Evaluating Versal ACAP and conventional FPGA platforms for AI inference

Aimilios Leftheriotis[*], Aphrodite Tzomaka, Dimitrios Danopoulos, George Lentaris, George Theodoridis, Dimitrios Soudris

School of Electrical & Computer Engineering, University of Patras, Patras, Greece
School of Electrical & Computer Engineering, National Technological University of Athens, Athens, Greece
Corresponding Author Email: aimilefth@microlab.ntua.gr

Abstract

Xilinx Versal ACAP is the newest acceleration platform, developed by Xilinx, proposed to enhance the capabilities of the conventional FPGA ones and meet the demands of modern applications. However, only few studies concerning its benefits have been performed. To address this issue, a comparison between this platform and the MPSoC FPGA is performed by targeting Deep Learning applications. Using the Vitis AI inference framework, a large number of convolution and fully-connected models were implemented. This exploration lead to several conclusions regarding the optimal platform selection depending on the AI model characteristics. Also, to further evaluate the benefits and the programmability trade-offs of the Versal ACAP, a custom architecture of an image super-resolution model (ESPCN) was developed. Compared to the implementation derived by the Vitis AI framework, the custom design improves latency by 4.5x.

Versal ACAP, Vitis AI, Deep Learning, ISR architecture, AI Inference

1 Introduction

Over the last few years, there has been a proliferation in the use of Deep Learning (DL) algorithms in almost every scientific domain. This is especially true for Computer Vision, where in multiple tasks, like Image Classification[1], [2], [3], Semantic Segmentation [4], Object Detection [5], [6], and Image Super-Resolution (ISR) [7], DL models are considered the best approach. Thus, performant implementations of such algorithms are of great importance.

Using CPU-only platforms can prove inefficient, given the many sources of parallelism that are left unexploited and the lack of computation-specific

[*]Work partially supported by H2020 project "AI@EDGE" (g.a. 101015922)

hardware. Hence, it is common practice to use specialized hardware such as FPGAs in an attempt to offload and accelerate critical parts of an algorithm [8]-[11]. Their reconfigurable hardware allows the creation of application-specific accelerators that utilize parallelism, streaming, pipelining, and arbitrary bit-width of data.

However, FPGAs often exhibit some inherent weaknesses. First of all, FPGAs operate on far lower clock frequencies than CPUs thus need very high parallelization to compensate for the slow clock speed. Additionally, the modern DL models contain so much parallelization potential that, while the resources of FPGAs (LUTs, Flip-Flops, DSPs) are abundant, they cannot fully utilize the available parallelism due to timing constraints, leading to missed acceleration opportunities. On top of that, the on-chip memory of FPGAs is limited, which hinders the exploitation of parallelism and data reuse. This increases the need for off-chip memory accesses, which are costly in latency and power consumption. In [12], it was concluded that limited memory bandwidth is the main factor that obstructs FPGA implementations.

Xilinx Versal Adaptive Compute Acceleration Platform (ACAP) is a new acceleration paradigm that aims to enhance the existing FPGA platforms[13]. Regarding conventional embedded platforms (eg. ZCU104 MPSoC), a Versal ACAP contains similar components to a Zynq Ultrascale+ MPSoC, with the addition of the Artificial Intelligence Engine (AIE) array and a programmable NoC[14]. Introducing the AIE array seemingly alleviates the issue of underutilized parallelism. Additionally, the AIEs are being clocked at a higher frequency. The computation and resource offloading to the AIE array assists in creating better designs. Lastly, the programmable NoC provides better off-chip memory bandwidth compared to the one offered by MPSoCs.

These improvements are promising and could install the Versal ACAP as the staple DL accelerator. However, to the best of our knowledge, other than the Vitis AI Model Zoo [15], there has been no study that attempts to explore and compare MPSoC and Versal ACAP implementations, quantifying how those improvements translate to acceleration and which are the most influential, in practice. Thus, the results and outcomes of this work can be exploited during the exploration phase of selecting the proper platform to implement the application.

In this paper, we use the Vitis AI inference framework to compare the ZCU104 and the VCK190 platforms by benchmarking multiple neural network models that cover a vast design space, with a focus on convolutional and fully connected networks. Implementing models with a wide variety of configurations is a systematic approach since the results can be extrapolated to most DL models and provide more insights into which parts of the platform are most influential.

Furthermore, while there is a plethora of implementations of DL models on MPSoCs [9]-[11], the implementations on Versal ACAPs are limited [16], [17]. We provide a custom implementation of an ISR model, improving latency by 4.5x than the respective Vitis AI framework implementation on the VCK190. This acts as proof that Versal ACAP programming enables the creation of powerful custom architectures.

The main contributions of this paper are:

- Exploration and comparison of the ZCU104 Ultrascale+ MPSoC and the VCK190 Versal ACAP platforms, using a large set of convolutional and fully connected Vitis AI model implementations.
- Analysis of which Versal ACAP architectural advances are the most influential for accelerating ML applications.
- A custom implementation of an ISR model (ESPCN) on VCK190, to explore the performance versus programming effort tradeoff on Versal ACAPs. Performance comparison between the custom architecture on VCK190 and the Vitis AI framework implementation.

The remainder of the paper is structured as follows. Section II provides a brief introduction to the Versal ACAP and Vitis AI framework. We also present our approach toward the custom implementation of the ISR model and discuss the selection of the architectural parameters of the implemented NN models. In Section III, we present and discuss the experimental results. Finally, in Section IV, we draw conclusions based on our findings.

2 Background and Proposed Approach

2.1 Xilinx Versal ACAP and Vitis AI

Xilinx Versal ACAP [13] is a hybrid compute platform that combines four main components: (a) the Processing System (PS), which includes the ARM processors, (b) the Programmable Logic (PL), which is the traditional FPGA reconfigurable fabric, (c) the AIE array, which contains the software programmable accelerator engines, and (d) the programmable NoC that connects all the components.

The most important units included in an AIE are: (a) the VLIW processor with SIMD vector registers, which is the main source of acceleration, (b) the scalar processor, which handles the scalar operations, and (c) the memory module, that contains 8 one-port memory banks of 4KB each. Each AIE has direct memory access at up to four memory modules (north, south, east, west). Additionally, all AIEs are connected via the AXI4 Interconnect, allowing the utilization of the AXI4-Stream protocol. Lastly, the cascade connections enable horizontal interconnection between AIEs.

The AIE array provides three levels of parallelism: (a) coarse-grained, the AIE array consists of up to 400 AIEs that can execute fully in parallel, (b) SIMD, with vector computing enabling multiple elements to be computed in parallel, and (c) instruction level, where the VLIW architecture allows up to 7 instructions to be executed in a clock cycle. The AIE kernels are programs written in C/C++ and compiled using specialized intrinsic calls or AIE APIs, targeting the VLIW processor.

Xilinx Vitis AI is an AI inference framework, targeted towards Xilinx devices, including Zynq Ultrascale+ MPSoCs and Versal ACAPs. The framework implements the inference of a DL model to the Deep-learning Processing Unit (DPU), which is a programmable engine optimized for deep neural networks. The DPU differs from device to device, both in terms of operation support and architecture. For MPSoCs, the DPU is implemented using solely PL resources, whereas for Versal ACAPs, the DPU contains both PL resources and AIEs. In this work, we aim to quantify how influential the addition of the AIEs is for the performance of the DPU.

2.2 Study of Convolution Blocks

A specific combination of multiple different layers, is called a *block* in the ecosystem of DL models. The vast majority of modern convolutional DL models consist of multiple convolutional blocks that contain the same architectural pattern. The architecture of the blocks, the number of blocks used, along with some fine-grained parameters, vary from model to model. To get concise and accurate results, we need some convolutional block architectures which contain key elements that appear among the most widely used convolutional DL models.

The ResNet Bottleneck block [1], the MobileNetV2 block [3], and the ConvNeXt block [2] are some representative convolutional blocks. Firstly, the Resnet Bottleneck block covers the ResNet-like family of DL models that are used widely and are considered a benchmark in the field of DL for Computer Vision. Additionally, the MobileNetV2 block and some slight variations appear in most edge devices, where the networks need to be concise to adhere to the hardware restrictions presented. Finally, the ConvNeXt block is the one used in the family of ConvNeXt models, which are considered State-of-the-art in Image Classification and Feature Extraction. Figure 1 depicts the detailed architecture of each block, in layer level. Each block contains three convolution-type layers which are accompanied by some lightweight layers and a residual connection. Table 1 provides the distinct and noteworthy characteristics that define each of the blocks.

Table 1: Characteristics of Convolutional Blocks

ResNet Bottleneck block	**MobileNetV2 block**	**ConvNeXt block**
Widespread use	Edge Devices	State-of-the-art
Residual Connection	Depthwise Conv	7x7 Depthwise Conv
Bottleneck	Inverted Bottleneck	GeLU
Batch Normalization	Linear Residuals	Linear Normalization

In order to explore the capabilities of the MPSoC and the Versal ACAP, we conducted experiments in an extended design space using the Vitis AI framework on the VCK190 and ZCU104 platforms. We explore models that are composed of various numbers of blocks. The considered convolution blocks are the ResNet

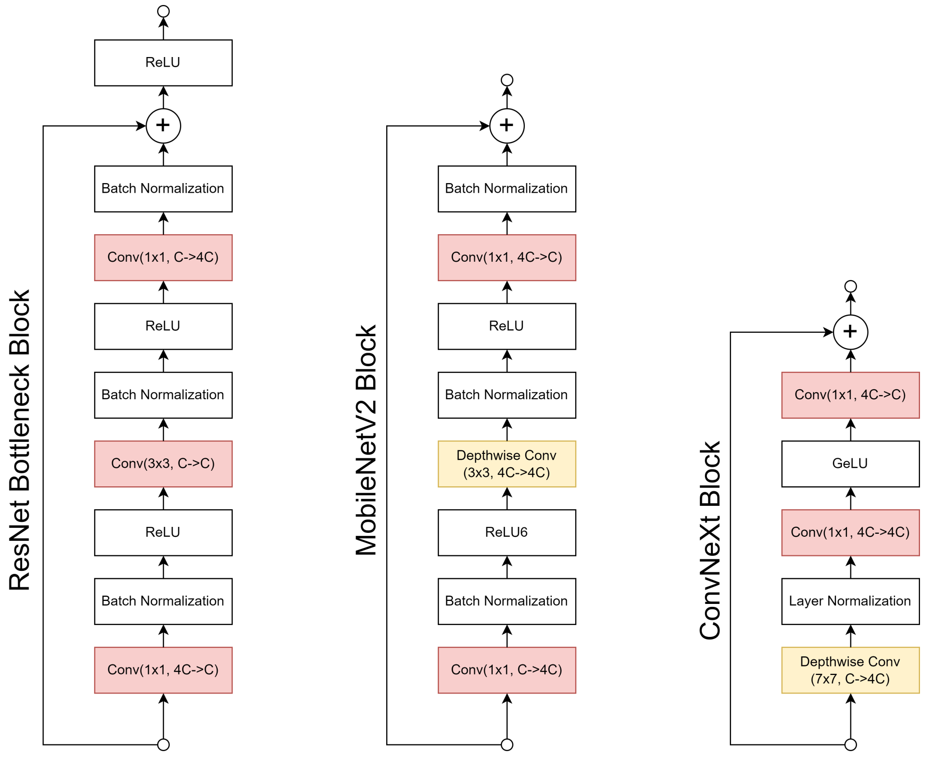

Figure 1: Layer Architecture of Convolutional Blocks (bottleneck = 4)

Bottleneck, the MobileNetV2, and the ConvNeXt blocks. Moreover, to expand our study beyond convolutional DL models, we explore a fourth case, where the block is comprised of a Fully Connected layer.

2.3 Custom Implementation of ISR model (ESPCN)

We propose a custom implementation of an ISR model on VKC190 platform. The ESPCN model was chosen as the network to be implemented, due to its efficiency and performance[7]. The network employs two classical and one Sub-Pixel convolution layers. The proposed mapping of these layers and the system design is illustrated in Figure 2.

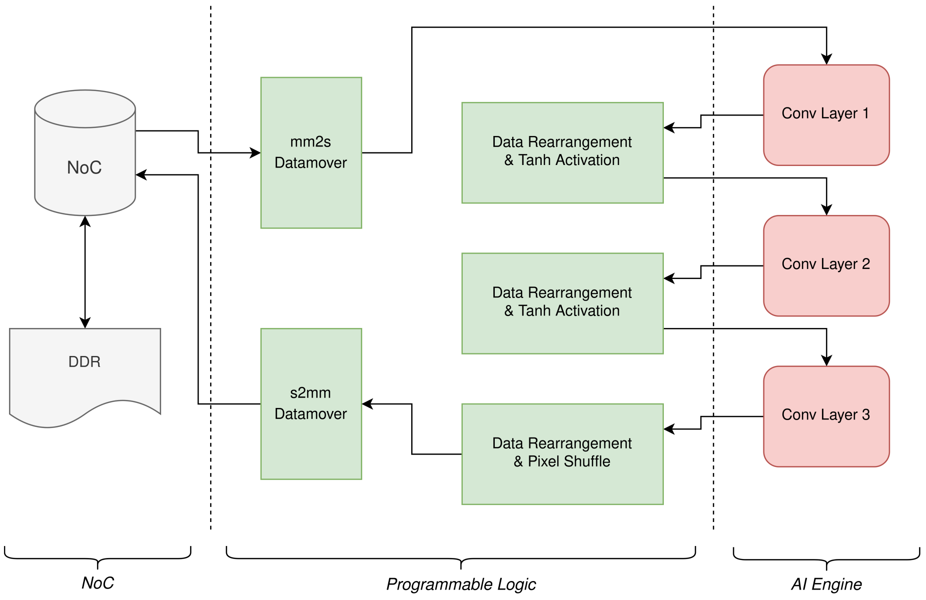
NoC
DDR
mm2s Datamover
s2mm Datamover
Data Rearrangement & Tanh Activation
Data Rearrangement & Tanh Activation
Data Rearrangement & Pixel Shuffle
Conv Layer 1
Conv Layer 2
Conv Layer 3
NoC
Programmable Logic
AI Engine

Figure 2: Block Diagram of Custom Implementation

In order to leverage the powerful computing capabilities of the AIEs, we implement the most computationally intensive convolution layers on the AIE array. With appropriate data rearrangement, the convolution computation is transformed to matrix multiplication and optimized to be implemented in the AIE array. The remaining operations, which include but are not limited to the Tanh activation function, the PixelShuffle function, and the data rearrangement, involve scalar byte operations and interact with memories, therefore are suitable for implementation in the PL. Finally, the design uses the PS as an external controller to orchestrate the data movements between the AIE graph and PL kernels. It should be highlighted that to avoid severe result degradation, which is common when quantizing ISR models, the implementation uses FP32 data types, while Vitis AI uses exclusively INT8.

3 Experimental Results

3.1 Experimental Setup

Table 2 displays the parameters for each block type and platform. Their selection is based on attempting to replicate prevalent model configurations while ensuring sufficient diversity to achieve generalizability on the pool of modern DL models. In more detail:

- The feature map size and the number of channels refer to the input to the convolutional blocks. By design, the input and the output of each block have the same dimensions.
- Bottleneck refers to the bottleneck architecture that exists at the ResNet Bottleneck block or the inverse bottleneck architecture that exists at MobileNetV2 and ConvNeXt blocks.
- The number of blocks indicates how many consecutive blocks a model is comprised of.
- The input size and the number of neurons of each Fully Connected layer is equal. This decision enables each block to have the same memory and computation requirements by forcing the input and output of each block to have the same dimensions.
- Finally, the VCK190 DPU enforces the batch size to be equal to 6, while when it comes to ZCU104, two ZCU104 DPUs work in parallel to achieve a batch size equal to two.

Table 2: Parameters for Vitis AI experiments

Block type	**Parameters**	**VCK190**	**ZCU104**
Convolutional blocks	**feature map size**	14x14, 28x28, 56x56	
	channels	32, 128, 512	
	bottleneck	2, 4, 8	
	number of blocks	1, 2, 4, 8, 16, 32	
	batch size	6	2
Fully Connected	**input size and number of neurons**	10, 20, 40, 80, 160, 320, 640, 1280, 2560	
	number of blocks	1, 2, 4, 8, 16, 32	
	batch size	6	2

The various parameter combinations lead to 162 ResNet Bottleneck, 162 MobileNetV2, 162 ConvNeXt, and 54 FC experiments per platform, which account for a total of 1080 experiments. Since our objective is to compare the MPSoC to the Versal ACAP, we get the VCK190 to ZCU104 ratio of each metric for each experiment. We will present the results following three approaches.

Firstly, we categorize the convolutional block experiments into 10 distinct categories, based on their Computation to Memory ratio (Comp/Mem). The Comp/Mem metric indicates whether a layer is dominated by the data transfers of the weights and feature maps or by the computation, information that is crucial to identify performance patterns and bottlenecks. Using this categorization, we can derive the Comp/Mem of any given convolution-based DL model and get an estimation of its relative performance on ZCU104 and VCK190. We split the experiments equally among the categories and average the ratios to achieve conciseness and presentability.

In the second approach, we categorize all the experiments based on their block type and their Comp/Mem ratio. Creating three categories based on the Comp/Mem ratio (small, medium, large), leads to a total of 12 distinct categories. This approach provides information about whether and how, the block architectural differences affect performance metrics.

In the last approach, the goal is to explore the effects of model depth. We categorized the experiments in the same way as in the second approach, but only included experiments with a single block. We then mapped each experiment with a certain configuration (e.g., feature map size, channels) and larger number of blocks to the same category as the one-block experiment with the same configuration. This allows us to study the performance trends of experiments with larger depth but the same parameters in other aspects.

The resource usage of the VCK190 Vitis AI design is: 192 AIEs, 403866 LUTs, 507308 FFs, 809 DSPs, 678 BlockRAMs, and 343 UltraRAMs. The resource usage of the ZCU104 Vitis AI design is: 102255 LUTs, 197691 FFs, 1420 DSPs, 255 BlockRAMs, and 64 UltraRAMs. Each BlockRAM and UltraRAM contain 36kb and 288Kb respectively.

3.2 Quantitative Metrics

The results of the first approach are presented in Figure 3. Each point represents the average of the VCK190 to ZCU104 ratio of the respective metric for each category, DPU throughput, DPU latency, and memory transfer bandwidth (between the PS and the DPU). Table 3 depicts the average Comp/Mem ratio of the experiments of each category.

Focusing on the DPU throughput, it is clear that the superior compute and parallelization power of the AIE engines enable the VCK190 DPU to outperform the ZCU104 one in terms of throughput. The performance improvements start from around 4x for the smaller category and rise as models get more computationally intensive, getting to a plateau of around 10x for category 60_70 and larger. This increase indicates that the Versal ACAP architecture is even more efficient when the implemented algorithms are computationally intensive and can be mapped appropriately to the capable AIEs.

Regarding the DPU latency, we observe that it follows a similar trajectory as the DPU throughput, starting at around 1x and falling almost to 0.3x. We observe that convolution-based models that belong to the category 0_10 (<0.2 GOP/MB) are more suitable for ZCU104. However implementations where latency is a concern would be benefited by choosing the VCK190 platform if the convolution-based model had a Comp/Mem ratio over 1 GOP/MB.

Lastly, the memory transfer bandwidth of the VCK190 is, on average, between 2x and 3x greater than that of the ZCU104. The addition of the NoC that connects the PS of the VCK190 with the PL and the AIEs can be attributed to this increase. However, in the context of DL, the impact of this improvement is not significant since feature map transfers should only occur at the start and end of inference, and weight transfers can be performed in parallel to layer computation. Therefore, the addition of the AIEs has a greater effect than the addition of the NoC in accelerating DL.

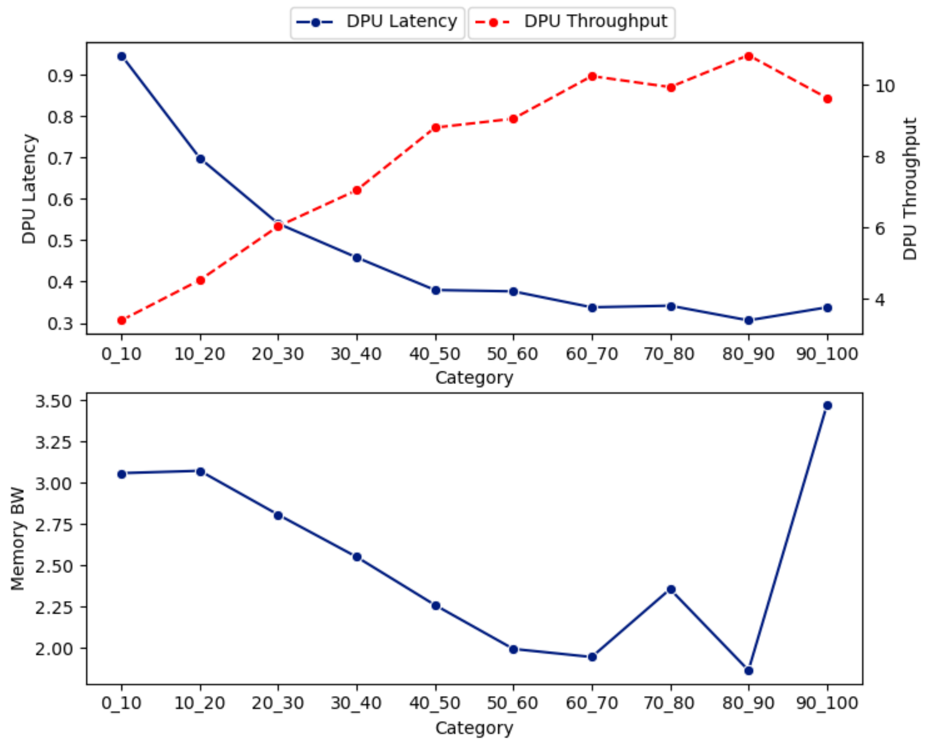

Figure 3: VKC190/ZCU104 Ratio for approach 1

The results of the second approach are presented in Figure 4. Table 4 depicts the average Comp/Mem ratio of the experiments of each category.

Similarly to the first approach, the increase in Comp/Mem leads to an increase in DPU throughput. However, there is a clear difference between models of different block architectures. Among the convolutional blocks, ConvNeXt is the most suitable for VCK190. Additionally, Fully Connected models can benefit from Versal ACAP too, especially ones with larger Comp/Mem.

Table 3: Average Comp/Mem ratio per Category on approach 1

Category	**0_10**	**10_20**	**20_30**	**30_40**	**40_50**
GOP/MB	0.112	0.372	0.699	1.137	1.647
Category	**50_60**	**60_70**	**70_80**	**80_90**	**90_100**
GOP/MB	2.026	2.595	4.471	7.111	17.137

The DPU latency graph demonstrates that ResNet-Like models, particularly those in the small category, can be targeted towards the ZCU104 platform with minimal loss compared to other block types. Conversely, ConvNeXt and MobileNetV2 models exhibit DPU latency of less than 0.6x, even in the small category. Finally, large Fully Connected models exhibit the most significant difference, at 0.2x.

Consistent with the memory bandwidth results of the first approach, the diagram suggests that the difference between the platforms is approximately between 1.5x to 3x.

Table 4: Average Comp/Mem ratio per Category in approach 2 and 3

Category	**small**		**medium**		**large**	
Approach	**2**	**3**	**2**	**3**	**2**	**3**
ResNet Block	0.141	0.034	0.741	0.14	4.0134	0.739
MobileNetV2 Block	0.894	0.299	2.447	0.901	10.792	3.002
ConvNeXt Block	0.924	0.316	2.481	0.902	10.826	3.002
Fully Connected	0.010	0.007	0.012	0.011	0.012	0.012

Figure 5 presents the results of the third approach, while Table 4 shows the average Comp/Mem ratio of the experiments of each category. The categorization diagrams of all experiments (approach two) and those with only one block are similar. However, the aim of the third approach is to emphasize the effects of model depth on performance metrics. Therefore, Figure 6 displays the more intriguing results.

DPU latency and DPU throughput depict similar behaviour. An increase in block number leads to an increase in Comp/Mem ratio, and hence, increased performance on VCK190, as the previous approaches indicated. However, this increase becomes less significant as the depth increases, with most categories reaching a plateau after depth 8 or 16. Additionally, blocks in categories with higher Comp/Mem ratios (large) receive more benefit from the Versal ACAP than their counterparts in categories with lower Comp/Mem ratios (small), a trend also observed in approach two.

Summarizing, based on the results of the three approaches, we can conclude that both Comp/Mem ratio and block architecture (in terms of block

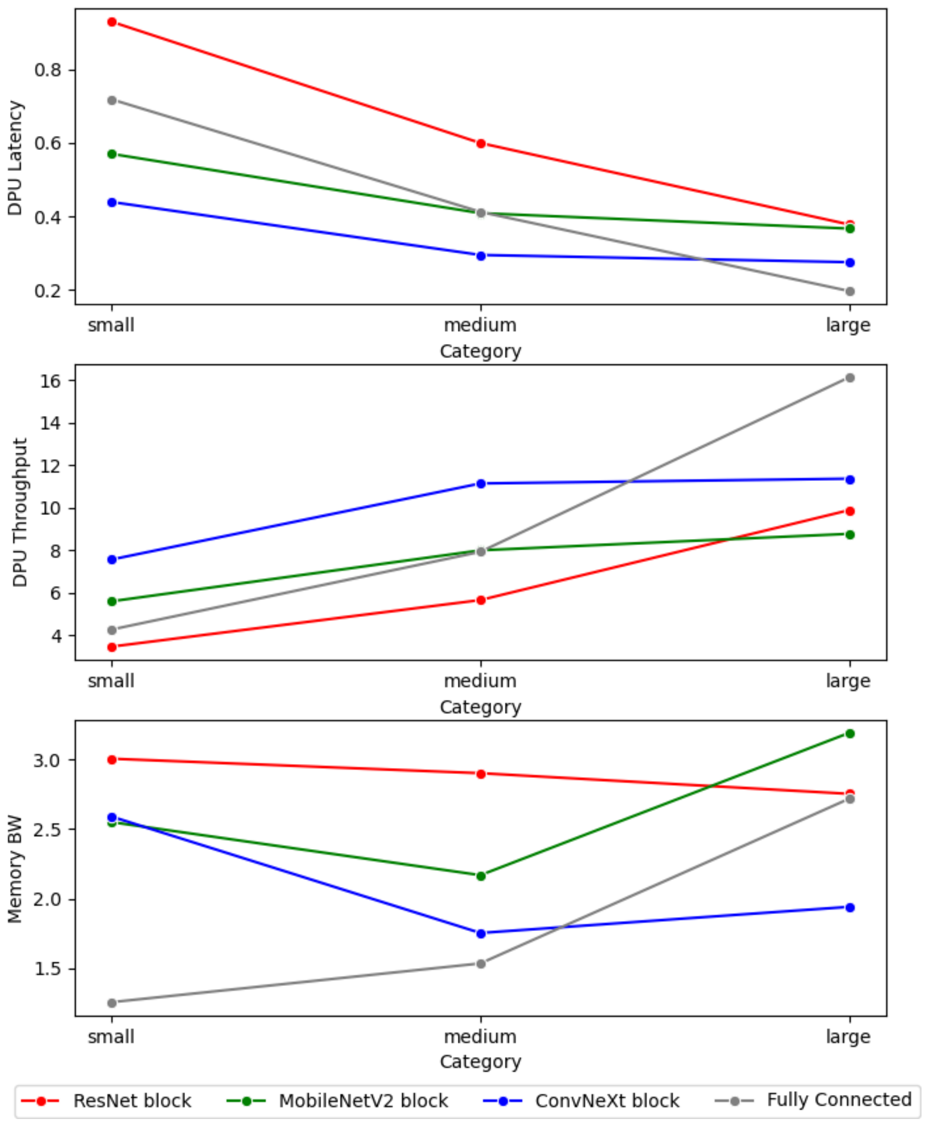

Figure 4: VKC190/ZCU104 Ratio for approach 2

parameters and model depth) significantly impact the performance difference between ZCU104 and VCK190. The results indicate that shallow (<4 blocks) convolutional-based models with smaller Comp/Mem ratio (<0.2 GOP/MB), particularly ResNet-like models, can be deployed in ZCU104 with negligible or no DPU latency loss and a 3x-4x DPU throughput loss Conversely, deeper models with larger Comp/Mem ratio should be directed towards Versal ACAP platforms, as they can utilize the AIEs better.

3.3 Qualitative Metrics

One significant qualitative metric is configurability, which specifies the extent that the Versal ACAP architecture can be programmed efficiently and whether the additional effort can lead to performance gains. To demonstrate that custom implementations in Versal ACAP are both viable and valuable, we compare our custom architecture on an ISR model on VCK190 and the Vitis AI framework implementation. The resource usage of the implementation is: 160 AIEs, 92654 LUTs, 136430 FFs, 30 DSPs, 342 BlockRAMs, and 81 UltraRAMs.

The latency of the custom implementation is 3.7 ms, while that of the Vitis AI implementation is 16.5 ms. This amounts to approximately a 4.5x latency decrease, despite the difference in data type usage (FP32 vs INT8). The results indicate that the massive parallelism capabilities of the AIEs can be exploited further and yield great results, which is proof of the configurability of the Versal ACAP.

An additional important qualitative metric is the scalability that the platform provides. We can quantify the scalability achieved by the Versal ACAP compared to the MPSoC by observing the availability of different layer types on the Vitis AI framework. While both VCK190 and ZCU104 DPUs support 2D layers, the VCK190 DPU is the only one supporting 3D layers and in particular: a) the 3D convolution layer, b) the depthwise 3D convolution layer, c) the transposed 3D convolution layer and d) the depthwise transposed 3D convolution layer [15]. The characteristic of those layers is that the added dimension both increases the memory requirements and complicates the computation, thus hindering the data reuse patterns. As a result, more off-chip memory accesses are required for the computation, requiring large memory bandwidth. The fact that the VCK190 DPU supports those 3D layers indicates that the programmable NoC, with the enhanced memory bandwidth, fulfills those requirements and it is the main reason why the Versal ACAP is more scalable than the MPSoC.

4 Conclusion

We presented a study on a new family of devices, namely the Versal ACAP and its potential in the software-hardware ecosystem for AI acceleration. We conducted several benchmarks using Vitis AI, an AI inference framework, for a variety of AI models and compared the results with a conventional FPGA board. The exploration concluded that deeper networks with larger Comp/Mem ratio

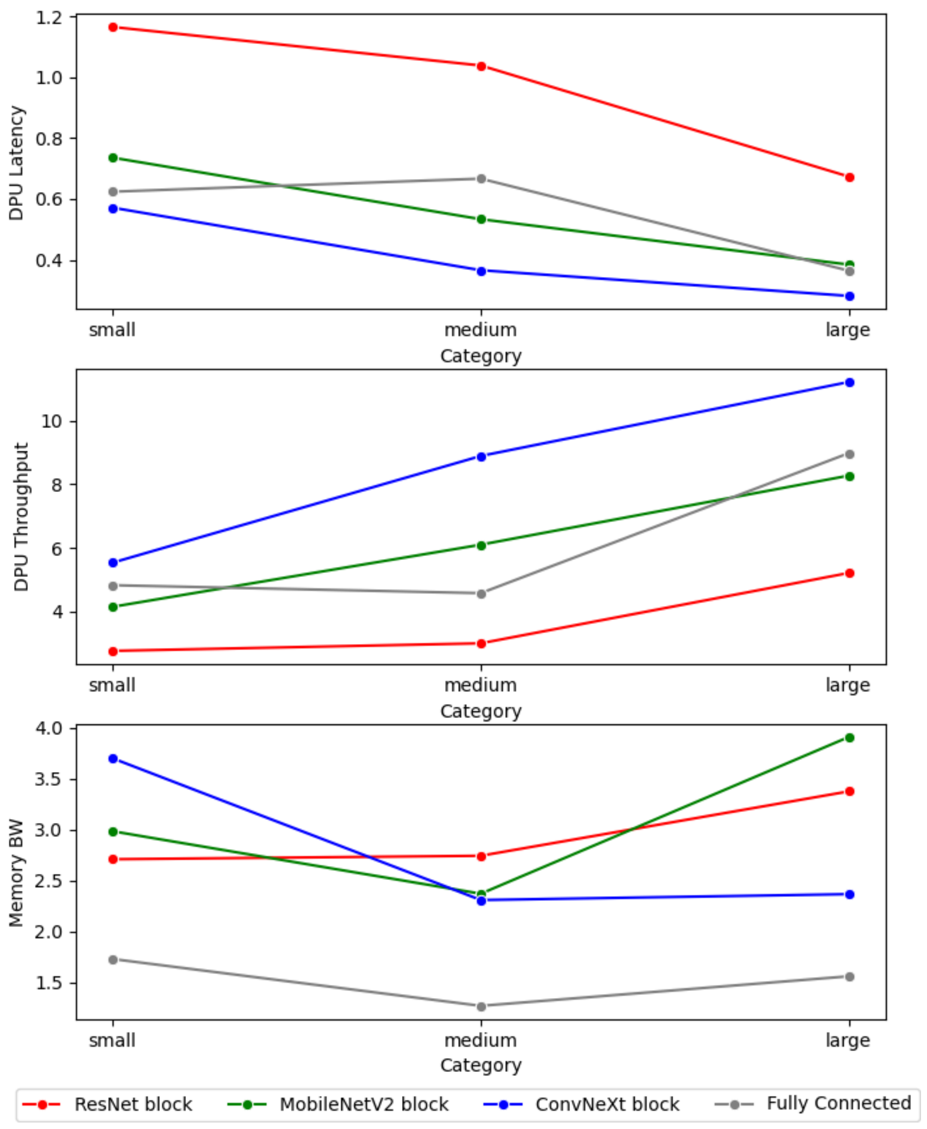

Figure 5: VKC190/ZCU104 Ratio for approach 3 (number of blocks = 1)

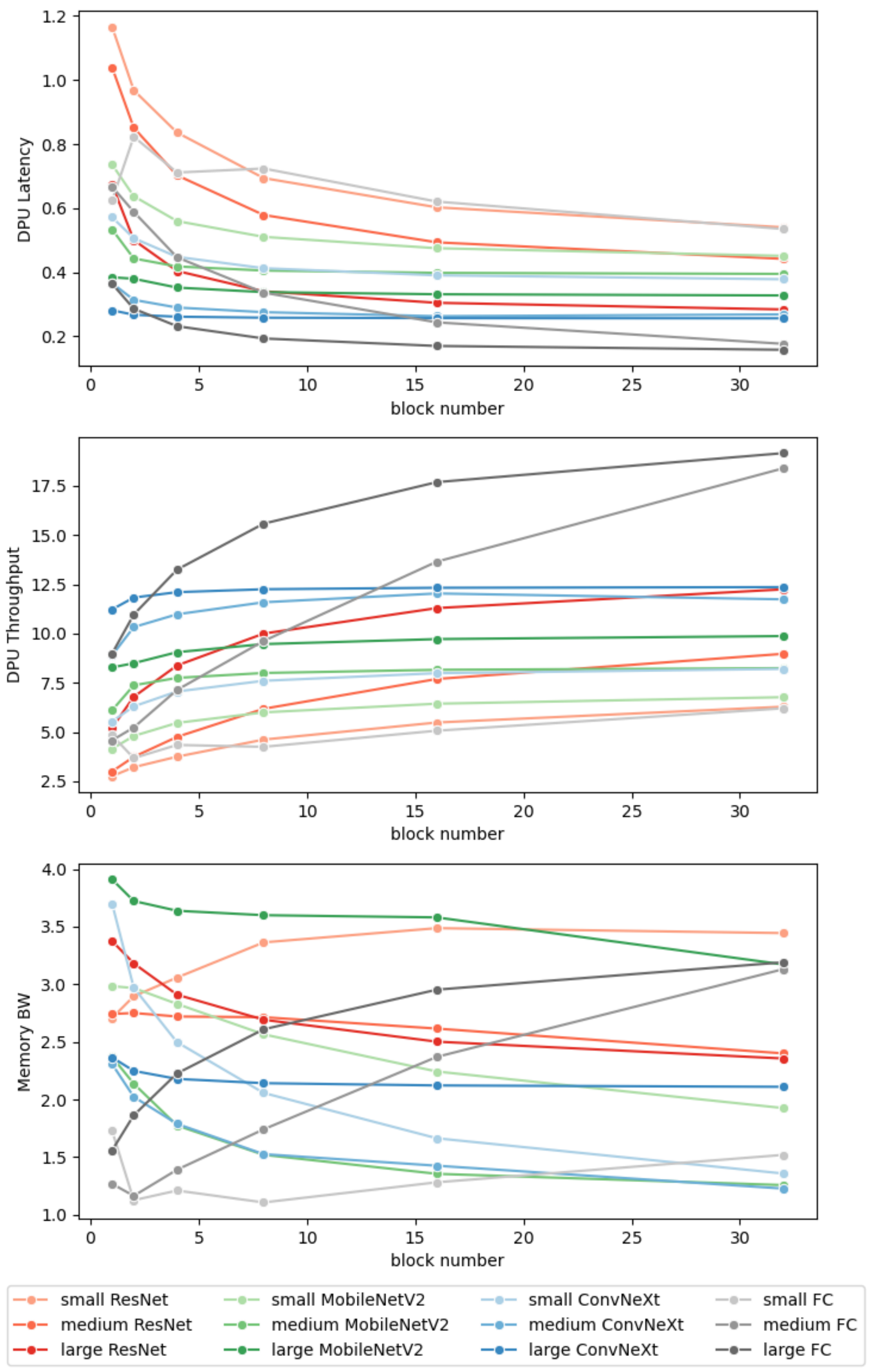

Figure 6: VKC190/ZCU104 Ratio for approach 3 (all experiments)

exhibit higher performance on Versal ACAP platforms. Also, we implemented a custom architecture of an ISR model (ESPCN) and showed that the custom architecture achieves better performance than the Vitis AI implementation, highlighting the performance versus programmability trade-off.

References

[1] K. He, X. Zhang, S. Ren and J. Sun,"Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778

[2] Z. Liu, H. Mao, C. -Y. Wu, C. Feichtenhofer, T. Darrell and S. Xie, "A ConvNet for the 2020s," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 11966-11976

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. -C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510-4520

[4] O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," arXiv:1505.04597, 2015

[5] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, June 2017, pp. 1137-1149

[6] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788

[7] J. Cai, Z. Meng, J. Ding and C. M. Ho, "Real-Time Super-Resolution for Real-World Images on Mobile Devices," 2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR), 2022, pp. 127-132

[8] S. Mittal, "A survey of FPGA-based accelerators for convolutional neural networks," in Neural Computing and Applications, vol. 32, 2020, pp. 1109–1139

[9] A. Shawahna, S. M. Sait and A. El-Maleh, "FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review," in IEEE Access, vol. 7, 2019, pp. 7823-7859

[10] D. Danopoulos, C. Kachris and D. Soudris, "Automatic Generation of FPGA Kernels From Open Format CNN Models," 2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2020, pp. 237-237

[11] D. Danopoulos, G. Zervakis, K. Siozios, D. Soudris and J. Henkel, "AdaPT: Fast Emulation of Approximate DNN Accelerators in PyTorch," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022

[12] J. Cong, Z. Fang, M. Lo, H. Wang, J. Xu and S. Zhang, "Understanding Performance Differences of FPGAs and GPUs," 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2018, pp. 93-96

[13] B. Gaide, D.Gaitonde, C. Ravishankar, and T. Bauer, "Xilinx Adaptive Compute Acceleration Platform: VersalTM Architecture," 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '19), 2019, pp. 84–93

[14] I. Swarbrick, D. Gaitonde, S. Ahmad, B. Gaide and Y. Arbel, "Network-on-Chip Programmable Platform in VersalTM ACAP Architecture," 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '19), 2019, pp. 212–221

[15] "Xilinx Vitis AI model zoo" https://github.com/Xilinx/Vitis-AI/tree/master/model_zoo, 2022, online

[16] C. Zhang *et al.*, "H-GCN: A Graph Convolutional Network Accelerator on Versal ACAP Architecture," 2022 32nd International Conference on Field-Programmable Logic and Applications (FPL), 2022

[17] J. Zhuang *et al.*, "CHARM: Composing Heterogeneous AcceleRators for Matrix Multiply on Versal ACAP Architecture," 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '23), 2023, pp. 153–164.