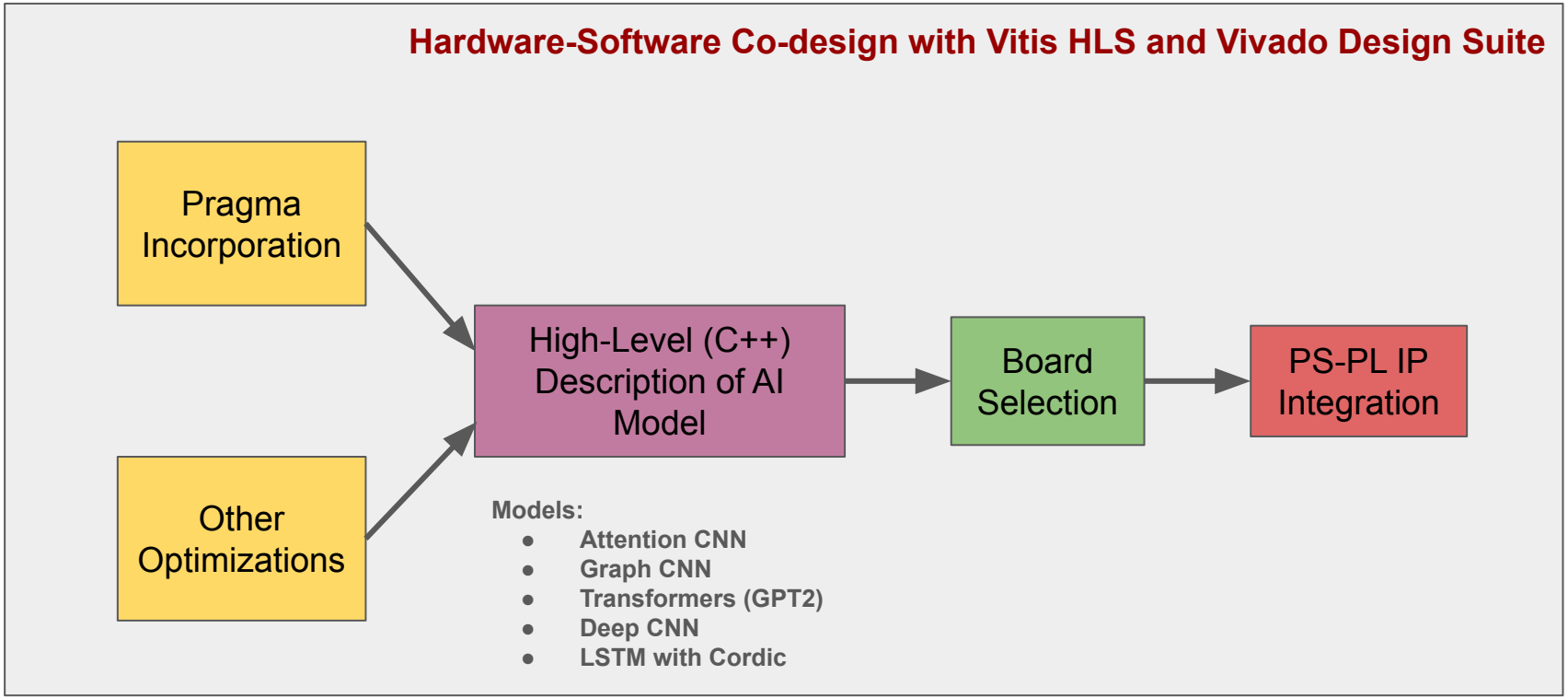
func_A
func_B
func_C



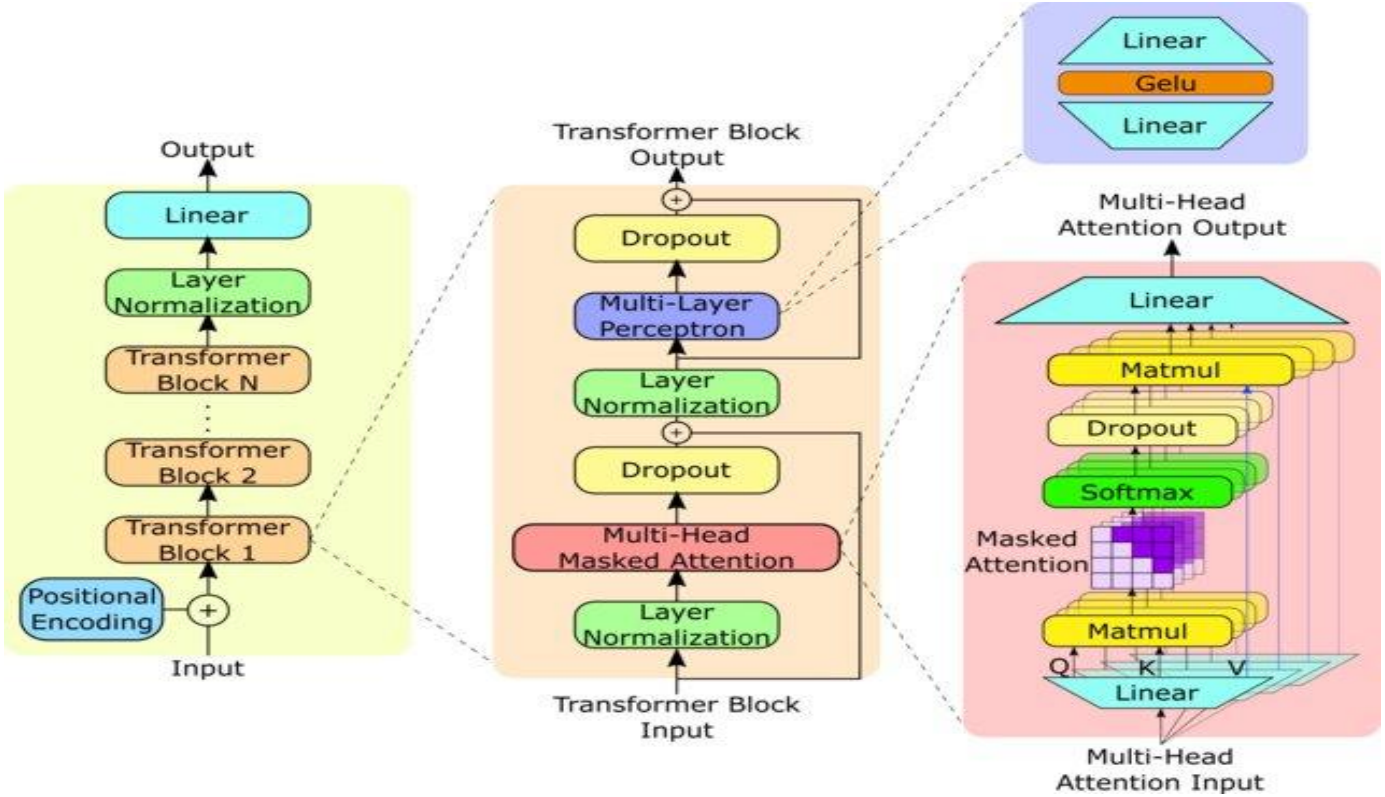
(A) Without Dataflow Pipelining

(B) With Dataflow Pipelining

Co-Design Methodology Flow

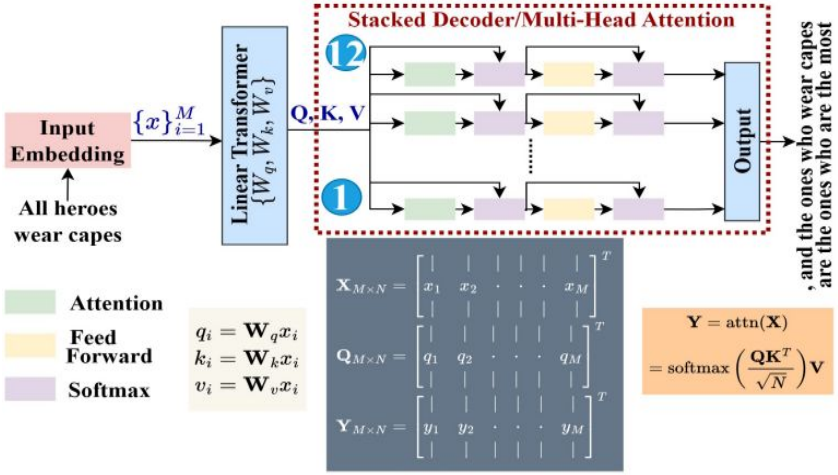


GPT-2 Architecture



Attention-based GPT-2 Accelerator for Resource-Constrained Platform

- Development of a baseline accelerator and evaluation of its performance against software implementation by analyzing throughput, power efficiency, and latency.
- Optimization of the baseline accelerator through incorporating layer-fusion, buffer-tiling, and compiler directives.
- Accessing the generalizability of the proposed optimization through implementing various configurations of the model.



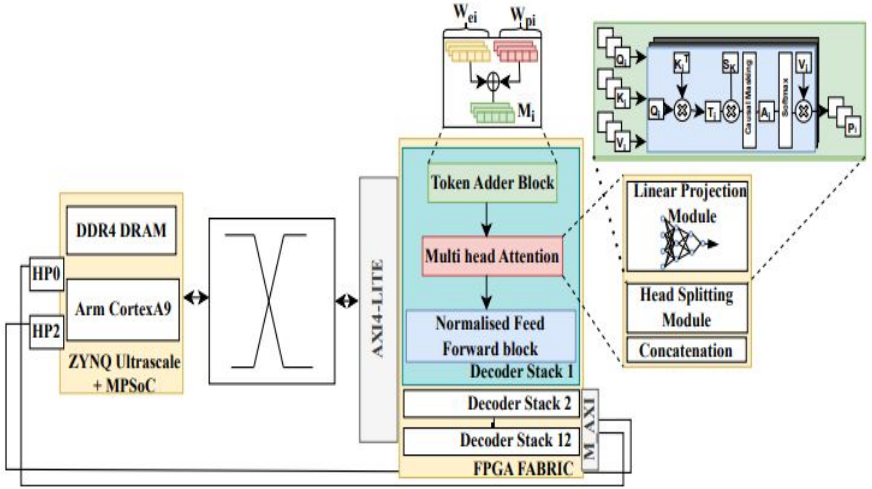
Standard GPT-2 Architecture

Accepted in IEEE Embedded Systems Letters. Early access, 4 pages,June 2025.

Attention-based GPT-2 Accelerator for Resource-Constrained Platform

FUNCTION USAGE	INPUT 1 MATRIX	INPUT 2 MATRIX	OUTPUT MATRIX
Token + Positional Embed. Addition	W_{ei5×768} (Word Emb.)	W_{pi5×768} (Positional Emb.)	X_{in5×768}
MHA: QKV Projection	X_{in5×768}	W_{QKV768×2304} (Fused QKV W)	A_{QKV5×2304} (Fused QKV Act)
MHA: Attention Calculation	Q_{5×768} (from A_{QKV})	K, V_{5×768} (from A_{QKV})	Attn_{concat5×768}
MHA: Output Projection	Attn_{concat5×768}	W_{dense768×768} (Dense W)	MHA_{out5×768}
FFN: Layer 1 (Expand)	X_{FFN_in5×768}	W_{FFN1 768×3072} (FFN W1)	A_{FFN1 5×3072}
FFN: Activation (GELU)	A_{FFN1 5×3072}	~	GELU_{out5×3072}
FFN: Layer 2 (Contract)	GELU_{out5×3072}	W_{FFN2 3072×768} (FFN W2)	FFN_{out5×768}

Sequence of Operations & Tensor Dimension for GPT-2 Model



Baseline Architecture for GPT-2 Model Comprises PS-PL Process

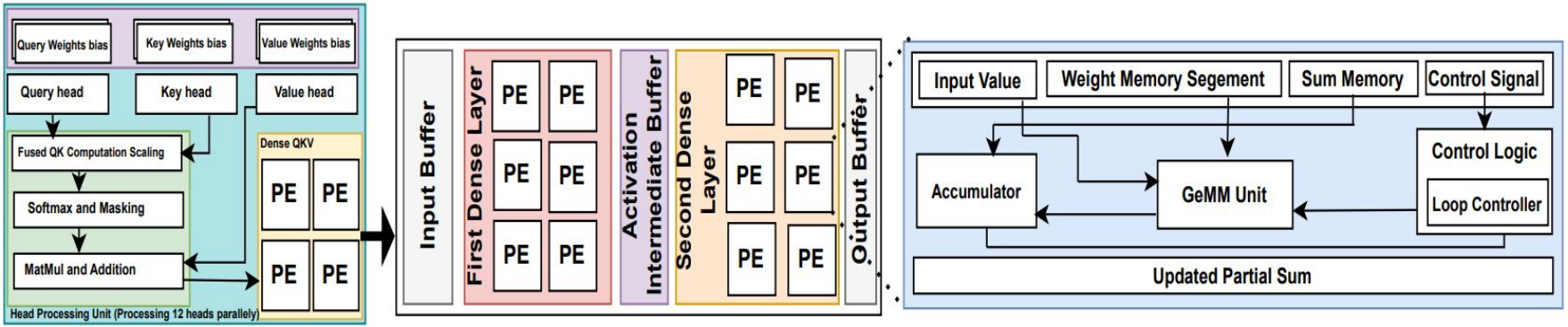
*MHA: Multi-head attention, FFN: Feed-forward network, *pi:* Positional embedding, *ei:* Input embedding, K:Key, V: Value, Q:Query, PS: Processing System, PL: Programmable Logic

Accepted in IEEE Embedded Systems Letters. Early access, 4 pages,June 2025.

Attention-based GPT-2 Accelerator for Resource-Constrained Platform

Decision Making:

- Layer Fusion (Minimize memory access)
- Buffer Tiling (Optimize on-chip usage)
- Unrolling (Maximize parallelism)
- Pipelining (Increase throughput)

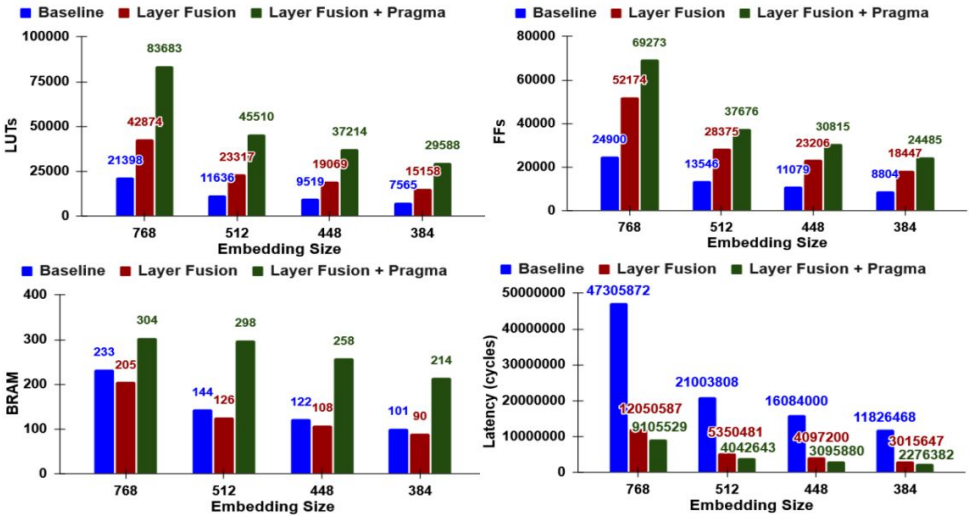


a) **Optimized Architecture for MHA Block processing 12 heads parallelly b) Optimized Architecture for FFN Block**

Accepted in IEEE Embedded Systems Letters. Early access, 4 pages,June 2025.

Attention-based GPT-2 Accelerator for Resource-Constrained Platform

Results:



Effect of layer fusion with varying embedding size for MHA Block

n_{dim}	M	Hardware Utilization				M	Hardware Utilization			
		LUTs	FFs	BRAM	DSP		LUTs	FFs	BRAM	DSP
768	5	83,683	107,025	205	173	8	125,007	159,720	290	258
512	5	37,155	47,519	91	77	8	57,448	76,030	145	123
448	5	28,452	36,389	70	59	8	45,622	58,501	112	94
384	5	20,921	26,756	51	43	8	33,473	42,610	81	69
768	6	100,420	128,430	246	208	9	127,205	162,409	296	270
512	6	44,586	57,023	109	92	9	63,879	85,534	164	138
448	6	34,142	43,667	84	71	9	51,412	66,112	126	106
384	6	25,105	32,107	61	52	9	37,657	47,761	91	77
768	7	117,156	149,835	287	242	10	127,192	162,394	298	259
512	7	51,017	66,527	127	108	10	70,310	95,038	182	154
448	7	39,832	50,945	98	83	10	57,203	73,723	140	118
384	7	29,289	37,458	71	60	10	41,841	52,913	101	86

Scalable Resource Utilization Of Optimized GPT-2 On ZCU104

*M: Sequence length, n_{dim}: Embedding size, Available resources on ZCU104: LUTs: 460800, FFs: 230400, BRAM: 312, DSP:1728

Accepted in IEEE Embedded Systems Letters. Early access, 4 pages,June 2025.

Attention-based GPT-2 Accelerator for Resource-Constrained Platform

Results:

Real-time evaluation of latency on different platforms

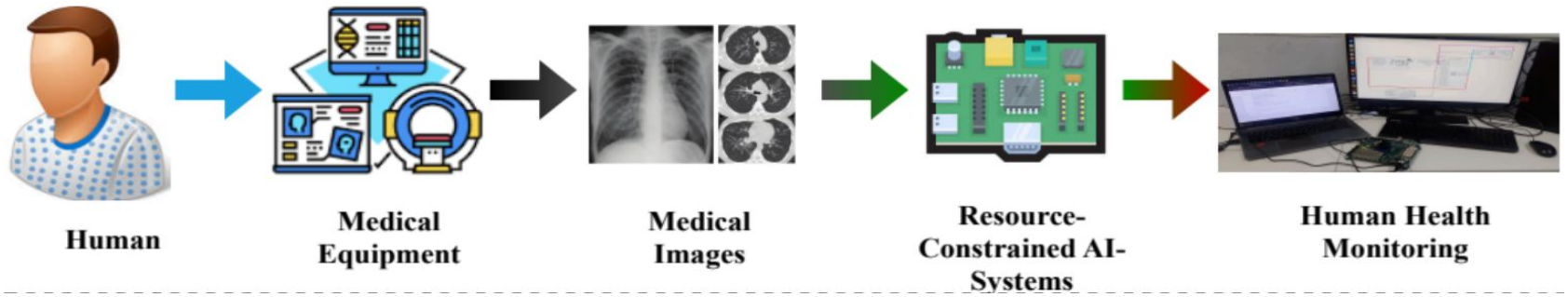
Ref.	Model	Hardware Platform	Power (W)	Through-put (SPS)	Efficiency (SPS/W)	Co-Design (?)
[7]	BERT	U280 @245MHz	×	38.45	×	✓
[8]	FQ-BERT	ZCU111 @214MHz	13.20	10.51	3.18	✓
[9]	TRAC	ZCU106 @200MHz	×	2.88	×	×
This Work	GPT-2 (Baseline Arch.)	ZCU104 @150MHz	3.21	67.11	20.90	✓
	GPT-2 (Optimized Arch.)	ZCU104 @150MHz	4.33	84.03	19.40	✓

Performance Comparison with State-of-the-art Works

*BERT: Bidirectional Encoder Representations from Transformers, SPS:Samples-per-second

[7] H. Chen, J. Zhang, Y. Du, S. Xiang, Z. Yue, N. Zhang, Y. Cai, and Z. Zhang, “Understanding the potential of fpga-based spatial acceleration for large language model inference,” ACM Transactions on Reconfigurable Technology and Systems, vol. 18, no. 1, p. 1–29, Dec. 2024. [Online]. Available: http://dx.doi.org/10.1145/3656177
[8] Z. Liu, G. Li, and J. Cheng, “Hardware acceleration of fully quantized bert for efficient natural language processing,” 2021. [Online]. Available: https://arxiv.org/abs/2103.02800
[9] P. Plagwitz, F. Hannig, and J. Teich, “Trac: Compilation-based design of transformer accelerators for fpgas,” in 2022 32nd International Conference on Field-Programmable Logic and Applications (FPL), 2022, pp. 17–23.

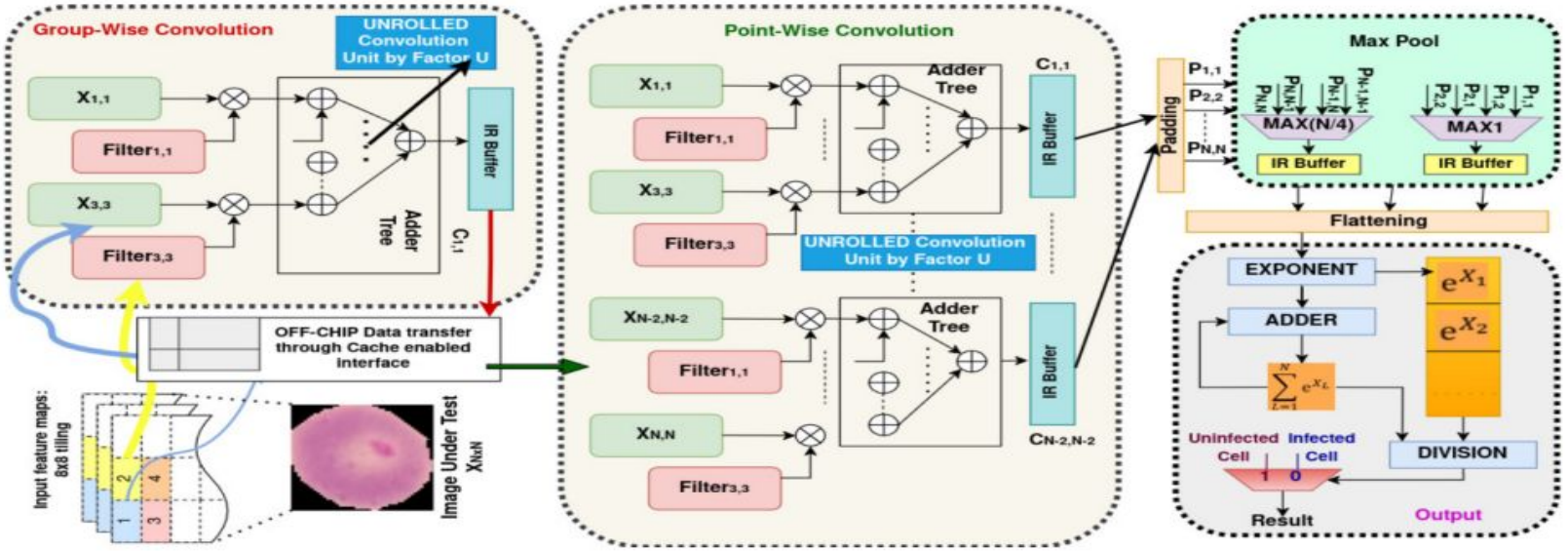
Medical Image Processing using Hardware-Software Co-design



- Integration of layer fusion, buffer tiling, and loop unrolling within the HLS pipeline of depth-wise convolutions to exploit fine-grained parallelism and improve execution efficiency on programmable logic.
- Design of a custom cache controller that enables burst-mode memory access, reuses depth-wise kernels, and reduces redundant data transfers, enhancing off-chip memory access efficiency.
- Deployment and real-time validation of the proposed CNN accelerators on the Zynq UltraScale+ MPSoC ZCU104 FPGA board, with comprehensive analysis of FPGA resource utilization and performance.

A. Yadav et al., IEEE Transactions on Biomedical Circuits and Systems. Early access, 12 pages, August 2025.

Medical Image Processing using Hardware-Software Co-design



Micro-architecture of the Proposed Depthwise-Separable Convolution (DSC) Accelerator

- Each input channel is convolved with its own K×K filter (depthwise), done in parallel by “unrolled” units.
- Stores depthwise outputs (partial feature maps) on-chip so the next stage can access them quickly.
- 1×1 convolutions combine the per-channel depthwise outputs to form new output channels (channel mixing).

Accepted in IEEE Transactions on Biomedical Circuits and Systems. Early access, 12 pages, August 2025.

Medical Image Processing using Hardware-Software Co-design

Algorithm 1 HLS Pipeline for Tiled Layer Fusion enabled DSC Acceleration

Input: Feature map $X \in \mathbb{R}^{H \times W \times C_{\text{in}}}$, depth-wise kernels F_d, point-wise kernels F_p
Parameters: Tile size $T \times T$, kernel size K_d, layers L, initiation interval $II = 1$
Output: Classification output y
for $l = 1$ to L **do**
    Partition $X^{(l)}$ into $T \times T$ spatial tiles
    Stream each tile and depth-wise kernels $F_d^{(l)}$ to on-chip buffers
    for each tile t **do**
        for each channel c **(parallel in PEs) do**
            Perform depth-wise conv with $F_{d,c}$ using unrolled, pipelined MACs ($II = 1$)
            Apply ReLU and store $IR_{t,c}$ to off-chip memory
        end for
    end for
    Fetch IR tiles and stream $F_p^{(l)}$ for point-wise conv
    for each tile t **do**
        for each output channel o **(parallel in PEs) do**
            Compute 1×1 conv across C_{in} using unrolled dot-product
            Store $Y_t^{(l)}$ to off-chip
        end for
    end for
    Set $X^{(l+1)} \leftarrow Y^{(l)}$
end for
Flatten final output and apply softmax to obtain y

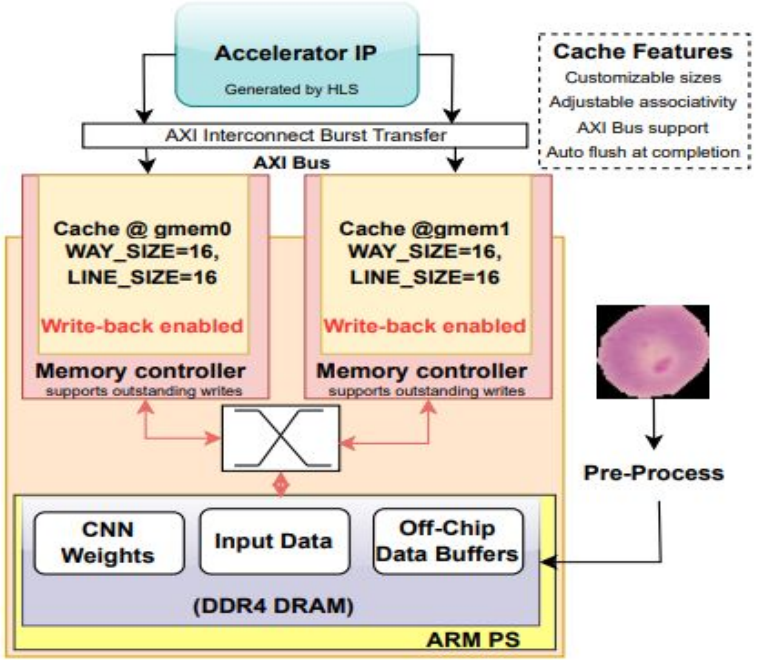


Fig.23: Custom Cache Integration

A. Yadav et al., IEEE Transactions on Biomedical Circuits and Systems. Early access, 12 pages, August 2025.

Medical Image Processing using Hardware-Software Co-design

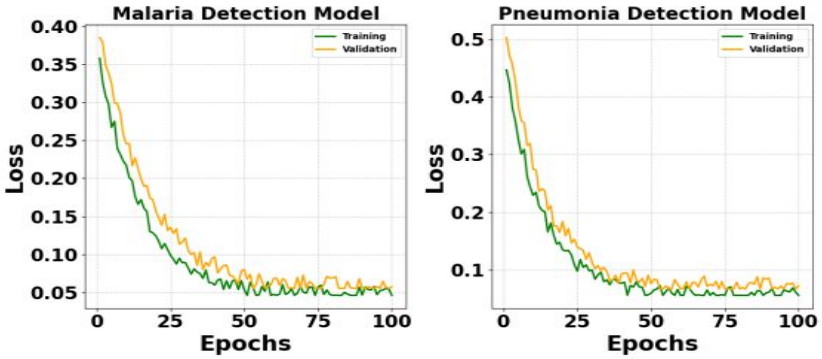
Results:

Application	n	H	W	D	k
Malaria Detection [23]	27,558	130	130	3	2
Pneumonia Detection [24]	5,863	150	150	1	2

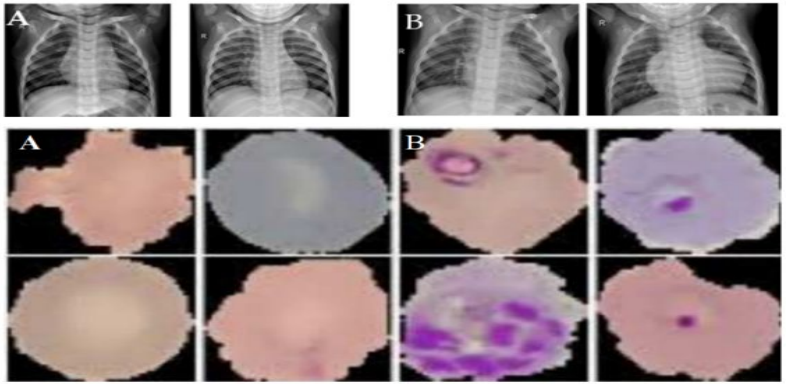
Table 6: Dataset Description (Here, n number of images with height H, width W, D channels, distributed among k different classes.)

Application	Conv	Filter	Dense	Params	N_{MAC}	Acc (%)
Malaria Detection	3	3x3	2	1,662,209	296	94.49
Pneumonia Detection	5	5x5	2	1,246,401	268	89.10

Model Performance For Different Applications



Comparison of Loss vs Epoch for Both Models



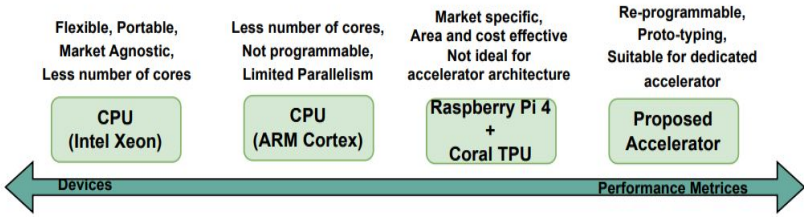
Simulation results: A) uninfected cell, B) infected cell, Pneumonia (top), Malaria (bottom)

[23] "Malaria dataset kaggle link," https://www.kaggle.com/datasets/ iarunava/cell-images-for-detecting-malaria, accessed: 2023-06-06.
[24] "Pneumonia dataset kaggle link," https://www.kaggle.com/datasets/ paultimothymooney/chest-xray-pneumonia, accessed: 2023-06-06.

A. Yadav et al., IEEE Transactions on Biomedical Circuits and Systems. Early access, 12 pages, August 2025.

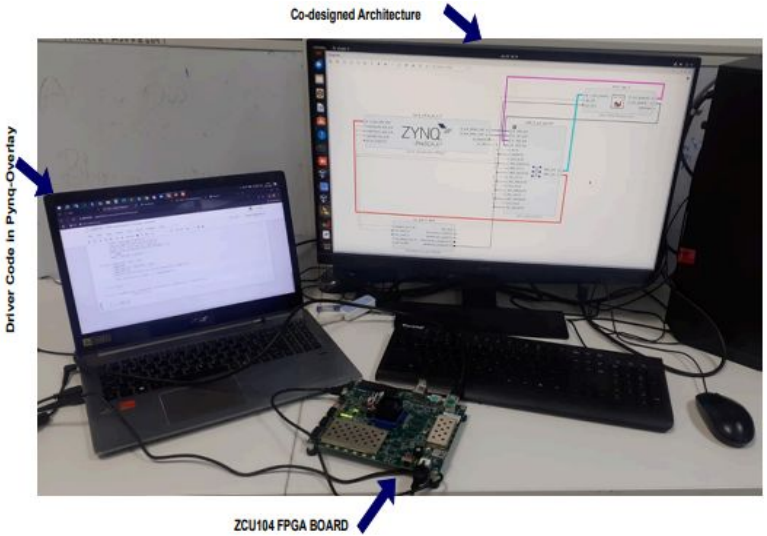
Medical Image Processing using Hardware-Software Co-design

Results:



Applications	Malaria Det.	Pneumonia Det.	Malaria Det.	Pneumonia Det.	Malaria Det.	Pneumonia Det.	Malaria Det.	Pneumonia Det.
Power (W)	10.89	9.11	2.26	2.01	2.68	2.34	4.36	4.92
GOPs	0.63	0.58	0.029	0.020	15.3	10.6	88.65	80.28
GOPs/W	0.057	0.063	0.012	0.009	5.7	4.5	20.33	16.32
Energy Efficiency	1.82 (J/Img)	2.07 (J/Img)	0.56 (J/Img)	0.51 (J/Img)	85.37 (mJ/Img)	80.21 (mJ/Img)	14.62 (mJ/Img)	23.89 (mJ/Img)
FPS	5.97	4.40	4.02	3.91	31.39	29.17	298.22	205.87

Performance Comparison of Different Metrics of Proposed Accelerators and Traditional Devices

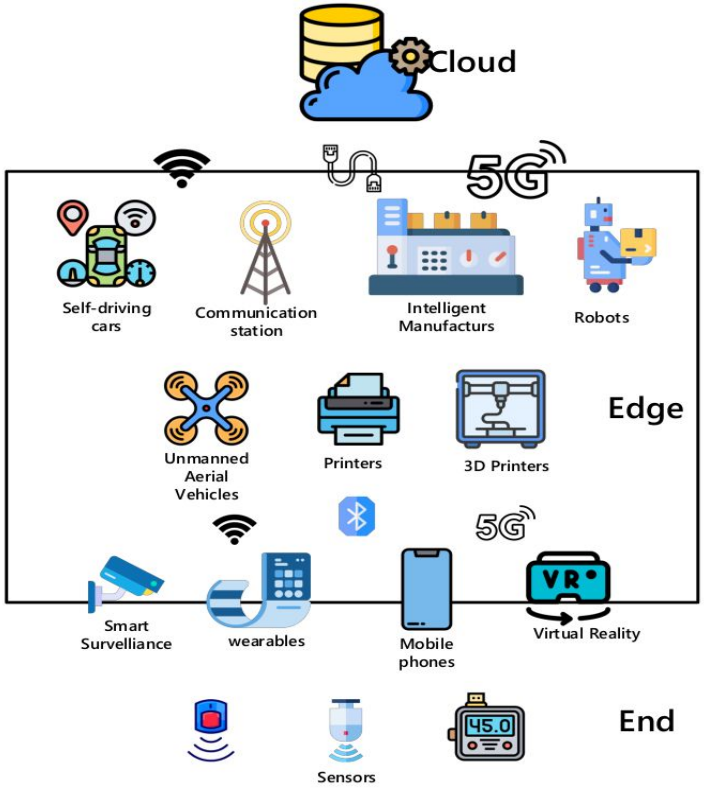


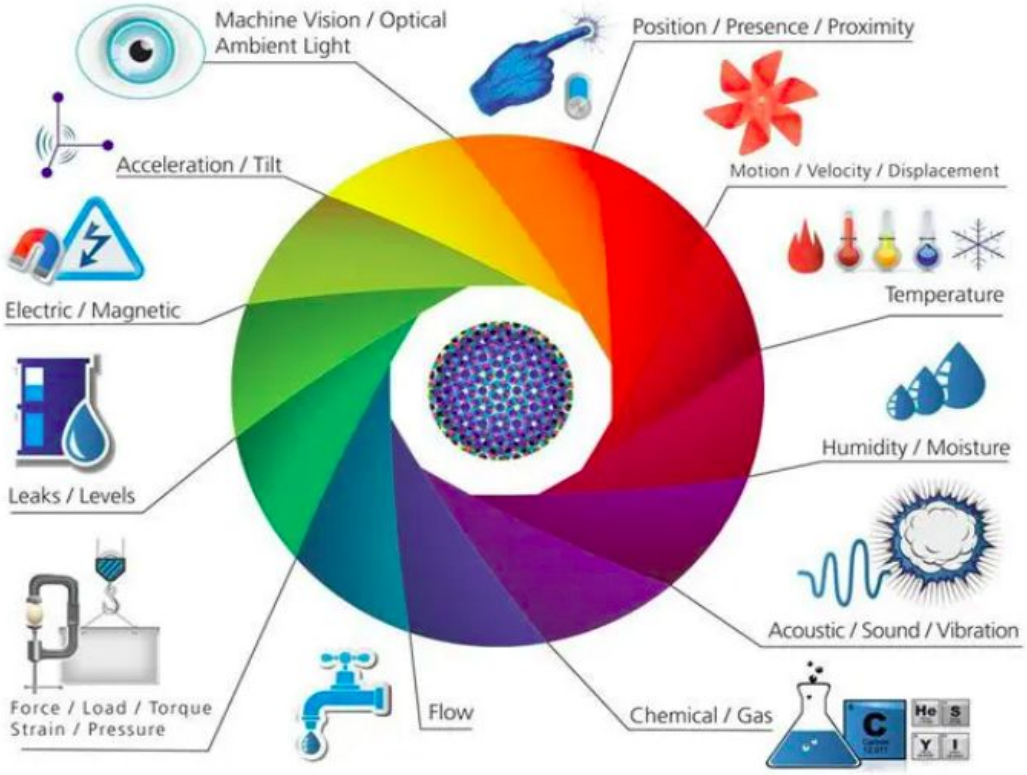
Real-time ZCU104 FPGA Setup of Co-designed Accelerator

* FPS:Frames-per-second, GOPs:Giga Operations Per Second

A. Yadav et al., IEEE Transactions on Biomedical Circuits and Systems. Early access, 12 pages, August 2025.

Conclusion: AI Moving Close to Edge

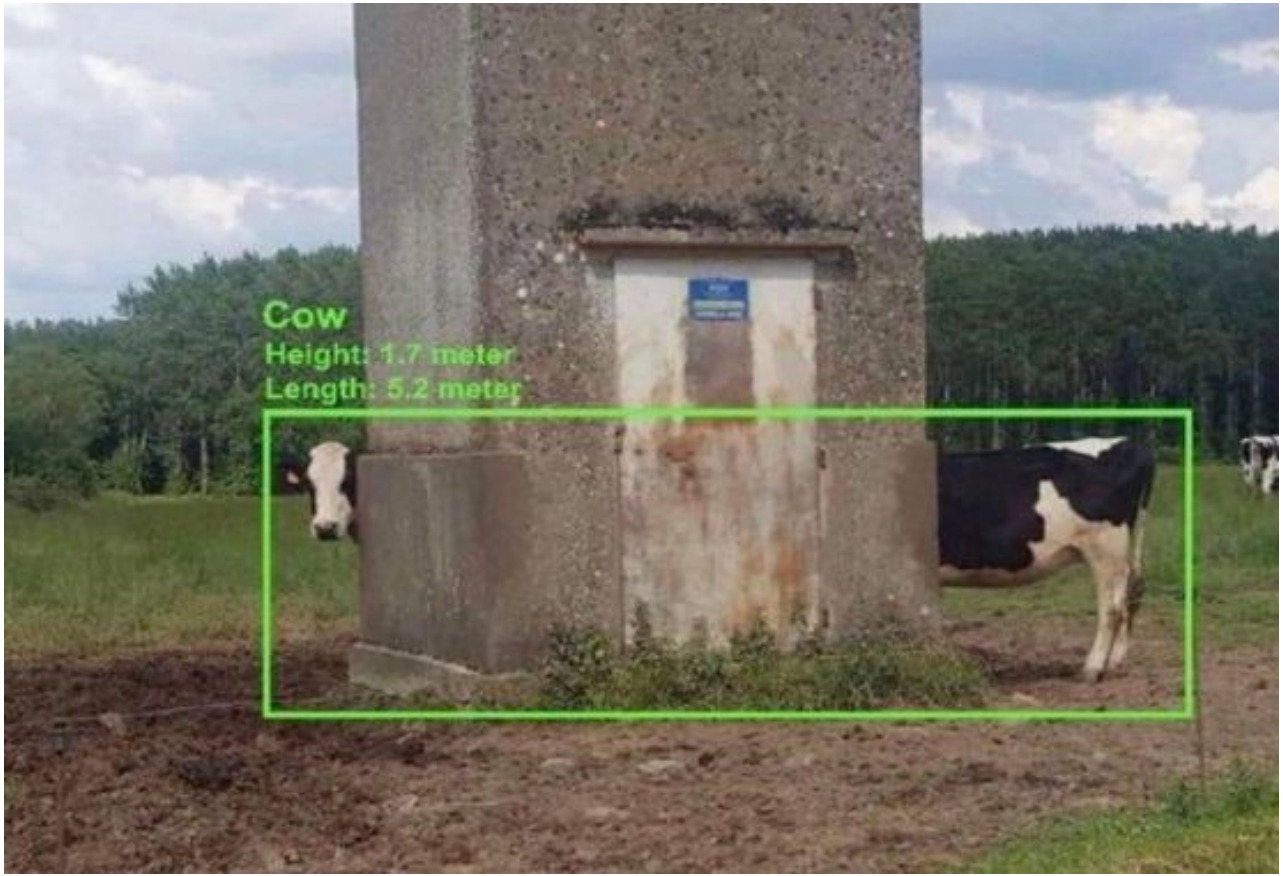




Sensors are everywhere!

Caveat!

Be Careful!



THANK YOU

Please write to binod@iitj.ac.in