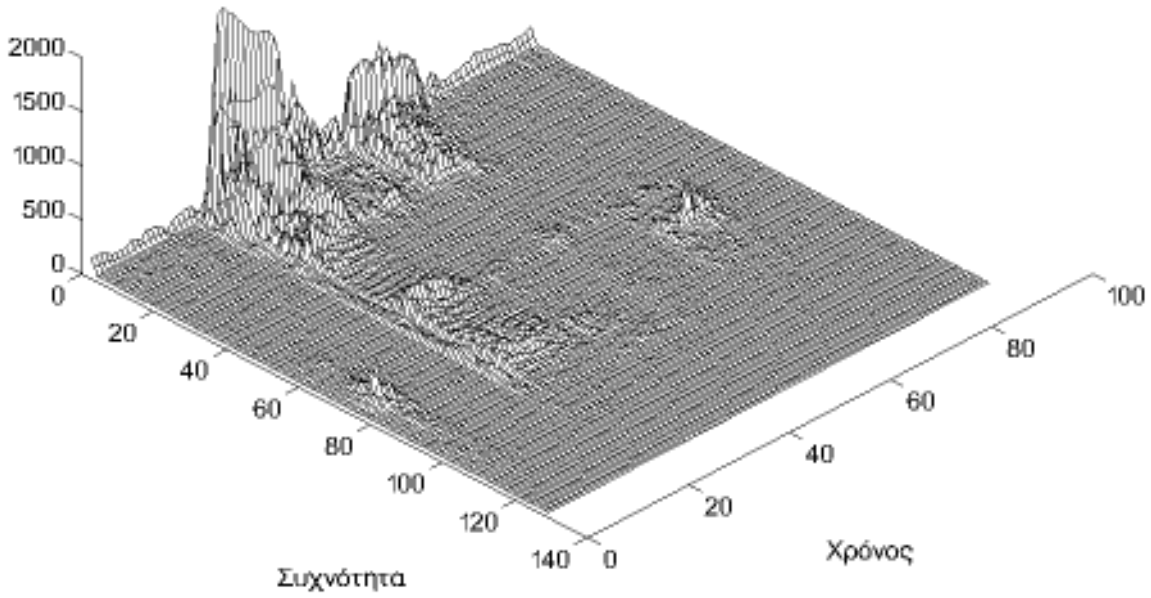
ΤΕΙ Πειραιά
Τμήμα Αυτοματισμού

ΑΝΑΓΝΩΡΙΣΗ ΦΩΝΗΣ ΜΕ ΤΟ DSP56000



Πτυχιακή Εργασία

Αργύρης Διαμαντής - Ηλίας Αλεξόπουλος

1996

Περιεχόμενα

Κεφάλαιο 1

Κεφάλαιο 2

Κεφάλαιο 3

Κεφάλαιο 4

Κεφάλαιο 5

Παράρτημα

Εισαγωγή

Η πτυχιακή αυτή εργασία ανατέθηκε στους σπουδαστές Αλεξόπουλο Ηλία, και Διαμαντή Αργύρη από τον καθηγητή του τμήματος Αυτοματισμού στο Τ.Ε.Ι. Πειραιά κ.Γ.Σύρκο το 1992 με τίτλο "Αναγνώριση Ομιλίας με το DSP56001". Σκοπός της εργασίας αυτής ήταν η κατασκευή μιας εφαρμογής που θα αναγνώριζε υπαγορευμένες λέξεις και θα εντολοδοτούσε ένα προσωπικό υπολογιστή, παράλληλα με το πληκρολόγιο.

Αυτό βέβαια θα ήταν το ιδανικό αποτέλεσμα. Λόγω της δυσκολίας και της πολυπλοκότητας του εγχειρήματος, υπήρχαν μεγάλες πιθανότητες αποτυχίας, οπότε σ' αυτή τη περίπτωση η εργασία θα ήταν ένα κείμενο αναφοράς και μελέτης για όσους επρόκειτο να ασχοληθούν με συναφή θέματα στο μέλλον.

Εκείνη την εποχή ο τομέας αυτός ήταν ακόμη σε ερευνητικό στάδιο και δεν υπήρχαν πολλές πληροφορίες διαθέσιμες για να μπορέσουμε να κινηθούμε. Έτσι καταναλώθηκε αρκετός χρόνος για την εύρεση αυτών των πληροφοριών, την κατανόηση τους και την αξιοποίηση τους. Παράλληλα επειδή υπήρχε φόρτος εργασίας λόγω των μαθημάτων της σχολής, και αργότερα τις εργασίας, η ταχύτητα ανάπτυξης δεν ήταν η μέγιστη δυνατή. Ο καθαρός χρόνος που υπολογίζουμε ότι εργαστήκαμε στο θέμα αυτό, είναι περίπου δύο χρόνια. Έτσι εξηγείται και ο μεγάλος χρόνος παράδοσης (τέσσερα χρόνια).

Η εργασία αυτή ξεκίνησε παράλληλα με την πτυχιακή εργασία του κ. Γαλανάκη "Αναγνώριση ομιλίας με PC". Έτσι επειδή η φιλοσοφία και στις δύο εργασίες είναι η ίδια, υπήρξε μια συνεργασία και κοινή προσπάθεια έρευνας. Αυτός είναι και ο λόγος της αναφοράς των προγραμμάτων σε γλώσσα C, αφού αυτά αναπτύχθηκαν από κοινού.

Στο γραπτό αυτό κείμενο έχουμε προσπαθήσει να μην κουράζουμε τον αναγνώστη με κουραστικές θεωρίες και περίπλοκα μαθηματικά, αφού αυτά υπάρχουν στη σχετική βιβλιογραφία άφθονα από πολλούς αρμοδιότερους από εμάς. Αντίθετα εμείς προσπαθήσαμε να παρουσίασουμε όσο μπορούσαμε την πρακτική εφαρμογή των θεωριών αυτών, ώστε αυτές να μπορούν να κατανοηθούν όσο το δυνατόν πιο εύκολα.

Η χρονική εξέλιξη της εργασίας αποδίδει τις σκέψεις μας και τα δεδομένα στην κάθε χρονική στιγμή, όπως αυτά έγιναν και καθρεφτίζει το λογικό τρόπο σκέψης μας, για την επίλυση των επιμέρους δυσκολιών. Στην αναλυτική περιγραφή, μπορεί κάποιος να διαπιστώσει τη μεθοδολογία επίλυσης μερικών χαοτικών προβλημάτων μέσα από την εμπειρία της εκσφαλμάτωσης, που αποκτήσαμε εργαζόμενοι στη βιομηχανία.

Περιγραφή Κεφαλαίων

Το πρώτο κεφάλαιο αναλύει τις βασικές αρχές της ομιλίας και της ακουστικής του ανθρώπου, που αποτελούν το βασικό πυρήνα γνώσεων, βοηθώντας στην κατανόηση των μηχανισμών της αναγνώρισης ομιλίας. Κατόπιν αναφέρονται τα διάφορα εργαλεία που έχει στη διάθεσή του ο μηχανικός, για να μπορέσει να εξάγει τα διάφορα χαρακτηριστικά της φωνής και να τα χρησιμοποιήσει ανάλογα με τις απαιτήσεις.

Το δεύτερο κεφάλαιο αναφέρεται στις υπάρχουσες μεθόδους αναγνώρισης της ομιλίας, και αναλύει τα πλεονεκτήματα και τα μειονεκτήματα της κάθε μίας. Περιγράφεται λεπτομερώς ο δυναμικός προγραμματισμός (DTW), και γίνεται μια αναφορά στα κρυφά μοντέλα Markov.

Το τρίτο κεφάλαιο περιγράφει συνοπτικά το αναπτυξιακό εργαλείο της Motorola για τον ψηφιακό επεξεργαστή 56001 και δίνεται μια εικόνα των δυνατοτήτων του επεξεργαστή.

Το τέταρτο κεφάλαιο είναι όλο το ιστορικό της ανάπτυξης. Εκεί στο πρώτο μέρος γίνεται μια συνοπτική αναφορά της πορείας μας από την αρχή ως το τέλος. Στο δεύτερο μέρος παρουσιάζεται αναλυτικά η εργασία μας, με όλα τα στάδια που πέρασαν από τη σχεδίαση ως το τελικό αποτέλεσμα της κάθε προσπάθειας, είτε στο υλικό, είτε στο λογισμικό. Επίσης εκεί υπάρχουν κάποια ενδιάμεσα αποτελέσματα σχετικά με την ομιλία, όπως πχ. το φασματόγραμμα στη σελίδα 4-7.

Το πέμπτο κεφάλαιο περιέχει τα αποτέλεσματα του αλγορίθμου αναγνώρισης ομιλίας, ποσοστά επιτυχίας, φασματογράμματα καθώς και τις επιδόσεις του σε ταχύτητα και μνήμη.

Τέλος, το παράρτημα περιέχει την τεκμηρίωση του έργου. Εκεί περιλαμβάνονται θεωρητικά σχέδια του υλικού, προγραμματιζόμενη λογική, και λογισμικό.

Ευχαριστίες

Τελειώνοντας αυτή τη μικρή εισαγωγή θα πρέπει να αναφέρουμε ότι αυτή η εργασία ήταν το αποτέλεσμα της προσωπικής προσπάθειας, της δίψας για γνώση, μάθηση, και χρήσης υψηλής τεχνολογίας, παράλληλα με το μεράκι της δημιουργίας.

Δεν πρέπει να παραγνωρίσουμε βέβαια το γεγονός ότι υπήρξαν πολλοί άνθρωποι που στήριξαν την προσπάθεια μας και θα θέλαμε να τους ευχαριστήσουμε.

Ειδικότερα θέλουμε να ευχαριστήσουμε τον καθηγητή μας κ. Γ. Σύρκο για την ευκαιρία που μας έδωσε να ασχοληθούμε με ένα τόσο ενδιαφέρον θέμα, παράλληλα με την άδεια χρήσης του αναπτυξιακού του DSP56001 που οικειοποιηθήκαμε κατά το μεγαλύτερο διάστημα της ανάπτυξης της εργασίας μας, και την εμπιστοσύνη που έδειξε απέναντί μας.

Επίσης θα θέλαμε να ευχαριστήσουμε τον καθηγητή μας κ. Γ.Αγγελόπουλο που μας επέτρεψε να γνωρίσουμε από κοντά τα θαύματα της προγραμματιζόμενης λογικής.

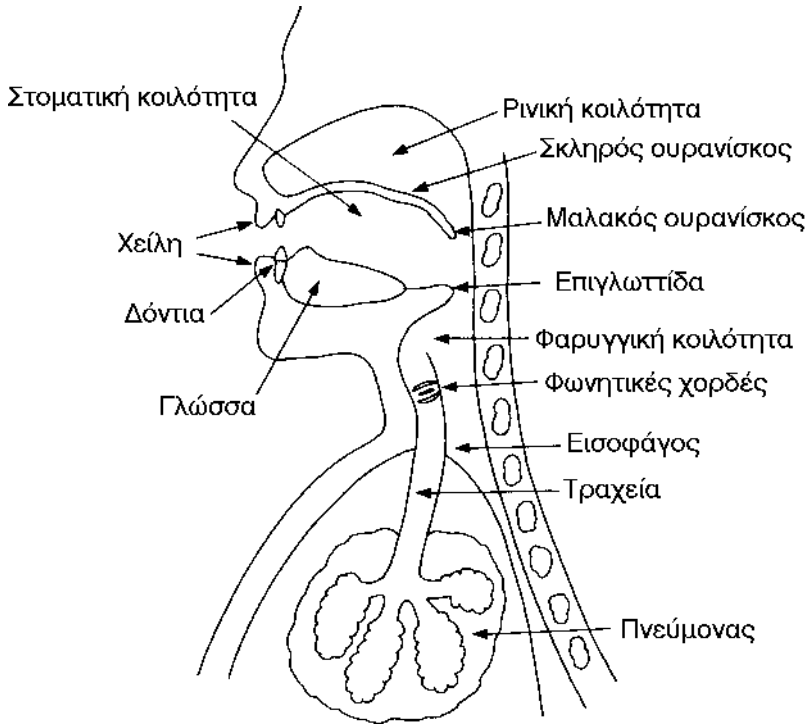
Τέλος ευχαριστούμε θερμά την INTRAKOM Α.Ε. και τους εξαίρετους συναδέλφους μας που μας βοήθησαν ποικιλοτρόπως στην υλοποίηση του υλικού μέρους.

Χαρακτηριστικά φωνής και μέθοδοι διακρίσεως τους

Για να μπορέσει να αναπτυχθεί ένας αλγόριθμος αναγνώρισης φωνής είναι σημαντικό να μελετηθεί πρώτα ο τρόπος παραγωγής της φωνής, καθώς επίσης και το ακουστικό σύστημα του ανθρώπου, μιας και πρέπει να γνωρίζουμε καλά την μορφή και τα χαρακτηριστικά του σήματος που θα πρέπει να επεξεργαστούμε. Τα διάφορα στοιχεία της ανθρώπινης φυσιολογίας που θα δούμε εδώ, βοηθούν στο να κατανοηθεί ο τρόπος που ο εγκέφαλος καταλαβαίνει και ερμηνεύει τα ακουστικά ερεθίσματα, ούτως ώστε να μπορέσουμε να εξομοιώσουμε τις λειτουργίες αυτές με τεχνητά μέσα και να φτάσουμε σε όσο το δυνατον καλύτερο αποτέλεσμα.

Η ομιλία

Οι ήχοι της ομιλίας δημιουργούνται από τον αέρα που εκπνέουν οι πνεύμονες, παράγοντας είτε ταλαντώσεις των φωνητικών χορδών για τα φωνήεντα είτε στροβιλισμούς του αέρα σε κάποιο σημείο του φωνητικής περιοχής (πχ. δόντια) για τα σύμφωνα. Οι ήχοι που παράγονται επηρεάζονται από τα σχήματα που παίρνει η φωνητική κοιλότητα με αποτέλεσμα να δημιουργούνται διάφορες αρμονικές του αρχικού σήματος. Τα διάφορα όργανα που συμμετέχουν στην παραγωγή των ήχων της φωνής φαίνονται στο σχήμα 1.1.

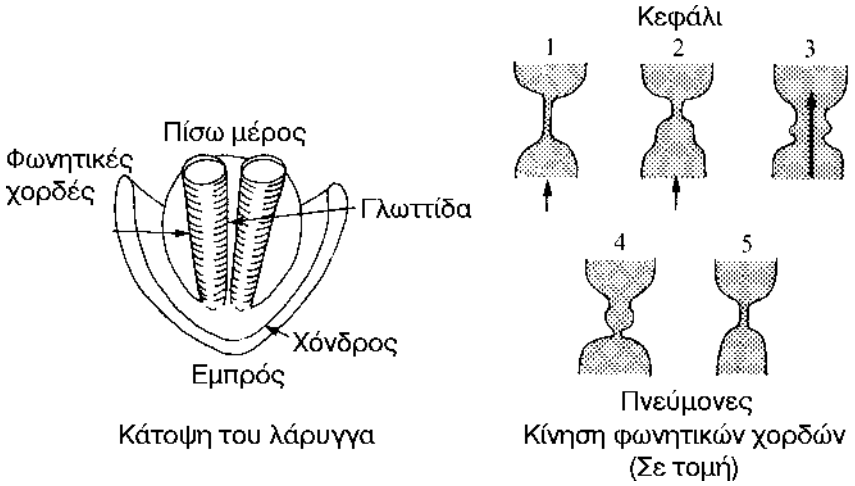


Σχ. 1.1. *Τα όργανα ομιλίας του ανθρώπου.*

Τα ανθρώπινα όργανα ομιλίας

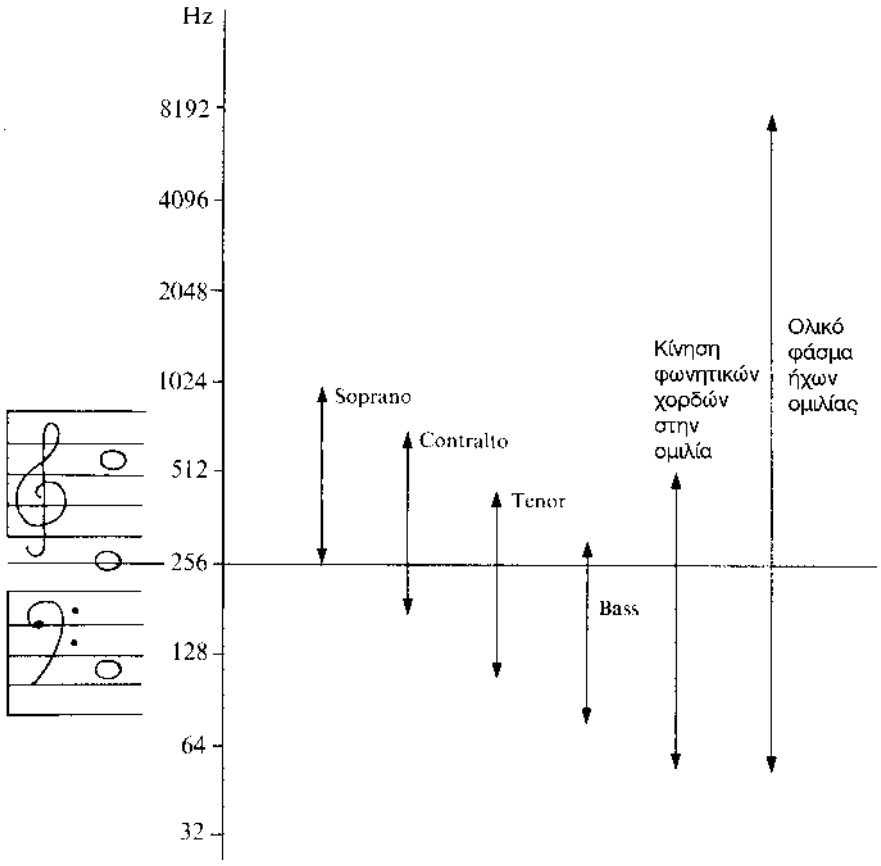
Τα κυριότερα σημεία του ανθρώπινου συστήματος ομιλίας είναι οι πνεύμονες, ο λάρυγγας, οι φωνητικές χορδές, η γλώσσα και τα δόντια. Δύο είναι οι βασικές μέθοδοι παραγωγής ήχων. Στην πρώτη οι φωνητικές χορδές που βρίσκονται στο λάρυγγα ταλαντώνονται σε σταθερές συχνότητες από τον αέρα που εκπνέουν οι πνεύμονες, οπότε παράγονται τα ηχηρά φωνήεντα. Η δεύτερη μέθοδος παράγει ήχους από τους στροβιλισμούς του αέρα σε κάποιες από τις φωνητικές περιοχές όπως τα δόντια ή τα χείλη, και παράγει τα σύμφωνα.

Οι φωνητικές χορδές είναι ελαστικές, οπότε ανοίγουν μετά από κάθε παλμό της γλωττίδας λόγω της πίεσης του αέρα, και μαζεύουν όταν η πίεση χαμηλώσει (λόγω του φαινομένου bernoulli). Αν δούμε τις χορδές σε κάθετη τομή όπως στο σχήμα σχ 1.2 θα δούμε ότι οι φωνητικές χορδές δεν ανοιγοκλείνουν ομοιόμορφα, αλλά κυματίζουν από κάτω προς τα πάνω.



Σχ. 1.2. *Οι κινήσεις των φωνητικών χορδών κατά την ομιλία*

Η συχνότητα ταλάντωσης εξαρτάται από την τάση που ασκούν οι μύες, τη μάζα και το μήκος των φωνητικών χορδών. Το μήκος των χορδών στους άνδρες είναι από 17 ως 24 mm, ενώ στις γυναίκες από 13 ως 17 mm. Η μέση βασική συχνότητα των παλμών της γλωττίδας είναι για τους άνδρες περίπου 125Hz, για τις γυναίκες 200Hz και για τα παιδιά 300Hz. Το σχήμα 1.3 δείχνει τις βασικές συχνότητες που παράγονται από διάφορες φωνές τραγουδιστών σε σχέση με τις νότες και την ομιλία.



Σχ. 1.3. *Οι συχνότητες που παράγονται κατά την ομιλία.*

Οι φωνητικές χορδές παράγουν και αρμονικές πολλαπλάσιες της βασικής αρμονικής. Το πλάτος των αρμονικών μειώνεται όσο αυξάνονται οι συχνότητες.

Κάθε αλλαγή στους μύες του προσώπου του ομιλητή, επηρεάζει την βασική αρμονική που παράγεται από τις φωνητικές χορδές. Η φωνητική κοιλότητα έχει έντονη κινητικότητα και αλλάζει γεωμετρία ανάλογα με την θέση της γλώσσας, του φάρυγγα, των χειλιών και των μαγούλων. Η αναπνευστική οδός είναι πιο σταθερή αλλά μπορεί να συνδέεται ακουστικά με την φωνητική περιοχή ανάλογα με την θέση που τοποθετείται ο ουρανίσκος.

Φθόγγοι

Το μικρότερο στοιχείο της φωνής είναι οι φθόγγοι. Οι φθόγγοι συμβολίζονται διεθνώς (κατα IPA, International Phonetic Alphabet) με μερικά γράμματα ή σύμβολα που γράφονται ανάμεσα σε δύο κάθετες γραμμές όπως /p/ για τη λέξη ‘pan’. Στη πραγματικότητα η προφορά του φθόγγου /p/ δεν είναι η ίδια πάντα αλλά αλλάζει ανάλογα με τη θέση του μέσα στη λέξη όπως στις αγγλικές λέξεις ‘pan’ και ‘span’. Στην αγγλική γλώσσα υπάρχουν 40 φθόγγοι που αρκούν για να περιγράψουν όλες τους φωνητικούς ήχους της ομιλίας. Ο πίνακας των διαφόρων φθόγγων φαίνεται στον πίνακα 1.1.

Our symbol	IPA symbol	Key word	Our symbol	IPA symbol	Key word
Vowels			*Fricatives*		
/ee/	i	each	/f/	f	free
/i/	I	it	/th/	θ	thin
/e/	ɛ	end	/s/	s	see
/ar/	ɑ	hard	/sh/	ʃ	shall
/u/	ʊ	good	/v/	v	vine
/uu/	u	ooze	/dh/	ð	then
/er/	ɜ	bird (neutral)	/z/	z	zoo
/aa/	æ	had	/xh/	ʒ	azure
/a/	ʌ	bud	/h/	h	he
/aw/	ɔ	hoard	*Affricates*		
/@/	ə	allow (schwa)	/tsh/	tʃ	chair
/Q/	ɒ	hot	/dzh/	dʒ	jar
Diphthongs			*Semi vowels*		
/ei/	ei	aid	/w/	w	we
/u@/	ʊə	pure	/y/	j	you
/au/	au	cow	/r/	r	red
/ou/	əu	own	/l/	l	live
Plosives			*Nasals*		
/p/	p	pie	/m/	m	me
/t/	t	ten	/n/	n	no
/k/	k	key	/ng/	ŋ	sing
/b/	b	be			
/d/	d	den			
/g/	g	go			

Πίνακας 1.1. *Η γραφή των φθόγγων της Αγγλικής γλώσσας με το διεθνές φωνητικό αλφάβητο*

Τα φωνήεντα /a/, /e/, /i/ παράγονται από τις ταλαντώσεις των φωνητικών χορδών που βρισκονται στην κορυφή της τραχείας. Η βασική τους συχνότητα εξαρτάται και από την ροή του αέρα, αλλά κυρίως από την δύναμη που εξασκείται στις χορδές. Το /@/ δεν είναι κάποιο συγκεκριμένο φωνήεν αλλά ένας ουδέτερος ήχος που προέρχεται από κάποιο άτονο φωνήεν μέσα σε μια λέξη όπως η λέξη 'but' που ακούγεται /b @ t/ και όχι /b a t/.

Τα σύμφωνα

Τα σύμφωνα παράγονται λόγω τριβής που προκαλείται από τους στροβιλισμούς του αέρα. Η φύση του ήχου εξαρτάται από την περιοχή παραγωγής του και τη θέση της γλώσσας και των υπολοίπων μερών του φωνητικού συστήματος. Τέτοια σύμφωνα είναι τα /f/, /s/, /sh/.

Ένα άλλο είδος συμφώνου είναι το άηχο σύμφωνο, το οποίο δημιουργείται από την απότομη διακοπή του ρεύματος του αέρα από κάποιο διάφραγμα, όπως τα χείλη ή τα δόντια. Τότε ο αέρας συσσωρεύεται, η πίεση αυξάνει και όταν το διάφραγμα ανοίξει, έχουμε απότομη εκροή αέρα (όπως στα /p/, /k/).

Κατά τη διάρκεια του συνεχούς λόγου έχουμε και διαστήματα ησυχίας, αλλά όχι στην αρχή και στο τέλος μιας λέξης, όπως ίσως θα περιμέναμε, αλλά πριν τα άηχα σύμφωνα. Η διάρκεια αυτών των διαστημάτων είναι της τάξης των 30-50ms.

Τα ρινικά σύμφωνα παράγονται από την ρινική κοιλότητα ενώ η στοματική λειτουργεί ως συντονισμένο αντηχείο. Η ένταση του ήχου που βγαίνει είναι χαμηλότερη από τους άλλους λόγω του ότι παράγεται εύρυτερο φάσμα συχνοτήτων, ενω παράλληλα η στοματική κοιλότητα απορροφά τον ήχο λόγω των μαλακών τοιχωμάτων.

Τα σύμφωνα χωρίζονται σε άφωνα τα οποία παράγουν μια τυχαία θορυβώδη ταλάντωση (λευκός θόρυβος) όπως /f/, /th/, /s/, /sh/ ή συνδιάζονται με παραγωγή φθόγγων όπως /v/, /z/, /dh/, /xh/.

Τα φωνήεντα

Η ταλάντωση των φωνητικών χορδών, με την παραγωγή των φωνηέντων παράγει κάποιες βασικές συχνότητες και τις αρμονικές τους οι οποίες είναι πολλαπλάσια της βασικής. Αν η βασική αρμονική είναι 100Hz τότε θα εχουμε αρμονικές 200, 300, 400, 500 Hz κλπ. Όμως, επειδή η κατασκευή του φωνητικού συστήματος έχει κάποιες ιδιοσυχνότητες, οι αρμονικές δεν ενισχύονται όλες το ίδιο. Συνήθως το φωνητικό σύστημα έχει δύο βασικές συχνότητες συντονισμού, f_1 και f_2.

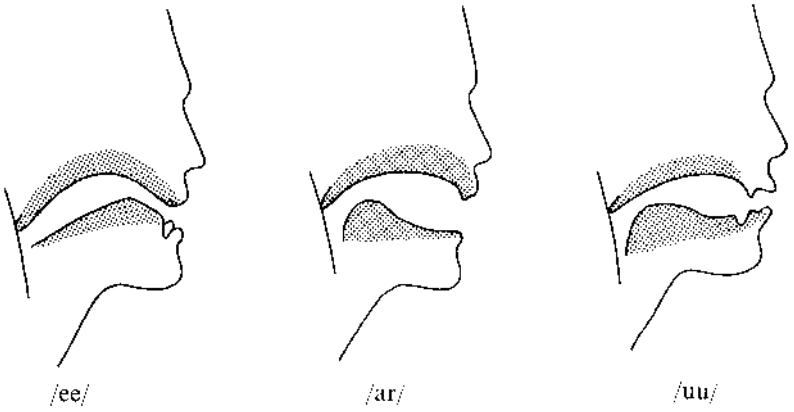
Η συχνότητες του κάθε φωνήεντος είναι αρκετά διαφορετικές από φωνήεν σε φωνήεν. Όμως για όλους τους ανθρώπους, οι συχνότητες συντονισμού για το ίδιο φωνήεν είναι περίπου οι ίδιες. Στο φωνήεν /ee/ όπως στη λέξη 'he' η f_1 είναι 300Hz και η f_2 είναι 2100Hz. Η βασική συχνότητα μπορεί να διαφέρει από άνθρωπο σε άνθρωπο, από τη διάθεσή του και την προσωδία. Τελικά αυτό που κάνει τους ήχους να ξεχωρίζουν είναι το πλάτος και οι σχέσεις των συχνότητων συντονισμού.

Στον παρακάτω πίνακα 1.2 φαίνεται η αντιστοιχία φωνηέντων με τις συχνότητες συντονισμού του φωνητικού συστήματος.

Φωνήεν		f_1	f_2	f_3
/ee/	beat	280	2620	3380
/i/	bit	360	2220	2960
/e/	bet	600	2060	2840
/er/	bird	560	1480	2520
/ar/	father	740	1110	2640
/a/	hut	760	1370	2500
/u/	hood	480	740	2620
/uu/	loot	320	920	2200

Πίνακας 1.2. *Οι βασικές συχνότητες συντονισμού για μερικά από τα φωνήεντα της Αγγλικής γλώσσας*

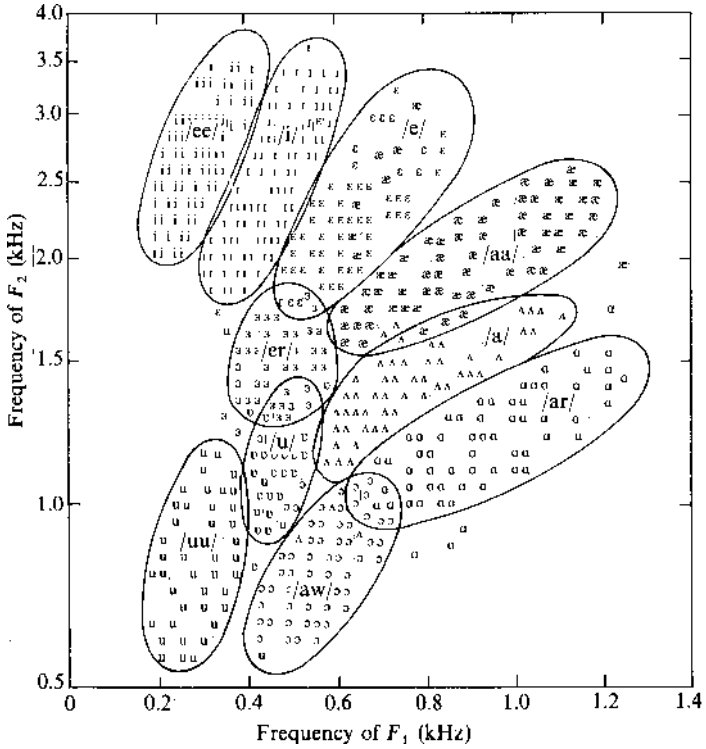
Στο σχήμα 1.4 μπορούμε να δούμε πως περίπου είναι η εσωτερική δομή της στοματικής κοιλότητας την ώρα που αρθρώνονται οι φθόγγοι /ee/ όπως 'ποίημα', /ar/ όπως 'αρχηγός' και /uu/ όπως 'ούτε'.



Σχ. 1.4. *Οι θέσεις που λαμβάνει το στόμα για την δημιουργία διαφόρων φθόγγων*

Όταν παράγουμε το /ee/ τότε η γλώσσα κινείται μπροστά και προς τα πάνω, μειώνοντας τον όγκο της στοματικής κοιλότητας. Αυτό έχει ως αποτέλεσμα την παραγωγή δεύτερων και τρίτων αρμονικών στις συχνότητες συντονισμού. Αντίθετα όταν προφέρουμε το /ar/ η γλώσσα πηγαίνει προς τα πίσω και κάτω, ενώ το στόμα ανοίγει, με αποτέλεσμα το μεγάλωμα της κοιλότητας και την παραγωγή υψηλότερων συχνοτήτων συντονισμού, που φτάνουν τα 750Hz με δεύτερη αρμονική τα 1100Hz. Τέλος στο /uu/ η πρώτη και η δεύτερη αρμονική χαμηλώνουν, σηκώνοντας την γλώσσα προς τον ουρανίσκο και εκτείνοντας τα χείλη προς τα έξω, οπότε έχουμε επιμηκύνση της στοματικής κοιλότητας. Στην περίπτωση αυτή οι τρεις συχνότητες συντονισμού είναι 300, 900, και 2500Hz.

Στο σχήμα 1.5 φαίνεται ένα διάγραμμα των δυο βασικών συχνοτήτων συντονισμού με φωνήεντα από διάφορους ομιλητές. Βλέπουμε ότι τα σύνολα σπάνια αλληλο-επικαλύπτονται και αυτό εξηγεί γιατί τα φωνήεντα είναι εύκολο να ξεχωρίσουν μεταξύ τους.



Σχ. 1.5. *Η κατανομή των βασικών συχνοτήτων για τα φωνήεντα, σε διάφορους ομιλητές.*

Δίφθογγοι

Οι δίφθογγοι είναι συνδιασμός δύο φθόγγων. Η στοματική κοιλότητα αλλάζει σιγά σιγά διάταξη έτσι ώστε να αλλάξει η βασική συχνότητα συντονισμού, οπότε έχουμε μια ομαλή μετάβαση αρμονικών.

Χαρακτηριστικά φωνής

Η μορφή της φωνής εξαρτάται από πολλούς παράγοντες όπως οι συχνότητα διέργεσης των φωνητικών χορδών, το μέγεθος του λάρυγγα, το μήκος των φωνητικών χορδών και τον τονισμό των συλλαβών. Επίσης, η ένταση της ομιλίας εξαρτάται από την συναισθηματική κατάσταση που βρίσκεται ο ομιλητής, από το πόσο μακρυά θέλουμε να μεταδοθεί το μύνημα, και από το πόσο θορυβώδες είναι το περιβάλλον. Ακόμη, κατά τη διάρκεια της ομιλίας δημιουργούνται διάφορα φωνητικά φαινόμενα τα οποία επηρεάζουν το τελικό αποτέλεσμα του λόγου.

Το πρώτο από αυτά τα φαινόμενα είναι η *προσαρμογή*. Αυτή προκύπτει επειδή το φωνητικό σύστημα κατά τη διάρκεια εκφώνησης κάποιου φθόγγου, προετοιμάζει τη θέση των οργάνων παραγωγής ήχου για τον επόμενο. Όταν μιλάμε γρήγορα τότε η προσαρμογή είναι πιό έντονη, γιατί η γλώσσα δεν πηγαίνει στις θέσεις που πρέπει όπως όταν μιλάμε αργά. Σε οριακές περιπτώσεις η προσαρμογή μπορεί να επιφέρει

αφομοίωση. Σ' αυτη την περίπτωση ο ήχος του φθόγγου αλλάζει και παίρνει πολλά χαρακτηριστικά από τον επόμενο με αποτέλεσμα την ενοποίηση του όπως για παράδειγμα στη λέξη 'ink' όπου το /n/ γίνεται /ng/.

Όταν δυο διαφορετικά όργανα κινούνται ταυτόχρονα για δυο διαφορετικούς φθόγγους τότε έχουμε *συνδυασμούς άρθρωσης*. Το αποτέλεσμα της χρήσης της προσαρμογής και των συνδυασμών άρθρωσης είναι η καλύτερη ποιότητα του ήχου.

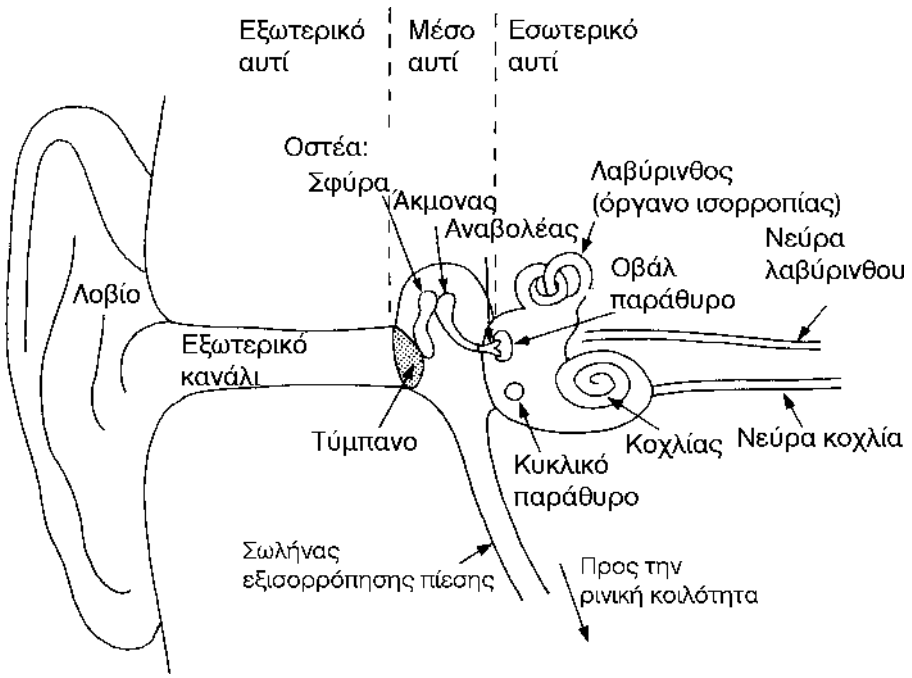
Δεν πρέπει όμως να βλέπουμε τον ήχο της ομιλίας απομονωμένα σαν σκέτους φθόγγους, αφού αυτοί αποτελούν την βασική μονάδα για την υλοποίηση των λέξεων, των προτάσεων και των φράσεων. Η ομιλία έχει και άλλα χαρακτηριστικά τα οποία εξαρτώνται από το νόημα της φράσης. Έτσι έχουμε το χαρακτηριστικό της έντασης. Η ένταση σημαίνει την αύξηση της έντασης, του τόνου, της διάρκειας και του τονισμού κάποιων συλλαβών. Ο τονισμός είναι ένα άλλο χαρακτηριστικό της φωνής. Σ' αυτή την περίπτωση έχουμε αλλαγή της βασικής συχνότητας κάνοντας τον τόνο της φωνής να ανεβαίνει ή να κατεβαίνει δίνοντας πρόσθετη σημασία, όπως για παράδειγμα στις ερωτήσεις. Τέλος υπάρχει και το χαρακτηριστικό της διάρκειας, όπου δύο προτάσεις που διαφέρουν σε νόημα ακούγονται το ίδιο, και ο μόνος τρόπος να διαχωριστούν είναι κάποια παύση στη μέση της πρότασης όπως 'an aim' και 'a name'. Ένα άλλο χαρακτηριστικό είναι η προφορά του κάθε ομιλητή ανάλογα με την καταγωγή του και την περιοχή όπου μεγάλωσε.

Ακουστική

Για να κατανοήσουμε τον τρόπο που ο άνθρωπος αντιλαμβάνεται την ομιλία και την αναγνωρίζει, είναι χρήσιμο να ερευνήσουμε όχι μόνο τον τρόπο παραγωγής αλλά και τον τρόπο λήψης των ήχων από το ακουστικό σύστημα του ανθρώπου.

Το αυτί

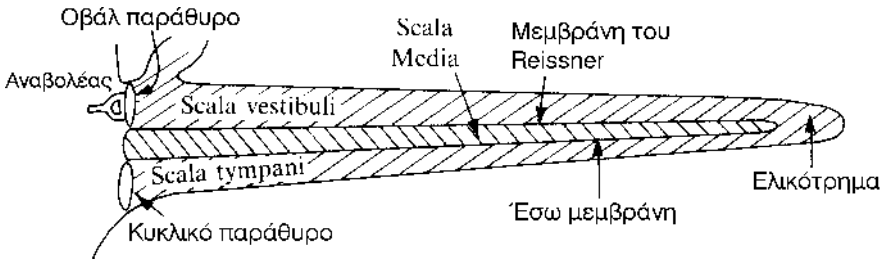
Το αυτί μπορεί να χωριστεί σε τρία βασικά κομμάτια. Το πρώτο είναι το εξωτερικό αυτί το οποίο προσφέρει προστασία στο μεσαίο και εσωτερικό αυτί. Επίσης είναι η διάταξη του επιτρέπει την μερική κατευθυντικότητα των λαμβανώμενων ήχων από την μπροστινή μεριά του κεφαλιού. Πρέπει να σημειωθεί βέβαια ότι η μηχανισμοί εντοπισμού της ηχητικής πηγής είναι με βάση την ένταση και το συγχρονισμό των δυο αυτιών.



Σχ. 1.6. *Το αυτί και τα όργανα της ακοής*

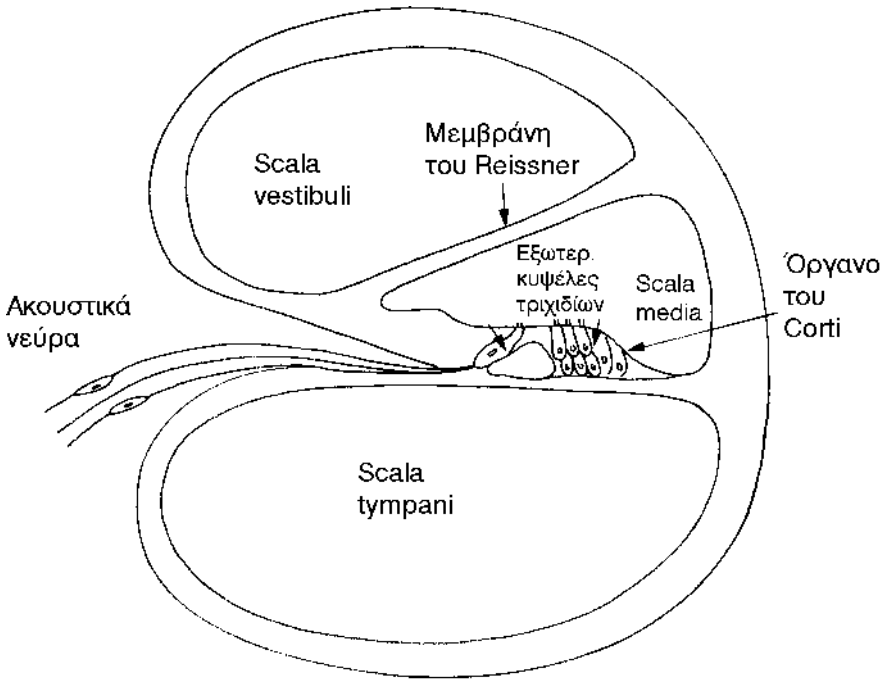
Το εξωτερικό αυτί χωρίζεται από το μεσαίο μέ το τύμπανο. Η διαδρομή από το εξωτερικό αυτί μέχρι το τύμπανο είναι περίπου 2.7 cm. Αυτό έχει ως αποτέλεσμα την επιλεκτική ενίσχυση των ήχων λόγω της συχνότητας ιδιοσυντονισμού, από 2000 ως 5500Hz με μέγιστη κορυφή τα 12db στα 4000Hz.

Το μέσο αυτί είναι ένας ενισχυτής. Η ενίσχυση επιτυγχάνεται με δυο τρόπους. Πρώτον με τρία οστά μεταφέρεται η πίεση από το τύμπανο σε μια ελλειψοειδή μεμβράνη. Η δεύτερον, υπάρχει ενίσχυση με τον λόγο επιφανειών του τυμπάνου προς τις αισθητήριες μεμβράνες, που είναι 18 φορές μικρότερες σε σχέση με την επιφάνεια του τυμπάνου. Αυτό έχει ως αποτέλεσμα την αύξηση της πίεσης κατά άλλα 30db. Μπροστά από το τύμπανο υπάρχουν προστατευτικοί μύες οι οποίοι προστατεύουν από βλάβες το αυτί όταν υπάρχει μεγάλη ηχητική ένταση. Το εσωτερικό αυτί μετατρέπει την μηχανική ενέργεια που έρχεται από το τύμπανο σε ηλεκτρικό ερέθισμα του εγκεφάλου, μέσω του κοχλία. Ο κοχλίας δεν έχει επίπεδη απόκριση, αλλά παράγει ερεθίσματα διαφορετικής έντασης ανάλογα με την συχνότητα.



Σχ. 1.7. *Το ανάπτυγμα του κοχλία.*

Το σημαντικότερο όργανο του κοχλία είναι το όργανο του Κόρτι (organ of Corti). Αυτό είναι το όργανο που μετατρέπει τα ακουστικά ερεθίσματα σε ηλεκτρικά σήματα. Περιέχει τριχοειδείς κυψέλες εσωτερικά και εξωτερικά. Οι εξωτερικές κυψέλες είναι πιο πολλές από τις εσωτερικές. Οι κυψέλες είναι διατεταγμένες σε γραμμές, και εσωτερικά αποτελούνται από 15000 τριχίδια. Κάθε κυψέλη έχει διαστάσεις 35μm μήκος επί 10μm διάμετρο. Τα νεύρα συνδέονται στις απολήξεις της κάθε κυψέλης που υποστηρίζει πάνω από τρείς σειρές από τριχίδια. Τα νεύρα αυτά μεταφέρουν πληροφορίες και προς τις δύο κατευθύνσεις, δηλαδή και από τις κυψέλες προς τον εγκέφαλο και από τον εγκέφαλο προς τις κυψέλες. Η απόκριση των κυψελών είναι πολώμενη προς τη μία κατεύθυνση, δηλαδή παράγουν ηλεκτρικό σήμα όταν διαγείρονται προς τη μία κατεύθυνση ενώ δεν παράγουν σήμα πρός την αντίθετη κατεύθυνση. Τα περισσότερα νεύρα συνδέονται στις εσωτερικές κυψέλες. Λέγεται ότι υπάρχει κάποιου είδους μηχανική ανάδραση μεταξύ των εσωτερικών και των εξωτερικών κυψελών ώστε να είναι δυνατή η στενή απόκριση σε συγκεκριμένες συχνότητες.



Σχ. 1.8. *Το εσωτερικό του κοχλία (τομή).*

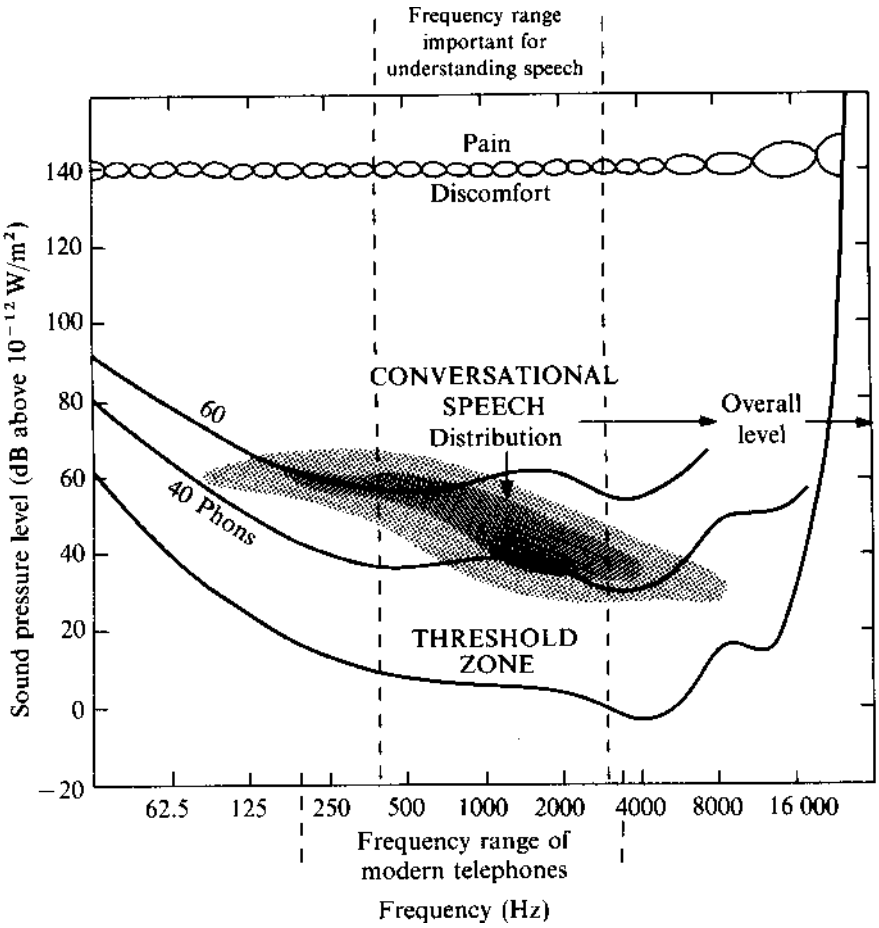
Το ακουστικό νεύρο

Το ακουστικό νεύρο χρησιμοποιεί τρεις τρόπους μεταφοράς της πληροφορίας προς τον εγκέφαλο. Στέλνει ριπές σήματος στον εγκέφαλο όσο υπάρχει η απουσία σχετικού σήματος. Το κάθε ακουστικό νεύρο έχει επιλεκτικότητα, δηλαδή καλύτερη απόκριση σε ορισμένες συχνότητες. Τέλος το νεύρο στέλνει ηλεκτρικά ερεθίσματα κατά ριπές που παρακολουθούν την φάση του βασικού σήματος που διεγείρει την μεμβράνη. Στο διάγραμμα της συχνότητας σε σχέση με την πίεση του ήχου (σχ. 1.9) παρατηρούμε ότι η ένταση που χρειάζεται για να ενεργοποιηθεί ένας νευρώνας είναι μικρή για κάποια ορισμένη χαρακτηριστική συχνότητα. Η καμπύλη από τις χαμηλότερες συχνότητες προς την χαρακτηριστική είναι λιγότερο απότομη από οτι

είναι στην χαρακτηριστική μέχρι τις υψηλότερες συχνότητες. Ο ρυθμός αποφόρτισης του νεύρου και η ένταση του ήχου μέχρι ένα σημείο είναι ανάλογα μεγέθη. Η ενεργοποιήση του νευρώνα αρχίζει από τα 20db περίπου και ο κορεσμός είναι στα 50db. Οι νευρώνες παρακολουθούν τη φάση στην περιοχή μεταξύ 4 και 5KHz. Σε μεγαλύτερες συχνότητες δεν υπάρχει παρακολούθηση της φάσης.

Όταν έχουμε παρουσία δύο τόνων η πρώτη συχνότητα μπορεί να μειωθεί ως προς τη δεύτερη. Αυτό το φαινόμενο καλείται απόκρυψη (two tone suppression). Αν η δεύτερη συχνότητα είναι πολύ κοντά στην πρώτη και μέσα στην περιοχή των ριπών, τότε αυτή ενισχύεται. Αν είναι εκτός της περιοχής αυτής, καταπνίγεται. Αυτό συμβαίνει λόγω της μη γραμμικότητας της μεμβράνης. Σε πειράματα που έγιναν μετρήθηκε μια σύνθετη συχνότητα $2f_1-f_2$ όπου f_1 η μια συχνότητα, f_2 η δεύτερη και $f1<f_2$.

Το ανθρώπινο αυτί δεν είναι το ίδιο ευαίσθητο σε όλη την περιοχή των συχνοτήτων. Η μεγαλύτερη ευαισθησία είναι μεταξύ των 1000Hz και 4000Hz. Οι υψηλότερες και χαμηλότερες συχνότητες χρειάζονται μεγαλύτερη ενέργεια για να γίνουν αντιληπτές. Στο παρακάτω σχήμα 1.9 βλέπουμε την απόκριση συχνότητας σε σχέση με την ένταση με 0 db σε πίεση 10^{-12} W/m^2.



Σχ. 1.9. *Η απόκριση του αυτιού και τα όρια ακουστότητας.*

Η χαμηλότερη καμπύλη δείχνει την ελάχιστη ένταση που απαιτείται για να γίνει αντιληπτός ο ήχος. Αυτή είναι μια τυπική καμπύλη και διαφέρει ανάλογα με τον άνθρωπο μέχρι και 20db. Η ευαισθησία στις υψηλές συχνότητες μειώνεται με την ηλικία. Οι άλλες δύο καμπύλες δείχνουν τα σημεία που έχουν την ίδια ένταση για το αυτί σε phon. Το phon μετριέται σε db με πίεση πάνω από 10^{-12} W/m^2 στα 1000Hz. Τέλος η επιφάνεια στη μέση δείχνει την κατανομή της ανθρώπινης ομιλίας, το μεγαλύτερο ποσοστό της οποίας πέφτει μέσα στην ευαίσθητη περιοχή του αυτιού. Μικρές αλλαγές στην πίεση (0.5-2db) μπορούν να γίνουν αντιληπτές λόγω του εύρους που μπορεί να μεταβάλλεται ο ρυθμός των ριπών στα νεύρα.

Διάκριση των τόνων

Η ανίχνευση των τόνων μας επιτρέπει να κατατάξουμε τους ήχους στη μουσική κλίμακα. Όταν το αυτί ακούει ένα καθαρό τόνο τότε αντιλαμβανόμαστε τη συχνότητα του. Όταν ακούμε ένα σύνθετο τόνο που περιέχει πολλές διαφορετικές συχνότητες τότε ακούμε τη βασική αρμονική. Υπάρχουν δυο θεωρίες που εξηγούν τον τρόπο διάκρισης των τόνων. Η πρώτη (place theory) συσχετίζει τον τόνο με την θέση της μέγιστης διέργεσης της βασικής μεμβράνης. Η δεύτερη (temporal theory) συσχετίζει τον τόνο με τις χρονικές παραλλάγες των νευρωνικών παλμών. Καμιά από τις δυο θεωρίες δεν εξηγεί πλήρως την λειτουργία του αυτιού. Ίσως η λύση να είναι ο συνδιασμός τους.

Με διάφορα πειράματα οι επιστήμονες διαπίστωσαν ότι το αυτί μπορεί να ξεχωρίσει μια διαφορά 2Hz στα 1000Hz με ηχητική πίεση 60-70db. Η ακρίβεια είναι καλύτερη για τις μεσαίες συχνότητες και για μεγαλύτερες διάρκειες των τόνων. Στη περίπτωση των σύνθετων τόνων ο άνθρωπος μπορεί να ξεχωρίσει κάποιες από τις δευτερεύουσες συχνότητες. Αυτό καλείται ακουστικός νόμος του Ohm και ισχύει στους τόνους που αποτελούνται από συχνότητες που δεν είναι αρμονικές μεταξύ τους ή είναι πολύ διαφορετικές. Μια άλλη δυνατότητα του αυτιού είναι, όταν έχουμε ένα σύνθετο τόνο από υψηλές αρμονικές αυτό να ακούει και τη βασική αρμονική. Αυτός ο τόνος καλείται περιοδικός ή διαφορικός τόνος (residue or periodicity pitch).

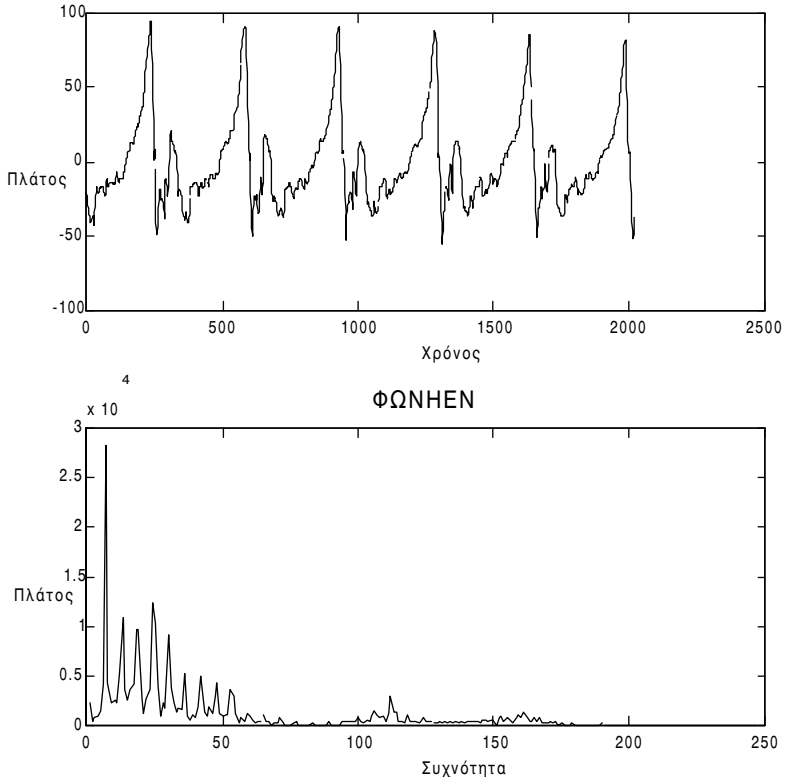
Είναι σημαντικό για τους ανθρώπους να μπορούν να διαχωρίσουν διαφορετικές συχνότητες, όπως για παράδειγμα στην ομιλία. Κάθε φωνήεν έχει πολλές συχνότητες που πρέπει να γίνουν αντιληπτές ταυτόχρονα ώστε να γίνει η αναγνώριση. Ο Fletcher (1940) πρώτος διατύπωσε τη θεωρία ότι οι διάφορες συχνότητες γίνονται αντιληπτές σάν από μια σειρά αλληλοεπικαλυπτόμενων ζωνοπερατών φίλτρων που η βασική τους συχνότητα αλλάζει σαρώνοντας όλη την ακουστική περιοχή. Αυτή η θεωρία έχει επιβεβαιωθεί με διάφορα πειράματα που έγιναν. Η δυνατότητα διαχωρισμού δύο γειτονικών συχνοτήτων που είναι πολύ κοντά μεταξύ τους, περιορίζεται από το πλάτος του ενός εκ των δύο ακουστικών φίλτρων το οποίο καλείται κρίσιμο πλάτος.

Η επιλεκτικότητα των συχνοτήτων συσχετίζεται και με την δυνατότητα της απόκρυψης συχνότητας. Αυτό σημαίνει ότι ένας ήχος παύει να ακούγεται μόλις εμφανιστεί ένας άλλος ήχος, όπως γίνεται στο αυτοκίνητο όταν ανάβουμε το ραδιόφωνο, όπου δεν ακούμε τον θόρυβο του αυτοκινήτου με την ίδια ένταση. Η απόκρυψη αυτή είναι πολύ πιο αποτελεσματική όταν οι δυο συχνότητες είναι πολύ κοντά μεταξύ τους. Ο Fletcher έκανε πειράματα για την απόκρυψη των συχνοτήτων. Χρησιμοποίησε μια γεννήτρια ημιτόνου, και μια γεννήτρια λευκού θορύβου με μεταβλητό εύρος. Παρατήρησε λοιπόν ότι όσο αύξανε το εύρος του θορύβου (με σταθερή την ισχύ) τόσο αύξανε και το επίπεδο της έντασης που χρειαζόταν για να γίνει αντιληπτή η βασική συχνότητα του ημιτόνου. Αυτό συνέβαινε μέχρι ενός σημείου, γιατί κατόπιν όσο αυξάνοταν το εύρος του θορύβου, δεν υπήρχε καμία αλλαγή στο επίπεδο της ένταση που ήταν απαραίτητη για την αναγνώριση του ήχου. Αυτό

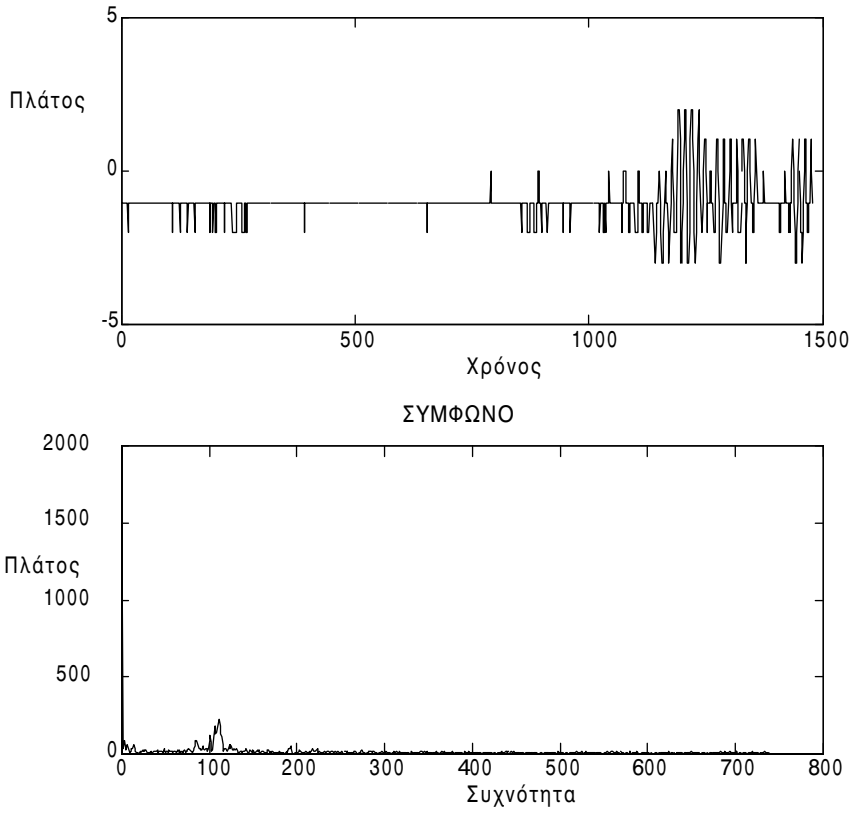
το εύρος το καλούμε κρίσιμο. Το κρίσιμο εύρος είναι περίπου το 10-20% της βασικής συχνότητας. Αυτή η περίπτωση απόκρυψης είναι η ταυτόχρονη. Υπάρχει και η δυνατότητα προ-απόκρυψης όπου ο ήχος μπορεί να αποκρυφθεί από έναν προγενέστερο του ή αντίστοιχα για την μετά-απόκρυψη από ένα μεταγενέστερο.

Χαρακτηριστικά ομιλίας

Το σήμα της ομιλίας είναι ένα πολυδιάστατο σήμα όπου το πλάτος, η συχνότητα και η φάση μεταβάλλονται συνεχώς. Αυτό καθιστά την αναγνώριση ομιλίας από τις μηχανές ένα ιδιαίτερα δύσκολο πρόβλημα. Στο παρακάτω σχήμα 1.10 βλέπουμε ένα φωνήεν στο πεδίο του χρόνου, δηλαδή όπς θα το βλέπαμε σ' ένα παλμογράφο. Παρατηρείστε την ένταση του σήματος και τη μορφή του. Στα φωνήεντα έχουμε μεγάλο πλάτος, και περιοδικότητα κάποιας βασικής συχνότητας. Στο πεδίο των συχνοτήτων αυτό σημαίνει μεγάλη ισχύ στις χαμηλές συχνότητες με κορυφές στις βασικές συχνότητες, και μειωμένη ισχύ στις υψηλές. Αντίθετα ένα σύμφωνο όπως φαίνεται στο σχήμα 1.11 έχει μικρό πλάτος και διασχίζει πολλές φορές τον άξονα του μηδενός (πολλές εναλλαγές προσήμου). Στο πεδίο των συχνοτήτων φαίνεται η χαμηλή ισχύς του σήματος, ενώ αντίθετα είναι σαφές ότι το εύρος των συχνοτήτων είναι πολύ μεγάλο. Παρατηρούμε ότι το φάσμα μοιάζει με λευκό θόρυβο (όπως ο ήχος που παράγεται από τον αέρα που βγαίνει μέσα από τα δόντια στο 'σ'), ενώ δεν υπάρχει κάποια συγκεκριμένη συχνότητα που να συγκεντρώνει μεγάλη ενέργεια.

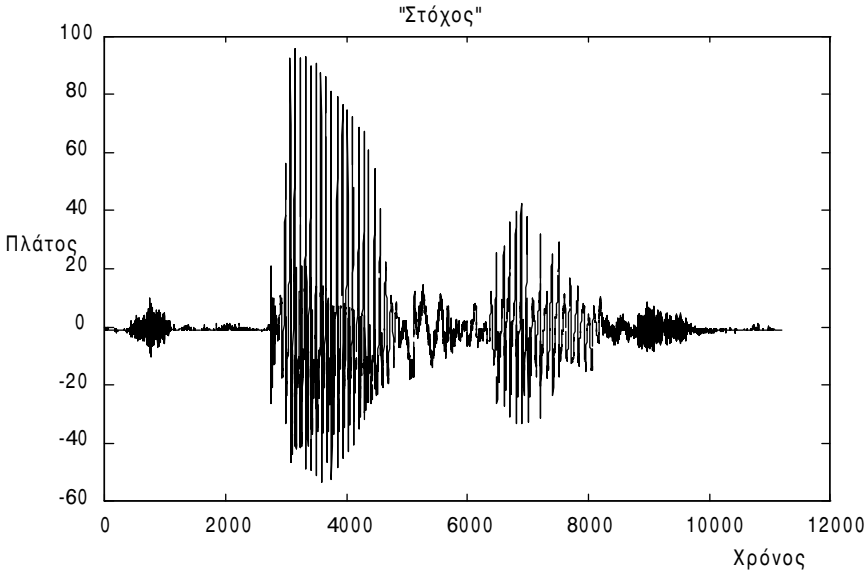


Σχ. 1.10. *Η κυματομορφή ενός φωνήεντος και το φάσμα του.*



Σχ. 1.11. *Η κυματομορφή ενός συμφώνου και το φάσμα του.*

Το πρώτο πρόβλημα που αντιμετωπίζει κανείς στην προσπάθεια κατασκευής ενός συστήματος αναγνώρισης ομιλίας είναι η αναγνώριση της αρχής και του τέλους μιας λέξης. Η αρχή μιας λέξης μπορεί να βρεθεί με την εύρεση της ενέργειας και τον αριθμό των εναλλαγών γύρω από τον άξονα του μηδενός. Όταν οι τιμές αυτές αυξηθούν πάνω από κάποιο όριο, έχουμε εντοπίσει την αρχή της λέξης. Το τέλος μιας λέξης είναι πιο δυσκολο να βρεθεί. Στο σχήμα 1.12 βλέπουμε τη λέξη 'στόχος'. Παρατηρείστε τα σύμφωνα 'στ' όπου έχουμε κυματομορφή παρόμοια με θόρυβο, και ένα μεγάλο περιθώριο ησυχίας στο 'τ'. Έτσι φαίνεται ότι δεν μπορούμε να προσδιορίσουμε το τέλος μιας λέξης απλώς και μόνο περιμένοντας τη σιγή, αλλά πρέπει να ληφθούν υπ' όψη και άλλοι παράγοντες. Ένα άλλο πρόβλημα είναι το ότι ο ομιλητής δεν μιλάει πάντα με την ίδια ταχύτητα αλλά μπορεί να εκφράσει την ίδια λέξη με πολλές διαφορετικές ταχύτητες ανάλογα με την περίπτωση. Ένα σύστημα αναγνώρισης θα πρέπει λοιπόν να λάβει και την ταχύτητα εκφώνησης υπ' όψη του.



Σχ. 1.12. *Η κυματομορφή της λέξης "στόχος".*

Ένα από τα πολλά μοντέλα αναγνώρισης της ομιλίας που έχουν προταθεί αναλύει τη διαδικασία σε διάφορα βήματα. Πρώτα ο βασικός ακουστικός μηχανισμός απομονώνει τα βασικά στοιχεία της ομιλίας. Κατόπιν τα στοιχεία αυτά φιλτράρονται για να εντοπιστούν οι βασικές συχνότητες. Το επόμενο βήμα εντοπίζει τα χαρακτηριστικά των φθόγγων. Τέλος γίνεται τμηματική και λεξική ανάλυση. Βλέπουμε δηλαδή μια μεθοδολογία από κάτω προς τα επάνω (down-top) όπου σε κάθε βήμα επεξεργαζόμαστε σε πιο υψηλό επίπεδο την πληροφορία ώστε στο τέλος να καταλήξουμε σε λέξεις.

Διαδικασίες ανάλυσης φωνής

Για την ανάλυση και την επεξεργασία της φωνής χρησιμοποιούνται δυο προσεγγίσεις. Η πρώτη προσέγγιση χρησιμοποιεί μεθόδους που εξάγουν συμπεράσματα αναλύοντας τη φωνή στο πεδίο του χρόνου, ενώ η δεύτερη προσέγγιση αναλύει τα δεδομένα της φωνής στο πεδίο των συχνοτήτων. Αν κάνουμε μια αντίστοιχη διαστρωμάτωση στην επεξεργασία όπως ο άνθρωπος τότε η επεξεργασία του ήχου έχει τα εξής στάδια:

1. Επίπεδο πρωτογενούς επεξεργασίας σήματος (prefiltering, αρχή-τέλος λέξης κλπ)
2. Επίπεδο φθόγγου (διαχωρισμός φθόγγων, δυναμικός προγραμματισμός)
3. Επίπεδο λέξης (αναγνώριση, νοηματική επαλήθευση)

Με αυτή τη μεθολογία είναι φανερό το είδος της επεξεργασίας σε κάθε βήμα. Στο πρώτο επίπεδο η επεξεργασία γίνεται στο πεδίο του χρόνου. Στο δεύτερο επίπεδο γίνεται κυρίως στο πεδίο των συχνοτήτων, ενώ στο τελευταίο επίπεδο η ανάλυση - επεξεργασία ξεφεύγει από τα στενά όρια της επεξεργασίας σήματος και μπορεί να είναι ένα στατιστικό σύστημα, ένα νευρωνικό δίκτυο ή ένα σύστημα ασαφούς λογικής.

Στο πεδίο του χρόνου μπορούμε να έχουμε σταθερές επεξεργασίες λόγω του ότι ο ήχος παρουσίαζεται στον επεξεργαστή ως μια ακολουθία αριθμών, όπου μια σύντομη επεξεργασία από δείγμα σε δείγμα αρκεί για να βγούν τα πρώτα συμπεράσματα.

Οι βασικές μέθοδοι επεξεργασίας συνεχούς ροής στο πεδίο του χρόνου είναι:
1. Μέσο πλάτος
2. Κατανομή πλάτους
3. Μέση ενέργεια
4. Ρυθμός διάβασης από το μηδέν

Μέσο πλάτος

Το μέσο πλάτος είναι ίσως ο ευκολότερος τρόπος να καταλάβει κανείς την ύπαρξη μιας λέξης ή αν υπάρχει φωνήεν ή σύμφωνο σε κάποια χρονική περιοχή. Το μέσο πλάτος υπολογίζεται σε ένα παράθυρο από τα τελευταία N δείγματα. Η σχέση είναι:

$$M(n)=\frac{1}{N}\sum_{m=n-N+1}^{n}|x(m)| \qquad \text{(Εξ. 1.1)}$$

Όπου $x(m)$ είναι το τρέχον δείγμα από μια απειροστή ακολουθία, ενώ το μέσο πλάτος είναι ένα παράθυρο στο χρόνο της ακολουθίας μήκους N.

Μια εναλλακτική μέθοδος υπολογισμού είναι η συνέλιξη μιας συνάρτησης παραθύρου με την απειρόστη ακολουθία των δειγμάτων έτσι ώστε :

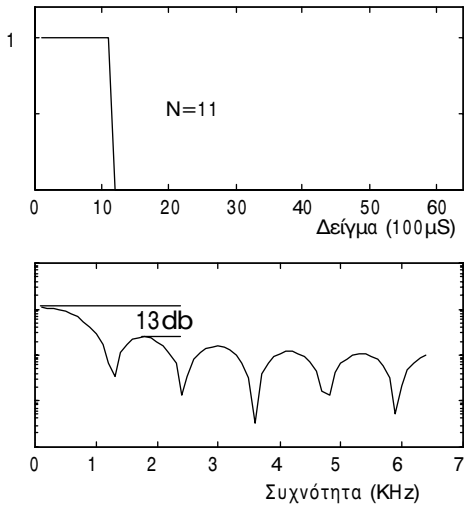
$$M(n)=\frac{1}{N}\sum_{m=-\infty}^{\infty}|x(m)|\cdot w(n-m) \qquad \text{(Εξ. 1.2)}$$

Λόγω του ότι η συνέλιξη στο πεδίο του χρόνου ισοδυναμεί με πολλαπλασιασμό στο πεδίο των συχνοτήτων το μέσο πλάτος πρακτικά αντιστοιχεί με ένα φίλρο με απόκριση $w()$. Τελικά η έξοδος της διαδικασίας θα είναι ανάλογη με ένα βαθυπερατό φίλτρο. Αυτό φαίνεται εύκολα αν υποθέσουμε ότι η $w(n)$ γράφεται

$$w(n)=\begin{cases}1, & 0\le n\le \mathrm{N}-1\\ 0, & n<0,\ n>\mathrm{N}-1,\end{cases} \qquad \text{(Εξ. 1.3)}$$

που ισοδυναμεί με ένα τετράγωνο παράθυρο.

Η φασματική απόκριση ενός τετραγωνικού παραθύρου μήκους N=11 με συχνότητα δειγματοληψίας 10KSamples/sec φαίνεται στο σχήμα 1.13



Σχ. 1.13. *Τεραγωνικό παράθυρο μήκους 11 και η απόκρισή του.*

Είναι φανερό ότι η απόκρισή του είναι ένα βαθυπερατό φίλτρο με συχνότητα αποκοπής τα 700Hz. Αυτό αντιστοιχεί σε υποδειγματοληψία αφού ο ρυθμός δειγματοληψίας από 10KSamples/sec μπορεί να κατέβει στα 1.4Ksamples/sec σύμφωνα με το θεώρημα της δειγματοληψίας. Το τετραγωνικό παράθυρο δεν είναι το καλύτερο που υπάρχει αλλά είναι πολύ εύκολο στην υλοποίησή του. Υπάρχουν παράθυρα όπως πχ. του Hamming όπου οι λοβοί στο πεδίο των συχνοτήτων έχουν πολύ μικρότερη ισχύ από αυτούς του τετραγωνικού παραθύρου.

Το μέσο πλάτος μπορεί να χρησιμοποιηθεί σε ένα σύστημα αυτομάτου ελέγχου της ενίσχυσης (AGC) ώστε να υπάρχει σταθερό πλάτος στα επόμενα στάδια της επεξεργασίας, όπως για παράδειγμα όταν το μικρόφωνο είναι σταθερό και ο ομιλητής κίνειται μέσα στο χώρο.

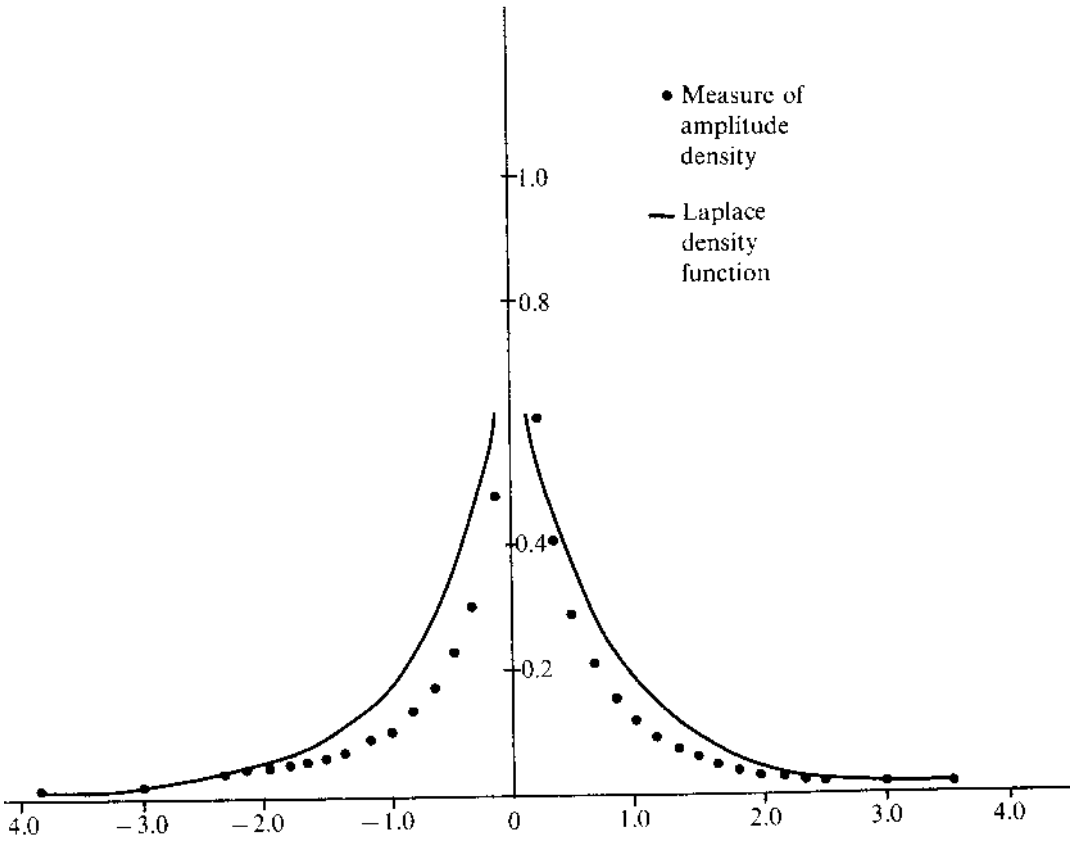
Κατανομή πλάτους

Το μέσο πλάτος μας δίνει τα χαρακτηριστικά της φωνής για μικρές χρονικές περιόδους. Σε περίπτωση που θέλουμε να γνωρίζουμε τα χαρακτηριστικά της φωνής που παράγονται σε μεγάλες χρονικές περιόδους, δεν αρκεί η χρήση του μέσου πλάτους. Σε αυτές τις περιπτώσεις χρησιμοποιείται η κατανομή πλάτους. Η κατανομή πλάτους είναι η στατιστική κατανομή του πλάτους των δειγμάτων. Η κατανομή των πιθανοτήτων μιας σειράς δειγμάτων μπορεί να υπολογιστεί από την αναλογία $p(x)$ των δειγμάτων εισόδου x_i που βρίσκονται στο διάστημα $x < x_i < x+dx$. Οι τιμές των δειγμάτων της ομιλίας βρίσκονται στο διάστημα από x=-∞ ως x=+∞ έχουμε ότι:

$$\int_{-\infty}^{+\infty} p(x)dx = 1 \qquad \text{(Εξ. 1.4)}$$

Το άνω όριο της περιοχής τιμών των πιθανοτήτων στην κατανομή είναι ίσο με τη μονάδα.

Στο παρακάτω σχήμα 1.14 έχουμε τη γραφική απεικόνιση της κατανομής των πιθανοτήτων από ομιλία μήκους 12 δευτερολέπτων, με δειγματοληψία 16Ksamples/sec και διακριτικότητα 14bits. Το δείγμα είναι οι αριθμοί από μηδέν ως εννιά από μια γυναίκα και έναν άντρα.



Σχ. 1.14. *Η κατανομή πλάτους μερικών λέξεων και η κατανομή Laplace.*

Οι τιμές των δειγμάτων έχουν κανονικοποιηθεί με βάση την rms τιμή του συνολικού δείγματος δηλαδή ένα δείγμα με πλάτος ίσο με τη ολική rms τιμή έχει τιμή 1.0. Εκτός από τα δείγματα στο γράφημα φαίνεται και η συνάρτηση πυκνότητας του Laplace

$$p(x) = \frac{1}{\sqrt{2}\sigma_x} \exp\left(-\frac{\sqrt{2}|x|}{\sigma_x}\right) \qquad \text{(Εξ. 1.5)}$$

Η συγκεκριμένη συνάρτηση έχει το πλεονέκτημα ότι σχετίζεται άμεσα με το πλάτος rms του σήματος.

Η κατανομή του πλάτους είναι χρήσιμη όταν πρέπει να γίνει κλιμάκωση του αναλογικού σήματος που δειγματοληπτείται από ένα γραμμικό μετατροπέα A/D, βοηθώντας μας να επιλέξουμε τις τιμές εξόδου που χρησιμοποιούνται περισσότερο και να απορρίψουμε αυτές που συμμετέχουν λιγώτερο.

Μέση ενέργεια

Η μέση ενέργεια είναι ο μέσος όρος των στιγμιαίων ενεργειών των δειγμάτων για κάποιο ορισμένο χρονικό διάστημα. Η ενέργεια σε κάποιο χρονικό διάστημα ορίζεται :

$$E(n) = \frac{1}{N} \sum_{m=-\infty}^{+\infty} [x(m)w(n-m)]^2 \qquad \text{(Εξ. 1.6)}$$

όπου $w()$ είναι η κατάλληλη συνάρτηση παραθύρου για να απομονώσει τα δείγματα που αντιστοιχούν στην περιοχή που μας ενδιαφέρει (η μέση ενέργεια γύρω από το δείγμα m και σε απόσταση n δειγμάτων).

Η παραπάνω εξίσωση μπορεί να γραφεί και ως:

$$E(n) = \sum_{m=-\infty}^{+\infty} x^2(m)h(n-m) \qquad \text{(Εξ. 1.7)}$$

με

$$h(n) = \frac{1}{N} w(\ )^2 \qquad \text{(Εξ. 1.8)}$$

Η μέση ενέργεια περιορισμένης χρονικής διάρκειας από την παραπάνω σχέση, προκύπτει οτι μπορεί να υπολογιστεί αν φιλτράρουμε το τετράγωνο της εισόδου με τη συνάρτηση $h()$.

Η μέση ενέργεια όπως φαίνεται και στο συνοπτικό σχήμα 1.15 είναι σε μορφή παρόμοια με το μέσο πλάτος, αλλά έχει τονίσει τις διαφορές μεταξύ των φωνήεντων και των συμφώνων. Αυτός είναι και ο λόγος που η μέση ενέργεια προτιμάται για την ανίχνευση της ύπαρξης μιας λέξης. Συνήθως χρησιμοποιείται όμως σε συνδιασμό με το ρυθμό διάβασης από το μηδέν.

Ρυθμός διάβασης από το μηδέν

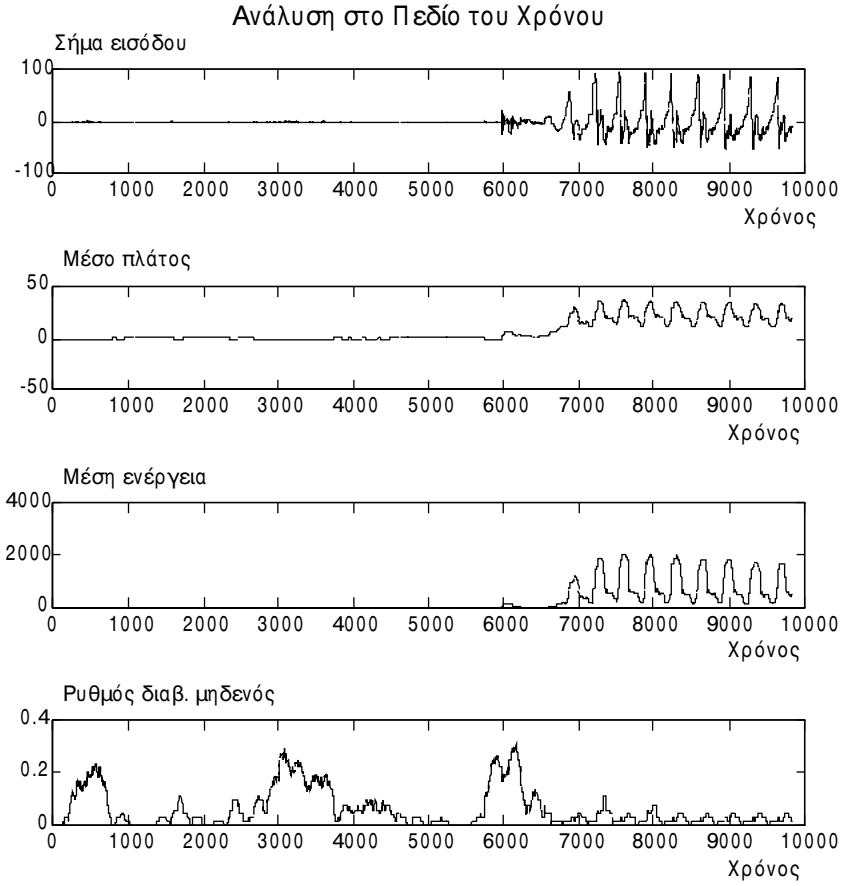
Η σημαντικότερη διαφορά μεταξύ των φωνηέντων και των συμφώνων είναι ο ρυθμός αλλαγής προσήμου. Τα σύμφωνα επειδή η κυματομορφή τους μοιάζει με λευκό θόρυβο με κατανομή Gauss, παρουσιάζουν πολλές εναλλαγές στο πρόσημο των δειγμάτων τους. Ο ρυθμός εναλλαγής του προσήμου για ένα φωνήεν είναι 0.5 εναλλαγές /ms ενώ για ένα σύμφωνο 3 εναλλαγές/ms. Ο ρυθμός αυτός μπορεί να υπολογιστεί εύκολα με ένα συγκριτή προσήμου, ο οποίος θα συγκρίνει το τρέχον δείγμα για να διαπιστώσει αν αυτό έχει θετική ή αρνητική τιμή. Το αποτέλεσμα θα είναι 0 για δείγματα εισόδου πάνω από το μηδέν και 1 για δείγματα μικρότερα από το μηδέν.

Ο ρυθμός των εναλλαγών είναι:

$$Z(n) = \frac{1}{2N} \sum_{m=-\infty}^{+\infty} \left| sign(x(m)) - sign(x(m-1)) \right| \ w(n-m) \qquad \text{(Εξ. 1.9)}$$

Το πρόβλημα της εύρεσης του ρυθμού εναλλαγών από το μηδέν είναι ότι επηρρεάζεται από dc συνιστώσες ή από θόρυβο, λόγω του ότι η στάθμη του σήματος στα σύμφωνα είναι πολή χαμηλή. Η λύση στο πρόβλημα αυτό είναι η χρήση ενός υψιπερατού φίλτρου με συχνότητα διέλευσης τα 70Hz, ώστε να αποκοπεί και ο θόρυβος από την τροφοδοσία (50 Hz) που πιθανόν να υπάρχει.

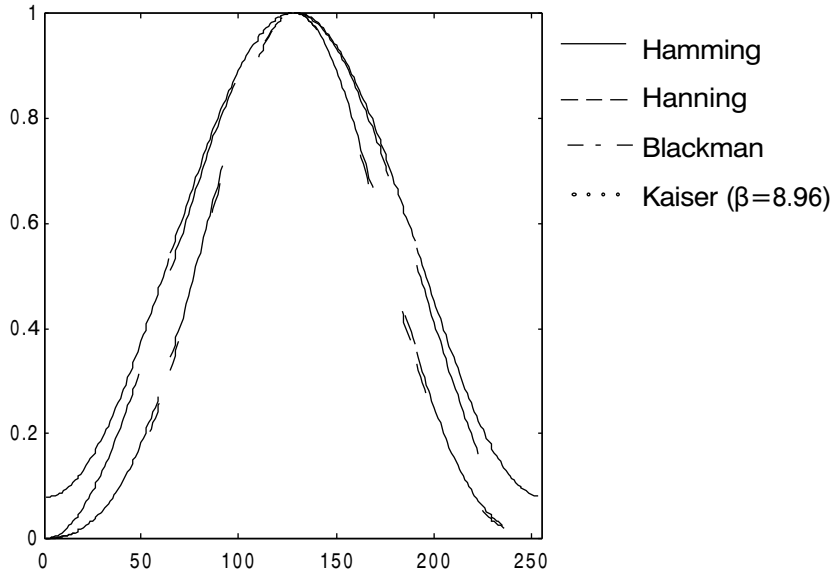
Ακολουθεί ένα συνοπτικό σχήμα με ένα δείγμα και τα αποτελέσματα της κάθε μιας απο τις προηγούμενες μεθόδους στο δείγμα αυτό.



Σχ. 1.15. *Τμήμα μιας λέξης με τα χαρακτηριστικά του στο πεδίο του χρόνου.*

Επεξεργασία φθόγγων

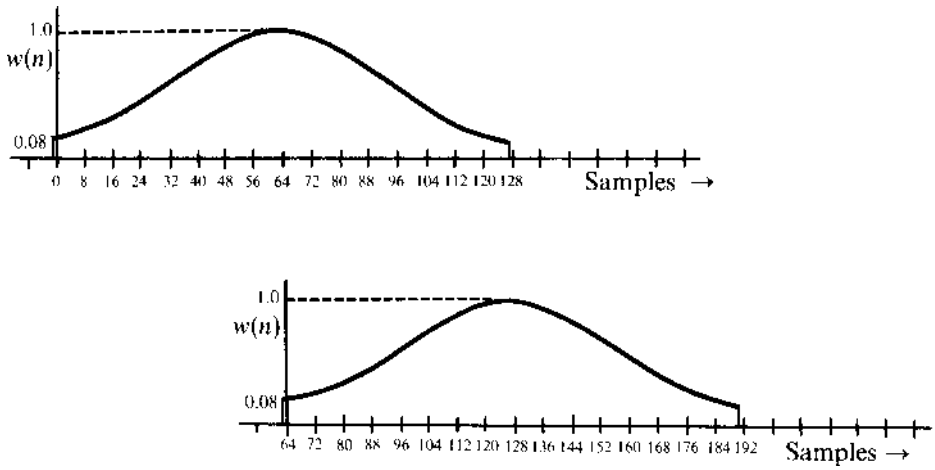
Στο επίπεδο των φθόγγων γίνεται τμηματική ανάλυση σε αντίθεση με την πρωτογενή επεξεργασία όπου έχουμε επεξεργασία σε συνεχή ροή δειγμάτων. Αυτό σημαίνει ότι η επεξεργασία γίνεται σε μια ομάδα δεδομένων όπως για παράδειγμα το FFT (Fast Fourier Transform) όπου υπολογίζει τα συχνοτικά χαρακτηριστικά σε ομάδα δειγμάτων με μήκος, στην βέλτιστη περίπτωση, μια δύναμη του 2. Στην τμηματική ανάλυση μαζεύεται ένας πίνακας δειγμάτων, ο οποίος πολλαπλασιάζεται με μια συνάρτηση παραθύρου και κατόπιν ακολουθεί η κυρίως επεξεργασία. Ο πολλαπλασιασμός με το παράθυρο έχει σκοπό το φιλτράρισμα του σήματος εισόδου λόγω του ότι αυτό δεν είναι ενα συνεχές σήμα, αλλά ένα κομμάτι από αυτό, οπότε άν το θεωρήσουμε αυτόνομο, αλλοιώνονται τα συχνοτικά χαρακτηριστικά των δεδομένων. Για να είναι αυτή η αλλοίωση ελάχιστη χρησιμοποιούμε τα παράθυρα. Το τετραγωνικό παράθυρο δεν είναι και τόσο αποτελεσματικό αφού οι δευτερεύοντες λοβοί έχουν μεγάλη ισχύ σε σχέση με τον κύριο. Έτσι χρησιμοποιούμε άλλα παράθυρα όπως αυτά του Hanning, Hamming ή του Kaiser. Στον παρακάτω πίνακα φαίνονται κάποιοι τύποι παραθύρων.



Σχ. 1.16. *Μερικοί τύποι παραθύρων.*

Βέβαια ένα παράθυρο πολλές φορές αν δεν πραγματοποιηθεί τη σωστή χρονική στιγμή μπορεί να απορρίψει κάποια σημαντικά χαρακτηριστικά από το σήμα εισόδου, όπως για παράδειγμα μπορεί να συμβεί με το γράμμα "τ". Το παράθυρο θα αποκόψει την απότομη εξαγωγή του αέρα οπότε και το φασμα που θα προκύψει από την επεξεργασία, δεν θα περιέχει το φαινόμενο αυτό.

Αυτός είναι και ο λόγος που για να μην αποκόψουν πληροφορίες από το σήμα εισόδου, τα παράθυρα επεξεργάζονται ομάδες δειγμάτων που είναι αλληλοεπικαλυπτόμενες έτσι ώστε κάθε δείγμα να βρίσκεται σε τουλάχιστον δυο ομάδες για επεξεργασία, όπως και στο σχήμα 1.17.



Σχ. 1.17. *Η μέθοδος τμηματικής ανάλυσης με αλληλοεπικαλυπτόμενα παράθυρα.*

Οι επεξεργασίες τμηματικής ανάλυσης είναι
1. (Ταχύς) μετασχηματισμός Φουριέ (FFT)
2. Αυτό συσχέτιση (Auto-covariance and correlation)
3. Πυκνότητα φάσματος (Power Spectral Density)
4. Cepstral analysis

Ο Μετασχηματισμός Φουριέ

Το φάσμα ενός περιοδικού σήματος που έχει υποστεί δειγματοληψία υπολογίζεται από το γνωστό διάκριτο μετασχηματισμό του Φουριέ (DFT) :

$$X(k)=\sum_{n=0}^{N-1}x(n)e^{-j2\pi nk/N} \qquad \text{(Εξ. 1.10)}$$

Ο διάκριτος μετασχηματισμός του Φουριέ δίνει μια ομάδα Ν τιμών που αντιστοιχούν στο φάσμα του σήματος. Επειδή η είσοδος που προέρχεται από φωνή έχει μόνο πραγματικό μέρος (πραγματική ακολουθία) το *X(k)* είναι συζυγής του *X(N-k)*. Τα μέτρα των συζυγών είναι ίσα οπότε προκύπτει ότι το μέτρο του φάσματος από τα Ν/2 σημεία και άνω θα είναι συμμετρικό με τα πρώτα Ν/2 σημεία. Αυτό σημαίνει ότι αρκεί να απεικονίζουμε τα πρώτα Ν/2 σημεία του φάσματος αφού αυτά περιέχουν όλη την απαιτούμενη πληροφορία του σήματος για να το χαρακτηρίσουν.

Ο αντίστροφος μετασχηματισμός του Φουριέ είναι:

$$x(n)=\frac{1}{N}\sum_{k=0}^{N-1}X(k)\,e^{j2\pi nk/N} \qquad \text{(Εξ. 1.11)}$$

Αν δεν χρησιμοποιηθούν όλα τα δείγματα ο αντίστροφος μετασχηματισμός δεν θα δώσει τις φανταστικές τιμές του αρχικού σήματος.

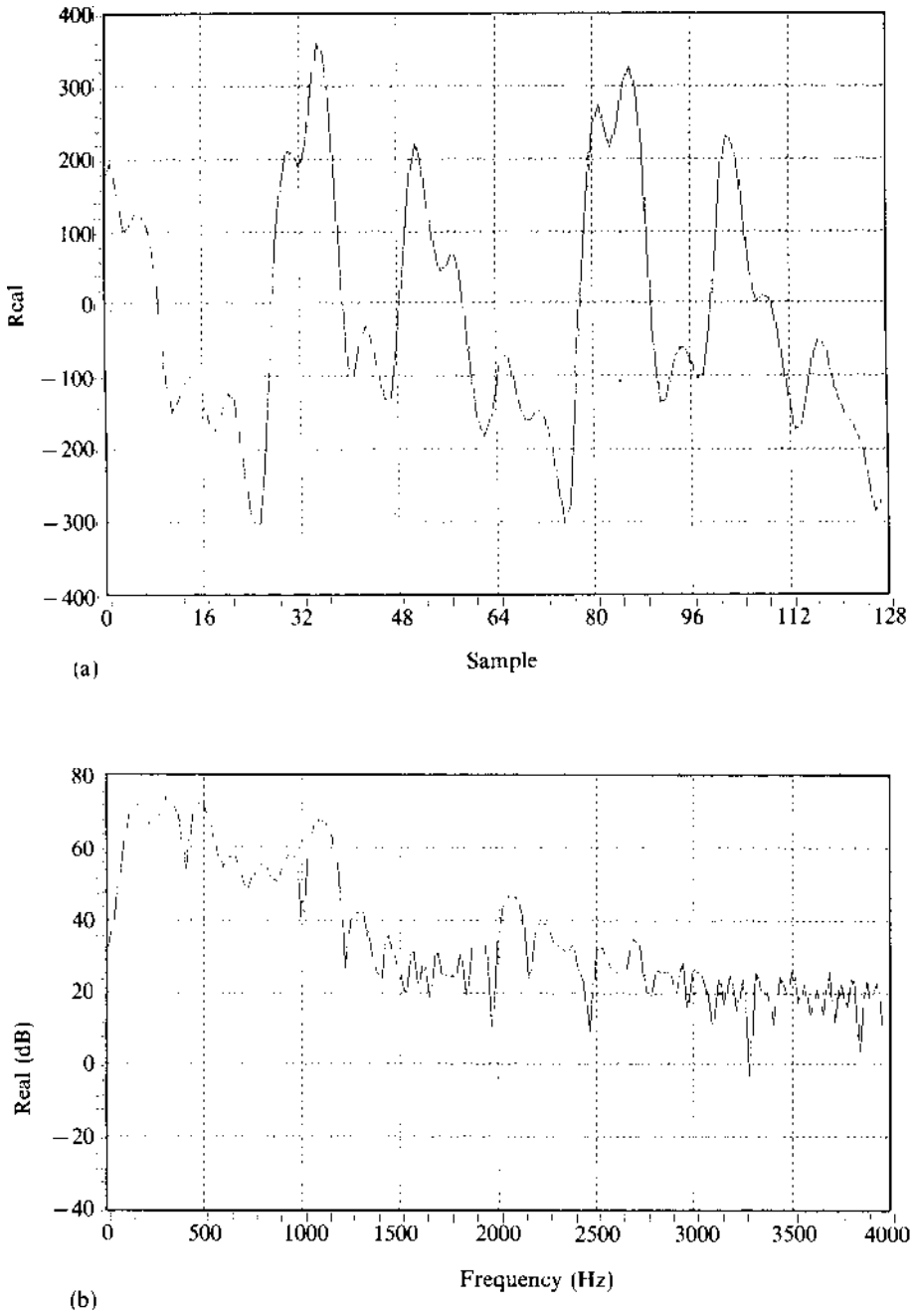
Η ανάλυση στο πεδίο των συχνοτήτων του μετασχηματισμού εξαρτάται από τη συχνότητα δειγματοληψίας και τον αριθμό των δειγμάτων:

$$T=\frac{1}{f_s}$$
$$\Omega=\frac{1}{NT}\ (\text{Hz})\ \text{ή}\ \frac{2\pi}{NT}\ \left(\text{rad}/\text{sec}\right)$$

Στην εφαρμογή στην ομιλία υπάρχουν δυσκολίες στη χρήση λόγω του ότι η περιοδικοτήτα του σήματος μεταβάλλεται με το χρόνο και μπορεί δυο συνεχόμενες ομάδες δειγμάτων φωνής να μην έχουν τις ίδιες συχνότητες , και το πλάτος της κάθε συχνότητας να είναι διαφορετικό.

Για να ξεπεραστούν τα παραπάνω προβλήματα χρησιμοποιούνται διάφορες τεχνικές εκ των οποίων οι πιο συνηθισμένες είναι η εύρεση του βέλτιστου μήκους της ομαδοποιήσης, και η χρήση των παραθύρων για να αποφευχθούν τα φαινόμενα της διαρροής των συχνοτήτων. Για να ρυθμιστούν τα παράθυρα να έχουν ίδιο μήκος μπορούν να προστεθούν μηδενικές τιμές στο τέλος της ομάδας των δειγμάτων. Αυτό δεν αλλάζει τίποτα στο φάσμα σε μορφή , αν η ομάδα των δειγμάτων τελειώνει σε μηδέν, αλλά επειδή αυξάνεται ο αριθμός των δειγμάτων έχουμε μεγαλύτερη ανάλυση στο πεδίο των συχνοτήτων.

Αν παρατηρήσει κανείς το φάσμα ενός σήματος φωνής όπως το παρακάτω, μπορεί να εντοπίσει την βασική συχνότητα του σήματος όπως φαίνεται στο σχήμα 1.18.



Σχ. 1.18. *Τμήμα λέξης και η ανάλυσή του κατά Fourier.*

Στο δείγμα αυτό ο βασικός λοβός είναι στα 1050Hz, ο δευτερεύων στα 2030Hz, και λιγότερο καθαρά, οι επαναλήψεις ανά 160Hz σε όλο το φάσμα. Οι επαναλήψεις αυτές είναι η βασική συχνότητα της φωνής που είναι στα 160Hz. Επίσης διακρίνονται οι συχνότητες συντονισμού του φωνητικού συστήματος που φαίνονται ως αρμονικές της βασικής συχνότητας. Η πρώτη αρμονική έχει ευρύ φάσμα και είναι στα 320 Hz. Η

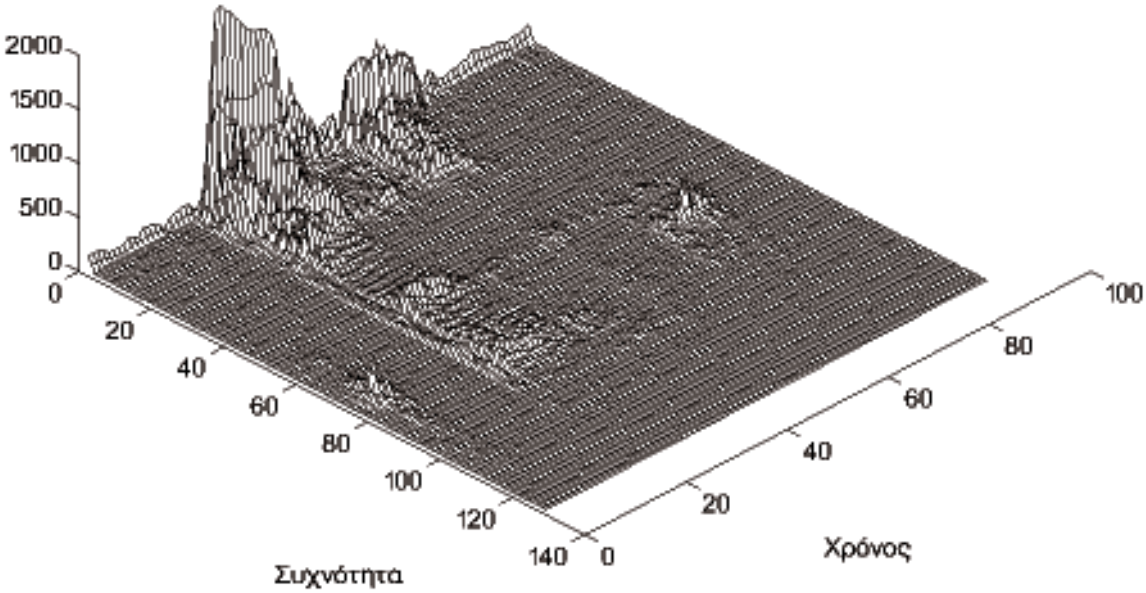
δεύτερη είναι στενότερη σε εύρος και είναι στα 1050 Hz. Τέλος η τρίτη αρμονική βρίσκεται στα 2030 Hz. Ανάμεσα στα 3 με 4 KHz η περιοδικότητα αλλάζει. Η αλλαγή αυτή οφείλεται στη συνάρτηση παραθύρου από τους δευτερεύοντες λοβούς του παραθύρου του Hamming.

Η φωνή έχει το χαρακτηριστικό οτι το πλάτος του σήματος στα σύμφωνα μειώνεται, με αποτέλεσμα να μειώνεται η ακρίβεια στις υψηλές συχνότητες (αφού τα σύμφωνα που τις περιέχουν έχουν χαμηλή ένταση). Για το λόγο αυτό χρησιμοποιούνται φίλτρα που ενισχύουν τις υψηλές συχνότητες ώστε να υπάρχει μια πιο επίπεδη απόκριση στο φάσμα. Τα φίλτρα αυτά καλούνται φίλτρα προέμφασης και δεν είναι τίποτε άλλο από ένα μηδενικό χαμηλών συχνοτήτων:

$$H_{preemph}(z) = 1 = az^{-1}, \quad 0.95 < a < 0.98 \qquad \text{(Εξ. 1.12)}$$

Είναι γεγονός ότι ο διάκριτος μετασχηματισμός του Φουριέ απαιτεί πολλές πράξεις και είναι απελπιστικά αργός σε εφαρμογές πραγματικού χρόνου. Για αυτό το λόγο χρησιμοποιείται στη πράξη μια παραλλαγή του, ο Ταχύς Μετασχηματισμός Φουριέ, γνωστός και ως FFT. Ο μετασχηματισμός αυτός εξοικονομεί μεγάλο ποσό πράξεων (πολλαπλασιασμών και προσθέσεων) βασισμένος στη συμμετρία του DFT. Η χρήση της συμμετρίας έχει ως αποτέλεσμα ο FFT να απαιτεί την ομαδοποίηση των δειγμάτων σε δυνάμεις του δύο (δηλαδή ομάδες των 2,4,8,16, κοκ δειγμάτων), οπότε πρέπει να ληφθούν τα απαραίτητα μέτρα για την προσθήκη των μηδενικών και τη χρήση παραθύρων.

Μια εξελιγμένη χρήση του FFT είναι η παραγωγή των φασματογραμμάτων. Αυτά είναι τρισδιάστατα γραφήματα ενέργειας ανά συχνότητας και μονάδα χρόνου. Αυτά απεικονίζονται είτε τρισδιάστατα είτε ως δισδιάστατος πίνακας συχνότητας-χρόνου με την ισχύ σε κάθε συχνότητα να απεικονίζεται ως διαφορετικό ύψος ή ένταση του μελανιού ή με διαφορετικό χρώμα όπως φαίνεται στο σχήμα 1.19.



Σχ. 1.19. *Τρισδιάστατο φασματόγραμμα μιας λέξης.*

Τυπική απόκλιση, Αυτοσυσχέτιση (Autocovariance,autocorrelation)

Σε περιοδικά σήματα φωνής, όπως τα φωνήεντα, η ανάλυση στο πεδίο του χρόνου δείχνει ότι υπάρχει κάποια σχέση μεταξύ του κάθε δείγματος και των γειτονικών του. Η τυπική απόκλιση για κάποια χρονική καθυστέρηση k δειγμάτων είναι ο μέσος όρος των γινομένων του κάθε δείγματος $s(n)$ με το αντίστοιχό του k δείγματα πιο πέρα $s(n+k)$. Η άθροιση αυτή πραγματοποιείται σε ένα μεγάλο εύρος δειγμάτων ώστε να καλυφθούν αρκέτες περίοδοι από το σήμα της φωνής. Η τυπική αυτο-απόκλιση είναι:

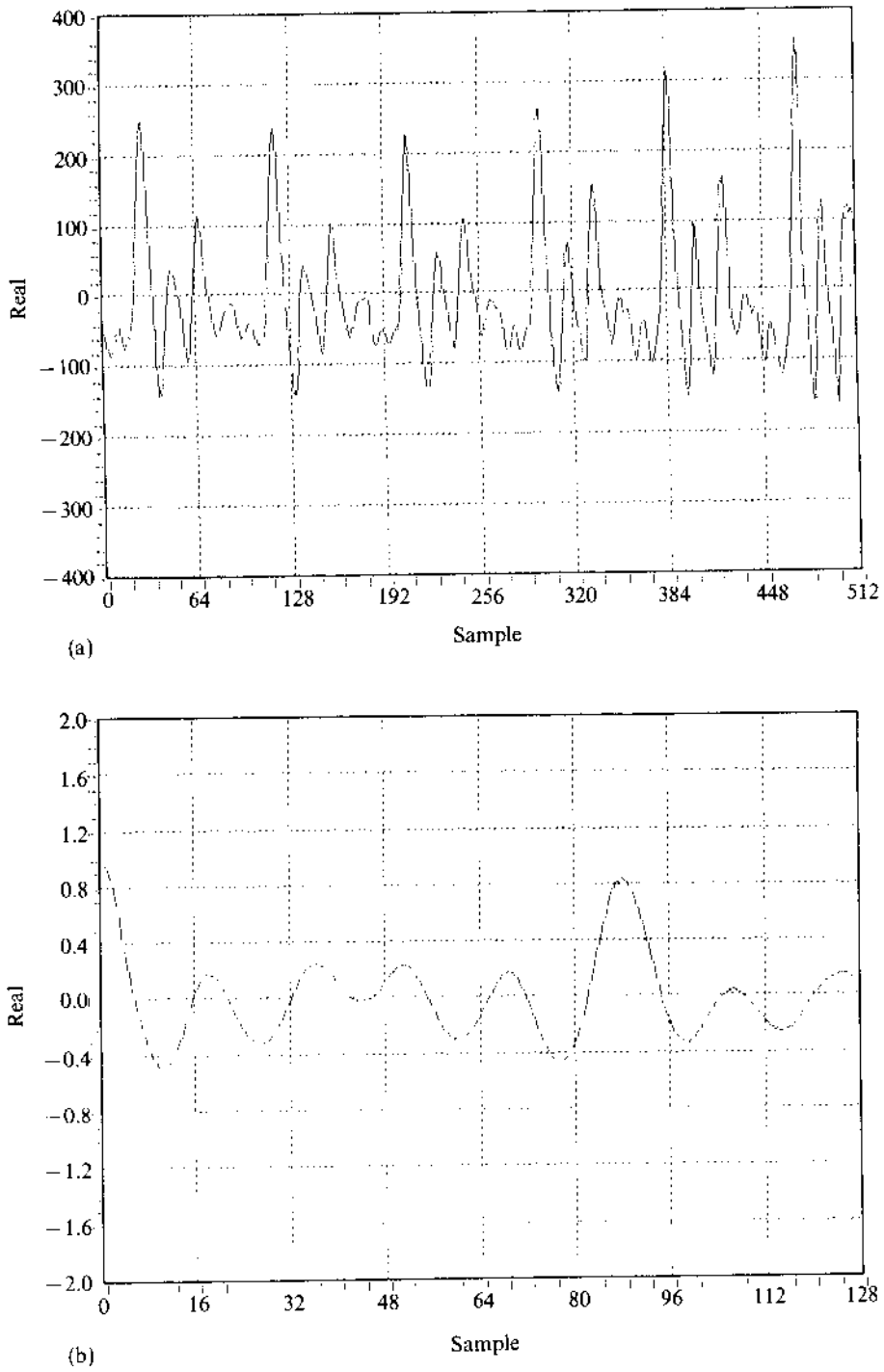
$$C(k) = \frac{1}{N}\sum_{n=0}^{N-1} s(n)s(n+k) \qquad \text{(Εξ. 1.13)}$$

με μήκος ίσο με 0..N-1+k. Επειδή η τιμή του $C(k)$ μπορεί να πάρει ποικίλες τιμές από πολύ μεγάλες μέχρι και πολύ μικρές για την σύγκριση μεταξύ λέξεων χρησιμοποιούμε κανονικοποίηση. Η κανονικοποίηση μας δίνει τη μέγιστη τιμή της όταν η καθυστέρηση k=0 οπότε έχουμε $C(0)$=1. Ο λόγος $R(k)$:

$$R(k) = \frac{C(k)}{C(0)} \qquad \text{(Εξ. 1.14)}$$

καλείται αυτοσυσχέτιση (autocorrelation). Αυτός είναι ο ορισμός της αυτοσυσχέτισης κατά τους Blackman and Tukey (1958) και μπορεί να βρεθεί με διάφορες παραλλαγές.

Μια κυματομορφή φωνής και η αυτοσυσχέτισή της φαίνονται στο σχήμα 1.20.



Σχ. 1.20. *Μιά κυματομορφή και η αυτοσυσχέτισή της.*

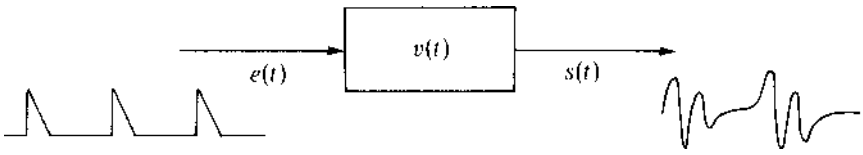
Το σήμα είναι ο φθόγγος /Q/. Η βασική συχνότητα είναι περίπου 90Hz με συχνότητα δειγματοληψίας τα 8KHz. Η αυτοσυσχέτιση δίνει τιμές κοντά στο ένα για πολύ μικρές καθυστερήσεις του k, πράγμα που σημαίνει ότι ένας αλγόριθμος πρόβλεψης μπορεί να προβλέψει την πορεία του σήματος για μικρές χρονικές περιόδους. Επίσης αφού παρουσιάζονται κορυφές με πλάτος κοντά στη μονάδα σε μακρυνές αποστάσεις k_p που αντιστοιχούν στον τόνο της φωνής. Ο αλγόριθμος πρόβλεψης μπορεί να χρησιμοποιήσει και μεγάλες καθυστερήσεις για την πρόβλεψη του σήματος. Το πρόβλημα όμως είναι ότι αυτές οι καθυστερήσεις δεν είναι σταθερές και αλλάζουν με τον τόνο της φωνής οπότε πρέπει να υπάρχει προσαρμογή (adaptive system).

Ενεργειακή Πυκνότητα φάσματος

Αν στη συνάρτηση της αυτοσυσχέτισης εκτελέσουμε μια ανάλυση κατά Φουριέ τότε παίρνουμε την πυκνότητα της ισχύς φάσματος. Η ισχύς του φάσματος είναι χρήσιμη διότι είναι μια πραγματική συνάρτηση της συχνότητας, και δείχνει πιο φανερά τα χαρακτηριστικά του σήματος σε σχέση με το πλάτος του φάσματος.

Cepstrum

Η φωνή αποτελείται από μια διέργεση που φιλτράρεται από τη στοματική κοιλότητα. Στο πεδίο του χρόνου αυτό σημαίνει συνέλιξη. Αν η είσοδος (διέργεση) είναι e(t), και η στοματική κοιλότητα έχει συνάρτηση μεταφοράς u(t) τότε το μοντέλο της ανθρώπινης ομιλίας φαίνεται στο παρακάτω σχήμα 1.21.



Σχ. 1.21. *Το μοντέλο παραγωγής της φωνής.*

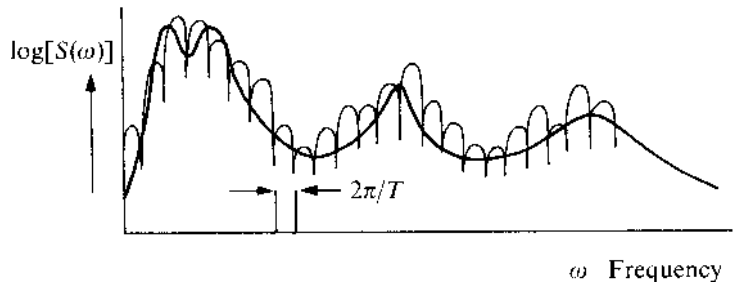
$$s(t) = e(t) * v(t) \qquad \text{(Εξ. 1.15)}$$

Οπότε στο πεδίο των συχνοτήτων η συνέλιξη είναι πολλαπλασιασμός:

$$S(\omega) = E(\omega) \cdot V(\omega) \qquad \text{(Εξ. 1.16)}$$

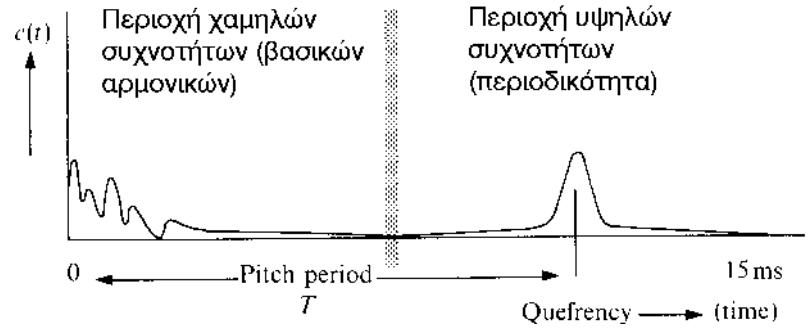
όπου τα S,E,U είναι οι μετασχηματισμοί κατά Φουριέ των s,e,v.

Τα χαρακτηριστικά της διέργεσης και της στοματικής κοιλότητας έχουν πολλαπλασιαστεί και είναι δύσκολο να διαχωριστούν. Αν όμως πάρουμε το λογάριθμο του S τότε ο πολλαπλασιασμός έχει μετατραπεί σε πρόσθεση δυο σημάτων όπως φαίνεται στο σχήμα 1.22.



Σχ. 1.22. *Ο λογάριθμος του φάσματος ενός τμήματος λέξης.*

Αν στο λογάριθμο αυτό εκτελέσουμε μια ανάλυση κατά Φουριέ, παίρνουμε το cepstrum του σήματος. Το cepstrum έχει διαστάσεις χρόνου. Στο διάγραμμα του cepstrum (με άξονα quefrency, "αντίστροφο" της συχνότητας) είναι φανεροί οι τόνοι που προέρχονται από τη διέργεση στα πρώτα ms ενώ ο τόνος της φωνής εμφανίζεται ως μια απότομη κορυφή δεξιότερα από τη μέση όπως και στο σχήμα 1.23.



Σχ. 1.23. *Το Cepstrum ενός τμήματος λέξης.*

Αυτό σημαίνει ότι μπορούμε να φιλτράρουμε με μια συνάρτηση παραθύρου το cepstrum και να κάνουμε τον αντίστροφο μετασχηματισμό Φουριέ ώστε να έχουμε απομονώσει τη συνάρτηση μεταφοράς του στόματος. Από εκεί μπορούν να υπολογιστούν η συχνότητα, το πλάτος και το εύρος των βασικών αρμονικών.

Μέθοδοι Αναγνώρισης Φωνής

Η αναγνώριση φωνής είναι η διαδικασία που προσπαθεί να εξομοιώσει στον υπολογιστή την ακουστική αντιληπτική ικανότητα του ανθρώπου. Ήδη από την περιγραφή της διαδικασίας της αναγνώρισης, καταλαβαίνουμε ότι πρόκειται για κάτι εξαιρετικά πολύπλοκο, αφού καλείται να προσεγγίσει μια από τις πολύπλοκες διαδικασίες της ανθρώπινης φύσης. Η διαδικασία της ακουστικής αντίληψης στον άνθρωπο, χρησιμοποιεί πολλά από τα μέρη του ανθρώπου, τόσο τα σωματικά όργανα (λάρυγγα, στόμα, πνεύμονες, νευρικό σύστημα) όσο και τα λογικά (μνήμη, αντίληψη, συσχετίσεις μέ άλλες αισθήσεις, κατανόηση). Άν σκεφτούμε ότι ακόμα δέν έχουμε κατανοήσει πλήρως τις διαδικασίες αυτές όπως συμβαίνουν στον άνθρωπο, γίνεται εύκολα αντιληπτό ότι δέν είναι δυνατόν, και ίσως δέν πρόκειται να γίνει, η πλήρης προσομοίωση της ικανότητας αυτής του ανθρώπου.

Λόγω του ενδιαφέροντος που έχει η αναγνώριση φωνής, έχει γίνει αντικείμενο μελέτης από πολλά χρόνια, ακόμη και πρίν την διάδοση των ηλεκτρονικών υπολογιστών. Από την εποχή του Α' Παγκοσμίου πολέμου, οι επιστήμονες είχαν ξεκινήσει να καταγράφουν τα χαρακτηριστικά της ανθρώπινης ομιλίας, όπως αυτά περιγράφονται στο προηγούμενο κεφάλαιο, και να προσπαθούν να ερμηνεύσουν με το χέρι τα πρώτα φασματογράμματα, συλλεγμένα από πολλούς ομιλητές, σε λέξεις. Μετά, στην δεκαετία του '70, υπήρχαν αρκετοί επιστήμονες που ξεκίνησαν να επεκτείνουν την εργασία αυτή, έχοντας όμως σαν δυνατό εργαλείο τους τον ηλεκτρονικό υπολογιστή. Τα πρώτα επιτυχή αποτελέσματα ήρθαν γύρω στο 1978, όπου μπορούσαμε να έχουμε αναγνώριση κάποιας λέξης από τον υπολογιστή.

Βέβαια, οι στόχοι που έχουν τεθεί για την ιδανική περίπτωση αλγορίθμου αναγνώρισης, είναι να αναγνωρίζονται άπειρες λέξεις από οποιονδήποτε ομιλητή, σε συνεχή ροή λόγου και σε πραγματικό χρόνο. Αυτό άλλωστε, είχε φανεί και στα κινηματογραφικά έργα επιστημονικής φαντασίας, όπου το διαστημόπλοιο έπαιρνε τις εντολές για τον χειρισμό του από το στόμα του κυβερνήτη. Άν και υπάρχουν πολλά χρόνια μελέτης στο θέμα, σήμερα μπορούμε να έχουμε αλγορίθμους που λειτουργούν με αρκετά μεγάλο ποσοστό επιτυχίας (πάνω από 85%), με διάφορες μεθόδους. Μερικοί από τους πιό προχωρημένους αλγόριθμους λειτουργούν με αρκετά μεγάλη βάση δεδομένων (μερικές χιλιάδες γνωστές λέξεις) και αρκετά μεγάλες ομάδες διαφορετικών χρηστών (μερικές δεκάδες). Υπάρχουν φυσικά και ενδιάμεσοι τρόποι λειτουργίας, με βάση τους οποίους ταξινομούνται οι αλγόριθμοι αναγνώρισης, όπως θα δούμε στην συνέχεια, μετά από μια σύντομη ματιά στα προβλήματα που παρουσιάζονται στην διαδικασία της αναγνώρισης.

Προβλήματα Στην Αναγνώριση

Στο προηγούμενο κεφάλαιο είδαμε ότι η δομή της ανθρώπινης ομιλίας είναι σχετικά απλή, με μερικές δεκάδες φθόγγους να συνθέτουν όλες τις λέξεις σε μια γλώσσα. Οι διάφοροι φθόγγοι παράγονται με κατάλληλες κινήσεις των πνευμόνων, του λάρυγγα, και την κατάλληλη άρθρωση του στόματος και την βοήθεια της ρινικής κοιλότητος. Όμως, το πρόβλημα είναι ότι υπάρχουν τεράστιες διαφορές στον τρόπο που ακούγονται αυτοί οι φθόγγοι, κυρίως λόγω των διαφορών που υπάρχουν στα φωνητικά όργανα ανάμεσα στους ανθρώπους.

Μεταξύ των διαφορών που παρατηρούνται είναι ότι η ανδρική φωνή είναι πιό "μπάσσα", δηλαδή έχει πολλή ενέργεια στις χαμηλές συχνότητες, ενώ η γυναικεία έχει μεγάλη ενέργεια στις υψηλότερες συχνότητες. Επίσης, ανάλογα με την ταχύτητα με την οποία ομιλούμε, διάφοροι φθόγγοι αλλάζουν σε διάρκεια, ακόμη καί άν έχουν εκφωνηθεί από τον ίδιο ομιλητή, ή μπορεί και να παραλειφθούν εντελώς. Ακόμα, όταν μιλάμε, συνήθως έχουμε προετοιμάσει στο νού μας και τις επόμενες λέξεις που θα πούμε. Έτσι, όταν προφέρουμε ένα φθόγγο, ήδη τα φωνητικά όργανα έχουν αρχίσει να προετοιμάζονται να προφέρουν τον επόμενο. Αυτό κάνει δύσκολο τον διαχωρισμό των φθόγγων, αφού υπάρχει όχι μόνο επικάλυψη, αλλά καί εξάρτηση των φθόγγων μεταξύ τους.

Αυτή η επικάλυψη, είναι ένα από τα πιό δύσκολα, ίσως το δυσκολότερο, από τα προβλήματα που πρέπει να αντιμετωπίσουμε στην αναγνώριση φωνής. Το πρόβλημα γίνεται πιό έντονο, όταν παρατηρήσουμε ότι στην ομιλία δέν υπάρχει σαφής διαχωρισμός ανάμεσα στις συλλαβές, αλλά ούτε καί στις λέξεις. Ακόμη και η ακριβής εύρεση του σημείου που αρχίζει και του σημείου που τελειώνει χρονικά μια ομιλιθείσα λέξη, είναι μια διαδικασία δύσκολη και περιέχει συνήθως μεγάλο σφάλμα. Η δυσκολία αυξάνεται από τους θορύβους του περιβάλλοντος, γιατί σε αντίθεση με τον άνθρωπο, ο υπολογιστής δέν έχει την ικανότητα να συγκεντρώσει την προσοχή του σε κάποιον ομιλητή. Έτσι, ενώ ο άνθρωπος μπορεί να αντιληφθεί την ομιλία ακόμη και σε πολύ θορυβώδη περιβάλλοντα (πχ βιομηχανικούς χώρους, πάρτυ κλπ), απορρίπτοντας τους θορύβους του περιβάλλοντος και προσαρμοζόμενος σε αυτούς, ο υπολογιστής βρίσκεται χαμένος σε έναν ωκεανό παρασίτων με το S/N να πλησιάζει το μηδέν από τα αρνητικά.

Άλλο πρόβλημα που παρουσιάζεται είναι η διαφορά των φθόγγων που οφείλεται στην ομιλία διαφόρων διαλέκτων της ίδιας γλώσσας. Μερικοί φθόγγοι μπορεί να λείπουν ή να έχουν αντικατασταθεί από άλλους. Επίσης, διαφορές προκύπτουν και από την προσωδία, δηλαδή τον τρόπο με τον οποίο τονίζουμε τους φθόγγους μιας λέξης για να εκφράσουμε μυνήματα όπως απορία ή ερώτηση. Επιπλέον, κατά την εισαγωγή των σημάτων στον υπολογιστή, έχουμε διάφορους τυχαίους θορύβους, όπως από το ανοιγόκλεισμα των χειλιών, το πλατάγισμα της γλώσσας και τον θόρυβο από την κυκλοφορία του αέρα για την αναπνοή ή την εκφώνηση των φθόγγων.

Μιά άλλη μεγάλη δυσκολία, είναι ότι πολλές λέξεις ακούγονται το ίδιο, αλλά δέν είναι ίδιες, όπως για παράδειγμα οι λέξεις "πολύ" και "πολλοί". Εμείς, έχουμε συνηθίσει να τοποθετούμε την κάθε λέξη μέσα στην γλωσσική περίοδό της και με την βοήθεια του συντακτικού να καταλαβαίνουμε το σωστό νόημα. Μάλιστα πολλές φορες είμαστε σε θέση να προβλέψουμε μερικές φορές την λέξη που θα ακολουθήσει κάποια άλλη, έχοντας την γνώση του συντακτικού και της γραμματικής.

Είδη Αλγορίθμων

Για να υλοποιηθεί η αναγνώριση, πρέπει να γίνουν κάποιοι συμβιβασμοί ως προς το ιδανικό σύστημα. Έτσι, ανάλογα με τους περιορισμούς που δεχόμαστε στα διάφορα μέρη του συστήματος, δημιουργούνται διάφορες κατηγορίες αλγορίθμων. Μπορεί ο αλγόριθμος να λειτουργεί για έναν χρήστη, ή να καταλαβαίνει λίγες μόνο λέξεις ή να χρειάζεται να υπαγορεύουμε τις λέξεις αντί να τις ομιλούμε κανονικά, ή ακόμα ο χρήστης να πρέπει να χρησιμοποιεί και μια ιδιότυπη γλώσσα με πολύ απλό συντακτικό και περιορισμένο λεξιλόγιο, που θα τα γνωρίζει ο υπολογιστής.

Έτσι λοιπόν, δημιουργείται ένας τρόπος με τον οποίο μπορούν να ταξινομηθούν οι διάφοροι αλγόριθμοι. Οι συνηθισμένες κατηγορίες και τα κριτήρια με τα οποία ταξινομούνται, είναι:

Με βάση τον τρόπο ομιλίας
    Σε ρυθμό φυσικής ομιλίας (συνεχούς)
    Σε ρυθμό υπαγόρευσης λέξεων
Με βάση το είδος της βάσης δεδομένων
    Για ένα χρήστη
    Για απεριόριστους χρήστες
Με βάση τον αλγόριθμο
    Με επεξεργασία φθόγγων
        Με φωνητική ανάλυση
        Με φασματική ανάλυση
    Με μοντελοποίηση
        Hidden Markov Models
        Χρήση LPC
    Με τεχνητή νοημοσύνη
        Νευρωνικά δίκτυα
        Ασαφής λογική

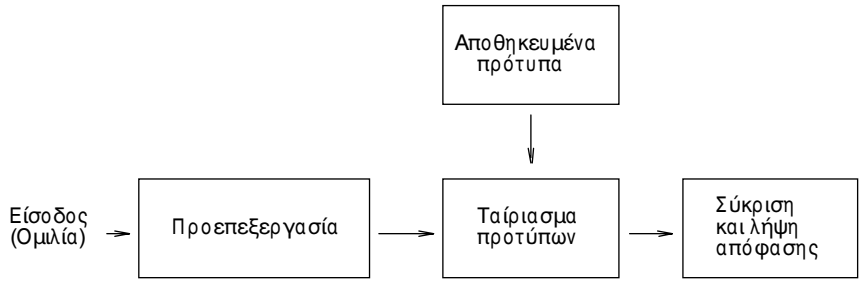
Τα συστήματα που χρησιμοποιούν σάν είσοδο την φυσική ομιλία του ανθρώπου, συνήθως απαιτούν μεγάλη υπολογιστική ισχύ και χρησιμοποιούν μεθόδους σχετικές με στοχαστικά μοντέλα της ανθρώπινης ομιλίας. Συνήθως υποστηρίζονται και από γλωσσικά στοιχεία, όπως η γραμματική και το συντακτικό. Επίσης, έχουν κάπως περιορισμένο λεξιλόγιο, πράγμα που είναι άμεση συνέπεια της έλλειψης ικανοποιητικής υπολογιστικής ισχύος σε ευρεία κλίμακα και σε χαμηλή τιμή. Τα συστήματα που δέχονται την είσοδό τους με ρυθμό υπαγόρευσης (μια λέξη κάθε φορά) μπορούν να επιτύχουν μεγαλύτερα ποσοστά επιτυχίας, γιατί είναι σε θέση να βρούν με μεγαλύτερη ακρίβεια την αρχή και το τέλος της λέξης, και να μπορέσουν έτσι να εφαρμόσουν καλύτερα τους αλγορίθμους δυναμικής παραμόρφωσης στο πεδίο του χρόνου (dynamic time warping, DTW), όπως θα δούμε και στην συνέχεια.

Οι αλγόριθμοι μπορεί να είναι βελτιστοποιημένοι είτε για έναν, είτε για πολλούς χρήστες. Άν ο αλγόριθμος είναι σχεδιασμένος για έναν χρήστη, τότε η βάση δεδομένων μπορεί να εκπαιδευτεί καί να προσαρμοστεί στα χαρακτηριστικά της φωνής του συγκεκριμένου ομιλητή. Αυτό αυξάνει τα ποσοστά επιτυχίας, αφού οι διακυμάνσεις στην προφορά μιάς συγκεκριμένης λέξης από τον ίδιο ομιλητή είναι σχετικά μικρές. Στα συστήματα που προορίζονται να χρησιμοποιηθούν από πολλούς χρήστες, στην βάση δεδομένων αποθηκεύεται ένας μέσος όρος γνωρισμάτων για κάθε λέξη, αφού υπάρχουν μεγάλες διακυμάνσεις από τον ένα ομιλητή στον επόμενο. Έτσι, τα πράγματα δυσκολεύουν, και για ικανοποιητικά αποτελέσματα απαιτείται να χρησιμοποιηθεί κάποιος στοχαστικός αλγόριθμος βασισμένος στην μοντελοποίηση της ανθρώπινης ομιλίας, ή να περιοριστεί πολυ το λεξιλόγιο σε δύο ή τρείς δεκάδες λέξεις που θα είναι εύκολα διακριτές μεταξύ τους.

Στην συνέχεια θα κάνουμε μια επισκόπηση στις αλγοριθμικές παραλλαγές που αφορούν την προσέγγιση την οποία χρησιμοποιούμε, δηλαδή τον αναλυτικό τρόπο, που χρησιμοποιεί τεχνικές επεξεργασίας σήματος, και τον στοχαστικό τρόπο, που χρησιμοποιεί μοντελοποίηση των χαρακτηριστικών της φωνής σάν μια ψευδοτυχαία ακολουθία, και την χρήση τεχνητής νοημοσύνης.

Φωνητική Ανάλυση

Με την προσέγγιση αυτή, η κάθε λέξη που εισάγεται στον υπολογιστή αποθηκεύεται σαν μια σειρά δειγμάτων που σχηματίζουν ένα πρότυπο. Σκοπός του αλγορίθμου είναι να συγκρίνει το πρότυπο που εισάγεται από το χρήστη, με τα πρότυπα της βάσης δεδομένων του, και να δώσει σάν έξοδο την αναπαράσταση του προτύπου της βιβλιοθήκης που πλησιάζει περισσότερο την λέξη που είπε ο ομιλητής. Στα συστήματα αυτά, άν προορίζονται για έναν χρήστη, πρέπει να γίνει εκπαίδευση, δηλαδή απομνημόνευση των προτύπων της βιβλιοθήκης στον υπολογιστή. Για να γίνει αυτό, κατά την διάρκεια της διαδικασίας εκπαίδευσης, ο χρήστης προφέρει την κάθε λέξη που θα περιλαμβάνεται στο λεξιλόγιο του υπολογιστή, και το αποτύπωμά της αποθηκεύεται στην μνήμη του υπολογιστή. Άν το σύστημα προορίζεται να χρησιμοποιηθεί από πολλούς χρήστες, τότε αποθηκεύονται από πρίν τα αποτυπώματα των λέξεων, που αυτή την φορά είναι ο μέσος όρος από τις λέξεις που έχουν προφέρει διάφοροι ομιλητές, με διαφορετική ηλικία, φύλο και διάλεκτο. Συνήθως, σε αυτά τα συστήματα δέν υπάρχει διαδικασία εκπαίδευσης, μιας και κάτι τέτοιο δέν είναι δυνατόν. Επίσης, υπάρχουν μερικά συστήματα, τα οποία προσαρμόζουν συνεχώς την βάση δεδομένων τους στον χρήστη τους. Τα συστήματα αυτά βελτιώνονται με την χρήση τους και μπορεί να έχουν αρκετά υψηλότερα ποσοστά επιτυχίας.



Σχ. 2.1. *Η δομή ενός συστήματος αναγνώρισης.*

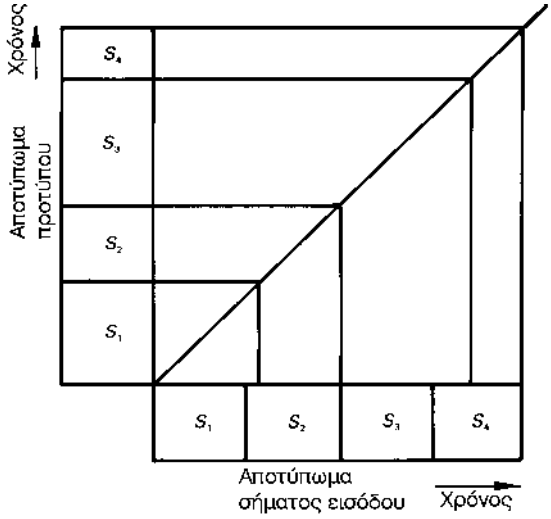
Το λεξιλόγιο αυτών των συστημάτων είναι πολύ περιορισμένο και δέν υπερβαίνει τις 100 λέξεις. Κύρια αιτία για τον περιορισμό αυτό, είναι ότι απαιτείται αρκετά μεγάλη ισχύς για να έχουμε γρήγορη και αποτελεσματική σύγκριση όλων των αποτυπωμάτων της βιβλιοθήκης, μέχρι να βρεθεί το σωστό ταίρι. Τα προγράμματα αναγνώρισης που χρησιμοποιούν αυτό τον αλγόριθμο, στηρίζουν ένα μεγάλο μέρος της επιτυχίας τους στην μέθοδο της Δυναμικής Παραμόρφωσης (Dynamic Time Warping - DTW), για να αντισταθμίσουν τις αποκλίσεις που υπάρχουν στην ταχύτητα ομιλίας και την προφορά των φθόγγων. Αξίζει λοιπόν να δούμε με λίγη λεπτομέρεια το θέμα αυτό.

Δυναμική Παραμόρφωση

Όταν πρέπει να συγκρίνουμε δύο διαφορετικές σειρές δειγμάτων, είναι περισσότερο από βέβαιο ότι θα υπάρχουν διαφορές στο μήκος τόσο της συνολικής ακολουθίας, όσο και στα επιμέρους τμήματά της, σε σχέση πάντα με το πρότυπό της. Διαφορές είναι σχεδόν σίγουρο ότι θα υπάρχουν και στις απόλυτες τιμές του πλάτους της κυματομορφής της λέξης. Για να μπορέσουμε να συγκρίνουμε τις δύο ακολουθιες, πρέπει πρώτα να βρούμε ένα μέτρο σύγκρισης. Από τα πιό απλά είναι μια Ευκλείδια απόσταση μεταξύ των δειγμάτων της ακολουθίας, δηλαδή ένα άθροισμα της μορφής

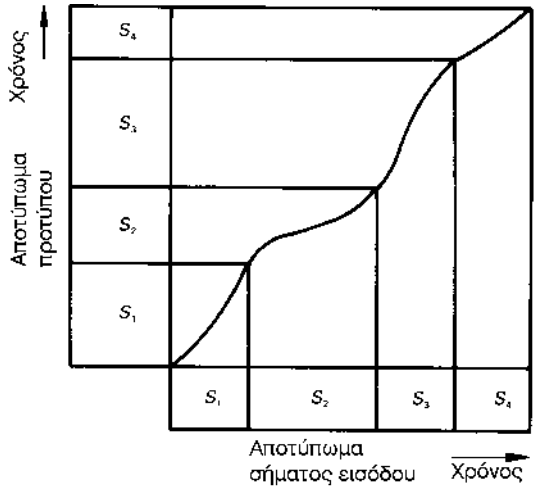
$$A = \sum \left(x_i - y_i \right)^2 \qquad \text{(Εξ. 2.1)}$$

αλλά υπάρχουν και άλλες περιπτώσεις, όπως θα δούμε και στην συνέχεια. Όπως είναι προφανές, για να γίνει η σύγκριση, πρέπει πρώτα να έχουμε δύο ακολουθίες με το ίδιο μήκος. Στην περίπτωση της ομιλίας, δύο λέξεις του ίδιου ομιλητή (πχ το αποτύπωμα στην βιβλιοθήκη στου συστήματος και η ομιληθείσα λέξη) όχι μόνο δεν θα ταιριάζουν, αλλά θα διαφέρουν τόσο στην συνολική τους διάρκεια, όσο και στην διάρκεια των επιμέρους φθόγγων τους. Μια πρώτη προσέγγιση, είναι να παραμορφώσουμε γραμμικά τα μήκη των ακολουθιών με βάση την αρχή και το τέλος τους, όπως φαίνεται στο σχήμα 2.2.



Σχ. 2.2. *Απλή γραμμική παραμόρφωση.*

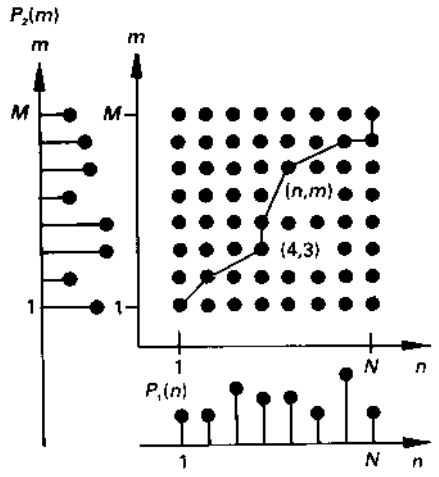
Βέβαια, αυτή η μέθοδος δέν είναι ικανοποιητική, επειδή δέν συγκλίνει στα ενδιάμεσα τμήματα της ακολουθίας. Όπως φαίνεται και στο σχήμα, τα ενδιάμεσα σημεία διαφέρουν. έστω και άν συμπίπτουν η αρχή και το τέλος. Άρα, αυτό που χρειάζεται είναι ένας τρόπος να αλλάξουν και τα μήκη των επιμέρους τμημάτων της λέξης, ώστε να έχουμε πλήρη ταύτιση. Αυτή η παραμόρφωση δέν θα είναι γραμμική, αφού το κάθε τμήμα της λέξης θα υποστεί διαφορεικό βαθμό παραμόρφωσης. Ήδη λοιπόν έχουμε φτάσει στην μή-γραμμική δυναμική παραμόρφωση (DTW) που φαίνεται σχηματικά στο σχ 2.3.



Σχ. 2.3. *Μή γραμμική παραμόρφωση.*

Ανάλυση

Έστω ότι έχουμε δύο ακολουθίες, την $P_1(n)$ και την $P_2(m)$, με μήκος 1..N και 1..M, αντίστοιχα. Άν θέλουμε να χρησιμοποιήσουμε την δυναμική παραμόρφωση, πρέπει να είμαστε σίγουροι ότι έχουμε εντοπίσει σωστά την αρχή και το τέλος της κάθε ακολουθίας, πράγμα που επηρεάζει την τιμή του N και του M. Με δεδομένο ότι αυτό έχει γίνει, πρέπει να βρούμε ένα τρόπο να αντιστοιχίσουμε τα τμήματα της $P_1(n)$ στην $P_2(m)$, δηλαδή μια καμπύλη υπο την μορφή μιας τρίτης ακολουθίας $\omega()$ που θα συνδέσει τους δείκτες n με τους δείκτες m, πχ $m=\omega(n)$. Το "μήκος" της $\omega(n)$ μπορεί να μας δώσει μια μέτρηση ταύτισης ή όχι του προτύπου της βιβλιοθήκης με το αποτύπωμα που εισάγεται για έλεγχο στον υπολογιστή, και είναι ανώτερο από την απλή Ευκλείδια απόσταση, ιδίως όταν κάνουμε συγκρίσεις πολυδιάστατων προτύπων (πχ εικόνες δύο διαστάσεων, επιφάνειες τριών διαστάσεων κλπ). Έχοντας ταιριάξει την αρχή και το τέλος των P_1 και P_2, ξέρουμε ήδη το πρώτο και το τελευταίο σημείο της $\omega(n)$, δηλαδή $\omega(1)=1$ και $\omega(N)=M$ (βλ. σχ. 2.4)



Σχ. 2.4. *Γραφική παράσταση της σύγκρισης της P_1 και της P_2*

Η $\omega(n)$ είναι ο δρόμος που αντιστοιχίζει τα $P_1(n)$ στα $P_2(m)$ ώστε να έχουμε την ελάχιστη διαφορά μεταξύ τους. Άρα, το πρόβλημα εύρεσης της $\omega(n)$, είναι πρόβλημα βελτιστοποίησης συνάρτησης σφάλματος. Η συνάρτηση σφάλματος θα είναι η συνολική απόσταση των δύο ακολουθιών P_1 και P_2, δηλαδή

$$\sum_{n=1}^{N}\left[d\left(P_1(n),\, P_2\left(\omega(n)\right)\right)\right] \qquad \text{(Εξ. 2.2)}$$

με d να είναι η συνάρτηση που δίνει την απόσταση μεταξύ των δύο στοιχείων, ενός από την P_1 και ενός από την P_2. Άρα λοιπόν, σκοπός είναι να ελαχιστοποιήσουμε αυτό το άθροισμα, βρίσκοντας μια λύση για την $\omega(n)$. Πρέπει λοιπόν να λύσουμε το πρόβλημα ελαχιστοποίησης

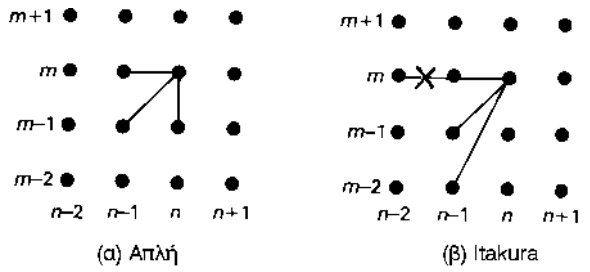
$$D^* = \min\left\{\sum_{n=1}^{N}\left[d\left(P_1(n),P_2\left(\omega(n)\right)\right)\right]\right\} \qquad \text{(Εξ. 2.3)}$$

δηλαδή για ποιά $\omega(n)$ το D γίνεται ελάχιστο.

Για να λυθεί το πρόβλημα χρησιμοποιούμε έναν αλγόριθμο ο οποίος εξελίσσεται σύμφωνα με τα δεδομένα της κάθε στιγμής και την πορεία που πρέπει να ακολουθήσει. Αυτός ο τρόπος προγραμματισμού (dynamic programming) είναι βασισμένος πάνω στην αρχή της μερικής βελτιστοποίησης, σύμφωνα με την οποία: "Από οποιοδήποτε σημείο και άν ξεκινήσει η διαδικασία βελτιστοποίησης, τα εναπομένοντα βήματα προς την επίτευξη του στόχου πρέπει να είναι και αυτά βέλτιστα". Για εμάς, αυτό σημαίνει ότι: άν για να φτάσουμε από το πρώτο σημείο της $\omega(n)$ (1,1) στο τυχαίο (n,m) πρέπει να περάσουμε από το πχ (4,3), τότε ο βέλτιστος δρόμος από το (1,1) στο (4,3) είναι ένα τμήμα του συνολικού βέλτιστου δρόμου από το (1,1) στο (n,m). Με άλλα λόγια, ο ολικός βέλτιστος δρόμος μπορεί να βρεθεί επιλέγοντας τις τοπικές βέλτιστες διαδρομές από το ένα σημείο στο επόμενο.

Βεβαίως, είναι απαραίτητο να θέσουμε και κάποιους περιορισμούς στην εύρεση της $\omega(n)$. Για παράδειγμα, δέν πρέπει να επιτρέψουμε να συμβεί υπερβολική συστολή ή διαστολή των τμημάτων της ακολουθίας που προσπαθούμε να ταιριάξουμε. Άλλος περιορισμός είναι ότι η ω(n) πρέπει να είναι αύξουσα, δηλαδή να μήν επιστρέφει πίσω στις τιμές των δεικτών, αλλά να δίνει όλο και αυξανόμενα αποτελέσματα (αλλιώς πώς θα είναι ο συντομώτερος δρόμος, και ταυτόχρονα θα έχει πλήρη ταύτιση;).

Αυτοί οι περιορισμοί εφαρμόζονται στα μικρότερα τμήματα που απαρτίζουν την ολική διαδρομή που υποδεικνύουν τα σημεία της $\omega(n)$. Για παράδειγμα, στα επόμενα σχήματα φαίνονται μερικοί περιορισμοί εφαρμοζόμενοι σε μια μεγέθυνση του σχήματος 2.4. Στο πρώτο σχήμα, υπάρχει περιορισμός στα τμήματα από τα οποία μπορούμε να μεταβούμε στο σημείο (n,m). Επιτρέπονται μόνο οι μεταβάσεις από τα σημεία εκείνα που θα κάνουν την μορφή της $\omega(n)$ να είναι αύξουσα. Στο διπλανό σχήμα, υπάρχει ένα διαφορετικό μονοπάτι μέσω του οποίου μπορούμε να φτάσουμε στο (n,m) σε σχέση με το πρώτο σχήμα, και επιπλέον ένας περιορισμός (σημειώνεται με X) που απαγορεύει την μετάβαση στο (n,m) από ένα σημείο που θα κάνει την $\omega(n)$ να είναι επίπεδη, δηλαδή χωρίς αύξηση, για περισσότερα από 2 σημεία.



Σχ. 2.5 Παραλλαγές του αλγορίθμου παραμόρφωσης (λεπτομέρεια).

Άν θέλουμε να υπολογίσουμε την ελάχιστη απόσταση που πρέπει να διανύσουμε από το σημείο (1,1) του σχήματος 2.4, έως το σημείο (*n,m*), με τους περιορισμούς που φαίνονται στο σχήμα 2.5, τότε μπορούμε να χρησιμοποιήσουμε ένα αναδρομικό μοντέλο υπολογισμού με βάση την διανυθείσα απόσταση. Άν συμβολίσουμε με $DA(n,m)$ την βέλτιστη (ελάχιστη) απόσταση μέχρι το σημείο (*n,m*), τότε μπορούμε να γράψουμε ότι

$$DA(n,m) = d\left[P_1(n), P_2(m)\right] + \\ + \min\left\{DA(n,m-1), DA(n-1,m-1), DA(n-1,m)\right\} \qquad \text{(Εξ. 2.4)}$$

δηλαδή ότι η ελάχιστη απόσταση μέχρι το σημείο (*n,m*) είναι η τοπική απόσταση των σημείων $P_1(n)$ και $P_2(m)$ σύν την ελάχιστη από τις αποστάσεις από την αρχή μέχρι τα 3 γειτονικά του σημεία που φαίνονται στο σχήμα. Ο υπολογισμός μπορεί να ξεκινήσει από το σημείο (1,1) και να προχωρήσει για όλους τους συνδυασμούς των *n* και *m*. Η ολική διαδρομή θα βρεθεί όταν υπολογιστεί η τιμή του (N,M). Άν ξεκινήσουμε τον υπολογισμό από το τέλος, δηλαδή το σημείο (N,M), τότε είναι δυνατόν να υπολογιστεί ακριβώς και η ακολουθία $\omega(n)$, αλλά συνήθως αυτό δέν χρειάζεται στην αναγνώριση φωνής, μιας και μας ενδιαφέρει μόνο το "μήκος" της $\omega(n)$, που χρησιμεύει σαν μέτρο σύγκρισης για την επίτευξη της αναγνώρισης.

Παράδειγμα

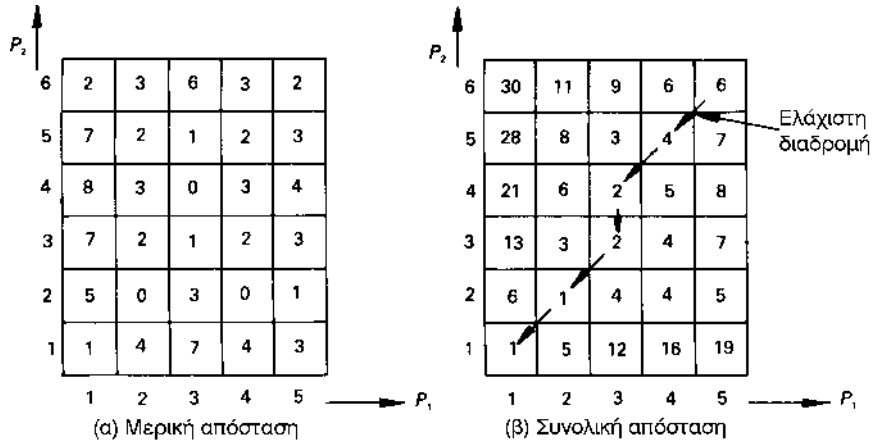
Η υλοποίηση του αλγορίθμου μπορεί να γίνει πιό ξεκάθαρη με την χρήση ενός παραδείγματος. Έστω λοιπόν ότι έχουμε τις ακολουθίες P_1={1,6,9,6,5} και P_2={2,6,8,9,8,3}, και έστω ότι σαν απόσταση θα χρησιμοποιήσουμε την συνάρτηση που μας δίνει την διαφορά του πλάτους των αντίστοιχων σημείων της P_1 και της P_2:

$$D\left(P_1,P_2\right) = \sum_i \left|P_1(i) - P_2(i)\right| \qquad \text{(Εξ. 2.5)}$$

Το πρώτο βήμα είναι να σχηματίσουμε ένα πίνακα με τις διαφορές των συνδυασμών σημείων που σχηματίζεται από τις στήλες και τις γραμμες του πίνακα (βλ. σχ. 2.6). Δηλαδή στην γραμμή 3 και στήλη 2 του πίνακα μπαίνει η διαφορά (με βάση τον παραπάνω τύπο) του σημείου $P_1(2)$ και $P_2(3)$, δηλαδή |8-6|=2.

Στην συνέχεια, περνάμε στο επόμενο βήμα, δημιουργώντας τον συνολικό πίνακα αποστάσεων, χρησιμοποιώντας την εξίσωση 2.4, που υπολογίζει την συνολική απόσταση του κάθε σημείου από την αρχή του πίνακα. Για παράδειγμα, η τιμή στην θέση (3,3) είναι το άθροισμα της απόστασης των δύο ακολουθιών στο 3ο σημείο (από τον πρώτο πίνακα) και της ελάχιστης των αποστάσεων του από τα σημεία (3,2), (2,2) και (2,3), δηλαδή είναι 1+min(4,1,3)=2. Η συνολική ελάχιστη (βέλτιστη) απόσταση

μεταξύ των δύο ακολουθιών είναι η τιμή που βρίσκεται στην πάνω δεξιά θέση του πίνακα, δηλαδή το 6. Από αυτό τον πίνακα μπορούμε να βρούμε και την βέλτιστη διαδρομή, δηλαδή τις τιμές της $\omega(n)$. Για να το κάνουμε αυτό, ακολουθούμε τις τιμές του πίνακα από το τέλος προς την αρχή, διαλέγοντας σαν την επόμενη διαδρομή αυτήν που έχει την μικρότερη τιμή (απόσταση), και βρίσκεται στα 3 γειτονικά σημεία από το δικό μας. Στον πίνακα φαίνεται η φορά με την οποία διασχίζουμε τον πίνακα για να βρούμε τον βέλτιστο δρόμο, που είναι η $\omega(n)$.



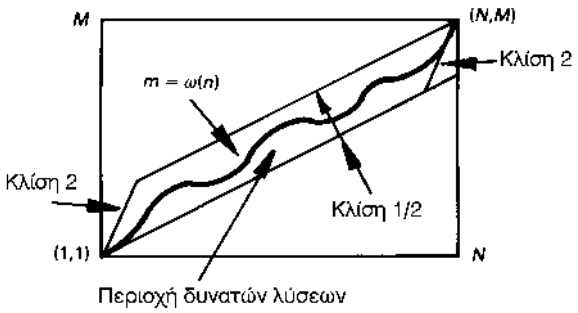
***Σχ. 2.6** Πίνακες υπολογισμού της ελαχίστης απόστασης.*

Οι τιμές της $\omega(n)$ είναι οι εξής:

n	m=ω(n)
1	1
2	2
3	3,4
4	5
5	6

***Σχ. 2.7.** Οι τιμές της ω(n) μετά το ταίριασμα.*

Παρατηρήστε ότι στην εύρεση της βέλτιστης ταύτισης των δύο ακολουθιών, χρειάστηκε να αντιστοιχισθούν δύο σημεία της P_2 στο σημείο 3 της P_1 (τα 3 και 4). Αυτό είναι άμεσο αποτέλεσμα των λιτών περιορισμών που επιβάλλαμε στις τοπικές διαδρομές ανάμεσα στα σημεία. Με τους περιορισμούς που φαίνονται στο σχ 2.5β, ο Itakura σχεδίασε έναν βελτιωμένο αλγόριθμο για την αναγνώριση φωνής, που έχει μερικά ενδιαφέροντα χαρακτηριστικά. Με τον περιορισμό ότι η $\omega(n)$ δέν μπορεί να έχει πάνω από 2 συνεχόμενα σημεία στα οποία να αντιστοιχίζεται ένα σημείο της ακολουθίας εισόδου, δημιουργείται ένα παραλληλόγραμμο επιτρεπτών τιμών (βλ. σχ. 2.8), εντός του οποίου θα πρέπει απαραιτήτως να περιλαβάνεται η βέλτιστη διαδρομή. Το παραλληλόγραμο αυτό αποτελείται από τμήματα που έχουν κλίση 2 και 0.5 μονάδες. Οι κλίσεις αυτές προκύπτουν από τους περιορισμούς ότι η βέλτιστη διαδρομή δέν μπορεί, όπως είπαμε, να είναι επίπεδη για παραπάνω από 2 σημεία (κλίση 0.5), και ότι η διαδρομή προς το σημείο (n,m), δέν μπορεί να έρθει από κάποιο σημείο χαμηλότερο από το $(n\text{-}1,m\text{-}2)$.

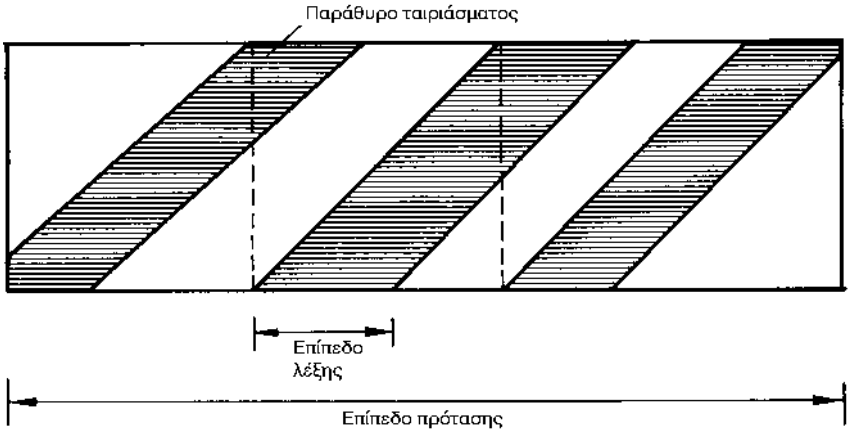


Σχ. 2.8. *Οι περιορισμοί του αλγορίθμου Itakura.*

Αυτοί οι περιορισμοί είναι πολύ σημαντικοί, γιατί περιορίζουν σημαντικά τον αριθμό των σημείων που πρέπει να υπολογιστούν, μειώνοντας έτσι τον απαιτούμενο χώρο και χρόνο για την εφαρμογή του DTW, που είναι μια απαιτητική διαδικασία, τόσο σε μνήμη, όσο και σε υπολογιστική ισχύ, ιδίως όταν πρόκειται να χρησιμοποιηθεί σε μεγάλα λεξιλόγια. Οι περισσότερες υλοποιήσεις αναγνώρισης, έχουν κάποιους περιορισμούς με τους οποίους προσπαθούν να αποφύγουν να υπολογίσουν όσο το δυνατόν περισσότερα σημεία, άν φαίνονται ότι δέν θα βοηθήσουν στην ανεύρεση του βέλτιστου δρόμου.

Εφαρμογές σε συνεχή ομιλία

Επειδή η ομιλία με μεγάλες παύσεις ανάμεσα στις λέξεις είναι αφύσικη, έχουν αναπτυχθεί μερικές εφαρμογές του DTW για την αναγνώριση λέξεων που ομιλούνται σε φυσιολογικό ρυθμό. Με μια πρώτη ματιά φαίνεται ότι ίσως είναι δυνατόν να διαχωριστούν οι λέξεις και στην συνέχεια να αναγνωριστούν ξεχωριστά η κάθε μιά. Αυτό όμως δέν γίνεται, γιατί υπάρχει όπως είπαμε και πιο πρίν, αλληλοεξάρτηση μεταξύ των λέξεων που προφέρουμε. Έτσι, ο μοναδικός δρόμος είναι να εφαρμόσουμε ένα "διπλό" DTW σε ολόκληρη την φράση του σήματος εισόδου. Το πρώτο επίπεδο (βλ. σχ. 2.9) προσπαθεί να αναγνωρίσει μορφές που μοιάζουν με λέξεις, και το δεύτερο επίπεδο προσπαθεί να σχηματίσει την απεικόνιση όλης της πρότασης, με βάση τις γνώσεις του στην γραμματική και το λεξιλόγιό του.

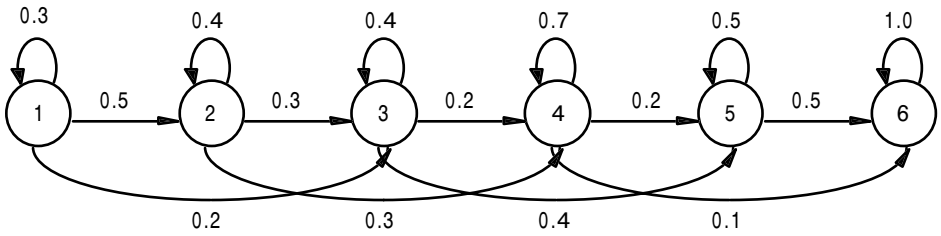


Σχ. 2.9. *Δυναμική παραμόρφωση για την αναγνώριση προτάσεων.*

Η απόσταση των ζωνών αναγνώρισης στο πρώτο επίπεδο αφήνει αρκετά περιθώρια για ρυθμίσεις και παραμόρφωση, ώστε να μπορούν να ταιριάξουν οι λέξεις, οι οποίες έχουν αλλοιωμένη αρχή και τέλος, με τα πρότυπα στην βιβλιοθήκη. Μια εξέλιξη του αλγορίθμου αυτού μπορεί και συγκρίνει διαδοχικά πολυπλοκότερες εισόδους (φράσεις) με την ομιληθείσα είσοδο, μέχρι να βρεθεί το κατάλληλο ταίριασμα. Οι μέθοδοι αυτές είναι αρκετά μνημοβόρες και απαιτητικές από πλευράς ισχύος. Γι' αυτό το λόγο, οι προσπάθειες για την επίτευξη της αναγνώρισης σε αυτό το επίπεδο στρέφονται σε μεθόδους όπως τα μοντέλα Markov, που θα δούμε στην συνέχεια.

Μοντέλα Markov

Τα κρυφά Μοντέλα Markov (Hidden Markov Models, HMMs) είναι μοντέλα με τα οποία μπορούμε να μοντελοποιήσουμε μια ακολουθία σήματος που μεταβάλλεται με το χρόνο, με μια έξοδο στοχαστικής ή τυχαίας διαδικασίας. Ένα HMM αποτελείται από μιά αλυσίδα Markov, όπως φαίνεται και στο σχήμα 2.10. Ο κάθε ένας από τους έξι κύκλους αποτελεί μια μεταβλητή κατάστασης του μοντέλου μια καθορισμένη χρονική στιγμή, και δίνει σαν έξοδο μια "παρατήρηση" ή "μέτρηση". Την επόμενη χρονική στιγμή, η κάθε μεταβλητή, ανάλογα με το σήμα εισόδου, μπορεί να παραμείνει στην ίδια κατάσταση, ή να πάει σε άλλη, οπότε αλλάζει και η έξοδος που μας δίνει. Καταγράφοντας τις εξόδους των μεταβλητών κατάστασης μέχρι το σήμα να περάσει ολόκληρο μέσα από το HMM, έχουμε το αποτύπωμα του σήματος εισόδου.



Σχ. 2.10. *Μια αλυσίδα μοντέλου Markov.*

Οι μεταβάσεις από την μια κατάσταση (*i*) στην επόμενη (*j*), εξαρτώνται από τις πιθανότητες μετάβασης $\{a_{ij}\}$. Επίσης, σε κάθε κατάσταση, το άθροισμα των πιθανοτήτων να μείνει η κάθε μεταβλητή στην παρούσα κατασταση ή να περάσει σε κάποια επόμενη, είναι ίσο με ένα. Σε κάθε κατάσταση του μοντέλου, η παραγωγή των παρατηρήσεων εξόδου που βγαίνουν από το μοντέλο (έχοντας σαν είσοδο μια επιλεγμένη ακολουθία από ένα σύνολο *M* ακολουθιών - σημάτων εισόδου), εξαρτάται από ένα άλλο σέτ πιθανοτήτων, το b_{jk} (βλ. και σχήμα 2.12). Η πιθανότητα b_{jk} είναι η πιθανότητα να παραχθεί ένα σύνολο παρατηρήσεων της μορφής *k* όταν το μοντέλο βρεθεί στην κατάσταση *j*. Οι αρχικές καταστάσεις των μεταβλητών είναι αβέβαιες, και ορίζονται με ένα τρίτο σετ πιθανοτήτων, το π_j, που δείχνει τις πιθανότητες που έχει κάθε μεταβλητή να βρεθεί στην αρχική κατάσταση *j* όταν t=0. Είναι προφανές, ότι το άθροισμα των πιθανοτήτων του συνόλου π για όλες τις μεταβλητές πρέπει να είναι ίσο με την μονάδα. Στο σχήμα 2.13, οι δύο μόνες πιθανές αρχικές καταστάσεις είναι η πρώτη και η δεύτερη, με 0.5 η κάθε μια (δηλαδή ίσες πιθανότητες και για τις δύο), που φυσικά έχουν άθροισμα ίσο με ένα.

$$A = [a_{ij}] = \begin{bmatrix} 0.3 & 0.5 & 0.2 & 0 & 0 & 0 \\ 0 & 0.4 & 0.3 & 0.3 & 0 & 0 \\ 0 & 0 & 0.4 & 0.2 & 0.4 & 0 \\ 0 & 0 & 0 & 0.7 & 0.2 & 0.1 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 1.0 \end{bmatrix}$$

Σχ. 2.11. *Ο πίνακας μετάβασης που περιγράφει την αλυσίδα του σχήματος 2.10*

$$B = [b_{jk}] = \begin{bmatrix} b_{11} & b_{21} & b_{31} & b_{41} & b_{51} & b_{61} \\ b_{21} & b_{22} & b_{32} & b_{42} & b_{52} & b_{62} \\ b_{1k} & b_{2k} & b_{3k} & b_{4k} & b_{5k} & b_{6k} \\ b_{1M} & b_{2M} & b_{3M} & b_{4M} & b_{5M} & b_{6M} \end{bmatrix}$$

Σχ. 2.12. *Πίνακας πιθανοτήτων εξόδου για κάθε κατάσταση του μοντέλου.*

$$\pi = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Σχ. 2.13. *Πίνακας πιθανοτήτων αρχικής κατάστασης.*

Το μοντέλο λέγεται "κρυφό", γιατί η σειρά των καταστάσεων από τις οποίες πέρασε το μοντέλο δέν μπορεί να βρεθεί. Μια άλλη διαφορά στα μοντέλα που χρησιμοποιούνται στην αναγνώριση από την γενική περίπτωση, είναι ότι στην αναγνώριση υπάρχει μια διαδοχή των πιθανών καταστάσεων από τα αριστερά προς τα δεξιά. Στην γενική περιπτωση ενός μοντέλου, αυτό δέν συμβαίνει, και μπορούμε να μεταφερθούμε από μια κατάσταση σε μια οποιαδήποτε άλλη, ακόμη και προς τα πίσω. Αυτή η διαφοροποίηση γίνεται για να εφαρμόσουμε καλύτερα τα μοντέλα στην μορφή της ανθρώπινης ομιλίας.

Ένα Παράδειγμα

Για να καταλάβουμε πώς λειτουργεί το μοντέλο στην παραγωγή "παρατηρήσεων" για την ανθρώπινη ομιλία, ας δούμε πρώτα ένα ανάλογο παράδειγμα. Ας υποθέσουμε ότι έχουμε έξι δοχεία, με το κάθε δοχείο να περιέχει ψηφίδες διαφόρων χρωμάτων. Υπάρχουν 10 διαφορετικά χρώματα, και κάθε δοχείο περιέχει διαφορετικό πλήθος ψηφίδων από κάθε χρώμα. Ρίχνουμε ένα κέρμα για να αποφασίσουμε από ποιό δοχείο θα ξεκινήσουμε (πχ κορώνα=δοχείο 1 και γράμματα=δοχείο 2). Ξεκινάμε λοιπόν από το δοχείο που έτυχε να δείξει το κέρμα, και παίρνουμε μία ψηφίδα από αυτό το δοχείο. Σημειώνουμε ο χρώμα της και μετά την ξανατοποθετούμε πίσω στην θέση της. Μετά ρίχνουμε ένα ζάρι για να δούμε με ποιό δοχείο θα συνεχίσουμε. Η διαδικασία επαναλαμβάνεται για το νέο δοχείο και συνεχίζεται μέχρι να έχουμε συλλέξει στις σημειώσεις μας, ας πούμε, πενήντα χρώματα (παρατηρήσεις).

Η ανάλογη διαδικασία που συμβαίνει στα μοντέλα Markov, έχει ως εξής. Τα δοχεία αντιστοιχούν στις μεταβλητές κατάστασης και η σειρά με την οποία εξελίσσονται από το αποτέλεσμα της ζαριάς. Η αρχική κατάσταση που βρίσκεται με το ρίξιμο του κέρματος, αντιστοιχεί στο σέτ πιθανοτήτων π_j={0.5, 0.5, 0, 0, 0, 0}, μιας και το κέρμα δίνει ίσες πιθανότητες να έρθει κάποια από τις δύο όψεις του. Μετά από το αρχικό στάδιο, η επόμενη κατάσταση εξαρτάται από το αποτέλεσμα του ζαριού, δηλαδή από το σέτ a_{ij}, που είναι ένας πίνακας 6x6 και δείχνει τις πιθανότητες να μεταβούμε από το δοχείο που βρισκόμαστε τώρα σε κάποιο άλλο. Η πιθανότητα b_{jk} να κάνουμε την παρατήρηση k ενώ είμαστε στην κατάσταση j (δηλαδή να τραβήξουμε την ψηφίδα χρώματος k από το δοχείο j), ποικίλει από δοχείο σε δοχείο, αφού κάθε ένα περιέχει διαφορετικές ποσότητες ψηφίδων από το κάθε χρώμα. Άρα το σέτ b_{jk} θα έχει δύο διαστάσεις, αφού εξαρτάται όχι μόνο από το δοχείο στο οποίο είμαστε, αλλά και από τον αριθμό των ψηφίδων που υπάρχουν από κάθε χρώμα. Έτσι, το b_{jk} θα έχει διαστάσεις 6x10. Οι σημειώσεις που κρατήσαμε κατά την εξέλιξη του πειράματος, είναι οι παρατηρήσεις, και κάθε σημείωση είναι η έξοδος του μοντέλου για ένα τμήμα του σήματος εισόδου.

Μεταφέροντας λοιπόν το παράδειγμα στην παραγωγή ομιλίας, καταλαβαίνουμε ότι η ομιλία παράγεται από την σύνθεση μερικών απλών φθόγγων. Άν παρομοιάσουμε τον κάθε φθόγγο με μια μεταβλητή κατάστασης του μοντέλου, τότε κάθε λέξη είναι μια ακολουθία καταστάσεων, με διαφορετικές και ποικίλες διάρκειες από τον ένα στον επόμενο. Άρα, οι μεταβάσεις μεταξύ των διαφόρων καταστάσεων μπορούν να αναπαρασταθούν από πιθανότητες (a_{ij}). Οι παρατηρήσεις που εξάγονται από το μοντέλο είναι οι φθόγγοι που περιέχονται στον ήχο που παράγεται κάθε στιγμή. Επειδή ο ήχος έχει πολλές διακυμάνσεις, οι παρατηρήσεις αυτές μπορούν να απεικονιστούν πιο καλά με ενα σύνολο πιθανοτήτων bjk.

Για την αναγνώριση, χρειαζόμαστε ένα μοντέλο για κάθε λέξη της βάσης δεδομένων. Για να γίνει η παραγωγή του μοντέλου, πρέπει ρυθμίσουμε τα δύο σέτ πιθανοτήτων με κάποια επαναληπτική διαδικασία, έτσι ώστε να μεγιστοποιήσουμε την περίπτωση του να παράγεται η ακολουθία εισόδου που χρησιμοποιήσαμε για την εκπαίδευση του μοντέλου, από το μοντέλο αυτό.

Για να γίνει ικανοποιητικά η εκπαίδευση του μοντέλου, απαιτούνται μεγάλες ποσότητες δεδομένων. Για συστήματα με ένα χρήστη, απαιτείται αρκετά μεγάλος αριθμός επαναλήψεων της κάθε λέξης της βάσης δεδομένων από τον χρήστη. Για ένα σύστημα με πολλούς χρήστες, απαιτείται μεγάλος αριθμός επαναλήψεων από πολλούς χαρακτηριστικούς ομιλητές. Η διαδικασία της αναγνώρισης υπολογίζει την πιθανότητα να παραχθεί μια λέξη που να μοιάζει με την λέξη που είπε ο ομιλητής, χρησιμοποιώντας το κάθε μοντέλο της βιβλιοθήκης. Το μοντέλο που παράγει την έξοδο που βρίσκεται κοντύτερα στην ομιληθείσα λέξη, είναι και το αποτέλεσμα της αναγνώρισης, δηλαδή η ομιληθείσα λέξη. Η ποσότητα των υπολογισμών που απαιτούνται κατα την φάση της αναγνώρισης είναι κατα πολύ μικρότερη από αυτήν που απαιτείται κατά την διάρκεια της εκπαίδευσης. Το μεγαλύτερο λοιπόν πρόβλημα είναι η εκπαίδευση των μοντέλων.

Παρατηρήσεις Στην Εφαρμογή της αναγνώρισης

Άν και υπάρχει αρκετά μεγάλη δυσκολία στην εκπαίδευση των μοντέλων Μαρκόβ, όλα δείχνουν ότι θα είναι η μέθοδος που τελικά θα επικρατήσει. Μεγάλο βάρος δίνεται και στην βελτίωση των συστημάτων προεπεξεργασίας του σήματος, μιας και είναι πολύ σημαντικό να έχουμε όχι μόνο ακριβή, αλλά και ξεκάθαρη εικόνα του σήματος εισόδου. Μια ιδιαίτερα ενδιαφέρουσα μελέτη γίνεται στο να εξομοιωθεί η λειτουργία

του ανθρώπινου αυτιού, μιας και έχει βρεθεί ότι αποκρίνεται εξαιρετικά σε θορυβώδη περιβάλλοντα. Οι επιστήμονες που εργάζονται σε αυτό τον τομέα, προσπαθούν να προσεγγίσουν και τα δυναμικά φαινόμενα που συμβαίνουν στο ανθρώπινο αυτί, όπως η επιλογή συχνοτήτων, η προσαρμογή στο θόρυβο και η απομόνωση ορισμένων χαρακτηριστικών της φωνής.

Ένας άλλος τομέας που έχει δώσει ενθαρρυντικά αποτελέσματα και εξελίσσεται πολύ γρήγορα, είναι η χρήση των νευρωνικών δικτύων. Σήμερα, τα νευρωνικά δίκτυα έχουν ξεπεράσει σε επιτυχία τις παλαιότερες μεθόδους, κυρίως στις εφαρμογές που περιλαμβάνουν πολλούς χρήστες ή χρήση σε θορυβώδη περιβάλλοντα. Οι ερευνητές πιστεύουν ότι τελικά θα χρησιμοποιηθεί η μέθοδος των μοντέλων Μαρκόβ για την αρχική επεξεργασία της λέξης, και η αναγνώριση θα γίνεται με την βοήθεια νευρωνικών δικτύων, που θα μπορούν να χρησιμοποιήσουν και άλλες μορφές βάσης δεδομένων, όπως οι γραμματικοί και συντακτικοί κανόνες μας γλώσσας.

Το Αναπτυξιακό Σύστημα του DSP56001

Το πακέτο του ADS περιλαμβάνει τρία τμήματα. Το πρώτο τμήμα είναι το ADM56001 που είναι η κύρια πλακέτα με το DSP56001, μια πλακέτα ελέγχου και επικοινωνίας του ADM με το PC, και το λογισμικό ελέγχου του ADM. Επίσης για την επικοινωνία με τον αναλογικό κόσμο έχει συνδεθεί και ένα αναπτυξιακό A/D-D/A EVB DSP56ADC16.

Το ADM είναι πρακτικά ένα πλήρες σύστημα μικροϋπολογιστή με μνήμη RAM/ROM, DSP επεξεργαστήρα, κυκλώματα αποκωδικοποιήσης, και είσοδο-έξοδο. Όλη η αρχιτεκτονική του συστήματος είναι με 24 bit στο δίαυλο των δεδομένων και 16 bit στο δίαυλο των διευθύνσεων.

Η πλακέτα ελέγχου μπορεί να δεχτεί μέχρι και 8 ADM παράλληλα και δεν είναι τίποτε άλλο από μια κάρτα ψηφιακών εισόδων/εξόδων για την αποστολή και λήψη δεδομένων, προγράμματος και ελέγχου της κάρτας ανάπτυξης.

Το αναπτυξιακό εργαλείο για τους αναλογικοψηφιακούς μετατροπείς αποτελείται από έναν A/D και έναν D/A των 16-Bit με σειριακή μετάδοση δεδομένων, χρονιζόμενους είτε από τον κρυσταλικό ταλαντωτή που βρίσκεται επί της κάρτας είτε από εξωτερικό σήμα.

Το λογισμικό αποτελείται από ένα πρόγραμμα ελέγχου που τρέχει στο ADM από τη ROM, και από το τερματικό πρόγραμμα που τρέχει στο PC. Αυτό προσφέρει στο χρήστη σε κατανοητή σ' αυτόν μορφή, τη δυνατότητα να δώσει όποια εντολή θέλει στο ADM, να σταματήσει το πρόγραμμα, να κάνει άμεση χρήση του συμβολομεταφραστή κ.α.

Επίσης υπάρχει διαθέσιμος και ο off-line συμβολομεταφραστής, C compiler, και ο εξομοιωτής.

Αναλυτική περιγραφή

Η πλακέτα ελέγχου

Αυτή αποτελείται από κάποιους απομονωτές και μανδαλωτές οι οποίοι φέρνουν τα σήματα του διαύλου του PC (θυρίδα XT) στο υπό έλεγχο ADM. Έτσι μεταφέρονται τα δεδομένα από το ADM στο PC και αντίστροφα. Το υλικό υλοποιεί κάποιο πρωτόκολλο επικοινωνίας με τα αντίστοιχα κυκλώματα του ADM. Επειδή υπάρχει η δυνατότητα να ελεγχθούν πολλά ADM από μια κάρτα ελέγχου (μέχρι 8) κάθε ADM έχει μια μοναδική διεύθυνση.

ADM 56001

Το ADM 56001 αποτελείται από τον επεξεργαστή DSP56001 της Motorola, 8KWords μνήμης RAM, 2KWords μνήμης ROM, τα κυκλώματα αποκωδικοποίησης της μνήμης, κυκλώματα ελέγχου του reset και των διακοπών, τα κυκλώματα επικοινωνίας με το PC, όπως επίσης και τους συνδετήρες για τις διάφορες επεκτάσεις.

DSP56001

Ας ρίξουμε μια ματιά στον επεξεργαστή. Ο επεξεργαστής DSP56001 έχει μήκος λέξης 24-bit. Είναι ένας ψηφιακός επεξεργαστής σήματος γενικού σκοπού. Αυτό σημαίνει ότι μπορεί να υλοποιηθεί οποιοσδήποτε αλγόριθμος επεξεργασίας σήματος (όπως φίλτρα IIR και FIR, FFT, CZT κλπ) και να εκτελεστεί σε πραγματικό χρόνο.

Η χρησιμότητα αυτού του τύπου των επεξεργαστών περιορίζεται μόνο από τη φαντασία του χρήστη και την υπολογιστική δύναμη που αυτοί έχουν. Ανάλογα με την εφαρμογή, εκτός από την ταχύτητα της επεξεργασίας ενδιαφέρει και η ακρίβεια στις μαθηματικές πράξεις, ώστε να μειωθούν τα σφάλματα στρογγυλοποιήσης όπως για παράδειγμα στα IIR φίλτρα όπου η ακρίβεια των πράξεων επηρρεάζει συνήθως και την ευστάθεια τους.

Επειδή οι DSP επεξεργαστές χρησιμοποιούνται για γρήγορες εφαρμογές (σε σχέση με άλλους μικρο-επεξεργαστές/υπολογιστές), όπου απαιτείται είτε ταχύτητα υπολογισμών είτε απόκρισης, έχουν ειδικούς μηχανισμούς για τον χειρισμό των διακοπών ώστε να είναι δυνατή η μέγιστη δυνατή ταχύτητα εξυπηρέτησης αν αυτό είναι επιθυμητό.

Όπως σε κάθε μικροεπεξεργαστή έτσι και στους DSP είναι επιθυμητό από προγραμματιστική άποψη το σύνολο των εντολών να είναι συμμετρικό (ορθογώνιο) αν αυτό είναι δυνατό, διότι έτσι το λογισμικό γίνεται πιο εύκολο να γραφεί και γίνεται πιο αποδοτικό.

Επίσης πολλές φορές επειδή οι εφαρμογές στον αυτόματο έλεγχο ποικίλουν (όπως για παράδειγμα ο έλεγχος DC κινητήρα για σερβομηχανισμούς) πολλοί επεξεργαστές διαθέτουν και περιφερειακά κυκλώματα όπως σειριακές επικοινωνίες κ.α.

Σημαντικό ρόλο σε εφαρμογές ελέγχου όπου απαιτείται μικρό μέγεθος πλακέτας είναι οι εσωτερικές μνήμες που διαθέτει ο επεξεργαστής ώστε να εξαλειφθεί η ανάγκη για χρήση εξωτερικών ακριβών εξαρτημάτων όπως για παράδειγμα γρήγορες μνήμες.

Η εσωτερική αρχιτεκτονική παίζει σημαντικό ρόλο στην αύξηση της ταχύτητας επεξεργασίας. Επειδή στους αλγορίθμους επεξεργασίας σήματος υπάρχουν πολλές μεταφορές δεδομένων (πχ. FIR φίλτρο με 110 taps) οι επεξεργαστές αυτοί διαθέτουν αρχιτεκτονική Harvard και όχι την κλασσική Von Neumann. Αυτό σημαίνει ότι υπάρχουν διαφορετικοί δίαυλοι για τη μεταφορά των δεδομένων από τις μνήμες προς τους καταχωρητές, και διαφορετικοί για τη μεταφορά των εντολών του προγράμματος.

Οι καταχωρητές έχουν λειτουργίες που δεν συναντώνται στους συμβατικούς επεξεργαστές, όπως κυκλική αποθήκευση, αύξηση κατά βήματα, προαύξηση κ.α.

Λόγω της μεγάλης αριθμοφαγίας που χαρακτηρίζει τους αλγόριθμους επεξεργασίας σήματος, το υλικό του επεξεργαστή DSP υποστηρίζει επεξεργασία με έντονη διοχέτευση (pipeline) και παραλληλία, με ένα σύνολο εντολών βελτιστοποιημένο για παραγωγή των υπολογισμών.

Τα ειδικότερα χαρακτηριστικά του DSP56001 είναι:

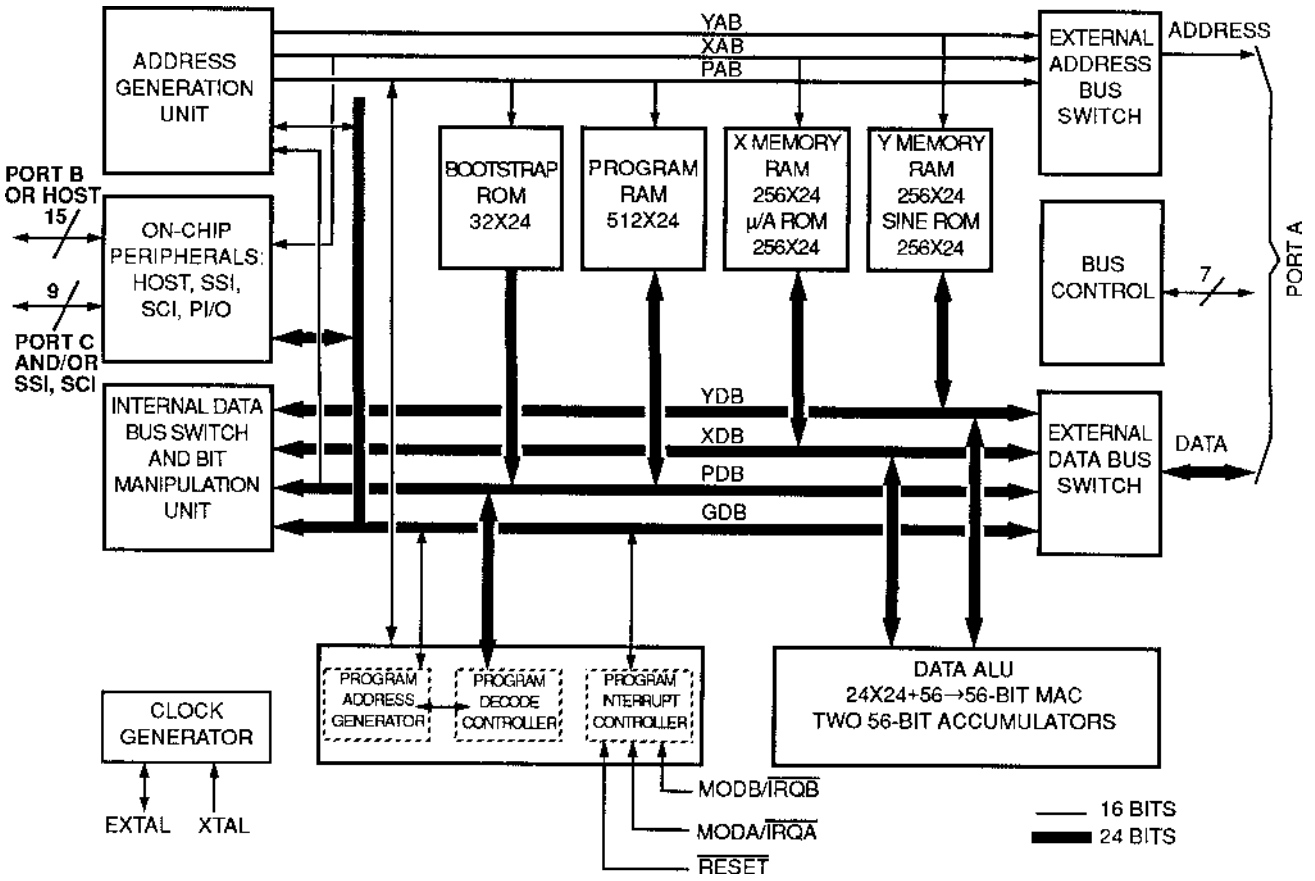
1. 13.5 MIPS στα 27MHz (MIPS: Million Instructions Per Second - εκατομύρια εντολές ανα δευτερόλεπτο)
2. 2 συσσωρευτές των 56-bit.
3. Δέκα καταχωρητές δεδομένων των 24-bit.
4. 24 καταχωρητές διευθύνσεων των 24-bit.
5. Συμμετρικό σύνολο εντολών
6. α. Εντολές βρόγχου με στοίβα υλικού
   β. Ταχεία επεξεργασία διακοπών
7. 4 εσωτερικοί δίαυλοι δεδομένων, 3 εσωτερικοί δίαυλοι διευθύνσεων
8. Περιφερειακά
   α. 8-bit Host port με δυνατότητα προσπέλασης με DMA.
   β. Σειριακή επικοινωνία με γεννήτρια Baud (για RS-232)
   γ. Σύγχρονη σειριακή επικοινωνία με γεννήτρια ρολογιού (για CODEC)
9. Μνήμη επί του ολοκληρωμένου
   α. 2 τράπεζες 256x24 bit μνήμης RAM δεδομένων
   β. 2 τράπεζες 256x24 bit μνήμης ROM (προγραμματισμένων με συντελεστές)
   γ. 512x24 bit RAM μνήμη προγράμματος
10. Εξωτερική μνήμη (μέγιστη χρήση)
    α. μνήμη δεδομένων 128Kx24 bit
    β. μνήμη προγράμματος 64Kx24 bit
    γ. Προγραμματιζόμενοι κύκλοι καθυστέρησης (wait states)
11. Δυναμική περιοχή 114db για τους διαύλους 24 bit, 336db για τους συσσωρευτές.
12. Η παραλληλία στη μονάδα παραγωγής διευθύνσεων, τον ελεγκτή προγράμματος, την αριθμητική και λογική μονάδα είναι τόσο μεγάλη που μπορεί να γίνεται μια διοχέτευση εντολής, ένας πολλαπλασιασμός 24x24 bit, μια πρόσθεση των 56 bit, δύο μεταφορές δεδομένων, και δυο μεταβολές σε καταχωρητές διευθύνσεων σε ένα κύκλο μηχανής. Και όλα αυτά χωρίς να επηρεάζονται άλλες λειτουργίες όπως μεταφορά Full-duplex από τη σειριακή θύρα, και half-duplex από το Host interface.

Η οργάνωση του επεξεργαστή είναι αυτή του σχήματος 3.1

Τα δομικά του στοιχεία είναι η μονάδα παραγωγής των διευθύνσεων, ο ελεγκτής προγράμματος, οι εσωτερικές μνήμες, οι διακόπτες για εσωτερική ή εξωτερική σύνδεση των διαύλων, τα περιφερειακά, η αριθμητική και λογική μονάδα, ο ελεγκτής του εξωτερικού διαύλου, και το ρολόι. Επίσης όπως παρατηρούμε και από το σχήμα υπάρχουν τέσσερις εσωτερικοί δίαυλοι δεδομένων, και τρεις διευθύνσεων.

Η μονάδα παραγωγής διευθύνσεων είναι υπεύθυνη για την διευθυνσιοδότηση της μνήμης προγράμματος, και των δεδομένων ταυτόχρονα. Τα δεδομένα έχουν χωριστεί σε δυο ομάδες τα X και Y. Αυτό σημαίνει ότι μπορούμε να προσπελάσουμε ταυτόχρονα δυο δεδομένα, ένα από κάθε ομάδα. Εξωτερικά το ποιός δίαυλος θα βγεί προς τα έξω (στους ακροδέκτες διευθύνσεων στο περίβλημα του ολοκληρωμένου) καθορίζεται από το διακόπτη, ο οποίος επιλέγει (μέσω του ελεγκτή προγράμματος) το ποιός δίαυλος πρέπει να εμφανιστεί στα εξωτερικά περιφερειακά ή μνήμες.

Η μονάδα εκτέλεσης εντολών (ελεγκτής προγράμματος) ελέγχει τη ροή του προγράμματος, αποκωδικοποιεί τις εντολές, και είναι υπεύθυνη για την εκτέλεση των διακοπών και τον κατάλληλο χειρισμό τους. Ο ελεγκτής προγράμματος ελέγχει όλες τις άλλες υπομονάδες αφού τελικά αυτός είναι ο ελεγκτής του υλικού.



Σχ. 3.1. *Η αρχιτεκτονική του πυρήνα του DSP56000*

Οι εσωτερικές μνήμες παίρνουν τους διαύλους των διευθύνσεων από τη μονάδα παραγωγής διευθύνσεων και ανταλλάσουν δεδομένα μέσω των αντίστοιχων διαύλων τους με την αριθμητική και λογική μονάδα, τον ελεγκτή προγράμματος, και τα περιφερειακά. Το ποιος δίαυλος θα εμφανιστεί στους εξωτερικους ακροδέκτες το καθορίζει ο ελεγτής προγράμματος μέσω του αντίστοιχου μεταγωγικού διακόπτη. Αυτοί οι διακόπτες έχουν τοποθετηθεί για να μην αυξηθεί ο αριθμός των ακροδεκτών στο περίβλημα πράγμα που θα σήμαινε μεγαλύτερο κόστος του ολοκληρωμένου, του τυπωμένου κυκλώματος, την αύξηση των εξωτερικών εξαρτημάτων. Έτσι γίνεται ένας συμβιβασμός κόστους και απόδοσης.

Η αριθμητική και λογική μονάδα είναι το "στομάχι" του επεξεργαστή αφού αυτή αναλαμβάνει όλη την αριθμοφαγία που προκύπτει. Μπορεί να εκτελέσει σε ένα κύκλο μηχανής ένα πολλαπλασιασμό και μια συσσώρευση, να πραγματοποιήσει λογικούς ελέγχους, λογικές πράξεις, πολλαπλασιασμούς, αθροίσεις, και στρογγυλοποίηση. Τα δεδομένα προέρχονται από τις δύο ομάδες δεδομένων X και Y. Έτσι μπορούν να μεταφερθούν ταυτόχρονα δυο δεδομένα πρός την ALU (όπως για παράδειγμα στη πρόσθεση δυο αριθμών) ώστε να μην υπάρχει καθυστέρηση στις μεταφορές των δεδομένων.

Τα εσωτερικά περιφερειακά έχουν διάφορες χρήσεις. Έτσι λοιπόν το SCI (Serial Communications Interface) λόγω του ότι έχει γεννήτρια Baud μπορεί να χρησιμοποιηθεί για ασύγχρονες επικοινωνίες όπως η RS-232. Το SSI (Synchronous Serial Interface) είναι η σύγχρονη σειριακή επικοινωνία και χρησιμοποιείται για τον έλεγχο εξωτερικών περιφερειακών συνήθως με αναλογικοψηφιακούς μετατροπείς CODEC. Ο όρος CODEC σημαίνει COder-DECoder. Η ονομασία αυτή βγαίνει επειδή οι αναλογικοψηφιακοί μετατροπείς κατασκευάζονται σε ένα ολοκληρωμένο κύκλωμα το οποίο

μπορεί να μετατρέψει τα δεδομένα του σε διάφορους τύπους (μ-law, A-law, PCM) με σκοπό τη σύμπίεση των δεδομένων. Η μετατροπή ισχύει και προς τις δυο κατευθύνσεις (από το A/D και προς τον D/A) γι' αυτό και το Code-Decode, κωδικοποιήση από τον A/D και αποκωδικοποιήση από τον D/A. Η επικοινωνία αυτή έχει ένα ρολόι το οποίο ελέγχει το ρυθμό μεταφοράς των δεδομένων και η μεταφορά των δεδομένων γίνεται κατά πλαίσια (Frames).

Το Host Interface χρησιμοποιείται για την εντολοδότηση του DSP επεξεργαστή από άλλο επεξεργαστή. Είναι μια εύκολη διασύνδεση που πραγματοποιείται με ελάχιστα εξωτερικά κυκλώματα. Με την πόρτα αυτή ο κεντρικός επεξεργαστής ενός συστήματος μπορεί να "κατεβάσει" κώδικα στο DSP όπως στη διαδικασία του Bootstrap όπου ο επεξεργαστής μη έχοντας φορτώσει τον κώδικα της εφαρμογής, περιμένει την ακολουθία των εντολών από τη πόρτα αυτή για να την αποθηκεύσει στην εσωτερική του μνήμη και κατόπιν να την εκτελέσει. Έτσι μπορεί ο DSP να μην χρησιμοποιείσει καθόλου εξωτερική μνήμη, και τα προγράμματα που θα εκτελεί να βρίσκονται μαζί με άλλα από άλλους επεξεργαστές σε φτηνές μνήμες. Η ταχύτητα επεξεργασίας παραμένει σταθερή αφού το κομμάτι του κώδικα που πρέπει να εκτελεστεί φορτώνεται στην εσωτερική μνήμη του DSP όπου εκτελείται με μέγιστη ταχύτητα. Με αυτή τη πόρτα δεν χρειάζεται και η διασύνδεση με άλλους επεξεργαστές να είναι πολύπλοκη ώστε να χρειάζεται σηματοφορείς (semaphores) ή δίπορτες μνήμες (dual-port memories) για την ανταλλαγή δεδομένων κατί που θα αύξανε το κόστος για τέτοιου είδους εφαρμογές.

Η θύρα αυτή προσφέρει επίσης τη δυνατότητα ο DSP επεξεργαστής να ενοχλήσει με διακοπή τον κύριο επεξεργαστή, και μάλιστα σε περίπτωση πολλών παράλληλων επεξεργαστών DSP ή άλλων αιτιών διακοπών, υπάρχει και η δυνατότητα των διανυσματικών διακοπών (Vectored Interrupts) όπου η κάθε συσκευή που διακόπτει τον επεξεργαστή τον ενημερώνει για το που βρίσκεται ο αντίστοιχος κώδικας εκτέλεσης της διακοπής. Αυτό επιτρέπει την άμεση εκτέλεση της διακοπής χωρίς καθυστέρηση όπως στη περίπτωση του polling. Η διασύνδεση με κάποιον επεξεργαστή της σειράς 68000 είναι, μπορεί να πει κανείς, άμεση. Τέλος για τη μεταφορά των προγραμάτων μπορεί να χρησιμοποιηθεί και η απευθείας προσπέλαση στη μνήμη (DMA). Εκεί ο κεντρικός επεξεργαστής δεν ασχολείται με τη μεταφορά του κώδικα στο DSP, διότι αυτή την έχει αναλάβει ο ελεγκτής του DMA.

Τέλος αν κάποια από τις πόρτες δεν χρησιμοποιηθεί όπως παραπάνω, τότε οι ακροδέκτες της μπορούν να χρησιμοποιηθούν ως παράλληλη θύρα επικοινωνίας, όπου η λογική κατάσταση τους εξαρτάται άμεσα από το λογισμικό.

Ο ελεγκτής του εξωτερικού διαύλου είναι υπεύθυνος για την επικοινωνία με τον εξωτερικό ψηφιακό κόσμο όπως οι μνήμες, ή περιφερειακά που είναι διευθυνσιοδοτημένα στο χάρτη μνήμης. Καθορίζει τους κύκλους καθυστέρησης (wait states) σε περίπτωση που τα περιφεριακά είναι αργά όπως για παράδειγμα αργές μνήμες. Οι κύκλοι καθυστέρησης ρυθμίζονται και από το λογισμικό και από το υλικό. Τέλος υπάρχει η δυνατότητα της κοινοκτημοσύνης της εξωτερικής μνήμης με άλλο επεξεργαστή μέσω υλικού. Αυτό γίνεται μεσω κάποιων σημάτων ελέγχου όπου ο DSP αφήνει τους διαύλους σε κατάσταση απείρου αντίστασης, οπότε ο κύριος επεξεργαστής μπορεί να αφήσει ή να πάρει δεδομένα ή πρόγραμμα για τον DSP. Αυτός ο τρόπος είναι πιό περίπλοκος από την απλή σύνδεση του DSP με το Host interface, αλλά είναι ένας απλός τρόπος από το υλικό ώστε να επιτευχθεί μια στενή σύνδεση επεξεργαστών χωρίς πολύ πολύ λογισμικό αλλά και υλικό (semaphores, FIFO κλπ).

Η μεταφορές των δεδομένων πραγματοποιούνται με διπλή αποθήκευση (double buffering). Αυτό σημαίνει ότι εκτός από τους καταχωρητές ολίσθησης των δεδομένων για την αποστολή ή τη λήψη υπάρχουν και προσωρινοί καταχωρητές όπου τα δεδομένα παραμένουν για λίγο χρονικό διάστημα μέχρι να τα παραλάβει ο επεξεργαστής, ο οποίος λόγω φόρτου εργασίας ίσως δεν μπορεί να τα παραλάβει τη στιγμή της λήψης.

Η Μνήμη

Η μνήμη του EVM χωρίζεται σε τρεις περιοχές όπως η εσωτερική μνήμη του επεξεργαστή σε X,Y, και P. Οι περιοχές X,Y είναι περιοχές δεδομένων όπως για παράδειγμα τα δείγματα από τους αναλογικοψηφιακούς μετατροπείς. Η περιοχή P χρησιμοποιείται ως μνήμη προγράμματος. Οι περιοχές X,Y αποτελούνται από ολοκληρωμένα RAM, ενώ η περιοχή P έχει και RAM και ROM. Οι RAM και οι ROM είναι ειδικά ολοκληρωμένα που μπορούν να ανταποκριθούν στις ανάγκες για ταχύτατη επεξεργασία. Αυτό σημαίνει ότι έχουν πολύ μικρούς χρόνους όπως κύκλοι εγγραφής/ανάγνωσης, έτσι ώστε να μην υπάρχει η παραμικρή καθυστέρηση αποστολής ή λήψης δεδομένων από/προς τον επεξεργαστή. Αυτό δεν σημαίνει οτι δεν μπορούν να χρησιμοποιηθούν πιο αργές μνήμες, αφού μπορούν να προγραμματιστούν κύκλοι καθυστέρησης, αλλά σε περίπτωση που χρειάζεται η μέγιστη ταχύτητα επεξεργασίας αυτή θα είναι διαθέσιμη από το αναπτυξιακό όταν χρησιμοποιούνται γρήγορες μνήμες. Οι μνήμες τυχαίας προσπέλασης έχουν κύκλο ανάγνωσης/εγγραφής 25nS ενώ οι μνήμες ROM έχουν αντίστοιχο κύκλο 45nS. Η χωρητικότητα της μνήμης RAM είναι 8KWords (8K x 24 bit) χωρισμένα ως εξής:

4KWords Program memory
2KWords Data memory X
2KWords Data memory Y

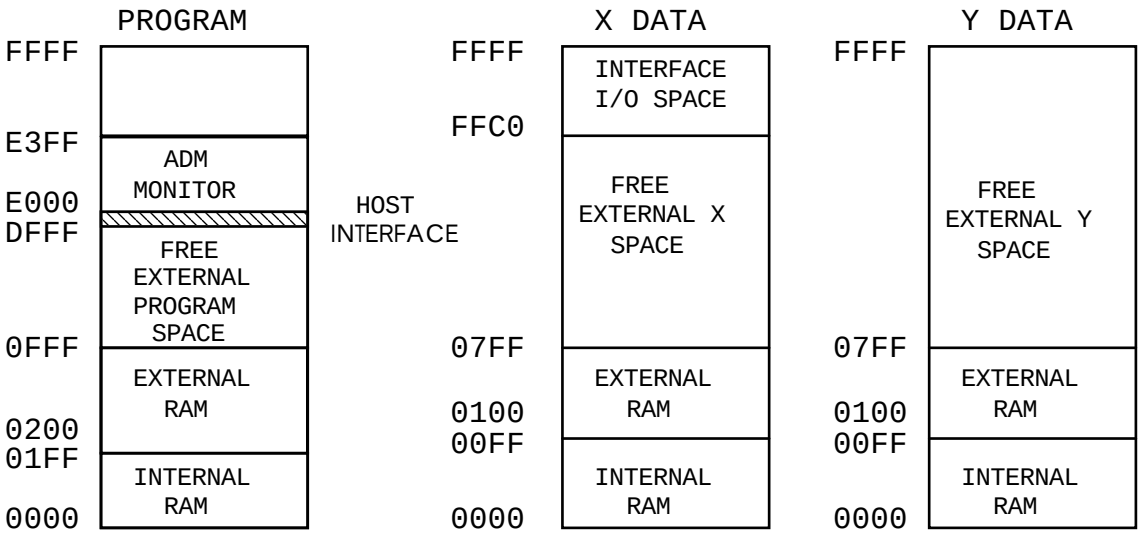
Αυτή η διάταξη μπορεί να αλλάξει αν θέλει κανείς μέσω των βραχυκυκλωτήρων. Επίσης μπορεί να χρησιμοποιθεί μεγαλύτερη μνήμη αν αντικατασταθούν τα υπάρχοντα ολοκληρωμένα με μεγαλύτερα των 32KByte.

Οι μνήμες ROM περιέχουν τον κώδικα του monitor που εκτελείται στο DSP για την ανάπτυξη και την εκσφαλμάτωση. Αυτές έχουν μέγεθος 2KWords. Πρέπει να σημειωθεί εδώ ότι η μνήμες αυτές είναι πολύ εξειδικευμένες (PROM με 45nS cycle time), και γι' αυτό δεν βρίσκονται καθόλου εύκολα στο εμπόριο, ακόμα και στο εξωτερικό (δηλαδή μην τολμήστε να κάψετε ΑΥΤΑ τα ολοκληρωμένα, καλύτερα κάψτε των επεξεργαστή...).

Αποκωδικοποίηση της μνήμης

Η μνήμη για λόγους ευελιξίας έχει ένα σύστημα επιλογής της διαμόρφωσης με συνδιασμό από βραχυκυκλωτήρες, διάκριτης λογικής και PAL (Programmable Array Logic). Αυτά επιτρέπουν την αλλαγή της διαμόρφωσης της μνήμης κατά βούληση, ανάλογα με την εφαρμογή, πολύ εύκολα. Αν χρειαστεί επέκταση στα 32KWords συνολικής μνήμης πρέπει να αναπρογραμματιστεί το PAL (το συγκεκριμένο που υπάρχει από το εργοστάσιο δεν αναπρογραμματίζεται, θα πρέπει να χρησιμοποιηθεί άλλο στη θέση του). Προσοχή έχει δοθεί στα κυκλώματα αποκωδικοποιήσης ώστε να είναι όσο το δυνατόν πιο γρήγορα, για να μη χρειαστούν κύκλοι καθυστέρησης από τον επεξεργαστή για την προσπέλαση της μνήμης. Αυτός είναι ο λόγος που έχουν χρησιμοποιηθεί σε κρίσιμα σημεία τα ενεργοβόρα ολοκληρωμένα της οικογένειας F-TTL (74Fxxx) με μέση κατανάλωση τα 100mA ανά ολοκληρωμένο.

Ο χάρτης μνήμης του EVM56001 φαίνεται στο παρακάτω σχήμα 3.2



Σχ. 3.2. Ο χάρτης μνήμης του ADS.

Εξωτερικές συνδέσεις

Οι συνδέσεις που προσφέρει το αναπτυξιακό αυτό εργαλείο της Motorola είναι οι εξής:

1. Host interface cable
2. Port B connector
3. Port C connector
4. Power Terminals
5. EuroConnector

Η πρώτη πόρτα (Host interface cable) αποτελεί το συνδετικό κρίκο του EVM με το PC. Από εκεί δίνουμε εντολές στο αναπτυξιακό σύστημα, κατεβάζουμε το πρόγραμμα και κάνουμε την εκσφαλμάτωση. Η πόρτα αυτή έχει το δικό της πρωτόκολλο επικοινωνίας που περιγράφεται στο εγχειρίδιο του EVM. Αυτή είναι και η μόνη σύνδεση που δεν έχει να κάνει άμεσα με μια θύρα επικοινωνίας του επεξεργαστή.

Η πόρτα B είναι το Host interface του DSP που περιγράψαμε στην παράγραφο του επεξεργαστή. Αν αυτή η πόρτα συνδιαστεί με την αλλαγή του τρόπου λειτουργίας του επεξεργαστή (Mode of operation) μπορούμε να κατεβάσουμε κώδικα εναλλακτικά από εκεί. Σε περίπτωση που αυτή η δυνατότητα δε μας χρειάζεται τότε μπορούμε να την χρησιμοποιήσουμε ως μια παράλληλη θύρα εισόδου/εξόδου.

Η πόρτα C είναι η πόρτα σειριακών επικοινωνιών, αφού εκεί στεγάζονται τα SCI και SSI. Μέσω αυτών των θυρών μπορούμε να συνδέσουμε το αναπτυξιακό των αναλογικοψηφιακών μετατροπεών της Motorola dsp56adc16. Αντίστοιχα στο SCI μπορούμε να συνδέσουμε ένα κύκλωμα διασύνδεσης με RS-232. Πάλι αν δεν χρειαζόμαστε αυτές της δυνατότητες μπορούμε να αλλάξουμε τη λειτουργία της θύρας σε γενική είσοδο/έξοδο.

Οι τροφοδοσίες του EVM μπορούν να έρθουν είτε από το ίδιο το PC, είτε να έρθουν από εξωτερικό τροφοδοτικό.

Τέλος η πιο ενδιαφέρουσα διασύνδεση είναι στον euroconnector. Εκεί καταλήγουν αυτούσιοι όλοι οι δίαυλοι του επεξεργαστή, σήματα αποκωδικοποίησης, και τροφοδοσίες. Εκεί μπορεί να συνδέσει κανείς ότι επιθυμεί, όπως για παράδειγμα επεκτάσεις μνήμης, αναλογικοψηφιακούς μετατροπείς, παράλληλες πόρτες κ.α. Η σύνδεση εξωτερικών περιφερειακών στο συνδετήρα αυτό θέλει μεγάλη προσόχη γιατι μπορούν να προκύψουν σοβαρές βλάβες στο αναπτυξιακό εργαλείο (όπως πχ. να καούν οι PROM...).

Ανάπτυξη και Εκσφαλμάτωση

Στο κεφάλαιο αυτό θα δούμε την συνολική ανάπτυξη του έργου. Στο πρώτο μερος, παρουσιάζονται, με συντομία και σε χρονολογική σειρά, όλες οι εργασίες που έγιναν. Στην συνέχεια, οι διάφορες επιμέρους εργασίες που έγιναν περιγράφονται πιό αναλυτικά, δίνοντας κάθε φορά βάρος στα σημεία που είναι πιό κρίσιμα ή που χρειάζονται περισσότερη προσοχή. Επίσης, οι αναφορές που γίνονται στις εφαρμογές για την αναγνώριση και το υλικό στο PC, είναι περιληπτικές, γιατί είναι τμήμα του πρώτου μέρους του έργου που παρουσιάστηκε απο τον Κώστα Γαλανάκη. Για περισσότερες πληροφορίες σε αυτά τα θέματα, ο αναγνώστης παρακαλείται να ανατρέξει στο κείμενο του κ. Γαλανάκη.

Επίσης, για τις αναφορές σε σχέδια, αρχεία προγραμματισμού και κώδικα, ο αναγνώστης παρακαλείται να ανατρέξει στο παράρτημα όπου αυτά φυλάσσονται. Στην αρχή του παραρτήματος, υπάρχει αναλυτικός κατάλογος με τα σχέδια και τον κώδικα που περιλαμβάνεται εκεί.

Συνοπτική Ανασκόπιση

Η εργασία αυτή ξεκίνησε με πενιχρά υλικά μέσα, αλλά με μια δυνατή ομάδα σπουδαστών που είχε σαν στόχο, και έχει επιτύχει, να κάνει μια εξερεύνηση στον χώρο της αναγνώρισης φωνής. Έτσι, ενώ αρχικά οι στόχοι της εργασίας ήταν μεγαλεπίβολοι και πρωτοποριακοί, οι ελλείψεις στον οικονομικό, κυρίως, τομέα όχι μόνο περιόρισαν τους αρχικούς στόχους, αλλά μας ανάγκασαν να ακολουθήσουμε στρατηγικές οι οποίες μας εξοικονόμησαν χρήματα, αλλά κόστισαν πολύ σε χρόνο. Έτσι, βλέπαμε άλλους να παράγουν εμπορικά προϊόντα αναγνώρισης φωνής, ενώ όταν πρωτοξεκίνησε η ασχολία μας με το θέμα αυτό, ήταν καθαρά μέσα στα όρια της ερευνητικής προσπάθειας.

Ο πρώτος στόχος ήταν να κατασκευαστεί μια “διαφανής” προσαρμογή του DOS ώστε να μπορεί να δέχεται είσοδο όχι μόνο από το πληκτρολόγιο, αλλά και από το μικρόφωνο. Εφόσον η διασύνδεση θα γινόταν σε επίπεδο λειτουργικού συστήματος, όλες οι εφαρμογές θα μπορούσαν να εκμεταλλευτούν αυτές τις νέες δυνατότητες. Το λεξιλόγιο του συστήματος θα ήταν πολύ περιορισμένο, γύρω στις 60 λέξεις, όμως θα είχε αρκετά μεγάλο βαθμό επιτυχίας, ώστε να μπορεί να διευκολύνει τον χρήστη στον χειρισμό του PC. Η κάθε λέξη θα έπρεπε να ομιλείται ξεχωριστά και καθαρά στο μικρόφωνο. Η βάση δεδομένων θα προοριζόταν για ένα χρήστη, και μετά από μια προκαταρκτική εκπαίδευση για να ξεκινήσει την λειτουργία του, θα προσαρμοζόταν δυναμικά στα χαρακτηριστικά της φωνής του χρήστη (speaker adaptive). Με αυτό τον τρόπο, αυξάνεται η αποτελεσματικότητα του συστήματος με την μακροχρόνια χρήση. Η επεξεργασία των σημάτων έπρεπε φυσικά να γίνεται σε πραγματικό χρόνο, πράγμα που έθεσε από την αρχή περιορισμούς τόσο στην επιλογή του χρησιμοποιούμενου επεξεργαστή, όσο και στην επιλογή της συχνότητας δειγματοληψίας, που έμμεσα καθορίζει το πλήθος των υπολογισμών που απαιτούνται για την επεξεργασία. Ας δούμε λοιπόν την χρονική εξέλιξη της εργασίας αυτής, που κατέληξε να είναι μια εξαιρετικά χρήσιμη και ενδιαφέρουσα εξερεύνηση σε ένα κόσμο γνώσεων, προβλημάτων, λύσεων, και κυρίως, αυτού που θα ονομάζαμε “Φιλοσοφία της Τέχνης μας”.

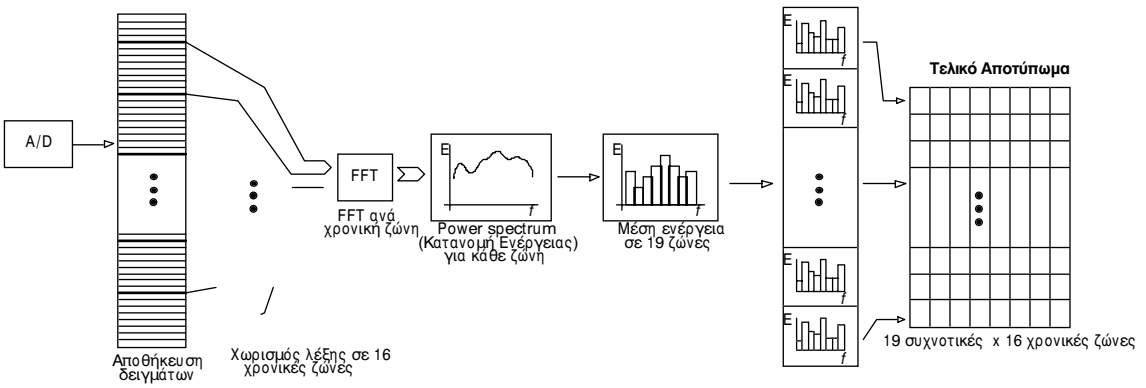
Αρχικό στάδιο

Από την αρχή είχαμε σαν στόχο να χρησιμοποιήσουμε έναν ειδικό επεξεργαστή σήματος, πράγμα που τελικά συνέβη. Όμως, είχαμε τεράστιες δυσκολίες όχι μόνο να αναπτύξουμε κώδικα σε αυτόν τον επεξεργαστή, αλλά και να βρούμε σε ποιό σημείο βρισκόταν η έρευνα εκείνη την εποχή. Τελικά μια πρώτη γεύση για την τρέχουσα κατάσταση έγινε μέσω του Internet και μέσω των Usenet News. Στο comp.speech και στο comp.dsp αρχίσαμε να ξετρυπώνουμε τις πρώτες πολύτιμες πληροφορίες που μας βοήθησαν να καλύψουμε τις αδυναμίες μας. Παράλληλα, ήρθαν και τα πρώτα βιβλία που παραγγείλαμε από τις ΗΠΑ, και πλέον από εκεί αρχίσαμε πραγματικά να αυξάνουμε το επίπεδο των γνώσεών μας στο θέμα. Με τα newsgroups στο Internet, καλύπταμε κενά που δέν μπορούσαμε να καλύψουμε από βιβλία ή τους καθηγητές μας. Έτσι, γρήγορα καταλήξαμε στα προσχέδια του αλγορίθμου που ακολουθήσαμε, τα οποία βεβαίως άλλαξαν πολλές φορές από τότε, αλλά η φιλοσοφία του έργου παρέμεινε η ίδια.

Η Προσέγγιση του θέματος

Η γενική φιλοσοφία της αναγνώρισης ήταν να προσπαθήσουμε να καταγράψουμε με όσο το δυνατόν μεγαλύτερη πιστότητα, αλλά καί σε μικρό χώρο, την χρονική εξέλιξη μιας λέξης. Αυτό θα μας έδινε ένα μέτρο σύγκρισης, δηλαδή κάποιες πρότυπες λέξεις με τις οποίες θα συγκρίνονταν οι λέξεις του ομιλητή ώστε να γίνεται η αναγνώρισή τους. Από την πρώτη στιγμή, φάνηκε οτι δέν ήταν δυνατό να χρησιμοποιηθούν οι παραστάσεις των λέξεων μόνο στο πεδίο του χρόνου, επειδή είχαν τεράστιες διακυμάνσεις στην εμφάνισή τους. Μια τέτοια προσέγγιση θα απαιτούσε να χρησιμοποιηθούν πολύ προχωρημένοι αλγόριθμοι pattern matching, και πάλι το αποτέλεσμα δέν θα ήταν καθόλου ικανοποιητικό. Έτσι, χρησιμοποιήσαμε μια διαφορετική απεικόνιση της κάθε λέξης, που θα μπορεί να αντιλαμβάνεται την εξέλιξή της και στο πεδίο των συχνοτήτων, που έχει λιγότερες διακυμάνσεις.

Κάναμε λοιπόν την εξής αρχική σχεδίαση. Η λέξη του ομιλητή θα χωρίζεται σε 16 χρονικές ζώνες. Με αυτό θέλουμε να επιτύχουμε την απεξάρτηση της εξέλιξης της λέξης από την χρονική της διάρκεια. Δηλαδή, γίνεται μια μορφή "Γραμμικής Δυναμικής Παραμόρφωσης", ώστε να μειωθούν κατά το δυνατόν οι διακυμάνσεις στις διαφορές που έχει κάποια λέξη από την μια της ομιλία στην επόμενη. Με τον χωρισμό της κάθε λέξης σε σταθερό αριθμό χρονικών ζωνών, έχουμε την κανονικοποίησή τους σε ένα σταθερό "καλούπι", τουλάχιστον όσον αφορά το πεδίο του χρόνου. Έτσι, γίνονται δυνατές οι συγκρίσεις μεταξύ των λέξεων.



Σχ. 1.1. *Ο αρχικός αλγόριθμος αναγνώρισης*

Το επόμενο βήμα είναι να παραχθεί, σε κάθε χρονική ζώνη ξεχωριστά, η φασματική εικόνα της λέξης στην ζώνη αυτή. Η ακολουθία των αποτελεσμάτων αυτού του βήματος, μας δίνει την φασματική εξέλιξη της λέξης με την πάροδο του χρόνου, δηλαδή ένα φασματόγραμμα. Αυτό είναι και η μορφή που θα χρησιμοποιούμε για την αποθήκευση και την αναγνώριση των λέξεων. Όμως, σ' αυτό το στάδιο, το φασματόγραμμα έχει πολύ μεγάλη περιεκτικότητα πληροφορίας, που όχι μόνο μεγαλώνει το μέγεθος της μνήμης που απαιτείται, αλλά δυσχεραίνει και την αναγνώριση, μιας και παρεμβάλλονται πολλές πληροφορίες (συχνότητες) που έχουν μεγάλες διακυμάνσεις.

Έτσι, προχωρούμε στη μείωση της πληροφορίας στο πεδίο των συχνοτήτων, κάνοντας έναν διαχωρισμό συχνοτήτων, και κρατώντας τα στοιχεία που μας χρειάζονται μόνο σε κάποιες ευρείες ζώνες, όπου παρουσιάζεται το μεγαλύτερο ενδιαφέρον. Μετά από την αναζήτησή μας στην βιβλιογραφία, καταλήξαμε στον χωρισμό 19 συχνοτικών ζωνών, που είναι επιλεγμένες ώστε να διαχωρίζουν τις βασικές συχνότητες των οργάνων παραγωγής ομιλίας του ανθρώπου. Άρα, παρακολουθούμε την εξέλιξη των συνχοτήτων της λέξης στην πιό γενική, "μακροσκοπική", τους εμφάνιση. Αυτό που μας ενδιαφέρει δέν είναι το ακριβές συχνοτικό περιεχόμενο της λέξης, αλλά το πώς εξελίσσονται οι βασικές συχνότητες με την πάροδο του χρόνου, και πιό συγκεκριμένα, η ενέργεια που περιέχεται στην κάθε συχνοτική περιοχή. Στον επόμενο πίνακα, φαίνονται οι ζώνες που χρησιμοποιήσαμε, και είναι οι ίδιες με αυτές που χρησιμοποιήθηκαν και από άλλους ερευνητές.

Ζώνη	Κάτω όριο	Άνω όριο
1	180Hz	300Hz
2	300Hz	420Hz
3	420Hz	540Hz
4	540Hz	660Hz
5	660Hz	780Hz
6	765Hz	915Hz
7	925Hz	1075Hz
8	1075Hz	1225Hz
9	1225Hz	1375Hz
10	1375Hz	1525Hz
11	1525Hz	1675Hz
12	1700Hz	1900Hz
13	1900Hz	2100Hz
14	2100Hz	2300Hz
15	2300Hz	2500Hz
16	2600Hz	2800Hz
17	2850Hz	3150Hz
18	3150Hz	3450Hz
19	3500Hz	4000Hz

Με βάση λοιπόν το φασματόγραμμα, που τελικά είναι ένας πίνακας 19 γραμμών (συχνοτικές ζώνες) και 16 στηλών (χρονικές ζώνες), δηλαδή 304 αριθμών που απεικονίζουν την ενέργεια του σήματος εισόδου στην κάθε ζώνη την κάθε χρονική στιγμή, έχουμε επιτύχει σημαντική μείωση της πληροφορίας, δηλαδή καποια μορφή συμπίεσης με απώλειες. Το αποτέλεσμα έχει όση πληροφορία χρειάζεται για να γινει η αναγνώριση, και είναι αρκετά μικρό ώστε να μήν καταλαμβάνει πολύ μνήμη.

Το τελευταίο στάδιο είναι η κανονικοποίηση του φασματογράμματος με βάση την μέγιστη τιμή του. Με αυτό τον τρόπο επιτυγχάνεται έμμεσα και κατά προσέγγιση η αντιστάθμιση του αλγορίθμου στις διακυμάνσεις της έντασης με την οποία

προφέρεται η λέξη από τον ομιλητή. Με τον έλεγχο του εύρους της μονάδας αποθήκευσης που θα χρησιμοποιηθεί στον υπολογιστή, μπορεί να γίνει έλεγχος του δυναμικού εύρους που θα έχει το τελικό αποτέλεσμα (πχ 96db στα 16bit ή 144db στα 24bit κλπ).

Προχωρώντας στην ανάπτυξη

Επειδή η ανάπτυξη των αλγορίθμων κατευθείαν στο DSP ήταν δύσκολη και επειδή βρισκόμασταν ακόμη σε διερευνητικό στάδιο, που όλα ήταν ρευστά, αποφασίσαμε να κάνουμε την πρώτη ανάπτυξη σε ένα PC, προγραμματίζοντας σε C. Αυτό θα μας έδινε την ευχέρεια να κάνουμε εύκολα αλλαγές στον αλγόριθμο, και ταυτόχρονα μας γλύτωσε από την απότομη καμπή του χρόνου εκμάθησης του DSP. Παράλληλα όμως, δημιουργήθηκαν και τα πρώτα προβλήματα: Δέν είχαμε τρόπο να κάνουμε δειγματοληψίες στο PC.

Επειδή το κόστος για την αγορά μιας κάρτας ήχου ήταν αρκετά υψηλό, καταλήξαμε στην τροποποίηση μιας κάρτας μετρήσεων που είχαμε κατασκευάσει για προσωπική χρήση, ώστε να μπορέσουμε να την συνδέσουμε σ' ένα μικρόφωνο και να δειγματοληπτήσουμε λέξεις. Η κάρτα ήταν αρκετά γρήγορη (500kHz), αλλά έχει κάπως μικρή ανάλυση (8bit). Για την προσαρμογή έπρεπε να σχεδιάσουμε και ένα αναλογικό τμήμα με προενισχυτές μικροφώνου και φίλτρα LP για την απόρριψη των ειδώλων που θα δημιουργούσε ο A/D άν το σήμα είχε μεγάλο εύρος συχνοτήτων.

Η κάρτα βασίζεται στον ADC0820 της National, στον οποίο προστέθηκε και ένας DAC0830, ώστε να έχουμε και μια αναλογική έξοδο από το PC. Με την προσθήκη και μερικών τελεστικών ενισχυτών είχαμε και τους ενισχυτές για το μικρόφωνο και την έξοδο, καθώς και τα φίλτρα που χρειαζόμασταν. Η κάρτα τοποθετείται στον εσωτερικό δίαυλο του PC (ISA, 8 bit). Τα σχέδια της κάρτας βρίσκονται στο παράρτημα, μαζί με τα υπόλοιπα σχέδια ανάπτυξης του DSP. Η κάρτα εργάζεται ακόμη άψογα, ακόμα και σε τελευταίου τύπου Pentium PC. Για την αναλογική έξοδο, μέχρι να προστεθεί στην κάρτα ο DAC0830, χρησιμοποιούσαμε έναν D/A με σκάλα αντιστάσεων R-2R συνδεδεμένη στην παράλληλη θύρα του PC. Το υλικό αυτό χρησιμοποιείται και από πολλά προγράμματα shareware για εφαρμογές ήχου, τα οποία χρησιμοποιήσαμε και εμείς για να επεξεργαζόμαστε και να ακούμε τα δείγματα που παίρναμε με την κάρτα.

Πάνω στην σχεδίαση των αναλογικών φίλτρων για τον A/D, φτάσαμε και στο στο πρόβλημα της επιλογής της συχνότητας δειγματοληψίας. Μετά από μελέτη της σχετικής βιβλιογραφίας, καταλήξαμε να χρησιμοποιήσουμε την συχνότητα των 10kHz. Η συχνότητα αυτή είναι αρκετά υψηλή ώστε να περιέχει όλες τις απαραίτητες πλητοφορίες για την αναγνώριση, δίνοντας ένα θεωρητικό εύρος 5kHz. Να σημειώσουμε εδώ οτι η τηλεφωνική γραμμή περιορίζει το εύρος σε 3.5kHz, και όμως μπορούμε να αντιλαμβανόμαστε άψογα τον ήχο και την ομιλία, έστω και σε τόσο μικρό εύρος. Μετά από μερικά πειράματα, διαπιστώσαμε και εμείς οτι το μεγαλύτερο μέρος της ενέργειας στην ομιλία, βρίσκεται σε πολύ χαμηλές συχνότητες, μέχρι τα 1250Hz, σε ποσοστό που ξεπερνά το 90%. Έχοντας δεχθεί και ότι η συχνότητα των 10 KHz δέν δημιουργεί υπερβολικό πλήθος δεδομένων προς επεξεργασία, καταλήξαμε να την χρησιμοποιήσουμε.

Τα πρώτα προγράμματα

Η αρχή του προγραμματισμού ήταν αρκετά δύσκολη, μιας και έπρεπε να ξεκινήσουμε από ένα τρόπο με τον οποίο να μιλάμε στο υλικό που κατασκευάσαμε, δηλαδή την κάρτα A/D και τον D/A. Στο περιβάλλον του PC υπάρχουν κυριολεκτικά εκατοντάδες διεργασίες που τρέχουν με task-switching, δηλαδή μοίρασμα του υπολογιστικού χρόνου της μονάδας επεξεργασίας (CPU) σε διάφορα προγράμματα.

Αυτό μας εμπόδιζε, γιατί τα δικά μας προγράμματα έπρεπε να τρέχουν με σταθερό ρυθμό, ώστε να έχουμε καλή δειγματοληψία, τόσο κατά την είσοδο, όσο και κατά την έξοδο των δειγμάτων. Επειδή, άν προγραμματίζαμε με τον κλασσικό τρόπο, δέν θα υπήρχε τρόπος να εξασφαλίσουμε μια τέτοιου είδους εκτέλεση, καταλήξαμε στην παγίδευση του μοναδικού interrupt του PC που εκτελείται σε σταθερά χρονικά διαστήματα, μιας και συνδέεται απευθείας στον χρονιστή του συστήματος. Το interrupt αυτό, έχει και την μεγαλύτερη προτεραιότητα μετά το NMI, οπότε είμαστε σίγουροι οτι δέν θα εμποδίσει την εκτέλεσή του σχεδόν κανένα άλλο πρόγραμμα. Βέβαια, υπήρχε πάντα η πιθανότητα να υπάρξει κάποιο άλλο πρόγραμμα που θα αλλοίωνε ελαφρά τον χρονισμό της διακοπής (interrupt), αλλά αυτό αποφασίσαμε να το δεχτούμε, μιας και η αλλοίωση δέν θα υπήρχε παρά σε ένα δείγμα ανάμεσα σε χιλιάδες.

Το πρόβλημα με την δέσμευση αυτής της γραμμής διακοπής είναι οτι το PC έχει ρυθμιστεί, ή μάλλον προγραμματιστεί, να την εκτελεί 18.2 φορές, κατά μέσο όρο, το δευτερόλεπτο. Βέβαια, η συχνότητα αυτή είναι πολύ μικρή για δειγματοληψίες, οπότε έπρεπε οπωσδήποτε να αλλάξει. Το πρόβλημα αυτό λύθηκε με έναν έξυπνο τρόπο, στον οποίο επαναπρογραμματίζεται ο χρονιστής του συστήματος να ενεργοποιεί την διακοπή με την επιθυμητή συχνότητα δειγματοληψίας, δηλαδή τα 10KHz. Για να μήν επηρεαστούν οι υπόλοιπες διεργασίες του PC που στηρίζονταν στον παλαιό κώδικα εξυπηρέτησης διακοπών, με συχνότητα εκτέλεσης 18.2Hz, ανά μερικές κλήσεις της νέας ρουτίνας εξυπηρέτησης διακοπών, που γράψαμε εμείς, καλούνταν και ο παλιός κώδικας, μέσα από το νέο. Έτσι, τα υπόλοιπα προγράμματα ξεγελιούνται, και νομίζουν οτι εκτελείται μόνο ο παλιός χειριστής, και μάλιστα στην σωστή του συχνότητα. Βέβαια, επειδή δέν έχουμε ακριβώς την παλιά συχνότητα, παρατηρούνται μερικά περίεργα φαινόμενα, όπως οτι το ρολόι του υπολογιστή τρέχει λίγο πιο γρήγορα. Αυτό όμως είναι μιά πολύ μικρή ατέλεια, την οποία δεχθήκαμε.

Μετά την λύση του προβλήματος αυτού, κατασκευάστηκε ολόκληρο το πρόγραμμα διασύνδεσης του χειριστή διακοπής που κάνει τις δειγματοληψίες και τις αναπαραγωγής των δειγμάτων. Μπορείτε να το δείτε στο παράρτημα, κάτω από το όνομα sample.h και sample.c. Η εκτέλεση του προγράμματος σε τακτικά διαστήματα επαληθεύτηκε με μετρήσεις που έγιναν με παλμογράφο. Συγκεκριμένα, κάναμε δύο ελέγχους. Στον πρώτο, ο χειριστής διακοπής ενεργοποιούσε σε κάθε εκτέλεσή του ένα από τους ακροδέκτες της παράλληλης θύρας, το οποίο παρατηρούσαμε στον παλμογράφο. Ο ρυθμός επανάληψης ήταν αρκετά σταθερός, αλλά σποραδικά παρατηρήσαμε μικρά “παιξίματα” (jitter). Ο δεύτερος έλεγχος, ήταν να παράγουμε με λογισμικό ένα ημίτονο, και να το στείλουμε στήν έξοδο του D/A. Παρατηρώντας το ημίτονο στον παλμογράφο, είδαμε οτι μετά τα 15KHz παρουσίαζε παραμορφώσεις. Αυτό οφείλεται στην συγγραφή του χεριριστή διακοπής, που σε μεγάλο του μέρος έγινε σε C, και όχι σε assembly, που θα έδινε μικρότερο και ταχύτερο κώδικα. Αποφασίσαμε να αναβάλλουμε την επανασύνταξη του χειριστή σε assembly, γιατί με την δειγματοληψία που χρησιμοποιούσαμε δέν είχαμε κανένα πρόβλημα.

Το επόμενο βήμα ήταν να χρησιμοποιήσουμε το πρώτο αυτό στρώμα επικοινωνίας με το υλικό, ώστε να βρούμε τα όρια του υλικού που είχαμε τότε στα χέρια μας. Έτσι, αρχίσαμε πρώτα να δοκιμάζουμε τα όρια της ισχύος των υπολογιστών, που ήταν δύο 386/40MHz, από τα οποία το ένα με μαθηματικό συνεπεξεργαστή. Είδαμε έτσι τους μέγιστους ρυθμούς δειγματοληψίας που μπορούσαμε να επιτύχουμε, και το δεύτερο βήμα ήταν να κάνουμε μια σειρά προγραμμάτων για να παρατηρούμε την συμπεριφορά των χαρακτηριστικών της φωνής.

Το πρώτο από αυτή την σειρά των προγραμμάτων ήταν ένα πρόγραμμα απεικόνισης, που εξομοίωνε την λειτουργία ενός παλμογράφου. Με αυτό μπορούμε να μιλάμε στο μικρόφωνο και να βλέπουμε σε γραφική μορφή το σήμα εισόδου στην

οθόνη. Προχωρώντας παραπέρα, γράψαμε τις πρώτες ρουτίνες που κάνουν αναγνώριση των βασικών χαρακτηριστικών της φωνής, όπως η μέση ενέργεια, το μέσο πλάτος, και τα περάσματα από το μηδέν. Αυτά έχουν υλοποιηθεί στην βιβλιοθήκη timedom.c και timedom.h. Με το ίδιο πρόγραμμα μπορούσαμε να εκτιμήσουμε τα χαρακτηριστικά του αναλογικού μέρους του μετατροπέα, μιας και η ενέργεια και τα περάσματα από το μηδέν με το μικρόφωνο κλειστό είναι η ενέργεια του ηλεκτρικού θορύβου και η συχνότητά του αντίστοιχα, το δέ μέσο πλάτος είναι το dc offset του αναλογικού συστήματος.

Πάνω στις δοκιμές, με το μικρόφωνο να λειτουργεί σε ένα τυπικό χώρο εργασίας γραφείου, είδαμε οτι ίσως θα μπορούσαμε να χρησιμοποιήσουμε έναν διαφορικό ενισχυτή με δύο μικρόφωνα, ένα κοντά στον ομιλητή και ένα λίγο μακρυτερα, ή με μικρότερη ευαισθησία. Με αυτό τον τρόπο θα πετυχαίναμε απόρριψη του θορύβου του περιβάλλοντος, μιας και το δεύτερο, απομακρυσμένο, μικρόφωνο θα λάμβανε τον ίδιο θόρυβο με το πρώτο, αλλά όχι και τον ομιλητή στην ίδια ένταση. Έτσι θα μπορούσαν να εξουδετερωθούν οι θόρυβοι του περιβάλλοντος. Δέν προχωρήσαμε παρακάτω στην υλοποίηση αυτής της μεθόδου, γιατί απαιτεί πολύπλοκη σχεδίαση ενός διαφορικού ενιχυτή με ελεγχόμενη φάση και ενίσχυση.

Στην συνέχεια, προσπαθήσαμε να φτιάξουμε μια βιβλιοθήκη με συναρτήσεις επεξεργασίας σήματος, όπως φιλτραρίσματα και μετασχηματισμό Φουριέ (FFT). Μετά από τρείς προσπάθειες, τελικά φτιάξαμε έναν μετασχηματισμό Φουριέ που ήταν καί γρήγορος, καί μπορούσε να χειριστεί αρκετά μεγάλα τμήματα δειγμάτων, μέχρι 16K. Ενσωματώνοντας το FFT με τις υπόλοιπες βιβλιοθήκες μπορέσαμε να κάνουμε ένα πρόγραμμα που απεικονίζει την αναλογική είσοδο από το μικρόφωνο και την δείχνει στο μισό μέρος της οθόνης, ενώ στο άλλο μισό φαίνεται το φάσμα του σήματος μέσω του μεταχηματισμού FFT. Το πρόγραμμα αυτό είναι πολύ χρήσιμο στην μελέτη των χαρακτηριστικών της φωνής, αφού δείχνει όχι μόνο αριθμητικά, αλλά και γραφικά αποτελέσματα, που είναι πιό εύκολο να ερμηνευτούν.

Κατά την λειτουργία του προγράμματος αυτού, έγινε αντιληπτό οτι ο υπολογιστής δέν μπορούσε να ανταπεξέλθει ικανοποιητικά, μιας και είχαμε μόνο μερικές ανανεώσεις της οθόνης ανά δευτερόλεπτο. Έτσι, έγινε έντονη και η ανάγκη για βελτιστοποιήσεις του αλγορίθμου, καθώς και η μεταφορά του σε έναν πραγματικά ισχυρό επεξεργαστή, όπως τον DSP56001.

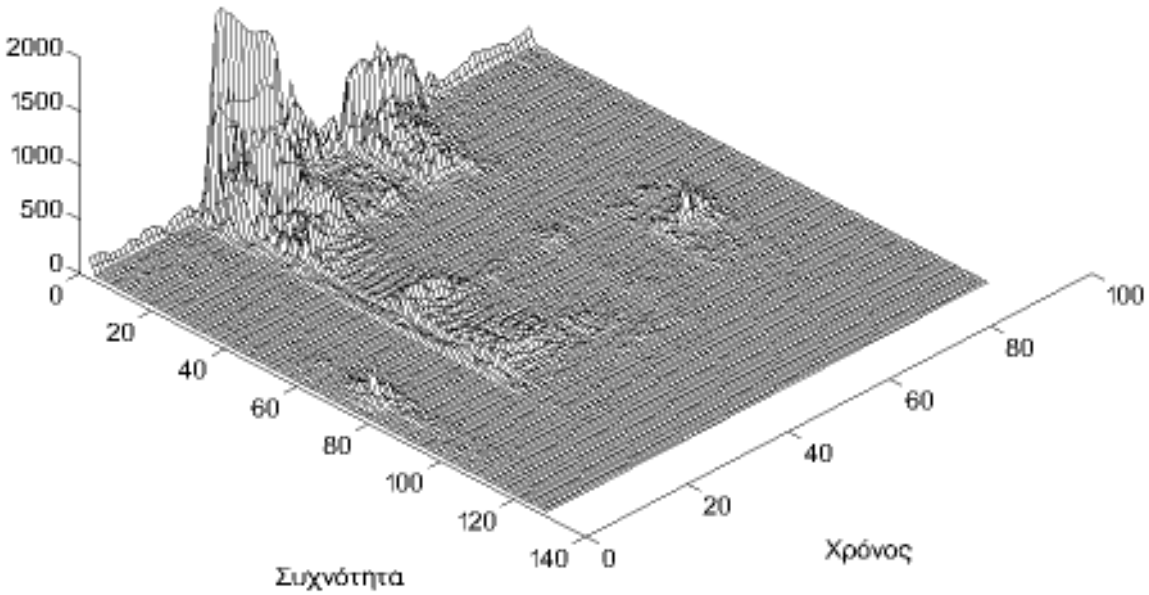
Αλλαγή πορείας

Προσπαθώντας να πραγματοποιήσουμε την εφαρμογή στον DSP56001, βρεθήκαμε μπροστά σε μερικά πολύ ενδιαφέροντα συμπεράσματα. Πρώτον, οτι η μνήμη του αναπτυξιακού συστήματος που είχαμε δέν επαρκούσε για την ανάπτυξη του προγράμματος, μιας και δέν μπορούσαμε να αποθηκεύσουμε μεγάλο αριθμό δειγμάτων, τόσων ώστε να μπορούμε να κρατάμε μια ολόκληρη λέξη στην μνήμη του DSP για επεξεργασία. Παρατηρήσαμε επίσης οτι αυτό ακριβώς ήταν το μειονέκτημα της μεθόδου που χρησιμοποιούσαμε ως τώρα, δηλαδή οτι απαιτεί να γίνεται η δειγματοληψία ολόκληρης της λέξης, πρίν από οποιαδήποτε επεξεργασία, αλλά στο αχανές από πλευράς μνήμης PC, το πρόβλημα δέν είχε γίνει, ως τώρα, ορατό. Φτάσαμε λοιπόν στο στάδιο που έπρεπε να αντιμετωπίσουμε το μεγαλύτερο πρόβλημα στην υλοποίηση του όλου έργου. Πώς θα μειώσουμε, κατά το δυνατόν, την απαιτούμενη μνήμη που χρησιμοποιεί ο αλγόριθμος, χωρίς όμως να μειώσουμε και την αποτελεσματικότητά του ή την ταχύτητά του.

Οπλισμένοι αυτή τη φορά καί με το περίφημο Matlab, αρχίσαμε να εξερευνούμε τις δυνατότητες που βλέπαμε μπροστά μας. Με το νέο μας εργαλείο, το Matlab, μπορέσαμε να κάνουμε πολλά πειράματα, και να ανακατασκευάσουμε όλη την μέχρι

τότε εργασία μας πάνω σε προγράμματα σε C, για την επεξεργασία σήματος, μόλις σε δύο-τρία απογεύματα. Αυτό μας στεναχώρησε λίγο, επειδή άν είχαμε από την αρχή το εργαλείο αυτό, θα μας είχε βοηθήσει πολύ στο να λύσουμε όλα τα προβλήματα σχετικά με την επεξεργασία σήματος πολύ-πολύ νωρίτερα. Πάντως, σε κάθε περίπτωση, κάναμε πολύ ενδιαφέρουσες παρατηρήσεις πάνω στον παλιό αλγόριθμο.

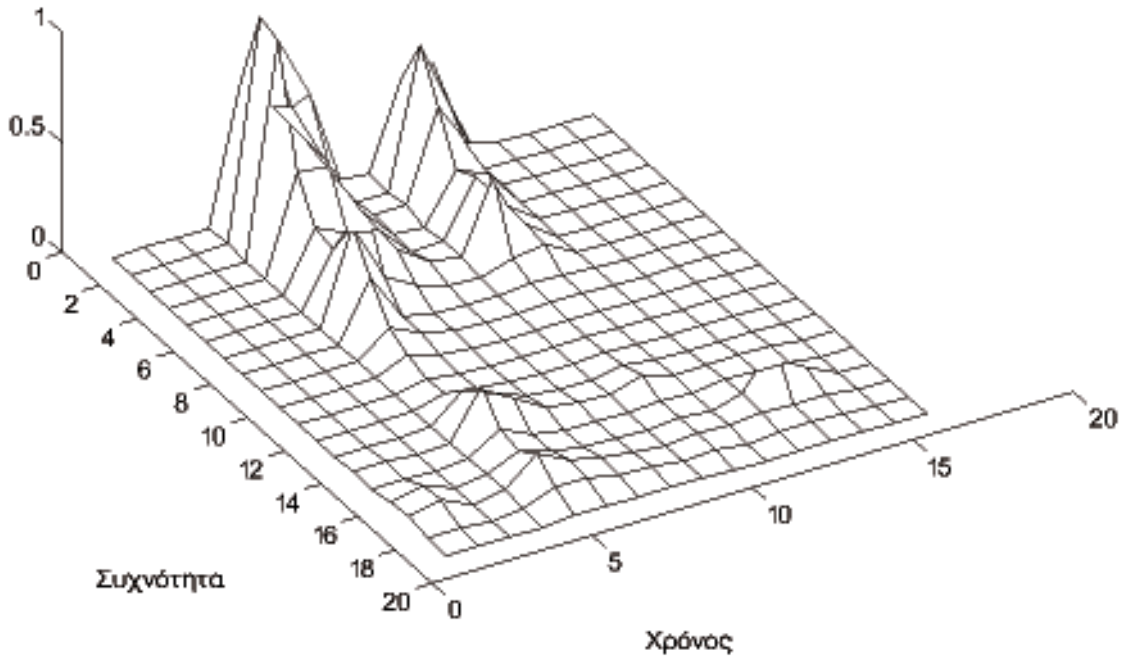
Πρώτον, παρατηρήσαμε οτι ο αλγόριθμος που είχαμε σκεφτεί, έκανε μια πολύ λεπτομερή επεξεργασία των δεδομένων του για να παράγει, με FFT, την στιγμιαία φασματική ανάλυση του σήματος εισόδου, όπως φαίνεται και στο σχήμα 4.1. Τα δεδομένα αυτά ήταν πολλά, αυξάνοντας έτσι όχι μόνο τις αποθηκευτικές απαιτήσεις, αλλά καθυστερώντας πολύ και τους υπολογισμούς, επειδή τα FFT γίνονταν πολύ μεγάλα (από 4K-16K). Επιπλέον, το αποτέλεσμα αυτού του κουραστικού FFT που μας έδινε τρομερή φασματική ανάλυση (μερικά εκατοστά του Hz), μας ήταν σχεδόν άχρηστο, και ενώ είχαμε "ταλαιπωρηθεί" υπολογίζοντάς το με τόση τρομερή ακρίβεια, στο επόμενο βήμα ουσιαστικά "πετούσαμε" το μεγαλύτερο μέρος από αυτή την πληροφορία, περνώντας το αποτέλεσμα του FFT από μερικούς μέσους όρους για να βρούμε αυτό που μας ενδιέφερε, δηλαδή την ενέργεια σε μερικές, χοντροειδείς μπροστά στην ανάλυση του FFT, ζώνες. Είχαμε δηλαδή μια τρομερή σπατάλη υπολογιστικής ισχύος.



Σχ. 4.2. *Το πλήρες φασματόγραμμα μιας λέξης.*

Δεύτερον, η όλη λειτουργία του αλγορίθμου, μας ανάγκαζε να δειγματο-ληπτήσουμε ολόκληρη την λέξη, μαζί με αρκετά δείγματα πρίν την αρχή και μετά το τέλος της, ώστε να μπορέσουμε να βρούμε την αρχή και το τέλος της, και μετά να δούμε το μέγεθος της κάθε μιας από τις 16 χρονικές ζώνες πάνω στις οποίες έπρεπε να γίνει η φασματική ανάλυση. Αυτό απαιτεί μνήμη πολλαπλάσια του ρυθμού δειγματοληψίας. Δηλαδή, άν θέλαμε να αποθηκεύσουμε μια λέξη των τριών δευτερολέπτων, θα έπρεπε να έχουμε 30KB μνήμης (για δειγματοληψία 10KHz). Αυτό, στο DSP ήταν αδύνατο, μιας και ο διαθέσιμος χώρος έπρεπε να μοιραστεί όχι μόνο στον χώρο για τα δείγματα, αλλά και στις υπόλοιπες λειτουργίες επεξεργασίας

σήματος. Για παράδειγμα, ένα FFT των 4096 σημείων, θα απαιτούσε σχεδόν 8K μνήμης, μαζί με τα ενδιάμεσα αποτελέσματα. Έτσι, στρέψαμε την προσοχή μας στο να βρούμε τρόπο να "συμπιέσουμε" την πληροφορία που αποθηκεύουμε στην μνήμη.



Σχ. 4.3. *Το φασματόγραμμα του προηγούμενου σχήματος, μετά την μείωση της πληροφορίας.*

Το υλικό για το DSP

Παράλληλα, για λόγους εκμάθησης του DSP και την εξοικείωση μας με το EVM από πλευράς υλικού αποφασίσαμε να κατασκευάσουμε ένα μετατροπέα από ψηφιακό σε αναλογικό. Η κάρτα αυτή θα συνδεόταν με το EVM μέσω του συνδετήρα γενικών επεκτάσεων (Euroconnector 96 pin). Το DSP θα έβλεπε το περιφερειακό αυτό ως μια θέση μνήμης, δηλαδή το περιφερειακό D/A θα ήταν memory mapped. Ο D/A που διαλέξαμε είναι ο ICL7121. Αυτός είναι ένας 16 bit μετατροπέας της Harris semiconductor με μέγιστο χρόνο σταθεροποιήσης της εξόδου τα 3μS, με μικρό γραμμικό σφάλμα χάρις στο πίνακα PROM που μπορεί να προγραμματιστεί κατάλληλα για να το μειώσει στο ελάχιστο, και κατανάλωση 500mW. Με αυτό τον τρόπο θα ελέγχαμε τη λογική λειτουργίας του διαύλου του DSP και θα είχαμε την πρώτη άποψη για το πως θα μπορούσε να υλοποιηθεί μια επέκταση μνήμης αργότερα.

Λόγω της βλάβης που προκαλέσαμε στο EVM κατά τη διάρκεια των δοκιμών θα έπρεπε να περιμένουμε λίγο για την ανάπτυξη του λογισμικού. Αυτό μας έδωσε χρόνο να αναδιοργανώσουμε τη σκέψη μας, ενώ παράλληλα ξεκινήσαμε τη σχεδίαση του θεωρητικού κυκλώματος και κατόπιν της πλακέτας της επέκτασης μνήμης. Στο μεταξύ επειδή υπήρχε μεγάλη καθυστέρηση για την εύρεση του κατεστραμένου υλικού προσπαθήσαμε να βρούμε εναλλακτικούς τρόπους προσπέλασης του DSP από το Port B (Host Interface).

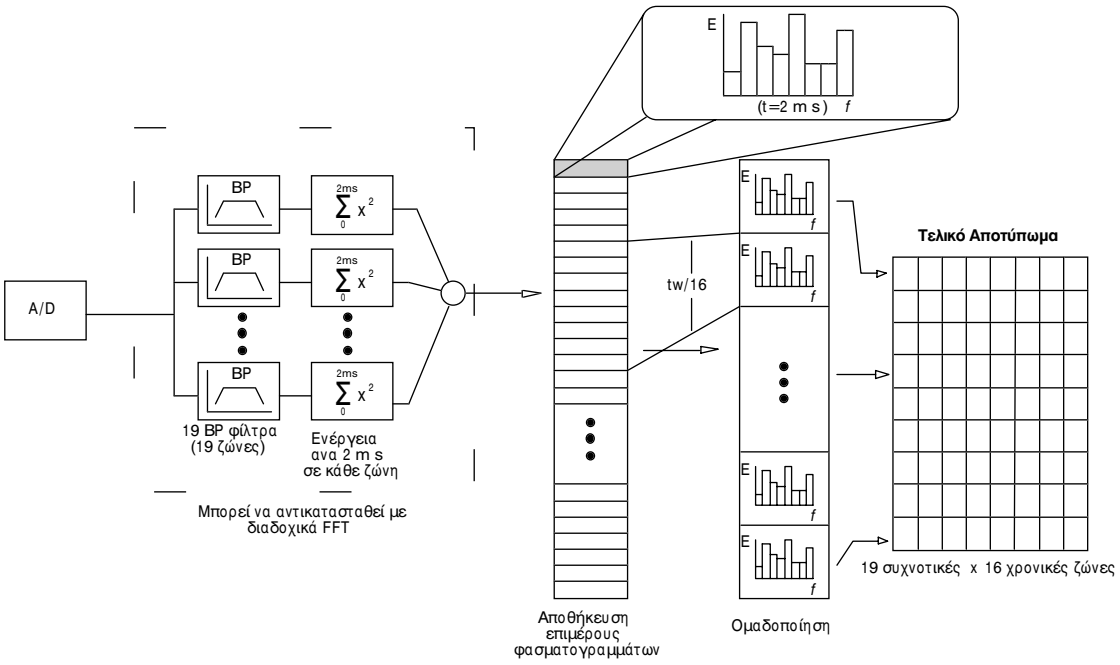
Όταν πήραμε στα χέρια μας το ολοκληρωμένο που θα αντικαθιστούσαμε την καμένη PROM, είχαμε μεγάλη αγωνία. Τελικά μετά από διάφορα μικροπροβλήματα που ξεπεράστηκαν σχετικά εύκολα μπορέσαμε να έχουμε το EVM πάλι σε πλήρη λειτουργία. Παράλληλα είχαμε σχεδόν έτοιμη την επέκταση μνήμης τόσο σε θεωρητικό επίπεδο όσο και σε σχέδιο πλακέτας. Η πλακέτα είχε σχεδιαστεί με τεράστια προσοχή χρησιμοποιώντας τις τελεύταιες γνώσεις που είχαμε εκείνη την εποχή πάνω σε φαινόμενα RF (Radio Frequency), γραμμών μεταφοράς, και υψηλών ταχυτήτων. Το μέγιστο πρόβλημα που θα αντιμετωπίζαμε σύμφωνα με τις απόψεις των ειδικών στη σχετική βιβλιογραφία, ήταν η πλακέτα διπλής όψης. Η διεθνής τακτική σε τέτοιες ταχύτητες (27MHz ρολόι) είναι η χρήση πλακέτας τεσσάρων επιπέδων με ground plane. Τελικά ευτυχώς για μας όλα δουλέψαν κανονικά έστω και σε δύο επίπεδα.

Η νέα μορφή

Ο πρώτος στόχος για την βελτιστοποίηση του αλγορίθμου, που ήταν απαιραίτητο να γίνει για να υλοποιηθεί η αναγνώριση στο DSP, ήταν να μειώσουμε τον αριθμό των δειγμάτων που κρατούσαμε στην μνήμη. Για να το κάνουμε αυτό, θα έπρεπε να βρούμε ένα τρόπο με τον οποίο θα μπορούσαμε να προβλέψουμε το μέγεθος της λέξης, ώστε να μπορέσουμε να ξεκινήσουμε να επεξεργαζόμαστε δείγματα πρίν από το τέλος της. Αυτό όμως είναι αδύνατο, και έτσι βρήκαμε μια άλλη λύση. Με την λογική οτι η πυκνή και πιστή παρακολούθηση των συχνοτήτων του σήματος εισόδου δέν μας είναι απαραίτητη, μιας και το τελικό αποτέλεσμα θα έχει μικρή ανάλυση, δηλαδή μόλις 16 χρονικές ζώνες, γίνεται φανερό οτι δέν έχει νόημα το να υπολογίζουμε με μεγάλη ακρίβεια το φασματόγραμμα στον άξονα του χρόνου. Άρα, μπορούμε να κάνουμε το εξής κόλπο.

Αντί να αποθηκεύουμε στην μνήμη τα δείγματα της λέξης, που πιάνουν πολύ χώρο επειδή έρχονται πολύ γρήγορα, περιμένουμε να μαζευτούν μερικά δείγματα που αντιστοιχούν σε μια χρονική διάρκεια μερικών msec. Επειδή η ομιλία και οι φθόγγοι δέν αλλάζουν πολύ γρήγορα, παρά μόνο κάθε αρκετά msec, στην ομάδα των δειγμάτων που θα μαζευτεί θα υπάρχει το χαρακτηριστικό περιεχόμενο συχνοτήτων σ' αυτό το χρονικό διάστημα, χωρίς να έχουν αναμιχθεί οι φθόγγοι μεταξύ τους. Άν προσέξουμε, θα δούμε οτι μας συμφέρει καλύτερα, από την πλευρά του αποθηκευτικού χώρου, να αποθηκεύουμε όχι τα δείγματα της χρονικής αυτής περιόδου, αλλά το συχνοτικό περιεχόμενό τους, δηλαδή το φάσμα τους. Μάλιστα, το φάσμα που αποθηκεύουμε είναι ήδη επεξεργασμένο και είναι χωρισμένο στις 19 συχνοτικές ζώνες που μας ενδιαφέρουν.

Για να δούμε ποιό είναι το κέρδος με αυτόν τον τρόπο, ας υποθέσουμε οτι δειγματοληπτούμε με συχνότητα 10KHz. Με αυτό τον ρυθμό, σε 20ms θα έχουν μαζευτεί 200 δείγματα. Κάνοντας την απαραίτητη επεξεργασία στο διάστημα αυτό των 20ms, το αποτέλεσμα που παράγεται είναι 19 αριθμοί που αντιπροσωπεύουν την ενέργεια του σήματος, στην διάρκεια αυτών των 20ms, σε κάθε μια από τις 19 συχνοτικές ζώνες που μας ενδιαφέρουν. Έτσι, μπορούμε να αποθηκεύσουμε μόνο τους 19 αριθμούς, έναντι των 200 που ήταν αρχικά, σαν δείγματα, πρίν από την επεξεργασία. Ήδη, με αυτό τον πρόπο πετύχαμε μια συμπίεση περίπου 95%, και μάλιστα χωρίς να χάσουμε σημαντική πληροφορία. Τώρα, το μόνο που μένει, είναι να αποθηκεύουμε διαδοχικά τα ενδιάμεσα αυτά αποτελέσματα για όσο διάστημα διαρκεί η λέξη, και όταν τελειώσει, μπορούμε να τα ομαδοποιήσουμε στις 16 χρονικές ζώνες που μας χρειάζονται. Η μέθοδος αυτή πετυχαίνει μεγαλύτερη συμπίεση όσο αυξάνεται το "χρονικό κβάντο", δηλαδή το μικρότερο διάστημα που επεξεργαζόμαστε, και όσο αυξάνεται η δειγματοληψία.



***Σχ. 4.4. Η** νέα μορφή του αλγορίθμου.*

Η επιλογή των χρονικών διαστημάτων δέν είναι τυχαία, αλλά προκύπτει από τα χαρακτηριστικά της ομιλίας. Κάθε φθόγγος αποτελείται από μερικές επαναλήψεις μιας περιοδικής κυματομορφής. Η βασική περίοδος είναι ίση με μερικά ms, δηλαδή την περίοδο που αντιστοιχεί στις θεμελιώδεις συχνότητες που παράγονται στον λάρυγγα. Η διάρκεια των επαναλήψεων, εξαρτάται από την ταχύτητα ομιλίας. Στην συνηθισμένη ομιλία ποικίλει από 30ms έως 50 ms. Άρα, με ένα "παράθυρο" των 10-20ms, μπορούμε να "φωτογραφίσουμε" την ενέργεια ενός περίπου φθόγγου. Η δυνατότητα αυτή μας δίνει και την βάση για μελλοντικές επεκτάσεις του αλγορίθμου σε επίπεδο φθόγγων.

Με αυτή, λοιπόν, την μέθοδο, καταφέραμε να ελαχιστοποιήσουμε τις απαιτήσεις του συστήματος από την πλευρά της μνήμης, κάνοντας μια παράλληλη επεξεργασία και στα δύο πεδία, τόσο του χρόνου, όσο και της συχνότητας. Η μόνη ερώτηση που είχαμε τώρα, ήταν άν θα χρησιμοποιούσαμε FFT ή μια ομάδα από BP ψηφιακά φίλτρα για να εξάγουμε τα επιμέρους φασματογράμματα, στα μικρά αυτά διαστήματα των 10-20ms. Μετά από δοκιμές, καταλήξαμε οτι τα FFT στο μέγεθος που μας ενδιέφερε (256 σημεία), μας έδιναν ικανοποιητική ανάλυση, και πολύ μεγαλύτερη ταχύτητα, μιας και το FFT επεξεργάζεται τα δείγματα σε ομάδες, ενώ τα φίλτρα επεξεργάζονται ανά δείγμα. Επειδή χρειαζόμαστε πολύ απότομες μεταβάσεις από την μιά περιοχή στη άλλη, δηλαδή φίλτρα υψηλού βαθμού (IIR από 10 έως 14), και έπρεπε να έχουμε 19 τέτοια φίλτρα, ο όγκος των υπολογισμών για τα φίλτρα είχε πλησιάσει το μέγιστο που μπορεί να δώσει ο 56000. Έτσι, αποφασίσαμε να δοκιμάσουμε με το FFT.

Μήκος FFT: 256

Αρχικά όρια		Ανάλυση	Τελικά όρια				Σημεία ζώνης	
Αρχή	Τέλος	FFT	Αρχή	% σφάλμα	Τέλος	% σφάλμα	Από σημείο	Αριθ. σημείων
180	300	39.0625	195.3	8.507	312.5	4.167	5	8
300	420	39.0625	312.5	4.167	429.7	2.307	8	11
420	540	39.0625	429.7	2.307	546.9	1.273	11	14
540	660	39.0625	546.9	1.273	664.1	0.616	14	17
660	780	39.0625	664.1	0.616	781.3	0.160	17	20
765	915	39.0625	781.3	2.124	898.4	-1.810	20	23
925	1075	39.0625	937.5	1.351	1093.8	1.744	24	28
1075	1225	39.0625	1093.8	1.744	1210.9	-1.148	28	31
1225	1375	39.0625	1210.9	-1.148	1367.2	-0.568	31	35
1375	1525	39.0625	1367.2	-0.568	1523.4	-0.102	35	39
1525	1675	39.0625	1523.4	-0.102	1679.7	0.280	39	43
1700	1900	39.0625	1718.8	1.103	1914.1	0.740	44	49
1900	2100	39.0625	1914.1	0.740	2109.4	0.446	49	54
2100	2300	39.0625	2109.4	0.446	2304.7	0.204	54	59
2300	2500	39.0625	2304.7	0.204	2500.0	0.000	59	64
2600	2800	39.0625	2617.2	0.661	2812.5	0.446	67	72
2850	3150	39.0625	2851.6	0.055	3164.1	0.446	73	81
3150	3450	39.0625	3164.1	0.446	3437.5	-0.362	81	88
3500	4000	39.0625	3515.6	0.446	3984.4	-0.391	90	102

Σχ. 4.5. *Η φασματική ανάλυση ενός FFT και η αντιστοιχία του με τις φασματικές περιοχές που χρησιμοποιούμε.*

Επιλέξαμε να έχουμε FFT 256 σημείων, μιας και με αυτό το μέγεθος μπορούσαμε να έχουμε φασματική ανάλυση ικανή να μας επιτρέψει να διαχωρίσουμε τις ζώνες που μας ενδιαφέρουν. Η συχνότητα δειγματοληψίας είναι 10kHz, και το FFT επαναλαμβάνεται κάθε 128 δείγματα εισόδου, ώστε να έχουμε επικάλυψη και να μήν χάνονται χρήσιμες πληροφορίες που μπορεί να τύχει να βρεθούν στις περιοχές που τα παράθυρα έχουν μεγάλη απόσβεση, με αποτέλεσμα να μήν συμμετέχουν στο αποτέλεσμα.

Τα προγράμματα του DSP

Το πρόγραμμα του DSP είναι μια ολοκληρωμένη εφαρμογή που αναλαμβάνει να υλοποιήσει όλες τις "βαριές" λειτουργίες επεξεργασίας σήματος που περιλαμβάνει η αναγνώριση. Είναι φτιαγμένο με την δομή ενός πλήρους λειτουργικού συστήματος, με δυνατότητες task-switching, που κάνουν δυνατή την εκτέλεση πολλών εργασιών παράλληλα. Αποφασίσαμε να μήν χρησιμοποιήσουμε το DSP για την λειτουργία της αποθήκευσης της βάσης δεδομένων, γιατί το PC είναι πολύ πιό κατάλληλο για αυτή την δουλειά, και επιλέον, έχει την δυνατότητα να κάνει πολύ καλά γραφικά που θα μας βοηθούσαν στην εκτίμηση των αποτελεσμάτων.

Το λειτουργικό σύστημα

Για να γράψουμε το πρόγραμμα, ξεκινήσαμε απο τα πιό βασικά σημεία, δηλαδή την κατασκευή ενός βασικού πυρήνα λειτουργικού συστήματος, που θα επιτρέπει την εκτέλεση (και το γράψιμο) της εφαρμογής χωρίς να μας απασχολούν διάφορες παράμετροι του συστήματος, όπως τα interrupts ή η αρχικοποίηση των περιφερειακών, οι επικοινωνίες κλπ. Το λειτουργικό σύστημα που γράψαμε, έχει προγραμματιστικό interface για την δειγματοληψία και την αναπαραγωγή ενός σήματος μέσω του EVM, χρησιμοποιώντας FIFO buffer, την επικοινωνία με κάποιον host computer (PC) μέσω

της σειριακής, πάλι χρησιμοποιώντας FIFO, την επικοινωνία με το προγραμματιζόμενο FPGA, που χρησιμοποιείται για την αλληλεπίδραση με το χρήστη την ρύθμιση διαφόρων παραμέτρων του συστήματος και την απεικόνιση της κατάστασής του, και την κυρίως εφαρμογή που χρησιμοποιεί όσους απο τους πόρους του συστήματος που απαιτεί. Επίσης, τα διάφορα σφάλματα του συστήματος αναφέρονται με ειδικούς κωδικούς, απο την σειρική θύρα, στο PC. Έτσι με την βοήθεια ενός απλού προγράμματος για εξομοίωση τερματικού (πχ Telix), μπορούμε όχι μόνο να παρακολουθούμε την κατάσταση του συστήματος, αλλά να του δίνουμε και εντολές προς εκτέλεση, πχ. για την ρύθμιση των παραμέτρων που αφορούν τα κατώφλια της ενέγειας που αποφασίζουν άν έχουμε αρχή/τέλος λέξης.

Η δειγματοληψία

Η δειγματοληψία απο το DSP γίνεται με την χρήση interrupt. Στο interrupt αυτό, γίνεται άμεσα ο υπολογισμός των χαρακτηριστικών της φωνής, δηλαδή της μέσης ενέργειας, του μέσου πλάτους και του ρυθμού περασμάτων απο το μηδέν. Η ενέργεια και ο αριθμός περασμάτων χρησιμοποιούνται για να βρούμε την αρχή και το τέλος της κάθε λέξης. Κατά την εκτέλεση της δειγματοληψίας, γίνεται και η αποστολή των δειγμάτων στον D/A, άν έχει ζητηθεί απο την εφαρμογή. Στην έξοδο του D/A μπορούμε να δούμε το σήμα εισόδου, τις μεταβλητές του συστήματος (ενέργεια, πλάτος, zero crossing κλπ). Επίσης, άν οι μετρήσεις πάνω στα δείγματα που έρχονται δείξουν οτι βρέθηκε η αρχή ή το τέλος μιας λέξης, ενημερώνονται οι υπόλοιπες διεργασίες του συστήματος, ώστε να ενεργοποιηθούν, άν χρειάζεται. Στην περίπτωση που έχουμε μια λέξη σε εξέλιξη, το μήκος της ετριέται και μετά το τέλος της, γίνεται διαθέσιμο στις άλλες διεργασίες.

Η επεξεργασία σήματος

Μετά την δειγματοληψία, το δεύτερο βασικότερο πράγμα είναι η επεξεργασία του σήματος, ώστε να παραχθεί το φασματικό αποτύπωμα της λέξης. Το πρόγραμμα ελέγχει τον χρόνο στον οποίο μαζεύονται αρκετά δείγματα ωστε να γίνει ένας μετασχηματισμός FFT. Μόλις λοιπόν μαζευτούν αρκετά δείγματα, αντιγράφεται ένα τμήμα του σήματος εισόδου σε ένα ειδικό χώρο αποθήκευσης για να γίνει το FFT, αφήνοντας παράλληλα την δειγματοληψία να συνεχίζεται στο υπόλοιπο του αρχικού χώρου αποθήκευσης. Κατά την διάρκεια της αντιγραφής, γίνεται κλιμάκωση στα δεδομένα ωστε να μήν υπερχειλίσει το FFT, και εφαρμόζεται και ένα παράθυρο Hamming, για να μειώσουμε την διαρροή συχνοτήτων. η εκτέλεση των FFT γίνεται κάθε 128 δείγματα, και έχει μήκος 256 στοιχείων. Σε πραγματικό χρόνο, το FFT εκτελείται περίπου κάθε 13ms, και η εκτέλεσή του διαρκεί περίπου 820μS, μαζί με το χρόνο μεταφοράς των δειγμάτων στον buffer του FFT.

Η επεξεργασία των αποτελεσμάτων

Μετά απο το FFT, η έξοδος (spectrum) ομαδοποιείται στις 19 συχνοτικές μπάντες, και φυλάσσεται στον χώρο αποθήκευσης αποτελεσμάτων. Σε αυτό το σημείο, τα αποτελέσματα έχουν την μορφή "κβάντων"12ms, δηλαδή "φωτογραφίες" του συχνοτικού περιεχόμενου του σήματος εισόδου για κάθε διάστημα 12ms. Η μείωση στον απαιτούμενο αποθηκευτικό χώρο είναι τρομερή. Χρειάζονται μόνο 1500 λέξεις αποθήκευσης ανά δευτερόλεπτο, ενώ στο αρχικό σήμα χρειάζονται 10000 λέξεις ανα δευτερόλεπτο.

Όταν ανιχνευτεί και το τέλος της λέξης, τα δεδομένα ομαδοποιούνται στον άξονα του χρόνου για να σχηματίσουν 16 χρονικές ζώνες, και τελικά το αποτέλεσμα είναι ο πίνακας που είναι το αποτύπωμα της λέξης που θα αποθηκευτεί στην βιβλιοθήκη, ή θα συγκριθεί με αυτήν. Τα δεδομένα αποστέλονται στον host computer (PC) για την

σύγκρισή τους μς την βάση δεδομένων σε μορφή 24 bit, δηλαδή με μεγάλο δυναμικό εύρος. Άν απαιτείται, το PC μπορεί να μειώσει την ποσότητα της πληροφορίας, άν και αυτή η περίπτωση δέν είναι πολύ πιθανή.

Η αποστολή των αποτελεσμάτων

Η αποστολή των αποτελεσμάτων γίνεται με την σειριακή (RS232) θύρα του υπολογιστή, άν και είναι δυνατόν να "ψαρέψουμε" τα αποτλέσματα απο την μνήμη του DSP χρησιμοποιώντας το πρόγραμμα του ADS, και την βοήθεια κάποιων μακροεντολών που γράψαμε γι' αυτό το σκοπό. Στο PC γίνεται η σύγκριση των αποτελεσμάτων με τα πρότυπα της βιβλιοθήκης, η εκπαίδευση του συστήματος, και τελικά, η χρήση του τελικού αποτελέσματος.

Αναλυτική Περιγραφή

Στο υπόλοιπο μέρος του κεφαλαίου, θα ασχοληθούμε με τα επιμέρους τμήματα που απαρτίζουν το όλο έργο. Ο αναγνώστης παρακαλείται να ανατρέχει στο παράρτημα για τα σχέδια ή προγράμματα που αναφέρονται εδώ.

Κάρτα A/D-D/A για δίαυλο ISA

Προδιαγραφές

Η κάρτα αυτή είναι μια κατασκευή που έχει στόχο την δειγματοληψία αναλογικών σημάτων, και συγκεκριμένα ηχητικά σήματα από μικρόφωνο, καθώς επίσης και την αναπαραγωγή τους. Συγκεκριμένα η κάρτα κατασκευάστηκε για να μπορούμε να μετατρέψουμε το αναλογικό σήμα σε ψηφιακό έτσι ώστε να μπορέσουμε να το επεξεργαστούμε με τον Η/Υ.

Η κάρτα αυτή αποτελείται βασικά από τα ολοκληρωμένα που μετατρέπουν τα αναλογικά σήματα σε ψηφιακά και αντίστροφα. Μαζί με αυτά υπάρχουν και τα ψηφιακά ολοκληρωμένα που είναι απαραίτητα για την διασύνδεση της κάρτας με τον Η/Υ και την κεντρική μονάδα επεξεργασίας, ενώ δεν λείπουν και τα απαραίτητα αναλογικά κυκλώματα (ενισχυτές, φίλτρα).Για την συγκεκριμένη εφαρμογή (αναγνώριση φωνής) οι προδιαγραφές και οι απαιτήσεις για την κάρτα ήταν οι ακόλουθες.

Σχετικά καλή ακρίβεια

Η ακρίβεια που χρειαζόμαστε για σήματα φωνής, για να είναι αυτά κατανοητά από τον άνθρωπο και άρα να μην έχουμε χάσει πληροφορία που να μας είναι απαραίτητη για την αναγνώριση, είναι 48db (SNR).Επειδή οι περισσότεροι A/D στο εμπόριο, αν όχι όλοι, χρησιμοποιούν γραμμική διακριτοποίηση (Linear Quantasation) πρέπει να υπολογίσουμε τον λόγο σήματος προς θόρυβο ενός τυπικού A/D. Έστω ότι έχουμε για είσοδο ένα ημιτονικό σήμα. Η rms τιμή του είναι$V_{ref}/\sqrt{2}$. Το μέσο τετράγωνο του σήματος είναι $s^2=V_{ref}^2/2$.Το μέσο τετράγωνο του σφάλματος είναι

$$\varepsilon^2 = 2^{-2N} \cdot V_{ref}^{2/3}$$

όπου V_{ref} η τάση αναφοράς (μέγιστη τάση εισόδου $2 \cdot V_{ref}$) και N ο αριθμός των bits του μετατροπέα. Ο λόγος σήματος προς θόρυβο (Signal toNoise Ratio: SNR) είναι

$$SNR = \frac{\sigma^2}{\varepsilon^2} = 1.5 \cdot 2^{2N} \text{ ή}$$

$$SNR(db) = 1.76 + 6.02 \cdot N$$

Από αυτή τη σχέση παρατηρούμε ότι κάθε πρόσθετο bit του μετατροπέα βελτιώνει το λόγο SNR κατά 6DB. Αν πάρουμε τα 48 DB (δυναμική περιοχή - Dynamic range) και το διαιρέσουμε με τα 6DB βλέπουμε ότι απαιτούνται 8-Bit.

Εύρος ζώνης μέχρι 0-40KHz

Αρχικά για λόγους πειραματισμών και έρευνας για τις συχνότητες που δημιουργεί ο άνθρωπος κατά την ομιλία του, και να εξακριβωθεί το φάσμα συχνοτήτων του, απαιτούμε δειγματοληψία μέχρι και 40KHz. Κατόπιν ανακαλύψαμε ότι η απόκριση που χρειαζόμασταν ήταν το πολύ μέχρι και τα 5KHz.

Μανδαλωμένες εξόδους

Για να υπάρχει ευκολία σύνδεσης του A/D με τα ψηφιακά κυκλώματα ελέγχου/ μεταβίβασης δεδομένων θεωρήσαμε χρήσιμο ο A/D που θα διαλέγαμε να είχε μανδαλωμένες εξόδους (latched).

Λειτουργία με 0-5Vdc

Επειδή η κάρτα αυτή συνδέεται στο δίαυλο του PC (8-bit ISA), και για λόγους ευχρησίας θεωρήσαμε ότι θα ήταν καλύτερο να μην υπήρχε εξωτερική τροφοδοσία, οπότε θεωρούμε την λειτουργία με 5V ένα σημαντικό πλεονέκτημα.

Εύκολη διασύνδεση με επεξεργαστή

Ως γνωστόν ο δίαυλος εισόδου εξόδου του PC αλλά και κάθε μικροϋπολογιστή είναι βασικά μια προέκταση της κεντρικής μονάδας επεξεργασίας (CPU). Από εκεί βλέπουμε ότι θα μας διευκόλυνε η σύνδεση ενός A/D που έχει ενσωματωμένη λογική για διασύνδεση με επεξεργαστή έτσι ώστε να χρησιμοποιήσουμε όσο το δυνατόν λιγότερα εξαρτήματα, με αποτέλεσμα μικρότερο κόστος και μεγαλύτερη αξιοπιστία.

Γραμμική καμπύλη μετατροπής

Μας ενδιαφέρει η καμπύλη διακριτοποίησης να είναι όσο το δυνατόν πιό ευθύγραμμη. Ο A/D που επιλέξαμε είναι ο ADC0820 της National Semiconductor. Τα κύρια χαρακτηριστικά του είναι η μεγάλη ταχύτητα μετατροπής που έχει, ακρίβεια 8 bit, ενσωματομένο Track and Hold, τεχνική μετατροπής Half-Flash. Η τεχνολογία κατασκευής είναι CMOS, αλλά το interfacing είναι TTL-CMOS συμβατό. Ο χρόνος μετατροπής του είναι 1.5-2.5μS σε Read Mode και 1.1-1.52μS σε WR-RD mode. Πρέπει να προσεχθεί όμως ότι μετά από κάθε μετατροπή ο A/D πρέπει να κάνει εσωτερικό reset έτσι ώστε να αποφορτίσει τους πυκνωτές που έχει στούς συγκριτές. Ο χρόνος αυτός είναι 500nS και πρέπει να προστεθεί στο συνολικό χρόνο μετατροπής εφ' όσον κάνουμε επαναληπτική δειγματοληψία. Η κατανάλωση ισχύος είναι 75mW που είναι χαμηλή και βοηθάει στο να μην καταπονούμε το τροφοδοτικό του H/Y. Το συνολικό σφάλμα εξόδου είναι +/-1LSB.

Ο D/A που έχουμε χρησιμοποήσει είναι ανάλογος σε χαρακτηριστικά με τον ADC0820:

α. Έχει 8 Bit ακρίβεια άρα δεν χάνουμε σε ποιότητα εξόδου.
β. Έχει χρόνο αποκατάστασης της τάσης εξόδου (από τη στιγμή που θα δώσουμε σήμα στην είσοδο) 1μS.
γ. Έχει δύο εσωτερικούς καταχωρητες έτσι ώστε να κρατάει σταθερή την τάση εξόδου μέχρι να γραφτούν τα ψηφιακά δεδομένα στον ενδιάμεσο καταχωρητή, και να αποφύγουμε τους σπινθηρισμους (Glitches).
δ. Είναι τεχνολογίας CMOS και έχει χαμηλή κατανάλωση ισχύος.

Σχεδίαση

Ο A/D εργάζεται στο γρήγορο τρόπο μετατροπής (WR-RD Mode). Αυτό επιτυγχάνεται συνδέοντας τον ακροδέκτη Mode στο Vcc. Η κάτω τάση αναφοράς (αρνητικότερη τάση εισόδου που μπορεί να δεχτεί ο AD, V_{ref}-) είναι 0V (γείωση). Η πάνω τάση αναφοράς (μέγιστη τάση εισόδου, V_{ref}+) είναι 3-5V ρυθμιζόμενη από πολυστροφικό ποτενσιόμετρο. Πρέπει να προσεχτούν οι τάσεις V_{ref}-, V_{ref}+ να μην ξεπεράσουν πάνω από 0.6 Volt την γή ή την τάση τροφοδοσίας γιατί υπάρχει κίνδυνος να καταστραφεί ο A/D. Η τάση εισόδου δεν πρέπει να ξεπεράσει την V_{cc} γιατί υπάρχει σοβαρός κίνδυνος καταστροφής του ολοκληρωμένου. Η γραμμή /INT συνδέεται μέσω απομονωτή τριών καταστάσεων στη γραμμή διακοπών 7 της CPU. Ο ρόλος του

απομονωτή είναι να προσφέρει μεγαλύτερη ικανότητα οδήγησης ολοκληρωμένων από το δίαυλο (Fan-Out). Αυτός είναι και ο ρόλος των τριών απομονωτών 74HCT541.Με την σύνδεση του ακροδέκτη αυτού με την CPU μπορούμε να εκμεταλευτούμε την δυνατότητα χρήσης διακοπών από το λογισμικό ώστε να είναι δυνατόν να τρέχει η ρουτίνα εξυπηρέτησης της διακοπής παράλληλα με κάποιο πρόγραμμα.

Το CS του A/D συνδέεται με τον αποκωδικοποιητή διευθύνσεων της κάρτας. Η διεύθυνση του A/D είναι στην (A_2, A_1, A_0) = (000) από την βάση της κάρτας (Base Address). Οι γραμμές WR, RD χρησιμοποιούνται για την εκκίνηση της μετατροπής, και για το διάβασμα των δεδομένων από την CPU. Ο δίαυλος δεδομένων (8 Bit) συνδέεται με τον δίαυλο του PC μέσω ενός απομονωτή διπλής κατευθύνσεως, τριών καταστάσεων. Ο έλεγχος της φοράς των δεδομένων γίνεται παρακολουθώντας την γραμμή RD του διαύλου, ενώ η ενεργοποίηση του απομονωτή γίνεται μόλις ανιχνευτεί η βασική διεύθυνση της κάρτας. Η ανίχνευση της διεύθυνσης της κάρτας γίνεται με έναν συγκριτή λέξης των 8-Bit. Αυτός είναι ο 74HCT688 που συγκρίνει δύο λέξεις των 8 Bit και αν αυτές είναι ίδιες, τότε δίνει σήμα ισότητας στον απομονωτή των δεδομένων και στον αποκωδικοποιητή διευθύνσεων. Οι λέξεις που συγκρίνει είναι οι γραμμές διευθύνσεων A_3-A_{10} (8-Bit) από τον δίαυλο, και με pull-up αντιστάσεις και DIP switches την διεύθυνση αναφοράς. Η διεύθυνση αναφοράς είναι στη θέση 300 hex.Ο συγκριτής εργάζεται μόνο στον I/O χώρο των διευθύνσεων όταν δηλαδή δεν γίνεται DMA από το AEN (Address Latch Enable), και έχουμε είτε IORD ή IOWR (διάβασμα-εγγραφή σε περιφεριακή συσκευή). Έτσι έχουμε επιτύχει τη σωστή διευθυνσιοδότηση της κάρτας.

Το CS του D/A συνδέεται με τον αποκωδικοποιητή έτσι ώστε ο D/A να απαντά στην διεύθυνση (A_2,A_1,A_0)=(001) από την βασική διεύθυνση της κάρτας.Το ILE είναι συνδεδεμένο μονίμως σε λογικό 1. Ο ακροδέκτης αυτός χρησιμεύει σε περίπτωση αναβάθμισης του D/A από 8 σε 12 Bit αλλάζοντας το ολοκληρωμένο κύκλωμα. Εδώ δεν χρησιμοποιείται με αυτόν το σκοπό. Το σήμα XFER χρησιμεύει στο να επιτρέψει την ενεργοποίηση του σήματος WR2. Είναι μονίμως γειωμένο όπως και το WR1. Το σήμα WR1 χρησιμεύει στην μεταφορά των δεδομένων από τον δίαυλο στο πρώτο εσωτερικό καταχωρητή, ενώ το σήμα WR2 μεταφέρει τα δεδομένα από τον πρώτο εσωτερικό καταχωρητή στον δεύτερο άρα κατά συνέπεια και στην αναλογική έξοδο του D/A.Εδώ δουλεύουμε τον D/A σε Single-Buffer mode οπότε ο δεύτερος καταχωρητής είναι διαφανής (Transparent) με τον πρώτο. Το σήμα WR1 συνδέεται με το IOWR του διαύλου. Κάθε φορά που θα έχουμε WR από τον δίαυλο στην κάρτα στην διεύθυνση Base+1 τότε το Data Byte θα γράφεται στον D/A και θα βγαίνει στην έξοδο του. Τα σήματα DI (Data Input) συνδέονται με τον δίαυλο δεδομένων μαζί με τον A/D. Επειδή έχουμε φτιάξει κύκλωμα διευθυνσιοδότησης, δεν υπάρχει περίπτωση να υπάρχει σύγκρουση δεδομένων στον κοινό δίαυλο των AD/DA (Bus Conflict). Η τάση αναφοράς Vref είναι η τάση που τροφοδοτεί το δίκτυο των αντιστάσεων με ρεύμα. Αυτή η τάση στη κάρτα είναι 5V. Οι έξοδοι $I_{out}1$ και $I_{out}2$ είναι οι έξοδοι του D/A.

Οι D/A συνήθως έχουν ώς έξοδο ρεύμα και όχι τάση, σε αντίθεση με τους A/D που δέχονται στην είσοδο τους τάση και όχι ρεύμα. Έτσι απαιτείται να προσθέσουμε στην έξοδο του D/A ένα μετατροπέα ρεύματος σε τάση. Αυτός υλοποιείται με ένα τελεστικό ενισχυτή (κλασσικό κύκλωμα τελεστικού ενισχυτή). Η γραμμή R_{fb} συνδέεται εσωτερικά με μια αντίσταση, και έχει σκοπό να εξαλείψει την εξωτερική αντίσταση που απαιτείται από τον τελεστικό για να μετατρέψει το ρεύμα σε τάση. Δεν πρέπει να χρησιμοποιήσουμε εξωτερική αντίσταση αφού θα χάσουμε την ακρίβεια της μετατροπής. Αυτό συμβαίνει γιατί η εσωτερική αντίσταση παρουσιάζει τις ίδιες μεταβολές με τις εσωτερικές αντιστάσεις μετατροπής οπότε υπάρχει αντιστάθμιση στο σφάλμα.

Τα αναλογικά κυκλώματα εισόδου για τον A/D είναι ένας ενισχυτής με ενίσχυση 50. Αποτελείται από δύο τελεστικούς ενισχυτές. Η πρώτη βαθμίδα ενισχύει 5 φορές το αρχικό σήμα, με αναστρέφουσα συνδεσμολογία. Η δεύτερη βαθμίδα ενισχύει 10 φορές και είναι και αυτή αναστρέφουσα. Έτσι τελικά η διαφορά φάσης μεταξύ αρχικού σήματος, και σήματος εξόδου των τελεστικών είναι 360 μοίρες. Ο ενισχυτής έχει δύο βαθμίδες για την ενίσχυση, για την αποφυγή παρασιτικών θορύβων από υπερβολική ενίσχυση. Ο τελεστικός που χρησιμοποιούμε είναι ο LM741.

Κατόπιν ένα δικτύωμα RC παράγει ένα dc offset έτσι ώστε να ανυψώσει την τάση εισόδου κατά $V_{ref}/2$, όπου $V_{ref}=(V_{ref}+)-(V_{ref}-)$.Οι επόμενοι δύο τελεστικοί αποτελούν ένα βαθυπερατό φίλτρο τετάρτης τάξεως. Η συχνότητα αποκοπής είναι 10KHz, με κλίση 12db/oct. Το φίλτρο είναι τύπου Butterworth. Κατόπιν για την προστασία του A/D από υπερτάσεις υπάρχει η δίοδος Zener. Αν η τάση εισόδου ξεπεράσει την τάση Zener ή γίνει αρνητικότερη από την γή, τότε η Zener άγει και περιορίζει την τάση εισόδου. Οι αντιστάσεις χρησιμοποιούνται για να περιορίσουν το ρεύμα σε χαμηλά επίπεδα, στην περίπτωση σφάλματος, οπότε δέν θα υπερθερμανθεί η Zener από το ρεύμα που θα περάσει από μέσα της. Κατά την κατασκευή της πλακέτας πρέπει να προβλεφτούν πολύ καλές γειώσεις για να μην υπάρχουν προβλήματα θορύβων. Επειδή ο Η/Υ είναι ένα ψηφιακό σύστημα, υπάρχουν στιγμιαία μεγάλες απαιτήσεις ρεύματος από τα κυκλώματα κατά την μεταγωγή κξαταστάσεων από 1 σε 0 και αντίστροφα. Αυτό συνεπάγεται θορυβώδεις γραμμές παροχής ισχύος. Γι' αυτό το σκοπό οι πλακέτα έχει πολλούς πυκνωτές στις γραμμές τροφοδοσίας, κυρίως 100nF για τις υψηλές συχνότητες και 10μF/16V τανταλίου (ή μεγαλύτερους) για να μην υπάρχουν μεγάλες βυθίσεις τάσης. Έτσι αποφεύγονται κάπως οι πιθανές εστίες θορύβων που θα επηρεάσουν τα ευαίσθητα αναλογικά κυκλώματα. Η διαδικασία μετατροπής από πλευράς λογισμικού είναι (η διεύθυνση της κάρτας στη θέση Base):

Χωρίς χρησιμοποίηση διακοπών

α. Κάνε WR στην διεύθυνση Base+000 (A/D addr. space)

```
OUT BASE+000,XX  (XX=οτιδήποτε)
```

β. Περίμενε για 500nS. (συνήθως με εκτέλεση μερικών εντολών NOP).

```
NOP
NOP
```

(ανάλογα με το μηχάνημα που τρέχουμε προσθέτουμε και NOP. Ο πιο σωστός τρόπος είναι να έχουμε Timer)

γ. Κάνε RD (διάβασε τα δεδομένα)

```
IN DX,BASE+000          (DX=καταχωρητής δεδομένων)
```

Με χρησιμοποίηση διακοπών

α. Πρώτα θέτουμε στην διεύθυνση εξυπηρέτησης διακοπών το διάνυσμα που δείχνει στην ρουτίνα δειγματοληψίας. Πρέπει να προσέξουμε να δούμε αν υπάρχουν και άλλες ρουτίνες εξυπηρέτησης διακοπών στη διακοπή 7 που χρησιμοποιούμε. Αν υπάρχουν και άλλες ρουτίνες τότε

πρέπει να φτιάξουμε ένα πρόγραμμα που να κάνει polling η κάτι αντίστοιχο και να αναγνωρίζει ποιά συσκευή έκανε διακοπή.

β. Από το κύριο πρόγραμμα, κάνε WR στην διεύθυνση Base+000

```
OUT BASE+000,XX      ;A/D addr   space, XX=data
```

γ. Συνέχισε την εκτέλεση του προγράμματος

δ. Διακοπή (Interrupt). Το κυρίως πρόγραμμα διακόπτεται. Ρουτίνα εξυπηρέτησης διακοπής:

```
IN DX,BASE+000        ;Κάνε RD. (διάβασε τα δεδομένα)
                      ;DX=καταχωρητής δεδομένων
MOV #FFFFH,DX         ;αποθήκευσε το δεδομένο
RETI              ;επέστρεψε από την διακοπή
```

ε. Επιστροφή στο κυρίως πρόγραμμα.

```
.....
.....
MOV DX,#FFFFH              ;χρησιμοποίησε το δεδομένο
.....
```

Αντίστοιχα για τον μετατροπέα από ψηφιακό σε αναλογικό είναι (η διεύθυνση της κάρτας στη θέση Base)

α. Κάνε WR στην διεύθυνση Base+001 (D/A addr space)

```
OUT BASE+001,DATA          ;DATA= το δείγμα
```

Εκσφαλμάτωση

Τα προλήματα που παρουσίαζε η κάρτα μετά την κατασκεύη της ήταν θορύβος στη δειγματοληψία από τον A/D, δυσλειτουργίες στο κύκλωμα εισόδου και αρχικά, αποτυχία λειτουργίας του D/A

Θόρυβος στη δειγματοληψία του A/D

Αρχικά η κάρτα τοποθετήθηκε σε ένα slot επέκτασης XT-AT. Κατόπιν τρέξαμε το πρόγραμμα για την πραγματοποιήση της δειγματοληψίας και την απεικόνιση σε πραγματικό χρόνο των μετρήσεων. Εκεί παρατηρήσαμε ότι ακόμη και με κλειστό μικρόφωνο το σήμα εισόδου είχε πολύ μεγάλο θόρυβο (περίπου 4-5 bit). Για να ανακαλύψουμε την πηγή του θορύβου τοποθετήσαμε ένα παλμογράφο σε διάφορα καίρια σημεία του κυκλώματος. Ελέγξαμε την είσοδο του μετατροπέα, τους ενισχυτές, και τέλος την τροφοδοσία των 5V. Εκεί προς μεγάλη μας έκπληξη είδαμε η τροφοδοσία να έχει ταλαντώσεις περιοδικές από ψηφιακό θόρυβο. Αυτό δεν το περιμέναμε σε τόσο μεγάλο επίπεδο αφού η κάρτα είχε πάνω ηλεκτρολυτικούς πυκνωτές, ενώ τα περισσότερα ολοκληρωμένα κυκλώματα είχαν και τους απαραίτητους κεραμικούς των 100nF. Τελικά όμως όπως αποδείχτηκε στην πράξη αυτή ήταν η αιτία του προβλήματος. Δηλαδή τα ολοκληρωμένα που δεν είχαν κεραμικούς πυκνωτές είχαν μεγάλα ρεύματα διαμεταγωγής με αποτέλεσμα τους ψηφιακούς θορύβους στην γραμμή τροφοδοσίας των 5V. Αυτό είχε ως αποτέλεσμα την δυσλειτουργία του ADC λόγω μεταβολών στην Vcc και στις τάσεις αναφοράς (λάθος μετρήσεις). Ενδεικτικά αναφέρουμε κάποια ολοκληρωμένα που δεν είχαν πυκνωτές των 100nF, 74LS688 (address comparator - λειτουργεί συνέχεια, έχει πολύ συχνές μεταβολές στην έξοδο του), 74LS138 (address decoder λειτουργεί όταν επιλέγουμε την κάρτα) και 74LS08/32 (πύλες “κόλλα” - glue logic που ελέγχουν τους ADC/DAC).

Πρώτη μας δουλειά ήταν λοιπόν να τοποθετήσουμε σε σχεδόν όλα τα ολοκληρωμένα που δεν είχαν πυκνωτές 100nF, τους πυκνωτές αυτούς. Σιγά σιγά τοποθετώντας σε όλο και περισσότερα ολοκληρωμένα τους πυκνωτές αυτούς, ο θόρυβος που φαινόταν αρχικά άρχισε να εξαφανίζεται μέχρι που έπεσε μέσα στα επιτρεπτά όρια (λιγότερο από 1LSB). Κεραμικοί πυκνωτές τοποθετήθηκαν και σε μερικούς τελεστικούς. Επίσης για να σταθεροποιήσουμε ακόμη περισσότερο την τάση τροφοδοσίας τοποθετήσαμε και ένα πυκνωτή 10μF/16V τανταλίου. Πρέπει να σημειωθεί εδώ ότι οι ηλεκτρολυτικοί πυκνωτές που χρειάζονται τα ψηφιακά κυρίως κυκλώματα δεν έχουν τις καλύτερες δυνατές θέσεις πάνω στην κάρτα, διότι υπάρχουν αποστάσεις μεγαλύτερες από 2-3 εκατοστά από τη θέση του πυκνωτή ως προς το ολοκληρωμένο. Σε αυτά τα ολοκληρωμένα, κολλήσαμε τους πυκνωτές απευθείας πάνω στα ολοκήρωμένα.

Δυσλειτουργίες στο κύκλωμα εισόδου

Όταν λύθηκε το πρόβλημα του θορύβου η ομάδα ασχολήθηκε λίγο με την ρύθμιση της κάρτας για βέλτιστη λειτουργία. Αρχικά παρατηρήθηκε ότι το ποτενσιόμετρο που ελέγχει την ενίσχυση του σήματος εισόδου δεν “πατούσε” καλά στην κάρτα με αποτέλεσμα να μην δουλεύει πάντα σωστά. Αφού το επισκευάσαμε, είδαμε ότι παράλληλα με την ενίσχυση αυξανόταν και το DC offset στην είσοδο του ADC. Έτσι αποφασίστηκε να μπεί ένας πυκνωτής αποσύζευξης πριν από τον αθροιστή του dc offset ακριβώς μετά τον ενισχυτή εξόδου. Τέλος με άλλο πρόγραμμα ρύθμισης (calibrating) ρυθμίσαμε την κάρτα ώστε να έχει (χωρίς σήμα στην είσοδο) όσο το δυνατόν μικρότερη ενέργεια και zero crossing rate.

Αποτυχία λειτουργίας του D/A

Το μόνο πρόβλημα που απέμενε ήταν να λειτουργήσει το D/A. Το πρόβλημα με τα υβριδικά κυκλώματα κατά την εκσφαλμάτωση συνήθως είναι ότι ο μηχανικός δεν ξέρει αν το πρόβλημα είναι στο ψηφιακό ή στο αναλογικό μέρος. Αρχικά πήγαμε να εξετάσουμε το ψηφιακό τμήμα της κάρτας. Με τον παλμογράφο προσπαθήσαμε να δούμε τα σήματα ελέγχου πρός τον μετατροπέα DAC. Όμως ο παλμογράφος ήταν μέχρι τα 20MHz με αποτέλσμα κάποια σήματα να μην τα βλέπουμε ή να τα βλέπουμε αλλοιωμένα. Έτσι σύμφωνα με τις αρχικές παρατηρήσεις δεν βλέπαμε να γίνεται chip select ο DAC από το κύκλωμα ελέγχου. Ψάχνοντας στο κύκλωμα είδαμε ότι αρχικά είχε τοποθετηθεί μια αντίσταση pull-up στη γραμμή εξόδου από το HC688 προς το LS245 η οποία είχε τιμή μόλις 100Ω. Τελικά αφού διαπιστώσαμε ότι η ύπαρξη της εκεί ήταν άχρηστη, αφού το HC μπορούσε να οδηγήσει τα 2 TTL φορτία που είχε, την βγάλαμε από την κάρτα. Διαπιστώσαμε όμως ότι η αντίσταση που είχε πάνω η κάρτα ήταν από λάθος 100ΚΩ. Μετά την αλλαγή αυτή δεν δούλευε τίποτα στην κάρτα! Απελπισία προς στιγμήν, αλλά οι θετικές σκέψεις επικράτησαν. Κάνοντας ελέγχους διαπιστώσαμε ότι ξεκολλώντας την αντίσταση αυτή ξεχάσαμε να συνδέδουμε τα δύο επίπεδα του χαλκού (layers) των 5V στις δύο όψεις της πλακέτας, με αποτέλεσμα να μην υπάρχουν τροφοδοσίες 5V σε ορισμένα σημεία του κυκλώματος. Μόλις το φτιάξαμε αυτό ο ADC δούλεψε όπως και πριν. Αντίθετα ο DAC δεν μπόρεσε να μας ικανοποιήσει. Ξανακοιτάξαμε τα αναλογικά περιφεριακά του DAC αλλά όλα ήταν εντάξει. Επίσης αποφασίσαμε να απλοποιήσουμε το κύκλωμα εξόδου του DAC για καλύτερη λειτουργία. Αρχικά το κύκλωμα εξόδου περιελάμβανε κύκλωμα ρύθμισης πλήρους κλίμακας, κάτι που κατά τον κατασκευαστή δεν είναι απαραίτητο, και είναι καλύτερα να μην χρησιμοποιηθεί για να υπάρχουν μικρότερα σφάλματα. Με την απλοποίηση αυτή στην έξοδο παίρναμε -2.5V.

Τελικά μετά από μια προσεκτική μελέτη είδαμε ότι το D/A απαιτούσε πολύ μεγάλο χρόνο σταθεροποίησης του σήματος (setup time) στα δεδομένα, με αποτέλεσμα να μην προλαβαίνει την εγγραφή από το PC, μιας και ο δίαυλος του PC ήταν πολύ πιό γρήγορος. Έτσι προσθέσαμε έναν εξωτερικό μανδαλωτή, αρκετά γρήγορο για να παραλαμβάνει τα δεδομένα από το PC, οπότε ο D/A δούλεψε, και μάλιστα με τρόπο "flow-through", δηλαδή χωρίς να χρησιμοποιηθούν οι εσωτερικοί του μανδαλωτές.

Τα Προγράμματα για το PC

Στην ενότητα αυτή, θα παρουσιάσουμε, σε συντομία, τα πρώτα προγράμματα που γράφτηκαν για το PC. Επειδή αποτελούν μέρος του πρώτου μέρους της εργασίας, θα αναφερθούμε σε αυτά σύντομα, και μόνο στα σημεία που είναι σημαντικά

Δειγματοληψίες

Οι δειγματοληψίες στ PC απο την κάρτα εισόδου γινονται με την βοήθεια της βιβλιοθήκης SAMPLE.C. Η δειγματοληψία από την A/D κάρτα γίνεται με software trigger βασισμένο στο Timer interrupt του PC. Με την ίδια τεχνική γίνεται αναπαραγωγή των δειγμάτων από την παράλληλη θύρα με την χρήση DA με σκάλα αντιστάσεων R-2R.

Επεξήγηση

Η ‘’καρδιά” της βιβλιοθήκης είναι ο χειριστής διακοπής 0x1C του PC, που αντιστοιχεί στο IRQ0, δηλαδή έχει την μεγαλύτερη προτεραιότητα απ’ όλες τις άλλες διακοπές, εκτός του NMI. Κανονικά το interrupt αυτό καλείται 18,2 φορές το δευτερόλεπτο, αλλά επαναπρογραμματισμός του 8254A timer που έχει το PC επιτρέπει την ταχύτερη εκτέλεσή του.

Το πρώτο βήμα για την χρήση της βιβλιοθήκης είναι η αρχικοποίηση του timer και του interrupt handler. η αρχικοποίηση του timer γίνεται με την συνάρτηση SetTimer(), η οποία παίρνει σαν όρισμα τον επιθυμητό ρυθμό εκτέλεσης του interrupt (σε KHz). Η συνάρτηση επαναπρογραμματίζει τον timer έτσι ώστε η διακοπή 1C να εκτελείται με τον επιθυμητό ρυθμό. Αν έχει οριστεί η μεταβλητή του μεταγλωττιστή DEBUG, όπως και σε όλες τις άλλες βιβλιοθήκες, εκτυπώνονται στην οθόνη διαγνωστικά μυνήματα σχετικά με τη λειτουργία της κάθε συνάρτησης που χρησιμοποιείται.

Να σημειώσουμε εδώ ότι κατά την διάρκεια εκτέλεσης πολλών από τις συναρτήσεις αυτής της βιβλιοθήκης, τα interrupts είναι απενεργοποιημένα, διότι δεν μπορεί να εξασφαλιστεί αλλιώς η σωστή εκτέλεση των προγραμμάτων. Αυτό όμως το χαρακτηριστικό δεν θα πρέπει να απασχολεί τον χρήστη, μιάς και η λειτουργία των συναρτήσεων μένει “κρυφή” από αυτόν.

Το επόμενο βήμα για την χρήση της βιβλιοθήκης είναι η αντικατάσταση του παλιού interrupt handler με το νέο, που εκτελείται γρηγορότερα από τον παλιό. Η αντικατάσταση γίνεται αυτόματα με την χρήση της συνάρτησης HookInterrupt(). Ο νέος χειριστής αναλαμβάνει να καλεί τον παλιό χειριστή 18,2 φορές το δευτερόλεπτο. Αυτό είναι πολύ σημαντικό για την σωστή λειτουργία του συστήματος, διότι πολλές βασικές λειτουργίες στηρίζονται στην χρονικά σωστή εκτέλεση αυτού του interrupt.

Η λειτουργία του νέου χειριστή διακοπής NewTickHandler() έχει ως εξής: Αν έχει ζητηθεί να γίνει δειγματοληψία από τον χρήστη (μέσω της GetSample(), που θα δούμε παρακάτω) ή αναπαραγωγή ενός δείγματος από την κατάλληλη θύρα (με την PlaySample()), τότε ενεργοποιούνται τα flags ή OKToSample ή OKToPlay, αντίστοιχα. Έτσι, ο χειριστής αναλαμβάνει να φέρει εις πέρας τις αντίστοιχες λειτουργίες και στο τέλος του απενεργοποιεί τα flags αυτά.

Η διαδικασία για τη λήψη ενός δείγματος από τον A/D είναι η εξής:

1. Εγγραφή στη βασική διεύθυνση του A/D ενός τυχαίου αριθμού για την έναρξη της δειγματοληψίας.
2. Αναμονή (NOP).
3. Διάβασμα του δείγματος από τον A/D και μεταφορά του στην global μεταβλητή SampleToRead.
4. Απενεργοποίηση της σημαίας OKToSample, ώστε να μη ληφθεί άλλο δείγμα.
5. Ενεργοποίηση του σήματος Sample Ready, για να ενημερωθούν οι άλλες συναρτήσεις ότι υπάρχει διαθέσιμο δείγμα εισόδου.

Η αναπαραγωγή δείγματος είναι απλούστερη, μιας και αποτελείται από 2 μόλις βήματα:

1. Εξαγωγή του δείγματος (που πρέπει να βρίσκεται στην μεταβλητή SampleToPlay) στην παράλληλη θύρα, χωρίς παλμούς handshake.
2. Απενεργοποίηση της σημαίας OKToSample.

Τελευταία λειτουργία του χειριστή είναι, όπως είπαμε, η χρήση του παλιού interrupt 18,2 φορές το δευτερόλεπτο. ο χρόνος που μεσολαβεί μεταξύ δύο διαδοχικών κλήσεων του παλιού χειριστή διακοπής, υπολογίζεται με την βοήθεια ενός μετρητή (i) και μιας σταθεράς (RepeatRate) που περιέχει τον αριθμό των επαναλήψεων του νέου χεριριστή που πρέπει να γίνουν πρίν κληθεί ο παλιός χειριστής. Η σταθερά RepeatRate υπολογίζεται από την συνάρτηση SetTimer(), και η τιμή της γίνεται γνωστή στις άλλες συναρτήσεις. Επειδή ο νέος χειριστής καλείται σε κανονικά χρονικά διαστήματα, είναι εύκολο να υπολογίσουμε μετά από ποιό αριθμό επαναλήψεων πρέπει να καλείται ο παλιός χειριστής, ώστε να διατηρείται ο ρυθμός των 18.2 κλήσεων ανά δευτερόλεπτο.

Η χρήση των συναρτήσεων απλοποιείται με τις συναρτήσεις GetSample() και PlaySample(). Οι συναρτήσεις αυτές αναλαμβάνουν εξ' ολοκλήρου την επικοινωνία με τον χειριστή διακοπής που ενεργεί τις δειγματοληψίες, και έτσι ο χρήστης αρκεί να χρησιμοποιήσει μια εντολή του τύπου i=GetSample() ή PlaySample(i).

Να σημειώσουμε εδώ οτι η GetSample() παραλαμβάνει τα δείγματα από τον χειριστή σε προσημασμένη μορφή, δηλαδή στην περιοχή -127 έως +128. Η περιοχή αυτή αντιστοιχεί σε κλίμακα 0-5V, με το "μηδέν" περίπου στα 2.5V. Αυτό γίνεται διότι ο A/D μετατροπέας που χρησιμοποιείται στο υλικό που κατασκευάστηκε, έχει την δυνατότητα να δεχθεί μόνο θετικά σήματα, και μάλιστα στην περιοχή 0-5V. Για να μπορέσει να μετρήσει και AC σήματα, το hardware αναλαμβάνει να προσθέσει μια DC συνιστώσα 2.5V στο σήμα εισόδου, ώστε να μετατραπεί σε 0-5V. Μετά την δειγματοληψία, η πρόσθετη DC συνιστώσα αφαιρείται από τα δείγματα μέσα στον χειριστή διακοπής.

Η χρήση της βιβλιοθήκης αυτής περατώνεται με την κλήση των συναρτήσεων UnHookInterrupt() και RestoreTimer(), που αναλαμβάνουν να επιστρέψουν τον χρονιστή του συστήματος και το διάστημα της διακοπής 0x1C στις παλιές τους τιμές. Συνιστάται να καλούνται με τέτοια σειρά, ώστε να ενεργούν με αντίστροφη σειρά ως προς τις αντίστοιχες HookInterrupt() και SetTimer(). Δηλαδή να ακολουθείται η σειρά κλήσεως

SetTimer() - HookInterrupt() - ...(εφαρμογή)... - UnHookInterrupt() - RestoreTimer()

Σφάλματα

Η βιβλιοθήκη αυτή μπορεί να χρησιμοποιηθεί για συχνότητες δειγματοληψίας μέχρι 15KHz, περίπου. Πάνω από αυτή την συχνότητα, ο ρυθμός δειγματοληψίας δεν είναι σταθερός, αλλά παρουσιάζει αυξομειώσεις. Αυτές οφείλονται πιθανόνατα στην κλήση του παλιού χειριστή διακοπής, που προφανώς έχει χρόνο εκτέλεσης πάνω από (1/15)msec.

Πρέπει να ερευνηθεί η ορθότητα κλήσης του παλιού Interrupt handler σαν κανονική συνάρτηση σε C. Το πρόβλημα είναι ότι ίσως υπάρχει κάποιο IRET αντί για RET που θα επιθυμούσαμε στον παλιό Interrupt handler. Ο μόνος τρόπος να βρεθεί αυτό είναι με ανάλυση του κώδικα που παράγεται από το μεταγλωττιστή. Αν βρεθεί ότι πράγματι ο παλιός χειριστής επιστρέφει με εντολή IRET, πρέπει να :

1. Αντικατασταθεί η IRET με RET στον χειριστή, που είναι δύσκολο.
2. Να κληθεί ο παλιός χειριστής σαν διακοπή, είτε πειράζοντας την κανονική δομή του stack, είτε τοποθετώντας τον σε κάποιο από τα υπόλοιπα διανύσματα του interrupt table του PC, και καλώντας τον με την εντολή INTR.

Επειδή η βιβλιοθήκη καλεί τον παλιό χειριστή σε διαστήματα που είναι μόνο προσέγγιση του 18,2Hz , παρατηρείται το φαινόμενο να ολισθαίνει το ρολόι του PC. Συνήθως παρατηρείται επιτάχυνση, δηλαδή το ρολόι πάει μπροστά όσο χρησιμοποιείται αυτή η βιβλιοθήκη. Έτσι συνιστάται να μη μένει ανοιχτή περισσότερο απ' όσο χρειάζεται, δηλαδή να απελευθερώνεται το διάνυσμα 0x1C και να επαναφέρεται ο χρονιστής στην αρχική του κατάσταση το συντομότερο δυνατό.

Επίσης πρέπει να ερευνηθεί ο αριθμός των NOP που περιέχονται στον χειριστή διακοπής, για να δημιουργήσουν την καθυστέρηση που απαιτείται για να γίνει η μετατροπή. Ο αριθμός αυτός εξαρτάται από το μηχάνημα στο οποίο τρέχει το πρόγρμμα, μιας και ένας 8086 λειτουργεί σε χαμηλότερες ταχύτητες από έναν 386. Η βελτιστοποίηση του χρόνου αυτού εξαρτάται και από την ταχύτητα του διαύλου στο PC, αλλά έχοντας υπ' όψη ότι το όλο πρόγραμμα της αναγνώρισης θα μπορεί να τρέχει κατ' ελάχιστο σε 386DX/40MHz με μαθηματικό συνεπεξεργαστή, μπορεί να γίνει μια εκτίμηση για τον αριθμό των NOP που απαιτούνται.

Υποπρογράμματα διαχείρησης αρχείων

Το σύστημα αναγνώρισης φωνής χρειάζεται μια βάση δεδομένων, όπου θα αποθηκεύονται τα πρότυπα φασματογράμματα. Τα δεδομένα αυτά θα πρέπει να είναι εύκολα προσπελάσιμα, και να μπορούν να μεταβληθούν απλά και γρήγορα. Έτσι λοιπόν αποφασίστηκε να γραφεί λογισμικό που θα είναι υπεύθυνο για τη διαχείρηση αρχείων (εγγραφή, διάβασμα, σβήσιμο, ενημέρωση εγγραφής), πράγμα απαραίτητο για την πιο εύκολη ανάπτυξη της βάσης δεδομένων.

Η βιβλιοθήκη αυτών των συναρτήσεων περιέχει τις εξής λειτουργίες:
1. Δημιουργία νέου αρχείου
2. Αρχικοποιήση αρχείου (άνοιγμα)
3. Έλεγχος αρχείου
4. Γράψιμο/διάβασμα γενικών πληροφοριών αρχείου
5. Εγγραφή/Διάβασμα/Διαγραφή εγγραφής
6. Υπολογισμού νέας επικεφαλίδας
7. Ανανέωση αρχείου
8. Κλείσιμο αρχείου
9. Άνοιγμα/κλείσιμο βοηθητικού αρχείου

Το σύστημα διαχείρησης αρχείων αποτελείται από δύο τμήματα. Το πρώτο, που είναι το βασικό, ασχολείται με το αρχείο αποθήκευσης πληροφοριών σχετικές με την αναγνώριση φωνής (φασματογράμματα). Το δεύτερο τμήμα ασχολείται με ένα βοηθητικό αρχείο ascii όπου αποθηκεύεται η προτεραιότητα ψαξίματος των φασματογραμμάτων από τη βάση δεδομένων.

Η δομή του βασικού αρχείου είναι βασισμένη στον τύπο IFF (Interchangeable File Format). Ο τύπος αυτός επιλέχτηκε διότι επιτρέπει την εύκολη καταχώριση διαφόρων πληροφοριών, που βοηθούν στις διάφορες λειτουργίες που εκτελούνται πάνω στο αρχείο. Επίσης αργότερα αυτός ο τύπος μπορεί εύκολα να γίνει standard της αγοράς.

Πρώτα γράφεται η επικεφαλίδα αναγνώρισης του αρχείου. Αυτή αποτελείται από μια λέξη 32 bit (long word) με τα γράμματα ‘FORM’ όπου δηλώνεται ότι το αρχείο είναι τύπου IFF. Κατόπιν ακολουθεί άλλη μια λέξη 32-bit όπου γράφεται το μήκος του υπόλοιπου αρχείου.

Κατόπιν έρχεται η επικεφαλίδα γενικών πληροφοριών, η οποία αποτελείται από την λέξη των 32-bit ‘REPA’ ακολουθούμενη από το μήκος της εγγραφής αυτής (άλλα 32 bit). Σε αυτό την εγγραφή καταγράφονται γενικές πληροφορίες του αρχείου όπως συγγραφέας, έκδοση του τύπου (version) κ.α.

Στο πρώτο πεδίο υποδηλώνεται το μήκος (int, float κλπ.) της λέξης του φασματογράμματος. Δηλαδή αν αργότερα δούμε ότι η ακρίβεια του ακεραίου (int) στο φασματόγραμμα δεν μας ικανοποιεί, στο πεδίο αυτό θα γραφεί άλλη τιμή. Έτσι στο μέλλον το πρόγραμμα αναγνώρισης θα μπορεί να χειριστεί αρχεία αναγνώρισης και από παλαιότερες εκδόσεις.

Στο δεύτερο πεδίο γράφεται η έκδοση του αρχείου (version), και στο τρίτο πεδίο γράφεται το μήκος του πεδίου που αποθηκεύονται τα δεδομένα του φασματογράμματος. Έτσι μπορεί στο μέλλον να αποθηκεύονται μεγαλύτερα φασματογράμματα χωρίς να χρειάζονται αλλαγή οι ρουτίνες διαχείρησης αρχείων. Στο τέταρτο πεδίο γράφεται ο αριθμός των φασματογραμμάτων που υπάρχουν αποθηκευμένες στο αρχείο αλλά έχουν διαγραφεί “λογικά”. Όταν γίνει ανανέωση του αρχείου θα σβηστούν και πραγματικά.

Το πέμπτο πεδίο είναι δεσμευμένο για μελλοντική χρήση. Το έκτο πεδίο έχει αφεθεί ελεύθερο για πληροφορίες του χρήστη. π.χ. Αριθμός βιβλιοθήκης φασματογραμμάτων. Τέλος το έβδομο πεδίο περιέχει το όνομα του συγγραφέα του αρχείου αναγνώρισης. Έτσι κάθε αρχείο μπορεί να αναγνωριστεί σε ποιόν ανήκει.

Μετά από τις γενικές πληροφορίες γράφεται η επικεφαλίδα δεδομένων, η οποία είναι η λέξη 32 bit προσδιορισμού της επικεφαλίδας 'REPS', ακολουθούμενη από το μήκος των δεδομένων.

Το μήκος των δεδομένων που υπάρχει σε κάθε επικεφαλίδα είναι το μήκος της εγγραφής μείον το μήκος των 8 byte (32+32 bit= 64 = 8 · 8) που προσδιορίζουν την εγγραφή. Μετά ακολουθούν η μια πίσω από την άλλη οι δομές που περιέχουν τα φασματογράμματα. Οι δομές αυτές περιέχουν τα πεδία:

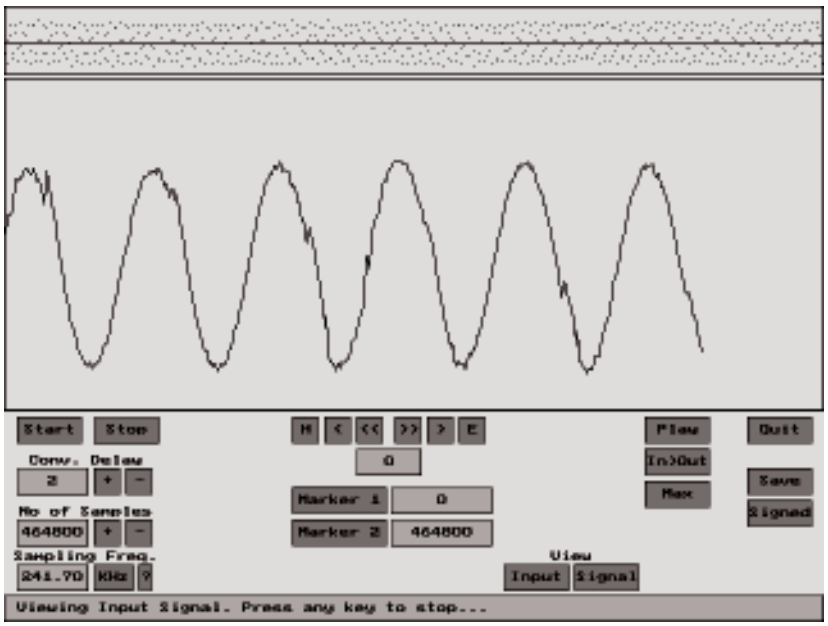
1. Ονομασία της λέξης. Εδώ περιέχεται η λέξη σε μορφή ascii, για να επιστραφεί από τον αλγόριθμο αναγνώρισης στο κύριο πρόγραμμα.
2. Πίνακας δεδομένων που περιέχει τα δεδομένα του φασματογράμματος με τη μορφή $(Freq_1,Time_1),(Freq_2,Time_1)$,.... $(Freq_{N-1},Time_M),(Freq_N,Time_M)$.
3. Στατιστικό στοιχείο της συχνότητας χρησιμοποιήσης της λέξης αυτής από το χρήστη.
4. Στατιστικό στοιχείο που προκύπτει από το προηγούμενο. Δηλώνει την προτεραιότητα που πρόκειται να παρθεί, για κάθε λέξη σε σχέση με τις άλλες κατά την διάρκεια της σύγκρισης. Χρησιμοποιείται για την αύξηση της ταχύτητας εύρεσης της εισερχόμενης λέξης από το σύστημα.

Προγράμματα εφαρμογών

Μεταξύ των διαφόρων προγραμμάτων που γράψαμε για το PC, είναι και τα ακόλουθα:

Τα προγράμματα δειγματοληψίας adc4.exe και adc6.exe, απεικονίζουν στην οθόνη την είσοδο απο την κάρτα, σάν σε οθόνη παλμογράφου. Το adc4 κάνει αργή απεικόνιση, ενώ το adc6 χρησιμοποιεί ενδιάμεσο buffer για να επιτύχει την μέγιστη ταχύτητα μετατροπής του A/D.

Το πρόγραμμα dsp2.exe είναι ένα ολοκληρωμένο γραφικό περιβάλλον, με την δυνατότητα χρήσης ποντικιού, για την δειγματοληψία και την αναπαραγωγή λέξεων. Το μέγιστο δείγμα που μπορούμε να πάρουμε με αυτό το πρόγραμμα περιορίζεται μόνο απο την ελεύθερη μνήμη του υπολογιστή (έχουμε πάρει και δείγμα μήκους 4MB). Η συχνότητα δειγματοληψίας είναι μεταβλητή, με την μέγιστη τιμή της να φτάνει τα 250kHz.



Σχ. 4.6. *Η οθόνη του προγράμματος dsp2.exe*

Το πρόγραμμα adcalib.exe είναι ένα πρόγραμμα διπλής χρήσης. Στην μιά χρήση του, δείχνει στην οθόνη τις μετρήσεις της μέσης ενέργειας, του μέσου πλάτους και του ρυθμού περάσματος απο το μηδέν για το σήμα εισόδου. Εκτός απο την χρήση του για την εκτίμηση των χαρακτηριστικών της φωνής, μπορεί να χρησιμεύσει και σαν πρόγραμμα ρύθμισης της ενίσχυσης και του offset των αναλογικών τμημάτων της κάρτας. Επίσης, μπορεί να γίνει και μια εκτίμηση για τον θόρυβο που προσθέτει η κάρτα στο σήμα εισόδου.

Τα προγράμματα specfast.exe και speclong.exe, κάνουν την απεικόνιση του σήματος εισόδου, και ταυτόχρονα την στιγμιαία ανάλυσή του κατα Fourrier. Η διαφορά των δύο προγραμμάτων είναι το μήκος του FFT που χρησιμοποιείται, που στο specfast είναι 256 δείγματα, ενώ στο speclong είναι 1024 δείγματα (είναι φανερό γιατί τα ονομάσαμε fast και long...).

Επίσης, χρησιμοποιήσαμε τα προγράμματα recon.exe και spectra.exe για να πάρουμε τα φασματογράμματα απο τις λέξεις που λέγαμε, στην οθόνη. Τα προγράμματα αυτά υλοποιούν το μεγαλύτερο, άν όχι όλο, μέρος του παλαιού αλγορίθομυ αναγνώρισης.

Κάρτα D/A για το DSP56001

Για λόγους εκμάθησης του DSP και την εξοικείωση μας με το ADS από πλευράς υλικού αποφασίσαμε να κατασκευάσουμε ένα μετατροπέα από ψηφιακό σε αναλογικό. Η κάρτα αυτή θα συνδεόταν με το ADS μέσω του συνδετήρα γενικών επεκτάσεων (Euroconnector 96 pin). Το DSP θα έβλεπε το περιφερειακό αυτό ως μια θέση μνήμης, δηλαδή το περιφερειακό D/A θα ήταν memory mapped. Ο D/A που διαλέξαμε είναι ο ICL7121. Αυτός είναι ένας 16 bit μετατροπέας της Harris semiconductor με μέγιστο χρόνο σταθεροποιήσης της εξόδου τα 3μS, με μικρό γραμμικό σφάλμα χάρις στο πίνακα PROM που μπορεί να προγραμματιστεί κατάλληλα για να το μειώσει στο ελάχιστο, και κατανάλωση 500mW.

Αναλυτική Περιγραφή

Η σχεδίαση του κυκλώματος μπορεί να χωριστεί σε τέσσερα τμήματα. Αυτά είναι:
1. Τα κυκλώματα απομόνωσης του διαύλου δεδομένων
2. Τα κυκλώματα ελέγχου
3. Τα κυκλώματα τροφοδοσίας
4. Τα αναλογικά κυκλώματα εξόδου και τα φίλτρα
5. Τάση αναφοράς

Τα κυκλώματα απομόνωσης

Τα κυκλώματα απομόνωσης έχουν σκοπό να αποσυμπλέξουν το δίαυλο του DSP από το περιφερειακό, όταν ο προηγούμενος δεν θέλει να επικοινωνήσει με το D/A αλλά με την εξωτερική μνήμη ή κάποιο άλλο περιφερειακό. Έτσι αποφεύγεται η περίπτωση της σύγκρουσης στο δίαυλο (bus conflict). Οι τρικατάστατοι απομονωτές δεν είναι άμεσα απαραίτητοι ως απομονωτές διότι το D/A έχει και αυτό εσωτερικά απομονωτές. Όμως τα κυκλώματα απομόνωσης δεν είναι απλές πύλες αλλά flip-flop με έξοδο απείρου αντίστασης. Τα flip-flop (74F374) έχουν στόχο την προσαρμογή του διαύλου του DSP στον D/A λόγω διαφορών στο χρονισμό τους και στην ταχύτητα προσπέλασης.

Η φιλοσοφία των κυκλωμάτων αυτών είναι να επιτρέψουν στο DSP να πραγματοποιήσει τους κύκλους εγγραφής με τη μέγιστη δυνατή ταχύτητα, χωρίς να παρεμβληθούν κύκλοι καθυστέρησης (wait states) ενώ παράλληλα να ικανοποιηθούν οι αργότεροι χρόνοι εγγραφής του D/A. Αυτό επιτυγχανεται με την μανδάλωση του διαύλου των δεδομένων του DSP από τα flip-flop αυτά κατά τον κύκλο εγγραφής, και το “πάγωμα” αυτών από τη μεριά του D/A για όσο χρόνο είναι αυτό απαραίτητο. Προσέξτε την οικογένεια των ολοκληρωμένων που χρησιμοποίηθηκε (σειρά TTL F - fast), ώστε να είναι δυνατή η μανδάλωση των δεδομένων λόγω ακόμα και στους πολύ μικρούς χρόνους σταθεροποίησης των δεδομένων πρίν και μετά το πέσιμο του παλμού του ρολογιού (setup και hold time) που έχει το DSP. Αντίστοιχα και η λογική ελέγχου έχει ως στόχο να αυξήσει όσο μπορεί τους χρόνους αυτούς ώστε να αποφευχθούν προβλήματα metastability.

Τα κυκλώματα ελέγχου

Τα κυκλώματα ελέγχου έχουν ως στόχο την υλοποίηση όλης της απαιτούμενης λογικής για τον πλήρη έλεγχο του ψηφιακού τμήματος της πλακέτας. Αυτό σημαίνει έλεγχο των απομονωτών/μανδαλωτών, και του χρονισμού των σημάτων εγγραφής στο D/A. Το περιφερειακό, όπως είπαμε και πριν, διευθυνσιοδοτείται στο χάρτη μνήμης του DSP (memory map).

Συγκεκριμένα επιλέξαμε την X περιοχή της μνήμης στη διεύθυνση 0xFFBx όπου x είναι οποιαδήποτε διεύθυνση από 0 ως F, δηλαδή η διεύθυνση του περιφερειακού είναι μερικώς αποκωδικοποιημένη από X:0FFB0 ως X:0FFBF. Αυτό το κάναμε διότι αφ' ενός η πλήρης αποκωδικοποίηση θα χρειαζόταν περισσότερο υλικό για να πραγματοποιηθεί και αφ' ετέρου η μερική αποκωδικοποιήση δεν μας καταλάμβανε πολύ μνήμη. Έτσι αποφασίσαμε αυτό το μοντέλο σχεδίασης. Ο έλεγχος γίνεται μέσω ενός ολοκληρωμένου κυκλώματος προγραμματιζόμενης λογικής (PLD). Αυτό μας εξοικονομεί χώρο στην πλακέτα, και παρέχει την δυνατότητα αναπρογραμματισμού αν αυτό απαιτηθεί (πχ αλλαγή διεύθυνσης, βοήθεια στην εκσφαλμάτωση κλπ) χωρίς να αλλάξει η πλακέτα, και μέγιστη ταχύτητα ανεξάρτητα, ως ένα σημείο, από τα επίπεδα λογικής που θα υλοποιηθούν.

Το πρόγραμμα του PLD, όπως φαίνεται στο παράρτημα των εγγράφων ανάπτυξης του υλικού, είναι πολύ μικρό. Η λειτουργία του είναι να μανδαλώσει τα δεδομένα του διαύλου δεδομένων του DSP, όταν προσπελαστεί η συγκεκριμένη θέση μνήμης (0xFFBx) σε κατάσταση εγγραφής, στη περιοχή των δεδομένων X. Τότε το PLD βγάζει ένα παλμό μανδάλωσης των δεδομένων (Latch) για τους μανδαλωτές (74F374). Παράλληλα ο παλμός αυτός ξαναμπάινει στο PLD για να καθυστερήσει κατά 25nS (τόσος είναι ο χρόνος διάδοσης pin-pin του PLD - propagation delay). Αυτό το κάνουμε για να προλάβουν τα δεδομένα του διαύλου να βγούν στην εξοδο των μανδαλωτών με κάποια ανοχή έτσι ώστε να εξασφαλιστούν οι χρόνοι σταθεροποίησης των δεδομένων για να μπορέσει να γίνει σωστά η εγγραφή στο D/A. Μόλις τα δεδομένα βγούν στην έξοδο των μανδαλωτών, σκανδαλίζεται ο μονοσταθής μονοδονητής για χρόνο ίσο με 250nS που ενεργοποιεί το σήμα εγγραφής στο D/A. Αυτός ο χρόνος δεν ρυθμίζεται με μεγάλη ακρίβεια διότι υπάρχουν ανοχές από τα RC κυκλώματα που ρυθμίζουν τη σταθερά χρόνου, αλλά είναι αρκετός για να γραφούν με ασφάλεια τα δεδομένα από τους μανδαλωτές στον D/A αφού ο ελάχιστος χρόνος που απαιτείται είναι 200nS. Ο μονοσταθής χρησιμοποιήθηκε γιατί το σήμα εγγραφής του DSP είναι πολύ γρήγορο για τον D/A ώστε να χρησιμοποιηθεί απ' ευθείας.

Το κύκλωμα τροφοδοσίας

Το κύκλωμα τροφοδοσίας έχει σκοπό την παραγωγή των απαραίτητων τάσεων που χρειάζονται για τη λειτουργία τα αναλογικά και τα ψηφιακά σήματα. Το κύκλωμα αυτό είναι ένα τροφοδοτικό με είσοδο DC τάση, με σκοπό να τη σταθεροποιήσει. Η αρχική η πρόβλεψη ήταν να χρησιμοποιηθεί ένα power pack τροφοδοτικό του εμπορίου που έχει μικρό όγκο και μέτρια σταθεροποίηση, για να τροφοδοτίσουμε την κάρτα αυτή, αλλά τελικά χρησιμοποίησαμε ένα μετασχηματιστή με μια γέφυρα πλήρους ανόρθωσης. Οι τάσεις που παράγονται είναι +/-12V και +5V. Ο λόγος που χρησιμοποιούμε εξωτερική τροφοδοσία αντί για αυτή που μας δίνει το ADS είναι ότι η τάση αυτή προέρχεται από το PC. Αυτό σημαίνει ότι υπάρχει πτώση τάσης στην καλωδιοταινία που τροφοδοτεί το ADS αφού οι αγωγοί δεν έχουν μεγάλη διατομή και υπάρχει μεγάλη κατανάλωση ρεύματος από πλευράς του ADS, οπότε με την παρουσία του παραπάνω φορτίου από τα κυκλώματα του D/A αυτή η πτώση τάσης θα είναι μεγαλύτερη. Αλλά πέραν τούτου όλοι γνωρίζουμε πόσο θορυβώδης είναι οι γραμμές τροφοδοσίας των ψηφιακών κυκλωμάτων λόγω των απότομων μεταβολών στις εισόδους και τις εξόδους τους. Στη συγκεκριμένη περίπτωση ο θόρυβος πρέπει να ελαχιστοποιηθεί διότι ο D/A έχει εύρος λέξης 16 Bit πράγμα που σημαίνει ότι έχουμε ανάλυση της τάξης των 80μV για τάση αναφοράς 2.5V!. Αν προσθέσουμε και το γεγονός ότι τα αναλογικά κυκλώματα χρειάζονται συμμετρικές τροφοδοσίες για να λειτουργήσουν έχουμε μια πλήρη εικόνα της κατάστασης.

Το τροφοδοτικό αποτελείται από δύο τμήματα τοποθετημένα σε σειρά. Το πρώτο τμήμα παράγει τις συμμετρικές τροφοδοσίες των +/-12V για τα αναλογικά κυκλώματα ενώ το δεύτερο τμήμα παράγει την τροφοδοσία των 5V από την έξοδο των +12V. Δεν συνδέσαμε τα τμήματα αυτά παράλληλα, αλλά σε σειρά, για να μην υπάρχει μεγάλη θερμική καταπόνιση του σταθεροποιητή των 5V λόγω της μεγαλύτερης πτώσης τάσης που θα είχε αφού η τάση εισόδου που θα δεχόταν θα ήταν τουλάχιστον 15V αντί για τα 12V που δέχεται τώρα. Προσέξτε και το πηνίο στη σύνδεση της γής του D/A με το ADS. Αυτό έγινε για να μειώσουμε τους υψίσυχνους θορύβους που μπορεί να έμπαιναν από τη γή του ADS πρός στο D/A. Επίσης μεγάλη προσοχή έχει δοθεί και στους πυκνωτές αποσύζευξης, ώστε οι γραμμές τροφοδοσίας να κρατούνται όσο το δυνατόν πιο ήσυχες.

Τα αναλογικά κυκλώματα εξόδου

Τα αναλογικά κυκλώματα εξόδου περιλαμβάνουν τους τελεστικούς ενισχυτές υποστήριξης του D/A, και το φίλτρο του post-filtering. Οι πρώτες μονάδες αποσκοπούν στην παραγωγή των τάσεων αναφοράς που χρειάζεται ο D/A σύμφωνα με τον κατασκευαστή για να λειτουργήσει. Πρόκειται βασικά για έναν αναστροφέα που παράγει την αρνητική τάση αναφοράς του D/A ώστε να μπορούν να παραχθούν θετικές και αρνητικές τιμές στην έξοδο του D/A και να υπάρχει συμμετρία. Αυτό απαιτεί μεγάλη ακρίβεια, γι' αυτό και οι αντιστάσεις καθορισμού της ενίσχυσης είναι ενσωματωμένες μέσα στο ολοκληρωμένο έτσι ώστε να υπάρχει και θερμική αντιστάθμιση του σφάλματος. Η εξοδος του D/A είναι όπως στους περισσότερους D/A, έξοδος ρεύματος. Για να μετατραπεί σε τάση η εξόδος αυτή πρέπει να χρησιμοποιηθεί ένας μετατροπέας από ρεύμα σε τάση. Ο πυκνωτής ανάμεσα στις δύο γραμμές εξόδου του D/A έχει σκοπό να μειώσει το φαινόμενο κουδουνισμού και τις ταλαντώσεις στην έξοδο, ενώ η δίοδος schottky εμποδίζει το ολοκληρωμένο να πάθει το φαινόμενο του latch-up αν η αρνητική έξοδος κατέβει κάτω από 0.4V από τη στάθμη της αναλογικής γης.

Τα φίλτρα

Η έξοδος του D/A περιέχει υψηλές συχνότητες λόγω της συνάρτησης μεταφοράς του (First Order Hold). Έτσι αν δώσουμε στην είσοδο του την ψηφιακή μορφή ενός ημιτόνου, στην αναλογική έξοδο του θα πάρουμε ένα ημίτονο όχι σε συνεχή μορφή, αλλά με τα γνωστά "σκαλοπάτια". Αυτά τα σκαλοπάτια περιέχουν συχνότητες πολύ μεγαλύτερες από το σήμα εξόδου που παράγουμε όταν η συχνότητα δειγματοληψίας απέχει πολύ απο τον μέγιστο ρυθμό εξόδου του D/A. Αυτό σημαίνει ότι μπορούμε να τις αποκόψουμε με ένα φίλτρο. Επιλέξαμε ένα φίλτρο γύρω στα 450KHz, για να μη χάσουμε τις δυνατότητες του D/A (πάνω από 300KHz) και παράλληλα να μην έχουμε πολύ ενέργεια στις υψηλές συχνότητες. Το φίλτρο που κατασκεύασαμε είναι ένα Butterworth 2ας τάξεως. Ο τελικός ενισχυτής εξόδου έχει ρύθμιση του offset και οδηγεί την αναλογική έξοδο της κάρτας. Οι τελεστικοί που έχουν χρησιμοποιηθεί είναι οι LF356 με εισόδους JFET με ευρύ φάσμα απόκρισης και μεγάλη ακρίβεια (80μV είναι αυτά...).

Τάση αναφοράς

Αυτό είναι ένα απλό κύκλωμα με μια zener αναφοράς (TL431) ρυθμισμένη στα 2.5V, με δυνατότητα μικρορύθμισης μέσω ενός τρίμμερ. Η σταθερότητα της τάσης αναφοράς είναι πολύ σημαντική, γι' αυτό και δεν χρησιμοποιήθηκε ένας σκέτος διαιρέτης τάσης μόνο με αντιστάσεις, αφού η παραμικρή μεταβολή στην τάση αναφοράς θα είχε άμεσες επιπτώσεις στην τάση εξόδου. Η τελευταία έχει άμεση εξάρτηση από την τάση αναφοράς.

Κατασκευή

Στην κατασκευή της πλακέτας πήραν μέρος τα μεγαλύτερα εργαστήρια του κόσμου όπως NASA-JPL,ESA, IRL, MOTOROLA κλπ.

Η σχεδίαση της πλακέτας είχε ιδιαίτερα χαρακτηριστικά λόγω των προβλημάτων που περιμέναμε να αντιμετωπίσουμε λόγω των πολύ μικρών τάσεων στην έξοδο και του θορύβου που θα υπήρχε. Οι γειώσεις είχαν τραβηχτεί προσεκτικά σε σύνδεση αστέρα (mecca ground ή star ground), δηλαδή η γη δεν συνδεόταν σε σειρά από κύκλωμα σε κύκλωμα, αλλά κάθε ομάδα κυκλωμάτων έπαιρνε τη γη του από τους σταθεροποιητές. Αυτό εξασφάλιζε την ελαχιστοποίηση του θορύβου στα αναλογικά κυκλώματα, αφού οι γραμμές επιστροφής δεν μοιραζόντουσαν μεταξύ των κυκλωμάτων, οπότε το επίπεδο του δυναμικού από ομάδα σε ομάδα στη γραμμή αναφοράς ήταν περίπου το ίδιο. Όπου υπήρχε ελεύθερος χώρος τοποθετήσαμε επιφάνεια γειωμένου χαλκού ώστε να δημιουργηθεί ένα ground plane.

Η διάταξη των κυκλωμάτων στην πλακέτα ήταν τέτοια ώστε να έχουμε ελάχιστη επαφή των αναλογικών με τα ψηφιακά κυκλώματα. Η διάταξη που δοκιμάσαμε ήταν να τοποθετηθούν τα ψηφιακά κυκλώματα όσο το δυνατόν πιο κοντά στο συνδετήρα του ADS, ώστε να έχουμε και ελαχιστοποίηση των αποστάσεων που διανύουν τα ψηφιακά σήματα από το ADS ώς την κάρτα του D/A. Αυτό το χρειαζόμαστε επειδή οι συχνότητες λειτουργίας του DSP είναι πολύ μεγάλες, με αποτέλεσμα αν δεν τηρούσαμε κάποιες βασικές αρχές σχετικά με γραμμές μεταφοράς και τερματισμούς, να αντιμετωπίζαμε αλλόκοτα προβλήματα, όπως πχ η τυχαία λειτουργία του D/A, που θα οφείλονταν σε ανακλάσεις των σημάτων και σε καθυστέρηση αυτών πάνω από τα επιτρεπτά όρια. Ελαχιστοποιώντας το μήκος των γραμμών, το μήκος κύματος των συχνοτήτων (η μέγιστη συχνότητα εξαρτάται από το χρόνο ανόδου/καθόδου - rise/fall time) θα ήταν μεγαλύτερο από το μήκος των γραμμών, με αποτέλεσμα την ασφαλή λειτουργία χωρίς την ανάγκη για τερματισμούς. Κατόπιν, το επόμενο κύκλωμα είναι ο D/A που είναι ανάμεσα στο ψηφιακά και στα αναλογικά κυκλώματα, αφού αυτός αποτελεί τη γέφυρα σύνδεσης των δύο κόσμων. Ο D/A δεν εργάζεται με την ίδια ταχύτητα που εργάζονται τα υπόλοιπα ψηφιακά κυκλώματα με αποτέλεσμα την πιο "ήσυχη" λειτουργία του σε σχέση με αυτά. Τέλος τα ευαίσθητα αναλογικά κυκλώματα είναι όσο το δυνατό περισσότερο απομακρυσμένα από τον συνδετήρα του ADS.

Αποφασίσαμε την κατασκευή της πλακέτας να την πραγματοποιήσουμε μόνοι μας, δοκιμάζοντας παράλληλα και τις δυνατότητες κατασκευής πλακετών διπλής όψης. Οι δυσκολίες που προέκυψαν ήταν σημαντικές με πιο σοβαρή τον χρόνο έκθεσης της κάθε όψης. Επειδή το φωτοευαίσθητο υλικό που ψεκάσαμε στην πλακέτα πριν την έκθεση στην υπεριώδη ακτινοβολία, δεν ήταν δυνατό να έχει το ίδιο πάχος και στις δύο όψεις της πλακέτας λόγω του ότι η διαδικασία γινόταν με το χέρι, ο χρόνος έκθεσης για τις δύο όψεις δεν ήταν ο ίδιος. Επίσης η ανομοιογένεια αυτή είχε ως αποτέλεσμα την ανομοιόμορφη εμφάνιση της κάρτας δηλαδή η μια όψη βγήκε υπερεμφανισμένη ενώ η άλλη βγήκε κανονική. Αυτό οφείλεται στο ότι η πλακέτα εμφανίζεται στο ίδιο διάλυμα και για τις δύο όψεις, οπότε αν υπάρχει διαφορετική ποσότητα φωτοευαίσθητης ουσίας στις δύο όψεις τότε ο εμφανιστής εμφανίζει περισσότερο τη μια από τις δύο όψεις.

Η κάρτα δεν είχε επιμεταλλωμένες τρύπες οπότε στη συναρμολόγηση έπρεπε να προσέχουμε να κολλάμε όσα εξαρτήματα χρειάζοταν και στις δύο όψεις της πλακέτας.

Εκσφαλμάτωση

Μετά τη συναρμολόγηση της πλακέτας σειρά είχαν οι δοκιμές. Αρχικά ο φόβος μας ήταν να μην προκληθεί ζημιά στο DSP από το ψηφιακό κομμάτι της πλακέτας λόγω κανενός βραχυκυκλώματος η λάθος σύνδεσης. Αφού λοιπόν συναρμολογήσαμε την πλακέτα ελέγξαμε με το πολύμετρο για σώστες συνδέσεις ή βραχυκυκλώματα. Ο πρώτος ενεργός έλεγχος ήταν η τροφοδότηση εν κενώ της κάρτας. Το εν κενώ σημαίνει χωρίς ολοκληρωμένα κυκλώματα (είχαμε τοποθετήσει μόνο τις βάσεις). Έτσι ελέγξαμε προβλήματα στη τροφοδοσία και στις γειώσεις. Κατόπιν τοποθετήσαμε τα αναλογικά κυκλώματα και επιβεβαιώσαμε εν μέρει την ορθή λειτουργία τους. Ύστερα προσθέσαμε και τα ψηφιακά κυκλώματα. Ελέγξαμε για διαρροή καπνού που θα εμπόδιζε τα κυκλώματα να λειτουργήσουν σωστά (ως γνωστόν τα ηλεκτρονικά εξαρτήματα δουλεύουν με καπνό και όχι με ηλεκτρόνια, αφού αν διαρρεύσει ο καπνός που περιέχουν παύουν να λειτουργούν). Αφού όλα πήγαιναν μια χαρά είπαμε να φτιάξουμε και εμείς τη μέρα μας και να συνδέσουμε την κάρτα του D/A πάνω στο ADS και η motorola βοηθός. Όντως το συνδέουμε στο συνδετήρα και βάζουμε μπρός τα γκάζια... Προς μεγάλη μας ευχαρίστηση όλα δουλεύανε ρολόι. Φτιάξαμε ένα δοκιμαστικό πρόγραμμα στο DSP (στη δεύτερη μητρική μας γλώσσα την assembly φυσικά) που έγραφε συνεχόμενα δύο λέξεις στη θέση μνήμης του περιφεριακού. Για να δοκιμάσουμε όλα τα bits οι λέξεις που επιλέξαμε ήταν 0x5A5A και 0xA5A5 όπως φαίνεται και στον κώδικα που παραθέτουμε.

```
        org p:$40
        move #$ffb0,r0
        move #$a5a5,x0
loop:
        move x0,x:(r0)
        move #$5a5a,x0
        move x0,x:(r0)
        move #$a5a5,x0
        nop
        nop
        nop
        nop
        jmp loop
```

Κατόπιν με τον παλμογράφο ελέγξαμε αν αυτά τα δεδομένα κλειδωνόντουσαν από τους μανδαλωτές, και αν ο χρόνος για το σήμα εγγραφής ήταν ο σωστός. Μετά από μια προσεκτική μελέτη όλα τα σήματα στον ψηφιακό τομέα φάνηκαν να δουλεύουν πολύ καλά. Σβήνουμε τα πάντα και τοποθετούμε τον D/A. Ξαναβάζουμε σε λειτουργία το σύστημα αλλά ο τελεστικός εξόδου δεν φάνηκε να ανταποκρίνεται. Μετά από μια προσεκτική μελέτη ανακαλύψαμε μια κυμάτωση 100Hz στην αναλογική τροφοδοσία. Αμέσως βγάζουμε το D/A και ξαναελέγχουμε, αλλά το πρόβλημα παρέμενε. Η πρώτη υποψία ήταν μήπως ο μετασχηματιστής τροφοδοσίας που είχαμε τοποθετήσει να μην έβγαζε το απαραίτητο ρεύμα πράγμα που με πρόχειρους υπολογισμούς δεν επιβεβαιώθηκε και πολύ. Η επόμενη κίνηση ήταν να βγάλουμε το PLD που καταναλώνει περίπου 100mA από μόνο του. Βέβαια ό χρόνος της όλης επιχείρησης ήταν Παρασκευή βράδυ (η χειρότερη ώρα δηλαδή για να κάνεις πειράματα) με αποτέλεσμα να σβήσουμε την τροφοδοσία του D/A, αλλά ξεχάσαμε να σβήσουμε το PC από όπου τροφοδοτείται το ADS με ρεύμα. Κατόπιν για κακή μας τύχη το PLD δεν

είχε κανονική βάση αλλά δύο βασοσειρές. Όταν λοιπόν το κατσαβίδι τοποθετήθηκε για να εξάγει το ολοκληρωμένο, βραχυκύκλωσε κάτι γραμμές διευθύνσεων του DSP. Μετά από λίγα δευτερόλεπτα συνηδητοποιήσαμε τι είχαμε κάνει αλλά ήταν πλέον αργά. Το ADS μας κρατούσε μούτρα και δεν απαντούσε. Μετά από αυτό αποφασίσαμε να ασχοληθούμε με το σοβαρό πρόβλημα που μας προέκυψε, την επιδιόρθωση του ADS. Ο πρώτος έλεγχος ήταν να ελέγξουμε το σήμα WR του DSP γιατί αυτό περνούσε κάτω από το PLD και θα πρέπει να βραχυκυκλώθηκε. Όντως το σήμα αυτό δεν φαινόταν να εργάζεται καθόλου. Όλα τα άλλα στον παλμογράφο φάνηκαν καλά.

Την επομένη το πρωί προς μεγάλη μας έκπληξη είδαμε το σήμα που την προηγουμένη μέρα δεν λειτουργούσε (είχε κολλήσει σε λογικό ένα) να εργάζεται κανονικά, ενώ αντίθετα το σήμα ανάγνωσης RD δεν εργαζόταν καθόλου. Αυτό δεν μπορούσε να εξηγηθεί με κάψιμο του DSP τουλάχιστον σε αυτά τα σήματα, διότι τα καμμένα ή δουλεύουν ή δεν δουλεύουν (ή μας δουλεύουν). Έτσι υποψιαστήκαμε τις μνήμες. Πράγματι αν αυτές είχαν πάθει βλάβη τότε ο επεξεργαστής θα εκτελούσε τυχαίο πρόγραμμα με αποτέλεσμα να κάνει ανεξήγητα πράγματα. Ο έλεγχος των μνημών (RAM/PROM) ήταν παιχνιδάκι. Αφαιρέσαμε όλα τα ολοκληρωμένα και τα τοποθετήσαμε στον προγραμματιστή που είχαμε και πραγματοποιήσαμε έναν έλεγχο μνήμης τυχαίας προσπέλασης. Όλες οι RAM πέρασαν τους ελέγχους. Κατόπιν ελέγξαμε τα περιεχόμενα των PROM. Εκεί προς μεγάλη μας έκπληξη (και χαρά που βρήκαμε κάποιο πρόβλημα) είδαμε ότι μια από τις PROM δεν την αναγνώριζε καθόλου ο προγραμματιστής. Ήταν σα να μην υπήρχε. Ελπίζοντας να μην είχε και άλλη παράλληλη βλάβη το σύστημα αρχίσαμε να ψάχνουμε στο εμπόριο για να την προμηθευτούμε.

Δυστυχώς δεν ήταν καθόλου εύκολο. Ότι και αν κάναμε δεν υπήρχε περίπτωση να μας φέρουν τέτοιες μνήμες (WSI, 2KByte, 45nS PROM) αφού ακόμα και ο αντιπρόσωπος της εταιρείας που τις έφερνε έπρεπε να παραγγείλει $1000 από την εταιρεία για να τα φερει. Τελικά βρήκαμε κάποιες αντίστοιχες από άλλη εταιρεία (Cypress 7C191/2/3) όπου μας τις έφεραν από το εξωτερικό χάρις στις φιλότιμες προσπάθειες ενός συναδέλφου από την εταιρεία που εργαζόμασταν παράλληλα. Τελικώς όταν τις πήραμε στα χέρια μας προγραμματίσαμε ένα κομμάτι αλλά ο προγραμματιστής έδειχνε λάθος στην επαλήθευση. Δοκιμάσαμε και ένα “τζάμι” (επανπρογραμματιζόμενο EPROM) αλλά το αποτέλεσμα το ίδιο. Αυτό μας άφησε άναυδους για λίγο αλλά αργότερα επικράτησε η λογική και είπαμε στον προγραμματιστή να διαβάσει τα περιεχόμενα της μνήμης. Τα διάβασε σωστά αποδεικνύοντας ότι το σφάλμα ήταν στο λογισμικό του προγραμματιστή, αφού προφανώς δεν μπορούσε να επιβεβαιώσει το σωστό προγραμματισμό των κυψελών EPROM. Επειδή τα ολοκληρωμένα είχαν άλλο πλαστικό περίβλημα (οι αρχικές ήταν dip 600mil ενώ οι καινούργιες ήταν dip 300mil) κάναμε μια προσαρμογή των ακροδεκτών ώστε να κολληθούν πάνω σε μια βάση 600mil, όπως και έγινε. Η προσαρμογή ήταν άψογη. Στην αρχή το ADS αρνήθηκε να δουλέψει, αλλά αυτό οφείλοταν στο γεγονός του ότι κάναμε δοκιμές σε άλλο mode λειτουργίας από το κανονικό με αποτελέσμα να μην τρέχει το πρόγραμμα του μόνιτορ. Μόλις επαναφέραμε τους βραχυκυκλωτήρες το DSP μας απάντησε όπως παλιά. Με τον καιρό όμως είχαμε ξεχάσει σε πιο σημείο της μνήμης έπρεπε να φορτώσουμε των κώδικα που δοκιμάζαμε, με αποτέλεσμα να φορτώνουμε τον δοκιμαστικό κώδικα στην περιοχή των διανυσμάτων διακοπών. Μόλις το “ανακαλύψαμε” και αυτό (μαζί με τον τροχό) όλα μπήκαν στην αρχική τους τροχιά.

Κάρτα διασύνδεσης μέσω του Host Interface στο DSP56001

Όσο διάστημα το ADS δεν μπορούσε να λειτουργήσει, προσπαθώντας να βρούμε το κατεστραμμένο εξάρτημα, ψάχναμε να βρούμε εναλλακτικούς τρόπους να συνεχίσουμε την ανάπτυξη του λογισμικού. Ο λόγος ήταν ότι εκτός από την ανάπτυξη του λογισμικού, θέλαμε να πραγματοποιήσουμε διαγνωστικά προγράμματα που θα μας επέτρεπαν και αργότερα με την επέκταση της μνήμης να δοκιμάσουμε το νέο, αδοκίμαστο, υλικό όπως και τώρα τον επεξεργαστή που δεν είμαστε σίγουροι για την καλή λειτουργία του. Η προφανής λύση ήταν το Host Interface. Αυτό πρόκειται για μια πόρτα του επεξεργαστή, που μπορεί να χρησιμοποιηθεί ώστε να κατέβει κάποιο κομμάτι κώδικα και να εκτελεστεί από την εσωτερική μνήμη του DSP. Με αυτό τον τρόπο δεν χρησιμοποιείται εξωτερική μνήμη και έτσι μπορούμε να τρέξουμε διαγνωστικά, και να πιάσουμε βραχυκυκλώματα η ανοικτοκυκλώματα στους εξωτερικούς διαύλους του επεξεργαστή.

Η πρόσβαση της πόρτας αυτής είναι πολύ εύκολη, αφού η πλακέτα του ADS διαθέτει ξεχωριστό βύσμα για την διασύνδεσή του, εκτός από την παράλληλη ύπαρξη των σημάτων και στο μεγάλο συνδετήρα γενικών επεκτάσεων. Αποφασίστηκε από την πλευρά του PC να χρησιμοποιηθεί η παράλληλη θύρα. Τα σήματα που έπρεπε να ελέγχουμε ήταν οι γραμμές δεδομένων, οι γραμμές διευθύνσεων των καταχωρητών, και οι γραμμές της επικοινωνίας χειραψίας.

Σχεδίαση

Οι γραμμές δεδομένων απομονώθηκαν με ένα 74LS245. Αυτός είναι ένας απομονωτής τριών καταστάσεων που εμείς των χρησιμοποιούμε ως ένα απλό ενισχυτή γραμμών. Στήν είσοδο του έχουμε τοποθετήσει αντιστάσεις pull-up για να βοηθήσει τους οδηγούς της κάρτας να οδηγήσουν το καλώδιο βελτιώνοντας τα ηλεκτρικά χαρακτηριστικά τους. Οι απομονωμένες γραμμές οδηγήθηκαν κατόπιν στις αντίστοιχες γραμμές δεδομένων του HI (Host Interface). Οι γραμμές ελέγχου του εκτυπωτή οδηγούν τις γραμμές επιλογής των καταχωρητών, ενώ για την επιλογή ανάγνωσης/εγγραφής χρησιμοποιούμε βραχυκυκλωτήρα αφού η διασύνδεση είναι απλή. Αυτό το κάναμε δίοτι δεν μας ενδιέφερε αρχικά η δυνατότητα να διαβάζουμε καταχωρητές, απλά να μεταφέρουμε τον κώδικα μας από το PC στο DSP. Τέλος το σήμα που έδινε την εντολή εγγραφής στο DSP ερχόταν κατεύθειαν από την παράλληλη πόρτα του PC και η μόνη βοήθεια που είχε ήταν μια αντίσταση πρόσδεσης με την τροφοδοσία (pull-up). Η τροφοδοσία ήταν εξωτερική με ένα μικρό τροφοδοτικό των 5V.

Εδώ θα υπενθυμήσουμε τον τρόπο μεταβίβασης του κώδικα χρήστη στο DSP από τη διασύνδεση αυτή. Το DSP πρέπει να πούμε εδώ ότι βρίσκεται στο Bootstrap mode (mode 1). Αυτό σημαίνει ότι μετά το reset ο επεξεργαστής θα πάει στη διεύθυνση P:$C000 για να δει την κατάσταση της γραμμής D23. Αν αυτή είναι σε λογικό ένα τότε διαβάζει μια εξωτερική μνήμη (ROM ή EPROM) και μεταφέρει τις πρώτες 512 λέξεις στην εσωτερική μνήμη. Αντίθετα αν αυτή η γραμμή είναι μηδέν τότε ο επεξεργαστής περιμένει δεδομένα από το HI. Ο τρόπος εγγραφής στο HI μπορεί να συνοψιστεί στα παρακάτω βήματα.

1. Θέσε στις γραμμές δεδομένων το πιο σημαντικό byte.
2. Επέλεξε την διεύθυνση του πρώτου καταχωρητή.
3. Δώσε ένα παλμό στο HEN.
4. Κάνε το ίδιο και με τα άλλα κομμάτια της λέξης (μεσαίο και λιγότερο σημαντικο byte).
5. Για να λήξει η μεταφορά ή γράφουμε 512 λέξεις (24 bit) ή θέτουμε σε λογικό ένα το bit HF0 στη διεύθυνση 00 του HI.

Σημειώστε εδώ ότι η γραμμές των δεδομένων είναι πλάτους 8 bit ενώ οι λέξεις του DSP έχουν πλάτος 24. Η μεταφορά γίνεται ανά οκτάδα bit με τελευταία το λιγότερο σημαντικό byte. Μόλις γραφεί το λιγότερο σημαντικό byte το DSP διαβάζει ολόκληρη τη λέξη από τους καταχωρητές. Έτσι πρακτικά η εγγραφή της λέξης ολοκληρώνεται με την εγγραφή της λιγότερο σημαντικής οκτάδας δεδομένων.

Εκσφαλμάτωση

Θέσαμε το DSP σε κατάσταση λειτουργίας Bootstrap με λίγο υλικό από τη θύρα γενικών επεκτάσεων αν και αυτό θα μπορούσε να είχε γίνει με την επιλογή των αντίστοιχων βραχυκυκλωτήρων (JG4 και JG5 στη θέση 2-3).

Το ADS δεν δούλευε, λόγω της βλάβης του, οπότε οι δοκιμές μας γίνοταν στα τυφλά! Στην αρχή κάτι φάνηκε να γίνεται αλλά το DSP δεν συμπεριφερόταν όπως έπρεπε. Λίγο οι αμφιβολίες για την καλή κατάσταση του, λίγο η αδυναμία να ελεγχθεί το οτιδήποτε δεν καταφέραμε και πολλά πράγματα.

Μόλις το ADS επισκευάστηκε ήταν αρκετά εύκολο να δούμε τι γινόταν. Γράψαμε μια παραλλαγή του αντίστοιχου bootstrap κώδικα της Motorola ώστε να δουλεύει σε κανονική λειτουργία (mode 2) παράλληλα με το μόνιτορ του ADS. Ο κώδικας αυτός είναι ο παρακάτω:

```
; Code to boot from host interface
;run this from ADM  (mode 2)
     org p:$40
     move #$ffe9,r2
     move #$200,r0        ;download code in p:$200
     do #512,_loop1
     bset #0,x:$ffe0
_lbla
     jclr #3,x:$ffe9,_lblb
     enddo
     jmp _bootend
_lblb
     jclr #0,x:(r2),_lbla
     move x:$ffeb,a1
     move a1,p:(r0)+
_loop1
_bootend
     movec #2,omr         ;return to Mode 2
     andi #$0,ccr
_here    jmp _here        ;wait for break
     end
```

Κατόπιν επιχειρήσαμε να κατεβάσουμε τα δεδομένα στο DSP. Μετά πηγαίνοντας στο μόνιτορ βλέπαμε τι δεδομένα είχαμε κατεβάσει. Η πρώτη διαπίστωση ήταν ότι όταν γράφαμε τη μεσαία λέξη στο HI, το DSP συμπεριφερόταν σα να είχε λάβει την τελευταία. Έτσι η κάθε 24 bit λέξη υπήρχε γραμμένη δύο φορές στη μνήμη, η πρώτη ημιτελής (με το μεσαίο byte να είναι το ίδιο με το λιγότερο σημαντικό) και η δεύτερη αυτή που έπρεπε να είχε μεταφερθεί. Το ADS στην πλακέτα είχε πολύ αδύναμες αντιστάσεις πρόσδεσης με την τροφοδοσία (47ΚΩ). Αυτός ήταν και ο λόγος που αρχικά δεν είχαμε τοποθετήσει δικές μας αντιστάσεις πρόσδεσης σε διάφορα σήματα ελέγχου. Με τη βοήθεια του comp.dsp που μας έριξε μερικές προτάσεις για το πρόβλημα οι υποψίες μας κινήθηκαν στο να τοποθετήσουμε μια καλύτερη αντίσταση πρόσδεσης στο HACK. Η αντίσταση αυτή είχε τίμη 14K. Πράγματι μετά την προσθήκη αυτή οι λέξεις γραφόντουσαν κανονικά.

Όμως τα προβλήματα δεν είχαν τελειώσει εδώ, γιατί κάθε λέξη τώρα υπήρχε πολλές φορές γραμμένη στη μνήμη του DSP. Αυτό συνέβαινε διότι το σήμα εγγραφής HEN όπως μας προειδοποίησαν και στο δίκτυο (Internet), είναι εκπληκτικά ευαίσθητο και μπορεί να καταλάβει και σπινθηρισμούς στην είσοδο του της τάξης των 2 νανοδευτερολέπτων. Αυτό το μοντέλο εξηγούσε αυτή τη συμπεριφορά. Το σήμα αυτό ερχόταν από το PC χωρίς ενδιάμεσο ολοκληρωμένο οδήγησης. Αυτό σημαίνε ότι η έξοδος της παράλληλης πόρτας του PC είχε να αντιμετωπίσει την παρασιτική χωρητικότητα του καλωδίου (μερικα pf) πράγμα που έκανε το σήμα να χανει την μονοτονία του, λόγω θορύβου και αργού χρόνου ανόδου. Το σήμα HEN δεν είναι τύπου schmitt trigger, οπότε η τάση πέρναγε από το κατώφλι αναγνώρισης του λογικού ένα ή μηδέν πολλές φορές λόγω του θορύβου, με αποτέλεσμα να γράφει πολλές φορές την ίδια λέξη (στην εγγραφή του low byte). Αυτό λύθηκε με τη χρήση ενός ενδιάμεσου οδηγού τύπου schmitt trigger που τοποθετήθηκε ενδιάμεσα στην παράλληλη θύρα και το ADS. Έτσι καί το σύστημα αυτό δούλεψε κανονικά.

Προενισχυτής μικροφώνου για το EVB56ADC16

Ο προενισχυτής αυτός έχει σχεδιαστεί για να δέχεται δυναμικό μικρόφωνο στην είσοδο του με εσωτερική αντίσταση τα 600R περίπου. Η μέγιστη τάση εξόδου είναι 7Vpp.

Η είσοδος έχει μια αντίσταση εισόδου 680R, έτσι ώστε να έχουμε σωστή μετάδοση ισχύος, και να μην έχουμε απώλειες στο σήμα εισόδου. Ο πρώτος τελεστικός ενισχύει κατά 470 φορές το σήμα εισόδου. Ο τελεστικός αυτός είναι ένας τελεστικός πολύ χαμηλού θορύβου με μεγάλο εύρος ζώνης, ειδικός για ακουστικές εφαρμογές υψηλής πιστότητας. Η έξοδος του οδηγείται σε ένα βαθυπερατό φίλτρο δεύτερης τάξης. Η συχνότητα αποκοπής επιλέγεται με βραχυκυκλωτήρες από 4KHz ως τα 22KHz σε διάκριτα βήματα (4KHz, 5KHz, 8KHz, 10KHz, 22KHz). Οι βραχυκυκλωτήρες JP1 και JP2 πρέπει να τοποθετηθούν στις ίδιες θέσεις (πχ. για 8KHz JP1 στη θέση 5-6, JP2 επίσης στη θέση 5-6). Τέλος ο τελευταίος τελεστικός έχει ως στόχο τη μεταβλητή ενίσχυση της εξόδου ανάλογα με τον τύπο του μικροφώνου, και την μέγιστη τάση εξόδου που μας ενδιαφέρει. Αυτός ο τελεστικός ενισχύει μόνο τις συχνότητες που μας ενδιαφέρουν, αφού η είσοδος του έχει περάσει από το βαθυπερατό φίλτρο πριν φτάσει σ' αυτόν.

Ο προενισχυτής αυτός συνδέεται με το EVB στη μια είσοδο. Το EVB πρέπει να προγραμματιστεί μέσω των βραχυκυκλωτήρων για είσοδο με μη διαφορική λειτουργία (single ended).

Η Επέκταση Μνήμης

Η αρχική σχεδίαση

Το ADS56K είναι το αναπτυξιακό εργαλείο για την ανάπτυξη λογισμικού, υλικού και εφαρμογών με το DSP56001. Έτσι η μνήμη (RAM) που διαθέτει το αναπτυξιακό αυτό πακέτο υλικού είναι συνολικά 8Kx24. Το DSP56001 έχει τη δυνατότητα να προσπελάσει συνολικά 192Kx24. Η μνήμη του DSP είναι χωρισμένη σε 3 τμήματα, μνήμη προγράμματος, μνήμη δεδομένων X και μνήμη δεδομένων Y μήκους 64Kx24 το κάθε ένα. Έτσι με 16 bit δίαυλο διευθύνσεων μπορεί να προσπελάσει μνήμη 64K για κάθε τμήμα προγράμματος ή δεδομένων. Η εφαρμογή μας, απαιτεί μεγάλα ποσά μνήμης (πολύ μεγαλύτερα από τα 8Kx24 που διαθέτει ήδη το ADS). Το ADS σύμφωνα με το εγχειρίδιο της εταιρίας μπορεί να δεχτεί μέχρι 32Kx24 στην αρχική κάρτα ανάπτυξης (on-board). Επειδή όμως αυτά τα ποσά μνήμης δέν μας φτάνουν, αποφασίσαμε να προστεθεί εξωτερική κάρτα επέκτασης μνήμης για 64Kx24 γιά κάθε τμήμα προγράμματος και δεδομένων.

Το ADS περιέχει όμως και κάποιο κώδικα σε ROM που αποτελεί τον αναπτυξιακό λογισμικό του DSP. Αυτό το λογισμικό θα έπρεπε είτε να αντιγράφεται από τη ROM στην επέκταση της μνήμης προγράμματος, είτε να κατασκευάσουμε κάποιο περίπλοκο κύκλωμα διευθυνσιοδότησης για να υπάρχει επικάλυψη της επέκτασης RAM από τη ROM στης κατάλληλες διευθύνσεις. Τελικά αποφασίστηκε να χρησιμοποιηθεί η πρώτη λύση λόγω ευκολίας, και λόγω του ότι αν προσθέταμε και άλλο κύκλωμα για τη σωστή αποκωδικοποίηση της επέκτασης, ίσως να υπήρχαν προβλήματα χρονισμού του επεξεργαστή με τη μνήμη, αφού οι ταχύτητες είναι πολύ μεγάλες (27MHz). Οι μνήμες που χρησιμοποιούμε είναι οι IDT71256. Οι μνήμες αυτές είναι στατικές 32Kx8 το κάθε chip μέ χρόνο προσπέλασης 25nS. Συνολικά έχουμε τοποθετήσει 18 chips μνήμης. Η επέκταση αποτελείται από μια ξεχωριστή κάρτα (αυτόνομη), και το κύκλωμα εκκίνησης (Boot) που είναι μια ενδιάμεση κάρτα. Η τελική διασύνδεση είναι: Επέκταση μνήμης <-> κύκλωμα εκκίνησης <-> ADS56K

Περιγραφή λειτουργίας του κυκλώματος εκκίνησης

Το κύκλωμα εκκίνησης πρέπει να λειτουργεί μετά το power-on του ADS56K. Η πρώτη σκέψη είναι μετά από κάθε Reset στο DSP να ενεργοποιείται το κύκλωμα εκκίνησης. Μια καλύτερη λύση είναι κατά τη φάση της ανάπτυξης, να έχουμε ένα ξεχωριστό πλήκτρο που να κάνει download το κώδικα της ROM στην επέκταση της RAM. Αργότερα όταν η εφαρμογή ολοκληρωθεί μπορεί να συνδεθεί με το Reset του ADS. Αυτό γίνεται λόγω του ότι κατά τη φάση της ανάπτυξης μπορεί να γίνει Reset για λόγους λογισμικού και δεν θα θέλαμε να περιμένουμε να κάνει το κύκλωμα εκκίνησης Download τον κώδικα της ROM ξανά. Το κύκλωμα της εκκίνησης είχει σχεδιαστεί έτσι ώστε να μπορέσει να λειτουργήσει ανά πάσα στιγμή. Αυτό σημαίνει ότι μπορούμε να χρησιμοποιούμε την αρχική διαμόρφωση του συστήματος (8Kx24) και όχι την δική μας επέκταση μνήμης.Ο τρόπος που δουλεύει το κύκλωμα αυτό θυμίζει κάπως DMA (Direct Memory Access).

Το κύκλωμα αποτελείται από μια μονάδα παραγωγής διευθύνσεων, από τους απομονωτές των σημάτων ελέγχου, από τον έλεγχο λογικής, και από τους απομονωτές των διευθύνσεων. Η λειτουργία του κυκλώματος περιλαμβάνει τα παρακάτω βήματα.

Αρχικά το κύκλωμα μηδενίζει τους μετρητές της μονάδας παραγωγής διευθύνσεων. Επίσης οι μονάδες απομόνωσης είναι ενεργοποιημένες και απομονώνουν τη μονάδα παραγωγής διευθύνσεων από το δίαυλο διευθύνσεων του DSP, κατά την κανονική λειτουργία, όπου ο DSP56001 είναι κύριος του διαύλου (bus master). Η προσπέλαση της ROM από το DSP56001 σε αυτό το σημείο επιτρέπεται.

Βήμα 1ο: Επειδή ο επεξεργαστής λειτουργεί και είναι κυρίαρχος του διαύλου (bus mastering), το κύκλωμα εκκίνησης μόλις πάρει το σήμα για το DownLoad ζητάει την άδεια χρήσης του διαύλου με το σήμα /BR (Bus Request) από το DSP.

Βήμα 2ο: Σε κάποια χρονική στιγμή ο επεξεργαστής δίνει την άδεια χρήσης του διαύλου στην εξωτερική συσκευή (κύκλωμα εκκίνησης) με το σήμα /BG (Bus Grand). Ο επεξεργαστής θέτει τις εξόδους του σε κατάσταση απείρου αντίστασης (Tri-State) και περιμένει τη λήξη του σήματος /BR.

Βήμα 3ο: Μόλις το κύκλωμα εκκίνησης πάρει την κυριότητα του διαύλου θέτει τα κυκλώματα ελέγχου στην εξής κατάσταση:

α. Επιτρέπει την χρήση της ROM διατηρώντας High το CE (μέσω του JG10). Ένα JK flip-flop σε συνδεσμολογία Toggle περιμένει να τελειώσει το Bus mastering για να απενεργοποιήσει τη ROM του ADS.
β. Ενεργοποιεί την ηλεκτρική απομόνωση των σημάτων ελέγχου μεταξύ του ADS και της RAM επέκτασης και επιτρέπει στο κύκλωμα εκκίνησης να μεταβάλλει τα σήματα ελέγχου /RD, /WR, /PS, /DS, X/Y.Τα σήματα ελέγχου /PS, /DS, X/Y είναι σε σταθερές στάθμες. Το /PS (program store) είναι ενεργό (‘0’) επειδή θέλουμε να μεταφέρουμε κώδικα από τη μνήμη προγράμματος.Το /DS (data store) είναι ανενεργό (‘1’) για τον ίδιο λόγο. Το X/Y το θέτουμε σε μια σταθερή λογική κατάσταση, όποια θέλουμε αφού δεν παίζει κανένα ρόλο για την μεταφορά κώδικα. Το σήμα αυτό χρειάζεται για να επιλέγει ποιά μνήμη δεδομένων θέλουμε να προσπελά σουμε. Απλώς το θέτουμε σε μια σταθερή λογική κατάσταση για λόγους θορύβου.Τα σήματα /RD /WR ελέγχονται από το κύκλωμα εκκίνησης συνεχώς.
γ. Οι απομονωτές της μονάδας παραγωγής διεύθυνσεων συνδέουν τους μετρητές με τον δίαυλο δεδομένων.

Βήμα 4ο: Ενεργοποιούνται οι απαριθμητές οι οποίοι είναι η μονάδα παραγωγής διευθύνσεων (address generation unit). Το ρολόι του συστήματος από όπου χρονίζονται όλα τα κυκλώματα είναι ένα 555 σε συνδεσμολογία ασταθούς πολυδονητή σε συχνότητα 500KHz. Σε κάθε παλμό του ρολογιού παράγεται μια διεύθυνση. Παράλληλα μέσω τον μονοσταθών μονοδονητών παράγεται ένας παλμός /RD διάρκειας T_1=60nS που οδηγείται στο ADS για να πάει στη ROM. Παράλληλα δημιουργείται και ο παλμός /WR που οδηγείται στη RAM επέκτασης διάρκειας T_2=40nS. Ο χρόνος T_1 πρέπει να είναι μεγαλύτερος σε διάρκεια από τον T_2, και να τον περιλαμβάνει. Έτσι επιτυγχάνεται το απ’ ευθείας γράψιμο των δεδομένων από τη ROM στη RAM χωρίς ενδιάμεσο μανδαλωτή (Latch). Κατόπιν έρχεται ο επόμενος παλμός του ρολογιού, η διεύθυνση αλλάζει, διαβάζεται το νέο δεδομένο από τη ROM και γράφεται στη RAM, κ.ο.κ. Αυτό συνεχίζεται για 1024K (addr A_0-A_9). Μόλις το Bit A_{10} γίνει λογικό ένα τότε δίνεται σήμα και όλες οι μονάδες γίνονται RESET.

Βήμα 5ο: Λόγω του RESET αποσύρεται το σήμα /BR και το κύκλωμα εκκίνησης αφήνει τον δίαυλο, αφού πρώτα έχει επαναφέρει τα κυκλώματα εξόδου του σε κατάσταση απείρου αντίστασης.

Βήμα 6ο: Το DSP απαντά με /BG=1 και ο διάυλος επιστρέφει πάλι στον επεξεργαστή.

Σημείωση: Για την εγκατάσταση της κάρτας αυτής απαιτείται να συνδεθεί στο JG10 pin 2 το σήμα CTRL1. Στο JG10 ΔΕΝ πρέπει να υπάρχει βραχυκυκλωτήρας.

Τελική σχεδίαση

Επειδή προείχε η ανάπτυξη του λογισμικού σε PC για την αναγνώριση φωνής η υλοποίηση του υλικού καθυστέρησε. Αυτό μας βοήθησε αργότερα στην αποσαφήνιση, του τι τελικά μας χρειαζόταν και τι όχι, τουλάχιστον από πλευράς υλικού.

Το σίγουρο είναι ότι θα χρειαζόμασταν τουλάχιστον 32KWords από κάθε τράπεζα μνήμης X,Y,P. Έτσι συνεχίσαμε να έχουμε στο μυαλό μας μια πλήρη επέκταση μνήμης για 64KWords. Επίσης, καθώς η εργασία στο λογισμικό αυξήθηκε, άρχισαν να παρουσιάζονται και άλλες ανάγκες, εκτός από μια σκέτη επέκταση μνήμης.

Επειδή θεωρήσαμε ότι λόγω των καταναλώσεων θα χρειαστούμε αρκετή ισχύ, αποφασίσαμε να χρησιμοποίησουμε εξωτερικό τροφοδοτικό για την επέκταση μνήμης, ώστε να μην επιβαρύνουμε με παραπάνω πτώση τάσης τους αγωγούς μεταφοράς ενέργειας από το PC προς το EVM. Αυτό οφείλεται στο ότι το ρεύμα τροφοδοσίας του αναπτυξιακού εργαλείου έρχεται διαμέσου μιας καλωδιοταινίας με μικρή διατομή, άρα μεγάλη σχετικά αντίσταση. Έτσι υπολογίσαμε ότι θα χρειαστεί και κάποιος σταθεροποιητής επί της πρόσθετης πλακέτας για την τροφοδοσία.

Μια άλλη ανάγκη θα ήταν κατά τη διάρκεια της εκσφαλμάτωσης ή και της τελικής σύνδεσης (στην εφαρμογή) του αναπτυξιακού εργαλείου με τον υπολογιστή. Ο πιο προσιτός και εύκολος τρόπος θεωρήθηκε η σειριακή θύρα RS-232, οπότε θα έπρεπε να συμπεριλάβουμε στην επέκταση μνήμης τα κυκλώματα για την σειριακή επικοινωνία και την διασύνδεση των διαφορετικών ηλεκτρικών χαρακτηριστικών του DSP με την RS-232.

Τέλος θεωρήθηκε χρήσιμο πάλι για λόγους εκσφαλμάτωσης, και για αργότερα ως ενδεικτικό λειτουργίας και χειριστήριο ελέγχου του υλικού, η χρήση μιας προγραμματιζόμενης λογικής μήτρας (FPGA) για είσοδο και έξοδο σε υλικό επίπεδο. Η λογική αυτή μήτρα οδηγεί ενδεικτικές φωτοδιόδους εκπομπής (LED), και διαβάζει την κατάσταση οκτώ μικροδιακοπτών (DIP switch) ενώ επικοινωνεί με το DSP. Στο FPGA θα μπορούν να ενσωματωθούν και άλλες ψηφιακές λειτουργίες, όταν ίσως αυξηθούν οι ανάγκες μας.

Τέλος αποφασίστηκε ότι δεν θα κατασκεύαζαμε το κύκλωμα εκκίνησης διότι αποδείχτηκε αρκετά περίπλοκο, και προτίμηθηκε η χρήση ενός PLD με πολύ μικρό χρόνο διάδοσης (propagation delay) για την “έξυπνη” αποκωδικοποίηση της μνήμης.

Οργάνωση της μνήμης

Η οργάνωση της μνήμης είναι σχετικά απλή. Επειδή τα ολοκληρωμένα που βρήκαμε στην αγορά με ικανοποιητική ταχύτητα είχαν χωρητικότητα 32KBx8 ή 32KByte ανά ολοκληρωμένο, η οργάνωση που σκεφτήκαμε για την πλήρη κάλυψη του χάρτη μνήμης ήταν η ακόλουθη. Χωρίσαμε όλες τις περιοχές (προγράμματος, και δεδομένων) σε δύο κομμάτια. Το πρώτο κομμάτι περιείχε τα χαμηλότερα 32KWords και το δεύτερο τα υψηλότερα 32KWords. Έτσι έχουμε τη χαμηλή και την υψηλή περιοχή της μνήμης. Η χαμηλή περιοχή ξεκινάει από τη δίευθυνση 0x0000 και τελειώνει στη διεύθυνση 0x7FFF. Η υψηλή μνήμη ξεκινάει από το 0x8000 και τελειώνει στη διεύθυνση 0xFFFF (σε όλες τις μνήμες P,X,Y). Βέβαια αν κοιτάξει κανείς το χάρτη μνήμης του DSP υπάρχουν κάποιες περιοχές που δεν μπορούν να

χρησιμοποιηθούν, αφού εκεί υπάρχει η εσωτερική μνήμη του DSP ή τα περιφερειακά του. Αυτό δεν μας πειράζει αφού το DSP θα αγνοήσει την ύπαρξη τους και δεν θα το επηρεάσουμε καθόλου στην λειτουργία του.

Για να πλατύνουμε το μήκος της λέξης από 8 bit σε 24 (και έτσι να κάνουμε μνήμη 32KWords και όχι 32KBytes) βάζουμε σε κάθε περιοχή μνήμης τρία ολοκληρωμένα μνήμης με κοινό δίαυλο διευθύνσεων ενώ οι τρείς 8 bit δίαυλοι δεδομένων χρησιμοποιούνται ως ένας με πλάτος 24 bits. Αυτό συμβαίνει σε όλες τις τράπεζες μνήμης. Η κάθε τράπεζα μνήμης έχει τρία ολοκληρωμένα μνήμης τυχαίας προσπέλασης στη χαμηλή περιοχή μνήμης και άλλα τόσα στην υψηλή περιοχή. Αυτό μας κάνει έξι ολοκληρωμένα ανά περιοχή (P,X,Y). Ο συνολικός αριθμός των ολοκληρωμένων μνήμης θα είναι (για επέκταση 64KWords σε όλες τις τράπεζες) 6 (ανα περιοχή) επί 3 (P,X,Y) =18 chips.

Είναι φανερό ότι η παρουσία τέτοιου αριθμού ολοκληρωμένων πάνω στους διαύλους του DSP, δεν θα ήταν κατι το ευχάριστο γι' αυτό. Οι λόγοι είναι πολλοί αλλά ο σημαντικότερος θα είναι οι χωρητικότητες που θα έχει να αντιμετωπίσει. Κάθε γραμμή της RAM λόγω της CMOS κατασκευής της παρουσιάζει φορτίο 11pF. Άρα σε κάθε γραμμή διευθύνσεων θα υπάρχουν 18 τέτοια φορτία δηλαδή 200pF!!. Αυτό θα έκανε τον επεξεργαστή να εργάζεται λες και έχει πάρει ναρκωτικά (ο επεξεργαστής καθυστερεί κατά 1nS για κάθε 12pF φορτίου). Στον υπολογισμό αυτού του φορτίου δεν έχουμε λάβει υπ' όψη μας ούτε τις παρασιτικές χωρητικότητες λόγω των γραμμών της πλακέτας, ούτε καν και το υπάρχον φορτίο στην ίδια την κάρτα ανάπτυξης.

Αυτό μας ανάγκασε να χρησιμοποιήσουμε ενδιάμεσους οδηγούς και για το δίαυλο διευθύνσεων και για το δίαυλο των δεδομένων. Φυσικά το φορτίο των 200pF παραμόνευε ακόμα και έτσι, όχι για τον επεξεργαστή, αλλά αυτή τη φορά για τους οδηγούς (buffers). Η λύση ήταν να μοιράσουμε στα δύο τις διευθύνσεις, δηλαδή να τοποθετήσουμε διπλούς οδηγούς στο δίαυλο των διευθύνσεων έτσι ώστε να πέσει το μισό φορτίο σε κάθε ομάδα οδηγών. Αυτό ήταν εύκολο. Ο χωρισμός είχε πρακτικά γίνει από τη στιγμή που χρησιμοποιήσαμε 32KWords μνήμη. Ο διαχωρισμός έγινε σε υψηλή και σε χαμηλή περιοχή διευθύνσεων. Έτσι χρησιμοποιήθηκαν δύο οδηγοί των 8 bits (ο δίαυλος διευθύνσεων είναι 16 bits) για την χαμηλή περιοχή (0x0000 ως 0x7FFF) και άλλοι δύο για την υψηλή περιοχή (0x8000 ως 0xFFFF). Ο δίαυλος των δεδομένων είναι πιο ελαφρύς γιατί έχει 6 ολοκληρωμένα ανά Byte δεδομένων οπότε η συνολική χωρητικότητα είναι μικρότερη (66pF). Έτσι εκεί τοποθετήσαμε μόνο μια ομάδα τριών απομονωτών, έναν για κάθε οκτάδα δεδομένων.

Ανάλογη απομόνωση χρειάζονται και τα σήματα ελέγχου εγγραφής, ανάγνωσης κλπ, αφού οδηγούνται σε όλα τα ολοκληρωμένα. Έτσι πέρασαν και αυτά από οδηγούς-ενισχυτές. Από κάθε σήμα ελέγχου παράχθηκαν δύο αντίτυπά τους που το κάθε ένα από αυτά οδηγούσε είτε την χαμηλή είτε την υψηλή περιοχή μνημών, σε όλες τις τράπεζες, ώστε να μπορούν να οδηγηθούν σε όλα τα ολοκληρωμένα RAM με μικρές καθυστερήσεις, μικρό jitter, και ελάχιστα φορτία.

Οι οδηγοί και οι απομονωτές των σημάτων από το ADS είναι της οικογένειας TTL - F (Fast) με πολύ μικρούς χρόνους διάδοσης αλλά με μεγάλη κατανάλωση. Προτιμήσαμε να χρησιμοποιήσουμε περιβλήματα SMT, για να εξοικονομήσουμε χρόνο, αλλά καί για να μήν δημιουργήσουμε μεγάλες διαφορές στην σύνθετη αντίσταση των γραμμών που φέρνουν τα σήματα από το ADS.

Προγραμματιζόμενη λογική (PLD)

Το κύριο πρόβλημά μας ήταν το ότι είχαμε ένα κομμάτι κώδικα σε ROM πάνω στην αναπτυξιακή κάρτα, οπότε είχαμε δύο επιλογές.

α. Μεταφέρουμε με λογισμικό ή υλικό το κομμάτι αυτό στη διάρκεια του Reset στην εξωτερική μνήμη.

β. Κάνουμε μια αποκωδικοποίηση με εξωτερική λογική και εμφανίζαμε την εσωτερική ROM ως ένα κομμάτι της εξωτερικής RAM. Αυτό σήμαινε ότι οι εξωτερικές διευθύνσεις της RAM που αντιστοιχούσαν στη ROM του DSP δεν θα μπορούσαν να χρησιμοποιηθούν (2KWords).

Μετά από πολύ σκέψη επιλέξαμε (σε αντίθεση με την αρχική σχεδίαση) τη δεύτερη λύση. Θα χάναμε 2KWords αλλά αυτό το ποσό μνήμης θα ήταν ελάχιστο μπροστά στη συνολική μνήμη που θα είχαμε στη διάθεση μας. Εξ' άλλου αν γίνοταν αντιγραφή στη RAM του κώδικα του μόνιτορ πάλι θα χάναμε αυτή μνήμη. Η διαφορά θα ήταν στην ταχύτητα εκτέλεσης αφού οι ROM έχουν κύκλο ανάγνωσης 45nS, οπότε θα χρειάζονται κύκλοι καθυστέρησης. Όμως αυτό δεν είναι κρίσιμο, καθώς η ταχύτητα εκτέλεσης του μόνιτορ δεν είναι κάτι που ενδιαφέρει ιδιαίτερα τον χρήστη σε αντίθεση με την εφαρμογή.

Για το λόγο αυτό θεωρήσαμε απαραίτητο να μην βασιστούμε στα κυκλώματα αποκωδικοποιήσης της κάρτας ανάπτυξης, αφού η λογική "κόλλα" που θα προσθέταμε θα μας καθυστερούσε σε ανεπίτρεπτο βαθμό τα σήματα ελέγχου του DSP, δεδομένου ότι υπήρχαν ήδη δύο τουλάχιστον επίπεδα αποκωδικοποίησης πάνω στην κάρτα ανάπτυξης. Και αυτό είναι φυσικό όταν χρειάζεσαι ταχύτητα "δίνης" με έναν επεξεργαστή που έχει κύκλο μηχανής 74nS, δεν μπορείς να προσθέτεις λογική για πλάκα μιας και αυτό μπορεί να σου κοστίσει μια προσωπική συνάντηση με κλίγκον.

Έτσι λοιπόν αποφασίσαμε τη χρήση ενός PLD που έκανε όλη τη δουλειά της αποκωδικοποίησης των διευθύνσεων μια και καλή, αγνοώντας τα αντίστοιχα ημι-έτοιμα σήματα του DSP. Η αποκωδικοποίηση γινόταν απ' ευθείας από το δίαυλο διευθύνσεων του επεξεργαστή. Το πρόγραμμα που γράψαμε γι' αυτό το σκοπό φαίνεται στο παράρτημα των σχεδίων της προγραμματιζόμενης λογικής.

Το PLD πρέπει να ενεργοποιήσει ανάλογα τα σήματα ενεργοποίησης και απενεργοποίησης των ολοκληρωμένων μνήμης (CS) ανάλογα αν γίνεται προσπέλαση στην υψηλή περιοχή της μνήμης, στην χαμηλή, σε οποιαδήποτε περιοχή προγραμμάτος ή δεδομένων. Επίσης ενεργοποιεί τους απομονωτές του διαύλου των δεδομένων όταν πρέπει, και δίνει μια ένδειξη για λόγους εκσφαλμάτωσης σε περίπτωση διακοπής του επεξεργαστή. Προσέξτε ότι και εδώ τα σήματα είναι μοιρασμένα σε χαμηλές και υψηλές περιοχές μνήμης για μείωση των χωρητικών φορτίων ανά γραμμή. Επίσης, οι RAM που έχουν απενεργοποιημένο το CS, βρίσκονται σε κατάσταση χαμηλής κατανάλωσης, κάτι που είναι σημαντικό, μιας και η κατανάλωση μιας ενεργής RAM μπορεί να φτάσει και τα 80mA.

Τέλος για την εμφάνιση της ROM στην περιοχή της RAM το PLD απλώς δεν ενεργοποιεί αυτές τις διευθύνσεις στη μνήμη τυχαίας προσπέλασης, οπότε οι ROM μπορούν να διαβαστούν από τον επεξεργαστή (το κύκλωμα αποκωδικοποιήσης της μνήμης στο αναπτυξιακό, εργάζεται παράλληλα με το δικό μας κύκλωμα) χωρίς να εμπλακούν οι RAM και να γίνει σύγκρουση στο δίαυλο.

Το PLD που τοποθετήσαμε είναι το iPLD22V10 -25ns. Αυτό μας εξασφάλισε αρκετούς ακροδέκτες για να περάσουμε ικανοποιητικό αριθμό από γραμμές διευθύνσεων, και καλή ταχύτητα απόκρισης.

Τροφοδοτικό

Η πλακέτα καταναλώνει περίπου 600mA στα 5V με 32KWords μνήμη σε όλες τις τράπεζες μνήμης. Η κατανάλωση προέρχεται κύρια από τους απομονωτές και οδηγούς των γραμμών διευθύνσεων και δεδομένων και από το ενεργοβόρο PLD. Οι μνήμες, λόγω του ότι είναι φτιαγμένες με τεχνολογία CMOS, είναι κάπως πιο ήπιες σε απαιτήσεις. Το τροφοδοτικό είναι ένας κλασσικός σταθεροποιητής με τις προστασίες του και τους πυκνωτές σταθεροποίησης.

Το κύκλωμα της σειριακής θύρας

Το κύκλωμα της σειριακής θύρας είναι ένας απλός μετατροπέας τάσεων από τα επίπεδα της RS-232 σε TTL και αντίστροφα. Είναι το κλασσικό πια MAX232 που έχει ενσωματωμένο παλμοτροφοδοτικό, ώστε από μονή τροφοδοσία 5V να παράγει τα +/-12V που χρειάζεται για τη σειριακή. Κατά τα άλλα το DSP56001 δεν χρειάζεται και τίποτε άλλο για να επικοινωνήσει με τον υπολογιστή. Το κύκλωμα αυτό δεν διαθέτει υλικό χειραψίας (Hardware Handshake). Η σύνδεση γίνεται με ένα απλό καλώδιο Null Modem (2->3, 3->2, 5 ->5 ανεστραμένο).

Τα κυκλώματα εισόδου-εξόδου

Αυτά τα κυκλώματα είναι οι οδηγοί των LED και οι αναγνώστες των διακοπτών DIP. Ο αριθμός των LED είναι μεγάλος (16) όπως επίσης και ο αριθμός των διακοπτών (8). Για να μπορέσουμε να τα οδηγήσουμε αυτά από το DSP θα έπρεπε να χρησιμοποιήσουμε ένα υπερβολικό αριθμό από ακροδέκτες και ολοκληρωμένα. Έτσι αποφασίσαμε να χρησιμοποιήσουμε την αιχμή της τεχνολογίας και να βάλουμε ένα FPGA (Field Programmable Gate Array). Με το DSP θα κατεβάζαμε σειριακά τα δεδομένα εισόδου-εξόδου και το FPGA θα αναλάμβανε την όλη διαδικασία. Για λόγους εκσφαλμάτωσης το FPGA μπορεί να αναπρογραμματιστεί με κάποιο εναλλακτικό πρόγραμμα από το PC, ώστε να μας βοηθήσει στη διαδικασία της ανάπτυξης.

Η σχεδίαση του FPGA περιλαμβάνει τρία βασικά τμήματα. Το πρώτο τμήμα είναι τα κυκλώματα διασύνδεσης με το DSP. Αυτά αναλαμβάνουν την διεκπεραίωση της επικοινωνίας με το DSP τόσο στην είσοδο όσο και στην έξοδο. Το δεύτερο τμήμα είναι τα κυκλώματα οδήγησης των LED με τους καταχωρητές τους, και το τρίτο τμήμα είναι τα κυκλώματα εισόδου από τους μικροδιακόπτες.

Η επικοινωνία του DSP με το FPGA γίνεται μέσω της πόρτας B του DSP. Από εκεί χρησιμοποιούμε τέσσερα σήματα για να επικοινωνήσουμε με το FPGA. Η πόρτα B για το DSP δεν χρησιμοποιείται ως θύρα Host Interface, αλλά ως γενική θύρα εισόδου-εξόδου. Το βασικό μοντέλο της επικοινωνίας είναι αυτό της σύγχρονης μεταφοράς δεδομένων. Υπάρχει μια γραμμή ρολογιού από το DSP προς το FPGA, και μια γραμμή δεδομένων. Το κάθε δεδομένο (bit) μεταφέρεται σε κάθε ένα παλμό του ρολογιού. Αν η γραμμή ανάγνωσης/εγγραφής είναι σε κατάσταση ανάγνωσης τότε μεταφέρονται οι καταστάσεις των μικροδιακοπτών από το FPGA προς το DSP. Αν η κατάσταση αυτή είναι εγγραφή τότε το DSP μεταφέρει την κατάσταση των LED προς το FPGA. Υπάρχει επίσης η δυνατότητα επιλογής περιφερειακού (CE) αν αυτό υπήρχε, ώστε να μπεί παράλληλα στις γραμμές σειριακής επικοινωνίας και άλλο ολοκληρωμένο κύκλωμα αν αυτό χρειάζοταν.

Το κύκλωμα επικοινωνίας του FPGA ανάλογα με την κατάσταση των γραμμών επικοινωνίας με το DSP, αναλαμβάνει το διαχωρισμό των δεδομένων και των σημάτων ελέγχου για κάθε μονάδα εισόδου/εξόδου. Η γραμμή μεταφοράς δεδομένων είναι διπλής κατεύθυνσης, οπότε πρέπει να επιτρέπεται η έξοδος των δεδομένων όταν το DSP διαβάζει τους διακόπτες, ενώ πρέπει να μετατρέπεται σε είσοδο όταν το DSP

στέλνει τις καταστάσεις των LED προς το FPGA. Αυτό γίνεται με τα σήματα επιλογής (εσωτερικά) LED_EN και SWITCH_EN, τα οποία επιλέγουν είτε τα κυκλώματα των LED, είτε τα κυκλώματα των διακοπτών αντίστοιχα, όπως επίσης και την κατεύθυνση των δεδομένων στην εξωτερική γραμμή.

Το κύκλωμα ενεργοποίησης των LED αποτελείται από δύο καταχωρητές-ολισθητές των 8 bit. Τα δεδομένα εισέρχονται σειριακά στην σειριακή είσοδο (αν πρόκειται για δεδομένα που πρέπει να πάνε στα LED), και ολισθαίνουν προς την τελική έξοδο τους. Η διαδικασία φόρτωσης είναι να φορτωθούν πρώτα τα bits 15 ως 8 (από το περισσότερο σημαντικό ψηφίο προς το λιγότερο σημαντικό), και κατόπιν τα bits 7 ως 0. Μόλις γραφτούν όλοι οι καταχωρητές (16 παλμοί ρολογιού), τότε το κύκλωμα ελέγχου της μονάδας, που αποτελείται από έναν απαριθμητή και ένα flip-flop, κλειδώνουν την ολίσθηση (για την περίπτωση που σταλούν περισσότεροι παλμοί ρολογιού). Οι ολισθητές επανέρχονται στην κανονική τους κατάσταση, μόλις απενεργοποιηθεί το εξωτερικό σήμα επιλογής του FPGA. Η τοποθέτηση των δεδομένων γίνεται μετά την κάθοδο του ρολογιού, ενώ αυτά εισάγονται στα κυκλώματα του FPGA με την άνοδο του παλμού του ρολογιού. Το συγκεκριμένο FPGA της XILINX μπορεί οδηγήσει απ' ευθείας φορτίο 24mA από τις εξόδους του, οπότε έτσι αποφεύγουμε και τη χρήση εξωτερικών κυκλωμάτων οδήγησης.

Το κύκλωμα ανάγνωσης των μικροδιακοπτών είναι πρακτικά ένας καταχωρητής με παράλληλη είσοδο και σειριακή έξοδο. Όταν στείλουμε οκτώ παλμούς στο ρολόι, θα πάρουμε την κατάσταση των μικροδιακοπτών. Όταν οι μικροδιακόπτες είναι όλοι στη θέση ON τότε αυτό που διαβάζουμε είναι 0xFF ενώ αν είναι όλοι στη θέση OFF τότε το DSP διαβάζει 0x00. Τα δεδομένα φορτώνονται στη γραμμή των δεδομένων με την άνοδο του ρολογιού και είναι διαθέσιμα με το πέσιμο του ρολογιού για το DSP.

Από τη μεριά του DSP, το ρολόι, καθώς και όλα τα σήματα παράγονται με λογισμικό.

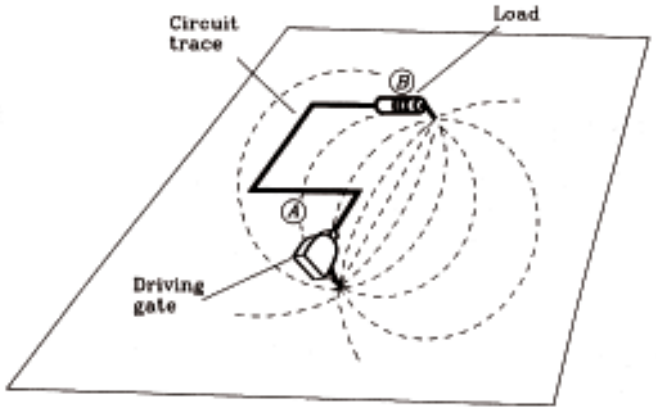
Η σχεδίαση της πλακέτας

Η σχεδίαση της πλακέτας είχε τρομερές δυσκολίες για την υλοποίηση της. Το κυριότερο πρόβλημα ήταν, ότι θα χρειαζόμασταν τουλάχιστον δύο επίπεδα, και αυτό θα σήμαινε πολλές μεταβάσεις από το ένα επίπεδο στο άλλο (με via) λόγω των πολλών γραμμών που είχαμε να περάσουμε. Το κόστος εδώ παίζει καθοριστικό ρόλο καθώς η κατασκεύη μια πλακέτας με επιμεταλλωμένες τρύπες κοστίζει γύρω στις 90.000 δρχ. πράγμα απαγορευτικό για μια πτυχιακή (μόνο τα ολοκληρωμένα της RAM κοστίζουν 45.000δρχ για τη μισή επέκταση μνήμης, 32KWords x 3). Φυσικά ούτε λόγος για πλακέτα τεσσάρων επιπέδων όπως θα έπρεπε κανονικά να την κατασκευάσουμε, για λόγους που θα εξηγήσουμε παρακάτω. Έτσι η μόνη λύση που υπήρχε ήταν η ένωση των via με το χέρι στα δύο επίπεδα, παρ' όλο που αυτό θα ήταν ιδιαίτερα επίπονο και επίφοβο για την εκσφαλμάτωση αφού και το παραμικρό λάθος θα μας κόστιζε πολύ χρόνο για να βρεθεί. Η σχεδίαση λοιπόν ξεκίνησε με αυτό το σκεπτικό (όσο το δυνατόν λιγότερες μεταβάσεις μεταξύ των επιπέδων).

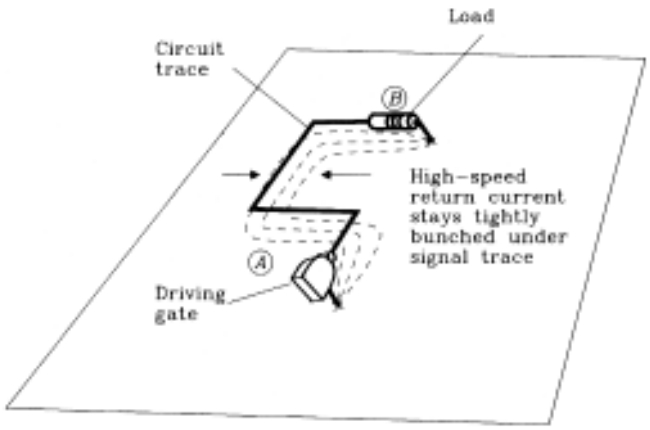
Από σχεδιαστικής πλευράς η χρήση πλακέτας δύο επιπέδων ήταν ένας πονοκέφαλος. Ο λόγος ήταν ότι οι συχνότητες που εργάζεται ο επεξεργαστής DSP είναι πολύ μεγάλες (με τα σημερινά δεδομένα η συχνότητα των 27MHz μπορεί να θεωρείται λίγο ξεπερασμένη, αλλά οι υπολογιστές του εμπορίου έχουν τουλάχιστον 4 επίπεδα στις μητρικές πλακέτες). Αν σκεφτεί κανείς ότι κάθε εντολή εκτελείται σε ένα κύκλο 74ns με μνήμες χωρίς κανένα κύκλο καθυστέρησης, είναι εύκολο κανείς να αντιληφθεί τη διαφορά από ένα απλό μικροελεγκτή των 12MHz και κύκλο εντολής 1μs. Βέβαια η μέγιστη τιμή της συχνότητας που υπάρχει στους διαύλους, δεν εξαρτάται από τον κύκλο εντολής του κάθε επεξεργαστή, αλλά από μια πολύ σημαντική

παράμετρο από ηλεκτρικής πλευράς, τους χρόνους ανόδου και καθόδου των παλμών (οι οποίοι έμμεσα μόνο εξαρτώνται από τη συχνότητα λειτουργίας του κάθε επεξεργαστή). Οι χρόνοι αυτοί καθορίζουν το πόσο μεγάλες συχνότητες παράγονται στις διάφορες γραμμές, και το ποιές από αυτές πρέπει να προσέξουμε και να τις χειριστούμε ως γραμμές μεταφοράς.

Γενικά σε συχνότητες πάνω από 12MHz παύει να δουλεύει το κλασσικό μοντέλο των ψηφιακών κυκλωμάτων καθώς όλα τα σήματα αρχίζουν να μικραίνουν το μήκος κύματος τους (αφού αυξάνεται η συχνότητα τους) με αποτέλεσμα αν οι γραμμές που κινούνται έχουν μήκος πάνω από το μήκος κύματος τους τότε παρουσιάζονται διάφορα ανεπιθυμήτα κυματικά φαινόμενα όπως για παράδειγμα οι ανακλάσεις. Μια διαφορά στο πλάτος της γραμμής ή το πέρασμα από ένα συνδετήρα (η διαφορά της σύνθετης αντίστασης πάνω σε μια γραμμή) επιδρά ώς εξασθενητής ή ακόμη και ως ενισχυτής ενός σήματος γιατί η γραμμή πλέον συμπεριφέρεται σαν ένας κυματοδηγός. Αυτό ισχύει για μήκη γραμμών που ξεπερνούν το μήκος κύματος του σήματος που μεταφέρει. Για να είναι ελεγχόμενες αυτές οι καταστάσεις πρέπει η κάθε γραμμή να συνοδεύεται από μια γείωση, και αυτός είναι ό λόγος που χρησιμοποιούνται πλακέτες τεσσάρων επιπέδων. Τα δύο εσωτερικά επίπεδα είναι αφοσιωμένα στη γή και στη τροφοδοσία (και για άλλους λόγους που θα πούμε παρακάτω). Έτσι το σήμα μπορεί και επιστρέφει από την πορεία που ήρθε ακριβώς (βλ. και σχήμα 4.7 και 4.8). Σε αυτές τις περιπτώσεις μπορούμε να υπολογίσουμε τις σύνθετες αντιστάσεις και να χρησιμοποιήσουμε τερματισμούς γραμμής έτσι ώστε να εξασφαλίσουμε τη σωστή μεταφορα του σήματος από σημείο σε σημείο και ιδιαίτερα τα σήματα που διανύουν μεγάλες αποστάσεις.

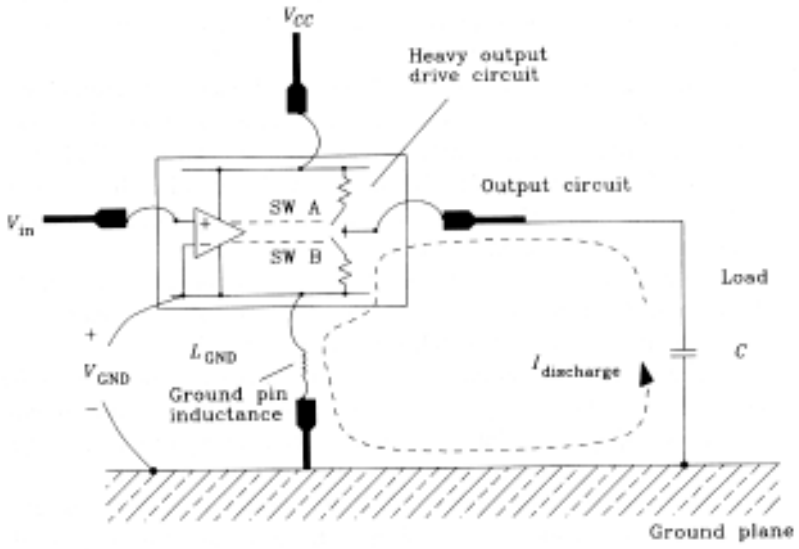


Σχ. 4.7. *Η επιστροφή του ρεύματος στις χαμηλές συχνότητες.*



Σχ. 4.8. *Η επιστροφή του ρεύματος στις υψηλές συχνότητες.*

Οι γραμμές μεταφοράς δεν είναι το μόνο δύσκολο σημείο που έχει κανείς να αντιμετωπίσει. Υπάρχουν και πιο άμεσοι κίνδυνοι που μπορεί να αντιμετωπίσει κανείς. Οι τροφοδοσίες και κυρίως οι γειώσεις πρέπει να είναι καθαρές αφού όλα τα ψηφιακά κυκλώματα (τουλάχιστον των κλασσικών οικογενειών TTL/CMOS) εκεί βασίζουν την αναφορά τους για να καταλάβουν τα σήματα εισόδου. Έτσι λοιπόν πρέπει να προσεχθούν οι επιστροφές των ρευμάτων για κάθε σήμα, ώστε αυτό να μπορεί να επιστρέψει στην πηγή που το παρήγαγε από τον πιο σύντομο δρόμο, χωρίς να μπερδεύεται με επιστροφές άλλων σημάτων. Αν γίνει αυτό, δηλαδή η ταυτόχρονη σύμπτωση επιστροφών πολλών σημάτων σε μια γραμμή, τότε η μικρή αντίσταση που είναι αμελητέα όταν ρέει λίγο ρεύμα, αποκτά μεγάλη τιμή και δημιουργεί πτώση τάσης με αποτέλεσμα να υπάρχει κίνδυνος η τάση αναφοράς για ορισμένα ολοκληρωμένα να μην είναι ίδια με τα υπόλοιπα, οπότε θα δημιουργηθούν ασυνενοησίες και θα καταλήξει να γίνει ένας πύργος της Βαβέλ. Αυτό μπορεί να γίνει στιγμιαία όταν πολλά ολοκληρωμένα αλλάζουν κατάσταση σχεδόν ταυτόχρονα, οπότε για κάποιο πολύ μικρό χρονικό διάστημα η τάση στη γή σε ορισμένα σημεία μπορεί να ανυψωθεί κατά μερικά βολτ χαλώντας τη “γλώσσα” επικοινωνίας των κυκλωμάτων.



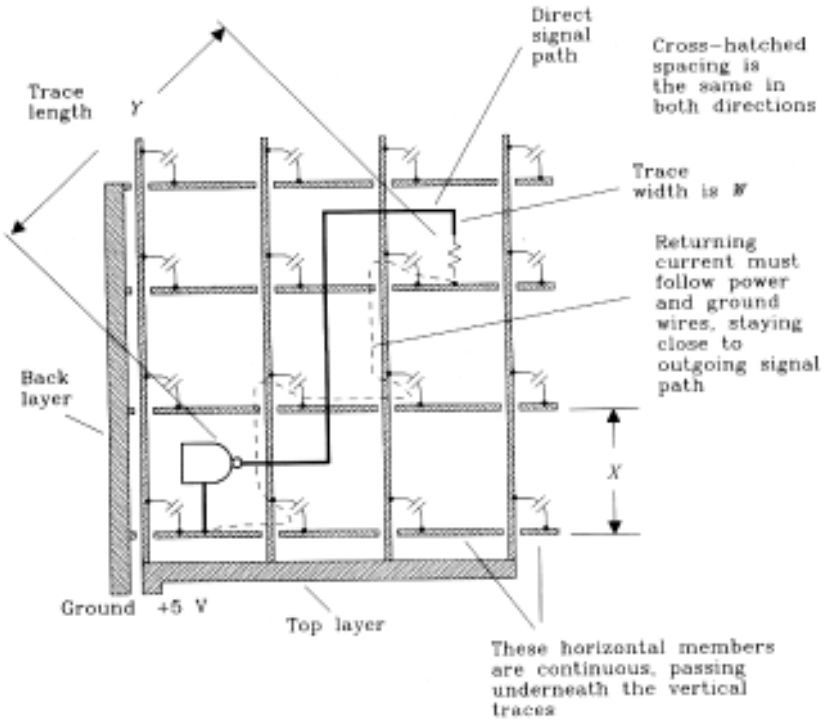
Σχ. 4.9. *Το φαινόμενο το ground bounce.*

Επίσης οι κάθε λογής χωρητικότητες (παρασιτικές) στις γραμμές μπορεί να δημιουργήσουν προβλήματα στους χρόνους ανόδου και καθόδου των σημάτων, όπως επίσης μπορεί να υπάρξει αλληλεπίδραση μεταξύ γειτονικών σημάτων που ταξιδεύουν πολύ κοντά μεταξύ τους.

Η σχεδίαση της πλακέτας έγινε λαμβάνοντας όλα όσα είπαμε πιο πάνω και παρ' όλο που η σχεδίαση έγινε για πλακέτα δύο επιπέδων (με αποτέλεσμα να μην μπορούμε να κάνουμε και πολλά) προσέξαμε όσο μπορούσαμε για την όσο το δυνατόν ασφαλέστερη μετακίνηση των σημάτων. Πρέπει να πούμε εδώ ότι υπήρχε και θέμα διαστάσεων της πλακέτας εκτός των υπόλοιπων σχεδιαστικών δυσκολιών.

Αρχίζοντας την πλακέτα ξεκινήσαμε με την τοποθέτηση των μνημών. Η διάταξη έγινε με βάση τις υψηλές και τις χαμηλές θέσεις μνήμης αφού κάθε ομάδα είχε δικούς τους απομονωτές από το δίαυλο του επεξεργαστή. Έτσι φτιάχτηκαν δύο σειρές με μνήμες που η κάθε μια αντιπροσωπεύει τη χαμηλή και την υψηλή μνήμη. Αυτό θα βοηθούσε και αργότερα αν χρειάζοταν να τοποθετηθεί μόνο η μισή επέκταση μνήμης (όπως και τελικά έγινε). Οι απομονωτές των διευθύνσεων καθώς και όλοι οι πυκνωτές αποσύξευξης, για λόγους οικονομίας χώρου αλλά και λόγω των μικρότερων παρασιτικών χωρητικοτήτων και αυτεπαγωγών, επιλέχτηκαν να είναι εξαρτήματα επιφανειακής στήριξης (SMD). Στους διαύλους των διευθύνσεων υπάρχει πρόβλεψη για τερματισμούς AC ή DC ή Thevenin, μήπως τυχόν και παρουσιάζοταν κάποιο πρόβλημα.

Οι γραμμές δεδομένων οδηγήθηκαν στους απομονωτές τριών καταστάσεων που είναι σε πολύ κοντινή απόσταση από τις μνήμες, ενώ οι γειώσεις και οι τροφοδοσίες προσέχτηκαν πάρα πολύ. Σχεδιάσαμε γύρω από τις μνήμες ένα πλέγμα με οριζόντιες και κάθετες γραμμές ώστε σε κάθε ολοκληρωμένο να υπάρχει μια διασταύρωση γης και τροφοδοσίας, επιτρέποντας την άμεση επιστροφή των σημάτων από ολοκληρωμένο σε ολοκληρωμένο με την πιο σύντομη διαδρομή.



Σχ. 4.10. *Η τεχνική που ακολουθήθηκε στην σχεδίαση της πλακέττας.*

Επίσης όπου υπήρχε κενός χώρος καλύφθηκε με γείωση ώστε να βελτιώσει τα χαρακτηριστικά των γραμμών. Αυτό έγινε χαρακτηριστικά στις γραμμές των δεδομένων που κατευθύνονται προς στο συνδετήρα για να πάνε στον επεξεργαστή.

Τελικά για καλή μας τύχη, μπορέσαμε και φτιάξαμε την πλακέτα με επιμεταλλωμένες τρύπες μειώνοντας έτσι τον κόπο και τον χρόνο εκσφαλμάτωσης σημαντικά.

Εκσφαλμάτωση

Μόλις συναρμολογήσαμε την πλακέτα αρχίσαμε τους ελέγχους. Λόγω της προηγούμενης κακής εμπειρίας μας με τον ψηφιακό-αναλογικό μετατροπέα είμαστε πολύ πιο προσεκτικοί αυτή τη φορά. Δεν είχαμε την πολυτέλεια του χρόνου. Έτσι ξεκινήσαμε ελέγχοντας όλες τις τάσεις τροφοδοσίας. Τα μόνα κυκλώματα που είχαμε τοποθετήσει ήταν αυτά της επιφανειακής στήριξης και το τροφοδοτικό. Για τα υπόλοιπα είχαμε τοποθετήσει βάσεις. Όταν όλα πήγαν καλά στις αυτόνομες δοκιμές συνδέσαμε την πλακέτα στο ADS θέλοντας να ελέγξουμε αν υπήρχε κάποιο βραχυκύκλωμα στην πλακέτα. Όντως για να μην πάνε τα πράγματα καλά το DSP αρνήθηκε να δουλέψει. Αμέσως αποσυνδέσαμε την κάρτα επέκτασης και επαληθέυσαμε τη σωστή λειτουργία του DSP.

Το πρώτο πράγμα που κάναμε ήταν να ψάξουμε για βραχυκυκλώματα στην πλακέτα μας. Κάνοντας έναν έλεγχο με το πολύμετρο ανακαλύψαμε μια γραμμή διευθύνσεων που είχε 6Ω αντίσταση με τη γή. Αυτή η γραμμή πήγαινε σε δύο ολοκληρωμένα απομονώσης των διευθύνσεων 74F541. Προφανώς κάποιο από τα δύο είχε πάθει βλάβη (ή ίσως και τα δύο). Για να βρούμε ποιό από τα δύο ολοκληρωμένα έφταιγε μετρήσαμε

την αντίσταση με τη γή από τα δύο ολοκληρωμένα. Ο ακροδέκτης του ολοκληρωμένου που παρουσίαζε την μικρότερη αντίσταση ήταν πιθανότατα το κατεστραμένο σημείο. Αυτό αποδείχτηκε εύκολα με την αποκοπή του ακροδέκτη από την πλακέτα. Πράγματι το ολοκληρωμένο ήταν χαλασμένο.

Με αρκετή απορία αλλά και ευχαρίστηση που βρήκαμε τη βλάβη αναρωτιόμαστε για το πως κάηκε η είσοδος. Μετά από λίγη σκέψη καταλάβαμε τι είχε συμβεί. Σε κάποια από τις δοκιμές είχαμε συνδέσει ένα μεταβλητό τροφοδοτικό απ' ευθείας στα 5V , το οποίο όμως ήταν ρυθμισμένο σε μεγαλύτερη τάση. Αυτό είχε ως αποτέλεσμα την καταστροφή του ολοκληρωμένου από την υπέρταση, και την πιθανή δυσλειτουργία των υπολοίπων ολοκληρωμένων. Για λόγους ασφαλείας αποφασίσαμε την αλλαγή όλων των ολοκληρωμένων. Μετά από την αλλαγή αυτή, κάνοντας όλους τους απαραίτητους ελέγχους ξανασυνδέσαμε την πλακέτα επέκτασης στο ADS. Προς μεγάλη μας χαρά δεν εμφανίστηκε κανένα πρόβλημα.

Το επόμενο βήμα ήταν η τοποθέτηση των μνημών. Συμπληρώσαμε τρία ολοκληρωμένα μνήμης στην αρχή (ένα byte σε κάθε περιοχή μνήμης P,X,Y), και αφού όλα πήγαν καλά, τοποθετήσαμε όλα τα ολοκληρωμένα. Ο πρώτος έλεγχος έγινε μέσα από το πρόγραμμα ελέγχου του αναπτυξιακού, όπου δώσαμε εντολή να γεμίσει με κάποιο συγκεκριμένο αριθμό όλη τη μνήμη και κατόπιν εμείς διαβάσαμε τυχαία κάποια κομμάτια να δούμε αν είχαν γραφτεί οι μνήμες. Δυστυχώς όμως οι μνήμες δεν εργάζοταν ή δεν εργάζοταν σωστά. Ήταν σχεδιαστικό σφάλμα ή κατασκευαστικό; Στην αρχή μας πέρασαν από το μυαλό ns, παρασιτικές χωρητικότητες και άλλοι εφιάλτες, αλλά μετά από μια προσεκτική μελέτη το πρόβλημα αποκαλύφθηκε όπως το αυγό του κολόμβου. Ναι ήταν σχεδιαστικό σφάλμα.

Το πρόβλημα ήταν οι απομονωτές τριών καταστάσεων στις γραμμές των δεδομένων, όπου έπρεπε να ελεχθούν με λογική για το πότε θα πρέπει να ενεργοποιηθούν και το πότε όχι. Όπως ήταν τα πράγματα τώρα οι απομονωτές μέσω ενός βραχυκυκλωτήρα εκσφαλμάτωσης ή ήταν μονίμως απενεργοποιημένοι, ή συνεχώς ενεργοποιημένοι. Έτσι παρουσιάζοταν πρόβλημα με τα υπόλοιπα δεδομένα αφού είχαμε σύγκρουση στο δίαυλο. Αυτό μπορούσε να λυθεί εύκολα μιας και στο PLD έιχαμε αφήσει μερικές εξόδους ελεύθερες μήπως τυχόν χρειαστούν (όπως τώρα). Αναπρογραμματίσαμε το PLD και αναδιατάξαμε τη γραμμή ελέγχου των απομονωτών, έτσι ώστε αυτή να οδηγείται από το PLD. Τότε, ως δια μαγείας οι μνήμες λειτούργησαν. Αργότερα με την ολοκλήρωση της πλακέτας θα είχαμε το χρόνο να κάνουμε πραγματικούς ελέγχους μέσω του επεξεργαστή.

Το επόμενο βήμα ήταν η σειριακή. Μετά το κλασσικό πρόβλημα με το αν πρέπει να χρησιμοποιήσουμε κανονικό ή χιαστί καλώδιο (ποτέ δεν θυμάσαι τι είχες σκεφτεί ότι θα ήταν λογικό κατά τη διάρκεια της σχεδίασης), πρόβλημα το οποίο λύθηκε αμέσως, η σειριακή επικοινωνία λειτούργησε κανονικά.

Όμως τα προβλήματα δεν είχαν τελειώσει εδώ. Το FPGA καροδοκούσε με τα δικά του προβλήματα. Τοποθετήσαμε το ολοκληρωμένο στη βάση του (βάση PLCC με 84 ακροδέκτες). Κατεβάσαμε τον κώδικα του υλικού μέσω του PC στο ολοκληρωμένο και κατόπιν ξεκινήσαμε τις δοκιμές. Το πρόγραμμα ελέγχου το είχαμε έτοιμο λίγο-πολύ αφού ξέραμε την συμπεριφορά του από τα διαγράμματα της εξομοίωσης. Όμως το ολοκληρωμένο αυτό δεν έπαιρνε κουβέντα. Ο λόγος αποκαλύφθηκε λίγο αργότερα με τη βοήθεια του παλμογράφου. Η γραμμή από το συνδετήρα γενικών επεκτάσεων του ADS που μετέφερε το σήμα του ρολογιού δεν υπήρχε στην πλακέτα.

Μετά την σύντομη επισκευή και σύνδεση της γραμμής αυτής το FPGA συνέχισε να μην "ακούει" τίποτα. Αυτή τη φορά ανακαλύψαμε άλλο ένα σχεδιαστικό λάθος, μικρό αυτή τη φορά. Χρειαζόταν μια αντίσταση πρόσδεσης στην τροφοδοσία στο σήμα D/N του FPGA. Έτσι ο κώδικας φαινόταν να κατεβαίνει σωστά στο ολοκληρωμένο, αν και δεν είχαμε τρόπο να το εξακριβώσουμε αυτό άμεσα.

Το περιφερειακό ολοκληρωμένο συνέχιζε την αδράνεια του. Καμιά φορά, τυχαία, αν ακουμπούσαμε το probe του παλμογράφου σε ορισμένους ακροδέκτες αυτό φαινόταν να δουλεύει, σαν να έφταιγε κάποια χωρητικότητα. Φυσικά η περιφερειακή αυτή διασύνδεση δεν ήταν κάτι το φοβερό από πλευράς ταχύτητας, ώστε να δικαιολογεί τέτοια συμπεριφορά. Δοκιμάσαμε να ξανακατεβάσουμε τον κώδικα στο FPGA άλλά το αποτέλεσμα ήταν το ίδιο.

Βγάλαμε το ολοκληρωμένο από τη βάση να δούμε μήπως τελικά το πρόβλημα ήταν πρόβλημα επαφής. Κοιτώντας δύο ακροδέκτες της βάσης του ολοκληρωμένου διαπιστώσαμε ότι ήταν μερικά δέκατα του χιλιοστού πιο μέσα γιατί κατά την πρώτη τοποθέτηση είχαν πιεστεί προς τα μέσα. Επισκευάζοντας αυτή την παραμόρφωση στους ακροδέκτες, το FPGA λειτούργησε κανονικά ή σχεδόν κανονικά. Σχεδόν, γιατί δεν άναβαν όλοι οι φωτοδίοδοι. Η τελευταία φωτοδίοδος δεν άναβε καθόλου. Μετά από έναν προσεκτικότερο έλεγχο στην πλακέτα διαπιστώσαμε ότι η φωτοδίοδος δεν συνδέοταν στο σωστό ακροδέκτη του FPGA. Μικρό το κακό σκεφτήκαμε, και αλλάξαμε τον ακροδέκτη εξόδου στα σχέδια του FPGA (η softwareοποίηση του hardware...). Έτσι όλα έπαιζαν ρολόι. Για λόγους συμβατικότητας τοποθετήσαμε και μερικούς αναστροφείς για να αντιστοιχεί το ON των μικροδιακοπτών με λογικό 1 στο πρόγραμμα του DSP.

Κατόπιν φτιάξαμε ένα πρόχειρο πρόγραμμα ελέγχου όλης της μνήμης στο DSP. Αυτό έκανε συνεχείς ελέγχους στη μνήμη σε όλες τις περιοχές, με και χωρίς κύκλους καθυστέρησης. Αν υπήρχε κάποιο πρόβλημα σταματούσε την εκτέλεση του θέτωντας σε λειτουργία κάποιες συγκεκριμένες φωτοδιόδους ανάλογα με το που βρήκε το λάθος. Αλλιώς άναβε με κυκλικό τρόπο κάποιες άλλες φωτοδιόδους.

Τέλος ένα άλλο διαγνωστικό πρόγραμμα που φτιάξαμε έκανε έναν έλεγχο σε επίπεδο firmware, όπου δοκιμάζαμε τις ρουτίνες αποστολής και λήψης για τους αναλογικοψηφιακούς μετατροπείς, τη σειριακή θύρα, και το FPGA. Στην αρχή υπήρχε ένας βρόγχος όπου η είσοδος του A/D οδηγούταν κατ' ευθείαν στην έξοδο του D/A φτιάχνοντας κατά κάποιο τρόπο ένα "καλώδιο". Αν ανιχνεύοταν ένας συγκεκριμένος συνδιασμός στους μικροδιακόπτες τότε σταματούσε η μεταβίβαση δεδομένων μεταξύ των A/D-D/A και ξεκινούσε η μεταφορά δεδομένων στη σειριακή όπου η είσοδος εμφανίζοταν μέσω του DSP πάλι στην έξοδο, ενώ παράλληλα τα LED του FPGA αντανακλούσαν το δυαδικό ισοδύναμο κάθε χαρακτήρα που μεταφέροταν. Αλλάζοντας πάλι το συνδιασμό των μικροδιακοπτών το πρόγραμμα πήγαινε πάλι στην αρχική του κατάσταση. Μετά και από αυτά τα τέστ, ήμασταν πιά έτοιμοι να προχωρήσουμε.

Λογισμικό του DSP

Το DSP έχει στόχο την ταχεία επεξεργασία των δεδομένων, ώστε να έχουμε πραγματική απόκριση ενός συστήματος. Έτσι και εμείς χρησιμοποιούμε το DSP για την εκτέλεση των επίπονων και χρονοβόρων υπολογισμών, ενώ το PC έχει να κάνει ελάχιστες εργασίες, που είναι πιο βολικό να γίνουν από αυτό.

Το DSP έχει να κάνει τις εξής λειτουργίες:
1. Δειγματοληψία από τον A/D
2. Συνεχή εύρεση ενέργειας
3. Συνεχή εύρεση ρυθμού μεταβάσεων του σήματος από το μηδέν
4. Αρχή - τέλος λέξης
5. Παράθυρα και FFT
6. Ομαδοποίηση στο πεδίο των συχνοτήτων
7. Ομαδοποίηση στο πεδίο του χρόνου
8. Αποστολή και λήψη δεδομένων στο PC
9. Απεικόνηση της τρέχουσας κατάστασης και λήψη εντολών

Το PC αντίστοιχα πρέπει να κάνει :
1. Λήψη και αποστολή δεδομένων από και προς το DSP
2. Κανονικοποίηση του χαρακτηριστικού διανύσματος της λέξης
3. Σύγκριση εισερχόμενου διανυσμάτος με τη βάση δεδομένων
4. Εκμάθηση διανυσμάτων

Περιγραφή του λογισμικού DSP

Η γενική δομή του προγράμματος μπορεί να χαρακτηριστεί σαν ένα σύστημα πολυ-επεξεργασίας (multi-tasking). Η κύρια διεργασία , που αντιστοιχεί στο λειτουργικό σύστημα ενός υπολογιστή, έχει την ευθύνη εκτέλεσης όλων των διεργασιών με τρόπο τέτοιο, ώστε να μην βλάπτει τη λειτουργία τους.

Οι κύριες διεργασίες που εκτελούνται είναι τρεις. Η πρώτη διεργασία (Task 1) ασχολείται με την δειγματοληψία και την προ-επεξεργασία του σήματος στο πεδίο του χρόνου (timedom.asm). Η δεύτερη διεργασία (Task 2) ασχολείται με την επεξεργασία των δεδομένων στο πεδίο των συχνοτήτων (freqdom.asm). Τέλος η τρίτη διεργασία (Task 3) αναλάμβάνει την εξαγωγή και την επεξεργασία των αποτελέσμάτων από τις δυο προηγούμενες διεργασίες και την αποστολή τους στο PC (procdata.asm). Η κύρια διεργασία (Main) συντονίζει το όλο έργο και καθορίζει τον τρόπο εκτέλεσης των διεργασιών (main.asm).

Η πρώτη διεργασία

Η πρώτη διεργασία απαρτίζεται απο τα προγράμματα DaqInt.asm και TimeDom.asm. Η διεργασία αυτή εκτελείται μέσω ρουτίνας εξυπηρέτησης διακοπών. Ο λόγος είναι ότι οι λειτουργίες εύρεσης της ενέργειας και του ρυθμού μετάβασης από το μηδέν, πρέπει να είναι συγχρονισμένες με το ρυθμό δειγματοληψίας. Αυτό προκύπτει από το ότι θέλουμε να υπολογίζουμε αυτές τις ποσότητες με την εισαγωγή κάθε νέου δείγματος. Έτσι λοιπόν με την είσοδο ενός δείγματος, εκτελούνται τα υποπρογράμματα υπολογισμού ενέργειας και ρυθμού μετάβασης από το μηδέν. Εκτός από αυτές τις λειτουργίες, στη διακοπή της δειγματοληψίας εκτελείται και ο έλεγχος για την εύρεση της αρχής και του τέλους της λέξης. Τέλος το δείγμα αφού χρησιμοποιηθεί αποθηκεύεται για παραιτέρω επεξεργασία.

Οι ρουτίνες εγκατάστασης και εξυπηρέτησης των διακοπών βρίσκονται στο αρχείο DaqInt. Εκεί καθορίζονται οι παράμετροι επικοινωνίας με το A/D, η χρήση ή όχι των διακοπών και η ενημέρωση του λειτουργικού συστήματος σε περίπτωση κάποιου σφάλματος σχετικά με την αποστολή και τη λήψη των δεδομένων στους αναλογικοψηφιακούς μετατροπείς.

Αναλυτική περιγραφή

Η δειγματοληψία

Η δειγματοληψία του συστήματος όπως έχει ήδη αναφερθεί είναι 10 KHz, μιας και οι ακουστικές συχνότητες που μας ενδιαφέρουν έχουν συχνότητες μέχρι 5KHz. Αυτό καθορίζεται από το υλικό δίνοντας ένα εξωτερικό ρολόι στο EVB56ADC16 με συχνότητα 128 φορές μεγαλύτερη από αυτή που θέλουμε να δειγματοληπτήσουμε. Αυτή η συχνότητα δειγματοληψίας έχει περίοδο 100μS.

Η πόρτα SSI (Synchronous Serial Interface) που χρησιμοποιείται για την επικοινωνία με το A/D διαμορφώνεται ώστε να έχει λέξεις μήκους 16-bits, χωρίς διαίρεση ρολογιού, χρονισμός από εξωτερική πηγή ρολογιού, λήψη δεδομένων από το περισσότερο σημαντικό ψηφίο προς το λιγότερο σημαντικό ψηφίο και σύγχρονη λήψη/αποστολή (βλ. τα προγράμματα στο Παράρτημα).

Η δειγματοληψία δεν πραγματοποιείται με polling γιατί θα χάναμε πολύτιμο χρόνο περιμένοντας το κάθε δείγμα. Αντίθετα χρησιμοποιούμε το μηχανισμό των διακοπών. Τη δειγματοληψία την αναλαμβάνει η ρουτίνα εξυπηρέτησης διακοπών (ΡΕΔ) που λαμβάνει τα δεδομένα από το A/D. Αυτή πρέπει να λάβει το νέο δείγμα από τον A/D, να το αποθηκεύσει, να εκτελέσει τις λειτουργίες του TimeDom και κατόπιν να τελειώσει. Το δείγμα, αφού αποθηκευτεί, θα χρησιμοποιθεί μέσα στη διακοπή από τις λειτουργίες του TimeDom (ενέργεια, ρυθμός μετάβασης από το μηδέν), και κατόπιν, μετά το τέλος της διακοπής, από το FFT.

Η προσωρινή μνήμη που χρησιμοποιούμε είναι αρκετά μεγάλη και την χρησιμοποιούμε ώς ένα κυκλικό buffer. Υπάρχει ένας δείκτης που δείχνει την επόμενη ελεύθερη θέση όπου μπορεί να γραφτεί ένα νέο δείγμα, ο οποίος είναι γνωστός σε όλες τις ενδιαφερόμενες συναρτήσεις (global) για να γνωρίζουν που βρίσκεται το τελευταίο δείγμα. Ο buffer πρέπει να έχει αρκετό μήκος γιατί από εκεί ανιχνεύεται η αρχή και το τέλος μιας λέξης οπότε χρειάζονται αρκετά mS πριν τον εντοπισμό της αρχής μιας λέξης για να την προλάβουμε όλη, όπως επίσης και το FFT χρειάζεται μεγάλο αριθμό δεδομένων για να εκτελεστεί (256 σημεία). Αν το FFT είναι 256 σημείων τότε ο buffer εισόδου πρέπει να έχει αρκετό μήκος ώστε τα δεδομένα να μην καταστραφούν από νέα δείγματα. Αυτό σημαίνει ότι απαιτείται να έχει μήκος πάνω από 512 λέξεις για ασφάλεια, ώστε να μήν υπάρχει η περίπτωση να σβηστούν τα παλαιότερα δείγματα καθώς έρχονται τα καινούρια. Επίσης, για την ανίχνευση του τέλους της λέξης χρειάζονται γύρω στα 150mS που αντιστοιχούν σε 1500 δείγματα με δειγματοληψία 10KHz. Για το λόγο αυτό επιλέχτηκε ένα μέσο μήκος 1024 δειγμάτων το οποίο είναι ικανοποιητικό για την εφαρμογή μας.

Επεξεργασία στο πεδίο του χρόνου

Ο υπολογισμός της ενέργειας και του ρυθμού μετάβασης από το μηδέν γίνονται σε ένα παράθυρο 128 σημείων το οποίο “κυλάει” στον κεντρικό buffer της δειγματοληψίας. Η κύλιση αυτή γίνεται με την είσοδο κάθε νέου δείγματος.

Υπολογισμός ενέργειας

Ο υπολογισμός της ενέργειας βασίζεται στη σχέση που παρουσιάστηκε στο κεφάλαιο 3 για τη μέση ενέργεια. Βλέπουμε λοιπόν ότι ο DSP επεξεργαστής δεν έχει παρά να πάρει το άθροισμα των τετραγώνων των 128 δειγμάτων και να βρεί το μέσο όρο τους. Αυτό απαιτεί 128 μεταφορές δεδομένων από την εξωτερική μνήμη προς τον επεξεργαστή. Δηλαδή ο βρόγχος υπολογισμού της ενέργειας χρειάζεται γύρω στα $128*t_{cyc}*2*2=25\mu S$. Η όλη εκτέλεση της ΡΕΔ σύμφωνα με τις μετρήσεις που είχαμε στον παλμογράφο ήταν 40μS.

Υπολογισμός του ρυθμού διάβασης από το μηδέν

Ο αλγόριθμος που υπολογίζει το ρυθμό διάβασης από το μηδέν έχει πιό πολλές εντολές μέσα στον κύριο βρόγχο, αφού ο υπολογισμός του προσήμου στα δείγματα δεν είναι υπόθεση μιάς πρόσθεσης ή ενός πολλαπλασιασμού. Παρ' όλο που ακούγεται αστείο, ο υπολογισμός του ρυθμού διάβασης από το μηδέν καθυστερεί πιο πολύ από τον υπολογισμό της μέσης ενέργειας. Αυτό οφείλεται στό ότι στο DSP υπάρχει εντολή πολλαπλασιασμού και συσσώρευσης που εκτελείται σε ένα κυκλο μηχανής, ενώ δεν υπάρχει έτοιμη εντολή ανίχνευσης προσήμου (μην τα θέλουμε όλα δικά μας..). Στον παλμογράφο είδαμε λοιπόν ότι ο υπολογισμός του ρυθμού διάβασης από το μηδέν ήταν 80μS.

Βελτιστοποίηση

Ο συνολικός χρόνος εκτέλεσης της ΡΕΔ υπολογίζοντας μόνο της βασικότερες παραμέτρους του σήματος ήταν 40μS+80μS=120μS > 100μS. που είναι ο ρυθμός δειγματοληψίας. Αυτό αμέσως απόκλεισε τη δυνατότητα οποιουδήποτε άλλου υπολογισμού αφού είχαμε υπερβεί το μέγιστο επιτρεπτό χρονικό όριο. Μα είναι δυνατόν ένας DSP επεξεργαστής στα 27MHz να μην μπορεί να εκτελέσει 2 απλές εργασίες σε ποιο στενά χρονικά περιθώρια; Δηλαδή ένα PC με 586/100 μπορεί να ξεπεράσει σε ταχύτητα ένα DSP;

Το πρόβλημα ήταν προφανώς στο ότι τα δεδομένα φυλάσσονται σε εξωτερική μνήμη. Επειδή η εξωτερική μνήμη χρησιμοποιεί τους ίδιους διαύλους για το πρόγραμμα και τα δεδομένα, η αρχιτεκτονική Harvard δεν ισχύει γι' αυτές. Έτσι μειώνεται η ταχύτητα επεξεργασίας του επεξεργαστή, αφού υπάρχει μποτιλιάρισμα στους εξωτερικούς διαύλους, αποδεικνύοντας για άλλη μια φορά ότι η είσοδος/έξοδος μπορούν να επιφέρουν απαράδεκτες καθυστερήσεις σε ένα σύστημα.

Έχοντας εξανεμίσει τα πλεονέκτηματα του DSP με τη χρήση της εξωτερικής μνήμης, αποφασίσαμε να αναδιοργανώσουμε τη χρήση της εσωτερικής μνήμης. Φτιάξαμε μια περιοχή με μεταβλητές και σημαίες όπου φυλάσσονται στην εσωτερική μνήμη του επεξεργαστή, έτσι ώστε να εκμεταλλευόμαστε στο έπακρο την εσωτερική αρχιτεκτονική του. Με αυτό το τρόπο μειώσαμε το συνολικό χρόνο εκτέλεσης της ΡΕΔ σε 80μS. Όμως αυτό δεν ήταν αρκετό. Τα 20μS δεν θα μας έφταναν για τις υπόλοιπες λειτουργίες.

Το δεύτερο βήμα βελτιστοποίησης απαιτούσε μαγκαϊβερική εξυπνάδα. Η λύση ήταν τόσο απλή όσο και η κατασκευή κινητήρων αντιΰλης. Αντί να υπολογίζουμε με την είσοδο του κάθε δείγματος ένα παράθυρο 128 σημείων δεν έχουμε παρά προσθέσουμε σε μια σταθερή ποσότητα (που αντιπροσωπεύει την τρέχουσα ενέργεια) τη μέση ενέργεια του νέου δείγματος, και κατόπιν να αφαιρέσουμε τη μέση ενέργεια του δείγματος που θα φύγει από το παράθυρο. Αυτό με τη γλώσσα των μαθηματικών εξηγείται ως εξής:

$$E(k)=\frac{1}{N}\sum_{i=1}^{128}x(i)^2=$$
$$=\left(\frac{1}{N}\sum_{i=1}^{127}x(i)^2\right)+\frac{1}{N}x(128)^2\Rightarrow$$
$$E(k)=\left(\frac{1}{N}\sum_{i=0}^{127}x(i)^2\right)+\left(\frac{1}{N}x(128)^2\right)-\left(\frac{1}{N}x(0)^2\right)$$

Με βάση τα επιμέρους αθροίσματα της τελευταίας σχέσης, η μέση ενέργεια μπορεί να γραφεί και

$$E(k)=E(k-1)+\frac{x(k)^2}{N}-\frac{x(k-128)^2}{N}$$
$$\mu\varepsilon$$
$$E(0)=0$$

Κατά την ίδια αναλογία το ίδιο συμβαίνει και με το ρυθμό μεταγωγής από το μηδέν. Δηλαδή στον τρέχον ρυθμό ελέγχουμε αν το νέο δείγμα έχει διαφορετικό πρόσημο από το προηγούμενο του. Αν ναί, αυξάνουμε κατά ένα το ρυθμό μετάβασης από το μηδέν. Αλλιώς, προχωράμε στο επόμενο βήμα όπου ελέγχουμε αν το δείγμα που θα φύγει από το παράθυρο έχει διαφορετικό πρόσημο από το επόμενο του. Αν ναι, μειώνουμε το ρυθμό διάβασης από το μηδέν, αλλιώς δεν κάνουμε τίποτα.

Τα αποτελέσματα μετά τις βελτιστοποιήσεις είναι κάτι παραπάνω απο δραματικά. Ο τελικός χρόνος εκτέλεσης της βελτιστοποιημένης ΡΕΔ είναι 8.2μS περίπου. Δηλαδή έχουμε μια βελτίωση της τάξης του 90% σε σχέση με την αρχική εκτέλεσή του. Έτσι, τώρα το σύστημα έχει άφθονο υπολογιστικό χρόνο να ασχοληθεί και με τα άλλα καθήκοντά του.

Η αρχή και το τέλος της λέξης εντοπίζονται με βάση τη μέση ενέργεια και το ρυθμό διάβασης από το μηδέν. Η αρχή της λέξης γίνεται μόλις η ενέργεια ξεπεράσει ένα κατώφλι, οπότε μαρκάρεται η αρχή της, έχοντας ρυθμίσει έτσι τους δείκτες, ωστε η λέξη ξεκινάει μερικά μιλιδευτερόλεπτα πιο νωρίς απο την στιγμή που ξεπεράστηκε το κατώφλι, για να μην χάσουμε την αρχή της λέξης. Το τέλος ανιχνεύεται με βάση την "ησυχία" στην είσοδο του συστήματος για μεγάλο χρονικό διάστημα (150-200 mS). Η "ησυχία", βέβαια, δέν μπορεί να είναι το μηδέν, γιατί στην είσοδο των δειγμάτων έχουμε συνεχώς σήμα. Έτσι, η "ησυχία", είναι μια μικρή περιοχή τιμών γύρω απο το μηδέν. Η όλη λειτουργία της ανίχνευσης αρχής-τέλους, θυμίζει την λειτουργία ενός schmitt-trigger που έχει δύο διαφορετικά κατώφλια για την ενεργοποίηση και την απενεργοποίηση της εξόδου του. Αυτό ακριβώς θέλαμε να πετύχουμε και εμείς, ώστε το σύστημα να μήν έχει διαστήματα "δισταχτικότητας" σχετικά με το άν θα μαρκάρει ή όχι την αρχή, ή το τέλος, μιας νέας λέξης. Έτσι απορρίπτονται οι περισσότερες απο τις λανθασμένες εκκινήσεις που θα είχαμε άν χρησιμοποιούσαμε ένα μόνο κατώφλι σύγκρισης.

Η δεύτερη διεργασία

Η δεύτερη διεργασία αποτελείται απο τα προγράμματα fftr2cc.asm και FreqDom.asm. Σαν στόχο της έχει τη μετάβαση από το πεδίο του χρόνου στο πεδίο των συχνοτήτων. Οι εργασίες που εκτελούνται σε αυτό το υποπρόγραμμα είναι το

μάζεμα των δεδομένων ώστε να μπορέσει να εκτελεστεί το FFT, ο πολλαπλασιασμός με τη συνάρτηση παραθύρου και η εκτέλεση του FFT. Τα δεδομένα της εξόδου βρίσκονται σε bit-reversed μορφή.

Αναλυτική περιγραφή

Σύμφωνα με τον νέο αλγόριθμο αναγνώρισης φωνής που αναπτύξαμε, η μετάβαση από το πεδίο του χρόνου στο πεδίο των συχνοτήτων γίνεται κατά τη διάρκεια της δειγματοληψίας. Η μετάβαση αυτή μπορεί να γίνει με δυο τρόπους. Ο πρώτος είναι να χρησιμοποιήσουμε ζωνοπερατά φίλτρα και να υπολογίζουμε την ενέργεια σε κάθε φίλτρο στην έξοδο τους. Η άλλη λύση είναι η χρήση FFT.

Μετάβαση στο πεδίο των συχνοτήτων με φίλτρα

Στις εφαρμογές όπου χρησιμοποιείται DSP προτιμάται η χρήση FIR φίλτρων, μιάς και αυτά έχουν γραμμική φάση και είναι πάντοντε υλοποιήσιμα. Αντίθετα, τα IIR φίλτρα συναντώνται πιο σπάνια. Το σαφέστατο πλεονέκτημα των IIR φίλτρων είναι η μικρή τάξη που απαιτείται σχετικά με τα FIR φίλτρα για να υλοποιήσουν φίλτρα με τις ίδιες προδιαγραφές. Για τη δική μας εφαρμογή χρειαζόμασταν 19 ζωνοδιαβατά φίλτρα με μεγάλη κλίση, γιατί βρίσκονται σε πολύ κοντινή απόσταση μεταξύ τους. Λόγω της τελευταίας αυτής προδιαγραφής αποκλείσαμε τα FIR φίλτρα, αφού η υλοποίηση 19 τέτοιων φίλτρων απέκλειαν την περίπτωση υλοποίησης σε αυτό το DSP τουλάχιστον, αφού θα ήταν τόσο μεγάλου βαθμού, που ο επεξεργαστής δεν θα προλάβαινε την εκτέλεση τους. Έτσι στραφήκαμε στα IIR φίλτρα. Κάνοντας μερικές δοκιμές με το Matlab καταλήξαμε σε ελλειπτικά φίλτρα 8 με 10 βαθμού.

Γράψαμε στο Matlab δυο προγράμματα για τον υπολογισμό των συντελεστών των φίλτρων, την απεικόνιση της απόκρισής τους και την δημιουργία κώδικα για το 56001. Έτσι υπολογίζαμε το φίλτρο που θέλαμε (IIR/FIR) και είχαμε κατ' ευθείαν τον κώδικα για το DSP όπου μπορούσαμε να το δοκιμάσουμε. Οι δοκιμές έγιναν με παλμογράφο και γεννήτρια ημιτόνου. Όμως για καλή μας τύχη τα φίλτρα δεν εργάζοταν σωστά, παρά τα αντίθετα λεγόμενα του Matlab. Κάνοντας διάφορα πειράματα και αλχημίες, βρήκαμε και μερικά σφάλματα - bugs του Matlab (όπως ότι η δημιουργία του ημιτόνου επηρεάζεται από το αν θα γράψεις sin(2*pi*t) ή sin(2*3.14159*t) !!). Τα φίλτρα στο DSP εργάζοταν σε άλλες συχνότητες. Παρατηρήσαμε ότι στα βαθυπερατά φίλτρα όπου τα είχαμε σχεδιάσει να κόβουν στα 5KHz, το γόνατο ήταν στα 6KHz. Τα ζωνοπερατά φίλτρα είχαν γίνει τιράντες, αφού η συχνότητα διάβασης ήταν πολλά Hz πιο πέρα από εκεί που έπρεπε. Το μυστήριο παρέμενε παρά τους συνεχείς ελέγχους σε όλα τα προγράμματα (Matlab και DSP).

Τελικώς με τη χρήση της αντίστροφης μηχανικής πετύχαμε διάνα. Η λογική ήταν η εξής. Σχεδιάσαμε ένα βαθυπερατό φίλτρο με συχνότητα αποκοπής 5KHz. Ο κώδικας που προέκυψε φορτώθηκε στο DSP καιι εκτελέστηκε με καραμπίνα. Μετράμε με τον παλμογράφο που παρουσιάζοταν το γόνατο. Ήταν περίπου στα 6KHz.

Ερώτηση: Πόση πρέπει να είναι η δειγματοληψία μου για να έχω 6KHz στο γόνατο;

Απάντηση:

$$\alpha = \frac{6000 - 5000}{5000} = 0.2$$
$$f_{snew} = f_s \cdot a + f_s$$

$$f_s = 44100Hz \Rightarrow f_{snew} = 52920Hz$$

Ξανασχεδιάζουμε το φίλτρο με τη νέα δειγματοληψία και ξανατρέχουμε τον κώδικα στο DSP. Συχνότητα αποκοπής: 5KHz. Άρα ήταν φανερό ότι υπήρχε κάποιο πρόβλημα με το ρυθμό δειγματοληψίας. Πολύ ωραία, αλλά ο ρυθμός δειγματοληψίας ορίζεται από το υλικό και το υλικό ΔΕΝ αλλάζει. Ιδού και η απόδειξη. Παίρνουμε τον παλμογράφο και μετράμε το ρολόι του αναλογικοψηφιακού μετατροπέα. Αχά .....!!!!!;;;; 20% απόκλιση από τη συχνότητα που έπρεπε να έχει; Αμέσως μετράμε τον κρύσταλο του ταλαντωτή και ο παλμογράφος μέτραγε 6.77MHz αντί για 5.6448.MHz Και αυτό το σφάλμα υπήρχε και στους τρεις κρυστάλους. Πως είναι δυνατόν τρεις κρύσταλοι να ταλαντώνουν 20% μακρύτερα, από εκεί που είναι κατασκευασμένοι να εργάζονται; Ή οι κρύσταλοι είναι από ποτήρι ή .... ο παλμογράφος έχει πρόβλημα. Και όντως, κάνοντας έναν πρόχειρο έλεγχο ανακαλύψαμε ότι ο παλμογράφος μας είχε απόκλιση 20% από την πραγματική συχνότητα. Η απόκλιση αυτή δημιουργήθηκε από μια επισκευή όπου κατόπιν το όργανο δεν ξαναρυθμίστηκε, καθυστερώντας μας ένα με ενάμιση μήνα ψάχνοντας να βρούμε τι στο καλό είχαν πάθει τα φίλτρα, και σκεπτόμενοι ότι θα έπρεπε να ασχολούμαστε με τίποτα πιο εύκολο, όπως η οδοκαθαριστική.

Μετάβαση στο πεδίο των συχνοτήτων με FFT

Αφού λύσαμε τα προβλήματα με την υλοποίηση των φίλτρων, διαπιστώσαμε ότι ακόμη και τα IIR φίλτρα χρησιμοπούσαν αρκετή υπολογιστική ισχύ. Τότε ωρίμασε η σκέψη και αποφασίσαμε να δοκιμάσουμε να μεταβούμε στο πεδίο των συχνοτήτων με FFT. Σύμφωνα με τους θεωρητικούς μας υπολογισμούς, η εκτέλεση των φίλτρων είναι πολύ πιο αργή διαδικασία από ένα FFT. Όμως πόσα σημεία θα έπρεπε να έχει το FFT ώστε και να μην κάνουμε πολλούς υπολογισμούς, υπολογίζοντας ενέργεια συχνοτήτων με υπερβολική ακρίβεια, ενώ ταυτόχρονα να υπάρχουν αρκετά σημεία ώστε να μπορέσουμε να ταιριάξουμε τις απαιτούμενες συχνότητες με αυτές που θα προέκυπταν από το FFT; Και δεν ήταν μόνο αυτό. Όσο μεγάλωνε το FFT τόσο μεγάλωνε ο χρόνος του χρονικού κβάντου οπότε χάναμε σε ακρίβεια στο πεδίο του χρόνου για τις χρονικές ζώνες. Αυτό φαίνεται από το ότι ένα FFT με 256 σημεία με συχνότητα δειγματοληψίας 10KHz μας δίνει φάσματα σε χρονικά διαστήματα των 25mS, ενώ ένα FFT με 1024 σημεία αντίστοιχα 100mS. Το FFT με τα 256 σημεία φάνηκε να είναι η ιδανική λύση για μας, αφού συνδιάζει ικανοποιητική ακρίβεια και στο πεδίο του χρόνου (20mS είναι ο μέσος όρος της διάρκειας ενός φθόγγου) και στο πεδίο των συχνοτήτων.

Τέλος όπως αναφέρθηκε και στο αντίστοιχο κεφάλαιο που ασχολείται με τα χαρακτηριστικά της φωνής, όταν γίνεται τμηματική στο πεδίο του χρόνου ανάλυση, είναι προτιμότερο το κάθε δείγμα να συμμετέχει σε δυο τμήματα, ώστε να μη χαθούν κάποια χαρακτηριστικά από το σήμα. Γι' αυτό το λόγο και εμείς τα FFT τα κάνουμε σε παράθυρα που αλληλοεπικαλύπτονται, ώστε να μην έχουμε απώλειες κάποιων χαρακτηριστικών που μπορούν να αλλοιώσουν το τελικό αποτέλεσμα.

Η τρίτη διεργασία

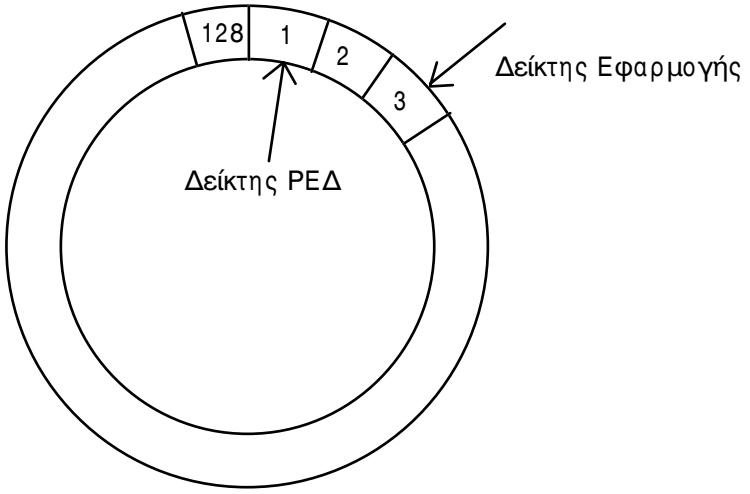
Η τρίτη διεργασία αποτελείται απο τα προγράμματα RS232int.asm και ProcData.asm. Η τρίτη διεργασία είναι υπεύθυνη για την ομαδοποίηση των φασμάτων στο πεδίο του χρόνου και στο πεδίο των συχνοτήτων. Μόλις μαζευτεί ο πίνακας των 304 στοιχείων που αποτελούν το χαρακτηριστικό διάνυσμα της λέξης που δειγματοληπτήθηκε, εκτελείται η μεταφορά των δεδομένων στο PC μέσω της σειριακής θύρας.

Αναλυτική περιγραφή

Η σειριακή θύρα

Σε ένα σύστημα μικροεπεξεργαστή όπου τα bits τρέχουν και δεν φτάνουν, τα αργότερα σύστηματα συνήθως είναι τα κυκλώματα εισόδου/εξόδου. Έτσι όταν επιθυμούμε τη μέγιστη ταχύτητα επεξεργασίας από έναν μικροεπεξεργαστή καλόν είναι να αποδεσμεύουμε το κυρίως πρόγραμμα από τις καθυστερήσεις της επικοινωνίας, ώστε να μην υπάρχουν άσκοπες καθυστερήσεις. Στη συγκεκριμένη περίπτωση, η σειριακή θύρα χρησιμοποιείται πολύ αραιά, κυρίως για να δοθούν εντολές από το PC προς το DSP, και για τη μεταφορά των χαρακτηριστικών διανυσμάτων. Εκτός αυτού, η διάρκεια της μεταφοράς ενός χαρακτήρα είναι πολύ μεγάλη σε σχέση με το χρόνο εκτέλεσης μιας εντολής. Αυτό μας οδήγησε να γράψουμε τις ρουτίνες μεταφοράς με τη χρήση των διακοπών. Οι ρουτίνες σειριακής μεταφοράς δεδομένων αποτελούνται από δυο τμήματα. Το πρώτο τμήμα είναι η ρουτίνα εξυπηρέτησης των διακοπών για αποστολή και λήψη, ενώ το δεύτερο τμήμα είναι υπορουτίνα που εκτελείται στο κυρίως πρόγραμμα για την μεταφορά και τη λήψη των δεδομένων. Πρακτικά υλοποιούμε ένα buffer FIFO (First In- First Out) με λογισμικό, ώστε το κυρίως πρόγραμμα να μην περιμένει την αποστολή του κάθε χαρακτήρα για να συνεχίσει. Ο τρόπος λειτουργίας για τη ρουτίνα αποστολής δεδομένων είναι ο ακόλουθος.

Υπάρχει μια ενδιάμεση περιοχή μνήμης (buffer) όπου λειτουργεί με κυκλικό τρόπο (circular buffer). Αυτό σημαίνει ότι αν η περιοχή αυτή γεμίσει, η επόμενη λέξη που θα έρθει θα γραφτεί στη θέση μνήμης που γράφτηκε πρώτη. Η λειτουργία του φαίνεται στο παρακάτω σχήμα 4.11.



Σχ. 4.11. *Η λειτουργία ενός κυκλικού buffer.*

Πάνω στον κυκλικό buffer δείχνουν δυο δείκτες. Ο ένας δείκτης ανήκει στην ρουτίνα εξυπηρέτησης των διακοπών, και δείχνει στο επόμενο δεδομένο που πρέπει να σταλεί. Ο δεύτερος δείκτης ανήκει στη ρουτίνα που καλεί η εφαρμογή, και δείχνει στην επόμενη ελεύθερη θέση όπου μπορεί να αφήσει ένα καινούργιο δεδομένο. Σε κάθε διακοπή "τέλους αποστολής", η ρουτίνα εξυπηρέτησης ελέγχει να δεί αν οι δυο δείκτες συμπίπτουν. Αν ναί, τότε δεν υπάρχουν δεδομένα για αποστολή οπότε απενεργοποιείται η διακοπή μέχρι να έρθουν νέα δεδομένα από την εφαρμογή. Αν οι δυο δείκτες δεν συμπίπτουν, τότε η ρουτίνα εξυπηρέτησης της διακοπής στέλνει ένα δεδομένο στην σειριακή έξοδο και αυξάνει κατά ένα το δείκτη της, μειώνοντας τον αριθμό των δεδομένων που αναμένουν στον buffer για αποστολή. Αντίστοιχα η εφαρμογή ενεργοποιεί τις διακοπές του αποστολέα όταν υπάρχει ένα νέο δεδομένο για εγγραφή. Παρόμοια είναι και η διαδικασία της λήψης, μόνο που η ΡΕΔ (ρουτίνα εξυπηρέτησης διακοπών) γράφει τα δεδομένα που έρχονται από τη σειριακή θύρα στο buffer, ενώ η εφαρμογή διαβάζει τα δεδομένα.

Με αυτό τον τρόπο έχουμε καταφέρει την ασύγχρονη μεταφορά των δεδομένων από την εφαρμογή στα περιφερειακά συστήματα. Η εφαρμογή μπορεί να γράψει με την δική της ταχύτητα, μεμιάς, έναν αριθμό δεδομένων (μέχρι το μήκος του buffer) και κατόπιν να συνεχίσει την εκτέλεση υπολογισμών, ενώ τότε, η σειριακή να αρχίσει να στέλνει τα δεδομένα κούτσα-κούτσα στο PC.

Η σειριακή αποφασίστηκε να δουλέψει στα 9600 Baud με ένα Start bit, ένα Stop bit, οκτώ bits δεδομένων, και ζυγό parity (11 bit async SCI DSP mode 4). Η διασύνδεση γίνεται μέσα από το SCI (Serial Communications Interface).

Η διακοπή για την ΡΕΔ της αποστολής εκτελείται από τον επεξεργαστή μόλις σταλεί ένα δεδομένο και ο καταχωρητής αποστολής είναι άδειος. Η διακοπή για την ΡΕΔ της λήψης εκτελείται από τον επεξεργαστή μόλις έρθει ένα δεδομένο, οπότε ο καταχωρητής λήψης είναι γεμάτος.

Η προτεραιότητα των διακοπών της σειριακής έχει τεθεί υψηλότερα από τις αντίστοιχες διακοπές του SSI που επικοινωνεί με τον A/D. Ο λόγος είναι ότι η σειριακές ΡΕΔ είναι πολύ μικρές σε σχέση με τις αντίστοιχες του A/D που έχουν μεγάλο χρόνο εκτέλεσης, οπότε μας συμφέρει να μην καθυστερεί η εξυπηρέτηση του ταχύτερου προγραμμάτος τη στιγμή που βρισκόμαστε σε διακοπή.

Η τελική επεξεργασία

Η τελική επεξεργασία των δεδομένων γίνεται από την ProcData. Οι διαδιακασίες που εκτελούνται εκεί είναι η αντιγραφή των αποτελεσμάτων του FFT από bit-reversed σε γραμμική μορφή, η ομαδοποίηση στο πεδίο των συχνοτήτων και η ομαδοποίηση στο πεδίο του χρόνου.

Η πρώτη λειτουργία είναι πολύ εύκολη λόγω της δυνατότητας του DSP μέσω υλικού να προσπελάσει με bit-reversed σειρά ένα buffer και να αντιγράψει τα δεδομένα με γραμμικό τρόπο.

Η ομαδοποίηση στο πεδίο των συχνοτήτων γίνεται με τη βοήθεια πινάκων που περιέχουν την αρχή και το πλάτος κάθε ζώνης συχνοτήτων που μας ενδιαφέρει, σε σημεία του FFT. Δηλαδή η πρώτη ζώνη ξεκινάει από το πέμπτο στοιχείο του FFT, και τελειώνει στο ένατο όπως φαίνεται από τους πίνακες StartOfBands και AveragePoints στο αρχείο ProcData.

Στο πεδίο του χρόνου τα πράγματα είναι πιο δύσκολα, αφού το μήκος της λέξης είναι μεταβλητό. Έτσι πρέπει να υπολογίσουμε πόσα πλαίσια έχει κάθε χρονική ζώνη (επειδή συνήθως δεν βγαίνει ακριβώς κάνουμε στρογγυλοποίηση) και κατόπιν μετά

την άθροιση βρίσκουμε το μέσο όρο. Επειδή σε αυτό το σημείο έχουμε διαίρεση δεκαδικού με ακέραιο δεν μπορούμε να χρησιμοποιήσουμε απ' ευθείας την εντολή διαίρεσης. Έτσι κάνουμε πολλαπλασιασμό του δεκαδικού με δεκαδικό (1/ακέραιο). Η τιμή 1/ακεραίο βρίσκεται σε ένα πίνακα look up με τιμές από 0 ως 255.

Μόλις τελειώσουν οι ομαδοποιήσεις το DSP είναι έτοιμο να στείλει τα δεδομένα στο PC μέσω της σειριακής θύρας.

Το λειτουργικό σύστημα

Το λειτουργικό σύστημα βρίσκεται κατανεμημένο σε διάφορα αρχεία, που εξυπηρετούν το κάθε ένα και ένα ορισμένο σύστημα (πόρο) του λειτουργικού. Έχουμε λοιπόν το FPGAint.asm, που ασχολείται με την επικοινωνία με το FPGA, το Stack.asm που είναι η διαχείρηση ενός τεχνητού stack, το Main.asm που αποτελεί τον πυρήνα του κεντρικού βρόγχου της εφαρμογής, και τέλος το Vectors.asm που είναι ο πίνακς με τα διανύσματα διακοπών του επεξεργαστή.

Το λειτουργικό σύστημα είναι το κεντρικό πρόγραμμα που αναλαμβάνει τον συντονισμό των άλλων διεργασιών. Εκεί γίνονται οι έλεγχοι για το ποιές διεργασίες θα πρέπει να εκτελεστούν, και να τεθούν σημαίες για την επικοινωνία των διεργασιών. Η πολυεπεξεργασία που χρησιμοποιούμε, δεν χρησιμοποιεί χρονιστές που κάνουν διακοπή στον επεξεργαστή σε τακτά χρονικά διαστήματα όπως συμβαίνει σε κλασικά πολυεπεξεργαστικά συστήματα (Unix), αλλά ο dispatcher που κανονίζει το ποιά διεργασία θα εκτελεστεί τρέχει μόνιμα στον κύριο βρόγχο. Τέλος το λειτουργικό σύστημα ενημερώνει την κατάσταση που βρίσκεται μέσω των LED που βρίσκονται στην επέκταση μνήμης, ενώ παράλληλα μπορεί να πάρει εντολές από τους μικροδιακόπτες.

Αναλυτική περιγραφή

Η πρώτη δουλειά του λειτουργικού συστήματος είναι να φέρει το υλικό σε μια γνωστή κατάσταση και να προγραμματίσει τα περιφερειακά κυκλώματα κατά βούληση. Ρυθμίζει τους καταχωρητές των διακοπών και του εξωτερικού διαύλου. Κατόπιν προγραμματίζει τη πόρτα B σε λειτουργία εισόδου-εξόδου για να χρησιμοποιηθεί αργότερα για εκσφαλμάτωση. Προγραμματίζει τη στοίβα λογισμικού, και τις ενδεικτικές σημαίες λαθών. Η αρχικοποίηση περιλαμβάνει τη σύγχρονη επικοινωνία με το A/D, την ασύγχρονη επικοινωνία με τη σειριακή θύρα, την επικοινωνία με το FPGA, και τα FFT. Αν υπάρξει κάποιο πρόβλημα στο λειτουργικό κατά την εκτέλεση της εφαρμογής, το λάθος αποστέλεται στη σειριακή θύρα.

Ο ελεγκτής της ροής προγράμματος ελέγχει τις σημαίες για κάθε διεργασία. Αν η υπό έλεγχο διεργασία δεν έχει τίποτα να κάνει, τότε αυτομάτως αναβάλεται η εκτέλεση της για τον επόμενο κύκλο του βρόγχου, και ελέγχεται η επόμενη διεργασία. Με αυτό το τρόπο δεν σπαταλάμε το χρόνο του επεξεργαστή σε άσκοπες λειτουργίες, αφού άλλωστε ποτέ δεν μπορεί να τύχει η ταυτόχρονη επεξεργασία όλων των διεργασιών. Η λειτουργία αυτή θυμίζει έντονα λειτουργικό σύστημα με πολυεπεξεργασία. Η κύρια διαφορα είναι ότι δεν χρησιμοποιούνται χρονιστές για τεμαχισμό και μοίρασμα του χρόνου στην κάθε διεργασία. Αυτό συμβαίνει διότι η εφαρμογή που κατασκεύασαμε είναι εξειδικευμένη, οπότε αφ' ενός ξέρουμε εκ των προτέρων ποιές είναι οι διεργασίες και αφ' ετέρου οι απαιτήσεις μας σε υπολογιστική ισχύ είναι μεγάλες, με αποτέλεσμα το παραμικρό nS να μας είναι απαραίτητο. Εκτός αυτού, η υλοποίηση είναι πιο εύκολη κατ' αυτό τον τρόπο, πράγμα που βοηθάει στην ταχύτερη υλοποίηση του αλγορίθμου αναγνώρισης ομιλίας. Κατά τα άλλα, ο έλεγχος των σημαίων και η αναβολή της εκτέλεσης κάποιας διεργασίας θυμίζει pre-emptive multitasking όπου οι διεργασίες "κοιμούνται" όσο χρόνο δεν έχουν να κάνουν τίποτε.

Ένα άλλο σημείο ομοιότητας του ένθετου αυτού λειτουργικού συστήματος, είναι το σύστημα ενημέρωσης της κάθε διεργασίας από κάποια άλλη με την ενεργοποίηση κάποιων bit - σημαίων. Έτσι υλοποιείται ένα απλοϊκό σύστημα επικοινωνίας (messaging system - intertask communication) σε αντίθεση με τη χρήση σηματοφόρων που χρησιμοποιούνται στα λειτουργικά συστήματα του εμπορίου.

Βέβαια οι διαφορές αυτές οφείλονται στη φύση και το είδος της εφαρμογής σε σχέση με τα λειτουργικά συστήματα γενικού σκοπού. Σε γενικές γραμμές, αυτό το απλό σε σχέση με τα κλασσικά λειτουργικά συστήματα πολυεπεξεργασίας, λειτουργικό σύστημα, ανταποκρίνεται τέλεια στις απαιτήσεις της εφαρμογής μας. Η απλότητά του επιτρέπει την εύκολη μετατροπή ή προσθήκη τμημάτων προγράμματος χωρίς πολύ κόπο.

Μοντέλο προγραμματισμού

Σε αυτό την εφαρμογή έχουμε θεωρήσει ότι οι παρακάτω καταχωρητές χρησιμοποιούνται ως γενικές μεταβλητές σε όλο το πρόγραμμα.

R3: Δείκτης στην επόμενη ελεύθερη θέση στο buffer των δειγμάτων
R7: Δείκτης στοίβας λογισμικού
N7: Ένδειξη τελευταίου σφάλματος

Οι ονομασίες στις ταμπέλες έχουν την εξής σημασία :
i_xxxx : ΡΕΔ (ISR : Interrupt Service Routine)
f_xxxx : Συνάρτηση (Function)
p_xxxx: Δείκτης (Pointer)
buf_xxx: Buffer
v_xxxx: Μεταβλητή (Variable)
c_xxxx: Σταθερά (Constant)
b_xxxx: Bit
_xxxx : Τοπική μεταβλητή

Ανάλυση Αποτελεσμάτων

Μετά την κατασκευή του υλικού και την συγγραφή των προγραμμάτων, ήμαστε πιά σε θέση να πάρουμε τα πρώτα αποτελέσματα. Βέβαια, υπήρξαν μερικά ακόμη μικροπροβλήματα στον κώδικα του DSP, αλλά γρήγορα τα βρήκαμε και τα διορθώσαμε. Έτσι, χρησιμοποιώντας είτε το πρόγραμμα επικοινωνίας που έχουμε φτιάξει, είτε κατευθείαν το πρόγραμμα του ADS, μπορούμε να πούμε μια λέξη στο μικρόφωνο και κατόπιν να παραλάβουμε το φασματόγραμμά της στο PC. Εμείς χρησιμοποιήσαμε το Matlab για να κάνουμε την επεξεργασία και απεικόνιση των αποτελεσμάτων, γιατί εκεί ο κώδικας γράφεται πιό εύκολα και γρήγορα. Επίσης, δέν χρειάζεται να ασχοληθούμε με την διαχείρηση δομών στην μνήμη του υπολογιστή, ούτε με την διαχείρηση αρχείων στο δίσκο, δύο από τις πιο χρονοβόρες και δύσκολες λειτουργίες που πρέπει να κάνει το λογισμικό του PC.

Αποτελέσματα βελτιστοποιήσεων

Πρίν ασχοληθούμε με την καθεαυτό αναγνώριση, κάναμε ένα συγκεντρωτικό έλεγχο των βελτιστοποιήσεων που είχαμε πετύχει από πλευράς χρόνου εκτέλεσης και απαιτούμενης μνήμης.

Function	*Time*	*tw=0.8 Sec*	*tw=1 Sec*	*tw=2 Sec*	*Memory*	*8000*	*10000*	*20000*
Window		2.3 mS	3.0 mS	5.9 mS		500	625	1250
FFT		16 mS	39 mS	95 mS		2560	5120	10240
Σk/N		4.5 μS	8 μS	16 μS				
Total		18.3 mS	42.7 mS	101 mS		10560	15120	30240

Σημειώσεις:

1. Το παράθυρο (window) είναι υπολογισμένο σάν f(t)=g(t)*i(t) N φορές, όπου N το μήκος του παραθύρου. Ο απαιτούμενος χρόνος είναι ο χρόνος για N πολλαπλασιασμούς με 16 επαναλήψεις, σύν τον χρόνο των μετακινήσεων δεδομένων.
2. Ο χρόνος εκτέλεσης του FFT είναι N*log2N προσθέσεις και N/2*log2N πολλαπλασιασμοί, με 16 επαναλήψεις.
3. Χρόνος εκτέλεσης των ομαδοποιήσεων Σk/N = χρόνος εκτέλεσης N προσθέσεων.
4. Στις απαιτήσεις μνήμης συμεριλαμβάνονται οι θέσεις του buffer της δειγματοληψίας και των μεταβλητών που απαιτούνται για τους υπολογισμούς. Δέν συμπεριλαμβάνεται η μνήμη που καταναλώνεται για τα παράθυρα, γιατί ξαναχρησιμοποιείται στο FFT

Πίνακας 5.1. *Χρήση μνήμης και χρόνοι επεξεργασίας λέξεων με διαφορετικές χρονικές διάρκειες, με τον αρχικό αλγόριθμο.*

Στον πίνακα 5.1 βλέπουμε την ταχύτητα επεξεργασίας που απαιτείται από το DSP56001/27MHz για τον υπολογισμό των FFT μετά την πλήρη δειγματοληψία μιας λέξης, μέ βάση τον αρχικό αλγόριθμο. Παρατηρήστε την αύξηση των απαιτήσεων μνήμης ανάλογα με την χρονική διάρκεια της λέξης.

Function Time	tw=0.8 Sec	tw=1 Sec	tw=2 Sec	Memory 8000	10000	20000
Filtering	384 mS	480 mS	960 mS	20	20	20
Summing	1.2 mS	1.5 mS	3 mS	7600	9500	19000
Grouping	3.7 μS	4.5 μS	9.2 μS			
Total	386 mS	482 mS	964 mS	7700	9600	19100

Σημειώσεις:

1. Το κάθε ένα απο τα 19 φίλτρα είναι ένα IIR 8ης τάξης. Κάθε ένα έχει χρόνο εκτέλεσης 2.4 μS.
2. Οι αθροίσεις καταναλώνουν N προσθέσεις και 1 μετακίνηση με 19 επαναλήψεις.
3. Κάθε μετακίνηση/πρόσθεση/πολλαπλασιασμός καταναλώνει 2 κύκλους του επεξεργαστή.

Πίνακας 5.2. *Χρήση μνήμης και χρόνοι επεξεργασίας λέξεων με διαφορετικές χρονικές διάρκειες, με τον τελικό αλγόριθμο.*

Στον πίνακα 5.2 παρατηρούμε τη συμπεριφορά του νέου αλγορίθμου αναγνώρισης ομιλίας, υλοποιημένου με την τεχνική των ζωνοδιαβατών φίλτρων IIR. Οι χρόνοι εκτέλεσης του αλγορίθμου είναι σαφέστατα πιό μεγάλοι ενώ αντίθετα, η χρήση της μνήμης μειώθηκε δραματικά.

	Αρχ. μέθοδος FFT	Μέθοδος Φίλτρων
Παράλληλη επεξεργασία	(Δέν υπάρχει)	384 / 480 / 960 mS
Περαιτέρω επεξεργασία	18.3 / 42.7 / 101 mS	3.7 / 4.5 / 9.2 μS
Χρόνος απόκρισης	18.3 / 42.7 / 101 mS	3.7 / 4.5 / 9.2 μS

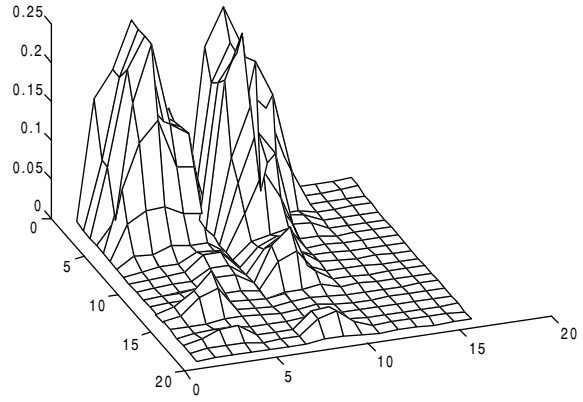
Πίνακας 5.3. *Η ταχύτητα εκτέλεσης του αρχικού αλγορίθμου σε σχέση με την ταχύτητα εκτέλεσης της παραλλαγής με φίλτρα.*

Τελικά η σύγκριση των δύο μεθόδων αποδεικνύει ότι η μέθοδος των φίλτρων είναι και ταχύτερη και απαιτεί σημαντικά μικρότερα ποσά μνήμης για την υλοποίησή της. Αυτό παρόλο που φαίνεται παράδοξο, όπως αυτό του Ζήνωνα, οφείλεται στον τρόπο διαχείρησης του χρόνου επεξεργασίας των δεδομένων από τον επεξεργαστή. Στη πρώτη περίπτωση ο επεξεργαστής την ώρα της δειγματοληψίας δεν κάνει τίποτα άλλο εκτός από το να συσσωρεύει δεδομένα στη μνήμη. Έτσι ο χρόνος επεξεργασίας προστίθεται στο χρόνο της δειγματοληψίας, που είναι σχετικά τεράστιος (όσο διαρκεί και η λέξη). Αντίθετα ο βελτιωμένος αλγόριθμος διασπείρει τον χρόνο επεξεργασίας ανάμεσα στα δείγματα.

Η βελτίωση του συστήματος όμως δεν σταματάει εδώ. Όπως βλέπουμε από τους πίνακες η εκτέλεση του FFT, είναι σαφέστερα ταχύτερη σε σχέση με την αντίστοιχη εκτέλεση των 19 ζωνοπερατών φίλτρων. Αυτό μας ώθησε στο να πραγματοποιήσουμε το πέρασμα από το πεδίο του χρόνου στο πεδίο των συχνοτήτων με τη βοήθεια των FFT (βλέπε 4ο κεφάλαιο). Έτσι, τελικά το λογισμικό του συστήματος αναγνώρισης είναι τόσο γρήγορο, που η επεξεργασία μπορεί να γίνεται σε πραγματικό χρόνο, ακόμη και με δειγματοληψίες πάνω απο 22 KHz, με το ADS56000.

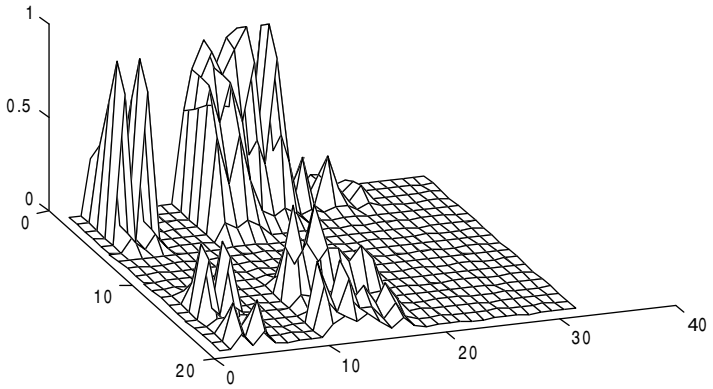
Τελικές ρυθμίσεις

Στην διάρκεια της εκτίμησης των πρώτων αποτελεσμάτων, είδαμε οτι το φασματόγραμμα εξόδου δέν είχε αρκετή διακριτότητα στο πεδίο του χρόνου, ώστε να μπορεί να ξεχωρίσει τις διαφορές στην χρονική εξέλιξη δύο λέξεων. Αυτό μείωνε πολύ την ικανότητά του να διακρίνει λέξεις που είναι παρόμοιες στην ακοή.



Σχ. 5.1. *Το αρχικό φασματόγραμμα*

Αποφασίσαμε λοιπόν, να αυξήσουμε τον αριθμό των χρονικών ζωνών από 16 σε 32. Μας συμφέρει να έχουμε τον αριθμό των ζωνών σαν δύναμη του 2, λόγω της επεξεργασίας που γίνεται στο DSP. Κάνοντας αυτή την αλλαγή, παρατηρήσαμε οτι αυξήθηκαν πολύ οι διαφορές μεταξύ των λέξεων που απλώς έμοιαζαν μεταξύ τους, ενώ δέν επηρεάστηκαν σημαντικά οι διαφορές στις όμοιες λέξεις, ώστε να μήν μειωθεί το ποσοστό επιτυχίας.



Σχ. 5.2. *Το φασματόγραμμα μετά την αύξηση των χρονικών ζωνών σε 32. Είναι φανερή η μεγάλη αύξηση της διακριτότητας.*

Ένα άλλο ελλάτωμα που παρατηρήσαμε στα γραφικά αποτελέσματα ήταν οτι, σε όλα τα φασματογράμματα, το σήμα δέν κατελάμβανε όλες τις διαθέσιμες χρονικές ζώνες, αλλά περιοριζόταν περίπου στις πρώτες 10 από τις 16 ζώνες (ή τις 20 από τις 32). Προς στιγμή, δέν μπορούσαμε να εξηγήσουμε αυτή την συμπεριφορά. Όμως, μετά από λίγο χρόνο, καταλάβαμε τί συνέβαινε. Παρατηρώντας τον κώδικα του DSP, είδαμε

οτι για να ανιχνευτεί το τέλος της λέξης, έπρεπε να ανιχνευτούν 150ms σιωπής στο σήμα εισόδου. Είχαμε ξεχάσει να αφαιρέσουμε αυτά τα επιπλέον ms από το χρήσιμο σήμα, οπότε αυτά έπιαναν τις τελευταίες χρονικές ζώνες στο αποτέλεσμα. Έτσι, το χρήσιμο σήμα περιοριζόταν στις πρώτες μόνο ζώνες. Γυρίσαμε για λίγο στο άλλο παράθυρο των windows που έτρεχε ένα command.com, και σε δευτερόλεπτα είχαμε ρυθμίσει το DSP να "πετάει" τα σκουπίδια από το τέλος της λέξης (δηλαδή τα τελευταία ms), οπότε τώρα το σήμα απλωνόταν σε όλες τις ζώνες. Ασφαλώς, αφήσαμε και ένα περιθώριο χάριτος μερικών ms, ώστε άν τυχόν υπάρχει κάποια χρήσιμη πληροφορία σε αυτό, όπως πχ το "σβήσιμο" ενός αδύναμου συμφώνου, να μήν το χάσουμε, αλλά να το συμπεριλάβουμε στα αποτελέσματα. Μετά και από αυτές τις τελευταίες ρυθμίσεις, περάσαμε στο στάδιο της εκπαίδευσης.

Εκπαίδευση

Κατά την διάρκεια της επαίδευσης, πήραμε ένα μεγάλο αριθμό επαναλήψεων από κάθε λέξη, από τον ίδιο ομιλητή. Πήραμε συνολικά τέσσερις διαφορετικές λέξεις, δύο από τις οποίες έμοιαζαν μεταξύ τους. Παρτηρήσαμε δέ, και τα εξής ενδιαφέροντα.

Πρώτον, επειδή χρησιμοποιούσαμε μικρόφωνο χειρός, η ένταση του σήματος στο τελικό φασματόγραμμα είχε διακυμάνσεις, ανάλογα με την απόσταση που είχε κάθε φορά το μικρόφωνο από το στόμα του ομιλητή. Προσπαθήσαμε λοιπόν να δούμε με ποιό τρόπο θα μειώναμε τις διακυμάνσεις αυτές, είτε με υλικό, είτε με λογισμικό. Ένα στοιχείο που θα βοηθήσει, είναι η χρήση σταθερού μικροφώνου, από αυτά που φοριούνται στο κεφάλι και η μικροσκοπική κάψα βρίσκεται εμπρός στο στόμα του ομιλητή. Δυστυχώς όμως, δέν είχαμε τέτοιου είδος μικρόφωνο, οπότε έπρεπε να σκεφτούμε κάτι άλλο.

Η δεύτερη ιδέα που είχαμε, ήταν να χρησιμοποιήσουμε κάποιο κύκλωμα AGC στον προενισχυτή του μικροφώνου. Η λύση αυτή θα δούλευε, αλλά η λειτουργία του κυκλώματος AGC θα επηρέαζε το σήμα εισόδου, ενισχύοντας τον θόρυβο του περιβάλλοντος όταν δέν είχαμε ομιλία. Η ενίσχυση του θορύβου θα προκαλούσε εσφαλμένη ανίχνευση αρχής λέξης, οπότε θα μειωνόταν η ευαισθησία του αλγορίθμου.

Τελικά, η καλύτερη λύση είναι να γίνεται χρήση της full-CMOS έκδοσης του 56000 (που δέν είχαμε), διότι εκείνος προσφέρει ένα πολύ χρήσιμο χαρακτηριστικό για την επεξεργασία σήματος με FFT. Επιτρέπει την δυναμική κλιμάκωση των δεδομένων κατά την διάρκεια επεξεργασίας του FFT, οπότε τα αποτελέσματα διατηρούν όλο το δυναμκό εύρος του επεξεργαστή, με συνέπεια να μήν υπάρχουν τόσο σημαντικές διαφορές στην έξοδο του FFT. Να σημειώσουμε εδώ, οτι στην υλοποίηση του FFT σε επεξεργαστές fixed-point (σταθερής υποδιστολής), απαιτείται η κλιμάκωση των δεδομένων είτε πρίν από το FFT, είτε κατά την διάρκειά του, γιατί καθώς υπολογίζεται το κάθε ένα από τα περάσματα του μετασχηματισμού, τα δεδομένα μπορούν να μεγαλώσουν μέχρι 2 φορές. Η κλιμάκωση πρίν από το FFT μειώνει το πλάτος των δειγμάτων ώστε το μέγιστο εύρος τους να είναι (N-x) bits, όπου x ο αριθμός των περασμάτων του FFT (πχ. 8 για FFT 256 σημείων) και N ο αριθμός των διαθεσίμων bit της ALU. Στην συνέχεια, το FFT υπολογίζεται και τα δεδομένα δέν αυξάνουν πάνω από το εύρος της ALU, οπότε δέν έχουμε υπερχείλησή της.

Ο τρόπος αυτός, προσφέρει μικρότερο εύρος από οτι η δυναμική κλιμάκωση, που υλοποιεί ο 56001. Στην περίπτωση αυτή, τα δεδομένα περιορίζονται προς τα κάτω μόνο όταν έχουν τιμή που είναι δυνατόν να προκαλέσει υπερχείληση στο επόμενο

πέρασμα. Έτσι, κρατείται όσο το δυνατόν περισσότερη πληροφορία κατά την διάρκεια της επεξεργασίας, και επιπλέον, χρησιμοποιείται στο έπακρο όλο το δυναμικό εύρος της ALU.

Για να επιστρέψουμε στο θέμα μας, επειδή δέν είμαστε σε θέση να υλοποιήσουμε καμμιά από τις λύσεις που είχαμε σκεφτεί, περιοριστήκαμε στο να κρατάμε σε σταθερή απόσταση το μικρόφωνο από το στόμα του ομιλητή, οπότε είχαμε σχεδόν σταθερή απόκριση.

Προχωρώντας στην εκπαίδευση, είδαμε οτι ακόμη και κατά τις επαναλήψεις της ίδιας λέξης, μερικές από τις επαναλήψεις μπορεί να απέχουν πολύ από τις υπόλοιπες, επειδή ο ομιλητής μπορεί καμμιά φορά να προφέρει εντελώς διαφορετικά την λέξη. Αυτό μας έδωσε την ιδέα να τροποποιήσουμε λίγο τον αλγόριθμο εκμάθησης. Σκεπτόμενοι οτι στην βιβλιοθήκη του συστήματος πρέπει να βρίσκονται λέξεις που αντιπροσωπεύουν την μέση περίπτωση, και όχι κάποια εξαίρεση στον τρόπο ομιλίας, αποφασίσαμε οτι θα ήταν καλό να προσέχουμε κάθε είσοδο που δίνουμε στο σύστημα κατά την διάρκεια της εκμάθησης. Άν η τελευταία είσοδος είναι πολύ διαφορετική από τις προηγούμενες, τότε θα πρέπει να απορριφθεί και να μήν συμμετέχει στην δημιουργία της βιβλιοθήκης, μιας και θα αποτελεί την ακραία περίπτωση προφοράς της λέξης στην οποία εκπαιδεύεται το σύστημα.

Εφαρμόζοντας αυτή την παραλλαγή, είχαμε 100% επιτυχία αναγνώρισης σε λέξεις που ανήκουν στο σέτ εκμάθησης της βάσης δεδομένων. Επεκτείνοντας αυτή την σκέψη, το σύστημα μπορεί να διατηρήσει τέτοια υψηλά ποσοστά επιτυχίας, όταν κατά την διάρκεια της χρήσης του η βάση δεδομένων δέν παραμένει στατική, αλλά συνεχώς, με κάθε νέα επιτυχημένη αναγνώριση, βελτιώνεται και προσαρμόζεται στον τρόπο ομιλίας του χρήστη. Έτσι, ο τελικός αλγόριθμος πρέπει να είναι προσαρμοζόμενος στον χρήστη, (speaker adaptive). Αυτό είναι ένα από τα πιό απαραίτητα χαρακτηριστικά που θα πρέπει να έχει η εφαρμογή στο PC που θα διαχειρίζεται την βάση δεδομένων, ώστε να είναι πρακτική στην χρήση της.

Αναγνώριση

Μετά την εκπαίδευση του συστήματος, ήρθε επιτέλους η ώρα να δούμε με ποιό ακριβώς τρόπο θα γίνεται η αναγνώριση. Με τις πρώτες δοκιμές, φάνηκε οτι το σύστημα τα πηγαίνει τέλεια στις λέξεις που προέρχονται από το σέτ της βάσης δεδομένων του. Θέλοντας να δούμε τα όριά του, προσπαθήσαμε να το “μπερδέψουμε”, δίνοντάς του διάφορες λέξεις.

Παρατηρήσαμε οτι ο αλγόριθμος DTW που χρησιμοποιούσαμε, δηλαδή ο χωρισμός της λέξης σε 32 ζώνες, ναί μεν είναι ικανοποιητικός, αλλά δέν μπορεί να αποκριθεί στα τεχνητά εμπόδια που του βάλαμε, δηλαδή την προφορά μιας λέξης με την διάρκειά της να είναι “ξεχειλωμένη” από μερικές εκατοντάδες msec σε αρκετά δευτερόλεπτα. Ασφαλώς, αυτό δέν είναι απρόσμενο, μιας και από την αρχή ξέραμε οτι ο DTW αυτός δέν είναι πιά *τόοοσοοο* τέλειος... Από την άλλη, οι διαφορές που παρατηρήσαμε στο φάσμα αυτής της λέξης, συνειδητοποιήσαμε οτι προέρχονταν κυρίως από την αλλαγή των αναλογιών μεταξύ των φθόγγων και των εναλλαγών τους. Δηλαδή, σε μια λέξη με κανονική διάρκεια, οι εναλλαγές των φθόγγων έχουν μια διάρκεια, που άν και μικρή, είναι συγκρίσιμη με την διάρκεια των φθόγγων, με αποτέλεσμα να συμμετέχουν και αυτές στο φασματόγραμμα. Όμως, σε μια “ξεχειλωμένη” λέξη, οι φθόγγοι επιμηκύνονται απαράδεκτα, και οι εναλλαγές μικραίνουν τόσο πολύ, που χάνονται. Αυτό ακριβώς το σημείο εξηγεί και την διαφορά στα τελικά φασματογράμματα.

Μετά τα πρώτα αποτελέσματα, κοιτάξαμε να εντείνουμε τις διαφορές που αντιλαμβάνεται ο αλγόριθμος σύγκρισης, ώστε να βελτιώσουμε τα ποσοστά επιτυχίας του σε λέξεις που μοιάζουν μεταξύ τους, αλλά δέν είναι ίδιες. Προσέξαμε οτι η ενέργεια του σήματος στα φασματογράμματα ήταν συγκεντρωμένη, όπως άλλωστε αναμενόταν, στις χαμηλές συχνότητες του σήματος. Έτσι, τοποθετήσαμε "βάρη" στις διαφορές των πινάκων από τα πρότυπα της βοβλιοθήκης, τονίζοντας τις διαφορές στις χαμηλές συχνότητες. Ενώ λοιπόν η σχέση με την οποία ανιχνεύουμε τις διαφορές ήταν στην αρχή

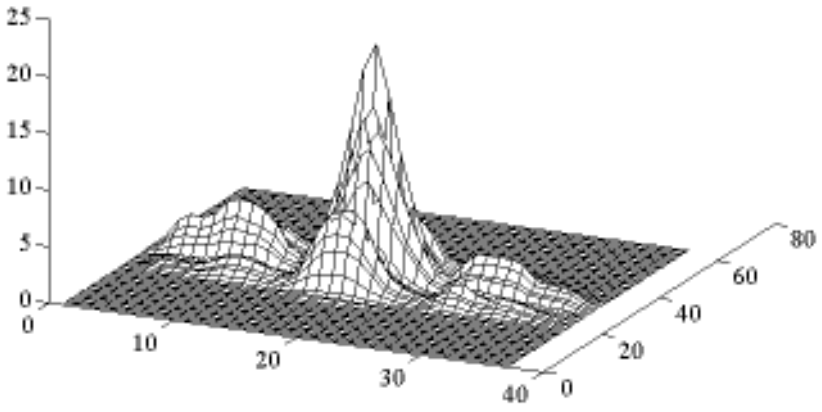
$$\Delta = \sqrt{\sum_{(i,j)=(1,1)}^{(19,32)} \left| x^2 - y^2 \right|}$$

τώρα αποκτά μια εξάρτηση από την συχνότητα, και γίνεται

$$\Delta = \sqrt{\frac{1}{f} \sum_{(i,j)=(1,1)}^{(19,32)} \left| x^2 - y^2 \right|}$$

Τώρα, οι εντείνονται οι διαφορές στις χαμηλές συχνότητες, κάνοντας τις ανόμοιες λέξεις να διαφέρουν ακόμη περισσότερο μεταξύ τους.

Ένα άλλο σημείο που σκεφτήκαμε, αλλά δέν είχαμε το χρόνο να το επεξεργαστούμε πλήρως, είναι η χρήση της συσχέτισης του φασματογράμματος εισόδου, με το αντίστοιχο της βιβλιοθήκης. Η συσχέτισή τους (correlation) θα μπορεί να μας πληροφορήσει άν έχουν υπάρξει μετατοπίσεις του φασματογράμματος εισόδου ως προς αυτό της βιβλιοθήκης. Άν υπάρχει αρκετή ομοιότητα (η κορυφή του correlation είναι αρκετά μεγάλη), τότε μπορούμε να δούμε σε ποιό σημείο η συνάρτηση της αυτοσυσχέτισης γίνεται μέγιστη, και να "ολισθήσουμε" την είσοδο σε εκείνη την θέση. Τότε, έχουμε τις περισσότερες πιθανότητες να ταιριάξουμε τα δύο σήματα μεταξύ τους. Δυστυχώς, δέν είχαμε τον χρόνο να ασχοληθούμε πρός αυτή την κατεύθυνση μέχρι να ετοιμαστούν αυτά τα κείμενα, οπότε τα αποτελέσματα δέν συμπεριλαμβάνονται εδώ.



Σχ. 5.3. *Το αποτέλεσμα της συσχέτισης δύο φασματογραμμάτων.*

Η τελευταία βελτίωση που κάναμε στον αλγόριθμο σύγκρισης, ήταν να προσθέσουμε και ένα μέτρο αβεβαιότητας στην απάντηση που δίνει, έτσι ώστε να μπορούμε να παρακολουθήσουμε άν οι αποφάσεις που παίρνει είναι σίγουρες ή όχι.

Στην αρχική του κατάσταση, ο αλγόριθμος σύγκρισης επέστρεφε την λέξη στην οποία το σήμα εισόδου ταίριαζε καλύτερα, έδινε δηλαδή το μικρότερο σφάλμα σε απόλυτη τιμή. Τί γίνεται όμως στην περίπτωση που τα πράγματα δέν είναι ξεκάθαρα, δηλαδή όταν το σήμα εισόδου ταιριάζει λίγο με όλες ή με μερικές λέξεις; Ασφαλώς, αυτό σημαίνει οτι δέν μπορούμε να απαντήσουμε με σιγουριά για το ποιά λέξη ήταν, μιας και μπορεί να είναι η λέξη *α* που ο ομιλητής πρόφερε λίγο σάν την *β,* οπότε ο αλγόριθμος τείνει να πεί οτι ήταν η *β,* που είναι λάθος.

Για να λύσουμε αυτά τα προβλήματα, θέσαμε ένα κατώφλι στο απόλυτο σφάλμα που ξεχωρίζει το άν έχουμε ικανοποιητικά αποτελέσματα ταύτισης ή όχι. Δηλαδή περιορίζει την απόκλιση της λέξης από το πρότυπό της σε κάποιον αριθμό (βλ. πίνακα 5.4). Άν σε κάποια προσπάθεια αναγνώρισης έχουμε ταύτιση με απόκλιση κάτω από το όριο για ένα πρότυπο, και απόκλιση πολύ πάνω από το όριο για τα υπόλοιπα πρότυπα, τότε έχουμε αναγνωρίσει με επιτυχία την λέξη. Το όριο αυτό είναι μεταβλητό, και υπολογίζεται δυναμικά, με βάση το μέσο όρο των αποκλίσεων της εισόδου από όλα τα πρότυπα της βιβλιοθήκης, και την απόλυτη τιμή της απόστασης των δύο κοντινότερων λέξεων.

Όταν έχουμε πολλά πρότυπα με αποκλίσεις που έχουν κοντινές τιμές, παρουσιάζεται δίλλημα, και επιστρέφεται η λέξη με την μικρότερη απόκλιση, αλλά παράλληλα και ένα μεγάλο ποσοστό αβεβαιότητας, που αυξάνεται όσο οι αποκλίσεις τείνουν να ξεπεράσουν το όριο. Όταν δέ δέν υπάρχει ταύτιση με απόκλιση κάτω από το αποδεκτό όριο, τότε η λέξη δέν αναγνωρίζεται, μιας και ο αλγόριθμος συμπεραίνει οτι δέν ταιριάζει με κανένα από τα πρότυπα της βιβλιοθήκης.

á/á	Åßóï äï ò	"Áåñï ðëÛí ï "	"EêôÝëåóç"	"Åí ôï ëÞ"	"ÕÝóåñá"
1	aero0	**36.5**	53.0	518	55.3
2	aero1	**28.6**	53.1	515	49.1
3	aero2	**24.3**	45.4	47.2	44.8
4	aero3	**22.3**	48.6	49.9	45.3
5	aero4	**29.2**	44.9	46.8	42.5
6	ektel0	53.0	**14.9**	24.2	32.9
7	ektel1	516	**18.1**	27.4	34.1
8	ektel2	48.6	**15.2**	25.9	27.2
9	ektel3	48.8	**21.1**	27.0	33.4
10	ektel4	46.8	**16.4**	27.5	30.9
11	entol0	52.0	29.7	**16.6**	37.5
12	entol1	47.7	26.0	**14.8**	34.2
13	entol2	49.8	23.1	**17.3**	32.6
14	entol3	54.5	28.2	**16.0**	36.1
15	entol4	52.3	27.2	**16.3**	35.3
16	tess0	45.2	28.5	30.0	**13.5**
17	tess1	49.2	30.7	35.9	**10.6**
18	tess2	49.7	310	37.1	**13.7**
19	tess3	48.9	30.8	36.4	**12.0**
20	tess4	49.8	37.2	34.0	**14.2**
21	tenfour	413	36.5	43.2	40.6

Πίνακας 5.4. *Τα αποτελέσματα αναγνώρισης των λέξεων της βιβλιοθήκης. Οι τιμές είναι οι αποστάσεις της λέξης εισόδου από το πρότυπο της βιβλιοθήκης.*

Ενδεικτικά αποτελέσματα

Όπως είπαμε, ο αλγόριθμος απέδωσε 100% επιτυχία στις λέξεις που βρίσκονται στην βιβλιοθήκη του. Αυτό μας ενθάρρυνε να συνεχίσουμε τους ελέγχου με άλλες λέξεις, ακόμη και εκτός βιβλιοθήκης. Για να δυσκολέψουμε ακόμη περισσότερο το σύστημα, δέν χρησιμοποιήσαμε αυτοπροσαρμοζόμενο αλγόριθμο για τα πρότυπα της βιβλιοθήκης. Άρα, ο αλγόριθμος έπαιρνε αποφάσεις μόνο με την βοήθεια των αρχικών γνώσεών του.

Κάναμε δύο διαφορετικά πειράματα. Στο πρώτο, δοκιμάσαμε νέα φασματογράμματα των λέξεων της βιβλιοθήκης, και προσπαθήσαμε να δούμε τί ποσοστό επιτυχίας υπάρχει σε αυτά. Έτσι, με βιβλιοθήκη 6 λέξεων, είχαμε 1 αποτυχία, 2 αναγνωρίσεις με κάποιο ποσοστό αβεβαιότητας, και 3 επιτυχίες. ‘Αν και το δείγμα δέν είναι αρκετά μεγάλο για να μιλήσουμε για ποσοστά, μπορούμε να πούμε οτι το ποσοστό επιτυχίας κυμαίνεται περίπου στο 70%

Στο δεύτερο πείραμα, δοκιμάσαμε την ικανότητα του συστήματος να απορρίπτει λέξεις που δέν ανήκουν στην βάση δεδομένων του. Στην περίπτωση αυτή, είχαμε 10 διαφορετικές εισόδους, με 4 ξένες προς την βιβλιοθήκη λέξεις. Απο τις 4 ξένες λέξεις, οι 3 απορρίφθηκαν σωστά, αλλά η τέταρτη αναγνωρίστηκε κατά λάθος, σαν μια από τις λέξεις της βάσης δεδομένων.

Δοκιμάζοντας και άλλες παρόμοιες τεχνικές, μπορούσαμε να βελτιώσουμε τα αποτελέσματα, βασιζόμενοι στην εφαρμογή καλύτερου αλγορίθμου σύγκρισης και την επιλογή καταλληλότερων συναρτήσεων βάρους.

Πάντως, τα αποτελέσματα είναι ικανοποιητικά, άν αναλογιστούμε οτι οι έλεγχοι έγιναν στις αντιξοότερες, για το σύστημα, συνθήκες. Περιμένουμε τεράστια βελτίωση των αποτελεσμάτων όταν εφαρμοστεί η τεχνική της αυτοπροσαρμοζόμενης βάσης δεδομένων, οπότε το σύστημα θα γίνει εξυπνότερο, και θα επιτυγχάνει ποσοστά επιτυχίας πάνω από το 90% για τον χρήστη του.

Παράρτημα

Στο παράρτημα αυτό υπάρχουν τα σχέδια, τα προγράμματα, τα αρχεία προγραμματισμού των λογικών κυκλωμάτων και τα αρχεία εξομοίωσής τους. Τα διάφορα έγγραφα έχουν κατανεμηθεί ανάλογα με το είδος τους, και όχι με το επιμέρους έργο στο οποίο ανήκουν. Έτσι, άν πχ σε κάποιο σχήμα εμφανίζεται κάποιο PLD, τα σχέδιά του θα είναι στο πρώτο μέρος, ενώ το πρόγραμμά του στο δεύτερο. Επίσης, τα σχέδια των πλακεττών δέν έχουν συμπεριληφθεί εδώ, μιας δέν χρησιμεύουν στον αναγνώστη. Υπάρχουν όμως, στην δισκέττα που συνοδεύει την εργασία, σε διάφορα format.

Στην δισκέττα θα βρείτε και τα πλήρη προγράμματα και αρχεία, που για λόγους οικονομίας χαρτιού δέν συμπεριλάβαμε εδώ (Βέβαια, 120 σελίδες προγραμμάτων που έχουμε βάλει δέν είναι και λίγες...) . Στα παραρτήματα περιέχονται μόνο μερικά απο τα προγράμματα που γράψαμε, τόσο σε C, όσο και σε assembly.

Σχέδια

1. Η κάρτα A/D και D/A για το PC.
2. Ο προενισχυτής μικροφώνου.
3. Ο μετατροπέας D/A για το ADS.
4. Τα κυκλώματα του Host Interface.
5. Η επέκταση μνήμης με τα περιφερειακά της.
6. Τα σχέδια με την λογική του FPGA της επέκτασης μνήμης
7. Η εξομοίωση για το FPGA της επέκτασης μνήμης

Αρχεία PLD

1. Αποκωδικοποίηση διευθύνσεων για την επέκταση D/A του ADS.
2. Αποκωδικωποίηση διευθύνσεων για την επέκταση μνήμης.

```
Title           Harris ICL7121 DAC unit for DSP 56001 ADM
Pattern         HDL-41-001/002-ILX
Revision        0.2
Author          Alexopoulos Ilias
Company         Firmware Co.
Date            21/05/95

;       HARDAC.PDS

CHIP  HARDAC  22V10

; pin assignments

;Inputs
PIN     [1:9]   A[12:4]
PIN     10      Y7
PIN     11      WR
PIN     13      XY
PIN     14      PS
PIN     15      DS
PIN     22      DELCLK

;Outputs
PIN     23      RCLK
PIN     21      DACWR

STRING  SEL     ' A12* A11* A10* A9* A8* A7*/A6* A5* A4'        ;addr.
FFBX

EQUATIONS

/rclk=SEL*/Y7*XY*PS*/DS*/WR
dacwr=delclk

SIMULATION

; set up vector and trace
; set to known state, preload registers (all low)

vector addr:=[a12,a11,a10,a9,a8,a7,a6,a5,a4]

setf    xy y7 ps ds wr
setf    addr:=0

for j:=300 to 512 do
begin
     setf ds
     setf wr
     setf y7
     setf addr:=j
     setf /ds
     setf /wr
     setf /y7
end
```

```
Title     DSP expansion memory decoder with boot support
Pattern   HDL-FRES0008-AI
Revision  0.8
Company   T.E.I. Peiraius
Author    Argiris Diamandis
Date      15/12/95

turbo on

chip    dspdec  pld22v10

;Inputs
pin  1  a15             ;address bit a15
pin  2  a14
pin  3  a13
pin  4  a12
pin  5  a11
pin  6  a10
pin  7  a9
pin  8  a8
pin  10 ps              ;Program memory select (active low)
pin  11 ds              ;Data memory select (active low)
pin  9  xy              ;X/Y memory select (1=X, 0=Y)

;Outputs
pin 21    CSph            ;CS for high program memory ($8000-$ffff)
pin 20    CSpl            ;       low
pin 19    CSxh            ;       high X data
pin 18    CSxl            ;       low X data
pin 17    CSyh            ;       high Y data
pin 16    CSyl            ;       low Y data

pin 15  EXCEPTION        ;Exception indicator (for debug)


equations

/csph=  (/ps * ds)     * (/(a15 * a14 * /a13*  a12 * a11  * a10  *  a9
*  a8   ;$df
                           +a15 * a14 * a13 * /a12 * /a11 * /a10 * /a9
* /a8    ;$e0
                           +a15 * a14 * a13 * /a12 * /a11 * /a10 * /a9
*  a8    ;$e1
                           +a15 * a14 * a13 * /a12 * /a11 * /a10 *  a9
* /a8    ;$e2
                           +a15 * a14 * a13 * /a12 * /a11 * /a10 *  a9
*  a8    ;$e3
                           )) * a15

/cspl=  /ps * ds * /a15
/csxh=  ps * /ds * xy * a15
/csxl=  ps * /ds * xy * /a15
/csyh=  ps * /ds * /xy * a15
/csyl=  ps * /ds * /xy * /a15
exception=      /ps * ds * /xy
```

```
simulation

vector addr:=[a15,a14,a13,a12,a11,a10,a9,a8]

setf /a15 /a14 /a13 /a12 /a11 /a10  /a9 /a8 ps ds xy
for i:=0 to 255 do
begin
        setf addr:=i
        setf /ps
        setf ps
        setf /ds
        setf /xy
        setf xy
        setf ds

end
```

Προγράμματα σε C.

1. Δειγματοληψίες σε πραγματικό χρόνο.
2. Καταγραφή και επεξεργασία δειγμάτων.
3. Φασματική ανάλυση με FFT σε πραγματικό χρόνο.
4. Υλοποίηση αρχικού αλγορίθμου αναγνωρισης.
5. Επικοινωνία του ADS με το PC μέσω του Host Interface.
6. Προγράμματα για την υποστήριξη της αναγνώρισης στο DSP.

adc6.c: Πρόγραμμα παλμογράφου για την κάρτα A/D - D/A του PC

```
#include <stdio.h>
#include <dos.h>
#include <graphics.h>
#include "grapha.h"

/*
     Change the SAMPLES to 640 and remove/add comments below to
     make for VGA
*/

#define BASE  0x300
#define SAMPLES 1024

int huge DetectVGA256()
     {
     return(4);
     };



void main()
{
unsigned int i,j;
unsigned int input, last;
char driver;
int gdriver=DETECT,gmode,color;
int scale, maxx, y;
clrscr();

/*installuserdriver("svga256",DetectVGA256);
initgraph(&gdriver,&gmode, "d:\\bc\\bgi");*/
InitGraph();

maxx=getmaxx();
setcolor(WHITE);
last=0;
y=(getmaxy()/2)+128;
disable();
while (!kbhit())
{
     for (j=0;j<maxx;j++)
     {
          outportb(BASE,_AX);
          input=y-inportb(BASE);


          line(j-1,last,j,input);
          last=input;
     }
 cleardevice();
 }
enable();
closegraph();
}
```

adc6.c: Πρόγραμμα παλμογράφου με μέγιστη δειγματοληψία

```
#include <stdio.h>
#include <dos.h>
#include <graphics.h>
#include "grapha.h"

/*
     The FASTEST program for real-time sampling
*/

void main()
{
register unsigned char Buffer[640];
register int i;

InitGraph();

while (!kbhit())
{
disable();         /*It means cli*/
for (i=0; i<640; i++)
{
     outportb(0x300,_AL);
     Buffer[i]=inportb(0x300);
}
enable();               /*That's sti */
clearviewport();
for (i=0; i<640; i++)
     line(i,340-Buffer[i],i+1, 340-Buffer[i+1]);
}
}
```

adcalib.c: Πρόγραμμα ρύθμισης του αναλογικού μέρους της κάρτας του PC.

```
#include <stdio.h>
#include <math.h>
#include "sample.h"
#include "timedom.h"

#define FRAME 1024

void main(void)
{
char Sample[FRAME];
int i;

     SetTimer(10);
     HookInterrupt();

     clrscr();
     printf("\n\n\t\t     ADC CALIBRATION PROGRAM\n\t Adjust for min
magnitude & energy with mike off\n");
     printf("\t\t\tFrame size:%d\n",FRAME);
     printf("\t\t     Press any key to exit\n");
     printf("Average Energy   :\n");
     printf("Average Magnitude:\n");
     printf("Zero crossings   :");
     while (!kbhit())
     {
          for (i=0; i<FRAME; i++) Sample[i]=GetSample();
          gotoxy(20,7);
          printf("%3.3f      ", Energy(FRAME,Sample,0));
          gotoxy(20,8);
          printf("%3.3f      ", Average(FRAME,Sample,0));
          gotoxy(20,9);
          printf("%3.3f      ", ZeroCrossings(FRAME,Sample,0));
     }
     RestoreTimer();
     UnHookInterrupt();
}
```

sample.h: Αρχείο δηλώσεων για την βιβλιοθήκη sample.c

```
/*********************************************************************
                    SAMPLE.H
                    v. 1.01
          Sampling functions for IBM PC machines only.

Usage:
     Before sampling:
          1. Call SetTimer()
          2. Call HookInterrupt()
          3. Use GetSample() to get samples from board

     When you are done:
          1. Call UnHookInterrupt()
          2. Call RestoreTimer()

*********************************************************************/

#ifndef   __SAMPLE_H__
#define __SAMPLE_H__

#include "defaults.inc"

#define INTR  0x1c /* Timer interrupt */
#define BASE  0x300    /* Base address of ADC board */
#define LPT1  0x378   /* Address of output port (LPT1) */


void SetTimer(float KiloHertz);
void RestoreTimer(void);
void HookInterrupt(void);
void UnHookInterrupt(void);
void interrupt NewTickHandler();
BASE_UNIT GetSample(void);
void PlaySample(BASE_UNIT Sam);

extern
char      OkToSample,
          OkToPlay,
          SampleReady;

extern
BASE_UNIT SampleToRead,
          SampleToPlay;
extern int    RepeatRate;
#endif
```

sample.c: Βιβλιοθήκη δειγματοληψίας για το PC

```
/*************************************************************************
                    SAMPLE.C
                    v. 1.01
          Sampling functions for IBM PC machines only.

Usage:
     Before sampling:
          1. Call SetTimer()
          2. Call HookInterrupt()
          3. Use GetSample() to get samples from board

     When you are done:
          1. Call UnHookInterrupt()
          2. Call RestoreTimer()

*************************************************************************/
#include <dos.h>
#include "sample.h"
#include "defaults.inc"

#ifdef DEBUG
#include <stdio.h>
#endif


char    OkToSample =0,
     OkToPlay  =0,
     SampleReady   =0;

BASE_UNIT SampleToPlay,
          SampleToRead;

int RepeatRate=1;

#ifdef __cplusplus
void interrupt (*OldHandler)(...);
#else
void interrupt (*OldHandler)();
#endif

void SetTimer(float KiloHertz) /* Prepair timer 0 for (KiloHertz)KHz
output */
{
unsigned char MSB, LSB;
int Divisor;

     Divisor = 1193.18 / KiloHertz;
     MSB = (unsigned char) (Divisor / 256);
     LSB = (unsigned char) (Divisor % 256);
     RepeatRate=(int) 55 / KiloHertz;   /* Call frequency of old
handler */

     #ifdef DEBUG
     printf("Timer Settings: \n");
     printf("\tRequested frequency:%f KHz\n",KiloHertz);
```

```
     printf("\tDivisor            :%d\n",Divisor);
     printf("\tOld Handler every  :%d ticks\n",RepeatRate);
     printf("\tMSB of counter     :%#4x (%u)\n",MSB,MSB);
     printf("\tLSB of counter     :%#4x (%u)\n",LSB,LSB);
     #endif

     disable();
     outportb(0x43,0x36);    /* counter 0, mode 3, load LSB,MSB */
     asm {nop;nop;nop}; /* Wait for the bus to settle */
     outportb(0x40,LSB);     /* LSB of counter */
     asm {nop;nop;nop};
     outportb(0x40,MSB);     /* MSB of counter */
     asm {nop;nop;nop};
     enable();


}

void RestoreTimer(void)         /* Restore timer 0 for 18.2 Hz output
*/
{
     outportb(0x43,0x36);    /* counter 0, mode 3, load LSB,MSB */
     outportb(0x40,0x00);    /* LSB of counter */
     outportb(0x40,0x00);    /* MSB of counter */
     #ifdef DEBUG
     printf("Timer reset to standard values\n");
     #endif
}


#ifdef __cplusplus
void interrupt NewTickHandler(...)
#else
void interrupt NewTickHandler()
#endif
{
static char i=0;

     disable();              /* Stop all other interrupts */

     if (OkToPlay) {                 /* Play a sample, if told to do
so */
          outportb(LPT1, SampleToPlay);
          OkToPlay=0;
          }
     if (OkToSample) {               /* Get a sample, if needed */
          outportb(BASE,_AL);
          _asm{
               nop             /* Wait for ADC to settle */
               nop
               nop             /* MAY REQUIRE ADJUSTMENT */
               nop
               nop
               nop
               nop
               nop
          }
          SampleToRead=inport(BASE)-127; /* Return sample & convert */
```

```
          OkToSample=0;                  /* it to a signed value    */
          SampleReady=1;
          }
     i++;
     if (i%RepeatRate) OldHandler();        /* Call old handler if
needed*/
     enable();          /* Restore interrupts */
}

void HookInterrupt(void)
{
     #ifdef DEBUG
     printf("Hooking timer interrupt (%#x)..",INTR);
     #endif
     OldHandler=getvect(INTR);
     setvect(INTR,NewTickHandler);
     #ifdef DEBUG
     printf("..done\n");
     #endif
}

void UnHookInterrupt(void)
{
     #ifdef DEBUG
     printf("Un-hooking interrupt (%#x)...",INTR);
     #endif
     setvect(INTR,OldHandler);
     #ifdef DEBUG
     printf("..done\n");
     #endif
}

BASE_UNIT GetSample(void)        /* Call this to get a sample */
{
     OkToSample=1;
     while(!SampleReady);
     SampleReady=0;
     return(SampleToRead);
}

void PlaySample(BASE_UNIT Sam){       /* Call this to play a sample */

     SampleToPlay=Sam;
     OkToPlay=1;
     while(OkToPlay==1);
     }
```

timedom.h: Αρχείο δηλώσεων βιβλιοθήκης timedom.c

```
/*********************************************************************

                    TIMEDOM.H
                    ver. 1.00
          INCLUDE file for basic time-domain functions

*********************************************************************/

#ifndef __TIMEDOM_H__        /* If this is not already included,*/


#define __TIMEDOM_H__            /* ..include it now */
#include "defaults.inc"

float Energy(int WindowSize, BASE_UNIT far *SampleArray, unsigned long
Index);
float Average(int WindowSize, BASE_UNIT far *SampleArray, unsigned
long Index);
float ZeroCrossings(int WindowSize, BASE_UNIT far *SampleArray,
unsigned long Index);


#endif
```

timedom.c: Βιβλιοθήκη επεξεργασίας λέξεων στο πεδίο του χρόνου

```
/*********************************************************************

                    TIMEDOM.C
                    ver. 1.00
          Implementation of basic time-domain functions
*********************************************************************/

#include <stdio.h>
#include <math.h>
#include "defaults.inc"
#include "timedom.h"

#undef DEBUG


float Energy(int WindowSize, BASE_UNIT far *SampleArray, unsigned long
Index)
{
     int i;
     long unsigned Sum=0;
     BASE_UNIT far *Sample_Index;


     Sample_Index=SampleArray+Index;
     for(i=0; i<WindowSize; i++) Sum += (*(Sample_Index+i) *
```

```
*(Sample_Index+i));

#ifdef DEBUG
     printf("Energy: %f, Index: %lu, Window size: %d \n",(float) Sum/
WindowSize, Index, WindowSize);
#endif

     return((float) Sum/WindowSize);
}

float Average(int WindowSize, BASE_UNIT far *SampleArray, unsigned
long Index)
{
     int i;
     double Sum=0;
     BASE_UNIT far *Sample_Index;

     Sample_Index=SampleArray+Index;
     for (i=0; i<WindowSize;i++) Sum += *(Sample_Index+i);

#ifdef DEBUG
     printf("Average: %f, Index: %lu, Window size: %d \n",(float) Sum/
WindowSize, Index,WindowSize);
#endif

     return((float) Sum/WindowSize);

}

float ZeroCrossings(int WindowSize, BASE_UNIT far *SampleArray,
unsigned long Index)
{
     int i;
     char SignofSample, SignofPrevSample;
     int Sum=0;
     BASE_UNIT far *Sample_Index;

     Sample_Index=SampleArray+Index;
     for(i=0;i<WindowSize-1;i++)
     {
          SignofPrevSample=(*(Sample_Index+i)>=0)?1:-1;
          SignofSample=(*(Sample_Index+i+1)>=0)?1:-1;
          Sum+=abs(SignofSample-SignofPrevSample);
     }


#ifdef DEBUG
     printf("ZeroCrossings: %f, Index:%lu, Window size: %d \n",(float)
Sum/(2*WindowSize), Index,WindowSize);
#endif

     return((float) Sum/(2*WindowSize));

}
```

speech.h: Αρχείο δηλώσεων βιβλιοθήκης speech.c

```
/*********************************************************************
                    SPEECH.C
                     v 1.11
              Misc. speech-processing functions

CatchWord(): Isolates a word spoken in the mike with lots of space
         before and after the true signal.
         Returns maximum possible position of end-of-signal.
*********************************************************************/

#include "defaults.inc"

#define LOOK_AHEAD_DURATION 250L   /* Duration of circular Buffer in
mS */
#define BUFFER_DURATION     3750L      /* Duration of normal buffer
in mS   */

#define SAMPLING_THRESHOLD      30
#define ZERO_THRESHOLD      6
#define SILENCE_DURATION    300       /* Duration of silence before
sampling stops */

#define  FRAME_SIZE         10        /* Frame size for DetectWord in
mS */


typedef struct{                          /* Holds indices for a
specific word */
unsigned long Start;
unsigned long End;
unsigned long Max;
} WordPoints;



WordPoints CatchWord(BASE_UNIT far *Circular, BASE_UNIT far *Normal);
WordPoints DetectWord(BASE_UNIT far *Buffer, WordPoints WordLimits);
```

speech.c: Βιβλιοθήκη ανίχνευσης αρχής-τέλους λέξης.

```
/*********************************************************************
                    SPEECH.C
                     v 1.11
              Misc. speech-processing functions

*********************************************************************/

#include <stdio.h>

#include "sample.h"
#include "timedom.h"
#include "speech.h"
#include "math.h"
```

```
/*#define DEBUG*/

#ifdef DEBUG
#include <stdio.h>
#include <conio.h>
#endif


WordPoints CatchWord(BASE_UNIT far *Circular, BASE_UNIT far *Normal)
{

unsigned cindex=0;         /* Indices for circular buffer */
unsigned long i=0, nindex=0;    /* Indices for normal buffer   */
BASE_UNIT Sample;
long CirSize;              /* Sizes of buffers, in elements */
long NorSize;
char Finished=0, LastWasZero=0;  /* Flags.. */
unsigned ZeroCount, ZeroLimit;
long Khz;
WordPoints WordEstimates;



     #ifdef DEBUG
     printf("\nInitializing variables at CatchWord()\n");
     #endif

     Khz     = (long) SAMPLE_RATE/1000;
     NorSize = (long) (BUFFER_DURATION * Khz);
     CirSize = (long) (LOOK_AHEAD_DURATION * Khz);
     ZeroLimit = (unsigned) (SILENCE_DURATION * Khz);

     #ifdef DEBUG
     printf("Cirsize=%lu, NorBufSiz=%lu, ZeroLimit=%u, Freq=%d Khz\n",
CirSize, NorSize,ZeroLimit,Khz);
     #endif


     #ifdef DEBUG
     printf("Waiting for threshold, using circular buffer..\n");
     #endif

     while ( abs(Sample=GetSample()) < SAMPLING_THRESHOLD )
     {
          *(Circular+cindex)=Sample;
          cindex=i++ % CirSize;
     }
     nindex=CirSize;   /* Start filling normal buffer further along,*/
                /* leaving space for the contents of         */
                /* the circular buffer (will be moved later) */


     #ifdef DEBUG
     printf("Moving in normal buffer\n");
     #endif

     ZeroCount=0L;
     while(!Finished)
```

```
     {
          *(Normal+nindex)=GetSample();

          if (abs(*(Normal+nindex)) < ZERO_THRESHOLD)  /* Zero handler
*/
          {       if (LastWasZero) ZeroCount ++;       /* of "zero-
level" samples */
               else                                    /* encountered so
far */
               {
                    LastWasZero=1;
                    ZeroCount=1;
                    WordEstimates.End=nindex;    /* Endpoint lies
beyond this number */
               }
          }
          else LastWasZero=0;

          nindex++;

          if ((ZeroCount > ZeroLimit) || (nindex > NorSize-1))
Finished=1;

               /* Stop Sampling when there is too much silence */
               /* or the buffer is full */
     }

     WordEstimates.Max = nindex;

     #ifdef DEBUG
     printf("Finished sampling, copying early data from circular
buffer\n");
     printf("Also approximating startpoint\n");
     #endif

     for (i=0; i<CirSize; i++)
     {
          *(Normal+i)=*(Circular+((cindex+i)%CirSize));      /* Copy
the start of circular buffer in the normal one */
          if ( abs(*(Normal+i)) < ZERO_THRESHOLD )
WordEstimates.Start=i;
     }


     #ifdef DEBUG
     printf("Exiting CatchWord()\n");
     #endif

     return(WordEstimates); /* Rough estimates of word limits     */
                       /* The true limits are always outside */
}                               /* this area                   */




WordPoints DetectWord(BASE_UNIT far *Buffer, WordPoints WordLimits)
{
```

```
int StatArea, Frame, dips;
float NoiseEnergy;
WordPoints NewLimits;
unsigned long i;
char finished;


     #ifdef DEBUG
     printf("Entering DetectWord()\n");
     printf("\tWord start is before %lu\n\tWord end is after
%lu\n\tSample ends at %lu\n",\
          WordLimits.Start, WordLimits.End, WordLimits.Max);
     #endif


     StatArea = (int) 100 * SAMPLE_RATE /1000;
     Frame     = (int) FRAME_SIZE * SAMPLE_RATE /1000; /* Frame size
for calculations */

     NoiseEnergy=Energy(StatArea, Buffer, (unsigned long)
(WordLimits.Max) - StatArea -1);

/*   temp1=ZeroCrossings(StatArea,Buffer,0);
     temp2=ZeroCrossings(StatArea, Buffer, (unsigned long)
(WordLimits.Max) - StatArea -1);
     AverageZC =(temp1+temp2)/2;*/

     #ifdef DEBUG
     /*printf("Average zero crossings of noise:%f\n",AverageZC);*/
     printf("Average energy of noise        :%f\n",NoiseEnergy);
     printf("Press any key\n");
     getch();
     #endif

     #ifdef DEBUG
     printf("Searching for startpoint...\n");
     #endif

     i=NewLimits.Start=WordLimits.Start;
     finished=0;
     while (!finished)
     {
          i-=Frame/2;
          if (i<Frame) finished=1;
          if ( Energy(Frame,Buffer,i) > NoiseEnergy )
          {
               NewLimits.Start=i;
               dips=0;
          }
          else
          {
               dips++;
               if (dips>2) finished=1;
          }

     }

     #ifdef DEBUG
```

```
     printf("Searching for endpoint...\n");
     #endif

     i=NewLimits.End=WordLimits.End;
     finished=0;
     while (!finished)
     {
          if (i>WordLimits.Max-Frame) finished=1;
          if ( Energy(Frame,Buffer,i) > NoiseEnergy )
          {
               NewLimits.End=i+Frame/2;
               dips=0;
          }
          else
          {
               dips++;
               if (dips>2) finished=1;
          }
          i+=Frame/2;
     }

     NewLimits.Max = WordLimits.Max;

     #ifdef DEBUG
     printf("Exiting DetectWord()\n");
     #endif
     return(NewLimits);
}
```

specfunc.h: Αρχείο δηλώσεων βιβλιοθήκης specfunc.c

```
/*****************************************************************
        Include file: specfunc.c

        Version 1.0
        Date: 03/01/94
        Source file: specfunc.c
*****************************************************************/

/*****************************************************************
     Prototypes for functions defined in averfreq.c
     include: averfreq.h


                    Frequency Elements Average

                       Version:1.0
                       date:02/01/94
                       Name: AverFreq.c

     Invector: float vector with data from the FFT to be averaged.
     Outvector: The averaged spectrum (spectogram) (float)
     lenFFT: the FFT size -> Invector size
     k: (silly name!) In which time band we are. (1 of 16)

     Error codes:
     OK: Everything juuuusstt ffiiinneee...
     DIVBYZERO: Division By Zero! (normally this should not happen)

                       Written by Ilias Alexopoulos

*****************************************************************/


/*****************************************************************
                Include: ipolat.h
                Time Domain Digital Interpolation

                Version: 0.1
                date:31/12/93
                Name: ipolat.c

                outvector: vector to add zeros at the time domain
                win_size: size of samples ( word/16)
                lenFFT: Size of the FFT window

                return code:
                OK: fine

                WARNING:
                outvector elements from win_size to lenFFT are zeroed.

                Written by Ilias Alexopoulos

*****************************************************************/
```

```
/*********************************************************************
        include: window.h

        window function to cut the signal in the time domain

        Version: 0.1
        date: 30/12/93
        Name: window.c

        invector: input vector with samples. (initial samples)
        outvector: output vector with length of lenFFT
        win_size: is the result of sample_length/16 (window size)
        index: which piece of 16th of the word are we now
               (index=i*win_size)

        returns error code.

        OK:                     everything fine.
        INVALID_WIN_SIZE        win_size=0

        The invector remains intact after the function call
        The window is currently: hanning

        Written by Ilias Alexopoulos

*********************************************************************/

/*********************************************************************
     Include: spectogram.h
     Spectorgam Function

     Version: 0.1
     Date: 31/12/93
     Name: spectogram.c

     Function name: scaledspectogram(..) -> scaled spectogram
                    spectogram(..)       -> unscaled spectogram

     Invector: signal sample set input [MAX_SAMPLE]
     Spectrum: Vector output of the signal [19*16]
           int for scaled, float for unscaled
     Word_len: Length of the word to analyse


     Error codes:
     OK:                    everything ok
     WORDTOOLARGE           Word larger than 32K
     WINDOWERROR            Window returned error
     INTERPOLATIONERROR     Interpolation error
     FFTERROR               FFT error
     SCALEERROR             Scaling error
     AVERFRERROR            Average frequency error


     Written by Ilias Alexopoulos

*********************************************************************/
```

```
#include "scale.h"

int scaledspectogram(BASE_UNIT far *invector,unsigned int far
*spectrum, unsigned long word_len);
int spectogram(BASE_UNIT far *invector,float far *spectrum, unsigned
long word_len);

int averfreq(float *invector,float *outvector,int lenFFT,int k);
int interpolation(float *outvector,int win_size,int lenFFT);
int Window(BASE_UNIT far *invector, float *outvector,int win_size,int
index);
```

specfunc.c Βιβλιοθήκη επεξεργασίας σήματος στο πεδίο της συχνότητας

```
/*********************************************************************
        Spectorgam Functions

                 Version: 1.1
                Date: 28/02/94
                Name: specfunc.c

     Functions:
                   scaledspectogram(..)    -> scaled spectogram
                   spectogram(..)     -> non scaled spectogram


                   Written By Ilias Alexopoulos

*********************************************************************/

#include <stdio.h>
#include <math.h>
#include "defaults.inc"
#include "errors.inc"
#include "bands.inc"
#include "..\final\fft.h"
#include "scale.h"
#include "specfunc.h"



int scaledspectogram(BASE_UNIT far *invector,unsigned int *spectrum,
unsigned long word_len)
{
int exponent,lenFFT,win_size,error,i;

         /* vectors for internal use */
float buffer[FREQZONES*TIMEZONES];
float outvector[MAXARRAY],powerspec[MAXARRAY];

     /* if word length >16K 2^exponent=2048
        .............. >8K  ......... =1024
        .............. >4K  ......... =512   */

       /*if(word_len<4096 && word_len>2048) exponent=8;
```

```
     if(word_len<8192 && word_len>4096) exponent=9; */
     if(word_len<16384 && word_len>2048) exponent=10;
     if(word_len<32768 && word_len>16384) exponent=11;
     if(word_len>32768 || word_len<2048) return(WORDOUTRANGE);

     lenFFT=1<<exponent;
     win_size=(int)(word_len/16);


     for(i=0;i<16;i++)
     {
          error=Window(invector,outvector,win_size,i*win_size);
          if(error!=OK)
          {
               printf("Error:%d", error);
               return(WINDOWERROR);
          }
          error=interpolation(outvector,win_size,lenFFT);
          if(error!=OK)
          {
               printf("Error:%d", error);
               return(INTERPOLATIONERROR);
          }
          error=fft(outvector,powerspec,exponent);
          if(error!=OK)
          {
               printf("Error:%d", error);
               return(FFTERROR);
          }
          error=averfreq(powerspec,buffer,lenFFT,i);
          if(error!=OK)
          {
               printf("Error:%d\n", error);
               return(AVERFRERROR);
          }

#ifdef DEBUG
          printf("timepart:%d \n",i);
#endif

     }

     error=ScaledSignal(TOTALELEMENTS,buffer,(int *) spectrum);
     if(error!=OK) return(SCALEERROR);

     return(OK);
}

int spectogram(BASE_UNIT far *invector,float *spectrum, unsigned long
word_len)
{
     int exponent,lenFFT,win_size,error,i;

          /* vectors for internal use */
     float outvector[MAXARRAY],powerspec[MAXARRAY];


     /* if word length >16K 2^exponent=2048
```

```
        .............. >8K  ......... =1024
        .............. >4K  ......... =512   */

     if(word_len<4096 && word_len>2048) exponent=8;
     if(word_len<8192 && word_len>4096) exponent=9;
     if(word_len<16384 && word_len>8192) exponent=10;
     if(word_len<32768 && word_len>16384) exponent=11;
     if(word_len>32768 || word_len<2048) return(WORDOUTRANGE);

     lenFFT=1<<exponent;
     win_size=(int)(word_len/16);


     for(i=0;i<16;i++)
     {
          error=Window(invector,outvector,win_size,i*win_size);
          if(error!=0) return(WINDOWERROR);
          error=interpolation(outvector,win_size,lenFFT);
          if(error!=0) return(INTERPOLATIONERROR);
          error=fft(outvector,powerspec,exponent);
          if(error!=0) return(FFTERROR);
          error=averfreq(powerspec,spectrum,lenFFT,i);
          if(error!=0) return(AVERFRERROR);
#ifdef DEBUG
          printf("timepart:%d \n",i);
#endif

     }

     return(OK);
}

/*********************************************************************

          Spectrum Functions for spectogram
          Version: 1.0
          Date: 02/01/94
          Name: specfunc.c

          Functions:
          Window.c        Window(..)
          ipolat.c        interpolation(...)
          averfreq.c      averfreq(...)
          scale.c         ScaledSignal(...)

*********************************************************************/


/*********************************************************************
     Window function to cut the signal in the time domain

     Version: 1.0
     date: 30/12/93
     Name: Window.c

     Written by Ilias Alexopoulos

*********************************************************************/
```

```
float win(int i,int ws) /* Window function: hanning window */
{
     float w;
     int k;
     k=ws/2;
     w=0.5*(1.0+cos(M_PI*(i-k)/ws));
     return(w);
}


int Window(BASE_UNIT far *invector, float *outvector,int win_size,int
index)
{
     int i;
     if(win_size==0) return(INVALID_WIN_SIZE);
     for(i=0;i<win_size;i++)
     *(outvector+i) = *(invector+index+i) * win(i,win_size);
     return(OK);
}



/*********************************************************************
          Time Domain Digital Interpolation

          Version: 1.0
          date:31/12/93
          Name: ipolat.c

          Written by Ilias Alexopoulos
*********************************************************************/
int interpolation(float *outvector,int win_size,int lenFFT)
{
     int i;

     /* zero elements from win_size to lenFFT */

     for(i=win_size;i<lenFFT;i++) *(outvector+i)=0;
     return(OK);
}

/*********************************************************************
                Frequency Elements Average

                Version:1.2
                date:02/01/94
                Name: AverFreq.c

                Written by Ilias Alexopoulos

*********************************************************************/

int averfreq(float *invector,float *outvector,int lenFFT,int k)
{
     int i,j;
```

```
     float index,max;
     float sum;
     extern float freqbands[],freqpoints[];


     for(i=0;i<19;i++)
     {
          /* find the ith analog freq in which freq. sample belongs */

          index= ((freqpoints[i] * lenFFT) / SAMPLE_RATE); /* freq to
begin averaging */
          max= ((freqbands[i] * lenFFT) / SAMPLE_RATE); /* freq range
to average */


          if(max==0) return(DIVBYZERO);

              /* average */

          for(j=0,sum=0;j<(int)(max);j++) sum+=
*(invector+((int)(index)+j));
/*#ifdef DEBUG
          printf("timepart:%d fstart:%d fend:%d sum:%f \n",k,(int)
index,((int)(index)+j),sum);
#endif*/

          *(outvector+k*19+i)=sum/(max);

     }
     return(OK);

}
```

fft.h: Αρχείο δηλώσεων για το fft.c

```
/*********************************************************************
     Prototypes for functions defined in fft.c
     file: fft.h
*********************************************************************/

/*********************************************************************

     Fast Fourier Transform Function
     found in "Interfacing" page 322.
     Modified by Ilias Alexopoulos

     date:   26/12/93
     version: 1.0
     Name of file:fft.c

     Parameters:

     INVECTOR: Input data to FFT. (type: float) only Real Samples

     OUTVECTOR: Output of Power Spectrum (Float)

     NU: Number of samples=2^NU (length of Invector)

     length of Invector = length of outvector = N < FFT_WINDOW_SIZE

     Invector and Outvector are pointers to preallocated arrays
     from the calling routine.

     Inputs remain intact after the call of this function

     Return codes:
     0 : Everythink fine you can drink your milk.
     1 : ooops! 2^NU is greater than FFT_WINDOW_SIZE

     FFT_WINDOW_SIZE defined in "default.inc"

*********************************************************************/
int fft(float *invector,float *outvector,int nu);
int ibitr(int j,int nu);
```

fft.c: Ταχύς Μετασχηματισμός Fourrier

```
/*********************************************************************
     Fast Fourier Transform Function
     found in "Interfacing" page 322.
     Modified by Ilias Alexopoulos

     date:   26/12/93
     version: 1.01
     Name: fft.c

     Parameters:

     INVECTOR: Input data to FFT. (type: float) only Real Samples
```

```
     OUTVECTOR: Output of Power Spectrum (Float)

     NU: Number of samples=2^NU (length of Invector)

     length of Invector = length of outvector = N < FFT_WINDOW_SIZE

     Invector and Outvector are pointers to preallocated arrays
     from the calling routine.

     Inputs remain intact after the call of this function

     Return codes:
     OK             : Everything fine, you can drink your milk.
     POWTOOLARGE    : ooops! 2^NU is greater than FFT_WINDOW_SIZE

     FFT_WINDOW_SIZE defined in "default.inc"

*********************************************************************/


#include "defaults.inc"
#include "errors.inc"
#include "..\final\fft.h"

#include <math.h>


#define TWOPI 6.283185307     /* 2*PI and PI/2 */
#define PITWO 1.570796327


int fft(float far *invector,float far *outvector,int nu)
{
     float stab[FFT_WINDOW_SIZE],ctab[FFT_WINDOW_SIZE];      /* define
local variables */
     int n,n1,n2,nu1,p,k1n2,i,k,l;                           /* loop
counters etc. */
     float arg,c,s,tr,ti;                                    /*
temporary vars */
     float xr[FFT_WINDOW_SIZE],xi[FFT_WINDOW_SIZE];          /*
interal arrays */

   n=1<<nu;
   if(n>FFT_WINDOW_SIZE)      /* check if n>FFT_WINDOW_SIZE if yes
return */
      return(POWTOOLARGE);


     for(i=0;i<n;i++)
     {
          xr[i]=*(invector+i);    /* copy vector to internal array */
          xi[i]=0;                /* sorry no imaginary parts... */
     }


     n2=n/2;
     nu1=nu-1;
```

```
     for(i=0;i<n;i++)
     {
          arg=TWOPI*i/n;
          stab[i]=sin(arg);
          ctab[i]=sin(arg+PITWO);
     }

     k=0;
     for(l=0;l<nu;l++)
     {
          while(k<n)
          {
               for(i=0;i<n2;++i)
               {
                    n1=1<<nu1;
                    p=ibitr(k/n1,nu);
                    s=stab[p];
                    c=ctab[p];
                    k1n2=k+n2;
                    tr=xr[k1n2]*c+xi[k1n2]*s;
                    ti=xi[k1n2]*c-xr[k1n2]*s;
                    xr[k1n2]=xr[k]-tr;
                    xi[k1n2]=xi[k]-ti;
                    xr[k]+=tr;
                    xi[k]+=ti;
                    k++;
               }

                         /* end i loop */

          k+=n2;
          }

          k=0;
          —nu1;
          n2=n2/2;

      } /* end l loop */

      for(k=0;k<n;++k)
      {

          i=ibitr(k,nu);
          if(i>k)
          {

               tr=xr[k];
               ti=xi[k];
               xr[k]=xr[i];
               xi[k]=xi[i];
               xr[i]=tr;
               xi[i]=ti;
          }
      }


      for(i=0;i<n;i++) /* calculate Power Spectrum and store to output
vec */
```

```
        *(outvector+i)=sqrt(xr[i]*xr[i]+xi[i]*xi[i]);

      return(OK);
}

int ibitr(int j,int nu)
{
     int rm=1,lm,i,bitr=0;
     lm=1<<(nu-1);

     for(i=0;i<nu;++i)
     {
          if((j&rm)!=0) bitr=bitr|lm;
          rm=rm<<1;
          lm=lm>>1;
     }
     return(bitr);
}
```

fileman.h: Αρχείο δηλώσεων βιβλιοθήκης fileman.c.

```
/*************************************************************************

          Misc. File Management Functions

          INCLUDE file for FileMan.c

          Name: FileMan.h
          Date:25/02/94
          Version:1.6

          Written by Ilias Alexopoulos

*************************************************************************/

/*************************************************************************

IFF FORMAT FILE:
     _______
     | FORM SIZE |
     _______ Be Careful: REPA Hunk must be The First
     | REPA SIZE | Chunk with the current routines (IFF VERSION 1).
     _______
     | REVISION  |
     _______
     | REPS SIZE | REPS Chunk must ALWAYS be the LAST chunk because
     _______ Of the modifications which are easy to do
     |  PATTERN  | (like append). This WILL NOT likely change in the
     |  .......  | future.
     |  .......  |
     |  PATTERN  |
     _______

*************************************************************************/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include ".\filestruct.h"
#include "..\final\defaults.inc"
#include "..\final\errors.inc"

#define ULONG unsigned long

#define  MakeID(a,b,c,d)    \
     ((ULONG) (a)<<24 | (ULONG) (b)<<16 | (ULONG) (c)<<8 | (ULONG)
(d))


#define       CURRENTVERSION1

#ifdef AMIGA
#define       ID_REPA       MakeID('R','E','P','A')
#define       ID_FORM       MakeID('F','O','R','M')
#define       ID_REPS       MakeID('R','E','P','S')
```

```
#endif

#ifdef PC
#define       ID_REPA       MakeID('A','P','E','R')
#define       ID_FORM       MakeID('M','R','O','F')
#define       ID_REPS       MakeID('S','P','E','R')
#endif


#define       ADD      1
#define       SUB      -1

#define       SEEKERROR 0
#define       MAXNAMELEN    40
#define       DELETEBUFSIZE sizeof(struct iff)+sizeof(struct
revision)+8

char oldname[MAXDATBASEACT][MAXNAMELEN];
FILE *rec_fp[MAXDATBASEACT];
unsigned int totalwords[MAXDATBASEACT];
unsigned char flag=0;

struct refhead pathead={
              ID_REPS,
              0x0000
              };

struct revision revs={
              ID_REPA,
              sizeof(struct revision)-8,
              SPUNIT,
              FMTVER,
              sizeof(struct pattern),
              0x0000,
              NULL,
              NULL,
              AUTHOR
              };


unsigned long basepos=sizeof(struct revision)+8+8; /* Pattern base
position */

struct iff iffhead={
              ID_FORM,
              (sizeof(struct revision)-8)
              };


int InitLFile(char *name,unsigned short findex);
int CreateIFF(char *name);
int CheckIFF(unsigned short findex);
int GetInfo(struct revision *revl,unsigned short findex);
int PutInfo(struct revision *revl,unsigned short findex);
int WriteRecord(unsigned short int index,struct pattern *pat,unsigned
short findex);
int ReadRecord(unsigned short int index,struct pattern *pat,unsigned
short findex);
```

```
int CalcNewHead(int operator,unsigned short findex);
int DeleteRecord(unsigned short index,unsigned short findex);
int RemakeFile(unsigned short findex);
int ExitFile(unsigned short findex);
int WriteControlFile(char *name,unsigned short *array);
int ReadControlFile(char *name,unsigned short *array);
unsigned char GetPriority(unsigned short indx,unsigned short findex);
```

filestruct.h: Συνοδευτικό αρχείο του fileman.c.

```
/*********************************************************************
          Misc. File Management Functions
          INCLUDE file for using the Fileman.c functions

          Name: FileStruct.h
          Date:25/02/94
          Version:1.6

          Written by Ilias Alexopoulos (Imaginary programmer)
*********************************************************************/
#include "..\final\defaults.inc"

struct iff {  /* IFF header */
              unsigned long chunkid;  /* ChunkID: FORM*/
              unsigned long formlen;  /* Length of the IFF Data   */
};

struct refhead {   /* Pattern's Chunk Header */

              unsigned long chunkid;  /* ChunkID:REfer.Pat. Struc.*/
              unsigned long repslen;  /* Length of the pattern Data
*/
};

struct revision {  /* we include the IFF chunk header */

              unsigned long chunkid;  /* ChunkID: REvision PArams */
              unsigned long revlen;   /* Chunk Length         */
              unsigned char specunit; /* Spectogram Unit      */
              unsigned char version;  /* Version of Recog. file   */
              unsigned short int patlen; /* Length of pattern struct*/
              unsigned short int deleted; /* Deleted Records?*/
              unsigned long reserved;     /* Future use?*/
              unsigned short int user;    /* User Data*/
              char author[MAX_ANLEN];     /* Author's name of file*/
};

struct pattern { /* The Pattern structure */

              char word[MAX_WLEN];    /* Word name of pattern */

              unsigned int mat[FREQZONES*TIMEZONES];  /* Spectogram
matrix */

unsigned char freq;              /* statistical word freq.    */
unsigned char pri;          /* statistical priority     */
};
```

bands.inc: Αρχείο περιγραφής συχνοτικών περιοχών. Χρησιμοποιείται απο το specfunc.c

```
/***********************************************
        bands for the AverFreq function
***********************************************/

/* frequency start bands */

float
freqpoints[19]={180,300,420,540,660,765,925,1075,1225,1375,1525,1700,1900,2100,2300,2600,2850,3150,3500};


/* frequency bandwidth */

float
freqbands[19]={120,120,120,120,120,150,150,150,150,150,150,200,200,200,200,200,300,300,500};
```

defaults.inc: Αρχείο ρυθμίσεων προγραμμάτων αναγνώρισης

```
/*********************************************************************

                  DEFAULTS.INC
                  ver    0.110
         Default states file. Include it with every source
         file to automatically configure your functions

*********************************************************************/
#ifndef __DEFAULTS_INC__
#define __DEFAULTS_INC__

/*****************************************************
         SAMPLING SPECIFIC
            (Argiris)
*****************************************************/
#define BASE_UNIT  char
#define  SAMPLE_RATE   10000L
#define BASE_SCALED    32766   /*Base for normalising the Signal
(32766 for int, 127 for byte)*/
#define NumofLines 19   /*Number of frequency bands */
#define NumofColums    16   /*Number of time zones */
#define Pi              3.141592654
#define Pi2        6.283185307


/*****************************************************
         DEFINE PLATFORM
              (All)
*****************************************************/
#define PC                  /* Set PC for a PC clone */


/*****************************************************
         DEBUG INFORMATION ON/OFF
              (All)
```

```
*******************************************************/
#define   DEBUG         /*comment this to get rid of debugging
output*/


/*******************************************************
         SIGNAL PROCCESSING
              (All)
*******************************************************/
#define FFT_WINDOW_SIZE     2048
#define FREQZONES       NumofLines         /* Redifinition */
#define TIMEZONES       NumofColums
#define TOTALELEMENTS       FREQZONES*TIMEZONES    /* Total Data */
#define MAXARRAY        2048          /* Internal Vectors in
Spectogram */


/*******************************************************
         TRAINING OPTIONS
*******************************************************/
#define NUM_REPEAT 3    /* Number of repeatition for mean term */


/*******************************************************
         FILE MANAGMENT
            (Ilias)
*******************************************************/
#define MAX_STRING      10            /* Maximum word length string */
#define MAX_WORDS       255           /* Maximum words in file pattern
database */
#define BUF_LEN             255           /* DOS Buffer length for
read */
#define WORD_STOP       ""            /* Null String */
#define SPUNIT          1             /* Spectrum unit: INT   */
#define FMTVER          1             /* Version of IFF Format */

#define AUTHOR          "012345678901234567891 23"   /* Default Author
String */
#define   MAX_ANLEN         24
#define   MAX_WLEN          10
#define   TEMPFNAME     "TEMP0.RFL"
#define MAXDATBASEACT   5             /* maximum data base files
active */


#endif
```

host.c: Πρόγραμμα επικοινωνίας του ADS με το PC μέσω του host interface

```
/* Host Interface DownLoad for the ADM board   */
/* DSP56001 mode on reset: 1          */
/* Download code to internal dsp memory for    */
/* Self Test and Diagnostics          */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
/* HEN pulse */
#define SETSTBcdata | 0x01
#define RSTSTBcdata & 0xfe
/* HA0 address */
#define SETA0   cdata & 0xfd
#define RSTA0   cdata | 0x02
/* HA1 address */
#define SETA1 cdata | 0x04
#define RSTA1 cdata & 0xfb
/* HA2 address            */
#define  SETA2    cdata & 0xf7
#define RSTA2 cdata | 0x08
#define CADDR 0x37A
#define DADDR 0x378
char cdata;
char ddata;
void wrhigh(char data);
void wrmid(char data);
void wrlow(char data);
void inithost(void);
void wrhost(void);
void readhost(void);
void endhost(void);
/*******************************************************
      HWMAP:
          SEL_IN : A1   (16)
          STROBE : WR     (01)
          INIT   : A0     (14)
          DATA   : D0-D7(2-9)
          SEL  :A2  ()
      IOMAP:
          SEL_IN: x7A/BIT 3
          STROBE: x7A/BIT 0
          INIT  : x7A/BIT 2
          DATA  : x78/BIT 0-7
          x: 3 For LPT1
          x: 2 For LPT2
      *******************************************************/
int main()
{
extern int _argc;
extern char **_argv;
     char filename[20];
     FILE *infile;
     unsigned int j,k,i=0;
     unsigned char high,mid,low;
```

```
     if (_argc<=1){
          printf("\nFile to send: ");
          scanf("%s",filename);
     }
     else strcpy(filename, _argv[1]);
     infile=fopen(filename,"rb");
     if (infile==NULL)
     {
          printf("No file to send\n");
          exit(1);
     }
     clrscr();
     printf("\n                    FIRMWARE\n");
     printf("              Ilialex Research Lab  IRL\n");
     printf("DSP56001 self test and diagnostics code downloader\n");
     printf("Don't forget to set DSP in mode 1 !\n");
     printf("Sending %s...\n",filename);
     disable();
     printf("Transfering Data\n");
     cdata=0;
/*   wrhf(0);   */
     inithost();
/*   readhost();*/
     i=0;
     low=0;
     mid=1;
     high=2;
     while (!feof(infile))
     {
          gotoxy(1,wherey());
          printf("Words sent: %u    ",i++);
          low=fgetc(infile);
          mid=fgetc(infile);
          high=fgetc(infile);
          wrhigh(high);
          wrmid(mid);
          wrlow(low);
     }
     endhost();
     enable();
     printf("Transfer Finished!\n");
     fclose(infile);
     return(0);
}
/* Write a data byte to dsp's RAM */
void wrhigh(char data)
{
     cdata=SETA2;
     cdata=SETA0;
     cdata=RSTA1;
     outport(CADDR,cdata);       /* HADDR: 5   */
     outport(DADDR,data);
     wrhost();
}
void wrmid(char data)
{
     cdata=SETA2;
     cdata=RSTA0;
```

```
     cdata=SETA1;                    /* HADDR: 6    */
     outport(CADDR,cdata);
     outport(DADDR,data);
     wrhost();
}
void wrlow(char data)
{
     cdata=SETA2;
     cdata=SETA0;
     cdata=SETA1;                    /* HADDR: 7    */
     outport(CADDR,cdata);
     outport(DADDR,data);
     wrhost();
}
void inithost(void)
{
     cdata=0;
     cdata=RSTA0;
     cdata=RSTA1;
     cdata=RSTA2;                    /* HADDR: 0    */
     cdata=RSTSTB;
     outport(CADDR,cdata);
     outport(DADDR,0x82);
     wrhost();
}
void endhost(void)
{
     cdata=RSTA0;
     cdata=RSTA1;
     cdata=RSTA2;                    /* HADDR: 0    */
     cdata=RSTSTB;
     outport(CADDR,cdata);
     outport(DADDR,0x88);
     wrhost();
}
void wrhost(void)
{
     cdata=SETSTB;
     outport(CADDR,cdata);       /* /HEN   */
     cdata=RSTSTB;
     outport(CADDR,cdata);
}
void readhost(void)
{
     cdata=0;
     cdata=RSTA0;
     cdata=RSTA1;
     cdata=RSTA2;                    /* HADDR: 0    */
     cdata=RSTSTB;
     outport(CADDR,cdata);
     wrhost();
     cdata=SETA0;
     cdata=RSTA1;
     cdata=RSTA2;                    /* HADDR: 1    */
     outport(CADDR,cdata);
     wrhost();
     cdata=RSTA0;
     cdata=SETA1;
```

```
    cdata=RSTA2;                    /* HADDR: 2    */
    outport(CADDR,cdata);
    wrhost();
    cdata=SETA0;
    cdata=SETA1;
    cdata=RSTA2;                    /* HADDR: 3    */
    outport(CADDR,cdata);
    wrhost();
}
```

read.c: Πρόγραμμα μεταφοράς αποτελεσμάτων αναγνώρισης απο το ADS στο PC, m;esv thw RS232

```
#include <stdio.h>
#include <conio.h>
#include <bios.h>
#include "serial.h"
#define COM4  0x2e8
#define SETPRM0
#define SEND  1
#define RECV  2
#define STATUS3
#define RESPONSEWIN     3,11,19,17
#define RESPONSEFRAME   1,10,20,18
#define RESPONSELAB      6,10
#define VALWIN           32,11,44,17
#define VALFRAME         25,10,45,18
#define VALLAB           29,10
#define VALHEX           27,12
#define VALDEC           27,11
#define STATUSFRAME     1,19,45,24
#define STATUSLAB        19,19
#define STATUSWIN        2,20,44,23
void Reset();
void ReadValues();
void DrawWindow(int x, int y, int x1, int y1);
void DownloadFrames();
main(void)
{
char ch=0,rx=0,lastrx=0;
int status;
int r1,r2,r3,dummy;
signed long result;
int i;
int rwinx=0,rwiny=0;
char Status;
     InitSerial();
     clrscr();
     printf("\nEnter\tR to arm for next word,\tD to download word
frames\n\tV to read back values\n\tEsc to quit\n");
     for (i=1;i<80;i++) printf("%c",196);
     printf("\n");
     DrawWindow(RESPONSEFRAME);
     gotoxy(RESPONSELAB);
     printf(" Receive ");
     DrawWindow(VALFRAME);
     gotoxy(VALLAB);
     printf(" Value Rx'ed ");
     gotoxy(VALDEC);
     printf("Dec:");
     gotoxy(VALHEX);
     printf("Hex:");
     DrawWindow(STATUSFRAME);
     gotoxy(STATUSLAB);
     printf(" Status ");
     while (ch!=27)
     {
```

```
          if (kbhit()) ch=getch();
          if ((inportb(COM4+5)&0x1f)==0x01)
          {
               rx=inportb(COM4);
               lastrx=rx;
               window(RESPONSEWIN);
               gotoxy(rwinx,rwiny);
               cprintf("\n\r%c (%x)  ",rx,rx);
               rwinx=wherex();
               rwiny=wherey();
               rx=0;
          }
          if (ch=='r'|| ch=='R')
          {
               Reset();
               ch=0;
          }
          if (ch=='v'|| ch=='V')
          {
               ReadValues();
               ch=0;
          }
          if ( (ch=='d'||ch=='D') && lastrx=='#')
          {
               DownloadFrames();
               ch=0;
               lastrx=0;
          }
     }
     window(1,1,80,25);
     clrscr();
     return(0);
}
void Reset()
{
     outportb(COM4, '!');    /* Send reset char. */
}
void ReadValues()
{
long result=0;
char ch;
char stat;
     while (!kbhit())
     {
          outportb(COM4,'@');
          ch=ReadSerial(COM4,&stat);
          if (stat==1) exit(1);  /*Discard first byte (is just echo)
*/
          result=Read24bitSigned(COM4);
          ch=ReadSerial(COM4,&stat);
          if (stat==1) exit(1);  /*Discard last byte too (is just
marker) */
          window(VALWIN);
          gotoxy(1,1);
          clreol();
          cprintf("% ld\n\r",result);
          clreol();
          cprintf(" %lx",result);
```

```
          delay(100);
     }
     ch=getch();
     clrscr();
}
void DrawWindow(int x, int y, int x1, int y1)
{
int i;
     gotoxy(x,y);
     putch(218);
     for (i=x+1;i<x1;i++) putch(196);
     putch(191);
     for (i=y+1;i<y1;i++)
     {
          gotoxy(x,i);
          putch(179);
          gotoxy(x1,i);
          putch(179);
     }
     gotoxy(x,y1);
     putch(192);
     for (i=x+1;i<x1;i++) putch(196);
     putch(217);
}
void DownloadFrames()
{
char ch,st;
int k,j;
long frames,i;
long frame[19*16];
FILE *outfile;
long buffer[600];
unsigned long r1,r2,r3;
     window(STATUSWIN);
     for(i=0;i<19;i++) frame[i]=0;
     clrscr();
     cprintf("Quering DSP...\r\n");
     outportb(COM4,'?');
     ch=ReadSerial(COM4,&st);
     if (st==1) exit(1);
     if (ch!='?') cprintf("Cannot communicate,\n\rplease reset and try
again.");
     frames=Read24bit(COM4);
     cprintf("There are %ld frames for download.\n\r",frames);
     cprintf("Downloading...\n\r");
     for (i=1;i<=frames;i++)
     {
          clreol();
          cprintf("%ld",i);
          outportb(COM4,'>');
          for (j=0;j<19*16;j++)
          {
               k=0;
               while( ((inportb(COM4)&0x1f) != 0x01) && k<5000 ) k++; /
*Wait*/
               r1=inportb(COM4);
               outportb(COM4,r1);
               k=0;
```

```
                while( ((inportb(COM4)&0x1f) != 0x01) && k<5000 ) k++; /
*Wait*/
                r2=inportb(COM4);
                outportb(COM4,r1);
                k=0;
                while( ((inportb(COM4)&0x1f) != 0x01) && k<5000 ) k++; /
*Wait*/
                r3=inportb(COM4);
                outportb(COM4,r1);
                frame[i,j]= r1<<24 | r2<<8 | r3;
           }
      }
/*   outfile=fopen("word.dat","wt");
     setbuf(outfile, buffer);
     fprintf(outfile,"\n");
     fflush(outfile);
     fclose(outfile);
*/
}
InitSerial()
{
char ch;
     printf("\n\nInitalizing...");
     outportb(COM4+1,0x00);  /* interrupts */
     printf("IRQ, ");
     outportb(COM4+3,0x80);  /* Select baud rate generator divider */
     printf("BAUD, ");
     outportb(0x2e8,0x0c);   /* Set divisor LSB=0c for 9600 (06 for
19200 bps) */
     outportb(0x2e9,0x00);   /* Set divisor MSB=0 */
     outportb(0x2eb,0x1b);    /* Select 8,e,1 */
     printf("Tx/Rx, ");
     ch=inportb(COM4);        /* make sure input is empty */
     ch=inportb(COM4);
     ch=inportb(COM4);
     ch=inportb(COM4+5);
     printf("Flush... ");
     printf("Done\n");
}
```

serial.c: Βιβλιοθήκη χρήσης της RS232 απο το PC.

```
char ReadSerialSigned(int comport, char *Status)
{
int i=0;
int addr=5;

     addr+=comport;
     while( ((inportb(addr)&0x1f) != 0x01) && i<5000 ) i++; /*Wait*/

     if (i<5000) *Status=0;
          else *Status=1;
     return(inportb(comport));
}

unsigned char ReadSerial(int comport, char *Status)
{
int i=0;
int addr=5;

     addr+=comport;
     while( ((inportb(addr)&0x1f) != 0x01) && i<5000 ) i++; /*Wait*/

     if (i<5000) *Status=0;
          else *Status=1;
     return(inportb(comport));
}

unsigned long Read24bit(int comport)
{
char status;
unsigned long ch1=0,ch2=0,ch3=0;

     ch1=ReadSerial(comport,&status);
     if (status==1) exit(1);
     outportb(comport,' ');

     ch2=ReadSerial(comport,&status);
     if (status==1) exit(1);
     outportb(comport,' ');

     ch3=ReadSerial(comport,&status);
     if (status==1) exit(1);
     outportb(comport,' ');
     return( ch1<<16 | ch2<<8 | ch3 );

}
long Read24bitSigned(int comport)
{
char status;
long ch1=0,ch2=0,ch3=0;

     ch1=ReadSerialSigned(comport,&status);
     if (status==1) exit(1);
     outportb(comport,' ');

     ch2=ReadSerialSigned(comport,&status);
```

```
    if (status==1) exit(1);
    outportb(comport,' ');

    ch3=ReadSerialSigned(comport,&status);
    if (status==1) exit(1);
    outportb(comport,' ');

    return( ch1<<16 | ch2<<8 | ch3 );

}
```

Προγράμματα DSP

Στην ενότητα αυτή παρουσιάζονται τα προγράμματα που γράφτηκαν για το DSP56000, και μιά συνοπτική λίστα των εντολών του.

Περίληψη εντολών του 56001

```
ABS <A,B>                       Take absolute value of the accumulator


ADC <X,Y>,<A,B>                 Add long with carry

ADD <A,B>,<B,A>                 Add 24/48/56 bits
ADD <X,Y>,<A,B>
ADD <Xn,Yn>,<A,B>

ADDL <A,B>,<B,A>                Left shift and add accumulators (*2)

ADDR <A,B>,<B,A>                Right shift and add accumulators (:2)

AND <Xn,Yn>,<A,B>               AND logical with accumulator (only A1/B1)
AND(I) #data8,<MR,CCR,OMR>      AND logical with control register

ASL <A,B>                       Arithmetic shift left accumulator (56bits)
ASR <A,B>                       Arithmetic shift left accumulator (56bits)

BCHG #n,<X,Y>:ea                Bit test and change
BCHG #n,register

BCLR #n,<X,Y>:ea                Bit test and clear
BCLR #n,register

BSET #n,<X,Y>:ea                Bit test and set
BSET #n,register

BTST #n,<X,Y>:ea                Bit test (move bit in carry)
BTST #n,register

CLR <A,B>                       Clear accumulator

CMP <A,B>,<B,A>                 Compare: s2-s1 (56 bits) (A1/B1 w/24 bits)
CMP <Xn,Yn>,<A,B>
CMPM <A,B>,<B,A>                Compare: |s2|-|s1| (56 bits) (A1/B1 w/ 24
CMPM <Xn,Yn>,<A,B>              bits)

DEBUG    Enter DEBUG mode
DEBUGcc

DEC <A,B>                       Decrement accumulator
DIV <Xn,Yn>,<A,B>               Divide accumulator (must be positive and
                                less than source

DO <X,Y>:ea,addr                Start hardware loop
DO #data12,addr
DO reg,addr

ENDDO                           Terminate hardware loop prematurely

EOR <Xn,Yn>,<A,B>               Exclusive or (A1/B1 - 24 bits only)

ILLEGAL                         Generate ILLEGAL exception
```

```
INC <A,B>                    Increase acumulator (56 bits)

JMP ea                       Jump unconditionally
JScc ea                      Jump to sub conditionally
Jcc ea                       Jump conditionally

      CC (HS)                carry clear (higher or same)
      CS (LO)                carry set (lower)
      EC                     extension clear
      ES                     extension set
      EQ                     equal
      NE                     not equal
      GT                     greater than
      GE                     greater than or equal
      LT                     less than
      LE                     less than or equal
      LC                     limit clear
      LS                     limit set
      PL                     plus (positive)
      MI                     minus (negative)
      NR                     normalized
      NN                     not normalized

JCLR #n,<X,Y>:ea, ea         Jump if bit clear
JCLR #n,reg, ea
JSET  #n,<X,Y>:ea, ea        Jump if bit set
JSET  #n,reg, ea

JSR ea   Jump to sub

JSCLR #n,<X,Y>:ea, ea        Jump to sub if bit clear
JSCLR #n,reg, ea
JSSET #n,<X,Y>:ea, ea        Jump to sub if bit set
JSSET #n,reg, ea

LSL <A,B>                    Logical shift left (24 bits) with carry
LSR <A,B>                    Logical shift right (24 bits) with carry
LUA ea,<Rn,Nn>               Load updated address

MAC (+/-)<Xn,Yn>, <Xn,Yn>, <A,B>
MAC <Xn,Yn>, #data24, <A,B>
                        Multiply the two sources and add the result to
                        the accumulator / Multiply by 2^(-data24).
MACR (+/-)<Xn,Yn>, <Xn,Yn>, <A,B>
MACR <Xn,Yn>, #data24, <A,B>
                        Multiply the two sources and add the result to
                        the accumulator / Multiply by 2^(-data24),then
                        round.
MOVE source,dest        Move data
MOVE(C) source,dest     Read/write control register
MOVE(M) source,dest     Read/write program memory
MOVE(P) source,dest     Read/Write I/O peripheral

MPY (+/-)<Xn,Yn>, <Xn,Yn>, <A,B>
MPY <Xn,Yn>, #data24, <A,B>
                        Multiply the two sources and move the resultto
                        the accumulator / Multiply by 2^(-data24),then
                        move.
```

```
MPYR (+/-)<Xn,Yn>, <Xn,Yn>, <A,B>
MPYR <Xn,Yn>, #data24, <A,B>
                        Multiply the two sources and move the result o
                        the accumulator / Multiply by 2^(-data24),then
                        move and round.

NEG <A,B>               Negate accumulator (56 bit, 2's copl.)

NOP                     Produce self-modifying code (viruses)

NORM Rn,<A,B>           Normalize accumulator iteration (56 bits)

NOT <A,B>               Coplement accumulator (A1/B1 - 24 bits)

OR <Xn,Yn>, <A,B>       Logical OR (A1/B1 - 24 bits)
ORI #data24,<MR,CCR,OMR> OR logical with control register

REP <X,Y>:ea            Repeat the next instruction that many times
REP #data12
REP reg

RESET                   Reset on-chip peripherals

RND <A,B>               Round accumulator (Scale bits affect result)

ROL <A,B>               Rotate left accumulator with carry (A1/B1-24)
ROR <A,B>               Rotate right accumulator with carry (A1/B1-24)

RTI                     Return from interrupt
RTS                     Return from subroutine

SBC <X,Y>, <A,B>        Subtract long (48-bit) with carry from acc

STOP                    Stop processor

SUB <A,B>, <B,A>        Subtract 24/48/56 bits
SUB <Xn,Yn>,<A,B>

SUBL <A,B>, <B,A>       Shift left and subtract accumulators
SUBR <A,B>, <B,A>       Shift right and subtract accumulators

SWI                     Trigger SWI exception

Tcc <A,B,Xn,Yn>,<A,B>  Rn,Rn
Tcc <A,B,Xn,Yn>, <A,B>
                        Move data conditionally

TFR <A,B>, <A,B>        Transfer data in ALU bypassing the shifter/
                        limiters
TFR <Xn,Yn>,<A,B>  (full 56 bits possible)
TST <A,B>Test accumulator (compare with zero)
WAIT Wait for interrupt (Halt mode)
```

main.asm: Κεντρικός βρόγχος προγράμματος

```
; Description   : Main program loop
; Code Number   : SRC-FRES0008agd
; Filename      : amain.asm
; Name          : Main
; Original Date :
; Current  Date :
; Revision      : 0.99
; Platform      : DSP56000 EVM
; Author        : AGD
; Approved by   :
;
          PAGE 132,45
          SECTION Main
          MODE RELATIVE

          include 'ioregs.asm'
;=========================================================================
          SECTION RamVars LOCAL
          org x:
v_LastCmd       dc 0    ;Last command sent by serial port
TaskStatus      dc 0    ;Task state machines and flags
v_LastSent      dc 0    ;Last value reported to serial port and DAC
;TaskStatus bits————————————————————————
DoFFT           equ 0
FFTDataOut      equ 1
WordDone_0      equ 2   ;00=Running             01=FFTs are stopped
WordDone_1      equ 3   ;10=11=Word is finished, ackn'ed by main.
ResultsReq      equ 4
          ENDSEC
;=========================================================================
          XDEF f_Tx_24
          XREF f_Init_Stack               ;Stack initalization

          XREF f_Init_FPGA                ;FPGA interface
          XREF f_Read_sw,f_Write_led
          XREF v_Leds,v_Switches

          XREF f_Init_RS232,f_Tx,f_Rx     ;RS232 interface
          XREF p_ser_status

          XREF f_Init_DAQ,f_Put_Sample    ;AD/DA interface
          XREF p_daq_status
          XREF v_Energy,v_Average         ;Timedom (Task1)
          XREF v_ZCross,v_WordLen
          XREF v_TimeStat
          XREF StartWord,StopWord         ;Bit positions in timestat

          XREF f_Init_FFT,ExecuteFFT      ;Freqdom (Task2)
          XREF ChkForFFTData
          XREF f_Init_PostProcess         ;ProcData (Task3)
          XREF ProcessData
          XREF FinalProcess,ReportData
          XREF v_QCollected
          PAGE
          org p:$200
```

```
;-----------------------------------------
StartMain       ;System Initalization
         ori #3,mr              ;Mask all interrupts off
         movep #0,IPR           ;Disable all interrupts
         movep #0,BCR           ;No wait states
         move #0,sp             ;Clear H/W stack pointer
         move #0,sr             ;Clear status register
         move #0,n7             ;Clear LastError status register
;-----------------------------------------
InitDebugPins   movep #0,PBC      ;make all of portB I/O (The FPGA
configures its own pins)
         movep #$7f00,PBDDR     ;High byte (7bits) are outputs
         movep #$0,PBD          ;Clear all bits
;-----------------------------------------
Init            ;System Setup
_InitMain       jsr f_Init_Main
_InitStack      jsr f_Init_Stack
_InitFPGA       jsr f_Init_FPGA
_Init232        jsr f_Init_RS232
_InitSSI        jsr f_Init_DAQ
_InitFFT        jsr f_Init_FFT
_InitProcData   jsr f_Init_PostProcess
         and #$fc,mr            ;Enable intr's in all levels
Init_Done
;-----------------------------------------
         PAGE
         bset #15,x:v_Leds      ;Show that the board is alive
;=====================================================================
;       MAIN LOOP
;=====================================================================
MLoop   bset #12,PBD
;-----------------------------------------
;Display Update
UpdLeds         jsr f_Write_led
;       led     function
;
;       0       Word start/in progress
;       1       Ready for word/Word stop
;       2
;       3
;       4
;       5 FFT in progress
;       6
;       7
;       8 Word too short to be analyzed
;       9       System error
;       10      RS232 Rx active
;       11      RS232 Tx active
;       12
;       13
;       14
;       15      System Init OK
;-----------------------------------------
;Switches Update
UpdSwitchStat   jsr f_Read_sw
;-----------------------------------------
;System error check and reporting
ChkError        move n7,a
```

```
          tst a                    ;Check if there is an error in n7
          jeq ChkError_end         ;if not, jump to next "task"
          rep #16
          rol a1
          jsr f_Tx                 ;Else, transmit error
          move #'*',a              ;and error marker
          jsr f_Tx
          move #0,n7               ;Clear error after trasmition
          bset #9,x:v_Leds         ;Indicate that a system error has
                        ;occured
ChkError_end
;---------------------------------------
;Check if finished processing one full word. If yes, prepair to
;transfer results (part of task 3)
CheckWordDone
          jclr #WordDone_0,x:TaskStatus,CheckWordDone_end ;If end not
detected, nop
          bclr #WordDone_0,x:TaskStatus   ;Else,
          bset #WordDone_1,x:TaskStatus   ;Acknowledge flag
          move #>32,x0                    ;Check if the word was long
          move x:v_QCollected,a           ;enough by checking the
number
          cmp x0,a                        ;of collected quantums.
          jge _Ok                         ;If more than 32, ok
          move #'~',a                     ;Else, send message that the
          jsr f_Tx                        ;word was rejected.
          bset #8,x:v_Leds                ;Notify user with TooShort
led.
          jmp CheckWordDone_end       ;and return
_Ok             move #'#',a
          jsr f_Tx                        ;and send message to the
host.
          jsr FinalProcess                ;Do final post-processing on
data.
CheckWordDone_end
;---------------------------------------
;This is the first part of task 3
PostProcessData jclr #FFTDataOut,x:TaskStatus,PostProcessData_end ;If
no data to process, jump
          bclr #FFTDataOut,x:TaskStatus           ;Clear flag
          jsr ProcessData                 ;Use FFT output to create
bands
PostProcessData_end
;---------------------------------------
;This is part of task 2
ChkDoFFT        jclr #DoFFT,x:TaskStatus,ChkDoFFT_end   ;If not enough
data, nop
          bclr #DoFFT,x:TaskStatus                ;Clear flag
     bset #14,PBD
          jsr ExecuteFFT                          ;and do fft.
     bclr #14,PBD
          bclr #7,x:v_Leds                        ;Clear FFT led
          bset #FFTDataOut,x:TaskStatus           ;Set flag showing
data ready
ChkDoFFT_end
;---------------------------------------
;This is a part of task 2, too.
ChkFFTData      jclr #StartWord,x:v_TimeStat,ChkFFTData_end ;If no
```

```
word detected, nop
         jsr ChkForFFTData                        ;See if there is
enough
                                  ;data for an FFT
         tst a                     ;Test return value in A
         jne _ChkWordDone          ;If not zero, go to next task
         bset #DoFFT,x:TaskStatus ;Else, data is ready for FFT.
         bset #7,x:v_Leds          ;Show at FFT led
_ChkWordDone    move #>1,x0
         cmp x0,a
         jne ChkFFTData_end              ;If word not done, move on.
         jset #WordDone_1,x:TaskStatus,ChkFFTData_end
                           ;If already ackn'ed flag, nop.
         bset #WordDone_0,x:TaskStatus   ;Else, notify all people
here
ChkFFTData_end
;————————————————————————————————
;Mostly a part of task3, although it is done in main()
ChkSerial       jclr #NewData,x:p_ser_status,ChkSerial_end ;If no new
char has
         bclr #NewData,x:p_ser_status    ;arrived in RS232,jump
         jsr f_Rx                        ;Else, RX it and loop it
back
         move a,x:v_LastCmd              ;and save it for reference
         jsr f_Tx
ChkSerial_end
;————————————————————————————————
;Use the DAC to report some internal thingies....
ChkSample       jclr #NewSample,x:p_daq_status,ChkSample_end
         bclr #NewSample,x:p_daq_status

         jset #0,x:v_Switches,_SendAverage ;If sw1, output the
average value
         jset #1,x:v_Switches,_SendEnergy  ;If sw2, output the mean
energy
         jset #2,x:v_Switches,_SendZCross  ;If sw3, the zero crossing
rate
         jset #3,x:v_Switches,_SendWordLen ;If sw4, send the last
word length
         ;jmp ChkSample_end                ;If none selected, do not
show anything
_SendSample     SAMPLE a
         jmp _Send
_SendAverage    move x:v_Average,a
         jmp _Send
_SendEnergy     move x:v_Energy,a
         jmp _Send
_SendZCross     move x:v_ZCross,a
         jmp _Send
_SendWordLen    move x:v_WordLen,a
         ;jmp _Send
_Send           move a,x:v_LastSent     ;Store for reference
         jsr f_Put_Sample
ChkSample_end

;————————————————————————————————
;
ReportResults
```

```
          jclr #ResultsReq,x:TaskStatus,ReportResults_end
          bclr #ResultsReq,x:TaskStatus
          jsr ReportData
ReportResults_end
;————————————————————————————————
;Check commands coming from the host computer
ChkCommand
          move x:v_LastCmd,b      ;Get last command
          tst b
          jeq ChkCommand_end      ;If no command, return
_ChkReport      move #'@',x0            ;Check if it was a '@'
          cmp x0,b
          jne _ChkReset           ;If not, check if it was a reset
command

          clr b
          move b,x:v_LastCmd      ;If yes, clear and execute command.
          move x:v_LastSent,a     ;Get last value reported on DAC
          jsr f_Tx_24             ;Send the whole a1 up the serial
port
          move #'@',a
          jsr f_Tx                ;Send a marker, too
          jmp ChkCommand_end

_ChkReset       move #'!',x0            ;Check if it is a reset
command
          cmp x0,b
          jne _ChkReadResult      ;If not, see next option
          clr b
          move b,x:v_LastCmd      ;If yes, clear and execute command.
          ;jmp Init       ;This in final version
          clr b           ;And these for now...
          move b,x:v_TimeStat
          jsr f_Init_FFT
          jsr f_Init_PostProcess
          jsr f_Init_Main
          bclr #9,x:v_Leds        ;Clear System Error led
          bclr #8,x:v_Leds        ;Clear WordTooShort led
          jmp ChkCommand_end
_ChkReadResult  move #'?',x0
          cmp x0,b
          jne _NxtCmd
          clr b
          move b,x:v_LastCmd
          bset #ResultsReq,x:TaskStatus
          ;jmp ChkCommand_end
_NxtCmd                 ;Might add some commands to change the energy
threshold
               ;and other system "constants"
ChkCommand_end
;————————————————————————————————
NxtLoop
          bclr #12,PBD
          jmp MLoop
;==============================================================================
          PAGE
;————————————————————————————————
;Initialize main loop variables
```

```
f_Init_Main     move #0,x0
          move x0,x:TaskStatus
          move x0,x:v_LastCmd
          rts
;————————————————————————————————————
;Transmit a full 24-bit number (in a1) up the serial port
f_Tx_24
          tfr a,b
          jsr f_Tx                ;Send last value to serial channel,
          jsr f_Rx
          tfr b,a
          rep #8                  ;first high,
          lsl a1                  ;
          jsr f_Tx                ;then middle,
          tfr a,b
          jsr f_Rx
          tfr b,a
          rep #8                  ;then low byte
          lsl a1                  ;
          jsr f_Tx
          jsr f_Rx
          rts
f_Tx_24_end
;*************************************************************************
          ENDSEC
     END
```

daqint.asm: Επικοινωνία με το A/D και το D/A

```
; DAQ Serial Interface
; Part of the I/O library functions

; Description   : Analog I/O buffered interface
; Code Number   : SRC-FRES0008-ILX
; Filename      : DAQINT.ASM
; Name          : DAQINT
; Original Date : 25/10/95
; Current  Date : 01/08/96
; Revision      : 1.0
; Platform      : EVM56001
; Author        : Alexopoulos Ilias
; Approved by   : Alexopoulos Ilias
;                 Diamantis Argiris


          page 132,45

          OPT DEX,INTR


;*********************************************************************

          SECTION DAQINT
          include 'ioregs.asm'

;*********************************************************************

          XDEF f_Init_DAQ,f_Put_Sample
          XDEF i_lssirx,i_lssirxe,i_lssitx,i_lssitxe
          XDEF p_daq_status

          XREF EnergyAndAverage,ZCrossings,UpdateTimeEst

;————————————————————————————

c_dacmodulo     EQU 512                         ;DAC modulo for buffer
c_adcmodulo     EQU 1024                        ;ADC modulo for buffer

;====================================================================
          SECTION RamVars LOCAL   ;Made local to avoid XDEFs for
DAQINT
          org x:
p_daq_status:   dc 0                    ;DAQ status flag
p_dacint:       dc buf_dac              ;DAC TX Interrupt Pointer
p_dacapp:       dc buf_dac              ;DAC TX Application Pointer
p_adcint:       dc buf_adc              ;ADC RX Interrupt Pointer
p_adcapp:       dc buf_adc              ;ADC RX Application Pointer
          ENDSEC
;====================================================================

;library data
          org y:
buf_adc:        bsm c_adcmodulo,0     ;Input Serial Reception buffer &
init
```

```
buf_dac:        bsm c_dacmodulo,0       ;Output Serial Transmition
buffer &init


          PAGE

;---------------------------------------
; Software Calls to Library
; Plus Sequence of Operations
;
; Function              Parameters              Description
;
; Init_DAQ              None                    Initialize DAQ library
; Get_Sample            A                       Receive a Sample
; Put_Sample            A                       Trasmit a Sample
;---------------------------------------

;=====================================================================

          org p:

;---------------------------------------
f_Init_DAQ:

          bclr #SSI_SAMPLE_OUT,x:p_daq_status ;Clear SampleOut bit on
our Serial Flag
          bclr #SSI_SAMPLE_IN,x:p_daq_status  ;Clear SampleIn on our
Serial Flag


          movep #$4000,SSICRA     ;Set 16-bit words no clock
prescaling
          movep #$F200,SSICRB     ;Set ext. clk, MSB first WL bit
frame
                        ;Synchronous Rx/Tx,SC2=in
                        ;Enable Tx/Rx interrupts


          movep PCC,x0
          move #$01FC,a1          ;Enable SSI STD,SRD,SCK,SC2 pins on
DSP
          or x0,a1
          movep a1,PCC

          bset #13,IPR            ;Enable SSI interrupt at medium
          bclr #12,IPR            ;priority

          bclr #SSITIE,SSICRB     ;No transmit Interrupt yet

          move #buf_adc,r3        ;Initialize R3 Global Sample Pointer
          move #(c_adcmodulo-1),m3        ;in Modulo fashion
          move #0,n3              ;and n3 clear for use

          rts

;---------------------------------------
f_Put_Sample:
          PUSH    r6                      ;Save Registers
```

```
          PUSH    m6

          move x:p_dacapp,r6              ;Application buffer pointer
          move #(c_dacmodulo-1),m6        ;and modulo loading
          nop
          move a,y:(r6)+                  ;Transfer data
          move r6,x:p_dacapp              ;save next data address
          bset #SSI_SAMPLE_OUT,x:p_daq_status    ;TX busy now

          bset #SSITE,SSICRB
          bset #SSITIE,SSICRB             ;Reenable DAC interrupt

          POP     m6                      ;Restore regs
          POP     r6
          rts
;-----------------------------------------
;-----------------------------------------

          PAGE

;*********************************************
;*      SSI INTERFACE   /DAQ                 *
;*      ISR                                  *
;*********************************************

;=============================================================================
          org p:
i_lssitxe:
i_lssitx:

          jclr #SSI_SAMPLE_OUT,x:p_daq_status,txd_int_end ;If no more
to send, return

          PUSH    r6                      ;Save Registers
          PUSH    m6
          PUSH    a0
          PUSH a1
          PUSH a2
          PUSH    x0

          movep SSISR,a                   ;clear TUE & Pipeline delay
          move #(c_dacmodulo-1),m6        ;Transmiter buffer modulo
          move x:p_dacint,r6              ;Initialise buffer pointer
          nop
          move y:(r6)+,a
          movep a,SSIDATA                 ;Transmit
          move r6,x:p_dacint              ;Store New Interrupt
buf.pointer

          move r6,x0                      ;Compare with application
buf. pointer
          move x:p_dacapp,a
          cmp x0,a                        ;If not equal then TX_BUSY
          jneq _txdend

          bclr #SSI_SAMPLE_OUT,x:p_daq_status
                              ;No further transaction
_txdend:
```

```
          POP x0                          ;Restore Registers
          POP a2
          POP a1
          POP a0
          POP m6
          POP r6
txd_int_end
          rti
;-----------------------------------------

          PAGE

;-----------------------------------------
i_lssirx:       bset #10,PBD            ;Set debug pin

          PUSH a0
          PUSH a1
          PUSH a2

          movep SSIDATA,a

          ;bchg #TrashSample,x:p_daq_status        ;Change bit
          ;jcs i_lssirx_ret                ;If bit was 1, skip this
sample

          ;R3 Global Sample pointer to FIFO Buffer (Current position)

          move a,y:(r3)+;Receive data
          move r3,x:p_adcint              ;Save next available pointer
                              ;location
          ;move a,SSIDATA         ;Loopback test

          bset #NewSample,x:p_daq_status  ;indicate a new sample has
arrived

          ;-------------------------
          PUSH r0                 ;These regs are used by all
          PUSH m0                 ;the following functions,
          PUSH n0                 ;so they are saved here to
          PUSH x0                 ;speed-up execution
          PUSH x1
          PUSH b0                 ;A, x0, r0 are destroyed.
          PUSH b1
          PUSH b2

                jsr EnergyAndAverage      ;Do short-time time domain
                              ;calculations on signal
          jsr ZCrossings

          jsr UpdateTimeEst       ;Update flags for other tasks

          POP b2
          POP b1
          POP b0
          POP x1
          POP x0
          POP n0
```

```
          POP m0
          POP r0
          ;-------------------------
          bclr    #SSITIE,SSICRB          ;Disable TX interrupt

i_lssirx_ret
          POP a2                          ;Restore registers used
here.
          POP a1
          POP a0

          bclr #10,PBD           ;Clear debug pin
          rti                             ;Also POPs the SR
;---------------------------------------
i_lssirxe:
          PUSH a0
          PUSH a1
          PUSH a2

          movep SSISR,a1                  ;Clear ROE
          movep SSIDATA,a1
          move a1,y:(r3)+                 ;Receive data
          move r3,x:p_adcint              ;Save next available pointer
                             ;location
          bset #NewSample,x:p_daq_status  ;indicate a new sample has
arrived
          move #SSI_RX_ERROR,n7           ;Store last error

          POP a2                          ;Restore used registers
          POP a1
          POP a0
          rti
;---------------------------------------

          ENDSEC                          ;End of DAQ ISRs
;*************************************************************************


          END
```

fpgaint.asm: Επικοινωνία με το fpga

```
; FPGA Serial Interface
; Part of the I/O library functions
; Description   : Digital I/O interface
; Code Number   : SRC-FRES0008-ILX
; Filename      : FPGAINT.ASM
; Name          : FPGA Interface
; Original Date : 11/07/96
; Current  Date : 08/08/96
; Revision      : 1.1
; Platform      : EVM56001
; Author        : Alexopoulos Ilias
; Approved by   : Alexopoulos Ilias
;                 Diamantis Argiris

         page 132,45
         OPT DEX,INTR
;***************************************************************************
         SECTION FPGA
         include 'ioregs.asm'
;***************************************************************************

         XDEF f_Init_FPGA,f_Read_sw,f_Write_led

;---------------------------------------
; Hardware connection to the FPGA
; PB0: PCLK  ( Peripheral Clock )
; PB1: PDAT  ( Peripheral Data  )
; PB2: PCE   ( Peripheral Chip Enable )
; PB3: PRW   ( Peripheral Read-/Write )
;---------------------------------------
; Bit positions
b_PCLK          EQU     0
b_PDAT          EQU     1
b_PCE           EQU     2
b_PRW           EQU     3

c_FPGA_rst      EQU     $C      ;PCLK,PDAT=0, PCE,PRW=1

;***************************************************************************
         SECTION RamVars Local
         XDEF v_Leds
         XDEF v_Switches
         org x:
v_Leds          dc 0
v_OldLeds dc 0
v_Switches      dc 0
         ENDSEC
;***************************************************************************
         PAGE
;---------------------------------------
; Software Calls to Library
; Plus Sequence of Operations
;
;***************************************************************************
; Function              Parameters              Description
```

```
;
; f_Init_FPGA           None                    Initialise FPGA
library and Led
;
; f_Read_sw             None, uses A            Read switch position
(32:24)
;                                               32=>SW8
;                                               24=>SW1
;
; f_write_led           None, uses A            Write Led display
(40:24)
;                                               40=>LD15
;                                               32=>LD08
;
;                                               31=>LD07
;                                               24=>LD00
;*********************************************************************
;=====================================================================

          org p:
f_Init_FPGA:
          bclr #0,PBC             ;Port B as parallel interface
          movep PBDDR,x0          ;Previous Direction condition
          move #$0d,a1            ;PCLK,PCE,PRW as output pins on DSP
          or x0,a1
          movep a1,PBDDR          ;Setup port B
          movep #c_FPGA_rst,PBD   ;Reset output lines on FPGA
          move #0,x0
          move x0,x:v_Leds
          move x0,x:v_OldLeds
          move x0,x:v_Switches
          rts
;_____________________________________
f_Read_sw:
          clr a
          bset #b_PRW,PBD                 ;Read mode
          bclr #b_PCE,PBD                 ;Chip enable

          do #8,_rd_loop
          bset #b_PCLK,PBD                ;Cycle one clock
          bclr #b_PCLK,PBD
          jclr #b_PDAT,PBD,_SW_low        ;Check data bit
          bset #0,a1                      ;High? then set accum.
_SW_low:
          lsl a                           ;Shift result
_rd_loop:
          bset #b_PCE,PBD                 ;chip disable
          lsr a                           ;7 rotations not 8
          move a,x:v_Switches
          rts
;_____________________________________
f_Write_led:
          move x:v_Leds,a
          move x:v_OldLeds,x0
          cmp x0,a           ;Check if there is any change
          jeq f_Write_led_ret         ;If not, return
          bclr #b_PRW,PBD                 ;Write mode
          bclr #b_PCE,PBD                 ;Chip enable
```

```
         bset #b_PDAT,PBDDR              ;PDAT is output
         move a,x:v_OldLeds
         rep #7                          ;Trash last 7 bits
         lsl a                           ;position bit 40 to 47
         do #16,_wr_loop                 ;shift data
         bclr #b_PDAT,PBD                ;clear PDAT line
         lsl a
         jpl _led_low                    ;Check data bit
         bset #b_PDAT,PBD                ;High? then set PDAT
_led_low:

         bset #b_PCLK,PBD                ;Cycle one clock
         bclr #b_PCLK,PBD
_wr_loop:
         bset #b_PCE,PBD                 ;chip disable
         bclr #b_PDAT,PBDDR              ;PDAT is input
f_Write_led_ret
         rts

         ENDSEC
;*****************************************************************
         END
```

timedom.asm: Επεξεργασία σήματος στο πεδίο του χρόνου

```
; Description   : Time-domain functions in task1
; Code Number   : SRC-FRES0008agd
; Filename      : timedom.asm
; Name          : Timedom
; Original Date :
; Current  Date :
; Revision      : 0.60
; Platform      : DSP56001 EVM
; Author        : agd
; Approved by   : ilx
;
     PAGE 132,45
          SECTION Timedom
          MODE RELATIVE
          XDEF EnergyAndAverage,ZCrossings
          XDEF UpdateTimeEst
          XREF v_Leds
;*****************************************************************
          SECTION RamVars LOCAL
          XDEF v_Energy,v_Average,v_ZCross
          XDEF p_WordStart,p_WordEnd,v_WordLen
          XDEF v_TimeStat
          XDEF StartWord,StopWord
          org x:
v_Energy        dc 0
v_EnergyL       dc 0
v_Average       dc 0
v_ZCross        dc 0
v_TimeStat      dc 0
StartWord        equ     0
StopWord         equ     1
p_WordStart     dc 0
p_WordEnd       dc 0
v_WordLen       dc 0
v_ZeroCount     dc 0
c_ThrEnerHigh   dc     $2500
c_ThrEnerLow    dc     $500
c_MaxZeros      dc     1500            ;150 mSec of samples @ fs=10kHz

          ENDSEC
;*****************************************************************
TimeWin         equ     128             ;Window to perform function
on.
NormDivisor     equ     7               ;This is TimeWin (x) =
2^NormDivisor

ZStep           equ     $1000          ;Increment vars for zero
crossings
ZDiv            equ     4


PBD     define  'x:$FFE4'
;================================================================
          PAGE
          ORG p:
```

```
;=====================================================================
;       Mean Average Energy and Magnitude function
;               r3 is the pointer to the last sample in the array
;               a, b, x0 and r0 are destroyed
;
;=====================================================================
EnergyAndAverage
         PUSH y0
         move m3,m0              ;r0 is to operate like r3
         move r3,r0
         move #TimeWin,n0        ;Point to the sample leaving the
window
         clr a                   ;We'll be using both accumulators
         clr b   (r0)-           ;Get in-sync with the buffer

         move y:(r0),a           ;Get the sample entering the window
and
         lua (r0)-n0,r0          ;point to the last sample in the
window.
         move a,y0               ;This is the new sample and
         move y:(r0),x0          ;this is the sample leaving the
window.
         sub x0,a                ;Subtract the sample leaving from
the one entering
         mac y0,y0,b             ;Square of the sample entering
         mac -x0,x0,b            ;and subtract the square of the
sample leaving
                        ;from the square of the sample entering
         do #NormDivisor,_Scale  ;Scale the result
         asr a
         asr b
_Scale
         move x:v_Average,x0     ;Now get the old average
         add x0,a                ;and update it with the value in acc
         move a,x:v_Average      ;Store updated average
         move x:v_EnergyL,x0     ;Now get the old energy
         move x:v_Energy,x1
         add x,b                 ;and update it with the value in acc
_Store          move b0,x:v_EnergyL     ;Store updated energy
         move b1,x:v_Energy
         POP y0
         rts
EnergyAndAverage_end
;=====================================================================
;
;               r3 is the pointer to the last sample in the array
;               a, b, x0, x1, y0 and r0 are destroyed
;
;=====================================================================
ZCrossings      PUSH y0
         move r3,r0
         move m3,m0
         move #TimeWin-1,n0      ;Pointer to the next sample
         move y:(r0)-,x0         ;Get in-sync with the large buffer
         move y:(r0)-,x0         ;Get the 2 fresh samples
         move y:(r0),x1

         lua (r0)-n0,r0          ;Point to the sample leaving the
```

```
window
          mpy x0,x1,a             ;Check sign
          move a,y0               ;and save result

          move y:(r0)-,x0         ;Get the last sample in the window
          move y:(r0),x1          ;Get the sample leaving the window
          mpy x0,x1,a             ;Check the old samples for sign
change
          move a,x0
          mpy y0,x0,a             ;And check the old and new changes
          jpl _NoChange           ;If equal changes, nothing to do
          move y0,a               ;Else check most recent sign change

          move #>((ZStep/ZDiv)),y0        ;Add or subract
          move #>(-(ZStep/ZDiv)),x0       ;an increment in b
          move y0,b                       ;depending on whether
          tst a                           ;we have an increase or
          tpl x0,b                        ;a decrease of crossings
          move x:v_ZCross,a
          add b,a
          tst a                   ;If A tries to go below zero band,
          jpl _StoreChange        ;hold it at 0.
          clr a
_StoreChange    move a,x:v_ZCross
_NoChange

          POP y0
          rts
ZCrossings_end
          PAGE
;=========================================================================
;
;    Updates time estimates (energy, zcrossings,start/stop)
;    and makes them available at the corresponding variables.
;
;=========================================================================

UpdateTimeEst   bclr #10,PBD
          jset #StartWord,x:v_TimeStat,ChkForStop
ChkForStart                               ;If already started, just
check for stop
          move x:v_Energy,a       ;Get current energy value
          move x:c_ThrEnerHigh,x0 ;Get high threshold for energy
          cmp x0,a                ;compare x0 with a1
          jlt _DontStart          ;If a<x0 then no start
_StartFFTs      bset #StartWord,x:v_TimeStat  ;Else, start the FFTs
          move #768,n3       ;Point back a few frames
     bset #0,x:v_Leds
          lua (r3)-n3,r0     ;Make r0 point back the frames
          move #0,x0
          move r0,x:p_WordStart   ;Save start of word for use by fft
          move x0,x:v_WordLen     ;Reset word length
          jmp UpdateTimeEst_ret   ;And return
_DontStart bclr #0,x:v_Leds
ChkForStop
          jclr #StartWord,x:v_TimeStat,UpdateTimeEst_ret
                        ;If not yet started, no need to check for stop
          jset #StopWord,x:v_TimeStat,UpdateTimeEst_ret
```

```
                        ;If already stopped, nothing to do

         move #>1,x0
         move x:v_WordLen,a ;This has to be one of the samples in
         add x0,a                ;the word, so increase the count
         move a,x:v_WordLen
         move x:v_Energy,a       ;Get energy
         move x:c_ThrEnerLow,x0    ;and compare with low threshold
         cmp x0,a
         jgt _ClearCount         ;If within active limits, don't stop
ffts
_CountZeros                                  ;Else, ( if a<x0 )
         move #>1,x0
         move x:c_MaxZeros,x1
         move x:v_ZeroCount,a       ;Get count of "zeros" in input
signal
         add x0,a                   ;so far, and increase them by one
         cmp x1,a a,x:v_ZeroCount   ;Compare with MaxZeros and update
count
         jlt _DontStop              ;If not yet reached max count
(a<MaxZeros), do not stop
_StopFFTs       bset #StopWord,x:v_TimeStat ;If max count reached,
raise flag
     bset #1,x:v_Leds
         move r3,x:p_WordEnd     ;Store word end
         jmp UpdateTimeEst_ret   ;and return
_ClearCount     move #0,x0
         move x0,x:v_ZeroCount
_DontStop bclr #1,x:v_Leds
UpdateTimeEst_ret
         bset #10,PBD
         rts
UpdateTimeEst_end
;=====================================================================
         ENDSEC ;Timedom
;*********************************************************************
     END
```

freqdom.asm: Επεξεργασία σήματος στο πεδίο της συχνότητας

```
; Description   : Frequency-domain functions in task2
; Code Number   : SRC-FRES0008agd
; Filename      : freqdom.asm
; Name          : Freqdom
; Original Date :
; Current  Date :
; Revision      : 0.50
; Platform      : DSP56001 EVM
; Author        : agd
; Approved by   : ilx
;
     PAGE 132,45
;=========================================================================
          SECTION Freqdom
          MODE RELATIVE
          XDEF f_Init_FFT,ChkForFFTData,ExecuteFFT
          XDEF FFTBuffer          ;For task 3

          XREF p_WordStart,p_WordEnd,v_WordLen
          XREF v_TimeStat,StopWord
;=========================================================================
          SECTION RamVars LOCAL
          ;XDEF
          org x:
v_BlocksDone    dc 0            ;Blocks of FFT data proccessed

          ENDSEC
;=========================================================================
;Equates
coef    equ     $100            ;address of sine-cosine coefficient
table
points  equ     256             ;FFT length
FFTsize equ     points
;————————————————————————————————
          sincos points,coef      ;sine & cosine tables for X and Y
memory

          ORG x:$200              ;Just after the sine table
FFTReal         bsr 256,0               ;Initialize a circular buffer
for use
                        ;by the FFT (Real part)
          ORG y:$200              ;The same here for the imaginary
part
FFTImag         bsr 256,0               ;

FFTBuffer       equ     FFTReal         ;This can be used for either
of the above

          org y:$0                ;Window coefficients buffer
          include 'hamming.inc'    ;(256 words long)
          PAGE
          ORG p:
;————————————————————————————————
ExecuteFFT
          bclr #11,sr     ;Set down-scaling mode
```

```
         bset #10,sr

         fftr2cc points,FFTBuffer,coef
         ;fftas points,FFTBuffer,coef
         ;fftbf points,FFTBuffer,coef

         bclr #10,sr     ;No scaling
         bclr #11,sr
         rts
ExecuteFFT_end
;————————————————————————————————
;————————————————————————————————
f_Init_FFT                              ;Initialize FFT and other
task2 functions
         clr a
         move a,x:v_BlocksDone
f_Init_FFT_ret
         rts
f_Init_FFT_end
         PAGE
;————————————————————————————————
;Checks to see if there is enough data gathered from the AD to
;do one FFT. If yes, copies data into the fft buffer.
ChkForFFTData   move x:v_BlocksDone,a   ;Get blocks processed so far.
         move #>1,x0             ;Increment block count
         add x0,a
         move a,x0
         move #>FFTsize/2,y0     ;and calculate:
         mpy  x0,y0,a            ;       (BlocksDone*128)+128
(overlapping FFTs)
         asr a                   ;compensate for fractional/integer
mpy
         move a0,a1              ;Bring result from a0 to a1
         rnd a x:v_WordLen,x0    ;Get word length
         cmp x0,a                ;If word length is less than acc
         jge _ChkStop            ;check if we have a stop. If yes,
                        ;we should be done with this word.
         move m3,m0              ;r0 works in Sample Buffer, like r3
         move #$ffff,m1          ;m1, m4 linear addressing
         move m1,m4
         move x:p_WordStart,r0   ;Pointer to samples
         move #FFTReal,r1        ;Real part in fft buffer
         move #FFTImag,r4        ;Imaginary part,
         move #0,y0              ;which is 0.
         move #$ffffff,m5        ;Pointer to window buffer
         move #Window,r5
         do #FFTsize,_CopyBuffer ;Copy samples to FFT buffer, also
apply window
         move y:(r0)+,x0         ;Get sample from samples buffer
         move y:(r5)+,y1         ;Get window coefficient
         mpyr x0,y1,a    y0,y:(r4)+      ;Apply window, store imag.
part
         move a,x:(r1)+                  ;Store result (real part)
_CopyBuffer
         move (r0)-
         move x:v_BlocksDone,a   ;Get blocks done, trim r0
         move r0,x:p_WordStart   ;store new start of word for next
pass
```

```
          move #>1,x0
          add x0,a                ;Increment block count
          move a,x:v_BlocksDone
          clr a                   ;Indicate to Main that there are
more data
          jmp ChkForFFTData_ret   ;and return.
_ChkStop        jclr #StopWord,x:v_TimeStat,ChkForFFTData_ret
                        ;If stop is not detected, wait for more data
                        ;If stop is detected and there are no
                        ;more data to look, our work is done.
          move #>1,a              ;Show that the FFTs are done
analyzing
                        ;this word and return.
ChkForFFTData_ret
          rts
ChkForFFTData_end
;_____________________________________
          ENDSEC ;Freqdom
;===============================================================================
     END
```

fftr2cc.asm: Radix 2, DIT, in-place FFT

```
;This program originally available on the Motorola DSP bulletin board.
; It is provided under a DISCLAMER OF WARRANTY available from
; Motorola DSP Operation, 6501 Wm. Cannon Drive W., Austin, Tx.,
78735.
;
; Radix 2, In-Place, Decimation-In-Time FFT (fast).
; Last Update 18-Aug-88   Version 1.0
fftr2cc  macro   points,data,coef
fftr2cc  ident   1,0
;
; Radix 2 Decimation in Time In-Place Fast Fourier Transform Routine
;
;    Complex input and output data
;        Real data in X memory
;        Imaginary data in Y memory
;    Normally ordered input data
;    Bit reversed output data
;       Coefficient lookup table
;        -Cosine values in X memory
;        -Sine values in Y memory
; Macro Call - fftr2cc   points,data,coef
;       points     number of points (16-32768, power of 2)
;       data       start of data buffer
;       coef       start of sine/cosine table
; Alters Data ALU Registers
;       x1      x0      y1      y0
;       a2      a1      a0      a
;       b2      b1      b0      b
; Alters Address Registers
;       r0      n0      m0
;       r1      n1      m1
;               n2
;
;       r4      n4      m4
;       r5      n5      m5
;       r6      n6      m6
;
; Alters Program Control Registers
;       pc      sr
;
; Uses 6 locations on System Stack
;
     move #data,r0            ;initialize input pointer
     move #points/4,n0        ;initialize input and output pointers
offset
     move n0,n4               ;
     move n0,n6               ;initialize coefficient offset
     move #points-1,m0        ;initialize address modifiers
     move m0,m1               ;for modulo addressing
     move m0,m4
     move m0,m5
;
; Do first and second Radix 2 FFT passes, combined as 4-point
butterflies
;
```

```
     move           x:(r0)+n0,x0
     tfr  x0,a      x:(r0)+n0,y1

     do   n0,_twopass
     tfr  y1,b      x:(r0)+n0,y0
     add  y0,a      x:(r0),x1                       ;ar+cr
     add  x1,b      r0,r4                           ;br+dr
     add  a,b       (r0)+n0                         ;ar'=(ar+cr)+(br+dr)
     subl b,a       b,x:(r0)+n0                     ;br'=(ar+cr)-(br+dr)
     tfr  x0,a      a,x0           y:(r0),b
     sub  y0,a                     y:(r4)+n4,y0    ;ar-cr
     sub  y0,b      x0,x:(r0)                       ;bi-di
     add  a,b                      y:(r0)+n0,x0    ;cr'=(ar-cr)+(bi-di)
     subl b,a       b,x:(r0)                        ;dr'=(ar-cr)-(bi-di)
     tfr  x0,a      a,x0           y:(r4),b
     add  y0,a                     y:(r0)+n0,y0    ;bi+di
     add  y0,b      x0,x:(r0)+n0                    ;ai+ci
     add  b,a                      y:(r0)+,x0      ;ai'=(ai+ci)+(bi+di)
     subl a,b                      a,y:(r4)+n4     ;bi'=(ai+ci)-(bi+di)
     tfr  x0,a                     b,y:(r4)+n4
     sub  y0,a      x1,b                            ;ai-ci
     sub  y1,b      x:(r0)+n0,x0                    ;dr-br
     add  a,b       x:(r0)+n0,y1                    ;ci'=(ai-ci)+(dr-br)
     subl b,a                      b,y:(r4)+n4     ;di'=(ai-ci)-(dr-br)
     tfr  x0,a                     a,y:(r4)+
_twopass
;
; Perform all next FFT passes except last pass with triple nested DO
loop
;
     move #points/8,n1         ;initialize butterflies per group
     move #4,n2                ;initialize groups per pass
     move #-1,m2               ;linear addressing for r2
     move #0,m6                ;initialize C address modifier for
                               ;reverse carry (bit-reversed) addressing
     do   #@cvi(@log(points)/@log(2)-2.5),_end_pass    ;example: 7
passes for 1024 pt. FFT
     move #data,r0                                     ;initialize A
input pointer
     move r0,r1
     move n1,r2
     move r0,r4                                        ;initialize A
output pointer
     move (r1)+n1                                      ;initialize B
input pointer
     move r1,r5                                        ;initialize B
output pointer
     move #coef,r6                                     ;initialize C
input pointer
     lua  (r2)+,n0                                     ;initialize
pointer offsets
     move n0,n4
     move n0,n5
     move (r2)-                                        ;butterfly loop
count
     move           x:(r1),x1      y:(r6),y0          ;lookup -sine
and -cosine values
     move           x:(r6)+n6,x0   y:(r0),b           ;update C
```

```
pointer, preload data
     mac  x1,y0,b                  y:(r1)+,y1
     macr -x0,y1,b                 y:(r0),a

     do   n2,_end_grp
     do   r2,_end_bfy
     subl b,a       x:(r0),b       b,y:(r4)            ;Radix 2 DIT
butterfly kernel
     mac  -x1,x0,b  x:(r0)+,a      a,y:(r5)
     macr -y1,y0,b  x:(r1),x1
     subl b,a       b,x:(r4)+      y:(r0),b
     mac  x1,y0,b                  y:(r1)+,y1
     macr -x0,y1,b  a,x:(r5)+      y:(r0),a
_end_bfy
     move (r1)+n1
     subl b,a       x:(r0),b       b,y:(r4)
     mac  -x1,x0,b  x:(r0)+n0,a    a,y:(r5)
     macr -y1,y0,b  x:(r1),x1      y:(r6),y0
     subl b,a       b,x:(r4)+n4    y:(r0),b
     mac  x1,y0,b   x:(r6)+n6,x0   y:(r1)+,y1
     macr -x0,y1,b  a,x:(r5)+n5    y:(r0),a
_end_grp
     move n1,b1
     lsr  b    n2,a1     ;divide butterflies per group by two
     lsl  a    b1,n1     ;multiply groups per pass by two
     move a1,n2
_end_pass
;
; Do last FFT pass
;
;
        move    #2,n0           ;correct input pointer A offset for
last pass
        move    n0,n1           ;correct input pointer B offset for
last pass
        move    n0,n4           ;correct output pointer A offset for
last pass
        move    n0,n5           ;correct output pointer B offset for
last pass
        move    #data,r0        ;initialize A input pointer
        move    r0,r4           ;initialize A output pointer
        lua     (r0)+,r1        ;initialize B input pointer
        move    #coef,r6        ;initialize C input pointer
        lua     (r1)-n1,r5      ;initialize B output pointer
        move    x:(r1),x1       y:(r6),y0
        move    x:(r5),a        y:(r0),b

        do      n2,_lastpass    ;Radix 2 DIT butterfly kernel with one
        mac     x1,y0,b x:(r6)+n6,x0    y:(r1)+n1,y1    ;butterfly per
group
        macr    -x0,y1,b        a,x:(r5)+n5     y:(r0),a
        subl    b,a     x:(r0),b        b,y:(r4)
        mac     -x1,x0,b        x:(r0)+n0,a     a,y:(r5)
        macr    -y1,y0,b        x:(r1),x1       y:(r6),y0
        subl    b,a             b,x:(r4)+n4     y:(r0),b
_lastpass
        move    a,x:(r5)+n5
        endm
```

procdata.asm: Τελική επεξεργασία αποτελεσμάτων

```
; Description   : Data processing functions for task3
; Code Number   : SRC-FRES0008agd
; Filename      : procdata.asm
; Name          : ProcData
; Original Date :
; Current  Date :
; Revision      : 0.60
; Platform      : DSP56001 EVM
; Author        : agd
; Approved by   : ilx
;
     PAGE 132,45
;=========================================================================
          SECTION ProcData
          MODE RELATIVE
          XDEF f_Init_PostProcess,ProcessData,ReportData,FinalProcess
          XDEF v_QCollected
          XREF FFTBuffer,f_Put_Sample,v_Switches,f_Tx_24,f_Rx
;=========================================================================
          SECTION RamVars LOCAL
          ;XDEF
          org x:
p_QuantumTop    dc 0
v_QCollected    dc 0
          ENDSEC
;=========================================================================
;Equates
FFTsize         equ     256
ScaleUp         equ     4
QBUFOVERFLOW    equ     15
TIMEZONES       equ     32
;————————————————————————————————
          ORG y:
TempBuf         ds FFTsize              ;Temp storage for FFT output

Inv                                     ;Table for inverse of numbers
1..255
          include 'inverse.inc'   ;(fractional) (1/1, 1/2, 1/3....1/
255)
Inv_end                                 ;To be used instead of short
division
QuantumBufLen   equ 10*10000*19/128     ;10 sec buffer @19 bands and
12.8msec frames
QuantumBuf      ds QuantumBufLen        ;(about 1484 words/sec)
          org y:$2900
OutBuf          ds 19*TIMEZONES         ;Final output result storage
;————————————————————————————————
          PAGE
;————————————————————————————————
          ORG p:
;————————————————————————————————
f_Init_PostProcess
          move #QuantumBuf,x0
          move x0,x:p_QuantumTop          ;Initialize pointer
          clr a
```

```
         move a,x:v_QCollected
f_Init_PostProcess_end
;————————————————————————————————
ProcessData     ;Takes data from the FFT, creates the 19 bands and
saves it

         PUSH m0
         PUSH m1
         PUSH m4
         PUSH m5
         move #0,m0              ;Reverse-carry
         move #FFTsize/2,n0      ;FFT was 256 points
         move #FFTBuffer,r0      ;Pointer to Real FFT data
         move m0,m4              ;Pointer to Imag FFT data
         move n0,n4              ;Same as real data pointer
         move r0,r4              ;Same as real data pointer
         move #TempBuf,r5        ;Pointer to output buffer
         move #$ffffff,m5        ;Linear arithmetic for output
                  ;move #FFTsize,n5        ;for debug, copy separate
data

                  ;move #1.,a             ;Trigger scope
                  ;jsr f_Put_Sample
                  ;jsr f_Put_Sample
                  ;jsr f_Put_Sample
                  ;jsr f_Put_Sample
                  ;jsr f_Put_Sample
                  ;move #0.5,a
                  ;jsr f_Put_Sample
                  ;clr a
                  ;jsr f_Put_Sample


         do #FFTsize,_CopyBuffer ;copy all bytes (change this to copy
                        ;only half of them, later)
         move x:(r0)+n0,x0 y:(r4)+n4,y0  ;Get real and imaginary part
         mpy x0,x0,a                     ;real^2
         mac y0,y0,a                     ;plus imag^2 makes magnitude

         asl a                   ;Magnify to keep some useful bits
         asl a
         asl a
         asl a
         asl a
         asl a
         asl a
         asl a
         move a,y:(r5)+          ;Store magnitude in TempBuf
                  ;jsr f_Put_Sample       ;Show at scope
_CopyBuffer
CreateBands     move #$ffff,m0
         move m0,m1
         move #StartOfBands,r0   ;Point to tables
         move #19,n0             ;Offset to the second table
         move #$ffff,m4
         move x:p_QuantumTop,r4  ;Point to the next available
position
                        ;in QuantumBuf
```

```
          move #$ffff,m2
          move #Inv,r2              ;Pointer to 1/x table
;.....................................................................
          do #19,_Average           ;Repeat for each band:
          move #TempBuf,r1          ;Point to magnitude data
          move p:(r0),n1            ;Get band start offset from first
table
          move p:(r0+n0),n2         ;Get band points from second table
          lua (r1)+n1,r1            ;Create pointer into magnitude data
                        ;(r1=Tempbuf+StartOfBand)
          clr a (r0)+               ;Clear a, adjust r0 for next pass.
          move #Inv,r2              ;Pointer to 1/x table (assigned here
due
                        ;to pipeline delays)
;.............................
          do n2,_EndBand            ;For all points in band:
          move y:(r1)+,y0           ;Get magnitude point
          add y0,a                  ;and add it in acc.
_EndBand
;.............................
          lua (r2)+n2,r2            ;Point to divisor (number of points
averaged)
          move a,x0
          move y:(r2),y0            ;Get sum and divisor
          mpyr x0,y0,a              ;Now divide to find average and
          move a,y:(r4)+            ;store band in output buffer
(QuantumBuf)
_Average
;.....................................................................
          move r4,x:p_QuantumTop    ;When all bands done, save pointer
for
                        ;next pass (next fft output).
          move #>1,x0
          move x:v_QCollected,a     ;Increase collected quantums counter
          add x0,a
          move a,x:v_QCollected
ChkOverFlow     move #>(QuantumBuf+QuantumBufLen),a
          move r4,x0
          cmp x0,a                  ;Check if r4 is out of the buffer.
          jgt _Ok                   ;If not, ok.
          move #QBUFOVERFLOW,n7     ;Else, notify system
          move #QuantumBuf,x0
          move x0,x:p_QuantumTop    ;and reset pointer.
_Ok
ProcessData_ret
          POP m5
          POP m4
          POP m1
          POP m0
          rts
ProcessData_end
;_____________________________________________
FinalProcess
          move x:p_QuantumTop,r4
          move #0,y0
          move #$ffff,m4
          do #(19*32),_filldummy
          move y0,y:(r4)+
```

```
_filldummy
          move x:v_QCollected,a   ;Get number of total frames
          move #>5,x0
          sub x0,a       ;Subtract the last 5 quantums (silence)
          rep #5
          lsr a1                  ;Divide by numberof timezones,
                         ;with carry
          move #0,a0
          tst a
          jgt _DontRoundUp        ;if numberof quantum<32 then
_RoundUp        move #>$1,a1             ;set one quantum per zone

_DontRoundUp
          move #$ffff,m4
          move #19,n4             ;Offset to the next row
          move #$ffff,m1
          move #QuantumBuf,r1     ;R1 is the base pointer for each
zone
          move #$ffff,m2          ;pointer to output buffer
          move #OutBuf,r2
          move #$ffff,m5          ;Used in Inv table
;.....................................................................
          ;Calculate step value for each zone
          move a,x1               ;Get number of quantums to average.
          move #>19,x0
          mpy x0,x1,a             ;Get number of elements in each Zone
          asr a                   ;Compensate for integer operands
          move a0,n1              ;Store at offset for zone base
pointer
;.....................................................................
_FindDivide
          move #Inv,r5            ;Get index into divisor table
          move x1,n5              ;Get offset into divisor table
          nop                     ;to calculate later, mean value of
quan.
          lua (r5)+n5,r5          ;Get 1/x1
          nop
          move y:(r5),y1          ;and put it in y1
;.....................................................................
          do #TIMEZONES,_AllZones        ;Repeat for all timezones
;.....................................................................
          do #19,_DoAllBandsInZone
          move r1,r4              ;Point to the first element of
                         ;the band to average.
;.....................................
          clr a   (r1)+           ;Clear acc

          do x1,_AddBand          ;and add to make one band of one
zone
          move y:(r4)+n4,y0       ;Get element, move to next position
          add y0,a                ;add to accumulator
_AddBand
          move a,x0               ;Move result in x0
          mpyr x0,y1,a            ;divide to find the average
;.....................................
          move a,y:(r2)+          ;Store in outbuf
;.....................................................................
                         ;and this zone is done.
```

```
_DoAllBandsInZone
         move (r1)+n1            ;Move to the next zone
_AllZones
;.....................................................................
FinalProcess_ret
         rts
FinalProcess_end
;———————————————————————————————————————
ReportData
         move #>16,x0            ;Get frames captured and
         move x0,a
         jsr f_Tx_24             ;send them to the host pc
         move #$ffff,m4
         move #OutBuf,r4         ;Now prepair to send frames
         do x0,_SendData         ;Repeat for all frames
         jsr f_Rx                ;Wait until frame is asked for
         move #'>',x1            ;by the host.
         cmp x1,a
         jeq _GoOn               ;If wrong char RX'ed,
         enddo                   ;abort transfer
                        ;Else, send all 19 words in the frame
_GoOn           do #19,_SendFrame       ;
         move y:(r0)+,a
         jsr f_Tx_24             ;to the serial port
_SendFrame
         nop                     ;DO loop limitation
_SendData                                ;and loop to the next frame
ReportData_ret
         rts
ReportData_end
;———————————————————————————————————————
;ReportData
;               move x:v_QCollected,x0  ;Get frames captured and
;               move x0,a
;               jsr f_Tx_24             ;send them to the host pc
;
;               move #$ffff,m4
;               move #QuantumBuf,r4     ;Now prepair to send frames
;
;               do x0,_SendData         ;Repeat for all frames
;               jsr f_Rx                ;Wait until frame is asked for
;               move #'>',x1            ;by the host.
;               cmp x1,a
;               jeq _GoOn               ;If wrong char RX'ed,
;               enddo                   ;abort transfer
;                                       ;Else, send all 19 words in
the frame
;_GoOn          do #19,_SendFrame       ;
;               move y:(r0)+,a
;               jsr f_Tx_24             ;to the serial port
;_SendFrame
;               nop     ;DO loop limitation
;_SendData                               ;and loop to the next frame
;
;
;ReportData_ret
;               rts
;ReportData_end
```

```
;-------------------------------------------------
;Table for the start of each frequency band.
;Number indicates starting point# of output FFT. Fs is 10KHz
StartOfBands    dc
5,8,11,14,17,20,24,28,31,35,39,44,49,54,59,67,73,81,90
;-------------------------------------------------
;Number of FFT points to average for each band
AveragePoints   dc 4,4,4,4,4,4,5,4,5,5,5,6,6,6,6,6,9,8,13
;-------------------------------------------------
          ENDSEC ;ProcData
;=============================================================================
     END
```

rs232int.asm: Επικοινωνίες μέσω της RS232

```
; Part of the IO functions
; Description   : RS232 buffered interface
; Code Number   : SRC-FRES0008-ILX
; Filename      : RS232INT.ASM
; Name          : RS232INT
; Original Date : 02/10/95
; Current  Date : 29/07/96
; Revision      : 1.0
; Platform      : EVM56001
; Author        : Alexopoulos Ilias
; Approved by   : Alexopoulos Ilias
;                 Diamantis Argiris

         page 132,45
         OPT DEX,INTR
;***************************************************************************
         SECTION RS232INT
         include 'ioregs.asm'
;***************************************************************************

         XDEF f_Init_RS232,f_Tx,f_Rx
         XDEF i_lscirx,i_lscitx,i_lscirxe
         XDEF p_ser_status
         XREF v_Leds
;_____________________________________
c_stxmodulo     EQU 128                         ;STX modulo for buffer
c_srxmodulo     EQU 16                          ;SRX modulo for buffer

;===========================================================================
         SECTION RamVars LOCAL
         org x:
p_ser_status:   dc  0                   ;Serial Interrupt Flag 1
p_stxint:       dc buf_stx              ;STX Interrupt Pointer
p_stxapp:       dc buf_stx              ;STX Application Pointer
p_srxint:       dc buf_srx              ;SRX Interrupt Pointer
p_srxapp:       dc buf_srx              ;SRX Application Pointer
         ENDSEC
;===========================================================================
;_____________________________________
;library data
         org y:
buf_stx:        dsm c_stxmodulo         ;Serial Transmition buffer
buf_srx:        dsm c_srxmodulo         ;Serial Reception buffer


     DEFINE  BAUD9600        '$102A'
     DEFINE  BAUD2400        '$10AF'
         PAGE
;_____________________________________
; Software Calls to Library
; Plus Sequence of Operations
;
; Function              Parameters              Description
;
; Init_RS232            None                    Initalise RS232
```

```
library
; TX                    A                       Transmit a Byte
; RX                    A                       Receive a Byte
;_____________________________________

;=========================================================================

        org p:
;_____________________________________
f_Init_RS232:

        bclr #SCI_DATA_OUT,x:p_ser_status ;Clear DataOut bit on our
Serial Flag
        bclr #SCI_DATA_IN,x:p_ser_status  ;Clear DataIn bit on our
Serial Flag
        movep #$0B24,SCR        ;1 start 8 data 1 stop Even parity,
no tx intrupt
        movep #BAUD9600,SCCR    ;9600
        movep PCC,x0
        move #>$03,a1           ;Enable TXD,RXD pins on DSP
        or x0,a1
        movep a1,PCC

        bset #15,IPR            ;Enable and set SCI interrupt
        bset #14,IPR            ;at highest priority
        rts
;_____________________________________
f_Tx:
        PUSH    r6                      ;Save Registers
        PUSH    m6

        move x:p_stxapp,r6              ;Application buffer pointer
        move #(c_stxmodulo-1),m6        ;and modulo loading
        nop                             ;Pipeline delay
        move a,y:(r6)+                  ;Transfer data
        move r6,x:p_stxapp              ;save next data address
        bset #SCI_DATA_OUT,x:p_ser_status ;TX busy now

        ;Enable Tx interrupts
        ;Because it may be disabled from the interrupt handler
        ;in case of its idle state

        bset #SCITIE,SCR
        bset #11,x:v_Leds               ;Show at leds
        POP     m6                      ;Restore regs
        POP     r6
        rts
;_____________________________________
f_Rx:
        PUSH    r6
        PUSH    m6
        PUSH    x0
        PUSH b0
        PUSH    b1
        PUSH b2
_rxloop:        jclr #SCI_DATA_IN,x:p_ser_status,_rxloop
        bclr #SCI_DATA_IN,x:p_ser_status        ;Rx Idle initial
condition
```

```
         bclr #NewData,x:p_ser_status            ;Clear flag if
nobody else has
         move #(c_srxmodulo-1),m6                ;Receive buffer
modulo
         move x:p_srxapp,r6                      ;compare interrupt
buf. pointer
         move x:p_srxint,x0
         move y:(r6)+,a                          ;receive char from
buffer
         move r6,x:p_srxapp
         move r6,b
         cmp x0,b                                ; if not equal then
RX_BUSY
         jeq _rxdend                             ; else RX_IDLE no
change from us
         bset #SCI_DATA_IN,x:p_ser_status        ; RX_IDLE could
change from the INTR
         jmp _f_Rx_ret                           ; In the mid-time
                                ;no further transaction

_rxdend:        bclr #10,x:v_Leds                        ;Clear Rx led
_f_Rx_ret
         POP b2
         POP     b1
         POP b0
         POP     x0
         POP     m6
         POP     r6

         rts
;_____________________________________
         PAGE
; Interrupt Servicing Routines
;
;*********************************************
;*      SCI INTERFACE   /RS232                *
;*********************************************
;=============================================================================
         org p:
;_____________________________________
i_lscitx:
         bset #8,PBD             ;Set debug pin

         ;if TX idle then get out
         jclr #SCI_DATA_OUT,x:p_ser_status,_txd_int_end

         PUSH    r6                      ;Save Registers
         PUSH    m6                      ;to save your life
         PUSH    a2
         PUSH a1
         PUSH a0
         PUSH    x0

         move #(c_stxmodulo-1),m6        ;Transmiter buffer modulo
         move x:p_stxint,r6              ;Initialize buffer pointer
         nop                             ;Pipeline delay
         move y:(r6)+,a1                 ;Read from buffer
         movep a1,STRXH                  ;Transmit
```

```
         move r6,x:p_stxint              ;Store New Interrupt
buf.pointer

         move r6,x0                      ;Compare with application
buf. pointer
         move x:p_stxapp,a
         cmp x0,a                        ;If not equal then TX_BUSY
         jneq _txdend

         bclr #SCI_DATA_OUT,x:p_ser_status ;No further transaction
         bclr #SCITIE,SCR                ;Disable TX interrupt
         bclr #11,x:v_Leds               ;Clear Tx led

_txdend:

         POP x0                          ;Restore Registers
         POP a0
         POP a1
         POP a2
         POP m6
         POP r6
_txd_int_end:
         bclr #8,PBD
         rti
;————————————————————————————
i_lscirx:
         PUSH r6                         ;Save registers
         PUSH m6                         ;and your life!
         PUSH a1
         move x:p_srxint,r6              ;load pointer, modulo etc.
         move #(c_srxmodulo-1),m6
         bset #SCI_DATA_IN,x:p_ser_status      ;RX not IDLE
         bset #NewData,x:p_ser_status    ;Set general purpose flag
         movep STRXH,a1                  ;Get data from port
         move a1,y:(r6)+                 ;Write data in buffer
         move r6,x:p_srxint              ;Save next available pointer
                         ;location
         bset #10,x:v_Leds               ;Set Rx led
         POP a1                          ;Restore used registers
         POP m6
         POP r6
         rti
;————————————————————————————
; Receive with Error
; Set accordingly the Error Flag
; Jetison STRXH value
i_lscirxe:

         jset #b_SSR_OVR,x0,_er_overrun
         jset #b_SSR_PE,x0,_er_parity
         jset #b_SSR_FE,x0,_er_framing
         move #SCI_RX_UKNOWN_ERROR,n7
         rti

_er_overrun
         move #SCI_RX_OVR,n7
         rti
```

```
_er_parity
          move #SCI_RX_PE,n7
          rti

_er_framing
          move #SCI_RX_FE,n7
          rti


            ENDSEC                    ;End of RS232 ISRs
;*************************************************************************

          END
```

Αρχεία Matlab

Στην ενότητα αυτή παρουσιάζονται μερικά απο τα αρχεία εντολών του Matlab που χρησιμοποιήσαμε στην ανάλυση των αποτελεσμάτων, αλλά και στην γρήγορη υλοποίηση των αλγορίθμων αναγνώρισης που μελετήσαμε. Υπάρχουν οι υλοποιήσεις των δύο παραλλαγών του αλγορίθμου αναγνώρισης που περιγράφονται στο 4ο κεφάλαιο, και τα αρχεία ανάλυσης αποτελεσμάτων του τελικού αλγορίθμου.

lsig.m:Φορτώνει και απεικονίζει ένα δείγμα σάν πίνακα του matlab

```
%Loads and displays a signal (8-bit signed)
%    Usage: [sig,len]=lsig(fname)
%
function [sig,len]=lsig(fname);
file=fopen(fname);
[sig,len]=fread(file,inf,'schar');
fclose(file);
%win=figure;
plot(sig);
title(fname);
```

spg.m: Παράγει το φασματογράφημα ενός σήματος

```
% spectorgram of data
% rdat=spg(samp);
function rdat=spg(samp);
dat=specgram(samp);
rdat=abs(dat);
clear dat;
figure;
colormap(cool);
surf(rdat);
view(90,90);
```

oldmeth.m:Υλοποιεί τον παλαιό αλγόριθμο αναγνώρισης

```
%Impliments the old pattern extraction alorithm
%    usage: [result]=oldmeth(fname)
%    where fname: filename of sample
function [result]=oldmeth(fname)
disp('Loading signal...')
[samples,len]=lsig(fname);            %Load & display signal
len                          %Show signal length
timezone=floor(len/16)                %Find timezone length
if timezone<=512                 %Determine length of FFT
     fftsize=512;
elseif timezone<=1024
     fftsize=1024;
elseif timezone<=2048
     fftsize=2048;
elseif timezone<=4096
     fftsize=4096;
elseif timezone<=8192
     fftsize=8192;
end
fftsize
disp('Breaking up ...')
for i=1:16
temp(i,:)=samples(i*timezone-timezone+1: i*timezone)';  %Break signal
in zones
end                               %with timezones in rows
disp('Windowing segments...')             %Apply windowing function
window=hanning(timezone)';
for i=1:16
```

```
temp(i,:)= temp(i,:) .* window;
end
disp('Calculating FFTs...')          %Calculate the 16 FFTs
for i=1:16
     spectrum(i,:)= fft(temp(i,:),fftsize);
end
clear samples window temp;           %Free up space
spectrum=abs(spectrum);              %Get the power spectrum
temp=spectrum(:,1:fftsize/2);             %Isolate the useful freqs
clear spectrum
bands=[           %Frequency bands in Hz. Each pair is (start,stop)
     180,300;
     300,420;
     420,540;
     540,660;
     660,780;
     765,915;
     925,1075;
     1075,1225;
     1225,1375;
     1375,1525;
     1525,1675;
     1700,1900;
     1900,2100;
     2100,2300;
     2300,2500;
     2600,2800;
     2850,3150;
     3150,3450;
     3500,4000
     ];
SamplesPerHz=fftsize/10000       %How many Hz/sample
freq=round(bands * SamplesPerHz);     %Transform into frequency bands
for i=1:16                   %Extract average energy in each band
for j=1:19
result(i,j)=mean(temp(i,freq(j,1):freq(j,2)));
end
end
largest=max(max(result));
result=result/largest;           %Normalize to 1.0
figure;                      %Plot result
colormap(cool);
surf(result);
view(75,40);
ylabel('Time');
xlabel('Frequency');
zlabel('Energy');
title(fname);
whos
```

newmeth.m: Υλοποιεί τον νέο αλγόριθμο αναγνώρισης

```
%Impliments the new pattern extraction alorithm
%    usage: [result]=newmeth(fname,filtername)
%    where     fname: filename of sample
%    Loads filter data from filtdata.mat
%function [result]=newmeth(fname);
fname='stoxos3.smp'
```

```
disp('Loading signal...')
[sig,len]=lsig(fname);           %Load & display signal
samp=decimate(sig,4);
len=len/4
clear sig
%len=2048;
%for i=1:len
%samp(i)=sin(2*3.1415*i*i);
%end
%plot(samp);
disp('Loading Filters...')
load 'filtdata.mat'
                          %Show signal length
%for loop1=1:len
%samples(loop1)=samp(loop1,1);
%end
disp('Press Any key');
pause
time_div=round(len/16)-1;
disp('Filtering...');
time_div
more off
for freqzone=1:19
     freqzone
     y=filter(be(freqzone,:),ae(freqzone,:),samp);

     for timezone=1:16
     temp=0;
          for k=1:(time_div-1)
          temp=temp+y((timezone-1)*time_div+k);
          end

     result(freqzone,timezone)=temp/time_div;
     end
end
result=abs(result);
largest=max(max(result));
result=result/largest;           %Normalize to 1.0
figure;                     %Plot result
colormap(cool);
for i=1:16
time(i)=i;
end;
for i=1:19
freq(i)=i;
end;
surf(freq,time,result');
view(90,90);
ylabel('Time');
xlabel('Frequency');
zlabel('Energy');
title(fname);
disp('Press Any key');
pause
view(75,45);
more on
```

normal.m: Κανονικοποιεί τα αποτελέσματα που παράγει το DSP.

```
function s=normal(t)
m=max(max(t));
s=t./m;
```

veccomp.m: Υλοποιεί τον αλγόριθμο σύγκρισης

```
% function out=veccomp(in,sp1,sp2,sp3,sp4);
% returns the magnitude of the difference of "in" with each
% one of the sp1,sp2 and sp3  word vectors
function out=veccomp(in,sp1,sp2,sp3,sp4);
out=[0 0 0 0];
for time=1:32
for freq=1:19
out(1)=out(1)+sqrt((1/freq)*abs(sp1(time,freq)^2-in(time,freq)^2));
out(2)=out(2)+sqrt((1/freq)*abs(sp2(time,freq)^2-in(time,freq)^2));
out(3)=out(3)+sqrt((1/freq)*abs(sp3(time,freq)^2-in(time,freq)^2));
out(4)=out(4)+sqrt((1/freq)*abs(sp4(time,freq)^2-in(time,freq)^2));
end
end
```

veccorr.m: Συσχετίζει δύο φασματογράμματα

```
% correlation of vectors
% function out=veccomp(in,sp1,sp2,sp3,sp4);
% returns the magnitude of the difference of "in" with each
% one of the sp1,sp2 and sp3  word vectors
function out=veccorr(in,sp1,sp2,sp3,sp4);
out=[0 0 0 0];
co1=xcorr2(in,sp1);
co2=xcorr2(in,sp2);
co3=xcorr2(in,sp3);
co4=xcorr2(in,sp4);
for time=1:63
for freq=1:37
out(1)=out(1)+sqrt(abs(co1(time,freq)^2));
out(2)=out(2)+sqrt(abs(co2(time,freq)^2));
out(3)=out(3)+sqrt(abs(co3(time,freq)^2));
out(4)=out(4)+sqrt(abs(co4(time,freq)^2));
end
end
```

Βιβλιογραφία

Signal Processing of Speech
F. J. Owens
MacMillan "New Electronics"
ISBN 0-333-51922-1
Speech Processing
Ed. by Chris Rowden
McGraw-Hill
ISBN 0-07-707324-X

Digital Signal Analysis
Samuel D. Stearns, Don R.Hush
Prentice-Hall
ISBN 0-13-211772-X

Digital Signal Processing
Allan V. Oppenheim, Ronald W.Schafer
Prentice-Hall
ISBN 0-13-214107-8

Interfacing
Stephen E. Derenzo
Prentice-Hall
ISBN 0-13-468380-3

A Practical Approach to Digital Sampling
Terry Fryer
HLP
ISBN 0-88188-925-3

High Speed Digital Design
Howard W.Johnson, Martin Graham
Prentice Hall
ISBN 0-13-395724-1

The Art of Electronics
Paul Horowitz, Winfield Hill
Cambridge Univercity Press
ISBN 0-521-37095-7

Digital Logic Circuit Analysis And Design
Victor P.Nelson, H.Troy Nagle, Bill D.Carroll, J.David Irwin
Prentice Hall
ISBN 0-13-463894-8

DSP56000/DSP56001
Digital Signal Processor User's Manual
Motorola
DSP56000UM/AD

DSP56000/1 Technical Data
Motorola
DSP56001/D

DSP56000ADS
Application Development System Reference Manual
Motorola

DSP56001
Technical Summary (databook)
Motorola

The Programmable Logic DataBook
Xilinx,1994

Intergrated Device Technology Databook
Static RAM (1991)
DBK-SRAM-00021