LP-BNN: Ultra-low-Latency BNN Inference with Layer Parallelism

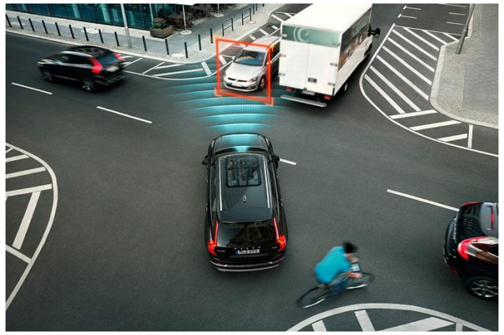
Tong Geng[1, 2], *Tianqi Wang[1]*, Chunshu Wu[1], Chen Yang[1], Shuaiwen Leon Song[2]
Ang Li[2], Martin Herbordt[1]

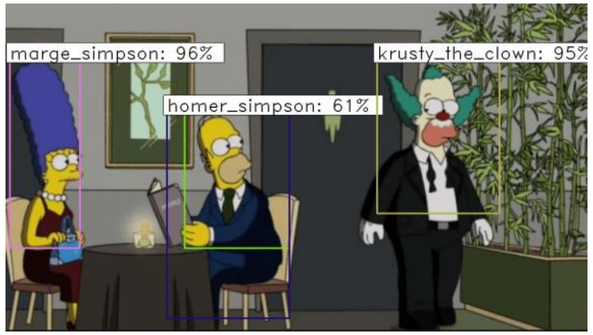
[1]Boston University

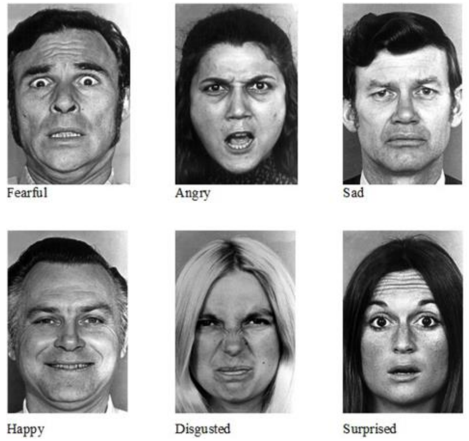
[2]Pacific Northwest National Laboratory

CNN: most popular algorithm in ML

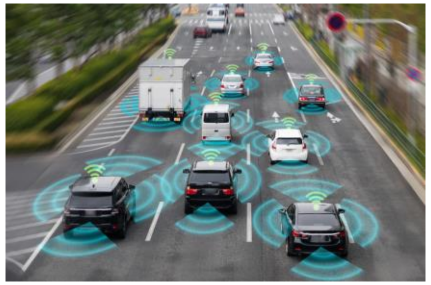
- Widely used in computer vision such as object classification, detection and recognition







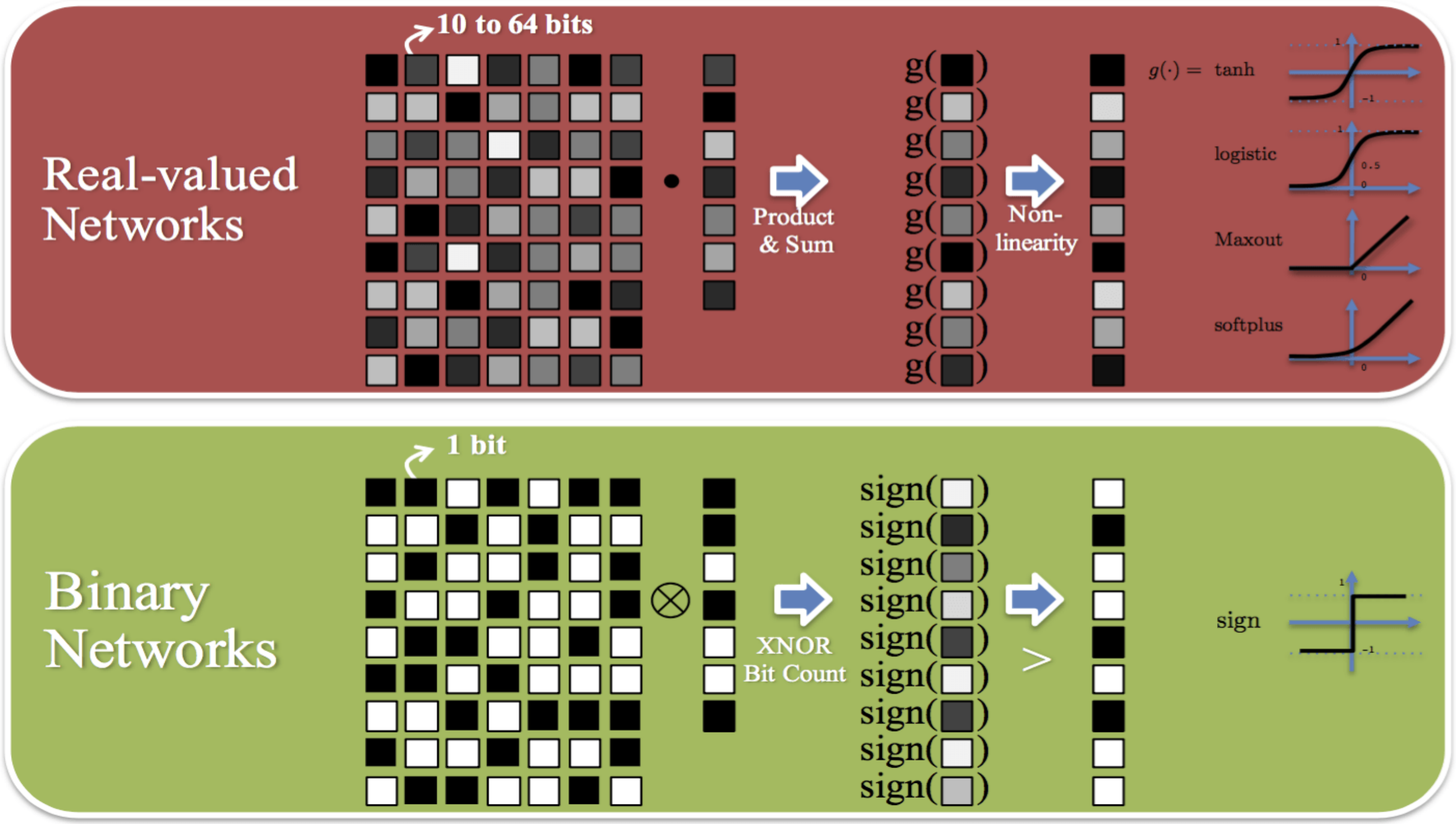
Birth of BNN



- Limitations of CNN
  - High computation + memory intensity
  - Long Latency
- 32-bit floating-point -> 8-bit fixed-point -> binary
  - Computation and Memory access are not intensive anymore
  - Potentially, much Lower Latency
  - Relatively lower accuracy than CNN, however, becoming better

DNN: BNN vs CNN

- ✓ **Easier Computation:** floating-point-multiply-add (FMA) operations ➔ single-bit XNOR/POPCOUNT.
- ✓ **Less Storage:** floating-point parameters and activations ➔ single-bit
- ✓ **Energy Efficient:** ideal for edge device



“Accelerating Neural Networks with Binary Arithmetic,” https://ai.intel.com/accelerating-neural-networks-binary-arithmetic.

Inference of BNN on different platforms

- **Utilization to accelerate the inference of AlexNet**
  - **batch size of 1:**

  × **GPU:** ~ 1%. **CPU:** ~ 6%.
  - **batch size of 10:**

  × **GPU:** ~ 7%. **CPU:** ~ 10%.

  ✓**FPGA:** >60%: Millions of one-bit ALUs on a single device.

Eriko Nurvitadhi, David Sheffield, Jaewoong Sim, Asit Mishra, Ganesh Venkatesh, and Debbie Marr. 2016. Accelerating binarized neural networks: comparison of FPGA, CPU, GPU, and ASIC. In Field-Programmable Technology (FPT), 2016 International Conference on. IEEE, 77–84.

Challenges

- **Challenges to make truly low-latency inference using FPGA**

1. The critical Normalization Layer (NL) uses full-precision floating point (i.e., 2 FP MUL/DIV + 3 FP ADD/SUB).
2. Existing works process layers sequentially. Hence, their latencies are accumulated with no overlapping.
3. Optimal designs for all layers need to be simultaneously configured on FPGA with no reconfiguration.

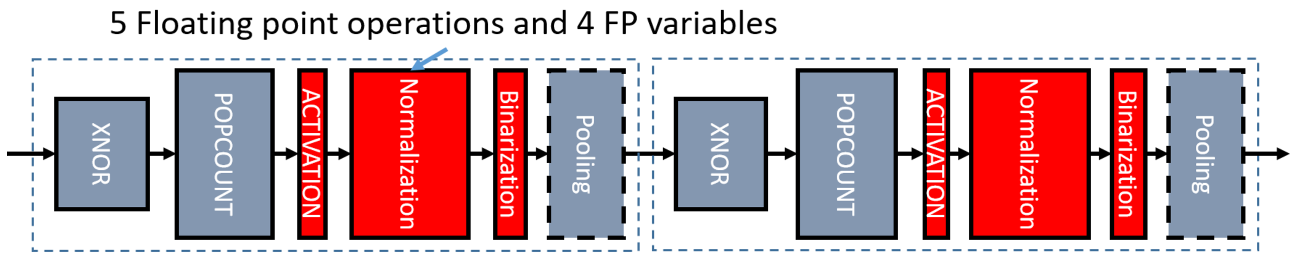
Challenges

- **Challenges to make truly low-latency inference using FPGA**

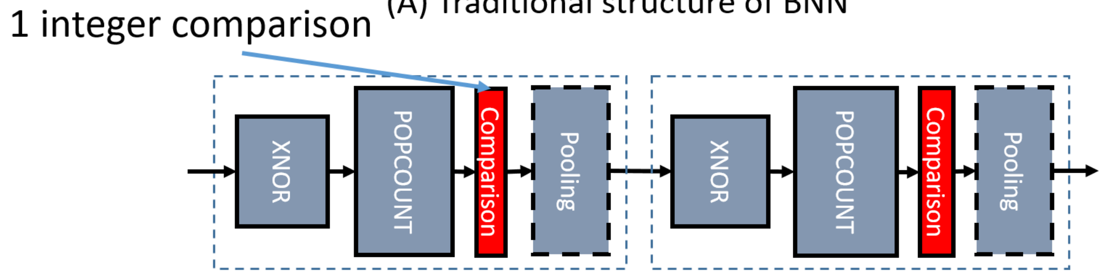
1. The critical Normalization Layer (NL) uses full-precision floating point (i.e., 2 FP MUL/DIV + 3 FP ADD/SUB).
2. Existing works process layers sequentially. Hence, their latencies are accumulated with no overlapping.
3. Optimal designs for all layers need to be simultaneously configured on FPGA with no reconfiguration.

Intra-layer Fusion

- **Addressing the 1st challenge: Intra-layer fusion**
- **3-to-1 fusion:** ACT, NL and BL are fused to a Comparison layer.
- **Simplified Computation:** 2 Comparisons from ACT and BL and 5 floating-point operations from NL become an integer- comparison.
- **Less Storage:** 4 floating-point variables in NL become 1 integer.



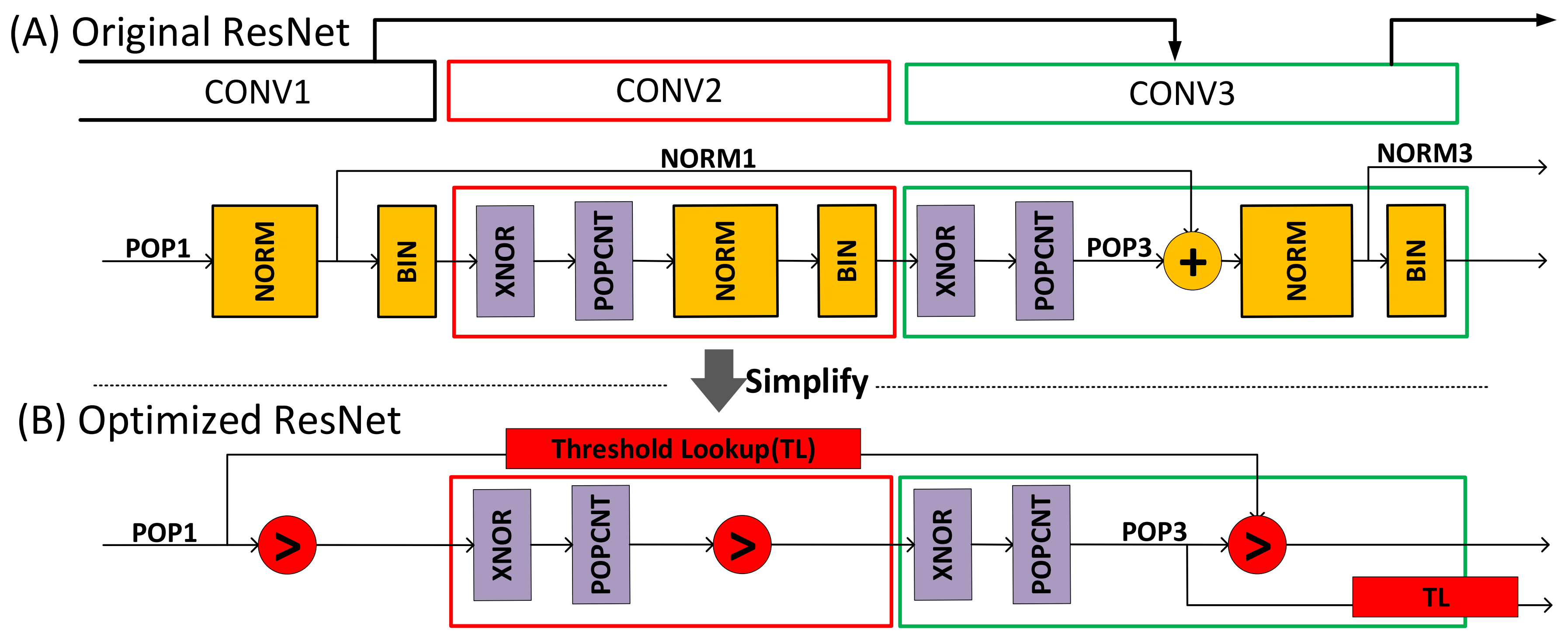
(A) Traditional structure of BNN



(B) Our optimized BNN

Intra-layer Fusion for Networks with Shortcuts

- ResNet



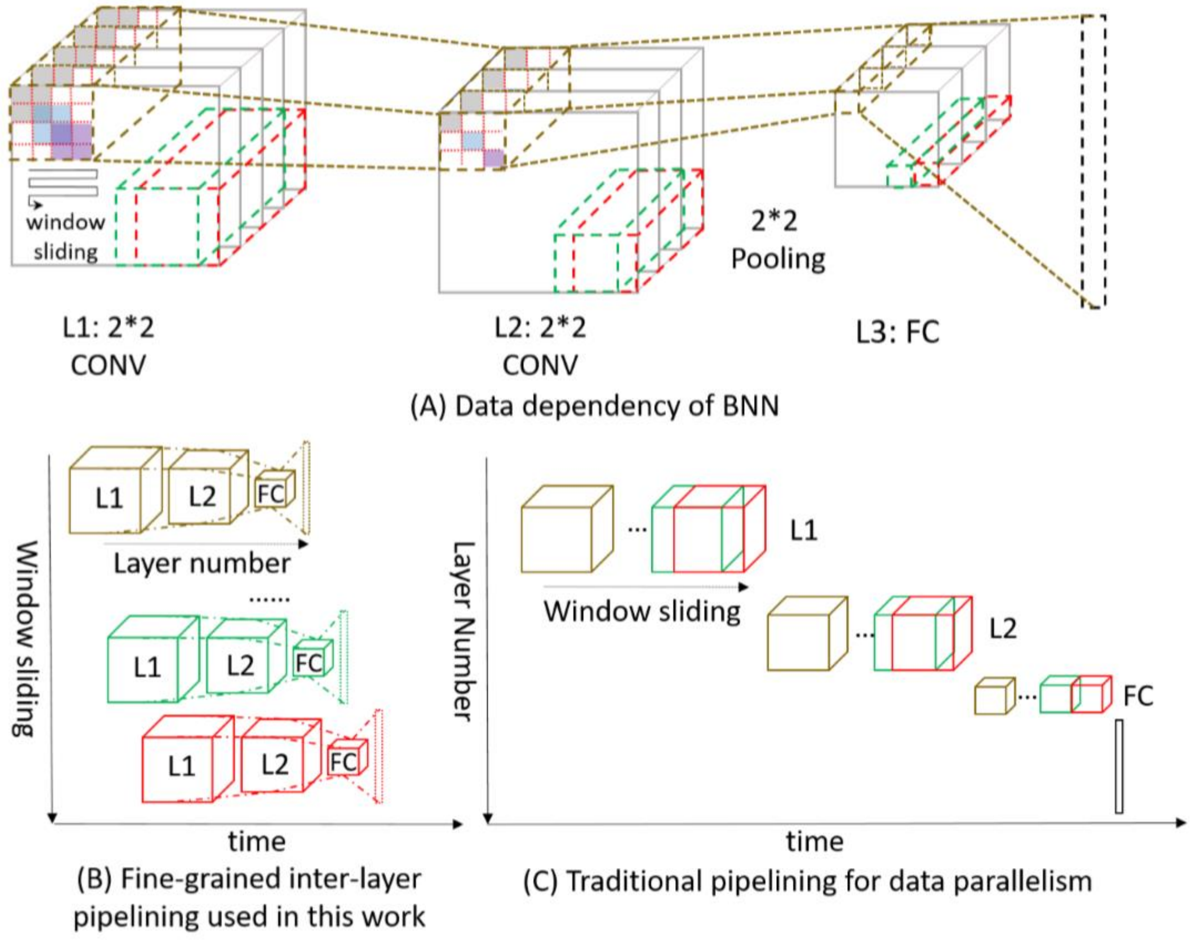
Challenges

- **Challenges to make truly low-latency inference using FPGA**

1. The critical Normalization Layer (NL) uses full-precision floating point (i.e., 2 FP MUL/DIV + 3 FP ADD/SUB).
2. Existing works process layers sequentially. Hence, their latencies are accumulated with no overlapping.
3. Optimal designs for all layers need to be simultaneously configured on FPGA with no reconfiguration.

Inter-layer Fusion

- **Addressing the 2nd challenge: inter-layer fusion**
- Fine-grained pipelining to fuse CONV & 1st FC layers.
- An image is processed based on the data dependency.
- Layers are processed in parallel ➔ overlapped latencies.



(A) Data dependency of BNN

(B) Fine-grained inter-layer pipelining used in this work

(C) Traditional pipelining for data parallelism

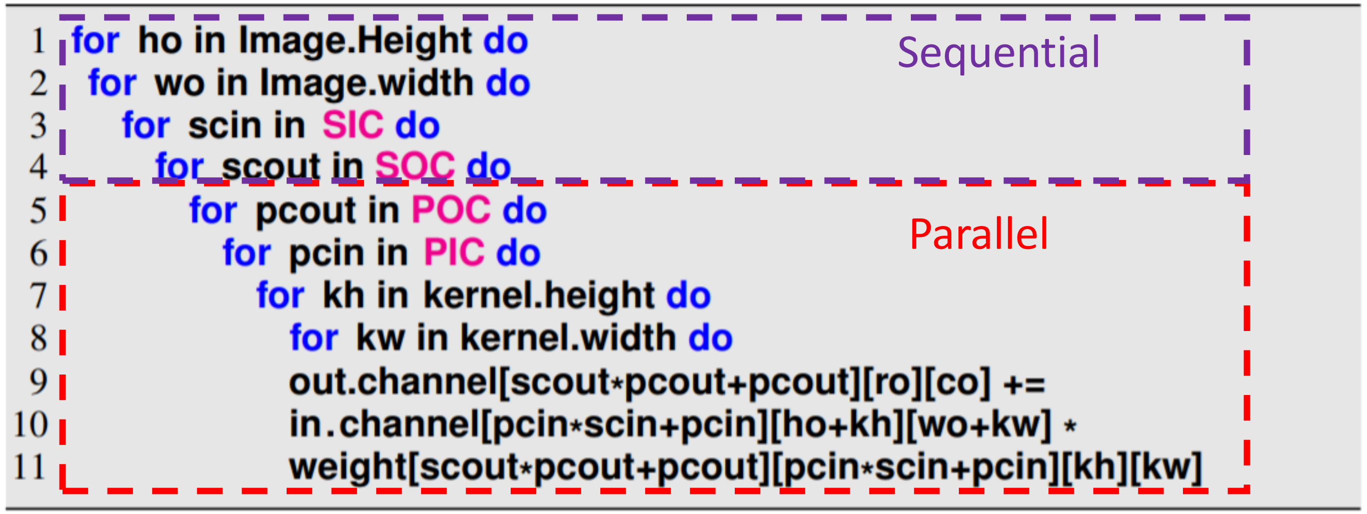
Challenges

- **Challenges to make truly low-latency inference using FPGA**

1. The critical Normalization Layer (NL) uses full-precision floating point (i.e., 2 FP MUL/DIV + 3 FP ADD/SUB).
2. Existing works process layers sequentially. Hence, their latencies are accumulated with no overlapping.
3. Optimal designs for all layers need to be simultaneously configured on FPGA with no reconfiguration.

Workload Balancing

- Each Layer: NIC*NOC*K*K
  - NIC=Num of Input Channel
  - NOC=Num of Output Channel
  - K=Kernel size
- Terms: PIC, POC, SIC, SOC
  - PIC=Parallelism of Input Channel
  - POC=Parallelism of Output Channel
  - SIC=Sequential of Input Channel
  - SOC=Sequential of Output Channel

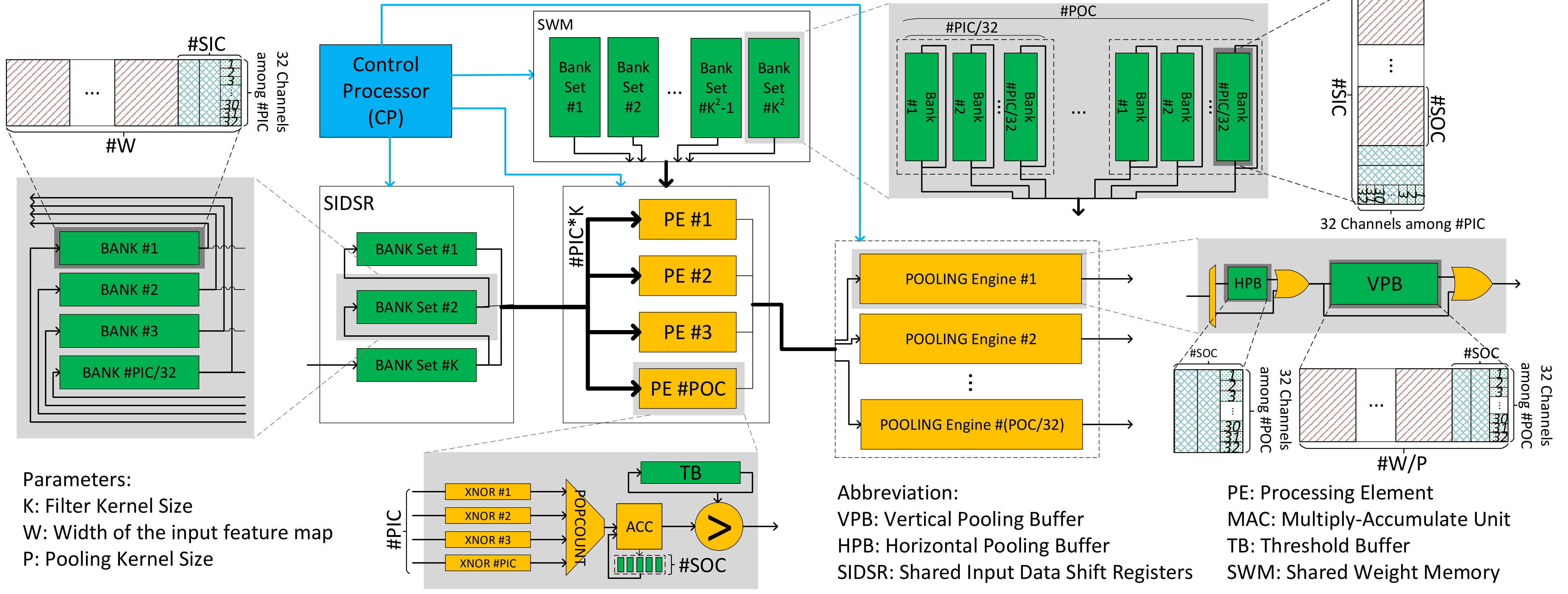


```
for ho in Image.Height do
 for wo in Image.width do
  for scin in SIC do
   for scout in SOC do
    for pcout in POC do
     for pcin in PIC do
      for kh in kernel.height do
       for kw in kernel.width do
        out.channel[scout*pcout+pcout][ro][co] +=
        in.channel[pcin*scin+pcin][ho+kh][wo+kw] *
        weight[scout*pcout+pcout][pcin*scin+pcin][kh][kw]
```

- Match Data Production and Consumption Rates of adjacent layers by adjusting PIC, POC, SIC and SOC

Parameterized Architecture

- **Addressing the 3rd challenge: decent architecture design**
- The architecture is flexible enough to support load balancing.
- All layers are fully configured on FPGA using model parallelism.

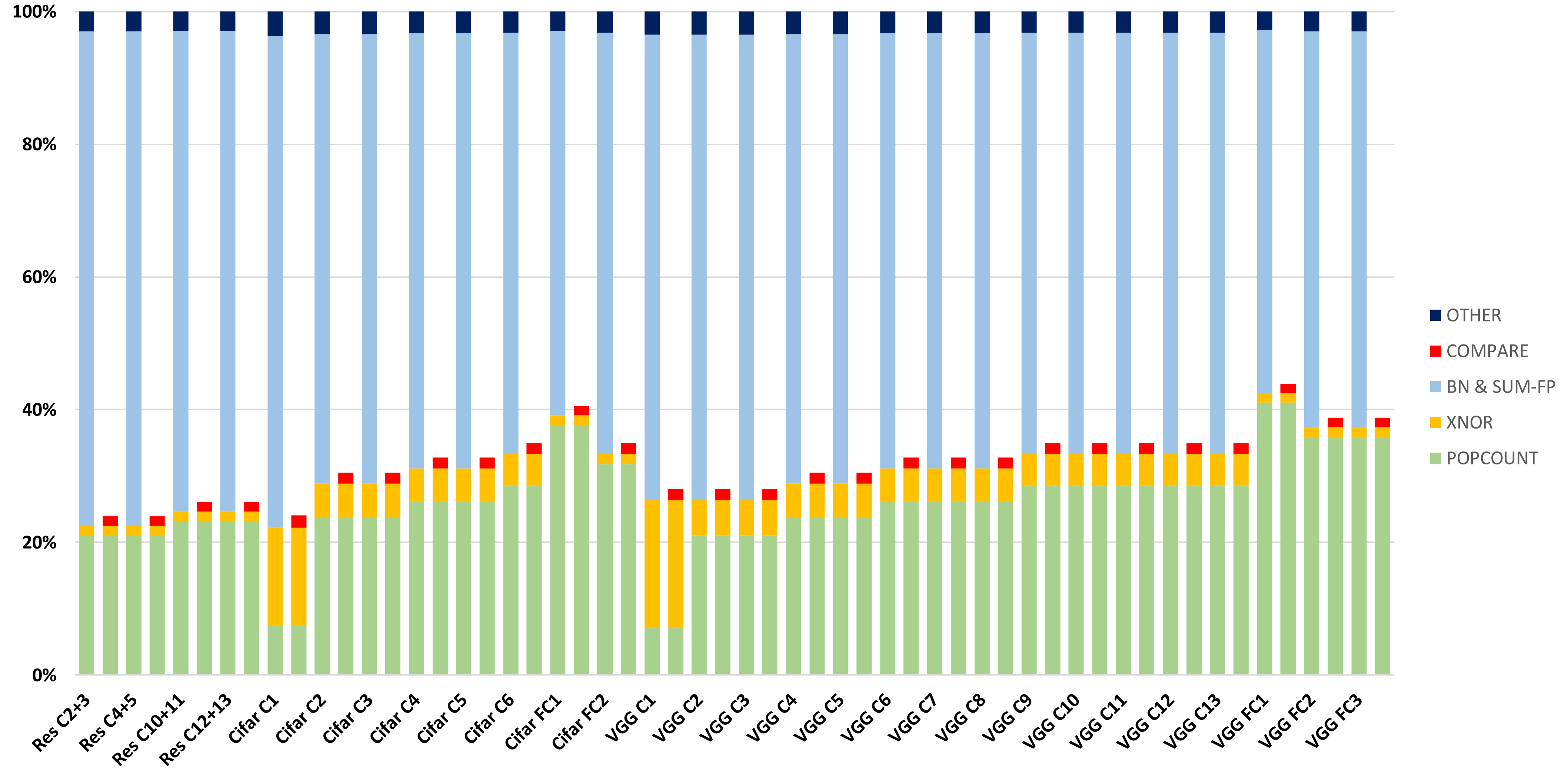


Parameters:
K: Filter Kernel Size
W: Width of the input feature map
P: Pooling Kernel Size

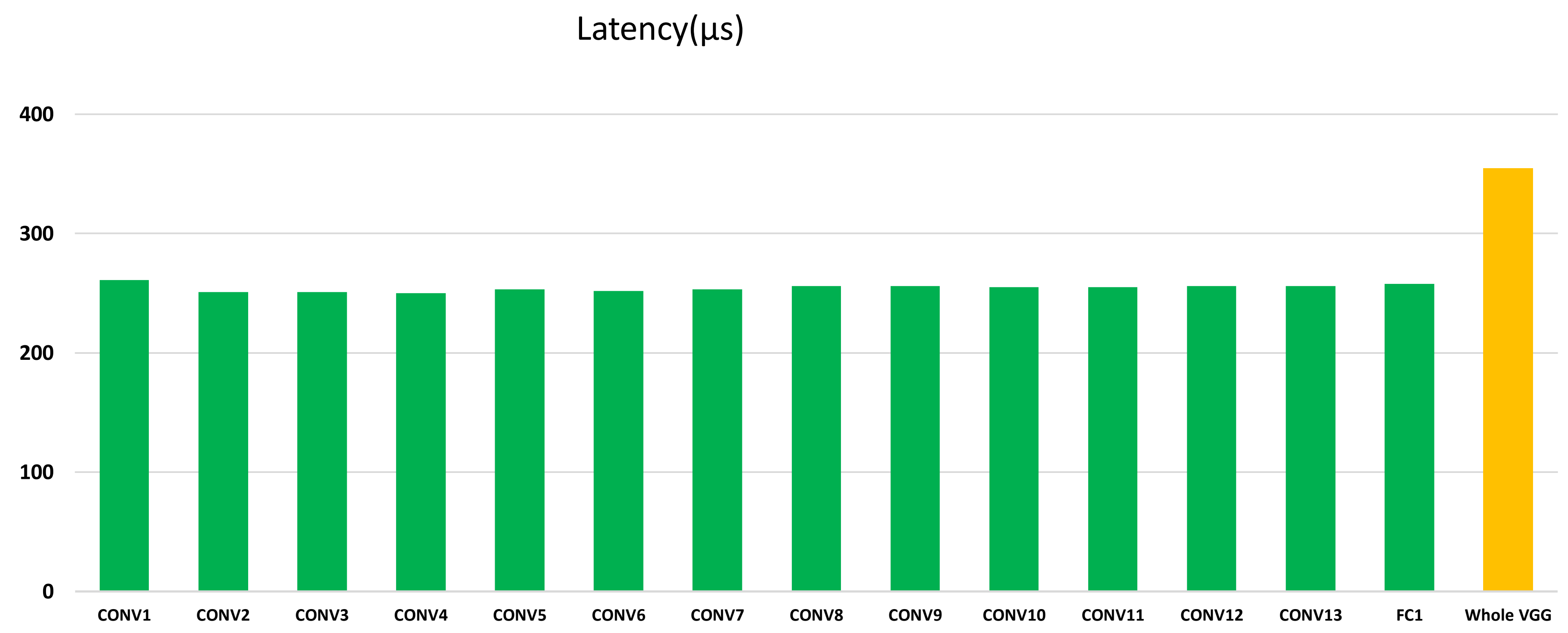
Abbreviation:
VPB: Vertical Pooling Buffer
HPB: Horizontal Pooling Buffer
SIDSR: Shared Input Data Shift Registers

PE: Processing Element
MAC: Multiply-Accumulate Unit
TB: Threshold Buffer
SWM: Shared Weight Memory

Evaluation: Latency Reduction from Intra-layer Fusion

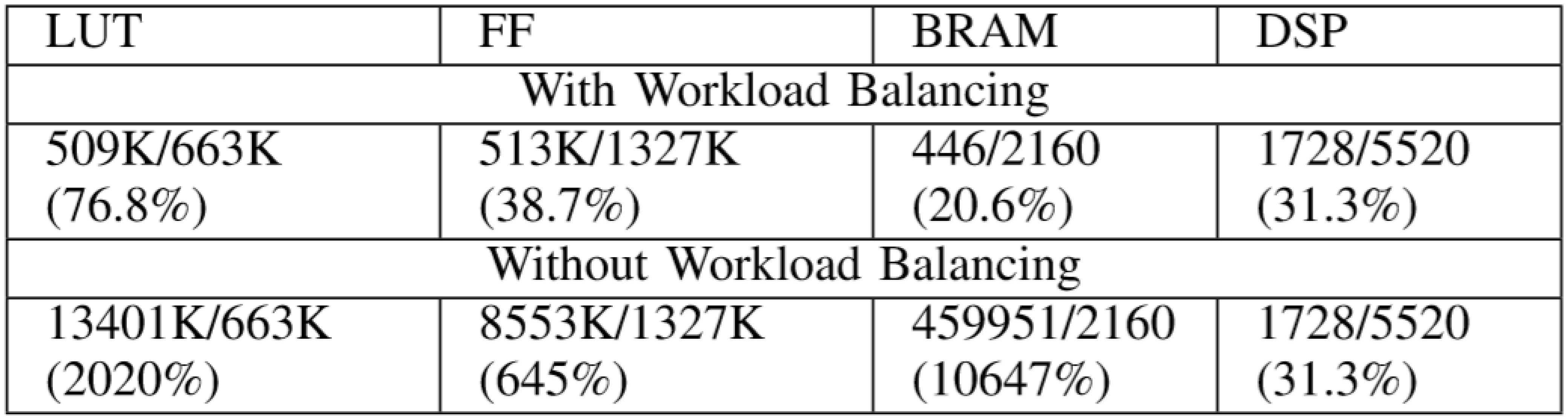


Evaluation: Single Layer Latency VS Whole Latency



Evaluation: Hardware Resource Saving from Workload Balancing

TABLE I
RESOURCE USAGE OF LUTS, FFS, BRAMS, AND DSPS FOR VGG INFERENCE WITH/WITHOUT WORKLOAD BALANCING (SAME LATENCY)



LUT	FF	BRAM	DSP
With Workload Balancing			
509K/663K (76.8%)	513K/1327K (38.7%)	446/2160 (20.6%)	1728/5520 (31.3%)
Without Workload Balancing			
13401K/663K (2020%)	8553K/1327K (645%)	459951/2160 (10647%)	1728/5520 (31.3%)

Evaluation: BNN Inference Latency using LP-BNN

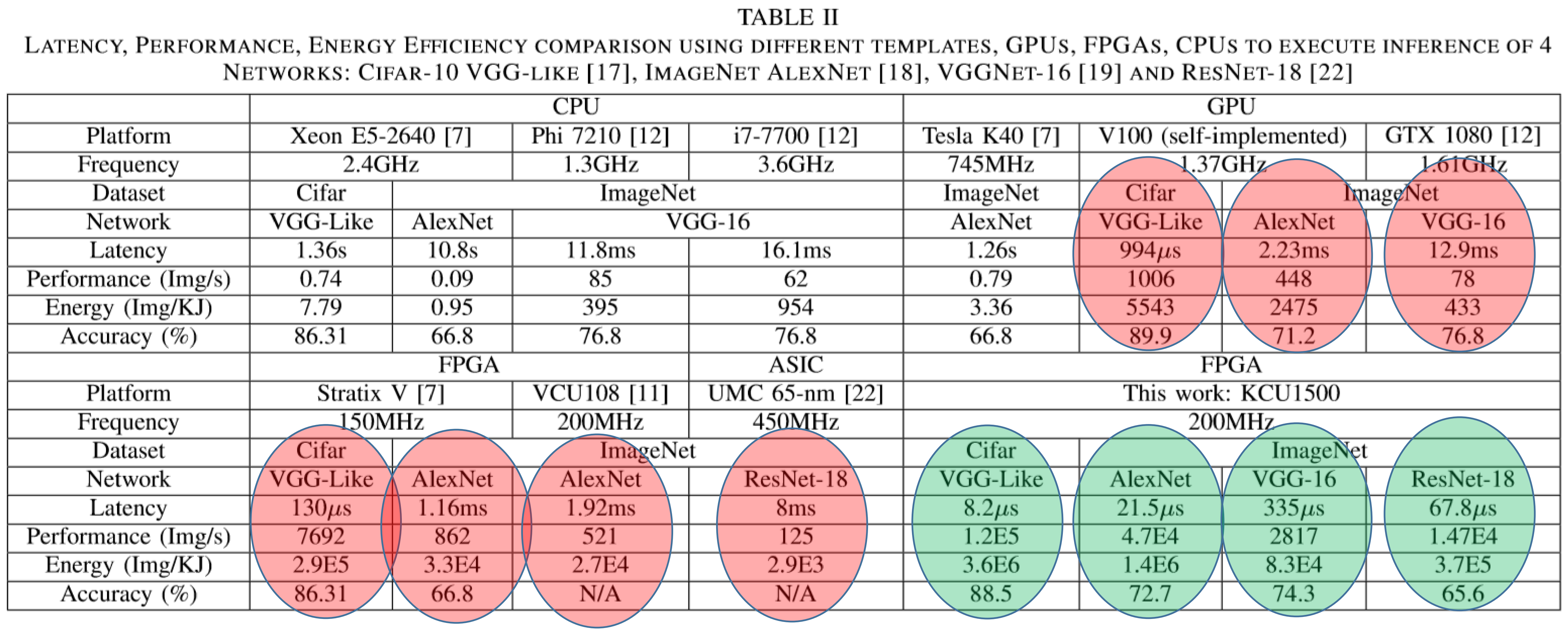


TABLE II

LATENCY, PERFORMANCE, ENERGY EFFICIENCY COMPARISON USING DIFFERENT TEMPLATES, GPUS, FPGAS, CPUS TO EXECUTE INFERENCE OF 4 NETWORKS: CIFAR-10 VGG-LIKE [17], IMAGENET ALEXNET [18], VGGNET-16 [19] AND RESNET-18 [22]

	CPU				GPU			
Platform	Xeon E5-2640 [7]		Phi 7210 [12]	i7-7700 [12]	Tesla K40 [7]	V100 (self-implemented)		GTX 1080 [12]
Frequency	2.4GHz		1.3GHz	3.6GHz	745MHz	1.37GHz		1.61GHz
Dataset	Cifar	ImageNet			ImageNet	Cifar	ImageNet	
Network	VGG-Like	AlexNet	VGG-16		AlexNet	VGG-Like	AlexNet	VGG-16
Latency	1.36s	10.8s	11.8ms	16.1ms	1.26s	994μs	2.23ms	12.9ms
Performance (Img/s)	0.74	0.09	85	62	0.79	1006	448	78
Energy (Img/KJ)	7.79	0.95	395	954	3.36	5543	2475	433
Accuracy (%)	86.31	66.8	76.8	76.8	66.8	89.9	71.2	76.8
	FPGA			ASIC	FPGA			
Platform	Stratix V [7]		VCU108 [11]	UMC 65-nm [22]	This work: KCU1500			
Frequency	150MHz		200MHz	450MHz	200MHz			
Dataset	Cifar	ImageNet			Cifar	ImageNet		
Network	VGG-Like	AlexNet	AlexNet	ResNet-18	VGG-Like	AlexNet	VGG-16	ResNet-18
Latency	130μs	1.16ms	1.92ms	8ms	8.2μs	21.5μs	335μs	67.8μs
Performance (Img/s)	7692	862	521	125	1.2E5	4.7E4	2817	1.47E4
Energy (Img/KJ)	2.9E5	3.3E4	2.7E4	2.9E3	3.6E6	1.4E6	8.3E4	3.7E5
Accuracy (%)	86.31	66.8	N/A	N/A	88.5	72.7	74.3	65.6