Implementation of ResNet-based model for RF Modulation Classification and Benchmarking on CPU,GPU, Zynq UltraScale+ MPSoC ZCU104

By

Subhasri chavakula

DEPARTMENT OF ASTRONOMY,
ASTROPHYSICS AND SPACE ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY
INDORE

[04/07/2024]

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled Implementation of ResNet-based model for RF Modulation Classification and Benchmarking on CPU,GPU, Zynq UltraScale+ MPSoC ZCU104 in the fulfillment of the requirements for the award of the degree of **MASTER OF SCIENCE (RESEARCH)** and submitted in the **Department of Astronomy,Astrophysics and space Engineering, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from Aug 2021 to May 2023 under the supervision of Prof. Abhirup Datta.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Subhasri Chavakula 05/07/2024
Signature of the student with date
(Subhasri Chavakula)

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge.

5/7/24
Signature of the Supervisor of
MS (Research) thesis #1 (with date)
(Prof Abhirup Datta)

Chavakula Subhasri has successfully given his/her MS (Research) Oral Examination held on 05/07/2024.

05/07/2024
SOMADITYA SEN
Signature of Chairperson (OEB) with date

5/7/24
Signature(s) of Thesis Supervisor(s) with date

05/07/24
Signature of Convener, DPGC with date

5/7/24
Signature of Head of Discipline with date

ABSTRACT

The spectrum sensing component that implements the function of automatic modulation classification (AMC) has always been a major impediment in the design of cognitive radios . This barrier can be overcome with the transition to software-defined radios (SDRs), followed by the introduction of field-programmable gate arrays (FPGAs) and deep learning (DL). However, in current implementation frameworks, the design of DL models is still separated from synthesised FPGA designs. As a result, the design process is complex and time-consuming.This thesis goal is to find the implementation framework for implementing deep learning inference models within signal processing chains on FPGAs.The goal of this framework is to achieving high throughput, low latency, and a small FPGA resource footprint that allows for scaling to larger DL models. This thesis presents a ResNet-based model for Automatic Modulation Classification. Initially, a model to identify only a single modulation type was implemented, and it was later developed to identify multiple modulation types contained in the same RF frame(mixed-signal modulation classification). while dealing with the real time applications Latency and throughput are the crucial factors,so we benchmarked both the single-signal and mixed-signal models on CPU (Central processing unit), GPU (Graphics processing unit), and Xilinx ZCU104 FPGA (Field programmable gate array) to test the time taken to generate predictions. From the benchmarking results, It has been demonstrated that the performance of FPGA surpasses that of CPU and GPU.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic recognition of the modulation type of a signal that has been detected is a crucial aspect of intelligent receiver design, with numerous applications in both civilian and military contexts. Modulation classification is an essential function for managing and monitoring communication systems, enabling the resolution of technical challenges such as spectrum awareness, interference mitigation, and adaptive transmissions. In order to address these challenges, it is necessary to accurately identify and classify the modulation type of the detected signal. With the development of advanced machine learning techniques and algorithms, modulation classification has become an increasingly sophisticated field of research, leading to improved performance and more efficient use of communication resources. Therefore, the accurate and efficient classification of modulation type is a critical component in the design of intelligent receivers, with significant implications for a wide range of communication systems and applications. Automatic modulation classification has numerous applications in both civilian and military contexts. One of the most significant applications of modulation classification is in

the management and monitoring of communication systems. By accurately identifying and classifying the modulation format of detected signals, modulation classification enables spectrum awareness and adaptive transmissions, which are critical for efficient use of communication resources. This, in turn, leads to improved performance, greater reliability, and higher efficiency in communication systems. In addition, modulation classification is essential for interference avoidance, enabling communication systems to operate more effectively in complex and dynamic environments.

Another important application of modulation classification is in the area of electronic warfare. Military communication systems operate in highly contested environments, where adversaries may attempt to disrupt, jam or intercept transmissions. Automatic modulation classification is a key component of modern electronic warfare systems, enabling rapid and accurate identification of signals in complex and dynamic environments, and facilitating the deployment of countermeasures to mitigate or neutralize any threats.

In addition to these applications, modulation classification has numerous civilian applications, including in wireless communication systems, radar systems, and satellite communication systems. For example, in wireless communication systems, modulation classification is essential for managing and monitoring the radio spectrum, identifying interference sources, and adapting transmission parameters to optimize system performance. In radar systems, modulation classification is used for target identification, discrimination, and tracking, enabling more precise and effective radar operations. In satellite communication systems, modulation classification is critical for optimizing bandwidth utilization and ensuring reliable communication links.

Overall, automatic modulation classification has a wide range of applica-

tions, from managing and monitoring communication systems to supporting electronic warfare and enhancing the performance of wireless, radar, and satellite communication systems. The development of advanced machine learning techniques and algorithms has enabled significant progress in the field of modulation classification, leading to more sophisticated and effective approaches for accurately identifying and classifying the modulation format of detected signals.

With the rise of Software Defined Radio (SDR), the task of Automatic Modulation Recognition (AMR) has become an essential component of intelligent communication systems. In particular, SDR allows for the use of general-purpose hardware to carry out the functions of radio frequency communication, which has led to an increase in the need for AMR. Additionally, Cognitive Radio (CR) also relies on AMR for spectrum sensing.

Recent advancements in Deep Learning (DL) have led to the development of new and powerful tools that are capable of addressing problems in AMR. As a result, many researchers are now incorporating DL models into AMR. These models are capable of handling complex signal processing tasks and can identify modulation formats with high accuracy.

The incorporation of DL models into AMR has also led to improvements in the efficiency and speed of the process. The use of DL models can significantly reduce the time required for AMR, making it possible to process large amounts of data in real-time. This has made AMR more practical for many applications, particularly in time-sensitive environments.

Moreover, DL models can be used in combination with traditional signal processing methods to further improve the accuracy and efficiency of AMR.

This hybrid approach can effectively leverage the strengths of both methods to achieve better performance in challenging environments.

Overall, the integration of DL models into AMR has opened up new avenues for research and development in this field. The combination of DL models with traditional signal processing methods has enabled the development of more accurate and efficient AMR systems, which have numerous applications in both civilian and military contexts.

In the modern era, machine learning (ML) and deep learning (DL) techniques have become increasingly popular and useful in a wide range of fields, including computer vision, healthcare, robotics, and communication systems. Unlike traditional ML algorithms, DL networks can learn from unstructured data without any guidance from expert engineers. This is because DL networks operate like the human brain, processing data and creating patterns that can be used for decision-making without any human supervision. As a result, DL networks have become indispensable tool for solving complex problems in various fields. With the continuous advancements in DL technology, researchers are exploring new ways to improve the performance of DL networks and make them more versatile and efficient.

Although deep learning technology has become increasingly popular in radio systems, its practical application remains a challenge. Radio systems typically use Application Specific Integrated Circuits (ASICs) or Field Programmable Gate Arrays (FPGAs) for signal processing, while most deep learning models require CPU or GPU platforms which are not suitable for digital signal processing due to their sequential nature or high power consumption. Thus, integrating deep learning models on FPGA platforms can potentially enhance their operability in the signal processing chain of a soft-

ware defined radio (SDR). The purpose of this study is to investigate the feasibility of integrated deep learning models on FPGA platforms, and to enable future research on the use of RF deep learning in communication systems.

Chapter 2

Background

2.1 Automatic Modulation classification

Automatic Modulation Classification (AMC) is the process of identifying the modulation type of a received signal. In other words, it is the process of determining whether the signal is modulated using Amplitude Modulation (AM), Frequency Modulation (FM), or other digital modulation types such as Phase Shift Keying (PSK) or Quadrature Amplitude Modulation (QAM). AMC is a crucial task in many communication systems, including military and civilian applications, where it is necessary to identify the signal modulation type for monitoring, management, and control.

In the field of communication systems, there are two types of modulation: analog and digital modulation. Analog modulation is a technique used to modulate a continuous wave (carrier) signal with an analog message signal, such as voice or music. On the other hand, digital modulation is a technique where a digital message signal is modulated onto a carrier wave.

Digital modulation types include Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), Phase Shift Keying (PSK), and Quadrature Amplitude Modulation (QAM).

2.2 Mixed signal Modulation Classification

As the demand for wireless communication continues to increase, the available spectrum resources are becoming more limited. This has led to the need for more efficient use of the spectrum through the use of mixed-signal modulation[8]. Mixed signal modulation involves the transmission of multiple signals on a single RF frame, which makes the automatic modulation classification of these signals a crucial task[9][10].

The emergence of effective mixed signal automatic modulation classification technology is of great significance as it enables us to efficiently detect and classify multiple signals on a single RF frame. This technique is useful in many applications such as wireless sensor networks, cognitive radio, and satellite communication systems.

In Conclusion, The applications of AMC are widespread, and it plays a crucial role in many communication systems. In military applications, AMC is used to identify the type of modulation used by the enemy, which can provide valuable intelligence. In civilian applications, AMC is used for spectrum awareness, adaptive transmissions, and interference avoidance. For example, in a wireless communication system, AMC can be used to adaptively select the best modulation type to transmit data in a changing radio environment.

2.3 Deep Learning for automatic Modulation Classification

Machine learning (ML) and deep learning (DL) are two different methods of data analysis used in various fields[2][3]. ML involves extracting features from raw data and using those features in a classifier to make predictions. Modulation classification prediction, for instance, involves statistical properties like moment and cumulants, and instantaneous properties like signal power. However, extracting features can be challenging in some cases.

On the other hand, DL combines feature extraction and classification into one model. This approach makes DL more effective in cases where it is difficult to identify the features in advance. For modulation classification, DL has proven to be a better option than ML because it can extract features autonomously and make accurate predictions based on those features.

In the case of modulation classification, DL models can be trained on large datasets of modulation types and features, allowing them to identify and classify new signals accurately. DL models can also adapt to new modulation types, making them more versatile than traditional ML models. As a result, DL has become a popular choice for modulation classification in recent years.

2.4 Convolutional Neural Network

Convolutional neural networks (CNNs) have become a popular tool for automatic modulation classification (AMC) due to their ability to effectively extract features from complex signal data.Convolutional neural networks

(CNNs) are a type of deep neural network that have proven to be effective in handling complex signal data. They are made up of convolution, pooling, and dense layers, and have become increasingly popular in many fields, including computer vision, natural language processing, and signal processing.Convolutional Neural Networks (CNNs) rely on the conventional artificial neuron model and aim to enhance model performance and precision. Their pipelined feed-forward execution flow is highly parallelizable, enabling them to optimize performance and accuracy.

In the context of automatic modulation classification (AMC), the goal is to use a CNN to accurately identify the modulation scheme used to encode information in a received signal. The CNN is trained on raw I/Q data, which is the in-phase and quadrature components of the signal. This raw data is fed into the CNN, which learns to extract useful features from it through its convolution and pooling layers.

Convolutional Neural Network (CNN), which comprises two main components: the feature extraction part and the classification part. The feature extraction is accomplished through the use of convolution layers, which perform a series of convolutions on the input data to extract relevant features. On the other hand, the classification part is generally achieved through one or multiple fully connected layers.

It is worth mentioning that although fully connected networks are also capable of extracting features from the input, the separation between the feature extraction and classification parts is more evident in a CNN. The convolution layers enable the network to learn distinctive spatial patterns and relationships within the input data, which is particularly useful in tasks such as image recognition.

In a CNN, the output of each convolution layer serves as the input to the subsequent layer, leading to a hierarchical feature representation of the input data. This hierarchical representation allows the network to capture increasingly complex and abstract features in the deeper layers, which ultimately leads to better classification performance.

After the convolutional layers, the network may include additional layers such as pooling layers and activation layers. Pooling layers reduce the dimensionality of the feature maps by summarizing local information, while activation layers introduce non-linearity to the network, enabling it to model complex relationships between the input data and the desired output.

In this context, we are utilizing Convolutional Neural Networks (CNNs) for modulation classification. CNNs have been found to be a popular choice for this task, likely due to their ability to learn and extract features from complex input data, such as radio frequency signals. Additionally, CNNs may be easier to train than other types of neural networks, as they can be trained in a highly parallelizable manner and can learn from small amounts of labeled data. CNNs have shown promising results in modulation classification, achieving high accuracy rates in various scenarios, including noisy and fading channels.

2.5 Residual Network (ResNet) :

Residual Network (ResNet) is a popular deep learning model widely used for computer vision tasks. It is a type of Convolutional Neural Network (CNN) architecture designed to accommodate a large number of convolutional layers[11].

During the training process of neural networks, which involves backpropagation and gradient descent, the gradient is used to update the weights in order to minimize the loss function. With too many layers, the gradient can become increasingly small through repeated multiplications, eventually becoming too insignificant to effectively update the weights. This phenomenon is known as the "vanishing gradient" problem, and it can lead to saturation or degradation in performance as more layers are added to the network.

ResNet presents an innovative solution to tackle the vanishing gradient problem by employing "skip connections". In ResNet, multiple identity mappings, which are essentially convolutional layers that initially do not alter the input, are stacked together, and these layers are skipped, allowing the activations from the previous layer to be reused. This approach accelerates the initial training process by effectively compressing the network into fewer layers.

As the network is further trained, all layers are expanded, and the residual parts of the network, referred to as residual blocks, are allowed to explore more diverse features of the input image. Typically, ResNet models skip two or three layers at a time, with nonlinearity and batch normalization applied in between, to maintain the network's ability to learn complex representations while mitigating the vanishing gradient issue. This strategy has been shown to improve the training dynamics and performance of deep neural networks like ResNet.

Residual Block : A residual block is a fundamental component of the ResNet architecture, a type of Convolutional Neural Network (CNN). It is composed of multiple convolutional layers and introduces skip connections, also known as shortcut connections, that allow the activations from the previ-

ous layer to bypass intermediate convolutional layers and directly contribute to the output of the residual block. This skip connection helps to mitigate the vanishing gradient problem, which can occur when training deep neural networks, by facilitating the flow of gradients during backpropagation.

Typically, a residual block includes batch normalization and nonlinearity (such as ReLU) applied after each convolutional layer to enhance the network's learning capacity. The combination of skip connections, batch normalization, and nonlinearity in the residual block enables the successful training of deep neural networks, allowing for the construction of ResNet models with a large number of layers. This architecture has been proven to achieve state-of-the-art performance in various computer vision tasks, such as image classification, object detection, and semantic segmentation, due to its ability to effectively address the challenges associated with vanishing gradients.

Chapter 3

FPGA-based acceleration of Deep Learning algorithms.

3.1 Field Programmable Gate array(FPGA)

A Field Programmable Gate Array (FPGA) is a programmable integrated circuit that can be customized to perform specific digital logic functions. Although ASICs typically outperform FPGAs in terms of power consumption and throughput, the development of FPGAs is more cost-effective and faster due to their reprogrammable nature.

The modern FPGA is composed of two primary elements: fine-grained programmable logic blocks, which typically consist of adaptive logic modules (ALMs), and coarse-grained functional logic components, including memory blocks, digital signal processing (DSP) blocks, communication blocks, and soft-core processors.

The use of ALMs in the fine-grained programmable logic blocks allows for high levels of flexibility and customization, enabling the FPGA to be adapted to a wide range of applications. Meanwhile, the coarse-grained functional logic components provide specialized hardware support for specific tasks, such as efficient memory access and signal processing.

FPGAs offer several advantages over traditional processors, including high performance, low latency, and low power consumption[6][7], making them ideal for applications such as digital signal processing, image processing, and machine learning acceleration. Additionally, the programmable nature of FPGAs allows for rapid prototyping and iterative design, enabling developers to quickly test and refine their designs.

Figure 3.1 shows the architecture of FPGA . The architecture of an FPGA typically includes three main types of modules: I/O blocks or pads, switch matrix/interconnection wires, and configurable logic blocks (CLBs). The basic architecture of an FPGA consists of two-dimensional arrays of logic blocks, with a user-configurable means of interconnecting these blocks.

I/O blocks provide input/output connections for the FPGA and allow it to interface with external devices. The switch matrix or interconnection wires provide a means of connecting the logic blocks to one another and to the I/O blocks, enabling the creation of custom logic functions. Configurable logic blocks (CLBs) are the building blocks of the FPGA and can be configured to perform a wide range of logic functions.

Configurable Logic Blocks (CLBs) are an essential part of the FPGA architecture, consisting of a MUX (Multiplexer), a D flip flop, and a Look Up Table (LUT). The LUT is responsible for implementing combinational

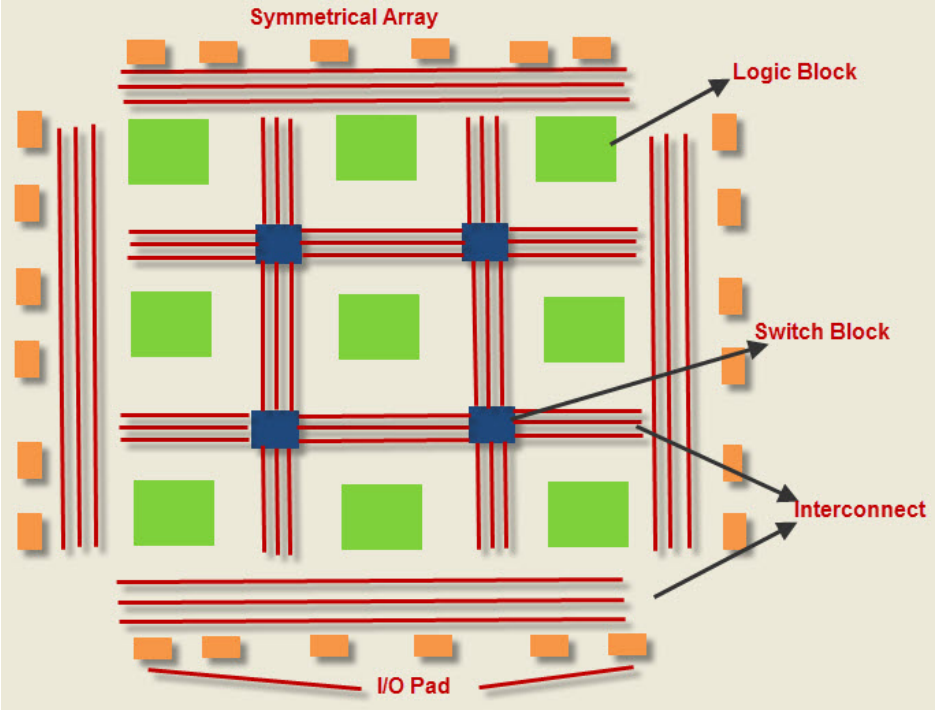

Figure 3.1: Architecture of FPGA[14]

logical functions, while the MUX is used for selection logic and the D flip flop stores the output of the LUT.

The basic building block of an FPGA is the Look Up Table (LUT) based function generator, which can have varying numbers of inputs ranging from 3, 4, 6, and even 8, depending on the specific FPGA design. Recent advances in FPGA technology have led to the development of adaptive LUTs, which can provide two outputs per single LUT by implementing two function generators within the same block.

CLBs are highly configurable, allowing users to customize the logic functions they perform by reprogramming the LUTs and MUXes. The adaptability and versatility of CLBs make them well-suited for a wide range of applications, including digital signal processing, cryptography, and machine

learning.

3.2 Hardware Comparison for Artificial Intelligence: FPGAs vs. GPUs vs. ASICs

Latency : FPGAs have a lower latency compared to CPUs and GPUs as they run on "bare metal" without an operating system, allowing for faster processing times. This makes them ideal for real-time AI applications. Both logic transistors and software programs can execute instructions, resulting in faster processing times[4][5]. ASICs and FPGAs are particularly useful in scenarios that require lower latencies.

Power Consumption : FPGAs excel over GPUs and CPUs in applications with limited power availability. Since they operate on "bare metal," they require less power consumption, making them a more efficient choice for such scenarios.

Flexibility : In comparison to ASICs, FPGAs offer superior flexibility and are becoming more competitive in terms of pricing. They are particularly beneficial for AI applications that evolve and change rapidly, where neural networks require frequent adjustments and retraining. FPGAs can be reprogrammed for design changes in a matter of hours to weeks, whereas ASICs have a longer production cycle time of 12-18 months. As a result, FPGAs are preferred for applications that require maximum flexibility and adaptability.

FPGAs offer a greater degree of programming flexibility compared to GPUs, allowing for the addition of new steps or outputs without changing the

GPU architecture itself. For example, if additional cleaning steps are required for the bottling process, an FPGA can be programmed to include them, while a GPU would require physical changes to accommodate such alterations. Furthermore, FPGAs offer greater flexibility beyond AI applications, such as adding networking or post-processing capabilities. In contrast, GPUs are more limited in their flexibility and can only perform one task at a time. Additionally, FPGAs can be reprogrammed to perform completely different functions without requiring the development of a new chip or waiting for an 18-month production cycle as is the case with ASICs. The ability of FPGAs to quickly adapt to changing requirements is a significant advantage over GPUs and ASICs.

Parallel Computing : The parallel computing used in DNNs can introduce complexities and computational hardware imbalances if irregular parallelisms occur. FPGAs are better than GPUs for applications with custom data types or irregular parallelism. Both GPUs and FPGAs can process in parallel, but FPGAs are more efficient in parallel processing, performing several steps at each cycle, whereas GPUs can only do one row at a time. FPGAs are faster, more flexible, and efficient than GPUs.

Conclusion : AI is a new and exciting field that can help solve problems, and improve efficiency. FPGAs are a type of technology that can help with AI because they are flexible, powerful, and can be easily adapted to new applications. Although they used to be hard to program, there are now tools that make it easier to use FPGAs in AI.

3.3 VITIS AI

The use of FPGAs to speed up deep learning algorithms is becoming popular among both academics and the industry. Various methods and frameworks are available for implementing deep learning models on an FPGA.

The Vitis AI platform is a complete solution for developing AI models on AMD devices, boards, and Alveo data center acceleration cards. It includes optimized DPU cores, AI models, libraries, tools, and examples for edge and data center AI. Its goal is to make AI acceleration efficient and easy to use on AMD FPGAs and SoCs.

VITIS AI provides assistance for common frameworks and the newest models used in deep learning, including CNNs, RNNs, and NLPs. It includes tools for optimizing and quantizing models for better accuracy and efficiency. Users can easily compile and deploy custom models using high-level APIs. Additionally, VITIS AI offers highly configurable DPU cores that can be tailored to specific requirements for throughput, latency, and power in both cloud and edge computing applications.

When it comes to accelerating deep learning, there are two primary FPGA architectures: streaming and computation unit. In streaming architecture, data samples are processed through the neural network in a continuous ”stream”, and the accelerator is tailored to the specific network being used. This means that if a different network is to be run, the accelerator must be reconfigured to accommodate it. On the other hand, the computation unit architecture executes instructions and creates an accelerator that can be used across different networks, as long as they can be translated into compatible instructions. This architecture offers more flexibility and is generally eas-

ier to maintain and adapt to different network models.Vitis AI is based on Computation unit architecture. Figure 3.2 from ref.[13] shows the VITIS AI structure.

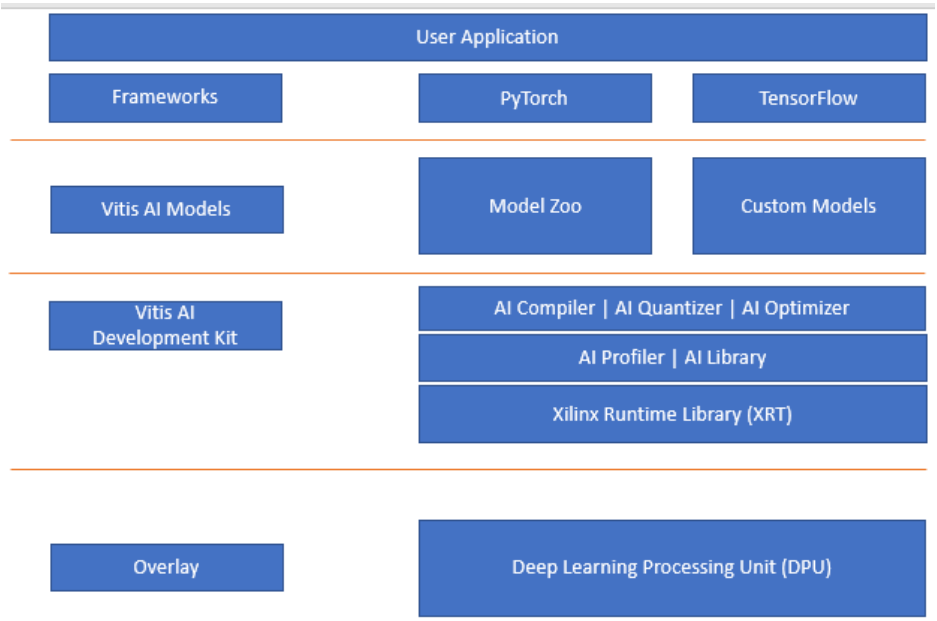

Figure 3.2: Vitis AI development environment[13]

Main Parts of VITIS AI :

1. **Model Zoo :** The AI Model Zoo is a collection of pre-built deep learning models from well-known frameworks like Pytorch, Tensorflow, Tensorflow 2, and Caffe. It is accessible to everyone and provides optimized models that can be retrained and used on AMD platforms for faster execution, improved performance, and production.

2. **Optimizer :** The AI optimizer technology can compress models up to 50 times, with minimal impact on accuracy. This deep compression technology can significantly enhance the performance of AI inference.

3. **Optimizer :** The process of inspecting custom operators, quantizing, calibrating, fine-tuning, and converting floating-point models into fixed-point models is completed to improve computing efficiency and speed by reducing memory bandwidth usage.

4. **Compiler :** The mapping of the AI model to a highly efficient instruction set and data flow is carried out by the AI compiler, which also applies sophisticated optimizations like layer fusion and instruction scheduling.

5. **Profiler:** Programmers can use the performance profiler to conduct a comprehensive analysis of the efficiency and resource utilization of their AI inference implementation.

6. **Library :** The Vitis AI Library comprises a collection of advanced libraries and APIs designed to optimize AI inference using DPU cores. Leveraging the Vitis AI Runtime (VART) with uniform APIs, it offers simple-to-use interfaces that enable hassle-free deployment of AI models on AMD platforms.

7. **Deep-Learning Processor Unit (DPU) :** The DPU is a specialized computing architecture that is capable of adapting to the rapidly evolving AI algorithms of CNNs, RNNs, and NLPs. It offers exceptional performance and is compatible with leading-edge technologies found on Zynq™ SoCs, Zynq UltraScale+™ MPSoCs, Alveo data center accelerator cards, and Versal ACAPs.

3.4 Deep-Learning Processor Unit (DPU)

A DPU consists of components that are present in the Xilinx programmable logic fabric, such as BlockRAM, DSP, UltraRAM, LUTs, and Flip-Flops[15]. Alternatively, it can be developed as a set of microcoded functions that are deployed on the AI Engine architecture, or "AI Engine." For certain applications, the DPU may comprise programmable logic and AI Engine array resources.

Vitis AI offers both the DPU IP and the essential tools for deploying neural networks on Xilinx targets, regardless of whether the neural networks are standard or custom.

3.5 Zynq® Ultrascale+™ MPSOC ZCU104

With the ZCU104 Evaluation Kit, designers can quickly begin creating embedded vision applications, including but not limited to Advanced Driver Assisted Systems (ADAS), machine vision, medical imaging, drones, Augmented reality(AR), and surveillance.

The board comes equipped with a quad-core ARM Cortex-A53 processor, a dual-core ARM Cortex-R5 real-time processor, a Mali-400 MP2 graphics processing unit, and an FPGA that can be programmed using the Vivado software suite. Additionally, it has several peripheral devices, including USB 3.0 ports, Gigabit Ethernet, HDMI, DisplayPort, and an SD card slot.

The ZCU104 board has gained popularity among developers, researchers, and engineers interested in designing and prototyping high-performance ap-

plications. Figure 3.3 shows the Zynq UltraScale+ XCZU7EV-2FFVC1156 MPSoC.

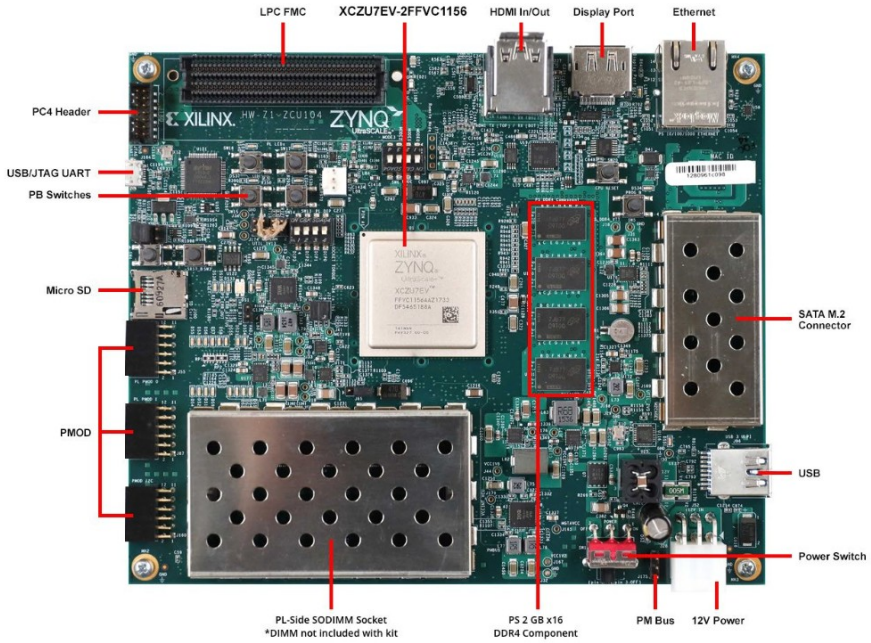

Figure 3.3: Zynq UltraScale+ XCZU7EV-2FFVC1156 MPSoC

Chapter 4

Experimental Results

Modulation classification is an important task in wireless communication systems, which involves identifying the modulation scheme of a given signal. Deep learning has shown great potential in addressing this problem, and Convolutional Neural Networks (CNNs) have emerged as a powerful tool for modulation classification. In this study, we developed a ResNet-based CNN model for modulation classification, and used the VITIS AI tool for inference on FPGA to improve the performance.

FPGAs are specialized hardware devices that can provide significant acceleration for deep learning inference, and have been shown to outperform CPUs and GPUs in many cases. By deploying our ResNet-based CNN model on FPGA, we aimed to achieve higher inference throughput and lower latency than on traditional CPU and GPU platforms.

To evaluate the performance of our model, we conducted benchmarking experiments on a range of platforms, including CPU, GPU, and FPGA. We

measured the inference time and throughput for each platform, and compared the results to determine the relative performance of each platform.

To achieve this goal, we followed a series of steps

1. **Data Collection :** In this study, for the single signal AMC Model, we used the Deepsig 2018 Dataset which was generated in [1], which comprises a variety of digital and analog modulation types, including OOK, 4ask, 8ask, bpsk, qpsk, 8psk, 16psk, 32psk, 16apsk, 32apsk, 64apsk, 128apsk, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, am-ssb-wc, am-ssb-sc, am-dsb-wc, am-dsb-sc, fm, gmsk, and oqpsk. This dataset contains both synthetic simulated channel effects and over-the-air recordings, providing a diverse range of signal characteristics.

   Each of the 2,555,904 data inputs in this dataset comprises complex (I Q) data that is 1024 samples long. The dataset provides 26 different Signal-to-Noise Ratio (SNR) values for each signal type, ranging from -20 dB to 18 dB. For each signal and SNR combination, there are 4096 instances, and each instance always has a length of 1024 samples.

   Using this dataset allowed us to train and evaluate our ResNet-based CNN model on a diverse range of modulation types and signal characteristics. The inclusion of both simulated and real-world recordings provides a more realistic representation of the challenges faced in wireless communication systems.

   For the mixed signal AMC Model, A subset of the original Deepsig 2018 dataset was extracted to create a smaller dataset. The new dataset was analyzed with respect to three fundamental signals, namely OOK, Four-ASK, and QPSK. These signals were subsequently combined to

generate a new dataset that consists of six signals: OOk, FOUR_ASK, QPSK, OOK+FOUR_ASK, OOK+QPSK, FOUR_ASK+QPSK.

2. **ResNet Model:**In our research, we used a ResNet-based model that drew inspiration from the work of [1]. Our model was trained on the Deepsig RadioML 2018.01A dataset, a widely used benchmark dataset for radio signal modulation classification. The architecture of our model comprised of four ResNet stacks, each consisting of two ResNet blocks followed by a max pooling layer.The ResNet model for single signal modulation classification is shown in Table 4.1 .

   The ResNet blocks in our model incorporate skip connections, which allow for the efficient training of deep neural networks by mitigating the vanishing gradient problem. The skip connections enable the gradient to flow more easily during backpropagation, promoting better convergence and enhancing the overall performance of the model.

   Furthermore, the inclusion of max pooling layers after each ResNet block helps in down-sampling the feature maps, allowing the model to capture both local and global contextual information from the input signals. This facilitates the extraction of relevant features for accurate signal modulation classification.

   By leveraging the strengths of the ResNet architecture, the Deepsig RadioML 2018.01A dataset, and our specific model configuration with multiple ResNet blocks and max pooling layers, we aimed to achieve superior performance in radio signal modulation classification.

   For our mixed-signal classification task, we employed the ResNet model that was originally developed and used for single signal classification. This choice was motivated by the success of the ResNet architecture in

Table 4.1: ResNet Based Model - Single signal AMC

Layer	Output Shape	Parameters
InputLayer	1024,1,2	0
Conv2D	1024,1,32	352
BatchNorm	1024,1,32	128
Activation	1024,1,32	0
	4*ResNetBlocks	
Conv2D	64,1,32	5152
BatchNorm	64,1,32	128
Activation	64,1,32	0
Reshape	8,8,32	0
GlobAvgPool	32	0
Dense	256	8448
Dropout	256	0
Dense	128	32896
Dropout	128	0
Dense	22	2838
SoftMax	22	0
Total Params		139158
Trainable Params		137750
Non Trainable		1408

various signal processing tasks and its proven effectiveness in capturing deep feature representations. For mixed signal modulation classification, Using the original Deepsig RadioML 2018.01A DATASET, we created a dataset for Mixed-signal modulation classification. In this case, we evaluated and mixed three signals: OOK, FOUR-ASK, and QPSK.

mixed-signall automatic modulation classification technology is of great significance as it enables us to efficiently detect and classify multiple

signals on a single RF frame. The ResNet model for mixed-signal modulation classification is shown in Table 4.2 .

Table 4.2: ResNet Based Model - Mixed Signal AMC

Layer	Output Shape	Parameters
InputLayer	1024,1,2	0
Conv2D	1024,1,32	352
BatchNorm	1024,1,32	128
Activation	1024,1,32	0
	4*ResNetBlocks	
Conv2D	64,1,32	5152
BatchNorm	64,1,32	128
Activation	64,1,32	0
Reshape	8,8,32	0
GlobAvgPool	32	0
Dense	256	8448
Dropout	256	0
Dense	128	32896
Dropout	128	0
Dense	6	774
SoftMax	6	0
Total Params		137094
Trainable Params		135686
Non Trainable		1408

3. **Train the Model:**Firstly, The Keras model needs to be trained. Once the training is completed, the next step is to import the model into Viti's AI tools.

4. **Evaluate the model :** When evaluating a floating point model, accuracy is influenced by the Signal-to-Noise Ratio (SNR), where lower SNR values tend to result in lower accuracy, while higher SNR values

tend to yield higher accuracy. It's important to consider the impact of SNR on model performance, as it can affect the model's ability to accurately process and interpret data. Additionally, SNR is a critical factor in determining the quality of the input data, and a higher SNR can lead to more reliable and accurate model predictions. Figure 4.1shows the SNR vs Accuracy plot for the floating point model.

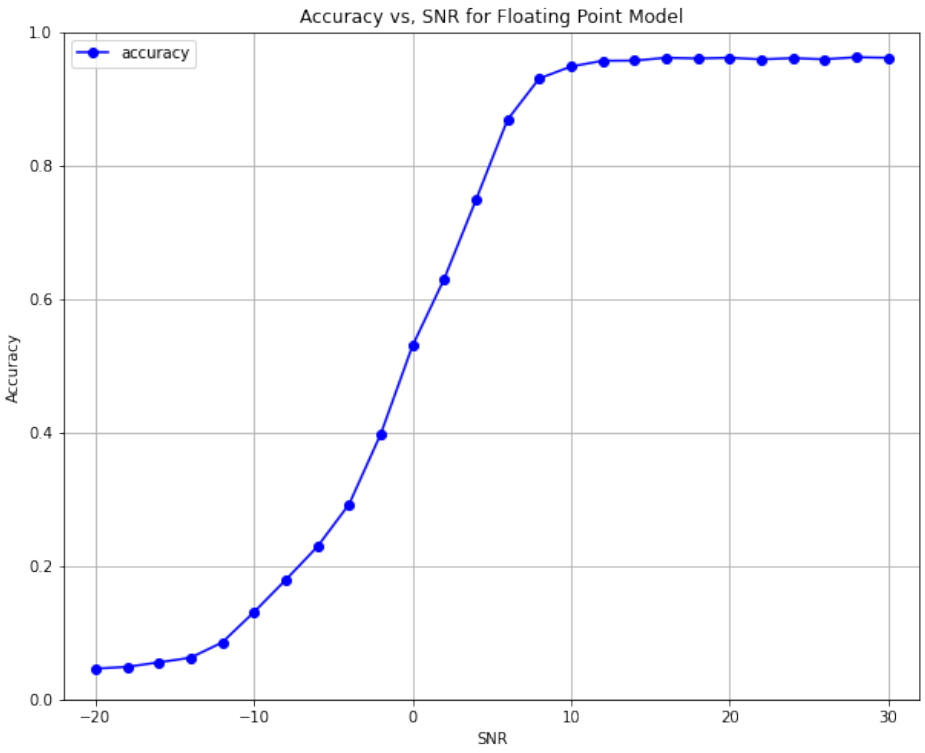

Figure 4.1: SNR vs Accuracy plot for the floating point model

5. **Quantize the model:** Vitis AI quantizer quantizes the 32-bit floating point model into INT8 model, After quantizing from a 32 bit model to an 8 bit model, accuracy decreased by only 5 percentage. Figure 4.2 shows the SNR vs Accuracy plot for the INT8 model.

   We are removing AM-SSB-WC and AM=SSB-Sc from our dataset to

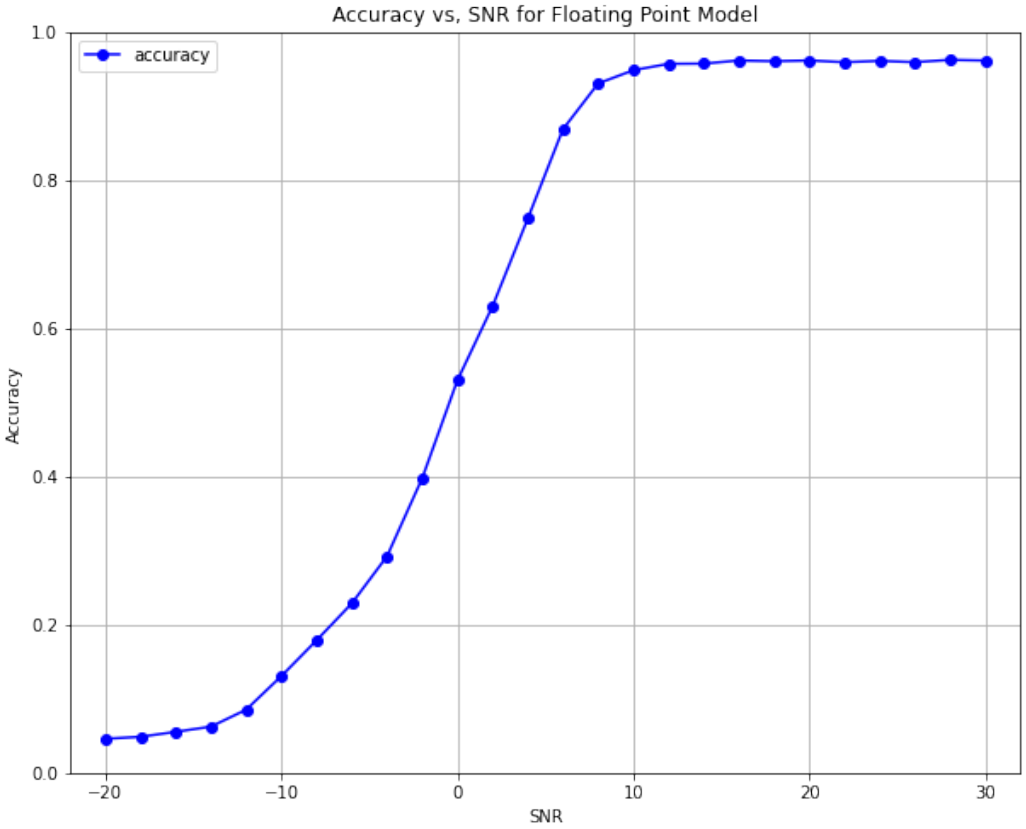

Figure 4.2: Accuracy vs SNR plot for INT8 Model

increase accuracy after quantizing; otherwise, more accuracy loss will occur.

6. **Finetune the Model :** Fine-tuning is the process of retraining the quantized model on the original dataset to restore some of the lost accuracies.After the fine-tuning process, we achieved a significant improvement in accuracy, with an increase of 6% compared to the initial performance.

7. **Compile model for Deep Learning processing unit (DPU) :** Vitis-AI reads the quantized model and generates an instruction set for the Xilinx DPU.

8. **Deploy the model on ZCU104 MPSOC :** finally deploy the generated files on the ZCU104 MPSOC.

4.1 ResNet model Deployment on ZCU104 - Results :

Single signal Modulation Classification results: Latency and throughput are critical parameters that can significantly impact the overall performance of a network or system. In simple terms, latency represents the time taken for a data packet to travel from the source to the destination, whereas throughput indicates the amount of data that can be transmitted over the network during a specific period.

After training the model, we used the Vitis AI quantizer to convert it from floating-point to fixed-point representation. Subsequently, we compiled the model using the Vitis AI compiler, which optimized it for FPGA deployment. This process transforms the quantized model into instructions suitable for the Deep Learning Processing Unit (DPU), resulting in a compiled .xmodel file ready for deployment on the DPU

For deployment on the FPGA, we first loaded the DPU bitstream onto the FPGA. Next, we loaded the model onto the DPU using the Vitis AI runtime, which provides runtime library APIs to execute the compiled model on Xilinx devices. We then ran the model on the MPSoC, which sends data to the DPU for inference and handles the results.

We utilized a Python script to perform classification and assess its performance. To test the model, we tested with 4000 RF frames and ran these

frames on 4 threads, parallelizing the computation to enhance processing speed. classification of all 4000RF frames was completed in approximately 2.537 seconds. This test resulted in a frame rate of 1576.54 frames per second, which demonstrates the ability of the ZCU104 board to process a high volume of data efficiently.

We utilized a Python script to perform Accuracy, we fed 998 RF frames to test the accuracy, with the accuracy test it is clear that the accuracy is increasing with the SNR, at lower SNR accuracy is very low and at higher SNR, accuracy is high, below 0db SNR, accuracy is very low, after 0 dB SNR, Accuracy increased, and at +10db accuracy reached to its maximum value. Figure 4.2 shows the accuracy vs SNR plot for the 8-bit model after quantization in the Vitis Ai quantizer.

To improve the accuracies lost while quantizing, we finetuned the model and we were able to recover the accuracy by 6%, we implemented the finetuned model on ZCU104, We tested with 4000 RF frames, and ran these frames on 4 threads, parallelizing the computation to enhance processing speed.4000 Rf frames that included various modulation schemes and different Signal-to-Noise Ratio (SNR) values. the classification of all 4000RF frames was completed in approximately 2.468 seconds. This test resulted in a frame rate of 1620.68 frames per second.

Mixed-signal modulation classification results : A subset of the original Deepsig 2018 dataset was extracted to create a smaller dataset. The new dataset was analyzed with respect to three fundamental signals, namely OOK, Four-ASK, and QPSK. These signals were subsequently combined to generate a new dataset that consists of six signals: OOK, FOUR_ASK, QPSK, OOK+FOUR_ASK, OOK+QPSK,FOUR_ASK+QPSK.

In order to evaluate the performance of the mixed-signal modulation classification model on ZCU104, we tested with 4000 RF frames from the generated mixed signal dataset that included six different signals with mixed signal modulation types with different Signal-to-Noise Ratio (SNR) values. We ran these frames on 4 threads, parallelizing the computation to enhance processing speed. the classification of all 4000RF frames was completed in approximately 2.535 seconds. This test resulted in a frame rate of 1577.55 frames per second, which demonstrates the ability of the ZCU104 board to process a high volume of data efficiently.

Additionally, the Accuracy for the mixed-signal AMC model was also tested on the ZCU104 FPGA board by feeding the 998 RF frames, with that it is clear that the accuracy is increasing with the SNR, at lower SNR accuracy is very low and at higher SNR, accuracy is high. Figure 4.3 shows the Accuracy Vs SNR plot for the 8-bit mixed signal AMC Model.

4.1.1 Benchmarking Results :

Benchmarking is a process of comparing the performance of a system or device against standard reference points or other similar systems. It involves measuring the speed, accuracy, and efficiency of a system when performing a particular task or executing a specific workload. The objective of benchmarking is to identify the strengths and weaknesses of a system and to evaluate how it compares to other systems in terms of performance. This type of analysis can be used to optimize system performance, identify areas for improvement, and make informed decisions when selecting hardware or software solutions for a particular application. Here we are performing benchmarking

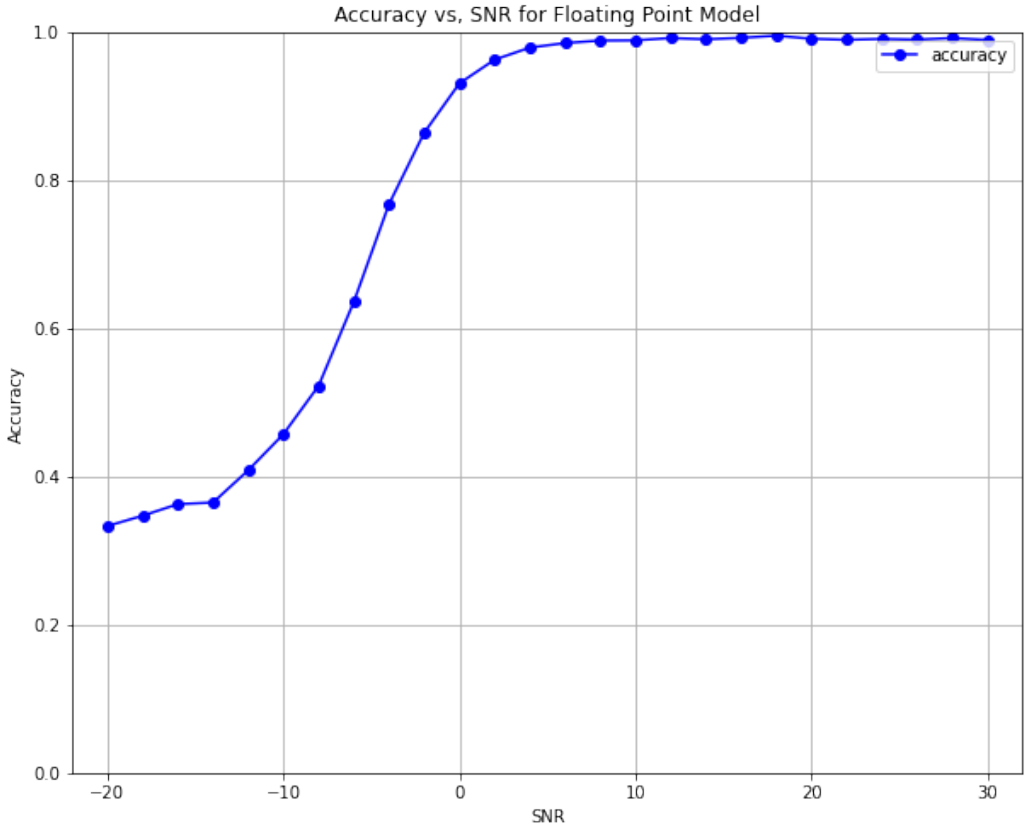

Figure 4.3: Accuracy vs SNR Plot for Mixed-signal AMC

tests on central processing unit (CPU), graphics processing unit (GPU), and field-programmable gate array (FPGA).

To compare the performance of the ResNet model on different processing systems, we employed the following hardware configurations as shown in Table 4.3.

Benchmarking Results of single signal AMC : We executed the predictions for the single signal AMC model with 1000 Rf frames on various processing systems, We evaluated and compared the execution time of predictions on different processing systems, including CPU, GPU, and FPGA as shown in Table 4.4.

Table 4.3: List of devices used

Device	Configuration
Intel i7 12800H	14 core, 2.40 GHz
NVIDIA RTX A2000 GPU	8GB
ZCU104	4-core ARM A-53, 33.33 MHz
AMD 7401P	48 core, 1.2 GHz
Quadro P2200	5GB

Table 4.4: Benchmarking Results of single signal AMC

Device	Time
Intel i7 14 Core	2.38s
NVIDIA A2000 8GB GPU	3.34s
AMD 48 Core	1.06s
Quadro P2200 GPU	1.31s
ZCU104	0.617s

After testing on 1000 RF frames, it is evident from Table 4.4 that the performance of FPGA exceeds that of CPU and GPU. Figure 4.4 shows the benchmarking results for single signal AMC.

Benchmarking Results of Mixed-signal AMC: We executed the predictions for the mixed-signal AMC model with 1000 Rf frames on various processing systems, We evaluated and compared the execution time of predictions on different processing systems, including CPU, GPU, and FPGA as shown in Table 4.5.

After testing on 1000 RF frames, it is evident from Table 4.5 that the performance of FPGA exceeds that of CPU and GPU. Figure 4.5 shows the benchmarking results for mixed-signal AMC.

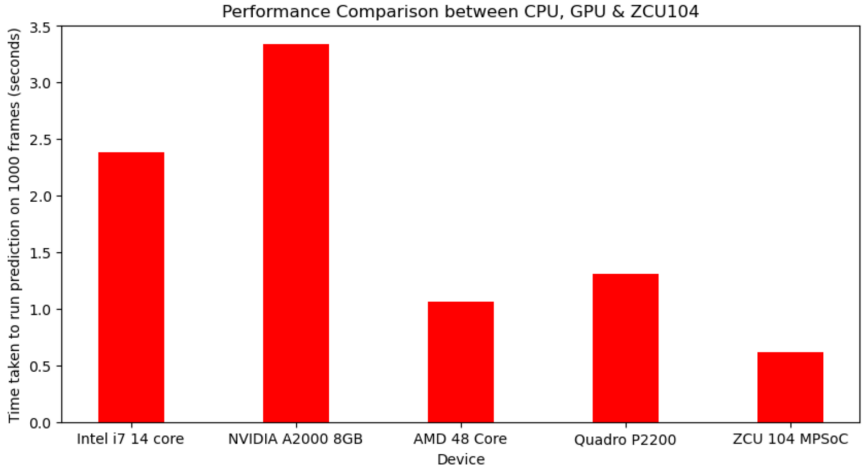

Figure 4.4: Benchmarking results for single signal AMC

Table 4.5: Benchmarking Results of Mixed-signal AMC

Device	Time
Intel i7 14 Core	2.32s
NVIDIA A2000 8GB GPU	3.08s
ZCU104	0.634s

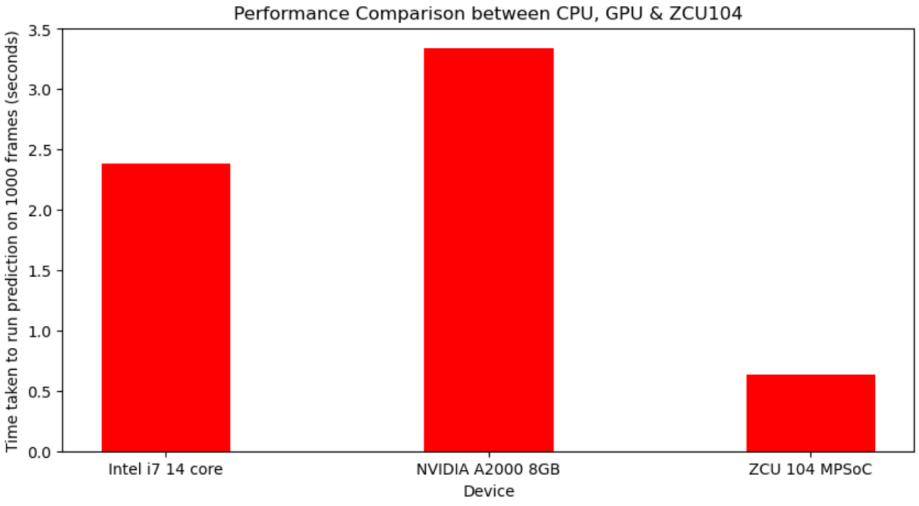

Figure 4.5: Benchmarking results for mixed signal AMC

Chapter 5

Conclusion for Thesis Stage 1

Automatic Modulation Classification (AMC) is the process of identifying the modulation type of a received signal.AMC is a crucial task in many communication systems, including military and civilian applications, where it is necessary to identify the signal modulation type for monitoring, management, and control.As the demand for wireless communication continues to increase, the available spectrum resources are becoming more limited. This has led to the need for more efficient use of the spectrum through the use of mixed-signal modulation. Mixed signal modulation involves the transmission of multiple signals on a single RF frame, which makes the automatic modulation classification of these signals a crucial task.

machine learning (ML) and deep learning (DL) techniques have become increasingly popular and useful in a wide range of fields, including computer vision, healthcare, robotics, and communication systems.Unlike traditional ML algorithms, DL networks can learn from unstructured data without any guidance from expert engineers. This is because DL networks operate like

the human brain, processing data and creating patterns that can be used for decision-making without any human supervision. Dl gives the advantage of automatic feature extraction over ML .As a result, DL networks have become an indispensable tool for solving complex problems in various fields.So here we are using Dl model for AMC.

In this context, we are utilizing Convolutional Neural Networks (CNNs) for modulation classification. CNNs have been found to be a popular choice for this task, likely due to their ability to learn and extract features from complex input data, such as radio frequency signals. Additionally, CNNs may be easier to train than other types of neural networks, as they can be trained in a highly parallelizable manner and can learn from small amounts of labeled data. CNNs have shown promising results in modulation classification, achieving high accuracy rates in various scenarios, including noisy and fading channels. Here we are using ResNet for AMC, Residual Network (ResNet) is a popular deep learning model widely used for computer vision tasks. It is a type of Convolutional Neural Network (CNN) architecture designed to accommodate a large number of convolutional layers.

Deep learning models often require CPU or GPU platforms, which are not ideal for digital signal processing due to their sequential nature and high power consumption. By integrating deep learning models onto FPGA platforms, we can enhance their operability in the signal processing chain of software-defined radios (SDRs) and improve their speed. FPGAs are flexible, powerful, and easily adaptable to new applications, making them an excellent choice for AI. Though they used to be difficult to program, modern tools have made it easier to use FPGAs in AI.

Here we are utilizing VITIS AI tool to implement the ResNet model on

ZCU104. The Vitis AI platform is a complete solution for developing AI models on AMD devices, boards, and Alveo data center acceleration cards. It includes optimized DPU cores, AI models, libraries, tools, and examples for edge and data center AI.

In this study, for the single signal AMC Model, we used the Deepsig 2018 Dataset , which comprises a variety of digital and analog modulation types .Each of the 2,555,904 data inputs in this dataset comprises complex (I Q) data that is 1024 samples long. The dataset provides 26 different Signal-to-Noise Ratio (SNR) values for each signal type, ranging from -20 dB to 18 dB.

In order to evaluate the performance of single-signal AMC, we feed 4000 RF samples that included various modulation schemes and different Signal-to-Noise Ratio (SNR) values. the classification of all 4000RF frames was completed in approximately 2.537 seconds. This test resulted in a frame rate of 1576.54 frames per second, which demonstrates the ability of the ZCU104 board to process a high volume of data efficiently.

For the mixed signal AMC Model, A subset of the original Deepsig 2018 dataset was extracted to create a smaller dataset. The new dataset was analyzed with respect to three fundamental signals, namely OOK, Four-ASK, and QPSK. These signals were subsequently combined to generate a new dataset that consists of six signals: OOK, FOUR_ASK, QPSK, OOK+FOUR_ASK, OOK+QPSK, FOUR_ASK+QPSK.

In order to evaluate the performance of the mixed-signal modulation classification model on ZCU104, we feed 4000 RF frames that included six different signals with mixed signal modulation types with different Signal-to-Noise

Ratio (SNR) values. the classification of all 4000RF samples was completed in approximately 2.535 seconds. This test resulted in a frame rate of 1577.55 frames per second, which demonstrates the ability of the ZCU104 board to process a high volume of data efficiently.

To improve the accuracies lost while quantizing, we finetuned the model and we were able to recover the accuracy by 6%, we implemented the fine-tuned model on ZCU104,we feed 4000 RF frames that included various modulation schemes and different Signal-to-Noise Ratio (SNR) values. the classification of all 4000RF frames was completed in approximately 2.468 seconds. This test resulted in a frame rate of 1620.68 frames per second.

Additionally, the Accuracy for both single-signal and mixed-signal AMC models was also tested on the ZCU104 FPGA board by feeding the 1000RF frames, with that it is clear that the accuracy is increasing with the SNR, at lower SNR accuracy is very low and at higher SNR, accuracy is high.

Finally, we benchmarked both the single-signal and mixed-signal models on CPU,GPU, and FPGA to test the time taken to generate predictions. From the benchmarking results, It has been demonstrated that the performance of FPGA surpasses that of CPU and GPU.

Chapter 6

Thesis stage 2

Hardware acceleration for Online 3D Bin Packing

Chapter 7

Abstract

The task of packing incoming parcels, usually in the form of rigid cuboids, onto a conveyor and fitting them into a larger container for transportation is known as online 3D bin packing, and it is a complex real-time combinatorial optimization problem.

Bin packing problems have been widely studied in computer science, as they have practical applications in the logistics, shipping, and storage industries. The aim is to optimize the use of space and reduce shipping costs by efficiently packing items into containers.

Online 3D bin packing is particularly challenging because it involves making decisions in real-time, as new parcels are continuously arriving on the conveyor. Reinforcement learning algorithms can be employed to optimize the packing process and improve the efficiency of packing. However, these algorithms require significant computational power and can be slow in real-time settings.

FPGA technology has been proposed as a potential solution to improve the performance of reinforcement learning algorithms for online 3D bin packing. By utilizing FPGA hardware, the algorithms can be accelerated, resulting in faster decision-making and better optimization of the packing process[19].

I aim to carry out distributive training for the bin packing problem on the CPU and then evaluate the performance by comparing its execution time with that of the FPGA, as presented in [19]. This comparative analysis would help in identifying the potential benefits of using FPGA technology as a hardware accelerator for the given problem. Additionally, it would enable us to explore the differences in the efficiency and speed of the algorithm between the two platforms, thereby providing insights into their comparative performance.

Chapter 8

Introduction

The concept of 3D bin packing revolves around optimizing the arrangement of boxes or parcels within larger containers, known as Long Distance Containers (LDCs), in order to make the most efficient use of available space. This involves carefully considering the size and shape of both the items being packed and the LDCs themselves, as well as any constraints or requirements related to the packing process. The ultimate goal is to minimize wasted space and maximize the number of items that can be packed within each LDC, thus reducing shipping costs and improving overall efficiency.

Online automated 3D bin packing refers to a dynamic scenario where a robot must assess and place each box in real time. This differs from offline bin packing, where the properties of the entire set of boxes are predetermined. The online nature of this scenario presents several challenges such as the need for fast and accurate decision-making, real-time tracking, and the ability to adapt to unforeseen obstacles.

One of the main advantages of online automated 3D bin packing is its

flexibility. The robot can adjust the packing strategy based on the properties of each box as it is presented, optimizing the use of available space and minimizing waste. This dynamic approach can result in more efficient use of resources and higher productivity.

However, the real-time nature of online automated 3D bin packing also poses several technical challenges. The robot must be able to accurately perceive the dimensions and orientation of each box, as well as the available space in the bin. It must also be able to quickly generate and evaluate possible packing configurations in order to make optimal decisions.

Traditionally, the 3D bin packing problem has been tackled using customized heuristics[16][17]. These techniques involve assessing the suitability of various packing locations for individual boxes by employing a predetermined combination of features and selecting the highest-scoring location for loading. In some cases, the proposed positions are refined by applying additional rules such as extreme-point or interior-point characteristics. Since positions are evaluated one box at a time, these heuristics can be utilized for both offline (all boxes available in advance) and online (boxes arriving in a stream) problems.

However, the main issue with heuristic approaches is the significant amount of design effort required to achieve high-quality solutions, as well as the need for manual tuning of parameters for different size distributions. Moreover, these methods may not always guarantee optimal solutions, and they do not take advantage of the latest advancements in machine learning and optimization techniques.

To overcome these limitations, alternative methods such as meta-heuristics and formal optimization have been explored. Meta-heuristics are general-

purpose algorithms that can be applied to a wide range of optimization problems, including bin packing. These techniques are based on the iterative refinement of candidate solutions and can provide high-quality results, but they may struggle with scalability in large-scale problems, especially in online settings.

Formal optimization, on the other hand, involves mathematically modeling the problem and using advanced optimization techniques to find the optimal solution. These methods can be computationally intensive but can provide optimal solutions for large and complex problems. However, they may require significant domain expertise and computational resources, making them challenging to implement in real-world applications.

Overall, the selection of the appropriate method for solving the 3D bin packing problem depends on the specific requirements of the application, including the size of the problem, the available computational resources, and the desired level of solution optimality. As the field of optimization continues to evolve, it is likely that more advanced and efficient methods will emerge for addressing this challenging problem.

Recently, several studies have investigated the potential of using deep reinforcement learning (Deep RL) to address the Online 3D Bin Packing problem. The general approach of these algorithms involves iteratively training a neural network using a combination of reinforcement learning and deep learning techniques.

The basic workflow of these algorithms is as follows: first, the neural network is initialized with random parameters and then interacts with the environment by selecting actions and receiving rewards. The actions correspond to selecting a box to place and the location to place it in the bin. The

rewards are based on a score that reflects the quality of the current packing configuration. The network uses this feedback to update its parameters, with the goal of maximizing the cumulative reward over the course of the packing process.

One advantage of Deep RL methods is that they can learn to make decisions based on complex and high-dimensional input data, such as the shape and size of each box and the current state of the bin. Additionally, they can adapt their strategies based on the properties of the boxes that are encountered during the packing process.

However, one potential limitation of these methods is the amount of training data required to obtain accurate and effective packing policies. Collecting this data can be challenging in real-world scenarios, and it may not always be possible to simulate all possible combinations of box shapes and sizes.

Despite these challenges, Deep RL approaches hold promise for improving the efficiency and effectiveness of online 3D bin packing in a range of practical applications. Ongoing research in this area is likely to yield even more sophisticated and efficient algorithms in the future.

The bin packing problem can be solved using value-based reinforcement learning algorithms, such as DQN, which can generate satisfactory results within reasonable computation times.In[18] a generalized reinforcement learning algorithm for online 3D bin-packing is implemented.

The utilization of FPGA technology as a hardware accelerator to reduce the inference time of the DQN algorithm, along with its pre and post-processing steps, was suggested in reference [19]. The implementation of this approach resulted in improved efficiency of the algorithm and enabled

it to explore the complete search space within the given time constraints, as discussed in reference [19].

Chapter 9

RL based Online 3D bin packing:

Several recent research studies have suggested employing deep reinforcement learning (Deep RL) to tackle the problem of Online 3D Bin Packing [18]. The general process of these algorithms involves the following steps:

1. identifying suitable positions based on specific criteria
2. creating bin states for the selected positions, either as images or processed features
3. assessing the shortlisted inputs in a batch using a Deep RL approach such as DQN
4. selecting the option with the highest anticipated reward and repeating the process.

To reduce the overall training time and reduce computation time during inference, the initial stage of the algorithm utilizes heuristics similar to those mentioned previously to narrow down the search space. However, this can result in suboptimal solutions. The first two steps in the process involve a significant amount of iterative computation for the search and pre-processing functions. In the third step, a deep learning model is employed to perform inference on a large batch of inputs.

9.1 Algorithm :

The algorithm comprises of five distinct steps: State and Environment, Feasibility Mask, Feature Generation, DQN Batch Inference, and Box Placement.

state and Environment: In the context of bin packing, the objects that need to be packed are referred to as Long Distance Containers (LDCs), and the environment in which they are packed is an LDC with dimensions of Length (L) = 80cm, Breadth (B) = 45cm, and Height (H) = 45cm. To represent the state of bin packing, a top-down view of the LDC is taken, which is represented by a rectangular grid or a 2D array of dimensions (80x45). Each cell in the grid represents a unit area and the value of the cell increases by the height of each box placed on it. When a box is placed on the grid, its top-left corner is placed at a specific coordinate on the grid.

Feasibility Mask : To create a feasibility mask, the process involves examining every coordinate within the LDC (45 x 80 = 3600), and selecting only those that are feasible. Since the purpose of this acceleration technique is to examine as many positions as possible, this stage performs the essential

checks to ensure that a box can be placed physically at a given coordinate. To qualify for the next step, coordinate must pass two tests, and only those that meet these criteria will be included in the shortlist.

The first test is a Dimension Eligibility Check, which involves comparing the size of the box to the LDC boundaries to ensure it fits within them when placed at the specific coordinate. The second test is a Surface Eligibility Check, which evaluates the stability of the surface where the box might be placed if the particular coordinate is chosen. This test involves multiple conditional checks to verify that the surface is smooth or has an acceptable level of unevenness. **Feature Generation stage:** a set of three features is created for a specific potential coordinate (i,j) and a given box (l,b,h). Each feature represents a different aspect of the state of the LDC if the particular coordinate is chosen.

One of the three features generated in the Feature Generation stage is the LDC State (I1(i,j)), which is a flattened and normalized version of the LDC array projection when the box is placed at the (i,j) coordinate. The length of this feature is 3600, corresponding to the 45x80 dimensions of the LDC array.

Another feature generated in the Feature Generation stage is the Border State (I2(i,j)), which is an array containing all the LDC cell values (heights) along the perimeter of the box if it is placed at the (i,j) coordinate. The maximum length of this feature is 254, corresponding to the perimeter of the largest possible box (80 + 80 + 45 + 45 + 4).

The third feature generated in the Feature Generation stage is the Aggregate State (I3(i,j)), which is an array of length 9. It comprises of several

parameters that are crucial for efficient packing, such as the flatness of the surface, the proximity of the box with adjacent boxes on all sides, and other similar factors.

DQN Batch inference :

In the DQN Batch Inference step, the three input vectors generated in the previous Feature Generation step are fed into a feed-forward network. This network comprises of 7 dense layers, as illustrated in Figure 9.1[19], and is used to compute a Q-value for each of the potential coordinates.

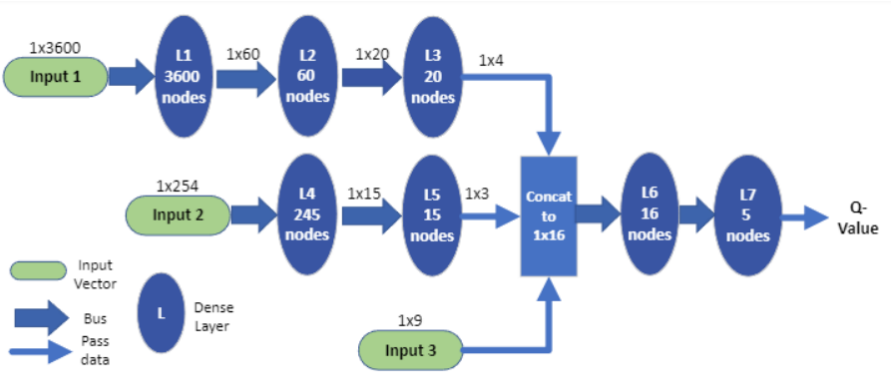

Figure 9.1: DQN architecture [19]

In each of the 7 dense layers in the feed-forward network, an input vector (Input) is multiplied with a weight matrix (Weight). The resulting product is added to a bias vector (Bias), and the resulting sum is then passed through the Tanh activation function (Eqn 1). The weight and bias values used in the layers are 32-bit floating point numbers.

In the Python implementation of the Bin-Packing algorithm, instead of performing a single inference at a time, this step is carried out in batches. This means that the input features for all feasible coordinates are combined and sent as a batch through the DQN to generate the corresponding Q-values.

Once all the feasible coordinates have been evaluated and their respective Q-values generated, the coordinate with the highest Q-value is selected as the optimal location for placing the box. The heights of the corresponding cells in the LDC grid are then upgraded accordingly. Moreover, a list containing the coordinates of corners (as indicated by Feature 3) is updated each time a new box is placed.

9.2 FPGA acceleration for Bin Packing Problem

Von Neumann architecture, which is used in CPU implementations, has limitations in optimizing the operations involved in bin packing because it cannot fully leverage parallel processing. In contrast, spatial architectures such as FPGA are more effective in utilizing pipelining and parallelism to speed up these operations.

The goal of [19] is to enhance the efficiency of an online 3D bin packing algorithm proposed in [18], using RL, while minimizing the impact on latency. This is achieved by attempting to accelerate the algorithm.

The bpp algorithm involves operations that require processing multiple elements of the LDC, which is a data structure resembling a matrix. On a CPU, these operations (performed to place a single box) can create a bottleneck in bin packing since they take longer than the DQN itself. Each dense layer of the DQN involves a vector matrix multiplication, which is an operation that can be optimized by an FPGA. The conditional complexity of the LDC operations may lead to divergent thread execution on a GPU,

resulting in low efficiency. Therefore, FPGA was considered to be a more suitable option for attempting to accelerate the algorithm.to accelerate the RL algorithm on FPGA we are using a tool called VITIS HLS.

9.3 VITIS HLS Tool :

Vitis HLS (High-Level Synthesis) is a tool developed by Xilinx that allows designers to use high-level languages such as C, C++, and SystemC to describe hardware functionality. The tool automatically generates RTL (Register Transfer Language) code from the high-level description, which can be synthesized into FPGA or ASIC hardware.

Vitis HLS provides designers with a way to efficiently implement complex algorithms and functions in hardware without having to manually write RTL code. This significantly reduces the development time for hardware and allows for faster iteration and design exploration. Moreover, it enables the designer to optimize the design for specific hardware platforms, resulting in more efficient and better-performing hardware designs.

Overall, Vitis HLS provides an efficient way to design and implement custom hardware solutions using high-level languages and is widely used in the FPGA and ASIC design industry.

To accelerate the bpp algorithm using an FPGA, Here we are using a High-Level Synthesis tool[19]. This tool utilizes functions written in high-level languages like C and C++ to generate synthesizable RTL (Register Transfer Language) code, which can then be implemented onto the FPGA. By doing so, it provides the designer with control over the resources and

implementation methods employed. This level of control allows for the implementation of the entire bin-packing algorithm as a continuous process, rather than just accelerating the DQN. This approach can result in a significant reduction in the time it takes to complete the bin packing process, leading to a more efficient implementation.

9.4 Alveo U280 Data Center Accelerator Card

Our future plan is to deploy the model on the Alveo U280 Data Center Accelerator Card[19].

The AMD Alveo U280 Data Center accelerator card is a cutting-edge solution designed to meet the ever-evolving needs of modern data centers. It is built on the advanced AMD 16nm UltraScale+ architecture and boasts 8GB of HBM2 memory with a bandwidth of 460 GB/s. This configuration enables the Alveo U280 to deliver exceptional performance and adaptable acceleration for compute-intensive, memory-bound applications such as analytics, machine learning inference, and database operations.

Moreover, the Alveo U280 accelerator card is equipped with PCI Express 4.0 support, which leverages the latest server interconnect infrastructure to deliver high-bandwidth connectivity with host processors.

One of the most significant advantages of the Alveo accelerator cards is their adaptability to changing acceleration requirements and algorithm standards. They can accelerate any workload without requiring hardware changes, thus reducing the overall cost of ownership for data center operators.

These production Alveo U280 Data Center Accelerator cards are readily available for volume deployment in data centers or for application development and prototyping. The U280 active cards come with a USB connector that facilitates development tasks for engineers and software developers.

Overall, the Alveo U280 Data Center Accelerator card is an excellent choice for businesses looking to improve their data center performance, reduce costs, and stay ahead of the rapidly changing technological landscape.

9.5 Acceleration of Bin packing on FPGA

Due to the lack of Python support in Vitis HLS, it was necessary to reprogram the original algorithm using C programming language and basic libraries. In a research paper[19], the bin packing algorithm is implemented on an alveo U280 Data center accelerator card mounted on a 56-core CPU with hyperthreading, and 256 GB memory was used to deploy the Bin packing problem. Upon evaluating all 256 boxes from the dataset, the maximum and average values indicate the maximum and average duration required to evaluate a single box. The average duration required to evaluate a single box is 0.014sec. The maximum duration required to evaluate a single box is 0.145 sec.

9.6 Distributive Training of RL-Based Bin Packing Algorithm on CPU

The complete dataset was partitioned into 1 bin data files, and the primary objective is to parallelize the episodes across different CPUs to optimize performance. The main goal is to compare the results of the distributed training on CPU with the FPGA implementation results. The focus is on evaluating the effectiveness of the distributed training approach in improving performance compared to the FPGA implementation.

To achieve this, the episodes are being executed in parallel across multiple CPUs, leveraging their collective computational power. This parallelization strategy aims to expedite the learning process, enhance exploration capabilities, and potentially yield improved policy convergence for the RL-based bin-packing algorithm.

The intention behind comparing the distributed training results on the CPU with the FPGA implementation is to assess the relative performance and efficiency of these two approaches. By conducting this comparison, it becomes possible to analyze the trade-offs, advantages, and limitations of each implementation method in the context of the bin-packing problem.

Overall, the research aims to provide insights into the efficacy and potential benefits of distributed training on CPUs compared to FPGA implementations, contributing to the advancement of efficient and scalable solutions for solving the bin packing problem.

Chapter 10

Conclusion for Stage 2 :

Online 3D bin packing is a complex combinatorial optimization problem that involves packing incoming parcels in real time onto a conveyor and fitting them into a larger container for transportation. It requires constant decision-making as new parcels continuously arrive on the conveyor. Reinforcement learning algorithms such as DQN can be used to optimize the packing process. paper [19] proposed FPGA technology as a hardware accelerator to reduce the inference time of the DQN algorithm and its pre- and post-processing steps. My main goal is distributive training of this bin-packing problem on the CPU, To achieve this, the episodes are being executed in parallel across multiple CPUs.The intention behind comparing the distributed training results on the CPU with the FPGA implementation is to assess the relative performance and efficiency of these two approaches. By conducting this comparison, it becomes possible to analyze the trade-offs, advantages, and limitations of each implementation method in the context of the bin-packing problem.

Bibliography

[1] T. J. O'Shea, T. Roy and T. C. Clancy, "Over-the-Air Deep Learning Based Radio Signal Classification," in IEEE Journal of Selected Topics in Signal Processing, vol. 12, no. 1, pp. 168-179, Feb. 2018, doi: 10.1109/JSTSP.2018.2797022.

[2] Sarker, I.H. Machine Learning: Algorithms, Real-World Applications and Research Directions. SN COMPUT. SCI. 2, 160 (2021). https://doi.org/10.1007/s42979-021-00592-x

[3] Sarker, I.H. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications, and Research Directions. SN COMPUT. SCI. 2, 420 (2021). https://doi.org/10.1007/s42979-021-00815-1

[4] Rapuano E, Meoni G, Pacini T, Dinelli G, Furano G, Giuffrida G, Fanucci L. An FPGA-Based Hardware Accelerator for CNNs Inference on Board Satellites: Benchmarking with Myriad 2-Based Solution for the CloudScout Case Study. Remote Sensing. 2021; 13(8):1518. https://doi.org/10.3390/rs13081518

[5] Wang C, Luo Z. A Review of the Optimal Design of Neural Networks Based on FPGA. Applied Sciences. 2022; 12(21):10771. https://doi.org/10.3390/app122110771

[6] Wu R, Guo X, Du J, Li J. Accelerating Neural Network Inference on FPGA-Based Platforms—A Survey. Electronics. 2021; 10(9):1025. https://doi.org/10.3390/electronics10091025

[7] H. den Boer, R. W. D. Muller, S. Wong and V. Voogt, "FPGA-based Deep Learning Accelerator for RF Applications," MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM), San Diego, CA, USA, 2021, pp. 751-756, doi: 10.1109/MILCOM52596.2021.9652891.

[8] R. R. Yakkati, R. R. Yakkati, R. K. Tripathy and L. R. Cenkeramaddi, "Radio Frequency Spectrum Sensing by Automatic Modulation Classification in Cognitive Radio System Using Multiscale Deep CNN," in IEEE Sensors Journal, vol. 22, no. 1, pp. 926-938, 1 Jan.1, 2022, doi: 10.1109/JSEN.2021.3128395.

[9] Xu, Jiahao and Zihuai Lin. Modulation and Classification of Mixed Signals Based on Deep Learning. (2022). https://doi.org/10.48550/arxiv.2205.09916

[10] Q. Gao, S. Huang, L. Wang, K. Wang, Y. Zhang and Z. Feng, "Modulation classification of mixed signals using independent component analysis," 2015 IEEE/CIC International Conference on Communications in China (ICCC), Shenzhen, China, 2015, pp. 1-5, doi: 10.1109/ICCChina.2015.7448675.

[11] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, 2015. arXiv: 1512.03385 [cs.CV].

[12] https://www.deepsig.ai/datasets.

[13] Vitis ai zynq dpu v3.3 product guide, 2021. [Online]. Available: https://docs.xilinx.com/r/3.3-English/pg338-dpu/DPU-Development-Flow

[14] https://www.logic-fruit.com/blog/fpga/fpga-design-architecture-and-applications/

[15] https://repository.tudelft.nl/islandora/object/uuid%3Ae2c82316-8f35-4e19-b5ab-29e2dbf93a30

[16] Teodor Gabriel Crainic, Guido Perboli, and Roberto Tadei. 2008. Extreme point-based heuristics for three-dimensional bin packing. Informs Journal on computing 20, 3 (2008), 368–38

[17] Silvano Martello, David Pisinger, and Daniele Vigo. 2000. The three-dimensional bin packing problem. Operations research 48, 2 (2000), 256–267.

[18] Richa Verma, Aniruddha Singhal, Harshad Khadilkar, Ansuma Basumatary, Siddharth Nayak, Harsh Vardhan Singh, Swagat Kumar, and Rajesh Sinha. 2020.A generalized reinforcement learning algorithm for online 3d bin-packing. In Generalisation in Planning Workshop at AAAI.

[19] "Performance improvement of reinforcement learning algorithms for online 3D bin packing using FPGA".Authors: Kavya Borra (TCS)*; Ashwin Krishnan (TCS Research); Harshad Khadilkar (TCS Research); Manoj Nambiar (TCS); Ansuma Basumatary (TCS); Rekha Singhal (TCS); Arijit Mukherjee (TCS Research). https://www.aimlsystems.org/2022/industry_track