INDIAN INSTITUE OF TECHNOLOGY ROPAR

Department of Electrical Engineering

Departmental Project Report

DNN based algorithm implementation on RFSoC

Submitted by:

Itte Revanth Kumar	2021EEB1178
Yash Vardhan Bhardwaj	2021EBB1226
Harshdeep Dey	2021EEB1286

ACKNOWLEDGEMENT

This report stands as a testament to the collaborative spirit that drove its completion, and we extend our deepest gratitude to those whose contributions were indispensable to this endeavour. Mr. Arhum Ahmad's unwavering support was a cornerstone throughout this project, as he provided invaluable guidance and assistance at every juncture, helping us surmount challenges and navigate complexities with ease.

We are also immensely grateful to Dr. Satyam Agarwal of the Department of Electrical Engineering for his pivotal role in shaping our involvement in this initiative. His insights and contributions significantly enriched our understanding and implementation of the BPSK demodulation technique using a deep neural network on the Zynq UltraScale RFSoC 4x2 board. Collaborating on this project has been a profound learning experience, offering insights into the intricacies of modern communication technologies. We extend our thanks to the Department of Electrical Engineering for their unwavering support, providing us with access to essential equipment, software tools, and infrastructure. Their resources played a crucial role in facilitating our research and experimentation, enabling us to explore new frontiers in signal processing and neural network applications.

In closing, we acknowledge the collective effort and commitment that made this project a success. Each contribution, whether in guidance, technical support, or resource provision, has been instrumental in shaping our journey and the outcomes presented in this report.

Introduction:

The evolution of communication systems has been characterized by a relentless pursuit of more advanced and adaptable methodologies aimed at processing signals with greater efficiency and precision. At the heart of this evolution lies demodulation, a critical aspect of communication that shoulders the responsibility of extracting valuable information from transmitted signals. While traditional demodulation techniques have proven effective in numerous scenarios, their efficacy diminishes when faced with the complexities of handling intricate modulation schemes and navigating through challenging channel conditions. In light of these inherent limitations, our project sets out to delve into the transformative potential offered by deep neural networks (DNNs) within the domain of demodulation, with a specific focus on Binary Phase Shift Keying (BPSK) modulation.

The implementation of this novel demodulation system is facilitated by the Zynq UltraScale RFSoC 4x2 board, a sophisticated platform that seamlessly integrates a processing system with programmable logic. This integration offers a conducive environment to harness the power of machine learning in demodulation tasks. The workflow of the project involves capturing signal data from an Analog-to-Digital Converter (ADC) and feeding it into a pre-trained DNN model deployed within the RFSoC 4x2 architecture. This DNN is specifically trained to decode the BPSK-modulated signal, extracting the underlying information accurately and efficiently.

Objective:

The core objective of this project is to design and implement a BPSK demodulation system using a DNN at the receiver end, thereby replacing conventional demodulation methods with a machine learning-based approach. This shift in methodology is motivated by the need for demodulation techniques that not only exhibit robustness but also possess the adaptability required to meet the dynamic demands of modern communication systems. By leveraging the capabilities of DNNs, we aim to develop a demodulation framework capable of efficiently handling diverse modulation formats while mitigating the effects of channel distortions and noise.

The implementation of this novel demodulation system is facilitated by the Zynq UltraScale RFSoC 4x2 board, a sophisticated platform that seamlessly integrates a processing system with programmable logic. This integration offers a conducive environment to harness the power of machine learning in demodulation tasks. The workflow of the project involves capturing signal data from an Analog-to-Digital Converter (ADC) and feeding it into a pre-trained DNN model deployed within the RFSoC 4x2 architecture. This DNN is specifically trained to decode the BPSK-modulated signal, extracting the underlying information accurately and efficiently.

Through rigorous experimentation, data analysis, and performance evaluation, we aim to elucidate the advantages and limitations of integrating DNNs into the demodulation process. The insights gained from this project are expected to contribute significantly to the advancement of demodulation techniques, paving the way for more intelligent and adaptive communication systems. By bridging the domains of traditional signal processing and cutting-edge machine learning, this project aims to spearhead the development of next-generation demodulation strategies, fostering innovation and advancement in wireless communication technologies.

Literature Review:

1. DeepDeMod:

In the past few years, numerous researchers have directed their attention towards utilizing neural networks (NN) within wireless communication contexts, including tasks like signal modulation recognition, optimization, demodulation, and more. For instance, [1] introduces *DeepDeMod*, a novel non-coherent binary phase shift keying (BPSK) demodulator based on deep neural networks (DNNs). The literature review focuses on non-coherent demodulation techniques, deep learning in signal processing, and challenges in demodulating signals under fading channels, noise, and hardware imperfections.

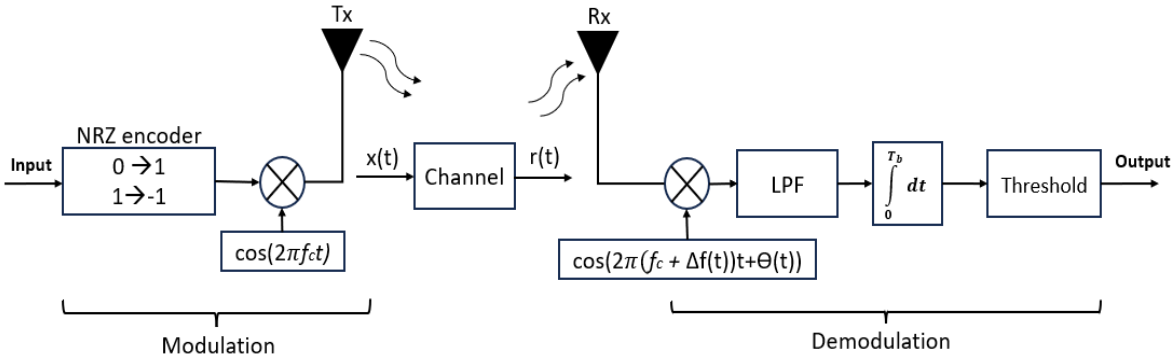


Fig.1 Conventional BPSK modulation and demodulation

As shown in Fig.1, the received signal undergoes several processing steps to extract the transmitted information. Initially, it is mixed with a reference carrier signal and then filtered through a low-pass filter (LPF) to eliminate high-frequency components and noise. The filtered signal is then integrated over a bit period (Tb) using an integrator, which accumulates the signal energy for

reliable detection. Finally, a threshold detector determines the transmitted bit by comparing the integrated signal with a predefined threshold value. Notably, when the transmitted bits are equally likely (equiprobable), the threshold value is set to zero to facilitate straightforward decision-making.

Achieving perfect synchronization of carrier frequency and phase at the receiver is essential for conventional coherent BPSK demodulation. However, in real-world scenarios, achieving absolute precision in frequency and phase synchronization is practically impossible. The proposed DeepDeMod scheme leverages DNNs to decode BPSK-modulated signals in the presence of fading channels, additive white Gaussian noise, and hardware imperfections like phase and frequency offset. This addresses the additional challenge posed by the time-varying nature of hardware imperfections and channel conditions, aiming to enhance demodulation performance under real-world scenarios.

As depicted in Fig.2, *DeepDeMod* differs from conventional demodulation using a low-pass filter (LPF), where the output is a constant message signal with noise dominance, making it challenging for the DNN to learn and map the message signal accurately, especially at low signal-to-noise ratio (SNR). In contrast, the BPF output includes both the message and carrier signal components, aiding the DNN in distinguishing the message signal amidst noise, enhancing learning accuracy. Additionally, sampling rate considerations are crucial, with a sampler or ADC operating at K/T_b to ensure precise signal conversion for DNN detection, capturing exactly K samples per bit period.

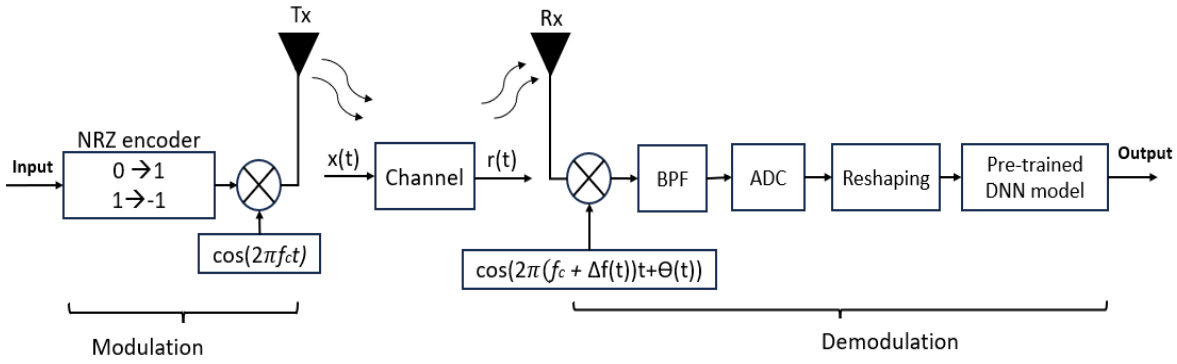


Fig.2 DeepDeMod demodulation receiver

The DNN functions as a detector, translating K samples per bit of the received data into a single bit. A reshaping block is employed to adjust the sampled signal to align effectively with the DNN's input requirements. Before detection, the DNN undergoes training using known signals to determine optimal weights, enabling accurate bit detection within the received signal.

2. Zynq Ultrascale + RFSoC 4X2:

The RFSoC 4x2 board is powered by the Zynq Ultrascale+ RFSoC XCZU48DR-1FFVG1517E, featuring a Quad-core ARM Cortex A53 Processing System (PS) and Xilinx Ultrascale+ Programmable Logic (PL). It boasts 8 RF ADCs and 8 DACs, with 4 RF ADCs running at 5 GSPS and 2 RF DACs at 9.85 GSPS. These capabilities are accessible through SMA connectors with integrated baluns, facilitating direct connectivity for antennas and external signal sources.

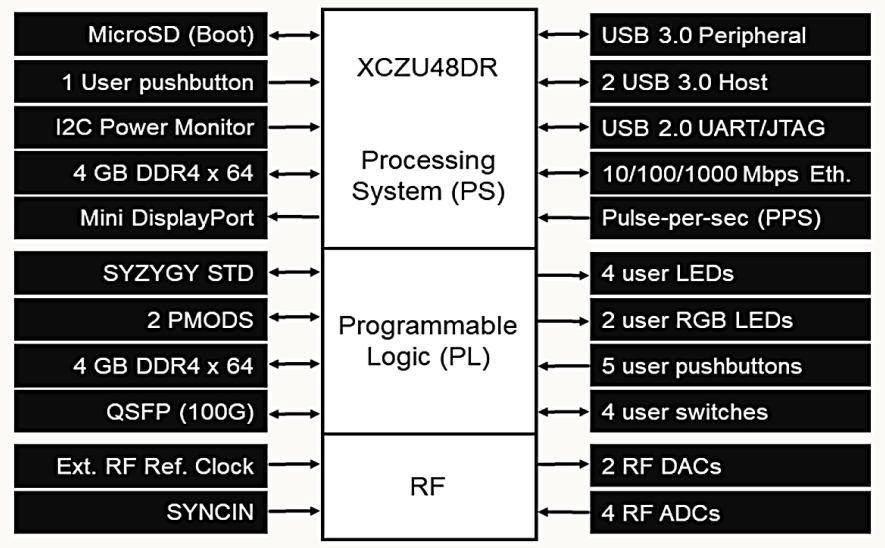


Fig.3 Block diagram for Zynq RFSoC 4x2

The RFSoC board is ideally suited to serve as a powerful and highly configurable software defined radio (SDR) system. The AMD-Xilinx ZYNQ UltraScale+ device includes a quadcore ARM Cortex-A53, a dual-core ARM Cortex R5F, monolithic direct RF sampling ADCs and DACs, and several other high-performance cores to assist with acquiring and processing high speed data.

3. Why is hardware setup better?

Conventional system-on-chip (SoC) technologies were tailored for specific applications like Bluetooth, Zigbee, Wi-Fi, and mobile phone chips (including GPS), focusing on compact transceiver designs. Over time, electronics have evolved with shrinking size and expanding functionality, exemplified by the latest advancement in transceiver technology—the RFSoC. RFSoC devices integrate RF-class multichannel analog-to-digital converters (ADCs) and

digital-to-analog converters (DACs) alongside Xilinx's multiprocessor system-on-chip (MPSoC) and an Arm processor-enhanced FPGA. The architecture, incorporates the mixed-signal interface (ADC/DAC) into the signal chain. These devices employ direct sampling ADC/DAC with digital down conversion (DDC) and digital up conversion (DUC). RFSoCs offer a notable advantage in terms of reduced power consumption. This reduction is primarily due to the consolidation of functions onto fewer boards, eliminating the need for interfaces that connect different ICs in a discrete setup.

One commonly used serial interface standard, JESD204B, has been phased out from Xilinx's RFSoC product line. Instead, RFSoCs integrate data converters directly into the FPGA using parallel interfaces, eliminating the high-pin-count external connections required for discrete parallel interface converters. Moreover, RFSoCs do not suffer from the latency associated with a JESD204 serial interface. This makes them highly appealing for applications requiring low power, multiple channels, and low latency. Additionally, the cost-effectiveness of SoC transceivers is evident in applications that demand limited functionality, as these functions are integrated onto a single chip. For such applications, RFSoC devices only require a few support circuits, including a microprocessor for control, a power supply, and antenna(s).

Proposed Neural Network:

The proposed deep neural network (DNN) detector is designed with a hierarchical structure comprising an input layer, four hidden layers, and an output layer. The primary learning and data processing occur within the hidden layers. Specifically, the first hidden layer with 40 neurons, followed by layers with 20, 7, and 5 neurons respectively, performs non-linear transformations through weighted connections and activation functions, leading to a single-neuron output layer for bit detection.

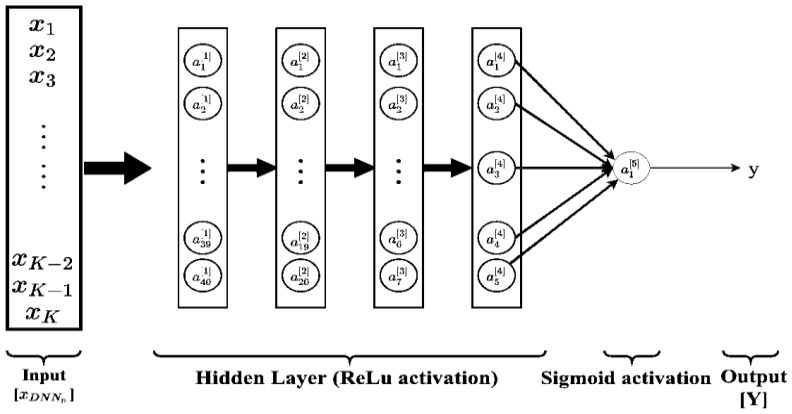


Fig.4 Proposed neural network

It operates on sampled signal inputs corresponding to a single bit duration, with the total number of samples in one bit period denoted as K, indicating K inputs for the DNN. The hidden layers in the proposed DNN serve as intricate layers of mathematical functions tailored to specific data transformations essential for detecting transmitted bits accurately. These layers are pivotal in assigning optimal weights to the inputs and applying non-linear operations that enhance the network's ability to discern patterns and features within the sampled signal inputs.

In essence, the proposed DNN detector encapsulates a sophisticated framework where the interplay between the input layer, hidden layers, and output layer orchestrates a process of data transformation and decision-making. Through forward propagation, the sampled signal inputs traverse the network, undergo intricate non-linear transformations in the hidden layers, and culminate in a single-neuron output layer that determines the transmitted bit, exemplifying the network's capability to extract meaningful information from raw signal data for reliable detection and classification tasks.

Parameter Name	Parameter value
Input Size (K)	10
Number of hidden layers	4
Number of neurons in each hidden layer	40, 20, 7, 5
Activation function in hidden layer	Rectified Linear Unit (ReLu)
Loss function	Binary Cross Entropy
Optimizer	Adam
Output Size	1
Activation function in output layer	Sigmoid

Fig.4 Proposed neural network parameters

Hardware & Software Setup:

1. PYNQ environment:

RFSoC-PYNQ is a robust framework tailored for RFSoC users, offering a range of tools and resources. At its core is the PYNQ framework, seamlessly integrated with Jupyter Lab to provide an accessible and interactive development environment. One of its standout features is the set of Python APIs dedicated to RFSoC's clock and data converters, empowering developers to control these critical elements directly through Python code. Alongside APIs, RFSoC-PYNQ supplies libraries and drivers that abstract complexities, making integration smoother. The framework includes a repository of example overlays and designs, serving as templates and guides for custom

application. Extensive tutorials and documentation further enhance the user experience, supporting users at all skill levels in harnessing RFSoC's potential efficiently.

The appropriate PYNQ image file needs to be downloaded from the PYNQ website which has to be written in the SD card. We need to be careful with selecting the correct drive while writing the PYNQ image. If we select the wrong drive, we might overwrite data on that drive. Once the image file is written in the SD card, we can access the PYNQ environment through the RFSoC 4x2 board.

2. Board Setup:

Following steps were performed to setup the board:

- Firstly, 4x2 RFSoC latest PYNQ image file is downloaded from the official website, and was flashed in SD card using Ubuntu Command line terminal.
- After that the card was inserted, and board was connected to the computer using micro–B USB 3.0 cable
- Set the boot mode switch to the **SD** position.
- Power supply was connected.
- PYNQ will initiate a boot sequence in the board.
- All 10 power status LEDs will initially turn on.
- Green DONE and INIT LEDs will illuminate after about 40 seconds, indicating successful boot progress. 4 white user LEDs (LED0-LED3) will briefly flash and then remain on.
- OLED will display network information and IP address, enabling connection to the board.

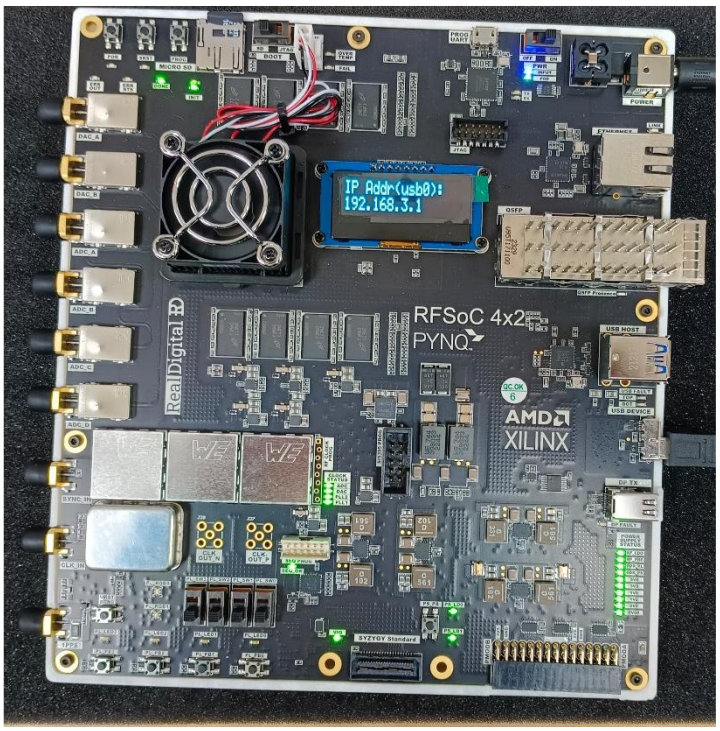


Fig5: Board after finishing the boot sequence

- Open Firefox (or any available browser) and navigate to http://192.168.3.1/lab.
- A login window will open, enter “xilinx” as the password.

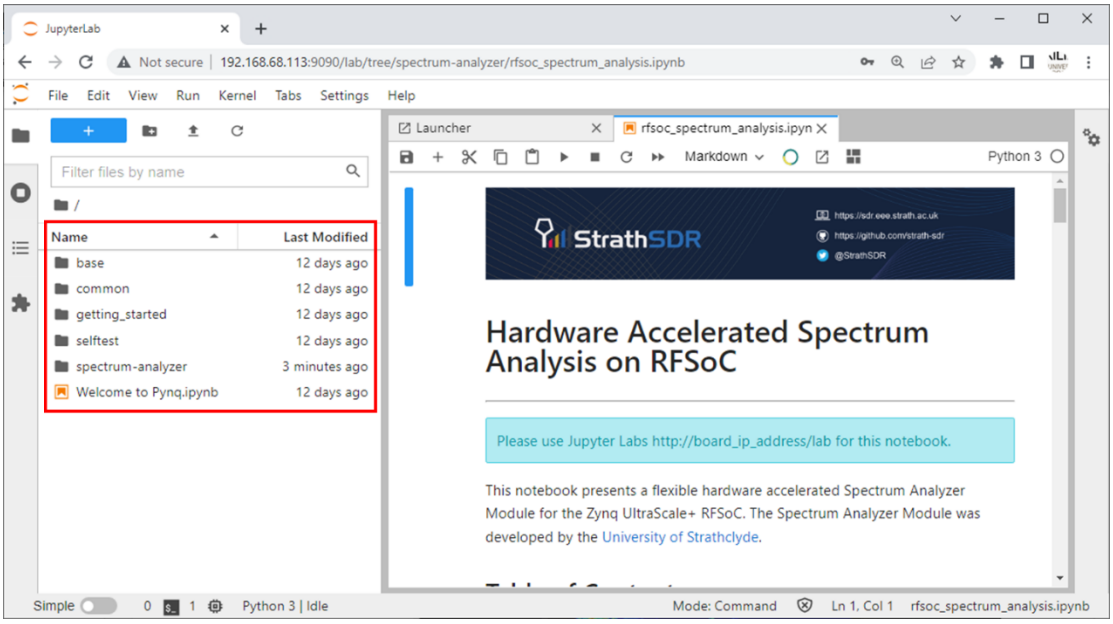


Fig6: PYNQ environment after logging in.

3. *Vivado Setup:*

- Download Vivado 2023.2.1 from the official website compatible with Ubuntu.
- Install Vivado on Ubuntu, ensuring compatibility with the operating system's version and architecture.
- Add the necessary paths to system environment variables like `$PATH` for Vivado's executables and `$XILINX_VIVADO` for Vivado's installation directory.
- Obtain the RF-SoC 4x2 board file and install it within the Vivado environment, allowing interaction with the programmable logic (PL) portion of the board.
- Vivado is ready to use now. Running the following command should open Vivado interface

```
$cd /opt/Xilinx
$source /opt/Xilinx/Vivado/2017.2/settings64.sh
$ vivado &
```

- Verify that the installed board file is shown in vivado UI.
- If its not visible changing the Jason file a bit would solve the problem

4. *System generator setup:*
    - Installed Matlab 2021b, the latest version compatible with System Generator.
    - Model Composer installed alongside Vivado.
    - Added Model Composer's path to Matlab environment.
    - Executed the "settings64.sh" shell script within Model Composer.
    - System Generator configured and ready.
    - To access, simply run "ModelComposer" in the terminal.

Setting up a radio communication between 2 antennas:

Overview

In the Radio Frequency System on Chip (RFSoC), the device architecture is divided into two primary customizable components: the Processing System (PS) and the Programmable Logic (PL). These distinct sections collaborate to facilitate complex signal processing tasks efficiently. In our implementation of a Binary Phase Shift Keying (BPSK) transceiver system, the digital signal processing (DSP) functionalities essential for BPSK modulation and demodulation are executed within the Programmable Logic (PL), while the control and coordination of these DSP elements are orchestrated by the Processing System (PS).

Within the Programmable Logic (PL), the BPSK transceiver's DSP components can be configured using hardware description languages (such as Verilog or VHDL) and exploiting the Vivado environment. In this project we have used preexisting IP blocks to configure the DSP elements of receiver and transmitter. These components encompass critical functions such as carrier generation, symbol modulation and demodulation, filtering, and error correction. Leveraging the FPGA fabric's parallel processing capabilities, we have implemented two hardware accelerators are provided; one to transmit data, and the other to receive data. Each accelerator is independent of one another and do not communicate. The transmitter modulates 100 kSa/s of data and it interpolates the signal to 1.024 GSa/s with subsequent interpolation stages. The RF DAC then transmits the data. The receiver is connected to the transmitter using an SMA loopback cable. The RF ADC will initially decimate the data and the receiver hardware accelerator will be responsible for synchronizing to the signal and extracting the modulated data

On the other hand, the Processing System (PS) serves as the brain of the RFSoC device, providing high-level control and management functionalities. Through the PS, we can configure, monitor, and adjust various parameters of the BPSK transceiver system, including modulation schemes, carrier frequencies, and signal bandwidths. The coordination and communication between the Processing System (PS) and the Programmable Logic (PL) are facilitated via the Advanced eXtensible Interface (AXI) and a Direct Memory access (DMA) controller to facilitate the transfer the data between Jupyter Labs environment and transmitter/receiver.

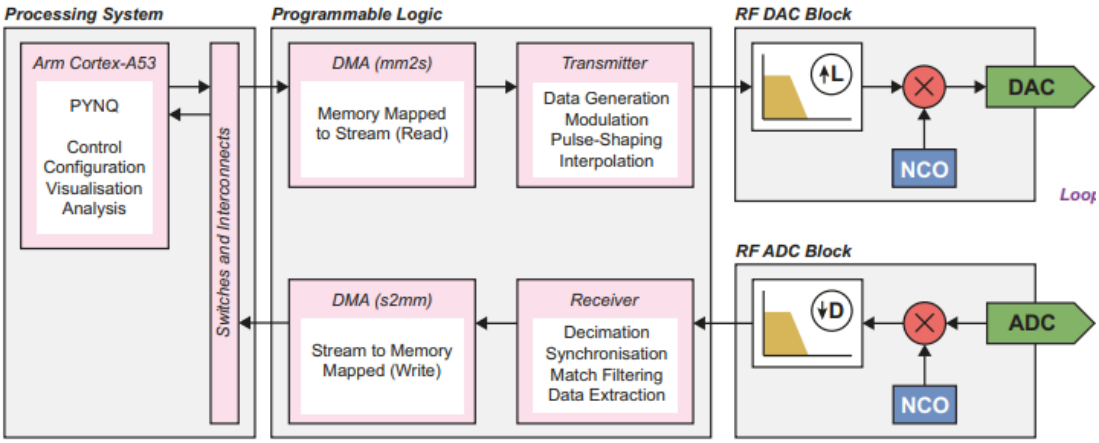


Fig7: Block diagram of BPSK trans receiver.

System architecture

Transmitter

At the transmitter side, the message data which was sent from PS to DMA is sent to transmitter block at a rate of 12.5 kSamples/s. This raw data then undergoes BPSK modulation, where it is encoded with information by altering the phase of the signal, operating at a rate of 100 kSamples/s. However, simply modulating the data isn't enough to ensure optimal transmission quality. This is where pulse shaping becomes crucial. The pulse shaping block refines the modulated signal by shaping its waveform, reducing spectral interference, and ensuring efficient use of the allocated bandwidth. By providing well-defined transitions between symbols, pulse shaping also aids in synchronization and timing recovery at the receiver, enhancing the system's overall performance.

Following pulse shaping, the signal enters the interpolation block, where its sample rate is increased to 400 kSamples/s. Interpolation plays a pivotal role in enhancing the signal fidelity and reducing quantization noise introduced during the digital-to-analog conversion process. By filling in the gaps between original samples, interpolation ensures that the transmitted signal retains its original shape and characteristics after up sampling. This higher sample rate allows for more precise control over the transmitted signal's characteristics, such as its bandwidth and transition properties, ultimately leading to improved performance and reliability in communication systems. Finally, the up sampled signal is passed to the DAC, which converts it into an analogy form at a rate of 128 MSamples/s for transmission over the channel.

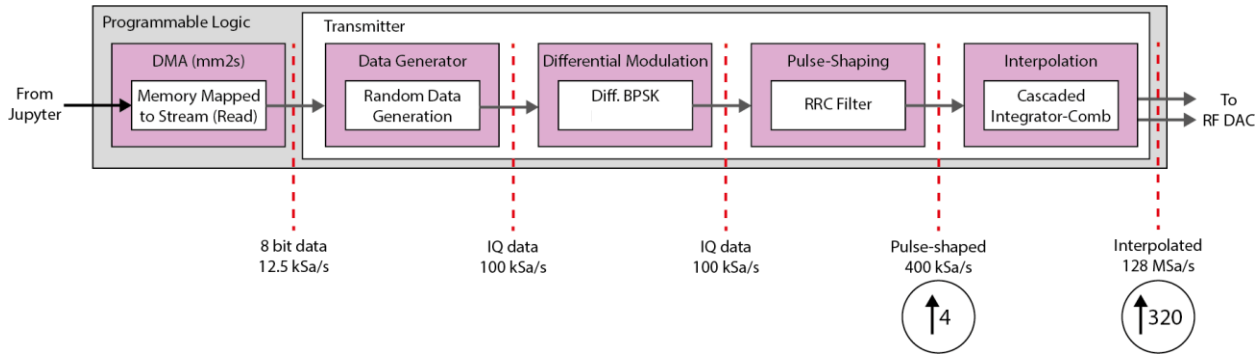


Fig8: System architecture of transmitter

Receiver

The receiver architecture framework comprises of five fundamental stages, each playing a critical role in the acquisition and processing of BPSK waveforms. Each stage is essential for the receiver's overall functionality and its ability to decode transmitted information accurately.

The first stage, decimation, serves as the initial processing step upon receiving the signal. Decimation involves reducing the sample rate of the incoming signal to 12.8 MSa/s. This reduction in sample rate not only helps optimize subsequent processing but also conserves computational resources, making the signal more manageable for further analysis.

Coarse frequency synchronization follows decimation, addressing any frequency offsets present in the received signal. This stage corrects frequency offsets up to 1.6 MHz, with offsets larger than 195.3125 Hz being eligible for correction by the coarse frequency synchronizer. This synchronization process is crucial for aligning the received signal with the receiver's frequency reference, ensuring accuracy in subsequent processing stages.

Once frequency synchronization is achieved, the signal undergoes matched filtering. This stage involves applying a matched root raised cosine filter to the signal, effectively suppressing inter-symbol interference. By enhancing the signal-to-noise ratio and minimizing interference, matched filtering improves the accuracy of demodulation and subsequent data processing.

Following matched filtering, time and phase synchronization become paramount. This stage identifies the maximum effect points within the signal, aiding in the determination of binary values (1s or 0s). Time and phase synchronization are essential for accurately demodulating the signal, ensuring that the receiver's timing and phase are synchronized with the incoming waveform, thereby facilitating precise data recovery.

The final stage in the receiver architecture is frame synchronization. Here, an extended barker sequence is correlated with the received binary data to detect the start of a frame. Frame synchronization is critical for accurate data framing and synchronization, enabling the receiver to

interpret and decode transmitted information correctly. Once frames are synchronized, demodulated data is transferred into system memory by the DMA controller for further analysis and processing within the receiver system.

In summary, each stage within the receiver architecture contributes uniquely to the overall process of signal acquisition and processing. From decimation to frame synchronization, these stages work together seamlessly to ensure the accurate decoding of BPSK.

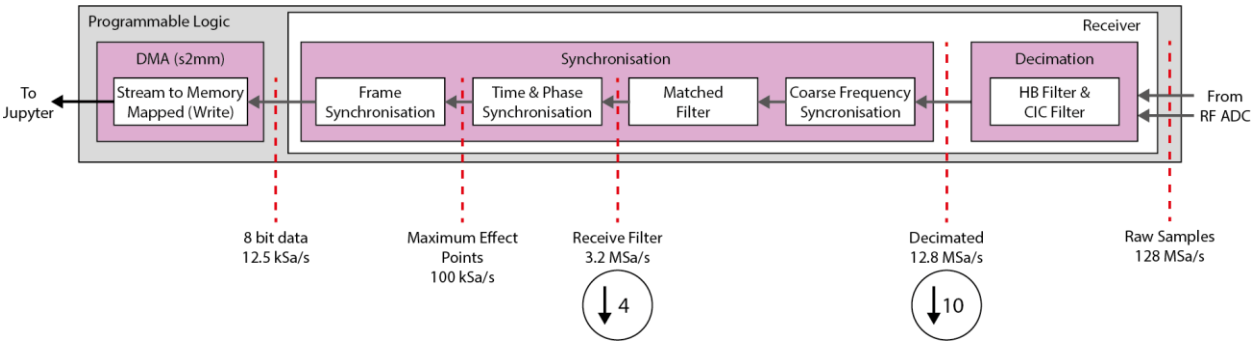


Fig9: System architecture of receiver

Hardware setup

After connecting the board with pynq environment, connect an antenna with ADC A and another with DAC A. We will be communicating through these antennas.

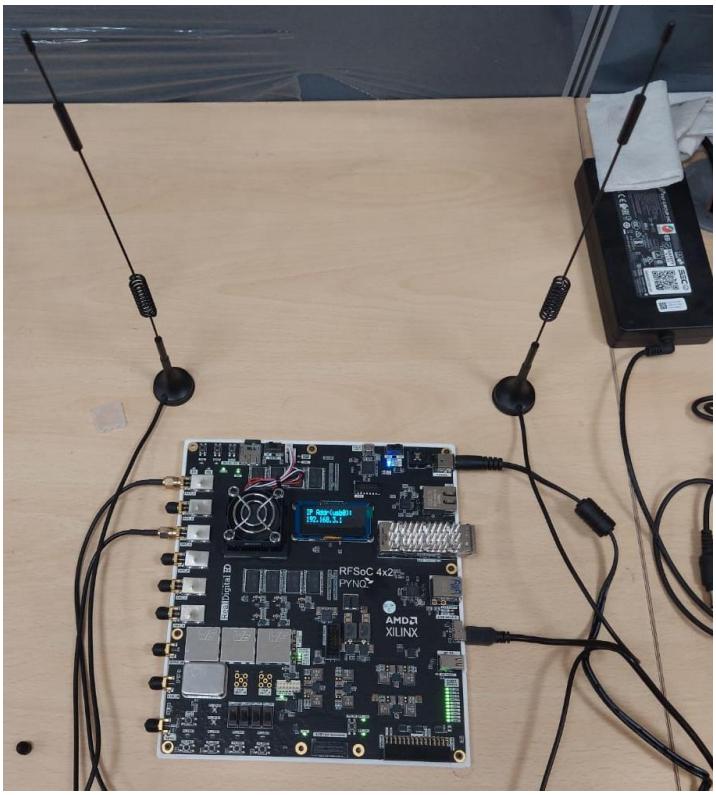


Fig10: Hardware setup for BPSK trans receiver

Software setup

Our initial task involved configuring the FPGA fabric for digital signal processing. We achieved this by installing "*rfsoc_radio*," a repository housing various bitstreams and IP blocks necessary for designing the BPSK transceiver. Following installation, we activated the overlay to download the bitstream into the FPGA fabric which enabled us to control over the parameters of the IP blocks through specific drivers. Overlays, or hardware libraries: which are wrapped with PYNQ Python API, are programmable/configurable FPGA designs that extend the user application from the Processing System of the Zynq into the Programmable Logic. We can then use the Python interface to program and control specialized hardware overlays without needing to design an overlay themselves. This trans receiver has two primary software objects in the overlay design. The first is "*ol.radio_transmitter"*, which is the software wrapper responsible for controlling the BPSK transmitter and corresponding Direct Memory Access (DMA) IP cores. Similarly, the second is "*ol.radio_receiver"*, which is another software wrapper that controls the BPSK receiver and associated DMA.

Two crucial RFSoC Python libraries used in this radio design are imported, namely "xrfclk" and "xrfdc". The xrfclk module is responsible for configuring the RFSoC's LMK and LMX clock devices. These devices are used to derive the RF data converter's sampling clock for RFSoC platform. RFSoC LMK device default frequency is 245.76MHz. The xrfdc module is a Python binding used to configure RF DCs from PYNQ environment. We can access the ADC and DAC tiles of RFSoC and change its sampling frequency according to our need.

After configuring the FPGA fabric, we ran default synchronization tests to ensure our system is ready for communication. Once synchronization is achieved, we activate an interactive interface to facilitate easy system control and received data visualization. To do so we initiate the "terminal" instances of "ol.radio_receiver," for viewing received message and "*ol.dashboad"* to control system parameters.

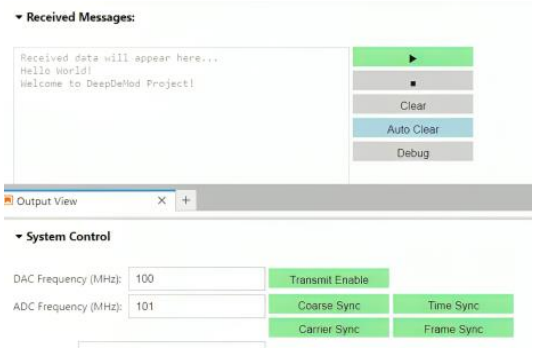


Fig11: Interactive terminals

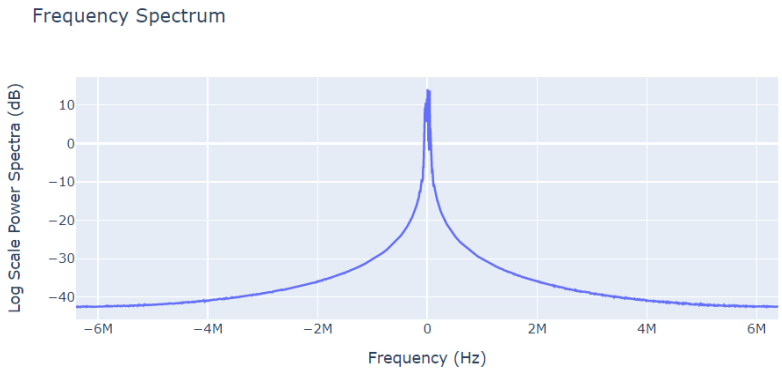
After completing the activation of interactive interface, we tested our communication system with a test message. It was successfully received by the receiver (shown in Fig11).

Results

Once the communication system setup is complete, we proceed to tap into the data at each stage of both the transmitter and receiver. To facilitate this, a data inspector was incorporated into the transmitter hardware system. This inspector enables the transfer of data frames from the programmable logic (PL) to external memory. Subsequently, interactive interface was developed to manipulate the data samples for visualization purposes using the Python Plotly library.

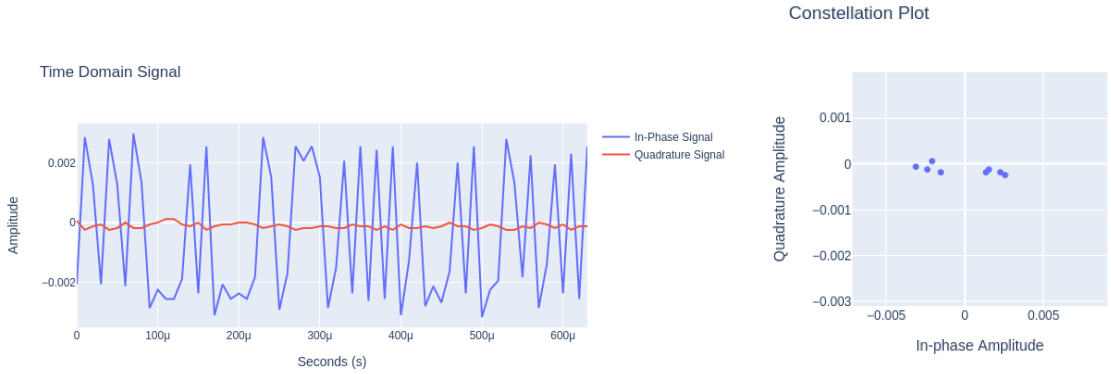
Receiver

- *Coarse frequency synchronisation*
  The coarse frequency synchronization stage is essential as it centers the received signal over 0Hz. This procedure is necessary to extract the data payload. This stage corrects offsets up to 1.6MHz. If the signal's offset exceeds 1.6MHz, the reported frequency offset will be inaccurate, hindering proper alignment.

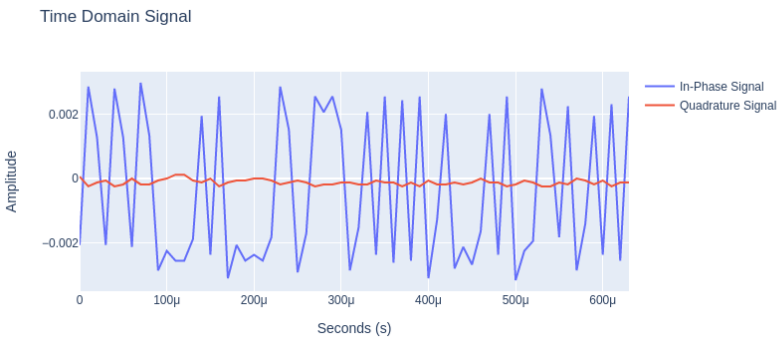


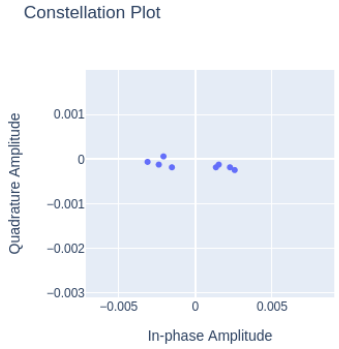
- *Time synchronization*

  In the constellation plot, of BPSK signal can be seen rotating in a circular motion. This rotation occurs because the real and imaginary components of the signal are changing in amplitude over time. The amplitude is changing because the signal is not carrier synchronized (or phase aligned correctly).



- *Phase synchronization*
  The problem of rotating constellation is solved at this stage. This stage allows us to rotate the received constellation to appropriately adjust the amplitude of the in-phase and quadrature components.

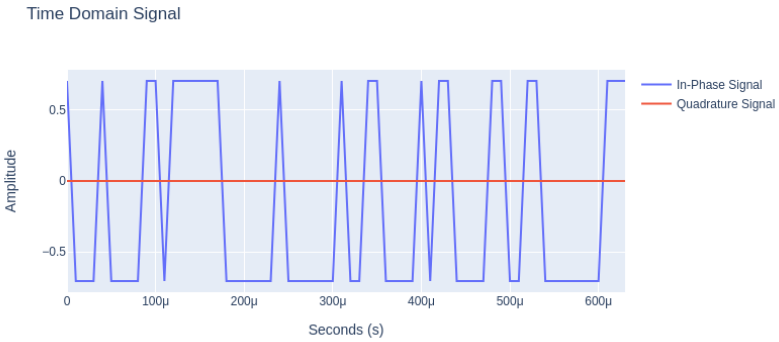


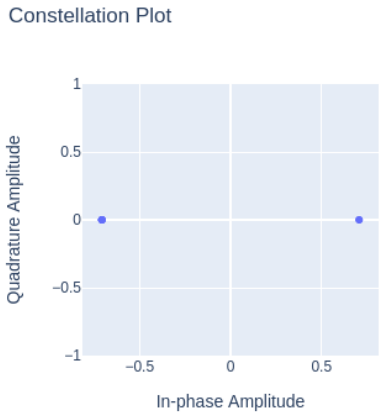


After phase synchronization a bit stream is generated that is send to PS. There the bit stream is converted into 8bits integers and then those integers are mapped to ascii symbols.

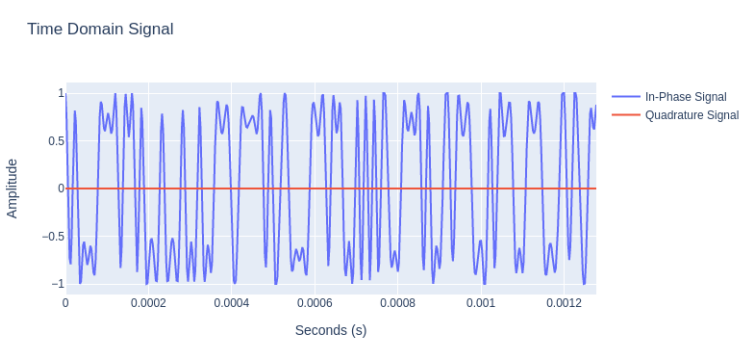
Transmitter

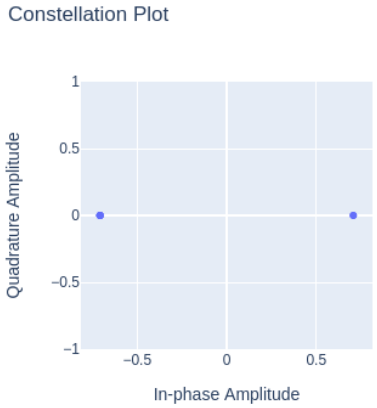
- *Symbol Generator*
  The ascii symbol number corresponding to our message signal is first calculated in PS, and then it is transmitted to PL. There it is converted to binary bitstream which is then modulated using BPSK at this stage

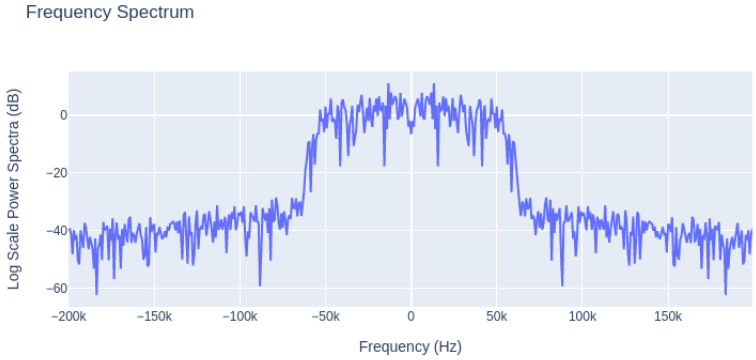




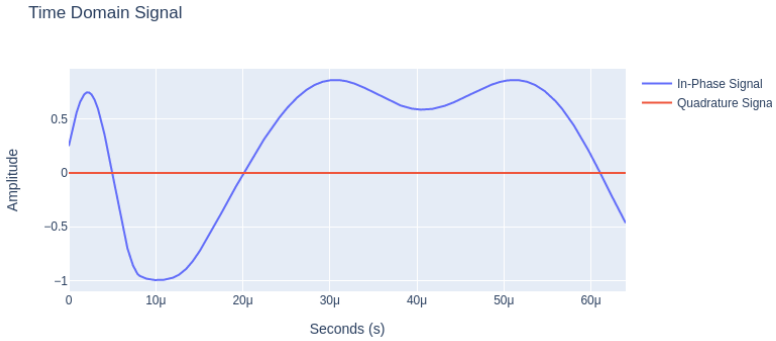
- *RRC transmit filter*
  Our radio system operates at 100kSa/s, which means we should be able to see a band that is 100kHz wide in the spectrum plot. When we select BPSK modulation, you will be able to see a double-sided spectrum that is symmetrical over 0Hz

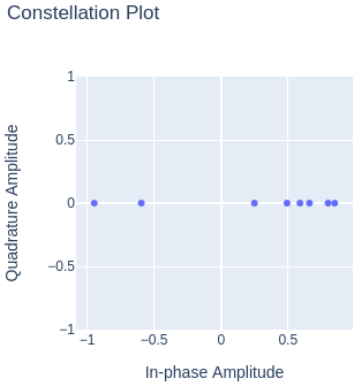






- *CIC interpolator*
  Interpolation increases the sample frequency of our waveform to meet the sampling requirements of the RF DAC. The CIC interpolator increases the sample frequency by a factor of 320.





Conclusion

In our project, we embarked on a comprehensive exploration of digital communication, focusing on the deployment of a Binary Phase Shift Keying (BPSK) transceiver in the RFSoC 4x2 environment. Throughout this endeavor, we gained profound insights into the intricacies of both the Programmable Logic (PL) and Processing System (PS) components of the FPGA. Our journey involved delving into various software tools, notably Vivado and System Generator, harnessing their capabilities to design and implement complex communication systems. Additionally, we leveraged open-source resources such as the Finn compiler, expanding our toolkit and enhancing our development process.

A significant aspect of our project involved delving into advanced concepts such as Deep Neural Networks (DNN) and Transfer Learning (TL) and their integration within the communication domain. This exploration illuminated the potential for incorporating machine learning techniques to enhance communication systems, paving the way for innovative solutions and improved performance. By synthesizing our knowledge of digital communication principles, FPGA architecture, and software tools, we successfully developed and deployed the BPSK transceiver. This hands-on experience provided invaluable insights into the practical aspects of communication system design and implementation.

As we conclude our project, we emerge with a deeper understanding of digital communication paradigms, reinforced by practical experimentation and exploration.

Future work

- Integrating the DeepDeMod neural network into the PL architecture for enhanced signal processing.
- Exploring the implementation of transfer learning techniques to adapt and optimize the neural network's performance.
- Applying transfer learning methodologies to improve MIMO-based communication models, potentially enhancing system efficiency and reliability.

REFERENCES

[1] *"DeepDeMod: BPSK Demodulation Using Deep Learning Over Software- Defined Radio*", Arhum Ahmad, Graduate Student Member, IEEE, Satyam Agarwal, Senior Member, IEEE, Sam Darshi, Senior Member, IEEE, and Sumit Chakravarty, Member, IEEE.

[2] Github link: https://github.com/strath-sdr