ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Συστημάτων Μετάδοσης Πληροφορίας και Τεχνολογίας Υλικών
Εργαστήριο Κινητών Επικοινωνιών

Υλοποίηση Αναλυτή Φάσματος με τεχνικές Software Radio

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

-του-

Γεώργιος Ι. Ανδρίτσος **Georgios I. Andritsos**

Επιβλέπων: Φ. Κωνσταντίνου
Καθηγητής Εθνικού Μετσοβίου Πολυτεχνείου

Αθήνα, Νοέμβριος 2007

Περιεχόμενα

0.1 Πρόλογος

Η διπλωματική αυτή εργασία απευθύνεται σε κοινό προπτυχιακού επιπέδου ηλεκτρολόγων μηχανικών και μηχανικών υπολογιστών. Παρόλα αυτά για την πλήρη κατανόηση και μελέτη αυτής της διπλωματικής χρειάζεται εμβάθυνση στη θεωρία σημάτων (Digital Signal Processing), αρχών τηλεπικοινωνιών, στον προγραμματισμό με C και σε αρχιτεκτονική υπολογιστών και συγκεκριμένα επεξεργαστών DSP.

0.2 Περίληψη

Το Software Defined Radio (SDR) είναι ένας αποτελεσματικότερος τρόπος ασύρματης επικοινωνίας. Η διαφορετικότητά του βρίσκεται στο γεγονός ότι διαδικασίες που παραδοσιακά είναι υπεύθυνο το hardware γίνονται από software, με αποτέλεσμα να υπάρχει μεγαλύτερη ευελιξία, αφού το software εύκολα αλλάζει και παραμετροποιείται. Δηλαδή, βασικός σκοπός είναι το ηλεκτρομαγνητικό σήμα να γίνει ένα ψηφιακό σήμα και στη συνέχεια να το επεξεργαστούμε με έναν κατάλληλο επεξεργαστή. Καταλληλότεροι επεξεργαστές για αυτή τη λειτουργία είναι οι επεξεργαστές σήματος (DSP). Είναι ειδικευμένοι επεξεργαστές που εκτελούν κάποιες συνηθισμένες μαθηματικές πράξεις πολύ γρήγορα εκμεταλλευόμενοι παράλληλες αρχιτεκτονικές που έχουν στο εσωτερικό τους. Για την υλοποίηση Αναλυτή Φάσματος στον υπολογιστή με τεχνικές software radio χρειάζεται ο κατάλληλος εξοπλισμός καθώς και κατάλληλος προγραμματισμός του εξοπλισμού αυτού. Ο απαραίτητος εξοπλισμός είναι μια κεραία, ένας γρήγορος A/D/A μετατροπέας, ένας επεξεργαστής και μια οθόνη. Μια κεραία με ευρύ φάσμα τροφοδοτεί τον A/D μετατροπέα που σερβίρει τα δεδομένα στον επεξεργαστή. Το πρόγραμμα το οποίο τρέχει στον επεξεργαστή πρέπει να είναι κατάλληλο και να υλοποιεί τέτοιες τεχνικές που θα επιτρέπουν την real-time παρουσίαση του φάσματος του λαμβανόμενου σήματος στην οθόνη.

ασύρματη επικοινωνία software radio SDR μετατροπέας επεξεργαστής σήματος DSP αναλυτής φάσματος

0.3 Summary

Software Defined Radio is a more efficient way of wireless communication. The advantage of SDR is that procedures that traditionally were dealt by the hardware now are dealt by software. This has as a result of more flexibility in terms of equipment and infrastructure. The basic goal is the electromagnetic signal to become a digital signal that can be easily processed on an appropriate processor. The most appropriate processors for these tasks are Digital Signal Processors (DSP). They are specifically designed to execute common mathematical procedures very fast taking advantage of the parallelism they have in their architecture. For the creation of the Spectrum Analyzer for a PC with software radio technics, special hardware and software is required. The hardware is a broadband antenna, a fast A/D/A converter, a precessing unit and a monitor. The broadband antenna is connected to the A/D converter that serves the data to the processor. The program that runs on that processor has to make use of mathematical technics that will make real-time monitoring of the frequency spectrum of the signal from a monitor possible.

wireless communication software radio SDR converter signal processor DSP spectrum analyzer

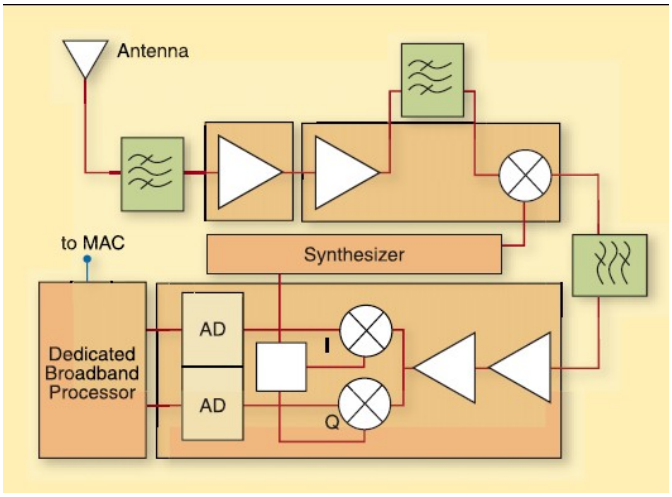
Κεφάλαιο 1

Εισαγωγή στο Software Defined Radio (SDR)

1.1 Τι είναι το SDR

Το Software Defined Radio (SDR) είναι ένας αποτελεσματικότερος τρόπος ασύρματης επικοινωνίας. Η διαφορετικότητά του βρίσκεται στο γεγονός ότι διαδικασίες που παραδοσιακά είναι υπεύθυνο το υλικό (hardware) γίνονται από λογισμικό (software), με αποτέλεσμα να υπάρχει μεγαλύτερη ευελιξία, αφού το software εύκολα αλλάζει και παραμετροποιείται. Το *ιδανικό* Software Defined Radio προσφέρει την μέγιστη ευελιξία και παραμετροποιησιμότητα που μπορεί να επιτευχθεί.

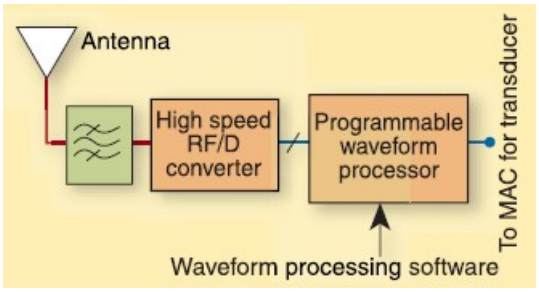
Η τεχνολογία των τηλεπικοινωνιών, η οποία φαντάζει παραδοσιακή, πρόκειται να αλλάξει άρδην στα προσεχή έτη. Οι παραδοσιακοί δέκτες φτιάχνονται με αναλογικά στοιχεία (πυκνωτές, πηνία και αντιστάσεις) όπου για να επιτευχθεί π.χ. ο συντονισμός με ένα φέρον θα έπρεπε να αλλάξεις με κάποιον τρόπο τα αναλογικά αυτά στοιχεία. Για πολλά χρόνια, μέχρι και σήμερα, η αναλογική αυτή τεχνολογία, έχοντας προφανώς αναπτυχθεί σφόδρα, μας δίνει ικανοποιητικές λύσεις για τις ανάγκες μας. Αλλά αυτό πρόκειται να αλλάξει στο προσεχές μέλλον. Η FCC (Federal Communications Commission), το SDR Forum και η βιομηχανία τηλεπικοινωνιών είναι ενωμένα στο κυνήγι του ιδανικού Software-defined radio (SDR). Σύμφωνα με την FCC, σε ένα σύστημα "software-defined radio", διαδικασίες οι οποίες αποκλειστικά εκτελούνταν σε υλικό, όπως π.χ. η δημιουργία του σήματος που επρόκειτο να αποσταλεί και ο συντονισμός και ανίχνευση του λαμβανόμενου σήματος, πραγματοποιούνται από λογισμικό που εκτελείται σε επεξεργαστές ειδικευμένους σε επεξεργασία σήματος. Τέτοιοι επεξεργαστές μπορεί να είναι ένας DSP (Digital Signal Processor) της TI πάνω σε κάποια βοηθητική πλακέτα ή ακόμα και μία κάρτα



Σχήματα 1.1: Παραδοσιακή Αρχιτεκτονική

γραφικών τελευταίας γενιάς της οποίας η λειτουργία περιορίζεται στο fade-in and fade-out των παραθύρων του εξερευνητή σε κάποιο λειτουργικό σύστημα.

Το SDR Forum ορίζει μια συσκευή σαν SDR-συσκευή όταν αυτή λειτουργεί ανεξαρτήτως συχνότητας φέροντος και μπορεί να λειτουργήσει σε μια πληθώρα από περιβάλλοντα και πρωτόκολλα τηλεπικοινωνιών. Πηγαίνοντας ένα βήμα παραπέρα, το SDR Forum ορίζει το ιδανικό SDR ως εκείνο που έχει πομποδέκτες που άνω-σχηματίζουν (upconversion) και κάτω-σχηματίζουν (downconversion) το σήμα από σήμα βάσης (baseband) σε σήμα φέροντος (carrier), αποκλειστικά ψηφιακά, περιορίζοντας το αναλογικό κομμάτι σε έναν ενισχυτή ισχύος στην αποστολή, σε έναν LNA στην λήψη και μικρό η καθόλου αναλογικό φιλτράρισμα. Σε αυτήν την ιδανική περίπτωση SDR, είναι δυνατόν να αναβαθμιστούν η να αλλάξουν τελείως τα χαρακτηριστικά ενός τηλεπικοινωνιακού συστήματος, απλά από την αναβάθμιση του λογισμικού. Αυτό η ιδανική περίπτωση που αναφέρει το SDR Forum, μέχρι πρότινος, φάνταζε αδύνατη λόγω της υποανάπτυκτης τεχνολογίας για την άμεση μετατροπή του υψίσυχνου σήματος RF σε ψηφιακό σήμα. Τώρα νέα ολοκληρωμένα κυκλώματα που υλοποιούν τέτοιες διαδικασίες προσφέρουν μεγαλύτερες ταχύτητες με χαμηλή κατανάλωση ισχύος. Το καλύτερο που έχει να αναδείξει η τεχνολογία σε ολοκληρωμένο κύκλωμα για αυτές τις λειτουργίες είναι η μετατροπή συχνοτήτων φέροντος της τάξης των 5GHz άμεσα σε ψηφιακό σήμα. Στο παρόν κεφάλαιο θα περιγράψουμε κάποια προϊόντα που είναι διαθέσιμα σήμερα, για να κάνουμε μια πιο χειροπιαστή εισαγωγή στο Software-Radio και να καταλάβουμε τι είναι εφικτό σήμερα, τι έπεται να γίνει και ποιες είναι οι δυνατότητες αυτής της τεχνολογίας.



Σχήματα 1.2: SDR Αρχιτεκτονική

1.2 Ρόλος του SDR

Το SDR παίζει πολύ σπουδαίο ρόλο για το μέλλον και την αξιοπιστία των τηλεπικοινωνιών που συμπεριλαμβάνουν διασυνεργασία (interoperability) μεταξύ των συστημάτων. Όπως είναι γνωστό, κάποιες υπηρεσίες όπως π.χ. η πυροσβεστική και η αστυνομία δεν έχουν επικοινωνία μεταξύ τους επειδή οι συχνότητες τις οποίες χρησιμοποιούν είναι διαφορετικές. Με το ιδανικό SDR, μπορεί να χρησιμοποιηθεί λογισμικό σαν μεταφραστής μεταξύ τελείως διαφορετικών όχι μόνο συχνοτήτων αλλά και τρόπου διαμόρφωσης. Για παράδειγμα, τηλεπικοινωνία με φέρον στα 900MHz θα μπορεί να μιλήσει με τηλεπικοινωνία στα 2.4GHz, GSM κινητά θα επικοινωνούν με CDMA τηλέφωνα.. Σημαντική εφαρμογή βρίσκει και ο στρατός όπου θα μπορεί να επικοινωνεί σε διάφορες συχνότητες. Ο στρατός ης Αμερικής (από όπου έχουμε και σημαντικές εφαρμογές του Software Radio) και συγκεκριμένα το JTRS (U.S. Military Joint Tactical Radio System) πρόγραμμα έθεσε σαν στόχο να υποστηρίξει 33 τεχνικές διαμόρφωσης και διαφορετικές συχνότητες φέροντος μέσα στο εύρος των 2MHz έως 55GHz με ένα frontend. Μέχρι τώρα, ο σχεδιασμός τηλεπικοινωνιακών συστημάτων περιοριζόταν μόνο σε αναλογικά frontends (RFEs) για upconversion και downconversion σε ενδιάμεση συχνότητα (IF Intermediate Frequency) που συνήθως είναι κάτω των 100MHz όπου Αναλογικοί-σε-Ψηφιακοί Μετατροπείς (ADC Analog-to-Digital Converters) μπορούσαν να ανταπεξέλθουν.

Ο δέκτης είναι το δυσκολότερο και πολυπλοκότερο σύστημα που αποτελεί πρόκληση για το SDR. Ο ιδανικός δέκτης πρέπει να είναι σε θέση να ανιχνεύει γρήγορα μεταβαλλόμενη πληροφορία από ασθενή RF σήματα πνιγμένα στη θάλασσα του θορύβου. Νέα RFEs χρησιμοποιούν εξελιγμένες τεχνικές (superheterodyne, direct-conversion, and hybrid techniques), αλλά δεν είναι μέσα στα πλαίσια που ορίζει το SDR Forum με το ιδανικό SDR. Αντιθέτως απλά διευρύνουν τη κλίμακα των επεξεργαστών που δουλεύουν στις συχνότητες βάσης (baseband) όπως τα σύγχρονα κινητά τηλέφωνα.

Για παράδειγμα, ένας GSM900 δέκτης με κεντρική συχνότητα λειτουργίας τα 947.5MHz με εύρος 25MHz έχει για ενδιάμεση συχνότητα τα 80MHz που είναι μέσα στα όρια των σύγχρονων συνηθισμένων υψηλής ανάλυσης ADCs. Αυτή η υψηλής συχνότητας επεξεργασία δημιουργεί ανεπιθύμητες εικόνες σε ανώτερες συχνότητες μεταξύ 1095MHz και 1220MHz. Βέβαια έχουμε τεχνικές απόρριψης αυτών των εικόνων με φίλτρα αλλά όχι με 110db που απαιτούν οι προδιαγραφές. Από την άλλη αν ανεβάσουμε την ενδιάμεση συχνότητα στα 500MHz θα δημιουργηθούν εικόνες 1GHz πάνω από τη συχνότητα του σήματος όπου εύκολα μπορούμε να κόψουμε, αλλά θα χρειαστούν πολλαπλές μετατροπές (downconversion steps) για να μπούμε στα όρια που δουλεύει ο ADC.

Σε επόμενο κεφάλαιο θα χρησιμοποιήσουμε έναν ADC/DAC που δουλεύει σε μέγιστη συχνότητα 105MHz. Οι περισσότεροι σχεδιαστές τηλεπικοινωνιακών συστημάτων χρησιμοποιούν μια ενδιάμεση συχνότητα της τάξης των 60-70MHz. Άλλες εταιρίες οι οποίες εξειδικεύονται στο να δημιουργούν γρήγορους ADC είναι: Linear Technology Corp., Maxim Integrated Products, Texas Instruments Inc. and TelASIC.

Κεφάλαιο 2

Επεξεργαστές Ψηφιακού Σήματος (DSP)

2.1 Επεξεργασία Ψηφιακού Σήματος (DSP) και DSP Συστήματα

Ένα *σύστημα επεξεργασίας ψηφιακού σήματος* (Digital Signal Processing System) είναι ένα σύστημα που όπως λέει και το όνομά του επεξεργάζεται ψηφιακό σήμα. Η επεξεργασία αυτή γίνεται μέσα από μαθηματικές διαδικασίες οι οποίες ορίζουν και το ψηφιακό σήμα. Τα ψηφιακά σήματα προσδιορίζονται από μια σειρά τιμών που αποκαλούνται *δείγματα*. Συνήθως τέτοιου είδους δείγματα λαμβάνουμε από φυσικά, αναλογικά σήματα μέσω συσκευών μετατροπής, όπως είναι ένα μικρόφωνο που ακολουθείται από έναν δειγματολήπτη (Analog-to-Digital Converter ADC). Μετά από μαθηματική επεξεργασία είναι δυνατόν τα ψηφιακά αυτά σήματα να μετατραπούν ξανά σε φυσικά μέσω των ψηφιοποιητών (Digital-to-Analog Converters DAC).

Σε μερικά συστήματα, η χρήση των DSP (Digital Signal Processors) είναι η καρδιά για την λειτουργία του συστήματος. Για παράδειγμα, μόντεμ (modems) και κινητά τηλέφωνα περιέχουν και εξαρτώνται εξ' ολοκλήρου από τη χρήση των DSP (Digital Signal Processor) και την τεχνολογία του DSP (Digital Signal Processing). Σε άλλα προϊόντα, η χρήση των DSP (Digital Signal Processor) δεν είναι τόσο ουσιώδης, αλλά χρησιμοποιούνται λόγω των πλεονεκτημάτων που έχουν όσον αφορά την επίδοση, των επιπλέον χαρακτηριστικών που προσφέρουν ή ακόμα και του χαμηλού κόστους. Για παράδειγμα, η βάση των νέων τεχνολογιών ασύρματης δικτύωσης βασίζεται σε DSP για λόγους κόστους και ευελιξίας.

2.2 Πλεονεκτήματα των Επεξεργαστών Ψηφιακού Σήματος

Η ψηφιακή επεξεργασία του σήματος έχει πολλά πλεονεκτήματα σε σχέση με την επεξεργασία στο αναλογικό επίπεδο. Το πιο σημαντικό από όλα είναι το γεγονός ότι κάποιες διεργασίες που στον αναλογικό κόσμο φαντάζουν αδύνατες ή εξαιρετικά κοστοβόρες, με τη χρήση DSP γίνονται με απλό τρόπο. Για παράδειγμα, τέτοιου είδους διεργασίες είναι η αναγνώριση και σύνθεση φωνής ή γρήγορα modem που περιλαμβάνουν αλγορίθμους διόρθωσης σφαλμάτων. Όλες αυτές οι διεργασίες απαιτούν συνδυασμό επεξεργασίας σήματος και έλεγχο (π.χ. για αποφάσεις που πρέπει να ληφθούν αναλόγως με τα εισερχόμενα δεδομένα) που είναι εξαιρετικά πολύπλοκες για να υλοποιηθούν αναλογικά.

Τα συστήματα DSP έχουν επιπλέον τα εξής πλεονεκτήματα σε σχέση με τα αναλογικά συστήματα:

- **Αναίσθητα στις περιβαντολλογικές μεταβολές.** Τα συστήματα DSP είναι πολύ λιγότερο ευαίσθητα από τα αναλογικά συστήματα λόγω της φύσης τους. Για παράδειγμα, τα αναλογικά κυκλώματα επηρεάζονται από εξωτερικές συνθήκες όπως η θερμοκρασία. Εν αντιθέσει, τα ψηφιακά συστήματα δεν εξαρτώνται από τις συνθήκες του περιβάλλοντος (αν βρίσκονται στο χιόνι ή στην έρημο), πράγμα που τα κάνει περισσότερο αξιόπιστα.

- **Ανεξάρτητα από τις προδιαγραφές των συσκευών τους.** Οι αναλογικές συσκευές από κατασκευής τους έχουν κάποιες τιμές ανοχής που παρεκκλίνουν από τις ονομαστικές. Για παράδειγμα, μια αντίσταση από κατασκευής είναι εγγυημένη ότι θα έχει ωμική αντίσταση όσο η ονομαστική της τιμή με παρέκκλιση 1%. Η συμπεριφορά του αναλογικού συστήματος λοιπόν εξαρτάται από αυτές τις προδιαγραφές, με αποτέλεσμα δύο ίδια συστήματα να μην έχουν ακριβώς τα ίδια αποτελέσματα. Εν αντιθέσει τα ψηφιακά συστήματα αν σχεδιαστούν και υλοποιηθούν σωστά και με τον ίδιο τρόπο θα έχουμε δύο ισοδύναμα συστήματα, όπου με ίδια είσοδο θα έχουμε την ίδια έξοδο.

Αυτά τα δύο πλεονεκτήματα λοιπόν συνδυασμένα μας δίνουν μια καλύτερη συμπεριφορά σε σύγκριση με τα αναλογικά συστήματα:

- **Προβλέψιμη και επαναλαμβανόμενη συμπεριφορά.** Εφόσον η έξοδος ενός DSP συστήματος δεν επηρεάζεται από εξωτερικούς παράγοντες η από τις συσκευές από τις οποίες αποτελείται, είναι δυνατόν να σχεδιασθούν και να υλοποιηθούν συστήματα που έχουν ακριβώς τις ίδιες συμπεριφορές και αποτελέσματα.

Τέλος, ορισμένα DSP συστήματα παρουσιάζουν τα εξής πλεονεκτήματα:

- **Επαναπρογραμματήσιμα.** Το πρόγραμμα που εκτελεί ένας DSP μπορεί εύκολα να αλλάξει ή να τροποποιηθεί ώστε να προσφέρει παραπάνω ή διαφορετικές λειτουργίες, τη στιγμή που σε περίπτωση αναλογικού συστήματος θα χρειαζόταν φυσική αλλαγή των συσκευών του συστήματος.

- **Μικρό μέγεθος.** Το μέγεθος των αναλογικών συσκευών ποικίλει σε σχέση με τις ονομαστικές του τιμές. Για παράδειγμα ένας πυκνωτής 100μF είναι μεγαλύτερος από έναν 10pF που χρησιμοποιούνται για τη σχεδίαση ενός αναλογικού φίλτρου. Στην περίπτωση του ψηφιακού συστήματος η διαφορά αυτή μεταφράζεται σε αλλαγή των δεδομένων του προγράμματος που εκτελεί ο επεξεργαστής, με αποτέλεσμα να μην επηρεάζεται το μέγεθος της συσκευής που συνήθως είναι μικρότερο και από τις δύο περιπτώσεις αναλογικής σχεδίασης στο παράδειγμά μας.

Τα παραπάνω πλεονεκτήματα σε συνδυασμό με το γεγονός ότι οι DSPs συνεχώς βελτιώνονται με την ανάπτυξη της τεχνολογίας των ολοκληρωμένων κυκλωμάτων (Integrated Circuits), αποτελούν μία πολύ καλή επιλογή για την σχεδίαση ενός ψηφιακού συστήματος.

2.3 Χαρακτηριστικά των DSP Συστημάτων

Όλα τα συστήματα DSP έχουν κάποια κοινά χαρακτηριστικά όπως, αλγόριθμοι, ρυθμός δειγματοληψίας, χρονισμός ρολογιού και αριθμητικοί τύποι.

2.3.1 Αλγόριθμοι

Τα συστήματα DSP συχνά κατηγοριοποιούνται σύμφωνα με τους αλγορίθμους τους οποίους υλοποιούν. Ο αλγόριθμος απαριθμεί τις διαδικασίες που πρέπει να εκτελεστούν αλλά όχι τον τρόπο με τον οποίο θα υλοποιηθούν. Υπάρχει επιλογή υλοποίησης σε λογισμικό (software) σε έναν μικροεπεξεργαστή ή DSP, ή μπορεί να υλοποιηθεί σε υλικό (hardware) σε ένα FPGA (Field-programmable gate array) ή ένα ASIC (Application-specific integrated circuit). Το ποια επιλογή θα γίνει για κάθε εφαρμογή εξαρτάται από τις ανάγκες για ταχύτητα και αριθμητική ακρίβεια. Στον παρακάτω πίνακα περιέχονται κάποιες τυπικές κατηγορίες αλγορίθμων που συχνά υλοποιούνται σε DSP και σε ποιές εφαρμογές βλέπουμε τέτοιες υλοποιήσεις.

Πίνακες 2.1: Συνήθεις αλγόριθμοι για DSP και κοινές εφαρμογές τους

Αλγόριθμος DSP	**Εφαρμογή σε σύστημα DSP**
Κωδικοποίηση και αποκωδικοποίηση φωνής	Κινητά τηλέφωνα, προσωπικά συστήματα τηλεπικοινωνιών, ασύρματα τηλέφωνα, προσωπικοί υπολογιστές πολυμέσων, ασφαλείς επικοινωνίες
Κρυπτογράφηση και αποκρυπτογράφηση φωνής	Κινητά τηλέφωνα, προσωπικά συστήματα τηλεπικοινωνιών, ασύρματα τηλέφωνα, ασφαλείς επικοινωνίες
Αναγνώριση και σύνθεση φωνής	Εξελιγμένα user interfaces, ρομποτική, προσωπικοί υπολογιστές πολυμέσων
Κωδικοποίηση και αποκωδικοποίηση υψηλής ποιότητας ήχου	Επαγγελματικά και ερασιτεχνικά ηχοσυστήματα και βίντεο, εκπομπή ψηφιακού ήχου, προσωπικοί υπολογιστές πολυμέσων
Αλγόριθμοι σε modems	Κινητά τηλέφωνα, προσωπικά συστήματα τηλεπικοινωνιών, ασύρματα τηλέφωνα, προσωπικοί υπολογιστές πολυμέσων, εκπομπή ψηφιακού ήχου, σηματοδοσία σε cable TV, ασύρματα δίκτυα, δρομολόγηση, ασφαλείς επικοινωνίες
Συμπίεση Θορύβου	Επαγγελματικά και ερασιτεχνικά ηχοσυστήματα, βιομηχανικές εφαρμογές, μουσική
Εξομάλυνση ήχου	Επαγγελματικά και ερασιτεχνικά ηχοσυστήματα, μουσική
Εξομοίωση ακουστικής χώρου	Επαγγελματικά και ερασιτεχνικά ηχοσυστήματα, μουσική
Σύνθεση Μουσικής	Επαγγελματικά και ερασιτεχνικά ηχοσυστήματα, μουσική, προσωπικοί υπολογιστές πολυμέσων
Τεχνητή Όραση	Ασφάλεια, Ιατρική, προσωπικοί υπολογιστές πολυμέσων, Εξελιγμένα user interfaces, ρομποτική, δρομολόγηση
Συμπίεση και αποσυμπίεση εικόνας	Ψηφιακή φωτογραφία, βίντεο, προσωπικοί υπολογιστές πολυμέσων, προσωπικά συστήματα τηλεπικοινωνιών

2.4 Χαρακτηριστικά των Επεξεργαστών Ψηφιακού Σήματος

Ο Επεξεργαστής Ψηφιακού Σήματος (Digital Signal Processor -DSP-) είναι ένας εξειδικευμένος επεξεργαστής, φτιαγμένος έτσι ώστε να μπορεί να κάνει τις διαδικασίες που χρειάζονται για την Επεξεργασία Ψηφιακού Σήματος (Digital Signal Processing -DSP-) τόσο γρήγορα ώστε να μπορεί να βγάζει αποτελέσματα και σε πραγματικό χρόνο (Real-Time). Τα χαρακτηριστικά που τον κάνουν που δίνουν στο DSP αυτήν την εξειδίκευση είναι τα εξής:

2.4.1 Γρήγορος πολλαπλασιασμός με συνάθροιση

Το πιο σημαντικό χαρακτηριστικό των επεξεργαστών ψηφιακού σήματος (DSP) είναι η δυνατότητα που έχουν να εκτελούν πολλαπλασιασμό και συνάθροιση (multiply-accumulate -MAC-) σε μια εντολή που χρειάζεται έναν κύκλο ρολογιού. Αυτή η δυνατότητα που έχουν είναι εξαιρετικά χρήσιμη σε αλγορίθμους που οι διαδικασίες που περιέχει είναι της φύσης του εσωτερικού γινομένου, όπως για παράδειγμα στα ψηφιακά φίλτρα, στον υπολογισμό της συσχέτισης ενός σήματος, στο υπολογισμό ενός μετασχηματισμού Fourier. Για να επιτευχθεί αυτό, οι DSP περιλαμβάνουν πολλαπλασιαστή και συναθροιστή στην κύρια μονάδα επεξεργασίας του (main processing unit) η οποία λέγεται μονοπάτι δεδομένων (data path) του επεξεργαστή. Επίσης, για να επιτραπούν πολλές εκτελέσεις πολλαπλασιασμού και συνάθροισης χωρίς το φόβο αριθμητικής υπερχείλισης, συνήθως οι DSP έχουν καταχωρητές συνάθροισης με επιπλέον bit για να χωρέσει χωρίς αριθμητική απώλεια το αποτέλεσμα της συνάθροισης.

2.4.2 Αρχιτεκτονική μνήμης πολλαπλής πρόσβασης

Ένα άλλο χαρακτηριστικό που έχουν οι περισσότεροι επεξεργαστές ψηφιακού σήματος (DSP) είναι η δυνατότητα που έχουν να κάνουν πάνω από μία προσβάσεις στη μνήμη σε έναν κύκλο ρολογιού. Αυτό δίνει τη δυνατότητα στον επεξεργαστή να αναγνώσει την επόμενη εντολή (instruction fetch) ενώ ταυτόχρονα φέρνει τις παραμέτρους της εντολής ή αποθηκεύει το αποτέλεσμα της προηγούμενης εντολής στη μνήμη. Απαιτείται μεγάλη ταχύτητα μεταξύ του επεξεργαστή και της μνήμης για να υπάρχει καλή απόδοση σε επαναληπτικές διαδικασίες που απαιτούν επεξεργασία δεδομένων με πολλές προσβάσεις στη μνήμη, πράγμα που είναι πολύ σύνηθες σε πολλές εφαρμογές όπου χρησιμοποιούνται DSP.

Σε πολλούς επεξεργαστές, η πολλαπλή πρόσβαση στη μνήμη μέσα σε ένα

κύκλο υπόκειται σε πολλούς περιορισμούς. Τυπικά, όλες εκτός από μία θέση μνήμης που πρόκειται να γίνει πρόσβαση πρέπει να βρίσκεται στο ίδιο κύκλωμα με τον επεξεργαστή (on-chip), και πολλαπλή πρόσβαση μπορεί να γίνει μόνο από συγκεκριμένες εντολές. Για να επιτευχθεί πολλαπλή πρόσβαση σε διαφορετικές θέσεις μνήμης, οι επεξεργαστές θα πρέπει να διαθέτουν πολλαπλούς διαδρόμους (buses) στο ίδιο κύκλωμα, multiported on-chip memories, και σε μερικές περιπτώσεις, multiple independent memory banks. Η αρχιτεκτονική της μνήμης στους DSP και η φιλοσοφία με την οποία είναι δομημένες είναι διαφορετική από τη φιλοσοφία που διέπει την αρχιτεκτονική των επεξεργαστών γενικού σκοπού (General Purpose Processors GPP).

2.4.3 Ειδικευμένες Μέθοδοι Διευθυνσιοδότησης

Για να επιτραπεί η μέγιστη ταχύτητα επεξεργασίας και ο προσδιορισμός πολλαπλών τελεστών σε μία μικρή λέξη εντολής, οι επεξεργαστές ψηφιακού σήματος περιέχουν μονάδες αφιερωμένες στη παραγωγή διευθύνσεων, οι οποίες δουλεύουν στο προσκήνιο, δημιουργώντας τις διευθύνσεις που απαιτούνται για την πρόσβαση των τελεστών με την ταυτόχρονη εκτέλεση αριθμητικών εντολών. Οι μονάδες αυτές συνήθως υποστηρίζουν κάποιες μορφές διευθύνσεων που έχουν επικρατήσει σε εφαρμογές που χρησιμοποιούν DSP. Οι πιο γνωστές από αυτές είναι η έμμεση-από-καταχωρητή διευθυνσιοδότηση με ταυτόχρονη προσαύξηση (register-indirect addressing with post-increment), η οποία χρησιμοποιείται σε περιπτώσεις όπου απαιτείται επεξεργασία δεδομένων όπου τα δεδομένα είναι στοιβαγμένα στη μνήμη στη σειρά. Άλλη μορφή διευθύνσεων είναι η κυκλική (circular) και η modulo, όπου χρησιμοποιούνται για να απλοποιηθεί η χρήση ενδιάμεσης προσωρινής μνήμης. Επίσης κάποιοι επεξεργαστές υποστηρίζουν διευθυνσιοδότηση με αντίστροφα-bit (bit-reversed) η οποία διευκολύνει την μετάφραση των αποτελεσμάτων ενός αλγορίθμου που υλοποιεί έναν μετασχηματισμό Fourier.

Κεφάλαιο 3

Υλοποίηση του Αναλυτή Φάσματος

3.1 Επιλογή Αρχιτεκτονικής

- Απαραίτητο Υλικό
- ASIC vs FPGA vs DSP vs GPU vs GPP
- Διαφορές FPGA και DSP

3.1.1 Απαραίτητο Υλικό

"Τι υλικό είναι κατάλληλο για την υλοποίηση του αναλυτή φάσματος;"

Τα απαραίτητα μέρη από τα οποία αποτελείται ο αναλυτής φάσματος ο οποίος δουλεύει με τεχνικές Software Radio είναι τα εξής:

- Κεραία Broadband η οποία είναι υπεύθυνη για τη συλλογή του σήματος.

  Η κεραία πρέπει να είναι μια κεραία όσο το δυνατόν περισσότερο broadband ώστε να έχουμε λήψη σήματος από όλο το φάσμα και ο αναλυτής φάσματος να έχει τη δυνατότητα να απεικονίζει μεγάλο εύρος.

- Ένας γρήγορος μετατροπέας A/D και D/A

  Ο μετατροπέας πρέπει να είναι όσο πιο γρήγορος γίνεται και είναι αυτός που συνήθως μας περιορίζει περισσότερο στο εύρος φάσματος που μπορούμε να επεξεργαστούμε. Για παράδειγμα ένας μετατροπέας A/D/A που δειγματοληπτεί στα 100MHz μας δίνει ένα πολύ μικρό εύρος φάσματος των 50MHz για φασματική απεικόνιση. Την στιγμή που γράφεται αυτό το κείμενο, ένας τέτοιος μετατροπέας είναι αιχμή της τεχνολογίας και πολύ ακριβός. Επίσης για την επιλογή μετατροπέα πρέπει να λάβουμε υπόψη και την ακρίβεια και την ευαισθησία που έχει. Η ακρίβεια μετριέται σε bit ενώ η ευαισθησία σε Volt.

- Ένα σύστημα επεξεργασίας του σήματος

  Το σύστημα που θα λαμβάνει τα δείγματα και θα κάνει μαθηματικές επεξεργασίες είναι και η καρδιά του αναλυτή φάσματος. Το τι επιλογή θα κάνουμε για την καρδιά επηρεάζει την απόδοση του αναλυτή φάσματος και τελικά είναι η κρίσιμη επιλογή για το αν ο αναλυτής θα δουλεύει ικανοποιητικά η όχι. Οι επιλογές που έχουμε είναι πολλές. Προφανώς ο αναγνώστης που έχει διαβάσει τα περιεχόμενα έχει ήδη καταλάβει ποια είναι και η επιλογή που προταθεί. Αλλά ας δούμε όλες τις επιλογές με τα θετικά και τα αρνητικά τους.

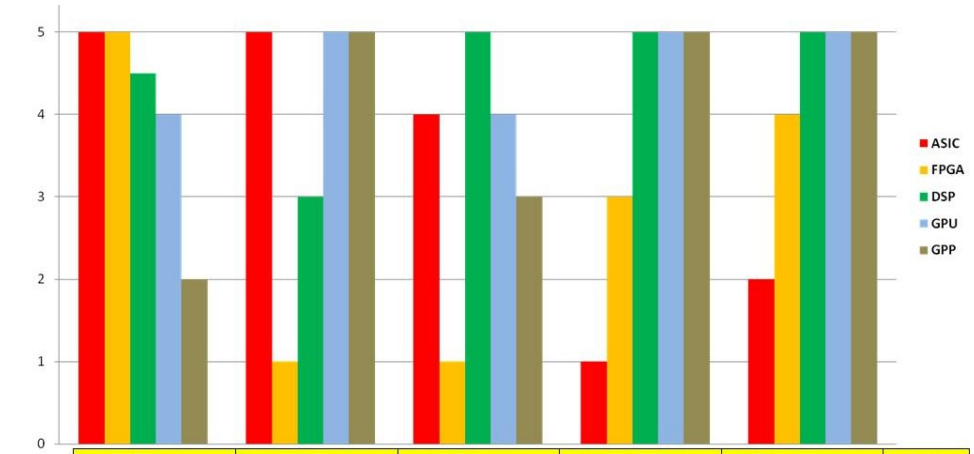
- Ένα σύστημα με οθόνη για την προβολή των αποτελεσμάτων

  Το σύστημα που θα προβάλλονται τα αποτελέσματα, μπορεί να είναι στην οθόνη του υπολογιστή που βρίσκεται μέσα ο DSP. Ακόμη μπορεί να είναι και ένα αρχείο στο σκληρό δίσκο όπου αποθηκεύονται κατάλληλα οι τιμές.

- Συνδέσεις μεταξύ των 4 προηγούμενων μερών

  Οι συνδέσεις των τεσσάρων αυτών μερών πρέπει να έχουν τα εξής χαρακτηριστικά: Το καλώδιο από την κεραία στον ADC πρέπει να είναι κατάλληλο έτσι ώστε να μην έχουμε απώλειες σε βαθμό να ξεφεύγει από τις τιμές σωστές λειτουργίας του ADC. Ο DAQ επικοινωνεί με τον DSP μέσω ενός πολύ γρήγορου καναλιού που σερβίρεται από ένα FPGA. Αυτό γίνεται γιατί χρειάζεται πάρα πολύ μεγάλη ταχύτητα στη μετάδοση των δειγμάτων. Αρκεί να σκεφτεί κανείς ότι σε 100MHz λειτουργία έχουμε 14bit δείγματος κάθε 1/100.000.000 του δευτερολέπτου ή 175MB/s σε real-time.

3.1.2 ASIC vs FPGA vs DSP vs GPU vs GPP



	Performance	Price	Power	Flexibility	Development	Score
ASIC	5	5	4	1	2	17
FPGA	5	1	1	3	4	14
DSP	4,5	3	5	5	5	22,5
GPU	4	5	4	5	5	23
GPP	2	5	3	5	5	20

Σχήματα 3.1: Σύκριση των Αρχιτεκτονικών

- ASIC (Application-specific integrated circuit)
  - Θετικά:
    - Η πιο καλή λύση από πλευράς απόδοσης σε ταχύτητα.
  - Αρνητικά:
    - hard-wired λογική

  Το γεγονός ότι είναι hard-wired η λογική, σημαίνει ότι δεν είναι ευέλικτη και δεν μπορείς να αλλάζεις ή να παραμετροποιείς την επεξεργασία του σήματος. Πρέπει εξ αρχής να έχεις αποφασίσει τι θέλεις και να έχεις προβλέψει όλες τις αλλαγές που πρόκειται να θες στο μέλλον. Η επιλογή αυτή απορρίφθηκε διότι θεωρήθηκε πολύ σημαντικό να μπορείς να κάνεις αλλαγές στην επεξεργασία του σήματος είτε real-time είτε όχι.

- FPGA (Field-programmable gate array)
  - Θετικά:
    - Καλή λύση από πλευράς απόδοσης σε ταχύτητα.
    - Σχετικά παραμετροποιήσιμο.

- Αρνητικά:
  * Αργό Synthesizing.
  * Δυσκολία Προγραμματισμού.
  * Ακριβή λύση.
  * Περιορισμένο μέγεθος προγράμματος.

Η επιλογή αυτή απορρίφθηκε διότι οι εξαιρετικά μεγάλοι χρόνοι για την αλλαγή και παραμετροποίηση της επεξεργασίας του σήματος, κυρίως όταν επρόκειτο για real-time αλλαγές το καθιστούσαν μη πρακτικό. Παρόλα αυτά FPGA χρησιμοποιήθηκε για την μεταφορά των δειγμάτων από τον A/D στο σύστημα επεξεργασίας όπου η δουλειά ήταν συγκεκριμένη και αυστηρώς καθορισμένη.

- DSP (Digital Signal Processor)
  - Θετικά:
    * Μέτρια προς καλή λύση από πλευράς απόδοσης σε ταχύτητα.
    * Συμφέρουσα οικονομικά λύση.
    * Άκρως παραμετροποιήσιμο και προγραμματίσιμο.
  - Αρνητικά:
    * Κατανάλωση περισσότερης ισχύος.
    * Όχι τόσο γρήγορο όσο οι προηγούμενες λύσεις.

Για το σύστημα επεξεργασίας χρησιμοποιήθηκε ένας τελευταίας τεχνολογίας DSP. Το γεγονός ότι ήταν άμεσα προγραμματίσιμος ο επεξεργαστής σε γλώσσα C και το γεγονός ότι είχε σχετικά καλή απόδοση παρόλο που δεν είχε ταχύτητες της hard-wired λογικής, ήταν ο λόγος για την επιλογή. Επίσης η τεχνολογία των DSP και οι ταχύτητές του αυξάνονται με μεγαλύτερο ρυθμό από τις δύο προηγούμενες λύσεις σε βαθμό που πιστεύεται ότι η ταχύτητα των FPGA θα ξεπεραστεί από τους DSP.

- GPU (Graphics Processing Unit)
  - Θετικά:
    * Μέτρια λύση από πλευράς απόδοσης σε ταχύτητα με προοπτικές για το μέλλον
    * Συμφέρουσα οικονομικά λύση.
    * Παραμετροποιήσιμο και προγραμματίσιμο.
  - Αρνητικά:

    * Κατανάλωση περισσότερης ισχύος.
    * Όχι τόσο γρήγορο όσο οι προηγούμενες λύσεις.

  Οι GPU είναι στην ουσία DSP επεξεργαστές επικεντρωμένοι σε λειτουργίες για γραφικά. Τελευταία υπάρχει η δυνατότητα προγραμματισμού αυτών των καρτών γραφικών με πρόγραμμα γραμμένο σε C όπως δίνει η Nvidia με την ονομασία CUDA. Είναι μια καλή ευκαιρία το software radio να έρθει πιο κοντά στο mainstream pc. Παροτρύνω κάποιον φοιτητή στην διπλωματική του εργασία να δοκιμάσει και να κάνει benchmarks του αναλυτή φάσματος σε μια κάρτα γραφικών νέας τεχνολογίας.

- GPP (General Processing Processor)
  - Θετικά:
    * Συμφέρουσα οικονομικά λύση.
    * Παραμετροποιήσιμο και προγραμματίσιμο.
  - Αρνητικά:
    * Κατανάλωση περισσότερης ισχύος.
    * Χαμηλές επιδόσεις.

  Οι επεξεργαστές γενικού σκοπού, παρόλο που στην αιχμή της τεχνολογίας σήμερα βλέπουμε τετραπύρηνους και σύντομα οχταπύρηνους, δεν μπορούν να φτάσουν σε επιδόσεις τους επεξεργαστές DSP. Παρόλα αυτά, με κατάλληλο προγραμματισμό ώστε να εκμεταλλευτεί κάποιος την παραλληλία αυτών των επεξεργαστών και με δεδομένο ότι έχουμε πολύ καλούς C compilers φτιαγμένους για αυτές τις πλατφόρμες, καθώς επίσης και το πλήθος των επενδύσεων για βελτίωση αυτού του τύπου των επεξεργαστών, είναι άξιοι να θεωρηθούν σαν μια επιλογή. Μάλιστα το GNU Radio (google it) είναι μια τέτοια εφαρμογή που έχει φτάσει σε τέτοιες επιδόσεις ώστε να μπορείς να ακούσεις ραδιόφωνο με τεχνικές software radio.

3.1.3 Διαφορές FPGA και DSP

Στο εμπόριο υπάρχουν πληθώρα συστήματα software radio που αντί για DSP χρησιμοποιούν FPGA. Γι αυτό το λόγο αναφέρονται ειδικότερα τα πλεονεκτήματα και τα μειονεκτήματα αυτών των δύο αρχιτεκτονικών.

- Υψηλός Ρυθμός Δειγματοληψίας χωρίς απώλειες.
  - DSP: δυσκολία λόγω των μοιραζόμενων μερών (memory buses, interrupts)

  - FPGA: αφιερωμένη για αυτό το σκοπό λογική στο υλικό (high I/O Rates)

- Μεγάλος όγκος δεδομένων για επεξεργασία.
  - DSP: έχει λειτουργίες και διαθέτει εξωτερική μνήμη
  - FPGA: περιορισμένος όγκος εσωτερικού αποθηκευτικού χώρου

- Όταν το πρόγραμμα έχει πολλές περιπτώσεις
  - DSP: εύκολα υλοποιήσιμο με το να γίνεται διακλάδωση στο κατάλληλο μέρος του προγράμματος
  - FPGA: χρειάζεται αφιερωμένη λογική στο υλικό για κάθε περίπτωση του προγράμματος

- Δυνατότητες διακλάδωσης και λήψης αποφάσεων
  - DSP: μπορεί να τρέξει ένα πρόγραμμα σε C όπως για παράδειγμα τα πρωτόκολλα επικοινωνιών
  - FPGA: αυτό είναι δύσκολο να πραγματοποιηθεί με FPGA

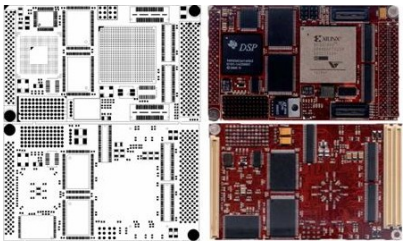
- Επαναπρογραμματισμός
  - DSP: απλά φορτώνεις ένα νέο πρόγραμμα στον επεξεργαστή
  - FPGA: μεγάλοι χρόνοι επαναπρογραμματισμού.

3.2 Περιγραφή επιλεγμένου υλικού

- Το πακέτο (SMT8036) είναι ένα επαγγελματικό σύστημα βασισμένο σε τεχνολογίες DSP, FPGA και DAQ που έχει:
  - Δύο κανάλια 14-bit ADC 105 MSPS το καθένα
  - FPGA για προεπεξεργασία
  - Έναν επεξεργαστή DSP υψηλής ταχήτητας TMS320C64x DSP για την εκτέλεση των προγραμμάτων
  - Ξανά FPGA για μετα-επεξεργασία
  - Και τέλος ένας δικάναλος TxDAC που μετατρέπει τα ψηφιακά δείγματα σε αναλογικά σήματα με ρυθμό 400 MSPS.
- Το SMT8036 αποτελείται από
  - ένα C64xx-based module (SMT365) συνδυασμένο με
  - ένα διπλό υψηλής ταχύτητας ADC/DAC module (SMT370)

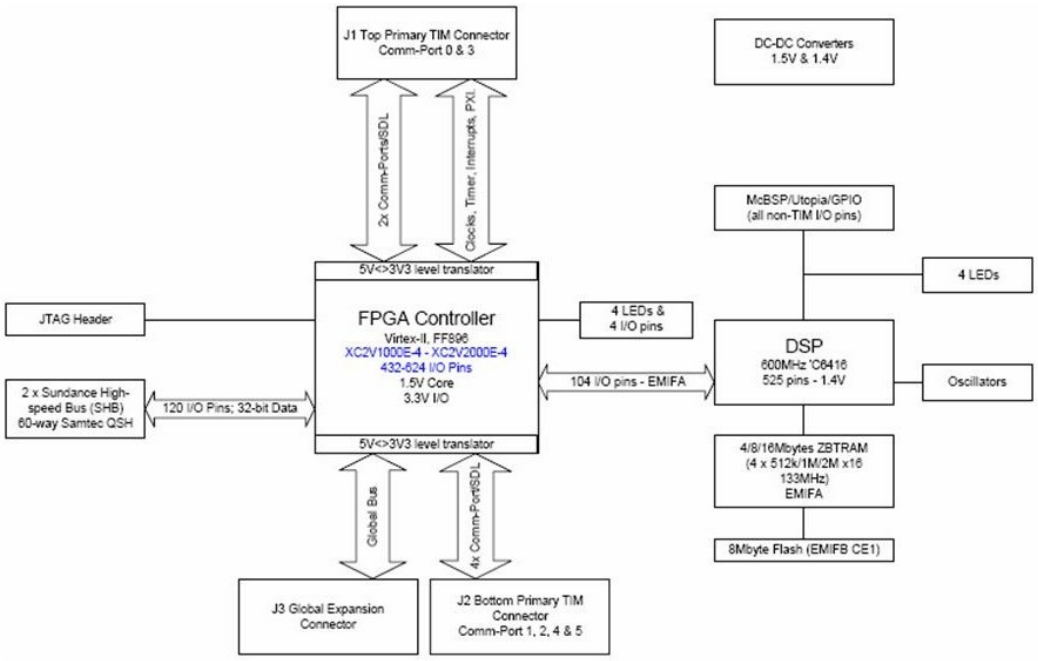
Σχήματα 3.2: SMT8036

- SMT365 χαρακτηριστικά:



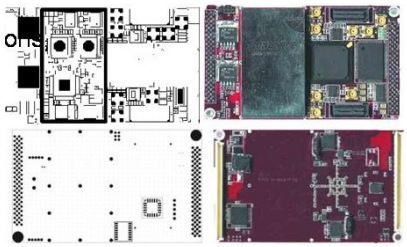
Σχήματα 3.3: SMT365

  - TMS320C6416 επεξεργαστής στα 600MHz
  - Έξι 20MB/s πόρτες επικοινωνίας (Comm.-ports)
  - 8MB από ZBTRAM (133MHz)
  - 8MByte Flash ROM για κώδικα εκκίνησης και προγραμματισμό του FPGA
  - Xilinx Virtex II XC2V2000-4
  - Global expansion connector
  - Υψηλού Bandwidth I/O δεδομένων μέσω 2 Sundance υψηλής ταχύτητας διαδρόμων (SHB).



Σχήματα 3.4: SMT365 block diagram

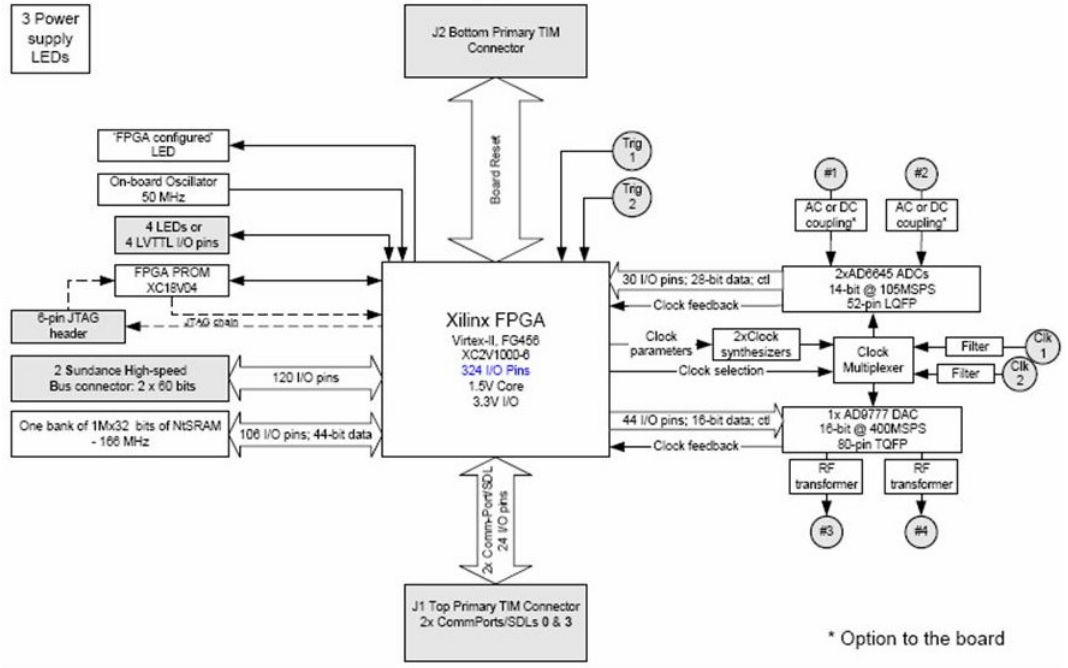
- SMT370 χαρακτηριστικά:



Σχήματα 3.5: SMT370

  - Δύο 14-bit ADCs (AD6645-105) μέγιστης δειγματοληψίας 105MHz,
  - Διπλός 16-bit TxDAC (AD9777) ρυθμού 400MHz (interpolation),
  - Two Sundance High-speed Bus (SHB) connectors,
  - 50-Ohm terminated analogue inputs and outputs, external triggers and clocks via MMBX (Huber and Suhner) connectors,

– Default FPGA firmware implementing all the functions described in the User Manual.
– User defined pins for external connections,
– TIM standard compatible
– Two 20 MegaByte/s comm.-ports,
– Low-jitter on-board system clock,
– Xilinx Virtex-II XC2V1000-6 FPGA,



Σχήματα 3.6: SMT370 block diagram

3.3 Περιγραφή των εργαλείων ανάπτυξης

- Γενικά Στοιχεία
- Αφηρημένο Μοντέλο
- Επεξεργαστές, καλωδιώσεις και συνδέσεις

3.3.1 Γενικά Στοιχεία

Ο τρόπος προγραμματισμού και δημιουργίας εφαρμογών στο περιβάλλον του Diamond διαφέρει σημαντικά από τον παραδοσιακό τρόπο προγραμματισμού των DSP που συνήθως γίνεται μέσω του περιβάλλοντος Code Composer Studio (CSS). Το CSS έχει σχεδιαστεί να παράγει εφαρμογές κυρίως σε συστήματα με μία μονάδα επεξεργασίας. Το περιβάλλον του Diamond προσφέρει απλές μεθόδους για δημιουργία εφαρμογών με πολυεπεξεργασία.

Το περιβάλλον του Diamond δεν είναι αυτό που θα χωρίσει ένα ήδη υπάρχον πρόγραμμα σε πολλά μικρότερα τα οποία θα τρέχουν σε διαφορετικούς επεξεργαστές. Αυτό είναι δουλεία του προγραμματιστή και γίνεται βάσει των επιδιώξεων του και τις απαιτήσεις του προγράμματος.

Το περιβάλλον του Diamond έχει τρία στάδια για τη δημιουργία μιας εφαρμογής:

- Μεταγλώττιση των αρχείων κώδικα με τον μεταγλωττιστή της Texas Instruments ο οποίος βρίσκεται ενσωματωμένος με το περιβάλλον CSS
- Σύνδεση με τον linker της Texas Instruments, για τη δημιουργία των αρχείων κάθε διεργασίας (.tsk)
- Χρήση του ρυθμιστή (configurer) για την ένωση των αρχείων των διεργασιών σε ένα και μόνο αρχείο της εφαρμογής (.app) που περιέχει οτιδήποτε χρειάζεται για να τρέξει η εφαρμογή στο DSP Board. Συνήθως δεν χρειάζεται να ανησυχεί ο προγραμματιστής για την ανάθεση μνήμης στην εφαρμογή, καθώς αυτό γίνεται αυτόματα από τον ρυθμιστή (configurer). Βεβαίως επιτρέπεται και χειροκίνητη ρύθμιση όπου απαιτείται.

Από τη στιγμή που θα δημιουργηθεί μια Diamond εφαρμογή, φορτώνεται στο DSP και εκτελείται από ένα Server πρόγραμμα που εκτελείται στον υπολογιστή.

3.3.2 Αφηρημένο Μοντέλο

Το μοντέλο της Diamond για παράλληλο προγραμματισμό είναι βασισμένο στην ιδέα της επικοινωνίας ακολουθιακών διεργασιών. Σε αυτό το μοντέλο, ένα υπολογιστικό σύστημα είναι μια συλλογή από ακολουθιακές διεργασίες που είναι ενεργές ταυτόχρονα. Στον επεξεργαστή εκτελείται μία διεργασία σε κάθε μονάδα επεξεργασίας, αλλά η γρήγορη εναλλαγή των διεργασιών δίνει την αίσθηση της ταυτόχρονης εκτέλεσης των διεργασιών. Ο τρόπος με τον οποίο γίνεται η επικοινωνία μεταξύ των διεργασιών είναι μέσω καναλιών.

Ένα κανάλι μπορεί να μεταφέρει μηνύματα από μια διεργασία σε μια άλλη. Επίσης μπορεί να μεταφέρει μηνύματα μόνο προς μια κατεύθυνση. Αν χρειασθεί αμφίδρομη επικοινωνία μεταξύ δύο διεργασιών θα πρέπει να χρησιμοποιηθούν δύο κανάλια. Εξίσου αυτός που στέλνει το μήνυμα και αυτός που το λαμβάνει θα πρέπει να συμφωνούν για ίδιο μέγεθος μηνύματος πριν γίνει η επικοινωνία. Όταν ένα μήνυμα στέλνεται ή λαμβάνεται, τότε η διεργασία μπλοκάρεται. Κάθε πρόγραμμα μπορεί να έχει όσα κανάλια για είσοδο ή έξοδο θέλει, αλλά αυτά τα κανάλια είναι στατικά και δηλωμένα εξ αρχής, δηλαδή δεν μπορούν να δημιουργηθούν δυναμικά κατά τη διάρκεια που τρέχει το σύστημα.

Για παράδειγμα, μια διαδικασία αντιγραφής δισκέτας σε μια άλλη, φτιαγμένη σε κάποιο λειτουργικό σύστημα θα μπορούσε να περιγραφεί σαν τρεις διεργασίες που εκτελούνται ταυτόχρονα: δύο διεργασίες για τον ελεγκτή δισκέτας και μια διεργασία που θα κάνει την αντιγραφή.

Δισκ1 -¿ αντιγραφή -¿ δισκ2

Αυτό το παράδειγμα μας δείχνει μια πολύ σημαντική ιδιότητα της επικοινωνίας μέσω καναλιών: είναι συγχρονισμένα. Μια διεργασία που θέλει να στείλει ένα μήνυμα πάνω από ένα κανάλι είναι πάντα υποχρεωμένη να περιμένει μέχρι η διεργασία που λαμβάνει διαβάσει όλο το μήνυμα: αυτό είναι γνωστό σαν blocking communication. Στο παράδειγμά μας αυτό σημαίνει, ότι αν σε περίπτωση που για κάποιο λόγο η δισκέτα εξόδου δουλεύει πιο αργά και δεν μπορεί να φθάσει τον ρυθμό της δισκέτας πηγής, το σύστημα παρόλα αυτά θα δουλέψει άψογα και η αντιγραφή θα γίνει κανονικά. Αυτό γίνεται γιατί η διαδικασία αντιγραφής θα υποχρεωθεί να περιμένει αν δοκιμάσει να στείλει μήνυμα, πριν ετοιμαστεί να το δεχθεί η δισκέτα που λαμβάνει. Μερικές φορές είναι χρήσιμο να επιτρέπεται στην διεργασία που στέλνει να προτρέχει εκείνης που λαμβάνει αλλά σε αυτές τις περιπτώσεις έχει προβλεφθεί και έχει προστεθεί ένας buffer για αυτό τον σκοπό.

Το μοντέλο της Diamond για παράλληλη επεξεργασία ακολουθεί το προηγούμενο αφηρημένο μοντέλο με κάποιες διαφορές και κάποιες ελαστικότητες

3.3.3 Επεξεργαστές, καλωδιώσεις και συνδέσεις

Το υλικό στο οποίο εκτελείται μια εφαρμογή Diamond περιγράφεται σαν ένα δίκτυο επεξεργαστών, όπου ο καθένας έχει συνδέσεις με καλώδια σε ζευγάρια. Κάθε καλώδιο είναι ένα αμφίδρομο μονοπάτι μεταξύ ακριβώς δύο επεξεργαστών. Στην πράξη, οι κατασκευαστές προσφέρουν διάφορους τρόπους για να ενώνονται οι επεξεργαστές: εξειδικευμένο υλικό για point-to-point ένωση, κοινόχρηστη μνήμη, δομές bus, και πολλούς άλλους. Το Diamond αποκρύπτει αυτές τις λεπτομέρειες από τον χρήστη και συμπεριφέρεται σαν να υπήρχε πραγματικό καλώδιο ανάμεσα στους επεξεργαστές, δίνοντας πρόσβαση σε αυτούς μέσω συνδέσεων που αριθμούνται ξεκινώντας από το 0. Στο παραπάνω διάγραμμα, ο επεξεργαστής P1 έχει δύο συνδέσεις: η σύνδεση 0 τον ενώνει με τον επεξεργαστή P2 χρησιμοποιώντας το καλώδιο W1, και η σύνδεση 1 τον ενώνει με τον επεξεργαστή P3 χρησιμοποιώντας το καλώδιο W2.

3.3.4 Ο κεντρικός επεξεργαστής

Το σημαντικό για όλα τα δίκτυα επεξεργαστών που υποστηρίζονται από την Diamond είναι ότι είναι συνδεδεμένα μεταξύ τους. Κάθε επεξεργαστής στο δίκτυο μπορεί να προσπελασθεί από οποιονδήποτε άλλον επεξεργαστή μέσω καλωδίωσης, κανείς επεξεργαστής δεν είναι απομονωμένος. Εξαιτίας αυτού, είναι δυνατόν να ξεχωρίσουμε κάθε επεξεργαστή που συμμετέχει μονοσήμαντα, ξεκινώντας από έναν σαν ρίζα και δίνοντας την ακολουθία συνδέσεων που πρέπει να ακολουθήσει κανείς για να φτάσει στους υπόλοιπους. Στο παραπάνω παράδειγμα, ξεκινώντας από τον P1, μπορούμε να βρούμε τον P5 ακολουθώντας τη σύνδεση 1 και ύστερα την σύνδεση 3.

Το σημείο έναρξης σε κάθε εφαρμογή diamond είναι ένας επεξεργαστής, ο οποίος καλείται κεντρικός επεξεργαστής ή ρίζα. Ο κεντρικός είναι η βάση του δικτύου και ο δρόμος για τον εντοπισμό των άλλων επεξεργαστών.

3.3.5 Ο host

Οι εφαρμογές του Diamond συνήθως τρέχουν σε DSP πλακέτες που είναι συνήθως πάνω σε ένα host PC, αν και αυτό δεν είναι απαραίτητο. Ο host είναι υπεύθυνος για την φόρτωση της εφαρμογής και για την επικοινωνία μαζί της. Παρόλο που είναι εφικτό να φτιάξει κανείς τη δική του εφαρμογή για αυτή τη δουλειά, το diamond προσφέρει το πρόγραμμα server που αναλαμβάνει αυτές τις υποχρεώσεις.

3.3.6 Διεργασίες (Tasks)

Μια ολοκληρωμένη εφαρμογή είναι μια συλλογή από μία η περισσότερες διεργασίες που εκτελούνται ταυτόχρονα και είναι συνδεδεμένες με κανάλια. Κάθε διεργασία είναι και ένα ξεχωριστό C πρόγραμμα, με τη δικιά του ξεχωριστή main function και τις δικές του περιοχές μνήμης για κώδικα και δεδομένα. Οι διεργασίες diamond θα πρέπει να πειθαρχούν στους περιορισμούς των καταχωρητών που βάζει ο C compiler της Texas Instruments. Ο κώδικας μιας διεργασίας είναι ένα αρχείο και μόνο που δημιουργείται από τον συνδέτη (linker). Οι διεργασίες diamond δεν περιέχουν στατικές διευθύνσεις μνήμης και γι αυτόν τον λόγο είναι επανεντοπίσημες. Ο διαμοιρασμός της μνήμης γίνεται σε μετέπειτα στάδιο. Είναι στην αρμοδιότητα του προγραμματιστή να διαλέξει την κάθε διεργασία σε ποιον επεξεργαστή θα εκτελείται. Μπορούμε να έχουμε όσες διεργασίες θέλουμε σε κάθε επεξεργαστή, το μόνο που μας περιορίζει είναι η διαθέσιμη μνήμη. Μπορούμε ακόμη να έχουμε και αντίγραφα της ίδιας διεργασίας σε έναν επεξεργαστή. Κάθε επεξεργαστής θα εκτελεί μόνο και μόνο τις διεργασίες οι οποίες έχουν προοριστεί για αυτόν από τον προγραμματιστή.

Υπάρχουν δύο ειδών διεργασίες: full διεργασίες και stand-alone διεργασίες

Full διεργασίες

Οι full διεργασίες είναι συνδεδεμένες με την standard run-time library (rtl.lib) με τη χρήση της εντολής 3L t. Με αυτόν τον τρόπο δίνεται στην διεργασία η δυνατότητα να επικοινωνεί με το host PC μέσω του server. Δηλαδή μπορεί να χρησιμοποιεί τις εντολές printf, scanf της standard C.

Stand-Alone διεργασίες

Οι stand-alone διεργασίες είναι συνδεδεμένες με την stand-alone library (sartl.lib) με τη χρήση της εντολής 3L ta. Αυτή η βιβλιοθήκη παραλείπει τις συναρτήσεις για επικοινωνία με τον host. Συγκεκριμένα, διεργασίες που είναι συνδεδεμένες με την stand-alone library δεν έχουν stdin, stdout και stderr και δεν μπορούν να πάρουν παράμετρο κατά την εκκίνηση τους από command line.

3.3.7 Πόρτες

Κάθε διεργασία έχει ένα διάνυσμα από "πόρτες εισόδου" και ένα διάνυσμα από "πόρτες εξόδου" που χρησιμοποιούνται για την σύνδεση των διεργασιών μεταξύ τους. Μια διεργασία είναι σαν ένα μαύρο κουτί από λογισμικό, έχοντας

επικοινωνία με τον έξω κόσμο μόνο μέσω των πόρτων του, όπως φαίνεται παρακάτω:

Οι διεργασίες ενώνονται συνδέοντας πόρτες εξόδου με πόρτες εισόδου χρησιμοποιώντας κανάλια, και αυτή η συλλογή από διεργασίες ενώνονται σε μία και μόνο εφαρμογή σε ένα αρχείο μέσω ενός εργαλείου που ονομάζεται ρυθμιστής (configurer).

3.3.8 Ορίσματα της main

```
int main(int argc, char *argv[], char *envp[],
            CHAN * in_ports[], int ins,
            CHAN *out_ports[], int outs)
```

Argc ο αριθμός των ορισμάτων από command-line Argv τα ορίσματα από command-line Envp πάντα null Inports το διάνυσμα των πόρτων εισόδου Ins το πλήθος των πόρτων εισόδου Outports το διάνυσμα των πόρτων εξόδου Outs το πλήθος των πόρτων εξόδου /par Για παράδειγμα, μια απλή διεργασία θα μπορούσε να δέχεται μια ροή από τιμές σε μια πόρτα εισόδου, όπως για παράδειγμα χαρακτήρες, και να τους μετατρέπει σε κεφαλαία, στέλνοντας τη ροή αυτή μέσω μιας πόρτα εξόδου σε μια άλλη διεργασία. Ο κώδικας μιας τέτοιας απλής διεργασίας φαίνεται παρακάτω.

```
#include <chan.h>
#include <ctype.h>
#include <stdio.h> // for EOF

main(int argc, char *argv[], char *envp[],
        CHAN * in_ports[], int ins,
        CHAN *out_ports[], int outs)
{
    int c;
    for (;;) {
    chan_in_word(&c, in_ports[0]);
    if (c == EOF) break; // terminate task
    chan_out_word(toupper(c), out_ports[0]);
    }
}
```

3.3.9 Πόρτες και Κανάλια

Μπορούμε να φανταστούμε τις διεργασίες σαν ατομικές μονάδες παράλληλων συστημάτων. Μπορούν να ενωθούν μεταξύ τους με κανάλια, όπως σε

ηλεκτρονικά συστήματα. Η σύνδεση μπορεί να επιτευχθεί στη μνήμη ή να είναι μια σύνδεση μεταξύ επεξεργαστών, είτε άμεσα μέσω καλωδίωσης ή έμμεσα με δρομολόγηση από λογισμικό. Κανάλια που συνδέουν διεργασίες στον ίδιο επεξεργαστή ονομάζονται εσωτερικά κανάλια και αυτά που συνδέουν διεργασίες σε διαφορετικούς ονομάζονται εξωτερικά κανάλια.

Τα διανύσματα των πόρτων εισόδου και εξόδου περνιούνται σαν παράμετροι στη main συνάρτηση. Κάθε πόρτα είναι τύπου "δείκτης σε κανάλι" (CHAN *). Η run-time βιβλιοθήκη του Diamond διαθέτει συναρτήσεις για την αποστολή και λήψη μηνυμάτων μέσα από κανάλια.

Ο ρυθμιστής μπορεί να καθοδηγηθεί ώστε να δημιουργήσει πολλές συνδέσεις. Κάθε σύνδεση είναι ένα κανάλι εισόδου συνδεδεμένο με ένα κανάλι εξόδου. Ο ρυθμιστής ενώνει αυτά τα κανάλια σε πόρτες διεργασιών και δημιουργεί διαδρομή από μια δεδομένη πόρτα εισόδου σε μια διεργασία σε μια δεδομένη πόρτα εισόδου μιας άλλης διεργασίας. Μπορεί κάποιος να έχει πρόσβαση σε αυτά τα κανάλια από τις διεργασίες είτε μέσω των παραμέτρων που έχουν περαστεί στην main, ή χρησιμοποιώντας το όνομα της σύνδεσης για να προσδιορίσει άμεσα το κανάλι.

3.3.10 Δίκτυα πολυεπεξεργασίας

Αν και κάθε επεξεργαστής μπορεί να τρέχει οσοδήποτε αριθμό από διεργασίες περιοριζόμενος μόνο από τη διαθέσιμη μνήμη, οι διεργασίες που επικοινωνούν μεταξύ τους δε χρειάζονται να βρίσκονται στον ίδιο επεξεργαστή. Στην πραγματικότητα, οι διεργασίες μιας εφαρμογής μπορούν να είναι διασκορπισμένες σε πολλούς επεξεργαστές στο δίκτυο.

Κάθε σύνδεση μέσω καναλιών από τον έναν επεξεργαστή στον άλλον υλοποιείται με μια φυσική σύνδεση ή καλώδιο μεταξύ των επεξεργαστών. Μπορεί να υλοποιηθεί με πολλούς τρόπους, αλλά η επικοινωνία που υποστηρίζεται είναι αμφίδρομη.

Όταν ένα κανάλι συνδέει διεργασίες που βρίσκονται σε διαφορετικούς επεξεργαστές, τα μηνύματα σε αυτό το κανάλι δρομολογούνται μέσω της σύνδεσης των επεξεργαστών. Κάθε σύνδεση μπορεί να υποστηρίξει μόνο δύο φυσικά κανάλια, ένα για κάθε φορά της διαδρομής. Ο προγραμματιστής έχει τη δυνατότητα να χρησιμοποιήσει αυτά τα δύο κανάλια ως έχουν ή να επιτρέψει στο diamond να τα χειριστεί και να δώσει ένα πλήθος από εικονικά κανάλια τα οποία θα είναι φυσικά και πιο αργά. Το δίκτυο δεν θα μπορεί να χτισθεί αν δεν υπάρχουν τόσες συνδέσεις μεταξύ δύο επεξεργαστών ώστε να υποστηρίξουν όλα τα απαιτούμενα κανάλια.

Κάθε διεργασία μπορεί να επικοινωνήσει με κάθε άλλη διεργασία, ανεξάρτητα από το πού είναι μέσα στο δίκτυο. Τα μηνύματα μπορούν να δρομολογηθούν μέσω ενδο-επεξεργαστικών συνδέσεων πάνω από εικονικά κανάλια.

Επιπλέον, όλες οι διεργασίες μπορούν να έχουν πρόσβαση στο C standard I/O.

3.3.11 Εικονικά και Φυσικά Κανάλια

Το diamond προκαθορισμένα, μετατρέπει όλα τα εξωτερικά κανάλια σε εικονικά, δηλαδή αυτόματα μεταφέρουν μηνύματα μεταξύ απομακρυσμένων επεξεργαστών μέσω ενδιάμεσων κόμβων του δικτύου. Εφόσον πολλά εικονικά κανάλια μπορούν και μοιράζονται τον ίδιο φυσικό σύνδεσμο, οσαδήποτε εικονικά κανάλια μπορούν να ενώσουν διεργασίες σε διαφορετικούς επεξεργαστές. Η μετάβαση πάνω από εικονικά κανάλια είναι εγγυημένα απαλλαγμένη από αδιέξοδα.

Παρόλα αυτά, τα εικονικά κανάλια είναι πιο αργά από τα υποβόσκοντα φυσικά κανάλια. Το diamond λοιπόν προσφέρει και φυσικά κανάλια, που κλειδώνουν κατευθείαν στο υλικό. Το πλεονέκτημά τους είναι η ταχύτητα ενώ το μειονέκτημα είναι ότι ο αριθμός των φυσικών συνδέσεων και τα μονοπάτια είναι περιορισμένα από το διαθέσιμο υλικό.

Όταν το throughput κάποιων καναλιών είναι κρίσιμο για την επίδοση της εφαρμογής, ο προγραμματιστής μπορεί να πειραματισθεί με τους τρόπους σύνδεσης και να επιλέξει τον καλύτερο για την εφαρμογή του. Ο πειραματισμός είναι εύκολος καθώς δεν χρειάζεται αλλαγή στον κώδικα γιατί οι συναρτήσεις για μεταβίβαση μηνυμάτων είναι ίδιες.

3.3.12 Ταυτόχρονη Είσοδος

Μερικές φορές, μια διεργασία που δέχεται είσοδο από δύο ή περισσότερα κανάλια δεν μπορεί να ξέρει πιο κανάλι θα είναι το επόμενο έτοιμο για να μεταφέρει το μήνυμα. Δεν μπορεί η διεργασία να διαβάζει κάθε κανάλι με τη σειρά, διότι αυτό θα μπλόκαρε τη διεργασία μέχρις ότου το ένα συγκεκριμένο κανάλι να γίνει έτοιμο, τα άλλα κανάλια που θα είναι ήδη έτοιμα να μεταφέρουν μηνύματα θα έπρεπε να περιμένουν.

Το diamond επιλύει αυτό το πρόβλημα με την ομάδα συναρτήσεων βιβλιοθήκης που βρίσκονται στο alt.h. Αυτές οι συναρτήσεις επιτρέπουν στο πρόγραμμα να περιμένει μέχρις ότου ένα κανάλι της ομάδας των εικονικών καναλιών ελευθερωθεί και είναι έτοιμο για επικοινωνία. Το κανάλι που ελευθερώνεται πρώτο αναγνωρίζεται από το καλών πρόγραμμα το οποίο έπειτα διαβάζει το μήνυμα χρησιμοποιώντας από τις ίδιες συναρτήσεις που χρησιμοποίησε για να στείλει μήνυμα.

3.3.13 Νήματα Παράλληλης Εκτέλεσης

Το diamond υποστηρίζει πολυνηματικές διεργασίες. Οι διεργασίες δημιουργούν δυναμικά νέα νήματα περνώντας ένα δείκτη σε μια συνάρτηση και το πλήθος της μνήμης σε μια συνάρτηση βιβλιοθήκης. Το νέο νήμα ξεκινάει να εκτελεί τον κώδικα της συνάρτησης ταυτόχρονα με το νήμα το οποίο το δημιούργησε. Το νέο νήμα τρέχει στο ίδιο περιβάλλον με τον δημιουργό του, μοιράζονται την στατική, εξωτερική και heap θέσεις μνήμης. Ο μόνος ιδιωτικός αποθηκευτικός χώρος του νέου νήματος είναι το workspace του. Το νήμα γονέας δεν έχει άμεσο έλεγχο του παιδιού του, το οποίο συνεχίζει να εκτελείται μέχρι να τερματίσει είτε γυρίζοντας από τη συνάρτηση από την οποία καλέστηκε ή από ειδική συνάρτηση βιβλιοθήκης. Αυτή η συμπεριφορά είναι παρόμοια με τα νήματα στον προγραμματισμό για windows και για μερικές εκδόσεις του Unix. Κάθε νήμα έχει την δική του stack μνήμη αλλά μοιράζεται τα υπόλοιπα δεδομένα του με όλα τα άλλα νήματα στην ίδια διεργασία.

Τα νήματα συνήθως δημιουργούνται κατά την αρχικοποίηση μιας διεργασίας. Η συχνή δημιουργία και τερματισμός νημάτων συνήθως προδίδει μια κακή υλοποίηση του προγράμματος.

Συναρτήσεις σημαφόρων στην run-time βιβλιοθήκη μπορούν να χρησιμοποιηθούν για να αποφευχθεί η παρέμβαση του ενός νήματος με το άλλο. Διαφορετικά, εσωτερικά κανάλια δηλωμένα σαν μεταβλητές του προγράμματος μπορούν να χρησιμοποιηθούν για να συγχρονιστούν οι λειτουργίες των νημάτων και να γίνει η μεταβίβαση μηνυμάτων για την μεταξύ τους επικοινωνία. Το diamond προσφέρει έναν τύπο δεδομένων τον CHAN που μπορεί να χρησιμοποιηθεί για την δήλωση τέτοιων καναλιών μεταβλητών.

Φυσικά, όπως σε οποιοδήποτε κτίσιμο μιας εφαρμογής, τα νήματα χρησιμοποιούνται και χωρίς να είναι αυστηρώς απαραίτητη η χρήση τους. Αυτό γίνεται γιατί πολλά προβλήματα από τη φύση τους επιλύονται καλύτερα και πιο ανθρώπινα με τη χρήση νημάτων και πολυνηματικών διεργασιών.

Όταν μια διεργασία ξεκινήσει την εκτέλεσή της, η κύρια συνάρτησή της συμπεριφέρεται σαν ένα νήμα, συγκεκριμένα, μπορεί να διακοπεί η εκτέλεσή της με την συνάρτηση thread_stop.

3.3.14 Ρυθμίζοντας μια εφαρμογή

Όταν μια εφαρμογή σχεδιαστεί και γραφτεί σαν μια συλλογή από διεργασίες που επικοινωνούν, πώς γίνεται να φορτωθεί σε ένα δίκτυο επεξεργαστών;

Πρώτα, κάθε ξεχωριστή διεργασία χτίζεται με το να μεταγλωτισθούν όλα τα αρχεία κώδικα με τον C compiler και χρησιμοποιώντας τον linker να συνδεθούν τα object αρχεία με τα απαραίτητα κομμάτια κώδικα από την run-time βιβλιοθήκη. Επαναλαμβάνοντας αυτή τη διαδικασία για κάθε διεργασία έχουμε

ως αποτέλεσμα πολλά αρχεία που κάθε ένα αντιπροσωπεύει και μια διεργασία.

Τώρα αυτά τα πολλά αρχεία-διεργασίες θα πρέπει να συνενωθούν σε ένα και μοναδικό εκτελέσιμο αρχείο που θα αποτελεί και το αρχείο της εφαρμογής. Το πρόγραμμα που κάνει όλες αυτές τις δουλειές ονομάζεται configurer. Σαν είσοδο το πρόγραμμα αυτό δέχεται ένα αρχείο γραμμένο από τον προγραμματιστή σε μορφή κειμένου και καθορίζει:

- Την δομή του υλικού
  - Διαθέσιμοι επεξεργαστές
  - Συνδέσεις μεταξύ των επεξεργαστών
- Την δομή του λογισμικού
  - Οι διεργασίες που πρόκειται να ενσωματωθούν
  - Συνδέσεις με κανάλια μεταξύ τους
- Τον τρόπο που θα γίνει η αντιστοίχηση του λογισμικού στο υλικό

Ο ρυθμιστής κρατάει μνήμη για τις διεργασίες και τις ενώνει σε ένα και μοναδικό αρχείο της εφαρμογής το οποίο μπορεί να φορτωθεί στο υλικό που έχει προσδιορισθεί και να εκτελεσθεί χρησιμοποιώντας την εντολή του Diamond: 3L x. Ο ρυθμιστής είναι επίσης υπεύθυνος για τον καθορισμό των διεργασιών που πρέπει να φορτωθούν.

Για να γίνει μια αλλαγή στον τρόπο με τον οποίο οι διεργασίες είναι συνδεδεμένες μεταξύ τους η στο ποιες διεργασίες είναι να εκτελεστούν, δεν είναι απαραίτητη μια αλλαγή στον κώδικα τον διεργασιών ούτε χρειάζεται κάποια επαναμεταγλώττιση και επανασύνδεση. Η αλλαγή γίνεται μόνο στον ρυθμιστή όπου αλλάζουμε το αρχείο κειμένου που προσδιορίζει αυτές τις πληροφορίες. Έτσι λοιπόν για παράδειγμα, είναι εφικτό να παράγουμε μια εφαρμογή που όλες οι διεργασίες θα εκτελούνται σε έναν επεξεργαστή, έπειτα να αλλάξουμε γνώμη, να την ρυθμίσουμε αλλιώς, και χωρίς καμία αλλαγή στον κώδικα να τρέξει η εφαρμογή σε δίκτυο επεξεργαστών. Τα φυσικά κανάλια μπορούν να αντικατασταθούν διαφανώς από εικονικά κανάλια με παρόμοιο τρόπο.

3.3.15 Δομή της Εφαρμογής

Μέχρι τώρα έχουμε δει εφαρμογές Diamond σαν ένα δίκτυο διεργασιών απλωμένες σε ένα πλήθος επεξεργαστών, με πιθανόν πολλές διεργασίες σε κάθε επεξεργαστή. Αυτές οι διεργασίες επικοινωνούν μεταξύ τους μόνο μέσω καναλιών. Αν μια διαδικασία επιθυμεί να επικοινωνήσει με μια διεργασία σε έναν

άλλον επεξεργαστή, τα μηνύματα θα μεταφερθούν σε αυτόν τον επεξεργαστή μέσω συνδέσεων των επεξεργαστών μεταξύ τους.

Τυπικά, το δίκτυο ελέγχεται από έναν άλλον επεξεργαστή τον οποίο ονομάζουμε host. Ο host επικοινωνεί άμεσα με μόνο έναν επεξεργαστή στο δίκτυο τον οποίο ονομάζουμε κεντρικό επεξεργαστή ή ρίζα. Στην πραγματικότητα, η εφαρμογή περιλαμβάνει πλήθος άλλων πραγμάτων τα οποία θα εξετάσουμε παρακάτω.

3.3.16 Ο Μικροπυρήνας

Ο ρυθμιστής αυτόματα τοποθετεί σε κάθε επεξεργαστή έναν κατάλληλο πυρήνα λογισμικού, ο οποίος κάποιες δουλειές που κάνει είναι:

- Χρονοπρογραμματισμός των νημάτων που τρέχουν στον επεξεργαστή
- Διαχείριση των εσωτερικών συνδέσεων και καναλιών των επεξεργαστών
- Ρύθμιση της διεργασίας που κρατάει τον χρόνο
- Διαθέτει τα απαραίτητα για την χρήση σεμαφόρων και συμβάντων
- Διαχειρίζεται τις διακοπές

Ο πυρήνας είναι ένα παθητικό στοιχείο στο σύστημα. Καταναλώνει χρόνο από τον επεξεργαστή μόνο όταν καλείται να κάνει κάτι, είτε σαν αποτέλεσμα μιας διακοπής η λόγω μιας αναφοράς από κάποιο πρόγραμμα (για παράδειγμα μια λειτουργία σύνδεσης ή σεμαφόρου). Οι διακοπές του TIMER0 χρησιμοποιούνται από τον πυρήνα για να διαχειρισθεί το εσωτερικό του ρολόι (timer_now, timer_delay, timer_wait) και για τον χωρισμό τον χρονοθυρίδων. Η διαχείριση αυτή του χρόνου χρειάζεται περίπου 25 κύκλους κάθε χιλιοστό του δευτερόλεπτου. Αν υπάρχει μόνο ένα νήμα στον επεξεργαστή δε θα γίνει χωρισμός σε χρονοθυρίδες. Αν θέσουμε CLOCK=0 στον επεξεργαστή, τότε ο χωρισμός σε χρονοθυρίδες διακόπτεται καθώς και οι διακοπές του ρολογιού του πυρήνα. Όλοι οι κύκλοι του επεξεργαστή θα είναι διαθέσιμοι για την εφαρμογή. Βέβαια αυτό που περιγράφηκε τώρα σπανίως είναι συμφέρον για μια εφαρμογή.

Ο πυρήνας είναι πολύ αποδοτικός και η απόδοσή του δεν εξαρτάται από τον αριθμό των νημάτων στο σύστημα.

3.3.17 Χρονοδρομολογητής

Σε έναν επεξεργαστή μπορούν να τρέχουν πολλές διεργασίες, και οι διεργασίες αυτές αποτελούνται από ένα η περισσότερα νήματα, τα οποία εκτελούνται

ταυτόχρονα. Ο μικροπυρήνας μοιράζει τον διαθέσιμο χρόνο του επεξεργαστή ανάμεσα στα διάφορα νήματα. Αυτή η διαδικασία είναι γνωστή ως χρονοδρομολόγηση. Κάθε νήμα έχει μια δεδομένη προτεραιότητα, ένα νούμερο μεταξύ 0 και 7, το οποίο καθορίζει το πόσο σημαντικό είναι το νήμα σε σχέση με τα υπόλοιπα που τρέχουν στον ίδιο επεξεργαστή. Όσο πιο μικρό είναι το νούμερο, τόσο μεγαλύτερη η προτεραιότητα του νήματος. Τα νήματα που έχουν την ύψιστή προτεραιότητα (δηλαδή έχουν 0) ονομάζονται και επείγον νήματα.

Ένα νήμα που εκτελείται μπορεί να μπει στην αναμονή και να μην μπαίνει στην ουρά της χρονοδρομολόγησης ή ένα νήμα μπορεί να συνεχίσει την εκτέλεσή του όταν συμβεί κάτι από αυτά:

- Το νήμα επιλέγει να μην χρονοδρομολογηθεί καλώντας την thread_deschedule
- Το νήμα πρέπει να περιμένει κάτι (επικοινωνία μέσω καναλιού, έναν σημαφόρο, ένα συμβάν, ή το ρολόι)
- Το νήμα παραγκωνίζεται από ένα νήμα με υψηλότερη προτεραιότητα το οποίο είναι έτοιμο να συνεχίσει την εκτέλεση
- Το νήμα δρομολογείται από τον χρονοδρομολογητή διότι έχει εκτελεστεί στον χρόνο που του διατέθηκε χωρίς διακοπή και η χρονοθυρίδα του τελείωσε.

Τα επείγον νήματα με την ύψιστη προτεραιότητα, με νούμερο 0, δεν επιτρέπεται να παραγκωνιστούν από άλλο νήμα ακόμα και να είναι επείγον και αυτό. Δηλαδή, τα επείγον μηνύματα είναι σαν να έχουν μια χρονοθυρίδα άπειρης διάρκειας και θα εκτελούνται για πάντα χωρίς κάποιος να τα βάζει σε αναμονή. Τα νήματα μικρότερης προτεραιότητας, αλλά ακόμα και τα επείγον νήματα, θα μένουν τελείως εκτός εκτέλεσης. Η σωστή χρήση των επείγον νημάτων μπορεί να βελτιώσει πάρα πολύ την απόδοση της εφαρμογής, αλλά μια αλόγιστη χρήση τους θα μπορούσε εξίσου να μειώσει πάρα πολύ την απόδοση. Για αυτόν το λόγο, τα επείγον νήματα πρέπει να χρησιμοποιούνται με πολύ προσοχή. Ένας γενικός κανόνας είναι ότι τα επείγον νήματα χρησιμοποιούνται για διεργασίες επικοινωνίας.

Άπαξ και ένα νήμα μπει σε αναμονή, το επόμενο νήμα που έχει τη μεγαλύτερη προτεραιότητα και είναι έτοιμο εκτελείται. Όταν τα νήματα έχουν την ίδια προτεραιότητα με εξαίρεση τα επείγον, ο μικροπυρήνας θα μοιράσει τον χρόνο του επεξεργαστή δίκαια σε όλα τα νήματα και με fifo λογική.

Η αλλαγή των νημάτων εξαιτίας διακοπής είναι λίγο πιο ακριβή σε χρόνο επεξεργαστή από την αλλαγή για οποιοδήποτε άλλο λόγο.

Στη δομή της εφαρμογής συγκαταλέγονται και το Universal Packer Router (UPR), το Global File Services (GFS) και το Virtual Channel Router (VCR).

Το UPR είναι υπεύθυνο για την επικοινωνία όταν δεν υπάρχει φυσικός σύνδεσμος ανάμεσα στους επεξεργαστές. Το GFS δίνει δυνατότητα σε επεξεργαστές που δεν είναι ρίζα να χρησιμοποιούν λειτουργίες I/O. Το VCR δίνει τη δυνατότητα στις διεργασίες να επικοινωνούν μεταξύ τους με κανάλια και είναι αυτό που δίνει τη δυνατότητα για εικονικά κανάλια πάνω από ένα φυσικό κανάλι.

3.3.18 Ο Server

Ο server είναι μια εφαρμογή για Windows που τρέχει στο host και έχει δύο σημαντικές λειτουργίες:

- φορτώνει το πρόγραμμα στο σύστημα
- εκτελεί είσοδο-έξοδο και άλλες λειτουργίες στον host παίρνοντας εντολές από το δίκτυο των επεξεργαστών

Λόγω της δεύτερης λειτουργίας, ο server συνήθως τρέχει στον host συνεχώς και όλη την ώρα που τρέχει η εφαρμογή στο board. Οι λεπτομέρειες αυτών των εργασιών που κάνει ο server είναι αδιαφανής. Ο προγραμματιστής απλά καλεί τις συναρτήσεις της C όπως πχ την printf. Η run-time βιβλιοθήκη μετατρέπει αυτές τις κλήσεις σε κατάλληλες εντολές στον server.

3.3.19 Αυτοδύναμες (stand-alone) εφαρμογές

Είναι εφικτό να δημιουργηθούν εφαρμογές οι οποίες δεν χρησιμοποιούν καθόλου τον sever. Ένα απλό παράδειγμα είναι μια εφαρμογή η οποία κρατείται στην ROM μνήμη και φορτώνεται αυτόματα όταν ξεκινήσει ο επεξεργαστής. Ένα άλλο παράδειγμα είναι μια εφαρμογή που φορτώνεται και μιλάει άμεσα με κώδικα γραμμένο στον host. Τέτοιες εφαρμογές έχουν πλήρη έλεγχο για την επικοινωνία με το host αν υπάρχει κάποια.

Όταν ο προγραμματιστής φτιάχνει μια αυτοδύναμη εφαρμογή θα πρέπει να έχει υπόψη ότι:

- Όλες οι διεργασίες πρέπει να συνδεθούν με μια stand-alone βιβλιοθήκη
- Πρέπει ο ρυθμιστής να εκτελεστεί με την παράμετρο -A.

3.3.20 Απόδοση

Το diamond έχει σχεδιασθεί έτσι ώστε να επιτρέπει στον προγραμματιστή να γράψει εφαρμογές υψηλής απόδοσης με τον μικρότερο δυνατό κόπο. Ο μικροπυρήνας μεριμνά ώστε η διαθέσιμη CPU να χρησιμοποιείται αποδοτικά. Όταν ένα νήμα περιμένει για κάτι (επικοινωνία με τον host η με έναν άλλον

επεξεργαστή για παράδειγμα), ο πυρήνας βάζει το νήμα αυτό σε αναμονή και δίνει τον έλεγχο στο επόμενο στη σειρά νήμα το οποίο είναι έτοιμο για εκτέλεση. Ένα νήμα σε αναμονή δεν χρησιμοποιεί κύκλους του επεξεργαστή. Το νήμα θα ξυπνήσει και θα ετοιμαστεί για εκτέλεση όταν εκτελεστεί το συμβάν που περιμένει.

3.3.21 Ακολουθιακά Προγράμματα

Η διαδικασία για την μετατροπή ενός η περισσότερων αρχείων κώδικα σε μια έτοιμη εφαρμογή απαιτεί τρία βήματα:

- Μεταγλώττιση των αρχείων κώδικα σε object αρχεία.
- Σύνδεση των αρχείων αυτών με βιβλιοθήκες για τη δημιουργία διεργασιών
- Ρύθμιση των διεργασιών για την ένωσή τους και τη δημιουργία της εκτελέσιμης εφαρμογής

3.3.22 Μεταγλώττιση

Κάθε αρχείο κώδικα μεταγλωττίζεται σε ένα object αρχείο που περιέχει τις εντολές σε γλώσσα του επεξεργαστή. Η μεταγλώττιση γίνεται με την ακόλουθη εντολή:
3L c source-file

Αυτή η εντολή καλεί τον μεταγλωττιστή για την C της Texas Instruments με τις κατάλληλες παραμέτρους. Το αρχείο κώδικα θα πρέπει να έχει την επέκταση ".c" και το όνομα του αρχείου θα πρέπει να γραφθεί με την επέκτασή του. Έτσι για να μεταγλωττίσει κανείς το hello.c θα πρέπει να δώσει την εξής εντολή:
3L c hello.c

Αν το πρόγραμμα δεν περιέχει κάποιο λάθος, θα δημιουργηθεί ένα object αρχείο με όνομα hello.obj. Αν ο μεταγλωττιστής συναντήσει λάθη στο πρόγραμμα, εμφανίζει διαγνωστικά μηνύματα στην οθόνη.

3.3.23 Παράμετροι και Επιλογές του Μεταγλωττιστή

Εκτός από τις επιλογές και παραμέτρους που η εντολή 3L c χρησιμοποιεί αυτόματα, ο μεταγλωττιστής διαθέτει και άλλες επιλογές. Αυτές δίνονται στην εντολή μεταγλώττισης, όπως φαίνεται παρακάτω:
3L c hello.c -dDEBUG -k

Αυτή η εντολή θα μεταγλωττίσει το πρόγραμμα hello.c με την επιλογή DEBUG. Η έξοδος του μεταγλωττιστή σε αρχείο με εντολές γλώσσας μηχανής θα κρατηθεί και μετά το τέλος της μεταγλώττισης, όπου θα σβηνόταν σε αντίθετη περίπτωση.

3.3.24 Σύνδεση (Linking)

Όταν ένα πρόγραμμα μεταγλωττιστεί σε ένα η περισσότερα αρχεία object, το επόμενο βήμα είναι να συνδεθεί με τις εξωτερικές συναρτήσεις που χρησιμοποιεί. Αυτό είναι απαραίτητο πριν τρέξει. Τέτοιες συναρτήσεις είναι για παράδειγμα συναρτήσεις από την run-time βιβλιοθήκη όπως η printf. Ο συνδέτης της Texas Instruments το κάνει αυτό και η έξοδός του είναι ένα αρχείο διεργασίας.

Οι διεργασίες του diamond είναι επανατοποθετήσημες στη μνήμη, αυτό σημαίνει ότι οι διάφορες περιοχές μνήμης για κώδικα και δεδομένα που περιέχουν δεν έχουν συγκεκριμένες θέσεις στη μνήμη. Η ανάθεση μνήμης θα γίνει δυναμικά σε επόμενα βήματα. Γι' αυτόν τον λόγο δεν πρέπει να διαθέτουμε στον συνδέτη την πληροφορία για τα τμήματα μνήμης όπως σε αντίθετη περίπτωση θα έπρεπε να είχε γίνει αν χρησιμοποιούσαμε τα εργαλεία της TI μόνον.

Παρακάτω φαίνονται οι βασικές λειτουργίες του συνδέτη. Μπορούμε να συνδέσουμε οσαδήποτε αρχεία object για τη δημιουργία μιας διεργασίας με την παρακάτω εντολή:
3L t object-file object-file ñ
Για παράδειγμα: 3L t hello

Η 3L t είναι μια εντολή που καλεί τον συνδέτη μια τον πιο σύνηθες τρόπο. Η επέκταση δεν είναι απαραίτητη κατά το κάλεσμα αυτής της εντολής καθώς υποθέτει την επέκταση .obj Το όνομα του πρώτου θα χρησιμοποιείται για να κατασκευαστεί το όνομα της διεργασίας και θα παίρνει επέκταση .tsk.

Έτσι στο παραπάνω παράδειγμα, το hello.obj θα συνδεθεί με τις απαραίτητες λειτουργίες από τη βιβλιοθήκη, και θα δημιουργηθεί ένα αρχείο διεργασίας hello.tsk.

Για παράδειγμα εάν έχουμε δύο αρχεία C με κώδικα, main.c και fns.c για να δημιουργήσουμε τη διεργασία main.tsk θα πρέπει να ακολουθήσουμε τις παρακάτω εντολές:
3L c main.c
3L c fns.c
3L t main fns

Είναι δυνατόν να ενσωματωθούν και παράμετροι του συνδέτη στην εντολή 3L t. Για παράδειγμα η παρακάτω εντολή θα δημιουργήσει ένα αρχείο χάρτη του συνδέτη στο mail.lis
3L t main fns -m main.lis

3.3.25 Ρύθμιση Εφαρμογής

Αφού έχουν συνδεθεί τα απαραίτητα στοιχεία για τη δημιουργία μιας διεργασίας, θα πρέπει να ρυθμισθεί. Αυτή η διαδικασία είναι εκείνη όπου αποφασίζεται πώς τα διάφορα μέρη του προγράμματος θα αναχθούν στη μνήμη, και δημιουργεί το αρχείο της εφαρμογής ενσωματώνοντας τον μικροπυρήνα της Diamond, τις διεργασίες που χρειάζονται για τις επικοινωνίες και λοιπά. Το πρόγραμμα που τα κάνει όλα αυτά ονομάζεται ρυθμιστής (configurer). Ο σκοπός αυτού του προγράμματος είναι να ενώσει μαζί πολλές διεργασίες έτοιμες να φορτωθούν σε ένα δίκτυο επεξεργαστών, αλλά ο προγραμματιστής το χρησιμοποιεί για πιο απλούς λόγους.

3.3.26 Καλώντας τον Ρυθμιστή

Το πρώτα πράγμα που πρέπει να γίνει πριν ρυθμιστεί μια εφαρμογή είναι να γραφθεί ένα αρχείο κειμένου. Αυτό το αρχείο περιέχει πληροφορίες για τις διεργασίες και τους επεξεργαστές που θα χρειαστεί η εφαρμογή.

Αρχικά θα υποθέσουμε μια εφαρμογή που έχει μόνο μια διεργασία που τρέχει σε έναν μόνο επεξεργαστή, έτσι το ρυθμιστικό αρχείο θα είναι όσο πιο απλό γίνεται. Για παράδειγμα το παρακάτω ρυθμιστικό αρχείο (myprog.cfg) που αποτελείται από τρεις γραμμές θα μπορούσε να χρησιμοποιηθεί για να δημιουργήσει μια εφαρμογή από την διεργασία myprog.tsk

```
PROCESSOR root TYPE=MyBoard (1)
TASK myprog DATA=? (2)
PLACE myprog root (3)
```

- Η εντολή PROCESSOR μας δίνει το υλικό το οποίο πρόκειται να χρησιμοποιήσουμε. Καθορίζει το όνομα του επεξεργαστή, που στην περίπτωσή μας θα πρέπει να είναι ο επεξεργαστής ρίζα, καθώς προσδιορίζει και τον τύπο του. Ο τύπος δείχνει όχι μόνο ότι πρόκειται για έναν DSP επεξεργαστή, αλλά δείχνει ακόμη ποιόν ακριβώς επεξεργαστή χρησιμοποιούμε. Στη θέση του MyBoard τοποθετείται ο κατάλληλος προσδιορισμός του επεξεργαστή, όπου στο δικό μας Board είναι SMT365 (C6416). Είναι σημαντικό ο τύπος να είναι σωστός, διότι κάθε διαφορετικός επεξεργαστής έχει διαφορετικούς τρόπους αρχιτεκτονικής μνήμης, επικοινωνιών και τρόπους διαχείρισης τους.

- Η εντολή TASK καθορίζει το όνομα της διεργασίας. Ο ρυθμιστής θα χρησιμοποιήσει αυτό το όνομα για να βρει το αρχείο της διεργασίας. Η παράμετρος DATA=? Λέει στον ρυθμιστή να αναθέσει μια συνεχής περιοχή στη μνήμη για την στοίβα και το heap της εφαρμογής. Αυτή η

περιοχή θα είναι η μεγαλύτερη περιοχή που θα έχει μείνει αφού ανατεθούν οι περιοχές για όλα τα υπόλοιπα (εντολές, στατικά δεδομένα για παράδειγμα). Εάν παραληφθεί το DATA=? Ο ρυθμιστής θα ενεργήσει με τον ίδιο τρόπο αλλά θα εμφανίσει ένα μήνυμα προειδοποίησης.

- Η τελευταία γραμμή του ρυθμιστικού αρχείου είναι μια εντολή PLACE, η οποία λέει στον ρυθμιστή να τοποθετήσει την διεργασία myprog στον επεξεργαστή ρίζα.

Τώρα που έχει δημιουργηθεί το ρυθμιστικό αρχείο μπορούμε να δημιουργήσουμε το αρχείο της εφαρμογής καλώντας τον ρυθμιστή όπως θα δείξουμε παρακάτω, όπου οι επεκτάσεις των αρχείων θα πρέπει να καθορίζονται.
3L A configuration-file application-file
Ο ρυθμιστής μπορεί να κληθεί και με μόνο μία παράμετρο.
3L A file
Αυτό έχει την ίδια ισχύ με την εντολή
3L A file.cfg file.app
Το πρόγραμμα myprog που περιγράφθηκε προηγουμένως θα μπορούσε να ρυθμιστεί με την παρακάτω εντολή
3L A myprog.cfg myprog.app

Ο ρυθμιστής θα πάρει το αρχείο της διεργασίας και αυτόματα θα αναθέσει μνήμα για όλα τα στοιχεία της διεργασίας. Αν είναι επιθυμητό να δούμε που αναθέτει ο ρυθμιστής τα διάφορα στοιχεία, μπορούμε να κάνουμε τον ρυθμιστή να στέλνει λεπτομερή αναφορά στην έξοδο με την επιλογή -L όπως παρακάτω:
3L A myprog.cfg myprog.app -L myprog.lis

Μπορούμε να έχουμε πλήρη έλεγχο των αναθέσεων στη μνήμη που κάνει ο ρυθμιστής, αλλά είναι προτεινόμενο τις αναθέσεις αυτές να τις αναλαμβάνει αυτόματα ο ρυθμιστής κατά τη διάρκεια της δημιουργίας της εφαρμογής.

3.3.27 Εκτέλεση

Οι εφαρμογές συνήθως φορτώνονται σε DSP επεξεργαστές και εκτελούνται χρησιμοποιώντας το Server πρόγραμμα που λέγεται WS3L. Αυτό είναι ένα κανονικό Windows πρόγραμμα που τρέχει στον υπολογιστή host. Αφού φορτωθεί το πρόγραμμα στους DSP, συνήθως παραμένει ενεργό καθ' όλη τη διάρκεια ζωής του προγράμματος. Η C run-time βιβλιοθήκη στέλνει εντολές στον server όταν χρειάζεται να εκτελεστεί μια λειτουργία στον host, όπως για παράδειγμα να διαβαστούν πληροφορίες από τον δίσκο, να εμφανισθούν πληροφορίες στην οθόνη ή κάτι άλλο. Ο server στέλνει τα αποτελέσματα αυτών των λειτουργιών πίσω στους DSP.

Η εκκίνηση του server μπορεί να γίνει με τέσσερις τρόπους:

- τρέχοντας το WS3L.EXE ή μια συντόμευση σε αυτό
- τρέχοντας το .app αρχείο που έχει δημιουργηθεί
- με drag&drop του αρχείου .app στο παράθυρο του server.
- δίνοντας τις εντολές από γραμμή εντολών 3L X ή 3L X όνομα-αρχείου

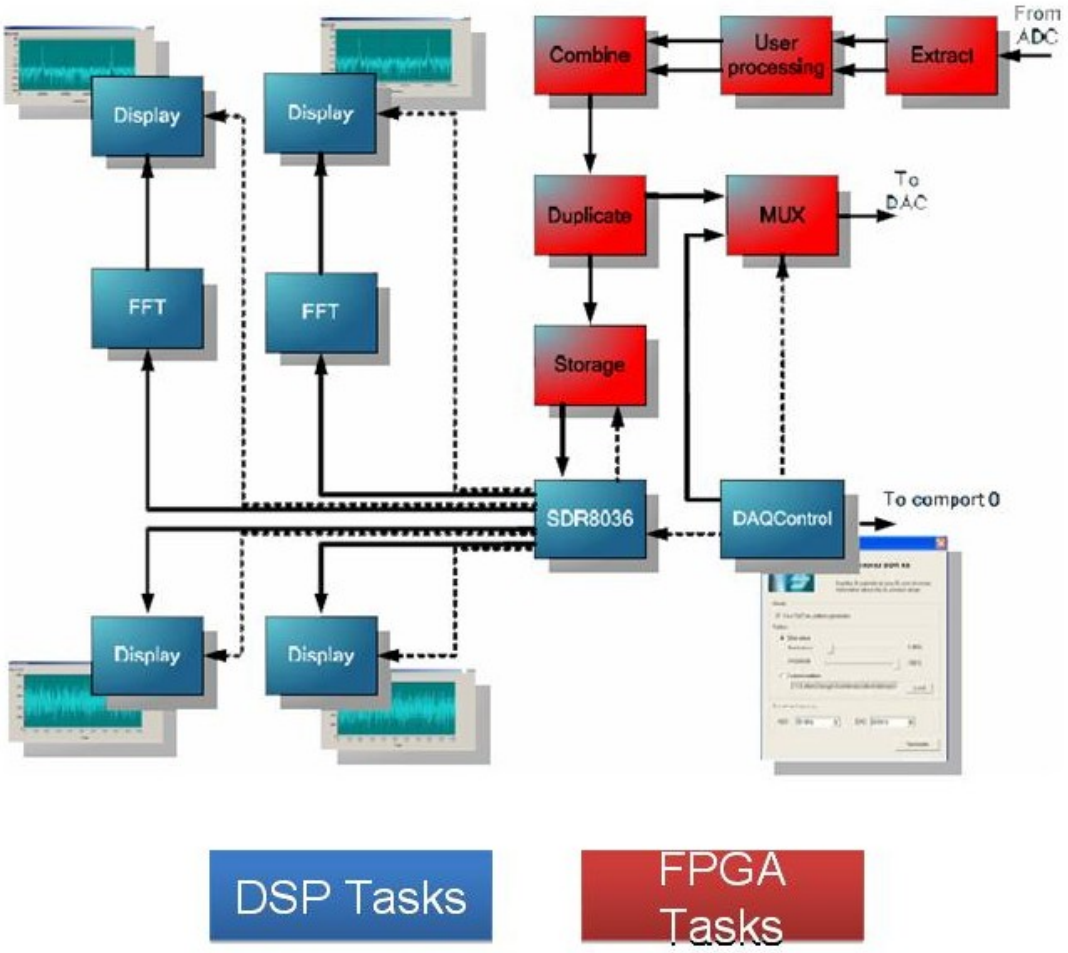
Το όνομα αρχείου θα πρέπει να είναι το όνομα του αρχείου της εφαρμογής που δημιουργήθηκε από τον ρυθμιστή. Αν δεν δοθεί επέκταση, προστίθεται η επέκταση .app.

Όλες αυτές οι εντολές θα φέρουν το παράθυρο του server. Αν δεν έχει χρησιμοποιηθεί ένα αρχείο .app για να φέρει το παράθυρο, από το file μενού μπορεί να εκτελεστεί μια εφαρμογή.

Πριν επιτρέψουμε στον server να φορτώσει την εφαρμογή στο δίκτυο των DSP και την εκτελέσει, θα πρέπει να έχουμε πει στον server πιο DSP Board χρησιμοποιούμε. Αυτό γίνεται από το μενού Board/Select. Εάν δεν έχει επιλεγεί κάποιο board, το server πρόγραμμα θα διαλέξει αυτόματα το πρώτο που θα βρει και έπειτα θα εκτελέσει την εφαρμογή.

Κεφάλαιο 4

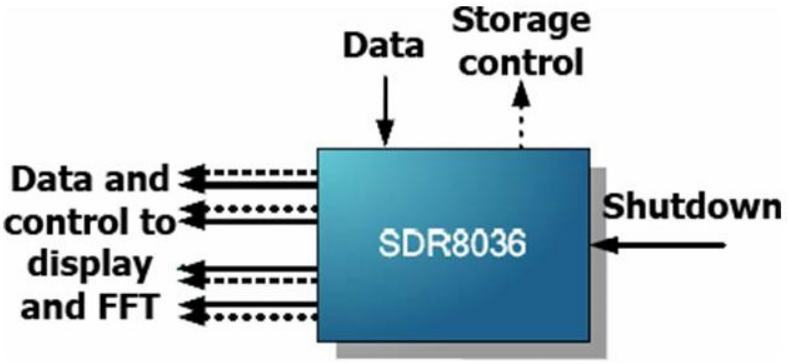
Κώδικας της Εφαρμογής



- Υλοποίηση του κυρίως προγράμματος
- Το configuration αρχείο
- Το MAKEFILE αρχείο
- Η Υλοποίηση του FFT
- Η Διεργασία εμφάνισης στην οθόνη
- Η Διεργασία ελέγχου των παραμέτρων
- Υλοποίηση με ανάλυση 1024-point FFT
- Υλοποίηση με ανάλυση 2048-point FFT
- Υλοποίηση με ανάλυση 4096-point FFT
- Οι διεργασίες του FPGA
- Συντελεστές των Φίλτρων

4.1 Υλοποίηση του κυρίως προγράμματος

Πηγαίος Κώδικας του SDR8036



Σχήματα 4.1: Η κυρίως διεργασία

Η διεργασία αυτή είναι η κυρίως διεργασία του αναλυτή φάσματος. Αυτή η διεργασία καλεί κάποια tasks, εκτελεί κάποιες λειτουργίες και αποτελεί τον διαιτητή όλης της εφαρμογής. Καλεί τις διεργασίες για την απεικόνιση στην

οθόνη και την διεργασία για τον έλεγχο των παραμέτρων της εφαρμογής. Αναλαμβάνει την λήψη των δεδομένων από την διεργασία του FPGA που λαμβάνει τα δείγματα από τον A/D μετατροπέα και τα σερβίρει στις κατάλληλες διεργασίες. Είναι υπεύθυνη για την αξιόπιστη και απρόσκοπτη λήψη των δεδομένων με τη χρήση σεμαφόρων. Τέλος είναι εκείνη που σταματάει τις διεργασίες κατά το κλείσιμο της εφαρμογής.

```
#include "coefs.h"

#define M                               8
#define LOG2N           9
#define SAMPLES         ((1 << LOG2N))
        // samples per channel
#define BYTES           ((SAMPLES)*2*sizeof(short))
#define MAXIMUM                 16000
```

```
#include <stdio.h>
#include <stdlib.h>
#include <thread.h>
#include <par.h>
#include <sema.h>
#include <alt.h>
#include <chan.h>
#include <string.h>
#include <math.h>
#include "dsp\display\display.h"                                    10
#include "sdr8036.h"

INPUT_PORT ( 0, SHUTDOWN)
    // a command to shutdown this tasks
INPUT_PORT ( 1, STORAGE)
    // data captured by the storage task
INPUT_PORT ( 2, FREQ_SETTINGS)
    // user selected ADC and DAC sampling frequencies
INPUT_PORT ( 3, FROMDAQ)
    // receive SMT370 configuration update requests                20

OUTPUT_PORT( 0, RAW1)
    // raw data display for channel 1
OUTPUT_PORT( 1, RAW1_CONFIG)
    // configuration control channel for display
OUTPUT_PORT( 2, RAW2)
    // raw data display for channel 2
OUTPUT_PORT( 3, RAW2_CONFIG)
    // configuration control channel for display
OUTPUT_PORT( 4, STORAGE_CMD)                                       30
```

```
    // command to storage task to capture data
OUTPUT_PORT( 5, TOVFFT)
    // channel 1 data to the VFFT task
OUTPUT_PORT( 6, VFFT_CONFIG)
    // configuration control channel for fft display
OUTPUT_PORT( 7, TOFFT2)
    // channel 2 data to the FFT task
OUTPUT_PORT( 8, FFT2_CONFIG)
    // configuration control channel for fft display
OUTPUT_PORT( 9, TODAQ)                                                  40
    // send SMT370 configuration update request

static unsigned int     FsAdc=10, FsDac=10;
static float            *stream1;
static float            *stream2;
static volatile int      shutdown = 0;

SEMA sema_read;
    // protect access to the SMT370 configuration
SEMA sema_first_dac_config;                                             50
    // indicate that the DAC has been configured at least once

void * GetMem(unsigned int size)
{
  void * P = memalign(128, size);
  if (P==0) {
    printf("Cannot allocate buffers\n");
    exit(1);
  }
  return P;                                                             60
}

void InitDisplay( char *idtag, char *title, char *xtitle,
                                  char *xunit, char *ytitle, char *yunit,
                  double pmin, double psize, double xmin,
                  double xsize, double ymin, double ysize,
                  CHAN *C, unsigned int log2n)
{
  Display_init  di;
  strcpy(di.idtag, idtag);                                              70
  di.log2n = log2n;
  strcpy(di.graph_title,  title);
  strcpy(di.x_title,      xtitle);
  strcpy(di.x_unit,       xunit);
  strcpy(di.y_title,      ytitle);
  strcpy(di.y_unit,       yunit);
  di.phys_min             = pmin;
  di.phys_ysize           = psize;
  di.user_xmin            = xmin;
```

```
  di.user_xsize          = xsize;                                          80
  di.user_ymin           = ymin;
  di.user_ysize          = ysize;
  chan_out_word(CMD_DISPLAY_INIT, C);
  chan_out_message(sizeof(Display_init), &di, C);
}

void SetDisplay(double xmin, double xsize,
                            double ymin, double ysize, CHAN *C)
{
  Display_setuser di;                                                      90
  di.user_xmin     = xmin;
  di.user_xsize    = xsize;
  di.user_ymin     = ymin;
  di.user_ysize    = ysize;
  chan_out_word(CMD_DISPLAY_SETUSER, C);
  chan_out_message(sizeof(Display_setuser), &di, C);
}

void SetupGraphs(unsigned int AdcFreq, unsigned int DacFreq)
{                                                                          100
  SetDisplay(0, SAMPLES*1000/AdcFreq, 0, 16*1024, &RAW1_CONFIG);
//  SetDisplay(0, SAMPLES*1000/AdcFreq, 0, 16*1024, &RAW2_CONFIG);
  SetDisplay(0, (FsAdc)*1000, -150, 200, &VFFT_CONFIG);
//  SetDisplay(0, (FsAdc)*1000, -150, 200, &FFT2_CONFIG);
}

// When the user selects a sampling frequency for the ADC and DAC,
// we need to change the scales of the displays
void monitor_thread(void* a)
{                                                                          110
  unsigned int sd;
  for (;;) {
    int n = alt_wait(2, &FREQ_SETTINGS, &SHUTDOWN);

    if (n) {
      chan_in_word(&sd, &SHUTDOWN);
      shutdown = sd;
      break;
    }
                                                                           120
    chan_in_word(&FsAdc, &FREQ_SETTINGS);
    chan_in_word(&FsDac, &FREQ_SETTINGS);
    SetupGraphs(FsAdc, FsDac);
  }
}


THREAD_HANDLE Startup(void (*Thread)(void *))
```

```
{
   THREAD_HANDLE H = thread_new(Thread, 1000, 0);
   if (!H) {
      par_printf("Could not start threads\n");
      exit(1);
   }
   return H;
}

// This thread is used to protect access to the SMT370
// configuration.
void protect_read_thread(void* a)
{
  int x;
  for (;;) {
    // wait for pending change to the SMT370 settings
    chan_in_word(&x, &FROMDAQ);
    // stop the main task from accessing the SMT370
    sema_wait(&sema_read);
    // instruct DACControl task to proceed with SMT370 update
    chan_out_word(0, &TODAQ);
    // wait for update to complete
    chan_in_word(&x, &FROMDAQ);
    // allow main task to access SMT370
    sema_signal(&sema_read);
    sema_signal(&sema_first_dac_config);
  }
}

void main()
{
  int     n,i=0,s,sw=0;
  short *data;
  THREAD_HANDLE hThread;

  printf("3L Diamond : SDR8036\n");

  data    = GetMem(BYTES);
  stream1 = GetMem(sizeof(float)*SAMPLES);
  stream2 = GetMem(sizeof(float)*SAMPLES);

  if ((data==0) || (stream1==0) || (stream2==0) ){
    printf("Cannot allocate buffers\n");
    exit(1);
  }

  hThread = Startup(monitor_thread);

  sema_init(&sema_read,1);
```

```
  sema_init(&sema_first_dac_config, 0);

  Startup(protect_read_thread);                                          180

  InitDisplay("Sdr8036_time1", "Time domain - Channel 1",
                    "Time", "[ns]", "Amplitude", "[unit]",
              0,   16*1024, 0, SAMPLES*1000/FsAdc,
              0, 16*1024, &RAW1_CONFIG, LOG2N);

  InitDisplay("Sdr8036_vfft", "Channel 1 VFFT",
                    "Frequency", "[KHz]", "Amplitude", "[Db]",
             -50, 200, 0, (FsAdc)*1000/2, -150, 200,
             &VFFT_CONFIG, (LOG2N+M/2)-1);                               190

  // wait here for DAC to be configured for the first time
  sema_wait(&sema_first_dac_config);

  while (!shutdown) {
    short *pS = data;

    sema_wait(&sema_read);

    // instruct the storage task to capture data. We do this             200
    // by sending the number of words to capture
    chan_reset(&STORAGE);
    chan_out_word(BYTES/sizeof(int), &STORAGE_CMD);

    // read the data from storage
    chan_in_message(BYTES, data, &STORAGE);

    sema_signal(&sema_read);

        sw=0;                                                            210
        switch(sw) {
            case 0:
        // EISODOS APO ADC
            for (n=0; n<SAMPLES; n++) {
              stream1[n] = *pS++;
              stream2[n] = *pS++;
//                  printf("\n%f",stream1[n]);
//                  printf("\n0");
            }
            break;                                                       220
        case 1:
    // KROUSTIKI
            i = (i++)%M;
            for (s=0;s<SAMPLES;s++) stream1[s] = stream2[s] = 8000;
                if  (i==1) stream1[0] = stream2[0] = 15000;
                break;
```

```
            case 2:
        // TETRAGONIKOS PALMOS
                for (s=0          ;  s<1*SAMPLES/4;s++)
                    stream1[s] = stream2[s] = 12000;                    230
                for (s=SAMPLES/4 ; s<2*SAMPLES/4;s++)
                    stream1[s] = stream2[s] = 4000;
                for (s=SAMPLES/2 ; s<3*SAMPLES/4;s++)
                    stream1[s] = stream2[s] = 12000;
                for (s=3*SAMPLES/4; s<4*SAMPLES/4;s++)
                    stream1[s] = stream2[s] = 4000;
                break;
        }

    chan_out_message(SAMPLES*sizeof(float), stream1, &RAW1);            240
//    chan_out_message(SAMPLES*sizeof(float), stream2, &RAW2);
    chan_out_message(SAMPLES*sizeof(float), stream1, &TOVFFT);
    //chan_out_message(SAMPLES*sizeof(float), stream2, &TOFFT2);
  };

  // send close command to graphs
  chan_out_word(CMD_DISPLAY_SHUTDOWN, &RAW1_CONFIG);
//  chan_out_word(CMD_DISPLAY_SHUTDOWN, &RAW2_CONFIG);
  chan_out_word(CMD_DISPLAY_SHUTDOWN, &VFFT_CONFIG);
//  chan_out_word(CMD_DISPLAY_SHUTDOWN, &FFT2_CONFIG);                   250

  thread_wait(hThread);

  free(data);
  free(stream1);
  free(stream2);
}
```

4.2 Το configuration αρχείο

Το configuration αρχείο μας ρυθμίζει τους ενεργούς επεξεργαστές με την λέξη PROCESSOR, τίς απαιτούμενες συνδεσμολογίες με την λέξη WIRE, η κάθε διεργασία πόσες πόρτες εισόδου και εξόδου έχει με τη λέξη TASK, τις συνδέσεις μεταξύ των διεργασιών με τη λέξη CONNECT, και το ποιές διεργασίες θα τρέχουν σε ποιόν επεξεργαστή με τη λέξη PLACE.

```
! Declare the processors in the system
PROCESSOR Root SMT365_8_1
PROCESSOR F FPGA      ATTACH=Root

! WIRE statements describe physical
! connections between processors
WIRE W1 Root[SDB:0] F[SDB_DEVICE:0]
WIRE ? Root[SDB:1] F[SDB_DEVICE:1]
WIRE ? Root[CP:1] F[CP_DEVICE:1]
                                                                    10
! The DSP tasks in the system
TASK sdr8036          INS=4 OUTS=10 DATA=500K
TASK DAQControl    INS=1 OUTS=6 DATA=500K
   file = "dsp\DAQControl\DAQControl.tsk"
TASK display          INS=2 OUTS=0 DATA=200K
   file = "dsp\display\display.tsk"
!TASK display2        INS=2 OUTS=0 DATA=200K
   file = "dsp\display\display.tsk"
TASK fftdisplay       INS=2 OUTS=0 DATA=200K
   file = "dsp\display\display.tsk"                                 20
!TASK fftdisplay2     INS=2 OUTS=0 DATA=200K
   file = "dsp\display\display.tsk"
TASK fft              INS=1 OUTS=1 DATA=200K
   file = "dsp\tifft\tifft.tsk"
!TASK ifft            INS=1 OUTS=1 DATA=200K
   file = "dsp\tiifft\tiifft.tsk"
TASK chan                     INS=3 OUTS=3 DATA=200K
   file = "dsp\chan\chan.tsk"
! The FPGA tasks in the system
TASK USERPROC    INS=2 OUTS=2                                       30
   file = "fpga\userproc\userproc.fcd"
TASK extract          INS=1 OUTS=2
   file = "fpga\extract\Extract.fcd"
TASK combine          INS=2 OUTS=1
   file = "fpga\combine\Combine.fcd"
TASK duplicate        INS=1 OUTS=2
   file = "fpga\duplicate\duplicate.fcd"
TASK mux              INS=3 OUTS=1
   file = "fpga\mux\Mux.fcd"
TASK storage          INS=2 OUTS=1                                  40
```

```
    file = "fpga\storage\Storage.fcd"

CONNECT ? sdr8036  [9]      DAQControl [0]
CONNECT ? DAQControl [5] sdr8036   [3]

! How the tasks are connected
CONNECT ? sdr8036  [0]      display    [0]
CONNECT ? sdr8036  [1]      display    [1]
!CONNECT ? sdr8036  [2]      display2   [0]
!CONNECT ? sdr8036  [3]      display2   [1]
CONNECT ? sdr8036  [4]      Storage    [1]
    ! storage command
CONNECT ? sdr8036  [5]      chan       [0]
!CONNECT ? sdr8036  [5]      fft        [0]
CONNECT ? sdr8036  [6]      fftdisplay [1]
!CONNECT ? sdr8036  [7]      fft2       [0]
!CONNECT ? sdr8036  [8]      fftdisplay2[1]

CONNECT ? DAQControl [1] Mux       [2]
    ! pattern for DAC
CONNECT ? DAQControl [2] Mux       [0]
    ! mux control
CONNECT ? DAQControl [3] sdr8036   [2]
    ! current sampling frequencies selected
CONNECT ? DAQControl [4] sdr8036   [0]
    ! shutdown command

CONNECT ? Extract  [0]      USERPROC [0]
CONNECT ? Extract  [1]      USERPROC [1]

CONNECT ? USERPROC [0] Combine  [0]
CONNECT ? USERPROC [1] Combine  [1]

CONNECT ? combine [0]       duplicate  [0]

CONNECT ? duplicate [0]     mux        [1]
    ! DAC direct DATA to mux
CONNECT ? duplicate [1]     storage    [0]
    ! duplicated DATA from fpga

CONNECT C1 Storage [0]      sdr8036    [1]
    ! storage DATA from fpga

CONNECT ? chan              [1]        fft              [0]
CONNECT ? fft               [0]        chan             [1]
CONNECT ? chan      [0]     fftdisplay [0]


! PLACE the tasks on the processors
```

```
PLACE  sdr8036          Root                                    90
PLACE  DAQControl       Root
PLACE  display          Root
!PLACE  display2         Root
PLACE  fft              Root
!PLACE  ifft             Root
PLACE  fftdisplay       Root
!PLACE  fftdisplay2      Root
PLACE  chan                 Root
PLACE  USERPROC         F
PLACE  Extract          F                                       100
PLACE  Combine          F
PLACE  Duplicate        F
PLACE  Mux              F
PLACE  Storage          F

! Special considerations
PLACE  C1               W1
    ! We want the connection C1 to be placed on
    ! WIRE W1 because the SDB interface is
    ! slightly faster than the comport interface,                110
    ! and we want to use it to read the
  ! DATA from the storage TASK

! Connecting tasks to connectors
CONNECT ? F[SDB:0]       Extract [0]
CONNECT ? DAQControl [0] Root [CP:0]
    ! control task uses a comport to configure the DAC
CONNECT ? Mux [0]        F [SDB:1]
    ! patern to the DAC
```

4.3 Το MAKEFILE αρχείο

Το makefile αρχείο μας επιτρέπει την αυτόματη μεταγλώττιση-σύνδεση και δημιουργία εκτελέσιμου με μία batch εντολή.

```
sdr8036.app: sdr8036.tsk sdr8036.cfg MAKEFILE
    3L a sdr8036 -v

sdr8036.tsk: sdr8036.obj MAKEFILE
    3L t sdr8036

sdr8036.obj: sdr8036.c MAKEFILE
    3L c sdr8036.c

clean :                                                         10
```

```
del sdr8036.app
del sdr8036.tsk
del sdr8036.obj
```

4.4 Η Υλοποίηση του FFT

Σχήματα 4.2: Η διεργασία που υλοποιεί τον FFT

Η διεργασία αυτή υλοποιεί 512 σημείων Fast Fourier Transform με αποδοτικό αλγόριθμο υλοποιημένο από την εταιρία TI που τον διαθέτει ελεύθερα για χρήση.

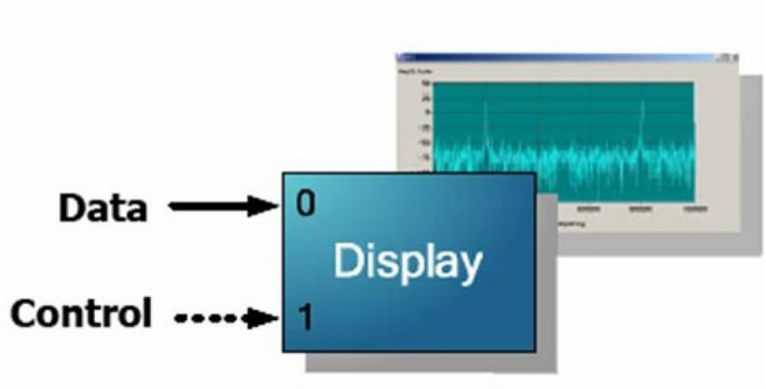
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <chan.h>
#include "../../sdr8036.h"

#include "dsp_fft32x32.h"

#define PI (3.141592654)                                    10
#define NN (512)

#pragma DATA_ALIGN(x, 8)
#pragma DATA_ALIGN(xx, 8)
#pragma DATA_ALIGN(y, 8)
#pragma DATA_ALIGN(w, 8)
#pragma DATA_ALIGN(yy, 8)
#pragma DATA_ALIGN(mag, 8)

INPUT_PORT (0, MAININPUT)                                   20
OUTPUT_PORT (0, MAINOUTPUT)

int x[2*NN];
int w[2*NN];
int xx[2*NN];
```

```
int y[2*NN];
float yy[2*NN];
float mag[NN];

static int d2i(double d)
{
    if (d >= 2147483647.0) return (int)0x7FFFFFFF;
    if (d <= -2147483648.0) return (int)0x80000000;
    return (int)d;
}


//GEN_TWIDDLE -- Generate twiddle factors for fft32x32().
//
//USAGE
//This routine is called as follows:
//
//int gen_twiddle_fft32x32(short *w, int n, double scale)
//
//int    *w     Pointer to twiddle-factor array
//int    n      Size of FFT
//double scale Scale factor to apply to values.
//
//The routine will generate the twiddle-factors directly into the
//array you specify. The array needs to be approximately 2*N
//elements long. (The actual size, which is slightly smaller, is
//returned by the function.)

int gen_twiddle_fft32x32(int *w, int n, double scale)
{
    int i, j, k, s=0, t;

    for (j = 1, k = 0; j < n >> 2; j = j << 2, s++)
    {
        for (i = t=0; i < n >> 2; i += j, t++)
        {
            w[k + 5] = d2i(scale * cos(6.0 * PI * i / n));
            w[k + 4] = d2i(scale * sin(6.0 * PI * i / n));

            w[k + 3] = d2i(scale * cos(4.0 * PI * i / n));
            w[k + 2] = d2i(scale * sin(4.0 * PI * i / n));

            w[k + 1] = d2i(scale * cos(2.0 * PI * i / n));
            w[k + 0] = d2i(scale * sin(2.0 * PI * i / n));

            k += 6;
        }
    }
```

```
    return k;
}


//RIVISION HISTORY
//30-Jun-2002 Initial revision                                          80
//31-Dec-2002 Improved benchmarking
//
//USAGE
//void main ();
//This program is to show how to use DSP_fft32x32()
//when data are in L2SRAM.
//
//DESCRIPTION
//First, function call overhead for timer functions is measured.
//Then, the execution cycles for the target function is measured.       90
//Finally, the timer function call overhead is excluded from the
//execution cycles measured.
//The cycle count measured will include L1P/D miss overhead as
//well as function call overhead.
//
//ASSUMPTIONS
//The benchmarking code assumes C64x, e.g., 1 count = 8 CPU cycles.
//
//     Copyright (c) 2003 Texas Instruments, Incorporated.
//                    All Rights Reserved.                            100

void * GetMem(unsigned int size)
{
  void * P = memalign(128, size);
  if (P==0) {
    printf("Cannot allocate buffers\n");
    exit(1);
  }
  return P;
}                                                                     110

void main(void)
{
    int i;
        float *data;
        data = GetMem(NN*sizeof(float));
            // Generate twiddle factors
        gen_twiddle_fft32x32(w, NN, 2147483647.);

        for(;;) {                                                     120
                chan_in_message(NN*sizeof(float), data, &MAININPUT);

            // Generate Q.31 input data
```

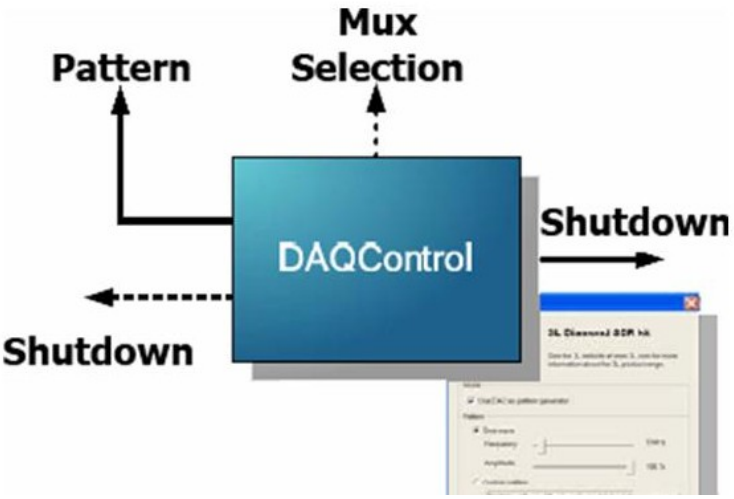
```
            for(i=0; i<NN; i++)
            {
                //printf("\n%d: %f ", i, *(data+i)/16384 );
                x[2*i] = xx[2*i] = 2147483647./16384/(NN) * (*(data+i));
                //printf("= %d ",x[2*i]);
                x[2*i+1] = xx[2*i+1] = 0;
            }

            DSP_fft32x32(w, NN, x, y);
            for(i=0;i<2*NN;i+=2) {
                yy[i]   = ((float)y[i])*(NN)/2147483647.;
                yy[i+1] = ((float)y[i+1])*(NN)/2147483647.;
                //printf("\nfourier%d: %f+j%f and ", i/2, yy[i],yy[i+1]);
                mag[i/2] = 20*log10(sqrt(yy[i]*yy[i]+yy[i+1]*yy[i+1]))+100;
                //printf("mag%d = %f ", i/2, yy[i/2]);
            }

            chan_out_message(NN*sizeof(float), mag, &MAINOUTPUT);
        }

}
```

4.5 Η Διεργασία εμφάνισης στην οθόνη



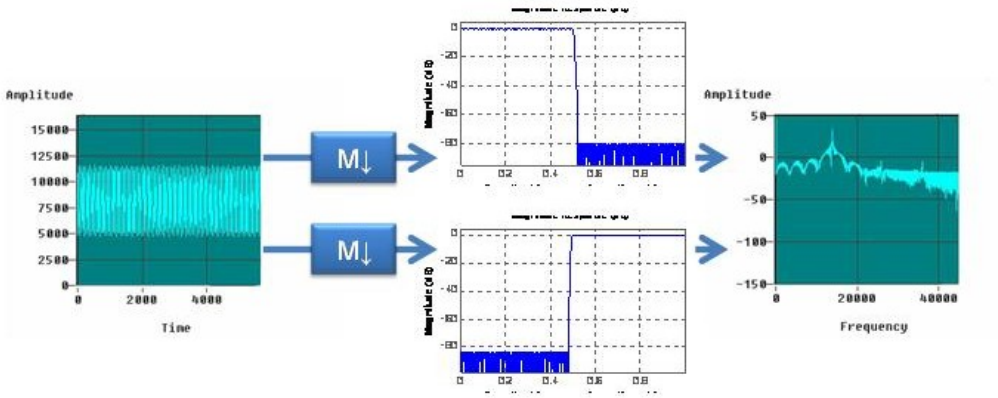
Αυτή η διεργασία εμφανίζει στην οθόνη το παράθυρο με τα αποτελέσματα.

4.6 Η Διεργασία ελέγχου των παραμέτρων



Αυτή η διεργασία παρέχει ένα User Interface για την μεταβολή των παραμέτρων του προγράμματος.

4.7 Υλοποίηση με ανάλυση 1024-point FFT



Σχήματα 4.3: Ανάλυση Φάσματος με ακρίβεια: 1024-point FFT

Εδώ βλέπουμε το αποτέλεσμα του αναλυτή φάσματος έχοντας ακρίβεια 1024 σημείων στον FFT του. Όσο μεγαλύτερη ακρίβεια ζητάμε από τον αναλυτή φάσματος τόσο περισσότερη υπολογιστική ισχύς απαιτείται από τον επεξεργαστή DSP. Για αυτό τον λόγο χρησιμοποιείται μια τεχνική όπου χρησιμοποιούνται FFT 512 σημείων δύο φορές, οι οποίοι δίνουν μικρότερη πολυπλοκότητα από τον έναν 1024 σημείων μόνο του, και κάποια δύο φίλτρα επιπλέον. Η τεχνική έχει ώς εξής και φαίνεται και στο παραπανω σχήμα.

Το σήμα περνάει πρώτα από M-decimation για να μικρύνει ο ρυθμός δειγματοληψίας του ανά M (στην περίπτωση μας M=2 και ο ρυθμός δειγματοληψίας γίνεται μισός) και έπειτα περνάει από δύο φίλτρα ένα βαθυπερατό και ένα υψιπερατό με συχνότητα αποκοπής την π/M (όπου στην περίπτωσή μας είναι η μισή της συχνότητας δειγματοληψίας). Το αποτέλεσμα των δύο φίλτρων παρατίθεται δίπλα-δίπλα και γίνεται η απεικόνιση συχνότητας. Το γεγονός ότι πρώτα γίνεται το decimation και ύστερα το φιλτράρισμα είναι για να αποφευχθούν οι πράξεις στον υψηλό ρυθμό. Αυτή η τεχνική λέγεται πολυφασικό φιλτράρισμα.

Τα καλύτερα φίλτρα για την εφαρμογή μας είναι τα φίλτρα Nyquist όπου στο σημείο αποκοπής το ένα φίλτρο συμπληρώνει το άλλο και δεν έχουμε ενεργειακή ενίσχυση ή απώλεια. Παρόλα αυτά ένα οποιοδήποτε φίλτρο με απότομη καμπύλη αποκοπής είναι εξίσου ικανοποιητικό για την ακρίβεια που επιζητούμε.

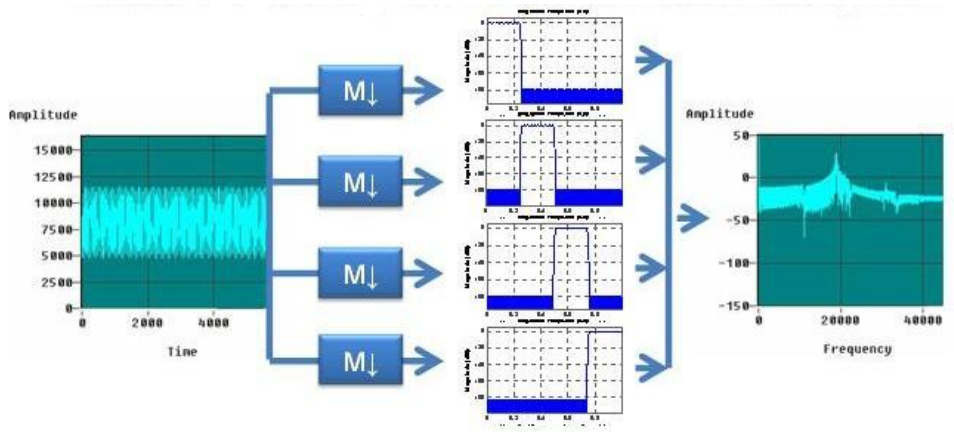
Το κεφάλαιο 10 του βιβλίου [3] της βιβλιογραφίας δίνει μία πολύ καλή θεωρητική προσέγγιση των multirate systems, που προτείνω ανεπιφύλακτα

και είναι αναγκαίο να γίνει κατανοητή για την πλήρη μελέτη της διπλωματικής αυτής εργασίας.

```
#include <stdio.h>
#include <stdlib.h>
#include <chan.h>
#include "vfft.h"
#include "../../sdr8036.h"

INPUT_PORT (0, MAININPUT)
INPUT_PORT (1, INPUTFFT)
INPUT_PORT (2, INPUTIFFT)
OUTPUT_PORT (0, MAINOUTPUT)
OUTPUT_PORT (1, OUTPUTFFT)
OUTPUT_PORT (2, OUTPUTIFFT)

#define FILTER          BANDPASS          //type of filter

void * GetMem(unsigned int size)
{
  void * P = memalign(128, size);
  if (P==0) {
    printf("Cannot allocate buffers\n");
    exit(1);
  }
  return P;
}

main()
{
    int i,j,f,s;
    float *data, *outp, *conv, *preconv, *ptr;
    data = GetMem(M*SAMPLES*sizeof(float));
    outp = GetMem(M*SAMPLES*sizeof(float));
    conv = GetMem(SAMPLES*sizeof(float));
    preconv = conv;

    if (data==0 || conv==0 || outp==0) {
        printf("Cannot allocate buffers\n");
        exit(1);
    }

  for(;;)
  {
    for (f=0; f<M; f++)
    {
        conv = preconv;
            for (i=0; i<M; i++)
```

```
        {
            chan_in_message
                (SAMPLES*sizeof(float), data+(i*SAMPLES), &MAININPUT);
        for (s=0;s<SAMPLES;s+=M) {
            ptr = data+(i*SAMPLES)+s;                                    50
          *conv=0;
          for(j=0;j<BL;j++) {
            *conv += (*ptr--)*(FILTER[M][f][ (int)BL-1-j]);
            if (ptr < data) ptr = data+M*SAMPLES-1;
            }
            conv++;
            }
        }
        chan_out_message
            (SAMPLES*sizeof(float), preconv, &OUTPUTFFT);                60
        if (f%2==0)
            chan_in_message
                (SAMPLES*sizeof(float), outp+(f*SAMPLES/2), &INPUTFFT);
        else {
            chan_in_message
                (SAMPLES*sizeof(float), preconv, &INPUTFFT);
            for (i=0; i<SAMPLES/2; i++)
                *(outp+(f*SAMPLES/2+i)) = *(preconv+SAMPLES/2+i);
        }
    }                                                                    70
    chan_out_message
        (M*SAMPLES/2*sizeof(float), outp, &MAINOUTPUT);

  }
}//main
```

4.8 Υλοποίηση με ανάλυση 2048-point FFT



Σχήματα 4.4: Ανάλυση Φάσματος με ακρίβεια: 2048-point FFT

Η τεχνική που χρησιμοποιείται εδώ είναι ίδια με την προηγούμενη περίπτωση μόνο που χρησιμοποιούμε M=4 έχοντας τώρα 4 FFTs 512-σημείων για την δημιουργία του 2048-σημείων. Το σήμα περνάει πρώτα από 4-decimation για να μικρύνει ο ρυθμός δειγματοληψίας του ανά 4 και έπειτα περνάει από τέσσερα φίλτρα ένα βαθυπερατό, δύο ζωνοπερατά και ένα υψιπερατό με συχνότητα αποκοπής την π/4. Το αποτέλεσμα των τεσσάρων φίλτρων παρατίθεται δίπλα-δίπλα και γίνεται η απεικόνιση συχνότητας. Το γεγονός ότι πρώτα γίνεται το decimation και ύστερα το φιλτράρισμα είναι για να αποφευχθούν οι πράξεις στον υψηλό ρυθμό.

Τα καλύτερα φίλτρα για την εφαρμογή μας είναι τα φίλτρα Nyquist όπου στο σημείο αποκοπής το ένα φίλτρο συμπληρώνει το άλλο και δεν έχουμε ενεργειακή ενίσχυση ή απώλεια. Παρόλα αυτά ένα οποιοδήποτε φίλτρο με απότομη καμπύλη αποκοπής είναι εξίσου ικανοποιητικό για την ακρίβεια που επιζητούμε.

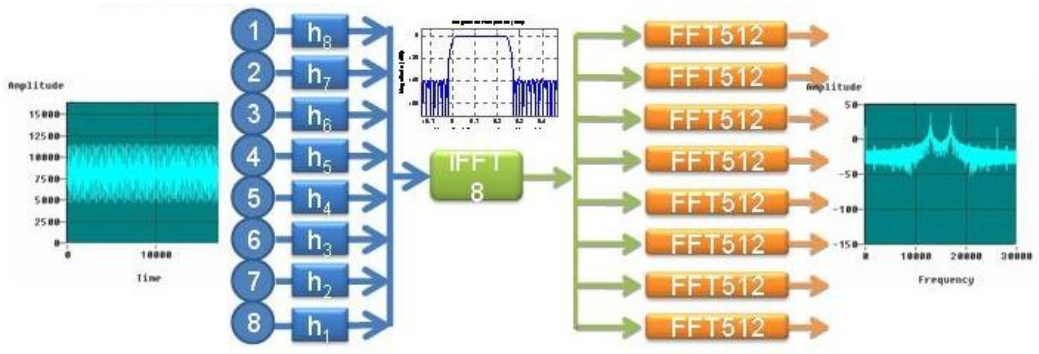
Το κεφάλαιο 10 του βιβλίου [3] της βιβλιογραφίας δίνει μία πολύ καλή θεωρητική προσέγγιση των multirate systems, που προτείνω ανεπιφύλακτα και είναι αναγκαίο να γίνει κατανοητή για την πλήρη μελέτη της διπλωματικής αυτής εργασίας.

```
#include <stdio.h>
#include <stdlib.h>
#include <chan.h>
#include "vfft.h"
```

```
#include "coefs.h"
#include "../../sdr8036.h"

INPUT_PORT (0, MAININPUT)
INPUT_PORT (1, INPUTFFT)
INPUT_PORT (2, INPUTIFFT)
OUTPUT_PORT (0, MAINOUTPUT)
OUTPUT_PORT (1, OUTPUTFFT)
OUTPUT_PORT (2, OUTPUTIFFT)

#define NYQUIST       N
#define BANDPASS      B
#define FILTER        BANDPASS          //type of filter
#define BL                FILTER[M][M][f] //length of filter

void * GetMem(unsigned int size)
{
  void * P = memalign(128, size);
  if (P==0) {
    printf("Cannot allocate buffers\n");
    exit(1);
  }
  return P;
}

main()
{
    int i,j,f,s;
    float *data, *outp, *conv, *preconv, *ptr;
    data = GetMem(M*SAMPLES*sizeof(float));
    outp = GetMem(M*SAMPLES*sizeof(float));
    conv = GetMem(SAMPLES*sizeof(float));
    preconv = conv;
    if (data==0 || conv==0 || outp==0) {
    printf("Cannot allocate buffers\n");
    exit(1);
  }

  for(;;)
  {
    for (f=0; f<M; f++)
    {
        conv = preconv;
            for (i=0; i<M; i++)
            {
                chan_in_message
                    (SAMPLES*sizeof(float), data+(i*SAMPLES), &MAININPUT);
            for (s=0;s<SAMPLES;s+=M) {
                ptr = data+(i*SAMPLES)+s;
```

```
            *conv=0;
            for(j=0;j<BL;j++) {
              *conv += (*ptr--)*(FILTER[M][f][ (int)BL-1-j]);
              if (ptr < data) ptr = data+M*SAMPLES-1;
              }
              conv++;
              }
          }
          chan_out_message
              (SAMPLES*sizeof(float), preconv, &OUTPUTFFT);
          if (f%2==0)
              chan_in_message
                  (SAMPLES*sizeof(float), outp+(f*SAMPLES/2), &INPUTFFT);
          else {
              chan_in_message
                  (SAMPLES*sizeof(float), preconv, &INPUTFFT);
              for (i=0; i<SAMPLES/2; i++)
                  *(outp+(f*SAMPLES/2+i)) = *(preconv+SAMPLES/2+i);
          }
      }
      chan_out_message
          (M*SAMPLES/2*sizeof(float), outp, &MAINOUTPUT);

  }
}//main
```

4.9 Υλοποίηση με ανάλυση 4096-point FFT



Σχήματα 4.5: Ανάλυση Φάσματος με ακρίβεια: 4096-point FFT

Η τεχνική που χρησιμοποιείται για τον 4096-σημείων FFT είναι περισσότερο αποδοτική. Αξιοποιεί ένα μαθηματικό τέχνασμα, με αποτέλεσμα να έχει καλύτερη απόδοση από τη προηγούμενη μέθοδο. Αυτό που γίνεται σε αυτή τη μέθοδο, είναι ότι τα δείγματα του σήματος χωρίζονται όπως στην περίπτωσή μας σε 8 μικρότερα υποσήματα που μπαίνουν σαν είσοδο σε 8 διαφορετικά φίλτρα που αλληλοσυμπληρούμενα καλύπτουν όλες τις συχνότητες. Την έξοδο αυτών των φίλτρων την περνάμε σε έναν αντίστροφο iFFT 8-σημείων που έχει ελάχιστη πολυπλοκότητα και δείχνουμε το αποτέλεσμα στο πεδίο συχνότητας.

Η μέθοδος αυτή αναλύεται θεωρητικά στο βιβλίο [5] της βιβλιογραφίας και δείχνει και εξηγεί με λεπτομέρεια την τεχνική αυτή.

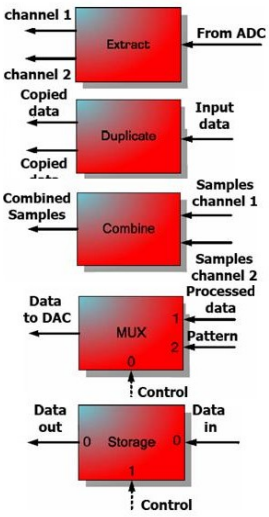
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <chan.h>
#include "dsp_fft32x32.h"
#include "dsp_ifft32x32.h"
#include "../../sdr8036.h"

#define PI (3.141592654)
#define NN M                                                    10

INPUT_PORT (0, MAININPUT)
INPUT_PORT (1, INPUTFFT)
INPUT_PORT (2, INPUTIFFT)
OUTPUT_PORT (0, MAINOUTPUT)
OUTPUT_PORT (1, OUTPUTFFT)
```

```
OUTPUT_PORT (2, OUTPUTIFFT)

#pragma DATA_ALIGN(x, 8)
#pragma DATA_ALIGN(xx, 8)                                                    20
#pragma DATA_ALIGN(w, 8)
#pragma DATA_ALIGN(y, 8)
#pragma DATA_ALIGN(t, 8)
#pragma DATA_ALIGN(f, 8)
int x[2*NN];
int w[2*NN];
int t[2*SAMPLES];
int xx[2*SAMPLES];
int y[2*NN];
int f[2*SAMPLES];                                                            30
float ff[2*SAMPLES];
float fft[M][2*SAMPLES];

void * GetMem(unsigned int size)
{
  void * P = memalign(128, size);
  if (P==0) {
    printf("Cannot allocate buffers\n");
    exit(1);
  }                                                                          40
  return P;
}

static int d2i(double d)
{
    if (d >= 2147483647.0) return (int)0x7FFFFFFF;
    if (d <= -2147483648.0) return (int)0x80000000;
    return (int)d;
}
                                                                             50
int gen_twiddle_fft32x32(int *w, int n, double scale)
{
    int i, j, k, s=0, t;

    for (j = 1, k = 0; j < n >> 2; j = j << 2, s++)
    {
        for (i = t=0; i < n >> 2; i += j, t++)
        {
            w[k + 5] = d2i(scale * cos(6.0 * PI * i / n));
            w[k + 4] = d2i(scale * sin(6.0 * PI * i / n));                   60

            w[k + 3] = d2i(scale * cos(4.0 * PI * i / n));
            w[k + 2] = d2i(scale * sin(4.0 * PI * i / n));

            w[k + 1] = d2i(scale * cos(2.0 * PI * i / n));
```

```
            w[k + 0] = d2i(scale * sin(2.0 * PI * i / n));

            k += 6;
        }
    }

    return k;
}

main()
{
    int i,j,s,c;
    float *data, *conv, *preconv, *ptr, *outp, norm;
    data = GetMem(M*SAMPLES*sizeof(float));
    outp = GetMem(M*SAMPLES*sizeof(float));
    conv = GetMem(2*M*sizeof(float));
    preconv = conv;
    for (i=0;i<2*SAMPLES;i++) {
        f[i]=0;
        ff[i]=0;
        for (j=0;j<M;j++) fft[j][i]=0;
    }
    if (data==0 || conv==0 || outp ==0) {
    printf("Cannot allocate buffers\n");
    exit(1);
  }

    gen_twiddle_fft32x32(w, NN, 2147483647.);
    gen_twiddle_fft32x32(t, SAMPLES, 2147483647.);
    norm = (FILTER[M][0][(int)BL/2/2]>0)
        ? MAXIMUM*FILTER[M][0][(int)BL/2/2]
            :-MAXIMUM*FILTER[M][0][(int)BL/2/2];
    for(i=0;i<M;i++)
        chan_in_message
            (SAMPLES*sizeof(float), data+(M*SAMPLES), &MAININPUT);
    //for(i=0;i<(int)BL;i++)
    //  printf("\nfilter%d: %f",i,(float)(FILTER[M][0][i]));
    for(;;)
  {
    c=0;
    //FILTERING STAGE
    for(i=0;i<M*SAMPLES;i+=M) {
        c++; //printf("\niteration:%d",c);
        if (i%SAMPLES==0) {
            chan_in_message
                (SAMPLES*sizeof(float), data+i, &MAININPUT);
                //for (s=0;s<SAMPLES;s++)
                //  printf("\n%f",(*(data+s)-MAXIMUM/2)/(MAXIMUM/M/2));
        }
```

```
        conv = preconv;
        for (s=0;s<M;s++) {
            ptr = data+i+s;
            *(conv+2*s) = *(conv+2*s+1) = 0;
          for(j=2*(M-s);j<(int)BL;j+=2*M) {
            *(conv+2*s)    +=                                          120
                (((*ptr)-MAXIMUM/2)/(MAXIMUM/M/2))
                    *(float)(FILTER[M][0][j-2]);
            *(conv+2*s+1) +=
                (((*ptr)-MAXIMUM/2)/(MAXIMUM/M/2))
                    *(float)(FILTER[M][0][j-1]);
            ptr -= M;
            if (ptr < data) ptr = data+M*SAMPLES-M+s;
            }//for j
            x[2*NN-2*s-2] = 2147483647./NN * (*(conv+2*s));
            x[2*NN-2*s-1] = 2147483647./NN * (*(conv+2*s+1));           130
            } //for s

            //IFFT STAGE
            DSP_ifft32x32(w, NN, x, y);
        for(s=0;s<NN;s++) {
            fft[s][2*i/M] = ((float)y[2*s])/2147483647.;
          fft[s][2*i/M+1] = ((float)y[2*s+1])/2147483647.;
        }
        }// for i
                                                                        140
        //FFT STAGE
        for(s=0;s<M/2;s++) {
            for(i=0;i<2*SAMPLES;i+=2) {
                xx[i]    = 2147483647./SAMPLES * fft[s][i];
                xx[i+1] = 2147483647./SAMPLES * fft[s][i+1];
            }
            DSP_fft32x32(t, SAMPLES, xx, f);
          for(i=0;i<2*SAMPLES;i+=2) {
                ff[i] = ((float)f[i])*(SAMPLES)/2147483647.;
                ff[i+1] = ((float)f[i+1])*(SAMPLES)/2147483647.;        150
                *(outp+s*SAMPLES+i/2) =
                    20*log10(sqrt(ff[i]*ff[i]+ff[i+1]*ff[i+1]))+100;
        }
      }
        chan_out_message
            (M*SAMPLES/2*sizeof(float), outp, &MAINOUTPUT);
    }
}//main
```

4.10 Οι διεργασίες του FPGA



Σχήματα 4.6: Διεργασίες του FPGA

- Extract: Η διεργασία αυτή λαμβάνει τα δείγματα από τον ADC. Έχει δύο κανάλια σε περίπτωση που θέλουμε να εργαστούμε με φανταστικά σήματα.
- Duplicate: Προσφέρει το δείγμα σε 2 διεργασίες
- Combine: Αυτή η διεργασία ενώνει τα δείγματα που θα σταλθούν στον DAC από την SHBB σε μια ροή από 32-bit τιμές.
- Mux: Αυτή η διεργασία είναι ένας 2σε1 μίκτης των καναλιών. Επιλέγει κάθε φορά τη σωστή διαδρομή ανάλογα με το δεδομένο που θα πάρει από τη διεργασία ελέγχου.
- Storage: Αυτή η διεργασία αποθηκεύει τα δεδομένα που έρχονται στην είσοδό του σε μια 16KB μνήμη. Τα αποθηκευμένα δεδομένα μετά στέλνονται μέσω του καναλιού 0.

4.11 Συντελεστές των Φίλτρων

```
/*
 * Filter Coefficients (C Source) generated
 * by the Filter Design and Analysis Tool
 *
 * Generated by MATLAB(R) 7.3 and the
 * Signal Processing Toolbox 6.6.
 *
 * Generated on: 02-Jul-2007 12:21:01
 *
 */

/*
 * Discrete-Time FIR Filter (real)
 */

/* General type conversion for MATLAB generated C-code */
#include "tmwtypes.h"
/*
 * Expected path to tmwtypes.h
 * C:\Program Files\MATLAB\R2006b\extern\include\tmwtypes.h
 */

#define NYQUIST         N
#define BANDPASS        B
#define FILTER          NYQUIST             //type of filter
#define BL                  FILTER[M][M][0] //length of filter

real64_T NP1of2[51] = {
0.0001106724158607,0,−0.0003428820301325,0,
0.0008340549136091,0,−0.001733499260122,0,
0.003244461098268,0,−0.005630672290565,0,
0.009243021075822,0, −0.01458764200536,0,
0.02250676355665,0, −0.0346983515693,0,
0.05551867613862,0, −0.1010280645801,0,
0.316585514921,0.5,0.316585514921,0,
−0.1010280645801,0, 0.05551867613862,0,
−0.0346983515693,0, 0.02250676355665,0,
−0.01458764200536,0, 0.009243021075822,0,
−0.005630672290565,0, 0.003244461098268,0,
−0.001733499260122,0,0.0008340549136091,0,
−0.0003428820301325,0,0.0001106724158607
};

real64_T NP2of2[51] = {
  0.0001106724158607,0,−0.0003428820301325,0,
  0.0008340549136091,0,−0.001733499260122,0,
```

```
    0.003244461098268,0,−0.005630672290565,0,
    0.009243021075822,0, −0.01458764200536,0,
     0.02250676355665,0, −0.0346983515693,0,
     0.05551867613862,0, −0.1010280645801,0,
       0.316585514921,−0.5, 0.316585514921,0,
     −0.1010280645801,0, 0.05551867613862,0,
     −0.0346983515693,0, 0.02250676355665,0,
    −0.01458764200536,0, 0.009243021075822,0,
   −0.005630672290565,0, 0.003244461098268,0,
   −0.001733499260122,0,0.0008340549136091,0,
   −0.0003428820301325,0,0.0001106724158607
};

real64_T NP1of8[224] = {
−0.004471747388754450000000000000, 0.000000000000000000000000000000,
0.002117533402534020000000000000, 0.000877111054107636000000000000,
0.001613813983398920000000000000, 0.001613813983398920000000000000,
0.000931986453960515000000000000, 0.002250014337099480000000000000,
0.000000000000000000000000000000, 0.002563806142544630000000000000,
−0.000963428510718290000000000000, 0.002325922176953010000000000000,
−0.001550333965533410000000000000, 0.001550333965533410000000000000,
−0.001432066410360170000000000000, 0.000593181329390137000000000000,
−0.000619183041586821000000000000, 0.000000000000000000000000000000,
0.000462278492959013000000000000, 0.000191482021377018000000000000,
0.001170393695632360000000000000, 0.001170393695632360000000000000,
0.001015936207435770000000000000, 0.002452686970497310000000000000,
0.000000000000000000000000000000, 0.003312585182657350000000000000,
−0.001327706841742710000000000000, 0.003205367864190800000000000000,
−0.002148393102261390000000000000, 0.002148393102261390000000000000,
−0.001864106235052300000000000000, 0.000772138084262914000000000000,
−0.000514053239592892000000000000, 0.000000000000000000000000000000,
0.001176425687378940000000000000, 0.000487291474836449000000000000,
0.002169015451821270000000000000, 0.002169015451821270000000000000,
0.001744810306114740000000000000, 0.004212344704790550000000000000,
0.000000000000000000000000000000, 0.005451413233693610000000000000,
−0.002111950181564030000000000000, 0.005098698771388200000000000000,
−0.003284500063592840000000000000, 0.003284500063592850000000000000,
−0.002652401420036320000000000000, 0.001098660641036700000000000000,
−0.000382569877565586000000000000, 0.000000000000000000000000000000,
0.002277906317873860000000000000, 0.000943539690678712000000000000,
0.003711168823405140000000000000, 0.003711168823405140000000000000,
0.002857752838759960000000000000, 0.006899225661244480000000000000,
0.000000000000000000000000000000, 0.008675872663273780000000000000,
−0.003273349477555660000000000000, 0.007902564703101770000000000000,
−0.004916964799334680000000000000, 0.004916964799334700000000000000,
−0.003659439289564970000000000000, 0.001515789384418780000000000000,
0.000093145378410432500000000000, 0.000000000000000000000000000000,
0.004300602870921380000000000000, 0.001781368035516300000000000000,
```

0.006376758428839390000000000000, 0.006376758428839400000000000000,
0.004752884433125930000000000000, 0.011474478058844500000000000000,
0.000000000000000000000000000000, 0.014153674117965300000000000000,
−0.005241640269872380000000000000, 0.012654439028606900000000000000,
−0.007644715362588320000000000000, 0.007644715362588340000000000000, 100
−0.005192360548358900000000000000, 0.002150746159861260000000000000,
0.001362098737496260000000000000, 0.000000000000000000000000000000,
0.008552977193936270000000000000, 0.003542759152396160000000000000,
0.011954005092780300000000000000, 0.011954005092780300000000000000,
0.008799185136729650000000000000, 0.021243112094924400000000000000,
0.000000000000000000000000000000, 0.026220418763653800000000000000,
−0.009748529482639100000000000000, 0.023535032090181300000000000000,
−0.014141502277238700000000000000, 0.014141502277238800000000000000,
−0.008798728548588460000000000000, 0.003644552696464670000000000000,
0.005655597921243290000000000000, 0.000000000000000000000000000000, 110
0.022795459812969100000000000000, 0.009442188615062610000000000000,
0.032632339050133200000000000000, 0.032632339050133300000000000000,
0.026147761241429300000000000000, 0.063126279814751800000000000000,
0.000000000000000000000000000000, 0.089258355314664500000000000000,
−0.040951052970668500000000000000, 0.098864587475246900000000000000,
−0.084782321073113500000000000000, 0.084782321073113800000000000000,
−0.117035596476349000000000000000, 0.048477731340928700000000000000,
−0.126678416782569000000000000000, 0.000000000000000000000000000000,
−0.110773440790388000000000000000, −0.045883861526111600000000000000,
−0.075667679348772300000000000000, −0.075667679348772300000000000000, 120
−0.034157693779078700000000000000, −0.082463967580838500000000000000,
0.000000000000000000000000000000, −0.068327392904962600000000000000,
0.017660494632580500000000000000, −0.042636205660193300000000000000,
0.017446889607134900000000000000, −0.017446889607135000000000000000,
0.005225091163550060000000000000, −0.002164303624578250000000000000,
−0.009523675153482170000000000000, 0.000000000000000000000000000000,
−0.018476763143155100000000000000, −0.007653325882650130000000000000,
−0.018012933722186200000000000000, −0.018012933722186200000000000000,
−0.010034119850525100000000000000, −0.024224508229614600000000000000,
0.000000000000000000000000000000, −0.022993378841483200000000000000, 130
0.006469460937340470000000000000, −0.015618660336170400000000000000,
0.006546165339031710000000000000, −0.006546165339031740000000000000,
0.001258415144832260000000000000, −0.000521252620085225000000000000,
−0.005620170558649610000000000000, 0.000000000000000000000000000000,
−0.009988301969779400000000000000, −0.004137290140960500000000000000,
−0.009685288324244060000000000000, −0.009685288324244060000000000000,
−0.005416376592039900000000000000, −0.013076289827422800000000000000,
0.000000000000000000000000000000, −0.012419885553319500000000000000,
0.003451076793262260000000000000, −0.008331636399084820000000000000,
0.003291538935766700000000000000, −0.003291538935766710000000000000, 140
0.000086055108661417300000000000, −0.000035645193119049400000000000,
−0.003960948544468670000000000000, 0.000000000000000000000000000000,
−0.006424324106411520000000000000, −0.002661042173956050000000000000,
−0.006048361170140400000000000000, −0.006048361170140360000000000000,

−0.003320112729544050000000000000, −0.008015461180272840000000000000,
0.000000000000000000000000000000, −0.007467668043788170000000000000,
0.002008470094211010000000000000, −0.004848875741065000000000000000,
0.001743434016660180000000000000, −0.001743434016660190000000000000,
−0.000353448479638194000000000000, 0.000146403153866292000000000000,
−0.002870938609091790000000000000, 0.000000000000000000000000000000,
−0.004291406141224510000000000000, −0.001777558625346390000000000000,
−0.003902375092861000000000000000, −0.003902375092861000000000000000,
−0.002086165527510350000000000000, −0.005036449109870690000000000000,
0.000000000000000000000000000000, −0.004559409053408250000000000000,
0.001173862703682500000000000000, −0.002833955259594270000000000000,
0.000900397239937269000000000000, −0.000900397239937267000000000000,
−0.000474923266680994000000000000, 0.000196719658145802000000000000,
−0.002017694049337030000000000000, 0.000000000000000000000000000000,
−0.002807010861410600000000000000, −0.001162701968524850000000000000,
−0.002453282352522590000000000000, −0.002453282352522570000000000000,
−0.001267671467701050000000000000, −0.003060429649957290000000000000,
0.000000000000000000000000000000, −0.002654769246625070000000000000,
0.000633412503711924000000000000, −0.001529193057038030000000000000,
0.000353812640788231000000000000, −0.000353812640788233000000000000,
−0.000572050539000148000000000000, 0.000236951091616703000000000000,
−0.001550057512874570000000000000, 0.000000000000000000000000000000,
−0.002025608942555880000000000000, −0.000839034696070872000000000000,
−0.001780183764180920000000000000, −0.001780183764180920000000000000,
−0.000981126134547683000000000000, −0.002368648020423690000000000000,
0.000000000000000000000000000000, −0.002435397969022540000000000000,
0.000873389834171957000000000000, −0.002108549582896760000000000000,
0.001620689901258890000000000000, −0.001620689901258890000000000000,
−0.004131355887031030000000000000, 0.001711263639398180000000000000
};

real64_T P1of2[254] = {
−0.001262150448263,−0.005333345456657, −0.01062070936177, −0.01170603119444,
−0.005274660077669, 0.003395355253078, 0.005381920661686,−8.918110279966e−005,
−0.004095183631453, −0.00116253012447, 0.003038018549388, 0.001598769879359,
−0.002295496208347,−0.001727475316983, 0.001813521473116, 0.001724985075742,
−0.001492968968984,−0.001684738103632, 0.001295287328391, 0.001627646332986,
−0.001172020894051,−0.001580119146577, 0.001108170200839, 0.001537755841207,
−0.001078380928863, −0.00150732034561, 0.001081204102925, 0.001482306330691,
−0.001105476150458,−0.001469610569598, 0.001148774502242, 0.001462186314831,
−0.001203487296429,−0.001460283062571, 0.00127346618727, 0.001461966143231,
−0.001353195027154,−0.001468579680238, 0.001442537926639, 0.001474465995065,
−0.001541629965744,−0.001482484818345, 0.001650273569975, 0.001489780132938,
−0.001765716070169,−0.001492507713536, 0.001896152357884, 0.001497882191451,
−0.002031966933425, −0.00149861912705, 0.002179567518017, 0.001497500525951,
−0.002335948584404,−0.001494520629175, 0.002498015247003, 0.001481401178321,
−0.002674191062274, −0.00146673816534, 0.002855917579575, 0.001439702201901,
−0.003052965361287,−0.001408418743639, 0.003258931234704, 0.001366455427607,

```
−0.003477940072306,−0.001315619376473, 0.003709544043945, 0.001255120749933,
−0.003952648953718,−0.001182171629991, 0.004209172012088, 0.00109598948326,
−0.004479307444241,−0.0009945241422194, 0.004764369778868,0.0008752400486084,
−0.005067420729286,−0.0007383998345528, 0.005389349927593,0.0005831191087228,
−0.005730175508933,−0.0004066536833661, 0.006091328335818,0.0002061216072315,
−0.006473997405271,2.322239732855e−005, 0.006881850816355,−0.000286201643543,
 −0.00732093633822,0.0005846999007115, 0.007794933131991,−0.0009228631208744,
−0.008308094101747, 0.001306682953058, 0.008866411697775,−0.001742421590733,
−0.009472446069594, 0.002250265678865, 0.01015089468842, −0.00282568209927,
 −0.01089428742791, 0.003513037313634, 0.01175616130907,−0.004294634411841,
−0.01272115715049, 0.005238207164078, 0.01385524803028,−0.006364751920131,
−0.01519641812834, 0.007740863064078, 0.01681490494959,−0.009477567615578,
−0.01885269782757, 0.01170139362945, 0.02148625938139, −0.01467506788484,
−0.0250522626585, 0.01888652273767, 0.03023393751475, −0.0253039784773,
−0.03852287782926, 0.0363147676516, 0.05404610091074, −0.05980204476039,
−0.09418404412835, 0.1456594039701, 0.4544317601654, 0.4544317601654,
0.1456594039701, −0.09418404412835, −0.05980204476039, 0.05404610091074,
0.0363147676516, −0.03852287782926, −0.0253039784773, 0.03023393751475,
0.01888652273767, −0.0250522626585, −0.01467506788484, 0.02148625938139,
0.01170139362945, −0.01885269782757,−0.009477567615578, 0.01681490494959,
0.007740863064078, −0.01519641812834,−0.006364751920131, 0.01385524803028,
0.005238207164078, −0.01272115715049,−0.004294634411841, 0.01175616130907,
0.003513037313634, −0.01089428742791, −0.00282568209927, 0.01015089468842,
0.002250265678865,−0.009472446069594,−0.001742421590733, 0.008866411697775,
0.001306682953058,−0.008308094101747,−0.0009228631208744, 0.007794933131991,
0.0005846999007115, −0.00732093633822,−0.000286201643543, 0.006881850816355,
2.322239732855e−005,−0.006473997405271,0.0002061216072315, 0.006091328335818,
−0.0004066536833661,−0.005730175508933,0.0005831191087228, 0.005389349927593,
−0.0007383998345528,−0.005067420729286,0.0008752400486084, 0.004764369778868,
−0.0009945241422194,−0.004479307444241, 0.00109598948326, 0.004209172012088,
−0.001182171629991,−0.003952648953718, 0.001255120749933, 0.003709544043945,
−0.001315619376473,−0.003477940072306, 0.001366455427607, 0.003258931234704,
−0.001408418743639,−0.003052965361287, 0.001439702201901, 0.002855917579575,
−0.00146673816534,−0.002674191062274, 0.001481401178321, 0.002498015247003,
−0.001494520629175,−0.002335948584404, 0.001497500525951, 0.002179567518017,
−0.00149861912705,−0.002031966933425, 0.001497882191451, 0.001896152357884,
−0.001492507713536,−0.001765716070169, 0.001489780132938, 0.001650273569975,
−0.001482484818345,−0.001541629965744, 0.001474465995065, 0.001442537926639,
−0.001468579680238,−0.001353195027154, 0.001461966143231, 0.00127346618727,
−0.001460283062571,−0.001203487296429, 0.001462186314831, 0.001148774502242,
−0.001469610569598,−0.001105476150458, 0.001482306330691, 0.001081204102925,
 −0.00150732034561,−0.001078380928863, 0.001537755841207, 0.001108170200839,
−0.001580119146577,−0.001172020894051, 0.001627646332986, 0.001295287328391,
−0.001684738103632,−0.001492968968984, 0.001724985075742, 0.001813521473116,
−0.001727475316983,−0.002295496208347, 0.001598769879359, 0.003038018549388,
 −0.00116253012447,−0.004095183631453,−8.918110279966e−005, 0.005381920661686,
 0.003395355253078,−0.005274660077669, −0.01170603119444, −0.01062070936177,
−0.005333345456657,−0.001262150448263
};
```

```
real64_T P2of2[277] = {
  −0.0006649861061179, 0.00217619192376,−0.002794504408304,−0.0001094941802929,
    0.00582816001826, −0.00910090822799, 0.006099506160502,−0.0003525959919162,
  −0.001348740278129,−0.001562394302825, 0.003117485262194,−0.0005564292481984,
  −0.001688732849943,0.0001282490365634, 0.001891080980135,−0.0006029785139787,
  −0.001452629527431,0.0005020601972844, 0.001422622845285,−0.0005959383070952,
  −0.001282437756185,0.0005598624612568, 0.001255329606785,−0.000565858312893,
  −0.001220464604992,0.0005422265005794, 0.001223843620537,−0.0005276549106904,
  −0.001235311772361,0.0005049147776042, 0.001264642393227,−0.0004832881675145,
  −0.001304002548758,0.0004603287083963, 0.001351282325685,−0.0004329987615139,
  −0.001409478822475,0.0004053717718839, 0.001474468578244,−0.0003759821557633,
  −0.001544107290605,0.0003407105785078, 0.001624018967242,−0.0003082481992315,
  −0.001701256535176,0.0002634968331367, 0.00179053716063,−0.0002200126908385,
  −0.001878854971793,0.0001665447127486, 0.001973577065666,−0.0001057819273822,
   −0.00207611353976,4.274333479068e−005, 0.002178160180917,3.245627034689e−005,
  −0.002289600053918,−0.000110010389126, 0.002400752022967,0.0001974137289913,
  −0.002514676647802,−0.0002947720081636, 0.002631268427121,0.0004040978013715,
  −0.002754332020448,−0.0005190947340348, 0.002873509569727,0.0006539052076742,
  −0.003005122793772,−0.0007906730750076, 0.003130698741833,0.0009451895607819,
  −0.003259969728339,−0.001112867526719, 0.003389800700745, 0.001296506216728,
  −0.003521362331756,−0.001495814913138, 0.003652245210763, 0.001716483335771,
  −0.003790368478806,−0.001948463329373, 0.003923962829685, 0.002200414526895,
  −0.004052702368927,−0.002480799230543, 0.004185877192402, 0.002784749507532,
  −0.004323686694946, −0.00310607987827, 0.004453183499801, 0.003458722553744,
  −0.004580392600588,−0.003844876466436, 0.004707611999751, 0.004265412775008,
  −0.004834588656695,−0.004721143269055, 0.00495599576425, 0.005221573753717,
  −0.005074360180719,−0.005771488705397, 0.005188853000708, 0.006379461003169,
  −0.005302322318808,−0.007049497535859, 0.005408639867056, 0.007797971307141,
  −0.005510505000677,−0.008638570563575, 0.005608205306285, 0.009589363451284,
  −0.005701319131664, −0.01067486203776, 0.005789520078731, 0.01192743630857,
  −0.005869635054114, −0.01339772804622, 0.005945774048097, 0.01514819452945,
  −0.006014946515512, −0.01727806394387, 0.006077954190801, 0.01993604664653,
  −0.006132906519284, −0.02336677332401, 0.006182765843022, 0.02798503651805,
  −0.006222953756514, −0.03458666039539, 0.00625805131268, 0.04486299878146,
  −0.006284675255608, −0.06322516284839, 0.006303615918548, 0.1058407228146,
  −0.006314820487163, −0.3182227736645, 0.5063194451248, −0.3182227736645,
  −0.006314820487163, 0.1058407228146, 0.006303615918548, −0.06322516284839,
  −0.006284675255608, 0.04486299878146, 0.00625805131268, −0.03458666039539,
  −0.006222953756514, 0.02798503651805, 0.006182765843022, −0.02336677332401,
  −0.006132906519284, 0.01993604664653, 0.006077954190801, −0.01727806394387,
  −0.006014946515512, 0.01514819452945, 0.005945774048097, −0.01339772804622,
  −0.005869635054114, 0.01192743630857, 0.005789520078731, −0.01067486203776,
  −0.005701319131664, 0.009589363451284, 0.005608205306285,−0.008638570563575,
  −0.005510505000677, 0.007797971307141, 0.005408639867056,−0.007049497535859,
  −0.005302322318808, 0.006379461003169, 0.005188853000708,−0.005771488705397,
  −0.005074360180719, 0.005221573753717, 0.00495599576425,−0.004721143269055,
  −0.004834588656695, 0.004265412775008, 0.004707611999751,−0.003844876466436,
  −0.004580392600588, 0.003458722553744, 0.004453183499801, −0.00310607987827,
```

```
  −0.004323686694946, 0.002784749507532, 0.004185877192402,−0.002480799230543,
  −0.004052702368927, 0.002200414526895, 0.003923962829685,−0.001948463329373,
  −0.003790368478806, 0.001716483335771, 0.003652245210763,−0.001495814913138,
  −0.003521362331756, 0.001296506216728, 0.003389800700745,−0.001112867526719,
  −0.003259969728339,0.0009451895607819, 0.003130698741833,−0.0007906730750076,
  −0.003005122793772,0.0006539052076742, 0.002873509569727,−0.0005190947340348,
  −0.002754332020448,0.0004040978013715, 0.002631268427121,−0.0002947720081636,
  −0.002514676647802,0.0001974137289913, 0.002400752022967,−0.000110010389126,
  −0.002289600053918,3.245627034689e−005, 0.002178160180917,4.274333479068e−005,
   −0.00207611353976,−0.0001057819273822, 0.001973577065666,0.0001665447127486,
  −0.001878854971793,−0.0002200126908385, 0.00179053716063,0.0002634968331367,
  −0.001701256535176,−0.0003082481992315, 0.001624018967242,0.0003407105785078,
  −0.001544107290605,−0.0003759821557633, 0.001474468578244,0.0004053717718839,
  −0.001409478822475,−0.0004329987615139, 0.001351282325685,0.0004603287083963,
  −0.001304002548758,−0.0004832881675145, 0.001264642393227,0.0005049147776042,
  −0.001235311772361,−0.0005276549106904, 0.001223843620537,0.0005422265005794,
  −0.001220464604992,−0.000565858312893, 0.001255329606785,0.0005598624612568,
  −0.001282437756185,−0.0005959383070952, 0.001422622845285,0.0005020601972844,
  −0.001452629527431,−0.0006029785139787, 0.001891080980135,0.0001282490365634,
  −0.001688732849943,−0.0005564292481984, 0.003117485262194,−0.001562394302825,
  −0.001348740278129,−0.0003525959919162, 0.006099506160502, −0.00910090822799,
   0.00582816001826,−0.0001094941802929,−0.002794504408304, 0.00217619192376,
  −0.0006649861061179
};

real64_T P1of4[254] = {
  −5.509248380491e−005,0.0001094666491473,0.0005173845223482, 0.001332444551284,
   0.002577871298759, 0.004105015246733, 0.005574123787516, 0.006524389410007,
   0.006527868912172, 0.005377605438386, 0.003227014100377,0.0006035552688861,
   −0.00173939214768, −0.00309231756619,−0.003077808679486,−0.001826691885967,
  5.222334694722e−005, 0.001715464465941, 0.002445864243663, 0.001970131058041,
  0.0005800588123735,−0.001009566508442,−0.002008010968529,−0.001936407900374,
  −0.000864232097113,0.0006285397071945, 0.001750706487919, 0.001911632209624,
   0.001033030994961,−0.0004060782882613,−0.001621410717763,−0.001944262258103,
  −0.001187877445693,0.0002414221252021, 0.001556482077897, 0.0020174075718,
   0.001345342271374, −0.00010874711914,−0.001542670962396,−0.002141098609936,
  −0.001536149184493,−3.162261558923e−005, 0.001545156916683, 0.002294573425825,
   0.001755182721417,0.0001851241747134,−0.001555732674865,−0.002473077781254,
  −0.002002925595027,−0.0003547305710462, 0.001574158409759, 0.002683411515483,
   0.00229418878436,0.0005604022651625,−0.001580241000084, −0.00290969842797,
  −0.002619435754719,−0.0007978634011403, 0.001576221556514, 0.003155191660107,
   0.002983862540499, 0.001072452200991,−0.001559806733975,−0.003423108099994,
  −0.003396068515086,−0.001395172417886, 0.001522131948413, 0.003710373421964,
   0.003859495748964, 0.001773214651132, −0.00145685423113,−0.004015118890048,
  −0.004377008776501,−0.002210831358554, 0.001363268789833, 0.004343321700686,
   0.004959986757554, 0.002718916658586,−0.001238540107098, −0.00470495891613,
  −0.005629543098631,−0.003321459819644, 0.001066774377899, 0.005099647981722,
   0.00639967363143, 0.004036715193861,−0.0008393641737652,−0.005537714406695,
  −0.007297370430479, −0.00489376506963,0.0005449825067804, 0.006038736455247,
```

```
  0.008369356939856, 0.005944270661095,−0.0001575969341236,−0.006623411106061,
  −0.00967883041713,−0.007261113348262,−0.0003583876862337, 0.007327326630417,
   0.01132755251054, 0.008960936374739, 0.001057914065828,−0.008215218268971,
  −0.01349434169771, −0.01125043990282,−0.002039840370673, 0.009406291557745,
   0.01651595781167, 0.01452966725363, 0.00350337127899, −0.01113882808376,
  −0.02109746441971, −0.01966996562208,−0.005900064015026, 0.01398522542873,
   0.02902079326139, 0.02901240067235, 0.01053318577909, −0.01973755395902,
  −0.04645225829903, −0.05170482185593, −0.02333364373956, 0.03838872359929,
    0.1190306383192, 0.1944711300158, 0.2399599507126, 0.2399599507126,
    0.1944711300158, 0.1190306383192, 0.03838872359929, −0.02333364373956,
  −0.05170482185593, −0.04645225829903, −0.01973755395902, 0.01053318577909,
   0.02901240067235, 0.02902079326139, 0.01398522542873,−0.005900064015026,
  −0.01966996562208, −0.02109746441971, −0.01113882808376, 0.00350337127899,
   0.01452966725363, 0.01651595781167, 0.009406291557745,−0.002039840370673,
  −0.01125043990282, −0.01349434169771,−0.008215218268971, 0.001057914065828,
  0.008960936374739, 0.01132755251054, 0.007327326630417,−0.0003583876862337,
 −0.007261113348262, −0.00967883041713,−0.006623411106061,−0.0001575969341236,
  0.005944270661095, 0.008369356939856, 0.006038736455247,0.0005449825067804,
  −0.00489376506963,−0.007297370430479,−0.005537714406695,−0.0008393641737652,
  0.004036715193861, 0.00639967363143, 0.005099647981722, 0.001066774377899,
 −0.003321459819644,−0.005629543098631, −0.00470495891613,−0.001238540107098,
  0.002718916658586, 0.004959986757554, 0.004343321700686, 0.001363268789833,
 −0.002210831358554,−0.004377008776501,−0.004015118890048, −0.00145685423113,
  0.001773214651132, 0.003859495748964, 0.003710373421964, 0.001522131948413,
 −0.001395172417886,−0.003396068515086,−0.003423108099994,−0.001559806733975,
  0.001072452200991, 0.002983862540499, 0.003155191660107, 0.001576221556514,
 −0.0007978634011403,−0.002619435754719, −0.00290969842797,−0.001580241000084,
 0.0005604022651625, 0.00229418878436, 0.002683411515483, 0.001574158409759,
 −0.0003547305710462,−0.002002925595027,−0.002473077781254,−0.001555732674865,
 0.0001851241747134, 0.001755182721417, 0.002294573425825, 0.001545156916683,
 −3.162261558923e−005,−0.001536149184493,−0.002141098609936,−0.001542670962396,
  −0.00010874711914, 0.001345342271374, 0.0020174075718, 0.001556482077897,
 0.0002414221252021,−0.001187877445693,−0.001944262258103,−0.001621410717763,
 −0.0004060782882613, 0.001033030994961, 0.001911632209624, 0.001750706487919,
 0.0006285397071945,−0.000864232097113,−0.001936407900374,−0.002008010968529,
 −0.001009566508442,0.0005800588123735, 0.001970131058041, 0.002445864243663,
  0.001715464465941,5.222334694722e−005,−0.001826691885967,−0.003077808679486,
  −0.00309231756619, −0.00173939214768,0.0006035552688861, 0.003227014100377,
  0.005377605438386, 0.006527868912172, 0.006524389410007, 0.005574123787516,
  0.004105015246733, 0.002577871298759, 0.001332444551284,0.0005173845223482,
 0.0001094666491473,−5.509248380491e−005
};

real64_T P2of4[254] = {
3.383690424752e−005,0.0003139276535622,0.0002459712363429,−0.0005310533627074,
−0.001156325186804,−0.0001607456501842, 0.001818680614639, 0.001887422665212,
−0.0008680240967284,−0.002963320638743,−0.001209287963225, 0.001948099285737,
 0.001909884998997,−0.0005086218906112,−0.0006652039736805, 0.001553182049336,
0.0004759282038958,−0.003952158544535, −0.00413205472305, 0.002815561028908,
```

0.00788585984413, 0.00288756574161,−0.006067293115373,−0.007079774743598,
−2.515725551656e−005, 0.004799404715953, 0.003013312261839,0.0001085331056907,
−0.0002741230806904,−0.000903408032478,−0.002597193345979,−0.001734931161095,
0.001558385205693, 0.002238348281946,−0.0002259392055546,−0.0005382954134802,
0.001958340189957, 0.001395560575732,−0.003091731470997,−0.004354314441231,
0.0002848650287114, 0.003919196204908, 0.002112081557704,−0.0003074265911241,
0.0003264905457444,−3.41073914583e−005,−0.003299528600665,−0.003648430145749,
0.001349484680102, 0.004590626372587, 0.001685952343335, −0.0012908592672,
0.0001906227580718,0.0006850367827976,−0.003281224971801, −0.00466997801874,
0.001003422701356, 0.00576772993376, 0.002826069905433,−0.001548591549828,
−0.0006657554777518,0.0002501369153061,−0.003688705987589,−0.005163538096563,
0.001542415517085, 0.007373556214718, 0.003551369610381,−0.002584310101347,
−0.001749504285676,0.0003208747325693,−0.003684551267978,−0.005923787876512,
0.001685116058304, 0.009178189379226, 0.004778735946107,−0.003560674349981,
−0.003216139865449,8.64021827889e−005,−0.003795714318713,−0.006722869872405,
0.002105687604652, 0.01162218619261, 0.006287672421993,−0.005099196721782,
−0.00537252394359,−0.0001414132952301,−0.003609862671128,−0.007611475354667,
0.002522624340593, 0.01475286328968, 0.008373940945545,−0.007166015161493,
−0.008476256146769,−0.0005462874581706,−0.003216391631212,−0.008707061045881,
0.003132524834879, 0.01927136814886, 0.01145746139572, −0.01035576944061,
−0.01341507461814,−0.001216095441579,−0.002312855826492, −0.01018300111254,
0.004060529206281, 0.02656299655599, 0.01663464520905, −0.01590140812945,
−0.02234939405829,−0.002493282175589,−0.0003435802948969, −0.0127637156956,
0.005870764921128, 0.04161807180646, 0.02791224943564, −0.02868411486553,
−0.04428243574023,−0.005875283604125, 0.005557828079746, −0.02010642079922,
0.01183451516926, 0.09826567322939, 0.07769394322777, −0.09805939554707,
−0.2031151933642, −0.04468845008458, 0.2000284287456, 0.2000284287456,
−0.04468845008458, −0.2031151933642, −0.09805939554707, 0.07769394322777,
0.09826567322939, 0.01183451516926, −0.02010642079922, 0.005557828079746,
−0.005875283604125, −0.04428243574023, −0.02868411486553, 0.02791224943564,
0.04161807180646, 0.005870764921128, −0.0127637156956,−0.0003435802948969,
−0.002493282175589, −0.02234939405829, −0.01590140812945, 0.01663464520905,
0.02656299655599, 0.004060529206281, −0.01018300111254,−0.002312855826492,
−0.001216095441579, −0.01341507461814, −0.01035576944061, 0.01145746139572,
0.01927136814886, 0.003132524834879,−0.008707061045881,−0.003216391631212,
−0.0005462874581706,−0.008476256146769,−0.007166015161493, 0.008373940945545,
0.01475286328968, 0.002522624340593,−0.007611475354667,−0.003609862671128,
−0.0001414132952301, −0.00537252394359,−0.005099196721782, 0.006287672421993,
0.01162218619261, 0.002105687604652,−0.006722869872405,−0.003795714318713,
8.64021827889e−005,−0.003216139865449,−0.003560674349981, 0.004778735946107,
0.009178189379226, 0.001685116058304,−0.005923787876512,−0.003684551267978,
0.0003208747325693,−0.001749504285676,−0.002584310101347, 0.003551369610381,
0.007373556214718, 0.001542415517085,−0.005163538096563,−0.003688705987589,
0.0002501369153061,−0.0006657554777518,−0.001548591549828, 0.002826069905433,
0.00576772993376, 0.001003422701356, −0.00466997801874,−0.003281224971801,
0.0006850367827976,0.0001906227580718, −0.0012908592672, 0.001685952343335,
0.004590626372587, 0.001349484680102,−0.003648430145749,−0.003299528600665,
−3.41073914583e−005,0.0003264905457444,−0.0003074265911241, 0.002112081557704,
0.003919196204908,0.0002848650287114,−0.004354314441231,−0.003091731470997,

0.001395560575732, 0.001958340189957,−0.0005382954134802,−0.0002259392055546,
0.002238348281946, 0.001558385205693,−0.001734931161095,−0.002597193345979,
−0.000903408032478,−0.0002741230806904,0.0001085331056907, 0.003013312261839,
0.004799404715953,−2.515725551656e−005,−0.007079774743598,−0.006067293115373,
0.00288756574161, 0.00788585984413, 0.002815561028908, −0.00413205472305,
−0.003952158544535,0.0004759282038958, 0.001553182049336,−0.0005652039736805,
−0.0005086218906112, 0.001909884998997, 0.001948099285737,−0.001209287963225,
−0.002963320638743,−0.0008680240967284, 0.001887422665212, 0.001818680614639,
−0.0001607456501842,−0.001156325186804,−0.0005310533627074,0.0002459712363429,
0.0003139276535622,3.383690424752e−005
};

real64_T P3of4[254] = {
0.0002107965425934,−0.0001268379724278,−0.0005362874905248,0.0009691432146328,
0.0001143897470137,−0.001960098379882, 0.001715630372765, 0.00139390886974,
−0.003627289395063, 0.001395530687748, 0.00291967245424, −0.00373958880893,
0.0003636163623789, 0.002415490258892,−0.002048208378893, 0.001052464675231,
−0.0007720068965339,−0.001226428908631, 0.004491163518329,−0.003393984228986,
−0.003302960501659, 0.007235614334233,−0.002044595174658,−0.005522805586534,
0.005073267287468, 0.001398295443234, −0.00349615419064,−0.0004249258468361,
0.0020213308571, 0.001584364771339,−0.003402034462191,−0.0002858412007821,
0.003733737644189,−0.002344229981565,−0.0005031167953113, 0.001268966264388,
−0.001722867228439, 0.002595238269891,−0.0007443198200637, −0.00325253410776,
0.003625354541485,0.0004301277022978,−0.002361988678499,−1.971388658871e−005,
0.0006096356533589, 0.002703495337202,−0.003018388360284,−0.002443710943523,
0.005673962818576,−0.001520859177847,−0.003225937629849, 0.002292112206736,
−0.0002230374936891, 0.001658424504226, −0.00178320065114,−0.002885087671501,
0.00548292262293,−0.0009724098722141,−0.003691923636189, 0.002149057870019,
5.493839737179e−005, 0.002238946389558,−0.002570519921144, −0.00371032501694,
0.007528344367494,−0.001432011522883,−0.005690012464837, 0.003914439120022,
0.0005134100798703, 0.001096422366257,−0.002493738179597,−0.003731514887005,
0.008553607499166,−0.001954224147993,−0.006954520962261, 0.005387787339485,
0.0004827832864881,0.0006226960193263,−0.002429626601493,−0.004624048608828,
0.01046996054577,−0.002198011716476,−0.009513902564848, 0.007565399194545,
0.001174808392004,−0.0007886997519853,−0.002493428653647,−0.005010063465219,
0.0127116676797,−0.003024985143453, −0.01252700387723, 0.01080367650965,
0.001629676227357,−0.002805177444671,−0.002067988294854, −0.00582563270058,
0.01566345469202,−0.003788428553463, −0.01714337755192, 0.01545086536745,
0.002690664165325,−0.006121027624246,−0.001574331168069,−0.006793728805771,
0.02042107738438,−0.005309718972745, −0.0249209305105, 0.02383348267983,
0.004442794736143, −0.01260074322333,−0.0001994343615058,−0.008502264408255,
0.0294621784598,−0.008278010365031, −0.04175708269883, 0.04291566190855,
0.008802264783336, −0.02955177063228, 0.003716345683893, −0.01342917670909,
0.05947695673601, −0.01955490159406, −0.1162877490393, 0.1467690624959,
0.04039975992738, −0.2247095704239, 0.1336745877574, 0.1336745877574,
−0.2247095704239, 0.04039975992738, 0.1467690624959, −0.1162877490393,
−0.01955490159406, 0.05947695673601, −0.01342917670909, 0.003716345683893,
−0.02955177063228, 0.008802264783336, 0.04291566190855, −0.04175708269883,
−0.008278010365031, 0.0294621784598,−0.008502264408255,−0.0001994343615058,

−0.01260074322333, 0.004442794736143, 0.02383348267983, −0.0249209305105,
−0.005309718972745, 0.02042107738438,−0.006793728805771,−0.001574331168069,
−0.006121027624246, 0.002690664165325, 0.01545086536745, −0.01714337755192,
−0.003788428553463, 0.01566345469202, −0.00582563270058,−0.002067988294854,
−0.002805177444671, 0.001629676227357, 0.01080367650965, −0.01252700387723,
−0.003024985143453, 0.0127116676797,−0.005010063465219,−0.002493428653647,
−0.0007886997519853, 0.001174808392004, 0.007565399194545,−0.009513902564848,
−0.002198011716476, 0.01046996054577,−0.004624048608828,−0.002429626601493,
0.0006226960193263,0.0004827832864881, 0.005387787339485,−0.006954520962261,
−0.001954224147993, 0.008553607499166,−0.003731514887005,−0.002493738179597,
0.001096422366257,0.0005134100798703, 0.003914439120022,−0.005690012464837,
−0.001432011522883, 0.007528344367494, −0.00371032501694,−0.002570519921144,
0.002238946389558,5.493839737179e−005, 0.002149057870019,−0.003691923636189,
−0.0009724098722141, 0.00548292262293,−0.002885087671501, −0.00178320065114,
0.001658424504226,−0.0002230374936891, 0.002292112206736,−0.003225937629849,
−0.001520859177847, 0.005673962818576,−0.002443710943523,−0.003018388360284,
0.002703495337202,0.0006096356533589,−1.971388658871e−005,−0.002361988678499,
0.0004301277022978, 0.003625354541485, −0.00325253410776,−0.0007443198200637,
0.002595238269891,−0.001722867228439, 0.001268966264388,−0.0005031167953113,
−0.002344229981565, 0.003733737644189,−0.0002858412007821,−0.003402034462191,
0.001584364771339, 0.0020213308571,−0.0004249258468361, −0.00349615419064,
0.001398295443234, 0.005073267287468,−0.005522805586534,−0.002044595174658,
0.007235614334233,−0.003302960501659,−0.003393984228986, 0.004491163518329,
−0.001226428908631,−0.0007720068965339, 0.001052464675231,−0.002048208378893,
0.002415490258892,0.0003636163623789, −0.00373958880893, 0.00291967245424,
0.001395530687748,−0.003627289395063, 0.00139390886974, 0.001715630372765,
−0.001960098379882,0.0001143897470137,0.0009691432146328,−0.0005362874905248,
−0.0001268379724278,0.0002107965425934
};

real64_T P4of4[277] = {
−1.63724710164e−005,0.0002425358330616,−0.000671990717356, 0.001403845483402,
−0.002379415838965, 0.003420513684886,−0.004238322032788, 0.004517983319141,
−0.004043217506348, 0.002813020931743,−0.001091667564487,−0.0006476129875672,
0.001880777554184,−0.002237461070739, 0.00166420471401,−0.000465411715076,
−0.0008106942052915, 0.001602464744862,−0.001582330985819,0.0008050556234288,
0.0003197180871611,−0.001225409694779, 0.001466201191324,−0.0009386350208707,
−6.658247633897e−005, 0.001013663850957,−0.001399974692227, 0.001021303650739,
−8.353775949798e−005,−0.0009029716134339, 0.001395459975491,−0.001114728574854,
0.0002043504783202,0.0008387932564023,−0.001431641430123, 0.001228194860526,
−0.00032090323114,−0.0007970670702112, 0.001496697517982,−0.001364495263188,
0.00044547335038, 0.000764474325733,−0.001583342301494, 0.00152496279027,
−0.0005856544626441,−0.0007324643243657, 0.001687262723703,−0.001712034338629,
0.0007496233032803,0.0006907188569489,−0.001800107065104, 0.001921919888946,
−0.0009383032099936,−0.000635736155299, 0.001918592094077,−0.002153683754737,
0.001153139671784,0.0005650052042226,−0.002041229404052, 0.002407848282152,
−0.001396349677886,−0.0004765052178361, 0.002168135718513,−0.002687756963215,
0.001673735461881,0.0003641319247279,−0.002295404140606, 0.002993017880147,
−0.001987844920296,−0.0002245473523432, 0.002421340491932,−0.003324818444244,

0.002342003703167,5.458131226216e−005,−0.002545284545671, 0.003685813104167,
−0.002740629079003,0.0001487276816414, 0.00266878316968,−0.004082860148428,
0.003193555436737,−0.0003936255453798,−0.002789487606451, 0.004520945421417,
−0.003710428930893,0.0006888012678326, 0.002905148232911,−0.005006267994284,
0.004302813404268,−0.001044369295106,−0.003013755067783, 0.005547043314098,
−0.004984670598281, 0.001470812056757, 0.003117011117976,−0.006159512104055,
0.005779986750874,−0.001986539303414,−0.003214664071993, 0.006865162751892,
−0.006722455870629, 0.002618180966946, 0.003305831288583,−0.007694856733472,
0.007861514759587,−0.003405723348499,−0.003387906910172, 0.008693120442614,
−0.009269603661786, 0.004407012047696, 0.003461461493013,−0.009935266195776,
0.01106731554779,−0.005719124993088,−0.003526607515549, 0.01154927506499,
−0.01346367515864, 0.007513664551839, 0.003583701550775, −0.01377195189116,
0.01685539153806, −0.01012623811781,−0.003630493873379, 0.01708926901601,
−0.02208857511024, 0.01430091478963, 0.003666691786231, −0.02269880410771,
0.0313667283619, −0.02210627532073, −0.003691982534, 0.03453778397679,
−0.05277213046261, 0.04223143622389, 0.003707591464077, −0.07755070863841,
0.159061702864, −0.2224210903649, 0.2462873426544, −0.2224210903649,
0.159061702864, −0.07755070863841, 0.003707591464077, 0.04223143622389,
−0.05277213046261, 0.03453778397679, −0.003691982534, −0.02210627532073,
0.0313667283619, −0.02269880410771, 0.003666691786231, 0.01430091478963,
−0.02208857511024, 0.01708926901601,−0.003630493873379, −0.01012623811781,
0.01685539153806, −0.01377195189116, 0.003583701550775, 0.007513664551839,
−0.01346367515864, 0.01154927506499,−0.003526607515549,−0.005719124993088,
0.01106731554779,−0.009935266195776, 0.003461461493013, 0.004407012047696,
−0.009269603661786, 0.008693120442614,−0.003387906910172,−0.003405723348499,
0.007861514759587,−0.007694856733472, 0.003305831288583, 0.002618180966946,
−0.006722455870629, 0.006865162751892,−0.003214664071993,−0.001986539303414,
0.005779986750874,−0.006159512104055, 0.003117011117976, 0.001470812056757,
−0.004984670598281, 0.005547043314098,−0.003013755067783,−0.001044369295106,
0.004302813404268,−0.005006267994284, 0.002905148232911,0.0006888012678326,
−0.003710428930893, 0.004520945421417,−0.002789487606451,−0.0003936255453798,
0.003193555436737,−0.004082860148428, 0.00266878316968,0.0001487276816414,
−0.002740629079003, 0.003685813104167,−0.002545284545671,5.458131226216e−005,
0.002342003703167,−0.003324818444244, 0.002421340491932,−0.0002245473523432,
−0.001987844920296, 0.002993017880147,−0.002295404140606,0.0003641319247279,
0.001673735461881,−0.002687756963215, 0.002168135718513,−0.0004765052178361,
−0.001396349677886, 0.002407848282152,−0.002041229404052,0.0005650052042226,
0.001153139671784,−0.002153683754737, 0.001918592094077,−0.000635736155299,
−0.0009383032099936, 0.001921919888946,−0.001800107065104,0.0006907188569489,
0.0007496233032803,−0.001712034338629, 0.001687262723703,−0.0007324643243657,
−0.0005856544626441, 0.00152496279027,−0.001583342301494, 0.000764474325733,
0.00044547335038,−0.001364495263188, 0.001496697517982,−0.0007970670702112,
−0.00032090323114, 0.001228194860526,−0.001431641430123,0.0008387932564023,
0.0002043504783202,−0.001114728574854, 0.001395459975491,−0.0009029716134339,
−8.353775949798e−005, 0.001021303650739,−0.001399974692227, 0.001013663850957,
−6.658247633897e−005,−0.0009386350208707, 0.001466201191324,−0.001225409694779,
0.0003197180871611,0.0008050556234288,−0.001582330985819, 0.001602464744862,
−0.0008106942052915,−0.000465411715076, 0.00166420471401,−0.002237461070739,
0.001880777554184,−0.0006476129875672,−0.001091667564487, 0.002813020931743,

```
−0.004043217506348, 0.004517983319141,−0.004238322032788, 0.003420513684886,
−0.002379415838965, 0.001403845483402,−0.000671990717356,0.0002425358330616,
−1.63724710164e−005
};

real64_T L2[2] = {254,277};
real64_T *B2[] = { P1of2, P2of2, L2 };

real64_T NL2[2]= {51,51};
real64_T *N2[] = { NP1of2, NP2of2, NL2 };

real64_T NL8[8]= {224,0,0,0,0,0,0,0};
real64_T *N8[] = { NP1of8, 0,0,0,0,0,0,0, NL8 };

real64_T L4[4] = {254,254,254,277};
real64_T *B4[] = { P1of4, P2of4, P3of4, P4of4, L4 };


real64_T **B[] = { 0,0,B2,0,B4 };
real64_T **N[] = { 0,0,N2,0,0,0,0,0,N8 };
```

Βιβλιογραφία

[1] Lapsley, Phil, Jeff Bier, and Edward A. Lee. *DSP Processor Fundamentals: Architectures and Features.* New York: IEEE Computer Society P, 1997.

[2] Mitola, Joseph. *Software Radio Architecture: Object-Oriented Approaches to Wireless Systems Engineering.* New York: John Wiley & Sons, Incorporated, 2004.

[3] Proakis, John G., and Dimitris G. Manolakis. *Digital Signal Processing.* Upper Saddle River: Pearson Education, 2006.

[4] Tuttlebee, Walter H. *Software Defined Radio - Enabling Technologies.* New York: John Wiley & Sons, Incorporated, 2002.

[5] Harris, Fredric J. *Multirate Signal Processing for Communication Systems.* Upper Saddle River: Prentice Hall PTR, 2004.

[6] Crochiere, Ronald E., and Lawrence R. Rabiner. *Multirate Digital Signal Processing.* Upper Saddle River: Prentice Hall, 1983.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Τομέας Συστημάτων Μετάδοσης Πληροφορίας
και Τεχνολογίας Υλικών
Εργαστήριο Κινητών Επικοινωνιών

Υλοποίηση Αναλυτή Φάσματος με τεχνικές Software Radio

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

-του-

Γεώργιος Ι. Ανδρίτσος **Georgios I. Andritsos**

Επιβλέπων: Φ. Κωνσταντίνου
Καθηγητής Εθνικού Μετσοβίου Πολυτεχνείου

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5 Νοεμβρίου 2007.

Φίλιππος Κωνσταντίνου	Νικόλαος Ουζούνογλου	Χρήστος Καψάλης Καθηγητής
Καθηγητής Εθνικού Μετσοβίου Πολυτεχνείου	Καθηγητής Εθνικού Μετσοβίου Πολυτεχνείου	Εθνικού Μετσοβίου Πολυτεχνείου

Αθήνα, (Νοέμβριος 2007).

Γεώργιος Ι. Ανδρίτσος
(Πτυχιούχος Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών)