ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΦΑΡΜΟΣΜΕΝΑ ΗΛΕΚΤΡΟΝΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

# «Υλοποίηση μη ορθογωνικής πολλαπλής πρόσβασης στο επίπεδο της ισχύος (Power Domain NOMA) για συστήματα OFDM με μονάδα ραδιοεπικοινωνίας καθορισμένη από λογισμικό»

**Του φοιτητή**
Γραικούση Χαράλαμπου
**Αρ. Μητρώου:** 51903

**Επιβλέπων Καθηγητής**
Ιωσηφίδης Αθανάσιος
**Βαθμίδα**
Αναπληρωτής Καθηγητής

*Θεσσαλονίκη, Φεβρουάριος 2022*

Τίτλος Δ.Ε.: «Υλοποίηση μη ορθογωνικής πολλαπλής πρόσβασης στο επίπεδο της ισχύος (Power Domain NOMA) για συστήματα OFDM με μονάδα ραδιοεπικοινωνίας καθορισμένη από λογισμικό»
Κωδικός Δ.Ε.: 2013
Ονοματεπώνυμο φοιτητή: Γραικούσης Χαράλαμπος
Ονοματεπώνυμο εισηγητή: Ιωσηφίδης Αθανάσιος
Ημερομηνία ανάληψης Δ.Ε.: 10/9/2020
Ημερομηνία περάτωσης Δ.Ε. 27/2/2022

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Μεταπτυχιακό Πρόγραμμα Σπουδών «Εφαρμοσμένα Ηλεκτρονικά Συστήματα» στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.



Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

*“If your hate could be turned into electricity, it would light up the whole world”*

*Nicola Tesla*

# Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται την υλοποίηση ενός συστήματος μη ορθογωνικής πολλαπλής πρόσβασης στο επίπεδο της ισχύος (PD – NOMA) με μονάδες ραδιοεπικοινωνίας καθορισμένες από λογισμικό (SDR). Πρόκειται για τεχνική που μελετάται ώστε να συμπεριληφθεί στα νέα συστήματα κινητών επικοινωνιών 5ης γενιάς. Ειδικότερα, η τεχνική PD – NOMA εξοικονομεί πόρους (χρόνο – συχνότητα) πολυπλέκοντας στο πεδίο της ισχύος πληροφορίες διαφορετικών χρηστών. Απαραίτητο για τη σωστή λειτουργία της είναι τα σήματα των χρηστών να ξεχωρίζουν μεταξύ τους ως προς την ισχύ. Μελετήθηκε ένα σενάριο κάτω ζεύξης με έναν σταθμό βάσης που επικοινωνεί με δύο κινητούς χρήστες. Τα σήματα που προορίζονται για τους χρήστες υπερτίθενται στο επίπεδο της ισχύος από το σταθμό βάσης και αποστέλλονται μέσω του ίδιου καναλιού. Με τον κατάλληλο σχεδιασμό, οι δέκτες καλούνται να εξάγουν τη πληροφορία που τους αντιστοιχεί. Οι μονάδες ραδιοεπικοινωνίας προγραμματίστηκαν κατάλληλα ώστε να προσομοιώνουν τη λειτουργία του σταθμού βάσης και των κινητών χρηστών. Για το προγραμματισμό χρησιμοποιήθηκε το λογισμικό MATLAB και το SDR τύπου Adalm Pluto της εταιρείας Analog Devices. Στόχος της εργασίας είναι η υλοποίηση της τεχνικής PD – NOMA σε πραγματικές συνθήκες και η εξαγωγή συμπερασμάτων από τα αποτελέσματα, ως προς της πιθανότητα σφάλματος bit για τους δύο διαφορετικούς χρήστες. Από τη μεριά του σταθμού βάσης φάνηκε η σημαντικότητα της διαχείρισης της ισχύος και της επιλογής των κατάλληλων χρηστών προς υπέρθεση. Από την άλλη, είναι σημαντικός ο κατάλληλος σχεδιασμός των δεκτών για την ορθή λήψη ενός υπερτεθειμένου σήματος. Επιπλέον, χρησιμοποιήθηκαν διαφορετικά σχήματα υπερτεθειμένης διαμόρφωσης και κατανομές της ισχύος και συγκρίθηκαν με αυτά των κλασσικών διαμορφώσεων. Συμπερασματικά, τα πειράματα έδειξαν ότι η τεχνική PD – NOMA εφαρμόζεται αποτελεσματικά σε συνθήκες κοντά στις πραγματικές και μπορεί να επιτύχει μια ικανοποιητική πιθανότητα σφάλματος bit, καταδεικνύοντας παράλληλα τη σημασία της κατανομής της ισχύος για τη βελτιστοποίηση της απόδοσης της.

“Implementation of power domain non orthogonal multiple access (PD-NOMA) for OFDM systems on software defined radio (SDR) platform”

“Charalampos Graikousis”

## Abstract

In this thesis, a power domain non-orthogonal multiple access (PD – NOMA) system is implemented on a software defined radio (SDR) platform. This technique has been studied lately in order to be included in future 5G mobile communication networks. PD – NOMA saves resources (time – frequency) by multiplexing information of different mobile users in the power domain, by using signals that stand out from each other in terms of power. A downlink scenario is studied with a base station communicating with two users. The signals intended for the users are superimposed in the power domain at the base station and transmitted through the same channel. The receivers have to extract the information corresponding to each one of them with proper design. The SDRs were programed to simulate the operation of the base station and the mobile users. MATLAB software and Adalm Pluto SDRs from Analog Devices were used for the experimental system. The aim of the work is to implement the PD - NOMA technique in real conditions and to draw conclusions regarding the probability of bit error for the two different users. The results have shown that, from the base station perspecitve, power management and the selection of suitable users for superposition is very important. On the other hand, the proper design of the receivers is important for the correct reception of a superimposed signal. Different modulation schemes and power distributions have been tested and their bit error probability results have been compared with those of the classical modulations. In conclusion, the experiments have shown that PD - NOMA can be effectively applied in practical applications and can achieve a satisfactory bit error probability while demonstrating the importance of power allocation for optimal performarnce.

# Περιεχόμενα

# Κατάλογος Σχημάτων

# Κατάλογος Πινάκων

# Συντομογραφίες

| | |
|---|---|
| BS | Base Station |
| UE | User Equipment |
| NOMA | Non Orthogonal Multiple Access |
| 3G | Third Generation |
| CDMA | Code Division Multiple Access |
| PD – NOMA | Power Domain – Non Orthogonal Multiple Access |
| OFDM | Orthogonal Frequency Division Multiple Access |
| SDR | Software Defined Radio |
| RF | Radio Frequency |
| MATLAB | MathWorks Laboratory |
| BPSK | Binary Phase Shift Keying |
| QPSK | Quadrature Phase shift Keying |
| QAM | Quadrature Amplitude Modulation |
| RAN | Radio Access Network |
| OMA | Orthogonal Multiple Access |
| OFDMA | Orthogonal Frequency Division Multiplexing |
| 5G | Fifth Generation |
| 4G | Fourth Generation |
| FDM | Frequency Division Multiplexing |
| TDM | Time Division Multiplexing |
| SNR | Signal to Noise Ratio |
| SINR | Signal to Interference and Noise Ratio |
| IC | Interference Cancellation |
| WLAN | Wide Local Area Network |
| OSI | Open Systems Interconnection |
| IEEE | Institute of Electrical and Electronics Engineers |
| STA | Station |
| AP | Access Point |
| BSS | Basic Service Set |
| ESS | Extended Service Set |
| DS | Distribution System |

| | |
|---|---|
| PHY | Physical Layer |
| LLC | Logical Link Control |
| MAC | Medium Access Control |
| PLCP | Physical Layer Convergence Procedure |
| FFT | Fast Fourier Transform |
| ISI | Inter Symbol Interference |
| LNA | Low Noise Amplifier |
| ADC | Analog to Digital Conversion |
| IIO | Industrial Input Output |
| API | Application Interface |
| DAC | Digital to Analog Conversion |
| L – LTF | Legacy – Long Training Field |
| L – STF | Legacy- Short Training Field |
| Non – HT | Non – High Throughput |
| PLL | Phase Locked Loop |
| PPM | Parts Per Million |
| CFC | Coarse Frequency Correction |
| FFC | Fine Frequency Correction |
| DA | Data Aided |
| NDA | Non Data Aided |
| CRC | Cyclic Redundancy Check |
| BCC | Binary Convolutional Coding |
| MER | Modulation Error Rate |
| AWGN | Additive White Gaussian Noise |
| SISO | Single Input Single Output |

# Κεφάλαιο 1ο: Εισαγωγή

Στις κινητές επικοινωνίες 4ης και 5ης γενιάς είναι σημαντική η διαχείριση των φυσικών πόρων για την δημιουργία αποδοτικών συστημάτων. Οι συχνότητες του ηλεκτρομαγνητικού φάσματος και ο χρόνος είναι δύο βασικοί πόροι που μας προσφέρει η φύση. Στις κυψελωτές – κινητές τηλεπικοινωνίες ο σταθμός βάσης (Base Station – BS) του παρόχου εξυπηρετεί τους κινητούς χρήστες (User Equipment – UE) μέσα σε μία κυψέλη. Η αποδοτικότητα ενός συστήματος αυξάνεται όταν με τους ίδιους πόρους εξυπηρετούμε περισσότερους χρήστες από όσους εξυπηρετούσαμε μέχρι τώρα. Οι τεχνικές μη ορθογωνικής πολλαπλής πρόσβασης (NOMA) μπορούν να το πετύχουν αυτό, πολυπλέκοντας τους χρήστες και αποστέλλοντας πληροφορία ταυτόχρονα και στην ίδια συχνότητα. Ένα παράδειγμα είναι η τρίτη γενιά κινητών επικοινωνιών (3G) όπου χρησιμοποιείται η τεχνική CDMA. Με αυτή τη τεχνική, ένας σταθμός βάσης χρησιμοποιώντας διαφορετικό κώδικα για κάθε χρήστη, μπορεί να εκπέμψει ταυτόχρονα στην ίδια συχνότητα και οι χρήστες να εξάγουν τη πληροφορία που τους αντιστοιχεί.

Στη παρούσα διπλωματική εργασία υλοποιήθηκε ένα σύστημα μη ορθογωνικής πολλαπλής πρόσβασης στο πεδίο της ισχύος (PD-NOMA). Αυτή η τεχνική μελετάται τα τελευταία χρόνια για να συμπεριληφθεί στις νέες τεχνολογίες 5ης γενιάς κινητών επικοινωνιών όπως επίσης και ως βελτίωση στα OFDM συστήματα της 4ης γενιάς. Η βασική ιδέα είναι ότι αναθέτοντας διαφορετικά ποσοστά ισχύος σε κάθε σήμα πληροφορίας, μπορούμε προσθέτοντας τα αλγεβρικά να τα αποστείλουμε μέσα από το ίδιο κανάλι. Αν για παράδειγμα έχουμε δύο σήματα πληροφορίας που προορίζονται για δύο διαφορετικούς χρήστες μιας κυψέλης, με αυτό τον τρόπο μπορούμε να τα στείλουμε ταυτόχρονα στην ίδια συχνότητα και οι δέκτες των κινητών χρηστών να εξάγουν από το συνολικό σήμα την πληροφορία τους. Βασισμένοι σε τεχνικές ακύρωσης της παρεμβολής που καθίστανται αποδοτικές λόγω της διαφοράς ισχύος των λαμβανόμενων σημάτων.

Για την υλοποίηση του συστήματος χρησιμοποιήθηκαν μονάδες ραδιοεπικοινωνίας καθορισμένες από λογισμικό (Software Defined Radio - SDR) τύπου Adalm – Pluto της εταιρείας Analog Devices. Οι συσκευές αυτές μπορούν να εκπέμψουν και να λάβουν σήματα προγραμματίζοντας τες με γλώσσες όπως η C, η Python και η MATLAB. Είναι εργαλεία με αρκετές δυνατότητες έτσι ώστε να μπορούν να εξομοιώσουν συστήματα τηλεπικοινωνιών με στόχο την εξαγωγή συμπερασμάτων για τη λειτουργία τους. Μπορούν να εκπέμψουν από τα 325MHz έως τα 3.8GHz ενώ το εύρος του καναλιού (channel bandwidth) κυμαίνεται από τα 200KHz στα 20MHz [1][2].

Για τον προγραμματισμό του SDR χρησιμοποιήθηκε το λογισμικό της MATLAB. Πιο συγκεκριμένα, υλοποιήθηκε κώδικας που εκτελεί την εκπομπή και τη λήψη σημάτων PD-NOMA. Για την υλοποίηση του χρησιμοποιήθηκαν οι εργαλειοθήκες (toolbox) της MATLAB με ονόματα “Communications System Toolbox” και “WLAN System Toolbox”. Περισσότερα για τον προγραμματισμό του συστήματος θα παρουσιαστούν σε επόμενο κεφάλαιο.

Σκοπός αυτής της εργασίας είναι η εξομοίωση ενός συστήματος PD – NOMA σε πραγματικές συνθήκες χρησιμοποιώντας μονάδες ραδιοεπικοινωνίας με στόχο την αξιολόγηση των αποτελεσμάτων. Υλοποιήθηκε ένα σύστημα που περιλαμβάνει δύο SDR, εκ των οποίων το ένα λειτουργεί ως σταθμός βάσης κινητής τηλεφωνίας και το άλλο ως κινητός χρήστης. Για τη διαμόρφωση των σημάτων χρησιμοποιήθηκαν σχήματα ψηφιακής διαμόρφωσης BPSK, QPSK και 16QAM. Αρχικά δημιουργήθηκε ένα απλό σύστημα εκπομπής και λήψης που χρησιμοποιεί τη τεχνική PD - NOMA και στη συνέχεια ένα σύστημα WLAN σε συνδυασμό με τη παραπάνω τεχνική. Αφού υλοποιήθηκε το σύστημα η έρευνα συνεχίστηκε με την εξαγωγή των αποτελεσμάτων. Αυτό που θέλαμε να δούμε είναι

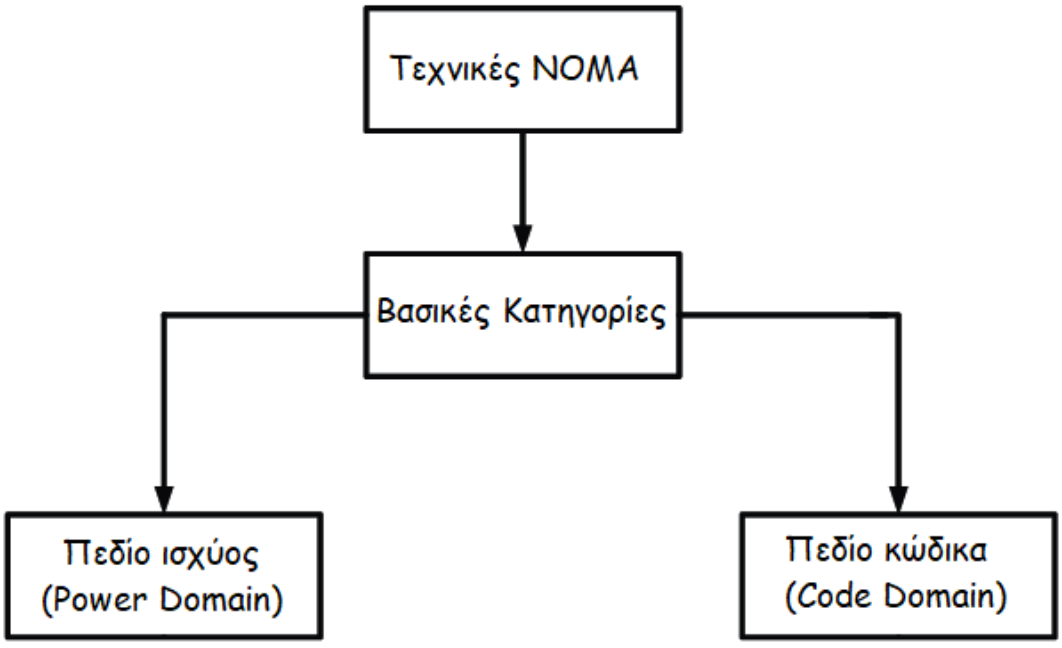
αν ένα τέτοιου τύπου σύστημα λειτουργεί σε πραγματικές συνθήκες όπως φαίνεται να λειτουργεί από τη θεωρία και αν παρατηρείται κάποια διαφορετική συμπεριφορά. Συμπεράναμε ότι η τεχνική PD – NOMA είναι εφικτή. Επίσης λάβαμε αποτελέσματα για διαφορετικές καταστάσεις του συστήματος και δείξαμε τη σημαντικότητα του καταμερισμού της ισχύος και τους συμβιβασμούς που πρέπει να γίνουν για να φτιάξουμε ένα αποδοτικό σύστημα. Τέλος, συγκρίναμε τη τεχνική της υπέρθεσης στο πεδίο της ισχύος με ένα απλό σύστημα κλασσικής διαμόρφωσης και παρουσιάσαμε τα πλεονεκτήματα και τα μειονεκτήματα τους.

Ανακεφαλαιώνοντας, η τεχνική της μη ορθογωνικής πρόσβασης στο πεδίο της ισχύος μπορεί να εξοικονομήσει πόρους (χρόνος – συχνότητα) σε ένα σύστημα κινητών ασυρμάτων επικοινωνιών, αυξάνοντας έτσι την αποδοτικότητα του. Σε αυτή τη διπλωματική εργασία υλοποιήθηκε ένα τέτοιο σύστημα πάνω σε μονάδες ραδιοεπικοινωνίας (SDR) και λήφθηκαν αποτελέσματα όσον αφορά τη πιθανότητα σφάλματος. Στο επόμενο κεφάλαιο γίνεται μια εκτενής θεωρητική ανάλυση της τεχνικής PD-NOMA. Στο τρίτο κεφάλαιο παρουσιάζεται ο κώδικας που χρησιμοποιήθηκε και αναλύεται ο τρόπος που λειτουργεί. Τέλος, στο τέταρτο και στο πέμπτο κεφάλαιο παρατίθενται τα αποτελέσματα και τα συμπεράσματα αντίστοιχα.

# Κεφάλαιο 2ο: Μη ορθογωνική πολλαπλή πρόσβαση στο πεδίο της ισχύος (PD – NOMA)

## 2.1 Βασικές έννοιες

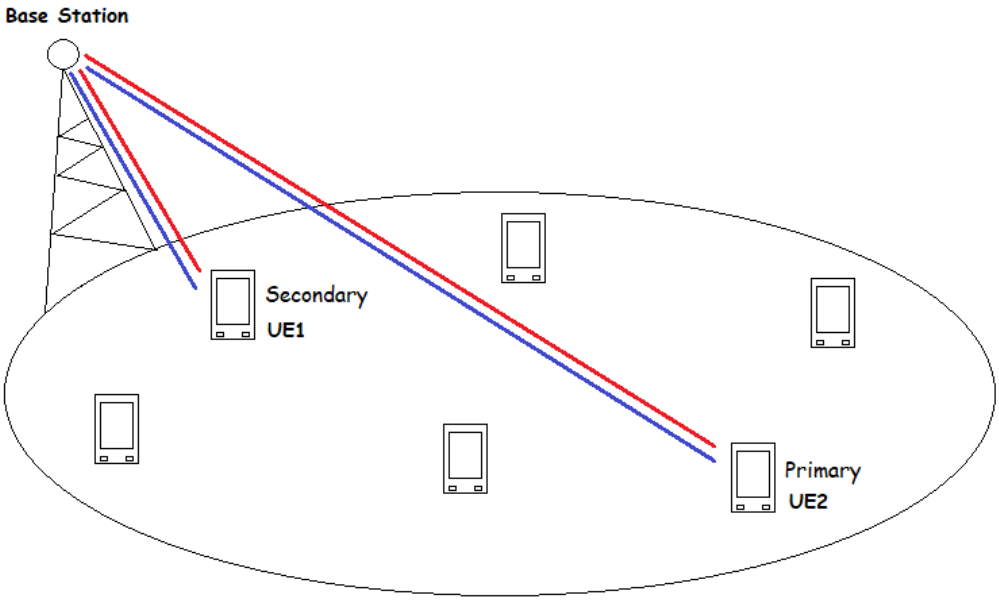
Η εξέλιξη της τεχνολογίας και η συνεχής ανάγκη για συνδεσιμότητα και μεγαλύτερους ρυθμούς μετάδοσης δεδομένων έχουν φέρει στο προσκήνιο τις τεχνικές μη ορθογωνικής πολλαπλής πρόσβασης. Πιο συγκεκριμένα, οι ρυθμοί μετάδοσης στη πέμπτη γενιά κινητών επικοινωνιών θα είναι 10 – 20 Gbps, επομένως 10 με 20 φορές μεγαλύτεροι από τα υπάρχοντα συστήματα τέταρτης γενιάς. Βασικό στοιχείο των κινητών δικτύων είναι το δίκτυο ραδιοπρόσβασης (RAN). Μέσω αυτού γίνεται η σύνδεση των χρηστών στο δίκτυο και επομένως η εξυπηρέτηση τους. Για τη σύνδεση των χρηστών σε ένα κινητό δίκτυο χρησιμοποιούνται οι τεχνικές πολλαπλής πρόσβασης. Αυτές χωρίζονται στις ορθογωνικές (OMA) και τις μη ορθογωνικές (NOMA). Στις τεχνικές OMA οι δέκτες μπορούν να ξεχωρίσουν στο χρόνο ή στη συχνότητα τα λαμβανόμενα σήματα. Ένα παράδειγμα τεχνικής OMA είναι η OFDMA που χρησιμοποιείται στη τέταρτη γενιά κινητών επικοινωνιών. Από την άλλη, στις τεχνικές NOMA οι πληροφορίες των χρηστών αποστέλλονται μέσω των ίδιων πόρων, δηλαδή ταυτόχρονα και στην ίδια συχνότητα. Οι τεχνικές NOMA χωρίζονται σε δύο βασικές κατηγορίες, οι οποίες φαίνονται στο Σχήμα 2.1 [3]. Από τις δύο βασικές κατηγορίες αυτή η εργασία ασχολείται με τις τεχνικές μη ορθογωνικής πρόσβασης στο πεδίο της ισχύος.




Σχήμα 2.1: Κατηγορίες NOMA

Οι τεχνικές μη ορθογωνικής πολλαπλής πρόσβασης μπορούνε να βελτιώσουν τη φασματική απόδοση αυξάνοντας έτσι τη συνδεσιμότητα των χρηστών σε μια κυψέλη. Μπορούν επίσης να αυξήσουν τους ρυθμούς μετάδοσης αλλά και να μειώσουν τη καθυστέρηση (latency) [4]. Οι παραπάνω ισχυρισμοί κάνουν τα συστήματα NOMA ιδανικά για τις μελλοντικές 5G κινητές επικοινωνίες αλλά μπορούν επίσης να βελτιώσουν και τα υπάρχοντα 4G συστήματα. Στόχος αυτής της εργασίας είναι η

προσομοίωση της κάτω ζεύξης (downlink), δηλαδή της επικοινωνίας από το σταθμό βάσης προς τους χρήστες μιας κυψέλης που χρησιμοποιεί τη τεχνική PD-NOMA. Ως κυψέλη ορίζεται μια γεωγραφική περιοχή που καλύπτεται από έναν σταθμό βάσης. Ας θεωρήσουμε μια κυψέλη όπως φαίνεται στο Σχήμα 2.2.

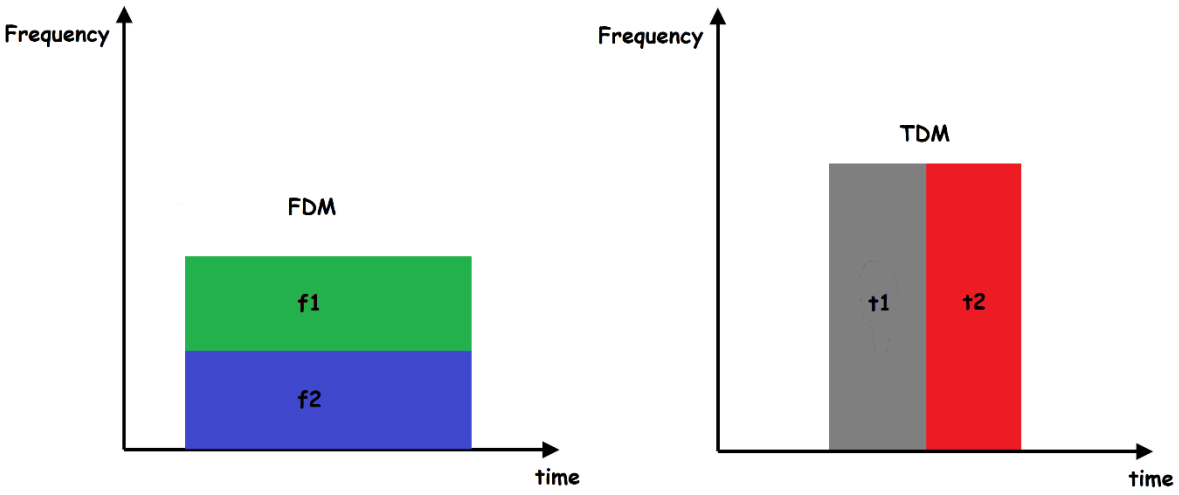



Σχήμα 2.2: Κυψέλη

Για την υλοποίηση της τεχνικής ο σταθμός βάσης πρέπει να χωρίσει τους χρήστες σε ομάδες και να τους αναθέσει ένα ποσοστό ισχύος από το συνολικό που μπορεί να διαθέσει. Στη δική μας περίπτωση οι ομάδες αποτελούνται από ζευγάρια κινητών χρηστών (UE). Η επιλογή των ζευγαριών γίνεται με βάση το κανάλι τους. Ο ένας θέλουμε να έχει καλό κανάλι ενώ ο άλλος όχι. Στο παραπάνω σχήμα παίρνοντας υπόψιν μόνο τις απώλειες από τη διάδοση ενός σήματος μπορούμε να πούμε ότι ο UE1 έχει καλό κανάλι λόγω του ότι είναι κοντά στο σταθμό βάσης, ενώ ο UE2 που βρίσκεται στην άκρη της κυψέλης δεν έχει. Αυτό θα μας βοηθήσει στο να διαχωρίσουμε καλύτερα τους χρήστες στο πεδίο της ισχύος [5]. Ο σταθμός βάσης προσθέτει αλγεβρικά (υπέρθεση) τις πληροφορίες των δύο χρηστών και τις στέλνει ταυτόχρονα και στην ίδια συχνότητα, χρησιμοποιώντας όμως διαφορετική ισχύ για κάθε σήμα [5][6], π.χ. μεγαλύτερη ισχύ για τον UE2 που είναι σε μεγάλη απόσταση για πετύχει ένα ικανοποιητικό λόγο σήματος προς θόρυβο. Ονομάζουμε τον UE1 ως Secondary και τον UE2 ως Primary.

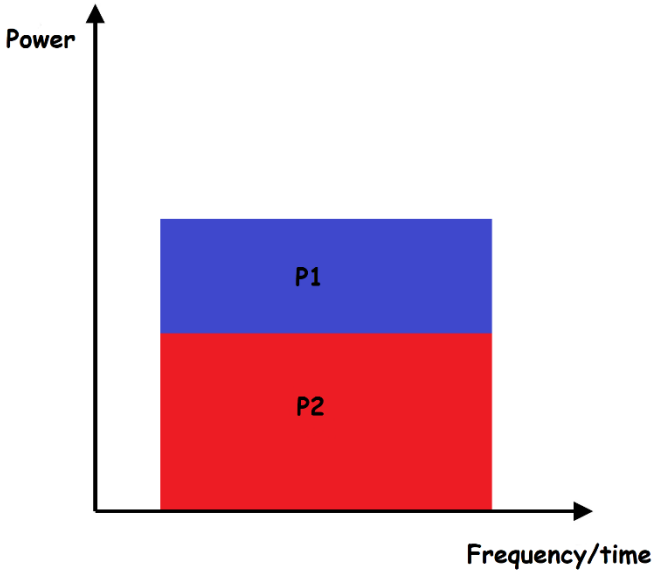
## 2.2 Καταμερισμός ισχύος

Όπως ειπώθηκε, ένα σύστημα NOMA μπορεί να στείλει πληροφορία σε διαφορετικούς χρήστες μέσα από το ίδιο κανάλι (ταυτόχρονα και στην ίδια συχνότητα), χωρίς αυτοί να παρεμβάλουν μεταξύ τους. Στο Σχήμα 2.3 φαίνεται η πολυπλεξία στα συστήματα FDM και TDM. Όπως είναι γνωστό στα συστήματα FDM στέλνουμε την ίδια χρονική στιγμή σε διαφορετικές συχνότητες, ενώ στα συστήματα TDM, στην ίδια συχνότητα αλλά διαφορετικές χρονικές στιγμές.



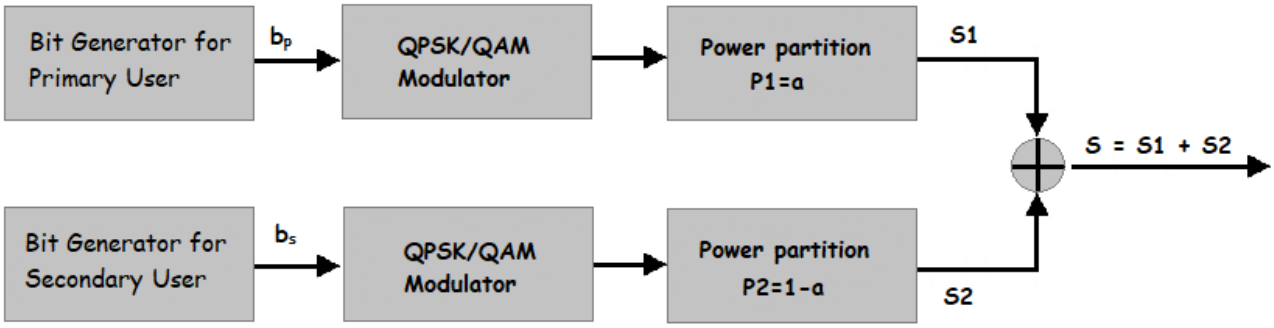

Σχήμα 2.3: Frequency Division Multiplexing vs Time division Multiplexing

Η παραπάνω αναφορά έγινε για να γίνει κατανοητή η λειτουργία του καταμερισμού της ισχύος σε ένα σύστημα PD-NOMA. Σε αυτό ο άξονας των τετμημένων έχει το χρόνο και τη συχνότητα ταυτόχρονα και ο άξονας των τεταγμένων την ισχύ. Στο Σχήμα 2.4 φαίνεται η πολυπλεξία στο επίπεδο της ισχύος [7].




Σχήμα 2.4: Πολυπλεξία στο επίπεδο της ισχύος

Ο καταμερισμός της ισχύος είναι μια διαδικασία που γίνεται στο σταθμό βάσης αφού πρώτα έχει επιλέξει τους χρήστες τους οποίους θα υπερθέσει. Ο χρήστης που έχει το καλύτερο κανάλι παίρνει μια μικρή ποσότητα ισχύος και ο χρήστης με το χειρότερο κανάλι τη μεγάλη. Αυτό που μας ενδιαφέρει είναι να εξυπηρετήσουμε με το καλύτερο δυνατό τρόπο τον Primary χρήστη που έχει το χειρότερο κανάλι και μαζί με αυτόν και τον Secondary. Στο Σχήμα 2.5 φαίνεται ένα απλοποιημένο διάγραμμα ροής της λειτουργίας του σταθμού βάσης [6] [8].




Σχήμα 2.5: Λειτουργία Σταθμού βάσης

Η διαδικασία ξεκινάει με τη δημιουργία των bit πληροφορίας. Τα bit προκύπτουν από τη διαδικασία της δειγματοληψίας, του κβαντισμού και της κωδικοποίησης ενός σήματος πληροφορίας στη βασική ζώνη. Στη προσέγγιση μας όμως δεν μας ενδιαφέρουν αυτές οι διαδικασίες και τα δημιουργούμε αυθαίρετα. Στη συνέχεια ακολουθεί ο διαμορφωτής, μέσα από τον οποίο αποτυπώνουμε τη πληροφορία σε ένα υψίσυχνο φέρον. Στο πείραμα χρησιμοποιήθηκαν ψηφιακές διαμορφώσεις φάσης και πλάτους της μορφής BPSK, QPSK και 16QAM. Αφού ολοκληρωθεί η διαμόρφωση πραγματοποιείται ο καταμερισμός της ισχύος. Αν θεωρήσουμε τη συνολική ισχύ μονάδα, ο καταμερισμός της ισχύος φαίνεται από τις παρακάτω σχέσεις:

$$P1 = a \tag{2.1}$$

$$P2 = 1 - a \tag{2.2}$$

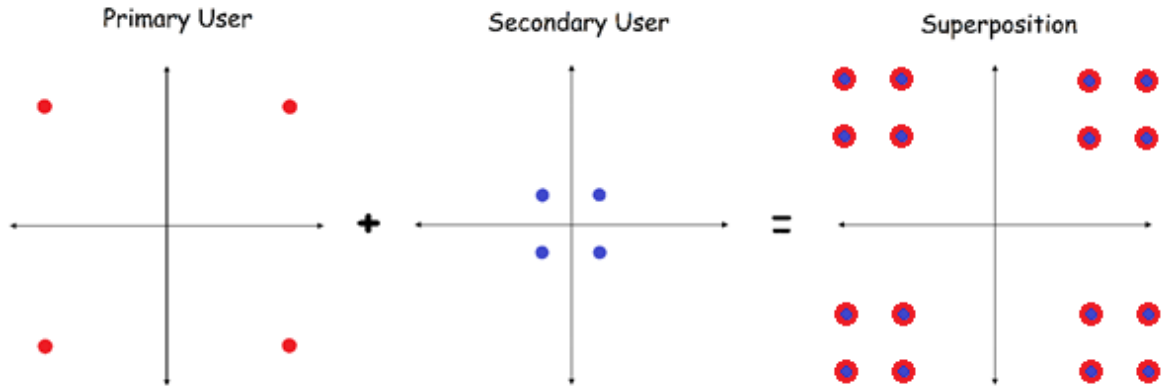
$$P_{TOTAL} = P1 + P2 = 1 \tag{2.3}$$

Η τελική πράξη είναι μια κοινή αλγεβρική πρόσθεση μεταξύ των δύο σημάτων. Σημειώνεται ότι τα σήματα αφού βγούνε από τον διαμορφωτή είναι μιγαδικοί αριθμοί, λόγω της ψηφιακής διαμόρφωσης. Άρα η τελική πρόσθεση γίνεται μεταξύ μιγαδικών αριθμών [6].

Ας υποθέσουμε ότι έχουμε δύο σήματα διαμορφωμένα κατά QPSK. Όπως ειπώθηκε προηγουμένως ξεχωρίζουν μεταξύ τους ως προς την ισχύ. Το ένα έχει το μεγάλο ποσοστό και το δεύτερο ό,τι περισσεύει από τη συνολική ισχύ. Στο καρτεσιανό επίπεδο μπορούμε να αποτυπώσουμε την υπέρθεση όπως φαίνεται στο Σχήμα 2.6.

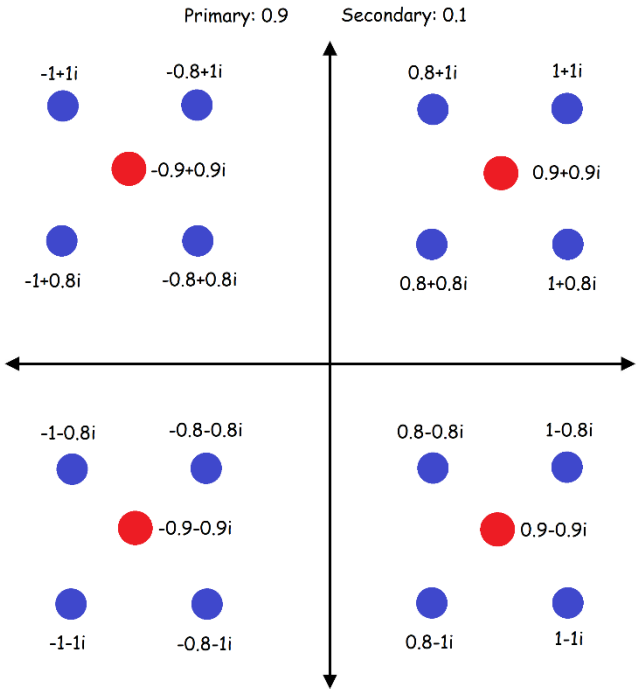
Το QPSK με το κόκκινο χρώμα αφορά τον Primary χρήστη και το μπλε τον Secondary. Αξίζει να σημειωθεί ότι στο παραπάνω σχήμα δεν απεικονίζεται η ισχύς αλλά το πλάτος. Η ισχύς ισούται θεωρητικά με το τετράγωνο του πλάτους. Κάθε σύμβολο του Secondary αποτυπώνεται γύρω από το σύμβολο του Primary. Οι δέκτες λαμβάνουν το υπερτεθειμένο σήμα που φαίνεται στο Σχήμα 2.6 μετά την ισότητα.

Έστω ότι έχουμε αναθέσει 0.9 πλάτος στον Primary και 0.1 στον Secondary, στο Σχήμα 2.7 φαίνεται η αποτύπωση των εκπεμπόμενων συμβόλων στο επίπεδο [9]. Τα εκπεμπόμενα σύμβολα προκύπτουν από τη πρόσθεση του κάθε συμβόλου του Primary με όλα τα σύμβολα του Secondary. Παρατηρείται ότι το τελικό σήμα είναι ένα ιδιόμορφο 16QAM. Επομένως, μπορούμε να πούμε ότι για τη πρόσθεση δύο σημάτων QPSK έχουμε ως αποτέλεσμα ένα ιδιόμορφο 16QAM. Αντίστοιχες παρατηρήσεις γίνονται και για άλλους συνδυασμούς διαμόρφωσης.




Σχήμα 2.6: Η Υπέρθεση στο καρτεσιανό επίπεδο

Τα Σχήματα 2.6 και 2.7 δείχνουν τη βασική ιδέα της τεχνικής PD – NOMA. Φαίνεται πως με αυτή τη τεχνική και δύο σήματα QPSK μπορούμε να πολυπλέξουμε δύο χρήστες χρησιμοποιώντας την διαφορά των σημάτων στην ισχύ. Έτσι μπορούμε να τους εξυπηρετήσουμε ταυτόχρονα και στην ίδια συχνότητα. Για να είναι επιτυχής αυτή η τεχνική πρέπει να σχεδιαστούν δέκτες που να μπορούν να διαχειριστούν το υπερτεθειμένο σήμα και να το αποπολυπλέξουν, με βάση τον καταμερισμό της ισχύος. Στην επόμενη παράγραφο θα παρουσιαστεί η χρησιμότητα του καταμερισμού για τον σχεδιασμό των δεκτών και ειδικότερα του Secondary.




Σχήμα 2.7: Αποτέλεσμα υπέρθεσης με πλάτη 0.9-0.1

## 2.3 Σχεδιασμός δέκτη - Ακύρωση της παρεμβολής

Έχοντας υπόψιν ό,τι ειπώθηκε παραπάνω μπορούμε να ξεκινήσουμε να σχεδιάζουμε τη λειτουργία των δεκτών. Όσον αφορά τον Primary, η διαδικασία είναι απλή. Θεωρεί τον Secondary ως παρεμβολή και συνεχίζει με την αποδιαμόρφωση του υπερτεθειμένου σήματος. Αυτό συμβαίνει διότι έχει τη μεγάλη ισχύ, άρα και περιθώριο στη σηματοθορυβική του σχέση (SNR) [6][7]. Στους δέκτες έχουμε σχέση σήματος προς θόρυβο συν τη παρεμβολή (SINR). Για να γίνει κατανοητό παραθέτουμε τις παρακάτω σχέσεις:

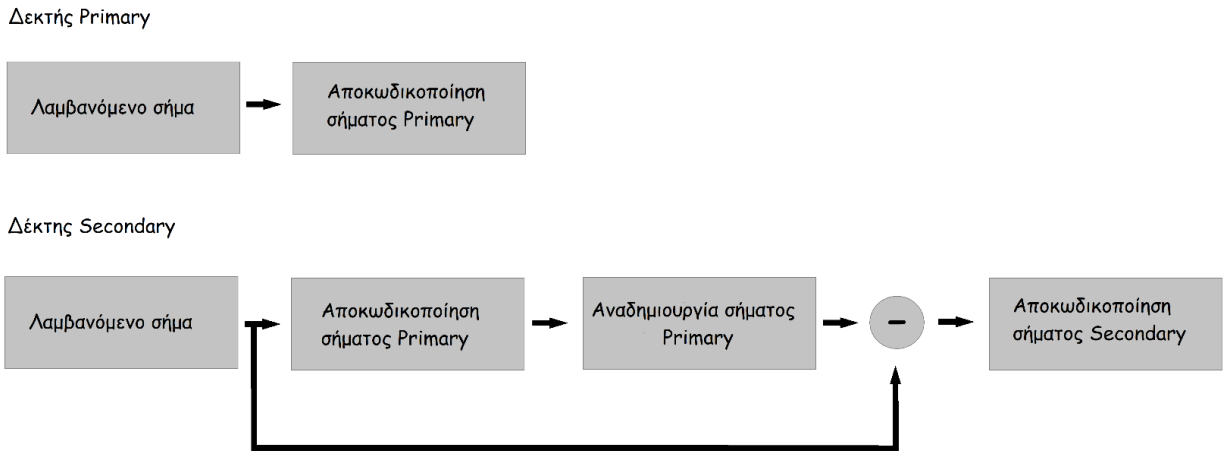
$$SINR_P = \frac{P_P}{P_S + P_N} \tag{2.4}$$

Για τον Secondary χρήστη, η σηματοθορυβική σχέση είναι αντίστοιχα:

$$SINR_S = \frac{P_S}{P_P + P_N} \tag{2.5}$$

όπου, $P_P$ η ισχύς του Primary, $P_S$ η ισχύς του Secondary και $P_N$ η ισχύς του θορύβου.

Από την άλλη, ο Secondary για να ανακτήσει τη πληροφορία του πρέπει να διεξάγει τη διαδικασία της ακύρωσης της παρεμβολής (IC) [6]. Η διαδικασία αποτελείται από την εύρεση του σήματος του Primary και την αφαίρεση του από το συνολικό λαμβανόμενο. Στο παρακάτω διάγραμμα ροής φαίνεται η λειτουργία των δεκτών Primary και Secondary [10].



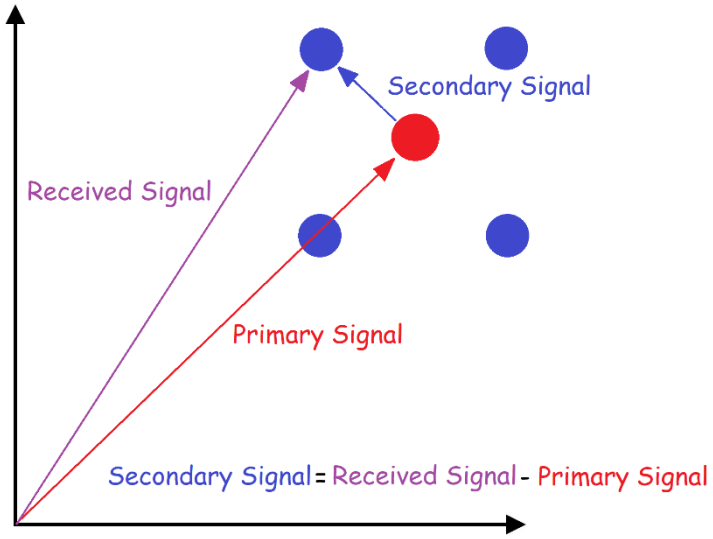

Σχήμα 2.8: Διαδικασία λήψης [10]

Φαίνεται από το Σχήμα 2.8 ότι ο Secondary πρέπει πρώτα να αποκωδικοποιήσει το λαμβανόμενο σήμα όπως ο Primary, στη συνέχεια να το ξαναδημιουργήσει, και τέλος να το αφαιρέσει από το συνολικό λαμβανόμενο. Με αυτή τη διαδικασία ο Secondary ακυρώνει τη παρεμβολή του Primary και μπορεί πλέον να προχωρήσει στην αποκωδικοποίηση του δικού του σήματος. Για να γίνει κατανοητό το IC παρουσιάζονται στο Σχήμα 2.9 τα διανύσματα των πλατών στο επίπεδο [11]. Αν υποθέσουμε ότι ο σταθμός βάσης στέλνει το μωβ σήμα και ότι ο θόρυβος είναι αμελητέος, τότε ο Primary το μόνο που έχει να κάνει είναι να αποκωδικοποιήσει το αρχικό λαμβανόμενο σήμα. Όπως φαίνεται και από το Σχήμα 2.9, ο δέκτης του Primary δεν θα έχει πρόβλημα να ανιχνεύσει το σωστό (κόκκινο) σύμβολο πληροφορίας μιας και τα δύο είναι στο ίδιο τεταρτημόριο. Από την άλλη, ο Secondary πρέπει να αφαιρέσει από το συνολικό σήμα το σήμα του Primary, έτσι ώστε να απομείνει από την αφαίρεση το σήμα που τον ενδιαφέρει. Η εξίσωση 2.6 εκφράζει την υπέρθεση των δύο σημάτων στο σταθμό βάσης.

$$X(n) = \sqrt{P_1}S_P(n) + \sqrt{P_2}S_S(n) \quad (2.6)$$

Όπου $P_1$ και $P_2$ οι ισχύες που περιγράφηκαν από τις σχέσεις 2.1 και 2.2, $S_P$ και $S_S$ τα σήματα πληροφορίας των χρηστών και n το μήκος των ακολουθιών bit που εκφράζουν τα σήματα. Οι δέκτες των κινητών χρηστών λαμβάνουν το σήμα Y(n) . Ο δέκτης του Primary το ανιχνεύει χωρίς περαιτέρω διαδικασίες. Η ακύρωση της παρεμβολή εκφράζεται με τη μαθηματική σχέση 2.7. Όπου $h_S$ το κανάλι του Secondary χρήστη.

$$Y_S(n) = Y(n) - \sqrt{P_1}\, S_P(n) \cdot h_S \quad (2.7)$$

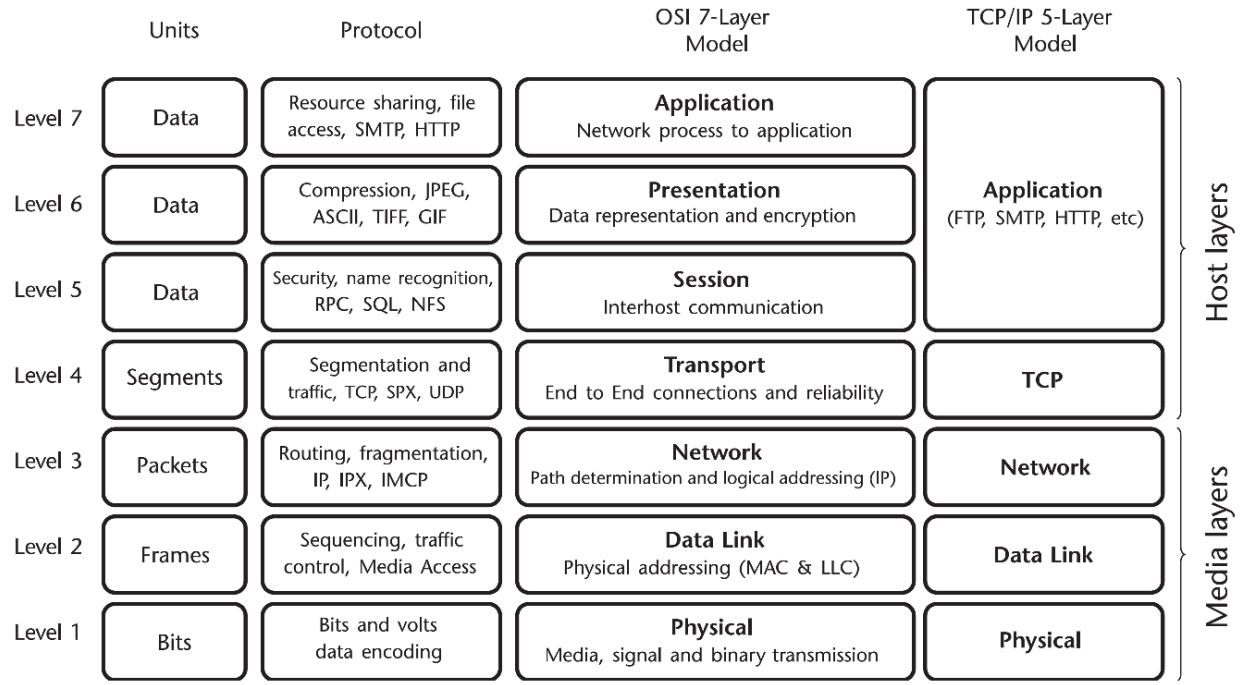



Σχήμα 2.9: Ακύρωση παρεμβολής [11]

Το αποτέλεσμα της ακύρωσης της παρεμβολής είναι να εξαλειφθεί το κόκκινο σήμα και ο Secondary να καταλήξει με την πληροφορία που τον αφορά. Τότε ο δέκτης του Secondary μπορεί να συνεχίσει με την αποκωδικοποίηση [6][7][11]. Οι διαδικασίες που αναλύθηκαν στις παραγράφους 2.2 και 2.3 θα παρουσιαστούν με λεπτομέρεια στο τρίτο κεφάλαιο όπου θα παρατεθούν και κομμάτια από τους κώδικες προσομοίωσης. Στην επόμενη παράγραφο θα αναλυθεί η λειτουργία του WLAN συστήματος πάνω στο οποίο κατασκευάστηκε η τεχνική PD-NOMA, αλλά και ο λόγος που επιλέχθηκε. Με τα παραπάνω ολοκληρώνεται η βασική θεωρία πίσω από την υπέρθεση δύο σημάτων στο πεδίο της ισχύος, περισσότερα θα αναλυθούν σε επόμενα κεφάλαια όπου θα παρουσιαστεί ο τρόπος που μπορεί να υλοποιηθεί ένα τέτοιο σύστημα.

## 2.4 WLAN PD-NOMA

Στις σύγχρονες ψηφιακές επικοινωνίες όπου το υλικό των συσκευών είναι πλέον πολύπλοκο από άποψη αρχιτεκτονικής, είναι χρήσιμο να ομαδοποιούμε τις διάφορες τηλεπικοινωνιακές λειτουργίες έτσι ώστε να είναι πιο εύκολες στη κατανόηση, στο σχεδιασμό και στην υλοποίηση. Έτσι ο σχεδιασμός των συστημάτων γίνεται σε επίπεδα. Στο Σχήμα 2.10 φαίνεται ο διαχωρισμός των επιπέδων τα οποία φέρουν εις πέρας διαφορετικές λειτουργίες και όλα μαζί ολοκληρώνουν ένα τηλεπικοινωνιακό σύστημα. Το μοντέλο αυτό ονομάζεται OSI και αποτελείται από τα παρακάτω βασικά επίπεδα.

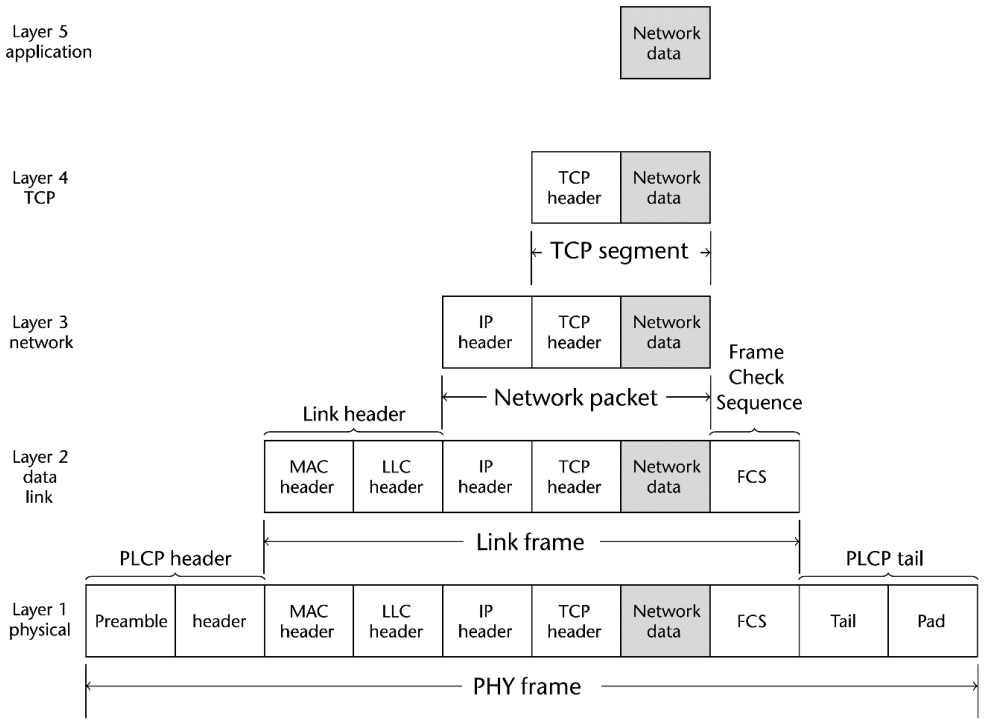



Σχήμα 2.10: Μοντέλο αναφοράς OSI [12]

- Επίπεδο εφαρμογών: Διασυνδέει τον χρήστη με τη πληροφορία του τηλεπικοινωνιακού συστήματος.
- Επίπεδο μεταφοράς: Είναι υπεύθυνο για τη μεταφορά μηνυμάτων μεταξύ του χρήστη και του διακομιστή. Εξασφαλίζει την αξιόπιστη μεταφορά δεδομένων.
- Επίπεδο δικτύου: Είναι υπεύθυνο για την μεταφορά πακέτων πληροφορίας στα σημεία του δικτύου.
- Επίπεδο ζεύξης δεδομένων: Διαχειρίζεται προβλήματα στην ανταλλαγή πληροφορίας μεταξύ συνδεδεμένων συσκευών.
- Φυσικό επίπεδο: Διαχειρίζεται τα bit πληροφορίας και τα αποστέλλει μεταξύ συνδεδεμένων συσκευών.

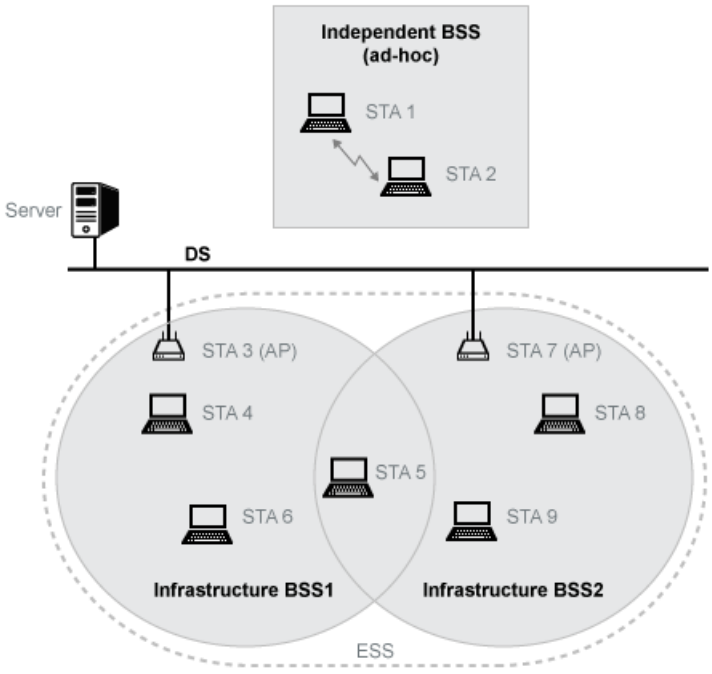
Από επίπεδα του Σχήματος 2.10 το πρότυπο WLAN (Wireless Local Area Network) λειτουργεί στο φυσικό επίπεδο και στο επίπεδο ζεύξης δεδομένων. Στο σχήμα 2.11 φαίνεται η πληροφορία την οποία διαχειρίζεται το κάθε επίπεδο. Παρατηρούμε ότι το φυσικό επίπεδο περιέχει το ολοκληρωμένο πακέτο εκπομπής με την πληροφορία που θέλουμε να μεταδώσουμε όπως επίσης και όλα τα απαραίτητα στοιχεία για την ορθή εκπομπή και λήψη ενός σήματος. Στο τρίτο κεφάλαιο θα δούμε πως κατασκευάζεται στη πράξη το πακέτο αλλά και που χρησιμεύουν τα επιμέρους στοιχεία του [12].

Η διαδικασία του πειράματος προσεγγίστηκε αρχικά με ένα πολύ απλό σύστημα εκπομπής και λήψης που στόχο είχε την υλοποίηση της τεχνικής PD-NOMA δημιουργώντας ένα σήμα και αποστέλλοντας το μέσα από το κανάλι, χωρίς παραπάνω διαδικασίες, όπως για παράδειγμα της κωδικοποίησης καναλιού στο πομπό ή της ισοστάθμισης (equalization) στο δέκτη. Το σύστημα αυτό όμως παρόλο που μπορούσε να αποστείλει και να λάβει ένα σήμα δε μπορούσε να διαχειριστεί σωστά την ακύρωση της παρεμβολής διότι δε μπορούσε να πετύχει μια καλή σηματοθορυβική σχέση. Έτσι, ο δέκτης του Secondary δεν μπορούσε να πετύχει μια επιθυμητή πιθανότητα σφάλματος bit. Για την εξάλειψη του παραπάνω προβλήματος χρησιμοποιήθηκε τελικά η εργαλειοθήκη προγραμματισμού της MATLAB “WLAN System toolbox”.




Σχήμα 2.11: Πακέτο πληροφορίας μοντέλου OSI [12]

Το WLAN είναι ένα σύστημα ασύρματης επικοινωνίας που υλοποιείται με βάση το πρωτόκολλο 802.11 της IEEE. Χρησιμοποιείται κατά κόρο στα Wi-Fi συστήματα. Ως τοπικό δίκτυο χρησιμοποιείται για την επικοινωνία υπολογιστών μέσα σε ένα χώρο όπως μια εταιρεία ή ένα σχολείο. Η αρχιτεκτονική του φαίνεται στο παρακάτω σχήμα [13].



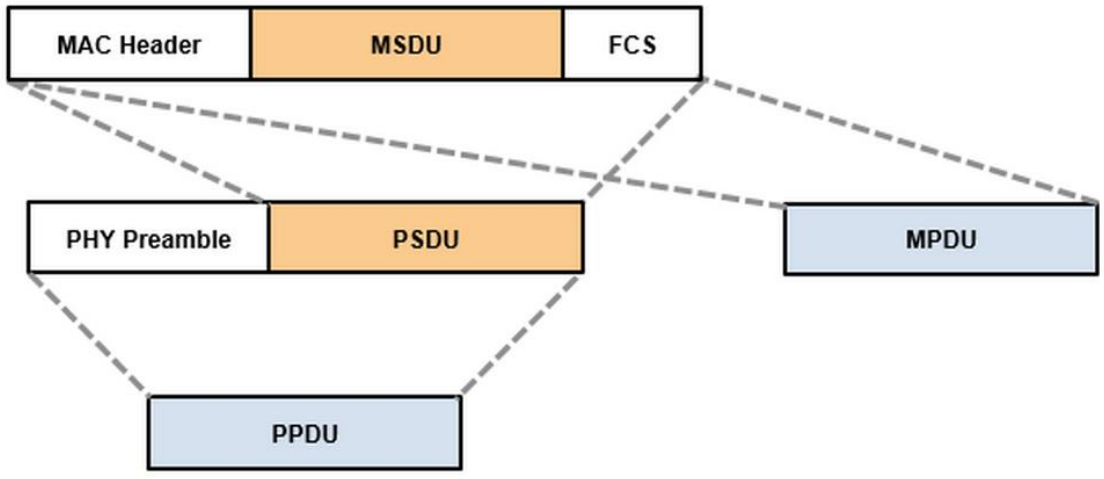

Σχήμα 2.12: Αρχιτεκτονική WLAN [13]

Τα βασικά του στοιχεία είναι οι σταθμοί (STA) και τα σημεία πρόσβασης (AP). Αυτά μαζί δημιουργούν ένα βασικό σετ εξυπηρέτησης (BSS). Κάθε σημείο πρόσβασης εξυπηρετεί ένα BSS. Ένα δίκτυο από περισσότερα BSS αποκαλείται εκτεταμένο σετ εξυπηρέτησης (ESS). Δύο BSS επικοινωνούν μεταξύ τους και με τον διακομιστή (server) μέσω των συστημάτων διανομής (DS) [13][14].

Η εκπομπή και η λήψη της πληροφορίας σε ένα σύστημα WLAN γίνεται μέσω του φυσικού επιπέδου (physical layer – PHY). Το φυσικό επίπεδο είναι το χαμηλότερο επίπεδο σε ένα σύστημα τηλεπικοινωνιών. Είναι υπεύθυνο για τη μετάδοση των bit πληροφορίας μέσω του μέσου διάδοσης. Στη δική μας περίπτωση το μέσο είναι ο αέρας. Η τεχνική αυτή οργανώνει τα δεδομένα σε πακέτα. Το κάθε πακέτο αποτελείται από τα στοιχεία που φαίνονται στο Πίνακα 2.1, στο Σχήμα 2.13 φαίνεται η δομή του [13].

Πίνακας 2.1: Στοιχεία πακέτου WLAN [13]

| Μήνυμα | Λεπτομέρειες |
|---|---|
| MSDU-MAC service data unit | Μηνύματα που μεταφέρουν πληροφορία μεταξύ του LLC επιπέδου και του MAC επιπέδου εντός ενός STA |
| MPDU-MAC protocol data unit | Μηνύματα που μεταφέρουν πληροφορία στο MAC επίπεδο των STAs |
| PSDU-PLCP service data unit | Μηνύματα που μεταφέρουν πληροφορία μεταξύ του MAC επιπέδου και του PHY επιπέδου, εντός ενός STA |
| PPDU-PLCP protocol data unit | Μηνύματα που μεταφέρουν πληροφορία στο PHY επίπεδο των STAs |



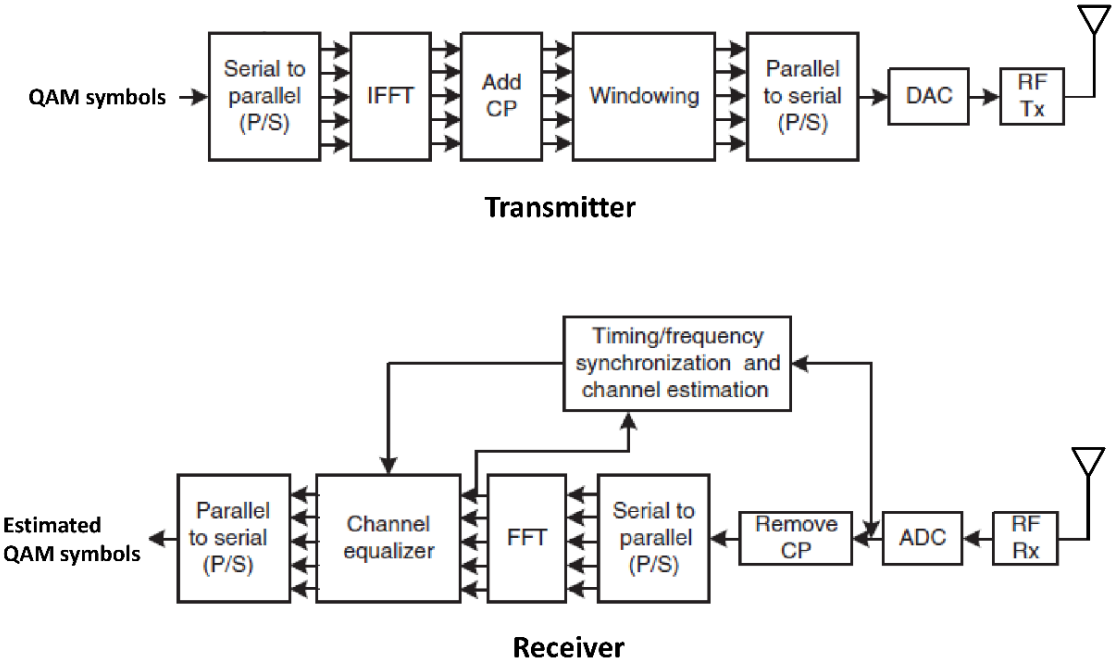

Σχήμα 2.13: Δομή πακέτου WLAN [13]

Για να δημιουργήσουμε ένα σύστημα WLAN PD-NOMA πρέπει να δημιουργήσουμε ένα πακέτο πληροφορίας PPDU, στο οποίο η πληροφορία του PSDU θα αποτελείται από δύο υπερτεθειμένους στην ισχύ χρήστες. Έτσι με τους ίδιους πόρους ενός συστήματος μπορούμε να εξυπηρετήσουμε περισσότερους χρήστες.

Τα παραπάνω ειπώθηκαν για να γίνει κατανοητή η λειτουργία του WLAN, παρόλα αυτά πρέπει να υπενθυμίσουμε ότι η τεχνική PD-NOMA μελετάται για να χρησιμοποιηθεί σε συστήματα κινητών επικοινωνιών. Αυτό δε σημαίνει όμως ότι δε μπορεί να ενσωματωθεί και σε άλλα δίκτυα όπως είναι το WLAN. Ο λόγος που επιλέχθηκε η εργαλειοθήκη WLAN της MATLAB είναι προγραμματιστικός. Μέσω αυτής επιτεύχθηκε ένα καλό SNR στους δέκτες και ειδικότερα στον δέκτη του Secondary, ο οποίος λόγω του ότι λαμβάνει μικρό ποσοστό ισχύος πρέπει να έχει καλή σηματοθορυβική σχέση. Στην επόμενη παράγραφο θα συζητηθεί το σύστημα OFDM και πως αυτό μπορεί να συνδυαστεί με τις τεχνικές NOMA.

## 2.5 OFDM και PD-NOMA

Ας υποθέσουμε ότι έχουμε μια ροή δεδομένων με ρυθμό πληροφορίας $R$ $(bps)$ και διαθέσιμο εύρος ζώνης $N \cdot f_b$ με κεντρική συχνότητα $f_0$. Χωρίζουμε τη ροή αυτή με έναν μετατροπέα σειριακού σε παράλληλο σε $N$ υπορροές. Κάθε υπορροή μεταδίδεται από μια υποφέρουσα του χωρισμένου φάσματος $N \cdot f_b$ και έχει ρυθμό πληροφορίας $R/N$ $(bps)$. Δύο γειτονικές υποφέρουσες έχουν απόσταση $f_b$ μεταξύ τους [15]. Στο Σχήμα 2.14 φαίνεται γραφικά ο πομπός και ο δέκτης ενός συστήματος OFDM.




Σχήμα 2.14: Πομποδέκτης OFDM [15]

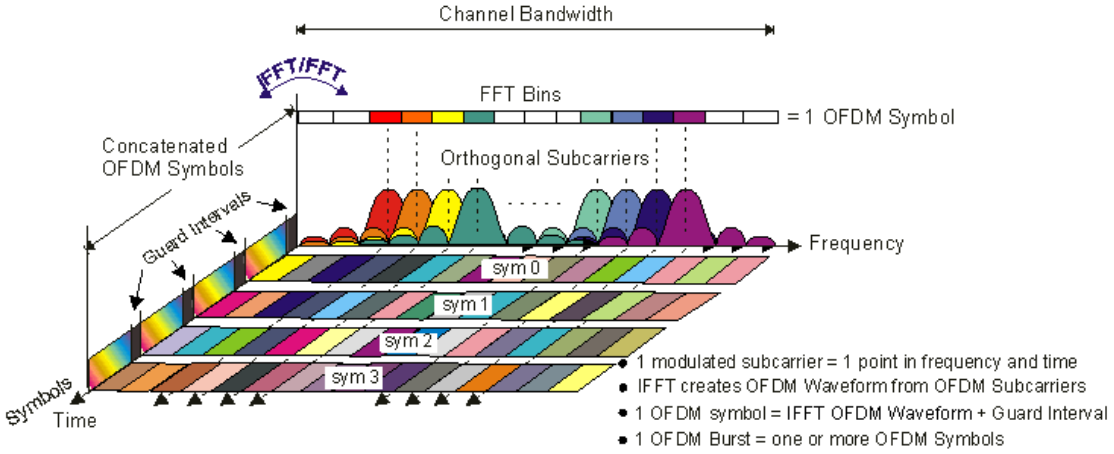
Στη διαδικασία του OFDM ένα από τα πιο σημαντικά στοιχεία είναι ο μετασχηματισμός Fourier (FFT). Είναι στην ουσία αυτός που αναθέτει τα σύμβολα πληροφορίας στα υποφέροντα και δημιουργεί το λεγόμενο «OFDM σύμβολο». Το «OFDM σύμβολο» δεν πρέπει να συγχέεται με το σύμβολο πληροφορίας. Πάνω σε κάθε υποφέρον διαμορφώνεται ένα σύμβολο πληροφορίας. Το σύνολο των υποφερόντων ονομάζεται «OFDM σύμβολο».

Βασικό χαρακτηριστικό του OFDM είναι η ορθογωνικότητα μεταξύ των υποφερουσών. Δύο σήματα $S_1(t)$ και $S_2(t)$ για να είναι ορθογώνια μεταξύ τους σε ένα διάστημα T πρέπει να ισχύει η παρακάτω σχέση:

$$\int_0^T S_1(t) \cdot S_2(t)dt = 0 \tag{2.8}$$

Το πλεονέκτημα της ορθογωνικότητας είναι ότι οι υποφέρουσες συχνότητες τοποθετούνται τόσο κοντά η μία στην άλλη ώστε να επικαλύπτονται, αλλά χωρίς να υπάρχουν παρεμβολές μεταξύ τους [15]. Στο Σχήμα 2.15 φαίνονται με διαφορετικούς χρωματισμούς τα ορθογώνια υποφέροντα (orthogonal subcarriers) που σχηματίζουν το «OFDM σύμβολο». Φαίνεται επίσης και η επαναχρησιμοποίηση τους στις διαφορετικές χρονικές στιγμές (time slots) οι οποίες χωρίζονται μεταξύ τους με τα κενά διαφύλαξης (Guard Intervals) [15][16].

Η τεχνική OFDM μπορεί να συνδυαστεί με τη τεχνική PD-NOMA αν σκεφτούμε ότι σε κάθε υποφέρον αντί για ένα διαμορφωμένο σύμβολο πληροφορίας μπορούμε να έχουμε δύο, υπερτεθειμένα. Αμέσως άμα έχουμε 64 υποφέροντα που μέχρι τώρα εξυπηρετούσαν 64 χρήστες, τώρα μπορούμε να εξυπηρετήσουμε τους διπλάσιους. Στην επόμενη παράγραφο θα αναλυθεί θεωρητικά το ασύρματο κανάλι και η πιθανότητα σφάλματος του bit πληροφορίας.




Σχήμα 2.15: Λειτουργία OFDM [16]

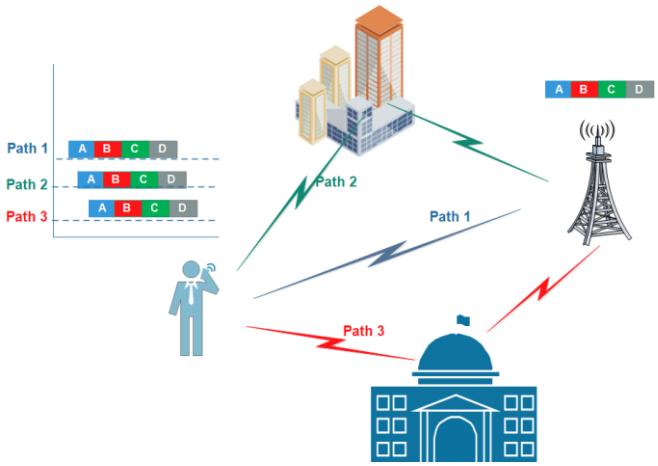
## 2.6 Κανάλι και πιθανότητα σφάλματος

Κανάλι ή μέσο μετάδοσης είναι η φυσική οδός μεταξύ πομπού και δέκτη. Μεταξύ σταθμού βάσης και κινητών χρηστών εν προκειμένω. Τα μέσα μετάδοσης χωρίζονται σε οδηγούμενα και μη οδηγούμενα. Οδηγούμενα ονομάζονται αυτά στα οποία το σήμα οδηγείται κατά μήκος ενός στερεού μέσου, όπως ένα χάλκινο καλώδιο ή μια οπτική ίνα [15][17].

Το σύστημα που υλοποιήθηκε είναι πραγματικό, η μονάδα ραδιοεπικοινωνίας που λειτουργεί ως πομπός οδηγεί το σήμα στη κεραία και από εκεί μέσω του αέρα διαδίδεται στο χώρο. Ο δέκτης λαμβάνει το σήμα με τη δική του κεραία και το επεξεργάζεται. Τα πειράματα διενεργήθηκαν τόσο σε κλειστό όσο και σε ανοιχτό χώρο. Ο πομπός και ο δέκτης τοποθετήθηκαν σε διάφορες αποστάσεις μεταξύ τους. Επίσης δοκιμάστηκαν οι συνθήκες οπτικής επαφής και μη. Οι παραπάνω διαδικασίες δημιουργούν πολύπλοκα κανάλια με απώλειες διάδοσης, διαλείψεις και παρεμβολές. Οι διαλείψεις είναι μεταβολές της λαμβανόμενης ισχύος του σήματος στο χρόνο που οφείλονται σε αλλαγές του περιβάλλοντος γύρω από τη ζεύξη [15]. Σε μια ζεύξη κινητής τηλεφωνίας ο κινητός χρήστης μπορεί να μετακινείται εντός μιας κυψέλης και γύρω του υπάρχουν εμπόδια τα οποία επίσης κινούνται. Σε αυτή τη περίπτωση έχουμε ένα κανάλι με γρήγορες διαλείψεις. Οι γρήγορες διαλείψεις προκαλούνται από τη πολυδιαδρομική διάδοση ενός σήματος λόγω των φαινομένων της σκέδασης της διάθλασης και της

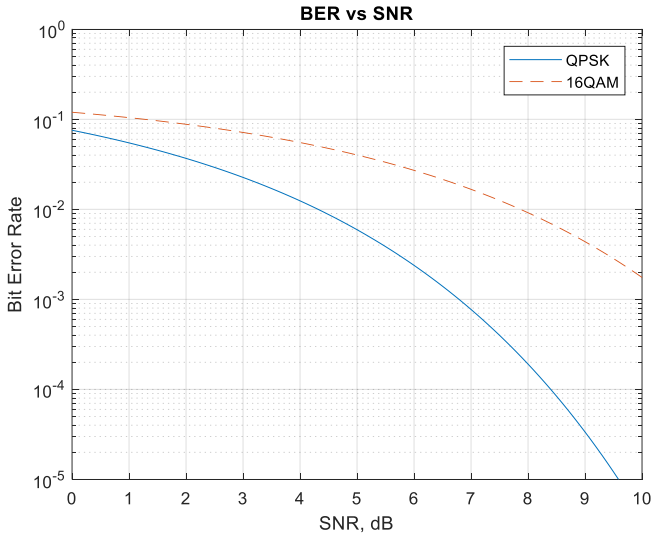
περίθλασης ενός σήματος. Έτσι η ισχύς του σήματος φτάνει στο δέκτη διασκορπισμένη στο χρόνο. Ως αποτέλεσμα στο δέκτη εμφανίζεται το φαινόμενο της διασυμβολικής παρεμβολής (ISI).

Η διασυμβολική παρεμβολή δημιουργεί αληλοκαλύψεις μεταξύ των συμβόλων που ως αποτέλεσμα έχουν την αύξηση της πιθανότητας σφάλματος στο δέκτη. Για να γίνει κατανοητό το φαινόμενο των γρήγορων διαλείψεων μπορούμε να δούμε το Σχήμα 2.16 όπου φαίνονται οι πολλές διαδρομές του σήματος, οι οποίες προκαλούν τη διασπορά της ισχύος στο χρόνο [17].




Σχήμα 2.16: Πολυδιαδρομική διάδοση [17]

Στο σύστημα που υλοποιήθηκε υπάρχουν διαλείψεις, απώλειες διάδοσης, παρεμβολές από άλλες γειτονικές εκπομπές, όπως για παράδειγμα μιας κοντινής συσκευής Wi-Fi ή ενός κινητού τηλεφώνου. Επίσης υπάρχει και ο θερμικός θόρυβος από τα RF στοιχεία του δέκτη. Το αποτέλεσμα είναι ο δέκτης να κάνει λάθη στα λαμβανόμενα bit. Για την εξαγωγή αποτελεσμάτων και συμπερασμάτων μετρήθηκε ο ρυθμός των σφαλμάτων, σε σχέση με τη σηματοθορυβική σχέση στο δέκτη, για διάφορες περιπτώσεις. Πιο συγκεκριμένα λήφθηκαν αποτελέσματα για διάφορες αποστάσεις μεταξύ πομπού και δέκτη, διαφορετική κατανομή ισχύος και διαφορετική ισχύς εκπομπής. Τα αποτελέσματα παρουσιάζονται στο τέταρτο κεφάλαιο. Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα υπολογισμού του ρυθμού σφαλμάτων σε σχέση με το SNR στο δέκτη για διαμορφώσεις QPSK και 16QAM.




Σχήμα 2.17: Θεωρητικός ρυθμός σφαλμάτων για διαμόρφωση QPSK και 16QAM

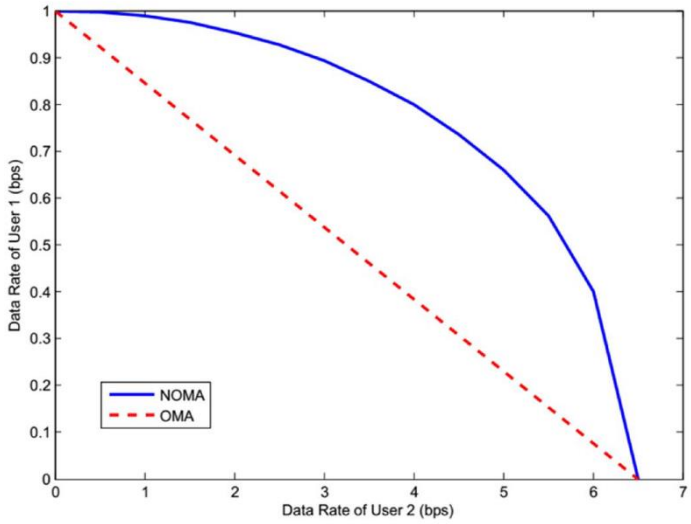
## 2.7 Ρυθμός μετάδοσης δεδομένων

Ο ρυθμός μετάδοσης των δεδομένων είναι ένας από τους βασικότερους παράγοντες λειτουργίας ενός τηλεπικοινωνιακού συστήματος. Σε ένα σύστημα PD – NOMA ο Secondary χρήστης μπορεί να ακυρώσει τη παρεμβολή (IC) του Primary, λόγω του κέρδους του καναλιού του είναι μεγαλύτερο. Στον Primary το σήμα αποδιαμορφώνεται απ' ευθείας χωρίς να διενεργηθεί IC. Αυτή η βασική υπόθεση διαμορφώνει και τους ρυθμούς μετάδοσης των δεδομένων. Στις εξισώσεις 2.10 και 2.11 εκφράζονται οι ρυθμοί ενός συστήματος PD – NOMA [3].

$$R_P = log_2 \left(1 + \frac{P_P|h_1|^2}{P_S|h_1|^2 + \sigma_n^2}\right) \tag{2.10}$$

$$R_S = log_2 \left(1 + \frac{P_S|h_S|^2}{\sigma_n^2}\right) \tag{2.11}$$

όπου $R_P$ και $R_S$ οι ρυθμοί δεδομένων Primary και Secondary αντίστοιχα, h οι συντελεστές των καναλιών και $P_P$ και $P_S$ οι ισχύες των χρηστών. Φαίνεται ότι ο ρυθμός των δεδομένων σε ένα σύστημα PD – NOMA συσχετίζεται με τη κατανομή της ισχύος. Αποδεικνύεται από την εργασία [3] ότι ένα σύστημα μη ορθογωνικής πολλαπλής πρόσβασης στο πεδίο της ισχύος μπορεί να πετύχει μεγαλύτερους ρυθμούς μετάδοσης. Τα αποτελέσματα της εργασίας [3] φαίνονται στο Σχήμα 2.18.




Σχήμα 2.18: Σύγκριση ρυθμών μετάδοσης συστημάτων OMA και NOMA [3]

## 2.8 Επίλογος δεύτερου κεφαλαίου

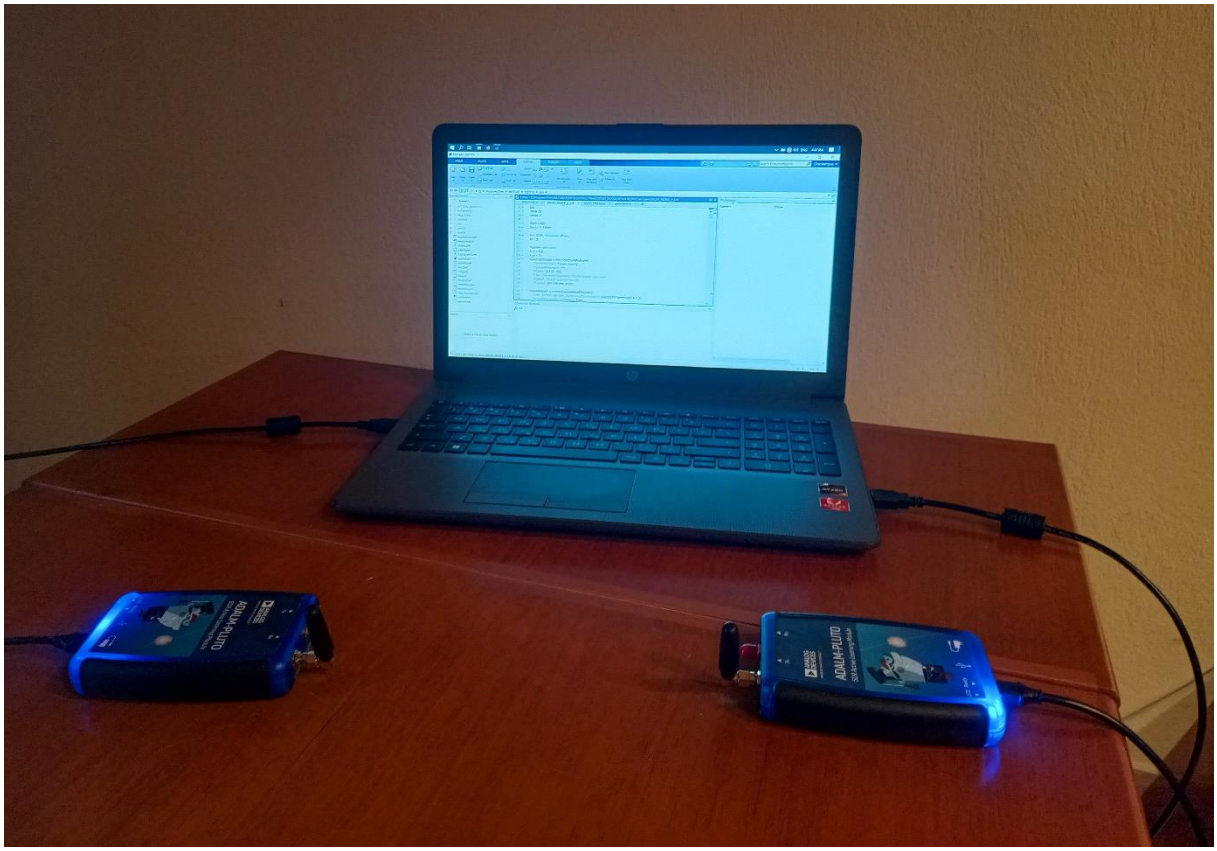
Το δεύτερο κεφάλαιο επικεντρώνεται στην ανάλυση της θεωρίας πίσω από το PD-NOMA, το WLAN και το OFDM που είναι βασικά στοιχεία του πειράματος και χρειάζονται για τη κατανόηση των κεφαλαίων που ακολουθούν. Θεωρούμε ότι έχουμε ένα σταθμό βάσης κινητής τηλεφωνίας που εξυπηρετεί μια γεωγραφική περιοχή (κυψέλη). Ο σταθμός βάσης επιλέγει δύο χρήστες. Ο πρώτος χρήστης ονομάζεται Primary και βρίσκεται στην άκρη της κυψέλης, πράγμα που συνεπάγεται ότι έχει κακό κανάλι. Ο δεύτερος χρήστης, που ονομάζεται Secondary, βρίσκεται κοντά στο σταθμό βάσης και έχει το καλύτερο δυνατό κανάλι. Λόγω της διαφοράς τους στη ποιότητα του καναλιού, ο σταθμός βάσης αναθέτει στο σήμα του Primary ένα μεγάλο ποσοστό ισχύος και ένα μικρότερο ποσοστό στον

Secondary. Ο σταθμός βάσης υπερθέτει τα δύο σήματα και τα αποστέλλει μέσω του ίδιου καναλιού (χρόνος – συχνότητα). Από τη μεριά των δεκτών των χρηστών, ο Primary αποκωδικοποιεί το σήμα του παραβλέποντας τη παρεμβολή που προκαλείται από το σήμα του Secondary. Από την άλλη, ο Secondary για να βρει το σήμα του, πρέπει να ακυρώσει τη παρεμβολή από τον Primary (Interference Cancellation). Η διαδικασία αυτή που περιγράφεται στη Παράγραφο 2.3 αποτελείται από τρείς τις βασικές διαδικασίες που φαίνονται στο Σχήμα 2.8.

Στη συνέχεια στις Παραγράφους 2.4 και 2.5 παρουσιάστηκαν τα συστήματα WLAN και OFDM και πώς αυτά συμπεριλήφθηκαν μέσα στην εργασία. Στη παράγραφο 2.6 αναλύθηκαν οι όροι «κανάλι» και «πιθανότητα σφάλματος» και δόθηκαν παραδείγματα για το πώς επηρεάζουν μια ασύρματη ζεύξη. Τέλος παρουσιάστηκαν οι σχέσεις που διέπουν τους ρυθμούς μετάδοσης των δεδομένων και το πώς σχετίζονται με τη κατανομή της ισχύος. Στο επόμενο κεφάλαιο γίνεται μια παρουσίαση του προγραμματισμού των SDR και παρατίθενται οι κώδικες που υλοποιήθηκαν για να επιτευχθεί τη τεχνική PD - NOMA.

# Κεφάλαιο 3ο: Εξομοίωση συστήματος

Το κεφάλαιο αυτό αφιερώνεται στο προγραμματισμό του συστήματος. Για την υλοποίηση χρησιμοποιήθηκε το λογισμικό της MATLAB και οι εργαλειοθήκες “WLAN System toolbox” και “Communications System Toolbox”. Οι εργαλειοθήκες αυτές περιέχουν συναρτήσεις για την επεξεργασία σημάτων προς αποστολή αλλά και τις αντίστοιχες για τη λήψη. Η τεχνική PD-NOMA προσεγγίστηκε με δύο διαφορετικές μεθόδους, η μια χρησιμοποιεί το πρότυπο WLAN.




Σχήμα 3.1: Σύστημα ζεύξης με Adalm Pluto SDR

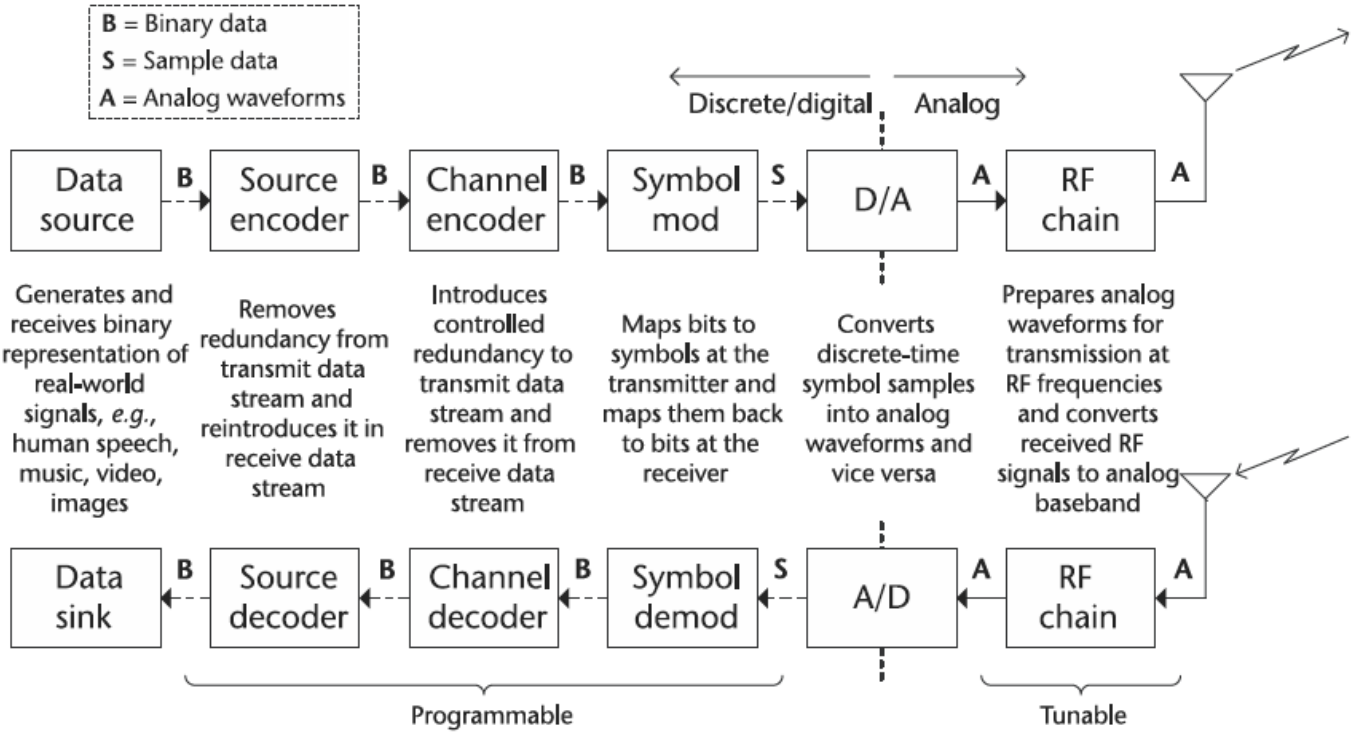
Στην παραπάνω φωτογραφία φαίνεται το σύστημα ζεύξης των δύο Adalm Pluto. Τα SDR συνδέονται με τον υπολογιστή στον οποίο γίνεται ο προγραμματισμός τους. Στις επόμενες παραγράφους παρουσιάζεται λεπτομερώς η μονάδα ραδιοεπικοινωνίας και στη συνέχεια η υλοποίηση των τεχνικών που συζητήθηκαν στα προηγούμενα κεφάλαια.

## 3.1 Adalm Pluto SDR

Σε αυτή τη παράγραφο γίνεται μια ανάλυση της λειτουργίας του SDR και των στοιχείων από τα οποία αποτελείται. Θα παρουσιαστεί με λεπτομέρεια το hardware θα αναλυθούν τα επιμέρους στοιχεία του και το πώς αυτά αλληλοεπιδρούν με το λογισμικό της MATLAB. Το Adalm Pluto SDR αποτελείται από το σύστημα επεξεργασίας “Xilinx Zynq” το οποίο βασίζεται στον μικροελεγκτή “ARM Cortex” και τον πομποδέκτη “AD9363”.

Πριν όμως εμβαθύνουμε στη λειτουργία του Adalm Pluto, πρέπει να εξηγήσουμε γενικά τη λειτουργία ενός SDR. Στο Σχήμα 3.2 φαίνεται γραφικά γενική λειτουργία ενός συστήματος ραδιοεπικοινωνίας. Αυτό που κάνει τη μονάδα ραδιοεπικοινωνίας καθορισμένη από λογισμικό χρήσιμη, είναι ότι μπορούμε να επέμβουμε προγραμματιστικά πάνω στα στοιχεία της, να δημιουργήσουμε

σήματα, να υλοποιήσουμε τεχνικές εκπομπής και λήψης ώστε να μπορέσουμε να εκπέμψουμε αυτά τα σήματα μέσω του καναλιού.

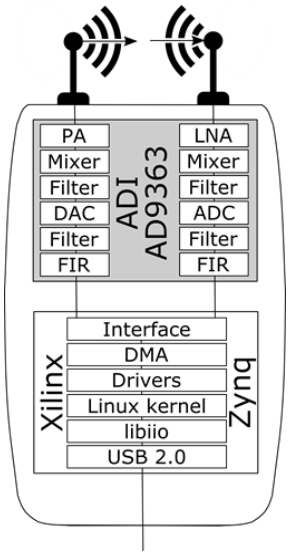



Σχήμα 3.2: Διάγραμμα ροής μονάδας ραδιοεπικοινωνίας [12]

Το παραπάνω σχήμα χωρίζεται σε δύο μέρη, το ένα είναι ο πομπός και το δεύτερο ο δέκτης. Παρατηρούμε ότι ο δέκτης κάνει την αντίστροφή διαδικασία από αυτή που κάνει ο πομπός. Αυτό συμβαίνει με όλα τα τηλεπικοινωνιακά συστήματα για την ανάκτηση του εκπεμπόμενου σήματος. Τα προγραμματιζόμενα κομμάτια (“programmable” στο σχήμα) του πομπού είναι αυτά που μας δίνουν την δυνατότητα να δημιουργήσουμε το σήμα που μας ενδιαφέρει. Από εκεί μπορούμε να επιλέξουμε το σχήμα διαμόρφωσης και κωδικοποίησης. Αυτές οι διαδικασίες γίνονται σε επίπεδο bit. Στη συνέχεια ο μετατροπέας ψηφιακού σε αναλογικό δημιουργεί την αναλογική κυματομορφή που πρόκειται να εκπεμφθεί. Σε αυτή τη διαδικασία δεν μπορούμε να επέμβουμε, η πληροφορία με την οποία θα τροφοδοτήσουμε τον D/A θα περάσει μέσα από τα RF στοιχεία και θα εκπεμφθεί στον αέρα. Μπορούμε όμως να επέμβουμε και να ρυθμίσουμε κατάλληλα τα RF στοιχεία έτσι ώστε να έχουμε τα κατάλληλα χαρακτηριστικά εκπομπής. Δηλαδή, μπορούμε να ρυθμίσουμε τη συχνότητα εκπομπής το εύρος ζώνης και την ισχύ εκπομπής. Από τη μεριά του δέκτη ρυθμίζουμε σωστά τα RF στοιχεία έτσι ώστε να συμπίπτουν με αυτά του πομπού και να έχουμε μια σωστή ζεύξη. Αφού ρυθμιστούν και συγχρονιστούν ο πομπός και ο δέκτης μπορεί να ξεκινήσει η αντίστροφη διαδικασία εύρεσης της πληροφορίας από τη μεριά του δέκτη. Εκεί ο δέκτης γνωρίζοντας το σχήμα διαμόρφωσης και κωδικοποίησης βρίσκει τα λαμβανόμενα σύμβολα πληροφορίας τα αποδιαμορφώνει και καταλήγει με τα bit πληροφορίας [12].

Στη συνέχεια αυτής της παραγράφου θα παρουσιαστούν τα βασικά κομμάτια του Adalm Pluto SDR. Αυτά χωρίζονται σε τρείς βασικούς τομείς:

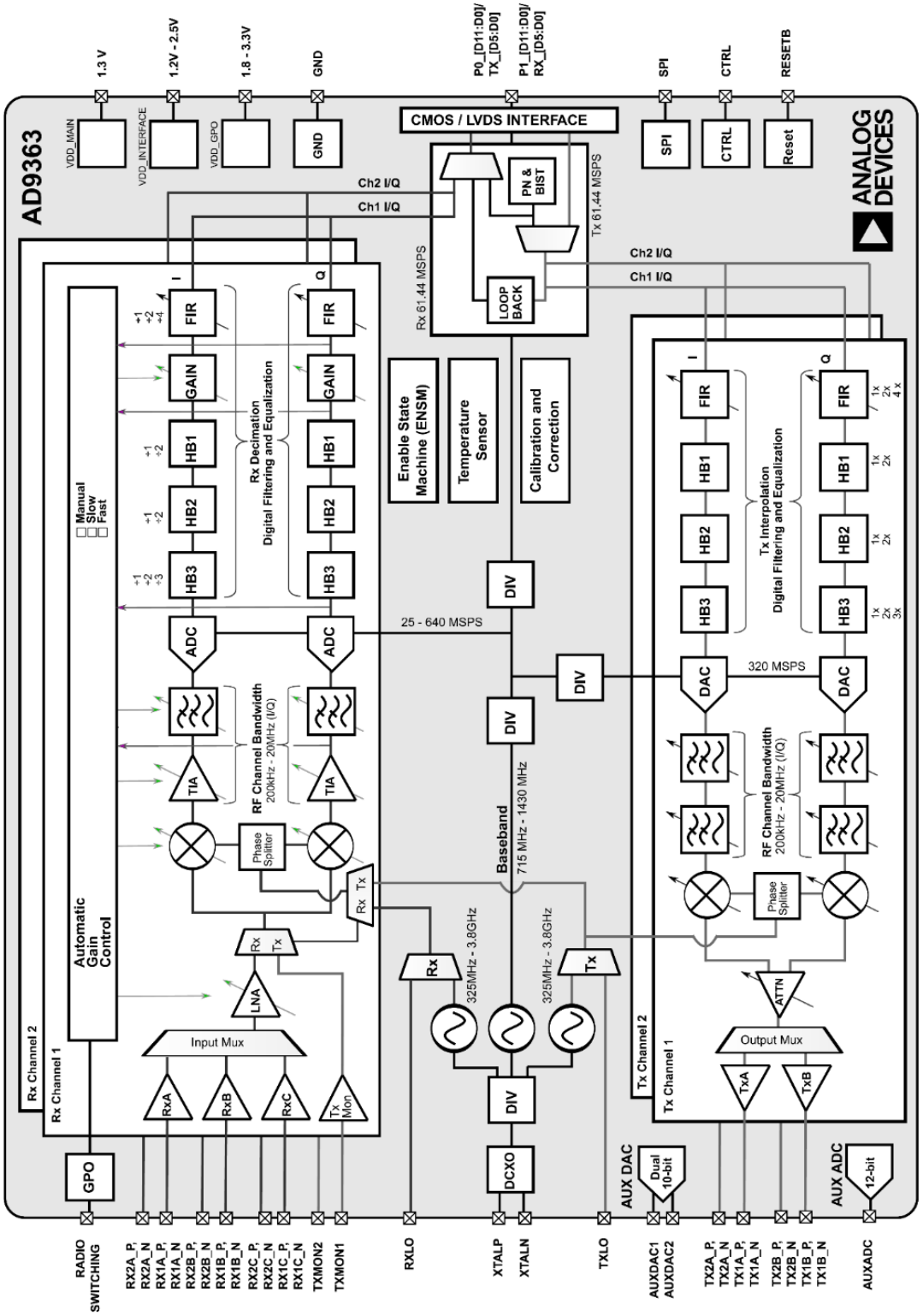
- Τομέας αναλογικών ραδιοσυχνοτήτων: Αποτελείται από τη κεραία τα RF φίλτρα, τους πολυπλέκτες και τους ενισχυτές χαμηλού θορύβου. Αυτά τα στοιχεία βρίσκονται εντός του πομποδέκτη AD9363.
- Τομέας αναλογικής επεξεργασίας βασικής ζώνης: Σε αυτό το τομέα βρίσκονται τα αναλογικά φίλτρα, όπως επίσης και οι μετατροπείς από ψηφιακό σε αναλογικό και αντίστροφα.
- Επεξεργασία σήματος: Κάποιες από τις λειτουργίες της επεξεργασίας σήματος γίνονται εντός του AD9363, ενώ άλλες γίνονται στο σύστημα επεξεργασίας του μικροελεγκτή ARM. Τα σήματα περνάνε από το USB που συνδέει το SDR με τον προσωπικό υπολογιστή για περαιτέρω επεξεργασία στο λογισμικό της MATLAB [12] [19].



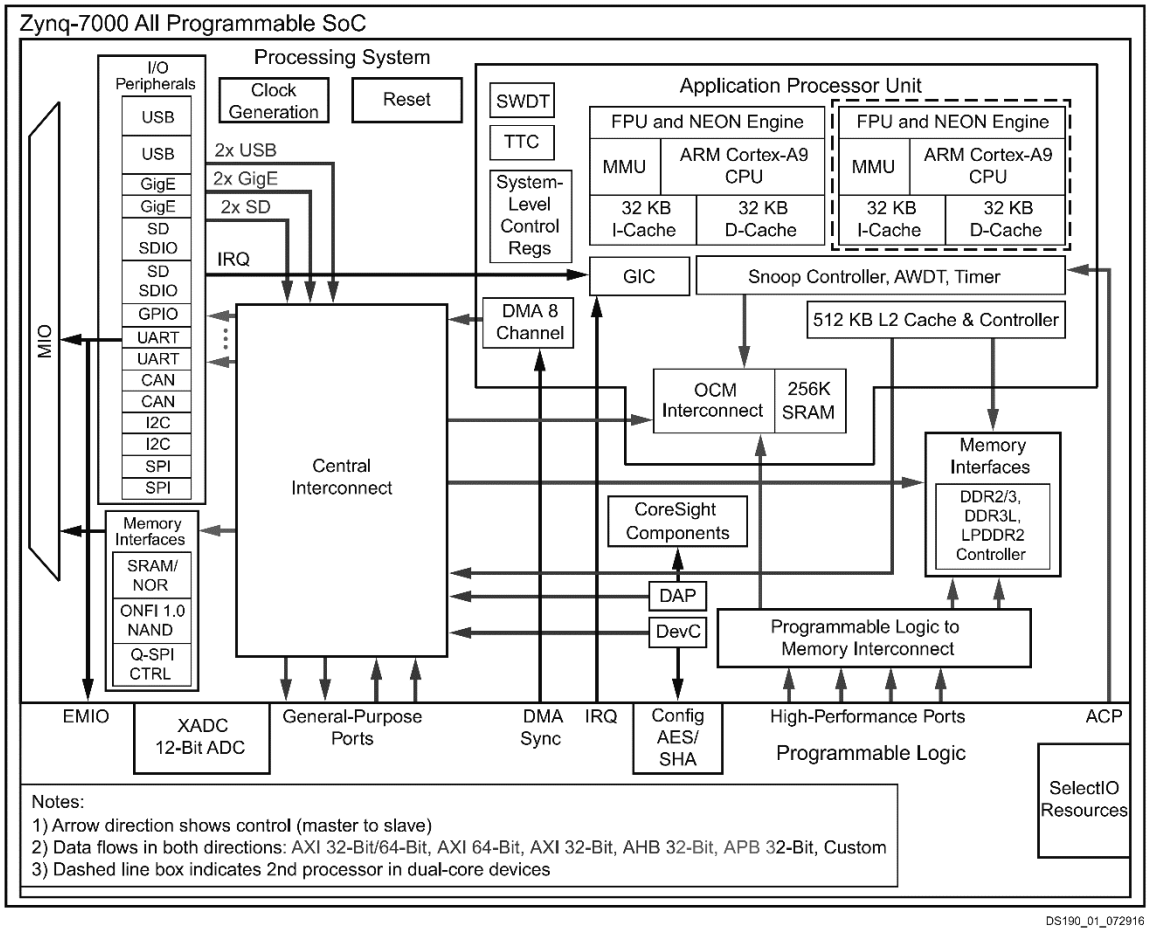

Σχήμα 3.3: Adalm Pluto SDR [12]

Στο Σχήμα 3.3 φαίνεται γενικά το εσωτερικό του Adalm Pluto SDR, το σύστημα επεξεργασίας όπως επίσης και ο πομποδέκτης AD9363. Στο πάνω μέρος του σχήματος βρίσκεται ο πομποδέκτης AD9363 της εταιρείας Analog Devices όπου είναι υπεύθυνος για τη λήψη των σημάτων. Βασική του λειτουργία είναι η ενίσχυση και το φιλτράρισμα των σημάτων προς εκπομπή και λήψη. Στην αριστερή πλευρά του AD9363 του Σχήματος 3.3 βρίσκεται ο πομπός και στο δεξιά μέρος ο δέκτης. Στη συνέχεια θα αναλυθούν τα στοιχεία του δέκτη με σκοπό τη κατανόηση της λειτουργίας του. Οι αντίστροφες διαδικασίες από αυτές που θα εξηγηθούν γίνονται στο πομπό. Αρχικά στο πάνω μέρος βρίσκεται ο ενισχυτής χαμηλού θορύβου (LNA) ο οποίος παρέχει ενίσχυση του λαμβανόμενου σήματος με τέτοιο τρόπο ώστε να κρατάει σε χαμηλά επίπεδα το σήμα του θορύβου. Αμέσως μετά βρίσκεται ο μείκτης (mixer). Ο μείκτης στις σύγχρονες ψηφιακές επικοινωνίες χρησιμοποιείται για τη διαμόρφωση του φέροντος προς εκπομπή σήματος [12][20]. Στη περίπτωση του Adalm Pluto SDR, ο μείκτης λειτουργεί από τα 325MHz έως τα 3.8GHz, αυτές είναι φέρουσες συχνότητες που μπορούμε να χρησιμοποιήσουμε για την εκπομπή και λήψη σημάτων. Μετά το μείκτη το σήμα φιλτράρεται έτσι ώστε να αφαιρεθούν παρεμβολές και παραποιήσεις στο σήμα. Στόχος είναι η μείωση των επιπέδων του θορύβου και των παρεμβολών ώστε να αυξηθεί η σηματοθορυβική σχέση. Στη συνέχεια βρίσκεται ο μετατροπέας από αναλογικό σε ψηφιακό (ADC). Πρόκειται για έναν ADC που προσφέρει ~4.5 bit ανάλυση. Αυτού του είδους οι μετατροπείς προσφέρουν αρκετά χαμηλά επίπεδα θορύβου. Στο Σχήμα 3.4 φαίνεται ολοκληρωμένο το εσωτερικό του AD9363. Αφού το σήμα βγει ψηφιοποιημένο από τον ADC περνάει

στον μικροεπεξεργαστή “Xilinx Zynq”. Το σύστημα επεξεργασίας βασίζεται στον ARM ελεγκτή και λειτουργεί με λογισμικό “Linux”. Το λογισμικό Linux λειτουργεί σε συνδυασμό με το σύστημα IIO. Πρόκειται για drivers μέσω των οποίων μπορούμε στο λογισμικό της MATLAB να ρυθμίσουμε παραμέτρους της λειτουργίας των SDR, όπως το εύρος ζώνης, τη φέρουσα συχνότητα και τις ρυθμίσεις των φίλτρων. Στο Σχήμα 3.5 φαίνεται το εσωτερικό του μικροεπεξεργαστή “Zynq” [21][22].




Σχήμα 3.4: Πομποδέκτης AD9363 [12]

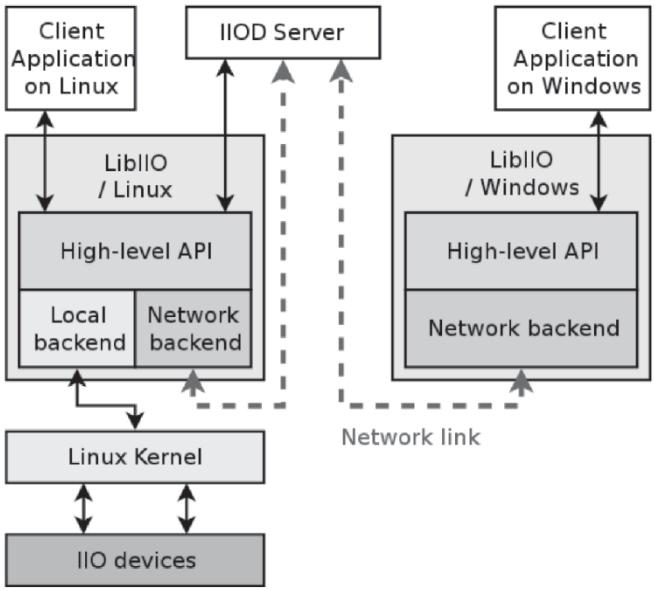



Σχήμα 3.5: Μπλοκ διάγραμμα μικροεπεξεργαστή [12] [21]

Εδώ πρέπει να σημειωθεί πως εμείς δεν ασχοληθήκαμε καθόλου με τα δύο παραπάνω διαγράμματα για τον προγραμματισμό τους SDR. Αυτό είναι και το πλεονέκτημα που δίνει το σύστημα Industrial Input Output (IIO) που αναφέρθηκε παραπάνω [22]. Μέσω αυτού δε χρειάζεται να επέμβουμε στο υλικό του πομποδέκτη και του μικροεπεξεργαστή. Το σύστημα IIO που λειτουργεί παράλληλα με τη MATLAB μας δίνει αυτή τη δυνατότητα. Είναι σχεδιασμένο να υποστηρίζει συστήματα ADC και DAC και δεν δημιουργήθηκε συγκεκριμένα για το Adalm Pluto SDR ή για κάποιο άλλο SDR. Μπορεί να υποστηρίξει διαφορετικά συστήματα παρέχοντας ένα API για τον έλεγχο των παραμέτρων τους. Συμπεριλαμβάνοντας ADCs, επιταχυνσιόμετρα, γυροσκόπια, αισθητήρες πίεσης, φωτός και θερμοκρασίας. Τα βασικά λειτουργικά μέρη του IIO είναι τρεις και φαίνονται παρακάτω [12]:

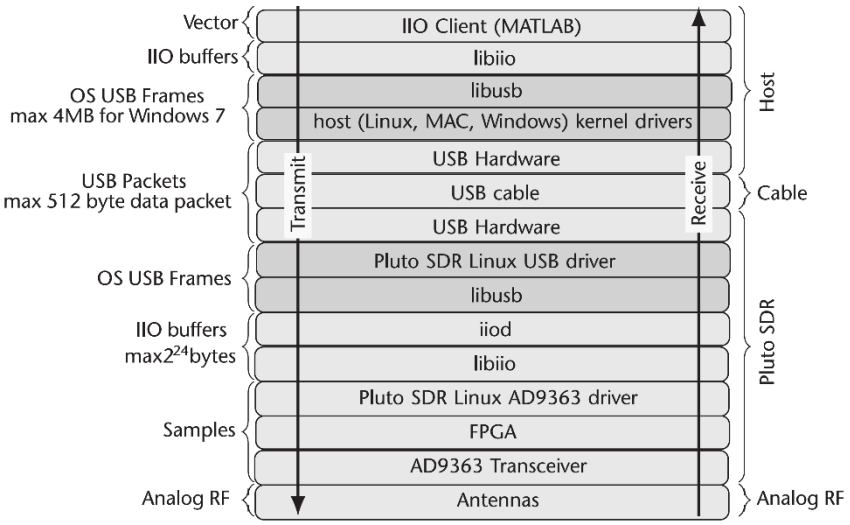
- Ο driver Linux Kernel IIO, που λειτουργεί εντός του ARM του SDR.
- Η βιβλιοθήκη Libiio που είναι υπεύθυνη για την επικοινωνία του SDR και του προσωπικού υπολογιστή [22].
- Το εργαλείο IIO Deamon, το οποίο είναι υπεύθυνο για την σύνδεση του SDR και βρίσκεται εγκατεστημένο στον ARM.

Στο Σχήμα 3.6 φαίνεται το μπλοκ διάγραμμα του Libiio και του IIO Deamon [22]. Το σύστημα μπορεί να λειτουργήσει σε συστήματα Windows, Linux και MAC. Για τη λειτουργία σε Windows είναι απαραίτητο το εργαλείο IIO Deamon [12] [22].




Σχήμα 3.6: Μπλοκ διάγραμμα λειτουργίας Libiio και IIO Deamon [12] [22]

Εν κατακλείδι, η συνολική λειτουργία του Adalm Pluto SDR περιγράφεται από το Σχήμα 3.7. Πολύ γενικά φαίνονται όλες οι διαδικασίες που εξηγήθηκαν παραπάνω για το για το πώς ένα σήμα φτάνει από το λογισμικό που προγραμματίζουμε, στη προκειμένη περίπτωση είναι η MATLAB, μέχρι τη κεραία. Η αντίστροφη διαδικασία ακολουθείται από το δέκτη.



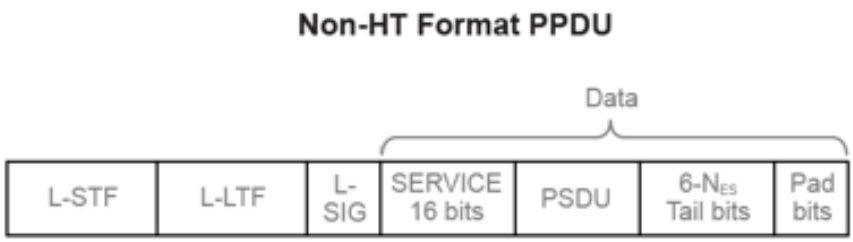

Σχήμα 3.7: Συνολική λειτουργία Adalm Pluto SDR [12]

## 3.2 Συγχρονισμός πομπού-δέκτη

Η λήψη ενός σήματος είναι μια πολύπλοκη λειτουργία που απαιτεί κάποιες απαραίτητες διαδικασίες για την ορθή επανάκτηση του. Αυτές οι διαδικασίες είναι οι παρακάτω:

- Συγχρονισμός στο χρόνο
- Συγχρονισμός στη συχνότητα – φάση

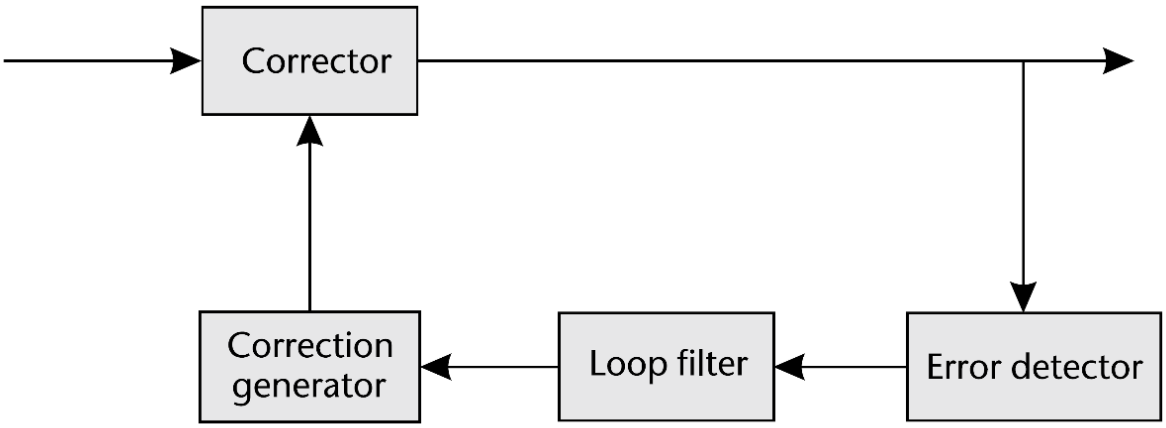
Οι συχνότητες που παράγουν οι ταλαντωτές των συσκευών ραδιοεπικοινωνίας δεν είναι ποτέ ακριβώς ίδιες, ακόμα και αν έχουν παραχθεί από το ίδιο εργοστάσιο και με τα ίδια χαρακτηριστικά. Έτσι απαιτείται ρύθμιση σε μια τηλεπικοινωνιακή ζεύξη. Στα WLAN συστήματα ο συγχρονισμός επιτυγχάνεται με τη χρήση των προοιμίων (preamble) που εισάγονται πάντα στην αρχή ενός εκπεμπόμενου σήματος [12][23]. Το προοίμιο αποτελείται από τα τρία πρώτα στοιχεία που φαίνονται στο Σχήμα 3.8. Για το συγχρονισμό στο χρόνο χρησιμοποιείται το πεδίο L-LTF (Legacy – Long Training Field) και για τη διαδικασία του συγχρονισμού στη συχνότητα το πεδίο L-STF (Legacy- Short Training Field). Στη συνέχεια ακολουθεί το PSDU που περιγράφηκε στο Κεφάλαιο 2. Το πρότυπο WLAN έχει πολλές δυνατότητες όσον αφορά το ρυθμό μετάδοσης της πληροφορίας και τη διαμόρφωση. Για το πείραμα επιλέχθηκε το πακέτο 802.11a με τη διάταξη Non - HT (Non - High Throughput) [12].




Σχήμα 3.8: Non HT WLAN PPDU [12]

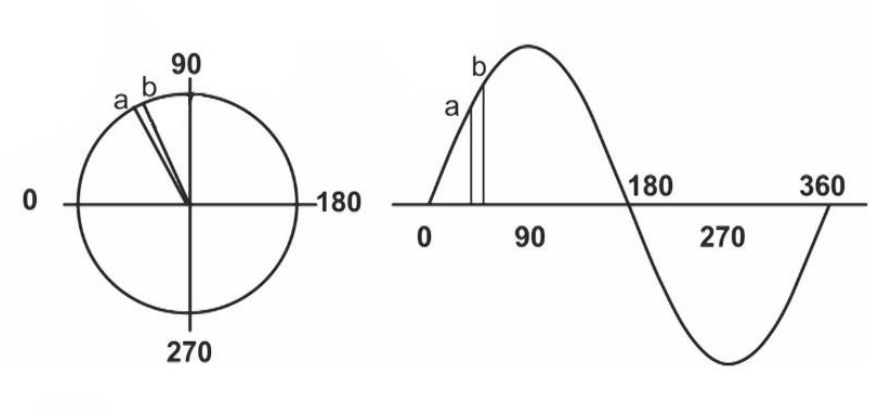
Γενικά στις τηλεπικοινωνίες ο συγχρονισμός γίνεται με PLL (Phase locked loop). Είναι συστήματα ελέγχου στα οποία η φάση εισόδου σχετίζεται με τη φάση εξόδου. Η συχνότητα και η φάση είναι έννοιες συνδεδεμένες. Ένα PLL μπορεί να διατηρήσει τη συχνότητα εξόδου ίδια ή πολλαπλάσια με τη συχνότητα εισόδου. Στο Σχήμα 3.9 φαίνεται γραφικά η λειτουργία του.

Ο βρόγχος ανάδρασης διορθώνει τη συχνότητα του εισερχόμενου σήματος με βάση τον υπολογισμό του σφάλματος και παράγει μια συχνότητα έτσι ώστε να συμπίπτουν ο πομπός με το δέκτη. Όταν υπάρχει διαφορά φάσης αναγκαστικά υπάρχει και διαφορά συχνότητας. Για αυτό το λόγο ένας βρόγχος διόρθωσης της φάσης διορθώνει και τη συχνότητα.




Σχήμα 3.9: Μπλοκ διάγραμμα PLL [12]

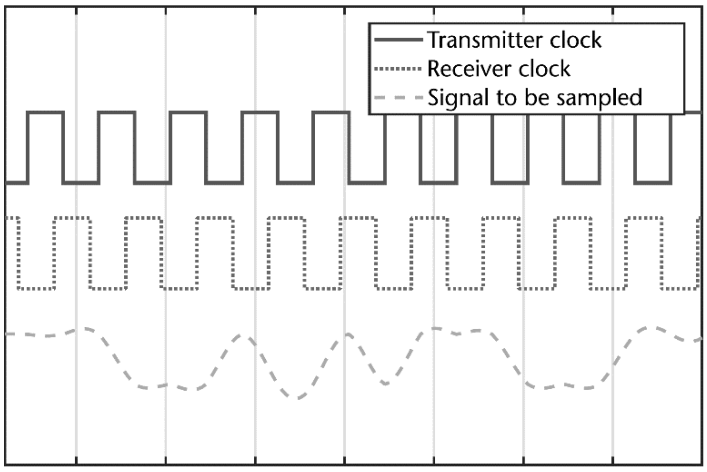
Ένα περιοδικό σήμα μπορεί να αποτυπωθεί ως ημίτονο και ως φάσορας στο καρτεσιανό επίπεδο. Το κλειδί στη λειτουργία των PLL είναι ότι όταν η διαφορά φάσης δεν αλλάζει, οι συχνότητες είναι ίδιες. Στο Σχήμα 3.10 φαίνεται γραφικά η διαφορά της φάσης ενός σήματος. Η διαδικασία συγχρονισμού στο χρόνο είναι παρόμοια με αυτήν του συγχρονισμού φάσης – συχνότητας. Ο συγχρονισμός πομπού και δέκτη κατά τη πρώτη προσέγγιση του πειράματος έγινε με τη χρήση PLL χρησιμοποιώντας της κατάλληλες συναρτήσεις του "Communication toolbox" της MATLAB. Στη συνέχεια, στις επόμενες δύο παραγράφους θα αναλυθούν με περισσότερη λεπτομέρεια οι λειτουργίες του συγχρονισμού.




Σχήμα 3.10: Διαφορά φάσης

### 3.2.1 Συγχρονισμός στο χρόνο

Στόχος με τη διαδικασία του συγχρονισμού στο χρόνο είναι η διόρθωση του χρονικού σφάλματος που προκαλείται από τα διαφορετικά ρολόγια του πομπού και του δέκτη. Κατά την αποστολή ενός σήματος από έναν πομπό σε ένα δέκτη προστίθεται στο σήμα μια καθυστέρηση $\tau$, η οποία είναι μικρότερη από το πλήρη κύκλο λειτουργίας του ρολογιού. Στο Σχήμα 3.11 φαίνεται η χρονική καθυστέρηση που εισάγεται στο δέκτη. Αποτέλεσμα αυτής είναι η λανθασμένη αποδιαμόρφωση ενός σήματος. Η χρονική καθυστέρηση περιγράφεται από τη σχέση 3.1.




Σχήμα 3.11: Χρονική καθυστέρηση [12]

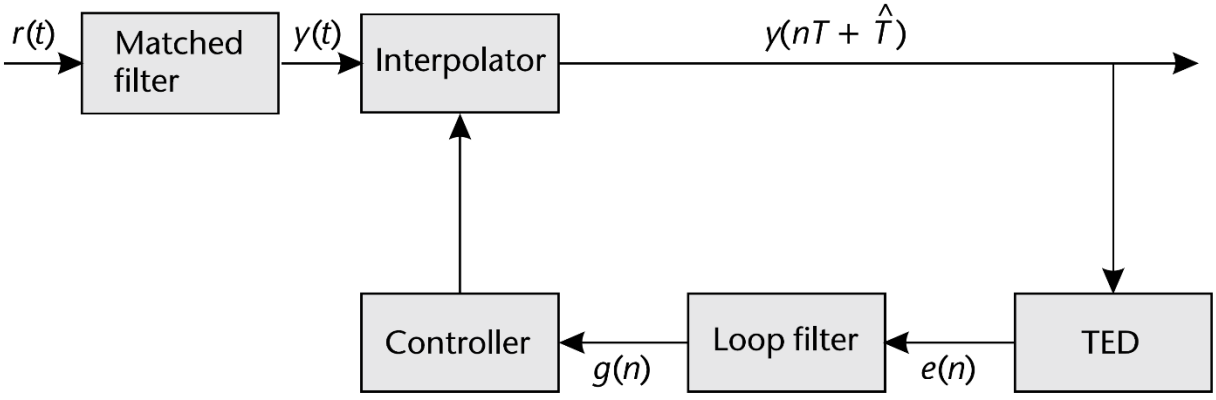
$$r(t) = \sum_n x(n)\, b(t - \tau(t) - nT_s) + n(t) \quad (3.1)$$

Όπου το r(t) είναι το λαμβανόμενο σήμα, x(n) είναι το n-οστό εκπεμπόμενο σύμβολο, b είναι ο παλμός του φίλτρου του πομπού, $T_S$ είναι ο ρυθμός δειγματοληψίας και n ο θόρυβος.

Για την επίλυση αυτού του προβλήματος προσθέτουμε δύο διαδικασίες στο δέκτη ενός συστήματος. Η πρώτη είναι ένα PLL όπως περιγράφηκε γενικά από το Σχήμα 3.9 και η δεύτερη είναι η χρήση όμοιων φίλτρων ανυψωμένου συνημίτονου σε πομπό και δέκτη. Η διαδικασία αυτή λέγεται "Matched Filtering". Το PLL στη περίπτωση του συγχρονισμού στο χρόνο παίρνει τη παρακάτω μορφή που φαίνεται στο Σχήμα 3.12. Αυτή η διαδικασία έχει ως σκοπό την βελτίωση της σηματοθορυβικής σχέσης στο δέκτη, όπως επίσης και την εξάλειψη φαινομένων όπως η διασυμβολική παρεμβολή [12] [24].

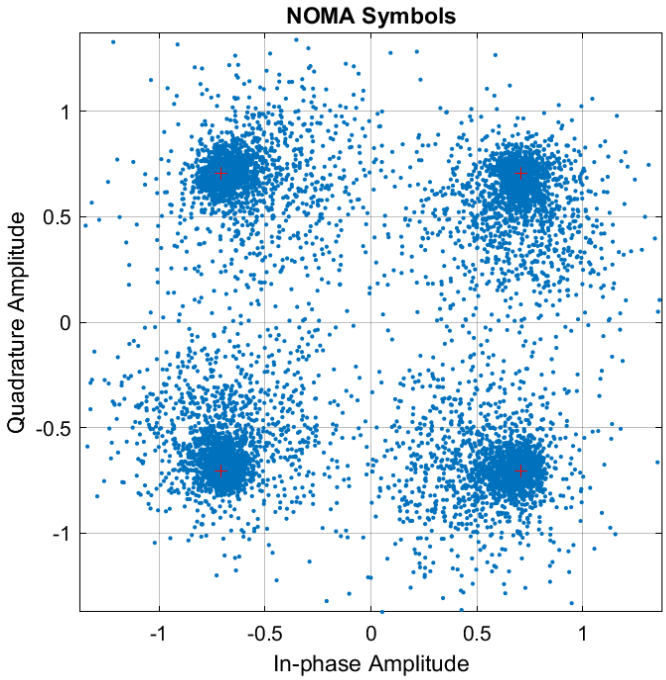
Το PLL του συγχρονισμού στο χρόνο διεξάγει τις διαδικασίες που φαίνονται στο παραπάνω Σχήμα 3.12. Αυτές είναι η ανίχνευση του χρονικού σφάλματος (TED), το φίλτρο του βρόγχου, και ο "interpolator" με τον ελεγκτή του (Controller) [24]. Οι παραπάνω διαδικασίες αρχικά ανιχνεύουν το σφάλμα και στη συνέχεια ενημερώνουν το σύστημα έτσι ώστε να μπορούν να διορθωθούν και τα σύμβολα πληροφορίας που θα τροφοδοτηθούν μελλοντικά.

Να σημειωθεί εδώ ότι το χρονικό σφάλμα *τ* είναι άγνωστο κάθε χρονική στιγμή. Έχοντας όμως μετρήσει το χρονικό σφάλμα σε προηγούμενες καταστάσεις του συστήματος μπορούμε μέσω του "interpolator" να εκτιμήσουμε το σφάλμα και να βρούμε σωστά το λαμβανόμενο σύμβολο. Το σφάλμα χρονισμού μπορεί εύκολα να γίνει αντιληπτό από διάγραμμα αστερισμού ενός λαμβανόμενου σήματος.




Σχήμα 3.12: PLL συγχρονισμού στο χρόνο [12] [24]

Στο Σχήμα 3.13 φαίνεται ένα διάγραμμα αστερισμού που λάβαμε με τα SDR κατά τη εκτέλεση των πειραμάτων. Αποτέλεσμα του σφάλματος χρονισμού είναι τα διασκορπισμένα σύμβολα που φαίνονται. Σε αυτή τη περίπτωση ο TED σε συνδυασμό με τον interpolator δε μπόρεσαν να ανιχνεύσουν σωστά το σφάλμα, έτσι προκλήθηκε ο διασκορπισμός των συμβόλων στο επίπεδο που είχε ως αποτέλεσμα τη μείωση του SNR και την εμφάνιση λανθασμένων bit στο δέκτη. Για τη διαδικασία του συγχρονισμού στη MATLAB μπορεί να χρησιμοποιηθεί η συνάρτηση " comm.SymbolSynchronizer" [25]. Στην επόμενη παράγραφο συζητείται η δεύτερη βασική διαδικασία της λήψης ενός σήματος που είναι ο συγχρονισμός στη συχνότητα.




Σχήμα 3.13: Διάγραμμα αστερισμού με σφάλμα συγχρονισμού

### 3.2.2 Συγχρονισμός στη συχνότητα

Όπως αναφέρθηκε προηγουμένως οι ταλαντωτές δύο διαφορετικών συσκευών, παράγουν συχνότητες με κάποια μετατόπιση. Αυτή η διαφορά στις συχνότητες πομπού και δέκτη προκαλεί προβλήματα στη λήψη σημάτων. Η μετατόπιση αυτή μετράτε σε ppm (parts per million). Συγκεκριμένα η μετατόπιση συχνότητας της συσκευή Adalm Pluto έχει μετρηθεί στα 25ppm πριν από το συγχρονισμό και στα 2ppm μετά από έναν επιτυχημένο συγχρονισμό. Η μετατόπιση υπολογίζεται από τη παρακάτω σχέση [12]

$$f_{Omax} = \frac{f_c \cdot PPM}{10^6} \tag{3.2}$$

όπου $f_C$ είναι η φέρουσα συχνότητα, $f_{omax}$ η μετατόπιση. Η παραπάνω σχέση είναι βασική για το σχεδιασμό ενός συστήματος λόγω του ότι μας δίνει τη τελική μετρούμενη τιμή της. Το παραμορφωμένο λαμβανόμενο σήμα που έχει μετατοπιστεί στη συχνότητα περιγράφεται από τη παρακάτω σχέση [12]

$$r(k) = S(k)e^{j(2\pi f_o kT+\theta)} + n(k) \tag{3.3}$$

όπου το r είναι το λαμβανόμενο σήμα, T είναι η περίοδος του συμβόλου, S είναι το σήμα πληροφορίας που έχει υποστεί τη μετατόπιση στη φάση, θ είναι η φάση του φέροντος σήματος και n ο θερμικός θόρυβος.

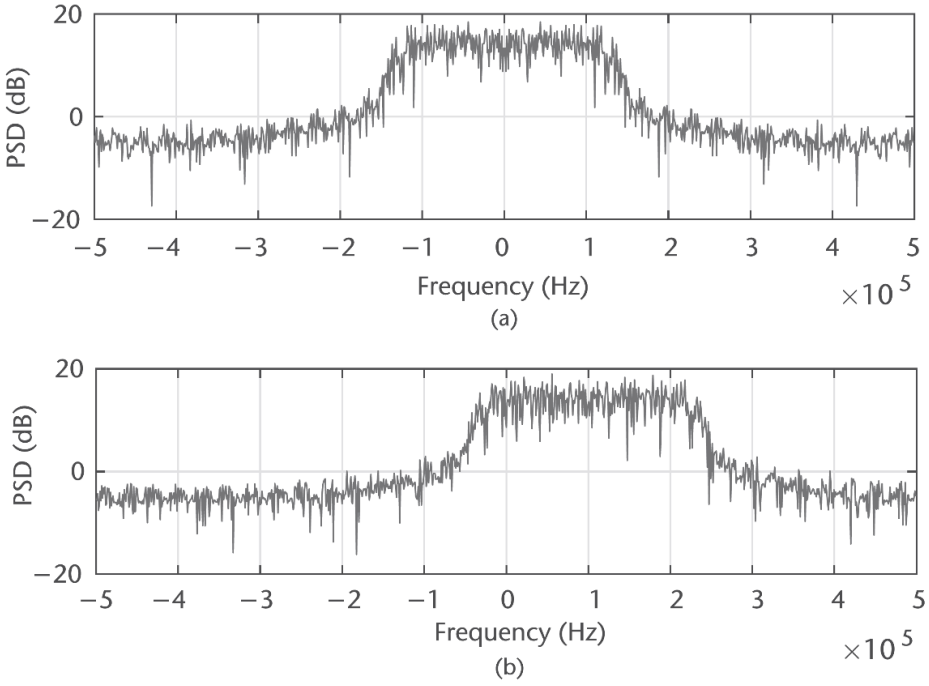
Όπως φαίνεται από το Σχήμα 3.10 ο συγχρονισμός στη συχνότητα έχει ως αποτέλεσμα και τον συγχρονισμό στη φάση. Στόχος αυτών των διαδικασιών είναι να παρέχουν ένα σταθερό διάγραμμα αστερισμού. Η σχέση της φάσης με τη συχνότητα δίνεται από τη παρακάτω σχέση

$$\omega = \frac{d\theta}{dt} = 2\pi f \tag{3.4}$$

όπου το θ είναι η γωνία του φάσορα που φαίνεται, ω η γωνιακή ταχύτητα και f η φέρουσα συχνότητα. Ο φάσορας θ είναι αυτός που υποδεικνύει τα σύμβολα της ψηφιακής διαμόρφωσης στο καρτεσιανό

επίπεδο. Από αυτόν γίνεται η τελική απόφαση στο δέκτη για το ποιο σύμβολο έστειλε ο πομπός. Για παράδειγμα στο Σχήμα 3.13 έχουμε σχήμα διαμόρφωσης QPSK, που σημαίνει ότι έχουμε τέσσερα σύμβολα πληροφορίας τα οποία έχουν από δύο bit το καθένα. Στη σχέση 3.5 φαίνεται πως αναλύεται ο φάσορας $\theta$ σε πραγματικό και φανταστικό μέρος. Δηλαδή σε έναν μιγαδικό αριθμό ο οποίος περιγράφει τα σύμβολα πληροφορίας. Επομένως είναι σημαντικό να μπορέσουμε να πετύχουμε ένα καλό διάγραμμα αστερισμού. Αν τα σύμβολα πληροφορίας λόγω της απόκλισης στη συχνότητα αλλάξουν τεταρτημόριο θα γίνουν λάθη στο δέκτη. Μέσω της διόρθωσης της συχνότητας εξασφαλίζουμε τη σωστή αποδιαμόρφωση του σήματος μειώνοντας έτσι τη πιθανότητα σφάλματος bit. Στο Σχήμα 3.14 φαίνεται γραφικά η απόκλιση στη συχνότητα [12].

$$\theta = tan^{-1}\left(\frac{Im(x(k))}{Re(x(k))}\right) \tag{3.5}$$




Σχήμα 3.14: Μετατόπιση συχνότητας [12]

Υπάρχουν δυο βασικές τεχνικές διόρθωσης της συχνότητας, η CFC (Coarse Frequency Correction) και η FFC (Fine Frequency Correction). Πρώτη στο δέκτη διεξάγεται η CFC, η οποία είναι διαδικασία ανοιχτού βρόγχου [12] [26]. Αυτή έχει σκοπό να διορθώσει το μεγαλύτερο μέρος της μετατόπισης, όπως φαίνεται στο Σχήμα 3.14. Υπάρχουν δύο τεχνικές CFC, μια που βασίζεται στη γνώση, από τη μεριά του δέκτη, χαρακτηριστικών του σήματος εκπομπής (DA – Data Aided), έτσι ώστε να γίνει η διαδικασία του συγχρονισμού πιο εύκολη. Στη δεύτερη ο δέκτης δεν έχει γνώση κανενός στοιχείου του λαμβανόμενου σήματος (NDA – Non Data Aided). Στο πείραμα που διεξάχθηκε στο πλαίσιο της εργασίας χρησιμοποιήθηκαν οι εσωτερικές συναρτήσεις της MATLAB "comm.CoarseFrequencyCompensator" και "comm.CarrierSynchronizer" [27] [28]. Στη συνέχεια του τρίτου κεφαλαίου θα παρουσιαστεί ο τρόπος λειτουργίας τους πάνω στο κώδικα που δημιουργήθηκε για την εξομοίωση του συστήματος. Η συνάρτηση "comm.CoarseFrequencyCompensator" (στη περίπτωση του πειράματος μας) λειτουργεί με βάση τον αλγόριθμο FFT και χωρίς να έχει ο δέκτης γνώση κάποιου στοιχείου του σήματος που εκπέμφθηκε. Η διαδικασία είναι ανοιχτού βρόγχου και περιγράφεται από τις εξισώσεις 3.6 και 3.7

$$r^M(k) = S^M(k)e^{j(2\pi f_o kT+\theta)M} \tag{3.6}$$

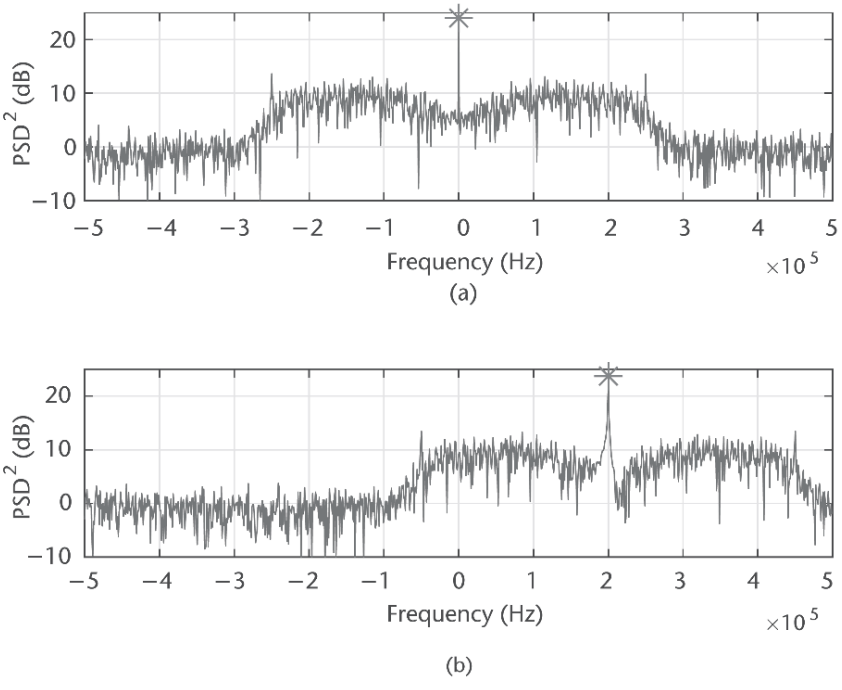
όπου M το σχήμα διαμόρφωσης, $r^M$ το λαμβανόμενο σήμα, k το δείγμα του FFT, S το σήμα πληροφορίας, $f_O$ η μετατόπιση συχνότητας, θ ο φάσορας που περιγράφηκε προηγουμένως και T η περίοδος του σήματος. Αρχικά η διαδικασία του CFC ανυψώνει τη σχέση 3.3 που εκφράζει το παραμορφωμένο σήμα με μετατόπιση της συχνότητας $f_o$ στο σχήμα διαμόρφωσης M αγνοώντας το θόρυβο. Στη συνέχεια για να υπολογίσουμε τη μετατόπιση στη συχνότητα, υπολογίζουμε τον FFT του $r^M(k)$, όπως φαίνεται στη σχέση 3.7

$$\hat{f}_O = \frac{1}{M\,K\,T}\arg\left|\sum_{k=0}^{K-1} r^M(k)e^{-j2\pi\kappa T/K}\right| \tag{3.7}$$

όπου K το μήκος του FFT. Από τη βιβλιογραφία [12], αποδεικνύεται ο τύπος 3.7 και υπολογίζεται η μετατόπιση. Στο Σχήμα 3.15 φαίνονται τα αποτελέσματα. Όσο μεγαλύτερο είναι το μήκος του FFT, τόσο μεγαλύτερη είναι και η ακρίβεια του υπολογισμού μας. Αυτή εκφράζεται από το πρώτο παράγοντα της εξίσωσης 3.7 που φαίνεται παρακάτω.

$$f_r = \frac{1}{M\,T\,K} \tag{3.8}$$

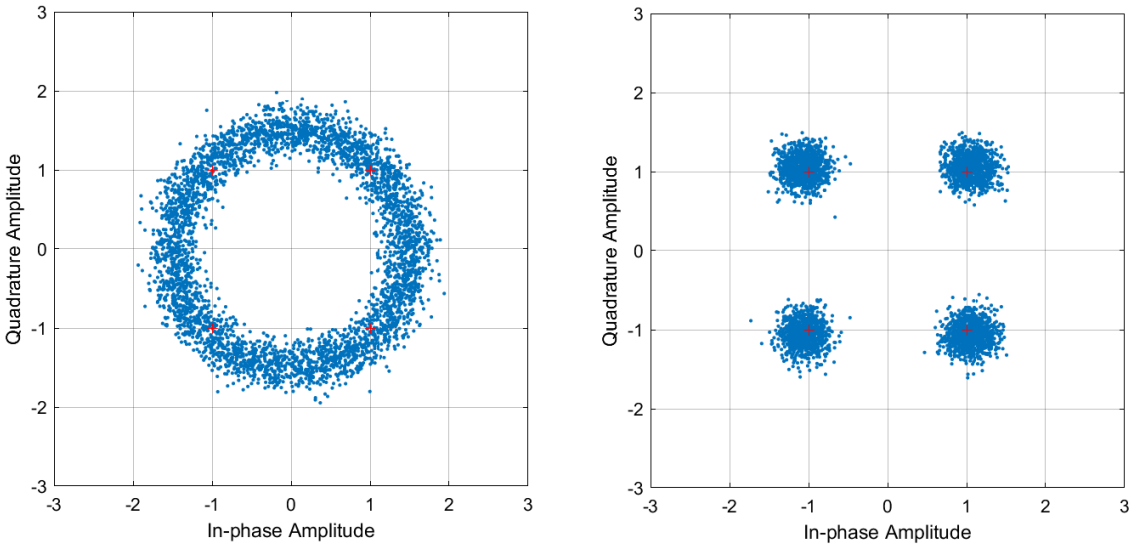
Η δεύτερη διαδικασία συγχρονισμού είναι η FFC όπου υλοποιείται με PLL. Μετά τη διαδικασία CFC υπάρχει ακόμα μετατόπιση, αυτή διορθώνεται μέσω του FFC [12]. Το αποτέλεσμα είναι ένα σταθερό διάγραμμα αστερισμού το οποίο μπορεί να αποδιαμορφωθεί με επιτυχία. Στο Σχήμα 3.16 φαίνονται τα διαγράμματα αστερισμού πριν και μετά τις διαδικασίες CFC και FFC, χρησιμοποιώντας τις συναρτήσεις “comm.CoarseFrequencyCompensator” και “comm.CarrierSynchronizer” στο λογισμικό της MATLAB.




Σχήμα 3.15: Εύρεση μετατόπισης συχνότητας μέσω της διαδικασίας CFC [12]

Φαίνεται επίσης η μετατόπιση στη φάση που προκαλείται από τη μετατόπιση στη συχνότητα. Αποτέλεσμα είναι να μειώνεται η σηματοθορυβική σχέση, τα σύμβολα διαμόρφωσης να αλλάζουν τεταρτημόριο και να γίνονται λάθη στα λαμβανόμενα bit. Όμως με τις διαδικασίες που περιγράφηκαν παραπάνω μπορούμε να μειώσουμε ή και να εξαλείψουμε το σφάλμα δημιουργώντας έτσι ένα δέκτη

που μπορεί να αντιμετωπίσει αυτά τα φαινόμενα. Αυτές οι διαδικασίες έχουν ως στόχο να αποδιαμορφωθεί σωστά το λαμβανόμενο σήμα στο δέκτη.




Σχήμα 3.16: Διάγραμμα αστερισμού πριν (αριστερά) και μετά (δεξιά) τις διαδικασίες Coarse Frequency Correction και Fine Frequency Correction

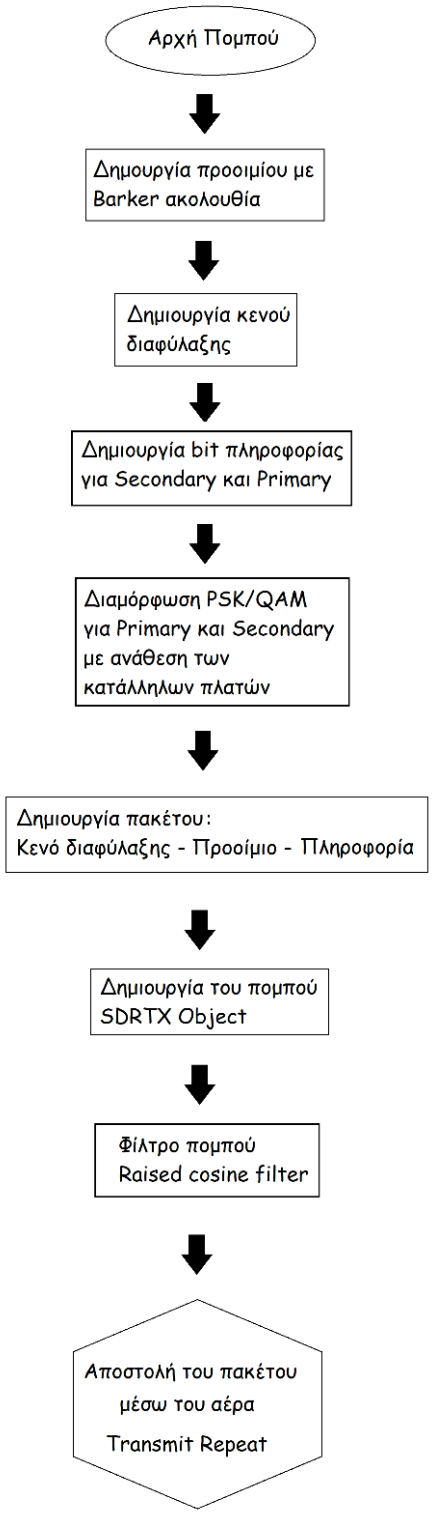
## 3.3 Πρώτη προσέγγιση – Απλό PD-NOMA

### 3.3.1 Σχεδιασμός πομπού

Αρχικός στόχος ήταν η μελέτη της τεχνικής PD-NOMA πάνω ένα απλουστευμένο σύστημα εκπομπής και λήψης έτσι ώστε να μπορέσουμε να αξιολογήσουμε τα αποτελέσματα της τεχνικής αυτής καθ' αυτής. Στο Σχήμα 3.17 φαίνεται το διάγραμμα ροής του κώδικα του πομπού που υλοποιήθηκε. Στη συνέχεια θα παρουσιαστούν κομμάτια από τους κώδικες της προσομοίωσης. Για το προγραμματισμό του συστήματος χρησιμοποιήθηκε η εργαλειοθήκη της MATLAB "Communication System Toolbox" [29].

Οι πρώτες ενέργειες του πομπού είναι να δημιουργήσει την ακολουθία του προοιμίου χρησιμοποιώντας τη συνάρτηση "comm.BarkerCode" και το κενό διαφύλαξης (guard interval). Το κενό διαφύλαξης αποτελείται από ένα σύνολο μιγαδικών αριθμών κοντά στο μηδέν και χρησιμεύει στο να ξεχωρίζουν τα πακέτα εκπομπής μεταξύ τους στο χρόνο [30].

```
barker = comm.BarkerCode('SamplesPerFrame',2^7,'Length',13);
preamble = complex(barker());
preamble_length = length(preamble);

Total_preamble = [preamble;preamble];
Total_preamble_length = length(Total_preamble);
release(barker);

%Create a Frame guard
guard = complex(0.0001*randn(1,100)');
guard_length = length(guard);
```




Σχήμα 3.17: Διάγραμμα ροής πομπού απλού PD-NOMA

Για τη δημιουργία και διαμόρφωση των bit πληροφορίας χρησιμοποιούνται οι εσωτερικές συναρτήσεις της MATLAB “randi” και “pskmod” αντίστοιχα. Τα bit αφού δημιουργηθούν, οργανώνονται σε σύμβολα με βάση το σχήμα διαμόρφωσης [31]. Μετά τη διαμόρφωση στις γραμμές 19 και 26 αναθέτονται τα πλάτη στα σήματα του Primary και Secondary χρήστη αντίστοιχα.

```
11 N = 300; %Number of bits
12 M = 2;
13 %Primary information
14 data_bits_p = rand(1,N*log2(M))>0.5;
15 data_bits_r_p = reshape(data_bits_p',log2(M),N)';
16 data_symbols_p = bi2de(data_bits_r_p,'left-msb');
17 data_symbols_length_p = length(data_symbols_p);
18 %Modulation for Primary
19 Mod_data_p = 0.9*pskmod(data_symbols_p,M,pi);

20 %Secondary information
21 data_bits_s = rand(1,N*log2(M))>0.5;
22 data_bits_r_s = reshape(data_bits_s',log2(M),N)';
23 data_symbols_s = bi2de(data_bits_r_s,'left-msb');
24 data_symbols_length_s = length(data_symbols_s);
25 %Modulation for Secondary
26 Mod_data_s = 0.1*pskmod(data_symbols_s,M,pi);
```

Όπως αναφέρθηκε η υπέρθεση δύο σημάτων είναι μια αλγεβρική πρόσθεση μεταξύ μιγαδικών αριθμών. Στις παρακάτω γραμμές κώδικα φαίνεται το τελευταίο κομμάτι της δημιουργίας της πληροφορίας του πομπού όπου είναι η υπέρθεση και η συναρμολόγηση του πακέτου πληροφορίας.

```
27 %Superposition
28 Mod_data = Mod_data_p + Mod_data_s;

29 %Create Frame
30 msg_tx = [guard;preamble;preamble;Mod_data];
```

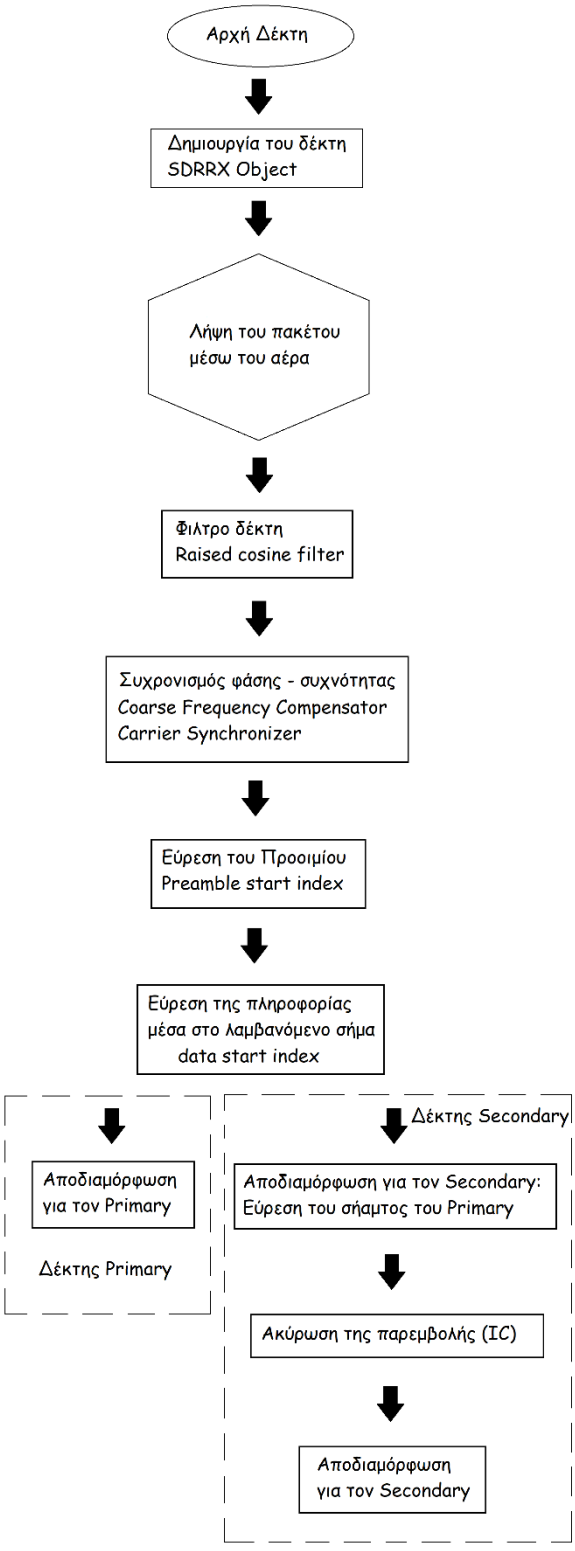
Για την εκπομπή του σήματος από τη κεραία του πομπού χρειάζεται να προγραμματιστεί το αντικείμενο (object) “sdrtx” της εργαλειοθήκης “communication system toolbox”. Αυτό είναι που δίνει στο πομπό της απαραίτητες πληροφορίες για την εκπομπή [32].

```
31 sampleRate = 300e3; %Nyquist Sampling Rate
32 centerFreq = 915e6; %Carrier Frequency
33 data_rate = 20e3; %Data rate

34 Samples_per_Symbol = sampleRate/data_rate; %Samples taken for one Symbol

35 samples_per_frame = 3*Samples_per_Symbol*
36 guard_length + data_symbols_length_s + Total_preamble_length);

37 tx = sdrtx('Pluto','RadioID','usb:0','CenterFrequency', centerFreq, ...
38         'BasebandSampleRate', sampleRate,'OutputDataType','double', ...
39    'Gain',0,'ShowAdvancedProperties', true,'FrequencyCorrection',0);

40 rctFilt = comm.RaisedCosineTransmitFilter('OutputSamplesPerSymbol', ...
41         Samples_per_Symbol,'FilterSpanInSymbols',10);

42 tx_signal = rctFilt(msg_tx);
43 transmitRepeat(tx,tx_signal);
```

Στο παραπάνω κώδικα παρουσιάζεται η δήλωση των μεταβλητών που είναι απαραίτητες για την εκπομπή σημάτων. Μεγέθη όπως ο ρυθμός δειγματοληψίας (sampleRate), που είναι συνδεδεμένος με το εύρος του φάσματος, η κεντρική φέρουσα συχνότητα (centerFreq) και η εκπεμπόμενη ισχύς πρέπει να δηλωθούν μέσα στο “sdrtx” για να εκπεμφθεί ένα σήμα. Αφού γίνουν οι απαραίτητες δηλώσεις το σήμα περνάει μέσα από το φίλτρο του πομπού (rctFilt) και στη συνέχεια αποστέλλεται μέσω του αέρα με την εντολή “transmitReapeat” [33] [34].

### 3.3.2 Σχεδιασμός δέκτη

Στο Σχήμα 3.18 φαίνεται το διάγραμμα ροής της λήψης και της ανίχνευσης του σήματος στους δέκτες του Secondary και Primary. Στα διακεκομμένα μπλοκ φαίνονται οι διαφορετικές λειτουργίες των χρηστών.




Σχήμα 3.18: Διάγραμμα ροής δέκτη Secondary απλού PD-NOMA

Παρακάτω φαίνεται η αντίστοιχη η διαδικασία του δέκτη για τη λήψη του σήματος και χρησιμοποιεί το object “sdrrx”, αντίστοιχο με αυτό του πομπού. Αφού ληφθεί το σήμα στο δέκτη περνάει από ένα φίλτρο ανυψωμένου συνημιτόνου. Εδώ αξίζει να σημειωθεί πως ο πομπός χρησιμοποιεί και αυτός ένα παρόμοιο φίλτρο πριν την εκπομπή. Η διαδικασία αυτή λέγεται “matched filtering” και στόχο έχει την μεγιστοποίηση του SNR στο δέκτη [35] [36].

```
rx = sdrrx('Pluto','RadioID','usb:1',...
   'GainSource','Manual','BasebandSampleRate',sampleRate,'Gain',35, ...
'CenterFrequency',centerFreq,'OutputDataType','double', ...
'SamplesPerFrame',samples_per_frame,'ShowAdvancedProperties',true, ...
'FrequencyCorrection',0);

  % Capture frames of data
[rx_signal,~] = rx();
release(tx);
release(rx);
rcrFilt = comm.RaisedCosineReceiveFilter('InputSamplesPerSymbol', ...
Samples_per_Symbol,'DecimationFactor',Samples_per_Symbol,...
  'FilterSpanInSymbols',10);
rx_filtered_signal = rcrFilt(rx_signal);
```

Για το συγχρονισμό πομπού και δέκτη χρησιμοποιήθηκαν οι συναρτήσεις “comm.CoarseFrequencyCompensator” και “comm.CarrierSynchronizer”. Οι συναρτήσεις αυτές βασίζονται στις αρχές των FFT και PLL. Στη συνέχεια ο δέκτης βρίσκει την αρχή του σήματος πληροφορίας με τη συνάρτηση “comm.PreambleDetector” χρησιμοποιώντας τις ρυθμίσεις που δώσαμε στο προοίμιο κατά τη κατασκευή του πομπού [37]. Θεωρείται ότι ο δέκτης έχει γνώση των βασικών παραμέτρων εκπομπής και λήψης, όπως τη κεντρική συχνότητα, το εύρος ζώνης και τις ρυθμίσεις του προοιμίου.

```
CFC = comm.CoarseFrequencyCompensator('SampleRate',fs, ...
   'Modulation','BPSK','FrequencyResolution',1);
FFC = comm.CarrierSynchronizer( ...
   'SamplesPerSymbol',1,'Modulation','BPSK');
Prbdet = comm.PreambleDetector([preamble;preamble], ...
   'Threshold',3,'Detections','All');

[rx_coarse,estfreqoff] = CFC(rx_filtered_signal);
[rx_fine,phErr] = FFC(rx_coarse);
[idx1,detmet1] = prbdet(rx_fine);
```

Βρίσκοντας με τη βοήθεια του προοιμίου την αρχή του λαμβανόμενου σήματος, ο δέκτης μπορεί πλέον να συνεχίσει για την αποδιαμόρφωση του. Στους κώδικες που παρατίθενται παρακάτω φαίνεται η διαδικασία της αποδιαμόρφωσης του Primary και του Secondary χρήστη.

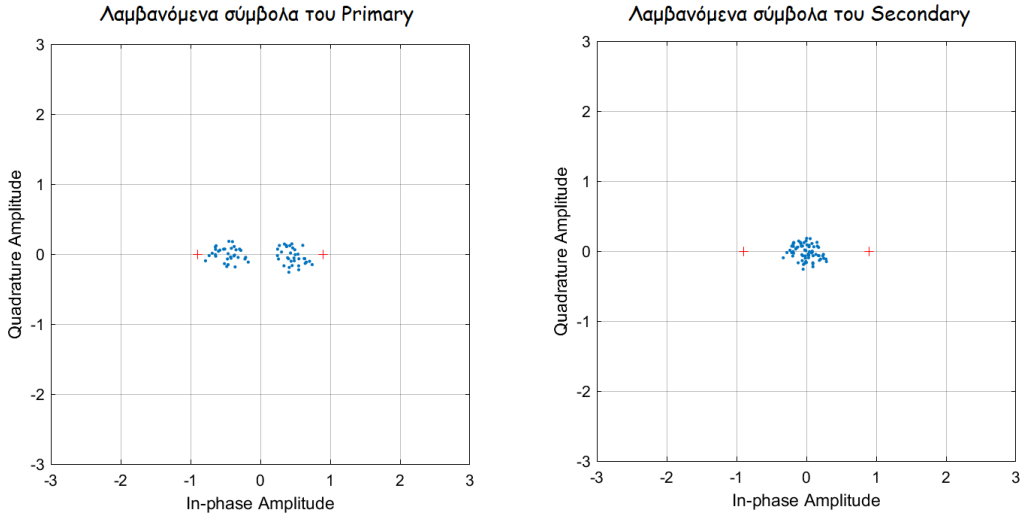
```
   %Primary Demodulation
rx_message_symbols = rx_fine(data_start_index:data_end_index);
Demod_p = pskdemod(rx_message_symbols,M,pi);
errors_p = Demod_p - data_symbols_p ;
numberErrors_p = sum(Demod_p ~= data_symbols_p)
constdiagram(rx_message_symbols)
```

Με βάση τη θεωρία ο Primary αγνοεί τη παρεμβολή του Secondary και συνεχίζει την αποδιαμόρφωση του λαμβανόμενου σήματος. Αυτό συμβαίνει διότι κατέχει το μεγάλο ποσοστό της ισχύος. Από την άλλη ο Secondary πρέπει να ακυρώσει τη παρεμβολή του Primary πριν αποδιαμορφώσει. Αυτό υλοποιείται με το παρακάτω κώδικα.

```
    %Secondary Demodulation
70  rx_message_symbols_1 = rx_fine(data_start_index:data_end_index);
71  rx_message_symbols_1_real = real(rx_message_symbols_1);
72  rx_message_symbols_1_abs = abs(rx_message_symbols_1_real);
73  mean_rx_message_symbols_1 = mean(rx_message_symbols_1_abs);
74  Demod_p_s = pskdemod(rx_message_symbols_1,M,pi); %Demodulate as Primary
75  est_signal_p = mean_rx_message_symbols_1*pskmod(Demod_p_s,M,pi); %Modulate again
76  Secondary_cancellation = rx_message_symbols_1 - est_signal_p; %SIC
77  Demod_s = pskdemod(Secondary_cancellation,M,pi);
78  errors_s = Demod_s - data_symbols_s;
79  numberErrors_s = sum(Demod_s ~= data_symbols_s)
```

Στον παραπάνω κώδικα φαίνεται η διαδικασία της ακύρωσης της παρεμβολής από το δέκτη του Secondary. Στη γραμμή 75 φαίνεται η αναδημιουργία του σήματος του Primary στο δέκτη του Secondary. Η διαδικασία της ακύρωσης της παρεμβολής πραγματοποιείται στη γραμμή 76, όπου φαίνεται η αλγεβρική αφαίρεση του σήματος του Primary από το συνολικό λαμβανόμενο. Τέλος αποδιαμορφώνεται το σήμα που προκύπτει από την αφαίρεση και γίνεται υπολογισμός των λανθασμένων bit που λήφθηκαν.

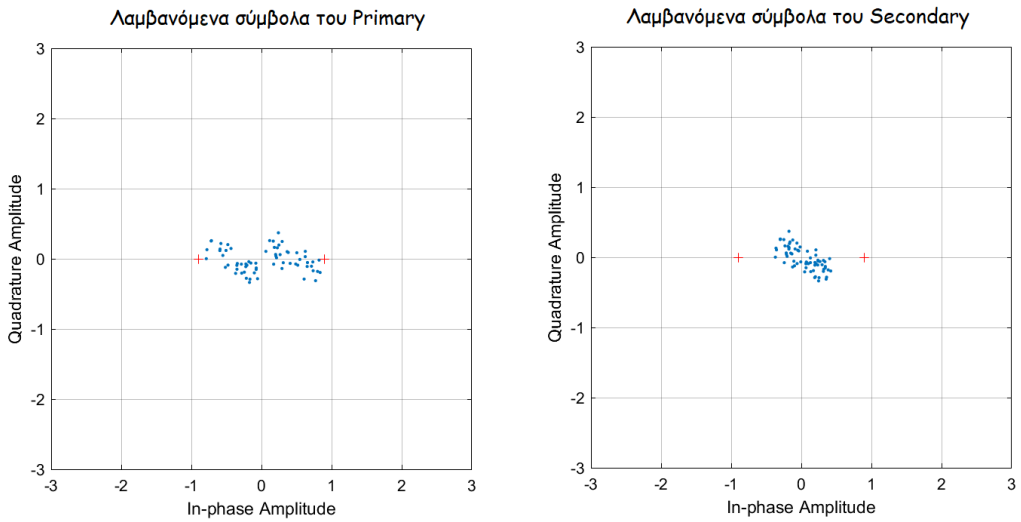
Η παραπάνω παρουσίαση έγινε για να γίνει κατανοητή η λειτουργία των SDR και το πώς μπορούν να προγραμματιστούν. Επίσης στις γραμμές 70 – 79 φαίνεται με ένα απλό πρόγραμμα πως μπορεί να γίνει η ακύρωση της παρεμβολής του σήματος του Primary στο δέκτη του Secondary. Η παραπάνω προσέγγιση δε λειτούργησε όπως αναμενόταν. Παρόλες τις προσπάθειες δε επιτεύχθηκε SNR πάνω από 10dB στο δέκτη. Το χαμηλό SNR σε συνδυασμό με το χαμηλό ποσοστό ισχύος που αναθέτουμε στον Secondary έκανε αδύνατη την επίτευξη μιας επιθυμητής πιθανότητας σφάλματος στο δέκτη του. Ο λόγος που συνέβαινε αυτό ήταν κατά κύριο λόγο οι διαδικασίες του συγχρονισμού σε συχνότητα και φάση των λαμβανόμενων συμβόλων στο δέκτη. Όταν αυτές δεν λειτουργούν βέλτιστα τότε τα λαμβανόμενα σύμβολα αποκλίνουν αρκετά από τις καταστάσεις αναφοράς τους. Στο Σχήμα 3.19 φαίνονται τα διαγράμματα αστερισμού στο δέκτη του SDR για διαμόρφωση BPSK και στους δύο χρήστες. Αριστερά απεικονίζονται τα λαμβανόμενα σύμβολα του Primary και δεξιά του Secondary.




Σχήμα 3.19: Αριστερά: Σύμβολα του Primary. Δεξιά: Σύμβολα του Secondary

Είναι φανερό ότι από τα λαμβανόμενα σύμβολα του Secondary θα γίνουν λάθη στα bit πληροφορίας. Στο Σχήμα 3.20 φαίνεται η απόκλιση στη φάση του λαμβανόμενου σήματος. Τα σύμβολα σε ένα σήμα διαμορφωμένο κατά BPSK βρίσκονται πάνω στον άξονα τετμημένων. Άμα η απόκλιση στη φάση είναι

μεγάλη μπορούν να δημιουργηθούν λάθη, ειδικά σε μεγαλύτερα σχήματα διαμόρφωσης όπου τα σύμβολα βρίσκονται πιο κοντά μεταξύ τους.

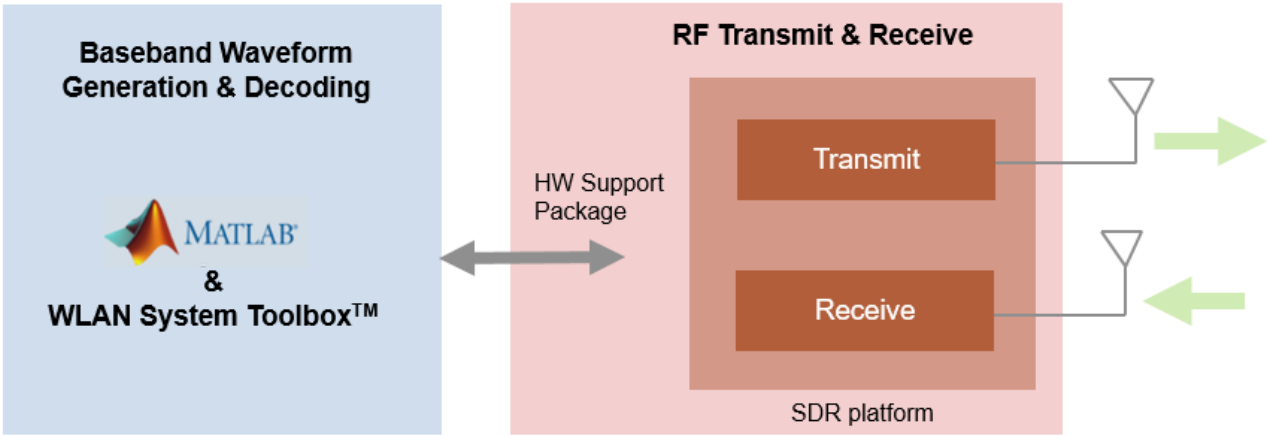



Σχήμα 3.20: Απόκλιση στη φάση σύμβολα του Primary (αριστερά) σύμβολα του Secondary (δεξιά)

Για να διορθωθούν τα παραπάνω προβλήματα και να βελτιωθεί το SNR και η πιθανότητα σφάλματος υλοποιήθηκε η τεχνική PD – NOMA πάνω στο σύστημα WLAN. Στο δεύτερο κεφάλαιο παρουσιάστηκε η βασική θεωρία. Στην παράγραφο 3.4 θα γίνει μια αναλυτική παρουσίαση του κώδικα που υλοποιήθηκε.

## 3.4 Δεύτερη προσέγγιση – WLAN PD-NOMA

Η υλοποίηση βασίστηκε πάνω στο "WLAN System toolbox" της MATLAB. Η παρακάτω εικόνα δείχνει το γενικό διάγραμμα εκπομπής και λήψης ενός σήματος WLAN. Το πακέτο έχει παραδείγματα προγραμματισμού στα οποία βασίστηκε η εργασία.




Σχήμα 3.21: Εκπομπή και λήψη ενός WLAN σήματος [39]

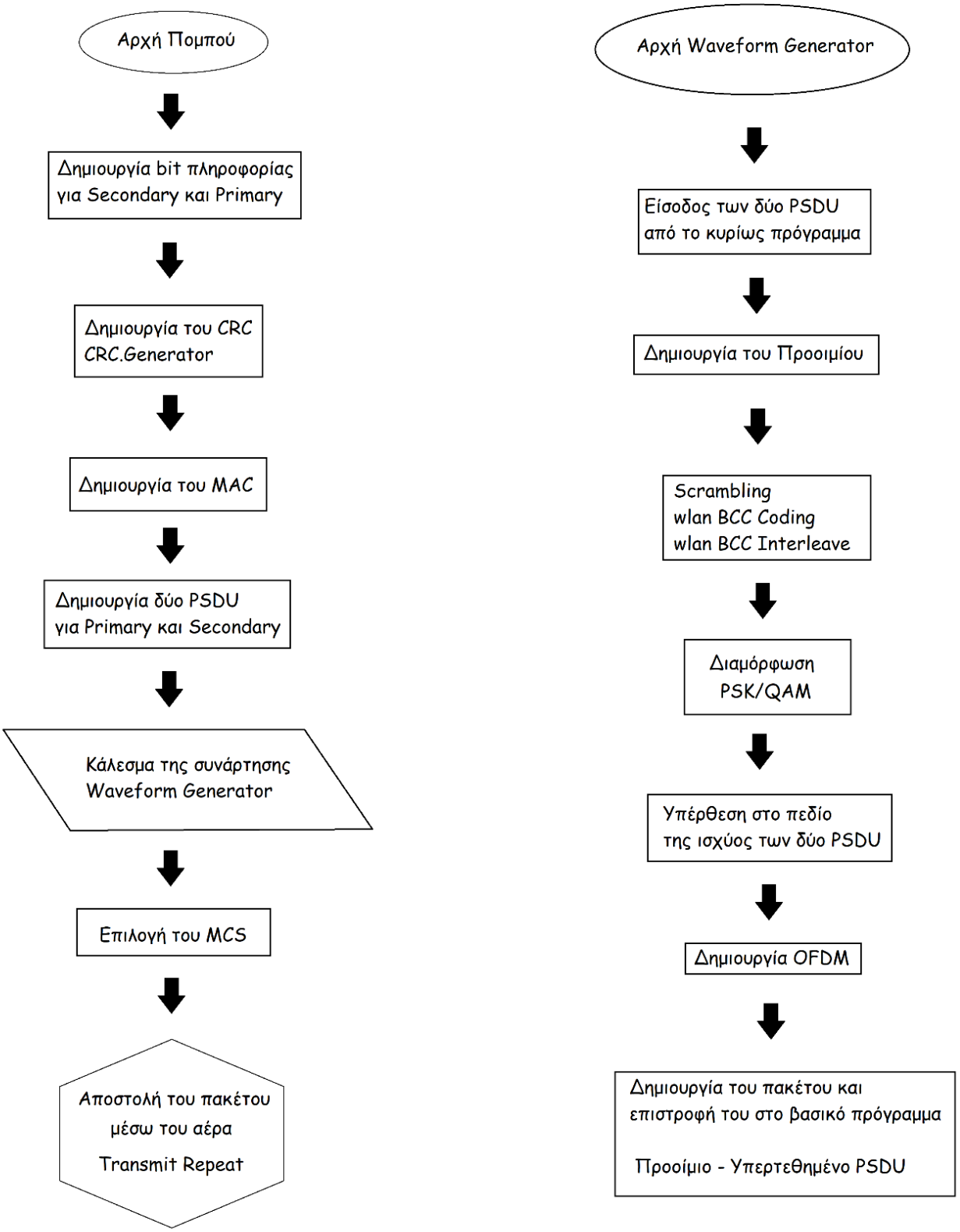
Βασικότερο από τα παραδείγματα είναι το " Image Transmission and Reception Using WLAN Toolbox and One PlutoSDR" [38]. Μέσω αυτού μπορεί ο χρήστης να αποστείλει μια εικόνα από το πομπό στο δέκτη του ίδιου Adalm Pluto SDR. Το παράδειγμα ξεκινάει με τη ψηφιοποίηση της εικόνας και τη δημιουργία μιας ακολουθίας από κωδικοποιημένων bit. Στη συνέχεια αφού διαμορφωθεί και κωδικοποιηθεί το σήμα αποστέλλεται μέσω του αέρα. Ο δέκτης διεξάγει τις διαδικασίες του συγχρονισμού στο χρόνο και στη συχνότητα και τέλος αποκωδικοποιεί και αποδιαμορφώνει το λαμβανόμενο σήμα. Εμείς παρεμβήκαμε σε αυτές τις διαδικασίες ρυθμίζοντας το πρόγραμμα να λειτουργεί σε δύο SDR. Να σημειωθεί, ότι ο κώδικας που αφορά την αποστολή και λήψη της φωτογραφίας δε συμπεριλήφθηκε, ως πληροφορία δημιουργήθηκαν τυχαία bit.

Στη συνέχεια υλοποιήθηκε κώδικας για τη δημιουργία του υπερτεθειμένου PSDU, όπως περιγράφηκε παραπάνω στη θεωρία. Ρυθμίστηκε κατάλληλα η δημιουργία του OFDM έτσι ώστε να μπορέσουμε να φορτώσουμε σε κάθε υποφέρον ένα υπερτεθειμένο σύμβολο πληροφορίας. Το σήμα περνάει μέσα από το κανάλι και λαμβάνεται από τη κεραία του δέκτη. Εκεί, αφού το σήμα διορθωθεί ως προς της συχνότητα και τη φάση, ξεκινάει η διαδικασία ανίχνευσης των PSDU του Primary και Secondary. Επίσης, προγραμματίστηκε κώδικας που διεξάγει τη διαδικασία της ακύρωσης της παρεμβολής και βρίσκει το PSDU του Secondary. Υπενθυμίζεται ότι ο Primary αγνοεί τη παρεμβολή του Secondary και αποδιαμορφώνει το σήμα σαν ένα κλασικό WLAN σύστημα. Στις επόμενες παραγράφους παρουσιάζονται κομμάτια του κώδικα που υλοποιήθηκε και βασίστηκε πάνω στο παράδειγμα που αναφέρθηκε παραπάνω. Θα αναλυθούν με λεπτομέρεια οι διαδικασίες της διαμόρφωσης, κωδικοποίησης από τη μεριά του πομπού και οι αντίστοιχες από τη μεριά του δέκτη μέσω των οποίων γίνεται η ανίχνευση της πληροφορίας.

### 3.4.1 Σχεδιασμός πομπού

Στο διάγραμμα ροής του Σχήματος 3.22 παρουσιάζεται γραφικά η λειτουργία του πομπού WLAN PD – NOMA. Η διαδικασία του πομπού ξεκινάει με τη δημιουργία των bit πληροφορίας, του MAC header και του CRC [39]. Το MAC χρησιμοποιείται για τον έλεγχο της ροής των πακέτων της πληροφορίας ενώ το CRC (Cyclic Redundancy Check) για την επισήμανση λαθών εντός του πακέτου πληροφορίας. Τα παραπάνω δημιουργούν το MPDU όπως φαίνεται στο Σχήμα 2.13. Στο παρακάτω κώδικα φαίνεται προγραμματιστικά η δημιουργία τους.

```
% Primary information
1  data_bits_p = rand(1,N*log2(Z))>0.5;
2  data_bits_p_r = reshape(data_bits_p',N,log2(Z));
   % Secondary information
3  data_bits_s = rand(1,N*log2(Z))>0.5;
4  data_bits_s_r = reshape(data_bits_s',N,log2(Z));
5  data_bits = [data_bits_p_r ; data_bits_s_r];
   % Data bits length
6  data_bits_length = length(data_bits);
7  msduLength = 2024; % MSDU length in bytes
8  msduBits = msduLength*8;
9  numMSDUs = ceil(length(data_bits)/msduBits);
10 padZeros = msduBits-mod(length(data_bits),msduBits);
11 txData = [data_bits; zeros(padZeros,1)];
   % Generate FCS and from an MPDU. The FCS is calculated using the standard
   % generator polynomial of degree 32 as defined in section 8.2.4.8 of
   % 802.11n HT standard
12 generatorPolynomial = [32 26 23 22 16 12 11 10 8 7 5 4 2 1 0];
13 fcsGenerator = comm.CRCGenerator(generatorPolynomial);
14 fcsGenerator.InitialConditions = 1;
15 fcsGenerator.DirectMethod = true;
16 fcsGenerator.FinalXOR = 1;
```




Σχήμα 3.22: Διάγραμμα ροής πομπού WLAN PD – NOMA

```
% Divide input data stream into fragments
numFragment = 0;
bitsPerOctet = 8;
lengthMACheader = 256; % MPDU header length in bits
lengthFCS = 32;        % FCS length in bits
lengthMPDU = (2*lengthMACheader)+(2*lengthFCS)+msduBits; % MPDU length in bits
data = zeros(lengthMPDU*numMSDUs,1);

for ind=0:numMSDUs-1
  % Extract bits for each MPDU
  frameBody = txData(ind*msduBits+1:msduBits*(ind+1),:);
  % Generate MPDU header bits
  mpduHeader = helperNonHTMACHeader(mod(numFragment, ...
                16),mod(ind,4096));

  % Create MPDU with header, body and FCS
  psdu1 = fcsGenerator([mpduHeader;frameBody(1:N)]);
  psdu2 = fcsGenerator([mpduHeader;frameBody((N+1):(2*N))]);
  psdu = [psdu1;psdu2];
  % Concatenate PSDUs for waveform generation
  data(lengthMPDU*ind+1:lengthMPDU*(ind+1)) = psdu;
end
```

Στις γραμμές 1 έως 22 δημιουργούνται τα στοιχεία του MPDU όπως εξηγήθηκαν παραπάνω. Για κάθε χρήστη δημιουργείται ένα δικό του PSDU με ξεχωριστό MAC header και CRC. Στο βρόγχο “for” της γραμμής 23 δημιουργείται το MPDU για κάθε πακέτο εκπομπής, στη δικιά μας περίπτωση στέλνουμε ένα πακέτο σε κάθε εκτέλεση του προγράμματος. Στη γραμμή 31 βάζουμε τα PSDU που Primary και Secondary στη σειρά για την περεταίρω επεξεργασία τους.

```
  % *Generate IEEE 802.11a Baseband WLAN Signal*
nonHTcfg = wlanNonHTConfig;         % Create packet configuration
nonHTcfg.MCS = 2;
nonHTcfg.NumTransmitAntennas = 1;   % Number of transmit antenna
chanBW = nonHTcfg.ChannelBandwidth;
nonHTcfg.PSDULength = (lengthMPDU/2)/bitsPerOctet; % Set the PSDU length

  %Transmitter Setup
sdrTransmitter = sdrtx(deviceNameSDR); % Transmitter properties
sdrTransmitter.RadioID = 'usb:0';

fs = wlanSampleRate(nonHTcfg); % Transmit sample rate in MHz
osf = 1.5;                     % Oversampling factor

sdrTransmitter.BasebandSampleRate = fs*osf;
sdrTransmitter.CenterFrequency = 2.432e9;  % Channel 5
sdrTransmitter.ShowAdvancedProperties = true;
sdrTransmitter.Gain = txGain;

  % Initialize the scrambler with a random integer for each packet
scramblerInitialization = 43;
  % Generate baseband NonHT packets separated by idle time
txWaveform = wlanWaveformGenerator_NOMA(data,nonHTcfg, ...
   'NumPackets',numMSDUs,'IdleTime',20e-6, ...
   'ScramblerInitialization',scramblerInitialization);

  % Transmit RF waveform
sdrTransmitter.transmitRepeat(txWaveform);
```

Από τη γραμμή 35 έως και τη γραμμή 47 δηλώνονται οι παράμετροι εκπομπής και λήψης. Το όνομα του SDR και σε ποια θύρα USB είναι συνδεδεμένο, η φέρουσα συχνότητα και το εύρος ζώνης, η ισχύς εκπομπής και το σχήμα διαμόρφωσης και κωδικοποίησης. Εδώ να σημειωθεί ότι το κλασικό WLAN χρησιμοποιεί BCC (Binary Convolutional Coding) κωδικοποίηση καναλιού. Η κωδικοποίηση

αυτή βασίζεται στη συνέλιξη για τη διόρθωση λαθών. Ο πίνακας με την ονομασία "txWaveform" εκφράζει το τελικό υπερτεθειμένο σήμα που προκύπτει από τη συνάρτηση "wlanWaveformGenerator_NOMA" και προγραμματίστηκε για να μπορέσουμε να δημιουργήσουμε την υπέρθεση. Στο παρακάτω κομμάτι κώδικα παρουσιάζεται η προαναφερθείσα συνάρτηση, η οποία βασίζεται στην εσωτερική συνάρτηση της MATLAB που ονομάζεται "wlanWaveformGenerator" [40]. Μέσω αυτής υλοποιείται η τεχνική PD – NOMA. Το διάγραμμα ροής της φαίνεται στο Σχήμα 3.22. Ο πίνακας – μεταβλητή "txWaveform" είναι το τελικό σήμα που οδηγούμε στη κεραία του SDR. Παρακάτω θα παρουσιαστούν οι αλλαγές που έγιναν στη κλασσική "wlanWaveformGenerator" έτσι ώστε να λειτουργήσει ως NOMA.

Το πρώτο πράγμα που κάνει η συνάρτηση "wlanWaveformGenerator" είναι να δημιουργήσει το προοίμιο. Όπως αναφέρθηκε παραπάνω το προοίμιο χρησιμεύει στο συγχρονισμό πομπού και δέκτη. Με μια αλληλουχία από εντολές ελέγχου τύπου "if" ο κώδικας ελέγχει το configuration του WLAN που έχει επιλεγεί ώστε να φτιάξει το κατάλληλο προοίμιο.

```
1 elseif isa(cfgFormat,'wlanHTConfig') % HT-MF format

2    htSig = wlanHTSIG(cfgFormat);
3    htstf = wlanHTSTF(cfgFormat);
4    htltf = wlanHTLTF(cfgFormat);
5    preamble = [lstf; lltf; lsig; htSig; htstf; htltf];

6 elseif isa(cfgFormat,'wlanNonHTConfig')

7    if strcmp(cfgFormat.Modulation, 'OFDM')
8        preamble = [lstf; lltf; lsig];
9    else % DSSS
10        preamble = [wlan.internal.wlanDSSSPreamble(cfgFormat); ...
11                    wlan.internal.wlanDSSSHeader(cfgFormat)];
12    end
```

Όπως ειπώθηκε παραπάνω επιλέχθηκε η ρύθμιση "NonHT" που σημαίνει Non High Throughput. Άρα πρόγραμμα επιλέγει το κατάλληλο "elseif" της γραμμής 6. Στη συνέχεια έχοντας επιλέξει τη διαμόρφωση να είναι OFDM το προοίμιο διαμορφώνεται όπως φαίνεται στη γραμμή 8. Το παραπάνω πρόγραμμα είναι μέρος που συνολικού κώδικα δημιουργίας του προοιμίου. Ολόκληρος ο κώδικας βρίσκεται στο παράρτημα της εργασίας.

```
   % Extract PSDU for the current packet
13 psdu_p = getPSDUForCurrentPacket(dataCell_p,numPSDUBits, i);
14 psdu_s = getPSDUForCurrentPacket(dataCell_s,numPSDUBits, i);
```

Τα PSDU εισέρχονται στη συνάρτηση "WaveformGenerator" ως ένας πίνακας που περιέχει και τις δύο πληροφορίες. Μέσα στη συνάρτηση η κάθε πληροφορία εξάγεται σε ξεχωριστό πίνακα. Η συνάρτηση "getPSDUForCurrentPacket" ορίζεται στο τέλος της "WaveformGenerator".

Στη συνέχεια παρουσιάζεται η διαδικασία που διαμορφώνει τα bit πληροφορίας σε σύμβολα πληροφορίας και τα αναθέτει σε OFDM σύμβολα. Πρόκειται για ένα βρόγχο που "for"που λειτουργεί για τον αριθμό των πακέτων εκπομπής. Όπως προ αναφέρθηκε έχουμε ένα πακέτο πληροφορίας σε κάθε εκπομπή, άρα ο βρόγχος τρέχει μια φορά. Το πρόγραμμα εισέρχεται στη κατάλληλη δομή ελέγχου "if" στη γραμμή 28 ανάλογα με το configuration που επιλέχθηκε. Από εκεί και πέρα τα bit των PSDU περνάνε στη συνάρτηση "wlanNonHTdata_NOMA" [41]. Εκεί κωδικοποιούνται και διαμορφώνονται ανάλογα με το σχήμα διαμόρφωσης και κωδικοποίησης που έχει επιλεχθεί.

```
15 for i = 1:numPackets
   % Generate the PSDU with the correct scrambler initial state
16 if  isa(cfgFormat,'wlanVHTConfig')
17  if any(cfgFormat.APEPLength > 0)
18    data = wlanVHTData(psdu,cfgFormat,pktScramInit(i,:));
19    else  % NDP
20        data = complex(zeros(0,cfgFormat.NumTransmitAntennas));
21    end
22  elseif isa(cfgFormat,'wlanHTConfig') % HT-MF format
23   if cfgFormat.PSDULength > 0
24      data = wlanHTData(psdu{1},cfgFormat,pktScramInit(i,:));
25     else  % NDP or sounding packet
26      data = complex(zeros(0,cfgFormat.NumTransmitAntennas));
27   end
28  elseif isa(cfgFormat,'wlanNonHTConfig') % non-HT format
29   if strcmp(cfgFormat.Modulation, 'OFDM')
30    [mappedData, numSym] = wlanNonHTData_NOMA(psdu_p{1},cfgFormat,pktScramInit(i,:));
31     mappedData_p = mappedData;
32     mappedData = wlanNonHTData_NOMA(psdu_s{1},cfgFormat,pktScramInit(i,:));
33     mappedData_s = mappedData;
       %Power allocation
34     k_p = 1;
35     k_s = 0.2;
36     mappedData_primary = k_p.*mappedData_p;
37     mappedData_secondary = k_s.*mappedData_s;
       %Superposition
38     mappedData = mappedData_primary + mappedData_secondary;
       % Determine number of symbols and pad length
39     chanBW = cfgFormat.ChannelBandwidth; %cfgNonHT
40        if (strcmp(chanBW, 'CBW10') || strcmp(chanBW, 'CBW5'))
41         numTx = 1;  % override and set to 1 only, for 802.11j/p
42         else
43            numTx  = cfgFormat.NumTransmitAntennas;
44        end
45     cfgOFDM = wlan.internal.wlanGetOFDMConfig(chanBW, 'Long', 'Legacy');
46     FFTLen = cfgOFDM.FFTLength; %waveform gen
47     CPLen  = cfgOFDM.CyclicPrefixLength; %waveform gen
48     z = 1; % Offset by 1 to account for HT-SIG pilot symbol
49     pilotValues = wlan.internal.nonHTPilots(numSym,z);
50     wout = complex(zeros((FFTLen + CPLen)*numSym, numTx));
51     packedData = complex(zeros(FFTLen, 1));
52     csh = wlan.internal.getCyclicShiftVal('OFDM', numTx, 20);
53     for i = 1:numSym
           % Data packing with pilot insertion
54         packedData(cfgOFDM.DataIndices, :) = mappedData(:,i);
           % Add pilots
55         packedData(cfgOFDM.PilotIndices, :) = pilotValues(:,i);
           % Tone rotation and replicate over Tx
56         pDataMat = repmat(packedData.*cfgOFDM.CarrierRotations, 1, numTx);
           % Cyclic shift applied per Tx
57         dataCycShift =  wlan.internal.wlanCyclicShift(pDataMat, csh, FFTLen, 'Tx');
           % OFDM modulate - pCPLen
58         wout((1:(FFTLen+CPLen)).' + (i-1)*(FFTLen+CPLen), :) = ...
59         wlan.internal.wlanOFDMModulate(reshape(dataCycShift, FFTLen, 1, ...
60         numTx), CPLen);
61      end
62    end

63   end
```

Στα διαμορφωμένα πλέον σύμβολα πληροφορίας, στις γραμμές 34-38, ανατίθενται τα ποσοστά των πλατών και γίνεται η υπέρθεση των δύο σημάτων. Από τις γραμμές 39 έως 60 υλοποιείται η τεχνική

OFDM. Στο βρόγχο “for” της γραμμής 53 αναθέτονται σε κάθε επανάληψη τα σύμβολα πληροφορίας στα υποφέροντα, ανάλογα με το configuration που έχει επιλεχθεί. Να σημειωθεί ότι στο OFDM χρησιμοποιούνται 52 υποφέροντα από τα οποία τα 48 φέρουν πληροφορία ενώ τα τέσσερα έχουν πιλοτικά σύμβολα. Στο παρακάτω κομμάτι κώδικα φαίνεται το εσωτερικό της συνάρτησης “wlanNonHTData_NOMA”.

```
  % SERVICE, Section 18.3.5.2, all zeros

1 serviceBits = zeros(16,1,'int8');
  % Scramble padded data
2 paddedData = [serviceBits; PSDU; zeros(Ntail,1); zeros(numPad, 1)];
3 scrambData = wlanScramble(paddedData, scramInitBits);
  % Zero-out the tail bits again for encoding
4 scrambData(16+length(PSDU) + (1:Ntail)) = zeros(Ntail,1);

  % BCC Encoding
5 encodedData = wlanBCCEncode(scrambData, rate);
  % BCC Interleaving
6 interleavedData = wlanBCCInterleave(encodedData, 'Non-HT', numCBPS);

  % Constellation mapping
7 mappedData = wlanConstellationMap(interleavedData, numBPSCS);
  % Reshape to form OFDM symbols
8 mappedData = reshape(mappedData, numCBPS/numBPSCS, numSym);
```

Στις παραπάνω εντολές δημιουργούνται τα σύμβολα πληροφορίας αφού πρώτα κωδικοποιηθούν τα bit πληροφορίας με τη χρήση του BCC κωδικοποιητή. Η διαδικασία του “interleaving” κάνει τη διαδικασία της κωδικοποίησης πιο αποτελεσματική τακτοποιώντας τα δεδομένα με ένα μη συνεχόμενο τρόπο. Ο πίνακας “mappedData” γυρνάει στη συνάρτηση “WaveformGenerator” τα σύμβολα πληροφορίας, έτσι ώστε να ανατεθούν στα υποφέροντα του OFDM.

```
      % Scale and output
64    y = wout * cfgOFDM.NormalizationFactor / sqrt(numTx);
65    data = y;
66    packet = [preamble; data];
```

Με τις παραπάνω γραμμές κώδικα γίνεται πολλαπλασιασμός του σήματος με το συντελεστή κανονικοποίησης του OFDM. Ο παρονομαστής της διαίρεσης αναφέρεται στις κεραίες του συστήματος. Το σύστημα που υλοποιήθηκε είναι σύστημα SISO, που σημαίνει ότι η κεραία εκπομπής είναι μια, άρα και ο παρονομαστής ισούται με μονάδα. Τέλος, στη γραμμή 66, το προοίμιο και τα δεδομένα δημιουργούν το τελικό πακέτο που οδηγείται στη κεραία του πομπού με την εντολή “transmitRepeat”.

### 3.4.2 Σχεδιασμός δέκτη

Σε αυτή τη παράγραφο θα συζητηθεί ο δέκτης WLAN PD – NOMA . Στο διάγραμμα ροής του Σχήματος 3.23 απεικονίζεται η λειτουργία του. Αρχικά στο δέκτη ορίζονται τα απαραίτητα δεδομένα της ζεύξης, όπως η κεντρική φέρουσα συχνότητα και το κέρδος της κεραίας. Στη συνέχεια λαμβάνεται το σήμα και απεικονίζεται το φάσμα του. Η διαδικασία αυτή φαίνεται στις παρακάτω γραμμές κώδικα.

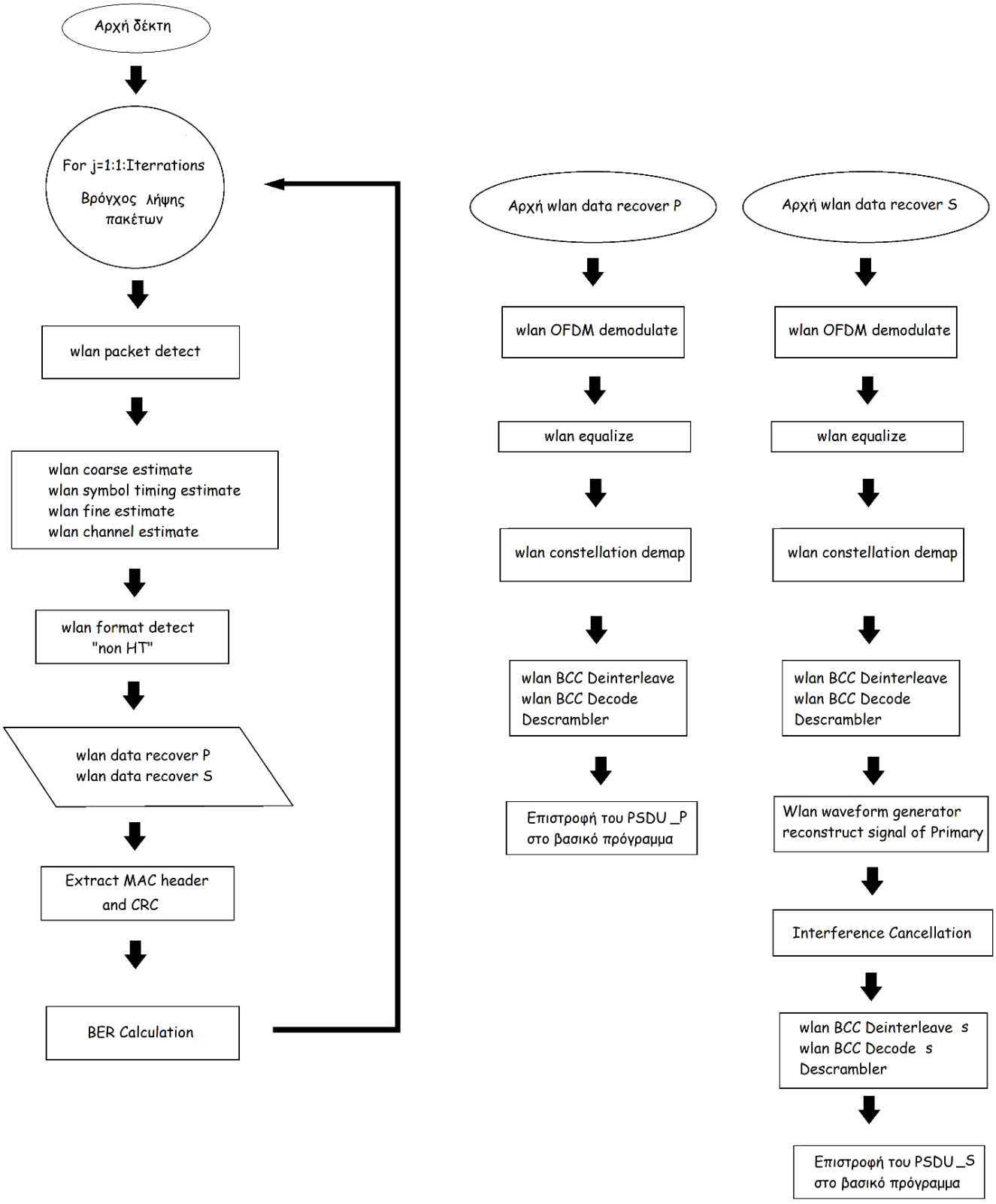
```
  %Receiver Setup
1 sdrReceiver = sdrrx(deviceNameSDR);
2 sdrReceiver.RadioID = 'usb:1';
3 sdrReceiver.BasebandSampleRate = sdrTransmitter.BasebandSampleRate;
4 sdrReceiver.CenterFrequency = sdrTransmitter.CenterFrequency;
5 sdrReceiver.GainSource = 'Manual';
6 sdrReceiver.Gain = 30;
7 sdrReceiver.OutputDataType = 'double';
```

```
8 burstCaptures = sdrReceiver();

% Show power spectral density of the received waveform
9 spectrumScope(burstCaptures);
```

Αμέσως μετά ο δέκτης ελέγχει αν έχει λάβει κάποιο πακέτο, σε περίπτωση που η σηματοθορυβική σχέση είναι πολύ χαμηλή, υπάρχει περίπτωση ο δέκτης να μη αντιληφθεί την ύπαρξη του λαμβανόμενου σήματος. Στη συνέχεια, όπως αναφέρθηκε στη θεωρία, μέσω του προοιμίου γίνονται οι διαδικασίες του συγχρονισμού σε συχνότητα και φάση. Παρακάτω φαίνεται αυτή η διαδικασία.

```
    % Packet detect
10  pktOffset = wlanPacketDetect(rxWaveform, chanBW, searchOffset, 0.8);

    % Adjust packet offset
11  pktOffset = searchOffset+pktOffset;
12  if isempty(pktOffset) || (pktOffset+double(indLSIG(2))>rxWaveformLen)
13     if pktInd==1
14         disp('** No packet detected **');
15     end
16     break;
17  end

    % Extract non-HT fields and perform coarse frequency offset correction
    % to allow for reliable symbol timing
18  nonHT = rxWaveform(pktOffset+(indLSTF(1):indLSIG(2)),:);
19  coarseFreqOffset = wlanCoarseCFOEstimate(nonHT,chanBW);
20  nonHT = helperFrequencyOffset(nonHT,fs,-coarseFreqOffset);

    % Symbol timing synchronization
21  fineTimingOffset = wlanSymbolTimingEstimate(nonHT,chanBW);

    % Perform fine frequency offset correction on the synchronized and
    % coarse corrected preamble fields
22  lltf = nonHT(indLLTF(1):indLLTF(2),:);           % Extract L-LTF
23  fineFreqOffset = wlanFineCFOEstimate(lltf,chanBW);
24  nonHT = helperFrequencyOffset(nonHT,fs,-fineFreqOffset);
25  cfoCorrection = coarseFreqOffset+fineFreqOffset;

    % Total CFO
    % Channel estimation using L-LTF
26  lltf = nonHT(indLLTF(1):indLLTF(2),:);
27  demodLLTF = wlanLLTFDemodulate(lltf,chanBW);
28  chanEstLLTF = wlanLLTFChannelEstimate(demodLLTF,chanBW);

    % Noise estimation
29  noiseVarNonHT = helperNoiseEstimate(demodLLTF);
```

Για την υλοποίηση αποδιαμόρφωσης και αποκωδικοποίησης ενός σήματος PD – NOMA προσαρμόστηκε κατάλληλα η εσωτερική συνάρτηση της MATLAB “wlanNonHTDataRecover” [42]. Όπως φαίνεται στο διάγραμμα ροής του Σχήματος 3.23 έχουμε δύο συναρτήσεις για τους δύο διαφορετικούς δέκτες των Primary και Secondary. Ο primary χρησιμοποιεί την πρώτη συνάρτηση στη γραμμή 30 και αποδιαμορφώνει θεωρώντας τον Secondary ως παρεμβολή. Από την άλλη ο secondary χρησιμοποιώντας τη συνάρτηση “wlanNonHTDataRecover_s” της γραμμής 33 βρίσκει το σήμα του Primary το αφαιρεί από το συνολικό και στη συνέχεια προχωράει με την αποδιαμόρφωση του σήματος του. Στις γραμμές κώδικα 30 έως 35 φαίνεται η κλήση των συναρτήσεων για την αποδιαμόρφωση του λαμβανόμενου σήματος. Η συνάρτηση “wlanNonHTDataRecover_s” προγραμματίστηκε για τις ανάγκες της ακύρωσης της παρεμβολής και βασίζεται στην εσωτερική συνάρτηση της MATLAB “wlanNonHTDataRecover”.




Σχήμα 3.23: Διάγραμμα ροής δέκτη WLAN PD - NOMA

```
    % Recover Primary's PSDU bits using transmitted packet parameters and channel
    % estimates from L-LTF
30  [rxPSDU_p,eqSym_p] = wlanNonHTDataRecover(rxWaveform(pktOffset+...
31          (indNonHTData(1):indNonHTData(2)),:), ...
32          chanEstLLTF,noiseVarNonHT,rxNonHTcfg);
    % Recover Secondary's PSDU bits using transmitted packet parameters and channel
    % estimates from L-LTF

33  [rxPSDU_s,Secondary_cancelation] = wlanNonHTDataRecover_s(rxWaveform(pktOffset+...
34            (indNonHTData(1):indNonHTData(2)),:), ...
35          chanEstLLTF,noiseVarNonHT,rxNonHTcfg);
36    constellation1(reshape(eqSym_p,[],1)); % Current constellation
37    pause(0); % Allow constellation to repaint
38    release(constellation1); % Release previous constellation plot
39    constellation2(reshape(Secondary_cancelation,[],1)); % Current constellation
40    pause(0); % Allow constellation to repaint
41    release(constellation2); % Release previous constellation plot
```

Οι συναρτήσεις “wlanNonHTDataRecover” και “wlanNonHTDataRecover_s” έχουν ως είσοδο το λαμβανόμενο σήμα αφού έχει διορθωθεί ως προς τη συχνότητα και τη φάση. Ως έξοδο δίνουν το λαμβανόμενο PSDU και τα equalized λαμβανόμενα σύμβολα. Στη συνέχεια από τις γραμμές 36 έως και 41 απεικονίζονται σύμβολα πληροφορίας και υπολογίζεται το Modulation Error Ratio (MER) μέσω της συνάρτησης “comm.ConstellationDiagram” που ορίζεται στην αρχή του προγράμματος και καλείται με τα ονόματα “ constellation1” και “constellation2” για τον κάθε χρήστη ξεχωριστά. Εδώ να σημειωθεί ότι, το MER είναι αναλογία της μέσης ισχύος του σήματος αναφοράς προς το μέσο τετραγωνικό σφάλμα. Η αναλογία αυτή αντιστοιχεί στο SNR ενός AWGN καναλιού [43][44]. Το MER υπολογίζεται από τη παρακάτω σχέση:

$$MER\ (dB) = 10 \cdot \log\left(\frac{\text{average symbol power}}{\text{average error power}}\right) \tag{3.9}$$

Στις παρακάτω γραμμές κώδικα φαίνεται το εσωτερικό της συνάρτησης “wlanNonHTDataRecover_s”. Μιας και οι δύο συναρτήσεις ανάκτησης της εκπεμπόμενης πληροφορίας είναι παρόμοιες, θα εξηγηθεί μόνο αυτή που υλοποιεί και το IC. Στο παράρτημα της εργασίας βρίσκονται ολόκληρες οι συναρτήσεις.

```
    % OFDM Demodulation
1   [ofdmDemodData, ofdmDemodPilots] = wlan.internal.wlanOFDMDemodulate(...
2   rxNonHTData(1:minInputLen, :), cfgOFDM, symOffset);

    %%%%%%Demodulate as primary%%%%%%%%

    % Equalization
3   [eqDataSym, csiData] = wlan.internal.wlanEqualize(ofdmDemodData, chanEstData,
4   eqMethod,    noiseVarEst);
    % Constellation demapping
5   qamDemodOut = wlanConstellationDemap(eqDataSym, noiseVarEst, mcsTable.NBPSCS);
    % Apply bit-wise CSI and concatenate OFDM symbols in the first dimension
6   qamDemodOut = bsxfun(@times, ...
7               reshape(qamDemodOut, mcsTable.NBPSCS, [], numOFDMSym), ...
8               reshape(csiData, 1, [])); % [Nbpscs Nsd Nsym]
9   qamDemodOut = reshape(qamDemodOut, [], 1);

    % Deinterleave
10  deintlvrOut = wlanBCCDeinterleave(qamDemodOut, 'Non-HT', mcsTable.NCBPS);

    % Channel decoding
11  decBits = wlanBCCDecode(deintlvrOut, mcsTable.Rate);

    % Derive initial state of the scrambler
12  scramInit = wlan.internal.scramblerInitialState(decBits(1:7));
```

```
   % Remove pad and tail bits, and descramble
13 if all(scramInit==0)
14    % Scrambler initialization invalid (0), therefore do not descramble
15    descramDataOut = decBits(1:(16+8*cfgNonHT.PSDULength));
16   else
17    descramDataOut = wlanScramble(decBits(1:(16+8*cfgNonHT.PSDULength)), scramInit);
18 end
   % Remove the 16 service bits
19  bits = descramDataOut(17:end);
20  bits_cell = mat2cell(bits,8384);
```

Η συνάρτηση ανάκτησης του PSDU αρχίζει με την αποδιαμόρφωση του OFDM με τη χρήση της κατάλληλης συνάρτησης. Στη συνέχεια με την ισοστάθμιση (equalization) του λαμβανόμενου σήματος παίρνουμε τα τελικά σύμβολα πληροφορίας, με τα οποία ο δέκτης θα αποφασίσει πιο σύμβολο του έστειλε ο πομπός. Η διαδικασία της ισοστάθμισης αυξάνει τη σηματοθορυβική σχέση, πράγμα σημαντικό για τους χρήστες σε ένα σύστημα PD – NOMA. Από τις γραμμές 3 έως και 20 γίνονται οι διαδικασίες τις αποδιαμόρφωσης και αποκωδικοποίησης, όπου καταλήγουν στα bit του PSDU. Στη γραμμή 19 λαμβάνουμε 8384 συνολικά bit από τα οποία τα 8096 είναι τα bit πληροφορίας, τα 256 το MAC header και 32 τα bit του CRC.

Τα bit αυτά με βάση τη θεωρία της ακύρωσης της παρεμβολής, κωδικοποιούνται και διαμορφώνονται ξανά με σκοπό να αφαιρεθούν από το συνολικό λαμβανόμενο σήμα. Αυτό μπορεί εύκολα να γίνει χρησιμοποιώντας τη συνάρτηση “wlanNonHTData_NOMA” που χρησιμοποιεί και ο πομπός. Στη γραμμή 21 δημιουργείται το σήμα του Primary στο δέκτη του Secondary και αμέσως μετά γίνεται η ακύρωση της παρεμβολής. Στη συνέχεια ο δέκτης του Secondary επαναλαμβάνει τη διαδικασία εύρεσης των bit όπως γίνεται από τις γραμμές 3 έως 20, μόνο που αυτή τη φορά τα τελικά bit είναι αυτά που προορίζονται για αυτόν.

```
21 mappedData = wlanNonHTData_NOMA(bits_cell{1},cfgNonHT,pktScramInit(1,:));

22 est_signal_p = mappedData;

   %Intereference Cancelation
23 Secondary_cancelation = eqDataSym - est_signal_p;

   % Constellation demapping
24 qamDemodOut_s = wlanConstellationDemap(Secondary_cancelation,
25     noiseVarEst, mcsTable.NBPSCS);

   % Apply bit-wise CSI and concatenate OFDM symbols in the first dimension
26 qamDemodOut_s = bsxfun(@times, ...
              reshape(qamDemodOut_s, mcsTable.NBPSCS, [], numOFDMSym), ...
              reshape(csiData, 1, [])); % [Nbpscs Nsd Nsym]
27 qamDemodOut_s = reshape(qamDemodOut_s, [], 1);

   % Deinterleave
28 deintlvrOut_s = wlanBCCDeinterleave(qamDemodOut_s, 'Non-HT', mcsTable.NCBPS);

   % Channel decoding
29 decBits_s = wlanBCCDecode(deintlvrOut_s, mcsTable.Rate);

   % Derive initial state of the scrambler
30 scramInit_s = wlan.internal.scramblerInitialState(decBits_s(1:7));

   % Remove pad and tail bits, and descramble
31 if all(scramInit_s==0)
    % Scrambler initialization invalid (0), therefore do not descramble
32    descramDataOut_s = decBits_s(1:(16+8*cfgNonHT.PSDULength));
33   else
34       descramDataOut_s = wlanScramble(
```

```
35        decBits_s(1:(16+8*cfgNonHT.PSDULength)), scramInit_s);
36 end

   % Remove the 16 service bits
37 bits_s = descramDataOut_s(17:end);

38 end
```

Η συνάρτηση γυρνάει στο κυρίως πρόγραμμα τον πίνακα “bits_s”. Από εκεί και πέρα αφαιρούνται τα MAC header και το CRC και τέλος υπολογίζεται και αποθηκεύεται ο ρυθμός των σφαλμάτων. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να ληφθεί ο απαραίτητος αριθμός bit για την εξαγωγή συμπερασμάτων.

```
   % Remove FCS from MAC header and frame body
42 [rxBit_p{pktInd},crcCheck_p] = fcsDetector(double(rxPSDU_p));
43 [rxBit_s{pktInd},crcCheck_s] = fcsDetector(double(rxPSDU_s));

   % Remove the MAC header and duplicate captured MAC fragment
44 rxBitMatrix_p = cell2mat(rxBit_p);
45 rxData_p = rxBitMatrix_p(lengthMACheader+1:end);

46 startSeq = find(packetSeq==0);
47 rxData_p = circshift(rxData_p,[0 -(startSeq(1)-1)]);% Order MAC fragments

   % Perform bit error rate (BER) calculation for primary
48 bitErrorRate = comm.ErrorRate;
49 err_p = bitErrorRate(double(rxData_p(:)), txData(1:8096));
50 fprintf('  \nBit Error Rate (BER) for Primary:\n');
51 fprintf('          Bit Error Rate (BER) = %0.5f.\n',err_p(1));
52 fprintf('          Number of bit errors = %d.\n', err_p(2));
53 fprintf('    Number of transmitted bits = %d.\n\n',length(data_bits_p_r));

   % Remove the MAC header and duplicate captured MAC fragment
54 rxBitMatrix_s = cell2mat(rxBit_s);
55 rxData_s = rxBitMatrix_s(lengthMACheader+1:end);

56 startSeq = find(packetSeq==0);
57 rxData_s = circshift(rxData_s,[0 -(startSeq(1)-1)]);% Order MAC fragments

   % Perform bit error rate (BER) calculation for secondary
58 bitErrorRate = comm.ErrorRate;
59 err_s = bitErrorRate(double(rxData_s(:)), txData(8097:16192));
60 fprintf('  \nBit Error Rate (BER) for Secondary:\n');
61 fprintf('          Bit Error Rate (BER) = %0.5f.\n',err_s(1));
62 fprintf('          Number of bit errors = %d.\n', err_s(2));
63 fprintf('    Number of transmitted bits = %d.\n\n',length(data_bits_s_r));
```

Με τις παραπάνω γραμμές κώδικα ολοκληρώνεται η υλοποίηση της τεχνικής WLAN PD – NOMA. Αφού αφαιρεθεί το MAC και το CRC χρησιμοποιείται η συνάρτηση “comm.ErrorRate” για τη καταγραφή των λαθών. Η συνάρτηση αυτή συγκρίνει τα bit πληροφορίας που προκύψανε από τη διαδικασία του δέκτη με αυτά που δημιουργήθηκαν αρχικά στο πομπό. Τέλος, τα λάθη αυτά αποθηκεύονται έτσι ώστε σε συνδυασμό με το συνολικό αριθμό των εκπεμπόμενων bit να εξαχθούν συμπεράσματα για τη λειτουργία του συστήματος.

## 3.5 Επίλογος τρίτου κεφαλαίου

Το τρίτο κεφάλαιο αφιερώθηκε στην εξομοίωση του συστήματος. Παρουσιάστηκαν κομμάτια του κώδικα που προγραμματίστηκε έτσι ώστε να υλοποιηθεί η τεχνική της πολυπλεξίας στο επίπεδο της ισχύος. Το ζήτημα προσεγγίστηκε από δύο πλευρές. Αρχικά υλοποιήθηκε ένα απλό σύστημα εκπομπής και λήψης ενός υπερτεθειμένου στην ισχύ σήματος που δεν λειτούργησε αποδοτικά. Το πρόβλημα ήταν

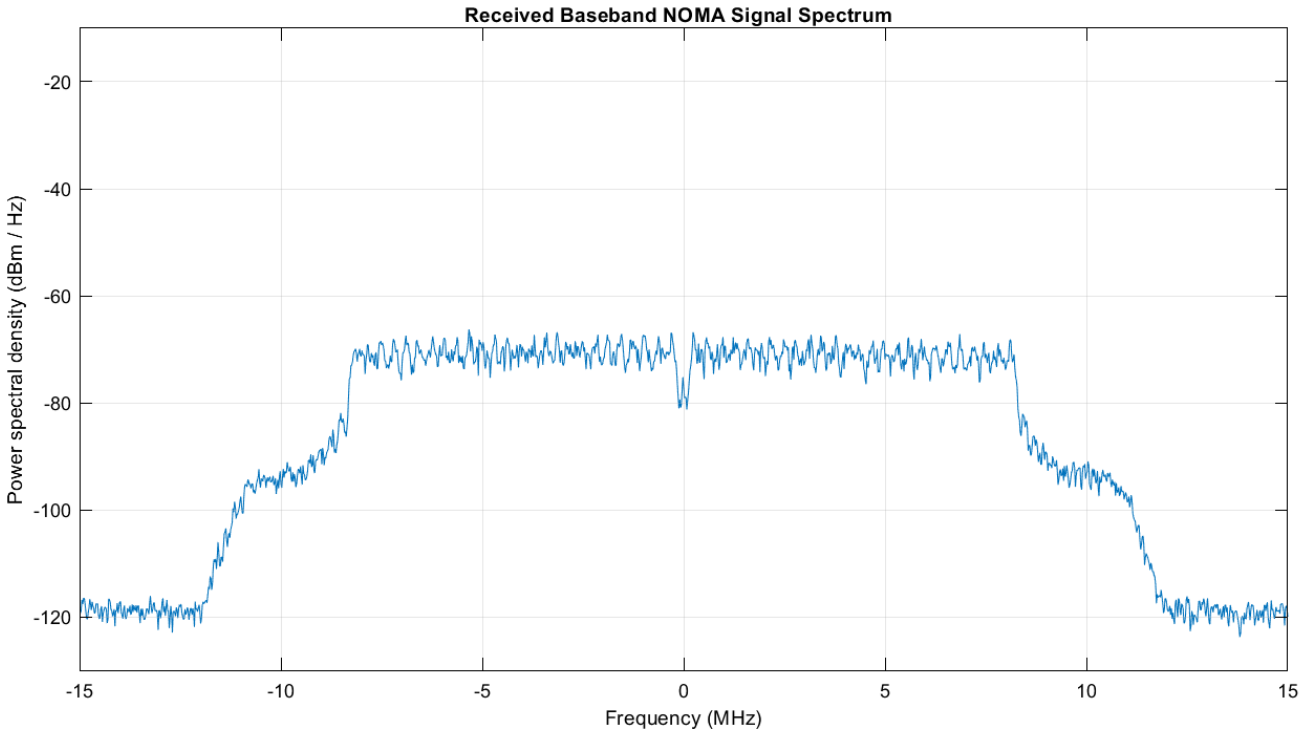
ότι ο δέκτης δε μπορούσε να πετύχει μια καλή σηματοθορυβική σχέση, πράγμα απαραίτητο για ένα σύστημα όπου ο Secondary χρήστης λαμβάνει ένα ποσοστό ισχύος της τάξης του 10% της συνολικής ισχύος. Το πρόβλημα το προκαλούσαν οι διαδικασίες συγχρονισμού και ισοστάθμισης του δέκτη καθώς η απόσταση των SDR και η φυσική ζεύξη δεν δικαιολογούσε αυτή την απόδοση. Αυτά τα προβλήματα δε μπόρεσαν να λυθούν και έτσι προσεγγίστηκε το θέμα διαφορετικά. Έτσι, χρησιμοποιήθηκε η εργαλειοθήκη της MATLAB “WLAN System”. Εκεί υπάρχουν παραδείγματα συγχρονισμού και ισοστάθμισης που χρησιμοποιήθηκαν έτσι ώστε να επιτευχθεί η τεχνική PD – NOMA. Αφού το σύστημα λειτούργησε, ξεκίνησε η διαδικασία εξαγωγής αποτελεσμάτων. Στο επόμενο κεφάλαιο θα παρουσιαστούν και θα αξιολογηθούν τα αποτελέσματα της εργασίας. Θα παρατεθούν διαγράμματα ρυθμού σφαλμάτων σε σχέση με το SNR για διαφορετικές περιπτώσεις του πειράματος, όπως επίσης και άλλες παρατηρήσεις πάνω στην εκπομπή και λήψη ασύρματων σημάτων.

# Κεφάλαιο 4ο: Αποτελέσματα

Στο τέταρτο κεφάλαιο παρουσιάζονται τα αποτελέσματα της εργασίας. Συγκρίνονται οι διαφορετικές παράμετροι του συστήματος και καταγράφονται παρατηρήσεις. Στη παράγραφο 4.1 συζητάται η επίδραση των διαλείψεων και των απωλειών της διάδοσης στο λαμβανόμενο σήμα. Θα παρουσιαστούν επίσης διαγράμματα αστερισμού και φάσματος για να γίνει πιο κατανοητή η επίδρασή τους. Στη συνέχεια στη παράγραφο 4.2 παρουσιάζονται τα αποτελέσματα ως προς τη πιθανότητα σφάλματος. Θα φανεί ότι το σύστημα που υλοποιήθηκε είναι σε θέση να επιτύχει μια καλή πιθανότητα σφάλματος. Επίσης θα παρουσιαστούν οι διαφορές για διαφορετικά σχήματα διαμόρφωσης και συντελεστές κατανομής της ισχύος.

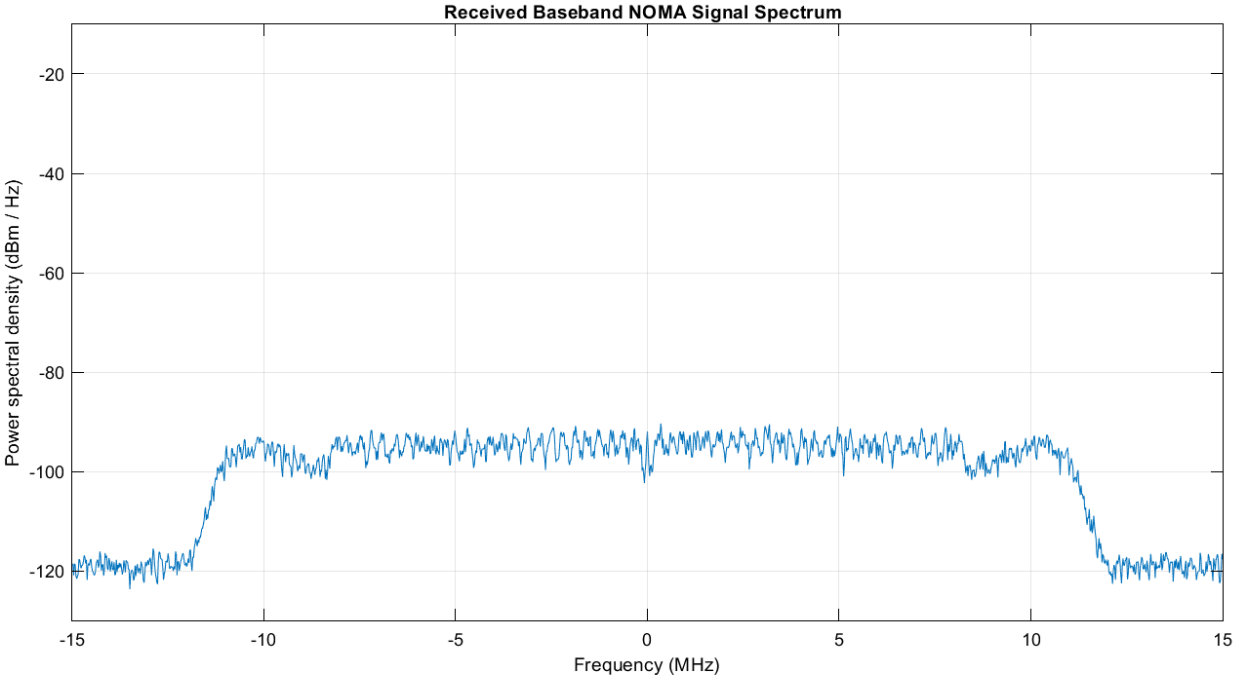
## 4.1 Παρεμβολές και διαλείψεις

Στο λαμβανόμενο φάσμα ενός σήματος φαίνεται η κατανομή της ισχύος ανά μονάδα συχνότητας. Στο Σχήμα 4.1 φαίνεται το γράφημα ενός φάσματος όπου, στον άξονα των τετμημένων έχει τη συχνότητα και στον άξονα των τεταγμένων τη φασματική πυκνότητα ισχύος σε dBm/Hz.



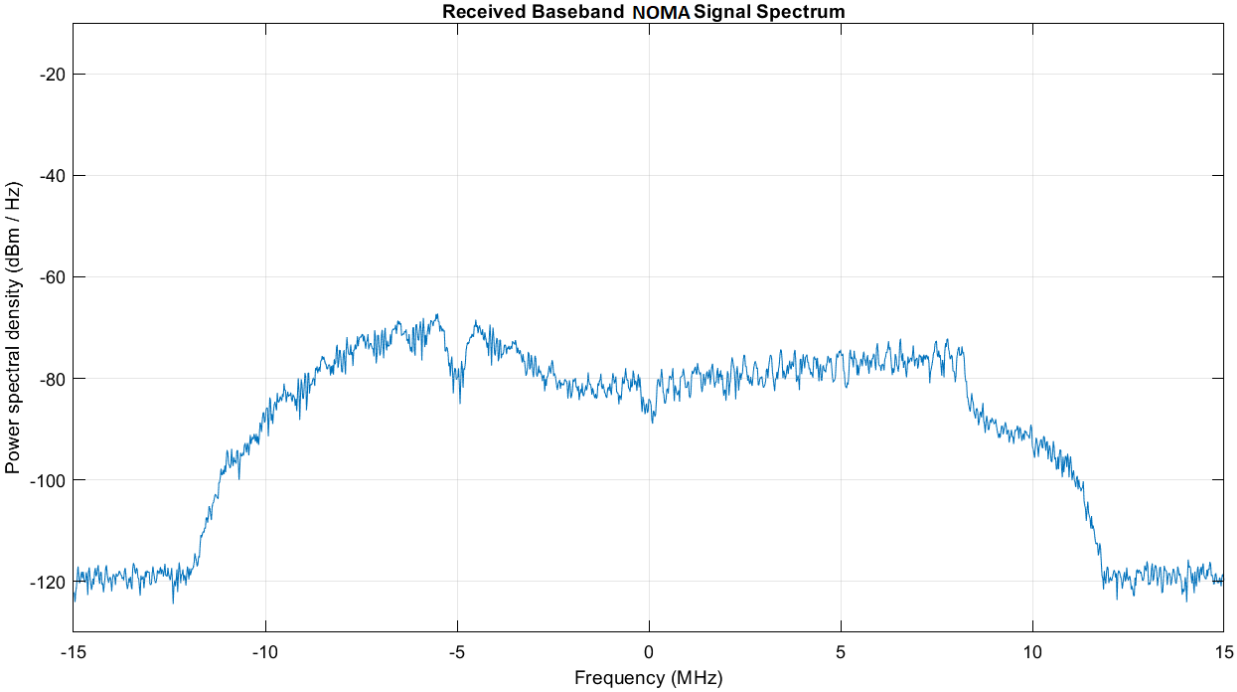

Σχήμα 4.1: Λαμβανόμενο φάσμα

Από το παραπάνω σχήμα μπορούμε να εξάγουμε κάποιες πληροφορίες. Κατ' αρχάς από των άξονα των συχνοτήτων φαίνεται το εύρος ζώνης των 20MHz που έχει επιλεγεί στην αρχικοποίηση του πομπού. Επίσης στον κάθετο άξονα βλέπουμε τη φασματική πυκνότητα ισχύος. Το συγκεκριμένο σήμα που απεικονίζεται παραπάνω έχει καλή στάθμη λήψης, περίπου -70dBm/Hz, επομένως ο ανιχνευτής δε θα έχει κάποιο πρόβλημα στην αποδιαμόρφωση του. Για να μη μπορέσει ο δέκτης να ανιχνεύσει το λαμβανόμενο σήμα θα πρέπει να μοιάζει όπως αυτό του Σχήματος 4.2.



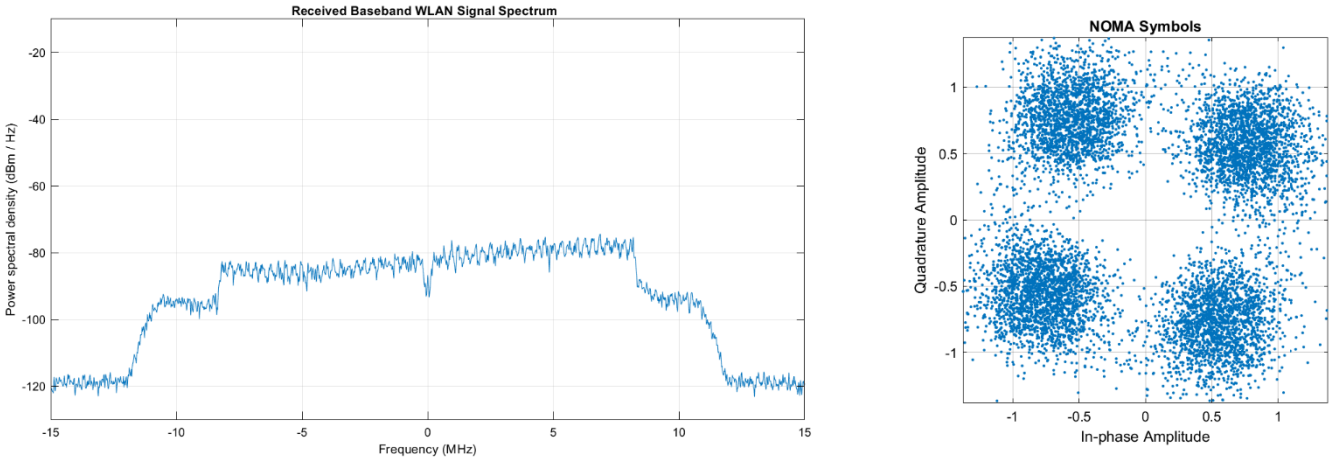

Σχήμα 4.2: Μη ανιχνεύσιμο λαμβανόμενο φάσμα

Τα παραπάνω αποτελέσματα πάρθηκαν από την εκπομπή και λήψη σημάτων με τα Adalm Pluto SDR και έχουν να κάνουν με τη λαμβανόμενη ισχύ και πως αυτή επηρεάζει το σήμα στο δέκτη, λόγω των απωλειών κατά τη διάδοση. Ένας άλλος παράγοντας παραμόρφωσης ενός σήματος είναι οι διαλείψεις που εξηγήθηκαν θεωρητικά στην έκτη παράγραφο του δεύτερου κεφαλαίου. Στο Σχήμα 4.3 φαίνεται ένα φάσμα παραμορφωμένο από τις διαλείψεις.



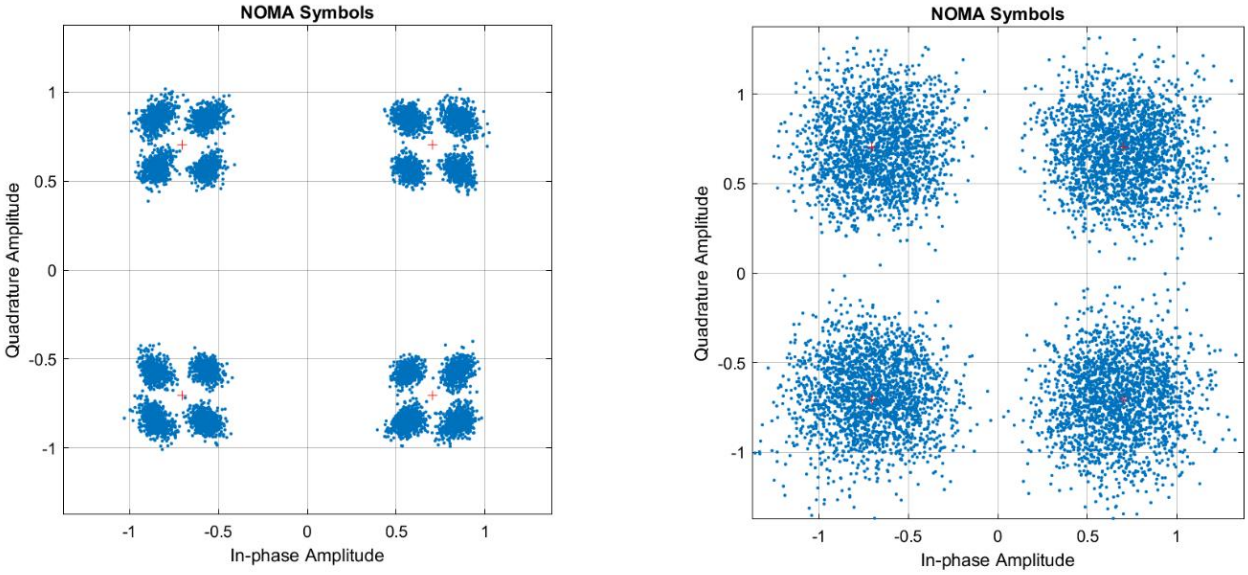

Σχήμα 4.3: Φάσμα παραμορφωμένο από διαλείψεις

Όταν το κανάλι επηρεάζει διαφορετικά τις συχνότητες του φάσματος είναι επιλεκτικό στη συχνότητα. Ένα παραμορφωμένο φάσμα προκαλεί παραμόρφωση και στο διάγραμμα αστερισμού. Το διάγραμμα αστερισμού αποτελείται από τα ισοσταθμισμένα σύμβολα πληροφορίας. Στο Σχήμα 4.4 φαίνεται ένα σήμα QPSK που λήφθηκε από το Adalm Pluto SDR. Το QPSK έχει 4 διαφορετικές καταστάσεις. Οι διαλείψεις και οι απώλειες διάδοσης μπορεί να παραμορφώσουν έτσι το διάγραμμα αστερισμού ώστε να αλλάξουν οι καταστάσεις και να δημιουργηθούν λάθη στα σύμβολα πληροφορίας, που θα επηρεάσουν άμεσα και τα bit.



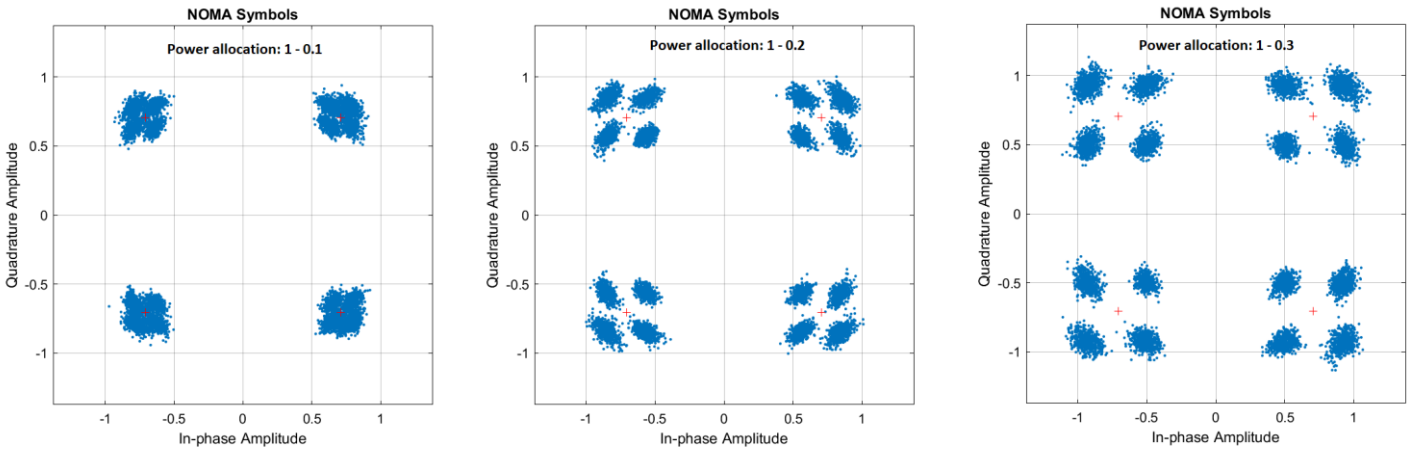

Σχήμα 4.4: Αριστερά το φάσμα, δεξιά το διάγραμμα αστερισμού

Το επιλεκτικό κανάλι έχει προσθέσει μια απόκλιση στη φάση στο διάγραμμα αστερισμού, επίσης παρατηρείται και μια διασκόρπιση των συμβόλων στο επίπεδο λόγω της χαμηλής σηματοθορυβικής σχέσης. Στο παρακάτω σχήμα φαίνονται δύο διαφορετικά διαγράμματα αστερισμού με διαφορά ως προς το SNR.

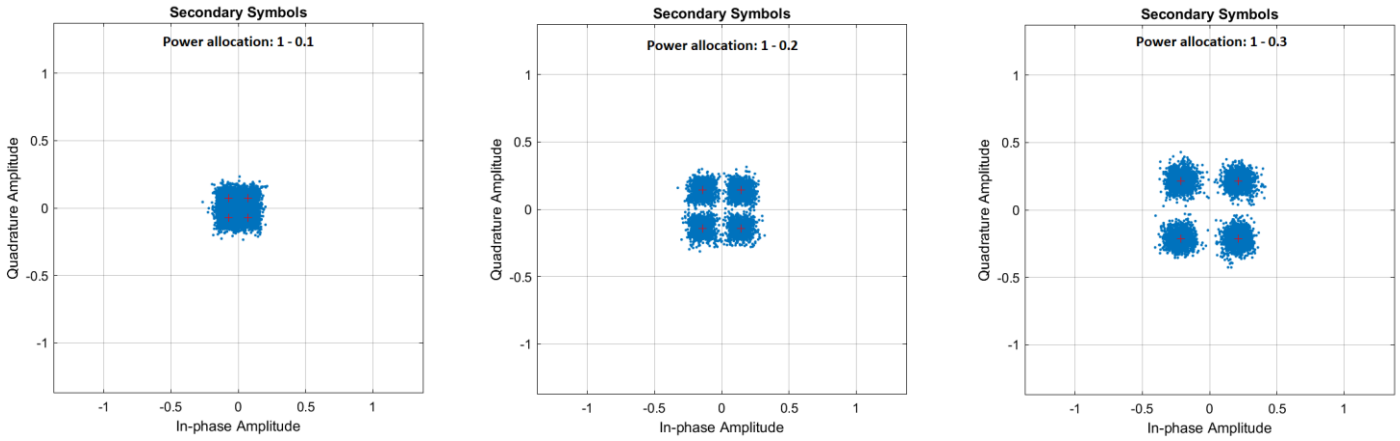



Σχήμα 4.5: Διαγράμματα αστερισμού με διαφορετικό SNR

Στο αριστερό μέρος του Σχήματος 4.5 βλέπουμε ένα υπερτεθειμένο σήμα που έχει λάβει ο δέκτης του SDR. Είναι το ίδιο που φαίνεται θεωρητικά στο Σχήμα 2.7. Αυτό το σήμα λαμβάνουν οι δύο δέκτες των Secondary και Primary. Ο Primary αποδιαμορφώνει αυτό καθαυτό το λαμβανόμενο σήμα αγνοώντας τη παρεμβολή του Secondary. Βλέπουμε από το σχήμα 4.5, ότι ακόμα και στη περίπτωση που φαίνεται δεξιά, όπου το σήμα του Secondary δεν είναι ορατό, οι τέσσερις καταστάσεις του QPSK σήματος του Primary ξεχωρίζουν. Επομένως δε θα έχει πρόβλημα με την αποδιαμόρφωση. Ο Secondary με βάση τη θεωρία διενεργεί τη διαδικασία της ακύρωσης της παρεμβολής για να εξάγει τα σύμβολα πληροφορίας. Στα επόμενα δύο σχήματα φαίνεται στο δέκτη του SDR τα διαγράμματα αστερισμού για διαφορετικές κατανομές πλατών (power allocation). Στο Σχήμα 4.6 φαίνονται τα διαγράμματα αστερισμού του Primary και στο Σχήμα 4.7 τα αντίστοιχα του Secondary μετά την ακύρωση της παρεμβολής.

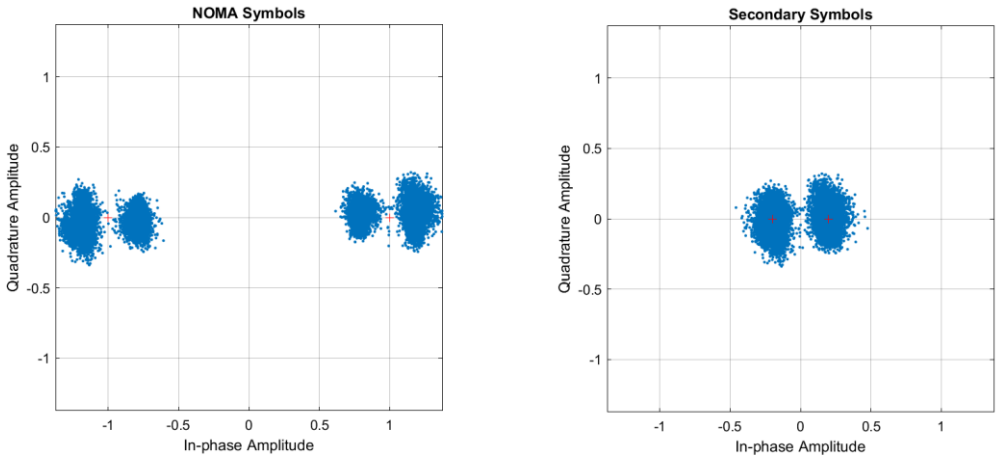



Σχήμα 4.6: Διαγράμματα αστερισμού Primary με διαφορετικές κατανομές πλατών



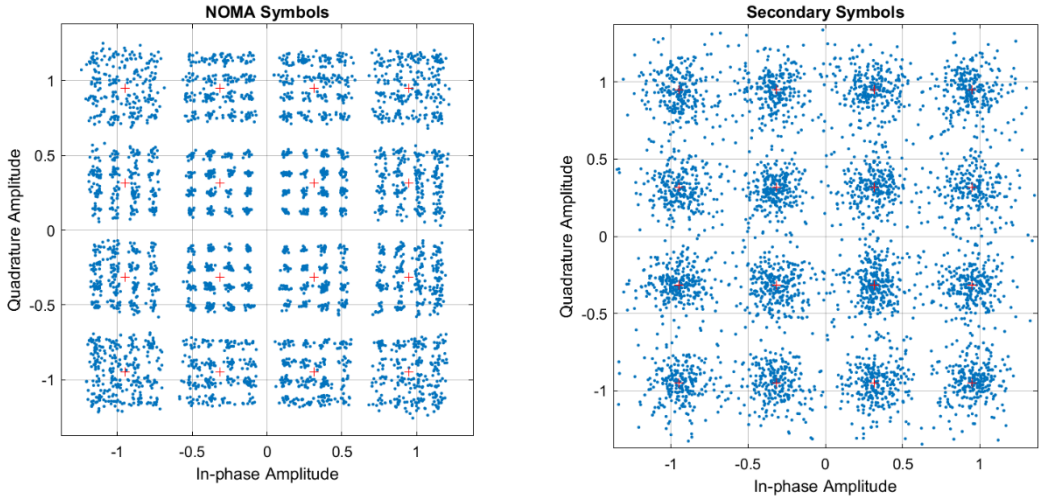

Σχήμα 4.7: Διαγράμματα αστερισμού Secondary με διαφορετικές κατανομές πλατών

Επιπρόσθετα με το σχήμα διαμόρφωσης QPSK που αναλύθηκε στα παραπάνω σχήματα χρησιμοποιήθηκαν και τα BPSK και 16QAM. Συνολικά στα πειράματα χρησιμοποιήθηκαν οι παρακάτω συνδυασμοί υπέρθεσης στο πεδίο της ισχύος, BPSK + BPSK, QPSK + QPSK, 16QAM + 16QAM. Με την ίδια λογική μπορεί να προστεθεί ένα QPSK με ένα 16QAM. Το τελικό σήμα που προκύπτει όταν προσθέτουμε δύο σήματα διαμορφωμένα κατά BPSK φαίνεται στο Σχήμα 4.8 αριστερά.




Σχήμα 4.8: Διάγραμμα αστερισμού υπέρθεσης δύο BPSK σημάτων

Η υπέρθεση ακολουθεί την ίδια λογική με όλα τα σχήματα διαμόρφωσης. Στο παραπάνω σχήμα, φαίνεται από αριστερό διάγραμμα αστερισμού, το σήμα του Secondary που «κάθεται» πάνω στο σήμα του Primary, όπως γίνεται και στη περίπτωση του QPSK που αναλύθηκε στο δεύτερο κεφάλαιο. Στο Σχήμα 4.9 φαίνεται η υπέρθεση δύο 16QAM που είναι ο τρίτος και τελευταίος συνδυασμός διαμορφώσεων που χρησιμοποιήθηκε.



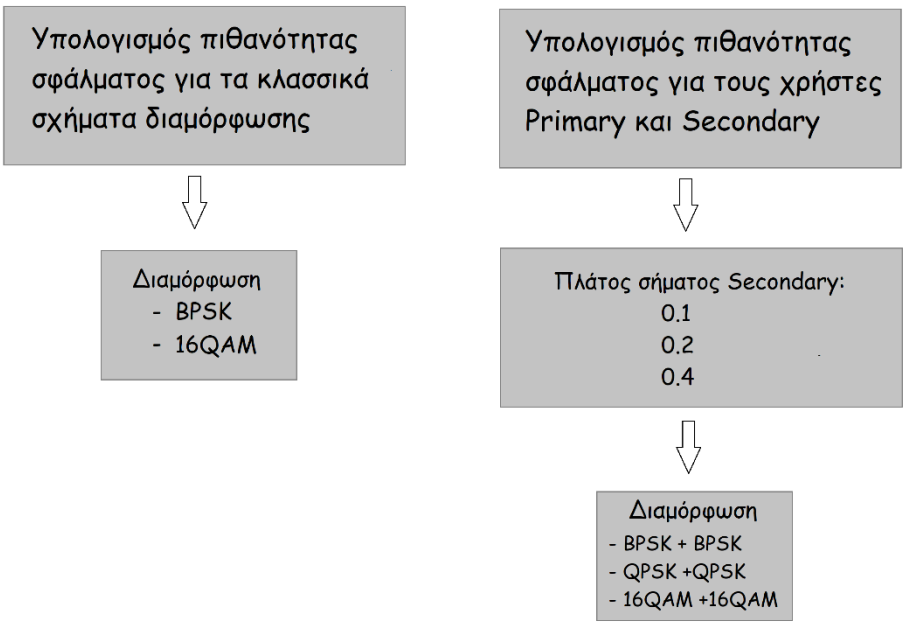

Σχήμα 4.9: Διάγραμμα αστερισμού υπέρθεσης δύο 16QAM σημάτων

Παρατηρούμε το υπερτεθειμένο σήμα στο αριστερό μέρος του Σχήματος 4.9 μοιάζει με ένα 64QAM λόγω της παρεμβολής από το σήμα του Secondary. Παρόλα αυτά δεν πρέπει να συγχέεται με αυτό, οι καταστάσεις των συμβόλων είναι 16 και φαίνονται από τα κόκκινα σημεία πάνω στο σχήμα. Στο δεξί μέρος του σχήματος φαίνεται το σήμα του Secondary όπως προέκυψε μετά την ακύρωση της παρεμβολής (IC). Η διασκόρπιση που υπάρχει στα σύμβολα είναι αποτέλεσμα του καναλιού και του θορύβου. Όσο μεγαλώνει το σχήμα διαμόρφωσης τόσο αυξάνεται και η πιθανότητα σφάλματος bit στο δέκτη, πόσο μάλλον στη περίπτωση που μελετάμε στην οποία έχουμε υπέρθεση σημάτων. Γίνεται αντιληπτό ότι ένα PD – NOMA σύστημα έχει κάποια όρια όσον αφορά την υπέρθεση σημάτων σε

μεγάλα σχήματα διαμόρφωσης. Αυτό συνδέεται με τις ικανότητες του διαθέσιμου εξοπλισμού ενός συστήματος. Στην επόμενη παράγραφο γίνεται μια μελέτη ως προς την πιθανότητα σφάλματος bit του συστήματος. Συγκρίνονται διαφορετικά σχήματα διαμόρφωσης και κατανομές ισχύος για τους δέκτες των Primary και Secondary.

## 4.2 Πιθανότητα σφάλματος

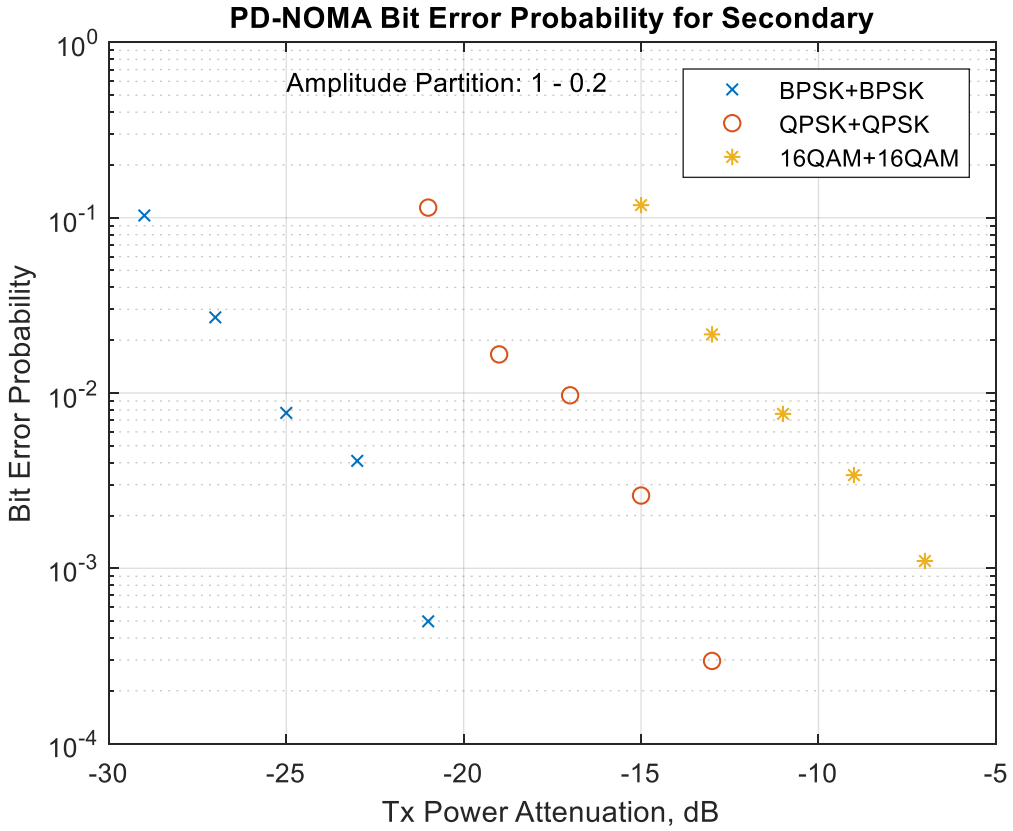
Ένα από τα βασικά χαρακτηριστικά των τηλεπικοινωνιακών συστημάτων είναι η πιθανότητα σφάλματος. Μέσω αυτής αξιολογούνται τα συστήματα ως προς την αξιοπιστία τους. Για να είναι μια ζεύξη επιτυχής πρέπει να μπορεί να μεταφερθεί πληροφορία με το λιγότερο δυνατό αριθμό λαθών. Η πιθανότητα σφάλματος συνδέεται με τη σηματοθορυβική σχέση στο δέκτη. Όσο χειρότερο SNR έχουμε τόσο αυξάνεται και η πιθανότητα σφάλματος. Στο σύστημα που δημιουργήθηκε για τη μελέτη της τεχνικής PD – NOMA λήφθηκαν αποτελέσματα για τους δύο χρήστες του συστήματος για διαφορετικές καταστάσεις. Χρησιμοποιήθηκαν τρεις διαφορετικές κατανομές του πλάτους του Secondary, 0.1, 0.2 και 0.4, επίσης οι τρείς συνδυασμοί διαμόρφωσης BPSK+BPSK, QPSK+QPSK, 16QAM+16QAM. Στο τέλος αυτά τα αποτελέσματα του PD – NOMA συγκρίθηκαν με αποτελέσματα που λήφθηκαν για κλασσικά σχήματα διαμόρφωσης. Για τη καλύτερη κατανόηση της προσέγγισης που πραγματοποιήθηκε παρατίθεται το παρακάτω διάγραμμα όπου φαίνεται η στρατηγική με την οποία λήφθηκαν τα αποτελέσματα.




Σχήμα 4.10: Στρατηγική λήψης αποτελεσμάτων

Για την μέτρηση της πιθανότητας σφάλματος δημιουργήσαμε μια περιπτωσιολογική μελέτη (case study). Σε αυτή χρησιμοποιήθηκε ο εξασθενητής σήματος του προγραμματιστικού εργαλείου της MATLAB “sdrtx”. Το εργαλείο αυτό το είδαμε στο τρίτο κεφάλαιο στη παρουσίαση του κώδικα. Η παράμετρος εξασθένησης ονομάζεται “Tx Gain” και μέσω αυτής μπορεί να αλλάξει η ισχύς εκπομπής του Adalm Pluto. Αυξομειώνοντας την ισχύ εκπομπής δημιουργούμε διαφορετικές καταστάσεις του SNR. Ουσιαστικά, με τη χρήση του εξασθενητή προσομοιώνουμε τις μεταβολές του SNR που έχει ένας χρήστης μέσα σε ένα τηλεπικοινωνιακό σύστημα όπως για παράδειγμα μια κινητή συσκευή μέσα σε

μια κυψέλη. Όπως ειπώθηκε στο τρίτο κεφαλαίο εμείς μετράμε το MER (Modulation Error Ratio) αντί για το SNR. Το MER εκφράζεται από τη σχέση 3.9 και συμπίπτει με το SNR όταν έχουμε AWGN κανάλι [43] [44]. Είναι μια αξιόπιστη μετρική που μπορεί να μη συμπίπτει πάντα με τη σηματοθορυβική σχέση, αλλά μπορεί να χρησιμοποιηθεί για την αξιολόγηση ενός συστήματος. Φάνηκε πως όσο μικραίνουμε την ισχύ εκπομπής, τόσο μικραίνει και το MER πράγμα που είναι φυσιολογικό. Έχοντας εξηγήσει τη σχέση του MER με το SNR, αλλά και τη λειτουργία του εξασθενητή, μπορούμε να παρουσιάσουμε τα αποτελέσματα ως προς τη πιθανότητα σφάλματος με βάση το σχήμα 4.10. Για να λάβουμε ένα ασφαλές αποτέλεσμα στείλαμε για κάθε περίπτωση εξασθένησης από δεκάδες χιλιάδες bit έως μερικά εκατομμύρια, φροντίζοντας να γίνουν τουλάχιστον μερικές εκατοντάδες λάθη έτσι ώστε το αποτέλεσμα να είναι ασφαλές.

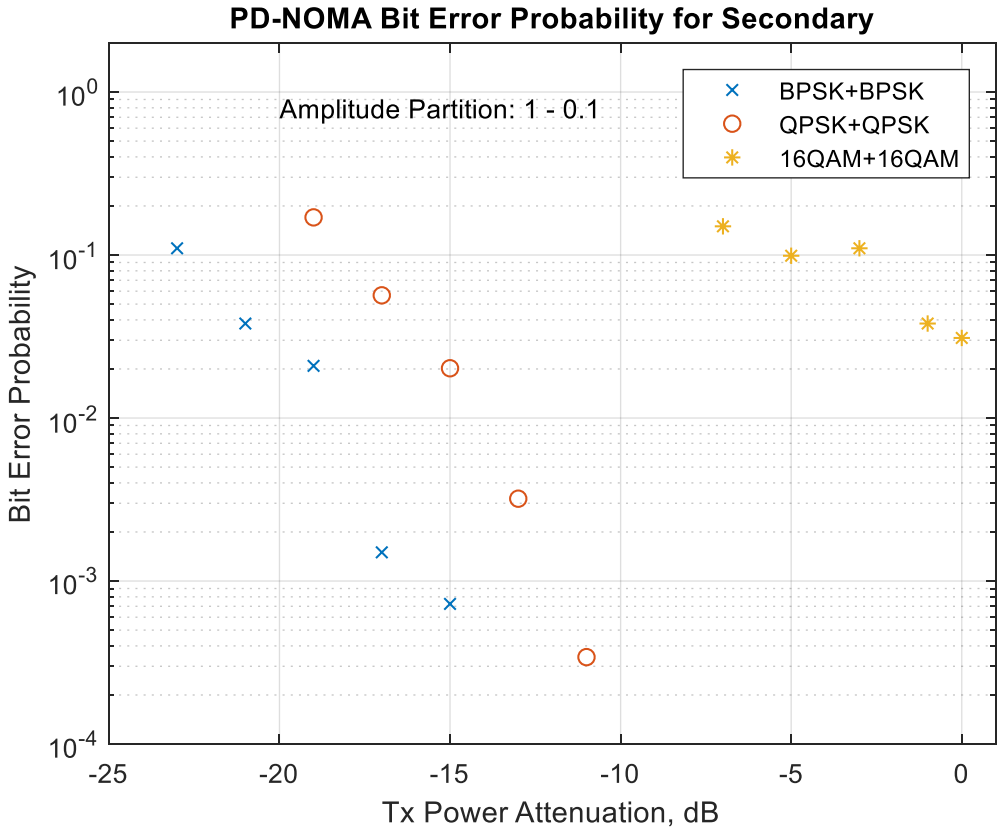



Σχήμα 4.11: Πιθανότητα σφάλματος έναντι εξασθένησης ισχύος για 0.2 κατανομή πλατών

Να σημειωθεί εδώ ότι το Adalm Pluto έχει μέγιστη ισχύ εκπομπής 5dBm [45]. Όταν για παράδειγμα επιλέγουμε εξασθένηση -15dB τότε η ισχύς εκπομπής είναι το άθροισμα των δύο, δηλαδή -10dBm. Επειδή η μέγιστη ισχύ εκπομπής δεν είναι ακριβής μέτρηση και με βάση τον κατασκευαστή μπορεί να διαφέρει από συσκευή σε συσκευή, όλη η ανάλυση θα γίνει με βάση την εξασθένηση. Στο Σχήμα 4.11 βλέπουμε τη πιθανότητα σφάλματος bit σε σχέση με την εξασθένηση της ισχύος εκπομπής για τον Secondary χρήστη. Θυμίζουμε ότι για να ανιχνευθεί το σήμα του Secondary πρέπει πρώτα να ακυρωθεί η παρεμβολή που προκαλείται από το σήμα του Primary. Στη συγκεκριμένη περίπτωση το συνολικό υπερτεθειμένο σήμα αποτελείται από το 0.2 πλάτος του Secondary συν μονάδα που είναι το πλάτος του Primary. Για την εξαγωγή των αποτελεσμάτων τα SDR τοποθετήθηκαν στα 30 εκατοστά απόσταση μεταξύ τους. Στο σχήμα φαίνονται τα αποτελέσματα για τους τρεις συνδυασμούς διαμόρφωσης. Τα καλύτερα αποτελέσματα λαμβάνονται για τη διαμόρφωση BPSK. Από τη θεωρία ισχύει ότι όσο μεγαλώνει το σχήμα διαμόρφωσης τόσο μεγαλώνει και ο ρυθμός σφαλμάτων. Πιο

συγκεκριμένα, από το Σχήμα 4.11 φαίνεται ότι πετυχαίνουμε πιθανότητα σφάλματος περίπου $10^{-3}$ με -7dB εξασθένηση για διαμόρφωση 16QAM. Ενώ για BPSK την ίδια πιθανότητα σφάλματος τη πετυχαίνουμε με -23dB περίπου εξασθένηση. Γίνεται αντιληπτό ότι χρειαζόμαστε μεγαλύτερη ισχύ εκπομπής για να εξυπηρετήσουμε έναν χρήστη με 16QAM πετυχαίνοντας την ίδια πιθανότητα σφάλματος με αυτήν του BPSK. Η πιθανότητα σφάλματος για την υπέρθεση δύο QPSK σημάτων βρίσκεται ανάμεσα από τις άλλες δύο, πράγμα αναμενόμενο με βάση τη θεωρία.

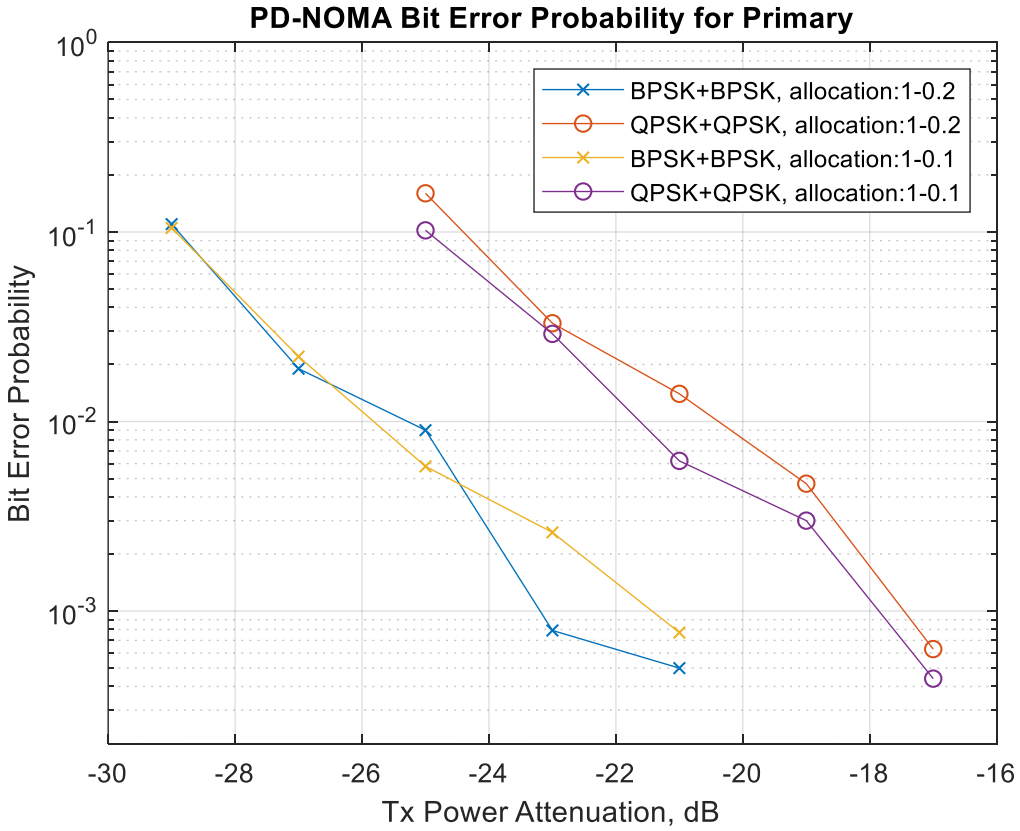
Στη συνέχεια λήφθηκαν αποτελέσματα αναθέτοντας στον Secondary χρήστη πλάτος 0.1. Στόχος είναι να δούμε ποια κατανομή πλατών μπορεί να πετύχει καλύτερα αποτελέσματα. Το γράφημα του Σχήματος 4.12 δείχνει τα αποτελέσματα.




Σχήμα 4.12: Πιθανότητα σφάλματος έναντι εξασθένησης ισχύος για 0.1 κατανομή πλατών

Όπως φαίνεται οι καμπύλες για 0.1 και 0.2 πλάτος στο σήμα του Secondary έχουν παρόμοια συμπεριφορά όσον αφορά τα διαφορετικά σχήματα διαμόρφωσης. Δηλαδή λαμβάνουμε τα καλύτερα αποτελέσματα για BPSK και τα χειρότερα για 16QAM. Όμως αν συγκρίνουμε τα Σχήματα 4.11 και 4.12 παρατηρούμε ότι χρειαζόμαστε μεγαλύτερη ισχύ εκπομπής για όλους τους συνδυασμούς διαμόρφωσης στη περίπτωση που ο Secondary έχει πλάτος 0.1. Για παράδειγμα, ας πάρουμε τη περίπτωση που έχουμε υπέρθεση δύο BPSK. Με πλάτος 0.1 πετυχαίνουμε πιθανότητα σφάλματος $10^{-3}$ στα -16dB εξασθένηση, ενώ με πλάτος 0.2 στα -22dB. Επίσης παρατηρούμε πως στη περίπτωση που το πλάτος του Secondary είναι 0.1 και ο συνδυασμός διαμόρφωσης 16QAM, ακόμα και χωρίς εξασθένηση (0dB) ο δέκτης του Secondary δε μπορεί να πετύχει πιθανότητα μικρότερη από $10^{-3}$. Φαίνεται από τα παραπάνω ότι το σύστημα για 16QAM υπέρθεση και 1 - 0.1 κατανομή πλάτων φτάνει στο όριο του μη μπορώντας να πετύχει μια θεωρητικά καλή πιθανότητα σφάλματος, μικρότερη του $10^{-3}$.

Είναι φυσιολογικό ο δέκτης του Secondary να λειτουργεί καλύτερα όσο αυξάνουμε το πλάτος του. Όμως αυτό προκαλεί μεγαλύτερη παρεμβολή στο δέκτη του Primary. Όσο μεγαλώνει το ποσοστό της ισχύος του Secondary τόσο χειροτερεύει η επίδοση του Primary. Αυτό φαίνεται από τα σχήματα 4.6 και 4.7 όπου αναθέτουμε 0.3 πλάτος στον Secondary. Από τη μία ο δέκτης του Secondary φαίνεται από το διάγραμμα αστερισμού ότι πετυχαίνει καλύτερο SNR, αλλά ταυτόχρονα η παρεμβολή του Secondary μειώνει το SINR του Primary. Στο Σχήμα 4.13 φαίνεται η πιθανότητα σφάλματος για τον δέκτη του Primary για συνδυασμούς διαμορφώσεων BPSK και QPSK. Η αύξηση του πλάτους του Secondary χρήστη μεγαλώνει τη παρεμβολή στον Primary με αποτέλεσμα να μειώνεται η σηματοθορυβική του σχέση και να αυξάνεται η πιθανότητα σφάλματος. Τα αποτελέσματα για τον δέκτη του Primary λήφθηκαν για απόσταση 1.6 μέτρα.



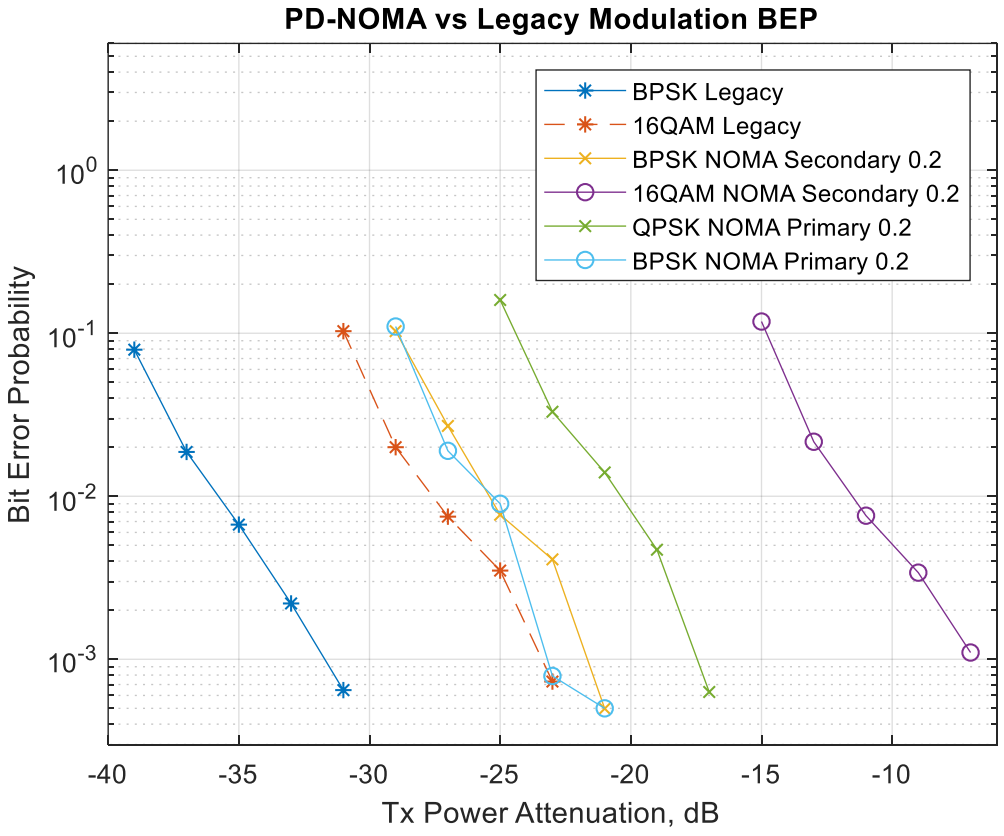

Σχήμα 4.13: Πιθανότητα σφάλματος bit του Primary χρήστη

Φαίνεται από τα παραπάνω ότι ο δέκτης του Primary πετυχαίνει παρόμοια πιθανότητα σφάλματος και για συνδυασμό διαμορφώσεων BPSK. Ενώ για QPSK φαίνεται στη περίπτωση που η κατανομή των πλατών είναι 1-0.1 να έχει ελαφρώς καλύτερη επίδοση. Με βάση τα παραπάνω φαίνεται πως και οι δύο κατανομές πλατών έχουν παρόμοια συμπεριφορά.

Από τα παραπάνω παρατηρούμε ότι σε ένα σύστημα PD – NOMA πρέπει να γίνει ένας συμβιβασμός όσον αφορά την κατανομή της ισχύος. Δε πρέπει να αυξήσουμε πολύ την ισχύ του Secondary διότι έτσι αυξάνουμε και τη παρεμβολή στον Primary. Από την άλλη πρέπει να αναθέσουμε στον Secondary την ισχύ που χρειάζεται, ανάλογα με τη σηματοθορυβική του σχέση, για να λειτουργήσει ικανοποιητικά η ακύρωση της παρεμβολής. Σε ένα πραγματικό σύστημα αυτή είναι δουλεία του σταθμού βάσης κινητής τηλεφωνίας. Πρέπει να βρει από τους χρήστες που έχει μέσα στη

κυψέλη αυτούς που μπορεί να τους χωρίσει σε ζευγάρια έτσι ώστε οι δέκτες να μπορούν να διαχειριστούν ο ένας τη παρεμβολή του άλλου.

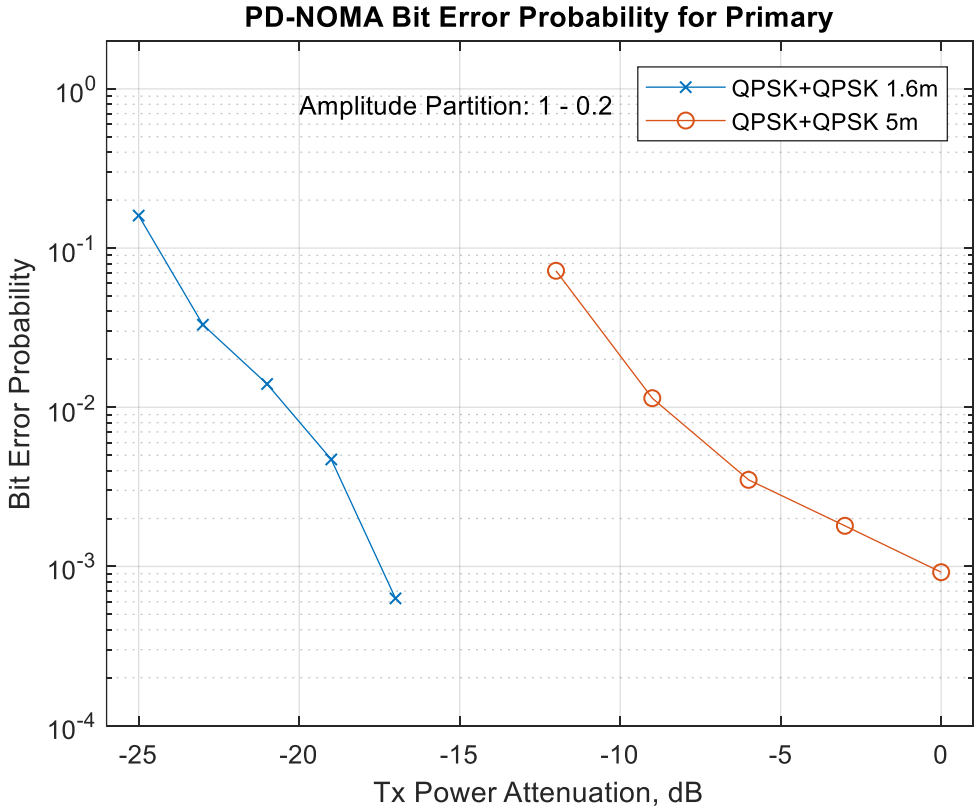
Επίσης λήφθηκαν αποτελέσματα για τα κλασσικά σχήματα διαμόρφωσης QPSK και 16QAM και συγκρίθηκαν με τη τεχνική PD – NOMA. Τα αποτελέσματα φαίνονται στο Σχήμα 4.14. Τα κλασσικά σχήματα διαμόρφωσης πετυχαίνουν ακόμα καλύτερη πιθανότητα σφάλματος πράγμα που είναι λογικό διότι σε αυτή την περίπτωση δεν υπάρχει υπέρθεση άρα και παρεμβολή των δύο χρηστών μεταξύ τους. Συγκεκριμένα στη περιπτωσιολογική μελέτη μπορέσαμε να μειώσουμε την ισχύ εκπομπής μέχρι και 39dB για διαμόρφωση BPSK. Η σύγκριση έγινε για να μπορέσουμε να δούμε τις σχέσεις μεταξύ των απλών διαμορφώσεων και των διαμορφώσεων υπέρθεσης. Ωστόσο είναι σημαντικό να τονιστεί ότι στην περίπτωση του PD – NOMA ο συνολικός ρυθμός μετάδοσης πληροφορίας από το σταθμό βάσης προς τους δύο χρήστες διπλασιάζεται (όταν και οι δύο χρήστες χρησιμοποιούν το ίδιο σχήμα διαμόρφωσης) σε σχέση με το ρυθμό πληροφορίας που μεταδίδεται σε ένα χρήστη. Ως εκ τούτου, με την κατάλληλη επιλογή των χρηστών (ως προς τη σηματοθορυβική τους σχέση) που θα υπερτεθούν, μπορεί συνολικά να επιτευχθεί αποδοτικότερη εκμετάλλευση των πόρων συχνότητας και χρόνου.




Σχήμα 4.14: Σύγκριση PD - NOMA με κλασσικές διαμορφώσεις

Τα παραπάνω αποτελέσματα λήφθηκαν με δύο SDR και έναν υπολογιστή. Το μήκος των καλωδίων περιόριζαν την μέγιστη απόσταση μεταξύ τους στα 1.6 μέτρα. Στη προσπάθειά μας να δημιουργήσουμε ένα σύστημα με περισσότερες δυνατότητες χρησιμοποιήσαμε δύο υπολογιστές, προγραμματίζοντας τον έναν ως πομπό και τον άλλον ως δέκτη. Έτσι μπορέσαμε να αυξήσουμε την απόσταση μεταξύ των SDR. Πάρθηκαν αποτελέσματα για τον χρήστη Primary για συνδυασμό διαμορφώσεων QPSK και κατανομή πλατών 1 – 0.2. Τα αποτελέσματα φαίνονται στο σχήμα 4.15. Στη κόκκινη γραμμή βλέπουμε την επίδοση του Primary στα 5 μέτρα. Είναι προφανές ότι χρειάστηκε περισσότερη ισχύ για να εξυπηρετηθεί με μια ικανοποιητική πιθανότητα σφάλματος, πιο συγκεκριμένα

από -12dB έως 0dB της συνολικής διαθέσιμης. Για σύγκριση, στο γράφημα προστέθηκε η αντίστοιχη περίπτωση για 1.6 μέτρα. Η αύξηση της απόστασης αυξάνει τις απώλειες διάδοσης και τις διαλείψεις που προκαλούνται από τη πολυδιαδρομική διάδοση του σήματος. Το αποτέλεσμα είναι η μείωση της σηματοθορυβικής σχέσης στο δέκτη.



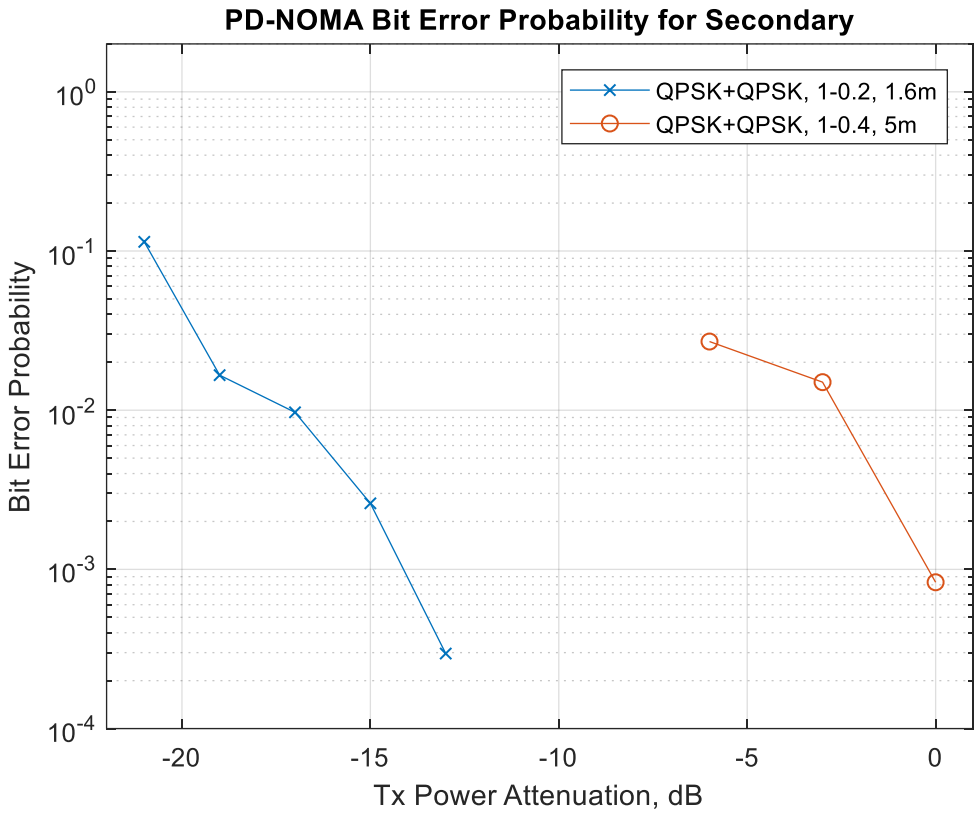

Σχήμα 4.15: Πιθανότητα σφάλματος έναντι εξασθένηση ισχύος για 5 μέτρα απόσταση για τον Primary χρήστη

Η τελευταία περίπτωση που δοκιμάστηκε ήταν η λειτουργία του δέκτη του Secondary για 5 μέτρα απόσταση μεταξύ των SDR. Για να λειτουργήσει έπρεπε να αυξηθεί το πλάτος του στο 0.4. Τα αποτελέσματα φαίνονται στο Σχήμα 4.16. Η κόκκινη γραμμή περιγράφει την επίδοση του. Όπως φαίνεται για να λειτουργήσει έπρεπε να αυξηθεί η ισχύς εκπομπής από τα -6dB έως τα 0dB της συνολικής. Για σύγκριση παρατίθεται στο γράφημα η περίπτωση που τα SDR είναι στα 30 εκατοστά απόσταση με κατανομή πλατών 1-0.2. Στην επόμενη παράγραφο γίνεται μια σύνοψη του τέταρτου κεφαλαίου.

## 4.3 Επίλογος τέταρτου κεφαλαίου

Στο τέταρτο κεφάλαιο παρουσιάστηκαν τα αποτελέσματα του πειράματος. Αρχικά αναλύθηκε η λήψη ενός σήματος από το Adalm Pluto SDR με βάση το φάσμα, δηλαδή τη λαμβανόμενη ισχύ ανά μονάδα hertz (dBm/Hz). Μέσω αυτού, μπορούμε να δούμε πότε ένα σήμα είναι ανιχνεύσιμο. Στη συνέχεια, παρατέθηκαν διαγράμματα αστερισμού για διαφορετικά σχήματα διαμόρφωσης και κατανομές ισχύος που προέκυψαν από το πείραμα. Από αυτά, είδαμε γραφικά πως φαίνεται το υπερτεθειμένο σήμα στο δέκτη του SDR, πριν και μετά την ακύρωση της παρεμβολής (IC), αλλά και πως η κατανομή των πλατών αλλάζει τα χαρακτηριστικά του λαμβανόμενου σήματος. Έπειτα, παρουσιάστηκαν τα αποτελέσματα ως προς τη πιθανότητα σφάλματος bit. Από αυτά έγινε αντιληπτή η σημαντικότητα του καταμερισμού της ισχύος σε ένα σύστημα PD – NOMA και ο συμβιβασμός που

πρέπει να γίνει για να έχουμε ένα αποδοτικό σύστημα. Πιο συγκεκριμένα, πρέπει να αναθέσουμε στον Secondary χρήστη τόση ισχύ ώστε να λειτουργεί με μια ικανοποιητική πιθανότητα σφάλματος. Από τη άλλη, το ποσοστό της ισχύος που αναθέτουμε στον Secondary θα πρέπει να μπορεί να το διαχειριστεί ο Primary. Επιπρόσθετα, λήφθηκαν αποτελέσματα για τα κλασσικά σχήματα διαμόρφωσης BPSK και 16QAM και συγκρίθηκαν με αυτά της υπερθετικής διαμόρφωσης. Φάνηκε ότι για την επιτυχής λήψη ενός υπερτεθειμένου σήματος χρειαζόμαστε περισσότερη ισχύ, πράγμα λογικό εφόσον στο παρονομαστή της σηματοθορυβικής σχέσης προστίθεται και η παρεμβολή μεταξύ των χρηστών. Η διαδικασία του πειράματος έδειξε ότι η τεχνική PD – NOMA είναι εφικτή, καταδεικνύοντας παράλληλα τη σχέση των υπερτεθειμένων σχημάτων διαμόρφωσης τόσο μεταξύ τους όσο και με τα κλασσικά σχήματα. Τέλος, φάνηκε η σημαντικότητα της κατανομής της ισχύος στο σχεδιασμό ενός συστήματος PD - NOMA. Στο επόμενο κεφάλαιο γίνεται μια σύνοψη της εργασίας και θα αναλυθούν τα συμπεράσματα.




Σχήμα 4.16: Πιθανότητα σφάλματος έναντι εξασθένηση ισχύος για 5 μέτρα απόσταση για τον Secondary χρήστη

# Κεφάλαιο 5ο: Σύνοψη και συμπεράσματα

Στη παρούσα διπλωματική εργασία υλοποιήθηκε η τεχνική της μη ορθογωνικής πολλαπλής πρόσβασης στο επίπεδο της ισχύος (PD – NOMA). Μια τεχνική που μπορεί να εξοικονομήσει πόρους (συχνότητα και χρόνο) σε ένα τηλεπικοινωνιακό σύστημα. Είναι μια τεχνική που μελετάται αυτή τη περίοδο από τη διεθνή επιστημονική κοινότητα και προορίζεται για δίκτυα κινητών επικοινωνιών 5ης γενιάς αλλά και ως βελτίωση στα ήδη υπάρχοντα 4ης γενιάς. Εν προκειμένω μελετήθηκε η τεχνική στο πλαίσιο της κάτω ζεύξης (downlink), δηλαδή όταν ο σταθμός βάσης επικοινωνεί με τους κινητούς χρήστες. Η βασική ιδέα είναι πως ένας σταθμός βάσης κινητής τηλεφωνίας επιλέγει δύο χρήστες, τον Primary και τον Secondary. Ο πρώτος βρίσκεται στην άκρη της κυψέλης έχοντας κακή σηματοθορυβική σχέση και ο δεύτερος κοντά στο σταθμό βάσης με καλή σηματοθορυβική σχέση. Από τη συνολική διαθέσιμη ισχύ ο σταθμός βάσης αναθέτει ένα μεγάλο ποσοστό της ισχύος στον Primary και όσο περισσεύει στον Secondary. Το τελικό σήμα προκύπτει από την υπέρθεση των δύο σημάτων και αποστέλλεται μέσω του καναλιού [3][4][8] . Οι δέκτες των χρηστών λαμβάνουν το ίδιο υπερτεθειμένο σήμα και καλούνται μέσα από αυτό να εξάγουν την πληροφορία που τους αντιστοιχεί. Ο δέκτης του Primary έχοντας το μεγάλο ποσοστό της ισχύος αποδιαμορφώνει και ανιχνεύει το λαμβανόμενο σήμα αγνοώντας τη παρεμβολή του Secondary. Από την άλλη ο Secondary για να βρει το σήμα του μέσα στο συνολικό πρέπει να ακυρώσει τη παρεμβολή του Primary. Η διαδικασία της ακύρωσης της παρεμβολής έχει δύο στάδια. Ο δέκτης του Secondary πρέπει πρώτα να ανιχνεύσει το σήμα του Primary και να το αφαιρέσει από το συνολικό λαμβανόμενο. Αποδιαμορφώνοντας το αποτέλεσμα της αφαίρεσης ο Secondary καταλήγει με τη πληροφορία που προορίζεται για αυτό [5][8]. Μέσω της τεχνικής PD – NOMA μπορεί ο σταθμός βάσης να στείλει ταυτόχρονα και στην ίδια συχνότητα δύο πληροφορίες υπερτεθειμένες στην ισχύ εξοικονομώντας πόρους.

Για τη μελέτη της τεχνικής χρησιμοποιήθηκαν μονάδες ραδιοεπικοινωνίας καθορισμένες από λογισμικό (SDR) τύπου Adalm Pluto της εταιρείας Analog Devices. Πρόκειται εργαλεία που μπορούν να στείλουν και να λάβουν σήματα μέσω του μικροεπεξεργαστή και του πομποδέκτη που διαθέτουν. Για τον προγραμματισμό τους, χρησιμοποιήθηκε το λογισμικό της MATLAB. Αρχικά προγραμματίστηκε ένα απλό σύστημα εκπομπής και λήψης με το οποίο δεν επιτεύχθηκε καλό SNR, το οποίο είναι απαραίτητο για τη σωστή λειτουργία του δέκτη του Secondary χρήστη. Για αυτό χρησιμοποιήθηκε η εργαλειοθήκη της MATLAB “WLAN system toolbox”. Παρόλο που τα συστήματα WLAN δεν είναι κινητών επικοινωνιών, μας έδωσαν τη δυνατότητα να υλοποιήσουμε τη τεχνική. Με τη χρήση των συναρτήσεων της εργαλειοθήκης επιτεύχθηκε συγχρονισμός στο χρόνο και στη συχνότητα του λαμβανόμενου σήματος αυξάνοντας έτσι τη σηματοθορυβική σχέση. Επίσης το WLAN μας έδωσε τη δυνατότητα να χρησιμοποιήσουμε κώδικα διόρθωσης λαθών καθώς και τη τεχνική OFDM.

Με τα παραπάνω εργαλεία υλικού και λογισμικού έγινε μια περιπτωσιολογική μελέτη όπου το πρώτο SDR λειτουργεί ως σταθμός βάσης και το δεύτερο σαν χρήστης. Επίσης χρησιμοποιήθηκε ο εξασθενητής σήματος της MATLAB έτσι ώστε να προσομοιωθούν διαφορετικές καταστάσεις σηματοθορυβικής σχέσης στο δέκτη. Δηλαδή μειώθηκε σταδιακά η ισχύς εκπομπής έτσι ώστε να δημιουργηθούν λάθη στο δέκτη και να μετρήσουμε τη πιθανότητα σφάλματος. Αρχικά υπολογίστηκε η πιθανότητα σφάλματος bit για τους χρήστες Primary και Secondary για διάφορους συνδυασμούς διαμόρφωσης και κατανομής ισχύος. Στη συνέχεια, λήφθηκαν αποτελέσματα για κλασσικά σχήματα διαμόρφωσης και συγκρίθηκαν με αυτά της υπερτεθειμένης διαμόρφωσης.

Από τα αποτελέσματα συμπεραίνουμε ότι ένα σύστημα μη ορθογωνικής πολλαπλής πρόσβασης στο επίπεδο της ισχύος είναι εφαρμόσιμο σε πραγματικές συνθήκες και μπορεί με την κατάλληλη παραμετροποίηση να πετύχει καλύτερη διαχείριση των πόρων χρόνου και συχνότητας. Φάνηκε ότι η κατανομή της ισχύος είναι ο πιο σημαντικός παράγοντας λειτουργίας του συστήματος. Ο σταθμός βάσης πρέπει να επιλέξει σωστά τους χρήστες τους οποίους θα υπερθέσει ανάλογα με τη σηματοθορυβική τους σχέση. Στη συνέχεια πρέπει να βρεθεί ένας συμβιβασμός μεταξύ των ισχύων των δύο χρηστών. Από τη μία, ο Secondary πρέπει να λάβει ένα ικανοποιητικό ποσοστό έτσι ώστε να μπορεί να πετύχει μια επιθυμητή πιθανότητα σφάλματος. Από την άλλη, αυτό το ποσοστό αυτό πρέπει να μπορεί να το διαχειριστεί ο Primary ως παρεμβολή. Από το πείραμα καταδεικνύεται επίσης η διαφορά της επίδοσης ανάλογα με το σχήμα διαμόρφωσης που χρησιμοποιείται. Όπως και σε ένα σύστημα κλασσικών διαμορφώσεων μπορούμε να αυξήσουμε το ρυθμό μετάδοσης των δεδομένων μεγαλώνοντας το σχήμα διαμόρφωσης, αυξάνοντας παράλληλα και τη πιθανότητα σφάλματος. Από τη διαδικασία του πειράματος επαληθεύεται η βασική θεωρία της τεχνικής PD – NOMA και τα όρια που υπάρχουν.

Εν κατακλείδι, η τεχνική PD – NOMA μελετάται τα τελευταία χρόνια γιατί μπορεί να εξοικονομήσει πόρους, συχνότητα και χρόνο, αυξάνοντας έτσι τη συνδεσιμότητα των χρηστών σε ένα δίκτυο. Η παρούσα εργασία προσπαθεί να συμβάλει σε αυτή τη διαδικασία υλοποιώντας την τεχνική με μονάδες ραδιοεπικοινωνίας, εξάγοντας αποτελέσματα και συμπεράσματα για διαφορετικές παραμέτρους του συστήματος. Η εφαρμογή που αναπτύχθηκε αλλά και τα πειράματα που πραγματοποιήθηκαν μπορούν να επεκταθούν περαιτέρω έτσι ώστε να διερευνηθεί η ορθή παραμετροποίηση του συστήματος, σε ότι αφορά την επιλογή των χρηστών και την κατανομή της ισχύος, σε πραγματικά περιβάλλοντα, προκειμένου να επιτυγχάνεται συνολικό όφελος στη χρήση των πόρων του ραδιοδικτύου αλλά και την ικανοποίηση των απαιτήσεων των χρηστών. Παρά το γεγονός ότι υπάρχουν πολλές σχετικές θεωρητικές μελέτες ή μελέτες μέσω προσομοίωσης, η διερεύνηση σε πραγματικές ή σχεδόν πραγματικές συνθήκες μπορεί να βοηθήσει στην εξαγωγή συμπερασμάτων για την εφαρμογή των τεχνικών PD – NOMA.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] Analog Devices Wiki, "ADALM-PLUTO Detailed Specifications", 2021. [Online]. Available: https://wiki.analog.com/university/tools/pluto/devs/specs.

[2] Analog Devices, "Software-Defined Radio Active Learning Module". [Online]. Available: https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards kits/adalm-pluto.html#eb-overview.

[3] S. M. Riazul Islam, Nurilla Avazov, Octavia A. Dobre, Kyung-Sup Kwak, "Power-Domain Non-Orthogonal Multiple Access (NOMA) in 5G Systems: Potentials and Challenges", IEEE communications surveys & tutorials, vol. 19, no. 2, pp. 721-742, second quarter 2017.

[4] Mahrukh Liaqat, Kamarul Ariffin Noordin, Tarik Abdul Latef, Kaharudin Dimyati, "Power-domain non orthogonal multiple access (PD-NOMA) in cooperative networks: an overview", Springer Science+Business Media, LLC, part of Springer Nature 2018.

[5] Chunlin Yan, Atsushi Harada, Anass Benjebbour, Yang Lan, Anxin Li, Huiling Jiang, "Receiver Design for Downlink Non-Orthogonal Multiple Access (NOMA)", IEEE 81st Vehicular Technology Conference (VTC Spring), 2015.

[6] Yifei Yuan, Zhifeng Yuan, Guanghui Yu, Chien-Hwa Hwang, Pei-Kai Liao, Anxin Li, and Kazuaki Takeda, "Non-Orthogonal Transmission Technology in LTE Evolution", IEEE Communications Magazine, pp. 68-74, 2016.

[7] Anass Benjebbour, Anxin Li, Keisuke Saito, Yuya Saito, Yoshihisa Kishiyama, Takehiro Nakamura, "NOMA: From Concept to Standardization", IEEE Conference on Standards for Communications and Networking (CSCN), pp. 18-23, 2015.

[8] Heunchul Lee, Sungsoo Kim, and Jong-Han Lim, "Multiuser Superposition Transmission (MUST) for LTE-A Systems", EEE ICC Mobile and Wireless Networking Symposium, 2016.

[9] Jose Armando Oviedo, Hamid R. Sadjadpour, "On the Power Allocation Limits for Downlink Multi-user NOMA with QoS", IEEE International Conference on Communications (ICC), 2018.

[10] Baoji Wang, Rongwing Zhang, Chen Chen, Xiang Cheng, Liuqing Yang, Ye Jin, "Interference Hypergraph-Based 3D Matching Resource Allocation Protocol for NOMA-V2X Networks", IEEE Access ( Volume: 7), 2019.

[11] 3GPP TSG RAN WG1, "Description of MUST Category 2", 2015.

[12] Travis F. Collins, Robin Getz, Di Pu, Alexander M. Wyglinski, *Software-Defined Radio for Engineers.* Analog Devices, 2018.

[13] The MathWorks Documentation, "What Is WLAN?". [Online]. Available: https://www.mathworks.com/help/wlan/gs/what-is-wlan.html.

[14] IEEE Std 802.11, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", 2016.

[15] Cory Beard and William Stallings, «Ασύρματες Επικοινωνίες, Δίκτυα & Συστήματα», Εκδόσεις Τζιόλα, 2017.

[16] Keysight Technologies, “Concepts of Orthogonal Frequency Division Multiplexing (OFDM) and 802.11 WLAN”. [Online]. Available: https://rfmw.em.keysight.com/wireless/helpfiles/89600b/webhelp/subsystems/wlan-ofdm/content/ofdm_basicprinciplesoverview.htm.

[17] Techplayon, “What is Inter Symbol Interference (ISI) in LTE, How Cyclic Prefix (CP) helps eliminating this problem ?”, 2017. [Online]. Available: https://www.techplayon.com/inter-symbol-interference-isi-lte-cyclic-prefix-cp-helps-eliminating-isi-problem/.

[18] Wireless Communications by Andrea Goldsmith, “BER calculation”, 2008. [Online]. Available: https://www.unilim.fr/pages_perso/vahid/notes/ber_awgn.pdf.

[19] Analog Devices, “AD9363 RF Agile Transceiver”, Rev. D datasheet, 2016.

[20] Behzad Razavi, “Design Considerations for Direct-Conversion Receivers”, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 44, no. 6, pp. 428-435, June 1997.

[21] Xilinx, “Zynq-7000 SoC First Generation Architecture”, DS190 (v1.11.1) datasheet, July 2018.

[22] Analog Devices Wiki, “What is libio”, 2021. [Online]. Available: https://wiki.analog.com/resources/tools-software/linux-software/libiio.

[23] The MathWorks Documentation, “Waveform Generation”, [Online]. Available: https://www.mathworks.com/help/wlan/gs/waveform-generation.html.

[24] Michael Rice, Fred Harris, “Polyphase filterbanks for symbol timing synchronization in sampled data receivers”, MILCOM 2002. Proceedings, pp. 982-986, 2002.

[25] The MathWorks Documentation, “Correct symbol timing clock skew”, [Online]. Available: https://www.mathworks.com/help/comm/ref/comm.symbolsynchronizer-system-object.html.

[26] Marco Luke, Ruggero Reggiannini, “Carrier Frequency Recovery in All-Digital Modems for Burst-Mode Transmissions”, IEEE Transactions on Communications, vol. 43, no. 2/3/4, February/ March/ April, pp. 1169-1178, 1995.

[27] The MathWorks, “Compensate for frequency offset for PAM, PSK, or QAM”, [Online]. Available: https://www.mathworks.com/help/comm/ref/comm.coarsefrequencycompensator-system-object.html.

[28] The MathWorks Documentation, “Compensate for carrier frequency offset”, [Online]. Available: https://www.mathworks.com/help/comm/ref/comm.carriersynchronizer-system-object.html.

[29] The MathWorks Documentation, “Communications Toolbox Support Package for Analog Devices ADALM-Pluto Radio”, [Online]. Available: https://www.mathworks.com/help/supportpkg/plutoradio/index.html.

[30] The MathWorks Documentation, “Generate bipolar Barker code”, [Online]. Available: https://www.mathworks.com/help/comm/ref/comm.barkercode-system-object.html.

[31] The MathWorks Documentation, “Phase shift keying modulation”, [Online]. Available: https://www.mathworks.com/help/comm/ref/pskmod.html.

[32] The MathWorks Documentation, “Create transmitter System object for Xilinx Zynq-based radio hardware”, [Online]. Available: https://www.mathworks.com/help/supportpkg/xilinxzynqbasedradio/ug/sdrtx.html.

[33] The MathWorks Documentation, “Apply pulse shaping by interpolating signal using raised-cosine FIR filter”, [Online]. Available: https://www.mathworks.com/help/comm/ref/comm.raisedcosinereceivefilter-system-object.html.

[34] The MathWorks Documentation, “Download waveform to radio and repeatedly transmit it over the air”, [Online]. Available: https://www.mathworks.com/help/supportpkg/plutoradio/ref/comm.sdrtxpluto.transmitrepeat.html.

[35] The MathWorks Documentation, “Create receiver System object for radio hardware”, [Online]. Available: https://www.mathworks.com/help/supportpkg/plutoradio/ref/sdrrx.html.

[36] The MathWorks Documentation, “Apply pulse shaping by decimating signal using raised-cosine FIR filter”, [Online]. Available: https://www.mathworks.com/help/comm/ref/comm.raisedcosinereceivefilter-system-object.html.

[37] The MathWorks Documentation, “Detect preamble in data”, [Online]. Available: https://www.mathworks.com/help/comm/ref/comm.preambledetector-system-object.html.

[38] The MathWorks Documentation, “Image Transmission and Reception Using WLAN Toolbox and One PlutoSDR”, [Online]. Available:https://www.mathworks.com/help//supportpkg/plutoradio/ug/transmission-and-reception-of-an-image-using-wlan-system-toolbox-and-a-single-pluto-radio.html.

[39] The MathWorks Documentation, “Generate CRC code bits and append to input data”, [Online]. Available: https://www.mathworks.com/help/comm/ref/comm.crcgenerator-system-object.html.

[40] The MathWorks Documentation, “Generate WLAN waveform”, [Online]. Available: https://www.mathworks.com/help/wlan/ref/wlanwaveformgenerator.html.

[41] The MathWorks Documentation, “Generate non-HT-Data field waveform”, [Online]. Available: https://www.mathworks.com/help/wlan/ref/wlannonhtdata.html.

[42] The MathWorks Documentation, “Recover non-HT Data”, [Online]. Available: https://www.mathworks.com/help/wlan/ref/wlannonhtdatarecover.html.

[43] The MathWorks Documentation, “Measure modulation error ratio”, [Online]. Available: https://www.mathworks.com/help/comm/ref/comm.mer-system-object.html.

[44] The MathWorks Documentation, “Measure signal-to-noise ratio (SNR) in digital modulation applications”, [Online]. Available: https://www.mathworks.com/help/comm/ref/mermeasurement.html.

[45] Diego Koch, “ADI SDR Transceivers Enable Amateur Space Communication”, Analog Dialog, vol.54, no.1, 2020.

# ΠΑΡΑΡΤΗΜΑ Α : Κώδικες εξομοίωσης απλού PD – NOMA

### *A. "Create a Signal"*

```
clear all
close all
clc

barker = comm.BarkerCode('SamplesPerFrame',2^7,'Length',13);
preamble = complex(barker());
preamble_length = length(preamble);
Total_preamble = [preamble;preamble];
Total_preamble_length = length(Total_preamble);
release(barker);

%Create a Frame guard
guard = complex(0.0001*randn(1,100)');
guard_length = length(guard);

N = 300; %Number of bits
M = 2;
%Primary information
data_bits_p = rand(1,N*log2(M))>0.5;
data_bits_r_p = reshape(data_bits_p',log2(M),N)';
data_symbols_p = bi2de(data_bits_r_p,'left-msb');
data_symbols_length_p = length(data_symbols_p);
%Modulation for Primary
Mod_data_p = 0.9*pskmod(data_symbols_p,M,pi);

%Secondary information
data_bits_s = rand(1,N*log2(M))>0.5;
data_bits_r_s = reshape(data_bits_s',log2(M),N)';
data_symbols_s = bi2de(data_bits_r_s,'left-msb');
data_symbols_length_s = length(data_symbols_s);
%Modulation for Secondary
Mod_data_s = 0.1*pskmod(data_symbols_s,M,pi);

%Superposition
Mod_data = Mod_data_p + Mod_data_s;

%Create Frame
msg_tx = [guard;preamble;preamble;Mod_data];

sampleRate = 300e3;
data_rate = 20e3;
Samples_per_Symbol = sampleRate/data_rate;

constdiagram = comm.ConstellationDiagram(...
    'ReferenceConstellation',0.9*pskmod(0:M-1,M), ...
    'SamplesPerSymbol',Samples_per_Symbol, ...
    'SymbolsToDisplaySource','Property','SymbolsToDisplay',1000, ...
    'XLimits',[-3 3],'YLimits',[-3 3]);

constdiagram(Mod_data)
release(constdiagram);
```

### *B. "Transmit a Signal"*

```
plutoradiosetup()
configurePlutoRadio('AD9363')

sampleRate = 300e3; %Nyquist Sampling Rate
centerFreq = 915e6; %Carrier Frequency
data_rate = 20e3; %Data rate
Samples_per_Symbol = sampleRate/data_rate; %Samples taken for one Symbol
samples_per_frame = 3*Samples_per_Symbol*(guard_length + data_symbols_length_s +
Total_preamble_length);
%global plutoRadio
%plutoRadio = findPlutoRadio;
tx = sdrtx('Pluto','RadioID','usb:0','CenterFrequency', centerFreq, ...
              'BasebandSampleRate', sampleRate,'OutputDataType','double', 'Gain',0,
...
              'ShowAdvancedProperties', true,'FrequencyCorrection',0);

rctFilt =
comm.RaisedCosineTransmitFilter('OutputSamplesPerSymbol',Samples_per_Symbol, ...

'FilterSpanInSymbols',10);

tx_signal = rctFilt(msg_tx);

transmitRepeat(tx,tx_signal);
```

### *C. "Receive a Signal"*

```
sampleRate = 300e3; %Nyquist Sampling Rate
centerFreq = 915e6; %Carrier Frequency
data_rate = 20e3; %Data rate
Samples_per_Symbol = sampleRate/data_rate; %Samples taken for one Symbol

samples_per_frame = 3*Samples_per_Symbol*(guard_length + data_symbols_length_s +
Total_preamble_length);

%global plutoRadio
%plutoRadio = findPlutoRadio;
rx = sdrrx('Pluto','RadioID','usb:1',...

'GainSource','Manual','BasebandSampleRate',sampleRate,'Gain',35,'CenterFrequency',cen
terFreq,...

'OutputDataType','double','SamplesPerFrame',samples_per_frame,'ShowAdvancedProperties
',true,'FrequencyCorrection',0);

% Capture frames of data
[rx_signal,~] = rx();

release(tx);
release(rx);

rcrFilt = comm.RaisedCosineReceiveFilter(...

'InputSamplesPerSymbol',Samples_per_Symbol,'DecimationFactor',Samples_per_Symbol,...
```

```
  'FilterSpanInSymbols',10);

rx_filtered_signal = rcrFilt(rx_signal);
```

### *D. "Detect a Signal"*

```
fs = 300e3; % Nyquist Sampling Rate

barker = comm.BarkerCode('SamplesPerFrame',2^7,'Length',13);
preamble = barker();
release(barker);

preamble_long = [preamble;preamble];

CFC = comm.CoarseFrequencyCompensator('SampleRate',fs, ...
    'Modulation','BPSK','FrequencyResolution',1);
FFC = comm.CarrierSynchronizer( ...
    'SamplesPerSymbol',1,'Modulation','BPSK');
prbdet = comm.PreambleDetector([preamble;preamble],'Threshold',3,'Detections','All');

[rx_coarse,estfreqoff] = CFC(rx_filtered_signal);

[rx_fine,phErr] = FFC(rx_coarse);

[idx1,detmet1] = prbdet(rx_fine);

sorted_detmet1 = sort(detmet1,'descend');

prbdet.Threshold = floor(sorted_detmet1(2));

[idx2,detmet2] = prbdet(rx_fine);

data_start_index = idx2(1) + 1;
data_end_index = data_start_index + data_symbols_length_p - 1;

%%% guard length = 100
guard_start_index = idx2(1) - 100 - length(preamble_long) + 1;
guard_end_index = guard_start_index + 100 - 1;

preamble_start_index = idx2(1) - length(preamble_long) + 1;
preamble_end_index = preamble_start_index + length(preamble_long) - 1;

%%% Check if synchronization algorithm added 180 degree phase offset
rx_preamble_symbols = rx_fine(preamble_start_index:preamble_end_index);
rx_preamble_bits = real(rx_preamble_symbols);
rx_preamble_data = zeros(length(preamble_long),1);
for i = 1:length(preamble_long)
    if rx_preamble_bits(i) > 0
        rx_preamble_data(i) = 1;
    else
        rx_preamble_data(i) = -1;
    end
end

phOffsetEst = angle(preamble_long.*rx_preamble_data);
ph_offset_degrees = rad2deg(mean(phOffsetEst));
```

```
if abs(ph_offset_degrees) > 90
    rx_fine = -rx_fine;
end

%Primary Demodulation
rx_message_symbols = rx_fine(data_start_index:data_end_index);

Demod_p = pskdemod(rx_message_symbols,M,pi);

errors_p = Demod_p - data_symbols_p ;

numberErrors_p = sum(Demod_p ~= data_symbols_p)

constdiagram(rx_message_symbols)

%Secondary Demodulation
rx_message_symbols_1 = rx_fine(data_start_index:data_end_index);

rx_message_symbols_1_real = real(rx_message_symbols_1);

rx_message_symbols_1_abs = abs(rx_message_symbols_1_real);

mean_rx_message_symbols_1 = mean(rx_message_symbols_1_abs);

Demod_p_s = pskdemod(rx_message_symbols_1,M,pi); %Demodulate as Primary

est_signal_p = mean_rx_message_symbols_1*pskmod(Demod_p_s,M,pi); %Modulate again

Secondary_cancellation = rx_message_symbols_1 - est_signal_p; %SIC

Demod_s = pskdemod(Secondary_cancellation,M,pi);

errors_s = Demod_s - data_symbols_s;

numberErrors_s = sum(Demod_s ~= data_symbols_s)
```

# ΠΑΡΑΡΤΗΜΑ Β : Κώδικες εξομοίωσης WLAN PD – NOMA

### *A. "Main program"*

```
clc
clear all
close all

Niter = 1;
for j = 1:1:Niter

N = 8096; %Number of bits
M = 2;

%power_allocation
k_s = 0.2;
k_p = 1;
spectrumScope = dsp.SpectrumAnalyzer( ...
    'SpectrumType', 'Power density', ...
    'SpectralAverages', 10, ...
    'YLimits', [-130 -10], ...
    'Title', 'Received Baseband WLAN Signal Spectrum', ...
    'YLabel', 'Power spectral density', ...
    'Position', [69 376 800 450]);

constellation1 = comm.ConstellationDiagram(...
    'Title', 'NOMA Symbols','ReferenceConstellation',(sqrt(2)/2)*qammod(0:4-1,4),
...
    'ShowReferenceConstellation', false, ...
    'Position', [878 376 460 460]);
%BPSK    %pskmod(0:2-1,2)
%QPSK    %(sqrt(2)/2)*qammod(0:4-1,4)
%16QAM  %(sqrt(10)/10)*qammod(0:16-1,16)
constellation2 = comm.ConstellationDiagram(...
    'Title', 'Secondary
Symbols','ReferenceConstellation',((sqrt(2)/2)*0.2)*qammod(0:4-1,4), ...
    'ShowReferenceConstellation', false, ...
    'Position', [878 376 460 460]);
%BPSK    %0.2*pskmod(0:2-1,2)
%QPSK    %((sqrt(2)/2)*0.2)*qammod(0:4-1,4)
%16QAM  %(sqrt(10)/10)*qammod(0:16-1,16)
mer = comm.MER('MinimumMEROutputPort',true, ...
    'XPercentileMEROutputPort',true,'XPercentileValue',90,...
    'SymbolCountOutputPort',true);

deviceNameSDR = 'Pluto';
radio = sdrdev(deviceNameSDR);
txGain = -10;

% Primary information
data_bits_p = rand(1,N*log2(M))>0.5;
data_bits_p_r = reshape(data_bits_p',N,log2(M));

% Secondary information
data_bits_s = rand(1,N*log2(M))>0.5;
data_bits_s_r = reshape(data_bits_s',N,log2(M));

data_bits = [data_bits_p_r ; data_bits_s_r];
```

```
%Data bits length
data_bits_length = length(data_bits);

msduLength = 2024; % MSDU length in bytes
msduBits = msduLength*8;

numMSDUs = ceil(length(data_bits)/msduBits);
padZeros = msduBits-mod(length(data_bits),msduBits);
txData = [data_bits; zeros(padZeros,1)];

% Generate FCS and from an MPDU. The FCS is calculated using the standard
% generator polynomial of degree 32 as defined in section 8.2.4.8 of
% 802.11n HT standard
generatorPolynomial = [32 26 23 22 16 12 11 10 8 7 5 4 2 1 0];
fcsGenerator = comm.CRCGenerator(generatorPolynomial);
fcsGenerator.InitialConditions = 1;
fcsGenerator.DirectMethod = true;
fcsGenerator.FinalXOR = 1;

% Divide input data stream into fragments
numFragment = 0;
bitsPerOctet = 8;
lengthMACheader = 256; % MPDU header length in bits
lengthFCS = 32;        % FCS length in bits
lengthMPDU = (2*lengthMACheader)+(2*lengthFCS)+msduBits; % MPDU length in bits
data = zeros(lengthMPDU*numMSDUs,1);

for ind=0:numMSDUs-1

   % Extract bits for each MPDU
   frameBody = txData(ind*msduBits+1:msduBits*(ind+1),:);
   % Generate MPDU header bits
   mpduHeader = helperNonHTMACHeader(mod(numFragment, ...
                   16),mod(ind,4096));

   % Create MPDU with header, body and FCS
   psdu1 = fcsGenerator([mpduHeader;frameBody(1:N)]);
   psdu2 = fcsGenerator([mpduHeader;frameBody((N+1):(2*N))]);
   psdu = [psdu1;psdu2];
   % Concatenate PSDUs for waveform generation
   data(lengthMPDU*ind+1:lengthMPDU*(ind+1)) = psdu;
end

% *Generate IEEE 802.11a Baseband WLAN Signal*
nonHTcfg = wlanNonHTConfig;         % Create packet configuration
nonHTcfg.MCS = 2;
nonHTcfg.NumTransmitAntennas = 1;   % Number of transmit antenna
chanBW = nonHTcfg.ChannelBandwidth;
nonHTcfg.PSDULength = (lengthMPDU/2)/bitsPerOctet; % Set the PSDU length


%Transmitter Setup
sdrTransmitter = sdrtx(deviceNameSDR); % Transmitter properties
sdrTransmitter.RadioID = 'usb:0';

fs = wlanSampleRate(nonHTcfg); % Transmit sample rate in MHz
osf = 1.5;                     % Oversampling factor

sdrTransmitter.BasebandSampleRate = fs*osf;
```

```
sdrTransmitter.CenterFrequency = 2.432e9;  % Channel 5
sdrTransmitter.ShowAdvancedProperties = true;
sdrTransmitter.Gain = txGain;

% Initialize the scrambler with a random integer for each packet
%scramblerInitialization = randi([1 127],numMSDUs,1);
scramblerInitialization = 43;
% Generate baseband NonHT packets separated by idle time
 txWaveform = wlanWaveformGenerator_NOMA(data,nonHTcfg, ...
    'NumPackets',numMSDUs,'IdleTime',20e-6, ...
    'ScramblerInitialization',scramblerInitialization);



% Resample transmit waveform
txWaveform  = resample(txWaveform,fs*osf,fs);
fprintf('\nGenerating NOMA transmit waveform:\n')

% Scale the normalized signal to avoid saturation of RF stages
powerScaleFactor = 0.8;
txWaveform = txWaveform.*(1/max(abs(txWaveform))*powerScaleFactor);



% Transmit RF waveform
sdrTransmitter.transmitRepeat(txWaveform);

%Receiver Setup
sdrReceiver = sdrrx(deviceNameSDR);
sdrReceiver.RadioID = 'usb:1';
sdrReceiver.BasebandSampleRate = sdrTransmitter.BasebandSampleRate;
sdrReceiver.CenterFrequency = sdrTransmitter.CenterFrequency;
sdrReceiver.GainSource = 'Manual';
sdrReceiver.Gain = 30;
sdrReceiver.OutputDataType = 'double';

% Configure receive samples equivalent to twice the length of the
% transmitted signal, this is to ensure that PSDUs are received in order.
% On reception the duplicate MAC fragments are removed.
samplesPerFrame = length(txWaveform);
sdrReceiver.SamplesPerFrame = samplesPerFrame*2;
spectrumScope.SampleRate = sdrReceiver.BasebandSampleRate;

% Get the required field indices within a PSDU
indLSTF = wlanFieldIndices(nonHTcfg,'L-STF');
indLLTF = wlanFieldIndices(nonHTcfg,'L-LTF');
indLSIG = wlanFieldIndices(nonHTcfg,'L-SIG');
Ns = indLSIG(2)-indLSIG(1)+1; % Number of samples in an OFDM symbol

% The transmitted waveform is captured using the PlutoSDR.
fprintf('\nStarting a new RF capture.\n')

burstCaptures = sdrReceiver();

% Show power spectral density of the received waveform
spectrumScope(burstCaptures);

% Downsample the received signal
rxWaveform = resample(burstCaptures,fs,fs*osf);
rxWaveformLen = size(rxWaveform,1);
searchOffset = 0; % Offset from start of the waveform in samples
```

```
% Minimum packet length is 10 OFDM symbols
lstfLen = double(indLSTF(2)); % Number of samples in L-STF
minPktLen = lstfLen*5;
pktInd = 1;
sr = wlanSampleRate(nonHTcfg); % Sampling rate
fineTimingOffset = [];
packetSeq = [];
displayFlag = 0; % Flag to display the decoded information


% Generate FCS for MPDU
fcsDetector = comm.CRCDetector(generatorPolynomial);
fcsDetector.InitialConditions = 1;
fcsDetector.DirectMethod = true;
fcsDetector.FinalXOR = 1;

% Perform EVM calculation
evmCalculator = comm.EVM('AveragingDimensions',[1 2 3]);
evmCalculator.MaximumEVMOutputPort = true;



% Receiver processing
while (searchOffset + minPktLen) <= rxWaveformLen
    % Packet detect
    pktOffset = wlanPacketDetect(rxWaveform, chanBW, searchOffset, 0.8);

    % Adjust packet offset
    pktOffset = searchOffset+pktOffset;
    if isempty(pktOffset) || (pktOffset+double(indLSIG(2))>rxWaveformLen)
        if pktInd==1
            disp('** No packet detected **');
        end
        break;
    end

    % Extract non-HT fields and perform coarse frequency offset correction
    % to allow for reliable symbol timing
    nonHT = rxWaveform(pktOffset+(indLSTF(1):indLSIG(2)),:);
    coarseFreqOffset = wlanCoarseCFOEstimate(nonHT,chanBW);
    nonHT = helperFrequencyOffset(nonHT,fs,-coarseFreqOffset);

    % Symbol timing synchronization
    fineTimingOffset = wlanSymbolTimingEstimate(nonHT,chanBW);

    % Adjust packet offset
    pktOffset = pktOffset+fineTimingOffset;

    % Timing synchronization complete: Packet detected and synchronized
    % Extract the NonHT preamble field after synchronization and
    % perform frequency correction
    if (pktOffset<0) || ((pktOffset+minPktLen)>rxWaveformLen)
        searchOffset = pktOffset+1.5*lstfLen;
        continue;
    end
    fprintf('\nPacket-%d detected at index %d\n',pktInd,pktOffset+1);

    % Extract first 7 OFDM symbols worth of data for format detection and
```

```
    % L-SIG decoding
    nonHT = rxWaveform(pktOffset+(1:7*Ns),:);
    nonHT = helperFrequencyOffset(nonHT,fs,-coarseFreqOffset);

    % Perform fine frequency offset correction on the synchronized and
    % coarse corrected preamble fields
    lltf = nonHT(indLLTF(1):indLLTF(2),:);           % Extract L-LTF
    fineFreqOffset = wlanFineCFOEstimate(lltf,chanBW);
    nonHT = helperFrequencyOffset(nonHT,fs,-fineFreqOffset);
    cfoCorrection = coarseFreqOffset+fineFreqOffset; % Total CFO

    % Channel estimation using L-LTF
    lltf = nonHT(indLLTF(1):indLLTF(2),:);
    demodLLTF = wlanLLTFDemodulate(lltf,chanBW);
    chanEstLLTF = wlanLLTFChannelEstimate(demodLLTF,chanBW);

    % Noise estimation
    noiseVarNonHT = helperNoiseEstimate(demodLLTF);

    % Packet format detection using the 3 OFDM symbols immediately
    % following the L-LTF
    format = wlanFormatDetect(nonHT(indLLTF(2)+(1:3*Ns),:), ...
        chanEstLLTF,noiseVarNonHT,chanBW);
    disp(['  ' format ' format detected']);
    if ~strcmp(format,'Non-HT')
        fprintf('  A format other than Non-HT has been detected\n');
        searchOffset = pktOffset+1.5*lstfLen;
        continue;
    end

    % Recover L-SIG field bits
    [recLSIGBits,failCheck] = wlanLSIGRecover( ...
           nonHT(indLSIG(1):indLSIG(2),:), ...
           chanEstLLTF,noiseVarNonHT,chanBW);

    if failCheck
        fprintf('  L-SIG check fail \n');
        searchOffset = pktOffset+1.5*lstfLen;
        continue;
    else
        fprintf('  L-SIG check pass \n');
    end

    % Retrieve packet parameters based on decoded L-SIG
    [lsigMCS,lsigLen,rxSamples] = helperInterpretLSIG(recLSIGBits,sr);

    if (rxSamples+pktOffset)>=length(rxWaveform)
        disp('** There is only one received packet **');
        break;
    end

    % Apply CFO correction to the entire packet
    rxWaveform(pktOffset+(1:rxSamples),:) = helperFrequencyOffset(...
    rxWaveform(pktOffset+(1:rxSamples),:),fs,-cfoCorrection);

    % Create a receive Non-HT config object
    rxNonHTcfg = wlanNonHTConfig;
    rxNonHTcfg.MCS = lsigMCS;
    rxNonHTcfg.PSDULength = lsigLen;
```

```
    % Get the data field indices within a PPDU
    indNonHTData = wlanFieldIndices(rxNonHTcfg,'NonHT-Data');

    % Recover Primary's PSDU bits using transmitted packet parameters and channel
    % estimates from L-LTF
    [rxPSDU_p,eqSym_p] = wlanNonHTDataRecover(rxWaveform(pktOffset+...
           (indNonHTData(1):indNonHTData(2)),:), ...
           chanEstLLTF,noiseVarNonHT,rxNonHTcfg);

    % Recover Secondary's PSDU bits using transmitted packet parameters and channel
    % estimates from L-LTF
    [rxPSDU_s,Secondary_cancelation] =
wlanNonHTDataRecover_s(rxWaveform(pktOffset+...
           (indNonHTData(1):indNonHTData(2)),:), ...
          chanEstLLTF,noiseVarNonHT,rxNonHTcfg);

    constellation1(reshape(eqSym_p,[],1)); % Current constellation
    pause(0); % Allow constellation to repaint
    release(constellation1); % Release previous constellation plot

    constellation2(reshape(Secondary_cancelation,[],1)); % Current constellation
    pause(0); % Allow constellation to repaint
    release(constellation2); % Release previous constellation plot


    refSym p = helperReferenceSymbols(eqSym p,rxNonHTcfg);
    [evm.RMS,evm.Peak] = evmCalculator(refSym_p,eqSym_p);

    refSym_s = helperReferenceSymbols(Secondary_cancelation,rxNonHTcfg);
    refSym_s = refSym_s*k_s;
    [evm.RMS,evm.Peak] = evmCalculator(refSym_s,Secondary_cancelation);

    refSym_p = reshape(refSym_p,[],1);
    eqSym_p = reshape(eqSym_p,[],1);
    MER_p = mer(refSym_p,eqSym_p);
    MER_p_Linear = 10.^(MER_p/10);
    %MER_p_Linear = sum(MER_p_Linear)/176;

    refSym_s = reshape(refSym_s,[],1);
    Secondary_cancelation = reshape(Secondary_cancelation,[],1);
    MER_s = mer(refSym_s,Secondary_cancelation);
    MER_s_Linear = 10.^(MER_s/10);
    %MER_s_Linear = sum(MER_s_Linear)/176;

    % Remove FCS from MAC header and frame body
    [rxBit_p{pktInd},crcCheck_p] = fcsDetector(double(rxPSDU_p)); %#ok<SAGROW>
    [rxBit_s{pktInd},crcCheck_s] = fcsDetector(double(rxPSDU_s)); %#ok<SAGROW>

    if ~crcCheck_p
         disp('  MAC CRC check pass');
    else
         disp('  MAC CRC check fail');
    end

     if ~crcCheck_s
         disp('  MAC CRC check pass');
    else
        disp('  MAC CRC check fail');
```

```
    end

    % Process receive MAC fragments, this is to retrieve the sequencing
    % information of the MAC fragments
    [mac_p,packetSeq(pktInd)] = ...
           helperNonHTMACHeaderDecode(rxBit_p{pktInd}); %#ok<SAGROW>

    [mac_s,packetSeq(pktInd)] = ...
           helperNonHTMACHeaderDecode(rxBit_s{pktInd}); %#ok<SAGROW>

    % Display decoded information
    if displayFlag
        fprintf('  Estimated CFO: %5.1f Hz\n\n',cfoCorrection); %#ok<UNRCH>

        disp('  Decoded L-SIG contents: ');
        fprintf('                            MCS: %d\n',lsigMCS);
        fprintf('                         Length: %d\n',lsigLen);
        fprintf('    Number of samples in packet: %d\n\n',rxSamples);

        fprintf('  EVM:\n');
        fprintf('    EVM peak: %0.3f%%  EVM RMS: %0.3f%%\n\n', ...
        evm.Peak,evm.RMS);

        fprintf('  Decoded MAC Sequence Control field contents:\n');
        fprintf('    Sequence number:%d\n',packetSeq(pktInd));
    end

    % Update search index
    searchOffset = pktOffset+double(indNonHTData(2));


    pktInd = pktInd+1;
    % Finish processing when a duplicate packet is detected. The
    % recovered data includes bits from duplicate frame
    if length(unique(packetSeq))<length(packetSeq)
        break
    end
end

% Release the state of sdrTransmitter and sdrReceiver object
release(sdrTransmitter);
release(sdrReceiver);

% Remove the MAC header and duplicate captured MAC fragment
 rxBitMatrix_p = cell2mat(rxBit_p);
 rxData_p = rxBitMatrix_p(lengthMACheader+1:end);

 startSeq = find(packetSeq==0);
 rxData_p = circshift(rxData_p,[0 -(startSeq(1)-1)]);% Order MAC fragments

 % Perform bit error rate (BER) calculation for primary
 bitErrorRate = comm.ErrorRate;
 err_p = bitErrorRate(double(rxData_p(:)), txData(1:8096));
% fprintf('  \nBit Error Rate (BER) for Primary:\n');
% fprintf('          Bit Error Rate (BER) = %0.5f.\n',err_p(1));
 fprintf('          Number of bit errors for Primary = %d.\n', err_p(2));
 fprintf('    Number of transmitted bits per loop = %d.\n\n',length(data_bits_p_r));
```

```
 % Remove the MAC header and duplicate captured MAC fragment
 rxBitMatrix_s = cell2mat(rxBit_s);
 rxData_s = rxBitMatrix_s(lengthMACheader+1:end);

 startSeq = find(packetSeq==0);
 rxData_s = circshift(rxData_s,[0 -(startSeq(1)-1)]);% Order MAC fragments

% Perform bit error rate (BER) calculation for secondary
bitErrorRate = comm.ErrorRate;
err_s = bitErrorRate(double(rxData_s(:)), txData(8097:16192));
%fprintf('  \nBit Error Rate (BER) for Secondary:\n');
%fprintf('          Bit Error Rate (BER) = %0.5f.\n',err_s(1));
fprintf('          Number of bit errors for Secondary = %d.\n', err_s(2));
fprintf('    Number of transmitted bits per loop = %d.\n\n',length(data_bits_s_r));



 Number_of_errors_p(j) = err_p(2);
 Number_of_errors_s(j) = err_s(2);
 Modulation_Error_Rate_for_primary(j) = MER_p_Linear;
 Modulation_Error_Rate_for_secondary(j) = MER_s_Linear;

 Spectrum_data = getSpectrumData(spectrumScope);
 Spectrum_data = table2array(Spectrum_data);
 Spectrum_data = cell2mat(Spectrum_data(2));
 Spectrum data lin = 10.^(.1*Spectrum data);
 Spectrum_data_lin_sum = sum(Spectrum_data_lin,1);
 powerdBm = 10*log10(Spectrum_data_lin_sum)+42.85;

 power_dBm(j) = powerdBm;

 pause(10);
end

%Number_of_errors_p = Number_of_errors_p';
%Number_of_errors_s = Number_of_errors_s';
%Modulation_Error_Rate_for_primary = Modulation_Error_Rate_for_primary';
%Modulation_Error_Rate_for_secondary = Modulation_Error_Rate_for_secondary';



Total_bits = N*Niter;
BER p = sum(Number of errors p)/Total bits;
fprintf('          Bit Error Rate for Primary (BER) = %0.5f.\n',BER_p);
BER_s = sum(Number_of_errors_s)/Total_bits;
fprintf('          Bit Error Rate for Secondary (BER) = %0.5f.\n',BER_s);
```

***B. Function: "WLAN Waveform Generator NOMA"***

```
function txWaveform = wlanWaveformGenerator_NOMA(dataBits,cfgFormat,varargin)

% Check number of input arguments
coder.internal.errorIf((nargin ~= 3) && mod(nargin, 2) == 1,
'wlan:wlanWaveformGenerator:InvalidNumInputs');

% Validate the format configuration object is a valid type
validateattributes(cfgFormat,{'wlanVHTConfig','wlanHTConfig','wlanNonHTConfig','wlan
S1GConfig','wlanDMGConfig'},{'scalar'},mfilename,'format configuration object');
s = validateConfig(cfgFormat);
inDSSSMode = isa(cfgFormat,'wlanNonHTConfig') &&
strcmpi(cfgFormat.Modulation,'DSSS');
isDMGOFDM = isa(cfgFormat,'wlanDMGConfig') && strcmp(phyType(cfgFormat),'OFDM');
inOFDMMode = isDMGOFDM || isa(cfgFormat,'wlanS1GConfig')||
isa(cfgFormat,'wlanVHTConfig') || isa(cfgFormat,'wlanHTConfig') || ...
    (isa(cfgFormat,'wlanNonHTConfig') && strcmpi(cfgFormat.Modulation,'OFDM'));

% Set maximum limits for windowing transition time based on bandwidth and
% format for the validity check performed later.
maxWinTransitTime = 6.4e-6;         % default

switch class(cfgFormat)
    case 'wlanNonHTConfig'
        if strcmpi(cfgFormat.Modulation, 'OFDM')
            switch cfgFormat.ChannelBandwidth
                case 'CBW5'
                    maxWinTransitTime = 6.4e-6; % seconds
                case 'CBW10'
                    maxWinTransitTime = 3.2e-6; % seconds
                otherwise  % 'CBW20'
                    maxWinTransitTime = 1.6e-6; % seconds
            end
        end
    case 'wlanS1GConfig'
        maxWinTransitTime = 16e-6; % seconds
    case 'wlanDMGConfig'
        maxWinTransitTime = 9.6969e-08; % seconds
    otherwise % HT/VHT
        maxWinTransitTime = 1.6e-6; % seconds
end

overrideObjectScramInit = false;

if (nargin == 3) % wlanGeneratorConfig
    % Use of wlanGeneratorConfig object not supported
    % Use errorif to throw an error during compilation

coder.internal.errorIf(true,'wlan:wlanWaveformGenerator:GeneratorConfigDeprecation')
;
else % P-V pairs
    % Define default P-V pair values
    numPackets     = 1;
    idleTime       = 0;
    scramblerInit  = 43;
    if isa(cfgFormat,'wlanDMGConfig')
        winTransitTime = 16/2640e6; % Windowing length of 16
    else
        winTransitTime = 1e-7;
```

```
    end

    % Validate each P-V pair
    for i = 3:2:nargin
        prop = varargin{i-2};
        val  = varargin{i-1};

        coder.internal.errorIf(~(ischar(prop) || (isstring(prop) && isscalar(prop)))
||...

~any(strcmp(prop,{'NumPackets','IdleTime','ScramblerInitialization','WindowTransitio
nTime'})),'wlan:wlanWaveformGenerator:InvalidProperty');

        switch prop
          case 'NumPackets'

validateattributes(val,{'numeric'},{'scalar','integer','>=',0},mfilename,'''NumPacke
ts'' value');
            numPackets = val(1);
          case 'IdleTime'
            validateattributes(val,{'numeric'}, ...
                {'scalar','real','>=',0},mfilename,'''IdleTime'' value');
                if isa(cfgFormat,'wlanDMGConfig')
                    minIdleTime = 1e-6;
                else % S1G, VHT, HT, non-HT
                    minIdleTime = 2e-6;
                end
                coder.internal.errorIf((val(1) > 0) && (val(1) <
minIdleTime),'wlan:wlanWaveformGenerator:InvalidIdleTimeValue',sprintf('%1.0d',minId
leTime));
            idleTime = val(1);
          case 'ScramblerInitialization'
            if ~inDSSSMode
                if isa(cfgFormat,'wlanDMGConfig')
                    if strcmp(phyType(cfgFormat),'Control')
                        coder.internal.errorIf(any((val<1) |
(val>15)),'wlan:wlanWaveformGenerator:InvalidScramblerInitialization','Control',1,15
);
                    else
                         coder.internal.errorIf(any((val<1) |
(val>127)),'wlan:wlanWaveformGenerator:InvalidScramblerInitialization','SC/OFDM',1,1
27);
                    end
                    overrideObjectScramInit = true;
                else

validateattributes(val,{'double','int8'},{'real','integer','2d','nonempty','>=',1,'<
=',127},mfilename,'''ScramblerInitialization'' value');
                end
            end
            scramblerInit = val;
          otherwise % 'WindowTransitionTime'
            if inOFDMMode
                validateattributes(val,
{'numeric'},{'real','scalar','>=',0,'<=',maxWinTransitTime},mfilename,'''WindowTrans
itionTime'' value');
            end
            winTransitTime = val(1);
        end
    end
    windowing = inOFDMMode && winTransitTime > 0;
```

```
end

if isa(cfgFormat,'wlanVHTConfig') || isa(cfgFormat,'wlanS1GConfig')
    numUsers = cfgFormat.NumUsers;
else
    numUsers = 1;
end

% Cross validation
coder.internal.errorIf(all(size(scramblerInit,2) ~= [1
numUsers]),'wlan:wlanWaveformGenerator:ScramInitNotMatchNumUsers');

% Validate that data bits are present if PSDULength is nonzero
if iscell(dataBits) % SU and MU
    % Data must be a scalar cell or a vector cell of length Nu
    coder.internal.errorIf(~isvector(dataBits) || all(length(dataBits) ~= [1
numUsers]), 'wlan:wlanWaveformGenerator:InvalidDataCell');

    for u = 1:length(dataBits)
        if ~isempty(dataBits{u}) && any(cfgFormat.PSDULength(:) > 0) % Data packet

validateattributes(dataBits{u},{'double','int8'},{'real','integer','vector','binary'
},mfilename,'each element in cell data input');
        else
            % Empty data check if not NDP
            coder.internal.errorIf(any(cfgFormat.PSDULength(:) > 0) &&
isempty(dataBits{u}),'wlan:wlanWaveformGenerator:NoData');
        end
    end
    if (numUsers > 1) && isscalar(dataBits)
        % Columnize and expand to a [1 Nu] cell
        dataCell = repmat({int8(dataBits{1}(:))},1,numUsers);
    else % Columnize each element
        dataCell = repmat({int8(1)},1,numUsers);
        for u = 1:numUsers
            dataCell{u} = int8(dataBits{u}(:));
        end
    end
else % SU and MU: Data must be a vector
    if ~isempty(dataBits) && any(cfgFormat.PSDULength(:) > 0) % Data packet

validateattributes(dataBits,{'double','int8'},{'real','integer','vector','binary'},
mfilename,'Data input');

        % Columnize and expand to a [1 Nu] cell
        dataCell_p = repmat({int8(dataBits(1:8384))}, 1, numUsers);
        dataCell_s = repmat({int8(dataBits(8385:16768))}, 1, numUsers);
    else % NDP
        % Empty data check if not NDP
        coder.internal.errorIf(any(cfgFormat.PSDULength(:) > 0) &&
isempty(dataBits),'wlan:wlanWaveformGenerator:NoData');

        dataCell = {int8(dataBits(:))};
    end
end

% Number of bits in a PSDU for a single packet (convert bytes to bits)
numPSDUBits = cfgFormat.PSDULength*8;
```

```
% Repeat to provide initial state(s) for all users and packets
scramInit = repmat(scramblerInit,1,numUsers/size(scramblerInit,2)); % For all users
pktScramInit = scramInit(mod((0:numPackets-1).',size(scramInit,1))+1, :);

% Get the sampling rate of the waveform
if isa(cfgFormat, 'wlanDMGConfig')
    if strcmp(phyType(cfgFormat),'OFDM')
        sr = 2640e6;
    else
        sr = 1760e6;
    end
    numTxAnt = 1;
    numPktSamples = s.NumPPDUSamples;
elseif inDSSSMode % DSSS format
    sr = 11e6;
    numTxAnt = 1;
    giType = ''; % for codegen
    FFTLen_1 = 0;  % for codegen
    info = wlan.internal.dsssInfo(cfgFormat);
    numPktSamples = info.NumPPDUSamples;

    lstf = []; % for codegen
    lltf = []; % for codegen
    lsig = []; % for codegen
else % OFDM format
    chanBW = cfgFormat.ChannelBandwidth;
    sr = wlan.internal.cbwStr2Num(chanBW)*1e6;
    switch class(cfgFormat)
        case 'wlanNonHTConfig'
            giType = 'Long';  % Always
            FFTLen_1 = 64;
        case 'wlanS1GConfig'
            giType = cfgFormat.GuardInterval;
            FFTLen_1 = (sr/2e6)*64;
        otherwise % For VHT/HT formats
            giType = cfgFormat.GuardInterval;
            FFTLen_1 = (sr/20e6)*64;
    end
    if any(strcmp(chanBW,{'CBW10','CBW5'}))
        numTxAnt = 1;  % override and set to 1 only, for 802.11j/p
    else
        numTxAnt = cfgFormat.NumTransmitAntennas;
    end
    numPktSamples = real(s.NumPPDUSamples); % real for codegen

    % Generate the legacy preamble fields for applicable formats
    if ~isa(cfgFormat,'wlanS1GConfig')
        lstf = wlanLSTF(cfgFormat);
        lltf = wlanLLTF(cfgFormat);
        lsig = wlanLSIG(cfgFormat);
    end
end

if isa(cfgFormat,'wlanVHTConfig') % VHT format
    % VHT Format
    vhtsiga = wlanVHTSIGA(cfgFormat);
    vhtstf = wlanVHTSTF(cfgFormat);
    vhtltf = wlanVHTLTF(cfgFormat);
    vhtsigb = wlanVHTSIGB(cfgFormat);
```

```
    preamble = [lstf; lltf; lsig; vhtsiga; vhtstf; vhtltf; vhtsigb];

elseif isa(cfgFormat,'wlanHTConfig') % HT-MF format

    htSig = wlanHTSIG(cfgFormat);
    htstf = wlanHTSTF(cfgFormat);
    htltf = wlanHTLTF(cfgFormat);
    preamble = [lstf; lltf; lsig; htSig; htstf; htltf];

elseif isa(cfgFormat,'wlanNonHTConfig')

    if strcmp(cfgFormat.Modulation, 'OFDM')
        preamble = [lstf; lltf; lsig];
    else % DSSS
        preamble = [wlan.internal.wlanDSSSPreamble(cfgFormat);
wlan.internal.wlanDSSSHeader(cfgFormat)];
    end

elseif isa(cfgFormat,'wlanS1GConfig')
    if ~strcmp(packetFormat(cfgFormat),'S1G-Long')
        stf = wlan.internal.s1gSTF(cfgFormat);
        [ltf1, ltf2n] = wlan.internal.s1gLTF(cfgFormat);
        sig = wlan.internal.s1gSIG(cfgFormat);
        preamble = [stf; ltf1; sig; ltf2n];
    else % Preamble == 'Long'
        stf = wlan.internal.s1gSTF(cfgFormat);
        ltf1 = wlan.internal.s1gLTF1(cfgFormat);
        siga = wlan.internal.s1gSIGA(cfgFormat);
        dstf = wlan.internal.s1gDSTF(cfgFormat);
        dltf = wlan.internal.s1gDLTF(cfgFormat);
        sigb = wlan.internal.s1gSIGB(cfgFormat);
        preamble = [stf; ltf1; siga; dstf; dltf; sigb];
    end
elseif isa(cfgFormat,'wlanDMGConfig')
    if strcmp(phyType(cfgFormat),'OFDM')
        % In OFDM PHY preamble fields are resampled to OFDM rate
        preamble = wlan.internal.dmgResample([wlan.internal.dmgSTF(cfgFormat);
wlan.internal.dmgCE(cfgFormat)]);
        brfields = wlan.internal.dmgResample([wlan.internal.dmgAGC(cfgFormat);
wlan.internal.dmgTRN(cfgFormat)]);
    else
        preamble = [wlan.internal.dmgSTF(cfgFormat);
wlan.internal.dmgCE(cfgFormat)];
        brfields = [wlan.internal.dmgAGC(cfgFormat);
wlan.internal.dmgTRN(cfgFormat)];
    end
end

if windowing
    % Calculate parameters for windowing
    % IdleSample offset due to windowing
    wlength = 2*ceil(winTransitTime*sr/2);
    bLen = wlength/2; % Number of samples overlap at the end of the packet
    if isa(cfgFormat,'wlanDMGConfig')
        % No waveform extension for the non-OFDM fields in the preamble
        aLen = 0;
        % No waveform extension due to windowing when BRP field is present
        bLen = bLen*(~wlan.internal.isBRPPacket(cfgFormat));
        windowedPktLength = numPktSamples+bLen;
    else
```

```
        aLen = bLen-1; % Number of samples overlap at start of packet
        windowedPktLength = numPktSamples+wlength-1;
    end
else
    % Define unused windowing variables for codegen
    wlength = 0;
    windowedPktLength = numPktSamples+wlength-1;
    aLen = 0;
    bLen = 0;
end

% Define a matrix of total simulation length
numIdleSamples = round(sr*idleTime);
pktWithIdleLength = numPktSamples+numIdleSamples;
txWaveform = complex(zeros(numPackets*pktWithIdleLength,numTxAnt));

% Extract PSDU for the current packet
    psdu_p = getPSDUForCurrentPacket(dataCell_p,numPSDUBits, i);
    psdu_s = getPSDUForCurrentPacket(dataCell_s,numPSDUBits, i);
    %psdu_mat = cell2mat(psdu);
    %psdu_length = length(psdu_mat);
    %psdu_p = psdu_mat(1:psdu_length/2);
    %psdu_s = psdu_mat(((psdu_length/2)+1):psdu_length);
    %psdu_p_cell = mat2cell(psdu_p,8384);
    %psdu_s_cell = mat2cell(psdu_s,8384);

for i = 1:numPackets

    % Generate the PSDU with the correct scrambler initial state
    if  isa(cfgFormat,'wlanVHTConfig')
        if any(cfgFormat.APEPLength > 0)
            data = wlanVHTData(psdu,cfgFormat,pktScramInit(i,:));
        else  % NDP
            data = complex(zeros(0,cfgFormat.NumTransmitAntennas));
        end
    elseif isa(cfgFormat,'wlanHTConfig') % HT-MF format
        if cfgFormat.PSDULength > 0
            data = wlanHTData(psdu{1},cfgFormat,pktScramInit(i,:));
        else  % NDP or sounding packet
            data = complex(zeros(0,cfgFormat.NumTransmitAntennas));
        end
    elseif isa(cfgFormat,'wlanNonHTConfig') % non-HT format
        if strcmp(cfgFormat.Modulation, 'OFDM')

            [mappedData, numSym] =
wlanNonHTData_NOMA(psdu_p{1},cfgFormat,pktScramInit(i,:));
            mappedData_p = mappedData;

            mappedData = wlanNonHTData_NOMA(psdu_s{1},cfgFormat,pktScramInit(i,:));
            mappedData_s = mappedData;

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %Power allocation
        k_pp = 1;
        k_ss = 0.4;
        mappedData_primary = k_pp.*mappedData_p;
        mappedData_secondary = k_ss.*mappedData_s;

        %Superposition
```

```
        mappedData = mappedData_primary + mappedData_secondary;
        % Determine number of symbols and pad length

        chanBW = cfgFormat.ChannelBandwidth; %cfgNonHT

        if (strcmp(chanBW, 'CBW10') || strcmp(chanBW, 'CBW5'))
        numTx = 1;  % override and set to 1 only, for 802.11j/p
        else
        numTx  = cfgFormat.NumTransmitAntennas;
        end


       cfgOFDM = wlan.internal.wlanGetOFDMConfig(chanBW, 'Long', 'Legacy');
%waveform gen
       FFTLen = cfgOFDM.FFTLength; %waveform gen
       CPLen  = cfgOFDM.CyclicPrefixLength; %waveform gen


       z = 1; % Offset by 1 to account for HT-SIG pilot symbol
       pilotValues = wlan.internal.nonHTPilots(numSym,z);
       wout = complex(zeros((FFTLen + CPLen)*numSym, numTx));
       packedData = complex(zeros(FFTLen, 1));
       csh = wlan.internal.getCyclicShiftVal('OFDM', numTx, 20);
       for i = 1:numSym
      % Data packing with pilot insertion
      packedData(cfgOFDM.DataIndices, :) = mappedData(:,i);

      % Add pilots
      packedData(cfgOFDM.PilotIndices, :) = pilotValues(:,i);

      % Tone rotation and replicate over Tx
      pDataMat = repmat(packedData.*cfgOFDM.CarrierRotations, 1, numTx);

      % Cyclic shift applied per Tx
      dataCycShift =  wlan.internal.wlanCyclicShift(pDataMat, csh, FFTLen, 'Tx');

      % OFDM modulate - pCPLen
      wout((1:(FFTLen+CPLen)).' + (i-1)*(FFTLen+CPLen), :) = ...
        wlan.internal.wlanOFDMModulate(reshape(dataCycShift, FFTLen, 1, ...
          numTx), CPLen);
       end

      % Scale and output
      y = wout * cfgOFDM.NormalizationFactor / sqrt(numTx);

      data = y;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        else % DSSS
            data = wlan.internal.wlanDSSSData(psdu{1},cfgFormat);
        end
    elseif isa(cfgFormat,'wlanS1GConfig') % S1G format
        data = wlan.internal.s1gData(psdu,cfgFormat,pktScramInit(i,:));
    elseif isa(cfgFormat,'wlanDMGConfig') % S1G format
        % Header and data scrambled so generate for each packet together
```

```
        % Override scrambler initialization in configuration object if
        % supplied by the user to the waveform generator
        if overrideObjectScramInit
            cfgFormat.ScramblerInitialization = pktScramInit(i,:);
        end

        data = [wlan.internal.dmgHeader(psdu{1},cfgFormat);
wlan.internal.dmgData(psdu{1},cfgFormat); brfields];
    end




% Construct packet from preamble and data
    packet = [preamble; data];

    if windowing
        % Window each packet
        if isa(cfgFormat, 'wlanDMGConfig')
            windowedPacket = wlan.internal.dmgWindowing(packet, wlength, cfgFormat);
        else
            if isa(cfgFormat, 'wlanS1GConfig') && strcmp(cfgFormat.GuardInterval,
'Short')
                % For S1G the first data symbol is always Long GI
                numSamplesSymbol = 40e-6*sr; %40 us from Table 24.4 IEEE
P802.11ah/D5.0
                numSamplesBeforeShortGI = size(preamble,1)+numSamplesSymbol;
            else
                % For other formats short GI begins after the preamble
                numSamplesBeforeShortGI = size(preamble,1);
            end
            windowedPacket =
wlan.internal.wlanWindowing(packet,FFTLen_1,wlength,giType,numSamplesBeforeShortGI);
        end

        % Overlap-add the windowed packets
        if numPackets==1 && numIdleSamples==0 % Only one packet which wraps
            txWaveform = windowedPacket(aLen+(1:numPktSamples), :);
            % Overlap start of packet with end
            txWaveform(1:bLen,:) = txWaveform(1:bLen,:)+windowedPacket(end-
bLen+1:end,:);
            % Overlap end of packet with start
            txWaveform(end-aLen+1:end,:) = txWaveform(end-
aLen+1:end,:)+windowedPacket(1:aLen,:);
        else
                txWaveform(1:(numPktSamples+bLen),:) = windowedPacket(1+aLen:end,:);
                txWaveform(end-aLen+1:end,:) = windowedPacket(1:aLen,:);
       end
    else
        % Construct entire waveform
        txWaveform((i-1)*pktWithIdleLength+(1:numPktSamples),:) = packet;
    end
end
end

function psdu = getPSDUForCurrentPacket(dataCell,numPSDUBitsPerPacket,packetIdx)
    numUsers = length(dataCell); % == length(numPSDUBits)
    psdu = repmat({int8(1)},1,numUsers); % Cannot use cell(1, numUsers) for codegen
    for u = 1:numUsers
```

```
        idx = mod((packetIdx-1)*numPSDUBitsPerPacket(u)+(0:numPSDUBitsPerPacket(u)-
1).',length(dataCell{u})) + 1;
        psdu{u} = dataCell{u}(idx);
    end
end
```

***C. Function: "WLAN Non HT Data NOMA"***

```
function [mappedData, numSym] = wlanNonHTData_NOMA(PSDU,cfgNonHT,varargin)

%#codegen
    scramInit = varargin{1};
    % Validate scrambler init
    validateattributes(scramInit, {'double', 'int8'}, ...
    {'real', 'integer', 'nonempty'}, mfilename, 'Scrambler initialization');
    if isscalar(scramInit)
        % Check for correct range
        coder.internal.errorIf(any((scramInit<1) | (scramInit>127)), ...
            'wlan:wlanNonHTData:InvalidScramInit');

        scramInitBits = uint8(de2bi(scramInit, 7, 'left-msb')).';
    else
        % Check for non-zero binary vector
        coder.internal.errorIf( ...
        any((scramInit~=0) & (scramInit~=1)) || (numel(scramInit)~=7) || ...
        all(scramInit==0) || (size(scramInit,1)~=7), ...
        'wlan:wlanNonHTData:InvalidScramInit');

        scramInitBits = uint8(scramInit);
    end

% Validate inputs
% Validate the format configuration object
validateattributes(cfgNonHT, {'wlanNonHTConfig'}, ...
    {'scalar'}, mfilename, 'format configuration object');
% Only applicable for OFDM and DUP-OFDM modulations
coder.internal.errorIf( ~strcmp(cfgNonHT.Modulation, 'OFDM'), ...
                        'wlan:wlanNonHTData:InvalidModulation');

s = validateConfig(cfgNonHT);
         validateattributes(PSDU, {'double', 'int8'},...
         {'real', 'binary', 'size', [cfgNonHT.PSDULength*8 1]}, ...
          mfilename, 'PSDU input');

numSym = s.NumDataSymbols;
numPad = s.NumPadBits;

mcsTable = wlan.internal.getRateTable(cfgNonHT);
rate     = mcsTable.Rate;
numBPSCS = mcsTable.NBPSCS;
numCBPS  = mcsTable.NCBPS;
Ntail = 6;



%% Generate the data field
% SERVICE, Section 18.3.5.2, all zeros
serviceBits = zeros(16,1,'int8');
```

```
% Scramble padded data
%   [service; psdu; tail; pad] processing
paddedData = [serviceBits; PSDU; zeros(Ntail,1); zeros(numPad, 1)];
scrambData = wlanScramble(paddedData, scramInitBits);
% Zero-out the tail bits again for encoding
scrambData(16+length(PSDU) + (1:Ntail)) = zeros(Ntail,1);
% BCC Encoding
encodedData = wlanBCCEncode(scrambData, rate);
% BCC Interleaving
interleavedData = wlanBCCInterleave(encodedData, 'Non-HT', numCBPS);
% Constellation mapping
mappedData = wlanConstellationMap(interleavedData, numBPSCS);
% Reshape to form OFDM symbols
mappedData = reshape(mappedData, numCBPS/numBPSCS, numSym);
```

***D. Function: "WLAN Non HT Data Recover NOMA"***

```
function [bits_s, Secondary_cancelation, varargout] = wlanNonHTDataRecover_s( ...
    rxNonHTData, chanEst, noiseVarEst, cfgNonHT, varargin)

narginchk(4,5);
nargoutchk(0,3);

% Calculate CPE if requested
if nargout>2
    calculateCPE = true;
else
    calculateCPE = false;
end

% Non-HT configuration input self-validation
validateattributes(cfgNonHT, {'wlanNonHTConfig'}, {'scalar'}, mfilename, 'format
configuration object');
% Only applicable for OFDM and DUP-OFDM modulations
coder.internal.errorIf(~strcmp(cfgNonHT.Modulation, 'OFDM'),
'wlan:wlanNonHTDataRecover:InvalidModulation');
s = validateConfig(cfgNonHT);

% Validate rxNonHTData
validateattributes(rxNonHTData, {'double'}, {'2d','finite'}, 'rxHTData', 'Non-HT
OFDM Data field signal');
% Validate chanEst
validateattributes(chanEst, {'double'}, {'3d','finite'}, 'chanEst', 'channel
estimates');
% Validate noiseVarEst
validateattributes(noiseVarEst, {'double'},
{'real','scalar','nonnegative','finite'}, 'noiseVarEst', 'noise variance estimate');

% Optional recovery configuration input validation
if nargin == 5
    validateattributes(varargin{1}, {'wlanRecoveryConfig'}, {'scalar'}, mfilename,
'recovery configuration object');

    symOffset = varargin{1}.OFDMSymbolOffset;
    eqMethod  = varargin{1}.EqualizationMethod;
    pilotPhaseTracking = varargin{1}.PilotPhaseTracking;
else % use defaults
```

```
    symOffset = 0.75;
    eqMethod  = 'MMSE';
    pilotPhaseTracking = 'PreEQ';
end

numRx = size(rxNonHTData, 2);

mcsTable = wlan.internal.getRateTable(cfgNonHT);

numOFDMSym = s.NumDataSymbols;

% Get OFDM configuration
[cfgOFDM,dataInd,pilotInd] =
wlan.internal.wlanGetOFDMConfig(cfgNonHT.ChannelBandwidth, 'Long', 'Legacy');

% Cross validate inputs
numST = numel([dataInd; pilotInd]); % Total number of occupied subcarriers
coder.internal.errorIf(size(chanEst, 1) ~= numST,
'wlan:wlanNonHTDataRecover:InvalidNHTChanEst1D', numST);
coder.internal.errorIf(size(chanEst, 2) ~= 1,
'wlan:wlanNonHTDataRecover:InvalidNHTChanEst2D');
coder.internal.errorIf(size(chanEst, 3) ~= numRx,
'wlan:wlanNonHTDataRecover:InvalidNHTChanEst3D');

% Extract data and pilot subcarriers from channel estimate
chanEstData = chanEst(dataInd,:,:);
chanEstPilots = chanEst(pilotInd,:,:);

% Cross-validation between inputs
minInputLen = numOFDMSym*(cfgOFDM.FFTLength+cfgOFDM.CyclicPrefixLength);
coder.internal.errorIf(size(rxNonHTData, 1) < minInputLen,
'wlan:wlanNonHTDataRecover:ShortNHTDataInput', minInputLen);

% Processing
% OFDM Demodulation
[ofdmDemodData, ofdmDemodPilots] =
wlan.internal.wlanOFDMDemodulate(rxNonHTData(1:minInputLen, :), cfgOFDM, symOffset);

% Pilot phase tracking
if calculateCPE==true || strcmp(pilotPhaseTracking, 'PreEQ')
    % Get reference pilots, from IEEE Std 802.11-2012, Eqn 18-22
    z = 1; % Offset by 1 to account for L-SIG pilot symbol
    refPilots = wlan.internal.nonHTPilots(numOFDMSym, z);

    % Estimate CPE and phase correct symbols
    cpe = wlan.internal.commonPhaseErrorEstimate(ofdmDemodPilots, chanEstPilots,
refPilots);
    if strcmp(pilotPhaseTracking, 'PreEQ')
        ofdmDemodData = wlan.internal.commonPhaseErrorCorrect(ofdmDemodData, cpe);
    end
    if calculateCPE==true
        varargout{1} = cpe.'; % Permute to Nsym-by-1
    end
end

%%%%%%Demodulate as primary%%%%%%%%

% Equalization
```

```
[eqDataSym, csiData] = wlan.internal.wlanEqualize(ofdmDemodData, chanEstData,
eqMethod, noiseVarEst);

% Constellation demapping
qamDemodOut = wlanConstellationDemap(eqDataSym, noiseVarEst, mcsTable.NBPSCS);
% Apply bit-wise CSI and concatenate OFDM symbols in the first dimension
qamDemodOut = bsxfun(@times, ...
              reshape(qamDemodOut, mcsTable.NBPSCS, [], numOFDMSym), ...
              reshape(csiData, 1, [])); % [Nbpscs Nsd Nsym]
qamDemodOut = reshape(qamDemodOut, [], 1);

% Deinterleave
deintlvrOut = wlanBCCDeinterleave(qamDemodOut, 'Non-HT', mcsTable.NCBPS);

% Channel decoding
decBits = wlanBCCDecode(deintlvrOut, mcsTable.Rate);

% Derive initial state of the scrambler
scramInit = wlan.internal.scramblerInitialState(decBits(1:7));

% Remove pad and tail bits, and descramble
if all(scramInit==0)
    % Scrambler initialization invalid (0), therefore do not descramble
    descramDataOut = decBits(1:(16+8*cfgNonHT.PSDULength));
else
    descramDataOut = wlanScramble(decBits(1:(16+8*cfgNonHT.PSDULength)), scramInit);
end

% Remove the 16 service bits
bits = descramDataOut(17:end);
bits_cell = mat2cell(bits,8384);

% Repeat to provide initial state(s) for all users and packets
scramblerInit  = 43;
numUsers = 1;
numPackets = 1;
scramInit = repmat(scramblerInit,1,numUsers/size(scramblerInit,2)); % For all users
pktScramInit = scramInit(mod((0:numPackets-1).',size(scramInit,1))+1, :);

mappedData = wlanNonHTData_NOMA(bits_cell{1},cfgNonHT,pktScramInit(1,:));
est_signal_p = mappedData;
%Intereference Cancelation
Secondary_cancelation = eqDataSym - est_signal_p;
% Constellation demapping
qamDemodOut_s = wlanConstellationDemap(Secondary_cancelation, noiseVarEst,
mcsTable.NBPSCS);
% Apply bit-wise CSI and concatenate OFDM symbols in the first dimension
qamDemodOut_s = bsxfun(@times, ...
              reshape(qamDemodOut_s, mcsTable.NBPSCS, [], numOFDMSym), ...
              reshape(csiData, 1, [])); % [Nbpscs Nsd Nsym]
qamDemodOut_s = reshape(qamDemodOut_s, [], 1);

% Deinterleave
deintlvrOut_s = wlanBCCDeinterleave(qamDemodOut_s, 'Non-HT', mcsTable.NCBPS);

% Channel decoding
decBits_s = wlanBCCDecode(deintlvrOut_s, mcsTable.Rate);

% Derive initial state of the scrambler
```

```
scramInit_s = wlan.internal.scramblerInitialState(decBits_s(1:7));

% Remove pad and tail bits, and descramble
if all(scramInit_s==0)
    % Scrambler initialization invalid (0), therefore do not descramble
    descramDataOut_s = decBits_s(1:(16+8*cfgNonHT.PSDULength));
else
    descramDataOut_s = wlanScramble(decBits_s(1:(16+8*cfgNonHT.PSDULength)),
scramInit_s);
end

% Remove the 16 service bits
bits_s = descramDataOut_s(17:end);

end
```