







Accelerated Deep-Learning Inference on the Versal adaptive SoC in the Space Domain

Michael Petry , *Graduate Student Member, IEEE, Student Member, Optica*, Gabriel Wuwer ,
Andreas Koch , Patrick Gest , Max Ghiglione , Martin Werner 

Abstract—Artificial intelligence has found its way into space and necessitates a powerful and flexible hardware platform to keep up with the fast-paced AI domain. With the space-grade variant of the Versal, AMD-Xilinx offers one of the first space-ready AI accelerators that combine multiple compute paradigms, i.e., scalar processing (CPU), adaptive engines (FPGA), and vector processing (AI-Engine array) into an adaptive System-on-Chip. This paper provides a thorough analysis of its AI capabilities with respect to throughput and power efficiency for Multi-Layer Perceptrons and CNNs, and takes a look under the hood by profiling the system's efficiency on an architectural level based on the idea of the Roofline model. We believe that the gained insights ultimately help to design optimal NN architectures for deployment on the Versal.

Index Terms—fpga, hardware accelerator, neural network, machine learning, AMD-Xilinx Versal, roofline model

I. INTRODUCTION

Future telecommunications satellites are envisioned to seamlessly integrate into the mobile communication networks of the next generation. First signs towards the integration are already visible today with the ongoing normative activities on non-terrestrial networks in the current 5G New Radio standard [1]. This is a novelty for satellite manufacturers, as it marks the first time that satellites are explicitly included in a mobile communications standard [2]. Within the same scope, efforts are made to integrate Artificial Intelligence (AI) / Machine Learning (ML) techniques into orbit to serve various use-cases, such as Cognitive Radio, Earth Observation, Anomaly Detection, and many more. There is a major desire in the satellite industry for deploying NNs and similar algorithms directly on-board of the satellites in space. The main challenges associated with that desire are the limited power budget and computing resources of satellites. Furthermore, due to the requirement to operate in the harsh space environment, telecommunications satellites are typically restricted to the use of radiation-hardened hardware, and in particular space-grade field-programmable gate arrays (FPGAs) as the most potent parallel compute platform for AI inference in space [3].

With the Versal adaptive System-on-Chip (SoC) in the XQR variant, AMD-Xilinx offers a chip in a space-grade

package that is particularly targeted for machine learning applications in space. By combining a scalar processing system (PS) with a high-compute-density FPGA fabric and ASIC-like vector engines, it promises more compute power at a higher efficiency than traditional single-technology systems [4], [5]. Its real strength, however, arises by utilizing these three compute paradigms in concert to perform ML operations in a deeply hardware optimized manner. To ease use for the developer, AMD-Xilinx provides an Intellectual Property (IP) block named Deep-Learning Processor Unit (DPU), which comprises an FPGA hardware image, instructions for the AI-Engines, and a data-handling and communication interface to the PS. An initial evaluation of the design flow and the basic capabilities of this framework on the Versal platform is investigated in [6].

Since the DPU is merely a black-box for the user, predictions on performance and suitability of specific NN architectures are hardly possible. Therefore, this work aims at providing transparency by investigating how different NN architectures impact the achievable performance on the Versal. By utilizing a grid-search we sweep through various network parameters to track throughput performance and power demands within the IP's core feature domain, i.e., Multi-Layer Perceptrons (MLP) and Convolutional Neural Networks (CNN). While this provides a broad superficial impression on the DPU's performance, we dive even deeper by profiling various architectural properties, such as hardware execution time, operational efficiency, and memory bandwidth utilization to study the internal reasons for the observed performance. This procedure aims at understanding how to design efficient NNs for hardware-accelerated scenarios, ultimately benefiting the complete range of ML application.

This paper is structured as follows: Section II provides a brief survey of state-of-the-art AI hardware accelerators for edge-deployment. Section III follows up on this by introducing the Versal hardware platform and the DPU, including a brief architectural review. The main contribution of this paper is given in section IV, which provides a throughput and power efficiency benchmark for several NN variations. Section V dives into the architectural level by computing a Roofline model, and investigates the origin of the observed behaviour. Lastly, a conclusion and outlook of future work is given in section VI.

This work is supported by the German Aerospace Center (DLR) under project Machine Learning on Telecommunications Satellites (MaLeTeSa) with Grant number 50 YB 2103. Corresponding author: Michael Petry (Email: michael.petry@airbus.com). M. Petry and A. Koch are with Technical University of Munich (TUM) and Airbus Defence and Space GmbH (ADS), G. Wuwer is with DHBW Ravensburg and ADS, Patrick Gest is with ADS, Max Ghiglione is with European Space Agency, and Martin Werner is with TUM.

II. STATE-OF-THE-ART AI ACCELERATORS

AI applications have found their way into our daily lives by penetrating into multiple domains such as entertainment, industrial, and health care, for instance with speech-to-text translation on phones, on-the-fly video post-processing for conference video chatting, and much more. ML-models have been observed to increase in size and complexity by one order of magnitude per year [7], which renders deploying hardware-compatible updates extremely difficult or even impossible. This gave rise to huge research and development efforts in building long-term hardware platforms. The proliferation of resource-intensive ML applications has pushed the tech industry into developing a "smarter" hardware platform that provides higher adaptivity and reprogrammability while outperforming CPUs and GPUs in metrics such as power and throughput. As a result, a variety of edge-targeted AI hardware accelerators have emerged, whereas the most modern platforms will be briefly summarized here.

The most popular AI accelerators are GPU-based. In terms of AI edge deployment, NVIDIA leads the field with its novel 7nm Orin System-on-Module (SoM), which is a successor to the Jetson platform. The Orin architecture comprises an NVIDIA Ampere GPU architecture with up to 2048 CUDA-cores, a 12-core ARM Cortex-A78 application processor, and a silicon-based AI computation unit, called Deep Learning Accelerator (DLA). The interplay of these elements makes the module extremely capable, yielding 275 Int-8 TOPS theoretic peak compute power at 4.58 TOPS/W, delivering superior power efficiency.¹ Native in-silicon execution of standard ML operations, such as CNNs or pooling, paired with the flexibility of a GPU provides the highest level of efficiency and flexibility. However, optimal utilization of the GPU depends on the concrete NN architecture to accelerate, since architectural restrictions, such as memory bandwidth limitations or limited interconnections between CUDA-cores, can pose significant bottlenecks [8].

A similar approach is applied at Intel for their AI-optimized Stratix NX series, which builds on 14nm FPGA technology. By replacing standard Digital Signal Processing (DSP)-slices in the fabric with 3960 AI-optimized Tensor Cores, a 15× higher Int-8 fixed point compute performance is achieved for matrix-matrix & matrix-vector multiplication, and element-wise operations, yielding a total compute power of 142.6 TOPS at around 1.5 TOPS/W. Alternatively, Intel pursues a second strategy with its Embedded Multi-Die Interconnect Bridge (EMIB) technology, which allows integrating external dies into the FPGA. One implementation that exploits this interface for accelerating deep learning operations are the Intel Stratix 10 FPGAs with TensorTiles [9].

Lastly, growing efforts go into designing custom application-specific integrated circuits for the AI domain. Google recently passed its fourth iteration for its Tensor Processing Unit (TPU), matching the Orin's compute

¹We want to note, that the Orin platform supports sparse processing with a sparsity factor of up to 2. Its effective peak TOPS is therefore only half.

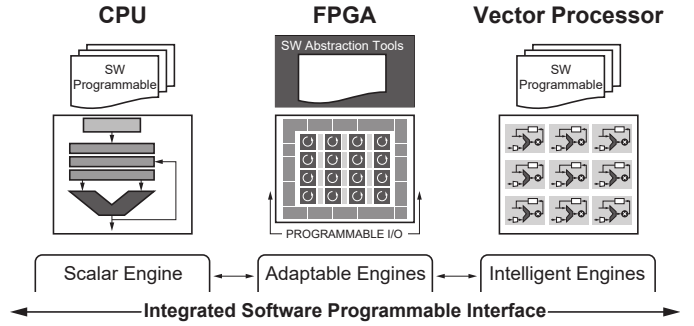


Fig. 1. Overview of different types of programmable compute resources integrated into the Versal's heterogeneous platform [10].

capabilities at a reduced power efficiency of 1.62 TOPS/W. However, the system is primarily not intended for standalone deployment, since low-level communication interfaces allow for clustering multiple of these boards. Other approaches consider waver-scale ASICs that utilize a complete silicon waver as one module without slicing individual chips.

In conclusion, it can be observed, that all modern AI-targeted hardware platforms based on GPU or FPGA include an AISC-like computation unit that is designed for specific AI operations. This is a key enabler for power-efficient and high-performance ML inference. However, none of the above mentioned products or their competitors provide a space-ready variant in terms of radiation-hardness or -tolerance. One exception that follows this approach is the Versal adaptive SoC, which is a space-ready FPGA-based AI-optimized hardware platform. The Versal is introduced in the following section.

III. VERSAL PLATFORM FOR DL INFERENCE

The previous section showed, that none of the prior mentioned technologies can be considered fully optimal, as they all excel in one or more specific characteristic, but do not provide a "one size fits it all" solution. The Versal AI Core series, introduced by AMD-Xilinx in 2019, is an approach to combat the existing problems by unifying multiple technologies in one SoC to extract and combine the best of all technologies. In the following, we will provide a detailed look into this system.

The Versal adaptive SoC is a fully software-programmable heterogeneous compute platform that combines scalar and vector processing elements tightly coupled to next-generation programmable logic, all interconnected with a high-bandwidth network-on-chip (NoC). This concept is shown in Fig. 1. Scalar processing, a domain where CPUs natively excel, is essential for algorithms that exhibit frequently occurring decision-based branches that control complex and nested program flows. Vector processing, as implemented in GPUs or DSPs, is optimal for domain-specific parallelization such as signal processing, e.g., complex math or convolutions. Lastly, adaptable hardware enables to operate on irregular data structures and workloads under tight latency constraints by providing flexible parallel compute and fast local memory, ultimately enabling real-time control. This hybrid architecture is promised to enable a performance increase through the

operation of multiple technologies in concert, outperforming single-technology-only solutions [10].

The major workhorse for AI computation is, similar as for the above technologies, an ASIC-like vector computation unit called AI-Engine Array, which is detailed in the following subsection.

A. AI-Engine Array

The intelligent engines, also called AI-Engines (AI-E) in the context of the Versal AI Core series, are an array of very long instruction word (VLIW) and single instruction, multiple data (SIMD) processing units combined with dedicated data and instruction memory. Each AI-E features dedicated fixed-point and floating-point vector units. The units support various combinations between eight 32-bit x 32-bit floating-point MACs or 128 8 x 8 bit MACs per clock cycle through a full permute unit with 32-bit granularity. Combining the VLIW and SIMD approach, two vector input streams can be concurrently loaded into the vector processing unit via a 256 bit-wide stream, and saved back to memory via a 256 bit-wide store unit, while performing additional functionality such as fix-to-float conversion or non-linear activation in silicon, all using a single 7-way instruction.

One of the strengths of the AI-E array lies in its extremely flexible interconnectivity. Thanks to various communication interfaces within the AI-E array, dataflow can be optimally routed either through a 1-dimensional cascade or a 2-dimensional dataflow. Furthermore, tight integration with the PL (0.96 TB/s) and the rest of the chip through the NoC (100 GB/s) avoids early memory bandwidth bottlenecks, although we show in Section V, that this limit can indeed be reached with certain NNs.

To utilize this array in concert with the PL, AMD-Xilinx provides the IP Deep-Learning Processor Unit, which is explained in the following.

B. Deep-Learning Processor Unit

The Deep-Learning Processor Unit is a configurable computation engine targeted for deep neural networks. AMD-Xilinx offers the DPU for multiple hardware platforms and with different optimization schemes, i.e., targeting throughput, latency, or scalability. Once selected, the DPU is deployed on the PL and the AI-Engines. It hence combines a synthesized code-block on the FPGA with a series of C++ programs deployed on the AI-Engines. Both parts together are denoted as a single IP block. It can be viewed as a microcoded co-processor with an instruction set optimized for Deep Neural Network (DNN) operations. Fig. 2 provides a system-model of the DPU. Various configurations, such as batch-sizes of up to six using Batch Handlers (BH), choosing 32 or 64 AI-Engines per BH, as well as running different NNs concurrently using multiple Compute Units (CU) are offered.

ML computation is split between FPGA and AI-E Array. Compute-intensive operations, such as MLPs, CNNs, and non-linear activations are computed on the AI-Engines, whereas data-handling, such as load and store operations, depth-wise

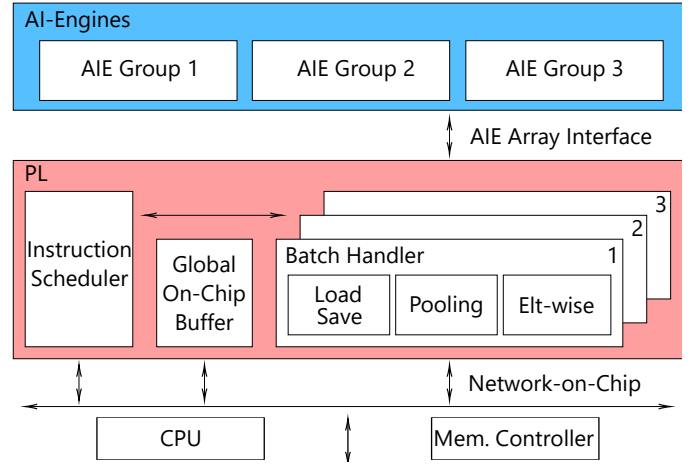


Fig. 2. Block diagram of the DPU architecture (DPUCVDX8G). Adapted from [12].

convolution, pooling, and element-wise operations, are deployed on the PL. Hence, its theoretic maximum compute capability depends mainly on the AI-E array and is computed as [11]

$$\text{DPU Perf.} = f_{\text{AI-E}} \cdot 256 \cdot \#\text{AI-E} \cdot \#\text{BH} \cdot \#\text{CU}. \quad (1)$$

The memory bandwidth available to the DPU depends on its configuration, specifically, on its number of batch handlers. It is calculated as

$$\text{DDR-BW} = (512 + 128 + 32 + 128 \cdot \#\text{BH}) \cdot f_{\text{PL}} \text{ bits/s}, \quad (2)$$

where 512, 128, 32, and 128 denote the bus widths for weight, bias, instruction, and data AXI memory mapped interfaces, respectively [11].

The native support of ML layers is limited to MLP- and CNN-like operations, however, custom operations of any kind can be integrated into the DPU dataflow either in a non-accelerated manner (PS) or on either PL and/or AI-E Array.

The DPU is benchmarked in three hardware configurations in this work, shown in Table I. We both analyze the impact of batch-parallelization, i.e., single-batch vs. multi-batch execution, and the trade-off between hardware resources and compute power, i.e. 32 vs. 64 AI-Engines per BH.

IV. BENCHMARK

This section discusses the benchmark strategy and its results. First, the challenge of selecting a representative set

TABLE I
BENCHMARKED DPU HARDWARE CONFIGURATIONS

Config	Freq. ^a [MHz]	FPGA [%]	AIE [#]	Batch [#]	Int8 Perf [TOPS]	Mem-BW [GB/s]
C32B1	333/1250	6%	32	1	10.24	33.3
C64B1	333/1250	7%	64	1	20.48	33.3
C32B5	333/1250	24%	160	5	51.20	54.6

^aFrequency of PL / AI-Engines

TABLE II
SELECTION OF NN VARIATIONS FOR THE BENCHMARK

Property	Start	Stop	Step size
Multi-Layer Perceptron			
Num. Input	64	512	64
Num. Neurons	64	512	64
Num. Layers	1	9	2
Convolutional Neural Network			
Input Resolution	128x128	512x512	128
Channels	0 ^a	256	64
Kernel-Size	1x1	8x8	2x2
Num. Layers	2	8	2
ResNet50			
Input Resolution	512x512 ^b	2048x2048	512
Channels	3	-	-

^a1 Channel is used as start value.

^bResolution 32x32 is additionally provided.

of NN architectures that represent the current landscape of ML models is discussed. Second, the benchmark process w.r.t. strategy, performance metrics, measurement methods, and GPU-based reference platform is elaborated. Afterwards, the results are presented and a first conclusion is taken from a high-level perspective.

A. Selection of Representative NN Architectures

The goal of this benchmark is to provide transparency for key performance metrics of NN execution on the Versal platform using the DPU. Choosing a suitable set of representative NN architectures that allow to infer predictions for modern ML models, which resemble a combination of various ML operations and layers in a single model, is a challenging task. In this work, we decided to separately focus on single ML operations in their various configurations. This allows for a more fundamental understanding of how performance scales w.r.t. key NN properties, such as num. of layers, input resolution, etc. By covering a sufficiently wide range, this allows for a more general understanding compared to analyzing very specific architectures. However, variants of ResNet50 are also considered to provide a practical reference.

To analyze the DPU's performance while fully utilizing the AI-E array, we select the MLP and "standard" convolutional layer, as discussed in Subsection III-B. Each layer is followed by a ReLU activation function, which is considered state-of-the-art and hardware efficient. Table II summarizes the settings for generating the benchmarked NN architectures using an uniform grid-search-wise setting.

The NN models are transformed to their hardware-accelerated equivalent according to the workflow described in [6].

B. Benchmark Strategy, Performance Metrics, and Methods

The benchmark strategy utilized in this work targets two levels of abstraction. On the system-level we analyze the average model inference time and corresponding power consumption over ten runs. The power consumption is determined by reading the Versal's Core and PL Device Rail 'VCCINT'

power sensor, which measures the effective power of the DPU (PL + AI-E array). These values provide primary information about the scaling of performance and power consumption as a function of NN parameters. On a deeper, architectural level, we concurrently measure the internal DPU execution time (without python overhead), computational efficiency, and occupied memory bandwidth for each model. This second-level analysis, discussed in detail in Section V, enables to understand the origin of certain hardware-related limitations and restrictions in achievable performance, ultimately benefiting the user in designing more hardware-suitable models for optimal performance.

All NN models are analyzed for all DPU configurations given in Table I. To provide reference performance values, the NNs' average inference times are additionally benchmarked on a consumer-grade NVIDIA Geforce RTX 4070TI GPU by utilizing TensorFlow's graph-execution with Just-In-Time (JIT) compilation.

C. System-Level Benchmark Results

Fig. 3 summarizes the system-level benchmark results for a selection (all combinations of minimum and maximum value for all variables) of the NNs analyzed. Multi-Layer Perceptrons, Convolutional Neural Networks, and ResNets are vertically arranged in groups. The first column denotes the NNs' architectural properties, the second column denotes avg. python inference time for Versal and GPU, and the third and forth columns denote avg. DPU runtime and DPU power consumption for the Versal only, respectively. Different hardware setups are denoted by colored bars. Blue, red, and yellow bars denote C32B1, C64B1, and C32B5 DPU configurations, respectively, whereas black bars denote the GPU's performance for single-batch configuration.

Intuitively, the python inference time grows with NN complexity for all models. A significant dynamic range of 0.1 ms - 0.5 ms for MLPs and 0.1 ms - 2.5 secs for CNNs is observed for the DPU in C32B1 configuration, whereas the GPU's overall dynamic range (0.25 ms - 0.4 ms) is much smaller. Extreme execution times in the order of seconds likely denote an internal anomaly, e.g., on-chip memory overflow for too big models. On average, the Versal's overall inference times for CNNs are approximately 2 orders of magnitudes slower than the GPU's, while they are equal for MLPs. Performance of multi-batch inference on the Versal shows an interesting behavior for MLPs and CNNs. By using a batch-size of five (C32B5 DPU configuration) on the Versal, nearly no python inference time increase is observed for any MLP variant, denoting an about 5× higher throughput. This is not the case for all CNNs, as some execute with nearly the same inference time, while others reach > 5× higher times, voiding any possible benefit from processing batches in parallel. The reason for this phenomenon is the rather limited memory bandwidth which prevents data-intensive CNNs to utilize the multi-batch processing capabilities effectively, as discussed in detail in Section V. Moreover, in C64B1 configuration, while the MLP's inference times on average increase by a little,

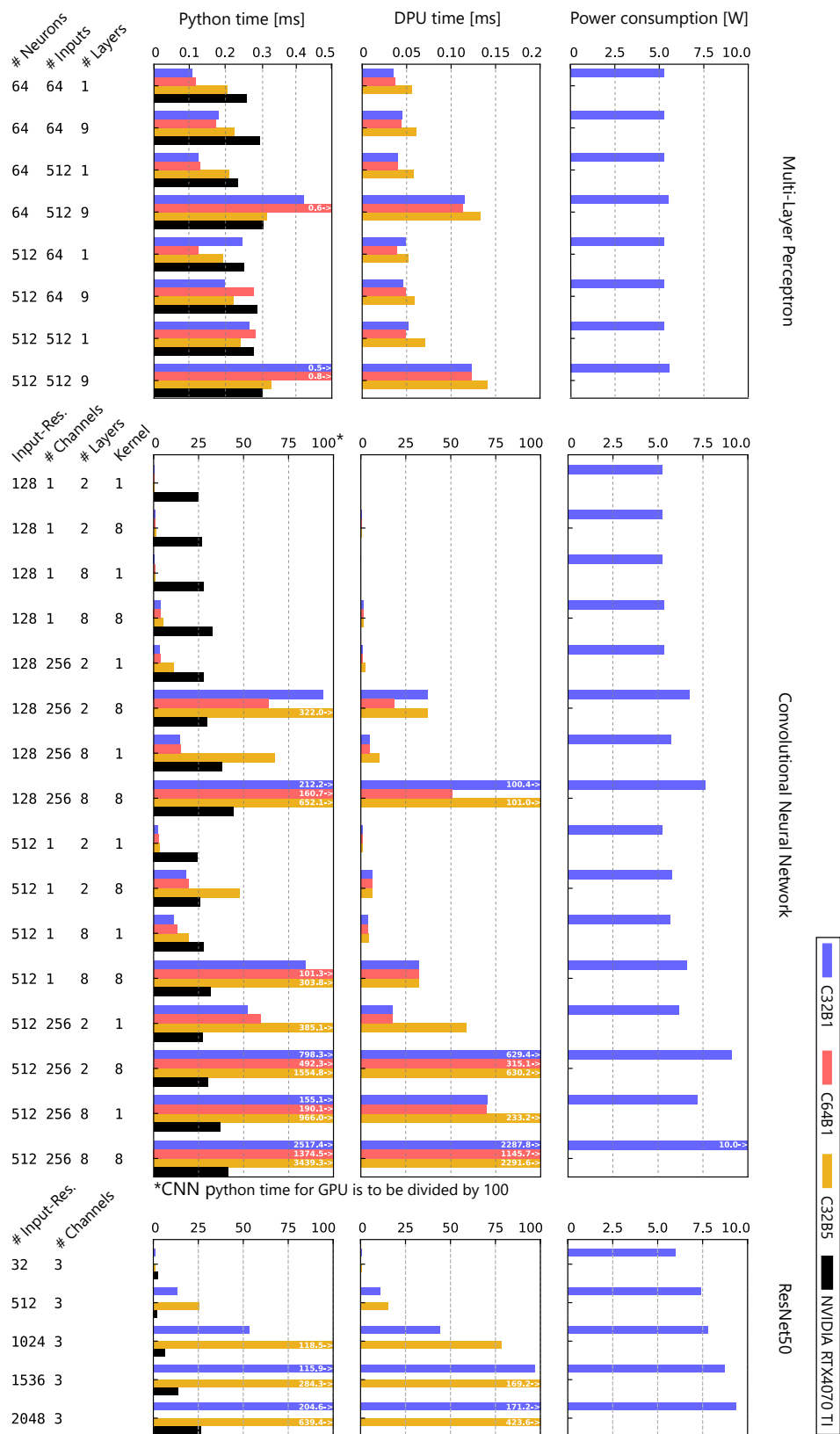


Fig. 3. System-Level benchmark results for MLPs, CNNs, and ResNet50s. Metrics: Python inference time (second column), DPU execution time (third column), power consumption (fourth column). Hardware configurations: DPU C32B1 (blue), C64B1 (red), C32B5 (yellow); GPU (black).

indicating a loss in throughput, the CNN’s inference times on average decrease, which means they can efficiently exploit the additional compute resources.

The DPU run-time time (without python overhead) shows a similar dynamic behavior w.r.t. to model type and complexity, however, in terms of magnitude it is about 25% and up to 70% smaller than the corresponding python inference times for MLPs and CNNs, respectively. This indicates a significant overhead introduced in the Python-DPU interface and hints a major area of improvement. Possible solutions could be to utilize the C++-API or to directly control the DPU via its registers.

Lastly, column four denotes the power consumption of the DPU in C32B1 configuration. The power consumption depends heavily on the NN complexity and is approximately stable at 5.4 W for all MLPs and ranges between 5.2 - 10.0 W for CNNs.

V. ARCHITECTURAL PROFILE

This section provides an in-depth look “under the hood” of the DPU’s performance by profiling it on an architectural level. To illustrate the low-level performance of the DPU on the Versal we adapt the Roofline Performance Model from high-performance computing (HPC) [13]. The core idea is that applications are either computationally bound by the maximal arithmetic performance of the system, or memory-bandwidth bound by the maximal speed of the data-transfer interface. This applies well to NNs, since they can be extremely computationally intensive and have to process large amounts of data. Here, the Roofline model offers insights on possible performance bottlenecks.

Fig. 4 summarizes the profiling results in the Roofline model. The y-axis denotes the performance in Giga operations per second of the DPU, which is measured by using AMD-Xilinx’s profiling tools. The x-axis denotes the NN’s *operational intensity* in terms of DPU operations per byte transfer on the DDR memory interface. This metric is computed as the fraction of the total operations performed for one NN forward pass and the total amount of bytes transferred between DPU and the DDR memory within the inference process. We want to clearly note, that the arithmetic intensity metric used in this work is not independent of the hardware architecture, since the DPU has full control over the weight and data buffering strategy (undisclosed) and, hence, utilization of the DDR memory interface. Therefore, measuring the arithmetic intensity using AMD-Xilinx’s profiling tools is inevitable, since calculating the arithmetic intensity analytically is not possible.

We track this interface, since we identify it as the memory-wise weakest element in the system (up to 100 GB/s, ref. to Subsection III-A). The total bytes measured include everything necessary w.r.t. to NN execution, i.e., loading input data from DDR memory to DPU, loading instructions and weights for each layer, storing and loading the independent outputs (feature maps) to/from DDR memory, and transferring the final output result back from DPU to DDR memory.

Fig. 4(a) denotes the location for every NN variant generated in Subsection IV-A by a scattered point, which is rectangular and circular for MLPs and CNNs, respectively. Visual indicators, such as size, background color, border color, and label (only for CNN), denote the corresponding NN’s architectural properties, i.e., num. of layers, num. of neurons for MLP (num. of channels for CNN), num. of inputs for MLP (input-resolution for CNN), and kernel size (only for CNNs), in this order. Their relations are visually shown in (c). All points are bounded by the maximum memory-bandwidth and the maximum compute capability of the DPU, which are indicated by a blue (grey) line for the DPU configuration C32B1 (C32B5). The following two sub-sections are limited to C32B1, while Sub-section V-C covers C64B1 and C32B5.

A. Performance of Multi-Layer Perceptrons

It can be seen, that the MLPs form a distinct group in the lower-left corner at operational intensities between 1.5-2 OPs/byte. Due to this low operational intensity, all MLPs operate in the memory-bandwidth bound domain. Depending on their specific architectural properties, MLPs are either primarily bandwidth limited (points near the blue line), i.e., they achieve their maximum theoretic performance possible, or are subject to additional bottlenecks (points below the blue line), which are not captured by the Roofline model. The primary property that decides between those two scenarios is the number of neurons per layer (blue background). A high neuron count (~ 500) leads to a performance near the theoretic limit, while a low neuron count (< 400) results in severely degraded performance. The second factor is the number of layers, with very few layers (2-4) performing worse than numerous (6+) layers. The number of inputs does not affect the performance in a noticeable way.

B. Performance of Standard Convolutional Layers

The performance of CNNs behaves in a more dynamic way compared to MLPs. The operational intensity of the CNNs benchmarked in this work stretches over 4 orders of magnitude (~ 1 to $\sim 10^4$ OPs/byte), hence, they operate in both memory-bound and compute-bound domains. In general, it can be observed, that well-performing models (near the blue line) follow the roof of the Roofline model closely, which indicates correct assumption of the architecture’s theoretical limits w.r.t. memory bandwidth and compute performance.

The most important variable is the num. of channels, which primarily determines whether a model will be situated in the bandwidth-bound or compute-bound region. It can be observed, that CNNs with 1 or 64 channels are subject to extremely bad performance and are not executed efficiently w.r.t. to their theoretical maximal performances, i.e., they are far below the blue line. For a higher channel number a computational performance greater than 2 orders of magnitude is observed, shifting the models significantly closer to the system’s memory and computational boundary. Second, the input resolution plays a non-negligible role in the models’ performance. In both domains it can be observed, that a higher

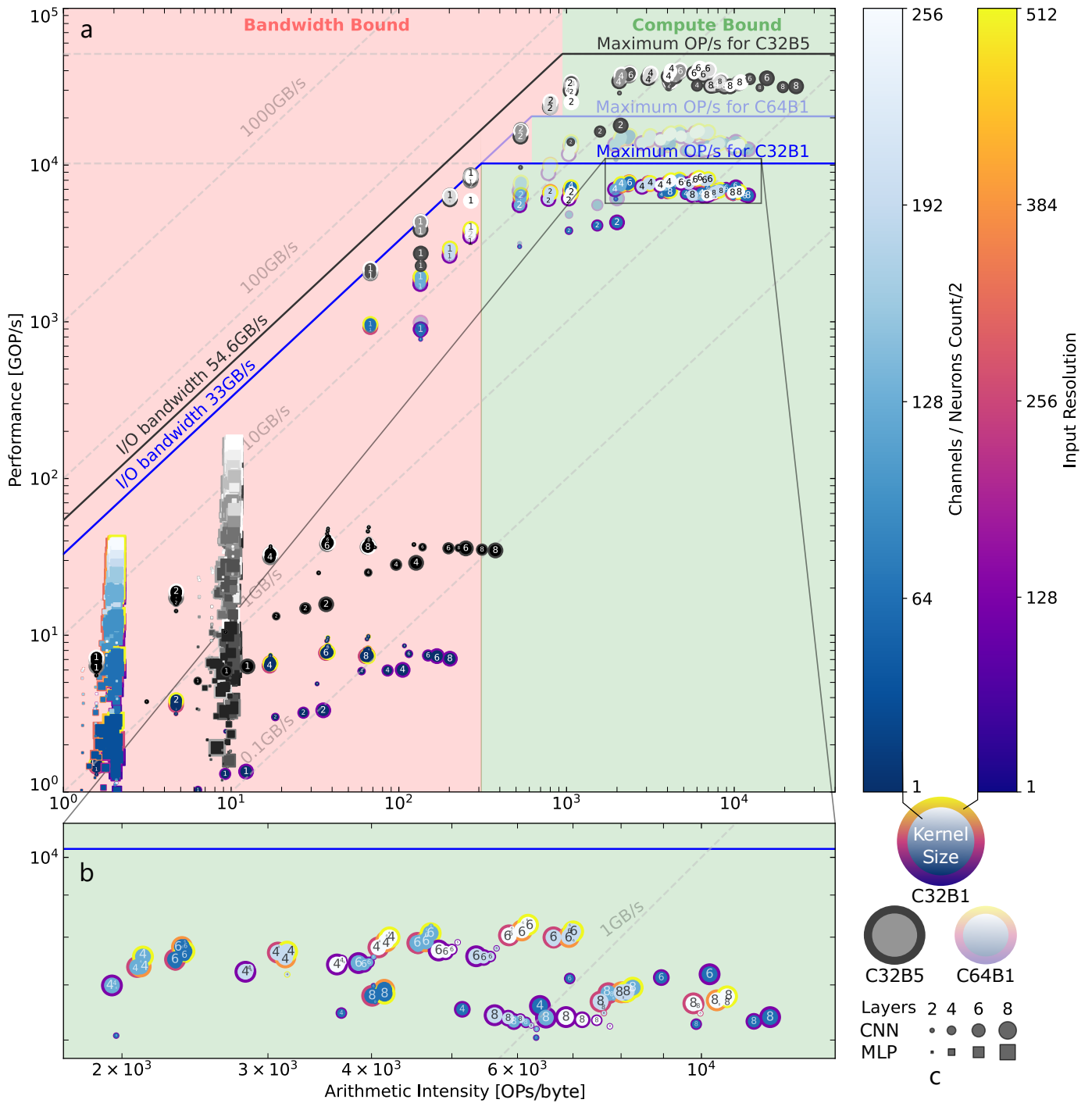


Fig. 4. Roofline model for DPU performance on the Versal, calculated for MLP and CNN NN architectures with various properties. Red area denotes memory-bandwidth bound region, green area denotes computationally bound region. The blue (grey) line denotes the maximum memory-bandwidth and the maximum compute capability for the DPU in configuration C32B1 (C32B5). Scattered rectangles and circles denote MLP and CNN architectures, respectively. Visual indicators for the points, i.e. size, background color, border color, and label (only for CNNs), denote the corresponding NN architecture as explained in (c). (a) sets the overall performance for both DPU architectures in perspective, while (b) provides a zoomed-version of the crowded computationally-saturated region of the Roofline model for the C32B1 DPU architecture, which is populated solely by CNNs.

input resolution generally raises the models' positions in the Roofline model quite significantly, although, a bigger impact is seen within the bandwidth-bound domain. The CNNs' performances are not impacted by the num. of layers.

Fig. 4(b) provides a zoomed-in view on the crowded computationally-saturated region of the Roofline model for the C32B1 DPU architecture. Within this region, all models reach between 6-8 TOPS, utilizing between 60%-80% of the DPU's maximal computational performance (10.24 TOPS). Interestingly, it can be seen, that models who only differ in their num. of layers reach higher arithmetic intensities for a lower number of layers, while the absolute computational performance remains the same. This is somewhat counter intuitive, as one might think, that the memory-wise overhead associated with one forward pass would impact the overall performance less for longer models.

C. Impact of Multi-Batch Inference and Additional AI-Engines

As stated in Subsection III-B, two additional DPU configurations, i.e., one batch handler (BH) with 64 AI-Engines, and five BHs with 32 AI-Engines per BH are analyzed. We want to provide a short summary what changes have been observed for those configurations:

a) *Doubling Compute Power (DPU C64B1)*: For MLPs no difference at all is observed, since all models are memory-bound or suffer additional internal bottlenecks, voiding any possible benefit of additional compute capacity. We refrain from adding them to Fig. 4 for visibility reasons. CNNs show little improved performance for operational intensities $< \sim 500$ OP/s, however, after entering the compute-limited domain, additional compute capacity leads to similar relative maximal performance w.r.t. to the maximal limit (20.48 TOPS) as observed for C32B1 configuration. C64B1 CNNs are denoted by transparent points in Fig. 4.

b) *Multi-Batch Inference (DPU C32B5)*: For all MLPs a shift of the operational intensities by about factor five is observed. This can be intuitively understood by the re-use of weights for the 5 BHs, resulting in more calculations without additional significant load on the DDR interface caused by data. Moreover, the available memory bandwidth is increased by 64% (ref. to Eq. 2), resulting in an overall higher compute performance.

CNN performance follows a different scheme. Since the memory interface is primarily occupied by intermediate data (feature maps) transfers for CNN inference and not for weights/biases, no change in operational intensity is observed, since the additional operations incurred by multi-batch execution require equally more data transfer. This also limits the performance increase for CNN models operating near the bandwidth limit. This is the reason, why certain CNN configurations have a significantly higher inference time, while others show no increase. However, in the compute limited domain, the performance rises up to a factor of five, exploiting up to 80% of the DPU's maximal compute power (51.2 TOPS).

VI. CONCLUSION AND OUTLOOK

In this work we performed an in-depth performance benchmark of the Deep-Learning Processor Unit on the Versal AI Core series platform. We provided a system-level analysis in terms of inference speed and power consumption for various MLP and CNN variants and ResNets, and performed an architectural profiling using AMD-Xilinx's profiling tools to understand the observed behavior better. Future work should consider analysing the so-called "Custom-Layer" feature w.r.t. overall performance, which allows to integrate unsupported operations (e.g., FFT, pre/post-processing, etc.) into the DPU dataflow for efficient execution.

REFERENCES

- [1] 3d Generation Partnership Project, "Release 17," <https://www.3gpp.org/specifications-technologies/releases/release-17>, 2023, [Online; accessed 22-January-2023].
- [2] Fraunhofer Institute for Integrated Circuits IIS, "5g satellite integration," <https://www.iis.fraunhofer.de/en/ff/kom/satkom/sat-5g.html>, 2023, [Online; accessed 22-January-2023].
- [3] M. Ghiglione and V. Serra, "Opportunities and challenges of ai on satellite processing units," in *Proceedings of the 19th ACM International Conference on Computing Frontiers*, ser. CF '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 221–224. [Online]. Available: <https://doi.org/10.1145/3528416.3530985>
- [4] Advanced Micro Devices, Inc., "Xqr versal for space 2.0 applications," <https://www.xilinx.com/content/dam/xilinx/publications/product-briefs/xilinx-xqr-versal-product-brief.pdf>, 2022, [Online; accessed 22-January-2023].
- [5] B. Gaide, D. Gaitonde, C. Ravishankar, and T. Bauer, "Xilinx adaptive compute acceleration platform: Versaltm architecture," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 84–93. [Online]. Available: <https://doi.org/10.1145/3289602.3293906>
- [6] M. Petry, P. Gest, A. Koch, M. Ghiglione, and M. Werner, "Accelerated deep-learning inference on fpgas in the space domain," in *Proceedings of the 20th ACM International Conference on Computing Frontiers*, ser. CF '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 222–228. [Online]. Available: <https://doi.org/10.1145/3587135.3592763>
- [7] M. Adhiwiyogo, R. D'Souza, S. Leibson, and R. Shak, "White paper: Pushing ai boundaries with scalable compute-focused fpgas," pp. 1–6, 2020.
- [8] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, and A. Borchers et al., "In-datacenter performance analysis of a tensor processing unit," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, p. 1–12, jun 2017. [Online]. Available: <https://doi.org/10.1145/3140659.3080246>
- [9] E. Nurvitadhi, J. Cook, A. Mishra, D. Marr, K. Nealis, P. Colangelo, A. Ling, D. Capalija, U. Aydonat, S. Shumarayev, and A. Dasu, "In-package domain-specific asics for intel® stratix® 10 fpgas: A case study of accelerating deep learning using tensortile asic(abstract only)," in *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 287. [Online]. Available: <https://doi.org/10.1145/3174243.3174966>
- [10] AMD-Xilinx, "Versal: The first adaptive compute acceleration platform (acap)(wp505)," <https://docs.xilinx.com/v/u/en-US/wp505-versal-acap>, [Online; accessed 05-July-2023].
- [11] AMD-Xilinx Inc., "Dpuvdx8g for versal acaps product guide (pg389)," 2023, [Online; accessed 21-August-2023].
- [12] AMD Xilinx, "Vitis ai user guide (ug1414)," <https://docs.xilinx.com/r/en-US/ug1414-vitis-ai/Vitis-AI-Overview>, [Online; accessed 12-July-2023].
- [13] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, p. 65–76, apr 2009. [Online]. Available: <https://doi.org/10.1145/1498765.1498785>