UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUÇÃO EM ENGENHARIA ELÉTRICA

FPGA-Based Testbed for Fronthaul Signal Compression: Implementation and Validation

Joary Paulo Wanzeler Fortuna

DM: 16/2017

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2016

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUÇÃO EM ENGENHARIA ELÉTRICA

Joary Paulo Wanzeler Fortuna

FPGA-Based Testbed for Fronthaul Signal Compression: Implementation and Validation

DM: 16/2017

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2016

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUÇÃO EM ENGENHARIA ELÉTRICA

Joary Paulo Wanzeler Fortuna

FPGA-Based Testbed for Fronthaul Signal Compression: Implementation and Validation

A dissertation submitted to the examination committee in the graduate department of Electrical Engineering at the Federal University of Pará in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering with emphasis in Telecommunications.

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2016

FPGA-Based Testbed for Fronthaul Signal Compression: Implementation and Validation

Dissertação de mestrado submetida à avaliação da banca examinadora aprovada pelo colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Pará e julgada adequada para obtenção do Grau de Mestre em Engenharia Elétrica na área de Telecomunicações.

Aprovada em _____ / _____ / _____

_____

Aldebaro Barreto da Rocha Klautau Júnior

ADVISER

_____

Leonardo Lira Ramalho

CO-ADVISER

_____

João Crisóstomo Weyl Albuquerque Costa

MEMBER OF EXAMINATION COMISSION

_____

Daniel Cardoso de Souza

MEMBER OF EXAMINATION COMISSION

Roberto Menezes Rodrigues

MEMBER OF EXAMINATION COMISSION

Evaldo Gonçalves Pelaes

DIRETOR DA PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Acknowledgments

The author would like to thanks firstly God for helping in the difficult moments. Then, his family for support and patience. Also, to his advisor and co-adivisor for all the hard work invested in this dissertation. Finally, thank to all the work team of LASSE for all the acquired experience.

Joary Paulo Wanzeler Fortuna

March 2017

”They did not know it was impossible so they did it”

Mark Twain

Acronyms

ADC	Analog to Digital Converter
AWG	Arbitrary Waveform Generator
AXI	Advanced Extensible Interface
BBU	Base Band Unit
BER	Bit Error Rate
BF	Basic Frame
BRAM	Block Random Access Memory
BS	Base Station
BW	Bandwidth
CAPEX	Capital Expenditures
CDMA	Code Division Multiple Access
CM	Configuration Management
CP	Cyclic Prefix
CPRI	Common Public Radio Interface
CPU	Central Processing Unit
C-RS	Cell-Specific Reference Signal
CW	Control Word
DAC	Digital to Analog Converter
DASH	Dynamic Adaptive Streaming over HTTP
DDR	Double Data Rate
DFT	Discrete Fourier Transform
DFTS	Discrete Fourier Transform-Spread
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access

DU	Digital Unit
EDGE	Enhanced Data Rates for GSM Evolution
EVM	Error Vector Magnitude
FFT	Fast Fourier Transform
FIFO	First In First Out
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
FTP	File Transfer Protocol
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HARQ	Hybrid Automatic Repeat Request
HDLC	High-Level Data Link Control
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
IP	Internet Protocol
LPC	Linear Predictive Coding
ISG	Industry Specification Group
ITU	International Telecommunications Union
JTAG	Joint Test Action Group
LSB	Least Significant Bit
LTE	Long Term Evolution
LUT	Lookup table
MAC	Media Access Control Layer
MIMO	Multiple Input Multiple Output
MMCM	Mixed-Mode Clock Manager
MSB	Most Significant Bit
NGFI	Next Generation Fronthaul Interface
NQ	Non-Quantized
OAM	Operation and Maintenance
OBSAI	Open Base Station Architecture Initiative
OFDM	Orthogonal Frequency-Division Multiplexing

OPEX	Operation Expenditures
ORI	Open Radio Equipment Interface
PBCH	Physical Broadcast Channel
PCFICH	Physical Control Format Indicator Channel
PDCCH	Physical Downlink Control Channel
PDSCH	Physical Downlink Shared Channel
PDV	Packet Delay Variation
PHICH	Physical Hybrid-ARQ Indicator Channel
PHY	Physical Layer
PLL	Phase Locked Loop
PTP	Precision Time Protocol
PUSQH	Predictive Uniform Scalar Quantization with Huffman
QAM	Quadrature Amplitude Modulation
RAN	Radio Access Network
RAT	Radio Access Technology
RB	Resource Block
RE	Resource Element
REC	Radio Equipment Controller
RF	Radio Frequency
RRU	Remote Radio Unit
RTC	Real-Time Clock
RTT	Round Trip Time
RU	Radio Unit
SMA	SubMiniature Version-A Cable
TCP	Transmission Control Protocol
TOC	Total Ownership Cost
TSN	Time Sensitive Network
UART	Universal Asynchronous Receiver/Transmitter
UFPA	Federal University of Pará
UMTS	Universal Mobile Telecommunications Service
VHDL	VHSIC Hardware Description Language
VSA	Vector Signal Analyzer

WCDMA	Wideband Code Division Multiple Access
WDM	Wavelength-Division Multiplexing
WIMAX	Worldwide Interoperability for Microwave Access

Symbols

N_{AxC}	Number of antennas carried over the Fronthaul
IQ_b	Number of bits per complex IQ sample
F_s	Sample rate of the LTE signal transported over the Fronthaul
R_{CPRI}	CPRI bit-rate used on the Fronthaul configuration
N_R	CPRI line-rate option used on the Fronthaul configuration
OV	Overhead generated by the Ethernet header and fronthaul Metadata on the CPRI stream
H_s	Number of bits allocated for the headers on each Ethernet frame
N_{BF}	Number of CPRI Basic Frames transported on each Ethernet frame
J	Maximum accumulation jitter created by Basic Frame accumulation
$T_{1,k}$	Timestamp acquired by PTP at the transmission of SYNC message
$T_{2,k}$	Timestamp acquired by PTP at the reception of SYNC message
$T_{3,k}$	Timestamp acquired by PTP at the transmission of DELAY_REQ message
$T_{4,k}$	Timestamp acquired by PTP at the reception of DELAY_REQ message
y	Calculated frequency offset error by PTP
x	Calculated time offset error by PTP
T_{sm}	One way delay from PTP slave to master
T_{ms}	One way delay from PTP master to slave
$\hat{d}$	One way delay estimation by PTP

List of Figures

List of Tables

Contents

Abstract

In recent years the world has seen an increasing demand for mobile services with high capacity and low cost. Such requirements forced the radio access technologies to be rethought. Among the different architectures proposed in literature, one that has got a lot of attention was the Centralized Radio Access Network (C-RAN). This strategy proposes to share the network resources through the centralization of the base-band processing and, as a result, reduce the network cost. Even though the centralization can bring several benefits, it also increases the distance between the point where the signal is captured and the point where it is processed. The link connecting both points is called fronthaul. In this scenario, the existing fronthaul technologies do not fit in the flexibility and cost requisites expected for the next generation mobile network, mainly due to the usage of dedicated optical links. One solution to these problems is the usage of Ethernet to transport fronthaul data, due to its ubiquitous presence, flexibility and low cost. In this work, a testbed for fronthaul based on Ethernet is presented, along with the implementation details and validation results. Also motivated by the Ethernet's bandwidth limitation, this work presents the VHDL implementation of a compression technique for LTE signals, evaluated in real transport conditions with the testbed. The obtained results indicate that it is possible to utilize the Ethernet network infrastructure for fronthaul transport. Although, it is necessary to reduce the requirements of Fronthaul stream through, for example, the application of signal compression techniques and synchronization methods.

Resumo

Nos últimos anos o mundo tem visto uma demanda crescente por serviços móveis de alta capacidade e baixo custo, isto tem forçado as tecnologias da rede de acesso móvel a serem repensadas. Entre as diferentes arquiteturas propostas na literatura, uma que ganhou bastente atenção é a centralização dos recursos da rede. Essa estratégia propõe compartilhar os recursos da rede através da centralização do processamento em banda-base, e como resultado reduzir o custo da rede. Apesar da centralização trazer vários benefícios, ela também aumenta a distancia entre o ponto onde os sinais são capiturados e o ponto onde são processados, o link que conecta esses dois pontos é chamado de fronthaul. Nesse cenário, as tecnologias de fronthaul existentes não são apropriadas para os requisitos de flexibilidade e custo esperados para a próxima geração de redes fronthaul, devido principalmente ao uso de links óticos dedicados. Uma solução para esses problemas é a utilização de Ethernet para transportar o trafego fronthaul, devido à sua onipresença, flexibilidade e baixo custo. Neste trabalho um testbed para fronthaul baseado em Ethernet é apresentado, juntamente com os detalhes de implementação e resultados de validação. Além disso, motivado pela limitação em banda existente no Ethernet, este trabalho apresenta a implementação em VHDL de um algorítmo de compressão de sinais LTE, avaliado em uma rede Ethernet real através do testbed. Os resultados obtidos indicam que é possivel utilizar a infraestrutura provida pela rede Ethernet no transporte de fronthaul. Por outro lado é necessário reduzir os requisitos exigidos pelo tráfego fronthaul. Através, por exemplo, da aplicação da compressão de sinais e de técnicas de sincronismo.

Chapter 1

Introduction and Motivation

One of the essential differences between mankind and other animals is the capability to communicate. From the Egypt's pictograms to the Internet, several techniques have been used to extend the human communication through the time and space. The Electrical Engineering field, in particular, has made several contributions to the communication systems. For example, the telegraph in early 1800's, the telephone in mid 1800's and the radio in late 1800's. But only in the late 1900's the mobile communication started to take the shape known nowadays, moved by the objective to communicate everywhere with everyone.

The first mobile transceivers were simple full-duplex analog radios, but soon became necessary to control the access to radio frequency resources. Then, a more complex network was created with the necessary control mechanism and analog voice transmission, called first generation mobile network (1G). Concurrently, the advances on voice coding and encryption made possible to transport voice over a digital network, which was applied in the second generation mobile network (2G). Also, this generation introduced beneffits such as voice encryption and data services, for example the SMS and MMS.

The evolution in capacity and coverage of the mobile networks was incremental, and the generations started to be defined by a set of prerequisites. For example, the IMT-2000 [5] and IMT-Advanced [6] defined the characteristics necessary for the 3rd and 4th generations, respectively. The first 3G systems appeared in 1998 with the UMTS (Universal Mobile Telecommunications Service) and CDMA2000 (Coded Division Multiple Access) technologies. Afterward, in 2008, the first 4G systems appeared bringing a set of new technologies, called Long Term Evolution (LTE) due to its extended compatibility for future technologies. Despite the mobile network has more than 7 billions mobile phones in 2015 [7], the market is still growing.

The predictions made by Cisco in [8] forecast an increase in the mobile data traffic by 7x between 2016 and 2021. Additionally, machine-to-machine communications will grow 34% per year in the same period, driven by the Internet of Things (IoT) adoption. Also, the media consumption will increase 9x, accounting as 78% of total mobile traffic in 2021. Furthermore, applications such as real-time video broadcast or tactile internet will require lower levels of network latency and delay. From the operator's perspective, others requirements are made such as flexibility on deployment, and cost reduction to maintain the Total Ownership Cost (TOC) relatively low.

Such requirements from users and operators shapes the target for the upcomming generation of mobile networks. To address all incoming requirements, the radio access network is evolving to use a variety of technologies. For example, Inter-Cell Interference Coordination (ICIC), massive MIMO (multiple-input, multiple-output), and operation of signals in the millimeter wave spectrum. In detail, the ICIC technology provides better signal quality and higher throughput on high interference scenarios. Also, Massive MIMO expands the capacity of the wireless link by the usage of several antennas. Finally, the operation in millimeter waves enables the transmission of higher bandwidth signals and provides more spectrum to be allocated. All these improvements being applied to the network require a great deal of engineering effort.

Due to the distributed characteristic of the existing mobile network, the implementation of interference coordination and spatial diversity between Base Stations are very expensive. In addition, the operation costs of cell-cites are high, mainly due to energy consumption and rent. In this scenario, the idea of a Centralized Radio Access Network (C-RAN) [9] gained attention, where a set of small and cheap remote distributed nodes are connected to a central processing unit. The cost reduction of this architecture can be around 53% in OPEX and 30% in CAPEX as shown in [10]. Furthermore, a centralized signal processing can make easier to implement the coordination and MIMO techniques expected for next generation mobile network.

Although the C-RAN brings several advantages, there are issues to be addressed in order to benefit from the centralization gains. The fronthaul is very important to the centralized RAN, since it connects the network units that capture and process the radio signals. But, the existing fronthaul technologies are very expensive, given that dedicated optical links are used to provide high bandwidth and low jitter. In this scenario, it is necessary to rethink the fronthaul infrastructure in order to reduce costs while still providing a reliable fronthaul transport.

One of the expected features of the next generation fronthaul is the usage of existing low

cost networks, such as Ethernet. This strategy should bring the benefits of statistical multiplexing provided by a packet-network, along with the re-usage of existing infrastructure. Also, the hierarchical structure of Ethernet is advantageous for dense deployment scenarios expected for 5G. In addition, the Ethernet protocols also provide a well known and broadly used set of Operation and Maintenance (OAM) functions. In summary, Ethernet is one of the main technologies in the next generation fronthaul.

This work presents a testbed implementation for Ethernet-Based fronthaul, along with the chalenges and solutions to provide reliable fronthaul transport over Ethernet infrastructure. The fronthaul implementation shown here could be used to test and evaluate a variety of algorithms for fronthaul and C-RAN. For example, this infrastructure was used to evaluate the LTE signal compression of [11] and the transport of compressed IQ signals. In this evaluation, the compressed data is transported over the fronthaul network, decompressed and sent to the air interface through the usage of an external digital-to-analog converter. In addition the analog signal is fed into a signal analyser in order to evaluate the quality metrics of the signal.

1.1 Objectives

Some of the main objectives of this work are:

- To review the basics of existing fronthaul technologies and future trends.
- To present an implementation of fronthaul architecture based on Ethernet.
- To evaluate the impairments of an Ethernet-Based fronthaul into the transported signal.
- To present an implementation of LTE signal compression algorithm in VHDL.
- To evaluate the quality metrics of signal compression over an Ethernet-Based fronthaul.

1.2 Dissertation Outline

Chapter 2 shows an overview of the fronthaul technology, the existing standards, the future trends and current solutions. Initially, a review of the existing fronthaul technologies is shown in Section 2.1. Then, the expected features for the next generation fronthaul are presented

in Section 2.2. Finally, an overview of the signal compression technique implemented in this work is shown in Section 2.3.

Chapter 3 presents the implemented Ethernet-based fronthaul testbed with compression. Initially, the used fronthaul architecture is shown in Section 3.1. After, the VHDL implementation of the compression algorithm is detailed in Section 3.2, along with the developed blocks and its engineering solutions.

Chapter 4 exposes the results collected from the testbed, the evaluated network parameters and the quality aspects of the reconstructed signal after fronthaul transport and decompression.

Chapter 5, contains the final considerations of this work, as long as suggestions future works.

Chapter 2

Fronthaul

Fronthaul is the name given for the portion of the Radio Access Network (RAN) between digital signal processing and Radio Frequency (RF) signal conditioning. In a high-level view the RAN architecture can be divided in two blocks: The Digital Unit (DU) that modulates the user data into a digital signal according to a specific Radio Access Technology (RAT) and the Radio Unit (RU) which converts the digital signal into analog RF signal. In the existing Base Stations (BS) these two units are connected through a high reliable and fast channel called fronthaul, as show in scenario (b) of Figure 2.1. Over the time the architectural usage of the DU and BU changed as shown in Figure 2.1 and further discussed in this section.

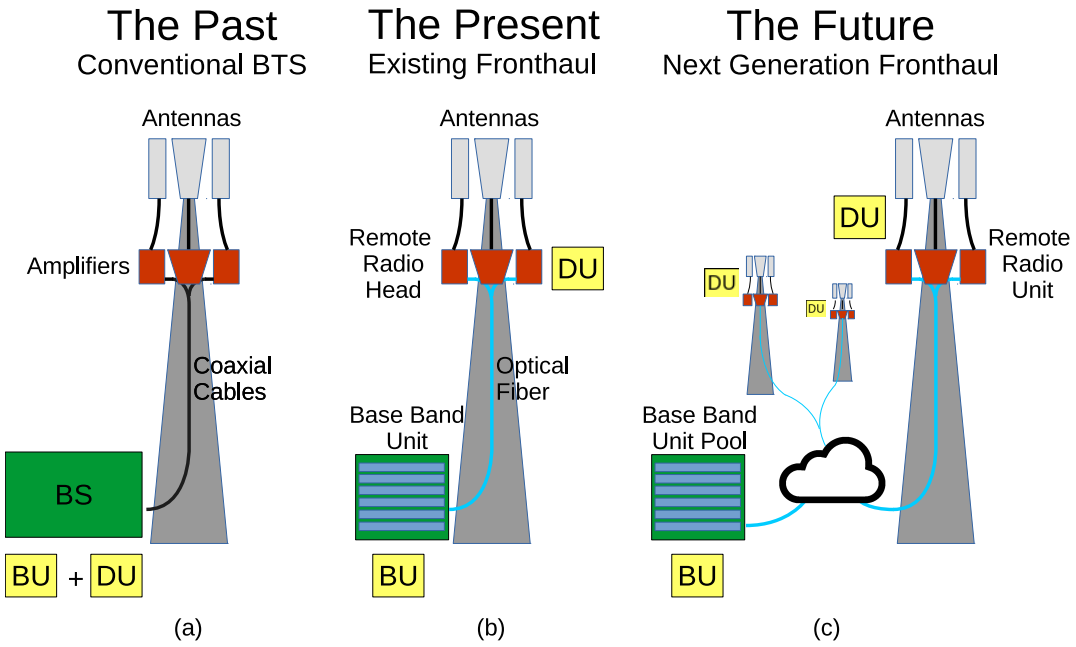


Figure 2.1: The fronthaul evolution.

The split of DU and RU is a well known practice in the telecommunications world. This paradigm is preserved even when both units are physically located at the same space, as shown in Figure 2.1 (a). In this scenario, the concept of fronthaul already existed, but just around 2002 it started to be standardized as a long range digital link. This transformation was motivated by the necessity to increase the mobile coverage and extended the Base Station through Remote Radio Head (RRH), as shown in Figure 2.1 (b). At that time several telecommunication actors started to define new types of interfaces between both units. The efforts pointed in the same direction, a digital interface using optical-fiber medium with high availability and throughput.

Different protocols were defined for the implementation and deployment of fronthaul links, some of the most important ones are: the Common Public Radio Interface (CPRI) [1], developed by group of companies including: Ericsson, Huawei, NEC, Alcatel, Nokia and others. Also, there were the Open Base Station Architecture Initiative (OBSAI) [2], created by another groups of companies including: Samsung, LG, Nokia, Hyunday and ZTE. Additionally, there is a recent effort based on CPRI to provide more interoperability between different RU's and DU's called Open Radio equipment Interface (ORI) [12].

Currently the RAN architecture is going through another round of architectural changes to provide better services at lower costs. These changes are motivated by the high demands for throughput and quality from the consumers, allied with the mobile operators competition for lower costs. In this scenario, the concept of Centralized Radio Access Network (C-RAN) [10] comes up as a solution for cost reduction. As discussed before, this concept proposes to centralize the DU's functions into computing centers, while keep the RU at the coverage areas, as shown in Figure 2.1 (c). This architecture has many advantages such as cost-reduction on deployment and operation, also it is possible to implement interference coordination algorithms easily with centralized processing.

In order to centralize the resources the fronthaul network needs to be redesigned [13] to support the C-RAN at reasonable costs. The existing fronthaul uses point-to-point optical links, which makes the cost of deployment on dense scenario very high. A fronthaul architecture suitable for C-RAN should benefit from the statistical multiplexing provided by packet networks. Also, take in account the network traffic on the network scheduling, providing dynamic flow-control based on the network usage.

Additionally, the network should provide the flexibility for different deployment topologies or even dynamic topology. To provide suitable fronthaul architecture it is possible to ex-

plore the functional split between DU and RU. This means to move some processing from the Base Band Unit to the Remote Radio Head. In this case the RU receives a difference nomenclature, namely Remote Radio Unit (RRU), due to its additional digital processing.

The existing fronthaul transports raw complex samples and uses huge amounts of bandwidth. For example, a single 4G-LTE 20 MHz signal can consume up to 1.2 Gbps, assuming the signal is using 40 bits on each complex sample, which is the maximum defined by CPRI in Section 7 of [1]. These bandwidth requirements are unfeasible for the network densification planned in 5G networks, especially in applications that use, e.g., massive MIMO [14], since the fronthaul bandwidth is proportional to the number of antennas used in the RU.

In this chapter a review of the essential characteristics of existing fronthaul technologies is shown in Section 2.1. Then, the research trends of the Next Generation Fronthaul Interface (NGFI) is presented in Section 2.2. Finally, to support the subsequent chapters a brief introduction to the compression algorithm implemented in this work in shown in Section 2.3.

2.1 Fronthaul Concepts and Existing Technologies

As introduced before, there are three well-defined fronthaul specifications: CPRI, OBSAI and ORI [15]. Besides having different objectives and procedures, they also have a lot in common when focusing in the fronthaul link itself. For example, both OBSAI and CPRI carry synchronization signals through the fronthaul. Additionally, they all define control and management procedures.

The Common Public Radio Interface (CPRI) specification is an industry cooperation started at 2003 aiming to define a fronthaul link between Radio Equipment (RE) and Radio Equipment Controllers (REC). The CPRI is a essentially simple protocol to transport IQ data, control and synchronization information from the RE to REC. The CPRI is the most widely used fronthaul standard, this specification will be better discussed in Section 2.1.1.

The Open Base Station Architecture Initiative (OBSAI) standard started in 2002 aiming to define a complete architecture for base stations, as will be discussed in Section 2.1.2. The objective of this specification is to standardize the internal blocks of the base station and enable the cell-site to be deployed with equipment from different vendors. The fronthaul as this work defines is actually one of the many interfaces specified by OBSAI standard.

The Open Radio equipment Interface (ORI) standard is relatively recent. It started in 2010

by the Industry Specification Group (ISG). This standard is based on CPRI and aims to specify a more detailed Control and Management Procedure. In CPRI most of these operations are vendor-specific, which makes it difficult of using products from different vendors. Furthermore, the ORI specification defines a compression algorithm to be applied on CPRI signals with large bandwidth [12]. The ORI will be briefly discussed in Section 2.1.3

2.1.1 Common Public Radio Interface (CPRI)

The CPRI specification is the commonly used standard of fronthaul. It defines a communication link between the DU and RU, in fact these entities are called respectively Radio Equipment Controller (REC) and Radio Equipment (RE) in CPRI specification. The link carries not only the complex signal (IQ Data), but also control and clock synchronization information, as shown in Figure 2.2. The CPRI payload can carry several IQ signals, each one encapsulated into an Antenna Carrier (AxC). It can transport different communication protocols, such as GSM, CDMA, LTE or WiMAX. Also, the protocol counts with a set of control information such as Control and Management (C&M) data, protocol extensions and vendor specific information [1].

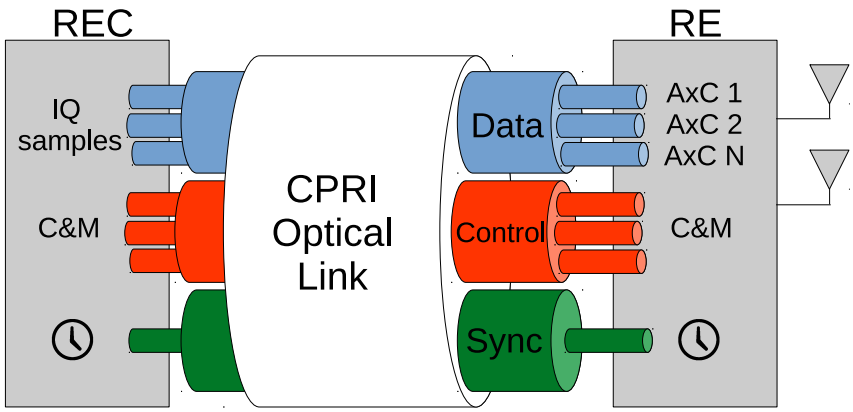


Figure 2.2: Information carried in CPRI link.

The CPRI link carries frequency and phase synchronization information. Actually this information is provided by the physical layer (PHY). In fact, the optical or electrical interfaces can use Phase Locked Loops (PLLs) to derive a synchronization signal based in the received bit-stream. Such implementation is common in modem applications [16] and is very precise in optical communication due to the low distortion provided by optical media. Also, in CPRI it is possible to measure and compensate the delay on the physical medium, since the topology is very simple. Moreover, the CPRI has control information to indicate the framing synchroniza-

tion. After all the synchronization procedures the CPRI link is capable of achieving a maximum frequency error of 0.002 ppm, and delay accuracy of 16.276 ns [1].

The CPRI specification defines both Optical and Electrical media but it does not describe how the physical layer should be implemented. In fact, the standard just specifies the performance requirements of the PHY such as the Bit Error Rate (BER) under 10^{-12} and the link line-rates, which are listed in Table 2.1. Although the specification defines an electrical interface the existing CPRI deployments for long distances are made with optical medium. The Electrical interface is commonly used in backplane implementations or short distances cooper cabling, under 10 m [1].

Table 2.1: CPRI configuration for some of the line rates defined in the specification.

Option	# of Bytes forming a Word	Bitrate (Mbps)	Line Coding	Line Rate (Gbps)
1	1	1 x 491.52	10/8	0.61
2	2	2 x 491.52	10/8	1.22
3	4	4 x 491.52	10/8	2.45
4	5	5 x 491.52	10/8	3.07
5	8	8 x 491.52	10/8	4.91
6	10	10 x 491.52	10/8	6.14
7	16	16 x 491.52	10/8	9.83
8	20	20 x 491.52	66/64	10.13
9	24	24 x 491.52	66/64	12.16
10	48	48 x 491.52	66/64	24.33

The smallest portion of a CPRI stream is the Basic Frame (BF), it has a unique characteristic, independent of the line rate used the BF has to depart at the fixed rate of 3.84 MHz, this rate is called chip-rate and is a legacy from CDMA applications. In other words, independent of the CPRI line rate, at each 260.41 ns there is a BF departing. The BF is divided internally into 16 words, as shown in Figure 2.3, the bit-width of these words changes with the line-rate option. For example, in the line-rate option 1 the bit-width of each word is 8 bits, while in the line-rate option 2 each word carries 16 bits. Also, the first word of each BF is reserved for CPRI control information, namely the Control Word (CW).

The BF IQ Data block is divided internally into Antenna Carrier Containers (AxC) which

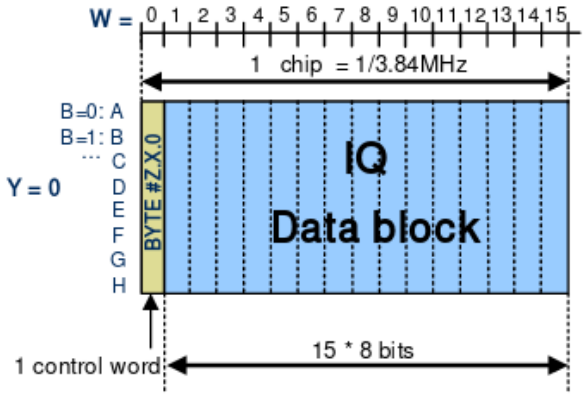


Figure 2.3: CPRI Basic Frame internal structure [1].

form a block of samples from the same antenna as ilustrated in Figure 2.4. The BF can carry several AxC's, organized in ascending order, as long as the link capacity is respected. The number of samples in each AxC container depends on the oversampling ratio with respect to the chip-rate (3.84 MHz). Inside each AxC, the samples are stored in chronological order and consecutively, from LSB to MSB, with interleaved I and Q samples, as shown in the Figure 2.4.

Both Figures 2.3 and 2.4 shows the same CPRI BF setup. They both have a control word of 8 bits followed by a data block of 120 bits, divided in 15 words of 8 bits. Note that the IQ samples of Figure 2.3 does not necessarily use an integer number of words from the data block. In fact, the data block is divided for each antenna carrier and the samples are allocated inside the AxC accordingly to the transported signal.

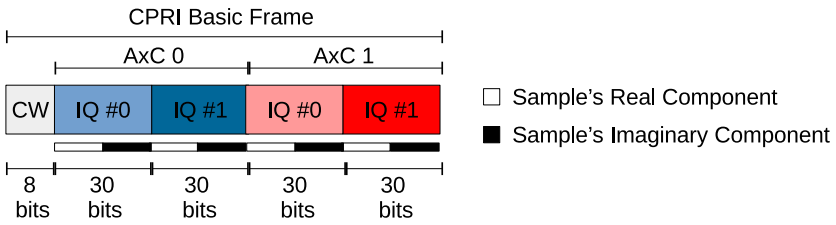


Figure 2.4: Example of CPRI Basic Frame carrying two signals with 30 bits per IQ sample.

The CPRI BFs are grouped in hyperframes, each one contains 256 BF and takes 66.67 μs of transmission time. Additionally, the hyperframe is grouped into a CPRI Frame, containing 150 hyperframes and takes 10 ms of transmission time. There is a nomenclature used to indicate the position of each bit in the CPRI stream. Such convention is represented by a tuple (Z.X.W.Y.B), where Z indicates one of the 150 hyperframes, X indicates one of the 256 Basic

Frames, W indicates one of 16 words within the Basic Frame, Y indicates the byte within a word and B indicates the bit in each byte, as shown in Figure 2.5.

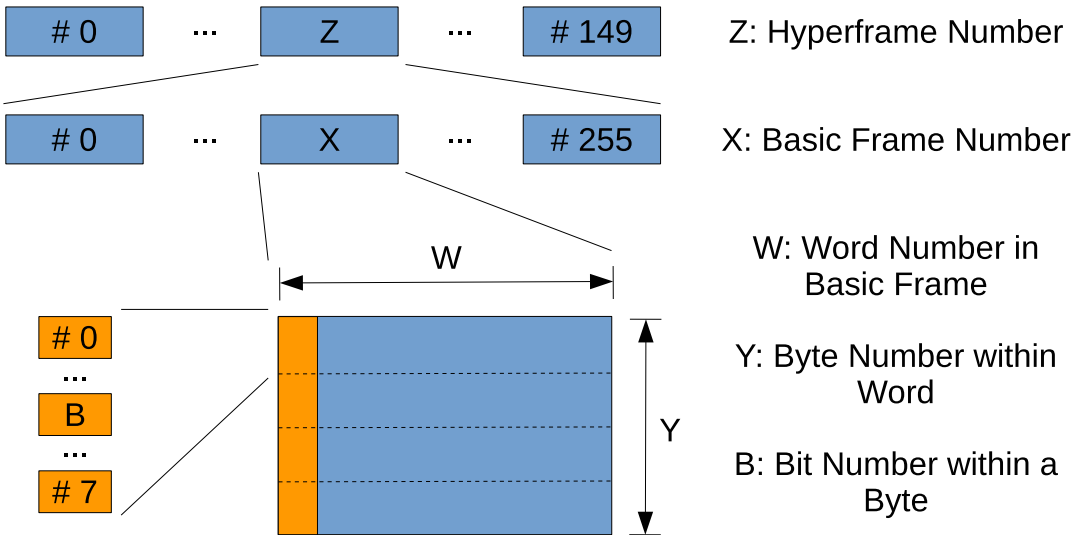


Figure 2.5: CPRI naming convention and frame structure [1].

The control words are grouped by hyperframes, each one has 256 Control Words which carries different types of control information. For example, a hyperframe Synchronization word is carried in the first word of each hyperframe. Also, some words are used to allow the communication between the PHY layers of endpoints, called L1 Inband Protocol. In addition, there are words carrying special sequences to help the PHY layer to train is internal PLL's in order to acquire the synchronization. Aside from these fields, there is also space for vendor specific information and others.

The amount of bits to be carried in each BF is dependent on the number of AxC containers (N_{AxC}), the amount of bits per IQ-sample (IQ_b) and the sample rate (F_s) of each AxC signal. As a matter of fact, the CPRI bit-rate (R_{CPRI}) has to be chosen based in the equation:

$$\sum_{i=0}^{N_{\text{AxC}}} IQ_{b,i} \times F_{s,i} \leq \frac{15}{16} \times R_{\text{CPRI}}, \tag{2.1}$$

where the $15/16$ value represents the Control Word overhead. For example, a CPRI stream with two AxC ($N_{AxC} = 2$), carrying an LTE 20 MHz signal ($F_s = 30.72 \times 10^6$), with IQ samples represented by 30 bits ($IQ_b = 30$), consumes a bit-rate around 1.84 Gbps. Although the closest bit-rate existent, as shown in Table 2.1, is 1.96 Gbps (4×491.52) which is relative to the usage of 4 bytes per word.

Although CPRI is the commonly used protocol, it has a very small protocol defition. On one hand such simplicity makes the standard very flexible and can be applied in different scenarios, on the other it became difficult to assure the interoperability of equipment based only in CPRI specification.

2.1.2 Open Base Station Architecture Initiative (OBSAI)

As discussed before, the OBSAI standard defines a whole architecture for the Base Station, including a fronthaul link [2]. The specification defines several basic blocks and interfaces, as shown in Figure 2.6. For example, the *Base Band Block* modulates the information into the IQ signal, while the *RF Block* converts the IQ signal to analog RF. Moreover, the *Control and Clock Block* generates clock synchronization and control information to all the blocks.

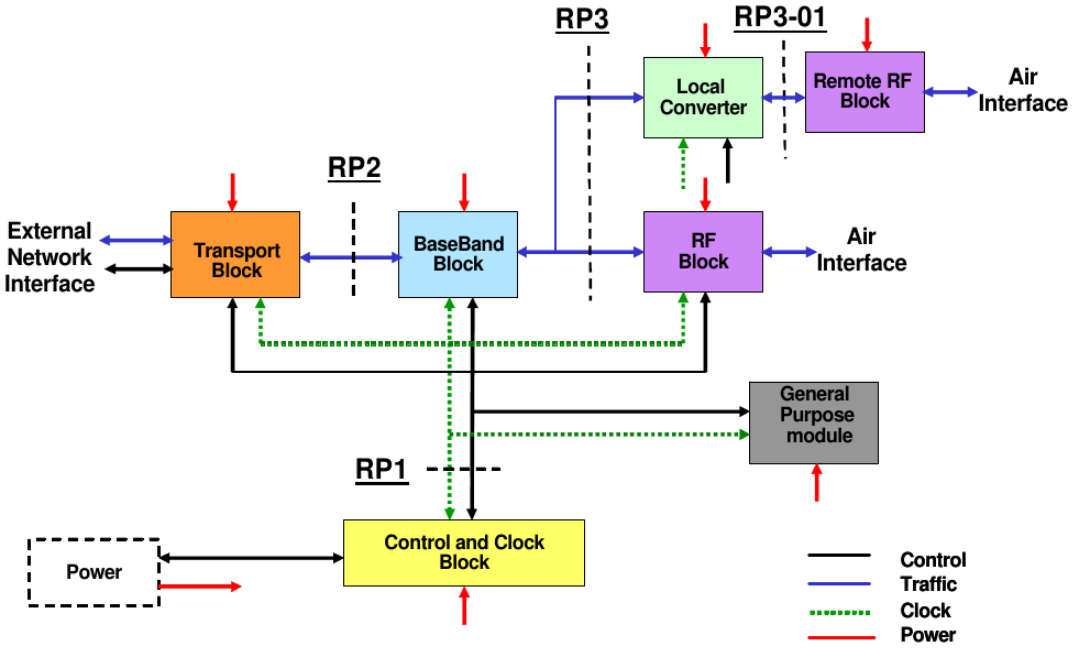


Figure 2.6: Overview of OBSAI building blocks [2].

The protocol [17] also specifies the interfaces between the blocks: the Reference Point 1 (RP1) distributes synchronization and control information between the blocks. In addition, the Reference Point 2 (RP2) interfaces the *Base Band Block* with *Transport Block* transporting the user data. Also, there is the Reference Point 3 (RP3) transports the samples between *Base Band* and *RF Blocks*.

In contrast to the CPRI specification, in OBSAI the IQ data, Control and Synchronization

are provided by three different entities. To extend the range of a divided architecture like this it is necessary a converter to group these information into a single channel. In this scenario, the OBSAI standard defines an extension to the RP3, called RP3-01, to make possible the utilization of Remote Radio Units. On RP3-01 the IQ-Data, Control and Synchronization are carried together into a single long-range link. In this extension the network topology is restricted to chain, ring or tree, just like CPRI. Although the optical link is more commonly used, the RP3-01 also specifies other media such as wireless and cooper cable, as long as the transmission characteristics can met the standard [2].

There are many similarities between OBSAI RP3-01 and CPRI, as shown in Figure 2.7. For example, both are capable to transport the same Radio Access Technologies: GSM/EDGE, CDMA, WCDMA, WIMAX and LTE. Also, both use a reference signal derived from physical layer to provide system-wide clock synchronization. In addition, both standards define optical and electrical interfaces. The former is applied in long-range applications while the latter is used in backplane applications.

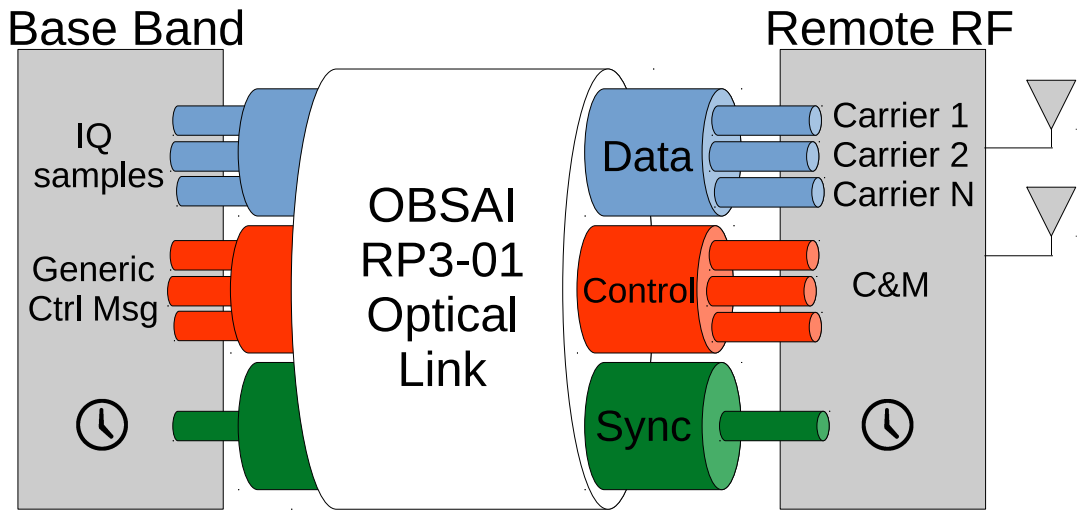


Figure 2.7: Information carried by OBSAI RP3 link.

Although the RP3-01 link actually represents the long-rage fronthaul, the characteristics of RP3 are very interesting. For example, IQ traffic is packet-based, so it is possible to use packet switching in hardware to increase efficiency. In addition, the topology of RP3 network is more complex, enabling the usage of mesh or centralized modes. Moreover, the *Base Band Block* could have access to any *RF Block* on the network making possible the cooperation between them.

The OBSAI is a very extensive standard, since it describes the complete architecture of a BS. For example, the C&M procedures for RP3-01 are more complex than the CPRI ones. This reduces the implementation freedom in the fronthaul, but allows the compatibility between equipment. The detailed study of OBSAI specification is out of scope of this work, but it is essential to evaluate the similarities and differences among the available fronthaul techonologies.

2.1.3 Open Radio Equipment Interface (ORI)

The ORI protocol [12] is fully compliant with CPRI specification. In fact it can be seen as an extension to the CPRI specification to provide more interoperability between equipment from different vendors. Among the extensions defined by ORI, the compression of IQ data is an important topic. The standard specifies how the compression should be made and also the allowable degradation such as EVM and latency. Also, in the control and management domain the specification defines software abstractions and configurations not covered by CPRI.

Besides it is possible to compress the stream transported by CPRI, which does not specifies how the signal can be compressed. The ORI, on the other hand, defines a compression technique to be used for the LTE signals with BW higher than 10 MHz. The specified algorithm uses a 3/4 down-sampling and non-linear quantization to remove the redundancy in both frequency and amplitude domains. The protocol also defines how the compressed stream should be mapped in the IQ-data space. Despite the fact that down-sampling provides a 3/4 compression ratio, the final compression ratio is not defined in the specification, since it depends on the statistical characteristics of the signal and the number of bits used in the non-linear quantization [12].

The control and managements features of ORI are much more complete than CPRI. The REC counts with a resource model to store the information of the Radio Equipment (RE). In fact, the REC interfaces with the resource model creating an abstraction layer between the nodes. The resource model provides functionalities to manage the Configuration (CM), State (SM) and Faul (FM) of each RE. With this software abstraction, it is possible to make changes in the RE without directly changing the REC.

In the CPRI specification, the 256 Control Words are aggregated in groups of four, called control sub-channels. The purpose of each one of the 64 sub-channels is predefined in the CPRI specification, as show in Figure 2.8. The ORI specification respects CPRI, but the vendor-

specific and Fast C&M sub-channels are remapped to allocate ORI data. The vendor-specific area is reduced and gives place to an ORI reserved area, as shown in Figure 2.8. Also, the CPRI's Ethernet data-link protocol defined in Fast C&M became a full TCP/IP stack protocol in ORI C&M. The usage of a full stack makes possible to use other well-known protocols in the ORI link. For example, it is possible to use DHCP for IP address distribution and FTP for configuration file transfer.

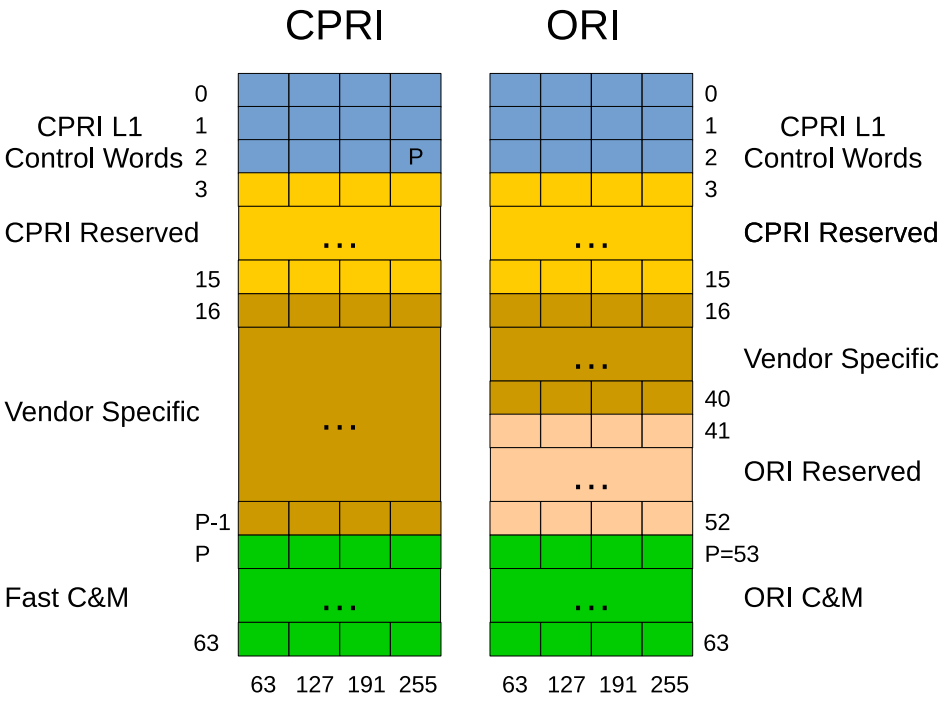


Figure 2.8: CPRI and ORI control sub-channels mapping.

As seen in Figure 2.8, the vendor specific fields are not removed from ORI, but they use a reduced space. In CPRI the vendor specific area had a configurable amount of words from 15 to 192. However, in ORI these fields uses 100 words while the rest of the Control Words are divided between ORI Reserved fields and ORI C&M link, the former uses 48 words while the latter uses 44 words.

2.2 Next Generation Fronthaul Interface (NGFI)

The existing fronthaul technologies meet the needs of the last generation RAN architecture. Although for the 5th generation of mobile communications, it is still necessary some effort to adapt or re-create the technologies to meet the requirements of capacity and cost. The main

objective of the fronthaul in the next generation is to enable the radio access network centralization and reduce the costs of network deployment. Of course, the centralization itself can bring several benefits such as resource re-utilization, virtualization and better spectrum usage.

The traffic demand shaped the evolution of physical layer modulation and coding techniques. As a result, the single-link capacity is very close to the Shannon's limit. On the other hand, the usage of multi-antenna techniques multiplied the overall link capacity, which generates the necessity of Base Stations with even more antennas. In addition, the signal interference once seen as a problem to be fought, now can be used constructively to increase the signal strength. In this scenario, the fronthaul has to connect several coordinated Radio Units to the central BS processing.

As shown previously, the existing fronthaul distributes the IQ samples, control and synchronization information, thus the NGFIs should be capable to transport the same information with lower costs. Between the difficulties to produce such network is the prohibitive amount of bandwidth used in the existing fronthaul. As already discussed, an LTE signal can consume it to 1.2 Gbps from the fronthaul. Obviously such bit-rate values can be decreased by reducing the number of bits used to represent each sample, but this will deteriorate the signal quality.

Moreover, the usage of dedicated optical links to transport the fronthaul information increases the overall cost of the network. Although the Wavelength-Division Multiplex (WDM) technology can be applied to aggregate several optical channels into a single optical fiber, the equipment cost is still high. On the other hand, the widely used Ethernet equipment provides a low-cost industry standard solution. In addition, the usage of packet-based statistical multiplexing promises gains to the network efficiency. Also, Ethernet is very simple to deploy and scale, which makes the Ethernet transport a desired solution for NGFI. In addition the Ethernet protocols already have a complete set of OAM functions for troubleshooting and management

In the following sections a set of characteristics desired for the NGFI will be highlighted. For example, the benefits and difficulties to use a packet-based fronthaul transport is shown in Section 2.2.1. Distinct functional splits are discussed in Section 2.2.2 which are used to lower the fronthaul transmission rate by move some functions from BBU to RRU. And finally, Section 2.3 shows how signal compression can reduce the bit rate requirement of the fronthaul.

2.2.1 Packet-Based Fronthaul

As discussed before, the existing fronthaul uses dedicated optical links due to the strict requirements of the fronthaul specification. However, this deployment is inefficient from the perspective of network utilization. For example, at night when the network usage decreases, the fronthaul link has to be active to provide the coverage, even if there is no user to be covered. Due to its static configuration, the existing fronthaul can not benefit from the tidal wave effect of network usage, which is the cyclic variation of the network load over the day.

On the other hand, packet-based networks such as Ethernet are very efficient to accommodate on network usage changes, due to the packet switching methodology. As said before [18], both industry and operators agree that the NGFI should be packet-based. In fact, this paradigm change would bring more flexibility to the fronthaul, although it is necesary to evaluate the challenges generated by the packet network.

There are two ways to think the adaptation of the fronthaul to packet-based network. The first consists to evolve the Ethernet infrastructure to provide the fronthaul requirements. Another is to change the base station architecture to take in account the packet-based impairments. There are several groups discussing which techniques should be used and the optimal solution, such as the IEEE-1914 [19] and IEEE-1904.3 [20].

Of course, the existing Ethernet infrastructure is not reliable: the best-effort characteristic forces the higher layer protocols to have extra procedures to assure the information transport. On the other hand, there is work being done in the IEEE Time-Senstive Network (TSN) task force to assure the quality of service for streams being transported over the Ethernet. The aforementioned task force specifies some techniques to assure the Ethernet quality such as: Scheduled Traffic [21], Frame Preemption [22] and Link Reservation [23].

The Ethernet standard is evolving to provide transport with less Packet Delay Variation (PDV) and Round Trip Time (RTT) while maintaining the high throughput. On the other hand, fronthaul requirements are very tight. A transport network to provide such parameters will eventually not be cost effective. In this scenario, the fronthaul has to introduce techniques to overcome the Ethernet impairments, and provide the fronthaul transport even on imperfect Ethernet network.

The packet delay variations can be reduced using buffers to store samples before forwarding to the next node. In fact, this technique reduces the PDV by introducing delay, relative to buffering time. Such technique can be applied as soon as the delay budget is met. Also, the

RTT requirements for fronthaul can be relaxed by rethinking the time budget over the communication stack. Since on centralized architecture the upper layers of the stack will use less time to communicate inside a Radio Computing Center.

In addition, the signal compression techniques shall evolve to provide more signal quality in bandwidth limited scenarios. Also, the evolution of transport protocols of fronthaul over Ethernet should provide more robustness to the network impairments, such as packet drop and network congestion. In this scenario, a desirable feature is the adaptation of the compression ratio to the network load state.

Although the centralization shall facilitate the implementation of MIMO and interference mitigation/coordination techniques, these applications require frequency, phase and time synchronization between the network nodes. In such circumstances, it is possible to use the Precision Time Protocol (PTP) [24] to provide the required synchronization. The PTP is a network protocol defined by IEEE-1588 and used in packet-based networks to provide clock synchronization.

Aside from PTP it is possible to use different synchronization sources over Ethernet fronthaul. For example, the Global Navigation Satellite Systems (GNSS) which is an effective solution to synchronize outdoor nodes. Moreover, it is possible to distribute synchronization through the physical layer as implemented in the ITU G.8262 specification, called SyncE [25]. This solution requires an special implementation of the Ethernet physical layer, since the Ethernet is intrinsically asynchronous.

The discussed impairments and solutions does not require the Radio Access Technology to be upgraded. On the other hand, by adaptations to the RAT it is possible to achieve better results in the centralization over a packet-based fronthaul. This scenario will be discussed in the next Section.

2.2.2 Functional Split between BBU and RRU

Beyond the search to provide a highly reliable packet-based fronthaul network, there are also other ways to provide the fronthaul service based on existing Ethernet infrastructure. For example, it is possible to rethink the split of functionalities between the BBU and RRU. In other words, offload some BBU functionalities to the RRU. This strategy can relax the requirements from the fronthaul while still maintain some centralization gain. There is a clear trade-off between centralization gain and fronthaul requirements [26].

There are different scenarios for functional split, for example the scenario implemented by CPRI and OBSAI is the full-centralization, shown in Figure 2.9 as (a). In this scenario, the RRU just contains the analog converters and RF conditioning circuitry, in other words all the signal processing and upper layers are executed in the BBU. As a result, this scenario demands high throughtput and low latency from the fronthaul. For example, in CPRI the maximum RTT is 5 μs, while the bandwidth for an LTE signal 20 MHz with 2x2 MIMO can be up to 1.8 Gbps, as shown in Table 2.2.

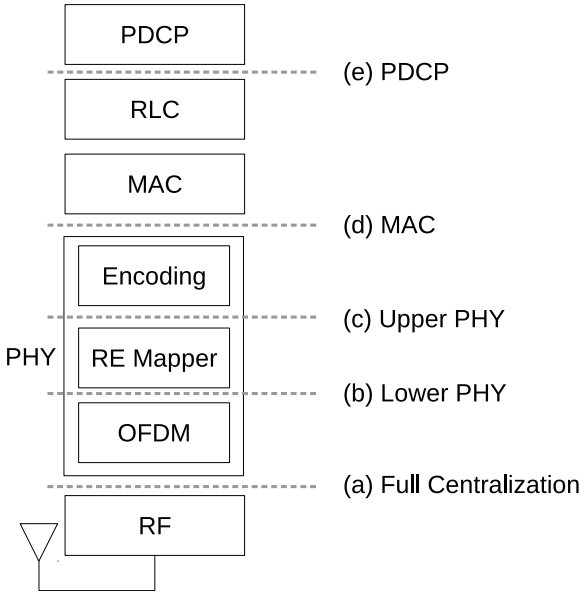


Figure 2.9: LTE base station possible functional splits: (a) All process centralized in the BBU, (b) The OFDM modulation is executed in RRU, (c) The resource mapping and modulation is executed in RRU, (d) The whole PHY layer is executed in RRU, (e) All functions except the PDCP is executed in the RRU.

Table 2.2: Fronthaul requirements of different functional split from [3].

Scenario	BW*	RTT	Latency
(a)	1.8 Gbps	5 μs	150 μs
(b)	590 Mbps	1 ms	150 μs
(c)	86 - 530 Mbps[1]	3 ms	150 μs
(d)	70 Mbps[1]	3 ms	2 ms
(e)	70 Mbps[1]	100 ms	5 ms

* Based on an LTE 20 MHz 2x2 MIMO signal

[1] Depends on LTE Network Load

The full-centralization scenario can provide complete centralization gains. On the other hand, the fronthaul necessary to accomplish this objective has very high requirements, which increase the overall cost of the fronthaul network. Not only the necessary bandwidth is very high but also the RTT delay and latency are very low, which requires a big effort to encapsulate this type of network on the existing Ethernet infrastructure.

Other functional splits are also possible, for example to offload the OFDM processing into the RRU, as represented in scenario (b) of Figure 2.9. In this scenario, just the frequency domain IQ samples are transported over the fronthaul, while the OFDM processing including the Cyclic Prefix insertion are executed in the RRU. This type of split can reduce the fronthaul bandwidth up to 30%, as shown in [3]. Also, the RTT delays are relaxed due to the LTE framing structure. On the other hand it is still attached to the LTE slot boundaries.

The Lower PHY scenario reduces the bandwidth requirements from the fronthaul, and is still capable to centralize the MIMO precoding operations. In addition, the interference coordination technique can also be applied in the frequency domain before the OFDM modulation. On the other hand, this scenario still demands very tight latency requirements, which makes difficult the usage of several hops in a Ethernet network.

Another important split is the scenario (c) in Figure 2.9, called Upper PHY. In this scenario, the resource mapping and all the modulation is made in the RRU. This method actually benefits from the eventual low usage of the LTE signal, since the unused Resource Blocks (RB) will not be transported over the fronthaul. In this scenario, the network tidal wave effect could be used to reduce the required bandwidth on low network usage.

Although the Upper PHY method provides lower bandwidth requirements, the centralization gains is also lower. Since in this method the MIMO precoding is moved to the RRU, it is more difficult to implement spatial multiplexing techniques. In addition, the latency requirements are still very tight due to the channel information validity. In fact, all the PHY based split functionalities suffer from the same effect, since the maximum allowed latency has to be within the channel information valid period.

In addition to the previous split methods, it is possible to use the well-defined interface between MAC and PHY as another functional split, as shown in the scenario (d) of Figure 2.9. Different from the previous scenarios this one has relaxed latency and RTT requirements. In fact these values are just bounded to the HARQ maximum response delay. Also, in this method the bandwidth utilization is essentially composed by the user's traffic.

Although for the fronthaul such relaxed parameters would represent a cost reduction, the gains with centralization would be low, given that most of the processing would be made in the RRU. But, In this scenario, it still possible to achieve some gain by the application of Coordinated Scheduling [27, 28]. In such method, the transmission of each node in the network is scheduled in order to reduce the overall interference.

The last functional split reviewed here, is the scenario (e) of Figure 2.9, which is actually very similar to the distributed architecture currently available. Such method centralizes the higher layers of the base station such as RRC and PDCP. In this scenario, the bandwidth usage on the fronthaul is slightly lower than the previously shown scenario (d). On the other hand, the RTT and latency are relaxed since there is no boundaries from the HARQ operations.

The low requirements of scenario (e) makes this functional split desirable, even if the centralization gains from coordination and interference mitigation are nonexistent. In fact, this scenario is already provided by products such as the Ericsson solution for C-RAN [29].

As discussed, each functional split can provide different requirements for the fronthaul and respectively distinct centralization gain. The possibility to relax the requirements on Ethernet-based fronthaul is very interesting to be applied on the scenarios where the Ethernet network is congested. With the different functional splits, a centralized controller can be able to chose the optimal scenario depending on the network state.

2.2.2.1 Antenna Independent Traffic

To satisfy the predicted increase in the network usage, hundreds of antennas are planned to be used on the Radio Access Technology. In this scenario, it is very desirable to transport a single stream independent of the number of antennas and separate this stream in the RRU. This is actually a trade-off with the centralized processing expected for interference coordination and mitigation. From one side it is useful to centralize the process, but with massive amount of antennas, the fronthaul will have problems to transport these signals.

In this scenario, the tradeoff has to be evaluated on each case. For example, on beam-forming applications it is interesting to decentralize the beam-form algorithm, since it can be applied in the time-domain samples at the PHY lower levels. On the other hand, the operations of coordination and spacial diversity would benefit more from the centralized method, since the coordination information is more easily collected.

2.2.2.2 BBU and RRU dynamic mapping

Another important topic is the mapping between BBU and RRU, that means which BBU will process the signals of each RRU. In the existing network this mapping is static and realized at the network configuration, but the fronthaul can enable a more flexible mapping. For example, on high traffic demands the network can choose to share the processing between different BBU pools. In this scenario, the fronthaul has to follow a defined mapping between RRUs and BBUs.

In addition, there is a cost reduction attached to a flexible mapping, since in the low usage scenarios the network could centralize the processing and turn-off the unused resources. In fact, this dynamic topology changes can benefit even more from the tidal wave effect in the network usage.

2.3 Fronthaul's Signal Compression

The bit-rate is a critical resource for the NGFI mainly due to the high requirements defined in the existing fronthaul network. As seen in the previous sections, it is possible to reduce these requirements by the usage of different functional split between BBU and RRU. On the other hand, this technique also changes the achieved centralization gains. Another possibility to reduce the bandwidth requirements while still providing the centralization gain is to compress the signal transported over fronthaul.

This section discusses the algorithm used as basis for the compression implementation made in this work.

2.3.1 PUSQH Compression Algorithm

The compression implementation made in this work uses the algorithm Predictive Uniform Scalar Quantization with Huffman (PUSQH), described in [11] and developed at UFPA. This algorithm compresses the LTE signals to be transported over the fronthaul. It is a combination of OFDM adapted Linear Predictive Coding (LPC) and Huffman coding. This section will give a high-level view of the algorithm while the deep theoretical investigation can be found in [11].

To easily understand the algorithm application, let us follow the operation of compression. The PUSQH method encodes the real (I) and imaginary (Q) samples independently. The first

step is to remove the cyclic prefix of the OFDM signal creating the signal $x[n]$, where $0 \leq n < N-1$ and N is the number of samples per OFDM symbol. The first P samples of each OFDM symbol are used to initialize the predictor filter in both, the encoder and decoder, as proposed in [11]. The remaining samples of $x[n]$ are encoded with LPC and Huffman.

In PUSQH the compression is applied on each time-domain OFDM symbol. The first step to be made in order to compress the signal is to remove the Cyclic Prefix, as show in Figure 2.10. Then, a Linear Predictive Coding (LPC) is applied to the signal and derives an error reference that represents the difference between the original signal and the predicted one. Finally, the error signal is compressed with Huffman code in order to further reduce the amount of bits transported.

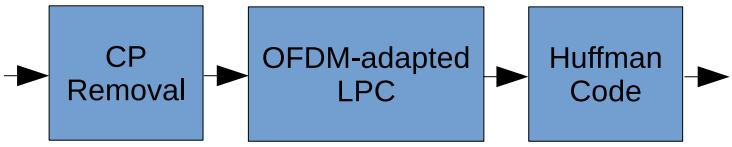


Figure 2.10: PUSQH compression flow graph.

The difference between the original signal and predicted signal, called prediction error, has a lower variance than the original signal. Thus, LPC enables to compress the prediction error with lower bits than the original signal and achieve the same level of distortion.

Based in the linear prediction theory [30] it is possible to estimate a system that predicts the behavior of a given signal, based on its characteristics. For example, the Levinson-Durbin algorithm uses the auto-correlation information of a signal to derive a FIR filter that can predict the next samples under certain error, based on a linear combination of the previous samples.

As a matter of fact, the PUSQH usage of prediction and quantization is also known as Differential Pulse-Coded Modulation (DPCM). It is a compression algorithm based in closed-loop predictive quantization. Such strategy is commonly used in the compression of audio and video and was shown by [11] that it can be also customized for LTE signals.

The P initial samples of each OFDM symbol are used to initialize the predictor filter, so these samples are losslessy encoded and transmitted to the decoder. This operation resets the prediction filter at the beginning of each OFDM symbol and is very important to keep the error within an acceptable range, as shown in [11]. Also, this imposition will drive some key project decisions in the hardware implementation.

After the application of the predictive quantization, the PUSQH algorithm also applies a

Huffman coding to the quantized error, in order to reduce the necessary amount of bits to be transported, without further distortion. This technique also benefits from the statistical characteristics of the error signal, since the probability distribution of the quantized error is approximately Gaussian.

One of the main advantages of the PUSQH algorithm is the low usage of computational resources. Even though the training stage of the prediction filter can use some computational resource, this stage can be performed off-line. As a matter of fact, the application of the prediction filter has very low cost, since the filter order is normally low. This makes the usage of PUSQH algorithm affordable even on scenarios with scarce resources.

Chapter 3

Fronthaul Ethernet Compression Testbed

As discussed in the previous sections, the usage of common Ethernet links to carry fronthaul streams could bring more flexibility and lower cost to the Next Generation Fronthaul Interface. In this context, the compression of fronthaul traffic is an enabling technology that reduces the huge requirements of the fronthaul transmission rate.

Similarly to the existing fronthaul technologies shown in Section 2.1, the hardware setup presented here consists of a BBU and RRU. The BBU reads LTE signals from its memory and sends over the network through the implemented fronthaul infrastructure. On the other end, the RRU receives Ethernet frames containing fronthaul data and recovers the transported LTE signal. At last the LTE signal reconstructed in the RRU is sent to an analog front-end to be analyzed in the real-world by external equipment.

The presented fronthaul testbed is implemented mainly in VHDL. Other resources were used such as IP-Cores and C code. In detail, the IP-Cores have been used for common hardware blocks such as the interfaces with Ethernet, Memory, and ADC/DAC interfaces. Also, the C language was used in software drivers to initialize and control the testbed. Moreover, MATLAB plays an important role to generate and validate the parameters used to configure the VHDL implementation.

The code is tested in real-time on a experimentation setup consists of two Virtex 7 FPGA boards and one Ethernet switch as shown in Figure 3.1. The process made on each board is shown in Figure 3.2 and described in subsequent sections. This setup makes possible to test the algorithm into real conditions. In addition, the signal transported over the fronthaul is reconstructed by the FPGA board represented by RRU and converted to analog to be analysed externally.

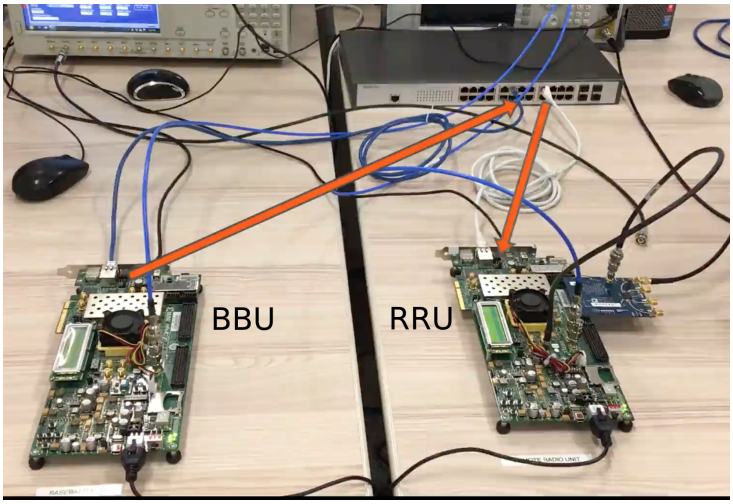


Figure 3.1: Hardware setup of the testbed FPGA boards.

This work presents an architecture to test and evaluate strategies and algorithms for next generation fronthaul that uses Ethernet as infrastructure, as detailed in Section 3.1. Furthermore, the work uses the architecture to implement a compression algorithm for LTE signals, as described in Section 3.2.

3.1 Ethernet Fronthaul Testbed

The fronthaul architecture presented in this work aims to map a CPRI stream into Ethernet frames. In other words, the CPRI data is generated and mapped into Ethernet frames. As shown in Figure 3.2, the proposed architecture employs two pairs of blocks: one to generate and consume CPRI streams and other to convert them into Ethernet Frames. Additionally to the CPRI traffic, there is also clock synchronization packets traveling over the Ethernet link provided by the IEEE-1588 protocol.

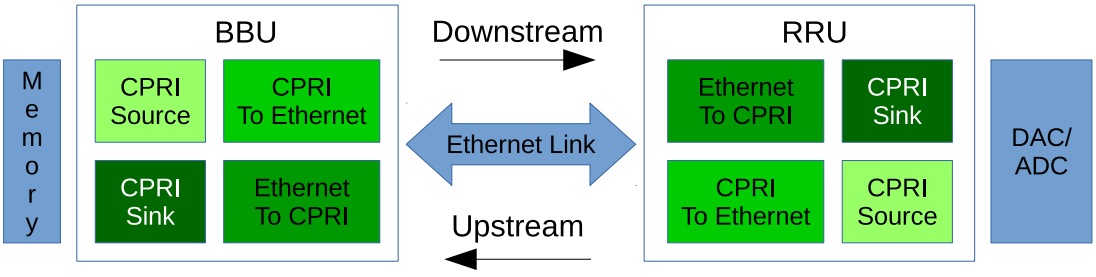


Figure 3.2: Fronthaul testbed architecture.

The blocks are designed to be reused on both Downstream (from BBU to RRU) and Upstream (from RRU to BBU) since the operation executed by each block is the same on both scenarios. On the other hand, the block configured for BBU operation have different input and output than in RRU. For example, the **CPRI Source** block at BBU receives samples from memory while the same block in RRU receives samples from the analog front-end.

The synchronization procedures defined on IEEE-1588 protocol, and further discussed in Section 3.1.3, use a special implementation strategy. In fact, these operations are hardware-assisted, which means that most of the protocol is implemented in software except the procedures that need strict timing. For example, the protocol messages are generated in software but fields with departure and arrival times are populated in hardware.

Figure 3.2 is a simplified block diagram of the implemented hardware. In fact, there are some extra blocks used to interface with external medium which are commonly available as IP-Blocks. For example, there are blocks specialized to read/write to FPGA's on-board memory. Also, there are others to send/receive data frames over Ethernet protocol on MAC and PHY layer. Equally, there is a block to interface with an analog front-end in order to reconstruct the signal in the real-world. Finally, the setup also counts with an embedded processor to run the necessary software drivers.

In addition, the majority of blocks have inputs and outputs compliant with the AXI-Stream protocol. This standard is part of the Advanced Micro-controller Bus Architecture (AMBA) an open-standard developed by ARM. The AXI-Stream is a specification for high-speed data streaming without support for address resolution, but with support for traffic flow-control. In fact, the AXI-Stream is normally used in point-to-point scenarios making not necessary the address resolution. This large utilization of AXI interfaces makes possible to reuse the blocks of the architecture in other scenarios. More information about AXI interfaces can be found in [31, 32]

The following sections will discuss more about the architecture, configuration and implementation details of each block.

3.1.1 CPRI Generation and Consuming

The presented architecture counts with two blocks specialized to generate and consume CPRI streams. These blocks implement a subset of CPRI protocol specification and focus on the necessary features to evaluate the impairments of Ethernet into CPRI stream. For example,

the CPRI Frame structure and CPRI departure timing have been implemented according to the standard specification. The objective is to implement the minimal subset of CPRI specification that still makes possible to evaluate the effects of Ethernet over CPRI.

The delineated blocks are called **CPRI Source** and **CPRI Sink**. Both are statically configured to reproduce a specific CPRI framing structure. The CPRI line rate (N_R) is one of these static configurations and indicates the throughput of CPRI stream. Another configuration is the number of Antenna Carrier Containers (S_{AxC}) used to indicate how many LTE signals each BF carries. Finally, the number of bits per IQ-Sample (IQ_b) which defines how much data per sample is inserted in each BF.

As discussed in Section 2.1.1, the aforementioned parameters are bounded by the Equation 2.1. To reduce the degrees of freedom over this equation, some assumptions have been made. Initially, it is assumed the whole BF payload is used so the inequality became an equality. Then, the CPRI line rate option 1 that has a bit-rate 491.52 Mbps is chosen to prevent the full usage of Ethernet link that has a peak rate of 1 Gbps, taking in account the encapsulation of CPRI into Ethernet will also add some overhead. As a result, (2.1) is reduced to the following equation:

$$\sum_{i=0}^{N_{\text{AxC}}} IQ_{b,i} \times F_{s,i} = \frac{15}{16} \times 491.52 = 460.8\ Mbps, \tag{3.1}$$

which depends only on the LTE signal configurations.

Any configuration that satisfies (3.1) can be used. Since the employed analog front-end provides two LTE signals at the same time, the N_{AxC} was set to 2. Also, the objective was to use the full dynamic range of the analog front-end. In order to accomplish that both sample rate and number of bits per sample have been chosen as 7.68 MHz and 30 bits. This sample frequency is relative to an LTE signal with bandwidth of 5 MHz [4]. All the presented values are recurrent over the work and composes the default configuration, as summarized in Table 3.1.

The mapping of IQ samples inside the BF is made as defined by the CPRI specification. The samples in the same AxC should be sent as a block respecting its sequence. Since the frame configuration is made to fully use the BF payload it is not necessary stuffing bits in the AxC containers. For example, in the default configuration the BF is organized as shown in Figure 3.3.

The CPRI Sink output is configurable in three modes: *DMA*, *DAC* and *TRASH*. The first writes the received CPRI stream into memory. The second sends the recovered LTE signal to an external digital-to-analog converter and the TRASH mode discards the received samples. The

Table 3.1: Default parameters of fronthaul's transported signal used in this work.

Symbol	Value	Description
F_S	7.68 MHz	Sample rate of transported LTE signal
IQ_b	30 bits	Number of bits on each LTE sample
N_{AxC}	2	Number of AxC transported by CPRI
N_R	1	CPRI Line Rate Option (see Table 2.1)
$N_{samples}$	2	Number of complex samples per AxC container

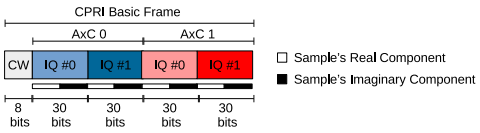


Figure 3.3: Default CPRI Basic Frame configuration.

block has one AXI-Stream input to receive CPRI data and two output buses, one for *DMA* and other for *DAC* operation.

The CPRI Source has to generate the BF exactly at the chip-rate (3.84 MHz). Since this block works at a base clock of 100 MHz, it is necessary some flow-control mechanism to provide the expected output BF rate. To accomplish that, the block uses an additional external clock derived from the reference used in IEEE-1588 protocol. This clock is multiple of the chip-rate and drives the input rate of samples in the CPRI Source. Since the sample rate changes with the type of LTE signal used, it is necessary to reconfigure the clocks for each signal. For example, in LTE 5 MHz the sample rate is 7.68 MHz, which is 2 times bigger than the chip-rate, so the additional clock should also use the same frequency.

3.1.2 Encapsulation of CPRI into Ethernet

The CPRI encapsulation into Enthernet can be made by two modes: Structure Aware or Structure Agnostic. Both modes are studied on the IEEE 1904.3 [20] task force. The Aware mode extracts the information from CPRI stream then creates the Ethernet Frames. On the other hand, the Agnostic mode uses the raw BF to compose the payload of the Ethernet Frame. The trade-off between both schemes lives on the capacity to map different CPRI information into better suited Ethernet frames versus the simplicity of not having to know the structure of each

CPRI BF.

Since the IEEE 1904.3 specification [20] is a work in progress, this work implements its own version of the Structure Agnostic Mode. The blocks **CPRI to Ethernet** and **Ethernet to CPRI** implements this operation of conversion between CPRI stream and Ethernet frames. The essential procedure is to accumulate a configurable amount of CPRI BFs and generate a single Ethernet frame. Additionally, a custom header is created with meta-data information from the frame such as: the number of encapsulated BF, the sequence number and a departure/arrival timestamp. This structure is depicted in Figure 3.4.

Additionally to the Ethernet header, there is an extension header called fronthaul Meta-Data. This extension provides useful information between the encapsulator and decapsulator blocks. For example the timestamp field is used to log the departure and arrival times of each Ethernet frame. Also, the sequence number field is necessary to discover possible frame drops and error conditions. Finally, the number of BF in the Ethernet Frame is used to indicate how much data must be recovered.

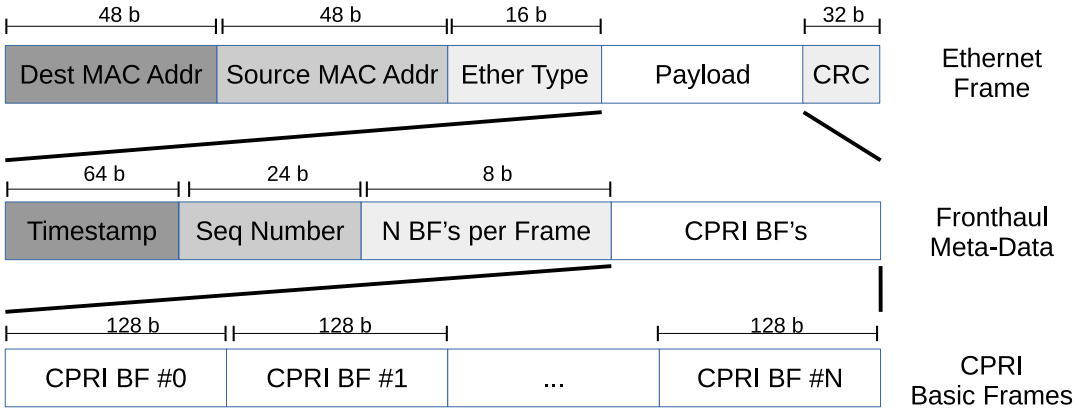


Figure 3.4: Example of CPRI encapsulation on Ethernet frame.

The Generated Ethernet frame contains three protocol layers: the Ethernet MAC, the fronthaul meta-data and the CPRI stream, as shown in Figure 3.4. The addition of these headers increases the overhead (OV) of the CPRI Stream. Of course, the overhead reduces with larger values of N_{BF}, as calculated in the following equation:

$$OV = \frac{H_s}{(N_{BF} \times 128)}, \tag{3.2}$$

where the H_s value is the total amount of bits used for headers. For example, the total number of bits used by both Ethernet and Meta-Data is 240 bits, which generates an overhead around

11% in a frame with $N_{BF} = 16$.

Also, there is a latency created by BF accumulation before the Ethernet frame encapsulation, which generates a delay variation on the BF delivery. The BFs are generated by CPRI at the exact rate of 3.84 MHz. The encapsulation process adds some latency since the first BF will wait until the N_{BF}-th BF to departure. It is possible to calculate the maximum jitter by evaluating the time that the first BF has to wait, which can be calculated as show in the following equation:

$$J = N_{BF} \times 260 \text{ ns}, \tag{3.3}$$

by multiplying the number of BF (N_{BF}) by the period of each BF (260 ns).

Header overhead and jitter are intrinsic to the encapsulation process based on the Structure Agnostic Mode. These impairments are mainly dependent on the number of BF each Ethernet frame transports. When the number of BF per packet increases, the jitter also increases and the proportional overhead reduces, and vice-versa. In this work the default value of N_{BF} is 16, which leads to an overhead of 11% and a peak jitter of 4.1 μs.

Despite these impairments affecting the CPRI stream encapsulated over Structure Agnostic mode, there are techniques to deflect these effects. The overhead impairment is easily removed by the usage of more bandwidth as long as the Ethernet link is not fully used. Otherwise, the jitter can be extinguished by implementing Dejjitering Buffers, which stores the BF before forwarding. This technique is a form to reduce the jitter by increasing the average delay.

Some configurations of the blocks can be changed in real-time, using a software driver. For example, the number of BF into each Ethernet Frame, as already discussed. Likewise there are other configurations such as the Ethernet frame header fields: MAC source/destination addresses and Ethernet Frame Type. It is also possible to enable/disable the block operations by software.

Another important aspect is the time between consecutive Ethernet frames. It is possible to use two distinct methods by enabling or not the internal flow-control procedure. When active, the flow-control will prevent the block to start a new frame for a configurable amount of time. On the other hand, with the flow-control disabled the Ethernet frame is generated as soon as the input provides enough data.

3.1.3 Clock Synchronization Procedures

The IEEE-1588, also known as Precision Time Protocol (PTP, see e.g. [33]), is a standard describing procedures to ensure the signal synchronization over a packet network. synchronization can be divided into three components: the syntonization, phase synchronization and time synchronization. Syntonization is related to the signal's frequency and aims to make the clock rate the same at both ends of the network. Phase synchronization otherwise aims to make both signals have the rising edge at the same instant. Finally, time synchronization is related to the time of day representation on network nodes.

The PTP uses a set of timestamped messages to discover the delays of the network. A visual representation of the PTP messages over the time is shown in Figure 3.5. In this representation, there is a master node, which has a high fidelity clock also know as Stratum 1, and a slave node which intends to synchronize to the master's clock over the network. The handshake shown intends to collect the timestamps T1, T2, T3 and T4 in the slave node, so it can measure and correct the offsets of its local oscillator to the master oscillator.

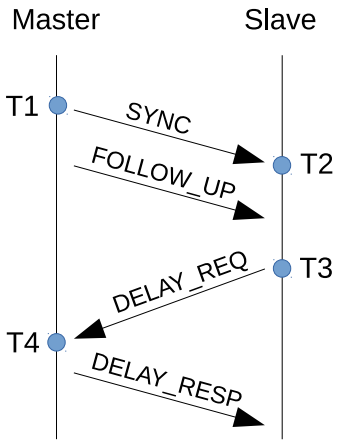


Figure 3.5: PTP messages and timestamps.

The handshake starts by the SYNC message, which is timestamped on both departure (T1) and arrival (T2). Since T1 value just exists on the master another message, called FOLLOW_UP, is sent carrying this information. Then, the slave executes the same procedure sending the DELAY_REQ message to the master. This message is also timestamped on departure (T3) and arrival (T4). Finally, the slave receives from master the DELAY_RESPONSE message containing the T4 value.

The timestamping procedure needs to be as fast and accurate as possible. In fact this

procedure is implemented closer to the Ethernet PHY. As long as the timestamping is accurate, the message departure instant does not need to be so strict. Then, the PTP implementation is normally done as a hardware-assisted protocol. The messages are created and send by a software implementation while the timestamping is executed in real-time by the hardware.

The syntonization or frequency synchronization can be achieved based on sequential values of T1 and T2 measurements. In other words, it is based on a one-way communication since the messages from slave to master are not used. Assuming two measurement realizations ($T_{1,k}$, $T_{2,k}$ and $T_{1,k+1}$, $T_{2,k+1}$), it is possible to calculate the frequency offset (y) by the equation:

$$y = \frac{(T_{2,k} - T_{1,k}) - (T_{2,k+1} - T_{1,k+1})}{T_{1,k} - T_{1,k+1}}. \tag{3.4}$$

This result assumes there is no influence of the network on the measurements, but if the network is congested there is also a noise parcel in the result.

To find the phase synchronization, all the timestamps of the handshake are necessary. The phase error (x) can be easily calculated when the one-way delay of the path ($\hat{d}$) is known, as shown in the following equation:

$$x = T_2 - (T_1 + \hat{d}). \tag{3.5}$$

Also, the one-way delay can be calculated by:

$$\hat{d} = \frac{T_{ms} - T_{sm}}{2} = \frac{(T_4 - T_1) - (T_3 - T_2)}{2}. \tag{3.6}$$

Specially, the delay calculation assumes there is no network delay asymmetry in the path. If the delay is asymmetrical, a parcel of noise will be added to the delay calculation and also the phase offset. Finally, the time synchronization is also achieved through message handshake, but in this case it is not necessary strict timestamping since the objective is to correct the time representation of the slave.

The calculated offsets are used to correct the local oscillator. There are two types of corrections: one coarse and other finer. They are usually used together to provide stability and control. The coarse correction uses a Digital Phase Locked Loop (D-PLL) to generate a signal with the corrected offsets. This solution have a stable result, but it is bounded to the correction levels of the D-PLL. The finer correction uses a digital counter to interpolate or decimate the time representation of each clock cycle in order to generate a signal based on the corrected time representation. For example, a clock with 100 ns period, can be used to derive another clock with period 50 ns by counting the first clock twice.

Since the network can introduce a big amount of noise into the PTP offset calculations, the protocol defines switching functionalities, which are capable to measure and signalize how much time each packet spent inside the switch, called residence time. With these correction values from the network, it is possible to have precise values for each timestamps (Tn) leading to better offset estimations. Current switches with these PTP functionalities are expensive and increases the cost of the network. On the other hand, the usage of Ethernet infrastructure with PTP support is still cheaper than other solutions such as GPS or dedicated synchronization links.

3.2 Fronthaul Compression Implementation

The architecture described on the previous section can be used to evaluate the implementation of several algorithms for fronthaul over Ethernet. With this testbed, the compression algorithm described in Section 2.3.1 was implemented and evaluated. In fact, from the architectural perspective shown in Figure 3.2, the compression algorithm can be placed inside the CPRI Source and Sink blocks. To put it differently, the compression is executed before the CPRI encapsulation and the decompression is performed after CPRI unpacking. This work in particular implements the Decompressor block on the RRU while the compression operation is emulated by forwarding compressed samples from BBU's memory.

The CPRI Sink block already receives and decodes the BF. The Decompressor then receives the BF payloads and decodes its contents. There is one Decompressor instance for each AxC. Since the default configuration uses two AxC there are two Decompressor instances. Each instance is formed by the pipeline shown in Figure 3.6. The **Frame Unpack** recovers the compressed stream of each signal component (I and Q). Both streams are passed through the decompression algorithm implementation on **Decompressor** block. Finally, the cyclic prefix (CP) is inserted back in the OFDM symbol by the **CP Insertion** block.

The Decompressor is also composed by a pipeline, as shown in Figure 3.6. Since the compression is executed for each symbol, this block decodes only one symbol at a time. This procedure assures a symbol-wide coordination between I and Q components, even when both streams are decompressed in separate blocks. The details of Decompressor implementation will be better described in Section 3.2.2.

When the OFDM symbol is compressed with the method [11], the boundaries of each sample is lost due to the variable rate encoding scheme. In this scenario, the whole symbol

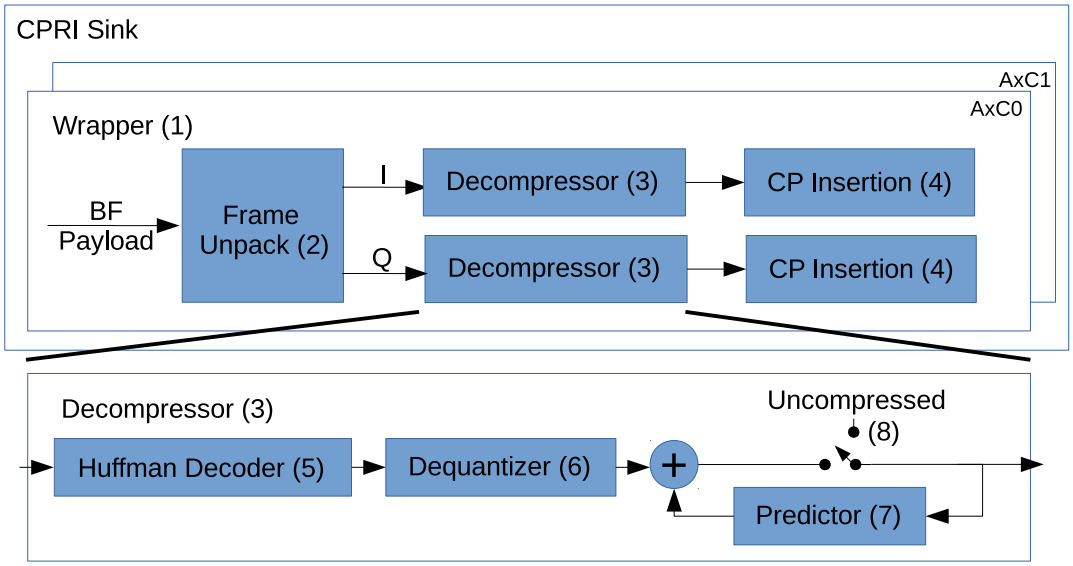


Figure 3.6: Decompressor blocks flow graph.

became a single stream. As a result, it is not possible to divide the same amount of compressed samples on each BF as specified by CPRI. Drove by this scenario this work decides to not follow the CPRI protocol when the compression is activated. Then, the BF payload is passed to another protocol layer to decode the compressed stream. This additional protocol layer is further described in Section 3.2.1 and is part of the Frame Unpack block.

The compression operations can be enabled or disabled. When disabled, the testbed is essentially the architecture presented in Section 3.1. Additionally, it is possible to choose the compression ratio of 2 or 4. Internally the pipeline has several other configurations, but they are arranged on presets to provide each target compression ratio. Indeed, the Decompressor algorithm can be configured to a variety of compression ratios, but in order to make the CPRI and Ethernet encapsulation simpler, the ratio was restricted to the integer values multiple of 2.

The following sections will discuss in more details the implementation and usage of each block. First, the Frame Unpacker and the protocol used to decode BF payload is shown in Section 3.2.1. Then, the general overview of the Decompressor block is given in Section 3.2.2. More details about the Huffman Decoder and Predictor Filter are shown in Sections 3.2.3 and 3.2.4, respectively. Finally, the Cyclic Prefix insertion is analyzed in Section 3.2.5.

3.2.1 Unpacking CPRI Basic Frames

The Unpacker was designed to consume the payload of CPRI BF and isolate the compressed stream. The compression operation needs different types of information that are transported over the CPRI BF. For example: compressed, non-compressed samples and flags indicating the start of an OFDM symbol or LTE Slot. In fact, the block has to define an additional protocol layer to extract the necessary information from BF payload.

The first special characteristic of compressed stream is the necessity of both compressed and non-compressed samples to be forwarded over the fronthaul. As described in Section 2.3.1, the Decompressor uses non-compressed samples at the beginning of each new OFDM symbol to reset the prediction filter. Only after the reset operation the block uses compressed data to feed the Decompressor pipeline.

Furthermore, these different types of data requires the utilization of bits from the BF payload to indicate how each one should be interpreted. These flags are realized by two fields: an Enable Flag and a Command Flag. The first is composed by a single bit, the MSB of AxC container. This flag signalizes if the data inside AxC container is compressed or not. The second flag is composed by the subsequent 3 MSB of the AxC Container. This flag just exists when the Enable Flag is active and indicates what type of information is carried in the container, as detailed in Table 3.2.

Table 3.2: List of Flags incorporated on the compressed stream.

Enable Flag	CMD Flag	Information
0	N/A	AxC carries compressed samples
1	111	Start of an OFDM symbol
1	110	Start of an LTE Slot
1	000	AxC carries non-compressed Samples
1	100	AxC carries nothing, should be discarded
1	Others	Reserved for future use

The real compression ratio generated by the algorithm is slightly decreased due to these set of others information that are transported with the compressed samples. Also, the padding used for achieving an integer compression ratio changes the compression ratio. In fact, the

user's configuration is a target to be achieved while the real compression ratio is the closest point. The implementation then assures the user's configuration by using a higher ratio and padding the stream with empty BF. For example, in the target compression ratio of 4, actually the compression ratio is 4.25, but empty BF are used to reduce this value into the target integer compression ratio.

Due to the stream padding with empty BF, it is necessary to use a flag to indicate which BF should be discarded. This flag is also useful to equalize the amount of BF used on the compressed streams of I and Q components. Both components are transported together, but compressed separately. Then it is possible that the Huffman encoder creates streams of different sizes for the I and Q compressed samples. Thus, by using the command flag 100 in Table 3.2, it is possible to use dummy BF to equalize the number of BF on each both streams (I and Q).

Figure 3.7 shows how the compressed streams of an entire OFDM symbol are mapped in basic frames. The first P basic frames transports information of the P initials samples of each OFDM symbol, it includes the real (I) and imaginary (Q) components. Then, the remaining Basic frames transports the Huffman bit-stream. In addition, the Huffman stream of I and Q components are interleaved among the basic frames. Eventually, some dummy BF may be used to equalize the number of BF used by I and Q compressed samples.

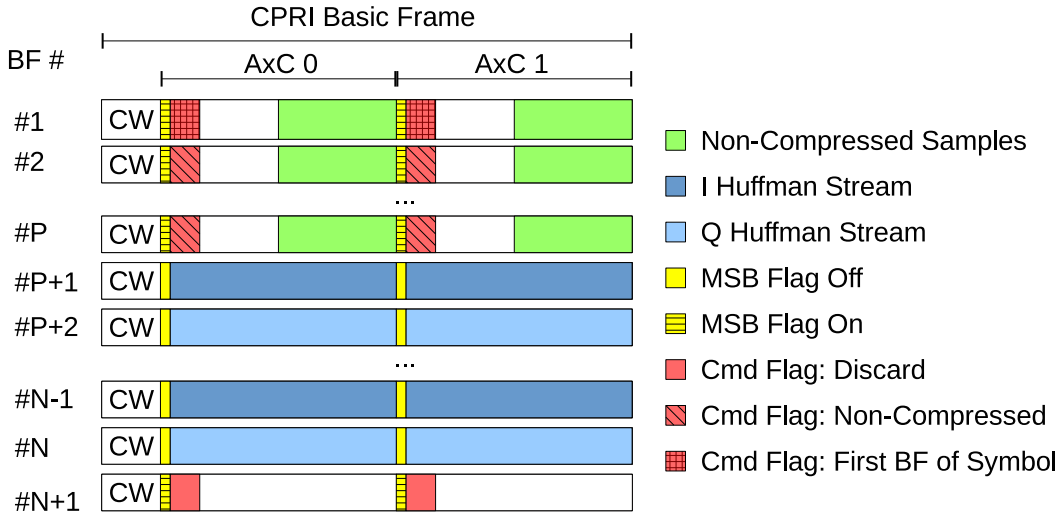


Figure 3.7: CPRI BF payload for the bit-stream of compressed OFDM symbol.

In order to correctly decode one OFDM symbol, the Basic Frame Unpacker has three main stages. The first is to wait for the flag indicating the start of an LTE slot or OFDM symbol. On the second stage, the non-compressed samples are received and forwarded to the predictor

filter, as expected on filter reset. Then, the final stage is to receive and forward the compressed samples taking care to discard any empty BF. The Figure 3.7 shows how the sequence of BF have to be in order to correctly unpack an OFDM symbol.

The number of non-compressed samples in the first stage of unpacking is linked to the prediction filter order. This configuration is made on the unpacker block instantiation and should use the same value as the filter order. Just one non-compressed sample is carried by AxC, when the AxC have the flags to do so. For example, in the default configuration each AxC container has 60 bits, of which 4 are used on flags and 30 bits used for non-compressed sample, remaining 26 unused bits. These unused bits represent a negligible overhead.

The unpacker blocks have to assure the first symbol to be decompressed is also the first symbol on LTE Slot. This requirement is made from CP insertion and is further discussed in Section 2.3.1. On the other hand, the unpacker does not use a flag to indicate the end of a compressed symbol. In fact, this information is provided by other blocks in the decompressor pipeline such as the Huffman Decoder and LTE CP insertion. These blocks signalizes when an OFDM symbol has been processed completely. This information is further used by the frame unpacker to prepare the pipeline for the next compressed symbol.

3.2.2 Decompressor Implementation

To understand the decompression, let us recall the compression operation. The real or imaginary parts of each OFDM symbol without CP are encoded with Linear Predictive Coding (LPC). In other words, the original signal is subtracted from its prediction and the prediction error is quantized. The quantization result is encoded with Huffman code and generates the Compressed Stream. The decompressor block executes the reverse order. It receives the compressed stream and returns an estimation of the original LTE signal, as shown in Figure 3.8.

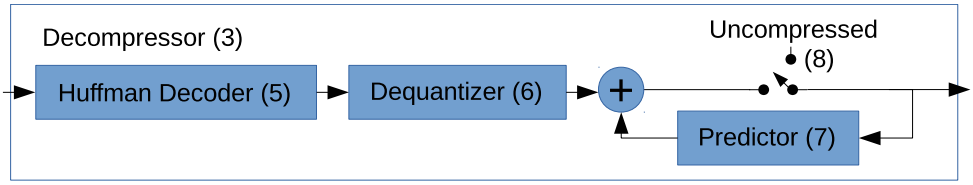


Figure 3.8: Decompressor with its internal blocks: Huffman and LPC decoders

The Huffman Decoder block process the received stream and returns the decoded indexes. The block implements the minimum redundancy code created by Huffman [34]. This block is

pre-configured with the same dictionary used in compressor block, which is generated based on an off-line training procedure. In this block the compressed stream is compared to the dictionary to discover the indexes. This block will be discussed in details in Section 3.2.3.

The Dequantizer block is essentially a lookup table and executes the common dequantization operation as defined in the literature. In this work the quantization levels are hardcoded into the VHDL code of the quantization block. The levels are defined on the training process as described in Section 2.3.1. Since the dequantization process is essentially asynchronous, the AXI-Stream control signals in the input are bypassed to the output AXI interface.

The Decompressor pipeline has a special switch to be able to insert non-compressed samples directly in the prediction filter during reset operation. The decompression reset procedure has already been discussed in Section 2.3.1, and is executed at the beginning of each symbol. In order to execute the reset procedure this switch block counts internally the samples to predict the beginning and end of each OFDM symbol. When a new symbol starts the block automatically switch the source of samples from the pipeline to the uncompressed input.

The predictor block is essentially a FIR filter with taps that minimizes the prediction error energy for real or imaginary parts of LTE signals. On the reset stage this block should receive non-compressed samples. On any other stage the block receives the sum of quantized prediction error and the last predicted sample. The existing feedback in the predictor filter makes the AXI-Stream compliance much times complex. In this case a simpler approach is used without the support for flow control. The characteristic of this block will be discussed in details in Section 3.2.4

3.2.3 Huffman Decoder Block

The Huffman algorithm [34] encodes different indexes with different amount of bits depending on their frequency of occurrence. The most frequent indexes are encoded with less bits while the less frequent indexes are encoded with more bits. Thus, the Compressed Stream indexes have in average less bits than original.

The Huffman decoder has been implemented based on a look-up table approach, such as in [35]. Other features have been added to the Huffman decoder such as the input and output AXI-Stream compliance and bit-width configuration of input and output. Most of the configurations including the Huffman Dictionary are made in block instantiation. The block receives the payload of AxC container and outputs one decoded index per clock cycle.

The block also relies on other configurations made on VHDL instantiation. For example, the minimum and maximum bit-width of Huffman dictionary indexes. These values are used to evaluate when the input has enough bits to be processed. Equally important are the configurations of predictor filter order and OFDM symbol size. These configurations are used to determine how much Huffman indexes the block has to generate.

The Huffman decoder can be divided in two major stages: the first concatenates input streams sequentially, while the second compares the concatenated stream with a pre-determined Huffman dictionary, as shown in Figure 3.9. The initial concatenation is necessary because the bit width of the next codeword is unknown until the comparison to the dictionary. Likewise a Huffman codeword can be split between two sequential input words. Hence the concatenation block has to assure there is always enough bits to be consumed even for the biggest codeword.

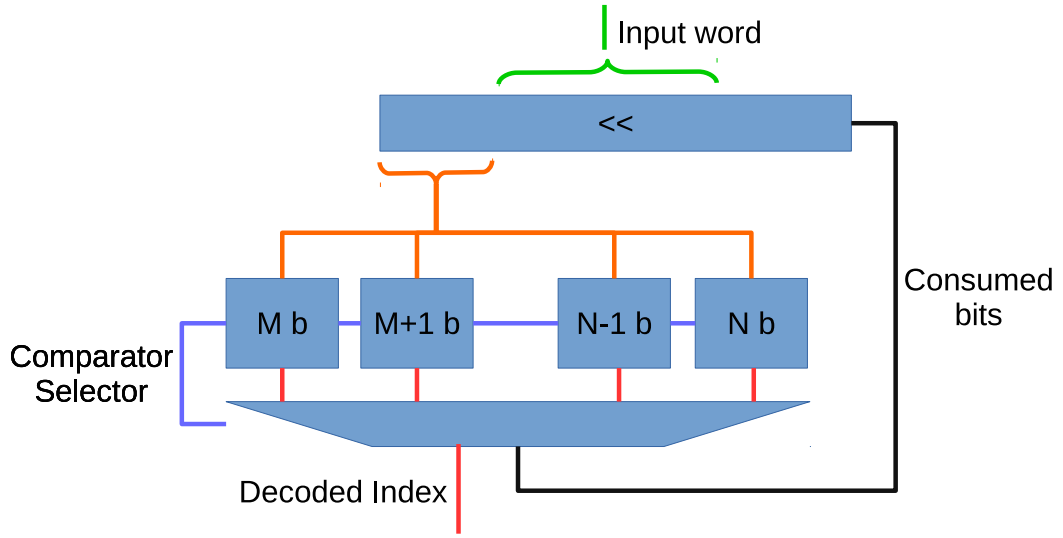


Figure 3.9: Huffman decoder internal flow graph.

At the first stage, when a codeword has been decoded, the dictionary comparator signalizes the number of bits consumed from input. Under those circumstances the concatenator discards those bits using a shift register in order to prepare for the next codeword. When the block realizes the number of bits remaining on the shift register is not enough, a new input is taken and inserted on the register. Otherwise the block continues inserting data into the comparator and discarding input with a shift-register.

The bit-shift operation executed by a variable number of bits can be trivially implemented with sequential flip-flops with each shift executed on a clock cycle. On the other hand, the same operation can be executed on a single clock cycle using a more complex solution. For example,

it is possible to use multiple right-shifted copies of the input and a multiplexer to update a register. In this work the latter is used to reach a high throughput implementation at the cost of complexity.

At the second stage, the dictionary comparator is composed by a series of lookup tables (LUT) one for each codeword size. The bits provided by the concatenation block are inserted into all the lookup tables at the same time. When the Huffman stream is received without errors only one look-up table has a valid output per time. This is achieved due to the prefix property of the Huffman code [36]. Each LUT outputs the decoded index and the bit width of the consumed codeword. If none or more than one LUT has an valid output an error condition is rose, e. g., due to Ethernet bit error.

The look-up tables are hardcoded, and automatically generated by a MATLAB script based on the Huffman dictionary. This stage of the block is asynchronous so it does not add any clock-cycle delay to the decoder. Due to its implementation the decoder block is capable to deliver one Huffman decoded index on each clock cycle. On the current configurations of clock rate it means a maximum throughput of 100 Millions indexes per second.

Finally, to avoid any misalignment on the OFDM symbols, the Huffman decoder block counts the number of decoded samples and stops its operation after the end of each OFDM symbol. Then the block waits for a reset signal, which signalizes the block can consume another OFDM symbol. Also, if any error occurred on the comparator block the condition is signalized to the higher blocks.

3.2.4 Prediction Filter Block

As explained in Section 2.3.1, the predictor filter is a Finite Impulse Response (FIR) filter with coefficients trained to predict an LTE signal based on its last samples. Although the filter implementation in VHDL is very common, the used strategy have specific details on fixed-point number format that will be further discussed. For example, how the fixed point operations are handled to avoid error propagation due to overflow or underflow of fixed-point values.

First of all it is important to note that the predictor block is not AXI-Stream compliant. In fact, the block counts with a very small set of input and output signals. Aside from the default signals (clock and reset) and the data signals (input and output) this block counts with only a write-enable signal. The default predictor operation can be seen on Figure 3.10 and is essentially based on the write-enable signal. When this signal is activated the input is consumed

and an output is generated in the next clock cycle. This operation guarantees that the internal filter multiplications are updated only when the input is valid.

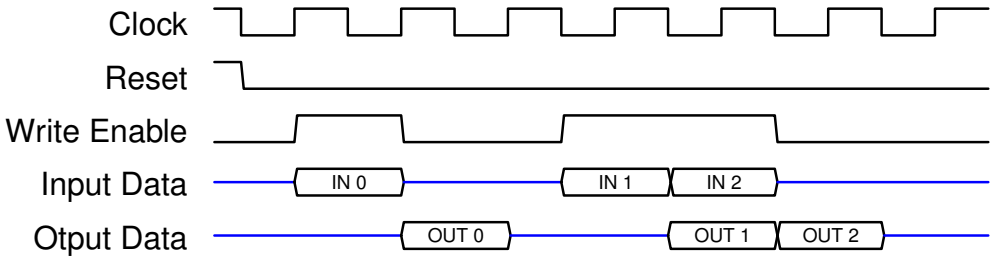


Figure 3.10: Timing overview of prediction filter input and output signals.

It is possible to configure the fixed-point number format, also called Q-Format, for both the input and the filter coefficients. Likewise the output uses the same format as the input. The Q-Format Q(M,N) is formed by the number of bits representing the integer part (M) and fractional part (N). For instance, the values of M or N can be negative, as long as the total bit-width (M+N) is non-negatives. For example the Q(-1,17) represents a 16 bit number with bits representing the fractional part from the 2nd to the 17th binary place.

All the arithmetic operations in the filter are thought to avoid any fixed-point overflow or underflow. For example, the multiplication product have more bits than its factors. In detail the Q-Format of product is the sum of factor's Q-Format. Additionally, result of the sums have one bit more than its biggest portion. Figure 3.11 shows the Q-Format in each part of the prediction filter. The input is represented by Q(A,B), the multiplier values and the result are represented by Q(C,D) and Q(A+C, B+D), respectively. The result of summing all multiplication taps is Q(A+C+P, B+D), where P is proportional to the number of adders.

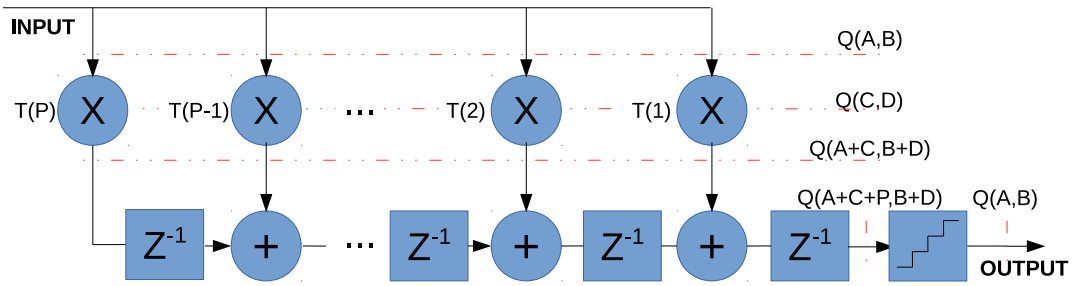


Figure 3.11: Fixed point Q-format of internal signals on prediction filter.

The filter result also needs to be resized and rounded in order to generate the same number of bits consumed, as shown in Figure 3.11 The effect of use more bits on each arithmetic

operation and execute a single round stage increases the result's precision by reducing the effects of error propagation. Any error propagation can be very dangerous especially assuming the output will feed-back into input, as shown in Figure 3.8.

3.2.5 Cyclic Prefix Insertion Block

The cyclic prefix insertion operation repeats the end of an OFDM symbol in the beginning to create a guard-time between consecutive OFDM symbols. As discussed before, the LTE Cyclic Prefix (CP) is used to prevent the effects of inter-symbol interference in dispersive channels. On the other hand, the CP itself is a source of redundancy for the LTE signal. In fact, to reduce the amount of data being transmitted on the fronthaul the CP can be added in the RRU.

The LTE specification defines two types of cyclic prefix insertion procedure [4], this block just implements the Normal Mode. In fact, the block adds $4.7\ \mu s$ of prefix to each symbol, with an exception for the first symbol of each LTE slot, which receives $5.2\ \mu s$ [4]. As a result, for an LTE signal with bandwidth 20 MHz, the number of samples in the CP is 160 for the first symbol of the LTE slot and 144 for the other symbols as shown in Table 3.3.

Both input and output of the implemented block respect the AXI Stream protocol. On each AXI transaction one sample (real or imaginary) is consumed and/or produced. In addition, the block configurations allow to change the sample width (in bits) of input/output as long as the number of symbols in an LTE slot and the size of LTE symbol (in samples). Also, the number of samples in the CP can be configured for both the first symbol in LTE slot and the subsequent.

Table 3.3: Number of samples and duration per Symbol OFDM in an LTE slot. The length of CP is different for the first OFDM symbol.

	LTE Slot						
	OFDM Symbol 1		OFDM Symbol 2		...	OFDM Symbol 7	
	CP	IFFT out	CP	IFFT out	...	CP	IFFT out
Samples	160	2048	140	2048	...	140	2048
Micro Seconds	5.2	66.7	4.7	66.7	...	4.7	66.7

The implemented block assumes that the first symbol received at block start is also the first symbol of an LTE slot. Therefore it is possible to derive the position of each symbol and sample in the slot by counting how much samples have been already processed. Moreover, the

block uses two FIFOs: one to store the entire OFDM symbol and other to store just the cyclic prefix. The CP insertion is executed by writing the prefix portion of the symbol in both FIFOs while reading first from prefix FIFO and then from the symbol FIFO, as show in Figure 3.12.

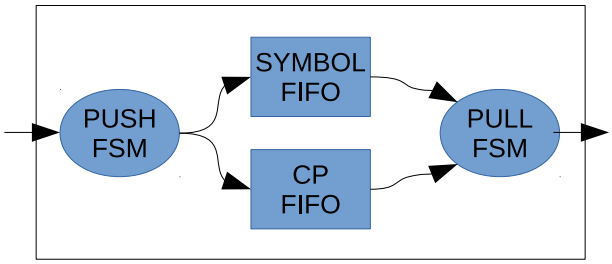


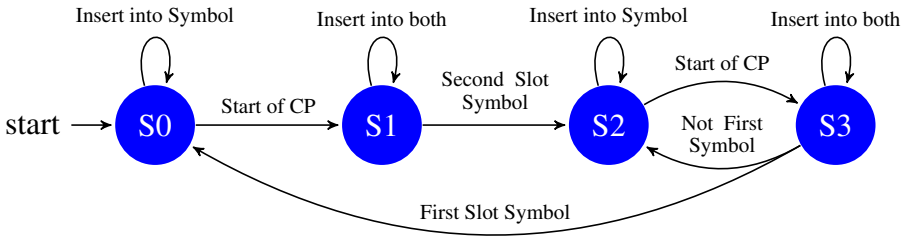
Figure 3.12: Machines and FIFOs used in CP insertion block.

To control the write and read operations on both FIFOs two independent state machines are used, respectively called PUSH_FSM and PULL_FSM. Each state machine has its own sample counter. In other words, the input and output can operate at different symbol time. As a result, this flexibility prevents input form stopping while output is momentarily halted and vice-versa.

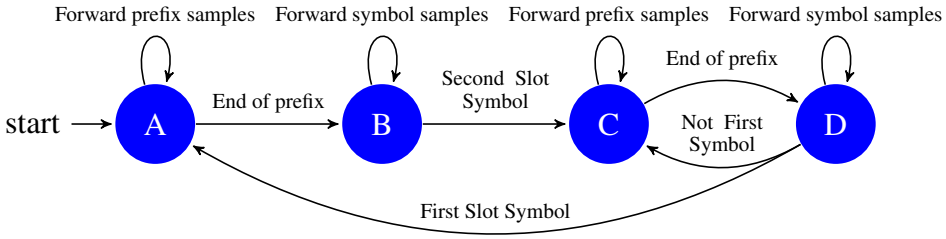
On PUSH_FSM the states **S0** and **S1** are dedicated to generate the prefix of the first slot symbol. Hence the states **S2** and **S3** generate the prefix for the others symbols on the slot. Therefore, the pairs (**S0**, **S2**) and (**S1**, **S3**) execute the same operations but with different prefix sizes in samples. In particular the states **S0** and **S2** insert samples into Symbol FIFO, while the states **S1** and **S3** forward input to both Symbol and CP FIFOs.

The sequence of states on PUSH machine is shown in Figure 3.13a. The states **S0** and **S1** are run for the first OFDM symbol of the slot, while the others states run in loop for the other OFDM symbols. Also, when a new slot starts the state goes back to the **S0**.

The PULL_FSM state machine structure is very similar to the PUSH, with the states **A** and **B** being used on first slot symbol and the states **C** and **D** to the others. The states **A** and **C** read samples from CP FIFO, while the states **B** and **D** read samples from Symbol FIFO. The sequence of states is shown in Figure 3.13b.



(a) LTE Cyclic Prefix Insertion PUSH State Machine.



(b) LTE Cyclic Prefix Insertion PULL State Machine.

Figure 3.13: Overview of the cyclic prefix insertion block state machines.

Chapter 4

Testbed Evaluation Results

4.1 Testbed evaluation

With the architecture presented in the previous chapters, the hardware testbed was built as shown in Figure 4.1. It is possible to classify the blocks in four Subsystems, namely Ethernet Subsystem, CPU Subsystem, ADC/DAC Subsystem and Ethernet Fronthaul Controller Subsystem. The previous chapter focused mainly in the Ethernet Fronthaul Controller, which is represented in blue color in Figure 4.1.

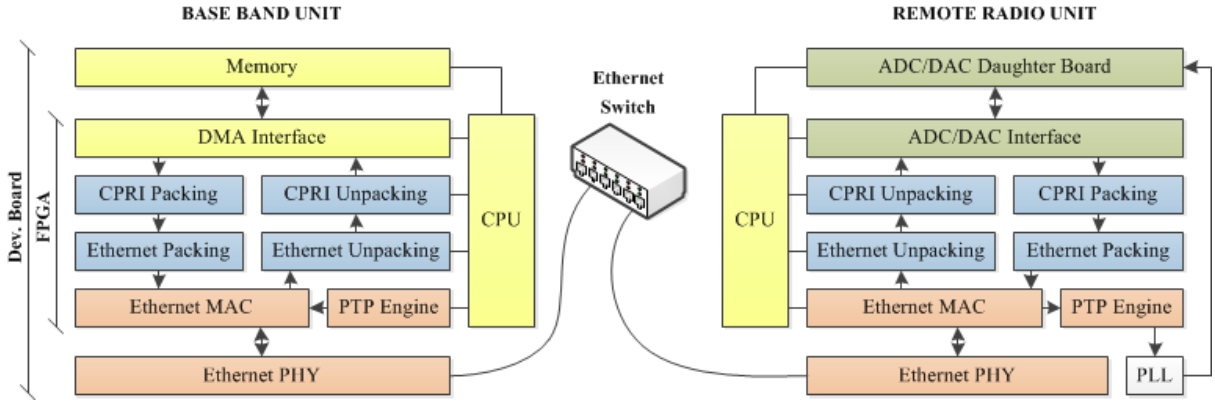


Figure 4.1: Fronthaul testbed blocks.

The Ethernet subsystem is responsible for transmiting and receiving Ethernet Packets. It also contains the necessary hardware to execute the PTP protocol. This infrastructure is provided by an on-board PHY (MARVELL 88E1111) [37] and a Xilinx's MAC soft-core [38] inside the FPGA. The MAC has a hardware-assisted PTP support as described in Section 3.1.3.

The CPU Subsystem configures and monitors the hardware setup. In other words, it starts

the setup by configuring and running subsystems. After start, the CPU collects information from the testbed and handles the interruptions generated by the hardware. This subsystem is implemented by a microprocessor soft-core with support to the on-board peripherals, such as DDR3, UART and JTAG.

The ADC/DAC subsystem is a soft-core block that interfaces to an external ADC/DAC board (FMCOMMS2). This board uses the AD9361 chip which is a complete RF Frontend designed for 3G and 4G base station applications. The chip has several software-configurable options such as the sampling rate, number of transmit/receive antennas and analog gains. Also, the FMCOMMS2 [39] has an external clock reference which can be used for synchronization.

The Ethernet Fronthaul Controller executes the fronthaul procedures described in details in Chapter 3. These blocks are developed in-house to transport CPRI traffic over Ethernet and also provide flow-control and synchronism. In addition, these blocks incorporate the implemented LTE signal compression algorithm. This subsystem also includes the software necessary to process and filter the PTP time-stamps gathered on the Ethernet Subsystem.

Finally, at the RRU there is a special setup to synchronize the ADC/DAC board with the reference generated by PTP. In detail, the clock recovered from PTP engine has 8 kHz in frequency, while the ADC/DAC requires a reference with frequency of 40 MHz. In this scenario, a Dejittering PLL was used to multiply the frequency of the reference clock. In addition, the ADC/DAC board requires the clock signal to have a high slew rate, that means the transition time between the clock on and off should be very small. Then, an external PLL board was used to increase the slew-rate of the signal generated by the FPGA.

All these subsystems combined compose the fronthaul over Ethernet evaluated in this chapter. Together, these subsystems use around 30% of the FPGA resources for both BBU and RRU as detailed in the Table 4.1. The FPGA model used is the Xilinx *xc7vx485t* that comes in the VC707 development board. But, based on the used resources, it is possible to map the same project into an smaller and cheaper FPGA, such as the *xc7a100t*.

To evaluate the signal results on the signal decompression and transport over the fronthaul a Vector Signal Analyzer (VSA) was used. The decoded LTE signal on the RRU is converted to the analog domain and analyzed by the VSA. In this scenario the EVM metrics evaluate the quality of the compression algorithm. In addition, the reconstructed signal can give some insight about the fronthaul transport, since a packet drop or protocol malfunction will affect directly the signal reconstruction.

Another evaluation method used was the measurement of the phase noise of the PTP-recovered clock signal. Again the VSA was used to measure the clock's phase noise characteristics in different scenarios. For example, how the addition of another hop in the network affects the phase-noise of the recovered clock.

Table 4.1: Hardware Utilization for BBU and RRU.

	Utilization				
Resource	**BBU**		**RRU**		**Available**
	Qty	**%**	**Qty**	**%**	
LUT	61076	20.12	87431	28.80	303600
FlipFlops	62593	10.14	91871	15.12	607200
Memory	4891	3.74	6142	4.70	130800
BRAM	33.5	3.25	98.5	9.56	1030
DSP48	4	0.14	52	1.86	2800
MMCM	2	14.29	2	14.29	14

The remaining of this chapter is organized as follows. Sections 4.1.1 shows the results of the compression implementation and Section 4.1.2 shows the performance of the synchronization.

4.1.1 Compression Evaluation

The compression algorithm was configured to a compression ratio of 4. Note that without compression the fronthaul is capable to transport two LTE 5 MHz signals with 30 bits per IQ sample. With this configuration, the required fronthaul bitrate is over 500 Mbps, taking into account the non-compressed IQ samples and headers overheads. If a compression ratio of 4 were applied to this LTE signal the required bitrate would decrease to approximately 125 Mbps. However, in this work, a more ambitious configuration was adopted using 20 MHz signals instead of 5 MHz.

An LTE signal with bandwidth of 20 MHz uses a sampling frequency of 30.72 MHz.

Assuming 30 bits per IQ sample, this LTE signal would require at least 921.6 Mbps for a single antenna. Since the analog front end is capable of transmitting two antennas, the total fronthaul bitrate should be at least 1843.2 Mbps. This bitrate would be impractical in a gigabit Ethernet. But with a compression ratio of 4 applied to the signal, it is possible to use the same fronthaul bitrate used to transport the non-compressed 5 MHz LTE signal. With compression, it was then possible to transport the 20 MHz LTE signal using a fronthaul with 500 Mbps. The decoded LTE signal, after fronthaul transport, decompression and RF transmission is shown in Figure 4.2.

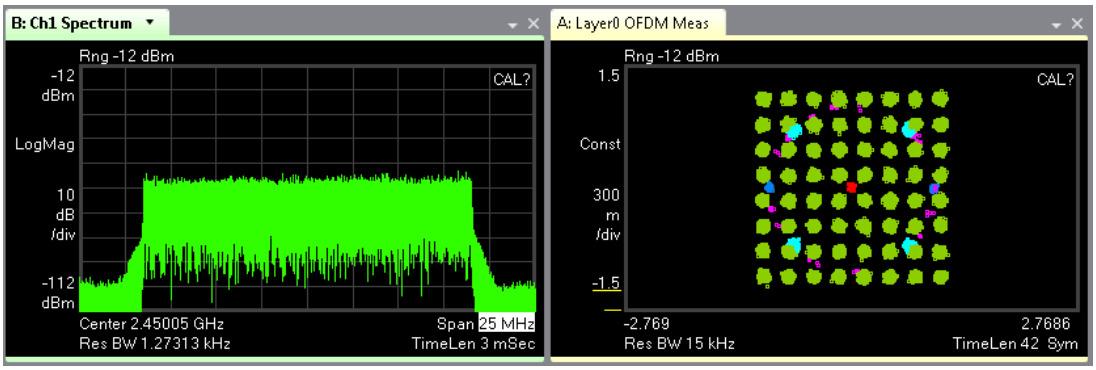


Figure 4.2: RF LTE signal, after fronthaul transport and decompression. The left graph is the spectrum of the signal, centred in 2.45 GHz. The right graph shows the constellation of the decoded physical channels used in LTE.

The applied compression algorithm has an impact in the EVM of the signal. For instance, in the downlink compressed signal, the measured EVMs for different LTE channels are presented in Table 4.2. For the compression ratio of 4, the MATLAB simulations show that only PUSQH compression presents an EVM of 2.65%. But, since the RF analog front end and the measurement equipment also adds distortion to the signal, the final EVM is in practice higher than in the simulation.

Table 4.2: EVM measurements break down of the different LTE channels [4]

Channel	P-SS	S-SS	PBCH	PCFICH	PHICH	PDCCH	C-RS	PDSCH
EVM (%)	3.0211	2.9577	3.1824	2.9099	2.6652	2.9484	3.2909	3.4006

Although the compressed setup carries two 20 MHz signals, the Ethernet link usage is around 527 Mbps. This measurement has been made based in the traffic passing in the switch port, and assumes that just the fronthaul is being transported in this port. In fact, based in the

frame structure of the Fronhtaul packets it is possible to derive the overheads and link usages for each protocol header as shown in Table 4.3.

Table 4.3: Ethernet link usage break down.

	Measured	Eth. Header	Meta-Data	CPRI CW	Compressed Data
Bytes Per Frame	284	14	14	16	240
Link Usage (%)	52.7%	2.6 %	2.6 %	3 %	44.5 %
Link Usage (Mbps)	527	26.0	26.0	29.7	445.3

The low measured EVMs shown numerically in Table 4.2 and visually in Figure 4.2 validates the compression algorithm implementation. In addition, over the measurement period, it was not verified any fronthaul transport error, such as packet loss or frame corruption. Also it is possible to observe the Ethernet load around 50%, while compressed stream transports two LTE 20 MHz as shown in the spectrum pannel of Figure 4.2.

4.1.2 The Phase Noise of recovered clock

The testbed has also been validated through the measurements of Phase Noise of the PTP synchronized clock. These measurements were made with the vector signal analyzer Keysight 9010A which has a specification of -116 dBc/Hz at 100 kHz. The PTP synchronized clock has a frequency of 8 kHz and is used as a reference to a jitter attenuator (Si5324) in order to generate a 40 MHz clock, as shown in Figure 4.3. The jitter-attenuated clock is used to drive the external ADC/DAC board. In addition, the same clock signal is used in the phase noise measurements shown in this work.

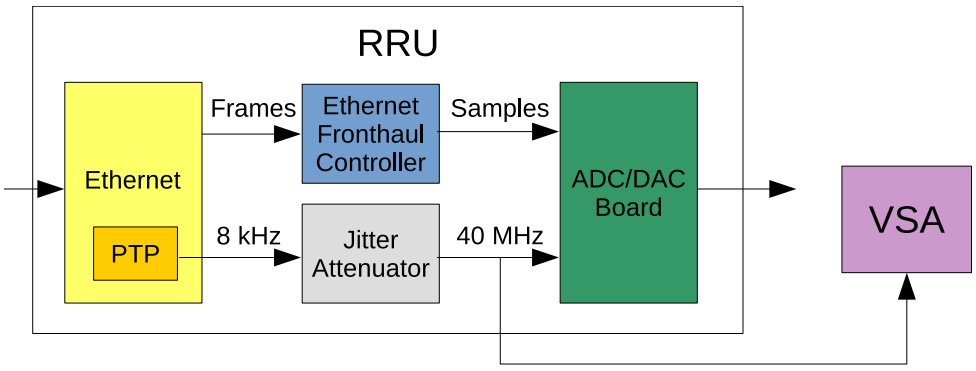


Figure 4.3: Clock and data path on the RRU.

The synchronized 8 kHz clock is derived from the PTP real-time clock which drives the timestamping unit. Since messages are traversed by legacy Ethernet, timestamps have a great amount of noise introduced by packet delay variations, specially due to queuing delay. Hence, it is common practice [40, 41, 42] to filter them to assure attenuation of noise on measurements.

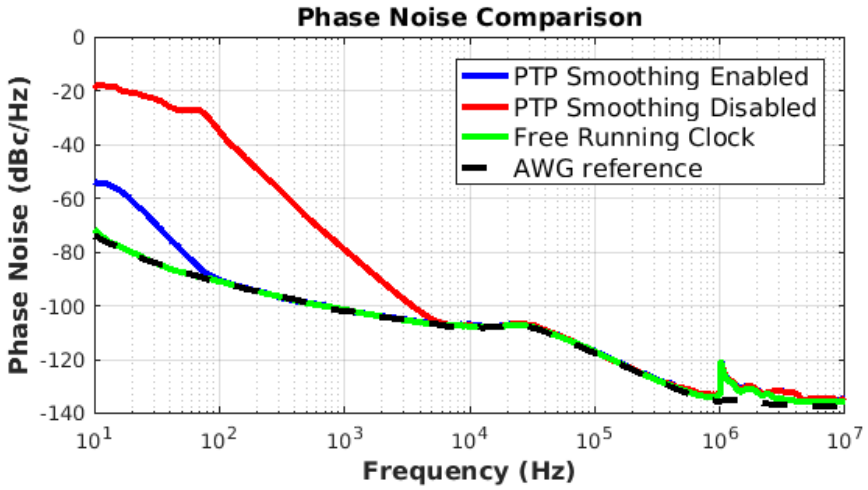


Figure 4.4: Phase Noise comparison between PTP, free-running clock and AWG reference.

Figure 4.4 compares phase noise measurements with and without the smoothing of time-offset estimations. It is possible to observe a difference of approximately 20 dBc/Hz between both phase noises. Furthermore, a reference measurement made with an Arbitrary Waveform Generator (AWG model 7082C) is presented for comparison, serving as a reference low-noise clock source.

Another relevant measurement is the phase noise difference between a PTP synchronized and a free-running (not disciplined) clock, which is also shown on Fig. 4.4. The free-running clock has a lower phase noise, which is expected due to the fact that the RTC used to generate the clock is not being frequently updated using time and frequency-offset estimations. However, in this case the fronthaul would not achieve time synchronization. Hence there is a tradeoff between phase-noise and timing alignment. The PTP corrections (primarily of time error) insert some noise aiming a better time alignment, so the processing made on PTP timestamps should be both precise for better timing performance and smooth for low phase-noise.

Finally, it is essential to evaluate phase-noise impairments due to the increase on the number of hops over the network link. Fig. 4.5 shows the results for one and two network hops when smoothing is enabled and disabled. It is possible to see that in this particular scenario, an

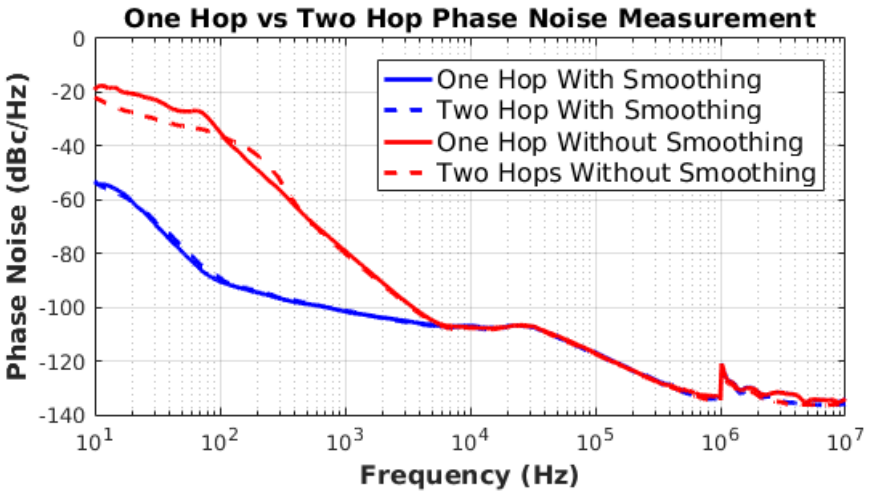


Figure 4.5: One Hop vs Two Hops Phase Noise measurement.

increment in the number of hops on the fronthaul link did not significantly impact the phase-noise.

Chapter 5

Conclusion

With the high requirements predicted for the 5th generation of mobile network, a review of the network architecture became necessary in order to provide the required features within the expected cost. The centralization of the network resources represents a favorable solution, but it is necessary to prepare the infrastructure for centralization with a new fronthaul network. Differing from the existing fronthaul, in the next generation, it is expected also to use the Ethernet infrastructure in order to bring low cost and more flexibility to C-RAN deployments.

In this scenario, this work presented an architecture of fronthaul based in the existing Ethernet standards. The architecture aims to evaluate the effects of CPRI encapsulation into Ethernet. As an extension, FPGA development boards have been used to deploy the architecture in a real Ethernet network in order to validate the effects of the Ethernet upon the fronthaul. Since the CPRI specification is used as base, the testbed focused in the full-centralization scenario, where all processing is centralized in the BBU while the fronthaul transport the IQ samples.

Moreover, in this work the described testbed was extended to compress the transmitted LTE signal through the PUSQH algorithm developed at UFPA. This work detailed the engineering solutions used to implement this algorithm in VHDL. In addition, the compression implementation was validated through the decompression and regeneration of the LTE signal after transport over the fronthaul. The compressed stream uses 4x less bandwidth being capable to carry more LTE signals in the same link while maintain the Error Vector Magnitude below the specifications.

In addition, in this work it was possible to visualize the application of Precision Time Protocol (PTP) to synchronize the network nodes. This synchronization method was validated through the evaluation of the Phase Noise on the PTP synchronized reference.

5.1 Future Works

Since the implemented testbed is essentially made by programmable resources (VHDL and C), it is possible to extend it to evaluate virtually any solution necessary for fronthaul over Ethernet. Moreover, it is possible to divide any future work in four main fields: Infrastructure, Compression, Synchronization and Transport. For each of these areas a list of improvements can be made.

The Infrastructure area is related to the testbed base architecture and the evolution of its features. The Compression area comprises the evolution of the compression techniques applied in the testbed. The Synchronization area involves the synchronization solution applied between the different endpoints of the network. The Transport area features the evolution on the transport of fronthaul traffic overt Ethernet.

Some possibilities are listed bellow.

- Infrastructure:
  - Improvements in the block implementation to provide more configuration flexibility.
  - Automatic evaluation and report of error conditions.
- Signal Compression:
  - Implement the compression operation in the BBU.
  - Implement the compression technique for LTE uplink.
  - Implement a network handshake for changes in compression algorithm in real-time.
- Synchronization:
  - Evaluate the utilization of Ethernet switches with PTP support.
  - Evaluate the impairments of network congestion in the synchronized reference.
- Transport:
  - Evaluate the utilization of Time Sensitive Network standards.
  - Study the utilization of streaming techniques to provide better transport.

5.2 Publication

Part of this work was published in the paper named: *FPGA-based testbed for synchronization on Ethernet fronthaul with phase noise measurements* [43]. Also, the aforementioned article was awarded with the best-paper award from the 1st INSCIT symposium at 2016.

Bibliography

[1] "Common Public Radio Interface (CPRI) specification v6.1," Available: http://www.cpri.info/downloads/CPRI_v_6_1_2014-07-01.pdf [Acessed 10 March 2017], jul, 2014.

[2] "Open base station architecture initiative (OBSAI) specification v2.0," Available: http://www.obsai.com/specs/OBSAI_System_Spec_V2.0.pdf [Acessed 10 March 2017], apr, 2006.

[3] U. Dötsch, M. Doll, H.-P. Mayer, F. Schaich, J. Segel, and P. Sehier, "Quantitative analysis of split base station processing and determination of advantageous architectures for LTE," *Bell Labs Technical Journal*, vol. 18, no. 1, pp. 105–128, 2013.

[4] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-Advanced for Mobile Broadband*, 1st ed. Academic Press, 2011.

[5] M. H. Callendar, "International Mobile Telecommunications-2000 Standards Efforts Of The ITU [Guest Editorial]," *IEEE Personal Communications*, vol. 4, no. 4, pp. 6–7, 1997.

[6] RAN, TSG, "Requirements for further advancements for E-UTRA (LTE-Advanced)," *June 200S*, 2008.

[7] ICT Data, "Statistics division," *Telecommunication Development Bureau, ITU." ICT facts and figures." Internet: https://www. itu. int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015. pdf. As of*, vol. 5, 2015.

[8] Cisco VNI Forecast, "Cisco visual networking index: Global mobile data traffic forecast update 2009-2014," *Cisco Public Information, February*, vol. 9, 2010.

[9] NGMN Alliance, "Suggestions on potential solutions to C-RAN," *White Paper, January*, 2013.

[10] K. Chen and R. Duan, "C-RAN: the road towards green RAN," *China mobile research institute*, vol. 2, 2011.

[11] L. Ramalho, M. N. Fonseca, A. Klautau, C. Lu, M. Berg, E. Trojer, and S. Host, "An LPC-Based Fronthaul Compression Scheme," *IEEE Communications Letters*, 2016.

[12] "Open radio equipment interface (ORI) release 4," Available: http://www.etsi.org/deliver/etsi_gs/ORI/001_099/001/04.01.01_60/gs_ORI001v040101p.pdf [Acessed 10 March 2017], apr, 2006.

[13] J. Liu, S. Xu, S. Zhou, and Z. Niu, "Redesigning fronthaul for next-generation networks: beyond baseband samples and point-to-point links," *IEEE Wireless Communications*, vol. 22, no. 5, pp. 90–97, 2015.

[14] T. E. Bogale and L. B. Le, "Massive MIMO and mmWave for 5G wireless HetNet: Potential benefits and challenges," *IEEE Vehicular Technology Magazine*, vol. 11, no. 1, pp. 64–75, 2016.

[15] A. Pizzinat, P. Chanclou, F. Saliou, and T. Diallo, "Things You Should Know About Fronthaul," *Journal of Lightwave Technology*, vol. 33, no. 5, pp. 1077–1083, mar 2015.

[16] M.-t. Hsieh and G. E. Sobelman, "Architectures for multi-gigabit wire-linked clock and data recovery," *IEEE Circuits and systems magazine*, vol. 8, no. 4, 2008.

[17] Open Base Station Architecture Initiative *et al.*, "BTS System Reference Document, Version 2.0," *URL: http://www. obsai. com/specs/OBSAI_System_Spec_V2. 0. pdf*, 2006.

[18] C. l. I, Y. Yuan, J. Huang, S. Ma, C. Cui, and R. Duan, "Rethink fronthaul for soft RAN," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 82–88, September 2015.

[19] "Next generation frothaul interface (1914) working group," Available: http://sites.ieee.org/sagroups-1914/ [Acessed 10 March 2017], mar, 2017.

[20] "Standard for Radio Over Ethernet Encapsulation and Mappings," Available: http://www.ieee1904.org/3/tf3_home.shtml [Acessed 10 March 2017], mar, 2017.

[21] *802.1Qbv-2015 - Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic*, IEEE Standard Association Std., 2017.

[22] *802.1Qbu-2016 - Bridges and Bridged Networks - Amendment 26: Frame Preemption*, IEEE Standard Association Std., 2017.

[23] *802.1Qca-2015 - Bridges and Bridged Networks - Amendment 24: Path Control and Reservation*, IEEE Standard Association Std., 2017.

[24] "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2002*, pp. i–144, 2002.

[25] *G.8262 : Timing characteristics of a synchronous Ethernet equipment slave clock*, International Telecommunication Union Std., July 2010.

[26] M. Gorgoglione, A. Dekorsy, and G. Fettweis, "Benefits and impact of cloud computing on 5g signal processing," 2014.

[27] S. Das, H. Viswanathan, and G. Rittenhouse, "Dynamic load balancing through coordinated scheduling in packet data systems," in *INFOCOM 2003. Twenty-second annual joint conference of the IEEE computer and communications. IEEE societies*, vol. 1. IEEE, 2003, pp. 786–796.

[28] D. Lee, H. Seo, B. Clerckx, E. Hardouin, D. Mazzarese, S. Nagata, and K. Sayana, "Coordinated multipoint transmission and reception in LTE-advanced: deployment scenarios and operational challenges," *IEEE Communications Magazine*, vol. 50, no. 2, 2012.

[29] Ericsson, "Cloud RAN the benefits of virtualization, centralization and coordination," 2015.

[30] P. Vaidyanathan, "The theory of linear prediction," *Synthesis lectures on signal processing*, vol. 2, no. 1, pp. 1–184, 2007.

[31] Xilinx, Inc., "AXI Reference Guide," mar 2011, UG761.

[32] Advanced Microcontroller Bus Architecture, "AMBA AXI Protocol Specification," mar 2004.

[33] I. Instrumentation and M. Society, "IEEE 1588-2008: Standard for a precision clock synchronization protocol for networked measurement and control systems," jul, 2008.

[34] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[35] Z. Aspar, Z. M. Yusof, and I. Suleiman, "Parallel Huffman decoder with an optimized look up table option on FPGA," in *TENCON 2000. Proceedings*, vol. 1. IEEE, 2000, pp. 73–76.

[36] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[37] Marvell, "88E1111 Product Brief - Integrated 10/100/1000 Ultra - Gigabit Ethernet Transceiver," oct 2013.

[38] Xilinx, Inc., "AXI 1G/2.5G Ethernet Subsystem v7.0," PG 138, sep 2015.

[39] "AD-FMCOMMS2-EBZ - AD9361 Software Defined Radio Board," Available: http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/eval-ad-fmcomms2.html [Acessed 10 March 2017], jul, 2014.

[40] M. Anyaegbu, C. X. Wang, and W. Berrie, "A sample-mode packet delay variation filter for IEEE 1588 synchronization," in *ITS Telecommunications (ITST), 2012 12th International Conference on*, Nov 2012, pp. 1–6.

[41] I. Hadžić and D. R. Morgan, "Adaptive packet selection for clock recovery," in *2010 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Sept 2010, pp. 42–47.

[42] ——, "On packet selection criteria for clock recovery," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Oct 2009, pp. 1–6.

[43] J. Paulo, I. Freire, I. Sousa, C. Lu, M. Berg, I. Almeida, and A. Klautau, "Fpga-based testbed for synchronization on ethernet fronthaul with phase noise measurements," in *2016 1st International Symposium on Instrumentation Systems, Circuits and Transducers (INSCIT)*, Aug 2016, pp. 132–136.