ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση αλγορίθμου ετεροσυσχέτισης υψηλών επιδόσεων σε fpga για τηλεπικοινωνιακές εφαρμογές

Διπλωματική Εργασία

του

Νικολαΐδη Δημήτρη

Επιβλέπων: Δημήτριος Σούντρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώμβριος 2019

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση αλγορίθμου ετεροσυσχέτισης υψηλών επιδόσεων σε fpga για τηλεπικοινωνιακές εφαρμογές

Διπλωματική Εργασία
του
Νικολαΐδη Δημήτρη

Επιβλέπων: Δημήτριος Σούντρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Οκτωμβρίου 2019.

(Υπογραφή)	(Υπογραφή)	(Υπογραφή)
.....	.....	.....
Δημήτριος Σούντρης	Παναγιώτης Τσανάκας	Ηρακλής Αβραμόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώμβριος 2019

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

ΠΕΡΙΛΗΨΗ

Στην πορεία υλοποίησης του 5G είναι πολύ σημαντική η ανάπτυξη εξαρτημάτων,σε ψηφιακό επίπεδο,που θα μπορούν να ανταποκριθούν στις απαιτήσεις των πρωτοκόλλων επικοινωνίας.Ένα από αυτά τα ψηφιακά εξαρτήματα είναι ο ανιχνευτής σήματος.Σκοπός του ανιχνευτή σήματος είναι η ανίχνευση και επαναμετάδοση συγκεκριμένου σήματος ,που υποδηλώνει προσπάθεια επικοινωνίας από τον πομπό, στο δέκτη.Ακόμα πέρα από την σωστή αναμετάδοση του σήματος,υπάρχει μεγάλη ανάγκη ο ανιχνευτής να καταναλώνει όσο το δυνατόν λιγότερους πόρους ,καθώς το 5G αποτελείται από πολλά μέρη και ο χώρος πάνω στα fpga είναι περιορισμένος.Τέλος, βασικότερο προαπαιτούμενο για την ενσωμάτωση του ανιχνευτή στο σύστημα, είναι η σωστή λειτουργία του σε μεγάλη συχνοτητα ρολογιού,έτσι ώστε να μπορεί να ανταποκριθεί στις μεγάλες ταχύτητες του 5G.

Για την επίτευξη της ανίχνευσης σημάτων χρησιμοποιείται η τεχνική της ετεροσυσχέτισης με την μέθοδο του πολλαπλασιασμού.Η ετεροσυσχέτιση εφαρμόστηκε αρχικά στο μέτρο του σήματος και στην συνέχεια στο πραγματικό του μέρος .Τελικά, αποδείχθηκε ότι ο συνδυασμός των δύο μεθόδων είναι αυτός που δίνει τα καλύτερα αποτελέσματα.Με την εξαγωγή του μέτρου και την αφαίρεση από αυτό συγκεκριμένης προυπολογισμένης ποσότητας καταλήγουμε να έχουμε θετικούς και αρνητικούς αριθμούς χωρίς να επηρεαζόμαστε από την γωνία του διανύσματος.Αυτό μας επιτρέπει να έχουμε μεγάλη ακρίβεια ανεξάρτητα από την επίδραση του διαύλου (στροφή φάσης).

Τα σήματα που θα ανιχνεύονται είναι σήματα των 320 σημείων με ακρίβεια 16 bit(full precision).Η αρχιτεκτονική θα πρέπει να διαθέτει την δυνατότητα εκμετάλλευσης όλου του εύρους των bit στις εισόδους, στο εσωτερικό και στις εξόδους της.

Η παρούσα διπλωματική εργασία στοχεύει στην υλοποίηση ενός τέτοιου ανιχνευτή σημάτων με την λιγότερη δυνατή κατανάλωση πόρων πάνω στην πλακέτα (XCZU28DR-2FFVG1517E RFSoC) και την σωστή λειτουργία στα 500Mhz.Στόχος είναι η υλοποίηση του κυκλώματος σε μία έκταση της τάξης των 9000-10000 LUTS και των 100 μονάδων DSPs.Ουσιαστικά το κύκλωμα πρέπει να καταλαμβάνει περίπου το 2% των πόρων του fpga.

Η αυστηρή απαίτηση υλικού και χρονισμού επιτεύχθηκε με την υλοποίηση δέντρου αθροιστών για την μείωση των αθροιστών στο υπολογισμό του τελικού αθροίσματος, και την εκμετάλλευση της περιοδικότητας των σημάτων ώστε να γίνει αποδοτικότερη χρήση των λειτουργικών μονάδων του fpga(DSPs).

Μετά από εκτενή έλεγχο με την βοήθεια του matlab διαπιστώθηκε, πως το κύκλωμα λειτουργεί σωστά στην εισαγωγή μεγάλων εισόδων, καθώς και ότι είναι ανθεκτικό στην παραμόρφωση και εξασθένηση που προκαλεί ο τηλεπικοινωνιακός δίαυλος πάνω στο διαδιδόμενο σήμα.

Λέξεις κλειδιά:5G, ανίχνευση σήματος, ετεροσυσχέτιση, δέντρο αθροιστών, περιοδικότητα , χαμηλή κατανάλωση υλικού, 500 Mhz, παραμόρφωση, εξασθένηση, τηλεποικοινωνιακός δίαυλος

ABSTRACT

In the process of implementing 5G it is very important to develop components, digitally, that can meet the requirements of communication protocols. One of these digital components is the signal detector. The purpose of the signal detector is to detect and retransmit a specific signal, which indicates an attempt to communicate from the transmitter to the receiver. Even beyond the proper signal relay, there is a great need for the detector to consume as few resources as possible, as 5G is made up of many parts and space on fpga is limited.Finally, a key prerequisite for integrating the detector into the system is its proper operation at high clock frequency, so that it can meet the high speeds of 5G.

Cross-correlation is the technique used to achieve signal detection using the multiplication method. The cross-correlation was first applied to the magnitude of the signal and then to its real part. Eventually, it was found that the combination of the two methods gave the best results. By extracting the magnitude and subtracting from it a specific budgeted amount we end up having positive and negative numbers without affecting from the angle of the vector. This allows us to be precise regardless of the effect of the channel (phase shift).

The signals to be detected are 320-point signals with 16-bit precision (full precision). The architecture should be capable of exploiting the full range of bits at its inputs, internals and outputs.

The present thesis aims to implement such a signal detector with the least consumption of resources on the board (XCZU28DR-2FFVG1517E RFSoC) and the proper operation at 500Mhz. The purpose is to implement the circuit in a range of 9000-10000 LUTS and 100 DSPs. Basically the circuit should occupy about 2% of the fpga resources.

The strict hardware and timing requirement was achieved by implementing an adder tree to reduce the adders in the calculation of the final sum, and exploit the periodicity of the signals to make fpga modules (DSPs) more efficient.

After extensive testing with the help of matlab, it was found that the circuit works correctly for the input of large inputs, as well as being resistant to the distortion and attenuation caused by the telecommunication bus on the transmitted signal.

Keywords: 5G, signal detection, heterogeneity, adder tree, periodicity, low resource consumption, 500 Mhz, distortion, attenuation, telecommunication channel

Ευχαριστίες

Για την πραγματοποίηση αυτής της διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω πρώτα απ' όλα τον καθηγητή μου κ. Σούντρη Δημήτριο που με δέχθηκε και με εμπιστοσύνη μου ανάθεσε την υλοποίηση αυτής. Η βοήθεια του ήταν πολύτιμη, η υποστήριξη του συνεχής, και οι συμβουλές του ουσιαστικές.

Ένα μεγάλο ευχαριστώ οφείλω επίσης στον μεταδιδακτορικό ερευνητή κ. Λεντάρη Γεώργιο, η συνδρομή του οποίου ήταν καθοριστική, όπως και η συνεχής καθοδήγηση και επίβλεψή του που κατέστησε εφικτή την ολοκλήρωση αυτής. Επιπλέον θα ήθελα να αναφέρω και τη συνεισφορά του ερευνητή κ. Μαραγκού, ο οποίος με βοήθησε κατά την αρχική μου προσπάθεια ανάλυσης του θέματος.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου και τους φίλους μου που με στήριξαν με υπομονή και επιμονή στην προσπάθεια μου αυτή.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Σκοπός αυτής της εργασίας είναι η δημιουργία ενός κυκλώματος,το οποίο ανιχνεύει προεπιλεγμένο σήμα μέσα σε δίαυλο.Αυτό το πετυχαίνει με την διαδικασία της συσχέτισης σήματος πεπερασμένου μήκους,το οποίο βρίσκεται αποθηκευμένο μέσα στο κύκλωμα, με το εισερχόμενο σήμα που έρχεται από τον τηλεπικοινωνιακό δίαυλο.

Η αρχιτεκτονική αυτή θα υλοποιηθεί πάνω στην πλακέτα XCZU28DR-2FFVG1517ERFSoC η οποία ανήκει στην οικογένεια Ultrascale RFSoc+ της Xilinx.

		XCZU21DR	XCZU25DR	XCZU27DR	XCZU28DR	XCZU29DR	XCZU39DR	XCZU43DR	XCZU46DR	XCZU47DR	XCZU48DR	XCZU49DR
12-bit RF-ADC w/ DDC	# of ADCs	0	8	8	8	16	16	–	–	–	–	–
	Max Rate (GSPS)	0	4.096	4.096	4.096	2.058	2.220	–	–	–	–	–
14-bit RF-ADC w/ DDC	# of ADCs	–	–	–	–	–	–	4	8 4	8	8	16
	Max Rate (GSPS)	–	–	–	–	–	–	5.0	2.5 5.0	5.0	5.0	2.5
14-bit RF-DAC w/ DUC	# of DACs	0	8	8	8	16	16	4	12	8	8	16
	Max Rate (GSPS)	0	6.554	6.554	6.554	6.554	6.554	10.0	10.0	10.0	10.0	10.0
SD-FEC		8	0	0	8	0	0	0	8	0	8	0
Application Processing Unit		Quad-core Arm Cortex-A53 MPCore with CoreSight™; NEON and Single/Double Precision Floating Point; 32KB/32KB L1 Cache, 1MB L2 Cache										
Real-Time Processing Unit		Dual-core Arm Cortex-R5 with CoreSight; Single/Double Precision Floating Point; 32KB/32KB L1 Cache, and TCM										
Embedded and External Memory		256KB On-Chip Memory w/ECC; External DDR4; DDR3; DDR3L; LPDDR4; LPDDR3; External Quad-SPI; NAND; eMMC										
General Connectivity		214 PS I/O; UART; CAN; USB 2.0; I2C; SPI; 32b GPIO; Real Time Clock; Watchdog Timers; Triple Timer Counters										
High-Speed Connectivity		4 PS-GTR; PCIe® Gen1/2; Serial ATA 3.1; DisplayPort 1.2a; USB 3.0; SGMII										
System Logic Cells		930,300	678,318	930,300	930,300	930,300	930,300	930,300	930,300	930,300	930,300	930,300
CLB Flip-Flops		850,560	620,176	850,560	850,560	850,560	850,560	850,560	850,560	850,560	850,560	850,560
CLB LUTs		425,280	310,088	425,280	425,280	425,280	425,280	425,280	425,280	425,280	425,280	425,280
Distributed RAM (Mb)		13.0	9.6	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0
Block RAM Blocks		1,080	792	1,080	1,080	1,080	1,080	1,080	1,080	1,080	1,080	1,080
Block RAM (Mb)		38.0	27.8	38.0	38.0	38.0	38.0	38.0	38.0	38.0	38.0	38.0
UltraRAM Blocks		80	48	80	80	80	80	80	80	80	80	80
UltraRAM (Mb)		22.5	13.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5
DSP Slices		4,272	3,145	4,272	4,272	4,272	4,272	4,272	4,272	4,272	4,272	4,272
CMTs		8	6	8	8	8	8	8	8	8	8	8
Maximum HP I/O		208	299	299	299	312	312	299	299	299	299	312
Maximum HD I/O		72	48	48	48	96	96	48	48	48	48	96
System Monitor		1	1	1	1	1	1	1	1	1	1	1
GTY Transceivers		16	8	16	16	16	16	16	16	16	16	16
Transceivers Fractional PLLs		8	4	8	8	8	8	8	8	8	8	8
PCIe Gen3 x16		2	1	2	2	2	2	–	–	–	–	–
PCIe Gen3 x16 / Gen4 x8 / CCIX(3)		–	–	–	–	–	–	2	2	2	2	2
150G Interlaken		1	1	1	1	1	1	1	1	1	1	1
100G Ethernet w/ RS-FEC		2	1	2	2	2	2	2	2	2	2	2

Σχήμα 1:Zynq UltraScale+ RFSoC Feature Summary.

Όπως φαίνεται στο παραπάνω σχήμα, πρόκειται για ιδιαίτερα μεγάλη πλακέτα με τον εντυπωσιακό αριθμό των 425,280 LUTS και 4,272 DSP slices.Παρότι οι πόροι που προσφέρονται είναι πάρα πολλοί σκοπός αυτής της εργασίας είναι η όσο το δυνατόν μεγαλύτερη βελτιστοποίηση της αρχιτεκτονικής έτσι ώστε το κύκλωμα να περιορίζεται σε λιγότερο από το 2% των πόρων του fpga.

Σε αυτό το κεφάλαιο θα εξεταστεί ο όρος συσχέτιση,τα είδη συσχέτισης που υπάρχουν καθώς και οι πιο συχνές μέθοδοι με τις οποίες γίνεται η συσχέτιση.Τέλος θα δοθούν παραδείγματα έτσι ώστε να γίνει πλήρως κατανοητός ο στόχος της υλοποίησης.

Στην επεξεργασία σήματος βασικά είδη συσχέτισης που συναντούμε είναι:

-Η αυτοσυσχέτιση, επίσης γνωστή ως σειριακή συσχέτιση, είναι η συσχέτιση ενός σήματος με ένα καθυστερημένο αντίγραφο του ίδιου, ως συνάρτηση της καθυστέρησης.
Χρησιμοποείται κυρίως για την διόρθωση γωνίας που προκαλείται από την στροφή φάσης που επιβάλλει ο δίαυλος στο σήμα.

-Η ετεροσυσχέτιση είναι η συσχέτιση μεταξύ δύο σημάτων για να διαπιστωθεί ο βαθμός ομοιότητας τους.Χρησιμοποιείται για την ανίχνευση σήματος σε δίαυλο και είναι ο τύπος συσχέτισης που θα υλοποιηθεί.

Υπάρχουν πολλές μέθοδοι με τις οποίες μπορεί na επιτευχθεί η συσχέτιση.Οι πιο διαδεδομένες

είναι οι εξής:
-πολλαπλασιασμός

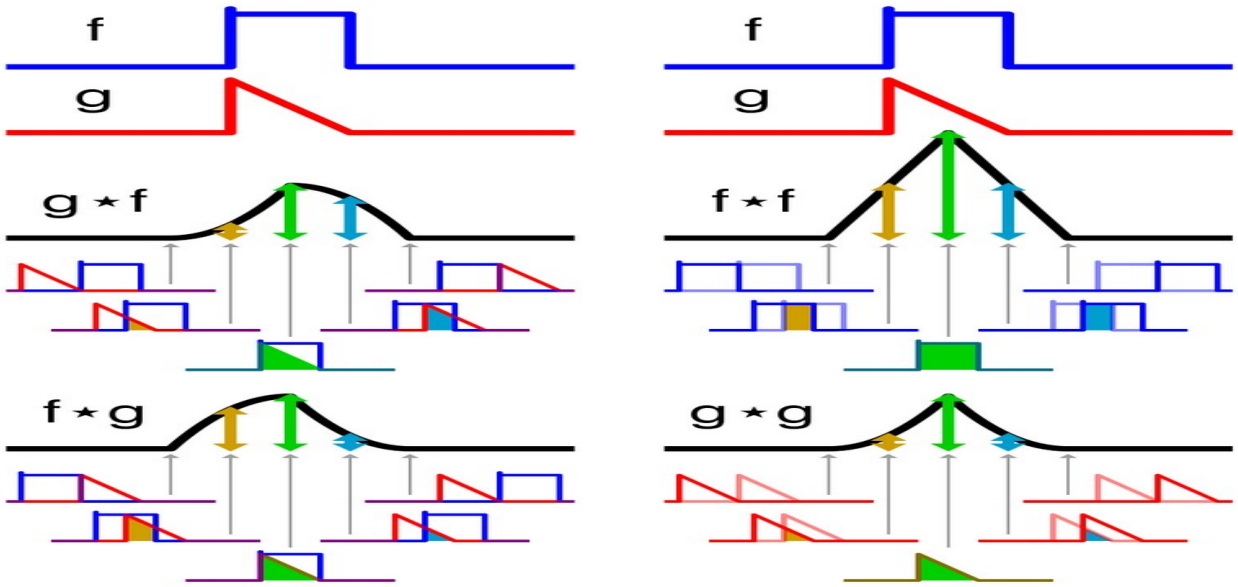
$$f*g(n)=\sum_{-\infty}^{+\infty} f(m-n)\,g(m)$$

-απόλυτο της διαφοράς

$$|(f-g)|(n)=\sum_{-\infty}^{+\infty}|(f(m-n)-g(m))|$$

-τετράγωνο διαφοράς

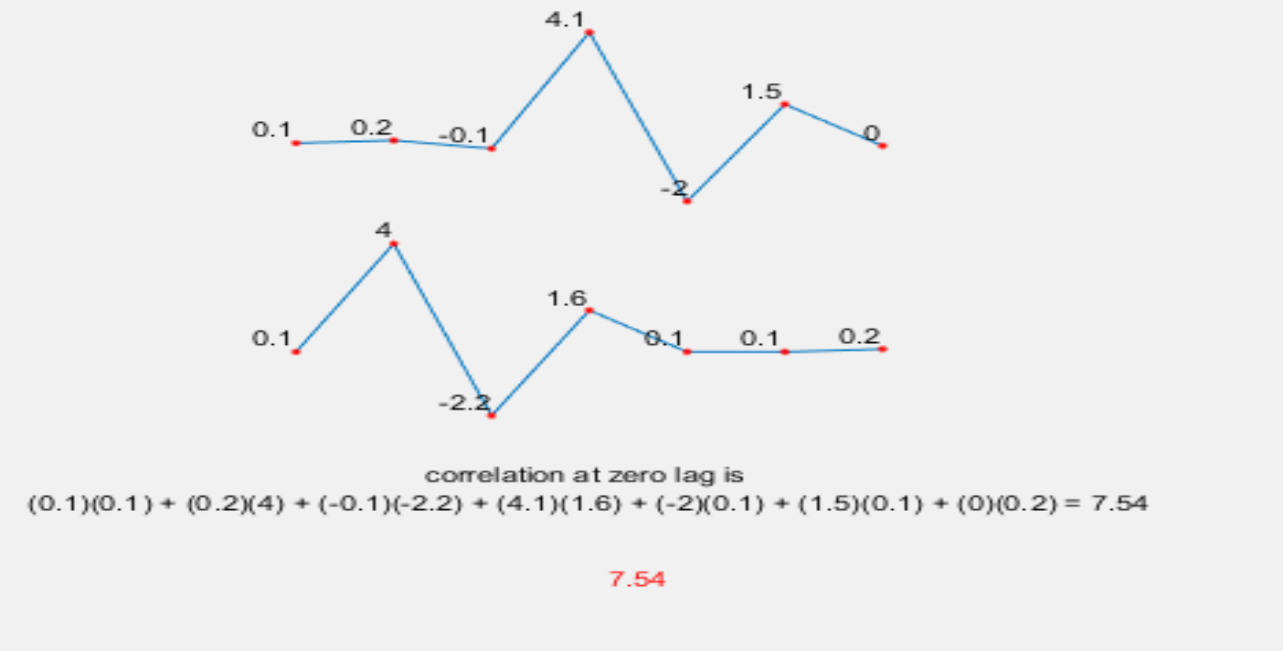
$$(f-g)^2(n)=\sum_{-\infty}^{+\infty}(f(m-n)-g(m))^2$$



<u>Σχήμα 2:Αριστερά ετεροσυσχέτιση με πολλαπλασιαμό,δεξιά αυτοσυσχέτιση με πολλαπλασιασμό.</u>

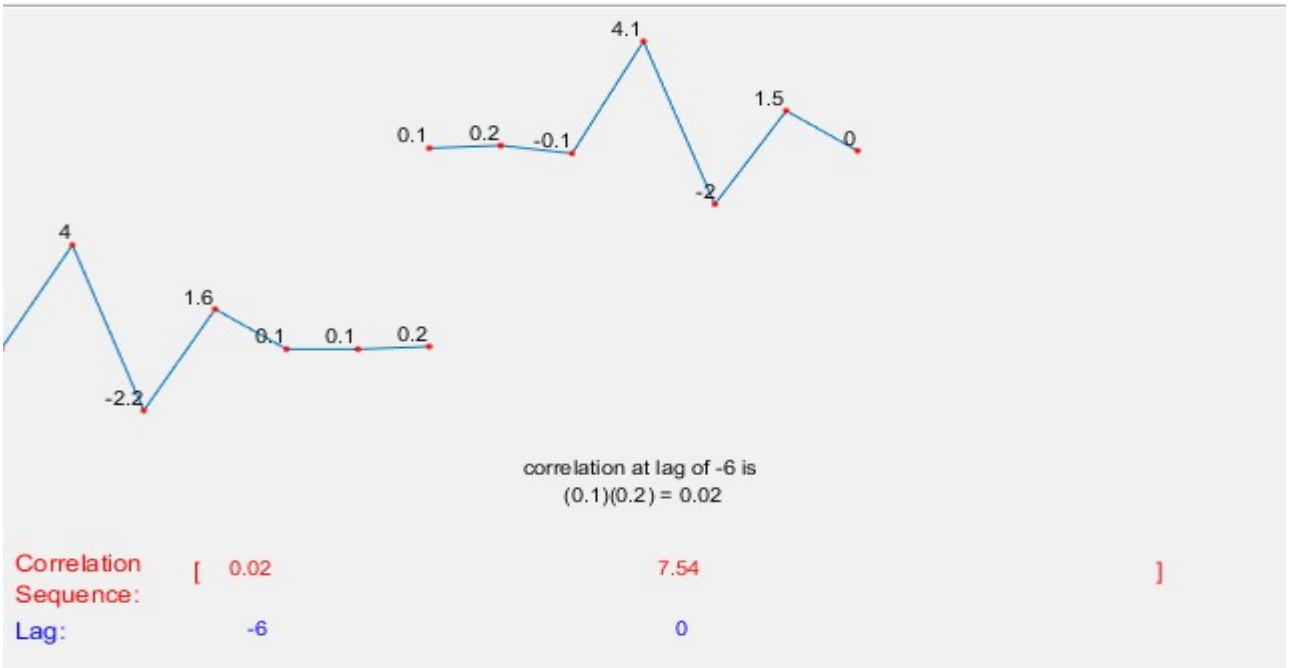
Όπως φαίνεται η συσχέτιση παίρνει την μεγαλύτερη τιμή της στην μεγαλύτερη ταύτιση που είναι δυνατόν να γίνει.

Παρακάτω ακολουθεί ένα παραδείγμα ετεροσυσχέτισης δύο διακριτών σημάτων κατά την διάρκεια της μετάδοσης του δεύτερου μέσα στο δίαυλο.Η ετεροσυσχέτιση γίνεται με πολλαπλασιασμό και για την διευκόλυνση του παραδείγματος θεωρούμε ότι όπου δεν υπάρχει σήμα έχουμε την τιμή 0.



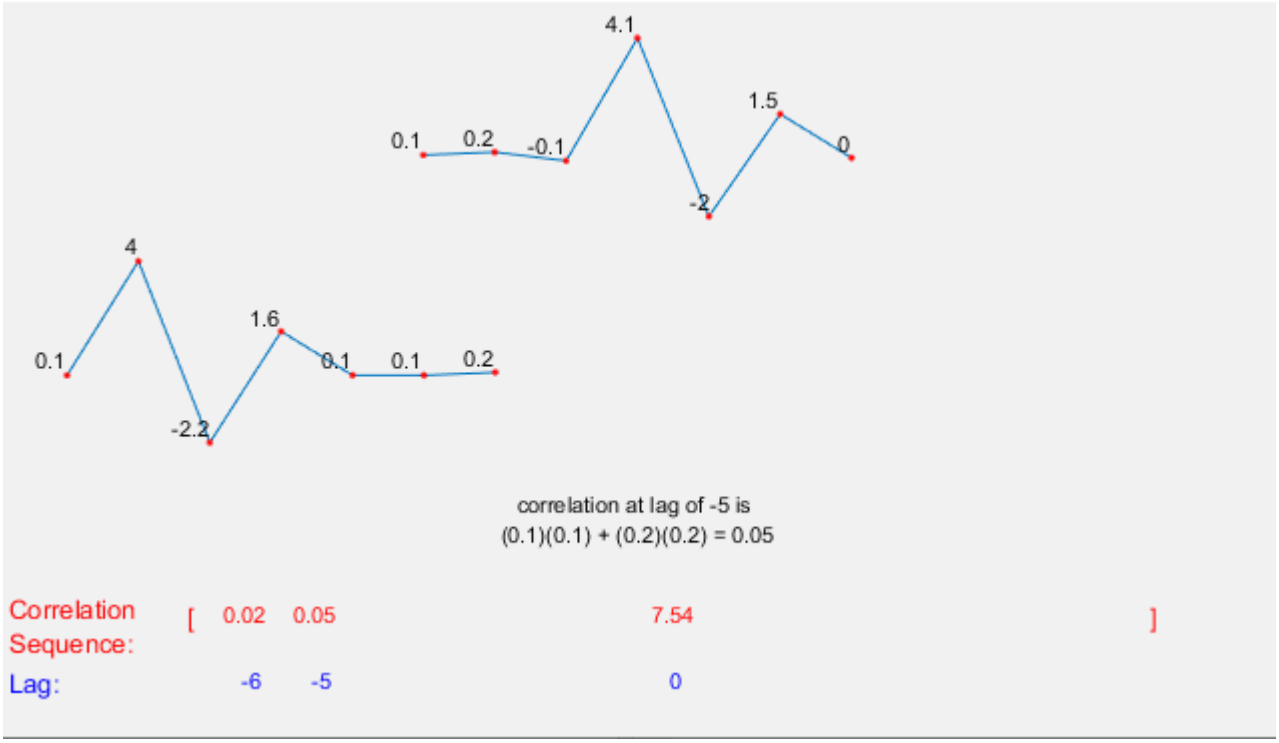
Σχήμα 3:τιμή ετεροσυσχέτισης των δύο σημάτων στο σημείο 0.

Στο πάνω σχήμα φαίνεται είναι το σταθερό σήμα που είναι γνωστό(αποθηκευμένο στο κύκλωμα) και το κάτω σχήμα είναι η ροή στο τηλεπικοινωνιακό δίαυλο.Όπως παρατηρούμε,πρακτικά η ετεροσυσχέτιση είναι η εφαρμογή της μεθόδου που έχει επιλεχθεί(π.χ πολλαπλασιασμός) σημείο προς σημείο του ενός σήματος προς τα αντίστοιχα σημεία του άλλου σήματος.Τα διαφορετικά στιγμιότυπα που είναι αποτυπωμένα στα παρακάτω σχήματα δείχνουν την κίνηση του δεύτερου σήματος χρονικά μέσα στο τηλεπικοινωνιακό δίαυλο.Η διαδικασία αυτή που περιγράφεται στις παραπάνω εικόνες είναι ο αλγόριθμος που μελετήθηκε και υλοποιήθηκε πάνω στο fpga.

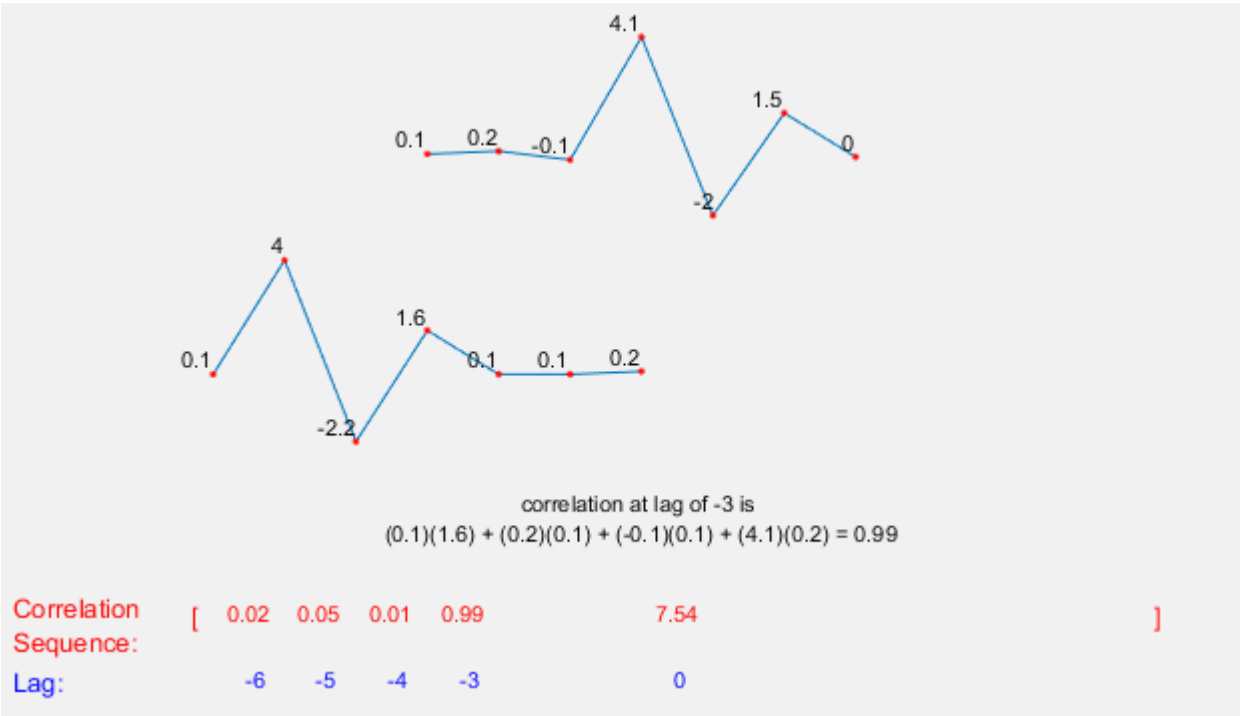


Σχήμα 4:τιμή ετεροσυσχέτισης για χρονική μετατόπιση -6 σημείων(προς τα αριστερά)

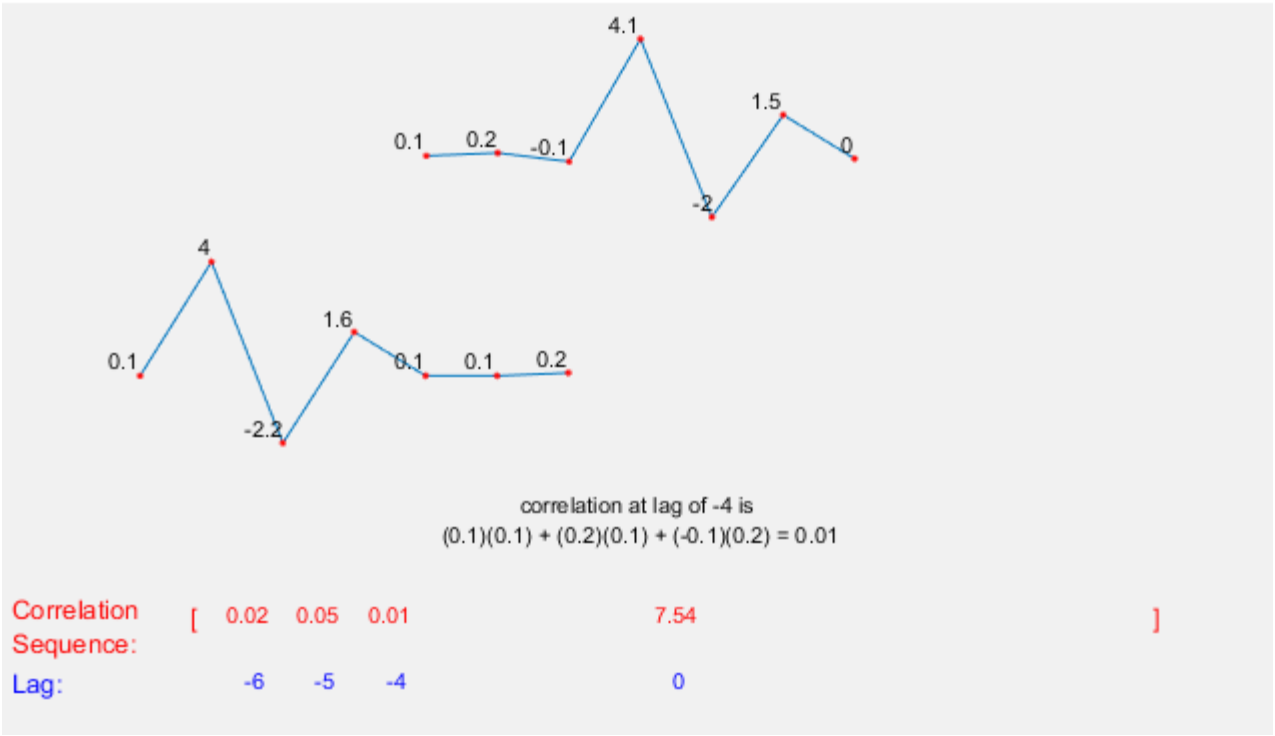
από την αρχική θέση.



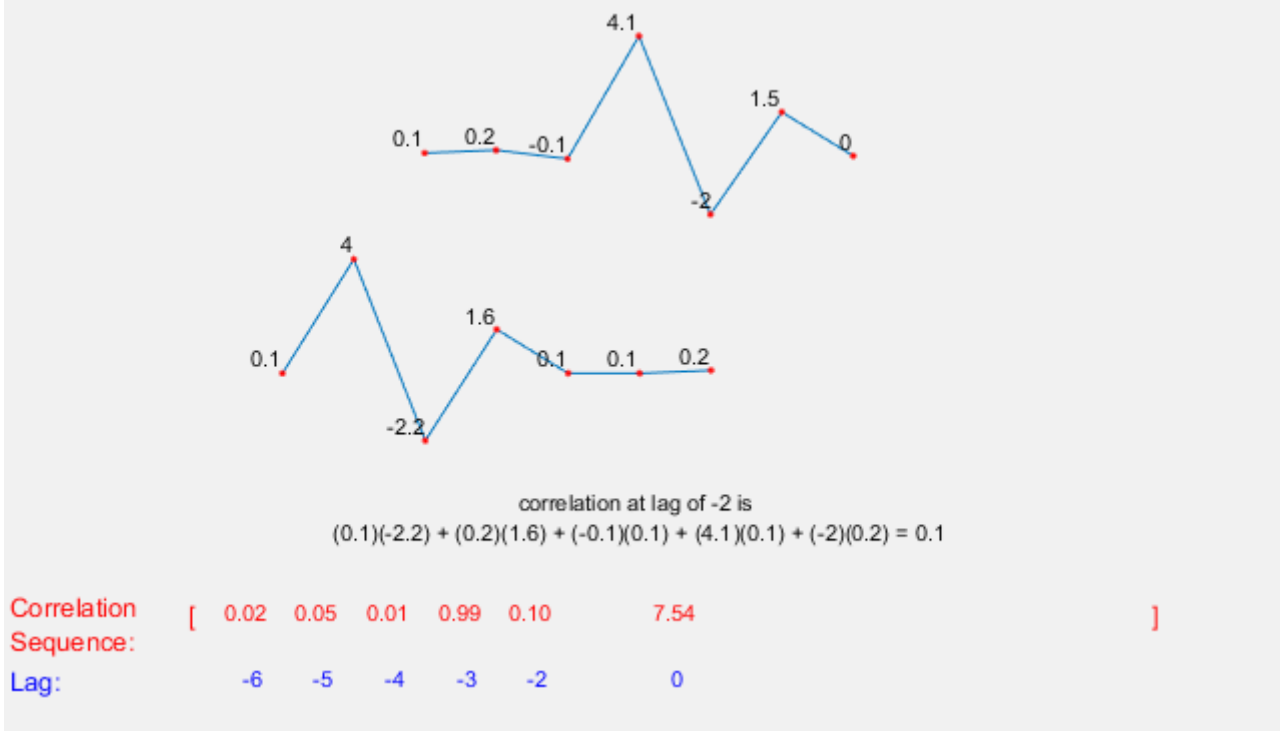
Σχήμα 5:τιμή ετεροσυσχέτισης για χρονική μετατόπιση -5 σημείων(προς τα αριστερά) από την αρχική θέση.



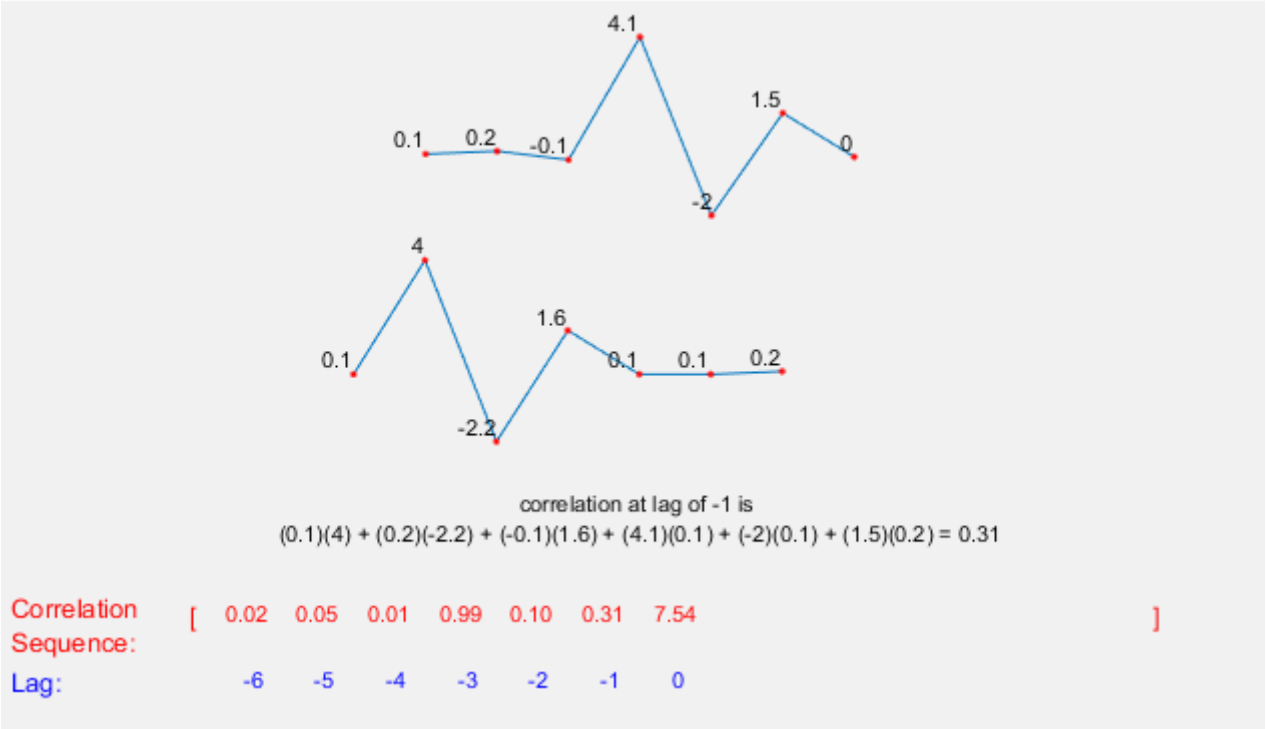
Σχήμα 6:τιμή ετεροσυσχέτισης για χρονική μετατόπιση -4 σημείων(προς τα αριστερά) από την αρχική θέση.



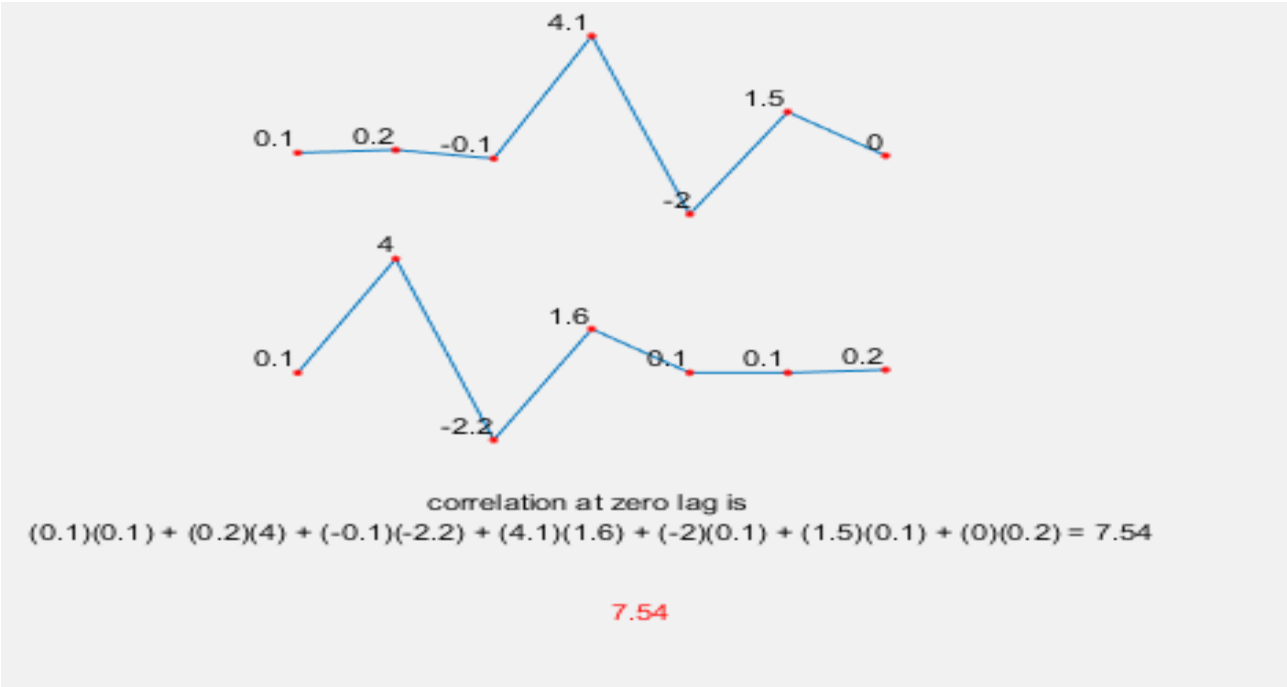
Σχήμα 7: τιμή ετεροσυσχέτισης για χρονική μετατόπιση -3 σημείων(προς τα αριστερά) από την αρχική θέση.



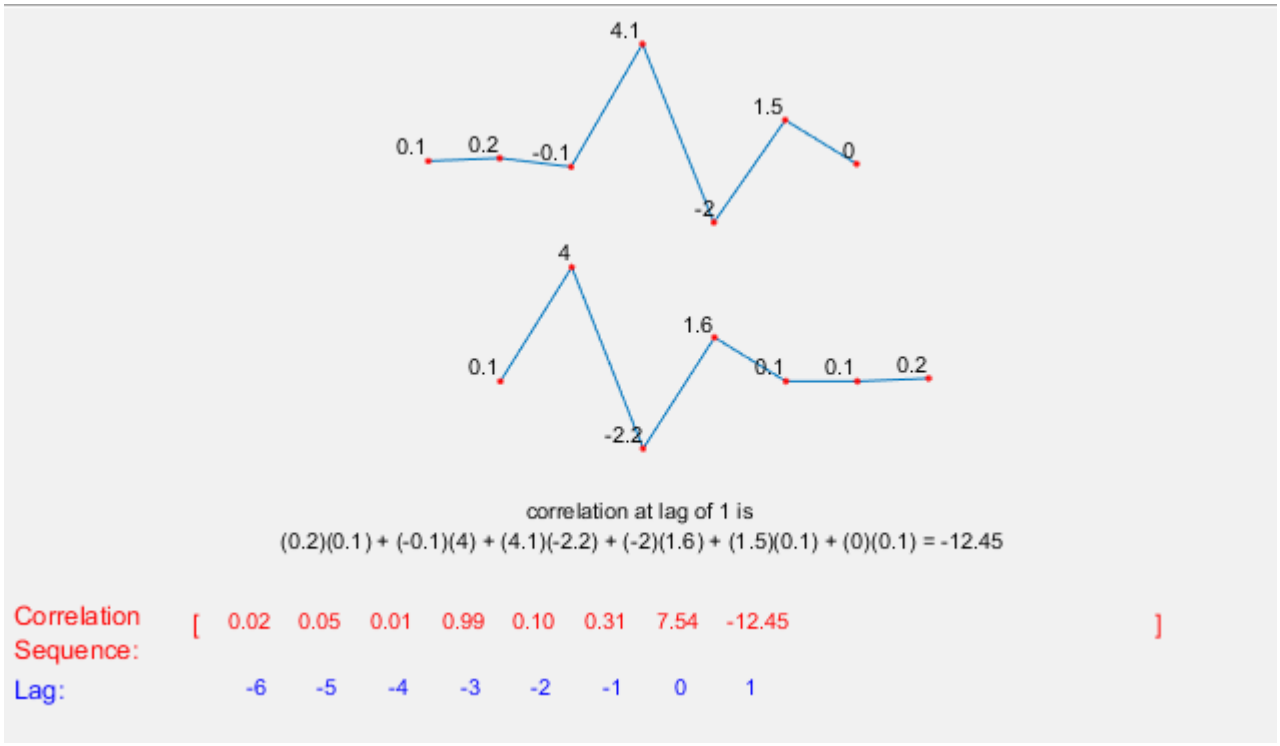
Σχήμα 8:τιμή ετεροσυσχέτισης για χρονική μετατόπιση -2 σημείων(προς τα αριστερά) από την αρχική θέση.



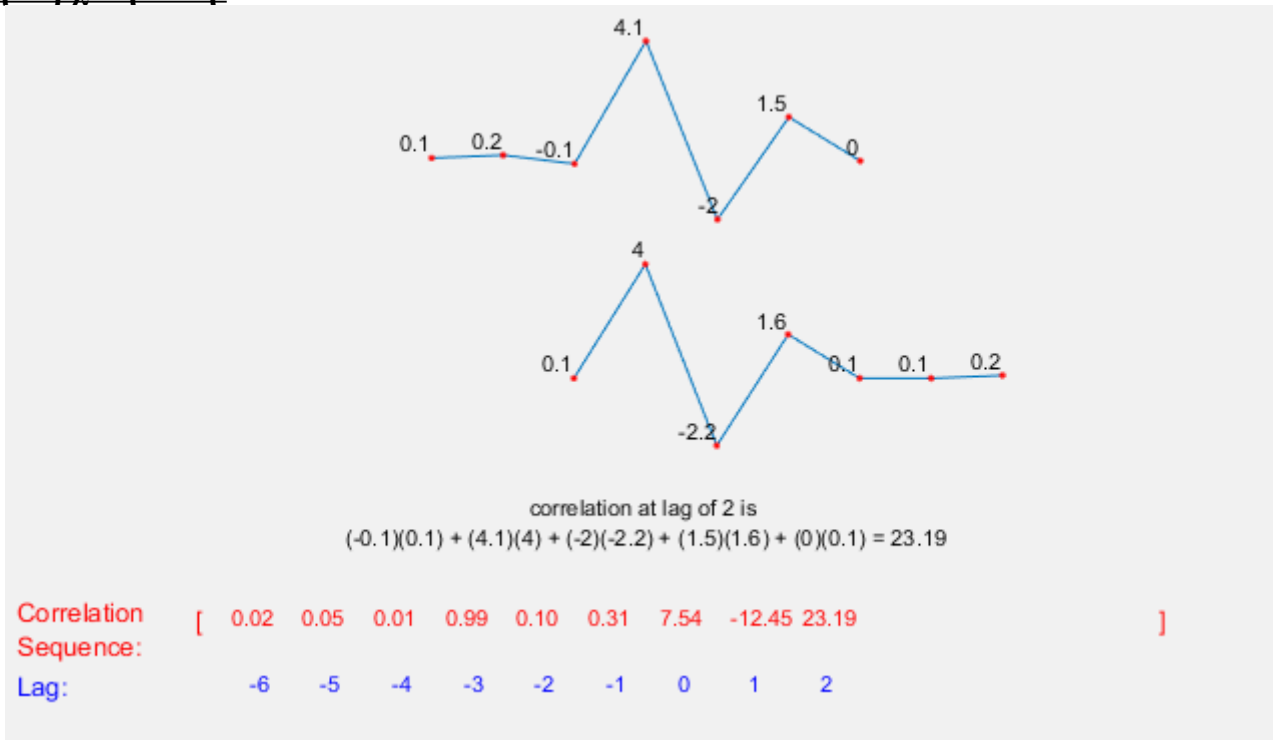
Σχήμα 9:τιμή ετεροσυσχέτισης για χρονική μετατόπιση -1 σημείου(προς τα αριστερά) από την αρχική θέση.



Σχήμα 10:αρχική εικόνα.



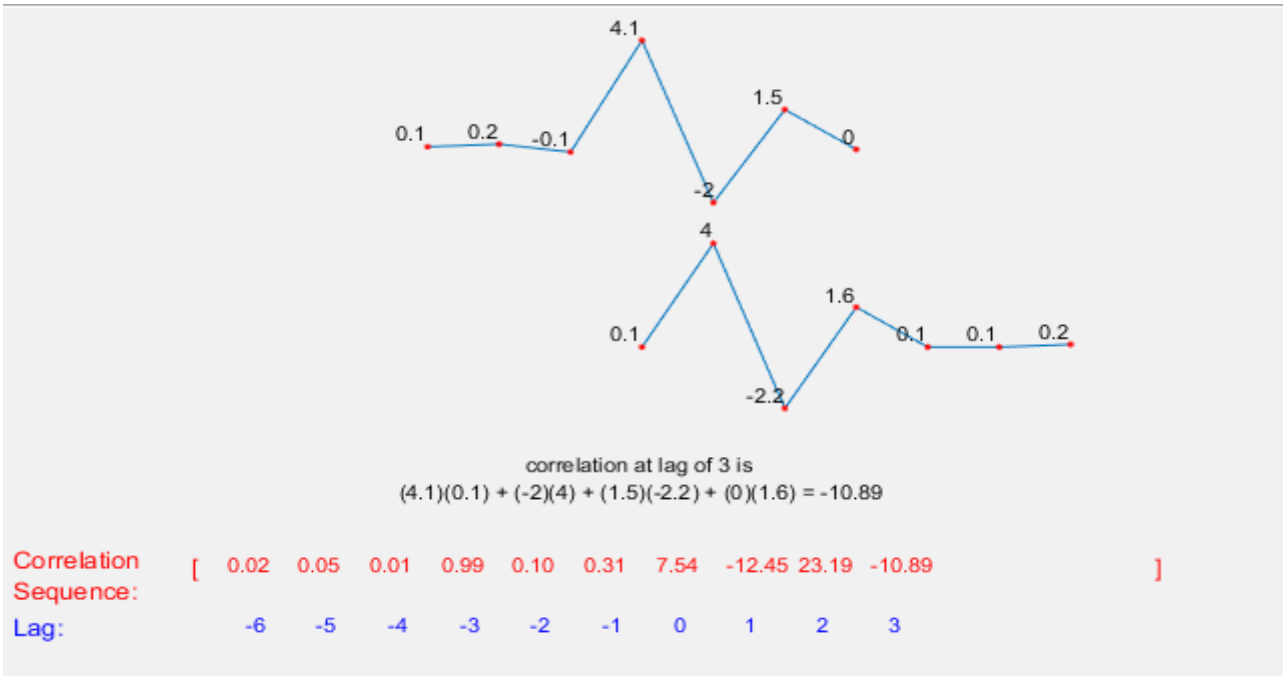
Σχήμα 11:τιμή ετεροσυσχέτισης για χρονική μετατόπιση 1 σημείου(προς τα δεξιά) από την αρχική θέση.



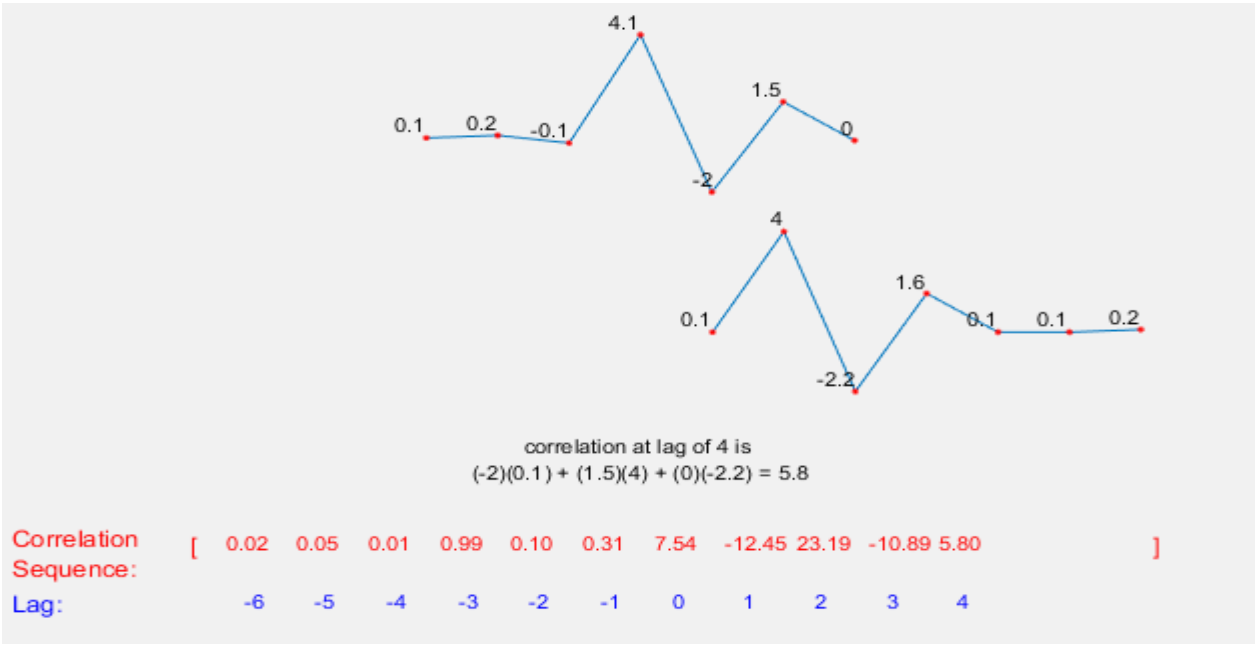
Σχήμα 12:τιμή ετεροσυσχέτισης για χρονική μετατόπιση 2 σημείων(προς τα δεξιά) από την αρχική θέση.

Παραπάνω βλέπουμε ότι σε αυτό το σημείο που τα σήματα έχουν την μεγαλύτερη δυνατή

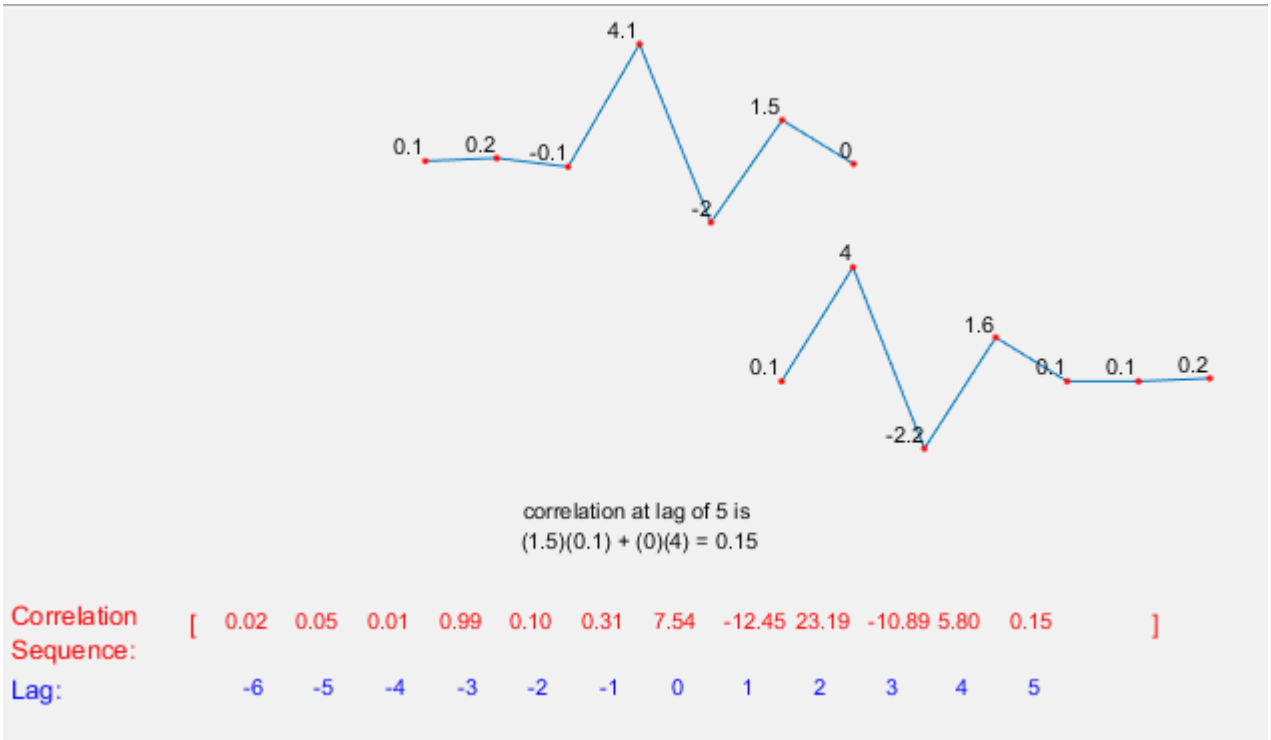
ομοιότητα η τιμή της συσχέτισης παίρνει την μεγαλύτερη δυνατή τιμή της.Αυτήν την ιδιότητα θα εκμεταλλευτούμε για να εντοπίσουμε τα ζητούμενα σήματα.



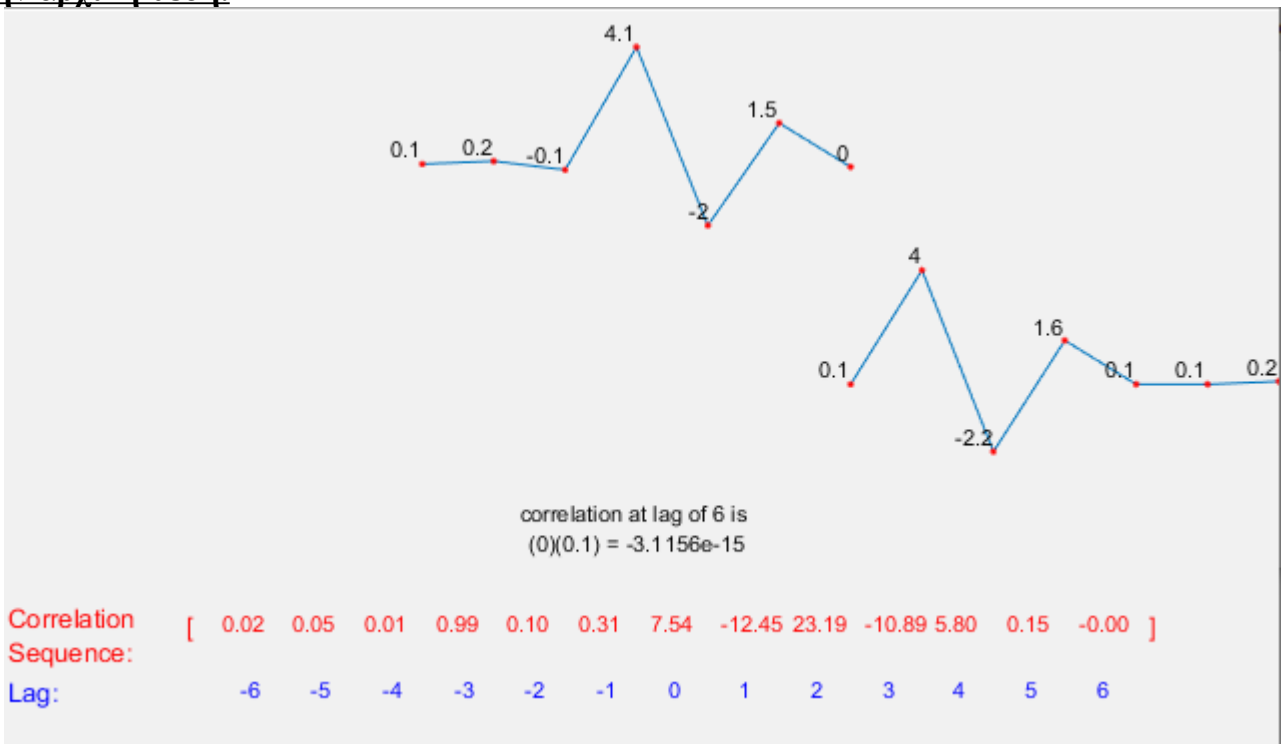
Σχήμα 13:τιμή ετεροσυσχέτισης για χρονική μετατόπιση 3 σημείων(προς τα δεξιά) από την αρχική θέση.



Σχήμα 14:τιμή ετεροσυσχέτισης για χρονική μετατόπιση 4 σημείων(προς τα δεξιά) από την αρχική θέση.



Σχήμα 15:τιμή ετεροσυσχέτισης για χρονική μετατόπιση 5 σημείων(προς τα δεξιά) από την αρχική θέση.



Σχήμα 16:τιμή ετεροσυσχέτισης για χρονική μετατόπιση 6 σημείων(προς τα δεξιά) από την αρχική θέση.

Μολονότι ο ανιχνευτής σήματος θα μπορούσε να χρησιμοποιηθεί σε πολλά πεδία,η υλοποίησή του σε αυτήν την διπλωματική εργασία έγινε με κύριο σκεπτικό τον εντοπισμό

προοιμίου(preamble) σήματος.Προοίμιο(preamble) στις τηλεπικοινωνίες ονομάζεται ένα σήμα μικρής διάρκειας που μπαίνει μπροστά από την μετάδοση των δεδομένων έτσι ώστε ο δέκτης να αντιληφθεί την πρόθεση του πομπού να επικοινωνήσει.Ευθύνη λοιπόν του ανιχνευτή είναι η ανίχνευση του προιμοίου και η μετάδοση της σωστής πληροφορίας του πομπού σε όσο το δυνατό δυσμενείς συνθήκες καναλιού.Η μέθοδος συσχέτισης που εξετάστηκε σε αυτή την εργασία είναι το ο πολλαπλασιασμός.

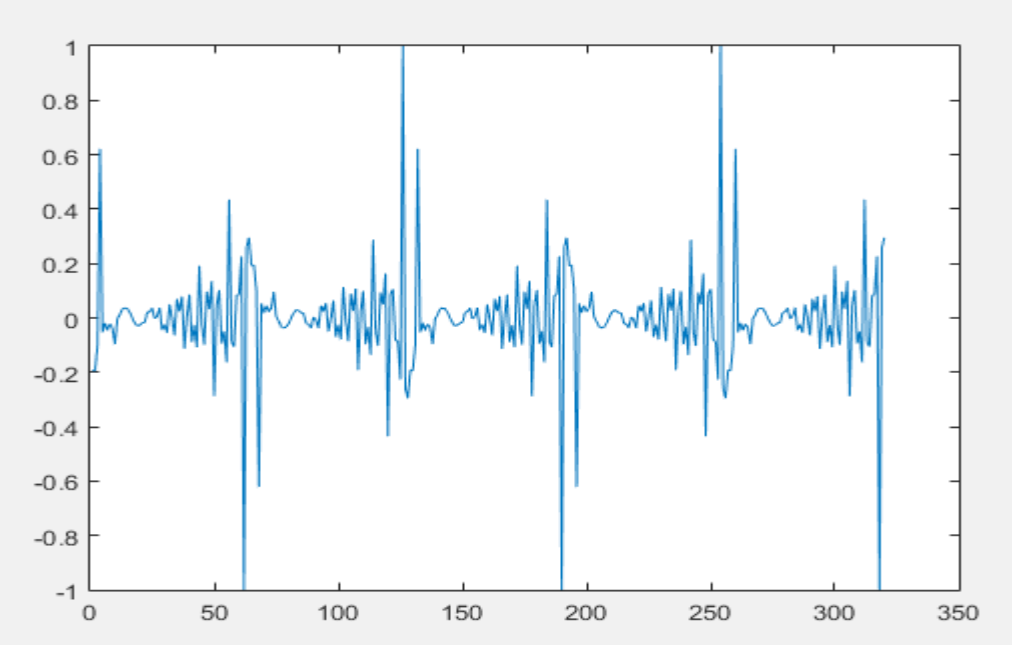
Στα κεφάλαια 1,2,3 θα ασχοληθούμε με την θεωρητική προσέγγιση της συσχέτισης έτσι ώστε να καταλήξουμε ποιο είναι το αποδοτικότερο θεωρητικό μοντέλο που στην συνέχεια θα οδηγηθεί στην υλοποίηση.Στα κεφάλαια 4,5,6 παρουσιάζεται η ακριβής υλοποίηση του ανιχνευτή και τέλος στα κεφάλαια 7,8,9 έχουμε τα αποτελέσματα της υλοποίησης και της προσωμοίωσης διαφορων εισόδων.

ΜΕΡΟΣ ΠΡΩΤΟ:ΘΕΩΡΙΑ

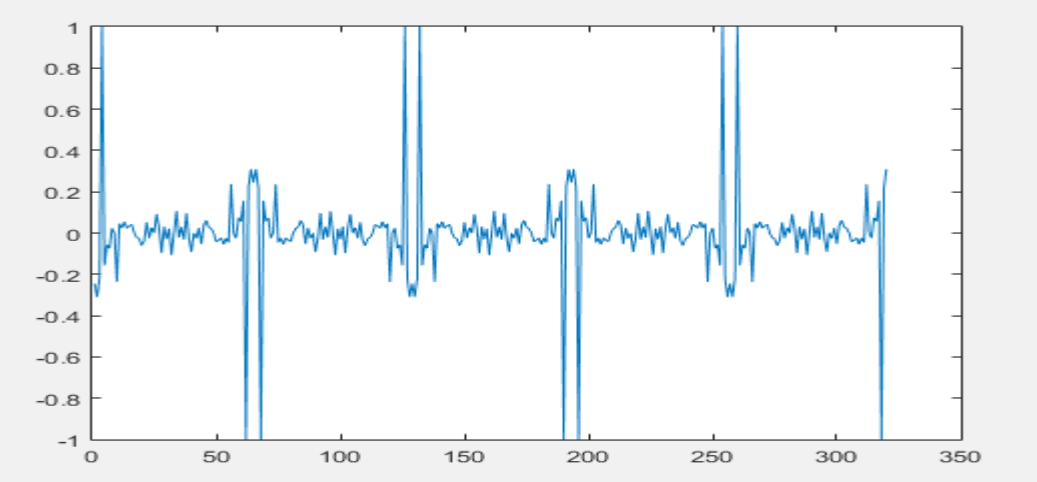
ΚΕΦΑΛΑΙΟ 1-ΑΝΙΧΝΕΥΣΗ ΜΕ ΤΟ ΜΕΤΡΟ ΣΗΜΑΤΟΣ

Το τηλεπικοινωνιακό σήμα που καλούμαστε να επεξεργαστούμε έχει την μαθηματική μορφή R+Ij όπου R είναι το πραγματικό μέρος και I το φανταστικό μέρος.Δυστυχώς η μιγαδική εφαρμογή της ετεροσυσχέτισης είναι πολύ δύσκολο να απεικονιστεί στο υλικό, συνεπώς θα στραφούμε στην εφαρμογή της ετεροσυσχέτισης στο πεδίο των πραγματικών αριθμών.Αρχικά θα εξάγουμε το μέτρο του μιγαδικού $\sqrt{R^2+I^2}$ και θα το χρησιμοποιήσουμε για την αναγνώριση του σήματος.

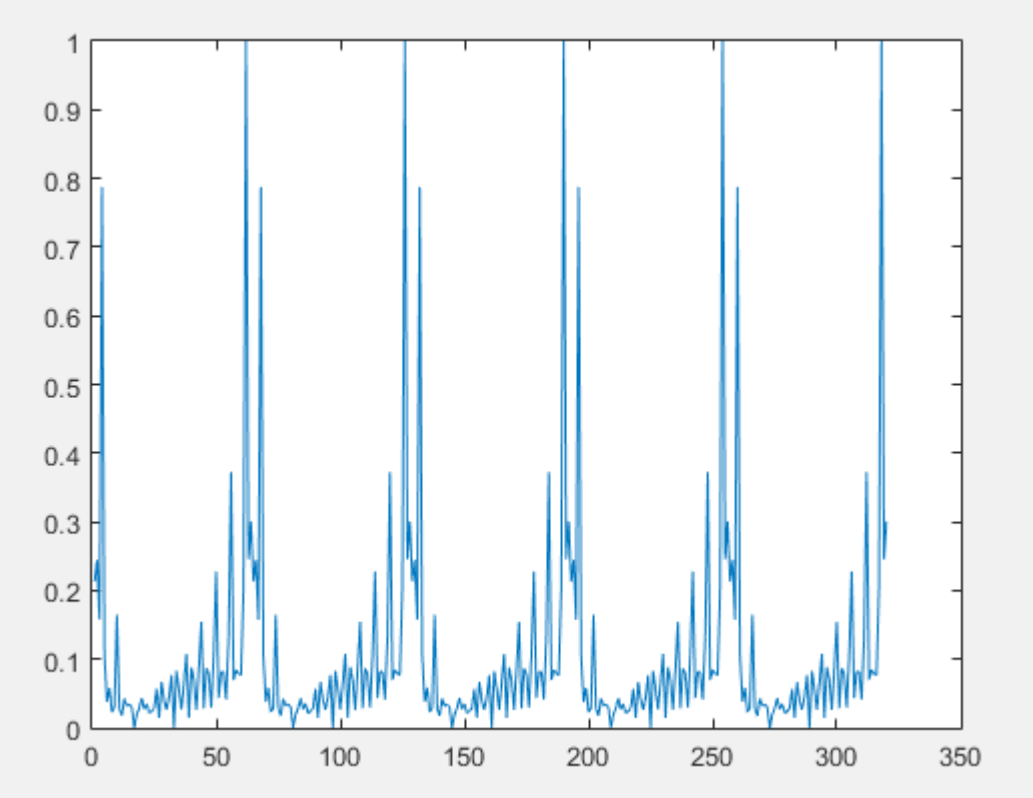
Παρακάτω βλέπουμε ένα τέτοιο παράδειγμα με ορισμένο preamble σήμα σε περιβάλλον matlab.



Σχήμα 17:Πραγματικό μέρος σήματος preamble.



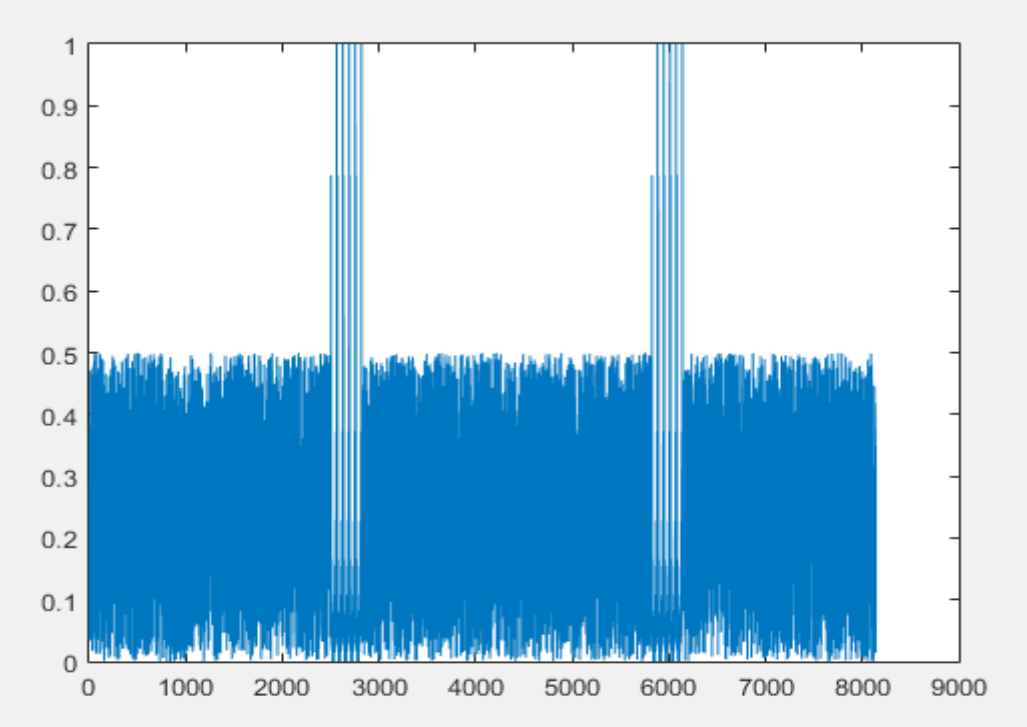
Σχήμα 18:Φανταστικό μέρος σήματος preamble.



Σχήμα 19:Μέτρο σήματος preamble.

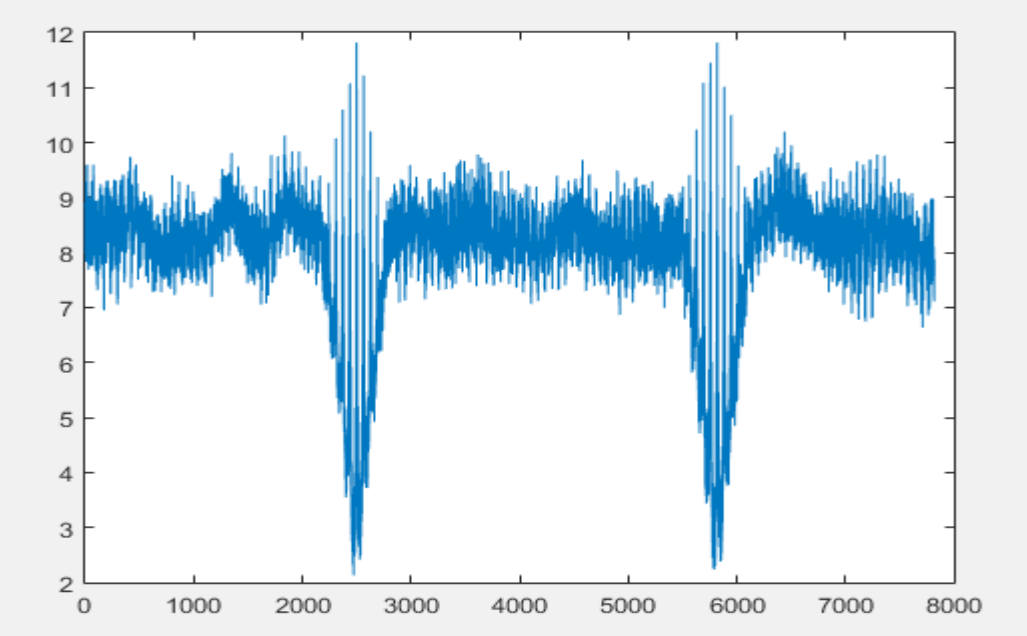
Στα επόμενα πειράματα θα τοποθετήσουμε το παραπάνω σήμα μέσα στον τηλεπικοινωνιακό δίαυλο και στη συνέχεια θα προσπαθήσουμε να το ανιχνεύσουμε κατά την διάρκεια της μετάδοσης.

Αρχικά τοποθετούμε το σήμα μέσα στο δίαυλο και υπολογίζουμε το μέτρο των σημάτων κατα μήκος του διαύλου.Το σήμα αυτό θα αποτελέσει την είσοδο του αλγορίθμου μας και δίνεται στο σχήμα 20.



Σχήμα 20: Σήμα εισόδου για μέθοδο μέτρου χωρίς θόρυβο(M/M0=2).

Στο παραπάνω σχήμα M καλείται το μέτρο του σήματος που αναζητούμε και M0 το μέσο μέτρο των υπολοίπων σημάτων στο δίαυλο.Όπως φαίνεται στην εικόνα έχουμε M/M0=2. Στην συνέχεια θα δούμε ότι εφαρμόζοντας την ετεροσυσχέτιση με πολλαπλασιασμό εμφανίζεται το σήμα που ψάχνουμε. Το αποτέλεσμα της ετεροσυσχέτισης του σήαμτος εισόδου (Σχήμα 20) με το αποθηκευμένο σήμα (σχήμα 19) δίνεται στο σχήμα 21.

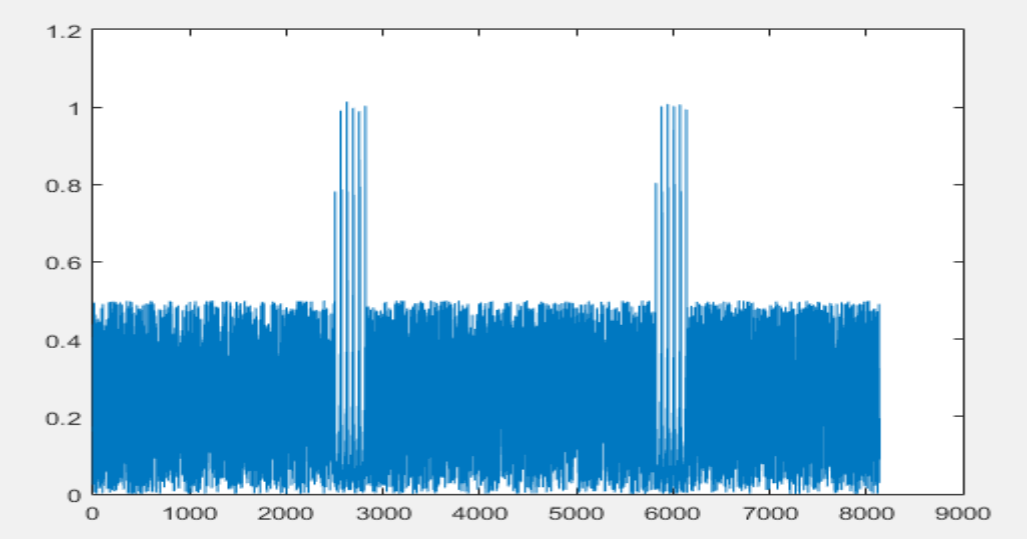


Σχήμα 21:Αποτέλεσμα ετεροσυσχέτισης για μέθοδο μέτρου χωρίς θόρυβο(M/M0=2).

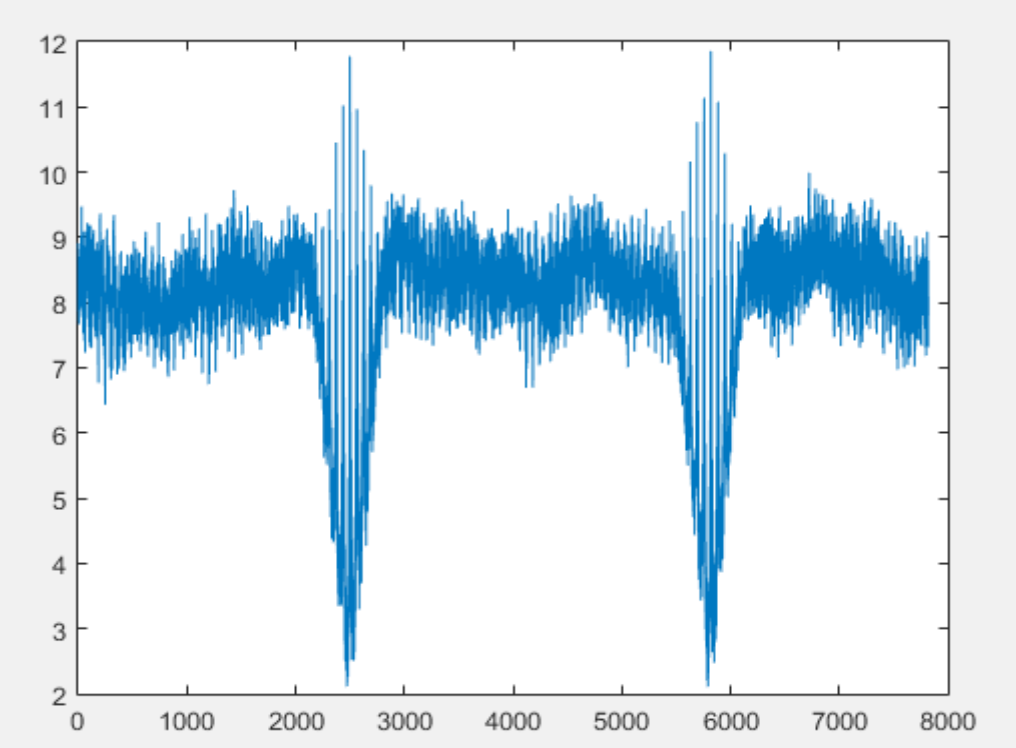
Το σήμα ανιχνεύται καθαρά στα σημεία που εμφανίζονται οι δύο κορυφές.

Δυστυχώς εάν το σήμα μεταφέρονταν σε πραγματικό τηλεπικοινωνιακό δίαυλο οι συνθήκες

δεν θα ήταν τόσο ιδανικές.Στο παραπάνω πείραμα δεν λάβαμε υπόψιν μας τον θόρυβο ο οποίος μπορεί να υπάρχει μέσα στο κανάλι,καθώς και το γεγονός ότι το ζητούμενο σήμα όχι μόνο δεν θα έχει διπλάσιο μέτρο απο τα υπόλοιπα αλλά υπάρχει πιθανότητα ,λόγω εξασθένησης ή συνειδητής επιλογής του πομπού για να εξοικονομήσει φάσμα συχνοτήτων,να έχει το ίδιο η ακόμα και μικρότερο μέτρο από τα υπόλοιπα σήματα.
Κατόπιν προσθέτουμε θόρυβο στο σήμα μέσα στο δίαυλο.

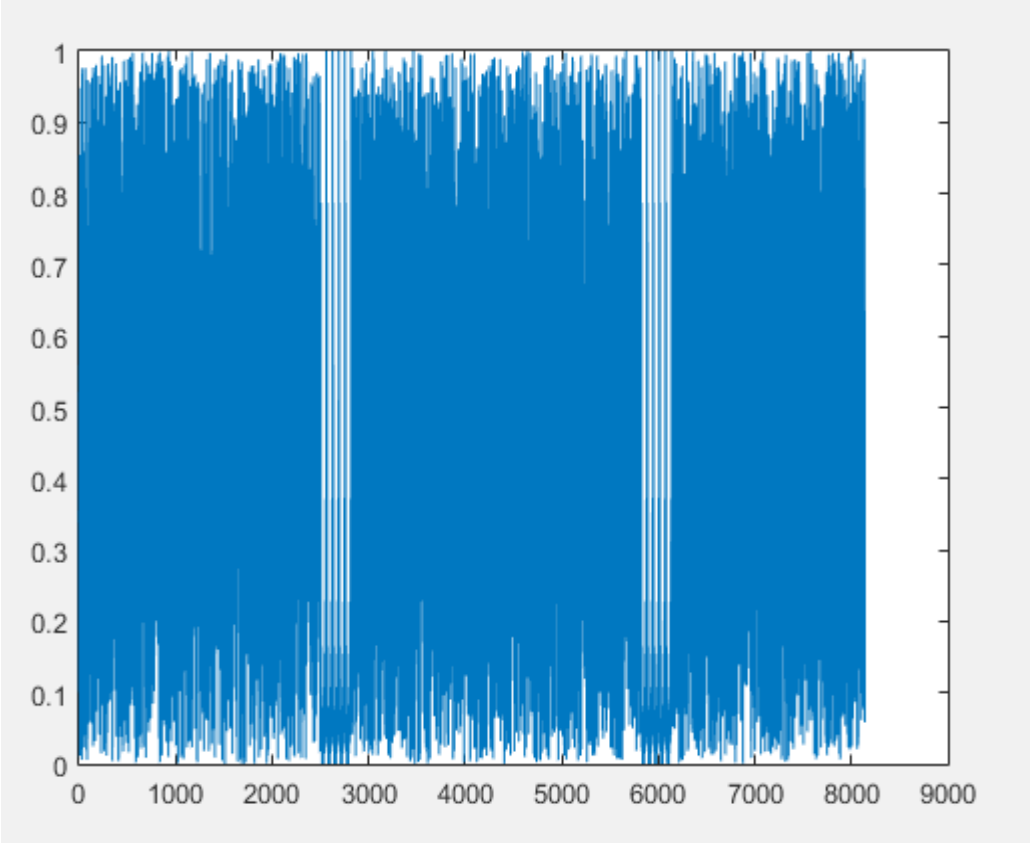


Σχήμα 22:Σήμα εισόδου για μέθοδο μέτρου με θόρυβο 40db(M/M0=2).
Εφαρμόζοντας την συσχέτιση όπως πριν για το νέο σήμα προκύπτει το αποτέλεσμα του σχήματος,



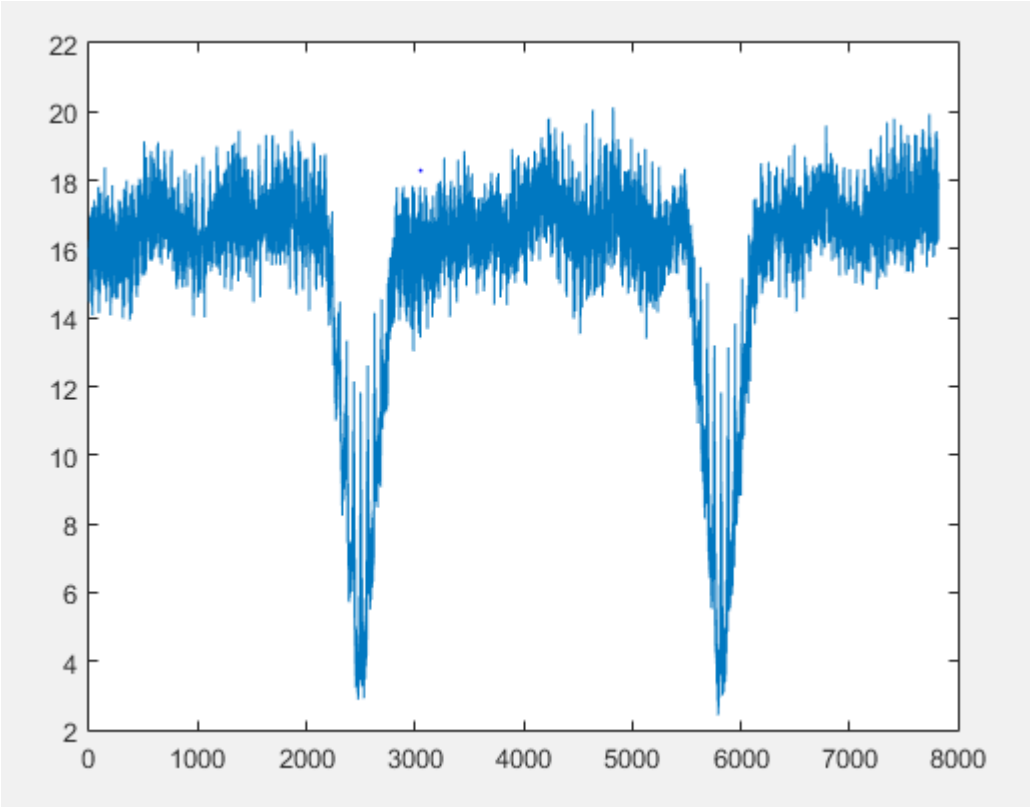
Σχήμα 23:Αποτέλεσμα ετεροσυσχέτισης για μέθοδο μέτρου θόρυβο 40db(M/M0=2).
Είναι φανερό ότι ο θόρυβος μας επηρεάζει το αποτέλεσμα καθώς είναι εμφανής οι

κορυφές.Τώρα θα μελετηθεί η περίπτωση που τα μέτρα όλων των σημάτων είναι ίδια όπως φαίνεται στο σχήμα 24.



<u>Σχήμα 24:Σήμα εισόδου για μέθοδο μέτρου με θόρυβο 40db(M/M0=1).</u>

Σε αυτή την περίπτωση όταν εφαρμόζουμε την συσχέτιση δεν έχουμε τα ίδια αποτελέσματα, όπως φαίνεται στο σχήμα 25.Δεν εμφανίζονται πλεόν οι κορυφές και άρα το σήμα δεν ανιχνεύεται.

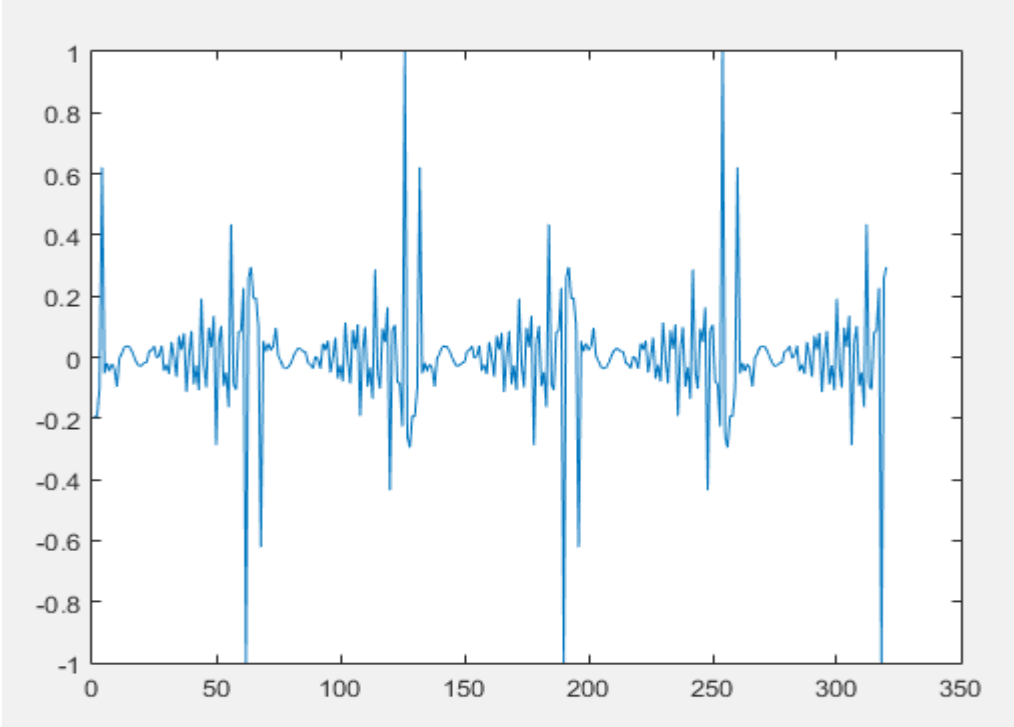


Σχήμα 25:Αποτέλεσμα ετεροσυσχέτισης για μέθοδο μέτρου χωρίς θόρυβο(M/M0=1).

Συμπαιρένουμε ότι όταν τα μέτρα είναι ίδια τότε με την μέθοδο του μέτρου δεν μπορούμε πλέον να ανιχνεύσουμε το σήμα καθώς πλεον,όπως αναφέρθηκε,δεν υπάρχουν κορυφές.Το πρόβλημα φαίνεται να εντοπίζεται στο γεγονός ότι στην περίπτωση των μέτρων έχουμε μόνο θετικούς αριθμούς,οι οποίοι όταν εφαρμόζεται η ετερόσυσχέτιση λειτουργούν μόνο προσθετικά στο άθροισμα.Επόμενος στόχος είναι να διαπιστωθεί εάν η ύπαρξη αρνητικών αριθμών θα βοηθούσε σε αυτήν την περίπτωση.

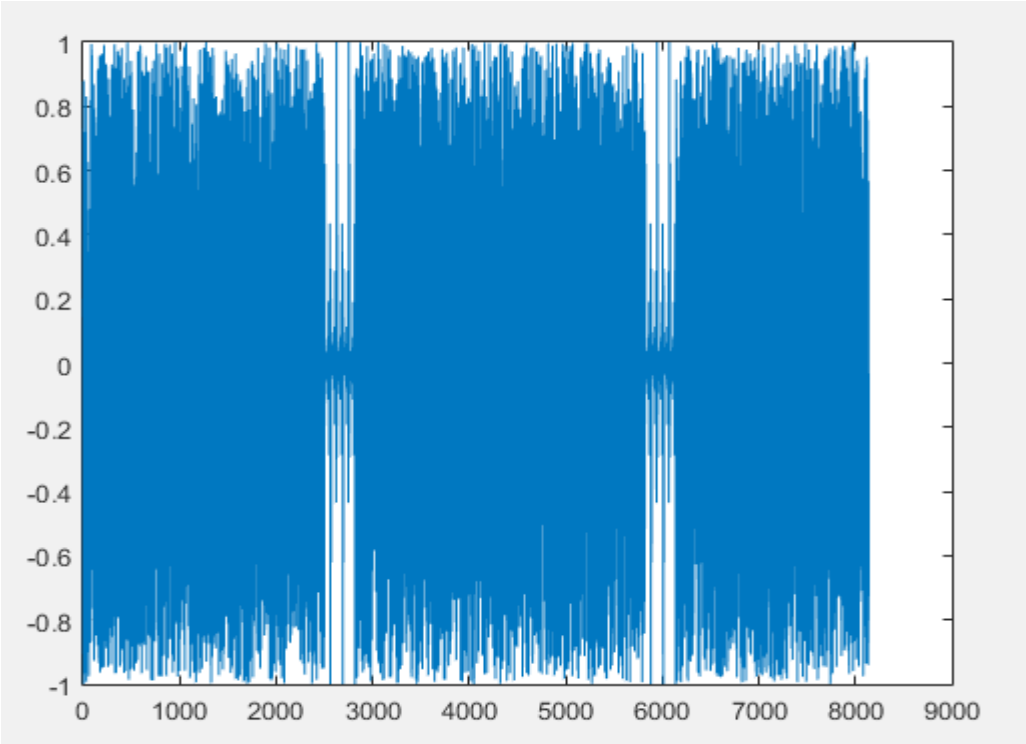
ΚΕΦΑΛΑΙΟ 2-ΑΝΙΧΝΕΥΣΗ ΜΕ ΤΟ ΠΡΑΓΜΑΤΙΚΟ ΜΕΡΟΣ ΣΗΜΑΤΟΣ

Η μέθοδος που εξετάστηκε στο προηγούμενο κεφάλαιο παρουσίασε ικανοποιητικά αποτελέσματα όταν το μέτρου του ζητούμενου σχήματος είναι διπλάσιο,κατά μέσο όρο, από τα μέτρα των υπόλοιπων σημάτων στο δίαυλο.Πρόβλημα παρουσιάστηκε όταν τα μέτρα όλων των σημάτων ήταν ίδια.Για να λυθεί αυτό το πρόβλημα εξετάζεται η εφαρμογή της ετεροσυσχέτισης μόνο στο πραγματικό μέρος του σήματος.



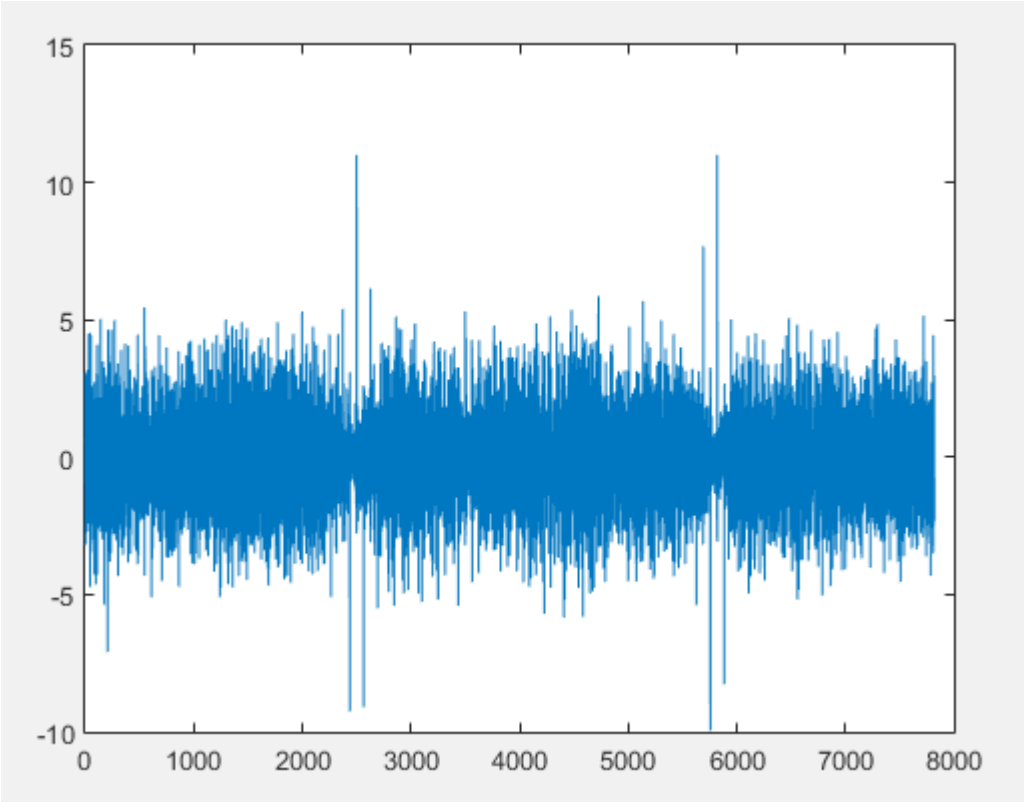
Σχήμα 26:Πραγματικό μέρος preamble.

Τοποθετούμε το σήμα σε τηλεπικοινωνιακό δίαυλο όπως και πριν.(σχήμα 27)



Σχήμα 27:Σήμα εισόδου για μέθοδο πραγματικού μέρους χωρίς θόρυβο (M/M0=1).

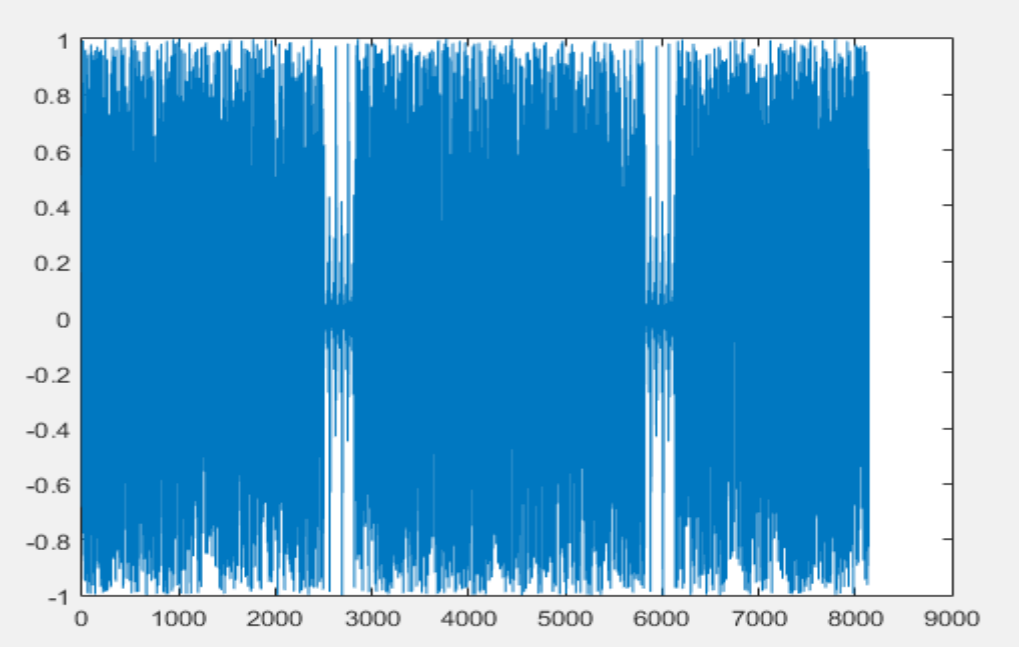
Στην συνέχεια εφαρμόζουμε πάνω στο δίαυλο την ετεροσυσχέτιση με το πραγματικό μέρος του σήματος.Το αποτέλεσμα δίνεται στο σχήμα 28.



Σχήμα 28:Αποτέλεσμα ετεροσυσχέτισης για μέθοδο πραγματικού μέρους χωρίς

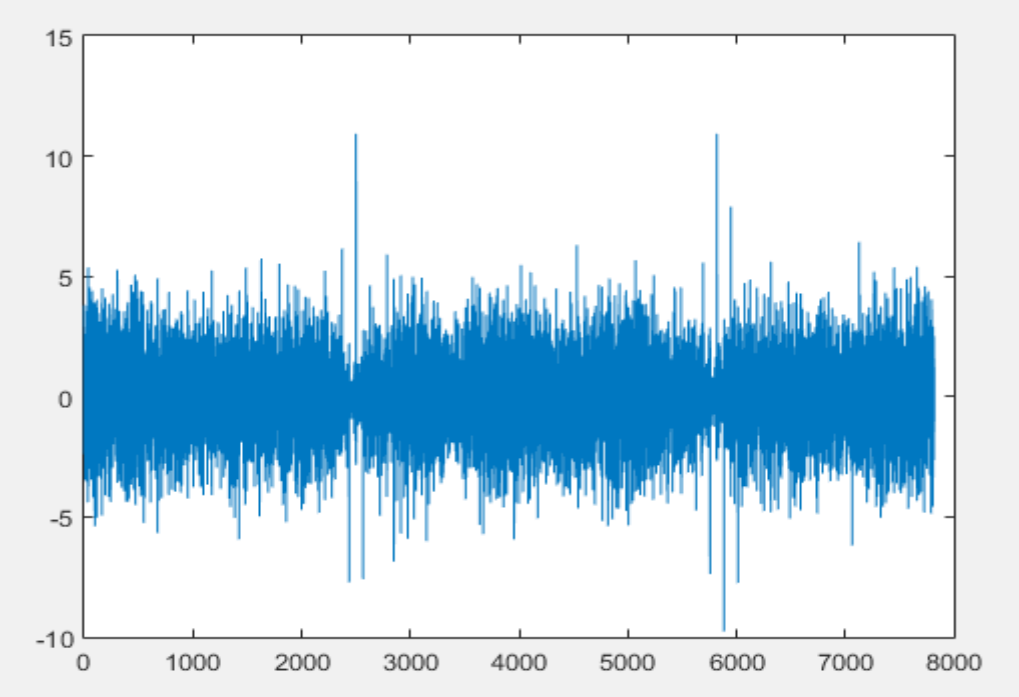
θόρυβο(M/M0=1).

Καθώς είναι εμφανείς οι κορυφές είναι προφανές ότι η ύπαρξη αρνητικών αριθμών διόρθωσε το πρόβλημα της ίσης μέγιστης τιμής ζητούμενου σήματος και υπόλοιπου διαύλου.
Στην συνέχεια προσθέτουμε θόρυβο 40db πάνω στο σήμα μέσα στο δίαυλο έτσι ώστε να προσομοιώσουμε καλύτερα τις πραγματικές συνθήκες.



Σχήμα 29:Σήμα εισόδου για μέθοδο πραγματικού μέρους με θόρυβο 40db (M/M0=1).

Το αποτέλεσμα της ετεροσυσχέτισης δίνεται σ το σχήμα 30

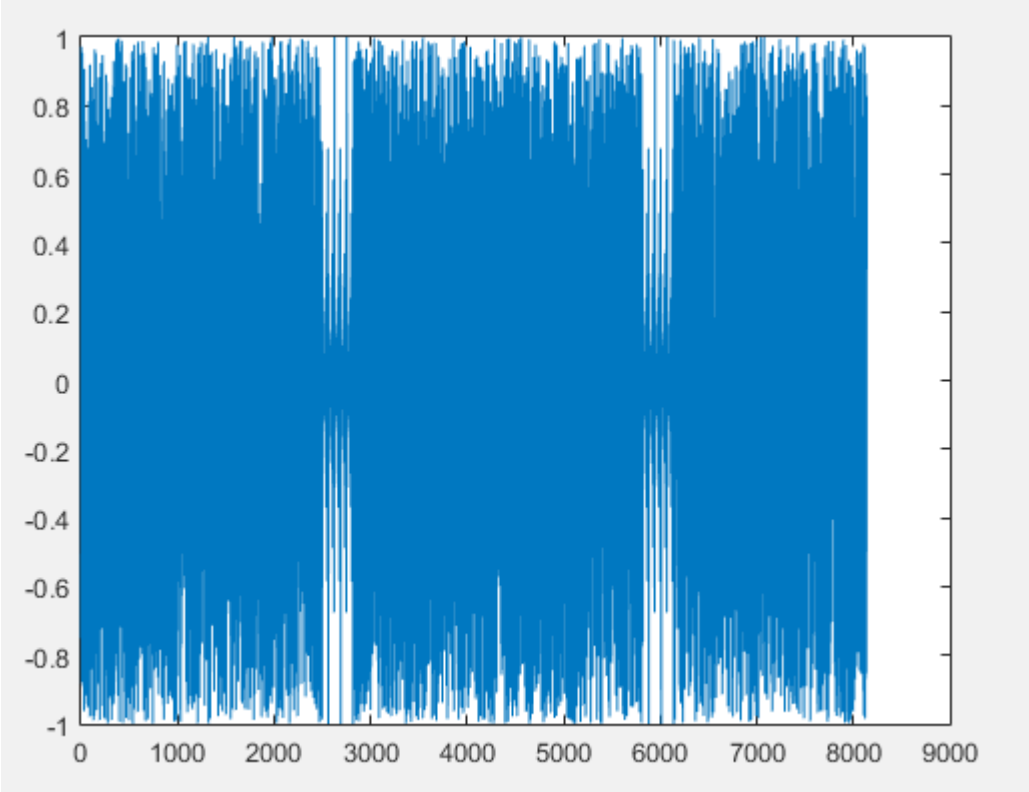


Σχήμα 30:Αποτέλεσμα ετεροσυσχέτισης για μέθοδο πραγματικού μέρους με θόρυβο 40db(M/M0=1).

Οι κορυφές είναι και εδώ εμφανής άρα ο θόρυβος και σε αυτήν την περίπτωση δεν επηρεάζει την πράξη της συσχέτισης.
Κανονικά,σύμφωνα με τα δεδομένα που είχαμε μέχρι τώρα φαίνεται πως το πρόβλημα λύθηκε.Το σήμα μας πλέον μπορεί να ανιχνεύται ακομά όταν κινείται στα ίδια όρια με τα υπόλοιπα σήματα του διαύλου ακόμα και με την παρουσία υψηλού θορύβου.Δυστυχώς αυτό δεν ισχύει στην πραγματική μετάδοση σήματος σε τηλεπικοινωανιακό δίαυλο.Με την εισαγωγή του πραγματικού μέρους στην ετεροσυσχέτιση, το σύστημα έγινε ευάλωτο στην στροφή φάσης που προκαλεί ο δίαυλος στο σήμα κατά την διάρκεια της μετάδοσής του.

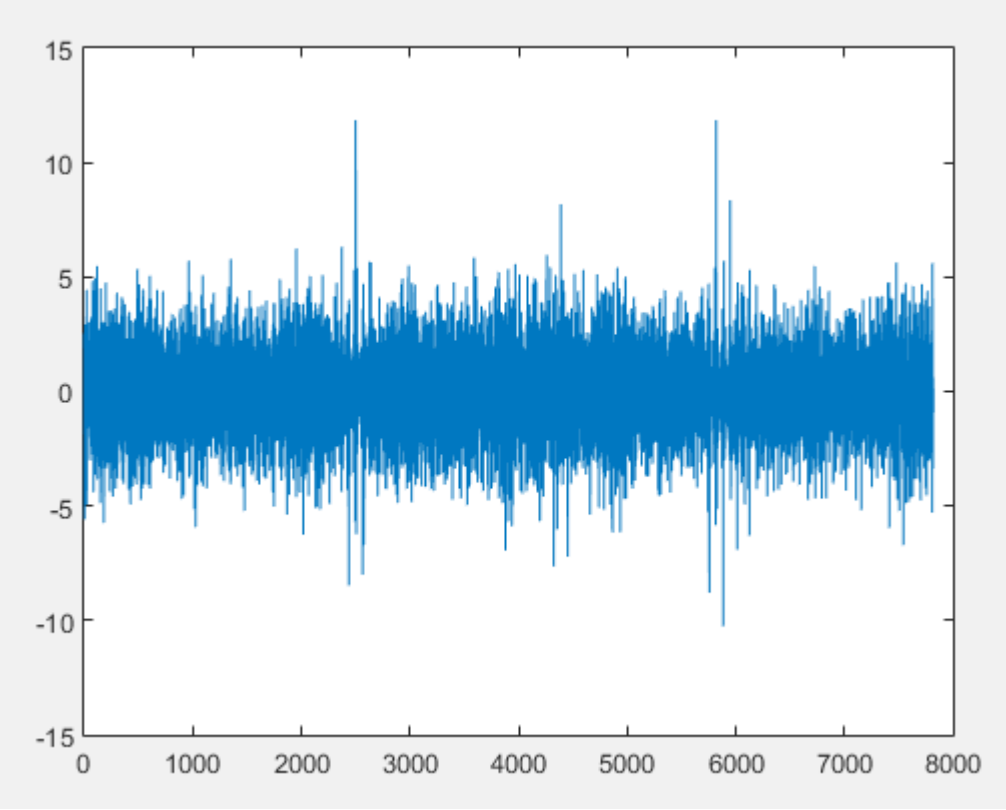
$$S'(t)=e^{j\theta}*S(t)$$

Όπου S(t) είναι το αρχικό σήμα και S'(t) είναι το σήμα μετά την μετάδοση.Προφανώς το μέτρο το S(t) δεν αλλάζει(αυτός είναι και ο λόγος που στην μέθοδο του μέτρου δεν εμφανίστηκε το πρόβλημα)αλλά αλλάζει πραγματικό και φανταστικό μέρος του S(t). Αυτό επαληθεύται στα σήματα που φαίνονται στα παρακάτω σχήματα, όπου προκαλέσαμε στροφή φάσης θ=35 και θ=55 μοιρών αντίστοιχα ώστε να μελετήσουμε την επίδραση πάνω στο αποτέλεσμα της συσχέτισης.
Αρχίζουμε στρέφοντας το σήμα κατά 35 μοίρες και μεταδίδοντας το μέσα σε δίαυλο(σχήμα 31).



Σχήμα 31:Σήμα εισόδου για μέθοδο πραγματικού μέρους με στροφή φάσης 35 μοιρών (M/M0=1).

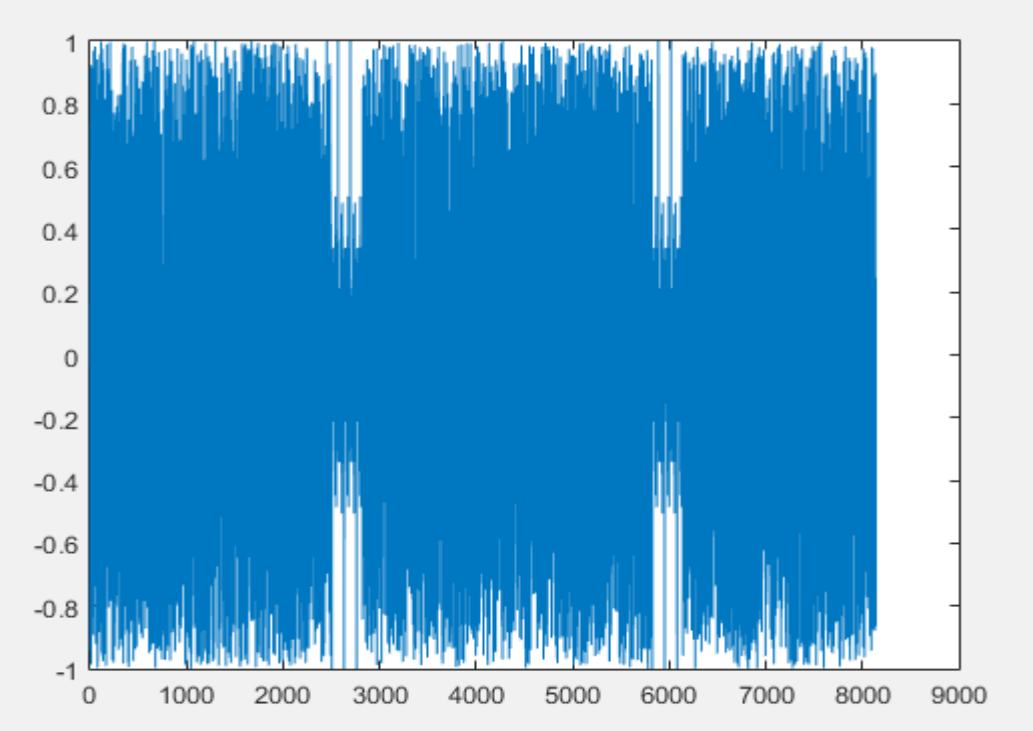
Το αποτέλεσμα της ετεροσυσχέτισης φαινεται στο σχήμα 32.



Σχήμα 32:Αποτέλεσμα ετεροσυσχέτισης για μέθοδο πραγματικού μέρους με στροφή φάσης 35 μοιρών(M/M0=1).

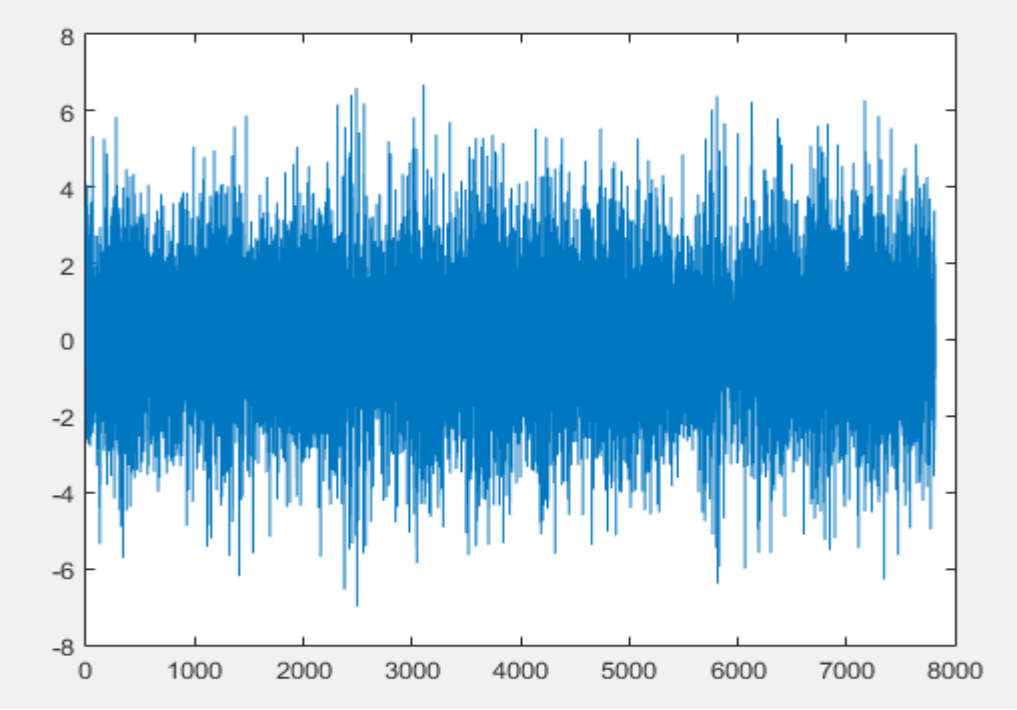
Για 35 μοίρες η στροφή φάσης δεν επηρεάζει την ανίχνευση.Οι δύο κορυφές φαίνονται καθαρά.

Ακολουθούμε την ίδια ακριβώς διαδικασία για 55 μοίρες.



Σχήμα 33:Σήμα εισόδου για μέθοδο πραγματικού μέρους με στροφή φάσης 55 μοιρών (M/M0=1).

Το αποτέλεσμα της ετεροσυσχέτισης φαινεται στο σχήμα 34.Δεν εμφανίζονται διακρητές κορυφές.



Σχήμα 34:Αποτέλεσμα ετεροσυσχέτισης για μέθοδο πραγματικού μέρους με στροφή φάσης 35 μοιρών(M/M0=1).

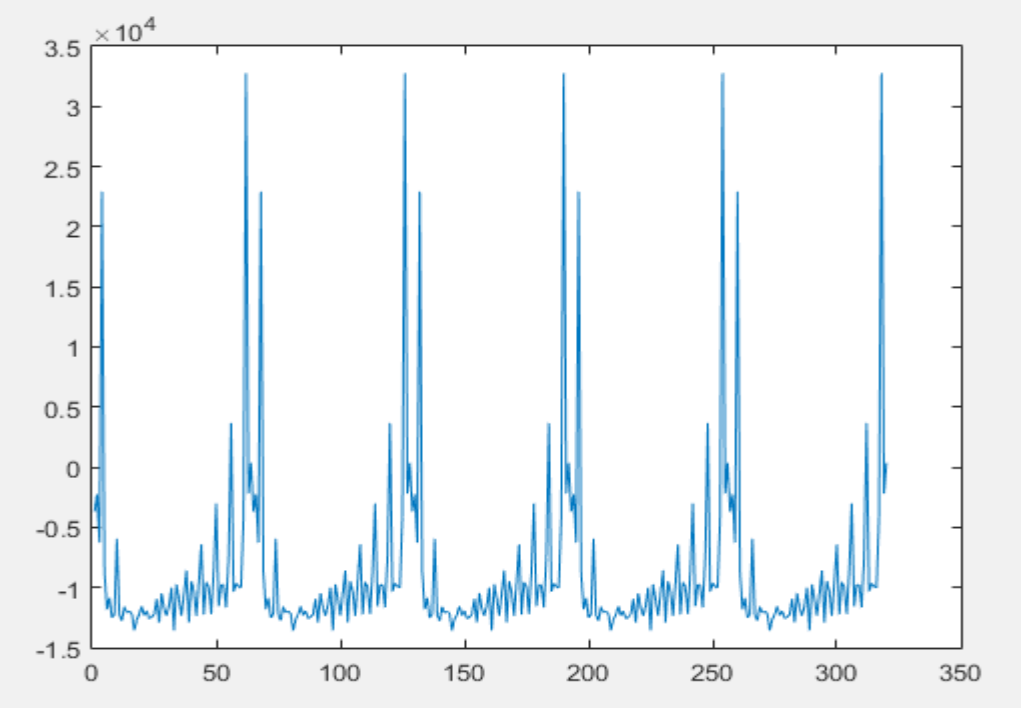
Δυστυχώς για 55 μοίρες η επίδραση πάνω στην ετεροσυσχέτιση είναι πλέον αποτρεπτική για την εξαγωγή σωστών συμπερασμάτων.
Για άλλη μια φορά οδηγούμαστε σε αδιέξοδο.Η μέθοδος με το πραγματικό μέρος διόρθωσε το πρόβλημα που υπήρχε στις ίσες στάθμες παρουσίασε όμως άλλο πρόβλημα και την καθιστά μη κατάλληλη για γενική χρήση.Βασικό μέλημα μας είναι με κάποιο τρόπο να βρίσκουμε σωστό αποτέλεσμα ανεξαρτήτως στάθμης σήματος και στροφής φάσης.Μοναδική λύση στο πρόβλημα είναι ο συνδυασμός των δύο παραπάνω μεθόδων.

ΚΕΦΑΛΑΙΟ 3-ΑΝΙΧΝΕΥΣΗ ΜΕ ΣΥΝΔΥΑΣΜΟ

Η μέθοδος που τελικά επιλέχθηκε και υλοποιήθηκε είναι ο συνδυασμός των δύο προηγούμενων μεθόδων.Ουσιαστικά

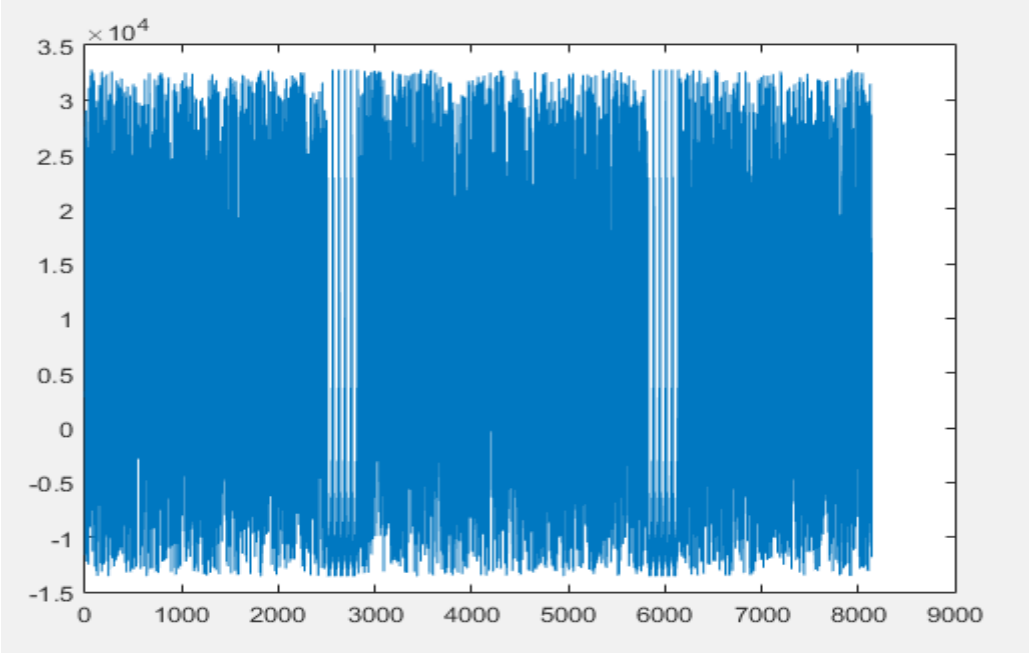
-Χρησιμοποιούμε το μέτρο του σήματος για να αποφύγουμε την στροφή φάσης απο τον δίαυλο.

-Αφαιρούμε από το μέτρο μία ποσότητα σύμφωνα με κάποιο κριτήριο ώστε το σήμα να αποκτήσει αρνητικές τιμές ώστε να μην περιοριζόμαστε από την στάθμη του σήματος μέσα στον δίαυλο.Με δοκίμες παρατηρήθηκε ότι η τιμή που θα πρέπει να αφαιρεθεί θα πρέπει να είναι τέτοια, έτσι ώστε το ολοκλήρωμα του σήματος στον χρόνο να είναι 0(μέση τιμή).Εξάγουμε επομένως το μέτρο και αφαιρούμε την τιμή που χρειάζεται



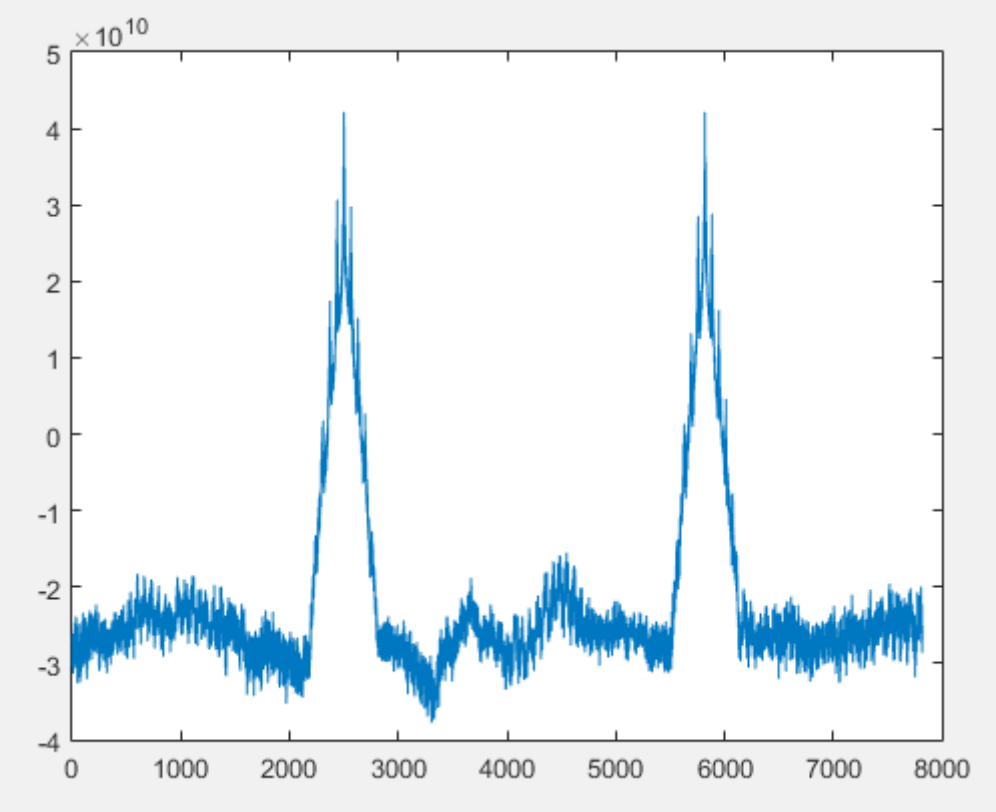
Σχήμα 35:σήμα προς αναζήτηση(preamble).

Το σήμα τοποθετείται στο δίαυλο με M=M0.



Σχήμα 36:Σήμα εισόδου για μέθοδο συνδυαμού χωρίς θορύβο (M/M0=1).

Εφαρμόζουμε την ετεροσυσχέτιση.Οι κορυφές είναι εμφανής.

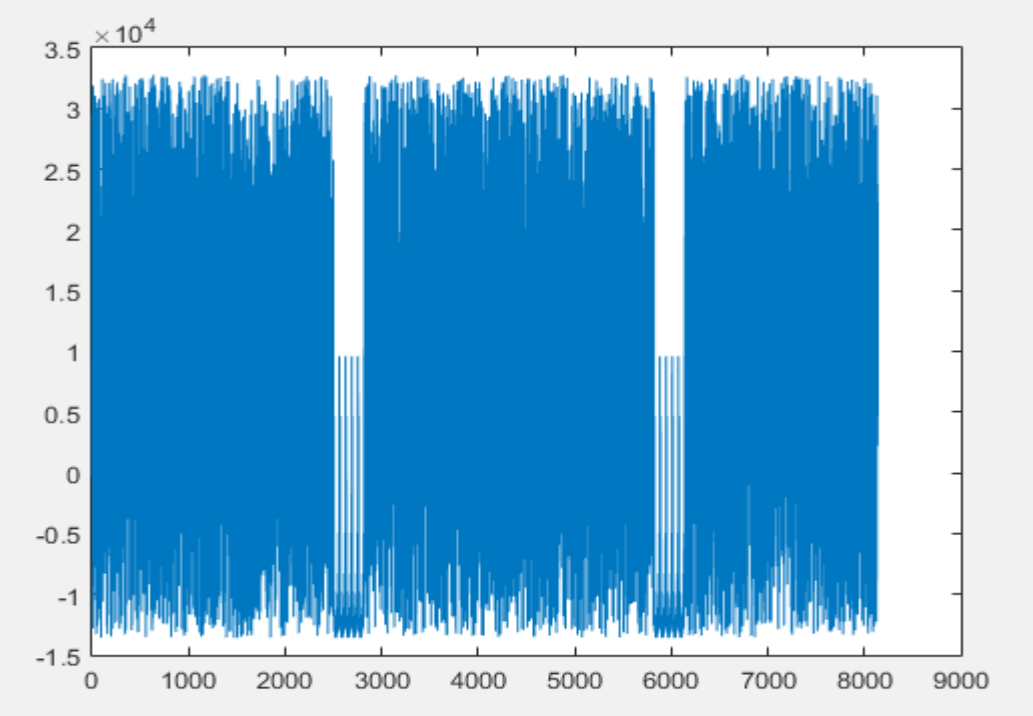


Σχήμα 37:Αποτέλεσμα ετεροσυσχέτισης για μέθοδο συνδυασμού χωρίς θόρυβο μοιρών(M/M0=1).

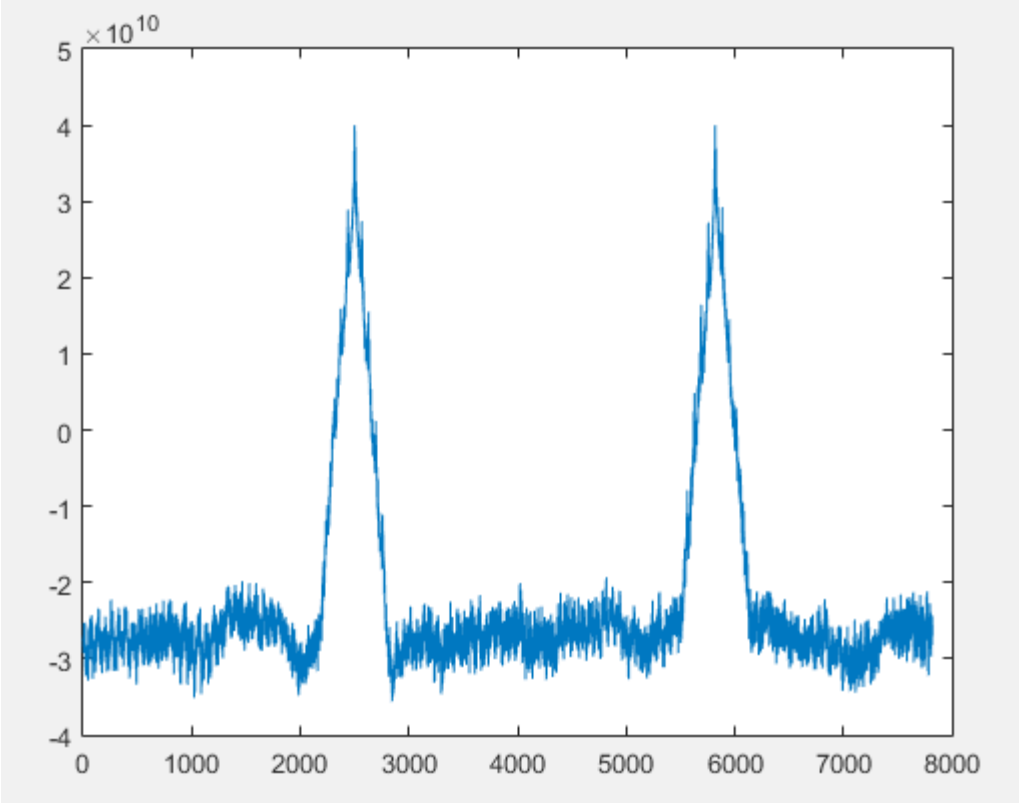
Είναι ξεκάθαρο ότι το σήμα ανιχνεύεται.

Στροφή φάσης δεν υπάρχει καθώς χρησιμοποιείται το μέτρο του σήματος συνεπώς το μόνο που

απομένει είναι να ελεγχθεί είναι αν το σήμα ανιχνεύται σε πραγματικές συνθήκες. Προσθίτεται θόρυβος 40db και μειώνεται το μέτρο στο μισό (απόσβεση απόστασης) όπως φαίνεται στο σχήμα 38.



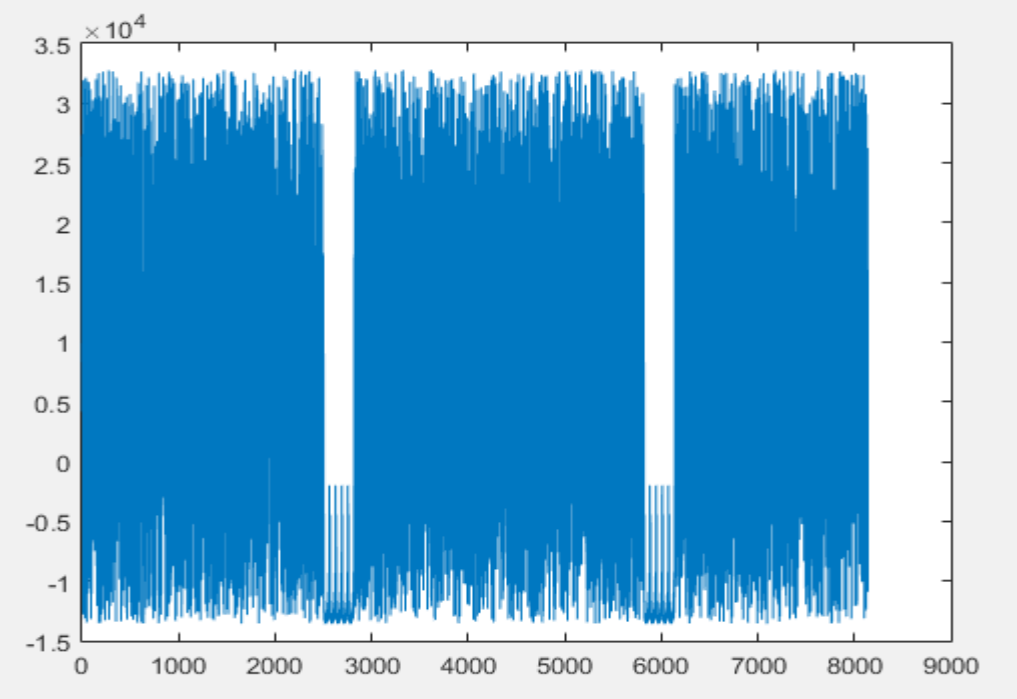
Σχήμα 38:Σήμα εισόδου για μέθοδο συνδυαμού με θόρυβο 40db (M/M0=1/2).



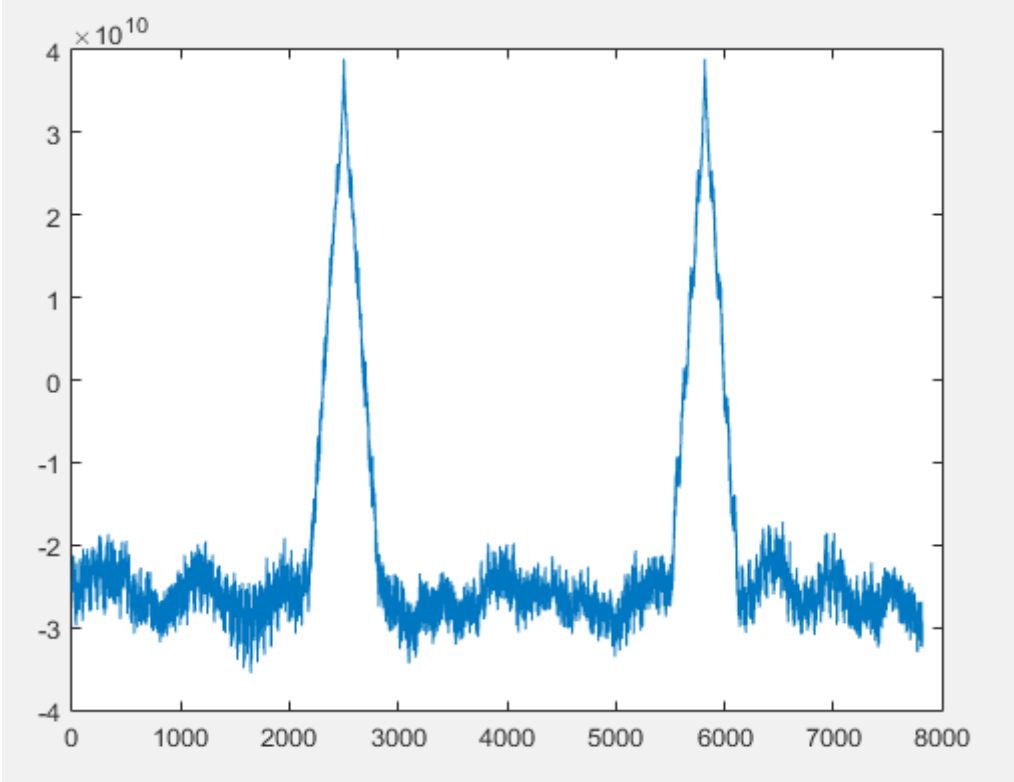
Σχήμα 39:Αποτέλεσμα ετεροσυσχέτισης για μέθοδο συνδυασμού με θόρυβο

40db(M/M0=1/2).

Το σήμα ανηχνεύεται καθαρά ακόμα και με μικρότερη στάθμη.Μικραίνουμε ακόμα περισσότερο την στάθμη του σήματος στο 1/4.



Σχήμα 40:Σήμα εισόδου για μέθοδο συνδυαμού με θόρυβο 40db (M/M0=1/4).



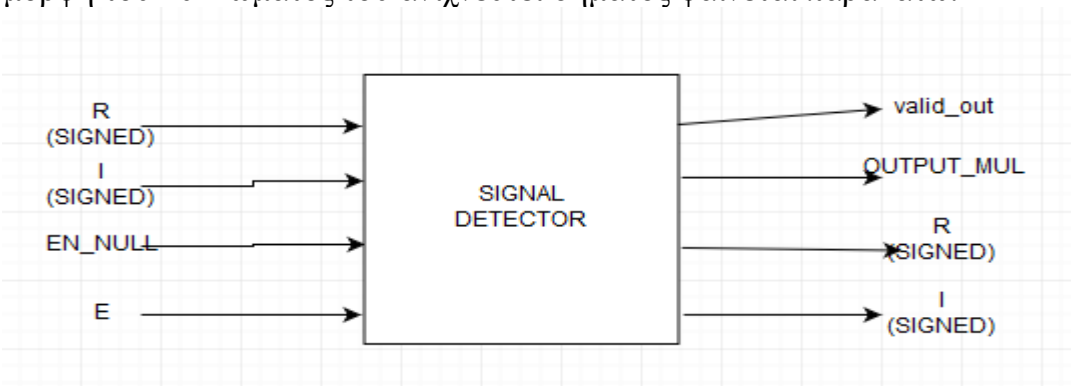
Σχήμα 41:Αποτέλεσμα ετεροσυσχέτισης για μέθοδο συνδυασμού με θόρυβο 40db(M/M0=1/4).

Τα αποτελέσματα είναι πολύ θετικά καθώς με το 1/4 της στάθμης τα σήματα ανιχνεύονται καθαρά.
Επιλέγουμε επομένως να υλοποιήσουμε αυτήν την μέθοδο πάνω στο υλικό του fpga.Στο επόμενο κεφάλαιο θα περιγράψουμε σταδιακά την υλοποίηση κάθε τμήματος της αρχιτεκτονικής καθώς και τον τρόπο με τον τρόπο με τον οποίο όλα τα τμήματα συνδέονται έτσι ώστε να αποτελέσουν ένα λειτουργικό όλον.

ΔΕΥΤΕΡΟ ΜΕΡΟΣ:ΥΛΟΠΟΙΗΣΗ

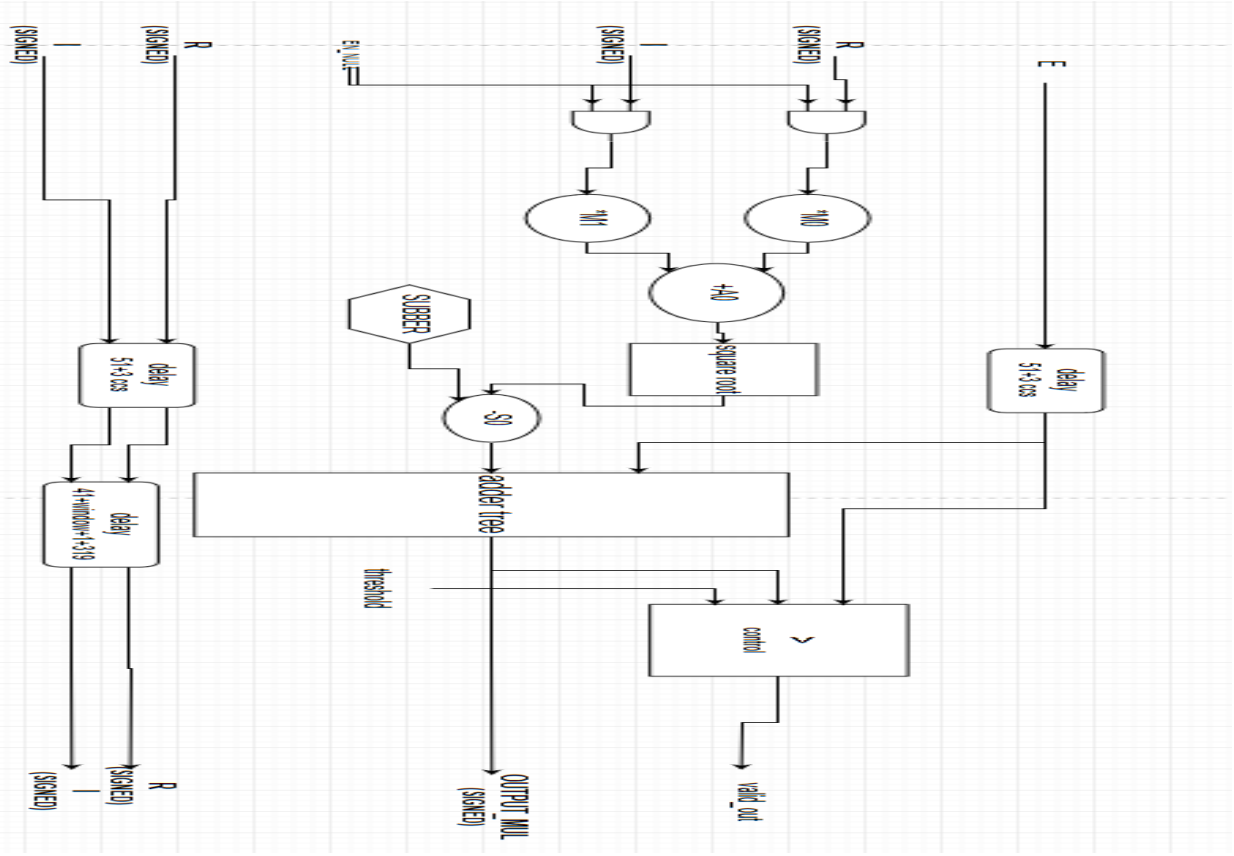
ΚΕΦΑΛΑΙΟ 4-ΓΕΝΙΚΟ ΚΥΚΛΩΜΑ

Η γενική μορφή του κυκλώματος του ανιχνεύτει σήματος φαίνεται παρακάτω:



Σχήμα 42:Μπλοκ γενικού κυκλώματος.

R:είναι το πραγματικό μέρος του σήματος εισόδου(16 bit)
I:είναι το φανταστικό μέρος του σήματος εισόδου(16 bit)
EN_NULL:είσοδος που μας δίνει την δυνατότητα να μηδενίσουμε τις εισερχόμενες τιμές στα R,I.Είναι χρήσιμη για να καθαρίσουμε την μνήμη του κυκλώματος.(1 bit)
E:είναι η επίτρεψη του κυκλώματος.(1 bit)
valid_out:έγκυρη έξοδος(1 bit)
OUTPUT_MUL:αποτέλεσμα ετεροσυσχέτισης πολλπλασιασμού σε συκγεκριμένο σημείο (41 bits για σήματα 320 σημείων)
R,I (στην έξοδο):είναι η αναμετάδοση του σήματος που λαμβάνεται απο τον τηλεπικοινωνιακό δίαυλο.Όταν ο ανιχνευτής το εντοπίσει σηκώνει απλά το valid_out στο σημείο όπου αναμεταδίδεται το ζητούμενο σήμα.
Το σχήμα στην επόμενη σελίδα αποτελεί την αρχιτεκτονική του κυκλώματος.Αναλυτικά έχουμε



Σχήμα 43:Γενικό κύκλωμα σε πιο αναλυτική μορφή.

Εσωτερικά του κυκλώματος,σε μια πιο αναλυτική αναπαράσταση

-R,I είναι το πραγματικό και το φανταστικό μέρος της μιγαδικής αναπαράστασης του σχήματος.Και οι δύο ποσότητες είναι προσημασμένοι αριθμοί.

-E είναι η επίτρεψη(enable) του κυκλώματος.Όταν έχει την τιμή 0 το κύκλωμα παγώνει.

-EN_NULL είναι μια είσοδος που μας δίνει την δυνατότητα να μηδενίσουμε τις εισόδους R,I που εισέρχονται στο κύκλωμα όταν της δίνουμε την τιμή 0.

-*M0 και *M1 είναι οι πολλαπλασιαστές από τους οποίους παίρνουμε τις ποσότητες R^2 και I^2 .

-A0 είναι ο αθροιστής $R^2 + I^2$.

-S0 είναι ο αφαιρέτης τον οποίο χρησιμοποιούμε για να αφαιρέσουμε από το μέτρου του μιγαδικού την προκαθορισμένη τιμή που βρίσκεται στον καταχωρητή SUBBER.

-Οι καταχωρητές καθυστέρησης (delay blocks) προκαλού καθυστέρηση αντίστοιχων κύκλων ,απαραίτητη για τον συγχρονισμό των διάφορων τμημάτων του κυκλώματος.

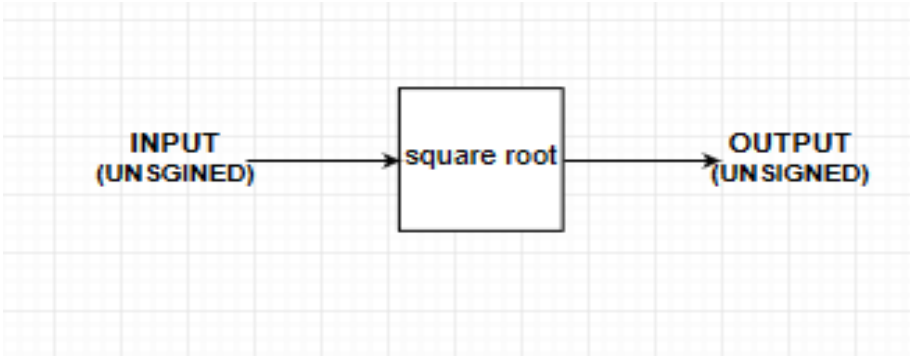
-Το κύκλωμα SQUARE ROOT υπολογίζει την τετραγωνική ρίζα της ποσότητας του A0 δηλαδή την ποσότητα $\sqrt{R^2+I^2}$.

-ADDER TREE ονομάζεται η λειτουργική μονάδα που αναλαμβάνει την μαθηματική υλοποίηση της πράξης της συσχέτισης.

-Τέλος η μονάδα CONTROL ανιχνεύει την ύπαρξη κορυφών και ανάλογα προσαρμόζει την τιμή valid_out έτσι ώστε να είναι 1 όταν αναμεταδίδεται το ζητούμενο σήμα.

ΚΕΦΑΛΑΙΟ 5-ΤΕΤΡΑΓΩΝΙΚΗ ΡΙΖΑ

Πρώτο μεγάλο ξεχωριστό τμήμα της συνολικής αρχιτεκτονικής είναι ο υπολογισμός της τετραγωνικής ρίζας που λαμβάνει χώρα στην λειτουργική μονάδα square root.



Σχήμα 44:Μπλοκ διάγρμμα της μονάδας υπολογισμού ρίζας.

Ευθύνη αυτής της λειτουργικής μονάδας είναι ο υπολογισμός του μέτρου της μιγαδικής αναπαράστασης του σήματος.Βασική στόχοι μας κατά την ανάπτυξη και την υλοποίηση το εξαρτήματος είναι ο περιορισμός όσο το δυνατόν των πόρων που καταναλώνονται πάνω στο fpga ,και η σωστή λειτουργία της δομής στα 500Mhz.

Η υλοποίηση της τετραγωνικής ρίζας έγινε χρησιμοποιώντας την τεχνική του digit-by-digit calculation.Όπως φαίνεται παρακάτω μπορούμε να υπολογίσουμε την ρίζα ενός δυαδικού αριθμού(unsigned) χρησιμοποιώντας αρχικά διαδοχικούς πολλπλασιασμούς.

Έστω ότι έχουμε αριθμό με bits α1 a2 a3 a4 ……. aN όπου N=2*κ και θέλουμε να βρούμε την ρίζα του με bits b1 b2 b3 b4 ……. bκ ,εργαζόμαστε ως εξής:
-χωρίζουμε ανά 2 bits το αρχικό αριθμό
-σε κάθε ζευγάρι αντιστοιχίζουμε ένα bit από την ρίζα

b1 b2 ……. bκ
α1 a2 | a3 a4| ……. aN

-σε κάθε επανάληψη i υποθέτουμε ότι γνωρίζουμε ότι το ψηφίο της ρίζας είναι 1 και ελέγχουμε αν το τετράγωνο της ρίζας ξεπερνάει τον γνωστό αριθμό, συγκρίνοντας το τετράγωνο της ποσότητας
b1 b2 ……. bi με α1 a2 | a3 a4| ……. a2*i .

b1 b2 ……. bi υποσύνολο του b1 b2 ……. bκ

α1 α2 | α3 α4| ……. α2*i υποσύνολο του α1 a2 | a3 a4| ……. aN

-αν το τετράγωνο είναι μεγαλύτερο τότε το ψηφίο δεν γίνεται να είναι 1 αλλά 0 καθώς σε διαφορετική περίπτωση ο αριθμός B θα ξεπέρναγε τον αριθμό A ήδη από το ψηφίο i.
b1 b2 ……. bi * b1 b2 ……. bi > α1 a2 | a3 a4| ……. a2*i

τότε σίγουρα b1 b2 ….... bκ >α1 a2 | a3 a4| ….... aN που είναι αδύνατο

-ακολουθούμε την ίδια διαδικασία για όλα τα bits και φτάνουμε στην λύση μέγιστης ακρίβειας

Παραδείγματα
Τετράγωνο:01010001 (81 unsigned)
Ρίζα:1001(9 unsigned)

-1ο βήμα:
 1
01|01|00|01

για το πρώτο ζευγάρι ψηφίων επιλέγουμε 1 και κάνουμε την σύγκριση 01*01>01.Αν η συνθήκη είναι αληθής τότε το 1 γίνεται 0 ,αν δεν ισχύει η συνθήκη τότε το 1 κατοχυρώνεται σαν τον πρώτο ψηφίο της ρίζας και συνεχίζουμε στο βήμα 2.

-2ο βήμα:
 1 1
01|01|00|01

για το δεύτερο ζευγάρι ψηφίων έχουμε 11*11>=0101=>1001>=0101.Βλέπουμε καθαρά ότι η συνθήκη είναι ψευδής άρα το δεύτερο ψηφίο κατοχυρώνεται ως 0.

 1 0
01|01|00|01

-3ο βήμα:
 1 0 1
01|01|00|01

101*101>=010100=>011001>=010100 ψευδής

 1 0 0
01|01|00|01

-4ο βήμα:
 1 0 0 1
01|01|00|01

1001*1001>=01010001(ισχύει η ισότητα) συνεπώς το 1 κατοχυρώνεται

 1 0 0 1
01|01|00|01

Τετράγωνο:11010011 (211 unsigned)
Ρίζα:1110(14 unsigned)

-1ο βήμα:
 1
11|01|00|11

-2ο βήμα:
 1 1
11|01|00|11

11*11>=1101=>1001>=1101 ψευδής άρα το 1 κατοχυρώνεται

-3ο βήμα:
 1 1 1
11|01|00|11

111*111>=110100=> 110001>=110100 ψευδής άρα το 1 κατοχυρώνεται

-4ο βήμα:
 1 1 1 1
11|01|00|11

1111*1111>=11010011=>11100001>=11010011 αληθής άρα το 1 γίνεται 0

 1 1 1 0
11|01|00|11

Τετράγωνο:10110001 (177 unsigned)
Ρίζα:1101(13 unsigned)

-1ο βήμα:
 1
10|11|00|01

-2ο βήμα:
 1 1
10|11|00|01

11*11>=1101=>1001>=1011 ψευδής άρα το 1 κατοχυρώνεται

-3ο βήμα:
1 1 1
10|11|00|01

111*111>=110100=> 110001>=101100 αληθής άρα το 1 γίνεται 0

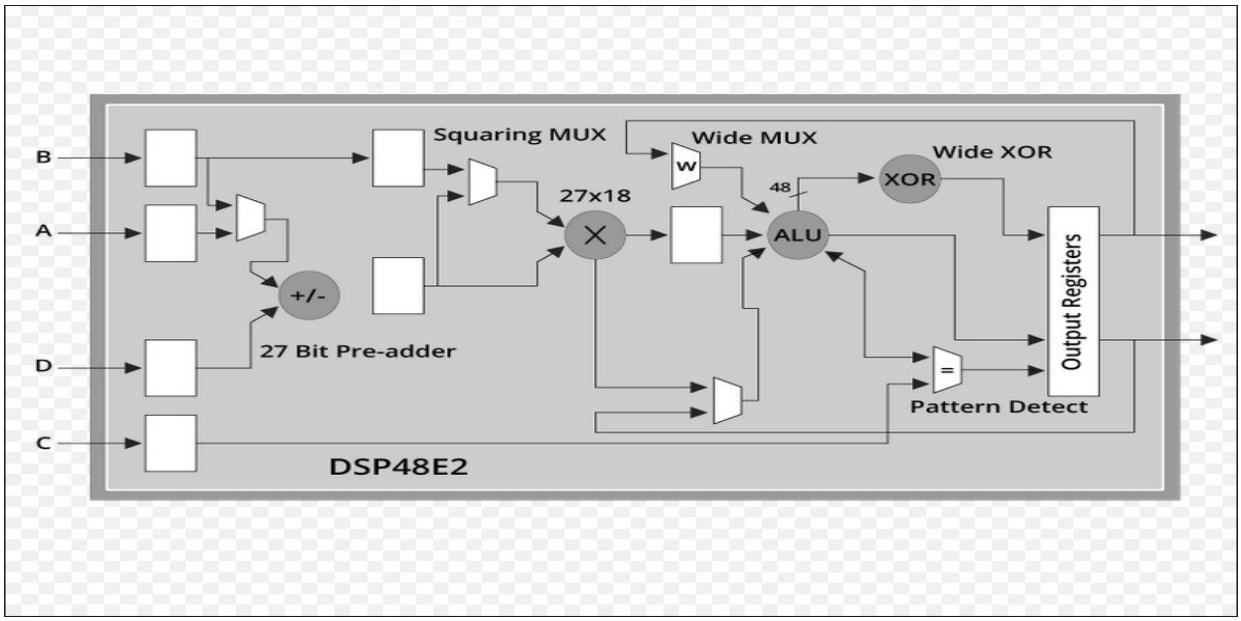
1 1 0
10|11|00|01

-4ο βήμα:
1 1 0 1
10|11|00|01

1111*1111>=11010011=>10101001>=10110001 ψευδής άρα το 1 κατοχυρώνεται

5.1 Προβλήματα βέλτιστης υλοποίησης για λειτουργία στα 500 Mhz

Όπως γνωρίζουμε βασικό προαπαιτούμενο για να λειτουργεί σωστά το module στο περιβάλλον του τηλεπικοινωνιακού συστήματος είναι η επίτευξη σωστής λειτουργίας στα 500 Mhz.Βασικό εμπόδιο για την επίτευξη αυτής της συχνότητας είναι η πράξη του πολλαπλασιασμού.Για το σκοπό αυτό, το fpga διαθέτει ειδικές μονάδες επεξεργασίας DSP οι οποίες είναι σχεδιασμένες ώστε να εκτελούν πολλαπλασιασμούς με τον βέλτιστο δυνατό τρόπο ώστε να είναι δυνατή η λειτουργία σε μεγάλες συχνότητες.Πιο κάτω φαίνεται η δομή του DSP48E2 .



Σχήμα 45:Δομή DSP48E2.

Δυστυχώς ο αγλόριθμος που περιγράφτηκε παραπάνω στην αρχική μορφή του περιέχει πάρα πολλούς πολλαπλασιασμούς γεγονός που θα οδηγήσει στην κατανάλωση πολλών DSPs.Πρέπει με κάποιο τρόπο να μειώσουμε δρασικά των αριθμό τον πολλαπλασιασμών ώστε να μειωθεί ο

αριθμός των DSPs που απαιτώνται για την σωστή λειτουργία του κυκλώματος.Παρατηρώντας την σχέση μεταξύ μεταξύ τον γινομένων στην επανάληψη j και στην επανάλψη j+1 καταλήγουμε στο συμπέρασμα ότι το πρόβλημα μπορεί να λυθεί χωρίς πολλαπλασιασμούς αλλά μονάχα με ολισθήσεις και προσθέσεις.

5.2 Βέλτιστη υλοποίηση ρίζας για λειτουργία στα 500 Mhz

Η βέλτιστη υλοποίηση της μονάδας της ρίζας συνίσταται στην απάλειψη του πολλαπλασιασμού και στην αντικατάστασή του από προσθέσεις και ολισθήσεις.

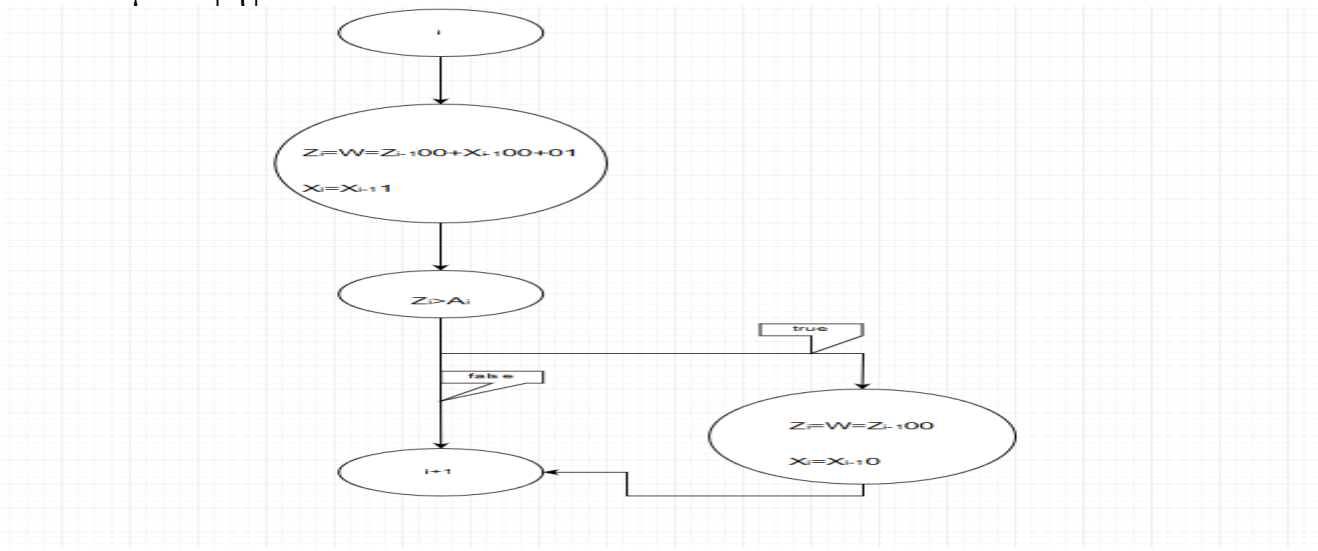
Έστω X αριθμός n bits με τετράγωνο Z=X*X.Θέλουμε να υπολογίσουμε το γινόμενο (X0+01)*(X0+01) με δεδομένο Z (όπως προκύτει όταν προσπαθούμε να υπολογίσουμε την επόμενη επανάληψη του αλγόριθμου της ρίζας).Έχουμε

W=(X0+01)*(X0+01)=X0*X0+X0*01+X0*01+01=Z00+X00+01

Αν το επόμενο ψηφίο είναι 0 τότε έχουμε απλά

W=Z00

Όπως πρατατηρούμε ,για να υπολογίσουμε το επόμενο γινόμενο i+1 χρειαζόμαστε το γινόμενο στο στάδιο i και τον ίδιο τον αριθμό X.Για να κατασευάσουμε την επαναληπτική εξίσωση έχουμε απλά
Zi=W=Zi-100+Xi-100+01
Xi=Xi-11
αν το επόμενο ψηφίο είναι 1
Zi=W=Zi-100
Xi=Xi-10
αν το επόμενο ψηφίο είναι 0



Σχήμα 46:Διάγραμμα ροής για τον υπολογισμό ρίζας.

Παραδείγματα
Τετράγωνο:01010001 (81 unsigned)
Ρίζα:1001(9 unsigned)

-1ο βήμα:
 1
01|01|00|01

Z=1,X=1

-2ο βήμα:
 1 1
01|01|00|01

11*11=100+100+01=1001>0101

1 0
01|01|00|01

Z=100,X=10

-3ο βήμα:
 1 0 1
01|01|00|01

101*101=10000+1000+01=11001>010100

 1 0 0
01|01|00|01

Z=10000,X=100
-4ο βήμα:
 1 0 0
01|01|00|01

1001*1001=1000000+10000+01=01010001>=01010001

 1 0 0 1
01|01|00|01

Τετράγωνο:11010011 (211 unsigned)
Ρίζα:1110(14 unsigned)

-1ο βήμα:
1
11|01|00|11

Z=1,X=1
-2ο βήμα:
1
11|01|00|11

11*11=100+100+01=1001<1101

1 1
11|01|00|11

Z=1001,X=11

-3ο βήμα:
1 1
11|01|00|11

111*111=100100+1100+01=110001<110100

1 1 1
11|01|00|11

Z=110001,X=111

-4ο βήμα:
1 1 1
11|01|00|11

1111*1111=11000100+11100+01=11100001>11010011

1 1 1 0
11|01|00|11

Τετράγωνο:10110001 (177 unsigned)
Ρίζα:1101(13 unsigned)

-1ο βήμα:
1
10|11|00|01

Z=1,X=1

-2ο βήμα:
 1
10|11|00|01

11*11=100+100+01=1001<1011

 1 1
11|01|00|11
Z=1001,X=11
-3ο βήμα:
 1 1
10|11|00|01

111*111=100100+1100+01=110001>10110001

1 1 0
11|01|00|11

Z=100100,X=110

-4ο βήμα:
 1 1 0
10|11|00|01

1101*1101=10010000+11000+01=10101001<10110001

 1 1 0 1
11|01|00|11

Στην συνέχεια παραχωρούμε τον κώδικα της τετραγωνικής ρίζας

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use ieee.std_logic_unsigned.all;

use ieee.std_logic_arith.all;
--use ieee.math_real.all;
```

```
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity square_root is
	generic
	(
		N : positive := 32
	);
	port
	(
		INPUT : in std_logic_vector(N-1 downto 0);
		rst:in std_logic;
		Clk : in std_logic;
		OUTPUT : out std_logic_vector(N/2-1 downto 0)
	);
end square_root;


architecture Behavioral of square_root is



type chunk_X is array (N/2-1 downto 0) of std_logic_vector(N/2-1 downto 0);
	signal X,X_d,X_dd,W2: chunk_X;


type chunk_Q is array (N/2-1 downto 0) of std_logic_vector(N-1 downto 0);
	signal W,M,B,Z,M_d,Z_d,B_d,Z_dd,M_dd,W1: chunk_Q;

--signal J:integer;

begin
```

```
bit_1:process(clk)
begin
  if (Clk'event and Clk='1') then
    if (rst='0') then
      for i in 0 to N/2-1 loop
                  W(i)<=(others=>'0');
                        M(i)<=(others=>'0');
                        B(i)<=(others=>'0');
                        Z(i)<=(others=>'0');
                        M_d(i)<=(others=>'0');
                        Z_d(i)<=(others=>'0');
                  X(i)<=(others=>'0');
                        X_d(i)<=(others=>'0');
                        W1(i)<=(others=>'0');
                        W2(i)<=(others=>'0');
                        W(i)<=(others=>'0');
                        B_d(i)<=(others=>'0');
                        M_dd(i)<=(others=>'0');
                        Z_dd(i)<=(others=>'0');
                        X_dd(i)<=(others=>'0');
                end loop;
    else
      M(0)<=INPUT;
                  X(0)(0)<=INPUT(N-1) or INPUT(N-2);
                  Z(0)(0)<=INPUT(N-1) or INPUT(N-2);
                  ---------------------------------------------
                  for i in 1 to N/2-1 loop
                      for t in 0 to 2*(i+1)-1 loop
                                B(i)(t)<=M(i-1)(N-2*(i+1)+t);
                            end loop;
                            for t in 2*(i+1) to N-1 loop
                              B(i)(t)<='0';
                            end loop;
                            Z_d(i)<=Z(i-1);
                            W1(i)<=Z(i-1)(Z(i-1)'left - 2 downto 0) & "00";  -- shift left 2
                            W2(i)<=X(i-1)(X(i-1)'left - 2 downto 0) & "00";  -- shift left 1
                            X_d(i)<=X(i-1)(X(i-1)'left - 1 downto 0) & '0';  -- shift left 1

                            M_d(i)<=M(i-1);
                  ---------------------------------------------
                        B_d(i)<=B(i);
                            Z_dd(i)<=Z_d(i);
                            W(i)<=W1(i)+W2(i)+1;
```

```
                        X_dd(i)<=X_d(i);
                        M_dd(i)<=M_d(i);
                ------------------------------------------
                        if (W(i)>B_d(i)) then
                                Z(i)<=Z_dd(i)(Z_dd(i)'left - 2 downto 0) & "00";  -- shift left
2
                                X(i)<=X_dd(i);
                        else
                          Z(i)<=W(i);
                                X(i)<=X_dd(i)+1;
                        end if;
                        M(i)<=M_dd(i);
      end loop;
    end if;
  end if;
end process;

OUTPUT<=X(N/2-1);


end Behavioral;
```

Σημαντικά σήματα:
W1<=Z00
W2<=X00
W<=W
B_d register που περιέχει τον αριθμό που πρέπει να γίνει κάθε φορά σύγκριση.

Σημαντικές σημειώσεις:
-Στο παραπάνω κώδικα ,τα σήματα X περιέχουν το ζητούμενο αριθμό που σταδιακά με κάθε i παίρνει την τελική του μορφή.Στην συγκεκριμένη υλοποίηση το Xi-1 πολλαπλασιάζεται αρχικά με το 2 και στην συνέχεια ανάλογα με το τι είναι το επόμενο ψηφίο προσθέτεται 1 η αφήνεται όπως είναι.
-Στο γενικό κύκλωμα φαίνεται ότι οι είσοδοι Real και Imag είναι προσημασμένοι αριθμοί,όμως ο αλγόριθμος της ρίζας απαιτεί μη προσημασμένη είσοδο.Το πρόβλημα αυτό διορθώνεται καθώς μετά την πράξη x^2 οι προσημασμένη αριθμοί γίνονται θετικοί,οπότε δεν υπάρχει διαφορά με τους μη προσημασμένους ,και προσθέτονται με αποτέλεσμα έναν αριθμό 32 bit μη προσημασμένο.
-Το σήμα εξόδου έχει τα μισά bits από το σήμα εισόδου καθώς είναι η τετραγωνική ρίζα του.Στο δικό μας κύκλωμα τα bits εισόδου είναι 32 και τα bits εξόδου 16.

ΚΕΦΑΛΑΙΟ 6-ΜΟΝΑΔΑ ΕΤΕΡΟΣΥΣΧΕΤΙΣΗΣ

Το επόμενο αύτονομο κομμάτι της συνολικής αρχιτεκτονικής είναι το κύκλωμα που ουσιαστικά εκτελεί την πράξης της ετεροσυσχέτισης.Πολλαπλασιάζει τους όρους των δύο σημάτων και στην συνέχεια αθροίζει τα αποτελέσματα.Ο τρόπος με τον οποίο λειτουργεί είναι παρόμοιως με τον τρόπο που λειτουργεί ένα φίλτρο FIR, του οποίου συντελεστές είναι οι τιμές του σήματος που προσπαθούμε να αναγνωρίσουμε.

$$f*g(n)=\sum_{m=-\infty}^{+\infty} f(m-n)h(m)$$

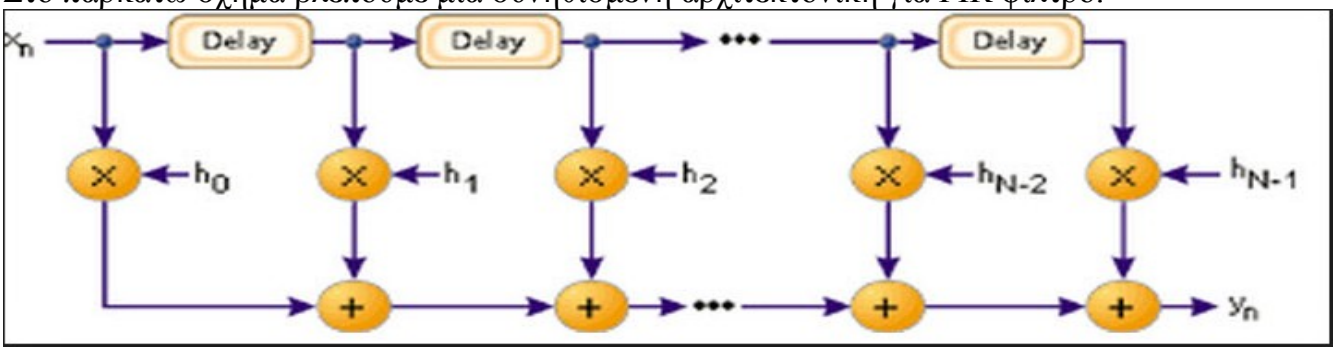
είναι η αρχική εξίσωση με την οπόία ξεκινήσαμε στο κεφάλαιο της εισαγωγής.Εάν το h(m) είναι σταθερό σήμα πεπερασμένης διάρκειας (320 σημείων σε αυτήν την υλοποίηση) τότε η εξίσωση γίενται

όπου hj οι συντελεστές του φίλτρου.Φαίνεται

$$f*g(n)=\sum_{m=a}^{b} f(m-n)hj$$

καθαρά ότι η εξίσωση λαμβάνει την μορφή FIR φίλτρου.

Στο παρκάτω σχήμα βλέπουμε μια συνηθισμένη αρχιτεκτονική για FIR φίλτρο.



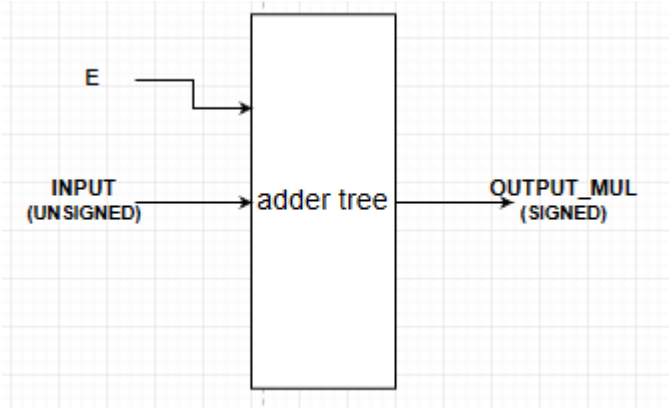
Σχήμα 47:Αρχιτεκτονική φίλτρου FIR.

Για μικρού μεγέθους σήματα (ήτοι <50 σημεία) η παραπάνω αρχιτεκτονική κρίνεται ικανοποιητική ,απο πλευράς κατανάλωσης πόρων πάνω στο fpga.Για σήματα όμως μεγαλύτερου μεγέθους το σχέδιο δεν κλιμακώνει αποδοτικά.Αυτό συμβαίνει καθώς η πολυπλοκότητα αυτής της υλοποίησης είναι O(n),καθώς για n όρους χρειαζόμαστε n αθροιστές και n πολλαπλασιαστές.

Επομένως είναι αναγκαία η ανάπτυξη ενός αλγορίθμου για την δραστική μείωση των αθροιστών και των πολλαπλασιαστών που χρειάζονται,καθώς και για την εξασφάλιση της αποδοτικότερης χρησιμοποίησης των λειτουργικών δομών της πλακέτας (μονάδες DSP).
Όπως θα αναλυθεί παρακάτω αυτό επιτυγχάνεται:
1)με την εκμετάλλευση της περιοδικότητας του preamble
2)με την χρήση δέντρου αθροιστών πολυπλοκοτητας O(log2n)
Στα επόμενα κεφάλαια αρχικά θα περιγράψουμε την δομή του δέντρου και στην συνέχεια θα παρουσιάσουμε την κατανάλωση πόρων για τις δύο μεθόδους(πολλαπλασιασμός και απόλυτο διαφοράς).

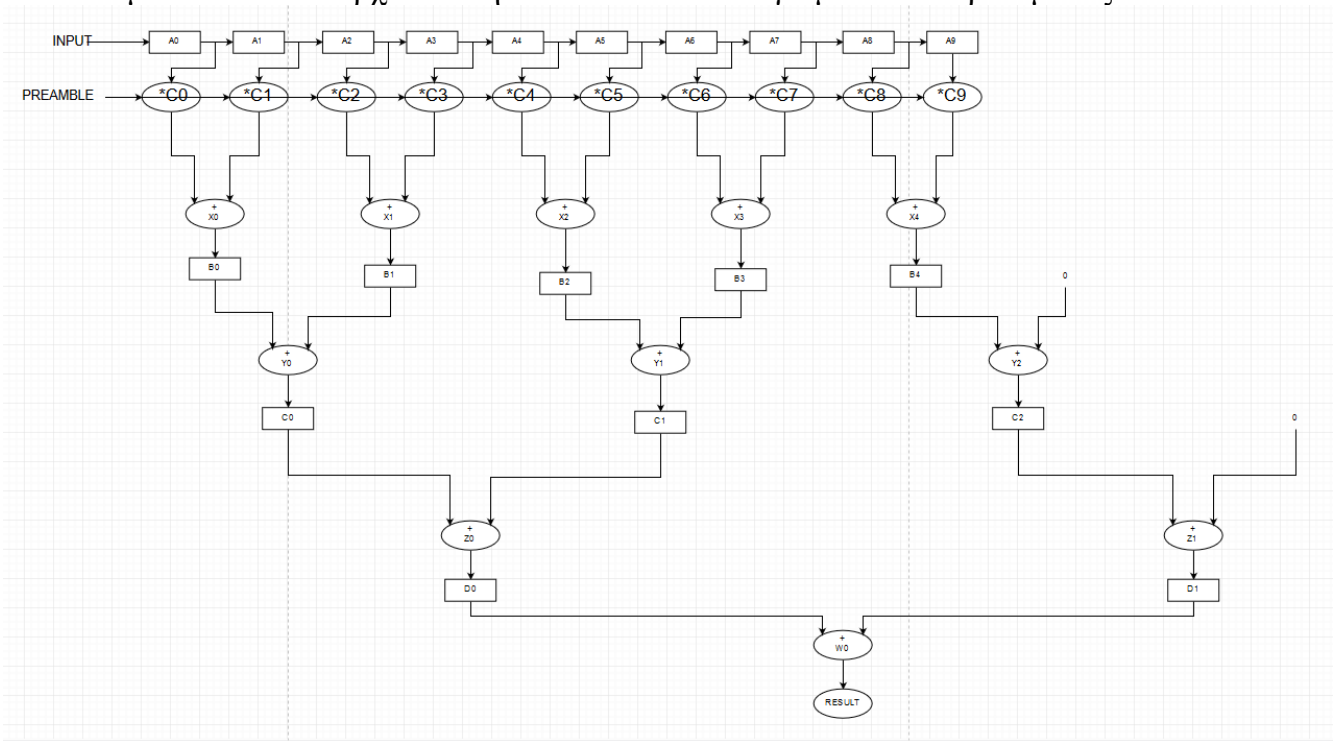
6.1 Μονάδα ετεροσυσχέτισης

Στο παρακάτω σχήμα βλέπουμε την μονάδα αθροιστών στην πιο απλοική της μορφή.

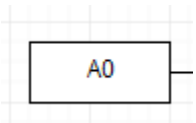


Σχήμα 48:Μπλο διάγραμμα μονάδας αθροιστών.

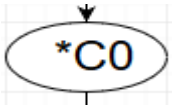
Ε είναι η είσοδος επίτρεψης και INPUT(16 bit) είναι το μέτρο του μιγαδικού σήματος αφού πρώτα έχουμε αφαιρέση την τιμή SUBBER(16 bit).Σε κάθε κύκλο ρολογιού στην είσοδο INPUT εμφανίζεται μια νέα τιμή.Η τιμή αυτή εισάγεται στους καταχωρητές του δέντρου και στον επόμενο κύκλο εισέρχεται στην διαδικασία υπολογισμού του αθροίσματος.



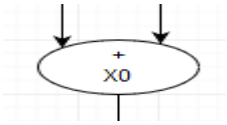
Σχήμα 49:Μονάδα ετεροσυσχέτισης με χρήση δέντρου αθροιστών(10 σημεία).



καταχωρητής



πολλαπλασιαστής



αθροιστής

Σχήμα 50:Βασικές μονάδες δεντρού αθροιστών.

Η συστοιχεία των καταχωρητών Aj υποδηλώνει ουσιαστικά την χρονική καθυστέρηση που χρειάζεται για να πολλαπλασιαστούν οι κατάλληλοι συντελεστές με τις παρελθοντικές τιμές της μορφής f(m-n) του αθροίσματος. Καταχωρητές υπάρχουν σε κάθε επίπεδο καθώς το άθροισμα υπολογίζεται.Το προοίμιο(preamble) είναι ένα διάνυσμα 10 σημείων που περιέχει τα σημεία του ζητούμενου σήματος.Σε κάθε πολλαπλασιαστή εισέρχεται το σημείο hj από το προοίμιο που αντιστοιχεί στην καθυστέρηση του κάθε καταχωρητή.Καταχωρητές υπάρχουν σε κάθε επίπεδο.

Αν στο επίπεδο j υπάρχουν 2*k όροι τότε ισχύει :

```
for w:0:k-1 loop
 tree(j+1)(w)<=tree(j)(2*w)+tree(j)(2*w+1);
end loop;
```

k adders

Αν στο επίπεδο j υπάρχουν 2*k+1 όροι τότε πρέπει να προσθέσουμε ακόμα έναν ώστε να έχουμε
2*k+2 όρους.Ο όρος που προσθέτεται είναι το 0 οπότε:

```
for w:0:k-1 loop
 tree(j+1)(w)<=tree(j)(2*w)+tree(j)(2*w+1);
end loop;
tree(j+1)(k)<=tree(j)(2*k)+0;
```

k+1 adders

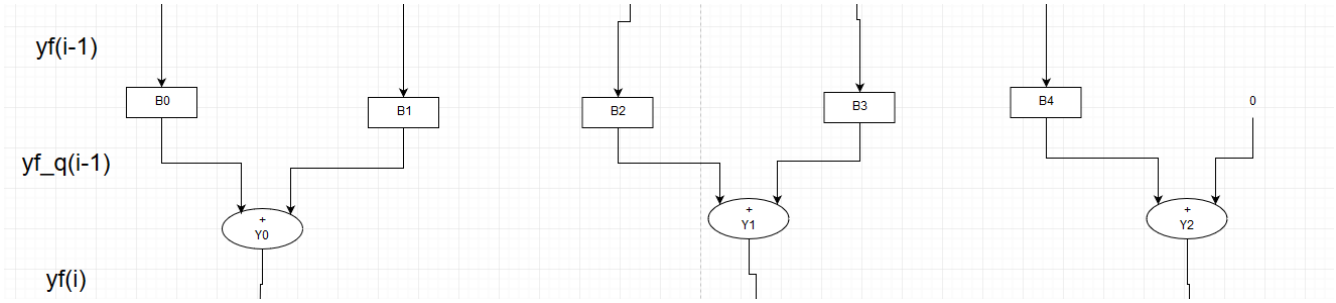
Η διαδικασία αυτή συνεχίζεται επαναληπτικά μέχρι να βρεθούν οι αριθμοί των adders που χρειάζονται σε κάθε επίπεδο.Υπολογίζουμε τον αριθμό αθροιστών ποθ χρειαζόμαστε σε κάθε επίπεδο με την συνάρτηση

```
function tree_Levels(X,Y: in integer) return integer_vector is
        variable val : integer_vector(X downto 0);
    begin
        val(0) := Y;
        for j in 1 to X loop
            val(j) := integer(ceil(real(val(j-1))/2.0));
        end loop;
        return val;
    end function;
```

Η μεταβλητή integer_vector είναι πίνακας ακεραίων που σε κάθε θέση περιέχει τον αριθμό των adders που απαιτούνται.

Από τα παραπάνω αποδεικνύεται ότι το πλήθος των επιπέδων είναι ceil(log2N).Σύμφωνα με αυτό ,σε κάθε ένα από αυτά τα επίπεδα

επίπεδο I



Σχήμα 51:Επίπεδο δέντρου αθροιστών.

οι είσοδοι(registers) οδηγούνται ανα δύο σε αθροιστές (στην περίπτωση του 2*k+1 προσθέτουμε το 0 και έχουμε 2*k+2 όρους).Αναλυτικότερα έχουμε τον κώδικα :

```
gi: for i in 1 to DEPTH generate -- If N = 1 --> LV = 0 and this loop is not run
                -- The registers are created:
                D_flip_flops: for k in 0 to ADD_PER_LVL(i-1)-1 generate
                                        Z: dff generic map (N => B + (i-1))
                                           port map (clock => clock, resetn => resetn, E => '1', sclr
=> '0', D => yf(i-1,k)(B + (i-1) - 1 downto 0), Q => yf_q(i-1,k)(B + (i-1) - 1 downto 0));
                                                --yf_q(i-1,k)(B + i-1) <= '0'; -- zero extension
                                                yf_q(i-1,k)(B + i-1) <= yf_q(i-1,k)(B + (i-1) - 1); --
sign extension, this would be 'yf_p'
                        end generate;

    ADDERS: for j in 0 to ADD_PER_LVL(i)-2 generate
                                        A: DSP_Pipeline_Adder generic map (N => B + i)
                                           port map ( A => yf_q(i-1,2*j)(B + i - 1 downto 0), B =>
yf_q(i-1,2*j + 1)(B + i - 1 downto 0),clk=>clock, S => yf(i,j)(B + i - 1 downto 0));
                        end generate;

                Z101: if (ADD_PER_LVL(i-1) rem 2 = 0) generate -- X(i-1) is even?, is the same
as asking 2*X(i) = X(i-1)
                                        X3: DSP_Pipeline_Adder generic map (B + i)
                                                port map ( A => yf_q(i-1, 2*(ADD_PER_LVL(i)-1))
(B + i - 1 downto 0), B => yf_q(i-1, 2*(ADD_PER_LVL(i)-1) + 1)(B + i - 1 downto
0),clk=>clock, S => yf(i,ADD_PER_LVL(i)-1)(B + i - 1 downto 0));
                                        end generate;

                Z102: if (ADD_PER_LVL(i-1) rem 2 = 1) generate -- X(i-1) is even?, is the same
as asking 2*X(i) = X(i-1)
                                        X3: DSP_Pipeline_Adder generic map (B + i)
                                                port map ( A => yf_q(i-1, 2*(ADD_PER_LVL(i)-1))
(B + i - 1 downto 0), B => (others=>'0'),clk=>clock, S => yf(i,ADD_PER_LVL(i)-1)(B + i - 1
downto 0));
```

end generate;
end generate; Yo <= yf (DEPTH,0);

Στην περιοχή D_flip_flops έχουμε τους αριθμημένους register του σχήματος ,στους οποίους ανάλογα με το είδος των αριθμών είτε γίνεται extend το bit προσήμου(σε περίπτωση signed) είτε προσθέτεται απλά 0 στην αρχή (σε περίπτωση unsigned) ώστε να μην υπάρχει περίπτωση υπερχείλησης.

Στην περιοχή ADDERS δημιουργούνται τα ζευγάρια των αθροιστών χωρίς όμως το τελευταίο ζευγάρι.

Αυτό δημιουργείται στις δύο επόμενες περιοχές Z101 και Z102 ανάλογα με το αν το πλήθος των όρων είναι πολλαπλάσιο του δύο(όπως αναφέρθηκε νωρίτερα).

Το αποτέλεσμα του δέντρου είναι το αποτέλεσμα του αθροιστή στο τελευταίο επίπεδο Yo <= yf (DEPTH,0); και είναι (16+16+integer(ceil(log2(real(320)))=41 bits.Ο όρος 2*16 προέρχεται από τον πολλαπλασιασμό και ο όρος integer(ceil(log2(real(320))) από τα επίπεδα του δέντρου αθροιστών τα οποία προσθέτουν ανάλογο αριθμό bit.

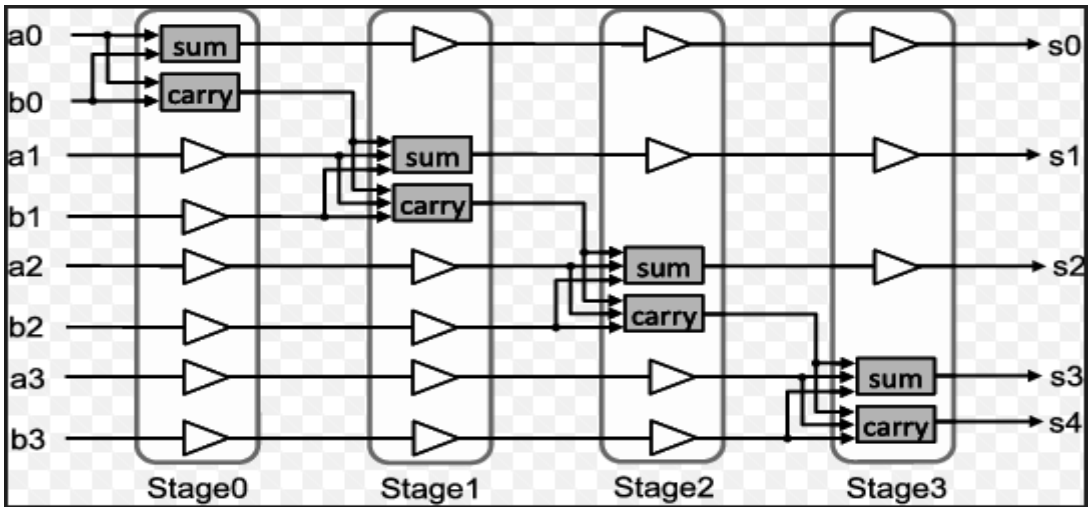
Λειτουργικές μονάδες αθροιστών και πολλαπλασιαστών.

Για την ικανοποίηση της απαίτησης των 500 Mhz χρησιμοποιήθηκαν 2 αρχιτεκτονικές αθροιστών και 2 αρχιτεκονικές πολλαπλασιαστών.

-Οι αθροιστής και πολλαπλασιαστής(bit level pipeline) στους οποίους όλες οι πράξεις είναι γραμμένες σε επίπεδο flip-flop (16 bit).

-Οι κανονικοί αθροιστές και πολλαπλασιαστές στους οποίους οι πράξεις γίνονται με inference στον synthesizer (16 bit) .

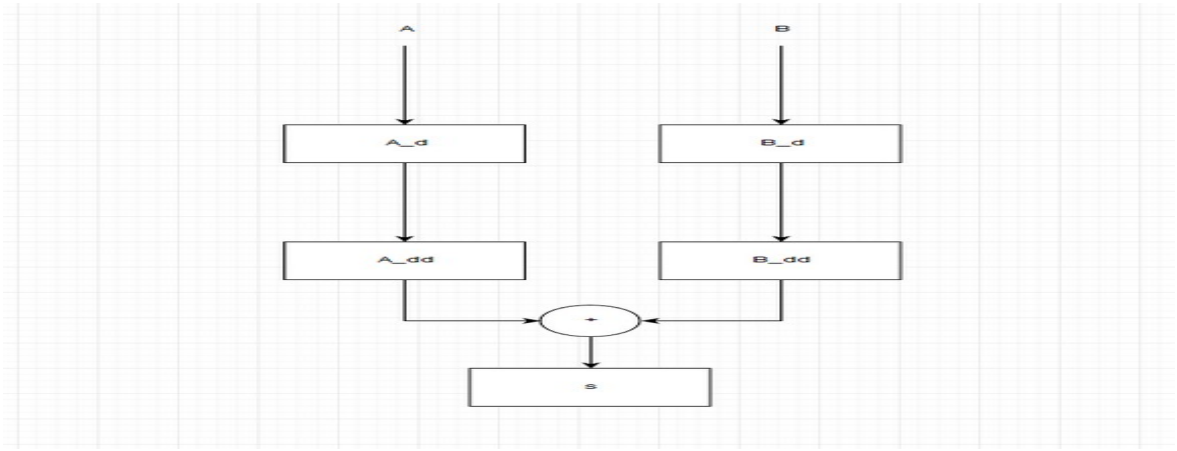
Στα επόμενα σχήματα θα δούμε πραδείγματα αυτών των λειτουργικών μονάδων.Στην περίπτωση των bit level μονάδων τα σχήματα παριστάνουν μικρότερο αριθμό bit λόγω μεγέθους των 16 bit αντίστοιχων κυκλωμάτων.



Σχήμα 52:Αθροιστές επιπέδου bit.

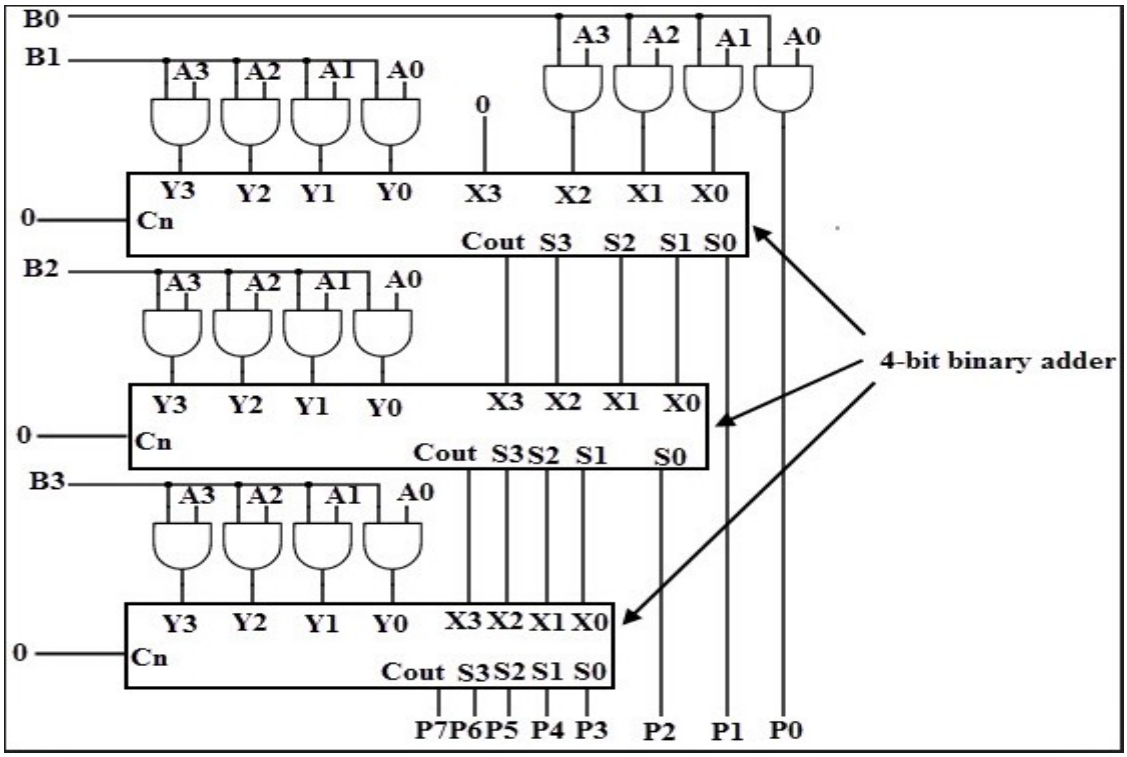
```
process(clk)
begin
if (clk'event and clk = '1') then
        A_d<=A;
        B_d<=B;
```

```
            A_dd<=A_d;
            B_dd<=B_d;
            S<=A_dd+B_dd;
 end if;
end process;
```



Σχήμα 53: 2 stage pipeline αθροιστής.

Όπως και στην περίπτωση των αθροιστών έχουμε 2 είδη πολλαπλασιαστών



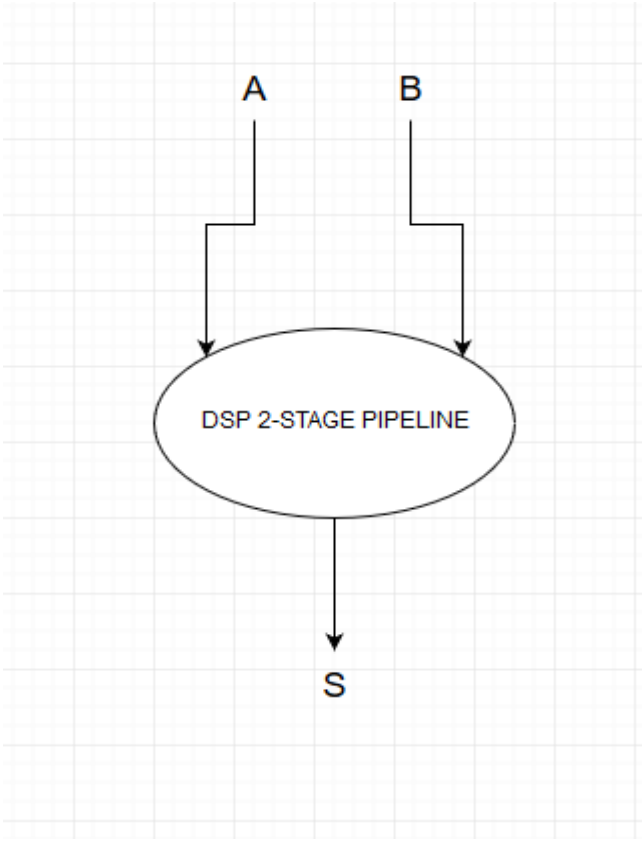
Σχήμα 54:Αρχιτεκτονική πολλαπλασιαστή.

Αυστηρά για unsigned αριθμούς.

Pipelined multipliers με χρήση DSP

```
process(clk)
begin
        if (clk'event and clk = '1') then
                A_d<=a;
                        B_d<=b;
                        A_dd<=A_d;
                        B_dd<=B_d;
                        --p<=A_dd*B_dd;
                        p<=std_logic_vector(signed(A_dd)*signed(B_dd));
end if;
end process;
```

Μπορούμε να επιλέξουμε μεταξύ signed kai unsigned αριθμών.

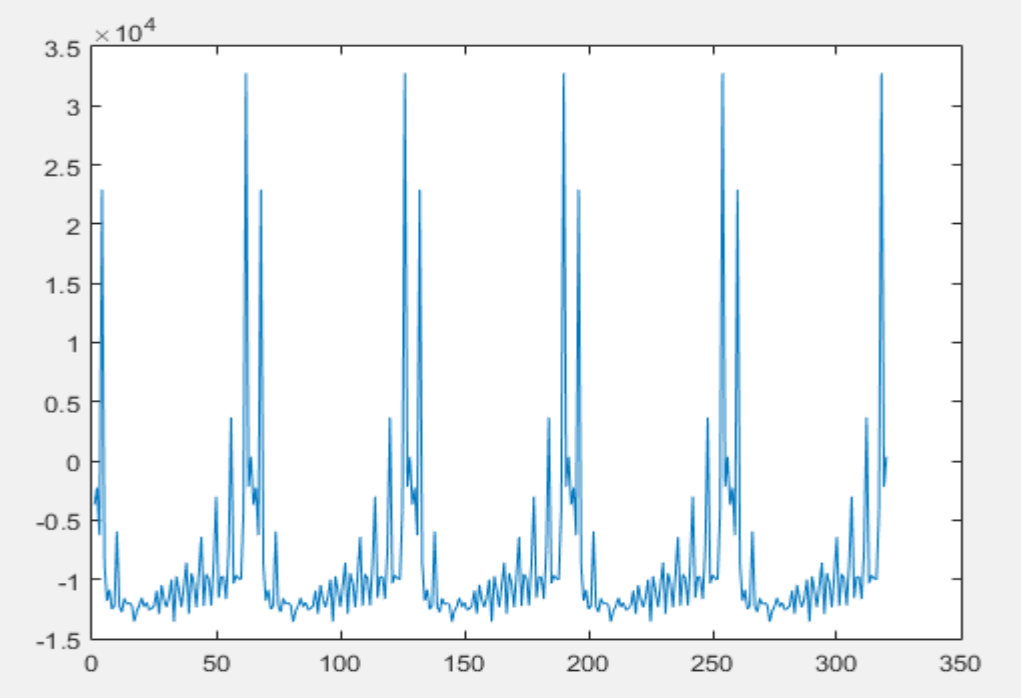


Σχήμα 55: 2 stage pipeline multiplier.

Οι κώδικες για τους bit-level αθροιστές και πολλαπλασιαστές δίνονται ξεχωριστά μαζί με το υπόλοιπο κύκλωμα καθώς είναι πολύ μεγάλοι .

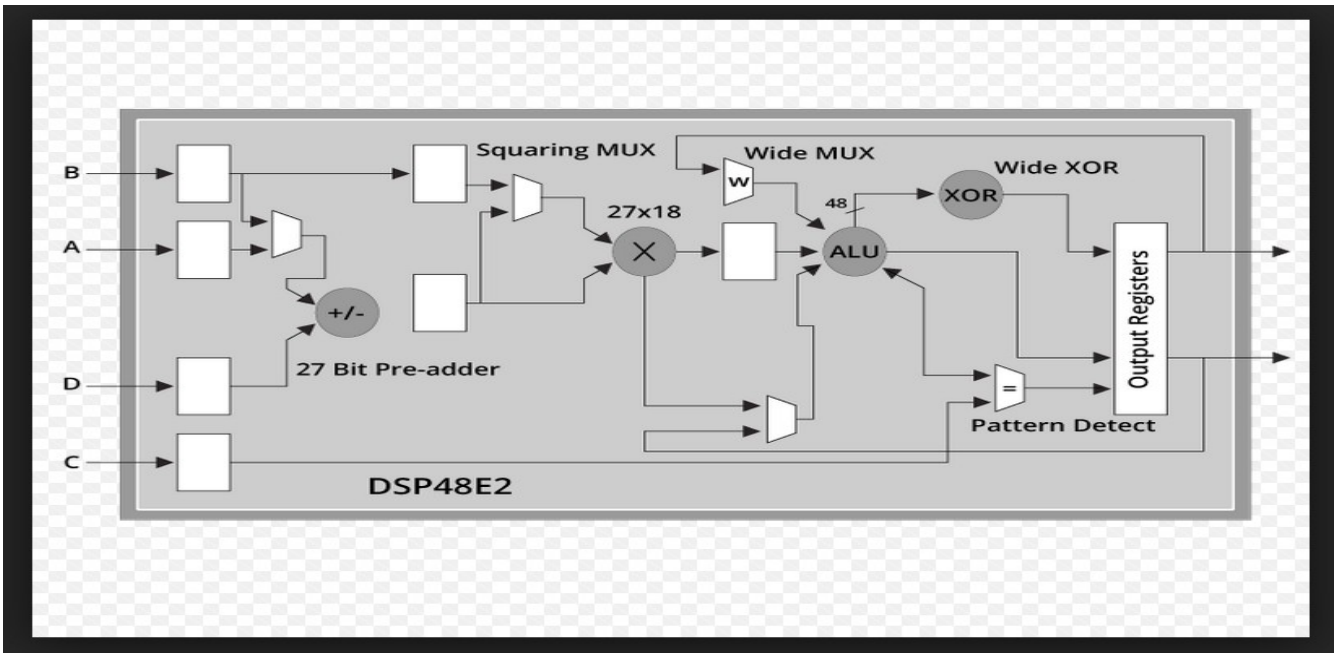
6.2 Περιοδηκότητα προοιμίου

Πριν ξεκινήσουμε την τελική υλοποίηση του κυκλώματος πρέπει να αναφερθούμε και σε μία ειδικού τύπου βελτιστοποίηση η οποία είναι δυνατή λόγο της περιοδηκότητας του μέτρου του προοιμίου.Εάν πρατηρήσουμε το σχήμα βλέπουμε ότι το προοίμιο είναι περιοδικό με περίοδο 64 σημεία(320/64=5 περίοδοι).



Σχήμα 56:Μέτρο σήματος μετά την αφαίρεση τιμής.

Ακόμα έχουμε την αναλυτικότερη εικόνα του DSP48E2 που χρησιμοποιούμε για την λειτουργία του κυκλώματος.



Σχήμα 57:DSP48E2.

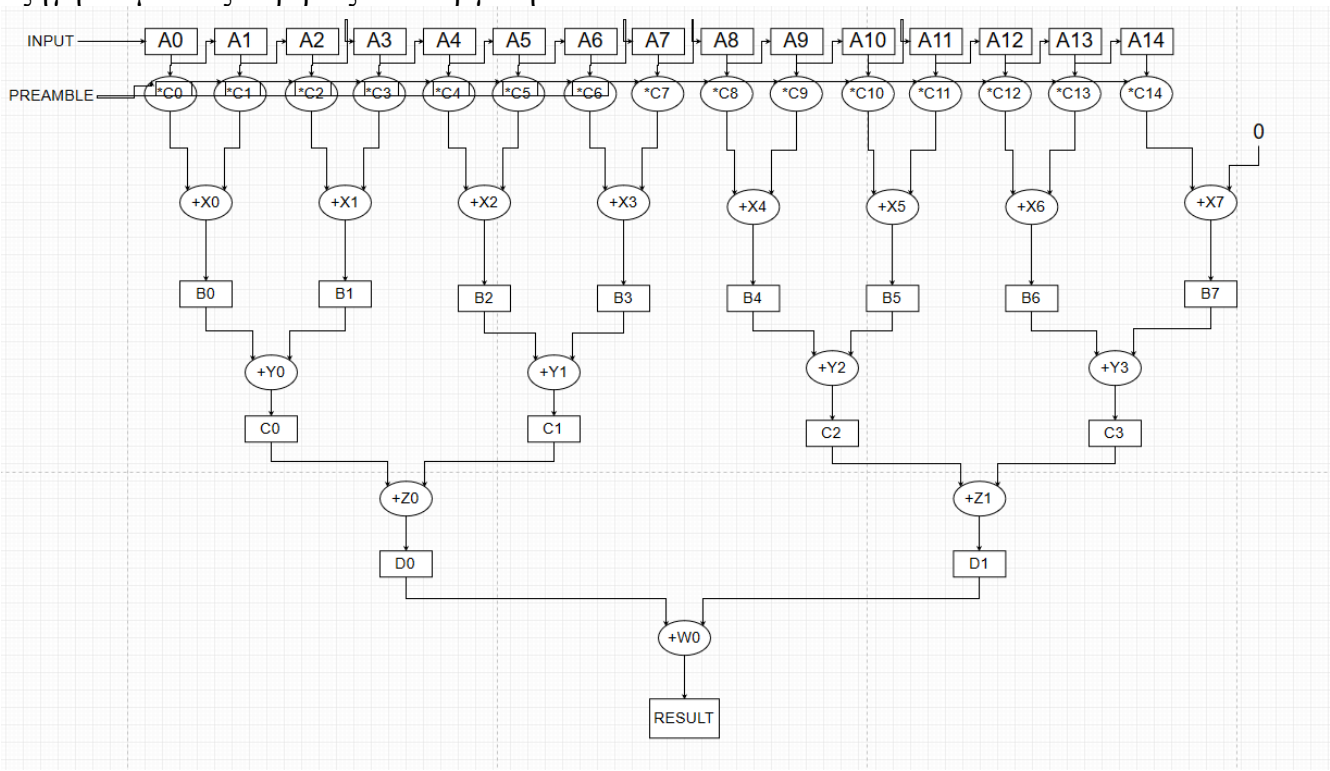
Όπως φαίνεται στην εικόνα υπάρχει ένας pre-adder πριν τον πολλαπλασιαστή ο οποίος δεν χρησιμοποιείται, καθώς στην αρχιτεκτονική των 320 taps η πρόσθεση έπεται του πολλαπλασιασμού.

Για την πλήρη συμμετοχή των λειτουργικών μονάδων του DSP θα πρέπει μέρος των πράξεων να μετασχηματιστούν σε προσθέσεις πριν τους multipliers.

Αυτό επιτυγχάνεται με την εκμετάλλευση της ιδιότητας της περιοδικότητας του preamble με τον ακόλουθο τρόπο.Το preamble είναι ένα σήμα με 320 σημεία συνολικά και 64 σημεία περίοδο.Έχει ουσιαστικά 5 περιόδους άρα 64 πεντάδες σημείων ίσων μεταξύ τους. Επομένως ,αντί να πολλαπλασιάζεται κάθε σημείο του εισερχόμενου σήματος με ένα σημείο του preamble είναι δυνατό όλα τα σημεία με κοινό συντελεστή πρώτα να αθροιστούν και στην συνέχεια να πολλαπλασιαστούν με τον κοινό τους παράγοντα.Ουσιαστικά πρόκειται για την εφαρμογή της επιμεριστικής ιδιότητας

a*x+b*x+c*x+d*x=x*(a+b+c+d).

Στην συνέχεια παραθέτουμε ένα παράδειγμα 15 σημείων με περιοδικότητα 5 έτσι ώστε να εξηγήσουμε πως ακριβώς λειτουργεί η διαδικασία.



Σχήμα 58: Αρχική μονάδα ετεροσυσχέτισης(15 σημεία).

Αυτό είναι το αρχικό δέντρο χωρίς την βελτιοστοποίηση της περιοδικότητας.Περιοδικότητα 5 σημείων σε ένα σήμα 15 σημείων πρακτικά σημαίνει ότι για τα σημεία του προοιμίου που εισέρχονται στου πολλαπλασιαστές ισχύει:

C0=C5=C10

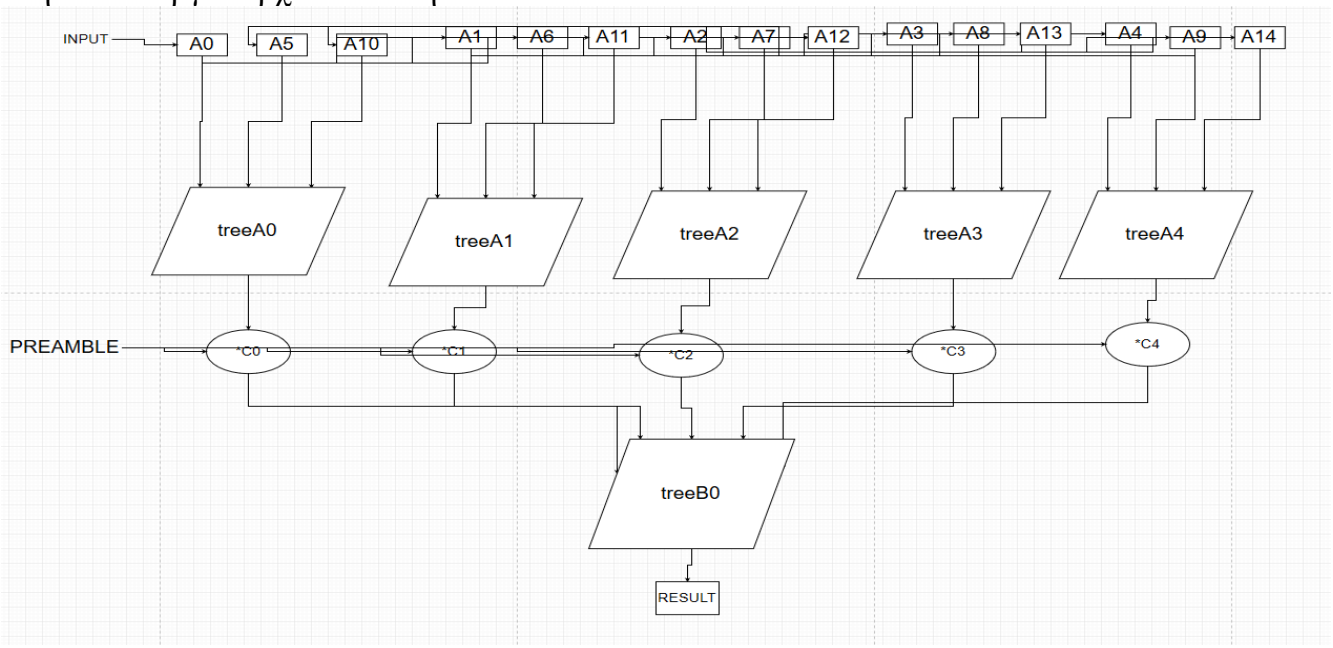
C1=C6=C11

C2=C7=C12

C3=C8=C13

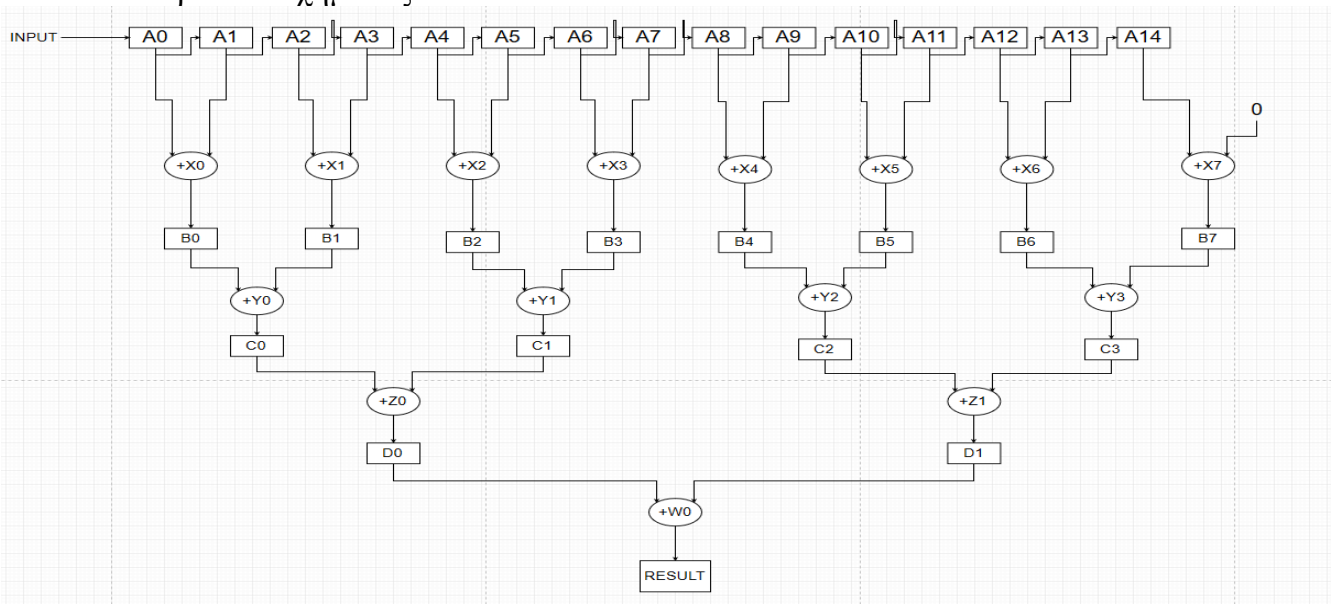
C4=C9=C14

Συνεπώς οι τιμές που βρίσκονται στους αντίστοιχους καταχωρητές πολλαπλασιάζονται με την ίδια ποσότητα.Προσθετουμε επομένως αυτές τις τιμές πριν τις πολλαπλασιάσουμε και έχουμε στην καινούργια αρχιτεκτονική.



Σχήμα 59:Δέντρο με βελτιστοποίηση περιοδικότητας.

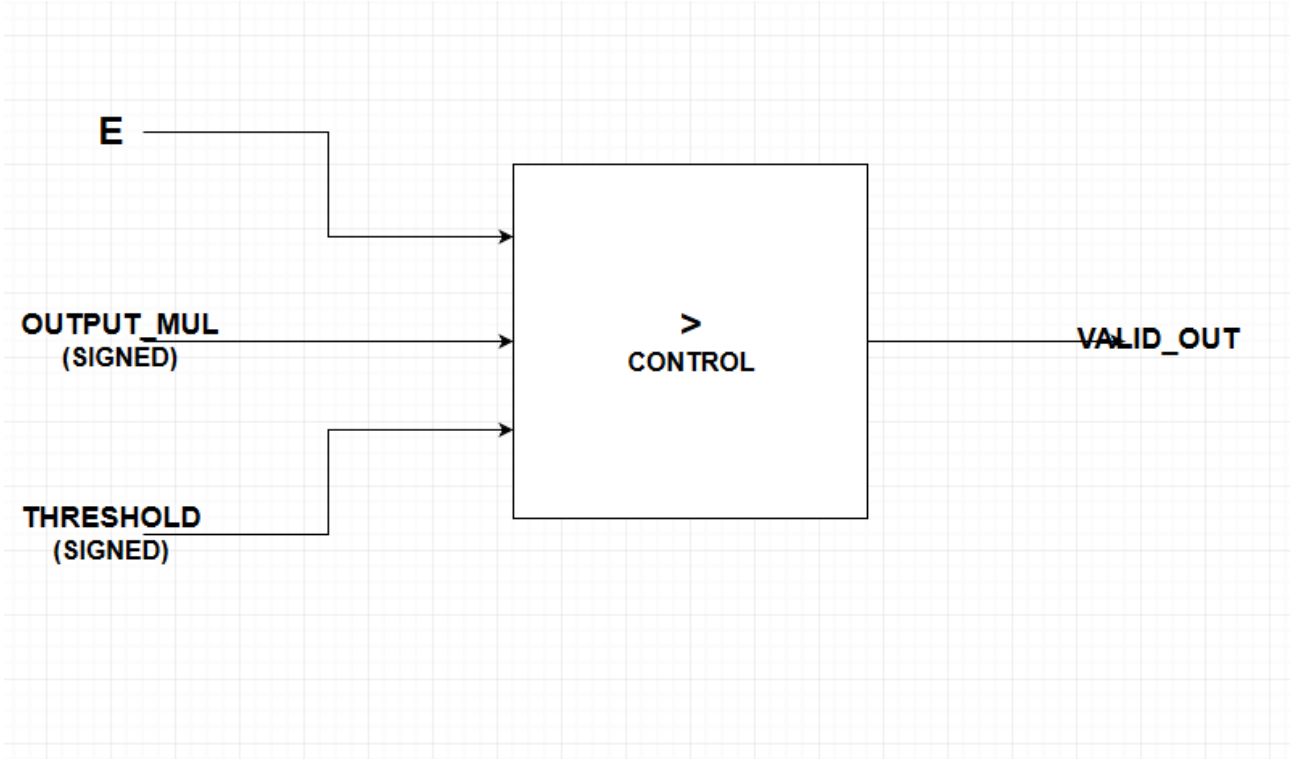
Το κύκλωμα treeA και treeB είναι κύκλωμα της μορφής του δέντρου αθροιστών χωρίς όμως τους πολλαπλασιαστές.Αθροίζει απλά τα περιεχόμενα όλων των καταχωρητών.Ένα τέτοιο είναι το δέντρο του σχήματος 60.



Σχήμα 60:Δέντρο αθροιστών.

Στο καταχωρητή RESULT βρίσκεται η τελική τιμή του αθροίσματος που αποτελεί και την τιμή της ετεροσυσχέτισης στην συκγεκριμένη χρονική στιγμή.Επόμενο βήμα είναι ο έλεγχος της συγκεκριμένης τιμής έτσι ώστε να ανιχνεύεται η κορυφή που σηματοδοτεί την ύπαρξη του σήματος που αναζητούμε.

ΚΕΦΑΛΑΙΟ 7-ΚΥΚΛΩΜΑ ΕΛΕΓΧΟΥ



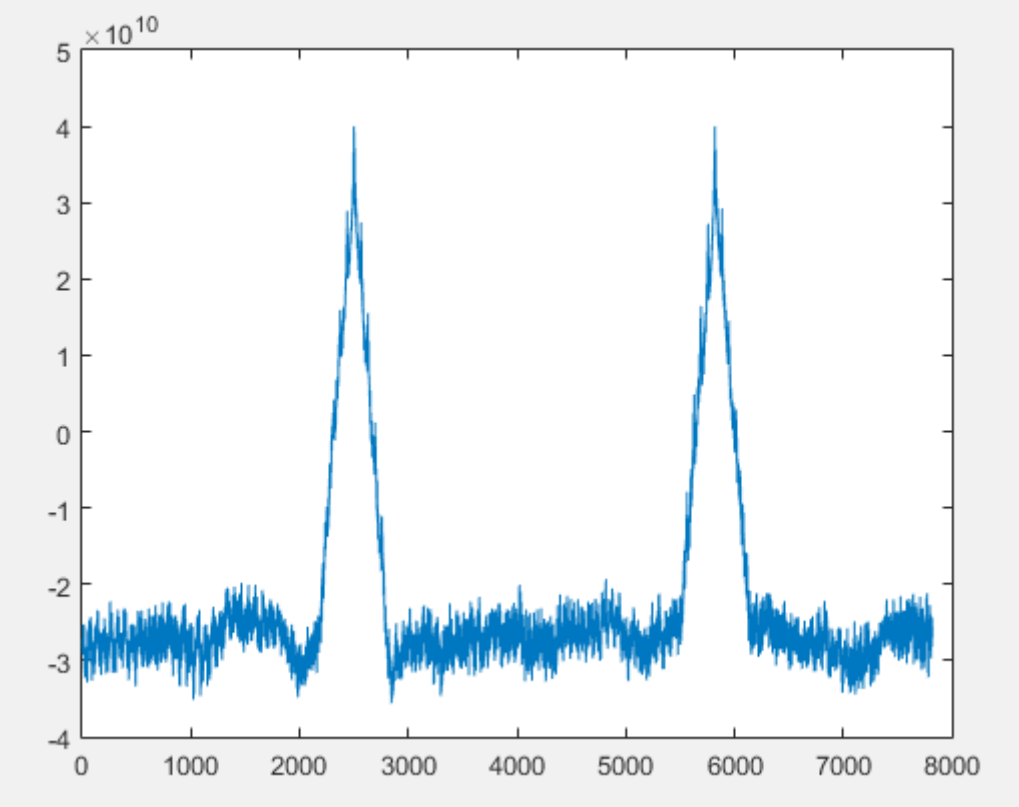
Σχήμα 61: Μπλοκ κυκλώματος ελέγχου.

Το κύκλωμα παίρνει ως εισόδους το Enable(επίτρεψη) ,την τιμή της ετεροσυσχέτισης(OUTPUT_MUL) και το όριο(threshold) κοντά στο οποίο θα αναζητηθεί το σήμα.Το Enable είναι κοινό με τον tree adder μετά το delay, έτσι ώστε σε περιπτώση διακοπής του stream για κάποιους κύκλους,να παγώσει ο υπολογισμός της ετεροσυσχέτισης .
Όταν η ροή του stream αποκατασταθεί, τότε το κύκλωμα ανίχνευσης της αυτοσυσχέτισης θα συνεχίσει από το σημείο που είχε σταματήσει.

Το design διαθέτει 3 μετρητές index , pos και out_timer.Ο μετρητής index χρησιμοποιείται για να υπολογιστεί η μέγιστη τιμή των ετεροσυσχετίσεων, μέσα στο παράθυρο τιμών που έχει οριστεί ,αφού ανιχνευθεί το threshold.Ο μετρητής pos συγχρονίζει την τιμή του valid_out.Τέλος,ο μετρητής out_timer μετράει τους 320 κύκλους για τους οποίους το valid_out πρέπει να μείνει 1.

Επειδή σε τηλεπικοινωνιακό δίαυλο πραγματικών δεδομένων ο οποίος αλλάζει σε πραγματικό χρόνο δεν υπάρχει τρόπος να βρούμε τις κορυφές καταφεύγουμε σε ένα τέχνασμα.Θεωρούμε ένα όριο (threshold) το οποίο αν ξεπεραστεί από τις τιμές της ετεροσυσχέτισης θεωρούμε ότι βρισκόμαστε κοντά σε τοπικό μέγιστο άρα σε κοντινή απόσταση αναζητούμε το μέγιστο.

Στα παρακάτω σχήματα φαίνεται η περιοχή του τοπικού μέγιστου.



Σχήμα 62:Παράδειγμα αποτελέσματος ετεροσυσχέτισης.

Στην συκγκριμένη περίπτωση το όριο(threshold) τοποθετείται κοντά στην τιμή 2x 10^{10} και το max αναζητείται στα επόμενα 500 σημεία.Και στις δύο περιπτώσεις βρίσκουμε σωστά την κορυφή.

Αναλυτικότερα η λειτουργία του κυκλώματος έχει ως εξής:

1)Το κύκλωμα ανιχνεύει την πρώτη τιμή που είναι ίση ή μεγαλύτερη του threshold.

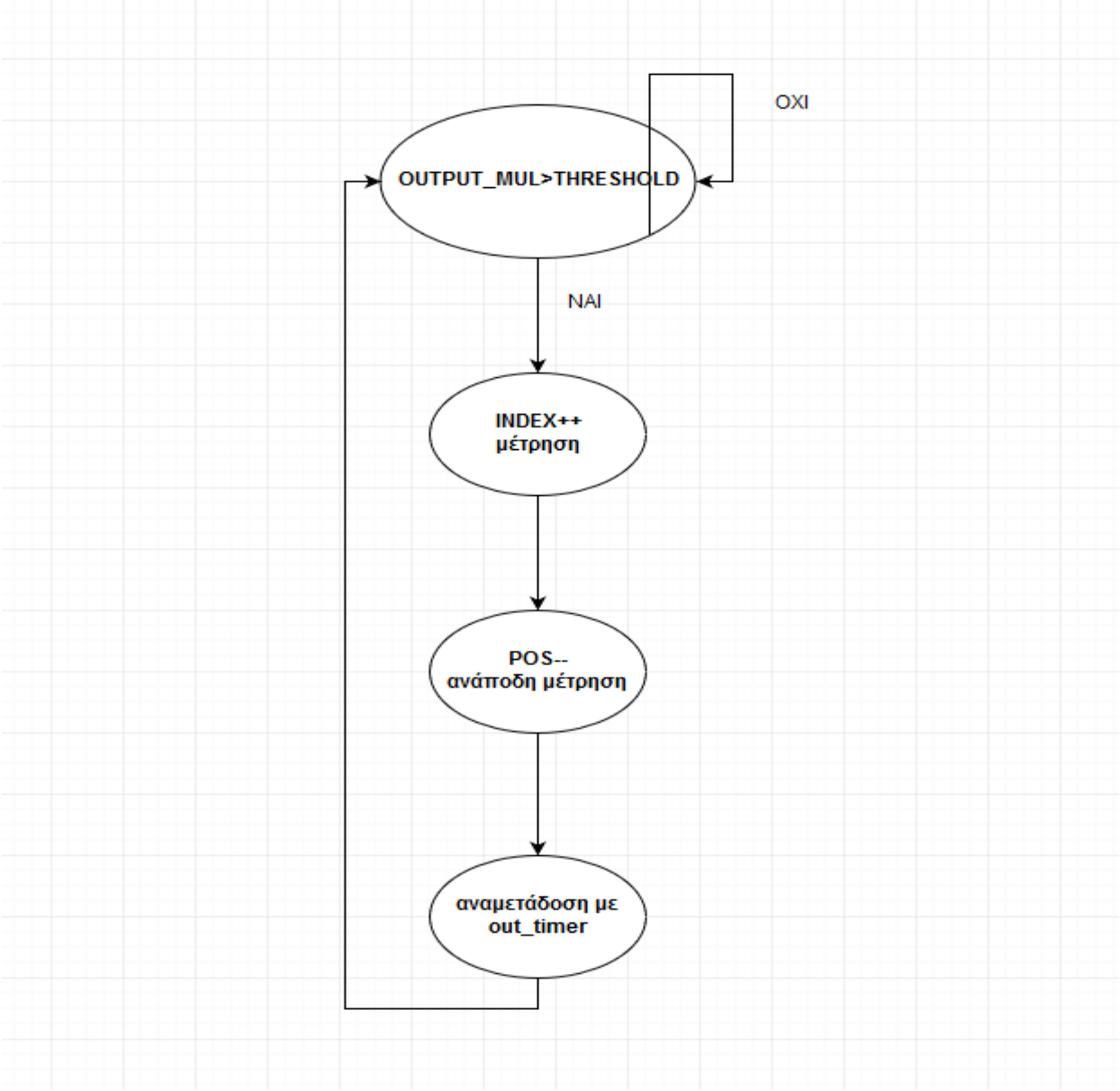
2)Εισέρχεται στην κατάσταση εύρεσης του max μέσα στα επόμενα 250 σημεία που ακολοθούν.

3)Ο μετρήτης pos έχει την θέση index στην οποία ανιχνεύθηκε το τελευταίο μέγιστο.

4)Στην συνέχεια το κύκλωμα εισέρχεται σε θέση αναμετάδωσης.Ο μετρητής αρχίζει και μετράει αντίστροφα μέχρι την τιμή 0 όπου και θέτει το valid_out στην τιμή 1.

5)Ο out_timer ενεργοποείται και μετράει 320 κύκλους για να μεταδοθεί όλο το preamble.

Για την σωστή αναμετάδοση του σήματος με τον μετρητή pos είναι πολή σημαντική η επιλογή της σωστής καθυστέρησης.Όπως φαίνεται στο αρχικό module η καθυστέρηση που αντιστοιχεί στο κύκλωμα της ετεροσυσχέτησης είναι 41 +WINDOW +1 +319 .41 κύκλοι είναι η καθυστέρηση που προκαλείται από το το δέντρο των αθροιστών.Η παράμετρος του παραθύρου είναι WINDOW και προκαλεί καθυστέρηση στο buffer όσο μετράει το index.Ο ένας παραπάνω κύκλος είναι η καθυστέρηση που χρειάζεται για να ανιχνευθεί το threshold.Οι 319 κύκλοι υπάρχουν για να μεταδοθεί σωστά το preamble .Η συσχέτιση ανιχνεύεται στο 320ο σημείο συνεπώς χρειζόμαστε επιπλέον καθυστέρηση 319 κύκλων για να μεταδώσουμε τα προηγούμενα 319 σημεία.



Σχήμα 63:FSM κυκλώματος ελέγχου.

Κώδικας κυκλώματος ελέγχου

```
process (resetn, clock)
        begin
          if (resetn='0') then
                  valid_out<='0';
                  valid<='0';
                  valid_next<='0';
                  pos<=(others=>'0');
                  max<=(others=>'0');
                  index<=(others=>'0');
                  out_timer<=(others=>'0');
          elsif (clock'event and clock = '1') then
           if (E='1') then
                  if (signed(Yxo)>=signed(spiked) and index=0 and pos=0 and Yxo/=0 and
out_timer=0) then
                     index<= std_logic_vector(to_unsigned(1, index'length));
                           pos<=std_logic_vector(to_unsigned(1, pos'length));
                           max<=Yxo;
                  end if;
                  if (index>=1) then
                     if (signed(Yxo)>signed(max)) then
                       max<=Yxo;
                       pos<=index+1;
                     end if;
                     index<=index+1;
                     if (index+1=WINDOW) then
                       index<=(others=>'0');
                       valid<='1';
                     end if;
                  end if;
                  if (valid='1') then
                    if (pos-1=0) then
                       valid<='0';
                       valid_out<='1';
                       out_timer<=std_logic_vector(to_unsigned(1, out_timer'length));
                    end if;
                    pos<=pos-1;
                    --valid_out_gate<=valid;
                  end if;
                  if (out_timer=N) then
                    valid_out<='0';
                    out_timer<=(others=>'0');
                  elsif (out_timer>0) then
                    out_timer<=out_timer+1;
                  end if;
          BUFFERS(0)<=stream_start;
```

```
                forE: for u in 0 to LATENCY-1-1 loop
                  BUFFERS(u+1)<=BUFFERS(u);
        end loop;
        BUFFERS_ANGLE(0)<=angle;
                forS: for u in 0 to LATENCY-1-1 loop
                  BUFFERS_ANGLE(u+1)<=BUFFERS_ANGLE(u);
        end loop;
        stream<=BUFFERS(LATENCY-1);
        stream_angle<=BUFFERS_ANGLE(LATENCY-1);
       end if;
     end if;
end process;
```

ΤΡΙΤΟ ΜΕΡΟΣ:ΣΥΜΠΕΡΑΣΜΑΤΑ

ΚΕΦΑΛΑΙΟ 8-ΑΠΟΤΕΛΕΣΜΑΤΑ ΥΛΟΠΟΙΗΣΗΣ

Σε αυτό το κεφάλαιο θα παρουσιαστούν τα αποτελέσματα των επιμέρους καθώς και της συνολικής μονάδας που σχεδιάστηκε.Οι βασικές προδιαγραφές ήταν η επίτευξη ρολογιού 500 Mhz και η κατανάλωση του 2 των πόρων του fpga.

8.1 Αποτελέσματα υλοποίησης τετραγωνικής ρίζας.

Για την τετραγωνική ρίζα τα αποτελέσματα υλοποίησης πάνω στο fpga είναι τα εξής.
Ο αριθμός των bits του σήματος εισόδου είναι 32 bit και εξόδου 16 bit.

Site Type	**Used**
CLB LUTSs	492
LUT as Logic	455
LUT as Memory	37
LUT as Distributed RAM	0
LUT as Shift Register	37
CLB Registers	2320
Registers as Flip Flop	2320
Register as Latch	0
CARRY8	69

Πίνακας 1:Αριθμός LUTs για υλοποίηση του κυκλώματος ρίζας πάνω στο fpga.

Βλέπουμε ότι με την μέθοδο που αναπτύξαμε στο κεφάλαιο 5 για την απαλοιφή της πράξης του πολλαπλασιασμού το κύκλωμα περιορίστικε σε έναν πολύ μικρό αριθμό LUTS ενώ δεν χρησιμοποιούνται καθόλου DSPs.Το γεγονός αυτό είναι πολύ σημαντικό καθώς θα μας βοηθήσει στο τέλος της συνολικής υλοποίησης να κρατήσουμε τον αριθμό των DSPs μέσα στους στόχους μας.

8.2 Αποτελέσματα υλοποίησης δέντρου αθροιστών.

Η υλοποίηση του δέντρου έγινε ,όπως είδαμε στο κεφάλαιο 6,με δύο τρόπους.Αρχικά χρησιμοποιήσαμε bit level αθροιστές και πολλαπλασιαστές και στην συνέχεια αθροιστές με inference.Στόχος ήταν το σήμα να αποτελείται από 320 σημεία με ακρίβεια τα 16 bit.
Αρχικά έχουμε τις εκτελέσεις της υλοποίησης με bit level αθροιστές και πολλαπλασιαστές.

Taps/Bits	8	12	16	20
16	2297	4522	7989	12764
32	4657	9769	16353	25701
64	9291	18979	32802	51349
136	21979	40931	-	-
256	57645	77018	-	-
272	40264	82012	-	-

Πίνακας 2:Αριθμός LUTs για υλοποίηση μονάδας ετεροσυσχέτισης με bit level αθροιστές

και πολλαπλασιαστές.

Είναι φανερό ότι η υλοποίηση δεν έφτασε μέχρι τα επιθημητά μεγέθη (αριθμό bits και σημεία). Όπως βλέπουμε η κλιμάκωση του αριθμού των LUTS είναι τόσο μεγάλη που πλέον το κύκλωμα δεν έχει νοήμα να ελεγχθεί για τα ζητούμενα μεγέθη.Σύμφωνα με αυτά απορρίπτουμε τους bit level αθροιστές και πολλαπλασιαστές και προχωράμε στις λειτουργικές μονάδες με inference.

Η κατανάλωση πόρων σε αυτήν την περίπτωση έγινε με και χωρίς την βελτιστοποίηση της περιοδικότητας που εξετάσαμε στο κεφάλαιο 6.3.Τα αποτελέσματα είναι το παρακάτω

taps	bits	architecture	LUTS	DSPs
320	16	No period opt.	12456	320
320	16	Perido opt.	8567	64

Πίνακας 3:Αριθμός LUTs και DSPs για υλοποίηση μονάδας ετεροσυσχέτισης με bit level αθροιστές και πολλαπλασιαστές.

Ο αριθμός των LUTS είναι και στις δύο περιπτώσεις διαχειρίσιμος,ο αριθμός όμως των DSPs στην πρώτη περίπτωση (χωρίς την βελτιστοποίηση της περιοδικότητας) είναι πολύ μεγαλύτερος και κατά συνέπεια αποτρεπτικός.Συνεπώς θα σταθούμε στον δεύτερο τρόπο υλοποίησης(με την βελτισοτποίηση περιοδικότητας).

8.3 Αποτελέσματα υλοποίησης ολόκληρης της αρχιτεκτονικής.

Site Type	**Used**
CLB LUTs	9617
LUT as Logic	7910
LUT as Memory	1707
LUT as Distributed RAM	0
LUT as Shift Register	1707
CLB Registers	65505
Registers as Flip Flop	65505
Register as Latch	0
CARRY8	1166

Πίνακας 4:κατανάλωση πόρων(LUTs) ολόκληρου κυκλώματος.

Site Type	**Used**
DSPs	66
DSP48E2 only	66

Πίνακας 5:κατανάλωση πόρων(DSPs) ολόκληρου κυκλώματος.

Βλέπουμε ότι οι αρχική στόχοι μας για 2% κατανάλωση πόρων πάνω στο fpga επιτεύχθηκαν καθώς

$\frac{9617}{425280}=0.022$ σε LUTs και $\frac{66}{4272}=0.015$ σε DSPs.

ΚΕΦΑΛΑΙΟ 9-ΑΠΟΤΕΛΕΣΜΑΤΑ ΠΡΟΣΟΜΟΙΩΣΗΣ

9.1 Ορισμός παραμέτρων

Υπενθυμίζουμε ότι το κύκλωμα λειτουργεί με εισόδους 16 bits και το σήμα θωρείται ότι έχει 320 σημεία και όλες οι προσωμοιώσεις έγιναν στα 550Mhz(αρχικός στόχος 500 Mhz).Τα μεταβλητά μεγέθη του κυκλώματος είναι το preamble που βρίσκεται στην μεταβλητή rom,η τιμή την οποία αφαιρούμε από το μέτρο κάθε σημείου(SUBBER) ώστε να εμφανιστούν οι αρνητικοί αριθμοί και το παράθυρο στο οποίο αναζητείται το max.Η αλλαγή της ακρίβειας των bits(από 16 σε άλλο αριθμό) είναι δυνατή όμως το κύκλωμα έχει ελεγχθεί ενδελεχώς μόνο στα 16 bits.

Για να βέλτιστη εκμετάλλευση του κυκλώματος πρέπει να καλύπτεται όλη η δυναμική περιοχή των σημάτων. Αυτο σημαίνει ότι το preamble πρέπει να κανονικοποιηθεί ώστε η μέγιστη θετική τιμή του να είναι 32767(2^15-1).Δεδουμένων των σημάτων REAL και IMAG στο [-1,1] ,με τις παρακάτω εντολές(matlab) μπορεί να παραχθεί το σωστό preamble για χρήση στην μονάδα(στην συγκεκριμένη περίπτωση 16 bit).

```
X=REAL;
Y=IMAG;
C=int32(X*(2^15-1));
D=int32(Y*(2^15-1));
ROM=int32(sqrt(double(C.^2+D.^2)));
MAGN=ROM;
ROM=MAGN-13573; %46340 είναι το max(MAGN) οπότε 46340-13573=32767
ROM(1:320)=ROM(320:-1:1);
temp = ROM;
mask = temp < 0;
temp(mask) = 2^16 + temp(mask) ;
binary = de2bi(temp, 16);
  for i=1:64
  fprintf('%d => "%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d", \n',i-
1,binary(i,1),binary(i,2),binary(i,3),binary(i,4),binary(i,5),binary(i,6),binary(i,7),binary(i,8),bin
ary(i,9),binary(i,10),binary(i,11),binary(i,12),binary(i,13),binary(i,14),binary(i,15),binary(i,16))
;
end
```

Το αποτέλεσμα αυτών των εντολών θα είναι της μορφής

```
0 => "0000000110111001",
1 => "1111011010011000",
2 => "0111111111111111",
3 => "1110111001000010",
4 => "1101100000111101",
5 => "1101100100011111",
```

6 => "1101100011011110",
7 => "1101011001001110",
8 => "0000101001100101",
9 => "1110000010100111",
10 => "1101000111110111",
11 => "1101100011101001",
12 => "1101100001101111",
13 => "1101001010001001",
14 => "1111000000101011",
15 => "1101110001010000",
16 => "1101000000010011",
17 => "1101011111100000",
18 => "1101100111100011",
19 => "1101000001111000",
20 => "1110010001101100",
21 => "1101100100100101",
22 => "1100111110111000",
23 => "1101011001001110",
24 => "1101101100010110",
25 => "1100110111110110",
26 => "1101111000100001",
27 => "1101010111101101",
28 => "1100111111101000",
29 => "1101010001011110",
30 => "1101101011101001",
31 => "1100101011111011",
32 => "1101101000010011",
33 => "1101001001111011",
34 => "1100111111100101",
35 => "1101000111111111",
36 => "1101100001001111",
37 => "1100110111110011",
38 => "1101011010111011",
39 => "1100111110101100",
40 => "1100111101101001",
41 => "1100111011111010",
42 => "1101001000010000",
43 => "1101000001000011",
44 => "1101001110001111",
45 => "1101000000101001",
46 => "1100111010011101",
47 => "1100101011111011",
48 => "1101000011111000",

```
49 => "1101000100100110",
50 => "1101000010000011",
51 => "1101001100110001",
52 => "1100111001101110",
53 => "1101000011011101",
54 => "1110101110111001",
55 => "1100111110101100",
56 => "1100111100101101",
57 => "1101011000101101",
58 => "1101001010011101",
59 => "1110000000101010",
60 => "0110000110111000",
61 => "1110101000001011",
62 => "1111100110111100",
63 => "1111001010111101",
```

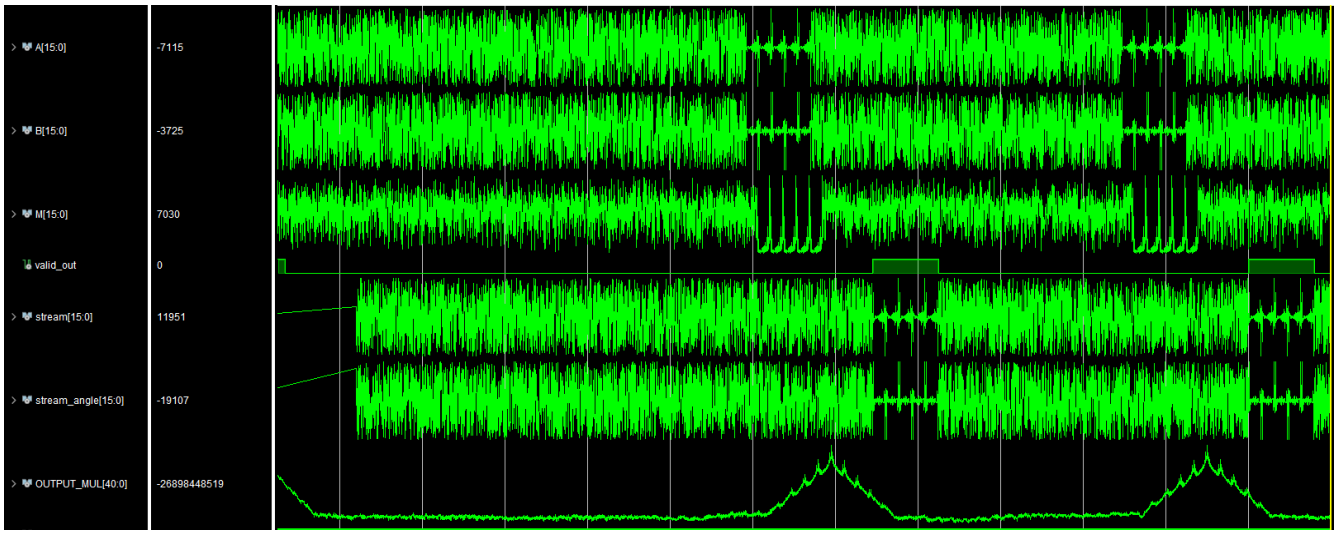
το οποίο μεταφέρεται αυτούσιο στο package helping_pack του project.

Για την τιμή που αφαιρούμε από το μέτρο(SUBBER) ισχύει 13573/2 καθώς χάνεται ένα bit απο την μετατροπή signed σε unsigned μεταξύ ρίζας και ετεροσυσχέτισης.

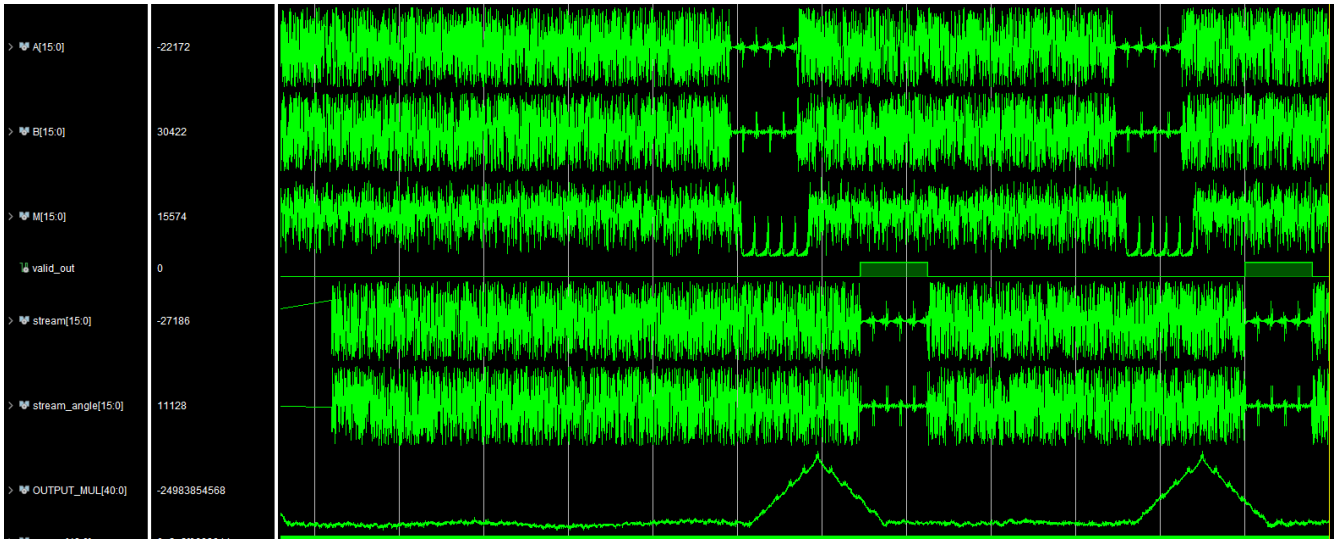
9.2 Αποτελέσματα προσομοιώσεων

Αρχικά έχουμε περιορισμένες εισόδους με λίγες εμφανίσεις του ζητούμενου σήματος.Σε κάποιες περιπτώσεις μειώνουμε σταδιακά το πλάτος του φανταστικού και του πραγματικού μέρους για να εξακριβωθεί αν το σήμα ανιχνεύεται σε δυσμενείς συνθήκες.

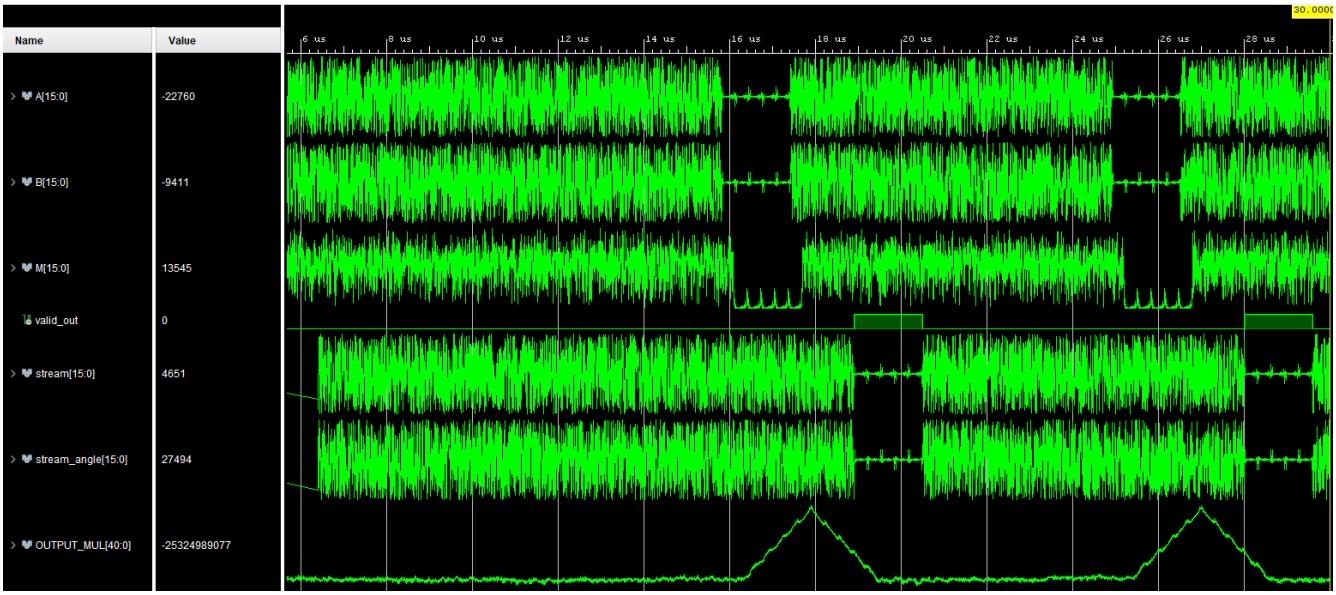
Ενδεικτικά εκτελέστηκαν κάποιες προσωμοιώσεις με ίση αλλά και μειωμένη ισχύ του ζητούμενου σήματος σε σχέση με τα υπόλοιπα σήματα στο δίαυλο.



Σχήμα 64:Το πλάτος πραγματικού και φανταστικού μέρους του σήματος είναι 1 προς 1 στα πραγμτικά και φανταστικά μέρη των άλλων σημάτων στο δίαυλο.



Σχήμα 65:Το πλάτος πραγματικό και φανταστικού μέρους του σήματος είναι 1 προς 2 στα πραγμτικά και φανταστικά μέρη των άλλων σημάτων στο δίαυλο.

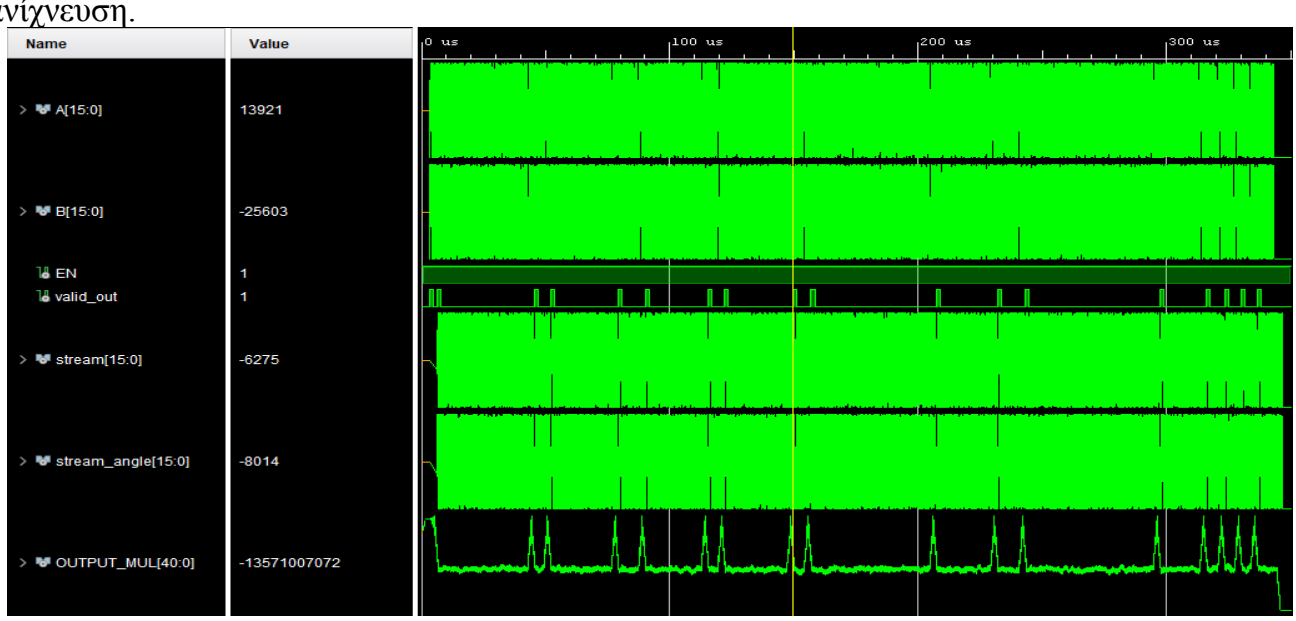


Σχήμα 66:Το πλάτος πραγματικό και φανταστικού μέρους του σήματος είναι 1 προς 4 στα πραγμτικά και φανταστικά μέρη των άλλων σημάτων στο δίαυλο.

Όπως αναφέραμε στο κεφάλαιο 7 μόλις το κύκλωμα ελέγχου ανιχνεύσει το ζητούμενο σήμα μετατρέπει την τιμή του valid_out σε 1 .Οι καταχωρητές καθυστέρησης είναι προσαρμοσμένοι έτσι ώστε την ίδια ακριβώς στιγμή να μεταδίδουν το εν λόγω σήμα.

Στην συνέχεια εισάγουμε πολύ μεγαλύτερες εισόδους και τοποθετούμε το ζητούμενο σήμα σε πολλά σημεία ώστε να διαπιστώσουμε αν το κύκλωμα λειτουργεί σωστά σε πραγματικές συνθήκες. Αρχικά ξεκινάμε με ζητούμενα σήματα ίδιου πλάτους με τον δίαυλο και σταδιακά μειώνουμε το πλάτος για να δούμε αν η εξασθένιση θα επηρεάσει πολύ την

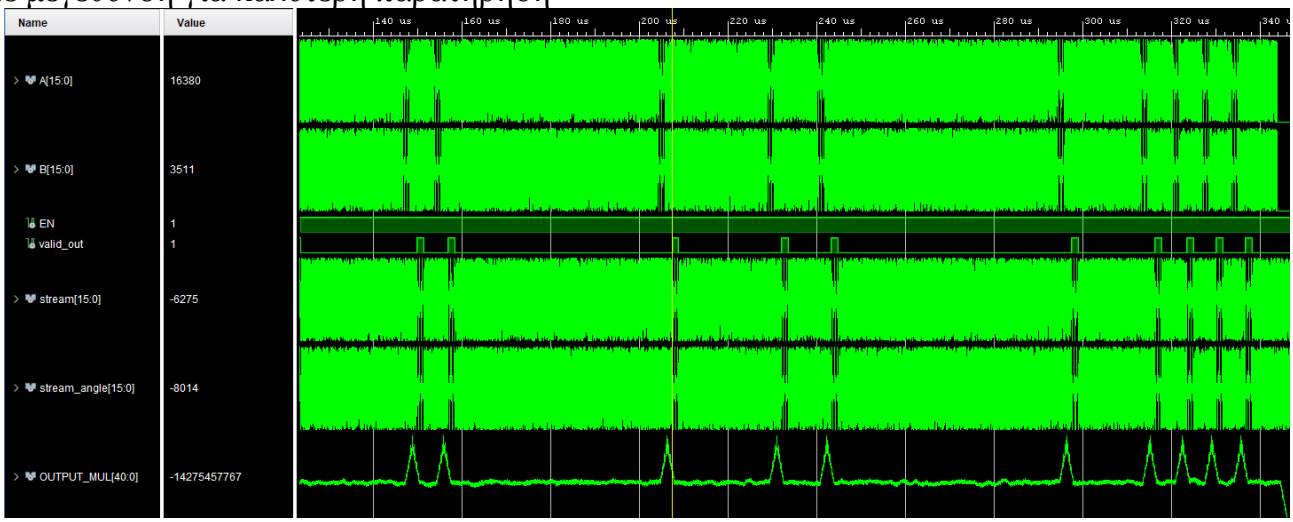
ανίχνευση.



Σχήμα 67:Μεγάλη είσοδος με ζητούμενο σήμα ίδιου πλάτους με τον υπόλοιπο δίαυλο.

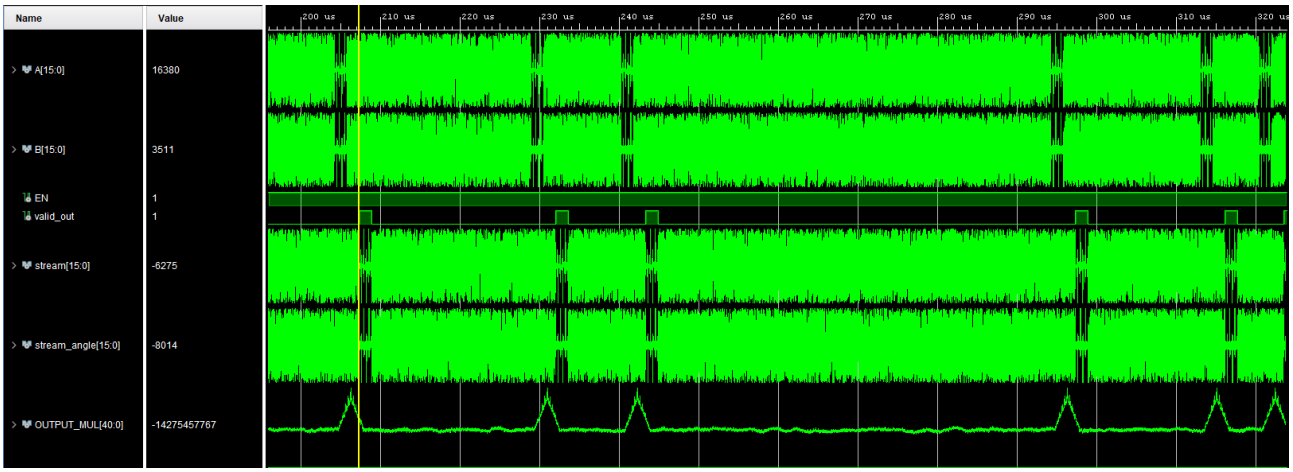
Στην έξοδος OUTPUT_MUL βλέπουμε την τιμή της ετεροσυσχέτισης οι οποία όταν το σήμα ανιχνεύεται παίρνει μεγάλη τιμή.Στη συγκεκριμένη περίπτωση το σήμα που αναζητούμε δεν φαίνεται λόγω του ίδιου πλάτους.Το valid_out γίνεται 1όταν το σήμα αναμεταδίδεται απο τις εξόδους stream και stream_angle(το valid_in παίρνει την τιμή 1 στην αρχή εσφαλμένα καθώς το κύκλωμα ετοιμάζεται να ξεκινήσει(set-up)δεν έχει λάβει εισόδους ακόμα).

Σε μεγέθυνση για καλύτερη παρατήρηση



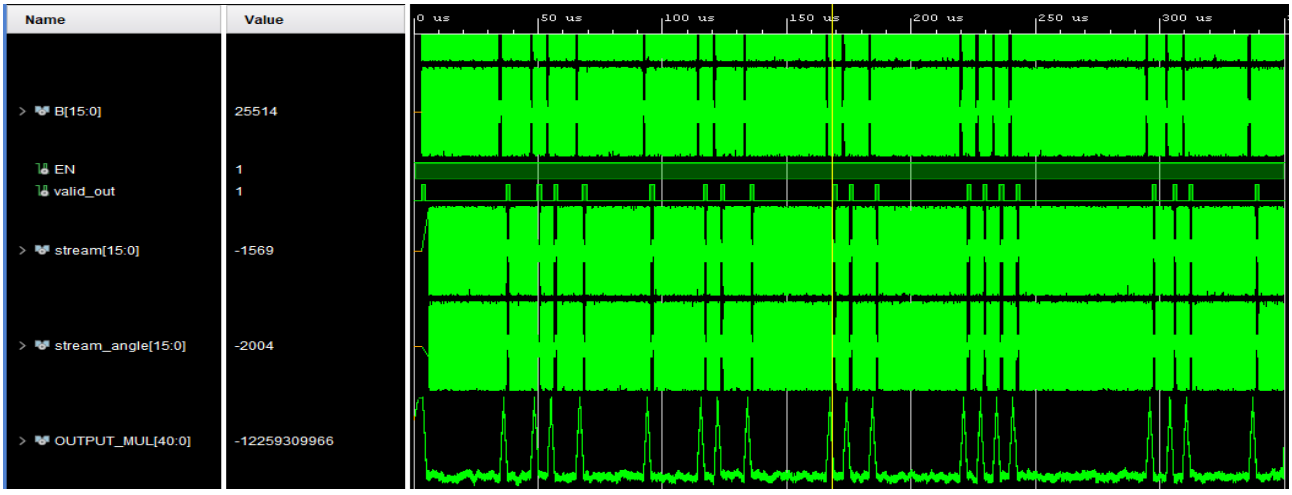
Σχήμα 68:Μεγάλη είσοδος με ζητούμενο σήμα ίδιου πλάτους με τον υπόλοιπο δίαυλο σε μεγέθυνση.

Σε ακόμα μεγαλύτερη μεγέθυνση



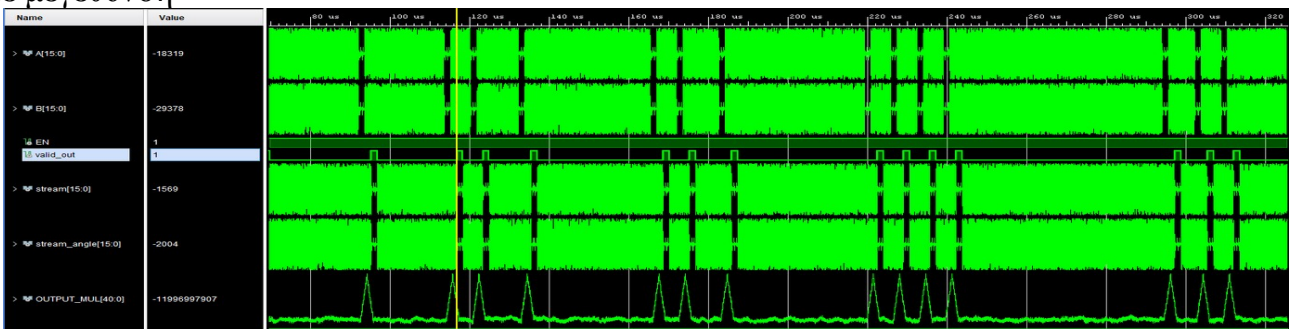
Σχήμα 69:Μεγάλη είσοδος με ζητούμενο σήμα ίδιου πλάτους με τον υπόλοιπο δίαυλο σε ακόμα μεγαλύτερη μεγέθυνση.

Στην συνέχεια μειώνουμε το πλάτος του ζητούμενου σήματος στο 1/4 και έχουμε



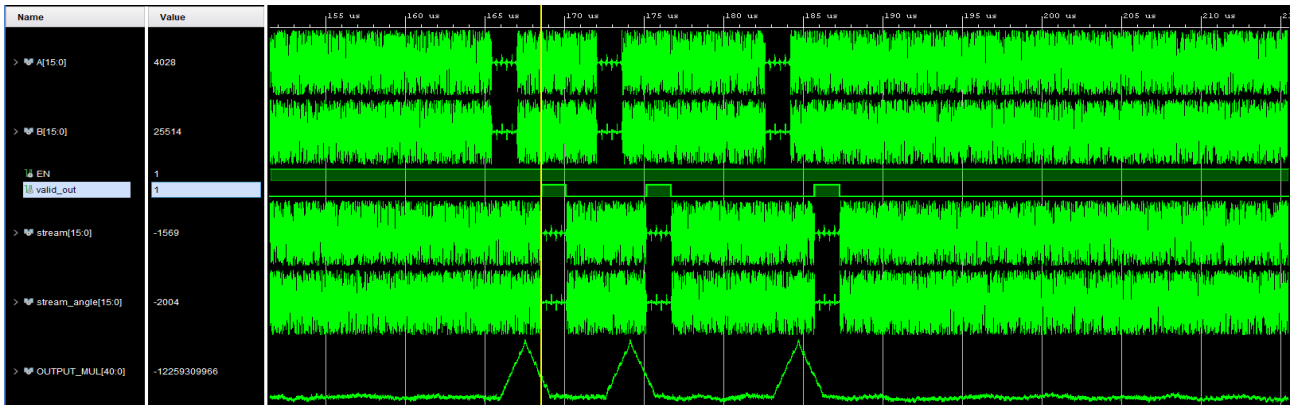
Σχήμα 70:Μεγάλη είσοδος με ζητούμενο σήμα πλάτους 1/4 σε σχέση με τον υπόλοιπο δίαυλο.

Σε μεγέθυνση



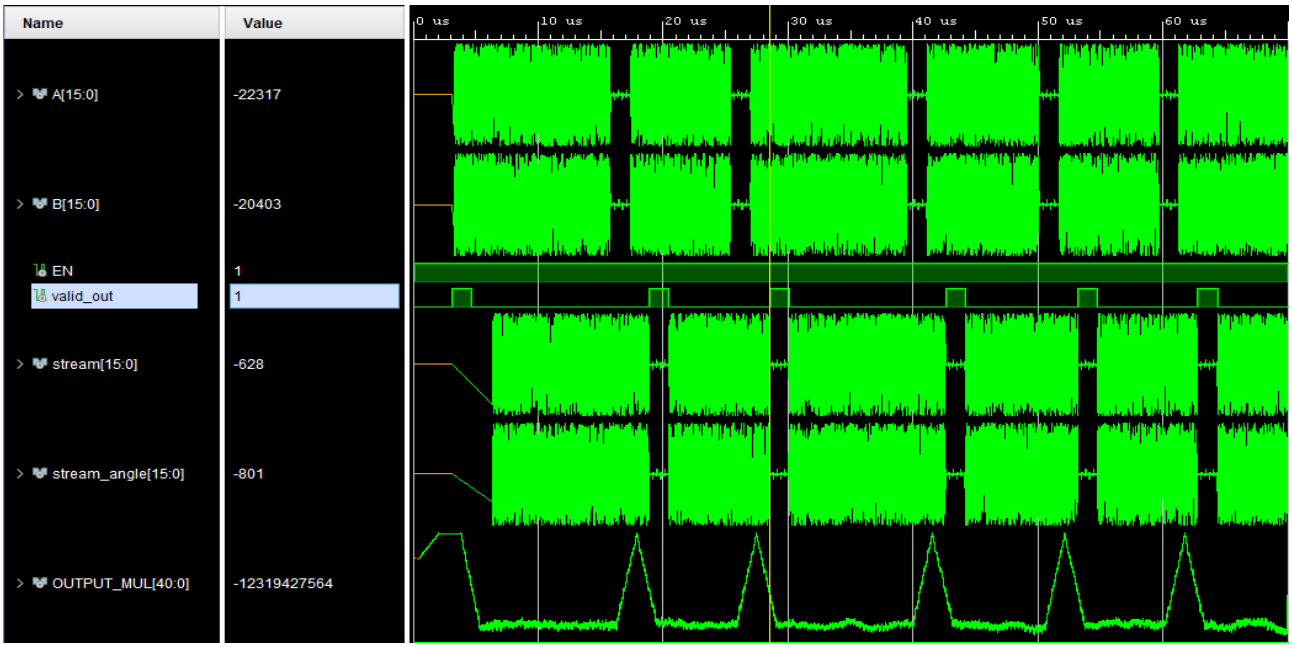
Σχήμα 71:Μεγάλη είσοδος με ζητούμενο σήμα πλάτους 1/4 σε σχέση με τον υπόλοιπο δίαυλο σε μεγέθυνση.

Σε ακόμα μεγαλύτερη μεγέθυνση



Σχήμα 72:Μεγάλη είσοδος με ζητούμενο σήμα πλάτους 1/4 σε σχέση με τον υπόλοιπο δίαυλο σε ακόμα μεγαλύτερη μεγέθυνση.

Τέλος μειώνουμε το πλάτος του ζητούμενου σήματος στο 1/10 του υπόλοιπου διαύλου και έχουμε



Σχήμα 73:Μεγάλη είσοδος με ζητούμενο σήμα πλάτους 1/10 σε σχέση με τον υπόλοιπο δίαυλο σε μεγέθυνση.

Όπως παρατηρούμε το κύκλωμα βρίσκει το σήμα κάθε φορά σε διάφορες στιγμές της μετάδοσης(valid_out='1').Η ανίχνευση γίνεται σωστά ακόμα και όταν το σήμα έχει χάσει αρκετό πλάτος.Συνεπώς η αρχιτεκτονική σαν συνολική λειτουργική μονάδα εκπληρώνει όλες τις προδιαγραφές και είναι έτοιμη για κανονική λειτουργία.

ΚΕΦΑΛΑΙΟ 10-ΓΕΝΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ

Από την εκπόνηση αυτής της διπλωματικής εργασίας προέκυψαν πολλά ενδιαφέροντα συμπεράσματα σε ότι αφορά την υλοποίηση παρόμοιων κυκλωμάτων στην VHDL.

-Ο συνδυασμός μεθόδων αποδείχθηκε η καλύτερη στρατηγική εντοπισμού σημάτων καθώς διαθέτει όλα τα πλεονεκτήματα των δύο επιμέρους μεθόδων(δεν επηρεάζεται από στροφή φάσης και είναι ανθεκτική στη εξασθένηση του ζητούμενου σήματος) τα οποία αλληλοαναιρούν τα μειονεκτήματα.

-Η κατάλληλη επιλογή παραμέτρων είναι πολύ σημαντική για τον σωστό εντοπισμό του σήματος για αυτό και δόθηκε ειδική μέριμνα έτσι ώστε να είναι εύκολα τροποποιήσημες

-Κατά την διάρκεια ανάπτυξης του κώδικα χρειάζεται μεγάλη προσοχή στο είδος των σημάτων που θα χρησιμοποιηθούν.Θα πρέπει οι είσοδοι και οι έξοδοι μεταξύ των διάφορων εξαρτημάτων να είναι σήματα ίδιου είδους καθώς η VHDL διαθέτει πολλά bugs συμβατότητας.Αυτό έχει ως αποτέλεσμα ένα εξάρτημα να λειτουργεί σωστά από μόνο του αλλά να μην λειτουργεί μέσα στο συνολικό design.Χαρακτηριστικό παράδειγμα αποτελεί η δυσκολία να εισαχθούν σωστά τα δεδομένα από ένα μονοδιάστατο πίνακα(array) διανυσμάτων(std_logic_vector) σε ένα δισδιάστατο πίνακα λογικών τιμών(std_logic).

-Ο τρόπος με το οποίο γράφουμε το κώδικα παίζει πολύ μεγάλο ρόλο στην κατανομή πόρων που θα κάνει ο synthesizer.

-Με inference μόνο ο πολλαπλασιαμός χρησιμοποιέι DSP.Η πρόσθεση τοποθετείται μέσα σε DSP μόνο αν ακολουθεί η έπεται πολλαπλασιασμού.

-Πολύ σημαντικός είναι ο έλεγχος κάθε εξαρτήματος ξεχωριστά πριν χρησιμοποιηθούν στο γενικό design ,καθώς μέτα την προσθήκη πολλών εξαρτημάτων η αρχιτεκτονική γίνεται ιδαίτερα πολύπλοκη ,και η διόρθωση τυχών bugs στα επημέρους τμήματα γίνεται απο πολύ χρονοβόρα εώς και αδύνατη.

-Απολύτως απαραίτητη είναι και η ανάπτυξη κάποιου είδους μηχανισμού για την δημιουργία μεγάλων εισόδων ,και τον έλεγχο των αντίστοιχων εξόδων καθώς μόνο έτσι μπορεί να διασφαλιστεί η ορθή λειτουργία του κυκλώματος.Πολύπλοκες αρχιτεκτονικές δεν είναι δυνατό να ελεγχθούν επαρκώς με μικρές εισόδους.Στην δική μας περίπτωση ο έλεγχος έγινε με την βοήθεια του matlab,όπου προσωμοιώθηκε η λειτουργεία του κυκλώματος.Με την βοήθεια των εργαλείων του matlab δημιουργήθηκαν μεγάλες είσοδοι που στην συνέχεια εισίχθησαν στο vivado και στο matlab.
Από τα αποτελέσματα των δύο προσωμοιώσεων ήταν πολύ εύκολο να αποφανθούμε για το αν η αρχιτεκτονική που χτίσαμε στο vivado δουλεύει σωστά .

-Ο υπολογισμός της καθυστέρησης(delay) είναι πολύ σημαντικός για τον συγχρονισμό των διαφορετικών τμημάτων της αρχιτεκτονικής και είναι σχεδόν πάντα καλύτερο να υπολογίζεται μέσα από μια πληθώρα επαναλαμβανόμενων εκτελέσεων.Με την αλλαγή των παραμέτρων και την άμεση εκτέλεση προσωμοίωσης μπορούμε τις περισσότερες φορές να έχουμε μια πολύ καλή εικόνα για το πως ακριβώς μεταβάλλεται η καθυστέρηση.

-Παρόλο που οι απαιτήσεις του έργου ήταν ιδιαίτερα απαιτητικές,2% κατανάλωση πόρων στο fpga και σωστή λειτουργία στα 500Mhz, η υλοποίηση του κυκλώματος ήταν επιτυχής.Το κύκλωμα,αν και θεωρητικά μεγάλο,με κατάλληλο προσανατολισμό στον κώδικα και με σωστή εκμετάλλευση των πόρων του fpga κατέληξε να καταλαμβάνει ένα πολύ μικρό ποσοστό του fabric.

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ ΚΑΙ ΠΙΝΑΚΩΝ

ΣΧΗΜΑΤΑ

ΠΙΝΑΚΕΣ

ΒΙΒΛΙΟΓΡΑΦΕΙΑ

[1] Sample code source:http://www.secs.oakland.edu/~llamocca/arithcores.html

[2]Sample code source:https://dadorran.wordpress.com/2014/04/25/cross-correlation-demo/

[3] Image 1 URL:https://www.xilinx.com/support/documentation/data_sheets/ds889-zynq-usp-rfsoc-overview.pdf

[4] Image 2 URL:https://en.wikipedia.org/wiki/Cross-correlation

[5] Image 45 URL: https://www.xilinx.com/content/xilinx/en/products/technology/dsp/_jcr_content/imageTabParsys/tab-solution/xilinxaccordion2/tab-HardwareDesigner/xilinximage_b361.img.png/1499296098380.png

[6] Image 47 URL:
https://www.elprocus.com/wp-content/uploads/2015/06/Logical-Structure-of-FIR-Filter.jpg

[7] Image 52 URL:
https://www.researchgate.net/profile/Masanori_Hariyama/publication/273835293/figure/fig3/AS:391925313097747@1470453684858/Pipelined-4-bit-ripple-carry-adder.png

[8] Image 54 URL:
https://www.electronicshub.org/wp-content/uploads/2015/06/4-bit-binary-multiplier.jpg

[9] Image 57 URL:
https://www.xilinx.com/content/xilinx/en/products/technology/dsp/_jcr_content/imageTabParsys/tab-solution/xilinxaccordion2/tab-HardwareDesigner/xilinximage_b361.img.png/1499296098380.png

Υπόλοιπα σχήματα φτιάχτηκαν με:
[9] online flowchart maker URL: https://www.draw.io/
[10] MATLAB :https://www.mathworks.com