# Accelerated Deep-Learning inference on FPGAs in the Space Domain

Invited Paper

Michael Petry*
Patrick Gest*†
michael.petry@airbus.com
patrick.gest@airbus.com
Airbus Defence and Space GmbH
Taufkirchen, Bavaria, Germany

Andreas Koch‡
andreas.c.koch@airbus.com
Airbus Defence and Space GmbH
Taufkirchen, Bavaria, Germany

Max Ghiglione
max.ghiglione@esa.int
European Space Agency
Noordwijk, The Netherlands

Martin Werner
martin.werner@tum.de
Technical University of Munich
Munich, Germany

## ABSTRACT

Artificial intelligence has found its way into space, and similar to the situation on ground demands powerful hardware to unfold its full potential. With the heterogeneous compute platform that is offered by the space-grade variant of the Versal, AMD Xilinx presents a system that is particularly targeted at accelerating AI inference in space. This paper investigates the design flow and the achievable performance of this novel device. We present benchmark results in terms of concrete figures and measurements, i.e., throughput, latency, and power consumption, achieved by a predesigned hardware accelerator realized on the system, and compare them to a previous generation platform.

## CCS CONCEPTS

• **Hardware** → **Emerging architectures**; • **Computing methodologies** → *Artificial intelligence.*

## KEYWORDS

FPGA, hardware accelerator, neural network, machine learning, Xilinx Versal

**ACM Reference Format:**
Michael Petry, Patrick Gest, Andreas Koch, Max Ghiglione, and Martin Werner. 2023. Accelerated Deep-Learning inference on FPGAs in the Space Domain: Invited Paper. In *20th ACM International Conference on Computing Frontiers (CF '23), May 9–11, 2023, Bologna, Italy.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3587135.3592763

*Both authors contributed equally to this research.
†Also with Technical University of Munich.
‡Also with Technical University of Munich.


*CF '23, May 9–11, 2023, Bologna, Italy*

ACM ISBN 979-8-4007-0140-5/23/05.
https://doi.org/10.1145/3587135.3592763

## 1 INTRODUCTION

Future telecommunications satellites are envisioned to seamlessly integrate into the mobile communication networks of the next generation, e.g., with the raise of regenerative payloads [26]. First signs toward 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 [17]. A general goal in these next generation communication networks is to employ radio terminals that are more intelligent, automatically adapting their transmission to the current state of the spectral environment, in order to make better use of the scarce frequency resources [20]. For realizing such highly adaptive radios, research mainly focuses on the use of algorithms in the Artificial Intelligence (AI) domain, in particular neural networks (NNs) [14]. As a result of the success that these methods showcase in the area of communications and beyond, 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 [19]. With the Versal Adaptive Compute Acceleration Platform (ACAP) in the XQR variant, AMD Xilinx (in the following referred to as Xilinx) now offers a chip in a space-grade package that is particularly targeted for machine learning applications in space, while promising more compute power than traditional FPGA-based systems-on-chip (SoCs) [7, 18]. Combining an FPGA fabric with a new class of compute engines intended for parallel computing, including AI inference acceleration, this device opens up new opportunities in terms of on-board processing performance, but also comes with novel challenges in terms of system integration and application development for satellite manufacturers. An initial evaluation of the design flow

and the capabilities of the platform is thus in the center of this paper.

This paper is structured as follows: Section 2 starts by providing background information on contemporary deployment techniques of NNs on FPGAs, followed by an introduction of both considered FPGA hardware platforms in section 3. The main contribution of this paper is given in section 4, which describes the deployment workflow including the utilized tool-chain in detail. Special emphasis is put on the steps required from transitioning a CPU/GPU-based deep-learning model to a FPGA-executable application, especially focusing on the quantization step and the computations involved in quantized convolutional NNs. It concludes with a description of the architecture of the generated Deep-Learning Processing Units (DPU) and the development of a control application supervising the NN inference. In Section 5, the selected deep-learning application for the benchmark is briefly summarized, and the results of benchmarking the generic inference accelerators, implemented on the Xilinx Versal and Zynq UltraScale+, are presented and compared. Lastly, a conclusion and outlook of future work is given in section 6.

## 2 BACKGROUND

Although the implementation of NNs on FPGA-based systems is still an active research topic, common implementation strategies can essentially be divided into three classes. The first and most complex one is the development of a custom hardware accelerator for a given type and architecture of a NN, either written in a hardware description language (HDL) or given as high-level synthesis (HLS) code [16]. Using an automatic framework like FINN [25], hls4ml [15], or MATLAB's HDL Coder [24] for generating a HLS/HDL description of a custom hardware accelerator for a given NN is a second option. Third and finally, it is possible to deploy a generic hardware accelerator for AI inference in form of an intellectual property (IP) core, which has the disadvantage of not being tailored toward a specific network architecture. However, due its generic and programmable architecture, such an IP core offers the possibility to execute a variety of NNs during operation. While the second approach seems to be promising in terms of obtaining a custom accelerator without the high development effort associated with the first approach, it is currently not an option for the Versal platform. The reason for this is that the Versal is a heterogeneous compute platform in which the FPGA fabric is only one class of hardware resources available for accelerating AI inference computations. At the time of this writing, none of the aforementioned frameworks supports a partitioning of NN computations into functional entities that make use of all of the system's capabilities. What is available, however, is a generic hardware accelerator provided by the manufacturer of the chip itself, which does incorporate the full set of compute functionalities.

## 3 FPGA PLATFORM DESCRIPTION

The first part of this section gives a short overview of the two hardware platforms which are evaluated in this paper. A larger emphasis is put on the Versal ACAP and its novel compute engines, which is in the main focus of this work. The description of the Zynq UltraScale+ SoC is kept relatively short as it is used mostly as a performance reference in this work.

### 3.1 Xilinx Versal

The Versal presents a powerful hardware platform, targeting a range from very demanding embedded tasks up to networking, communications and data center applications [6]. Its heterogeneous architecture, which Xilinx refers to as an adaptive compute acceleration platform (ACAP), extends the combination of processing system (PS) and programmable logic (PL) with a third class of compute engines. A coarse overview of the Versal's computing resources is shown in Fig. 1. These novel engines are particularly designed for accelerating signal processing and AI inference computations, and thus are known as AI Engines in the context of the Versal [28]. The following paragraph gives a short description of their architecture.

*AI Engines.* In the variant that ships with the Versal AI Core Series VCK190 Evaluation Kit, the system comes with 400 AI Engines, which are arranged in a two-dimensional grid [11]. The architecture of one AI Engine is shown in Fig. 2. Each AI Engine is a software programmable very long instruction word (VLIW) single instruction multiple data (SIMD) processor that excels in vector computations [5]. Its VLIW capability allows an engine to simultaneously execute a scalar operation, two vector move operations, two vector load operations, one vector computation, and one vector store operation during one clock cycle [5]. The read and write operations are executed by the two vector load units and the single vector store unit. A program memory of 16KB with an associated instruction fetch and decode unit completes the design [5]. One example for an operation the SIMD vector unit is able to execute in a single clock cycle is an element-wise multiplication of two 32-bit integer vectors of size eight. This form of data-level parallelism alone would give rise to a speed-up by a factor of eight compared to scalar execution. Multi-core processing, as the highest level of parallelism, is achieved by the fact that all AI Engines can theoretically operate in parallel. Efficient data access and intercommunication of the AI Engines required for this high utilization gives rise to an accompanied 32KB data memory each that can be grid-wise accessed from all four sides [4].

Besides the AI engines, another architectural feature is the use of a programmable Network-on-Chip (NoC) connecting the various computing resources to the memory interface, I/Os, etc. as shown in Fig. 1.

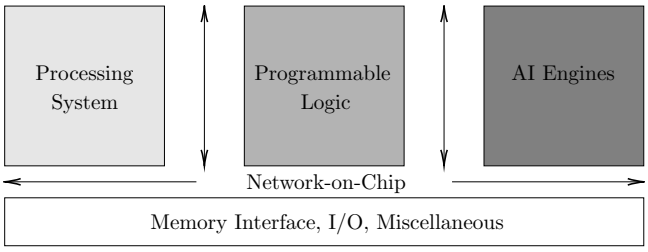


**Figure 1: Versal computing resources. Adapted from [6].**

## 3.2 Xilinx Zynq UltraScale+

The Zynq UltraScale+ is a SoC that combines a multi-core processing system with programmable logic, which is intended for use in a broad range of embedded applications [8]. Its computing power comes from offloading compute intensive tasks to processing blocks on the FPGA fabric. These could be graphics and video pipelines or, in the frame of this paper, AI inference computations. The variant of the chip used in this paper comes with a quad-core Arm Cortex-A53 application processing unit and a dual-core CortexR5F real-time processor forming its processing system [3]. Besides the configurable logic blocks based on look-up tables (LUTs) and flip-flops, the programmable logic part of the chip comprises block RAM for data storage and digital signal processor (DSP) slices for accelerating signal processing tasks [29]. Each DSP splice has a 27-bit x 18-bit binary multiplier with a 48-bit accumulator and a single-instruction-multipledata (SIMD) arithmetic unit with two 48-bit inputs that can simultaneously execute two 24- or four 12-bit additions or subtractions [29]. These DSP slices are extensively used in Xilinx's inference accelerator, as discussed in subsection 4.2.1. The various components of the processing system are connected to the programmable logic via an Arm AMBA AXI4 interconnect [29]. Note that in comparison to the Versal, the UltraScale+ only comes in a defence-grade variant, not a space-grade one, which generally restricts its usage on telecommunications satellites [9].

Table 1 provides a comparison of the two hardware platforms.

# 4 DEPLOYMENT WORKFLOW

This section provides a description of the key steps in the development flow for deploying an AI application on a hardware accelerated platform. First, an outline of the necessary steps in the creation of an inference application suitable for bridging the gap between the NN models and the hardware accelerators is given. Following that, a detailed description of the Vitis development flow for generating a hardware-accelerated Deep-Learning Processing Unit (DPU) is provided. This is followed by the architectural explanation of the generated DPUs. Lastly, the development of a control application supervising the inference application is summarized, including the utilized benchmarking procedure.

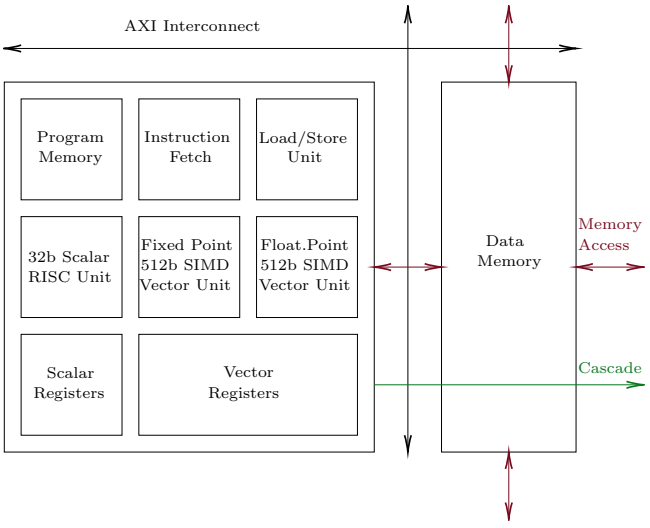


**Figure 2: Versal AI Engine Architecture. Adapted from [5]**

**Table 1: Key Figures of the Hardware Platforms**

| | Zynq US+ ZU9EG [29] | Versal AI Core VC1902 [6] |
|---|---|---|
| Processors | Cortex-A53 (Quad-Core) Cortex-R5F (Dual-Core) | Cortex-A72 (Dual-Core) Cortex-R5F (Dual-Core) |
| Flip-Flops | 548,000 | 1,800,000 |
| LUTs | 274,000 | 900,000 |
| Block RAM | 32 Mb | 34 Mb |
| UltraRAM | - | 130 Mb |
| DSP Slices | 2,500 | 1,968 |
| AI Engines | - | 400 |
| Interconnect | AMBA AXI4 | Network-on-Chip |

## 4.1 Inference Application Development

All of the tasks related to building an AI application, which is intended to be directly run on an embedded hardware platform, can be carried out in Vitis AI, Xilinx's development environment for AI inference [12]. The environment contains three major components that are required for creating a NN and bringing it to a format suitable for execution on a programmable device equipped with an inference accelerator. These are a deep learning framework for training and evaluating the NN, a quantizer that transforms the floating-point operations involved in NNs to fixed-point operations, and a compiler that translates the resulting operations into instructions understood by the generic inference accelerators from Xilinx.

*4.1.1 HW-Supported Neural Network Implementations.* The inference accelerators provided by Xilinx only support a subset of the most commonly used operations in (convolutional) NNs. Therefore, it is strictly necessary to evaluate the compatibility of a given CNN before starting with its implementation. If certain layers are not supported by the accelerator, it is generally still possible to execute the network on the hardware system. However, instead of an accelerated execution on the co-processor, the unsupported operations have to be executed on the embedded CPU cores. This requires intermediate data values to be moved back and forth between the accelerator and the CPU, and leads to a noticeable performance penalty due to the data transfer and the reduced processing power of the CPU as analyzed in [13].

*4.1.2 Quantization and Compilation of Neural Networks.* Following a standard training process in TensorFlow, the next step is to quantize the network's parameters from floating-point to fixed-point values. For purely FPGA-based systems, a quantization is very beneficial due to the much lower hardware complexity for realizing fixed-point units compared to floating-point units. Using fixed-point computations yields an implementation that consumes less power and has lower latency [21]. Even though the AI Engines

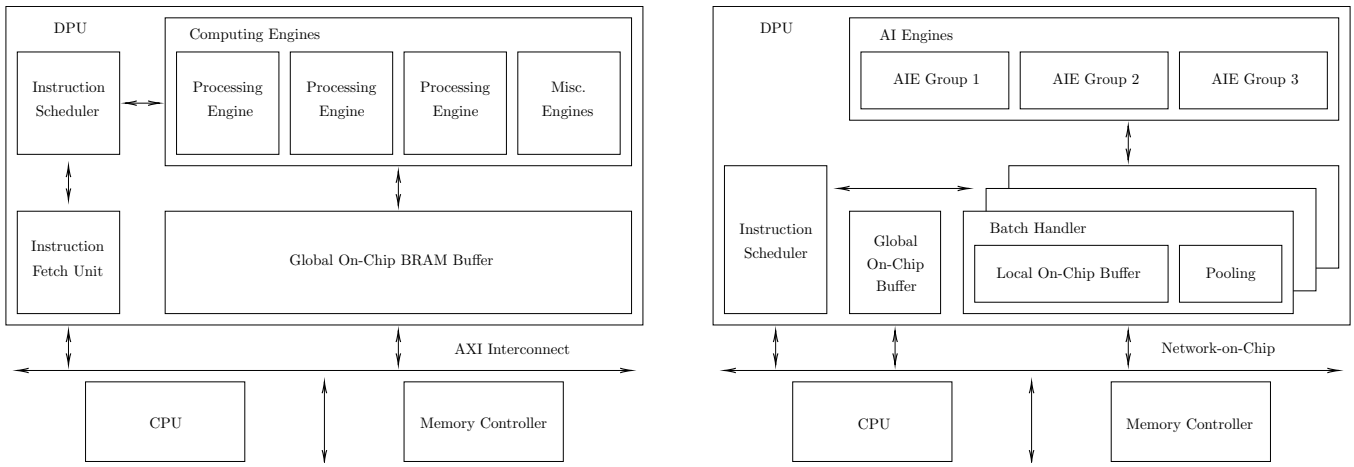


**Figure 3: Deep-Learning Processing Unit (DPU) for Zynq UltraScale+ (left) and Xilinx Versal (right). Adapted from [10] and [27], respectively.**

also support floating-point based vector operations, a fixed-point implementation is still more efficient on the Versal due to the particular high integer compute performance of the AI Engines [2]. Furthermore, quantized values with a reduced bit width require less memory space, which is important for storing parameters and intermediate results directly in the on-chip memory in order to avoid data transfers to the external memory [27]. Due to the lack of floating-point operations in Xilinx's generic inference accelerator, an 8-bit fixed-point quantization of inputs, weights, biases, and activations is required. Besides that, the Vitis AI quantizer implements a power-of-two quantization scheme, a special form of quantization that is particularly well-suited for a hardware efficient realization on a programmable FPGA [22].

*4.1.3 Compilation.* Having obtained a quantized NN, the final step in the inference application development flow is to invoke the Vitis AI Compiler, which compiles the quantized model description into instructions for the generic accelerators [12]. The compiler outputs a file comprising the instructions and quantized parameters, that can be deployed on the embedded hardware platform. An embedded program running on the CPU cores loads this file from memory and prepares the co-processor for the inference task by transferring weights and instructions to the accelerator.

## 4.2 Deep-Learning Processing Unit

The Vitis tool-flow generates a so-called Deep-Learning Processing Unit, which from a high-level perspective can be described as a microcoded co-processor with an instruction set optimized for NN inference [10]. The set of instructions that the DPU needs in order to perform inference for a specific NN is generated by the Vitis AI Compiler for a given NN description, as mentioned in the prior subsection. For the various hardware platforms that generally provide different classes and amounts of compute resources, such as the Zynq UltraScale+ and the Versal, Xilinx offers individually designed accelerator architectures. Both offer a certain amount of configurability. These configuration options determine the underlying level of parallelism in the computations of the DPU, and allow trading off more compute power with a higher usage of hardware resources. Both architectures are briefly analyzed in the following two subsections.

*4.2.1 Zynq UltraScale+.* The architecture of the generic inference accelerator for the Zynq UltraScale+ is illustrated on the left-hand side of Figure 3. It generally consists of an instruction fetch unit, an instruction scheduler, an array of computing engines and an on-chip BRAM buffer [10]. Fully implemented on the programmable logic portion of the chip, the DPU is attached to the AXI interconnect for communication with the embedded CPU and external memory. After receiving a start signal from the CPU at one of its low-level control registers, the accelerator fetches instructions from the off-chip memory [10]. Based on the given instructions, it loads the NN parameters and input data into the on-chip BRAM buffer. The instruction scheduler then delegates various compute tasks to to the different processing engines, which operate on the data in the buffer. As indicated by the resource usage numbers, these processing engines heavily rely on the DSP slices and their underlying multipliers, adders and accumulators for their operation [10]. Intermediate feature values produced by the computing engines are written back to the on-chip buffer. The final inference results are transferred back to and stored in the off-chip memory, which concludes the operation.

In this paper, only a single configuration of the DPU is considered. This configuration, which in the documentations is referred to as the "B4096" configuration [10], tries to make as much use of parallelism as possible and promises the highest performance, while also consuming the most amount of resources among all configurations. Besides the configuration options for a single DPU, it is theoretically possible to implement multiple instances of the DPU in parallel. This work, however, only considers the use of a single instance of each inference accelerators.

*4.2.2 Xilinx Versal.* Looking at the architecture of the inference accelerator for the Versal, which is illustrated on the right-hand side of Figure 3, one can see that the general concept behind it is very

similar to the DPU for the Zynq UltraScale+. The main difference is that the heavy computations involved in the convolutional layers are delegated to the AI Engines instead of executing them in the programmable logic [27]. Less demanding pooling layers on the other hand are still implemented in the PL portion of the system. A second difference is that the Versal DPU provides one global on-chip memory buffer for shared weights and multiple local memories integrated into batch handlers, that can operate on different data samples in parallel [27]. To achieve this parallelism, each batch handler has a private group of AI Engines [27]. The operation of the Versal DPU is otherwise similar to the UltraScale+ DPU. Instructions, parameters and inputs are loaded into the respective on-chip memory portions of the accelerator, the scheduler initiates a data transfer to the AI Engines for processing, and processed inputs are sent back to the memory. This work considers multiple configurations of the Versal DPU, but only ones that rely on a single batch handler. This is due to unresolved problems when using multiple batch handlers, which is potentially due to an error in the own implementation. The main parameter that is varied in the configurations that are considered in this work, is the number of AI Engines used for processing the data. Valid values include either 32 or 64 AI Engines, which correspond to the two configurations that in the documentation are referred to as "C32B1" and "C64B1", and represent a DPU with one batch handler and 32 or 64 AIEs, respectively [27].

### 4.3 Development of Control Applications

After having obtained a running version of the generic inference accelerator, control programs running on top of the embedded Linux system for the communication with the accelerator have to be written. A majority of the complexity associated with this task is abstracted away by the availability of a high-level application programming interface (API) for C++ and Python [12]. In this work, the Python interface was used, yielding a fairly compact script that roughly consists of the following elements:

- Instantiate a so-called runner that encapsulates the generic inference accelerator.
- Feed the runner with the resulting instructions for the accelerator from the compiled neural network.
- Allocate an input and output buffer in main memory for the data samples.
- Fill the input buffer with data.
- Give the accelerator a start signal to begin execution.
- Wait for the accelerator to finish execution.

*Frequency Scaling.* Lastly, the issue of power consumption in the context of satellite systems is to be targeted. As the Versal is known to be more power hungry than its predecessors, an important task of this work was to get quantitative measurements of power values and investigate whether these could be reduced without sacrificing too much performance. One possible approach that is considered in this paper, is the reduction of the operating frequencies of the chip. In case of the Versal, there are two critical frequency domains: The frequency with which the programmable logic is driven, and the operating frequency of the AI Engines. By default, the generic inference accelerator runs at the maximum recommended frequencies, which are 333 MHz for the PL and 1.25 GHz for the AIEs [27]. Through a simple change in the configuration of the accelerator, these frequencies can theoretically be arbitrarily lowered. Following a recommendation about using a multiple between two and four for the ratio between the two frequencies in order to achieve good performance, two further frequency combinations were tried out in this work, that are depicted in the first column of Tab. 2.

## 5 BENCHMARKING RESULTS

In this section, the benchmarking results of the generic inference accelerators are presented. First, the selected deep-learning application is briefly summarized. Afterwards, various hardware configurations of the accelerators implemented on the Versal and Zynq UltraScale+ are presented. Finally, benchmark performance of the application is evaluated.

### 5.1 Deep-Learning Application

The application considered solves an important task from the signal analysis and spectrum awareness domain by determining which frequency bands are currently occupied and which are not, based on an input stream of sampled I/Q data. With the means of convolutional neural networks (CNNs), precomputed spectrogram input images are fed into the neural network and transformed to a binary 2D image in the same dimensions that denotes presence or absense of a signal for each frequency-time combination along both axis. The architecture utilized is the [23].

### 5.2 Hardware utilization

For benchmarking both hardware platforms, one hardware accelerator configuration for the Zynq UltraScale+ and two for the Xilinx Versal were selected, as presented in table 2. In addition, various clock speeds of the programmable logic and, in case of the Versal, AI-Engines were analyzed.

The implementation U-Net makes use of around 500,000 trainable parameters. A forward pass through the quantized network requires more than 3 billion 8-bit Multiply-Accumulate (MAC)-operations. With 512 by 512 wide input images, we believe that even though this model might not seem that large compared to typical CNNs in computer vision applications, however, in the signal processing domain such modest sized input data is typical, and thus the benchmark results are nonetheless interesting. The benchmark results are summarized in Tab. 3 and will be explained in detail in the following two subsections.

### 5.3 Zynq UltraScale+

As shown in the first row of Table 3, the Zynq UltraScale+ DPU in its B4096 configuration operating at a PL frequency of 325 MHz achieves a throughput of 50 frames per second. In terms of latency, 19.38 ms was measured. In the idle state, the device consumes around 3.9 W of power. Once the inference computation starts and the DPU is active, the power consumption jumps to a value of 6.1 W. This marks a change of 2.2 W compared to the idle state.

### 5.4 Xilinx Versal

In the configuration with 32 AI Engines operating at the maximum recommended frequency of 333 MHz for the PL and 1250 MHz for the AIEs, the Versal DPU achieves a throughput of around

**Table 2: Hardware Utilization Configurations of the Generic Inference Accelerator on Zynq UltraScale+ and Versal**

| Configuration | Zynq UltraScale+ B4096 | Xilinx Versal C32B1 | Xilinx Versal C64B1 |
|---|---|---|---|
| Flip-Flops | 98,000 / 548,000 (18%) | 110,000 / 1.8M (6%) | 132,000 / 1.8M (7%) |
| LUTs | 52,000 / 274,000 (19%) | 81,000 / 900,000 (9%) | 92,000 / 900,000 (10%) |
| Block RAM | 9 Mb / 32 Mb (28%) | 0 Mb / 32 Mb (0%) | 0 Mb / 34 Mb (0%) |
| DSP Slices | 710 / 2,520 (28%) | 139 / 1,968 (7%) | 139 / 1,968 (7%) |
| UltraRAM | - | 57 Mb / 130 Mb (44%) | 57 Mb / 130 Mb (44%) |
| AI Engines | - | 32 / 400 (8%) | 64 / 400 (16%) |

79 frames per second, which roughly corresponds to a factor of 1.5 compared to the UltraScale+. Consequently, the latency for performing inference on a single data sample is reduced to 12.34 ms. However, as depicted in Tab. 3, the Versal consumes a tremendous amount of power for achieving this performance gain. In the idle state, the system already consumes 17.1 W of power, which is increased to 19.6 W during the computations, a change of 2.5 W. Comparing idle states, the Versal thus consumes more than 4 times the power of the UltraScale+. Equipping the DPU with more AI Engines increases the power consumption even more, as might be expected. In this configuration the device consumes 19.3 W in the idle state, a plus of 2.2 W compared to the configuration with half the AI Engines. As indicated by the last row in Table 3, doubling the amount of AI Engines for this specific application results only in slightly increased performance, but not to an extent that would justify the increase in power consumption. This might potentially be a consequence of the higher complexity of the resulting DPU implementation for handling more AI-Engines.

Due to this high power consumption, it is particularly interesting to investigate how a change in the operation frequencies affects the system. As shown in the third row of Table 3, a reduction of the operating frequencies from 333 and 1250 MHz to 200 and 800 MHz for the PL and AIE reduces the throughput from 79 to 55 frames per second. The gain is a decrease in the power consumption during the computation time from 19.6 W to 17 W, a notable change of 2.6 W. Further decreasing the AIE frequency to 400 MHz goes along with a reduction of the power consumption in the active state to 15.9 W. Operating at these frequencies, the Versal produces a similar throughput as the Zynq UltraScale+ operating at the highest frequency, but still consumes more than 2.5 times the power.

In the case of dynamic power consumption, the perspective changes. Considering only the change of power from idle to active state vs. the throughput, the Versal does show significant gains compared to the UltraScale+, as depicted in Fig. 4. Using the metric of required watts per frame per second, the Versal interestingly shows its most efficient configuration in the mid-frequency range, with 27.3 mW/fps dynamic power. Contrary to that, the UltraScale+ requires a dynamic power of 44 mW/fps. Moreover, considering this metric, the Versal performs better in all its configurations than the UltraScale+.

Lastly, we want to note that for all hardware configurations throughput and latency are reciprocal. This is due to utilizing only a since batch handler. Contrary to the US+, the Versal does support multiple batch handlers, which can be imagined as an inherent parallelized processing of the input by realizing multiple copies of the inference network [27]. Using this technique, the throughput can be $m$-fold multiplied by utilizing $m$ batch handlers (neglecting performance degradation), while the same latency would be maintained. Impact to the power consumption in terms of total throughput per total power consumption is subject of further investigation.

Summarizing, the Versal offers a decent increase in performance over the UltraScale+ for the wideband signal localization application. Nevertheless, it requires substantially more power. However, when considering the idle power consumption as inherent and only consider the change in power when active, the metric $P_\Delta$/Throughput, which denotes the Watts per FPS needed, shows, that the Versal is slightly ahead of the Zynq UltraScale+ in terms of dynamic power.

**Table 3: Benchmark Results for the Wideband Signal Localization Application**

| Config. | Frequency [MHz] | TP [FPS] | Latency [ms] | $P_{\text{idle}}$ [W] | $P_{\text{active}}$ [W] | $P_\Delta$ [W] |
|---|---|---|---|---|---|---|
| B4096 | 325 | 50 | 19.38 | 3.9 | 6.1 | 2.2 |
| C32B1 | 333/1250 | 79 | 12.34 | 17.1 | 19.6 | 2.5 |
| C32B1 | 200/800 | 55 | 18.05 | 15.5 | 17.0 | 1.5 |
| C32B1 | 200/400 | 37 | 26.75 | 14.6 | 15.9 | 1.3 |
| C64B1 | 333/1250 | 81 | 12.10 | 19.3 | 22.3 | 3.0 |

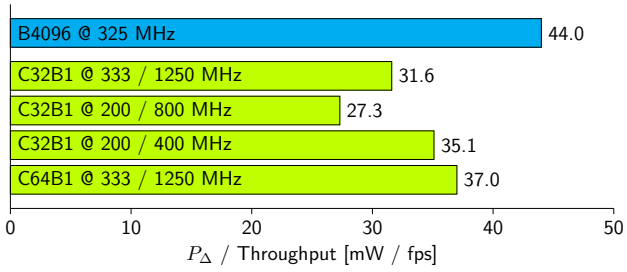


**Figure 4: Efficiency comparison of dynamic power vs. throughput.**

## 6 CONCLUSION AND OUTLOOK

This paper covered the workflow of accelerating AI inference applications on the new Versal ACAP. The Versal is a particularly interesting hardware platform in the context of telecommunications satellites, as it is currently one of the most powerful commercial off-the-shelf devices that comes in a space-grade variant. With the AI Engines as a novel class of compute engines, the chip promises to provide the compute performance that is required for the next-generation telecommunications infrastructure that increasingly relies on artificial intelligence. In a deep learning application selected from the radio frequency domain, the Versal outperformed the UltraScale+ performance-wise by 50%. However, this performance gain comes at the cost of a drastically increased power consumption. This work considered the generic implementation approach that is currently available for the acceleration of inference applications on the Versal.

This paper opens further research questions in two separate directions. The primary objective is to complete the implementation of the custom inference accelerator and evaluate potential performance gains compared to the generic accelerators. Within this, the design space in terms of mapping and implementing different compute functions on the different classes of hardware resources on the Versal could be further explored. Second, a further investigation is to explore potential ways to gain a reduction in the power consumption of the chip. In a space-related environment, this is a key factor not only for power consumption but also thermal dissipation. As the power demand of the Versal is factors above the UltraScale+ and the available power budget of telecommunications satellites is limited, this could otherwise be a real restriction for the use of the Versal on such systems.

## REFERENCES


[1] 3d Generation Partnership Project. 2023. Release 17. https://www.3gpp.org/specifications-technologies/releases/release-17. [Online; accessed 22-January-2023].

[2] A. Gupta. 2020. Architecture apocalypse dream architecture for deep learning inference and compute - versal ai core. https://www.xilinx.com/content/dam/xilinx/support/documents/white_papers/EW2020-Deep-Learning-Inference-AICore.pdf. [Online; accessed 28-January-2023].

[3] Advanced Micro Devices, Inc. 2019. ZCU102 Evaluation Board User Guide. https://docs.xilinx.com/v/u/en-US/ug1182-zcu102-eval-bd. [Online; accessed 25-January-2023].

[4] Advanced Micro Devices, Inc. 2022. AI Engine Kernel and Graph Programming Guide. https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2022_2/ug1079-ai-engine-kernel-coding.pdf. [Online; accessed 25-January-2023].

[5] Advanced Micro Devices, Inc. 2022. AI Engines and Their Applications. https://docs.xilinx.com/v/u/en-US/wp506-ai-engine. [Online; accessed 25-January-2023].

[6] Advanced Micro Devices, Inc. 2022. Versal Architecture and Product Data Sheet: Overview. https://docs.xilinx.com/v/u/en-US/ds950-versal-overview. [Online; accessed 25-January-2023].

[7] Advanced Micro Devices, Inc. 2022. XQR Versal for Space 2.0 Applications. https://www.xilinx.com/content/dam/xilinx/publications/product-briefs/xilinx-xqr-versal-product-brief.pdf. [Online; accessed 22-January-2023].

[8] Advanced Micro Devices, Inc. 2022. Zynq UltraScale+ MPSoC Product Brief. https://www.xilinx.com/content/dam/xilinx/support/documents/product-briefs/zynq-ultrascale-plus-product-brief.pdf. [Online; accessed 25-January-2023].

[9] Advanced Micro Devices, Inc. 2023. Defense-Grade Zynq UltraScale+ MPSoCs. https://www.xilinx.com/products/silicon-devices/soc/xq-zynq-ultrascale-mpsoc.html. [Online; accessed 25-January-2023].

[10] Advanced Micro Devices, Inc. 2023. Dpuczdx8g for zynq ultrascale+ mpsocs. https://docs.xilinx.com/r/en-US/pg338-dpu. [Online; accessed 29-January-2023].

[11] Advanced Micro Devices, Inc. 2023. Versal AI Core Series VCK190 Evaluation Kit. https://www.xilinx.com/products/boards-and-kits/vck190.html. [Online; accessed 25-January-2023].

[12] Advanced Micro Devices, Inc. 2023. Vitis ai user guide. https://docs.xilinx.com/r/en-US/ug1414-vitis-ai. [Online; accessed 29-January-2023].

[13] Ahmad Al-Zoubi, Gianluca Martino, Fin H. Bahnsen, Jun Zhu, Holger Schlarb, and Goerschwin Fey. 2022. CNN Implementation and Analysis on Xilinx Versal ACAP at European XFEL. In *2022 IEEE 35th International System-on-Chip Conference (SOCC)*. 1–6. https://doi.org/10.1109/SOCC56010.2022.9908101

[14] Daniel Chew and A. Brinton Cooper. 2020. Spectrum Sensing in Interference and Noise Using Deep Learning. In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*. 1–6. https://doi.org/10.1109/CISS48834.2020.1570617443

[15] J. Duarte, S. Han, P. Harris, S. Jindariani, E. Kreinar, B. Kreis, J. Ngadiuba, M. Pierini, R. Rivera, N. Tran, and Z. Wu. 2018. Fast inference of deep neural networks in FPGAs for particle physics. *Journal of Instrumentation* 13, 07 (Jul 2018), P07027–P07027. https://doi.org/10.1088/1748-0221/13/07/p07027

[16] Clément Farabet, Berin Martini, Polina Akselrod, Selçuk Talay, Yann LeCun, and Eugenio Culurciello. 2010. Hardware accelerated convolutional neural networks for synthetic vision systems. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 257–260. https://doi.org/10.1109/ISCAS.2010.5537908

[17] Fraunhofer Institute for Integrated Circuits IIS. 2023. 5G Satellite Integration. https://www.iis.fraunhofer.de/en/ff/kom/satkom/sat-5g.html. [Online; accessed 22-January-2023].

[18] Brian Gaide, Dinesh Gaitonde, Chirag Ravishankar, and Trevor Bauer. 2019. Xilinx Adaptive Compute Acceleration Platform: VersalTM Architecture. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (Seaside, CA, USA) *(FPGA '19)*. Association for Computing Machinery, New York, NY, USA, 84–93. https://doi.org/10.1145/3289602.3293906

[19] Max Ghiglione and Vittorio Serra. 2022. Opportunities and Challenges of AI on Satellite Processing Units. In *Proceedings of the 19th ACM International Conference on Computing Frontiers* (Turin, Italy) *(CF '22)*. Association for Computing Machinery, New York, NY, USA, 221–224. https://doi.org/10.1145/3528416.3530985

[20] S. Haykin. 2005. Cognitive radio: brain-empowered wireless communications. *IEEE Journal on Selected Areas in Communications* 23, 2 (2005), 201–220. https://doi.org/10.1109/JSAC.2004.839380

[21] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. 2021. A White Paper on Neural Network Quantization. arXiv:arXiv:2106.08295

[22] Dominika Przewlocka-Rus, Syed Shakib Sarwar, H. Ekin Sumbul, Yuecheng Li, and Barbara De Salvo. 2022. Power-of-Two Quantization for Low Bitwidth and Hardware Compliant Neural Networks. arXiv:arXiv:2203.05025

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. https://doi.org/10.48550/ARXIV.1505.04597

[24] The MathWorks, Inc. 2022. Custom IP Core Generation. https://www.mathworks.com/help/hdlcoder/ug/custom-ip-core-generation.html. [Online; accessed 22-January-2023].

[25] Yaman Umuroglu, Nicholas J. Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. 2017. FINN: A Framework for Fast, Scalable Binarized Neural Network Inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (Monterey, California, USA) *(FPGA '17)*. Association for Computing Machinery, New York, NY, USA, 65–74. https://doi.org/10.1145/3020078.3021744

[26] Woodrow Wilson International Center for Scholars. 2021. Seizing Opportunities: Four National Security Questions to Ask About the Use of Satellites in 5G Networks. https://5g.wilsoncenter.org/sites/default/files/media/uploads/documents/STIP-SeizingOpportunities.pdf. [Online; accessed 22-January-2023].

[27] Xilinx Inc. 2022. DPUCVDX8G for Versal ACAPs. https://www.xilinx.com/content/dam/xilinx/support/documents/ip_documentation/dpucvdx8g/v1_1/pg389-dpucvdx8g.pdf. [Online; accessed 24-January-2023].

[28] Xilinx Inc. 2022. System-Level Benefits of the Versal Platform. https://www.xilinx.com/content/dam/xilinx/support/documents/white_papers/wp539-versal-system-level-benefits.pdf. [Online; accessed 25-January-2023].

[29] Xilinx Inc. 2022. UltraScale Architecture and Product Data Sheet: Overview. https://www.xilinx.com/content/dam/xilinx/support/documents/data_sheets/ds890-ultrascale-overview.pdf. [Online; accessed 25-January-2023].