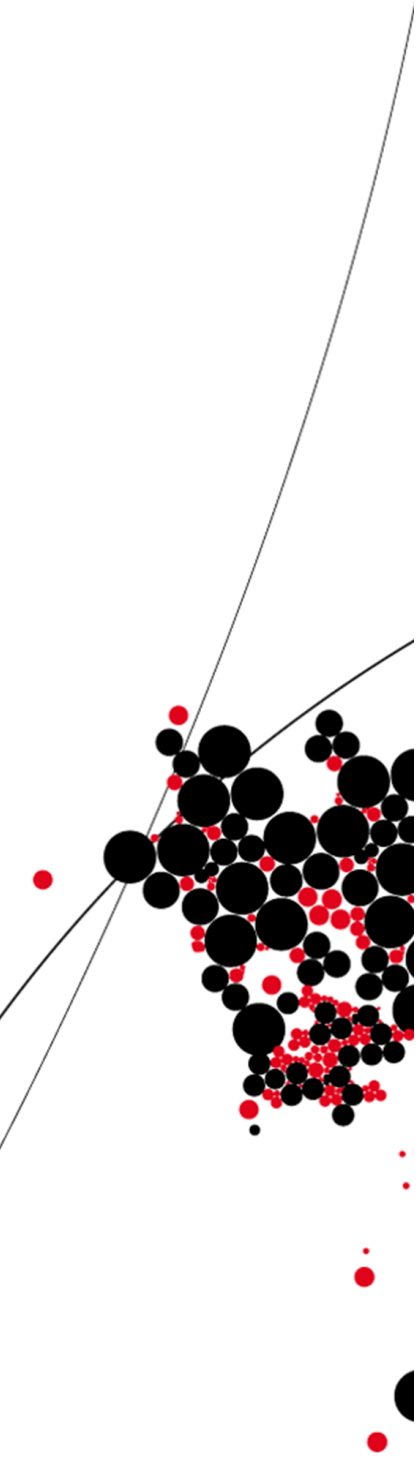
UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

SDR based module for Low Power WAN nodes development

Muhammad Rizwan

M.Sc. Thesis
November 2020



Supervisors:
DR.IR. A.B.J. KOKKELER (ANDRE)
DR. A. ALAYON GLAZUNOV (ANDRÉS)
DR.IR. R.A.R. VAN DER ZEE (RONAN)
Daily Supervisor:
ZAHER MAHFOUZ MSc

Radio Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

In the present digital and high-speed wireless era, there is an increased need to meet the data services demand while maintaining the power consumption to a minimum. It is always a compromise between critical requirements with the lesser critical ones.

Wireless Sensor Networks (WSN) have revolutionized the common man's life with their enhanced usage in a multitude of applications ranging from personal space to industrial monitoring. The Internet of Things (IoT) is a sub-domain of the WSN which has been used in several applications such as health monitoring, temperature monitoring, humidity monitoring, pressure monitoring, and calculating electricity consumption in smart meters. In an application, a sensor node is present where data rates and energy requirements are low except for smart power meters. Smart power meter sensors have the main power supply as a primary energy source, so power consumption is not an issue there. Due to the appealing features of sensor nodes, they are being deployed almost in every field.

Although sensor nodes are an attractive choice due to their appealing features, they can pose some additional challenges. One of the key challenges is their accessibility in case of an update of the technology, a protocol, or a new physical layer installation. Since a network consists of hundreds of sensor nodes that are spread over a geographical area, the only plausible option in before mentioned circumstances is the redeployment of nodes. In addition, most of the nodes are placed in a hardly accessible area; replacement is not desirable due to logistics and installing issues.

In our work, we focus on the development of a reconfigurable Low Power Wide Area Network (LPWAN) nodes, to circumvent the challenges and make the upgrading process cost-effective and simple. We investigate several software-defined radio (SDR) choices, and present a hybrid solution for final implementation. The hybrid solution consists of a Field Programmable Gate Array (FPGA), a microcontroller, and a radio frequency (RF) transceiver chip.

A point to point transceiver is designed and tested at a data rate of 100 bps using quadrature phase shift keying (QPSK) modulation. A sensor node works on low data rates, which require narrowband filters for improving signal-to-noise ratio (SNR) and sample rate conversion. Two types of low pass filters are simulated for sample

rate conversion and efficient hardware resources realization. The Finite Impulse Response (FIR) filters are simulated in the initial design due to the simplicity and ease of implementation on hardware. The FIR filters are easy to implement, but they have high hardware resource requirements due to extensive multipliers usage. The cascaded integrator-comb (CIC) low pass filters are efficient in such applications due to their multiplier-less structure, but the passband droop limits their advantages. Moreover, hardware resources are fewer in a sensor node for minimizing power consumption on a coin cell battery. The proof of concept is verified and analyzed through simulations in the Simulink. It is deduced that the CIC filters are an optimum solution for resource constraint design, and cascading an FIR filter at the lower sampling frequency side can correct the passband droop .

A successful transmission and reception of a 200-bit packet verify the proof of concept in Hardware Description Language (HDL) based simulations. The MATLAB HDL coder shortens the development time by directly generating the HDL codes for the system design. The block diagram developed in the Simulink can serve as a basic structure on which new physical layers can be implemented and verified for the sensor node.

Contents

List of acronyms

ADC	analog to digital converter
BER	bit error rate
BLE	bluetooth low energy
CLBs	configurable logic blocks
DAC	digital to analog converter
DSP	digital signal processor
FIR	finite impulse response
FPGA	field programmable gate array
GPP	general purpose processor
HDL	hardware description language
IF	intermediate frequency
IoT	internet of things
LNA	low noise amplifier
LTE	long-term evolution
LPWAN	low power wide area network
LVDS	low voltage differential signal
MOSI	master out slave in
MISO	master in slave out
MSPS	mega samples per second
QPSK	quadrature phase shift keying

PHY	physical layer
PSoC	programmable system on chip
RF	radio frequency
RFSoC	radio frequency system on chip
SCLK	serial clock
SDR	software defined radio
SELN	select active low
SNR	signal-to-noise ratio
SPI	serial peripheral interface
SoC	system on chip
SRAM	static random access memory
Wi-Fi	wireless fidelity
WSN	wireless sensor network
UNB	ultra narrow band

Chapter 1

Introduction

Advancement in wireless technology is paving grounds for new communication devices. More and more devices are being deployed into the already crowded electromagnetic spectrum. Internet-of-Things (IoT) is one of the emerging fields in the current era, which was envisioned in 1990. IoT includes a multitude of wireless devices such as wireless sensor networks (WSN), Near Field communication, machine-to-machine communications, Body Area Networks (BAN) and already mature personal area networks such as Wi-Fi, Bluetooth, cellular, etc [1]. The vision behind the WSN is the collection and monitoring of the environmental variables (temperature, sound, vibration, pressure, and motion, etc) [2]. Wireless sensor nodes have affected the common-man lifestyle due to the ease of accessibility and the new concept of smart homes. WSN are mostly deployed in places that are not easily accessible for human intervention and battery replacement. Some of the applications of WSN can be seen in Figure 1.1.

Figure 1.1 only gives a glimpse of the vast number of applications of wireless

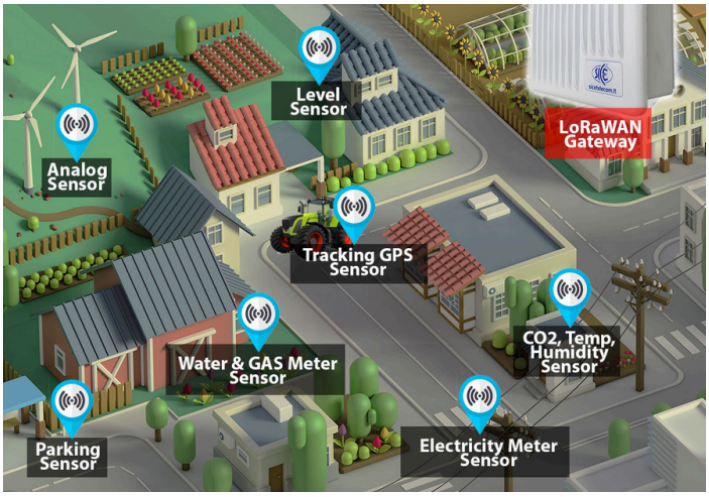


Figure 1.1: Application of wireless sensor nodes WSN [2]

sensor networks. As per the studies of [3], by the end of 2020, the world will have twenty-five billion sensor nodes. Ownership of 6-7 nodes per person can easily elaborate the humongous problem of spectrum usage by billion of nodes. Communication in such an environment is a challenge for communication engineers, who have to ensure reliable communications among end nodes but not at the cost of high power consumption. High power consumption is a threat to the life of a coin cell operated sensor node. One example of a gigantic network that contains thousands of nodes is the low power wide area network (LPWAN). Large area transmissions and remote terminals follow the Shannon-Hartley channel capacity theorem as follows:

$$C = B \log_2(1 + SNR) \tag{1.1}$$

C is the capacity of the medium in bits per second, B is the bandwidth of the signal transmitted in hertz (Hz) and SNR is the signal-to-noise ratio. From 1.1, it can be deduced that if the bandwidth is lowered then SNR must be increased for maintaining the same channel capacity and BER. Low data rates and larger coverage area resulted in the emergence of LPWAN. It has gained importance over the competing radio technologies (Zigbee, bluetooth low energy (BLE) etc) due to the competing technologies having shorter coverage range and higher device cost. The cellular networks can be a favorable option for LPWAN based applications due to the already deployed vast network of gateway nodes. High device power consumption and expensive frequency spectrum had hindered their scalability in a long-range, low data rate and low power consuming option for LPWAN.

LPWAN nodes are designed on a single-hop communication basis in which nodes are connected in a star topology (Figure 1.2). In star topology, every user has to transmit/receive data to/from the gateway nodes. The gateway nodes are interlinked over a fixed communication medium (e.g. the internet) to form an interconnected information cloud. Interconnectivity among the gateway nodes and the end nodes

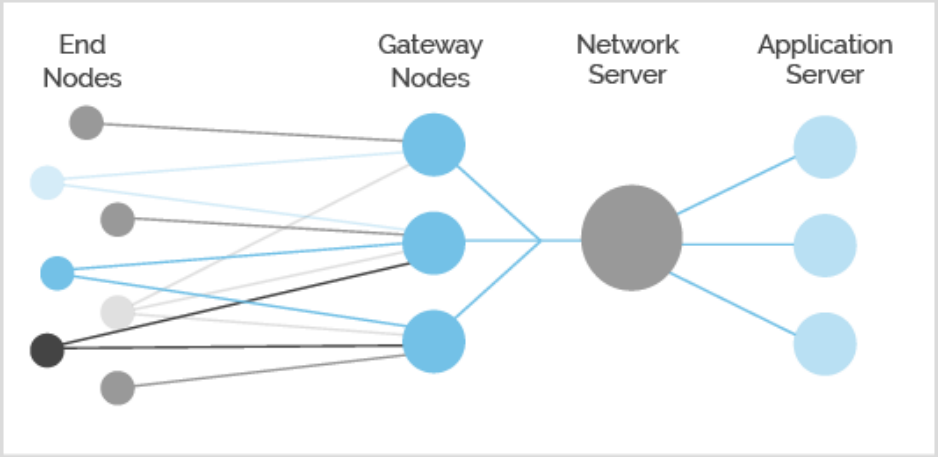


Figure 1.2: LPWAN topology [4]

is a key feature in transforming normal factories into the smart factories [5]. The challenge for a base station is to serve thousands of end nodes without requesting repeated transmissions and dropped packets. Keeping in mind the severity of the problem many protocols, standards, and technologies have evolved. Among the available technologies, three of them are the main competitors and are highly adopted by different countries (Sigfox, LoRa, NB-IoT).

1.1 Available LPWAN standards

1.1.1 Sigfox

Sigfox is a patented technology of a French company founded in 2010. It provides end-to-end IoT connectivity to nodes based upon a proprietary protocol [6]. Sigfox deploys its own base stations which are equipped with an IP-based backbone. It can support two transmission data rates of 100 bps and 600 bps depending upon the region of deployment [7]. A Sigfox gateway restricts 140 uplink messages per day with a maximum of 12 bytes payload [7]. The communication link between a node and a gateway is asymmetric which allows a maximum of 4 downlink messages of 8 bytes per day. Asymmetry of the link requires highly reliable uplink communication. That's why it supports both frequency and time diversity. Uplink messages are transmitted multiple times (three times by default) to ensure successful information reception at the gateway. The sub-1GHz band (Europe 868.180MHz-868.220MHz) is divided into 400 orthogonal channels with 100 Hz bandwidth (40 channels are reserved) [6]. Sigfox utilizes frequency and time diversity (Figure 1.3) by transmitting messages multiple times at different frequencies. Spatial diversity is achieved by the reception of the same message by multiple of neighbouring gateways (three by default).

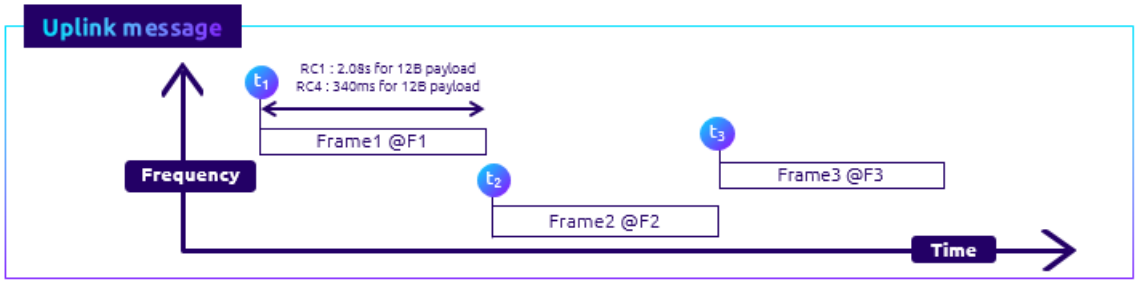


Figure 1.3: LPWAN frequency and time diversity plot [8]

1.1.2 LoRa

Lora is developed by Semtech for long-range, low power and low data rate applications [9]. It operates in the same unlicensed ISM sub-1GHz band as Sigfox. It uses a proprietary spread spectrum technique for modulation with a maximum data rate of 50 kbps. It supports bidirectional communication but most of the traffic is generated by the end nodes. It has an adaptive data rate feature that changes data rate depending upon the communication link status and spreading factor used from the 6 available choices. Data transmitted by the end device is received simultaneously by the neighboring gateways as shown in Figure 1.4. The redundant reception helps in successful data transfer but the network server is intelligent in discarding the redundant packets based on the time difference of arrival (TDOA).
The LoraWAN supports three classes of end devices namely A, B, and C. Class differentiation depends on the reception mechanism for a bidirectional data link. A class-A end device supports bi-directional communication whereby the uplink transmission period is followed by two receive windows. The transmission period depends on the application's data rate requirements. Class-A end device consumes less power among the three device classes. These devices are preferred when short downlink transmission messages are required after uplink messages. A Class-B device in conjunction with a random receive window opens a scheduled receive slot. Time-synchronized beacons are transmitted by the base station which helps the network server to know when the device is in listening mode. A Class-C device is always in listening mode except when they are transmitting. The next version of the LoraWAN is under development that will support roaming and temporary switching between device class from A to C [6].

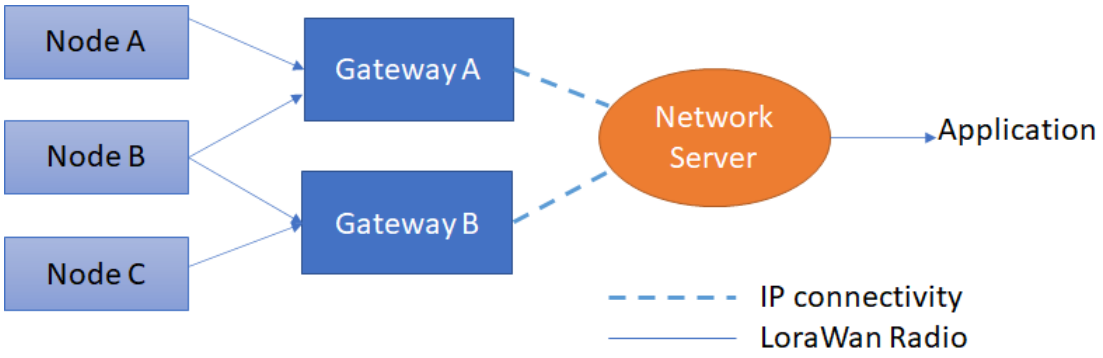


Figure 1.4: LoraWAN network architecture

1.1.3 Narrowband IoT (NB-IoT)

The NB-IoT is a narrowband IoT technology, introduced by the 3GPP group in June 2016. Unlike its competitors, NB-IoT shares a licensed frequency band with a global system for mobile (GSM) and long-term evolution (LTE). NB-IoT has a bandwidth of 200 kHz which is equal to one resource block in GSM and LTE transmission [10]. NB-IoT is regarded as a new air interface but it is being developed on already present LTE infrastructure. NB-IoT is scalable up to 100k end devices per cell with the capability of adding more devices by allocating more carriers to the NB-IoT frequency bank. NB-IoT employs QPSK modulation with a maximum payload size of 1600 bytes. The data rate can varry up to 200 kbps in downlink and 20 kbps in uplink [10] NB-IoT uses single carrier frequency division multiple access (FDMA) for uplink and orthogonal frequency division multiple access (OFDMA) in the downlink. NB-IoT future improvements are suggested by the 3GPP group in their 15^{th} release, one of them is the mobility for the upcoming NB-IoT devices.
Depending on the application requirements and data communication factors, one of the IoT technologies is selected from the available LPWAN solutions. Requirements can be low cost, long battery life, quality of service (QoS), coverage range or ease of scalability, etc.

The differences between the three technologies are summarized in Table 1.1. From the table, it can be concluded that the LoRa and Sigfox can be preferred over NB-IoT based on spectrum cost. However, if low latency and Quality-of-Service (QoS) are required then NB-IoT is the only choice among the three of them due to the licensed spectrum. If scalability is the main goal then NB-IoT will be the best due to already developed LTE and GSM infrastructure.

1.2 Motivation

Recently we have seen much development in the LPWAN field but mostly it is software oriented. Fewer research is conducted in the node hardware reconfigurability area. The available LPWAN sensor nodes are ASIC-based which means their hardware interconnections can not be changed after final product deployment. LPWAN is a developing technology, which sees breakthroughs every year. To make already deployed infrastructure compatible with a new technology or protocol is nearly impossible. Compatibility requires new node deployment in the same coverage area. The choice of redeployment is not favorable because sometimes nodes are deployed in hardly accessible areas. The redeployment of nodes requires time and labor that companies can not afford in the race of capturing the market first.

Already available SDR platforms are bulky and expensive, which makes them an unfavorable choice for being used as LPWAN sensor nodes. Moreover, some SDR platforms also require an external processor for processing data.
A LPWAN is intended for low power, low data rates, and long-range communication among interconnected sensor nodes. These are the distinguishing features compared to the conventional wireless networks that require high operating power and connect multiple users or businesses for high data rates. The LPWAN supports up to 50 kbits per channel which suffice sensor node communication needs [11]. This study focuses on the feasibility and software verification of reconfigurable LPWAN nodes. The main focus of this work will be on developing a standalone reconfigurable node that is suitable for LPWAN communication. The reconfigurability of LPWAN nodes will help companies to get their infrastructure deployed first and PHY/MAC layers may be upgraded or reconfigure later without the need for redeployment. These nodes will be economical relative to the high-end processing SDR platforms prices and comparable to the available ASIC-based LPWAN nodes available in the market.

Table 1.1: Sigfox, LoRa and NB-IoT features overview

	LoRa	Sigfox	NB-IoT
Frequency	Unlicensed ISM band (868 MHz in Europe)	Unlicensed ISM band (868 MHz in Europe)	Licensed LTE frequency band
Modulation	CSS	BPSK	QPSK
Bandwidth	250 kHz and 125 kHz	100 Hz	200 kHz
Transmission constraint	Unlimited	140 uplink messages and 4 downlink messages per day	unlimited
Maximum Payload	243 bytes	12 bytes uplink and 8 bytes downlink	1600 bytes
Adaptive rate	Yes	No	No
Latency	variable	variable	fixed
QOS	No	No	Yes
Scalability	Requires standard specific base station	Requires standard specific base station	compatible with GSM or LTE network
Standardization	LoRa Alliance	SigFox company	3GPP

1.3 Work of others

The LPWAN allows new applications and devices development but due to several constraints, it requires nonconventional protocol design [2]. Having an adequate amount of resources on board while minimizing the power consumption of the device is a challenge for a designer. Several constraints motivate huge research in the standardization process and attract industrial investments in this field [2]. In [12], the author discusses the equal importance of the transmission protocol and the processing algorithm on basis of energy consumption. RF transmission consumes high power in the conversion of a signal from the digital domain to the analog space. Signal processing software/protocol energy requirements can not be left unattended. It can happen that signal processing may take a considerable time which can make energy consumption in comparable to the transmission of a signal [12]. In [13] the authors developed a delay aware algorithm which controls the total number of active nodes while keeping optimum connectivity of the network. The algorithm will help in keeping information delay to sink under the required value and meanwhile reducing the energy consumption by the network. Most of the research work is carried out in the development of a power-saving algorithm and cross-layer protocol design for sensor nodes. Less work is carried out in the domain of reconfigurable sensor nodes.
Openchirp is a management framework for LPWAN that gives access to the user over the web [14]. The Openchirp currently supports loRaWAN (LoRa LPWAN protocol) with future support for Bluetooth, IEEE 802.15.4, and other IoT communication protocols is envisioned. The authors developed the hardware (name as LPRAN) shown in Figure 1.5. The LPRAN board consist of an FPGA, an RF chip, and a low noise amplifier (LNA). The RF chip provides an I and Q raw signal data as a sigma-delta modulated streams in reception. An onboard FPGA and external microcontroller Raspberry pi 3 do the rest signal processing tasks. The LPRAN board is developed for gateway nodes thats why it has more hardware resources on-board than actually required for end node. The LPRAN board currently supports LoRaWan based reception only. The LPRAN board doesn't fit the thesis scope owing to multiple reasons: high hardware cost, the receiver only functionality, and not a compact solution for the end node device.

In [15] the authors demonstrated a reconfigurable LPWAN solution on the Cypress programmable system on chip (PSoC) technology. Unlike openchirp, the PSoC solution is a proof of concept for the reconfigurable LPWAN nodes with an external protocol specific (LoRa) RF IC. This hardware has a limitation of working on LoRa protocol only. The discussed PSoC hardware is not suitable for reconfigurable node implementation due to the limited application on one LPWAN protocol.

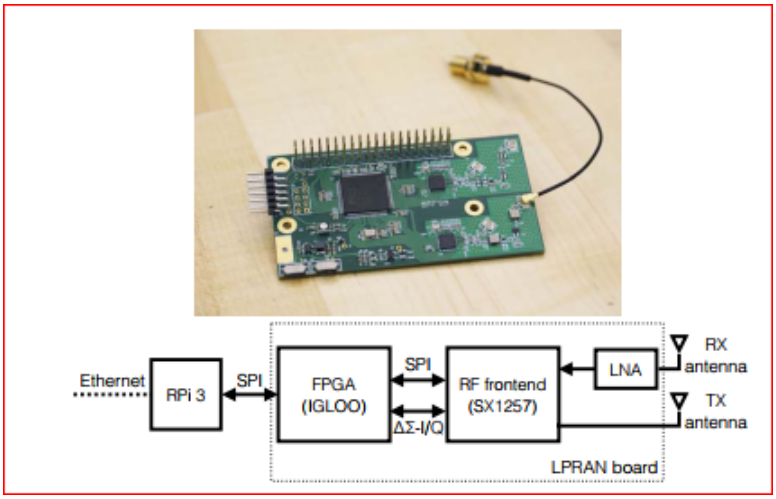


Figure 1.5: The LPRAN board and its block diagram

As most of the research is software oriented, the thesis focuses on the hardware design aspects. The problem of transitioning from one service (e.g Lora) to other service (e.g. Sigfox or NB-IoT) is nearly impossible with the same gateway or end node. A user must buy new supplier specific gateway node for communication over the new desired network. A reconfigurable LPWAN node that can be reprogrammed in the field with a new protocol, a new physical layer, or on which new investigations can be carried out without taking the pain of redesigning a product is envisioned in the thesis. This will tremendously reduce the time to market for any product in this domain. The prime focus of the thesis is to design cost and power-efficient reconfigurable LPWAN end device. The thesis motto is **One node for all**.

1.4 Research scope

WSN nodes are mostly scattered in a sensor field as shown in Figure 1.2. They have the capability of sensing, monitoring, and routing data to the sink. The WSN protocol stack consists mainly of five layers as shown in Figure 1.6 [16].

- Application layer:
  The highest layer in the protocol stack is responsible for formatting user data according to the standard and acts as an interface between the lower layers.

- Transport layer:
  It helps in maintaining the flow of data if required by sensors.

- Network Layer:
  Network layer takes care of routing the data over the network between nodes

and the sink.

- Data Link Layer:
  Also known as MAC layer, it is responsible for medium access control (MAC), frame detection and avoids collision over the air of the data packets between users.

- Physical Layer:
  This layer addresses the needs of physical channel parameters like modulation, frequency selection, data encryption, transmission, and reception techniques. It acts as a conversion medium between digital data (bits) and analog data (radio signals).

The basic hardware architecture of a sensor node consists of four subunits as shown in Figure 1.7: sensor unit, processing unit, transceiver unit, and power unit. [16].
The research's main focus will be to design and develop a reconfigurable LPWAN node PHY layeer. The designed node will operate in the sub-1Ghz band and it will be limited to ultra narrowband communications only. A proof of concept is to be verified after the hardware implementation by using development kits but due to COVID-19, it is limited to the simulational verification only. As stated earlier, sensor nodes consist of mainly four sub units but the thesis scope deals with the processing unit and the transceiver part. The sensor unit selection or power/voltage management is out of the thesis scope. The reconfigurable node estimated cost will be less than € 30 making it economical w.r.t other LPWAN solutions available.

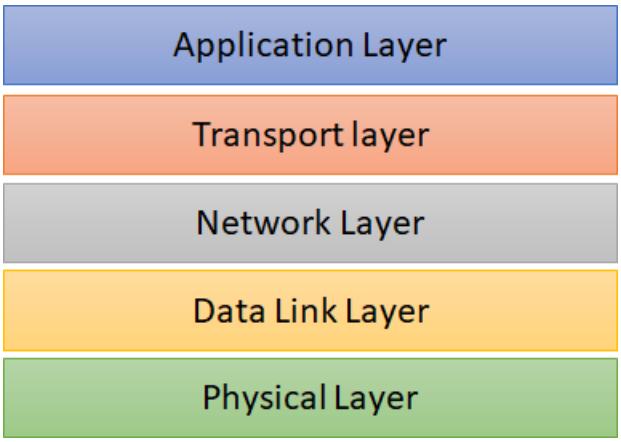


Figure 1.6: Wireless sensor network protocol stack [2]

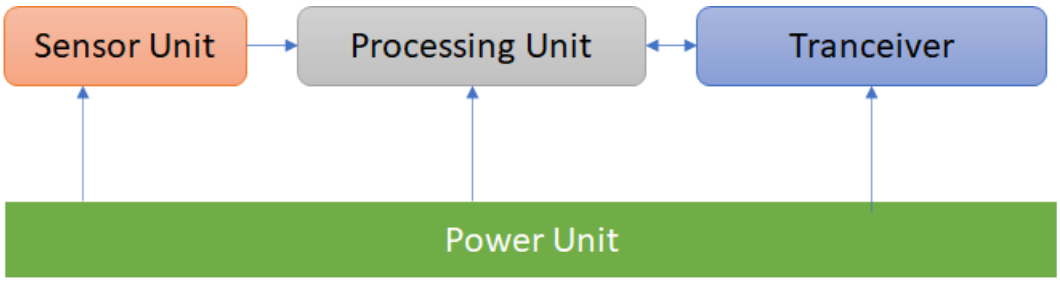


Figure 1.7: Sensor basic hardware structure [16]

1.5 Research goal

In order to design a reconfigurable node for UNB communication in the sub-1Ghz band this thesis focuses on the following points.

1. Investigate the feasibility of the reconfigurable node design according to the scope defined based upon the literature survey.
   Three different architecture implementations are investigated and analyzed for the final node design.

2. Validating the finalized architecture on development kits.
   Design validation was to be done on the evaluation kits for a proof of concept. Due to the COVID-19 outbreak, the hardware implementation was limited to simulation verification only. Simulation design and implementation are limited to the PHY layer only.

3. System performance evaluation in a given scenario:
   The bit error rate (BER) vs EbNo graph is plotted for benchmarking system performance in a wireless AWGN channel. The performance of the designed node is compared with the theoretical performance as described in [17] and MATLAB implementation.

4. The final hardware design is proposed based on the results of the simulations. The final hardware will have the following characteristics:

   - SDR-based standalone module
   - Supports sub-1Ghz band
   - Max Bandwidth 600 kHz
   - Cost $<$ €30
   - low power consumption
   - GPIO capability for interfacing sensors

System validation will be done on the parameters provided in Table 1.2.

Data Rate	100bps
Packet size	200 bits
Modulation	QPSK
Samples per symbol	16
Max Frequency offset	10 Hz

Table 1.2: Reconfigurable node hardware design parameters

1.6 Report organization

To answer the research questions, available software-defined radio (SDR) architecture is discussed in Chapter 2. The SDR architecture investigation will conclude the sensor node hardware architecture design. In Chapter 4 and Chapter 3 the FPGA and microcontroller-based wireless system modules are discussed and critically evaluated. Chapter 5 is about the integrated system testing results of the whole system. The last chapter concludes the study and proposes some recommendations for future work.

Chapter 2

Hardware Architecture Design

Abstract:

The software-defined radio basic implementation concepts are discussed in this chapter. Some available SDR platforms are studied for a hardware implementation design of the required sensor node. The three different architectural designs are investigated and an optimal LPWAN hardware architecture is proposed based on the thesis scope.

2.1 Introduction

Since the creation of the universe, human beings had vastly relied on communication for information transfer. Though the mode of communication has evolved from fire signs to high-speed wireless communication. What has remained the same is the telecommunication fundamentals from the time of Shannon [18]. With the advent of fast processing hardware like DSP, FPGA, and SoC computationally intensive algorithm solutions are now possible. Fast processing hardware requires communication engineers to solve the latest technical challenges like fast switching clock management, data transfer between different domains, power management, etc.
Software-defined radio (SDR) can be broken down into two words. First, software-defined that includes the implementation of key elements of the transmission using programming [18]. Second, the term ”Radio” that includes the means of communications through the air.
An SDR is a class of reconfigurable devices that can alter the systems software implementation or PHY after been deployed in the field. This ability of SDRs to reconfigure helps in interoperation among different standards. The reprogramming of SDR is done in software while hardware remains the same. The same SDR can be reconfigured for different frequency bands, modulation, data rates, and basic ar-

chitecture, etc. Reconfiguring the same hardware helps in lowering the time of the design cycle.
The SDR basic hardware architecture can be seen in the Figure 2.1. The main part of the SDR concept lies in the baseband processing module, where the physical layer is implemented and reconfiguration takes place when it is desired. The SDR concept can be subdivided into a transmitter module and a receiver module.

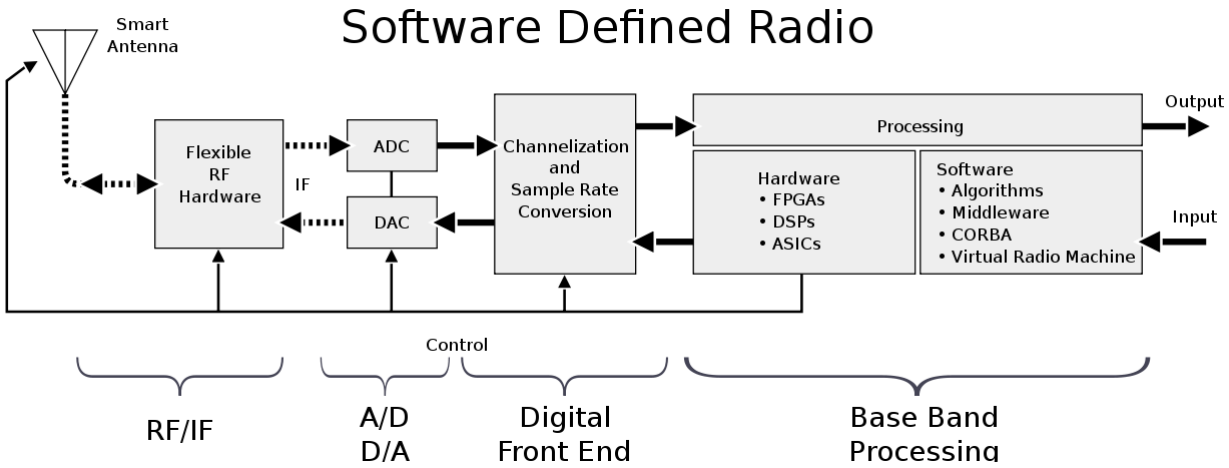


Figure 2.1: Software defined radio basic architecture [19]

2.1.1 SDR transmitter block diagram

The SDR transmitter (Tx) block diagram can be seen in Figure 2.2. The baseband processing unit acts as an interface between the user data and the rest of the SDR units. The baseband processor provides data to the digital upconverter (DUC) in dotted lines for filtering and up-conversion to the intermediate frequency (IF). The baseband sampling frequency and the digital local oscillator frequency must be equal to the digital-to-analog converter (DAC) sampling frequency to avoid signal harmonics. The digital local oscillator frequency is mostly equal to the DAC sampling frequency but the baseband sample frequency is normally lower. The baseband sampling frequency is increased by a factor of N (interpolation factor) with the help of an interpolation filter. The interpolated and filtered data is transferred to the DAC.
DAC data is upconverted to RF frequency by a mixer and an analog local oscillator. Finally before transmission, the power amplifier boosts the signal power for successful transmission via the medium.

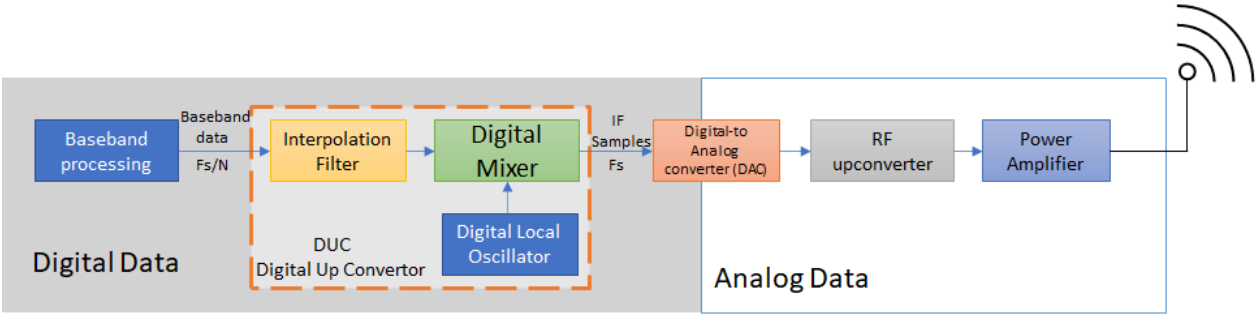


Figure 2.2: Software defined radio Tx functional block diagram [20]

2.1.2 SDR receiver block diagram

The receiver (Rx) module is the opposite of the transmitter module as discussed in the section earlier. Figure 2.3 shows a receiver block diagram of an SDR. The RF tuner receives and translates the frequency of the incoming signals to IF. The downconverted IF analog data is converted to digital samples by an analog-to-digital (ADC) converter. This digitized data is transferred to the digital down converter (DDC) block (dotted lines). The DDC consists of three major sections:

- **A digital mixer**
- **A digital local oscillator**
- **A FIR lowpass filter**

The digital mixer and a local oscillator translate the received IF samples to the baseband samples for further processing by the FIR low-pass filter. The FIR low-pass filter acts as a decimator and it also limits the baseband signal bandwidth.
The digital baseband samples are further processed by the baseband processor to recover the original data which was transmitted.

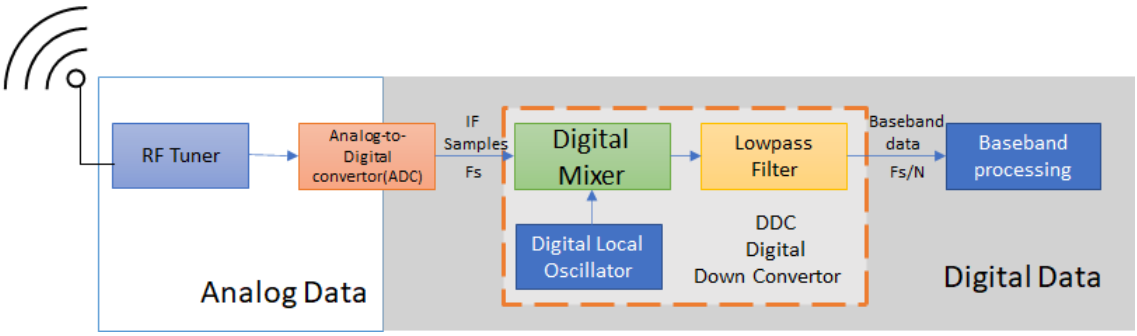


Figure 2.3: Software defined radio Rx functional block diagram [20]

2.2 Available SDR platforms

The SDR concept motivates researchers, scientists, and hobbyists which are coming up with new designs and solutions for SDR. SDR software can be an open-source (GNU Radio) or licensed (Matlab, NI Labview, etc) package. A licensed hardware requires a software subscription before any development. An SDR comes in different configurations from full transceiver reconfigurability to receiver reconfigurable only. The cost price of an SDR depends on the functionality and add-on features available.
In the thesis, low-cost ($\leq$\$ 500) SDR transceivers are studied for the tentative hardware architecture design.

2.2.1 BladeRF x40

The bladeRF x40 is a next-generation full-duplex SDR platform developed by the Nuand LLC with an operating RF frequency from 300 MHz to 3.8 GHz [21]. The BladeRF functional block diagram can be seen in Figure 2.4. The main core of SDR lies in the Altera Cyclone IV FPGA, which performs signal processing, data buffering and controls RF transceiver chip. The FPGA can be programmed via USB/JTAG interface with the help of free open source software available online. The Lime Micro LMS6002D RF transceiver chip is capable of managing RF signals from simple Frequency modulation (FM) to latest 4G LTE [21]. The price for the SDR is \$ 420.

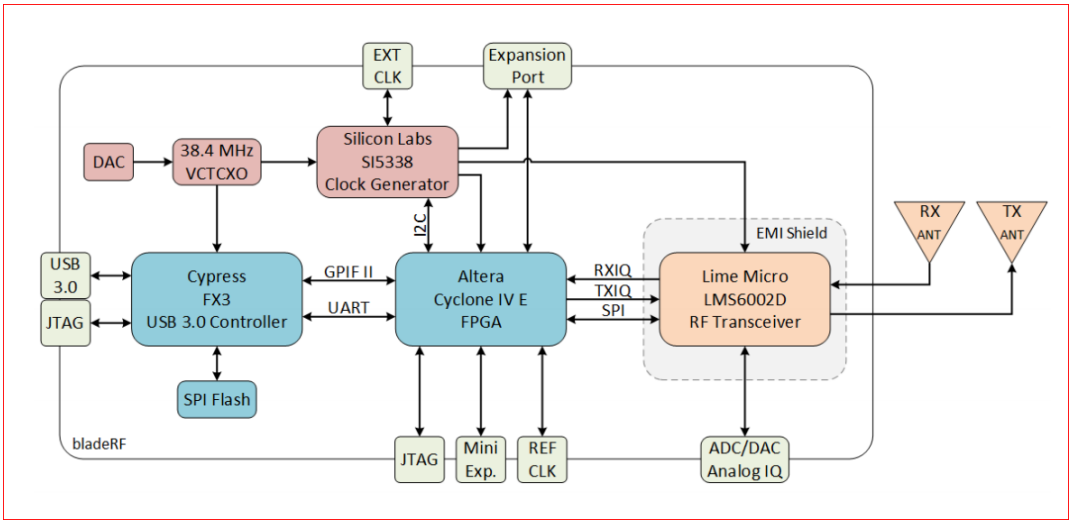


Figure 2.4: The BLadeRF functional block diagram [22]

2.2.2 LimeSDR mini

LimeSDR mini is another option for low budget SDR enthusiasts with lower specifications than its' predecessor LimeSDR, developed by the Lime Microsystems. The basic block diagram of the LimeSDR mini is given in Figure 2.5. It contains Altera MAX 10 FPGA that is interfaced to the Lime LM7002 RF transceiver and USB 3.0 controller for data communication. The FPGA programming is done via the JTAG port. The RF transceiver IC covers range from 10 MHz up to 3.5 GHz. It has only one transmit and one receive channel with a bandwidth of 30.72 MHz and sampling frequency of 30.72 MSPS. The price of limeSDR mini is $ 175.

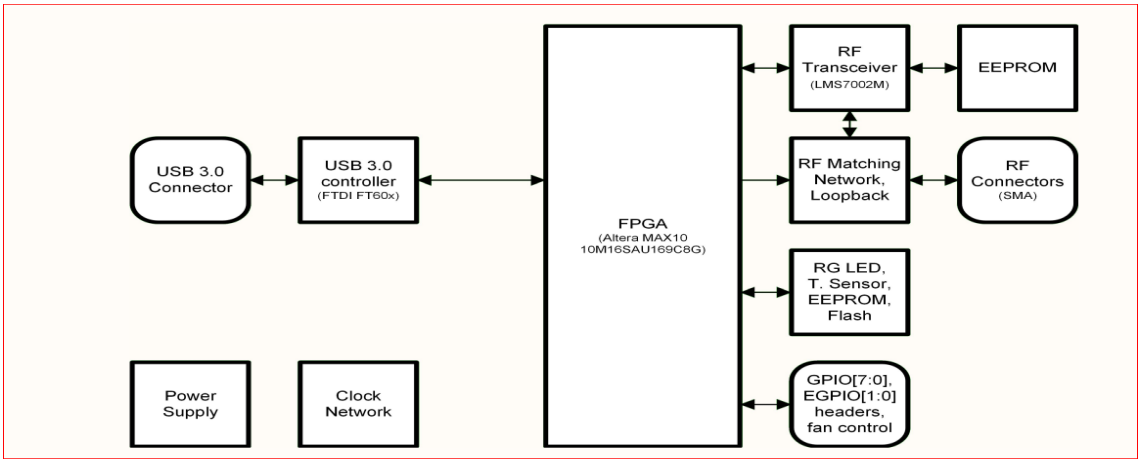


Figure 2.5: LimeSDR mini block diagram [23]

2.2.3 ADALM PLUTO

The ADALM-Pluto SDR is a portable RF lab developed for understanding SDR concepts. It can generate and acquire RF signals from 325 MHz to 3800 MHz with a sampling rate varying up to 61.44 MSPS. It can easily be operated via a USB interface in Windows and Linux interface. A basic block diagram is shown in Figure 2.6.

ADALM-Pluto contains AD9363 high-performance RF agile transceiver designed by the Analog Devices. The RF transceiver is interfaced with the Xilinx Zynq all-programmable SoC (AP SoC) [24]. The Zynq SoC is complemented by an ARM-based processor with hardware programmability of an FPGA.

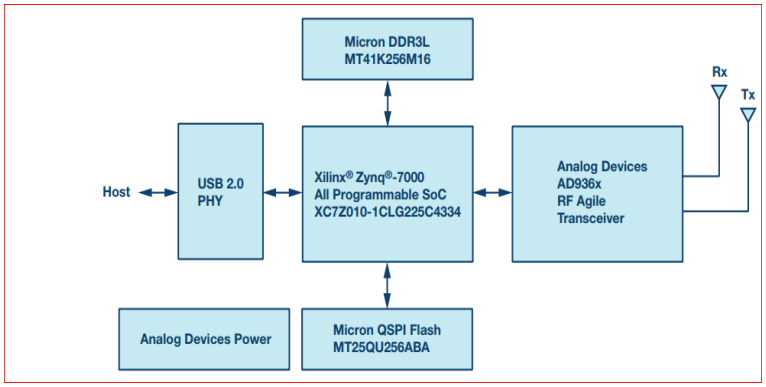


Figure 2.6: ADALM-PLUTO block diagram [24]

2.2.4 Conclusion

Table 2.1 lists some of the SDR platforms studied for reconfigurable sensor node hardware design. The SDR platforms lesser than $ 500 are investigated for node hardware layout design.

Table 2.1: SDR platform comparison

Name	Frequency (MHz)	Bandwidth	ADC sampling rate (Msps)	Tx/Rx capability	Processing chip	RF chip used	Cost($)
AD-FMCOMMS4	70-6000	0.2-56 MHz	61.44	2/2	No	AD9364	399
ADALM-Pluto	325-3800	upto 20 MHz	0.0652 - 61.44	1/1	Zynq Z-7010	AD9363	150
AirSpyR2	24-1700	10 Mhz	2.5-80	Rx only	No	Rafael Micro R820T2	169
BLadeRF	300-3800	upto 28 MHz	40	1/1	Altera Cyclone IV	LMS6002	420
FREESRP	70-6000	56 MHz	61.44	1/1	Xilinx Artix 7	AD9364	420
HACKRF	1-6000	20 MHz	2-20	1/1	Microcontroller	Maxim 2837	299
LimeSDR mini	10-3500	30.72 MHz	30.72	1/1	Altera Max 10	LMS7002	175
Myriad RF-1	300-3800	28 MHz	40	1/1	No	LMS6002	299

The AD-FMCOMMS4 and the Myriad RF-1 are only RF transceiver chipboards with an FPGA Mezzanine Card (FMC) connector for an external processor. An external processor can be an FPGA/SoC/microcontroller depending on the application requirements. The AirSpyR2 can only receive RF signals which are then transferred to an off-board processor for further processing. The three mentioned SDR solutions are not appropriate w.r.t the scope of the thesis. The HackRF SDR is a standalone

solution that has a microcontroller as the main processing device. The rest of the SDR mentioned in the table have FPGA based processing. SDR insight can be concluded that the basic elements of an reconfigurable device are an RF transceiver chip, data processing element, and power management module. Data processor can be an FPGA, an SoC, a microcontroller or it can be a mixture of multiple data processing devices.

2.3 Reconfigurable hardware layout

After the architecture study of the SDR, it is clear that there can be many alternatives to design a reconfigurable SDR sensor node. A block diagram of a sensor node is the same as Figure 1.7. The PHY can be implemented on different baseband processors such as an FPGA, GPP and DSP [25]. The reconfigurable node hardware design according to the sensor node block diagram can be implemented on a single chip platform, two-chip platform, or a three-chip platform.

2.3.1 One chip solution

A one chip solution of reconfigurable sensor node is shown in Figure 2.7. A radio

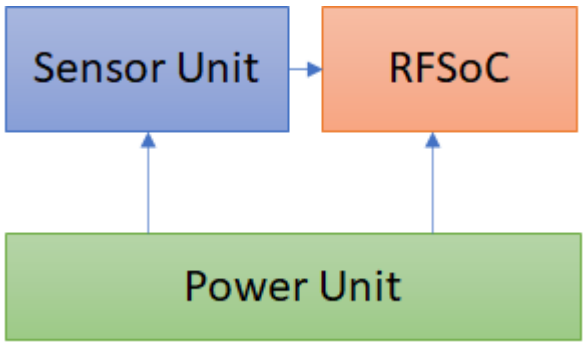


Figure 2.7: Illustration of a one chip design for a reconfigurable node using RFSoC as a single processing, and a transceiver unit.

frequency system on chip (RFSoC) combines the RF upconversion/downconversion and direct RF signal sampling onto a single chip. This removes the need for buffers in data paths and packs the RF components into a single chip package [26]. The Xilinx has produced industry-only RFSoC adaptable platform [27]. The Xilinx RFSoC gen 1 can support up to 4 GHz analog bandwidth. The RF sampling rate is 4.094 GSPS with 12 bit sample depth. Built-in RF-DAC and RF-ADC remove the need of external DAC/ADC for upconverion or downconversion respectively.

	RF Transceiver IC requirements
Frequency Band	862 MHz - 860 MHz (European ISM band)
Data interface	I/Q data interface
Power consumption	30mA-70mA (Active current rating)
cost(€)	$\leq$ 30 (complete sensor node)

Table 2.2: RF tranceiver IC required parameters

This is not an optimum solution for a hardware layout due to the expensive RFSoC chip that costs approximately $ 4000.

2.3.2 Two chip solution

The next option is a two-chip solution which can be seen in Figure 2.8. Before finalizing the baseband processing chip it is wise to select an RF transceiver IC. The RF IC will govern the interfaces required for interconnection with the baseband processing unit. The RF transceiver chip selection criteria are given in Table 2.2 based on the scope defined in chapter 1. The power consumption of the available LPWAN nodes gives an estimate for the desired node power consumption. The Laird RM186 LoRa transceiver module consumes 30.9 mA while transmitting at 3.3 V supply voltage. Another Lora solution manufactured by Techship (AcSIP S76s) consumes 65 mA at 3.3 V supply voltage. Sigfox module consumes 33 mA (InnoComm SN10) to 58 mA (Radiocrafts RC1682) at 3.3 V supply voltage. Based on the current ratings of the available LPWAN modules the reconfigurable node RF transceiver IC must have low or approximately equal power consumption compared to the available modules in the market.

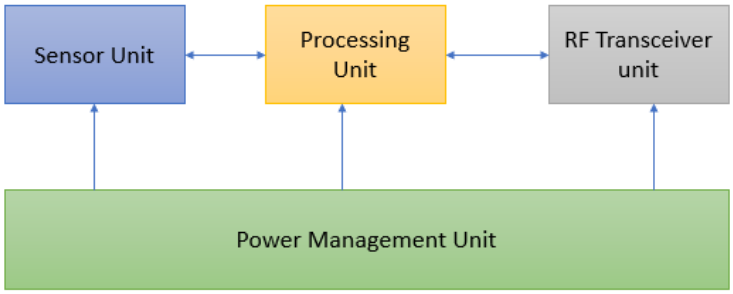


Figure 2.8: Illustration of a two chip design for a reconfigurable node using an FPGA or a microcontroller as a processing unit and an RF IC as a transceiver unit.

2.3.2.1 RF Transceiver Selection

According to the thesis scope, two RF ICs were found to satisfy the requirements given in Table 2.2. One IC is from the Semtech corporation and the other one is from Atmel Corporation. The Semtech RF IC was used in the OpenChirp project discussed earlier for the development of the LPRAN hardware board. The differences between the two ICs are summarized in Table 2.3. The Atmel transceiver has

Table 2.3: Available RF transceiver IC

		Semtech 1257	Atmel AT86RF215IQ
Cost ($)		6.6	4.93
Size		5 x 5 mm	7 x 7 mm
RF Band		862-960 MHz	863-879 MHz & 2.4 GHz
Tx Bandwidth		210 - 870 kHz	80 - 1000 kHz
RX Bandwidth		250 - 750 kHz	160 - 2000 kHz
IQ data standard		1 bit serial LVDS	1 bit serial LVDS/SLVDS
Current Rating	Tx	58 mA @ 3.3 V and -5 dBm	67 mA @ 3.0 V and - 5 dBm
	RX	20 mA @ 3.3 V	23 mA @ 3.0 V
	Standby	1.5 mA @ 3.3 V	6.28 mA @ 3.0 V
	Sleep	0.5 uA @ 3.3 V	30 nA @ 3.0 V
Noise figure (dB)		7-10	4.5
Tx power max (dBm)		+8	+16
ADC/DAC sampling frequency		32 or 36 MSPS	32 MSPS
Available Transceiver		1Tx 1 Rx	1 Tx 2 Rx

a bit high (4.7 % in transmission mode) power consumption at 3.0 V as compared to the Semtech 1257 IC. A lesser noise figure of about 2.5 dB and less expensive than its competitor makes it a favorable choice. The Atmel IC can also operate on dual bands mentioned in the table because of a separate transceiver. Based upon the parameters (highlighted in green) in Table 2.3 the Atmel AT86RF215IQ IC is selected as an RF transceiver IC for the reconfigurable node.

RF IC interface layout

The transceiver IC selected in the last section will now define the interface connections with the baseband processing unit. The Atmel RF IC transfers data in and out at 64 MHz with double data rate (DDR) technology. In DDR technology, data is transferred at both of the clock edges to the baseband processor. Data is transferred serially via a differential (LVDS) interface. The effective data rate is 128 Mb/s composed of 16bits at 4MHz sampling frequency for each of the I-data and Q-data streams [28]. The I/Q interface layout of the Atmel86RF215IQ is given in Figure 2.9. The receiver and transmitter interfaces of the IC are serial and differential as shown

in Figure 2.9. The AT86RF215 IC generates a clock signal (RXCLKP/RXCLKN) for the external baseband processor to get it synced to the internal clock. The same clock can be used by an external processor to feed TXCLKP/TXCLKN to the RF IC back for data transmission.

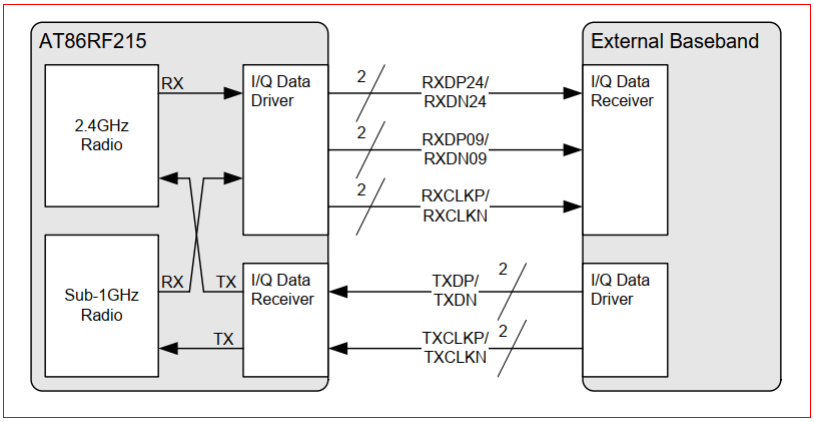


Figure 2.9: Atmel RF86215 IQ data interface Layout [28]

IQ word format

The data transmission rate is same for the IQ interface which can handle different sampling rate from 400 ksamples/s up to 4 Msamples/s. The sample rate must be same for TX I/Q data streams as configured in the internal TXDFESR register. Zero words must be padded by baseband processor in between data samples as shown in the Figure 2.10 if the data sampling rate is lesser than 4 MSPS. The sample rate of RX I/Q data is consolidated by the transmitter interface of RF IC with zero paddings (Figure 2.10), to keep the interface data rate 128Mb/s. The I/Q interface serializes/deserializes the 32 bit word. The 32-bit word contains two bit I-channel

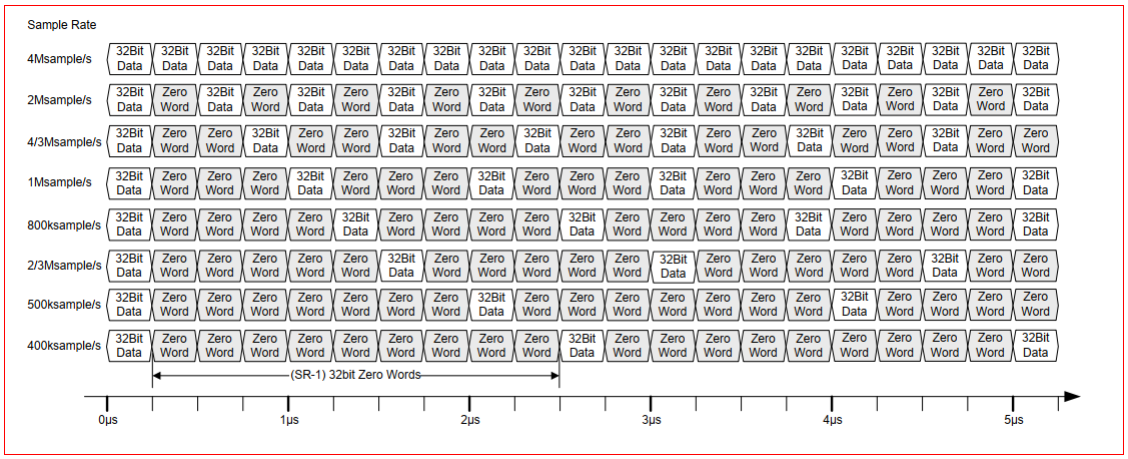


Figure 2.10: I/Q word format [28]

synchronization pattern followed by 14 I-data and then a two bit Q-channel sync pattern followed by 14 bit Q-data as given in Table 2.4. The external baseband processor must insert the required number of zero words according to the sample rate (SR) register, else the data will be discarded by the RF IC I/Q interface.

Table 2.4: I/Q data interface word frame format

Bit[31:30]	Bit[29:16]	Bit[15:14]	Bit[13:0]
I_SYNC=0b10	I_DATA[13:0]	Q_SYNC=0b01	Q_DATA[13:0]

RF IC control interface

The operation of the transceiver IC is controlled via serial peripheral interface (SPI) interface that is connected to the external processor. The external processor acts as an SPI master for accessing registers and frame buffers of AT86RF215IQ.

Table 2.5: SPI signals

SPI Signal	Direction	Description
SCLK	Input	SPI clock signal
SELN	Input	SPI select signal, active low
MOSI	Input	SPI data master output, slave input signal
MISO	Output	SPI data master input, slave output signal

RF interface functional drawing

The RF IC requires 5 differential (LVDS) data communication ports for information transfer with the external processor. Three differential signals are required at the reception channel; one differential signal for the clock and the other two data signals for both RF bands one each. Two differential signals are required for the transmission channel; one for clock input and the other for data transfer. This can be seen in Figure 2.9. To control the interface selection and data flow parameters the SPI interface is required. The SPI protocol requires 4 connection lines(serial clock (SCLK),select active low (SELN),master out slave in (MOSI),master in slave out (MISO)) for communication between RF transceiver chip and baseband processor.
The updated two chip reconfigurable design can be seen in Figure 2.11.

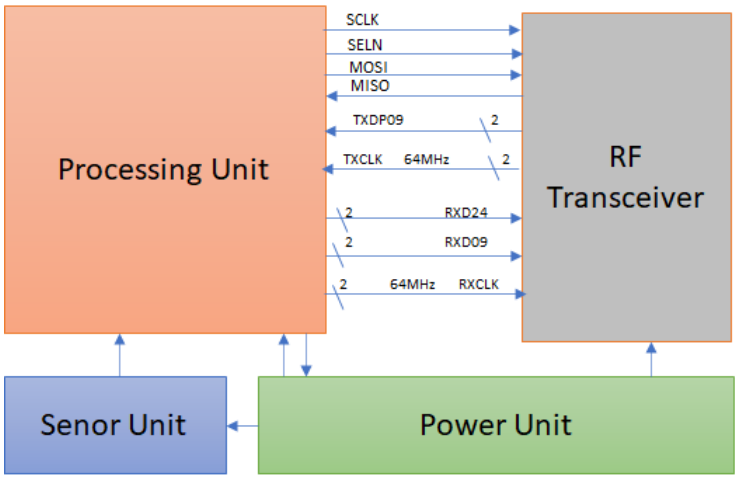


Figure 2.11: RF interface connection diagram

	Baseband processor interface
Minimum input clock rate	64 MHz (DDR) or 128 MHz (Single clock edge)
Data interface	5 x I/Q LVDS interface
SPI interface	RF IC registers configuration
Power consumption	lowest
cost(€)	$\leq$ 25.07 (Node cost without RF IC)

Table 2.6: RF tranceiver IC required parameters

2.3.2.2 Baseband Processing unit

Baseband processing tasks can be done on an FPGA, DSP, SoC/PSoC or microcontrollers [25], [29]. Which semiconductor device is used is purely dependent on the requirement of the application. Selection depends on the number of required resources for a specific implementation. A best-fit solution for one problem doesn't mean it will be best for every solution. Each semiconductor device has its own merits and demerits. We investigated different options for the reprogrammable LPWAN node baseband processor design. The RF IC selection has resulted in additional requirements for the baseband processor interface as listed in Table 2.6.

System on Chip (SoC) FPGA

An SoC FPGA is an integrated chip with a central processing unit, a memory unit, reconfigurable logic elements and input/output ports on a single substrate [30]. There are many types of SoC available for the intended application. A general-purpose SoC is characterized by large processors, large-on-chip cache, memory controllers and a few high speed interface peripheral connectivity [31]. A high-end SoC often

has multiple processor cores, multiple caches, memory controllers and different assortments of interfaces [31]. SoCs are preferred over conventional hardware due to their compactness and most of the required features are on a single chip. Less external interconnections are required for data transfer. SoCs from different vendors have been considered to be used as a baseband processor. The comparison of available SoCs is given in Table 2.7. The lowest configuration SoC FPGAs by the Altera and the Xilinx have 25k logic blocks. The SoC FPGA by the Altera, the Xilinx and the Atmel are SRAM-based while the Microsemi provides flash-based SoC FPGA. The price range for a 25k logic block SoC FPGA by any vendor is from €40 to €47. An on-chip ADC is only available on the Xilinx SoC FPGA. On-chip ADC can be beneficial in sampling the sensor data.

Table 2.7: SoC comparison table

	Altera SOC V CSEA2	Xilinx 7007S	MicroSemi(Actel)		FPSLIC(Atmel)
Logic Blocks	25k	23k	6k(M2S005)	27k(M2S025)	5K
Block Memory	10MiB	1.8MiB	191kB	400kB	16KB
Number of I/O	145	128	84		93
Controller	Dual Core ARM cortex A9	Single Core ARM Cortex A9	ARM M3 processor (RISC based)		AVR 8Bit RISC
FPGA type	SRAM based	SRAM based	Flash based		SRAM based
Interface	SPI, QSPI, USB, Ethernet, I2C and CAN	2xQSPI, 2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI	2xSPI, I2C, CAN		UART, 2 Wire Serial
Package	UBGA-484	CSBGA-225	VF(G)256	FCSG325	LQFP-144
Dimension (mm^2)	19x19	13x13	14x14	11x11	22x22
Cost(€)	41.79	40.28	17.54	46.54	10.53
Voltage(core)	1.1 V	1.2V	1.2 V		3 V
Remarks	LVDS	LVCMOS,LVDS and SSTL	LVDS,LVTTL,LVCMOS		———
ADC	No	2x12Bit ADC (1 MSPS)	No		No

Conclusion

According to the project's scope, an SoC FPGA didn't seem to be a suitable option w.r.t. the high price per logic element. Although they qualified the interface requirements (5x LVDS interface, an SPI, and minimum interface clock of 64 MHz support) as listed in Table 2.6 but IC cost is a bottleneck in SoC FPGA based solution. In chapter 3 logic elements requirements are discussed in more detail.

Field Programmable gate Array (FPGA)

FPGA is a family of semiconductor devices that consists of a matrix of a configurable logic block (CLB) that can be interconnected in many ways via programmable

interconnects [32]. An FPGA can be configured using a hardware description language by the customer after it is delivered. The reconfigurability after deployment has earned them the name of field-programmable. FPGAs comparison Table 2.8 presents the available FPGAs that can be used in a two chip design.

Table 2.8: Available FPGA comparison for two chip design

	Lattice LFE5UM-85F-6MG285C	Xilinx(Spartan 7) XC7S50-1FTGB196C	Altera (10M50SCE144C8G)	MicroChip ProASIC Plus(APA075)
Logic Blocks	84k	52k	50k	75k
Block Memory	3744 kb	2700 kb	1638 kb	27 kb
Number of I/O	118	100	101	158
FPGA type	SRAM	SRAM	Non Volatile FPGA	Flash Based
Interface	LVDS25 (400 MHZ)	LVDS25	LVDS (up to 500 Mbps)	No lvds
Package	285-CSFBGA	196-CSBGA	EQFP-144	PQFP-208
Dimension (mm^2)	10 x 10	15x15	22x22	28x28
Cost(€)	25.474	43.32	43.61	58.65
Supply current	212 + 9.5 mA(SERDES)	300mA power on ,95 VCCInt	25mA/I/O	15mA quiscent

Conclusion

An FPGA seems to be a good option for the reprogrammable node processing unit based upon cost, interface and logic elements count. The SRAM-based FPGA takes high starting currents due to the programming of the logic elements. The LPWAN node will be in sleep mode mostly, so every wake-up call requires FPGA to be reprogrammed. The SRAM or partial SRAM FPGAs were not considered based upon power consumption. More discussion is done in chapter 3 about the FPGA type and logic resources requirements. A true flash-based FPGA retains the logic programming code even after set to sleep. The logic retention removes the need for reprogramming and conserves valuable node power. The fLash-based FPGAs are mainly manufactured by the Microsemi Corporation.

Microcontrollers

A microcontroller is a small computer on a single chip similar to SoC but it is less sophisticated. An SoC may contain a couple of microcontrollers in a single chip as a component. The recent development in the silicon industry has also pushed a couple of microcontrollers into a single die [33]. Microcontrollers are used in automated systems, embedded products, and devices. Some microcontrollers can operate as low as 4 kHz making them power saving devices. A nice feature about microcontrollers is that they retain the algorithm even in sleep mode. Microcontrollers initially

were only programmed by assembly language, but now high-level programming languages such as C, Python, and JavaScript are widely used by programmers [33]. To make a required standalone reconfigurable node, the microcontroller must be capable of processing I/Q interface data at both clock edges with a 64 MHz clock or 128 MHz at any one edge of the clock. As per the investigation, there were no microcontrollers found that can operate on both clock edges. Microcontrollers having a minimum clock frequency of 128 MHz are listed in Table 2.9 for comparison.

Table 2.9: Microcontrollers with operating frequency 128 MHz or above

		STMicroelectronics	NXP	Cypress	MicroChip	Toshiba
Name		STM32F730R8T6	LPC54005	CY8C6036BZI-F04	ATSAMS70J19A-AN	TMPM4G6FEFG(DBB)
Price(Euro)		4.46	5.64	6.28	7.15	7.88
Package		LQFP-64(10x10mm)	LQFP-100(14x14mm)	BGA-124(9x9mm)	LQFP-64(14x14)	LQFP-100(14x14mm)
Core		ARM cortex M7(32 bit)	ARM cortex M4(32bit)	ARM Cortex M4F(32bit)	ARM cortex M7	ARM cortex M4
LVDS interface		No	No	No	No	No
Data Ram Size		276 kB	360kB	128kB	256kB	128kB
Interface		I2S, SAI, SPI, USB	I2C, I2S, SPI, USART	UART, SPI, I2C, S/PDIF,	SPI, UART/USART, USB	I2C, SPI, UART
I/O voltage		1.7 V to 3.6V	1.71 V to 3.6 V	1.7 V to 3.6 V	1.7 to 3.6V	
Max Clock(MHz)		216	180	150	300	160
Number of I/O		50	64	104	44	91
ADC/DAC		3x12bit/2x12 bit	12bit	12bit/12bit	5 channel 12 bit/12 bit	12 bit/ 8 bit
Supply Voltage		1.7∼3.6V	1.71∼3.6V	1.7 V to 3.6 V	1.7 V to 3.6 V	2.7∼3.6V
Power consumption	Run	138mA(max clock)	35mA	10mA approx. for both cores	57mA(90ma max)	50mA
	Stop	0.45mA	8.3mA	4mA(core only)	20mA	9.5mA
	Standby	1.09uA(at 250C)	55uA	7uA	3.8∼8uA	9.6uA

Conclusion

The microcontroller only solution looks good as a baseband processor due to low price but power consumption at high frequencies is higher due to increased clocking/switching activity. This increased clock speed is not desired, as the reconfigurable maximum available bandwidth is 600 kHz as per LPWAN standard. Moreover, microcontrollers lack an LVDS interface which is a compulsory requirement of the I/Q interface. This interface requirement can be fulfilled by using an interface converter chip between RF IC and microprocessor. Lastly, the microcontroller's inherent nature of sequential operation requires microcontrollers to be faster than 128 MHz in order to avoid overrun or overflow conditions.

2.3.3 Three chip solution

Finally, a three-chip solution is investigated in light of the requirements imposed by the reconfigurable LPWAN nodes. A three-chip solution block diagram can be seen in Figure 2.12. As explained earlier in Section 2.1, IF samples are first down-converted or upconverted before or after baseband processing. An FPGA is the best option for digital upconversion and downconversion due to flexibility and a high degree of programmability [34]. Parallelism is inherent in an FPGA due to which it can implement and perform parallel computation as required in the filtering process. More information on FPGA types is provided in chapter 3 but for the solution, a true flash-based FPGA is selected as an upconverter or downconverter. The flash-based FPGAs are produced by the Microsemi only.

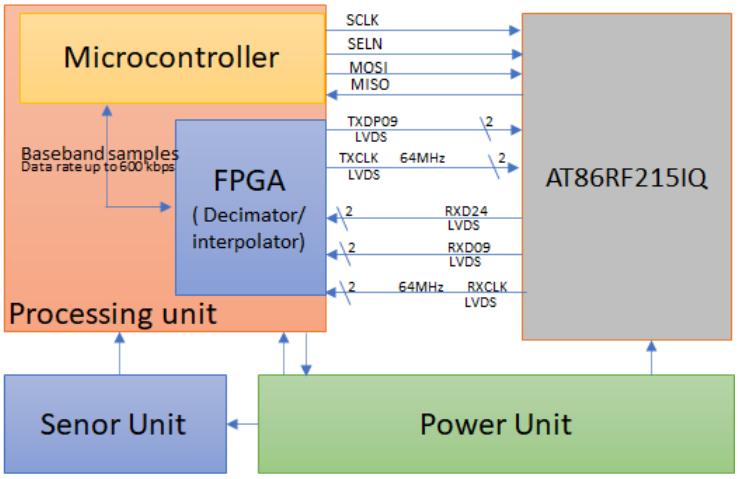


Figure 2.12: Illustration of a three-chip design for a reconfigurable node using an FPGA as a data reformatting and filtering device, a microcontroller as a signal processing device inside the processing unit, and the Atmel RF IC as a transceiver device.

The microcontroller specifications can now be relaxed as high-speed data will be upsampled/downsampled inside an FPGA. The maximum allowable analog bandwidth as per LPWAN ECC standard specifications can be 600 kHz. The Nyquist sampling criteria for aliasing free signals states that sampling frequency must be 1200 kHz or above. The RF IC selected can support a maximum of 4 MSPS at the I/Q interface so a microcontroller operation clock frequency of 4 MHz or more can do the tasks for the desired node over the entire LPWAN frequency band.

Conclusion

Low power FPGAs and their inherent parallelism nature make them an optimum solution for any design. They can be used as a sampling rate converter with low

pass filtering. The FPGA will act as an interface between the RF IC and the microcontroller. It will receive data from the RF IC serially and transfers it to the microcontroller and vice versa. The microcontroller will further process the low data rate signals for recovery of the information. Tentative interface requirements of the FPGA and RF IC are summarized in Table 2.10

Table 2.10: Reconfigurable node tentative requirements based on thesis scope

Frequency band	862 MHz - 860 MHz (European ISM band)
Minimum analog bandwidth	600 kHz
FPGA Clock Frequency	64 MHz (DDR) or 128 MHZ
Microcontroller clock frequency	4 MHz (minimum)
Number of SPI interface	3 (minimum)
Number of LVDS interface	5
Cost	€30
Number of LVDS interface	5
Dimensions (cm^2)	$\leq$ 5 x 5

2.4 Conclusion

There are three alternatives for designing a reconfigurable LPWAN node, namely one-chip design, two-chip design, or three-chip design. The one-chip design is the most convenient one as all of the required resources are in a single chip (RFSoC). The single-chip solution doesn't fit the node requirement of low cost. An RFSoC is too expensive to be used as a transceiver in a reconfigurable node.
The two-chip solution that contains an FPGA or a microcontroller as the main processing unit seems to be a good option. The microcontroller-based solution is power expensive as it requires an interface level converter for LVDS to TTL/LVCMOS. Fast sampling clock requirements is also a drawback in a microcontroller-based design. Further investigation of the FPGA-based two-chip solution w.r.t. hardware resources and power consumption is done in Chapter 3.
The three-chip solution seems to be an optimum solution in which the FPGA acts as a sample rate converter and low pass filter. While baseband processing is performed by the microcontroller at the lower sampling rates. The three-chip design is further investigated in Chapter 3 for FPGA modules and Chapter 4 for the microcontroller-based subsystems.

Chapter 3

Field Programmable Gate Array (FPGA)

In this chapter, the FPGA basic architecture and the underlying technologies are discussed. Further investigation is done on the two-chip design and the three-chip design. The FIR low pass filters and the CIC filters are investigated for UNB signals. Results of the designed filters are discussed based upon power consumption and the required amount of FPGA resources.

3.1 Introduction

The interesting thing that happened in the silicon industry is the explosion of the number of transistors on a unit area. Moore's observation states that the number of transistors on a chip will be doubled every two years. [35]. The industry is transitioning from simple central processing unit (CPU) and microcontrollers based devices to single-chip solutions having multiple architectures, one of the well-known types is an field programmable gate array (FPGA). FPGAs can be configured by the designer or customer after manufacturing or after their deployment. In the past, FPGAs were used for low complexity designs but due to unprecedented logic density and amalgamate of different features like DSP blocks, clocking circuitry, and memory blocks made them a favorable option for any design. The advantage of an FPGA based design is that both hardware and software designs can be started simultaneously for development. Multiple design iterations and testing can be done before freezing the final design. An ASIC fabrication can take weeks and modifications are not possible after the fabrication process. On the contrary, an FPGA can be reconfigured in minutes at the desktop [36].
An FPGA basic architecture is vendor-specific but in general, they have the same underlying concept as shown in Figure 3.1. A general FPGA contains CLBs, config-

urable I/O blocks, and programmable interconnect.

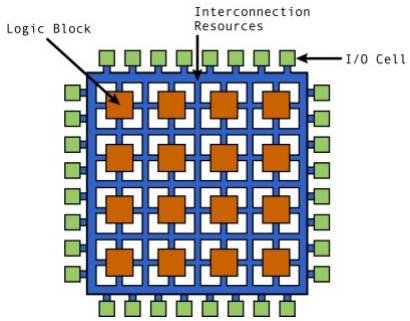


Figure 3.1: FPGA generic architecture [36]

3.1.1 Configurable logic blocks

The FPGA basic building block consists of CLBs (brown colored Figure 3.1). It consists of vendor specific programmable logic structure. Different vendors have different number of logic elements in a single CLB. A CLB is connected to the other CLBs with the help of programmable interconnection for implementing the logic design. The CLBs contain Look Up tables (LUTs), multiplexers, and flip flops. Special flip-flops are used as a clocked storage elements [37].

3.1.2 Configurable I/O blocks

A configurable I/O block (Figure 3.1 ”colored green”) contains input buffer and output buffer to carry logic into or out of the FPGA environment. These are tri-state buffers, configured according to the application requirements.

3.1.3 Programmable interconnects

Programmable interconnects as the name suggest, connect different CLBs to the rest of the available elements depending on the design specifications. A basic block diagram of programmable interconnections is shown in Figure 3.2. There is a mixture of short lines (black arrow lines) and long lines (black solid lines) for interconnection. The special long lines are known as global clock lines intended for clocking purposes. These special clock lines are connected to CLBs and clock buffers for low latency and fast propagation of the signal. [37].

3.1.4 Clock circuitry

The special I/O blocks with fast clock driving buffers are distributed around the chip. They connect clock input pads to the global clock lines for low latency across the FPGA dye.

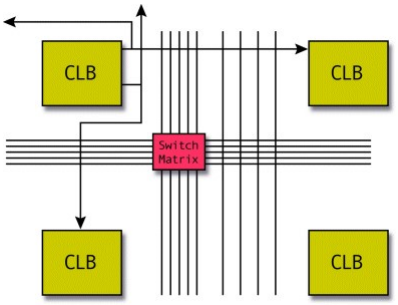


Figure 3.2: Programmable interconnect (Xilinx Fpga) [36]

3.2 Types of FPGA

In the race of FPGA reconfigurability, different programming technologies had been introduced by vendors. The difference lies in the implementation of the technology. Some of the renowned programming technologies are the static memory FPGA, the flash-based memory FPGA and the anti-fuse FPGA [37].

3.2.1 Static memory

The static random access memory (SRAM) based FPGAs use static memory cells which are distributed throughout the FPGA dye. The SRAM cells are used to program the routing interconnections and it also retains the code for the CLB logic design. The SRAM-based FPGA technology is widely adopted due to its use of the standard CMOS process technology. The SRAM-based FPGAs can be indefinitely reprogrammed theoretically. The main disadvantage of SRAM-based FPGA is the volatile nature. The moment it is powered off the logic code disappears. Due to this reason, external nonvolatile memory is attached to reconfigure the FPGA after rebooting. Programming of an FPGA from an external source increases the device cost, area overhead, and power requirements. Programming externally can be a security concern in a critical application as the configuration file is readable by the intruder. In some cases, it can be decoded also. These security concerns are addressed by SRAM-based FPGA manufacturers by pushing programming flash into the FPGA dye.

3.2.2 Flash programming

Due to the volatile nature of SRAM-based FPGAs, the flash-based FPGAs emerged as an alternative. A flash-based FPGA can retain the logic code even after it is powered off. However, they have a limit on the number of times it can be reconfigured or reprogrammed. The Flash-based technology uses non-standard CMOS process technology. This is one of the reasons that the SRAM-based FPGAs are ahead in the number of transistors in a chip unit area.

3.2.3 Anti-Fuse technology

This technology is an alternative to both the technologies as discussed earlier due to its non-volatile nature and area efficiency. The main drawback of the technology is non reprogrammability after it is programmed once. Programming can be done either by the vendor or at a user's desktop.

3.2.4 Examples of FPGA families

The Table 3.1 shows different FPGA families availble in the market for different technology type. Most of the vendors manufacture the SRAM-based based FPGAs. The Flash-based FPGAs are manufactured by limited vendors like the Microsemi (Actel).

Table 3.1: Available FPGA families [36]

SRAM	FLASH	Anti-Fuse	Hybrid (Embedded FLash/SRAM)
Altera Stratix	Actel ProAsic	Actel SX	Lattice XP Family
Cyclone II			
Atmel AT6000			
AT40K	Actel Igloo	Quicklogic Eclipse	
Lattice EC and ECP			
Xilinx Spartan and Virtex			

3.3 Two chip FPGA based design

A larger amount of logic elements, numerously inbuilt features, and instant programmability by HDL have made FPGAs a favorable choice for approximately every system design. A two-chip solution was discussed earlier in Section 2.3.2 in which an FPGA acts as the main processing unit. After the signal processing, information is transferred directly to the RF transceiver IC for upconversion over the serial

interface. For the receiving part, this is done in the reverse order. The baseband processor comprises of two top modules, a QPSK transmitter and a QPSK receiver (Figure 3.3).

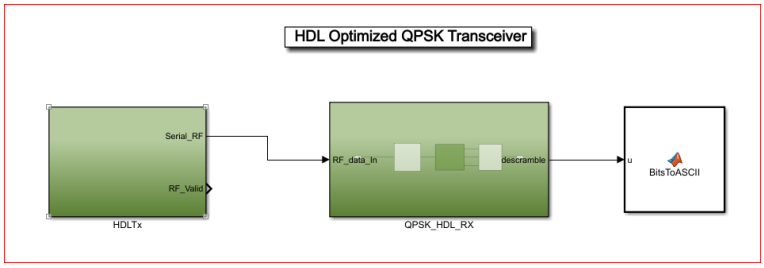


Figure 3.3: QPSK transceiver block diagram

3.3.1 QPSK transmitter

A QPSK transmitter is designed using HDL compatible blocks available in the Simulink. The HDL compatible blocks can be used to generate VHDL/Verilog code directly from the Simulink implementation and design verification. The HDL codes are used in the synthesis and the power estimation of the design in the LiberoSoc 12.2v software (free license provided by Microsemi Corporation). The HDL based QPSK transmitter is adapted from the MATLAB example available online [38]. The block diagram of the QPSK transmitter is shown in Figure 3.4. The QPSK transmitter consists of a data generation and packetization block, a symbol mapping block, a pulse shaping block, and an interface formatter block. Pipeline registers are used between blocks and components for reducing the combinatorial path latency and achieve maximum clocking frequency.

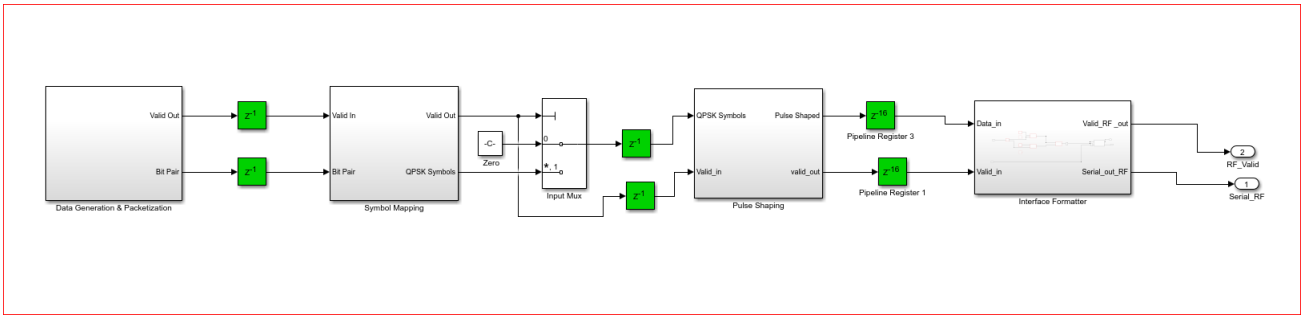


Figure 3.4: QPSK transmitter block diagram

3.3.1.1 Data generation and packetization

The data generation and packetization module accepts the incoming serial data bits and converts them into a packet of 200-bits. A 26-bit barker code header is inserted by the transmitter for packet detection at the receiver side by auto correlating the receiver's matched filter output. The block diagram for the data generation and paketization and generation module is shown in Figure 3.5.

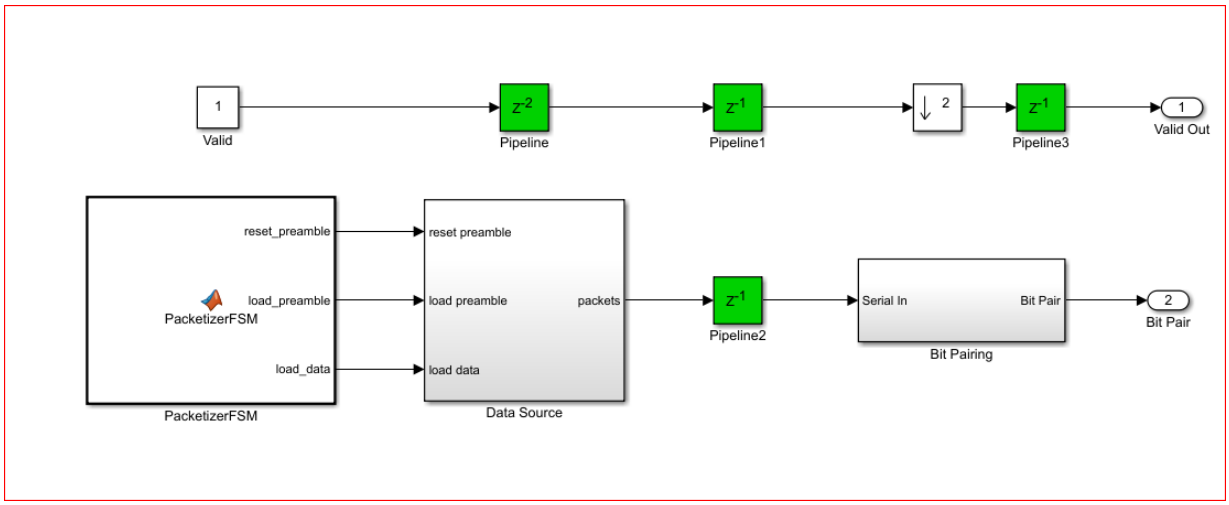


Figure 3.5: QPSK transmitter generation and packetization block diagram

A Packetizer FSM

The packetizer state machine is designed on Moore's state machine principles which consist of two states: pack preamble and load data. The pack preamble state asserts the load_preamble signal for the initialization of counter. The pack preamble state moves to the append data state after loading the 26-bit preamble. The state machine remains in the append data state for the next 174-bits. The preamble bits appended with the data bits make up a 200-bit packet ready for further processing.

Data source

Two lookup tables (LUTs) are used in the data source block: the preamble lookup and the data lookup as shown in Figure 3.6. The preamble lookup is invariant during the whole transmission because it contains the barker code header bits sequence. It resets after reaching the maximum preamble counter value (i.e. 26). The Data lookup contents vary during the whole transmission process and wrap around after achieving the last data entry. The data lookup first entry is "Hello World 000" and the number field increments to 999 before wrapping. The load data strobe from finite state machine (FSM) enables the scrambler at the start of the data bits and also

controls the 2:1 mux for routing preamble or data bits to the output. The load data strobe is high when data bits are been padded to the preamble bits. This ensures that preamble bits are not scrambled. Additive scrambler is used to randomize the

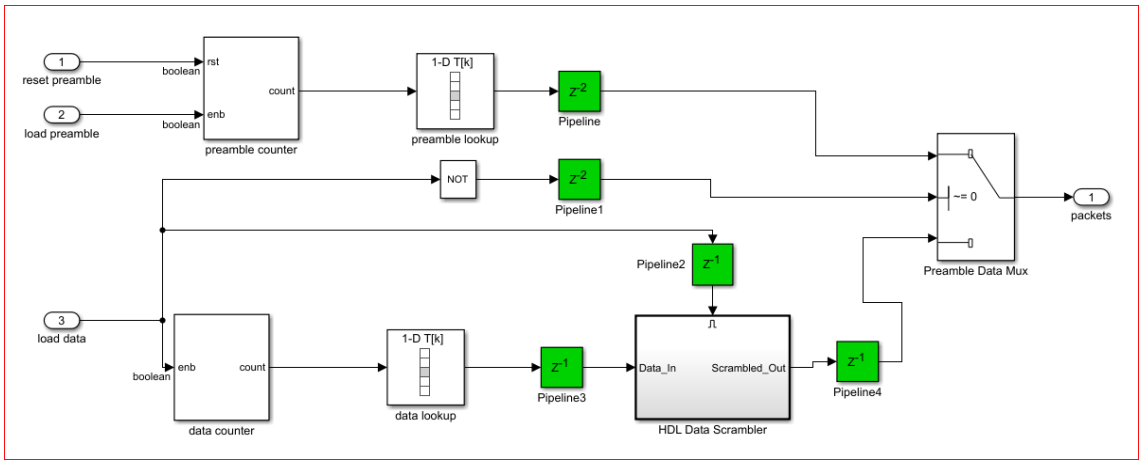


Figure 3.6: QPSK transmitter data source block diagram

data bits. It is common that long 1's or 0's are present in information bits which make timing recovery difficult at the receiver side. Additive scrambler with the help of shift registers and XOR gates randomizes the data pattern based upon the scrambling polynomial. The scrambler polynomial is represented by equation.

$$ScrambledOut = 1 + x^2 + x^3 + x^4 \tag{3.1}$$

The initial states of the shift registers are loaded with zeros. The block implementation of the scrambler is shown in Figure 3.7. The enabled system is used to ensure scrambling of new data to be done only.

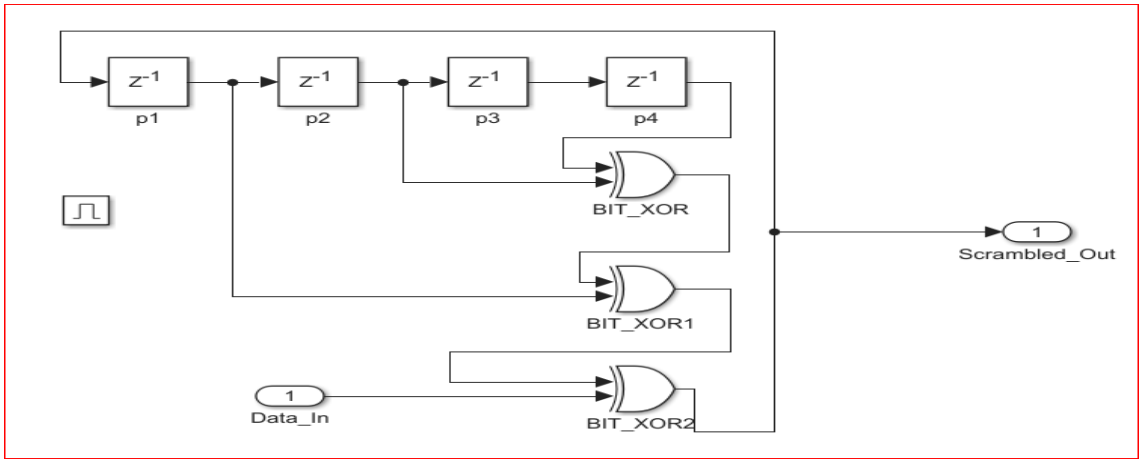


Figure 3.7: Scrambler block diagram

Bit pairing Scrambled data is packed into two unsigned bits symbols as required by the QPSK symbol mapper. The bit pairing reduces the sample rate to half (i.e. 50 bps). The bit pairing block is given in Figure 3.8 The upper downsampler selects the second phase (Quadrature phase) and the lower downsampler selects the first phase (In phase) bit. The concatenation block packs both the phase bits for further symbol mapping.

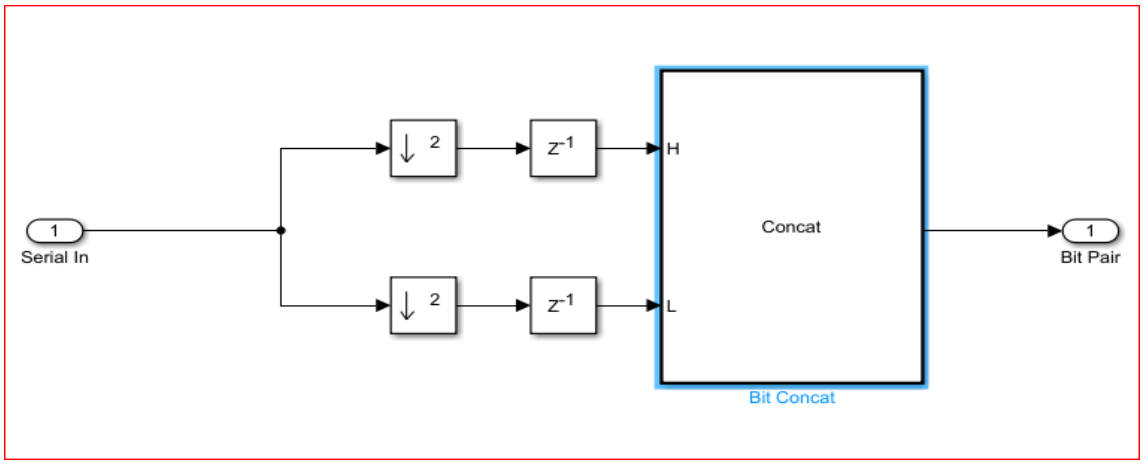


Figure 3.8: Bit pairing block diagram

A valid signal strobe is generated to control the scrambler on time and transfer valid data to the succeeding blocks. The valid data signal is also downsampled by 2 due to downsampling of data by the bit pairing block (Figure 3.5). The Pipeline registers are used inbetween data transfers for minimizing critical path delays.

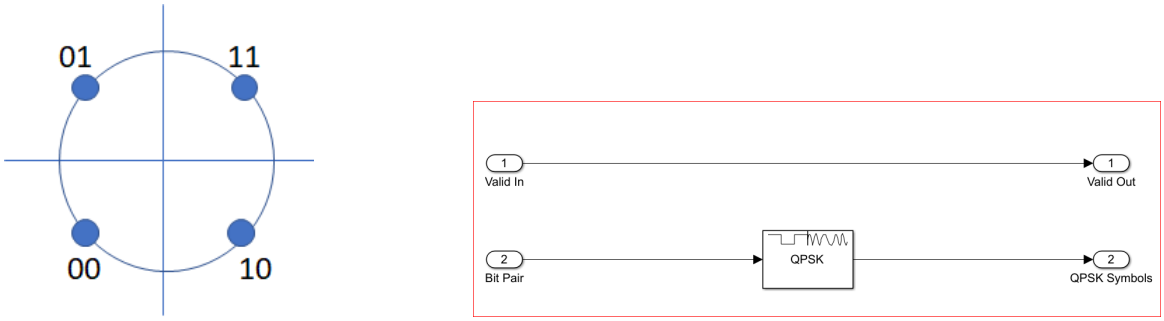
3.3.1.2 QPSK symbol mapping

The QPSK modulation technique alters the phase of the carrier to transfer information. The general formula for a PSK signal can be given as [39]

$$s_i(t) = \sqrt{\frac{2E}{T}}\cos\left(\omega_0 t + \phi_i\right) \quad 0 \leq t \leq T \quad and \quad i = 1...M \tag{3.2}$$

Where *E* represents symbol energy, T is the symbol period, ω_0 equals to $2\pi f_0$ (fundamental frequency), ϕ_i is equal to $\frac{2\pi}{M}i$ different phases and *M* equals to modulation order (4 for QPSK). The signal space (s_i) specifies the point where the incoming bits will be mapped on a constellation diagram. Figure 3.9 shows constellation points of a QPSK modulator using Gray coding. Digital data is packed into 2 bits per symbol by bit pairing block and mapped according to the constellation points by the QPSK modulator block. The block diagram depicts the QPSK modulator used from Communications toolbox^{TM} in Simulink. The incoming data is digital which have infinite

bandwidth due to square pulse characteristics. Infinite bandwidth only exists theoretically but in actual scenario spectrum/bandwidth is limited for a specific user. The spectrum confinement is done by pulse shaping filters according to the applicable standards.



(a) QPSK constellation diagram

(b) Block diagram

Figure 3.9: QPSK constellations and block diagram

3.3.1.3 Pulse shaping filters

In a digital communication system, two requirements make pulse shaping a signal necessary. The first requirement is to limit the bandwidth of the signal and the second one is to minimize ISI in bandwidth-limited channels. Both requirements are accomplished by a pulse-shaping filter. There are three types of pulse shaping filters such as sinc filters (Boxcar filters), Gaussian filters, and raised cosine filters. The raised cosine filters are widely used in reducing the ISI by attenuating the start and the end portion of a symbol. They have a maximum amplitude in the middle of the symbol period and almost zero amplitude at integer multiples of symbol frequency (1/ T_s). The major drawback of these filters is that they have an additional bandwidth requirement, which can be represented as :

$$BW = R_s(1 + \alpha) \tag{3.3}$$

Where BW specifies final bandwidth, R_s is the symbol rate and α is the roll-off factor. The roll-off factor varies from 0 to 1. To have zero ISI, the transmitter filter, channel response, and receiver filter altogether must satisfy Nyquist ISI criteria. The best which fulfills the Nyquist criteria is raised cosine filter. Due to this reason square root raised cosine filters are used at the transmitter and receiver in conjunction [40] as seen in 3.4.

$$|H_{rrc}(f)| = \sqrt{|H_{rc}(f)|} \tag{3.4}$$

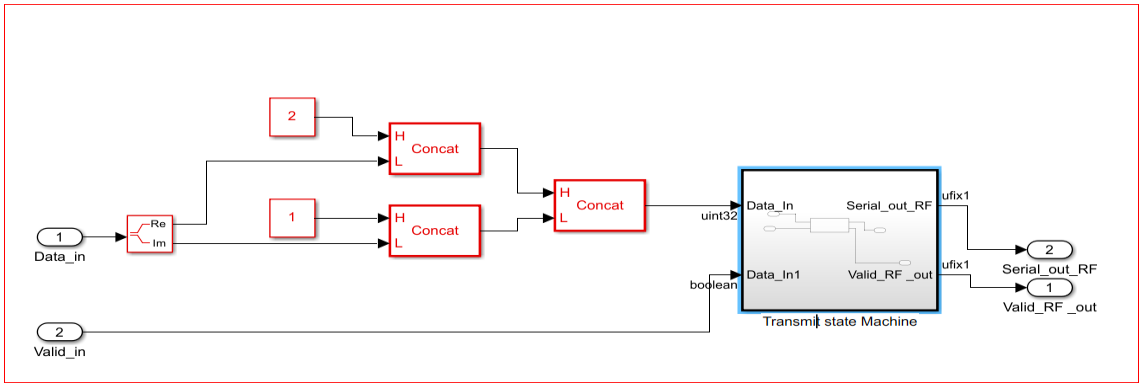
The transmit pulse shape filter is used from the Communication Toolbox library with a roll-off factor of 1 and 16 output samples per input symbol. Effective sample rate is increased to 800 (50 symbol/sec * 16) samples/sec after the pulse shaping filter.

3.3.1.4 Transmit interface formatter

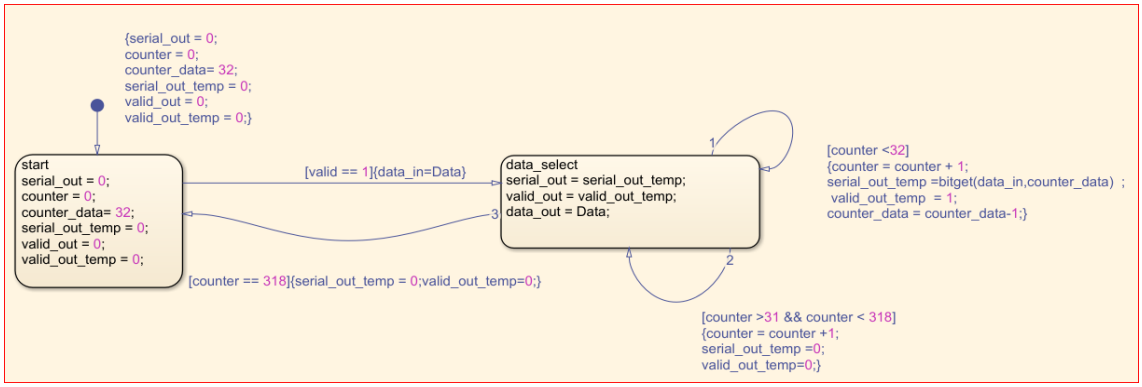
The pulse-shaped and bandlimited digital data for further transmission requires reformatting according to the requirement of the RF IC interface. The RF transceiver interface is discussed earlier in Figure 2.10. The transmitter interface formatter performs two tasks on the user data. Firstly, it appends a 2-bit sync mark to the I and Q channel data. Secondly, it increases the data rate from 11.2 kbps (I and Q channel each) to 128 Mbps by appending the required number of zeros. The bit concat block from the HDL Coder library appends the sync mark bits at the beginning of I-data and Q-data streams. After the symbol mapper subsystem. the data output type is complex (x $+\iota y$). The complex data is converted to real and imaginary components for further processing on both the chains separately. The real part is represented by I data and the imaginary part is represented by Q data as shown in Figure 3.10. This separated data is fed to the least significant bits port of the bit concate block (Figure 3.10(a)). The constant 2 in the figure represents ('1' and '0') I data sync mark to be inserted, while constant 1 represents ('0' and '1') Q data start sync mark as specified by the RF chip interface specifications. The I-data and Q-data are 14 bits signed integers but after appending the 2-bits sync mark it converts to 16-bit unsigned integers each. The I and the Q data is joined together by the bit concat block as shown in Figure 3.10 (a) to make it a 32-bit unsigned integer. The final 32-bit integer is fed to the state machine with a valid strobe.
The state machine has two states; start and data select (Figure 3.10(b). The Start state initializes the state machine with the required variable values and waits for the valid strobe to get high. When the Start state encounters valid high at the input port, the state changes to the next state (data select). The data select state has two conditional statements; one condition outputs 32-bit data serially which was received at the input port as a 32-bit word. After transferring I-data and Q-data serially, the second condition gets true which outputs 288 (9x32) zeros serially at 128 Mbps as required by the RF IC receiver interface at 400 kSamples/s.

3.3.2 QPSK receiver

The information transmitted by the sender is intercepted at the receiver antenna. The received data is corrupted by interference present in the medium. Which can be other users operating in the same frequency band. The task of the receiver is difficult than the transmitter as it has to decode the information from the received distorted



(a) Transmit interface block diagram



(b) Block Diagram

Figure 3.10: Interface formatter state machine

data. Data recovery at the receiver requires frequency and phase compensation to undo the effects of the channel. A QPSK receiver is designed using HDL optimized blocks available in the Simulink. The receiver design is adopted from MATLAB HDL optimized QPSK example [41]. The receiver block diagram can be seen in Figure 3.11. The receiver block consists of a receiver interface block, HDLRX block, Packet decoding block, and display block (BitsToASCII).

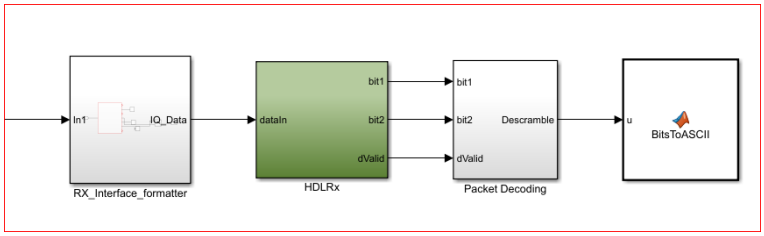


Figure 3.11: Receiver Simulink block diagram

3.3.2.1 Receiver interface

The receiver interface acquires the serial data from the RF chip and discards the unwanted zero words as per sampling requirement. It is the opposite of the transmitter interface. The RF receiver interface is implemented in the state transition diagram using Moore's state machine. In the state transition flow graph, there are five states as shown in Figure 3.12. State 1 is the default state which initializes the registers and checks for data activity. If data bit '1' is received then state is changed to State 2. Which checks for the next data bit. If the next bit is zero then it transitions to the State 3. A state transition to State 3 confirms a positive I channel sync mark reception. State 3 stores the remaining 14 data bits and waits for zero-valued bit after I data storage. If zero is received at the right instant then the state is changed to State 4 for Q sync mark verification. After the Q sync mark verification, data is latched in State 5. which marks the end of I/Q data packet and resets the whole procedure for new data packet reception.

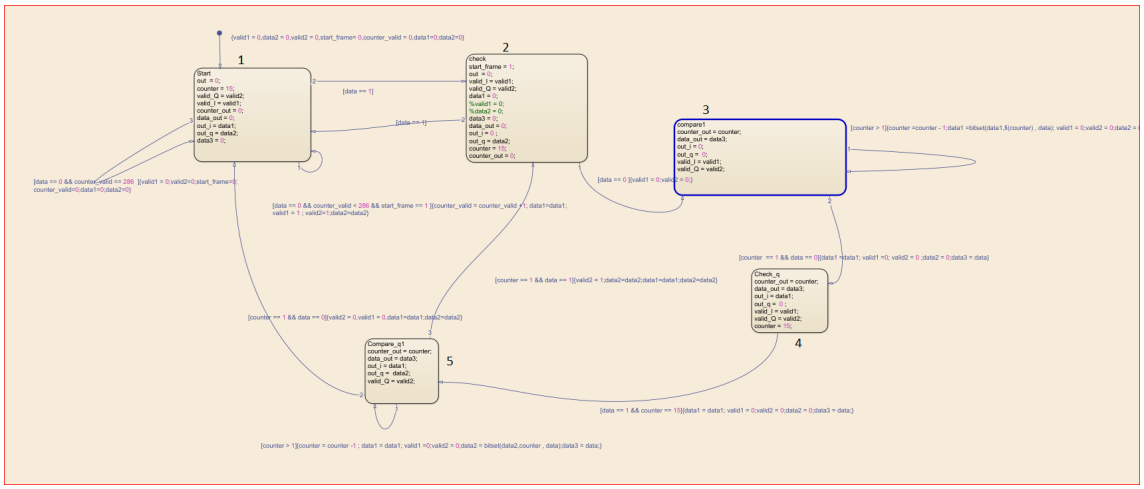


Figure 3.12: State transition diagram for RF receiver interface

After the reformatting of the data, it gets processed by the HDLRx block. The HDLRx block is the core of the receiver where baseband processing takes place. The HDLRx block diagram can be seen in Figure 3.13. The HDLRx block consists of an automatic gain control (AGC), a matched filter, a frequency compensation (coarse and fine), and a data decoding subsystem.

3.3.2.2 Automatic gain control (AGC)

An AGC block is used at the start of the receiver chain to restrict the amplitude of a signal at an optimum level for timing error detector and carrier synchronizer blocks. The timing error detector and carrier phase detector outputs depend on the

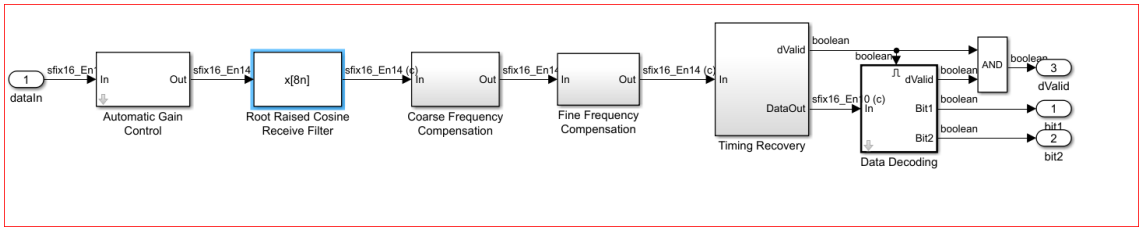


Figure 3.13: Receiver HDL optimized (HDLRx) simulink block diagram

received signal amplitude. It is convenient to fix amplitude rather than recomputing loop filter constants of phase synchronizer [17]. The AGC restricts amplitude to 1/upsamplingfactor (i.e 16) for coarse frequency compensation subsystem. The AGC block diagram is shown in Figure 3.14. The received signal modulus is computed by hardware friendly algorithm by Marvin Freking [42]. He states that modulus can be estimated by adding $|L|$ + 0.4 $|S|$, where $|L|$ represents absolute value larger among both I or Q channel, and $|S|$ represents lower value of the both. Mod value is used to compute the error signal for the feedback loop. The feedback loop error value is multiplied by the predefined loop gain value and then applied to the integrator. A gain block is used to restrict the maximum gain values to save the circuit from saturation due to high signal amplitude levels at the input. Integrator output is multiplied by the received signal to adjust the gain accordingly. The output of the AGC is a complex signal representing both I and Q channel data.

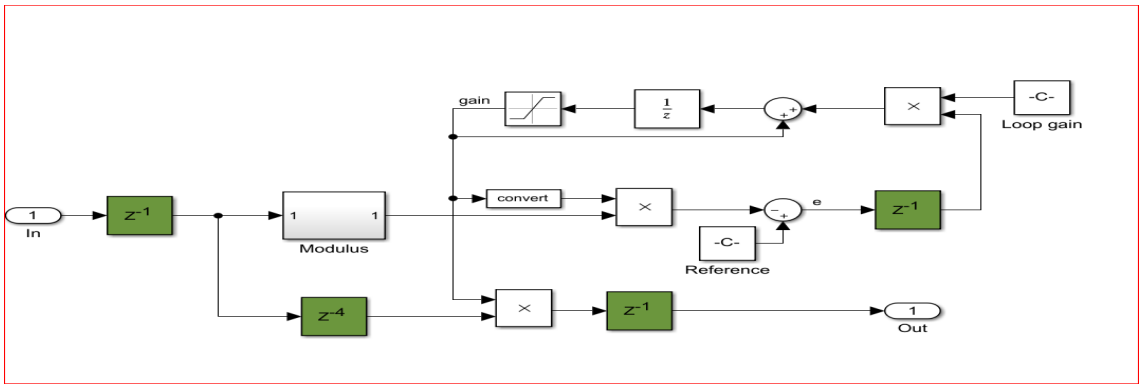


Figure 3.14: Automatic gain control block diagram

3.3.2.3 Matched filter

The goal of the digital receive filter is to extract the baseband pulse from the noisy signal and improves the SNR ratio simultaneously. An optimal solution is the matched

filter [43]. The impulse response of the matched filter can be written as

$$h(t) = \begin{cases} ks(T-t) & 0 \leq \text{t} \leq \text{T} \\ 0 & \text{elsewhere} \end{cases} \tag{3.5}$$

Where k is a constant, T is symbol duration and s(T-t) is the original signal time-reversed and shifted to the positive time axis. From 3.5, it can be deduced that match filters impulse response is maximum when it is equal to received signal (s) time-reversed and shifted else it is zero. The root raised cosine filters are commonly used at transmitter and receiver in conjunction. The match filter gives the highest amplitude pulse at the middle of the eye-opening due to maximum correlation and it has almost zero amplitude at $nT_s/2$. The matched filtered output can be seen in the Figure 3.15 with roll-off factor = 1 and samples per symbol = 16. The blue line represents filtered data while the yellow line represents the input data. It is observed that at a given noise level SNR is improved by the filtering process. This will ease data recovery processing in later stages. A matched filter can be used as a decimator for reducing data rates during further processing. Data is upsampled at the transmitter by 16 but the receiver match filter decimates data by 8 and makes 2 samples available per symbol.

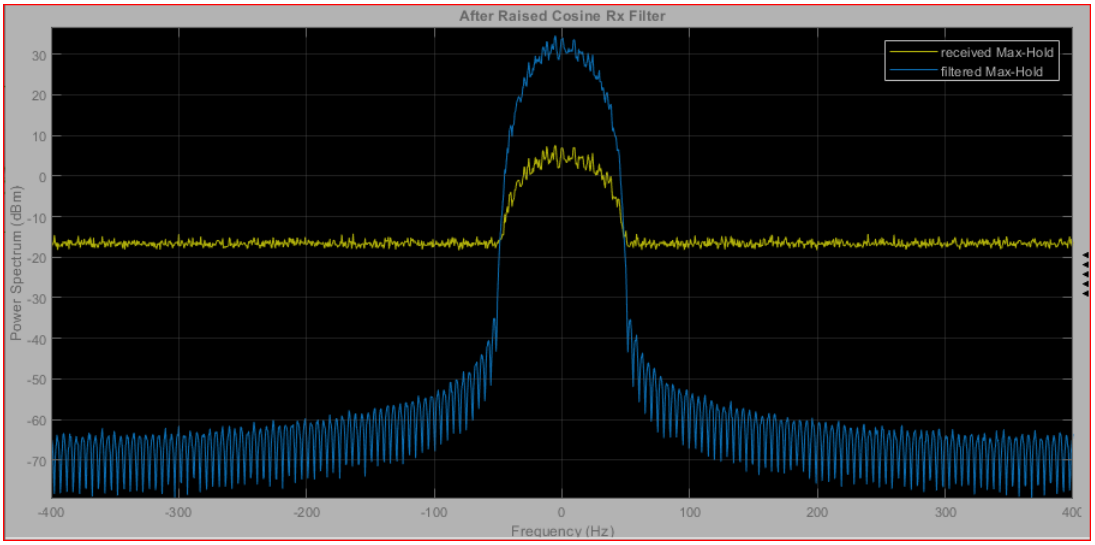


Figure 3.15: Matched filtered output

3.3.2.4 Coarse frequency compensation

The difference in the intercepted data frequency and the oscillator frequency at the receiver drifts the signal spectrum resulting in constellation rotation. If the frequency offset is greater than 10% of carrier frequency then frequency compensation is done

in two steps. First, coarse frequency compensation is done to lessen the frequency offset lower than 10%. Then residual frequency offset is corrected by fine frequency compensation block [44], [45]. If frequency impairments are not compensated besides having perfect timing synchronization, it can drastically affect the BER. The correlation-based algorithm in coarse frequency compensator block corrects input frequency offset. A residual frequency offset is still present which rotates the constellation. This residual frequency offset is corrected by a carrier synchronizer block later. The block diagram of the coarse frequency compensation subsystem is shown in Figure 3.16. The incoming complex signal is raised to power 4 for computing the fourth tone of the incoming signal. The Luise algorithm is implemented to calculate phase offset using coordinate rotation digital computer (CORDIC) algorithm. The offset phase calculated by the algorithm is fed to the complex-to-magnitude-angle HDL optimized block from the DSP toolbox library. The error magnitude is sent to the numerically controlled oscillator (NCO) for the generation of a complex sinusoidal signal at the output. This complex exponential signal is multiplied by the received signal to remove phase offset. The NCO maps lookup table into the ROM as suggested by the Luise algorithm. Residual frequency error can be observed in the constellation diagram (Figure 3.17).

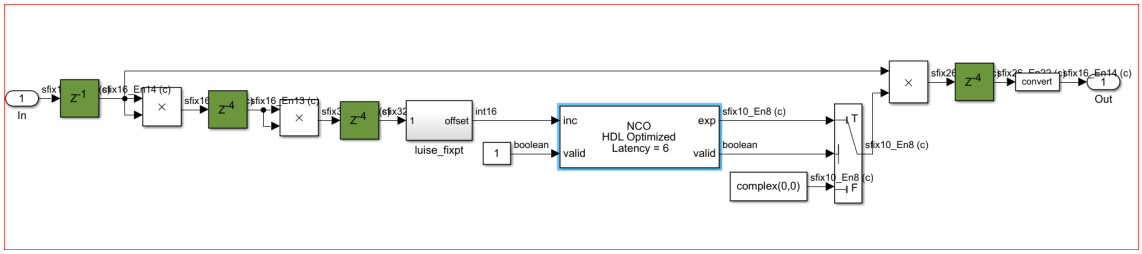


Figure 3.16: Coarse frequency compensation subsystem block diagram

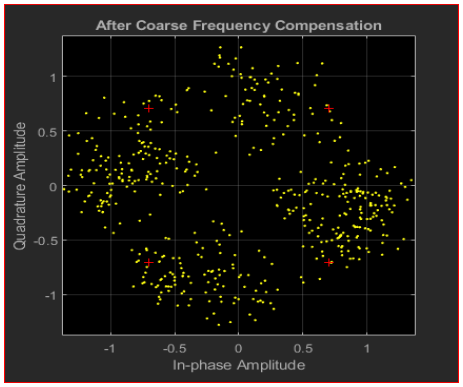
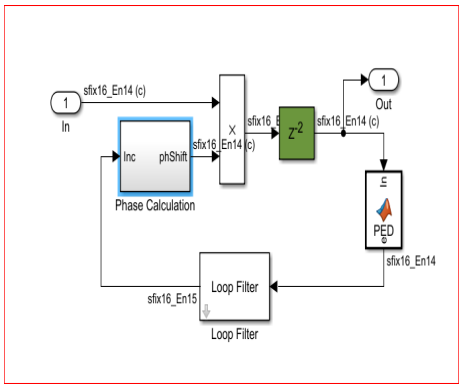


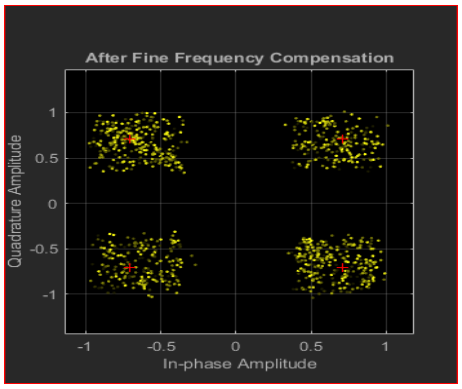
Figure 3.17: Coarse frequency compensation constellation diagram

3.3.2.5 Carrier synchronization

Carrier phase error causes the symbol to rotate on the constellation diagram and it keeps on changing regions. This region shift can cause large bit error even in the presence of symbol synchronization and no additive white Gaussian noise (AWGN). The carrier phase block at the receiver mimics the phase and frequency of the local oscillator at the transmitter. It removes the residual frequency and phase offset left after coarse carrier synchronizer. The carrier synchronizer block operates on phase-locked loop (PLL) principal, which uses a direct digital synthesizer (DDS) for estimating the phase error based on the received signal phase and nearest constellation phase [17]. The fine frequency compensation output in Figure 3.18(b) can be seen with data points at perfect constellation instances for QPSK signals. The randomness of the constellation is due to the pulse shaping filters quantization and imperfect sampling instances. The block diagram for fine frequency and phase offset subsystem is shown in Figure 3.18 (a). The input signal from the coarse frequency compensation block is fed to the maximum likelihood phase error detector (PED) block. The PED output is filtered by a loop filter for output stability. The phase calculation block computes the residual phase offset and generates a complex sinusoidal signal. The generated complex signal is used to remove the offset of the incoming signal from the coarse frequency offset block.



(a) Fine frequency compensation subsystem block diagram



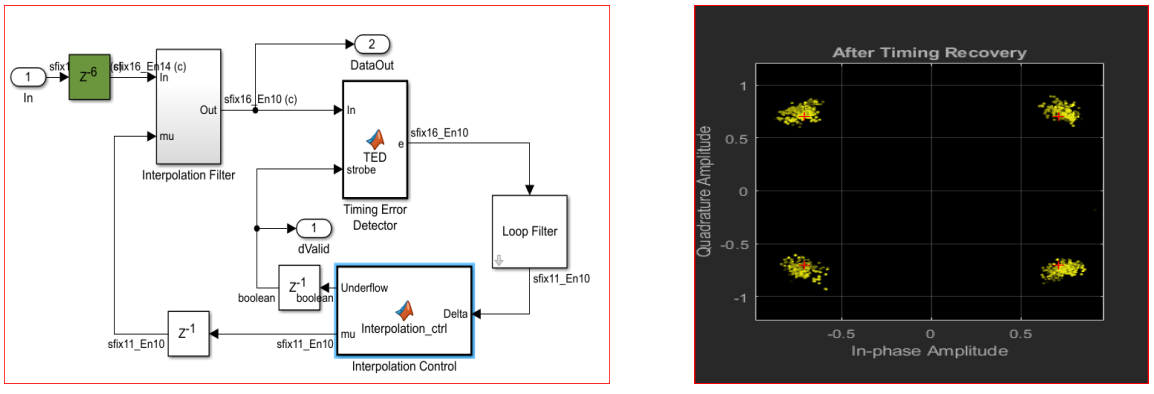
(b) Constellation diagram of QPSK fine tuned signal

Figure 3.18: Fine frequency and phase compensation

3.3.2.6 Timing recovery

To recover the transmitted signal with maximum SNR, a perfect clock sampling instant is required like at the transmitter. Due to wastage of spectrum, the data generation clock at the transmitter is never sent with the data signal. The received symbols get dispersed if they are not sampled at the instant when SNR maximum

by the receiver sampling clock. Symbol timing synchronization operates on phase locked loop (PLL) principal. Which generates an error at every sampling period to correct the clock timimng mismatch. The timing error detection (TED) can be done by decision-directed or non-data-aided methods. The decision-directed TED can be done by zero-crossing timing error detector (ZCTED) or Mueller-Mueller timing error detector (MMTED) technique. The non-data-aided TED can be done by Gardener or Early-Late method. The ZCTED method is used in the given system for computing timing error. The zero-crossing requires 2 samples/symbol at the input for estimating the error. The block diagram of timing synchronizer is given in Figure 3.19. The input samples are passed through a ZCTED to calculate the timing error. The timing error is calculated by detecting zero crossings in the eye diagram. The timing recovery system consists of four sub-blocks as shown in Figure 3.19(a) -TED, loop filter, interpolation filter control, and interpolation filter. The timing error signal is filtered before applying it to the interpolation control subsystem. The interpolation control subsystem performs two tasks. Firstly, it generates a valid signal for data decoding in the later stages. Secondly, it enables the TED at the right instant to sample the incoming data. The interpolation control subsystem updates the timing error and feeds it to the interpolation filter. The interpolation filter acts accordingly and generates an interpolated output at the required maximum eye amplitude instant for minimizing the sampling time error. Interpolation filter is farrow parabolic filter with α = 0.5 and coefficients 1, -1/2 and 3/2 as illustrated in [17] (Chapter 8). In the Figure 3.19(b), symbol synchronizer output can be seen. It can be observed that the data points lie near the constellation points. It confirms that the timing synchronizer has attained a lock to the transmitter sampling frequency.



(a) Timing recovery subsystem block diagram

(b) Constellation diagram of QPSK signal after timing recovery

Figure 3.19: Timing recovery subsystem

3.3.2.7 Data decoding

In the receiver chain, the data decoding subsystem is the last stage where operations are carried out on the received samples. The data decoding is done via phase estimation correction by correlating the barker code sequence with the input stream. The phase rotation compensation is required due to the possibilty that the carrier synchronizer block may lock at 0 ,π /2, 3 π /2 or π phase offset [17] (Chapter 7). The block diagram of the data decoding subsystem is shown in Figure 3.20. The data is transferred with a valid strobe signal to the data decoding subsystem by timing synchronizer subsystem. The data decoding subsystem performs frame synchronization by autocorrelating the received data with barker codes inserted at the transmitter. The modulus of matched filter output is compared to the threshold. If the output is equal to or greater than the threshold, an enable signal turns high. Meanwhile, the conjugate of the received signal is calculated and halted at the enabled high signal. The halted conjugate signal phase tells about phase rotation if any was present in the received signal. If phase rotation was observed then derotation is done by multiplying the conjugate value with all the received frame bits. After phase derotation data is demodulated using a QPSK demodulator from the Communications Toolbox HDL support.
The QPSK demodulator maps received complex signals to bits for information recovery to be done in the packet decoding subsystem.

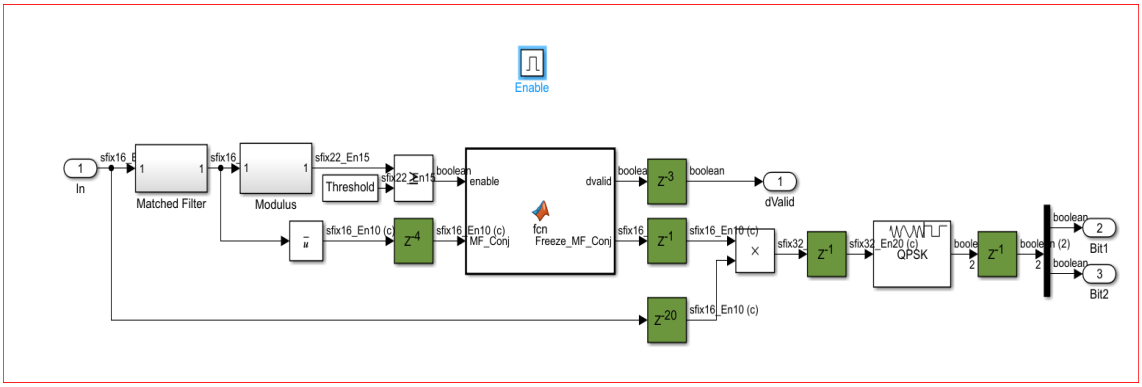


Figure 3.20: Data decoding subsystem block diagram

3.3.2.8 Packet decoding subsytem

In Subsection 3.3.2.7, it is stated that the data processing is sample-based, but for complete information reception, the whole packet should be stored, processed and decoded. The packet decoding subsystem stores the data in 200-bits packet (26bits barker code and 174-bit data payload). This buffered data is decoded after ripping off the preamble header (26-bits) from the packet length. The block diagram

for the packet decoding subsystem is given in Figure 3.21. The deframer MATLAB function block stores the incoming bits and packs them into a 200-bits packet. The packet is transferred to the descrambler block with a valid data (framed_data) strobe. The descrambler block from Communication Toolbox is used to descramble the data bits. It can be observed in Figure 3.21 that the descrambler is placed inside an enabled subsystem, which turns on when the valid strobe is high. A high valid strobe means data of 200-bits is ready to be fetched from deframer output. The descrambler block diagram in Figure 3.22 shows a selector block from the HDL coder library which selects the required number of bits from the packet stored. The selector block reformats or discards bits as per requirement. The remaining 174-bits (data bits) are descrambled by the descrambler block. The descrambler polynomial is the same as the transmitter scrambler polynomial as stated earlier.

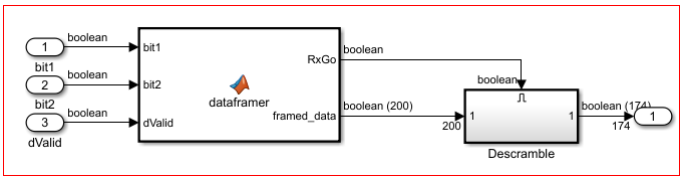


Figure 3.21: Packet decoder subsystem block diagram

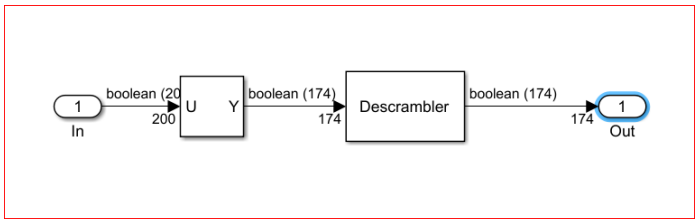


Figure 3.22: Descramble subsystem block diagram

Descrambled data is displayed on MATLAB diagnosis window by BitstoASCII block (Figure 3.11).

3.3.3 Simulation Verification, FPGA synthesis, and power report

3.3.3.1 Simulink Verification

The QPSK transmitter and receiver are interconnected as shown in Figure 3.3 for validating systems performance. The Bit-to-Ascii MATLAB function block was used to fetch the descrambled data and then pack it into 7 bits. The ASCII represented by 7 bits is then displayed on the MATLAB's diagnostic window. The output of the diagnostic window can be seen in Figure 3.23 for characters (Hello World 000-999) sent from the transmitter which were accurately received without errors. Moreover, it

also calculates BER by comparing the bits received with the transmit pattern known locally. The BER plots are given in the Chapter 5 for the whole integrated system.

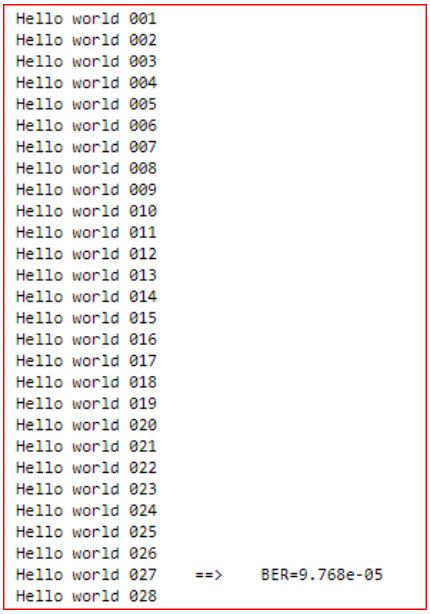


Figure 3.23: Diagnostic window output

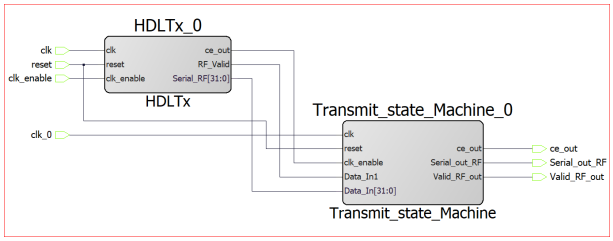
3.3.3.2 Hardware synthesis report

The HDL code was generated after simulation verification to have an estimate of the FPGA resources needed in the transceiver system design. Based on the resources required, conclusion can be drawn in light of the thesis scope. The HDL code generated from Simulink HDL coder was synthesized using Libero Soc v12.2 software.

Transmitter FPGA resources report

Figure 3.24(a) displays a smart design implemented in Libero SoC of a QPSK transmitter. The HDL_Tx depicts the transmission part which contains data generation and packetization subsystem, symbol mapping subsystem, pulse shaping subsystem and concatenation of I-data and Q-data in transmit interface block. The Transmit_state_Machine depicts the state machine as mentioned in Figure 3.10(a). The resource utilization report in Figure 3.24(b) displays the number of resources required to implement a transmitter on an IGLOO2-M2GL005 FPGA. The transmitter requires a minimum of 2566 look-up table (LUT) and a 2097 D-flip flop (DFF) FPGA. In the designed system, data was generated by data generation and packetization subsystem as a packet of 200-bits. Due to packet generation, data was initialized as a primitive in the data source submodule HDL implementation which utilized 1497

LUTs and 42 DFFs. If the data comes from any external serial source then resource requirements will be lesser than what was displayed in the synthesis report. Approximately 1069 (2566 − 1497) 4LUTs and 2055 (2097 − 42) DFFs are required when an external serial data source is present. A total of 7 I/O ports are required for communication with the external modules. 4 ports are used as an input (CLK (800 Hz), Clk_0 (128Mhz), reset and clk_enable) pads and the remaining 3 ports are used as an output (chip enable out (ce_out), serial-out data (Serial_Out_RF), and valid strobe (debug only)) pads. As discussed earlier, the data generation source is internal in the transmitter designed above, but if a data source is external then input pad requirement will increase based on the interface connections between the sensor and the FPGA.



(a) QPSK transmitter RTL diagram

Module Name	Fabric 4LUT	Fabric DFF	Single-Ended I/O	Chip Globals
Top	2566	2097	7	2
Primitives	0	0	7	2
HDLTx_0	2442	2027	0	0
Primitives	2	478	0	0
u_Data_Generation_Pa...	1528	67	0	0
Primitives	2	6	0	0
u_Bit_Pairing	0	6	0	0
u_Data_Source	1500	46	0	0
Primitives	1497	42	0	0
u_HDL_Data_S...	3	4	0	0
u_PacketizerFSM	26	9	0	0
u_HDLTx_tc	15	10	0	0
u_Pulse_Shaping	897	1472	0	0
Transmit_state_Machine_0	124	70	0	0

(b) QPSK transmitter FPGA synthesis report

Figure 3.24: QPSK transmitter HDL based design

Receiver FPGA resources report

The receiver plays a vital role in the whole communication chain because it has to deal with the noisy channel. Extracting information from the received signals requires some additional steps to be done as discussed earlier. These essential tasks make receiver circuitry heftier in terms of area requirements. The register transfer level (RTL) schematic in Figure 3.25 shows the HDL blocks of the receiver which resembles the Simulink block diagram as discussed earlier. The intercon-

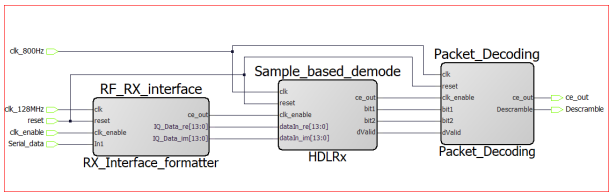


Figure 3.25: Receiver circuitry RTL schematic

nections of the three modules displayed in the RTL schematic are according to the Simulink block diagram. After the interconnection, the whole system was synthesized in Libero SoC v12.2 for M2GL-025 FPGA due to limited resources on M2GL-

005 FPGA. The synthesis report (Figure 3.26) shows that approximately 24k 4LUTs and 14k DFFs are required for the receiver implementation on an FPGA. Moreover, it also requires 34 Multiply Accumulate (MAC) blocks for multiplications. Data transfer in and out of an FPGA requires 9 I/O pads (5 inputs and 4 outputs). The output ports can vary depending on the sensor node interface and functionality in response to data received. Functionality can involve the collection of data from an external sensor or execute instructions accordingly.

Resource Usage

Type	Used	Total	Percentage
4LUT	23779	27696	85.86
DFF	14102	27696	50.92
I/O Register	0	621	0.00
User I/O	9	207	4.35
-- Single-ended I/O	9	207	4.35
-- Differential I/O Pairs	0	103	0.00
RAM64x18	0	34	0.00
RAM1K18	0	31	0.00
MACC	34	34	100.00
Chip Globals	3	16	18.75
CCC	0	6	0.00
RCOSC_25_50MHZ	0	1	0.00
RCOSC_1MHZ	0	1	0.00
XTLOSC	0	1	0.00
HPMS	0	1	0.00

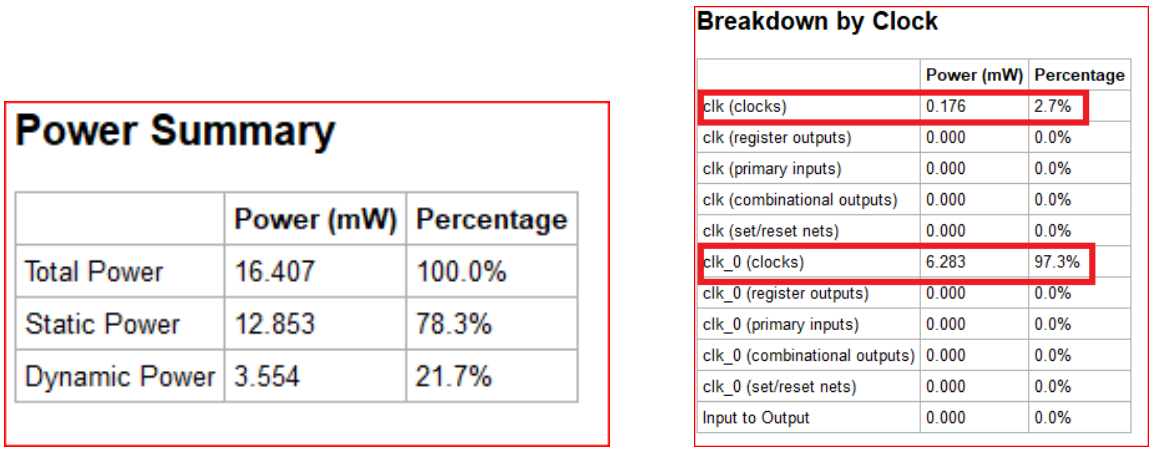
Figure 3.26: Receiver synthesis report

3.3.3.3 Power utilization

In a power limited system, energy source is the most important part of a WSN nodes. The power consumption analysis report was generated by the SmartPower tool available as a widget in Libero SoC v12.2.

Transmitter power consumption report

Power consumption for transmitter can be seen in Figure 3.27(a) which is approximately 17 mW (12.8 mW static and 3.55 mW dynamic). Static power is calculated when there is no circuit activity and dynamic power is computed when the circuit is operational. In Figure 3.27(b) both clocks of 800 Hz (CLK) and 128 MHz (clk_0) power usage percentage is displayed. It is observed that 97% power is consumed in 128 MHz clock circuitry switching.



Power Summary

	Power (mW)	Percentage
Total Power	16.407	100.0%
Static Power	12.853	78.3%
Dynamic Power	3.554	21.7%

(a) QPSK transmitter power estimates

Breakdown by Clock

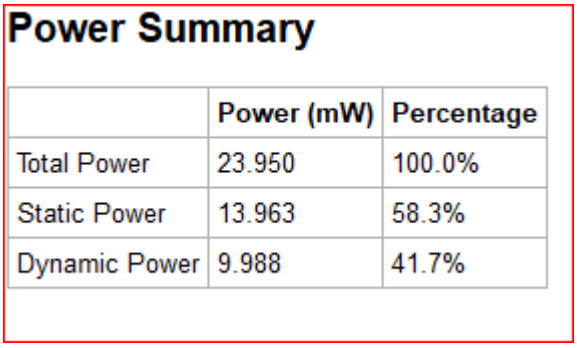
	Power (mW)	Percentage
clk (clocks)	0.176	2.7%
clk (register outputs)	0.000	0.0%
clk (primary inputs)	0.000	0.0%
clk (combinational outputs)	0.000	0.0%
clk (set/reset nets)	0.000	0.0%
clk_0 (clocks)	6.283	97.3%
clk_0 (register outputs)	0.000	0.0%
clk_0 (primary inputs)	0.000	0.0%
clk_0 (combinational outputs)	0.000	0.0%
clk_0 (set/reset nets)	0.000	0.0%
Input to Output	0.000	0.0%

(b) QPSK transmitter FPGA synthesis report

Figure 3.27: QPSK transmitter power analysis and synthesis

Receiver power consumption report

The receiver circuitry needs more power than the transmitter system due to higher logic resources requirements. The power consumption report (Figure 3.28) shows that a receiver requires approximately 3 times more dynamic power than a transmitter. This difference will increase at faster data rates.



Power Summary

	Power (mW)	Percentage
Total Power	23.950	100.0%
Static Power	13.963	58.3%
Dynamic Power	9.988	41.7%

Figure 3.28: Receiver power consumption report

3.4 Three-chip based solution design

In Chapter 2, a hybrid solution was discussed as an option. A hybrid solution can consist of multiple digital processing devices like a microcontroller, an FPGA, and an SoC. In a three-chip solution, the microcontroller will act as a baseband signal processor while the FPGA performs upsampling or downsampling. The FPGA at the transmitter interpolates data received from the pulse shaping filter by 500 to make

400 kSamples/s, as the minimum requirement of the RF chip. The receiver FPGA decimates the data by a factor of 500 to make it 800 bps (Figure 3.29).

3.4.1 FPGA filter and interface design

The block diagram in Figure 3.29 displays the FPGA connections with the RF IC and the microcontroller. The microcontroller acts as a data source at the transmitter side and as a data sink in the reception chain. The data is generated at a rate of 100 bps, scrambled, QPSK modulated, and pulse shaped by the microcontroller. The pulse shaped data is transferred to FPGA via the serial interface at 11.2 kbps (800 x 14 bps). After interpolation at the FPGA, the data is formatted according to the RF IC serial interface requirements. The sample rate after interpolation is increased by an interpolation factor of 500 (400 kSamples/s). The RF IC has a fixed sampling rate of 4 MSamples/s with I-data and Q-data 16 bits each. If the information data sample rate is less than 4 Msps then zero padding is done according to Figure 2.10. The serial data rate of RF IC after zero padding converts to 128 Mbps. In the reception, all the steps are reversed. First, the data is received by the FPGA from the RF IC interface serially at 128 Mbps. After reception required I-data and Q-data bits are extracted from the received stream. The I and Q data is decimated by 500 and transferred to the microcontroller over the serial interface at 11.2 kbps (800x14 bps) data rate. A microcontroller performs the rest of the demodulation and decoding part. In the given LPWAN scenario, filters plays an important role in improving the SNR of the received signal and it is used as a sample rate converter too. The low data rates necessitates the use of sample rate converters. The RF IC sampling rate is fixed to 4Msps and each sample is of 32 bits. It can support a minimum sampling rate of 400 ksps with the help of zero padding. The LPWAN node maximum allowable bandwidth is 600 kHz at the sub-1Ghz band in Europe as per ECC standards.

3.4.1.1 Filter design

An ultra-wideband signal is a reality due to fast data converters and complex digital hardware that can operate on a GHz scale. The extraction of narrowband signals from the wideband signals is now a requirement of low data rate systems. This extraction requires narrowband filters and sample rate converters. The sample rate conversion can not be done by simply discarding or padding samples due to the aliasing effect. A low pass filter is required to limit the frequency spectrum of resampled signal and attenuation of high-frequency aliasing interference. A low-pass filter can be implemented in the microcontroller, GPP, DSP, or FPGA. In [46] the authors recommended implementing filters on an FPGA to be a better option due

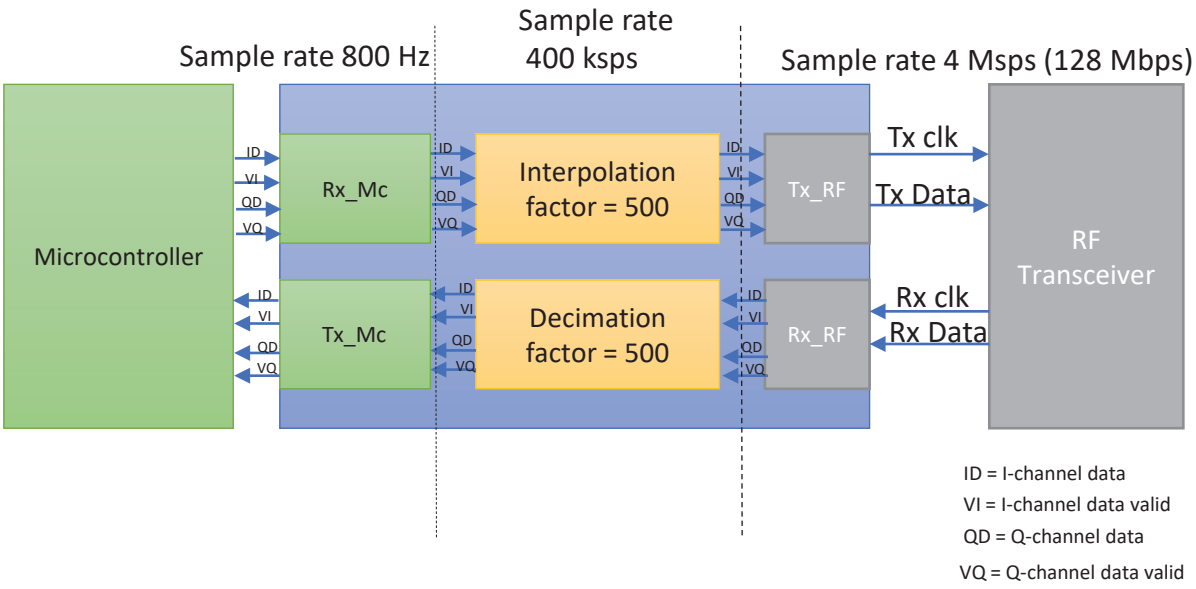


Figure 3.29: FPGA implementation block diagram

to the inherent parallel processing nature. Parallel processing is advantageous in filter processing due to concurrent multiplications and addition. There are two basic types of filters; finite impulse response (FIR) filters and infinite impulse response (IIR) filters. The filter general transfer function is expressed as follows [39].

$$H(z) = \frac{b_0 + b_1 z^{-1} + ... + b_n z^{-n}}{1 + a_1 z^{-1} + ... + a_M z^{-M}} \tag{3.6}$$

Digital filters with M $\geq$ 0 ($a_M \neq 0$) are called IIR filters. The denominator of the transfer function shows poles of the filters and in the time domain these poles are modeled as feedback delay elements [17]. The current output depends on the previous input values with feedback that makes impulse response infinite. The IIR filters have an advantage in terms of lesser hardware area requirements as compared to the FIR filters. The disadvantage of an IIR filter is the nonlinear group delay. The nonlinear group delay alters the phase of passband frequency components differently. The Filters with M = 0 and a_1 = 0 are called as FIR filters.

3.4.1.2 FIR filter

The FIR filter is a special case of an IIR filter. If a sequence x(n) is applied to the FIR filter then output sample y(n) is given as.

$$y(n) = \sum_{n=0}^{M} h(k)x(n-k) \tag{3.7}$$

y(n) is the convolution of the input sequence with the filter impulse response. In a multi-rate system, sometimes large rate changes are required which results in the

need of fast multipliers and very long filters length of the FIR filter implementation [47]. The most popular filter implementations are shown in Figure 3.30 and 3.31.

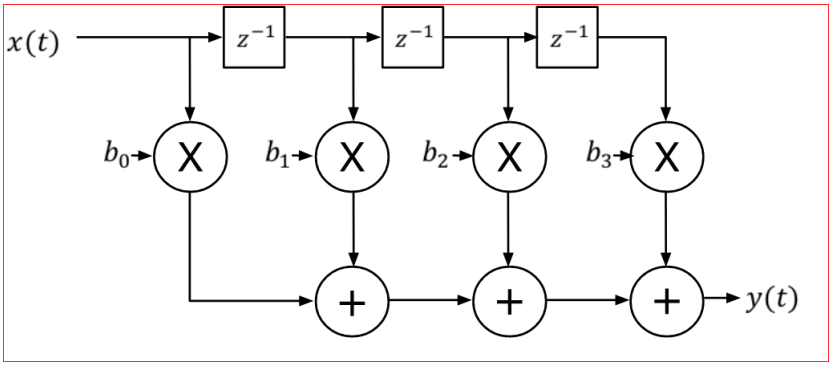


Figure 3.30: Tapped Delay line filter [48]

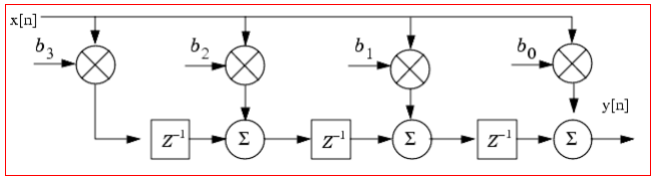


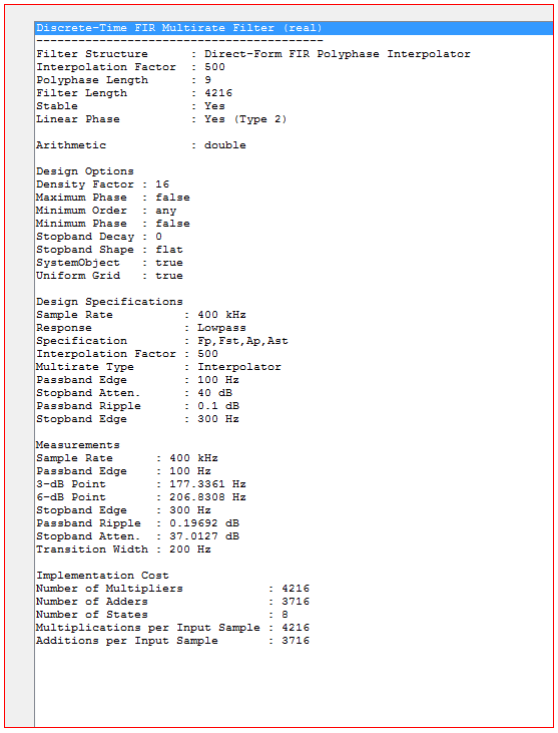
Figure 3.31: Partial sum accumulator [17]

In Figure 3.30, delay elements are seen in the input signal path. Due to this structure this type of FIR filters is called tapped-delay-line filters. A two-input adder is connected to the coefficient multiplier, which in hardware language is known as Multiply-Accumulate (MAC) block. The filter in Figure 3.31 is known as the transposed form of FIR filter. In this structure, delay elements are connected to the adders, and input is directly multiplied by the filter coefficients. This type of structure is also known as a partial sum accumulator [17]. The FIR multi-rate filters are used in hardware design, but due to massive hardware requirements make them an unfavorable choice. The hardware requirements can be relaxed by the half-band filters or the polyphase filter implementation [49]. The half-band filters can work on a rate change factor of 2's power only, while polyphase filters still require a large number of multipliers. Figure 3.32(a) shows an extensive amount of FPGA resources required for the FIR polyphase interpolater filter based on parameters specified in the Table 3.2. It is not practical to implement such a large amount of multipliers on any hardware. A multi-stage filter design can be adopted to reduce hardware resources utilization, but from Figure 3.32 (b) it can be observed that still for the same implementation, 116 multipliers are needed.

The FIR filter can be implemented in multistages to reduce hardware resource utilization but large amount of multipliers are still required. The synthesis report

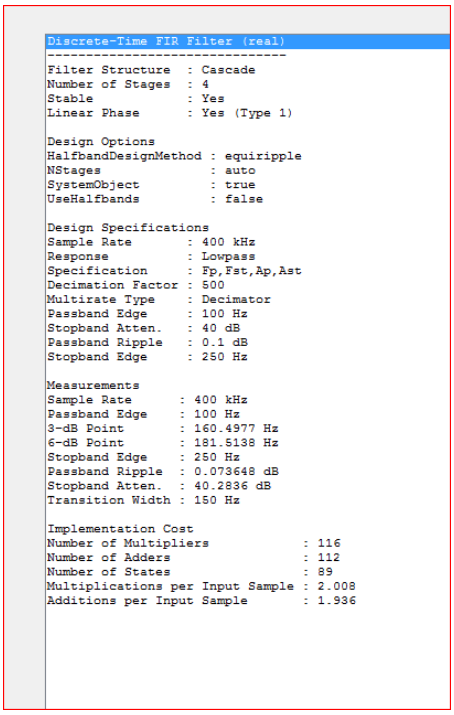
Table 3.2: Parameters for filter design

pass band frequency	100 Hz
stopband frequency	250 Hz
pass band ripple	0.1 dB
stopband attenuation	60 dB
interpolation/decimation factor	500



```
Discrete-Time FIR Multirate Filter (real)
-----------------------------------------
Filter Structure      : Direct-Form FIR Polyphase Interpolator
Interpolation Factor  : 500
Polyphase Length      : 9
Filter Length         : 4216
Stable                : Yes
Linear Phase          : Yes (Type 2)

Arithmetic            : double

Design Options
Density Factor : 16
Maximum Phase  : false
Minimum Order  : any
Minimum Phase  : false
Stopband Decay : 0
Stopband Shape : flat
SystemObject   : true
Uniform Grid   : true

Design Specifications
Sample Rate           : 400 kHz
Response              : Lowpass
Specification         : Fp,Fst,Ap,Ast
Interpolation Factor : 500
Multirate Type        : Interpolator
Passband Edge         : 100 Hz
Stopband Atten.       : 40 dB
Passband Ripple       : 0.1 dB
Stopband Edge         : 300 Hz

Measurements
Sample Rate      : 400 kHz
Passband Edge    : 100 Hz
3-dB Point       : 177.3361 Hz
6-dB Point       : 206.8308 Hz
Stopband Edge    : 300 Hz
Passband Ripple  : 0.19692 dB
Stopband Atten.  : 37.0127 dB
Transition Width : 200 Hz

Implementation Cost
Number of Multipliers            : 4216
Number of Adders                 : 3716
Number of States                 : 8
Multiplications per Input Sample : 4216
Additions per Input Sample       : 3716
```

(a) MATLAB resources estimation for FIR polyphase interpolator



```
Discrete-Time FIR Filter (real)
-------------------------------
Filter Structure  : Cascade
Number of Stages  : 4
Stable            : Yes
Linear Phase      : Yes (Type 1)

Design Options
HalfbandDesignMethod : equiripple
NStages              : auto
SystemObject         : true
UseHalfbands         : false

Design Specifications
Sample Rate       : 400 kHz
Response          : Lowpass
Specification     : Fp,Fst,Ap,Ast
Decimation Factor : 500
Multirate Type    : Decimator
Passband Edge     : 100 Hz
Stopband Atten.   : 40 dB
Passband Ripple   : 0.1 dB
Stopband Edge     : 250 Hz

Measurements
Sample Rate       : 400 kHz
Passband Edge     : 100 Hz
3-dB Point        : 160.4977 Hz
6-dB Point        : 181.5138 Hz
Stopband Edge     : 250 Hz
Passband Ripple   : 0.073648 dB
Stopband Atten.   : 40.2836 dB
Transition Width : 150 Hz

Implementation Cost
Number of Multipliers            : 116
Number of Adders                 : 112
Number of States                 : 89
Multiplications per Input Sample : 2.008
Additions per Input Sample       : 1.936
```

(b) Multistage implementation of FIR direct-form filter

Figure 3.32: MATLAB filter resources estimates (Interpolation factor = 500)

for resource estimation is shown in Figure 3.33. It can be observed that the FIR filter synthesized using integrated MAC blocks requires lesser FPGA resources but a large amount of MAC blocks(34). If synthesis process is restricted to the available logic resources use only then total count jumps to 8730 from 2441 (4LUT). Power consumption for both cases is almost the same 22.4 mW (15.9 mW static and 6.5 mw dynamic) as given in Figure 3.34 .

3.4.1.3 Cascaded integrator-comb (CIC) filters

The cascaded integrator-comb (CIC) filters were proposed by the Hogenaur [50] in 1981. Since then, they have become a popular choice when large sample rate

Resource Usage

Type	Used	Total	Percentage
4LUT	8730	27696	31.52
DFF	640	27696	2.31
I/O Register	0	621	0.00
User I/O	35	207	16.91
-- Single-ended I/O	35	207	16.91
-- Differential I/O Pairs	0	103	0.00
RAM64x18	0	34	0.00
RAM1K18	0	31	0.00
MACC	0	34	0.00
Chip Globals	3	16	18.75
CCC	0	6	0.00
RCOSC_25_50MHZ	0	1	0.00
RCOSC_1MHZ	0	1	0.00
XTLOSC	0	1	0.00
HPMS	0	1	0.00

(a) Synthesis report without MAC block

Resource Usage

Type	Used	Total	Percentage
4LUT	2441	27696	8.81
DFF	1782	27696	6.43
I/O Register	0	621	0.00
User I/O	35	207	16.91
-- Single-ended I/O	35	207	16.91
-- Differential I/O Pairs	0	103	0.00
RAM64x18	0	34	0.00
RAM1K18	0	31	0.00
MACC	34	34	100.00
Chip Globals	3	16	18.75
CCC	0	6	0.00
RCOSC_25_50MHZ	0	1	0.00
RCOSC_1MHZ	0	1	0.00
XTLOSC	0	1	0.00
HPMS	0	1	0.00

(b) Synthesis report with MAC block

Figure 3.33: Multistage FIR filter synthesis on M2GL025 (interpolation factor = 500)

Power Summary

	Power (mW)	Percentage
Total Power	22.484	100.0%
Static Power	15.933	70.9%
Dynamic Power	6.551	29.1%

Figure 3.34: Power consumption for IGLOO2 M2GL-025 FPGA with MAC blocks

conversion is needed. They have a multiplierless structure which consist of adders and delay elements only. A commonly used decimation filter is recursive running sum (RRS) filter with transfer function [51] as below:

$$H(z) = \left[\frac{1}{R}\left(\frac{1-z^{-R}}{1-z^{-1}}\right)\right]^{N} \tag{3.8}$$

Where R is the decimation ratio, and N is the number of filter stages. An efficient realisation of RRS requires equal number of comb and integrator stages separated by sample rate changing switch [50] . A CIC filter basic block diagram can be seen in Figure 3.35. Where C represent comb stages with a transfer function given as :

$$H_C(z) = 1 - z^{-RM} \tag{3.9}$$

The comb sections operates at low sampling rate f_s/R, where R is the required rate change factor. M is a differential delay which is mostly 1 or 2 [50]. The blue box represents an integrator block that operates at the sampling frequency f_s and act as

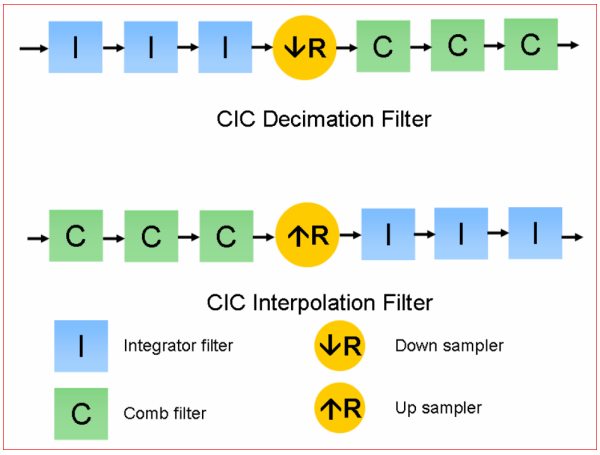


Figure 3.35: CIC decimator and integrator block diagram [52]

a single pole filter with a unity feedback. The system function of an intergrator is represented as:

$$H_I(z) = \frac{1}{1 - z^{-1}} \tag{3.10}$$

The combined CIC filter system function for N stages can be obtained by multiplying the 3.9 with 3.10 as the linear time invariant systems (LTI) property holds. The transfer function is as follows.

$$H(z) = H_I^N(z) H_c^N(z) = \frac{\left(1 - z^{-RM}\right)^N}{\left(1 - z^{-1}\right)^N} = \left[\sum_{k=0}^{RM-1} z^{-k}\right]^N \tag{3.11}$$

It can be observed from 3.11 that a CIC filter frequency response is equivalent to N cascaded FIR filters. The magnitude response of the filter can be given as

$$|H(f)| = \left|\frac{\sin \pi M f}{\sin \frac{\pi f}{R}}\right|^{2N} \tag{3.12}$$

After doing little bit of manipulation and using approximation sinx $\approx$ x (for small values of x) magnitude response can be estimated as

$$|H(f)| = \left|RM \frac{\sin \pi M f}{\pi f M}\right|^{2N} \qquad 0 \leq f < 1/M \tag{3.13}$$

This approximation helps in determining the nulls of the power spectrum, which are located at reciprocal of differential delay value (M). Another observation about pass-band droop is the dependency on filter stages. Larger the number of the stages, higher will be the passband droop. The passband droop can be compensated by reducing stages but then it will increase side lobe levels as an artifact. The Filter stages (N), rate change factor (R) and differential delay (M) are the only tuneable parameters for obtaining required frequency response. The gain of the CIC decimation filter is calculated as

$$G = \left(RM\right)^N \tag{3.14}$$

The gain in 3.14 helps in calculating the total number of output bits required for registers in comb and integrator stages of a decimator filter.

$$B_{out} = \lceil N \log_2 RM + B_{in} \rceil \tag{3.15}$$

B_{out} is the number of bits at the output of the filter without pruning. B_{in} represents number of input bits of the filter. For the given system, output bits are calculated to be 50 bits (4*$\log_2$ 500 + 14). These 50 bits can be reduced to required output bits by pruning as discussed by Hogenaur [50].

Compensation filter

The magnitude droop can be compensated by applying an FIR filter which has a frequency response equals to the inverse of the CIC filter frequency response [52]. A widely known compensation filter is the inverse sinc filter with a magnitude response as

$$|H(z)| = \left| MR \frac{\sin\left(\pi f/R\right)}{\sin\left(\pi M f\right)} \right| \tag{3.16}$$

When the rate change factor (R) is large, then the magnitude can be approximated to an inverse sinc function. That is why the FIR compensation filter is known as an inverse sinc filter too.

3.5 Simulation and results

The CIC filters were implemented and tested in the Simulink (MATLAB). The parameters for analyzing filter are the same as in Table 3.2. A 4-stage CIC interpolator and decimator was selected based upon the design procedure given in [50].

3.5.1 Interpolation

An interpolator is used to upsample a signal by inserting zeros in between the samples. The number of zeroes depends on the rate factor R. If R is 500 then 499 (R-1) zeroes are inserted between consecutive samples. An interpolator was first implemented in the MATLAB script for comparing it with the Simulink model. The filter response is shown in Figure 3.36. It can be seen from the filter response that the interpolator circuit alone does not qualify for the parameters selected. The sidelobe level is higher than 60 dB with 1 dB passband ripple. If we zoom the frequency response to passband frequency, we can see passband droop of the CIC interpolator filter (blue line) (Figure 3.37). This droop is adjusted by the compensation filter when both are cascaded together. The cascaded CIC-compensation filter

frequency response (red line) has a passband ripple of 0.1 dB while the CIC interpolater alone passband ripple was approximately 1 dB. The quantized filter response is also shown in the figure which specifies that how much a filter differs in frequency response when its arithmetic type is changed from double to fixed-point. The filter coefficients quantization error is negligible that's why the plot shows two lines (blue and red solid lines). Fixed-point filter designs are preferred due to limited hardware resources. The filter frequency graphs are plotted using the MATLAB FVTool. The

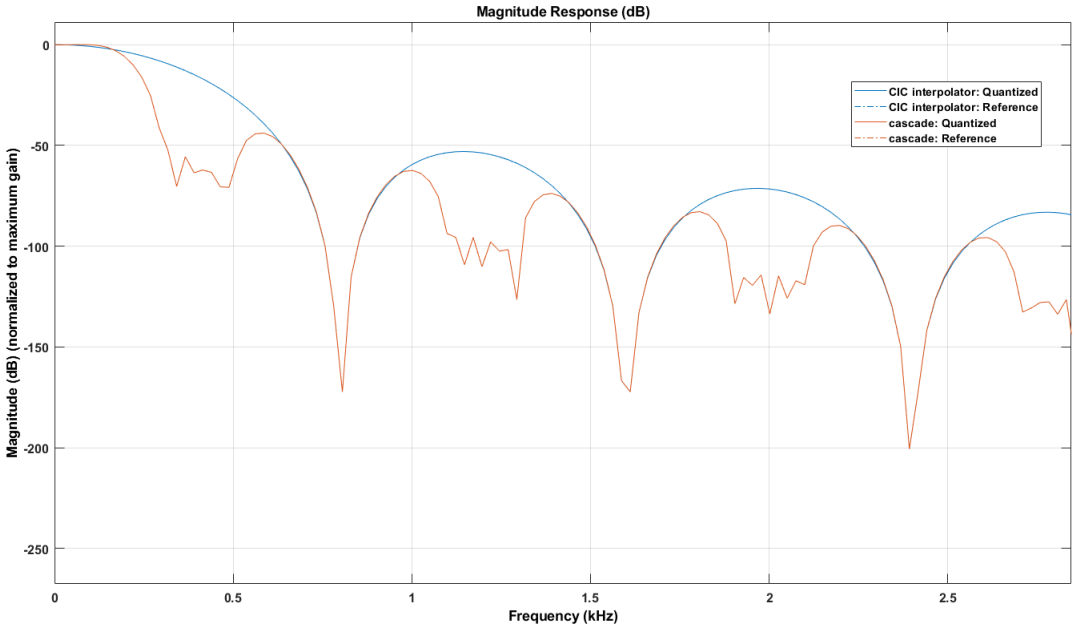


Figure 3.36: MATLAB Interpolator with FIR compensator output

FVTool is a filter visualization software which displays magnitude response, phase response, step response, impulse response and hardware resource estimates. The CIC interpolator and FIR compensator filter cascaded system requires 12 multipliers and 19 adders as shown in Figure 3.38.

3.5.1.1 Simulink system verification

Implementation The cascaded filter was implemented in the Simulink as shown in Figure 3.39. A source generator block was connected to the CIC interpolation filter and compensation filter subsystem. The source generator block uses a phase/frequency offset block for frequency and phase shifting the incoming signal. An input signal with frequencies starting from 0 Hz to 1200 Hz was applied to the CIC-FIR compensation cascaded system for plotting the system transfer funstion. The compensation filter is placed before an interpolator filter due to the low sampling rate at the input of the filter. An economical filter design in terms of hardware

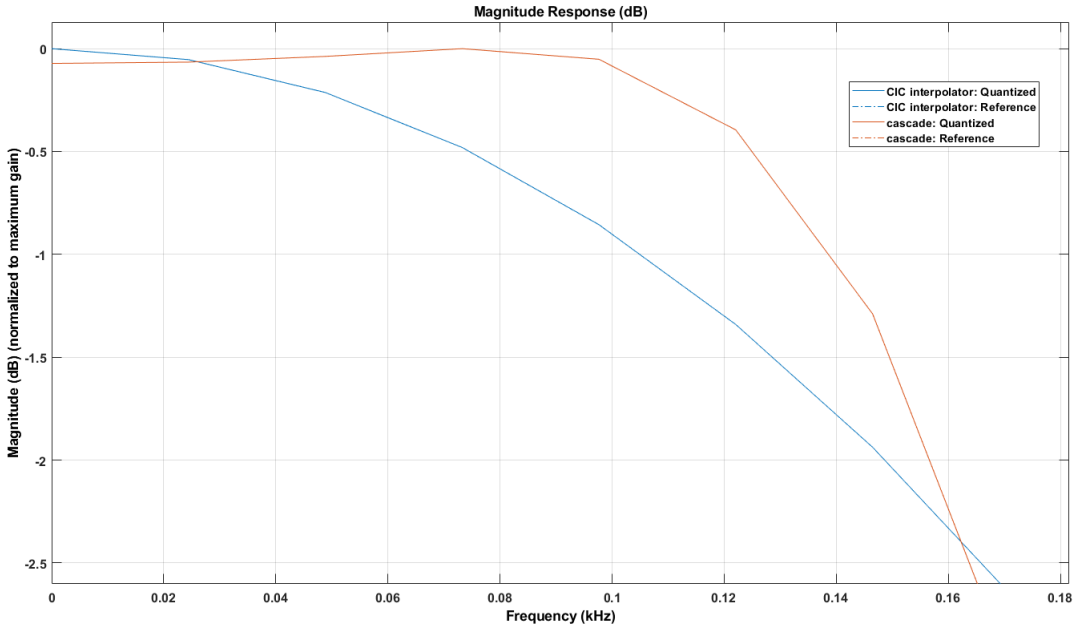


Figure 3.37: Passband droop and compensation of CIC interpolator

```
% --------------------------------
% cascade
% --------------------------------

Discrete-Time FIR Filter (real)
--------------------------------
Filter Structure  : Cascade
Number of Stages  : 2
Stable            : Yes
Linear Phase      : Yes (Type 3)

Implementation Cost
Number of Multipliers            : 12
Number of Adders                 : 19
Number of States                 : 19
Multiplications per Input Sample : 12
Additions per Input Sample       : 2015
```

Figure 3.38: MATLAB resourses utilization information for cascaded CIC filter

resources can be achieved by connecting the FIR compensation filter before the interpolation CIC filter as shown in Figure 3.40.

Cascaded CIC-FIR interpolator filter frequency response The system verification is done for the interpolation filter by applying signals at the input of the CIC interpolator and compensation filter cascade with a frequency of 0 to 1200 Hz using the phase/frequency offset block from the Communication Toolbox library. Figure 3.41 shows the cascaded CIC-FIR filter frequency response. It is observed that passband droop is compensated well with a 0.1 dB passband ripple. The aliasing

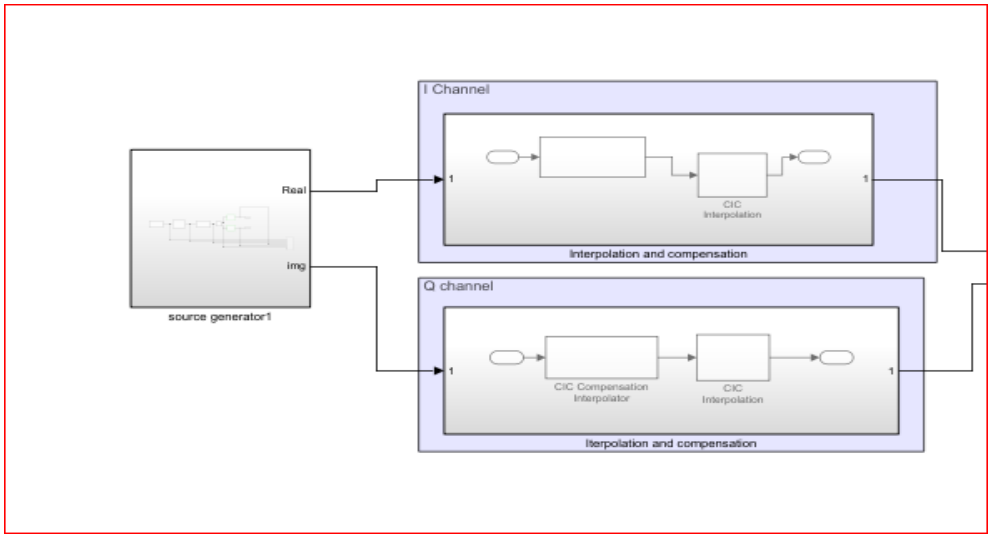


Figure 3.39: Cascaded CIC-FIR interpolation filter implementation block diagram

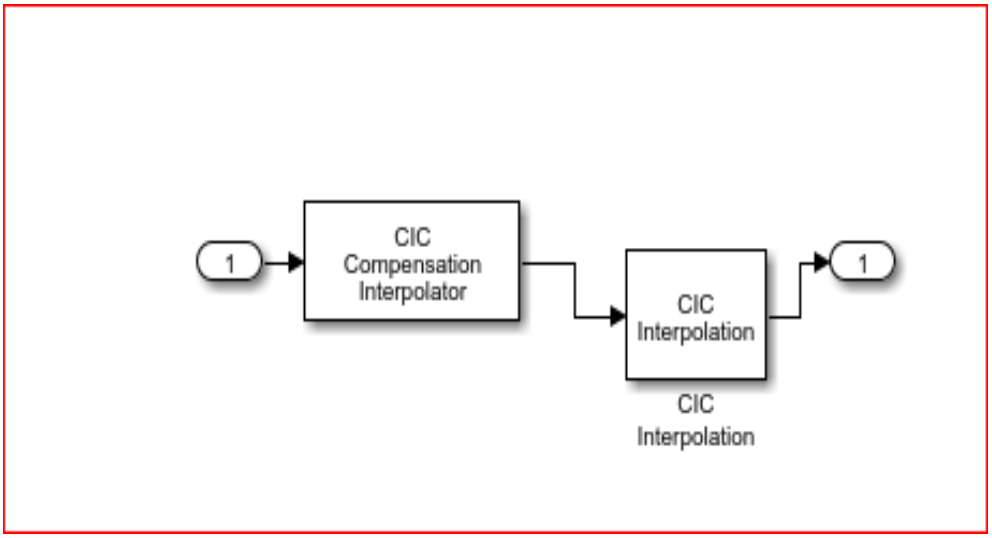


Figure 3.40: CIC-FIR filter simulink implementation block diagram

level is below 40 dB too. If more attenuation is needed of the aliasing components then CIC interpolation stages can be increased.

Eye diagram for ISI distortion An eye diagram represents the effects of inter symbolic interference (ISI) and quantization noise on the signals applied to the cascaded system. Parameters like undershoot, overshoot, sync delays with the system clock, and signal levels can be seen in the eye diagram. All of these parameters affect the eye structure. The more the distortion is more it is susceptible to noise. An eye diagram is drawn overlaying the consecutive bits on the same plot. The wider the eye-opening is the better the BER system will achieve. Easily distinguishable high and low levels and fewer amplitude variations represent a system with high BER.

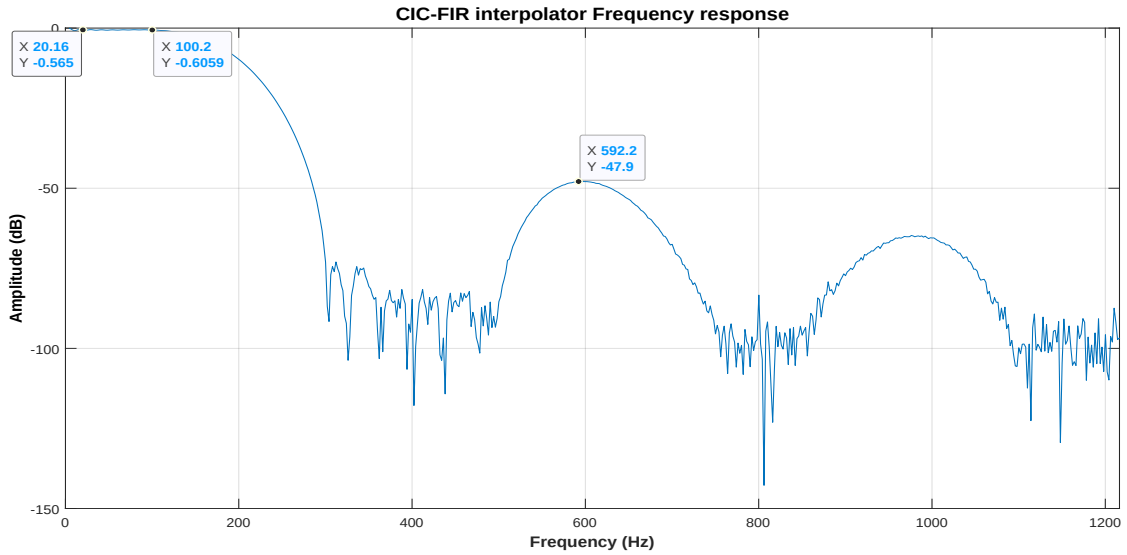


Figure 3.41: CIC interpolator and Compensator frequency response

While ISI and noise distortion results in closure of the eye. The eye diagram can be seen in Figure 3.42. Perfect sampling time is at the middle of the eye where SNR is

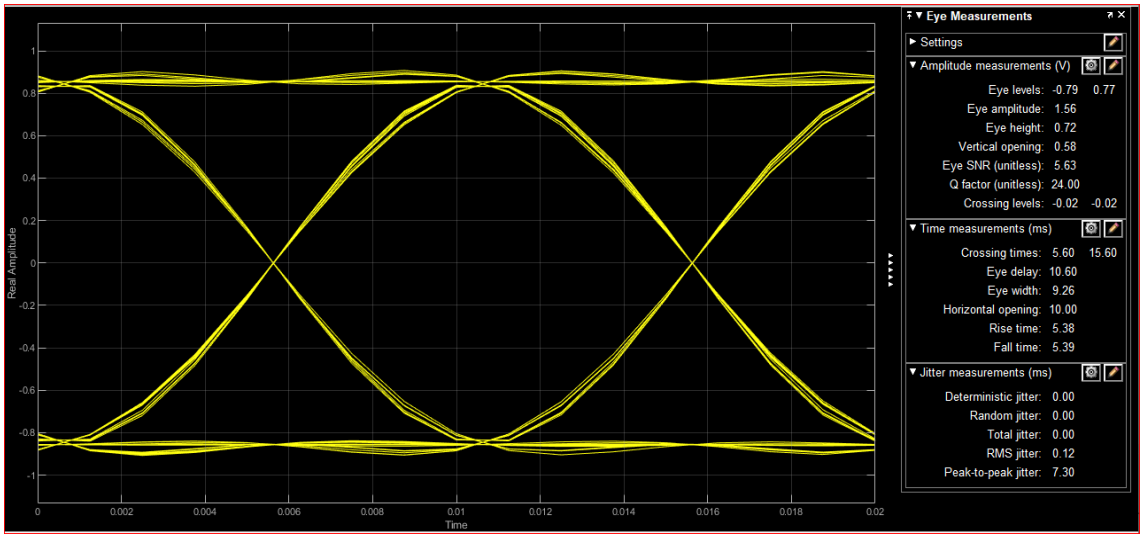


Figure 3.42: Eye diagram for cascaded CIC-Compensator filter

maximum. Horizontal eye width is the same as the bit duration of 10 ms. The crossing level for the filter should be ideally 0 but due to filtering it is 0.02 . This increased level will induce ISI of approximately -31 dB (20(log(0.02/0.72)) normalized to max amplitude.

Dynamic range and effective bits of the filter The dynamic range in dB specifies the difference between the levels of the highest signal to the lowest signal a system can measure. For example, if the filter has a dynamic range of 40 dB and can measure a maximum signal of 10 dB then the lowest signal it can measure will be -30 (10 − 40) dB. The designed cascaded filter outputs one value with multiple samples (500 for the given scenario) for two consecutive input values as shown in Figure 3.43. The upper subplot in the figure shows input samples with an 800 Hz sample rate while the lower plot is the output of the CIC-FIR interpolator cascaded filter. Input data width is 14 bits but after 1-bit loss (one output value for 2 input values) output data width reduces to 13 bits. The dynamic range can be calculated by the formula 20log (minimum amplitude / maximum amplitude) as 78.26 dB (20 log ($1/2^{13}$)).

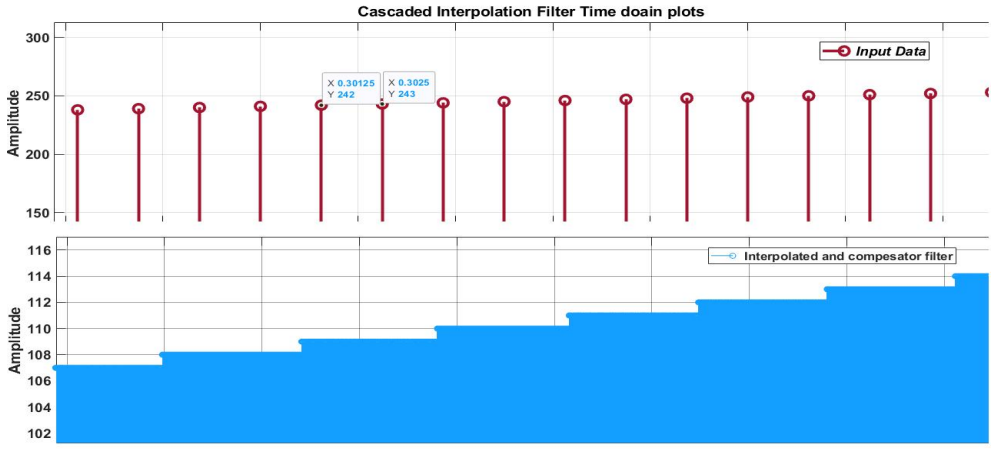


Figure 3.43: Cascaded interpolator and FIR filter time domain response

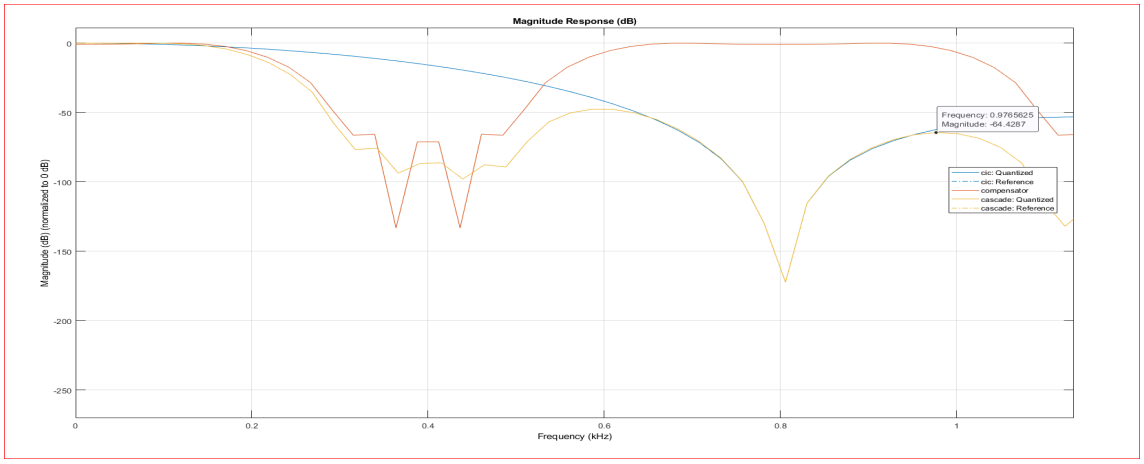
3.5.2 Decimation

Decimation is done by downsampling the incoming sequence and then low pass filtering it. A lowpass filter is applied to remove high-frequency interference components. The frequency response of the filter is given in Figure 3.44. The passband droop is corrected by compensation filter as plotted in Figure (3.44b). The passband ripple is also adjusted as can be seen from the Figure 3.44b (orange line) to 0.1 dB as per requirements.

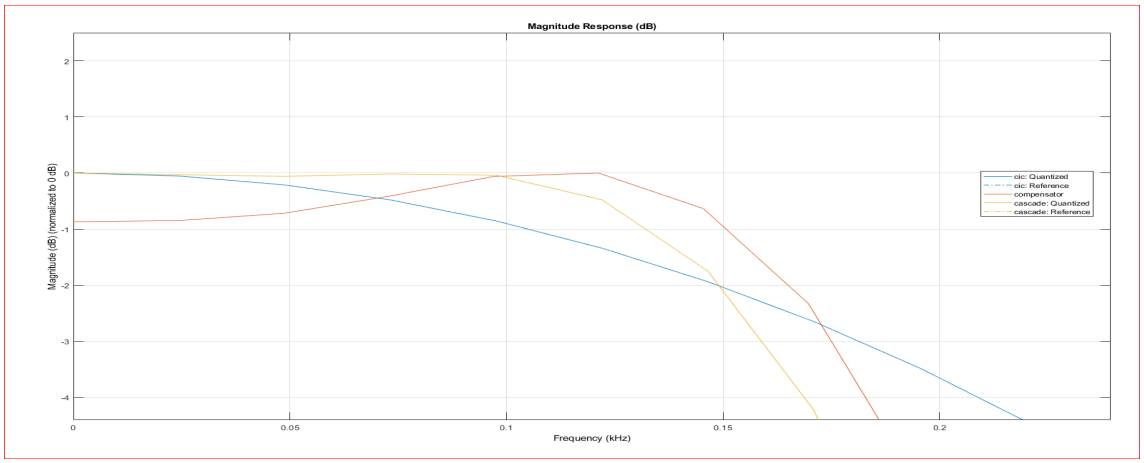
3.5.2.1 Simulink verification

Implementation

The cascaded decimation filter was implemented in Simulink as shown in Figure 3.45. A signal generation block same as the previous section was connected to



(a) CIC-Compenstor cascade frequency response



(b) Zoomed frequency response of CIC-Compensation filter

Figure 3.44: MATLAB implementation of CIC-Compensation filter

the CIC decimator and compensation filter cascade submodule. In the decimation process, an FIR filter is placed after the CIC decimator. The compensation FIR filter is placed at low sampling frequency due to fewer hardware resources required for computations [50]. The cascaded filter implementation block is shown in Figure 3.45. A CIC Decimation HDL optimized block from the DSP system Toolbox HDL support library is connected to the FIR compensation filter subsystem. A compensation filter is placed inside enable subsystem for making sure it operates only when valid data from the CIC decimator is received. Valid data is received when a valid strobe is set high by the CIC decimation block.

Overall system frequency response

Figure 3.46 shows the frequency response of the decimator and compensator cascaded system. The simulink implementation verifies the MATLAB implementation

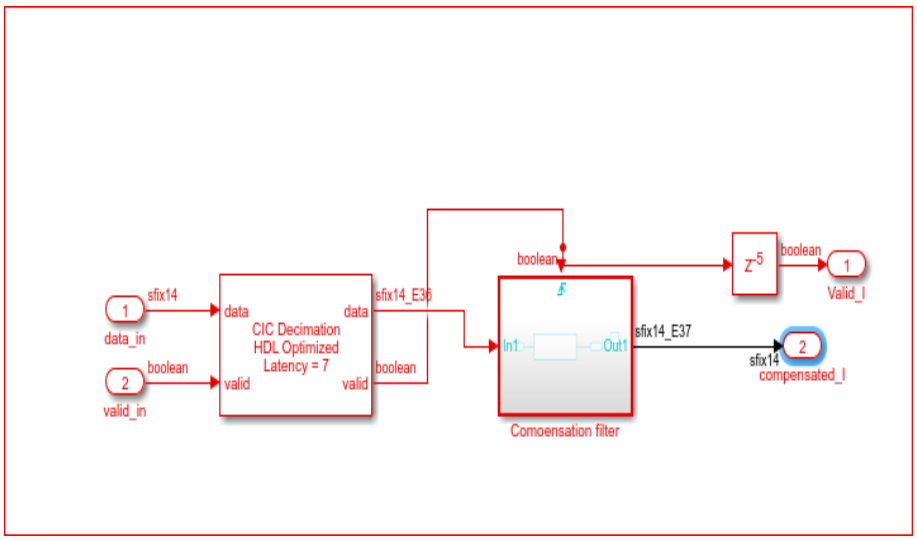


Figure 3.45: Cascaded filter implementation

as the side lobe level is less than 60 dB. The black line represents the CIC decimator frequency response compared to the cascaded system response (red line). The pass band droop is corrected by the cascaded filter to 0.1 db and aliasing signals are also attenuated by approximately 60 dB.

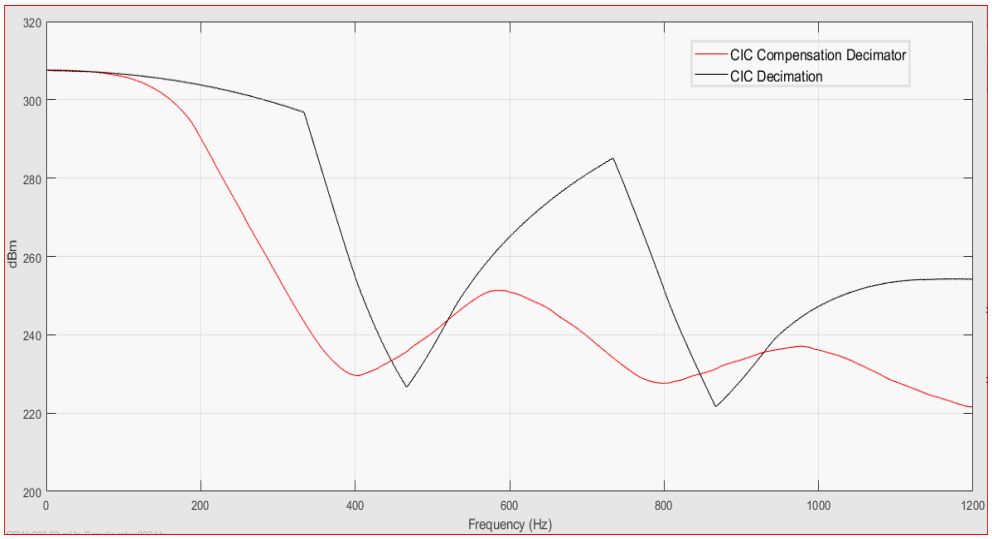


Figure 3.46: Decimation and compensation filter frequency response

Eye diagram

An eye diagram is used to analyze the filter distortions. If an eye is close or distorted then we say it is a bad system. The eye diagram for the cascaded response is given in Figure 3.47. It shows a high Q factor (23) at BER threshold 10^{-12} settings (ITU specified) which means it induces little to no distortion in the signal. The higher the

Q-factor cleaner is the signal.

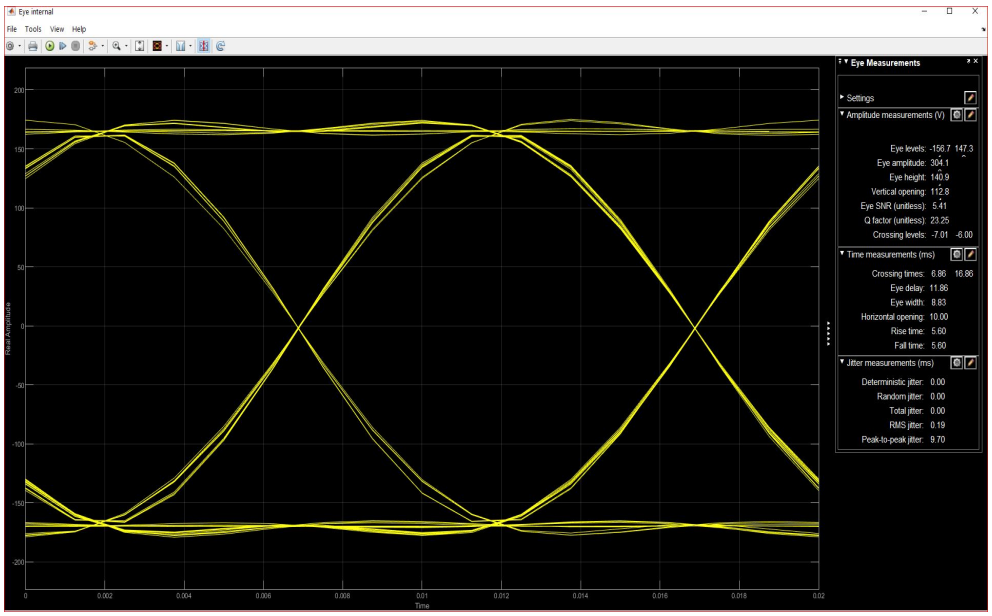


Figure 3.47: Eye diagram for cascaded decimation filter

Dynamic range

The dynamic range of the decimation filter is the same as the interpolation filter. The decimation CIC-FIR cascaded filter inserts 1-bit loss like the interpolation filter as shown in Figure 3.48. The uppermost stem plot shows the input data received at the input of the cascade system at a sample rate of 400 kHz.The stem plot's high density means that it has 500 samples at 400 Ksps. The Middle plot displays a valid strobe for the succeeding system to fetch valid data. The lowermost plot displays the final output of the cascaded system decimated data. The Decimator system produces 1 output value for every two input values. This can be confirmed by two high valid strobes at the same level of output data while two input values are fed to the decimator system. The valid signal frequency is 800 Hz (400e3 / 500 (decimation ratio) which is the decimated data rate. The decimator cascaded systems dynamic range is the same as the interpolator cascaded system discussed earlier (i.e. 78.26 dB).

3.5.3 Hardware synthesis and power estimates

The cascaded FIR-CIC filters HDL files were generated by the MATLAB HDL coder (version 3.17). The overall design of the CIC-FIR interpolator and CIC-FIR decimator system with the required data interfaces were synthesized in Libero SoC v12.2 for IGLOO2-005 flash-based FPGA. The block diagram shown in Figure 3.29 was

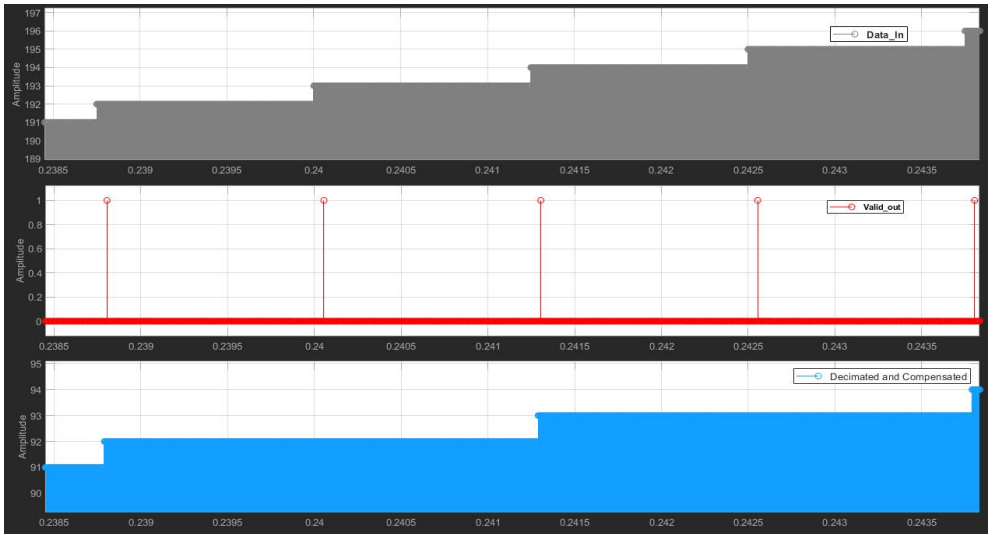
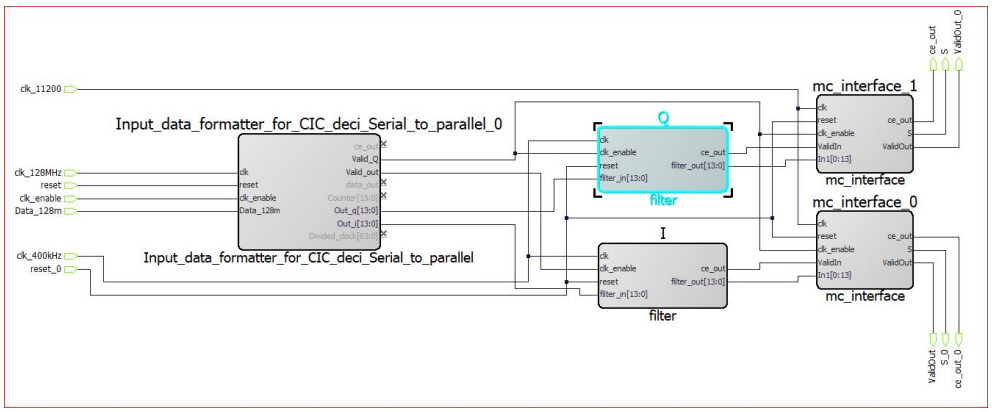
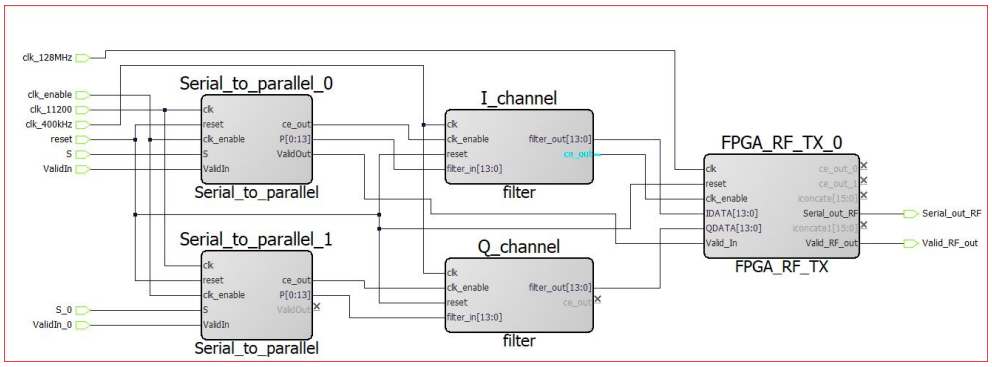


Figure 3.48: Decimation filter time domain stem plot

followed for HDL synthesis.
Synthesis report informs us about the hardware resources required in an implemen-



(a) Receiver implementation design



(b) Transmitter implementation design

Figure 3.49: Transmitter and receiver FPGA implementation design

tation of a design. In a three-chip solution, filters are implemented on an FPGA.

The FPGA implementation schematic drawing of the transmitter and the receiver are shown in Figure 3.49. In Figure 3.49 (a) receiver's input data formatter block receives data from RF IC and transfers it to the I and Q filter chain as shown. The I and Q block performs decimation on the received data and sends it to the microcontroller interface (mc_interface). The microcontroller interface transfers data to the external microcontroller via a serial interface (@11.2kbps). The transmitter in Figure 3.49 (b) receives data from external microcontroller on a serial interface (@11.2 kbps). The serial data is deserialized by serial to parallel block for I_channel and Q_channel filter blocks. These filter blocks interpolate data by 500 (interpolation ration) and transfer it to the FPGA_RF subsystem. The FPGA_RF subsystem deserializes the data and appends the required number of zeros. The output of the FPGA_RF block (Serial_out_RF) is connected to the RF chip. Figure 3.50 a and b displays the required number of resources needed for implementing receiver chain and transmitter chain in FPGA respectively. The CIC interpolator cascaded with FIR filter occupies only 2569 (42 %) 4LUT and 1575 (25 %) DFFs of the IGLOO2-005 FPGA.

Type	Used	Total	Percentage
4LUT	2574	6060	42.48
DFF	1280	6060	21.12
I/O Register	0	252	0.00
User I/O	13	84	15.48
-- Single-ended I/O	13	84	15.48
-- Differential I/O Pairs	0	38	0.00
RAM64x18	0	11	0.00
RAM1K18	0	10	0.00
MACC	11	11	100.00
Chip Globals	5	8	62.50
CCC	0	2	0.00
RCOSC_25_50MHZ	0	1	0.00
RCOSC_1MHZ	0	1	0.00
XTLOSC	0	1	0.00
HPMS	0	1	0.00

(a) FPGA receiver chain resources

Type	Used	Total	Percentage
4LUT	2703	6060	44.60
DFF	1746	6060	28.81
I/O Register	0	252	0.00
User I/O	11	84	13.10
-- Single-ended I/O	11	84	13.10
-- Differential I/O Pairs	0	38	0.00
RAM64x18	0	11	0.00
RAM1K18	0	10	0.00
MACC	11	11	100.00
Chip Globals	4	8	50.00
CCC	0	2	0.00
RCOSC_25_50MHZ	0	1	0.00
RCOSC_1MHZ	0	1	0.00
XTLOSC	0	1	0.00
HPMS	0	1	0.00

(b) FPGA transmitter chain resources

Figure 3.50: Hardware resource synthesis report on IGLOO2-M2GL005 (Microsemi)

3.5.4 Power consumption estimate

In a battery limited product power consumption plays a vital role in product life. The power consumption of the transmitter and receiver chain is shown in Figure 3.51. It

can be observed from Figure 3.51 that both the transmitter and the receiver chain consumes almost equal power which is logical because the CIC interpolator and the decimator are the mirror image of each other for the same number of filter stages. Mirror image means integrator filter precedes comb filters in CIC decimation filters while in CIC interpolation filters comb filters leads the integrator filters. A small difference in resource requirements is observed due to the RF interface implementation design. A transmitter RF interface requires a concatenate block for sync mark insertion and a state machine for serializing the data with required zero bit padding. While receiver RF interface just checks for sync mark and stores data accordingly.

Power Summary

	Power (mW)	Percentage
Total Power	14.558	100.0%
Static Power	11.382	78.2%
Dynamic Power	3.176	21.8%

(a) FPGA transmitter chain power consumption estimates

Power Summary

	Power (mW)	Percentage
Total Power	14.024	100.0%
Static Power	10.532	75.1%
Dynamic Power	3.492	24.9%

(b) FPGA receiver chain power consumption estimates

Figure 3.51: Power consumption report for Tx and Rx FPGA filter design on IGLOO2-M2GL005 (Microsemi)

3.6 Conclusion

The reconfigurable nodes will be in sleep mode for approximately 99 % of the time. So it is advisable to turn off the FPGA for saving power. If FPGA is turned off then SRAM-based FPGA will lose its configuration. To reconfigure again, high currents at startup are required and it will take time to reconfigure itself. The flash-based FPGAs are handy in this scenario because they can be turned off and they do not require reconfiguration too. A flash-based FPGA consumes 33% less power than their competitors [53]. Due to low power operation and almost zero reconfiguration time, the flash-based FPGAs are the best option for decimation and interpolation stages. Table 3.3 shows the tentative minimum resources requirements for a two-chip and three chip-based node design.

It is concluded from the table that a two-chip transmitter design requires 5140 (2566 + 2574 (CIC-FIR interpolator)) 4LUTs and 3377 (CIC-FIR interpolator)) DFFs.

Table 3.3: Hardware resources required for FPGA implementation

	Two Chip Design		Three Chip design	
Parameters	Tx	Rx	Interpolator	Decimator
4LUT	5140	26482	2574	2703
DFF	3377	15848	1280	1746
MAC	0	34	11	11
Interfaces	3 input 2x output (LVDS)	2 LVDS input and 3 x output	8 x input 2 x output (LVDS)	2x LVDS, 3 x input 6 x output
Minimum Clock frequency (MHz)	128	128	128	128
Dynamic power (mW)	3.55	9.988	3.176	3.492

While receiver requires 26482 (23779 + 2703 (CIC-FIR decimator)) 4LUTs and 15848 (14102 + 1746 (CIC-FIR decimator)) DFFs. In a three-chip solution, transceiver filters require 5277 4LUTs and 3026 DFFs. Another important factor affecting the choice of an FPGA is the math block requirements. A three-chip design transceiver requires 22 MAC blocks. One MAC block consumes 213 4LUTs if hard MAC blocks are not used while synthesizing. The Microsemi IGLOO 2 M2GL010 has 22 MAC blocks and double the number of resources than the M2GL005 FPGA. The M2GL-005 FPGA is not suitable for a three-chip transceiver design because it only has 11 MAC blocks. The approximate logic resources required to implement the remaining 11 MAC blocks are 2343 4LUTs. That means a total of 7620 4LUTs are required. M2GL-005 FPGA has only 6060 4LUTs so M2GL-010 qualifies for a three chip-based implementation depending on the resources required. The power consumption and price per piece (€13.7 (M2GL010T-FGG484)) further make M2GL-010 a favorable option. The Microsemi's other FPGA families were discarded as an option due to the price per piece for required amount of the logic resources. The IGLOO2 M2GL005 has 6060 4LUTs and 11 MAC blocks that can be used for either transmitter or receiver chain only.

Chapter 4

Microcontroller

In this chapter, a microcontroller-based processing module design and simulations results are discussed. The transmitter and the receiver modules are the same as discussed earlier in chapter 3.

4.1 Introduction

A microcontroller can be considered as an independent system with a processor, RAM, ROM, and interface peripherals. They are found in a computer, laptops, industrial instruments, voltmeters, cell phones, airbags, routers, etc. They have instruction based architecture either it is RISC (Reduced Instruction set computer) or CISC (Complex instruction set computer). In Chapter 3 we have designed a decimator and an interpolator filter. This filter helps in down/up sampling the data by decimation/interpolation ratio of 500. The modulator and demodulator with required baseband signal processing blocks are implemented in a microcontroller. The PHY will be implemented on a microcontroller and end-to-end verification is done without rate conversion filters (FPGA). The microcontroller PHY testing the transmitter modules are connected directly to the receiver system.

4.2 Three chip design

A microcontroller is a vital component in the three-chip design. The microcontroller collects data, processes it, and transfers it to either an FPGA or to the sensors attached. A transmitter accepts user data in analog or digital format. If data is analog it is first digitized by the ADC. The digital data is source coded first, packetized, modulated, and pulse shaped as per standards. The block diagram of the transmitter is shown in Figure 4.1. The transmitter consists of a packet generator, a QPSK modulator, a pulse-shaping filter, and a serializer interface. It can be noticed that

in a three-chip solution no RF interface is required at the microcontroller side. The

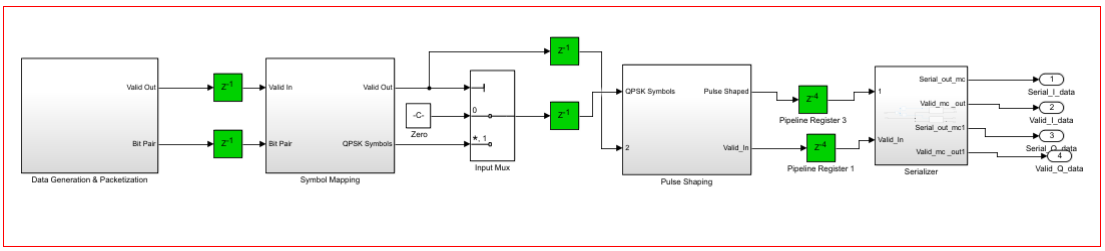


Figure 4.1: Microcontroller based QPSK transmitter

microcontroller transfers or receives data at 11.2 kbps to or from an FPGA for I-data and Q-data streams via the serializer interface. The first three submodules are the same as the FPGA QPSK transmitter discussed earlier. The Serializer interface is implemented here because data needs to be transferred between the FPGA and the microcontroller.

The serializer submodule consists of Compex to Real-Imag block from the HDL Coder library which converts complex input to real and imaginary components. The real part represents I-data and the imaginary branch represents Q-data. Divided channel data is then serialized using Serializer1D block from the HDL Coder library as shown in Figure 4.2. The I-data and the Q-data are transferred to the FPGA with a valid strobe indicating valid data. The I-data and Q-data of 14-bits each are converted to a serial data rate of 11.2 kbps (800 * 14). These separated branches are filtered and interpolated in the FPGA.

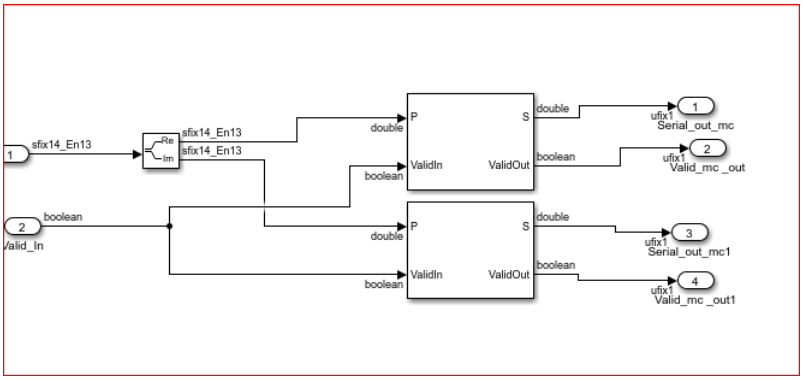


Figure 4.2: Transmitter serial interface

4.3 Receiver

The receiver block diagram can be seen in the Figure **??**. The receiver block consists of the Desrializer1D block from the HDL library, an adaptive gain control (AGC),

a matching filter, a frequency compensation(Fine and coarse) block, a symbol timing recovery block, and a data decoding subsystem (Figure **??**).

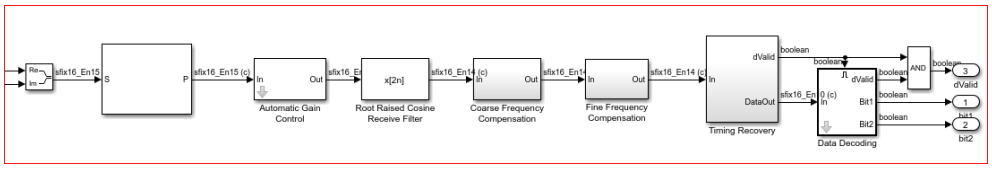


Figure 4.3: Microcontroller based receiver simulink block diagram

4.3.1 Deserializer1D HDL coder

Deserializer1D converts the incoming serial stream to parallel data. I-data and Q-data is 14-bits so deserializer converts serial data to 14 bits parallel bus for each channel. After converting serial stream to parallel bits, data rate drops to 800 bps from 11.2 kbps. This 800 bps is processed for information recovery.

4.4 Testing and verification

The testing is done to quantify the system performance according to predefined standards or conditions. The quantization of floating data and pulse shaping filters truncated impulse response introduces some error in the final output. This error is recorded for system performance. The advantage of pulse shaping a signal is that it reduces bandwidth but that comes at a cost of introducing quantization error to the signal. The QPSK transmitter and the receiver were interconnected to evaluate the system performance. The system performance can be judged by calculating the ISI levels, error vector magnitude (EVM), Root-mean-square (RMS) error percentage, and observing eye diagrams.

4.4.1 Pulse shaping filters

Two filters were tested for reconfigurable nodes implementation design. Parameters for the filters are shown in Table 4.4.1

	Filter 1	Filter 2	Filter 3
Filter span	10	10	10
Roll off	1	1	0.5
Samples per symbol	8	16	16

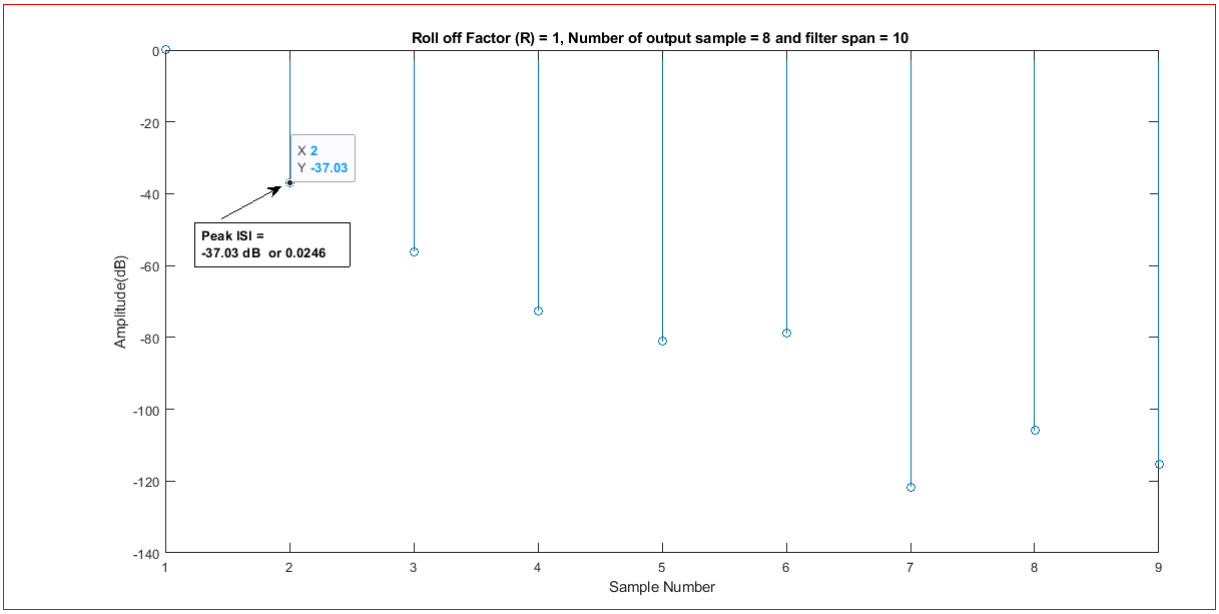


Figure 4.4: Filter 1 ISI measurement

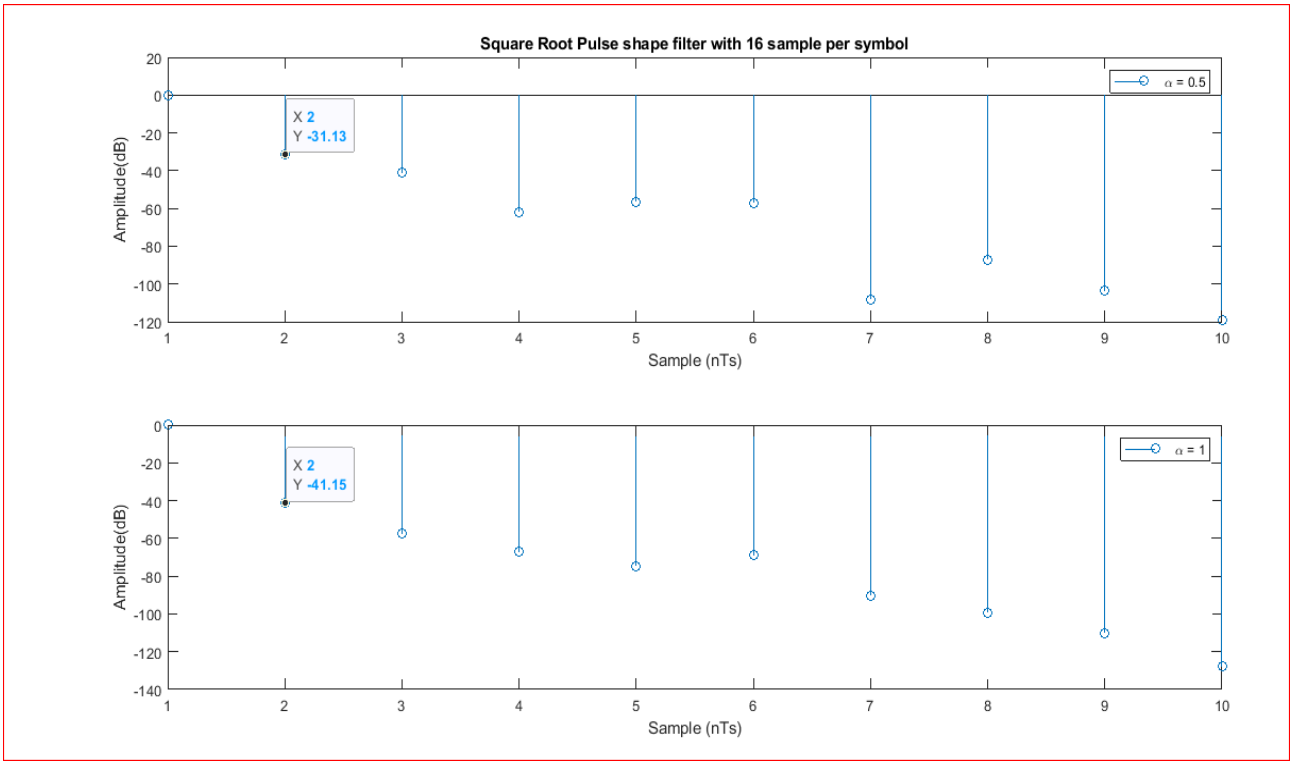


Figure 4.5: Filter 2 and Filter 3 ISI measurements

The ISI measurements were taken by convolution of the filter impulse response by its own impulse response (convolving transmit filter with receive filter) and then skipping intermediate sample while keeping sample at nTs (n = 1-10). It can be observed from the Figure 4.4 and Figure 4.5 that as the samples per symbol at

the output of the filter increase the ISI decreases. Figure 4.4 represents peak ISI amplitude of -37 dB for the output samples per symbol = 8 and the roll-off equal to 1, while filter 2 have a peak ISI amplitude of -41dB. This ISI increased attenuation of filter 2 w.r.t filter 3 is due to excessive roll-off bandwidth. The larger the filter samples per symbol lesser will be the ISI but more computation-intensive will it be due to multipliers requirement. Total multipliers required can be calculated as samples per symbol multiplied by the filter symbol span.
The filter 2 and the filter 3 have a different roll-off of 1 and 0.5 respectively. It can be concluded that as the filter excess bandwidth increases ISI will decrease. This is again counter-intuitive that infinite in frequency domain means limited in the time domain and vice versa [43].
Visually the effect of excess bandwidth of the pulse shaping filter can be seen from Figure 4.6 and 4.7. Figure 4.6 is more distorted than the Figure 4.7 due to roll-off factor difference which induces noise in the given eye diagram. In the given scenario,

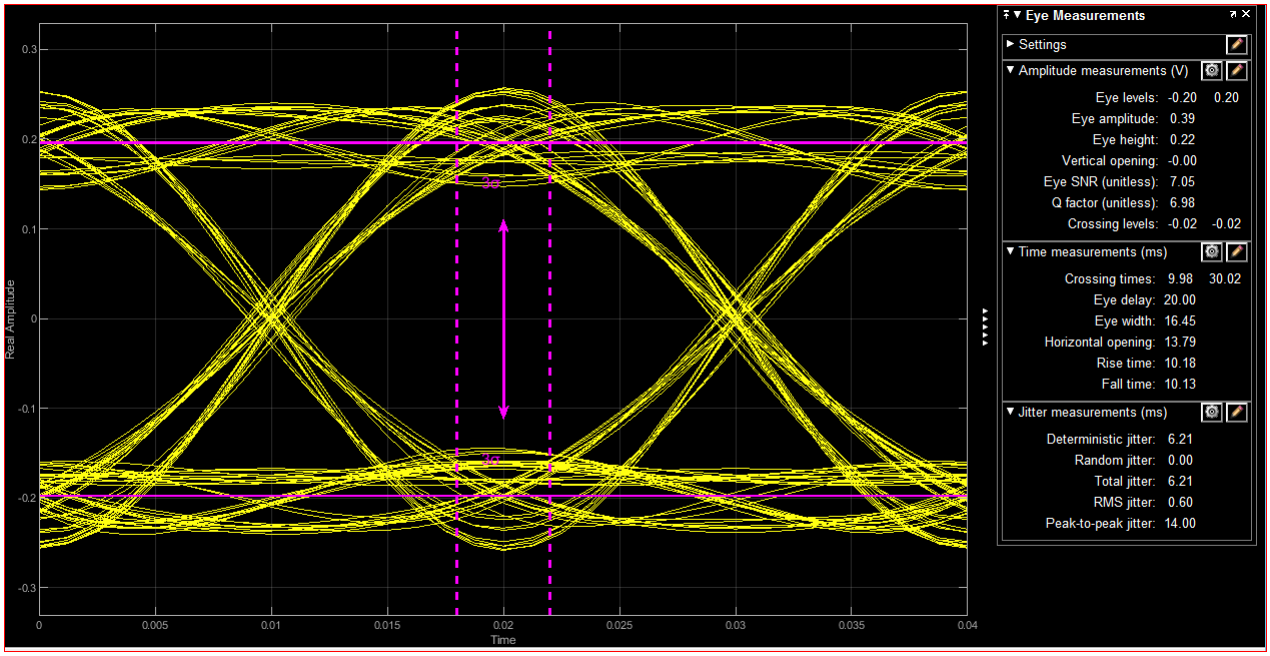


Figure 4.6: Filter 3 eye diagram

the roll of factor = 1 is selected due to minimum ISI injection in the initial transmission stage and secondly, we can use upto 600 kHz bandwidth in sub-1GHz band while the given node only occupies up to 100 Hz bandwidth only.
The input signal after pulse shaping at transmitter can be seen in the Figure 4.8. We can observe that square pulse are now smoothly transitioning from high to low and vice versa. Smoother transitions in time domain have lesser high frequency components thus limiting the bandwidth of the signal of interest.

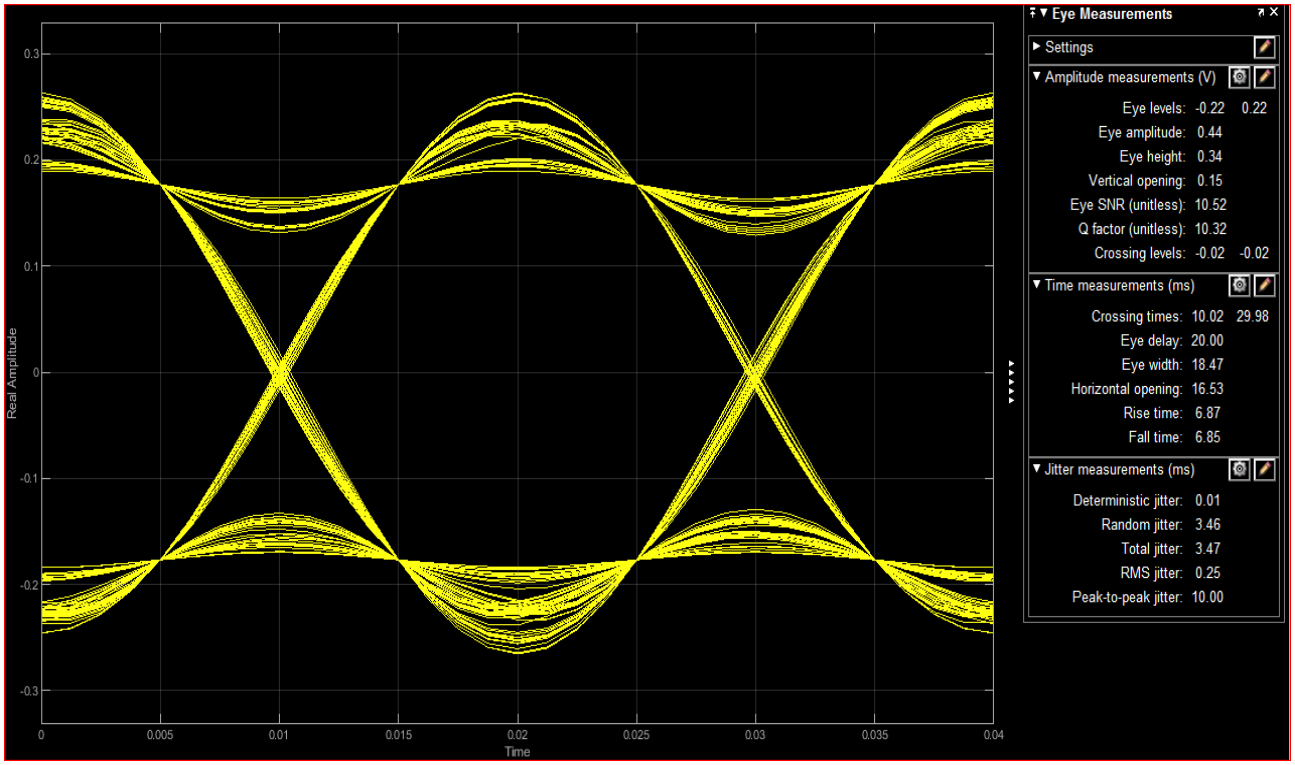


Figure 4.7: Filter 2 eye diagram

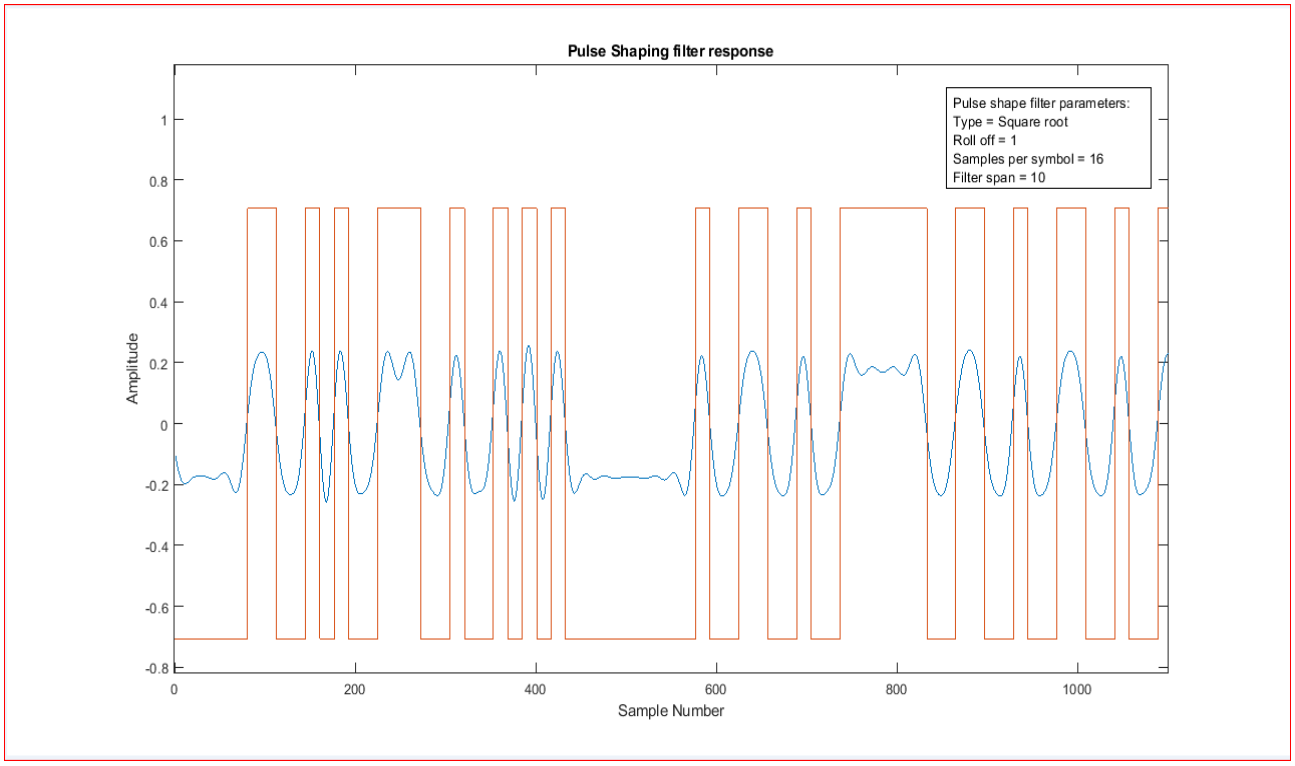


Figure 4.8: Pulse shaped data output of filter 2 at the transmitter

Constellations plots are obtained at the receiver end after the matched filters to get the EVM and RMS error. Figure 4.9 shows the constellation plot at the output of matched filter. Transmitter and receiver filters had the parameters of filter 2 as

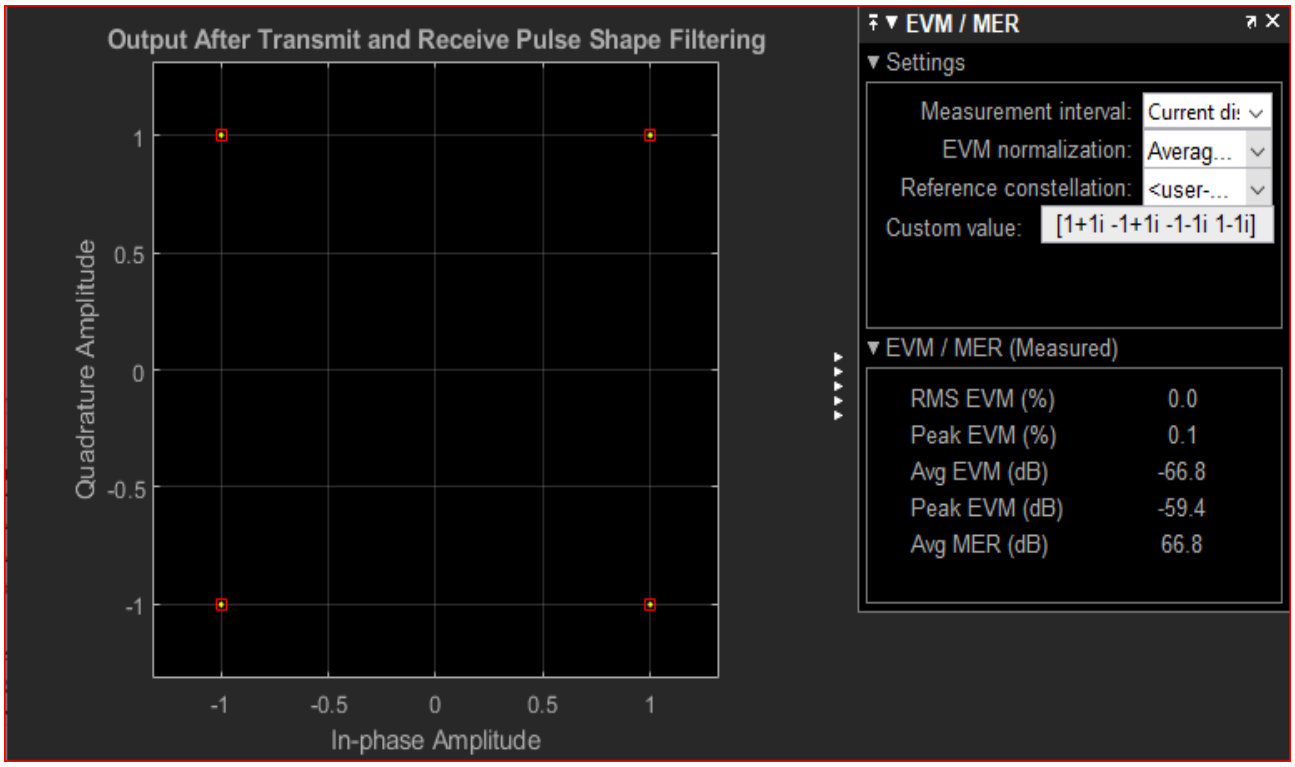
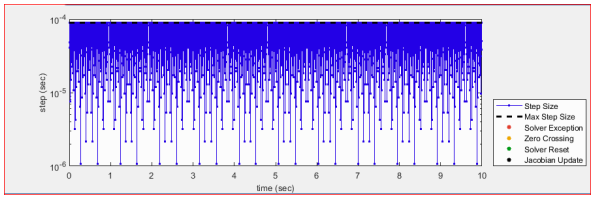


Figure 4.9: Receiver matched filter output constellation diagram

given in Table 4.4.1. The quantization noise due to truncated filters response and the coefficients in RRC filters is almost negligible. The EVM for the given system is below 65 dB and RMS EVM (%) is almost zero upto 1 decimal point. This means that both transmit and receive filters are adding barely any noise to the signal.

4.4.2 Microprocessor profiling

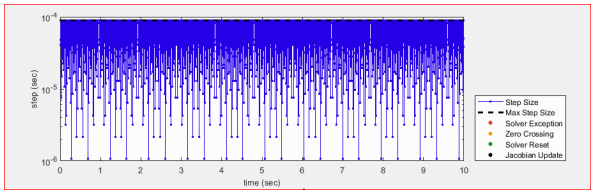
In the Simulink, a solver profiler was used to get the estimate for the required maximum sampling frequency for the baseband processor. Figure 4.10 shows the solver profiler output for the transmitter and receiver modules. It can be deduced from the figure that the maximum sample rate required, is equal to the serial data interface clock, that is between FPGA and microcontroller. The minimum sampling clock which a microcontroller must have is $\approx$ 22.4kHz.



(a) Transmitter baseband profile chart

STEP INFORMATION	
Max step size	8.93e-05
Min step size	auto
Average step size	8.86e-05
Max step size usage(%)	98.49
Total steps	112854

(b) Tx profiler information



(c) Receiver baseband profile chart

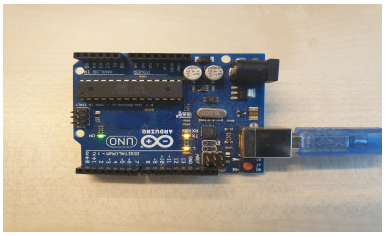
STEP INFORMATION	
Max step size	8.93e-05
Min step size	auto
Average step size	8.86e-05
Max step size usage(%)	98.49
Total steps	112854

(d) RX profiler information

Figure 4.10: Solver profiler for baseband transmitter and receiver

4.4.3 Processor-In-Loop (PIL) testing

MATLAB Simulink supports PIL testing in which code is compiled and run on hardware connected to the model. The code is directly downloaded to the hardware and it can be analyzed in the Simulink like a normal program. A QPSK transmitter code with an output data rate of 800 bps was compiled on the available board (Arduino UNO Figure 4.11 (a)). The program memory and data memory required for a microcontroller can be seen in Figure 4.11 (b) The output of the simulation block was compared to the output from the code running on Arduino UNO. The Figure 4.12 (a) displays a zero error between PIL output and the simulation output for both I-channel and Q-channel data. It can be concluded from PIL testing that a microcontroller having 11276 bytes of program memory and 1386 bytes of data memory can be used as a transmitter only node.



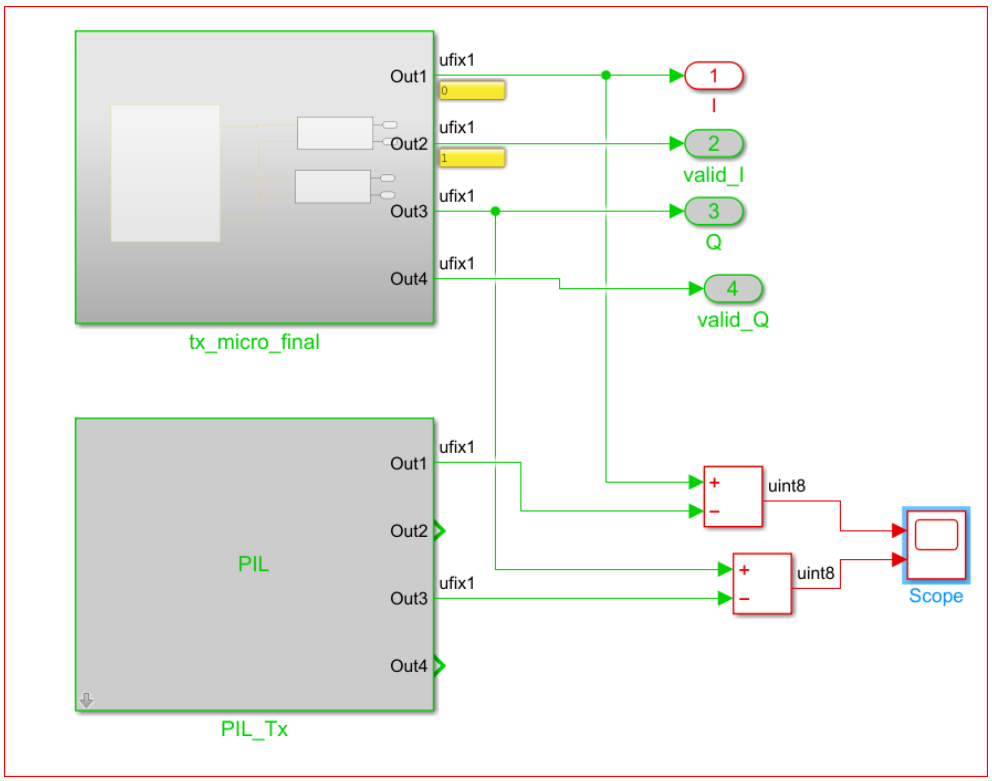
(a) Arduino Uno board PIL setup

```
### Starting application: 'tx_ert_rtw\pil\tx.elf'
AVR Memory Usage
----------------
Device: atmega328p

Program:   11276 bytes (34.4% Full)
(.text + .data + .bootloader)

Data:       1386 bytes (67.7% Full)
(.data + .bss + .noinit)
Starting PIL Simulation
```

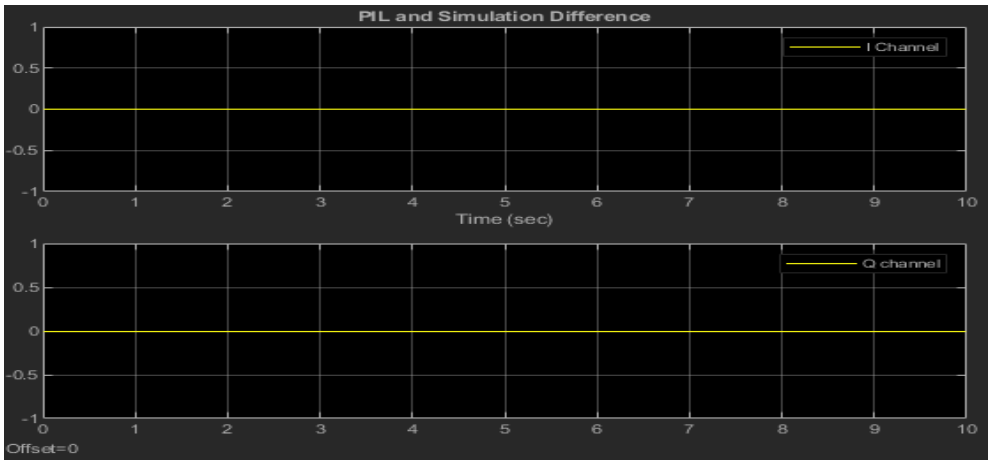
(b) Transmitter microcontroller Parameters

Figure 4.11: Microcontroller based QPSK transmitter testing

Likewise transmitter, the receiver PIL testing steps were performed on the Arduino Uno board. The receiver chain is more complicated than the transmitter



(a) PIL block verification Simulink



(b) PIL and Simulink block output difference

Figure 4.12: PIL verification on Arduino Uno board

side so the whole receiver code memory and program memory requirements were greater than the available resources on an Arduino Uno board. That's why separately each block was compiled and placed inside the receiver chain and errors were observed at the packet decoding end. All of the blocks worked on the Arduino Uno board as per simulation results discussed in Chapter 3. The separate compilation helped in getting an estimate of microcontroller resources for the whole receiver chain. The data memory and program memory for receiver subblocks are given

in Table 4.1. To implement a QPSK receiver, approximately 22 kbytes of program memory and 2 Kbytes of data memory are required.

Block name	Program memory (ROM) kilobyte	Data memory (RAM) kilobyte
AGC and RRC filter	5	2.5
Coarse frequency offset	4	0.5
Fine frequency offset	4	0.2
Timing Synchronizer block	5	0.1
Data Decode block	3	0.4
Packet decode	1	0.2
Total (approx)	22	3.9

Table 4.1: Memory requirments for microcontroller-based QPSK receiver implementation

Table 4.2 shows some of the low-powered microcontrollers designed specifically for IoT products with longer battery life.

Table 4.2: Low-power 32-bit microcontrollers

Name	Speed (MHz)	Core	RAM (kilobytes)	ROM (kilobytes)	Interfaces	Power				Package (mm^2)	Cost ($)
						Active ($\mu A/MHz$)	Standby μA	Sleep μA	Deep sleep μA		
STM32L081CZ	32	ARM Cortex M0+	20	128	40 gpio , 2x SPI	93	0.86	0.43	0.29	7 x 7	3.55
STM32L431	80	ARM Cortex M4 with FPU	64	128	83 gpio, 3x SPI	84	280 nA	28 nA	8 nA	5 x 5	4.2
MicroChip SAM 10/11E16A	32	ARM Cortex M23	16	16/32/64	25 gpio , 1x SPI	25	1.5	0.5	100 nA	2.79 x 2.79	1.49
SAM L21E18	48	ARM cortex M0+	16	256	51 gpio, 1 xSPI	35	1.2	0.9	200 nA	9 x 9	3.79
Intel SEC1000	32	X86ISA	80	384	32 gpio , 1x SPI	250	-	-	650 nA	6.4 x 6.33	11.12
Maxim 32666	96	ARM Cortex M4	384	1000	37 gpio, 1 xSPI	27	12.3	7.8	0.39	4.1 x 3.7	6.52
Analog ADuCM355	26	ARM Cortex M3	64	128	17 gpio, 1 x SPI	30	8.5	-	2	6 x 5	5.90

4.5 Conclusion

The main concern in the microcontroller selection is the power consumption, memory (program and data), and the required number of interface peripherals. Program memory and data memory controls the size of code that can be uploaded to the microcontroller. Table 4.1 indicates memory requirements for a receiver. In a QPSK transceiver design, approximately 33 kbytes of program memory and 5kbyes of data memory is required. The minimum number of GPIOs required in a transmitter implementation is 8. All of them as an output (clock, reset, valid strobe, data) for I-channel data and Q-channel data 4 each. The microcontroller-based receiver requires 6 input (chip enable, valid strobe, data) ports for I-channel and Q-channel 3

each. Additionally, the microcontroller must have an SPI interface for RF transceiver IC registers configuration.
All of the microcontrollers listed qualified for the reconfigurable node hardware w.r.t memory requirements and peripheral requirements. The power consumption metrics for low power microcontrollers are defined by the Embedded Microprocessor Benchmark Consortium (EEMBC). This metric is also known as the ULP mark. The ULP mark is mostly quoted in the datasheet by the vendor. Higher the ULP mark better is the device power saving capability [54].
All microcontrollers given in the table doesnot have the ULP mark stated. To decide among these microcontrollers node daily activity specification were decided on following parameters as given in Table 4.3. Day active time means how many times a node is on for transmission in a day. Rest of the time is considered as a sleep period.

According to node activity parameters devices were considered either active or in

Clock frequency	32 MHz
Data rate	100 bps
Packet size	200 bits
Messages per day	96 (1 message every 15 minutes)
message duration	2.0 s

Table 4.3: power consumption calculating parameters

deep sleep. Only two modes were considered due to non availability of current consumption value for different modes. Number of peripherals active in different modes are also not same. Table 4.4 shows the current consumption per day calculated based on the formula given.

$$C = F * A * N_m * M_d + C_d * T_i \tag{4.1}$$

Where C is power consumption in a day (mAsec), F is the operating frequency (32 MHz), A is the active state current (μ A) , N_m is number of messages per day, M_d is message duration (sec), C_d is deep sleep current (μ A) and T_i is the inactivity time per day (sec).

Based on the values in Table 4.4 the Microchip's Sam10 microcontroller had the least current consumption. The ULP mark for SAM 10 is 410 as reported which is the highest among the reported microcontrollers in Table 4.2. The Microchip SAM 10/11E16A microcontrollers are best fit for the reconfigurable LPWAN nodes with a program memory usage of 51% (22k/64k) and data memory consumption of 31%

Microcontroller	Active current μ A	Deep Sleep current μ A	Power consumtion (mAsec)
STM32L081CZ	93	0.29	596
STM32L4	84	0.08	522
Microchip SAM10/11E16A	25	0.1	162
SAM L21/22	35	0.2	232
Intel SEC1000	250	0.65	1592
Maxim 32666	27	0.29	190
Analog ADUC355	8.5	2	224

Table 4.4: Approximate power consumption per day

(5k/16k) and transmission of 4 messages per hour with a packet size of 200-bits.

Chapter 5

Integrated System Verification

In this chapter, an end-to-end communication system was simulated for validation of the concept. The system was connected as shown in Figure 5.1. The transmitter chain is connected to the receiver chain via the AWGN channel.

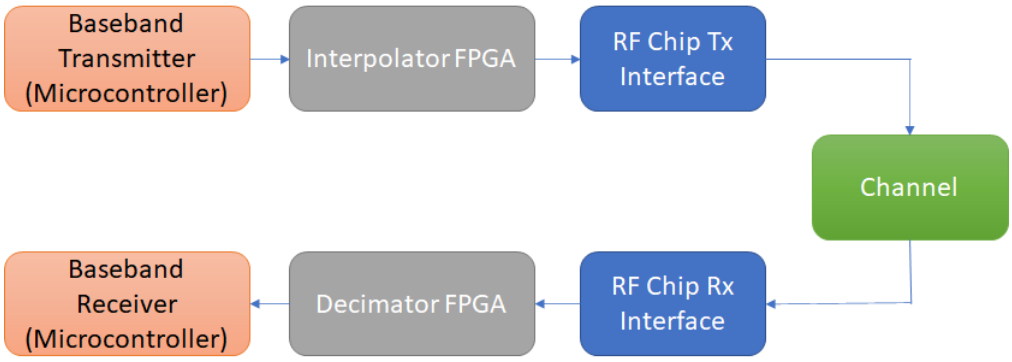


Figure 5.1: Overall system testing block diagram

The baseband modulator was connected to the FPGA interpolator via the serial interface. The serial interface consists of 14 bits of signed data at 800 bps data rate. The serial data is deserialized inside the FPGA for the interpolation process. The upsampled data is transferred to the RF transmitter interface. The RF interfaces are explained in Chapter 3. The RF Tx interface formats the data according to the Atmel RF IC data requirements. The RF IC transmits data to the receiver via the channel interface. A noisy data is received serially by the RF Rx interface. The receiver RF interface provides data to the FPGA decimator for downsampling and filtering. The downsampled data is transferred via a serial interface to the receiver baseband processor.

5.1 Channel

The performance of a system in a noisy channel medium plays a vital role in benchmarking a system's performance. An AWGN channel was inserted between the transmitter and the receiver to replicate a real scenario. Figure 5.2 shows a Phase/Frequency offset block from the Communication ToolBox library which gives a frequency shift of 10 Hz to the incoming signal. The frequency-shifted signal is delayed using a delay block with a random fractional frame delay value ranging from 0 to 4. In the last step, an AWGN channel noise is added to the signal using an AWGN block from Communication Toolbox library. This way a real-world communication scenario in a noisy medium was modeled.

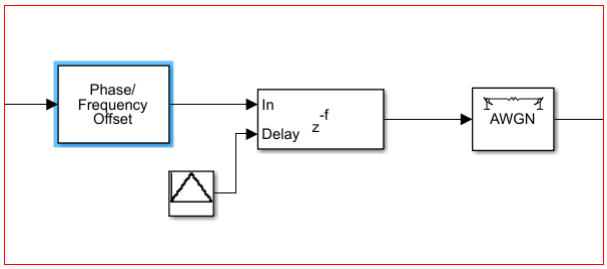


Figure 5.2: Channel simulink block diagram

5.2 Testing Results

5.2.1 Data reception at constant EbNo

A successful transmission of data packets was observed in the Simulink diagnostic window. The received packet contain 11 characters of "Hello World" appended with a repeated counter values from 000 to 099. The packet reception was successful for whole transceiver chain as can be seen in Figure 5.3.

5.2.2 BER vs EbNo Plot

A BER vs EbNo plot depicts the system performance in the noisy channel. It plots the number of error received at a specific EbNo value The overall system ber curve plot in Figure 5.4 shows the variation of simulated values (red line) to the theoretical ones (blue line). The difference is due to the known phase of the received signal in the theoretical calculation but in simulation, the signal phases are estimated. The signal phase estimation can have residual values which can worsen the BER. At low EbNo values, the AWGN noise is a dominant factor in having high BER. But at

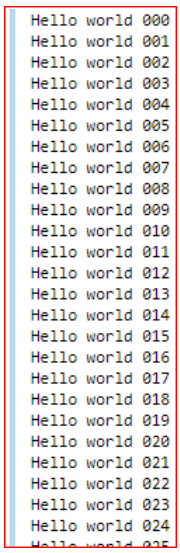


Figure 5.3: Demodulated packets at the receiver output

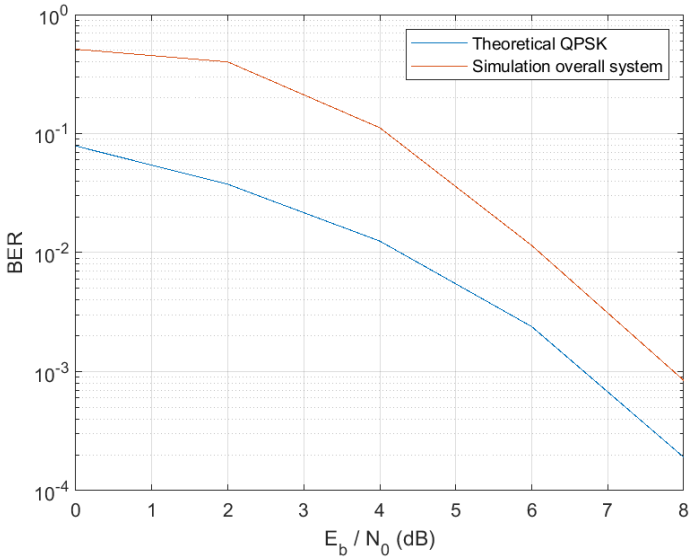


Figure 5.4: Overall system BER vs EbNo

the high EbNo values, the BER value difference to the reference value is less due to the lesser impact of the AWGN noise than the combined effect of sync errors at the receiver and ISI induced by the channel. A BER of 10^{-3} is sufficient for the LPWAN nodes as data is transmitted multiple times and environment variables did not change frequently so low BER can be tolerated. A BER of 10^{-3} means for every 1000 bits received, 1 bit will be corrupted. If we translate this BER to our scenario then it will equal to 1-bit error for every 5 packets received.

Chapter 6

Conclusion And Future Work

6.1 Conclusion

In this thesis, design and simulational verification was done for designing a reconfigurable SDR based LPWAN node. A one-chip solution consists of an RFSoC that is much too expensive for a low-cost solution. A two-chip solution based on an FPGA or a microcontroller and an RF transceiver chip was also not suitable because a substantial amount of resources is required (approx 30k LE) in case of an FPGA and a very high operation clock ($>$ 128 MHz) for a microcontroller-based design. The simulation results confirm that a reprogrammable node can be implemented on a standalone reconfigurable embedded three-chip platform consisting of an FPGA, a microcontroller, a sensor unit, and a power management unit. The FPGA logic resources required for transceiver filter implementation are approximately 8k LUTs. While the microcontroller minimum program memory should be 33 kilobytes and the data memory must be approximately 6 kilobytes. The latest microcontrollers have a sufficient number of interface general-purpose input/output (GPIO) required for data communication in sensor nodes.
The designed block diagram can be used for further development of the node PHY design. The model can be used for further testing of a different PHY or implementing an adaptable data rate design up to 600 kbps (upper limit of the sub-1GHz standard). The block diagram for the tentative layout of the node is shown in Figure 6.1.

The cost budget in Table 6.1 shows that the data processing part will cost approx-

Table 6.1: Cost and power budget for reconfigurable node

	FPGA	Microcontroller	RF IC	Total
Cost (€)	15.93	1.56	4.02	21.51
Current requirements (mA)	9.817 (TX) + 9.723 (Rx)	2.9	67	89.44

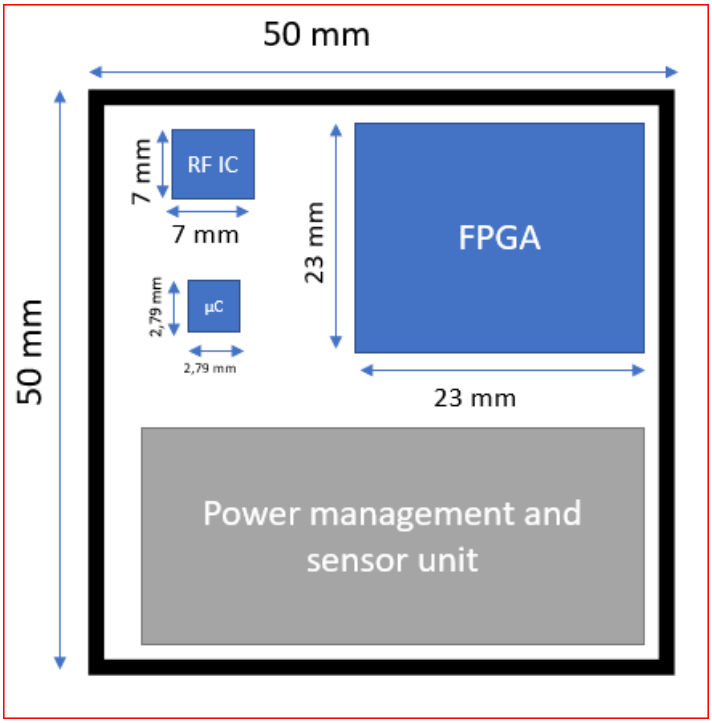


Figure 6.1: Sensor node layout

imately €22. The remaining €8 will be used for the power management unit, sensor unit, printed circuit board (PCB) manufacturing, and miscellaneous components. The pcb area constraint for the design implementation is in the limits as specified in the thesis scope. The power consumption for a three chip solution is lower than the other available options. To conclude, at this point reconfigurable node is feasible based upon the simulation results and scope defined.

6.2 Future work

Sample rate conversion and low-pass filtering the signal is the main bottleneck for system design, which can be tackled by implementing the CIC filters. The CIC filters are good at large sample rate conversion but the effectiveness was limited to 25 % of the passband frequency. A new class of filters can be investigated like CIC filters with sharpening techniques in multiple stages which can use fewer FPGA resources. When extra resources are available on an FPGA then the coordinate rotation digital computer (CORDIC) algorithm can also be implemented inside the FPGA, in order to replace multipliers with shifters, adders, and subtraction blocks.
The CIC decimator and interpolator are structurally the same for a given number of stages so a multiplexed system can be designed in an FPGA that uses the same CIC filter blocks for the decimation and the interpolation.

The scope of the thesis was limited to the narrowband signals but in the future maximum allowable data rates for LPWAN standard can be simulated or validated for the reprogrammable node.
In this study, power management unit for the reconfigurable node was not considered, power planning for such a node can be a good idea for a future research topic and finding ways to prolong node life. The feasibility of a secondary power source can be investigated like an energy harvesting technique. Energy harvesting techniques can make reconfigurable nodes to be batteryless.

Bibliography

[1] S. TABBANE. (2016) Iot network planning•. [Online]. Available: https://www.itu.int/en/ITU-D/Regional-Presence/AsiaPacific/SiteAssets/Pages/Events/2016/Dec-2016-IoT/IoTtraining/IoT%20network%20planning%20ST%2015122016.pdf

[2] *Wireless Sensor Networks Technology and protocol*. IntechOpen, 2012.

[3] S.-H. H. Rashmi Sharan Sinha, Yiqiao Wei, "A survey on lpwa technology: Lora and nb-iot."

[4] L. IOT. (2015) Which wireless network should you choose•. [Online]. Available: http://www.lprsiot.com/networks/

[5] D. M. Hernandez, G. Peralta, L. Manero, R. Gomez, J. Bilbao, and C. Zubia, "Energy and coverage study of lpwan schemes for industry 4.0," in *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, 2017, pp. 1–6.

[6] F. C. Kais Mekki, Eddy Bajic and F. Mayer, "A comparative study of lpwan technologies for large-scale iot."

[7] SIGFOX. (2019) Sigfox technology. [Online]. Available: https://www.sigfox.com/en/what-sigfox/technology

[8] AVENT. (2019) About sigfox. [Online]. Available: https://www.avnet.com/wps/portal/apac/products/c/lpwan/lpwan-nbiot-technology/about-sigfox?locale=zh_

[9] L. A. T. Committee, "Lorawan^{TM} 1.0.3 specification."

[10] Y. . E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi, "A primer on 3gpp narrowband internet of things," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 117–123, 2017.

[11] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[12] C. Buratti, A. Conti, D. Dardari, and R. Verdone, “An overview on wireless sensor networks technology and evolution,” *Sensors (Basel, Switzerland)*, vol. 9, pp. 6869–96, 09 2009.

[13] C. Cheng, C. K. Tse, and F. C. M. Lau, “A delay-aware data collection network structure for wireless sensor networks,” *IEEE Sensors Journal*, vol. 11, no. 3, pp. 699–710, 2011.

[14] A. Dongare, A. Luong, A. Balanuta, C. Hesling, K. Bhatia, B. Iannucci, S. Kumar, and A. Rowe, “Demo abstract: The openchirp low-power wide-area network and ecosystem,” in *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2018, pp. 138–139.

[15] T. Truong, H. Le, and T. Nguyen, “A reconfigurable hardware platform for low-power wide-area wireless sensor networks,” *Journal of Physics: Conference Series*, vol. 1432, p. 012068, 01 2020.

[16] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Computer Networks*, vol. 38, no. 4, pp. 393 – 422, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128601003024

[17] *Digital communication A Discrete-Time Approach*. Pearson International Edition, 2009.

[18] *Software Receiver Design*. Cambridge press, 2011, ch. 1.

[19] Wikipedia. (2020) Software-defined radio. [Online]. Available: https://en.wikipedia.org/wiki/Software-defined_radio#:~:text=Software%2Ddefined%20radio%20(SDR),personal%20computer%20or%20embedded%20system.

[20] *Software Defined R ware Defined Radio Handbook adio Handbook adio Handbook*. Pentek, Inc, 2011.

[21] nuand. (2020) bladerf x40. [Online]. Available: https://www.nuand.com/bladerf-1/

[22] ——. (2020) bladerf x40. [Online]. Available: https://www.nuand.com/product/bladerf-x40/

[23] LimeSDR. (2020) Limesdr-mini. [Online]. Available: https://wiki.myriadrf.org/LimeSDR-Mini

[24] Mouser. (2020) Adalam-pluto. [Online]. Available: https://nl.mouser.com/datasheet/2/609/ADALM-PLUTO-Product-Highlight-1633770.pdf

[25] K. Konkani, “Efficient reconfigurable architecture of baseband demodulator in sdr,” *International Journal of Research in Engineering and Technology*, vol. 3, no. 2, pp. 536–541, 2014. [Online]. Available: http://www.ijret.org

[26] W. G.Wong. (2019) Rfsoc delivers fpga flexibility with high-speed rf. [Online]. Available: https://www.electronicdesign.com/industrial-automation/article/21807610/rfsoc-delivers-fpga-flexibility-with-highspeed-rf

[27] Xilinx. (2020) Zynq ultrascale+ rfsoc. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/rfsoc.html

[28] *Atmel AT86RF215 Device Family*, ATmel.

[29] T. P. Truong, H. T. Le, and T. T. Nguyen, vol. 1432, p. 012068, jan 2020. [Online]. Available: https://doi.org/10.1088%2F1742-6596%2F1432%2F1%2F012068

[30] Xilinx. (2020) System on a chip. [Online]. Available: https://en.wikipedia.org/wiki/System_on_a_chip

[31] B. Moyer, “Chapter 5 - design considerations for multicore soc interconnections,” in *Real World Multicore Embedded Systems*. Oxford: Newnes, 2013, pp. 117 – 197. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124160187000018

[32] wikipedia. (2020) Field-programmable gate array. [Online]. Available: https://en.wikipedia.org/wiki/Field-programmable_gate_array

[33] ——. (2020) Microcontroller. [Online]. Available: https://en.wikipedia.org/wiki/Microcontroller

[34] R. K. Kodali, L. Boppana, and G. Gayathri, “Fpgas in software defined radio,” in *2013 15th International Conference on Advanced Computing Technologies (ICACT)*, 2013, pp. 1–6.

[35] Intel. (2020) Excerpts from ’a conversation with gordon moore: Moore’s law. [Online]. Available: http://large.stanford.edu/courses/2012/ph250/lee1/docs/Excepts_A_Conversation_with_Gordon_Moore.pdf

[36] B. Zeidman(EETimes). (2020) All about fpgas. [Online]. Available: https://www.eetimes.com/all-about-fpgas/

[37] *Tree-Based Heterogeneous FPGA Architectures*. Springer Science+Business Media New York 2012, 2012.

[38] I. The MathWorks, “Hdl optimized qpsk transmitter.” [Online]. Available: https://nl.mathworks.com/help/comm/ug/hdl-optimized-qpsk-transmitter.html

[39] *Digital Signal Processing Principles Algorithms and Applications*, 3rd ed. Prentice-Hall, 1996.

[40] “Pulse shaping.” [Online]. Available: https://en.wikipedia.org/wiki/Pulse_shaping

[41] I. The MathWorks, “Hdl optimized qpsk receiver.” [Online]. Available: https://nl.mathworks.com/help/comm/ug/hdl-optimized-qpsk-receiver-with-captured-data.html

[42] M. E. Frerking, *Digital Processing in communication systems*. Kluwer Academic Publishers, 2003.

[43] B. Sklar, *Digital Communications: Fundamentals and Applications*. USA: Prentice-Hall, Inc., 1988.

[44] *Synchcronization Techniques for Digital Receivers*. Plenium Press, 1997.

[45] E. Sourour, H. El-Ghoroury, and D. McNeill, “Frequency offset estimation and correction in the ieee 802.11a wlan,” in *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, vol. 7, 2004, pp. 4923–4927 Vol. 7.

[46] F. Salewski and S. Kowalewski, “Exploring the differences of fpgas and microcontrollers for their use in safety-critical embedded applications,” in *2006 International Symposium on Industrial Embedded Systems*, 2006, pp. 1–4.

[47] M. P. Donadio. (2020) Cic filter introduction. [Online]. Available: http://home.mit.bme.hu/~kollar/papers/cic.pdf

[48] keysight. (2020) Choosing a filter response type. [Online]. Available: http://literature.cdn.keysight.com/litweb/pdf/ads15/dfilter/df042.html

[49] S. Sharma, S. Kulkarni, V. M., and P. Lakshminarsimhan, “Hardware realization of modified cic filter for satellite communication,” in *2010 International Conference on Computational Intelligence and Communication Networks*, 2010, pp. 41–44.

[50] E. Hogenauer, “An economical class of digital filters for decimation and interpolation,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 2, pp. 155–162, 1981.

[51] G. Jovanovic Dolecek and F. Harris, “On design of two-stage cic compensation filter,” in *2009 IEEE International Symposium on Industrial Electronics*, 2009, pp. 903–908.

[52] A. Corporation. (2007) Understanding cic. [Online]. Available: Applicationnote455

[53] L. Cattaneo. (2016) Microsemi soc fpgas. [Online]. Available: https://indico.cern.ch/event/489996/contributions/2285681/attachments/1343728/2024822/InvitedSpeaker_Luca_Cattaneo.pdf

[54] "Eembc reveals benchmark results from low power microcontrollers." [Online]. Available: https://www.elektroniknet.de/elektronik/halbleiter/eembc-enthuellt-benchmark-ergebnisse-von-low-power-mikrocontrollern-113319.html