POLITECNICO DI TORINO

Department of Electronics and Telecommunications (DET)

Master Degree Thesis

DESIGN AND IMPLEMENTATION OF MIMO OFDM IEEE 802.11N RECEIVER BLOCKS ON HETEROGENEOUS MULTICORE ARCHITECTURE

Communications and Computer Networks Engineering

Supervisors:

Prof. Roberto Garello

Prof. Jari Nurmi

Dr. Daniel Riviello

Candidate:

Mohammad Hosseinvand

October 2018

TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MOHAMMAD HOSSEINVAND

DESIGN AND IMPLEMENTATION OF MIMO OFDM IEEE802.11N RECEIVER BLOCKS ON HETEROGENEOUS MULTICORE ARCHITECTURE

Master of Science Thesis

Examiners: Prof. Jari Nurmi
M.Sc. Sajjad Nouri

Examiners and topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on 30th of May 2018

ABSTRACT

MOHAMMAD HOSSEINVAND: Design and Implementation of MIMO OFDM IEEE802.11n Receiver Blocks on Heterogeneous Multicore Architecture
Tampere University of Technology
Master of Science Thesis, 60 pages
October 2018
Master's Degree Programme in Communications And Computer Networks Engineering
Major: Communication Systems and Networks
Examiners: Prof. Jari Nurmi
M.Sc. Sajjad Nouri
Keywords: Software-Defined Radio, WLAN, OFDM, MIMO, Heterogeneous, Application-specific Accelerator, Multicore, FFT, HARP, RISC Processor, Network-on-Chip, CGRA, COFFEE, FPGA, Reconfiguration, Time Synchronization, Frequency Offset Estimation, Channel Estimation, Symbols Demapping

In this thesis, the performance of a heterogeneous multicore platform in terms of technical capability is evaluated. Therefore, the choice of architecture in general can be based on a set of diverse applications. Selected applications can be parallel or serial in nature. Applications evaluation are often based on various performance metrics including the resource utilization and execution time. The wireless communication systems are expanded to accelerate their functions execution in both software and hardware. The embedded systems which involve several types of communication systems perform a large number of computations which require short execution time and minimized power consumption. Also, there is a growing demand for application-specific accelerators aiding general-purpose. One feasible way is to use heterogeneous multi-core platforms. Furthermore, many application-specific accelerators are loosely connected with each other.

In this study, the implementation of Multiple-Input Multiple-Output (MIMO) Orthogonal Frequency Division Multiplexing (OFDM) receiver is evaluated by applying a Heterogeneous Multicore Architecture (HMA). The MIMO OFDM receiver is composed of computationally intensive and general-purpose processing tasks and can serve maximum coverage for evaluation of the HMA. The receiver blocks are designed by crafting template-based Coarse-grained Reconfigurable Array (CGRA) devices. In this case study, four streams (antennas) are proposed in order to process the data over CGRAs simultaneously. HMA nodes will be reconfigured at run-time in different blocks of the receiver. In this experimental work, according to the performance of each CGRA, the collective performance of the entire platform as well as NoC traffic is recorded considering the number of clock cycles and also several high-

level performance criteria. The implementation of OFDM receiver scaled CGRAs to various dimensions. The data can also be exchanged between diverse nodes on the NoC structure by utilizing direct memory access (DMA) devices independently.

PREFACE

This thesis work presented here was accomplished in the Laboratory of Electronics and Communications Engineering at Tampere University of Technology, Tampere, Finland to pursue a Master of Science degree in the Information Technology Program in 2018.

At first, I would like to acknowledge my mother Soudabeh Memar, my father Asadollah Hosseinvand and my sister Maryam Hosseinvand for their patience and consistent struggle, passionate love, and tremendous support in all moments throughout my life to bring me up to a stage, at which I became able to conduct this thesis and write it. It is clear to me that I owe all my accomplishments to them and without their support, all this would not have been possible.

I would like to express my deepest gratitude and respect to my supervisor Prof. Jari Nurmi, who made possible accomplishment of this research work at Tampere University of Technology as well as he welcomed me in his research group, always his expertise, motivation and patience was a great value for my Master thesis. In addition, I appreciate his support, which helped me manage the achievement of my Master thesis.

I am really thankful to Prof. Roberto Garello for being my supervisors at Politecnico di Torino, Turin, Italy for the given opportunity, to believe in my abilities to do my thesis abroad, his enthusiastic support and always guided me to choose the best decision. I express my thanks to Dr. Daniel Riviello for providing valuable comments through this research.

I would like to express my warmest thanks to my dear friend, Yekta Lajevardi for endless support and for helping me to stay positive and focused. I am thankful to her for being an honest and lovely friend.

I am also grateful to my friend, Javad Malek Shahkoohi for long support and advices given through all the ups and downs of my studies. He has been my very caring and amazing friend. I would like to extend my gratitude to my friends at Tampere University of Technology, Ritayan Biswas, Alberto García, Marcelo Fierro, Bahareh SadeQian, Danial Parsa, Farhad Javanmardi, Kamran Mohamadi, Anastasia Yastrebova, Manlio D'Agostino, Giulia Sanfilippo for their friendship and good moments we have shared together and also, for their invaluable support throughout my Master degree. Living in Tampere would have been too colorless without you.

Tampere, October 2018
Mohammad Hosseinvand

نگفتم گفتنی ها رو تو هم هرگز نپرسیدی...

"I never told you what I should have and you have never asked"
- Ardalan Sarfaraz

I dedicate this thesis to my father and my mother, whose love and affection, support, encouragement, motivation and prayers of day and night make me capable to get honor and achievement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

ACK	ACKnowledgment
ADC	Analog-to-Digital Converter
AGC	Automatic Gain Control
ALM	Adaptive Logic Module
ALU	Arithmetic Logic Unit
ALUT	Advanced Look-Up Table
ASIC	Application Specific Integrated Circuit
ASK	Amplitude Shift Keying
ATM	Asynchronous Transfer Mode
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
CC	Clock Cycle
CCB	Core Configuration Block
CE	Channel Estimation
CFO	Carrier Frequency Offset
CGRA	Coarse Grain Reconfigurable Array
CIR	Channel Impulse Response
CISC	Complex Instruction Set Computing
CP	Cyclic Prefix
CPU	Central Processing Unit
CREMA	Coarse grain REconfigurable array with Mapping Adaptiveness
DAC	Digital-to-Analog Converter
DC	Delay and Correlate
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DSP	Digital Signal Processor
FEC	Forward Error Correction Code
FireTool	FIeld programming and REconfiguration management Tool
FPU	Floating Point Unit
FF	Flip Flop
FFT	Fast Fourier Transform
FOE	Frequency Offset Estimation
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FSK	Frequency Shift Keying

FSM	Finite State Machine
FU	Functional Unit
Gbps	Giga Bit Per Second
GCC	GNU Compiler Collection
GI	Guard Interval
GOPS	Giga Operation Per Second
GPP	General Purpose Processor
GUI	Graphical User Interface
HDD	Hard Decision-based Detection
HDL	Hardware Description Language
HMA	Heterogeneous Multicore Architecture
HW	HardWare
I	In-Phase
IEEE	Institute of Electrical and Electronics Engineers
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
I/O	Input/Output
IP	Intellectual Property
I/Q	In-phase and Quadrature-phase
ISI	Inter-Symbol Interference
LTS	Long Training Symbols
LUT	Look Up Table
MAC	Medium Access Control
MCM	Multi-Carrier Modulation
MIMD	Multiple-Instruction Multiple-Data
MIMO	Multiple-Input Multiple-Output
ML	Maximum Likelihood
MPSoC	Multi Processor System-on-Chip
MT	Mobile Terminal
MVM	Matrix Vector Multiplication
NOP	No-OPeration
NoC	Network-on-Chip
OFDM	Orthogonal Frequency Division Multiplexing
PAPR	Peak-to-Average Power Ratio
PCB	Peripheral Control Block
PE	Processing Elements
PN	Pseudo Noise
PPDU	Physical Protocol Data Unit
PSDU	Physical layer Service Data Unit

PSK	Phase Shift Keying
PTS	Partial Transmit Sequences
Q	Quadrature-phase
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
RISC	Reduced Instruction Set Computing
RPU	Reconfigurable Processing Unit
RTL	Register Transfer Level
RTOS	Real-Time Operating System
SDD	Soft Decision-based Detection
SDR	Software Defined Radio
SER	Symbol Error Rate
SIMD	Single-Instruction Multiple-Data
SISO	Single-Input Single-Output
SLM	SeLected Mapping
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip
STS	Short Training Symbols
TUT	Tampere University of Technology
TS	Time Synchronization
VC	Virtual Carrier
VHDL	Very-high-speed integrated circuit Hardware Description Language
VLIW	Very Long Instruction Word
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network

1. INTRODUCTION

Today's human society is reliant on computers more than ever before. Their use in the everyday life of the human society has lead to scientific advancements, events as well as new fields of science, which have prompted the formation of modern world. An integrated part of modern life is communication which has evolved greatly throughout the years. The latest means of communication is mobile phones which are categorized into embedded systems. Embedded processors are dominant in communication systems such as mobile phones in which short execution time is highly desired. Embedded processors are expected to run many concurrent applications. Reconfigurability is a solution for achieving the goal of short execution time, since many embedded applications are computationally intensive. At the same time power dissipation must be limited. Over the years, there has been a pattern to increase the performance of the system by scaling the frequency of single-core architectures. This led to considerable increase in power dissipation. This issue has caused vendors to offer multicore systems. However, Dark-Silicon [37] is a critical issue in multicore systems. We could overcome the Dark-Silicon challenge by introducing many application-specific accelerators that provide high performance at low frequencies.

In this thesis, the focus is on the template-based Coarse-Grained Reconfigurable Arrays (CGRAs), to generate special-purpose accelerators. These platforms have relatively low power consumption. CGRAs are recommended since they operate at very low frequencies while they can yield enormous performance improvements. CGRAs are reconfigurable and are programmed using a high level language such as C. Also, they contain many specialized accelerators and are designed in a way to perform massively-parallel tasks in critical applications.

Moreover, there are many heterogeneous platforms which have almost identical design philosophies, e.g. NineSilica [1], Platform 2012 [2] and MORPHEUS ([3], [4]). According to the guidelines published in [5], this study uses heterogeneous platform in an accurate testing condition to identify possible architectural problems. The tested units are Orthogonal Frequency Division Multiplexing (OFDM) receiver blocks which are computationally complex tasks. Time Synchronization (TS) and Fast Fourier Transform (FFT) are examples of computationally intensive tasks. In

addition, OFDM application as a test was used to evaluate the ability of the platform's general-purpose processing for algorithms like CORDIC algorithms and the Taylor series. The RISC processor performs general-purpose tasks while OFDM-specific tasks are carried out by the CGRA, and the two are interconnected using a Network-on-Chip (NoC) for complete OFDM functionality. Heterogeneous Multicore Architecture (HMA) platform used in this thesis has been designed to maximize computing resources to enhance the performance of many particular algorithms of various types. Heterogeneous Accelerator Rich Platform (HARP) is a specific instance of HMA platforms. It has nine nodes placed in a 3×3 topology.

The HARP platform includes a RISC core which performs two specific tasks; it acts as a system controller and distributes the configuration streams as well as the data to all other nodes that are interconnected through the NoC platform. After the configuration and data distribution tasks, the RISC core performs general computations and enforces synchronization between the nodes. All OFDM receiver block accelerators are designed based on the template-based CGRAs. Lastly, the OFDM receiver is evaluated to determine that it supports many communication systems algorithms.

1.1 Objective and Scope of Thesis

In this thesis, the feasibility of implementation and design of scalable CGRAs for MIMO OFDM running on a heterogeneous accelerator-rich platform is studied. This research work explores general issues as well as generation of application-specific accelerators for Software-Defined Radio (SDR) baseband processing utilizing the CGRA template. They are integrated with each other and then to a RISC core on a NoC. This thesis will be expanded to measurement, estimation and mapping of intensive signal processing algorithms. The main objectives of this thesis are to design and implement specific accelerators for an OFDM receiver baseband in a MIMO setup. The designed accelerator performance for each OFDM receiver block regarding the clock cycles, the use of resources and also the maximum operating frequency by synthesis on the family of Altera Stratix-IV FPGAs is addressed.

1.2 Thesis Outline

The thesis consists of chapters as follows; Chapter 2 overviews the literature. Chapter 3 presents the main part of OFDM system based on IEEE 802.11n standard where various approaches are explained for each OFDM receiver block. Chapter 4 describes

the platform architecture and template-based CGRAs. Additionally, the whole performance of the HARP platform and the nodes of NoC is explained completely. In Chapter 5, the design and execution of MIMO OFDM receiver blocks are explained by employing template-based CGRAs. Then Chapter 6 explains the integration of baseband processing blocks on HARP and the distribution of data between different nodes. Chapter 7 covers measurement and estimation of different levels of HARP related performance metrics. Finally, the last chapter presents the conclusion and future work.

2. LITERATURE REVIEW

Nowadays, the computationally intensive tasks are allocated to Multi-processor System-on-Chip (MPSoC) or accelerators based on the processor/co-processor model. Moreover, Coarse-Grained Reconfigurable Array (CGRA) is one of the most powerful classes of accelerators, which is suitable for signal processing applications by providing high throughput and parallelism. CGRAs contain a lot of gates, which makes sense when they are used most of the time [7]. BUTTER [8], Morphosys [12], ADRES [13] and PACT-XPP [14] are examples of CGRA. In order to complete different applications at the same time, the CGRAs working as coprocessors should be combined to make a heterogeneous multicore platform [15].

2.1 Processor/Co-processor Models

This section will briefly discuss processor and co-processor models. Regarding the history of the processors, it can be seen that single-core processors were used for the general-purpose approach of numerous applications. Additionally, it was used in some proprietary accelerators, including audio, video, etc. as well as computationally intensive applications. Throughout the years, different processors for a various set of requirements have appeared. It can be seen that Very Long Instruction Word machines (VLIW) have been grown in large-scale parallel applications [17]. Also, to support high-grade communication mobile applications, a combination of VLIW and Digital Signal Processing (DSP) architecture was developed by supporting multiple applications simultaneously [18].

Loosely Coupled (LC) accelerators communicate easily with a low bandwidth, which allows multiple accelerators to be connected to the processor. This model of accelerators can be obtained by connecting the accelerator to a system bus, on a local node or remote network. In this case, multiple accelerators can exchange data with each other, as well as they can also work concurrently. As a prototype of a loosely coupled architecture on NoC, the platform P2012 [2] can be expressed.

In the Tightly Coupled (TC) model, accelerators have high bandwidth for communication with the processor. This makes it possible to provide faster data transmis-

sion as well as synchronization. In this model, using a dedicated co-processor bus or directly integrating an accelerator in the processor data-path are viable solutions. A CGRA can be an example of an accelerator tightly coupled to a processor, which can utilize a network of switched interconnections.

2.2 Reconfigurable Devices

In recent years, reconfigurable architectures have become more popular platforms due to particular capabilities and abilities to perform computational tasks. Reconfigurable architectures have different levels of parallelism. Reconfigurability means modifying their functionality at run-time for various applications. Regarding the characteristics of reconfigurable computing systems, some of the most significant features can be as follows [20].

- **Reconfigurability:** This refers to altering the internal architecture for the purpose of running various applications at a high degree of performance.
- **Computation Model:** The computational models such as Single-Instruction Multiple-Data (SIMD) or Multiple-Instruction Multiple-Data (MIMD) can be used. Moreover, some systems may follow the Very Long Instruction Word (VLIW) model.
- **Granularity:** Refers to the data size for operations of Reconfigurable Processing Unit (RPU) of a system.

Considering a wide range of different models of reconfigurable devices, the reconfigurable devices can be categorized according to their granularity level into three different classes; Fine-Grained, Middle-Grained and Coarse-Grained [3].

2.2.1 Fine-Grained Devices

The FPGAs have been in the market for a few decades. They are well suited for fine-grained reconfigurable architectures. Logic Element (LE) is the smallest unit of processing in an FPGA which is composed of a Look-Up Table (LUT), a few Flip-Flops (FFs), 2-to-1 multiplexers and some logic gates. Two notable FPGA vendors are Xilinx [22] and Altera [21]. For Altera, the goal is to reach at higher synthesis frequencies in their tools, while Xilinx focuses on resource utilization [23]. MOLEN model is another fine-grained device which operates as a co-processor to a General-Purpose Processor (GPP) ([24], [25]).

2.2.2 Middle-Grained Devices

The word length of middle-grained devices is less than or equal to 8. On the other hand, irregular subword-length calculation increases trouble in the mapping of the algorithm, when the processing width is increased. There is a good compromise between the area and performance in this model. One of middle-grained devices is PiCoGA-III, which includes a matrix of Reconfigurable Data-path Units (RDUs), each of them composed of a 4-bit LUT, 4-bit ALU and 4-bit integer and Galois field multiplier [27], [28].

2.2.3 Coarse-Grained Devices

One of the most successful platforms in the academic research and industrial environment is CGRAs due to their high-level of granularity and also the number of diverse applications that can be targeted on them without difficulty. In fact, CGRAs have a record of processing numerous data parallel applications for academic research, e.g., Image and video processing ([8], [29]), Finite Impulse Response (FIR) filtering [30], Wideband Code Division Multiple Access (WCDMA) cell search [31] and Turbo Codes [32]. Applying the CGRAs provides access to a large bandwidth and high throughput. However, CGRAs can engage a wide area of a few million gates and have potentially high transient power dissipation [26]. XPP-III is one of the coarse-grained devices. Another CGRA platform is Adjustable Dynamic Embedded System (ADRES) as a reconfigurable array of 8×8 elements, which is strongly integrated with a VLIW processor [13]. Each of the processing elements in the ADRES includes Functional Units (FUs) and Register Files (RFs) linked to a mesh topology. The ADRES particular instances can be produced by utilizing an XML-based architecture specification language.

2.3 Multi-core Platforms

In this section, we will focus on the subject of multi-core platforms, consisting of both homogeneous and heterogeneous core models. In homogeneous model, the cores are all similar, in heterogeneous they are of different types. Additionally, other features of multi-core platforms (homogeneous and heterogeneous) are that they are C-programmable but heterogeneous platform cores may need extra customized tools compared to a simple compiler in C language, with data flow-level support. The following section will introduce some of the state-of-the-art platforms in detail.

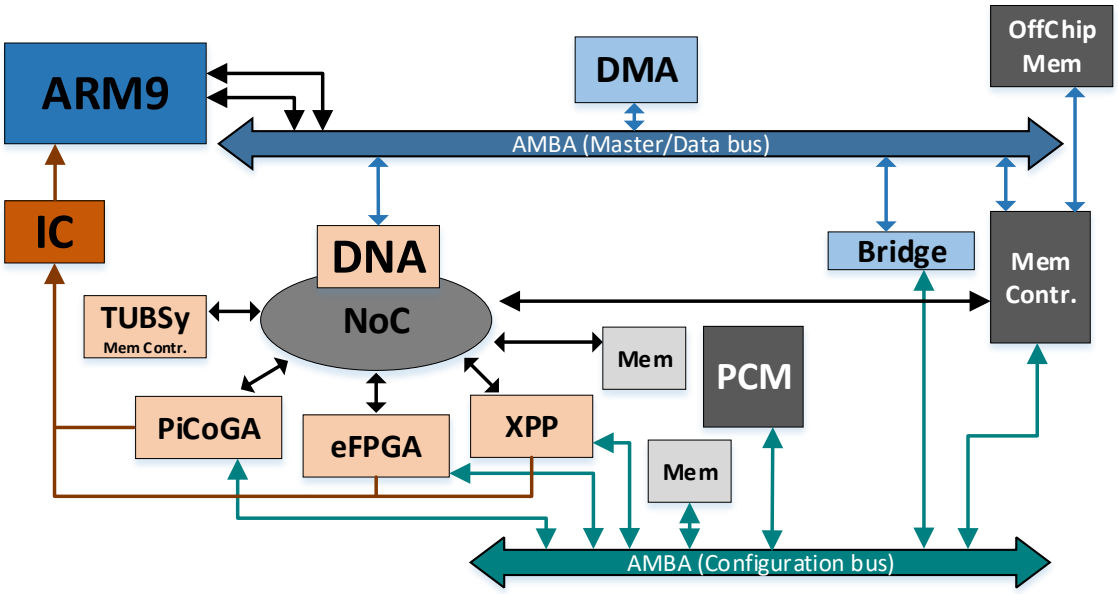

Figure 2.1 *MORPHEUS architecture [4]*

2.3.1 MORPHEUS

MORPHEUS ([3], [4]) is an integrated platform containing three different models: a fine-grained, middle-grained and coarse-grained reconfigurable accelerator while all of them are Heterogeneous Reconfigurable Engines (HREs). These devices are called FlexEOS, DREAM and XPP-III, and they can communicate together over an NoC. One of the suitable cases for fine-grained algorithms is the FlexEOS. Also, FlexEOS is an SRAM-based scalable FPGA that is programmable in VHDL. DREAM is a middle-grained reconfigurable DSP core that can carry out general-purpose processing by using a 32-bit RISC processor. PiCoGA core is a medium-grained reconfigurable array composed of 4-bit oriented ALUs, where up to four configurations may be kept concurrently in shadow registers [4].

XPP-III is one of the architectures of data processing based on CGRA and it can provide highly parallel processing performance. Also, XPP-III is a model of heterogeneous reconfigurable processor architecture composed of a dataflow array and VLIW processor. Figure 3.1 shows the MORPHEUS architecture. The Morpheus platform is a complex System-on-Chip that performs run-time programmability at different levels to provide a competitive computing solution [4] and dynamic reconfigurability.

2.3.2 P2012

Another homogeneous multi-core platform is P2012 including 16 general-purpose processors divided into four clusters communicating with each other by using a NoC ([2], [34]). Moreover, all the processors are locally synchronous in a cluster [2]. In other clusters, processors are globally asynchronous. The P2012 platform has been tested for algorithms related to signal processing.

2.3.3 NineSilica

NineSilica [1] is a homogeneous MultiProcessor System-on-Chip (MPSoC) platform. NineSilica platform includes a network of nine nodes placed in a mesh topology of 3×3 processing nodes (PNs) with three rows and three columns. The interconnection between PNs is done by a hierarchical Network-on-Chip (NoC). Each node of NoC includes a 32-bit COFFEE RISC processor. All the nodes can exchange data by packet switching technique [26]. Also, NineSilica is programmable in C language. Ninesilica architecture indicates that MPSoC can obtain high parallelization efficiency [35]. Many software-defined radio applications such as correlations and FFT can be designed and implemented over the NineSilica platform. The HARP platform used in this study is a heterogeneous derivative of NineSilica.

2.3.4 RAW

Reconfigurable Architecture Workstation (RAW) is composed of 16 slices of 32-bit MIPS2000 processors arranged in a 4×4 array ([36], [37]). The use of NoC has facilitated communication between processors. RAW provides both a static (determined at compile-time) and a dynamic network (determined at run-time: wormhole routing for the data forwarding) [38]. Its characteristics are similar to a reconfigurable fabric. Furthermore, programmable NoC in RAW has employed only one communication resource, resolving the wire selection problem from routing [38].

3. OFDM WLAN OVERVIEW

One of the spread spectrum techniques is Orthogonal Frequency-Division Multiplexing (OFDM). It divides the available bandwidth into several narrow-band channels, with orthogonal carriers. This modulation performs multiplexing operations by using frequency division. The orthogonality concept in frequency division refers to orthogonal signals, which returns to a mathematical definition in which, if two sinusoidal functions are multiplied, then the integral of this product is zero in any period of time. In fact, OFDM is a method of the general digital multi-carrier modulation to reach higher data rates close to the Shannon limit [42]. The advantage of this method is to send the data in parallel and to overcome the frequency selective fading because in this case, every part of the data is carried over a small range of frequency band. This kind of fading on this small interval practically appears linearly and can be compensated until the signal is eventually extracted.

The main benefits of using the OFDM method are its frequency selective fading due to multi-path propagation in wireless communication systems, narrowband interference, and reduction of the Inter-Symbol Interference (ISI). This means that the ISI can be decreased by transmission of several parallel symbols and increasing the symbol duration [43]. OFDM splits a higher bit rate encoded data stream into different streams of lower bit rate, then transmits them in parallel on different subcarriers all of which are orthogonal with respect to each other [39]. It is necessary to explain that for the purpose of maintaining orthogonality, both transmitter and receiver must use the same modulation method.

The benefits of OFDM are high spectral efficiency, adaptive modulation, and robustness against narrow-band co-channel interference [40]. Also, its disadvantages include the loss of efficiency due to the Cyclic Prefix (CP) and sensitivity to doppler shift [40]. The following section describes the OFDM structure consisting of transmitter, channel and receiver based on IEEE 802.11n specifications.

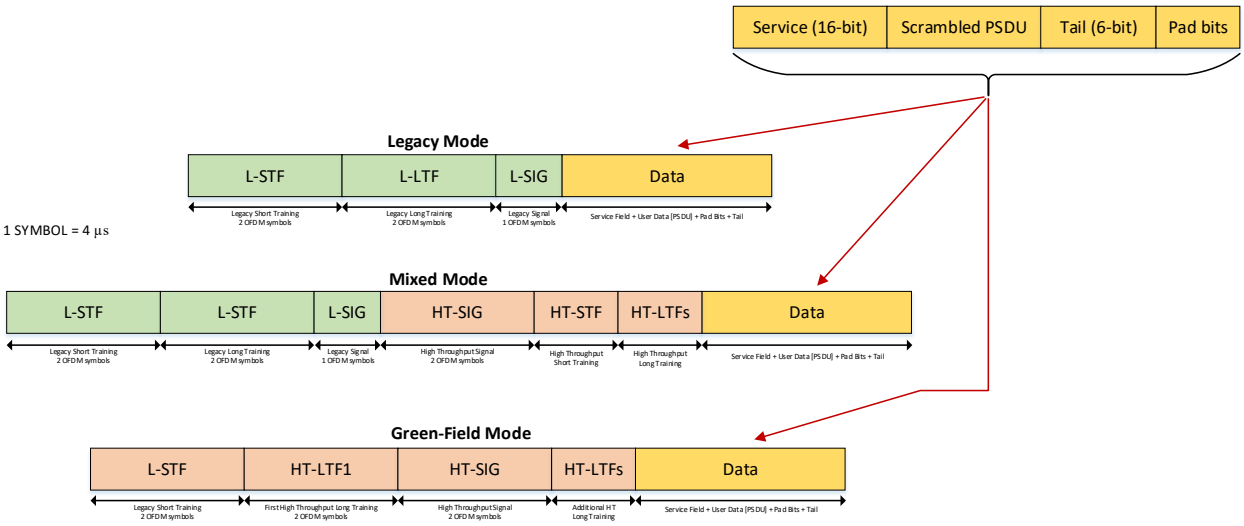

Figure 3.1 *IEEE 802.11n PPDU formats in Legacy, Mixed and Green-field modes [44]*

3.1 MAC Frame Structure for WLAN Standards

At present, there are three types of generally accepted WLAN standards in the world. They differ only in Medium Access Control (MAC). For this purpose, table 3.1 lists these standards [42]. The first two options are applied in Europe and North America, and the last option is used in Japan. IEEE802.11 will be explained in detail below.

Table 3.1 *Standards for OFDM WLANs [42]*

No	Standards	No	Type of MAC
1.	IEEE802.11	1.	Distributed MAC on the basis of Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) protocol
2.	HIPERLAN/2	2.	Centralized mode and scheduled MAC by means wireless Asynchronous Transfer Mode (ATM)
3.	MMAC	3.	Both the MACs listed above

Since the packet is transmitted, Mobile Terminal (MT) must wait for an Acknowledgment frame (ACK) that is required to avoid collisions. The received packet file header consists of information about the transfer rate, payload length, and transmission model, which is a modulation method.

According to the IEEE 802.11n specifications, MIMO OFDM receiver baseband processing is designed onto the HARP platform [33]. MIMO OFDM is a propitious technique to reach at high data rate, which includes benefits such as resilience to frequency selective fading caused by the multi-path propagation and to ISI [42].

Also, receiver generally carries out Time Synchronization (TS), Frequency Offset Estimation (FOE), Channel Estimation (CH), FFT and Symbols Demapping based on this standard. Predefined samples in preamble are known to the receiver [33]. Short training symbols can be used for packet detection, frequency offset estimation and timing synchronization. Furthermore, the long training symbols are used for channel estimation. The following subsections give a detailed explanation of these operations [33].

The structure of the IEEE 802.11n MAC frame is illustrated in Figure 3.1. The IEEE 802.11n standard supports the legacy IEEE 802.11a/b/g Physical Protocol Data Unit (PPDU) formats. In the IEEE 802.11n specifications, the PPDU can have several formats [44], depending on the abilities of the transmitter device such as:

- **Non-High Throughput (Non-HT) Legacy mode:** Composed of the preamble, which uses short and long training symbols as well as support for this format is compulsory for IEEE 802.11n standard. In addition, this may occur as either 20.0 MHz Bandwidth or a 40.0 MHz Bandwidth.
  - **20.0 MHz:** The signal has 64 subcarriers with 4 pilots. pilots are inserted in subcarriers ± 21 and ± 7. The signal is transmitted on sub-carriers -26 to -1 and $+1$ to $+26$ in the legacy mode.
  - **40.0 MHz:** For this model, two adjacent 20.0 MHz channels are employed. The signal has 128 subcarriers with 6 pilots. Pilots are inserted in sub carriers ± 53, ± 25 and ± 11. Also, the signal is transmitted on subcarriers -58 to -2 and $+2$ to $+58$.
- **HT-Mixed mode:** Preamble consist of the Non-HT short and long training symbol, which can be decoded by legacy mode in IEEE 802.11a/g. The HT-Mixed mode is compatible with IEEE 802.11a/g PLCP headers. Among other features of this mode, transmissions can occur both in 20.0MHz and 40.0MHz channels.
- **HT-Greenfield mode:** HT packet does not require a legacy compatible part to be transferred. As a result, the maximum data throughput is much higher.

The various parameters in the headers are [75]:

- Rate (4 bits): Signifies the type of modulation (8 combinations) and data Forward Error Correction (FEC) coding.

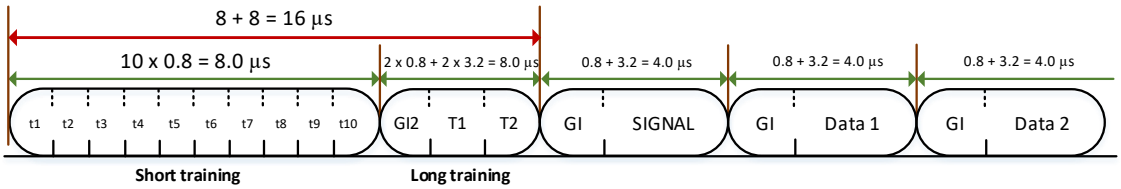

***Figure** 3.2 PLCP Preamble for OFDM training structure [44]*

- Length (12 bits): Number of octets (bytes) carried in the Payload in Physical Layer Service Data unit (PSDU).
- Tail (6 bits): Utilized for SIGNAL symbol FEC decoding.
- Service (16 bits): To synchronize the descrambler, bits from 0 to 6 should be set to zeros and reserve the last 9 bits for subsequent purposes.
- Parity (1 bit): Parity-check on RATE + LENGTH data.
- Pad bits: Variable-length field needed to guarantee that the non-HT and HT-Data data field, including an integer number of symbols.

Figure 3.2 shows the PLCP header includes a preamble, signal and data field. There are 10 short training symbols and 2 long training symbols in the preamble. It can be noted that the length of both training symbols is 8.0 μs with the total time of 16.0 μs.

L-STF consists of a sequence of tones, which belong to the values $+1+j$ and $-1-j$. These tones are performed on a small part of the sub-carrier, whilst the other sub-carriers will reach zero. The reasons behind this choice are that there are properties of best correlation and also low peak-to-average power ratio, which means L-STF is actually used for automatic gain control in MIMO transmission, as well as for fine-tuning the time synchronization. Furthermore, a 128-point IFFT is essential for the purpose of making a time domain sequence in the transmitter side. L-STF is known for the receiver side. For instance, time acquisition or packet detection are done by utilizing its properties of correlation peaks [43]. In addition, due to the iteration of the samples, frequency offset estimation is required, which will be described later. The sequences in frequency domain for the 20.0MHz and the 40.0MHz bandwidths are similar to the ones in the IEEE 802.11n specification as shown in Equations 3.1

and 3.2 [44].

$$\begin{aligned} S_{L-STF(-26,26)} = \sqrt{\frac{1}{2}}\{&0,0,1+j,0,0,0,-1-j,0,0,0,1+j,0,0,0,-1-j,0,0,0,\\ &-1+j,0,0,0,1+j,0,0,0,0,0,0,0,-1-j,0,0,0,-1-j,\\ &0,0,0,1+j,0,0,0,1+j,0,0,0,1+j,0,0,0,1+j,0,0\} \end{aligned} \tag{3.1}$$

$$S_{L-STF(-58,58)} = \{S_{L-STF(-26,26)},0,0,0,0,0,0,0,0,0,0,0,0,S_{L-STF(-26,26)}\} \tag{3.2}$$

Long Training Sequence (LTS) for 40.0MHz bandwidth mode is another preamble sequence, which has similar long symbols T1 and T2 as in Figure 3.2, as well as a guard interval (GI2) with duration of 1.6 μs ahead of these two symbols for coping with the phenomenon of Inter-Symbol Interference (ISI). Both T1 and T2 symbols take 3.2 μs, which means that the duration of the long training sequence (LTS) is equivalent to 8.0 μs.

$$T_{Long} = 1.6 + 2 \times 3.2 = 8.0$$

Furthermore, T1 and T2 will be defined in the frequency domain with 115 sub-carriers, which are modulated instead of 57 sub-carriers in 20.0MHz (based on the definitions of the legacy long training). The resulting sequence is as follows [45]

$$\begin{aligned} S_{L-LTF(-58,58)} =\{&1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,1,-1,-1,1,1,-1,1,\\ &-1,1,1,1,1,1,1,-1,-1,1,1,-1,1,-1,1,-1,-1,-1,-1,-1,\\ &1,1,-1,-1,1,-1,1,-1,1,1,1,1,-1,-1,-1,1,0,0,0,-1,1,\\ &1,-1,1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,1,-1,-1,1,1,-1,\\ &1,-1,1,1,1,1,1,1,-1,-1,1,1,-1,1,-1,1,-1,-1,-1,-1,-1,1,\\ &1,-1,-1,1,-1,1,-1,1,1,1,1\} \end{aligned} \tag{3.3}$$

Channel estimation (CE) is done in IEEE 802.11n specification utilizing the LTF fields at the beginning of each packet as well as LTF fields will be applied for more precise time synchronization and frequency offset estimation [43].

The next field in the PLCP header is the signal field containing information about TXVECTOR length and coding rate. The signal field in legacy mode informs to the receiver about the type of modulation used in the system and also the rate of coding along with the packet data length. The L-SIG field is composed of an OFDM

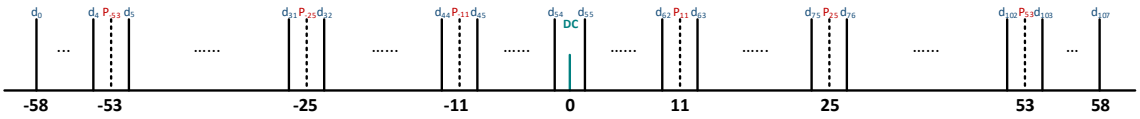

Figure 3.3 *Subcarrier frequency allocation for 40.0 MHz with 128 subcarriers [44]*

symbol, which is always BPSK modulated, as well as the length of OFDM symbol is 4.0 μs in legacy mode. The High Throughput Signal (HT-SIG) field is composed of the information about the burst. It operates the same task as the L-SIG field unless used with one of the HT modes. In that case, it operates two OFDM symbols.

For a 40.0MHz transmission, there are 114 subcarriers in each data OFDM symbol that contains 108 data subcarriers and 6 pilot subcarriers. In other words, six of the subcarriers are allocated to pilot signals for the purpose to create the coherent detection, robust against phase noise and frequency offsets. As it should be noted, in order to prevent the production of spectral lines, pilots are produced as a sequence of pseudo-binary. Thus, pilot signals should be inserted in subcarriers ± 53, ± 25 and ± 11. Based on Figure 3.3, it can be seen as the sub-carrier frequency allocation.

Equation 3.4 shows the pilot sequence for symbol n and space-time stream i_{STS} [44].

$$
\begin{aligned}
P_{-58,58} = \Big\{ & 0,0,0,0,0,\Psi^{(N_{STS})}_{i_{STS},n\oplus 6},0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\\
& 0,\Psi^{(N_{STS})}_{i_{STS},(n+1)\oplus 6},0,0,0,0,0,0,0,0,0,0,0,0,0,0,\Psi^{(N_{STS})}_{i_{STS},(n+2)\oplus 6},0,0,0,0,0,0,\\
& 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\Psi^{(N_{STS})}_{i_{STS},(n+3)\oplus 6},0,0,0,0,0,0,0,0,0,\\
& 0,0,0,0,\Psi^{(N_{STS})}_{i_{STS},(n+4)\oplus 6},0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\\
& 0,0,0,0,0,0,0,0,0,0,0,0,\Psi^{(N_{STS})}_{i_{STS},(n+5)\oplus 6},0,0,0,0,0\Big\}
\end{aligned}
\tag{3.4}
$$

Here the patterns $\Psi^{(N_{STS})}_{i_{STS}}$ are determined in Table 3.2 [44] and $n \oplus a$ symbol represents the integer number a of the module. The point is that there is a different pilot pattern for each space-time stream. These patterns are cyclically rotated over the symbols. Subsequently, according to the design parameters in this thesis, the contribution of the pilot sub-carriers for each OFDM symbol is generated by Fourier

Table 3.2 *Pilot specific values for 40.0MHz [44]*

N_{STS}	i_{STS}	$\Psi^{(N_{STS})}_{i_{STS},0}$	$\Psi^{(N_{STS})}_{i_{STS},1}$	$\Psi^{(N_{STS})}_{i_{STS},2}$	$\Psi^{(N_{STS})}_{i_{STS},3}$	$\Psi^{(N_{STS})}_{i_{STS},4}$	$\Psi^{(N_{STS})}_{i_{STS},5}$
1	1	1	1	1	-1	-1	1
2	1	1	1	-1	-1	-1	-1
2	2	1	1	1	-1	1	1
3	1	1	1	-1	-1	-1	-1
3	2	1	1	1	-1	1	1
3	3	1	-1	1	-1	-1	1
4	1	1	1	-1	-1	-1	-1
4	2	1	1	1	-1	1	1
4	3	1	-1	1	-1	-1	1
4	4	-1	1	1	1	-1	1

transform of sequence $P_{-58,58}$ based on Equation 3.5.

$$
\begin{aligned}
P_{-58,58} = \{&0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,\\
&0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\\
&0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,\\
&0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0\}
\end{aligned}
\tag{3.5}
$$

Table 3.3 *IEEE 802.11n OFDM parameter values [45]*

Parameter	**Non-HT**$_{(20.0MHz)}$	**HT**$_{(20.0MHz)}$	**HT**$_{(40.0MHz)}$
FFT length	64	64	128
N_{SD}: No.data carriers	48	52	108
N_{SP}: No. pilot carriers	4	4	6
N_{ST}: Total no. carriers	52	56	114
Δ_f: Carrier spacing	312.5 kHz	312.5 kHz	312.5 kHz
T_{DFT}: IDFT/DFT period	3.2 μs	3.2 μs	3.2 μs
T_{GI}: Guard interval	0.8 μs = $T_{DFT}/4$	0.8 μs	0.8 μs
T_{GI2}: Double GI	1.6 μs	1.6 μs	1.6 μs
T_{L-STF}: L-STF duration	8.0 μs = $T_{DFT}/4$	8.0 μs	8.0 μs
T_{L-LTF}: L-LTF duration	8.0 μs = $2 \times T_{DFT} + T_{GI2}$	8.0 μs	8.0 μs
T_{SYM}: OFDM symbol duration	4.0 μs = $T_{DFT} + T_{GI}$	4.0 μs	4.0 μ
T_{L-SIG}: L-SIG field duration	4.0 μs = T_{SYM}	4.0 μs	4.0 μ
T_{HT-SIG}: HT-SIG field duration	N/A	8.0 μs = 2 T_{SYM}	8.0 μs
T_{HT-STF}: HT-STF field duration	N/A	4.0 μs	4.0 μs
T_{HT-LTF}: HT-LTF field duration	N/A	4.0 or 8.0 μs	4.0 or 8.0 μs

The types of modulation of data can be BPSK, QPSK, 16QAM and also 64QAM. They are similar for each burst, as well as the length of IFFT per symbol that is 128. Based on the 40.0MHz transmission for all 128 subcarriers, the frequency range

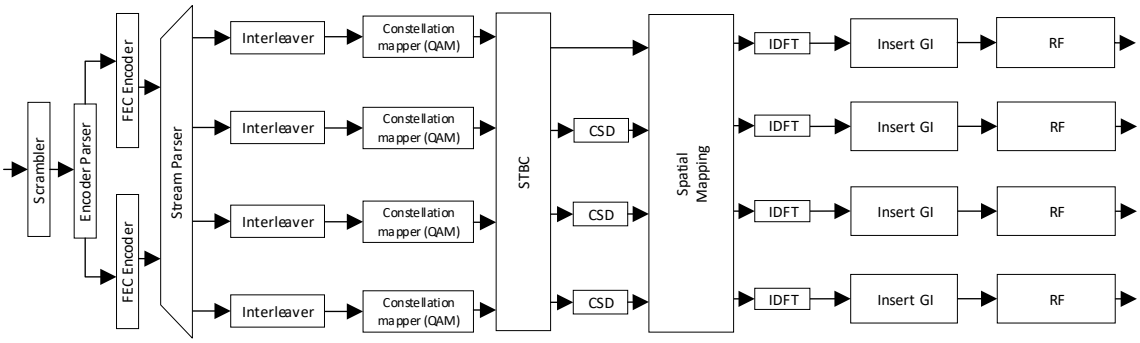

Figure 3.4 *Block diagram of IEEE 802.11n transmitter [44]*

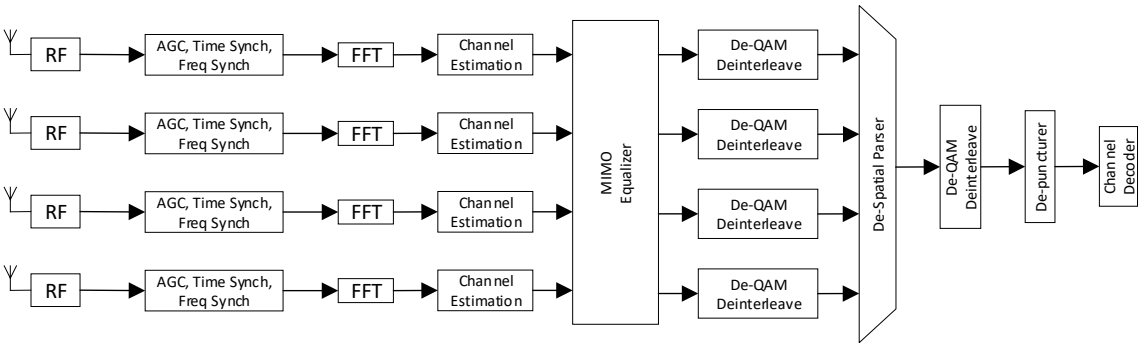

Figure 3.5 *Block diagram of IEEE-802.11n receiver [10]*

of each subcarrier is 312.5 kHz. According to Table 3.3, the timing parameters associated with the IEEE 802.11n signal are listed for different conditions [45].

3.2 Physical Layer Specifications for WLAN Standards

In broadband communication system, nowadays, OFDM is vastly utilized to tackle the frequency-selective fading. The transmitter, receiver and channel are blocks of an OFDM system. In IEEE 802.11n specification, the transmitter and receiver blocks are shown in Figures 3.4 and 3.5. The scrambler block that reduces the probability of long sequences of zeros or ones [45]. The encoder parser which de-multiplexes the scrambled bits among N_{ES} (number of FEC encoders). Forward Error Correction (FEC) encoder block is used for controlling errors in data transmission. The stream parser splits the encoder's output into blocks which are sent to (N_{SS} number of spatial streams) different interleaver and mapping devices. To avoid long sequences of adjacent noisy bits, the interleaver block interleaves the spatial stream bits. Also, the constellation mapper maps the sequence of bits in each spatial stream to constellation points. For each spatial stream, the modulation is carried out separately.

The modulation processes are two-dimensional, they are used for carrier waves of

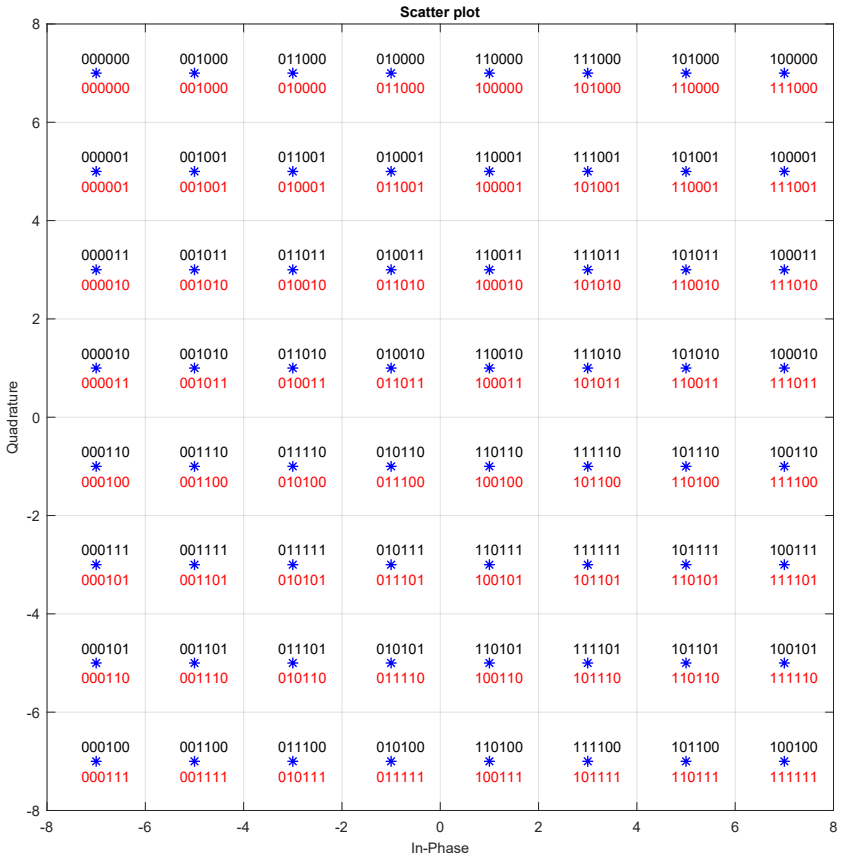

Figure 3.6 *QAM naturally ordered (Red color) and Gray coded (Black color)*

In-phase (I) and Quadrature (Q). Equation 3.6 shows a QAM modulation which is a combination of two modulations, Phase Shift Keying (PSK) and Amplitude Shift Keying (ASK). Moreover, 64-QAM modulation is used in this thesis work.

$$s(t) = I_k cos(\omega_c t) - Q_k sin(\omega_c t) = A_k cos(\omega t + \phi_k) \tag{3.6}$$

By designating a mapping, all constellation points must be labeled. There are two ways to consider to solve this issue; Grey-coding (Black color) or by natural order (Red color) as illustrated in Figure 3.6. However, there are variations between these two ways. Natural coding allows decimal numbers ordered from 0 to 63, but in Gray-coding, the adjacent representations (symbols) differ by only one bit. Therefore, two-bit errors which are also the most typical type of error between neighboring points can be reduced by applying Gray-coding, which reduces Bit Error Rate (BER) and Symbol Error Rate (SER) [42].

Then there is the space-time block coding (STBC), which is a method used in wireless communications systems to transmit multiple copies of a data stream among

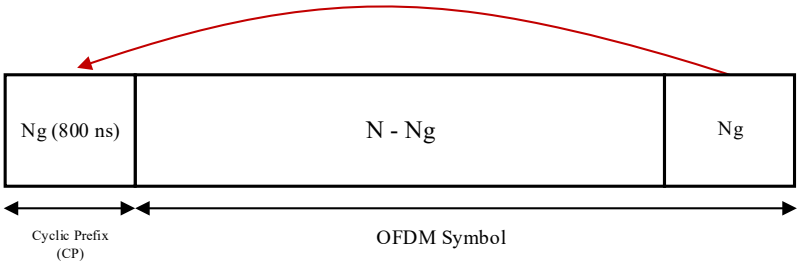

Figure 3.7 *Cyclic Prefix (CP) in OFDM Symbol [45]*

a number of antennas. In IEEE 802.11n, STBC is applied to extend the spatial stream to double their number of space-time streams (STBC is not always used). In the cyclic shift diversity (CSD) block, cyclic shifts are used to avoid unintentional beam-forming. The values of the shifts used in the preamble fields and the data field may be different. Then, in the spatial mapper block, it can be stated that spatial mapper extends the space-time streams into a number of transmit chains.

The next block and one of the most important ones is the Inverse Discrete Fourier Transform (IDFT) that will be briefly described below. With the help of IDFT, a block of constellation points in frequency domain is converted to a time domain block. On the other hand, Discrete Fourier transform (DFT) is defined as *X(k)* in the frequency domain based on Equation 3.7 and also according to the sequence of N samples *x(n)* in the time domain. Based on Equation 3.8 [40], IDFT is used to calculate *x(n)* from *X(k)*.

$$X_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \tag{3.7}$$

$$x_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X(k) e^{j2\pi nk/N} \tag{3.8}$$

Therefore, IFFT is done on the frequency domain QAM subcarriers to produce time domain sum of sinusoids. Subsequently, these operations will be reversed in the receiver side in order to retrieve the original data by using the Fast Fourier Transform (FFT).

The next block is Guard Interval (GI) Block. When the IFFT is carried out, GI or CP will be added to the IFFT output [42]. In other words, guard interval insertion prepends to the symbol a circular extension of itself as shown in Figure 3.7. Accordingly, to add a CP, 32 samples (0.8 μs) from the end of the OFDM symbol are annexed to the beginning of the OFDM symbol. The OFDM symbols must be

coordinated by a GI or CP to resist InterSymbol Interference (ISI) as well as time synchronization errors [45]. It should be noted that the ISI is basically created by receiving multiple copies of the transmitted signal due to multi-path effects and channel dispersion [47].

To clarify the issue, it will be assumed that there are two OFDM symbols, the last part of the first OFDM symbol makes interference with the first part of the second OFDM symbol upon it is received. Thus, according to the above-mentioned conditions, the amplitude and phase of the sub-carriers may deviate. Subsequently, the cyclic prefix is very important in terms of solving this problem. The delay portion of the first OFDM symbol is absorbed through the cyclic prefix of the second OFDM symbol [48]. In the windowing section, it is optionally used for smoothing the edges of each symbol in order to increase the spectral decay. After adding CP, preambles are produced which consists of short and long training symbols. Also, before transmitting the signal on the air interface by antennas, the signal must be converted from digital to analog through a digital-to-analog converter (DAC). Since the samples go through the DAC, a renovation filter is needed to eliminate the replication of the spectrum, which makes the design of this model much simpler.

In the cellular wireless communications, the transmission channel causes various unwanted changes in the signal of information resulting from reflections and diffractions. Indeed, these changes may cause noise, interference, cancellation and distortion in the systems. In addition, the channel can be described as a linear time-invariant transfer function with Additive White Gaussian Noise (AWGN). The received signal is as $y(t) = x(t) + n(t)$, because the noise $n(t)$ is added to transmitted original signal $x(t)$. The signal-to-noise ratio (SNR), the signal strength to noise, is measured in (dB) unit. Based on Equation 3.9, SNR is the ratio between the power of the transmitted signal and the undesirable noise.

$$SNR_{dB} = 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) = 10\log_{10}(P_{signal}) - 10\log_{10}(P_{noise}) \tag{3.9}$$

Figure 3.8 shows the Power Spectral Density (PSD) of the OFDM system in terms of the amount of SNR according to the IEEE 802.11n. PSD is the frequency response of a random or periodic signal. It tells us where the average power is distributed as a function of frequency. In other words, PSD is distribution of power, and it can be calculated by Fourier Transform of auto-correlation function of the signal. Therefore, the quality of the signal is improved by increasing SNR.

The first parts of the receiver are mostly used to detect the synchronization, estimate

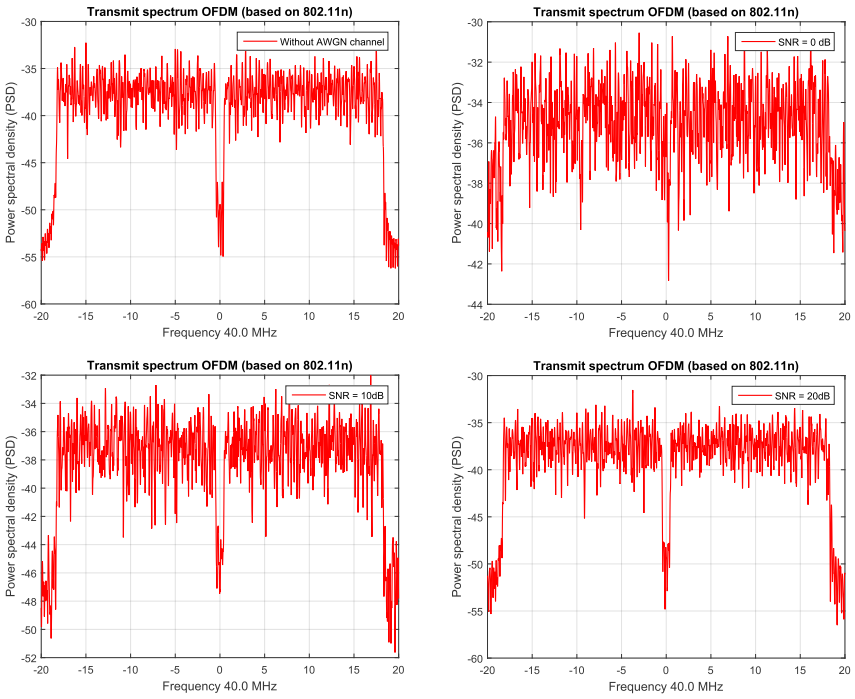

Figure 3.8 *Transmit spectrum of OFDM (PSD) based on IEEE 802.11n standard*

the channel and equalize the symbols, whereas the remaining of the receiver blocks must be done in reverse order in the transmitter. The analog signal received for the first time is sampled and converted to a digital signal by an Analog to Digital Converter (ADC). The gain will be adjusted to an appropriate input signal level by employing the Automatic Gain Control (AGC). When the ADC is accomplished, the next block has the task of detecting packets and time synchronization. Generally, detection of the packet serves to detect the beginning of the packet, which can be performed by utilizing correlation with the short training symbols. Additionally, the time synchronization could be done using the VHT-STF field that can determine the starting point of received packets by the correlation the delayed version of itself or the inbound packet with known training symbols.

After removing the cyclic prefix, frequency offset estimation is needed to estimate the amount of frequency offset that is added to the transmitted signal on the channel, which means that the frequency offset is estimated from the VHT-STF field and corrected for the entire stream of data. Then, the stream is divided into symbols and the guard interval is removed.

The next block is channel estimation for estimating the channel impulse response by comparing the received pilots and known transmitted ones. The channel estimation of MIMO is carried out by using the VHT-LTF fields. These training symbols are

transmitted in each stream with various polarities, making them orthogonal to each other, and then the receiver has the ability to perform a channel estimation evaluation for each sub-carrier separately. Once pilots are the same in all streams, it can be stated that they are not orthogonal to each other. On the other hand, channel estimation for the subcarrier of pilots cannot be estimated by utilizing the VHT-LTF as the other subcarriers. As an alternative, interpolation between surrounding subcarriers is used to estimate the channel for the subcarriers of the pilot.

In the following receiver blocks, a linear equalization algorithm employs a reverse of the frequency response of the channel to the received signal using a Zero-Forcing equalizer [45]. Based on this model of the equalizer, it will eliminate the whole of ICI and it is ideal when the channel is without noise. When the channel is in noisy environments, the zero-forcing equalizer amplifies the noise strongly at frequencies where the channel response has a low magnitude. The equalizer converts a number of multiplexed received chains into a number of equalized space-time streams where each space-time stream provides a pilot tracker.

In the next block, the demodulation extracts the original transmitted bits from the received modulated constellations and the deinterleaver reverses the process of interleaving. Interleaver block interleaves the bits of each spatial stream to prevent long sequences of adjacent noisy bits. The block deinterleaver performs the inverse operation of the interleaver. Also, the demodulated bits are crossed through the stream deparser. In the last stages, the data bits are depunctured with appending dummy zeros in the locations where encoded bits were punctured, and the symbols are transformed into a bit stream.

3.2.1 Time Synchronization

In OFDM systems, the time estimation block is used for two specific tasks which are packaging detection and symbol timing synchronization. When there is no information about the starting point of the received packet, packet detection will be required for OFDM systems. Also, in order to find the exact start point of of the OFDM symbols which determines the true position of FFT window, time synchronization is needed [55]. Correlation algorithm implies to the similarity between two signals. There are two types of correlations which are auto-correlation algorithm and cross-correlation algorithm.

An auto-correlation algorithm is the correlation between a signal with its delayed version or its shifted version. The cross-correlation algorithm refers to the correlation between two different signals. Also, correlation algorithm is intensive when it is

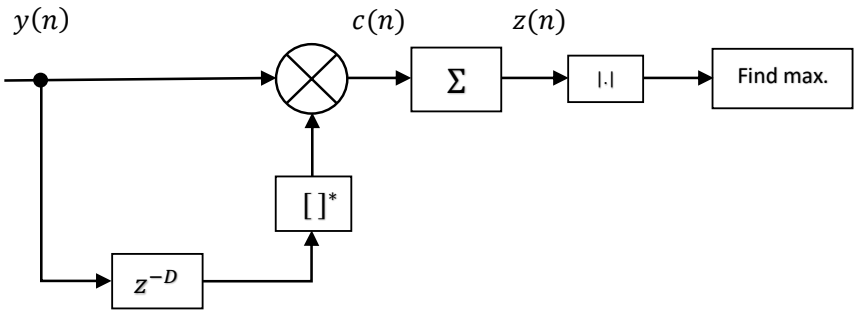

Figure 3.9 *Block diagram of correlation algorithm for time synchronization [55]*

.

computed in time domain. So, different fast algorithms are carried out to reduce the time consumed. Packet detection is determined by applying delay and correlation [42]. The output of this algorithm, c_n is shown in Equation 3.10.

$$c_n = \sum_{k=0}^{L-1} y_{n+k} y^*_{n+k+D} \tag{3.10}$$

Note that y_n shows the received packet, D equals 32 (in IEEE 802.11n with 64-QAM modulation) and L stands for the length of correlation. The time synchronization block can be implemented by using two different methods as follows [54]; first, using special symbols like training symbols or null symbols, second, cyclic prefix (CP) or Guard Interval (GI) correlation algorithm. The start point of the actual data including OFDM symbols can be detected by transmitting a particular symbol, that is known to the receiver, by the transmitter in the first method. The end of short or long training symbols of a received data packet which can be seen in Equation 3.11 is used for timing synchronization in IEEE 802.11n.

$$z_n = \sum_{k=0}^{L-1} y_{n+k} t^*_n \tag{3.11}$$

Note that y_n is the received signal, t_n stands for the known symbols and (*) shows the complex conjugate operation. As the most common way in OFDM systems, cyclic prefix method is applied when the data content is not clear. Since CP or GI is used for combatting against Inter-symbol interference (ISI). Figure 3.9 shows the signal flow structure. The amount of delay z^{-D} equals CP length which is 32 according to IEEE 802.11n standard specifications. An output c_n and z_n which are given by

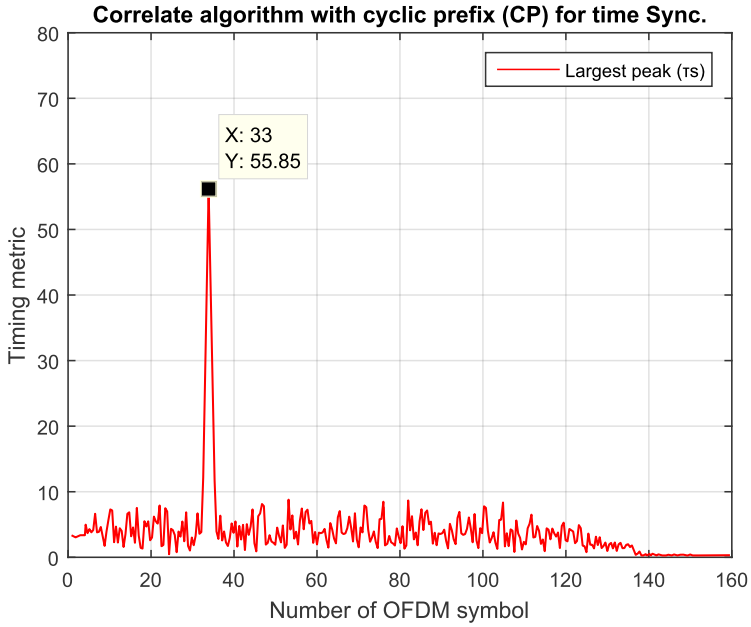

Figure 3.10 *Cyclic prefix (CP) correlation along with SNR 20 dB*

.

Equations 3.12 and 3.13 are produced in this method.

$$c_n = c_n y^*_{n-D} \tag{3.12}$$

$$z_n = \sum_{i=0}^{L-1} c_{i+n} \tag{3.13}$$

Depending on Equation 3.14, when the correlation finishes, its largest peak should be taken to estimate the index of time offset specifying the first FFT window edge.

$$\hat{\tau_s} = \underset{n}{\operatorname{argmax}} \mid z_n \mid \tag{3.14}$$

Treatment of $|z_n|$ in a noisy channel with no multipath propagation can be observed in Figure 3.10. If there is not any multipath propagation and received data symbol length in IEEE 802.11n equals 160 (128 FFT and 32 CP), the data symbol correlated with itself has one peak ($\hat{\tau_s}$) such that the length of the cyclic prefix equals the peak location minus one. Corresponding to cyclic prefix removal, samples before the peak value are skipped when the time offset is found.

3.2.2 Frequency offset Estimation

FOE will be discussed in this section according to IEEE 802.11n standard. OFDM waveform is made of multiple sinusoidal components. Before transmission, a signal is upconverted to carrier frequency. The received signal on the receiver is downconverted to demodulation from the same carrier frequency prior. The OFDM's sensitivity to carrier frequency offset is one of its drawbacks which causes device impairments [42]. Based on Equation 3.15, f_Δ is the difference between the carrier frequencies on the transmitter and receiver side.

$$f_\Delta = f_{Tx} - f_{Rx} \tag{3.15}$$

There are many reasons that might create a Carrier Frequency Offset (CFO) in OFDM systems because of either inconformity of frequencies between the oscillators of the transceivers or because of the Doppler spread [56]. As result of the CFO, the rotation of demodulated symbols in the constellation or ISI [53] can be noticed. Frequency synchronization should be performed very precisely at the receiver for the purpose of preventing losing orthogonality between the samples while frequency offset measurement in time domain could be done by applying maximum likelihood estimator. For this reason it is possible to use short training sequences with the duration of $0.8\mu s$ each. Let us presume that x_n is our transmitted signal, then passband signal y_n could be modeled from the complex baseband one as

$$y_n = x_n e^{j2\pi f_{Tx} n T_s}, \tag{3.16}$$

Here f_{Tx} is carrier frequency of the transmitter. As mentioned before, upon receiving signal, it should be downconverted to baseband signal r_n with a carrier frequency f_{Rx} that can be seen from Equation 3.17. Moreover, f_Δ refers to frequency offset.

$$r_n = s_n e^{j2\pi f_\Delta n T_s} \tag{3.17}$$

Frequency offset that can be gained from Equation 3.18, is calculated by the same delay and correlate method.

$$\begin{aligned} y_{\hat{\tau}} &= \sum_{n=0}^{L-1} r_n r_{n+D}^* = \sum_{n=0}^{L-1} s_n s_{n+D}^* e^{j2\pi f_\Delta n T_s} e^{-j2\pi f_\Delta (n+D) T_s} \\ &= e^{-j2\pi f_\Delta D T_s} \sum_{n=0}^{L-1} \mid sn \mid^2 \end{aligned} \tag{3.18}$$

Where D is 32 ($0.8\mu s \times 40MHz(f_s)$) according to IEEE 802.11n standard. FOE can be expressed based on Equation 3.19 when multiplication between the received signal and the complex conjugation of its delayed version is done.

$$\hat{f_\Delta} = -\frac{1}{2\pi D T_s} \angle y_{\hat{\tau}}, \tag{3.19}$$

Consider T_s as the sampling period and $\angle$ as the angle of $y_{\hat{\tau}}$, that is a correlation output in the last equation. Based on Equation 3.20, frequency offset correlation is gained by using Frequency Offset Estimation (FOE) and multiplied by the received signal. Here ${r_n}'$ as the corrected signal, n stands for the sample index and N is the number of samples in a symbol.

$${r_n}' = r_n \times \ e^{-j2\pi f_\Delta \frac{n}{N}} \tag{3.20}$$

3.2.3 FFT

The most time consuming and computationally intensive block is FFT. The 128-point FFT has to be implemented within 4 μs [12] according to IEEE 802.11n standard. The DFT will be obtained as the following Equation 3.21 [57].

$$X_{[k]} = \sum_{n=0}^{N-1} x_{[n]} W_N^{nk} \tag{3.21}$$

Where $k = 0, 1, 2, ..., N-1$ and $e^{-j2\pi \frac{nk}{N}}$ refers to the twiddle factor for W_N^{nk}. DFT complexity is equal to $O(N^2)$ and for FFT is $O(\frac{N}{2} Log_r N)$. The FFT block is expressed based on Equation 3.22 for radix-2.

$$X_{[k]} = W_N^k \sum_{m=0}^{\frac{N}{2}-1} x_{[2m+1]} W_N^{2mk} + \sum_{m=0}^{\frac{N}{2}-1} x_{[2m]} W_N^{2mk} \tag{3.22}$$

Also, Equation 3.23 is used for radix-4.

$$\begin{aligned} X_{[k]} = & \sum_{n=0}^{\frac{N}{4}-1} x_{[n]} W_N^{nk} + W_N^{\frac{Nk}{4}} \sum_{n=0}^{\frac{N}{4}-1} x_{[n+\frac{N}{4}]} W_N^{nk} \\ & + W_N^{\frac{Nk}{2}} \sum_{n=0}^{\frac{N}{4}-1} x_{[n+\frac{N}{2}]} W_N^{nk} + W_N^{\frac{3Nk}{4}} \sum_{n=0}^{\frac{N}{4}-1} x_{[n+\frac{3N}{4}]} W_N^{nk} \end{aligned} \tag{3.23}$$

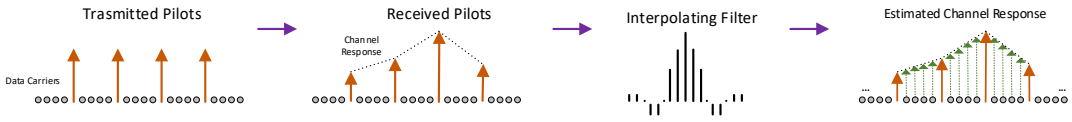

Figure 3.11 *Linear interpolation algorithm to perform the channel estimation [33]*

.

3.2.4 Channel Estimation

Channel impairments may need correction because the transmitted symbols will be affected by various impairments while passing through the wireless channel. The frequency spectrum of received signal should be determined [42] when data symbols are recovered after FFT which is found by a channel estimation block. Received and demultiplexed OFDM block can be expressed according to the following Equation 3.24.

$$Y_n = X_n H_n + N_n \tag{3.24}$$

Where H_n refers to the channel response, N_n is the additive noise and n stands for the number of subcarrier. There are two ways of performing channel correction [40], one way is to estimate channel attempts to correct H_n, and the other way is to equalize the channel attempts in order to correct Y_n according to X_n. There are many different methods to compute channel estimation such as Pilot-assisted linear interpolation and Least Square (LS) [61]. There are some training symbols in WLAN OFDM such as pilots which are mostly known for the receiver [59]. Six specific values are adjusted as pilots between data subcarriers in the transmitter, as mentioned in IEEE 802.11n specifications. At first, a diagonal matrix in the receiver side, O, is formed from transmitted pilots to accomplish the channel estimation. The matrix O expressed as

$$O = \begin{bmatrix} O_{1,1} & 0 & 0 & \cdots & 0 \\ 0 & O_{2,2} & 0 & \cdots & 0 \\ 0 & 0 & O_{3,3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & O_{j,j} \end{bmatrix} \tag{3.25}$$

Subsequently, the Channel Impulse Response (CIR) will be gained by:

$$\tilde{H}_j = O^{-1} P_{Rx}, \tag{3.26}$$

Here P_{Rx} refers to the received pilot, which may be noisy, $\tilde{H}_j$ is representing the received pilot for channel impulse response (CIR) and j stands for the number of pilots. One of the methods for estimating the channel is using the linear interpolation algorithm, which can be seen in Figure 3.11 [60]. Furthermore, the linear interpolation algorithm is the approximate model of value in any position between two samples and due to pilot overhead on the receiver side, this method is used to solve this problem. The two sequential known pilot subcarriers in the linear interpolation are used to specify the channel response for data subcarriers. Then, the intermediate estimates will be assessed through the linear sum of known elements on both sides. The channel estimation at the data sub-carriers k will be expressed as follows [61]:

$$mL < k < (m+1)L \tag{3.27}$$

Here mL and $(m+1)L$ are two points. Also, using a linear interpolation method in [61] can be expressed as:

$$\begin{aligned} H(n) &= H(mL+1), 0 < l < L \\ &= \left(\Big(H_j(m+1) - H_j(m) \Big) \times \frac{m}{L} \right) + H_j(m) \end{aligned} \tag{3.28}$$

Since linear interpolation is one of the easiest methods and the samples connect to each other with a straight line, then Equation 3.29 can be expressed based on the (`interp` function in the Matlab) [51] by extending Equation 3.28, where $\hat{H}_n$ refers to the channel frequency response of all subcarriers.

$$\hat{H}_n = \sum_{a=1}^{N_p-1} \sum_{b=1}^{P_l} \left(\Big(\tilde{H}_j(a+1) - \tilde{H}_j(a) \Big) \times \frac{b-1}{P_l} \right) + \tilde{H}_j(a) \tag{3.29}$$

N_p is the number of pilots and P_l refers to the data length between two successive pilots. In order to rectify OFDM symbols carrying noisy data by channel equalization, the channel frequency response must be estimated. Then the obtained Y_n symbols are similar to the X_n data. This procedure can be implemented as a result of the received signal divided by its channel frequency response, as shown in Equation 3.30.

$$\hat{Y_n} = \frac{Y_n}{\hat{H}_n} \tag{3.30}$$

3.2.5 Symbols Demapping

The last part of the OFDM receiver is demapping of the symbols. After performing all the synchronization and demodulation operations, demapping step will be run. Then, the true value of received data bits should be determined. The main purpose of demapping symbols is to transform received data symbols to data bits without loss of accuracy. Moreover, 64-QAM modulation was used based on IEEE 802.11n specification for this thesis work. Decisions about received data bits should be taken according to the system modulation. Also, decisions are divided into two parts, soft decisions and hard decisions [42].

- **Hard Decision:** If the number of transferred data bits is identical to the number of received data bits, a hard decision demodulator will be used and if the received data bits are noisy, a Gaussian cloud in the constellation points will be made by them. The difficulty is to decide on transferred data symbols related to the received data bits in this part. Maximum- likelihood decision says that allocation of bits will be carried out by hard decision if the constellation points and received bits are close to each other.
- **Soft Decision:** Soft Decision implies to use information bits about forwarded symbols which will give acceptable results in performance related to execution complexity [62].

When received data symbols are demapped to data bits, the quality of OFDM systems is measured in terms of bit error rate (BER). A part of bits that has errors over the total transmitted bits is called BER and it is varying as SNR changes. It reduces while SNR increases [64]. Moreover, BER is related on modulation type for the equal SNR.

4. PLATFORM ARCHITECTURES

The HARP platform is an experimental platform which permits to integrate at most nine NOC nodes, with the node in the center combined with a RISC core named COFFEE [65]. Other than the fact that a RISC core it is beneficial for general-purpose processing, it is necessary to be programmable and having constant supervision of the platform. Moreover, AVATAR accelerator can be combined with other nodes to speed up the tasks which are intense computationally.

4.1 Coarse-Grained Reconfigurable Arrays

As illustrated in Figure 4.1, CREMA and AVATAR accelerators have similar architectural attributes, and the only difference between them is their sizes that are modified according to the applications which are proposed. CREMA consists of R rows $\times$ 8 columns of PEs whereas AVATAR that is the developed version of CREMA has R rows $\times$ 16 columns of PEs. Also, CREMA contains 32-bit local memories of size 16 $\times$ 256 while the size of each local memory is 32 $\times$ 512 for AVATAR. I/O buffers insert the data among local memories and the PEs. In CREMA, I/O buffers contains sixteen 16 $\times$ 1 multiplexers and sixteen 32-bit registers, and the size is twice for AVATAR. The data that is processed into the local memories during an operation is loaded and stored on the PE array sequentially by applying Direct Memory Access (DMA) device [66].

AVATAR is a highly parallel template-based CGRA [66]. The computationally intensive cores are run by the accelerators generated by AVATAR, while the general-purpose processing is carried out by COFFEE that can be programmed in C. The DMA is interconnected with COFFEE, I/O peripherals and system memory through a matrix of switched interconnection. Each PE operates on two 32-bit operands and performs integer and floating-point computations. Furthermore, PE core components might be divided to two major parts: firstly, the Functional Units (FU), secondly, the control blocks of configuration. A PE includes a LookUp Table (LUT), addition, subtraction, shifter, multiplication, instant register and floating point logic that is selectable at design time based on the processing needs of the application. Figure

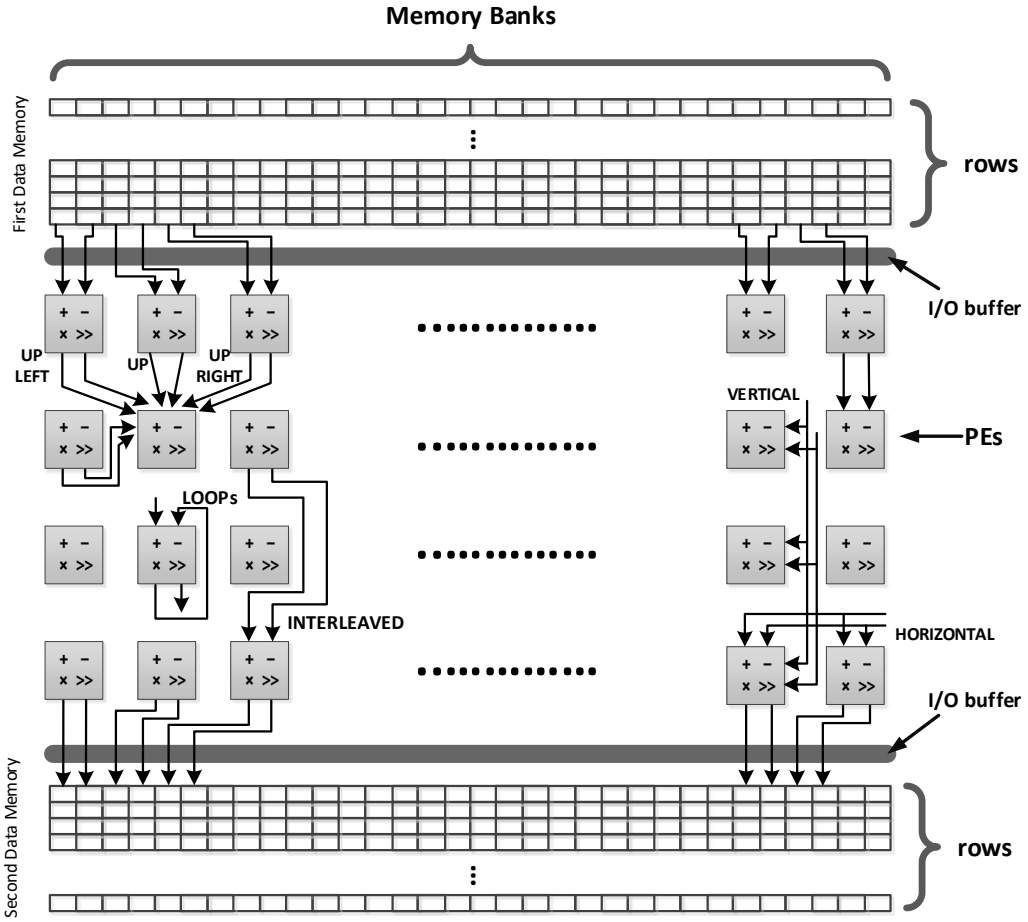

Figure 4.1 *The scalable template-based CGRA architecture.*

4.1 shows how each PE in a CGRA has point to point connection with neighboring PEs.

4.1.1 CGRA Execution Flow

Placement and routing for any particular application must be done by respecting its algebraic expressions. The context describes the interconnection between the PEs and the operation to be performed by each PE at any clock cycle. Configuration words are saved at system start-up and all PEs have their own memory. Also, configuration words apply to PE arrays with a pipelined structure [67] by DMA device and each of them shows an operation and address field. By using C language, it is possible to program the control flow of AVATAR accelerators, COFFEE RISC core performs the control operation and writes control words to the control registers of the CGRA accelerators. Depending on the configuration data in the CGRA, the context changes. Therefore, whenever replacement of existing configuration stream is required, reconfiguration will be performed. Here is a list of execution flow [33]:

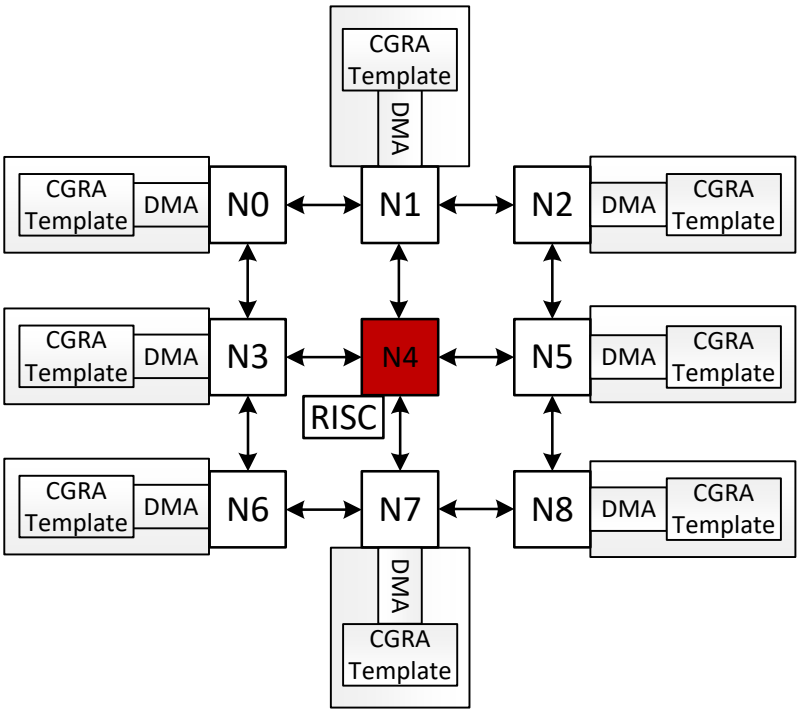

Figure 4.2 *Heterogeneous Accelerator-Rich Platform (HARP) [33].*

1. The DMA device at the system start-up time will facilitate the configuration data loading in the CGRA.
2. The data to be processed is loaded into the local memories of the CGRA.
3. Configuring the functionality of the PEs and interconnection among them by enabling each context.
4. Processing the data over PE array
5. As needed, CGRA will be reconfigured by changing the context.
6. Processing a new set of data from step 3

In order to complete the execution an algorithm will repeat these phases, and will transform the result to the local memory of another CGRA (RISC processor).

4.2 Heterogeneous Accelerator-Rich Platform

This thesis work employs HARP, depicted in Figure 4.2. The HARP is built with nine nodes over a NoC in a 3×3 mesh topology. The HARP platform is written in parametric VHDL. As can be seen from Figure 4.2, the COFFEE RISC core is combined with the node in the center and the rest include a template-based CGRA,

data memory and DMA device. All of the nodes can have a template-based CGRA accelerator except the central node which is integrated with COFFEE RISC core. By utilization of NoC the CGRAs and RISC core can communicate with each other. HARP comprises multiple CGRAs of particular dimensions in order (rows ×columns) of PEs. These CGRAs together form a test-case which can be used to examine the overall design. COFFEE RISC provides supervision in terms of control and communication. CGRA of a particular size for any of the existing nodes can be integrated since HARP is a template-based architecture. Alternatively, the node can be considered as a data routing resource.

4.2.1 Internal Structure of NoC

Figure 4.3 illustrates how the master and slave nodes are utilized in details. It can be seen that NoC nodes are connected to one master and two slaves where the mentioned master node is in charge of communication within the node as well as publishing the data on the network, hence it is combined with RISC core. On the other hand, the master node of other nodes are linked to the master of DMA device and the slaves utilize local memory and slave node of DMA device. The supervisor node consists of a RISC processor which is in charge of data transmission through its own and the other slave nodes' data memory. RISC cores provide the ability of synchronous data transmission between each two nodes by reserving a shared space in their memory for setting and resetting track of read and write flags which is accessible by the other nodes.

While the system is starting up, the data transfer is begun by the supervisor node (N4) with sending the configuration stream. Also, the data will be processed between its own and other data memories. The packet has two parts: routing information in the header and data and configuration words in the rest. At the first, the packet will be received by the initiator and then will be forwarded to the request switch of the destination node. As the node arbiter resides between the request and response switches to set up connections through various modules, the initiator module notifies a node. The targeted slave device then gets selected by the request switch based on the address field of the routed packet. Through the target module the data can be written to NoC. However reading/writing data from/to the instruction or data memory are requirements for given RISC core so the request switch has to interact with a node's master. As soon as the transport route is determined, DMA devices will load the processed data and configuration stream into the local memory banks of the template-based CGRA in the slave nodes. The RISC core can carry out the same operations for other nodes. Also, shared memory space of which size is

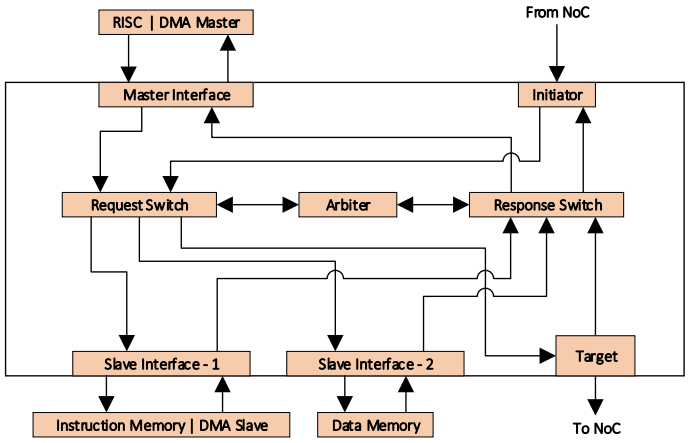

Figure 4.3 *A view of master and slave node of HARP [78]*

related to number of slave nodes must make synchronization among slave nodes and supervisor nodes. Writing '1' in the shared memory placed at the target node by the RISC core indicates the way of establishing synchronization is correct. Then the RISC core transmits the packet to the target node and asks to start the data transfer from DMA slave. In the next phase, the DMA master is responsible for fetching the stream of node's data memory and diffusing it to the configuration memories. After finishing the data transfer, the DMA's master will transfer an acknowledgment and the RISC core writes '0' over the NoC which results to reset the shared memory location related to the supervisor node.

When the DMA devise accomplishes the internal data transfer of the template-based CGRAs totally and as soon as the configuration stream and the data are filled in the local memories of slave nodes, the RISC core will forward the control words to the CGRAs. The slave nodes include the CGRAs that can work in parallel and independently of each other. Hence, the dependency of some data in the flow of the program may occur that needs the exchange of data between the nodes of the CGRA. To make more processing, the results stored in one of the local CGRA memories can be carried to other slave nodes. However, the same synchronization process should be accomplished via the RISC core because it specifies the shared memory location allocated to the receiver node, and then the DMA device targets the sender node.

5. DESIGN AND IMPLEMENTATION OF IEEE 802.11N ON TEMPLATE-BASED CGRA

In previous chapters the concepts and basic structure of OFDM systems and the platform architecture used in this work were explained. This chapter describes the execution of OFDM blocks in detail and focuses on the design of CGRAs on HARP platform. In others words, the design of an application-specific accelerator using AVATAR for implementing the IEEE 802.11n specification is elaborated. In this process a baseband receiver executes the algorithms of digital signal processing, in order to utilize the received data bits with high accuracy. Moreover, MATLAB is used to test the functionality of baseband receiver and all of the transceiver algorithms implemented. During the first step, a MATLAB script generates random data symbols by employing a specific constellation based on IEEE 802.11n specifications.

Subsequently, the accelerator was designed and then implemented for each block such as TS, FOE, FFT and CE. Each output was compared to the corresponding MATLAB results. Moreover, the script generates OFDM data symbols based on 64-QAM modulation, and the channel with AWGN is modeled assuming that CFO is 40 kHz and the SNR value is equal to 20 dB. In the following section, the process of designing and implementing the application-specific accelerator for each receiver block is explained. COFFEE is used for calculating the number of clock cycles and execution time for each one of them. Also, ModelSim software [79] is utilized for simulation purposes and for testing the functionality of each accelerator. Ultimately, the designed accelerators are synthesized onto Altera Stratix FPGAs.

5.1 Time Synchronization

After the analog to digital conversion, time synchronization is the first block of OFDM receiver. As mentioned earlier, there are two known methods to achieve time synchronization: using special symbols or Cyclic Prefix (CP). In this thesis, cyclic prefix method is employed [54]. A correlation algorithm is performed between the received signal and its delayed version. In accordance to the IEEE 802.11n specification, the delay length z^{-D} is equal to the length of CP, $L = 32$.

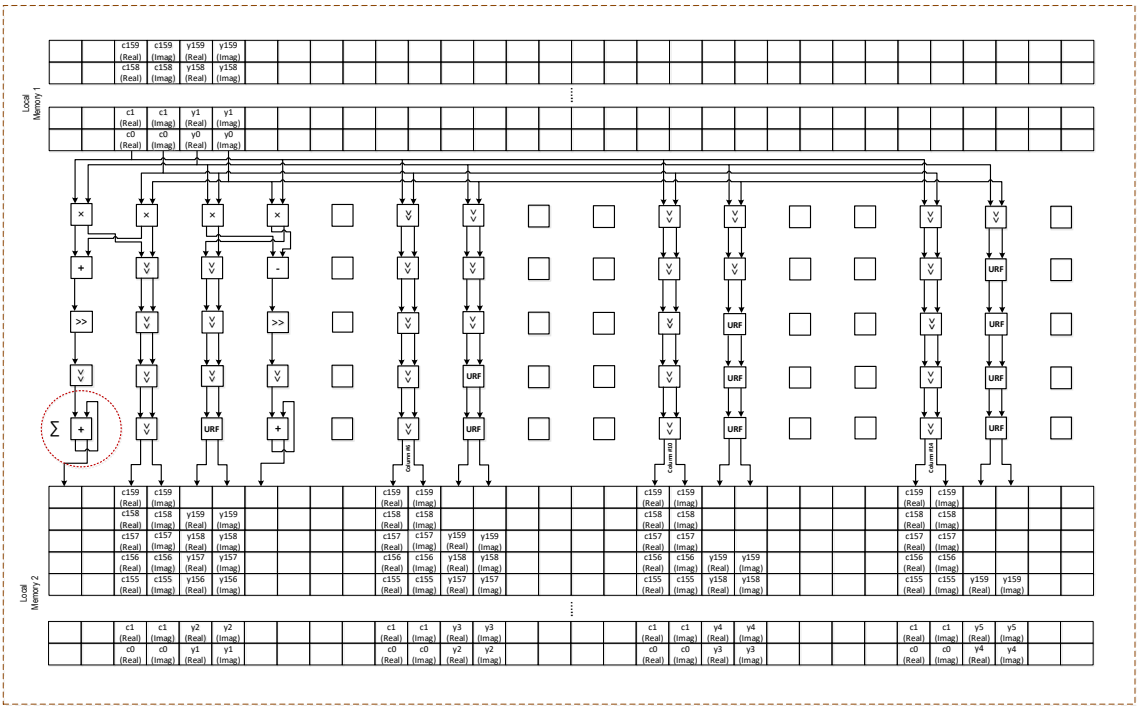

Figure 5.1 *Second context for the calculation of the correlations*

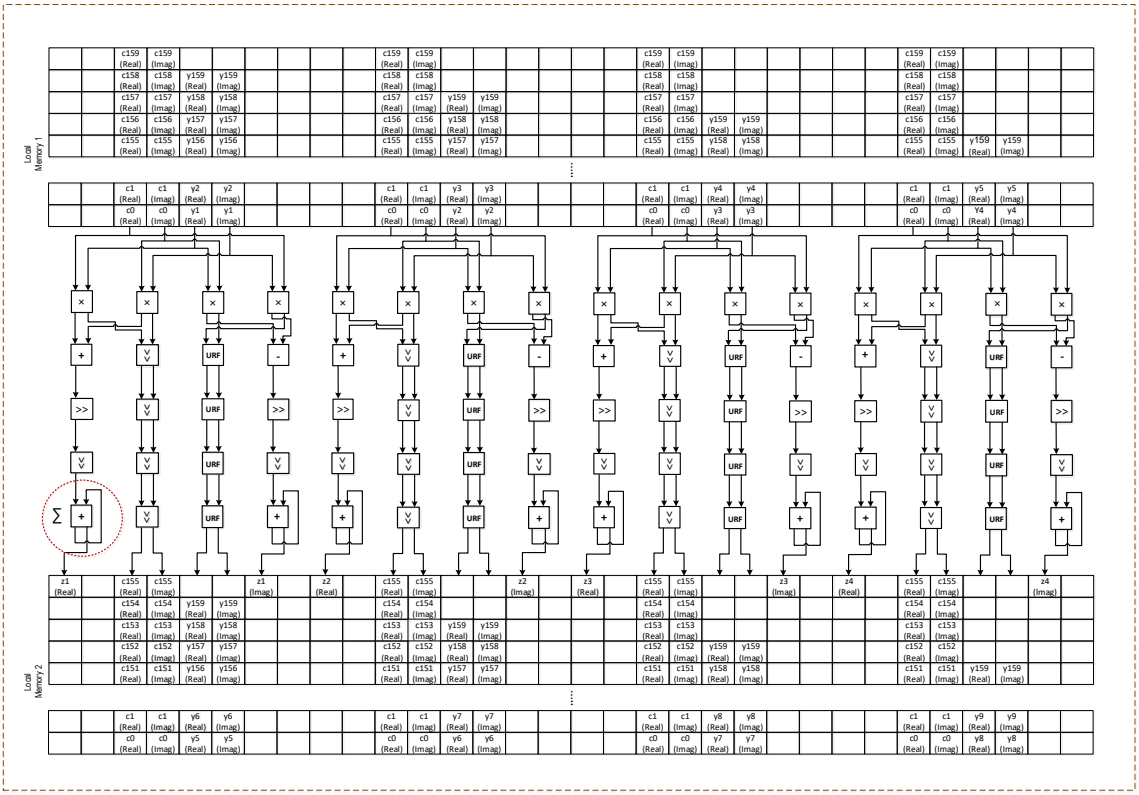

Figure 5.2 *Third context for the calculation of the correlations*

Outputs c_n and z_n of the correlation algorithm are expressed by Equations 5.1 and 5.2 respectively. Symbol of $*$ stands for the complex conjugate.

$$c_n = y_n y^*_{n-D} \tag{5.1}$$

$$z_n = \sum_{i=0}^{L-1} c_{i+n} \tag{5.2}$$

Equations 5.1 and 5.2 should be mapped over the PE array of AVATAR. For this specific design, AVATAR is further scaled up to 5×16 PE array. Also, Equation 5.1 can be simplified to Equation 5.3, to perform a more effective way for placement and routing. R stands for Real and I for Imaginary part of the received signal.

$$c_n = \underbrace{((y_{n(R)} \times y_{n-D(R)}) + (y_{n(I)} \times y_{n-D(I)}))}_{Real} + \underbrace{((y_{n(R)} \times y_{n-D(I)}) - (y_{n(I)} \times y_{n-D(R)}))}_{Imaginary} \tag{5.3}$$

In Figures 5.1 and 5.2 the mapping of a 160-point correlation algorithm is shown. The two contexts must be used consecutively. The first context (is not illustrated) loads immediate values to the PEs to operate shift after any multiplication. With regard to interconnections between PEs and the processes that the PEs should perform, they are completely identical. The sole dissimilarity, though, is their I/O buffers. First, the received data symbols must be loaded into the first local memory of AVATAR. As can be observed in Figure 5.1, two different tasks are performed in this context. The multiplication among the received data symbols and the complex conjugation of its delayed version is related to the first task based on Equation 5.3. The data demonstrated by indexes from 0 to 159 belong to 160-point correlation.

To perform time synchronization for 160 data symbols requires 160 correlations. Data distribution in other columns of local memory is the second task in this context so as to maximize the parallel usage of resources. As discussed earlier, this context can only be used to distribute data and implement the first correlation. For executing four correlations simultaneously, the results stored in the second local memory are applied by altering the data flow direction to the first local memory.

Figure 5.2 shows in last row of PEs in the third context, a sum-of-products of the results of complex multiplications is performed based on Equation 5.2. The delayed version undergoes a shift and the procedure is repeated. Unregistered-Feed Through (URF) operation is used to solve this issue. According to Figure 5.2 in the third context, four URFs shift the delayed version of data symbols four units during each run. Then, in the next step and when all the correlations are completed (160

correlations), the maximum value is looked for by RISC processor (N3). Moreover, after the execution of C code the largest value is obtained. This variable corresponds to the time offset index, which is equal to the first FFT window by using Equation 5.4. Program 5.1 that looks for the largest value for each data symbol is performed by COFFEE RISC processor.

```
1  int  z, max_val, position = 1;
2  max_val = output[160];
3
4  for (z = 1; z < 160; z++)
5  {
6    if (output[z] > max_val)
7    {
8        max_val  = output[z]; position = z+1;
9        }
10 }
```

Program 5.1 *C code for the seek of largest value*

The Square Modulus (SM) is used when the correlation results are complex. SM uses Equation 5.4 to calculate the magnitude of complex numbers.

$$\hat{\tau_s} = \underset{n}{\operatorname{argmax}} \mid z_n \mid = \underset{n}{\operatorname{argmax}} \mid z_{n(R)} \times z_{n(R)} + z_{n(I)} \times z_{n(I)} \mid \tag{5.4}$$

Where $\hat{\tau_s}$ identifies the maximum value among the 160 correlation results of (z_n). Besides, R and I illustrate the Real and Imaginary parts in this Equation. At the end, after finding the index of the time offset, the data symbol should be transmitted to the next block which is the FOE for further processing discussed in detail in Section 5.2.

5.2 Frequency Offset Estimation

As explained previously, Carrier Frequency Offset (CFO) in OFDM systems is the consequence of incompatibility between the transmitter and receiver oscillator. By adding training symbols on the transmitter side, the CFO can be estimated. Based on Equation 3.18, for this intention, the delay and correlation algorithm is used with a delay value equal to 32. That is to say, to obtain the phase difference between the received training symbols, a multiplication by the complex conjugation of their delayed version must be done. Therefore, there needs to be 160 complex multiplications, equal to the length of the short training sequence.

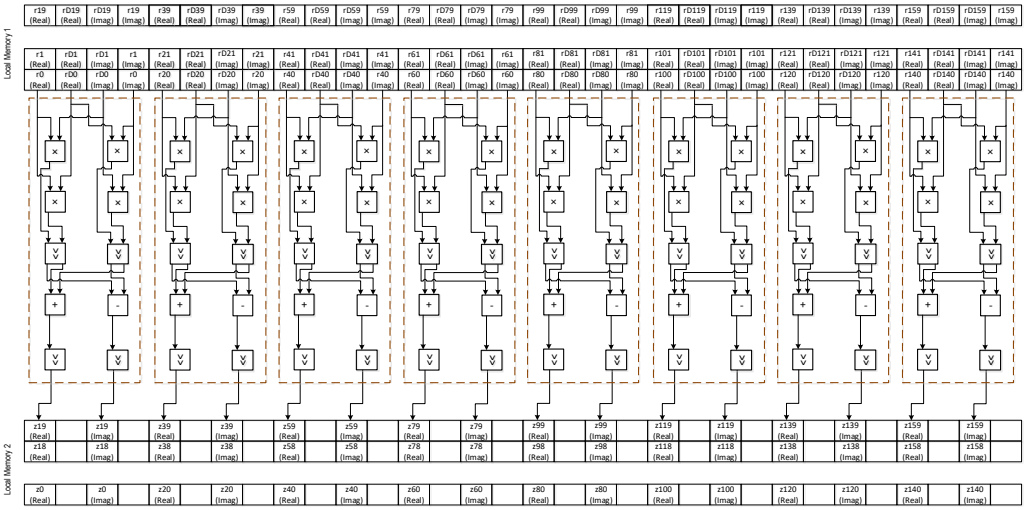

Figure 5.3 *The context for the multiplication between a signal and its complex conjugation*

The short training symbols used to estimate frequency offset estimation is consisting of a certain number of predefined data symbols, added in the transmitter side. The first context is shown in Figure 5.3, in which short training symbols in the first local memory along with its delayed version are loaded using DMA. Here, r and rD refer to short preamble and its delayed version respectively. This context is ideal for performing multiplication between complex numbers. Accordingly, after multiplication between the received signal and its delayed version, the phase difference value should be calculated in the next step. In this way, the result of the first context of second local memory is transferred to the main memory for more processing.

`ATAN` function is one of the procedures for calculating the phase angle of a complex value. To find the phase angles of a complex number such as $x+iy$, x and y are real and imaginary parts, respectively. The following equation can be expressed as

$$w = atan(\frac{y}{x}), \tag{5.5}$$

Where w stands for the phase angle of a complex number. Firstly, the imaginary part should be divided by the real part, and then the phase angle is calculated using the ATAN operation. This can be accomplished in software by utilizing a CORDIC algorithm that is one of the well-known algorithms because of the simplicity of its hardware implementation ([68], [69] and [70]). The algorithms of complex algebraic equations and a large number of repetitions like CORDIC are not effectual to be mapped on CGRA in terms of execution time. In addition, the designed CGRA is simple as well as incapable of performing complex algebraic and trigonometric ope-

rations. Accordingly, they can be done in shorter running times by using processor software at the cost of more power and energy. The use of CORDIC algorithms is most beneficial when no predefined hardware multiplier exists as they only use addition, subtraction, bit-shift and lookup table [69]. Once the division is done according to Equation 5.5, the phase angle of a complex number should be computed by means of the result of the division from the prior section. As previously mentioned, there are no predefined functions on the COFFEE RISC processor, so ATAN function has to be done using another method like the Taylor series [71]. Finally, the phase angle can be calculated in processor software by expanding Taylor series for the arctangent ratio as Equation 5.6.

$$arctan\ x = \sum_{n=0}^{N} \frac{(-1)^n}{(2n+1)} x^{2n+1} \tag{5.6}$$

Where the value of N is dependent on the needed precision, which is equal to 4 in this particular case. When the phase angles are found from the received data symbols, carrier frequency offset should be estimated by means of Equation 3.19 previously explained. Afterward, using Equation 3.20, the data symbols must be corrected separately on the basis of the estimated frequency offset, in which the exponential function is needed. The Taylor series has to be employed based on the following equation.

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \tag{5.7}$$

The equation above works only for integer numbers, so for complex numbers, the Equation 5.7 should be modified as

$$e^z = e^x\big(cos(y) + isin(y)\big), \tag{5.8}$$

z consists of the real part x and the imaginary part y. cos and sin functions can be expanded by the Taylor series, which are shown in Equation 5.9 and 5.10, respectively.

$$cos\ y = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} y^{2n} \tag{5.9}$$

$$sin\ y = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} y^{2n+1} \tag{5.10}$$

Finally, the received signal should be multiplied by the correction factor which is calculated above. Once all of the steps above are carried out on the processor software

to perform the complex multiplication, the data can be transferred again to the local CGRA memory. This can be done by using almost the same context as depicted in Figure 5.3. Subsequently, after performing frequency offset estimation operation the local memories have the data symbols that can be directly demodulated. It has to be mentioned that before the demodulation of the data symbols, the cyclic prefix is removed.

5.3 Fast Fourier Transform

In this section, the Fast Fourier Transform (FFT) block will be explained in detail. The data symbols in this step must be converted from time domain to frequency domain after the received signal has been corrected in terms of the frequency offset; it's named demodulation. It can be executed by FFT as a particular class of Discrete Fourier Transform (DFT). When comparing the FFT block with other blocks on the receiver side, FFT block is one of the most time consuming. More details about the implementation of the radix-N algorithms can be found in [58] and [10]. The Discrete Fourier Transform (DFT) can be defined as

$$X[k] = \sum_{n=0}^{N-1} x[k] W_N^{nk} \tag{5.11}$$

Where $W_N^{nk} = exp(-j2\pi\frac{nk}{N})$ is a twiddle factor and n is any sample that has been processed among samples of N. DFT is obtained by FFT radix-2^m structures [26], where $m \in Z^+$ and its structural unit is a butterfly. Furthermore, with the increase of m, the arithmetic resources required by the butterfly will increase along with the complexity of the FFT structure, but the number of implementation steps for FFT processing will be significantly reduced. Based upon IEEE 802.11n specifications with frequency bandwidth 40.0 MHz, demodulation can be done by a 128-point FFT within $4\mu s$.

In addition, a 128-point FFT can not be processed by a radix-4 butterfly, while the radix-2 needs expensive seven stages to process it. However, if a mixed-radix is used, the 128-point FFT can be implemented in four stages. In the first stage, a radix-2 butterfly is used for processing and as a result, the structure of 128-point FFT is divided into two parts. Each half of the FFT will be 64-points. Also, these halves can be processed by a radix-4 butterfly in three steps.

Mapping four different contexts, a mixed-radix accelerator was designed utilizing AVATAR, two of which comprise radix-2 and radix-4 butterflies and the remaining

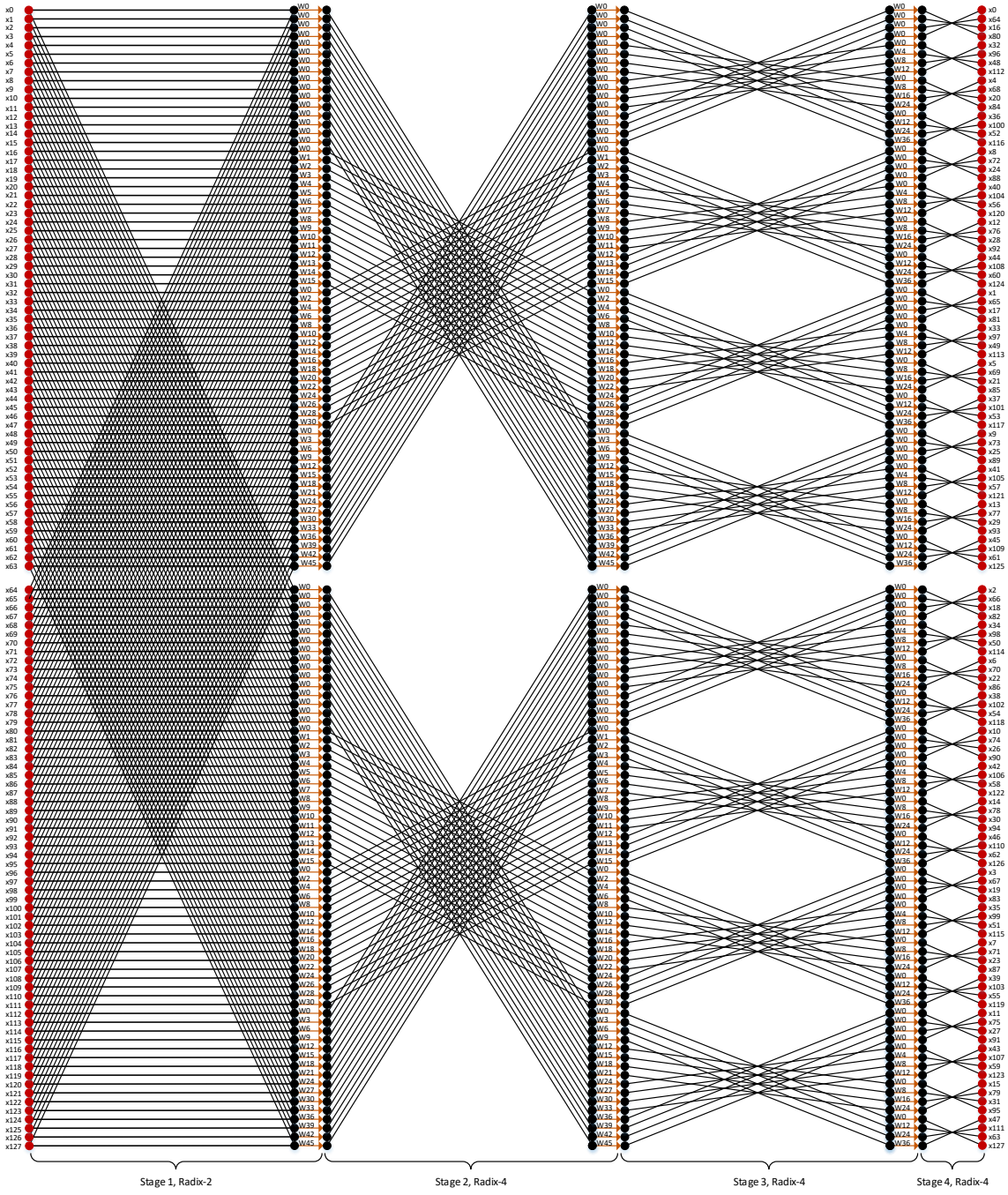

Figure 5.4 *Fast Fourier Transform [10]*

operate for the purpose of data reordering. FIeld programming and REconfiguration management Tool (Firetool) [77] is used in order to design and execute the mentioned contexts. The desirable efficiency of the accelerator processing can be achieved with the help of a mixed-radix scheme in variety of FFT lengths. Also, specific number of processing stages is proper for each FFT structure. For instance, 128-point FFT structure with radix-2 requires seven processing stages.

Table 5.1 *Different types and lengths of FFT and their complexity in number of stages and in number of operations per butterfly [10].*

FFT types	Length	Stages	Additions	Subtractions	Shifts	Multiplications
Radix-2	128	7	3	3	4	4
Radix-4	1024	5	15	15	12	12
Radix-2,4	4×128	4	18	18	16	16

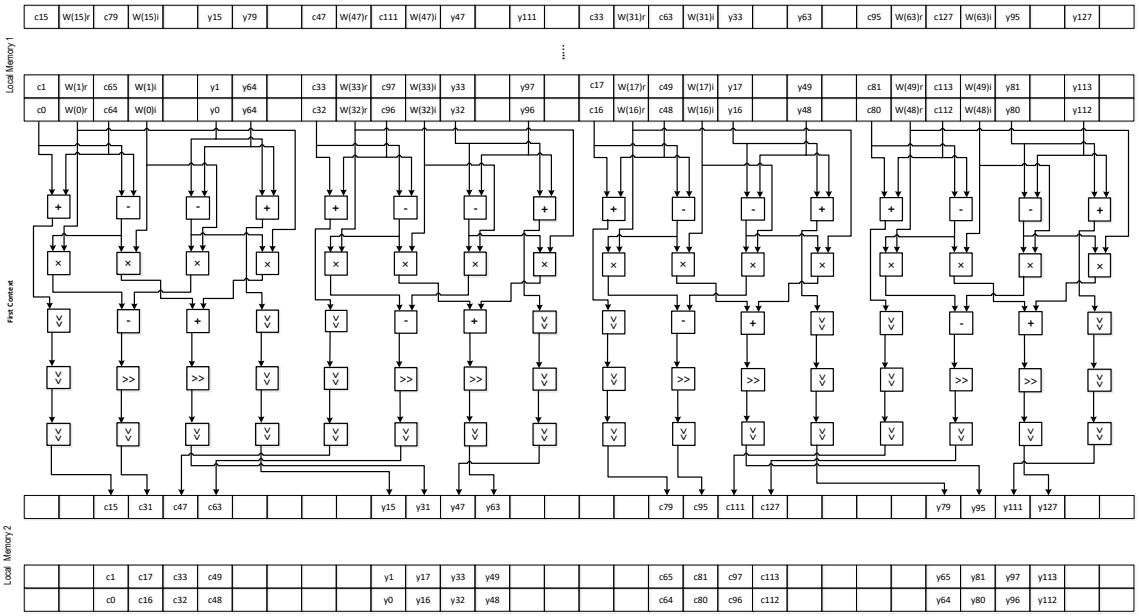

Figure 5.5 *The first context includes four radix-2 butterflies.*

As shown in Figure 5.4, this context is utilized for processing of the first stage of the 128-point FFT structure. The twiddle factors and sample points are loaded in such a way so the task of processing is distributed among four Radix-2 butterflies. As can be seen from Table 5.1, radix-2 FFT butterfly needs four multiplication operations as well as four shift operations and since the 12-bit numbers are processed into the 32-bit register elements, overflow never happens even if an addition or a subtraction operation is performed after multiplication.

Figure 5.5 shows that after an addition or subtraction, a shift operation can be executed. In the first context, there is a fixed value of 12 bits. They are loaded in the immediate registers of the PE on the vertical connections. This is performed with the shift operation. In the second context, a radix-4 butterfly is used. In this context, the radix-4 butterfly can processes the second, third and fourth stages based on 64-point FFT. When using the radix-4 FFT butterfly, there are three twiddle factors and four complex samples as inputs. In the design of the second context by radix-4, it is required to load extra data for twiddle factors. The twiddle factors are provided

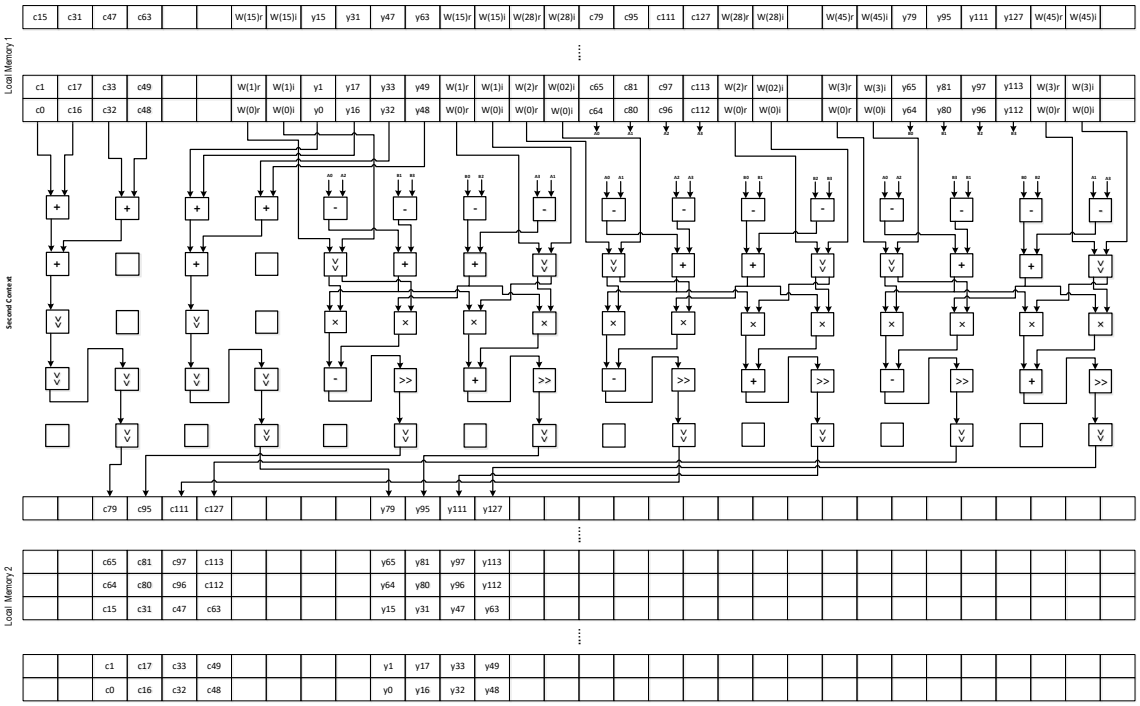

Figure 5.6 *A radix-4 butterfly for the second context*

Table 5.2 *Clock cycles (cc) based on the type of FFT accelerator and length [10].*

FFT types	Length	IEEE standard	CGRA template	Clock cycles	Execution time
Mixed-Radix (2,4)	4×128	802.11n	AVATAR	759 cc	$3.9\mu s$

in the array by the vertical connections. According to the radix-2 in the first context, the processed samples should be distributed to the four radix-2 butterflies so that the results obtained from the first context based on radix-2 FFT structure is split into two equal parts. Also, in the beginning of the second stage of radix-4 processing, each stage needs 32 CC and a latency of 5 CC, which is shown in Figure 5.6. Based on Table 5.2, AVATAR accelerator of mixed-radix can satisfy the timing constraint.

5.4 Channel Estimation

In this section, the structure and mechanism of the channel estimation for the OFDM systems will be expressed in detail. There are various techniques for estimating the channel for OFDM systems. The channel estimation provides an estimated channel impulse response to the equalizer. Moreover, by the channel estimation at the receiver, channel state information can be traced and obtained. As previously mentioned, the data symbols transmitted over the wireless channel may be distorted due to different disturbances in the channel before reaching the receiver antenna.

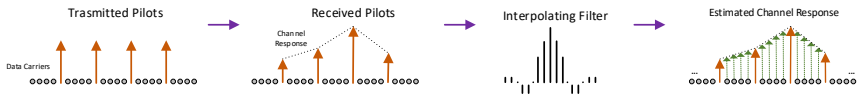

Figure 5.7 *Linear Interpolation algorithm based on pilot-assisted for Channel Estimation [33]*

There are two techniques for estimating the channel based on pilot arrangement. The first technique is called block-type channel estimation and the second technique is called comb-type channel estimation. The first method is best suited for slow fading channel, in which pilot tones will be inserted into all the subcarriers of OFDM symbols within a specific period. In other words, in time domain, symbols are transmitted periodically by Least Square (LS) and Least Mean Square (LMS) estimation channel. With this technique, initially the channel will be estimated, and subsequently the same estimates will be used throughout the block. In the second method, suited for fast fading channel, some subcarriers are reserved for pilot tones in each OFDM symbol. This technique can estimate the channel at pilot frequencies based on Least Square (LS), Minimum Mean Square Error (MMSE) or Least Mean Square (LMS) [60].

For performing the channel estimation, we can utilize one of the simplest interpolation methods, which is named Linear Interpolation algorithm. This method is not very accurate, but it is a quick and easy method for performing channel estimation as shown in Figure 5.7. In addition, the channel estimation can executed on pilots, which are known to the receiver. According to Equation 3.26, the channel impulse response has to be computed at the beginning of the channel estimation block. This equation is the complex multiplication between inverse of the transmitted pilots and the received pilots. Furthermore, on the basis of IEEE 802.11n specification with 40.0 MHz, the number of pilots is equal to $N_{pilot} = 6$. The pilots are inserted among the subcarriers in the transmitter as depicted in Figure 5.8.

The first context is appertaining to loading immediate values for shift operations. Figure 5.8 shows the four columns of PEs used in the second context of the channel estimation. For more processing, the received pilots (RP) and the inverse of transmitted pilots (ITP) are immediately loaded into the first local memory. In the second context of channel estimation, the complex multiplication between RP and ITP by Linear Interpolation can be mapped on AVATAR. Thereby, by only one context, the channel response can be computed and stored in the second local memory. After calculating the channel impulse response of pilots, it should be expanded for the remaining subcarriers by collaborating Linear Interpolation algorithm according to

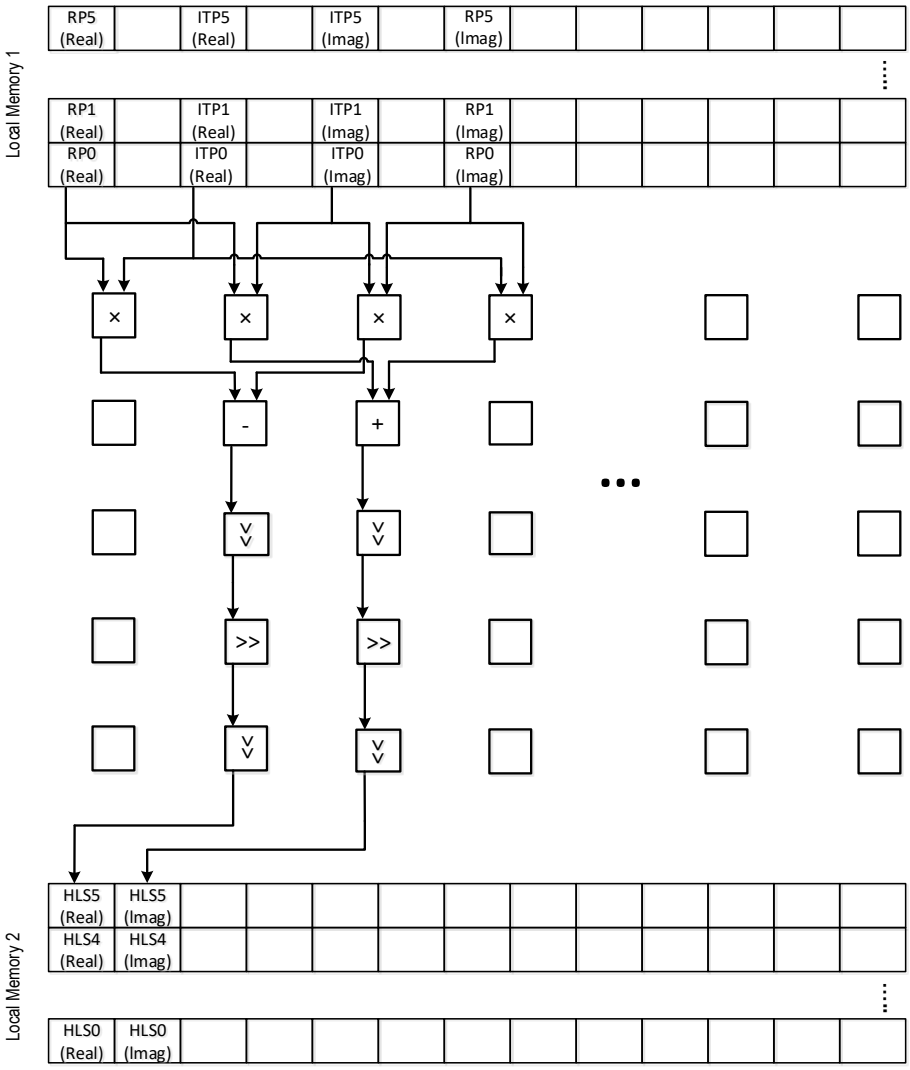

Figure 5.8 *Second context of the channel estimation*

Equation 3.29.

Figure 5.9 shows the first context for the linear interpolation. Initially, the real part is loaded into the first local memory coupled with the step size. When the linear interpolation algorithm is done for real part and imaginary part as shown in Figure 5.10, the channel frequency response is transmitted to the main memory for all of subcarriers because it is needed in the last step for implementation of the channel equalization. For the purpose of performing the channel equalization, the receiver needs to have the channel frequency response of all of subcarriers. When the channel response $\tilde{H}_k$ of the received pilot is calculated and then stored in the second local memory, eventually it's necessary to estimate the channel frequency response of the adjacent subcarriers using linear interpolation algorithm.

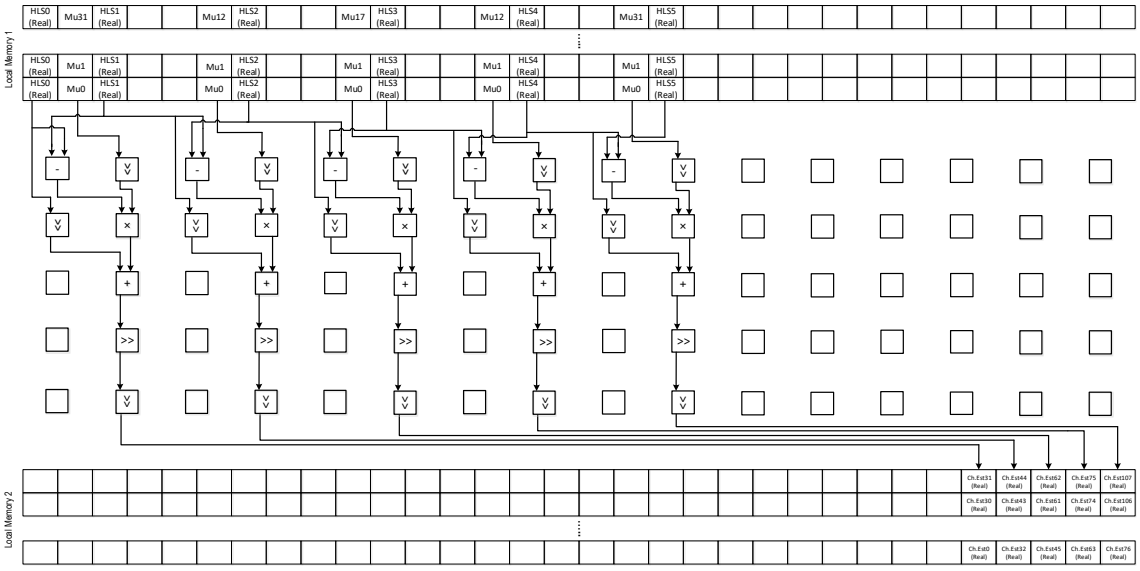

Figure 5.9 *First context of the Linear Interpolation*

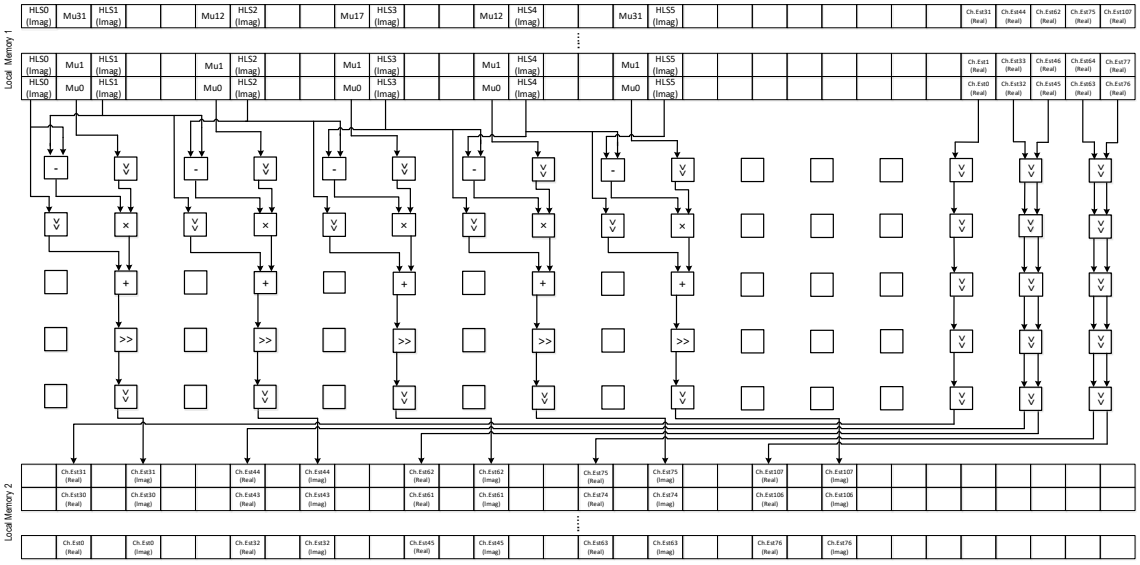

Figure 5.10 *Second context of the Linear Interpolation*

In addition, with respect to Equation 3.30, the demodulated data symbols have to be equalized after calculating the channel estimation. There is an important point in this step that should be taken into consideration and it is that in COFFEE RISC and AVATAR, division operation is not available. Another algorithm required to do a division operation is Newton-Raphson iteration algorithm [72]. Newton-Raphson method is an algorithm for seeking the root of an equation. For instance, if there is a given function of *f(x)*, Newton-Raphson method [72] can be used by the Equation 5.12 to acquire the first approximation of its root.

$$x_{i+1} = x_i + \frac{f(x_i)}{f'(x_i)}, \tag{5.12}$$

where i is the number of iterations such as *i = 0, 1, 2, ...*, and x_i will be the initial guess for the root. The function $f^{'}(x_i)$ is derivative of a function $f(x_i)$ and division operation based on this Equation 5.13 can also be modified to use other purposes.

$$x_{i+1} = x_i.(2 - Dx_i) \tag{5.13}$$

Assuming that finding $\frac{1}{D}$ is desired, we need to find the function *f(x)* whose value is zero at $x = \frac{1}{D}$. The resulting function is $f(x) = \frac{1}{x} - D$ based on Equation 5.13. where x_i is representing the initial guess and D stand for the denominator. Based on the noisy channel, the denominator is a complex number. Therefore, for more processing by Newton-Raphson method and in order to map on CGRA Equation 5.13 can be simplified to an integer number as

$$\frac{x + jy}{a + jb} \times \frac{a - jb}{a - jb} = \frac{(x + jy) \times (a - jb)}{a^2 + b^2} \tag{5.14}$$

where $x+jy$ stands for received data symbols from FFT block, $a+jb$ is representing estimated channel response and $a - jb$ signify complex conjugation of the channel response. In the first step, Newton-Raphson method should be mapped on AVATAR for calculating the value of $\frac{1}{a^2+b^2}$ in order to carry out the channel equalization. Here according to the Equation 5.13, D is equivalent to $(a^2 + b^2)$. a stands for the real part of the channel frequency response and b is representing the imaginary part of the channel frequency response. The mapping of Newton-Raphson method is shown in Figures 5.11 and 5.12.

The mapping employs eleven columns of the first context and ten columns of the second context. So as to compute the square values of the real and imaginary parts of the channel frequency response, in the first context, the first PEs' row act as multiplication. These values, then, are added to each other on the second row. Computed results, in the next step, must be multiplied by the initial guess Dx_n. Then, **2** is loaded into the local memory as well as results obtained from the previous context based on Equation 5.13. The local memories in CGRAs are line-readable in order to be faster and simpler. The values of **X** and **2** are loaded along with every column. The first row of PEs performs preprocessing of data and with the rest of the rows of PEs, it can execute the required shift operations, subtractions and multiplications. Finally, specific DMA operations transfer the result of division $(\frac{1}{a^2+b^2})$ back to the main memory.

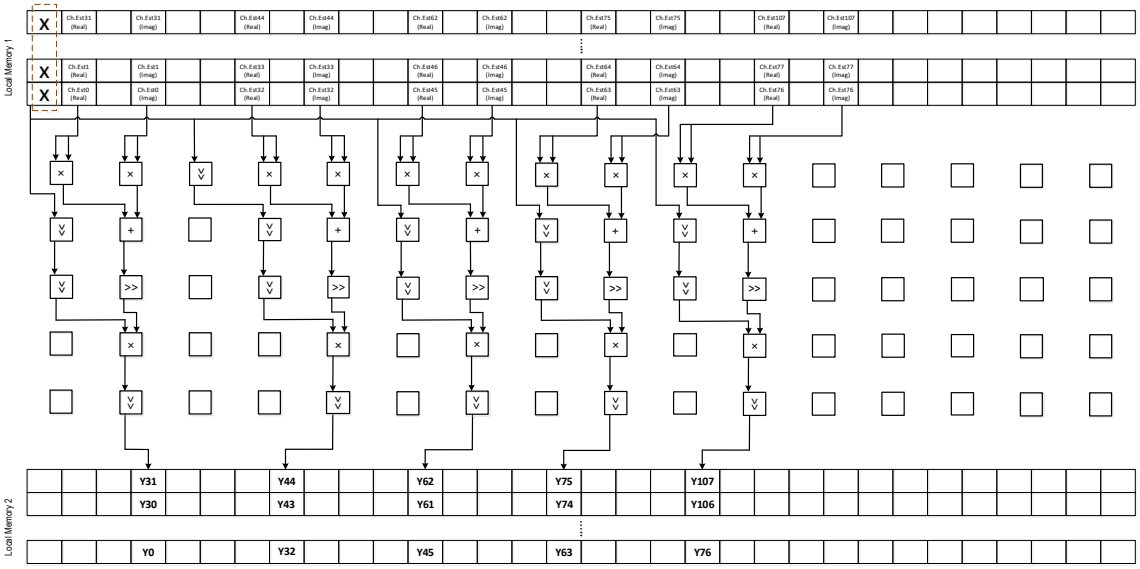

Figure 5.11 *First context of the Newton-Raphson method*

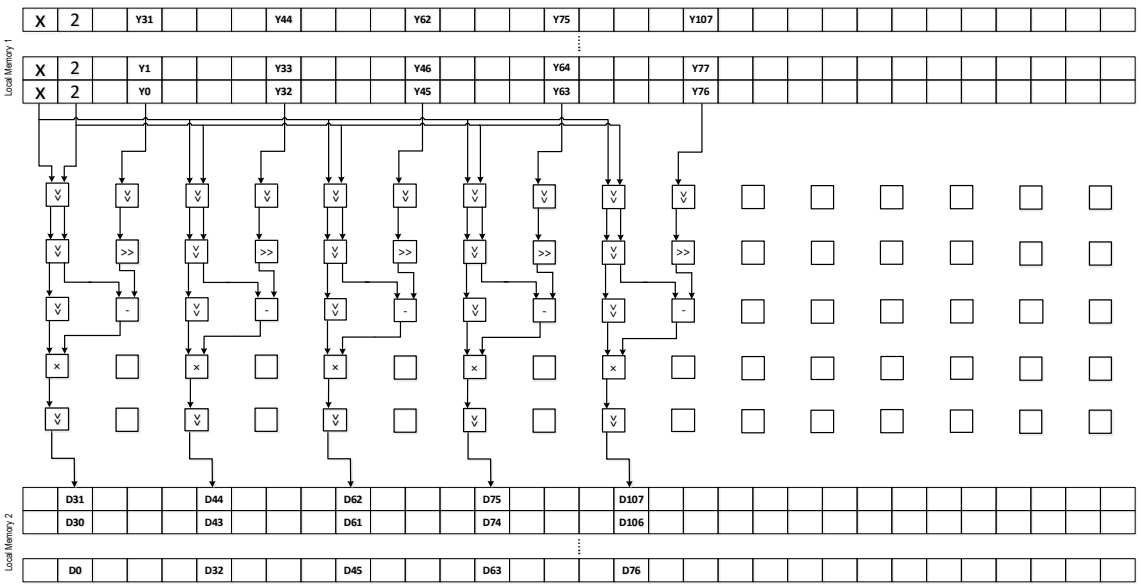

Figure 5.12 *Second context of the Newton-Raphson method*

In order to perform the channel equalization based on Equation 5.14, the numerator of this equation must be calculated. It is a complex multiplication between complex conjugation of estimated channel frequency response and the demodulated data symbols. As demonstrated in Figure 5.13 for executing complex multiplication in a context, data will be returned from the main memory to the first local memory of AVATAR. During the last context of a channel estimation block as shown in Figure 5.14, the achieved results from the Newton-Raphson algorithm will be multiplied by the results of complex multiplication. In this stage, shift and multiplica-

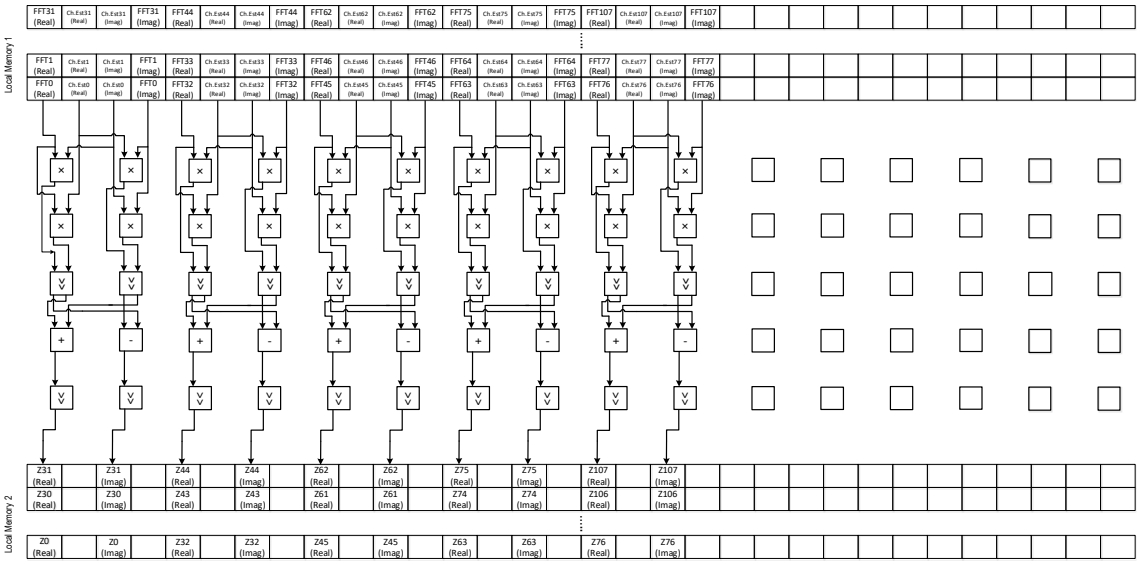

Figure 5.13 *Sixth context of the channel estimation*

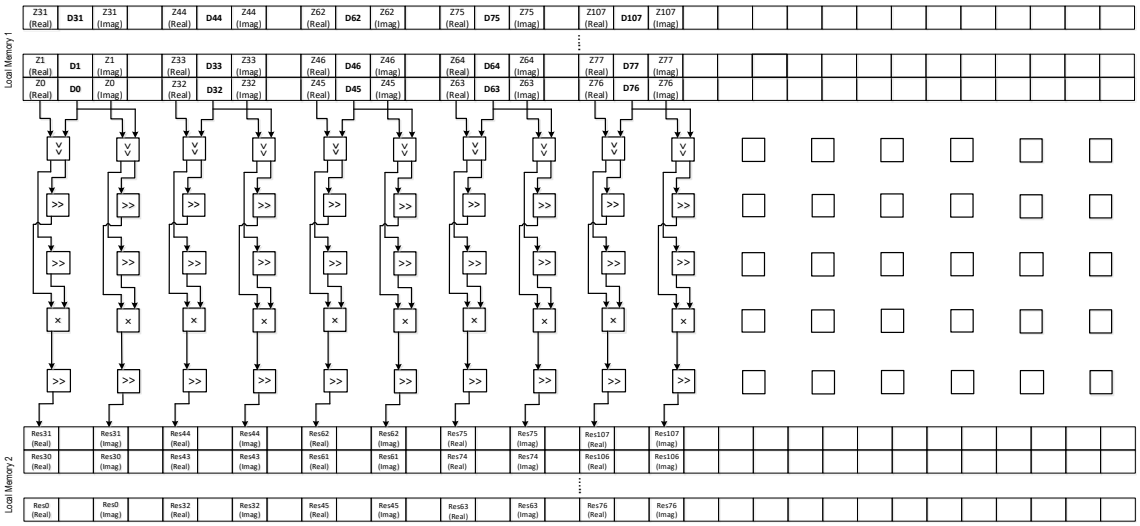

Figure 5.14 *Seventh context of the channel estimation*

tion operations will be used to generate the final product. The shift operation is also required after each multiplication to prevent the data overflow. Res_i is representing the equalized received data symbols.

5.5 Symbols Demapping

In this section, a decision on the received data symbols should be made with the help of the decision boundaries after performing the channel estimation block and equalization. For each received data symbol, the most likely transmitted data bits

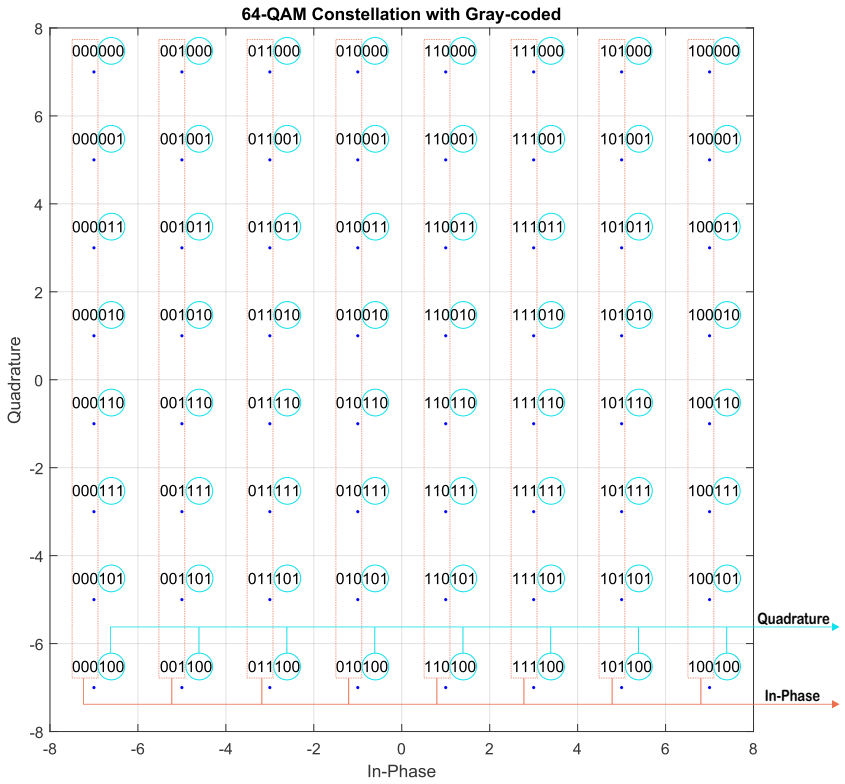

Figure 5.15 *Decision regions of 64-QAM Gray-coded constellation*

must be determined. As previously mentioned, for making the decisions about received data bits there are two ways, one is hard decision and another is soft decision. To execute the symbols demapping of 64-QAM modulation with Gray coded bit mapping, hard decision will be used.

The transmitted data symbols consist of two real baseband signals, which are independent. Therefore, the complex plane can be divided into regions of decision with each region composed of the set of points that are very close to a certain symbol. This is called Maximum likelihood detection. Generally, a Gray code can be expressed in which all adjacent constellation symbols differ by exactly one bit. Figure 5.15 shows how the complex plane is divided into the In-phase part (I) and the Quadrature part (Q), which are equivalent to real (R) and imaginary (I) parts of received data symbols. Accordingly, in 64-QAM constellation for each data symbol containing six data bits that might be used for symbols demapping purpose individually. By hard decision way, in the beginning, the three leftmost bits can be detected based on the `C` code that is written for symbols demmaping, as shown in Program 5.2.

Figure 5.15 shows there are eight areas for each three bits (rightmost and leftmost) after dividing the complex plane into Quadrature and In-phase regions. Meanwhile, the first three bits *(Bit1 Bit2 Bit3)* are repeated frequently for each region of I-axis constantly, whereas the last three bits *(Bit4 Bit5 Bit6)* are similar for each region of Q-axis. Table 5.3 shows how each constellation point is represented by a 6-bit symbol composed of three bits each from the In-Phase axis and the Quadrature axis respectively.

Example. Assume that in the receiver side, '$4.8 + 2.8i$' is received as a demodulated data symbol instead of '$5 + 3i$' because of the Additive White Gaussian Noise (AWGN) channel. In the first step, a decision boundary can be made for real and imaginary parts, respectively. There are eight states (≥ 6, < 6 & ≥ 4, < 4 & ≥ 2, < 2 & ≥ 0, < 0 & ≥ -2, < -2 & ≥ -4, < -4 & ≥ -6, ≤ -6), which might happen for each real and imaginary part. As '4.8' is greater than '4', specific bits can be determined: '101' is assigned to the first three data bits. Figure 5.15 shows that received data bits only vary between eight different values, which contains bits ('101000','101001','101011','101010','101110','101111','101101' and '101100') that are located within the same zone of I-axis. In the final step, an imaginary part is detected in the same way which is equal to '011' in this case.

Table 5.3 *64-QAM constellation mapping with gray coded*

In-phase	Bit1 Bit2 Bit3	Quadrature	Bit4 Bit5 Bit6
-7	0 0 0	+7	0 0 0
-5	0 0 1	+5	0 0 1
-3	0 1 1	+3	0 1 1
-1	0 1 0	+1	0 1 0
+1	1 1 0	-1	1 1 0
+3	1 1 1	-3	1 1 1
+5	1 0 1	-5	1 0 1
+7	1 0 0	-7	1 0 0

```
for(j = 0 ; j < data ; j++)
        {

        // The real section

                if ( OUTPUT_REAL >= 6){
                        bit1[j] = 1; bit2[j] = 0; bit3[j] = 0;
                }
```

```
9                   else if ( OUTPUT_REAL >= 4 )
10                  if( OUTPUT_REAL < 6 ){
11                          bit1[j] = 1; bit2[j] = 0; bit3[j] = 1;
12
13                          }
14                  }
15                  else if ( OUTPUT_REAL >= 2 ){
16                  if( OUTPUT_REAL < 4 ){
17                          bit1[j] = 1; bit2[j] = 1; bit3[j] = 1;
18
19                          }
20                  }
21                  else if ( OUTPUT_REAL >= 0 ){
22                  if( OUTPUT_REAL < 2 ){
23                          bit1[j] = 1; bit2[j] = 1; bit3[j] = 0;
24
25                          }
26                  }
27                  else if ( OUTPUT_REAL >= -2 ){
28                  if( OUTPUT_REAL < 0 ){
29                          bit1[j] = 0; bit2[j] = 1; bit3[j] = 0;
30
31                          }
32                  }
33                  else if ( OUTPUT_REAL >= -4 ){
34                  if( OUTPUT_REAL < -2 ){
35                          bit1[j] = 0; bit2[j] = 1; bit3[j] = 1;
36
37                          }
38                  }
39                  else if ( OUTPUT_REAL >= -6 ){
40                  if( OUTPUT_REAL < -4 ){
41                          bit1[j] = 0; bit2[j] = 0; bit3[j] = 1;
42
43                          }
44                  }
45                  else{
46                          bit1[j] = 0; bit2[j] = 0; bit3[j] = 0;
47
48                  }
49
50
51
52          // The imaginary section
53
54                  if ( OUTPUT_IMAGE >= 6){
55                          bit4[j] = 0; bit5[j] = 0; bit6[j] = 0;
```

```
56                  }
57                  else if ( OUTPUT_IMAGE >= 4 )
58                  if( OUTPUT_IMAGE < 6 ){
59                          bit4[j] = 0; bit5[j] = 0; bit6[j] = 1;
60
61                          }
62                  }
63                  else if ( OUTPUT_IMAGE >= 2 ){
64                  if( OUTPUT_IMAGE < 4 ){
65                          bit4[j] = 0; bit5[j] = 1; bit6[j] = 1;
66
67                          }
68                  }
69                  else if ( OUTPUT_IMAGE >= 0 ){
70                  if( OUTPUT_IMAGE < 2 ){
71                          bit4[j] = 0; bit5[j] = 1; bit6[j] = 0;
72
73                          }
74                  }
75                  else if ( OUTPUT_IMAGE >= -2 ){
76                  if( OUTPUT_IMAGE < 0 ){
77                          bit4[j] = 1; bit5[j] = 1; bit6[j] = 0;
78
79                          }
80                  }
81                  else if ( OUTPUT_IMAGE >= -4 ){
82                  if( OUTPUT_IMAGE < -2 ){
83                          bit4[j] = 1; bit5[j] = 1; bit6[j] = 1;
84
85                          }
86                  }
87                  else if ( OUTPUT_IMAGE >= -6 ){
88                  if( OUTPUT_IMAGE < -4 ){
89                          bit4[j] = 1; bit5[j] = 0; bit6[j] = 1;
90
91                          }
92                  }
93                  else{
94                          bit4[j] = 1; bit5[j] = 0; bit6[j] = 0;
95                  }
96          }
```

Program 5.2 *C code of Data Symbols Demapping*

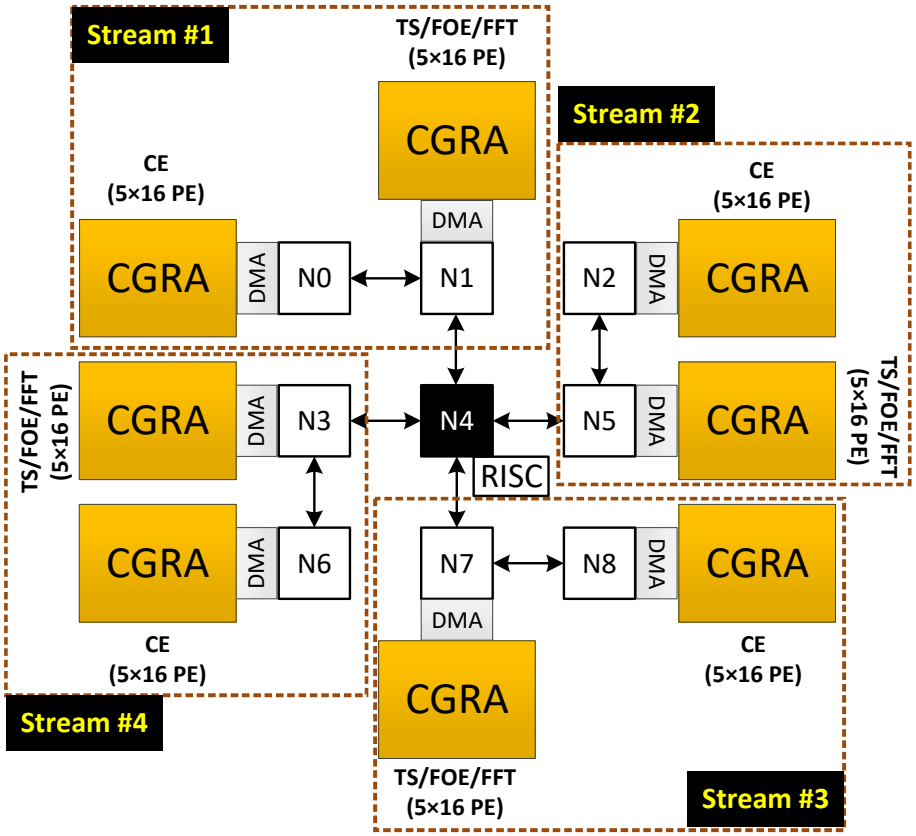

Figure 6.1 *Abridged general view of IEEE 802.11n MIMO receiver on HARP platform.*

6. INTEGRATION OF BASEBAND PROCESSING BLOCKS ON HARP

Figure 6.1 shows the overall architecture for processing MIMO OFDM after integration over HARP. The CGRAs are used to create application-specific accelerators for the receiver blocks. Also, the size of each template-based CGRA on HARP is shown in Table 6.1. In the proposed platform for this thesis, the designer can write the distributed control for the data transfer. The RISC core transfers the configuration stream and data for processing to CGRAs where node N4 RISC is responsible for N1, N3, N5 and N7 CGRAs. By considering the MIMO OFDM as a case-study, the designed platform is divided into 4 parts including streams 1-4. The RISC core is instantiated on N4, which sends the data to all the streams. N1, N5, N7 and N3 are

responsible for streams 1 to 4 respectively. To perform TS, FOE and FFT blocks, the received data will be loaded and transferred to the N1 prior to the system start-up and transfer of the configuration stream with the RISC core.

Table 6.1 shows the Clock Cycles (CC) needed for data transfer from data memory of a node to the other one and to CGRA's local memory while processing at different stages. Also, using the other CGRA's local memory from Stream 1, CH block can be performed. Transfer of data from local memory to another local memory is performed directly. To execute CE block, the output of the FFT is stored to the local memory of N1 which has to be transferred to the local memory of N0 directly by the DMA. Moreover, to complete the task of CE, the results should be first transferred back to the data memory of N1 and at a later stage, to the data memory of N4. Table 6.1 shows that for all data transfers from N4 RISC to N1 CGRA, 4,595 CC are required. Also, correlation can be done in 2,371 CC. When the algorithm of correlation is executed by N1, the results will return to N4 for calculating the offset time index. This process needs 13,738 CC in order to transfer data to data memory from CGRA. The calculation of the SM as well as finding the time offset index can be carried out in 4,179 CC by the RISC core.

After that, short training symbols must be loaded to the local data memory of N1 in order to execute complex multiplication. It takes 4,665 CC to load the data from data memory of N4 to N1. A RISC core is required to perform some parts of FOE block that needs data exchange between N1 and N4 twice. Therefore, in order to perform some parts of FOE by N4 RISC core, the result of complex multiplication must come back to the data memory of N4. Generally, it is possible to divide execution time for performing FOE in 12,634 and 74 CCs which are executed by N4 RISC core and the template-based CGRA mapped on N1, respectively. After completing Equation 3.20 by the designed CGRA located at N1 in the last stage, it can be reconfigured at run-time to perform a 128-point FFT (radix(2-4)) on the results from N1 in 571 CC. When the FFT execution is completed, a transfer of the results by the DMA of N1 from the local memory of CGRA to the local memory of N0 is performed. Also, the data symbols have to be transferred to the local data memory of N0 to execute the channel estimation, which takes 2,585 CC. Once the channel estimation is completed by N0 in 479 CC, the final results should be transferred back to the data memory of N4 to perform symbols demapping in the last stage. Furthermore, the configuration streams of all the slave nodes are loaded in parallel mode to speed-up the execution time of the entire platform.

Moreover, Table 6.1 shows the number of CC needed to run each block by the CGRA and the data transfer among the data memories. The table demonstrates

Table 6.1 *The required clock cycles at different stages for data transfer and processing. In the table, D. Mem, Trans and Exe. are referring to Data memory, Transfer and Execution respectively, while Clock cycles with * sign indicate data transfer from CGRA to Node's data memory.*

Node-to -Node	**D. Mem to D. Mem**	**D. Mem to CGRA**	**Trans. Total**	**Exe. Total**
N4-N1 (Correlation)	2,680	1,915	4,595	2,371
N1-N4 (SM)	-	13,738*	-	4,179
N4-N1	47	-	-	-
N4-N1 (FOE)	1,961	2,704	4665	12,634 (+74)
N4-N1 (FFT)	2,042	1,033	3,075	571
N1-N0 (CE)	1,753	832	2,585	479
N0-N1-N4	-	815*	-	4,378

three types of data transfers. The first category relates to the transmission of data from the RISC core data memory to the CGRA's data memory, the second, data transfer to the local memory banks from the data memory and the third category, the transfer within a slave node by employing a DMA device. Furthermore, the total run-time for FOE, TS, CE blocks and FFT will be 12708, 6550, 479, 571 CCs, respectively. Also, RISC core can perform symbols demapping in 4,378 CCs. Based on the algebraic equations, CGRAs are designed to perform MIMO OFDM receivers. These CGRAs are made in an optimal way as most of the PEs are used in each context. The desirable mapping of an application at design time is essential for performance improvement, power and development time, and region utilization which need scaling up/down the CGRA.

7. MEASUREMENTS AND ESTIMATION

For prototyping purposes, the entire platform is incorporated to a Stratix-V (5SGSED8N3F45I3YY) FPGA device. The operating temperatures are considered to be $-40°C$ for low and $100°C$ for high junction temperature points. With the mentioned conditions, operation frequencies after the placement and routing are 197.82 MHz and 187.72 MHz at $-40°C$ and $100°C$ for slow timing model (900mV). On the other hand, the fast timing model (900 mV) led to higher frequencies of 321.75 MHz at $-40°C$ and 284.17 MHz at $100°C$, respectively. It is worth mentioning that a single clock source has been utilized throughout the work.

Table 7.1 illustrates how the resources are employed for the proposed platform by presenting the number of Adaptive Logic Modules (ALMs), Memory Bits, Registers and DSP elements. As it can be seen, about 98 percent of the entire resources are used in the design stage while a total of 1,164 (59 percent) "18-bit DSPs" resources are employed which are dependent on the number of "32-bit multipliers" since each of these "32-bit multipliers" defined in PE consists of two "18-bit DSPs" for being able to be synthesized on FPGA. The mentioned "32-bit multipliers" in each receiver block as well as a RISC on a NoC node are also provided in Table 7.1.

Post placement and routing information evaluate the platform's power dissipation in such a way that PowerPlay Power Analyzer Tool of Quartus II 15.0 at $25°C$ temperature with frequency of 200.0 MHz is used. Therefore, the overall estimation process result is highly reliable. The gate-level netlist of the platform has obtained these estimations and ModelSim software was used to produce the Value Change Dump (VCD) file taht includes the signal transition information when OFDM receiver is executed. Signal switching activity contributes to the dynamic power and powering on the FPGA contributes the static power. The estimation tool Quartus II computes the dynamic power, static power and I/O thermal power dissipation equal to 5,506.4 mW, 1,541.82 mW and 56.68 mW respectively. Accordingly, the total power dissipation of the FPGA by MIMO OFDM platform is 7.1 W.

From the table 7.2, the dynamic power rises along with the size of CGRA. It should be noticed that with scaling up the CGRA, the static power remains almost the

Table 7.1 *Summary of resource utilization based on the breakdown of node-by-node for Stratix-V (5SGSED8N3F45I3YY) FPGA device*

Node	ALMs	Registers	Memory Bits	(32-bit Multipliers) DSPs
N0 (TS/FOE/FFT)	32,237	21,252	2,634,456	(80) 160
N1 (CE)	30,101	16,625	2,632,672	(64) 128
N2 (TS/FOE/FFT)	32,234	21,249	2,634,456	(80) 160
N3 (CE)	30,099	16,621	2,632,672	(64) 128
N4 (RISC)	6,604	5,752	4,194,304	(6) 12
N5 (CE)	30,101	16,623	2,632,672	(64) 128
N6 (TS/FOE/FFT)	32,240	21,254	2,634,456	(80) 160
N7 (CE)	30,100	16,631	2,632,672	(64) 128
N8 (TS/FOE/FFT)	32,239	21,267	2,634,456	(80) 160
NoC	2,592	4,218	-	-
	258,547	161,492	25,262,816	(582) 1164
Total	98.53%	30.77%	48.05%	59.3%

Table 7.2 *Dynamic power of each CGRA node and the NoC.*

Node	Dynamic Power (mW)
N0	518.51
N1	532.96
N2	521.12
N3	534.18
N4	358.79
N5	537.08
N6	520.17
N7	531.83
N8	515.93
NoC	101.68
Integration Logic	834.17
Total	5,506.4

same. Also, energy consumption for each node is computed as the product of power dissipation and runtime.

In this case study, the current instance of the platform contains 640 PEs. By taking into account the operating frequency of 200 MHz and also total power dissipation of 7.1 W in this model, this HARP instance can provide 128 Giga Operations Per Second (GOPS) and 0.018 GOPS/mW for Altera Stratix-V chips in 28 nm.

8. CONCLUSIONS AND FUTURE WORK

In this thesis work, OFDM receiver blocks are implemented to fully examine functional features and architectural capabilities of a HMA such as HARP platform. The computational and processing power required by the MIMO OFDM receiver for parallel and serial algorithms is intensive enough to make them potential candidates for discovering all HARP design features. In this case, algorithm implementation involves TS, FOE, FFT, CE and symbols demapping for four parallel streams.

The designed receiver using HARP platform has been prototyped on the FPGA device which operates at frequency of 200.0 MHz at room temperature of 25°C. Furthermore, the HARP platform offers a performance of 128 GOPS and 0.018 GOPS/mW.

The designed application-specific accelerators satisfy the IEEE 802.11n standard. Moreover, the simulation results and comparisons with other modern platforms prove the advantages of maximizing the number of reconfigurable processing resources over a platform since the integrated CGRAs can be scaled to numerous dimensions.

There are two motivations for the design of HARP, one that deals with the dark-silicon problem by replacing the underutilized section of the chip with special-purpose accelerators, and the other which maximizes the number of PEs available on the chip so that the demand for throughput can be met.

In this thesis, a MIMO OFDM according to IEEE 802.11n standard specification has been designed by crafting template-based CGRAs. It makes the whole design process easier for those application developers who are not familiar with VHDL. Moreover, the computationally intensive tasks have been parallelized which led to achieve a considerable level of performance. According to the conducted experimental results, the total power dissipation is equal to 7.1 W for the whole platform. In terms of resource utilization, in total, 98.53% ALMs, 30.77% Registers, 48.05% Memory Bits and 59.3% 32-bit DSPs have been utilized. In terms of the required clock cycles for executing each task by designed CGRAs, we achieved 2,371 CC for TS block, 12,708 CC for FOE block, 571 CC for FFT block and 479 CC for CE block.

As a future work, the HARP architecture can be selected in order to implement Massive-MIMO OFDM as a candidate for 5G. Moreover, the power dissipation of the implemented design in this thesis can be mitigated by applying self-aware computing models.

BIBLIOGRAPHY

[1] R. Airoldi, F. Garzia, O. Anjum and J. Nurmi, "Homogeneous MPSoC as baseband signal processing engine for OFDM systems", International Symposium on System on Chip (SoC), 2010, pp. 26-30, Sept. 2010, doi: 10.1109/ISSOC.2010.5625562.

[2] D. Melpignano, L. Benini, E. Flamand, B. Jego, T. Lepley, G. Haugou, F. Clermidy and D. Dutoit, "Platform 2012, a Many-Core Computing Accelerator for Embedded SoCs: Performance Evaluation of Visual Analytics Applications", in Proc. 49th Annual Design Automation Conference (DAC '12). ACM, pp. 1137-1142, New York, NY, USA.

[3] N. S. Voros, M. Hübner, J. Becker, M. Kühnle, F. Thomaitiv, A. Grasset, P. Brelet, P. Bonnot, F. Campi, E. Schler, H. Sahlbach, S. Whitty, R. Ernst, E. Billich, C. Tischendorf, U. Heinkel, F. Ieromnimon, D. Kritharidis, A. Schneider, J. Knaeblein and W. Putzke-Rming, "MORPHEUS: A Heterogeneous Dynamically Reconfigurable Platform for Designing Highly Complex Embedded Systems", ACM Trans. Embed. Comput. Syst. 12, 3, Article 70, 33 pages, April 2013.

[4] F. Thoma, M. Kuhnle, P. Bonnot, E. M. Panainte, K. Bertels, S. Goller, A. Schneider, S. Guyetant, E. Schuler, K. D. Muller-Glaser and J. Becker, "MORPHEUS: Heterogeneous Reconfigurable Computing", International Conference on Field Programmable Logic and Applications, FPL 2007, pp. 409-414, August 2007.

[5] M.B. Taylor, "Is dark silicon useful?: harnessing the four horsemen of the coming dark silicon apocalypse", In proceedings of the 49th Annual Design Automation Conference (DAC 2012), pp. 1131-1136, San Francisco, CA, USA.

[6] A. Shrivastava, J. Pager, R. Jeyapaul, M. Hamzeh and S. Vrudhula, "Enabling Multithreading on CGRAs", 2011 International Conference on Parallel Processing.

[7] W. Hussain, R. Airoldi, H. Hoffmann, T. Ahonen and J. Nurmi, "Design of an accelerator-rich architecture by integrating multiple heterogeneous coarse grain recongurable arrays over a networkon-chip," in Proc. IEEE 25th Int. Conf. Appl.-Specific Syst. Archit. Processors, 18-20 Jun 2014, pp. 131-138.

[8] C. Brunelli, F. Garzia and J. Nurmi, "A Coarse-Grain Reconfigurable Architecture for Multimedia Applications Featuring Subword Computation Capabilities", in Journal of Real-Time Image Processing, Springer Verlag, 2008, 3 (1-2): 21-32. doi:10.1007/s11554-008-0071-3.

[9] F. Garzia, W. Hussain and J. Nurmi, "CREMA, A Coarse-Grain Reconfigurable Array with Mapping Adaptiveness", in Proc. 19th International Conference on Field Programmable Logic and Applications (FPL 2009). Prague, Czech Republic: IEEE, September 2009.

[10] W. Hussain, F. Garzia, T. Ahonen and J. Nurmi, "Designing Fast Fourier Transform Accelerators for Orthogonal Frequency-Division Multiplexing Systems", Journal of Signal Processing Systems for Signal, Image and Video Technology, Springer, Vol. 69, pp. 161-171, November 2012.

[11] W. Hussain, X. Chen, G. Ascheid and J. Nurmi, "A Reconfigurable Application-specific Instruction-set Processor for Fast Fourier Transform processing", 2013 IEEE 24th International Conference on Application- Specific Systems, Architectures and Processors (ASAP), pp. 339-345, 5-7 June 2013, Washington, D.C., USA.

[12] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh and E. M. C. Filho, "Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications", IEEE Trans. Computers, vol. 49, no. 5, pp. 465-481, 2000.

[13] B. Mei, S. Vernalde, D. Verkest, H. D. Man and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix", Field-Programmable Logic and Applications, vol. 2778, pp. 61-70, September 2003, ISBN: 978-3-540-40822-2.

[14] J. M. P. Cardoso and M. Weinhardt, "XPP-VC: A C Compiler with Temporal Partitioning for the PACT-XPP Architecture", in Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream, Editors: M. Glesner and P. Zipf and M. Renovell, Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 864-874, Vol. 2438, ISBN: 978-3-540-44108-3, 2002.

[15] W. Hussain, T. Ahonen F. Garzia and J. Nurmi, "Application-Driven Dimensioning of a Coarse-Grain Reconfigurable Array", 2011 IEEE NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2011), pp. 234-239, San Diego, California, USA.

[16] P. Bonnot, F. Lemonnier, G. Edelin, G. Gaillat, O. Ruch and P. Gauget, "Definition and SIMD implementation of a multi-processing architecture approach on FPGA". In Proc. of Design, Automation and Test in Europe (DATE '08). ACM, pp. 610-615, New York, NY, USA.

[17] Hennessy JL, Patterson DA (1990) Computer Architecture: A Quantitative Approach. 3rd edn. Elseview Morgan Kaufmann, San Francisco.

[18] C. Panis, "VLIW DSP Processor for High-End Mobile Communication Applications", In Processor Design: System-on-Chip Computing for ASICs and FPGAs, J. Nurmi, Ed. Kluwer Academic Publishers / Springer Publishers, June 2007, ch. 5, pp. 83-100, ISBN-10: 1402055293, ISBN-13: 978-1-4020-5529-4.

[19] D. Vassiliadis, N. Kavvadias, G. Theodoridis and S. Nikolaidis, "A RISC architecture extended by an efficient tightly coupled reconfigurable unit". In Proc. ARC, 2005.

[20] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh and E. M. C. Filho, "Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications", IEEE Trans. Computers, vol. 49, no. 5, pp. 465-481, 2000.

[21] www.altera.com

[22] www.xilinx.com

[23] Altera(2006) Stratix II vs. Virtex-4 Density Comparison. Consulted on 16 February 2007. Altera white paper at http://www.altera.com

[24] S. Vassiliadis, S. Wong, G. N. Gaydadjiev, K. Bertels, G. Kuzmanov and E. M. Panainte, "The Molen Polymorphic Processor", IEEE Transactions on Computers, vol. 53, pp. 1363-1375, November 2004.

[25] E. M. Panainte, K. Bertels and S. Vassiliadis, "The Molen Compiler for Reconfigurable Processors", ACM Trans. Embed. Comput. Syst., Vol. 6, ISSN: 1539-9087, New York, USA, February 2007.

[26] W. Hussain, "Design and development from single core reconfigurable accelerators to a heterogeneous accelerator-rich platform", Tampere University of Technology, p. 128, ISBN: 978-952-15-3406-5, Tampere, Finland, 27 Nov 2014.

[27] A. Lodi, C. Mucci, M. Bocchi, A. Cappelli, M. De Dominicis and L. Ciccarelli, "A Multi-Context Pipelined Array for Embedded Systems", International Conference on Field Programmable Logic and Applications, 2006. FPL06, pp. 18, Aug. 2006.

[28] A. Lodi, M. Toma, F. Campi, A. Cappelli, R. Canegallo and R. Guerrieri, "A VLIW Processor with Reconfigurable Instruction Set for Embedded Applications", IEEE Journal of Solid-State Circuits, Vol. 38, pp. 1876-1886, Nov. 2003, doi: 10.1109/JSSC.2003.818292.

[29] Chia-Cheng Lo, Shang-Ta Tsai and Ming-Der Shieh, "A reconfigurable architecture for entropy decoding and IDCT in H.264" VLSI Design, Automation and Test, 2009. VLSI-DAT 09. International Symposium on, pp. 279-282, April 2009, doi: 10.1109/VDAT.2009.5158149, ISBN: 978-1-4244-2781-9.

[30] P. Kunjan and C. Bleakley, "Systolic Algorithm Mapping for Coarse Grained Reconfigurable Array Architectures", Reconfigurable Computing: Architectures, Tools and Applications, Lecture Notes in Computer Science, 2010, Springer Berlin Heidelberg, pp. 351-357, Vol. 5992, doi: 10.1007/978-3-642-12133-3 33.

[31] F. Garzia, W. Hussain, R. Airoldi and J. Nurmi, "A Reconfigurable SoC tailored to Software Defined Radio Applications", in Proc of 27th Norchip Conference, Trondheim (NO), 2009.

[32] Y. Kishimoto, S. Haruyama and H. Amano, "Design and Implementation of Adaptive Viterbi Decoder for Using A Dynamic Reconfigurable Processor", in Proc. Reconfigurable Computing and FPGAs, Dec. 2008, pp. 247-252, doi= 10.1109/ReConFig.2008.39, ISBN: 978-1-4244-3748-1.

[33] S. Nouri , W. Hussain and J. Nurmi, "Evaluation of a Heterogeneous Multicore Architecture by Design and Test of an OFDM Receiver", IEEE Transactions on Parallel and Distributed Systems, VOL. 28, NO. 11, NOVEMBER 2017.

[34] F. Conti, C. Pilkington, A. Marongiu and L. Benini, "He-P2012: architectural heterogeneity exploration on a scalable many-core platform", in Proc. of the 24th edition of the great lakes symposium on VLSI (GLS-VLSI '14), pp. 231-232, ACM, New York, NY, USA.

[35] Roberto Airoldi, "Design and Implementation of Software Defined Radios on a Homogeneous Multi-Processor Architecture", Tampere University of Technology, Vol. 1136, ISBN: 978-952-15-3078-4, May 2013, Tampere, Finland.

[36] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, Jae-Wook Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe and A. Agarwal, "The Raw microprocessor: a computational fabric for software circuits and general-purpose programs," Micro, IEEE, vol. 22, no. 2, pp. 25-35, 2002.

[37] M. B. Taylor, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, J. Kim, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe and A. Agarwal, "Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams", SIGARCH Comput. Archit. News 32, March 2004.

[38] R. Hartenstein, "Coarse grain reconfigurable architecture", ASP-DAC '01 Proceedings of the 2001 Asia and South Pacific Design Automation Conference, pp. 564-570, Yokohama, Japan.

[39] R. Airoldi, F. Garzia, O. Anjum and J. Nurmi, "Homogeneous MPSoC as baseband signal processing engine for OFDM systems", International Symposium on System on Chip (SoC), 2010, pp. 26-30, Sept. 2010, doi: 10.1109/ISSOC.2010.5625562.

[40] Man-On Pun, M. Morelli and C-C Jay Kuo, "Multi-carrier techniques for broadband wireless communications: a signal processing perspective", copyright ©2007 by Imperial College Press, December 2007.

[41] S. Afrasiabi Gorgani, "Peak Power Reduction In Multicarrier Waveforms", Master Thesis, pp. 77, May 2014, Tampere University of Technology, Tampere, Finland.

[42] J. Heiskala and J. Terry, "OFDM Wireless LANs: A Theoretical and Practical Guide", 336 pages, SAMS, 2001, ISBN: 0672321572, Indianapolis, Indiana, USA.

[43] E. Perahia and R. Stacy, "Next Generation Wireless LANs 802.11n and 802.11ac", 2nd edition, p. 452, 2013, Cambridge University Press.

[44] Supplement to IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-Speed Physical Layer in the 5 GHz Band, IEEE Std 802.11a-1999, 1999, New York, NY, USA.

[45] JERRY R. HAMPTON, "Introduction to MIMO Communications", Published in the United States of America by Cambridge University Press, New York, ISBN: 978-1-107-04283-4, First published 2014.

[46] NADER AL-GHAZU, "A Study of the Next WLAN Standard IEEE 802.11ac Physical Layer", KTH School of Electrical Engineering (EE) Signal Processing, p. 68, 2013, XR-EE-SB 2013:001.

[47] A. F. Molisch, "Wireless Communication", 2nd Edition, John Wiley and Sons Ltd., p. 884, December 2010, ISBN: 978-0-470-74186-3.

[48] A. Peled and A. Ruiz, "Frequency domain data transmission using reduced computational complexity algorithms", Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '80, Vol. 5, IEEE, April 1980, doi: 10.1109/ICASSP.1980.1171076.

[49] M. Valkama and M. Renfors, "COMMUNICATION THEORY", http://www.cs.tut.fi/kurssit/TLT-5206/general.html

[50] L. Liang, J. Shi, L. Chen and S. Xu, "Implementation of Automatic Gain Control in OFDM digital receiver on FPGA", 2010 International Conference on Computer Design and Applications (ICCDA), vol. 4, pp. 446-449, June 2010.

[51] http://www.mathworks.com

[52] A. Mueen, A. Nath and J. Liu, "Fast approximate correlation for massive time-series data", Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp. 171-182, Indianapolis, Indiana, USA, June 2010.

[53] L. Harju and J. Nurmi, "Hardware platform for software-defined WCDMA/OFDM baseband receiver implementation", IET Computers and Digital Techniques, vol. 1, no. 5, pp. 640-652, September 2007.

[54] J.-J. van de Beek, P.O. Borjesson, M.-L. Boucheret, D. Landstrom, J.M. Arenas, P. Odling, C. Ostberg, M. Wahlqvist and S.K. Wilson, "A time and frequency synchronization scheme for multiuser OFDM", IEEE Journal on Selected Areas in Communications, vol. 17, no. 11, pp. 1900-1914, 1999.

[55] J. Rinne, "Multicarrier Techniques", http://www.cs.tut.fi/kurssit/ TLT-5706/

[56] T. Hwang, C. Yang, G. Wu, S. Li and G.Y. Li, "OFDM and Its Wireless Applications: A Survey", IEEE Transactions on Vehicular Technology, vol. 58, no. 4, pp. 1673-1694, May 2009.

[57] R. G. Lyons, "Understanding Digital Signal Processing", Boston, MA, USA: Addison-Wesley, 1999.

[58] W. Hussain, F. Garzia and J. Nurmi, "Evaluation of Radix-2 and Radix-4 FFT Processing on a Reconfigurable Platform", in Proc. IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems, pp. 249-254, Vienna, Austria, April 2010, doi: 10.1109/DDECS.2010.5491775.

[59] S. Takaoka and F. Adachi, "Pilot-assisted adaptive interpolation channel estimation for OFDM signal reception", IEEE 59th Vehicular Technology Conference (VTC 2004-Spring), vol. 3, pp. 1777-1781, May 2004.

[60] R. Hajizadeh, K. Mohamedpor and M.R. Tarihi, "Channel Estimation in OFDM system Based on the Linear Interpolation, FFT and Decision Feed-back", 18th Telecommunication forum TELFOR, Serbia, Belgrade, November 2010.

[61] S. Coleri, M. Ergen, A. Puri and A. Bahai, "Channel Estimation Techniques Based on Pilot Arrangement in OFDM Systems", IEEE TRANSACTIONS ON BROADCASTING, pp. 223-229, vol. 48, no. 3, 2002.

[62] M. Renfors and M. Valkama, "DIGITAL TRANSMISSION", http://www.cs.tut.fi/kurssit/TLT-5406/

[63] Filipo Tosato and Paola Bisaglia, "Simplified Soft-Output Demapper for Binary Interleaved COFDM with Application to HIPERLAN/2", IEEE International Conference on Communications, HPL-2001-246, October 2001, New York, NY, USA.

[64] R.V. Nee and R. Prasad, "OFDM for Wireless Multimedia Communications", Copyright 2000 by Artech House, Inc. Norwood, MA, USA.

[65] J. Kylliainen, T. Ahonen and J. Nurmi, "General-purpose embedded processor cores - the COFFEE RISC example", Processor Des.: Syst.-Chip Comput. ASICs FPGAs, J. Nurmi, Ed. Kluwer Academic Publishers Springer Publishers, ch. 5, pp. 83-100, 2007, doi: 10.1007/978-1-4020-5530-0-5.

[66] C. Brunelli, F. Garzia, C. Giliberto and J. Nurmi, "A dedicated DMA Logic addressing a time multiplexed memory to reduce the effects of the system buss bottleneck", in Proc. 18th Int. Conf. Field Program. Logic Appl., 2008, pp. 487-490.

[67] F. Garzia, C. Brunelli and J. Nurmi, "A pipelined infrastructure for the distribution of the configuration bitstream in a coarse-grain reconfigurable array", in Proceedings of the 4th International Workshop on Reconfigurable Communication-centric System-on-Chip (ReCoSoC '08). Univ Montpellier II, July 2008, pp. 188-191, ISBN:978-84-691-3603-4.

[68] J. E. Volder, "The CORDIC Trigonometric Computing Technique", IRE Transactions on Electronic Computers, pp. 330-334, September 1959.

[69] P. K. Meher, J. Valls, T-B Juang, K. Sridharan and K. Maharatna, "50 Years of CORDIC: Algorithms, Architectures and Applications", IEEE Transactions on

Circuits and Systems-I: Regular Papers, vol. 56, no. 9, pp. 1893-1907, September 2009.

[70] J. E. Meggitt, "Pseudo Division and Pseudo Multiplication Processes", IBM Journal of Research and Development, pp. 210-226, April 1962, doi: 10.1147/rd.62.0210.

[71] M. D. Greenberg, "Foundations of Applied Mathematics", Dover Publications, Inc. Mineola, New York, USA, p. 656, 1978, ISBN: 9780486492797.

[72] V. S. Ryabenkii and S. V. Tsynkov, "A Theoretical Introduction to Numerical Analysis", p. 537, Boca Raton, FL, USA: CRC Press, ISBN: 1584886072, 2006.

[73] "Design implementation and optimization quartus-II handbook version 13.1", San Jose, CA, USA, Altera Corporation, May 2013.

[74] W. Hussain, R. Airoldi, H. Hoffmann, T. Ahonen and J. Nurmi, "HARP2: An X-scale reconfigurable accelerator-rich platform for massively-parallel signal processing algorithms", Springer's J. Signal Process. Syst., vol. 85, no. 3, pp. 341-353, 2015.

[75] D. Westcott, D. Coleman, P. Mackenzie and B. Miller, "CWAP Certified Wireless Analysis Professional Official Study Guide", Sybex, p. 696, March 2011, ISBN: 978-0-470-76903-4.

[76] K. Ian and J. Rose, "Measuring the gap between FPGAs and ASICs", IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 26, no. 2, pp. 203-215, 2007.

[77] F. Garzia, "From run-time reconfigurable coarse-grain arrays to application-specific accelerators design", Ph.D. dissertation, Tampere University of Technology (TUT), Tampere, Finland, 2009, p. 125, TUT Publications 860, ISBN: 978-952-15-2280-2.

[78] W. Hussain, J. Nurmi, J. Isoaho and F. Garzia, "Computing Platforms for Software-Defined Radio", p. 240, Springer, ISBN: 978-3-319-49678-8, doi: 10.1007/978-3-319-49679-5, 2017.

[79] https://www.mentor.com/products/fv/modelsim

APPENDIXE 1

Simulation of The OFDM

```
%% OFDM transceiver with IEEE 802.11n %%%%%%%
%% By Mohammad Hosseinvand %%%%%%%%%%%%%%%%%%
%% FFT: 128   %64 QAM %%%%%%%%%%%%%%%%%%%%%%%%
%% Number of Pilots:6 {-53,-25,-11,11,25,53}%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%
clear all;
clc;

%% *** Initialize the parameters ***
% 802.11n 40MHz channel: 114 sub-carriers (108 data, 6 pilot)

Bit_Num = 6;
FFT_Num = 128;
Carrier_Num = 108;
OFDM_per_Symbol = 1;
% CP=32;
LI = 19;  % B/W  -11 to +11 pilots , there are 19 data
Np = 6;
Carriers = 1:Carrier_Num+Np; % Total no. carreirs 114
N_Num = Bit_Num*Carrier_Num*OFDM_per_Symbol;

%% *** Generate the random binary stream ***

BitTx = round(rand(1,(N_Num)));

%% *** Modulate (generates 64QAM symbols) ***

N_Num =length(BitTx);
SymQAMtmp = reshape(BitTx,6,N_Num/6).';
SymQAMtmptmp = bi2de(SymQAMtmp,2,'left-msb');

QAMTable = [ -7-7j, 7-7j, -1-7j, 1-7j, -5-7j, 5-7j,...
-3-7j, 3-7j, -7+7j, 7+7j, -1+7j, 1+7j,...
```

```
-5+7j , 5+7j , -3+7j , 3+7j , -7-1j , 7-1j ,...
-1-1j , 1-1j , -5-1j , 5-1j , -3-1j , 3-1j ,...
-7+1j , 7+1j , -1+1j , 1+1j , -5+1j , 5+1j ,...
-3+1j , 3+1j , -7-5j , 7-5j , -1-5j , 1-5j ,...
-5-5j , 5-5j , -3-5j , 3-5j , -7+5j , 7+5j ,...
-1+5j , 1+5j , -5+5j , 5+5j , -3+5j , 3+5j ,...
-7-3j , 7-3j , -1-3j , 1-3j , -5-3j , 5-3j ,...
-3-3j , 3-3j , -7+3j , 7+3j , -1+3j , 1+3j ,...
-5+3j , 5+3j , -3+3j , 3+3j ];

SymQAM = QAMTable(SymQAMtmptmp+1);

%% *** Generate the preamble signal ***

fsMHz = 40; % sampling frequency
nFFTSize = 128;
% for each symbol bits a1 to a114 are assigned to subcarrier
% Short preamble
subcarrierIndex = [-58:-2 2:58];
S1 = [zeros(1,6) 0 0 1+1j 0 0 0 -1-1j 0 0 0 1+1j ...
        0 0 0 -1-1j 0 0 0 -1-1j 0 0 0 1+1j 0 0 0 ...
    0 0 0 0 -1-1j 0 0 0 -1-1j 0 0 0 1+1j 0 0 0 ...
        1+1j 0 0 0 1+1j 0 0 0 1+1j 0 0 0 0 0 ...
    0 0 0 0 0 0 0 0 0 0 1+1j 0 0 0 -1-1j ...
        0 0 0 1+1j 0 0 0 -1-1j 0 0 0 -1-1j 0 0 0 1+1j ...
    0 0 0 0 0 0 0 -1-1j 0 0 0 -1-1j 0 0 0 1+1j ...
        0 0 0 1+1j 0 0 0 1+1j 0 0 0 1+1j 0 0 zeros(1,5)];

S1 = sqrt(19/4)*(S1);

% Long preamble
S2 = [zeros(1,8),1,1,-1,-1,1,1,-1,1,1,1,1,1,1,...
         -1,-1,1,1,-1,1,-1,1,1,1,1,1,1,...
     -1,-1,1,1,-1,1,-1,1,-1,-1,-1,-1,...
         -1,1,1,-1,-1,1,-1,1,-1,...
     1,1,1,1,-1,-1,-1,1,0,0,0,-1,1,...
         1,-1,1,1,-1,-1,1,1,-1,1,-1,...
     1,1,1,1,1,1,-1,-1,1,-1,1,-1,1,...
         1,1,1,1,-1,-1,1,1,-1,1,-1,1,...
```

```
    -1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1,zeros(1,7)];


S1_td = ifft(S1,128);
S2_td = ifft(S2,128);

% STEP3 Adding CP to the time domain preamble symbol 1 and symbol 2
CP = ceil(FFT_Num/4);
S2_tdcp = [S2_td(end-CP+1:end) S2_td S2_td];

% Concatenating multiple symbols to form 10 short preamble
S1_tdcp = [S1_td S1_td S1_td(1:32)];
preamble = [S1_tdcp S2_tdcp];


%% *** Generate the pilot signal ***

train_sym1 = round(rand(1,(2*OFDM_per_Symbol)));
train_sym2 = round(rand(1,(2*OFDM_per_Symbol)));

t = 2*(train_sym1.*2-1)+(train_sym2.*2-1);

treal = t(1:2:2*OFDM_per_Symbol);
timage = t(2:2:2*OFDM_per_Symbol);

training_symbols1 = treal+1i*timage;
training_symbols2 = training_symbols1.';
training_symbols = repmat(training_symbols2,1,Np);

pilot = [1,29,43,62,76,104]; % Start position of pilots

%% *** P/S transform ***

SymQAM1 = reshape(SymQAM, Carrier_Num, OFDM_per_Symbol).';

%% *** Insert the pilot symbols ***

signal = 1:Carrier_Num+Np;
signal(pilot) = [];
```

```
SymQAM2(:,pilot) = training_symbols;
SymQAM2(:,signal) = SymQAM1;
SymQAM3 = SymQAM2.';

%% *** IFFT transform ***

SymIFFT=ifft(SymQAM3,FFT_Num,1);

%% *** GI Insertion ***

SymTxtmp = zeros(FFT_Num+CP,OFDM_per_Symbol);
SymTxtmp(1:CP,:) = SymIFFT(FFT_Num-CP+1:FFT_Num,:);
SymTxtmp(CP+1:CP+FFT_Num,:) = SymIFFT;
SymTx = reshape(SymTxtmp,1,(FFT_Num+CP)*OFDM_per_Symbol);

%% *************** Transmitter ends ***************

%% *************** Receiver starts ****************


%% *** Packet detection ***
SymRxdel = [zeros(1,CP) SymTx(1:length(SymTx)-CP)]; % Delayed
y = xcorr(SymTx,SymRxdel);
[m,n] = max(y);
t_offset_Packet_detection = n-1;
SymRx = y(t_offset_Packet_detection+1:end);


plot(abs(y))
% set(gca,'xtick',[0:10:160] )
% set(gca,'ytick',[0:5:50] )
yy = filter(ones(1,CP),1,y); % Sum-window
yy_abs = abs(yy);
pos_s = [];
val_s = [];


% for ii=1:4,
% ys=yy_abs((ii-1)*(FFT_Num+CP)+(1:FFT_Num)); % Search window
```

```
% [val est_pos]=max(ys);
% val_s=[val_s val];
% pos_s=[pos_s est_pos];
% end

%% *** GI Removal ***
SymRxtmp = reshape(SymTx,FFT_Num+CP,OFDM_per_Symbol);
SymRx = SymRxtmp(CP+1:CP+FFT_Num,:);
SymRx = reshape(SymRx,1,128);

%% *** Time offset estimation ***

Toff = input('Enter time offset (for example 100):');
SymRx2 = [zeros(1,Toff),SymRx];
if length(SymRx2)>length(S1_tdcp)
    pad = length(SymRx2)-length(S1_tdcp);
    S1_tdcp = [S1_tdcp zeros(1,pad)];
elseif length(SymRx2)<length(S1_tdcp)
pad = length(S1_tdcp)-length(SymRx2);
SymRx2 = [SymRx2 zeros(1,pad)];
end

out_len = length(SymRx2);
out = zeros(1,out_len);

tmp = S1_tdcp;
for k = 1:out_len
out(k) = SymRx2*tmp';
tmp = [0 tmp(1:end-1)];
end
figure()
% abs_corr = abs(out);
[maxval,index] = max(out);
plot(abs(corr(out)),'Linewidth',1.5)
axis([0 160 0 80])

[m,n] = max(out);
t_offset_Time_estimation = n-1;
plot(abs(corr(out)),'Linewidth',1.5)
```

```
%% *** FFT  transform ***

SymRx = reshape(SymRx,FFT_Num,OFDM_per_Symbol);
SymFFT = fft(SymRx,FFT_Num,1);
SymFFT = SymFFT(1:Carrier_Num+Np,:);

%% *** Channel Estimation ***

% Extract pilots from received signal
SymFFT1 = SymFFT.';
Rx_training_symbols = SymFFT1(:,pilot);
Rx_training_symbols0 = reshape(Rx_training_symbols,...
OFDM_per_Symbol*Np,1);

% reshape original pilots
training_symbols0 = reshape(training_symbols,1,OFDM_per_Symbol*Np);
training_symbols1 = diag(training_symbols0);
training_symbols2 = inv(training_symbols1);
Hls = (training_symbols2)*Rx_training_symbols0;
Hls1 = reshape(Hls,OFDM_per_Symbol,Np);

HLs = [];
HLs2 = [];

for k = 1:Np-1
        HLs2 = [];
        for j = 1:18
            HLs1(:,1) = (Hls1(:,k+1)-Hls1(:,k))*(j-1)/LI+Hls1(:,k);
            HLs2 = [HLs2,HLs1];
        end
        HLs=[HLs,HLs2];
    end

HLs = [HLs,HLs2];
SymFFT1 = SymFFT.';
SymFFT2 = SymFFT1(:,signal);
SymFFT3 = SymFFT2./HLs;
SymFFT = SymFFT3.';
```

```
%% *** Demodulate ***

SymDeQAMtmp = reshape(SymFFT,1,OFDM_per_Symbol*Carrier_Num);

m = real(SymDeQAMtmp);
n = abs(real(SymDeQAMtmp));

p = imag(SymDeQAMtmp);
q = abs(imag(SymDeQAMtmp));

s = 2*(m==n)-1;
t = 2*(p==q)-1;

SymDeQAMtmpshift = SymDeQAMtmp+((-2)*(s+1i*t));

bit0 = imag(SymDeQAMtmp);
bit1 = imag(SymDeQAMtmpshift);
bit2 = real(SymDeQAMtmp);
bit3 = real(SymDeQAMtmpshift);

SymDeQAMtmptmp = zeros(Bit_Num,OFDM_per_Symbol*Carrier_Num);

SymDeQAMtmptmp(1,:)= bit0;
SymDeQAMtmptmp(2,:)= bit1;
SymDeQAMtmptmp(3,:)= bit2;
SymDeQAMtmptmp(4,:)= bit3;

for j =1:(Bit_Num*OFDM_per_Symbol*Carrier_Num)
    if SymDeQAMtmptmp(j)>0
        SymDeQAMtmptmp(j)=0;
    else SymDeQAMtmptmp(j)=1;
    end
end

SymDeQAM = reshape(SymDeQAMtmptmp,1,...
Bit_Num*OFDM_per_Symbol*Carrier_Num);
```

APPENDIXE 2

Power Spectral Density of The OFDM

```
close all;
clear all;
clc;

nFFTSize = 128;
% for each symbol bits a.1 to a.108 are assigned to subcarrier
subcarrierIndex = [-58:-2 2:58];
nBit = 2500;
ip = rand(1,nBit) > 0.5; % generating 1's and 0's
nBitPerSymbol = 114;

nSymbol = ceil(nBit/nBitPerSymbol);

ipMod = 2*ip - 1;
ipMod = [ipMod zeros(1,nBitPerSymbol*nSymbol-nBit)];
ipMod = reshape(ipMod,nSymbol,nBitPerSymbol);

st = []; % empty vector

for ii = 1:nSymbol

inputiFFT = zeros(1,nFFTSize);
inputiFFT(subcarrierIndex+nFFTSize/2+1) = ipMod(ii,:);
inputiFFT = fftshift(inputiFFT);
outputiFFT = ifft(inputiFFT,nFFTSize);
outputiFFT_with_CP = [outputiFFT(97:128) outputiFFT];
st = [st outputiFFT_with_CP];
end

% snr = 0; %Use variable length;
% z = awgn(st,snr,'measured');

close all
fsMHz = 40;
```

```
[Pxx,W] = pwelch(st,[],[],4096,40);
plot([-2048:2047]*fsMHz/4096,10*log10(fftshift(Pxx)),...
'-r','Linewidth',1);
xlabel('Frequency 40.0 MHz');
ylabel('Power spectral density (PSD)');
title('Transmit spectrum OFDM (based on 802.11n)');
legend('Without AWGN channel');
%legend ('SNR = 0 dB')
grid on
```

APPENDIXE 3

Binary to Gray Symbol Mapping For 64 QAM Modulation

```
%% Binary to Gray Symbol Mapping

% Make a vector of 64-QAM
x = (0:63)';

%%
% Convert the input vector (binary order to a Gray-coded)
[y,mapy] = bin2gray(x,'qam',64);

%%
% Convert the Gray-coded symbols, back to a binary ordering using
z = gray2bin(y,'qam',64);

%%
% Vector |z| are identical.
isequal(x,z)

%%
% To make a constellation plot
Mod = comm.RectangularQAMModulator('ModulationOrder',64,...
'BitInput',true);
symbols = constellation(Mod);

%%
% Plot the constellation symbols, Gray-coded |y| and binary |z|
scatterplot(symbols,1,0,'*b');
for k = 1:64
    text(real(symbols(k))-0.3,imag(symbols(k))+0.3,...
        dec2base(mapy(k),2,6));

    text(real(symbols(k))-0.3,imag(symbols(k))-0.3,...
        dec2base(z(k),2,6),'Color',[1 0 0]);
end
grid on
axis([-8 8 -8 8])
```