ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ,

ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΘΕΣΣΑΛΙΑΣ

# Υλοποίηση και βελτιστοποίηση benchmark kernels από software σε hardware

# Hardware implementation and optimization of software benchmark kernels

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ευτυχία Παπούλια

Επιβλέποντες

Νικόλαος Μπέλλας
Αναπληρωτής Καθηγητής

Αντωνόπουλος Χρήστος
Επίκουρος Καθηγητής

Βόλος, 9 Ιουλίου 2012

Αρχικά θα ήθελα να ευχαριστήσω τους καθηγητές κ. Μπέλλα Νικόλαο και κ. Αντωνόπουλο Χρήστο για την υποστήριξη και τις πολύτιμες συμβουλές που μου έδωσαν κατά την διάρκεια της εκπόνησης της διπλωματικής μου εργασίας. Επίσης, θα ήθελα να ευχαριστήσω τον διδακτορικό φοιτητή Muhsen Owaida για την βοήθεια που μου παρείχε όλο αυτό το διάστημα.

Τίποτα από αυτά δεν θα ήταν πραγματικότητα χωρίς την στήριξη των γονέων μου, Κυριαζή και Γαρυφαλλιάς, καθώς και της αδερφής μου, Μυρτώς. Τέλος, να ευχαριστήσω τους φίλους μου Χρήστο, Ελένη, Ελένη, Μάγδα και Υρώ για την συμπαράσταση και υπομονή που επέδειξαν όλα αυτά τα χρόνια.

# Περιεχόμενα

# Περίληψη

Με την πάροδο του χρόνου το κόστος του υλικού όσο και το μέγεθός του ολοένα και μειώνεται. Ως αποτέλεσμα έχει να χρησιμοποιείται όλο και πιο πολύ και να αποκτά όλο και μεγαλύτερες δυνατότητες. Είναι σημαντικό να υπάρχει μια κοινή βάση για την μελέτη αυτών των δυνατοτήτων του υλικού. Στο επίπεδο λογισμικού υπάρχουν πολλά μετροπρογράμματα που μελετούν την απόδοσή του σε διάφορες αρχιτεκτονικές. Σε επίπεδο υλικού, όμως, τα μετροπρογράμματα δεν είναι πολλά κι αυτό μας οδήγησε στη δημιουργία μιας συλλογής για να μπορούμε να μελετάμε και την απόδοση στο υλικό ακόμα και σε σύγκριση με το λογισμικό. Έτσι, μελετώντας τους αλγορίθμους που υπάρχουν στη συλλογή μετροπρογραμμάτων PARBOIL, που είναι για επίπεδο λογισμικού, διαλέξαμε τρεις από αυτούς και τους μετατρέψαμε σε επίπεδο υλικού. Στη συνέχεια τους υλοποιήσαμε για μια συσκευή FPGA και πήραμε τα αποτελέσματά τους για αυτή. Μετρήσαμε, δηλαδή πόσο χώρο πιάνουν στην FPGA, καθώς και τις μέγιστες συχνότητες ρολογιού, στις οποίες μπορούν να δουλέψουν.

# Κεφάλαιο 1

# Εισαγωγή

Η τεχνολογία των υπολογιστών αναπτύσσεται ραγδαία τα τελευταία χρόνια. Ως αποτέλεσμα έχει το υλικό να αποκτά ολοένα και μεγαλύτερες δυνατότητες με ακόμα μικρότερο κόστος και να το εκμεταλλευόμαστε όλο και πιο πολύ. Η απόδοση αυτών των συστημάτων παίζει πολύ σημαντικό ρόλο και γι' αυτό η μέτρησή της γίνεται αντικείμενο μελέτης.

Η μελέτης της απόδοσης των διάφορων υπολογιστικών συστημάτων γίνεται μέσω των μετροπρογραμμάτων. Τα μετροπρογράμματα είναι αλγόριθμοι από διάφορα επιστημονικά πεδία που χρησιμοποιούνται για την σύγκριση διάφορων αρχιτεκτονικών ακόμα και μεταγλωττιστών. Έχοντας ένα κοινό μετροπρόγραμμα η σύγκριση των αποτελεσμάτων από διαφορετικούς ερευνητές γίνεται πιο εύκολη και πιο κατανοητή.

Σε επίπεδο λογισμικού υπάρχουν πολλά πακέτα μετροπρογραμμάτων. Η ανάπτυξη πολυπύρηνων επεξεργαστών, καθώς και των επεξεργαστών GPU, εκτός των μονοπύρηνων υπολογιστών έχει οδηγήσει στην ανάπτυξη μετροπρογραμμάτων τα οποία είναι για όλες αυτές τις αρχιτεκτονικές. Υπάρχουν πια εκδόσεις των μετροπρογραμμάτων τόσο σε παράλληλο όσο και σε σειριακό προγραμματισμό καθώς και σε διάφορες γλώσσες προγραμματισμού. Σε επίπεδο υλικού, όμως, η χρήση τους δεν είναι διαδεδομένη καθώς δεν υπάρχουν πολλοί αλγόριθμοι υλοποιημένοι σε αυτό.

Καθώς, ο τομέας των ενσωματωμένων συστημάτων αναπτύσσεται ολοένα και πιο πολύ η ύπαρξη μετροπρογραμμάτων σε επίπεδο υλικού γίνεται αναγκαία. Οι συσκευές αυτού του τομέα, όπως είναι οι FPGA, χρησιμοποιούνται σε πολλές εφαρμογές καθώς το κόστος τους είναι μικρό, ενώ είτε μπορούν να επαναπρογραμματιστούν πολύ εύκολα (FPGA) είτε είναι οι βέλτιστες για συγκεκριμένες λειτουργίες (ASIC). Η μελέτη της απόδοσης αλγορίθμων σε επίπεδο υλικού για τις διάφορες συσκευές είναι σημαντική για τη σύγκριση αυτών των συσκευών έτσι ώστε να μπορέσει κάποιος να κατανοήσει είτε ποια συσκευή είναι η βέλτιστη για τον σκοπό που τη θέλει είτα πώς μπορεί να υλοποιήσει τον αλγόριθμό του με βέλτιστο τρόπο για μια συγκεκριμένη συσκευή.

Η έλλειψη διάφορων αλγορίθμων σε επίπεδο υλικού ήταν και το έναυσμα για αυτήν την διπλωματική εργασία. Σκοπός της είναι η δημιουργία μιας συλλογής μετροπρογραμμάτων σε επίπεδο υλικού. Τα μετροπρογράμματα με τα οποία ασχοληθήκαμε υπάρχουν ήδη σε επίπεδο λογισμικού για διάφορες αρχιτεκτονικές, έτσι ώστε να μπορεί να γίνει σύγκριση και με αυτές. Η συλλογή από την οποία επιλέξαμε τους αλγορίθμους είναι το PARBOIL.

Το PARBOIL έχει αρκετούς αλγορίθμους στη συλλογή του και καλύπτει ένα μεγάλο εύρος επιστημονικών πεδίων. Επίσης, κάθε αλγόριθμος είναι υλοποιημένος σε διάφορες γλώσσες προγραμματισμού, τόσο για μονοπύρηνους όσο και πολυπύρηνους επεξεργαστές.

Οι αλγόριθμοι που επιλέξαμε εμείς είναι ο γρήγορος μετασχηματισμός Fourier,το ιστόγραμμα και έναν αλγόριθμο που χρησιμοποιείται στην μαγνητική τομογραφία. Τους σχεδιάσαμε σε επίπεδο υλικού με τη βοήθεια της γλώσσας περιγραφής υλικού VERILOG και τους υλοποιήσαμε για μια συσκευή FPGA. Με τη βοήθεια του εργαλείου XILINX ISE πήραμε τα αποτελέσματα για την FPGA για

κάθε αλγόριθμο. Τα αποτελέσματα μας δείχνουν πόσο χώρο στην FPGA πιάνει το κάθε μετροπρόγραμμα, καθώς και την μέγιστη συχνότητα ρολογιού που μπορεί η κάθε σχεδίαση να δουλέψει.

Στο δεύτερο κεφάλαιο εξηγούμε την τεχνολογία FPGA που χρησιμοποιήσαμε. Αναλύουμε την συσκευή FPGA καθώς και την διαδικασία που ακολουθείται από το εργαλείο XILINX ISE για να υλοποιηθεί ένα μετροπρόγραμμα για αυτή την συσκευή.

Στο τρίτο κεφάλαιο κάνουμε αναφορά στη συλλογή μετροπρογραμμάτων PARBOIL.

Στο τέταρτο κεφάλαιο αναλύουμε τον αλγόριθμο του γρήγορου μετασχηματισμού Fourier, παραθέτουμε τις σχεδιάσεις των διάφορων μονάδων που υλοποιήσαμε και στο τέλος παίρνουμε τα αποτελέσματα για την συσκευή FPGA.

Στο πέμπτο κεφάλαιο εξηγούμε τον αλγόριθμο για το ιστόγραμμα και την σχεδίαση που έγινε σε επίπεδο υλικού για να πάρουμε τα αποτελέσματα στην συσκευή FPGA και για αυτόν τον αλγόριθμο.

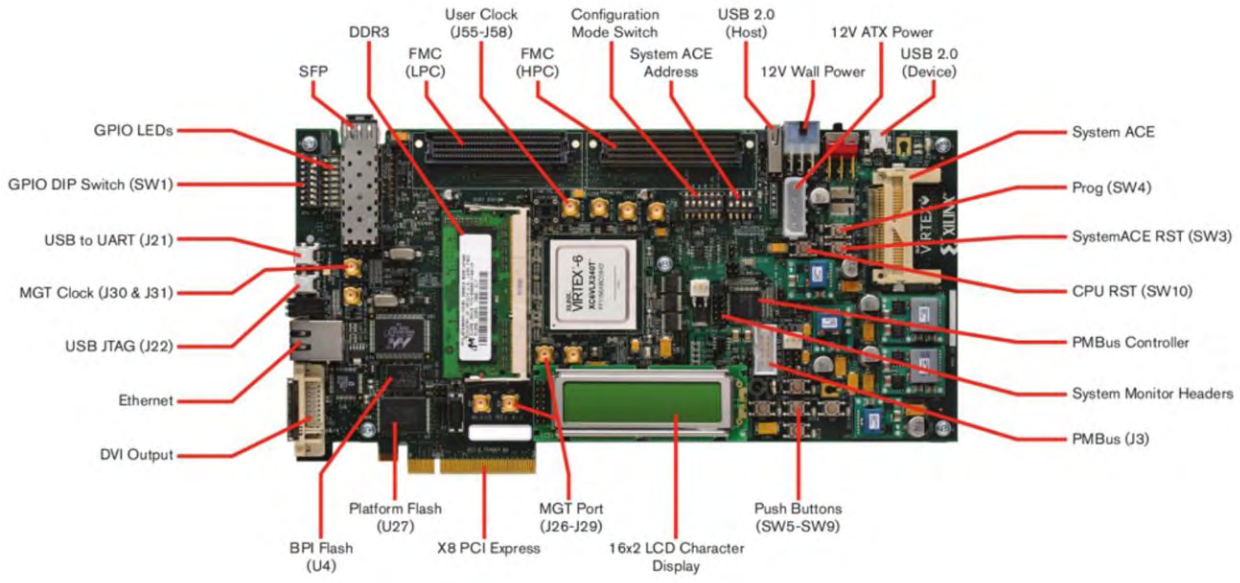
Στο έκτο κεφάλαιο αναλύουμε τον αλγόριθμο που χρησιμοποιείται στην μαγνητική τομογραφία και πιο συγκεκριμένα ο υπολογισμός ενός πίνακα Q. Εξηγούμε τις μονάδες που δημιουργήθηκαν για αυτόν τον αλγόριθμο και γίνεται και παράθεση τον αποτελεσμάτων για την συσκευή FPGA.

# Κεφάλαιο 2

# Τεχνολογία FPGA

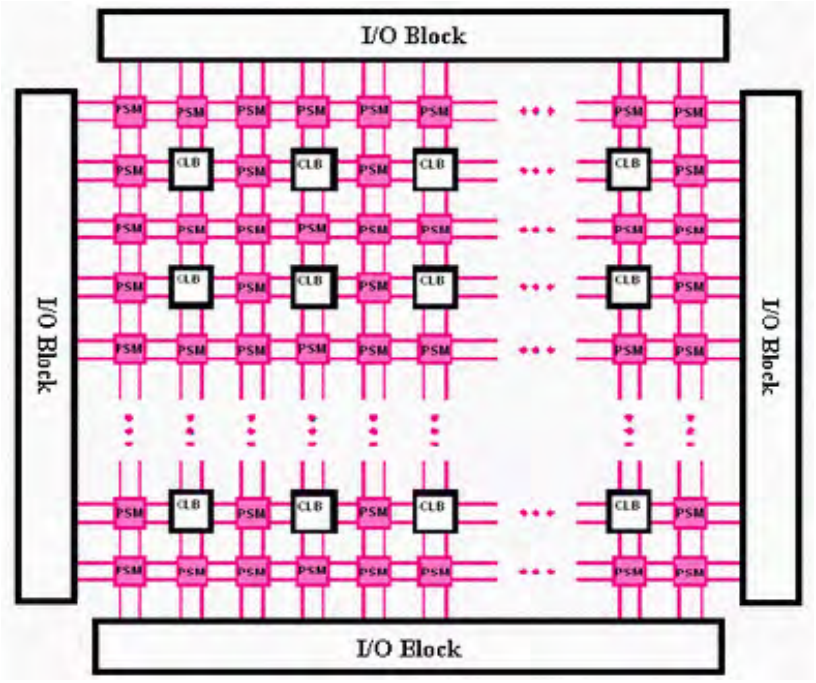
## 2.1 Field – Programmable Gate Array (FPGA)

Μια συσκευή FPGA είναι ένα ολοκληρωμένο κύκλωμα με επαναπρογραμματιζόμενη λογική. Δηλαδή μας δίνει την δυνατότητα εκτός από το λογισμικό να αλλάζουμε και το υλικό με σκοπό να υλοποιείται από αυτό η σχεδίαση που θέλουμε. Δεν υπάρχει όριο στο πόσες φορές μπορεί να προγραμματιστεί, ενώ χρειάζεται μνήμη για μόνιμη συγκράτηση δεδομένων σε περίπτωση που θέλουμε να συγκρατεί τον προγραμματισμό του, επειδή τον χάνει κάθε φορά που διακόπτεται η τάση τροφοδοσίας. Μια FPGA προγραμματίζεται με γλώσσα περιγραφής υλικού, όπως VERILOG και VHDL.




**Σχήμα 2.1 Συσκευή FPGA**

Οι συσκευές FPGA έχουν ένα μεγάλο αριθμό τυποποιημένων πυλών και άλλων ψηφιακών λειτουργιών, όπως μνήμες, καθώς και διάφορες διασυνδέσεις μεταξύ αυτών των στοιχείων, έτσι ώστε όταν προγραμματίζονται να επιτελούν μια συγκεκριμένη λειτουργία, όπως κάνει ένα ολοκληρωμένο κύκλωμα. Οι συσκευές FPGA αποτελούνται από τρία κύρια χαρακτηριστικά: από τα Configurable Logic Blocks (CLBs), τα Programmable Switch Matrices (PSMs) και blocks εισόδων κι εξόδων. Το παρακάτω σχήμα (Σχήμα 2.2) είναι ένα γενικό σχεδιάγραμμα μιας FPGA για τα χαρακτηριστικά αυτά.



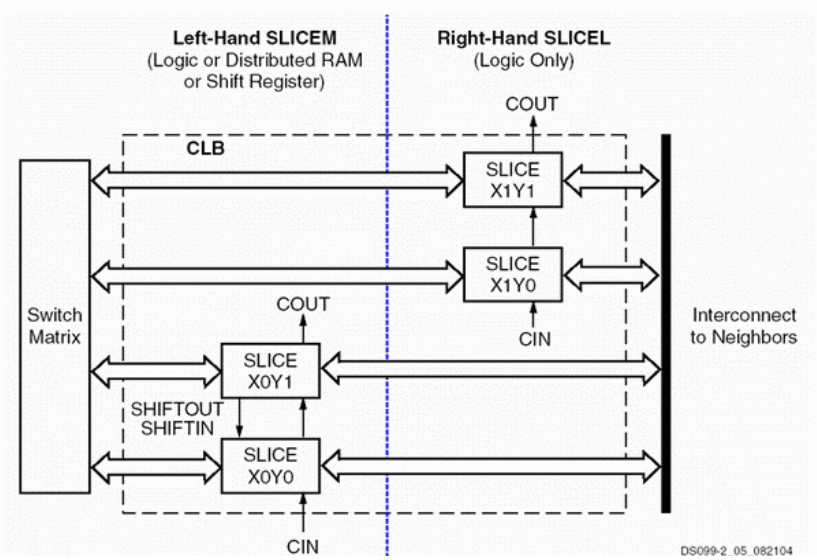

**Σχήμα 2.2 Δομικά στοιχεία μιας FPGA**

Πολλά Configurable Logic Blocks συνδέονται μεταξύ τους για να υλοποιήσουν την σχεδίαση, την οποία φτιάχνει ο χρήστης της FPGA. Ένα CLB αποτελείται από slices και επικοινωνεί με τα γειτονικά CLBs μέσω των Programmable Switch Matrices. Τα PSMs είναι ουσιαστικά «διακόπτες» που καθορίζουν τις διασυνδέσεις μεταξύ των CLBs κι των block εισόδων/εξόδων κι αλλάζουν αυτές τις διασυνδέσεις αναλόγως τον προγραμματισμό που γίνεται. Ένα σχεδιάγραμμα ενός CLB, το οποίο ανήκει σε μια FPGA της οικογένειας VIRTEX4, φαίνεται στο σχήμα 2.3.

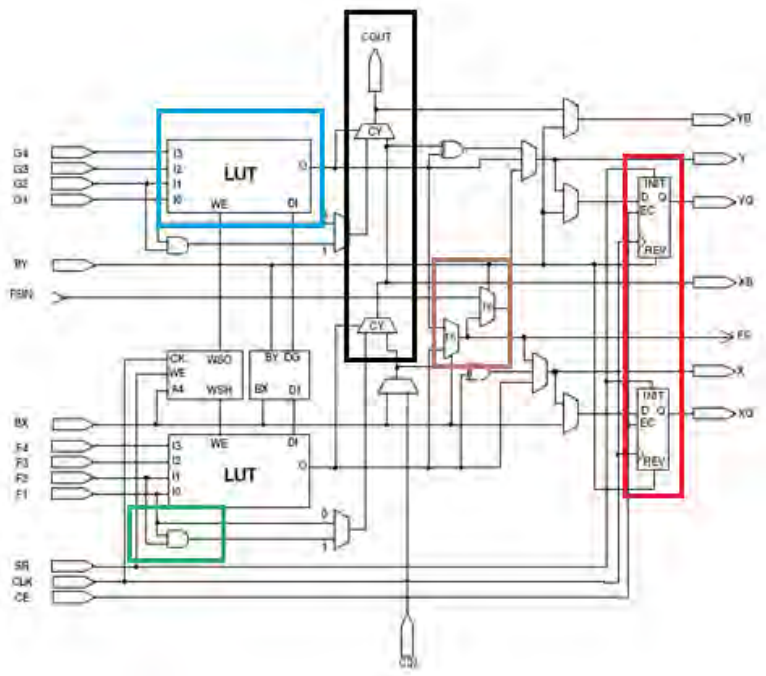
Με αυτόν τον τρόπο φτάνουμε στο μικρότερο προγραμματιζόμενο λογικό κελί που υπάρχει σε μια FPGA και είναι το slice (Σχήμα 2.4). Ένα slice αποτελείται από N-εισόδων Look up Tables, τα οποία ουσιαστικά είναι μια μνήμη με πίνακες αληθείας για την υλοποίηση συνδυαστικών boolean συναρτήσεων N εισόδων, λογικές πύλες, λογική κρατουμένου, πολυπλέκτες και Flip Flops και latches. Για κάθε σχεδίαση που φτιάξαμε μετρήσαμε αυτά τα λογικά κελιά που αυτή χρειάστηκε.

Εκτός από τα παραπάνω τμήματα, μια συσκευή FPGA έχει κι άλλα στοιχεία. Έχει επίσης μνήμες τύπου flash, γρήγορες μνήμες BRAM και μεγαλύτερη μνήμη, αλλά πιο αργή SDRAM. Επίσης, έχει υποδοχές για εξωτερικές συσκευές, διάφορους διακόπτες, leds, ακόμα και μικρές οθόνες.

Οι συσκευές FPGA χρησιμοποιούνται σε ένα μεγάλο εύρος εφαρμογών. Μερικοί από αυτούς είναι η επεξεργασία σημάτων, η επεξεργασία εικόνας, αναγνώριση φωνής, αστρονομία, κρυπτογραφία και βιοπληροφορική.




**Σχήμα 2.3 Configurable Logic Block μιας VIRTEX 4 FPGA**




**Σχήμα 2.4 Slice μιας VIRTEX4 FPGA. Με πράσινο χρώμα φαίνεται μια πύλη, με μαύρο η λογική κρατουμένου, με κόκκινο τα Flip Flops, με καφές πολυπλέκτες και με μπλε τα Look-up Tables.**

## 2.2 FPGA design flow

Μετά τη σχεδίαση μιας μονάδας υλικού σε γλώσσα περιγραφής υλικού ακολουθεί μια διαδικασία, με σκοπό το τελικό αποτέλεσμα να διαμορφώσει έτσι την FPGA ώστε να επιτελεί τη λειτουργία αυτής της μονάδας. Αυτή την διαδικασία την κάνουμε σε ένα περιβάλλον ανάπτυξης της FPGA.

Αφού έχουμε γράψει τη σχεδίασή μας σε μια γλώσσα περιγραφής υλικού, προσθέτουμε το αρχείο σε ένα καινούριο project στο παραπάνω εργαλείο και επιλέγουμε την FPGA για την οποία θα γίνει η υλοποίηση καθώς κι ότι άλλες ρυθμίσεις θέλουμε. Έπειτα κάνουμε τα παρακάτω.
Η διαδικασία που ακολουθούμε είναι η εξής:

1. Synthesis
2. Implementation
    a. Translate
    b. Mapping
    c. Place and Route
3. Bit generation

Με το *Synthesis* παράγεται το netlist, δηλαδή αντιστοιχίζεται η σχεδίασή μας σε λογικές πύλες, LUTs, Flop/ Flops κι ότι άλλο χρειάζεται σε αυτό το επίπεδο.

Κατά την διαδικασία του *Translate* ελέγχεται κατά πόσο το netlist είναι συνεπές με την αρχιτεκτονική της FPGA που έχουμε επιλέξει. Το *Mapping* παίρνει την έξοδο του Translate και υπολογίζει πόσα Configurable Logic Blocks θα χρειαστεί η σχεδίαση για την FPGA που έχει επιλεχθεί, ενώ στη συνέχεια η διαδικασία του *Place and Route* τοποθετεί τα CLBs που βρήκε η διαδικασία Mapping στην αντίστοιχα της FPGA και τα συνδέει μεταξύ τους με βέλτιστο τρόπο.

Με τη διαδικασία *Bit generation* δημιουργείται ένα αρχείο bit – stream, το οποίο περιέχει την έξοδο ουσιαστικά της διαδικασίας Place and Route, έτοιμο να φορτωθεί στην FPGA κι έτσι αυτή θα υλοποιεί την σχεδίαση που έχουμε κάνει.

Με την περάτωση των διαδικασιών του Synthesis και του Implementation παίρνουμε τα αποτελέσματα για την σχεδίασή μας, τα οποία δείχνουν πληροφορίες για το πόσο χώρο πιάνει στην FPGA, όπως slices, LUTs, Flip Flops, καθώς και την μικρότερη περίοδο του ρολογιού που μπορεί να έχει.

Εμείς μετά από τις περιγραφές των μονάδων που υλοποιήσαμε θα παραθέτουμε και πίνακες με αυτά τα αποτελέσματα.

## 2.3 Τι χρησιμοποιήσαμε

<u>Γλώσσα Περιγραφής Υλικού</u>

Η γλώσσα περιγραφής υλικού που χρησιμοποιήσαμε είναι η VERILOG

<u>Αριθμοί Κινητής Υποδιαστολής</u>

Για κάποιες μονάδες που σχεδιάσαμε ήταν απαραίτητη η υποστήριξη για αριθμούς κινητής υποδιαστολής. Ακολουθήσαμε το πρότυπο IEEE 754 και χρησιμοποιήσαμε μονάδες πράξεων κινητής υποδιαστολής για πρόσθεση, πολλαπλασιασμό και διαίρεση. Όλες οι μονάδες πράξεων είναι fully pipelined, δηλαδή σε κάθε κύκλο μηχανής μπορούν να βγάζουν ένα καινούριο αποτέλεσμα. Η μονάδα

πολλαπλασιασμού έχει latency πέντε κύκλους μηχανής κι αυτή της διαίρεσης έχει εννιά κύκλους μηχανής. Για την πρόσθεση έχουμε χρησιμοποιήσει μια μονάδα με latency τέσσερις κύκλους μηχανής και μία με τρεις κύκλους μηχανής.

Να σημειώσουμε εδώ ότι αυτές οι μονάδες παίρνουν ως είσοδο αριθμούς 34 bit κι ως έξοδο αριθμούς με άλλα τόσα bit. Κι αυτό, προσθέτουμε αριστερά των δύο πιο σημαντικών bit του αριθμού κινητής υποδιαστολής άλλα δύο bit που υποδεικνύουν, αν ο αριθμός είναι έγκυρος, μηδέν ή άπειρο.

Όπου χρειαστήκαμε αφαίρεση, την υλοποιήσαμε με την βοήθεια πυλών XOR. Κάναμε, δηλαδή xor τον αφαιρετέο με το $2^{31}$, έτσι ώστε να όταν ο αφαιρετέος είναι θετικός αριθμό να γίνεται το πρόσημό του αρνητικό (1) κι όταν είναι αρνητικός να γίνεται το πρόσημό του θετικό (0).

Συσκευή FPGA

Για την διεκπεραίωση αυτής της διπλωματική εργασίας επιλέξαμε μια συσκευή FPGA της οικογένειας VIRTEX6, η XC6VLX760.

Περιβάλλον ανάπτυξης FPGA

Το περιβάλλον ανάπτυξης FPGA που χρησιμοποιήσαμε για τις διαδικασίες του Synthesis, Mapping και Place and Route είναι το XILINX ISE, έκδοση 12.4.

# Κεφάλαιο 3

# Parboil Benchmark Suite

Οι αλγόριθμοι, οι οποίοι μετατράπηκαν σε επίπεδο υλικού για αυτήν την διπλωματική, είναι από ένα πακέτο αλγορίθμων, το PARBOIL. Το PARBOIL είναι ένα σύνολο μετροπρογραμμάτων γραμμένων σε διάφορες γλώσσες λογισμικού, όπως C, C++, CUDA, OpenMP και OpenCL.

Τα μετροπρογράμματα αυτά είναι χρήσιμα για την μελέτη της απόδοσης διάφορων αρχιτεκτονικών υπολογιστών ή μεταγλωττιστών. Τόσο οι αρχιτεκτονικές και τα προγραμματιστικά μοντέλα, όσο και τα εργαλεία που τα υποστηρίζουν, εξελίσσονται πολύ γρήγορα, γι' αυτό και το PARBOIL δεν είναι ένα "στατικό" σύνολο μετροπρογραμμάτων, αλλά κι αυτό αλλάζει και παρέχει πολλές εναλλακτικές των αλγορίθμων για να είναι χρήσιμο και αποδοτικό. Παρέχει ένα μεγάλο εύρος αλγορίθμων και διάφορες υλοποιήσεις του κάθε αλγορίθμου. Υποστηρίζει, δηλαδή αρχιτεκτονικές τόσο μονοπύρηνες όσο και πολυπύρηνες (CPU και GPU). Έτσι, παρέχονται παράλληλες και σειριακές υλοποιήσεις των αλγορίθμων.

Οι αλγόριθμοι είναι από διάφορα επιστημονικά και εμπορικά πεδία, όπως επεξεργασία εικόνας, αστρονομία, ρευστοδυναμική. Μερικοί από τους αλγορίθμους είναι η αναζήτηση κατά πλάτος, άθροισμα απόλυτων διαφορών, πολλαπλασιασμός πυκνών πινάκων, πολλαπλασιασμός αραιού πίνακα με πυκνό διάνυσμα και MRI gridding.

Εμείς ασχοληθήκαμε με τρεις από τους αλγορίθμους του PARBOIL. Αυτοί είναι:

- ο γρήγορος μετασχηματισμός Fourier
- το ιστόγραμμα
- mri q

# Κεφάλαιο 4

# Γρήγορος Μετασχηματισμός Fourier (FFT)

## 4.1 Εισαγωγή

Ο πρώτος αλγόριθμος των μετροπρογραμμάτων PARBOIL που ασχοληθήκαμε είναι ο γρήγορος μετασχηματισμός Fourier ή Fast Fourier Transform. Ο γρήγορος μετασχηματισμός Fourier είναι ένας αλγόριθμος με τον οποίο μπορούμε πολύ γρήγορα να υπολογίσουμε τον διακριτό μετασχηματισμό Fourier (Discrete Fourier Transform), ο οποίος χρησιμοποιείται σε πολλές εφαρμογές, όπως στη συμπίεση δεδομένων, στον πολλαπλασιασμό πολυωνύμων και μεγάλων ακεραίων και στην επίλυση διαφορικών εξισώσεων.

Ο διακριτός μετασχηματισμός Fourier μπορεί να υπολογιστεί σύμφωνα με τον παρακάτω τύπο:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i 2\pi \frac{k}{N} n}.$$

Με την ανάλυση αυτού του τύπου βλέπουμε ότι χρειαζόμαστε $N^2$ πολλαπλασιασμούς για τον υπολογισμό του διακριτού μετασχηματισμού Fourier μιας ακολουθίας με N στοιχεία. Ο αριθμός αυτός είναι πολύ μεγάλος ειδικά για ακολουθίες μεγάλου μήκους. Γι΄ αυτό το λόγο, το διακριτό μετασχηματισμό Fourier τον υπολογίζουμε με τον γρήγορο μετασχηματισμό Fourier, ο οποίος χρειάζεται $N*\log_2 N$ πολλαπλασιασμούς, αριθμός αισθητά μικρότερος του $N^2$. Ο γρήγορος μετασχηματισμός Fourier λειτουργεί όπως θα εξηγήσουμε παρακάτω.

Ο γρήγορος μετασχηματισμός Fourier βασίζεται στην ιδέα του διαίρει και βασίλευε. Υπάρχουν πολλοί τρόποι υπολογισμού του. Εδώ ακολουθήσαμε αυτόν των Cooley – Turkey και ακολουθεί τον τρόπο με αποδεκάτιση στο χρόνο (decimation in time). Έστω μια ακολουθία με N όρους, με το N να είναι άρτιος αριθμός, για την οποία πρέπει να υπολογίσουμε τον γρήγορο μετασχηματισμό Fourier. Διαιρούμε το N διά του 2 κι έτσι παίρνουμε δύο υποπροβλήματα με N/2 όρους που θα τα συνδυάσουμε κατάλληλα μεταξύ τους για να επιλύσουμε το αρχικό πρόβλημα. Το κάθε υποπρόβλημα το διαιρούμε με το 2 μέχρις ότου φτάσουμε σε ένα υποπρόβλημα με δύο όρους , το οποίο επιλύεται εύκολα. Για να γίνει αυτή η διαδικασία πρέπει το N να είναι δύναμη του 2. Την αρχική ακολουθία την χωρίζουμε σε δύο υπακολουθίες: μία με τους όρους με μονό δείκτη και μία με τους όρους με ζυγό δείκτη. Έτσι, ο τύπος για τον διακριτό μετασχηματισμό Fourier γίνεται:

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m+1)k}.$$

Βγάζοντας κοινό παρονομαστή τον όρο $e^{-\frac{2\pi i}{N}k}$ η εξίσωσή γίνεται:

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N/2}mk} + e^{-\frac{2\pi i}{N}k} \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2}mk}.$$

Παρατηρούμε ότι το πρώτο μέρος της εξίσωσης είναι ο διακριτός μετασχηματισμός Fourier για τους όρους με ζυγό δείκτη (έστω $A_k$)και το δεύτερο μέρος ο αντίστοιχος για τους όρους με μονό δείκτη (έστω $B_k$) πολλαπλασιασμένος με τον όρο $e^{-\frac{2\pi i}{N}k}$. Αυτοί οι δύο διακριτοί μετασχηματισμοί Fourier έχουν μήκος N/2. Στη συνέχεια θα εκμεταλλευτούμε μια ιδιότητα του όρου $e^{-\frac{2\pi i}{N}k}$. Ισχύει ότι:

$e^{(-2\pi i(k-N/2)/N)} = e^{\pi i}e^{(-2\pi ik/N)} = -e^{-\frac{2\pi i}{N}k}$ (αφού $e^{\pi i} = \cos\pi + i\sin\pi = -1$).

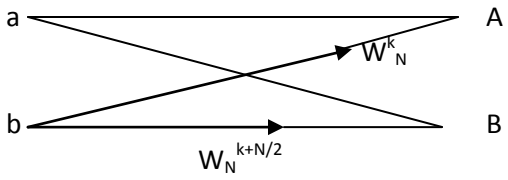
Οπότε ο τύπος για τον διακριτό μετασχηματισμό Fourier γίνεται:

$$X_k = \begin{cases} A_k + e^{-\frac{2\pi i}{N}k} B_k & \text{για } k<N/2 \\ A_{k-N/2} + e^{(-2\pi i(k-N/2)/N)} B_{k-N/2} & \text{για } k\geq N/2 \end{cases}$$

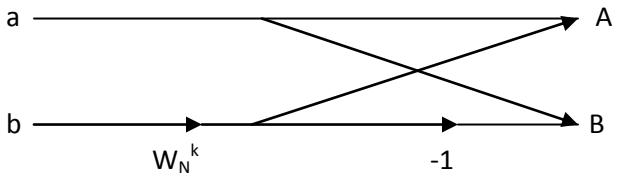
και εφαρμόζοντας την παραπάνω ιδιότητα παίρνουμε τον τελικό τύπο για τον γρήγορο μετασχηματισμό Fourier:

$$X_k = \begin{cases} A_k + e^{-\frac{2\pi i}{N}k} B_k & \text{για } k<N/2. \\ A_{k-N/2} - e^{-\frac{2\pi i}{N}k} B_{k-N/2} & \text{για } k\geq N/2. \end{cases}$$

Αυτό που ουσιαστικά χρησιμοποιεί ο γρήγορος μετασχηματισμός Fourier για να υπολογίσει τον διακριτό μετασχηματισμό Fourier είναι ζευγάρια πράξεων γνωστά και ως "πεταλούδα". Η μονάδα αυτή φαίνεται στο παρακάτω σχεδιάγραμμα. Οι αριθμοί a, b είναι είσοδοι, οι A, B οι έξοδοι και το $W^k_N$ αντιστοιχεί στον παράγοντα $e^{-\frac{2\pi i}{N}k}$.




Χρησιμοποιώντας την ιδιότητα του όρου $e^{-\frac{2\pi i}{N}k}$ που είδαμε και πριν, απλοποιούμε το σχεδιάγραμμα της πεταλούδας ως εξής:




Στον αλγόριθμό μας ως είσοδο θα παίρνουμε αριθμούς τόσο με πραγματικό όσο και με φανταστικό μέρος. Αυτό που μας μένει είναι να αποδείξουμε τις πράξεις με τις οποίες θα υπολογίζουμε τον διακριτό μετασχηματισμό Fourier σύμφωνα με τον γρήγορο μετασχηματισμό Fourier που εξηγήσαμε παραπάνω. Θα το δείξουμε για είσοδο δύο αριθμών που είναι άλλωστε και το μικρότερο υποπρόβλημα σύμφωνα με το οποίο υπολογίζεται αναδρομικά ολόκληρο το πρόβλημα. Έστω οι δύο αριθμοί a = R[a] + j*I[a] και b = R[b] + j*I[b] του παραπάνω σχεδιαγράμματος. Επίσης, έστω ο παράγοντας $W^k_N$ που κι αυτός αποτελείται από πραγματικό μέρος R[W] και φανταστικό μέρος I[W].

$A = a + b*W_N^k$ =>
A = R[a] + j*I[a] +( R[b] + j*I[b])*( R[W] + j* I[W]) =>
A = R[a] + j*I[a] + R[b]*R[W] + R[b]*I[W]*j + I[b]*R[W]*j - I[b]*I[W] =>
A = R[a] + R[W]*R[b] – I[W]*I[b] +(I[a] + I[W]*R[b] +I[b]*R[W])*j

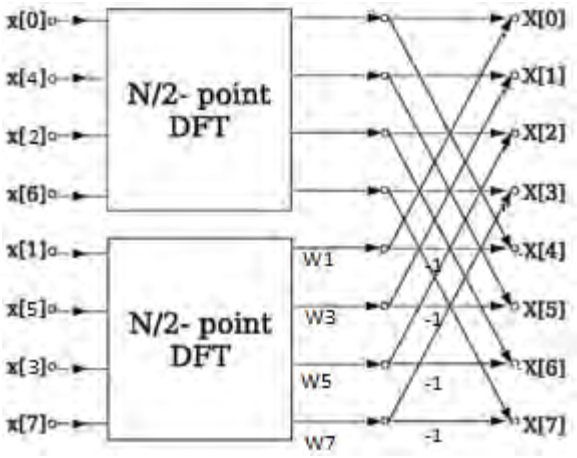
$B = a-b*W_N^k$ =>
B = R[a] + j*I[a] -( R[b] + j*I[b])*( R[W] + j* I[W]) =>
B = R[a] + j*I[a] - R[b]*R[W] - R[b]*I[W]*j - I[b]*R[W]*j + I[b]*I[W] =>
B = R[a] - R[W]*R[b] + I[W]*I[b] +(I[a] - I[W]*R[b] -I[b]*R[W])*j

Άρα, ο διακριτός μετασχηματισμός Fourier σύμφωνα με τον γρήγορο μετασχηματισμό Fourier για τα πραγματικά και φανταστικά μέρη των εισόδων στην μονάδα της πεταλούδας υπολογίζεται από τους παρακάτω τύπους:

R[A] = R[a] + R[W]*R[b] – I[W]*I[b]
I[A] = I[a] + I[W]*R[b] +I[b]*R[W]
R[B] = R[a] - R[W]*R[b] + I[W]*I[b]
I[B] = I[a] - I[W]*R[b] -I[b]*R[W].

Το τελευταίο που πρέπει να σημειώσουμε είναι ότι οι είσοδοι έρχονται με σειρά που υποδεικνύεται από την αντιστροφή των bit του δείκτη του κάθε όρου και όχι με τη σειρά. Παρακάτω φαίνεται ένα παράδειγμα υπολογισμού του διακριτού μετασχηματισμού Fourier μιας ακολουθίας με οκτώ όρους, που δείχνει τόσο την διαίρεση του προβλήματος σε υποπροβλήματα όσο και τη σειρά με την οποία έρχονται οι είσοδοι (Σχήμα 4.1).

Έχουμε W1 = $W^0_8$ =1, W3 = $W^1_8$ = 0.707 – 0.707*j, W5 = $W^2_8$ = -1*j και W7 = $W^3_8$ = -0.707 - 0.707*j. Αν υποθέσουμε ότι κάθε σημείο x αποτελείται από ένα πραγματικό μέρος r κι ένα φανταστικό μέρος i κι ότι κάθε μονάδα N/2-point DFT έχει ως έξοδο τον διακριτό μετασχηματισμό Fourier και των δύο μερών κάθε σημείου x, τότε ο υπολογισμός του τελικού DFT και για τα δύο μέρη των σημείων θα γίνει ως εξής:
Ας ονομάσουμε τις εξόδους των μονάδων N/2-point DFT (r0, i0), (r1, i1), (r2, i2), (r3, i3), (r4, i4), (r5, i5), (r6, i6), (r7, i7). Κάθε ζευγάρι r, i αντιστοιχεί σε μια έξοδο στο παραπάνω σχήμα. Όπως βλέπουμε υπάρχουν τέσσερα ζευγάρια σε μορφή πεταλούδας και σε αυτά θα εφαρμόσουμε τους τέσσερις τύπους που αποδείξαμε σε κάθε ζευγάρι. Άρα θα έχουμε:




**Σχήμα 4.1 Παράδειγμα FFT για είσοδο 8 σημείων**

R[X0] = r0 + r4
I[X0] = i0 + i4
R[X4] = r0 – r4
I[X4] = i0 – i4

R[X1] = r1 + 0.707*r5 + 0.707*i5
I[X1] = i1 + 0.707*i5 – 0.707*r5
R[X5] = r1 – 0.707*r5 – 0.707*i5
I[X5] = i1 – 0.707*i5 + 0.707*r5

R[X2] = r2 + i6
I[X2] = i2 - r6
R[X6] = r2 – r6
I[X6] = i2 – i6

R[X3] = r3 - 0.707*r7 + 0.707*i7
I[X3] = i3 - 0.707*i7 – 0.707*r7
R[X7] = r3 + 0.707*r7 - 0.707*i7
I[X7] = i3 + 0.707*i7 + 0.707*r7

## 4.2 Σχεδίαση

Για να υλοποιήσουμε τον γρήγορο μετασχηματισμό Fourier σε επίπεδο υλικού στηριχτήκαμε στους τύπους για τον υπολογισμό του διακριτού μετασχηματισμού Fourier των πραγματικών και φανταστικών μερών των εισόδων που αποδείξαμε παραπάνω. Υλοποιήσαμε πέντε μονάδες που η καθεμία υπολογίζει τον γρήγορο μετασχηματισμό Fourier για 2, 4, 8, 16 και 32 σημεία αντίστοιχα. Για κάθε σημείο στις μονάδες εισάγεται το πραγματικό και το φανταστικό μέρος των σημείων. Τα σημεία έρχονται στη μονάδα με σειρά που υποδεικνύει η αναστροφή του δείκτη του σημείου, όπως

περιγράψαμε πριν, κι εξάγονται από αυτή με την κανονική σειρά. Και οι πέντε μονάδες δουλεύουν για αριθμούς κινητής υποδιαστολής. Έτσι, για τις πράξεις της πρόσθεσης, της αφαίρεσης και του πολλαπλασιασμού χρησιμοποιήθηκαν αντίστοιχες μονάδες για πράξεις με αριθμούς κινητής υποδιαστολής. Η μονάδα πολλαπλασιασμού χρειάζεται 5 κύκλους μηχανής για να βγάλει αποτέλεσμα κι η μονάδα πρόσθεσης άλλους 4. Για τον ίδιο λόγο οι είσοδοι των μονάδων είναι 32 bit ακολουθώντας το πρότυπο IEEE 754. Ενώ οι έξοδοι είναι 34 bit, γιατί τα δύο πιο σημαντικά ψηφία δείχνουν κατά πόσο είναι έγκυρος ο αριθμός, λειτουργία που γίνεται από τις μονάδες κινητής υποδιαστολής. Κάθε μονάδα περιλαμβάνει την αμέσως προηγούμενη της (δηλαδή η μονάδα για τα 4 σημεία περιλαμβάνει δυο μονάδες των 2 σημείων, η μονάδα για τα 8 περιλαμβάνει δυο μονάδες των 4 σημείων κ.ο.κ.) και μετά γίνονται οι πράξεις σύμφωνα με τους τύπους που προείπαμε και όπως φαίνεται στο παράδειγμα της εισαγωγής.

Οι μονάδες έχουν ως είσοδο σημεία και ως έξοδο τον γρήγορο μετασχηματισμό Fourier των εισόδων. Ένας άλλος τρόπος σχεδίασης θα ήταν να έχουμε μόνο ένα σημείο ως είσοδο κι άλλο ένα ως έξοδο για να έχουμε λιγότερες εισόδους κι εξόδους, αλλά αυτό θα δούλευε μόνο για μια ακολουθία σημείων που όλα έχουν την ίδια τιμή. Διαφορετικός τρόπος σχεδίασης ήταν με BRAMs αντί για εισόδους, στις οποίες θα αποθηκεύαμε εκεί τα σημεία. Με αυτό τον τρόπο όμως θα χρειαζόμασταν πολλές BRAMs, αφού θέλαμε από μία για κάθε πραγματικό και φανταστικό μέρος τόσο των εισόδων όσο και των εξόδων, έτσι ώστε να παίρναμε ταυτόχρονα τα δεδομένα από αυτές. Τέλος, εντελώς διαφορετική σχεδίαση θα ήταν να ακολουθούσαμε την αποδεκάτιση στη συχνότητα, που σημαίνει ότι τα σημεία έρχονται με την κανονική σειρά, γίνονται πρώτα οι πολλαπλασιασμοί κι οι προσθέσεις και μετά το αποτέλεσμα να έμπαινε στον μηχανισμό της πεταλούδας. Έτσι, όμως, θα παίρναμε στις εξόδους τους μετασχηματισμένους αριθμούς με την ανάστροφή σειρά των δεικτών τους.

Όλες οι μονάδες έχουν μια αρχική καθυστέρηση (latency), αλλά βγάζουν αποτέλεσμα σε κάθε κύκλο μηχανής αν εισέρχονται δεδομένα σε κάθε κύκλο μηχανής.

### 4.2.1 FFT για 2 σημεία

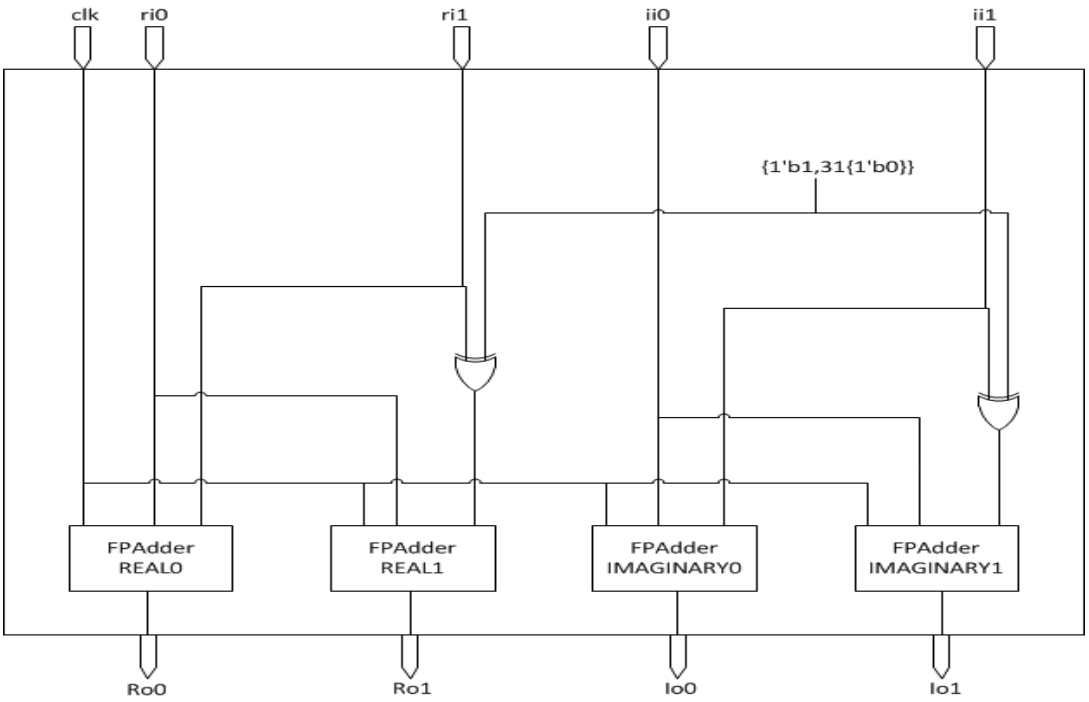
Η μονάδα αυτή υπολογίζει τον γρήγορο μετασχηματισμό Fourier για 2 σημεία. Ως εισόδους έχει μία του ενός bit για το ρολόι, τέσσερις των 32 bit για τα πραγματικά και φανταστικά μέρη των σημείων που εισέρχονται στη μονάδα και άλλες τέσσερις των 34 bit για τα πραγματικά και φανταστικά μέρη των σημείων που έχουν μετασχηματιστεί και εξέρχονται από την μονάδα.

Η σχεδίαση για αυτή τη μονάδα είναι ουσιαστικά η υλοποίηση μιας μονάδας πεταλούδας που περιγράψαμε παραπάνω με $W_N^k = W^0_2 = e^0 = 1$. Η μονάδα αυτή λέγεται fft2 για να βοηθηθούμε στη συνέχεια και φαίνεται στο σχήμα 4.2.

Βλέπουμε ότι χρησιμοποιήθηκαν τέσσερις προσθετές κινητής υποδιαστολής για να γίνουν οι πράξεις της αφαίρεσης και της πρόσθεσης. Για να γίνει η αφαίρεση απλά αλλάζουμε το πρόσημο του αριθμού με μια XOR πύλη. Οι πράξεις που υλοποιούνται είναι σύμφωνα με τους τύπους οι εξής:

Ro0 = ri0 + ri1
Io0 = ii0 +ii1
Ro1 = ri0 – ri1
Io 1 = ii0 - ii1

Οι μονάδες FPAdder REAL0 και FPAdder IMAGINARY0 βγάζουν τον διακριτό μετασχηματισμό Fourier του πραγματικού και φανταστικού μέρους αντίστοιχα του σημείου με δείκτη 0 και οι μονάδες FPAdder REAL1 και FPAdder IMAGINARY1 τον αντίστοιχο μετασχηματισμό του σημείο με δείκτη 1. Η μονάδα αυτή χρειάζεται τέσσερις κύκλους μηχανής για να βγάλει αποτέλεσμα.




**Σχήμα 4.2 Μονάδα FFT για 2 σημεία**

### 4.2.2 FFT για 4 σημεία

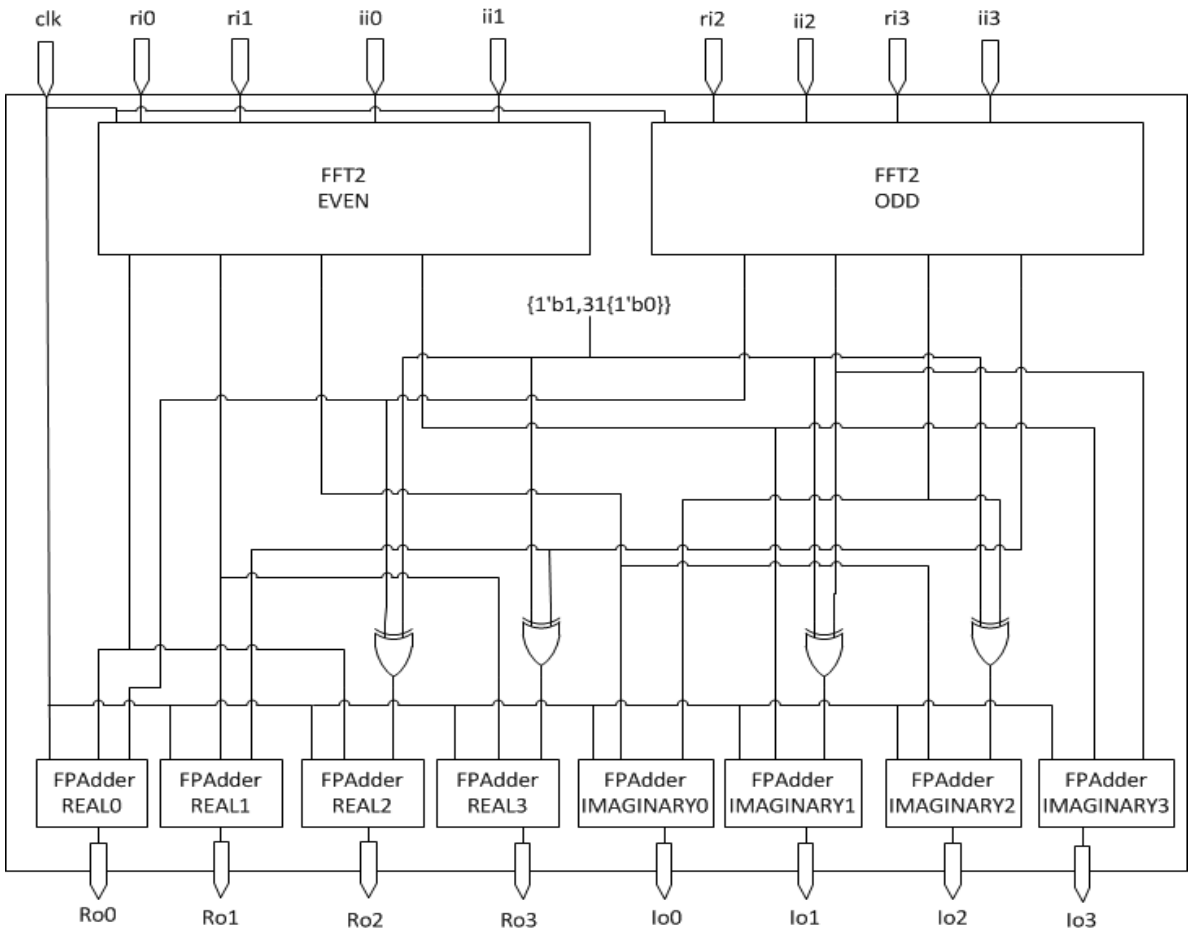
Αυτή η μονάδα υπολογίζει τον γρήγορο μετασχηματισμό Fourier για 4 σημεία. Ως εισόδους έχει μία του 1 bit για το ρολόι, οχτώ των 32 bit για τα πραγματικά και φανταστικά μέρη των σημείων της εισόδου και οχτώ εξόδους των 34 bit για τα αντίστοιχα μέρη των σημείων που έχουν μετασχηματισθεί. Η σχεδίαση της μονάδας φαίνεται στο σχήμα 4.3.

Όπως βλέπουμε χρησιμοποιούμε δύο μονάδες fft2, μία για τους όρους με ζυγό δείκτη και μία για αυτούς με μονό δείκτη. Έπειτα εφαρμόζοντας τους τύπους στα ζευγάρια που δημιουργούνται με $W^0_4 = 1$ και $W^1_4 = -1*j$, παίρνουμε το κύκλωμα που φαίνεται στο σχήμα. Οι πράξεις είναι οι εξής:

Ro0 = ro0 + ro2
Io0 = io0 + io2
Ro1 = ro1 + io3
Io 1 = io1 – ro3
Ro2 = ro0 - ro2
Io2 = io0 - io2
Ro3 = ro1 - io3
Io 3 = io1 + ro3

Το κύκλωμα αυτό χρειάζεται οχτώ κύκλους μηχανής για να βγάλει αποτέλεσμα. Όπως και στη μονάδα fft2 από τις μονάδες FPAdders REAL 0 έως 1 και IMAGINARY 0 έως 1 εξάγονται τα τελικά αποτελέσματα του γρήγορου μετασχηματισμού Fourier με την κανονική σειρά που υποδηλώνεται από τους δείκτες των αριθμών. Οι μεταβλητές ro0-3 και io0-3 είναι ουσιαστικά οι έξοδοι των δύο μονάδων fft2. Δηλαδή

η μονάδα FFT2 EVEN έχει ως έξοδο τις μεταβλητές ro0, ro1, io0 και io1 και η μονάδα FFT2 ODD έχει ως έξοδο τις μεταβλητές ro2, ro3, io2 και io3. Για τη συνέχεια αυτή τη μονάδα την ονομάζουμε fft4.




**Σχήμα 4.3 Μονάδα FFT για 4 σημεία**

### 4.2.3 FFT για 8 σημεία

Όπως και οι προηγούμενες μονάδες, έτσι κι αυτή έχει μια είσοδο του 1 bit για το ρολόι. Οι υπόλοιπες δεκαέξι είσοδοι είναι των 32 bit και οι δεκαέξι έξοδοι είναι των 34 bit. Συνεχίζοντας την εφαρμογή της θεωρίας, κι αυτή η μονάδα περιλαμβάνει δύο μονάδες fft4 κι από κει κι έπειτα γίνεται εφαρμογή των τύπων στα υπάρχοντα ζευγάρια για να γίνει ο σχεδιασμός της.

Η μονάδα αυτή είναι το παράδειγμα στην εισαγωγή κι οι τύποι φαίνονται εκεί. Έτσι, σε αυτή τη μονάδα έχουν προστεθεί τέσσερις μονάδες πολλαπλασιασμού κινητής υποδιαστολή για να σχηματιστεί το γινόμενο των W = 0.707 με τους όρους που πρέπει για να συνεχίσουμε με τις προσθέσεις. Οι οχτώ FPAdders TR1, TR3, TR5, TR7, TI1, TI3, TI5 και TI7 χρησιμοποιούνται για γίνουν οι προσθέσεις στους τύπους που προσθέτονται τρεις όροι. Έτσι, με αυτές τις μονάδες κάνουμε πρώτα τις προσθέσεις με τους δύο όρους και για να βγει το τελικό αποτέλεσμα προσθέτουμε αυτό το προσωρινό άθροισμα με τον τρίτο όρο του τύπου στον επόμενο FPAdder. Πάλι χρησιμοποιούμε πύλες XOR για τις αφαιρέσεις και παίρνουμε τον τελικό διακριτό μετασχηματισμό Fourier από τις μονάδες FPAdd REAL 0 έως 7 και FPAdd IMAG 0 έως 7.

Στο σχήμα 4.4 βλέπουμε ότι υπάρχουν και κάποιοι καταχωρητές, οι οποίοι είναι τα ορθογώνια με το χαρακτηριστικό reg ή r μπροστά από την υπόλοιπη περιγραφή. Αυτοί οι καταχωρητές κρατούν τα σήματα για όσους κύκλους χρειάζεται μέχρι να χρησιμοποιηθούν από τις μονάδες κινητής υποδιαστολής κι είναι 34 bit. Για παράδειγμα η μεταβλητή ro1 μένει στους αντίστοιχους καταχωρητές για πέντε κύκλους, όσους χρειάζεται δηλαδή η μονάδα του πολλαπλασιασμού να βγάλει αποτέλεσμα για να προστεθούν η ro1 με το αντίστοιχο γινόμενο στη μονάδα FPAdder TR1. Στο σχήμα δεν φαίνονται όλοι οι καταχωρητές λόγω έλλειψης χώρου, αλλά μπορούμε εύκολα να καταλάβουμε ποιοι είναι. Βρίσκονται σε ζευγάρια και στο σχήμα υπάρχει ο πρώτος κι ο τελευταίος καταχωρητής για κάθε μεταβλητή κι ανάμεσά τους έχουμε τοποθετήσει διακεκομμένο καλώδιο. Για να μην υπάρχουν τόσοι καταχωρητές θα μπορούσαμε να χρησιμοποιήσουμε FSM, αλλά έτσι θα χάναμε σε throughput.

Η μονάδα fft8 χρειάζεται 21 κύκλους μηχανής για να βγάλει αποτέλεσμα.

### 4.2.4 FFT για 16 σημεία

Αυτή η μονάδα υπολογίζει τον γρήγορο μετασχηματισμό Fourier για δεκάξι σημεία. Έχει τριάντα δύο εισόδους των 32 bit, μία του 1 bit και τριάντα δύο εξόδους των 34 bit. Όπως κι οι προηγούμενες μονάδες, έτσι κι αυτή έχει δύο μονάδες fft8 στις εξόδους των οποίων εφαρμόζονται οι μαθηματικοί τύποι που έχουμε αποδείξει για την εξαγωγή του τελικού αποτελέσματος.

Τα W που χρειαζόμαστε είναι τα εξής:

$W^0_{16} = 1$, $W^1_{16} = 0.9239 - 0.3827* j$, $W^2_{16} = 0.707 – 0.707*j$, $W^3_{16} = 0.3827 – 0.9239*j$,
$W^4_{16} = -1*j$, $W^5_{16} = 0.3827 – 0.9239*j$, $W^6_{16} = -0.707 – 0.707$, $W^7_{16} = 0.9239 – 0.3827*j$.

Η σχεδίαση δεν θα γίνει καθώς είναι παρόμοια με αυτή της μονάδας fft8 απλά έχει περισσότερες μονάδες κινητής υποδιαστολής για τις απαραίτητες πράξεις και περισσότερους καταχωρητές. Πάλι χρησιμοποιούμε μονάδες πολλαπλασιασμού για το γινόμενο με τα W και μονάδες πρόσθεσης για να υπολογίσουμε τα αθροίσματα με τον ίδιο τρόπο που τα υπολογίσαμε και στη μονάδα fft8. Επίσης, υπάρχουν οι κατάλληλοι καταχωρητές με εύρος 34 bit που κρατάν ε τα σήματα για όσους κύκλους χρειάζονται μέχρι να χρησιμοποιηθούν για να πάρουμε το σωστό αποτέλεσμα και οι απαραίτητες πύλες XOR για να βοηθήσουν στην πράξη της αφαίρεσης, όπου είναι απαραίτητο.

Η μονάδα fft16 χρειάζεται 34 κύκλους μηχανής για να βγάλει αποτέλεσμα.

### 4.2.5 FFT για 32 σημεία

Η τελευταία μονάδα για τον γρήγορο μετασχηματισμό Fourier που υλοποιήσαμε είναι αυτή για τα τριάντα δύο σημεία. Έχει μια είσοδο του 1 bit για το ρολόι, εξήντα τέσσερις εισόδους των 32 bit για τα πραγματικά και φανταστικά μέρη των σημείων και εξήντα τέσσερις εξόδους των 34 bit για τα αντίστοιχα μέρη των μετασχηματισμένων σημείων. Κι αυτή η μονάδα έχει δύο μονάδες fft16 κι έχει σχεδιαστεί, έτσι ώστε να υλοποιούνται οι πράξεις της πεταλούδας ανάμεσα στα ζευγάρια που έχουν δημιουργηθεί από τις εξόδους των μονάδων fft16 .

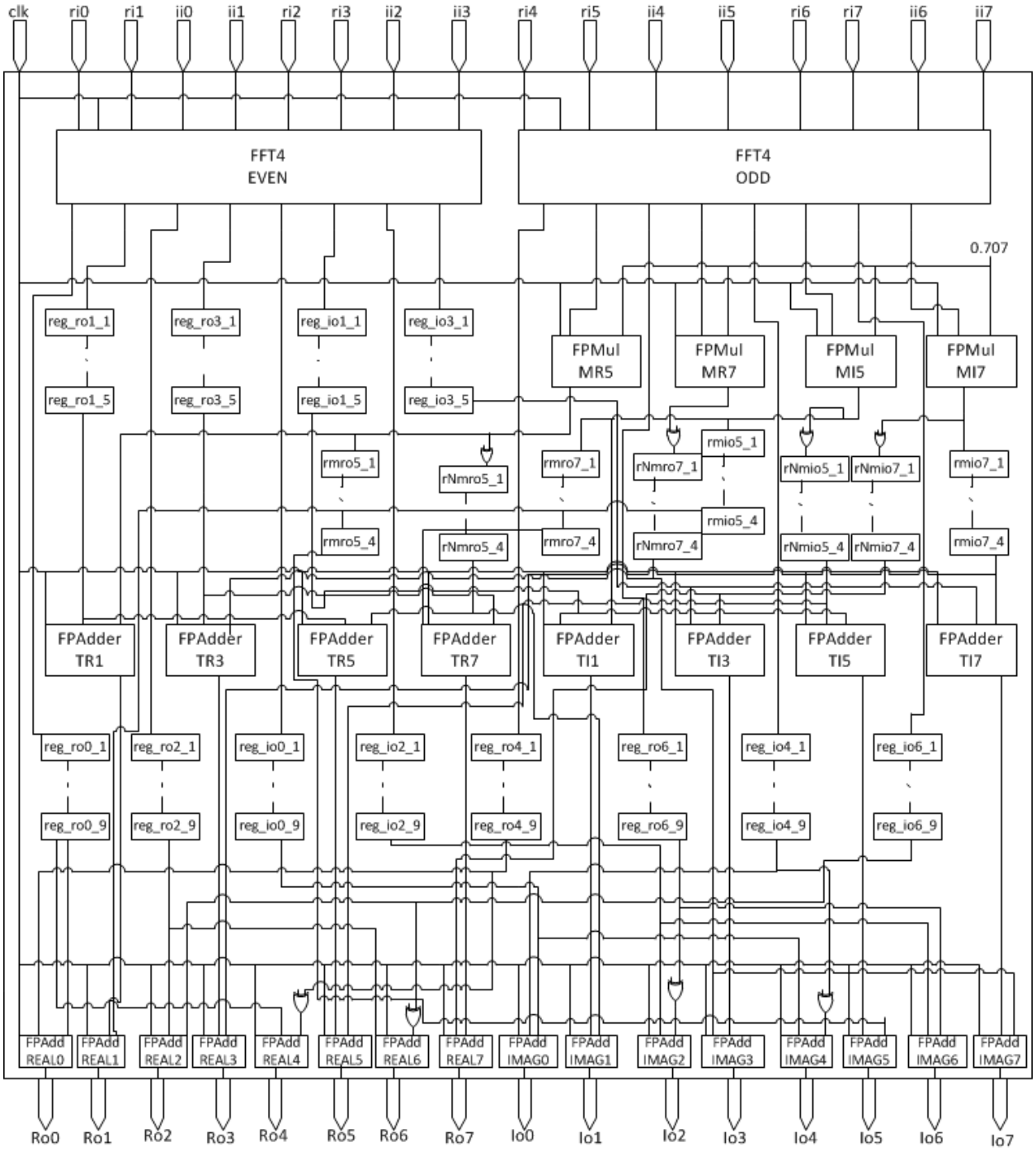
Τα W που χρησιμοποιεί αυτή η μονάδα είναι τα εξής:

$W^0_{32} = 1$, $W^1_{32} = 0.9808 - 0.1951 * j$, $W^2_{32} = 0.9239 – 0.3827*j$,
$W^3_{32} = 0.8315 – 0.5556*j$, $W^4_{32} = 0.707 - 0.707*j$, $W^5_{32} = 0. 5556 – 0.8315*j$,
$W^6_{32} = 0.3827 – 0.9239*j$, $W^7_{32} = 0.1951 – 0.9808*j$, $W^8_{32} = -1*j$,
$W^9_{32} = -0.1951 - 0.9808*j$, $W^{10}_{32} = 0.3827 – 0.9239*j$, $W^{11}_{32} = -0.5556 – 0.8315*j$,
$W^{12}_{32} = -0.707 - 0.707*j$, $W^{13}_{32} = -0.8315 – 0.5556*j$, $W^{14}_{32} = 0.9239 – 0.3827*j$,
$W^{15}_{32} = -0.9808 – 0.1951*j$.

Η σχεδίαση της μονάδας έγινε με τον ίδιο τρόπο που σχεδιάστηκαν κι οι προηγούμενες μονάδες και λόγω της περιπλοκότητάς της δεν θα παραθέσουμε το σχεδιάγραμμά της. Απλά γίνεται

εφαρμογή των τύπων της πεταλούδας και χρησιμοποιούμε μονάδες πολλαπλασιασμού κινητής υποδιαστολής για τα γινόμενα με τα W και όσες μονάδες πρόσθεσης είναι απαραίτητες για τα αθροίσματα. Επίσης, έχουμε καταχωρητές των 34 bit για αν κρατούν τα σήματα για όσους κύκλους μηχανής χρειάζεται και πύλες XOR για να βοηθούν στην πράξη της αφαίρεσης.

Η μονάδα fft32 χρειάζεται 47 κύκλους μηχανής για να βγάλει αποτέλεσμα.

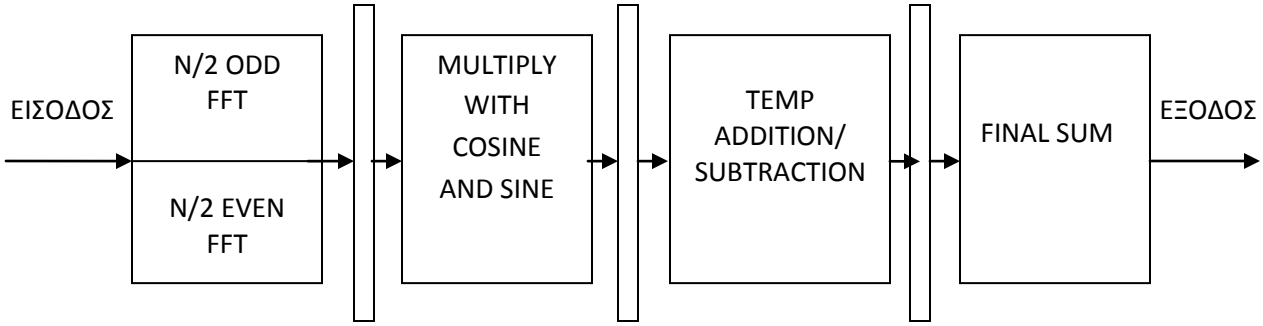



**Σχήμα 4.4 Μονάδα FFT για 8 σημεία**

### 4.2.6 Γενική μορφή – Στάδια pipeline

Σύμφωνα με τα παραπάνω , η σχεδίαση για τις μονάδες έχει γίνει, έτσι ώστε να είναι πλήρως ομόχειρη (fully pipelined). Για τα τις μονάδες για τα 8, 16 και 32 σημεία καταλήγουμε στα παρακάτω στάδια. Οι μονάδες για 2 και 4 σημεία δεν ακολουθούν αυτή τη μορφή, αφού η πρώτη έχει μόνο τις προσθέσεις με δύο όρους κι η δεύτερη έχει ένα στάδιο με τον fft2 και μετά τις τελικές προσθέσεις.

Το πρώτο στάδιο είναι ο υπολογισμός των γρήγορων μετασχηματισμών Fourier των ζυγών και μονών όρων (N/2 σημεία ο καθένας). Το στάδιο αυτό κρατάει όσους κύκλους μηχανής έχουμε υπολογίσει για τις αντίστοιχες μονάδες. Το δεύτερο στάδιο είναι οι υπολογισμοί των γινομένων που φαίνονται στους μαθηματικούς τύπους και κρατάει 5 κύκλους μηχανής. Δηλαδή, ο πολλαπλασιασμός των εξόδων των N/2 FFT μονάδων με το αντίστοιχο ημίτονο και συνημίτονο. Το τρίτο στάδιο είναι η μια πρόσθεση στους μαθηματικούς τύπους με το άθροισμα τριών όρων και το τέταρτο και τελευταίο στάδιο είναι το τελικό άθροισμα κι η έξοδος αποτελέσματος. Τα δύο τελευταία στάδιο κρατάνε από 4 κύκλους μηχανής.




**Σχήμα 4.5 Στάδια pipeline για τις μονάδες FFT για 8, 16 και 32 σημεία**

## 4.3 Implementation – Αποτελέσματα

Όλες τις παραπάνω μονάδες τις κάναμε σύνθεση (Synthesis) και υλοποίηση (Implementation) σε FPGA συσκευές. Για τις μονάδες fft2, fft4, fft8 και fft16 χρησιμοποιήσαμε την συσκευή της οικογένειας VIRTEX 6, XC6VCX240T και την 12.4 έκδοση του εργαλείου XILINX ISE και για την μονάδα fft32 την συσκευή της οικογένειας VIRTEX 7, XC7V2000T και την έκδοση 13.2 του εργαλείου XILINX ISE.

Για να γίνει η σύνθεση κι η υλοποίηση των μονάδων αφήσαμε στο εργαλείο XILINX ISE της επιλογές στις ιδιότητες των δύο αυτών ενεργειών, όπως είναι εκτός από τις εξής. Για της μονάδες fft4, fft8, fft16 και fft32 στις επιλογές της σύνθεσης δεν αφήσαμε να γίνει πρόσθεση των I/O Buffers, διότι δεν υπάρχουν όσες χρειαζόμασταν στις συσκευές FPGA. Οι είσοδοι για αυτές της μονάδες είναι πολλές, αφού πρέπει να είναι των 32 και 34 bit λόγω αριθμών κινητής υποδιαστολής. Επίσης, στις επιλογές για το mapping δεν αφήσαμε να αφαιρεθούν από το εργαλείο όσα σήματα δεν συνδέονται κάπου. Αυτό το απενεργοποιήσαμε, γιατί λόγω της επιλογής στη σύνθεση για τις εισόδους κι εξόδους θα αφαιρούσε όλη τη λογική που σχεδιάσαμε, αφού όλα συνδέονται με αυτές.

Αρχικά θα παραθέσουμε τα αποτελέσματα για την συχνότητα του ρολογιού που βγήκαν μετά τη σύνθεση των μονάδων.

| ΜΟΝΑΔΑ | ΕΛΑΧΙΣΤΗ ΠΕΡΙΟΔΟΣ (ns) | ΜΕΓΙΣΤΗ ΣΥΧΝΟΤΗΤΑ (MHz) |
|---|---|---|
| fft2 | 4.295ns | 232.818 MHz |
| fft4 | 7.750 ns | 129.038 MHz |
| fft8 | 7.755 ns | 128.948 MHz |
| fft16 | 7.739 ns | 129.210 MHz |
| fft32 | 7.538 ns | 132.664 MHz |

Στην συνέχεια υπάρχουν πίνακες με την κατανομή των πόρων της FPGA για την κάθε μονάδα που σχεδιάσαμε. Τα αποτελέσματα αυτά έχουν βγει μετά και τη διαδικασία του place and route.

Πίνακας κατανομής πόρων της FPGA για την μονάδα fft2

| FFT2 Device Utilization Summary | | | |
|---|---|---|---|
| | **Used** | **Available** | **Utilization** |
| **Slice Logic Utilization** | | | |
| *Number of Slice Registers* | 816 | 948,480 | 1% |
| Number used as Flip Flops | 816 | | |
| *Number of Slice LUTs* | 1,328 | 474,240 | 1% |
| Number used as logic | 1,304 | 474,240 | 1% |
| Number using O6 output only | 820 | | |
| Number using O5 output only | 132 | | |
| Number using O5 and O6 | 352 | | |
| Number used exclusively as route-thrus | 24 | | |
| Number with same-slice register load | 16 | | |
| Number with same-slice carry load | 8 | | |
| **Slice Logic Distribution** | | | |
| *Number of occupied Slices* | 398 | 118,560 | 1% |
| *Number of LUT Flip Flop pairs used* | 1,352 | | |
| Number with an unused Flip Flop | 704 | 1,352 | 52% |
| Number with an unused LUT | 24 | 1,352 | 1% |
| Number of fully used LUT-FF pairs | 624 | 1,352 | 46% |
| **IO Utilization** | | | |
| Number of bonded IOBs | 265 | 1200 | 22% |
| **Specific Feature Utilization** | | | |
| *Number of BUFG/BUFGCTRLs* | 1 | 32 | 3% |
| Number used as BUFGs | 1 | | |

Πίνακας κατανομής πόρων της FPGA για την μονάδα fft4

| FFT4 Device Utilization Summary | | | |
|---|---|---|---|
| | **Used** | **Available** | **Utilization** |
| **Slice Logic Utilization** | | | |
| *Number of Slice Registers* | 3,323 | 948,480 | 1% |
| Number used as Flip Flops | 3,323 | | |
| *Number of Slice LUTs* | 5,736 | 474,240 | 1% |
| Number used as logic | 5,689 | 474,240 | 1% |
| Number using O6 output only | 3,820 | | |
| Number using O5 output only | 530 | | |
| Number using O5 and O6 | 1,339 | | |
| Number used as Memory | 4 | 132,480 | 1% |
| Number used as Shift Register | 4 | | |
| Number using O6 output only | 4 | | |
| Number used exclusively as route-thrus | 43 | | |
| Number with same-slice register load | 11 | | |
| Number with same-slice carry load | 32 | | |
| **Slice Logic Distribution** | | | |
| *Number of occupied Slices* | 1,744 | 118,560 | 1% |
| *Number of LUT Flip Flop pairs used* | 5,905 | | |
| Number with an unused Flip Flop | 3,338 | 5,905 | 56% |
| Number with an unused LUT | 169 | 5,905 | 2% |
| Number of fully used LUT-FF pairs | 2,398 | 5,905 | 40% |

Πίνακας κατανομής πόρων της FPGA για την μονάδα fft8

| FFT8 Device Utilization Summary | | | |
|---|---|---|---|
| | **Used** | **Available** | **Utilization** |
| **Slice Logic Utilization** | | | |
| *Number of Slice Registers* | 13,290 | 948,480 | 4% |
| Number used as Flip Flops | 13,290 | | |
| *Number of Slice LUTs* | 21,678 | 474,240 | 4% |
| Number used as logic | 20,788 | 474,240 | 4% |
| Number using O6 output only | 14,050 | | |
| Number using O5 output only | 1,972 | | |
| Number using O5 and O6 | 4,766 | | |
| Number used as Memory | 578 | 132,480 | 1% |
| Number used as Shift Register | 578 | | |
| Number using O6 output only | 578 | | |
| Number used exclusively as route-thrus | 312 | | |
| Number with same-slice register load | 196 | | |
| Number with same-slice carry load | 116 | | |
| **Slice Logic Distribution** | | | |
| *Number of occupied Slices* | 6,788 | 118,560 | 5% |
| *Number of LUT Flip Flop pairs used* | 22,605 | | |
| Number with an unused Flip Flop | 11,793 | 22,605 | 52% |
| Number with an unused LUT | 927 | 22,605 | 4% |
| Number of fully used LUT-FF pairs | 9,885 | 22,605 | 43% |
| **Specific Feature Utilization** | | | |
| *Number of DSP48E1s* | 8 | 864 | 1% |

Πίνακας κατανομής πόρων της FPGA για την μονάδα fft16

| FFT16 Device Utilization Summary | | | |
|---|---|---|---|
| | **Used** | **Available** | **Utilization** |
| **Slice Logic Utilization** | | | |
| *Number of Slice Registers* | 44,271 | 948,480 | 4% |
| Number used as Flip Flops | 44,271 | | |
| *Number of Slice LUTs* | 68,030 | 474,240 | 14% |
| Number used as logic | 64,132 | 474,240 | 13% |
| Number using O6 output only | 44,211 | | |
| Number using O5 output only | 6,414 | | |
| Number using O5 and O6 | 13,507 | | |
| Number used as Memory | 2,310 | 132,480 | 1% |
| Number used as Shift Register | 2,310 | | |
| Number using O6 output only | 2,309 | | |
| Number using O5 and O6 | 1 | | |
| Number used exclusively as route-thrus | 1,588 | | |
| Number with same-slice register load | 1,224 | | |
| Number with same-slice carry load | 364 | | |
| **Slice Logic Distribution** | | | |
| *Number of occupied Slices* | 20,458 | 118,560 | 17% |
| *Number of LUT Flip Flop pairs used* | 71,423 | | |
| Number with an unused Flip Flop | 35,810 | 71,423 | 50% |
| Number with an unused LUT | 3,393 | 71,423 | 4% |
| Number of fully used LUT-FF pairs | 32,220 | 71,423 | 45% |
| **Specific Feature Utilization** | | | |
| *Number of DSP48E1s* | 56 | 864 | 6% |

Πίνακας κατανομής πόρων της FPGA για την μονάδα fft32

| FFT32 Device Utilization Summary | | | |
|---|---|---|---|
| | **Used** | **Available** | **Utilization** |
| **Slice Logic Utilization** | | | |
| *Number of Slice Registers* | 128,394 | 948,480 | 13% |
| Number used as Flip Flops | 128,394 | | |
| *Number of Slice LUTs* | 190,335 | 474,240 | 40% |
| Number used as logic | 177,950 | 474,240 | 37% |
| Number using O6 output only | 122,833 | | |
| Number using O5 output only | 18,415 | | |
| Number using O5 and O6 | 36,702 | | |
| Number used as Memory | 6,952 | 132,480 | 5% |
| Number used as Shift Register | 6,952 | | |
| Number using O6 output only | 6,952 | | |
| Number used exclusively as route-thrus | 5,433 | | |
| Number with same-slice register load | 4,413 | | |
| Number with same-slice carry load | 1,020 | | |
| **Slice Logic Distribution** | | | |
| *Number of occupied Slices* | 56,681 | 118,560 | 47% |
| *Number of LUT Flip Flop pairs used* | 198,999 | | |
| Number with an unused Flip Flop | 96,532 | 198,999 | 48% |
| Number with an unused LUT | 8,664 | 198,999 | 4% |
| Number of fully used LUT-FF pairs | 93,803 | 198,999 | 47% |
| **Specific Feature Utilization** | | | |
| *Number of DSP48E1s* | 216 | 864 | 25% |

# Κεφάλαιο 5

# Ιστόγραμμα (Histogram)

## 5.1 Εισαγωγή

Ο δεύτερος αλγόριθμος των μετροπρογραμμάτων PARBOIL, με τον οποίο ασχοληθήκαμε είναι το ιστόγραμμα. Με το ιστόγραμμα απεικονίζουμε σε ένα διάγραμμα τη συχνότητα τιμών ή περιοχών τιμών ενός μεγέθους. Το ιστόγραμμα έχει πολλές εφαρμογές, όπως η απεικόνιση αποτελεσμάτων ερευνών και η απεικόνιση τιμών εικονοστοιχείων στην ψηφιακή φωτογραφία.

Υπάρχουν διάφορα είδη ιστογράμματος. Εκτός από αυτό που δείχνει απλώς την συχνότητα εμφάνισης τιμών, υπάρχουν ιστογράμματα που δείχνουν την κατανομή εκτίμηση πυκνότητας πιθανότητας καθώς και συσσωρευτικά ιστογράμματα.

Ο αλγόριθμος ιστόγραμμα που εξετάσαμε παίρνει ως είσοδο μια εικόνα και ως έξοδο βγάζει το δισδιάστατο ιστόγραμμά της για την συχνότητα των τιμών. Μια εικόνα αποτελείται από εικονοστοιχεία και κάθε εικονοστοιχείο έχει μια τιμή. Το ιστόγραμμα δείχνει πόσα εικονοστοιχεία υπάρχουν από κάθε τιμή. Το συγκεκριμένο μετροπρόγραμμα θεωρεί ότι οι τιμές των εικονοστοιχείων είναι από 0 μέχρι 255.

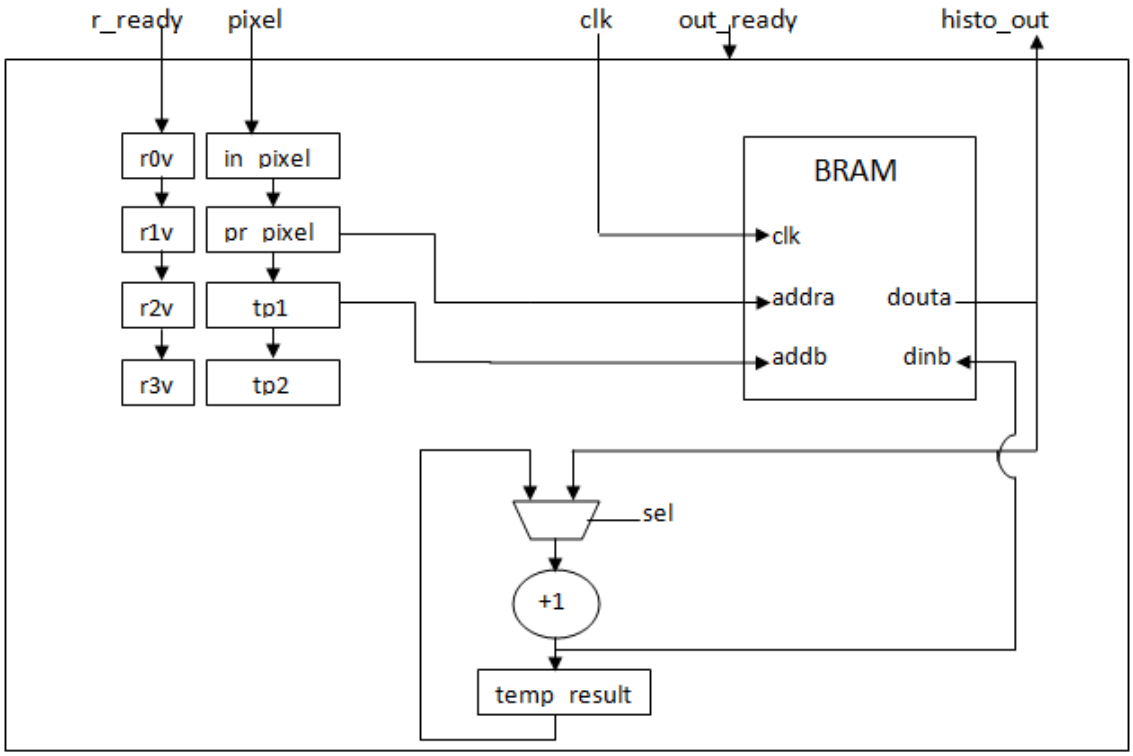
## 5.2 Σχεδίαση

Τον αλγόριθμο αυτόν τον υλοποιήσαμε σε επίπεδο υλικού στη γλώσσα verilog. Έχει τέσσερις εισόδους και μία έξοδο. Η μία είσοδος είναι το ρολόι (clk) και είναι 1 bit. Δύο ακόμα εισόδους του ενός bit έχουμε για να υποδεικνύουν η μεν r_ready, αν τα δεδομένα που έρχονται είναι έγκυρα για να ληφθούν υπόψη από την μονάδα κι η out_ready ότι θέλουμε η μονάδα να βγάλει από την έξοδό της τα αποτελέσματα. Η είσοδος pixel είναι το εικονοστοιχείο που θα επεξεργαστούμε και είναι 8 bit, αφού η μεγαλύτερη τιμή της είναι $256 = 2^8$. Τέλος, η μονάδα αυτή έχει και μια έξοδο histo_out των 32 bit. Η έξοδος κάθε φορά βγάζει τη συχνότητα της τιμής του εικονοστοιχείου που ζητήθηκε. Με έξοδο 32 bit η μονάδα μπορεί να χειριστεί εικόνες μέχρι $2^{32}$ εικονοστοιχεία.

Στο σχήμα 5.1 φαίνεται η σχεδίαση της μονάδας.

Η μονάδα αποτελείται από μια μνήμη τύπου block RAM (BRAM). Τη μνήμη την χρησιμοποιούμε για να αποθηκεύουμε εκεί τη συχνότητα εμφάνισης της κάθε τιμής των εικονοστοιχείων. Η μνήμη έχει 2 θύρες A και B. Χρησιμοποιήσαμε 2 θύρες για να μπορούμε να διαβάζουμε και να γράφουμε στον ίδιο κύκλο μηχανής. Κάθε μία έχει το δικό της ρολόι, τη δική της είσοδο κι έξοδο δεδομένων, καθώς από ένα σήμα write_enable, που υποδηλώνει αν θα γράψουμε στη μνήμη ή θα διαβάσουμε από αυτή. Η BRAM χρειάζεται έναν κύκλο μηχανής για να διαβάσουμε από αυτή κι άλλον έναν κύκλο για να γράψουμε σε αυτή. Αποφασίσαμε να διαβάζουμε από την θύρα A, δίνοντας την διεύθυνση στην είσοδο addra και παίρνοντας τα δεδομένα από την έξοδο douta, και να γράφουμε στη θύρα B, δίνοντας την διεύθυνση στην είσοδο addrb και εισάγοντας τα δεδομένα στην είσοδο dinb. Έτσι, το σήμα

write_enable για την θύρα A είναι πάντα 0 και για την θύρα B είναι πάντα 1. Το βάθος της μνήμης, δηλαδή από πόσες θέσεις αποτελείται, είναι 256, όσες κι οι δυνατές τιμές των εικονοστοιχείων. Έτσι, η τιμή του εικονοστοιχείου είναι η διεύθυνση της αντίστοιχης θέσης μνήμης. Κάθε θέση της μνήμης είναι 32 bit όσο κι η μέγιστη συχνότητα μιας τιμής ενός εικονοστοιχείου. Έτσι, το μέγεθος της μνήμης είναι 1KB. Όλες οι θέσεις μνήμης είναι αρχικοποιημένες στο 0. Η μονάδα παίρνει ένα ένα τα εικονοστοιχεία, διαβάζει από την αντίστοιχη θέση μνήμης την παλιά τιμή, προσθέτει σε αυτή το 1 κι αποθηκεύει στην ίδια θέση μνήμης την νέα τιμή.




**Σχήμα 5.1 Μονάδα ιστόγραμμα**

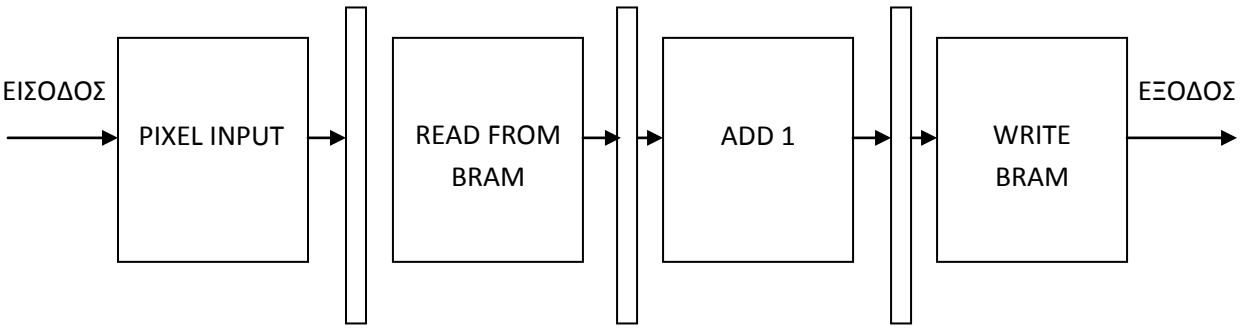
Στην υλοποίηση αυτή υπάρχουν ακόμα 9 καταχωρητές. Οι πέντε είναι των 8 bit, γιατί χρησιμοποιούνται για να κρατούν την τιμή του εικονοστοιχείου για όσους κύκλους μηχανής χρειάζεται, κι οι υπόλοιποι 4 είναι του ενός bit και χρησιμοποιούνται για έλεγχο. Όπως φαίνεται και στο σχήμα, σε κάθε κύκλο μηχανής κρατάμε την είσοδο στον καταχωρητή in_pixel και ταυτόχρονα οι προηγούμενες τιμές της εισόδου πηγαίνουν στους υπόλοιπους καταχωρητές. Έτσι, κρατάμε τις τιμές των εικονοστοιχείων για όσους κύκλους μηχανής τις χρειαζόμαστε για να κάνουμε τις απαραίτητες ενέργειες. Στην μνήμη δίνουμε ως διεύθυνση ανάγνωσης την τιμή του καταχωρητή pr_pixel και το δεδομένο που διαβάζουμε βγαίνει μετά από έναν κύκλο μηχανής στην έξοδο douta της μνήμης. Στον κύκλο που βγαίνει το αποτέλεσμα από την μνήμη, έχουμε περάσει την τιμή του εικονοστοιχείου που επεξεργαζόμαστε από τον καταχωρητή pr_ pixel στον καταχωρητή tp1 που είναι η διεύθυνση εγγραφής στην μνήμη. Στον ίδιο κύκλο έχουμε κάνει την πρόσθεση του δεδομένου douta με το 1 και το αποτέλεσμα πηγαίνει στην είσοδο dinb της μνήμης. Το αποτέλεσμα θα αποθηκευτεί στην μνήμη στον

επόμενο κύκλο, στον οποίο το εικονοστοιχείο που επεξεργαζόμαστε έχει περάσει στον καταχωρητή tp2. Θα πρέπει να περιμένουμε άλλον έναν κύκλο για να διαβάσουμε από την ίδια διεύθυνση, διαφορετικά θα πάρουμε παλιά τιμή κι όχι τη σωστή αφού δεν έχει προλάβει να αποθηκευτεί. Το πρόβλημα αυτό εμφανίζεται όταν έρχονται συνεχόμενα εικονοστοιχεία με την ίδια τιμή.

Για να αντιμετωπίσουμε το παραπάνω πρόβλημα έχουμε τον καταχωρητή tp2 για να συγκρίνουμε αν το εικονοστοιχείο που μόλις έκανε εγγραφή στη μνήμη (tp2) είναι ίδιο με αυτό θα αποθηκεύσει στη μνήμη στον επόμενο κύκλο (tp1), ενώ βάλαμε άλλον έναν βοηθητικό καταχωρητή temp_result των 8 bit για να κρατάει το αποτέλεσμα της πρόσθεσης. Εδώ, θα εξηγήσουμε και τους άλλους τέσσερις καταχωρητές ελέγχου r0v, r1v, r2v, r3v, που είπαμε παραπάνω του ενός bit. Όταν αυτοί οι καταχωρητές ελέγχου είναι 1, τότε σημαίνει ότι οι αντίστοιχοι καταχωρητές τους, που κρατούν τις τιμές των εικονοστοιχείων, έχουν έγκυρο δεδομένο. Ενώ όταν είναι 0, τότε οι αντίστοιχοι καταχωρητές τους δεν έχουν έγκυρο δεδομένο. Αυτό βοηθάει στην σύγκριση που κάνουμε αν ο tp1 είναι ίσος με τον tp2 σε περίπτωση που το περιεχόμενο είτε του ενός είτε και των δύο είναι ίσο με XXXXXXX, γιατί στην verilog τέτοια σύγκριση δεν βγάζει σωστό αποτέλεσμα. Αν το περιεχόμενο των καταχωρητών είναι XXXXXXXX, τότε οι καταχωρητές ελέγχου είναι 0. Έτσι, βάλαμε ακόμα μια σύγκριση με τους καταχωρητές r2v και r3v, η οποία πια δίνει το σωστό αποτέλεσμα. Αν οι καταχωρητές r2v και r3v είναι κι οι δύο 1 κι αν οι καταχωρητές tp1 και tp2 είναι ίσοι, τότε σημαίνει ότι έχουμε δύο συνεχόμενα εικονοστοιχεία με την ίδια τιμή, αλλιώς τα εικονοστοιχεία έχουν διαφορετική τιμή. Αυτό είναι και το σήμα sel που φαίνεται στον πολυπλέκτη του σχήματος. Στην δεύτερη περίπτωση η πρόσθεση με το 1 γίνεται με το δεδομένο από την έξοδο douta της μνήμης, αφού είναι το σωστό. Στην πρώτη περίπτωση η πρόσθεση με το 1 γίνεται με το δεδομένο του καταχωρητή temp_result, αφού αν πάρουμε το δεδομένο douta θα είναι λάθος, γιατί δεν θα έχει προλάβει να αποθηκευτεί στη μνήμη το αποτέλεσμα της πρόσθεσης που έγινε στον αμέσως προηγούμενο κύκλο για την ίδια τιμή του εικονοστοιχείου.

Με την παραπάνω σχεδίαση έχουμε καταφέρει η μονάδα να επεξεργάζεται ένα εικονοστοιχείο ανά κύκλο μηχανής. Όταν ένα εικονοστοιχείο έρθει ως είσοδος στη μονάδα, θα χρειαστούν τέσσερις κύκλοι μηχανής μέχρι το αποτέλεσμα για αυτό να αποθηκευτεί στην μνήμη και να είναι έτοιμο προς ανάγνωση. Όσον αφορά την έξοδο, όταν το σήμα out_ready γίνει 1 κι αρχίζουν να έρχονται τα εικονοστοιχεία, τότε η έξοδος histo_out της μονάδας βγάζει το αποτέλεσμα που υπάρχει στη μνήμη για τα εικονοστοιχεία που ζητήθηκαν. Από τη στιγμή που θα έρθει το πρώτο εικονοστοιχείο θα χρειαστούν τρεις κύκλοι για να πάρουμε αποτέλεσμα στην έξοδο.

Έτσι, έχουμε πετύχει τέσσερα στάδια ομοχειρίας και φαίνονται στο σχήμα 5.2. Στο πρώτο το εικονοστοιχείο εισέρχεται στη μονάδα. Στο δεύτερο γίνεται η ανάγνωση από τη μνήμη και στο τρίτο γίνεται η πρόσθεση με το 1. Στο τελευταίο στάδιο γίνεται εγγραφή στη μνήμη.




**Σχήμα 5.2 Στάδια pipeline της μονάδας ιστόγραμμα**

Η συγκεκριμένη υλοποίηση ελαχιστοποιεί τους κύκλους μηχανής που χρειάζεται η επεξεργασία για κάθε εικονοστοιχείο ξεχωριστά. Όμως, έτσι μειώνεται και η μέγιστη συχνότητα στην οποία μπορεί να λειτουργήσει η μονάδα. Άλλη υλοποίηση μπορεί να αυξήσει την συχνότητα και τους κύκλους μηχανής, κάνοντας πιο περίπλοκο όμως την σχεδίαση στη γλώσσα verilog, αφού θα χρειάζονταν περισσότεροι καταχωρητές για να κρατάμε για περισσότερους κύκλους μηχανής ένα εικονοστοιχείο και περισσότερες συγκρίσεις μεταξύ αυτών για να αποφευχθεί το πρόβλημα ανάγνωσης λάθος τιμής από την μνήμη που αναφέραμε παραπάνω .

## 5.3 Implementation - Αποτελέσματα

Με τη βοήθεια του προγράμματος Xilinx ISE κάναμε synthesis και implementation τη μονάδα ιστόγραμμα για να δούμε πόσο χώρο πιάνει στην πλακέτα FPGA. Το device που χρησιμοποιήσαμε είναι το VIRTEX 6 XC6VLX760 και τα αποτελέσματα που πήραμε για την σχεδίαση μας είναι στον παρακάτω πίνακα.

Τα αποτελέσματα για το ρολόι είναι τα εξής:
Ελάχιστη περίοδος: 3.920ns
Μέγιστη Συχνότητα: 255.070MHz

Στην επόμενη σελίδα βρίσκεται ο συγκεντρωτικός πίνακας με τα αποτελέσματα μετά και την διαδικασία του Place and Route.

Πίνακας κατανομής πόρων της FPGA για την μονάδα Ιστόγραμμα

| HISTOGRAM DEVICE UTILIZATION SUMMARY | | | |
|---|---|---|---|
| | **Used** | **Available** | **Utilization** |
| **Slice Logic Utilization** | | | |
| *Number of Slice Registers* | 89 | 948,480 | 1% |
| Number of Slice Registers | 89 | | |
| *Number of Slice LUTs* | 44 | 474,240 | 1% |
| Number used as Logic | 36 | 474,240 | 1% |
| Number using O6 output only | 5 | | |
| Number using O5 and O6 | 31 | | |
| Number used as Memory | 5 | 132,480 | 1% |
| Number used as Shift Register | 5 | | |
| Number using O6 output only | 1 | | |
| Number using O5 and O6 | 4 | | |
| Number used exclusively as route-thrus | 3 | | |
| Number with same-slice register load | 3 | | |
| **Slice Logic Distribution** | | | |
| *Number of occupied Slices* | 22 | 118,560 | 1% |
| *Number of LUT Flip Flop pairs used* | 80 | | |
| Number with an unused LUT | 36 | 80 | 45% |
| Number of fully used LUT-FF pairs | 44 | 80 | 55% |
| **IO Utilization** | | | |
| Number of bonded IOBs | 43 | 1200 | 3% |
| **Specific Feature Utilization** | | | |
| *Number of RAMB36E1/FIFO36E1s* | 1 | 720 | 1% |
| Number using RAMB36E1 only | 1 | | |
| *Number of BUFG/BUFGCTRLs* | 1 | 32 | 3% |
| Number used as BUFGs | 1 | | |

# Κεφάλαιο 6

# Μαγνητική Τομογραφία (Magnetic Resonance Imaging)

## 6.1 Εισαγωγή

Ο τελευταίος αλγόριθμος των μετροπρογραμμάτων PARBOIL, που ασχοληθήκαμε είναι ο αλγόριθμος, ο οποίος χρησιμοποιείται στην μαγνητική τομογραφία. Πιο συγκεκριμένα, ασχοληθήκαμε με ένα κομμάτι του, το οποίο υπολογίζει ένα επίπεδο του τρισδιάστατου πίνακα Q.

Η ανακάλυψη της μαγνητικής τομογραφίας ήταν ένα μεγάλο βήμα για την ιατρική. Εκμεταλλεύεται ιδιότητες των διάφορων πυρήνων των ατόμων του ανθρώπινου σώματος, όταν βρίσκονται σε μαγνητικό πεδίο, για να απεικονίσει τα εσωτερικά μέρη του σώματος. Η μαγνητική τομογραφία σε αντίθεση με τις κοινές ακτινογραφίες και την αξονική τομογραφία δεν χρησιμοποιεί ιονίζουσα ακτινοβολία που είναι επιβλαβής για τον άνθρωπο. Έχει πολύ μεγάλη ευκρίνεια και χρησιμοποιείται για να ξεχωρίζουμε των παθολογικό ιστό από τον υγιή. Για παράδειγμα, με την μαγνητική τομογραφία οι γιατροί μπορούν να δουν έναν όγκο, μια κύστη, μια ζημιά σε έναν μυ. Ακόμα κι αν αυτά έχουν πολύ μικρό μέγεθος, με τη μαγνητική τομογραφία φαίνονται καθαρά μιας και αυτή προσφέρει πολύ καλή χωρική ανάλυση.

Η μαγνητική τομογραφία αποτελείται από δύο φάσεις. Η πρώτη φάση είναι η σάρωση κι η δεύτερη είναι η ανακατασκευή της τελική εικόνας. Κατά την σάρωση γίνεται η λήψη δεδομένων στο πεδίο του μετασχηματισμού Fourier κατά μήκος μιας προκαθορισμένης τροχιάς και κατά τη δεύτερη φάση αυτά τα δεδομένα μετασχηματίζονται και παίρνουμε την τελική εικόνα.

Γενικά, το πρόβλημα της μαγνητική τομογραφίας περιγράφεται από τον τύπο $m(r) = \sum_j W(k_j)s(k_j)e^{i2\pi k(j)r}$, όπου m(r) είναι η τελική εικόνα, s(k) τα δεδομένα που μετρήθηκαν στο πεδίο του μετασχηματισμού Fourier και W(k) μια βεβαρυμμένη συνάρτηση που χρησιμοποιείται, όταν έχουμε μη ομοιόμορφη δειγματοληψία. Το W(k) βοηθά στη μείωση της επίδρασης δεδομένων από περιοχές του πεδίο μετασχηματισμού Fourier, από όπου η πυκνότητα των δειγμάτων είναι μεγάλη.

Υπάρχουν δύο τρόποι λήψης δεδομένων και ανακατασκευής της τελικής εικόνας. Ο πρώτος, ο οποίος σήμερα είναι κι ο πιο διαδεδομένος, είναι να γίνεται δειγματοληψία από ένα ομοιόμορφο καρτεσιανό πλέγμα σημείων. Σε αυτή την περίπτωση η συνάρτηση W(k) είναι σταθερά και βγαίνει έξω από το άθροισμα στον τύπο του m(r). Τότε, ο υπολογισμός που γίνεται για να δημιουργήσουμε την τελική εικόνα είναι ο αντίστροφος μετασχηματισμός Fourier που είναι και το πλεονέκτημα αυτής της μεθόδου. Το μειονέκτημά της είναι ότι για να γίνει η σάρωση στο ομοιόμορφο καρτεσιανό πεδίο Fourier χρειάζεται πολύ χρόνος ,που οδηγεί σε δυσφορία του ασθενή, όσο περιμένει στο μηχάνημα. Ο δεύτερος τρόπος είναι να γίνεται η δειγματοληψία ακολουθώντας μια μη ομοιόμορφη καρτεσιανή τροχιά, όπως είναι η σπείρα. Έτσι, μειώνεται ο χρόνος σάρωσης, αλλά η συνάρτηση W(k) δεν είναι πια σταθερά και παραμένει μέσα στο άθροισμα του τύπου για το m(r) κάνοντας τον υπολογισμό του πιο δύσκολο, αφού αυτός δεν είναι ο αντίστροφος μετασχηματισμός Fourier. Γι' αυτό το λόγο, τα δείγματα

προβάλλονται πρώτα σε ένα ομοιόμορφο καρτεσιανό πλέγμα μέσω συνέλιξης κι έπειτα εφαρμόζεται ο αντίστροφος μετασχηματισμός Fourier, διαδικασία όμως που είναι πιο αργή από τον πρώτο τρόπο.

Στα μετροπρογράμματα του PARBOIL υπάρχουν δυο αλγόριθμοι που ασχολούνται με τον δεύτερο τρόπο που εξηγήσαμε και συγκεκριμένα με έναν τρόπο επίλυσης του προβλήματος που πρότειναν οι Haldar και Liang. Οι Haldar και Liang πρότειναν μια γραμμική επαναληπτική μέθοδο που περιγράφεται από τον εξής τύπο: $(F^HF + \lambda W^HW)\ \rho = F^Hd$ . Το ρ είναι ένα διάνυσμα που περιέχει τιμές για τα εικονοστοιχεία της τελικής εικόνας, το F είναι ένας τρισδιάστατος πίνακας που μοντελοποιεί φυσικές ιδιότητες κατά τη διαδικασία της απεικόνισης, το W είναι κι αυτός ένας τρισδιάστατος πίνακας που περιέχει κάποιες απαραίτητες πληροφορίες για την μαγνητική τομογραφία ,όπως ανατομικούς περιορισμούς, και το d είναι ένα διάνυσμα για με τα δεδομένα από την σάρωση.

Εμείς, ασχοληθήκαμε με τον υπολογισμό του τρισδιάστατου πίνακα $Q = F^HF + \lambda W^HW$. Αυτός ο πίνακας αντιπροσωπεύει τις ρυθμίσεις που γίνονται στον σαρωτή για την προσαρμογή σε κάθε ασθενή. Ο αλγόριθμος του PARBOIL υπολογίζει ένα επίπεδο αυτού του πίνακα. Ο αλγόριθμος έχει δέκα εισόδους και δύο εξόδους. Ως είσοδο παίρνει καταρχάς δύο ακέραιους αριθμούς τον indexX και τον indexK, οι οποίοι είναι ο αριθμός των εικονοστοιχείων και ο αριθμός των δειγμάτων της τροχιάς που χρησιμοποιήθηκαν αντίστοιχα. Επίσης, είσοδοι σε αυτόν τον αλγόριθμο είναι τα διανύσματα kx, ky και kz που είναι τα δείγματα και τα διανύσματα x, y και z που είναι τα αρχικά εικονοστοιχεία. Τέλος, είσοδοι είναι και τα phiR και phiI, διανύσματα που δείχνουν την βάση των kx, ky και kz του μετασχηματισμού Fourier. Οι δύο έξοδοι είναι τα διανύσματα Qr και Qi που είναι τα πραγματικά και φανταστικά μέρη για κάθε στοιχείο του πίνακα Q. Τα kx, ky, kz, phiR και phiI έχουν μέγεθος indexK και τα x, y, z, Qr και Qi έχουν μέγεθος index.

Ο πίνακας Q υπολογίζεται όπως φαίνεται στο παρακάτω κομμάτι κώδικα.

```
(1)   for (int indexK = 0; indexK < numK; indexK++) {

(2)       phiMag[indexK] = phiR^2[indexK] + phiI^2[indexK];

(3)       for (int indexX = 0; indexX < numX; indexX++) {

(4)          expArg = 2*pi * (Kx[indexK] * x[indexX] +  Ky[indexK] * y[indexX] +  Kz[indexK] * z[indexX]);

(5)          cosArg = cosf(expArg);

(6)          sinArg = sinf(expArg);

(7)          Qr[indexX] += phiMag[indexK] * cosArg;

(8)          Qi[indexX] += phiMag[indexK] * sinArg;

(9)         }

(10)  }
```

Όπως βλέπουμε, για κάθε στοιχείο των Qr και Qi υπολογίζουμε το phiMag, που το πολλαπλασιάζουμε με το συνημίτονο και το ημίτονο μιας γωνίας expArg αντίστοιχα, και το αποτέλεσμα το προσθέτουμε στην προηγούμενη τιμή τους.

## 6.2 Σχεδίαση

Τον παραπάνω κώδικα για τον υπολογισμό ενός επιπέδου του πίνακα Q τον σχεδιάσαμε σε επίπεδο υλικού. Οι αριθμοί που θα διαχειριστούμε είναι κινητής υποδιαστολής, οπότε θα συμπεριλάβουμε στη σχεδίαση αντίστοιχες μονάδες. Από τον κώδικα φαίνεται ότι θα χρειαστούμε μονάδες πρόσθεσης και πολλαπλασιασμού. Διαθέσαμε για τον συγκεκριμένο αλγόριθμο μονάδες πολλαπλασιασμού που χρειάζονται πέντε κύκλους μηχανής και μονάδες πρόσθεσης που χρειάζονται τρεις κύκλους μηχανής.

Βλέποντας τον κώδικα, παρατηρούμε ότι λόγω των επαναλήψεων, οι αριθμοί κινητής υποδιαστολή kx, ky, kz, x, y και z επαναλαμβάνονται, όπως και τα Qr και Qi, για τα οποία θα χρειαστούμε τις παλιές τιμές. Αυτό μας οδήγησε να σκεφτούμε ότι θα μπορούσαμε μέσα στη μονάδα να έχουμε είτε μνήμες είτε καταχωρητές, στους οποίους θα έμεναν αποθηκευμένες οι τιμές που θα χρειαζόμασταν. Όμως, επειδή τα indexX και indexK είναι μεγάλη αριθμοί (σε ένα αρχείο για μικρή σε μέγεθος είσοδο στον αλγόριθμο των μετροπρογραμμάτων PARBOIL βρέθηκε ότι indexX=32768 και indexK=3072), θα χρειαζόμασταν μεγάλη μνήμη και πολλούς καταχωρητές αντίστοιχα. Επίσης, η σχεδίαση θα λειτουργούσε μόνο για συγκεκριμένο μέγεθος των indexX και indexK, αφού η μνήμη θα είχε σχεδιαστεί για το ακριβές μέγεθος των indexX και indexK, κι οι καταχωρητές θα ήταν επίσης άλλοι τόσοι , κάτι που θα ήταν κακή σχεδίαση του υλικού, εφόσον εμείς θέλουμε να λειτουργεί για οποιοδήποτε μέγεθος εισόδου μας δοθεί.

Έτσι, αποφασίσαμε να δημιουργήσουμε έναν hardware accelerator για τον αλγόριθμο αυτόν. Δηλαδή δεν θα υλοποιείται ολόκληρος ο αλγόριθμος παρά μόνο ένα κομμάτι του. Αυτό το κομμάτι είναι οι πράξεις που γίνονται μέσα στις επαναλήψεις του κώδικα. Ως είσοδο θα δέχεται ένα ένα τα στοιχεία των διανυσμάτων phiR, phiI, kx, ky, kz, x, y και z, καθώς επίσης και τις παλιές τιμές των στοιχείων των διανυσμάτων Qr και Qi και ως έξοδο θα έχει τις καινούριες τιμές των Qr και Qi. Όλες αυτές οι μεταβλητές θα έχουν εύρος 32 bit, αφού είναι αριθμοί κινητής υποδιαστολής. Με αυτόν τον τρόπο δεν χρειάζεται να κρατάμε καμιά τιμή και απλά θα υλοποιούνται οι αριθμητικές πράξεις στο υλικό. Τέλος, ως είσοδο θα έχει το ρολόι (clk) και ένα σήμα reset (rst), και τα δύο εύρους 1bit για επαναφορά της μονάδας σε αρχική κατάσταση, λειτουργία που υπάρχει στις μονάδες κινητής υποδιαστολής.

### 6.2.1 Υπολογισμός ημιτόνου - συνημίτονου

Όπως είδαμε, ο αλγόριθμος απαιτεί τον υπολογισμό του ημιτόνου και συνημίτονου μιας γωνίας. Οπότε, έπρεπε να δημιουργήσουμε και μια μονάδα υλικού που να υπολογίζει αυτά τα δύο. Στο υλικό, συνήθως , για να υπολογίσει κάποιος μια τριγωνομετρική συνάρτηση χρησιμοποιεί μια μέθοδο που λέγεται CORDIC, όταν δεν υπάρχει διαθέσιμη μονάδα πολλαπλασιασμού υλοποιημένη στο υλικό ή όταν ο αριθμός των πυλών που χρειάζονται για την υλοποίηση μιας συνάρτησης πρέπει να είναι ο μικρότερος δυνατός (FPGA). Αυτή η μέθοδος υπολογίζει το αποτέλεσμα χρησιμοποιώντας μόνο πράξεις πρόσθεσης, αφαίρεσης, ολίσθησης και look-up tables κι έτσι ικανοποιεί τις παραπάνω συνθήκες. Το CORDIC είναι μια επαναληπτική μέθοδος που κάνει αναστροφές στο διάνυσμα μέχρι το επιθυμητό αποτέλεσμα. Για να πετύχουμε ακρίβεια στη χειρότερη $10^{-5}$ χρησιμοποιήσαμε μια μονάδα CORDIC που κάνει 12 επαναλήψεις και μετρήσαμε τη συχνότητα ρολογιού της και πόσο χώρο πιάνει στην FPGA (σε συσκευή της οικογένειας VIRTEX 6 XC6VLX760) και καταλήξαμε στα εξής. Για να βγει αποτέλεσμα χρειάζεται 48 κύκλους μηχανής, μπορεί να δουλέψει σε μέγιστη συχνότητα 213.311MHz και τα slices που πιάνει στην FPGA είναι 3,860. Επίσης, παρατηρήσαμε ότι για να ισχύει η παραπάνω ακρίβεια

πρέπει η γωνία να είναι μεταξύ των αριθμών -π/2 και π/2. Αν βγαίνει από αυτά τα όρια τότε η ακρίβεια μειώνεται πάρα πολύ.

Εμείς προσπαθήσαμε, όμως, να σχεδιάσουμε μια μονάδα υπολογισμού του ημιτόνου και του συνημίτονου με τη βοήθεια της σειράς Taylor, η οποία προσεγγίζει μια συνάρτηση με ένα άπειρο άθροισμα όρων, που υπολογίζεται μέσω παραγώγων της συνάρτησης σε ένα σημείο. Η σχεδίαση έγινε για να πετύχουμε ακρίβεια στη χειρότερη $10^{-4}$, για να τη συγκρίνουμε με την μονάδα CORDIC και να επιλέξουμε την κατάλληλη για εμάς μονάδα. Ο τύπος της σειράς Taylor για μια συνάρτηση f(x) βάση ενός σημείου α είναι ο εξής:

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots.$$

Στον παραπάνω τύπο αντικαθιστούμε, όπου f τις συναρτήσεις sin(x) και cos(x) για το ημίτονο και το συνημίτονο αντίστοιχα.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots, \ -\infty < x < \infty$$

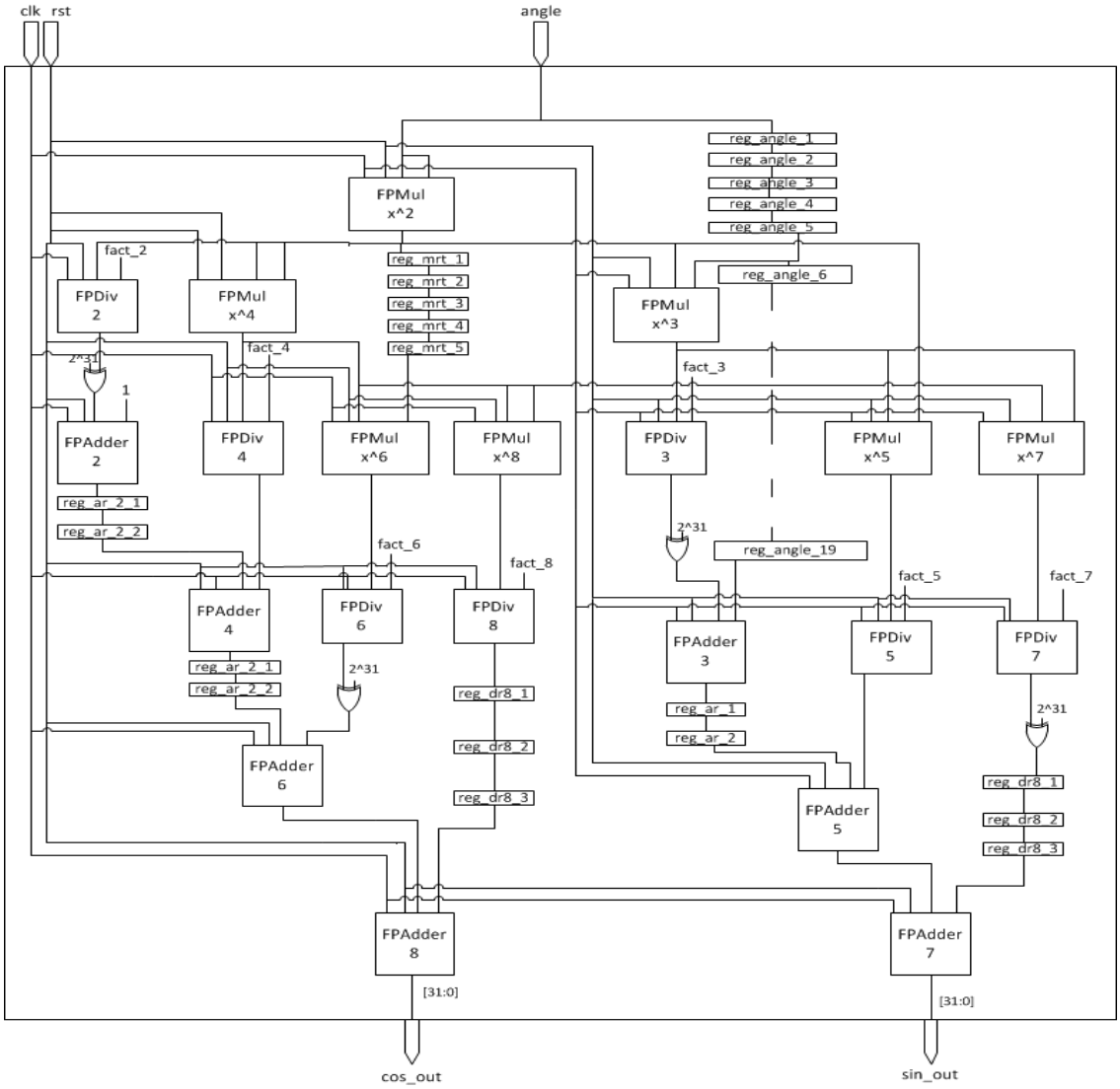
$$\cos(\mathrm{x}) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \ldots, -\infty < x < \infty$$

Ακολουθώντας τους παραπάνω τύπους σχεδιάσαμε την μονάδα για τον υπολογισμό του ημιτόνου και του συνημίτονου χρησιμοποιώντας τις μονάδες πολλαπλασιασμού, διαίρεσης και πρόσθεσης κινητής υποδιαστολής με 5, 9 και 3 κύκλους μηχανής αντίστοιχα. Κρατήσαμε 4 όρους για το ημίτονο και 5 όρους για το συνημίτονο.

Όπως βλέπουμε στο σχήμα 6.1, έχει δύο εισόδους του 1 bit για το ρολόι (clk) και το reset(rst) και μία είσοδο των 32 bit για την γωνία, για την οποία θέλουμε να υπολογίσουμε το ημίτονο και το συνημίτονο. Επίσης, έχει δύο εξόδους των 32 bit, την cos_out για το συνημίτονο και την sin_out για το ημίτονο. Η μονάδα έχει σχεδιαστεί, έτσι ώστε τα δύο αποτελέσματα να βγαίνουν ταυτόχρονα, ενώ μπορεί σε κάθε κύκλο μηχανής να βγάλει αποτέλεσμα, έπειτα από την αρχική καθυστέρηση των 30 κύκλων μηχανής. Η ακρίβεια είναι στη χειρότερη περίπτωση $10^{-5}$ πάλι για εύρος γωνιών από –π/2 έως π/2 και μπορεί να υπολογιστεί μέσω της ιδιότητας της σειράς Taylor, σύμφωνα με την οποία το σφάλμα είναι ο πρώτος όρος που υπολείπεται από τον μαθηματικό τύπο.

Το αριστερό μέρος του σχήματος υπολογίζει το συνημίτονο και το δεξί το ημίτονο. Δηλαδή, οι μονάδες κινητής υποδιαστολής, με τα χαρακτηριστικά FPMul για τον πολλαπλασιασμό, FPadder για την πρόσθεση και FPDiv για την διαίρεση, που έχουν ζυγό αριθμό (με εξαίρεση τον FPMul για το $x^2$ που χρειάζεται και για τις δύο συναρτήσεις)είναι για το συνημίτονο και αυτές που έχουν μονό αριθμό είναι για το ημίτονο. Με αυτόν τον τρόπο ικανοποιούνται οι δύο μαθηματικοί τύποι για τις δύο συναρτήσεις. Επίσης, τα ορθογώνια με τις ονομασίες, που έχουν το χαρακτηριστικό reg μπροστά, είναι καταχωρητές και κρατάν τα αντίστοιχα σήματα μέχρι να έρθει ο σωστός κύκλος μηχανής που θα χρησιμοποιηθούν από τις μονάδες. Όπου υπάρχει διακεκομμένο καλώδιο παραλείπονται κάποιοι καταχωρητές. Πιο συγκεκριμένα, υπολογίζουμε αρχικά το $x^2$. Έπειτα το $x^4$ υπολογίζεται πολλαπλασιάζοντας το $x^2$ με τον εαυτό του και το $x^3$ με την αρχική γωνία x. Ταυτόχρονα έχει αρχίσει κι ο υπολογισμός του κλάσματος $x^2$/fact(2), όπου fact(2) το παραγοντικό του 2. Ο FPAdder2 υπολογίζει το πρώτο άθροισμα 1- $x^2$/fact(2) του τύπου του συνημίτονου. Παρομοίως όλες οι FPDiv μονάδες υπολογίζουν τα κλάσματα των δυνάμεων του x με τα αντίστοιχα παραγοντικά και όλες οι μονάδες FPMul τις δυνάμεις του x πολλαπλασιάζοντας μεταξύ τους όποιες δυνάμεις έχουν υπολογιστεί όσο πιο νωρίς γίνεται. Για παράδειγμα το $x^6$ είναι το γινόμενο του $x^4$ με το $x^2$ και το $x^7$ είναι το γινόμενο του $x^4$ με το $x^3$. Τέλος,

όλες οι μονάδες FPAdder υπολογίζουν τα αθροίσματα των όρων με τον FPAdder8 να βγάζει το τελικό αποτέλεσμα του συνημίτονου και τον FPAdder7 το αντίστοιχο για το ημίτονο. Παρατηρούμε ότι οι έξοδοι έχουν εύρος 32 bit παρότι οι μονάδες κινητής υποδιαστολής έχουν έξοδο 34 bit προσθέτοντας όπως έχουμε εξηγήσει άλλα δύο bit στην αρχή των αριθμών για να βλέπουμε αν είναι έγκυρος ο αριθμος. Πριν την τελική έξοδο εμείς αφαιρούμε αυτά τα δύο bit και κρατάμε μόνο το μέρος που μας ενδιαφέρει. Η αφαιρέσεις έχουν γίνει με τη βοήθεια πυλών XOR.



Σχήμα 6.1 Μονάδα υπολογισμού ημιτόνου – συνημίτονου για γωνία από -π/2 έως π/2

Την μονάδα αυτή την κάναμε implementation στην ίδια συσκευή με την μονάδα CORDIC για να τις συγκρίνουμε. Τα αποτελέσματα που πήραμε για την μονάδα με το Taylor είναι: μέγιστη συχνότητα ρολογιού 161.162MHz και slices 2295.

Βλέπουμε ότι για την ίδια ακρίβεια και μετρημένες στην ίδια συσκευή, ο υπολογισμός του ημιτόνου και του συνημίτονου πιάνει πολύ περισσότερο χώρο στην FPGA με το CORDIC και το ίδιο χρειάζεται περισσότερους κύκλους μηχανής για να βγάλει αποτέλεσμα. Η μέγιστη συχνότητα ρολογιού

είναι καλύτερη για το CORDIC, αλλά και του Taylor κρίνεται ικανοποιητική. Γι' αυτούς τους λόγους επιλέξαμε να χρησιμοποιήσουμε για το MRI Q την μονάδα με τη σειρά Taylor.

| **ΣΥΓΚΡΙΤΙΚΟΣ ΠΙΝΑΚΑΣ ΜΕΘΟΔΩΝ CORDIC – ΣΕΙΡΑΣ TAYLOR** | | |
|---|---|---|
| *-π/2<γωνία<π/2* | CORDIC | ΣΕΙΡΑ TAYLOR |
| ΧΕΙΡΟΤΕΡΗ ΑΚΡΙΒΕΙΑ | $10^{-5}$ | $10^{-5}$ |
| ΚΥΚΛΟΙ ΜΗΧΑΝΗΣ | 48 | 30 |
| SLICES | 3,860 | 2295 |
| ΜΕΓΙΣΤΗ ΣΥΧΝΟΤΗΤΑ ΡΟΛΟΓΙΟΥ | 213.311 MHz | 161.162 MHz |

Δυστυχώς, η παραπάνω μονάδα δουλεύει καλά μόνο για γωνίες από –π/2 έως π/2 και στον αλγόριθμο η αντίστοιχη γωνία expArg, για την οποία θα υπολογίζουμε το ημίτονο και συνημίτονό της, βρήκαμε ότι θα βγαίνει έξω από αυτά τα όρια. Οπότε το επόμενο βήμα μας ήταν να εκμεταλλευτούμε τις ιδιότητες των γωνιών σύμφωνα με τις οποίες κάθε γωνία μπορεί να αναχθεί στο πρώτο τεταρτημόριο του τριγωνομετρικού κύκλου και να βρεθεί έπειτα το ημίτονο και το συνημίτονό της αναλόγως σε ποιο τεταρτημόριο ήταν αρχικά η γωνία. Έπειτα σχεδιάσαμε έναν μηχανισμό σε επίπεδο υλικού και τον προσαρτήσαμε στην παραπάνω μονάδα υπολογισμού των δύο τριγωνομετρικών συναρτήσεων.

Για να γίνει αναγωγή της γωνίας expArg στο πρώτο τεταρτημόριο θα πρέπει να τη διαιρέσουμε με το π/2 κι έπειτα να βρούμε το υπόλοιπο αυτής της διαίρεσης. Σε επίπεδο υλικού για αριθμούς κινητής υποδιαστολής μονάδα υπόλοιπο δεν υπάρχει. Έτσι, σκεφτήκαμε ότι θα πάρουμε το πηλίκο της διαίρεσης expArg/(π/2), θα κρατήσουμε το δεκαδικό μέρος του και πολλαπλασιάζοντάς το με το π/2 θα βρούμε το επιθυμητό υπόλοιπο. Αυτό στηρίζεται στο ότι αν έχουμε έναν αριθμό A και τον διαιρέσουμε με έναν αριθμό β, παίρνουμε ένα πηλίκο q,το οποίο έχει ένα ακέραιο i(q) κι ένα δεκαδικό μέρος f(q). Τότε ο τύπος της διαίρεσης γίνεται A=q*β = (i(q)+ f(q))*β = i(q)*β + f(q)*β. Το f(q) είναι το υπόλοιπο της διαίρεσης. Αρχικά με έναν FPDivider διαιρέσαμε την expArg με το π/2 και βρήκαμε ένα πηλίκο ας υποθέσουμε q.

Για να κρατήσουμε το δεκαδικό μέρος του q φτιάξαμε μια μονάδα keep_fract που παίρνει έναν αριθμό κινητής υποδιαστολής 32 bit και βγάζει το δεκαδικό του μέρος πάλι σε μορφή κινητής υποδιαστολής 32 bit. Αυτό γίνεται βλέποντας τον εκθέτη του αρχικού αριθμού και ολισθαίνοντας τον αριστερά τόσες θέσεις όσες είναι ο μη κανονικοποιημένος εκθέτης, δηλαδή εκθέτης - 127. Από αυτόν τον αριθμό κρατάμε μόνο το δεκαδικό μέρος και τον ολισθαίνουμε τόσες θέσεις αριστερά όσες είναι ο πρώτος άσσος, έτσι ώστε να τον φέρουμε πάλι σε κανονική μορφή και ο εκθέτης είναι μείον όσες θέσεις έγινε η ολίσθηση +127. Για καλύτερη κατανόηση παραθέτομε ένα παράδειγμα. Έστω ο κανονικοποιημένος αριθμός κινητής υποδιαστολής $1.01100010111*2^{130}$. Αφαιρούμε από τον εκθέτη 127 και μας μένει 3. Άρα ολισθαίνουμε τον αριθμό 3 θέσεις αριστερά: 1011.00010111. Κρατάμε το δεκαδικό μέρος 0. 00010111 και βλέπουμε ότι ο πρώτος άσσος είναι μετά από 4 θέσεις. Για να κανονικοποιήσουμε αυτόν τον αριθμό, ολισθαίνουμε αριστερά κατά 4 θέσεις, φτιάχνουμε τον εκθέτη ως -4+127 και παίρνουμε το τελικό αποτέλεσμα που είναι το δεκαδικό μέρος του αρχικού αριθμού: $1.0111*2^{123}$. Το πρόσημο που βγαίνει είναι πάντα θετικό, αφού δεν μας ενδιαφέρουν οι αρνητικοί αριθμοί. Αυτόν τον αριθμό που επιστρέφεται από αυτή την μονάδα τον πολλαπλασιάζουμε σε έναν FPMultiplier με το π/2 και παίρνουμε το υπόλοιπο που θέλαμε. Έτσι, αυτό το γινόμενο/υπόλοιπο το προωθούμε στην μονάδα υπολογισμού των δύο τριγωνομετρικών συναρτήσεων που σχεδιάσαμε πριν για να πάρουμε το ημίτονο και συνημίτονο του υπολοίπου της γωνίας που βρίσκεται πάντα στο πρώτο τεταρτημόριο.

Εξίσου σημαντικό με το υπόλοιπο είναι και το τεταρτημόριο, στο οποίο βρίσκεται η αρχική γωνία για τις τελικές τιμές των ημιτόνων και ημιτόνων. Οι ιδιότητες είναι εξής:

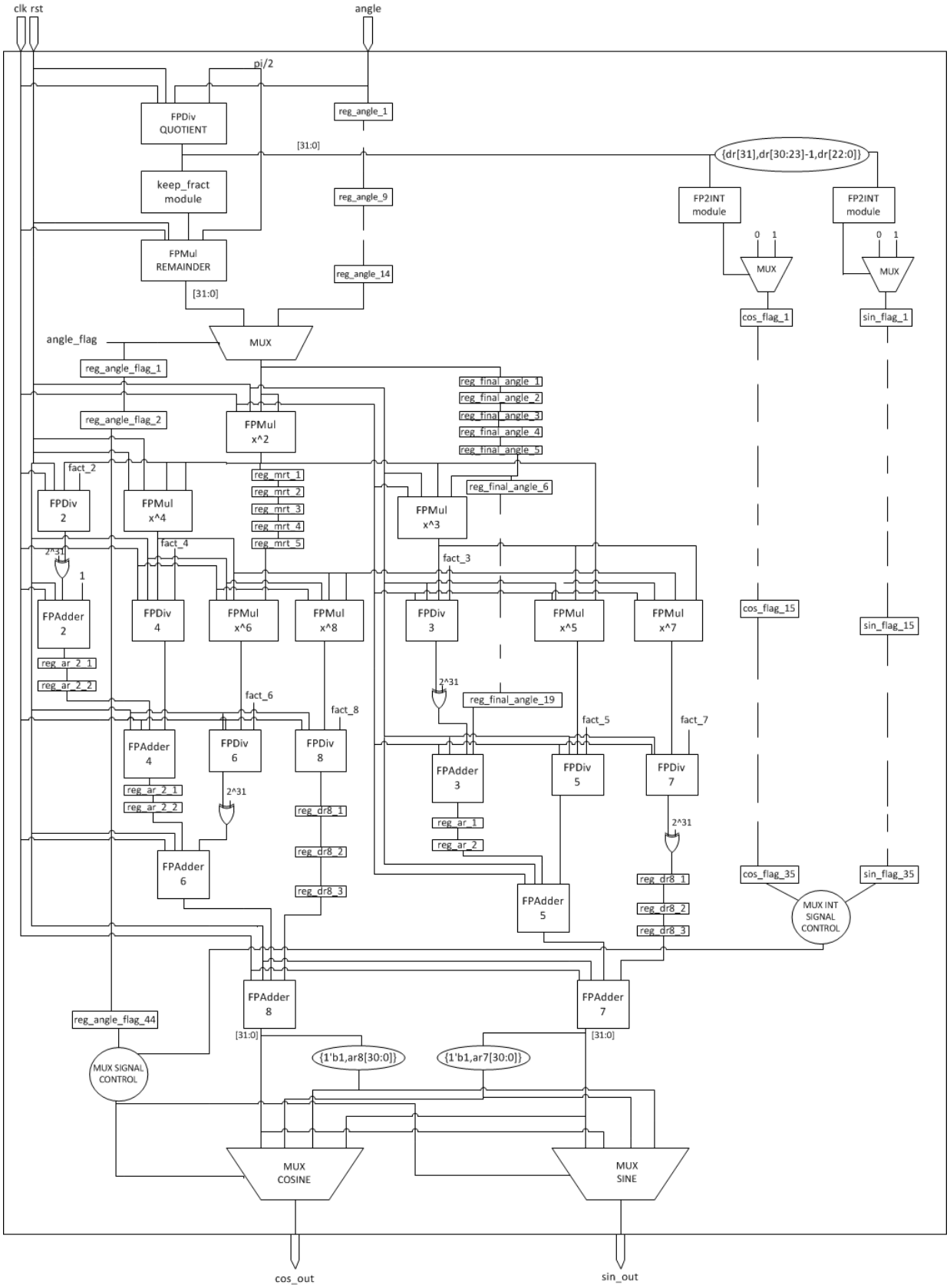
- αν η αρχική γωνία είναι στο πρώτο τεταρτημόριο, τότε το τελικό ημίτονο και συνημίτονο είναι το ίδιο με αυτά του υπολοίπου της γωνίας.
- αν η αρχική γωνία είναι στο δεύτερο τεταρτημόριο, τότε το τελικό ημίτονο είναι η τιμή του συνημίτονου του υπολοίπου της γωνίας και το τελικό συνημίτονο είναι η αντίθετη τιμή του ημιτόνου της γωνίας.
- αν η αρχική γωνία είναι στο τρίτο τεταρτημόριο, τότε το τελικό ημίτονο είναι η αντίθετη τιμή του ημιτόνου του υπολοίπου της γωνίας και το τελικό συνημίτονο είναι η αντίθετη τιμή του συνημίτονου της γωνίας.
- αν η αρχική γωνία είναι στο τέταρτο τεταρτημόριο, τότε το τελικό ημίτονο είναι η αντίθετη τιμή του συνημίτονου του υπολοίπου της γωνίας και το τελικό συνημίτονο είναι η τιμή του ημιτόνου της γωνίας.

Η τελική σχεδίαση της μονάδας φαίνεται στο επόμενο σχήμα (Σχήμα 6.2).

Για να βρούμε σε ποιο τεταρτημόριο είναι η αρχική γωνία expArg χρησιμοποιήσαμε τη μονάδα που κρατάει το ακέραιο μέρος ενός αριθμού κινητής υποδιαστολής, η οποία πάλι βλέπει τον εκθέτη, ολισθαίνει τον αριθμό αριστερά και κρατάει τα ψηφία του ακέραιου μέρους. Εμείς σε αυτή τη μονάδα περνάμε το πηλίκο q για να κρατήσουμε το ακέραιο μέρος του. Η γωνία expArg μπορεί να εκφραστεί ως ένα γινόμενο q*π/2 ή ως p*π. Το q είναι το πηλίκο που έχουμε ήδη πάρει. Το p είναι το q διαιρεμένο με το 2 και βρίσκεται εύκολα αν αφαιρέσουμε από τον εκθέτη του q ένα (δηλαδή q[30:23] - 1). Τόσο το p όσο και το q το περνάμε από μονάδα κράτησης του ακέραιου μέρους τους (FP2Int). Αυτά τα δύο ακέραια μέρη τα θέλουμε για να εκμεταλλευτούμε το γεγονός ότι όταν το ακέραιο μέρος του p είναι ζυγό, τότε η γωνία είναι είτε στο πρώτο είτε στο δεύτερο τεταρτημόριο, ενώ όταν είναι μονό τότε η γωνία βρίσκεται είτε στο τρίτο είτε στο τέταρτο τεταρτημόριο. Ενώ όταν το ακέραιο μέρος του q είναι ζυγό, τότε η γωνία βρίσκεται είτε στο πρώτο είτε στο τρίτο τεταρτημόριο, και όταν το ακέραιο μέρος του q είναι μονό, τότε η γωνία βρίσκεται είτε στο δεύτερο είτε στο τέταρτο τεταρτημόριο. Συνδυάζοντας αυτά τα δύο ακέραια μέρη (cos_flag, sin_flag) δημιουργήσαμε σήμα ελέγχου (MUX INT SIGNAL CONTROL) που δείχνει σε πιο τεταρτημόριο βρίσκεται η αρχική γωνία expArg και πια έχουμε ολοκληρώσει την μονάδα υπολογισμού του ημιτόνου και του συνημίτονου για όλο το εύρος των αριθμών, αφού θα εφαρμόσουμε τις ιδιότητες για την εύρεση της τελικής τιμής αυτών των δύο συναρτήσεων.

Ένα τελευταίο σήμα που υπάρχει στη μονάδα είναι αν η γωνία είναι μεταξύ του –π/2 και του π/2 ή όχι (angle_flag). Αν είναι τότε περνάμε την αρχική γωνία στο τμήμα που υπολογίζει το ημίτονο και το συνημίτονο, ενώ αν δεν είναι, πρώτα κάνουμε την διαδικασία υπολογισμού του υπολοίπου κι έπειτα υπολογίζονται οι τιμές των τριγωνομετρικών συναρτήσεων. Συνδυάζοντας το angle_flag με το MUX INT SIGNAL CONTROL γίνεται η επιλογή για την τελική τιμή του συνημίτονου και του ημιτόνου της αρχικής γωνίας βάσει όλων των ιδιοτήτων που περιγράψαμε.

Η συγκεκριμένη μονάδα βγάζει αποτέλεσμα σε κάθε κύκλο μηχανής με αρχική καθυστέρηση 44 κύκλων. Στη συσκευή XC6VLX760 πιάνει 2588 slices και έχει μέγιστη συχνότητα 101.956MHz.



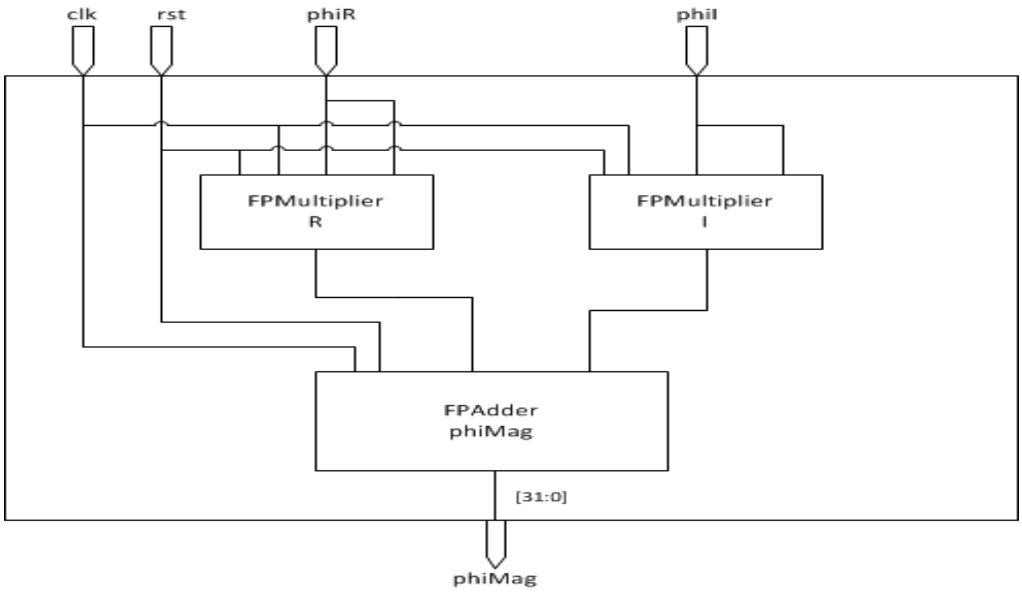

Σχήμα 6.2 Μονάδα υπολογισμού ημιτόνου – συνημίτονου για όλο το εύρος των γωνιών

### 6.2.2 Υπολογισμός phiMag

Για τον υπολογισμό του phiMag δημιουργήσαμε μια ξεχωριστή μονάδα. Έχει δύο εισόδους εύρους 1 bit για το ρολόι και το reset και δύο εισόδους των 32 bit για το phiR και phiI. Επίσης, έχει μία έξοδο των 32 bit για το phiMag.

Αποτελείται από δύο μονάδες πολλαπλασιασμού κινητής υποδιαστολής στις οποίες πολλαπλασιάζονται τα phiR και phiI με τον εαυτό τους και μία μονάδα πρόσθεσης κινητής υποδιαστολής, στην οποία προσθέτουμε τα αποτελέσματα των μονάδων πολλαπλασιασμού.

Η μονάδα βγάζει αποτέλεσμα σε κάθε κύκλο μηχανής με αρχική καθυστέρηση 8 κύκλων μηχανής. Η έξοδος είναι 32 bit παρότι η μονάδα πρόσθεσης έχει έξοδο 34 bit. Αυτό γίνεται κρατώντας μόνο τα 32 bit.




**Σχήμα 6.3 Μονάδα υπολογισμού του phiMag**

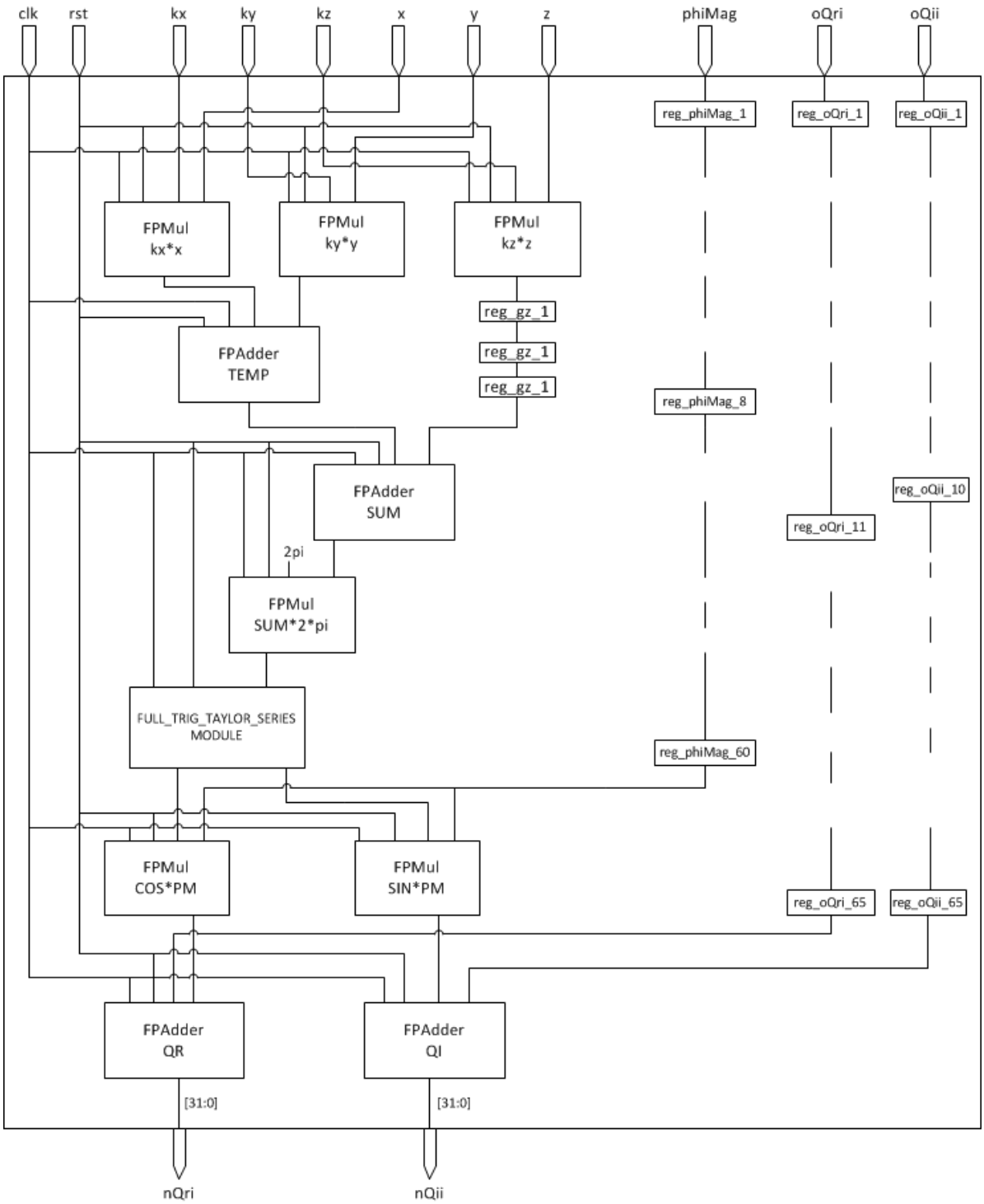
### 6.2.3 Υπολογισμός Q

Σε αυτή τη μονάδα (Σχήμα 6.4) γίνεται ο υπολογισμός της γωνίας expArg, η εύρεση του ημιτόνου και συνημιτόνου της και ο τελικός υπολογισμός των Qr και Qi. Έχει δύο εισόδους του 1 bit για το ρολόι και το reset και άλλες εννιά εισόδους των 32 bit για τα x, y, z, kx, ky, kz, phiMag και τις παλιές τιμές των Qr και Qi. Επίσης, έχει δύο εξόδους των 32 bit για τα τελικά αποτελέσματα Qr και Qi. Σε κάθε κύκλο μηχανής βγάζει αποτέλεσμα με αρχική καθυστέρηση 68 κύκλους.

Οι μονάδες πολλαπλασιασμού X, Y, Z υλοποιούν τα γινόμενα kx*x, ky*x, kz*z και στη συνέχεια με τις δύο μονάδες πρόσθεσης βρίσκουμε το άθροισμά τους. Στη μονάδα πολλαπλασιασμού SUM*2*pi υπολογίζουμε την γωνία expArg πολλαπλασιάζοντας το άθροισμα με το 2π. Έπειτα, με τη μονάδα υπολογισμού του ημιτόνου και του συνημίτονου, υπολογίζουμε τις δύο αυτές τριγωνομετρικές συναρτήσεις για την γωνία expArg. Στη συνέχεια με τις μονάδες πολλαπλασιασμού C και S βρίσκουμε το γινόμενο phiMag*cos(expArg) και phiMag*sin(expArg) για να υπολογίσουμε το τελικό άθροισμα της

παλιάς τιμής του Q με αυτά τα γινόμενα. Παρατηρούμε στο σχήμα ότι οι τιμές phiMag, oQr και oQi κρατούνται σε καταχωρητές μέχρι να χρησιμοποιηθούν.

Πάλι για να πάρουμε έξοδο με 32 bit κόβουμε από τις εξόδους των μονάδων πρόσθεσης κινητής υποδιαστολής τα δύο πιο σημαντικά bit.

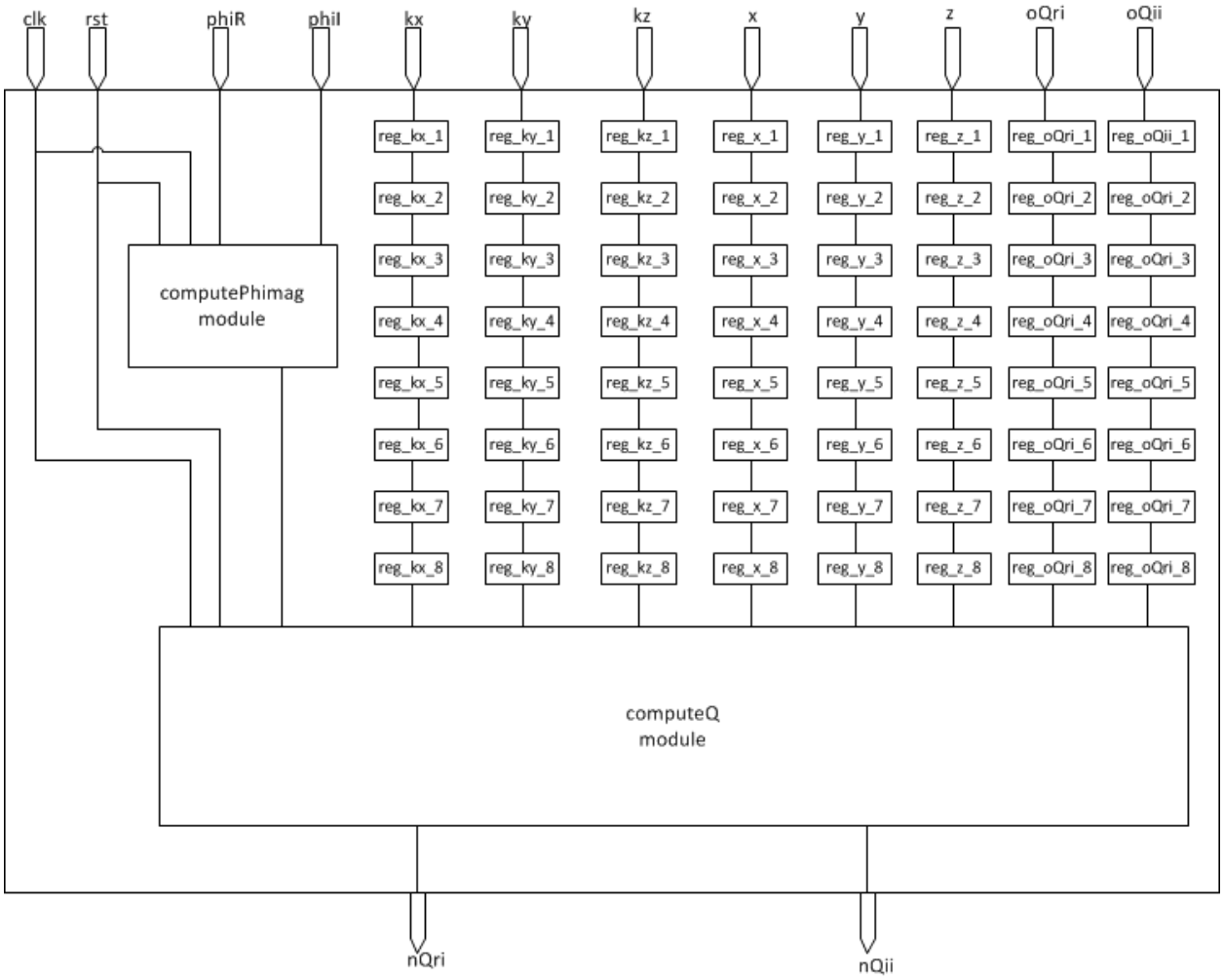



**Σχήμα 6.4 Μονάδα υπολογισμού των τιμών Qr και Qi**

### 6.2.4 Τελική μονάδα MRI Q

Τις παραπάνω μονάδες τις συνδυάσαμε σε μια άλλη για να πάρουμε το τελικό αποτέλεσμα και να γίνουν οι μετρήσεις για την FPGA. Έχει εισόδους 1 bit για το ρολόι και το reset και δέκα εισόδους των 32 bit για τα x, y, z, kx, ky, kz, phiMag και τις παλιές τιμές των Qr και Qi. Επίσης, έχει δύο εξόδους των 32 bit για τις καινούριες τιμές των Qr και Qi.

Το σχεδιάγραμμα της μονάδας φαίνεται παρακάτω:




Σχήμα 6.5 Τελική μονάδα MRI Q

Βλέπουμε ότι τα x, y, z, kx, ky, kz, oQr και oQi περιμένουν σε καταχωρητές για οχτώ κύκλους μηχανής μέχρι να γίνει το phiMag διαθέσιμο από την αντίστοιχη μονάδα για να γίνουν οι υπόλοιποι υπολογισμοί στη μονάδα computeQ.

## 6.3 Implementation – Αποτελέσματα

Την μονάδα MRI Q την κάναμε σύνθεση (Synthesis) κι υλοποίηση (Implementation) με το πρόγραμμα XILINX ISE για την FPGA της οικογένειας VIRTEX6 και συγκεκριμένα στη συσκευή XC6VLX760.

Όσον αφορά το ρολόι πήραμε το παρακάτω αποτέλεσμα:
Ελάχιστη περίοδος: 9.835ns
Μέγιστη Συχνότητα: 101.676MHz

Παραθέτουμε τα αποτελέσματα που βγήκαν μετά τη διαδικασία του Place and Route.

| **MRI Q DEVICE UTILIZATION SUMMARY** | | | |
|---|---|---|---|
| | **Used** | **Available** | **Utilization** |
| **Slice Logic Utilization** | | | |
| *Number of Slice Registers* | 8,924 | 948,480 | 1% |
| Number used as Flip Flops | 8,924 | | |
| *Number of Slice LUTs* | 10,762 | 474,240 | 2% |
| Number used as Logic | 9,199 | 474,240 | 1% |
| Number using O6 output only | 6,572 | | |
| Number using O5 output only | 1,101 | | |
| Number using O5 and O6 | 1,526 | | |
| Number used as Memory | 943 | 132,480 | 1% |
| Number used as Shift Register | 943 | | |
| Number using O6 output only | 745 | | |
| Number using O5 output only | 66 | | |
| Number using O5 and O6 | 132 | | |
| Number used exclusively as route-thrus | 620 | | |
| Number with same-slice register load | 563 | | |
| Number with same-slice carry load: | 56 | | |
| Number with other load | 1 | | |

Συνέχεια από τον προηγούμενο πίνακα:

| **Slice Logic Distribution** | | | |
|---|---|---|---|
| *Number of occupied Slices* | 3,667 | 118,560 | 3% |
| *Number of LUT Flip Flop pairs used* | 12,297 | | |
| Number with an unused Flip Flop | 4,810 | 12,297 | 39% |
| Number with an unused LUT | 1,535 | 12,297 | 12% |
| Number of fully used LUT-FF pairs | 5,952 | 12,297 | 48% |
| **IO Utilization** | | | |
| *Number of bonded IOBs* | 384 | 1200 | 32% |
| **Specific Feature Utilization** | | | |
| *Number of BUFG/BUFGCTRLs* | 2 | 32 | 6% |
| Number used as BUFGs | 2 | | |
| *Number of DSP48E1s* | 32 | 864 | 3% |

# Κεφάλαιο 7

# Επίλογος

Σε αυτή την διπλωματική είδαμε ότι υπάρχουν διάφορα μετροπρογράμματα σε επίπεδο λογισμικού και μετατρέψαμε κάποια από αυτά σε επίπεδο υλικού για να υπάρχει σύγκριση και με αυτό. Είδαμε τι είναι οι συσκευές FPGA, για τις οποίες σχεδιάσαμε τις μονάδες μετροπρογραμμάτων, καθώς και την διαδικασία που ακολουθεί κάποιος για να φορτώσει τελικά την σχεδίασή του σε μια συσκευή FPGA. Είδαμε το σύνολο μετροπρογραμμάτων PARBOIL από το οποίο πήραμε τους αλγορίθμους που μετατρέψαμε σε επίπεδο υλικού. Τέλος, είδαμε πώς σχεδιάσαμε τρία μετροπρογράμματα σε επίπεδο υλικού και τα αποτελέσματα που πήραμε από τις FPGA.

Τα μετροπρογράμματα είναι χρήσιμα για τη μελέτη απόδοσης συστημάτων. Το να δημιουργηθούν μετροπρογράμματα και σε επίπεδο υλικού είναι εξίσου χρήσιμο, μιας και το υλικό έχει εξαιρετικές δυνατότητες σε σχέση με το λογισμικό και όσο προχωράει η τεχνολογία και μειώνεται το κόστος και το μέγεθος του υλικού τόσο θα στηριζόμαστε σε αυτό και η ανάγκη για μελέτη της απόδοσής του θα είναι μεγάλη.

## 7.1 Μελλοντική προέκταση

Στόχος είναι να μετατραπούν κι άλλοι αλγόριθμοι σε επίπεδο υλικού για να υπάρχει ένα μεγάλο εύρος από μετροπρογράμματα σε επίπεδο υλικού. Είδαμε ότι πολλοί αλγόριθμοι χειρίζονται αριθμούς κινητής υποδιαστολής. Οι μονάδες για τις πράξεις τους μπορούν να βελτιωθούν κι άλλο κι έτσι τα μετροπρογράμματα θα βελτιώσουν κι άλλο την απόδοσή τους.

# Βιβλιογραφία


[1] http://www.asic-world.com/verilog/veritut.html

[2] http://inf-server.inf.uth.gr/courses/CE430/index.php

[3] http://www.h-schmidt.net/FloatConverter/IEEE754.html

[4] http://www.xilinx.com

[5] http://en.wikipedia.org/wiki/Field-programmable_gate_array

[6] http://students.ceid.upatras.gr/~mprokala/techarticles/FPGA/VirtexPresentation/virtex.htm

[7] http://www.1-core.com/library/digital/fpga-logic-cells/

[8] Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing

[9] http://impact.crhc.illinois.edu/parboil.aspx

[10] Σ. Θεοδωρίδης, Κ. Μπερμπερίδης, Λ. Κοφίδης. Εισαγωγή στη θεωρία σημάτων και συστημάτων

[11] http://www.dspguide.com/ch12/2.htm

[12] http://net.pku.edu.cn/~course/cs101/2007/resource/Intro2Algorithm/book6/chap32.htm

[13] http://www.relisoft.com/science/physics/fft.html

[14] http://en.wikipedia.org/wiki/Fast_Fourier_transform

[15] http://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm

[16] http://www.hep.upatras.gr/class/download/psi_epe_iko/kef2.pdf

[17] http://en.wikipedia.org/wiki/CORDIC

[18] David Kirk/NVIDIA and Wen-mei Hwu. Chapter 7: Application Case Study – Quantitative MRI Reconstruction

[19] J.A.Stratton, C.Rodrigues, I-J.. Sung, N.Obeid, Li-Wen Chang, N.Anssari, G.D.Liu, Wen-mei W. Hwu. Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing

[20] http://www.revisemri.com/tutorials/