ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ

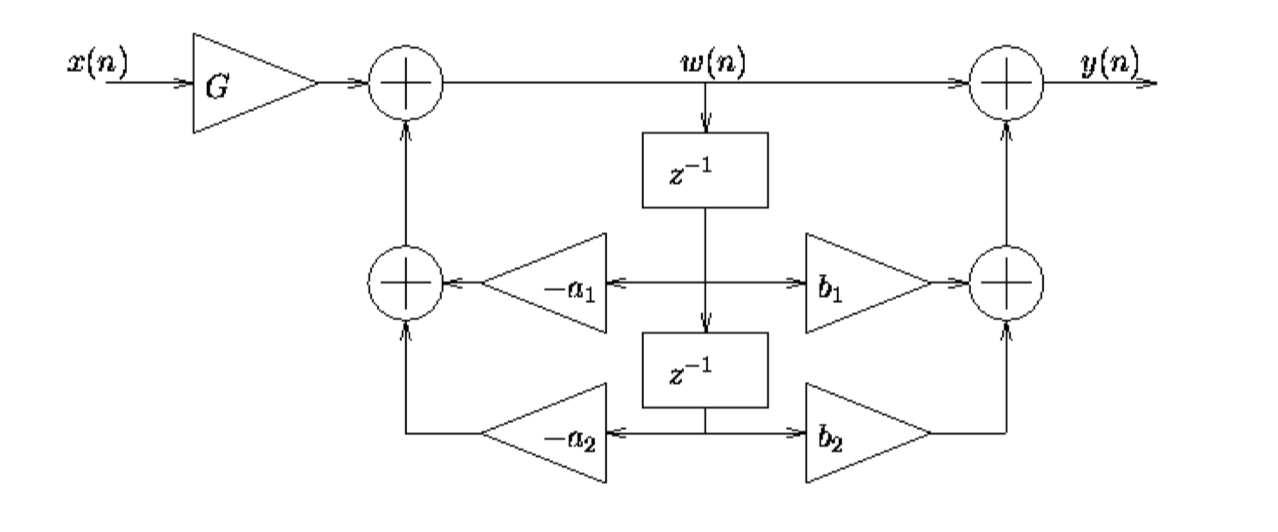
## Εργαστήριο 6

## Σχεδιασμός και υλοποίηση IIR φίλτρων

|  |
| --- |
| Ομάδα 17 14-12-2018 |
| Ασημακόπουλος Κωνσταντίνος 1046966 |
| Λουκαρέας Παύλος 1046970 |

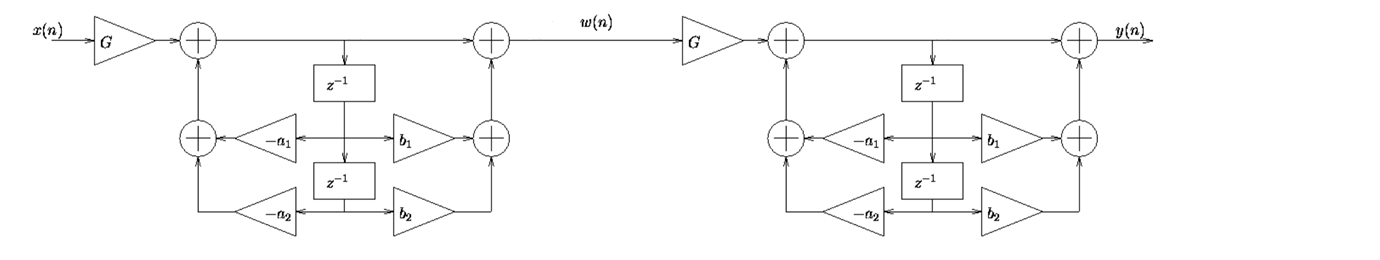
**Περίληψη**

Σε αυτή την εργαστηριακή άσκηση θα εξοικειωθούμε με θέματα που έχουν σχέση με την υλοποίηση σε επεξεργαστή σταθερής υποδιαστολής ενός IIR συστήματος 4ης τάξης. Η υλοποίηση θα γίνει με διασύνδεση δύο συστημάτων δεύτερης τάξης που χρησιμοποιούν την τύπου άμεσης δομής ΙΙ διάταξη.



Εικόνα 1 μπλοκ διάγραμμα της τύπου άμεσης δομής ΙΙ διάταξης

Στην άσκηση θα υλοποιήσουμε σε γλώσσα C ένα φίλτρο IIR 4ης τάξης λαμβάνοντας υπόψη την αριθμητική σταθερής υποδιαστολής του επεξεργαστή. Αρχικά, για λόγους μεγαλύτερης ευστάθειας και καλύτερο έλεγχο του συστήματος θα υλοποιήσουμε ως cascade δύο φίλτρων 2ης τάξης. Η σύνδεση του συστήματος που θα υλοποιήσουμε θα είναι η παρακάτω:



Εικόνα 2 Η διασύνδεση του συστήματος που υλοποιεί ο κώδικας

**Άσκηση 6.1**

Από τη συνάρτηση μεταφοράς (1) της άσκησης προκύπτει με τη χρήση των ιδιοτήτων του μετασχηματισμού Ζ η παρακάτω εξίσωση διαφορών για το κάθε δεύτερης τάξης σύστημα:

**y[n] + a1 \* y[n – 1] + a2 \* y[n – 2] = G \* (x[n] + b1 \* x[n – 1] + b2 \* x[n - 2]) (2)**

Με χρήση της Matlab και της παρακάτω εντολής εξάγουμε τους συντελεστές του τεταρτοβάθμιου ελλειπτικού φίλτρου ([Β, Α] = ellip(4, 0.25, 10, 0.25)). Στη συνέχεια για την εξαγωγή των συντελεστών των δευτεροβάθμιων φίλτρων ακολουθούμε τα παρακάτω βήματα:

* Βρίσκουμε τους πόλους και τα μηδενικά του τέταρτης τάξης συστήματος χρησιμοποιώντας τη συνάρτηση roots() για τους Β και Α πίνακες αντίστοιχα. Πόλοι {0.6715 ± j0.7012}, {0.4759 ± j0.4613}, Μηδενικά {0.2697 ± j0.9629}, {0.6684 ± j0.7438}
* Για το πρώτο φίλτρο εξάγουμε το πολυώνυμο χρησιμοποιώντας το πρώτο ζευγάρι πόλων και μηδενικών. Συγκεκριμένα με τη συνάρτηση poly() βρίσκουμε τον αριθμητή και τον παρονομαστή της συνάρτησης μεταφοράς.
* Από αυτή προκύπτουν οι συντελεστές του πρώτου φίλτρου, οι οποίοι είναι οι παρακάτω:

a1 = -1.3432, a2 = 0.9426, b1 = -0.5394, b2 = 0.9999

* Μετατρέπουμε σε μορφή Q15 χρησιμοποιώντας τη συνάρτηση twocomplement που μας δόθηκε. Όσοι συντελεστές είναι μεγαλύτεροι ή ίσοι κατά απόλυτη τιμή με τη μονάδα διαιρούνται με δύο πριν τη μετατροπή. Ξαναπολλαπλασιάζεται με δύο κατά την εκτέλεση του κύριου προγράμματος όταν συμβαίνει το παραπάνω.
* Ακόμα για τους συντελεστές a για να τους μεταφέρουμε απευθείας στο άλλο μέλος της εξίσωσης τους αλλάζουμε πρόσημο πριν τη μετατροπή σε Q15.
* Οι συντελεστές που προκύπτουν είναι:

a1 = 22006, a2 = 34648, b1 = 47860, b2 = 32764

* Για τον υπολογισμό του κέρδους G1 ακολουθούμε την παρακάτω διαδικασία:

Με χρήση της συνάρτησης h = filter(B, A, t) υπολογίζουμε την κρουστική απόκριση για τα πρώτα 500 δείγματα και έπειτα υπολογίζουμε την ακριβή τιμή με την εντολή G = 1/abs(sum(h))

Η τιμή που προκύπτει είναι G1 = 0.4104 (Q15: 13447)

* Για το δεύτερο φίλτρο ακολουθούμε την ίδια διαδικασία και προκύπτουν οι τιμές
* -a1 = 0,9518(Q15: 31188), -a2 = -0.4393 (Q15: 51141), b1 = -1.3368/2 = 0.6684 (Q15: 43633), b2 = ½ = 0.5 (Q15: 16384), G2 = 0.7350 (Q15: 24084)

**Άσκηση 6.2**

Στο ερώτημα αυτό μας ζητήθηκε να υλοποιήσουμε ένα φίλτρο 4ης τάξης χρησιμοποιώντας δύο φίλτρα 2ης τάξης σε σειρά. Η υλοποίηση έγινε χρησιμοποιώντας fixed point μεταβλητές για λόγους ταχύτητας και επειδή η αρχιτεκτονική της πλακέτας δεν επιτρέπει floating point πράξεις.

1. Αρχικά υλοποιήσαμε σε γλώσσα C ένα φίλτρο 2ης τάξης. Χρησιμοποιώντας τους συντελεστές που υπολογίσαμε παραπάνω οι οποίοι είναι σε μορφή Q15 και υλοποιώντας τη συνάρτηση Q15­\_Mult για τον πολλαπλασιασμό δεδομένων σε αυτή τη μορφή, υλοποιήσαμε το φίλτρο 2ης τάξης. Οι συναρτήσεις που περιγράφουμε είναι οι παρακάτω.

short int Q15\_Mult(short int coeff, short int data) {

    int temp;

    short int Q;

    temp = (int)coeff \* (int) data;

    Q = (short) temp >> 15;

    return Q;

}

void First\_2nd(int data) {

    xn = (short)data;

    short int Gb11  = Q15\_Mult(G11, b11), Gb21 = Q15\_Mult(G11, b21);

    // Calculate the output of the first filter

    wn = (Q15\_Mult(2\*a11, wn\_1)) +

(Q15\_Mult(a21, wn\_2)) + Q15\_Mult(G11, xn) +

Q15\_Mult(Gb11, xn\_1) + Q15\_Mult(Gb21, xn\_2);

    wn\_2 = wn\_1;   // Save current values

    wn\_1 = wn;

    xn\_2 = xn\_1;

    xn\_1 = xn;

}

1. Σε αυτό το ερώτημα θέλουμε να υλοποιήσουμε το 4ης τάξης φίλτρο. Για να το κάνουμε αυτό υλοποιούμε ένα ακόμα 2ης τάξης φίλτρο χρησιμοποιώντας ξανά τους συντελεστές που βρήκαμε πριν. Ο τρόπος υλοποίησης είναι παρόμοιος με το προηγούμενο ερώτημα

void Second\_2nd(short int wn) {

  short int Gb12  = Q15\_Mult(G2, 2 \* b12), Gb22 = Q15\_Mult(G2,2 \* b22);

    // Calculate the output of the second filter

    yn = (Q15\_Mult(a12, yn\_1)) + (Q15\_Mult(a21, yn\_2)) +

Q15\_Mult(G2, wn) + Q15\_Mult(Gb12, wn\_1) + Q15\_Mult(Gb22, wn\_2);

    yn\_2 = yn\_1;           // Save current values

    yn\_1 = yn;

    wn\_2 = wn\_1;

    wn\_1 = wn;

}

Η έξοδος του συνολικού φίλτρου είναι 16-bit, αλλά στην έξοδο στο κανάλι ήχου πρέπει να δώσουμε 32-bit έξοδο. Οπότε πριν γράψουμε την έξοδο κάνουμε αντιγραφή των 16-bit σε έναν 32-bit αριθμό που αντιστοιχεί σε δεξί και αριστερό κανάλι. Η συνάρτηση που εξυπηρετεί τη διακοπή είναι η παρακάτω.

interrupt void serial\_port\_rcv\_isr() {

    data = input\_leftright\_sample();

    // "data" contains both audio channels

    // ---------------------- Useful code goes here ----------------------

    First\_2nd(data);

    Second\_2nd(wn);

    int temp = (int)yn;

    temp = temp << 16;

    data = temp | (int) yn;

    // -------------------------------------------------------------------

    // process "data", or pass another variable to change the output

    output\_leftright\_sample(data);

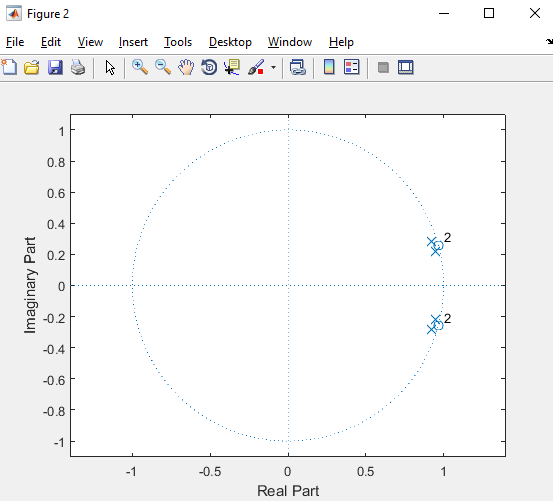
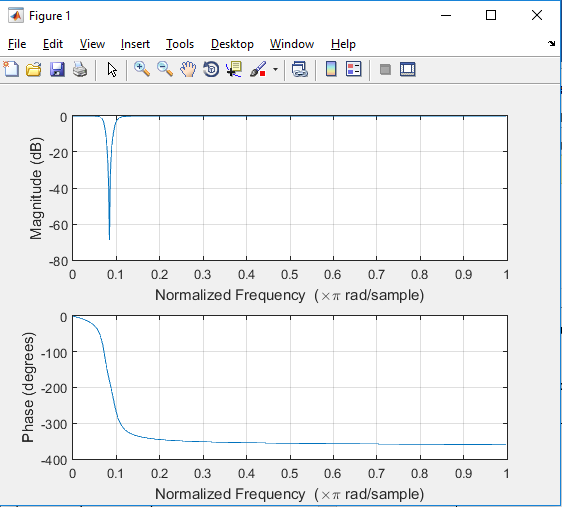
}

Σημειώνεται ότι οι είσοδοι, έξοδοι και συντελεστές των φίλτρων έχουν οριστεί ως global μεταβλητές.

**Άσκηση 6.3**

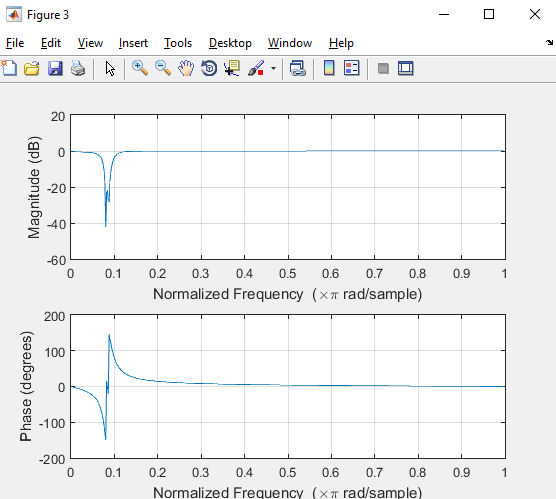
Σε αυτό το ερώτημα μας ζητείται να υλοποιήσουμε ένα φίλτρο εγκοπής 4ης τάξης με τη χρήση MATLAB. Οι συχνότητες αποκοπής θα πρέπει να είναι από μια κανονικοποιημένη συχνότητα 0.07 έως 0.1. Η εντολή με την οποία θα εκτελέσουμε τη συνάρτηση του φίλτρου είναι

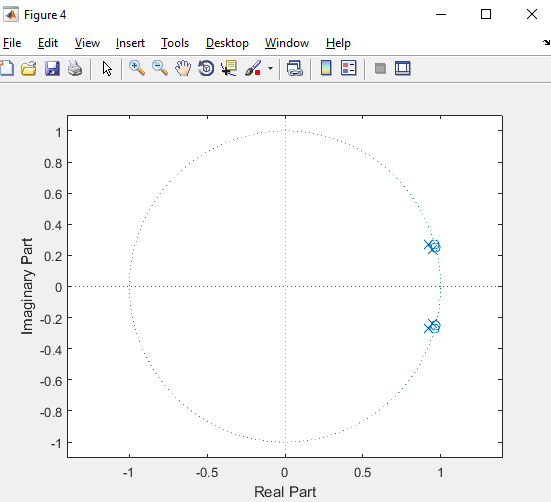
[Β, Α] = butter(2, [0.07 0.1], ‘stop’)

Η παραπάνω εντολή δημιουργεί συντελεστές που δεν είναι κβαντισμένοι. Παρακάτω βλέπουμε την απόκριση συχνότητας και το διάγραμμα πόλων και μηδενικών.

Με τη χρήση της συνάρτησης quant() της MATLAB κβαντίζουμε τους συντελεστές του αριθμητή και του παρονομαστή και στη συνέχεια με παρόμοιο τρόπο εξάγουμε την απόκριση συχνότητας και το διάγραμμα πόλων και μηδενικών.

A\_q = quant(A), B\_q = quant(B)





Παρατηρούμε τη δημιουργία δύο κορυφών μέσα στην απόκριση συχνότητας του κβαντισμένου φίλτρου, ενώ αλλάζει και η φάση κοντά στην εγκοπή του φίλτρου. Αυτό συμβαίνει διότι υπάρχει μετακίνηση των πόλων και των μηδενικών λόγω των κβαντισμών που εκτελέσαμε στους συντελεστές, όπως και σφάλματα στρογγυλοποίησης.

Υλοποιώντας με τον ίδιο τρόπο ένα κβαντισμένο φίλτρο 4ης τάξης ως σειριακά κβαντισμένα φίλτρα 2ης τάξης θα πάρουμε πολύ καλύτερα αποτελέσματα. Ο λόγος για τον οποίο συμβαίνει αυτό είναι ότι τα σφάλματα σε αυτή την περίπτωση είναι μικρότερα για τα δυο φίλτρα ξεχωριστά, οπότε η ανάδραση δεν ενισχύει την κατάσταση αυτή.

