ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών

Διπλωματική Εργασία

Πειραματική Αξιολόγηση Απόδοσης Μερικώς Αναδιατασσόμενου Συστήματος για Εφαρμογές Αντοχής σε Σφάλματα

Ανάργυρος Ηλίας

Εξεταστική Επιτροπή:
Απόστολος Δόλλας , Καθηγ. Π.Κ. (Επιβλέπων)
Διονύσιος Πνευματικάτος , Αν. Καθηγ. Π.Κ.
Ιωάννης Παπαευσταθίου , Επίκ. Καθηγ. Π.Κ.

Χανιά 2009

ΕΥΧΑΡΙΣΤΙΕΣ

Πρώτα απ’όλους θα ήθελα να ευχαριστήσω τον καθηγητή Α. Δόλλα, για την καθοδήγηση του κατά τη διάρκεια υλοποίησης αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να δουλέψω στο εργαστήριο Μικροεπεξεργαστών και Υλικού.

Επίσης θα ήθελα να ευχαριστήσω τον αναπληρωτή καθηγητή κ Διονύσιο Πνευματικάτο και τον επίκουρο καθηγητή κ. Ιωάννη Παπαευσταθίου, οι οποίοι δέχτηκαν να αξιολογήσουν την διπλωματική μου εργασία.

Ευχαριστώ τον κ. Κυπριανό Παπαδημητρίου διδακτορικό φοιτητή, για τη βοήθεια καθώς και τις συμβουλές του σε διάφορα σημεία της εργασίας.

Θα ήθελα να ευχαριστήσω τον κ. Μ. Κιμιωνή και τους προπτυχιακούς, μεταπτυχιακούς και διδακτορικούς φοιτητές του εργαστηρίου Μικροεπεξεργαστών και Υλικού για τη βοήθεια τους όποτε αυτή χρειάστηκε, καθώς και για το συνεχές ενδιαφέρον που έδειχναν για την πρόοδο της εργασίας αυτής.

Τους συμφοιτητές μου και τους φίλους που απέκτησα κατά την διάρκεια των σπουδών μου.

Τέλος περισσότερο απ' όλους, ευχαριστώ την οικογένεια μου για την βοήθεια,την συμπαράσταση και υπομονή τους σε όλα τα χρόνια σπουδών μου.

Περίληψη

Τα τελευταία χρόνια έχει συντελευτεί ραγδαία πρόοδος τόσο στην τεχνολογία ανάπτυξης όσο και στην αρχιτεκτονική των FPGAs. Οι παραπάνω εξελίξεις έχουν ως αποτέλεσμα την αύξηση σε πόρους, της απόδοσης των SRAM FPGAs καθώς και την ενσωμάτωση επιπλέον χρήσιμων μονάδων όπως μνήμες, πολλαπλασιαστές και μικροεπεξεργαστές

Τα παραπάνω χαρακτηριστικά καθιστούν τις FPGAs ιδανικές επιλογές για την ανάπτυξη εφαρμογών. Ιδιαίτερα η δυνατότητα τους για αναδιάταξη καθιστά εφικτή την πολυπλεξία πόρων την δυνατότητα ενημέρωσης των αλγορίθμων καθώς και την διόρθωση σχεδιαστικών σφαλμάτων.

Όμως η συρρίκνωση των τρανζίστορ καθώς και η αύξηση της πυκνότητας μιας FPGA επιφέρουν και αρνητικά αποτελέσματα όπως η αύξηση της πιθανότητας αντιστροφής κάποιου bit της μνήμης αναδιάταξης εξαιτίας κύμανσης της τάσης η εξαιτίας κάποιου ηλεκτρομαγνητικού πεδίου ή ακτινοβολίας.

Παρόλο που οι FPGAs είναι ευάλωτες στην ακτινοβολία έχουν επιλεγεί για υλοποίηση εφαρμογών σε αποστολές διαστήματος αντί των FPGAs τεχνολογίας anti-fuse εξαιτίας της μεγαλύτερης χωρητικότητας τους, της καλύτερης απόδοσης τους και της δυνατότητας αναδιάταξης τους .

Περιεχόμενα

1. Εισαγωγή **9**

1.1 Τεχνολογία των SRAM Based FPGAs……. 9

1.1.1 Περιγραφή Ενός Γενικού Μοντέλου …… 9

1.1.2 Περιγραφή της Xilinx Virtex–II Pro XC2VP30 …… 11

1.2 Μερική Αναδιάταξη ……. 12

1.2.1 Ροή Deference Based Μερική Αναδιάταξη……. 13

1.2.2 Ροή Module Based Μερική Αναδιάταξη…… 14

1.3 Σφάλματα Σε FPGAs…… 16

1.3.1 Πως η Ακτινοβολία Επηρεάζει μια FPGA……. 16

1.4 Δική μας Συνεισφορά ……... 19

1.5 Εργαλεία και Πλατφόρμα Ανάπτυξης…….. 19

1.6 Χρησιμοποίηση των FPGA σε Εφαρμογές Αντοχής Σφαλμάτων…… 20

2 Σχετικές Εργασίες Δυναμικής Αναδιάταξης και Ανάπτυξης Εφαρμογών Αντοχής Σφαλμάτων με Χρήση FPGAs **21**

2.1 Δυναμική Αναδιάταξη……. 21

2.2 Αντιμετώπιση Σφαλμάτων σε FPGAs…….. 22

2.2.1 Ανοχή σφαλμάτων σε επίπεδο συσκευής (Device Level)…. 22

2.2.2 Αντιμετώπιση Σφαλμάτων σε Επίπεδο Διάταξης της FPGA (Configuration – level)……. 23

2.2.2.1 Μέθοδοι Ανίχνευσης Σφαλμάτων …… 23

2.2.2.2 Μέθοδοι Αντιμετώπισης Σφαλμάτων …….. 27

2.2.2.3 Εγγύηση Λειτουργίας των Μονάδων Αναδιάταξης. 30

3 Υλοποίηση ενός FIR φίλτρου για την Εξαγωγή Πειραμάτων ……..…….. **32**

3.1 Υλοποίηση ενός FIR φίλτρου για την Εξαγωγή Πειραμάτων …… 32

3.1.1 Υλοποίηση στην VIRTEX – II PRO…… 33

4 Αρχιτεκτονική Συστήματος……... **36**

4.1 Γενική Λειτουργία Συστήματος ……. 37

4.2 FSM Ανάγνωσης Δεδομένων ……. 38

4.3 FIFO Δεδομένων …… 40

4.3.1 Υλοποίηση του Controller της FIFO……..…….. 40

4.4 Σύστημα Επεξεργαστών…….. 41

4.4.1 Υλοποίηση Επικοινωνίας Επεξεργαστών ……..….. 42

4.4.2 Υποσύστημα Επεξεργαστή PPC0……. 44

4.4.3 Υποσύστημα Επεξεργαστή PPC1…… 46

5 Υλοποίηση Συστήματος……. **48**

5.1 Εργαλεία Υλοποίησης…….. 48

5.2 Υλοποίηση Στατικής Μονάδας……..... 48

5.3 Υλοποίηση Δυναμικά Αναδιατασσόμενων Φίλτρων ……. 49

5.3.1 Τοποθέτηση σε μία PRR……... 50

5.3.2 Τοποθέτηση σε διαφορετικές PRR ……... 52

5.4 Υλοποίηση FIFO Controller……. 54

5.5 Ανάπτυξη Λογισμικού Επεξεργαστών…….. 56

5.5.1 Λογισμικό PPC0…… 56

5.5.2 Λογισμικό PPC1…… 59

6 Θεωρητική Ανάλυση Συστήματος…….……... **62**

6.1 Συσχέτιση Βάθους FIFO με Ρυθμό Εισόδου/Εξόδου και Χρόνου Δυναμικής Αναδιάταξης…….. 62

7 Πειραματικά Αποτελέσματα …….. **65**

7.1 Προετοιμασία Πειραματικών Μετρήσεων……...…… 65

7.1.1 Μέθοδος Εισαγωγής Σφάλματος στην Σχεδίαση…….. 65

7.1.2 Μεθοδολογία Λήψης Μετρήσεων……. 66

7.1.3 Δεδομένα Εισόδου …… 66

7.2 Σκοπός Πειραμάτων…….. 67

7.3 Σταθερές Παράμετροι του Συστήματος……. 68

7.4 Χρόνοι Αναδιάταξης των Μονάδων του FIFO Controller……. 69

7.5 Απόδοση Σχεδίασης όπου τα δύο Φίλτρα Τοποθετούνται στην ίδια PRR……. 70

7.6 Απόδοση Σχεδίασης όπου τα δύο Φίλτρα Τοποθετούνται σε Διαφορετικές PRRs…… 74

7.7 Σύγκριση Σχεδιάσεων……. 78

8 Συμπεράσματα και Μελλοντική Εργασία……... **80**

Αναφορές…… **83**

Παράρτημα Α…….. **86**

1 Εισαγωγή

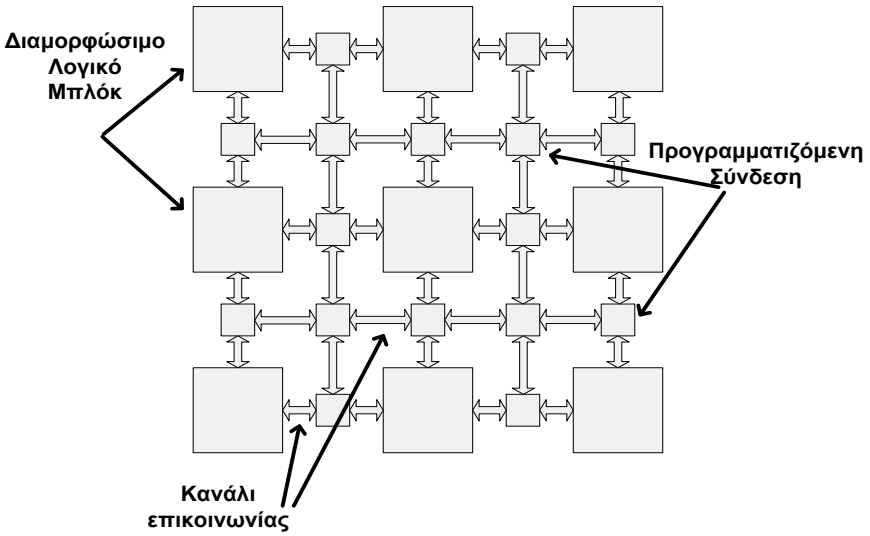
1.1 Τεχνολογία των SRAM Based FPGAs

Η δυνατότητα ολοκληρωμένων ψηφιακών κυκλωμάτων, τα οποία περιέχουν αναδιατασσόμενη λογική, να υλοποιούν μια ευρεία γκάμα λογικών συναρτήσεων και να μπορούν να μεταβάλουν την λειτουργικότητα τους σε σύντομο χρονικό διάστημα έχει ως αποτέλεσμα τον συνδυασμό ευελιξίας παρόμοιας των εφαρμογών λογισμικού με την απόδοση εφαρμογών υλοποιημένων σε hardware. Επιπλέον η δυνατότητα αναδιάταξης δίνει το πλεονέκτημα αναβάθμισης της λειτουργίας καθώς και της διόρθωσης σχεδιαστικών λαθών με μηδενικό επιπλέον κόστος σε υλικό. Για τους παραπάνω λόγους καθώς και επιπλέων χαρακτηριστικών κυκλώματα με αναδιατασσόμενη λογική χρησιμοποιούνται για την ανάπτυξη αρκετών εφαρμογών.

Στην παρούσα ενότητα παρουσιάζουμε μια γενική περιγραφή της αρχιτεκτονικής μίας κατηγορίας ολοκληρωμένων ψηφιακών κυκλωμάτων τα οποία περιέχουν αναδιατασσόμενη λογική των FPGAs καθώς και μία αναλυτικότερη παρουσίαση της FPGA την οποία χρησιμοποιήσαμε για την υλοποίηση της διπλωματικής εργασίας.

1.1.1 Περιγραφή Ενός Γενικού Μοντέλου

Οι FPGAs (field programmable gate arrays) είναι ψηφιακά ολοκληρωμένα κυκλώματα τα οποία διαθέτουν κάποια διάταξη προγραμματιζόμενων μπλοκ λογικής καθώς και κάποια ιεραρχία καναλιών διασύνδεσης και προγραμματιζόμενων συνδέσεων μεταξύ των λογικών μπλοκ η οποία τους επιτρέπουν να συνδέονται και να υλοποιούν μεγαλύτερα κυκλώματα .Ένα απλό γενικό μοντέλο μιας FPGA δίνεται στο σχήμα 1.1



Σχήμα 1.1 γενική δομή μιας FPGA

Οι FPGAs είναι προγραμματιζόμενες υπό την έννοια ότι τα λογικά μπλοκ τα οποία διαθέτουν έχουν διαθέσιμα look up tables (LUTs) με τα οποία μπορούν να υλοποιήσουν ένα ευρύ φάσμα από λογικές συναρτήσεις προγραμματίζοντας τα κατάλληλα . Επιπλέον μέσω των προγραμματιζόμενων συνδέσεων και των καναλιών επικοινωνίας μπορούμε να συνδέσουμε τις εξόδους του ενός μπλοκ με τις εισόδους ενός άλλου και να υλοποιήσουμε μεγαλύτερα ψηφιακά κυκλώματα. Η πληροφορία η οποία απαιτείται για τον προγραμματισμό της FPGA (τόσο των λογικών μπλοκ όσο και των συνδέσεων) αποθηκεύεται σε μια SRAM η οποία ονομάζεται Configuration Memory. Για να υλοποιήσουμε ένα κύκλωμα με μια FPGA συνήθως το περιγράφουμε μέσω μιας γλώσσας HDL (Vhdl ,Verilog) και έπειτα με τη χρήση κάποιας αλυσίδας εργαλείων παράγουμε το configuration file το οποίο περιέχει εντολές και τα δεδομένα (configuration bitstream)για τον προγραμματισμό της FPGA. Οι πρώτες FPGA αναπτύχθηκαν από την εταιρία Xilinx στα μέσα του 1985. Οι FPGAs υστερούν των ASIC chips όσον αφορά την ταχύτητα την κατανάλωση ισχύς το μέγεθος και το κόστος ανά μονάδα συγκεκριμένα:

- Στις συσκευές FPGA ένα σημαντικό ποσοστό της σιλικόνης καταναλώνεται για την υλοποίηση της δυνατότητας της αναδιάταξης. Η ύπαρξη της configuration memory και του γεγονότος ότι συνήθως μια εφαρμογή δεν εκμεταλλεύεται το σύνολο των διαθέσιμων πόρων έχει ως αποτέλεσμα την χρησιμοποίηση μεγαλύτερου κυκλώματος για την ανάπτυξη μίας εφαρμογής συγκριτικά με την δημιουργία ενός ASIC.
- Η ταχύτητα ενός ψηφιακού κυκλώματος εκτός από την αρχιτεκτονική του κυκλώματος εξαρτάται και από το μέγεθος του κυκλώματος. Συνήθως ένα μεγαλύτερο ψηφιακό κύκλωμα είναι αργότερο εξαιτίας της μεγάλης παρασιτικής χωρητικότητας. Επιπλέον οι προγραμματιζόμενες συνδέσεις μίας FPGA παρουσιάζουν μεγάλη χωρητικότητα και αντίσταση μειώνοντας επιπλέον την ταχύτητα.
- Η κατανάλωση ισχύς σε ορισμένες εφαρμογές αποτελεί έναν από τους σημαντικότερους περιορισμούς. Ο μεγαλύτερος αριθμός transistors καθώς και η ύπαρξη πόρων μιας FPGA οι οποίοι δεν χρησιμοποιούνται από μια συγκεκριμένη εφαρμογή αυξάνει την κατανάλωση ισχύος έναντι των ASICs.
- Το κόστος ανά μονάδα των FPGAs έναντι των ASICs είναι μεγαλύτερο εξαιτίας των περισσότερων πόρων που απαιτούνται για την κατασκευή του κυκλώματος.

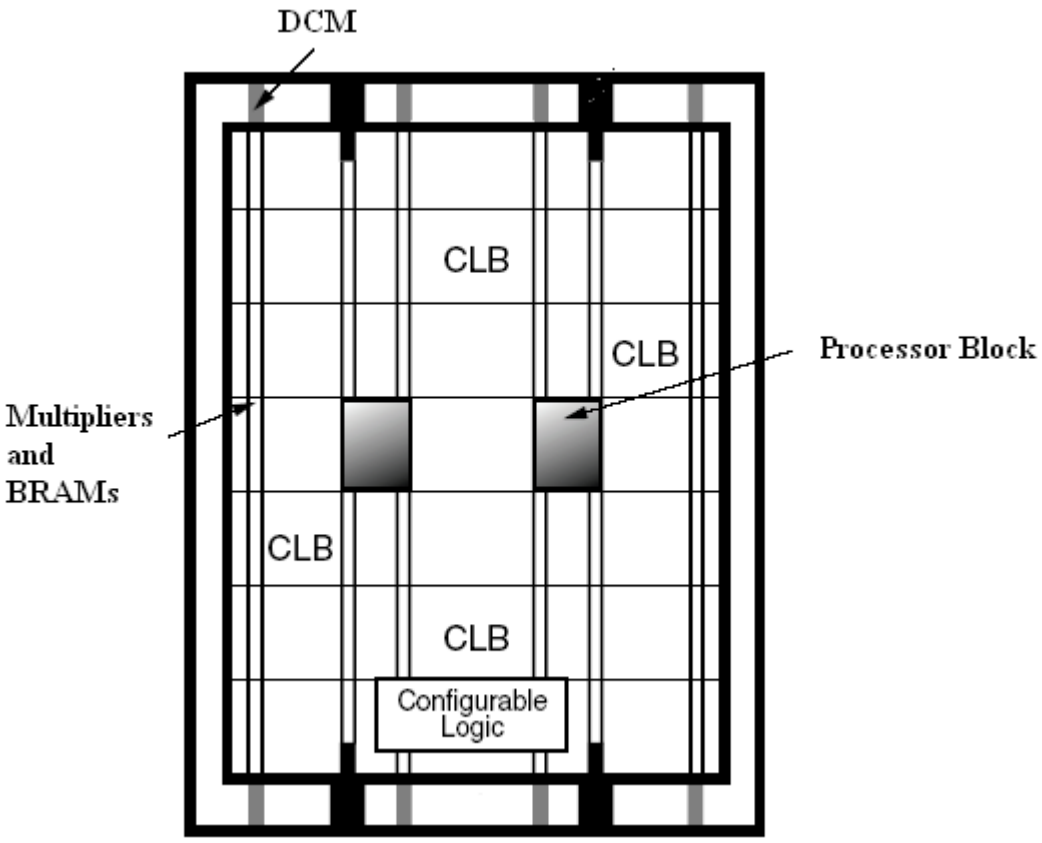
Παρόλα τα μειονεκτήματα έναντι των ASICs οι FPGAs παρουσιάζουν και μερικά ενδιαφέροντα πλεονεκτήματα έναντι τους, γεγονός που έχει ως αποτέλεσμα την προτίμηση τους σε ορισμένες περιπτώσεις έναντι των ASICs. Συγκεκριμένα το κόστος ανάπτυξης ενός

κυκλώματος ASIC είναι αρκετά μεγάλο και χρονοβόρο. Αυτό έχει ως αποτέλεσμα όταν ο αριθμός των chips που θα παραχθούν είναι μικρός να συμφέρει η χρησιμοποίηση των ήδη διαθέσιμων FPGAs. Επίσης ο απαιτούμενος χρόνος για την ανάπτυξη εφαρμογών μέσω FPGAs είναι αρκετά μικρός με αποτέλεσμα μικρό time-to-market κόστος. Τέλος λόγω της δυνατότητας επαναπρογραμματισμού έχουν την δυνατότητα αναβάθμισης της υλοποιημένης εφαρμογής με μηδενικό κόστος .

1.1.2 Περιγραφή της Xilinx Virtex–II Pro XC2VP30

Η συσκευή FPGA που χρησιμοποιήθηκε για την υλοποίηση της διπλωματικής εργασίας είναι η XC2VP30 της Xilinx η οποία ανήκει στην οικογένεια της Virtex II Pro. Τα διαμορφώσιμα λογικά μπλοκ (CLBs) της συσκευής τα οποία αποτελούν την γεννήτρια των λογικών συναρτήσεων οργανώνονται σε στήλες δημιουργώντας έναν δυσδιάστατο πίνακα. Στην περίμετρο του πίνακα αυτού υπάρχουν μπλοκ εισόδου/εξόδου (IOBs) τα οποία είναι υπεύθυνα για την διαχείριση των pins της FPGA. Για την δρομολόγηση των σημάτων εισόδου και εξόδου των CLBs χρησιμοποιείται μια ιεραρχική βαθμίδα (GRM) πινάκων δρομολόγησης οι οποίοι αποτελούνται από προγραμματιζόμενους διακόπτες τοποθετημένους στις τομές οριζοντίων και καθέτων καναλιών δρομολόγησης. Εκτός από τα παραπάνω η συγκεκριμένη FPGA περιλαμβάνει και ενσωματωμένες βαθμίδες μνήμης (BRAMS) πολλαπλασιαστές, βαθμίδες διαχείρησης ρολογιού (DCM), δύο ενσωματωμένους επεξεργαστές Power PC405 (ppc405) καθώς και μία βαθμίδα για την εσωτερική προσπέλαση της configuration memory το ICAP. Σε κάθε προγραμματίσιμη βαθμίδα αντιστοιχεί ένα καρτεσιανό ζεύγος συντεταγμένων όπου η αρχή των αξόνων είναι η κάτω αριστερή γωνία της FPGA. Με τον τρόπο αυτό μπούμε να επιλέξουμε συγκεκριμένους πόρους της συσκευής.

Η configuration memory της FPGA είναι οργανωμένη σε 1756 κάθετα frames μήκους 6592 bits το καθένα. Ένα frame είναι η μικρότερη μονάδα της configuration memory η οποία μπορεί να προγραμματιστεί. Υπάρχουν δυο τρόποι για να διαμορφωθεί η configuration memory της FPGA. Ο ένας είναι να διαμορφωθεί εξωτερικά συνδεόμενη με ένα σύστημα το οποίο θα γράψει τα configuration bitstreams στην configuration memory μέσω του JTAG και ο δεύτερος τρόπος είναι να γίνει εσωτερική διαμόρφωση όπου κάποιος εσωτερικός επεξεργαστής της FPGA στέλνει τα configuration bitstreams στην configuration memory μέσω του ICAP. Περισσότερες πληροφορίες για την αρχιτεκτονική της σειράς Virtex II FPGAs αναφέρονται στο παράρτημα Α.



Σχήμα 1.2 Διάταξη μίας Virtex II Pro

1.2 Μερική Αναδιάταξη

Μερική αναδιάταξη είναι η δυνατότητα μιας FPGA να μπορεί να προγραμματίσει μέρος της configuration μνήμης της και τα δεδομένα της υπόλοιπης μνήμης να παραμένουν ανέπαφα από αυτήν την διαδικασία. Η μερική αναδιάταξη προσφέρει σημαντικά πλεονεκτήματα συγκριτικά με τον επαναπρογραμματισμό ολόκληρης της configuration μνήμης της συσκευής.

Η μερική αναδιάταξη επιτρέπει την πραγματοποίηση σχεδιαστικών αλλαγών στα κυκλώματα για την αναβάθμιση των αλγορίθμων ή και την διόρθωση σχεδιαστικών σφαλμάτων σε συντομότερο χρονικό διάστημα εφόσον στις περισσότερες περιπτώσεις απαιτούνται αλλαγές σε τμήματα του κυκλώματος και όχι σε ολόκληρο.

Ένα ακόμα όφελος της μερικής αναδιάταξης είναι η δυνατότητα υλοποίησης δυναμικά αναδιατασσόμενων συστημάτων τα οποία ανάλογα με τις απαιτήσεις της εργασία που τους έχει ανατεθεί μπορούν να αναδιατάσσουν τμήματα τους ενώ το υπόλοιπο κύκλωμα να συνεχίζει την λειτουργία του ανεπηρέαστο. Η παραπάνω προοπτική καθιστά δυνατή την χρονική πολυπλεξία πόρων της FPGA το οποίο συνεπάγεται μείωση των απαιτούμενων πόρων για την υλοποίηση μιας σχεδίασης και μείωση της κατανάλωσης ενέργειας.

Τέλος μια νέα δυνατότητα η οποία καθιστάτε εφικτή με την μερική αναδιάταξη είναι η αντιμετώπιση σφαλμάτων τα οποία μπορούν να προκύψουν είτε από καταστροφή κάποιων πόρων μιας FPGA είτε εξαιτίας αντιστροφών bits της configuration μνήμης της λόγω εξωτερικών παραγόντων, οι οποίες μπορούν να προκαλέσουν αλλοίωση του κυκλώματος το οποίο υλοποιείται μέσω της FPGA. Με την μερική αναδιάταξη μπορούμε να αναδιατάξουμε το τμήμα της FPGA το οποίο επηρεάζεται από το σφάλμα και είτε να παρακάμψουμε τους κατεστραμμένους πόρους η να διορθώσουμε τα δεδομένα της μνήμης. Έτσι η μερική αναδιάταξη καθιστά εφικτή την ανάπτυξη fault tolerant εφαρμογών σε FPGAs.

Για την αξιοποίηση της μερικής αναδιάταξης έχουν δημιουργηθεί δυο σχεδιαστικές ροές η deference based partial reconfiguration και module based partial reconfiguration. Η πρώτη σχεδιαστική ροή χρησιμοποιείται για την πραγματοποίηση σχεδιάσεων οι οποίες διαφέρουν μεταξύ τους στα βασικά στοιχειά μιας FPGA όπως στις εξισώσεις που υλοποιούν τα LUTs η στα περιεχόμενα των BRAMs που διαθέτει η συσκευή. Η δεύτερη σχεδιαστική ροή χρησιμοποιεί την μερική αναδιάταξη για την δημιουργία σχεδιάσεων οι οποίες διαφέρουν σε ανώτερο επίπεδο από αυτό του προγραμματισμού των στοιχειωδών πόρων μιας FPGA. Η ροή αυτή χρησιμοποιεί την μερική αναδιάταξη για την πραγματοποίηση σχεδιάσεων οι οποίες διαφέρουν μεταξύ τους ως προς την διάταξη ορισμένων περιοχών της FPGA.

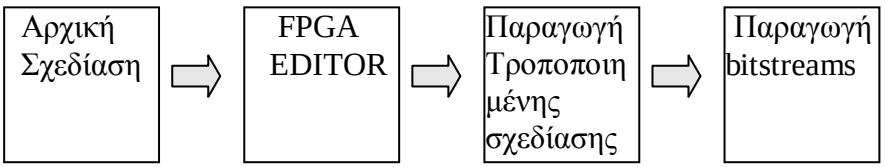
1.2.1 Ροή Deference Based Μερική Αναδιάταξη[1]

Η deference based μερική αναδιάταξη επιτρέπει στον σχεδιαστή να πραγματοποιήσει μικρές λογικές μετατροπές σε ένα σχέδιο και να παράγει ένα bitstream το οποίο πραγματοποιεί μόνον τις διαφορές των δυο σχεδιάσεων. Η εναλλαγή μεταξύ των δύο αυτών σχεδιάσεων γίνεται αρκετά γρήγορα εξαιτίας του γεγονότος ότι το παραγόμενο bitstream είναι συνήθως μικρότερο από αυτό που απαιτείται για τον προγραμματισμό ολόκληρης της FPGA.

Οι μετατροπές τις οποίες μπορεί να πραγματοποιήσει ένας σχεδιαστής μέσω αυτής της ροής στο αρχικό σχέδιο είναι να επαναπρογραμματίσει κάποια LUTs να αλλάξει τα δεδομένα ορισμένων block Rams και να αλλάξει το πρότυπο εισόδου/εξόδου. Επίσης αν και είναι δυνατό να αλλάξει το routing της σχεδίασης κάτι τέτοιο δεν συνιστάται.

Η ροή για την παραγωγή διαφορετικών σχεδιάσεων είναι η παρακάτω. Αρχικά παράγουμε το αρχείο τοποθέτησης και δρομολόγησης (NCD file) του κυκλώματος και με βάση αυτό το αρχείο θα υλοποιήσουμε τις διάφορες σχεδιάσεις. Έπειτα ανοίγουμε κάποιο αντίγραφο του παραπάνω αρχείου με τον FPGA_Editor και πραγματοποιούμε τις επιθυμητές μετατροπές.

Το τελευταίο στάδιο είναι η παραγωγή των bitstreams των διαφορετικών σχεδιάσεων. Συνήθως παράγουμε ένα bitstream το οποίο υλοποιεί την αρχική σχεδίαση και έπειτα ένα bitstream μερικής αναδιάταξης το οποίο περιέχει τις διαφορές μεταξύ των δυο σχεδιάσεων έτσι ώστε όταν επαναπρογραμματιστεί μερικά η FPGA να υλοποιήσουμε την δεύτερη σχεδίαση. Συνήθως παράγεται και ένα τρίτο bitstream μερική αναδιάταξης έτσι ώστε να είναι δυνατή η επαναφορά της αρχικής σχεδίασης μέσω μερικής αναδιάταξης.



Σχήμα 1.5 Απεικόνιση Deference Based ροής

1.2.2 Ροή Module Based Μερική Αναδιάταξη[2]

Η module based μερική αναδιάταξη σε αντίθεση με την deference based επιτρέπει στον σχεδιαστή να πραγματοποιήσει περισσότερο εκτενείς μετατροπές σε ένα κύκλωμα.

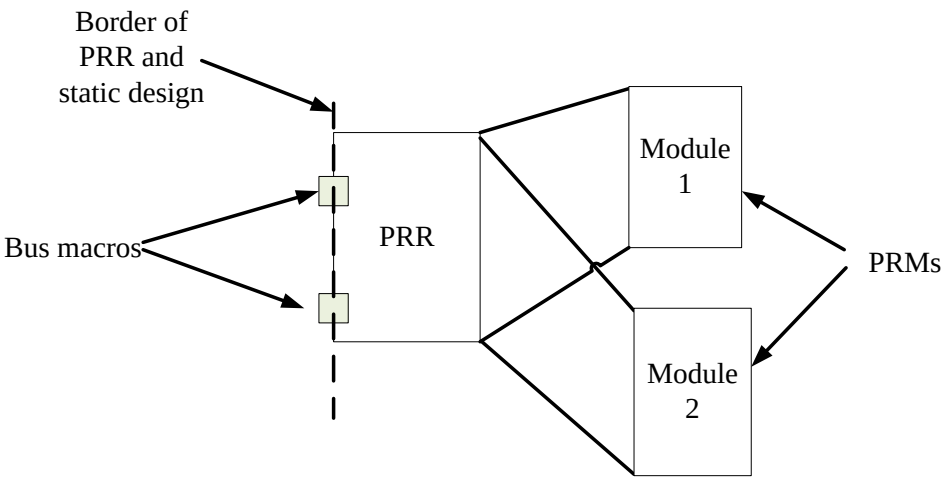
Για την δημιουργία ενός μερικά αναδιατασσόμενου συστήματος με την συγκεκριμένη ροή αρχικά πρέπει να γίνει διαχωρισμός του κυκλώματος σε στατικό και σε αναδιατασσόμενο. Με τον όρο στατικό εννοούμε τις μονάδες του κυκλώματος οι οποίες παραμένουν αμετάβλητες σε όλες τις διαφορετικές σχεδιάσεις. Με τον όρο αναδιατασσόμενο εννοούμε τις μονάδες οι οποίες μεταβάλλονται στις διάφορες σχεδιάσεις. Οι μονάδες αυτές καλούνται μερικά αναδιατασσόμενες μονάδες (Partial Reconfigurable Modules PRM) .

Μια σημαντική έννοια είναι αυτή της μερικά αναδιατασσόμενης περιοχής (Partial Reconfigurable Region PRR). Ουσιαστικά μια PRR ορίζει μια συγκεκριμένη περιοχή της FPGA η οποία θα αναδιατάσσεται μερικά. Οι πόροι κάθε PRR δεσμεύονται για την υλοποίηση ενός η περισσοτέρων PRM τα οποία δεν είναι ταυτοχρόνως απαραίτητα για την λειτουργία του συστήματος και επομένως μπορούμε να εφαρμόσουμε χρονική πολυπλεξία πόρων. Μια PRM μπορεί να είναι επίσης ανενεργή και να μην υλοποιεί καμία μονάδα. Μια PRR πρέπει να διαθέτει τόσους πόρους έτσι ώστε να μπορεί να υλοποιήσει κάθε PRM που της έχει ανατεθεί.

Ένα σημαντικό δεδομένο είναι ότι τα PRMs τα οποία υλοποιούνται στην ιδία PRR πρέπει να έχουν το ίδιο interface. Επομένως σε περιπτώσεις κατά τις οποίες τα PRMs δεν έχουν τα ιδία σήματα εισόδου/εξόδου θα πρέπει να υλοποιηθεί ένα γενικό interface το οποίο θα περιέχει όλα τα σήματα και επιπλέον τα PRMs θα πρέπει να τροποποιηθούν ώστε να διαχειρίζονται τις περιττές εισόδους/εξόδους.

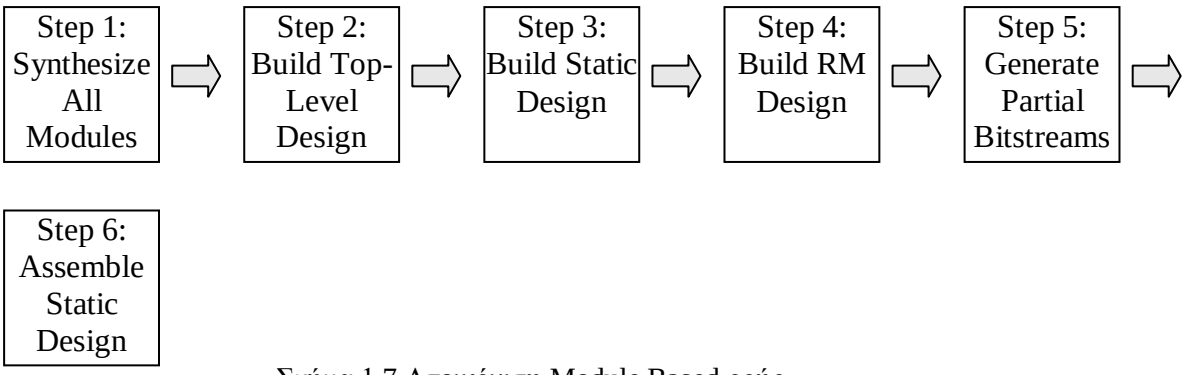
Η επικοινωνία μεταξύ των PRRs και της στατικής σχεδίασης γίνεται μεταξύ ειδικών μονάδων (bus macros) οι οποίες τοποθετούνται στα σύνορα μεταξύ στατικής σχεδίασης και

PRRs και η λειτουργία τους είναι να δρομολογήσουν τα σήματα της διεπαφής από και προς τις PRRs.



Σχήμα 1.6 Απεικόνιση των εννοιών της Module Based Partial Reconfiguration

Η ροή για την παραγωγή διαφορετικών σχεδιάσεων είναι η παρακάτω. Αρχικά συνθέτουμε όλες τις επιμέρους μονάδες της σχεδίασης στατικών και μη. Στο στάδιο αυτό αποκτάμε κάποιες αρχικές εκτιμήσεις για τους πόρους που απαιτούν οι PRMs για την υλοποίηση τους και με βάση αυτά τα στοιχεία να εκτιμήσουμε το μέγεθος κάθε PRR. Έπειτα δημιουργούμε το ανώτερο επίπεδο της σχεδίασης όπου ορίζουμε και τις PRRs της σχεδίασης και τοποθετούμε τα bus macros. Στο επόμενο στάδιο παράγουμε το αρχείο τοποθέτησης και δρομολόγησης (NCD file) του στατικού κυκλώματος και στο αμέσως επόμενο στάδιο παράγουμε τα αρχεία τοποθέτησης και δρομολόγησης των μερικά αναδιατασσόμενων μονάδων. Το επόμενο στάδιο είναι να παράγουμε τα partial bitstreams τα οποία περιέχουν τις απαραίτητες πληροφορίες για τον προγραμματισμό μιας PRR έτσι ώστε να υλοποιήσει κάποιο PRM που της έχει ανατεθεί. Επιπλέον των bitstreams τα οποία υλοποιούν τα PRMs της PRR κατά το στάδιο αυτό παράγονται και κενά bitstreams για τις PRRs. Το τελευταίο στάδιο της ροής είναι η συγχώνευση των στατικών μονάδων καθώς και των PRMs με τα οποία επιθυμούμε να αρχικοποιήσουμε τις PRR σε ένα αρχείο έτσι ώστε να φορτώνεται στην FPGA όταν αυτή θα αναδιατάσσεται αρχικά .



Σχήμα 1.7 Απεικόνιση Module Based ροής

1.3Σφάλματα σε SRAM FPGAs

Όπως κάθε κύκλωμα VLSI έτσι και οι SRAM FPGAs υπόκεινται στην φθορά του χρόνου λόγω μερικών φυσικών φαινόμενων τα οποία λαμβάνουν χώρα στα κυκλώματα και μερικές φορές εξαιτίας ατελειών στην διαδικασία παραγωγής ψηφιακών κυκλωμάτων. Έτσι φαινόμενα όπως τα hot carrier effect, electromigration και time – dependent Dielectric Breakdown καθώς και ψεγάδια τα οποία εισάγονται κατά την παραγωγή προκαλούν σε βάθος χρόνου μείωση της ταχύτητας απόκρισης μιας πύλης, βραχυκυκλώματα η ανοιχτό-κυκλώματα .

Επιπλέον η ιδιαιτερότητα μιας FPGA να μπορεί να επαναπρογραμματίζεται γράφοντας κατάλληλα bitstream στην configuration memory της συσκευής την καθιστά ιδιαίτερα ευπαθή σε μη επιθυμητές, λόγω εξωτερικών παραγόντων (πχ ακτινοβολία), αντιστροφές των υποθηκευμένων bit της configuration memory αλλάζοντας την δομή του κυκλώματος το οποίο υλοποιείται.

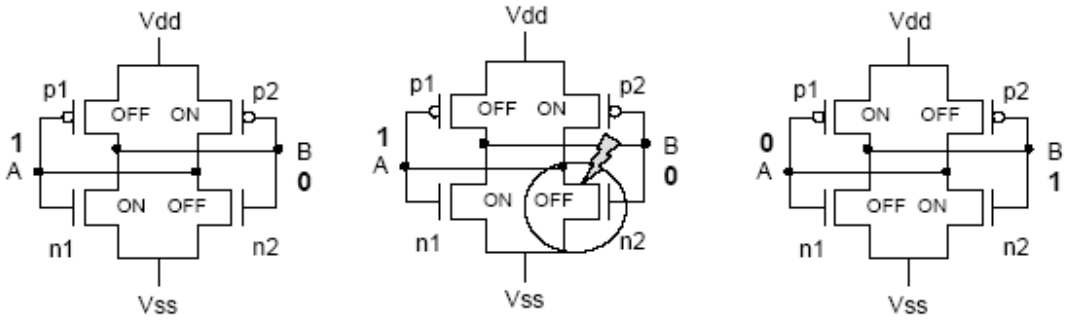
Έτσι σύμφωνα με τα παραπάνω τα σφάλματα στις FPGA μπορούν να χωριστούν σε δυο κατηγορίες ανάλογα με το αν πρόκειται για σφάλματα τα οποία οφείλονται σε φθορά του υλικού της FPGA (hard faults) η σε σφάλματα τα οποία οφείλονται στην αλλοίωση των δεδομένων της configuration memory (soft faults).

1.3.1 Πως η Ακτινοβολία Επηρεάζει μια SRAM FPGA [3]

Η ακτινοβολία μπορεί να προκαλέσει σοβαρά προβλήματα στα ηλεκτρονικά ενός συστήματος. Η κύρια ανησυχία από την ακτινοβολία του διαστήματος είναι το Single Event Effect (SEE) το οποίο συμβαίνει όταν φορτισμένα μόρια χτυπήσουν το πυρίτιο μεταφέρουν ικανή ποσότητα ενέργειας. Ένα SEE μπορεί να έχει καταστροφικές η παροδικές επιπτώσεις ανάλογα με την ποσότητα της ενέργειας που μεταφέρθηκε και το σημείο του χτυπήματος. Οι επιπτώσεις μιας παροδικής επίδρασης ονομάζονται Single Event Upset (SEU), και είναι

αντιστροφές bit σε στοιχεία μνήμης. Τα SEUs εξαιτίας των τεχνολογικών εξελίξεων που έχουν οδηγήσει σε πολύπλοκες αρχιτεκτονικές και μικρότερες κλίμακες αποτελούν έναν σοβαρό παράγοντα ανησυχίας για την σωστή λειτουργία μιας FPGA.

Όταν ένα φορτισμένο σωματίδιο χτυπήσει κάποιο από τους ευαίσθητους κόμβους ενός κελιού μνήμης(όπως τον εκπομπό ενός τρανζίστορ το οποίο είναι σε κατάσταση λειτουργίας off) τότε μπορεί να προκαλέσει έναν παροδικό παλμό ρεύματος ο οποίος μπορεί να προκαλέσει αντιστροφή της κατάστασης των τρανζίστορ τα οποία αποτελούν το κελί μνήμης και κατά συνέπεια να μεταβάλει την αποθηκευμένη τιμή .

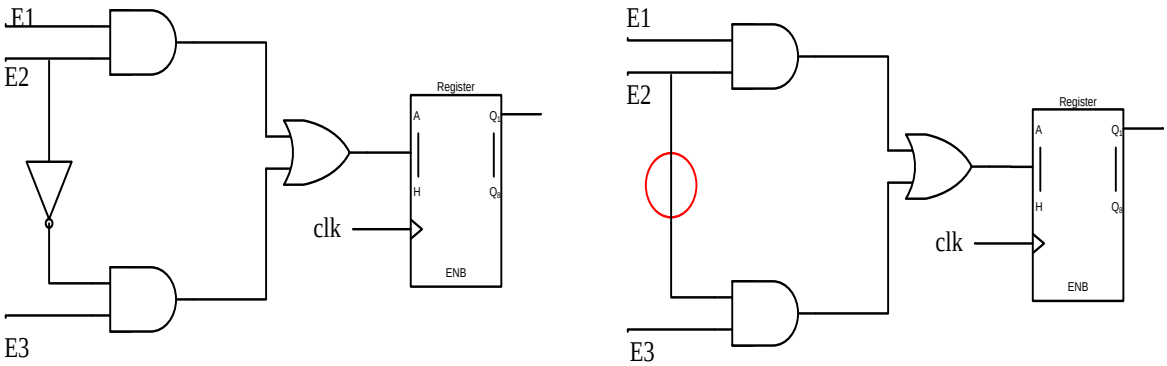


Σχήμα 1.8 Single Event Upset σε ένα κελί μνήμης SRAM . Αριστερά είναι η αρχικά αποθηκευμένη τιμή και δεξιά η τελική τιμή του αποθηκευμένου bit

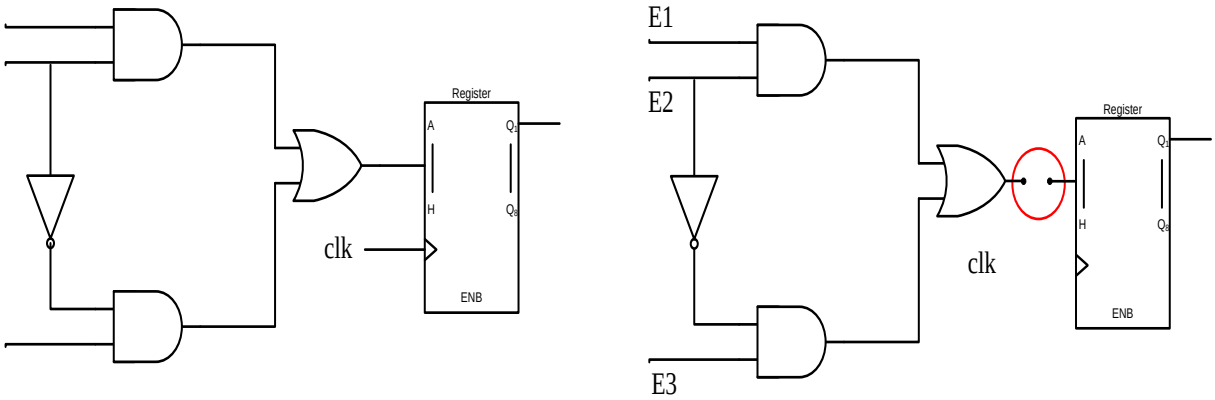
Στο σχήμα 1.8 βλέπουμε την επίδραση της ακτινοβολίας σε ένα κελί μνήμης SRAM την αρχική κατάσταση της μνήμης και την τελική κατάσταση της. Όταν συμβεί κάποιο SEU σε κάποιο κελί της configuration memory τότε υπάρχουν οι εξής περιπτώσεις :

-Το bit το οποίο αντιστράφηκε να ελέγχει την εξίσωση που υλοποιεί κάποιο LUT και κατά συνέπεια έχουμε κάποια αλλαγή της λογικής εξίσωσης.

-Το bit το οποίο αντιστράφηκε να ελέγχει την δρομολόγηση του αρχικού κυκλώματος και κατά συνέπεια να έχουμε κάποιο ανεπιθύμητο ανοιχτοκύκλωμα η βραχυκύκλωμα.



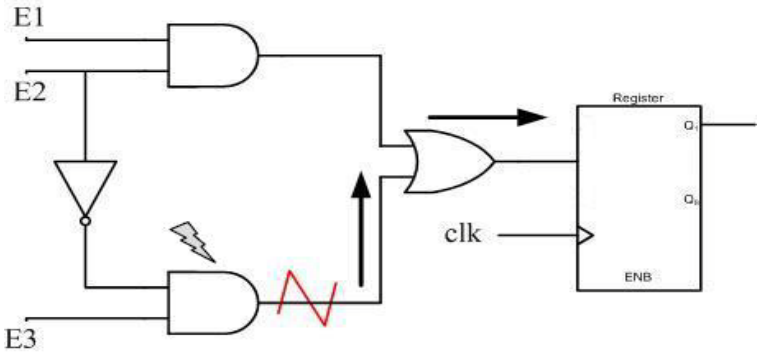
Σχήμα 1.9 Αλλαγή της εξίσωσης που υλοποιεί ένα LUT εξαιτίας ενός SEU.Αριστερά το αρχικό κύκλωμα δεξιά το αλλοιωμένο κύκλωμα.



Σχήμα 1.10 Αλλαγή της συνδεσμολογίας ενός κυκλώματος εξαιτίας ενός SEU . Αριστερά το αρχικό κύκλωμα δεξιά το αλλοιωμένο κύκλωμα

Σε περίπτωση οπού το κελί της μνήμης τα περιεχόμενα του οποίου αλλοιωθούν δεν ανήκουν στην configuration memory αλλά σε κάποια μνήμη όπου κρατούνται δεδομένα για τους υπολογισμούς του συστήματος (πχ καταχωρητές η block Rams) τότε το κύκλωμα θα υπολογίσει λάθος αποτελέσματα έως ώσπου να ξαναγράψει την συγκεκριμένη θέση μνήμης.

Εκτός από κάποιο κελί μνήμης ένα φορτισμένο σωματίδιο μπορεί να χτυπήσει κάποιο συνδυαστικό μπλοκ λογικής και να προκαλέσει έναν παροδικό παλμό ρεύματος ο οποίος εάν το κύκλωμα είναι αρκετά γρήγορο μπορεί να διαδοθεί στο κύκλωμα και να αλλάξει την έξοδο του. Το φαινόμενο αυτό ονομάζεται Single Event Transient (SET). Σε αυτή την περίπτωση δεν θα αλλάξει η δομή του κυκλώματος.



Σχήμα 1.11 Single Event Transient .Δημιουργία παλμού και διάδοση του προς την έξοδο του κυκλώματος

Τα σφάλματα τα οποία προκαλούνται από SEE είναι κατά κανόνα soft.Επίσης μια άλλη κατηγοριοποίηση που γίνεται στα σφάλματα αυτά είναι το αν είναι παροδικά (Transient) η μόνιμα λάθη (permanent). Τα SEU τα οποία συμβαίνουν στην configuration memory είναι μόνιμα υπό την άποψη ότι αλλοιώνεται η δομή του κυκλώματος και αυτό έχει σαν

αποτέλεσμα την μόνιμη παραγωγή λανθασμένων αποτελεσμάτων εφόσον δεν ανακτηθεί η σωστή δομή του κυκλώματος ενώ αντίθετα τα υπόλοιπα σφάλματα που περιγράψαμε ανήκουν στην κατηγορία των παροδικών σφαλμάτων.

Τέλος εκτός από επιδράσεις σε ένα μονό bit μπορεί να έχουμε και επιδράσεις σε περισσότερα του ενός bit. Το φαινόμενο αυτό ονομάζεται Multiple upsets bit (MTB). Το φαινόμενο αυτό μπορεί να συμβεί όταν ένα σωματίδιο χτυπήσει περισσότερες της μιας επαφές τρανζίστορ με αποτέλεσμα να έχουμε εναλλαγές σε αρκετές θέσεις μνήμης. Επίσης μπορεί να προκληθεί και από κάποιο SET όταν η έξοδος του κυκλώματος το οποίο επηρεαστεί οδηγεί τις εισόδους περισσοτέρων του ενός καταχωρητών.

1.4 Η Δική μας Συνεισφορά

Σκοπός της παρούσας διπλωματικής εργασίας είναι η μελέτη των δυνατοτήτων της μερικής αναδιάταξης για την ανάπτυξη δυναμικών αναδιατασσόμενων συστημάτων για εφαρμογές αντοχής σε σφάλματα. Η ροή η οποία χρησιμοποιείται είναι η Module Based. Κυρίως μελετάται ο απαιτούμενος χρόνος για την αναδιάταξη κάποιας σχεδίασης η επίδραση στην απόδοση του συστήματος καθώς επίσης οι περιορισμοί που επιβάλει ο χρόνος αυτός στην υλοποίηση του συστήματος και τρόποι για την μείωση του.

Στα πλαίσια της εργασίας αυτής δημιουργήθηκε και αξιολογήθηκε πειραματικά ένα δυναμικά αναδιατασσόμενο συστήματα δύο επεξεργαστών το οποίο λειτουργεί επικουρικά σε εφαρμογές αντοχής σφαλμάτων οι οποίες δεν είναι safety critical και επομένως μπορούν για μικρά χρονικά διαστήματα να είναι ανενεργές. Στόχοι του συστήματος αυτού είναι η αντιμετώπιση σφαλμάτων τα οποία συμβαίνουν στην εφαρμογή που εκτελείται στην FPGA και η ανάληψη της λειτουργίας της εφαρμογής έως ώσπου να ολοκληρωθεί η αναδιάταξη και να είναι έτοιμη η εφαρμογή που εκτελείται στην FPGA να αναλάβει ξανά την λειτουργία . Οι χρόνοι οι οποίοι μελετώνται είναι αυτοί της μερικής αναδιάταξης και ο χρόνος της επεξεργασίας δεδομένων από τον δεύτερο επεξεργαστή.

1.5 Εργαλεία και Πλατφόρμα Ανάπτυξης

Για την υλοποίηση της διπλωματικής χρησιμοποιήθηκε η πλατφόρμα ανάπτυξης XUPV2P της Digilent η οποία περιλαμβάνει μια Virtex II Pro στην οποία θα υλοποιήσουμε το σύστημα μας.

Τα εργαλεία CAD τα οποία χρησιμοποιήσαμε είναι τα εξής : το EDK της Xilinx με το οποίο δημιουργήσαμε το σύστημα των επεξεργαστών και γράψαμε τον κώδικα των

προγραμμάτων τα οποία εκτελούνται στους επεξεργαστές , το ISE της Xilinx με το οποίο υλοποιήσαμε τα απαραίτητα ψηφιακά κυκλώματα για την υλοποίηση του φίλτρου εφαρμογής και των βοηθητικών κυκλωμάτων και το Planahead της Xilinx με το οποίο ορίσαμε τις περιοχές στις οποίες τοποθετήθηκαν τα μέρη του συστήματος και μέσω του οποίου ολοκληρώσαμε την σχεδίαση.

1.6 Χρησιμοποίηση των FPGA σε Εφαρμογές Αντοχής Σφαλμάτων

Παραπάνω αναφερθήκαμε στην αρχιτεκτονική των SRAM FPGAs καθώς και στα πλεονεκτήματα και μειονεκτήματα της συγκεκριμένης αρχιτεκτονικής. Για την αντιμετώπιση της έμφυτης αδυναμίας που εισάγει η configuration μνήμη μιας FPGA τα τελευταία χρόνια έχουν δημιουργηθεί σειρές FPGA οι οποίες προσφέρουν επιπλέον αντοχή σε ακτινοβολία όπως η σειρά QPro της Xilinx. Οι FPGAs οι οποίες ανήκουν στον παραπάνω τύπο έχουν επιλεγεί για την ανάπτυξη εφαρμογών αντοχής σφαλμάτων έναντι των ASICs όπως σε αποστολές διαστήματος όπως η MARS2003 Lander, η MARS2003 Rover και άλλες. Επιπλέον οι παραπάνω FPGAs έχουν χρησιμοποιηθεί σε στρατιωτικές εφαρμογές και σε τηλεπικοινωνιακά συστήματα [4]. Οι παραπάνω εξελίξεις καθιστούν τον τομέα έρευνας για την ανάπτυξη εφαρμογών fault tolerant και failure safe με την χρήση FPGAs ιδιαίτερα ενδιαφέρον ο οποίος αν και σχετικά νέος αναπτύσσεται ταχύτατα.

2 Σχετικές Εργασίες Δυναμικής Αναδιάταξης και Ανάπτυξης Εφαρμογών Αντοχής Σφαλμάτων με Χρήση FPGAs

2.1 Δυναμική Αναδιάταξη

Ο Ανυφαντής[5] προχώρησε σε πειραματική ανάλυση του χρόνου που απαιτείται για την εφαρμογή της μερικής αναδιάταξης στην Virtex-II Pro FPGA. Με χρήση της Difference-Based ροής εξήγαγε τον απαιτούμενο χρόνο για την αναδιάταξη ενός frame.

Ο Ευφραιμίδης[6] προχώρησε στην δημιουργία ενός αυτόνομου, μερικά αναδιατασσόμενου, συστήματος γενετικού αλγόριθμου, στο οποίο υποστηρίζεται η αλλαγή της fitness function του συστήματος σε χρόνο εκτέλεσης. Το σύστημα μέσω της αναδιάταξης μπορεί να υποστηρίξει θεωρητικά απεριόριστο αριθμό από fitness functions.

Ο Νικολουδάκης[7] προχώρησε στην δημιουργία και αξιολόγηση ενός ενσωματωμένου συστήματος κρυπτογραφίας όπου ο κρυπτογραφικός αλγόριθμος του συστήματος μπορεί να εναλλαχθεί μέσω δυναμικής αναδιάταξης.

Ο Σπανάκης[8] προχώρησε στην σχεδίαση και υλοποίηση ενός Linux-based, Task manager δυναμικής αναδιάταξης ο οποίος αναλαμβάνει την αναδιάταξη των PRRs. Εκτος των άλλων πλεονεκτημάτων της ύπαρξης ένος λειτουργικού συστήματος η ύπαρξη της εξωτερικής DDR μνήμης η οποία χρησιμοποιείται και ως cache από το λειτουργικό μειώνει αρκετά τον απαιτούμενο χρόνο αναδιάταξης.

Οι Παπαδημητρίου και Δόλλας[9] δημιούργησαν μία μέθοδο προφόρτωσης μονάδων σε μία σχεδίαση όπου όλες οι μονάδες του συστήματος δεν χωράνε σε μία FPGA. Μεταξύ των μονάδων που προφορτώνονται είναι δυνατή η χρονική πολυπλεξία έτσι προφορτώνοντας μέρος μίας μονάδας η οποία θα χρησιμοποιηθεί στο μέλλον και το υπόλοιπο τμήμα της όταν είναι απαραίτητη για την λειτουργία του συστήματος μειώνεται η επίδραση του χρόνου αναδιάταξης στην λειτουργία του συστήματος.

Οι Παπαδημητρίου et al [10] παρουσίασαν μία λεπτομερέστατη πειραματική μέτρηση του χρόνου της δυναμικής αναδιάταξης ενός συστήματος το οποίο έχει αποθηκευμένα τα partial bitstreams βρίσκονται σε μία compact flash και η αναδιάταξη γίνεται μέσω του ICAP.

Οι Claus et al[11] προχώρησαν στην δημιουργία ενός PLB-ICAP ο οποίος με την χρήση DMA κατάφεραν να μεταφέρουν γρηγορότερα τα δεδομένα στην CF και σε συνδυασμό με την συνδιαστική χρήση ροών όπως η Difference-based, η PartialMask και η Module-based μείωσαν το μέγεθος των partial bitstreams και ακολούθως, επιπλέον το χρόνο της μερικής αναδιάταξης της FPGA.

2.2 Αντιμετώπιση Σφαλμάτων σε FPGAs

Οι μέθοδοι οι οποίες έχουν αναπτυχθεί για αντιμετώπιση σφαλμάτων σε FPGAs γενικά μπορούν να κατηγοριοποιηθούν σε δυο κατηγορίες ανάλογα στο επίπεδο στο οποίο αντιμετωπίζουν τα σφάλματα. Στην μια κατηγορία ανήκουν αυτές που αντιμετωπίζουν τα σφάλματα σε επίπεδο σχεδίασης της FPGA αλλάζοντας την αρχιτεκτονική της συσκευής (device- level) και στην δεύτερη ανήκουν αυτές που αντιμετωπίζουν τα σφάλματα σε επίπεδο διάταξης της FPGA (configuration – level) .

2.2.1 Ανοχή σφαλμάτων σε επίπεδο συσκευής (Device Level)

Γενικά οι εργασίες οι οποίες προτείνουν ανοχή σφαλμάτων σε επίπεδο συσκευής βασίζονται στην εισαγωγή επιπλέον στηλών αναδιατασσόμενης λογικής και επιπλέον πόρων διασύνδεσης ώστε να παρακάμπτονται τα ελαττωματικά στοιχεία της FPGA .

Η μέθοδος η οποία προτείνεται από τους Hatori et al [12] χρησιμοποιεί επιπλέον κωδικοποιητές σειρών και λογική επιλογής σειρών και σε περίπτωση που κάποια σειρά περιέχει κάποιο σφάλμα τότε παρακάμπτεται μέσω των κωδικοποιητών και της λογικής επιλογής.

Η μέθοδος η οποία προτείνεται από τους Kelly και Ivey [13] χρησιμοποιεί επιπλέον ένα ειδικό On-chip κύκλωμα για να εκτελεί αναδιάταξη του routing. Όταν εντοπιστεί κάποιο σφάλμα σε κάποια σειρά τότε η λογική η οποία περιέχεται στο ελαττωματικό PLB καθώς και όλων όσων βρίσκονται δεξιά του ολισθαίνουν δεξιά κατά μια θέση και οι αλλαγές που χρειάζονται στις διασυνδέσεις των PLBs υλοποιούνται από το ειδικό κύκλωμα παρακάμπτοντας το ελαττωματικό PLB.

Η μέθοδος η οποία προτείνεται Durand και Piguet [14] χρησιμοποιεί κάποιες εφεδρικές στήλες και μια αρχιτεκτονική για τα PLB έτσι ώστε να είναι ικανά να αυτοελέχονται. Επίσης πόροι διασύνδεσης υπάρχουν μονό μεταξύ γειτονικών PLBs. Όταν κάποιο PLB βρεθεί ότι έχει κάποιο σφάλμα τότε η στήλη στην οποία ανήκει ολισθαίνει μια θέση δεξιά .

Η μέθοδος η οποία προτείνεται από τους Howard et al [15] συγκεντρώνει τα PLBs σε γκρουπ. Έτσι ώστε εάν συμβεί κάποιο σφάλμα σε κάποιο γκρουπ τότε η λειτουργικότητα του να μεταφέρεται σε κάποιο εφεδρικό. Ο κύριος σκοπός αυτής της μεθόδου είναι να ελαττωθεί η καθυστέρηση η οποία προκύπτει από την μεταφορά μέρος του κυκλώματος από μια ελαττωματική τοποθεσία σε μια εφεδρική με το να μεταφέρουμε μεγαλύτερα μέρη του

κυκλώματος αλλάζοντας έτσι λιγότερο την αρχική δομή του κυκλώματος. Επίσης προτείνεται και η ύπαρξη εφεδρικών PLB μέσα στα γκρουπ αντί για ολόκληρα εφεδρικά γκρουπ.

Οι μέθοδοι οι οποίες προτείνονται από τους Hanchek και Dutt [16] αντιμετωπίζουν σφάλματα στην διασύνδεση των PLBs. Στην πρώτη σε κάθε κανάλι διασυνδέσεις προστίθεται μια εφεδρική σύνδεση έτσι ώστε σε περίπτωση σφάλματος να χρησιμοποιηθεί. Πλεονέκτημα της μεθόδου αυτής είναι ότι δεν εισάγει επιπλέον καθυστέρηση στο κύκλωμα. Η δεύτερη μέθοδος χρησιμοποιεί ένα εφεδρικό δίκτυο στην FPGA. Η μέθοδος αυτή δίνει την δυνατότητα για αντιμετώπιση περισσότερων σφαλμάτων αλλά μπορεί να εισάγει επιπλέον καθυστέρηση .

Η μέθοδος η οποία προτείνεται από τους Doumar και Ito [17] προτείνει την διαμόρφωση της configuration μνήμης μιας κανονικής FPGA έτσι ώστε να υλοποιεί ένα shift reregister. Επιπλέον εισάγονται πολυπλέκτες μεταξύ των στοιχείων της μνήμης έτσι ώστε να μπορέσουν να ολισθήσουν κάθετα η οριζόντια. Σε περίπτωση σφάλματος το κύκλωμα ολισθαίνει για την αποφυγή του ελαττωματικού πόρου. Η καθυστέρηση που επιφέρει η μέθοδος αυτή είναι αρκετά μικρή γιατί το κύκλωμα μεταφέρεται ολόκληρο.

Η μέθοδος η οποία προτείνεται από τον Κυριακουλάκο [18] προτείνει σχεδιαστικές αλλαγές στην αρχιτεκτονική των slices της Virtex – 5 FPGA για την αντιμετώπιση soft σφαλμάτων με την υλοποίηση της τεχνικής του TMR σε επίπεδο LUTs. Προτείνεται τα δύο LUTs ενός slice και ένα LUT από το επόμενο η το προηγούμενο slice να υλοποιούν την ίδια λογική συνάρτηση και με την εισαγωγή πυλών XOR στις εξόδους να εντοπίζονται τυχών σφάλματα .Η συγκεκριμένη μέθοδος λόγω της ιδιαιτερότητας των slices της συγκεκριμένης οικογένειας FPGAs απαιτεί λιγότερους πόρους για την υλοποίηση του TMR.

2.2.2 Αντιμετώπιση Σφαλμάτων σε Επίπεδο Διάταξης της FPGA (configuration level)

Οι εργασίες οι οποίες προτείνουν ανοχή σφαλμάτων σε επίπεδο διάταξης της FPGA βασίζονται στην τροποποίηση της σχεδίασης της αρχικής εφαρμογής ούτως ώστε να υλοποιηθεί κάποια μέθοδος ανίχνευσης σφαλμάτων και στην μερική αναδιάταξη για την διόρθωση του σφάλματος.

2.2.2.1 Μέθοδοι Ανίχνευσης Σφαλμάτων

Για τον εντοπισμό των σφαλμάτων τα οποία προκύπτουν οι μέθοδοι οι οποίες έχουν αναπτυχθεί χωρίζονται γενικά σε πέντε κατηγορίες. Σε αυτές που βασίζονται σε πλεονασμό μονάδων (modular redundancy) ,σε αυτές που χρησιμοποιούν κάποια μορφή concurrent error detection , σε αυτές που χρησιμοποιούν δομές BIST για τον off line έλεγχο ενός κυκλώματος,

σε αυτές που προοδευτικά ελέγχουν την FPGA (roving) και σε αυτές που διαβάζουν τα δεδομένα της configuration memory και ελέγχουν εάν έχουν αλλοιωθεί.

Η ανίχνευση με την χρησιμοποίηση επιπλέον μονάδων προσφέρει ανίχνευση κάθε σφάλματος το οποίο μπορεί να συμβεί σε μία μονάδα, εφόσον εκδηλωθεί, γρήγορο εντοπισμό του σφάλματος κατά την διάρκεια λειτουργία του συστήματος καθώς επίσης και εντοπισμό σφαλμάτων τα οποία προκύπτουν στα δεδομένα του κυκλώματος. Το μειονέκτημα των μεθόδων αυτών είναι το μεγάλο κόστος υλοποίησης.

Η ανίχνευση με την χρησιμοποίηση concurrent error detection μεθόδων προσφέρει και αυτή τα πλεονεκτήματα της modular redundancy χρησιμοποιεί λιγότερους πόρους αλλά εντοπίζει λιγότερο ποσοστό σφαλμάτων.

Η ανίχνευση με τεχνικές BIST έχει το πλεονέκτημα ότι δεν παρεμβαίνει στην κανονική λειτουργία του συστήματος. Προϋπόθεση για την χρησιμοποίηση των μεθόδων αυτών είναι να υπάρχουν διαστήματα όπου το σύστημα να είναι ανενεργό έτσι ώστε να μπορεί να ενεργοποιηθεί ο έλεγχος.

Η ανίχνευση με την μέθοδο του Roving επιτυγχάνει γρηγορότερη ανίχνευση ενός σφάλματος από αυτό των BIST και επιπλέον έχει ελάχιστο κόστος σε πόρους. Μεγάλο μειονέκτημα της μεθόδου είναι οι επιπτώσεις στην λειτουργία του συστήματος.

Η ανίχνευση σφαλμάτων με τη μέθοδο Read Back προσφέρει εντοπισμό των αλλοιώσεων των δεδομένων της configuration memory συγκρίνοντας τα δεδομένα της η κάποιο κώδικα που παράγεται από αυτά (πχ CRC) με τα αντίστοιχα της αρχικής σχεδίασης. Για την εφαρμογή αυτής της μεθόδου πρέπει να έχει αποθηκευτή η αρχική διάταξη της FPGA. Η μέθοδος αυτή δεν μπορεί να ανιχνεύσει σφάλματα στα δεδομένα που παράγονται κατά την διάρκεια λειτουργίας του συστήματος και αποθηκεύονται σε στοιχεία μνήμης όπως καταχωρητές η BRAMs.

Στον Πίνακα 2.1 αναφέρουμε τις συνηθέστερες μεθόδους εντοπισμού σφαλμάτων οι οποίες χρησιμοποιούνται σε FPGAs καθώς και μερικά από τα πλεονεκτήματα και μειονεκτήματα τους.

Μέθοδος	**Ταχύτητα ανίχνευσης**	**Κόστος σε πόρους**	**Επίδραση στην συχνότητα λειτουργίας**
Modular redundancy	Γρήγορη	Αρκετά μεγάλο	Πολύ μικρό
Concurrent error detection	Γρήγορη	Μεσαίο	Μικρό
Off-line BIST	Αργή	Πολύ μικρό	Μικρό

Roving	Μεσαία	Μεσαίο	Μεγάλο
Read Back	Αργή	Πολύ μικρό	Μικρό

Πίνακας 2.1 Σύγκριση Μεθόδων Ανίχνευσης Σφαλμάτων

Παρακάτω αναφέρουμε ορισμένες εργασίες οι οποίες κατά κύριο λόγο ασχολούνται με τον εντοπισμό των σφαλμάτων τα οποία μπορούν να προκύψουν θεωρώντας δεδομένο την διόρθωση τους ή αναφέρουν κάποια μέθοδο με την οποία θα μπορούσε να συνδυαστεί η τεχνική που ανάπτυξαν ή χρησιμοποιώντας απλή αναδιάταξη

Η μέθοδος η οποία δημιουργήθηκε από τους Emmert et al [19] βασίζεται στην δυνατότητα αναδιάταξης ώστε να είναι δυνατόν να ελέγχεται on-line περιοδικά ολόκληρη η FPGA για hard faults με ελάχιστο κόστος σε πόρους υλικού. Σε αυτήν την εργασία χρησιμοποιείται μια οριζόντια και μια κάθετη στήλη πλάτους δυο CLBs αμφότερες (roving stars) οι οποίες χρησιμοποιούνται για την υλοποίηση BIST τεχνικών στους πόρους που καλύπτονται από τις δύο στήλες. Όταν ολοκληρωθεί ο έλεγχος μιας περιοχής τότε αυτή αποδεσμεύεται δίνεται πίσω στο σύστημα και δεσμεύεται μια γειτονική περιοχή και οι δυο στήλες μετακινούνται σε αυτή ώστε να την ελέγξουν. Μειονέκτημα της μεθόδου είναι ότι παρεμβαίνει στην λειτουργία της όποιας εφαρμογής υλοποιείται, μετατοπίζοντας πόρους της, με αποτέλεσμα τόσο την μείωση της συχνότητας της όσο και την αναγκαστική παύση της λειτουργίας της κατά την διαδικασία μετακίνησης των στηλών σε μέρος όπου χρησιμοποιείται από την εφαρμογή ώστε να γίνει η ανταλλαγή πόρων.

Η μέθοδος η οποία δημιουργήθηκε από τους Das και Touba [20] εντοπίζει σφάλματα όπως bridging faults, stuck-at-0 και stuck -at-1 στις διασυνδέσεις μιας FPGA. Η μέθοδος αυτή δημιουργεί διατάξεις αλλάζοντας μονό τις λογικές συναρτήσεις των CLBs της αρχικής σχεδίασης έτσι ώστε αυτά να λειτουργούν ως δρομολογητές σημάτων σχηματίζοντας εικονικά scan paths. Με τον τρόπο αυτό εξετάζονται μόνο οι πόροι που χρησιμοποιούνται από την αρχική διάταξη. Ένα CLB ανήκει σε ένα μόνο scan path. Για τον εντοπισμό stuck-at-0 και wired-OR σφαλμάτων τα CLBs διατάσσονται έτσι ώστε η έξοδος τους να είναι 0 μόνον όταν μια είσοδος τους έχει τιμή 0 ενώ για τον εντοπισμό stuck-at-1 και wired-AND σφαλμάτων τα CLBs διατάσσονται έτσι ώστε η έξοδος τους να είναι 1 μόνον όταν μια είσοδος τους έχει τιμή 1.

Η μέθοδος η οποία δημιουργήθηκε από τους Metra et al[21] εντοπίζει σφάλματα όπως stuck-at (SAs), transistors stuck-on (SONs), transistors stuck open (SOPs), και bridgings (BFs) faults στα CLBs μιας FPGA της σειράς XILINX XC4000. Τα LUT ενός CLB διατάσσονται έτσι ώστε να υλοποιούν τις συναρτήσεις NOT , XOR και Transparent. Επίσης τα CLBs διατάσσονται σε σειρές/στήλες και οι έξοδοι του προηγούμενου αποτελούν εισόδους του επομένου. Οι έξοδοι κάθε στήλης CLB χρησιμοποιούνται για τον έλεγχο. Για

τον διεξοδικό έλεγχο των CLB χρησιμοποιούνται 7 συνολικά configurations οπού 4 είσοδοι ενός CLB δρομολογούνται στις εξόδους από ανεξάρτητα μονοπάτια καθώς επίσης και 16 test vectors. Τέλος για την εισαγωγή και την εξαγωγή των αποτελεσμάτων χρησιμοποιείται η μονάδα Boundary Scan της FPGA .

Οι Andraka και Brady [22] εφάρμοσαν την τεχνική BIST (Bult –In Self Test) για την εγγύηση της λειτουργίας ενός κυκλώματος υπολογισμού μετασχηματισμού Fourier. Για την υλοποίηση αυτού του ελέγχου χρησιμοποιήθηκε ένας LFSR shift register για την παραγωγή των test vectors και CRC κώδικες που πραγματοποιούν signature analysis για τον έλεγχο της ορθότητας του συστήματος. Η εφαρμογή στην οποία εφαρμόζεται αυτή η τεχνική είναι ένας 4K block floating point FFT ο οποίος λόγω του μεγάλου κυκλώματος υλοποίησης και τον περιορισμό της χρονικής απόκρισης μέθοδοι όπως modular redundancy ήταν ανέφικτες. Τέλος επειδή οι κώδικες CRC δεν ελέγχουν ολόκληρη την σχεδίαση για ένα μικρό μέρος αυτής χρησιμοποιείται η τεχνική read back της configuration memory .

Η μέθοδος η οποία μελετάτε από τους Bolchini et al [23] είναι η τεχνική του TMR όπου χρησιμοποιείται ένας majority voter και τρεις replicas για την εξασφάλιση της σωστής λειτουργίας του συστήματος. Η τεχνική αυτή μπορεί να εφαρμοστεί σε πολλά επίπεδα αφαίρεσης από ολόκληρο το σύστημα έως τα στοιχειώδη δομικά συστατικά του συστήματος προσφέροντας έτσι εναλλακτικά οφέλη με κάθε υλοποίηση. Επιπλέον σε αυτήν την εργασία εξετάζονται και διαφορετικές τοποθετήσεις των τριπλών μονάδων σε ιδία η διαφορετικές PRR και για κάθε τοποθέτηση αναφέρονται τα πλεονεκτήματα και τα αντίστοιχα μειονεκτήματα. Γενικά μεγάλο πλεονέκτημα της μεθόδου είναι ότι είναι από της πιο διαδεδομένες τεχνικές όχι μονό σε FPGAs και έχει γίνει αρκετά μεγάλη έρευνα πάνω σε αυτή. Το κυριότερο μειονέκτημα της είναι το επιπλέον κόστος σε υλικό που απαιτείται για την υλοποίηση της. Η μέθοδος του TMR έχει υλοποιηθεί και από την Xilinx[24] .

Οι Bolchini et al [25] μελετούν τη τεχνική του duplication με σύγκριση (DWC) σε αναδιατασσόμενη λογική οπού χρησιμοποιούνται δυο αντίγραφα της εφαρμογής των οποίων οι έξοδοι συγκρίνονται για την ανίχνευση τυχόν σφαλμάτων. Η μέθοδος DWC χρησιμοποιείται γιατί προσφέρει όλα τα πλεονεκτήματα του TMR όσον αφορά την ανίχνευση σφάλματος με λιγότερες απαιτήσεις σε πόρους. Το πλεονέκτημα του TMR έναντι του DWC είναι η δυνατότητα του σε περιπτώσεις σφάλματος να μπορεί να το αποκρύψει και να παράγει σωστά αποτελέσματα .

Για τον εντοπισμό σφαλμάτων τα οποία προκλήθηκαν από SEU οι Gokhale et al [26] χρησιμοποιούν την τεχνική του Read Back. Κάθε frame της configuration memory διαβάζεται από την FPGA και υπολογίζεται ένας κώδικας CRC. Ο κώδικας CRC ο οποίος υπολογίστηκε έπειτα συγκρίνεται με ένα αποθηκευμένο κώδικα CRC για το συγκεκριμένο frame. Κάθε configuration διαβάζεται κάθε 180ms. Για τον σωστό εντοπισμό λαθών με τη μέθοδο αυτή δεν θα πρέπει να χρησιμοποιούνται LTU σαν RAMs η σαν shift-registers. Ένα

άλλο πρόβλημα με την μέθοδο αυτή είναι ότι τα σφάλματα δεν εντοπίζονται απευθείας καθώς και ότι σφάλματα τα οποία προκύπτουν στα δεδομένα δεν εντοπίζονται. Ένα από τα πλεονεκτήματα τις μεθόδου αυτής είναι ότι δεν χρησιμοποιείται περιττό hardware(χρήση επιπλέον μονάδων) για την ανίχνευση των λαθών. Την ίδια τεχνική χρησιμοποιούν και οι Asadi και Tahoori όπου τα δύο τελευταία bytes κάθε frame χρησιμοποιούνται για την αποθήκευση του κώδικα CRC του συγκεκριμένου frame [27]. Για την επιτυχή χρήση της μεθόδου Read Back και την σύγκριση των κωδίκων CRC θα πρέπει ο έλεγχος να γίνεται σε τακτά χρονικά διαστήματα ώστε να αποφεύγεται η συσσώρευση σφαλμάτων με ενδεχόμενη αποτυχία του ελέγχου.

Οι Fernanda Lima et al[28] συνδυάζουν την μέθοδο του duplication με την μέθοδο του time redundancy ώστε σε περίπτωση ασυμφωνίας των δύο αντιγράφων τα δεδομένα εισόδου κωδικοποιούνται και ξαναεπεξεργάζονται, και στα δύο αντίγραφα, και το αποτέλεσμα κάθε αντιγράφου αποκωδικοποιείται και έπειτα συγκρίνεται με το αντίστοιχο προηγούμενο ώστε να βρεθεί ποία μονάδα παρήγαγε λανθασμένα αποτελέσματα.

Οι Armin Alaghi et al [29] παρουσιάζουν μια BIST μέθοδο για τον έλεγχο των LUTs. Η μέθοδος δημιουργήθηκε για LUTs τεσσάρων εισόδων. Τα LUTs διατάσσονται σε αλυσίδες όπου τροφοδοτούνται από έναν TPG ο οποίος παράγει διευθύνσεις για τα LUTs. Για τον συνολικό έλεγχο των LUTs χρησιμοποιήθηκαν συνολικά 8 διατάξεις. Η συγκεκριμένη μέθοδος μπορεί να εντοπίσει σφάλαματα επίσης και στον TPG όχι όμως στον ORA.

Οι Michel Renovell et al [30] δημιούργησαν μια μέθοδο για τον έλεγχο των πόρων διασύνδεσης των FPGAs. Συγκεκριμένα δημιούργησαν σχεδιάσεις οι οποίες διατάσσουν τα connection points των πινάκων δρομολόγησης της FPGA ώστε να δημιουργήσουν εικονικούς διαύλους τους οποίους ελέγχουν για τυχόν βραχυκυκλώματα και ανοιχτοκυκλώματα μεταξύ των γραμμών τους. Η μεθοδός τους μπορεί να εντοπίσει σφάλματα μέσα στους πίνακες δρομολόγησης όσο και έκτος αυτών στις γραμμές δρομολόγησης.

2.2.2.2 Μέθοδοι Αντιμετώπισης Σφαλμάτων

Για την αντιμετώπιση των σφαλμάτων τα οποία προκύπτουν οι μέθοδοι μερικής αναδιάταξης οι οποίες έχουν αναπτυχθεί χωρίζονται γενικά σε τρεις κατηγορίες. Σε αυτές που προγραμματίζουν την FPGA με ακριβώς την ιδία σχεδίαση, σε αυτές που προγραμματίζουν την FPGA με προκαθορισμένες εναλλακτικές, ήδη υλοποιημένες, σχεδιάσεις και σε αυτές που τροποποιούν την υπάρχουσα σχεδίαση στην περίπτωση σφάλματος έτσι ώστε να παρακαμφτεί το σφάλμα (αυξητικές σχεδιάσεις).

Η αναδιάταξη με ακριβώς την ιδία υλοποίηση σχεδίαση βασίζεται στο γεγονός ότι τα συχνότερα σφάλματα τα οποία προκύπτουν είναι αντιστροφές των bit της

configuration μνήμης και επομένως μπορούν να αντιμετωπιστούν με επανεγραφή της με τα σωστά δεδομένα. Σε περίπτωση που έχει συμβεί κάποιο hard σφάλμα τότε αυτό δεν μπορεί να αντιμετωπιστεί με την συγκεκριμένη μέθοδο.

Η αναδιάταξη με εναλλακτικές σχεδιάσεις προσφέρει αντιμετώπιση hard σφαλμάτων προγραμματίζοντας την FPGA με κάποια υλοποίηση η οποία δεν χρησιμοποιεί τους κατεστραμμένους πόρους μιας συσκευής. Ο αριθμός των εναλλακτικών υλοποιήσεων καθορίζεται από το επιθυμητό ποσοστό αντιμετώπισης hard σφαλμάτων και από το επίπεδο της στοιχειώδους μονάδας στην οποία δημιουργούνται διαφορές μεταξύ των σχεδιάσεων (πχ στήλες CLBs). Έτσι για να επιτευχθεί υψηλό ποσοστό αντιμετώπισης είναι απαραίτητο να υλοποιηθούν αρκετές εναλλακτικές σχεδιάσεις με αποτέλεσμα στην αύξηση του χώρου αποθήκευσης και του απαιτούμενου χρόνου υλοποίησης.

Οι αυξητικές σχεδιάσεις προσπαθούν αντιμετωπίσουν τα σφάλματα στο επίπεδο του υλικό με χρήση διαθέσιμων πόρων οι οποίοι δεν χρησιμοποιούνται στην αρχική σχεδίαση. Γενικά η μέθοδος αυτή εφόσον υπάρχουν οι απαραίτητοι πόροι μπορούν να αντιμετωπίσουν οποιοδήποτε σφάλμα προκύψει. Επειδή δεν υπάρχουν προυλοποιημένες εναλλακτικές σχεδιάσεις η παραγωγή μιας νέας σχεδίασης μέσω τροποποίησης της αρχικής μπορεί να διαρκέσει αρκετά μεγάλο χρονικό διάστημα κάτι το οποίο έχει επιπτώσεις στον χρόνο αποκατάστασης του σφάλματος.

Στον Πίνακα 2.2 αναφέρουμε τις συνηθέστερες μεθόδους αντιμετώπισης σφαλμάτων μέσω μερικής αναδιάταξης, οι οποίες χρησιμοποιούνται σε FPGAs καθώς και μερικά από τα πλεονεκτήματα και μειονεκτήματα τους.

Μέθοδος	**Αντιμετώπιση hard faults**	**Αντιμετώπιση soft faults**	**Χώρος αποθή-κευσης**	**Χρόνος παραγωγής**	**Χρόνος αποκατά-στασης**
Απαράλλακτη σχεδίαση	Όχι	Ναι	Μικρός	Μέτριος	Τάξεως milliseconds
Εναλλακτικές σχεδιάσεις	Ναι	Ναι	Μεγάλος	Μεγάλος	Τάξεως milliseconds
Αυξητική σχεδίαση	Ναι	Ναι	Μικρός	Μέτριος	Αρκετά μεγάλος

Πίνακας 2.2 Σύγκριση μεθόδων αντιμετώπισης σφαλμάτων

Παρακάτω αναφέρουμε ορισμένες εργασίες οι οποίες κατά κύριο λόγω ασχολούνται με την αντιμετώπιση των σφαλμάτων τα οποία μπορούν να προκύψουν θεωρώντας δεδομένο τον εντοπισμό τους η απλά αναφέρουν κάποια μέθοδο με την οποία θα μπορούσε να συνδυαστεί η τεχνική που ανάπτυξαν .

Η μέθοδος η οποία δημιουργήθηκε από τους Huang και McCluskey [31] αντιμετωπίζει τα σφάλματα (σε περίπτωση hard faults) με εναλλακτικές διατάξεις έτσι ώστε

να αποφεύγεται η χρησιμοποίηση της στήλης η οποία περιέχει κάποιο χαλασμένο CLB. Όταν εντοπιστεί κάποιο hard fault τότε χρησιμοποιείται μια Precompiled Configuration (Precompiled για να γίνεται ταχυτέρα η αναδιάταξη) η οποία δεν χρησιμοποιεί την παραπάνω στήλη. Η μέθοδος αυτή παρουσιάζει δυο παραλλαγές την Overlapping και την Non-overlapping. Στην Overlapping παραλλαγή διατηρούμε κάποιες στήλες αχρησιμοποίητες και όταν συμβεί κάποιο permanent σφάλμα τις χρησιμοποιούμε. Η Non-overlapping παραλλαγή χρησιμοποιείται εάν το κύκλωμα μπορεί να χωρέσει στην μισή FPGA και οι εναλλακτικές configuration μετατοπίζουν ολόκληρο το κύκλωμα σε αχρησιμοποίητες περιοχές τις FPGA. Όπως είναι φανερό αυτή η μέθοδος απαιτεί αρκετά περισσότερο αποθηκευτικό χώρο για την αποθήκευση των επιπλέων διατάξεων. Για τον περιορισμό του χώρου αποθήκευσης οι εναλλακτικές διατάξεις συμπιέζονται με τη χρήση κωδίκων Golomb.

Η μέθοδος η οποία δημιουργήθηκε από τους Lach et al [32] προτείνουν τον διαχωρισμό της φυσικής σχεδίασης σε ένα σύνολο από ψηφίδες οι οποίες θα καταλαμβάνουν την ίδια ποσότητα πόρων. Κάθε ψηφίδα θα υλοποιεί κάποιο μέρος της λειτουργίας του συστήματος και για κάθε ψηφίδα υλοποιούνται εναλλακτικές διατάξεις οι οποίες χρησιμοποιούν διαφορετικούς πόρους της ψηφίδας αλλά το ίδιο interface με τις γειτονικές ψηφίδες έτσι ώστε να μπορούν να αντιμετωπίσουν σφάλματα που μπορούν να προκύψουν στο εσωτερικό της ψηφίδας. Όσο μεγαλύτερη είναι η ψηφίδα τόσο περισσότερα σφάλματα στους πόρους της FPGA μπορούν να αντιμετωπιστούν. Για την υλοποίηση της μεθόδου αυτής απαιτείται επιπλέον αποθηκευτικός χώρος για τις εναλλακτικές διατάξεις. Επίσης η μέθοδος αυτή υλοποιείται ευκολότερα σε FPGA οι οποίες παρουσιάζουν κάποια ιεραρχία στην αρχιτεκτονική τους.

Στην εργασία των Emmert και Bhatia [33] μελετάται ο τρόπος με τον οποίο ένα σύνολο CLBs η λειτουργία των οποίων πρέπει να μεταφερθεί αντικαθίστανται από ισάριθμα εφεδρικά. Ο εντοπισμός των χαλασμένων CLB θεωρείται δεδομένος. Η μεθοδολογία η οποία αναπτύσσεται ασχολείται με την καταλληλότερη δέσμευση των εφεδρικών και την τροποποίηση του κυκλώματος ώστε να χρησιμοποιηθούν τα εφεδρικά CLBs. Η επιλογή του καταλλήλου εφεδρικού CLB γίνεται με την χρήση του αλγορίθμου Minimax Grid Matching και η τροποποίηση του κυκλώματος γίνεται με ολίσθηση των CLBs που βρίσκονται μεταξύ του χαλασμένου (συμπεριλαμβανομένου του ελλατωματικού CLB) και του εφεδρικού προς το εφεδρικό CLB ούτως ώστε να έχουμε τις λιγότερες αλλαγές στην δρομολόγηση των σημάτων του κυκλώματος.

Οι Emmert και Bhatia [34] αναπτύσσουν μια μέθοδος για την εύρεση των συντομότερων μονοπατιών δρομολόγησης μεταξύ των CLBs τα οποία πρέπει να μεταφερθούν με αυτά που διασυνδέονται και μένουν στην θέση τους. Η μέθοδος αρχικά βρίσκει όλα αυτά τα ζευγάρια των CLBs και καταργεί την διασύνδεση μεταξύ τους (καθώς και των δικτύων τα οποία χρησιμοποιούν μια διασύνδεση προς κατάργηση) και στην

συνέχεια εκτελώντας έναν αυξητικό αναδρομικό άπληστο αλγόριθμο βρίσκει σύντομα μονοπάτια για την επανασύνδεση των CLBs. Ο αλγόριθμος αυτός ορίζει ένα μέγιστο μήκος μονοπατιού και σε περίπτωση που δεν βρεθεί κάποιο μονοπάτι το αυξάνει κατά ένα. Επίσης τα μακρύτερα μονοπάτια εξετάζονται πρώτα. Το ίδιο πρόβλημα μελετάται και από τους Dutt et al [35]. Μια διαφορά της τεχνικής που αναπτύσσεται εδώ είναι ότι ο αλγόριθμος που αναπτύχθηκε για την διασύνδεση δύο CLB μπορεί και χρησιμοποιήσει ήδη δεσμευμένους πόρους διασύνδεσης αποδεσμεύοντας τους κάτι το οποίο δημιουργεί επιπλέον ζεύγη CLBs τα οποία στην συνέχεια θα πρέπει να συνδεθούν.

2.2.2.3 Εγγύηση Λειτουργίας των Μονάδων Αναδιάταξης

Για την αποφυγή των σφαλμάτων τα οποία προκύπτουν και εντοπίζονται είναι απαραίτητη η ορθή αναδιάταξη της FPGA με τις εναλλακτικές σχεδιάσεις οι οποίες αντιμετωπίζουν το σφάλμα το οποίο προέκυψε. Επιπλέον αρκετές μέθοδοι testing οι οποίες αναπτύχθηκαν για τον εντοπισμό σφαλμάτων σε FPGAs βασίζονται στην δυνατότητα της μερικής αναδιάταξης ούτως ώστε να “κατεβούν ’’ οι κατάλληλες διατάξεις οι οποίες θα ελέγξουν την FPGA ή ώστε να μεταφέρουν τον έλεγχο σε κάποια άλλη περιοχή της συσκευής. Επιπλέον τα test τα οποία παράγονται για την ανίχνευση σφαλμάτων για την αποφυγή ανάπτυξης μεγάλων ακολουθιών test vectors οι οποίες θα απαιτούσαν μεγάλο αποθηκευτικό χώρο, χρόνο παραγωγής και θα οδηγούσαν σε χρονοβόρες διαδικασίες ελέγχου καθώς και οι μέθοδοι πλειοψηφίας (TMR) βασίζονται στην υπόθεση ότι κάθε χρονική στιγμή είναι δυνατόν να υπάρχει το πολύ ένα σφάλμα στο κύκλωμα μία υπόθεση η οποία είναι γενικά σωστή υπό την προϋπόθεση ότι τα σφάλματα τα οποία εντοπίζονται διορθώνονται ώστε να αποφεύγεται η συσσώρευση τους.

Από τα παραπάνω είναι φανερό ότι για την αξιόπιστη υλοποίηση fault tolerant εφαρμογών σε SRAM FPGAs είναι αναγκαίο να διασφαλιστεί η ορθή λειτουργία των μονάδων οι οποίες εκτελούν την μερική αναδιάταξη.

Οι περισσότερες εργασίες τις οποίες αναφέραμε στις δύο προηγούμενες ενότητες δεν αναφέρονται σε αυτό το θέμα ή θεωρούν ότι υπάρχει ένα αξιόπιστο σύστημα το οποίο αναλαμβάνει να εκτελέσει την λειτουργία της αναδιάταξης αξιόπιστα. Οι εργασίες που αναφέρονται στην ικανοποίηση των παραπάνω απαιτήσεων είναι οι [26] και [27]. Στην πρώτη την λειτουργία της αναδιάταξης αναλαμβάνει ένας radiation-hardened μικροεπεξεργαστής , ο 30-MHz RAD6000, ο οποίος σε περίπτωση σφάλματος διακόπτεται και αναδιατάσσει το frame της configuration memory όπου εντοπίστηκε το σφάλμα. Οι διατάξεις ελέγχου της FPGA υλοποιούνται σε μια βοηθητική Actel radiation-hardened anti-

fuse FPGA η οποία ελέγχει τα frames της FPGA και σε περίπτωση ανίχνευσης σφάλματος ειδοποιεί τον μικροεπεξεργαστή για να αναδιατάξει το συγκεκριμένο frame. Στην δεύτερη εργασία χρησιμοποιείται μια βοηθητική anti-fuse FPGA η AX125 στην οποία υλοποιούνται με την μέθοδο του TMR τα κυκλώματα τα οποία απαιτούνται για τον εντοπισμό και την έναρξη της αναδιάταξης της κύριας FPGA.

3 Υλοποίηση ενός FIR φίλτρου για την Εξαγωγή Πειραμάτων

Στο κεφάλαιο αυτό παρουσιάζουμε την υλοποίηση της εφαρμογής με την οποία θα μελετήσουμε την απόδοση του συστήματος.

3.1 Υλοποίηση ενός FIR φίλτρου για την Εξαγωγή Πειραμάτων

Για την αποτίμηση του συστήματος δημιουργήσαμε ένα fir φίλτρο 8 συντελεστών ακεραίων μήκους 12 bits του οποίου την λειτουργία θέλουμε να εγγυηθούμε . Τα δεδομένα εισόδου του φίλτρου είναι ακέραιοι αριθμοί μήκους 12 bits. Η απόκριση του ενός fir φίλτρου δίνεται από τον παρακάτω τύπο :

$$y[n] = \Sigma\ \alpha[k]\ x[n-k]$$

όπου α[k] είναι η κρουστική απόκριση του φίλτρου. Οι συντελεστές της κρουστικής απόκρισης του φίλτρου που υλοποιήθηκε είναι οι παρακάτω:

$$\alpha[k]= \{838\ \ -244\ \ 214\ \ -111\ \ -111\ \ 214\ \ -244\ \ 838\}$$

Οι λόγοι για τους οποίους χρησιμοποιήσαμε ένα fir φίλτρο είναι ότι τα φίλτρα είναι από τα ποιο διαδεδομένα κυκλώματα στις τηλεπικοινωνίες και ότι η συγκεκριμένη εφαρμογή όπως και τα κυκλώματα επεξεργασίας δεδομένων δεν αποτελούν safety –critical τμήματα ενός μεγαλύτερου συστήματος .

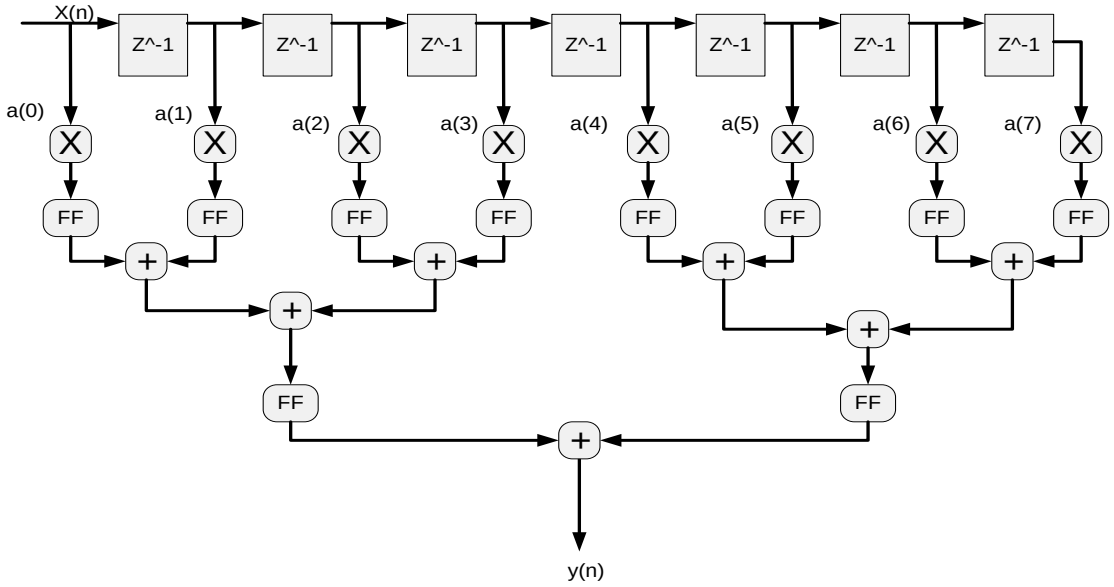
Το φίλτρο έχει τις παρακάτω εισόδους και εξόδους

Σήμα	**Τύπος**	**Μήκος**	**Λειτουργία**
Clk	In	1	Το ρολόι του συστήματος
Reset	In	1	reset του συστήματος
Enable	In	1	Ενεργοποίηση δειγματοληψίας
Datain	In	12	Δεδομένα εισόδου
Dataout	Out	24	Δεδομένα εξόδου

Πίνακας 3.1 Διεπαφή του φίλτρου

3.1.1 Υλοποίηση στην VIRTEX – II PRO

Για την υλοποίηση του φίλτρου χρησιμοποιήθηκε η γλώσσα VHDL. Οι απαιτήσεις του φίλτρου σε αριθμητικές μονάδες και καταχωρητών είναι 7 καταχωρητές των 12 bits, 10 καταχωρητές των 24 bits, 8 πολλαπλασιαστές δύο εισόδων ακεραίων αριθμών μήκους 12 bits η κάθε μία και 7 αθροιστές ακεραίων αριθμών μήκους 12 bits. Για την επίτευξη περιόδου ρολογιού άνω των 100 MHz το φίλτρου υλοποιήθηκε ως pipe line τριών σταδίων. Στο πρώτο στάδιο γίνεται ο πολλαπλασιασμός των συντελεστών με την ακολουθία εισόδου. Στο δεύτερο στάδιο αθροίζονται ανά τέσσερεις οι όροι της εξόδου και στο τελευταίο στάδιο αθροίζονται οι όροι που παράχθηκαν στο δεύτερο στάδιο και παράγεται το τελικό αποτέλεσμα. Οι πολλαπλασιαστές που χρησιμοποιήθηκαν παράχθηκαν μέσω του core generator και οι πόροι που χρησιμοποιήθηκαν είναι CLBs αντί των hardware πολλαπλασιαστών οι οποίοι υπάρχουν ενσωματωμένοι στην XC2VP30. Ο λόγος για τον οποίον έγινε αυτή η επιλογή είναι ότι χρησιμοποιώντας hardware πολλαπλασιαστές χρησιμοποιούμε σε μια περιοχή της FPGA ελάχιστους πόρους των LUTs κάτι το οποίο είναι ανεπιθύμητο εφόσον θέλουμε να μελετήσουμε την επίδραση των SEUs στην λειτουργία της εφαρμογής. Στο Σχήμα 3.1 βλέπουμε την δομή του κυκλώματος :



Σχήμα 3.1 Δομή του FIR φίλτρου

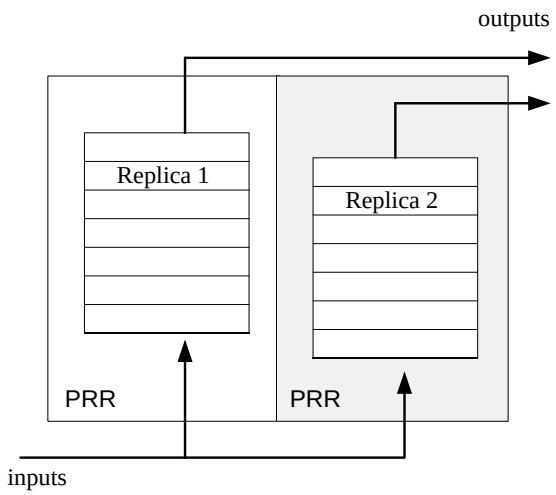
Για να αρχικοποιήσουμε το φίλτρο χρειαζόμαστε 7 κύκλους του ρολογιού ώστε να γράψουμε δεδομένα στους 7 καταχωρητές των 1 2 bits.Το φίλτρο υλοποιήθηκε για την XC2VP30 FPGA. Το φίλτρο έχει την δυνατότητα να επεξεργάζεται δεδομένα κάθε έναν κύκλο. Η μέγιστη συχνότητα λειτουργίας του φίλτρου είναι 111.857 MHz. Οι πόροι οι οποίοι

χρησιμοποιήθηκαν για την υλοποίηση του φίλτρου καθώς και χαρακτηριστικά της λειτουργίας του παρουσιάζονται στον παρακάτω πίνακα.

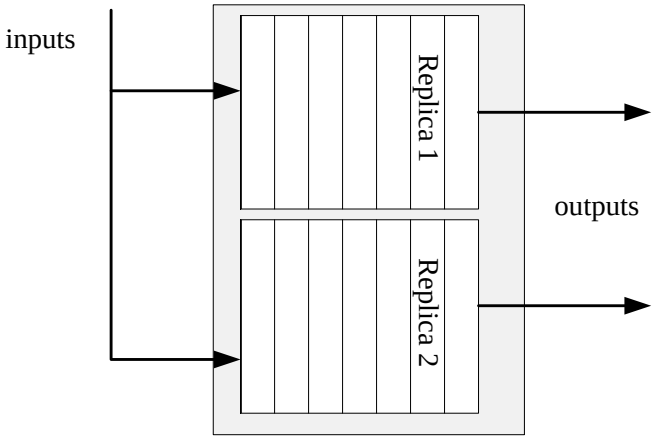
Πόροι	**Απαιτούμενοι Αριθμοί**	**Ποσοστό**
Slices	901	6%
Flip-flops	384	1%
LUTs	144	5%

Πίνακας 3.2. Απαιτούμενοι Πόροι για την υλοποίηση του φίλτρου

Η μέθοδος η οποία χρησιμοποιήθηκε για τον εντοπισμό σφαλμάτων είναι η μέθοδος του duplication αν και θα είχε νόημα να χρησιμοποιηθεί οποιαδήποτε άλλη τεχνική on line testing εκτός των NMR. Η μέθοδος του duplication επιλέχθηκε λόγω της απλότητα της. Έτσι χρησιμοποιούμε δύο αντίγραφα του φίλτρου τα οποία δέχονται ως είσοδο τα ίδια δεδομένα και οι έξοδοι τους συγκρίνονται μέσω μίας πύλης XOR της οποίας η έξοδος στέλνεται στον PPC μέσω ενός περιφερειακό του. Για την επιδιόρθωση των δύο αντιγράφων του φίλτρου σε περίπτωση που συμβεί κάποιο SEU αυτά θα τοποθετηθούν σε κάποια PRR ώστε να είναι δυνατόν να αναδιαταχθούν δυναμικά. Τα δύο αντίγραφα μπορεί να τοποθετηθούν στην ίδια ή σε διαφορετική PRR. Τοποθετώντας τα δύο αντίγραφα στην ίδια PRR έχουμε το πλεονέκτημα ότι αξιοποιούμε σε μεγαλύτερο ποσοστό τους δεσμευμένους πόρους της PRR ενώ τοποθετώντας σε διαφορετικές PRRs τα δύο αντίγραφα εκτός του ότι είναι εντελώς ανεξάρτητα μας δίνεται και η δυνατότητα να εντοπίζουμε την περιοχή στην οποία συνέβη το σφάλμα και να αναδιατάσσουμε μόνο αυτή .



Σχήμα 3.2 Τοποθέτηση των δύο αντιγράφων σε διαφορετικές PRR



Σχήμα 3.3 Τοποθέτηση των δύο αντιγράφων στην ίδια PRR

4 Αρχιτεκτονική Συστήματος

Σκοπός της εργασίας όπως έχει ήδη αναφερθεί είναι η δημιουργία και η αξιολόγηση ενός συστήματος το οποίο λειτουργεί επικουρικά σε μη safety critical εφαρμογές, υλοποιημένες σε SRAM-based FPGAs, δηλαδή σε εφαρμογές οι οποίες έχουν το περιθώριο να παράγουν λανθασμένα αποτελέσματα εξαιτίας κάποιου σφάλματος για μικρά χρονικά διαστήματα έως ώσπου να εντοπιστεί, να αντιμετωπιστεί και να αποκατασταθεί η λειτουργία της εφαρμογής .

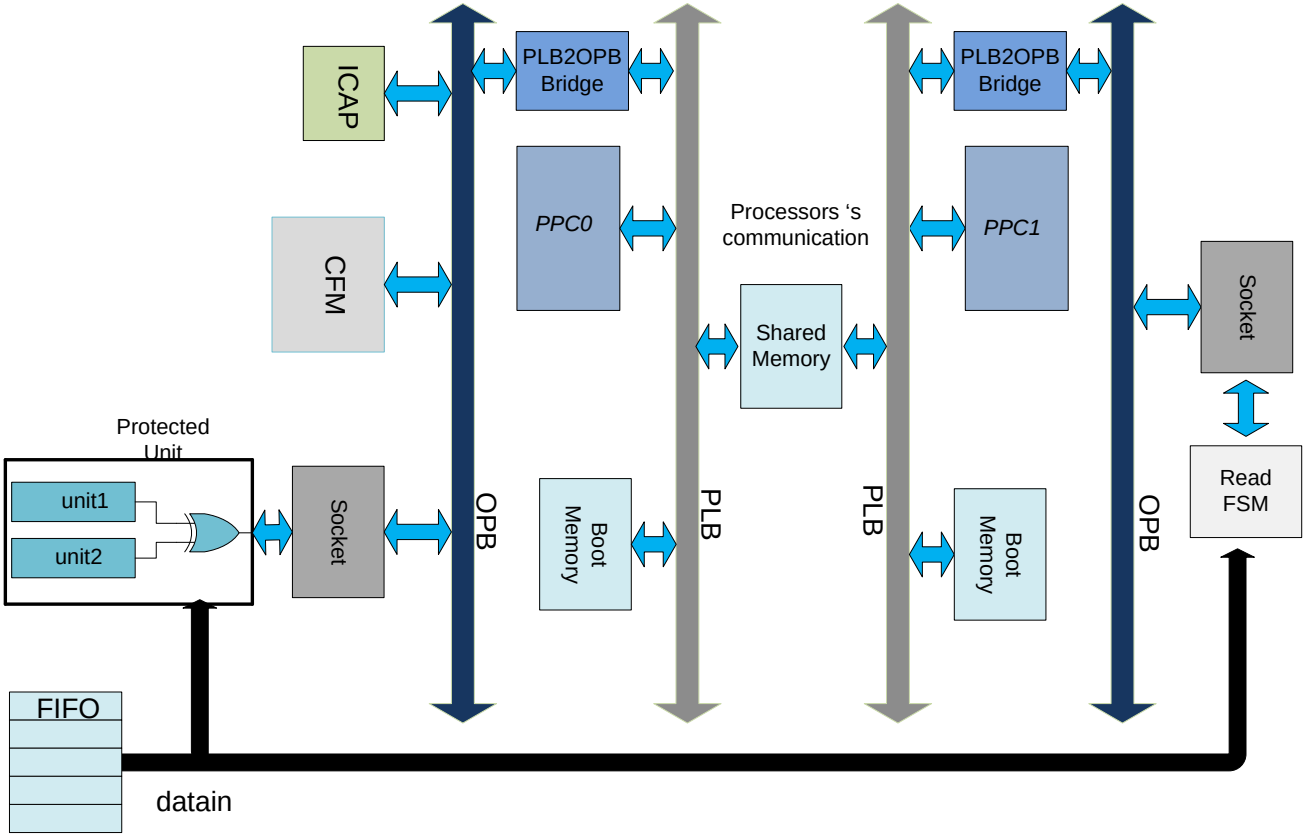
Όπως είδαμε στην περιγραφή της αρχιτεκτονικής των FPGAs ορισμένες συσκευές εκτός από τους αναδιατασσόμενους πόρους περιέχουν και ενσωματωμένους μικροεπεξεργαστές οι οποίοι παρέχουν την δυνατότητα υλοποίησης αυτόνομων συστημάτων μέσα στις FPGAs χρησιμοποιώντας τους αναδιατασσόμενους πόρους της συσκευής για την υλοποίηση των περιφερειακών τους η των ελεγκτών τους. Η XC2VP30 FPGA που χρησιμοποιήσαμε διαθέτει δύο VLSI επεξεργαστές. Συγκεκριμένα διαθέτει δυο Power Pc 405 της IBM (ppc405). Επιπλέον η ύπαρξη του ICAP επιτρέπει την εσωτερική (άρα και γρηγορότερη) αναδιάταξη της FPGA κάνοντας εφικτή την δημιουργία αυτόνομων αναδιατασσόμενων συστημάτων.

Αρχική ιδέα ήταν η χρησιμοποίηση ενός ενσωματωμένου επεξεργαστή σε συνεργασία με μια FIFO. Ο επεξεργαστής θα αναλάμβανε τον έλεγχο της λειτουργίας της υλοποιημένης εφαρμογής και σε περίπτωση που εντοπιζότανε κάποιο σφάλμα θα πρόβαινε στην διόρθωση του μέσω μερικής αναδιάταξης καθώς και στην ενεργοποίηση της FIFO ώστε να συγκεντρωθούν τα δεδομένα που στέλνονται στην εφαρμογή όσο αυτή είναι ανενεργή ώστε να αποφευχθεί η απώλεια τους και να επεξεργαστούν όταν θα έχει ολοκληρωθεί η αποκατάσταση της εφαρμογής από το σφάλμα.

Το γεγονός της ύπαρξης και δεύτερου VLSI επεξεργαστή μας δίνει την δυνατότητα για κάποιες βελτιστοποιήσεις της αρχικής ιδέας. Ο δεύτερος επεξεργαστής μπορεί να χρησιμοποιηθεί ώστε επεξεργάζεται τα δεδομένα το χρονικό διάστημα κατά το οποίο η εφαρμογή βρίσκεται στην φάση αποκατάστασης από κάποιο σφάλμα. Η χρησιμοποίηση του δεύτερου επεξεργαστή μειώνει το απαιτούμενο βάθος της FIFO ούτως ώστε να αποφύγουμε απώλεια δεδομένων. Επιπλέον το γεγονός ότι χρησιμοποιούμε έναν ήδη προϋπάρχον επεξεργαστή έχει ως αποτέλεσμα των περιορισμό των επιπλέων απαιτούμενων αναδιατασσόμενων πόρων για την λειτουργία του.

Το τελικό σύστημα το οποίο υλοποιήθηκε αποτελείται από τους δύο επεξεργαστές Power Pc (PPC405) που διαθέτει η FPGA, από την FIFO η οποίο παρέχει τα δεδομένα προς επεξεργασία καθώς και την εφαρμογή η οποία είναι υλοποιημένη με τους διατασσόμενους πόρους της FPGA. Κάθε επεξεργαστής έχει το δικό του υποσύστημα από περιφερειακές

μονάδες και τους δικούς του διαύλους. Γενικά τα δύο υποσυστήματα είναι ανεξάρτητα και η επικοινωνία μεταξύ τους γίνεται μέσω μίας μνήμης η οποία είναι προσπελάσιμη και στα δύο συστήματα των επεξεργαστών μέσω δύο ξεχωριστών ελεγκτών μνήμης. Το σύστημα το οποίο υλοποιήθηκε απεικονίζεται στο παρακάτω σχήμα.



Σχήμα 4.1 Απεικόνιση του συνολικού συστήματος

4.1 Γενική Λειτουργία Συστήματος

Το σύστημα λειτουργεί ως εξής:

- Ο PPC0 ελέγχει τις εξόδους της εφαρμογής (ένα φίλτρο fir ακαριαίων συντελεστών) το οποίο θέλουμε να προστατεύσουμε από SEUs και σε περίπτωση που διαπιστωθεί κάποιο σφάλμα θα ειδοποιεί τον PPC1 για την ύπαρξη σφάλματος και θα προχωράει στην αποκατάσταση του περιφερειακού μέσω μερικής αναδιάταξη. Όταν ολοκληρώσει την αναδιάταξη θα ειδοποιήσει τον PPC1. Εκτός των φίλτρων ελέγχει και την FIFO για την ύπαρξη σφάλματος.

- Ο PPC1 αναλαμβάνει την λειτουργία της εφαρμογής, στην περίπτωση που ειδοποιήθηκε για την ύπαρξη σφάλματος από τον PPC0, διαβάζοντας τα δεδομένα εισόδου και επεξεργάζοντας τα για όσο χρονικό διάστημα διαρκέσει η αποκατάσταση του περιφερειακού που υλοποιεί την εφαρμογή .

Τόσο ο PPC1 όσο και το περιφερειακό που υλοποιεί την εφαρμογή λαμβάνουν τα δεδομένα μέσω μίας FIFO στην οποία κατά την λειτουργία της εφαρμογής ο ρυθμός με τον οποίο δύναται να διαβαστούν τα περιεχόμενα της είναι μεγαλύτερος ή ίσος από τον ρυθμό εισόδου των δεδομένων στο σύστημα αποφεύγοντας με αυτόν τον τρόπο το γέμισμα της. Το βάθος της FIFO εξαρτάται από την ταχύτητα που επεξεργάζεται ο επεξεργαστής τα δεδομένα, την ώρα που χρειάζεται για την αναδιάταξη της PRR και τον ρυθμό εισόδου των δεδομένων.

4.2 FSM Ανάγνωσης Δεδομένων

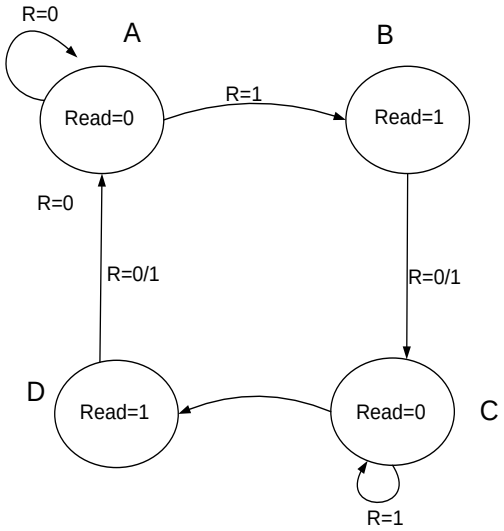
Όπως αναφέρθηκε παραπάνω όταν εντοπιστεί κάποιο σφάλμα στα δύο φίλτρα τότε την επεξεργασία των δεδομένων την αναλαμβάνει ο PPC1. Έτσι θα πρέπει να ενεργοποιεί και να απενεργοποιεί το σήμα read της FIFO ώστε να διαβάζει τα επόμενα δεδομένα εισόδου. Επειδή χρειάζονται περισσότεροι του ενός κύκλου για την ενεργοποίηση και απενεργοποίηση του σήματος Read της FIFO μέσω των περιφερειακών ενός επεξεργαστή ενσωματωμένου στην FPGA είναι απαραίτητη η ύπαρξη κάποιου συγχρονισμού για την αποφυγή απώλειας δεδομένων που θα ενεργοποιεί ακριβώς για έναν κύκλο το σήμα Read της FIFO. Για τον συγχρονισμό της επικοινωνίας της FIFO με τον PPC δημιουργήσαμε μία FSM η οποία ανάλογα με τις μεταβολές της εισόδου δεδομένων της μεταβαίνει στις κατάλληλες καταστάσεις και ενεργοποιεί η απενεργοποιεί το σήμα Read της FIFO. Στον παρακάτω πίνακα παρουσιάζεται η διεπαφή της FSM.

Σήμα	**Τύπος**	**Μήκος**	**Λειτουργία**
Clk	In	1	Το ρολόι του συστήματος
Reset	In	1	reset του συστήματος
R	In	1	Αίτηση για ανάγνωση
Read	Out	1	Ενεργοποίηση σήματος Read της FIFO

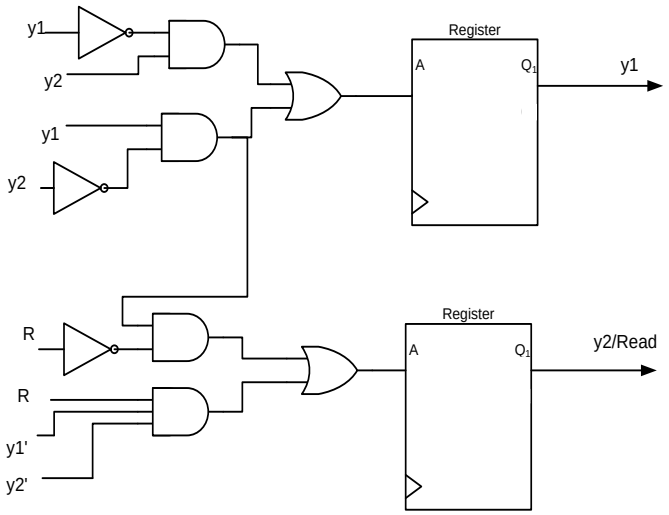
Πίνακας 4.1. Διεπαφή FSM Ανάγνωσης

Η επικοινωνία του PPC με την FSM γίνεται μέσω του περιφερειακού OPB DCR Socket. Η FSM παραμένει στην αρχική κατάσταση για όσο χρονικό διάστημα η είσοδος της

παραμένει σταθερή. Μόλις ζητήσει ο επεξεργαστής κάποιο νέο δεδομένο αντιστρέφοντας το σήμα εισόδου R της FSM τότε μεταβαίνει στην δεύτερη κατάσταση της και ενεργοποιεί το σήμα read της FIFO για ένα κύκλο και στην συνέχεια θα μεταβεί στην τρίτη κατάσταση της και θα παραμείνει σε αυτήν μέχρι να ζητηθούν νέα δεδομένα από τον επεξεργαστή και στην συνέχεια επιστρέφει στην αρχική της κατάσταση. Το διάγραμμα καταστάσεων της FSM ανάγνωσης παρουσιάζεται στο παρακάτω σχήμα.



Σχήμα 4.3 Διάγραμμα καταστάσεων της FSM ανάγνωσης



Σχήμα 4.5 Υλοποίηση της FSM ανάγνωσης .

4.3 FIFO Δεδομένων

Τα δεδομένα τα οποία στέλνονται προς επεξεργασία στο σύστημα προσωρινά αποθηκεύονται σε μία FIFO από όπου στην συνέχεια θα διαβαστούν από το σύστημα. Η FIFO έχει την παρακάτω διεπαφή.

Σήμα	**Τύπος**	**Μήκος**	**Λειτουργία**
Clk	In	1	Το ρολόι του συστήματος
Reset	In	1	reset του συστήματος
Datain	In	12	Δεδομένα εισόδου
Write	In	1	Ενεργοποίηση εγγραφής
RdfromPPC0	In	1	Ενεργοποίηση ανάγνωσης από PPC0
RdfromPPC1	In	1	Ενεργοποίηση ανάγνωσης από PPC1
Empty	Out	1	Σήμα ειδοποίησης ότι η μνήμη είναι άδεια
Full	Out	1	Σήμα ειδοποίησης ότι η μνήμη είναι γεμάτη
Enfir0	Out	1	Σήμα ενεργοποίησης μονάδας φίλτρου #0
Enfir1	Out	1	Σήμα ενεργοποίησης μονάδας φίλτρου #1
Dataout	Out	12	Δεδομένα εξόδου

Πίνακας 4.2. Διεπαφή της FIFO

Για την υλοποίηση της μνήμης της FIFO χρησιμοποιήθηκαν εσωτερικές BRAMs της FPGA ενώ για τον controller χρησιμοποιήθηκε αναδιατασσόμενη λογική της FPGA (LUTs κλπ) . Παρακάτω αναφέρουμε τις λεπτομέρειες υλοποίησης του controller .

4.3.1 Υλοποίηση του Controller της FIFO

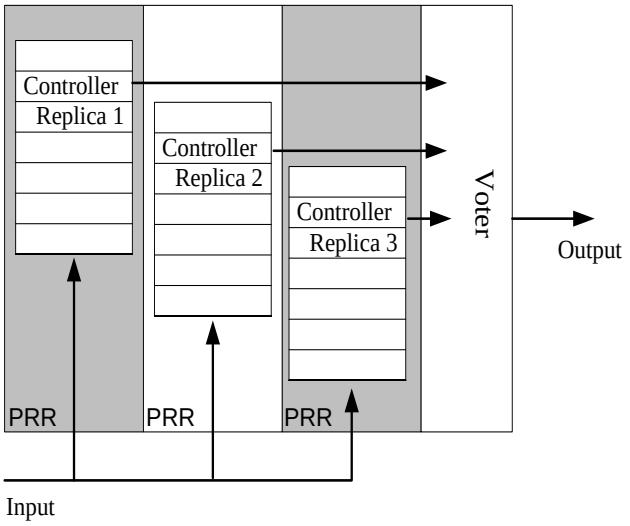
Σκοπός του controller της FIFO είναι η παραγωγή των διευθύνσεων ανάγνωσης και εγγραφής δεδομένων, η ενεργοποίηση και απενεργοποίηση των σημάτων full και empty της FIFO καθώς και η ενεργοποίηση του σήματος enable των φίλτρων όταν η FIFO δεν είναι άδεια και κάποιο από τα σήματα ανάγνωσης τα οποία οδηγούνται από τους επεξεργαστές είναι ενεργοποιημένο. Για την υλοποίηση των καταχωρητών που περιέχουν τις διευθύνσεις ανάγνωσης και εγγραφής χρησιμοποιήσαμε LFSR.

Σήμα	**Τύπος**	**Μήκος**	**Λειτουργία**
Clk	In	1	Το ρολόι του συστήματος
Reset	In	1	reset του συστήματος
Write	In	1	Ενεργοποίηση εγγραφής
Read	In	1	Ενεργοποίηση ανάγνωσης
RdfromPPC0	In	1	Ενεργοποίηση ανάγνωσης από PPC0
RdfromPPC1	In	1	Ενεργοποίηση ανάγνωσης από PPC1
InitData	In	22	Δεδομένα αρχικοποίησης του Controller
Empty	Out	1	Σήμα ειδοποίησης ότι η μνήμη είναι άδεια

Full	Out	1	Σήμα ειδοποίησης ότι η μνήμη είναι γεμάτη
Enfir0	Out	1	Σήμα ενεργοποίησης μονάδας φίλτρου #0
Enfir1	Out	1	Σήμα ενεργοποίησης μονάδας φίλτρου #1

Πίνακας 4.3. Διεπαφή του FIFO Controller

Η FIFO τροφοδοτεί με δεδομένα τόσο τα δύο φίλτρα όσο και τον PPC1 επομένως θα πρέπει να λειτουργεί συνέχεια. Εξαιτίας αυτού του γεγονότος πρέπει να παράγει συνεχώς σωστά αποτελέσματα και να προστατεύεται από SEUs. Για την εγγύηση της λειτουργίας του controller χρησιμοποιήθηκε η τεχνική του TMR όπου κάθε αντίγραφο τοποθετείται σε ξεχωριστή PRR ώστε σε περίπτωση που συμβεί κάποιο SEU σε ένα από τα αντίγραφα αυτό να μπορεί να αναδιατάσσεται δυναμικά ενώ τα υπόλοιπα δύο να συνεχίζουν απρόσκοπτα την λειτουργίας τους. Τέλος όσον αφορά την μνήμη της FIFO λόγω του γεγονότος ότι τα δεδομένα της διαβάζονται σε πολύ σύντομο χρονικό διάστημα από την στιγμή που θα εγγραφούν σε αυτή (μικρότερο τον 300 κύκλων) δεν θεωρήθηκε σκόπιμο να υπάρξει κάποια τεχνική για την ανίχνευση σφαλμάτων σε αυτά .



Σχήμα 4.4 Τοποθέτηση των αντιγράφων του Controller της FIFO στις PRRs

4.4 Σύστημα Επεξεργαστών

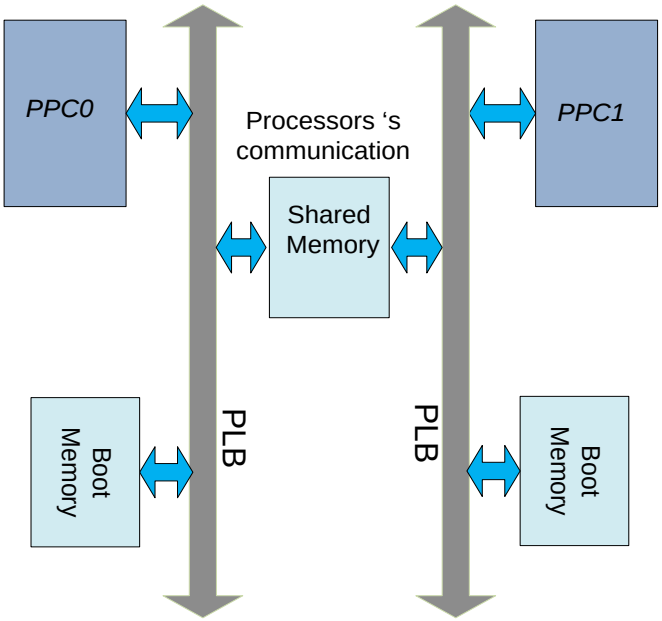
Όπως έχει ήδη αναφερθεί το σύστημα των επεξεργαστών είναι υπεύθυνο για τον έλεγχο της λειτουργίας του συνολικού συστήματος και για την διόρθωση SEUs τα οποία μπορούν να προκύψουν στην configuration memory της FPGA. Το σύστημα αυτό αποτελείται από τα γενικώς ανεξάρτητα υποσυστήματα των δύο ενσωματωμένων Power PC επεξεργαστών. Στις επόμενες ενότητες περιγράφουμε αναλυτικά την αρχιτεκτονική των δύο υποσυστημάτων.

4.4.1 Επικοινωνία Επεξεργαστών

Η υλοποίηση της επικοινωνίας δύο επεξεργαστών μπορεί να γίνει με δύο τρόπους . Μπορεί να χρησιμοποιηθεί κάποια κοινή μνήμη στην οποία έχουν πρόσβαση και οι δύο επεξεργαστές ή μπορεί να υλοποιηθεί μέσω κάποιου περιφερειακού XPS Mailbox το οποίο μπορεί να συνδεθεί στους PLB διαύλους των επεξεργαστών που χρειάζεται να επικοινωνήσουν. Παρακάτω αναφέρουμε λεπτομέρειες για τους δύο αυτούς τρόπους επικοινωνίας επεξεργαστών.

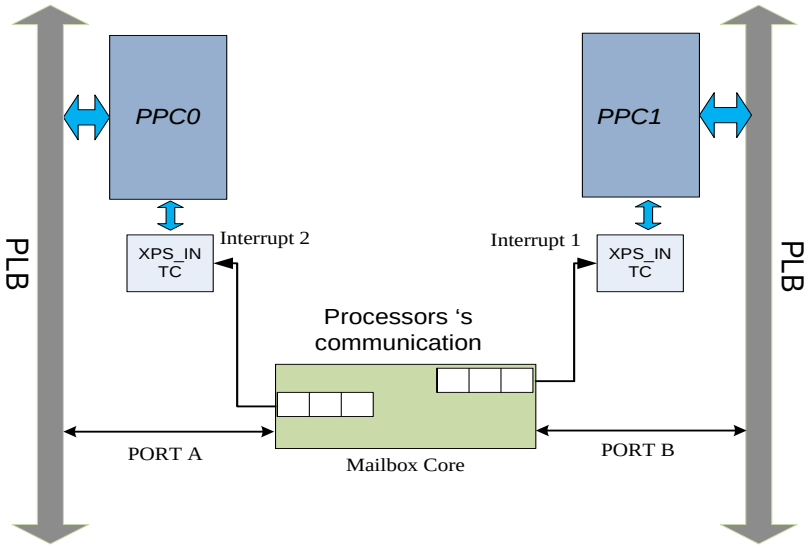
Η επικοινωνία μέσω κοινής μνήμης είναι ο απλούστερος και πιο διαδεδομένος τρόπος επικοινωνίας μεταξύ συστημάτων. Η χρησιμοποίηση κοινής μνήμης είναι ο ταχύτερος ασύγχρονος τρόπος επικοινωνίας για την μεταφορά μεγάλων μηνυμάτων (>1000 bytes). Ένας επεξεργαστής μπορεί να στείλει η να παραλάβει κάποιο μήνυμα απλά μέσω εντολών load/store. Ο τύπος της κοινή μνήμης που μπορεί να χρησιμοποιηθεί μπορεί να είναι είτε κάποια εσωτερική BRAM της FPGA ή κάποια εξωτερική μνήμη. Στην περίπτωση της εξωτερικής κοινής μνήμης είναι απαραίτητη η χρήση MPMC memory controller όπου κάθε θύρα του controller χαρτογραφείται στην ίδια εμβέλεια διευθύνσεων και οποιοσδήποτε επεξεργαστής έχει πρόσβαση σε ολόκληρη την εξωτερική μνήμη. Οι μνήμες BRAM που βρίσκονται ενσωματωμένες στην FPGA είναι dual ported το οποίο έχει ως αποτέλεσμα να μπορούν να χρησιμοποιηθούν ταυτόχρονα το πολύ από δύο επεξεργαστές. Οι επεξεργαστές συνδέονται στις δύο διαφορετικές θύρες της μνήμης μέσω memory controllers.

Για τον προγραμματισμό ενός συστήματος επεξεργαστών οι οποίοι επικοινωνούν μέσω κοινής μνήμης μπορεί να χρησιμοποιηθεί η προσέγγιση των νημάτων με την διαφορά ότι αντί να έχουμε έναν επεξεργαστή στον οποίο εκτελούνται πολλαπλά νήματα έχουμε ένα νήμα για κάθε επεξεργαστή του συστήματος. Από τα παραπάνω γίνεται φανερό ότι για να αποφευχθούν συνθήκες ανταγωνισμού είναι απαραίτητη ή ύπαρξη λειτουργιών Lock και Unlock. Για την υποστήριξη των παραπάνω λειτουργιών υπάρχει το περιφερειακό XPS Mutex Core. Συνήθως δύο επεξεργαστές που επικοινωνούν μέσω κάποιος κοινής μνήμης εκτός της κοινής μνήμης κάθε επεξεργαστής έχει και μία και ιδιωτική μνήμη όπου βρίσκεται το εκτελέσιμο του.



Σχήμα 4.5 Υλοποίηση επικοινωνίας μέσω κοινής μνήμης

Η επικοινωνία μέσω Mailboxes είναι κατάλληλη για μεταφορά μικρών μηνυμάτων (συνήθως μικρότερα των 100 bytes). Η επικοινωνία γίνεται ουσιαστικά με την χρησιμοποίηση μίας FIFO για την αποστολή και μία FIFO για την λήψη μηνυμάτων σε κάθε άκρο. Η λήψη από τον παραλήπτη ενός μηνύματος μπορεί να γίνει είτε σύγχρονα μέσω polling είτε ασύγχρονα μέσω interrupts. Τέλος τα Mailboxes μπορούν να συνδυαστούν και με κάποια κοινή μνήμη για την αποστολή μεγάλου όγκου δεδομένων .



Σχήμα 4.6 Υλοποίηση επικοινωνίας μέσω Mailbox

Οι ανάγκες επικοινωνίας στο σύστημα που υλοποιήσαμε είναι αρκετά απλές. Σε περίπτωση σφάλματος στα φίλτρα ο PPC0 πρέπει να ενημερώσει τον PPC1 ώστε να αναλάβει την επεξεργασία των δεδομένων και όταν αποκατασταθεί η λειτουργία των

φίλτρων να τον ενημερώσει ώστε να σταματήσει. Από της δύο παραπάνω μεθόδους επικοινωνίας που παρουσιάστηκαν και σύμφωνα με τις δικές μας απαιτήσεις η καλύτερη επιλογή είναι αυτή των Mailboxes. Όμως λόγω του ότι η τελευταία έκδοση των εργαλείων EDK και ISE η οποία υποστηρίζει την μερική αναδιάταξη δεν υποστηρίζει τα Mailboxes χρησιμοποιήσαμε την μέθοδο της κοινής μνήμης. Επίσης λόγω του ότι η συγκεκριμένη έκδοση δεν υποστηρίζει ούτε το περιφερειακό XPS Mutex Core για την αποφυγή συνθηκών ανταγωνισμού χρησιμοποιήσαμε δύο διευθύνσεις μνήμης όπου στην πρώτη γράφει την τιμή 1 ο PPC0 (παραγωγός) σε περίπτωση που προκύψει σφάλμα στο φίλτρο και ο PPC1 (καταναλωτής) διαβάζει την τιμή της διεύθυνσης αυτής ενώ την δεύτερη θέση μνήμης την χρησιμοποιεί ο PPC1 (παραγωγός) για να ενημερώσει τον PPC0 (καταναλωτής).

4.4.2 Υποσύστημα Επεξεργαστή PPC0

Σκοπός του συστήματος του PPC0 όπως αναφέρθηκε είναι η διόρθωση SEUs τα οποία συμβαίνουν καθώς και ο έλεγχος των δύο αντιγράφων του φίλτρου. Τα δεδομένα για τον έλεγχο της σωστής λειτουργίας των φίλτρων τα λαμβάνει μέσω ενός περιφερειακού το οποίο βρίσκεται πάνω στο OPB DCR Socket. Η διαδρομή των δεδομένων ελέγχου για να φθάσουν στον επεξεργαστή είναι η παρακάτω : Socket => OPB => PLB2OPB Bridge => PLB =>PPC0 .

Για την επίτευξη του παραπάνω σκοπού το υποσύστημα του PPC0 έχει τις εξής περιφερειακές μονάδες οι οποίες συνδέονται μαζί του μέσω των δίαυλων PLB, OPB και DCR:

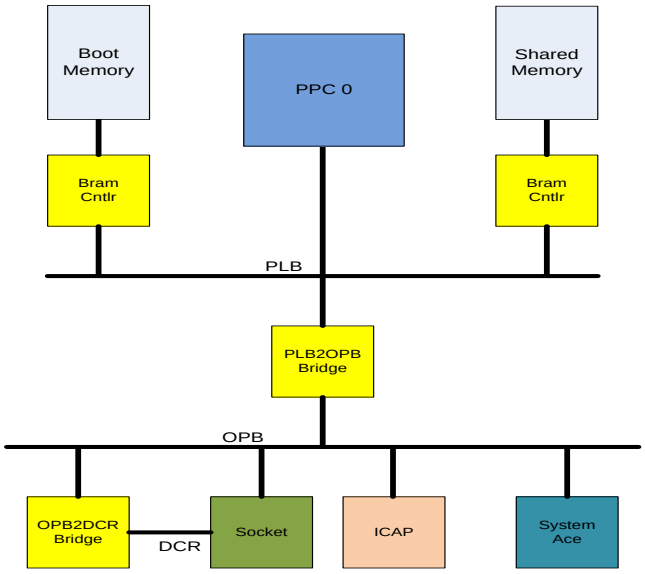
- ICAP: Το περιφερειακό αυτό βρίσκεται πάνω στον δίαυλο OPB και μέσω αυτού ο PPC0 έχει πρόσβαση στην configuration memory της FPGA επίσης το περιφερειακό αυτό έχει την δική του BRAM μέσω τις οποίας επικοινωνεί με τον PPC0
- OPB DCR Socket: Το περιφερειακό αυτό βρίσκεται πάνω στον δίαυλο OPB και DCR. Μέσω των καταχωρητών του DCR μεταφέρουμε τα σήματα του διαύλου OPB του συστήματος έξω από το σύστημα ώστε ο PPC0 να μπορεί να επικοινωνεί με τα φίλτρα όπως και με τον ελεγκτή της FIFO.
- Ελεγκτή System Ace: Το περιφερειακό αυτό βρίσκεται πάνω στον δίαυλο OPB και μέσω αυτού ο PPC0 έχει πρόσβαση σε μία εξωτερική Compact Flash Memory (CFM) στην οποία βρίσκονται αποθηκευμένα η συνολική σχεδίαση καθώς και τα partial bitstreams.
- Boot Memory : Η μνήμη αυτή βρίσκεται πάνω στον δίαυλο PLB και περιέχει το πρόγραμμα και τα δεδομένα του προγράμματος το οποίο εκτελείται στον PPC0.
- Shared Memory : Η μνήμη αυτή βρίσκεται πάνω στον δίαυλο PLB και χρησιμοποιείται για την επικοινωνία με τον PPC1.

Στον παρακάτω πίνακα παρουσιάζουμε τον χάρτη διευθύνσεων του επεξεργαστή PPC0

Αρχική Διεύθυνση	**Μέγεθος**	**Περιφερειακό**
0x40200000	64K	ICAP
0x7d800000	64K	OPB DCR Socket
0x41800000	64K	System Ace
0xffff0000	64K	Boot Memory
0x00000000	16K	Shared Memory

Πίνακας 4.4 Memory Map PPC0

Στο παρακάτω σχήμα παρουσιάζεται η αρχιτεκτονική του υποσυστήματος του PPC0



Σχήμα 4.7 Υποσύστημα PPC0

Στον παρακάτω πίνακα δείχνουμε τους πόρους με τους οποίους υλοποιούνται οι διάφορες λειτουργίες του υποσυστήματος του επεξεργαστή PPC0.

Εργασία	**Πόροι συστήματος**
Ανίχνευση Σφάλματος	PPC0 ,δίαυλοι του επεξεργαστή και περιφερειακού OPB DCR Socket
Επικοινωνία Επεξεργαστών	PPC0, PPC1 και Shared Memory
Μεταφορά δεδομένων από CFM προς PPC0	PPC0, δίαυλοι του επεξεργαστή και CFM
Μεταφορά δεδομένων από PPC0 προς BPAM ICAP	PPC0 , δίαυλοι του επεξεργαστή και ICAP
Εγγραφή δεδομένων στην configuration Memory	PPC0 , δίαυλοι του επεξεργαστή και ICAP

Πίνακας 4.5Υλοποίηση εργασιών υποσυστήματος PPC0

4.4.3 Υποσύστημα Επεξεργαστή PPC1

Σκοπός του συστήματος του PPC1 όπως αναφέρθηκε είναι η επεξεργασία των δεδομένων εισόδου στην περίπτωση όπου έχει συμβεί κάποιο SEU στο περιφερειακό το οποίο υλοποιεί την εφαρμογή του FIR. Η διαδρομή των δεδομένων προς επεξεργασία και των δεδομένων ελέγχου για να φθάσουν στον επεξεργαστή είναι η παρακάτω : Socket => OPB => PLB2OPB Bridge => PLB =>PPC1.

Για την επίτευξη του παραπάνω σκοπού το υποσύστημα του PPC0 έχει τις εξής περιφερειακές μονάδες οι οποίες συνδέονται μαζί του μέσω των δίαυλων PLB, OPB και DCR:

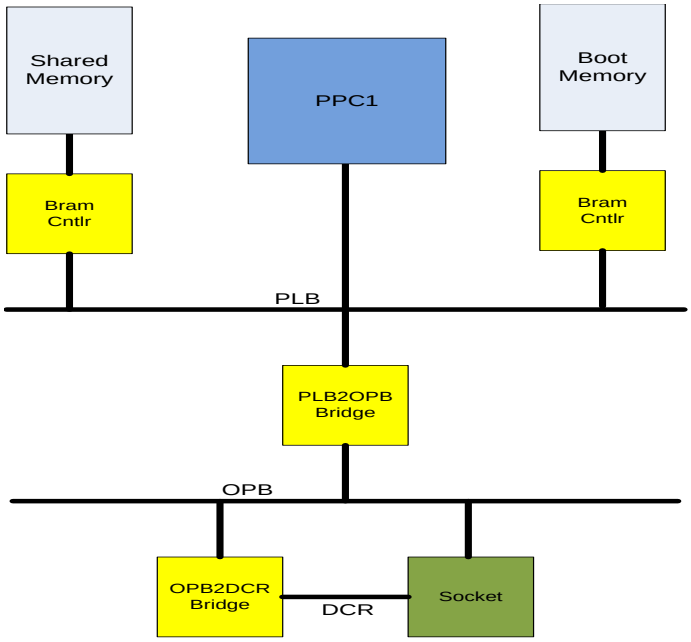
- OPB DCR Socket: Το περιφερειακό αυτό βρίσκεται πάνω στον δίαυλο OPB και DCR. Μέσω των καταχωρητών του DCR μεταφέρουμε τα σήματα του διαύλου OPB του συστήματος έξω από το σύστημα ώστε μέσω να μπορεί ο PPC1 να διαβάζει τα δεδομένα τα οποία βρίσκονται αποθηκευμένα στην FIFO.
- Boot Memory : Η μνήμη αυτή βρίσκεται πάνω στον δίαυλο PLB και περιέχει το πρόγραμμα και τα δεδομένα του προγράμματος το οποίο εκτελείται στον PPC1. Η μνήμη αυτή είναι επίσης προσπελάσιμη και από τον PPC0.
- Shared Memory : Η μνήμη αυτή βρίσκεται πάνω στον δίαυλο PLB και χρησιμοποιείται για την επικοινωνία με τον PPC0.

Στον παρακάτω πίνακα παρουσιάζουμε τον χάρτη διευθύνσεων του επεξεργαστή PPC1

Αρχική Διεύθυνση	**Μέγεθος**	**Περιφερειακό**
0x7d800000	64K	OPB DCR Socket
0xffffc000	16KB	Boot Memory
0x00000000	16K	Shared Memory

Πίνακας 4.6 Memory Map PPC1

Στο παρακάτω σχήμα παρουσιάζεται η αρχιτεκτονική του υποσυστήματος του PPC1



Σχήμα 4.8 Υποσύστημα PPC1

Στον παρακάτω πίνακα δείχνουμε τους πόρους με τους οποίους υλοποιούνται οι διάφορες λειτουργίες του υποσυστήματος του επεξεργαστή PPC1.

Εργασία	**Πόροι συστήματος**
Επικοινωνία Επεξεργαστών	PPC0, PPC1 και shared memory
Επεξεργασία Δεδομένων	PPC1, δίαυλοι του επεξεργαστή και περιφερειακού OPB DCR Socket

Πίνακας 4.7 Υλοποίηση εργασιών υποσυστήματος PPC0

5 Υλοποίηση Συστήματος

Στο κεφάλαιο αυτό παρουσιάζεται η σχεδίαση και η υλοποίηση των υποσυστημάτων καθώς και η ενοποίηση τους για την δημιουργία του συνολικού συστήματος. Επιπλέον αναλύονται οι απαιτήσεις του συστήματος σε πόρους υλικού καθώς και η ανάπτυξη του λογισμικού των δύο επεξεργαστών .

5.1 Εργαλεία Υλοποίησης

Τα εργαλεία που χρησιμοποιήθηκαν είναι τα ISE 9.1.02 (SP2 + PR Flow) , EDK 9.1.02 και το Planahead 10.1.

Το ISE χρησιμοποιήθηκε για την σχεδίαση και την σύνθεση των φίλτρων , της FSM ανάγνωσης και της FIFO. Η γλώσσα που χρησιμοποιήθηκε για την ανάπτυξη των παραπάνω μονάδων ήταν η VHDL. Επίσης το ISE χρησιμοποιήθηκε για την ένωση των παραπάνω μονάδων καθώς και του συστήματος των επεξεργαστών για την ολοκλήρωση και την σύνθεση της τελικού συστήματος.

Το EDK χρησιμοποιήθηκε για την σχεδίαση του συστήματος των επεξεργαστών καθώς και για την υλοποίηση και την αποσφαλμάτωση (μέσω XMD) του λογισμικού κάθε επεξεργαστή .

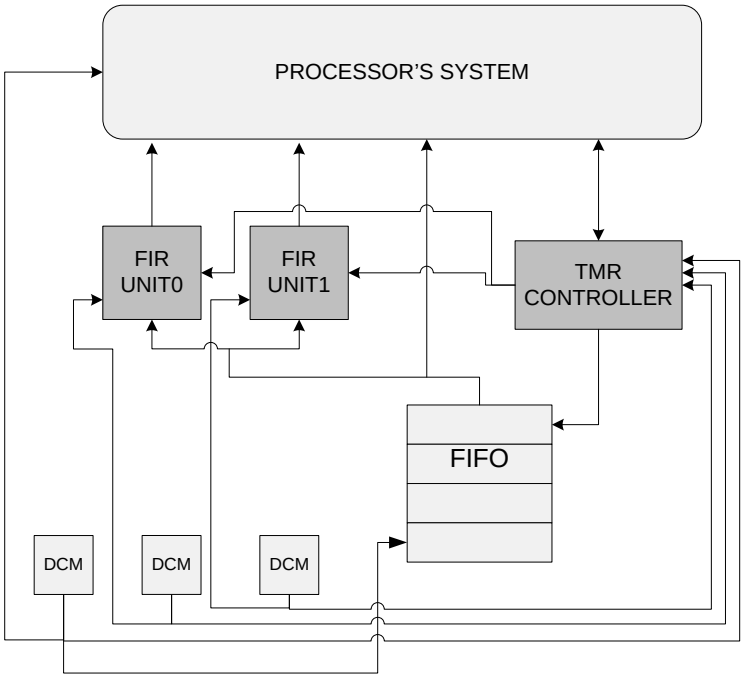
Το Planahead χρησιμοποιήθηκε για την τοποθέτηση των μονάδων του συνολικού συστήματος σε συγκεκριμένες περιοχές της FPGA (floorplaning) . Επίσης χρησιμοποιήθηκε στον ορισμό των PRRs καθώς και των PRMs κάθε PRR και στην τοποθέτηση των bus macros και του DCM που παράγει το ρολόι του συστήματος. Τέλος μέσω του Planahead δημιουργήθηκαν τα partial bitstreams κάθε PRM καθώς και του ολικού bitstream.

Συνολικά υλοποιήθηκαν δύο σχεδιάσεις μία στην οποία τα δύο φίλτρα τοποθετήθηκαν στην ίδια PRR και μία στην οποία τα δύο φίλτρα τοποθετήθηκαν σε διαφορετικές .

5.2 Υλοποίηση Στατικής Μονάδας

Με τον όρο στατική μονάδα εννοούμε το σύστημα των δύο επεξεργαστών των περιφερειακών τους καθώς την μνήμη την οποία χρησιμοποιούμε ως FIFO και τα DCM τα οποία χρησιμοποιούμε για την παραγωγή των ρολογιών. Για την σωστή υλοποίηση των τεχνικών του TMR και του Duplication είναι απαραίτητο να χρησιμοποιήσουμε διαφορετικά σήματα εισόδου στα διάφορα αντίγραφα των μονάδων. Η απαίτηση αυτή ισχύει και για το

σήμα του ρολογιού. Συνολικά χρησιμοποιήσαμε 3 DCM τα οποία παράγουν τρία σήματα ρολογιού συχνότητας 100 MHz. Στο Σχήμα 5.1 παρουσιάζεται οι συνδεσμολογία των βασικών στοιχείων ώστε να δημιουργηθεί το συνολικό σύστημα .



Σχήμα 5.1 Σύνδεση στατικής μονάδας με τις αναδιατασσόμενες. Οι αναδιατασσόμενες μονάδες παρουσιάζονται με σκούρο χρώμα. Και τα τρία DCM παράγουν σήματα ρολογιού των οποίων η συχνότητα είναι 100 MHz .

Οι συνολικοί πόροι οι οποίοι χρησιμοποιήθηκαν για την υλοποίηση παρουσιάζεται στον Πίνακα 5.1

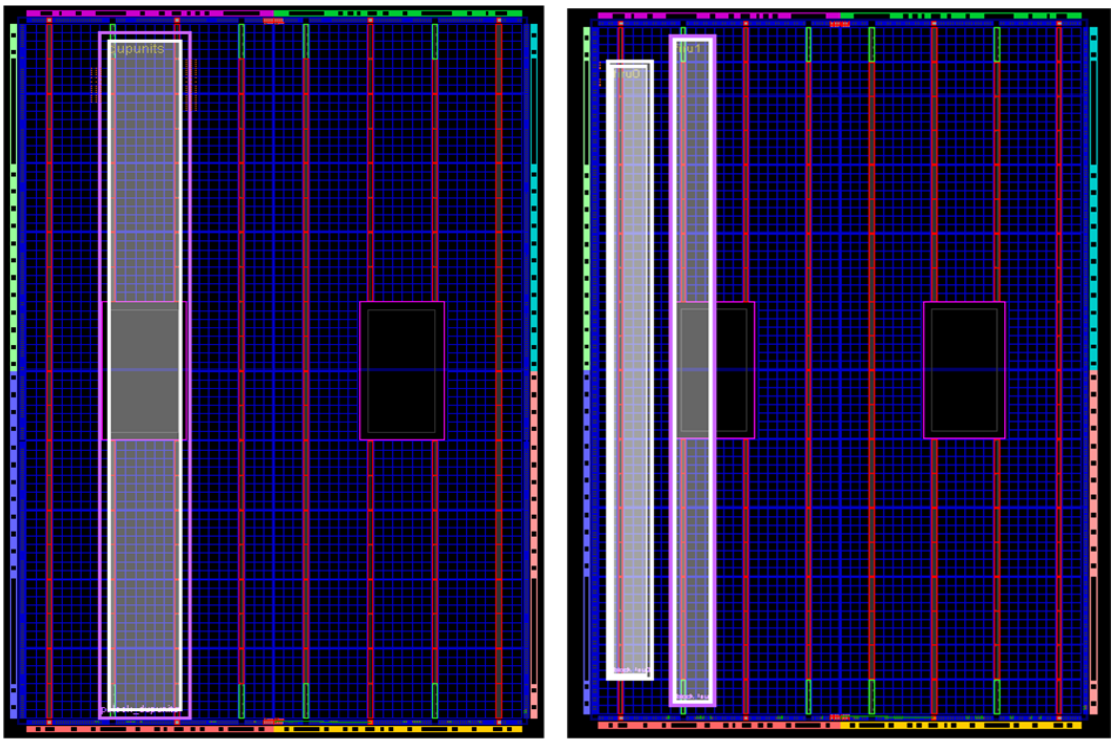
Τύπος Πόρων	**Διαθέσιμα**	**Απαιτούμενα**	**%**
PPC405	2	2	100%
LUTs	27392	3654	13%
BRAMS	136	50	36%
FF	27392	3488	12%
Slices	13696	3181	23%
DCMs	8	3	37.5%

Πίνακας 5.1 Απαιτούμενοι πόροι για την υλοποίηση της Στατικής Σχεδίασης

5.3 Υλοποίηση των Δυναμικά Αναδιατασσόμενων Φίλτρων

Για την υλοποίηση και την τοποθέτηση των δύο αντιγράφων του φίλτρου δημιουργήσαμε δύο εναλλακτικές σχεδιάσεις. Στην πρώτη δημιουργήθηκε μία μονάδα η οποία περιέχει και τα δύο φίλτρα, και τοποθετήθηκε σε μία PRR ενώ στην δεύτερη σχεδίαση δημιουργήσαμε δύο ανεξάρτητες μονάδες οι οποίες τοποθετήθηκαν σε διαφορετικές PRRs .

Στο παρακάτω σχήμα παρουσιάζουμε τις δύο αυτές σχεδιάσεις όπως υλοποιήθηκαν μέσω του Planahead .



Σχήμα 5.2 Οι δύο δυναμικά αναδιατασσόμενες σχεδιάσεις.Αριστερά τα δύο αντίγραφα τοποθετήθηκαν στην ίδια PRR. Δεξιά τα δύο αντίγραφα τοποθετήθηκαν σε διαφορετικές PRRs

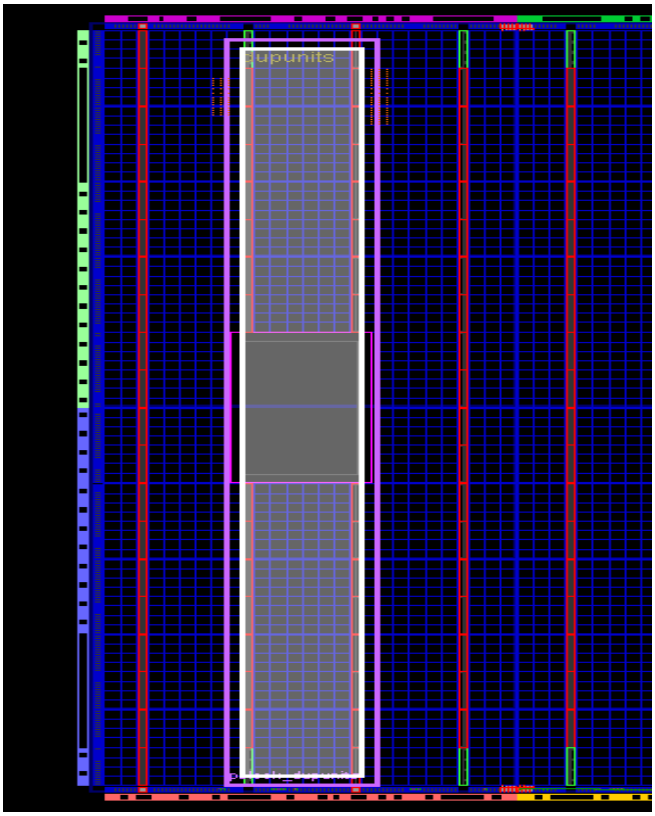
5.3.1 Τοποθέτηση σε μία PRR

Σε αυτήν την περίπτωση ορίστηκε μια μερικά αναδιατασσόμενη περιοχή η οποία βρίσκεται μεταξύ των συντεταγμένων X19 Y157 και X29 Y0. Για την επικοινωνία της περιοχής αυτής χρησιμοποιήθηκαν bus Macros που τοποθετήθηκαν αριστερά ή δεξιά της PRR ανάλογα με το αν δρομολογούν εισόδους ή εξόδους στη PRR. Εξαιτίας του γεγονότος ότι η συγκεκριμένη PRR διαθέτει πόρους όπως ενσωματωμένους πολλαπλασιαστές και BRAMs οι οποίο δεν χρησιμοποιούνται για την υλοποίηση των φίλτρων οι πόροι αυτοί αποδεσμεύτηκαν κατά την οριοθέτηση της PRR ώστε να είναι διαθέσιμοι.

Το σύνολο των πόρων της PRR καθώς και του ποσοστού το οποίο εκμεταλλευόμαστε δίνεται στον παρακάτω πίνακα.

Τύπος	**Διαθέσιμα**	**Απαιτούμενα**	**%**
LUT	4032	2888	71.63 %
FLIP- FLOP	4032	770	19.10 %
Slices	2016	1702	87.40 %

Πίνακας 5 2 Αξιοποίηση της αναδιατασσόμενης περιοχής που περιέχει και τα δύο αντίγραφα του φίλτρου.



Σχήμα 5.3 Άποψη των Bus Macros και της PRR η οποία περιέχει και τα δύο αντίγραφα του φίλτρου. Αριστερά τοποθετήθηκαν τα Bus Macros εισόδου και δεξιά τα Bus Macros εξόδου

Όπως φαίνεται στο παραπάνω σχήμα τα δύο φίλτρα τοποθετήθηκαν σε μια PRR οι οποία καταλαμβάνει το μεγαλύτερο ποσοστό των στηλών. Αυτό έγινε εξαιτίας του γεγονότος ότι η αναδιάταξη της XC2VP30 είναι column based ώστε να επιτύχουμε καλύτερους χρόνους αναδιάταξης.

Για την επικοινωνία της μερικά αναδιατασσόμενης περιοχής με την υπόλοιπη σχεδίαση, έκτος για την δρομολόγηση του ρολογιού το οποίο δεν χρειάζεται να περάσει μέσα από bus macros, για τα υπόλοιπα σήματα της διεπαφής χρησιμοποιήσαμε bus macros των οποίων η διεπαφή έχει πλάτος 8 bits και παρουσιάζονται στον Πίνακα 5.3

Τύπος Σήματος	**Αριθμός bits**	**Αριθμός Bus Macros**	**Τύπος Bus Macros**
Δεδομένα εισόδου	28	4	LeftToRight(8 bits)
Δεδομένα εξόδου	48	6	LeftToRight(8 bits)

Πίνακας 5.3 Απαιτούμενα bus macros για την επικοινωνία με την στατική σχεδίαση .

Για την συγκεκριμένη PRR δημιουργήθηκαν δύο partial bitstreams . Ένα που περιέχει την υλοποίηση των δύο φίλτρων και ένα το οποίο την αφήνει κενή (Blank bitstream) το partial bitstream το οποίο διατάσσει την περιοχή έχει μέγεθος 172 KB .

5.3.2 Τοποθέτηση σε διαφορετικές PRRs

Σε αυτήν την περίπτωση ορίστηκε μια μερικά αναδιατασσόμενη περιοχή για κάθε φίλτρο. Η περιοχή στην οποία βρίσκεται το πρώτο αντίγραφο βρίσκεται μεταξύ των συντεταγμένων X2 Y151 και X9 Y8 και η περιοχή στην οποία βρίσκεται το δεύτερο αντίγραφο βρίσκεται μεταξύ των συντεταγμένων X14 Y157 και X21 Y2. Και στις δύο περιοχές οι πόροι των BRAMs και των πολλαπλασιαστών αποδεσμεύτηκαν .

Το σύνολο των πόρων της PRR καθώς και του ποσοστού το οποίο εκμεταλλευόμαστε για τις δύο περιοχές δίνεται στους Πίνακές 5.4 και 5.5.

Τύπος	**Διαθέσιμα**	**Απαιτούμενα**	**%**
LUT	2304	1444	62.67 %
FLIP- FLOP	2304	384	16.67 %
Slices	1152	881	76.48 %

Πίνακας 5.4 Αξιοποίηση της αναδιατασσόμενης περιοχής που περιέχει το πρώτο αντίγραφο του φίλτρου.

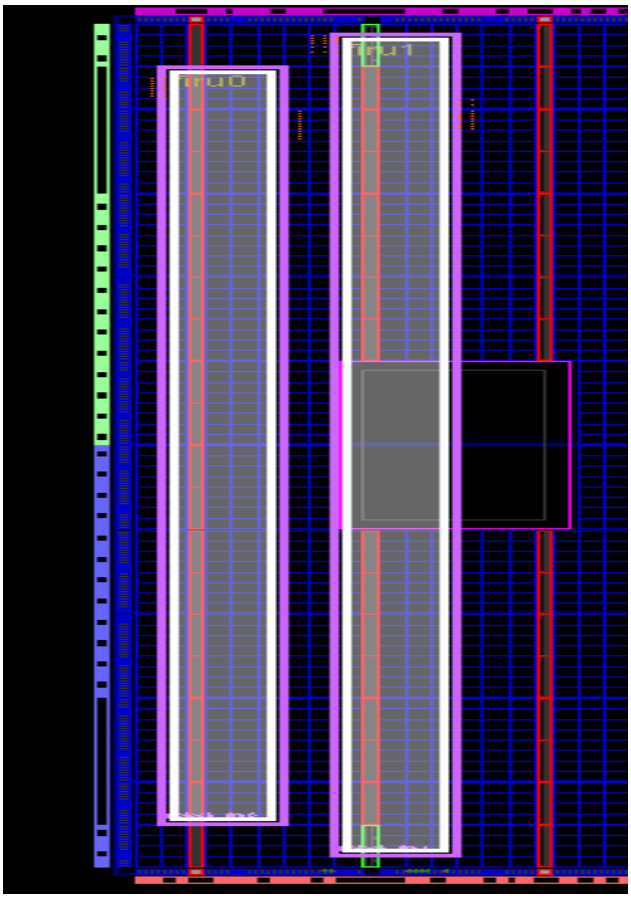
Τύπος	**Διαθέσιμα**	**Απαιτούμενα**	**%**
LUT	1984	1444	72.78 %
FLIP- FLOP	1984	384	19.35 %
Slices	992	881	88.81 %

Πίνακας 5 5 Αξιοποίηση της αναδιατασσόμενης περιοχής που περιέχει το δεύτερο αντίγραφο του φίλτρου.

Τέλος οι συνολικά διαθέσιμοι πόροι και το ποσοστό αξιοποίησης τους και για τις δύο PRR παρουσιάζεται στον Πίνακα 5.6

Τύπος	**Διαθέσιμα**	**Απαιτούμενα**	**%**
LUT	4288	2888	67.35 %
FLIP- FLOP	4288	768	17.91 %
Slices	2144	1762	82.18 %

Πίνακας 5.6 Συνολική αξιοποίηση των αναδιατασσόμενων περιοχών



Σχήμα 5.2 Άποψη των Bus Macros και των PRRs οι οποίες περιέχουν τα δύο αντίγραφα του φίλτρου

Όπως φαίνεται στο παραπάνω σχήμα τα δύο φίλτρα τοποθετήθηκαν σε PRRs οι οποίες καταλαμβάνουν το μεγαλύτερο ποσοστό των στηλών. Αυτό έγινε εξαιτίας του γεγονότος ότι η αναδιάταξη της XC2VP30 είναι column based ώστε να επιτύχουμε καλύτερους χρόνους αναδιάταξης.

Για την επικοινωνία κάθε μερικά αναδιατασσόμενης περιοχής με την υπόλοιπη σχεδίαση, έκτος για την δρομολόγηση του ρολογιού το οποίο δεν χρειάζεται να περάσει μέσα από bus macros, για τα υπόλοιπα σήματα της διεπαφής χρησιμοποιήσαμε τα παρακάτω bus macros των οποίων η διεπαφή έχει πλάτος 8 bits και παρουσιάζονται στον Πίνακα 5.7

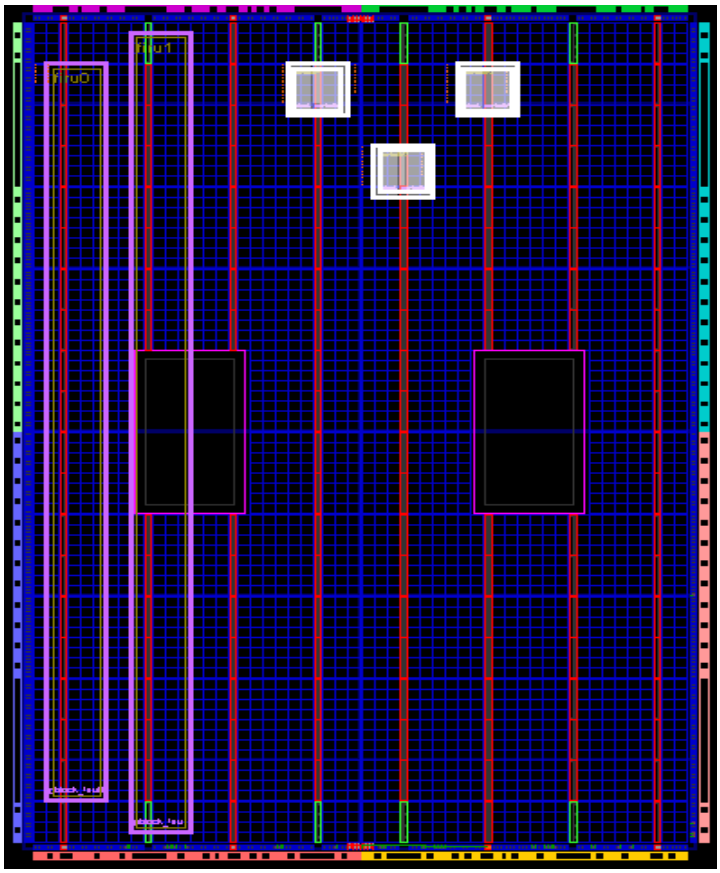
Τύπος Σήματος	**Αριθμός bits**	**Αριθμός Bus Macros**	**Τύπος Bus Macros**
Δεδομένα εισόδου	14	2	LeftToRight(8 bits)
Δεδομένα εξόδου	24	3	LeftToRight(8 bits)

Πίνακας 5.7 Απαιτούμενα bus macros για την επικοινωνία με την στατική σχεδίαση .

Αντίστοιχα με την περίπτωση που έχουμε μια PRR, εδώ δημιουργήθηκαν δύο partial bitstreams για κάθε μερικά αναδιατασσόμενη περιοχή. Το partial bitstream το οποίο αναδιατάσσει την περιοχή που περιέχει το πρώτο φίλτρο έχει μέγεθος 81 KB ενώ το δεύτερο 73 KB.

5.4 Υλοποίηση FIFO Controller

Όπως είδαμε στο προηγούμενο κεφάλαιο η FIFO λειτουργεί καθ'όλη τη διάρκεια λειτουργίας του συστήματος επομένως πρέπει να εγγυηθούμε τη σωστή λειτουργία της .Για τον λόγω αυτό ο controller της ο οποίος υλοποιήθηκε με αναδιατασσόμενους πόρους της FPGA σχεδιάστηκε με την τεχνική του TMR. Για να μπορούμε όμως να διορθώνουμε SEUs τα οποία συμβαίνουν στον controller κάθε αντίγραφο τοποθετήθηκε σε διαφορετική PRR ώστε να μπορούμε να το αναδιατάσσουμε δυναμικά ενώ τα άλλα δύο να συνεχίζουν απρόσκοπτα την λειτουργία τους.



Σχήμα 5.3 Άποψη των Bus Macros και των PRRs κάθε μία από τις οποίες προγραματίζεται με ένα αντίγραφο του FIFO Controller

Το σύνολο των αναδιατασσόμενων πόρων καθώς και του ποσοστού το οποίο του κάθε PRR το οποίο αξιοποιούμε δίνεται στον πίνακα 5.8

Τύπος	**Διαθέσιμα**	**Απαιτούμενα**	**%**
LUT	160	67	41.88 %
FLIP- FLOP	160	22	13.75 %
Slices	80	41	51.25 %

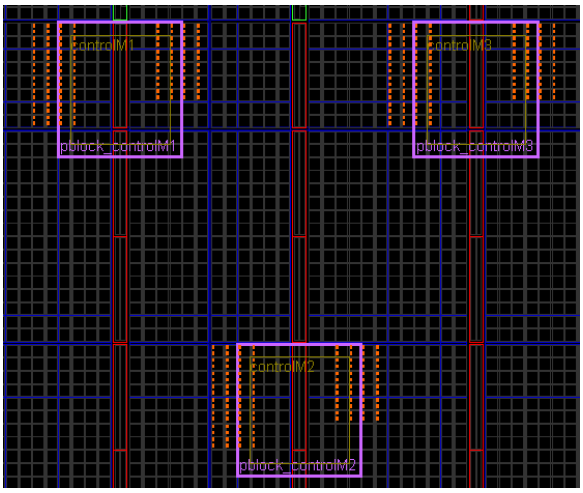
Πίνακας 5.8 Αξιοποίηση της αναδιατασσόμενης περιοχής που περιέχει το πρώτο αντίγραφο

Για την επικοινωνία κάθε μερικά αναδιατασσόμενης περιοχής με την υπόλοιπη σχεδίαση έκτος για την δρομολόγηση του ρολογιού χρησιμοποιήσαμε τα παρακάτω bus macros

Τύπος Σήματος	**Αριθμός bits**	**Αριθμός Bus Macros**	**Τύπος Bus Macros**
Δεδομένα εισόδου	28	4	LeftToRight(8 bits)
Δεδομένα εξόδου	24	3	LeftToRight(8 bits)

Πίνακας 5.9 Απαιτούμενα bus macros για την επικοινωνία με την στατική σχεδίαση .

Συνολικά και για τα τρία αντίγραφα χρειαζόμαστε 21 bus macros πλάτους 8 bits. Στο Σχήμα 5.4 παρουσιάζεται σε μεγέθυνση η περιοχή που τοποθετούνται τα τρία αντίγραφα του Controller. Ο μεγάλος αριθμός σημάτων εισόδου/εξόδου σε μία μικρή περιοχή της FPGA δημιούργησε προβλήματα κατά την διαδικασία του routing επομένως έπρεπε οι τρεις PRR να απομακρυνθούν ώστε να ολοκληρωθεί με επιτυχία η δρομολόγηση των σημάτων .



Σχήμα 5.4 Η περιοχή όπου ορίζονται οι PRRs των Controllers σε μεγέθυνση

Για κάθε μερικά αναδιατασσόμενη περιοχή δημιουργήθηκαν δύο partial bitstreams. Ένα που περιέχει την υλοποίηση του Controller και ένα το οποίο την αφήνει κενή (Blank bitstream)

.Τα partial bitstreams τα οποία αναδιατάσσουν τις περιοχές που περιέχουν το πρώτο και το δεύτερο αντίγραφο του Controller έχουν μέγεθος 80.9 KB ενώ το τρίτο έχει μέγεθος 90.3 KB. Το βάθος της FIFO είναι 1024 θέσεις.

5.5 Ανάπτυξη Λογισμικού Επεξεργαστών

Για κάθε επεξεργαστή γράφηκε ένα πρόγραμμα με το οποίο ελέγχει τα περιφερειακά του και διεκπεραιώνει της εργασίες που του έχουν ανατεθεί. Παρακάτω αναφέρουμε την λειτουργία του λογισμικού κάθε επεξεργαστή καθώς και τον τρόπο υλοποίησης του .

5.5.1 Λογισμικό PPC0

Στην περίπτωση της σχεδίασης όπου τα δύο αντίγραφα του φίλτρου τοποθετούνται στην ίδια PRR, ο PPC0 ξεκινάει την λειτουργία του, εκτελώντας polling, διαβάζει συνεχόμενα την τιμή ενός καταχωρητή ο οποίος περιέχει την έξοδο της πύλης XOR που συγκρίνει τα αποτελέσματα των φίλτρων καθώς και τα error signals του TMR Voter ο οποίος ελέγχει τα τρία αντίγραφα του FIFO Controller. Σε περίπτωση που η τιμή του καταχωρητή είναι μη μηδενική τότε αυτό σημαίνει ότι συνέβη κάποιο σφάλμα είτε στον FIFO Controller ή στα δύο φίλτρα. Σε περίπτωση σφάλματος ο PPC0 επεξεργάζεται την τιμή του καταχωρητή ώστε να εντοπίσει την περιοχή που συνέβη το σφάλμα και να την αναδιατάξει. Εάν το σφάλμα συμβεί στα φίλτρα τότε ειδοποιεί τον PPC1 ώστε να αναλάβει την επεξεργασία των δεδομένων .

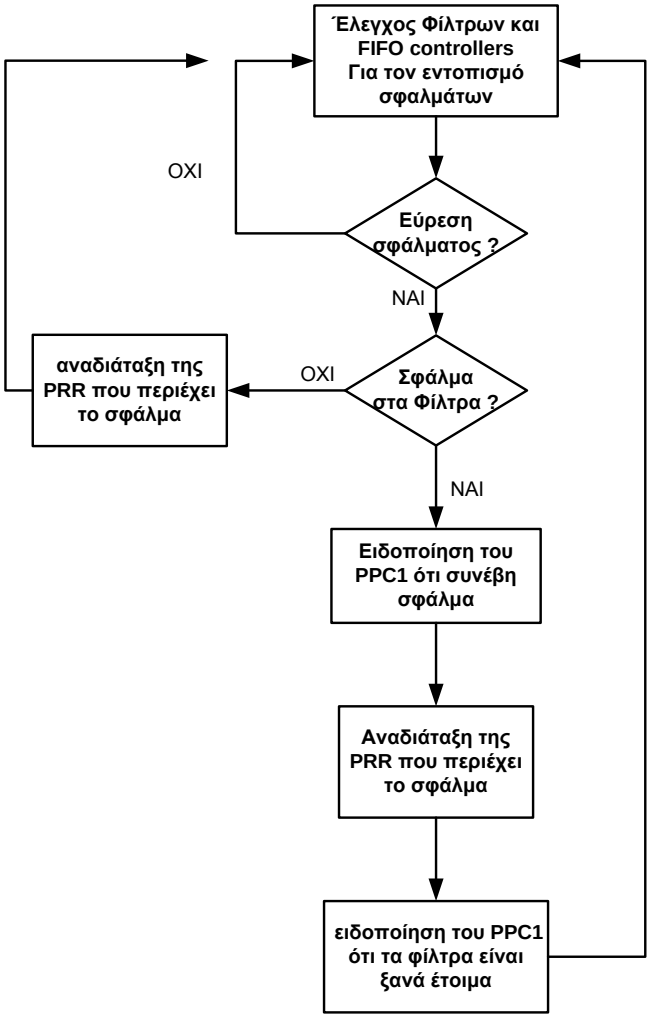
Όταν συμβεί κάποιο σφάλμα ο PPC0 αναλαμβάνει την αναδιάταξη της περιοχής στην οποία συνέβη. Η διαδικασία της αναδιάταξης μίας περιοχής έχει ως εξής :

- Ο PPC0 αρχικά ανοίγει το bitstream αρχείο της CFM το οποίο περιέχει τις πληροφορίες και τα δεδομένα για την αναδιάταξη της περιοχής και αρχικά διαβάζει το header του bitstream το οποίο περιέχει γενικές πληροφορίες για το bitstream όπως το μέγεθος του.
- Έπειτα ο PPC0 διαβάζει δεδομένα από την CFM και τα αποθηκεύει στην μνήμη του.
- Στην επόμενη φάση μεταφέρει τα δεδομένα που διάβασε από την CFM στην μνήμη του ICAP έως ώσπου να την γεμίσει .Έπειτα τα δεδομένα μεταφέρονται μέσω του ICAP στην configuration memory της FPGA.

Οι δύο τελευταίες φάσεις επαναλαμβάνονται μέχρι να μεταφερθούν όλα τα δεδομένα από την CFM στην configuration Memory .Η μεταφορά των δεδομένων για την πραγματοποίηση της μερικής αναδιάταξης είναι η παρακάτω : CFM => PPC0 => ICAP =>CM. Όταν ολοκληρωθεί

η μεταφορά των δεδομένων από την CFM στην CM ο PPC0 κλείνει το bitstream αρχείο της CFM.

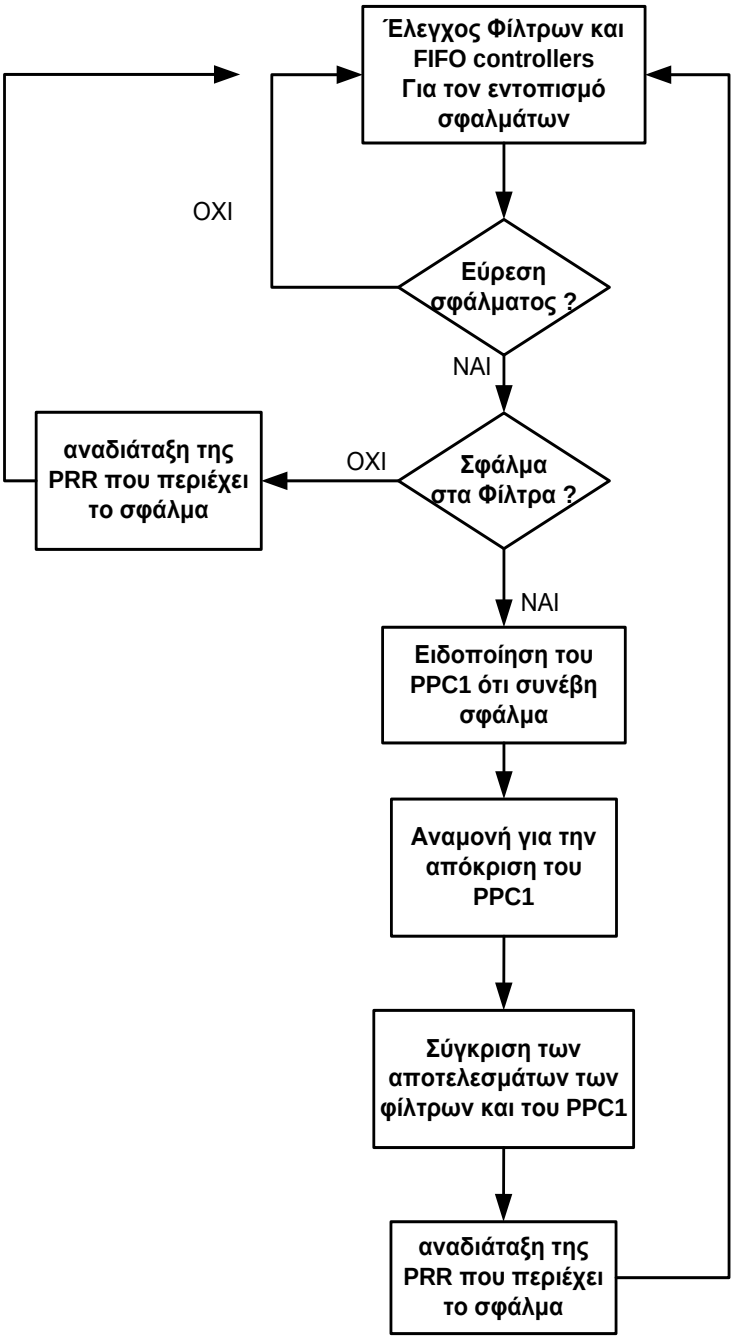
Στην περίπτωση όπου το σφάλμα συμβεί στην περιοχή που υπάρχουν τα δύο φίλτρα ο PPC0 ενημερώνει τον PPC1 να αναλάβει την επεξεργασία των δεδομένων και ο PPC0 αναλαμβάνει την αναδιάταξη της PRR. Όταν ολοκληρωθεί η αναδιάταξη των φίλτρων ο PPC0 ενημερώνει τον PPC1 ότι ολοκληρώθηκε η αποκατάσταση της εφαρμογής και ότι είναι έτοιμη να αναλάβει ξανά την επεξεργασία των δεδομένων εισόδου. Το Σχήμα 5.4 παρουσιάζει το διάγραμμα ροής του λογισμικού του PPC0.



Σχήμα 5.5 Διάγραμμα ροής του προγράμματος του PPC0

Εναλλακτικά, έχει προταθεί προηγουμένως τα δύο αντίγραφα του φίλτρου να τοποθετηθούν σε δύο διαφορετικές PRR. Σε αυτή την περίπτωση είναι εφικτό να εντοπίσουμε την μονάδα στην οποία εκδηλώθηκε το σφάλμα και να αναδιατάξουμε δυναμικά μόνο αυτήν ενώ η δεύτερη συνεχίζει να λειτουργεί κανονικά και να επεξεργάζεται τα δεδομένα. Για να

εκμεταλλευτούμε την δυνατότητα αυτή το πρόγραμμα του PPC0 έχει ελαφρώς μετατραπεί και στην περίπτωση που συμβεί σφάλμα σε ένα από τα δύο φίλτρα ειδοποιεί τον PPC1 να επεξεργαστεί μερικά δεδομένα ταυτόχρονα με τα δύο φίλτρα και να του στείλει το αποτέλεσμα ώστε να μπορέσει να εντοπιστεί η μονάδα με το σφάλμα.



Σχήμα 5.6 Διάγραμμα ροής του προγράμματος του PPC0 στην σχεδίαση όπου τα δύο φίλτρα τοποθετούνται σε διαφορετικές PRR. Ο PPC1 σε αυτή την περίπτωση επεξεργάζεται 8 δείγματα και το τελικό αποτέλεσμα το στέλνει στον PPC0 ο οποίος θα το συγκρίνει με τα

αποτελέσματα των δύο φίλτρων ώστε να αναδιατάξει την περιοχή που συνέβη το σφάλμα. Στον Πίνακα 5.10 παρουσιάζουμε τις λειτουργίες του υποσυστήματος του PPC0 και τον τρόπο με τον οποίο υλοποιούνται και στις δύο σχεδιάσεις.

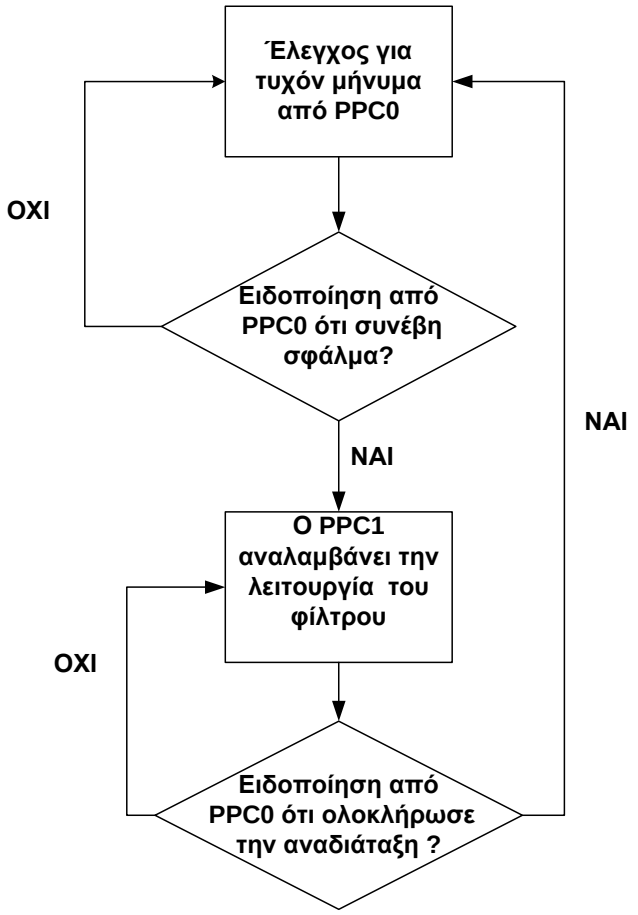
Διεργασία	**Τρόπος υλοποίησης/ εντολή**
Ανίχνευση Σφάλματος	Polling των εξόδων των μονάδων μέσω της XIo_In32()
Επικοινωνία Επεξεργαστών	Read/Write των δύο καθορισμένων περιοχών της shared memory μέσω δεικτών
Άνοιγμα αρχείου bitstream	Χρήση συνάρτησης sysace_fopen()
Μεταφορά δεδομένων από CFM προς PPC0	Χρήση συνάρτησης sysace_fread()
Μεταφορά δεδομένων από PPC0 προς BPAM ICAP	Χρήση συνάρτησης XHwIcap_StorageBufferWrite()
Εγγραφή δεδομένων στην configuration Memory	Χρήση συνάρτησης XHwIcap_DeviceWrite()

Πίνακας 5.10 Υλοποίηση διεργασιών υποσυστήματος PPC0

Στην περίπτωση αυτή ο PPC1 μπορεί να μην χρησιμοποιηθεί και ο έλεγχος τον δύο φίλτρων ώστε να εντοπιστεί το φίλτρο με το σφάλμα να γίνεται από τον PPC0. Οπότε μελλοντικά μπορεί να μην χρησιμοποιείται μειώνοντας έτσι τους πόρους του συστήματος .

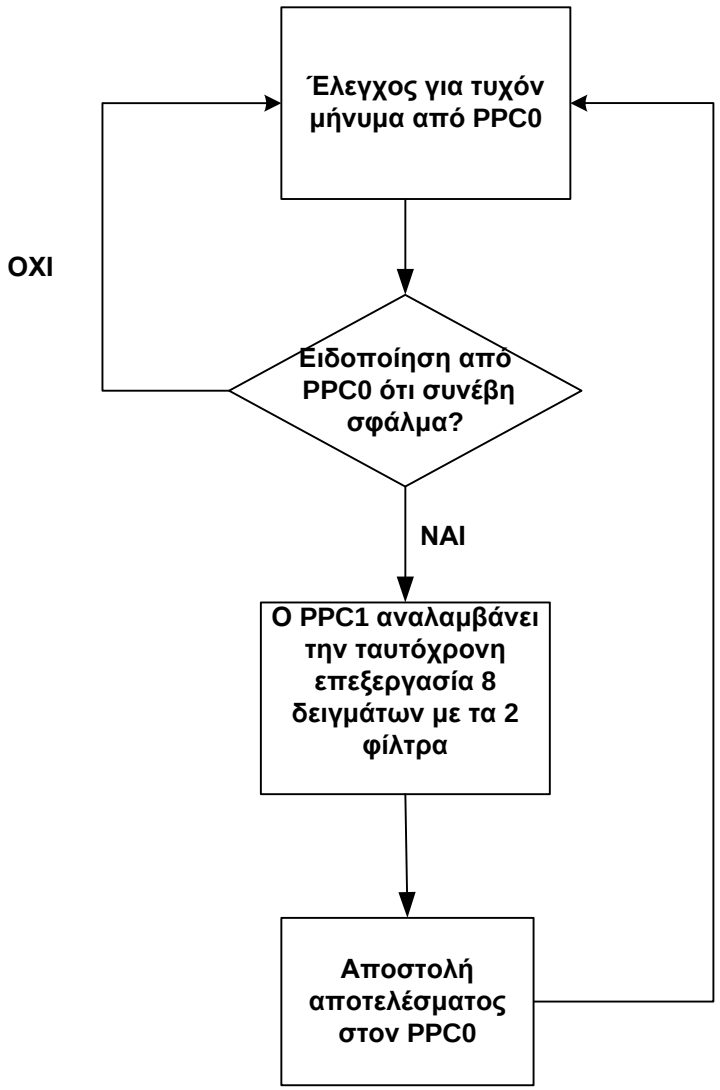
5.5.2 Λογισμικό PPC1

Στην περίπτωση της σχεδίασης όπου τα δύο αντίγραφα του φίλτρου τοποθετούνται στην ίδια PRR ο PPC1 ξεκινώντας την λειτουργία του εκτελώντας polling ελέγχει για το αν τον έχει ειδοποιήσει ο PPC0 για την ύπαρξη σφάλματος στα φίλτρα. Σε περίπτωση που έχει συμβεί κάποιο σφάλμα ξεκινάει να διαβάζει και να επεξεργάζεται τα δεδομένα εισόδου. Για να διαβάσει ο PPC1 δεδομένα από την FIFO πρέπει να ενεργοποιήσει το σήμα read της FIFO να διαβάσει τα δεδομένα και στην συνέχεια να απενεργοποιήσει το σήμα read. Για να το κάνει αυτό γράφει την δυαδική τιμή από αυτήν που ήδη έχει ο καταχωρητής της διεπαφής του περιφερειακού το οποίο επικοινωνεί με την FIFO.



Σχήμα 5.7 Διάγραμμα ροής του προγράμματος του PPC1

Όσον αφορά το πρόγραμμα του PPC1 στην περίπτωση της σχεδίασης όπου τα δύο φίλτρα τοποθετούνται σε διαφορετικές περιοχές για να εκμεταλλευτούμε τις δυνατότητες που μας προσφέρει η συγκεκριμένη σχεδίαση το πρόγραμμα του PPC1 έχει αλλάξει ώστε σε περίπτωση που λάβει μήνυμα από τον PPC0 τότε αντί να επεξεργάζεται τα δεδομένα εισόδου καθ'όλο το χρονικό διάστημα που διαρκεί η αναδιάταξη της περιοχής που έχει συμβεί SEU επεξεργάζεται συνολικά 8 δείγματα (με τον τρόπο που αναφέρθηκε προηγουμένως) και στέλνει το τελικό αποτέλεσμα πίσω στον PPC0 ώστε να το συγκρίνει με τα αντίστοιχα των δύο φίλτρων και να εντοπιστεί αυτό με το σφάλμα .



Σχήμα 5.8 Διάγραμμα ροής του προγράμματος του PPC1 όπου τα δύο φίλτρα τοποθετούνται σε διαφορετικές PRR

Στον παρακάτω πίνακα παρουσιάζουμε τις λειτουργίες του υποσυστήματος του PPC1 και τον τρόπο με τον οποίο υλοποιούνται και στις δύο σχεδιάσεις .

Εργασία	**Εντολή/Τρόπος Υλοποίησης**
Επικοινωνία Επεξεργαστών	Read/Write των δύο καθορισμένων περιοχών της shared memory
Επεξεργασία Δεδομένων	Χρήση εντολών XIo_Out32 XIo_In32 και συνάρτησης φίλτρου

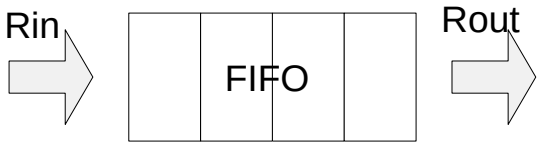
Πίνακας 5.7 Υλοποίηση διεργασιών υποσυστήματος PPC1

6 Θεωρητική Ανάλυση Συστήματος

Στο κεφάλαιο αυτό θα μελετήσουμε τις απαιτήσεις του συστήματος όσον αφορά το βάθος της FIFO. Το βάθος της όπως θα δούμε παρακάτω εξαρτάται από τον χρόνο που χρειαζόμαστε για να αναδιατάξουμε τα δύο φίλτρα από τον χρόνο που απαιτείται για να επεξεργαστούμε ένα δεδομένο μέσω ενός επεξεργαστή PPC405 καθώς και από τον ρυθμό εισόδου δεδομένων στην FIFO.

6.1 Συσχέτιση Βάθους FIFO με Ρυθμό Εισόδου/Εξόδου και Χρόνου Δυναμικής Αναδιάταξης.

Σε πολλές εφαρμογές όταν δύο συστήματα επικοινωνούν μεταξύ τους ο ρυθμός με τον οποίο στέλνει ένα πακέτο δεδομένων ο αποστολέας είναι μεγαλύτερος από τον ρυθμό με τον οποίο μπορεί ο παραλήπτης να τα απορροφήσει. Για την αποφυγή της απώλειας δεδομένων είναι απαραίτητη η εισαγωγή μίας FIFO στην μεριά του παραλήπτη όπου τα δεδομένα θα αποθηκεύονται προσωρινά μέχρι να τα διαβάσει ο παραλήπτης. Το βάθος της FIFO εξαρτάται από τρεις παράγοντες το ρυθμό αποστολής δεδομένων, το ρυθμό λήψης δεδομένων και από το συνολικό αριθμό των δεδομένων προς αποστολή.



Σχήμα 6.1 Αναπαράσταση μίας FIFO

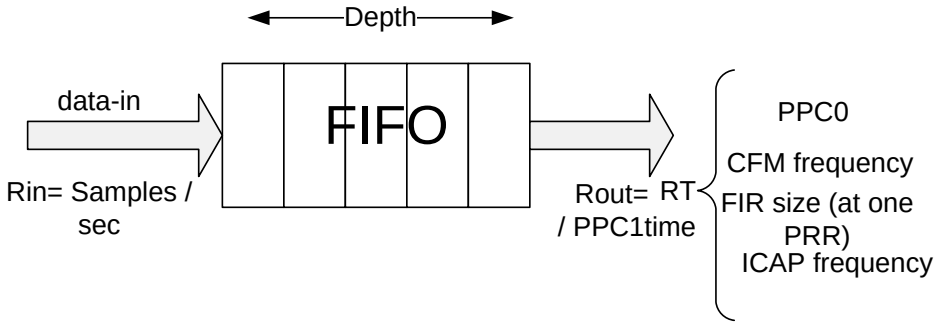
Έστω ότι ο συνολικός αριθμός των δεδομένων προς αποστολή είναι M αντίστοιχα Rin και Rout είναι ο ρυθμός εισόδου και εξόδου δεδομένων από τη FIFO. Τότε ο χρόνος που θα διαρκέσει η αποστολή των δεδομένων είναι t = M/Rin. Ο αριθμός των δεδομένων που θα προλάβει να παραλάβει ο παραλήπτης στο χρονικό διάσημα t είναι t*Rout. Επομένως το απαιτούμενο βάθος της FIFO για να αποφευχθεί η παραμικρή απώλεια δεδομένων είναι ίσος με Depth = M – t*Rout. Αντικαθιστώντας το χρόνο t καταλήγουμε στον παρακάτω τύπο Depth = M (1- Rout/Rin).

Έχει ήδη αναφερθεί ότι στην περίπτωση σφάλματος στα hardware φίλτρα στην σχεδίαση όπου αυτά είναι τοποθετημένα στην ίδια PRR το σύστημα το οποίο υλοποιήσαμε θα συνεχίσει να επεξεργάζεται τα δεδομένα εισόδου αναλαμβάνοντας την λειτουργία αυτή, όσο διαρκεί η διαδικασία αναδιάταξης των φίλτρων, ο δεύτερος επεξεργαστής. Στην περίπτωση όπου τα δεδομένα επεξεργάζονται από τα hardware φίλτρα ο ρυθμός εξόδου

μπορεί να είναι έως 100 Msamples/sec. Στην περίπτωση που συμβεί κάποιο σφάλμα λόγω της μικρότερης ταχύτητας επεξεργασίας των δεδομένων στο λογισμικό του επεξεργαστή ,ο ρυθμός εξόδου δεδομένων από την FIFO θα μειωθεί και ίσως να είναι αναγκαία η ύπαρξη μιας FIFO η οποία θα αποθηκεύει τα δεδομένα προς επεξεργασία το βάθος της οποίας εξαρτάται από το ρυθμό εισόδου δεδομένων, το ρυθμό επεξεργασίας τους μέσω του PPC1 και του χρόνου που χρειάζεται για να επανέλθει το σύστημα. Για να βρούμε την σχέση των παραπάνω μεγεθών ορίζουμε τις παρακάτω μεταβλητές:

- Rin : (samples/sec) ο επιθυμητός ρυθμός δεδομένων
- Depth : το μήκος της FIFO
- PPC1time : ο χρόνος που χρειάζεται ο PPC1 για να επεξεργαστεί ένα δεδομένο όταν έχει πέσει το φίλτρο (sec)
- RT : (restoration time) ο χρόνος που χρειάζεται για την αποκατάσταση του συστήματος (sec)

Οι μεταβλητές PPC1time και RT εξαρτώνται από την συχνότητα λειτουργίας του συστήματος καθώς και από τον τρόπο με τον οποίο υλοποιούνται οι λειτουργίες αναδιάταξης και απόκτησης δεδομένων του PPC1 από την FIFO. Συγκεκριμένα ο PPC1time εξαρτάται από την συχνότητα λειτουργίας του επεξεργαστή PPC1, από τον τρόπο επικοινωνίας των δύο επεξεργαστών, και από το hardware το οποίο χρησιμοποιείται (PLB, PLB2OPB Bridge, OPB, Socket, read fsm) ενώ ο RT εξαρτάται από την ταχύτητα μεταφοράς δεδομένων από την compact flash , το μέγεθος του partial bitstream που θέλουμε να μεταφέρουμε (FIR Size) την λειτουργία του PPC0 ο οποίος μεταφέρει τα δεδομένα από την compact flash στην BRAM του ICAP καθώς και από την συχνότητα λειτουργίας του ICAP. Τόσο ο RT όσο και ο PPC1time έχουν μετρηθεί πειραματικά και τα αποτελέσματα παρουσιάζονται στο επόμενο κεφάλαιο.



Σχήμα 6.2 Συσχέτιση του ρυθμού εξόδου από την FIFO με την απόδοση του συστήματος όταν έχει συμβεί κάποιο SEU.

Επομένως σε περίπτωση σφάλματος θεωρώντας ότι χρειάζεται να κρατήσουμε έναν συγκεκριμένο ρυθμό δεδομένων εισόδου κατά την διάρκεια όπου το σφάλμα διορθώνεται η σχέση των παραπάνω μεταβλητών που μας δίνει το ελάχιστο βάθος της FIFO που απαιτείται για τον συγκεκριμένο (ομοιόμορφα κατανεμημένο) ρυθμό εισόδου είναι η παρακάτω:

$$Depth= (Rin - 1/PPC1time) *RT \quad (6.1)$$

Ενώ θεωρώντας σταθερό το βάθος της FIFO η σχέση των παραπάνω μεταβλητών που μας δίνει τον μέγιστο δυνατό ρυθμό εισόδου δεδομένων που μπορεί να υποστηριχθεί σε περίπτωση σφάλματος και επεξεργασία των δεδομένων από τον PPC1 είναι η παρακάτω :

$$Rin = (Depth + RT / PPC1time) / RT \quad (6.2)$$

Παρόμοια στην περίπτωση της σχεδίασης όπου τα φίλτρα τοποθετούνται σε διαφορετικές PRRs ο PPC1 επεξεργάζεται 8 δεδομένα εισόδου παράλληλα με τα δύο φίλτρα ώστε να στείλει το αποτέλεσμα πίσω στον PPC0 ο οποίος το συγκρίνει με τα αντίστοιχα των δύο φίλτρων ώστε να εντοπίσει σε ποιο από τα δύο φίλτρα συνέβη το σφάλμα και να προχωρήσει στην αναδιάταξη του ενώ το φίλτρο που δεν επηρεάζεται συνεχίζει να επεξεργάζεται τα δεδομένα εισόδου. Επομένως ο ρυθμός εξόδου δεδομένων από την FIFO θα παραμείνει μειωμένος όσο χρόνο διαρκεί η παραπάνω διαδικασία και συγκεκριμένα ίσος με τον μέγιστο ρυθμό εξόδου που μπορεί να υποστηρίξει ο PPC. Σε περίπτωση σφάλματος θεωρώντας ότι χρειάζεται να κρατήσουμε έναν συγκεκριμένο αριθμό εισόδου δεδομένων η σχέση των παραπάνω μεταβλητών που μας δίνει το ελάχιστο βάθος της FIFO που απαιτείται για τον συγκεκριμένο ρυθμό εισόδου είναι η παρακάτω:

$$Depth = Rin *RT – 8 \quad (6.3)$$

Ενώ θεωρώντας σταθερό το βάθος της FIFO η σχέση των παραπάνω μεταβλητών που μας δίνει τον μέγιστο δυνατό ρυθμό εισόδου δεδομένων που μπορεί να υποστηριχθεί σε περίπτωση σφάλματος είναι η παρακάτω:

$$Rin =(Depth + 8)/RT \quad (6.4)$$

Όπου ο RT (restoration time) εξαρτάται από τον PPC1time καθώς , από το χρόνο που παίρνει η επικοινωνία των δύο επεξεργαστών καθώς και από τον αντίστοιχο χρόνο για την σύγκριση των δεδομένων. Προσεγγιστικά ο RT όπως θα δούμε και από τα πειραματικά αποτελέσματα στο επόμενο κεφάλαιο ισούται με : RT= 8 * PPC1time . Στην συγκεκριμένη σχεδίαση ο RT δεν εξαρτάται από τον χρόνο της Δυναμικής Αναδιάταξης.

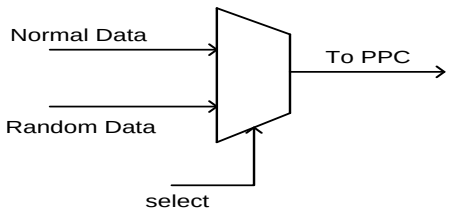
7 Πειραματικά Αποτελέσματα

Στο κεφάλαιο αυτό θα μελετήσουμε την απόδοση των σχεδιάσεων. Αρχικά θα παρουσιάσουμε τον μηχανισμό εισαγωγής λαθών και τη μεθοδολογία λήψης μετρήσεων και εισαγωγής δεδομένων στο σύστημα. Στην συνέχεια αναλύεται ο σκοπός των πειραμάτων. Τέλος το πειραματικό μέρος αποτελείται από τη μέτρηση των σταθερών παραμέτρων του συστήματος, την μέτρηση των χρόνων αναδιάταξης των FIFO controllers και των αποκλίσεων τους και την αξιολόγηση των δύο σχεδιάσεων και τη μεταξύ τους σύγκριση.

7.1 Προετοιμασία Πειραματικών Μετρήσεων

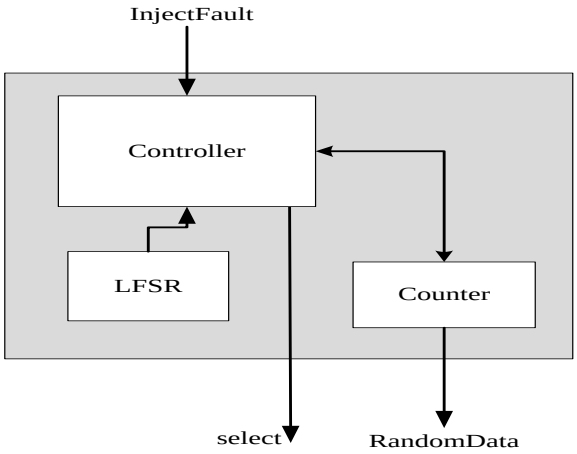
7.1.1 Μέθοδος Εισαγωγής Σφάλματος στην Σχεδίαση

Για την εισαγωγή σφαλμάτων κατά την διάρκεια λειτουργίας του συστήματος χρησιμοποιήσαμε την παρακάτω μέθοδο. Στην έξοδο κάθε μονάδας υπάρχει ένας πολυπλέκτης με τον οποίο επιλέγουμε αν θα κατευθυνθούν προς τους συγκριτές τα πραγματικά δεδομένα που παρήγαγε η μονάδα ή κάποια τυχαία δεδομένα .



Σχήμα 7.1 Επιλογή των δεδομένων τα οποία θα κατευθυνθούν προς τον PPC για σύγκριση

Για την δημιουργία των τυχαίων δεδομένων καθώς και του σήματος επιλογής του πολυπλέκτη δημιουργήσαμε μια μονάδα η οποία αποτελείται από έναν μετρητή μια FSM η οποία είναι ο ελεγκτής της μονάδας και έναν LFSR 32 bits.



Σχήμα 7 .2 Μονάδα εισαγωγής Σφαλμάτων

Όταν ενεργοποιηθεί το σήμα inject τότε ο μετρητής ξεκινά να μετρά και σταματά όταν η τιμή του LFSR γίνει 0xFFFFFFFF. Τα πρώτα bits του μετρητή τα χρησιμοποιούμε για να παράγουμε το σήμα select των πολυπλεκτών ενώ τα υπόλοιπα αποτελούν τα τυχαία δεδομένα τα οποία θα στείλουμε αντί των κανονικών. Το σήμα inject συνδέεται με ένα push button της FPGA. Από την στιγμή που θα πατηθεί κάποιο push button έως την στιγμή που θα εκδηλωθεί το σφάλμα στην αντίστοιχη μονάδα μπορεί να περάσουν το πολύ έως 43 δευτερόλεπτα. Η τυχαιότητα αυτού του χρόνου οφείλεται στην τυχαία κατάσταση στην οποία βρίσκεται ο LFSR καταχωρητής την στιγμή που θα ενεργοποιηθεί το σήμα Injectfault της μονάδας.

Για την εισαγωγή σφάλματος στις μονάδες των φίλτρων χρησιμοποιήθηκε το push button enter της αναπτυξιακής πλατφόρμας .Το μήκος του σήματος select είναι 2 bits ενώ ο μετρητής που παράγει τα τυχαία δεδομένα έχει μήκος 25 bits .Το πρώτο καθορίζει την τιμή του σήματος select ενώ τα υπόλοιπα είναι τα δεδομένα τα οποία κατευθύνονται στους πολυπλέκτες.

Για την εισαγωγή σφάλματος στις μονάδες των FIFO Controllers χρησιμοποιήθηκε το push button left της αναπτυξιακής πλατφόρμας .Το μήκος του σήματος select είναι 3 bits ενώ ο μετρητής που παράγει τα τυχαία δεδομένα έχει μήκος 24 bits.Τα δύο πρώτα καθορίζουν την τιμή του σήματος select ενώ τα υπόλοιπα είναι τα δεδομένα τα οποία κατευθύνονται στους πολυπλέκτες .

7.1.2 Μεθοδολογία Λήψης Μετρήσεων

Για την πραγματοποίηση των μετρήσεων χρησιμοποιήσαμε τον καταχωρητή time based register (TBR) που διαθέτει ο ppc405. Ο συγκεκριμένος καταχωρητής έχει μήκος 64 Bits και μετράει κύκλους ρολογιού. Για την πραγματοποίηση της μέτρησης χρησιμοποιούμε δύο μεταβλητές στις οποίες στην μεν πρώτη αποθηκεύουμε την τιμή του TBR πριν την εκτέλεση του τμήματος του κώδικα που θέλουμε να μετρήσουμε και στην δεύτερη αποθηκεύουμε την τιμή του TBR αμέσως μετά το τέλος του τμήματος. Ο αριθμός των κύκλων ρολογιού κατά τον οποίο εκτελέστηκε το τμήμα του κώδικα προκύπτει από την αφαίρεση των παραπάνω μεταβλητών. Η συχνότητα λειτουργίας του επεξεργαστή είναι 100 MHz.

7.1.3 Δεδομένα Εισόδου

Για τα δεδομένα εισόδου του συστήματος υλοποιήθηκε ένας free running counter μήκους 12 bits ο οποίος τροφοδοτεί την FIFO του συστήματος με δεδομένα. Ο ρυθμός

παραγωγής δεδομένων είναι μεταβλητός και η μέγιστη τιμή του είναι 1 δεδομένο/κύκλο ρολογιού. Η συχνότητα λειτουργία του είναι 1 0 0 MHz. Ο λόγος που δημιουργήθηκε εσωτερική γεννήτρια δεδομένων είναι η δυνατότητα επίτευξης υψηλού ρυθμού δεδομένων εισόδου (μέχρι 100 MSamples/sec , 12 bits per sample). Αποφεύχθηκε η χρήση της σειριακής θύρας λόγω του χαμηλού ρυθμού αποστολής δεδομένων που προσφέρει (μέγιστος 115200 bps περίπου ίσος με 0.009 MSamples/sec).

7.2 Σκοπός Πειραμάτων

Σκοπός των πειραμάτων είναι η μελέτη της επίδρασης της μερικής αναδιάταξης στην απόδοση του συστήματος. Ειδικότερα θα μελετήσουμε των χρόνο που χρειαζόμαστε για να αναδιατάξουμε τα φίλτρα καθώς και τον χρόνο που απαιτείται για να επεξεργαστούμε ένα δεδομένο μέσω του λογισμικού ενός επεξεργαστή PPC405. Αυτοί οι χρόνοι επηρεάζουν τον χρόνο αποκατάστασης (restoration time) του συστήματος και άρα συμφώνα με τις εξισώσεις του προηγούμενου κεφαλαίου επηρεάζουν και το μέγεθος της FIFO και τον μέγιστο ρυθμό εισόδου δεδομένων που μπορεί να υποστηρίξει το σύστημα. Ο χρόνος αποκατάστασης μελετάται και για τις δύο σχεδιάσεις. Στον πίνακα 7.1 παρουσιάζουμε της διεργασίες που επηρεάζουν τον χρόνο αποκατάστασης στις δύο σχεδιάσεις.

Διεργασία	**1 PRR**	**2 PRRs**
Χρόνος Αναδιάταξης	Ναι	Όχι
Επικοινωνία επεξεργαστών	Ναι	Ναι
Αρχικοποίηση φίλτρων	Ναι	Δεν υπάρχει
Υπολογισμός δεδομένων για την ανίχνευση της μονάδας που εκδηλώθηκε το σφάλμα	Όχι	Ναι
Σύγκριση αποτελέσματος hardware φίλτρων και PPC1	Δεν υπάρχει	Ναι

Πίνακας 7.1 Διεργασίες που επηρεάζουν τον χρόνο αποκατάστασης στις δύο σχεδιάσεις.

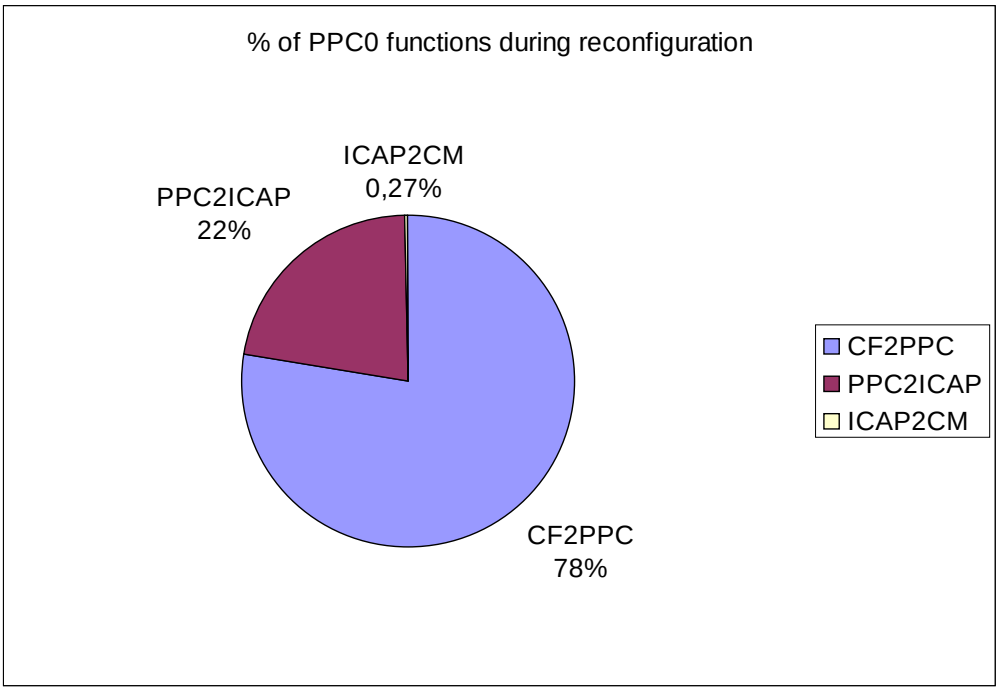
7.3 Σταθερές Παράμετροι του Συστήματος

Ανεξάρτητα από τον τρόπο με τον οποίο τοποθετούνται τα δύο φίλτρα στις PRRs υπάρχουν ορισμένες διεργασίες των οποίων ο χρόνος εκτέλεσης είναι σταθερός και στις δύο περιπτώσεις. Παρακάτω παρουσιάζουμε τις διεργασίες αυτές και τους αντίστοιχους χρόνους που απαιτούνται για την ολοκλήρωση τους.

Διεργασία	**Συνάρτηση Λογισμικού στον PPC**	**Μέσος Χρόνος**	**Απόκλιση %**
Ανταλλαγή μηνύματος μεταξύ PPC0-PPC1	Load/Store στην μνήμη	$1.687 * 10^{-3}$ ms	±8%
Χρόνος για την μεταφορά 512 bytes από CF στην μνήμη του PPC0	sysace_fread( )	1.93302 ms	±0.16%
Μεταφορά 4 bytes από τον PPC0 στην μνήμη του ICAP	XHwIcap_StorageBuffer Write( )	$4.315 * 10^{-3}$ ms	±3.1%
Μεταφορά 2048 bytes από την μνήμη του ICAP στην config. Memory της FPGA	XHwIcap_DeviceWrite( )	$2.67574 * 10^{-2}$ ms	±5.2%

Πίνακας 7.2 Απαιτούμενος χρόνος ολοκλήρωσης διεργασιών αναδιάταξης και επικοινωνίας επεξεργαστών. Οί διεργασίες της αναδιάταξης υλοποιούνται μέσω των συναρτήσεων του PPC0 ενώ η ανταλλαγή μηνυμάτων μέσω εντολών load/store των δύο επεξεργαστών.

Στο γράφημα 7.1 παρουσιάζουμε την κατανομή του χρόνου που απαιτείται για την μεταφορά 2048 Bytes από την compact flash στην configuration memory της FPGA. Από το παρακάτω γράφημα βλέπουμε ότι ο περισσότερος χρόνος της διαδικασίας αναλίσκεται στην μεταφορά των δεδομένων από την compact flash στην μνήμη του PPC επίσης ο χρόνος για να μεταφερθούν τα δεδομένα αυτά από το ICAP στην configuration memory είναι πρακτικά αμελητέος (0.27%)



Γράφημα 7.1 Κατανομή του απαιτούμενου χρόνου για την ολοκλήρωση της μεταφοράς 2048 bytes από την Compact Flash στην Configuration Memory της FPGA. Τα παραπάνω ποσοστά εξάχθηκαν σύμφωνα με τα αποτελέσματα του Πίνακα 7.1

7.4 Χρόνοι Αναδιάταξης των Μονάδων του FIFO Controller

Όπως έχει αναφερθεί προηγουμένως ο FIFO Controller αποτελείται από τρία αντίγραφα τα οποία τοποθετούνται σε διαφορετικές PRR. Στον Πίνακα 7.2 παρουσιάζουμε τον χρόνο που απαιτείται για την αναδιάταξη των τριών αυτών περιοχών.

Μονάδα FIFO Controller	**Μέσος Χρόνος Αναδιάταξης**	**Απόκλιση %**
Unit1	0.46956 sec	±0.23%
Unit2	0.46957 sec	±0.23%
Unit3	0.52014 sec	±0.21%

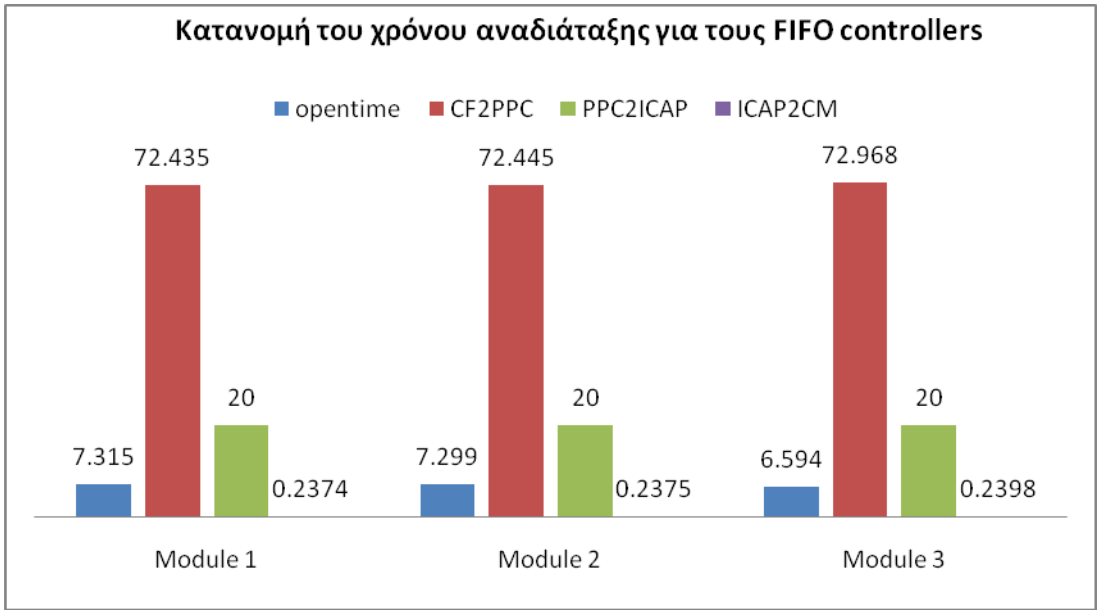
Πίνακας 7.3 Χρόνοι αναδιάταξης των PRRs οι οποίες περιέχουν τα τρία αντίγραφα του FIFO Controller

Εξαιτίας της τεχνικής του TMR ο χρόνος που απαιτείται για την διόρθωση σφαλμάτων στα αντίγραφα του FIFO controller μέσω μερικής αναδιάταξης δεν επηρεάζει την λειτουργία του συστήματος.

Όπως αναφέρθηκε και στην ενότητα 5.5.1 η διαδικασία της Δυναμικής Αναδιάταξης αποτελείται από τις παρακάτω διεργασίες :

- Άνοιγμα και κλείσιμο του bitstream αρχείο , έλεγχοι για την σωστή πραγματοποίηση των παραπάνω διαδικασιών καθώς και ανάγνωση του header του αρχείου (open/close file)
- Μεταφορά δεδομένων από την CFM στον PPC0 (CF2PPC)
- Μεταφορά δεδομένων από τον PPC0 στον ICAP (PPC2ICAP)
- Μεταφορά δεδομένων από τον ICAP στην CM (ICAP2CM)

Στο Γράφημα 7.2 παρουσιάζουμε το ποσοστό που καταλαμβάνει κάθε μία από τις παραπάνω διεργασίες έναντι του συνολικού χρόνου Αναδιάταξης για τις τρεις μονάδες του FIFO Controller.



Γράφημα 7.2 Ποσοστό του χρόνου των επιμέρους διεργασιών της Δυναμικής Αναδιάταξης, επί του συνολικού χρόνου αναδιάταξης, των περιοχών που περιέχουν τα αντίγραφα του FIFO Controller.

7.5 Απόδοση Σχεδίασης όπου τα δύο Φίλτρα Τοποθετούνται στην ίδια PRR

Στην περίπτωση όπου τα δύο φίλτρα βρίσκονται στην ίδια PRR ο restoration time (RT) δηλαδή ο χρόνος ο οποίο ς θα περάσει ώσπο υ τα δύο φίλτρα να είναι έτο μα να αναλάβουν ξανά την επεξεργασία των δεδομένων είναι ίσος με τον χρόνο αναδιάταξης της περιοχής συν τον χρόνο για την επικοινωνία των δύο επεξεργαστών συν τον χρόνο αρχικοποίησης των δύο φίλτρων. Στον Πίνακα 7.2 παρουσιάζουμε τις παραπάνω διεργασίες και τους αντίστοιχους χρόνους ολοκλήρωσης.

Διεργασία	**Χρόνος**
Αναδιάταξη φίλτρων	0.90764 sec
Επικοινωνία μεταξύ PPC0 και PPC1 (2 μηνύματα)	$3.374*10^{-6}$ sec
Αρχικοποίηση των δύο φίλτρων (επεξεργασία 7 δειγμάτων)	$19.04*10^{-6}$ sec
Restoration Time(άθροισμα των παραπάνω)	0.90767 sec
Χρόνος επεξεργασίας ενός δείγματος από PPC1 (PPC1Time)	$2.72*10^{-6}$ sec

Πίνακας 7.4 Απαιτούμενος χρόνος αποκατάστασης συστήματος (αναδιάταξη, επικοινωνία επεξεργαστών, αρχικοποίηση των φίλτρων) και επεξεργασία δειγμάτων εισόδου.

Αντικαθιστώντας τους χρόνους που υπολογίσαμε παραπάνω στην εξίσωση 6.1 παίρνουμε την παρακάτω εξίσωση:

$$Depth= (Rin - 1/2.72*10^{-6}) *0.90767$$

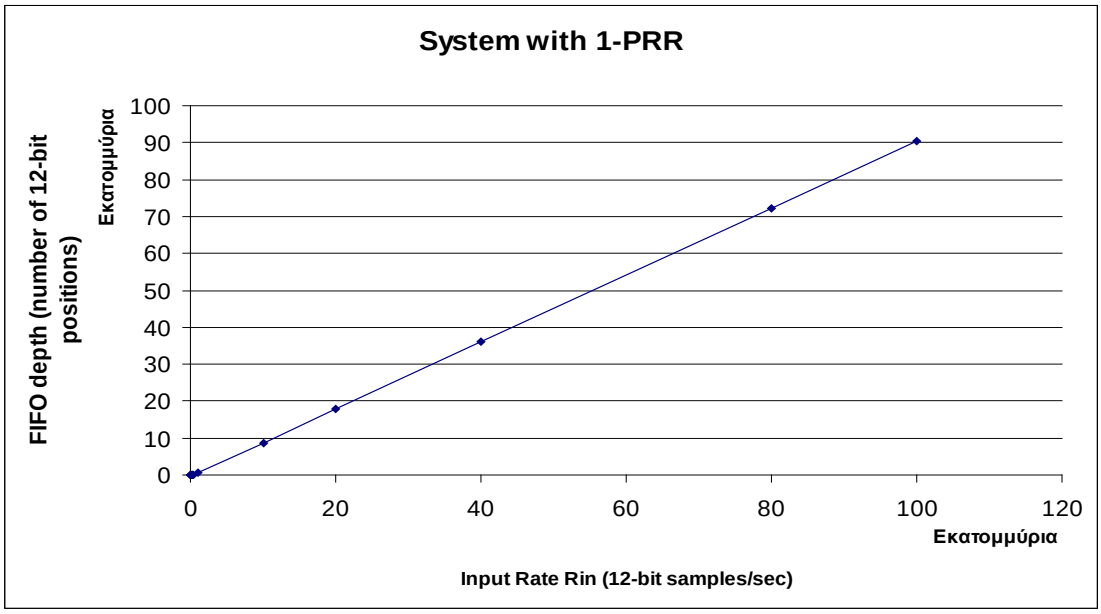
Στον Πίνακα 7.5 δίνουμε το απαιτούμενο βάθος της FIFO για διαφορετικές τιμές του ρυθμού εισόδου δεδομένων ώστε να αποφευχθεί η απώλεια δεδομένων στην περίπτωση που τα δύο αντίγραφα του φίλτρου βρίσκονται στην φάση της αναδιάταξης στην σχεδίαση όπου αυτά βρίσκονται στην ίδια PRR με βάση την παραπάνω σχέση.

Ρυθμός εισόδου δεδομένων(Rin)	**Απαιτούμενο βάθος FIFO (Depth) 12-bit positions**
5 Ksamples/sec	0
10 Ksamples/sec	0
100 Ksamples/sec	0
300 Ksamples/sec	0
367,647 Ksamples/sec	0
400 Ksamples/sec	29366
1 Msamples/sec	573968
10 Msamples/sec	8742998
20 Msamples/sec	17819698
40 Msamples/sec	35973098
80 Msamples/sec	72279898
100 Msamples/sec	90433298

Πίνακας 7.5 Αντίστοιχες τιμές ρυθμού εισόδου (Rin) και βάθος FIFO (Depth) στην σχεδίαση όπου τα δύο φίλτρα τοποθετούνται στην ίδια PRR. Στην δεξιά στήλη παρουσιάζεται το

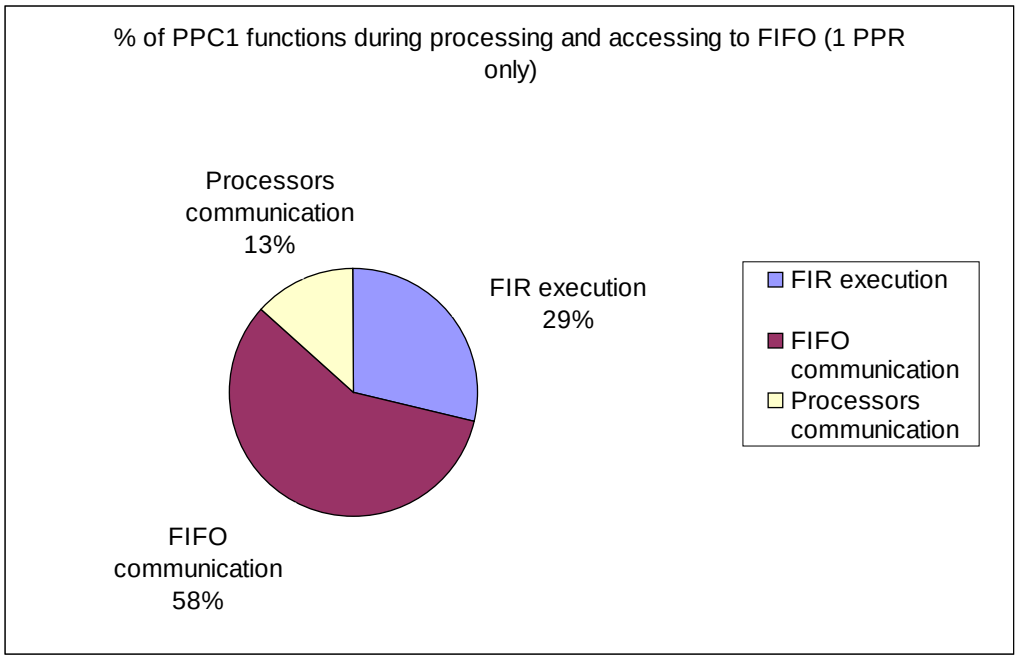
απαιτούμενο βάθος της FIFO ώστε να μην υπάρξει απώλεια δεδομένων κατά την διάρκεια αναδιάταξης της PRR, για τον αντίστοιχο ρυθμό εισόδου δεδομένων.

Στο πειραματικό σύστημα το βάθος της FIFO είναι 1024 θέσεις. Σύμφωνα με την εξίσωση 6.2 ο μέγιστος ρυθμός εισόδου δεδομένων χωρίς να έχουμε κάποια απώλεια για αυτό το βάθος είναι Rin= 368,774 Ksamples/sec το οποίο σημαίνει ένα δείγμα ανά 271,16 κύκλους ρολογιού και επειδή στην ανάλυση που πραγματοποιήσαμε θεωρούμε ότι ο ρυθμός εισόδου είναι ομοιόμορφα κατανεμημένος ο παραπάνω ρυθμός γίνεται ένα δείγμα ανά 272 κύκλους ρολογιού. Η μη απώλεια δεδομένων για τον παραπάνω ρυθμό εισόδου επιβεβαιώθηκε και πειραματικά.



Γράφημα7.3 Γραφική αναπαράσταση πίνακα 7.5

Από το παραπάνω γράφημα βλέπουμε ότι η σχέση μεταξύ βάθους FIFO και ρυθμού εισόδου δεδομένων είναι γραμμική και για μεγάλες τιμές του ρυθμού εισόδου το απαιτούμενο βάθος της FIFO γίνεται απαγορευτικά μεγάλο. Το βάθος της FIFO μπορεί να μειωθεί μειώνοντας τον χρόνο που απαιτείται για να επεξεργαστεί ένα δεδομένο ο PPC1. Ο χρόνος αυτός όπως είδαμε για την συγκεκριμένη σχεδίαση είναι ίσως με $2.72 * 10^{-6}$ sec. Στο κεφάλαιο 3 αναφέραμε ότι το HW φίλτρο μπορεί να επεξεργαστεί ένα δεδομένων κάθε κύκλο. Στην περίπτωση όπου ο ρυθμός εισόδου δεδομένων στο σύστημα είναι ο μέγιστος δυνατός (100 Msamples/sec) η ύπαρξη της FIFO των 90433298 θέσεων θα εμποδίσει την απώλεια δεδομένων όταν θα συμβεί το πρώτο σφάλμα. Όταν θα συμβεί και δεύτερο σφάλμα τότε κάποια δεδομένα εισόδου θα χαθούν. Στο γράφημα 7.4 παρουσιάζεται η κατανομή του χρόνου επεξεργασίας δεδομένων από τον PPC1 στις διεργασίες εισόδου δεδομένων και επεξεργασίας.



Γράφημα 7.4 Κατανομή του απαιτούμενου χρόνου για την ολοκλήρωση της επεξεργασίας ενός δείγματος από τον PPC1. 29% από το σύνολο των 2.72 μs (272 κύκλων) καταναλώνεται στην πραγματική επεξεργασία, 1 3 % στην επικοινωνία με τον PPC0 και 58% στην επικοινωνία με την FIFO για την πρόσβαση στα δεδομένα εισόδου

Ο παράγοντας με την μεγαλύτερη βαρύτητα στον restoration time και κατ'επέκταση στο βάθος της FIFO συμφώνα με τα πειραματικά αποτελέσματα είναι ο χρόνος αναδιάταξης της PRR που περιέχει τα δύο φίλτρα. Στο γράφημα 7.5 παρουσιάζουμε το ποσοστό του χρόνου των επιμέρους διεργασιών της Δυναμικής Αναδιάταξης επί του συνολικού χρόνου αναδιάταξης της PRR που περιέχει τα δύο αντίγραφα του φίλτρου.

Γράφημα 7.5 Ποσοστό του χρόνου των επιμέρους διεργασιών της Αναδιάταξης της PRR που περιέχει τα δύο αντίγραφα του φίλτρου.

Ο απαιτούμενος χρόνος για την αναδιάταξη της PRR μπορεί να μειωθεί χρησιμοποιώντας αντί της CF μία DDR RAM όπως φάνηκε και στο [8]. Η ταχύτητα η οποία επιτεύχθηκε κατά την μεταφορά δεδομένων από την DDR RAM στην CM είναι 5.64 MB/sec. Το μέγεθος του αρχείου το οποίο προγραμματίζει την PRR έχει μέγεθος 162 KB. Επομένως ο χρόνος αναδιάταξης με την χρήση DDR RAM εκτιμάται σε $2.81 * 10^{-4}$ sec και αντίστοιχα ο χρόνος αποκατάστασης εκτιμάται σε $3,03*10^{-4}$ sec.

7.6 Απόδοση Σχεδίασης όπου τα δύο Φίλτρα Τοποθετούνται σε Διαφορετικές PRRs

Στην περίπτωση όπου τα δύο φίλτρα βρίσκονται σε διαφορετικές PRRs ο restoration time (RT) δηλαδή ο χρόνος ο οποίος θα περάσει ώσπου το φίλτρο το οποίο δεν επηρεάζεται από το σφάλμα να αναλάβει ξανά την επεξεργασία των δεδομένων είναι ίσος με τον χρόνο επεξεργασίας 8 δειγμάτων από τον επεξεργαστή PPC1, συν τον χρόνο για την επικοινωνία των δύο επεξεργαστών συν τον χρόνο επεξεργασίας από τον PPC0 για την σύγκριση των αποτελεσμάτων των δύο φίλτρων και του αποτελέσματος του PPC1 ώστε να εντοπίσει ποιο πρέπει να αναδιατάξει. Στον Πίνακα 7.6 παρουσιάζουμε τις παραπάνω διεργασίες και τους αντίστοιχους χρόνους ολοκλήρωσης καθώς και τον χρόνο αναδιάταξης των δύο περιοχών.

Διεργασία	**Χρόνος**
Επικοινωνία μεταξύ PPC0 και PPC1 (1 μήνυμα)	$1.68*10^{-6}$ sec
Σύγκριση στον PPC0 για τον εντοπισμό του φίλτρου που επηρεάζεται από το σφάλμα	$1.09*10^{-6}$ sec
Restoration Time (άθροισμα των παραπάνω συν ο χρόνος επεξεργασίας 8 δειγμάτων)	$2.333*10^{-5}$ sec
Χρόνος επεξεργασίας δεδομένων από PPC1 (PPC1time)	$2.57*10^{-6}$ sec
Αναδιάταξη α' αντιγράφου του φίλτρου	0.4507 sec
Αναδιάταξη β' αντιγράφου του φίλτρου	0.4109 sec

Πίνακας 7.6 Απαιτούμενος χρόνος ολοκλήρωσης διεργασιών αναδιάταξης, επικοινωνίας επεξεργαστών, επεξεργασία δειγμάτων εισόδου και Restoration Time.

Αντικαθιστώντας τους χρόνους που υπολογίσαμε παραπάνω στην εξίσωση 6.3 παίρνουμε την παρακάτω:

$$\text{Depth} = \text{Rin} *2.333* 10^{-5} - 8$$

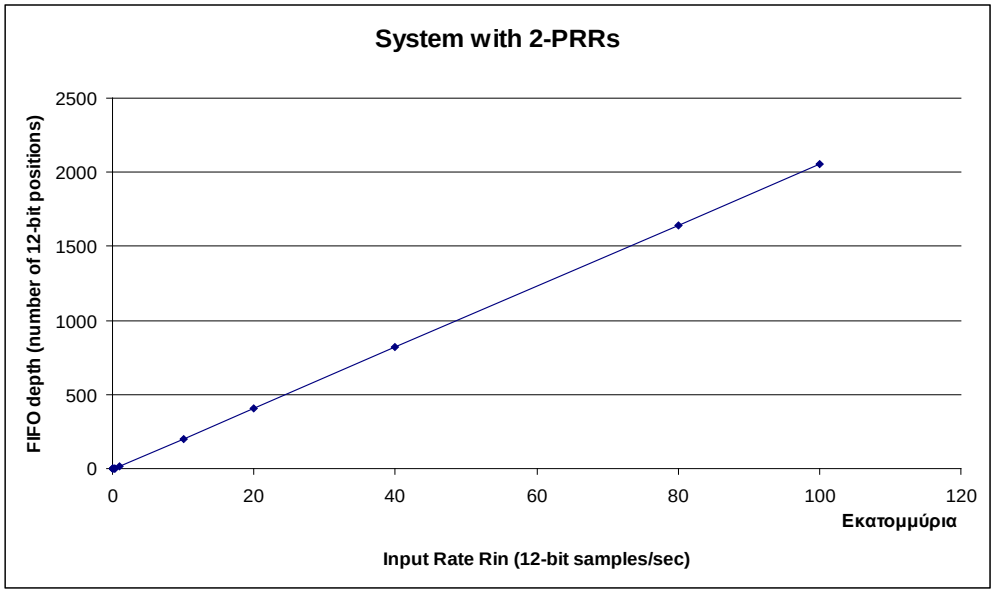
Στον παρακάτω πίνακα δίνουμε το απαιτούμενο βάθος της FIFO για διαφορετικές τιμές του ρυθμού εισόδου δεδομένων ώστε να αποφευχθεί η απώλεια δεδομένων στην περίπτωση που τα δύο αντίγραφα του φίλτρου βρίσκονται στην φάση της αναδιάταξης στην σχεδίαση όπου τα δύο φίλτρα βρίσκονται στην ίδια PRR με βάση την παραπάνω σχέση.

Ρυθμός εισόδου δεδομένων(Rin)	**Απαιτούμενο βάθος FIFO (Depth) 12-bit positions**
5 Ksamples/sec	0
10 Ksamples/sec	0
100 Ksamples/sec	0
300 Ksamples/sec	0
400 Ksamples/sec	1
1 Msamples/sec	13
10 Msamples/sec	199
20 Msamples/sec	405
40 Msamples/sec	818
80 Msamples/sec	1644
100 Msamples/sec	2057

Πίνακας 7.7 Αντίστοιχες τιμές ρυθμού εισόδου (Rin) και βάθος FIFO (Depth) στην σχεδίαση όπου τα δύο φίλτρα τοποθετούνται σε διαφορετικές PRRs. Στην δεξιά στήλη παρουσιάζεται το απαιτούμενο βάθος της FIFO ώστε να μην υπάρξει απώλεια δεδομένων κατά την διάρκεια αναδιάταξης της PRR, για τον αντίστοιχο ρυθμό εισόδου δεδομένων.

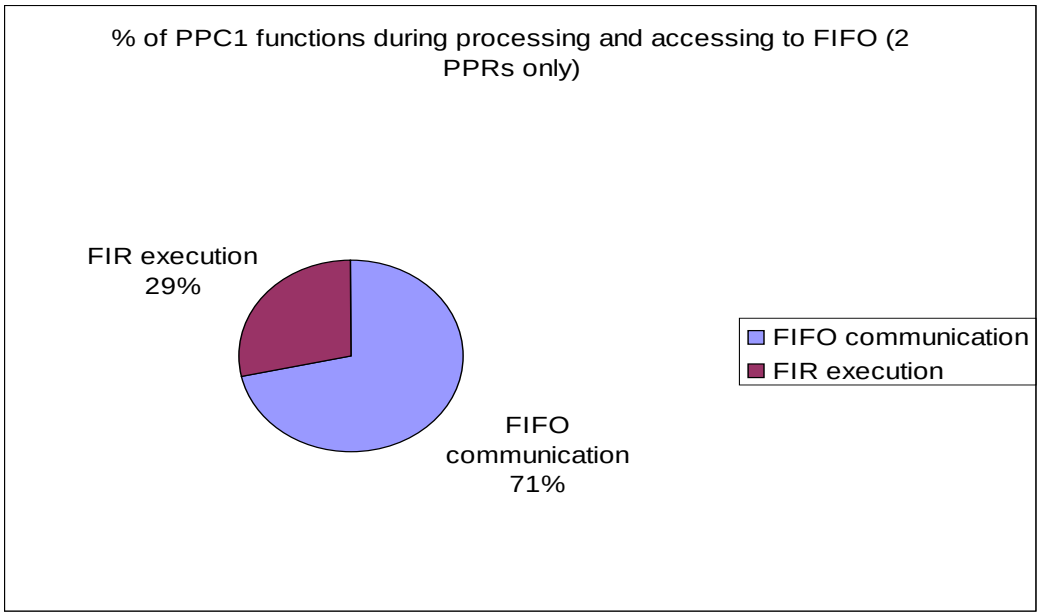
Στο πειραματικό σύστημα το βάθος της FIFO είναι 1024 θέσεις. Σύμφωνα με την εξίσωση 6.4 ο μέγιστος ρυθμός εισόδου δεδομένων χωρίς να έχουμε κάποια απώλεια για αυτό το βάθος είναι Rin= 44,632034 Msamples/sec το οποίο σημαίνει ένα δείγμα ανά sec το οποίο σημαίνει ένα δείγμα ανά 2,2,4 κύκλους ρολογιού και επειδή στην ανάλυση που πραγματοποιήσαμε θεωρούμε ότι ο ρυθμός εισόδου είναι ομοιόμορφα κατανεμημένος ο παραπάνω ρυθμός γίνεται ένα δείγμα ανά 3 κύκλους ρολογιού. Η μη απώλεια δεδομένων για τον παραπάνω ρυθμό εισόδου επιβεβαιώθηκε και πειραματικά. Στο κεφάλαιο 3 αναφέραμε

ότι το HW φίλτρο μπορεί να επεξεργαστεί ένα δεδομένων κάθε κύκλο. Στην περίπτωση όπου ο ρυθμός εισόδου δεδομένων στο σύστημα είναι ο μέγιστος δυνατός (100 Msamples/sec) η ύπαρξη της FIFO των 2057 θέσεων θα εμποδίσει την απώλεια δεδομένων όταν θα συμβεί το πρώτο σφάλμα. Όταν θα συμβεί και δεύτερο σφάλμα τότε κάποια δεδομένα εισόδου θα χαθούν.



Γράφημα 7.6 Γραφική αναπαράσταση του πίνακα 7.7

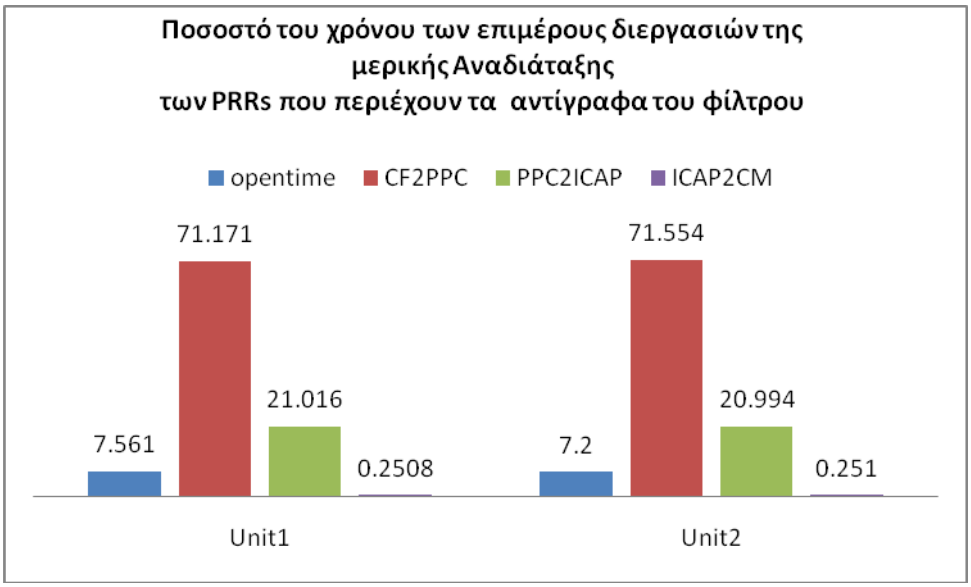
Από το παραπάνω γράφημα βλέπουμε ότι η σχέση μεταξύ βάθους FIFO και ρυθμού εισόδου δεδομένων είναι γραμμική και για μεγάλες τιμές του ρυθμού εισόδου το απαιτούμενο βάθος της FIFO παραμένει μικρό. Το βάθος της FIFO μπορεί να μειωθεί μειώνοντας τον χρόνο που απαιτείται για να επεξεργαστεί ένα δεδομένο ο PPC1. Ο χρόνος αυτός όπως είδαμε για την συγκεκριμένη σχεδίαση είναι ίσως με $2.57 * 10^{-6}$ sec. Στο γράφημα 7.7 παρουσιάζεται η κατανομή του χρόνου επεξεργασίας δεδομένων από τον PPC1 στις διεργασίες εισόδου δεδομένων και επεξεργασίας.



Γράφημα 7.7 Κατανομή του απαιτούμενου χρόνου για την ολοκλήρωση της επεξεργασίας ενός δείγματος από τον PPC1. 29% από το σύνολο των 257 μs (257 κύκλων) καταναλώνεται στην πραγματική επεξεργασία και 71% στην επικοινωνία με την FIFO για την πρόσβαση στα δεδομένα εισόδου.

Όπως παρατηρούμε και από τα πειραματικά αποτελέσματα ο PPC1time σε αυτή την σχεδίαση είναι ελαφρός μικρότερος από αυτόν της πρώτης σχεδίασης. Αυτό συμβαίνει επειδή στην σχεδίαση με την μία PRR ο PPC1 είναι υποχρεωμένος για κάθε δεδομένο που επεξεργάζεται να ελέγχει αν ο PPC0 έχει ολοκληρώσει την αναδιάταξη ώστε να τερματίσει την επεξεργασία των δεδομένων.

Η μερική αναδιάταξη σε αυτή την σχεδίαση δεν επηρεάζει την απόδοση του συστήματος. Στο γράφημα 7.8 παρουσιάζουμε το ποσοστό του χρόνου των επιμέρους διεργασιών της μερικής αναδιάταξης επί του συνολικού χρόνου αναδιάταξης των PRR που περιέχουν τα δύο αντίγραφα του φίλτρου.



Γράφημα 7.8 Ποσοστό του χρόνου των επιμέρους διεργασιών της Αναδιάταξης των PRRs που περιέχουν τα δύο αντίγραφα του φίλτρου

7.7 Σύγκριση Σχεδιάσεων

Όπως φάνηκε και από τις παραπάνω ενότητες οι δύο σχεδιάσεις διαφέρουν αρκετά ως προς το χρόνο αποκατάστασης του συστήματος. Το γεγονός αυτό οφείλεται στους διαφορετικούς παράγοντες που επηρεάζουν τις αντίστοιχες διεργασίες αποκατάστασης. Στον πίνακα 7.8 παρουσιάζουμε τις διεργασίες κάθε σχεδίασης και τους αντίστοιχους χρόνους όπως προέκυψαν πειραματικά.

Διεργασία	**1 PRR**	**2 PRRs**
Χρόνος Αναδιάταξης	0.90764 sec	-
Επικοινωνία επεξεργαστών	$3.374 * 10^{-6}$ sec	$1.68 * 10^{-6}$ sec
Αρχικοποίηση φίλτρων	$1.904 * 10^{-5}$ sec	-
Υπολογισμός δεδομένων για την ανίχνευση της μονάδας που εκδηλώθηκε το σφάλμα	-	$2.056 * 10^{-5}$ sec
Σύγκριση αποτελέσματος hardware φίλτρων και PPC1	-	$1.09 * 10^{-6}$ sec
Χρόνος Αποκατάστασης (RT)	**0.90767 sec**	**$2.333 * 10^{-5}$ sec**

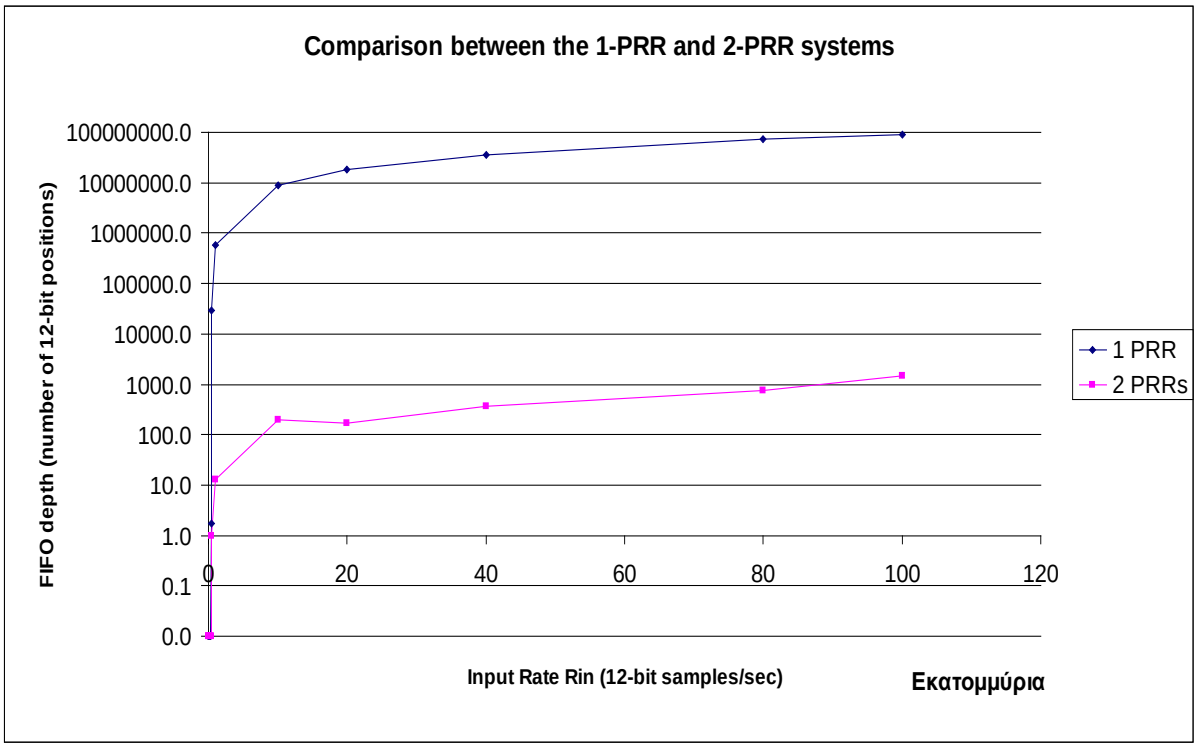
Πίνακα 7.8 Διεργασίες που επηρεάζουν τον χρόνο αποκατάστασης και οι αντίστοιχοι χρόνοι για τις δύο σχεδιάσεις

Εκτός από τον χρόνο αποκατάστασης το βάθος της FIFO το επηρεάζει τόσο ο ρυθμός εισόδου δεδομένων στο σύστημα όσο και ο χρόνος που απαιτείται για την επεξεργασία των δεδομένων από τον PPC1. Στον πίνακα 7.9 παρουσιάζουμε τις παραμέτρους που επηρεάζουν τις εξισώσης για το βάθος της FIFO.

	1PRR	2 PRRs
PPC1time(sec/samples)	$2.72*10^{-6}$ sec	$2.57*10^{-6}$ sec
Χρόνος αποκατάστασης RT (sec)	0.90767 sec	$2.333*10^{-5}$ sec
FIFO Depth (12 bits positions)	(Rin-1/PPC1time)*RT= $0.90767*Rin-3.33703*10^{5}$	Rin*RT-8= $2.333*10^{-5}*Rin$ -8

Πίνακας 7.9 Παράμετροι που επηρεάζουν τις εξισώσεις για το βάθος της FIFO για τις δύο σχεδιάσεις και σύγκριση του βάθους της FIFO.

Όπως βλέπουμε και από τον παραπάνω πίνακα ο πολλαπλασιαστικός παράγοντας του ρυθμού εισόδου δεδομένων στην σχεδίαση με την μία PRR είναι 5 τάξεις μεγέθους μεγαλύτερος από τον αντίστοιχο παράγοντα στην σχεδίαση με τις δύο PRRs. Από αυτό και μόνο μπορούμε να διαπιστώσουμε την μεγάλη διαφορά των δύο συστημάτων σε απαιτήσεις μνήμης για την FIFO. Στο Γράφημα 7.9 συγκρίνουμε τις απαιτήσεις των δύο σχεδιάσεων όσον αφορά το βάθος τις FIFO για διάφορες τιμές του ρυθμού εισόδου δεδομένων (Rin).



Γράφημα 7.9 Σύγκριση των δύο σχεδιάσεων ως προς το βάθος της FIFO (ο άξονας Y είναι σε λογαριθμική κλίμακα).

8 Συμπεράσματα και Μελλοντική Εργασία

Στην παρούσα διπλωματική δημιουργήθηκε και αξιολογήθηκε ένα αυτόνομο σύστημα για τον εντοπισμό και την διόρθωση SEUs στην Configuration Memory της FPGA μέσω δυναμικής αναδιάταξης. Αν και η εργασία μας αναπτύχθηκε πάνω στην XC2VP30 FPGA εντούτοις υπάρχουν radiation tolerant FPGAs της Xilinx οι οποίες διαθέτουν ενσωματωμένους επεξεργαστές τις οποίες θα μπορούσαμε να χρησιμοποιήσουμε.

Όσον αφορά το σύστημα των επεξεργαστών είδαμε ότι η έκδοση των εργαλείων υλοποίησης επιβάλει σημαντικούς περιορισμούς στην υλοποίηση του συστήματος όπως στην επικοινωνία των επεξεργαστών καθώς και στην σύνδεση των περιφερειακών των δύο επεξεργαστών κυρίως μέσω του OPB BUS. Τα παραπάνω έχουν ως αποτέλεσμα την πτώση της απόδοσης του συστήματος τόσο στην επικοινωνία των δύο επεξεργαστών όσο και στην επικοινωνία με την FIFO όσο και στην επεξεργασία των δεδομένων εισόδου μέσω λογισμικού. Η χρησιμοποίηση νεότερων εκδόσεων των εργαλείων για την υλοποίηση του συστήματος των επεξεργαστών θα προσφέρει σημαντικά οφέλη εφόσον θα μειώσει τον χρόνο επικοινωνίας των επεξεργαστών με την FIFO μειώνοντας τον PPC1time. Επιπλέον θα καταστήσει εφικτή την χρησιμοποίηση xps_mailboxes για την επικοινωνία των επεξεργαστών αποφεύγοντας την σπατάλη BRAMs και καθιστώντας εφικτή την χρησιμοποίηση interrupts με αποτέλεσμα να μειωθεί ο χρόνος επικοινωνίας των δύο επεξεργαστών.

Σχετικά με την μερική αναδιάταξη μέσω ICAP σύμφωνα με τα πειραματικά αποτελέσματα του κεφαλαίου 7 είδαμε ότι είναι μία αρκετά χρονοβόρα διαδικασία και συγκεκριμένα η μεταφορά δεδομένων από την CF στον PPC καταλαμβάνει σε όλες τις περιπτώσεις το μεγαλύτερο ποσοστό (σχεδόν 72 % του συνολικού χρόνου αναδιάταξης) γεγονός το οποίο στην σχεδίαση με την μία PRR έχει σοβαρό αντίκτυπο στην απόδοση του συστήματος εφόσον ο χρόνος αναδιάταξης προστίθεται στον restoration time. Για την μείωση του χρόνου της μερικής αναδιάταξης μπορούμε να χρησιμοποιήσουμε τις παρακάτω προτάσεις

- DMA Controller για την απευθείας μεταφορά δεδομένων χωρίς την μεσολάβηση του επεξεργαστή.
- Χρήση DDR RAM έναντι της Compact Flash για την αποθήκευση των Partial Bitstreams.
- Αύξηση της λειτουργίας της συχνότητας λειτουργίας του ICAP μέσω της δημιουργίας καινούργιας διεπαφής.
- Διάσπαση των δυναμικά αναδιατασσόμενων περιοχών σε μικρότερες ώστε σε περίπτωση σφάλματος να μεταφέρουμε στην CM μικρότερο αριθμό δεδομένων.

Η βασική υπόθεση που κάναμε κατά την διάρκεια υλοποίησης της διπλωματικής είναι ότι το σύστημα των επεξεργαστών δεν επηρεάζεται από σφάλματα. Κάτι τέτοιο όμως δεν είναι αληθές αφού μπορούν να συμβούν αλλοιώσεις στα δεδομένα των επεξεργαστών Επίσης χρησιμοποιούνται αναδιατασσόμενοι πόροι για την δημιουργία των περιφερειακών των επεξεργαστών όπως και bits της CM για την παραμετροποίηση των ίδιων των επεξεργαστών. Επομένως το σύστημα των επεξεργαστών δεν παρουσιάζει στο σύνολο του ανοσία από SEUs. Όπως είδαμε στην περίπτωση της σχεδίασης όπου χρησιμοποιούνται δύο PRRs για τα φίλτρα ο δεύτερος επεξεργαστής μπορεί και να μην χρησιμοποιηθεί. Το γεγονός αυτό μας δίνει την δυνατότητα να τον χρησιμοποιήσουμε ώστε να υλοποιήσουμε το σύστημα των επεξεργαστών με την μέθοδο του Duplication όπως έχει ήδη γίνει σε δύο παλαιότερες εργασίες [36,37].

Τέλος σε περίπτωση που αλλάξει η εφαρμογή το λογισμικό στο PPC1 θα αλλάξει το οποίο μπορεί να έχει ως αποτέλεσμα αυξημένες απαιτήσεις στην boot μνήμη του PPC1 όπως και στο πλάτος της FIFO(ανάλογα με τα δεδομένα εισόδου). Επιπλέον φυσιολογικά θα αλλάξει και ο χρόνος επεξεργασίας δεδομένων μέσω λογισμικού κάτι το οποίο με τη σειρά του θα μεταβάλει το βάθος της FIFO και τον μέγιστο ρυθμό εισόδου δεδομένων στο σύστημα. Η αρχιτεκτονική του στατικού μέρους (δηλαδή τα περιφερειακά των επεξεργαστών εκτιμούμε ότι θα παραμείνει) εκτιμούμε ότι θα παραμείνει σταθερή εκτός του γεγονότος ότι μπορεί να χρειαστούν μερικοί επιπλέον καταχωρητές για την υλοποίηση της διεπαφής μεταξύ επεξεργαστών και εφαρμογής που εκτελείται στο hardware. Στον Πίνακα 8.1 παρουσιάζονται radiation tolerant FPGAs της Xilinx καθώς και το ποσοστό το οποίο εκτιμούμε ότι θα καταλαμβάνει σε αυτές η στατική σχεδίαση ώστε να ξέρουμε εκ των προτέρων αν η συνολική σχεδίαση θα χωράει στην FPGA.

FPGA	**Διαθέσιμος # PPCs**	**Διαθέσιμος # Slices**	**Εκτίμηση % Static**
XQ5VFX100T	2	16000	9,9%
XQ5VFX130T	2	20480	7,76%
XQ4VFX60	2	25280	12,58%
XQ4VFX100	2	42176	7,54%
XQ2VP40	2	19392	16,4%
XQ2VP70	2	33088	9,61%
XQR4VFX60	2	25280	12,58%
XQR4VFX140	2	63168	5,03%

Πίνακας 8.1 Εκτίμηση ποσοστού δέσμευσης πόρων του στατικού μέρους του συστήματος σε εξειδικευμένες FPGAs για την ανάπτυξη εφαρμογών αντοχής σφαλμάτων

Ενδεικτικά προχωρήσαμε σε place and route του στατικού μέρους του συστήματος για την XC4VFX60 FPGA. Στον Πίνακα 8.2 παρουσιάζουμε το ποσοστό δέσμευσης πόρων για την συγκεκριμένη FPGA.

Τύπος	**Διαθέσιμα**	**Απαιτούμενα**	**Αξιοποίηση %**
LUTs	50560	3690	7 %
FLIP- FLOPs	50560	3206	6 %
Slices	25280	3089	12 %
Rams	232	50	21%

Πίνακας 8.2 Ποσοστό δέσμευσης πόρων για την XC4VFX60 FPGA από το στατικό μέρος του συστήματος.

Αναφορές

[1] Xilinx Inc, XAPP290(v2.0) Difference-Based Partial Reconfiguration, Application Note, December 3, 2007,
http://www.xilinx.com/support/documentation/application_notes/xapp290.pdf

[2] Xilinx Inc , 2005 Xilinx
http://www.xilinx.com/itp/xilinx7/books/data/docs/dev/dev0038_8.html

[3] F. Lima Kastensmidt, L. Carro and R. Reis,"FAULT-TOLERANCE TECHNIQUES FOR SRAM-BASED FPGAS" Published by Springer, ISBN-10 0-387-31068-1.

[4] Xilinx Inc, Xilinx Defense and Aerospace Presentation,
http://www.xilinx.com/publications/prod_mktg/MilAero.pdf

[5] A. Anifantis ,"Performance Evaluation of Dynamic Reconfiguration in ModernField Programmable Gate Arrays", Diploma Thesis. Technical report, ECE Dpt.,Technical University of Crete, 2007.

[6] C. Effraimidis, "A Self Reconfigurable Architecture To Support Multiple Fitness Functions In Genetic Algorithm", Diploma Thesis. Technical report, ECE Dpt.,Technical University of Crete, 2009.

[7] Γ. Νικολουδάκης, "Σχεδίαση και υλοποίηση με δυναμικά αναδιατασσόμενη λογική συστήματος για υλοποίηση πολλαπλών κρυπτογραφικών αλγορίθμων", Διπλωματική εργασία, Τμήμα ΗΜΜΥ, Πολυτεχνείο Κρήτης, Χανιά Ιούλιος 2009

[8] E. Spanakis, "Design and Implementation of a Linux-basedDynamic Reconfiguration Task manager in FPGAs", Diploma Thesis, Technical report, ECE Dpt.,Technical University of Crete, 2009.

[9] K. Papadimitriou and A. Dollas, "Performance Evaluation of a Preloading Model in Dynamically Reconfigurable Processors", IEEE Conference on Field Programmable Logic and Applications (FPL '06), pp. 901-904, Madrid, Spain, August, 2006

[10] K. Papadimitriou, A. Anyfantis, A. Dollas, "An Effective Framework to Evaluate Dynamic Partial Reconfiguration in FPGA Systems", IEEE Transactions on Instrumentation and Measurement 2009.

[11]C. Claus, F. Muller, J. Zepenfeld, W. Stechele, "A new framework to accelerate Virtex-II Pro dynamic Partial self-reconfiguration", 14th Reconfigurable Architectures Workshop, Long Beach, CA, March 26-27, 2007.

[12] F. HATORI, T. SAKURAI, K. SAWADA, M. TAKAHASHI, M. ICHIDA, M. UCHIDA, I. YOSHII, Y. KAWAHARA, T. HIBI, Y. SAEKI, H. MUROGA, A. TANAKA, AND K. KANZAKI, "Introducing redundancy in field programmable gate arrays. In *Proceedings of the IEEE Custom Integrated Circuits Conference"*. IEEE Computer Society Press, Los Alamitos, CA, 7.1.1–7.1.4, 1993.

[13] J. L. KELLY, AND P. A. IVEY, "A novel approach to defect tolerant design for SRAM based FPGAs, In *Proceedings of the ACM International Workshop on Field Programmable Gate Arrays"*, ACM, New York, pp 1–11, 1994.

[14] S. DURAND, AND C. PIGUET, “FPGA with selfrepair capabilities”, In *Proceedings of the ACM International Workshop on Field Programmable Gate Arrays*. ACM, New York, pp 1–6, 1994.

[15] N. J. HOWARD, A. M. TYRELL, AND N. M. ALLINSON . “The yield enhancement of field programmable gate arrays”. *IEEE Trans. VLSI Syst. 2*, 1 (Mar.), pp 115 –123, 1994.

[16] F. HANCHEK AND S. DUTT, “Design methodologies for tolerating cell and interconnect faults in FPGAs”, In *International Conference on Computer Design*, pp 326–331, 1996a.

[17] A. DOUMAR AND H. ITO, “Defect and fault tolerance SRAM-based FPGAs by shifting the configuration data”. *IEICE Trans. Inf. Syst, pp* 1104–1115, 2000.

[18] K. Κυριακουλάκος, “Toward Efficient TMR Support For SRAM –Based FPGAs”, MSC Thesis, ECE Dpt.,Technical University of Crete, 2009.

[19] J**.** Emmert , C. Stroud, B. Skaggs, M. Abramovici, “Dynamic Fault Tolerance in FPGAs via Partial Reconfiguration”, IEEE Symposium on Field-Programmable Custom Computing Machines (fccm), pp.165, 2000.

[20] D. Das and N. A. Touba “A Low Cost Approach for Detecting, Locating, and Avoiding Interconnect Faults in FPGA-Based Reconfigurable Systems”, VLSI Design, Proceedings, Twelfth International Conference, pp 266-269, Jan 1999.

[21] C. Metra, G. Mojoli, S. Pastore, D. Salvi and G. Sechi, “Novel Technique for Testing FPGAs” Design, Automation, and Test in Europe .Proceedings of the conference on Design, automation and test in Europe, pp 89 – 95, 1998.

[22] R. Andraka and J. Brady, “A Low Complexity Method for Detecting Configuration Upset in SRAM Based FPGAs”, Proceedings of the 2002 Military and Aerospace Applications of Programmable Devices and Technologies Conference, Sept 10-12, 2002.

[23] C. Bolchini, A. Miele, M. D. Santambrogio, “TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs” Proceedings of the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, pp 87-95, 2007.

[24] Xilinx Inc , Xilinx TMRTool ,http://www.xilinx.com/ise/optional_prod/tmrtool.htm

[25] C. Bolchini, D. Quarta, M. D. Santambrogio, “SEU Mitigation for SRAM-Based FPGAs through Dynamic Partial Reconfiguration”, Great Lakes Symposium on VLSI Proceedings of the 17th ACM Great Lakes symposium on VLSI, pp 55 – 60, 2007

[26] M. Gokhale, P. Graham, E. Johnson, N. Rollins, and M. Wirthlin “Dynamic Reconfiguration for Management of Radiation-Induced Faults in FPGAs”, International Journal of Embedded Systems , Volume 2, pp. 28 – 38, 2006.

[27] G. Asadi, M. B. Tahoori, “Soft Error Rate Estimation and Mitigation for SRAMBased FPGAs” International Symposium on Field Programmable Gate Arrays. Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays, pp 149-160, 2005.

[28] F. Lima, L. Carro, R. Reis, "Designing Fault Tolerant Systems into SRAM-based FPGAs,",40th Design Automation Conference (DAC'03), pp.650, 2003.

[29] M. S. Yarandi, A. Alaghi, Z. Navabi, “An Optimized BIST Architecture for FPGA Look-Up Table Testing”, IEEE Computer Society Annual Symposium on VLSI: Emerging VLSI Technologies and Architectures (ISVLSI'06), pp.420-421, 2006.

[30] M. Renovell, J. M. Portal , J. Figueras, Y. Zorian, “Testing the Interconnect of RAM-Based FPGAs”, IEEE Design and Test of Computers, vol. 15, no. 1, pp. 45-50, Jan.-Mar. 1998.

[31] Wei-Je, Huang and E. J. McCluskey, “Column-Based Precompiled Configuration Techniques for FPGA Fault Tolerance”, Field-Programmable Custom Computing Machines, FCCM '01. The 9th Annual IEEE Symposium, pp 137 – 146, 2001.

[32] J. LACH, W. H. MANGIONE-SMITH, AND M. POTKONJAK, “Algorithms for efficient runtime fault recovery on diverse FPGA architectures”, In *Proceedings of the International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp.385–394, 1999.

[33] J. M. Emmert and D. Bhatia, “Partial Reconfiguration of FPGA Mapped Designs with Applications to Fault Tolerance and Yield Enhancement”, Field-Programmable Logic and Applications, 7th International Workshop, pp 141-150, 1997.

[34] J. M. Emmert and D. Bhatia, “Incremental Routing in FPGAs”, ASIC Conference, Proceedings,Eleventh Annual IEEE International, pp. 217 – 221, 13-16 Sept. 1998.

[35] S. Dutt, V. Shanmugavel, S. Trimberger, “Efficient Incremental Rerouting for Fault Reconfiguration in Field Programmable Gate Arrays” Computer-Aided Design,Digest of Technical Papers.IEEE/ACM International Conference on Volume, Issue, pp. 173 – 176, 1999.

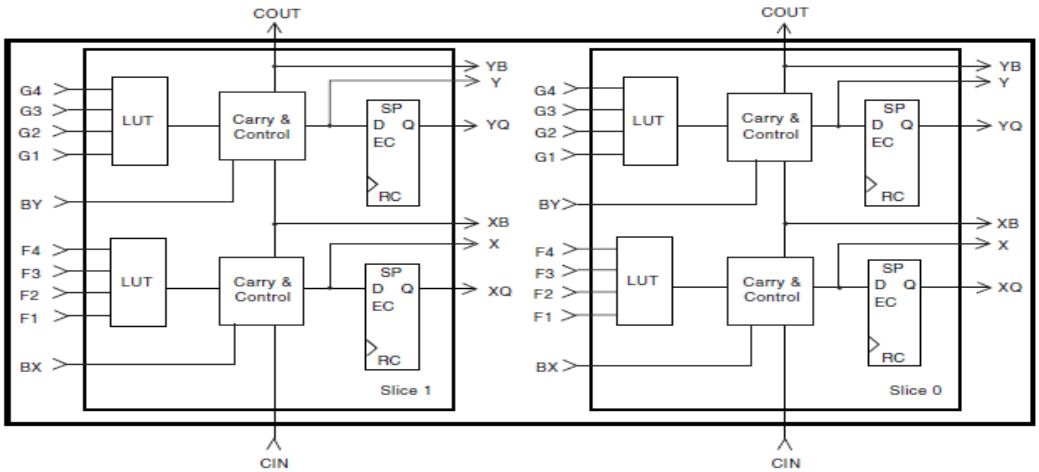
[36] W. Mandy, G. S. Bolotin, **“**IBM PowerPC 405 SEU Mitigation Using Processor Voting Techniques in Xilinx Virtex-I1 Pro FPGA” NASA Center: Jet Propulsion Laboratory. Publication Year: 2004

[37] Xilinx Inc, XAPP564(v1.0.2) PPC405 Lockstep System on ML310, Application Note, January 29,2007
http://www.xilinx.com/support/documentation/application_notes/xapp564.pdf

ΠΑΡΑΡΤΗΜΑ Α

Η δομή των FPGAs της σειράς Virtex αποτελούνται από δυο βασικά στοιχεία. Μπλοκ διατασσόμενης λογικής (CLBs) και μπλοκ εισόδου εξόδου input/output (IOBs). Τα CLBs διατάσσονται σε στήλες σχηματίζοντας έναν δυσδιάστατο πίνακα. Μεταξύ των CLBs υπάρχουν κανάλια δρομολόγησης όπου μέσω πινάκων δρομολόγησης (GRM) οι οποίοι αποτελούνται από διακόπτες τοποθετημένους στις τομές οριζοντίων και καθέτων καναλιών δρομολόγησης μπορούν να συνδεθούν μεταξύ τους

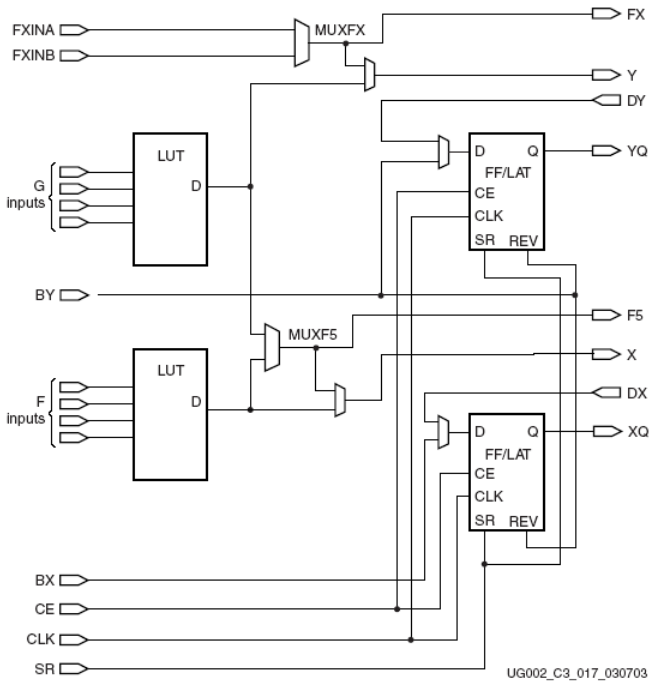
Τα CLBs αποτελούνται από δυο πανομοιότυπα μέρη (slices) από τα οποία έκαστο δομείται από δυο LUTs τεσσάρων εισόδων δυο στοιχεία μνήμης και από λογική carry out



Σχήμα 1 Δομή ενός CLB

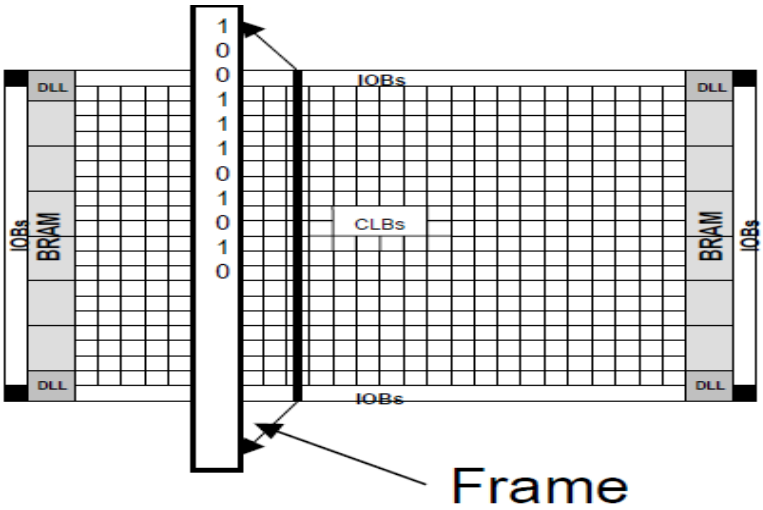
Για την υλοποίηση λογικών συναρτήσεων χρησιμοποιούνται τα LTUs τα οποία μπορούν να συνδυαστούν μέσω πολυπλεκτών και άλλης λογικής ώστε να υλοποιήσουν συναρτήσεις περισσοτέρων των τεσσάρων εισόδων. Επίσης τα LUTs μπορούν να χρησιμοποιηθούν και για την υλοποίηση κατανεμημένων μνημών. Τα στοιχεία μνήμης τα οποία υπάρχουν σε ένα CLB μπορούν να διαμορφωθούν έτσι ώστε να λειτουργούν ως flip-flops η ως latches. Στο σχήμα 1.3 φαίνεται η δομή ενός Virtex slice. Για την διαμόρφωση ενός CLB χρειαζόμαστε 64 bits για τα τέσσερα LUT καθώς και μερικά επιπλέον bits για την διαμόρφωση των καταχωρητών και για τα σήματα ελέγχου των πολυπλεκτών

Η δρομολόγηση των σημάτων του κυκλώματος υλοποιείται σε δυο επίπεδα σε τοπικό και σε γενικού σκοπού δρομολόγηση. Με τον ορό τοπικό επίπεδο δρομολόγησης εννοούμε τις συνδέσεις που ενεργοποιούνται μεταξύ CLBs και GRM, μεταξύ CLBs (χωρίς διαμεσολάβηση GRM και μεταξύ στοιχειών των CLBs δημιουργώντας βρόγχους ανάδρασης . Με τον όρο γενικού σκοπού δρομολόγηση εννοούμε την δρομολόγηση σημάτων με τους πόρους των καναλιών δρομολόγησης που βρίσκονται ανάμεσα σε στήλες και γραμμές CLBs



Σχήμα 2 Δομή ενός slice

Ο προγραμματισμός των FPGAs γίνεται φορτώνοντας τα απαραίτητα bitstreams στην configuration memory που διαθέτει κάθε συσκευή . Η configuration memory των συσκευών της σειράς Virtex είναι διαχωρισμένη σε frames. Ένα frame είναι η μικρότερη μονάδα η οποία μπορεί να αναδιαταχθεί . Ένα frame έχει πλάτος ένα bit και το μήκος του εξαρτάται από τον αριθμό των στηλών της εκάστοτε συσκευής. Επίσης ένα CLB για να διαταχθεί χρειάζεται περισσότερα του ενός frames. Για την συσκευή XC2V30 την οποία χρησιμοποιούμε , ένα frame έχει μήκος 206 λέξεις των 32 bit και ένα CLB απαιτεί 22 frames για τον προγραμματισμό του. Συνολικά διαθέτει 1756 frames



Σχήμα 3 Απεικόνιση ενός frame

Επίσης εκτός από τους παραπάνω πόρους οι FPGA που ανήκουν στην οικογένεια Virtex II Pro περιέχου και άλλους όπως, ενσωματωμένους επεξεργαστές BRAMS και άλλους οι οποίοι παρουσιάζονται στον παρακάτω πίνακα.

Device(1)	RocketIO Transceiver Blocks	PowerPC Processor Blocks	Logic Cells(2)	CLB (1 = 4 slices = max 128 bits)		18 X 18 Bit Multiplier Blocks	Block SelectRAM+		DCMs	Maximum User I/O Pads
				Slices	Max Distr RAM (Kb)		18 Kb Blocks	Max Block RAM (Kb)		
XC2VP2	4	0	3,168	1,408	44	12	12	216	4	204
XC2VP4	4	1	6,768	3,008	94	28	28	504	4	348
XC2VP7	8	1	11,088	4,928	154	44	44	792	4	396
XC2VP20	8	2	20,880	9,280	290	88	88	1,584	8	564
XC2VPX20	8(4)	1	22,032	9,792	306	88	88	1,584	8	552
XC2VP30	8	2	30,816	13,696	428	136	136	2,448	8	644
XC2VP40	0(3), 8, or 12	2	43,632	19,392	606	192	192	3,456	8	804
XC2VP50	0(3) or 16	2	53,136	23,616	738	232	232	4,176	8	852
XC2VP70	16 or 20	2	74,448	33,088	1,034	328	328	5,904	8	996
XC2VPX70	20(4)	2	74,448	33,088	1,034	308	308	5,544	8	992
XC2VP100	0(3) or 20	2	99,216	44,096	1,378	444	444	7,992	12	1,164

Πίνακας 1 Πόροι των συσκευών που ανήκουν στην οικογένεια Virtex II Pro .

Στον παρακάτω πίνακα παρουσιάζεται ο αριθμός των στηλών που περιέχει η συσκευή XC2VP30 την οποία χρησιμοποιήσαμε καθώς και τον απαιτούμενο αριθμό των Frames για την διάταξή τους.

Τύπος Στήλης	IOB	IOI	CLB	BRAM	BRAM Interconnect	GCLK
# Στηλών	2	2	46	8	8	1
Frames/Στήλη	4	44	44	64	22	4

Πίνακας 2 Αριθμός διαθέσιμων στηλών ανά τύπο και απαιτούμενος αριθμός frames για την διαμόρφωση του συγκεκριμένου τύπου στην XC2VP30