ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΣΗΜΕΙΩΣΕΙΣ ΜΑΘΗΜΑΤΟΣ

ΕΦΑΡΜΟΓΕΣ ΤΗΣ ΨΗΦΙΑΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ

ΠΑΤΡΑ 2007

ΕΡΓΑΣΤΗΡΙΑΚΗ 1 ΑΣΚΗΣΗ

Υλοποίηση FIR Φίλτρων

1. Εισαγωγή

Στα πλαίσια αυτής της άσκησης θα υλοποιηθούν ψηφιακά FIR φίλτρα στην αναπτυξιακή κάρτα *TMS320C6711 DSK*. Ο σκοπός της άσκησης συνοψίζεται στα ακόλουθα σημεία :

- Εξοικείωση με το περιβάλλον προγραμματισμού του επεξεργαστή σήματος κινητής υποδιαστολής TMS320C6711.
- Εξοικείωση με τον προγραμματισμό και το χειρισμό των περιφερειακών συσκευών του.
- Υλοποίηση FIR φίλτρων σε πραγματικό χρόνο.

2. Η TMS320C6711 DSK Κάρτα και το CCS v2.2

2.1. Γενικά Χαρακτηριστικά

Η αναπτυξιακή κάρτα που φαίνεται στο Σχήμα 1, αναπτύχθηκε από την *Texas Instruments* (*TI*) με σκοπό να προσφέρει ένα φιλικό στο χρήστη αναπτυξιακό περιβάλλον χαμηλού κόστους. Τα χαρακτηριστικά υψηλής απόδοσης που διαθέτει εξασφαλίζονται από τις δυνατότητες που παρέχει ο επεξεργαστής στον οποίο βασίζεται και ο οποίος την καθιστά μια από τις πιο αποτελεσματικές αναπτυξιακές πλατφόρμες.

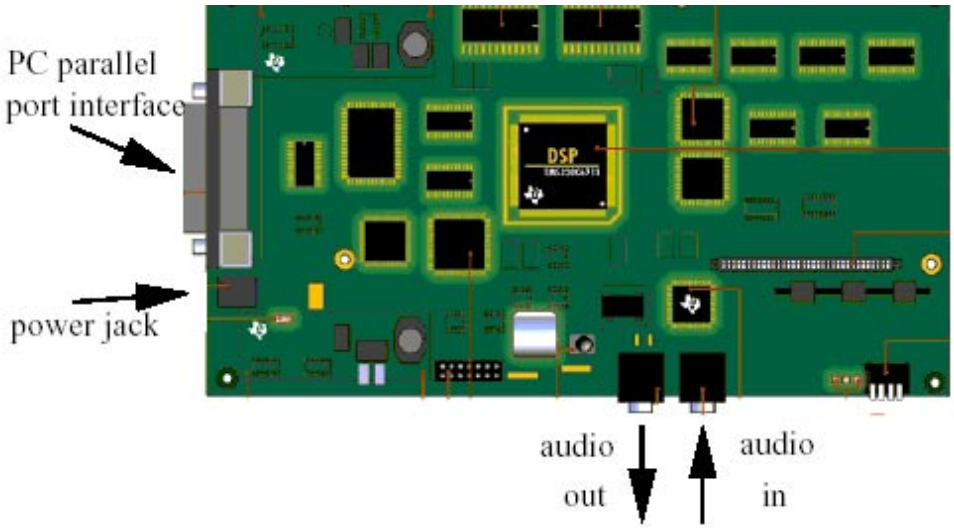
Το *DSK* αλληλεπιδρά με τον ξενιστή υπολογιστή μέσω της παράλληλης θύρας επιτρέποντας έτσι την αποδοτική ανάπτυξη και τον εύκολο έλεγχο των εφαρμογών του χρήστη. Παρακάτω παρουσιάζονται, επιγραμματικά, τα βασικά δομικά στοιχεία από τα οποία αποτελείται η πλατφόρμα καθώς και βασικά χαρακτηριστικά της :

- Τον ψηφιακό επεξεργαστή σήματος TMS320C6711, ικανό να εκτελεί *900* εκατομμύρια εντολές κινητής υποδιαστολής το δευτερόλεπτο (*MFLOPS*)

- Υποστήριξη διπλού σήματος χρονισμού, ένα για την *CPU* στα *150 MHz*,κι ένα για την προσαρμοστική μονάδα εξωτερικής μνήμης (*EMIF*) στα *100MHz*
- Ελεγκτή παράλληλης θύρας (*PPC*) που είναι υπεύθυνος για τον έλεγχο της προσαρμοστικής μονάδας της παράλληλης θύρας του ξενιστή υπολογιστή (*host PC*)
- 16M Bytes των 100MHz σύγχρονη δυναμική RAM (*SDRAM*)
- 128K Bytes ROM
- Θύρα εισόδου/εξόδου *8bit* (*memory – mapped*)
- Προσαρμοστική μονάδα ξενιστή (*Host Port Interface*) που εξασφαλίζει μέσω της παράλληλης θύρας πρόσβαση σε όλα τα είδη μνημης του επεξεργαστή
- Κωδικοποιητή/Αποκωδικοποιητή ηχητικών (audio) σημάτων των *16 bits*
- Έξι (6) φωτεινές ενδεικτικές διόδους (*LEDs*) για διάφορες λειτουργίες

Ο επεξεργαστής διασυνδέεται με τις παραπάνω περιφερειακές συσκευές μέσω της θύρας *memory expansion* και των δύο σειριακών θυρών *McBSP0* και *McBSP1* (*M*ultichannel *B*uffered *S*erial *P*orts).

Πάνω στη σειριακή θύρα *McBSP0* είναι συνδεδεμένη η περιφερειακή συσκευή *AD535*, στην οποία υπάρχουν οι μετατροπείς Αναλογικό-σε-Ψηφιακό (Α-Ψ) και Ψηφιακό σε Αναλογικό (Ψ-Α), οι οποίοι βασίζονται στη διαμόρφωση Σίγμα-Δέλτα (για περισσότερες λεπτομέρειες δες το Εγχειρίδιο Χρήστη του DSK). Η συσκευή *AD535* μπορεί να χρησιμοποιηθεί για είσοδο και έξοδο ακουστικών σημάτων (mono), χρησιμοποιεί μια σταθερή συχνότητα δειγματοληψίας 8 *KHz* και μήκος λέξης 16 *bits*.



Σχήμα 1: Άποψη της Αναπτυξιακής κάρτας DSP Starter Kit (DSK).

2.2. Αρχικοποίηση & Έλεγχος Σωστής Λειτουργίας της Κάρτας

Για να γίνουν οι σωστές αρχικοποιήσεις (Initialization) του επεξεργαστή και όλων των περιφερειακών συσκευών του χρειάζεται να εκτελέσετε τα ακόλουθα απλά βήματα :

Βήμα 1: Συνδέστε τη κάρτα *DSK*, μέσω της παράλληλης θύρας, στο *host PC* και στην τροφοδοσία ρεύματος. Μετά την τροφοδοσία, τα *LEDs* θα πρέπει να αναβοσβήνουν επιβεβαιώνοντας έτσι την ορθή λειτουργία της.

Βήμα 2: Επιλέγοντας το εικονίδιο *CCS 2 (C'6000)* στην επιφάνεια εργασίας ενεργοποιείται το *Code Composer Studio*. Πρόκειται για ένα ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού για ψηφιακούς επεξεργαστές της εταιρείας *Texas Instruments*. Επισημαίνεται ότι αν χρειαστεί να γίνει *reset* η κάρτα θα πρέπει, ανάλογα με το αν το *CCS* είναι ή δεν είναι σε κατάσταση run, να κάνετε μία από τις παρακάτω ενέργειες:

Ενέργεια 1: Αν το *CCS* είναι σε κατάσταση run, επιλέγουμε : Debug > Reset DSP .

Ενέργεια 2: Αν το *CCS* δεν είναι σε κατάσταση run, κλείνουμε το *CCS* και κάνουμε *reset* την κάρτα είτε πατώντας το πλήκτρο *reset,* είτε διακόπτοντας την τροφοδοσία της κάρτας για μερικά δευτερόλεπτα και ενεργοποιώντας στη συνέχεια το *CSS*.

Βήμα 3: Για τον έλεγχο της ορθής λειτουργίας του *DSK* επιλέγουμε στο *CCS* περιβάλλον: GEL > Check DSK > QuickTest. Κατά τη διάρκεια αυτού του ελέγχου τα LEDs στην κάρτα θα πρέπει να αναβοσβήνουν κι ο *επεξεργαστής* βρίσκεται σε κατάσταση *self-test*.

2.3. Δημιουργία Αρχείου Project

Για να δημιουργήσετε στο περιβάλλον *CCS* ένα αρχείο *project* (επέκταση *.pjt*) θα πρέπει να ακολουθήσετε τα παρακάτω βήματα:

Βήμα 1: Δημιουργείστε ένα νέο *project* με την ονομασία *askisi_1.pjt*, επιλέγοντας: Project > New και αποθηκεύστε το, με το όνομα *askisi1*, στο *directory C:\E_PS_E_S*.

Σαν Project Type: θέτουμε *Executable (.out)* ενώ ως Target επιλέγουμε *TMS320C67XX*.

Βήμα 2: Για τη σωστή εκτέλεση της άσκησης, θα πρέπει να αντιγράψετε τα αρχεία που υπάρχουν στον φάκελο *C:\E_PS_E_S\yliko1* στο φάκελο *C:\E_PS_E_S\askisi1*.

Βήμα 3: Στη συνέχεια επιλέξτε: Project > Add Files to Project ... και στο παράθυρο που ανοίγει πάμε στο *directory C:\E_PS_E_S\askisi1*. Στο πεδίο Files of type: επιλέγουμε C Source Files (*.c,*.ccc) και στο πεδίο File name: το αρχείο *loopback*.

Βήμα 4: Επαναλάβετε δύο (2) φορές το Βήμα 3 και:

1) τη πρώτη φορά στο πεδίο Files of type: επιλέξτε Asm Source Files (*.a*) και File name: *vectors*

2) τη δεύτερη, στο πεδίο Files of type: Linker Command File (*.cmd) ενώ στο πεδίο File name: *lnk*.

Βήμα 5: Για να ολοκληρώσετε τη δημιουργία του αρχείου *project*, επιλέξτε Project > Scan All Dependencies.

2.4. Οι Διαδικασίες Build, Load και Run

Για να κάνουμε *Build* το *project* χρειάζονται να εκτελέσουμε τις διαδικασίες *compile* και *link*. Για το σκοπό αυτό, επιλέγουμε : Project > Rebuild All. Μόλις το *CCS* ολοκληρώσει επιτυχώς τα δύο παραπάνω βήματα, θα δημιουργηθεί το εκτελέσιμο αρχείο *askisi1.out* το οποίο αποθηκεύεται στον υποκατάλογο *C:\E_PS_E_S\askisi1\Debug*. (Ουσιαστικά το αρχείο .out είναι το μεταγλωττισμένο εκτελέσιμο αρχείο σε γλώσσα κατανοητή από τον DSP).

Επόμενο βήμα στη διαδικασία για την εκτέλεση του προγράμματος μας είναι να γίνει *Load* το εκτελέσιμο πρόγραμμα *askisi1.out* που δημιουργήσαμε. Αυτό γίνεται ακολουθώντας τη παρακάτω διαδικασία:

Βήμα 1: Επιλέγουμε : File > Load Program ... και επιλέγουμε το αρχείο *askisi1.out* που δημιουργήσαμε προηγουμένως.

Βήμα 2: Τρέχουμε το πρόγραμμα, επιλέγοντας : Debug > Run .

Βήμα 3: Η εκτέλεση του προγράμματος σταματά επιλέγοντας: Debug > Halt.[1]

3. Το πρόγραμμα *loopback*

Το πρώτο πρόγραμμα που θα εκτελέσετε στα πλαίσια της άσκησης αυτής θα υλοποιεί ένα σύστημα δειγματοληψίας και ανακατασκευής ενός ακουστικού σήματος. Για το σκοπό αυτό θα χρησιμοποιηθούν οι μετατροπείς Α-Ψ και Ψ-Α της συσκευής *AD535*. Είναι προφανές ότι στην περίπτωση αυτή ο DSP δεν θα κάνει καμία επεξεργασία στο σήμα εισόδου και θα περιοριστεί μόνο στο ρόλο του διακομιστή. Το πρόγραμμα που υλοποιεί το παραπάνω σύστημα είναι το πρόγραμμα *loopback* το οποίο και θα σας δοθεί. Στη συνέχεια θα αναλύσουμε τον κώδικα αυτού του προγράμματος, με την βοήθεια του οποίου θα γίνουν κατανοητές κάποιες βασικές λειτουργίες της αναπτυξιακής πλατφόρμας επεξεργασίας σήματος. Πριν γίνει οποιαδήποτε επεξεργασία των δεδομένων, θα πρέπει να γίνουν κάποιες απαραίτητες αρχικοποιήσεις (θα πρέπει να δοθούν κάποιες συγκεκριμένες τιμές) στους απαραίτητους καταχωρητές μέσω των οποίων καθορίζεται η μετέπειτα λειτουργία, συμπεριφορά και επικοινωνία της κάρτας.

Έτσι στο βρόχο της συνάρτησης *main*( ), αρχικά ορίζεται ένας πίνακας (out) του οποίου οι 16 πρώτες τιμές (οι οποίες και παρατίθενται παρακάτω) θα αποτελέσουν τις τιμές αρχικοποίησης των καταχωρητών ελέγχου της περιφερειακής συσκευής *AD535*. Οι καταχωρητές αυτοί διαβάζονται ή γράφονται κατά τη δευτερεύουσα σειριακή επικοινωνία και προγραμματίζουν τόσο τις επιλογές όσο και τις ρυθμίσεις του κυκλώματος του codec είτε για το κανάλι φωνής, ή για το κανάλι δεδομένων. Συγκεκριμένα μέσω των τιμών του out[3] = 0x0386 και του out[7] = 0x0306 εγγράφεται ο καταχωρητής ελέγχου 3 (Control Register 3). Ο καταχωρητής ελέγχου 3 καθορίζει κάποιες παραμέτρους αρχικοποίησης για τη μετάδοση και την επικοινωνία στο κανάλι φωνής, καθώς και κάποια τεχνικά χαρακτηριστικά του codec π.χ. επιλέγεται προενίσχυση της εισόδου του μετατροπέα Α-Ψ κατά τη χρήση μικροφώνου. Στη συνέχεια μέσω του out[11] = 0x400 εγγράφεται ο καταχωρητής ελέγχου 4 (Control Register 4) από τις τιμές του οποίου καθορίζονται οι παράμετροι για την ADC είσοδο του καναλιού φωνής. Συγκεκριμένα ρυθμίζεται ώστε να μην ενισχύεται καθόλου το σήμα στην ADC είσοδο του συγκεκριμένου καναλιού μέσω του PGA (Programmable Gain Amplifier). Τέλος εγγράφεται, μέσω της τιμής out[15] = 0x0502, ο καταχωρητής ελέγχου 5 (Control Register 5) που καθορίζει τις ρυθμίσεις για την έξοδο του DAC του καναλιού φωνής.

[1] Επισημαίνεται ότι σε περίπτωση τροποποίησης του προγράμματος σας θα πρέπει να επαναλάβετε τη διαδικασία *Build* που περιγράφηκε παραπάνω και να ακολουθήσετε την επιλογή: File > Reload Program .

Μέσω της συγκεκριμένης τιμής που εγγράφεται, αποφασίζεται να μη γίνει καμία ενίσχυση ούτε στην έξοδο του DAC του συγκεκριμένου καναλιού.

```
out[0] = 0xaa;
out[1] = 0;
out[2] = 0x1;
out[3] = 0x0386;
out[4] = 0;

out[5] = 0;
out[6] = 0x1;
out[7] = 0x0306;
out[8] = 0;

out[9]  = 0;
out[10] = 0x1;
out[11] = 0x0400;
out[12] = 0;

out[13] = 0;
out[14] = 0x1;
out[15] = 0x0502;
out[16] = 0;
```

Μετά την ανάθεση τιμών σε κάποιες βοηθητικές μεταβλητές, ακολουθούν οι εξής εντολές:

```
CSR=0x100;
IER=1;
ICR=0xffff;
*(unsigned volatile int *)EMIF_GCR = 0x3300;
*(unsigned volatile int *)EMIF_CE1 = 0xffffff03;
```

Οι καταχωρητές αυτοί χρησιμοποιούνται για την αρχικοποίηση του ψηφιακού επεξεργαστή σήματος και των περιφερειακών του. Με χρήση των δύο πρώτων εντολών γίνεται απενεργοποίηση (disabling) όλων των διακοπών (interrupts) εκτός της διακοπής NMI. Αυτό επιτυγχάνεται μέσω των τιμών που καταχωρούνται στο **C**ontrol **S**tatus **R**egister (CSR) και στον **I**nterrupt **E**nable **R**egister (IER). Ο **I**nterrupt **C**lear **R**egister (ICR) επιτρέπει να εκκαθαριστούν όλα τα maskable interrupts. Χρησιμοποιείται αρνητική λογική και επομένως προκειμένου να καθαριστούν όλα τα maskable interrupts εγγράφεται η τιμή 0xffff. Ο **G**lobal **C**ontrol **R**egister (GCR) της προσαρμοστικής μονάδας EMIF αποτελεί τον γενικό καταχωρητή ελέγχου που ρυθμίζει κατάλληλα όλες εκείνες τις παραμέτρους που είναι κοινές για όλα τα **C**hip **E**nable (CE) της εξωτερικής

μνήμης (πχ ROM, SDRAM, SBSRAM). Τέλος, ρυθμίζεται και το Chip Enable 1 μέσω του καταχωρητή Space Control Register. Ο καταχωρητής αυτός αντιπροσωπεύει τον CE1 χώρο μνήμης (από τους 4 που υποστηρίζει η προσαρμοστική μονάδα EMIF). Εδώ παίρνει την τιμή 0xffffff03 και έτσι καθορίζεται μια ασύγχρονη διεπαφή εύρους 8 – bit, ως ο τύπος μνήμης για το CE1.

Μόλις ληφθεί ένα δείγμα από τη συσκευή *AD535*, δηλαδή μόλις ο DRR της McBSP0 λάβει και τα 16 bit του δείγματος, παράγεται το σήμα RINT0. Με παρόμοια διαδικασία μόλις αδειάσει ο DXR και είναι έτοιμος πλέον να δεχθεί το επόμενο δείγμα, παράγεται το σήμα XINT0. Προκειμένου αυτά τα σήματα να γίνουν αντιληπτά από τη CPU θα πρέπει να συνδεθούν με κάποια από τα διαθέσιμα interrupts της (INT4 – INT15). Αυτό επιτυγχάνεται αρχικοποιώντας κατάλληλα τα πεδία των καταχωρητών IMH και IML:

```
*(unsigned volatile int *)IML = 0x310718A4;
*(unsigned volatile int *)IMH = 0x08202DA3;
```

Για τη συγκεκριμένη εφαρμογή έχουν χρησιμοποιηθεί τα interrupts INT9 και INT11. Στη συνέχεια θα πρέπει τα interrupts αυτά να ενεργοποιηθούν. Αυτό επιτυγχάνεται μέσω των ακόλουθων εντολών :

```
ICR  = 0xffff;
IER  = 0xA02;
CSR = 1;
```

Κάθε φορά που θα γεννιέται το σήμα XINT0 και RINT0, μέσω των INT9 και INT11 καλούνται οι αντίστοιχες ρουτίνες εξυπηρέτησης της CPU. Για το INT9 δημιουργήθηκε η ρουτίνα mcbsp0_x_isr() και για το INT11 δημιουργήθηκε η ρουτίνα mcbsp0_r_isr(). Οι ρουτίνες αυτές δηλώνονται όπως όλες οι συναρτήσεις στη C με τη μόνη διαφορά ότι χρησιμοποιείται η λέξη κλειδί *interrupt* στην αρχή των δηλώσεων τους. Ο μεταφραστής (compiler) αναγνωρίζοντας αυτή τη λέξη, δημιουργεί τον απαραίτητο κώδικα για την αποθήκευση και την ανάκτηση της κατάστασης μηχανής (machine state), κατά την είσοδο και την έξοδο από την ρουτίνα εξυπηρέτησης αντίστοιχα. Η δήλωση για την αντιστοίχηση των ρουτινών εξυπηρέτησης με τα INT9 και INT11 γίνεται στα αρχεία .asm και .cmd τα οποία δίνονται.
Η προτεινόμενη υλοποίηση για τις ρουτίνες εξυπηρέτησης στην περίπτωση μία απλής μεταφοράς ενός δείγματος από την είσοδο στην έξοδο χωρίς καμία επεξεργασία, δίνεται παρακάτω. Πάνω στο format αυτό ζητείται να γίνει και η υλοποίηση της παρούσας άσκησης φιλτραρίσματος:

```
interrupt void mcbsp0_x_isr()
{
        if (start_flag==0)
                out_sample = out[count++];
        else
                out_sample=buffer&0xfffe;
        *(unsigned volatile int *)McBSP0_DXR  = out_sample;
}

interrupt void mcbsp0_r_isr()
{
        in_sample = *(unsigned volatile int *)McBSP0_DRR;
        if (start_flag==1)
                buffer=in_sample;

        /* Εδώ θα πρέπει να τοποθετηθεί ο κώδικας φιλτραρίσματος*/
}
```

Η πρώτη ρουτίνα διαβάζει τη τιμή του καταχωρητή McBSP0_DRR και την αποθηκεύει σε μία μεταβλητή, ενώ η δεύτερη γράφει την αποθηκευμένη πλέον τιμή στον καταχωρητή McBSP0_DXR.
Ο έλεγχος που γίνεται στις ρουτίνες αυτές οφείλεται στην διαδικασία αρχικοποίησης του μετατροπέα Α-Ψ.

Μετά την ολοκλήρωση των παραπάνω εντολών, γίνεται αρχικοποίηση του McBSP0 μέσω της συνάρτησης *msbsp0_init()* που καλείται στη συνάρτηση *main()*. Ακολουθεί ο κώδικας της συγκεκριμένης συνάρτησης ο οποίος αναλύεται στη συνέχεια:

```
void mcbsp0_init()
{
   *(unsigned volatile int *)McBSP0_SPCR = 0;
   *(unsigned volatile int *)McBSP0_PCR  = 0;
   *(unsigned volatile int *)McBSP0_RCR  = 0x10040;
   *(unsigned volatile int *)McBSP0_XCR  = 0x10040;
   *(unsigned volatile int *)McBSP0_DXR  = 0;
   *(unsigned volatile int *)McBSP0_SPCR = 0x12001;
}
```

Με τις δυο πρώτες εντολές εγγράφεται οι τιμή μηδέν τόσο στον **S**erial **P**ort **C**ontrol **R**egister (SPCR) όσο και στον **P**in **C**ontrol **R**egister (PCR). Οι δύο αυτοί καταχωρητές περιέχουν τα bits ελέγχου κατάστασης του McBSP0 στη συγκεκριμένη περίπτωση, και ουσιαστικά οι δυο τους ρυθμίζουν τη σειριακή θύρα. Επιπλέον, ο PCR χρησιμοποιείται για να ορίσει τις ακίδες (pins) της σειριακής θύρας σαν γενικού σκοπού εισόδους ή εξόδους κατά τη διάρκεια της επανεκκίνησης του πομπού (transmitter) ή / και του δέκτη (receiver).

Οι επόμενοι δυο καταχωρητές είναι οι **R**eceive **C**ontrol **R**egister (RCR) και **T**ransmit **C**ontrol **R**egister (TCR). Ο πρώτος καταχωρητής ρυθμίζει όλες εκείνες τις παραμέτρους που καθορίζουν τη λειτουργία της λήψης δεδομένων, ενώ ο δεύτερος καθορίζει τις αντίστοιχες για τη λειτουργία εκπομπής δεδομένων. Στους δύο αυτούς καταχωρητές εγγράφεται η ίδια τιμή και αυτό γίνεται γιατί επιθυμούμε οι λειτουργίες αποστολής και λήψης δεδομένων, μέσω του McBSP0, να είναι πανομοιότυπες και να ακολουθούν την ίδια λογική. Μια από τις βασικές παραμέτρους που προσδιορίζεται μέσω των τιμών που ανατίθενται σε αυτούς είναι πως η λήψη και η αποστολή γίνεται σε frames των 16bits το καθένα.

Στη συνέχεια, ο καταχωρητής **D**ata **T**ransmit **R**egister (DXR) αρχικοποιείται με τη τιμή 0. Στο καταχωρητή DXR εγγράφονται δεδομένα από τη CPU, προκειμένου αυτά να καταλήξουν στην έξοδο του McBSP0 από όπου μπορεί να τα παραλάβει ο DAC. Στον παραπάνω κώδικα γράφεται η τιμή 0 στον DXR προκειμένου να μην αποσταλεί τιμή στον Ψ-Α.

Στην τελευταία εντολή μεταβάλλεται το περιεχόμενο του SPCR λαμβάνοντας την τιμή 0x12001. Με την τιμή αυτή γίνονται οι κατάλληλες ρυθμίσεις στην σειριακή θύρα έτσι ώστε να είναι πλέον έτοιμη τόσο για λήψη όσο και για αποστολή δεδομένων.

Μετά την ολοκλήρωση των παραπάνω αρχικοποιήσεων το πρόγραμμα μπαίνει σε έναν ατέρμονα βρόχο (while( )), κατά τη διάρκεια του οποίου εκτελούνται συνέχεια οι δυο ρουτίνες εξυπηρέτησης διακοπών.

Ερώτημα 1: Υποθέστε ότι επιθυμούμε να επεξεργαστούμε το σήμα εισόδου και ότι η επεξεργασία αυτή θα υλοποιηθεί εντός της *ISR* συνάρτησης *mcbsp0_r_isr* ( ), η οποία εξυπηρετεί το σήμα διακοπής *RINT0* . Δεδομένου, ότι έχει ενεργοποιηθεί και η διακοπή *XINT0*, σε ποια περίπτωση θα υπάρχει σύγκρουση μεταξύ αυτών των δύο και γιατί ;

4. FIR Φίλτρα

4.1 Στοιχεία Θεωρίας

Η έξοδος ενός FIR φίλτρου μήκους N, δίνεται από την ακόλουθη γραμμική συνέλιξη :

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] = h[0]x[n] + ... + h[N-1]x[n-N+1] \qquad (1)$$

Η παραπάνω σχέση μπορεί να γραφεί ισοδύναμα στην ακόλουθη μορφή εσωτερικού γινομένου

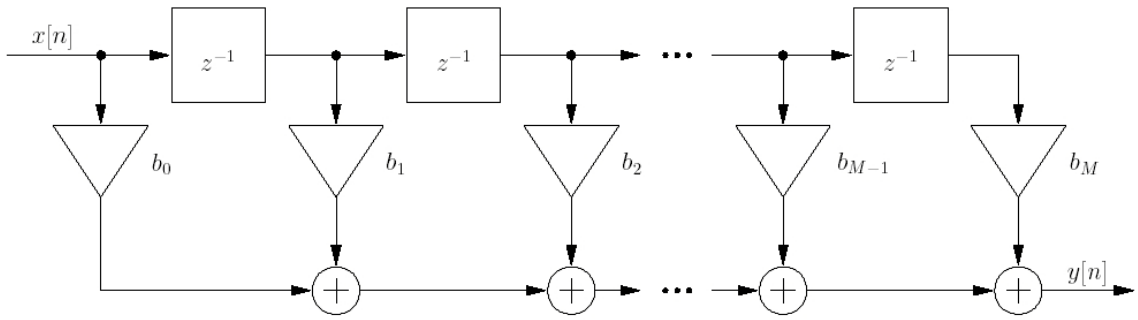
$$y[n] = h^T x_n \qquad (2)$$

όπου τα διανύσματα $\boldsymbol{h}$ και $\boldsymbol{x}_n$ ορίζονται ως ακολούθως:

$$h = \left[h[0]\ h[1] \ldots\ h[N-1]\right]^t$$

$$x_n = \left[x[n]\ x[n-1] \ldots\ x[n-N+1]\right]^t \qquad (3)$$

Παρατηρήστε ότι το διάνυσμα x_n , τη χρονική στιγμή n περιέχει τα N πιο πρόσφατα δείγματα του σήματος εισόδου και επομένως με κάθε νέο δείγμα του σήματος θα πρέπει να ενημερώνεται. Στο Σχήμα 2 φαίνεται η απευθείας υλοποίηση τύπου I του παραπάνω φίλτρου.



Σχήμα 2: Δομή Απευθείας Υλοποίηση Τύπου I ενός FIR Φίλτρου.

4.2 Σχεδιασμός FIR Φίλτρων στο περιβάλλον Matlab

Ένα φίλτρο χαρακτηρίζεται μοναδικά από την κρουστική του απόκριση ή ισοδύναμα από την απόκριση συχνότητάς του. Για τη σχεδίαση των FIR φίλτρων που επιθυμείτε

μπορείτε να χρησιμοποιήσετε μία εκ των συναρτήσεων fir1() ή remez() του υπολογιστικού περιβάλλοντος Matlab. Οι συντελεστές που σας επιστρέφουν στα ορίσματα εξόδου τους οι παραπάνω συναρτήσεις είναι οι συντελεστές της κρουστικής απόκρισης του επιθυμητού φίλτρου. Οι συντελεστές αυτοί θα χρησιμοποιηθούν παρακάτω στο πρόγραμμά σας για την υλοποίηση των φίλτρων με τη βοήθεια του *CCS*.

4.3 Υλοποίηση FIR Φίλτρων στον Ψηφιακό Επεξεργαστή Σημάτων

Από τη Σχέση (1) είναι προφανές ότι ο υπολογισμός της εξόδου του φίλτρου, απαιτεί *N* πολλαπλασιασμούς και (*N-1*) προσθέσεις. Ισοδύναμα, η υπολογιστική πολυπλοκότητα αυτού του FIR φίλτρου είναι *N multiply-accumulates* (*MACs*). Ο TMS320C6711 επεξεργαστής που θα χρησιμοποιηθεί έχει απόδοση 300.000 *MACs*/*sec*.

Ερώτημα 2: Υπολογίστε το πλήθος των *MACs*/*sec* που απαιτούνται για τον υπολογισμό της γραμμικής συνέλιξης ενός φίλτρου μήκους 20 και ενός φίλτρου μήκους 100. Θεωρώντας ότι η συχνότητα δειγματοληψίας που χρησιμοποιεί ο Α-Ψ της συσκευής *AD535* είναι τα 8KHz, υπολογίστε το μέγιστο μήκος φίλτρου που μπορείτε να χρησιμοποιήσετε ώστε η υλοποίησή του στον TMS320C6711 να λειτουργεί σε πραγματικό χρόνο.

Από τις Σχέσεις (1) και (3) είναι προφανές ότι το παλαιότερο δείγμα εισόδου $x[n-(N-1)]$ πολλαπλασιάζεται με τον συντελεστή $h[N-1]$ της κρουστικής απόκρισης του φίλτρου και το πιο πρόσφατο δείγμα $x[n]$ πολλαπλασιάζεται με τον συντελεστή $h[0]$ της κρουστικής απόκρισης. Από πλευράς υλοποίησης, τα αναγκαία *N* δείγματα του σήματος εισόδου θα μπορούσαν να αποθηκευθούν σε ένα πίνακα-διάνυσμα στη μνήμη και κάθε φόρα που θα θέλαμε να εισάγουμε ένα νέο δείγμα του σήματος εισόδου στον πίνακα θα μετατοπίζαμε τα στοιχεία του κατά μία θέση. Ο παραπάνω τρόπος υλοποίησης της γραμμικής συνέλιξης δεν είναι αποδοτικός. Για το λόγο αυτό στην υλοποίηση μας θα χρησιμοποιηθεί ένα σχήμα κυκλικής δεικτοδότησης των στοιχείων ενός διανύσματος. Επισημαίνεται, αν και δεν αποτελεί σκοπό αυτής της άσκησης, ότι οι '*C6000 DSPs* διαθέτουν κατάλληλο υλικό που δύναται να προσπελαστεί από εντολές assembly, ώστε να υλοποιηθεί αυτό το κυκλικό σχήμα δεικτοδότησης. Η ιδέα της κυκλικής δεικτοδότησης, φαίνεται στο Σχήμα 3. Οι συντελεστές του φίλτρου και τα δείγματα του σήματος εισόδου αποθηκεύονται στα μήκους *N* διανύσματα *h*[] και *xcirc*[] αντίστοιχα. Ας υποθέσουμε επίσης ότι ο δείκτης *newest* δείχνει σε εκείνη τη θέση του *κυκλικού*

πίνακα, στην οποία βρίσκεται το πιο πρόσφατο δείγμα εισόδου και *oldest* τη θέση του *κυκλικού* πίνακα, στην οποία βρίσκεται το πιο παλαιό δείγμα εισόδου. Όταν, τη χρονική στιγμή *n*+1 λαμβάνεται ένα νέο δείγμα εισόδου, θα πρέπει να καταχωρείται στη θέση που δείχνει ο δείκτης *oldest* και οι δείκτες να ενημερώνονται ως ακολούθως: *newest*=*oldest,* και *oldest*=(*oldest* –1) *modN*. Παρατηρήστε ότι, όταν ο δείκτης *oldest,* αρχικά, έχει την τιμή *0*, μετά την ενημέρωσή του γίνεται *N*-1. Η έξοδος του φίλτρου υπολογίζεται, από τη σχέση :

$$y[n] = \sum_{m=0}^{N-1} h[m]\,xcirc[(newest + m) \bmod N] \qquad (4)$$

όπου, $(newest + m) \bmod N$ ακέραιος στο διάστημα {*0, ..., N*-1}.

Παρατηρήστε ότι με το παραπάνω σχήμα δεικτοδότησης ο υπολογισμός της εξόδου του φίλτρου απαιτεί *N MACs* και μια καταχώρηση της τιμής του νέου δείγματος του σήματος εισόδου.

Θέση	Διάνυσμα *h*[ ]	Διάνυσμα *xcirc*[ ]
newest	h[0]	x[n]
1	h[1]	x[n-1]
2	…	…
…	…	…
…	…	…
…	…	…
N-2	h[N-2]	x[n- 2]
Oldest	h[N-1]	x[n -1]

Σχήμα 3: Τα περιεχόμενα των Διανυσμάτων των Συντελεστών της κρουστικής απόκρισης και των δεδομένων εισόδου.

4.4. Υλοποίηση FIR Φίλτρων σε DSP-Τροποποίηση του προγράμματος loopback

Στο ήδη υπάρχον πρόγραμμα *loopback.c*, θα πρέπει να προσθέσετε τον κατάλληλο κώδικα που θα εξασφαλίζει την (σωστή) υλοποίηση ενός FIR φίλτρου.[2]

4.5 Οι Συντελεστές του Φίλτρου

Στο ζητούμενο αρχείο .c τοποθετούμε στο τμήμα των καθολικών δηλώσεων, τους

[2] Σε περίπτωση που θέλετε να υλοποιήσετε το ζητούμενο πρόγραμμα σε διαφορετικό αρχείο, προσθέστε το νέο αρχείο .c στο *Project*, επιλέγοντας : Project > Add Files to Project .

συντελεστές του προς υλοποίηση FIR φίλτρου. (Άλλος τρόπος είναι η δημιουργία ενός *header* αρχείου που θα γίνεται *include* και θα περιέχει τον αντίστοιχο κώδικα των δηλώσεων). Αυτό επιτυγχάνεται, για παράδειγμα, ως ακολούθως :

```
#define FilterLength 3
float x[FilterLength];
float h[FilterLength] = {0.5 , 0.5 , 0.5 }
```

4.3 Συνάρτηση Υλοποίησης FIR Φίλτρου

Ερώτημα 4: Υλοποιείστε το FIR φίλτρο εντός της *ISR* συνάρτησης *mcbsp0_r_isr ()*. Επισημαίνεται ότι ο πίνακας με τα δείγματα εισόδου θα πρέπει να αρχικοποιηθεί με μηδενικά στην αρχή της συνάρτησης *main*( ).

Ερώτημα 5: Συνδέστε στην είσοδο της κάρτας ένα *ακουστικό* σήμα και στην έξοδο τα ηχεία. Χρησιμοποιώντας τα φίλτρα που θα σας υποδειχθούν στο εργαστήριο, συγκρίνατε τον ήχο του αρχικού και του τελικού σήματος κι εξηγήστε την επίδραση των δύο FIR φίλτρων.

Ερώτημα 6: Συνδέστε ένα ημιτονικό σήμα στο *jack* εισόδου με *πλάτος* μικρότερο από *±1 V*, χωρίς *DC offset*. Οδηγήστε στον παλμογράφο την αρχική και την τελική κυματομορφή και μετρήστε το gain του φίλτρου για ένα επαρκές σύνολο συχνοτήτων. Επίσης, επιβεβαιώστε τις αποκρίσεις συχνότητας των φίλτρων από τις παραπάνω διαθέσιμες κυματομορφές.

5. Βιβλιογραφία

[1] TMS320C6711 Digital Signal Processor Data Sheet (Literature Number SPRS088),Texas Instruments, Dallas, TX, 2002.

[2] TMS320C6000 Peripherals Guide (Literature Number SPRU190), Texas Instruments, Dallas, TX, 2001.

[3] TLC320AD535C/I Data Manual Dual Channel Voice/Data Codec, (Literature Number SLAS202), Texas Instruments, Dallas, TX, 1999.

[4] Code Composer Studio Getting Started Guide (Literature Number SPRU509), Texas Instruments, Dallas, TX, 2001.

[5] Γ. Β. Μουστακίδης, "Βασικές Τεχνικές Ψηφιακής Επεξεργασίας Σημάτων,", Εκδόσεις Τζιόλα 2004.

[6] Sanjit K. Mitra, "Digital Signal Processing, A Computer-Based Approach," McGraw-Hill, 1998.

[7] Vinay K. Ingle and John G. Proakis, "Digital Signal Processing Using MATLAB®," PWS Publishing Company, 1997.

[8] Steven A. Tretter, Communication System Design Using DSP Algorithms. With Laboratory Experiments for the TMS320C6701 and TMS320C6711, Kluwer Academic / Plenum Publishers, New York 2003.