

Simulation-based RISC-V DSP Accelerator Verification and FPGA-Based Data Transmission on ZCU104 Platform

Xiaojing Han

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of
Science in Technology.

Espoo 31.12.2024

Supervisor

D.Sc Marko Kosunen

Advisor

M.Sc Otto Simola, M.Sc Alekski
Korsman

Copyright © 2024 Xiaojing Han

Author Xiaojing Han

Title Simulation-based RISC-V DSP Accelerator Verification and FPGA-Based Data Transmission on ZCU104 Platform

Degree programme Electronics and Nanotechnology

Major Micro- and Nanoelectronic Circuit Design **Code of major** ELEC3036

Supervisor D.Sc Marko Kosunen

Advisor M.Sc Otto Simola, M.Sc Aleksi Korsman

Date 31.12.2024 **Number of pages** 67 **Language** English

Abstract

This thesis focuses on the verification of a RISC-V DSP accelerator with A-Core processor, emphasizing functional verification through simulation, and the verification of FPGA-based data transfer on the ZCU104 platform. The primary goal is to ensure signal processing accuracy and system functionality, providing a detailed account of the verification processes while addressing practical challenges. The reliability of the RISC-V DSP accelerator with A-Core processor for future use in contemporary communication systems is confirmed by this work through the analysis of important performance indicators and the documentation of debugging methods.

The functional verification of the DSP accelerator was conducted in a simulation environment, with metrics such as Error Vector Magnitude (EVM), Power Spectral Density (PSD), and Adjacent Channel Leakage Ratio (ACLR) evaluated to assess signal fidelity and performance. These analyses highlighted the effects of specific design elements, including the inclusion of the Up Rate Converter (URC), on overall system behavior. This work provides valuable insights into the DSP accelerator's functionality and forms a foundation for its potential integration into hardware platforms.

In addition, this thesis verifies data transfer within the FPGA-based platform, leveraging the A-Core processor and the ZCU104 board. With the FPGA implementation developed by the Electronic Circuit Design (ECD) group, this work documents the challenges encountered during the verification process, including open-source setup issues, configuration mismatches, and debugging errors. Practical solutions to these challenges are detailed, offering a reference for future researchers working on similar projects.

The results of this thesis demonstrate the effectiveness of the verification methodologies employed, ensuring the functional correctness of the DSP accelerator and validating FPGA-based data transfer. This thesis provides a solid foundation and direction for further research on FPGA verification of the DSP accelerator. The systematic approach, insights, and practical methods documented here contribute to advancing RISC-V-based DSP accelerator design, ensuring performance reliability and scalability in communication systems.

Keywords RISC-V, DSP, accelerator, A-Core, verification, simulation, FPGA

Contents

Abstract	iii
Contents	iv
Symbols and abbreviations	vi
1 Introduction	1
2 Background	3
2.1 RISC-V Architecture	4
2.2 Digital Signal Processor	5
2.2.1 Signals and Systems	6
2.2.2 Core DSP Operations	7
2.2.3 DSP Workflow	10
2.3 DSP Accelerator: Quadrature Amplitude Modulation (QAM) Technology	12
2.3.1 Realization of QAM	12
2.3.2 Graphical Representation of QAM	14
2.3.3 Application of QAM	17
2.3.4 DSP Accelerator Application: 5G Transmission System	18
2.4 Signal Transmission Performance Metric	19
2.4.1 Error Vector Magnitude (EVM)	20
2.4.2 Power Spectral Density (PSD)	22
2.4.3 Adjacent Channel Leakage Ratio (ACLR)	23
2.5 Transmitter (TX) Chain	24
2.6 Receiver (RX) Chain	25
2.7 Verification Method	26
2.7.1 FPGA Verification	26
2.7.2 ASIC Verification	29
2.7.3 Functional Verification	30
3 Introduction of the FPGA board ZCU104	32
3.1 ZCU104 Board	32
3.2 A-Core Structure	34
3.3 TheSyDeKick	36
3.4 A-Core Signal Transition Flow With DSP Accelerator	37
4 Verification	39
4.1 Verification Flow Overview	39
4.2 Verification of FPGA Implementation	40
4.2.1 Open Source Version Setup	40
4.2.2 Modification of Vivado Design	41
4.2.3 Key Components in The Design	41
4.2.4 Bitstream Generation Challenges	42

4.2.5	Modifying Source Code and USB Issues	42
4.2.6	Configuration	44
4.2.7	Output Verification	44
4.3	Functional Verification of DSP Accelerator	45
4.3.1	Open Source Version Setup	45
4.3.2	Simulation Command	47
4.3.3	Java.io.IOException	47
4.3.4	Issues With External Source	48
4.3.5	Test Fail Troubleshooting	49
4.3.6	Outcome and Verification Continuation	50
5	Result and Analysis	51
5.1	DSP Chain without URC	51
5.1.1	EVM	51
5.1.2	PSD	52
5.1.3	ACLR	53
5.2	DSP Chain with URC	54
5.2.1	EVM	55
5.2.2	PSD	56
5.2.3	ACLR	57
5.3	Comparison and Analysis of Results with and without URC	58
5.3.1	Results Comparison	58
5.3.2	Analysis of Impacts	58
5.4	Summary	59
6	Conclusion	60
	References	61

Symbols and abbreviations

Abbreviations

5G	Fifth-Generation
ADAS	Advanced Driver-Assistance Systems
AI	Artificial Intelligence
AR	Augmented Reality
ARM	Advanced RISC Machine
ASIC	Application-Specific Integrated Circuit
ASICs	Application-Specific Integrated Circuits
CIC	Cascaded Integrator-Comb
CORDIC	Coordinate Rotation Digital Computer
CPU	Central Processing Unit
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DOCSIS	Data Over Cable Service Interface Specification
DSP	Digital Signal Processor
ESA	European Space Agency
EVM	Error Vector Magnitude
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
GPIO	General-Purpose Input/Output
ISA	Instruction Set Architecture
LO	Local Oscillator
ML	Machine Learning

NASA	National Aeronautics and Space Administration
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
RISC-V	Reduced Instruction Set Computer V
SNR	Signal-to-noise ratio
TheSyDeKick	The System Design Kit
UART	Universal Asynchronous Receiver/Transmitter
URC	Up Rate Converter
Wi-Fi	Wireless fidelity

Symbols

$\cos(2\pi f_c t)$	Cosine orthogonal carrier signal
$\sin(2\pi f_c t)$	Sine orthogonal carrier signal
θ	Phase
A	Amplitude
f_c	Carrier frequency
$I(t)$	In-phase signal
$Q(t)$	Quadrature signal
$s(t)$	QAM modulated signal

1 Introduction

In recent years, modern electronic devices have become indispensable in daily life. From smartphones to supercomputers, processors act as critical components that drive computation and functionality. At the heart of processor design lies the Instruction Set Architecture (ISA), which defines the language that computers understand. Similar to how human language facilitates communication, ISA consists of a set of instructions that dictate execution flows and operations, such as mathematical calculations, logical functions, and data transfers [1]. Popular ISAs such as Advanced RISC Machine (ARM) and Intel x86 have long dominated the market, setting benchmarks in the fast-evolving ultramobile device industry [2]. However, these proprietary ISAs come with notable limitations. For instance, ARM's lower peak performance limits its suitability for high-performance tasks, while x86's higher power consumption reduces its compatibility with mobile devices. Furthermore, the complexity of x86 designs increases manufacturing costs and error risks [3]. Additionally, both ISAs are tightly controlled, requiring high licensing fees and offering limited flexibility for customization or innovation.

To address these limitations, open-source hardware architectures have gained significant attention due to their adaptability, flexibility, and cost efficiency. Among these, the Reduced Instruction Set Computer V (RISC-V) has become a popular ISA. Unlike proprietary ISAs, RISC-V has no licensing fee and is freely available, offering designers the flexibility to implement and extend its architecture without licensing constraints. This makes it an ideal platform for both academic research and industrial applications [4]. For example, in Artificial Intelligence (AI) and Machine Learning (ML), RISC-V supports custom extensions for specialized workloads, enhancing efficiency and reducing costs. Similarly, RISC-V's open-source nature and flexibility have expanded its application in industries like automotive (e.g., Advanced Driver-Assistance Systems (ADAS)), in-car entertainment, and power control. Furthermore, organizations such as NASA (NASA) and the European Space Agency (ESA) are exploring the use of RISC-V in satellites, avionics systems, and other aerospace applications due to its power efficiency and long lifespan [5].

Modern communication systems, particularly those supporting 5G technology, demand highly efficient and reliable hardware solutions. The RISC-V DSP accelerator, integrated into a transceiver system, plays a crucial role in meeting these demands by enabling advanced signal processing capabilities [6].

However, the increasing complexity of RISC-V implementations including the application of DSP accelerator demands robust verification methodologies to ensure functional correctness and system reliability [7]. Verification is crucial to identify and eliminate functional and integration errors while meeting performance requirements. Customized verification approaches are developed for specific platforms, including simulation-based functional verification, Application-Specific Integrated Circuit (ASIC) and Field-Programmable Gate Array (FPGA) platforms [8].

This thesis focuses on the verification of the RISC-V DSP accelerator and related FPGA-based implementations within a 5G transceiver system. Exploring upon the existing designs and frameworks developed by the Electronic Circuit Design (ECD)

group at Aalto University [9], the work emphasizes functional verification of the DSP accelerator through simulation and verification of data transfer within an FPGA-based platform implemented on the ZCU104 board. By addressing both functional and hardware-specific challenges, this thesis aims to enhance the understanding and reliability of the verification processes.

The functional verification of the DSP accelerator was conducted through simulation, focusing on key performance metrics [10] such as Error Vector Magnitude (EVM), Power Spectral Density (PSD), and Adjacent Channel Leakage Ratio (ACLR). These metrics were analyzed to evaluate the signal processing capabilities of the DSP accelerator and to understand the effects of specific design choices, such as the inclusion of the Up Rate Converter (URC). This analysis highlights the DSP accelerator's suitability for 5G applications and forms a critical foundation for future research and development [7].

Additionally, this thesis documents the verification of FPGA-based data transmission within the ZCU104 platform. With the FPGA implementation and associated verification environment developed by the ECD group in Aalto University, this thesis provides a detailed description of the verification processes, including addressing challenges such as open-source setup issues, configuration mismatches, and debugging errors. By showing systematic solutions to these challenges, this thesis offers practical guidance for future researchers and developers.

The remainder of this thesis is structured as follows: Chapter 2 provides background information on RISC-V architecture, digital signal processor and its accelerator, signal transmission performance metrics, transmitter and receiver chains, and existing widely used verification methodologies. Chapter 3 describes the verification framework proposed by ECD group in Aalto University, including tool integration, module design, and practical implementation steps. Chapter 4 demonstrates the verifications based on the framework and shows how to fix common errors, making it easier for future researchers to run the process. Chapter 5 presents and analyzes the results of the verification and simulation, highlighting the performance of different digital signal processing chain. Finally, Chapter 6 concludes the thesis by summarizing its contributions, discussing limitations, and proposing future work.

2 Background

The development of the Fifth-Generation (5G) networks is leading the rapid advancement of wireless communication technologies in recent years. 5G networks have a better performance in reducing latency and improving connectivity [11], which promise higher data rates, low latency, and stable connectivity. 5G environment is most demanding in the factors of high efficiency need and the reliability of hardware systems which is critical when it is utilized to meet strict requirements [12]. Among all the existing hardware platforms, the RISC-V architecture has obtained magnificent attention which is due to the open-source, flexibility, and costamization of the RISC-V architecture. This also enables RISC-V to be utilized in various applications [13].

The verification of hardware designs plays an essential role in ensuring functional correctness. Especially the ones that based on RISC-V performance reliability and with 5G transceiver integration systems. With the increasing complexity of RISC-V implementations, traditional verification approaches have evolved, integrating advanced methodologies and tools to address the unique challenges posed by modern designs. FPGAs have been developed as a magnificent platform for prototyping and verifying these designs, offering rapid iteration and real-world testing capabilities [14].

This chapter introduces and explores the generic concepts underlying RISC-V architectures, Quadrature amplitude modulation, 5G transmission systems and the evaluation metrics of it, detailed construction of transmitter chain and receiver chain, and two different FPGA-based verification strategies.

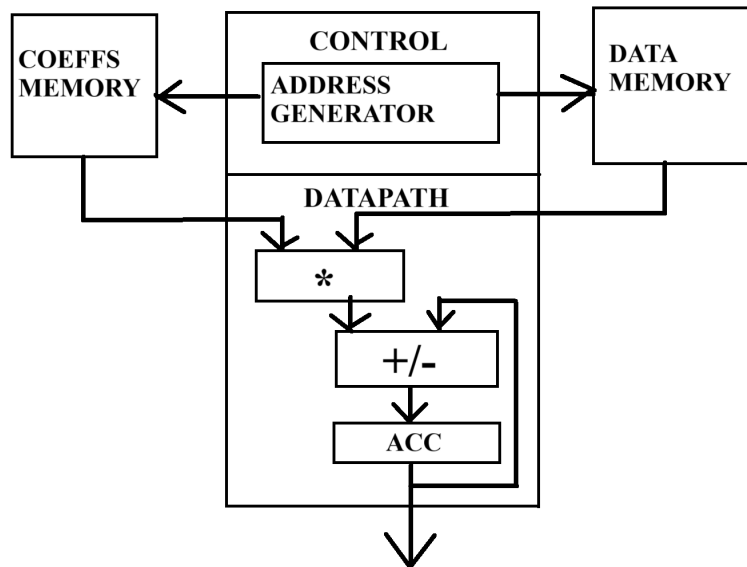


Figure 1: Generic block diagram of a DSP [15]

2.1 RISC-V Architecture

RISC-V is an open-source instruction set architecture (ISA) that has gained widespread attention in academia and industry for its simplicity, modularity, and scalability. Initially developed at the University of California, Berkeley, RISC-V provides a clean-slate design that addresses many of the limitations of legacy ISAs, such as proprietary restrictions and excessive complexity [13]. Its open-source feature enables free use, extension, and customization, making it particularly appealing for research and commercial applications.

One of the defining features of RISC-V is its modularity. The ISA is built around a minimal base instruction set, supplemented by a series of optional extensions, including:

M: Multiplication and Division instructions.

A: Atomic operations for concurrency support.

F/D: Single- and Double-Precision Floating-Point instructions.

V: Vector operations for high-performance computing [16].

This structure enables designers to select only the components needed for a specific application, ranging from low-power microcontrollers to high-performance processors. Additionally, RISC-V allows extensions to be customized, facilitating innovations tailored to specialized domains such as machine learning or cryptography [17].

Another critical advantage of RISC-V is its simplicity. The ISA is intentionally designed to minimize the complexity of hardware implementation, with a focus on reducing the number of instructions and streamlining encoding. This simplicity not only eases implementation but also makes RISC-V cores easier to verify—a crucial consideration in hardware development [18].

The rapid adoption of RISC-V has been driven by a robust ecosystem of open-source tools and projects. For example, the Rocket Chip generator provides a flexible framework for generating RISC-V cores with various configurations [19]. OpenTitan, an open-source silicon project, demonstrates the viability of RISC-V for secure hardware applications [20]. Software support for RISC-V is extensive, with compatibility across popular compilers such as GCC and LLVM, as well as operating systems like Linux.

digital form, advanced algorithms are employed to modify, filter, or enhance the signal according to the different needs of the application [6].

DSP plays a significant role in modern technology. It has become inevitable due to its precision, flexibility, and efficiency in processing signals compared to traditional analog methods. Analog signal processing is extremely vulnerable to noise and interference, which can degrade the quality of the signal over time or distance. This degradation especially causes problems in the systems that requiring high fidelity or long-term stability. DSP, on the other hand, transforms analog signals into digital form so that they can be handled as discrete numbers. Even in noisy environments, digital signals are much less likely to degrade, which highly guarantee more precise and reliable processing. This makes DSP a recommended option in applications that call for signal manipulation with high quality.

The performance of DSP largely depend on the manipulation of digital signals as well as the processing systems. The following sections introduce some fundamental concepts that form the basis of DSP, including signals and systems, core operations, and the process of sampling and quantization.

2.2.1 Signals and Systems

In DSP, a signal is a representation of data that varies with time or another independent variable. A digital signal is obtained by sampling an analog signal and representing it as a sequence of discrete numerical values, as depicted in Figure 3. These digital signals are less sensitive to noise and interference. As a result, they are easier to process, store, and transmit compared to their analog counterparts. The digital signals are often represented mathematically as sequences, such as in form of $x[n]$, where n denotes the discrete time index [10].

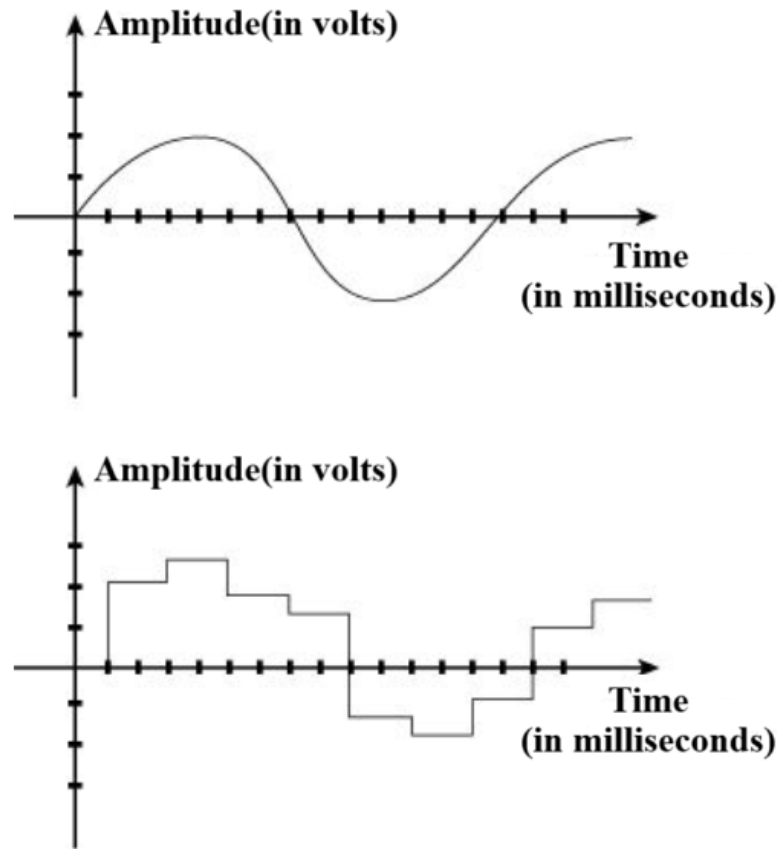


Figure 3: Comparison of an analog signal and its sampled digital representation¹

The systems that process these signals are referred to as discrete-time systems. Unlike analog systems, discrete-time systems work with sequences of discrete values instead of continuous signals. These systems are defined by mathematical models, often involving operations such as addition, multiplication, and shifts in the signal. The discrete nature of these systems enables precise control over the signal processing operations, making them more adaptable and reliable than analog systems in most modern applications.

2.2.2 Core DSP Operations

DSP employs a range of fundamental operations to analyze, modify, and interpret signals. One of the most common operations is filtering, which is used to remove unwanted components from a signal, such as noise, or to emphasize specific features of interest. Filters can be implemented digitally with high precision, enabling applications such as noise cancellation in audio processing or interference suppression in communication systems.

¹<https://learn.andoyaspace.no/ebook/the-cansat-book/common/getting-started/using-the-sensors/>

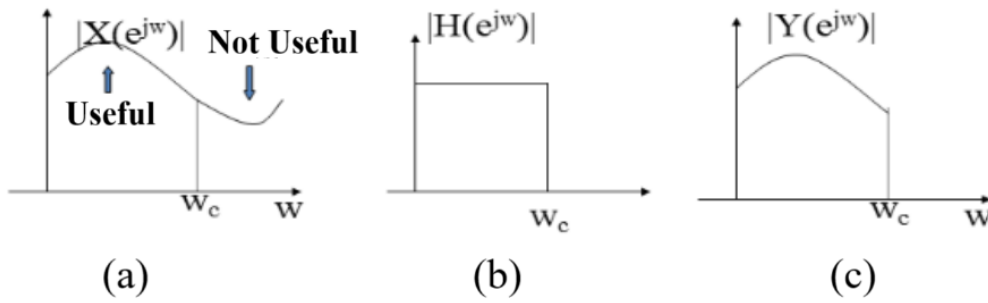


Figure 4: Visualization of the filtering process: (a) input signal before filtering, (b) frequency response of the digital filter, and (c) output signal after filtering.²

The figure 4 demonstrates the process of digital filtering through three plots: the input signal, the filter's frequency response, and the output signal after filtering. The (a) plot shows the input signal, which contains both desired components and unwanted noise. The (b) plot presents the frequency response of the digital filter, illustrating how specific frequencies should be filtered or preserved. The filter's design, such as low-pass, high-pass, or band-pass, determines its effect on the signal. The (c) plot shows the output signal after filtering. The filter removes noise or interference, resulting in a expected signal. This process shows the effectiveness of digital filtering in improving signal quality which is applied in noise reduction and communication systems.

Another important operation in DSP processing is the Fourier transform. Fast Fourier Transform (FFT) is a widely-used efficient algorithm in fields of modern DSP systems which is introduced by Cooley and Tukey in 1965 [23]. This technique converts signals from the time domain to the frequency domain. This transformation enabling applications such as spectrum analysis and signal modulation by providing information of the frequency components of a signal. FFT is applied to calculate the discrete Fourier transform (DFT) due to its efficiency. The FFT reduces the computational complexity from $O(N^2)$ in computation of the DFT to $O(N \log N)$ [23]. This improvement makes the FFT particularly valuable for real-time DSP applications, ensuring the fast and efficient processing.

²<https://cloud.tencent.com/developer/article/1847740>

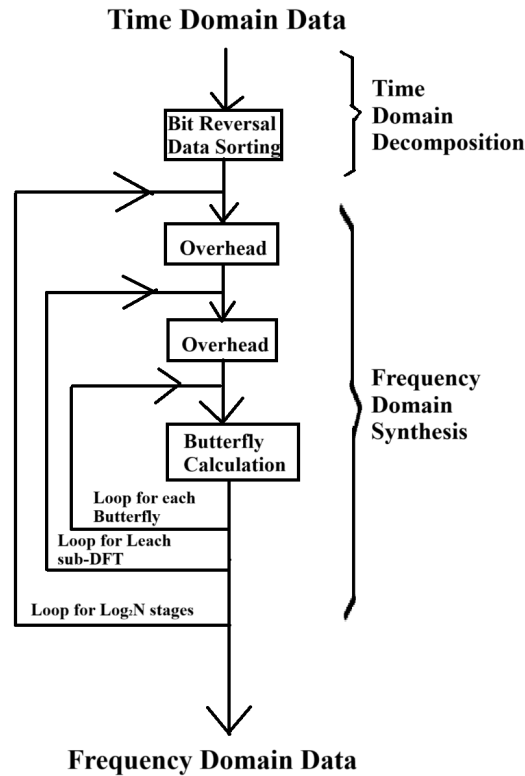


Figure 5: FFT process flow block diagram

As mentioned by S. W. Smith [24], the process flow of FFT can be illustrated by the diagram Figure 5. The FFT process mainly consists of two stages: time domain decomposition and frequency domain synthesis. Each of the stages includes specific computational steps.

In the time domain decomposition stage, the input signal is first sorted using bit-reversal data sorting. This operation reorders the data points into a specific sequence required for the FFT computation. By arranging the data in this way, the algorithm can minimize redundant calculations, making the following steps more efficient.

The frequency domain synthesis stage involves iterative calculations. Initially, overhead operations are performed to manage the structure of the algorithm. The most critical part of this stage is the butterfly calculation, where pairs of data points are combined through addition and subtraction. This operation is repeated in multiple stages, progressively transforming the data into its frequency-domain representation. The iterative process operates in $\log_2 N$ stages, where N is the total number of data points, highlighting the efficiency of the FFT compared to the traditional DFT.

The diagram demonstrates how the FFT achieves computational efficiency by breaking down the transformation into manageable steps. This structured process reduces the computational complexity, making the FFT a fundamental tool in digital signal processing applications such as spectrum analysis and signal modulation.

The FFT is widely used in modern DSP systems across diverse fields. In audio processing, it is employed for tasks such as equalization, noise reduction, and pitch analysis. In telecommunications, it enables efficient spectrum analysis and signal modulation, contributing to the development of high-speed communication systems. Its applications also extend to image processing, where it is used for feature extraction and filtering in the frequency domain [24].

The development of the FFT marked a significant advancement in DSP, making complex frequency-domain operations computationally feasible. Its versatility and efficiency have cemented it as a cornerstone of modern DSP techniques [23].

Convolution and correlation are also fundamental DSP operations. Convolution is used to determine the output of a system given an input signal and its impulse response, making it essential in system analysis and filter design. Correlation, on the other hand, measures the similarity between two signals, and it finds applications in pattern recognition, signal matching, and time delay estimation in communication systems.

These operations form the backbone of many DSP applications, from audio and image processing to biomedical signal analysis and wireless communication. Their ability to extract, transform, and enhance information from signals makes them indispensable in modern technology.

2.2.3 DSP Workflow

To process analog signals digitally, they must first be converted into digital form through a process known as sampling and quantization. From which, Sampling involves measuring the value of an analog signal at regular intervals, creating a sequence of discrete samples. The Nyquist sampling theorem plays a crucial role in this process, stating that the sampling rate must be at least twice the highest frequency present in the analog signal to avoid aliasing. When higher frequency components of the signal overlap and distort the sampled data, aliasing occurs, and leading to inaccuracies in digital representation. Figure 6 depicts a block diagram of DSP typical flow.

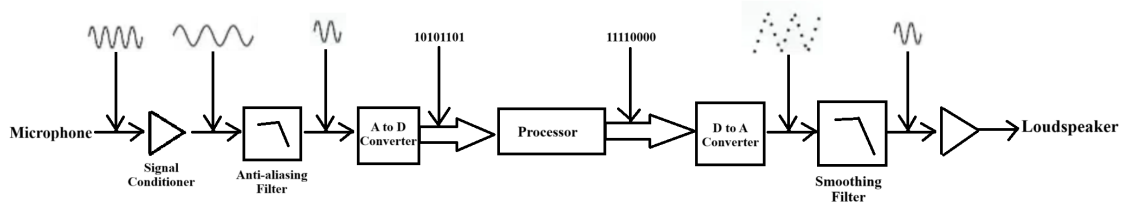


Figure 6: DSP block diagram [25]

Quantization follows sampling and involves approximating the continuous range of the signal's amplitude values with a finite set of levels. This step introduces a small error known as quantization noise, which can be decreased by increasing the number

of quantization levels. Together, sampling and quantization convert an analog signal into a digital format that can be processed by DSP systems.

The significance of the Nyquist theorem cannot be overstated, as it ensures that no critical information is lost during the sampling process, preserving the fidelity of the original signal. This principle underpins the success of digital communication systems, audio recording, and various other applications where accurate signal representation is paramount.

In summary, these key concepts in DSP—signals and systems, fundamental operations, and sampling and quantization—form the foundation of modern signal processing techniques. Their applications span numerous fields, enabling advancements in communication, healthcare, multimedia, and beyond.

Another major advantage of DSP is its flexibility. Analog systems often rely on dedicated hardware configurations to perform specific tasks, such as filtering or amplification. Modifying these systems can be cumbersome, as it usually involves redesigning or replacing physical components. On the other hand, DSP uses software-based algorithms to perform similar operations. This means that a single DSP system can be easily reprogrammed to adapt to new tasks or evolving requirements without needing significant hardware changes. This flexibility makes DSP highly versatile and cost-effective for modern applications where adaptability is crucial.

Efficiency is another area where DSP outperforms traditional analog methods. While analog systems can handle simple tasks effectively, they become increasingly complex and less efficient as the required operations grow in sophistication. DSP systems, however, are designed to handle complex signal processing tasks with greater speed and lower power consumption. Advances in digital technology, such as optimized processors and compact hardware designs, further enhance the efficiency of DSP systems, allowing them to perform intricate operations in smaller physical spaces. This efficiency is particularly important in portable and embedded systems, where size and power constraints are critical.

In summary, DSP surpasses traditional analog methods by providing higher precision, greater flexibility, and improved efficiency. Its ability to process signals digitally ensures better performance in environments where signal quality, adaptability, and resource optimization are essential. These advantages have cemented DSP's role as a foundational technology in fields such as communication, audio processing, and biomedical applications.

The importance of DSPs lies in their ability to process and analyze real-world signals, such as audio, video, and sensor data, in their digital form. By performing operations in the digital domain, DSPs provide greater precision, reliability, and flexibility, which are essential in modern technological applications. For example, in telecommunications, DSPs play a vital role in enabling data transmission with a high speed by performing functions like error correction, modulation, and signal compression. Similarly, in audio signal processing, DSPs are used to enhance sound quality through noise reduction, equalization, and echo cancellation techniques [6].

Beyond these, DSPs are integral to a wide range of applications, including graph signal processing for analyzing interconnected data, image and video processing for multimedia applications, and radar systems for defense and navigation. Their

compact and energy-efficient designs also make them a natural fit for portable devices, such as mobile phones, hearing aids, and portable media players. As the world moves toward increasingly intelligent systems, DSPs are also finding applications in emerging fields like machine learning, biomedical devices, and Internet-of-Things (IoT) devices [7].

The versatility and efficiency of DSPs have solidified their role as an indispensable technology in both consumer and industrial domains. Their ability to adapt to new challenges and meet the stringent performance requirements of modern applications continues to drive innovation in this field.

2.3 DSP Accelerator: Quadrature Amplitude Modulation (QAM) Technology

Quadrature Amplitude Modulation (QAM) is a widely used technique in DSP Accelerator within modern digital communication systems, designed to improve the efficiency of information transmission. Traditional methods of data transmission often involve modulating a single carrier wave to encode information, effectively utilizing only one dimension of the signal space. In contrast, QAM achieves a higher level of efficiency by encoding data using two orthogonal components of a carrier wave: the in-phase signal $I(t)$ and the quadrature signal $Q(t)$. These components are combined to create a composite signal that carries information in two dimensions, effectively doubling the data-carrying capacity of the transmission channel.

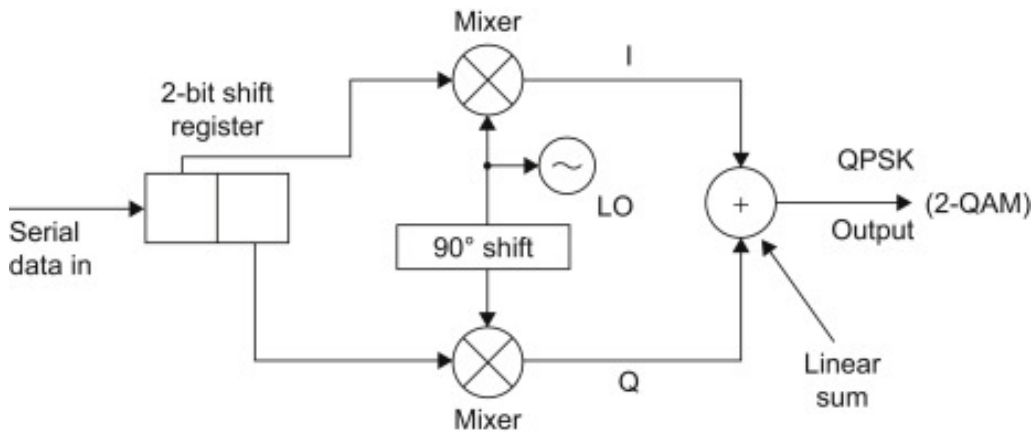


Figure 7: 4-QAM modulation generation

2.3.1 Realization of QAM

Figure 7 illustrates the generation of a 4-QAM modulated signal, demonstrating how input data is transformed into a composite signal which is suitable for transmission [26].

The modulation process begins with a serial input bitstream, which is grouped into symbols, each representing a fixed number of bits. In the depicted example,

a 2-bit shift register is used to group the input data into symbols of two bits each. These symbols are then mapped onto two separate streams: one for the in-phase $I(t)$ component and the other for the quadrature $Q(t)$ component. This double data stream approach allows QAM to encode information in two dimensions, enhancing its spectral efficiency compared to single-dimension modulation.

A local oscillator (LO) generates the carrier signal, which serves as the base for modulation. The LO output is split into two paths. One remains unchanged for the I component, while the other is passed through a 90-degree phase shifter to generate the Q component. This ensures that the I and Q signals are orthogonal, which is a critical feature for maintaining their independence and enabling accurate demodulation.

The I and Q data streams are then modulated onto their respective carriers using mixers, which effectively multiply the carrier signals with the corresponding data. The mixer outputs represent the amplitude-modulated in-phase and quadrature components of the signal. These two modulated signals are then combined using a linear summer, which is also named as an adder, to produce the final QAM-modulated signal. The resulting signal is a combination of the I and Q components, where the amplitude and phase vary based on the input data.

The modulated signal can be expressed as [27]:

$$s(t) = I(t) \cdot \cos(2\pi f_c t) - Q(t) \cdot \sin(2\pi f_c t) \quad (1)$$

where:

- $s(t)$ is the QAM modulated signal at time t ,
- $I(t)$ is the in-phase component,
- $Q(t)$ is the quadrature component,
- f_c is the carrier frequency,
- $\cos(2\pi f_c t)$ and $\sin(2\pi f_c t)$ are the orthogonal carrier signals.

In this formulation, $I(t)$ and $Q(t)$ represent the modulating signals that transfer the data with the carrier frequencies. The orthogonality of the cosine and sine functions ensures that the in-phase and quadrature components stay independent, allowing for the simultaneous transmission of two signals over the same frequency band, and enabling QAM to achieve higher efficiency compared to modulations that apply only a single carrier.

The negative sign before the quadrature component $Q(t) \cdot \sin(2\pi f_c t)$ is a matter of convention [28] and depends on the implementation of the modulation. Although there exist some occasions where a positive sign is applied in the formula, the essential concept of the two different expressions remains the same, which is, two orthogonal carriers are modulated in amplitude by the respective in-phase and quadrature components.

The amplitude and phase of the resulting QAM signal $s(t)$ are determined by the values of $I(t)$ and $Q(t)$. Thus, the amplitude A and phase θ can be calculated as [27]:

$$A = \sqrt{I(t)^2 + Q(t)^2} \quad (2)$$

$$\theta = \tan^{-1} \left(\frac{Q(t)}{I(t)} \right) \quad (3)$$

These equations explain how QAM combines both amplitude and phase modulation to represent data symbols, with each unique pair of $I(t)$ and $Q(t)$ corresponding to a specific point in the signal constellation diagram.

2.3.2 Graphical Representation of QAM

As a result, the output of QAM modulation process corresponds to a constellation point on a two-dimensional plane, with the in-phase and quadrature components representing the x and y axes, respectively.

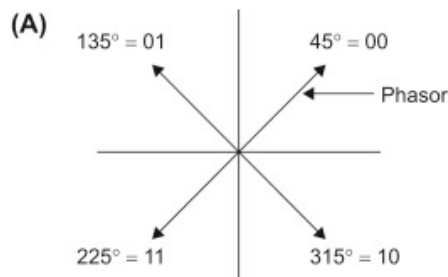


Figure 8: 4-QAM phasor diagram

For example, figure 8 depicts 4-QAM phasor diagram, where the amplitude and phase of every possible combination of two bits are shown [26]. Each arrow in this diagram, referred to as a phasor, represents the signal's amplitude through its length and its phase through the angle of the arrow. For example, when the receiver detects a signal with a 135-degree phase angle, it corresponds to the two-bit combination 01 being transmitted. The representation of binary combinations by phase angles allows efficient transmission and interpretation of data.

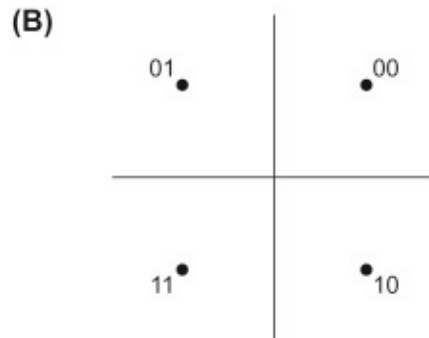


Figure 9: 4-QAM constellation diagram

Figure 9 illustrates a 4-QAM constellation diagram [26], which is a simplified version of Figure 8. Unlike figure 8, it does not show the arrows or phasors. A constellation diagram is a visual representation of the signal points, where each point corresponds to a specific amplitude and phase. These diagrams explain the concept of phasors and constellation diagrams in digital communication systems and are widely used in modulation schemes to analyze and understand signal transmission, ensuring accurate data recovery at the receiver end.

To be more efficient with signal transmission, higher-order QAM schemes increase number of bits transmitted per symbol by expanding the number of constellation points in the signal space. This allows for greater data rates within a fixed bandwidth, which is a key advantage of QAM in modern communication systems. Figure 10 depicts the constellation diagram for 16-QAM [26].

In the case of 16-QAM, the constellation diagram is consisted of 16 distinct points, each representing a special combination of amplitude and phase. These points are arranged in a grid structure, ensuring equal spacing to protect against noise. Each constellation point corresponds to a unique 4-bit binary sequence, as 16 points can encode $2^4 = 16$ combinations. For example, a specific point on the diagram may correspond to the binary sequence 1010, which is determined by the in-phase $I(t)$ and quadrature $Q(t)$ components of the signal.

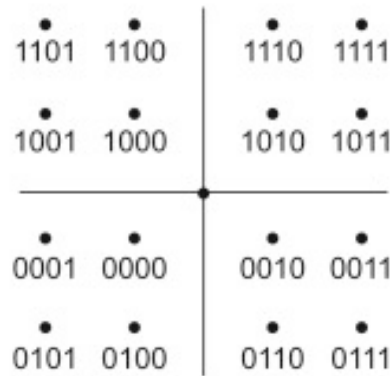


Figure 10: 16-QAM constellation diagram

Higher-order QAM, such as 64-QAM and 256-QAM, extend this principle by further increasing the number of constellation points. In 64-QAM, there are 64 distinct points, enabling each symbol to encode 6 bits ($2^6 = 64$), while 256-QAM uses 256 points to encode 8 bits ($2^8 = 256$) per symbol. These higher-order QAMs allow significantly higher data rates without requiring additional bandwidth. For example, increasing from 16-QAM to 64-QAM can increase data throughput by 50% within the same spectral allocation.

Figure 11 demonstrates the scalability of QAM by depicting how constellation points increase as the modulation order grows [29]. By leveraging the principles of amplitude and phase modulation, higher-order QAM schemes achieve a delicate balance between data rate efficiency and transmission robustness, solidifying QAM's role as a cornerstone of modern digital communication systems.

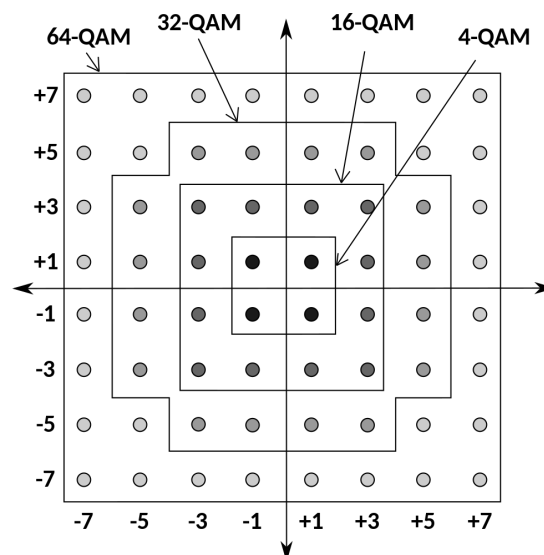


Figure 11: Constellation points for QAM with points numbers being powers of 2

The ability of QAM to achieve high data rates within a constrained bandwidth makes it a preferred modulation scheme for modern high-speed communication systems, such as 4G LTE, 5G NR, Wi-Fi (IEEE 802.11 standards), and cable internet. By efficiently utilizing the available frequency spectrum, QAM supports the growing demand for high data rates in applications such as video transfer and real-time communication.

However, higher-order QAM schemes come with trade-offs. As the number of constellation points increases, the points become closer together in the signal space, making the system more susceptible to noise, interference, and hardware imperfections. This susceptibility necessitates a higher signal-to-noise ratio (SNR) for accurate symbol detection, particularly in 64-QAM and 256-QAM. Advanced error correction techniques, such as forward error correction (FEC), and complicated receiver designs are therefore essential to face with these challenges and maintain reliable communication.

2.3.3 Application of QAM

The realization of high-efficient data transmission has made QAM a fundamental technique in various applications. An important use of QAM is in field of telecommunication networks such as 4G LTE and 5G. These systems rely on higher order QAM level, for example, 256-QAM, to deliver faster data rates and maintain spectrum efficiency at the same time. In 4G LTE, QAM enhances the network's ability to support data-heavy tasks like video streaming [30]. Similarly, in 5G, advanced QAM techniques allow connection speeds at gigabit range, enabling modern applications such as augmented reality (AR) and autonomous vehicles. The efficient use of spectrum through QAM ensures that these networks can handle the increasing demand for faster and more reliable wireless connectivity.

In addition to telecommunications, QAM also plays a vital role in broadband and cable systems. It is a key technology in the Data Over Cable Service Interface Specification (DOCSIS) standard, which enables high speed internet, digital video and audio transmission, as well as mobile service networks [30]. Modulation levels like 64-QAM and 256-QAM allow networks to deliver multiple services simultaneously and optimizing bandwidth usage at the same time. This makes QAM inevitable for broadband systems, especially for the system that has highe demand of a balanced efficiency with signal transfer quality.

QAM is also a basic technology for optical communication, where even higher-order levels, such as 1024-QAM, are used to meet the increasing demand of data transmission over fiber-optic networks[31]. These networks are the foundation of cloud computing and international telecommunications. By increasing the number of bits per symbol, QAM allows optical systems to improve data capacity without the need for additional physical infrastructure. This scalability is crucial for dealing with the rapid growth of data traffic and ensuring the efficiency of global telecommunication systems.

Furthermore, QAM plays a vital role in wireless fidelity (Wi-Fi). The implementation of 1024-QAM in Wi-Fi 6 enables a 25% increase in data throughput

compared to previous standards, such as Wi-Fi 5 (802.11ac), which used 256-QAM [32]. This improvement is achieved by encoding 10 bits per symbol, as opposed to 8 bits in 256-QAM. The higher order of modulation allows Wi-Fi 6 to transmit more information within the same bandwidth, enhancing spectral efficiency. This capability is particularly critical in high density occasions, such as smart homes and IoT networks, where multiple devices are connected at the same time.

In summary, QAM is not only a fundamental technology of current communication technologies, but also a critical technology of innovation, which supporting various applications and paving the way for future development in global telecommunication. Its key advantages lie in the ability to encode multiple bits into a single symbol, allowing for faster data transmission and efficient use of available bandwidth. This makes QAM an essential technology of various technologies, including telecommunication networks, broadband internet, optical communication, and Wi-Fi standards. While higher-order QAM level bring challenges, these issues are handled through advanced error correction techniques and improved receiver designs. The ability of QAM to balance data efficiency and transmission quality ensures it to remain a crucial component in meeting the increasing data transmission needs around the world.

2.3.4 DSP Accelerator Application: 5G Transmission System

5G stands for the Fifth Generation of wireless communication technology. It is designed to support a diverse range of applications, from wide mobile broadband to long distance communications with massive information [33]. These features position 5G as a cornerstone of modern technologies such as autonomous systems, smart cities, and entertainments.

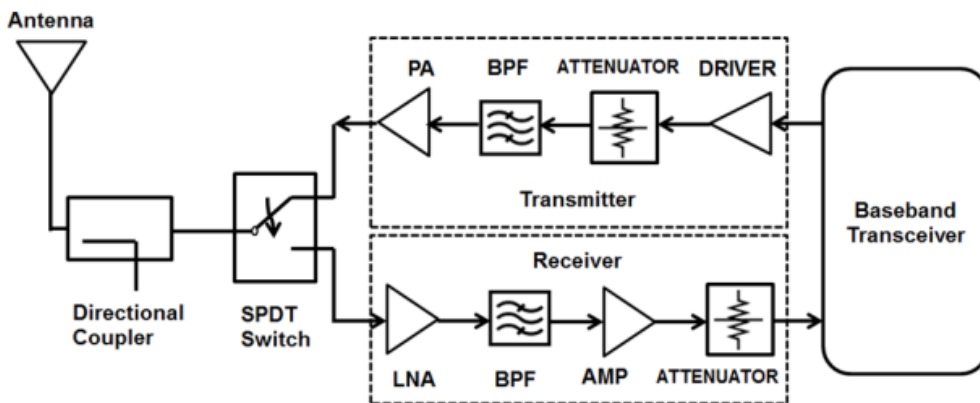


Figure 12: 5G transceiver block diagram

Figure 12 illustrates a 5G transceiver block diagram, which demonstrates the signal flow in both the transmitter and receiver paths, as well as the components that enable the system to operate efficiently [34]. At the heart of the system is the

antenna, which is responsible for radiating and receiving radio frequency (RF) signals. A directional coupler and an SPDT (Single Pole Double Throw) switch ensure that the same antenna can be used for both transmitting and receiving signals. The SPDT switch efficiently routes the signals to either the transmitter or receiver path depending on the operational mode of the transceiver.

In the transmitter path, the baseband transceiver generates the signal to be transmitted. This signal passes through a series of components that prepare it for wireless transmission. First, the signal is amplified by a Power Amplifier (PA) to ensure it has sufficient strength for transmission. Next, a Band Pass Filter (BPF) is used to filter the signal, allowing only the desired frequency range to pass through while rejecting any unwanted noise or out-of-band components. Following this, an attenuator adjusts the signal level when necessary to prevent it from becoming distorted or overloading subsequent components. Finally, a driver amplifies the filtered and adjusted signal further and prepares it for transmission through the antenna.

On the receiver side, the incoming signal, which is often weak, is first processed by a Low Noise Amplifier (LNA). The LNA plays a critical role by amplifying the signal while adding minimal noise, which is essential for maintaining signal integrity. The amplified signal is then passed through a Band Pass Filter (BPF), which removes any unwanted frequencies and isolates the desired signal. To further strengthen the signal for processing, it is passed through an amplifier. An attenuator then ensures that the signal level is properly adjusted before it is sent to the baseband transceiver for further processing, such as demodulation and decoding.

The baseband transceiver acts as the interface between the RF domain and the digital domain. It is responsible for generating the signals for transmission and processing the received signals for data recovery. This component plays a central role in ensuring that the system operates efficiently and reliably.

The overall structure of the 5G transceiver highlights its ability to seamlessly handle both transmission and reception. The modular nature of the design ensures that the system can be optimized for specific applications, balancing power, performance, and complexity. By combining key elements such as amplifiers, filters, and signal routing mechanisms, the transceiver effectively processes high-frequency signals while maintaining signal quality and integrity.

A key innovation in 5G is its ability to operate across different frequency bands, including sub-6 GHz and mmWave bands. While mmWave frequencies enable ultra-high data rates, they also face significant challenges due to propagation limitations and increased susceptibility to attenuation [35]. Furthermore, the implementation of 5G networks requires careful management of power consumption, particularly in densely populated urban environments, where energy efficiency becomes critical [36].

2.4 Signal Transmission Performance Metric

The 5G wireless network is engineered to meet the escalating demands of contemporary communication systems by delivering enhanced performance across multiple metrics. These metrics are crucial for assessing the efficiency, reliability, and capability of

5G networks in providing high-speed connectivity and low-latency communication, essential for applications such as autonomous vehicles, remote surgeries, and smart cities.

Key performance metrics in 5G transmission include data rate, latency, reliability, spectral efficiency, and energy efficiency. High data rates ensure rapid information transfer, while ultra-low latency minimizes delays, enabling real-time interactions. High reliability guarantees uninterrupted communication, spectral efficiency maximizes the utilization of available bandwidth, and energy efficiency ensures sustainable operations for both devices and infrastructure [37].

This subchapter explores more into these performance metrics, providing a comprehensive understanding of their significance in 5G networks. By analyzing these metrics, we can better appreciate how 5G addresses the challenges of modern communication and achieves its ambitious goals for enhanced connectivity and efficiency.

2.4.1 Error Vector Magnitude (EVM)

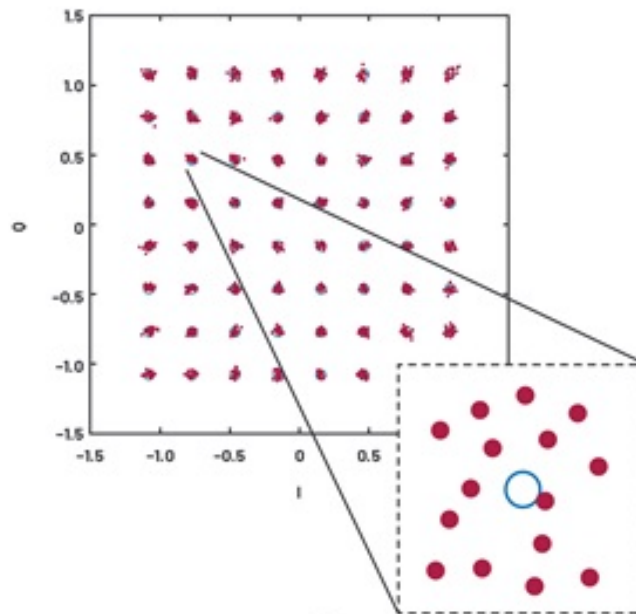


Figure 13: Signal deviation

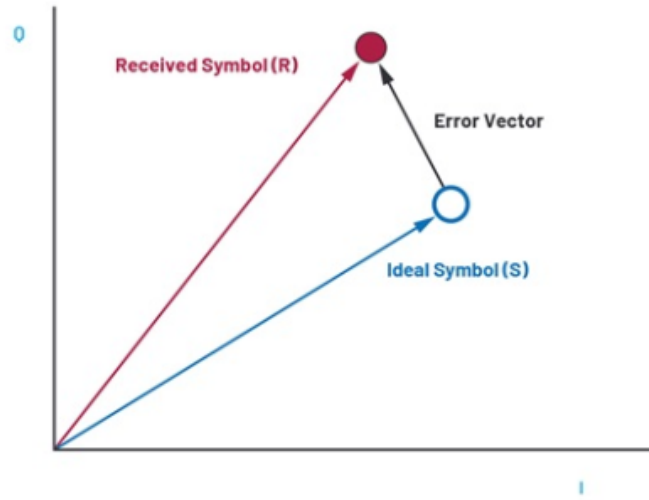


Figure 14: EVM vector

Error Vector Magnitude (EVM) is a critical metric for evaluating signal quality in communication systems by quantifying signal loss and deviations. As illustrated in Figure 13, each red point represents the actual output signal, while the blue points in the enlarged section indicate the expected positions of the signal symbols [38]. This grid diagram effectively visualizes the signal deviation and provides a clear representation of EVM, highlighting the extent of error between the transmitted and received signals.

It measures how much the received signal deviates from the ideal signal. This difference, known as the Error vector, represents the distortion introduced during transmission as illustrates in Figure 14 [38]. A higher EVM value indicates greater distortion, which can result in data errors or reduced signal quality. The EVM equation is given as [39]:

$$\text{EVM (\%)} = \frac{\sqrt{\frac{\sum_{i=1}^N |S_{\text{actual},i} - S_{\text{ideal},i}|^2}{N}}}{\sqrt{\frac{\sum_{i=1}^N |S_{\text{ideal},i}|^2}{N}}} \times 100$$

where $S_{\text{actual},i}$ is the actual signal, $S_{\text{ideal},i}$ is the ideal signal, and N is the total number of symbols.

By analyzing EVM, we can assess the performance of the system and identify areas where signal quality is being degraded. This makes it an essential tool for testing and improving communication systems like 5G transceivers, where maintaining accurate signal transmission is critical.

2.4.2 Power Spectral Density (PSD)

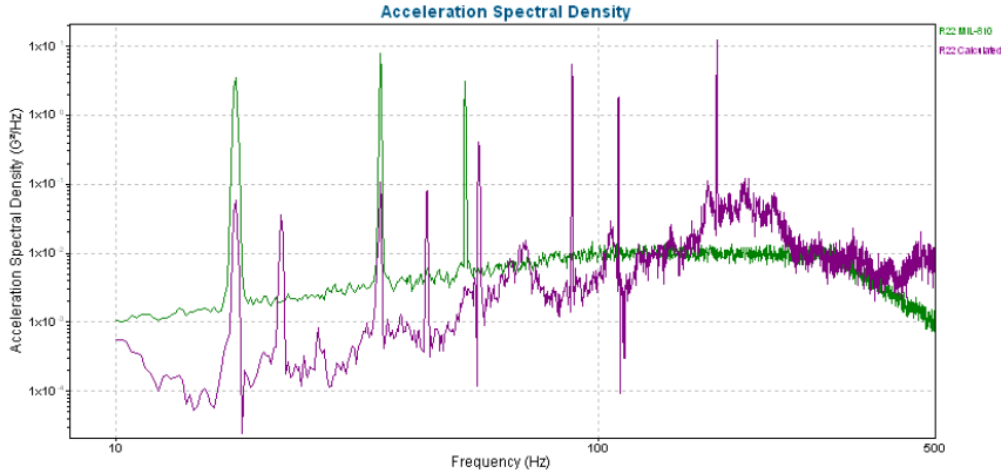


Figure 15: Comparison of calculated PSD and test standard PSD

Power Spectral Density (PSD) is a key metric in evaluating the performance of 5G transmission systems, representing the power distribution of a signal as a function of frequency. It provides insights into spectral efficiency, interference management, and regulatory compliance, making it essential for assessing 5G signal quality and system design [40]. Figure 15 shows an example comparison of calculated PSD and test standard PSD, where the x-axis is frequency and y-axis is spectral density [41]. Mathematically, PSD is defined as the Fourier transform of a signal function, capturing its frequency-domain characteristics.

In 5G systems, PSD analysis is critical due to the wide range of operating frequency bands, from sub-6 GHz to millimeter-wave (mmWave) bands. By examining PSD, engineers can evaluate in-band power utilization, out-of-band emissions, and overall spectral efficiency. For instance, high PSD values in specific bands indicate efficient spectrum use, while excessive out-of-band emissions can cause interference with adjacent systems [42]. Figure 1 illustrates a typical PSD plot for a 5G signal, highlighting the distribution of in-band and out-of-band power.

PSD is also instrumental in comparing the performance of different modulation schemes (e.g., QPSK, 16-QAM, 64-QAM), as higher-order modulations generally result in more compact PSD profiles, albeit with stricter signal-to-noise ratio (SNR) requirements [43]. Figure 11 compares PSD plots for these modulation schemes, emphasizing their trade-offs in bandwidth efficiency and noise resilience.

By analyzing the PSD of simulated and hardware-generated signals, this study assesses the compliance of the designs with 5G standards and analysis of their performance for real-world applications.

2.4.3 Adjacent Channel Leakage Ratio (ACLR)

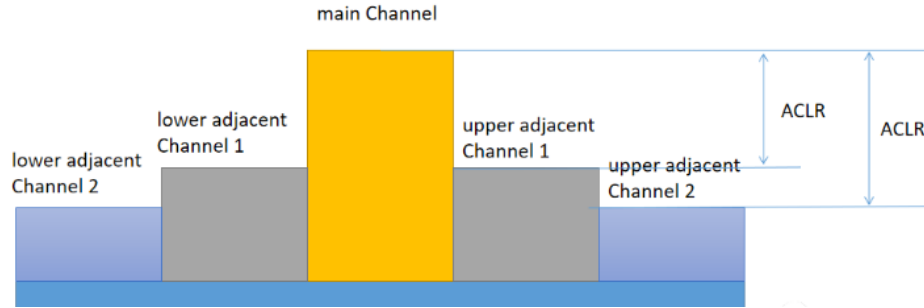


Figure 16: ACLR bandwidth and main channel bandwidth

Adjacent Channel Leakage Ratio (ACLR) is a critical performance metric for 5G transmission systems, measuring the ratio of the power leaked into adjacent frequency channels to the total power within the intended transmission channel. It is a key indicator of spectral efficiency and interference mitigation, as excessive leakage can degrade the performance of neighboring channels, violating regulatory standards and impairing overall network operation [44]. Figure 16 shows an example of ACLR representation, and the equation of calculating ACPR [45].

In 5G systems, maintaining a high ACLR is particularly challenging due to the use of wideband signals and high-frequency carrier components, such as those in the millimeter-wave (mmWave) range. Factors such as non-linearities in the power amplifier and imperfect modulation contribute to adjacent channel leakage, necessitating advanced techniques such as digital predistortion (DPD) and enhanced filtering [46]. Figure 1 illustrates the concept of ACLR, showing the power distribution across the main and adjacent channels.

ACLR is closely linked to regulatory compliance, as standards such as those defined by 3GPP impose strict limits on out-of-band emissions. Evaluating ACLR ensures that 5G systems not only meet these standards but also reduce interference in densely populated frequency bands. Moreover, it provides a measure of the efficiency and linearity of the transmitter chain, particularly for power amplifier designs [47].

In the context of this thesis, ACLR is used to evaluate the performance of RISC-V-based hardware implementations of 5G signal processing modules. By analyzing ACLR through simulation and hardware testing, this study ensures that the designs meet spectral efficiency requirements while minimizing interference.

2.5 Transmitter (TX) Chain

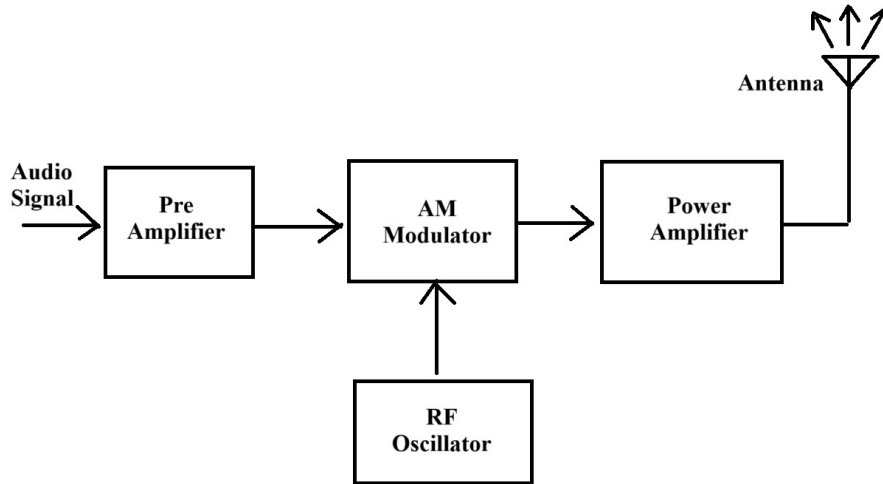


Figure 17: Typical transmitter block diagram

The transmitter (TX) chain is a critical part of any communication system, responsible for taking data from the source and preparing it for transmission over a communication medium, such as wireless, fiber, or cable. The TX chain consists of multiple stages, each performing specific functions to modulate and amplify the signal for efficient transmission to the receiver [48].

The first stage of the TX chain involves data encoding and modulation. The raw data is converted into a format that can be transmitted, such as bits or symbols. Then, the data undergoes modulation, where the information is embedded into a carrier signal [48]. Different modulation techniques like QAM (Quadrature Amplitude Modulation) and PSK (PSK) are used depending on the application and desired data rates. The modulated signal represents the data in terms of amplitude, frequency, or phase variations of the carrier wave [48].

Next, the modulated signal is transferred to a digital-to-analog converter (DAC), which enables the conversion of digital signal into an analog waveform suitable for transmission. The analog signal then enters the up-conversion stage, where it is shifted to a higher frequency, known as the carrier frequency, that matches the requirements of the transmission medium. This is typically done by mixing the baseband signal with a high-frequency oscillator in the mixer [49].

Following up-conversion, the signal is passed through a power amplifier (PA) to increase its power level. This amplification ensures that the signal can travel long distances without significant degradation [48]. The amplifier must be carefully designed to maintain the integrity of the signal, as excessive amplification can introduce distortion. After amplification, the signal is filtered to remove unwanted frequency components, ensuring a clean transmission at the desired frequency band.

Finally, the prepared signal is sent to the antenna or transmission medium. The TX chain must arrange the power, frequency, and bandwidth efficiently to meet

communication standards while minimizing interference with adjacent channels [49].

2.6 Receiver (RX) Chain

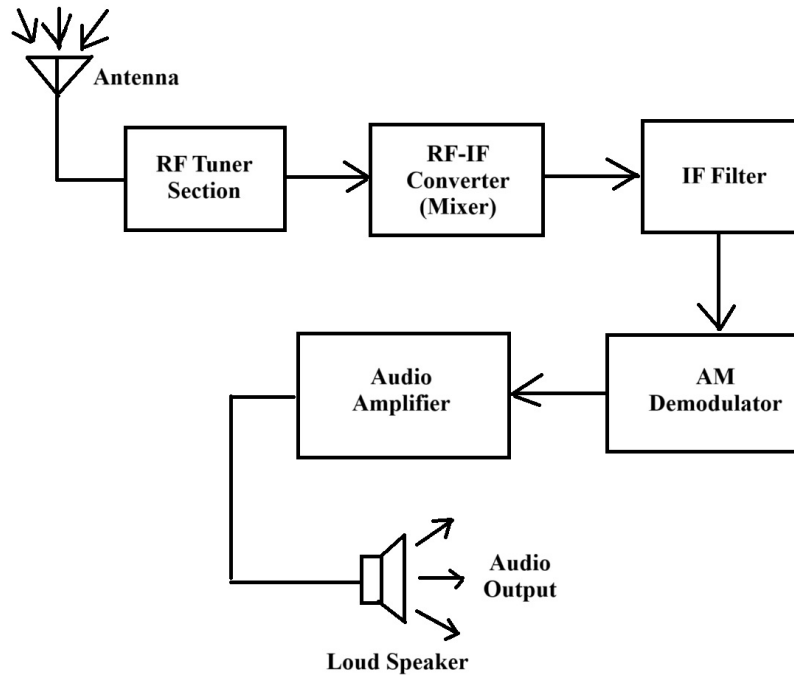


Figure 18: Typical receiver block diagram

The receiver (RX) chain is responsible for capturing the transmitted signal, processing it, and converting it back into data that can be understood by the system. The RX chain must handle various challenges, such as signal attenuation, noise, and interference, to accurately recover the original information [50].

As illustrated in the figure 18, the first stage of the RX chain is the antenna or input interface, which captures the transmitted signal [50]. This signal is typically weak due to the distance traveled and environmental factors, so it is first passed through a low-noise amplifier (LNA). The LNA enhances the signal while decreases the additional noise, ensuring that the received signal maintains a good signal-to-noise ratio (SNR) [50].

After amplification, the signal enters the down-conversion stage. The RX chain uses a mixer to shift the high-frequency received signal down to a lower frequency, known as the intermediate frequency (IF) or baseband, for easier processing [49]. This is the reverse process of the up-conversion stage in the TX chain. Filters are then applied to remove unwanted signals and noise from adjacent channels.

Next, the signal is transferred to an analog-to-digital converter (ADC), where the analog waveform is converted into the form of digital signal [50]. The digital signal then undergoes demodulation and decoding, which reverse the modulation

and encoding processes used in the TX chain. This stage extracts the original data from the modulated carrier, recovering the transmitted information.

Finally, the decoded data is processed and sent to the appropriate system or user application. Throughout the RX chain, techniques such as equalization, error correction, and signal filtering are often employed to compensate for any signal degradation that occurred during transmission [49]. The RX chain must be designed to be highly sensitive and selective to ensure reliable communication, especially in challenging environments with interference or noise.

Together, the TX and RX chains form the core of any communication system, working in tandem to ensure efficient data transmission and reception.

2.7 Verification Method

This section introduces some verification methods widely applied in various areas, including three verification methods as follows.

FPGA Verification involves synthesizing the RTL design onto an FPGA to verify the system in a real hardware environment. This allows testing under real-world conditions and checking how the system interacts with peripherals. It is useful for both functional validation and performance analysis [51].

ASIC verification is a critical technique for ensuring that custom-designed chips meet their functional and performance criteria. Unlike generic processors, ASICs combine all components, including processing units, memory, and interfaces, onto a single chip designed for a specific purpose. This integration enables efficient verification of the complete system flow using a uniform framework [52].

Functional verification has significant importance in areas such as telecommunications, automobiles, and consumer electronics, where dependability and performance are paramount. In complicated systems, such as DSP accelerators or processors, functional verification guarantees that each module functions correctly and that module integration does not create unexpected behavior [53].

2.7.1 FPGA Verification

FPGA hardware devices are reconfigurable, which allow users to develop special digital circuits by programming their internal structure [54]. Introduced in the 1980s as an alternative to fixed-function Application-Specific Integrated Circuits (ASICs), FPGAs have developed into various platforms for a wide range of applications, from prototyping and verification to deployment in high performance systems [51].

The cores of FPGAs are integrated circuits composed of a grid of programmable logic elements, connected by configurable interconnects [51]. These logic elements, often called Configurable Logic Blocks (CLBs), which contains blocks such as look-up tables (LUTs), flip-flops, and multiplexers, providing possibilities to the implementation of arbitrary logic functions [54]. The interconnects can be dynamically configured to establish connections between logic elements, creating custom digital circuits tailored to specific applications.

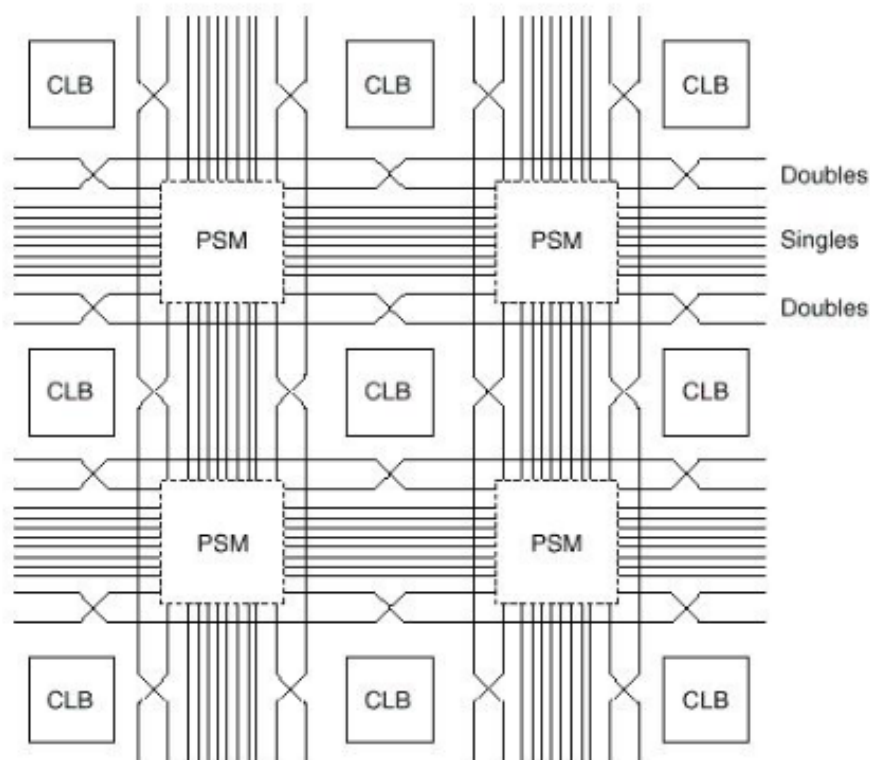


Figure 19: Typical FPGA architecture³

Figure 19 illustrates the general architecture of an FPGA, highlighting the CLBs, interconnects, and specialized resources.

The development of FPGAs began in the 1980s as a response to the increasing demand for flexible and reusable hardware. Xilinx introduced the first commercially available FPGA in 1985, marking a significant milestone in digital design. Early FPGAs were simple, containing a limited number of logic blocks and interconnects, but they provided an alternative to fixed-function chips for prototyping and small-volume applications [51].

Over the decades, advancements in semiconductor technology have transformed FPGAs into highly sophisticated devices. Modern FPGAs feature millions of logic elements, high-speed interconnects, and specialized resources such as DSP blocks, embedded processors, and transceivers. These developments have expanded their applications into fields such as telecommunications, artificial intelligence, and aerospace [55].

FPGAs are designed to be programmable by end-users, typically through a multi-step process as follows [51].

Hardware description: Users describe the desired hardware behavior using Hardware Description Languages (HDLs) like VHDL or Verilog. Synthesis: The translation is done on HDL code and send it into a netlist. The netlist contains the information of the logic gates as well as the connections required to be implemented in the design.

³<https://courses.csail.mit.edu/6.111/f2005/handouts/L10.pdf>

Map and route: According to the netlist, designs are mapped with the FPGA's resources. This determines how the logic blocks and the connections are implemented. Bitstream generation: A bitstream file is generated, which represents the programmed design. It is uploaded to the FPGA to implement the circuit. The programmability makes FPGAs highly versatile for iterative design, testing, and optimization.

FPGAs are widely used across various domains due to their flexibility and high performance. For example [55], FPGAs are used in 5G signal transmission and network equipment for real-time signal processing. FPGAs are also integrated into devices for image processing and video encoding. Furthermore, FPGAs are applied in scientific research due to the support of high-speed data acquisition and analysis in fields like physics and genomics.

FPGAs also play an important role in hardware verification because they provide a physical platform for testing and validating designs [54]. Verification workflows frequently begin with software-based simulations or functional verification of the hardware design, which is then implemented on an FPGA for real-world testing. This method allows engineers to test designs under realistic timing and signal conditions, detect and correct faults that may not be seen in software simulations, and evaluate the integration of hardware components and peripherals. For example [22], in RISC-V verification, FPGAs are used to run compiled programs, such as ELF files, to test the processor's basic capabilities and custom extensions.

One of the most significant advantages of FPGA verification is its speed. FPGAs run designs far quicker than software simulators, allowing for real-time testing and early detection of faults. FPGAs' flexibility enables for easy design modification and reprogramming, simplifying iterative testing during development [54]. Furthermore, FPGA verification is cost-effective because it eliminates the significant costs associated with ASIC manufacture during the prototyping stage. Furthermore, FPGAs provide scalability, allowing for the testing of big and complicated designs, such as multi-core computers [22].

However, FPGA verification has limits as compared to ASIC verification. FPGAs have limited logic, memory, and routing resources, limiting the size and complexity of the designs that may be implemented. Timing behavior on FPGAs frequently differs from that of ASICs, necessitating careful adaptation to guarantee that the design adheres to the FPGA fabric restrictions. Furthermore, FPGAs are less power-efficient and slower than ASICs for identical designs due to programmability overhead [56]. Furthermore, certifying a design on an FPGA does not ensure same behavior when migrated to an ASIC, requiring additional verification processes to ensure compatibility [57].

Despite these challenges, FPGA verification is still an important step in the development process, bridging the gap between simulation and final implementation by providing a versatile and efficient platform for early-stage testing.

2.7.2 ASIC Verification

ASICs are customized integrated circuits tailored for specific applications, offering optimized performance, power efficiency, and area utilization. Unlike FPGAs, ASICs are fixed-function devices that cannot be reprogrammed after fabrication. Due to their bespoke nature and high production cost, rigorous verification is essential to ensure their functionality and reliability before manufacturing [58].

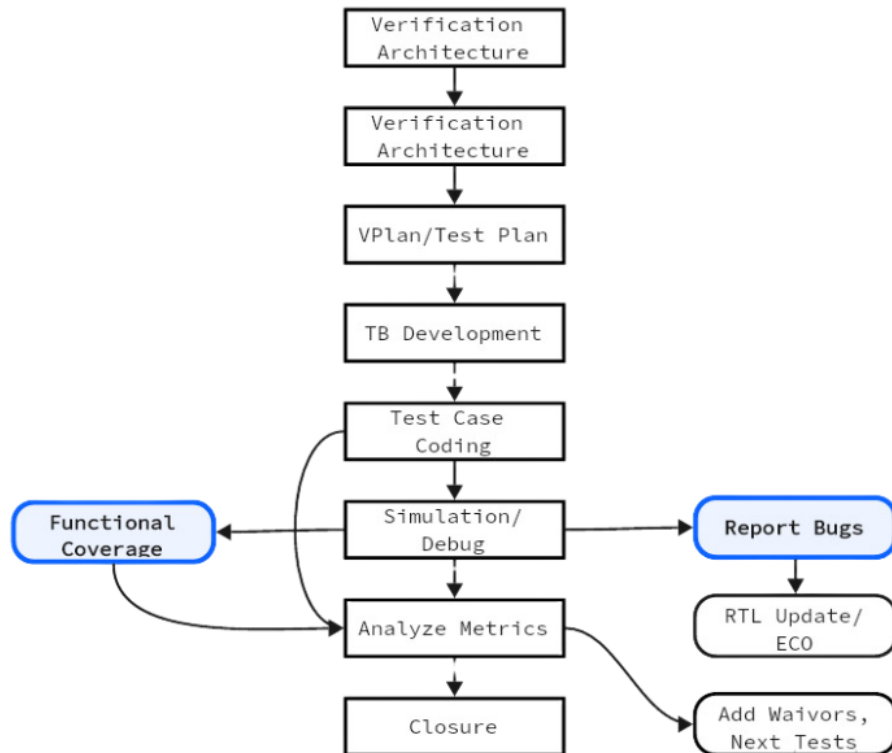


Figure 20: Typical ASIC verification flow

ASIC verification methods began to evolve in the late 1970s and early 1980s, corresponding with the rise of chip design in industries such as communications and computing. As ASIC complexity grows, manual verification methods become insufficient, and automated tools and structural verification methods emerge. Over time, developments in sensing technologies will be important in maintaining the availability and reliability of more sophisticated ASIC designs [59].

In the 1990s, hardware description languages (HDLs) like Verilog and VHDL transformed ASIC design by allowing simulation-based verification [60]. These languages standardize the description of hardware architectures, making it easier to develop rigorous testing tools and methodologies. The scope of ASIC testing has expanded with the development of methodologies such as RTL (Register Transfer Level) simulation and formal verification. Nowadays, advanced techniques like the Universal Verification Method (UVM), random probabilistic testing, and hardware replication are ubiquitous in the business [60].

As illustrated in Figure 20 [61], ASIC verification process ensures that a design meets functional and performance requirements before it is put into production [59]. This method usually consists of multiple steps [60]. Functional testing use approaches such as directed testing and limited random testing to guarantee that the design works as intended in all stated circumstances. Formal verification examines the state space and provides mathematical proof that the design fits the standards. Timing verification verifies that your design satisfies worst-case timing requirements, assuring signal integrity and proper functioning at the desired clock frequency. Power analysis measures power usage and leakage in order to improve energy efficiency. Together, these phases offer a thorough framework for identifying and resolving difficulties early in the design cycle [60].

Modern verification flows heavily rely on simulation, emulation, and synthesis tools to streamline the debugging and validation process. Verification testbenches, which are integral to these flows, are implemented using HDLs, SystemVerilog, or C-based verification frameworks [62]. These tools enable engineers to simulate complex scenarios, detect potential faults, and ensure the robustness of the design before committing to fabrication.

ASIC testing is especially crucial in businesses that demand high performance and dependability, such as telecommunications, automotive, smartphones, and gaming consoles [52]. The precision of ASIC verification ensures that your design is perfectly optimized for its intended use. Furthermore, verified ASICs are better than the traditional or reconfigurable hardware in aspects of performance, power economy, and footprint optimization. Once confirmed and built, ASICs are very reliable and cost-effective for mass production, making them an excellent choice for applications requiring long-term dependability and performance [52].

While ASIC testing has numerous benefits, it also poses a number of obstacles. The procedure is resource-intensive, necessitating significant processing power and engineering skill, resulting in higher costs and time [63]. As projects grow in complexity, so do the issues of proving component interoperability. Furthermore, design flaws cannot be addressed after construction, thus thorough inspections are essential to avoid costly repeated work. ASIC verification also relies significantly on complicated tools, which are costly and require specialized skills to run efficiently [63].

A comparison of ASIC and FPGA verification shows clear tradeoffs [64]. FPGA verification enables more flexible and iterative prototyping and reconfiguration, while ASIC verification ensures that the final design meets stringent performance and reliability requirements. ASICs offer high accuracy, but the downside is that they are less flexible than FPGAs. These differences highlight the importance of choosing the appropriate checking method based on specific project requirements.

2.7.3 Functional Verification

Functional verification using simulation is a critical step in assuring that a hardware or software system executes its intended functions correctly. This procedure is more cost-effective than physical prototype or hardware-based verification, making it an important approach in early-stage design and development [53].

Simulation-based functional verification tests the system by simulating its source code, which is usually written in hardware description languages (HDLs) like Verilog, VHDL, or SystemVerilog [65]. Engineers can use these simulations to construct virtual test environments in which to assess designs under a variety of conditions, such as edge cases and stress scenarios [65]. Engineers can identify and rectify functional faults early in the development cycle by creating test inputs and analyzing the results, considerably lowering the chance of problems happening in the future.

One of the primary benefits of simulation is its low cost. Simulations, as opposed to hardware-based verification, which involves real devices or prototypes, can be carried out entirely in software, minimizing the need for costly fabrication and testing equipment. This low cost makes simulation accessible to projects with minimal resources while still offering detailed insights into the design's functionality[66].

Furthermore, simulation-based verification is quite adaptable. It allows for the testing of both individual components (such as modules or cores) and integrated systems [66]. Simulation ensures the entire system's robustness and dependability when implemented in hardware by isolating and fixing functional faults early on.

In short, functional verification by simulation is a critical step in the design and development process. It ensures that systems function properly, detects faults early on, and provides a low-cost, adaptable alternative for testing designs before transitioning to more expensive hardware-based verification approaches.

3 Introduction of the FPGA board ZCU104

This chapter introduces the FPGA board ZCU104 to be verified in this work. ZCU104 was implemented in Aalto university by ECD group [9].

3.1 ZCU104 Board

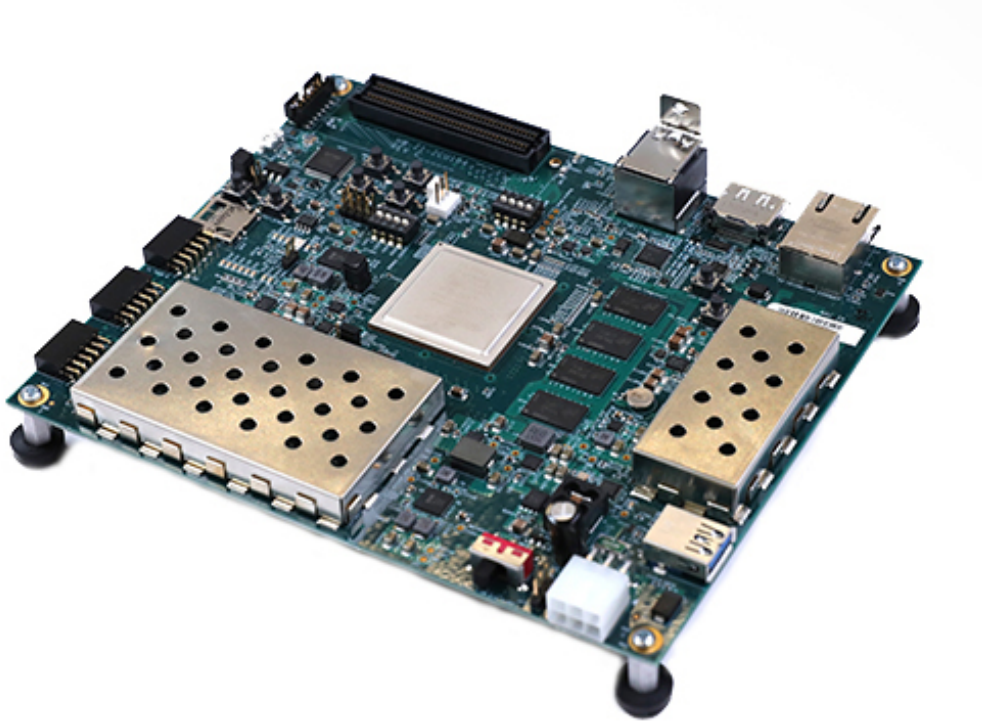


Figure 21: ZCU104 FPGA Board [67]

The ZCU104 Evaluation Board is a versatile development platform designed by Xilinx, based on the Zynq UltraScale+ MPSoC architecture. This board combines high-performance processing capabilities with rich hardware interfaces, making it suitable for applications in embedded systems, signal processing, and wireless communications. It has been widely adopted in academic and industrial settings for prototyping and verification of advanced digital systems, including 5G communication modules [67].

The ZCU104 board was introduced as part of the Zynq UltraScale+ MPSoC family, which builds on Xilinx's legacy of integrating programmable logic with high-performance processors. Released in the mid-2010s, the ZCU104 was designed to offer a balanced mix of processing power, reconfigurability, and interface options, making it a preferred choice for signal processing and AI applications [68, 69].

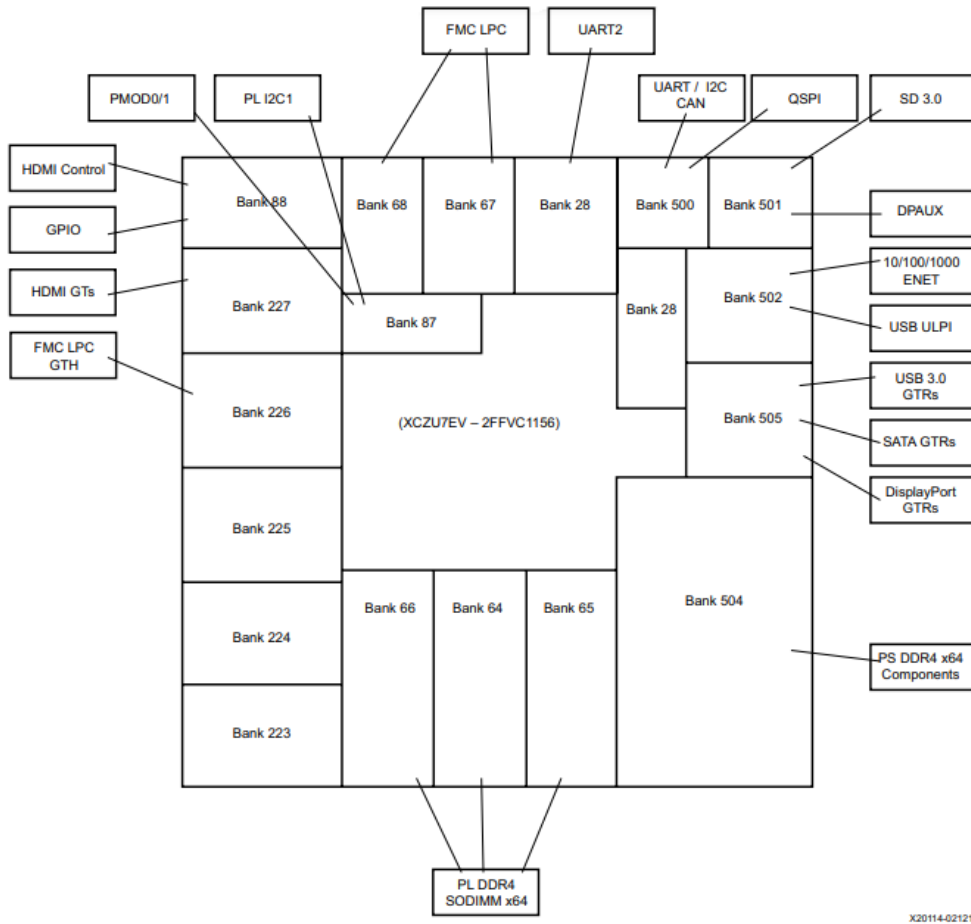


Figure 22: ZCU104 Block Diagram [70]

The ZCU104 board provides several essential features, including high-speed connectivity options like Gigabit Ethernet, USB 3.0, and PCIe, as well as memory interfaces such as DDR4 SDRAM for fast and reliable data storage [1]. Additionally, it offers expandable I/O options through FMC connectors and PMOD headers, allowing for easy integration of custom peripherals [67].

Compared to boards like the ZCU102, the ZCU104 provides a more compact and cost-effective solution, with slightly fewer resources, making it an excellent option for mid-range applications [71]. The ZCU104 emphasizes flexibility and heterogeneous processing, which makes it particularly well-suited for embedded applications where a combination of performance, adaptability, and integration is required [67]. The ZCU104 board stands out as an optimal platform for verifying a RISC-V-based 5G transmitter due to its unique blend of features and capabilities.

First, its high-performance processors, including ARM Cortex-A53 cores, provide a robust environment for running verification software [67]. These processors are well-equipped to handle tasks such as executing test harnesses, simulation frameworks, and performance analysis tools, which are essential for functional and system-level verification.

Second, the ZCU104’s relevance to 5G applications is evident in its high-speed interfaces and advanced DSP capabilities [72]. These features make it ideal for implementing and verifying 5G-specific components, including modulation techniques, beamforming techniques, and baseband signal processing.

While the ZCU104 may have slightly higher costs and power requirements compared to simpler boards [71], its advanced capabilities justify these trade-offs for complex tasks such as 5G transmitter verification [72]. The integration of programmable logic with high-performance processors enables efficient execution of both hardware implementation and software-driven testing, ensuring a comprehensive verification process.

3.2 A-Core Structure

The A-Core is a modular RISC-V processor developed at Aalto University [9], designed as an open-source platform to support research and practical applications. It is implemented using the Chisel hardware description language, which provides a high level of abstraction, enabling flexibility and ease of customization. The architecture of A-Core incorporates a variety of features, extensions, and peripherals, making it adaptable to diverse use cases and application domains.

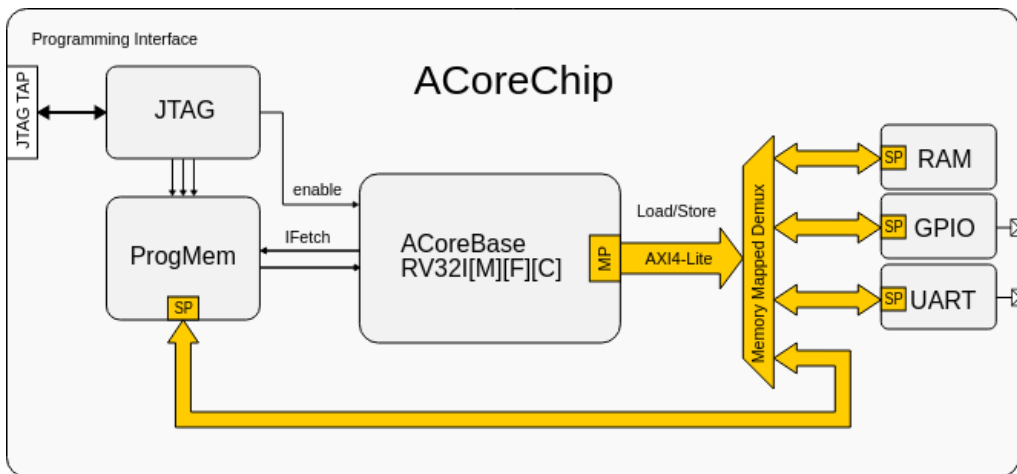


Figure 23: Top block diagram of acore chip. [73]

The RISC-V A-Core processor is built to support the RV32I instruction set as its foundation, with additional extensions designed to enhance its functionality for a wide range of applications. These extensions include the M extension for integer multiplication as well as division, the C extension for compressed instructions, which reduce the overall memory usage, and the F extension for float point operations [74]. The modular nature of the RISC-V architecture enables the A-Core processor to be easily customized for specific requirements, offering exceptional flexibility for applications such as IoT, AI, and telecommunications. This nature of A-Core processor with RISC-V architecture provides a robust solution for both general purpose and specialized demand.

To boost instruction execution efficiency, the A-Core chip incorporates a seven-stage pipeline. This pipeline consists of stages for instruction fetch, decode, execution, memory access, and write-back, among others[74]. By allowing multiple instructions to progress simultaneously through different stages of the pipeline, this design significantly reduces latency and increases throughput. This efficient pipeline architecture ensures that the processor can meet the high-performance demands of modern computational tasks [73, 9].

The A-Core processor further enhances its capabilities with the integration of custom accelerators tailored for specific tasks. For instance, dedicated accelerators for vector-matrix multiplication (VMM) make the processor particularly well-suited for AI workloads involving intensive mathematical computations. Additionally, accelerators for tasks such as power management and cryptography provide energy-efficient processing and secure data handling, addressing specialized needs across various domains [74].

Peripheral integration is another key strength of the A-Core design. The processor supports a range of peripherals through memory-mapped interfaces, including RAM, GPIO, and custom accelerators. Communication between the processor and these peripherals is managed using the AXI4-Lite protocol, ensuring reliable and efficient data exchange. Moreover, a dedicated JTAG interface is included for programming and debugging, enabling developers to monitor, test, and refine the system throughout the development and verification process [74].

For development and testing, the A-Core processor employs a co-simulation framework that facilitates seamless hardware-software integration. This framework supports repeated design and testing which reducing the time required to create a functional prototype. The design is also compatible with FPGA platforms, allowing real-world prototyping and thorough verification before deployment [73].

The flexibility of the RISC-V A-Core processor extends well beyond its instruction set and architectural features. Developers can configure key components such as instruction memory, debug signals, and ISA extensions to meet specific application requirements. This modular approach ensures that the processor can be optimized for a diverse range of workloads, whether it is deployed in low-power IoT devices, AI systems, or high-performance communication platforms [5]. This adaptability, coupled with its robust feature set, establishes the RISC-V A-Core as a versatile solution for modern computational challenges.

3.3 TheSyDeKick

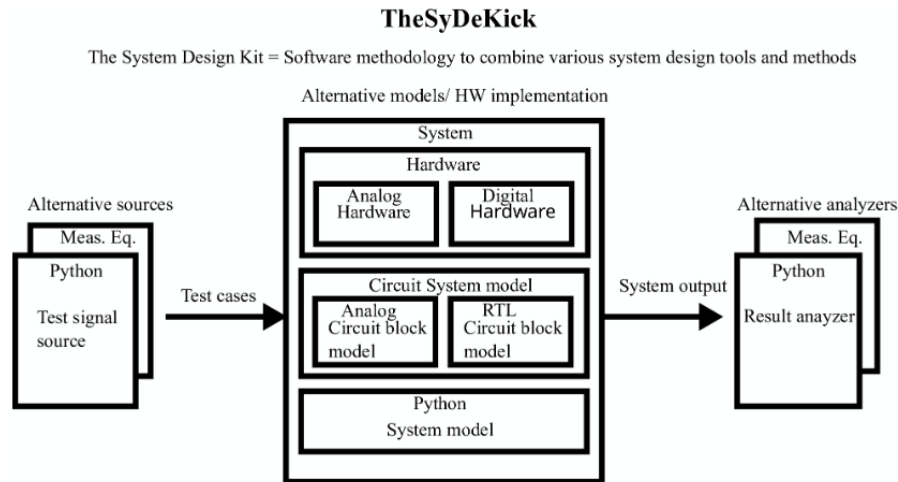


Figure 24: TheSyDeKick block diagram

TheSyDeKick is an open-source design and verification environment written in Python [75]. It serves as a versatile framework that integrates various system design tools and methods, enabling hardware-software co-simulation, verification, and system modeling. TheSyDeKick supports a modular architecture, composed of individual modules known as entities and a collection of helper scripts that facilitate tasks like parasitic extraction and netlisting [76].

The primary goal of TheSyDeKick is to streamline the design and verification of complex systems by combining hardware and software tools in a cohesive environment. Its flexibility and robust features make it suitable for research, education, and industrial applications, particularly in areas like AI, 5G signal processing, and power management [76].

Figure 24 illustrates the structure of TheSyDeKick [76]. It is organized into several layers, each supporting alternative models and hardware implementations [74]:

- **Alternative Sources:** The system accepts input from various test signal sources, including measurement equipment and Python-based test cases.
- **System Model:** The system model is implemented using Python, providing a high-level abstraction for design and verification.
- **Circuit System Model:** This layer includes analog circuit block models, RTL circuit models, and Python-based system models. These models represent different levels of abstraction and enable co-simulation between analog and digital components.
- **Hardware Implementation:** The system supports analog and digital hardware implementations, allowing seamless integration with physical test setups.

- **Alternative Analyzers:** Outputs from the system can be analyzed using measurement equipment or Python-based result analyzers.

TheSyDeKick offers a wide range of features that make it an essential tool for system design and verification. One of its most significant capabilities is hardware-software co-simulation, which enables simultaneous evaluation of both hardware and software components. This integrated approach allows designers to better understand the interaction between hardware and software, ensuring a smoother and more efficient development process [76].

Another key feature of TheSyDeKick is its integration of tools for various stages of system design and verification [75]. It provides support for RTL simulation, post-layout simulation, and FPGA-based verification, ensuring that designs meet both functional and physical requirements. These tools enable designers to perform comprehensive verification, from high-level simulations to low-level hardware implementations.

TheSyDeKick combines advanced simulation tools, hardware accelerators, and a robust verification framework to provide a comprehensive environment for system design and development. Its flexibility and wide range of features make it a valuable resource for research, education, and practical applications in modern system design.

3.4 A-Core Signal Transition Flow With DSP Accelerator

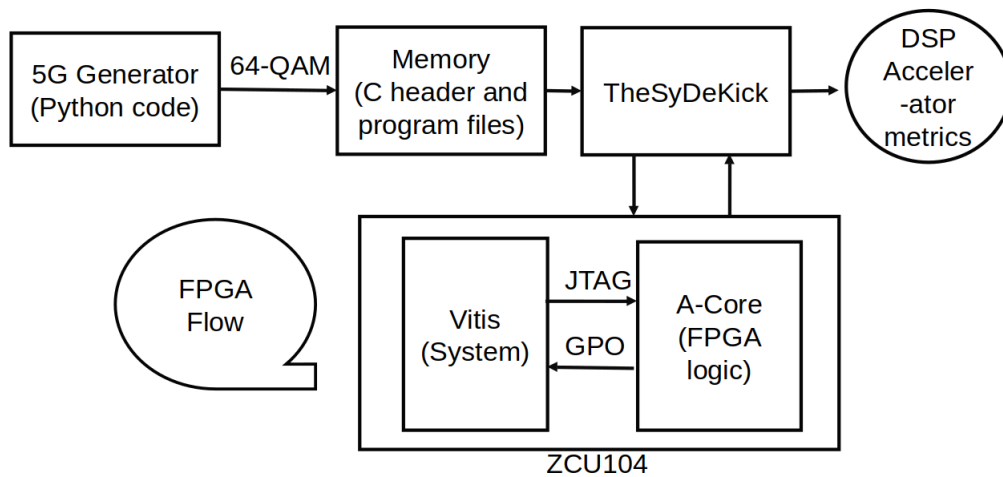


Figure 25: Signal transition flow in the verification process developed by ECD group in Aalto University

Figure 25 shows the 5G signal transmission flow developed in Aalto University [9] employed during the verification process on the ZCU104 platform. This flow demonstrates the interaction between software, hardware, and signal processing components to validate the performance of the 5G transceiver design.

The process begins with the 5G Generator, developed using Python code, which generates a 64-QAM signal. This entity is responsible for creating a modulated

signal, specifically employing 64-QAM [47]. The 64-QAM modulation level is widely used in 5G communication systems due to its ability to obtain spectral with high efficiency by encoding multiple bits per symbol.

The generated signal data is stored in the Memory component, represented as C header files and program files, which serve as the interface for transferring signal information to other processing units [9]. From the memory, the signal is passed to TheSyDeKick, a specialized processing framework that bridges the communication between the generated 5G data and the hardware components [76]. TheSyDeKick plays a critical role in transferring and managing data while supporting subsequent processing, including interaction with the DSP Accelerator [75].

The DSP Accelerator evaluates key performance metrics such as EVM, PSD, and ACLR, which are essential for determining the quality and reliability of the signal transmission [10]. By analyzing these metrics, the DSP Accelerator ensures the integrity of the design and its compliance with signal transfer standards.

At the core of the verification flow developed in Aalto University [9] lies the ZCU104 platform, which integrates software and hardware components to enable real-time signal processing. Within the ZCU104, two primary subsystems are utilized: Vitis and the A-Core. The Vitis System acts as the software execution environment, responsible for configuring the FPGA, controlling the signal flow, and managing data transfer. Meanwhile, the A-Core represents the FPGA logic implementation, where critical hardware operations such as modulation, filtering, and data processing are performed. Communication between the Vitis system and the FPGA logic is established through JTAG (Joint Test Action Group) and GPO (General Purpose Output) interfaces. These interfaces facilitate configuration, debugging, and monitoring of the FPGA to ensure proper functionality.

4 Verification

This chapter provides a detailed description of the verification conduction process based on the verification flow developed by ECD group in Aalto University [9]. The verification process focuses on two primary aspects: the functional verification of the DSP accelerator through simulation and the FPGA implementation verification, both of which were developed by the ECD group at Aalto University [9].

The functional verification aims to ensure that the DSP accelerator operates correctly and meets the expected functional requirements. This is achieved through simulation, which allows for comprehensive testing of the DSP accelerator under various conditions without requiring physical hardware.

In addition to simulation-based functional verification, the chapter also explores the verification of FPGA implementation, specifically the verification of the design customized for the A-Core processor. This implementation, developed by the ECD group [9], is tailored for FPGA platforms using the Vivado design suite. By detailing the specific Vivado design, this chapter provides insights into the customizations made to adapt the A-Core processor for FPGA-based verification.

4.1 Verification Flow Overview

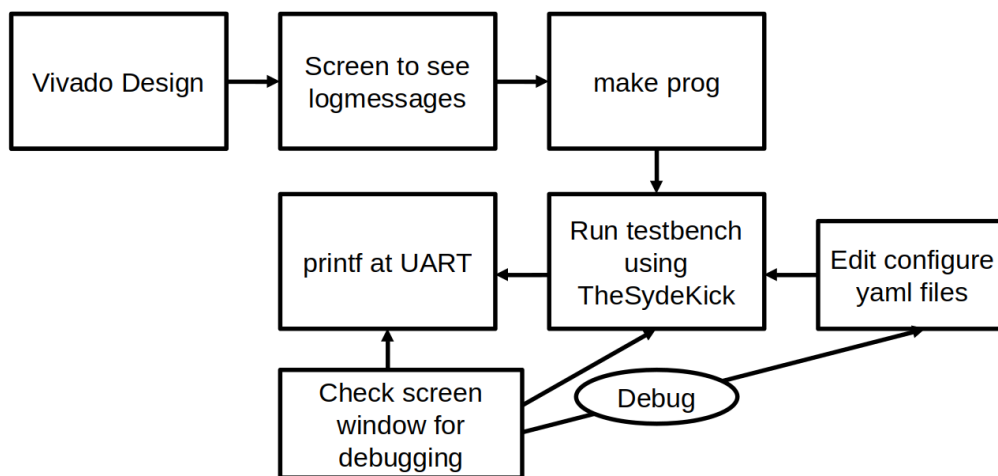


Figure 26: Verification flow developed by ECD group in Aalto University

As shown in Figure 26, the verification flow developed by ECD group in Aalto University [9] starts with creating the hardware design using tools such as Vivado. This step involves developing and preparing the design so it can be implemented on the FPGA device. Once the design is ready, it undergoes a screening process to check for log messages. These messages help verify that the design is functioning as expected at this stage. After screening, the design is programmed onto the FPGA device. This step ensures that the system is transferred to the physical hardware, making it ready for testing.

Testing is carried out using the simulation environment TheSydeKick. In this environment, the system's behavior is tested with predefined scenarios to confirm that it works correctly. To make the tests flexible and customizable, configuration files in the YAML format are used. These files allow users to adjust test conditions and parameters without changing the actual design. During testing, output messages are transmitted through Universal Asynchronous Receiver/Transmitter (UART), provide an easy way to confirm the execution situation of the system as well as to debug.

If any issues are found during testing, debugging tools are used to find and fix the problems. Debugging is an essential part of the process, as it allows for careful examination of the system to ensure it meets all requirements. The flow is repeated with the modification of the configuration and the testing setup until achieving the desired results.

This systematic approach ensures that the programme is reliable and performs as expected. By following these steps, we can verify and improve the designs, making the signal transition flow an important part of developing modern wireless communication systems.

4.2 Verification of FPGA Implementaion

The verification process involves configuring the system, adapting the testbench for FPGA testing, and running the test program using TheSyDeKick framework. This ensures that the A-Core processor design functions as intended and produces accurate outputs on the FPGA platform.

4.2.1 Open Source Version Setup

The open-source setup for the A-Core Vivado design involved navigating various repository versions and resolving multiple issues related to submodule access, design mismatches, and version alignment. Below is a detailed account of the setup process, the challenges encountered, and the steps taken to resolve them.

The A-Core Vivado project is part of an open-source repository with multiple branches and submodules. Initially, the work began by using the master branch of the repository. However, running the `make init` command resulted in errors due to access restrictions for certain submodule repositories. Without access, the initialization process could not complete successfully.

To address this issue, the submodule URL is updated to point to the correct version of the A-Core chip design. Despite this change, the error persisted. We then switched to the FPGA-specific branch of the repository, but the problem remained unresolved. Upon further investigation, discrepancies are discovered in the settings files, indicating version differences in the A-Core chip design.

To overcome these issues, the submodules which could not be initialized are deleted and only the necessary ones are retained. This adjustment allowed the `make init` command to complete, successfully initializing the A-Core Vivado project.

4.2.2 Modification of Vivado Design

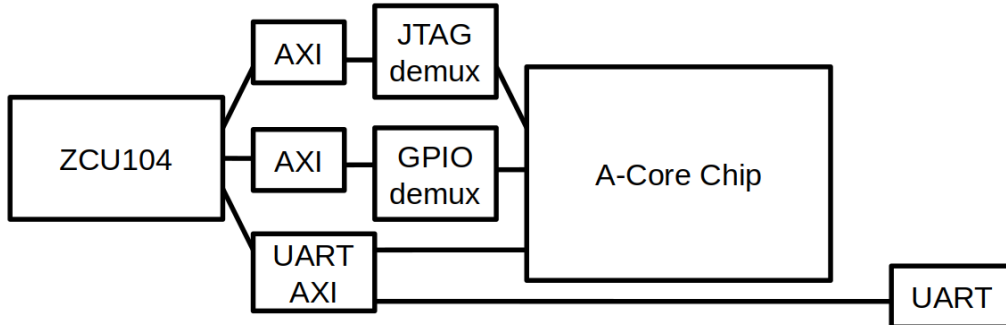


Figure 27: Vivado Design customized for FPGA Verification

The modified Vivado design is originally designed by ECD group in Aalto University, shown in Figure 27 [9], incorporates an additional AXI block configured with a 32-bit port. This AXI block facilitates seamless communication between the A-Core Chip and other components in the design. The integration ensures that the data flow between the modules and the ZCU104 board is efficient and error-free, enabling reliable system operation during the verification process.

4.2.3 Key Components in The Design

As illustrated by figure 27, the design contains several key components.

The AXI bus handles communication between the ZCU104 board and the A-Core Chip. The AXI bus connects different parts of the design and ensures that data flows smoothly across the system [74]. This connection allows the modules to work together, helping the A-Core Chip perform as expected during testing. By enabling reliable communication, the AXI bus plays an important role in checking and verifying the design's operation.

The JTAG Demux is another important component that routes the JTAG signals to the FPGA. JTAG is used for programming the FPGA and debugging the design [74]. With JTAG, developers can access the internal parts of the FPGA to test and fix any problems. This makes it easier to analyze the A-Core Chip's behavior and correct errors during the implementation process. The ability to debug at this level is essential for ensuring that the design works properly.

The design also includes a GPIO Demux, which manages input and output connections. The GPIO Demux makes sure that signals are correctly routed to connect with external devices or peripherals [74]. This helps in testing the design and checking how it interacts with other hardware. By handling these connections, the GPIO Demux simplifies the process of verifying the signals and ensures smooth communication with the outside environment.

Another important interface in the design is UART, which allows serial communication. The UART interface is useful for sending and receiving data during testing.

It helps monitor how the system behaves and observe the design outputs [74]. UART is also a key tool for debugging, as it provides real-time feedback, helping developers identify and fix any unexpected issues in the design.

At the center of the design is the A-Core Chip, which is the main part being tested. The A-Core Chip is built and connected to the ZCU104 board using the AXI, JTAG, GPIO, and UART interfaces.

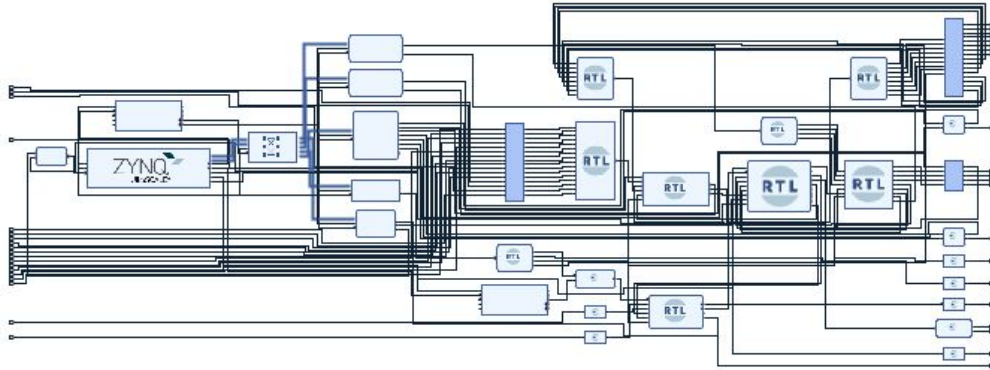


Figure 28: Vivado view of block diagram

Figure 28 shows the detailed block diagram in Vivado, which includes specific buffers and splitters [9].

4.2.4 Bitstream Generation Challenges

After the initialization, errors occurred during the `make bitstream` step. Through repeated testing, a potential issue related to the capitalization of A-Core chip file names was identified, likely caused by variations between different repository versions. Actions were taken to resolve the issue by renaming the corresponding files, but the error persisted.

Further analysis suggested that the existing design files were not fully aligned due to differences in Git repository versions. Additionally, access issues arose when attempting to clone an aligned version of the repository. After several consulting sessions, the access to the required version of the repository were granted to the work. With this aligned version, we successfully generated the bitstream.

4.2.5 Modifying Source Code and USB Issues

Following the bitstream generation, next step is uploading the bitstream to FPGA. To execute the operation, we switched to Vitis working environment under the ZCU104 folder to modify the source code according to the new design. However, an error occurred when attempting to connect to the Vitis serial terminal. After several consulting sessions, the access permissions were confirmed to be correct, but the serial terminal connection still failed.

Access to USB devices is essential for debugging during the verification process, as it allows the system to display execution logs and output messages. These logs provide critical insights into the behavior of the FPGA and help identify potential issues during testing. Therefore, successfully connecting to USB devices is a crucial step for effective debugging and verification.

An issue arose during the USB setup process when attempting to use the `screen` command to access the USB interface. The initial command, executed on the `hp8` machine, was as follows:

```
screen /dev/ttyUSB2 115200
```

However, this command returned an error:

```
No such file or directory
```

Upon investigating the available USB devices using the command `ls /dev/ttyUSB*`, the devices `tttyUSB7`, `tttyUSB8`, and `tttyUSB9` were identified. Attempts to use these devices also failed, with the `screen` command returning the message:

```
[screen is terminating]
```

Further investigation revealed that the issue was related to user permissions. Running the `groups` command on `hp8` showed that the user was only part of the `masters` group, which did not provide the required access to the USB devices. As a result, the USB devices could not be accessed for the verification process.

To resolve this issue, access to the required groups (`dialout` and `uucp`) was requested. After some time, the access permissions were activated. Once the updated permissions were confirmed, the setup was switched to the `benjy` machine. Running the `groups` command on `benjy` machine confirmed that the user had the appropriate permissions:

```
(user) uucp dialout masters
```

Using the `screen` command on `benjy` machine, the USB devices were successfully accessed. However, further challenges arose as the USB connections on `benjy` were configured differently compared to `hp8`. This was due to the fact that `hp8` and `benjy` were part of different setups, with separate FPGA connections and USB configurations.

To identify the correct USB device on `benjy`, the `ls /dev/ttyUSB*` and `usb-devices` commands were used. These commands helped locate the available USB ports and determine which one was associated with the FPGA. After targeting the correct device, the issue was resolved, and the setup proceeded without further errors.

In a summary, proper user permissions and the understanding of system-specific configurations are vital to ensure the connection to USB devices. Addressing these issues enabled successful access to the USB devices and ensured continuity in the verification.

4.2.6 Configuration

The initial step in the verification process is to align the system configurations with the Vivado design. This involves adding new YAML configuration files that reflect the updated design and ensure compatibility with the FPGA implementation. The configuration files define critical parameters such as clock alignment, data paths, and module interconnections. For example, the `clk` was changed to 250000000 ns to align with the Vivado design, and data paths were changed to FPGA testbench and correct hardware configuration. Proper alignment of the clock signals is essential to maintain synchronization across different components and to prevent timing-related errors during testing.

Testbench Preparation The next step is to adapt the testbench for FPGA testing. Using the Vitis environment, the existing testbench is loaded, and modifications are made to ensure compatibility with FPGA-specific requirements. This includes adding FPGA check bits and adapting the testbench logic to interact with the hardware. These modifications allow the testbench to verify the design functionality directly on the FPGA, rather than in a purely simulated environment.

Testing Program The testing program is executed using TheSyDeKick framework, which provides a streamlined environment for hardware-software co-simulation and verification. The following command is used to initiate the testing program:

```
make test_config=/home/.../test_configs/accnet.yml \  
sim_config=/home/.../sim_configs/fpga.yml
```

This command specifies the test configuration file (`accnet.yml`) and the FPGA simulation configuration file (`fpga.yml`). The test program interacts with the A-Core processor to validate its functionality, ensuring that the design meets its intended specifications.

4.2.7 Output Verification

The outputs of the A-Core processor are monitored via the UART interface, which facilitates serial communication between the FPGA and the host system. The following command is used to view the outputs [74]:

```
nc fpga6 8
```

Additionally, the outputs are displayed on the USB screen and through the UART interface, allowing for real-time observation and debugging. The UART interface provides valuable insights into the behavior of the design, enabling the identification of any anomalies or deviations from expected performance.

The verification began by initiating a 5G signal transmission simulation. A flag signal was sent through the DSP accelerator design, with the string configured as `reset`, `dut_sel`, `fpga`, or `asic`. This flag serves as a control signal to trigger and monitor the data flow through the design. However, the last bit of the signal did not arrive as expected, indicating a potential issue in the data transfer process.

To investigate the issue, a fourth bit was added to the flag signal, but the problem persisted. Further analysis revealed that the bit sequence was not arriving correctly at the destination. The clock was modified to ensure proper synchronization, but the sequence misalignment remained unresolved.

The next step involved checking which part of the design might be reversing the bit sequence. After identifying the affected modules, additional tests were conducted to isolate the problem.

To gain further insights, the setup was tested on hardware. The updated flags were uploaded to the FPGA board, and real-time observations were made using an oscilloscope to monitor the trigger signals. Simultaneously, the flag transfer status was checked at the UART port to determine whether the signal passed successfully.

The verification process, encompassing configuration alignment, testbench adaptation, and testing program execution, ensures the reliable operation of the A-Core processor on the FPGA platform. The use of TheSyDeKick framework and UART-based output monitoring provides a comprehensive approach to validating the design. This process highlights the importance of systematic testing and verification in achieving a robust and functional implementation.

4.3 Functional Verification of DSP Accelerator

In order to make sure the design being tested satisfies its functional and performance requirements, debugging and problem-solving are essential steps in the verification process. In complex systems, unexpected-arising issues are inevitable. These challenges can be caused by design errors, incorrect configuration setups, or unexpected behavior during simulation and testing.

Thus, effectively identifying and resolving such issues is crucial for verification. This chapter highlights the key problems encountered during the work process and the strategies employed to resolve them, offering valuable insights into the practical aspects of debugging in DSP accelerator verification.

4.3.1 Open Source Version Setup

During the setup of the open-source environment, the first issue was encountered when attempting to clone the repository using the `git clone` command. The following error message was displayed:

```
fatal: unable to access 'https://...' (the url of the repository):  
Peer's Certificate has expired.
```

This error occurred because the SSL certificate of the server hosting the repository had expired. SSL certificates are used to establish secure connections between clients and servers, and an expired certificate prevents the Git from verifying the authenticity of the server. As a result, Git denies access to the repository to avoid potential security risks.

To resolve this issue, the SSL verification was turned off temporarily. This was achieved by setting the `sslVerify` flag in the Git configuration file to `false`. The following command was used to disable SSL verification:

```
git config --global http.sslVerify "false"
```

This command modifies the global Git configuration file, telling Git to ignore SSL certificate checks. After applying this fix, the `git clone` command was executed again, and the cloning process proceeded without this error.

Although disabling SSL verification solved the issue, it is important to note that this turning off SSL verification reduces the security of the connection. It is generally recommended to address the center of the problem to ensure secure communication in the long term, such as by renewing the SSL certificate of the server.

Another issue while setting up the open-source environment occurred when following the steps in the README file. After running the `./configure && make` command in the `Entities/acoretets` directory, the process failed with the following error:

```
main.S:3:10: fatal error: test_macros.h: No such file or directory
3 | #include "test_macros.h"
  |           ^~~~~~
compilation terminated.
make[1]: *** [sim.elf] Error 1
[FATAL] ACoreSWCompiler: Error while compiling sources!
```

The error indicated that the file `test_macros.h` could not be found in the expected directory. Upon further investigation, it was discovered that the root cause of the issue was a mismatch in the version of the `acorechip` repository being used. The `acorechip` entity is actively under development, and certain versions may lack specific modules required for successful compilation. In this case, the branch of the repository being used did not include the necessary module containing `test_macros.h`.

To fix the issue, the version of the `acorechip` was switched to a repository that included the required modules while avoiding components unrelated to this verification task. The steps to resolve the issue were as follows:

- Updated the submodule URL for the `chisel/` directory to point to the correct branch of the repository.
- Reinitialized and updated the submodules by running the `./init_submodules` command in the `acorechip` folder.

Furthermore, the updated version is necessary to be ensured that it is synchronized in both the `acorechip` Vivado design folder and the `TheSDK` folder. After completing these updates, the missing module issue was resolved.

In the setup process of work with `acorechip`, it is important to pay attention to the correct repository version and having access to the necessary modules. Regular updates and careful management of repository branches are crucial to avoid compatibility issues and ensure a smooth verification workflow.

4.3.2 Simulation Command

The verification of the DSP accelerator involved rigorous debugging and simulation efforts, focusing on resolving software-related issues encountered during the process. The simulation was conducted under the directory `acoretests`, which provides the testing framework for the A-Core DSP accelerator, by running command:

```
make test_config=/home/.../accnet_test/accnet-test.yml
sim_config=/home/.../sim_configs/thesdk.yml
```

In the above command, "make" is used to launch the test and simulation process. "test_config" specifies the path to the test configuration file, which is "accnet-test.yml" for DSP accelerator functional verification. "accnet-test.yml" is a YAML file containing the test configuration for a specific simulation or test scenario related to the DSP accelerator test. This file defines key parameters such as input stimulators, expected outputs, and specific settings required for the test. "sim_config" provides the path to the simulation configuration file, which is "thesdk.yml" in this verification. "thesdk.yml" defines the settings for the simulation environment, such as simulation tools, backend configurations, clock settings, and environment variables. It ensures that the test is executed in a controlled environment that mirrors the intended operational conditions. By running the command, the process of verifying the functionality of the DSP accelerator would start within a specific test scenario and monitoring the outputs against the expected results.

4.3.3 Java.io.IOException

During the simulation, the error showed in figure 29 was encountered:

```
[error] java.io.IOException: Operation not permitted. file: /home/pro/mastrs/xhan/acore_thesdk/Entities/acorech
ip/chisel/urcblock/urc/target/streams/_global/dependencyPositions/_global/streams/update_cache_2.13/output_dsp
```

Figure 29: Java.io.IOException

This error typically occurs due to permission restrictions when accessing files or performing operations within the simulation environment. As the simulation was being executed on a shared server rather than a personal PC, there was uncertainty about applying administrative commands to resolve the issue.

To address the issue, the following steps were taken:

- **Permission Adjustment Attempts:** The possibility of using `sudo make` to bypass the permission error was considered. However, due to the shared nature of the server environment, there was hesitation about potentially causing conflicts or affecting other users' workflows.
- **SBT Commands for Cleanup and Rebuild:** Cache files sometimes would stop program from running smoothly. Thus, the following commands were used attempting to resolve the issue:

```
sbt clean
sbt publishLocal
```

These commands were executed under the `acorechip/chisel` directory. The purpose of `sbt clean` is to remove any residual files or build artifacts that might be causing the issue, while `sbt publishLocal` rebuilds and re-publishes the necessary dependencies locally. The error remained after this execution

- **Manually Cleanup Cache Folders:** The issue was resolved in the end by manually cleanup the directories given in the error log. Once the directories being called in the process were all cleaned up, the error is no longer reported.

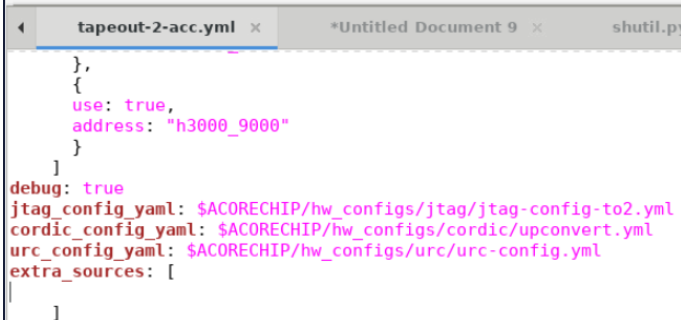
4.3.4 Issues With External Source

With the execution of the `make` command under the `acoretests` folder, a second error occurred during the process, as shown in Figure 30.

```
FileNotFoundError: [Errno 2] No such file or directory: '/home/pro/masters/xhan/acore_thesdk/Entities/acorechip/sv/xip
if_xip4133c_pkg.sv'
make: *** [Makefile:12: sim] Error 1
```

Figure 30: `FileNotFoundError` encountered during the simulation process.

The error indicates that certain files were not found in the specified directory. After investigation, it was determined that the issue stemmed from the hardware configuration file `tapeout-2-acc.yml`, which was attempting to call external sources that were inaccessible. The resolution involved modifying the hardware configuration file by removing the references to external sources. Specifically, the content under the `extra_sources` field was deleted, as illustrated in Figure 31.



```
tapeout-2-acc.yml x *Untitled Document 9 x shutil.py
},
{
  use: true,
  address: "h3000_9000"
}
]
debug: true
jtag_config_yaml: $ACORECHIP/hw_configs/jtag/jtag-config-to2.yml
cordic_config_yaml: $ACORECHIP/hw_configs/cordic/upconvert.yml
urc_config_yaml: $ACORECHIP/hw_configs/urc/urc-config.yml
extra_sources: [
|
]
```

Figure 31: Modification of the hardware configuration file to resolve the issue.

By performing this modification, the simulation was configured to operate exclusively within the local repository's available resources. This adjustment ensured that access problems and `FileNotFoundError` issues were effectively resolved, allowing the simulation to proceed without further interruptions.

4.3.5 Test Fail Troubleshooting

When the test fails without issues such as cache or access problems, it reports a FAIL, as shown in Figure 32.

```
21:41:03 [INFO] GenericTests: Test results:
accnet-test FAIL
```

Figure 32: `accnet-test` FAIL message during the simulation.

This failure can occur due to multiple reasons. The first approach to troubleshooting is to modify the configuration header file, such as the one shown in Figure 33.

```
// MODES
// - 0: TRANSMIT
// - 1: RECEIVE
#define ACCNET_MODE 1

// CONFIGS
// - 0: DMA -> URC -> CORDIC -> GPIO (reverse for RECEIVE)
// - 1: DMA -> CORDIC -> GPIO (reverse for RECEIVE)
#define ACCNET_CONFIG 0

// TARGETS
// - 0: GPIO
// - 1: RAM
// - 2: UART
#define ACCNET_TARGET 1
```

Figure 33: Configuration header file showing key parameters for DSP chain setup.

This header file defines key configurations for the test, including:

ACCNET_MODE: Configures the DSP chain mode between transmitter and receiver. Changing this mode allows for testing of both transmission and reception processes in the DSP chain.

ACCNET_CONFIG: Configures the DSP chain to include the Up Rate Converter (URC) or not. The two DSP chains are:

- *DMA* → *URC* → *CORDIC* → *GPIO*
- *DMA* → *CORDIC* → *GPIO*

In these configurations, the Direct Memory Access (DMA) module enables data transfer between devices without involving the Central Processing Unit (CPU), leaving the CPU available for other tasks. The URC modifies the sampling rate using decimation or interpolation techniques, reducing noise and enhancing the signal when the sampling rate is chosen appropriately. The CORDIC module performs coordinate rotation, crucial for applications such as phase and frequency adjustments. The General-Purpose Input/Output (GPIO) manages input and output connections. When the mode is set to receiver, the DSP chains reverse their direction.

ACCNET_TARGET: Determines the output target, which can be GPIO, RAM memory, or UART.

```

/*
End condition:
00 -> LOOP - DMA loops read and write addresses until it's reset
01 -> READ - Transfer until read address reaches end address
10 -> WRITE - Transfer until write address reaches end address
11 -> EITHER - Transfer until either read or write reaches its end address
*/
#define DMA_CONTROL_END_CONDITION_OFFSET          0x10
#define DMA_CONTROL_END_CONDITION_MASK          0x30000
#define DMA_CONTROL_END_CONDITION_INIT          0x3
*/

```

Figure 34: End condition definitions in the configuration header file.

Another possible cause of failure is an incorrectly defined end condition for the DMA. This can lead to the DMA continuing its operation indefinitely. The end condition definitions are located in the source code header file, as shown in Figure 34. The following options are available:

- 00: The loop ends only when a reset signal is received.
- 01: The loop ends when the read signal reaches its end.
- 10: The loop ends when the write signal reaches its end.
- 11: The loop ends when either the read or write signal reaches its end.

To resolve the issue, the end condition must be set appropriately based on the desired operation.

4.3.6 Outcome and Verification Continuation

After applying the necessary modifications to the configuration and header files, the error was resolved, and the simulation proceeded without further issues. It is crucial to use correct setting configuration parameters and aligning the DSP chain components. The simulation was successfully conducted, verifying the functionality of the DSP accelerator in the testing environment.

5 Result and Analysis

The results for the DSP accelerator functional verification were obtained for two kinds of DSP chain: one **without URC**, namely $DMA \rightarrow CORDIC \rightarrow GPIO$; the other **with URC**, namely $DMA \rightarrow URC \rightarrow CORDIC \rightarrow GPIO$. The results for both chains are under the key performance metrics EVM, PSD and ACLR. These metrics are analyzed in the context of performance requirements for modern communication systems, such as 5G.

5.1 DSP Chain without URC

Results for the chain $DMA \rightarrow CORDIC \rightarrow GPIO$.

5.1.1 EVM

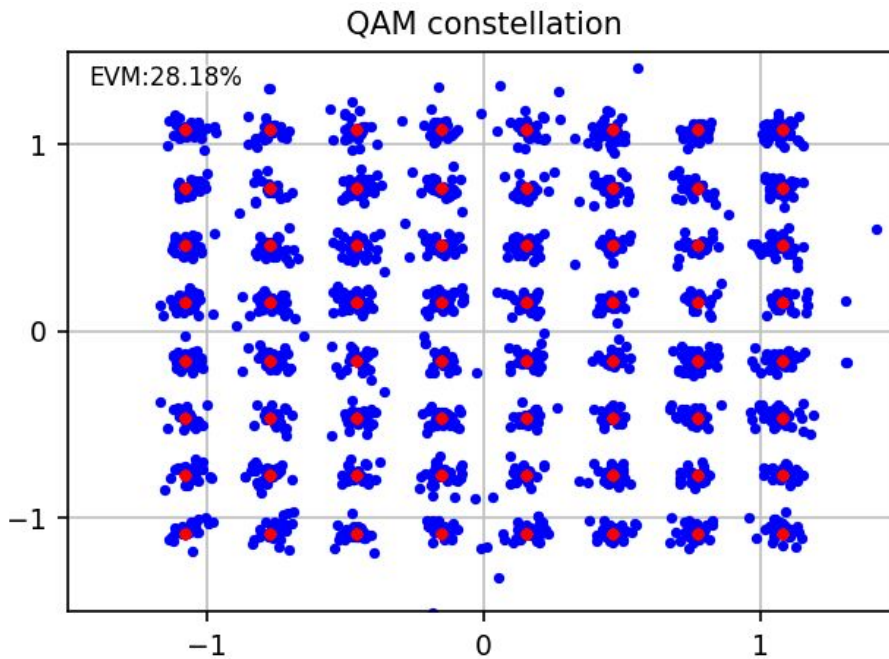


Figure 35: EVM result plot, without URC module.

Figure 35 illustrates the functional verification results of the constellation diagram for the 64-QAM DSP accelerator design implemented with ACoreChip. In the diagram, the red points represent the expected positions of the signal constellation points, while the blue points indicate the actual signal positions after transmission through the DSP chain using the DSP accelerator. The clustering of the blue points around the red points demonstrates that the transmitted signals closely approximate the original signals, validating the functionality of the DSP accelerator to a certain extent.

The EVM for this simulation is reported as 28.18%, which could be considered sensible within the context of functional verification and the processing chain being tested. But as it represents significant signal distortion, it can be concluded that the DSP chain introduces some distortion to the signal, and the requirement of further improvement are highlighted. On the other hand, when compared to 5G standard, EVM 28.28% in 64-QAM DSP chain exceeds the maximum permissible limits specified in the 3GPP 5G standards for different modulation levels [77]:

- **4-QAM:** Maximum EVM of 17.5%.
- **16-QAM:** Maximum EVM of 12.5%.
- **64-QAM:** Maximum EVM of 8%.
- **256-QAM:** Maximum EVM of 3.5%.

The reported EVM suggests that the DSP accelerator needs to be improved before it become suitable for high-order modulation levels commonly used in 5G. This high EVM could be attributed to factors such as implementation inaccuracies, noise, or suboptimal configurations. Improvements in filtering, algorithm optimization, or precision enhancements are required to address this issue.

5.1.2 PSD

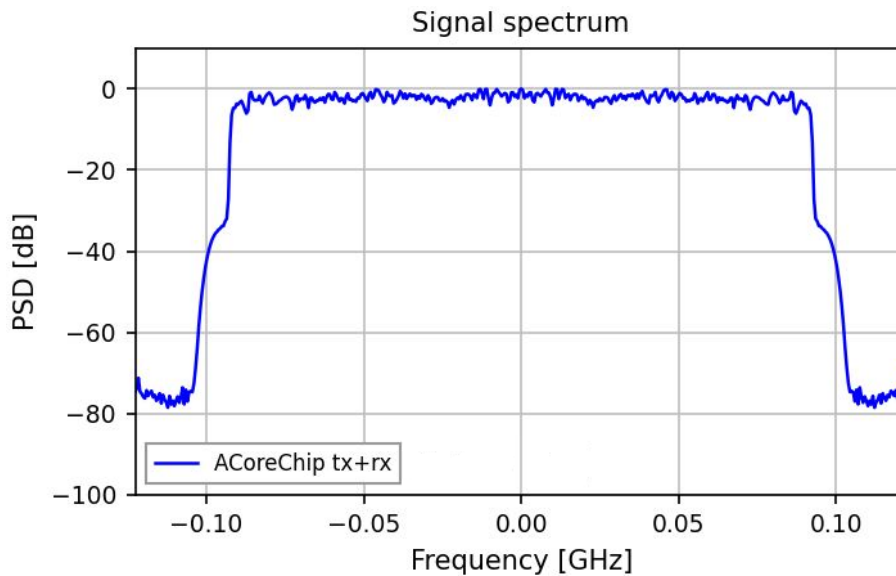


Figure 36: PSD result plot, without URC module.

Figure 36 presents the PSD curve of the signal. The x-axis represents the frequency in GHz, ranging from -0.125 GHz to 0.125 GHz, while the y-axis indicates the PSD in dB, spanning from -100 dB to 10 dB. The blue line represents the signal spectrum.

The PSD curve starts at approximately -80 dB near -0.125 GHz and begins to rise around -0.1 GHz. The signal power increases to around 0 dB, maintaining this level across the frequency range from approximately -0.1 GHz to 0.1 GHz. Beyond 0.1 GHz, the curve drops steeply back to approximately -80 dB.

Two distinct bumps are observed on the rising slopes at -0.1 GHz and 0.1 GHz. These irregularities may indicate potential signal leakage or distortions in the spectrum. Such features suggest areas requiring further investigation to improve spectral purity and meet system performance requirements.

The PSD result figure 36 provides a detailed view of the signal's power distribution across the frequency spectrum. The reported center frequency is reported as 0 , with a range of $[-500$ MHz, $+500$ MHz], and the bandwidth is 200 MHz, which matches 5G requirements and is typical for 5G channels.

5.1.3 ACLR

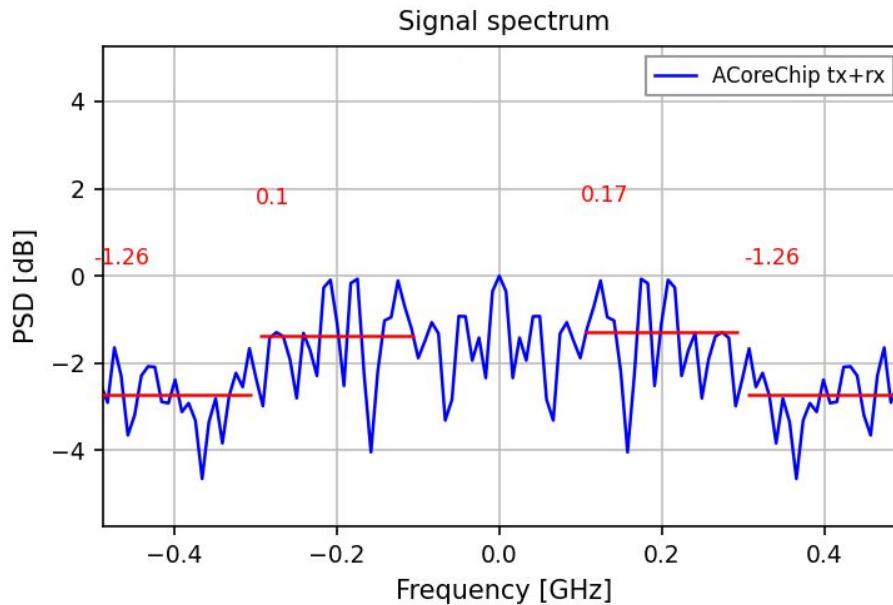


Figure 37: ACLR result plot, without URC module.

Figure 37 illustrates the signal spectrum for ACLR (Adjacent Channel Leakage Ratio) of the DSP accelerator, displaying the PSD in dB across a range of -4 dB to 4 dB over a frequency span of -0.5 GHz to $+0.5$ GHz. The blue curve represents the PSD of the transmitted and received signal through the DSP chain.

The PSD fluctuates between approximately -4 dB and -2 dB in the frequency range of -0.5 GHz to -0.3 GHz. In the central region, from -0.3 GHz to 0.3 GHz, the fluctuations widen, ranging from -4 dB to 0 dB, representing higher spectral activity. Beyond 0.3 GHz to 0.5 GHz, the fluctuations narrow again to a range of -4 dB to -2 dB.

The red annotations and lines indicate the calculated ACLR values in specific frequency regions:

- An ACLR value of -1.26 dB is observed in the ranges -0.5 GHz to -0.3 GHz and 0.3 GHz to 0.5 GHz.
- An ACLR value of 0.1 dB is noted within the range -0.3 GHz to -0.1 GHz.
- An ACLR value of 0.17 dB is recorded in the range 0.1 GHz to 0.3 GHz.

The output accurate ACLR values are reported as $[-1.258, 0.105, 0.0, 0.173, -1.261]$. The figure 37 and the values highlight the power leakage into adjacent channels, with notable deviations from the expected levels, particularly in the regions with positive ACLR values (0.1 dB and 0.17 dB). As ACLR quantifies power leakage into adjacent channels, which can cause interference, the result suggests the need for possible improvement in filtering to minimize adjacent channel interference and improve spectral efficiency.

On the other hand, according to 3GPP standards that the ACLR should be at least -45 dB for adjacent channels [77], the reported ACLR values indicate significant leakage, which does not meet the required spectral purity for 5G systems. Enhancing the design of filters and improving spectral shaping algorithms are necessary to reduce leakage and comply with the possible application in 5G signal transmission.

Additionally, the spectrum exhibits some irregular fluctuations in the PSD curve, particularly in the regions between the main band $[-0.1$ GHz, $+0.1$ GHz] and adjacent channels. These irregular fluctuations could be due to losses in filtering or spectral shaping issues in the DSP accelerator.

5.2 DSP Chain with URC

Results for the chain $DMA \rightarrow URC \rightarrow CORDIC \rightarrow GPIO$.

5.2.1 EVM

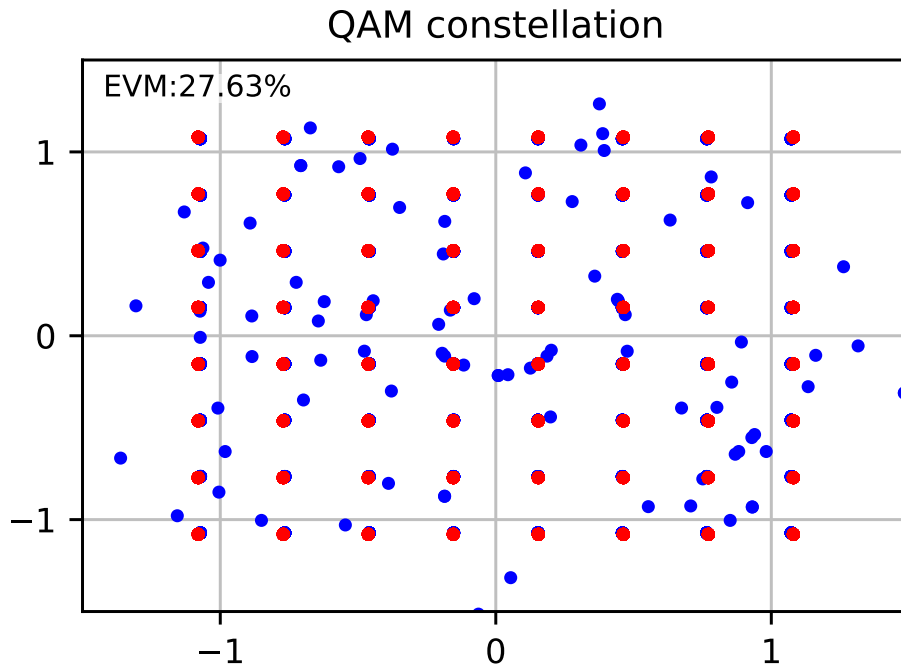


Figure 38: EVM result plot, with URC module.

Figure 38 illustrates the functional verification results of the constellation diagram for the 64-QAM DSP accelerator design implemented with ACoreChip through DSP chain including URC module. The diagram shows the red points as the ideal positions of the signal constellation, which are the expected values. The blue points show the actual positions of the signals after being transmitted through the DSP chain using the DSP accelerator. The blue points are clustered around the red points, indicating that the transmitted signals are similar to the original ones. This result demonstrates that the DSP accelerator can process and transmit signals with reasonable accuracy, validating its functionality in this configuration.

The EVM for this simulation is reported as 27.63%, which could be also considered sensible within the context of functional verification and the processing chain being tested. However, like the result of DSP chain without URC module indicates, it represents significant signal distortion, which can be concluded that the DSP chain introduces some distortion to the signal. Further improvements are required to expand the application of the DSP accelerator. Additionally, according to 3GPP standards for 5G systems, the simulated EVM of 27.63% exceeds the limitation of 64-QAM, indicating that the current configuration of the accelerator does not meet the requirements for high-order modulations typically used in 5G. This high EVM could be caused by distortion introduced in the DSP processing chain or implementation issues. To address this, improvements such as filter optimization or algorithm enhancements are necessary. Nevertheless, the DSP accelerator could be employed in other applications which are not so demanding as 5g transmission is.

5.2.2 PSD

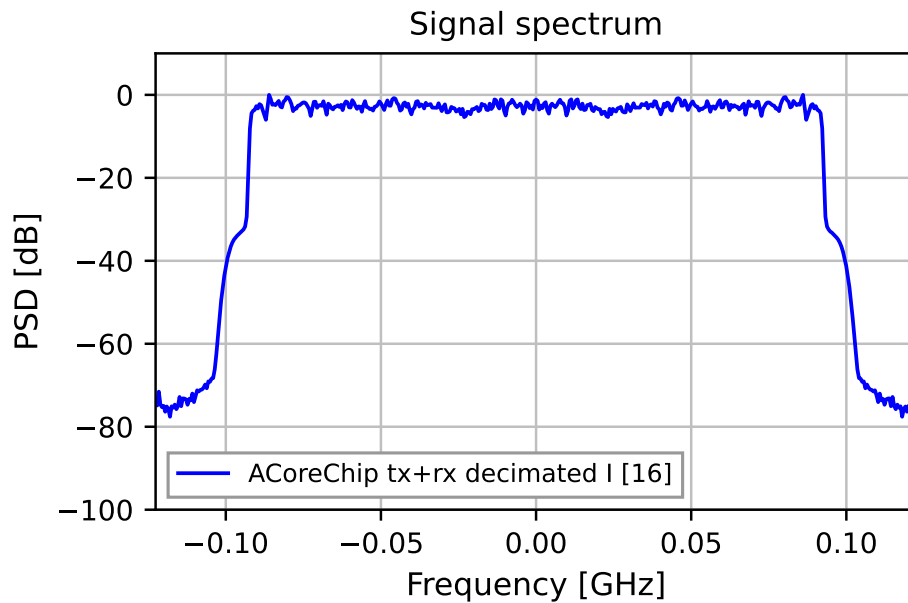


Figure 39: PSD result plot, with URC module.

Figure 39 shows the PSD curve of the signal. The x-axis represents the frequency in GHz, ranging from -0.125 GHz to 0.125 GHz, while the y-axis shows the PSD in dB, spanning from -100 dB to 10 dB. The blue line represents the signal spectrum.

The curve starts at around -80 dB near -0.125 GHz and begins to rise at approximately -0.1 GHz. The signal power increases to about 0 dB, maintaining this level from -0.1 GHz to 0.1 GHz. Beyond 0.1 GHz, the curve sharply drops back to around -80 dB.

Two noticeable bumps are seen on the rising slopes at approximately -0.1 GHz and 0.1 GHz. These irregular fluctuations suggest possible signal leakage or distortions within the spectrum. Such features indicate areas that might require further analysis and improvements to enhance spectral purity and meet system performance standards.

The PSD result also confirms a center frequency of 0 GHz and a bandwidth of 200 MHz. This aligns with 5G requirements, where channel bandwidths typically range from 20 MHz to 400 MHz. The reported values verify the ability of the DSP accelerator to support bandwidths commonly used in 5G communication systems.

5.2.3 ACLR

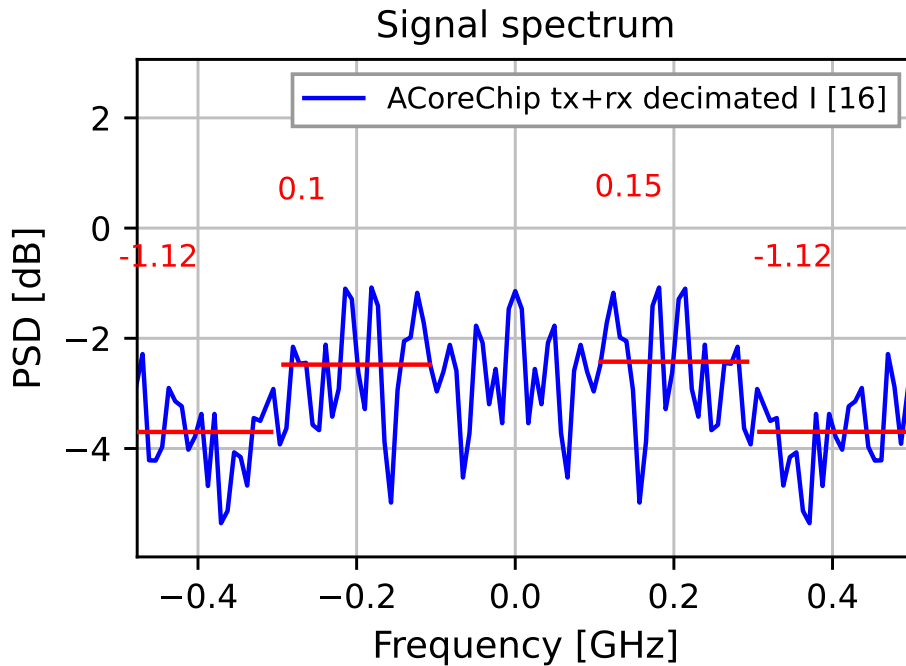


Figure 40: ACLR result plot, with URC module.

Figure 40 shows the signal spectrum for ACLR of the DSP accelerator chain with URC module. The plot displays the PSD in dB, ranging from -6 dB to 2 dB, over a frequency range of -0.5 GHz to $+0.5$ GHz. The blue curve represents the PSD of the signal as it passes through the DSP chain.

The PSD fluctuates between -5 dB and -2 dB within the frequency range of -0.5 GHz to -0.3 GHz. In the central band, from -0.3 GHz to 0.3 GHz, the fluctuations become wider, ranging from -5 dB to -1 dB, indicating increased spectral activity. Beyond 0.3 GHz, the PSD narrows again, oscillating between -5 dB and -2 dB up to 0.5 GHz.

Red annotations and lines in the graph indicate ACLR values calculated in specific frequency ranges:

- An ACLR value of -1.12 dB is observed in the frequency ranges -0.5 GHz to -0.3 GHz and 0.3 GHz to 0.5 GHz.
- An ACLR value of 0.1 dB is noted between -0.3 GHz and -0.1 GHz.
- An ACLR value of 0.15 dB is calculated for the frequency range 0.1 GHz to 0.3 GHz.

The reported ACLR values, $[-1.120, 0.101, 0.0, 0.150, -1.118]$, reflect the degree of power leakage into adjacent channels. The positive ACLR values, 0.1 dB and 0.15 dB, highlight significant leakage that could interfere with neighboring frequency

bands. Since ACLR quantifies power leakage into adjacent channels, these results emphasize the need for better filtering and spectral shaping to reduce interference and enhance spectral consistency.

According to the 3GPP standard, ACLR values must meet a minimum requirement of -45 dB for adjacent channels [77]. The reported values significantly deviate from this standard, indicating substantial power leakage and insufficient spectral purity for 5G applications. To achieve compliance, improvements in filter design and spectral shaping techniques are necessary.

Similar to the results observed for the DSP chain without the URC module, the PSD spectrum exhibits irregular fluctuations, particularly in the transition regions between the main band $[-0.1$ GHz, $+0.1$ GHz] and adjacent channels. These irregularities might be caused by filtering inefficiencies or spectral shaping issues within the DSP accelerator. Further improvement and investigation are required to address these challenges and enhance overall system performance.

5.3 Comparison and Analysis of Results with and without URC

The results of the DSP accelerator functional verification reveal differences in performance between the two DSP chains: one without URC and the other with it. These differences are most apparent in EVM, highlighting the impact of the URC on the overall signal quality. This section provides a detailed analysis of the results, explaining the observed bias and the reasons behind it.

5.3.1 Results Comparison

In the simulation, the DSP chain without URC had an EVM of 28.18%, while the DSP chain with URC produced an EVM of 27.63%; additionally, there are much more data points in the EVM figure 35 (without URC) than in the EVM figure 38 (with URC). Despite these differences on EVM, no significant biases are observed with PSD or ACLR curve results. Although the difference in EVM is small, the lower value in the chain with URC indicates slightly better signal performance. EVM quantifies the distortion between the transmitted and received signals, measuring how closely the actual signals match the ideal constellation points. A lower EVM indicates lower distortion and improved accuracy in signal representation.

5.3.2 Analysis of Impacts

The improvement in EVM on the chain with URC can be attributed to the sample rate conversion process performed by the URC. Sample rate conversion, which involves interpolation (upsampling) or decimation (downsampling), helps align the sampling rates between different processing stages.

The URC employs filters like Finite Impulse Response (FIR) filters and Cascaded Integrator-Comb (CIC) filters to perform sample rate conversion. These filters play a critical role in smoothing the signal, reducing high-frequency noise, and

preventing aliasing during interpolation or decimation. The FIR filters ensure precise signal processing, and CIC filters handle rough sample rate changes with calculation efficiency. These filters in the URC are possibly the reason of the observed improvement in EVM by reducing unwanted distortions. Additionally, the process also leads to fewer data points compared to the DSP chain without URC module.

However, flaws in filter design may still introduce errors that limit the improvement in EVM. As a result, these distortions remain a factor in the signal observed in the URC-enabled DSP chain figures even though are reduced by the URC module.

Furthermore, the Coordinate Rotation Digital Computer (CORDIC) module, which presents in both DSP chains to performs frequency shifting and phase rotation, could also have impacts on reducing the accuracy of frequency adjustments and the overall signal quality, which additionally, could cause the defects in PSD and ACLR results.

5.4 Summary

The comparison of results highlights the subtle but important role of the URC in the DSP accelerator's signal processing chain. By performing sample rate conversion and incorporating filters, the URC reduces distortions and improves signal fidelity, resulting in a lower EVM. However, the processes of interpolation, decimation, and filtering introduce their own limitations, which prevent a more significant improvement in EVM performance.

Both DSP chains remain space for improvement. Future work should focus on optimizing the FIR and CIC filters in the URC, refining the CORDIC module's precision, and investigating the effects of sample rate conversion parameters. These enhancements will help achieve further reductions in EVM and better compliance with modern communication standards, such as 5G.

6 Conclusion

This thesis focuses on the verification of the RISC-V DSP accelerator and related FPGA implementations within a 5G transceiver system. The work was carried out in the context of existing designs and verification frameworks developed by the Electronic Circuit Design (ECD) group. Specifically, the thesis concentrates on functional verification of the DSP accelerator through simulation and verification of data transmission within an FPGA-based platform implemented on the ZCU104 board with the A-Core processor.

The primary contribution of this work lies in providing valuable insights into the performance and the functionality of the DSP accelerator through functional verification. This work also verified a successful FPGA-based data transmission on the ZCU104 platform, following the verification flow implemented by Aalto University. In the practical perspective, this work provided a thorough description for the verification processes and showed how to fix common errors, making it easier for future researchers to run the process.

The DSP accelerator was functionally verified in a simulation environment, with key performance metrics such as EVM, PSD, and ACLR analyzed to evaluate its signal processing capabilities. These analyses showed the effects of specific design choices, such as the inclusion of the URC, on the system's performance. This detailed functional verification forms a critical step in presenting the performance of the RISC-V DSP accelerator with A-Core processor.

While the FPGA implementation and verification environment, including the A-Core processor and the ZCU104 platform, were developed by the ECD group, this thesis emphasizes the practical challenges encountered during the verification process, including issues related to open-source setup, configuration mismatches, and debugging errors. Systematic approaches to address these challenges, such as modifying configuration files, adapting testbenches, and resolving access issues, are thoroughly documented. These methods and experiences serve as a guide for future researchers working on similar verification tasks, streamlining their efforts and reducing potential roadblocks.

In conclusion, this thesis contributes to the field by demonstrating functional verification of the DSP accelerator through simulation and verifying FPGA-based data transfer on the ZCU104 platform. The work builds upon the existing designs and frameworks of the ECD group, and the results provide a foundation for future verification efforts and possible DSP accelerator improvement direction. This thesis emphasizes the importance of precise and structured verification methodologies in ensuring the reliability and performance of RISC-V-based hardware systems for modern communication applications.

References

- [1] What is instruction set architecture (ISA)? Accessed: 11.10.2024. [Online]. Available: <https://www.lenovo.com/us/en/glossary/instruction-set-architecture>
- [2] H. Karaki, H. Akkary, and S. Shahidzadeh, "X86-arm binary hardware interpreter," in *2011 18th IEEE International Conference on Electronics, Circuits, and Systems*, 2011, pp. 145–148.
- [3] E. Blem, J. Menon, and K. Sankaralingam, "Power struggles: Revisiting the risc vs. cisc debate on contemporary arm and x86 architectures," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, 2013, pp. 1–12.
- [4] RISC-V Architecture: A Comprehensive Guide to the Open-Source ISA. Accessed: 11.10.2024. [Online]. Available: <https://www.wevolver.com/article/risc-v-architecture>
- [5] How RISC-V standards are changing the world. Accessed: 11.10.2024. [Online]. Available: <https://betanews.com/2024/11/20/how-risc-v-standards-are-changing-the-world-qa>
- [6] Applications of Digital Signal Processing. Accessed: 10.12.2024. [Online]. Available: <https://www.geeksforgeeks.org/applications-of-digital-signal-processing>
- [7] Digital Signal Processing: What Is It And What Are Its Applications. Accessed: 10.12.2024. [Online]. Available: <https://thinkpalm.com/blogs/digital-signal-processing-what-is-it-what-are-its-applications>
- [8] S. Y. Neyaz, I. Saxena, N. Alam, and S. A. Rahman, "Fpga and asic implementation and comparison of multipliers," in *2020 International Symposium on Devices, Circuits and Systems (ISDCS)*, 2020, pp. 1–4.
- [9] A. Korsman, V. Hirvonen, O. Simola, A. Tarkka, M. Kosunen, and J. Rynänen, "End-to-end multi-target verification environment for a risc-v microprocessor," in *2023 19th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2023, pp. 1–4.
- [10] R. L. Allen and D. Mills, *Signals: Analog, Discrete, and Digital*, 2003, pp. 1–108.
- [11] 5G - Vision for the next generation of connectivity. Accessed: 11.10.2024. [Online]. Available: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/whitepaper_5g_vision_for_the_next_generation_of_connectivity.pdf

- [12] A Global Perspective of 5G Network Performance. Accessed: 11.10.2024. [Online]. Available: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/srg_5g_benchmark_study_-_final.pdf
- [13] A. Waterman, “Design of the risc-v instruction set architecture,” 2016. [Online]. Available: <https://api.semanticscholar.org/CorpusID:63861396>
- [14] Software-defined processors: the promise of RISC-V. Accessed: 11.10.2024. [Online]. Available: <https://next.redhat.com/2024/10/09/software-defined-processors-the-promise-of-risc-v>
- [15] M. Mehendale, S. Sherlekar, and G. Venkatesh, “Techniques for low power realization of fir filters,” in *Proceedings of ASP-DAC’95/CHDL’95/VLSI’95 with EDA Technofair*, 1995, pp. 447–450.
- [16] E. Cui, T. Li, and Q. Wei, “Risc-v instruction set architecture extensions: A survey,” *IEEE Access*, vol. 11, pp. 24 696–24 711, 2023.
- [17] Y. Zhao, H. Kuang, Y. Sun, Z. Yang, C. Chen, J. Meng, and J. Han, “Enhancing risc-v vector extension for efficient application of post-quantum cryptography,” in *2023 IEEE 34th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2023, pp. 10–17.
- [18] K. Shi, S. Xu, Y. Diao, D. Boland, and Y. Bao, “Encore: Efficient architecture verification framework with fpga acceleration,” 02 2023, pp. 209–219.
- [19] A. K. Singh and S. K. Shukla, “Complete and efficient verification for a RISC-V processor using formal verification,” in *Proceedings of the IEEE International Conference on VLSI Design (VLSID)*, 2023, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/10546693>
- [20] M. N. Rizzi, N. Zidaric, L. Batina, and N. Mentens, “Optimised aes with risc-v vector extensions,” in *2024 27th International Symposium on Design & Diagnostics of Electronic Circuits & Systems (DDECS)*, 2024, pp. 57–60.
- [21] H. Liang, N. Tan, Y. Ren, W. Hu, J. He, and J. Xia, “Python based testbench for coverage driven functional verification,” in *2022 7th International Conference on Integrated Circuits and Microsystems (ICICM)*, 2022, pp. 361–365.
- [22] S. K. Goel and E. Cerny, “Design and verification of risc-v cpu based on hls and uvm,” in *Proceedings of the IEEE International Conference on VLSI Design*, 2021, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/9574575>
- [23] 50 Years of FFT Algorithms and Applications. Accessed: 10.12.2024. [Online]. Available: <https://link.springer.com/article/10.1007/s00034-019-01136-8>
- [24] S. W. Smith, *The Scientist and Engineer’s Guide to Digital Signal Processing*. San Diego, California: California Technical Publishing, 1997. [Online]. Available: <https://www.dspguide.com/>

- [25] V. S. Chakravarthi and S. R. Koteswar, *Introduction to Systems*. Cham: Springer Nature Switzerland, 2023, pp. 1–15. [Online]. Available: https://doi.org/10.1007/978-3-031-36242-2_1
- [26] L. E. Frenzel, “Chapter 8 - cell phones: It is now possible to do anything wirelessly: Talk, text, email, web browse, games, whatever,” in *Electronics Explained (Second Edition)*, second edition ed., L. E. Frenzel, Ed. Newnes, 2018, pp. 195–216. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780128116418000084>
- [27] L. L. Hanzo, S. X. Ng, T. Keller, and W. Webb, *QAM Basics*, 2004, pp. 1–1.
- [28] What is the physical significance of negative frequencies? Accessed: 10.12.2024. [Online]. Available: <https://dsp.stackexchange.com/questions/431/what-is-the-physical-significance-of-negative-frequencies>
- [29] J. Gilb, *Wireless Dictionary*. Wiley, 2005. [Online]. Available: <https://books.google.fi/books?id=zhnYAAAAMAAJ>
- [30] QAM. Accessed: 10.12.2024. [Online]. Available: <https://www.qsfptek.com/network-glossary/qam.html>
- [31] Y. Koizumi, K. Toyoda, M. Yoshida, and M. Nakazawa, “1024 qam (60 gbit/s) single-carrier coherent optical transmission over 150 km,” *Opt. Express*, vol. 20, no. 11, pp. 12 508–12 514, May 2012. [Online]. Available: <https://opg.optica.org/oe/abstract.cfm?URI=oe-20-11-12508>
- [32] Wi-Fi 6 fundamentals: What is 1024-QAM. Accessed: 10.12.2024. [Online]. Available: <https://www.commscope.com/blog/2018/wi-fi-6-fundamentals-what-is-1024-qam>
- [33] C. Bockelmann, N. Pratas, H. Nikopour, K. Au, T. Svensson, C. Stefanovic, P. Popovski, and A. Dekorsy, “Massive machine-type communications in 5g: physical and mac-layer solutions,” *IEEE Communications Magazine*, vol. 54, no. 9, pp. 59–65, 2016.
- [34] L. Punitha, S. P. Sugumar, and P. H. Rao, “Analysis of rf transceiver for 5g applications,” *2019 URSI Asia-Pacific Radio Science Conference (AP-RASC)*, pp. 1–4, 2019. [Online]. Available: <https://api.semanticscholar.org/CorpusID:195223830>
- [35] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez, “Millimeter wave mobile communications for 5g cellular: It will work!” *IEEE Access*, vol. 1, pp. 335–349, 2013. [Online]. Available: <https://ieeexplore.ieee.org/document/6515173>
- [36] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, “What will 5g be?” *IEEE Journal on Selected Areas in*

- Communications*, vol. 32, no. 6, pp. 1065–1082, 2014. [Online]. Available: <https://ieeexplore.ieee.org/document/6824752>
- [37] T. Norp, “5g requirements and key performance indicators,” *Journal of ICT Standardization*, vol. 6, no. 1-2, pp. 15–30, 2018.
- [38] How Error Vector Magnitude (EVM) Measurement Improves Your System-Level Performance. Accessed: 10.12.2024. [Online]. Available: <https://www.analog.com/en/resources/technical-articles/how-evm-measurement-improves-system-level-performance.html>
- [39] What are the factors influencing Error Vector Magnitude (EVM)? Reference: <https://www.physicsforums.com/threads/what-are-the-factors-influencing-error-vector-magnitude-evm.1005652/>. Accessed: 10.12.2024. [Online]. Available: <https://www.physicsforums.com/threads/what-are-the-factors-influencing-error-vector-magnitude-evm.1005652/>
- [40] H. Kim, *5G Wireless Communication System Parameters and Requirements*, 2020, pp. 13–20.
- [41] What is the PSD? Accessed: 10.12.2024. [Online]. Available: <https://vru.vibrationresearch.com/lesson/what-is-the-psd/>
- [42] R. kaur and A. Singh, “Papr and spectrum analysis of 4g and 5g techniques,” in *2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*, 2019, pp. 18–21.
- [43] S. Thangamayan, M. D. Walunjkar, D. Kumar Ray, M. Venkatesan, A. Banik, and K. P. Amrutkar, “5g modulation technique comparisons using simulation approach,” in *2022 3rd International Conference on Intelligent Engineering and Management (ICIEM)*, 2022, pp. 848–856.
- [44] Adjacent Channel Leakage Power Ratio (ACLR). Accessed: 19.11.2024. [Online]. Available: <https://itecspec.com/spec/3gpp-37-141-6-6-4-adjacent-channel-leakage-power-ratio-aclr>
- [45] What id ACLR. Accessed: 11.10.2024. [Online]. Available: <https://www.ednchina.com/technews/22985.html>
- [46] J. Ordosgoitti and A. Eroglu, “Multitone digital predistortion method of rf power amplifiers for 5g systems,” in *2022 IEEE International RF and Microwave Conference (RFM)*, 2022, pp. 1–4.
- [47] TS 138 104 - V18.7.0 - 5G; NR; Base Station (BS) radio transmission and reception. Accessed: 12.1.2024. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138100_138199/138104/18.07.00_60/ts_138104v180700p.pdf
- [48] D. Hamidovic, J. Markovic, P. Preyler, C. Mayer, M. Huemer, and A. Springer, “A Comparison of All-Digital Transmitter Architectures for Cellular Handsets,” in *2019 Austrochip Workshop on Microelectronics (Austrochip)*, 2019, pp. 14–20.

- [49] A. Kumar and D. K. Jhariya, “Low-noise amplifier design: An open-source perspective,” in *2023 2nd International Conference on Paradigm Shifts in Communications Embedded Systems, Machine Learning and Signal Processing (PCEMS)*, 2023, pp. 1–6.
- [50] Understanding the Rf Transmitter and Receiver Block Diagram: A Comprehensive Guide. Accessed: 12.1.2024. [Online]. Available: <https://schempal.com/rf-transmitter-and-receiver-block-diagram>
- [51] H. Sadrozinski and J. Wu, *Applications of Field-Programmable Gate Arrays in Scientific Research*, 04 2016.
- [52] M. H. Assaf, S. R. Das, W. Hermas, and W. B. Jone, “Promising complex asic design verification methodology,” in *2007 IEEE Instrumentation & Measurement Technology Conference IMTC 2007*, 2007, pp. 1–6.
- [53] F. Plasencia-Balabarca, E. Mitacc-Meza, M. Raffo-Jara, and C. Silva-Cárdenas, “Alternative functional verification methodology for low and medium level designs (applied to an aes encryption module),” in *2018 IEEE 19th Latin-American Test Symposium (LATS)*, 2018, pp. 1–4.
- [54] FPGA Verification. Accessed: 11.10.2024. [Online]. Available: <https://verificationacademy.com/topics/fpga-verification/>
- [55] J. J. Rodríguez-Andina, M. D. Valdés-Peña, and M. J. Moure, “Advanced features and industrial applications of fpgas—a review,” *IEEE Transactions on Industrial Informatics*, vol. 11, no. 4, pp. 853–864, 2015.
- [56] I. Kuon and J. Rose, “Measuring the gap between fpgas and asics,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, 2007.
- [57] M. Hutton, R. Yuan, J. Schleicher, G. Baeckler, S. Cheung, K. K. Chua, and H. K. Phoon, “A methodology for fpga to structured-asic synthesis and verification,” in *Proceedings of the Design Automation & Test in Europe Conference*, vol. 2, 2006, pp. 1–6.
- [58] A. Evans, A. Silburt, G. Vrckovnik, T. Brown, M. Dufresne, G. Hall, T. Ho, and Y. Liu, “Functional verification of large asics,” in *Proceedings 1998 Design and Automation Conference. 35th DAC. (Cat. No.98CH36175)*, 1998, pp. 650–655.
- [59] J. M. Ludden, R. J. Bowman, and J. M. Emmert, “Practical concurrent asic and system design and verification,” in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 1997, pp. 11–14. [Online]. Available: <https://ieeexplore.ieee.org/document/582412>
- [60] S. K. Goel and E. Cerny, “Promising complex asic design verification methodology,” in *Proceedings of the IEEE International Symposium*

- on Circuits and Systems*, 2007, pp. 169–172. [Online]. Available: <https://ieeexplore.ieee.org/document/4258344>
- [61] Application-Specific Integrated Circuit (ASIC). Accessed: 11.10.2024. [Online]. Available: <https://piembstech.com/application-specific-integrated-circuit-asic/>
- [62] N. K. Doshi, S. Suryawanshi, and G. N. Kumar, “Development of generic verification environment based on uvm with case study on hmc controller,” in *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 2016, pp. 550–553.
- [63] W. Chen, S. Ray, J. Bhadra, M. Abadir, and L.-C. Wang, “Challenges and trends in modern soc design verification,” *IEEE Design & Test*, vol. 34, no. 5, pp. 7–22, 2017.
- [64] Comparing FPGAs, Structured ASICs, and Cell-Based ASICs. Accessed: 11.10.2024. [Online]. Available: <https://www.intel.com/content/www/us/en/products/programmable/fpga-vs-structured-asic.html>
- [65] O. Guzey, L.-C. Wang, J. R. Levitt, and H. Foster, “Increasing the efficiency of simulation-based functional verification through unsupervised support vector analysis,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 1, pp. 138–148, 2010.
- [66] S. Gogri, A. Tyagi, M. Quinn, and J. Hu, “Transaction level stimulus optimization in functional verification using machine learning predictors,” in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, 2022, pp. 71–76.
- [67] X. Inc., *ZCU104 Evaluation Board User Guide*, 2017, uG1267 (v1.0). [Online]. Available: https://www.xilinx.com/support/documentation/boards_and_kits/zcu104/ug1267-zcu104-eval-bd.pdf
- [68] Zynq UltraScale+ MPSoC Data Sheet: Overview. Accessed: 19.11.2024. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds891-zynq-ultrascale-plus-overview.pdf
- [69] Zynq UltraScale+ MPSoC Product Tables and Product Selection Guide. Accessed: 19.11.2024. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds891-zynq-ultrascale-plus-product-selection-guide.pdf
- [70] Xilinx ZCU104: Overview, User Guide, Features and Application. Accessed: 20.12.2024. [Online]. Available: <https://www.bitfoic.com/blog/xilinx-zcu104-overview-user-guide-features-and-application?id=164s>

- [71] What is ZCU102, and the Difference Between ZCU104 and ZCU102? Accessed: 19.12.2024. [Online]. Available: <https://www.fpgaakey.com/technology/details/what-is-zcu102-and-the-difference-between-zcu104-and-zcu102?srsltid=AfmBOooj5yLofJjfnfoWXpjkOIaHI5Bq1CshxizfeIKs-M-CV28jhXRx>
- [72] Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit. Accessed: 11.10.2024. [Online]. Available: <https://www.xilinx.com/products/boards-and-kits/zcu104.html>
- [73] ACoreChip. Accessed: 12.1.2024. [Online]. Available: https://gitlab.com/a-core/a-core_chisel/ACoreChip
- [74] ACoreBase. Accessed: 12.1.2024. [Online]. Available: https://gitlab.com/a-core/a-core_chisel/ACoreBase
- [75] M. Kosunen. TheSyDeKick- Complete Kit for System-on-Chip development. Accessed: 11.10.2024. [Online]. Available: <https://github.com/TheSystemDevelopmentKit>
- [76] Introduction to TheSyDeKick. Accessed: 19.12.2024. [Online]. Available: <https://thesystemdevelopmentkit.github.io/docs/introduction.html>
- [77] ETSI TS 123 501 V15.2.0. Accessed: 10.12.2024. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/15.02.00_60/ts_123501v150200p.pdf