Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

A High Performance FPGA Implementation of a Feed-Forward Carrier-Phase Recovery Algorithm for 16-QAM Real-Time Coherent Receivers

Διπλωματικη Εργασια

του

ΙΩΑΝΝΗ - ΒΑΤΙΣΤΑ ΚΩΣΤΑΛΑΜΠΡΟΥ

Επιβλέπων: Δημήτριος Σούντρης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εργαστηριο Μικροϋπολογιστων και Ψηφιακων Συστηματων

Αθήνα, Ιούλιος 2017

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Μικροϋπολογιστών και Ψηφιακών Συστημάτων

A High Performance FPGA Implementation of a Feed-Forward Carrier-Phase Recovery Algorithm for 16-QAM Real-Time Coherent Receivers

Διπλωματικη Εργασια

του

ΙΩΑΝΝΗ - ΒΑΤΙΣΤΑ ΚΩΣΤΑΛΑΜΠΡΟΥ

Επιβλέπων: Δημήτριος Σούντρης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Ιουλίου 2017.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Δημήτριος Σούντρης	Ηρακλής Αβραμόπουλος	Κιαμάλ Πεκμεστζή
Αν. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2017

(Υπογραφή)

.....

Κωσταλαμπρος Ιωαννης - Βατιστας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Μικροϋπολογιστών και Ψηφιακών Συστημάτων

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τους επιβλέποντες καθηγητές μου, τους κυρίους Δημήτριο Σούντρη και Ηρακλή Αβαμόπουλο που μου έδωσαν τη δυνατότητα να εκπονήσω ένα άκρως ενδιαφέρον θέμα διπλωματικής εργασίας που ¨παντρεύει' τον κόσμο των οπτικών τηλεπικοινωνιών με αυτό των ευέλικτων και υψηλών επιδόσεων υπολογιστικών συστημάτων. Η εργασία μου στην επιστημονική αυτή τομή μου προσέφερε πολλές απαντήσεις, ενώ μου έθεσε ακόμα περισσότερα ερωτήματα.

Επίσης, ευχαριστώ θερμά τους υποψήφιους Διδάκτορες Κωνσταντίνο Μαραγκό, Χρήστο Σπαθαράκη, τον Δόκτορα Γιώργο Λεντάρη και τον Δόκτορα Στέφανο Δρύ για την αμέριστη βοήθεια που μου προσέφεραν καθώς και για τις στοχευμένες παρατηρήσεις τους που με οδηγούσαν πιο κοντά στην αλήθεια. Ευχαριστώ επίσης όλα τα παιδιά των εργαστηριών PCRL και Microlab του Ε.Μ.Π που έκαναν πιο όμορφες τις μέρες που πέρασα εκεί δημιουργώντας ένα όμορφο κλίμα στο χώρο εργασίας τους.

Επιπλέον, δεν θα μπορούσα να μην αναφέρω τους γονείς μου Μαρία και Θανάση ,τον αδερφό μου Κώστα και όλη την οικογένειά μου. Ένα ευχαριστώ θα ήταν λίγο για την στήριξη και την αγάπη τους όλα αυτά τα χρόνια και δίχως αυτά τα εφόδια δεν θα ήμουν εδώ που είμαι τώρα.

Ένας άνθρωπος που μου στάθηκε όσο κανένας όλη την περίοδο εκπόνησης της εργασίας και θα ήθελα να του εκφράσω την αγάπη μου είναι η Ειρήνη. Όταν όλα πήγαιναν τέλεια, αλλά κυρίως όταν οι αντοχές μου λιγόστευαν ήταν εκεί συνοδοιπόρος και στήριγμα και το πόνημα αυτό το οφείλω σε αυτή. Για όλες τις μέρες που η σκέψη σου ήταν και είναι μαζί μου.

Σε μια μεγαλύτερη οικογένεια, αυτή που διάλεξα εδώ και πολλά χρόνια, στου φίλους μου από το Ναύπλιο και όχι μόνο θα ήθελα να πω ένα τεράστιο ευχαριστώ για ακόμα μια φορά. Τα ονόματα είναι πολλά αλλά γνωρίζετε πολύ καλά ό,τι αναφέρομαι σε εσάς που περάσαμε τόσα χρόνια τα μαζί και θα περάσουμε άλλα τόσα, δύσκολα και εύκολα.

Τέλος για όλους τους δασκάλους της ζωής μου και κυρίως για τον Δάσκαλο μου Θοδωρή θα ήθελα να εκφράσω την ευγνωμοσύνη μου. Μέσα από τη σχολή απέκτησα μια δεύτερη οικογένεια που μου δίδαξε πολλές αρετές αλλά κυρίως με έκανε καλύτερο άνθρωπο.

Σε όλους όσους δεν ανέφερα αλλά και σε όλους τους παραπάνω λέω ένα μεγάλο ευχαριστώ.

Abstract

The majority of the modern high speed (exceeding 100Gbps) fiber-optic networks utilize both high throughput and higher order modulation schemes to cope with the bandwidth hungry modern telecommunication needs. A key enabler of such systems is coherent detection and at the heart of a coherent detection receiver lies the Carrier Recovery sub-module.

While the bandwidth efficiency increases a lot with higher constellations usage (16-QAM, 64-QAM, 256-QAM etc) the noise tolerance of the link becomes even more critical and Carrier Recovery requires state-of-the-art hardware solutions. Unlike in wireless communications where the Carrier Recovery is accomplished with Digital Phase Locked Loops (PLL) in high speed optical networks there is the need of high-performance and high-parallelism hardware to keep up with the increased speeds and overcome the barrier of the limited CMOS circuit speeds.

The current thesis aims at implementing a high efficiency, feed-forward Carrier Phase Recovery algorithm on an FPGA platform, targeted at 16-QAM real-time Digital Coherent Receivers. The highly-efficient FPGA platforms give us the opportunity to boost up the speed of our design by supporting an big external parallelization order.

In the beginning, we compare the industry benchmarks Carrier Phase Recovery algorithms (Viterbi-Viterbi, Blind Phase Search) with a less popular but highly-efficient alternative (NLS Estimator). During the comparison stage, we derive its optimal parameters (filter's type, filter's length) through simulation. To facilitate its efficient hardware implementation we perform a polynomial approximation on the core element of the algorithm.

After implementing the algorithm on the FPGA platform we perform various measurements to evaluate the system's performance and efficiency.

From the performance investigation we have found that the NLS Estimator algorithm can be successfully ported on the Virtex-7 Family FPGA platform and it can provide us with high throughput beyond 46 Gbaud net rate. The system also exhibits good linewidth tolerance since all of the hardware configurationswe tested presented ..... SNR penalty for 28 GBd and for linewidth between 200 kHz and 2 MHz which is a safe margin covering more than the bandwidth of modern External Cavity Lases. All things considered our system implementation can support not only the current state-of-the-art networks but also the tomorrow's bandwidth demanding ones (300Gbps +) while yielding efficient performance and increased flexibility.

Keywords

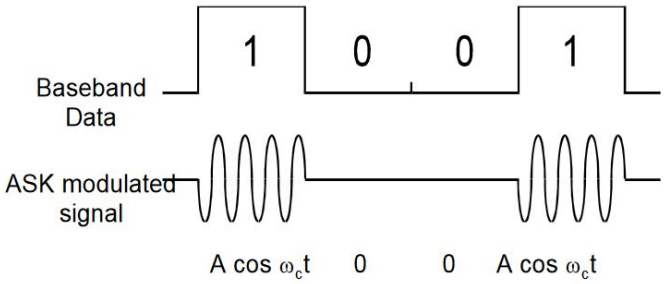
FPGA, VHDL, Carrier Phase Recovery, 16-QAM Modulation, Coherent detection, Hardware Efficiency, Performance, Precision, Parallel Architecture, BER, SNR Penalty, Viterbi-Viterbi, Blind Phase Search, Linewidth

Περιληπτική Απόδοση

Τα τελευταία χρόνια η χρήση του Internet και των διαδικτυακών πόρων (cloud services, mobile internet, online gaming) παρουσιάζει εντυπωσιακή αύξηση. Ταυτόχρονα, παρατηρείται μια αλλαγή στη φύση της κίνησης τόσο η απαίτηση για γρηγορότερη πρόσβαση όσο και η χρήση φορητών συσκευών την έχουν καταστήσει πολύ πιο δυναμική αλλά και απρόβλεπτη. Σε αυτό το πλαίσιο, η χρήση οπτικών επικοινωνιών καθίσταται απαραίτητη ώστε να καλυφθούν οι απαιτήσεις των σημερινών αλλά και μελλοντικών τηλεπικοινωνιακών δικτύων. Πράγματι, τα σύγχρονα οπτικά δίκτυα προσφέρουν πολύ μεγαλύτερες ταχύτητες μεταφοράς δεδομένων (100-400 Gbit/s ανά μήκος κύματος) και αποδοτικότερη χρήση του διαθέσιμου εύρους ζώνης (bandwidth) σε σχέση με τις συμβατικές τηλεπικοινωνιακές ζεύξεις (ενσύρματα, ασύρματα). Επιπλέον, η υιοθέτηση καινοτόμων τεχνολογιών και αρχιτεκτονικών στα σύγχρονα οπτικά δίκτυα (coherent optical communication, flexible optical networks) προσφέρει τη δυνατότητα για πιο ευέλικτα οπτικά δίκτυα ικανά να διαχειριστούν την δυναμική και αυξανόμενη διαδικτυακή κίνηση. Βασικό στοιχείο ενός ευέλικτου οπτικού δικτύου είναι οι ευέλικτοι, επαναρυθμιζόμενοι οπτικοί πομποδέκτες (flexible optical transceivers), ικανοί να διαχειριστούν δυναμικά τους διαθέσιμους δικτυακούς πόρους ανάλογα με τις απαιτήσεις της τηλεπικοινωνιακής κίνησης. Για την υλοποίηση ενός τέτοιου πομποδέκτη είναι απαραίτητη η χρήση ψηφιακών ηλεκτρονικών (FPGAs, ASICs) καθώς και η εφαρμογή της ψηφιακής επεξεργασίας σήματος (DSP).

Οι σύγχρονοι (coherent) τηλεπικοινωνιακοί δέκτες μπορούν να πολλαπλασιάσουν το ρυθμό μετάδοσης δεδομένων στο ήδη εγκατεστημένο οπτικό δίκτυο. Αυτό καθίσταται δυνατό μέσω της χρήσης υψηλότερων ρυθμών διαμόρφωσης.

Οι κλασσικές οπτικές ζεύξεις κωδικοποιούν την πληροφορία στο φέρον σήμα μέσω της διαμόρφωσης πλάτους (ASK εικόνα 2.2), δηλαδή αναπαριστούν την ψηφιακή δυαδική πληροφορία (0 ή 1) με διαφορετικό πλάτος (δηλαδή τάση) στο μεταδιδόμενο ηλεκτρομαγνητικό κύμα. Με αυτό τον τρόπο, ο δέκτης χρησιμοποιώντας μια απλή διάταξη φωτοδιόδου μπορεί να μεταφράσει την εισερχόμενη πληροφορία συγκρίνοντας την προσπίπτουσα τάση με ένα συγκεκριμένο και προκαθορισμένο κατώφλι.



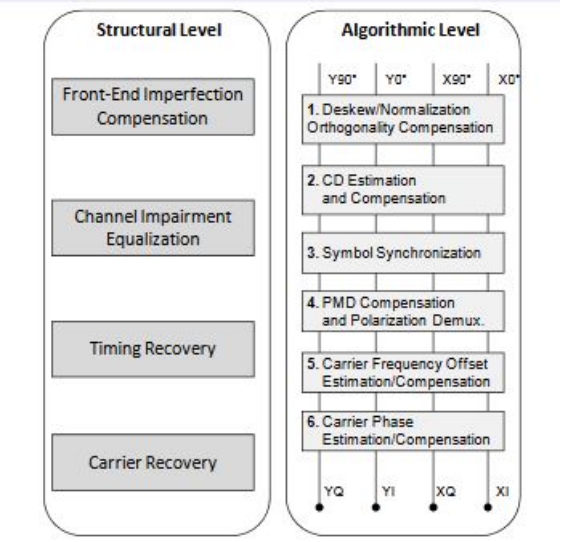
Σχήμα 1: Αρχή λειτουργίας της διαμόρφωσης πλάτους ASK

Επιπρόσθετα, με την χρήση υψηλότερων ρυθμών διαμόρφωσης μπορούμε πλέον να κωδικοποι-

ήσουμε πληροφορία όχι μόνο στο πλάτος του H/M κύματος αλλά και στη φάση και τη συχνότητα του. Έτσι, ενώ μέσω της οπτικής ίνας μεταδίδεται ο ίδιος αριθμός κυμάτων, στην πράξη λαμβάνουμε περισσότερη πληροφορία.

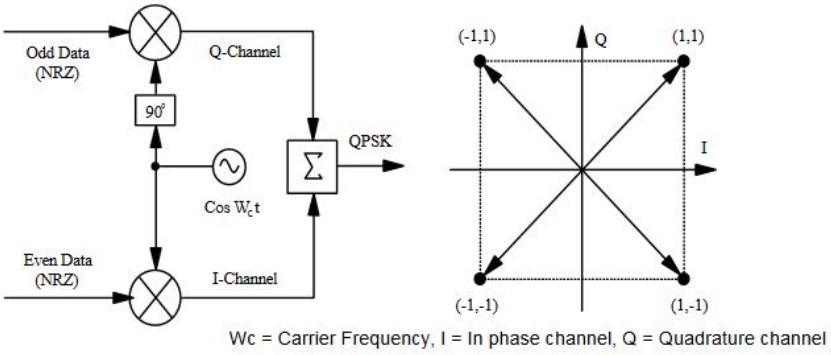
Βέβαια, για την σωστή και έγκαιρη αποκωδικοποίηση αυτής της πληροφορίας απαιτείται η χρήση πολυπλοκότερων κυκλωμάτων στο δέκτη. Ιδανικά, με τη χρησιμοποίηση Ψηφιακής επεξεργασίας σήματος ο δέκτης λαμβάνει το σύνολο της πληροφορίας του μεταδιδόμενου κύματος και μπορεί πλέον να το επεξεργαστεί και να πραγματοποιήσει αντιστάθμιση φαινομένων όπως (Chromatic Dispersion (CD), Polarization Mode Dispersion (PMD)) καθώς και να λάβει τη μεταδιδόμενη διαμορφωμένη πληροφορία από τη φάση και τη συχνότητα.

Παρακάτω βλέπουμε ενα παράδειγμα σύγχρονου δέκτη που χρησιμοποιεί αμιγώς ψηφιακή επεξεργασία σήματος για την αποθορυβοποίηση και την ανάκτηση του σήματος 2.1.



Σχήμα 2: Ενδεικτικά, Οι βασικές δομικές μονάδες ενός σύγχρονου δέκτη που χρησιμοποιεί ψηφιακή επεξεργασία σήματος.

Όσον αφορά τους υψηλότερους ρυθμούς κωδικοποίησης η πλέον ευρέως χρησιμοποιούμενη τεχνική είναι η M-QAM κωδικοποίηση. Στην ουσία κάθε κύμα περιέχει πληροφορία τόσο στο πλάτος του όσο και στη φάση του. Μια ειδική μορφή αυτής της κωδικοποίησης βλέπουμε παρακάτω στην εικόνα (2.5). Στην συγκεκριμένη κωδικοποίηση, κάθε κύμα φέρει δύο βαθμούς διαμόρφωσης κατά φάση και έναν κατά πλάτος. Κωδικοποιώντας δηλαδή την πληροφορία σε τέσσερις καταστάσεις μπορούμε να μεταδώσουμε 2 bit ανά H/M κύμα.

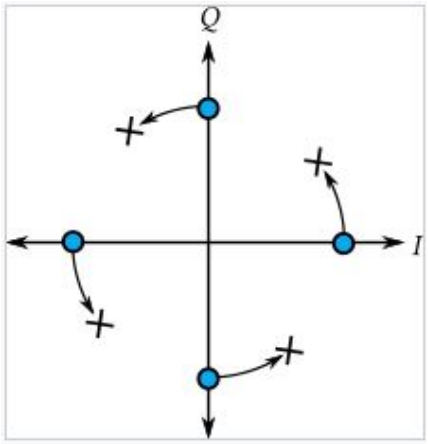


Σχήμα 3: Αρχή λειτουργίας της κωδικοποίησης QPSK

Βέβαια, πέρα απο την QPSK κωδικοποίηση υπάρχουν και κωδικοποιήσεις υψηλότερων ρυθμών. Τέτοιες μπορεί να είναι οι (16-QAM, 64-QAM, 256-QAM κοκ). Σε αυτές κωδικοποιούνται 4, 5, 6 bits αντίστοιχα σε κάθε H/M κύμα. Όσο μεγαλύτερος είναι ο ρυθμός κωδικοποίησης τόσο μικρότερο είναι το επιτρεπτό περιθώριο θορύβου καθώς για την ίδια ισχύ περισσότερα σύμβολα κωδικοποιούνται στον ίδιο χώρο με αποτέλεσμα να μειώνεται η μεταξύ τους απόσταση.

Όσον αφορά τον εισερχόμενο θόρυβο στο σήμα μας έχουμε κάνει τις εξής παραδοχές. Λαμβάνουμε υπόψιν μάς μόνο το θόρυβο φάσης, ο οποίος προέρχεται από την ασυμφωνία φάσης των laser καθως και τον εισερχόμενο από το περιβάλλον θόρυβο ο οποίος μοντελοποιείται ως Additive White Gaussian Noise (AWGN).

Καθώς, όπως προείπαμε, κάθε laser διαφέρει σημαντικά από κάθε άλλο ως προς την εκπεμπόμενη ακτινοβολία το σύστημα υποφέρει από τον λεγόμενο θόρυβο φάσης. Ακόμα και για laser που έχουν αρκετά παραπλήσια συχνότητα , οι μικρές διαφορές σε αυτήν οδηγούν σε αλλοίωση του σήματος κατά την ανάκτηση του στο δέκτη. Η αλλοίωση αυτή είναι ορατή με τη μορφή ¨στριψίματος' του αστερισμού διαμόρφωσης (4)(Ως αστερισμός διαμόρφωσης εννοείται η αναπαράσταση του συνόλου του αλφάβητου των μεταδιδόμενων σημάτων σε πολικές συντεταγμένες).



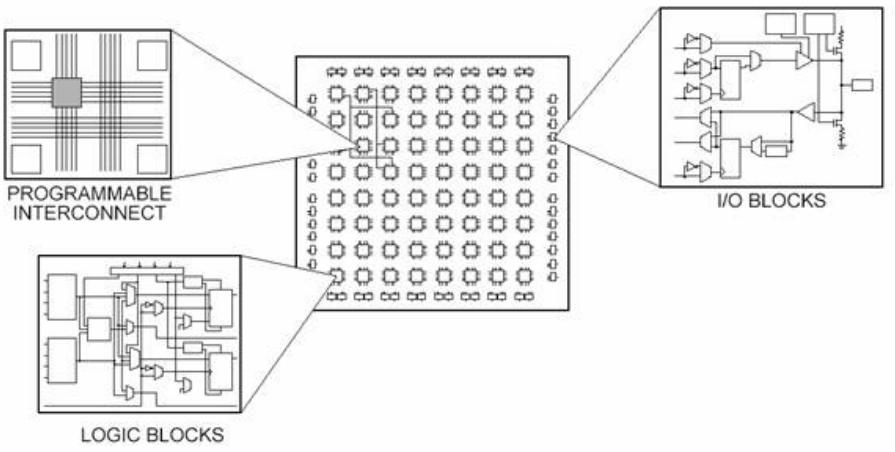
Σχήμα 4: Σήμα αλλοιωμένο με θόρυβο φάσης

Είδαμε λοιπόν την πρακτική εφαρμογή της σύγχρονης διαμόρφωσης καθώς και το μεγάλο κέρδος που αυτή δίνει σε ένα οπτικό τηλεπικοινωνιακό σύστημα. Η αλλοίωση της φάσης του φέροντος όμως καθιστά επιτακτική την αντιστροφή αυτού του φαινομένου ώστε να υπάρχει αποδοτική ανάκτηση της πληροφορίας. Καθώς οι ταχύτητες μετάδοσης πληροφορίας στα σύγχρονα τηλεπικοινωνιακά δίκτυα

γίνονται όλο και πιο γρήγορες τα αναλογικά ηλεκτρονικά συστήματα δεν μπορούν να προσαρμοστούν στους υψηλούς ρυθμούς μετάδοσης. Ακόμα και οι κλασσικοί τρόποι ανάκτησης φάσης, όπως οι βρόχοι ανάδρασης Costas, αποδεικνύονται εξίσου ανίκανοι να ακολουθήσουν τις σύγχρονες ιδιαίτερα υψηλές ταχύτητες.

Συνυπολογίζοντας όλα τα προεκτεθέντα, καταδεικνύεται η ανάγκη για την χρησιμοποίηση συστημάτων υψηλής ταχύτητας και ακόμα υψηλότερης παραλληλίας της επεξεργασίας. Τέτοιες λύσεις αποτελούν οι πλατφόρμες FPGA οι οποίες έχουν, πλέον, έχουν μεγάλο βαθμό ενσωμάτωσης στα σύγχρονα τηλεπικοινωνιακά συστήματα υψηλής απόδοσης.

Τα FPGA είναι στην ουσία ένας συγκερασμός λογισμικού και υλικού. Η δυνατότητα αναπρογραμματισμένος τους, ακόμα και κατά τη διάρκεια λειτουργίας τους τα καθιστούν τις πλέον βέλτιστες λύσεις σε εφαρμογές με δυναμική ανάθεση πόρων όπως είναι οι σύγχρονές τηλεπικοινωνιακές εφαρμογές. Παρακάτω (εικόνα 2.11) βλέπουμε την αρχιτεκτονική ενός FPGA.



Σχήμα 5: Μια αφαιρετική αναπαράσταση της αρχιτεκτονικής ενός FPGA. Στο σχήμα φαίνονται τα μέρη CLB, IOB καθώς και οι προγραμματιζόμενες εσωτερικές διασυνδέσεις

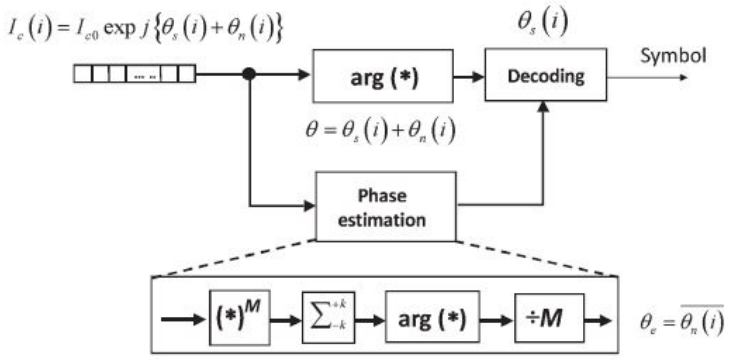
Τα συστατικά στοιχεία ενός FPGA αρχικά διατάσσονται και τροποποιούνται απο τον χρήστη με προγραμματιστικό τρόπο. Έπειτα μέσω της ¨εγγραφής' του ψηφιακού σχεδίου στο FPGA υλοποιούμε στην ουσία ένα ASIC ολοκληρωμένο κύκλωμα ειδικώς προσαρμοσμένο στην λειτουργία που επιθυμούμε. Με αυτό τον τρόπο είναι εφικτό να επιτύχουμε πολύ μεγάλη παραλληλία και κατ επέκταση πολύ μεγάλες ταχύτητες επεξεργασίας των δεδομένων σε σύγχρονα οπτικά τηλεπικοινωνιακά δίκτυα. Το επιστημονικό αυτό πεδίο τα τελευταία χρόνια εξερευνάται εντατικά, καθώς οι ταχύτητες που απαιτούνται στην επεξεργασία και τη μεταφορά δεδομένων χρήζουν αρτιότερων τεχνικών αποθορυβοποίησης, οι οποίες καθίστανται δυνατές μέσα από την χρήση των αρχιτεκτονικών FPGA.

Η παρούσα εργασία ασχολείται με την μελέτη και την υλοποίηση σε αρχιτεκτονική FPGA ενός αλγορίθμου ανάκτησης της φάσης του φέροντος κύματος, που στοχεύει σε σύγχρονα τηλεπικοινωνιακά συστήματα υπέρ-υψηλών ταχυτήτων (100Gbps - 400Gbps) .

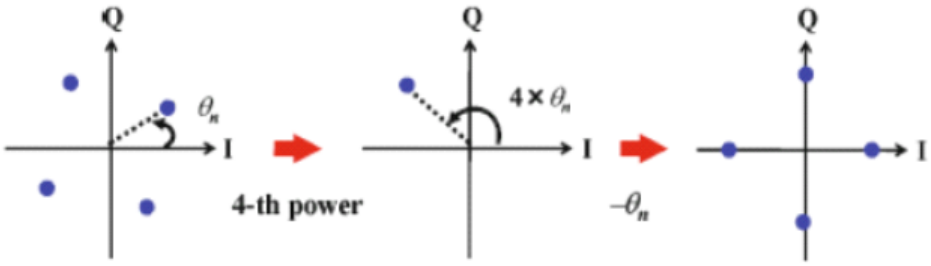
Υπάρχει πληθώρα τεχνικών και αλγορίθμων ανάκτησης της φάσης του φέροντος κύματος. Μια ειδοποιός διαφορά των αλγορίθμων που μελετάμε στην παρούσα, είναι το γεγονός οτι κανένας αλγόριθμος δεν περιέχει βρόχους ανάδρασης. Το χαρακτηριστικό αυτό καθιστά τους υπό εξέταση αλγορίθμους ικανούς να ενσωματωθούν σε ενα υψηλής απόδοσης τηλεπικοινωνιακό σύστημα, εν αντιθέσει με αυτούς που περιέχουν βρόχους ανάδρασης, όπως ο πολύ δημοφιλής Costas Loop.

Μια ευρέως διαδεδομένη τεχνική, ειδικότερα για διαμορφώσεις QPSK αποτελεί ο αλγόριθμος Viterbi-Viterbi [1]. Το σχηματικό διάγραμμα και η αρχή λειτουργίας του παρουσιάζονται στις παρα-

κάτω εικόνες 3.1 και3.2



Σχήμα 6: Σχηματικό διάγραμμα του αλγορίθμου. VV4PE



Σχήμα 7: Αρχή λειτουργίας του αλγορίθμου VV4PE. Με την ύψωση στην τέταρτη δύναμη αφαιρούμε την διαμόρφωση από το σήμα και μπορούμε να εκτιμήσουμε την απόκλιση της φάσης του.

Για να νοηθεί δίκαιη, η επιχειρούμενη με την παρούσα μελέτη σύγκριση της παραγόμενης ακρίβειας μεταξύ των αλγορίθμων, θα στοχεύσουμε αρχικά στην βελτιστοποίηση των αποτελεσμάτων τους. Αυτό θα επιτευχθεί μέσω της βέλτιστης ρύθμισης των παραμέτρων που τις χαρακτηρίζουν. Οι παράμετροι με τις οποίες θα ασχοληθούμε για τον κάθε αλγόριθμο είναι οι παρακάτω:

- Τυπος Φίλτρου
    1. Φίλτρο κινούμενου μέσου
    2. Φίλτρο Μπλόκ
    3. Φίλτρο Wiener
- Μέγεθος Φίλτρου

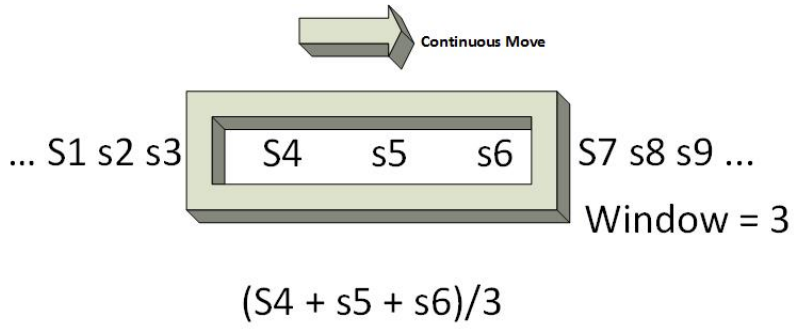
Για τον συγκεκριμένο αλγόριθμο VV4PE, όπως και για τους επόμενους αποδείχθηκε πειραματικά ότι το φίλτρο Wiener έχει την καλύτερη αποκριση τόσο στο θορυβο φάσης όσο και στο επίπεδο του σηματοθορυβικού λόγου.

Το φίλτρο αυτό αποτελεί στην ουσία ένα πεπερασμένο φίλτρο FIR του οποίου οι σταθερές αποτελούν αριθμητικές σειρές εκθετικά μειούμενες και συμμετρικές ως προς το κεντρικό σημείο του φίλτρου. Η επιτυχία του έγκειται στο γεγονός ό,τι η μειούμενη σημασία που δίνεται στα σύμβολα που

απέχουν χρονικά από το κεντρικό σημείο εξασφαλίζει επιτυχές φιλτράρισμα του θορύβου και συνεχή ανίχνευση των αλλαγών που προκαλούνται από το θόρυβο φάσης.

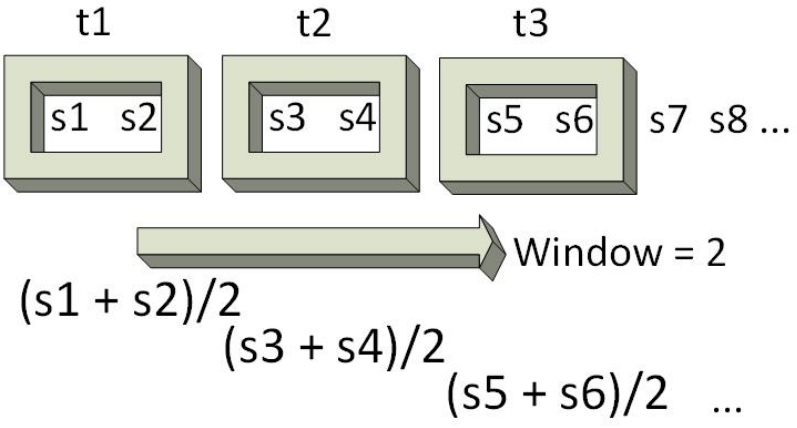
Βέβαια, ή χρήση του σε πραγματικά συστήματα δεν συνηθίζεται καθώς παρά την επιτυχή λειτουργία του, απαιτεί μεγάλη πολυπλοκότητα στον σχεδιασμό του.

Τα φίλτρα μπλοκ και κινούμενου μέσου παρουσιάζονται σχηματικά στις παρακάτω εικόνες (3.4, 3.5).



Σχήμα 8: Αρχη λειτουργίας του φίλτρου κινούμενου μέσου

Το φίλτρο κινούμενου μέσου παράγει ξεχωριστό αποτέλεσμα για κάθε καινούριο σύμβολο που επεξεργάζεται και ανάλογα με το μέγεθος του μπορεί να ακολουθεί σε μεγαλύτερο ή μικρότερο βαθμό την αλλαγή φάσεων. Σε γενικές γραμμές δεν έχει πολύπλοκη υλοποίηση και γιαυτό προτιμάται σε εφαρμογές ψηφιακής επεξεργασίας σήματος.



Σχήμα 9: Αρχη λειτουργίας του φίλτρου μπλόκ

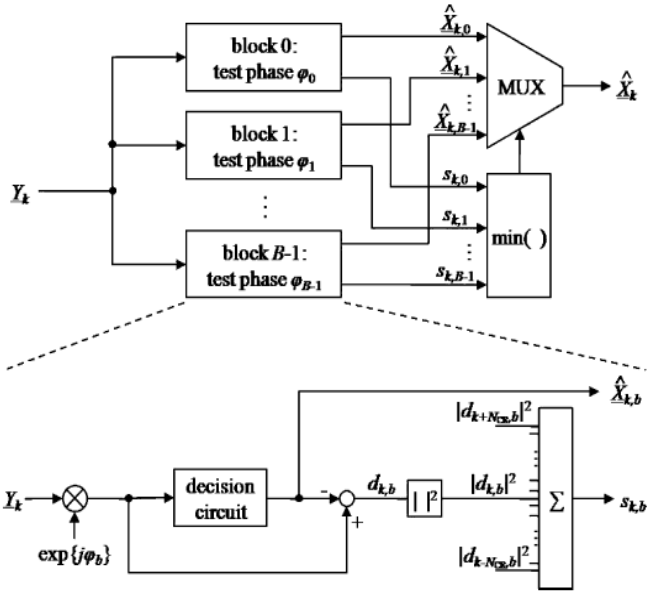
Το μπλοκ φίλτρο είναι ακόμα ένα απλό στην υλοποίηση του φίλτρο, που όμως παράγει αρκετά καλά αποτελέσματα όσον αφορά την αποθορυβοποίηση. Η απόκρισή του βέβαια στον θόρυβο φάσης είναι χειρότερη από την αντίστοιχη του φίλτρου κινούμενου μέσου.

Όπως προείπαμε λοιπόν, ο αλγόριθμος VV4PE αποτελεί ιδανική λύση, χαμηλής πολυπλοκότητας για σήματα με διαμόρφωση QPSK. Καθώς η διαμόρφωση μεγαλώνει ο αλγόριθμος αδυνατεί να ανιχνεύσει το λάθος φάσης αφού πλέον τα σύμβολα του αστερισμού δεν αποβάλλουν τη διαμόρφωση τους με ύψωση στην τέταρτη δύναμη. Η λειτουργία του αλγορίθμου σε τέτοιες περιστάσεις απαιτεί πολύ

μεγάλο μέγεθος φίλτρου για την εξαγωγή αξιόλογων αποτελεσμάτων. Αυτό βέβαια αποτελεί μεγάλο μειονέκτημα σε μια πιθανή υλοποίηση σε πλατφόρμα FPGA.

Ένας αλγόριθμος ανάκτησης φάσης φέροντος ο οποίος θεωρείται σημείο αναφοράς, είναι ο αλγόριθμος Blind Phase Search (BPS) [2]. Αποτελεί, μάλιστα συχνό σημείο σύγκρισης σε επιστημονικές μελέτες της τεχνικής ανάκτησης φάσης λόγω των πολύ καλών αποτελεσμάτων του.

Όσον αφορά τη λειτουργία του, ο (BPS) αρχικά διαιρεί το πολικό επίπεδο σε φ_b γωνίες ελέγχου. Το σύμβολο περιστρέφεται φ_b φορές και για κάθε μια από αυτές τις γωνίες υπολογίζεται η απόσταση από το κοντινότερο σύμβολο του αστερισμού. Έπειτα για να αφαιρεθεί η παραμόρφωση του σήματος λόγω θορύβου, προσθέτουμε την απόσταση N διαδοχικών συμβόλων που έχουν περιστραφεί κατά την ίδια γωνιά. Η βέλτιστη γωνία βρίσκεται υπολογίζοντας το ελάχιστο άθροισμα αυτών των αποστάσεων. Έπειτα εφαρμόζουμε αυτή τη γωνία σε κάθε ένα από τα N διαδοχικά σύμβολα. Στην παρακάτω εικόνα (3.12) φαίνεται το σχηματικό διάγραμμα του αλγορίθμου.



Σχήμα 10: Σχηματική αναπαράσταση του αλγορίθμου BPS χρησιμοποιώντας φ_b γωνίες ελέγχου.

Για τον (BPS) βρέθηκε έπειτα από προσομοιώσεις πώς η ιδανική τιμή για το πλήθος των γωνιών ελέγχου είναι $B = 10$ ενώ το μέγεθος του φίλτρου εξαρτάται από το μέγεθος του θορύβου στο σύστημα.

Συγκεκριμένα το μέγεθος του φίλτρου που δίνει τα βέλτιστα αποτελέσματα εξαρτάται άμεσα από το επίπεδο φασικού θορύβου. Ως εκ τούτου, για χαμηλά επίπεδα φασικού θορύβου, επιλέγουμε μεγάλα μεγέθη φίλτρων καθώς έτσι μπορούμε να αποβάλλουμε τον επιπρόσθετο AWGN θόρυβο από το σύστημα. Αντίθετα, σε περιπτώσεις υψηλού φασικού θορύβου χρειαζόμαστε μικρά φίλτρα έτσι ώστε να ανιχνεύουμε επαρκώς τις ταχύτατες αλλαγές της φάσης του φέροντος.

Το μεγάλο πλεονέκτημα του (BPS) είναι το γεγονός ότι μπορεί να επεξεργαστεί τον θόρυβο σε οποιοδήποτε σύστημα ανεξαρτήτως του ρυθμού διαμόρφωσης (16-QAM, 32-QAM κοκ). Έτσι ο

δέκτης καθίσταται πιο ευέλικτος και προσαρμόζεται στις εκάστοτε απαιτήσεις της τηλεπικοινωνιακής ζεύξης. Βέβαια, είναι άξιο αναφοράς πώς ο συγκεκριμένος αλγόριθμος είναι αρκετά πολύπλοκος καθώς για κάθε σύμβολο απαιτούνται πολυάριθμες πράξεις αφαιρέσεων και προσθέσεων, οι οποίες μάλιστα αυξάνουν ανάλογα με την τιμή της παραμέτρου φ_b καθώς και με τον υποκείμενο ρυθμό διαμόρφωσης.

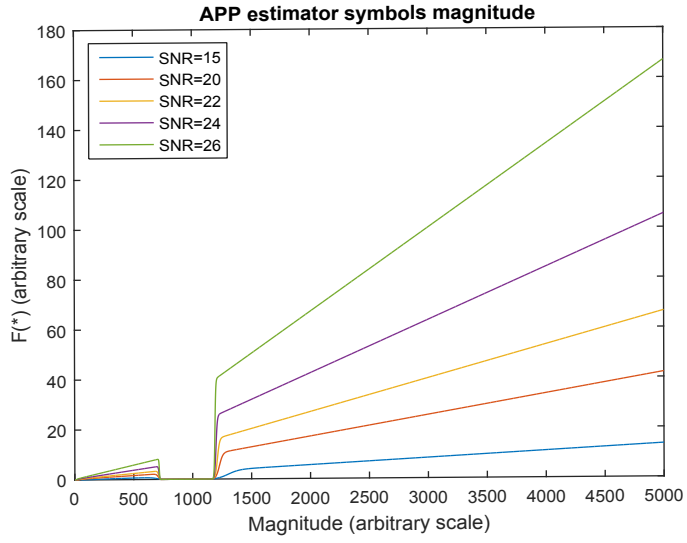
Τέλος θα αναφερθούμε σε εναν αλγόριθμο που βασίζεται στον VV4PE τροποποιώντας τον βέβαια ανάλογα, έτσι ώστε να καθίσταται δυνατή η χρησιμοποίησή του σε διαμορφώσεις 16QAM [3]. Ο αλγόριθμός αυτός ονομάζεται NLS Estimator και χρησιμοποιεί την ιδέα της τμηματοποίησης του αστερισμού 16QAM. Πιο συγκεκριμένα τα σύμβολα που μπορούν να συμβάλλουν στην εκτίμηση της φάσης του φέροντος (τα κοινά σύμβολα με τον αστερισμό QPSK) αποκτούν επιπλέον βάρος στην εκτίμηση της φάσης, ενώ το πλάτος των υπόλοιπων συμβόλων μηδενίζεται έτσι ώστε να μην υπολογίζεται στην εκτίμηση της φάσης. Η μη γραμμικότητα της συνάρτησης εξυπηρετεί επίσης στην παρακάτω λειτουργία. Καθώς ο θόρυβος AWGN επιδρά σε μικρότερο βαθμό στα σύμβολα μεγάλου πλάτους είναι επόμενο τα τελευταία να δίνουν ακριβέστερη εκτίμηση του φασικού θορύβου και ως εκ τούτου, δίνοντας σε αυτά μεγαλύτερο βάρος έχουμε καλύτερα αποτελέσματα. Η οικογένεια των μη γραμμικών συναρτήσεων παράγεται από την παρακάτω συνάρτηση.

$$F(\rho[n]) = \frac{g_2(\rho[n])}{g_1(\rho[n]) - g_3(\rho[n])} \tag{1}$$

όπου

$$g_1(\rho[n]) = (-1)^{i-1}\frac{8\rho[n]}{M\sigma^2}e^{(-\rho^2[n]/\sigma^2)}\sum_{\rho_\sigma,\phi_\sigma\in\beta}[cos(4(i-1)\phi_\sigma)e^{(-\rho_\sigma^2/\sigma^2)}I_{4(i-1)}(\frac{2\rho[n]\rho_c}{\sigma^2})] \tag{2}$$

όπου σ^2 είναι η συνδιακύμανση του θορύβου AWGN, $I_n(z)$ είναι η n^{th} βαθμού μετασχηματισμένη συνάρτηση Bessel του πρώτου είδους και β είναι το σύνολο των M σημείων του αστερισμού. Στην παρακάτω εικόνα 11 βλέπουμε μερικές από της καμπύλες που παράγονται από την παραπάνω συνάρτηση και απονέμουν το κατάλληλο βάρος σε κάθε σύμβολο ανάλογα με το πλάτος του.

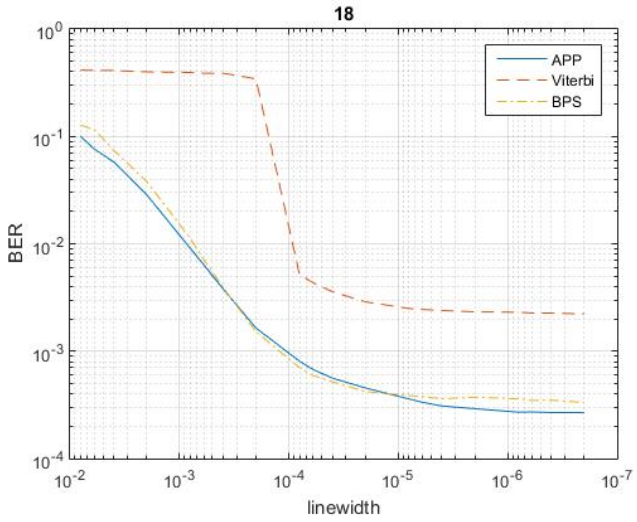


Σχήμα 11: Οικογένεια μη γραμμικών συναρτήσεων απονομής βάρους του αλγορίθμου NLS Estimator.

Και οι τρεις αλγόριθμοι που αναλύσαμε προηγουμένως δύνανται να υλοποιηθούν σε αρχιτεκτονική FPGA. Βέβαια, εφόσον πρόκειται για διαμόρφωση 16 QAM και εφαρμογή σε υψηλής ταχύτητας

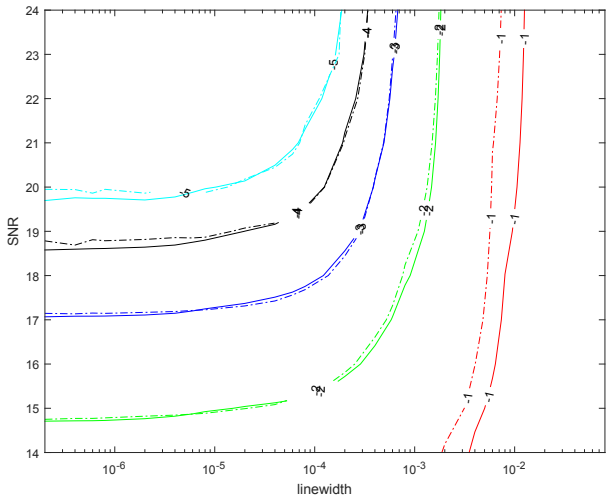
δίκτυα οι πρώτοι δύο αλγόριθμοι (VV4PE, BPS) δεν είναι ικανοί να δώσουν καλά αποτελέσματα, είτε λόγω των δομικών αδυναμιών τους(VV4PE)είτε λόγω της μεγάλης πολυπλοκότητας που απαιτεί η υλοποίηση τους (BPS).

Παρακάτω παρουσιάζουμε τη σύγκριση των τριών ανωτέρω αλγορίθμων μεταξύ τους 3.23 για ρυθμό διαμόρφωσης 16-QAM.



Σχήμα 12: Η επίδραση του φασικού θορύβου στους τρεις αλγορίθμους για σταθερό σηματοθορυβικό λόγο SNR= 20.

Είναι φανερή η διαφορά που εντοπίζεται στην ακρίβεια των αποτελεσμάτων και στην ανοχή στο επίπεδο φασικού θορύβου για τους τρεις αλγορίθμους. Αρχικά ο VV4PE δεν μας δίνει καθόλου καλά αποτελέσματα και απέχει αρκετά από τις καμπύλες των άλλων δυο λόγω της αδυναμίας του να λειτουργήσει σωστά για 16-QAM. Απο την άλλη ο NLS estimator φαίνεται να παρουσιάζει μεγάλη ανοχή στο φασικό θόρυβο και επιπλέον φαίνεται να ακολουθεί αρκούντως καλά την καμπύλη του BPS. Μπορούμε πλέον, εξαιρώντας από την σύγκριση τον VV4PE, να εξετάσουμε αναλυτικότερα την ακρίβεια των αποτελεσμάτων των συστημάτων που χρησιμοποιούν τους NLS estimator και BPS. Η σύγκριση αυτή φαίνεται στην παρακάτω εικόνα 3.24.

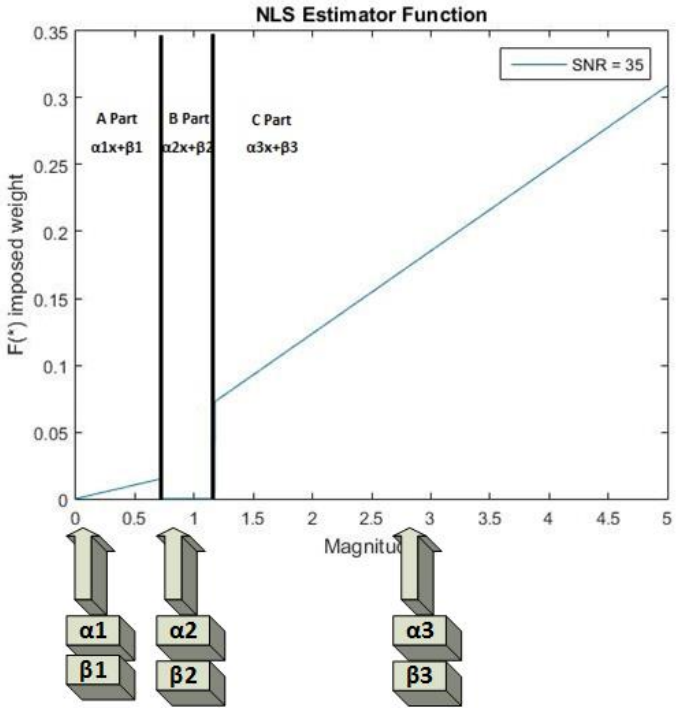


Σχήμα 13: Ισοδυναμικές καμπύλες του φασικού θορύβου σε σχέση με το σηματοθορυβικό λόγο για τους αλγορίθμους BPS (διακεκομμένη γραμμή) και NLS Estimmator (συνεχής γραμμή)

Βλέπουμε λοιπόν πώς η χρησιμοποίηση του αλγορίθμου NLS Estimator είναι ικανή να αποφέρει αποτελέσματα συγκρίσιμα με αυτά του BPS για σήματα με διαμόρφωση 16 QAM. Η ύπαρξη της πληθώρας μη γραμμικών καμπυλών βέβαια θα αποτελέσει σκόπελο στην αποδοτική υλοποίηση του αλγορίθμου στην πλατφόρμα FPGA για αυτό προτείνουμε την παρακάτω βασική τροποποίηση για λόγους ευκολίας υλοποίησης της εφαρμογής.

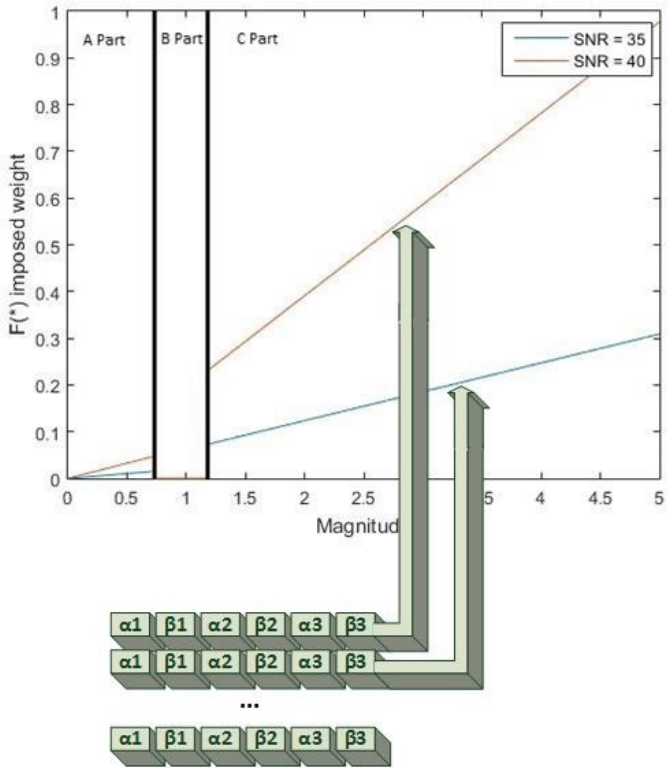
Η υλοποίηση της παραπάνω οικογένειας των μη γραμμικών συναρτήσεων στο υλικό θα ήταν δυνατή με την αποθήκευση τους με μια δομή μνήμης. Βέβαια ο πίνακας που περιέχει τα σημεία των καμπυλών έχει μέγεθος 301×5000. Αποτελείται από 301 μη γραμμικές καμπύλες (μια για κάθε τιμή του SNR απο 10 εώς 40 με λεπτομερειακότητα 0.1) οι οποίες λαμβάνουν 5000 διαφορετικές τιμές πλάτους στον χ άξονα που κυμαίνονται (από 0 έως 5 με λεπτομερεικότητα 0.001). Είναι φανερό ότι η προσπέλαση μιας τέτοιας μνήμης δεν επαρκεί για την επίτευξη ταχυτήτων ικανών να υποστηρίξουν τις σύγχρονες τηλεπικοινωνιακές υποδομές. Η απευθείας υλοποίηση της συνάρτησης στο υλικό φαίνεται εξίσου δύσκολη καθώς πρόκειται για μια τροποποιημένη συνάρτηση Bessel με έντονη μη γραμμικότητα.

Η ιδέα κλειδί πίσω από την απλούστευση την υλοποίησης είναι η πολυωνυμική προσέγγιση. Αν σκεφτούμε κάθε καμπύλη σαν μια τμηματικώς γραμμική συνάρτηση (αποτελούμενη από τρεις ευθείες) τότε μας δίνεται η δυνατότητα να προσεγγίσουμε κάθε μια από αυτές τις ευθείες γραμμικά με ένα πολυώνυμο πρώτου βαθμού. Βέβαια οι 1806 συντελεστές που θα προέκυπταν από 301 καμπύλες (με 6 συντελεστές η κάθε μια) είναι ένας πολύ μεγάλος αριθμός. Η προσέγγιση πρώτου βαθμού φαίνεται στην παρακάτω εικόνα 4.2.



Σχήμα 14: Κάθε τμήμα της συνάρτησης προσεγγίζεται από ένα πολυώνυμο πρώτου βαθμού αφού αποτελεί μια ευθεία.

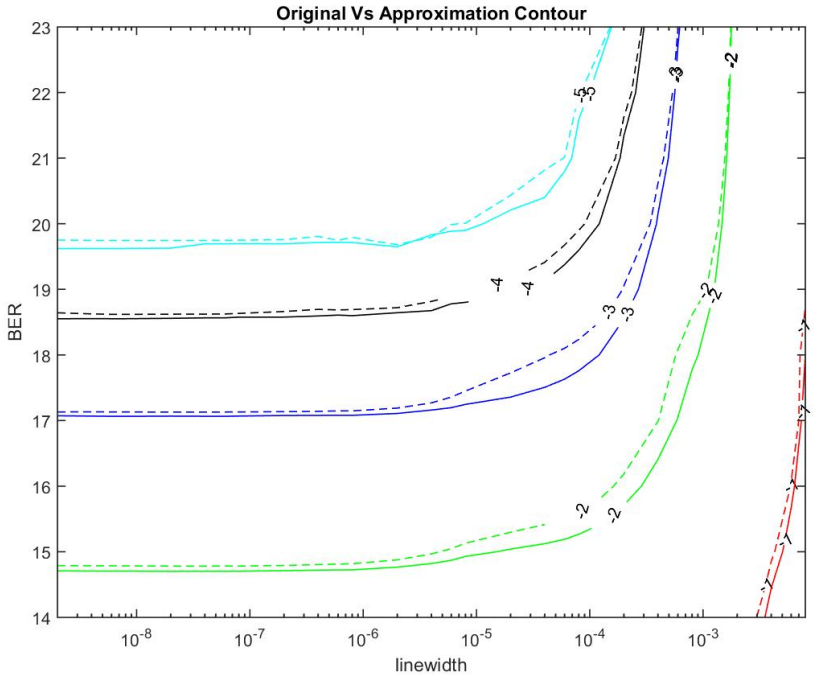
Έτσι μπορούμε να καταφύγουμε σε μια εκ νέου πολυωνυμική προσέγγιση. Αυτή τη φορά θα ομαδοποιήσουμε τον κάθε συντελεστή τμήματος της καμπύλης με τους αντίστοιχους συντελεστές των άλλων καμπυλών (κάθε τμήμα αποτελείται από δυο συντελεστές εφόσον προσεγγίστηκε ως ευθεία). Έτσι έχοντας δύο σύνολα από συντελεστές για κάθε τμήμα θα καταλήγαμε σε έξι πίνακες με συντελεστές που θα περιέγραφαν ανά δύο την αλλαγή των συντελεστών του τμήματος για διαφορετικές τιμές του σηματοθορυβικού λόγου. Αυτό το δεύτερο στάδιο προσέγγισης φαίνεται στην παρακάτω εικόνα 4.3.



Σχήμα 15: Κάθε οριζόντιος 6-στηλος πινακας αναπαριστά μια καμπύλη. Μια οικογένεια ν καμπυλών αναπαριστάται απο ενα πίνακα $n \times 6$

Η εκ νέου πολυωνυμική προσέγγιση αυτών των πινάκων μειώνει δραστικά τον αριθμό των συντελεστών, ενώ ταυτόχρονα μας επιτρέπει γνωρίζοντας μόνο το πλάτος και το SNR κάθε συμβόλου να αποκτήσουμε μια προσεγγιστική τιμή της συνάρτησης ανάθεσης βάρους. Η ποιότητα της προσέγγισης βέβαια εξαρτάται από το βαθμό των πολυονυμικών προσεγγίσεων του πρώτου και του δεύτερου σταδίου που περιγράψαμε πιο πάνω.

Τα σημεία στα οποία αποφασίσαμε να χωρίσουμε σε τμήματα κάθε καμπύλη αποδείχτηκαν μετά από προσομοιώσεις πως πρέπει να είναι τα $x_1 = 0.723, x_2 = 1.170$. Επίσης, ο βαθμός προσέγγισης της κάθε καμπύλης του δευτέρου επιπέδου επιλέχτηκε μετά από σύγκριση όλων των βαθμών από πρώτου μέχρι και δεκάτου. Παρακάτω φαίνεται η σχεδίαση των ισοδυναμιών καμπυλών φασικού θορύβου σε σχέση με το σηματοθορυβικό λόγο για τον αρχικό αλγόριθμό καθώς και για την εκδοχή με την προσέγγιση δεύτερου σταδίου με πολυώνυμο πρώτου βαθμού που τελικά είναι η επιλογή μας.



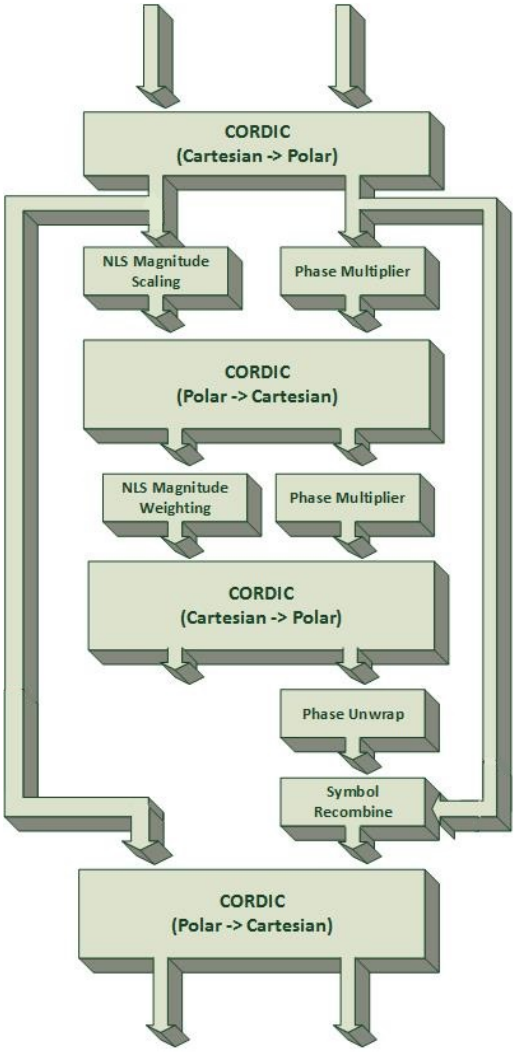
Σχήμα 16: Ισοδυναμικές καμπύλες φασικού θορύβου σε σχέση με το σηματοθορυβικό λόγο. Η καμπύλη με τη διακεκομμένη γραμμή είναι ο αλγόριθμος με την πολυωνυμική προσέγγιση ενώ η άλλη είναι η κλασσική εκδοχή του αλγορίθμου.

Από την παραπάνω γραφική παράσταση (εικόνα 4.13) είναι φανερό πώς η προσέγγισή με πολυώνυμο πρώτου βαθμού δίνει αρκετά καλα αποτελέσματα. Ο τελικός πίνακας των συντελεστών από τους οποίους ανακατασκευάζουμε τις καμπύλες έχει μέγεθος μόλις $2 \times 6 = 12$. Ένα τέτοιο μέγεθος είναι εύκολα διαχειρίσιμο σε μια υλοποίηση για FPGA καθώς και η εύρεση του βάρους του συμβόλου αποτελείται από μια απλή πράξη μερικών πολλαπλασιασμών και προσθέσεων, κάτι το οποίο μπορεί να υλοποιηθεί αρκετά αποδοτικά και γρήγορα στο υλικό μέσω της παρακάτω συνάρτησης(εξίσωση 3) .

$$(p_{11} * SNR + p_{12}) * Mag + (p_{22} + p_{22}) \tag{3}$$

όπου SNR η τιμή του σηματοθορυβικού λόγου και Mag το πλάτος του εισερχόμενου συμβόλου.

Έχοντας λοιπόν μειώσει την πολυπλοκότητα ενός κρίσιμου σημείου του αλγορίθμου NLS Estimator μπορούμε να προχωρήσουμε στην υλοποίηση αυτού σε πλατφόρμα FPGA. Η αρχιτεκτονική του συστήματος που υλοποιήσαμε παρουσιάζεται παρακάτω.

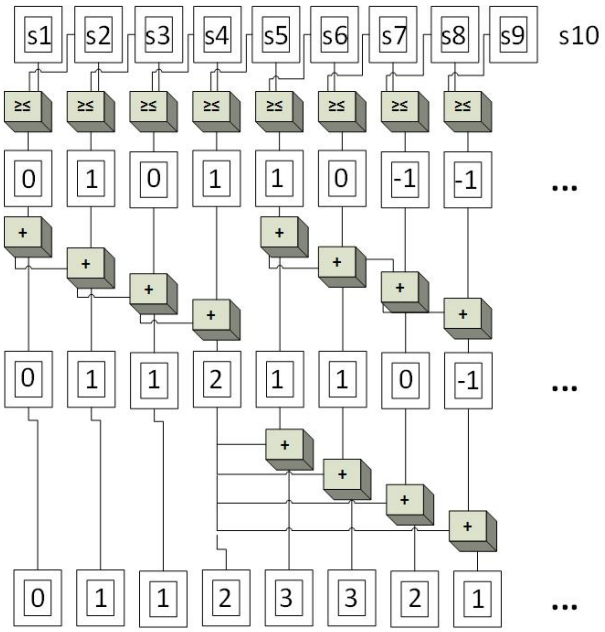


Σχήμα 17: Αποψη της αρχιτεκτονικής του συστήματος που υλοποιήσαμε στην πλατφόρμα FPGA

Όλο το σύστημα υλοποιήθηκε αμιγώς σε γλώσσα VHDL ενώ όλα τα μεταβλητά στοιχεία (όπως φίλτρα) καθώς και τα σημεία που επιλέξαμε να ελέγχουμε το πλήθος των bit που επεξεργάζονται είναι δυναμικά διαμορφούμενα και οριζόμενα από το χρήστη. Έτσι φτιάξαμε ένα χρηστικό δυναμικό εργαλείο ικανό να μας βοηθήσει να μελετήσουμε σε βάθος και να προσαρμόσουμε τη συμπεριφορά του συστήματος στις βέλτιστες συνθήκες λειτουργίας.

Αφού υλοποιήθηκε και ελέγχθηκε ο κώδικας για την ορθότητα των αποτελεσμάτων του ξεκινήσαμε την προσπάθεια να παραλληλοποιήσουμε την αρχιτεκτονική του συστήματος. Καθώς οι ύψιστες συχνότητες λειτουργίας που καλύπτει ένα FPGA δεν ξεπερνούν τα $400-500$ MHz είναι επιτακτική ανάγκη να αυξήσουμε τον ρυθμό επεξεργασίας δεδομένων του συστήματος μέσω της παραλληλοποίησής του.

Βέβαια, η παραλληλοποίηση ενός συστήματος απέχει αρκετά από τον απλή αντιγραφή και διασύνδεση των δομικών του μονάδων. Ιδιαίτερη μελέτη πρέπει να γίνει για υπομονάδες του συστήματος των οποίων τα αποτελέσματα επιβάλλουν μια σχέση μεταξύ των εισόδων τους. Στην παρούσα μελέτη τέτοιες μονάδες ήταν η ξεδίπλωση φάσης phase unwrap καθώς και το φιλτράρισμα των συμβόλων. Ενδεικτικά θα παρουσιάσουμε την αρχιτεκτονική της μονάδας phase unwrap (εικόνα 5.13).



Σχήμα 18: Η αρχιτεκτονική της παραλληλοποιημένης μονάδας phase unwrap για βαθμό παραλληλοποίησης $p = 4$.

Η μονάδα αυτή απαιτεί κάθε φάση που υπολογίζεται εσωτερικά να γνωρίζει το προηγούμενο από αυτήν αποτέλεσμα για να μπορέσει να υπολογίσει σωστά την μετατόπιση φάσης.

Αφού ελέγχθηκε η ορθή λειτουργία της παράλληλης διάταξης του συστήματος μας συνεχίσαμε με την προσπάθεια βελτιστοποίησης του με σκοπό να παράγουμε υψηλή ακρίβεια σε συνδυασμό με υψηλή ταχύτητα επεξεργασίας. Αυτό, όπως προείπαμε έγινε με τη βοήθεια σταδίων περικοπής των bit εξόδου από συγκεκριμένους καταχωρητές του συστήματος.

Αφού πειραματιστήκαμε με την παραγόμενη ακρίβεια καταλήξαμε σε ορισμένα σετ απο bit τα οποία θα εξάγει η κάθε υπομονάδα του συστήματος που μας εξασφαλίζουν λειτουργία με διαφορετικής ακρίβειας αποτελέσματα και κατ επέκταση διαφορετική επίδοση ως προς την ταχύτητα επεξεργασίας δεδομένων. Αυτά φαίνονται στον παρακάτω πίνακα.

			Bits που περικόπτονται						
Κωδικός σχεδιασμού	Είσοδος	Μέγεθος συντελεστών	CORDIC-1	NLS	CORDIC-2	Φίλτρο	CORDIC-3	Symbol Recombine	CORDIC-4
1	10	6	0	5	1	2	2	2	2
2	10	6	0	5	1	3	2	2	2
3	11	12	1	7	1	3	2	2	4
4	10	6	0	5	2	2	2	2	1
5	11	6	1	5	1	2	2	2	2

Πίνακας 1: Επιλεγμένα σετ από bit που θα περικοπούν για να προσδώσουν στο σύστημα έμφαση στην ακρίβεια των αποτελεσμάτων ή στο ρυθμό επεξεργασίας δεδομένων.

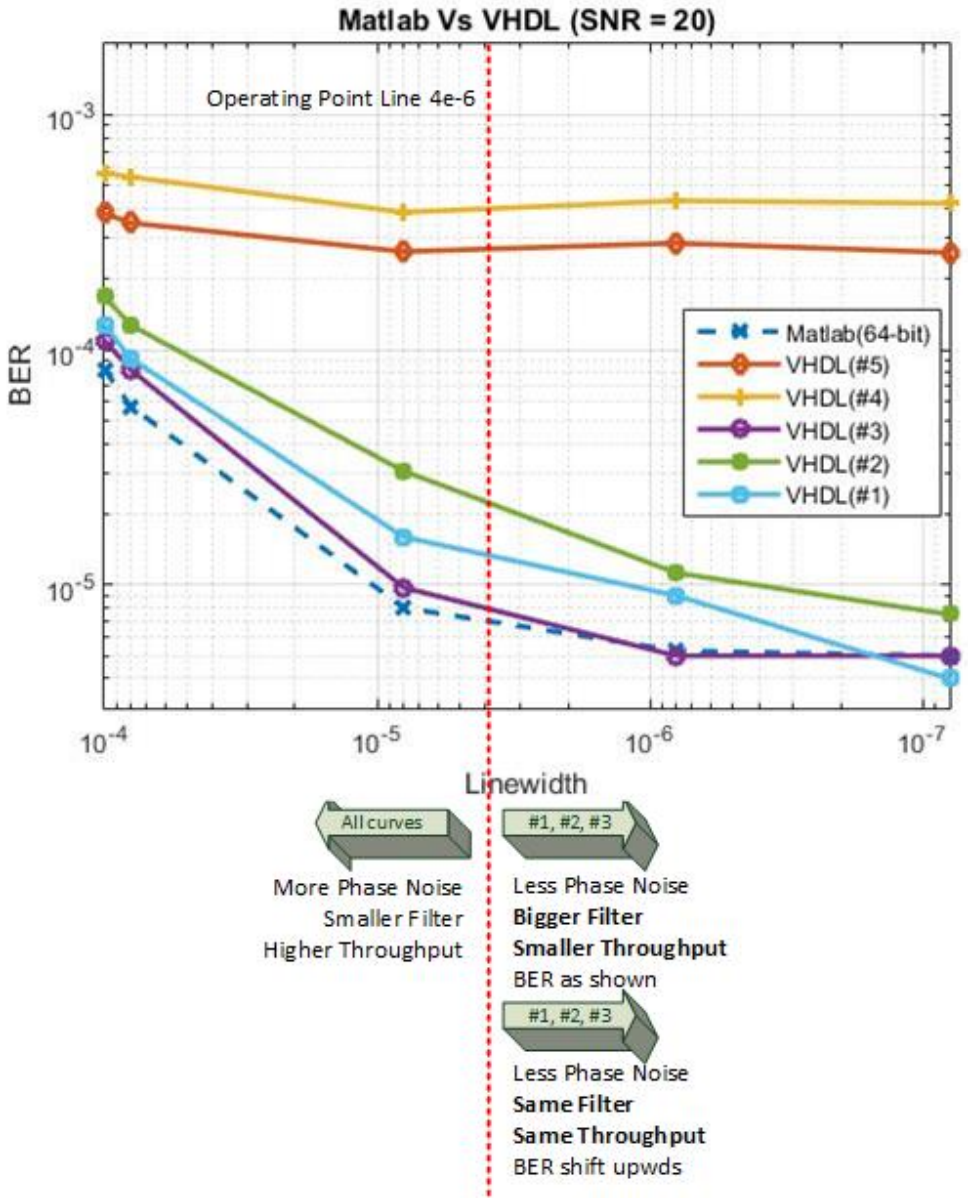
Στις επόμενες σελίδες λοιπόν θα εκτεθεί σταδιακά και αναλυτικώς η αξιολόγηση του συστήματος ως προς την ακρίβεια των αποτελεσμάτων του σε σχέση με το θεωρητικό μοντέλο καθώς και η μελέτη της απόδοσης που μπορεί να μας παρέχει σε σχέση με τις απαιτήσεις των σύγχρονών τηλεπικοινωνιακών συστημάτων.

Αρχικά, αξίζει να τονίσουμε πως όλες οι προσομοιώσεις λειτουργίας του συστήματος πραγματοποιήθηκαν με την οικογένεια FPGA Virtex-7 της εταιρείας Xilinx. Συνεπώς, κάθε χαρακτηριστικό

του συστήματος μπορεί να αλλάξει με καλύτερα ή χειρότερα αποτελέσματα ανάλογα πάντα με την χρησιμοποιούμενη πλατφόρμα.

Αφού μελετήθηκε διεξοδικά η συμπεριφορά του αλγορίθμου σε σχέση με τον αριθμό των bits ακρίβειας που αυτός χρειάζεται για να αναπαραστήσει οποιαδήποτε ποσότητα καθώς και για να ελαχιστοποιήσει τους χρησιμοποιούμενους πόρους, έπειτα προχωρήσαμε σε σύγκριση του υλοποιημένου συστήματος με το θεωρητικό μοντέλο της Matlab.

Παρακάτω μπορούμε να διαπιστώσουμε πως η μελέτη μας παρήγαγε ένα σύνολο λύσεων το οποίο μπορεί να ωθήσει το σύστημα σε διαφορετικά σημεία λειτουργίας παρέχοντας μας έτσι αρκετή ευελιξία. Συγκεκριμένα οι παρακάτω καμπύλες 6.4 περιγράφουν την επίδοση του συστήματος σε σχέση με το θεωρητικό μοντέλο. Μπορούμε να δούμε πως τα διαφορετικά σετ διαμορφώσεων παράγουν διαφορετικά αποτελέσματα.



Σχήμα 19: Οι καμπύλες του υλοποιημένου σε FPGA σύστημα μαζί με το θεωρητικό μοντέλο του Matlab.

Παραπάνω, μπορούμε να δούμε πως έχει εκλεχθεί ένα σημείο λειτουργίας για το σύστημα. Σε αυτό το σημείο λειτουργίας πάρθηκαν οι μετρήσεις που αφορούν τη συχνότητα λειτουργίας και το ρυθμό επεξεργασίας δεδομένων από το σύστημά μας. Η μετακίνηση του σημείου λειτουργίας μπορεί να συνεπάγεται καλυτέρευση ή χειροτέρευση των χαρακτηριστικών του συστήματος σε συνάρτηση πάντα με τις μετρικές τις οποίες λαμβάνουμε περισσότερο υπ όψιν μας (ακρίβεια αποτελεσμάτων, ταχύτητα και συχνότητα λειτουργίας).

Τα αποτελέσματα που αφορούν τη συχνότητα λειτουργίας και το ρυθμό διακίνησης δεδομένων παρουσιάζεται στον παρακάτω πίνακα 6.4.

	Επίδοσεις			Παρατήρηση
Κωδικός σχεδιασμού	Μέγιστη συχνότητα μονού pipeline	Μεγιστη εξωτερική παραλληλία	Μεγιστο Baud rate	Φιλτρο
1	300.3 ΜΗζ	105	31.5 ΓΒδ	32 ταπς
2	303 ΜΗζ	119	36.1 ΓΒδ	32 ταπς
3	300.3 ΜΗζ	84	25.2 ΓΒδ	32 ταπς
4	333.3 ΜΗζ	140	46.6 ΓΒδ	16 ταπς
5	333.3 ΜΗζ	139	46.3 ΓΒδ	16 ταπς

Πίνακας 2: Διαφορετικές υλοποιήσεις του συστήματος και οι αντίστοιχες συχνότητες λειτουργίας καθώς και οι μέγιστοι ρυθμοί επεξεργασίας δεδομένων.

Είναι εμφανές πως το σύστημα μπορεί να λειτουργήσει σε αρκετά υψηλούς ρυθμούς μετάδοσης. Μπορεί επίσης να χαμηλώσει την συχνότητα και την εξωτερική του παραλληλία έτσι ώστε να καταφέρει να παράγει πιο ακριβή αποτελέσματα ανάλογα με τις ανάγκες του τηλεπικοινωνιακού συστήματος.

Εκτός απο την εγγύτητα των καμπυλών στο θεωρητικό μοντέλο και τη συχνότητα λειτουργίας, είναι επιτακτικό να εξετάσουμε και την ποινή (penalty) του SNR. Αυτή η ποσότητα δείχνει τι ποσο ενέργειας θα πρέπει να δαπανήσουμε έτσι ώστε με το υλοποιημένο σύστημα να παράγουμε τα αποτελέσματα του θεωρητικού μοντέλου. Οι τιμές αυτές φαίνονται στον παρακάτω πίνακα.

	Επίδοση	
Κωδικός σετ	100 kHz	2 MHz
1	0.3 dB	0.2 dB
3	0.5 dB	0.3 dB
4	2 dB	1.6 dB
5	1.7 dB	1.3 dB

Πίνακας 3: Η τιμή του SNR που απαιτείται για να επιτύχουμε αποτέλεσμα BER ίσο με 1×10^{-3} για διαφορετικές τιμές φασικού θορύβου. Η διαφορά υπολογίζεται βάση του θεωρητικού μοντέλου.

Συνοψίζοντας, μπορούμε να διαπιστώσουμε πως σε όλες τις περιπτώσεις μπορούμε να διαλέξουμε κάποιο αρχιτεκτονικό σχεδιασμό λειτουργίας που να μας δίνει τα επιθυμητά αποτελέσματα. οι διαφορές σε SNR της τάξης του 0.2 είναι μηδαμινές και προσφέρουν μεγάλη αξιοπιστία στο σύστημα. Απο την άλλη, οι μεγαλύτερες τιμές σημαίνουν την μη ακριβέστατη λειτουργία, αλλά οι συγκεκριμένοι σχεδιασμοί προσφέρουν ταχύτατες επιδόσεις στο πεδίο της συχνότητας και του ρυθμού επεξεργασίας δεδομένων.

Στην παρούσα εργασία υλοποιήσαμε ένα υψηλων επιδόσεων αλγόριθμο σε αρχιτεκτονική FPGA που έχει τη δυνατότητα ενσωμάτωσης σε σύγχρονούς τηλεπικοινωνιακούς δέκτες. Ο σχεδιασμός έγινε με γνώμονα την ευελιξία και την προσαρμοστικότητα και αυτό φάνηκε στις ποικίλες λύσεις

που δόθηκαν έτσι ώστε να ικανοποιούν πληθώρα τηλεπικοινωνιακών αναγκών. Ο αλγόριθμος NLS Estimator αποτελεί, λοιπόν μια βέλτιστη λύση για ταχύτατη και ακριβής ανάκτηση φάσης φέροντος σε ένα σύγχρονο τηλεπικοινωνιακό συστημα.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the recent years the use of Internet services and telecommunication resources (cloud services, mobile internet, online gaming) presents a tremendous increase. This massive exchange of information demands the extensive use of fiber-optic infrastructures to support the needs of today's and tomorrow's telecommunication networks.

The modern optic networks offer not only higher data transfer speeds (100 - 400 Gbit/s) but also more efficient utilization of the existing bandwidth in comparison with the conventional telecommunication links (wired, wireless). What more, the adaptation of state of the art technologies in the optic networks (coherent optical communication, flexible optical networks) paves the way for more flexible networks, able to manage the dynamic and ever increasing internet traffic.

As stated above the coherent detection technology has been leading the telecommunication field the recent years. A coherent optical transmission system is characterized by its capability to do coherent detection, which means that an optical receiver can track the phase of an optical transmitter (and hence phase coherence) so as to extract any phase and frequency information carried by a transmitted signal [4]. This way, by changing one or more properties (i.e. amplitude, frequency, phase) of each wave we can "pack" more information together. Modulation techniques define which of the properties are being manipulated. Higher order modulation schemes allow more information to fit into a single radio wave. In other words, higher order modulation equals more bits per wave. This is a powerful way of improving spectral efficiency [5].

Nevertheless, the disadvantage of higher order modulation is that the data becomes more susceptible to noise and interferers since the receiver must accurately detect more discrete phases and amplitudes of a signal. In order to achieve compensation for waveform distortion as well as improve both receiving sensitivity and frequency utilization coherent detection technology employs digital signal processing techniques. Considering this need the use of high performance computing is considered a key enabler in high order digital coherent modulation systems.

The established platform in the world of digital signal processing are FPGAs (Field Programmable Gate Arrays). In terms of their size and processing speeds, modern FPGAs have attained a level that makes it possible not only to perform individual mathematical operations but also to accommodate entire DSP algorithms. At the same time, leading manufacturers have released tools that specifically support the development of digital algorithms for FPGAs.

Because of their size and the components they contain, FPGAs now offer a wide variety of interesting possibilities in the field of digital signal processing. The difference between the classical solution - using a Digital Signal Processor (DSP) - and implementation on an FPGA lies in the fact that the DSP has to be programmed in Assembler or C whereas FPGA algorithms are described

in VHDL. While a DSP works through its program more or less sequentially, an FPGA maps the entire algorithm at the hardware level [6].

To sum up, FPGAs seem the ideal candidate to push the modern sophisticated high speed and increased bandwidth networks to their limits, providing a "fertile soil" for the steady advance of the telecommunication field.

1.1 Motivation and thesis objectives

The telecommunication evolution from TDMA, to WDMA and lastly to digital coherent detection has without any doubt given us the opportunity to scale up the transmitting data rates resulting in faster, more cost effective and quality communication experience.

As the properties of a single transmitted waveform (amplitude, phase, frequency) carry useful information (coherent detection) the receiver has to decompose the received wave and make sense of the transmitted data. This effort though is far more complex than detecting the amplitude of the wave and producing an 1 or a zero accordingly (direct detection). In fact, the more information is packed in one signal the less noisy it has to be and the more the signal has to be digitally processed so as to correctly decode it.

This notion shows a clear direction of embedding dsp processing in the receiver side of a network. A specific problem the receiver has to tackle is the recovery of the phase information of a signal. That recovery, which will be explained later in more detail, suffers from a specific kind of noise called phase noise which is introduced by mismatching local oscillators (lasers) in the transmitter and receiver sides. A lot of effort has been placed lately in clever approaches to eliminate the phase noise embedded in a signal. Nevertheless since coherent technologies shape and probably will shape the networks of today and tomorrow there is a lot of space for experimentation and progress in this field.

To sum up, in the particular thesis we will study the problem as well as implement in hardware fabric a specific Carrier Phase Recovery algorithm.

More specific this thesis is aiming at :

- Studying and understanding the various algorithms used in Carrier Phase Recovery.
- Evaluating their performance taking into consideration their potential complexity in a hardware implementation.
- Comparing the above algorithms with a low complexity but high performing CPR algorithm (NLS Estimator) on a theoretical level.
- Slightly altering some of the algorithm's characteristics aiming at an efficient hardware implementation.
- Implementing in an FPGA platform (Virtex - 7) the NLS estimator CPR algorithm.
- Fine-tuning some crucial algorithm's parameters (input/output bit widths) in order to achieve better results.
- Parallelization of the system's structure to exploit the full capacity of the FPGA device.
- Evaluating the algorithm's results in the hardware implementation taking into consideration the system performance and precision.

- Comparing the experimental results with other similar endeavors and conclude to some observations.

1.2 Outline of the thesis

In chapter 2 we will provide some theoretical background mandatory to understand the progress of this thesis. Specifically, key aspects of optical communications (coherent detection, laser phase noise, principles of digital modulation etc) as well as basic operating principles of FPGA architecture will be explained in detail.

In chapter 3 we will introduce and analyze two baseline Carrier Recovery algorithms along with our proposed NLS Estimator algorithm. Their theoretical performance, based on Matlab simulations, will be assessed and their basic parameters (filtering type, filtering length etc) will be calibrated for optimal performance.

Chapter 4 will be dedicated in a key insight we applied in the NLS estimator algorithm that dramatically limits its complexity while retaining its good performance.

Following, in chapter 5 we will present the whole process of implementing the algorithm on the FPGA platform. The whole pipeline will be presented in detail both in its single and parallel form. At the end of the chapter we will compare the FPGA experimental results along with the theoretical ones (derived from MATLAB) and present trade-offs between performance (operating frequency, maximum throughput, consumed power) and algorithm precision (maximum BER achieved).

Finally, in chapter 5 we present the conclusions made through the course of this thesis, as well as some suggestions for future work.

Chapter 6 will be dedicated in the bibliography used for the realization of this thesis.

Chapter 2

Theoretical background

2.1 Coherent Optical Communications

2.1.1 Basic Operating Principle

At its most basic, coherent optical transmission is a technique that uses modulation of the amplitude and phase of the light, as well as transmission across two polarizations, to enable the transport of considerably more information through a fiber optic cable.

To clarify the operating principle of the coherent optical communication concept, consider two traveling electromagnetic waves, with carrier frequencies f_s and f_{lo} respectively from two independent laser sources, labeled the received signal and the local oscillator signal, respectively. The waves propagate in the same direction with identical states of polarization (SOP). Therefore, the electric fields of the two waves can be treated as scalars and they are denoted by $\overrightarrow{E}_s(t)$ and $\overrightarrow{E}_{lo}(t)$ respectively.

For simplicity, it is assumed that $\overrightarrow{E}_s(t)$ and $\overrightarrow{E}_{lo}(t)$ are both unmodulated (CW) sinusoidal signals

$$\begin{aligned} E_s(t) &= 2\sqrt{2P_s}\cos\omega_s t + \phi_s \\ E_{lo}(t) &= 2\sqrt{2P_{lo}}\cos\omega_{lo} t + \phi_{lo} \end{aligned} \tag{2.1}$$

where P_s, P_{lo} are the average optical powers, ω_s and ω_{lo} are the angular carrier frequencies and ϕ_s and ϕ_{lo} are the initial phases of the received signal and the local oscillator signal, respectively. In the previous formula, intensity and phase noises of the lasers are neglected.

The electric field of the combined signal impinging upon the photodiode, at a single detection point, can be written as the superposition of the electric fields of the received signal and the local oscillator

$$E_r(t) = E_s(t) + E_{lo}(t) \tag{2.2}$$

The photodiode is modeled as a square-law detector which responds to the square of the electric field

$$i(t) = R\langle E_s(t)^2 \rangle \tag{2.3}$$

where R is the responsivity of the photodiode and the angle brackets denote time averaging over an interval proportional to the response time of the photodiode.

By substituting Eqs. 2.1, 2.2 into Eq. 2.3 and using trigonometric identities, we obtain the following expression for the photocurrent in the absence of noise

$$i(t) = R[P_s + P_{lo}] + 2R\sqrt{P_s P_{lo}} \cos \omega_{IF} t + \phi_{IF} t \tag{2.4}$$

where

$$\begin{aligned} \omega_{IF} &= 2\pi(f_s - f_{lo}) \\ \phi_{IF} &= \phi_s - \phi_{lo} \end{aligned} \tag{2.5}$$

In the Eq 2.4 above the first term represents the direct detection term whereas the second term represents the coherent detection term.

It is observed that Eq. 2.4 is the sum of three terms due to the direct-detection of the received signal and the local oscillator signal, and their mixing (coherent detection term), respectively. The latter preserves the information transferred by the amplitude, the frequency and the phase of the received signal. Therefore, this type of detection can be used in conjunction with amplitude, frequency or phase modulation formats. In addition, the amplitude of the coherent detection term depends on the power of the local oscillator, which can be made very large. This is the reason for the improved receiver sensitivity exhibited by coherent detection.[7]

2.1.2 Homodyne Receiver

Homodyne detection refers to the case that $\omega_{IF} = 0$. The photodiode current from the homodyne receiver becomes

$$I(t) = 2R\sqrt{P_s P_{lo}} \cos \phi_s - \phi_{lo} \tag{2.6}$$

So it is possible to estimate the phase noise ϕ_{lo} and restore the signal complex amplitude through digital signal processing on the homodyne-detected signal given by Eq. 2.6. This is the basic idea of the "digital coherent receiver".[8]

2.2 Digital Signal Processing on Coherent Receivers

Coherent detection and DSP were the key enabling technologies in the development of 100G optical transmission systems. 400G systems will continue this trend with DSP playing even more ubiquitous role at both transmitter and receiver.

As far as the receiver is concerned, the major advantage of receiver-side DSP stems from the ability to arbitrarily manipulate the electrical field after ADC enabling sampled signals in the digital domain. As shown in Figure 2.1, the fundamental DSP functionality in a digital coherent receiver can be illustrated by the following flow of steps from structural level and algorithmic level of details. Firstly, the four digitized signals (i.e. in-phase (I) and quadrature (Q) components for X and Y polarization) after an ADC are passed through the block for the compensation of front-end imperfection equalization. The imperfections may include timing skew between the four channels due to the difference in both optical and electrical path lengths within the coherent receiver. Then, the clock recovery for symbol synchronization can be processed to track the timing information of incoming samples.

Then, the frequency offset between the source laser and the LO laser is estimated and removed to prevent the constellation rotation at the intradyne frequency. Finally, the carrier phase noise is

estimated and removed from the modulated signal, which is then followed by symbol estimation and hard or soft-decision FEC for channel decoding.

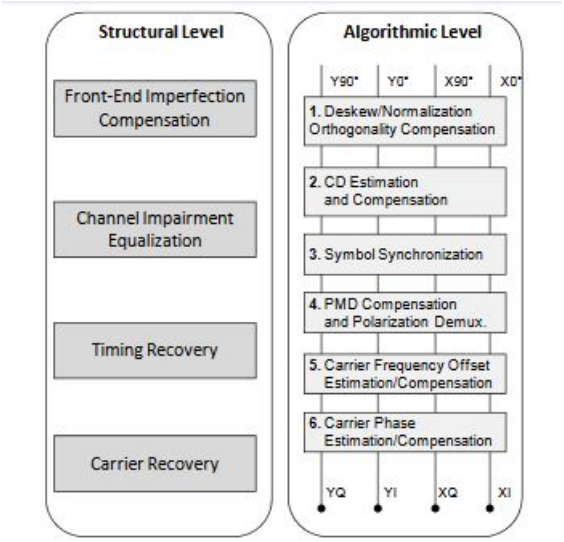


Figure 2.1: The basic building block algorithms of a integrate coherent DPS receiver

2.3 Principles of Digital Modulation

Here, we briefly present the operating principles of the basic digital modulations.

2.3.1 Amplitude Shift Keying

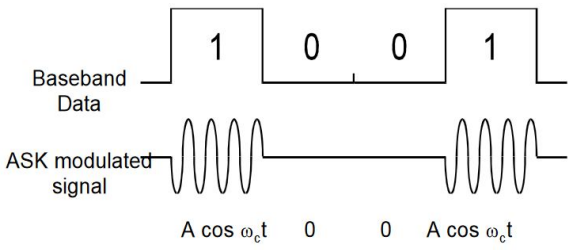


Figure 2.2: Operating principle of the amplitude shift keying (ASK)

- ASK demonstrates poor performance, as it is heavily affected by noise and interference

2.3.2 Frequency Shift Keying

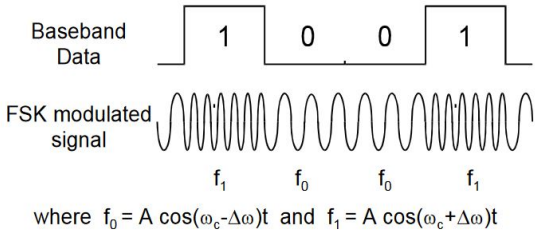


Figure 2.3: Operating principle of the frequency shift keying modulation (FSK)

- Bandwidth occupancy of FSK is dependant on the spacing of the two symbols. A frequency spacing of 0.5 times the symbol period is typically used.

- FSK can be expanded to a M-ary scheme, employing multiple frequencies as different states.

2.3.3 Phase Shift Keying

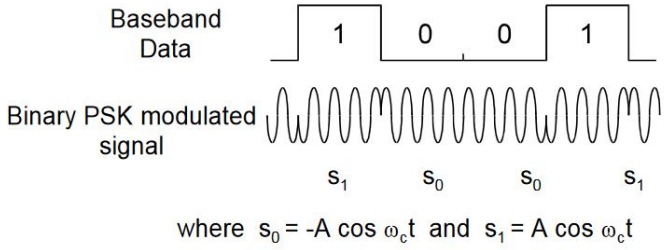


Figure 2.4: Operating principle of the phase shift keying modulation (PSK)

- Binary Phase Shift Keying (BPSK) demonstrates better performance than ASK and FSK.

- PSK can be expanded to a M-ary scheme, employing multiple phases and amplitudes as different states.

2.3.4 Quadrature Phase Shift Keying

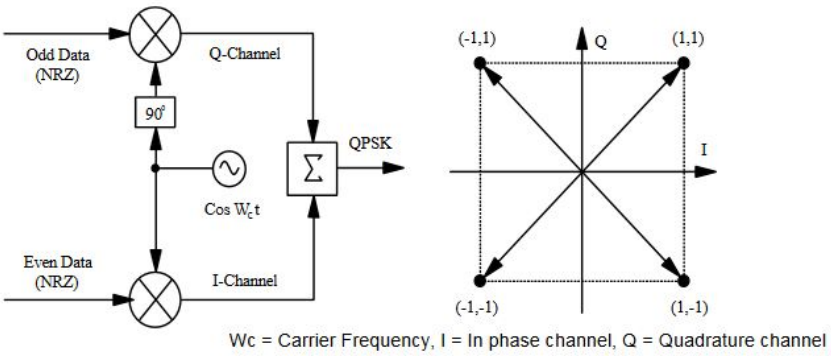


Figure 2.5: Operating principle of the Quadrature phase shift keying modulation (QPSK)

- Quadrature Phase Shift Keying is effectively two independent BPSK systems (I and Q), and therefore exhibits the same performance but twice the bandwidth efficiency.

Here Quadrature means that the signal shifts among phase states that are separated between 90 degrees. Each of the channels above modulates a single-carrier.

2.3.5 M - Quadrature Amplitude Modulation

It is simply a combination between amplitude modulation (ASK) and phase shift keying (PSK). By defining different levels of amplitude as well as different levels of phase and mixing them together we get a higher order constellation diagram thus representing a higher order modulation format. In the current thesis we will study the 16-QAM modulation format. Such a modulation gets produced by having a modulation alhabet of 4 bits thus resulting in 16 symbols, each of whom encodes 4 bits.

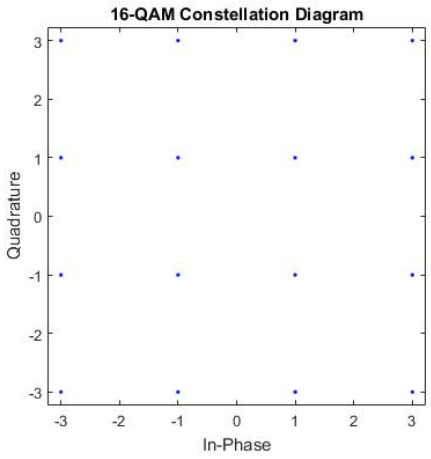


Figure 2.6: Example constellation of 16 QAM modulated signals. The groups of four bits are presented on top of every transmitted symbol.

- Carries higher data rates than ordinary amplitude modulated schemes and phase modulated schemes.

- However the points are closer together and they are therefore more susceptible to noise and data errors.

To provide an example of how QAM operates, the constellation diagram above (fig 2.6) shows the values associated with the different states for a 16QAM signal. From this it can be seen that a continuous bit stream may be grouped into fours and represented as a sequence.

2.4 Gray Encoding

The idea behind the Gray coding is that you are minimizing bit errors by causing adjacent words to only be one bit off from their neighbors. That way, a bit of noise that's only enough to push the received signal over by one spot in the constellation only causes a single bit error.The idea behind the Gray coding is that you are minimizing bit errors by causing adjacent words to only be one bit off from their neighbors. That way, a bit of noise that's only enough to push the received signal over by one spot in the constellation only causes a single bit error. In the following figure we can see a Gray encoded 16-QAM constellation where every symbol differ from its neighbor by just one bit.

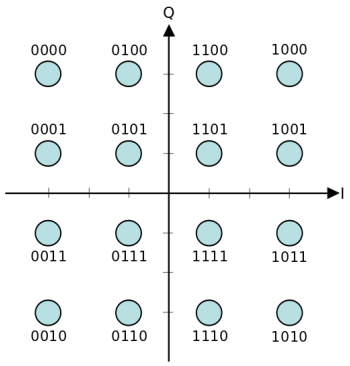


Figure 2.7: An example of a gray encoded 16 QAM constellation. Neighbors differ by just one bit.

2.5 Laser Induced Phase Noise

Laser phase noise is caused by spontaneous emission, and is modeled as a Wiener process:

$$\phi(t) = \int_{-\infty}^{t} \delta\omega(\tau)dt \tag{2.7}$$

where $\phi(t)$ is the instantaneous phase, $\delta\omega(\tau)$ is frequency noise with zero mean and auto correlation $R_{\delta\omega\delta\omega}(\tau) = 2\pi\Delta v\delta(\tau)$. It can be shown that the laser output $E_0(t) = Ae^{j\omega_c(t)+\phi(t)}$ has a Lorentzian spectrum with a 3-dB linewidth Δv.

The term "linewidth" for lasers actually describes how stable the semiconductor laser's phase is.

It has been shown that laser linewidth is inversely proportional to output power, so it is desirable to operate the TX and LO lasers at maximum power, attenuating their outputs as required.

Phase noise is an important impairment in coherent systems as it impacts carrier synchronization. In noncoherent detection, the carrier phase is unimportant because the receiver only measures energy. In DPSK, information is encoded by phase changes, and only needs to be small enough

such that the phase fluctuation over a symbol period is small. We know that the baseband signal is modulated by $e^{j\phi(t)}$. In the absence of other impairments, this manifests as a rotation of the received constellation.

This rotation has an angular speed of $2\pi\Delta n$ for every T seconds (where $1/T$ denotes the symbol rate). Thus, the strength of the phase noise will depend on the product ΔnT and a larger value of ΔnT means a faster changing carrier phase.[9]

Carrier synchronization is required to ensure $\phi(t)$ is small so the transmitted symbols can be detected with low power penalty. Since phase noise is a Wiener process with temporal correlation, it can be mitigated by signal processing.[10]

The result of a phase noise contaminated signal is visible in the figure below.

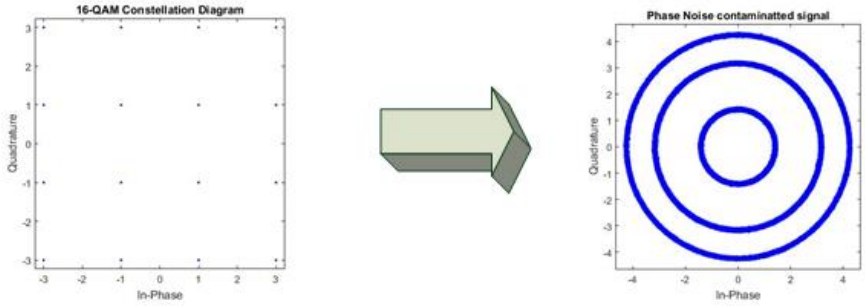


Figure 2.8: A 16 QAM constellation before and after being altered by phase noise.

2.6 Additive White Gaussian Noise

A coherent optical system is corrupted by Additive White Gaussian Noise (AWGN), which includes amplified spontaneous emission (ASE) from inline amplifiers, receiver LO shot noise and receiver thermal noise. In the canonical transmission model, we model the cumulative effect of these noises by an equivalent noise source referred to the input of the receiver [10].Therefore, by averaging the carrier phase over many symbol intervals, it is possible to obtain an accurate phase estimate. Fig 2.9 shows the effect of filtering the signal to supress the AWGN noise.

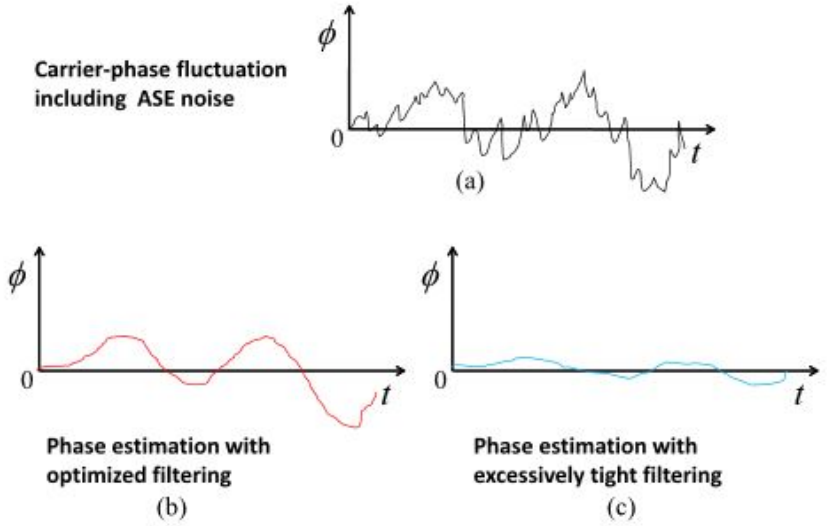


Figure 2.9: Effect of filtering the signal for AWGN supressing. In (a), the received signal is contaminated by AWGN noise. In (b), optimum filtering of the signal improves the SNR (signal to noise ratio), while it accurately tracks the phase fluctuation. On the other hand, in (c), excessively tight signal filtering proves to be unable to track the changes of the carrier phase.

The noise impact on a constellation of 16-QAM is also visible on the figure below.

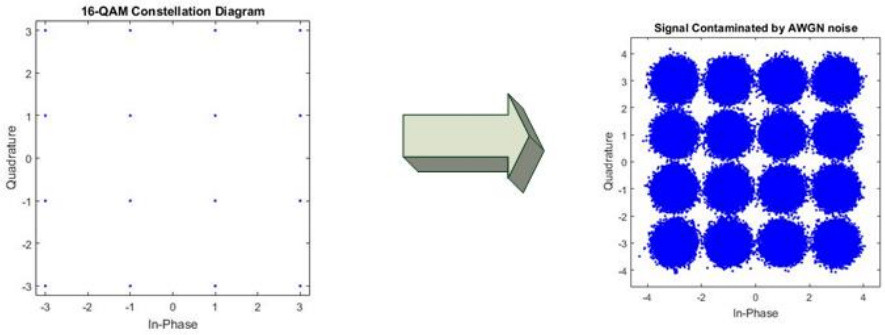


Figure 2.10: A 16 QAM constellation before and after being altered by AWGN noise.

2.7 Digital Electronic Systems

In the world of digital electronic systems there are three basic kind of devices. Memory, microprocessor and logic devices.

- Memory devices store random information such as the content of a database.
- Microprocessors execute software instructions to perform a variety of tasks such as processing a word document.
- Logic devices provide specific function such as data communication , signal processing, timing and control operations and almost every other function a system must perform.

The logic devices are further subdivided into two main categories. Those are

- Fixed logic devices in which as the name suggests the circuits embedded are permanent and once manufactured they cannot be changed.

- Programmable ones. These devices can be changed by the user at any time thus ending in an altered circuit which can perform various tasks.

In the recent years there has been many engineers developing applications on programmable devices and especially on a specific type of device called FPGA (Field programmable Gate Array).

2.7.1 Field Programmable Gate Arrays (FPGA) Architecture basics

FPGA stands for Field Programmable Gate Array. An FPGA is a component that can be thought of as a giant ocean of digital components (gates, look-up-tables, flip-flops) that can be connected together by wires. The code that you write makes real physical connections with wires to perform the function that you need. What makes FPGAs special is that they are very good at performing a large number of operations in parallel (at the same time). They are used in high-speed, high-performance tasks such as image processing, telecommunications, digital signal processing, high-frequency stock market trading, and many others. The basic building blocks of an FPGA are the following:

- Configurable Logic Blocks (CLBs). A CLB consists of registers (memory), MUXs and Combinatorial Functional Units.

- Configurable Input-Output blocks (IOBs). IOBs are an arrangement of transistors for configurable IO drivers and surround the above explained CLBs.

- A metal network for interconnecting the CLBs (programmable interconnect). These are unprogrammed interconnection resources on the chip which have channel routing with fuse links.

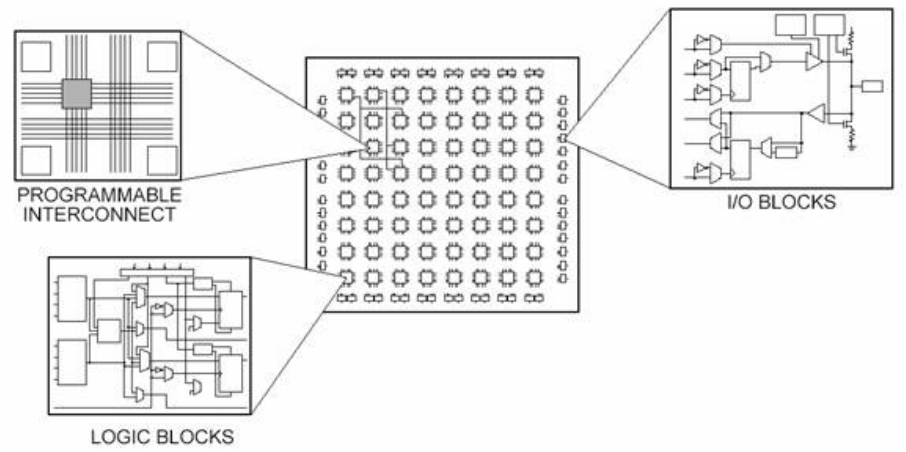


Figure 2.11: A higher level abstraction of an FPGA device with visible CLBs , IOBs and programmable interconnect as described above.

2.7.2 FPGA advantages and disadvantages

The use of such devices can have a great impact on the development of a product or the implementation of an ides. Specifically :

- Design cycle is significantly reduced. An engineer can programm the actual chip in a matter of seconds or minutes rather than weeks or months required by mask designed parts (ASICs)
- Since no custom mask tooling is required it saves a lot of cost.
- They are low risk and highly flexible devices.
- Suitable for prototyping.
- FPGAs are inherently parallel and have very efficient hardware.

On the other hand there are also some drawbacks in the use of FPGAs.

- Operating speed is comparatively less than CPUs, GPUs or ASICs.
- The circuit delay depends on the performance of the design implementation tools.
- The design and debugging time is far more time consuming comparing to CPU or microcontroller programming paradigms.

2.7.3 FPGA Applications

FPGA design can be applied to almost every aspect of embedded device design and some of its main application areas are:

- Low cost customizable digital circuitry
- High performance computing.
- Evolvable hardware. Evolvable hardware is when hardware can change its own circuitry.
- Digital Signal Processing (DSP).

2.7.4 Merits of FPGA use in DSP Applications

There is an increasing tendency to choose modern FPGA platforms over traditional DSP processors in high-speed telecommunication networks. In many cases, today's systems are so complex that single-DSP implementations have insufficient processing power. At the same time, system architects simply can't afford the costs, complexities and power requirements of multiple-chip systems.

FPGAs have now emerged as a great choice for systems requiring high-performance DSP functionality. In fact, FPGA technology can often provide a much simpler solution to difficult DSP challenges than a standalone digital signal processor.

The main advantage to digital signal processing within an FPGA is the ability to tailor the implementation to match system requirements. This means in a multiple-channel or high-speed system, you can take advantage of the parallelism within the device to maximize performance, while in a lower-rate system the implementation may have a more serial nature. Thus the designer can tailor the implementation to suit the algorithm and system requirements rather than compromising the desired ideal design to conform to the limitations of a purely sequential device. Very high-speed

I/O further reduces cost and bottlenecks by maximizing data flow from capture right through the processing chain to final output.

On the contrary, DSPs are limited in performance by clock rate and the sequential nature of their internal design.

To sum up, it seems that the use of FPGA in modern telecommunication applications is mandatory. In the heart of these high-performance and noise-intolerant applications lie the digital coherent receivers used in fiber-optic networks for high order modulated signals. (CD and PMD compensation, Symbol synchronization, Carrier Recovery etc).

2.7.5 Related work on Carrier Phase Recovery and FPGAs

During the recent years there has been a great deal of research regarding embedding FPGA in the future fiber-optic telecommunication systems. Especially, concerning the field of digital coherent communications and more specifically the Carrier Phase Recovery problem, we will briefly present some of the most notable and recent work that has been carried out.

In many works we have encountered the implementation of digital phase locked loops (DPLL). While faster than their alternatives (all analog PLL) the phase locked loops are not preferred for use in burst transmission systems due to their high internal transition delay. Contrary to the phase locked loops, pure feed-forward structures provide phase estimates without any transition delay. In addition coherent receivers with PLL have more stringent linewidth requirements making their exploitations in modern high-speed high-linewidth systems impossible. With the ever increasing computational power of FPGA devices a feed-forward CPR system can be successfully implemented yielding not only high-performance but also high-precision results. We will briefly mention some of the most notable and similar researches that have been carried out by other teams.

In [11] a phase locked loop for 16 QAM modulation has been implemented in an FPGA platform. The maximum operating frequency achieved was 150 Mhz which translates to a throughput of 15 Msamples/s.

Another endeavor of implementing a PLL on FPGA can be seen in [12]. The team implemented the PLL on a Cyclone IV fpga platform (from Altera). They achieved a 101.3 Mhz maximum operating frequency while occupying the following resources at the FPGA fabric.

- 2 % (1867) LEs
- 2 % (1859) Combination Functions
- < 1 % (936) Logic Registers
- 2 % (12) Embedded multipliers

Additional work in the field of QPSK modulated signals can be seen at [13]. The team implemented the Viterbi-Viterbi 4th power algorithm and achieved an operating frequency of 2.8 Gbps while having a BER floor of 1.210^{-7}. The signals transmitted were in a dual polarization multiplexing mode.

A particularly interesting work has been carried out in [14]. They have implemented a low complexity alternative of the classical the Blind Phase Search algorithm resulting in a hardware performance of 38.4 GBd for QPSK modulated signals. The architecture could process 128 symbols in parallel with a maximum operating frequency of 300 Mhz. This implementation was realized in a Virtex - 7 device using the xc7vx690t part. The hardware requirements of a full system for QPSK signals were claimed to be

- 11.58% (50181) Slice LUTs
- 6.83% (59197) Slice Registers
- 0% DSPs

The SNR penalty for a QPSK constellation is claimed to be below 0.5 dB for the required SNR at a BER of 10^{-3} and laser linewidths below 2MHz in 28GBd systems. For 16 QAM the penalty was shown to be below 2.3dB.

Chapter 3

Carrier Phase Recovery Algorithms Presentation

Phase noise can prove to be destructive for a telecommunication system. It makes the intact transmission of information from the transmitter to the receiver impossible. Much debate and scientific research is going on in the reverse of this phenomenon. In this chapter we present the reader with three phase error correction algorithms. The thorough description of each algorithm's operating principle is combined along with the improvement of its performance. The latter is accomplished through experimentation with its parameters.

3.1 Viterbi - Viterbi 4th power estimator (VV4PE)

3.1.1 Algorithm's description

This algorithm is essentially an indent of the classical Viterbi - Viterbi algorithm [1]. A simple schematic design of the algorithm is presented below (figure 3.1). In the current study the applied modulation is QAM and more specifically QPSK (4QAM) and 16QAM.

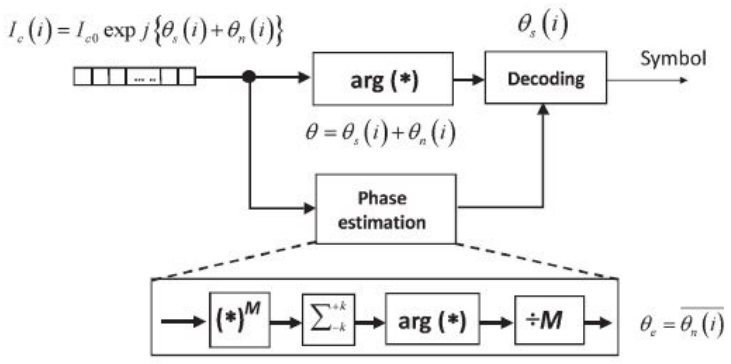


Figure 3.1: Algorithm's schematic diagram VV

For a better understanding we present the algorithm's operating principle.(figure 3.2) .

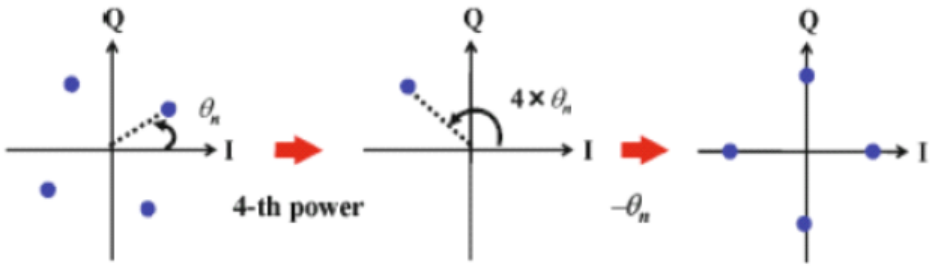


Figure 3.2: VV operating principle. By exponentiating the signal at the fourth power we manage to strip the underlying modulation and measure the added phase noise.

As we note from the figure above 3.2 the algorithm has an inherent $\frac{\pi}{2}$ symmetry. This fact justifies its use in 4QAM and 16QAM constellations even if its performance in 16QAM will be later proved to be inadequate.

Let the signal output from the transmitter be

$$r_k = a_k e^{j\theta_k} + n_k \qquad k = 0, 1, \ldots, N-1 \tag{3.1}$$

where n_k represents the additive white Gaussian noise (awgn) superimposed at the signal through the transmission channel.

After exponentiation we get:

$$(r_k)^4 = (a_k e^{j\theta_k} + n_k)^4 \tag{3.2}$$

and subsequently

$$r_k^4 = e^{4j\theta_k} + m_k \tag{3.3}$$

The amplitude a_k of the signal equals to 1. Moreover the AWGN (Additive White Gaussian Noise) n_k is removed from the system. This happens because the phase estimator described above makes up a filter which affects the signal phase for $2N+1$ discrete values.Acquiring the signal phase and dividing it by four in order to counterbalance the effect of exponentiation, we are able to get an estimation of the symbol's phase θ_r^k. A mathematical form of the phase estimation at the k_{th} block is described below:

$$\theta_r^k = \frac{1}{4}\measuredangle\Big(\sum_{l=s_k}^{s_k+M_s+1} r_k^4\Big) \tag{3.4}$$

where M_s represents the filter's size and s_k is the index of the first symbol belonging to the k_{th} block.

The operator $\measuredangle(\cdot)$ at the eq 3.4 gives us an estimation of the phase at the interval $[0, \frac{\pi}{2}]$. Therefore we conclude that choosing a big value for the M_s variable can eliminate the awgn but on the other hand it dooms the system to produce a bigger error in the phase estimation especially when the phase noise is big and therefore it alters the phase accordingly fast. This observation drives the need to perform an investigation on choosing the most efficient filter length in order to not only eliminate the awgn but also protect the performance of the system from the phase noise.

3.1.2 Phase Unwrap

The symbols thus obtained have the phase ambiguity by $\pi/2$ because we cannot know the absolute phase. It is important to note that the data should be differentially precoded. Differentially decoding the discriminated symbol after symbol discrimination, we can solve the phase ambiguity problem although the bit-error rate is doubled by error multiplication. The phase estimate $\theta_e(i)$ ranges between $-\pi/4$ and $+\pi/4$. Therefore, if $\theta_e(i)$ exceeds $\pi/4$, the phase jump of $\pi/2$ occurs inevitably as shown in Fig 3.3.

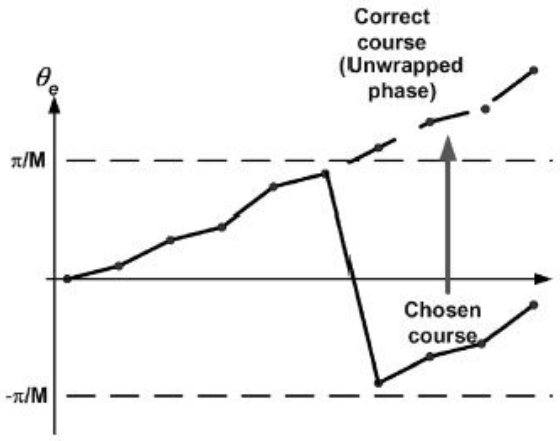


Figure 3.3: Phase jump and its correction during the phase estimation process. The correct course of the phase drift is determined by removing the phase jump.

To cope with this problem, the correction for the phase jump is done by obeying the following rule:

$$p = 0.5 + \frac{\pi}{2}\lfloor \Delta\phi[n-1] - \Delta\phi[n] \rfloor \tag{3.5}$$

This adjustment ensures that the phase estimate follows the trajectory of the physical phase and cycle slips are avoided [**?**].

3.1.3 Fine tuning of VV4PE algorithm parameters

The VV4PE algorithm parameters that need to be further investigated for the system's improvement are the following:

- Filter's type
    1. Moving average filter
    2. Block filter
    3. Wiener filter
- Filter's length

To begin with, we will investigate the behavior of the algorithm using a 4QAM modulated signal. Our findings will be further assessed with 16QAM modulation which is eventually our implementation aim.

Filter's type

As stated above we are choosing between three rather simple but powerful approaches for filtering out the phase noise of the symbols.

A **Moving Average Filter** is a widely used filter in the domain of signal processing. It is a simple approach that outputs great results. It is optimal for reducing random white noise (which is the case for the random walk phase noise) and it has a rather simple design. Its functionality can be summed up in the paragraph below supported by the figure 3.4 [15].

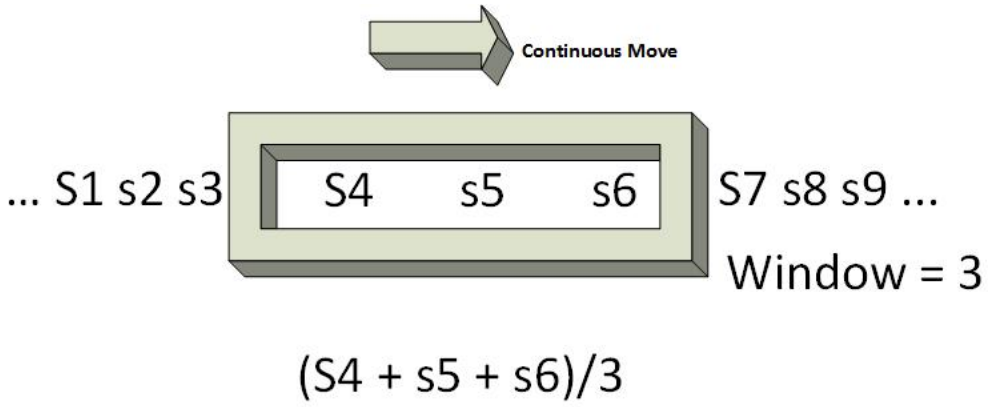


Figure 3.4: Moving average filter's operating principle.

Given a series of symbols and a fixed subset size (filter window), the first element of the moving average is obtained by taking the average of the initial fixed subset of the symbol series. Then the subset is modified by "shifting forward", that is, excluding the first symbol of the series and including the next symbol following the original subset in the series. This creates a new subset of symbols, which is averaged.

On the other hand a **Block Filter** is a filter averaging the symbols in its window range but in a slightly different manner (as presented in the figure 3.5 below).

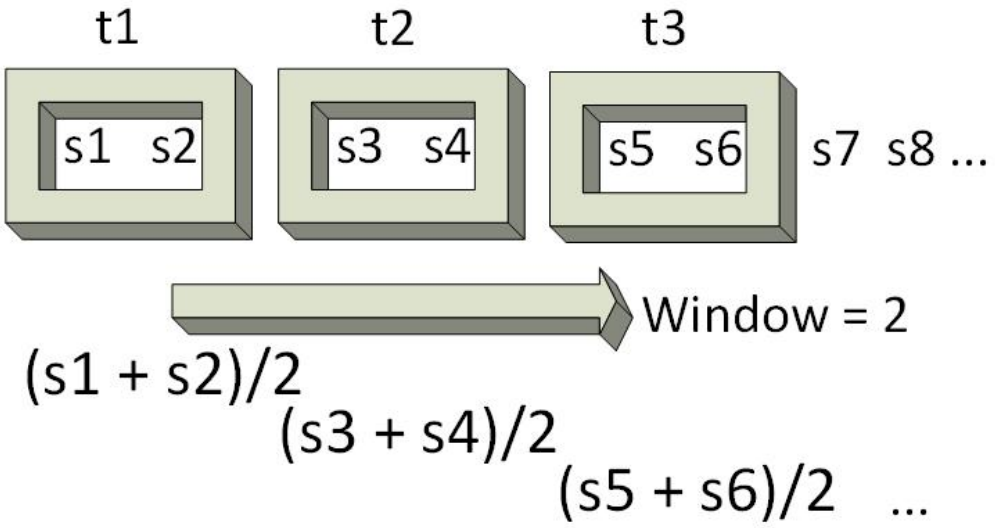


Figure 3.5: Block average filter's operating principle.

The difference with the rolling average filter is that the filter window in this case is not "rolling". That means that the output is one average value for every N-values contained inside the filter window.

Last but not least **Wiener Filter** and more specifically its FIR approximation, is in terms of simulation the best filtering algorithm we applied at the incoming signal. And that is, because Wiener-filter coefficients consist of two exponentially decaying sequences that are symmetric about the center of the filter. That attribute , in our case proves to be really important since the filter gives decreasing emphasis to the phase estimates that are temporally far away from the symbol which the system is now processing. That way the fitter dumps excessive white noise superimposed at the signal while at the same time it tracks successfully any rapid or slow change the phase might exhibit.

From a hardware implementation point of view though, one has to take into account different metrics and attributes of the filter. A Wiener Filter might be the best filter in terms of performance but it lacks design simplicity. Unequal coefficient values, especially exponentially decaying ones pose a hurdle to the hardware designer. One that he is not willing to surpass if the performance gain is not going to be astonishingly lower than with the other filters.

On the other hand moving average, and block average filters are relatively easier and cheap to implement in hardware. The moving average though might require slightly bigger effort but they both remain a good trade off between hardware simplicity and good performance.

A comparison of the filtering techniques can be shown below at the figure 3.6. It is clear that the lowest value achieved by all 3 filters is quite the same they differ greatly in another domain. It is evident that the Wiener filter behaves better in terms of linewidth. For a higher linewidth value Wiener produces acceptable results (less than 10^{-3}). Then comes the moving average filter and the least good behavior is produced when using the block average filter.

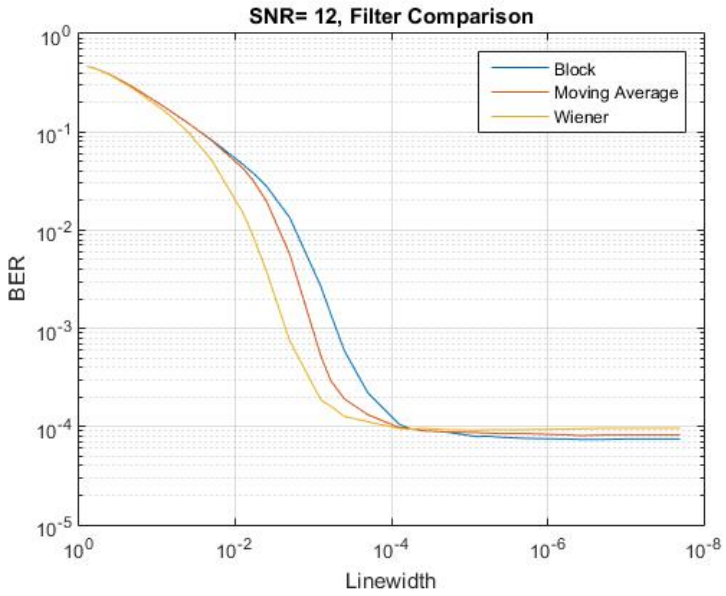


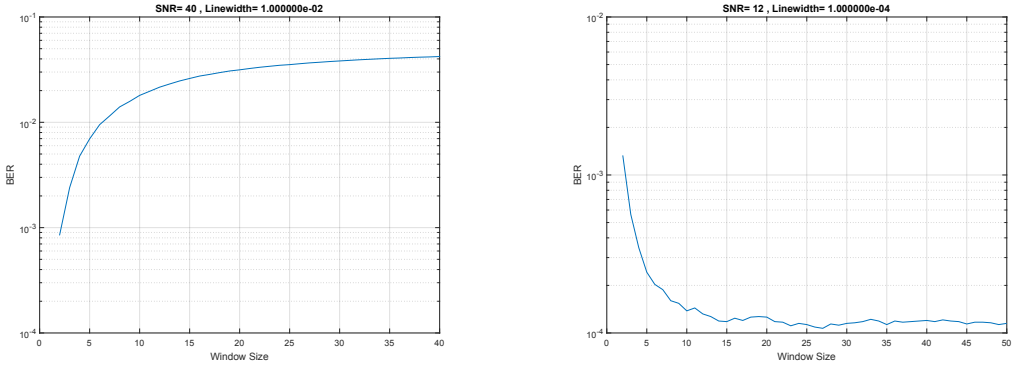
Figure 3.6: Plot of the BER value produced while utilizing different filtering techniques. While all the filters have the same minimum value the Wiener one outperforms the others in the linewidth domain.

Filter's length

Having discussed upon the filter type we can go on investigating the effect of the filter length at its performance. Intuitively, we expect the filter's length to be indirectly proportional to the linewidth. The higher the phase noise (the lower the linewidth) the smaller the block size should be. As explained before, the phase noise is a random walk process. With amplified noise one should keep the filter small in order to track the phase changes successfully along the incoming symbol series.

On the other hand when the signal does not suffer from high phase noise then by making the filtering window bigger we guarantee less white noise present at the signal.

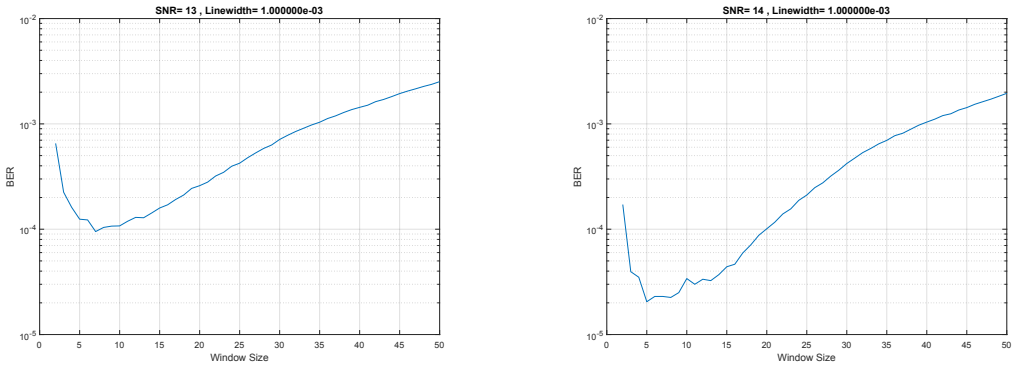
In the figures 3.7a and 3.7b below we can see the curves showing the optimal filter value ,which is of course the value that minimizes the BER value.



(a) Optimal Length = 2 (b) Optimal Length = 27

Figure 3.7: Optimal filter length

During this process we noted that the optimal value depended more upon the linewidth value and less upon the SNR value. That means that the values computed for the filter's length are not optimal since those depend on both the SNR and the linewidth values. Nevertheless, they fit our needs since they produce good results without needing the computing power that an optimization problem with many parameters needs. That is evindent from the curves below (figures 3.8a, 3.8b).



(a) $SNR = 13, Linewidth = 1 \cdot 10^{-3}$ (b) $SNR = 14, Linewidth = 1 \cdot 10^{-3}$

Figure 3.8: The optimal filter length hardly changes while varying the SNR value. The BER on the other hand is getting better.

Here we can see a table (table 3.1) containing the computed near-optimal number of filter taps for a range of linewidth values.

Optimal Filter Length (VV4PE)								
High Linewidth			Mid Linewidth			Low Linewidth		
10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
2	3	7	27	37	70	63	63	63

Table 3.1: Near-optimal number of filter taps for the various linewidth values.

We clarify again here that the search for the near-optimal values has been based on keeping the SNR value stable and varying the linewidth value. This of course produces sub-optimal results but we accept the values produced since we noticed that the system's filter length gets affected more by the linewidth and less by the SNR.

Combining the above computed parameters we get the following contour plot for a system using VV4PE and 4QAM modulation.

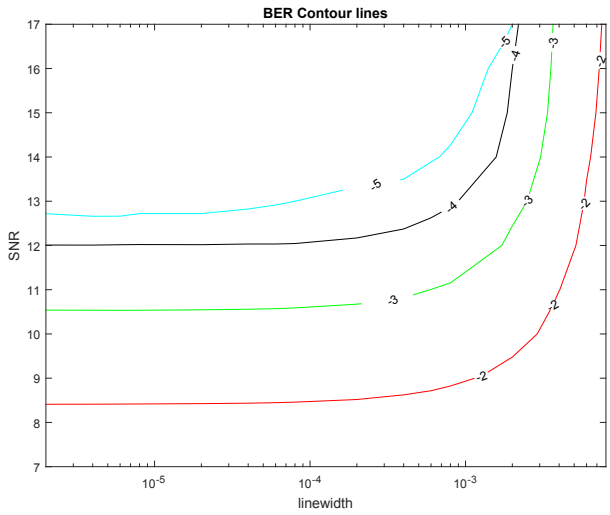


Figure 3.9: Contour plot showing the relationship between the algorithms performance in combination with laser linewidth and SNR. Colored lines represent a constant value of BER

3.1.4 Performance on bigger modulation schemes

As stated before the performance of the VV4PE algorithm proves to be sub-optimal for higher modulations such as 16-QAM. That is of course expected since the basis of retrieving the phase error lies upon the modulation stripping. That is done by exponentiating the symbol to the fourth power as analyzed before. The problem is that a 16-QAM constellation will not have all of its points lying in the equidistant 90 ° axes. That way the fourth power operation will not strip the modulation from these symbols entirely but it will leave them symbols lying above or below the axis line even if they do not contain any phase noise. The problematic symbols are shown in the figure 3.10 below.

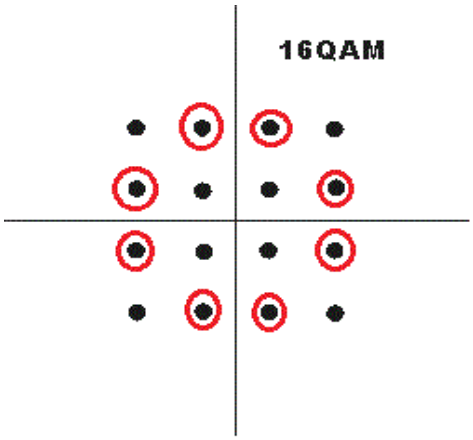


Figure 3.10: The symbols indicated with red circles are the ones unable to strip their modulation with the fourth power operation.

From the table 3.2 below it is evident that a VV4PE algorithm operating on 16QAM modulation needs really big filter values that make the use of such an algorithm non-viable.

Optimal Filter Length (VV4PE)								
High Linewidth			Mid Linewidth			Low Linewidth		
10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
95	95	95	95	110	126	150	163	165

Table 3.2: Near-optimal number of filter taps for the various linewidth values of a 16QAM modulated signal using VV4PE algorithm.

The contour plot below (figure 3.11) proves that even with those big filters the system needs significantly lower linewidth noise in order to yield acceptable BER values. That means a lot more cost for a hardware implementation , since the lower linewidth is achieved with more precise and thus more expensive equipment.

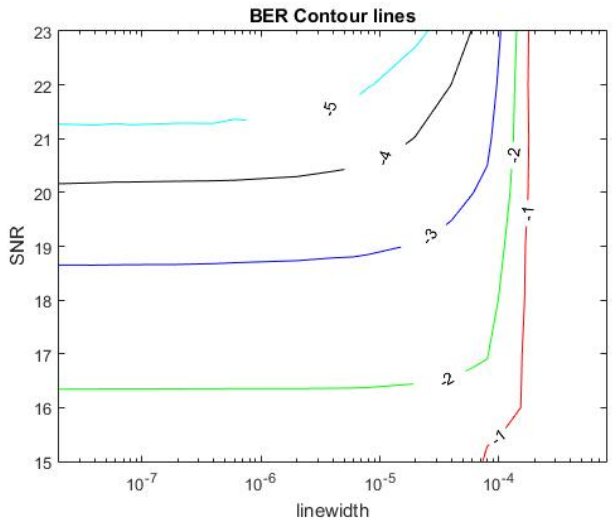


Figure 3.11: Contour plot showing the relationship between the algorithms performance in combination with laser linewidth and SNR. Colored lines represent a constant value of BER

3.1.5 Conclusions about Viterbi Viterbi algorithm

Viterbi - Viterbi algorithm proves to be a simple yet powerful solution to perform phase error correction. Its simplicity is though its greatest drawback since it is inadequate to yield reliable performance for higher than 4QAM constellations. Yet the idea of modulation stripping, which will be shown later, along with some modifications can churn out great results even on higher order constellations.

3.2 Blind Phase Search

3.2.1 Algorithm's description

Blind phase search algorithm is a forward error correction algorithm. Viewed from the control system point of view it does not contain any feedback path. In the figure below (fig.3.12) we present the schematic diagram of the algorithm .

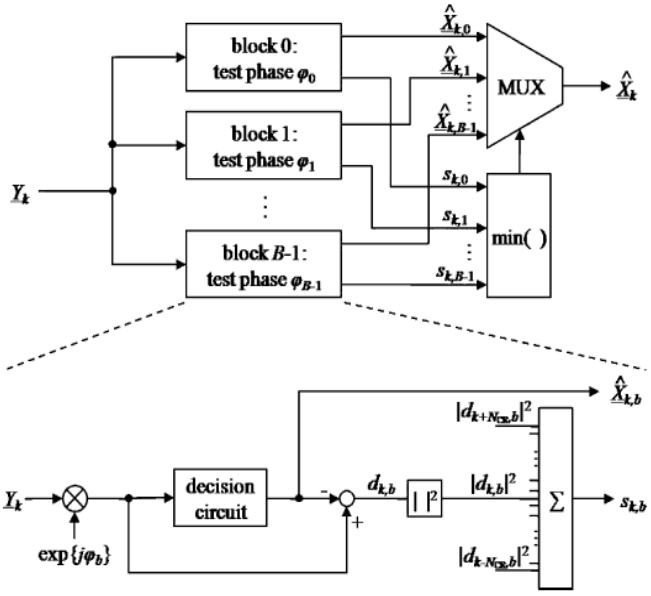


Figure 3.12: Carrier phase recovery using B test phase angles φ_b

At first, the incoming signal Z_k is rotated B times at φ_b angle units per time.

$$\varphi_b = \frac{b}{B} \cdot \frac{\pi}{2} \quad b \in \{0, 1, \ldots, B-1\} \tag{3.6}$$

We then compute which is the closest constellation point at the already rotated symbol, as well as its squared distance from it.

$$|d_{k,b}|^2 = |Z_k \exp(j\varphi_b) - \hat{X}_{k,b}|^2 \tag{3.7}$$

The angle for which we get the minimal total distance is the optimal rotation angle. To reduce the effect of noise in our system we take the sum of $2N+1$ consecutive distances rotated by the same test angle (block filtering). φ_b

$$s_{k,b} = \sum_{n=-N}^{N} |d_{k-n,b}|^2. \tag{3.8}$$

As mentioned before the sum $s_{k,b}$ represents the the optimal rotation angle. The latter acquires a unique value at a window of $2N+1$ symbols.

Two of the most important algorithm parameters are :

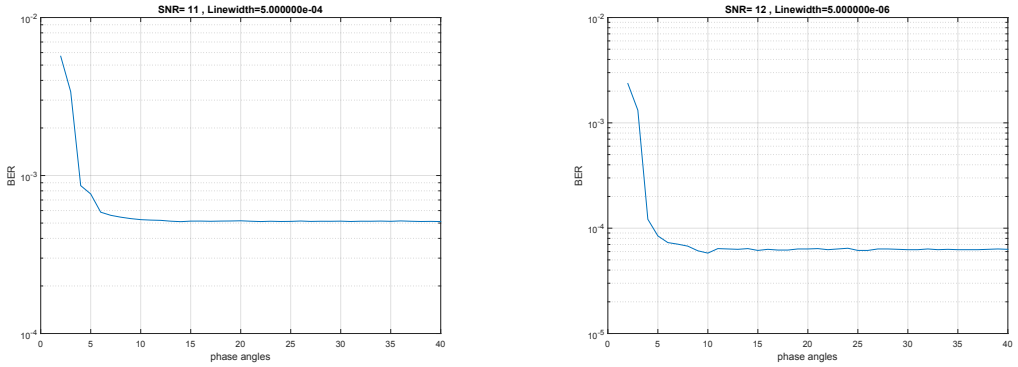
- The number of test rotation angles B
- The block filter's length N

At the next chapter we will experiment with those parameters and end up with their optimum values.

3.2.2 Fine tuning of BPS algorithms parameters

In this section we will investigate the optimal parameter values for the algorithm. We are starting by experimenting with 4QAM modulation and we will go on discussing bigger modulation schemes like 16QAM. In general, blind phase search algorithm is operating greatly independent of the modulation scheme as we will show later.

As stated at [2] the ideal values for the filters half length is derived from the product of laser's linewidth with the symbol rate. A value range of $N = 6, \ldots, 10$ is considered to be a fairly good choice. So we start experimenting by setting the filter length to $N = 10$. By sweeping the various values of B we get the plots below (fig 3.13a and fig 3.13b).



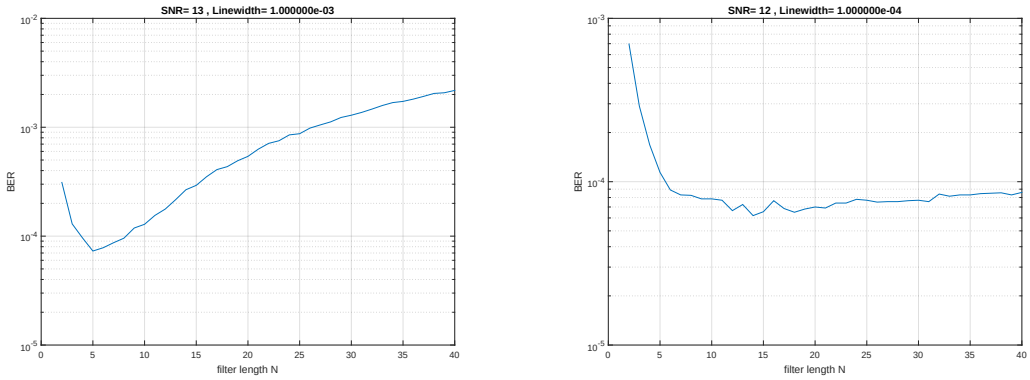
(a) $SNR = 11 \quad Linewdith = 5 \cdot 10^{-4}$ (b) $SNR = 12 \quad Linewdith = 5 \cdot 10^{-6}$

Figure 3.13: System BER results for various values of test phase angles (B).

We note from the figures 3.13a and 3.13b that the value B= 10 gives us really good results at the simulation. This is the value that seems to set the BER value to a stable point. We also observe that increasing the number of test phase angles does not improve the actual BER value achieved. This is due to the fact that choosing a specific value of B, big enough to produce an acceptable value of BER, means that we have "sliced" the cartesian plane enough and any further "slicing" by increasing B will not improve the performance dramatically.

After settling with a value for rotation test angles it is time to improve the algorithms filter. Intuitively we claim that its should be indirectly proportional to the laser linewidth.

Indeed from the figures below (3.14a, 3.14b) we note that in order to obtain optimal results the filter length should dynamically vary according to both the linewidth and the SNR values. Like in the VV4PE algorithm above a more naive search has been done to investigate some near-optimal results varying only the linewidth for a stable SNR value since the main factor affecting the filter length is linewidth. Those values are presented in the table below 3.3).



(a) Optimal filter length = 5

(b) Optimal filter length = 14

Figure 3.14: BER results for various values of filter length N

Optimal Filter Length (BPS)4QAM								
High Linewidth			Mid Linewidth			Low Linewidth		
10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
N/A	N/A	5	14	36	38	56	66	72

Table 3.3: Near-optimal number of filter taps for the various linewidth values. Values "N/A" indicate the fact the for these linewidth values no SNR value no matter how big could give us an acceptable BER value.

By simulating a carrier phase recovery system having the parameters computed above and implementing the BPS algorithm we get the following contour plot.

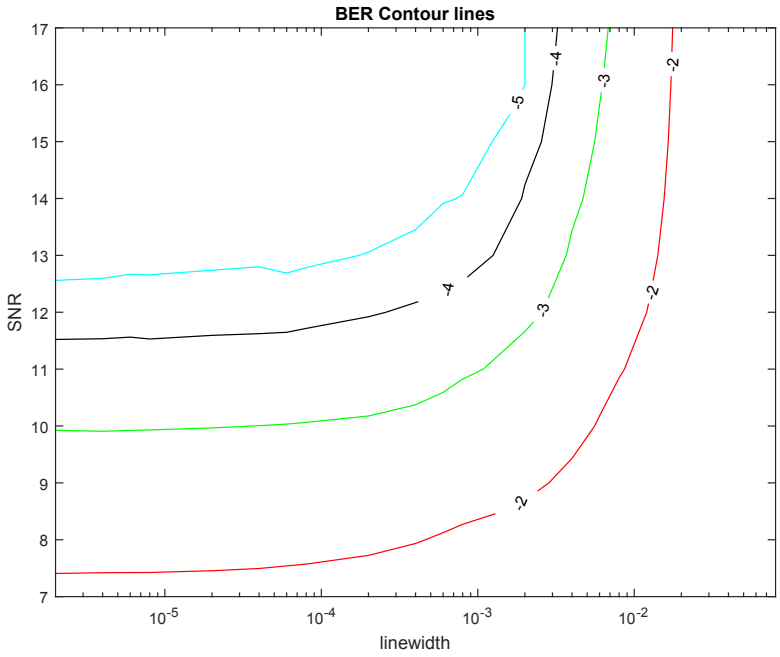


Figure 3.15: Contour plot showing the relationship between the algorithms performance in combination with laser linewidth and SNR. Colored lines represent a constant value of BER

3.2.3 Performance on bigger modulation schemes

Observing the algorithm we can deduce the feasibility of its successful operation not only on a 4QAM modulation like above but also in higher modulation schemes like 16QAM. That is indeed true since the algorithm itself is not depending on any specific information from the underlying signal modulation but it is rather treating every modulation scheme with the same manner no matter if it contains 4 signal states (4QAM) or 16 (16QAM) or even 32 (32QAM).

That way by applying the same method as for 4QAM we derive the optimal test phase angles number (fig 3.16), an optimal filter length table (table 3.4) and the respective contour plot showing the relationship of linewidth and SNR (fig 3.17).

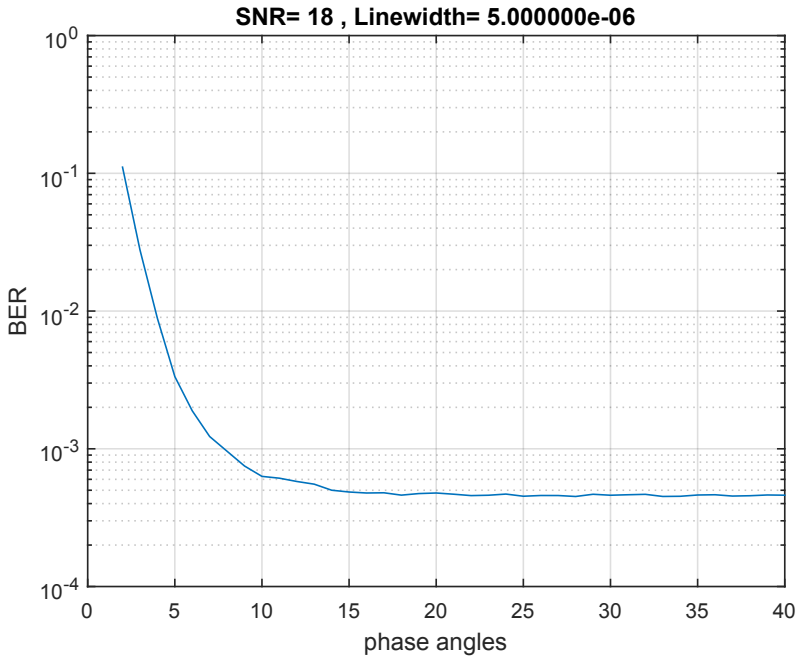


Figure 3.16: BER results for various test phase angles B. We note a bigger optimal value of B = 18

Optimal Filter Length (BPS)16QAM								
High Linewidth			Mid Linewidth			Low Linewidth		
10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
N/A	N/A	6	12	20	28	49	66	75

Table 3.4: Near-optimal number of filter taps for the various linewidth values. Values "N/A" indicate the fact the for these linewidth values no SNR value no matter how big could give us an acceptable BER value.

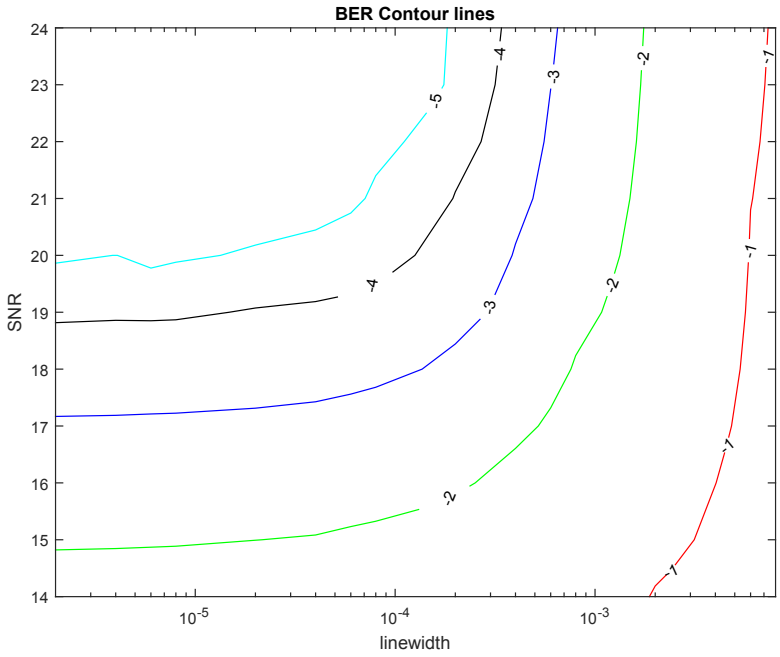


Figure 3.17: Contour plot showing the relationship between the algorithms performance in combination with laser linewidth and SNR. Colored lines represent a constant value of BER

3.2.4 Conclusions about BPS algorithm

As seen before the algorithm's simulation provides us with good results. BPS is an algorithm operating in a purely feed forward manner and not containing any feedback path. Feedback can be a dissuasive factor for a system that needs to provide a real time response and used in the realm of optical communications. The biggest advantage of BPS is surely its universality when it comes to the different modulation schemes. It is indeed possible and viable to use BPS in QPSK, 16-QAM, 32-QAM and also in bigger modulation schemes (symmetric or even non symmetric). Its drawback, though, in comparison with other carrier phase recovery algorithms is its advanced complexity. BPS is a "brute force" algorithm requiring many computations for every incoming symbol, especially in high noise conditions where we need a big number of test phase angles.

3.3 NLS Estimator

3.3.1 Algorithm's description

In this section we are going to analyze a low complexity, yet powerful blind feed forward carrier phase estimator. The proposed estimator represents a generalized form of a low SNR-approximation of the maximum likelihood (ML) estimator, that was originally proposed by Viterbi and Viterbi(VV4PE) as a blind carrier phase estimator for fully modulated phase-shift keying (M-PSK) transmissions [16]. During Viterbi and Viterbi algorithm analysis we bumped into an obstacle while trying to move on to higher than 4QAM constellations. The fourth power operation was

not enough for a symmetric 16QAM constellation due to its indisposition to strip the modulation format off the symbols not lying in the equidistant 90 ° axes. Such schemes, therefore rely on alternative methods such as QPSKpartitioning [17] or the computationally intensive Blind Phase Search (BPS) algorithm mentioned before [2].

In this algorithmic approach we can at first denote the complex signal transmitted from the transmitter side as $a_c + jb_c = \rho_c e^{j\phi_c}$

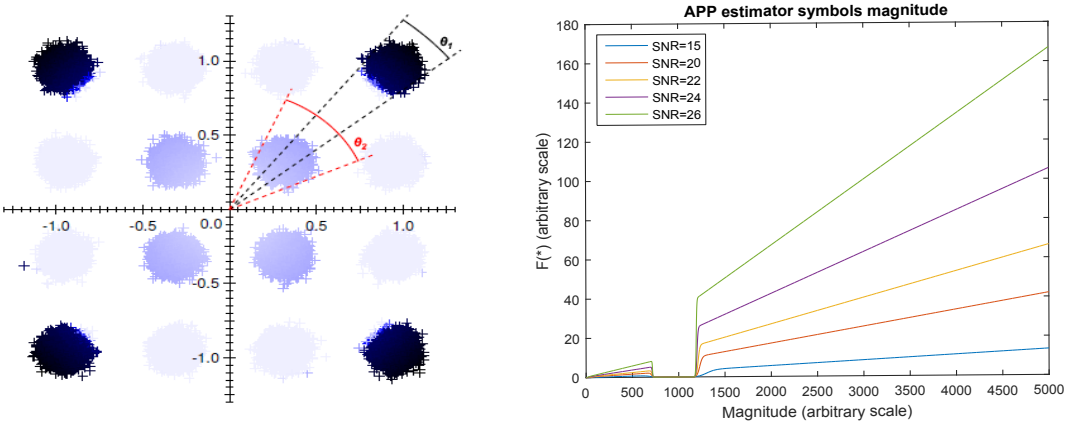
After fiber transmission and intradyne detection, the transmitted symbols are corrupted by AWGN and phase noise due to the combined linewidths of the free-running Tx (transmitter) and Lo (Local Oscillator) lasers [3]. Following ideal symbol clock recovery and equalization in DSP, the received noisy sequence at times t where $t = nT_s (n \in \mathbb{Z})$ can be denoted in polar form as $\chi[n] = \rho[n]e^{j\phi[n]}$. In the proposed scheme, a nonlinear transformation is applied to $\chi[n]$:

$$y[n] = F(\rho[n])e^{j\phi[n]} \tag{3.9}$$

The resulting complex signal, $y[n]$, is then passed through an FIR filter (we will discuss on its type later) $w[k]$, with L coefficients. The phase estimate at the $n - \Delta$ received symbol (where $\Delta = \frac{L-1}{2}$) is then given by:

$$\theta[n - \Delta] = \frac{1}{4} arg\{\sum_{k=0}^{L-1} w[k]y[n - k]\} \tag{3.10}$$

in Eq 3.9 $F(\cdot)$ is a non-linear function that is dependent on the modulation format. When considered in the context of the overall algorithm, it constitutes a weighting function that determines how much each received noisy symbol contributes to the final phase estimate, depending on its magnitude. The fig 3.18b below shows plots of $F(\cdot)$ as a function of the received symbol magnitude for 16-QAM and several SNRs. Clearly, the phases of symbols having magnitudes between ≈ 0.7 and 1.2 do not contribute at all to the final estimate; this is because they belong to the middle ring of the constellation, with phases that do not lie on equidistant 90 ° axes. In contrast, only the symbols belonging to the innermost and the outermost rings are assigned weights toward the final estimate, as each of these rings forms a QPSK subset. Consequently, the multiplication of the phases of these symbols by 4 in Eq. 3.9 results in complete modulation stripping, thus allowing carrier isolation and accurate phase tracking [3].



(a) Cartesian Plane symbols

(b) Fapp Estimator

Figure 3.18: F() output values and the corresponding 16QAM symbols color highlighted according to the weight given to them by F()

Fig. 3.18a depicts noisy 16-QAM symbols, with the weights applied by $F(\cdot)$ indicated by varying color intensities. Clearly, the middle ring symbols are disregarded (These symbols correspond to the middle near zero region of fig. 3.18b). The plot also illustrates another aspect of the non-linear function: Even amongst the symbols that do lie on the equidistant 90 ° axes, those with higher magnitudes contribute more to the final estimate than the symbols with lower magnitudes. The intuitive explanation is that the phase error due solely to AWGN is lower for higher-amplitude symbols ($\theta 1 < \theta 2$ in Fig3.18a). The estimate from higher magnitude symbols will therefore be more accurate, and giving more weight to this will yield better estimation.

3.3.2 Fine tuning of NLS algorithm parameters

The function $F(\cdot)$ that minimizes the asymptotic covariance of the phase estimator is given by the equation

$$F(\rho[n]) = \frac{g_2(\rho[n])}{g_1(\rho[n]) - g_3(\rho[n])} \tag{3.11}$$

where

$$g_1(\rho[n]) = (-1)^{i-1}\frac{8\rho[n]}{M\sigma^2}e^{(-\rho^2[n]/\sigma^2)}\sum_{\rho_\sigma,\phi_\sigma\in\beta}[cos(4(i-1)\phi_\sigma)e^{(-\rho_\sigma^2/\sigma^2)}I_{4(i-1)}(\frac{2\rho[n]\rho_c}{\sigma^2})] \tag{3.12}$$

where σ^2 is the AWGN covariance $I_n(z)$ is the n^{th} order modified bessel function of the first kind and β is the set of M constellation points.

With the aid of numerical computation environment and by the use of the above equations (3.11, 3.12) we get a matrix with the necessary values of the $F(\cdot)$ function for $SNR = 10 \ldots 40$dB which is an adequate range for both the lowest and highest BER values possible.

The algorithm parameters that affect the performance of Fapp estimator are

- Filter's type

1. Moving average filter
2. Block filter
3. Wiener filter

- Filter's Length
- Output scale of the $F(\cdot)$ function

Filter's type

In its essence Fapp algorithm shares high resemblance with VV4PE algorithm which we analyzed in a previous section. Having covered the same comparison at that exact section we can safely conclude about the performance of its type of filter. We expect Wiener filter to yield the best performance, at the cost of an overall bigger complexity. On the other hand the simpler filters (moving average, block) produce again a sufficiently good BER so as to consider a hardware implementation using them.

More specifically we can see in the plot below the performance of the three aforementioned filters and the comparative advantages among them.

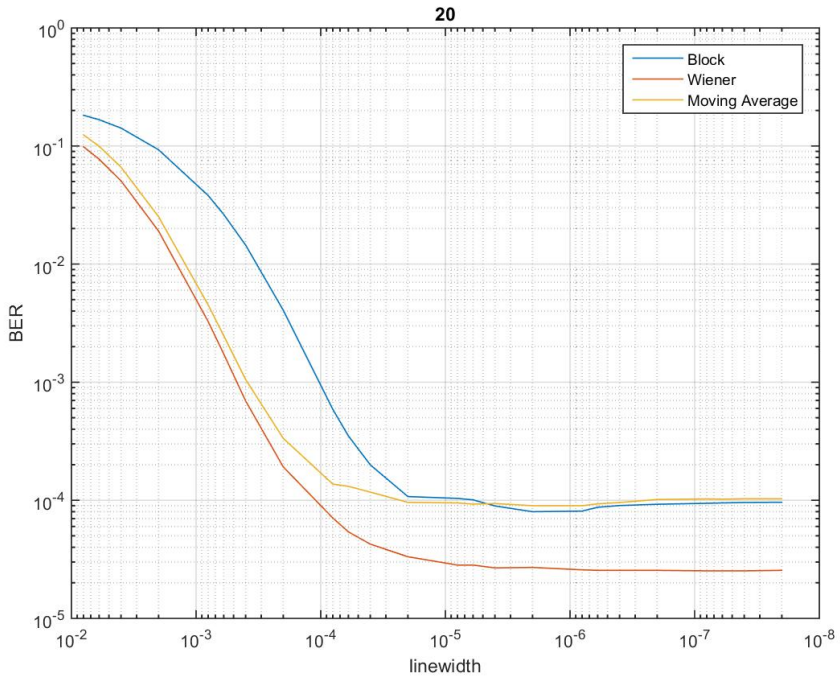


Figure 3.19: Plotting linewidth versus BER for all three filters (SNR = 20). A clear performance gain is noted for Wiener filter followed close by the moving average filter

While the performance gain is by far in favor of the wiener filter we note that in terms of linewidth performance the less simple moving average filter does not differentiate by a big performance gap. In fact both filter have identical performance up until linewidth = 10^{-3}. From linewidth values less than that one moving average filter moves towards the block filter BER value

whereas wiener filter creates a big gap of circa one order of magnitude. Although we should not forget the much simpler design of the moving average filter comparing it to the more complicated.

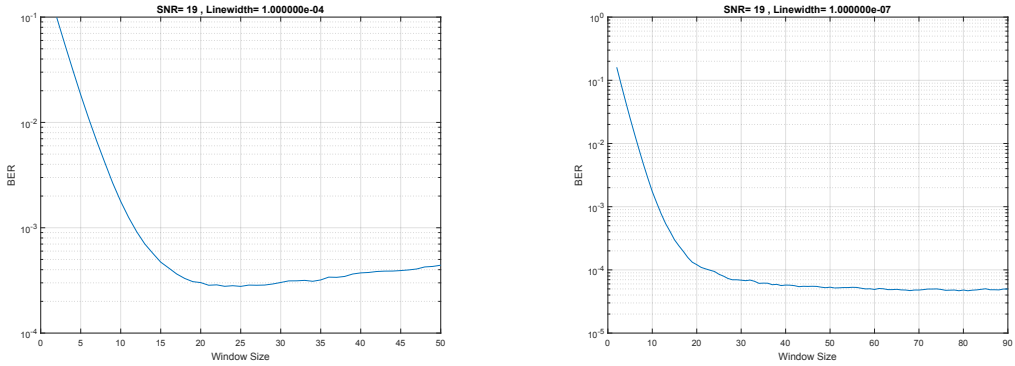
Filter's length

Having decided upon the filter's type (moving average) we move on to decide upon the optimal number of taps it should contain. But that is not a one size fits all answer. The performance of the system and also the filter relies not only on the linewidth of the laser but also on the respective SNR for that linewidth. So for every linewidth-SNR pair there is presumably a filter length value that minimizes the system's error and provides us with optimal results. The table 3.5 below presents us the values that we computed making the simplification, which holds true, that the main factor affecting the filter;s length is the linewidth of the system.

Optimal Filter Length (APP Estimator)								
High Linewidth			Mid Linewidth			Low Linewidth		
10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
2	5	11	25	32	70	79	79	79

Table 3.5: Near-optimal number of filter taps for the various linewidth values.

The figures 3.20a and 3.20b below show some of the curves dictating the optimal values for the filter at the respective linewidth.

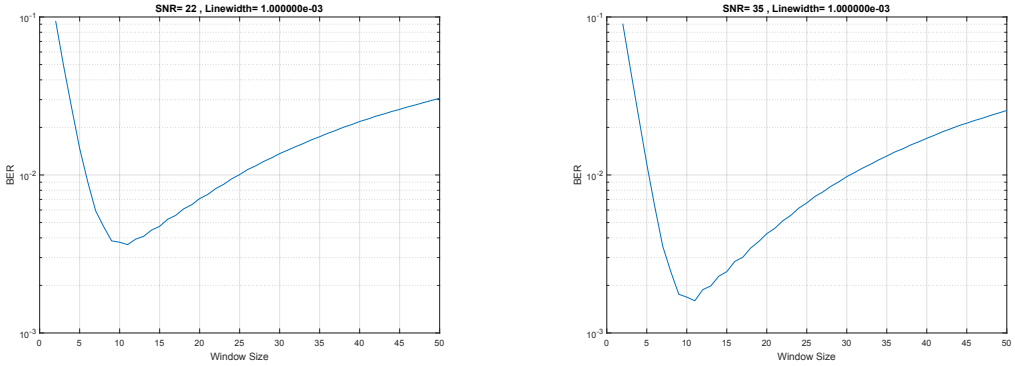


(a) Optimal Length = 25

(b) Optimal Length = 79

Figure 3.20: Optimal filter length

Lastly, in the figures 3.21a and 3.21b below we note that while the shape of the curve stays the same as the SNR changes , the BER value achieved is improving. This makes evident that the main factor affecting the optimal filter length and generally the performance of the system is the linewidth and not the SNR.



(a) $SNR = 22, Linewidth = 1 \cdot 10^{-3}$ (b) $SNR = 35, Linewidth = 1 \cdot 10^{-7}$

Figure 3.21: The optimal filter length 11 stays intact while varying the SNR value. The BER on the other hand is getting better.

By combining the above computed parameters we present a contour plot showing the relationship between SNR and Linewidth for a carrier recovery system of 4QAM modulation that is implementing the NLS estimator algorithm.

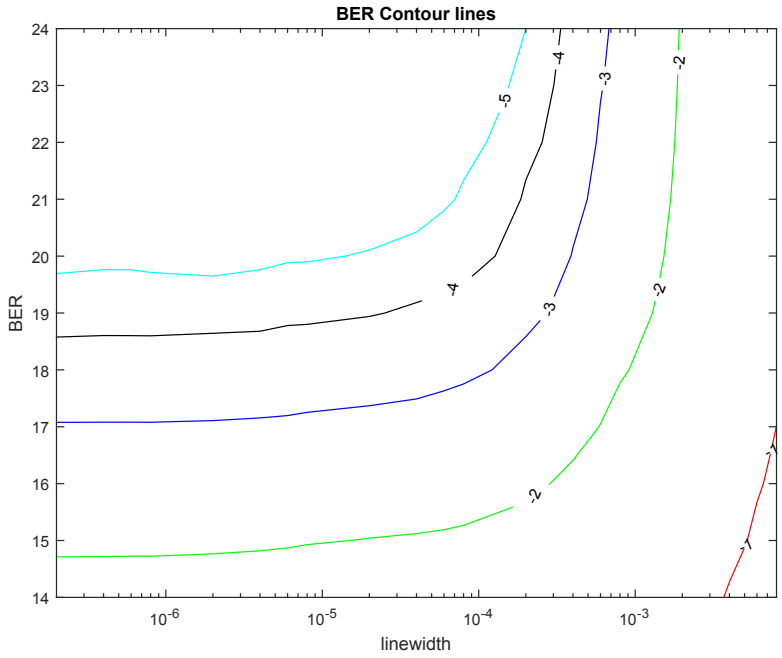


Figure 3.22: Contour plot showing the relationship between the algorithms performance in combination with laser linewidth and SNR. Colored lines represent a constant value of BER

The optimal length of the filter for every linewidth value has been computed by a imposing a

linear interpolation to the optimal values found before.

3.3.3 Performance on bigger modulation schemes

Although the operating principle of the APP algorithm does not prohibit it to perform phase error correction on higher constellations the scope of this research is targeted at 16QAM and we are going to limit our investigation to this modulation.

3.3.4 Conclusions about NLS algorithm

To sum up, the APP estimator algorithm provides a powerful yet simple alternative to the algorithms mentioned above. Its genius nature gets rid of the problems caused by the points not lying in the equidistant 90 ° axes while simultaneously exploiting the modulation stripping property of Viterbi and Viterbi algorithm. Its results seem really promising at first but it needs to be evaluated together with the benchmark algorithms (BPS, VV4PE) in order to conclude about its usefulness. The figure below gives us a bird eye view of the APP estimator algorithm behavior both on the linewidth and the SNR domain.

3.4 Comparison of the presented algorithms on 16QAM modulations

As we have already mentioned the purpose of this thesis is the implementation of the novel Fapp carrier phase estimator algorithm at a 16QAM modulation scheme.

For this reason and to sum up the previous investigation findings, we will now present a comparison of the above algorithms. This way we will obtain a theoretical insight on the relative performance between the Fapp algorithm and the commonly used Viterbi - Viterbi 4th power estimator and Blind phase search estimator.

A first glance at the comparison of the carrier phase estimation algorithms is presented below (fig 3.23).

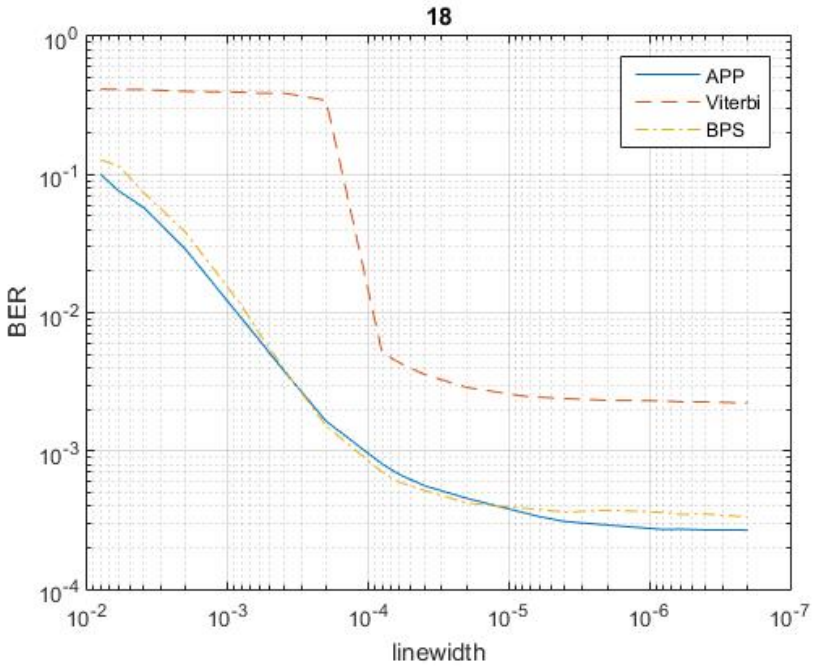


Figure 3.23: Linewidth versus BER plotting for all three estimator algorithms.

We immediately note the bad behavior of Viterbi - Viterbi algorithm in the linewidth domain. Especially in the high linewidth range ($linewidt \geqslant 10^{-4}$) the latter gives BER values way above the acceptable BER values (i.e values $\leqslant 10^{-3}$). On the other hand moving to the lower linewidth range the three algorithms present good BER values $\approx 2 \cdot 10^{-4}$. The NLS estimator though, produces the same if not better results for low linewidth values.

It is now evident, that the Viterbi algorithm does not pose a viable solution for a carrier phase recovery algorithm at the 16QAM modulation scheme. Its overall performance, especially in the high linewidth range , where good performance is mostly needed, is a prohibitive factor for a telecommunication pipeline implementation.

Finally to see both Fapp and BPS in full scale comparison we present the contour plot (fig 3.24) of linewidth versus SNR for both the above algorithms.

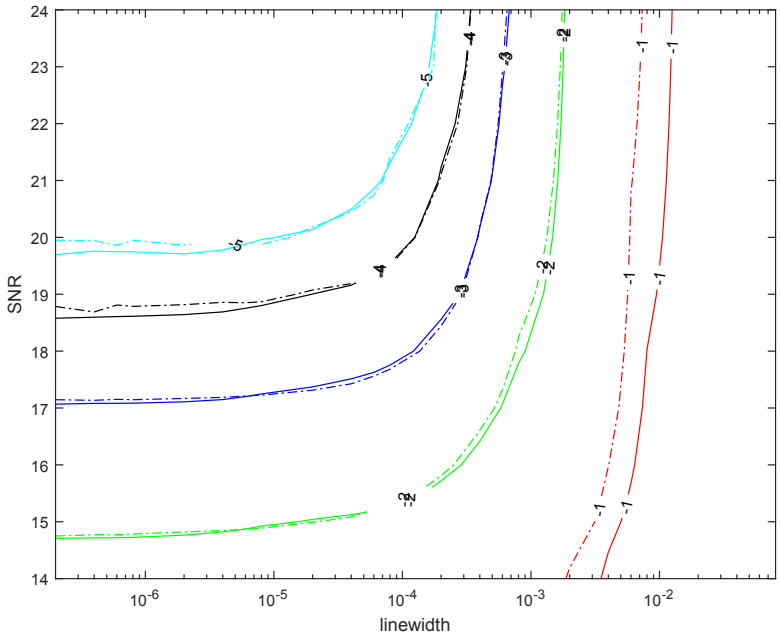


Figure 3.24: Contour plot of linewidth versus SNR for APP (continuous line) and BPS (dashed line).

From the plot presented above we can clearly see that whereas APP does not outperforms BPS on most cases (though it does performs better for on less noisier environment i.e lower linewidth) it does provide surprisingly good results given the lower complexity of it comparing it to the "brute force" costly BPS algorithm. All things considered it is definitely worth trying a hardware porting of the algorithm into an FPGA architecture in order to further asses its weaknesses and strong points.

However before getting to the hardware system design we should pay close attention to the form of the algorithm. Since, we set sail to this journey with implementation efficiency into mind we will firstly try, in the next chapter, to modify and simplify the algorithm so as to present more hardware friendly features needed for a successful hardware implementation.

Chapter 4

NLS EStimator Polynomial Approximation

4.1 Introduction

Every algorithm, especially a computational intense one (i.e DSP algorithms) needs extensive study in order to be able to port it successfully in the desired application specific architecture. Either targeting DSP's or FPGA's the individual building blocks of the algorithm have to be simplified in a manner suitable for hardware implementation.

At a first glance the algorithm we chose, which is of course APP estimator, seems to rely heavily on a QPSK partitioning scheme. This is used, as mentioned above, to evaluate a relative weight for every input symbol according to its magnitude. Different magnitudes get amplified differently.This is achieved through a rather complex mathematical formula that seems to oppose the simplicity notion we embraced above.

Mathematical functions are implemented in hardware in a couple of ways. The first and more frequently used is the lookup table method. This technique might be adequate for a couple of outputs but in our case we have to deal with one function per SNR value (each of them covering the whole magnitude axis). With such a large matrix, we could right away forecast a system bottleneck located just at the reading and writing operations on the latter one.

Another obvious solution would be to directly implement the mathematical function in hardware. But by taking a look at it one notices right away many "unwanted" terms like trigonometrical functions, divisions, exponential, powers and so on and so forth. Those kind of operations may be fast and painless in a general purpose processor but they truly become a designers nightmare in hardware. As we said above, our goal is to simplify as much as possible the algorithm and that necessarily means favoring, summations, subtractions and low memory usage.

So, the path we choose to take is the function approximation one. By that I mean that our ultimate goal was to end up with a set of coefficients, precise enough to let us adequately describe every function while having a small impact on the memory and computational logic. All the specific information on that subject are going to be presented to you on this chapter.

4.2 Visualizing the problem

The function we are going to evaluate is described by the equations 3.11 and 3.12 as presented on the second chapter.

We are seeking a way to approximate the function family presented below whose members are implemented through the APP estimator Bessel function described above.

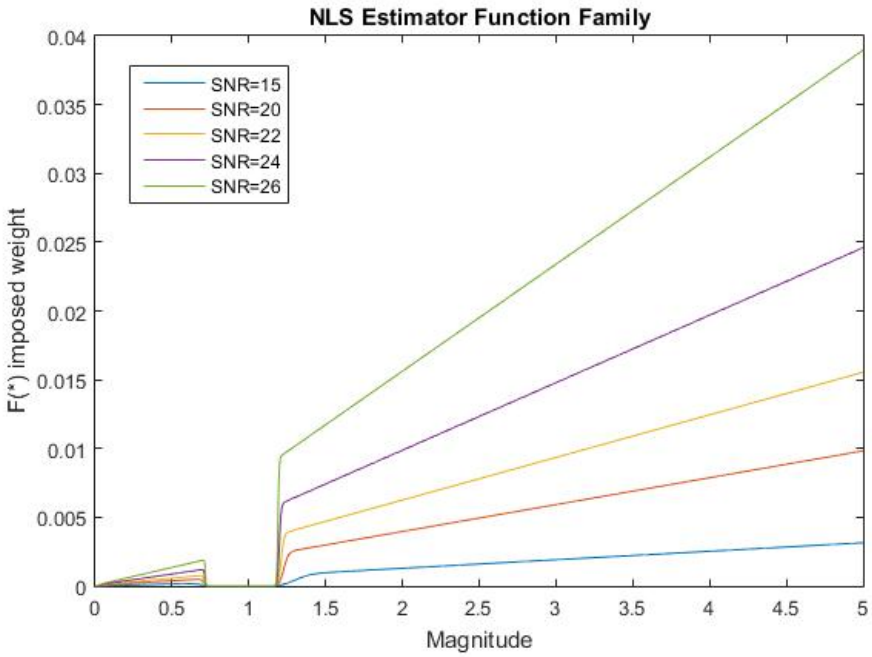


Figure 4.1: The APP estimator functions family. Each single curve represents a specific SNR value.

4.3 Method of polynomial approximation

We note that each function has 3 distinct parts. Let's call them A, B and C respectively. The A part spans from zero magnitude up until approximately 0.7.

The magnitude takes values from 0 up until 5. The normal 16QAM constellation has its outer symbols at the magnitude = 3. The reason the magnitude values are extended until 5 is due to the noise impact on the signal.

The C part spans from approximately 1.2 till the end. All that is left is the B part which seems to be constantly zero.

Therefore every part of the function can be described by some coefficients (i.e a straight line possibly which is described by a first order polynomial). So we could approximate all three parts and make a matrix holding those coefficients. From then on it would just be a matter of reevaluating the line equations and getting the value for the point in question just by providing the magnitude in the equation.

Suppose we have n SNR values. Then the matrix size would be $n \times 6$ where $6 = 3 \times 2$. 3 is for the number of partitions of every function (i.e A, B, C) and 2 are the first order polynomial coefficients (as in a straight line $\alpha\chi + \beta$). An example is shown in the figure 4.2 below.

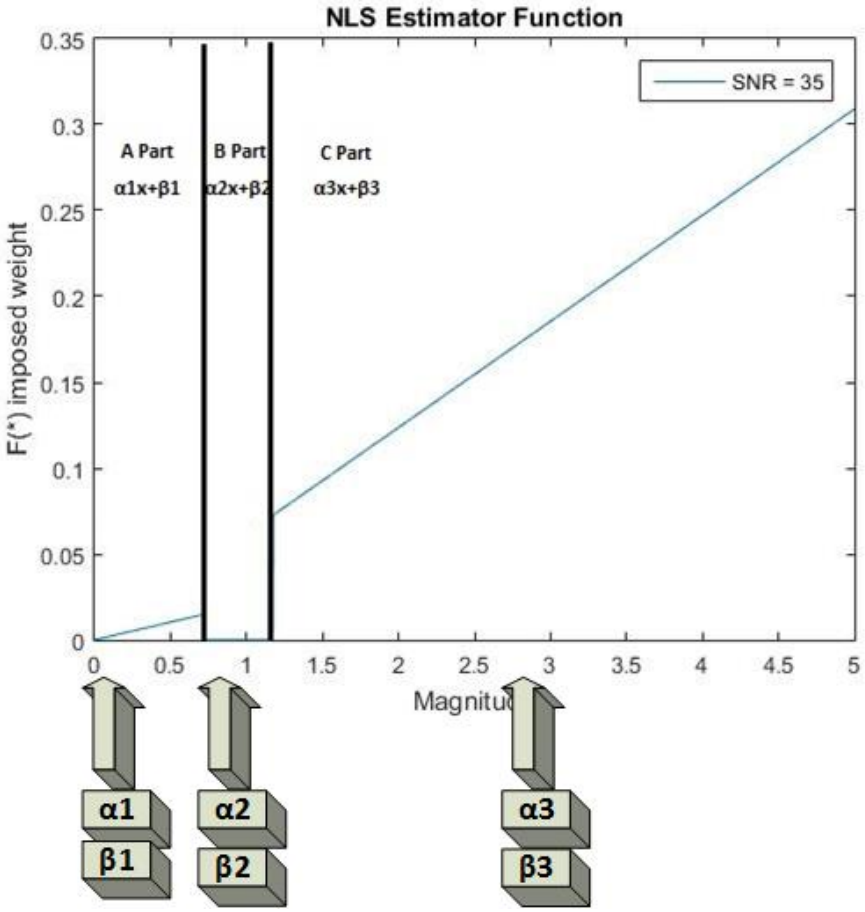


Figure 4.2: Every piece is being approximated by 2 coefficients since it is a straight line.

But still we have 3 pairs of coefficients for every function. That means that for a family of n functions we construct a matrix of n rows (one for every SNR value) and 6 columns (as in the example before). That process is shown in the figure 4.3 below.

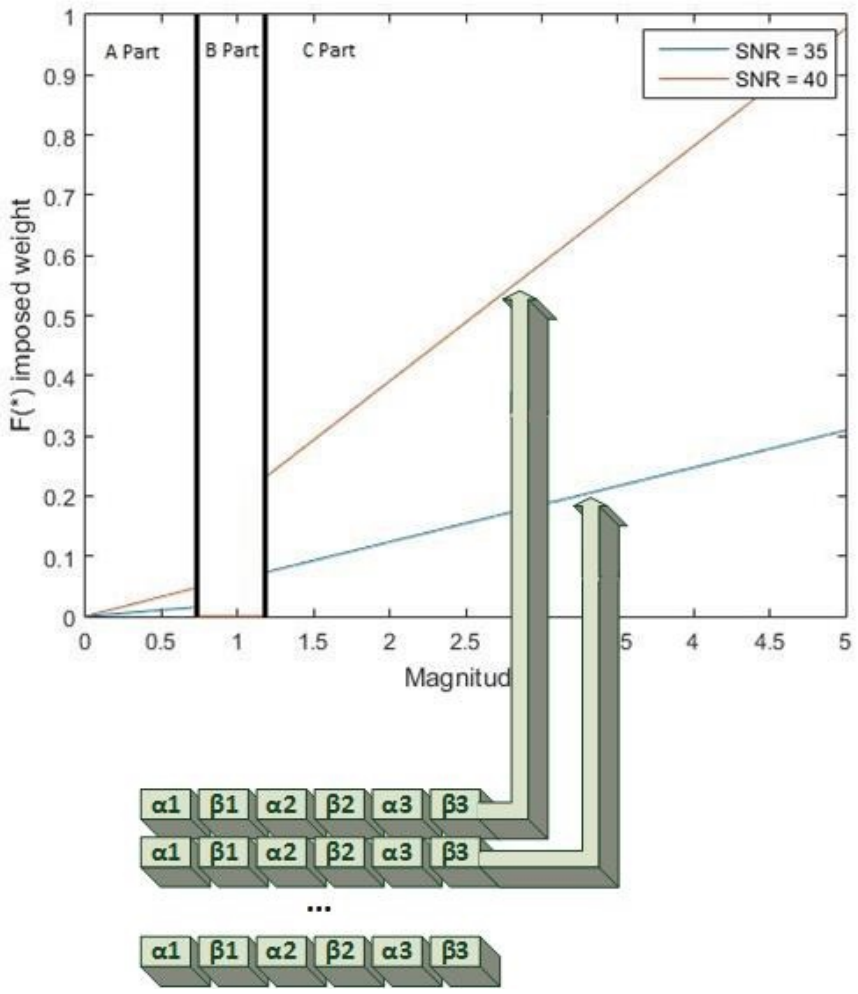


Figure 4.3: Every horizontal 6-column table represents a curve. A family with n curves is represented by a table n $\times$ 6.

If we isolate the coefficients α and β of a single partition then we can see that they also form a curve respective to the SNR values. An example can be seen below.

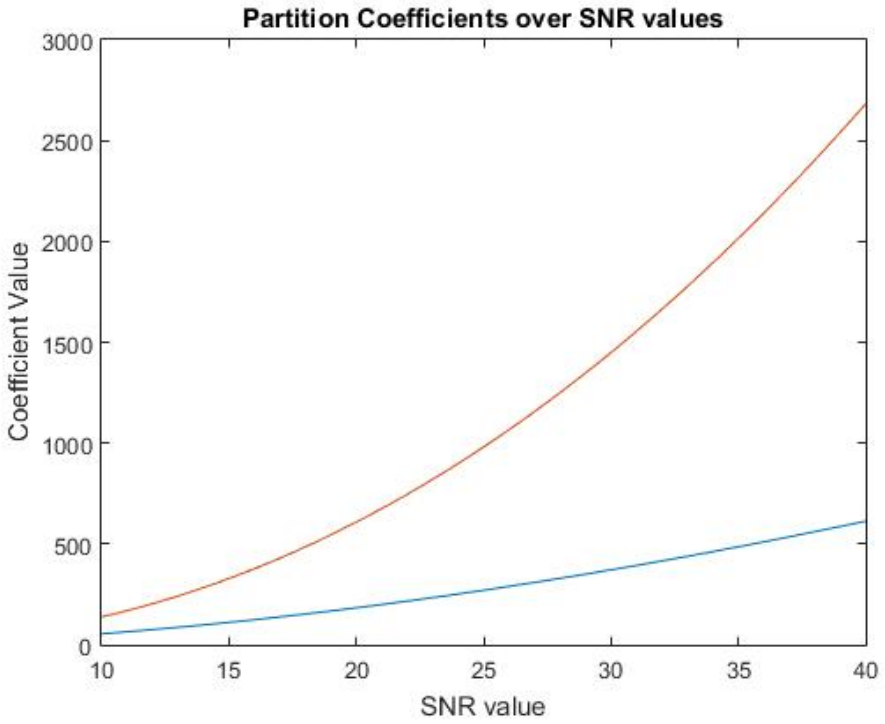


Figure 4.4: Sweeping over the coefficient values of a specific partition we get this plot. Each curve corresponds to a specific coefficient (here α and β).

By linear approximating those curves we get n coefficients for every curve (where n is the order of the polynomial approximation we perform). If we do this for every partition we get a matrix whose size is $3 \times m \times n$ where m is the polynomial order of the 1st approximation and n is that of the 2nd approximation accordingly. Now we can see the potential shrink of the initial matrix if we choose the right values for the approximations.

4.4 Reconstruction of the function from its coefficients

In the previous chapter we described the method to minimize the APP estimator function into a coefficient table smaller than the initial one. Here we are going to describe the way that every symbol will get its responding APP estimator value from the reduced coefficient table. As we already mentioned there are two approximation stages in our algorithm.

The first one gives $m+1$ (where m is the order of the 1st approximation) coefficients for every SNR value, for every function partition.

The second one approximates the plots like figure 4.4 above (for every partition) and produces another set of coefficients $n+1$ (where n is the order of the 2nd approximation). So the final matrix will be of length *matrix size* where

m =first approximation polynomial's order,
n =second approximation polynomial's order
and *matrix size*= $3(o_1+1) \times (o_2+1)$

and the function to evaluate the APP value will be derived from the following equation

$$(c_{(m+1)P(n+1)}(SNR)^{(m+1)} + c_{(m)P(n+1)}(SNR)^{(m)} + \ldots + c_{0P(n+1)}(SNR)^0)mag^n$$
$$+ (c_{(m+1)Pn}(SNR)^{(m+1)} + c_{(m)Pn}(SNR)^{(m)} + \ldots + c_{0Pn}(SNR)^0)mag^{n-1} + \ldots +$$
$$(c_{(m+1)P0}\dot{(}SNR)^{(m+1)} + c_{(m)P0}(SNR)^{(m)} + \ldots + c_{0P0}(SNR)^0)mag^0 \quad (4.1)$$

where mag is the symbol's magnitude value
,SNR is the signal's SNR value
and P is the part of the function the current symbol is at.

As seen at the equation 4.1 above the APP estimator value is being computed by replacing the values of SNR and magnitude in the polynomials. At first, though, we must decide on which part (A, B or C) of the function is our symbol lying. The magnitude of the incoming symbol shows us the partition and the matrix gives us the respective coefficients for that partition.

4.5 Choosing the breaking points for approximation

The first choice we have to make even before starting the approximation and the evaluation of the results is to decide how many and where are the breaking points of the function. As it was clear from the start we choose to separate the function in three partitions (fig 4.2). That means we have to deal with three breaking points. They should be located where the functions exhibits its discontinuities. By examining the figure we conclude that the first breaking point should be around $(719 \ldots 723)$ whereas the second one should be located at circa $(1169 \ldots 1175)$.

We could describe beforehand the possible behavior of the coefficients in order to recognize the fitting breaking points as soon as we see the coefficients matching our forecast.

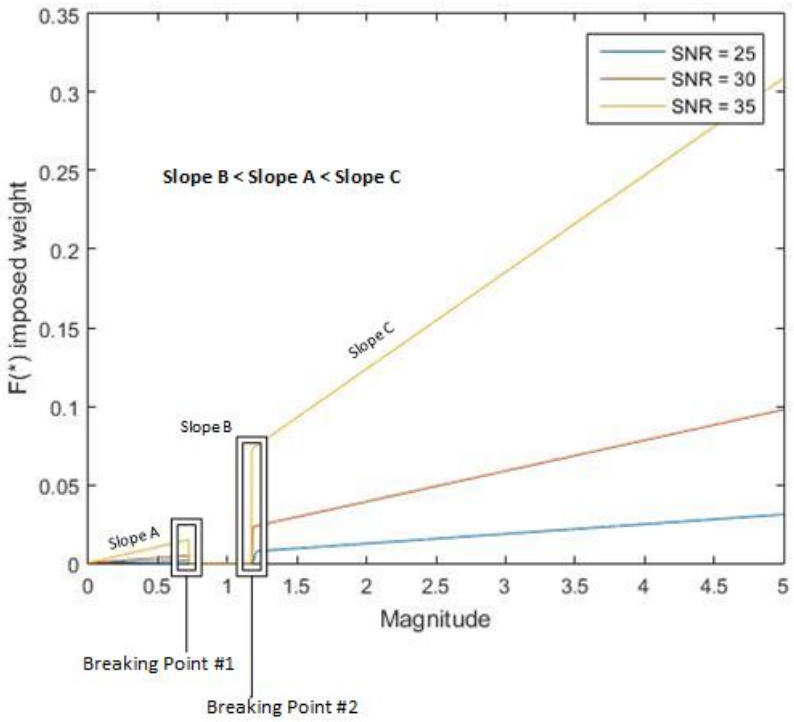


Figure 4.5: In the plot we can see the breaking point areas. Also the part C seems to have a negative β coefficient. We can also see the partition slope's ordering in respect to their values.

Watching the plot above we can deduce that the first order coefficients of all three parts should be steadily increasing curves with that of the C part being the biggest one. The B part curve should be almost zero at all points.

On the other hand the zero order coefficients exhibit different behavior. That of the B part should also be near zero at all points. A part should have a small positive coefficient whereas the one of the C part will probably have a small negative coefficient as we can see from the figure 4.5. As stated before we will approximate each part as a 1st order polynomial which means that we will have 2 coefficients for each line (i.e $\alpha\chi + \beta$).

4.6 First stage approximation

Having approximated the APP function partitions we present you with three possible breaking point choices.

Breaking points	
Breaking point 1	Breaking point 2
720	1173
722	1171
723	1170

Table 4.1: Possible breaking points of the APP estimator function.

4.6.1 Optimal breaking point choice

After the approximation we noted that the 1st order coefficients exhibited similarity in all 3 cases. Their behavior was as depicted in figure 4.6

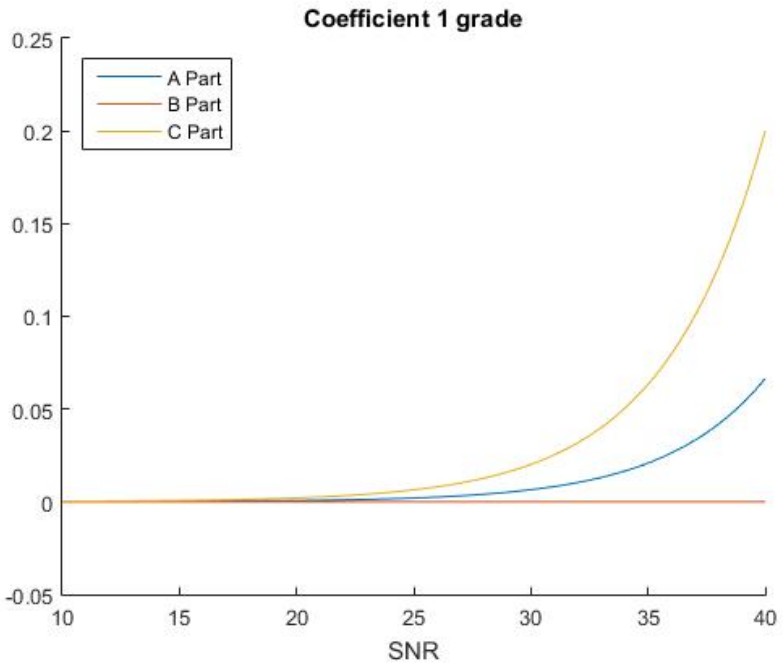


Figure 4.6: The behavior of all 1st order coefficients in all three breaking point couples seemed both identical and close to the behavior predicted.

Due to the similarity of the 1st order coefficients the choice regarding the optimal breaking point will be made after examining the 0th order coefficients. Those vary greatly between the 3 breaking point choices.

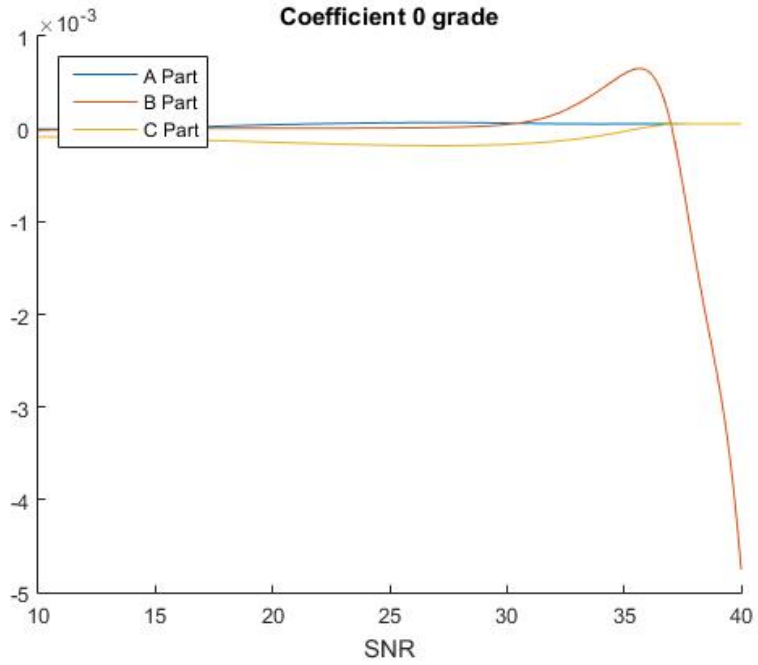


Figure 4.7: Behavior of 0th order coefficient on the 1st breaking point couple.

The zero order coefficients seem different from the ones expected. Specifically the B part one exhibits a large drop in its value which is probably due to inappropriate breaking points.

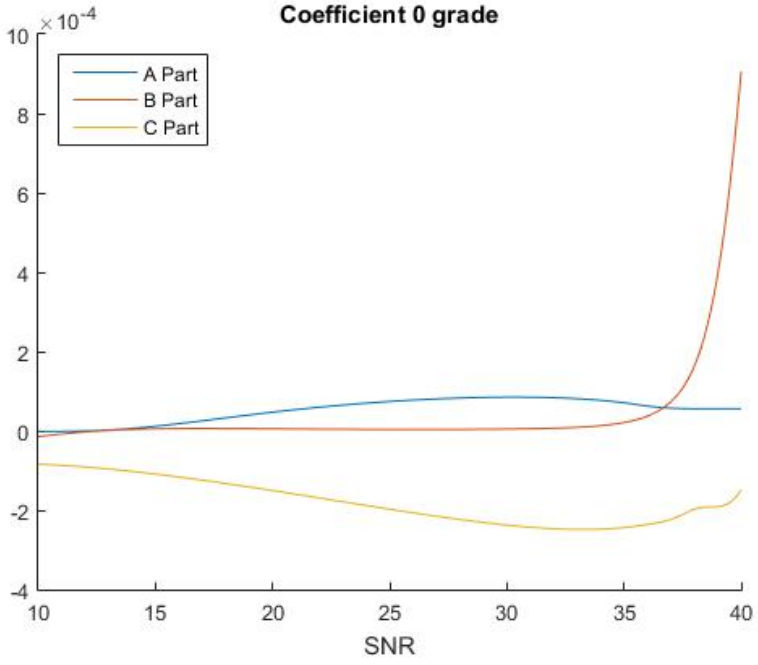


Figure 4.8: Behavior of 0th order coefficient on the 2nd breaking point couple.

Even in this breaking point couple the zero order coefficient of the B part exhibits a large positive slope on its last points. That is a clear indication that we have to decrease the second

breaking point by some points.

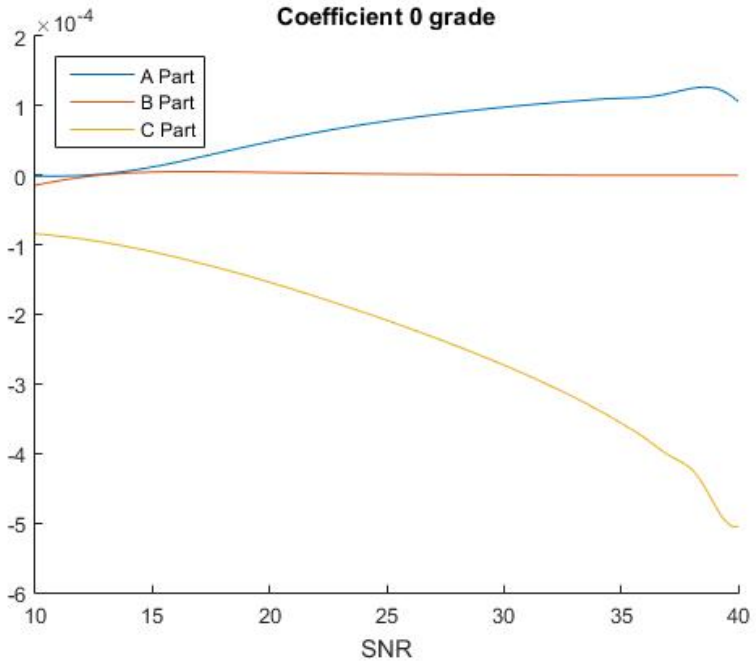


Figure 4.9: Behavior of 0th order coefficient on the 3rd breaking point couple.

Lastly, the combination of $723-1170$ seems to be the closest one to our forecast. The difference is at the zero order coefficients, since the first order one more or less stayed the same for all three cases. The zero ones, though, behave almost like we previously discussed. All three are straight lines, each of them increasing or decreasing according to its respective part at the estimator function.

4.7 Second stage approximation

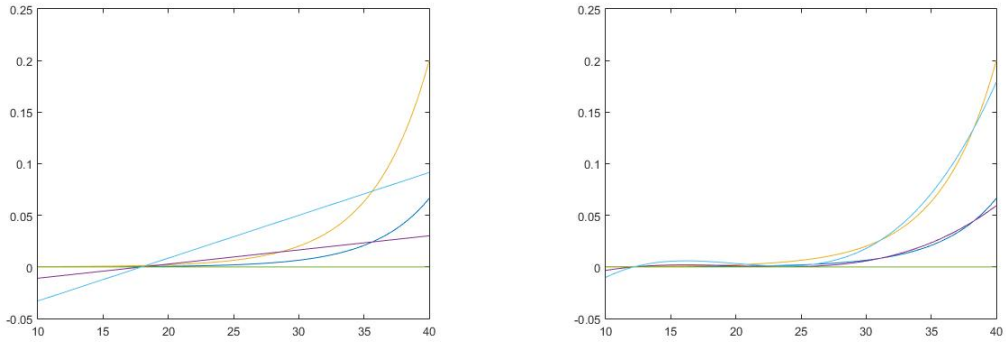
After having decided upon the order and the breaking points of the first stage we can go on with the second stage approximation as we have analyzed in the beginning of this chapter.

At this point the only decision that has to be made is the order of the second polynomial approximation. From the form of the curves we conclude that a second order approximation will produce a sufficient fitting goodness.

A comparison of this fitting will be made in order to choose the most appropriate one. The efficiency of a hardware implementation lies upon not only the efficiency of the approximation but also its potential complexity when translated into hardware.

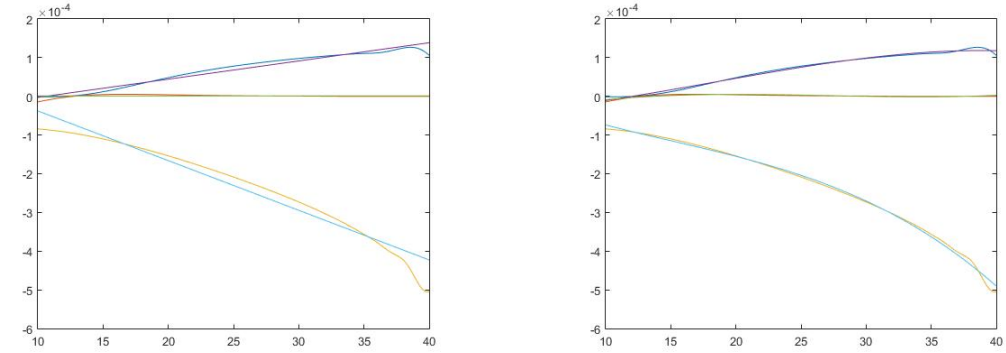
4.8 Goodness of fit

As expected, the bigger the approximation order, the better the goodness if fitting. We present some results with the figures below.



(a) Polynomial order = 1 (b) Polynomial order = 3

Figure 4.10: The 1st order coefficients approximated by 1st and 3rd order polynomials respectively.

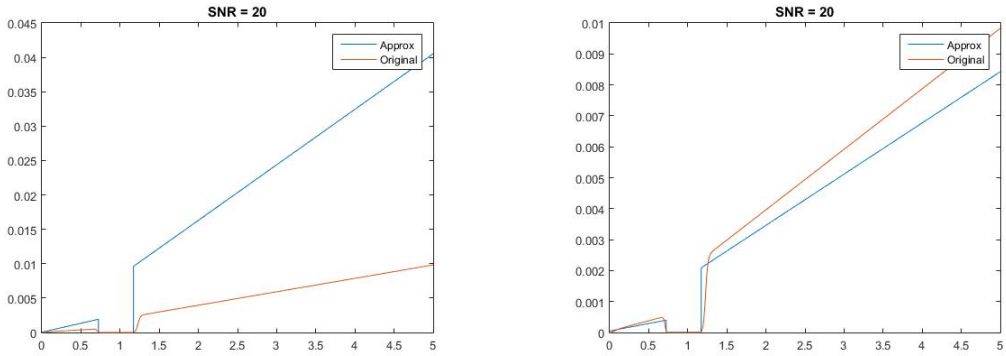


(a) Polynomial order = 1 (b) Polynomial order = 3

Figure 4.11: The 0th order coefficients approximated by 1st and 3rd order polynomials respectively.

As we can see from the figures (4.11a, 4.10a, 4.11b and 4.10b) above the zero order coefficients, since they are partially straight, are approximated successfully even by the 1st order polynomial. On the other hand the 1st order, since it is a curved line, needs a bigger order polynomial like the 3rd.

The actual estimator functions (approximated and original) are presented below (4.12a and 4.12b) for polynomial orders (1, 5).



(a) Polynomial order = 1

(b) Polynomial order = 5

Figure 4.12: The actual APP estimator function plotted alongside its approximation for polynomial orders 1 and 5 respectively.

4.9 Original vs. Approximated

Though the goodness of fit maybe a good indicator of how well approximated is the estimator function, the final and most important metric should be the actual BER difference between the approximated NLS estimator and the original function. The next plots are going to present this difference in order to end up with a choise about the approximation order.

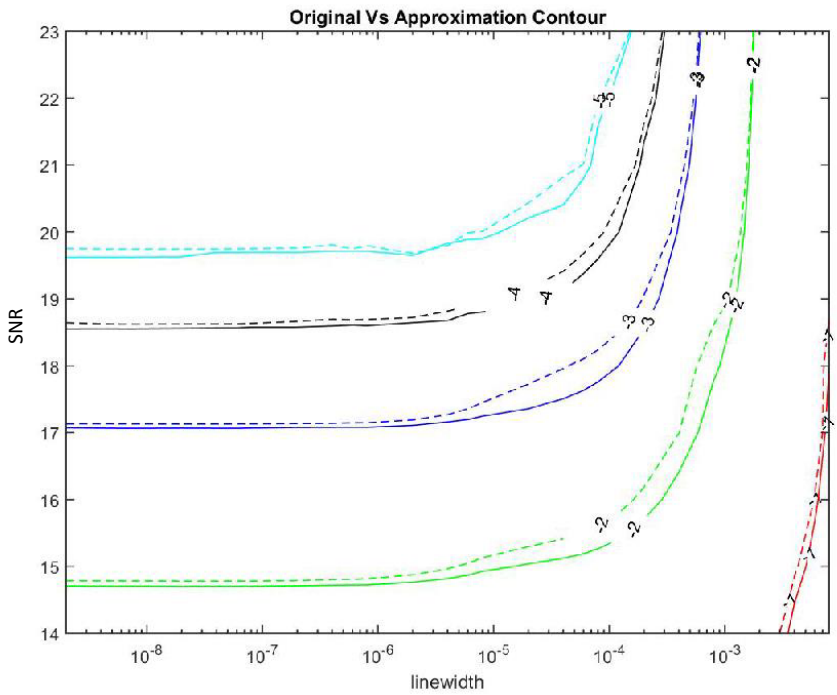


Figure 4.13: The contour plot of Carrier phase recovery using the original APP estimator and its approximation. (Approximated with 1st order polynomial)

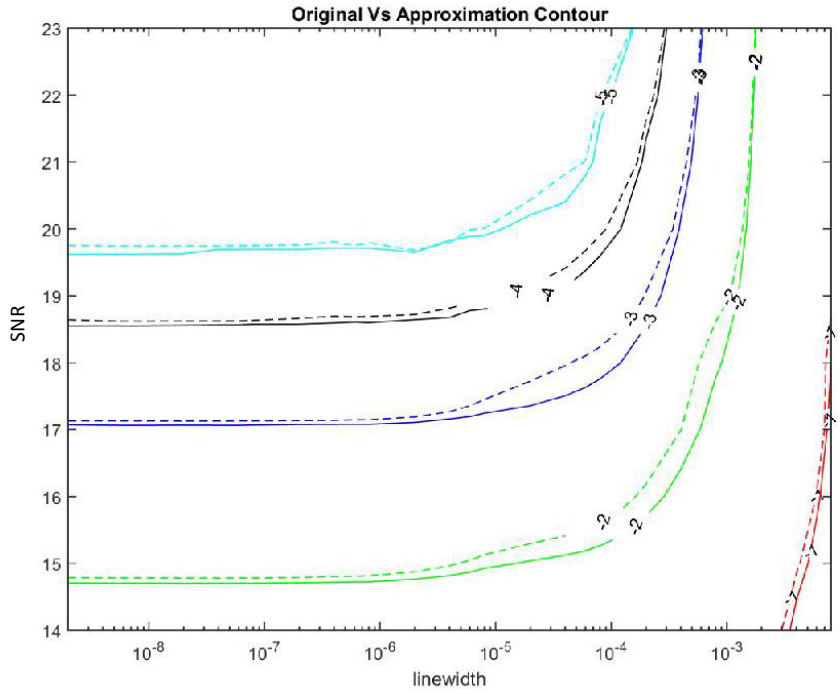


Figure 4.14: The contour plot of Carrier phase recovery using the original APP estimator and its approximation (Approximated with 9th order polynomial)

From the plots above (figures 4.13 and 4.14) we see that even though the 1st and 9th order approximation has a big gap in terms of goodness if fit, they exhibit the almost the same BER behavior. They both have values really similar to the original function.

4.10 Conclusion

To sum up, we conclude that the NSL estimator can be successfully approximated while decreasing the overall complexity of the system. That is possible with a 2-stage approximation.

At the first stage we partition the function in 3 parts ans we approximate each part with a straight line (i.e 2 coefficients). Putting together those coefficients for every partition we come up with some curves again. Then we repeat a polynomial approximation and the coefficients we are left with populate the final matrix.

From the simulation we saw that even with a 1st order approximation of the second stage we get good results that are really close to the original non-approximated function.

The reason that even the 1st order approximation yields good results, despite its non optimal goodness of fit relies on the next factors.

1. The breaking points were chosen accurately and the curves of the 0th order coefficients exhibit a linear behavior.

2. The 0th order coefficients are approximated good even by a 1st order polynomial due to their linearity

3. It seems that since the 1st order coefficients steadily grow as the SNR grows then the NLS estimator produces good results.

4. Getting good results is more closely correlated, up to a point, with having the right form of the weighting function than with actually being extremely close to the original NLS estimator function.

Chapter 5

FPGA System Design

5.1 Introduction

Up until now, we have theoretically studied and derived the optimal characteristics of a Carrier Phase Recovery pipeline operating with the NLS phase estimator algorithm. This system was implemented in VHDL language with the use of Vivado 2016.2 tool of Xilinx. In the next pages we are going to analyze the implementation of each hardware module that our system constitutes of.

5.2 Pipeline Overview

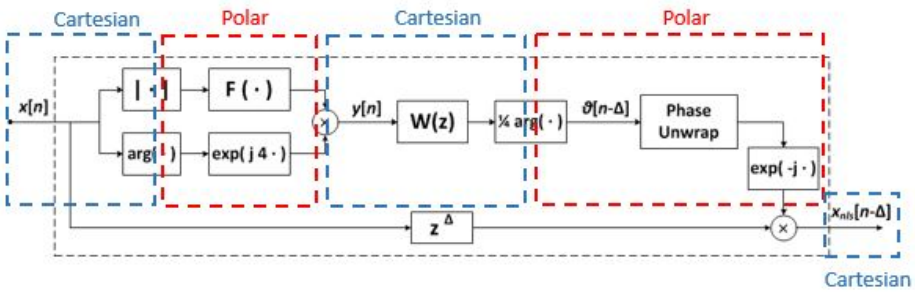


Figure 5.1: A higher level of abstraction overview of the Carrier Phase Recovery pipeline.

The figure 5.1 above depicts an overview of the system implemented in the FPGA architecture. Each of these blocks was either further analyzed to hardware sub-modules or merged with other modules to finally form another alternative view of the system from the hardware implementation perspective.

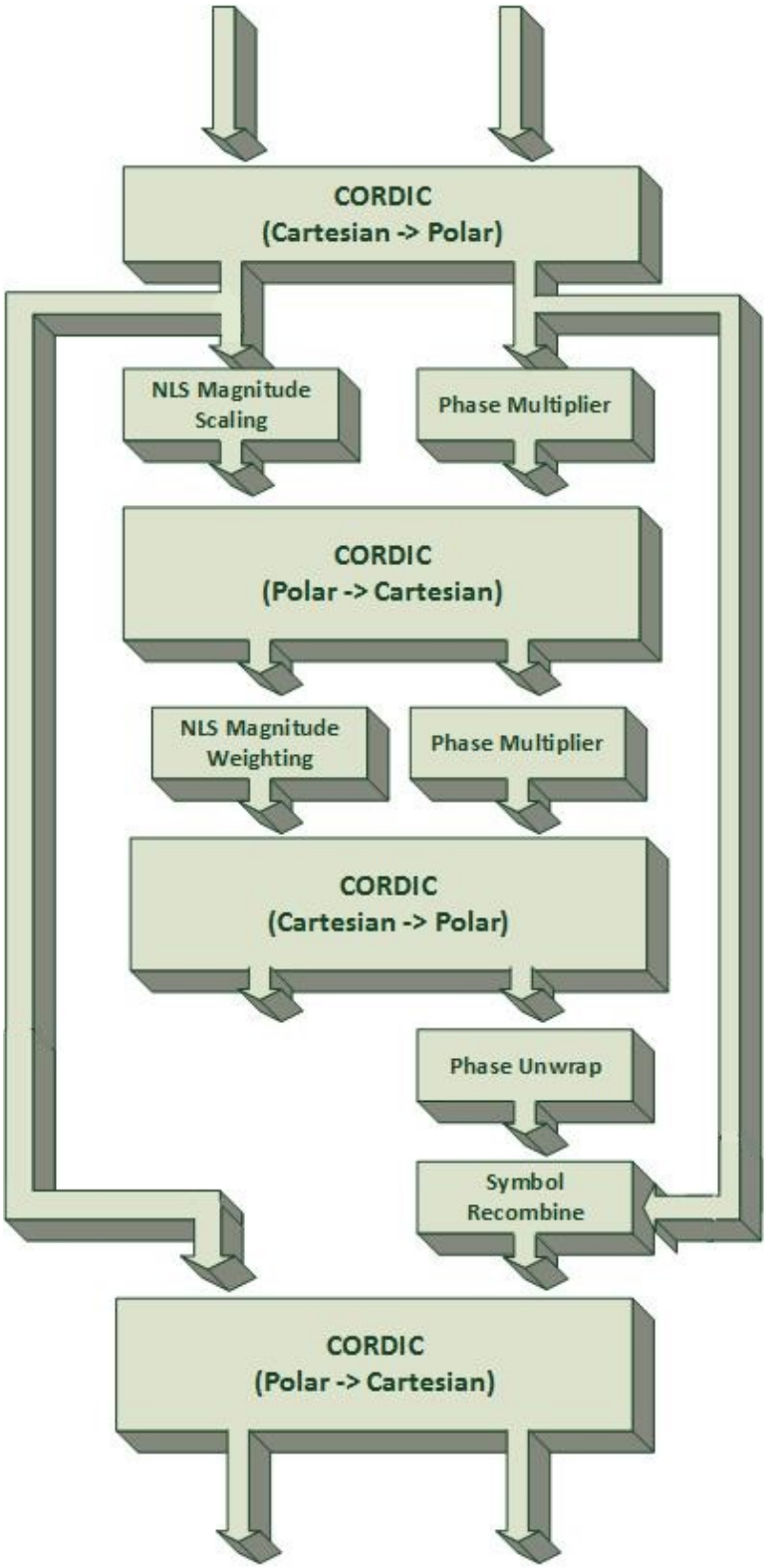


Figure 5.2: A higher level of abstraction overview of the Carrier Phase Recovery pipeline.

The schematic diagram 5.2 above depicts the NLS carrier phase recovery pipeline at a hardware

module level. It is the structural model of our VHDL architecture. Each of the blocks will be separately analyzed and discussed in the following sections.

5.3 Hardware Modules

5.3.1 Valid output and System reset Control module

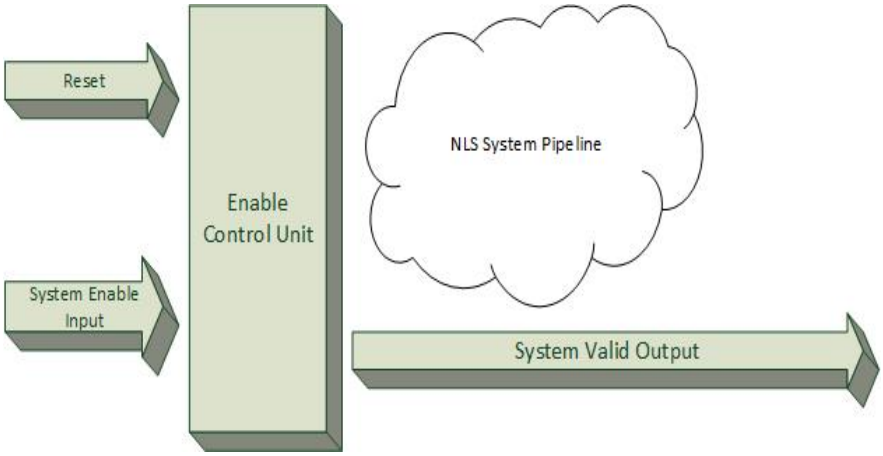


Figure 5.3: The module that is responsible for signaling the output for valid signal according to the input.

In order to avoid unnecessary delays in our circuit and since the majority of the hardware modules are purely computational and thus do not need any specific enable or reset logic we provide the valid signal to the output bypassing the main pipeline. A shift register with latency equal to the pipeline latency shifts the enable signal to the system output. In case the module receives a reset signal it immediately clears the pipeline of any remaining signal making the data present in the pipeline useless since they will not be accompanied by a valid output signal. This way the enable signal has to travel only through a shift register and not to every module.

5.3.2 Coordinate conversion module

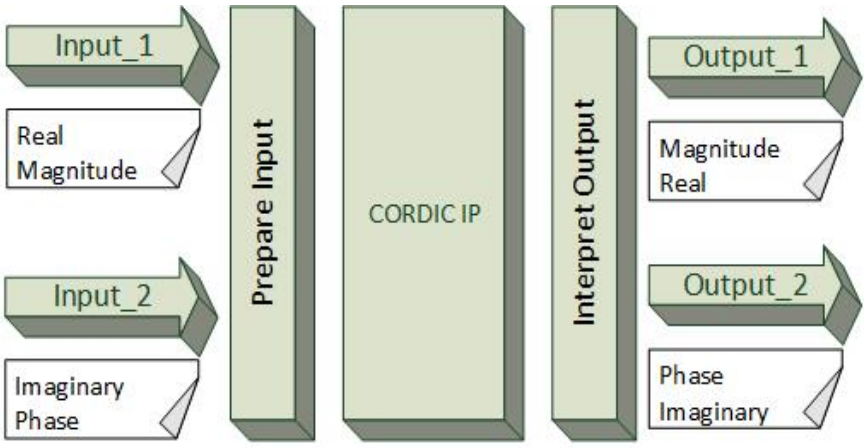


Figure 5.4: The CORDIC IP (cartesian to polar) and some mandatory modules for the input format and output translation.

As we stated before the conversion between Cartesian and Polar coordinates in our system is realized via the CORDIC algorithm. CORDIC stands for (Coordinate Rotational Digital Computer) and it is essentially an effective algorithm to implement trigonometric functions with limited computational complexity, since all the trigonometric numbers are compute with the use of Look-up tables. The specific component is imported as a ready IP in out system and is parametrized via the system generator GUI of Xilinx Vivado.

The prepare input module is important to form the input operands in the form acceptable by the CORDIC IP.

Lastly the interpret output module helps us in the separation of the output word of CORDIC into the two components (here phase and magnitude).

The same layout of modules is further used in every conversion of Polar to Cartesian and vice versa. The only difference is that the IP used for each conversion is different and requires different parametrization from the user (figure 5.5).

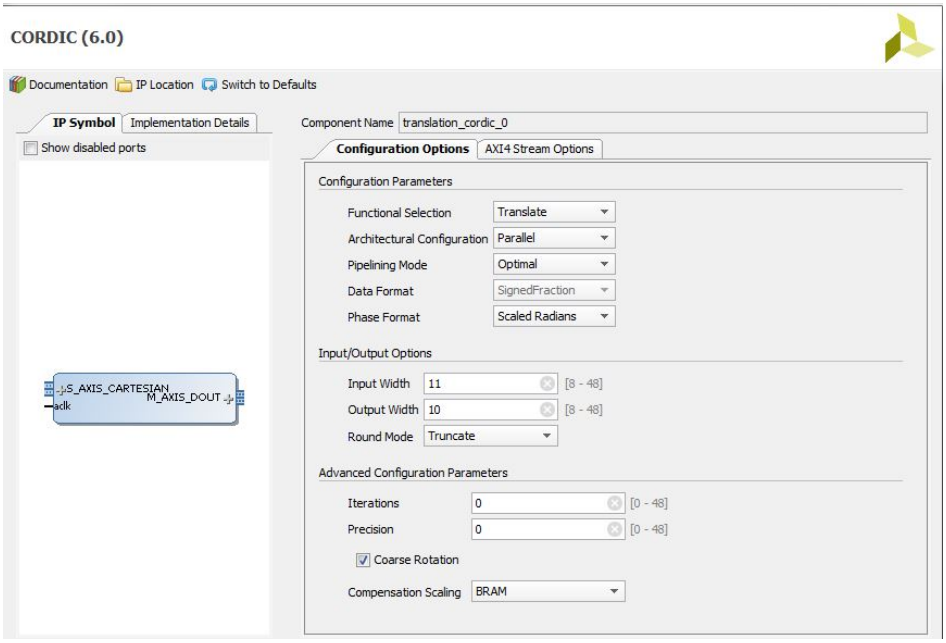


Figure 5.5: The CORDIC IP system generator GUI. All the parameters get defined from here.

An important note is that our whole design gets rid of the need to do computations with the π quantity. That happens since the CORDIC gives us the option to get all the results in scaled format. That means that a phase of 2.7π would be represented as 2.7.

5.3.3 NLS Magnitude Scaling

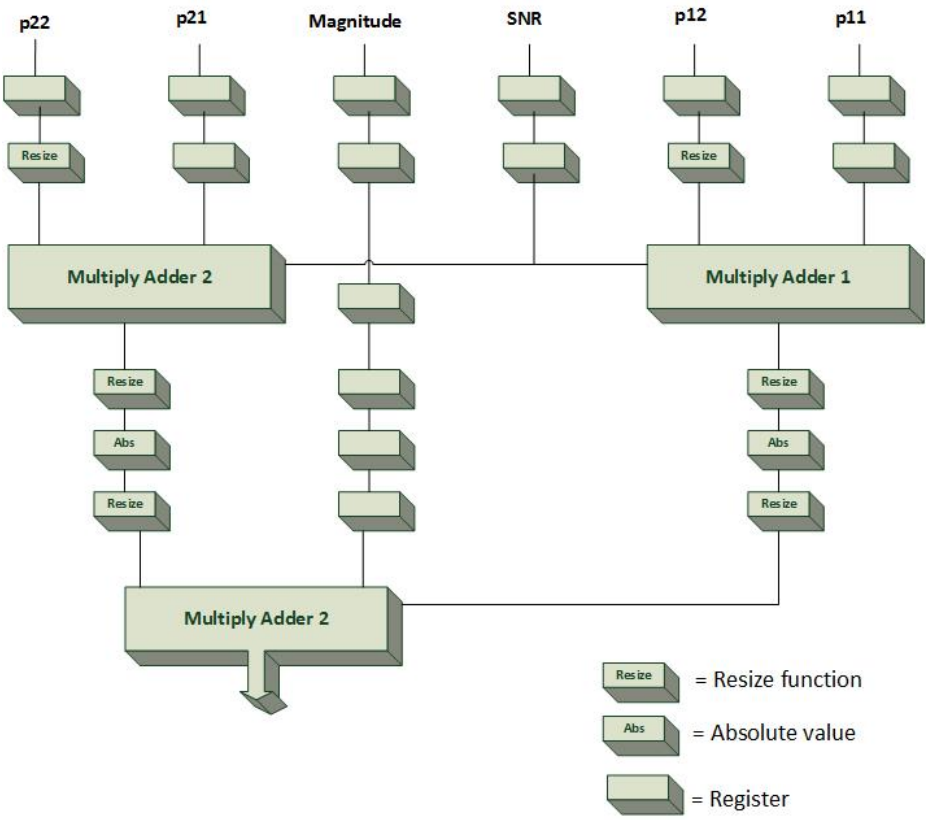


Figure 5.6: The magnitude scaling module. This module implements the magnitude scaling according to the QPSK partition.

As described on chapter 3, this module implements the function responsible for the "weighting" of the symbol magnitude. The function is described below.

$$(p_{11} * SNR + p_{12}) * Mag + (p_{22} + p_{22}) \tag{5.1}$$

5.3.4 Phase Multiplier

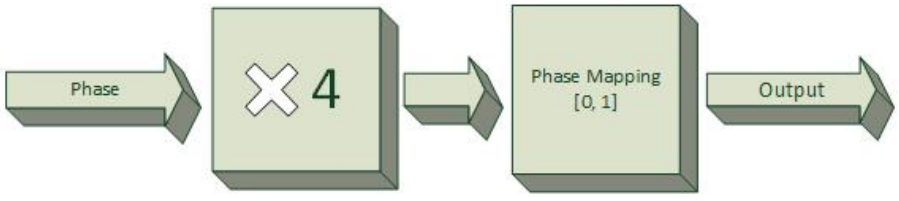


Figure 5.7: The phase multiplication module. After the multiplication the phase must be scaled down to the boundaries dictated by the CORDIC module.

The mandatory multiplication by 4 happens by just left shifting the phase 2 places. Then, a mapping of the phase back to the [0, π] has to be done to assure the CORDIC's valid operation.

In fact since we work with scaled radians as a format the necessary mapping has to be done in the [0, 1] region.

The mapping follows the following algorithm.

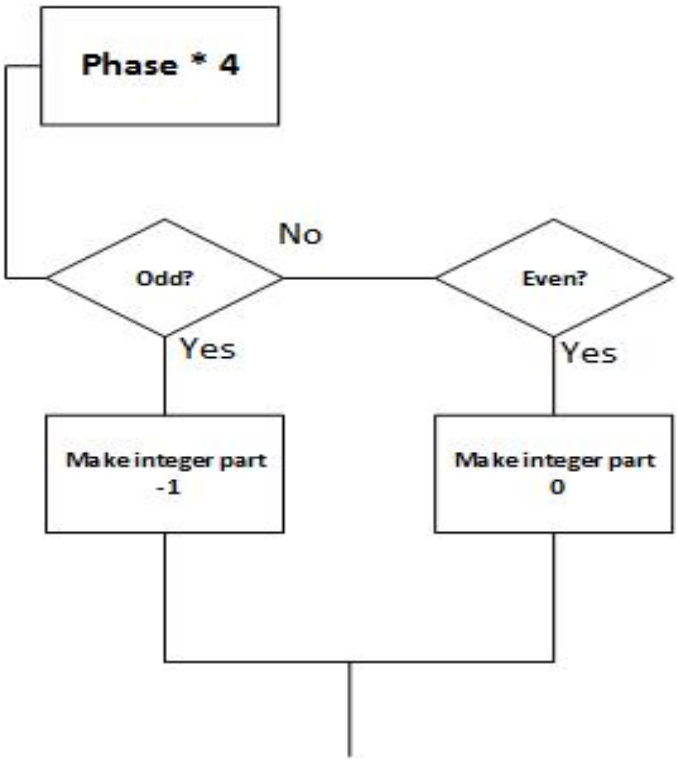


Figure 5.8: Flowchart algorithm making the phase mapping of the multiplied phase.

5.3.5 Symbol Filter

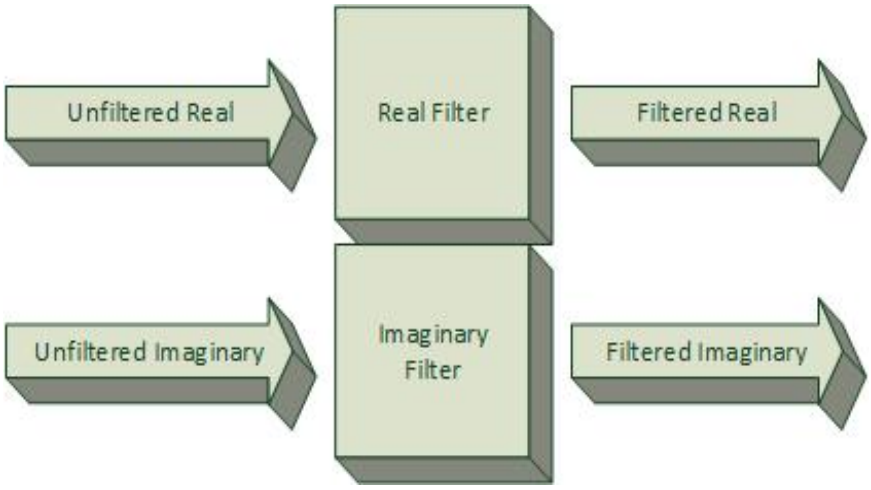


Figure 5.9: The symbol filtering schematic. We need a module both for the real and the imaginary part.

The symbol's real and imaginary parts are passed through the moving average filter. The filter's purpose and functionality has been already discussed in the previous chapter 2.

5.3.6 Phase Unwrap

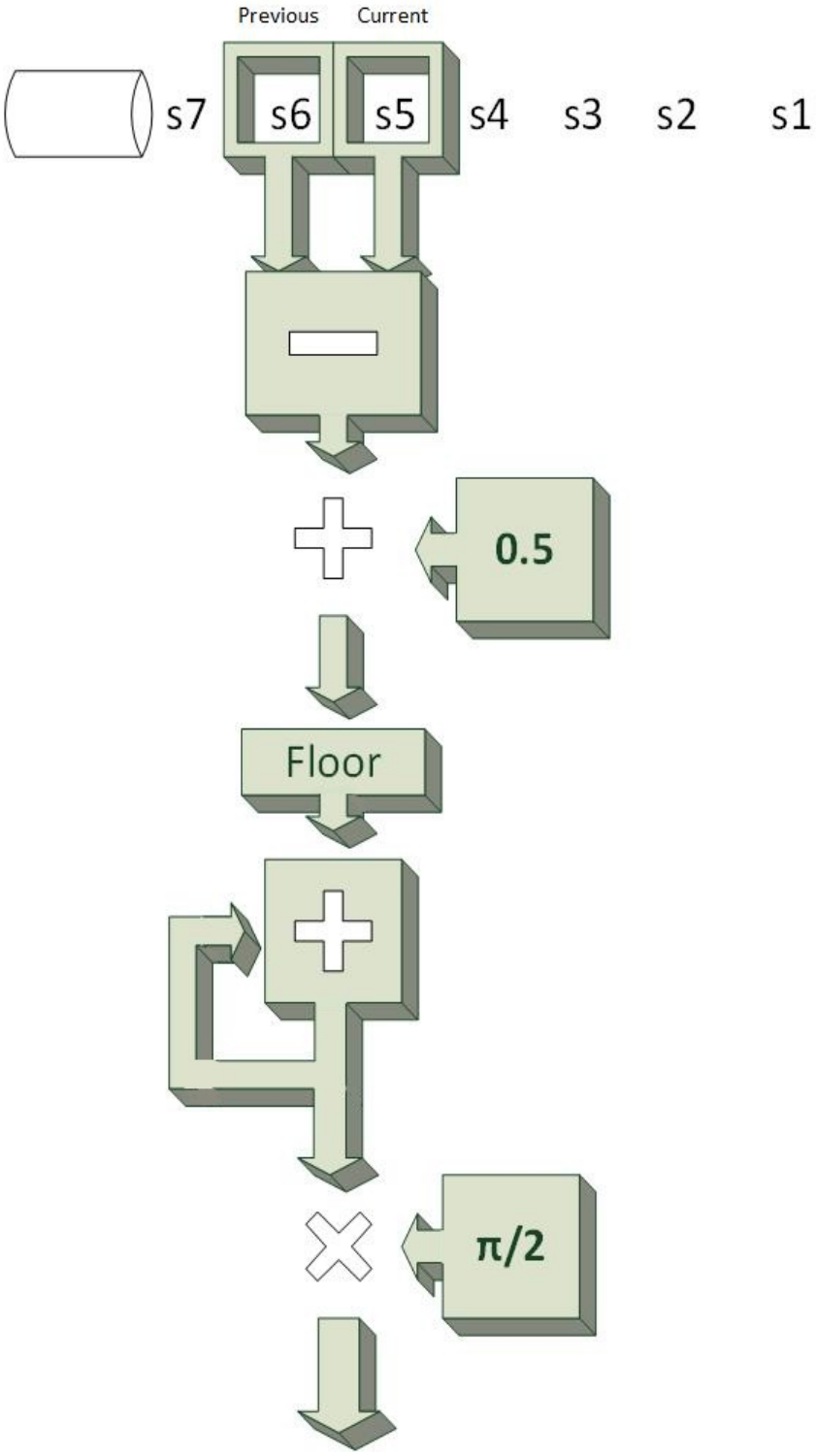


Figure 5.10: The phase unwrap module schematic. The flow as well as the operations happening on the data are clearly depicted.

The phase unwrap schematic above operates as you can see on an incoming stream of data. The function implemented is the one following

$$p = 0.5 + \frac{\pi}{2}\lfloor \Delta\phi[n-1] - \Delta\phi[n] \rfloor \tag{5.2}$$

5.4 Single Pipeline

The modules above, when connected successfully, implement the carrier phase recovery algorithm. As stated the whole system is synchronous and fully pipelined. That means that after the data has passed through the pipeline we get 1 output at every clock cycle. The aforementioned modules have latencies described by the following table 5.1.

Hardware Module	Latency Cycles
CORDIC 0	$n + 4$
NLS Magnitude Scaling	13
Phase Multiplier	13
CORDIC 1	$n + 4$
Symbol Filter	2
CORDIC 2	$n + 4$
Phase Unwrap	5
Phase Mapping	3 + (filter delay - 1)
Magnitude Shift Register	2

Table 5.1: Number of latency clock cycles of the various modules present in our design. The n in the CORDIC modules means the number of latency cycles that is reported by the system generator tool and depends on the number of bits it internally processes.

5.5 Pipeline parallelization

Though the single pipeline functionality might be acceptable, it surely lacks the optimal operating frequency. With a single pipeline, no matter how "fast" our design is we cannot obtain optimal results, since the telecommunication field requires very high operating frequencies. Even the state-of-the-art FPGA designs hardly reach 700 or 800 MHz which is an unacceptable frequency for high-speed fiber-optic systems.

The answer lies in the massive parallelism we can achieve through an FPGA architecture. With a precise enough algorithm, that is also efficiently paralleled we can achieve frequencies orders of magnitude greater than a few MHz.

5.5.1 Parallel Architecture Design

Most of the modules present in our pipeline can operate on data regardless of the knowledge of any other preceding or following data. In a parallel implementation those modules get just copied across the parallel branches and pose no other challenges.

Other modules though, like the phase unwrap, pose some difficulties in the process of embedding them in a parallel system. The latter for example requires knowledge of all the other branches

results. The need of inter-branch communication arises and we realize that such modules require specific custom design in order to be used in a parallel structure.

In the following schematics we show the architecture of the parallel implementation of the phase unwrap and the symbol filter modules.

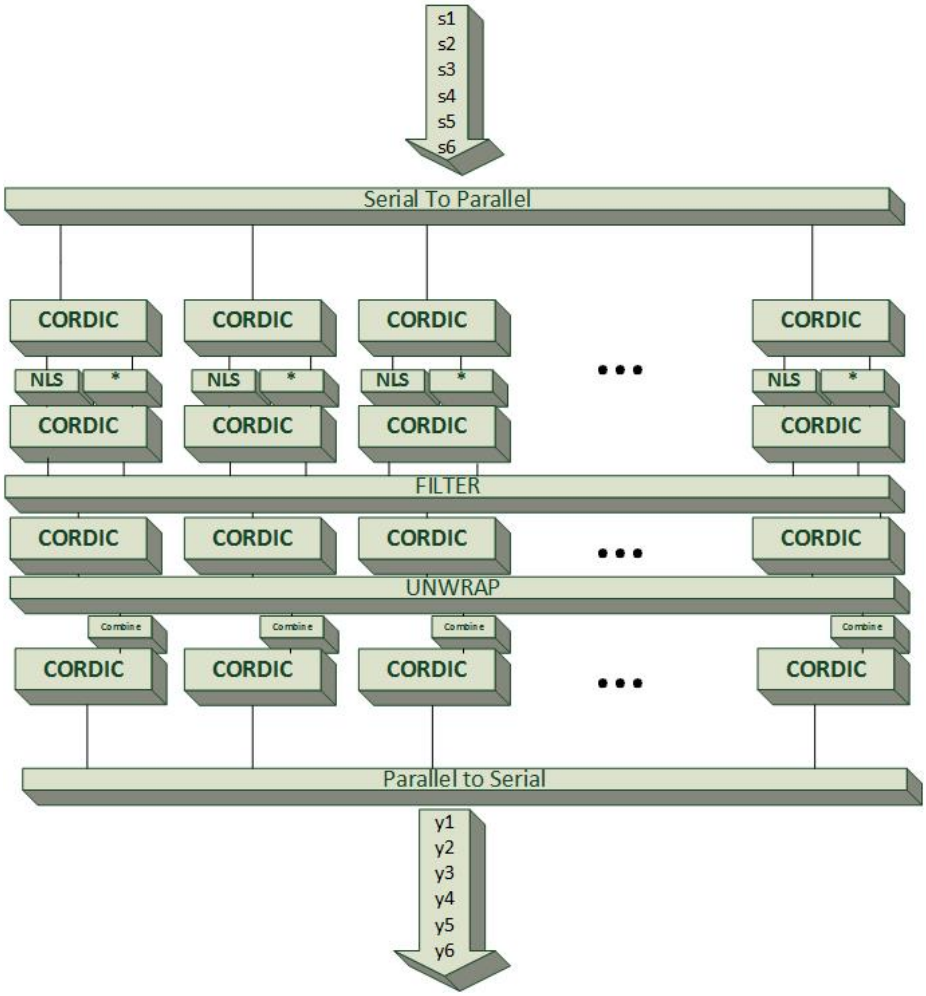


Figure 5.11: The overview of the architecture of the parallel pipeline. We can see that the modules of the filter and the phase unwrap have a unifying parallel architecture unlike the others which are just multiplied by the parallelization order we want to achieve.

5.5.2 Phase Unwrap Parallel Architecture

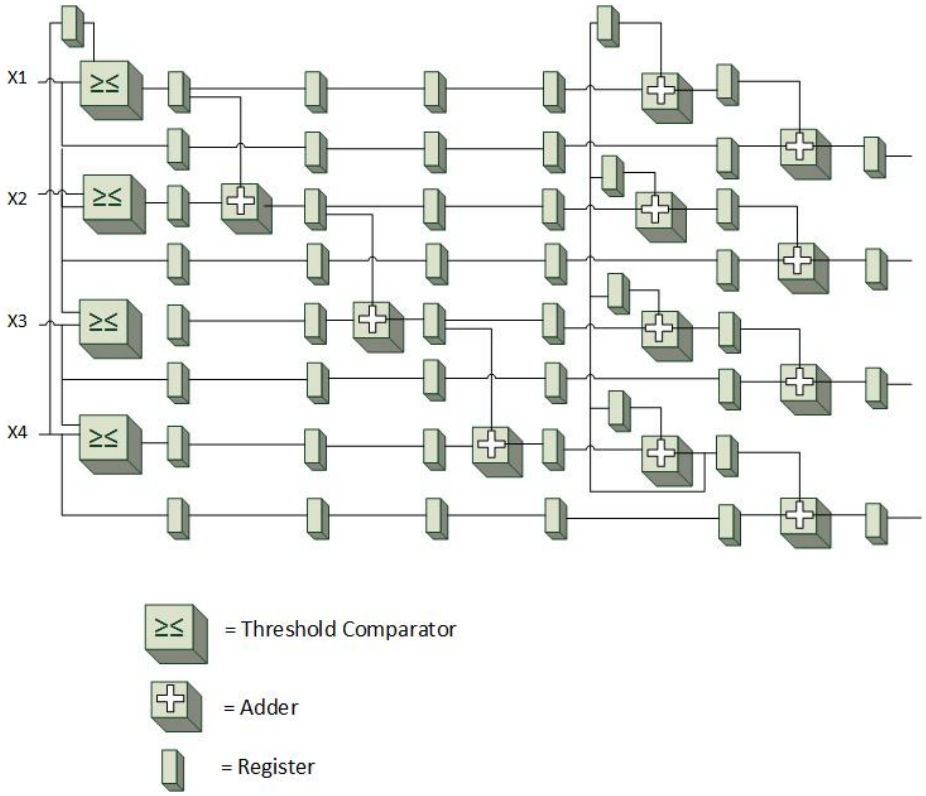


Figure 5.12: The parallel architecture of phase unwrap module. Here the parallelization factor is $p = 4$

In the schematic 5.12 above we see the proposed parallel architecture of the phase unwrap module. It can support any order of parallelization.

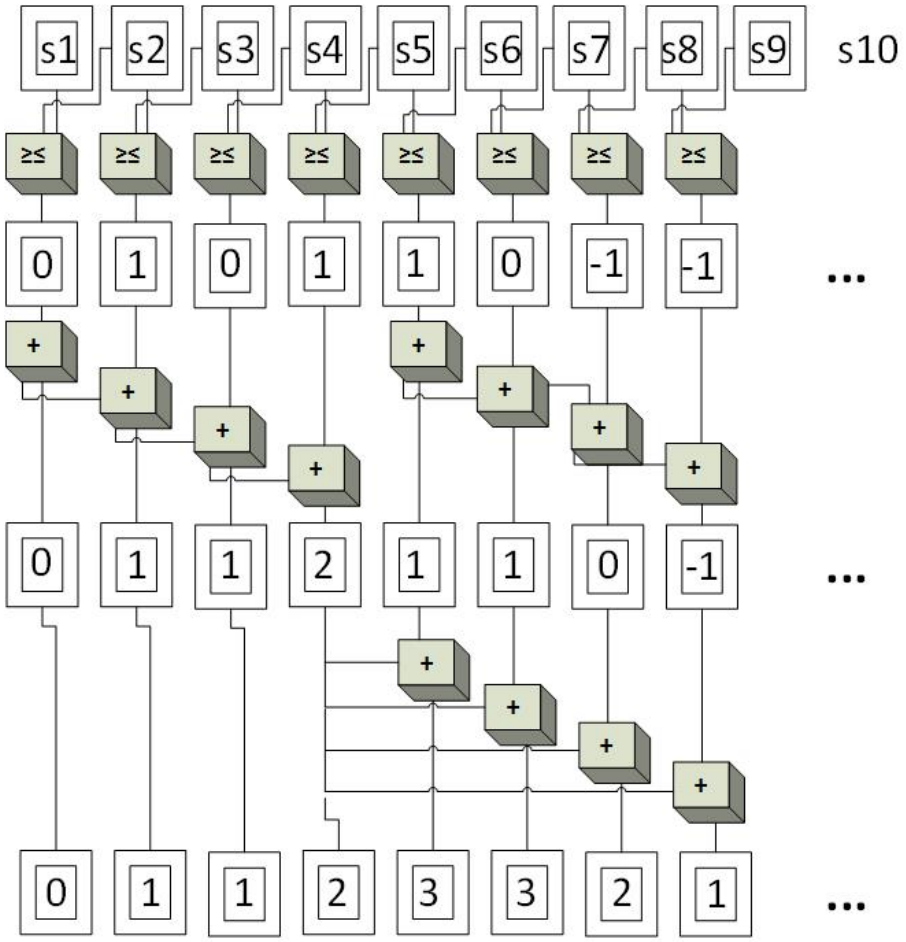


Figure 5.13: The operating principle of the parallel architecture of 5.12. Here the parallelization factor is $p = 4$

From the figure 5.13 we can see the data flow in the parallel architecture. At first the threshold comparator outputs the difference between the previous and the current symbol. This can take 3 values $-1, 0, 1$. After we have to compute the inner sum for every n symbols, where n is the parallelization order. At last the final iner sum of every block of inner sums has to be fed to the next block. That is crucial in order to keep the information about previous sums. The final output is summed again with the initial symbol and the unwrapped phase is produced.

5.5.3 Symbol Filter Parallel Architecture

The design of the symbol filter may seem rather simple at first but it contains some tricky parts. The challenging part is that a filter may have 2 degrees of freedom. The filter length, as well as the parallelization order. This fact makes the architecture design harder to implement since the design should be dynamic reconfigured in both axis. Both the y one (parallelization order) and the x one (filter length).

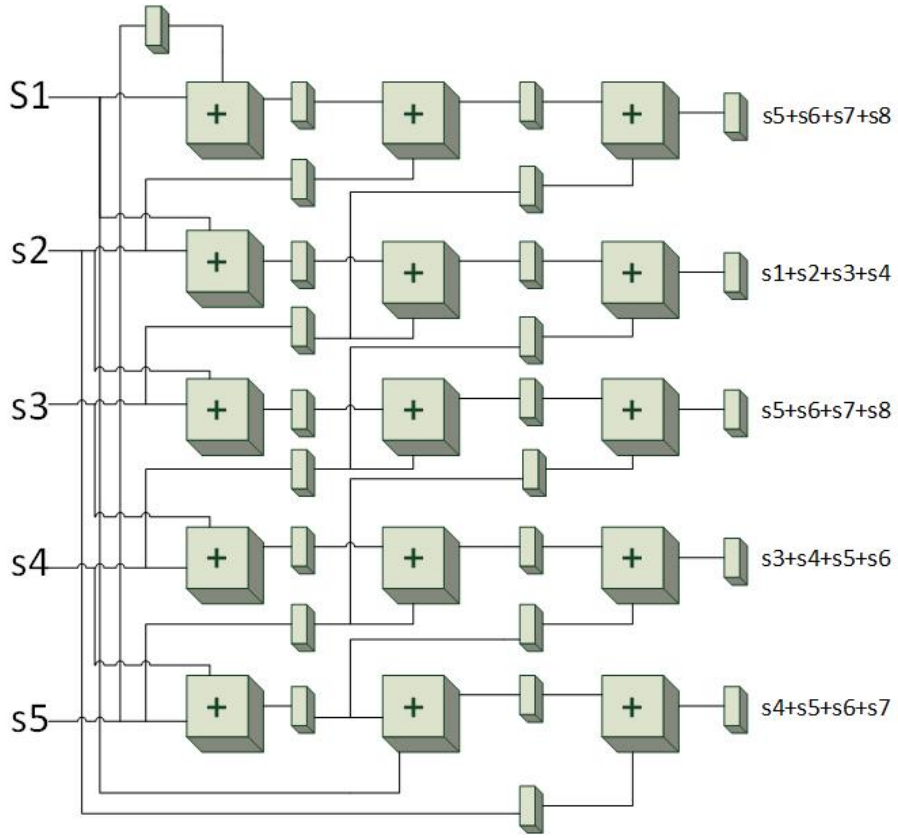


Figure 5.14: The parallel architecture of moving average filter. The parallelization order is $p = 5$ and the filter length is $n = 4$

In the figure 5.14 above we depicted the architecture of a moving average filter for a parallelization order of $p = 5$ and a filter length of $n = 4$. Those 2 parameters can change their value according to our needs and this gives us a great advantage on tailoring the design to perfectly fit our needs. As we have noted before though, the filter length parameter will only take values which are power of 2 since the optimal values found are close to powers of 2 and the performance gap is not that big. The gain from such an approach is that we do not use any hardware module for the divide operation (which is a great advantage) and therefore we save on valuable hardware resources and coding complexity.

5.6 Design Verification

A very important part of every hardware design stage is the design verification. Fortunately VHDL provides us with the ease of creating fully custom testbenches to test our design's functionality.

In order to accelerate the process we have created a verification workflow that utilizes the Matlab environent in order to create inputs and asses the outputs of the Vivado testbench that is simulating the actual hardware. That way we could compare the actual results derived both from theoretical simulations on Matlab and from behavioral simulation on the hardware.

5.6.1 Gluing together VHDL and Matlab

In order to obtain the BER result of the VHDL code we had to combine the operation of VHDL and Matlab. The workflow was as it follows.

5.6.2 Feeding VHDL with input symbols

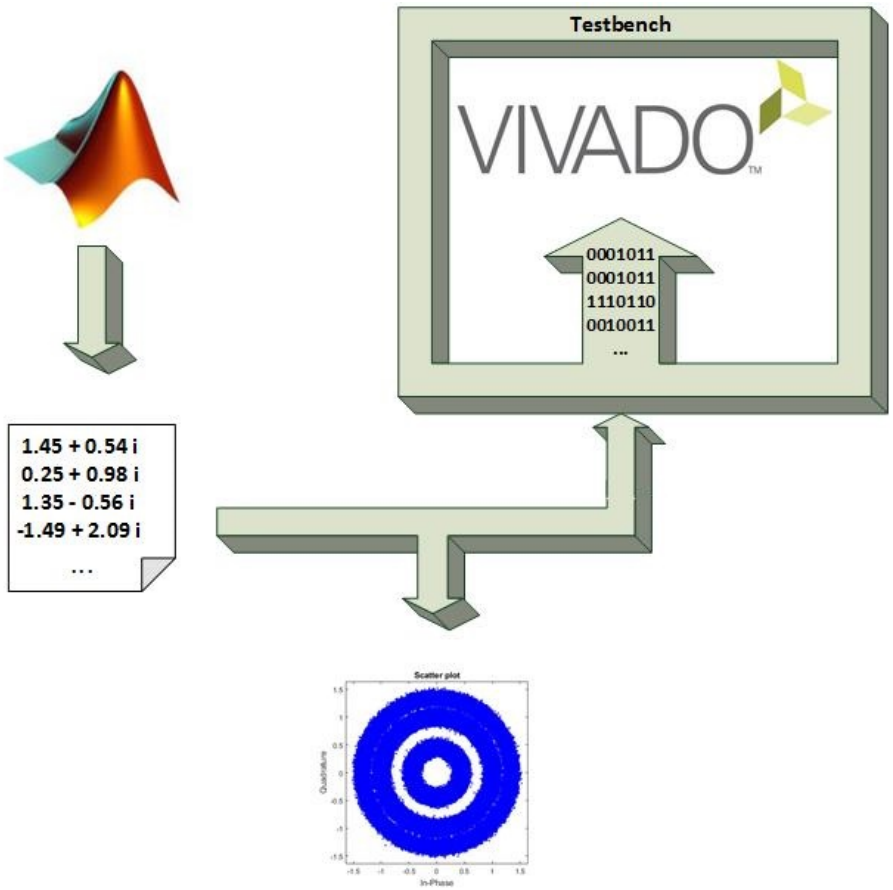


Figure 5.15: The matlab noisy symbols feed the VHDL through files read by the testbench. Then the latter goes on computing the output symbols.

5.6.3 Interpreting VHDL output with Matlab scripts

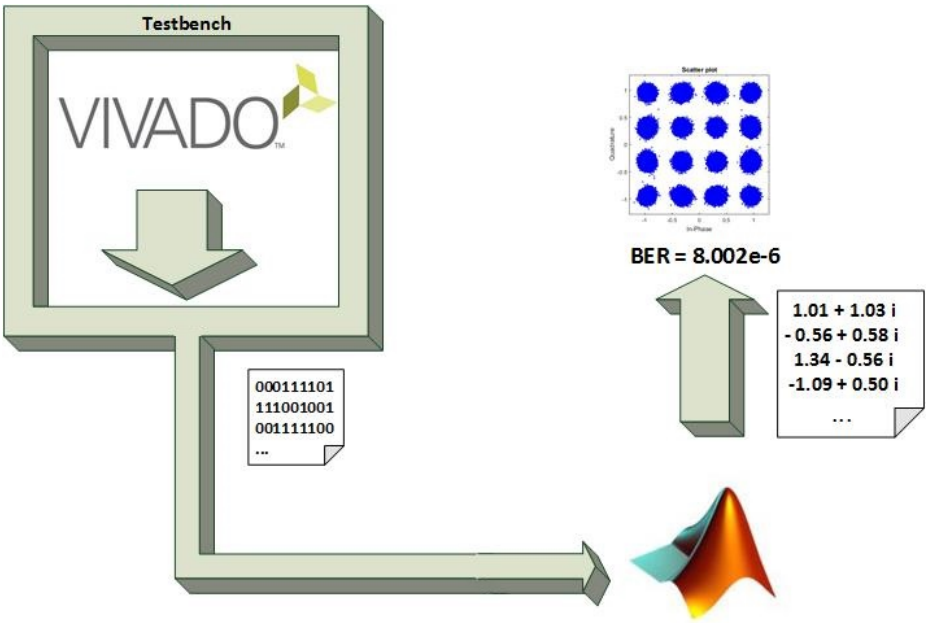


Figure 5.16: The output of the VHDL system is written to an output txt file. Then matlab scripts read that input and interpret it accordingly to produce BER results.

To sum up, we have designed ,implemented and verified the correct function of a fully-parametrized and fully-parallelized Carrier Phase Recovery hardware module. Its performance, its compliance with the modern telecommunication standards as well as its comparison with related work are going to be assessed in the next chapter.

Chapter 6

Experimental Results

This final chapter focuses in evaluating the FPGA implemented system regarding the performance and the accuracy that it is able to provide in a real telecommunication system.

6.1 Synthesis and Implementation

After deriving the pruning configuration sets we continued with the synthesis and implementation of the system in the FPGA platform. This process is crucial to evaluate the performance along with the precision that the system yields and the trade off between them. The target device is Virtex - 7 family FPGA board and the part number is xc7vh580thcg1155-2G which has an underlying technology of 28nm. The speed grade of the specific board is among the fastest in the Virtex - 7 family. It has to be noted that the results we will present here correspond to this specific device and therefore any experiments attempted on better or worse boards can possibly yield better or worse performance accordingly.

The synthesis and implementation work flow was mostly a trial and error procedure. After each run we evaluated the resulting operating frequency and applied corrections to further improve the results.

Our aim was mostly oriented at obtaining a high frequency design able to accommodate the high speeds that are mandatory for the fiber-optic networks.

The main correcting actions we had to take were limited at adding registers in the output of modules as well as in some critical paths in order to pass some timing barriers. After each correction, we had to check the functionality of the system again to ensure we are still getting the optimal results.

The implementation strategy was most of the times set at the "Performance Explore" or "Performance Extra TimingOpt" selections which produce elevated performance in the cost of more utilized FPGA fabric.

6.2 Performance Oriented Improvements

6.2.1 Dynamic Range Study

When dealing with a constrained precision system such as a fixed point FPGA implementation then we have to know the dynamic range of our algorithm. Dynamic range is in fact the extreme values that the algorithm is able to process and output. By being aware of these extreme states

we can correctly define a finite number of integer bits that the system needs in every stage of its pipeline. The following table has been populated after Matlab simulations that have been carrier out with two different input sets. One produced a very high AWGN noise while the other induced a big phase noise in the system. This way we test the extreme values in every state of our system. The first input set with high AWGN is this one:

- SNR $= 17.2dB$
- Linewidth $= 1 \times 10^{-6}$
- Filter Length $= 45$
- Symbols $= 1 \times 10^{6}$

The second set with high laser phase noise is this one:

- SNR $= 23dB$
- Linewidth $= 1 \times 10^{-3}$
- Filter Length $= 10$
- Symbols $= 1 \times 10^{6}$

The output of the system for both cases was BER$= 1 \times 10^{-3}$ which is the lowest possible value we consider for the system to operate. To sum up, by examining the worst case scenario with the lowest SNR possible we will get the extreme values that we have to consider at the system simulation. The table 6.1 below shows the results.

Pipeline Stage	Min Value	Max Value	Integer Bit Width
Input	≈ -1.7	≈ 1.7	2
CORDIC-1 (Magnitude)	≈ 1.8	-	2
CORDIC-1 (Phase)	1	-	3
NLS Magnitude Scaling	0	≈ 0.01	1
CORDIC-2 (Real)	≈ -0.01	≈ 0.01	1
CORDIC-2 (Imaginary)	≈ -0.01	≈ 0.01	1
Filter (Real)	≈ -0.053	≈ 0.0052	1
Filter (Imaginary)	≈ -0.049	≈ 0.005	1
Phase Unwrap (Internal Accumulator)	≈ -22	≈ 30	7
Phase Unwrap (Output)	≈ -11	≈ 15	6
Output	≈ -1.4	≈ 1.3	2

Table 6.1: Dynamic Range of the various processing stages of the algorithm and the required integer width.

The values for the integer bit widths above have been incorporated in out system. The only exception was that for the input and output widths we had to provide 3 bits in order for the CORDIC Xilinx IP to operate succesfully, since if provided with two input bits the might be unknown behavior at the output [18].

6.2.2 Register Pruning

In order to be able to dynamically diverge from a precision oriented system into a performance oriented one we have implemented a 8 stages of register pruning. This structure gives us the ability to truncate the registers present in the pipeline so as to minimize the hardware utilized by our design and therefore gain more performance. Of course this performance is gained in the expense of less precise outputs. Nevertheless, the ability to make compromises between accuracy and performance is able to drive a design in its ideal operating point as dictated by the context in which it is used (environment noise, laser linewidth level, desired accuracy etc).

Below we can see the pipeline module that every pruning stage was inserted at. These places are:

- The input registers of every CORDIC IP core.
- The output registers of every CORDIC IP core.
- The output registers of the NLS Magnitude Scaling module.
- The output registers of the moving average filter.
- The output registers of the phase unwrap module
- The output registers of the phase recombine module.

Each register, starting from the first modules of the pipeline and proceeding to the last ones, was set to a fixed value and the BER output was assessed after VHDL simulation with the help of MATLAB (as described in the previous chapter). Iterating through the whole pipeline we derived a variety of different pruning "sets" each one yielding different accuracy results while using different number of registers.

In the following table 6.2 we present some of the sets that were found to provide good performance while not using a big number of registers throughout the pipeline.

			Bits to Truncate						
ID	Input	Coefficient Width	CORDIC-1	NLS	CORDIC-2	Filter	CORDIC-3	Symbol Recombine	CORDIC-4
# 1	10	6	0	5	1	2	2	2	2
# 2	10	6	0	5	1	3	2	2	2
# 3	11	12	1	7	1	3	2	2	4
# 4	10	6	0	5	2	2	2	2	1
# 5	11	6	1	5	1	2	2	2	2

Table 6.2: Chosen sets of register widths that will be evaluated for their performance versus precision trade-off.

6.2.3 Filter's Role In Performance

While evaluating the system's precision we had to change the filter length according to the ideal value for the specific SNR - Linewidth combination. As mentioned before, the filterin our system is designed to operate with a window that is always a power of 2 (i.e 2, 4, 8, 16, 32, etc). So the ideal length always moves up or down to match a power of two. As an example we present the Matlab Filtering window Vs. VHDL filtering in the table 6.3 below.

Linewidth	8×10^{-3}	5×10^{-3}	1×10^{-3}	8×10^{-4}	5×10^{-4}	1×10^{-4}	8×10^{-7}
Filter (Matlab)	5	6	11	12	15	25	70
Filter (VHDL)	4	8	8	16	16	32	64

Table 6.3: The scaling up or down of the FPGA filter regarding the ideal value from simulation.

Due to this diverge from the ideal filtering lengths there is some loss in precision which is counterbalanced by a simpler design and therefore a better performance. That happens because a normal filter, able to have any length would require a divider circuit to handle lengths that are now a power of two.

Therefore, in our performance evaluation we have sometimes set the upper filter length to be 16 taps. This way we sacrifice precision but we gain a lot of performance since the 32-tap filter for the parallel design has decreased throughput performance.

6.3 Combined Simulation and Results

In this chapter we are going to present our findings regarding the hardware simulation for the various configuration sets. We will investigate not only the systems precision but also the systems performance and its behavior under the effect of phase noise and AWGN.

In the next figure 6.1 we present a comparison plot between the Matlab full precision simulation curve with the hardware derived curve (set # 4). We define the specific configuration as the lower limit (less precise) of our system. This choice will be later justified as we are going to present its linewidth tolerance and power penalty values.

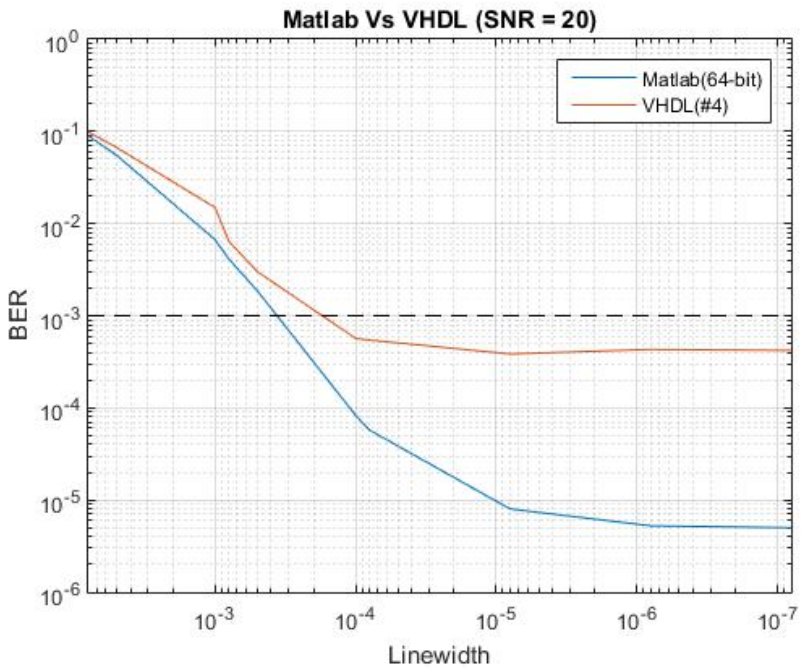


Figure 6.1: Plot showing the Simulation curve versus the VHDL curve # 4 for SNR = 20 dB.

After implementing in FPGA the specific configuration we derived its hardware utilization 6.2

graph. It is visible that in our design the most used resource are the LUT's and in many cases this high utilization is the factor that we cannot push our design to a higher performance.

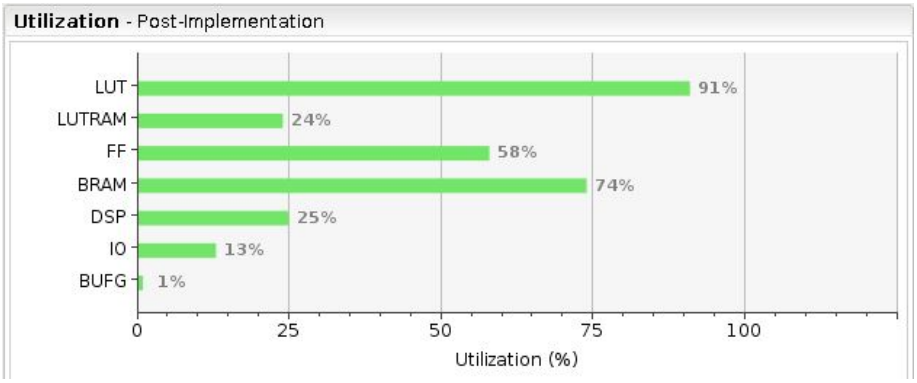


Figure 6.2: The utilization graph for the # 4 configuration set as produced by the Vivado Implementation process.

A schematic overview of the device utilization can be seen at the next figure 6.3. The light blue areas symbolize the utilized fabric while the dark blue areas indicate the unused fabric. We can clearly see that our design has used most of the FPGA fabric.

Figure 6.3: The FPGA device (Virtex-7) utilization as shown from the Vivado Implementation tool.

In the previous paragraphs we presented the performance of set # 4 alone. In the next figure (6.4) we present various hardware configurations, that were presented in the table 6.2. We have chosen our configurations so that their performance and precision spans throughout the desired solutions space. As solutions space we define a space where hardware configurations not only provide BER $\leq 1 \times 10^{-3}$ for the desired linewidth-SNR combination, but also demonstrate elevated performance ($\geq 25GBd$ throughput rate). This way we will derive various configurations which will provide a flexible solution set so as to be able to operate under variable noise environment.

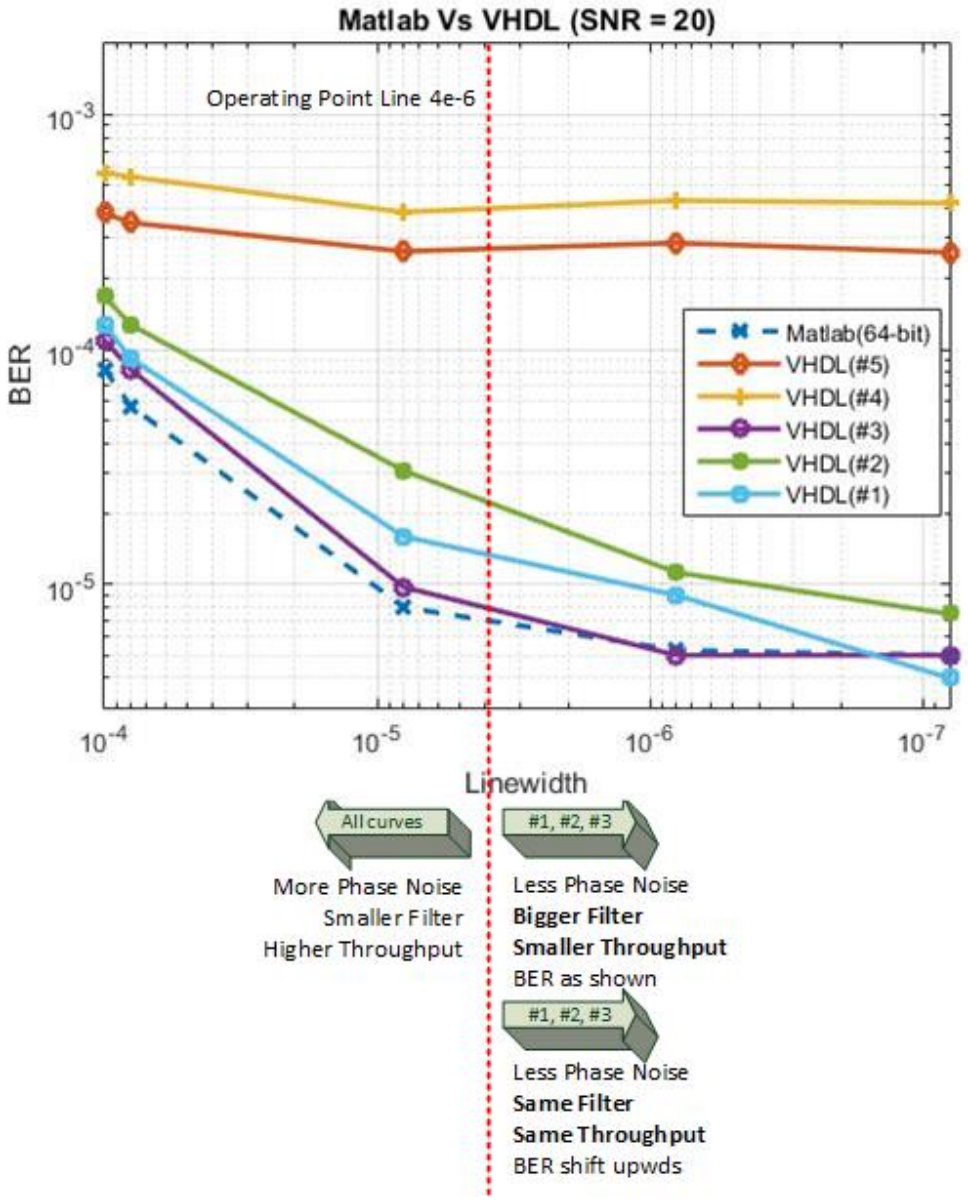


Figure 6.4: The plot shows all the BER curves for the configurations presented above along with the operating point line.

At the plot above we note that curve # 1 seems to go below the Matlab curve for low linewidth values. This BER value can be explained for the system gives only ≈ 15 wrong bits at the output when linewidth is 8×10^{-8}. A BER value is reliable when in produces hundreds of wrong bits (≥ 200) In order to obtain reliable BER results at this point we have to simulate the system with more than 10^6 symbols which is really computing intensive and out of the scope of this thesis.

At a first glance we notice two teams of curves. The more precise ones (closer to the theoretical optimal), and the less precise ones (shifted upwards and more far from the Matlab curve). This is logical since for the curves # 4 and # 5 we defined the maximum filtering length to be 16-taps which is of course sub-optimal in the low linewidth cases. When our system's noise is mainly AWGN then a bigger filter means better suppression of AWGN noise since the phase is no more rapidly fluctuating.

The more precise curves (# 1, # 2 and # 3) are derived from a configuration where the filter followed the optimal length at every case (as described on the section about filtering above). This way the curves do not exhibit a floor but steadily increase their performance as the linewidth decreases.

Those filter choices though, affect the systems performance in terms of maximum achievable frequency and throughput.

In order to asses the performance we choose the operating point at

- Linewidth $= 4 \times 10^{-6}$ (as presented in fig. 6.4)

The table 6.4 below presents the performance results for the above stated operating point.

	Performance			Note
Configuration ID	Single Pipeline Max Frequency	Max Parallel Pipelines	Parallel Pipeline Max Baud Rate	Filter
# 1	300.3 MHz	105	31.5 GBd	32 taps
# 2	303 MHz	119	36.1 GBd	32 taps
# 3	300.3 MHz	84	25.2 GBd	32 taps
# 4	333.3 MHz	140	46.6 GBd	16 taps
# 5	333.3 MHz	139	46.3 GBd	16 taps

Table 6.4: Different hardware configurations and their respective performance for the specified operating point in terms of parallelization, maximum frequency and throughput.

In the table we notice the maximum filter length described above. The performance gap between the two teams (optimal fltering, sub-optimal filtering) is visible. Both of the configurations using a 16-tap filter produce a throughput rate beyond 46 GBd while on the other hand the configurations with optimal filtering yield a less fast design operating from 25GBd up to $\approx$ 36GBd.

The laser linewidth chosen for the operating point translates to (112 kHz for 28GBd throughput) which describes an ECL laser well within the range of the modern commercially available ECL lasers. If the phase noise were greater, the the operating points would move up the respective curves (to the left) yielding less performance but potentially producing better throughput rate (since bigger phase noise demands smaller filter which in turns means better achievable throughput).

On the other hand, with a smaller phase noise, the operating points of curves (# 4 and # 5) would just move to the right maintaining the throughput (since the filter length would not change) while those of the curves (# 1, # 2 and # 3) could either operate with the same filter, and therefore maintain their throughput, but produce a less precise result (curve would shift upwards), or use a bigger filter, and therefore stay on the plotted curve, but lower their data rate performance.

To sum up, in our design we had the opportunity to choose either between a more performance oriented configuration were we will get potentially worse BER values (like sets # 4, # 5) or aim for a more precise and accuracy oriented implementation that however provides us with lower baud rates (configurations # 1, # 2 and # 3). This flexibility of system reconfiguration for different environment is a powerful feature that adapts the system at a potentially changing environment.

Another important factor we have to we have to take into account in our design's evaluation is its power penalty (or SNR penalty) respective to the theoretical simulation. . This issue is going to be discussed in the next section.

6.3.1 SNR Penalty

This metric describes the amount of power we have to provide to our system, with reference to the power provided to the theoretical model in order to achieve the same BER value. As a ceiling for a successfully operating system we define a BER value of 1×10^{-3}.

We defined the limits of our laser linewidth values to be 100 kHZ and 2 MHz. The upper limit (2 MHz) is quite big for a modern commercially available ECL laser (normally 100 - 300 kHZ) but we wanted to test our system for a worst case scenario with big phase noise. The lower limit (100 kHZ) is a good value to test our smallest power penalty with, though state-of-the-art lasers can produce a much smaller phase noise. All things considered the values that are going to be presented here are the limits for the system's performance and provided with better quality lasers (less noisy) the system exhibits a significantly smaller power penalty.

In the table 6.5 below we can see the SNR penalty values that the system exhibits for different linewidth values for a FEC limit of 1×10^{-3} with a 11 % overhead.

	Performance	
Configuration ID	100 kHz	2 MHz
# 1	0.3 dB	0.2 dB
# 3	0.5 dB	0.3 dB
# 4	2 dB	1.6 dB
# 5	1.7 dB	1.3 dB

Table 6.5: The SNR penalty required for every hardware configuration to achieve a BER equal to 1×10^{-3} for different laser linewidth values. The penalty is calculated with respect to the theoretical limit calculated with Matlab.

We notice that the system exhibits its maximum power penalty value for the # 4 configuration. That is expected since this configuration, along with the # 5, are the ones tailored for better performance than BER results.

The fact that the system under better linewidth conditions exhibits worse SNR penalty can be attributed to the fact that since our configurations are operating with fixed filter length they cannot improve their values when phase noise becomes negligible and AWGN governs the system. At that point the filter needed is exceeding 64 taps and such a filter would be a performance bottleneck for our system.

To sum up, we notice that we can choose to operate either with high precision or with high performance.

All things considered, we proved that the implemented FPGA design gave us the both the power to create a functional design and the flexibility to tune it according to our needs (precision, throughput rate).

Chapter 7

Conclusions

In the current thesis we presented, assessed through simulations and finally implemented on an FPGA platform a hardware-efficient and high-performance carrier phase recovery algorithm aiming at state-of-the-art digital coherent 16-QAM receivers.

The implemented system was designed from scratch at VHDL code and is fully-parallel and fully-parametrizable. That was the key enabler to explore the infinite hardware configuration schemes and present the ones that are capable of either opting for higher performance or aiming at high output precision. Moreover, with some design compromises that undoubtedly made the algorithm hardware friendly we were able to achieve operating frequencies of up to 333 MHz and external parallelization order of up to 140, thus fully exploiting the parallelism and adaptability that FPGA platforms offer.

This way ,our carrier phase recovery implementation operates at a throughput rate beyond 46 GBd which is considered state-of-the-art even for today's telecommunication standards. With such a throughput rate, and provided that we proved the successful implementation of a dual polarization system, our design could be used for systems operating beyond 300 or even reaching 400 Gbps with the use of super-channels in backbone fiber interconnects. The design is also able to yield negligible power penalty ,with respect to the theoretical limit, such as 0.2 dB even for incredibly noisy lasers (with linewidth of 2MHz and throughput rate of 28 Gbd). All those features make the system an ideal candidate for a next-generation highly-efficient digital coherent receiver fro 16-QAM constellations.

To sum up, we proved that the NLS Estimator algorithm, slightly altered by our polynomial approximation, becomes a low-complexity but highly efficient algorithm able to provide great results, comparable with the widely used industry benchmark (Blind Phase Search algorithm).

A future project could be the actual FPGA implementation of the algorithm and embodiment in an optical pipeline n order to experiment with real telecommunication data. Another interesting and potentially prosperous endeavour, would be the partial reconfiguration of the FPGA platform so as to fully alter its configuration in real-time and thus be able to adapt to different environments (more AWGN noise present, more phase noise present, demand for higher throughput) and yield the required results without wasting any power or system resources from the telecommunication system.

Bibliography

[1] Ezra Ip and Joseph M Kahn. Feedforward carrier recovery for coherent optical communications. *Journal of Lightwave Technology*, 25(9):2675–2692, 2007.

[2] Timo Pfau, Sebastian Hoffmann, and Reinhold Noé. Hardware-efficient coherent digital receiver concept with feedforward carrier recovery for m-qam constellations. *Journal of Lightwave Technology*, 27(8):989–999, 2009.

[3] Nikolaos Argyris, Stefanos Dris, Christos Spatharakis, and Hercules Avramopoulos. High performance carrier phase recovery for coherent optical qam. In *Optical Fiber Communications Conference and Exhibition (OFC), 2015*, pages 1–3. IEEE, 2015.

[4] Winston Way. Merits of coherent detection optical transmission. `https://www.neophotonics.com/merits-coherent-detection-optical-transmission/`, 2015.

[5] Kelly Hill. What is 64 qam? `http://www.rcrwireless.com/20160901/test-and-measurement/what-is-64-qam-tag6-tag99`, 2016.

[6] Dr. Jürg M. Stettbacher. Dsp functions on fpgas. `https://in.mathworks.com/company/newsletters/articles/dsp-functions-on-fpgas.html`, 2004.

[7] Ioannis Roudas. Coherent optical communication systems. In *WDM Systems and Networks*, pages 373–417. Springer, 2012.

[8] Kazuro Kikuchi. Coherent optical communications: Historical perspectives and future directions. In *High Spectral Density Optical Communication Technologies*, pages 11–49. Springer, 2010.

[9] Pedro Fernandez Acuna. Phase noise filtering for coherent optical communications.

[10] Ezra Ip, Alan Pak Tao Lau, Daniel JF Barros, and Joseph M Kahn. Coherent detection in optical fiber systems. *Optics express*, 16(2):753–791, 2008.

[11] Michael Rice, Chris Dick, and Fred Harris. Maximum likelihood carrier phase synchronization in fpga-based software defined radios. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 2, pages 889–892. IEEE, 2001.

[12] Pavel Fiala and Richard Linhart. High efficient carrier phase synchronization for sdr using cordic implemented on an fpga. In *Telecommunications Forum Telfor (TELFOR), 2015 23rd*, pages 512–515. IEEE, 2015.

[13] Reinhold Noe, Sebastian Hoffmann, Timo Pfau, Olaf Adamczyk, Vijitha Herath, Ralf Peveling, and Mario Porrmann. Realtime digital golarization and carrier recovery in a polarization-multiplexed optical qpsk transmission. In *IEEE/LEOS Summer Topical Meetings, 2008 Digest of the*, pages 99–100. IEEE, 2008.

[14] Benedikt Baeuerle, Arne Josten, Felix C Abrecht, Edwin Dornbierer, Jonathan Boesser, Michael Dreschmann, Juergen Becker, Juerg Leuthold, and David Hillerkuss. Multiplier-free, carrier-phase recovery for real-time receivers using processing in polar coordinates. In *Optical Fiber Communication Conference*, pages W1E–2. Optical Society of America, 2015.

[15] Wikipedia. Moving average — wikipedia, the free encyclopedia, 2017. [Online; accessed 20-June-2017].

[16] Yan Wang, Erchin Serpedin, and Philippe Ciblat. Optimal blind nonlinear least-squares carrier phase and frequency offset estimation for general qam modulations. *IEEE Transactions on Wireless Communications*, 2(5):1040–1054, 2003.

[17] Irshaad Fatadin, David Ives, and Seb J Savory. Laser linewidth tolerance for 16-qam coherent optical systems using qpsk partitioning. *IEEE Photonics Technology Letters*, 22(9):631–633, 2010.

[18] Logicore ip product guide. `https://www.xilinx.com/support/documentation/ip_documentation/cordic/v6_0/pg105-cordic.pdf`, 2016.