ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Μικροεπεξεργαστών και Υλικού

Διπλωματική Εργασία

"Ανάπτυξη επεξεργαστή για εφαρμογές κρυπτογράφησης, βασισμένου στον DLX"

Κλατσιάς Κωνσταντίνος

Επιβλέπων

Επίκουρος Καθηγητής Ι. Παπαευσταθίου

Εξεταστική Επιτροπή

Επίκουρος Καθηγητής Ι. Παπαευσταθίου

Αναπληρωτής Καθηγητής Δ. Πνευματικάτος

Καθηγητής Α. Δόλλας

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επίκουρο καθηγητή Ιωάννη Παπαευσταθίου για την πολύτιμη βοήθεια που μου προσέφερε κατά τη διάρκεια της παρούσας διπλωματικής εργασίας.

Επίσης, ένα μεγάλο ευχαριστώ στον κ. Μάρκο Κιμιωνή καθώς και σε όλη την ομάδα του εργαστηρίου μικροεπεξεργαστών και υλικού για την σημαντική βοήθεια που παρείχαν στην διεκπεραίωση της εργασίας.

Στην οικογένεια μου

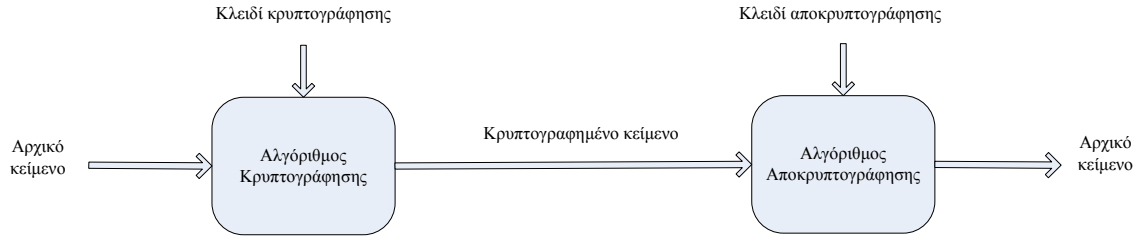
Περιεχόμενα

1. Εισαγωγή

Είναι γενικά αποδεκτό ότι η ανάπτυξη του διαδικτύου τα τελευταία χρόνια είναι ραγδαία. Σύμφωνα με πρόσφατες έρευνες η παγκόσμια αύξηση των χρηστών του διαδικτύου στο διάστημα 2000 με 2008 αγγίζει το 290% με αποτέλεσμα ο αριθμός των χρηστών να έχει ξεπεράσει το 1,4 δισεκατομμύριο [I1]. Υπηρεσίες όπως το ηλεκτρονικό ταχυδρομείο, οι ηλεκτρονικές αγορές, οι τραπεζικές καθώς και οι κάθε είδους επαγγελματικές συναλλαγές είναι από τις κυριότερες που προσφέρει το διαδίκτυο και απαιτούν απόλυτη μυστικότητα. Για αυτό το λόγο έχουν σχεδιαστεί πρωτόκολλα που καθιστούν ασφαλείς τις διαδικτυακές συνδέσεις και προστατεύουν τη μετάδοση δεδομένων από κακόβουλους χρήστες. Τα πρωτόκολλα αυτά χρησιμοποιούν την κρυπτογραφία για να κρατήσουν το απόρρητο στις ψηφιακές επικοινωνίες.

Η κρυπτογραφία είναι ένας κλάδος της επιστήμης της κρυπτολογίας, η οποία ασχολείται με την μελέτη της ασφαλούς επικοινωνίας. Η κρυπτολογία χωρίζεται σε δύο κλάδους: την κρυπτογραφία και την κρυπτανάλυση και προέρχεται από τις ελληνικές λέξεις "κρυπτός" και "λόγος". Ο κύριος στόχος της κρυπτογραφίας είναι να παρέχει μηχανισμούς για 2 ή περισσότερα μέλη να επικοινωνήσουν χωρίς κάποιος άλλος να είναι ικανός να διαβάζει την πληροφορία εκτός από τα μέλη.

Η πλήρης διαδικασία αποτελείται από δύο κύρια μέρη, την κρυπτογράφηση και την αποκρυπτογράφηση του μηνύματος που είναι προς μετάδοση. Ο αποστολέας με βάση ένα κρυπτογραφικό αλγόριθμο και ένα μυστικό κλειδί μετασχηματίζει το αρχικό κείμενο σε μια ακατανόητη μορφή το κρυπτογραφημένο κείμενο. Στη μεριά του παραλήπτη είναι γνωστός ο κρυπτογραφικός αλγόριθμος και με την ανάκτηση του μυστικού κλειδιού μπορεί να γίνει η αντίστροφη διαδικασία όπου από το κρυπτογραφημένο κείμενο παράγεται το αρχικό κείμενο. Η παραπάνω διαδικασία παρουσιάζεται στο Σχήμα 1.1.



Σχήμα 1.1 Ένα τυπικό σύστημα κρυπτογράφησης – αποκρυπτογράφησης

Η εξέλιξη της χρήσης της κρυπτογραφίας ολοένα αυξάνεται καθιστώντας πλέον αξιόπιστη την μεταφορά της πληροφορίας και για διάφορους άλλους λειτουργικούς σκοπούς όπως:

- Ασφάλεια συναλλαγών σε τράπεζες δίκτυα ATM
- Κινητή τηλεφωνία
- Σταθερή τηλεφωνία
- Στρατιωτικά δίκτυα
- Διπλωματικά δίκτυα
- Συστήματα συναγερμών
- Έξυπνες κάρτες
- Δορυφορικές εφαρμογές
- Ασύρματα δίκτυα

1.1 Ιστορία της Κρυπτογραφίας

Η πρώτη στρατιωτική χρήση της κρυπτογραφίας αποδίδεται στους Σπαρτιάτες. Γύρω στον 5ο π.Χ. αιώνα εφεύραν τη «σκυτάλη», την πρώτη κρυπτογραφική συσκευή, στην οποία, χρησιμοποίησαν για την κρυπτογράφηση, τη μέθοδο της αντικατάστασης. Η «Σπαρτιατική Σκυτάλη», ήταν μια ξύλινη ράβδος, ορισμένης διαμέτρου, γύρω από την οποία ήταν τυλιγμένη ελικοειδώς μια λωρίδα περγαμηνής. Το κείμενο ήταν γραμμένο σε στήλες, ένα γράμμα σε κάθε έλικα, όταν δε ξετύλιγαν τη λωρίδα, το κείμενο ήταν ακατάληπτο εξαιτίας της ανάμειξης των γραμμάτων. Το «κλειδί» ήταν η διάμετρος της σκυτάλης.

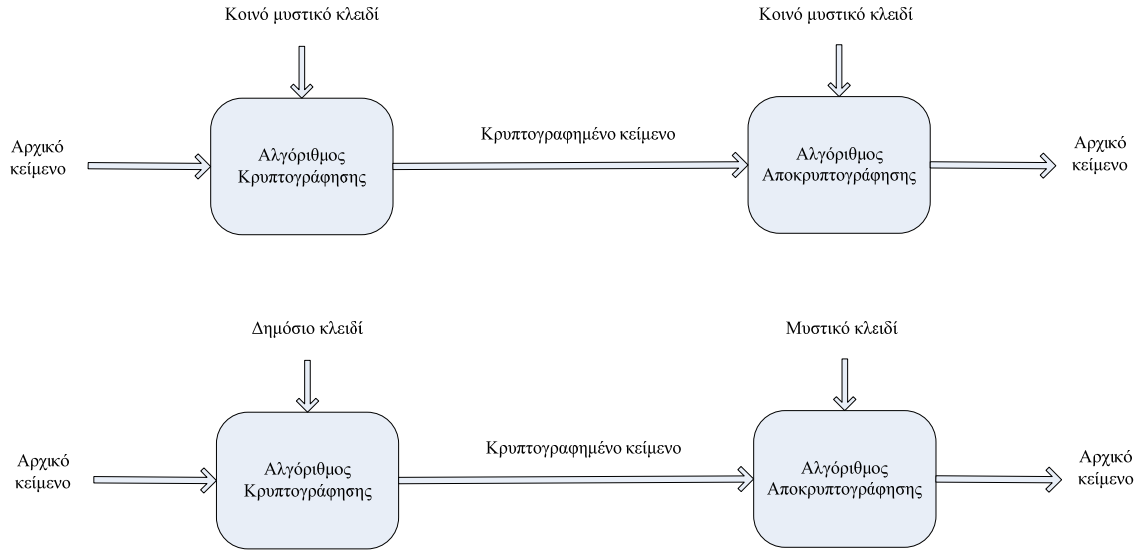
Ο Ιούλιος Καίσαρας έγραφε αντικαθιστώντας τα γράμματα του κειμένου, με γράμματα, που βρίσκονται 3 θέσεις μετά, στο λατινικό αλφάβητο. Για παράδειγμα η λέξη “SECRET” μετά την κρυπτογράφηση γίνεται “VHFUHW”. Έτσι, σήμερα ο κρυπτογραφικός αλγόριθμος που στηρίζεται στην αντικατάσταση των γραμμάτων του αλφαβήτου με άλλα που βρίσκονται σε καθορισμένο αριθμό θέσης πριν ή μετά λέγεται κρυπτοσύστημα αντικατάστασης του Καίσαρα. Από τα παραπάνω γίνεται αντιληπτό ότι στα συγκεκριμένα κρυπτοσυστήματα το κλειδί μπορεί να είναι οποιοσδήποτε αριθμός από το 1 μέχρι το 25.

Κατά τη διάρκεια των δύο παγκοσμίων πολέμων, λόγω της εξαιρετικά μεγάλης ανάγκης που υπήρξε για ασφάλεια στην μετάδοση ζωτικών πληροφοριών μεταξύ των στρατευμάτων των χωρών, η ανάπτυξη της κρυπτογραφίας ήταν ραγδαία. Ένα σημαντικό

ιστορικό παράδειγμα είναι ο κρυπτογραφικός αλγόριθμος «Αίνιγμα», που σχεδίασαν οι γερμανοί κατά τη διάρκεια του 2^{ου} παγκοσμίου πολέμου, ο οποίος βασίστηκε στον αλγόριθμο του Καίσαρα [I2].

1.2 Σύγχρονη Κρυπτογραφία

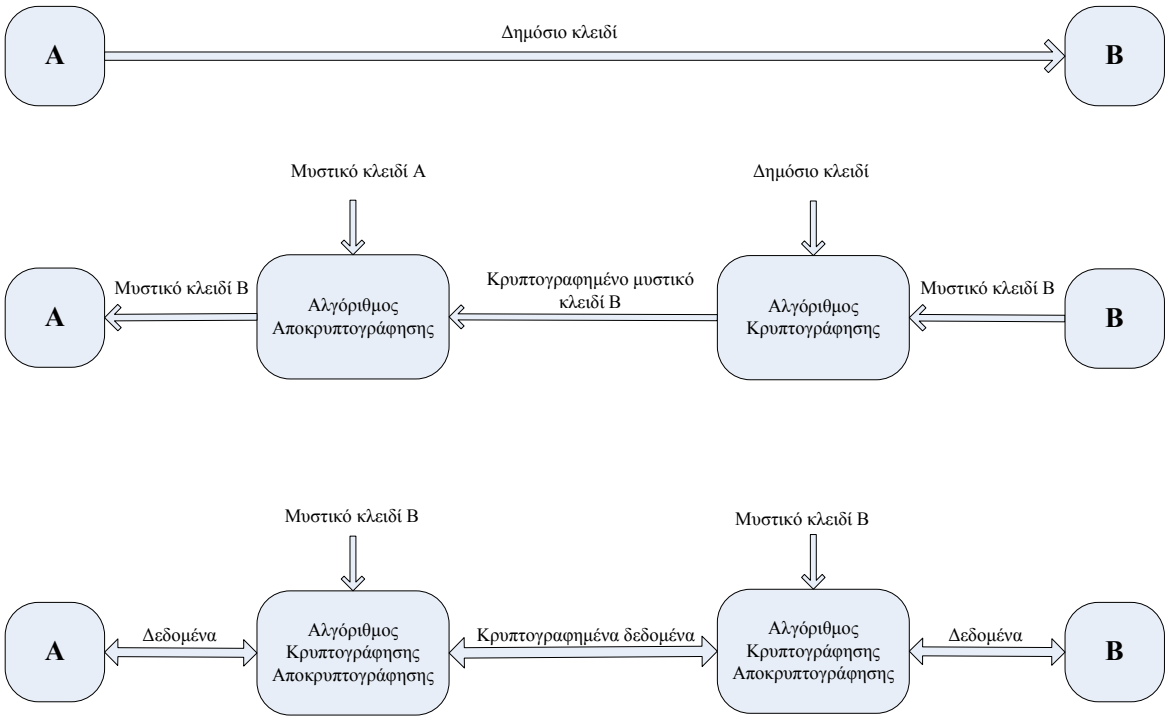
Στις μέρες μας υπάρχουν αρκετοί κρυπτογραφικοί αλγόριθμοι που χρησιμοποιούνται κυρίως στη προστασία της πληροφορίας κατά τη μετάδοση της σε δημόσια επικοινωνιακά δίκτυα. Οι αλγόριθμοι αυτοί χωρίζονται σε δύο κύριες κατηγορίες, στους συμμετρικούς και στους ασύμμετρους.



Σχήμα 1.2 Συμμετρικού κλειδιού (πάνω) και Ασύμμετρου κλειδιού (κάτω) αλγόριθμοι

Όπως φαίνεται στο Σχήμα 1.2 οι κρυπτογραφικοί αλγόριθμοι ασύμμετρου κλειδιού χρησιμοποιούν δύο τύπους κλειδιών, ένα δημόσιο για την κρυπτογράφηση των δεδομένων και ένα ιδιωτικό για την αποκρυπτογράφηση τους. Από την άλλη μεριά οι κρυπτογραφικοί αλγόριθμοι συμμετρικού κλειδιού χρησιμοποιούν ένα κοινό μυστικό κλειδί τόσο για την κρυπτογράφηση όσο και την αποκρυπτογράφηση. Είναι γεγονός ότι οι κρυπτογραφικοί αλγόριθμοι συμμετρικού κλειδιού είναι αρκετά πιο γρήγοροι από τους κρυπτογραφικούς αλγόριθμους ασύμμετρου κλειδιού, δεν μπορούν όμως να εγγυηθούν το πως θα γίνει η ανταλλαγή του μυστικού κλειδιού.

Για να επιτευχθεί μια ασφαλής ανταλλαγή δεδομένων χρησιμοποιείται ένας υβριδικός κρυπτογραφικός αλγόριθμος που εκμεταλλεύεται την ταχύτητα του συμμετρικού για μετάδοση μεγάλου όγκου δεδομένων και την ευελιξία του ασύμμετρου για την μετάδοση του μυστικού κλειδιού.



Σχήμα 1.3 Διαδικασία εγκαθίδρυσης ασφαλούς καναλιού επικοινωνίας

Όπως φαίνεται στο Σχήμα 1.3 για την εγκαθίδρυση ενός ασφαλούς καναλιού επικοινωνίας χρησιμοποιείται ένας υβριδικός κρυπτογραφικός αλγόριθμος. Πιο συγκεκριμένα η διαδικασία αναλύεται στις 3 παρακάτω φάσεις:

1. Το άτομο Α στέλνει το δημόσιο κλειδί στο άτομο Β χωρίς να χρησιμοποιεί κάποιο κρυπτογραφικό αλγόριθμο.
2. Το άτομο Β στέλνει το μυστικό κλειδί του στο άτομο Α χρησιμοποιώντας ένα ασύμμετρο κρυπτογραφικό αλγόριθμο.
3. Από αυτό το σημείο όλα τα δεδομένα και από τα δύο άτομα κρυπτογραφούνται και αποκρυπτογραφούνται με τη χρήση κάποιου συμμετρικού κρυπτογραφικού αλγόριθμου μια και έχει διαμοιραστεί το κοινό μυστικό κλειδί.

1.3 Οργάνωση Κεφαλαίων

Σκοπός της διπλωματικής εργασίας αυτής είναι σε πρώτη φάση ο σχεδιασμός και η υλοποίηση ενός ομόχειρου επεξεργαστή που να υποστηρίζει ένα υποσύνολο εντολών του γνωστού επεξεργαστή DLX της οικογένειας RISC. Σε δεύτερη φάση στόχος ήταν να σχεδιαστεί και να υλοποιηθεί μια τροποποιημένη εκδοχή του συγκεκριμένου επεξεργαστή, έτσι ώστε να επιτευχθεί μια σαφής βελτίωση της απόδοσης του όταν τρέχει συμμετρικούς κρυπτογραφικούς αλγορίθμους. Η κύρια παράμετρος που χρησιμοποιήθηκε

για την μέτρηση της βελτίωσης της απόδοσης, ήταν ο χρόνος εκτέλεσης σε κύκλους ρολογιού του κρυπτογραφικού αλγόριθμου AES-Rijndael. Το τελικό αποτέλεσμα είναι ένας επεξεργαστής τύπου VLIW που δίνει τα επιθυμητά αποτελέσματα.

Στο 2^{ο} κεφάλαιο παρουσιάζονται κάποιες ιδιότητες των συμμετρικών κρυπτογραφικών αλγορίθμων που επηρεάζουν άμεσα τη διαδικασία κρυπτογράφησης και αποκρυπτογράφησης. Το κεφάλαιο κλείνει με μια συνοπτική περιγραφή του σημαντικότερου αλγόριθμου αυτής της κατηγορίας του AES-Rijndael.

Στο 3^{ο} κεφάλαιο περιγράφεται η σχετική δουλειά που έχει γίνει. Η σχετική δουλειά αρχικά κατηγοριοποιείται σε υλοποιήσεις λογισμικού και υλικού. Παρουσιάζονται κάποια αποτελέσματα για τις υλοποιήσεις λογισμικού και στη συνέχεια οι υλοποιήσεις υλικού κατηγοριοποιούνται περαιτέρω σε τρεις υποκατηγορίες. Οι τρεις υποκατηγορίες για τις οποίες παραθέτονται υλοποιήσεις είναι οι εξής: α) υλικό εξειδικευμένο για αλγόριθμο, β)επέκταση αρχιτεκτονικής συνόλου εντολών και γ) συνεπεξεργαστές κρυπτογράφησης.

Στο 4^{ο} κεφάλαιο περιγράφονται τα βασικά χαρακτηριστικά για την υλοποίηση του επεξεργαστή τύπου DLX. Για κάθε βαθμίδα του επεξεργαστή υπάρχει αναλυτική περιγραφή των κυκλωμάτων που την αποτελούν καθώς και σχηματική παρουσίαση αυτών. Το κεφάλαιο ολοκληρώνεται με την συνένωση των επιμέρους βαθμίδων και τη παρουσίαση του πλήρους μονοπατιού δεδομένων.

Στο 5^{ο} κεφάλαιο αρχικά θα γίνει μια μελέτη για το είδος των εντολών που πρέπει να προστεθούν στην υλοποίηση που έχει παρουσιαστεί στο προηγούμενο κεφάλαιο. Στη συνέχεια παρουσιάζονται βήμα βήμα οι τροποποιήσεις πάνω στον επεξεργαστή τύπου DLX έτσι ώστε να υλοποιηθεί ένα σύνολο εντολών ικανό να ανταποκριθεί στις ανάγκες των σύγχρονων κρυπτογραφικών εφαρμογών.

Το 6^{ο} κεφάλαιο ξεκινάει με τη παρουσίαση των εργαλείων που χρησιμοποιήθηκαν για την επιβεβαίωση της λειτουργίας των δύο υλοποιήσεων, κάποια ενδεικτικά κομμάτια κώδικα που έτρεξαν στους δύο επεξεργαστές και την παρουσίαση μιας τεχνικής για την επίτευξη βελτιωμένης απόδοσης μέσο κώδικα στον AES-Rijndael αλγόριθμο. Στη συνέχεια έγινε μια αποτίμηση της απόδοσης των δύο επεξεργαστών σε κάποιες συσκευές FPGA της Xilinx. Για την παρουσίαση της απόδοσης χρησιμοποιήθηκε ως δείκτης ο ρυθμός ολοκλήρωσης ενώ έγινε και εκτίμηση της επιτάχυνσης που προσφέρει η επέκταση της αρχιτεκτονικής για κάθε συσκευή FPGA. Το κεφάλαιο κλείνει με μια σύγκριση των αποτελεσμάτων των δύο υλοποιήσεων με αποτελέσματα παλιότερων υλοποιήσεων.

Στο $7^{ο}$ κεφάλαιο παρουσιάζονται συνοπτικά τα κυριότερα χαρακτηριστικά της αρχιτεκτονικής μετά την επέκταση και προτείνονται κάποιες τροποποιήσεις που πιθανόν πετύχουν περαιτέρω αύξηση της απόδοσης.

2. Συμμετρικοί Κρυπτογραφικοί Αλγόριθμοι

Σε αυτό το κεφάλαιο αναλύονται κάποιες ιδιότητες των συμμετρικών κρυπτογραφικών αλγορίθμων όπως προκύπτουν μέσα από τη διαδικασία κρυπτογράφησης αποκρυπτογράφησης. Κάποιοι από τους δημοφιλέστερους αλγόριθμους αυτού του τύπου είναι οι εξής: AES-Rijndael [1], MARS, Twofish, Serpent. Το κεφάλαιο κλείνει με μία συνοπτική περιγραφή του AES-Rijndael που θεωρείται ο σημαντικότερος από τους παραπάνω.

2.1 Διαδικασία Κρυπτογράφησης και Αποκρυπτογράφησης

Κάθε συμμετρικός κρυπτογραφικός αλγόριθμος έχει τρεις παραμέτρους που παίζουν σημαντικότατο ρόλο στην επίτευξη της μέγιστης δυνατής ασφάλειας. Οι παράμετροι αυτοί είναι οι εξής :

1. Ο αριθμός των bit του μυστικού κλειδιού.
2. Ο αριθμός των bit των δεδομένων που κρυπτογραφεί (μπλοκ).
3. Ο αριθμός των υπολογιστικών κύκλων.

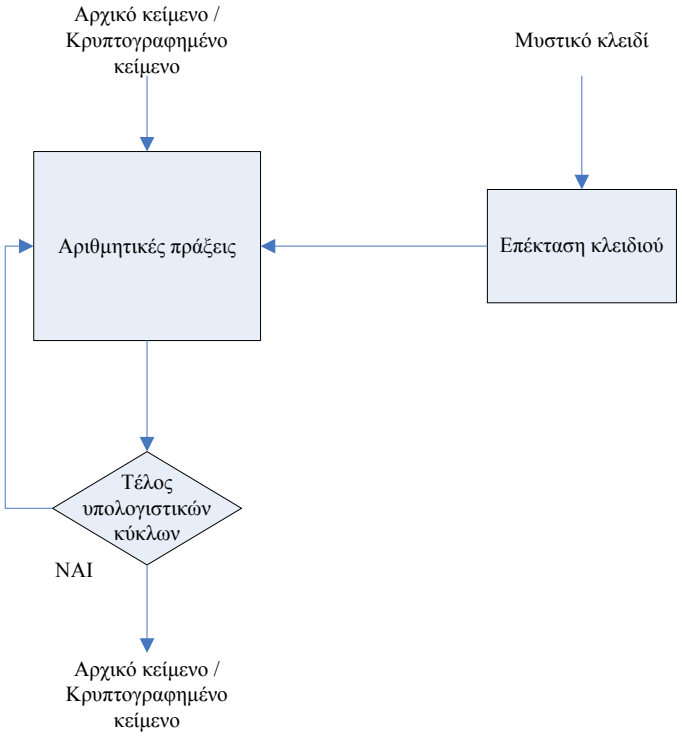
Οι δύο πρώτοι παράμετροι επηρεάζουν αρκετά του πόρους που θα χρησιμοποιηθούν σε μία υλοποίηση σε υλικό (αριθμό καταχωρητών και κατανομή μνήμης). Ο αριθμός των bit του μυστικού κλειδιού είναι άμεση συνάρτηση του επιπέδου ασφαλείας του κρυπτογραφικού αλγόριθμου. Το παραπάνω προκύπτει από το γεγονός ότι για μια επίθεση με εξαντλητική δοκιμή όλων των πιθανών κλειδιών η απαιτούμενη υπολογιστική ισχύς αυξάνεται εκθετικά σε σχέση με το μέγεθος του κλειδιού.

Η τρίτη παράμετρος είναι εξίσου σημαντική για το επίπεδο ασφαλείας που προσφέρει ο κρυπτογραφικός αλγόριθμος. Αυτό συμβαίνει λόγω του ότι σε κάθε υπολογιστικό κύκλο τα δεδομένα «ανακατεύονται» όλο και περισσότερο. Είναι επιλογή του σχεδιαστή να επιλέξει τον αριθμό των υπολογιστικών κύκλων λαμβάνοντας υπόψη το επίπεδο ασφαλείας που θέλει να πετύχει αλλά και την επιθυμητή ταχύτητα επεξεργασίας του μπλοκ. Ο αριθμός των υπολογιστικών κύκλων μπορεί να συγκρατηθεί αν ο ίδιος ο κύκλος είναι αρκετά ισχυρός από μόνος του, κάνει δηλαδή τις κατάλληλες αριθμητικές πράξεις στα δεδομένα του μπλοκ.

Κάθε συμμετρικός κρυπτογραφικός αλγόριθμος ξεκινάει με μια φάση αρχικοποίησης που είναι η επέκταση κλειδιού. Πιο συγκεκριμένα ο αλγόριθμος επεξεργάζεται το μυστικό

κλειδί έτσι ώστε να παράγει ένα αριθμό από καινούρια κλειδιά που χρησιμοποιούνται σε διάφορους υπολογιστικούς κύκλους κρυπτογράφησης ή αποκρυπτογράφησης.

Μόλις ολοκληρωθεί η φάση της αρχικοποίησης ξεκινάει η διαδικασία της κρυπτογράφησης. Η τελευταία αποτελείται από ένα αριθμό από διάφορες αριθμητικές πράξεις πάνω στο αρχικό κείμενο για ένα συγκεκριμένο αριθμό υπολογιστικών κύκλων. Μόλις εξαντληθεί αυτός ο αριθμός υπολογιστικών κύκλων το κείμενο θεωρείται κρυπτογραφημένο και είναι έτοιμο για μετάδοση. Η διαδικασία της αποκρυπτογράφησης είναι ακριβώς η ίδια με μόνη διαφορά ότι σαν είσοδο παίρνει το κρυπτογραφημένο κείμενο και σαν έξοδο βγάζει το αρχικό κείμενο. Στο Σχήμα 2.1 παρουσιάζεται ολόκληρη η διαδικασία κρυπτογράφησης αποκρυπτογράφησης.



Σχήμα 2.1 Διαδικασία κρυπτογράφησης / αποκρυπτογράφησης

Ένα τελευταίο στοιχείο για τους συμμετρικούς κρυπτογραφικούς αλγόριθμους είναι ότι χρησιμοποιούν κάποιους τρόπους λειτουργίας για να επεξεργαστούν ολόκληρο το μήνυμα. Οι δύο πιο διαδεδομένοι είναι ο ECB (Electronic Code Book), όπου κάθε μπλοκ δεδομένων κρυπτογραφείται ξεχωριστά και ο CBC (Cipher Block Chaining), όπου κάθε μπλοκ δεδομένων γίνεται XOR (αποκλειστικό-Ή) με το κρυπτογραφημένο μπλοκ που μόλις πριν είχε παραχθεί. Ο CBC προσφέρει μεγαλύτερη ασφάλεια, αλλά ο ECB προσφέρει μεγαλύτερη ταχύτητα κρυπτογράφησης του μηνύματος.

2.2 Ο Κρυπτογραφικός Αλγόριθμος AES-Rijndael

Ο αλγόριθμος κρυπτογράφησης AES-Rijndael περιγράφει μια διαδικασία κρυπτογράφησης του αρχικού κειμένου βασισμένη στην λογική της κωδικοποίησης μπλοκ δεδομένων με κάποιο μυστικό κλειδί. Έχει προτυποποιηθεί από το NIST (National Institute of Technology) τον Νοέμβριο του 2001, αντικαθιστώντας το πρότυπο DES (Data Encryption Standard) και πλέον αποτελεί τον προτεινόμενο αλγόριθμο για εφαρμογές κρυπτογράφησης.

Το πρότυπο υποστηρίζει την χρήση κλειδιών μήκους 128, 192 και 256 bit. Ανεξάρτητα από το μήκος κλειδιού, ο αλγόριθμος επενεργεί πάνω σε μπλοκ δεδομένων μήκους 128 bit. Η κρυπτογραφική διαδικασία είναι επαναληπτική. Αυτό σημαίνει ότι σε κάθε μπλοκ δεδομένων γίνεται μια επεξεργασία η οποία επαναλαμβάνεται έναν αριθμό από φορές ανάλογα με τον αριθμό των bit του κλειδιού. Κάθε επανάληψη ονομάζεται κύκλος επεξεργασίας. Στον πρώτο κύκλο επεξεργασίας είσοδος είναι ένα μπλοκ αρχικού κειμένου και το μυστικό κλειδί, ενώ στους γύρους που ακολουθούν είσοδος είναι το μπλοκ που έχει προκύψει από τον προηγούμενο γύρο καθώς και ένα κλειδί που έχει παραχθεί από το αρχικό με βάση τη διαδικασία επέκτασης κλειδιού που ορίζει ο αλγόριθμος. Το αποτέλεσμα της επεξεργασίας είναι το κρυπτογραφημένο μπλοκ. Το μπλοκ αυτό πρέπει να σημειωθεί ότι έχει ακριβώς το ίδιο μέγεθος σε bit με το μπλοκ του αρχικού κειμένου.

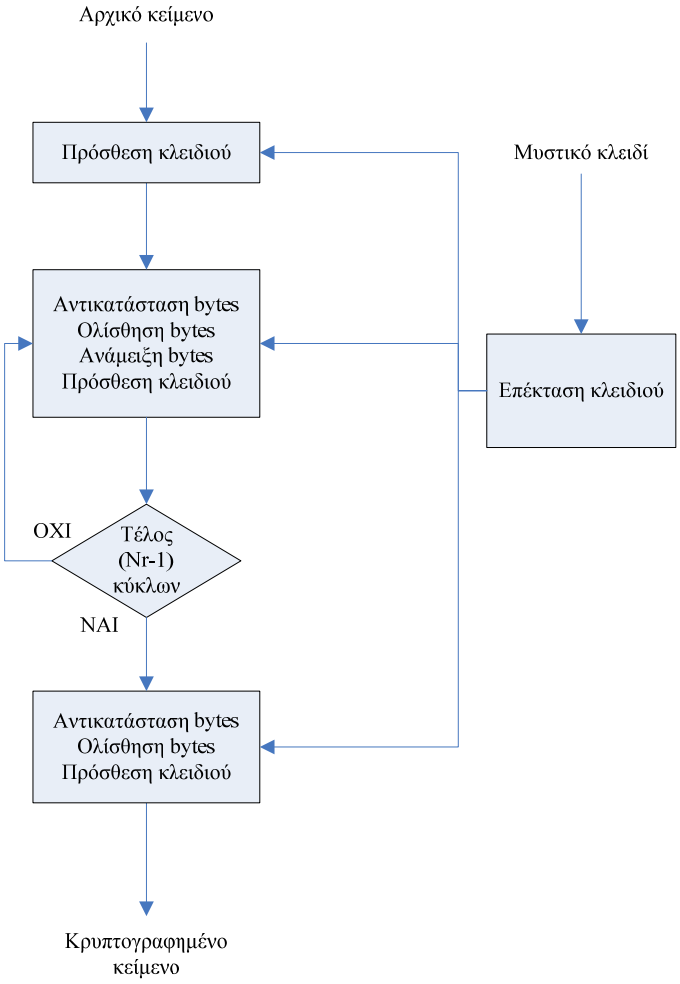
Για την περιγραφή του αλγορίθμου ορίζονται οι μεταβλητές Nb, Nk, Nr. Από τον αριθμό των bit του μπλοκ δεδομένων προκύπτει η ποσότητα Nb = 4 , που συμβολίζει τον αριθμό των 32-bit λέξεων στο μπλοκ. Αντίστοιχα η μεταβλητή Nk συμβολίζει τον αριθμό των 32-bit λέξεων που μπορεί να περιλαμβάνει ένα κλειδί και κατά συνέπεια μπορεί να πάρει τις τιμές 4, 6 και 8. Επίσης από το μέγεθος του κλειδιού προκύπτει και η μεταβλητή Nr που χρησιμοποιείται για να δηλώσει το πλήθος των υπολογιστικών κύκλων. Αν χρησιμοποιηθεί μήκος κλειδιού 128 bit τότε απαιτούνται 10 κύκλοι επεξεργασίας. Για μήκη κλειδιού ίσα με 192 και 256 bits απαιτούνται 12 και 14 κύκλοι αντίστοιχα.

2.2.1 Κρυπτογράφηση

Όπως φαίνεται και στο Σχήμα 2.1 το βασικότερο τμήμα της διαδικασίας κρυπτογράφησης είναι αυτό των αριθμητικών πράξεων. Στον AES-Rijndael το τμήμα αυτό αποτελείται από 4 μετασχηματισμούς σε επίπεδο byte:

1. Ένα μετασχηματισμό αντικατάστασης bytes χρησιμοποιώντας κάποιον σχετικό πίνακα αντικατάστασης.
2. Ένα μηχανισμό ολίσθησης bytes κατά κάποιες θέσεις.
3. Μια διαδικασία ανάμειξης των bytes.
4. Μια πρόσθεση ενός κλειδιού.

Η διαδικασία ξεκινάει με μια πρόσθεση κλειδιού και συνεχίζει με 10, 12 ή 14 κύκλους επεξεργασίας, με τον τελευταίο να διαφέρει από τους υπόλοιπους. Προσαρμόζοντας τα παραπάνω στο Σχήμα 2.1 που παρουσιάζει την γενική περίπτωση προκύπτει το Σχήμα 2.2 που παρουσιάζει την AES-Rijndael διαδικασία κρυπτογράφησης.



Σχήμα 1.2 Διαδικασία κρυπτογράφησης AES-Rijndael

Ο μετασχηματισμός της πρόσθεσης κλειδιού επιβάλει την πρόσθεση των δεδομένων με τα κλειδιά που έχουν παραχθεί από την διαδικασία επέκτασης κλειδιού. Πρέπει να

διευκρινιστεί ότι η πρόσθεση στον AES-Rijndael αλγόριθμο γίνεται modulo-2, αντιστοιχεί δηλαδή στην λογική πράξη XOR.

Ο μετασχηματισμός αντικατάστασης των bytes αποτελεί μια μη γραμμική αντικατάσταση των bytes με την χρήση ενός πίνακα αντικατάστασης (S-Box). Το S-Box δεν υπολογίζεται κατά την κρυπτογράφηση αλλά οι τιμές του έχουν προϋπολογιστεί με την σύνθεση κάποιων μετασχηματισμών. Στον Πίνακα 2.1 που ακολουθεί παρατίθενται οι τιμές του πίνακα S-Box όπως τις παρουσιάζει το NIST στο επίσημο έγγραφο για τον AES-Rijndael.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Πίνακας 2.1 Ο πίνακας αντικατάστασης (S-Box) για τον AES-Rijndael

Ο μηχανισμός ολίσθησης bytes αρχικά διαχωρίζει τις τέσσερις 32-bit λέξεις του μπλοκ δημιουργώντας ένα πίνακα τεσσάρων στηλών όπου κάθε στήλη είναι μία λέξη η οποία αναπαρίσταται σε byte. Σε αυτό τον πίνακα γίνεται μία μετατροπή των γραμμών με τις στήλες δημιουργώντας τέσσερις νέες λέξεις για επεξεργασία. Η πρώτη λέξη παραμένει ανέπαφη, ενώ στις υπόλοιπες τα bytes ολισθαίνουν διαφορετικό αριθμό θέσεων. Η δεύτερη λέξη ολισθαίνει αριστερά κατά μία θέση με αποτέλεσμα το πρώτο byte της γραμμής να βρεθεί τελευταίο (κυκλική ολίσθηση). Με αντίστοιχο τρόπο ολισθαίνουν η τρίτη και η τέταρτη λέξη αλλά κατά 2 και 3 θέσεις αντίστοιχα. Ο μετασχηματισμός τελειώνει κάνοντας την αντίστροφη μετατροπή των γραμμών σε στήλες.

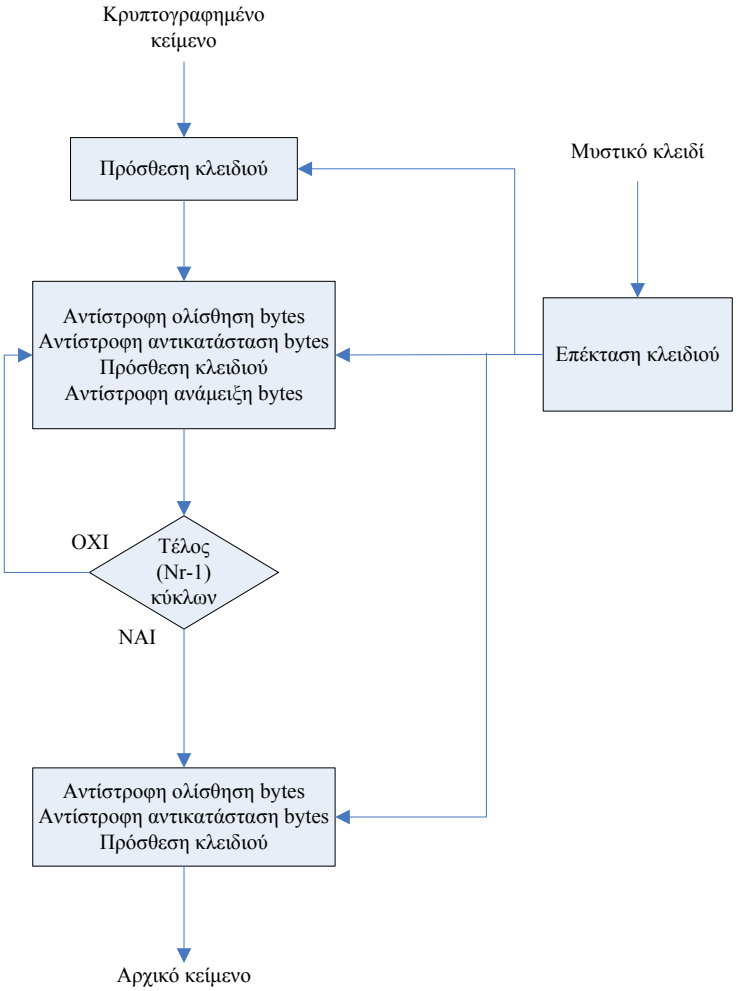
Η διαδικασία ανάμειξης των bytes είναι στην ουσία ένας πολλαπλασιασμός σε ένα πεπερασμένο πεδίο. Η κάθε 32-bit λέξη του μπλοκ θεωρείται πολυώνυμο τρίτης τάξης με συντελεστές τις τιμές των bytes της. Η κάθε λέξη πολλαπλασιάζεται modulo (x^4+1) με ένα προκαθορισμένο πολυώνυμο.

Η διαδικασία επέκτασης του κλειδιού δέχεται ως είσοδο το αρχικό μυστικό κλειδί , μήκους Nb λέξεων, και παράγει μια ακολουθία κλειδιών. Παράγονται συνολικά από αυτή την διαδικασία Nb(Nr+1) λέξεις καθώς σε κάθε έναν από τους Nr γύρους επεξεργασίας απαιτούνται Nb λέξεις από δεδομένα κλειδιού.

2.2.2 Αποκρυπτογράφηση

Για την διαδικασία της αποκρυπτογράφησης θα χρησιμοποιηθούν περίπου οι ίδιοι μετασχηματισμοί τροποποιημένοι έτσι ώστε να κάνουν την αντίστροφη λειτουργία. Εξαίρεση αποτελεί ο μετασχηματισμός της πρόσθεσης κλειδιού λόγω του ότι είναι μια απλή XOR πράξη που είναι από μόνη της αντιστρέψιμη.

Η διαδικασία ξεκινάει πάλι με μια πρόσθεση κλειδιού και συνεχίζει με 10, 12 ή 14 κύκλους επεξεργασίας (με τους αντίστροφους μετασχηματισμούς), με τον τελευταίο να διαφέρει από τους υπόλοιπους. Αναλυτικότερα η διαδικασία φαίνεται στο Σχήμα 2.3.



Σχήμα 2.3 Διαδικασία αποκρυπτογράφησης AES-Rijndael

Για τον μετασχηματισμό αντίστροφης αντικατάστασης των bytes αντί για τον S-Box πίνακα αντικατάστασης χρησιμοποιείται ο αντίστροφος του (inverse S-Box), ο οποίος και παρουσιάζεται στον Πίνακα 2.2.

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Πίνακας 2.2 Ο πίνακας αντικατάστασης inverse S-Box

Η διαδικασία που χρησιμοποιείται για τον αντίστροφο μηχανισμό ολίσθησης bytes είναι ακριβώς η ίδια με τη κανονική με μόνη διαφοροποίηση ότι τα bytes σε αυτή ολισθαίνουν προς τα δεξιά τον αντίστοιχο αριθμό θέσεων.

Η αντίστροφη διαδικασία ανάμειξης των bytes προϋποθέτει να υπάρχει το αντίστροφο πολυώνυμο από αυτό που είχε χρησιμοποιηθεί για τον πολλαπλασιασμό σε πεπερασμένο πεδίο. Ένα ζευγάρι τέτοιων πολυώνυμων είναι τα παρακάτω:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$
$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}.$$

Συνεπώς αν ο αρχικός πολλαπλασιασμός είχε γίνει με το $a(x)$ για την αντίστροφη διαδικασία ανάμειξης των bytes κάθε λέξη πολλαπλασιάζεται modulo (x^4+1) με το $a^{-1}(x)$.

3. Σχετική Δουλειά

Το κεφάλαιο αυτό εστιάζει στη σχετική δουλειά που έχει γίνει τόσο σε επίπεδο λογισμικού όσο και σε επίπεδο υλικού. Οι υλοποιήσεις σε λογισμικό είναι σαφώς φθηνότερες αλλά παρουσιάζουν κάποια μειονεκτήματα σε σχέση με τις υλοποιήσεις σε υλικό. Οι υλοποιήσεις σε υλικό είναι ταχύτερες και προσφέρουν υψηλότερο επίπεδο ασφάλειας.

Είναι εύκολα κατανοητό ότι λόγω της μεγάλης χρηστικότητας των συμμετρικών κρυπτογραφικών αλγορίθμων υπάρχουν αναρίθμητες υλοποιήσεις. Το ινστιτούτο NIST μετά από έρευνες τριών χρόνων, σε ένα μεγάλο αριθμό συμμετρικών κρυπτογραφικών αλγορίθμων, κατέληξε σε ένα σύνολο που περιέχει μόλις 5 [I3]. Αυτοί οι 5 αλγόριθμοι θεωρείται ότι παρέχουν την υψηλότερη δυνατή ασφάλεια και για αυτό ακριβώς το λόγο χρησιμοποιούνται συχνότερα. Το παραπάνω επιβεβαιώνεται από το γεγονός ότι το μεγαλύτερο μέρος της σχετικής δουλειάς για συμμετρικούς κρυπτογραφικούς αλγόριθμους αφορά τους συγκεκριμένους.

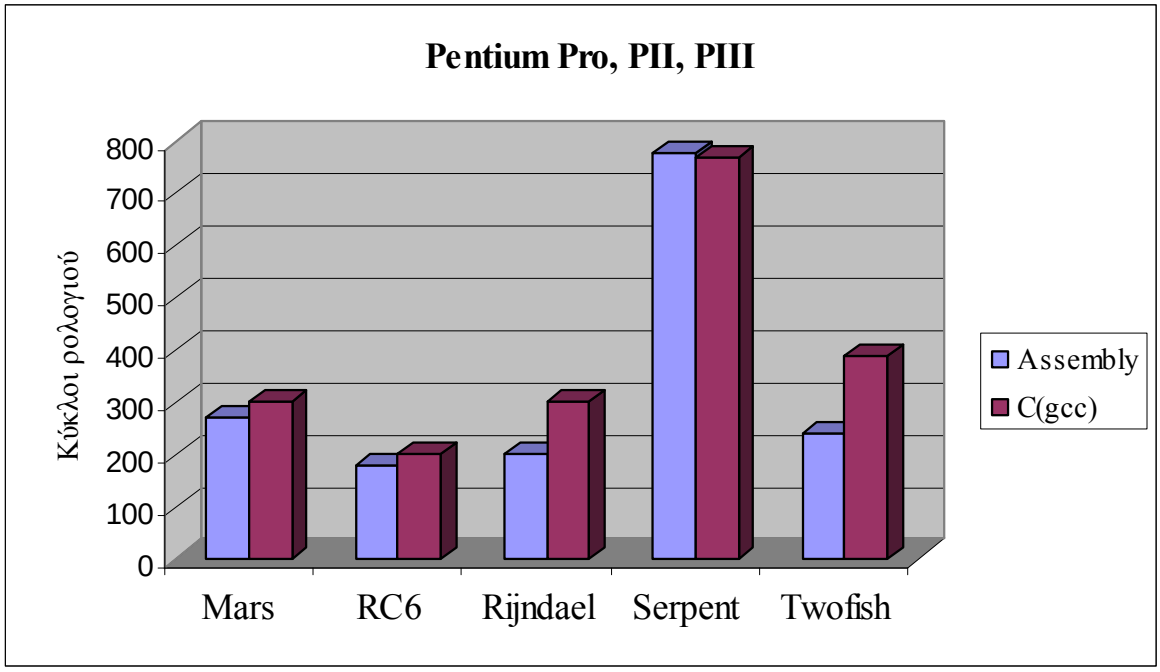
3.1 Υλοποιήσεις Λογισμικού

Σε αυτή την ενότητα θα παρουσιαστούν διάφορες υλοποιήσεις, των 5 αλγορίθμων που αναφέρθηκαν παραπάνω, σε γλώσσα assembly αλλά και σε υψηλού επιπέδου γλώσσα προγραμματισμού τη C. Οι υλοποιήσεις αυτές που βρέθηκαν στην [2] χρησιμοποιούν ως δείκτη για την μέτρηση της ταχύτητας τους κύκλους ρολογιού που χρειάζονται για να εκτελεστεί ο εκάστοτε αλγόριθμος.

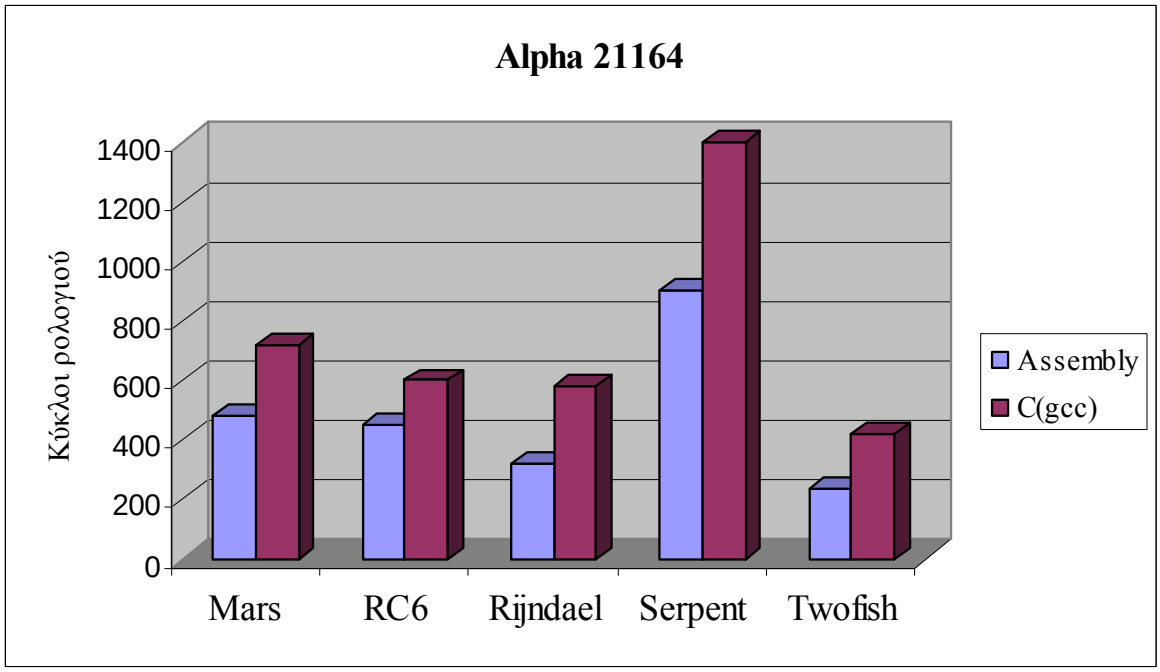
Στο Διάγραμμα 3.1 παρουσιάζονται οι χρόνοι εκτέλεσης σε assembly και C των συγκεκριμένων συμμετρικών κρυπτογραφικών αλγορίθμων στους επεξεργαστές της Intel Pentium Pro [3], Pentium II [4], Pentium III [5] . Στο Διάγραμμα 3.2 παρουσιάζονται οι αντίστοιχοι χρόνοι για τον επεξεργαστή Alpha 21164. Όπως ήταν αναμενόμενο οι υλοποιήσεις σε assembly είναι σχεδόν πάντα ταχύτερες από αυτές σε C με τους χρόνους στον Alpha 21164 να τονίζουν αυτή τη διαφορά. Μοναδική εξαίρεση αποτελεί ο αλγόριθμος Serpent που όταν τρέχει στους επεξεργαστές της Intel ο χρόνος εκτέλεσης σε assembly είναι ελάχιστα χειρότερος.

Λόγω του ότι ο AES-Rijndael κρυπτογραφικός αλγόριθμος θεωρείται σύμφωνα με το NIST το νέο AES πρότυπο θεωρήθηκε σωστό να εξετασθούν οι υλοποιήσεις του και σε κάποιους άλλους επεξεργαστές. Στο Διάγραμμα 3.3 παρουσιάζονται οι χρόνοι για τους

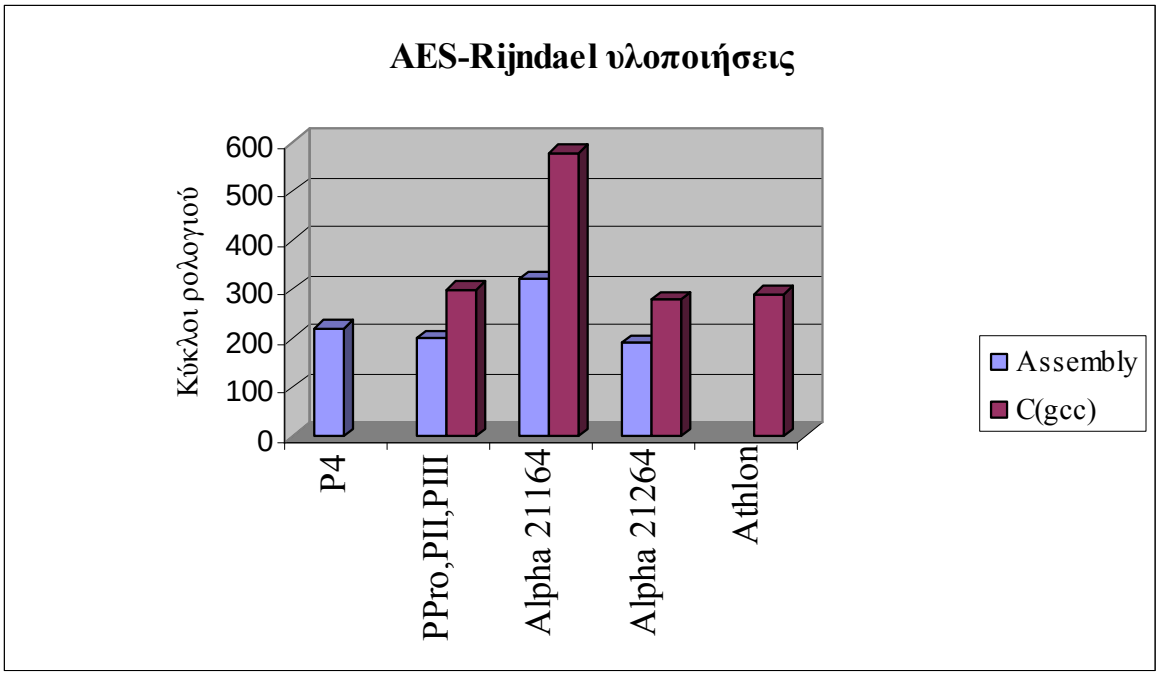
προηγούμενους επεξεργαστές και επιπλέον για τους Pentium 4 της Intel [6], Alpha 21264 της Digital Equipment [7] και Athlon της AMD [8].



Διάγραμμα 3.1 Υλοποιήσεις σε επεξεργαστές της Intel



Διάγραμμα 3.2 Υλοποιήσεις στον επεξεργαστή Alpha 21164



Διάγραμμα 3.3 Υλοποιήσεις του AES-Rijndael σε διάφορους επεξεργαστές

3.2 Υλοποιήσεις σε Υλικό

Οι υλοποιήσεις σε υλικό διακρίνονται σε τρεις κατηγορίες που θα εξεταστούν σε ξεχωριστές υποενότητες. Η πρώτη κατηγορία είναι η υλοποίηση εξειδικευμένου υλικού για αλγόριθμο. Η δεύτερη κατηγορία είναι η επέκταση κάποιας αρχιτεκτονικής συνόλου εντολών για την αποδοτικότερη υποστήριξη των αλγορίθμων. Τρίτη και ίσως σημαντικότερη κατηγορία είναι αυτή των συνεπεξεργαστών κρυπτογράφησης. Στην ενότητα αύτη θα παρουσιαστούν υλοποιήσεις και από τις τρεις παραπάνω κατηγόριες και θα δοθεί έμφαση στην απόδοση τους κατά την εκτέλεση του AES-Rijndael κρυπτογραφικού αλγορίθμου.

3.2.1 Υλικό εξειδικευμένο για αλγόριθμο

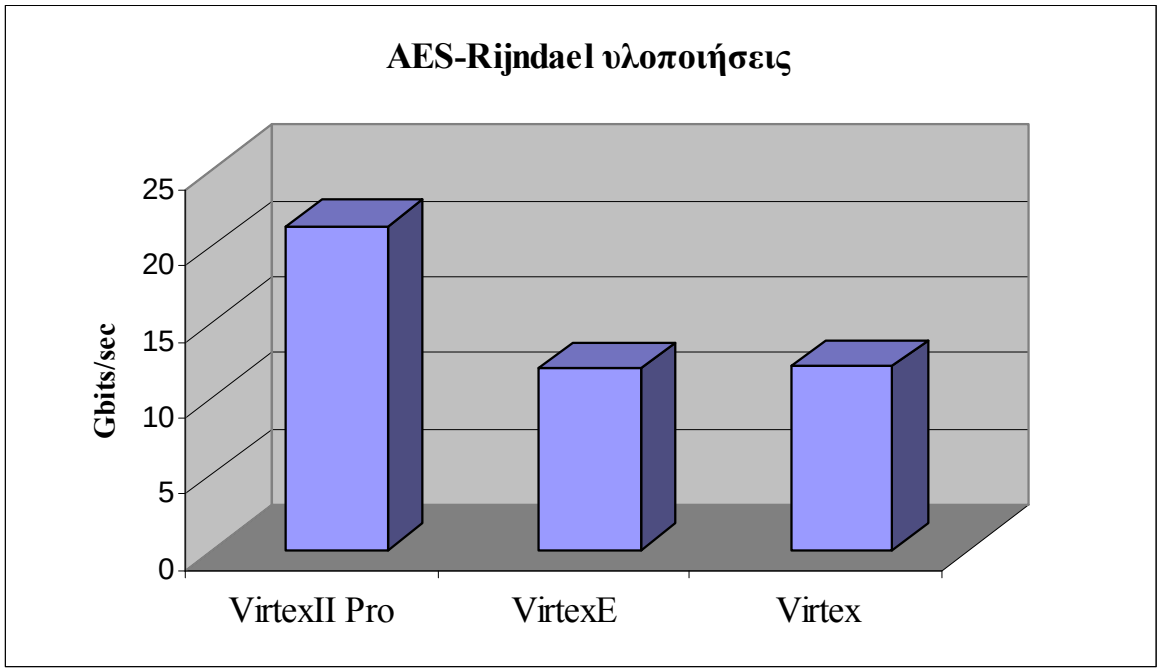
Υπάρχουν αρκετές υλοποιήσεις υλικού που βασίζονται σε συσκευές FPGA. Αυτή η κατηγορία υλοποιήσεων παρέχει πολύ μεγάλη ταχύτητα πράγμα απολύτως λογικό μια και όλοι οι πόροι είναι αφοσιωμένοι στις ανάγκες του κάθε αλγορίθμου για τον οποίο έγινε η υλοποίηση. Από την άλλη μεριά το αντίτιμο για αυτή την υψηλή ταχύτητα είναι η ευελιξία που στις συγκεκριμένες υλοποιήσεις είναι μηδενική. Η μηδενική ευελιξία προκύπτει από το γεγονός ότι οι συγκεκριμένες υλοποιήσεις έχουν σχεδιαστεί με μοναδικό στόχο την αποδοτικότερη δυνατή υποστήριξη ενός συγκεκριμένου αλγορίθμου

με αποτέλεσμα να μην έχουν καμία χρησιμότητα όταν ο συγκεκριμένος αλγόριθμος δεν είναι ο κατάλληλος για κάποια εφαρμογή.

Μερικές από τις ταχύτερες υλοποιήσεις για τον AES-Rijndael αλγόριθμο όπως παρουσιάστηκαν στην [2] είναι οι εξής :

- Οι Hodjat και Verbauwhede στην [9] με τη χρήση μίας VirtexII Pro FPGA [10] πέτυχαν 21.54 Gbits/sec ρυθμό ολοκλήρωσης .
- Οι McLoone και McCanny στην [11] με τη χρήση μιας VirtexE FPGA [12] πέτυχαν 12 Gbits/sec ρυθμό ολοκλήρωσης.
- Οι Chodowiec, Khuon και Gaj στην [13] με τη χρήση μιας Virtex XCV1000 FPGA [14] πέτυχαν 12.1Gbits/sec ρυθμό ολοκλήρωσης.

Οι παραπάνω αποδόσεις παρουσιάζονται συγκεντρωτικά στο Διάγραμμα 3.4



Διάγραμμα 3.4 Υλοποιήσεις σε εξειδικευμένο υλικό για τον AES-Rijndael αλγόριθμο

3.2.2 Επέκταση αρχιτεκτονικής συνόλου εντολών

Στις υλοποιήσεις αυτής της κατηγορίας στόχος είναι η βελτίωση των επιδόσεων κάποιας αρχιτεκτονικής για συμμετρικούς κρυπτογραφικούς αλγορίθμους. Αυτό επιτυγχάνεται κυρίως με την προσθήκη νέων δομών στο μονοπάτι δεδομένων που υλοποιούν καινούριες αποδοτικότερες εντολές. Οι υλοποιήσεις αυτές προσφέρουν μεγαλύτερη ευελιξία από την

πρώτη κατηγορία υλοποιήσεων αλλά και πάλι οι υψηλές ταχύτητες αφορούν μόνο τους αλγόριθμους για τους οποίους έχουν γίνει οι επεκτάσεις.

- Οι Abdurohman και Sutikno στην [15] βελτίωσαν την απόδοση του DLX επεξεργαστή για τον AES-Rijndael κρυπτογραφικό αλγόριθμο. Το παραπάνω επετεύχθη με την προσθήκη στον DLX ενός εξαρτήματος που αναλαμβάνει τον μετασχηματισμό αντικατάστασης byte. Η προσθήκη αυτή απέδωσε επιτάχυνση 1.82x.
- Οι Burke, Mcdonald και Austin στην [16] προσπάθησαν να βελτιώσουν την απόδοση του επεξεργαστή Alpha 21264 για συμμετρικούς κρυπτογραφικούς αλγορίθμους. Η προσπάθεια αυτή τους οδήγησε σε μια επέκταση της αρχιτεκτονικής συνόλου εντολών έτσι ώστε να υπάρχουν εντολές που να εκτελούν πράξεις όπως κυκλικές περιστροφές, πολλαπλασιασμούς modulo, μεταθέσεις και προσβάσεις σε πίνακες αντικατάστασης. Η επέκταση απέδωσε επιτάχυνση που φτάνει μέχρι και 1.74x.
- Οι Fiskiran και Lee στην [17] δημιούργησαν μια νέα δομή που λέγεται PTLU (Parallel Table Look Up) που στηρίζεται στην παράλληλη πρόσβαση πολλαπλών πινάκων αναφοράς (LUTs) για να πετύχει ταχύτερη πρόσβαση στον πίνακα αντικατάστασης (S-Box). Το αποτέλεσμα ήταν επιτάχυνση 7.7x για τον AES-Rijndael κρυπτογραφικό αλγόριθμο σε ένα επεξεργαστή RISC 64-bit.

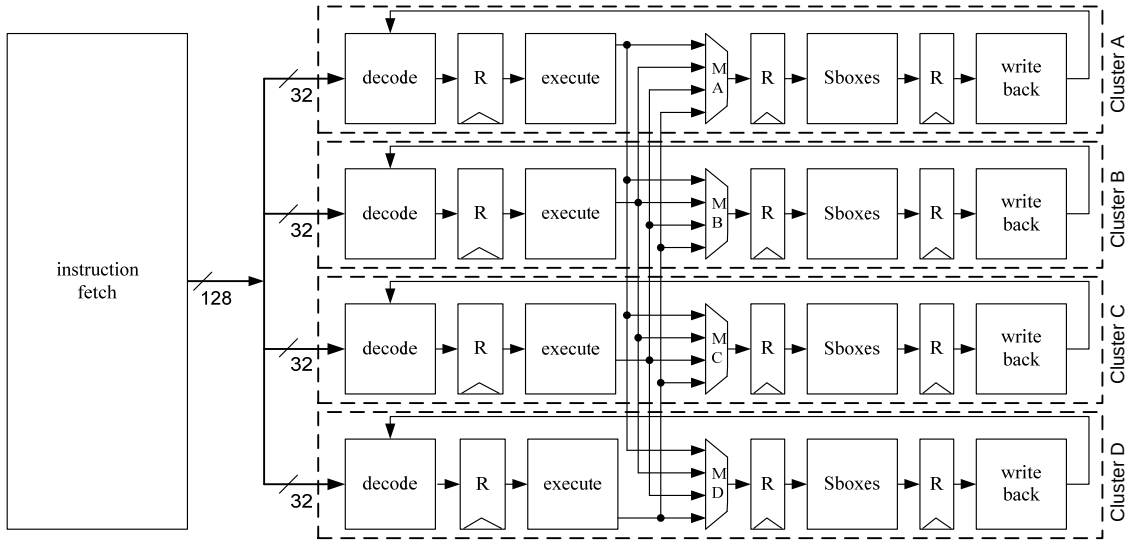
3.2.3 Συνεπεξεργαστές Κρυπτογράφησης

Οι υλοποιήσεις αυτής της κατηγορίας έχουν στόχο να καλύψουν το κενό μεταξύ απόδοσης και ευελιξίας. Για να επιτευχθεί το παραπάνω οι σχεδιαστές πρέπει να λαμβάνουν υπόψη τις ιδιότητες όσο το δυνατόν περισσότερων συμμετρικών κρυπτογραφικών αλγορίθμων.

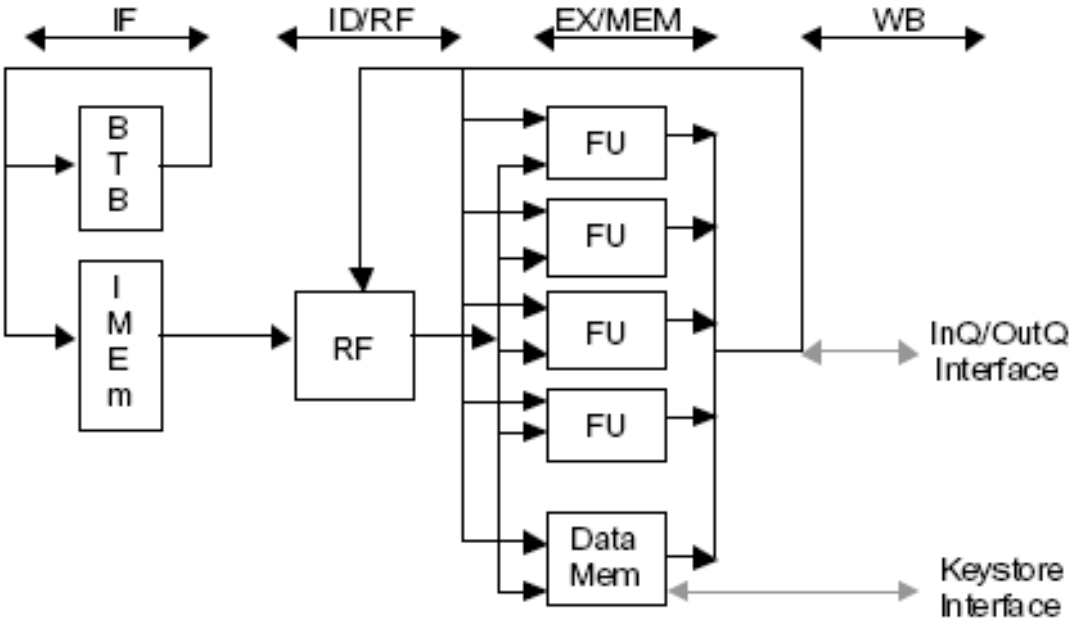
Ο Δ. Θεοδωρόπουλος στην [2] παρουσίασε τον επεξεργαστή CCproc, μια υλοποίηση που υποστηρίζει το σύνολο των 5 αλγορίθμων που το NIST επέλεξε ως πρότυπα AES. Πρόκειται για ένα VLIW επεξεργαστή με πλάτος 4 λέξεις των 32 bit που χρησιμοποιεί τη δομή ενός RISC μονοπατιού δεδομένων. Η σχεδίαση αυτή πέτυχε ρυθμό ολοκλήρωσης 175 Mbits/sec. Στο Σχήμα 3.1 φαίνεται μια περιληπτική απεικόνιση του CCproc.

Οι Wu, Weaver, και Austin στην [18] παρουσίασαν τον επεξεργαστή Cryptomaniac, μια υλοποίηση που παρέχει ευελιξία και υψηλή απόδοση. Πρόκειται για ένα VLIW

επεξεργαστή με πλάτος 4 λέξεις των 32 bit, χωρίς κρυφή μνήμη και με ένα απλό μοντέλο πρόβλεψης διακλάδωσης . Η υλοποίηση πέτυχε ρυθμό ολοκλήρωσης 512 Mbits/s για τον AES-Rijndael κρυπτογραφικό αλγόριθμο. Στο Σχήμα 3.2 περιγράφεται σε υψηλό επίπεδο η αρχιτεκτονική του Cryptomaniac.

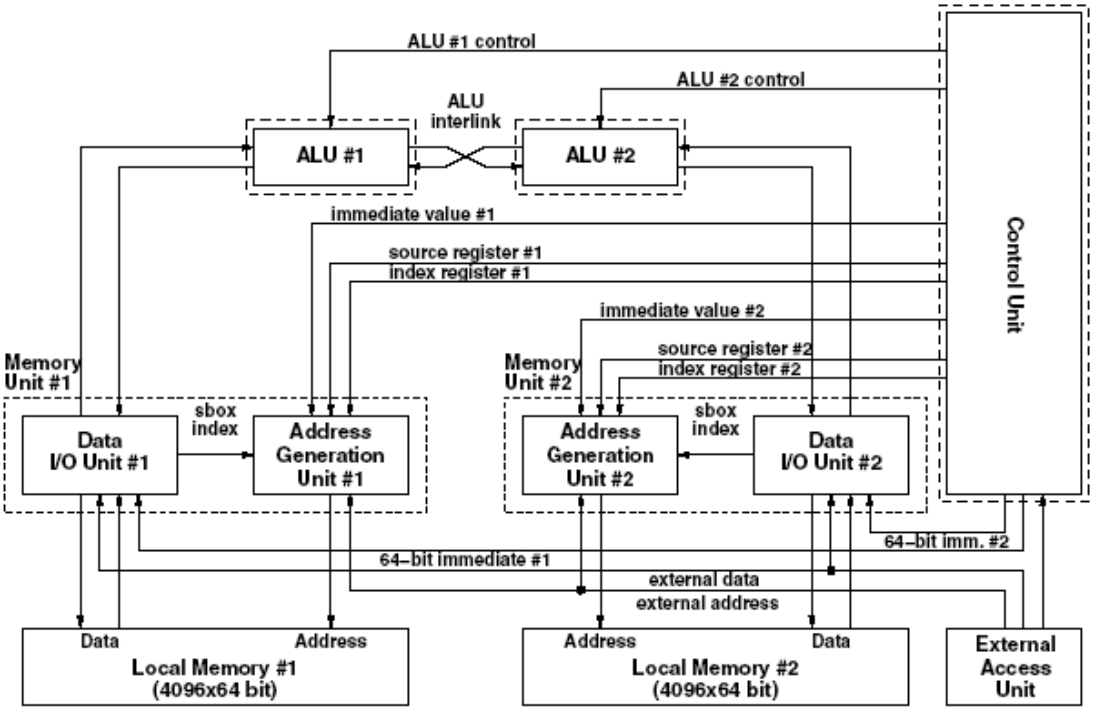


Σχήμα 3.1 Ο επεξεργαστής CCproc



Σχήμα 3.2 Ο επεξεργαστής Cryptomaniac

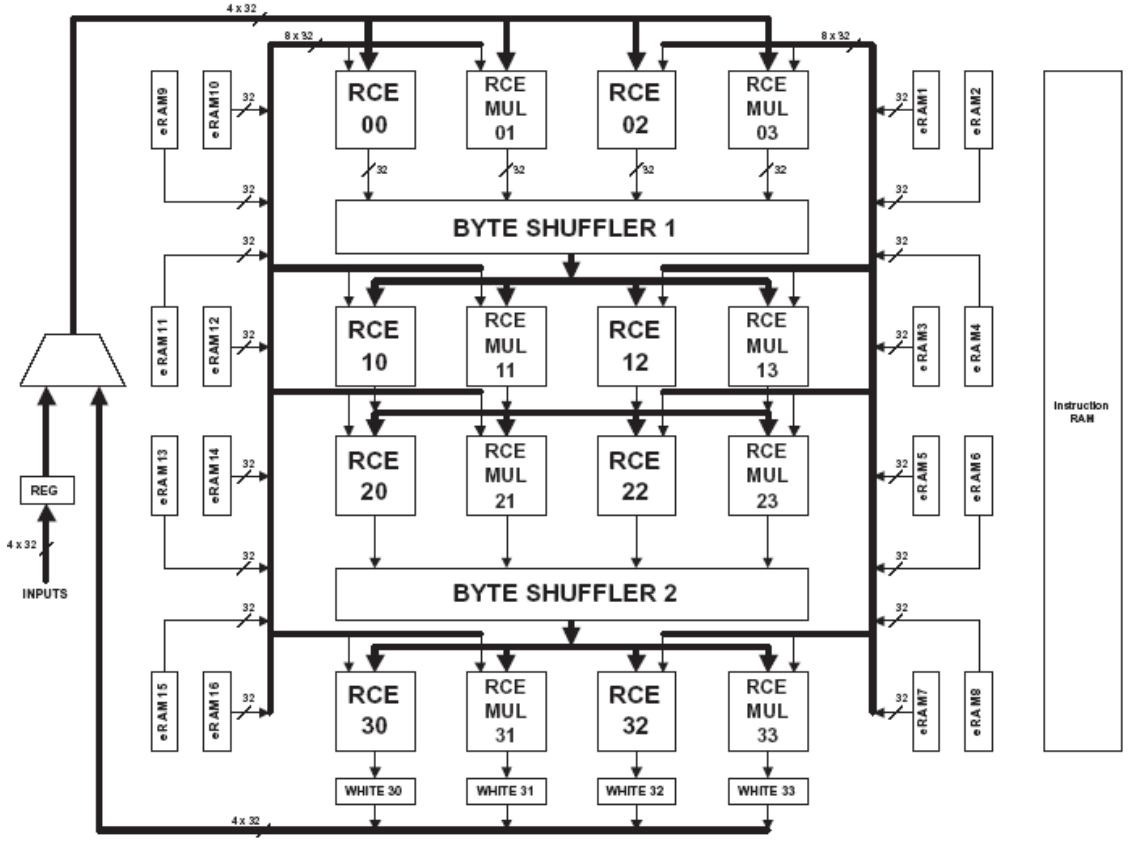
Οι Oliva, Buchty και Heintze στην [19] παρουσίασαν τον επεξεργαστή Cryptonite που χρησιμοποιεί εντολές 64 bit σε μία ομοχειρία τριών σταδίων. Η υλοποίηση αποτελείται από δυο πυρήνες που ο καθένας περιέχει 4 καταχωρητές 64 bit, μια ALU, μια μονάδα παραγωγής διευθύνσεων και μια τοπική μνήμη. Ο Cryptonite πέτυχε περίπου 700 Mbits/sec ρυθμό ολοκλήρωσης για τον AES-Rijndael κρυπτογραφικό αλγόριθμο. Στο Σχήμα 3.3 παρουσιάζεται μια περιληπτική απεικόνιση του Cryptonite



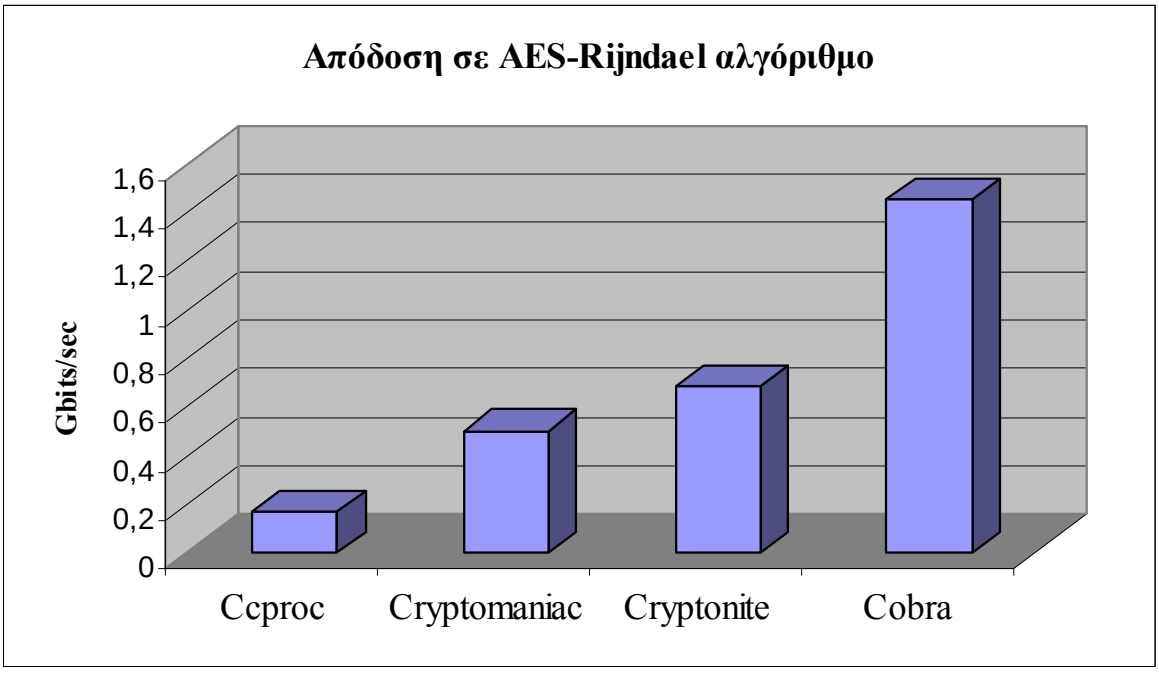
Σχήμα 3.3 Ο επεξεργαστής Cryptonite

Οι Elbirt και Paar στην [20] παρουσίασαν τον επεξεργαστή COBRA που παρέχει μια ειδική αρχιτεκτονική για την βέλτιστη απόδοση των κρυπτογραφικών αλγορίθμων που ενεργούν σε μπλοκ δεδομένων. Η αρχιτεκτονική συνόλου εντολών του COBRA περικλείει εντολές για αριθμητικές πράξεις, πολλαπλασιασμούς modulo, πολλαπλασιασμούς σε πεπερασμένα πεδία και προσβάσεις σε πίνακες αντικατάστασης. Η σχεδίαση αυτή πετυχαίνει 1.451 Gbits/sec ρυθμό ολοκλήρωσης για τον AES-Rijndael κρυπτογραφικό αλγόριθμο. Στο Σχήμα 3.4 παρουσιάζεται η αρχιτεκτονική του COBRA και κάποιες διασυνδέσεις.

Το Διάγραμμα 3.5 συγκεντρώνει τις επιδόσεις των συνεπεξεργαστών κρυπτογράφησης που αναφέρθηκαν για τον AES-Rijndael κρυπτογραφικό αλγόριθμο.



Σχήμα 3.4 Ο επεξεργαστής Cobra



Διάγραμμα 3.5 Αποδόσεις των συνεπεξεργαστών κρυπτογράφησης για τον AES-Rijndael κρυπτογραφικό αλγόριθμο

4. Αρχιτεκτονική Υποσυνόλου Εντολών του DLX

Ο DLX είναι ένας τυπικός RISC επεξεργαστής που χρησιμοποιεί την τεχνική της ομοχειρίας [21]. Σχεδιάστηκε από τους John Hennesy και David Patterson με σκοπό να εκμεταλλευτούν το γεγονός ότι στοιχειώδης εντολές έχουν πολλή μεγαλύτερη συχνότητα χρήσης.

4.1 Μελέτη Σχεδίασης του DLX

Η συγκεκριμένη υλοποίηση αφορά ένα υποσύνολο εντολών του DLX. Το υποσύνολο αυτό περιλαμβάνει τους εξής τύπους εντολών:

1. Εντολές πρόσβασης μνήμης.
2. Εντολές αριθμητικών και λογικών πράξεων.
3. Εντολές διακλάδωσης και άλματος.

Η υλοποίηση ακολουθεί πιστά το μοντέλο DLX χωρίζοντας το σχέδιο σε 5 ομόχειρες βαθμίδες :

1. Βαθμίδα ανάκτησης εντολών.
2. Βαθμίδα αποκωδικοποίησης εντολών.
3. Βαθμίδα εκτέλεσης εντολών.
4. Βαθμίδα πρόσβασης μνήμης.
5. Βαθμίδα εγγραφής αποτελέσματος.

Η σχεδίαση πρέπει να υποστηρίζει την επίλυση όλων τον κινδύνων που προκύπτουν από την ομόχειρη εκτέλεση εντολών. Οι κίνδυνοι αυτοί μπορεί να είναι δεδομένων, δομικοί ή ελέγχου. Η επίλυση των κινδύνων δεδομένων και δομικών κινδύνων γίνεται με τα κατάλληλα κυκλώματα ανίχνευσης τους και προώθησης των δεδομένων. Ωστόσο είναι γνωστό ότι σε τέτοιου τύπου ομοχειρία υπάρχει ένας κίνδυνος δεδομένων, που παρόλο που ανιχνεύεται, δεν μπορεί να επιλυθεί και οδηγεί σε ανάσχεση της ομοχειρίας για ένα κύκλο ρολογιού. Ο συγκεκριμένος είναι αυτός που προκύπτει από την απόπειρα διαβάσματος καταχωρητή που στην προηγούμενη εντολή φορτώνεται από την μνήμη δεδομένων. Όσων αφορά τους κινδύνους ελέγχου η επίλυση τους γίνεται με την διαδικασία εισαγωγής «φυσαλίδας» στην ομοχειρία. Ο όρος «φυσαλίδα» υπονοεί τον μηδενισμό των ομόχειρων καταχωρητών, που κρατάνε πληροφορία για εντολές που

εντέλει δεν θα εκτελεστούν αμέσως μετά, οδηγώντας στην άσκοπη απασχόληση πόρων της ομοχειρίας για κάποιους κύκλους ρολογιού. Σε τέτοιου τύπου ομοχειρία χρησιμοποιείται το μοντέλο πρόβλεψης ανεπιτυχούς διακλάδωσης με κόστος την εισαγωγή «φυσαλίδας» που καταναλώνει 2 κύκλους ρολογιού.

4.2 Αρχιτεκτονική Συνόλου Εντολών της Σχεδίασης

Οι εντολές της συγκεκριμένης σχεδίασης, όπως και κάθε RISC υπολογιστή, έχουν σταθερό μήκος 32 bit. Τα 32 bit χωρίζονται σε διάφορα πεδία ανάλογα με τη μορφή της εντολής. Υπάρχουν τρεις διαφορετικές μορφές εντολών, οι R-type, οι I-type και οι J-type που παρουσιάζονται στον Πίνακα 4.1. Στο πίνακα παρουσιάζονται και τα αντίστοιχα πεδία με την αρίθμηση να ξεκινάει από το bit 31 που είναι το πιο σημαντικό bit και να καταλήγει στο bit 0.

	31-26	25-21	20-16	15-11	10-6	5-0
R	opcode	rs	rt	rd	nu	func
I				imm[15-0]		
J		disp[25-0]				

Πίνακας 4.1 Μορφές και πεδία εντολών

Ακολουθεί ο Πίνακας 4.2 με την εξήγηση καθώς και την περιγραφή του κάθε πεδίου.

Πεδίο	**Εξήγηση**	**Περιγραφή**
opcode	κωδικός λειτουργίας	Καθορίζει την μορφή της εντολής
func	κωδικός πράξης	Πράξη που θα γίνει στη ALU
rs	καταχωρητής πηγής	$1^{ος}$ καταχωρητής που διαβάζεται
rt	καταχωρητής πηγής (R-type) καταχωρητής προορισμού (I-type)	$2^{ος}$ καταχωρητής που διαβάζεται Καταχωρητής που γράφεται
rd	καταχωρητής προορισμού	Καταχωρητής που γράφεται
imm	σταθερά	Σταθερή τιμή δεδομένων
disp	μετατόπιση	Καθορίζει που θα γίνει το άλμα
nu	εκτός χρήσης	Αυτά τα bit δεν χρησιμοποιούνται

Πίνακας 4.2 Εξήγηση πεδίων

Το πεδίο opcode υπάρχει και στις τρεις μορφές εντολών για να καθορίσει τη λειτουργία τους. Ωστόσο για όλες τις R-type εντολές είναι κοινό και οι εντολές αυτές διαχωρίζονται μέσο του πεδίου func. Στον πίνακα 4.3 παρουσιάζεται η DLX κωδικοποίηση για το πεδίο opcode στις εντολές της σχεδίασης.

Πεδίο Opcode	Εντολή
000000	R-type εντολές
001000	addi
001010	subi
001100	andi
001101	ori
001110	xori
000100	beqz
000101	bnez
100011	lw
101011	sw
000010	j
010110	jr
011001	shri
011010	shli
011101	roli
011110	rori
101010	lui

Πίνακας 4.3 Ανάλυση του πεδίου opcode

Στη συνέχεια παρατίθενται οι εντολές που υποστηρίζει η σχεδίαση στους Πίνακες 4.4, 4.5, 4.6.

Μορφή	Εντολή	Σύνταξη	Περιγραφή
I	addi	addi rt,rs,Imm	Προσθέτει rs , Imm αποθηκεύει στον rt
	subi	subi rt,rs,Imm	Αφαιρεί το Imm απο τον rs αποθηκεύει στον rt
	andi	andi rt,rs,Imm	Λογικό ΚΑΙ μεταξύ rs , Imm αποθηκεύει στον rt
	ori	ori rt,rs,Imm	Λογικό Η μεταξύ rs , Imm αποθηκεύει στον rt
	xori	xori rt,rs,Imm	Αποκλειστικό Ή μεταξύ rs, Imm αποθηκεύει στον rt
	beqz	beqz rs,Imm	Αν ο rs είναι μηδέν διακλάδωση σε PC+4+επέκταση προσήμου(Imm)
	bnez	bnez rs,Imm	Αν ο rs δεν είναι μηδέν διακλάδωση σε PC+4+επέκταση προσήμου(Imm)
	lw	lw rt,Imm(rs)	Φόρτωση του rt από την Imm+rs διεύθυνση της μνήμης δεδομένων
	sw	sw rt,Imm(rs)	Αποθήκευση του rt στην Imm+rs διεύθυνση της μνήμης δεδομένων
	shri	shri rd,rs, Imm	Ολίσθηση του rs κατά Imm[5-0] θέσεις δεξιά, αποθηκεύει στον rd
	shli	shli rd,rs, Imm	Ολίσθηση του rs κατά Imm[5-0] θέσεις αριστερά, αποθηκεύει στον rd
	rori	rori rd,rs, Imm	Κυκλική ολίσθηση του rs κατα Imm[5-0] θέσεις δεξιά, αποθηκεύει στον rd
	roli	roli rd,rs, Imm	Κυκλική ολίσθηση του rs κατά Imm[5-0] θέσεις αριστερά, αποθηκεύει στον rd
	lui	lui rt,Imm	Φόρτωση των 16 msb του rt με το Imm

Πίνακας 4.4 I-Type εντολές που υποστηρίζει η σχεδίαση

Μορφή	Εντολή	Σύνταξη	Περιγραφή
R	add	add rd,rs,rt	Προσθέτει rs , rt αποθηκεύει στον rd
	sub	sub rd,rs,rt	Αφαιρεί τον rt απο τον rs αποθηκεύει στον rd
	or	or rd,rs,rt	Λογικό Ή μεταξύ rs , rt αποθηκεύει στον rd
	and	and rd,rs,rt	Λογικό ΚΑΙ μεταξύ rs , rt αποθηκεύει στον rd
	xor	xor rd,rs,rt	Αποκλειστικό Ή μεταξύ rs , rt αποθηκεύει στον rd
	shr	shr rd,rs,rt	Ολίσθηση του rs κατά rt θέσεις δεξιά, αποθηκεύει στον rd
	shl	shl rd,rs,rt	Ολίσθηση του rs κατά rt θέσεις αριστερά, αποθηκεύει στον rd
	ror	ror rd,rs,rt	Κυκλική ολίσθηση του rs κατά rt θέσεις δεξιά, αποθηκεύει στον rd
	rol	rol rd,rs,rt	Κυκλική ολίσθηση του rs κατά rt θέσεις αριστερά, αποθηκεύει στον rd

Πίνακας 4.5 R-Type εντολές που υποστηρίζει η σχεδίαση

Μορφή	Εντολή	Σύνταξη	Περιγραφή
J	j	j label	Άλμα στη διεύθυνση «label»
	jr	jr rs	Άλμα στη διεύθυνση που περιέχει ο rs

Πίνακας 4.6 J-Type εντολές που υποστηρίζει η σχεδίαση

4.3 Δομή του Μονοπατιού Δεδομένων

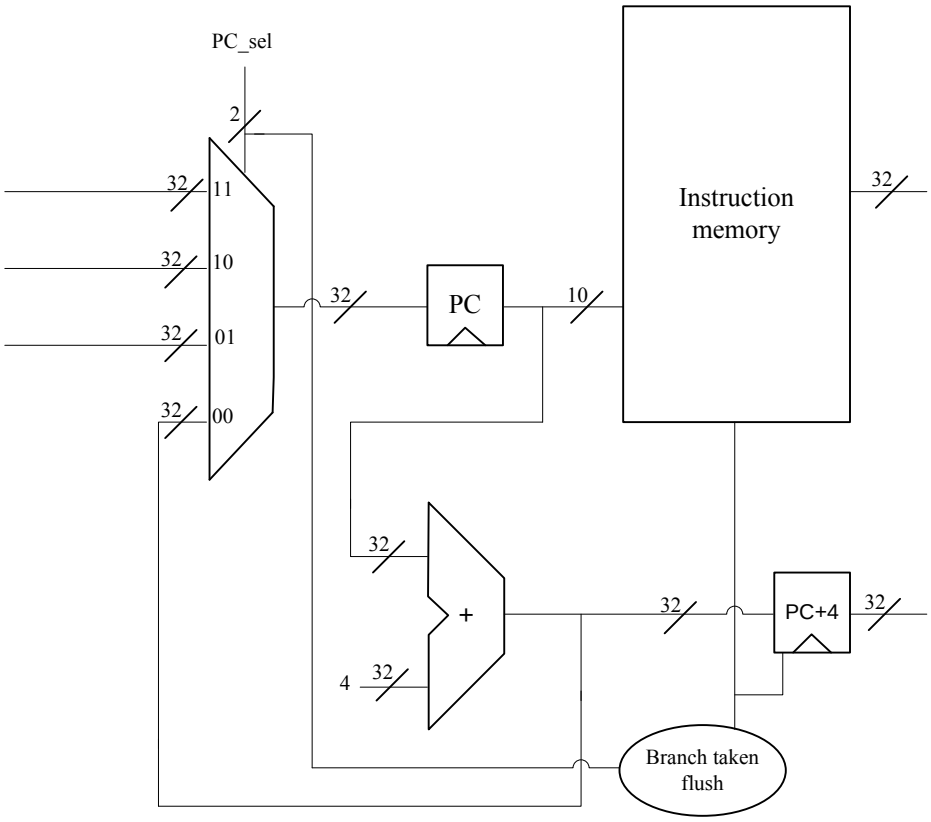
Στο κεφάλαιο αυτό παρουσιάζονται οι 5 βασικές βαθμίδες που αναφέρθηκαν παραπάνω καθώς και κάποια βασικά υποκυκλώματα που μπορεί να διαθέτουν. Επίσης παρουσιάζεται η μονάδα εισαγωγής «φυσαλίδας» και ελέγχου ανάσχεσης της ομοχειρίας που τοποθετείται στο υψηλότερο επίπεδο της σχεδίασης.

4.3.1 Βαθμίδα ανάκλησης εντολών

Όπως φαίνεται στο Σχήμα 4.1 στη βαθμίδα αυτή υπάρχουν τα εξής κυκλώματα:

- Η μνήμη εντολών 32 bit, 1024 θέσεων, μιας θύρας ανάγνωσης που δημιουργήθηκε με το core generator του εργαλείου ISE 7.1 της Xilinx. Να σημειωθεί ότι η έξοδος της μνήμης χρησιμοποιείται σαν ομόχειρος καταχωρητής.
- Ο μετρητής προγράμματος που κρατάει τη διεύθυνση της επόμενης προς ανάκληση εντολής.
- Ένας αθροιστής που αυξάνει την τιμή του μετρητή προγράμματος κατά 4 (επόμενη εντολή).

- Ένας πολυπλέκτης 4 σε 1 με εισόδους την διεύθυνση της επόμενης εντολής στην μνήμη εντολών (έξοδος του αθροιστή), τις διευθύνσεις διακλάδωσης που προκύπτουν από εντολές διακλάδωσης, άλματος και άλματος σε καταχωρητή στη βαθμίδα εκτέλεσης εντολών. Ο επιλογέας του πολυπλέκτη έρχεται επίσης από τη βαθμίδα εκτέλεσης εντολών όπου και αποφασίζεται η πραγματοποίηση και ο τύπος της διακλάδωσης.
- Ένας καταχωρητής για να κρατάει την αυξημένη κατά 4 τιμή του μετρητή προγράμματος για τα επόμενα σταδία του ομόχειρου μονοπατιού δεδομένων.
- Ένα κύκλωμα εισαγωγής «φυσαλίδας» στο μονοπάτι δεδομένων που διαγράφει τις εντολές που ακολουθούν επιτυχή διακλάδωση και δεν πρέπει να εκτελεστούν. Η «φυσαλίδα» υλοποιείται μηδενίζοντας τον ομόχειρο καταχωρητή και την έξοδο της μνήμης.



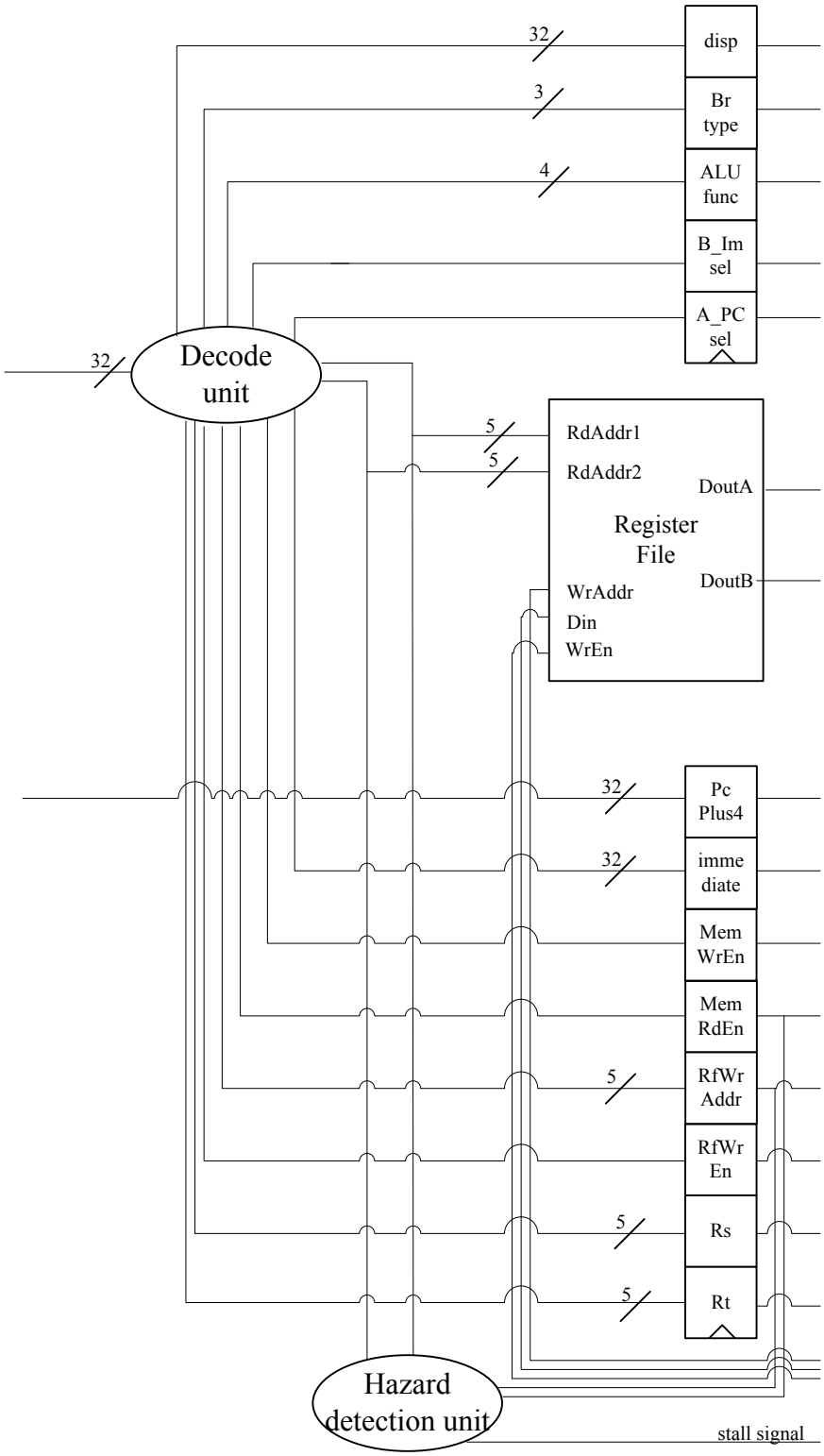
Σχήμα 4.1 Βαθμίδα ανάκλησης εντολών

Η βαθμίδα προωθεί στο επόμενο στάδιο της ομοχειρίας την εντολή που ανακλήθηκε και τον αυξημένο κατά 4 μετρητή προγράμματος.

4.3.2 Βαθμίδα αποκωδικοποίησης εντολών

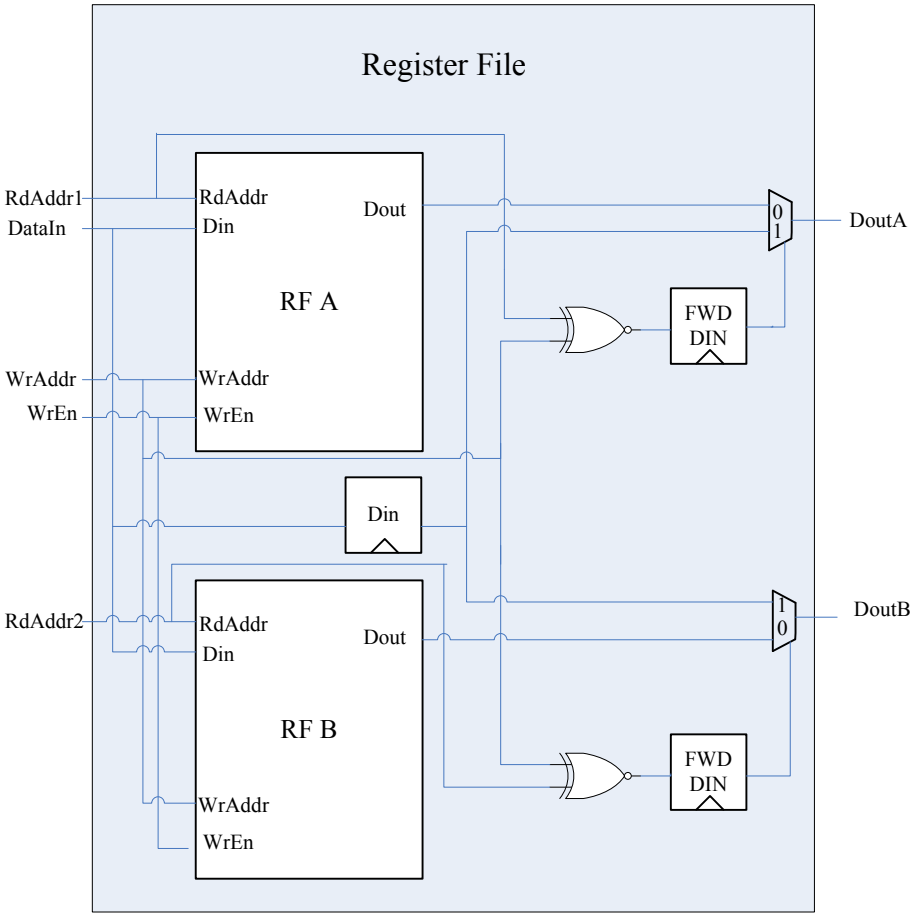
Σε αυτή τη βαθμίδα που φαίνεται στο Σχήμα 4.2 υπάρχουν τα εξής κυκλώματα:

- Το κύκλωμα αποκωδικοποίησης της εντολής και παραγωγής των κατάλληλων σημάτων για τα υπόλοιπα κυκλώματα της βαθμίδας.
- Το αρχείο καταχωρητών που οι έξοδοι του θεωρούνται ομόχειροι καταχωρητές. Όπως φαίνεται και στο Σχήμα 4.2 το αρχείο των 32 καταχωρητών έχει δύο θύρες ανάγνωσης και μία θύρα εγγραφής. Πιο συγκεκριμένα στο Σχήμα 4.3 φαίνεται ότι για την υλοποίηση του χρησιμοποιήθηκαν 2 αντίγραφα μνήμης 32 bit, 32 θέσεων που δημιουργήθηκε με το core generator του εργαλείου ISE 7.1 της Xilinx. Επίσης φαίνεται η λογική που υλοποιεί μια εσωτερική προώθηση δεδομένων στο αρχείο καταχωρητών εξαλείφοντας έτσι τους δομικούς κινδύνους που μπορεί να προκύψουν.
- Οι ομόχειροι καταχωρητές που προωθούν στη βαθμίδα εκτέλεσης εντολών, τον αυξημένο μετρητή προγράμματος, την σταθερά στην οποία έχει γίνει επέκταση προσήμου (όταν είναι απαραίτητο), ένα επιλογέα για τα δεδομένα της πρώτης θύρας ανάγνωσης του αρχείου καταχωρητών ή του μετρητή προγράμματος, ένα επιλογέα για τα δεδομένα της δεύτερης θύρας ανάγνωσης του αρχείου καταχωρητών ή της σταθεράς, την πράξη που θα γίνει στην αριθμητική λογική μονάδα, τον τύπο της διακλάδωσης, τη μετατόπιση για εντολές άλματος, την διεύθυνση εγγραφής στο αρχείο καταχωρητών, την σημαία ενεργοποίησης της εγγραφής στο αρχείο καταχωρητών, την σημαία ενεργοποίησης της εγγραφής στη μνήμη δεδομένων, την σημαία ενεργοποίησης της ανάγνωσης από τη μνήμη δεδομένων και τέλος τις διευθύνσεις των δύο καταχωρητών πηγής που θα χρειαστούν στην επόμενη βαθμίδα για το κύκλωμα προώθησης δεδομένων.
- Το κύκλωμα παραγωγής σήματος ανάσχεσης της ομοχειρίας σε περίπτωση κινδύνου δεδομένων λόγω εντολής φόρτωσης. Τον κίνδυνο τον ανιχνεύουμε εξετάζοντας τους ομόχειρους καταχωρητές που κρατούν την σημαία ενεργοποίησης της ανάγνωσης από τη μνήμη δεδομένων και την διεύθυνση εγγραφής στο αρχείο καταχωρητών και συγκρίνοντας την δεύτερη με τις διευθύνσεις των καταχωρητών πηγής. Έτσι στον επόμενο κύκλο είναι εφικτή η σωστή προώθηση δεδομένων.



Σχήμα 4.2 Βαθμίδα αποκωδικοποίησης εντολών

Να σημειωθεί ότι στο Σχήμα 4.2 τα δεδομένα κυκλοφορούν από αριστερά προς τα δεξιά εκτός από αυτά που καταλήγουν στο αρχείο καταχωρητών για την εγγραφή δεδομένων και αυτά που καταλήγουν στο κύκλωμα παραγωγής σήματος ανάσχεσης από τους ομόχειρους καταχωρητές.



Σχήμα 4.3 Αρχείο καταχωρητών

4.3.3 Βαθμίδα εκτέλεσης εντολών

Όπως φαίνεται στο Σχήμα 4.4 στη βαθμίδα αυτή υπάρχουν τα εξής κυκλώματα:

- Μια αριθμητική λογική μονάδα στην οποία γίνονται οι πράξεις της πρόσθεσης, της αφαίρεσης, του λογικού και, του λογικού ή και του αποκλειστικού ή. Η πρώτη είσοδος της μονάδας επιλέγεται από ένα πολυπλέκτη 2 σε 1 ανάμεσα στο αποτέλεσμα του δικτύου προώθησης δεδομένων και του αυξημένου μετρητή προγράμματος. Η δεύτερη είσοδος της μονάδας επιλέγεται από ένα πολυπλέκτη 2 σε 1 ανάμεσα στο αποτέλεσμα του δικτύου προώθησης δεδομένων και την σταθερά.
- Το κύκλωμα προώθησης δεδομένων παίρνει εισόδους από ομόχειρους καταχωρητές τριών διαφορετικών βαθμίδων και επιλύει τους κινδύνους δεδομένων. Η προώθηση πρέπει να γίνεται και για τους δύο καταχωρητές πηγής και για αυτό έχουν αποθηκευτεί σε ομόχειρους καταχωρητές στη βαθμίδα αποκωδικοποίησης. Ο πρώτος έλεγχος που γίνεται για τους δύο αυτούς

καταχωρητές είναι ο εξής, αν κάποιος από αυτούς είναι ίσος με τον καταχωρητή προορισμού της βαθμίδας εκτέλεσης εντολών που δεν είναι ο καταχωρητής μηδέν και επιπλέον είναι ενεργοποιημένη η σημαία εγγραφής στο αρχείο καταχωρητών που βρίσκεται σε καταχωρητή της ίδιας βαθμίδας τότε προωθούμε το αποτέλεσμα που βρίσκεται στον καταχωρητή που κρατάει το προηγούμενο αποτέλεσμα της αριθμητικής λογικής μονάδας. Αν αποτύχει ο παραπάνω έλεγχος κάνουμε ένα δεύτερο αυτή τη φορά συγκρίνοντας τους ομόχειρους καταχωρητές του σταδίου αποκωδικοποίησης με αυτούς του σταδίου πρόσβασης μνήμης και αν έχουμε επιτυχία προωθούμε το αποτέλεσμα από αυτή τη βαθμίδα. Αν αποτύχουν οι παραπάνω έλεγχοι τότε δεν υπάρχει λόγος προώθησης και στην αριθμητική λογική μονάδα εισέρχονται τα δεδομένα από το αρχείο καταχωρητών.

- Το κύκλωμα ανίχνευσης επιτυχής διακλάδωσης το οποίο παίρνει τον τύπο της διακλάδωσης από την προηγούμενη βαθμίδα και κάνει τους απαραίτητους ελέγχους βγάζοντας ως έξοδο τον επιλογέα του πολυπλέκτη του μετρητή προγράμματος για την βαθμίδα ανάκλησης εντολών.

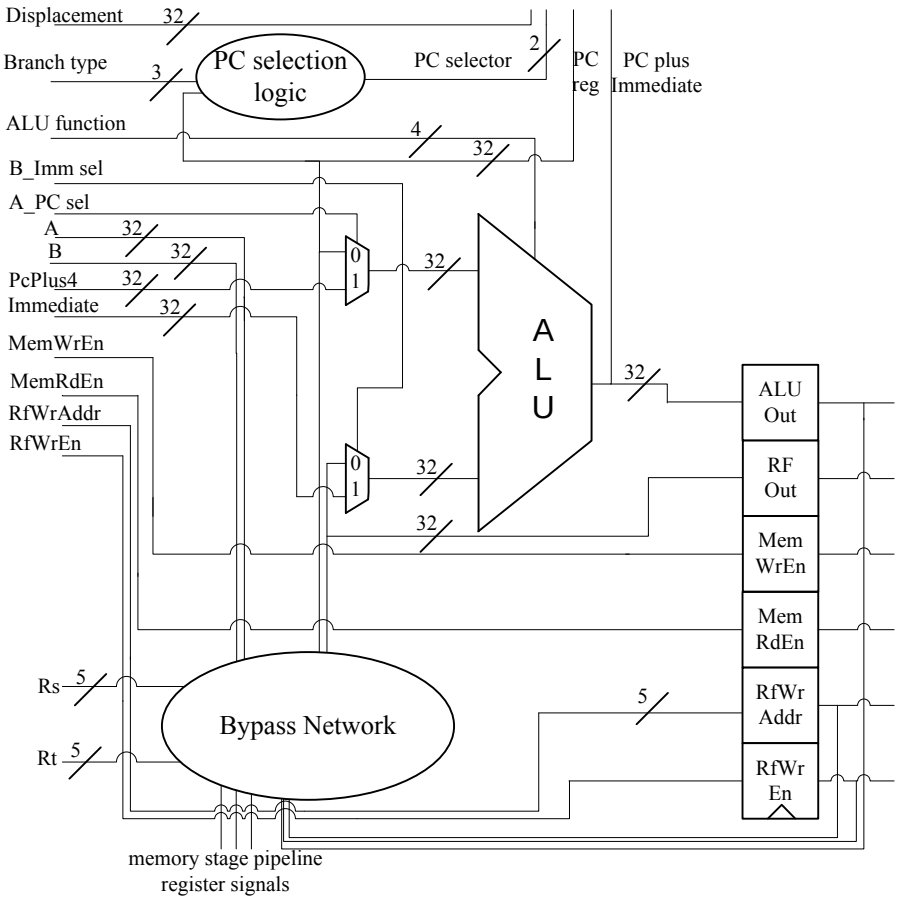
- Τους ομόχειρους καταχωρητές που προωθούν στη βαθμίδα πρόσβασης μνήμης, την διεύθυνση εγγραφής στο αρχείο καταχωρητών, την σημαία ενεργοποίησης της εγγραφής στο αρχείο καταχωρητών, την σημαία ενεργοποίησης της εγγραφής στη μνήμη δεδομένων, την σημαία ενεργοποίησης της ανάγνωσης από τη μνήμη δεδομένων, τα δεδομένα για εγγραφή στην μνήμη και το αποτέλεσμα της αριθμητικής λογικής μονάδας.

4.3.4 Βαθμίδα πρόσβασης μνήμης

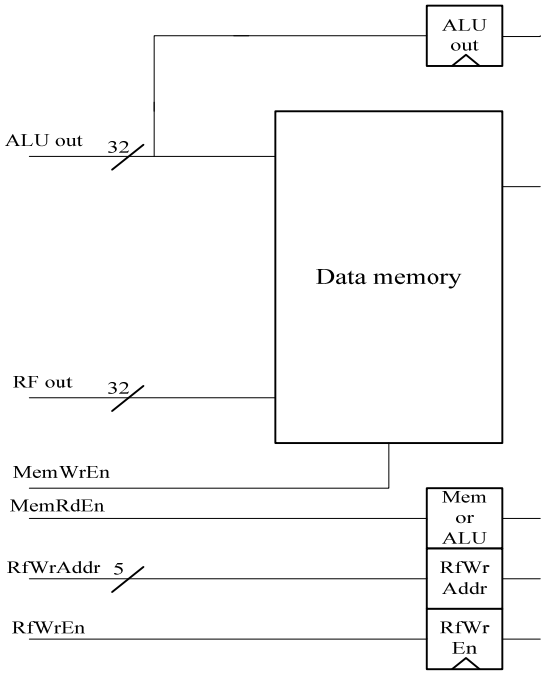
Σε αυτή τη βαθμίδα που φαίνεται στο Σχήμα 4.5 υπάρχουν τα εξής κυκλώματα:

- Η μνήμη δεδομένων 1024 θέσεων, των 32 bit, μιας θύρας ανάγνωσης και μιας θύρας εγγραφής που δημιουργήσαμε με το core generator του εργαλείου ISE 7.1 της Xilinx.

- Τους ομόχειρους καταχωρητές που προωθούν στη βαθμίδα εγγραφής αποτελέσματος, την διεύθυνση εγγραφής στο αρχείο καταχωρητών, την σημαία ενεργοποίησης της εγγραφής στο αρχείο καταχωρητών, το αποτέλεσμα της αριθμητικής λογικής μονάδας και μια σημαία για τα δεδομένα που θα εγγραφούν στο αρχείο καταχωρητών (δεδομένα από την ALU ή την μνήμη).



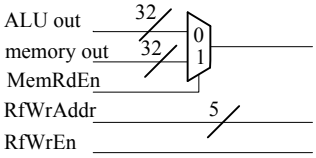
Σχήμα 4.4 Βαθμίδα εκτέλεσης εντολών



Σχήμα 4.5 Βαθμίδα πρόσβασης μνήμης

4.3.5 Βαθμίδα εγγραφής αποτελέσματος

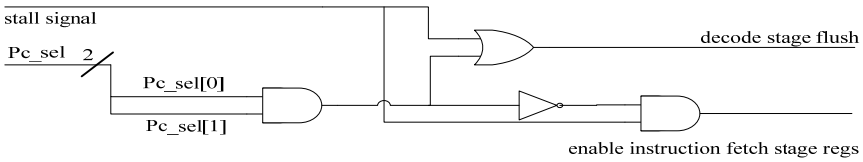
Όπως φαίνεται στο Σχήμα 4.6 στη βαθμίδα αυτή υπάρχει μόνο ένα κύκλωμα. Ένας πολυπλέκτης που επιλέγει αν τα δεδομένα για το αρχείο καταχωρητών είναι από την μνήμη ή από την αριθμητική λογική μονάδα. Τα σήματα για την διεύθυνση και την ενεργοποίηση εγγραφής απλά προωθούνται από την είσοδο στην έξοδο της βαθμίδας.



Σχήμα 4.6 Βαθμίδα εγγραφής αποτελέσματος

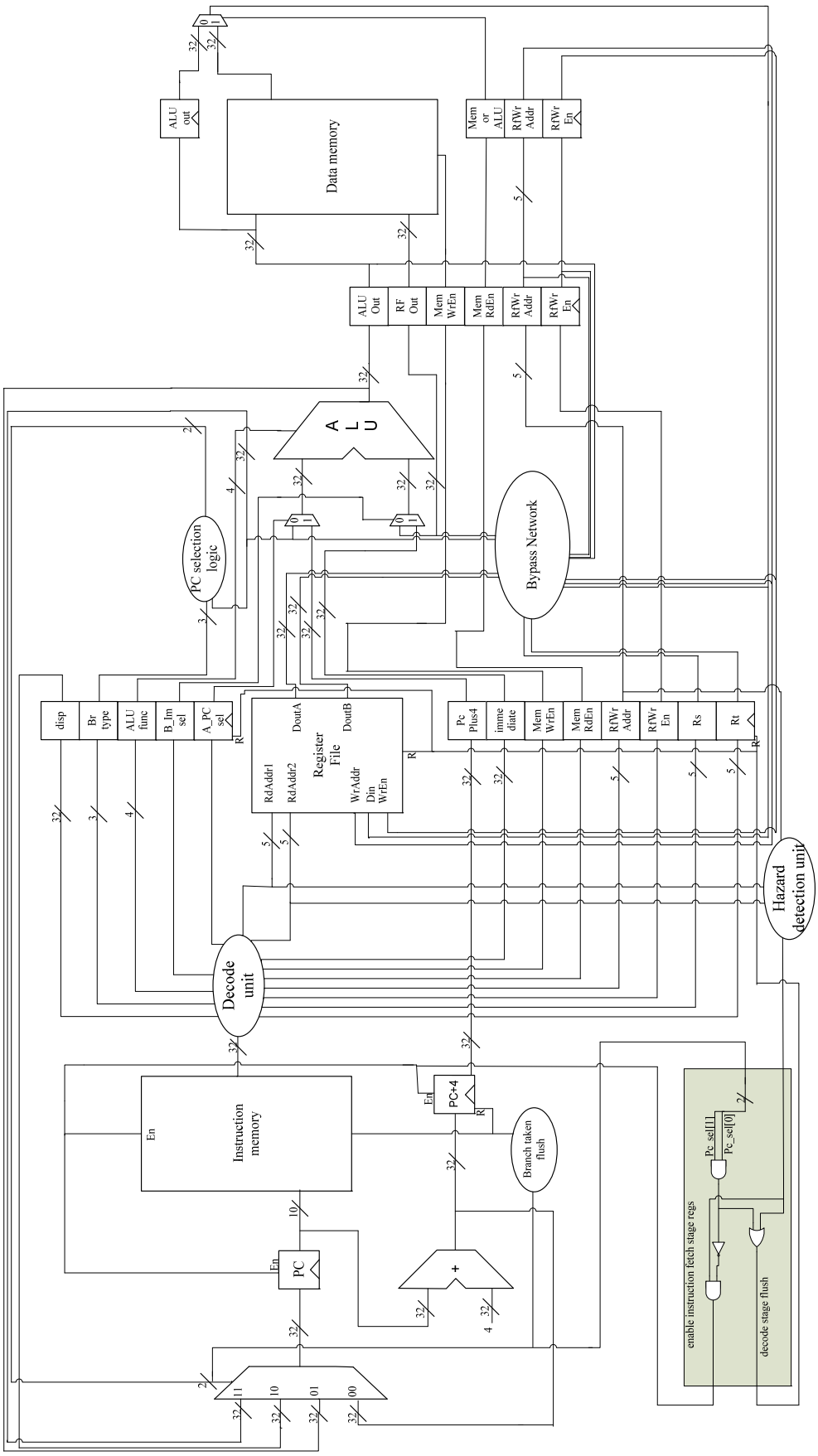
4.3.6 Πλήρες μονοπάτι δεδομένων της σχεδίασης

Το υψηλότερο επίπεδο της σχεδίασης, όπου ενώνονται όλες οι επιμέρους βαθμίδες που αναλύθηκαν, παρουσιάζεται στο Σχήμα 4.8. Σε αυτό το επίπεδο υπάρχει και η λογική για την εισαγωγή «φυσαλίδας» και τον έλεγχο ανάσχεσης της ομοχειρίας. Το κύκλωμα που υλοποιεί τη συγκεκριμένη λογική παρουσιάζεται στο Σχήμα 4.7.



Σχήμα 4.7 Κύκλωμα εισαγωγής «φυσαλίδας» και ελέγχου ανάσχεσης της ομοχειρίας

Η μονάδα αυτή ελέγχει αν έχει ανέβει το σήμα για ανάσχεση από τη βαθμίδα αποκωδικοποίησης εντολών. Αν έχει συμβεί αυτό τότε ή υπάρχει επιτυχής διακλάδωση, στέλνει το κατάλληλο σήμα έτσι ώστε να μηδενιστούν οι ομόχειροι καταχωρητές της βαθμίδας αποκωδικοποίησης εντολών. Αν έχει ανέβει το σήμα για ανάσχεση και δεν υπάρχει επιτυχής διακλάδωση το κύκλωμα στέλνει το κατάλληλο σήμα έτσι ώστε να μην φορτωθούν νέες τιμές στους καταχωρητές της βαθμίδας ανάκλησης εντολών (σταματάει η έκδοση εντολών) καθώς και στον μετρητή προγράμματος . Τα παραπάνω εξασφαλίζουν ότι στην περίπτωση που έχει ανέβει το σήμα για ανάσχεση αλλά προηγείται επιτυχής διακλάδωση οι εντολές συνεχίζουν να εκδίδονται και γίνεται μόνο ο μηδενισμός των ομόχειρων καταχωρητών. Σημειώνεται ότι η λογική για τη «φυσαλίδα» στη βαθμίδα ανάκλησης εντολών λόγω επιτυχής διακλάδωσης υλοποιείται στο εσωτερικό της μια και ο επιλογέας για τον μετρητή προγράμματος καταλήγει εκεί.



Σχήμα 4.8 Πλήρες μονοπάτι δεδομένων της σχεδίασης. Παρουσιάζονται όλες οι ενώσεις των επιμέρους βαθμίδων και το κύκλωμα εισαγωγής «φυσαλίδας» και ελέγχου ανάσχεσης της ομοχειρίας στο γκρι πλαίσιο

5. Αρχιτεκτονική Συνόλου Εντολών του CryptoDLX

Το κεφάλαιο αυτό επικεντρώνεται στην αρχιτεκτονική του CryptoDLX. Όπως υπονοεί και το όνομα του πρόκειται για ένα επεξεργαστή για εφαρμογές κρυπτογράφησης βασισμένο στο DLX. Αρχικά θα γίνει μια μελέτη για το είδος των εντολών που πρέπει να προστεθούν στην υλοποίηση που έχει παρουσιαστεί στο προηγούμενο κεφάλαιο. Από αυτή τη μελέτη θα προκύψει το σύνολο εντολών του CryptoDLX. Ακολούθως θα παρουσιαστούν βήμα βήμα οι αναγκαίες τροποποιήσεις τόσο στο μονοπάτι δεδομένων όσο και στην αρχιτεκτονική συνόλου εντολών του DLX.

5.1 Μελέτη Σχεδίασης του CryptoDLX

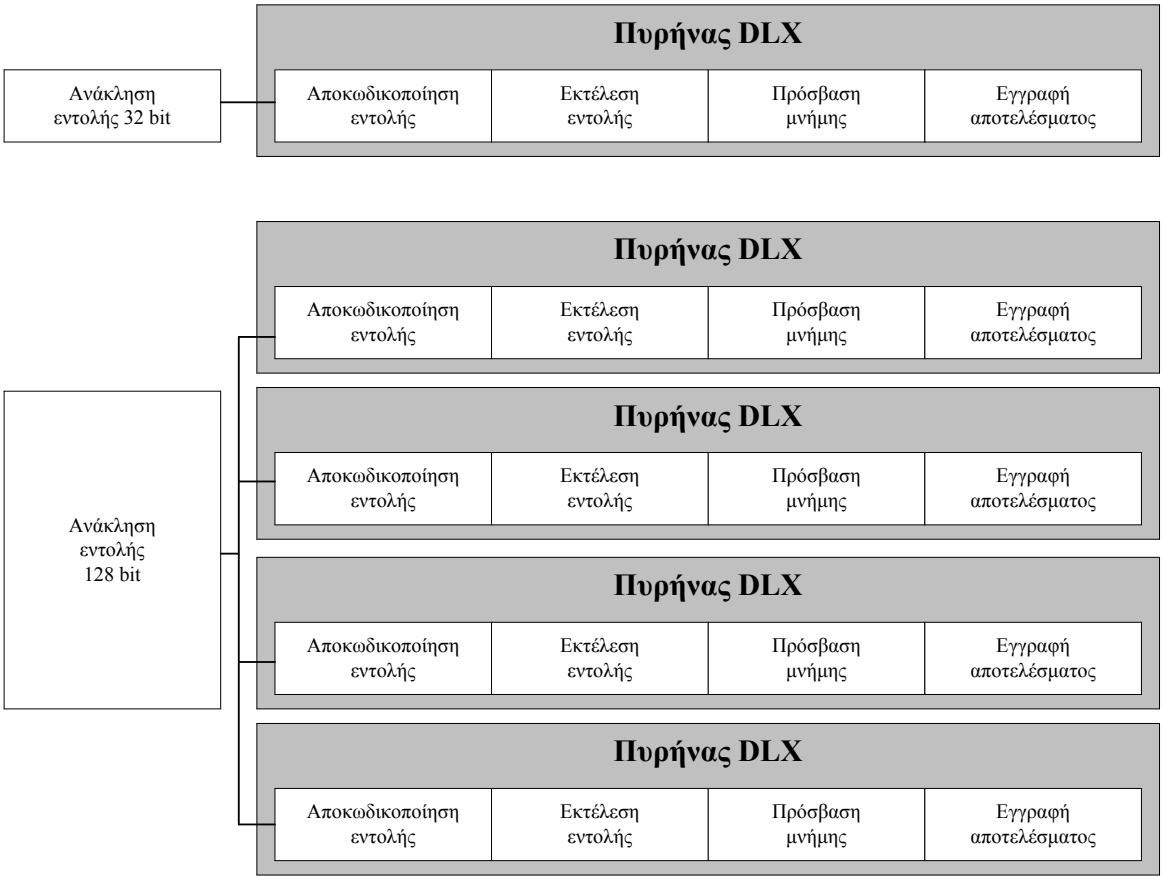
Μετά από μελέτη των βασικότερων ιδιοτήτων των συμμετρικών κρυπτογραφικών αλγορίθμων προέκυψε το συμπέρασμα ότι στην αρχιτεκτονική συνόλου εντολών του DLX αρχικά θα πρέπει να προστεθούν οι εξής εντολές:

1. Εντολές αριθμητικών και λογικών πράξεων μεταξύ τριών καταχωρητών πηγής.
2. Μια εντολή πολλαπλασιασμού modulo 32.
3. Εντολές για την επίτευξη πολλαπλασιασμού σε ένα πεπερασμένο πεδίο modulo ένα πρώτο πολυώνυμο.
4. Εντολές πρόσβασης σε πίνακες αντικατάστασης (S-boxes).

Στη συνέχεια κρίθηκε απαραίτητο στη σχεδίαση να προστεθεί ένα ειδικό αρχείο καταχωρητών για την αποθήκευση των κυκλικών κλειδιών που προκύπτουν από την διαδικασία επέκτασης του μυστικού κλειδιού. Συνεπώς για την εκμετάλλευση του αρχείου καταχωρητών των κλειδιών θα πρέπει να προστεθούν εντολές στην αρχιτεκτονική που να κάνουν αριθμητικές και λογικές πράξεις μεταξύ δύο ή ενός καταχωρητή από το κλασικό αρχείο καταχωρητών και ενός καταχωρητή από το αρχείο καταχωρητών των κλειδιών.

Λόγω του γεγονότος ότι οι συμμετρικοί κρυπτογραφικοί αλγόριθμοι λειτουργούν πάνω σε μπλοκ μεγέθους 128 bit και στον DLX οι πράξεις γίνονται πάνω σε δεδομένα μεγέθους 32 bit κρίθηκε ότι το υπάρχον μονοπάτι δεδομένων δεν είναι το αποδοτικότερο για μια τέτοια λειτουργία. Αυτό συμβαίνει γιατί οι ίδιες πράξεις γίνονται διαδοχικά 4 φόρες για να επιτευχθεί η επιθυμητή λειτουργία σε όλο το μπλοκ. Για την βελτιστοποίηση της απόδοσης, σε τέτοιου τύπου αλγόριθμους, σχεδιάστηκε ένα πιο περίπλοκο μονοπάτι δεδομένων που λειτουργεί σε όλο το μπλοκ. Χρησιμοποιώντας 4 αντίγραφα του πυρήνα

του DLX που λειτουργούν παράλληλα και μία καινούρια βαθμίδα ανάκλησης εντολής 128 bit δημιουργήθηκε ο VLIW επεξεργαστής CryptoDLX. Στο Σχήμα 5.1 παρουσιάζεται η δομή της DLX και της CryptoDLX σχεδίασης και πώς η δεύτερη εκμεταλλεύεται τον πυρήνα της πρώτης.



Σχήμα 5.1 DLX επεξεργαστής (πάνω), VLIW CryptoDLX επεξεργαστής (κάτω)

Λόγώ του ότι η βαθμίδα ανάκλησης εντολών 32 bit δεν υπάρχει στο νέο μονοπάτι δεδομένων από τον πυρήνα του DLX μπορεί να αφαιρεθεί η λογική που υποστηρίζει τις εντολές ελέγχου (j, jr, beqz, bnez) οι οποίες δεν έχουν νόημα ύπαρξης πλέον. Μοναδική εντολή ελέγχου στη νέα σχεδίαση θα είναι μια διακλάδωση εφόσον ένας καταχωρητής, που κρατάει του κύκλους επεξεργασίας του κρυπτογραφικού αλγορίθμου, δεν έχει μηδενιστεί. Η λογική για την εντολή αυτή υλοποιείται στην βαθμίδα ανάκλησης εντολής 128 bit και οδηγεί στην ανάκληση μιας παλιότερης 128 bit εντολής ή στην επόμενη 128 bit εντολή.

Κλείνοντας πρέπει να τονιστεί ότι είναι πολύ χρήσιμο για τους συμμετρικούς κρυπτογραφικούς αλγορίθμους να μπορεί να γίνει μέσο μιας εντολής μετακίνηση δεδομένων μεταξύ των πυρήνων και μέσο μιας άλλης εντολής αναστροφή ενός πίνακα 4

επί 4. Ως τέτοιος πίνακας μπορούν να θεωρηθούν οι 4 λέξεις των 4 byte που είναι για επεξεργασία στο μονοπάτι δεδομένων.

5.2 Ανάπτυξη του CryptoDLX επεξεργαστή

Στην ενότητα αυτή θα παρουσιαστεί όλη η διαδικασία ανάπτυξης του CryptoDLX επεξεργαστή ξεκινώντας από το μονοπάτι δεδομένων και την αρχιτεκτονική συνόλου εντολών του απλού DLX. Στις πρώτες υποενότητες που ακολουθούν θα παρουσιασθούν οι προσθήκες που θα γίνουν στον πυρήνα του DLX που στην ουσία είναι ένα 32 bit μονοπάτι δεδομένων. Στις τελευταίες υποενότητες θα παρουσιασθεί ο VLIW CryptoDLX που θα προκύψει από την ένωση των 4 αντιγράφων του πυρήνα του DLX μαζί με όλες τις απαραίτητες προσθήκες για την υλοποίηση ενός συνόλου εντολών ικανού να ανταποκριθεί στις ανάγκες των σύγχρονων κρυπτογραφικών εφαρμογών.

5.2.1 Υποστήριξη εντολών τριών καταχωρητών πηγής

Για την κωδικοποίηση τέτοιων εντολών χρησιμοποιείται η μορφή R-Type. Συγκεκριμένα το πεδίο nu που είναι εκτός χρήσης για τις R-Type εντολές με δύο καταχωρητές πηγής χρησιμοποιείται για τον τρίτο καταχωρητή πηγής που ονομάζεται rs2. Από τα παραπάνω προκύπτει η νέα μορφή των R-Type εντολών που φαίνεται στον Πίνακα 5.1.

31-26	25-21	20-16	15-11	10-6	5-0
opcode	Rs	Rt	rd	**rs2**	func

Πίνακας 5.1 R-Type μορφή για την υποστήριξη εντολών τριών καταχωρητών πηγής

Οι εντολές τριών καταχωρητών πηγής που θα προστεθούν στη σχεδίαση συνδυάζουν δυο αριθμητικές πράξεις ή δύο λογικές ή μια αριθμητική και μια λογική.

Εντολή	Σύνταξη	Περιγραφή
addadd	addadd rd,rs,rt,rs2	Προσθέτει rs, rt, rs2, αποθηκεύει στον rd
subadd	subadd rd,rs,rt,rs2	Αφαιρεί τον rt απο τον rs, προσθέτει τον rs2 στο αποτέλεσμα της αφαίρεσης και αποθηκεύει στον rd
addsub	addsub rd,rs,rt,rs2	Προσθέτει rs και rt, αφαιρεί τον rs2 από το αποτέλεσμα της πρόσθεσης, αποθηκεύει στον rd
subsub	subsub rd,rs,rt,rs2	Αφαιρεί τον rt απο τον rs, αφαιρεί τον rs2 από το αποτέλεσμα της αφαίρεσης και αποθηκεύει στον rd

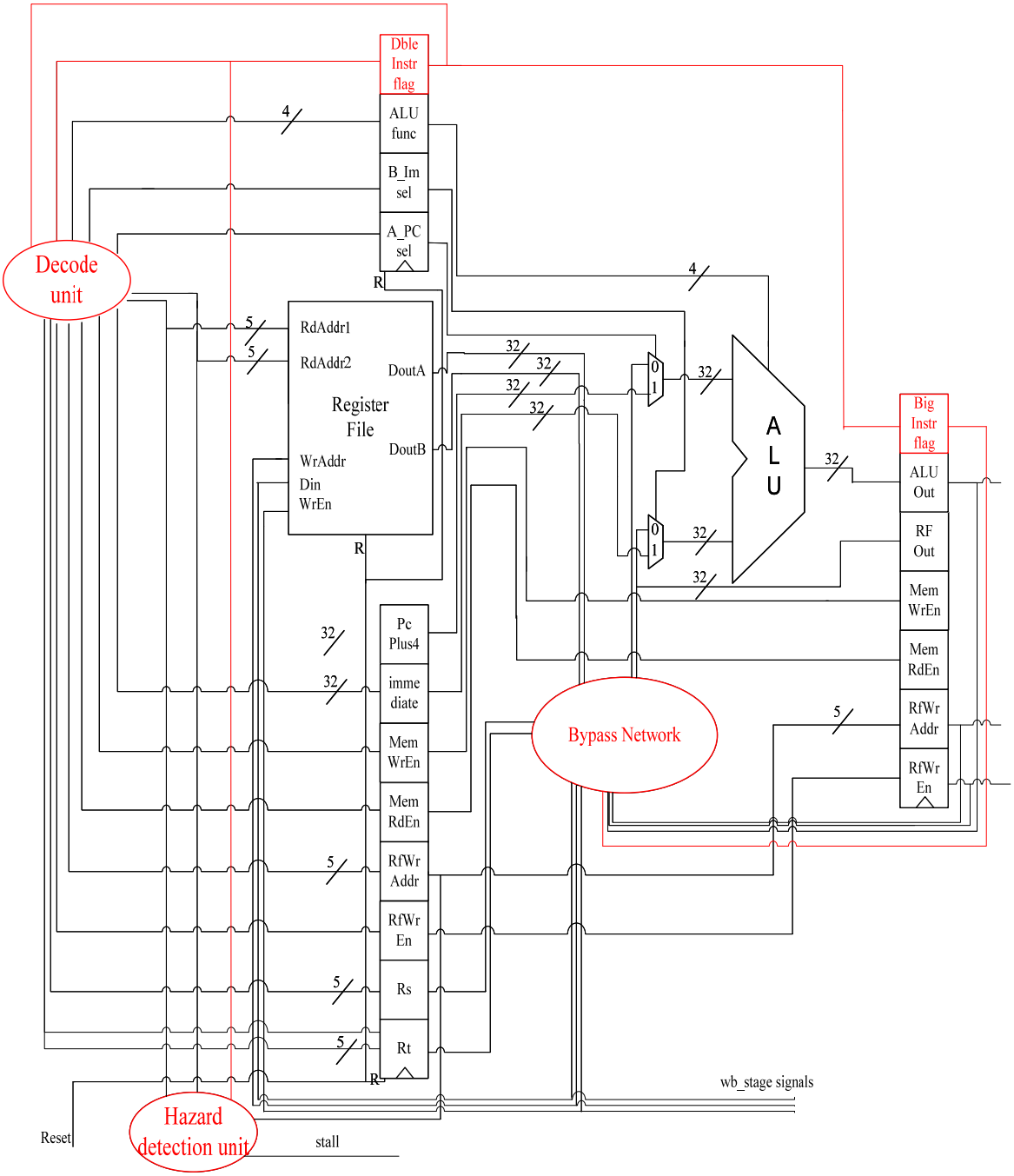
Πίνακας 5.2 R-Type εντολές που συνδυάζουν 2 αριθμητικές πράξεις

Εντολή	Σύνταξη	Περιγραφή
xoradd	xoradd rd,rs,rt,rs2	Λογικό xor μεταξύ rs, rt, προσθέτει τον rs2 στο αποτέλεσμα της λογικής πράξης, αποθηκεύει στον rd
xorsub	xorsub rd,rs,rt,rs2	Λογικό xor μεταξύ rs, rt, αφαιρεί τον rs2 από το αποτέλεσμα της λογικής πράξης, αποθηκεύει στον rd
addxor	addxor rd,rs,rt,rs2	Προσθέτει rs και rt, λογικό xor μεταξύ του rs2 και του αποτελέσματος της πρόσθεσης, αποθηκεύει στον rd
subxor	subxor rd,rs,rt,rs2	Αφαιρεί τον rt απο τον rs, λογικό xor μεταξύ του rs2 και του αποτελέσματος της αφαίρεσης, αποθηκεύει στον rd
xorxor	xorxor rd,rs,rt,rs2	Λογικό xor μεταξύ rs,rt,rs2

Πίνακας 5.3 R-Type εντολές που συνδυάζουν 1 αριθμητική και μία λογική πράξη ή δυο λογικές

Για την υποστήριξη αυτών των εντολών σχεδιάστηκαν και υλοποιήθηκάν 2 εναλλακτικές δομές του μονοπατιού δεδομένων. Η πρώτη υλοποίηση εκμεταλλεύεται τη δομή του μονοπατιού δεδομένων και με την προσθήκη μιας σχετικά απλής λογικής προκαλεί ανάσχεση της ομοχειρίας για ένα κύκλο ρολογιού στον οποίο διαβάζεται ο τρίτος καταχωρητής πηγής που στην συνέχεια μπαίνει στη βαθμίδα εκτέλεσης εντολών για να γίνει η πράξη με το αποτέλεσμα της πράξης των δύο πρώτων καταχωρητών πηγής. Η δεύτερη υλοποίηση τροποποιεί τη δομή του μονοπατιού δεδομένων προσθέτοντας μια επιπλέον θύρα στο αρχείο καταχωρητών και στην αριθμητική λογική μονάδα.

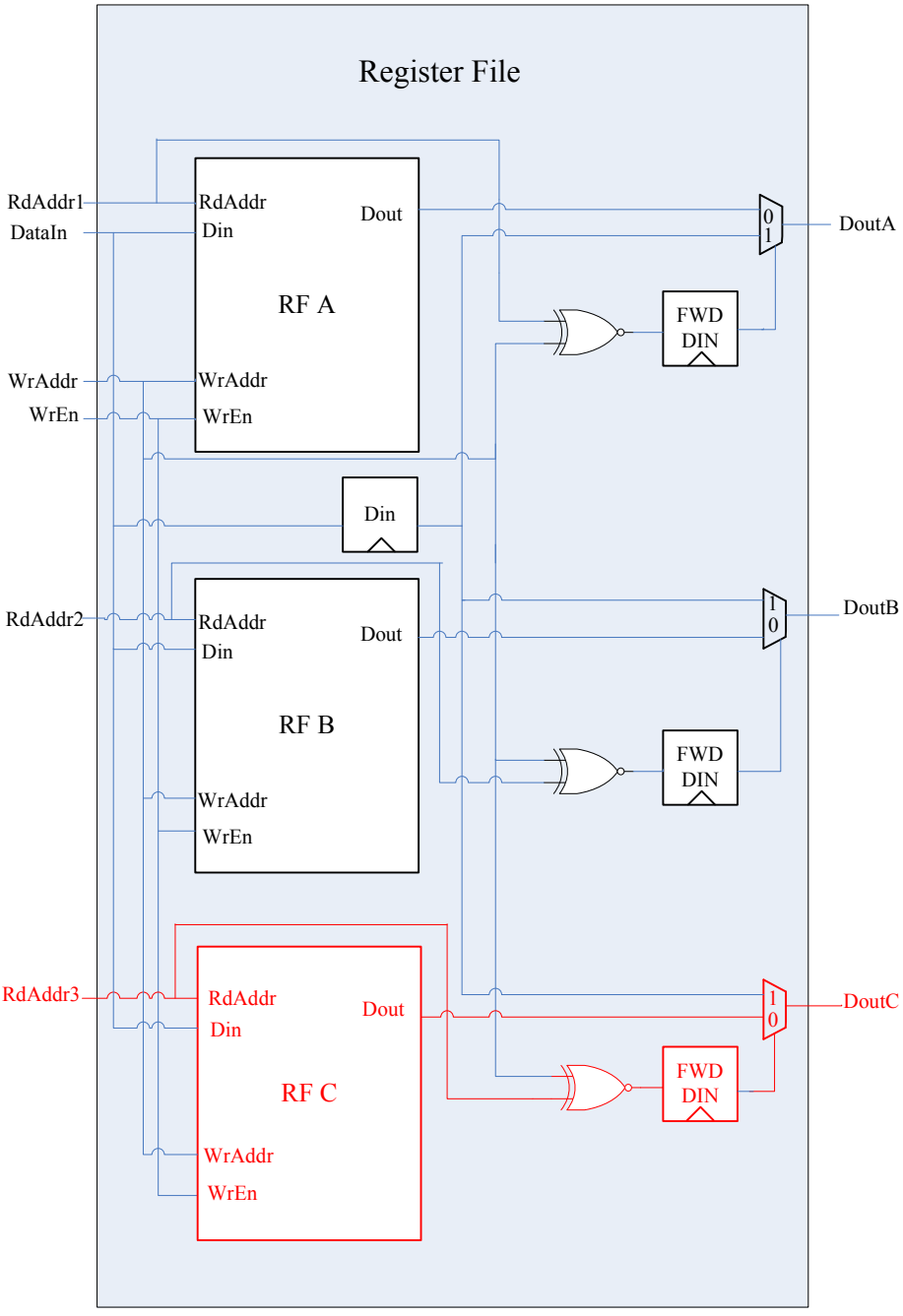
Όπως αναφέρθηκε και παραπάνω για να επιτευχθεί η ανάγνωση τριών καταχωρητών από ένα αρχείο καταχωρητών που έχει δύο πόρτες ανάγνωσης εισάγεται μια ανάσχεση στην ομοχειρία μόλις εντοπιστεί εντολή τριών καταχωρητών πηγής. Έτσι υπάρχει ένας επιπλέον κύκλος για να διαβαστεί ο τρίτος καταχωρητής και να γίνουν οι επιπλέον πράξεις. Για αυτή την υλοποίηση χρησιμοποιήθηκε ένας ομόχειρος καταχωρητής στο στάδιο αποκωδικοποίησης που χρησιμοποιείται σαν σημαία που δείχνει σε ποιο κύκλο της εντολής είναι η εκτέλεση (πρώτο ή δεύτερο). Ανάλογα με τον κύκλο γίνεται η αποκωδικοποίηση στέλνοντας τα σωστά δεδομένα στην επόμενη βαθμίδα. Έτσι στη βαθμίδα αποκωδικοποίησης προστίθεται μία έξοδος ομόχειρου καταχωρητή(αυτή της σημαίας) για τη βαθμίδα εκτέλεσης εντολών. Στη βαθμίδα εκτέλεσης εντολών υπάρχει επίσης ένας καταχωρητής σημαία που έχει ενημερωθεί από την προηγούμενη βαθμίδα. Το κύκλωμα προώθησης δεδομένων τροποποιείται έτσι ώστε όταν εκτελείται ο δεύτερος κύκλος της εντολής γίνεται πράξη μεταξύ του αποτελέσματος της ALU του προηγούμενου κύκλου και του τρίτου καταχωρητή πηγής. Οι παραπάνω προσθήκες καθώς και τα κυκλώματα του επεξεργαστή που τροποποιήθηκαν φαίνονται στο Σχήμα 5.2 με κόκκινο χρώμα.



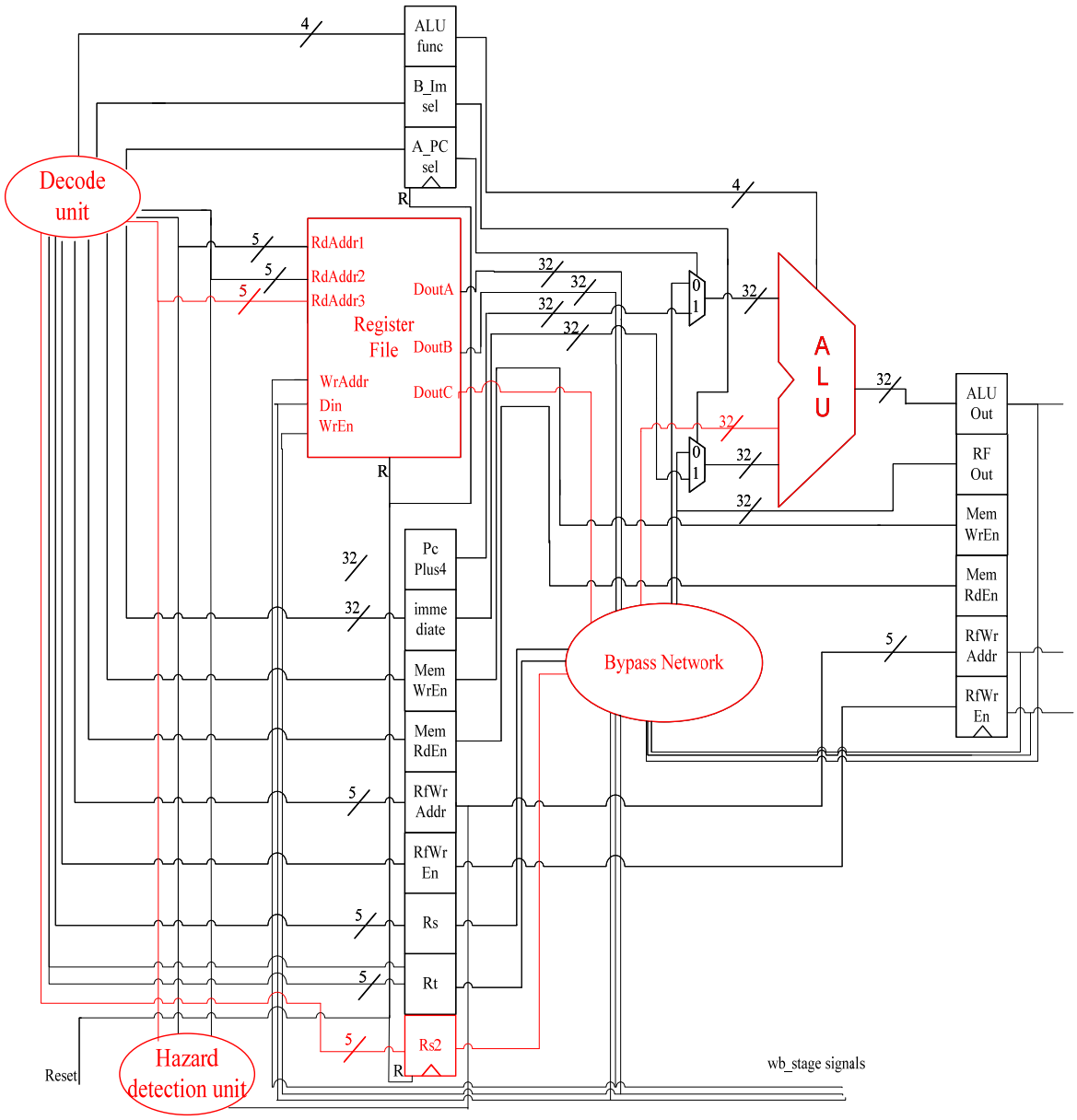
Σχήμα 5.2 1η υλοποίηση για την υποστήριξη εντολών τριών καταχωρητών πηγής

Η δεύτερη υλοποίηση απαιτεί σαφώς περισσότερες αλλαγές στη δομή του μονοπατιού δεδομένων καθώς τα δυο βασικότερα κυκλώματα που είναι το αρχείο καταχωρητών και η αριθμητική λογική μονάδα αντικαθίστανται από αναβαθμισμένες εκδόσεις τους που έχουν μια θύρα ανάγνωσης παραπάνω. Για την επιπλέον θύρα στο αρχείο καταχωρητών θα χρησιμοποιηθεί ένα ακόμα αντίγραφο μνήμης 32 bit, 32 θέσεων και η αντίστοιχη λογική για την προώθηση δεδομένων που εξαλείφει τους δομικούς κινδύνους που μπορεί να προκύψουν. Η συγκεκριμένη τροποποίηση παρουσιάζεται στο Σχήμα 5.3 με κόκκινο

χρώμα. Εκτός από τις παραπάνω αλλαγές, είναι αναπόφευκτες και οι τροποποιήσεις στη μονάδα αποκωδικοποίησης, στη μονάδά ανίχνευσης κινδύνων και στο δίκτυο προώθησης δεδομένων έτσι ώστε να είναι δυνατή η επιθυμητή λειτουργία και για τα επιπλέον δεδομένα που κυκλοφορούν πλέον στο μονοπάτι δεδομένων. Συνολικά οι προσθήκες και οι τροποποιήσεις παρουσιάζονται στο Σχήμα 5.4 με κόκκινο χρώμα.



Σχήμα 5.3 Υλοποίηση αρχείου καταχωρητών με τρεις θύρες ανάγνωσης



Σχήμα 5.4 2η υλοποίηση για την υποστήριξη εντολών τριών καταχωρητών πηγής

Το θέμα επιλογής της υλοποίησης με την οποία θα συνεχιζόταν η ανάπτυξη του μονοπατιού δεδομένων έμοιαζε με διελκυστίνδα μεταξύ χρονικής και χωρικής πολυπλοκότητας. Στη πρώτη υλοποίηση η εκτέλεση των εντολών αυξάνεται κατά ένα κύκλο έτσι επηρεάζεται αρνητικά το CPI του μονοπατιού δεδομένων μας. Στη δεύτερη το CPI παραμένει σταθερό αλλά αυξάνεται κατά πολύ η πολυπλοκότητα των κυκλωμάτων μας.

Χρησιμοποιώντας το Xilinx XST (Xilinx Synthesis Tool) και το ISE 7.1 μετρήθηκε για τις δύο υλοποιήσεις η μέγιστη συχνότητα και το ποσοστό των πόρων που χρησιμοποιούν σε κάποιες FPGA. Στα αποτελέσματα που φαίνονται στον Πίνακα 5.4 παρατηρήθηκε ότι η πρώτη υλοποίηση δίνει πάντα καλύτερη συχνότητα και χρησιμοποιεί λιγότερους πόρους.

FPGA	1η Υλοποίηση		2η Υλοποίηση	
	Συχνότητα (MHz)	Ποσοστό πόρων σε χρήση	Συχνότητα (MHz)	Ποσοστό πόρων σε χρήση
xc3s1000	47,275	16%	37,172	18%
xc3s1500	57,774	9%	42,453	10%
xc3s5000	54,774	3%	42,453	4%
xc2v8000	70,249	2%	51,962	2%

Πίνακας 5.4 Στατιστικά απόδοσης των 2 υλοποιήσεων

Ωστόσο η καλύτερη συχνότητα και το μικρότερο ποσοστό χρησιμοποίησης πόρων της $1^{ης}$ έναντι της $2^{ης}$ υλοποίησης δεν είναι ικανοί παράγοντες για να υπερκαλύψουν την αύξηση του CPI που επιφέρει στην ομοχειρία η $1^{η}$ υλοποίηση. Αυτό συμβαίνει γιατί οι διαφορές στα στατιστικά απόδοσης είναι σχετικά μικρές και επιπλέον όσο μεγαλώνει η FPGA οι διαφορές στα ποσοστά χρησιμοποίησης πόρων μηδενίζονται. Συνοψίζοντας προκύπτει το συμπέρασμα ότι η ανάπτυξη του μονοπατιού δεδομένων πρέπει να συνεχιστεί πάνω στην $2^{η}$ υλοποίηση που δεν επηρεάζει αρνητικά το CPI .

5.2.2 Πολλαπλασιασμός 32 bit modulo 2^{32}

Για την κωδικοποίηση της εντολής του πολλαπλασιασμού, που παρουσιάζεται στον Πίνακα 5.5, χρησιμοποιήθηκε η R-Type μορφή. Ο πολλαπλασιαστής που χρησιμοποιήθηκε είναι ένα σύγχρονο κύκλωμα που χρειάζεται 2 κύκλους ρολογιού για να βγάλει αποτέλεσμα. Το παραπάνω γεγονός περιέπλεξε κάπως τα πράγματα λόγω του ότι εκτός από τη βαθμίδα εκτέλεσης εντολών στην οποία εισάγεται το συγκεκριμένο εξάρτημα έπρεπε να τροποποιηθούν και όλες οι άλλες βαθμίδες του πυρήνα του DLX.

Εντολή	Σύνταξη	Περιγραφή
mmult	mmult rd,rs,rt	Πολλαπλασιάζει rs με rt και αποθηκεύει στον rd

Πίνακας 5.5 R-Type εντολή για τον πολλαπλασιασμό

Στη βαθμίδα αποκωδικοποίησης εντολής εκτός από την προφανή προσθήκη λογικής για την παραγωγή των κατάλληλων σημάτων για την σωστή εκτέλεση της εντολής του πολλαπλασιασμού έπρεπε να ρυθμιστεί και το κύκλωμα παραγωγής σήματος ανάσχεσης της ομοχειρίας. Στη περίπτωση που η επόμενη εντολή έχει καταχωρητή πηγής τον

καταχωρητή προορισμού του πολλαπλασιασμού εισάγεται μία ανάσχεση στην ομοχειρία για ένα κύκλο έτσι ώστε τα σωστά δεδομένα να είναι διαθέσιμα και έτοιμα να προωθηθούν. Για τη σωστή λειτουργία του κυκλώματος παραγωγής σήματος ανάσχεσης έπρεπε να προστεθεί στη βαθμίδα και ένας καταχωρητής που να κρατάει τη διεύθυνση του καταχωρητή προορισμού. Τέλος στη βαθμίδα έχει προστεθεί και ένας ομόχειρος καταχωρητής που κρατάει πληροφορία για το είδος της πράξης που θα γίνει στη βαθμίδα εκτέλεσης εντολής (πολλαπλασιασμός ή αριθμητική / λογική).

Στη βαθμίδα εκτέλεσης εντολών τοποθετείται το κύκλωμα του πολλαπλασιαστή. Η έξοδος του συγκεκριμένου εξαρτήματος θεωρείται ομόχειρος καταχωρητής συνεπώς δεν χρειάζεται να προστεθεί επιπλέον καταχωρητής για την αποθήκευση του αποτελέσματος του πολλαπλασιασμού. Λόγω του ότι το ο πολλαπλασιαστής χρειάζεται 2 κύκλους για να βγάλει το αποτέλεσμα πρέπει να προστεθεί στη βαθμίδα ένας καταχωρητής σημαία που να δείχνει ότι έχει γίνει πολλαπλασιασμός και σε μετέπειτα στάδια της ομοχειρίας όταν το αποτέλεσμα του πολλαπλασιαστή έχει παραχθεί να το επιλέγει.

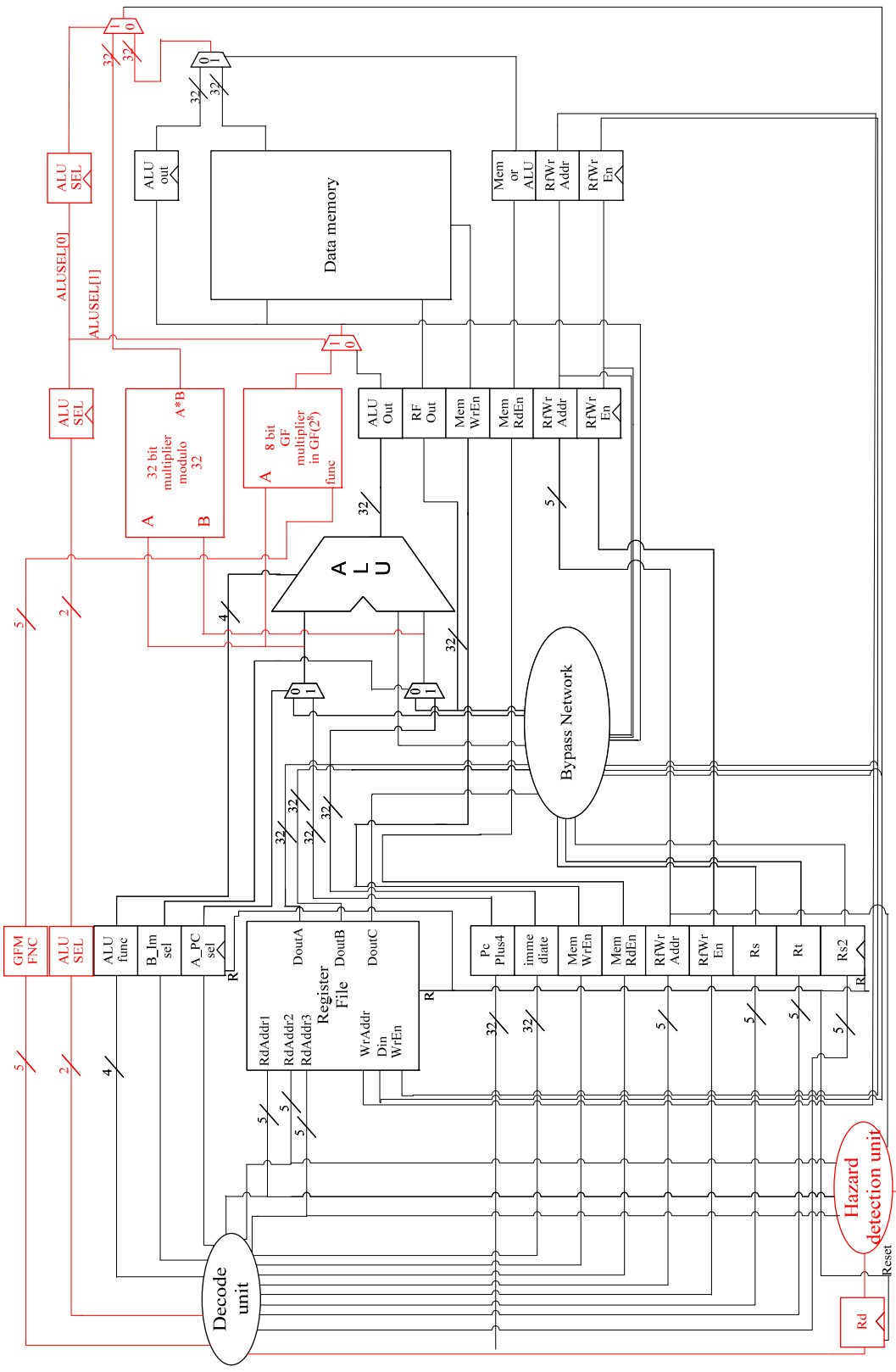
Στη βαθμίδα πρόσβασης μνήμης προστίθεται μόνο ο καταχωρητής σημαία για τον πολλαπλασιασμό καθώς το αποτέλεσμα της πράξης απλώς διαπερνάει τη βαθμίδα χωρίς να αποθηκεύεται σε κάποιο καταχωρητή. Το τελευταίο συμβαίνει λόγω του ότι η έξοδος του πολλαπλασιαστή είναι ομόχειρος καταχωρητής που χρειάζεται 2 κύκλους, επομένως το αποτέλεσμα θα είναι διαθέσιμο στην επόμενη βαθμίδα.

Στη βαθμίδα εγγραφής αποτελέσματος προστέθηκε ένας πολυπλέκτης για να στέλνει στο αρχείο καταχωρητών το αποτέλεσμα του πολλαπλασιασμού στη περίπτωση που έχει εκτελεστεί η συγκεκριμένη εντολή.

Στο Σχήμα 5.5 παρουσιάζεται η προσθήκη του πολλαπλασιαστή στο μονοπάτι δεδομένων. Είναι πολύ σημαντικό ότι το εξάρτημα συνδέθηκε με τέτοιο τρόπο ώστε να μην εισάγεται ανάσχεση στην ομοχειρία παρά μόνο όταν ανιχνευθεί κίνδυνος δεδομένων.

5.2.3 Πολλαπλασιασμός σε ένα πεπερασμένο πεδίο

Στην ενότητα αυτή θα προστεθεί στη σχεδίαση ένας πολλαπλασιαστής 8 bit σε ένα πεπερασμένο πεδίο το $GF(2^8)$. Για την επίτευξη αυτής της πράξης το μονοπάτι δεδομένων θα πρέπει να υποστηρίζει τρεις ειδικές εντολές. Οι εντολές αυτές κωδικοποιούνται με την R-Type μορφή και παρουσιάζονται στον Πίνακα 5.6.



Σχήμα 5.5 Προσθήκη στο μονοπάτι δεδομένων των 2 πολλαπλασιαστών και της απαραίτητης λογικής. Οι πολλαπλασιαστές είναι σύγχρονα κυκλώματα και οι έξοδοι τους θεωρούνται ομόχειροι καταχωρητές.

Εντολή	Σύνταξη	Περιγραφή
gfm	gfm rd,rs	Πολλαπλασιασμός στο πεδίο GF(2^8) μεταξύ του rs και του τελεστή x του GF, αποθήκευση στον rd
ldgfopx	ldgfopx rs	Φόρτωση του τελεστή x του GF με το περιεχόμενο του καταχωρητή rs
ldgfmr	ldgfmr rs	Φόρτωση του καταχωρητή που κρατάει το πολυώνυμο για την modulo πράξη με το περιεχόμενο του rs

Πίνακας 5.6 R-Type εντολές για τον πολλαπλασιασμό σε πεπερασμένο πεδίο

Η προσθήκη στην σχεδίαση του συγκεκριμένου πολλαπλασιαστή καθώς και των απαραίτητων σημάτων για την σωστή λειτουργία του παρουσιάζονται στο Σχήμα 5.5. Το αποτέλεσμα αυτού του πολλαπλασιαστή επιλέγεται στη βαθμίδα πρόσβασης μνήμης από ένα καταχωρητή σημαία που χρησιμοποιείται για να διακρίνει τις τρεις πιθανές πράξεις του σταδίου εκτέλεσης (ALU, πολλαπλασιασμός, πολλαπλασιασμός σε κάποιο πεδίο).

5.2.4 Πρόσβαση σε πίνακες αντικατάστασης (S-Boxes)

Σε αυτή την ενότητα θα μελετηθεί η διασύνδεση των πινάκων αντικατάστασης των AES-Rijndael, MARS, Twofish, Serpent αλγορίθμων στο υπάρχον μονοπάτι δεδομένων. Στον Πίνακα 5.7 παρουσιάζονται οι εντολές που υλοποιούν τις προσβάσεις σε αυτούς τους πίνακες.

Εντολή	Σύνταξη	Περιγραφή
aesX	aesX rd,rs	Πρόσβαση του rs στο Sbox κατά την κρυπτογράφηση ή αποκρυπτογράφηση (X=E,D) με το αποτέλεσμα να αποθηκεύεται στον rd
marsX	marsX rd,rs	Πρόσβαση του rs στο Sbox σε κάποια μορφή λειτουργίας (X=F,BE) με το αποτέλεσμα να αποθηκεύεται στον rd
serX	serX rd,rs	Πρόσβαση του rs στο Sbox κατά την κρυπτογράφηση ή αποκρυπτογράφηση (X=E,D) με το αποτέλεσμα να αποθηκεύεται στον rd
tX	tsld rs,rt / tsbox rd,rs	Φορτώνει S0 και S1 με rs και rt αντίστοιχα / Πρόσβαση του rs στο Sbox με το αποτέλεσμα να αποθηκεύεται στον rd

Πίνακας 5.7 R-Type εντολές για την πρόσβαση σε πίνακες αντικατάστασης

Στο 2^o κεφάλαιο παρουσιάστηκαν ο πίνακας αντικατάστασης καθώς και αυτός της αντίστροφης αντικατάστασης για τον AES-Rijndael αλγόριθμο. Οι δύο αυτοί πίνακες υλοποιούνται από μία μονάδα που έχει μια είσοδο 32 bit για τα δεδομένα, μια είσοδο ενός bit για τον τρόπο λειτουργίας (κρυπτογράφηση / αποκρυπτογράφηση) και μία έξοδο 32 bit για τα δεδομένα.

Ο πίνακας αντικατάστασης για τον Twofish είναι ένα σύγχρονο κύκλωμα που περιέχει δύο καταχωρητές και την απαραίτητη λογική για τις αναγκαίες λειτουργίες. Το συγκεκριμένο εξάρτημα έχει μία είσοδο 32 bit για τα δεδομένα, μια είσοδο ενός bit για τον τρόπο λειτουργίας και μία έξοδο 32 bit για τα δεδομένα. Να σημειωθεί ότι για την κρυπτογράφηση και την αποκρυπτογράφηση χρησιμοποιείται το ίδιο κύκλωμα και το bit για τον τρόπο λειτουργίας διακρίνει αυτή την περίπτωση από το φόρτωμα των δυο καταχωρητών που αναφέρθηκαν στην αρχή.

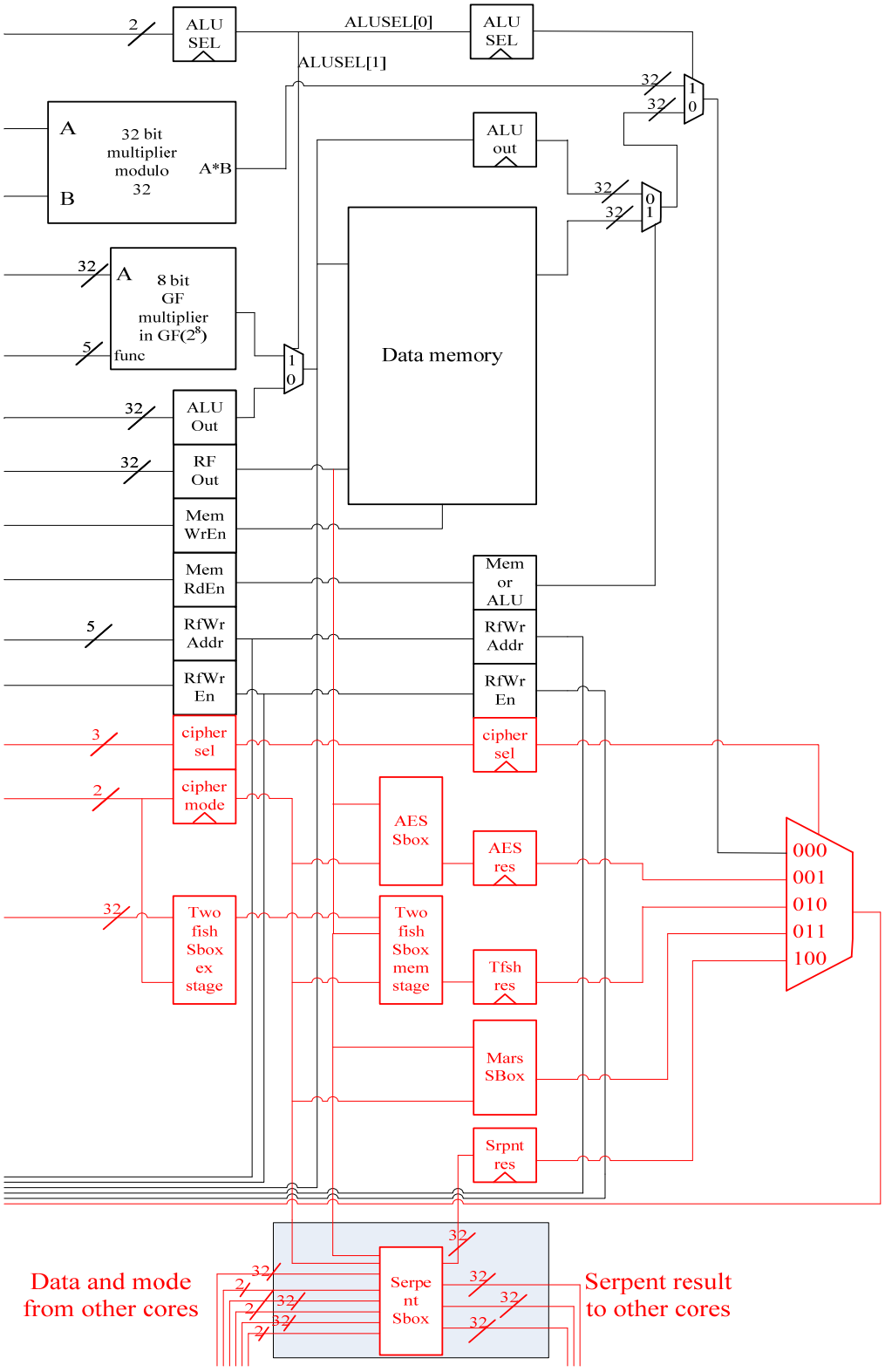
Το κύκλωμα που χρησιμοποιείται για τον πίνακα αντικατάστασης του Serpent αλγορίθμου χρειάζεται ως εισόδους τέσσερις 32 bit λέξεις. Για αυτό το λόγο τοποθετήθηκε στο υψηλότερο επίπεδο της σχεδίασης όπου εξάγουμε τις 4 λέξεις από τους επιμέρους πυρήνες. Πρόκειται για ένα σύγχρονο κύκλωμα που έχει επιπλέον μια είσοδο ενός bit για τον τρόπο λειτουργίας (κρυπτογράφηση / αποκρυπτογράφηση), μια είσοδο ενός bit για την αύξηση του μετρητή που περιέχει και τέλος 4 εξόδους των 32 bit για την επιστροφή των δεδομένων στη βαθμίδα εγγραφής αποτελέσματος.

Για την υλοποίηση του πίνακα αντικατάστασης του MARS αλγορίθμου χρησιμοποιήθηκαν 4 διαφορετικά κυκλώματα, ένα για κάθε πυρήνα της VLIW υλοποίησης. Αυτό συμβαίνει για την εκμετάλλευση της παράλληλης λειτουργικότητας που παρέχει η συγκεκριμένη υλοποίηση. Τα κυκλώματα είναι σύγχρονα με μια είσοδο μεταβλητού μεγέθους ανάλογα με τον πυρήνα για τον οποίο προορίζεται το κύκλωμα, μια είσοδο 2 bit για τον τρόπο λειτουργίας του πίνακα και μία έξοδο δεδομένων 32 bit. Η είσοδος μεταβλητού μεγέθους είναι μεταξύ 16 και 25 bit.

Η προσθήκη των παραπάνω πινάκων που φαίνεται στο Σχήμα 5.6 γίνεται κατά κύριο λόγο στο τμήμα πρόσβασης μνήμης εκτός από τον πίνακα αντικατάστασης του Twofish που για λόγους απόδοσης διαμοιράζεται μεταξύ του σταδίου εκτέλεσης και του σταδίου πρόσβασης μνήμης. Λόγω του ότι υπάρχει πιθανότητα να δημιουργηθεί κίνδυνος δεδομένων τροποποιείται κατάλληλα και η μονάδα που ελέγχει την ανάσχεση της ομοχειρίας.

5.2.5 Προσθήκη αρχείου καταχωρητών κλειδιών

Το αρχείο καταχωρητών των κλειδιών είναι ο χώρος που αποθηκεύονται τα κυκλικά κλειδιά που προκύπτουν από τον αλγόριθμο επέκτασης κλειδιού. Μετά από παρατηρήσεις του μεγέθους των δεδομένων των κυκλικών κλειδιών κρίθηκε ότι το συγκεκριμένο αρχείο καταχωρητών θα περιέχει 64 λέξεις των 32 bit.

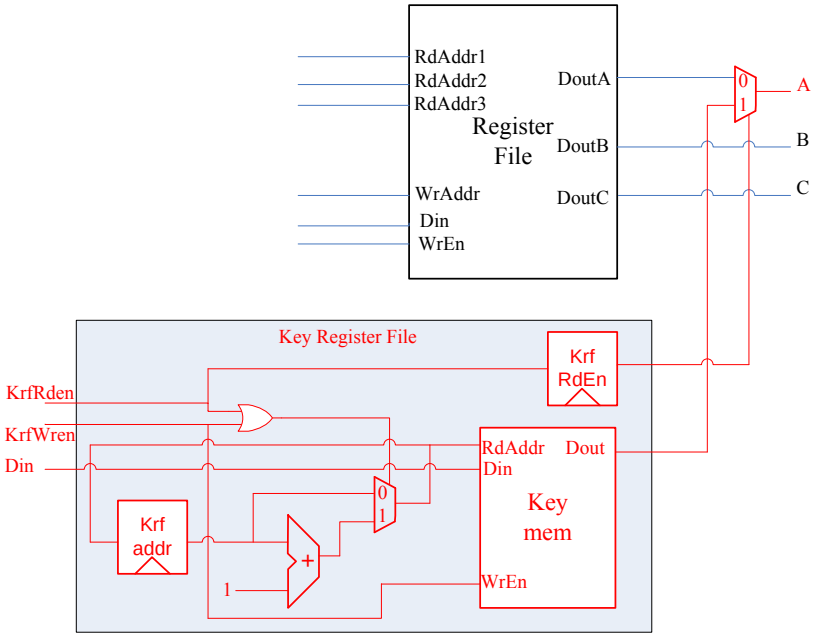


Σχήμα 5.6 Προσθήκη των πινάκων αντικατάστασης στο μονοπάτι δεδομένων. Όλες οι πράξεις αντικατάστασης γίνονται κατά τη διάρκεια του σταδίου πρόσβασης μνήμης εκτός αυτή του Twofish αλγορίθμου που ξεκινάει στο στάδιο εκτέλεσης εντολής. Το S-Box του Serpent αλγορίθμου υλοποιείται στο υψηλότερο επίπεδο της σχεδίασης λόγω του ότι χρειάζεται τα δεδομένα από όλους τους πυρήνες.

Το αρχείο καταχωρητών των κλειδιών αποτελείται από:

- Μια μνήμη 64 θέσεων των 32 bit, μιας θύρας ανάγνωσης και μίας θύρας εγγραφής που δημιουργήθηκε με το core generator του εργαλείου ISE 7.1.
- Ένα καταχωρητή για την αποθήκευση της διεύθυνσης στην οποία θα γίνει πρόσβαση καθώς και την απαραίτητη λογική για την σειριακή αύξηση αυτής της διεύθυνσης .
- Ένα ομόχειρο καταχωρητή για την αποθήκευση της σημαίας που επιλέγει στην επόμενη βαθμίδα από ποιο αρχείο καταχωρητών θα γίνει ανάγνωση.

Η μνήμη αυτή προστέθηκε στη βαθμίδα αποκωδικοποίησης εντολών έτσι ώστε τα κλειδιά να διαβάζονται σαν να ήταν απλοί καταχωρητές. Στο Σχήμα 5.7 παρουσιάζονται οι συνδέσεις των παραπάνω κυκλωμάτων που υλοποιούν το αρχείο καταχωρητών των κλειδιών καθώς και η διασύνδεση του τελευταίου με το κανονικό αρχείο καταχωρητών.



Σχήμα 5.7 Προσθήκη του αρχείο καταχωρητών των κλειδιών στο μονοπάτι δεδομένων

Οι εντολές που έχουν πρόσβαση στο αρχείο καταχωρητών των κλειδιών είναι αριθμητικών και λογικών πράξεων και κωδικοποιούνται με τη μορφή R-Type. Η κύρια λειτουργία των εντολών αυτών είναι να διαβάζουν ένα ή δυο καταχωρητές από το κλασικό αρχείο καταχωρητών και ένα καταχωρητή κλειδί και να κάνουν κάποια πράξη. Ωστόσο έχει υλοποιηθεί και μία εναλλακτική λειτουργία κατά την οποία γίνονται αριθμητικές ή λογικές πράξεις μεταξύ 2 ή 3 καταχωρητών από το κλασικό αρχείο

καταχωρητών με το αποτέλεσμα να αποθηκεύεται στο αρχείο καταχωρητών των κλειδιών. Τέλος λόγω του ότι η πρόσβαση στο αρχείο καταχωρητών των κλειδιών γίνεται σειριακά, με την βοήθεια ενός καταχωρητή μετρητή που κρατάει την διεύθυνση, προστίθεται μια εντολή για τον μηδενισμό αυτού του καταχωρητή. Οι παραπάνω εντολές παρουσιάζονται στους πίνακες 5.8, 5.9, 5.10.

Εντολή	Σύνταξη	Περιγραφή
addkey	addkey rd,kr,rt	Προσθέτει kr , rt αποθηκεύει στον rd
subkey	subkey rd,kr,rt	Αφαιρεί τον rt απο τον kr αποθηκεύει στον rd
orkey	orkey rd,kr,rt	Λογικό Ή μεταξύ kr , rt αποθηκεύει στον rd
andkey	andkey rd,kr,rt	Λογικό ΚΑΙ μεταξύ kr , rt αποθηκεύει στον rd
xorkey	xorkey rd,kr,rt	Αποκλειστικό Ή μεταξύ kr , rt αποθηκεύει στον rd
shrkey	shrkey rd,kr,rt	Ολίσθηση του kr κατά rt θέσεις δεξιά, αποθηκεύει στον rd
shlkey	shlkey rd,kr,rt	Ολίσθηση του kr κατά rt θέσεις αριστερά, αποθηκεύει στον rd
rorkey	rorkey rd,kr,rt	Κυκλική ολίσθηση του kr κατά rt θέσεις δεξιά, αποθηκεύει στον rd
rolkey	rolkey rd,kr,rt	Κυκλική ολίσθηση του kr κατά rt θέσεις αριστερά, αποθηκεύει στον rd
addaddkey	addaddkey rd,kr,rt,rs2	Προσθέτει kr, rt, rs2, αποθηκεύει στον rd
subaddkey	subaddkey rd,kr,rt,rs2	Αφαιρεί τον rt απο τον kr, προσθέτει τον rs2 στο αποτέλεσμα της αφαίρεσης, αποθηκεύει στον rd
addsubkey	addsubkey rd,kr,rt,rs2	Προσθέτει kr και rt, αφαιρεί τον rs2 από το αποτέλεσμα της πρόσθεσης, αποθηκεύει στον rd
subsubkey	subsubkey rd,kr,rt,rs2	Αφαιρεί τον rt απο τον kr, αφαιρεί τον rs2 από το αποτέλεσμα της αφαίρεσης, αποθηκεύει στον rd
xoraddkey	xoraddkey rd,kr,rt,rs2	Λογικό xor μεταξύ kr, rt, προσθέτει τον rs2 στο αποτέλεσμα της xor πράξης, αποθηκεύει στον rd
xorsubkey	xorsubkey rd,kr,rt,rs2	Λογικό xor μεταξύ kr, rt, αφαιρεί τον rs2 από το αποτέλεσμα της xor πράξης, αποθηκεύει στον rd
addxorkey	addxorkey rd,kr,rt,rs2	Προσθέτει kr και rt, λογικό xor μεταξύ του rs2 και του αποτελέσματος της πρόσθεσης, αποθηκεύει στον rd
subxorkey	subxor rd,kr,rt,rs2	Αφαιρεί τον rt απο τον kr, λογικό xor μεταξύ του rs2 και του αποτελέσματος της αφαίρεσης, αποθηκεύει στον rd
xorxorkey	xorxorkey rd,kr,rt,rs2	Λογικό xor μεταξύ kr,rt,rs2 αποθήκευση στον rd

Πίνακας 5.8 R-Type εντολές για την πράξη μεταξύ ενός ή δυο καταχωρητών από το κλασικό αρχείο καταχωρητών και ενός καταχωρητή από το αρχείο καταχωρητών των κλειδιών.

Εντολή	Σύνταξη	Περιγραφή
keyadd	keyadd kr,rs,rt	Προσθέτει rs , rt αποθηκεύει στον kr
keysub	keysub kr,rs,rt	Αφαιρεί τον rt απο τον rs αποθηκεύει στον kr
keyor	keyor kr,rs,rt	Λογικό Ή μεταξύ rs , rt αποθηκεύει στον kr
keyand	keyand kr,rs,rt	Λογικό ΚΑΙ μεταξύ rs , rt αποθηκεύει στον kr
keyxor	keyxor kr,rs,rt	Αποκλειστικό Ή μεταξύ rs , rt αποθηκεύει στον kr
keyshr	keyshr kr,rs,rt	Ολίσθηση του rs κατά rt θέσεις δεξιά, αποθηκεύει στον kr
keyshl	keyshl kr,rs,rt	Ολίσθηση του rs κατά rt θέσεις αριστερά, αποθηκεύει στον kr
keyror	keyror kr,rs,rt	Κυκλική ολίσθηση του rs κατά rt θέσεις δεξιά, αποθηκεύει στον kr
keyrol	keyrol kr,rs,rt	Κυκλική ολίσθηση του rs κατά rt θέσεις αριστερά, αποθηκεύει στον kr
keyaddadd	keyaddadd kr,rs,rt,rs2	Προσθέτει rs, rt, rs2, αποθηκεύει στον kr
keysubadd	keysubadd kr,rs,rt,rs2	Αφαιρεί τον rt απο τον rs, προσθέτει τον rs2 στο αποτέλεσμα της αφαίρεσης, αποθηκεύει στον kr
keyaddsub	keyaddsub kr,rs,rt,rs2	Προσθέτει rs και rt, αφαιρεί τον rs2 από το αποτέλεσμα της πρόσθεσης, αποθηκεύει στον kr
keysubsub	keysubsub kr,rs,rt,rs2	Αφαιρεί τον rt απο τον rs, αφαιρεί τον rs2 από το αποτέλεσμα της αφαίρεσης, αποθηκεύει στον kr
keyxoradd	keyxoradd kr,rs,rt,rs2	Λογικό xor μεταξύ rs, rt, προσθέτει τον rs2 στο αποτέλεσμα της xor πράξης, αποθηκεύει στον kr
keyxorsub	keyxorsub kr,rs,rt,rs2	Λογικό xor μεταξύ rs, rt, αφαιρεί τον rs2 από το αποτέλεσμα της xor πράξης, αποθηκεύει στον kr
keyaddxor	keyaddxor kr,rs,rt,rs2	Προσθέτει rs και rt, λογικό xor μεταξύ του rs2 και του αποτελέσματος της πρόσθεσης, αποθηκεύει στον kr
keysubxor	keysubxor kr,rs,rt,rs2	Αφαιρεί τον rt απο τον rs, λογικό xor μεταξύ του rs2 και του αποτελέσματος της αφαίρεσης, αποθηκεύει στον kr
keyxorxor	keyxorxor kr,rs,rt,rs2	Λογικό xor μεταξύ rs,rt,rs2 αποθήκευση στον kr

Πίνακας 5.9 R-Type εντολές για την πράξη μεταξύ δύο ή τριών καταχωρητών από το κλασικό αρχείο καταχωρητών με το αποτέλεσμα να αποθηκεύεται στο αρχείο καταχωρητών των κλειδιών.

Εντολή	Σύνταξη	Περιγραφή
krfaddrz	krfaddrz	Μηδενίζει την διεύθυνση πρόσβασης στο αρχείο καταχωρητών των κλειδιών

Πίνακας 5.10 Εντολή για τον μηδενισμό του καταχωρητή που κρατάει τη διεύθυνση πρόσβασης στο αρχείο καταχωρητών των κλειδιών.

5.2.6 Παράλληλη διασύνδεση τεσσάρών πυρήνων: VLIW CryptoDLX

Όπως έχει αναφερθεί και στην αρχή του κεφαλαίου για να είναι δυνατή η παράλληλη λειτουργία των τεσσάρων πυρήνων θα πρέπει να δημιουργηθεί μια νέα βαθμίδα ανάκλησης εντολών η οποία θα εκδίδει μία εντολή για κάθε πυρήνα. Ο πυρήνας πλέον δεν είναι αυτός του απλού DLX αλλά ο τροποποιημένος όπως έχει παρουσιαστεί στις ενότητες 5.2.1 έως 5.2.5.

Όπως φαίνεται στο Σχήμα 5.8 στη νέα βαθμίδα ανάκλησης εντολών υπάρχουν τα εξής κυκλώματα:

- Η μνήμη εντολών 128 bit, 256 θέσεων, μιας θύρας ανάγνωσης που δημιουργήθηκε με το core generator του εργαλείου ISE 7.1 της Xilinx.
- Ο μετρητής προγράμματος που κρατάει τη διεύθυνση της επόμενης προς ανάκληση εντολής.
- Ένας αθροιστής που αυξάνει την τιμή του μετρητή προγράμματος κατά 16 (επόμενη εντολή).
- Μια μονάδα επιλογής του μετρητή προγράμματος που θα αναλυθεί παρακάτω.
- Ένας πολυπλέκτης 2 σε 1 με εισόδους την διεύθυνση της επόμενης εντολής στην μνήμη εντολών (έξοδος του αθροιστή) ή τη διεύθυνση διακλάδωσης που προκύπτει από την κατάλληλη εντολή .Ο επιλογέας του πολυπλέκτη έρχεται από την μονάδα επιλογής του μετρητή προγράμματος .
- Ένα κύκλωμα που όταν έχει ανακληθεί εντολή διακλάδωσης υπολογίζει την διεύθυνση της.

Όπως προκύπτει από το κεφάλαιο που μελετήθηκαν οι συμμετρικοί κρυπτογραφικοί αλγόριθμοι για να λειτουργήσουν οι τελευταίοι θα χρειαστεί να προστεθούν στο μονοπάτι δεδομένων δυο εντολές οι οποίες και παρουσιάζονται στον Πίνακα 5.11.

Εντολή	**Σύνταξη**	**Περιγραφή**
ldlc	ldlc Imm	Φορτώνει τον καταχωρητή lc με την τιμή Imm
loop	loop label	Διακλάδωση στη διεύθυνση που δείχνει το label

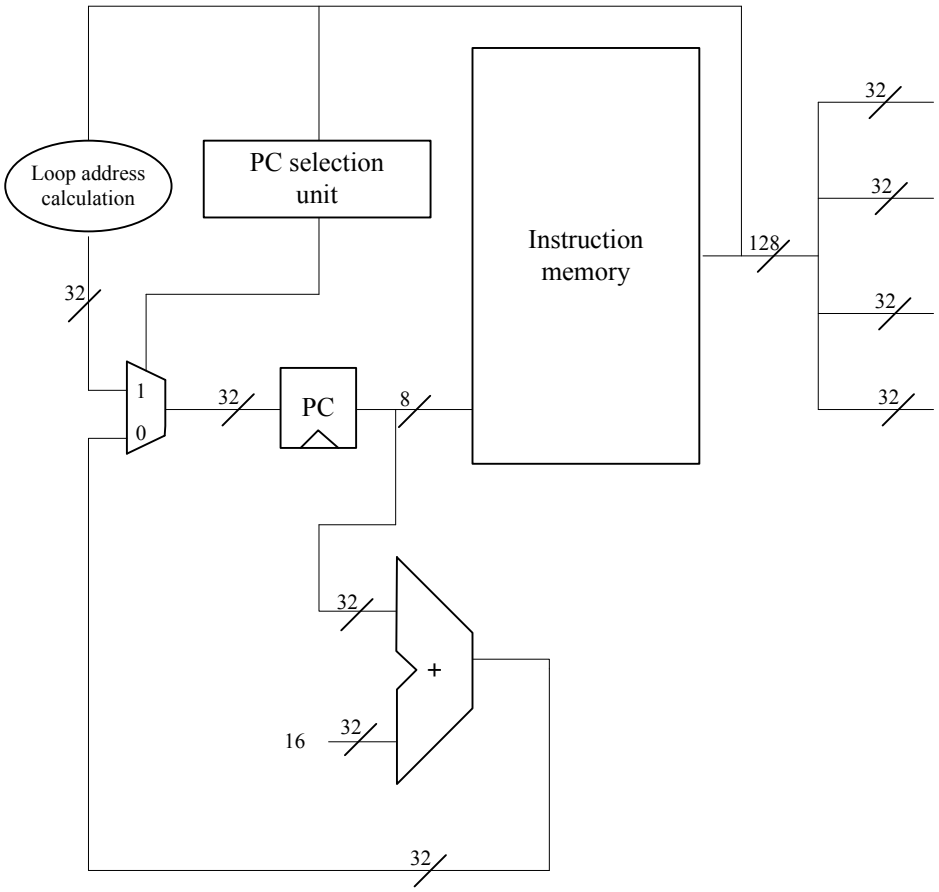
Πίνακας 5.11 Εντολές ελέγχου ροής προγράμματος

Οι εντολές που παρουσιάζονται στον Πίνακα 5.11 υλοποιούνται από τη βαθμίδα ανάκλησης εντολών που παρουσιάστηκε στο Σχήμα 5.8. Η διεύθυνση που δείχνει το label υπολογίζεται από το αρμόδιο κύκλωμα. Η απόφαση για την πραγματοποίηση ή όχι της διακλάδωσης παίρνεται από την μονάδα επιλογής του μετρητή προγράμματος η οποία περιέχει τον καταχωρητή lc και παρουσιάζεται στο Σχήμα 5.9. Σημειώνεται ότι μετά την loop εντολή εκτελείται πάντα και η επόμενη εντολή (ακόμα και αν η διακλάδωση είναι επιτυχημένη).

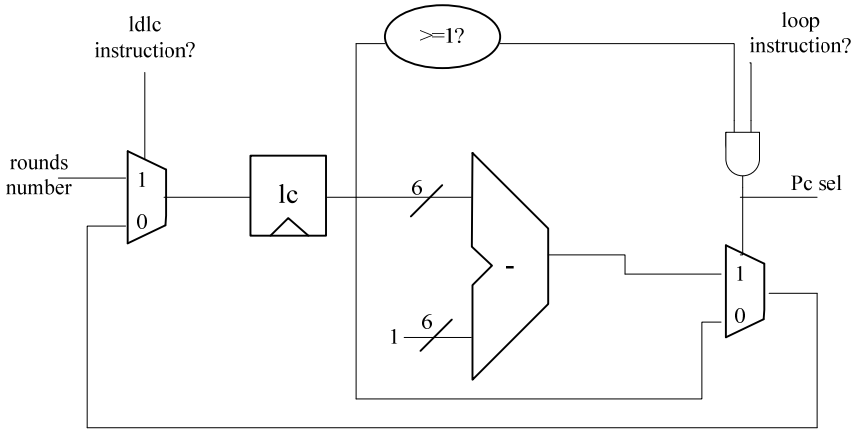
Η μονάδα αυτή δέχεται ως είσοδο την 128 bit εντολή. Η κύρια λειτουργία της μονάδας επιλογής του μετρητή προγράμματος είναι να ελέγχει αν έχει ανακληθεί loop εντολή και ο μετρητής lc δεν έχει μηδενιστεί και να παράγει το κατάλληλο σήμα. Επιπλέον αυτή η μονάδα εξυπηρετεί και τις ldlc εντολές ενημερώνοντας τον lc καταχωρητή.

Με την ολοκλήρωση τής νέας βαθμίδας ανάκλησης εντολών είναι πλέον εφικτή η παράλληλη διασύνδεση των τεσσάρων πυρήνων. Στο Σχήμα 5.10 παρουσιάζεται η αρχική έκδοση του VLIW επεξεργαστή CryptoDLX.

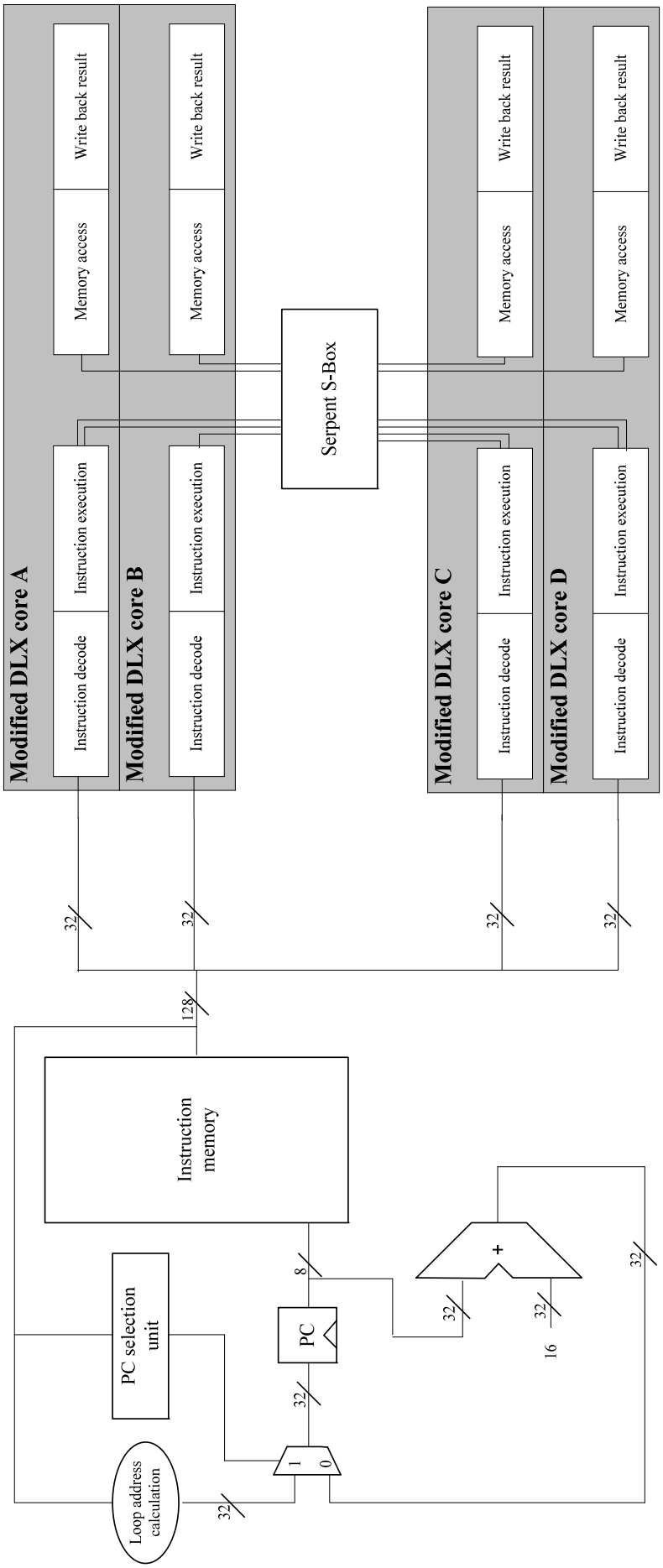
Στο Σχήμα 5.10 παρατηρείται η διασύνδεση των τεσσάρων τροποποιημένων πυρήνων του DLX. Οι πυρήνες αυτοί διαφέρουν μεταξύ τους μόνο στον πίνακα αντικατάστασης για τον MARS αλγόριθμο όπου ο καθένας από αυτούς χρησιμοποιεί ένα διαφορετικό κύκλωμα. Τέλος παρουσιάζεται και η διασύνδεση του πίνακα αντικατάστασης του Serpent αλγόριθμου με τους 4 πυρήνες. Ο συγκεκριμένος πίνακας λειτουργεί παράλληλα με το στάδιο πρόσβασης μνήμης και δεν πρέπει να δημιουργηθεί από το Σχήμα 5.10 η λανθασμένη εντύπωση ότι η πρόσβαση σε αυτόν γίνεται νωρίτερα.



Σχήμα 5.8 Βαθμίδα ανάκλησης εντολών του CryptoDLX επεξεργαστή



Σχήμα 5.9 Μονάδα επιλογής του μετρητή προγράμματος



Σχήμα 5.10 Αρχική έκδοσή του VLIW επεξεργαστή CryptoDLX

Η επόμενη προσθήκη που θα γίνει στο μονοπάτι δεδομένων προσθέτει μια πολλή σημαντική λειτουργικότητα στον επεξεργαστή χρησιμοποιώντας ελάχιστους επιπλέον πόρους. Για την ακρίβεια θα προστεθούν στην σχεδίαση μόλις 4 πολυπλέκτες 4 σε 1, αμέσως μετά τη βαθμίδα εκτέλεσης εντολών. Η συγκεκριμένη λειτουργικότητα είναι ή ανταλλαγή δεδομένων μεταξύ των πυρήνων. Είναι ιδιαίτερα χρήσιμο για τους συμμετρικούς κρυπτογραφικούς αλγόριθμους να μπορεί να γίνεται μια αριθμητική η λογική πράξη σε ένα πυρήνα και το αποτέλεσμα της να μπορεί να χρησιμοποιηθεί και στους υπόλοιπους πυρήνες. Στον Πίνακα 5.12 παρουσιάζεται η εντολή μετακίνησης.

Εντολή	**Σύνταξη**	**Περιγραφή**
movex	movex<instr> rd,rs,rt,rs2	Ο πυρήνας στέλνει δεδομένα στο στάδιο πρόσβασης μνήμης από τον x πυρήνα (x=a, b, c, d). Το <instr> θα γίνει μεταξύ rs, rt και rs2,αλλά το αποτέλεσμα δεν θα αποθηκευτεί στον rd. Μπορεί όμως να αποθηκευτεί το αποτέλεσμα από άλλους πυρήνες .Το <instr> μπορεί να πάρει τις ακόλουθες τιμές: add sub shr shl or rol and ror xor addadd addsub addxor subadd subsub subxor xoradd xorsub xorxor

Πίνακας 5.12 Εντολή μετακίνησης δεδομένων μεταξύ των πυρήνων

Για την λειτουργία της move εντολής τροποποιήθηκε η μορφή R-Type όπως φαίνεται στον Πίνακα 5.13. Το πεδίο mx καθορίζει από ποιον πυρήνα τα δεδομένα θα περάσουν στη βαθμίδα πρόσβασης μνήμης . Για να μην αλλάξει όλη η κωδικοποίηση της σχεδίασης όταν το mx παίρνει μηδενική τιμή προωθούνται τα δεδομένα από τον ίδιο πυρήνα και δεν υπάρχει καμία μετακίνηση. Όταν το mx παίρνει κάποια άλλη τιμή τότε ανάλογα με τον πυρήνα στον οποίο βρίσκεται η εντολή προωθούνται δεδομένα από κάποιο άλλο πυρήνα.

Στον Πίνακα 5.14 παρουσιάζονται οι τιμές που μπορεί να πάρει το mx σε κάθε πυρήνα για να γίνει μετακίνηση από κάποιο άλλο πυρήνα.

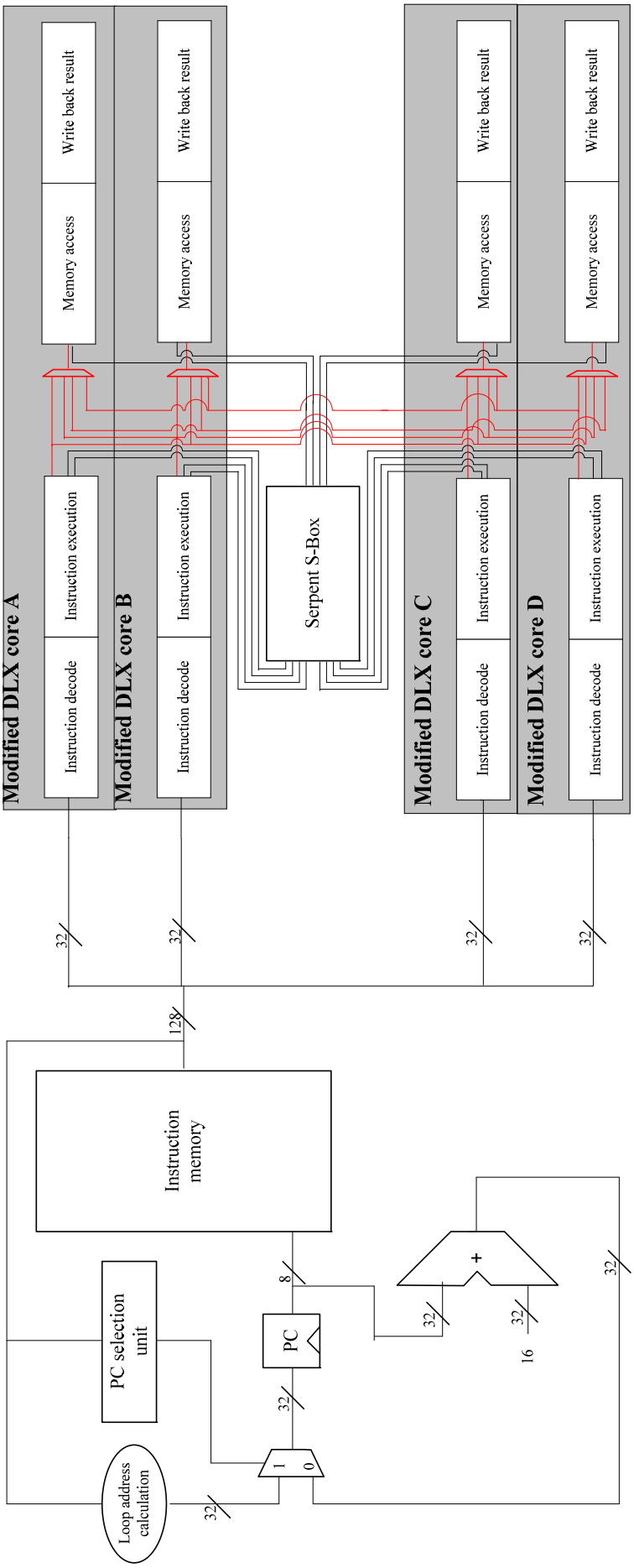
31-26	25-21	20-16	15-11	10-6	**5-4**	3-0
opcode	rs	rt	rd	rs2	**mx**	func

Πίνακας 5.13 R-Type μορφή εντολών για την υποστήριξη εντολών μετακίνησης

Πυρήνας Α		**Πυρήνας Β**		**Πυρήνας Γ**		**Πυρήνας Δ**	
Τιμή ma	**Πυρήνας προέλευσης**	**Τιμή mb**	**Πυρήνας προέλευσης**	**Τιμή mc**	**Πυρήνας προέλευσης**	**Τιμή md**	**Πυρήνας προέλευσης**
00	Α	**00**	Β	**00**	Γ	**00**	Δ
01	Β	**01**	Α	**01**	Α	**01**	Α
10	Γ	**10**	Γ	**10**	Β	**10**	Β
11	Δ	**11**	Δ	**11**	Δ	**11**	Γ

Πίνακας 5.14 Τιμές του πεδίου mx ανάλογα με το πυρήνα που εκτελεί την εντολή

Το πεδίο mx είναι στην ουσία ο επιλογέας του κάθε πολυπλέκτη που προστίθεται σε κάθε πυρήνα για να επιλέξει από ποιο πυρήνα θα προωθηθούν τα δεδομένα. Στο Σχήμα 5.11 που ακολουθεί παρουσιάζεται η προσθήκη των πολυπλεκτών στο μονοπάτι δεδομένων. Για λόγους ευκρίνειας τα σήματα mx που ενώνονται στους πολυπλέκτες δεν φαίνονται στο σχήμα.



Σχήμα 5.11 Προσθήκη στο μονοπάτι δεδομένων του επεξεργαστή CryptoDLX τεσσάρων πολυπλεκτών για την υποστήριξη εντολών μετακίνησης δεδομένων μεταξύ των πυρήνων

Η τελευταία εντολή που θα υλοποιηθεί είναι αυτή της αναστροφής ενός πίνακα 4 επί 4 όπου κάθε στοιχείο του είναι ένα byte. Η εντολή αυτή παρουσιάζεται στον Πίνακα 5.15. Είναι προφανές ότι η εντολή αυτή χρειάζεται δεδομένα ταυτόχρονα και από τους τέσσερις πυρήνες επομένως η λογική που την υλοποιεί θα τοποθετηθεί στο υψηλότερο επίπεδο της σχεδίασης. Για να μην εισαχθεί ανάσχεση στην ομοχειρία το κύκλωμα για την υλοποίηση της εντολής λειτουργεί παράλληλα με τη βαθμίδα εκτέλεσης εντολών.

Εντολή	**Σύνταξη**	**Περιγραφή**
r2c/c2r	r2c/c2r rd rs	Μετατροπή των στηλών σε γραμμές και τον ανάποδο σε μια 128 bit ποσότητα δεδομένων που θεωρούνται 4 επί 4 πίνακας

Πίνακας 5.15 R-Type μορφή εντολή για την αναστροφή ενός πίνακα 4x4 byte

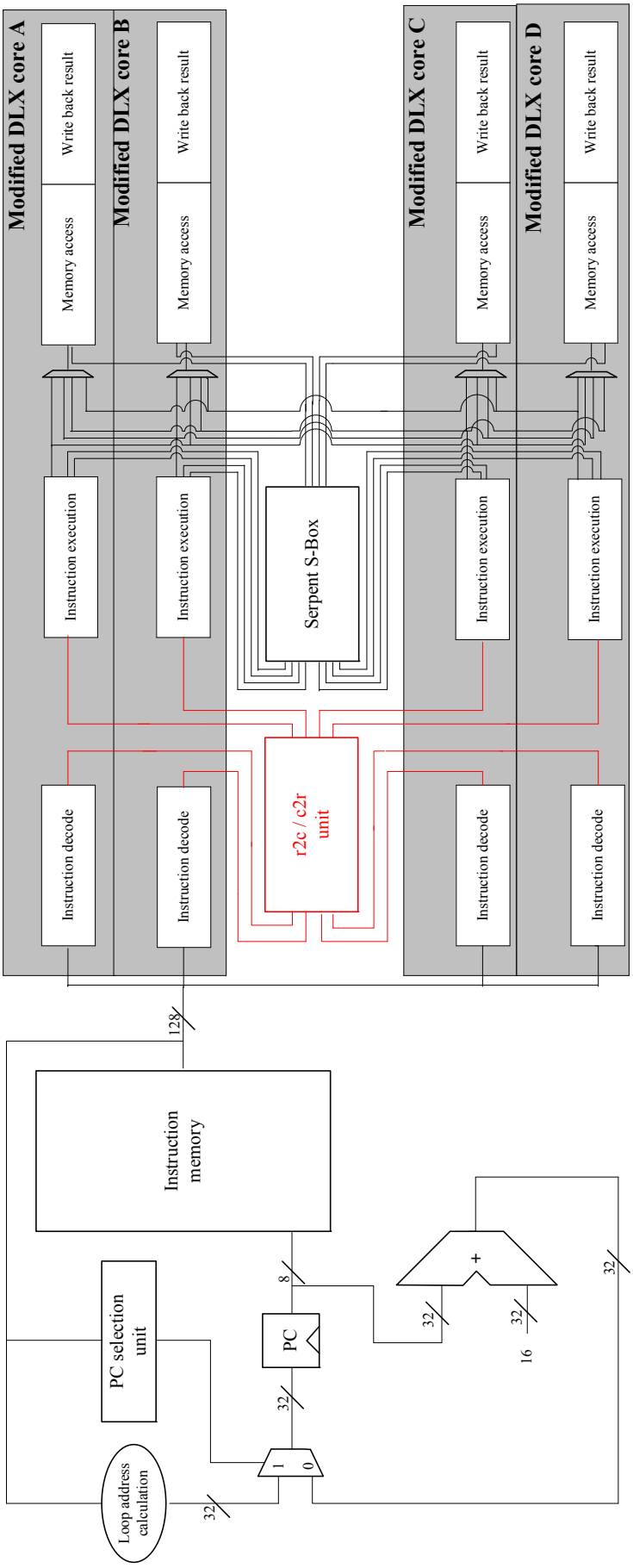
Η κάθε 32 bit λέξη που διαβάζεται από κάθε πυρήνα θεωρείται μια σειρά του 4x4 πίνακα, έτσι μετά τη βαθμίδα αποκωδικοποίησης έχοντας διαβάσει τέσσερις φορές τον rs καταχωρητή προκύπτει ο Πίνακας 5.16. Το $\Pi_{i,j}$ αντιστοιχεί στο j byte του i πυρήνα. Το κύκλωμα, του οποίου η προσθήκη στο μονοπάτι δεδομένων φαίνεται στο Σχήμα 5.12, επιστρέφει στους πυρήνες στο στάδιο εκτέλεσης εντολών τις 4 32 bit λέξεις που φαίνονται στις γραμμές του Πίνακα 5.17.

$\Pi_{1,1}$	$\Pi_{1,2}$	$\Pi_{1,3}$	$\Pi_{1,4}$
$\Pi_{2,1}$	$\Pi_{2,2}$	$\Pi_{2,3}$	$\Pi_{2,4}$
$\Pi_{3,1}$	$\Pi_{3,2}$	$\Pi_{3,3}$	$\Pi_{3,4}$
$\Pi_{4,1}$	$\Pi_{4,2}$	$\Pi_{4,3}$	$\Pi_{4,4}$

Πίνακας 5.16

$\Pi_{1,1}$	$\Pi_{2,1}$	$\Pi_{3,1}$	$\Pi_{4,1}$
$\Pi_{1,2}$	$\Pi_{2,2}$	$\Pi_{3,2}$	$\Pi_{4,2}$
$\Pi_{1,3}$	$\Pi_{2,3}$	$\Pi_{3,3}$	$\Pi_{4,3}$
$\Pi_{1,4}$	$\Pi_{2,4}$	$\Pi_{3,4}$	$\Pi_{4,4}$

Πίνακας 5.17



Σχήμα 5.11 Προσθήκη στο μονοπάτι δεδομένων του επεξεργαστή CryptoDLX τεσσάρων πολυπλεκτών για την υποστήριξη εντολών μετακίνησης δεδομένων μεταξύ των πυρήνων

6. Επιβεβαίωση Λειτουργίας και Αποτίμηση της Απόδοσης

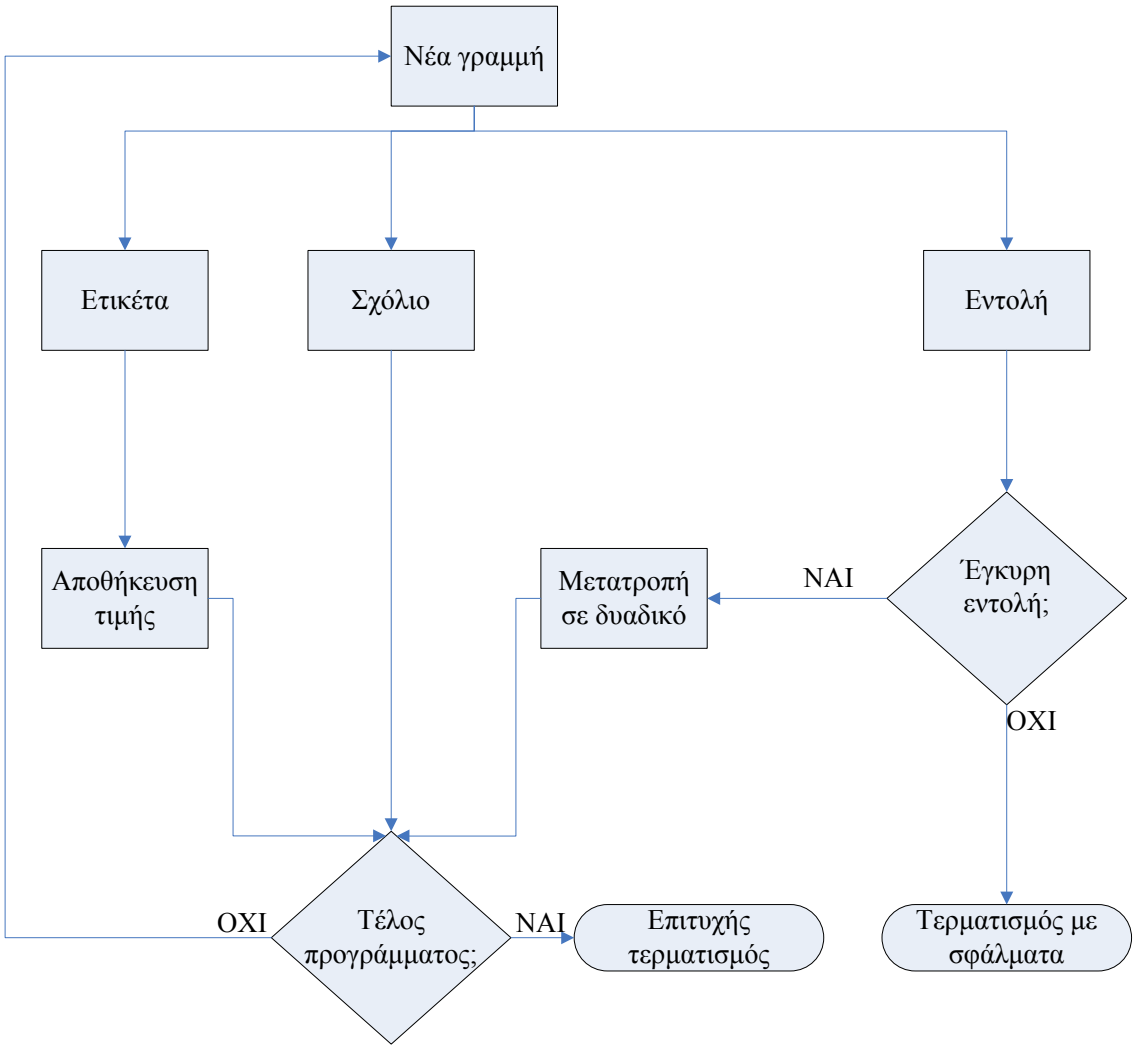
Σε αυτό το κεφάλαιο αρχικά παρουσιάζονται διάφορα δοκιμαστικά προγράμματα που γράφτηκαν για την επιβεβαίωση της σωστής λειτουργίας των δύο υλοποιήσεων που περιγράψαμε στα προηγούμενα κεφάλαια. Για την εξομοίωση της λειτουργίας χρησιμοποιήθηκε ο Mentor Graphics Modelsim SE 6.0a [I4]. Στη συνέχεια του κεφαλαίου υπολογίστηκαν για τις 2 υλοποιήσεις μεγέθη όπως η συχνότητα λειτουργίας και το ποσοστό των πόρων που καταλαμβάνουν σε διάφορες συσκευές FPGA. Το τελευταίο έγινε με το εργαλείο Xilinx ISE 7.1i [I5].

6.1 Επιβεβαίωση Λειτουργίας με Χρήση ενός Python Assembler

Με σκοπό να ελεγχθούν όσο το δυνατόν περισσότερα δοκιμαστικά προγράμματα σε ένα συγκεκριμένο χρονικό διάστημα υλοποιήθηκε ένας assembler. Για αυτή την υλοποίηση επιλέχθηκε ως γλώσσα η Python κυρίως λόγω της μεγάλης ευελιξίας που δείχνει στη διαχείριση συμβολοσειρών αλλά και λόγω της φορητοτητάς της σε διάφορες πλατφόρμες.

Το Σχήμα 6.1 παρουσιάζει σε αρκετά αφαιρετικό επίπεδο τη λειτουργία του assembler. Ο assembler διαβάζει γραμμή με γραμμή το πρόγραμμα από ένα txt αρχείο το μετατρέπει σε γλώσσα μηχανής και το αποθηκεύει σε ένα mem αρχείο που μπορεί να διαβαστεί από τον Modelsim για την αρχικοποίηση της μνήμης κατά την εξομοίωση. Πιο συγκεκριμένα ο assembler διαβάζοντας κάθε γραμμή του αρχείου διακρίνει αν είναι σχόλιο, ετικέτα, ή εντολή. Σημειώνεται ότι για να είναι μια γραμμή σχόλιο πρέπει να ξεκινάει με «--» και ότι τα σχόλια επιτρέπονται μόνο στην αρχή της γραμμής. Αν ή γραμμή είναι ετικέτα τότε η διεύθυνση της εντολής αποθηκεύεται για να μπορεί να ανακτηθεί αργότερα σε τυχόν διακλάδωση στην ετικέτα. Ο assembler θεωρεί εντολή οτιδήποτε δεν έχει αναγνωρισθεί ως σχόλιο ή ως ετικέτα. Έτσι αν υπάρξει διάβασμα μη έγκυρης εντολής η διάβασμα έγκυρης εντολής με συντακτικά λάθη τυπώνεται το αντίστοιχο μήνυμα λάθους και σταματάει η διαδικασία δημιουργίας του δυαδικού αρχείου. Αν η εντολή είναι έγκυρη και χωρίς συντακτικά λάθη, τα πεδία που διαβάστηκαν μετατρέπονται σε δυαδικούς αριθμούς και συνενώνονται κατάλληλα. Τέλος αν υπάρχει και άλλη γραμμή ακολουθείται η ίδια διαδικασία διαφορετικά ο assembler τερματίζεται επιτυχώς και το δυαδικό αρχείο είναι έτοιμο να φορτωθεί στη μνήμη.

Ο assembler αρχικά χρησιμοποιήθηκε για την DLX υλοποίηση στη συνέχεια όμως έγιναν οι απαραίτητες τροποποιήσεις για να εξυπηρετεί και την CryptoDLX υλοποίηση.



Σχήμα 6.1 Διάγραμμα ροής του assembler

Για την επιβεβαίωση της σωστής λειτουργίας τόσο του DLX όσο και του CryptoDLX γράφτηκαν αρκετά προγράμματα. Το μέγεθος αυτών των προγραμμάτων ξεκινάει από μερικές εντολές (για την επιβεβαίωση της σωστής λειτουργίας κάθε εντολής) και φτάνει μέχρι και ολοκληρωμένα προγράμματα δεκάδων εντολών που εκτελούν διάφορους αλγόριθμους όπως τον AES-Rijndael.

Στην Εικόνα 6.1 παρουσιάζεται ένα πρόγραμμα για τον απλό DLX που συγκεντρώνει όλες τις μορφές εντολών(R/I/J-Type), επιτυχημένες και αποτυχημένες διακλαδώσεις, ακολουθίες εντολών που απαιτούν προώθηση δεδομένων ή ακόμα και ανάσχεσή της ομοχειρίας. Στο συγκεκριμένο πρόγραμμα στηρίχτηκε μεγάλο μέρος της αποσφαλμάτωσης της υλοποίησης του απλού DLX καθώς με την βοήθεια του assembler και του Modelsim εντοπίστηκαν και επιλύθηκαν διάφορα λάθη που υπήρχαν στην υλοποίηση.

```
--DLX test
      addi r1,r0,2
      addi r2,r1,3
      addi r3,r2,4
      addi r5,r0,0
      addi r7,r0,0
      addi r16,r0,16
      addi r20,r0,20
      j    COMPARE1
      and  r2,r1,r0
      or   r3,r1,r2
      lw  r30,0(r16)
COMPARE1:
      addi r5,r5,1
      subi r1,r1,1
      bneq r1,COMPARE1
      j    COMPARE2
      andi r2,r1,0
      ori r3,r2,0
COMPARE2:
      addi r6,r6 ,1
      subi r2,r2,1
      bnez r2 ,COMPARE2
      j    COMPARE3
      jr   r0
COMPARE3:
      addi r7,r7,1
      subi r3,r3,1
      bnez r3,COMPARE3
      lw r11 , 0(r8)
      sw r11 , 4(r8)
      sw r5,   8(r0)
      sw r6,  12(r0)
      sw  r7,  16(r0)
COUNTER:
      sw r20, 20(r0)
      addi  r20,r20,1
      bnez r20,COUNTER
```

Εικόνα 6.1 Δοκιμαστικό πρόγραμμα για τον απλό DLX

Στην Εικόνα 6.2 παρουσιάζεται ο assembly κώδικας που υλοποιεί τον κύκλο επεξεργασίας του AES-Rijndael αλγορίθμου για τον CryptoDLX επεξεργαστή. Στα προγράμματα για αυτό τον επεξεργαστή κάθε εντολή ξεκινάει με το πεδίο "CX:" με το X να παίρνει τιμές A, B, C, D ανάλογα με τον πυρήνα για τον όποιο προορίζεται. Το πρόγραμμα που περιέχει τον συγκεκριμένο κώδικα μετατράπηκε σε γλώσσα μηχανής με χρήση του assembler και στη συνέχεια φορτώθηκε για εξομοίωση στον Modelsim. Λόγω του γεγονότος ότι το κομμάτι αυτό κώδικα είναι αυτό που απασχολεί τον περισσότερο χρόνο τον επεξεργαστή (9 κύκλοι επεξεργασίας για 128 bit κλειδί) όταν τρέχει ο AES-Rijndael αλγόριθμος είναι εξαιρετικά σημαντικό για την απόδοση της υλοποίησης.

Στην εξομοίωση με τον Modelsim παρατηρήθηκε ότι εισάγεται ανάσχεση στην ομοχειρία μετά τις `aesE` εντολές. Το τελευταίο είναι αναπόφευκτο από τη στιγμή που οι πίνακες αντικατάστασης τοποθετούνται στη βαθμίδα πρόσβασης μνήμης. Οι υπόλοιπες εντολές εκτελούνται κανονικά χωρίς καμία καθυστέρηση.

```
--AES-Rijndael encryption loop
encrypt:
--SubBytes                          --MixColumns
      CA: aesE r1,r1                      CA: gfm r1,r1
      CB: aesE r1,r1                      CB: gfm r1,r1
      CC: aesE r1,r1                      CC: gfm r1,r1
      CD: aesE r1,r1                      CD: gfm r1,r1

--Shift Rows                        --AddRoundKey
      CA: c2r r1,r1                       CA: xor r1,kr,r1
      CB: c2r r1,r1                       CB: xor r1,kr,r1
      CC: c2r r1,r1                       CC: xor r1,kr,r1
      CD: c2r r1,r1                       CD: xor r1,kr,r1

      CA: nop                             CA: loop encrypt
      CB: roli r1,r1,8                    CB: nop
      CC: roli r1,r1,16                   CC: nop
      CD: roli r1,r1,24                   CD: nop

      CA: r2c r1,r1                       CA: nop
      CB: r2c r1,r1                       CB: nop
      CC: r2c r1,r1                       CC: nop
      CD: r2c r1,r1                       CD: nop
```

Εικόνα 6.2 Κύκλος επεξεργασίας AES-Rijndael αλγόριθμου για τον CryptoDLX

Μετά την τετράδα εντολών που περιέχει την `loop` ακολουθεί μία τετράδα από `nops`. Αυτό συμβαίνει γιατί μετά από `loop` εντολές πάντα εκτελούνται και οι αμέσως επόμενες. Το τελευταίο αρχικά μπορεί να φαίνεται μειονέκτημα της σχεδίασης αλλά στην πράξη φάνηκε ιδιαίτερα χρήσιμο. Σημειώνεται ότι στον απλό DLX χρησιμοποιήθηκε το μοντέλο πρόβλεψης ανεπιτυχούς διακλάδωσης στο οποίο χανόντουσαν 2 κύκλοι ρολογιού όταν η διακλάδωση ήταν επιτυχής και έπρεπε να γίνει εισαγωγή «φυσαλίδας». Στον CryptoDLX κρίθηκε ότι αυτό το μοντέλο δεν είναι το αποδοτικότερο μια και στους συμμετρικούς κρυπτογραφικούς αλγόριθμους το ποσοστό των επιτυχημένων διακλαδώσεων είναι συντριπτικά μεγαλύτερο. Στον AES-Rijndael για παράδειγμα η συγκεκριμένη διακλάδωση 8 από τις 9 φορές θα είναι πετυχημένη.

Η εισαγωγή των `nops` είναι στην ουσία μια εισαγωγή «φυσαλίδας» στην ομοχειρία μέσο λογισμικού. Συνεπώς η διακλάδωση στον κώδικα της Εικόνας 6.2 θα κοστίσει το ίδιο με μια διακλάδωση σε επεξεργαστή που χρησιμοποιεί μοντέλο πρόβλεψης ανεπιτυχούς διακλάδωσης. Για την ακρίβεια θα κοστίσει και ένα κύκλο παραπάνω λόγω του ότι στην τελευταία επανάληψη το μοντέλο θα προβλέψει σωστά ενώ ο κώδικας θα εκτελέσει πάλι `nops.`

Η τεχνική που χρησιμοποιείται για να γίνει η εκμετάλλευση αυτής της ιδιαιτερότητας της υλοποίησης είναι γνωστή ως «καθυστερημένη διακλάδωση» [22]. Στη συγκεκριμένη περίπτωση η εφαρμογή της είναι εξαιρετικά απλή και αποδοτική. Το μόνο που χρειάζεται να γίνει είναι η προηγούμενη τετράδα εντολών της `loop` να μεταφερθεί μετά από αυτή και να πάρει τη θέση των `nops`. Συνεπώς το τελευταίο κομμάτι του κώδικα για τον κύκλο επεξεργασίας μπορεί να γραφτεί όπως φαίνεται στην Εικόνα 6.3.

```
--AddRoundKey optimized
      CA: loop encrypt
      CB: nop
      CC: nop
      CD: nop

      CA: xor r1,kr,r1
      CB: xor r1,kr,r1
      CC: xor r1,kr,r1
      CD: xor r1,kr,r1
```

Εικόνα 6.3 Εφαρμογή της τεχνικής της «καθυστερημένης διακλάδωσης»

Ο κώδικας της Εικόνας 6.2 κοστίζει 8 κύκλους ρολογιού (8 τετράδες εντολών) συν 1 κύκλο ρολογιού (κύκλος ανάσχεσης μετά την `aesE`) 9 κύκλους ρολογιού. Με την εφαρμογή της τεχνικής της «καθυστερημένης διακλάδωσης» το κόστος μειώνεται κατά ένα κύκλο. Επομένως η εφαρμογή αυτής της τεχνικής οδηγεί σε μία επιτάχυνση 1,125x του κύκλου επεξεργασίας.

Η συγκεκριμένη επιτάχυνση μόνο αμελητέα ποσότητα δεν μπορεί να χαρακτηριστεί λόγω του ότι μπορεί να επιτευχθεί απλά με μια διαφορετική δρομολόγηση του κώδικα assembly χωρίς να χρειάζεται να γίνει κάποια τροποποίηση στη σχεδίαση του υλικού. Εν κατακλείδι ο κώδικας assembly είναι αρκετά σημαντικός για την επίτευξη της υψηλότερης δυνατής απόδοσης για ένα επεξεργαστή και ο Python assembler προσέφερε αρκετή ευελιξία προς αυτή την κατεύθυνση.

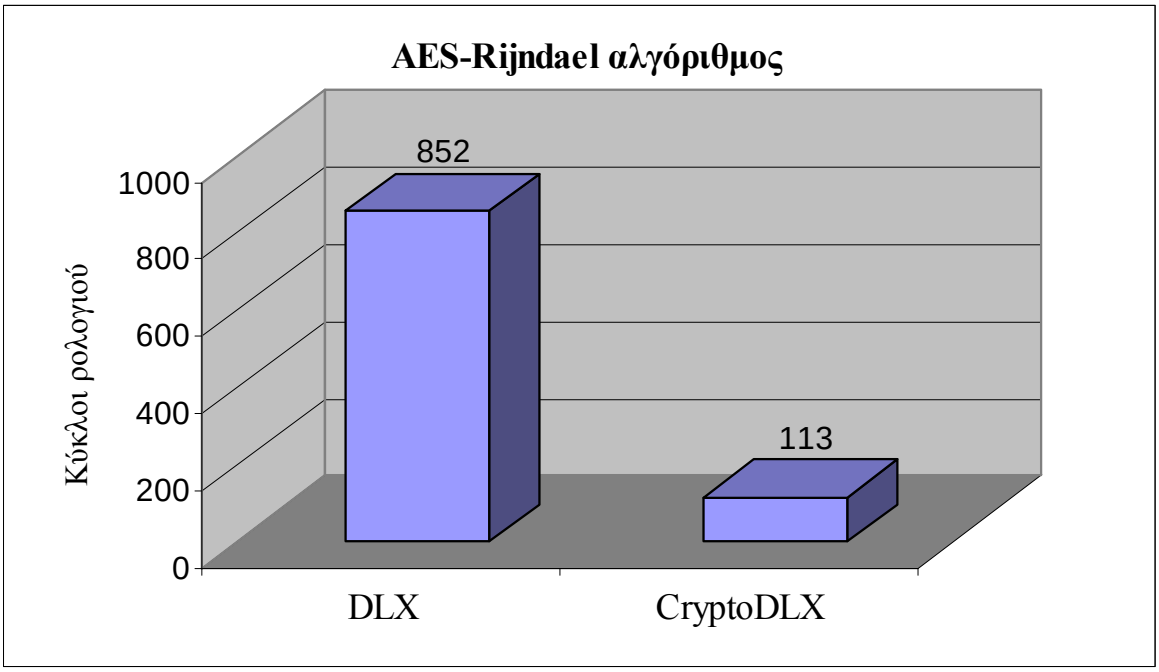
6.2 Αποτίμηση της Απόδοσης σε Συσκευές FPGA της Xilinx

Σε αυτή την ενότητα γίνεται μια αποτίμηση της απόδοσης της DLX και της CryptoDLX υλοποίησης για την επεξεργασία του AES-Rijndael κρυπτογραφικού αλγόριθμου. Για τον υπολογισμό της απόδοσης θα χρησιμοποιηθεί ως δείκτης ο ρυθμός ολοκλήρωσης του συγκεκριμένου αλγορίθμου. Ο ρυθμός ολοκλήρωσης δίνεται από τον ακόλουθο τύπο:

$$\text{Ρυθμός Ολοκλήρωσης} = \frac{128 \cdot F(Mhz)}{cc} Mbits/\sec$$

Το cc αντιστοιχεί στους κύκλους ρολογιού που χρειάζεται η κάθε υλοποίηση για να εκτελέσει τον αλγόριθμο και το F στην συχνότητα λειτουργίας της. Στο Διάγραμμα 6.1 παρουσιάζεται ο συνολικός αριθμός των κύκλων ρολογιού που χρειάζεται για την επεξεργασία του αλγόριθμου η κάθε υλοποίηση.

Πρέπει να σημειωθεί ότι οι συγκεκριμένοι αριθμοί αφορούν μόνο τη διαδικασία κρυπτογράφησης ή αποκρυπτογράφησης καθώς δεν περικλείουν τη διαδικασία επέκτασης κλειδιού στις δύο υλοποιήσεις και την διαδικασία δημιουργίας των πινάκων αντικατάστασης για την DLX υλοποίηση. Το παραπάνω στηρίχτηκε στο γεγονός ότι οι διαδικασίες αυτές εκτελούνται μόνο μια φορά κατά την κρυπτογράφηση ενός μηνύματος. Αντίθετα η διαδικασία που μετρήθηκε μπορεί να εκτελεστεί n φορές με το n να μεγαλώνει ανάλογα με το μέγεθος του μηνύματος.



Διάγραμμα 6.1 Απόδοση των δύο υλοποιήσεων σε κύκλους ρολογιού για τον AES-Rijndael αλγόριθμο

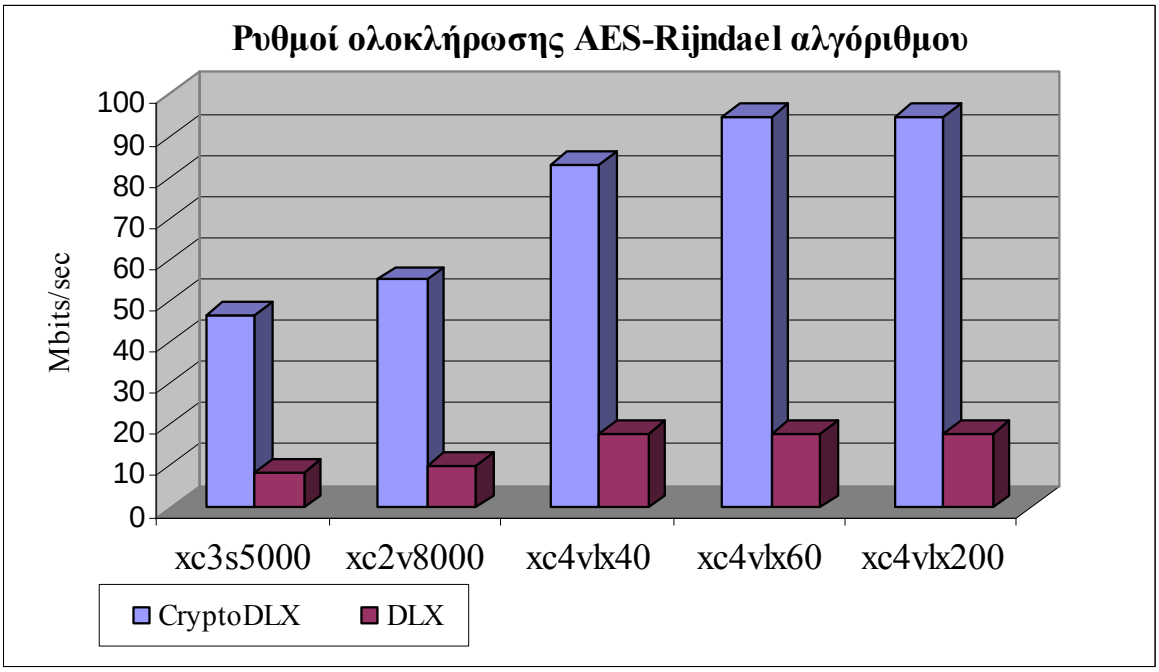
Από το Διάγραμμα 6.1 προκύπτει ότι ο CryptoDLX προσφέρει μια επιτάχυνση 7,54x σε σχέση με τον DLX. Το αποτέλεσμα αυτό δεν είναι απόλυτα κατατοπιστικό λόγω του ότι η DLX σχεδίαση πετυχαίνει υψηλότερη συχνότητα όταν υλοποιηθεί σε μια συσκευή FPGA.

Με χρήση του Xilinx XST του ISE 7.1 μετρήθηκε για τις δύο υλοποιήσεις η μέγιστη συχνότητα και το ποσοστό των πόρων που χρησιμοποιούν σε κάποιες FPGA. Σημειώνεται ότι για την ταχύτητα ολοκλήρωσης (speed grade) χρησιμοποιήθηκε η καλύτερη δυνατή που δίνει το εργαλείο για κάθε FPGA.Τα αποτελέσματα παρουσιάζονται στον Πίνακα 6.1.

FPGA	CryptoDLX		DLX	
	Συχνότητα (MHz)	Ποσοστό πόρων σε χρήση	Συχνότητα (MHz)	Ποσοστό πόρων σε χρήση
xc3s5000	41,310	53%	56,532	3%
xc2v8000	48,770	36%	69,101	2%
xc4vlx40	73,453	88%	119,814	6%
xc4vlx60	83,724	63%		4%
xc4vlx200		18%		1%

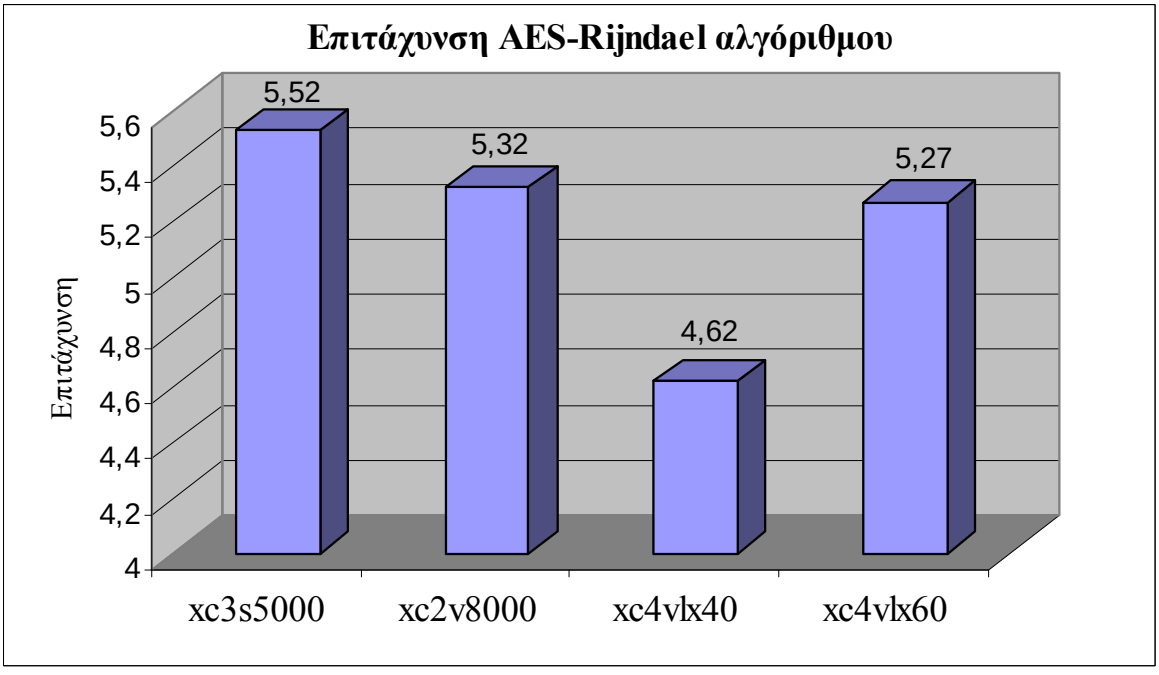
Πίνακας 6.1 Στατιστικά απόδοσης των δύο υλοποιήσεων

Με βάση τα αποτελέσματα του Πίνακα 6.1 προκύπτει το Διάγραμμα 6.2 με τους ρυθμούς ολοκλήρωσης των δύο υλοποιήσεων για τις παραπάνω FPGA. Από το Διάγραμμα 6.2 φαίνεται ότι η xc4vlx200 και η xc4vlx60 δίνουν τον καλύτερο ρυθμό ολοκλήρωσης που φτάνει τα 95Mbits/sec.



Διάγραμμα 6.2 Απόδοση με βάση τον ρυθμό ολοκλήρωσης για τον AES-Rijndael αλγόριθμο

Από το Διάγραμμα 6.2 προκύπτει η επιτάχυνση που δίνει ο CryptoDLX για τον AES-Rijndael σε κάθε FPGA. Το παραπάνω στατιστικό στοιχείο παρουσιάζεται στο Διάγραμμα 6.3 όπου φαίνεται ότι η επιτάχυνση κυμαίνεται από 4,62x μέχρι 5,52x.



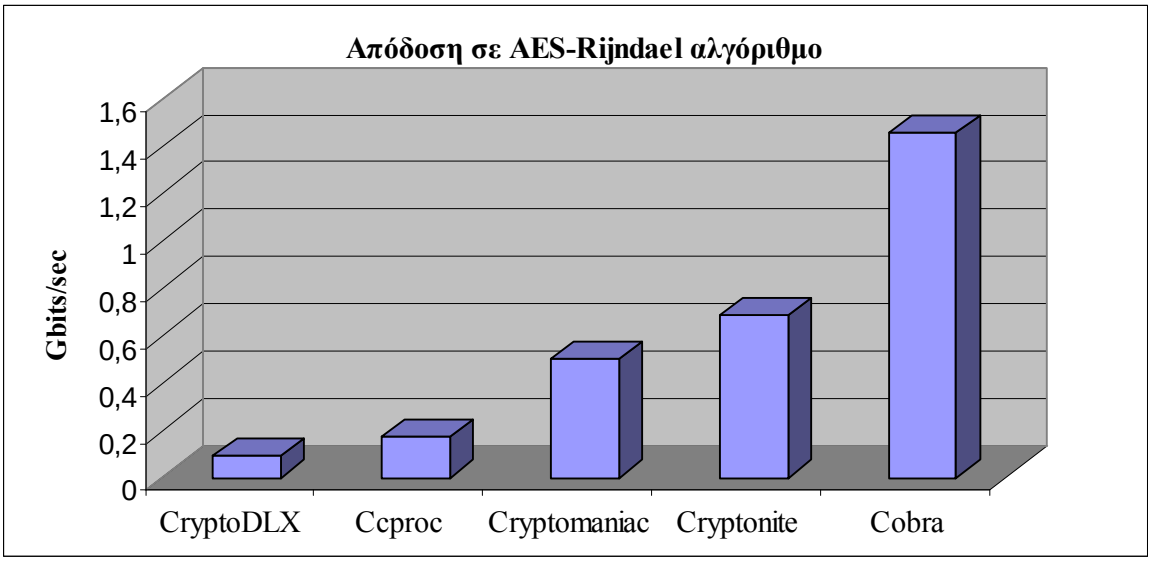
Διάγραμμα 6.3 Επιτάχυνση που προσφέρει ο CryptoDLX σε σχέση με τον απλό DLX για τον AES-Rijndael αλγόριθμο

6.3 Σύγκριση Απόδοσης με Παλιότερες Υλοποιήσεις

Μετά την παρουσίαση των στατιστικών απόδοσης του CryptoDLX σε αυτή την ενότητα θα γίνει μια σύγκριση των επιδόσεων του με αυτές παλιότερων υλοποιήσεων που παρουσιάστηκαν στο 3^{o} κεφάλαιο.

Οι Abdurohman και Sutikno στην [15] τροποποίησαν τον DLX και πέτυχαν με δείκτη τους κύκλους ρολογιού επιτάχυνση για τον AES-Rijndael 1.82x, πολύ μικρότερη από την 7.54x που πετυχαίνει ο CryptoDLX. Αντίθετα οι Fiskiran και Lee στην [17] πέτυχαν ελαφρώς καλύτερη επιτάχυνση που φτάνει τα 7.7x.

Όπως ήταν αναμενόμενο οι επιδόσεις του CryptoDLX δεν μπορούν να πλησιάσουν αυτές των υλοποιήσεων όπου το υλικό είναι εξειδικευμένο για τον AES-Rijndael. Τα αποτελέσματα της σύγκρισης της απόδοσης του CryptoDLX με τους συνεπεξεργαστές κρυπτογράφησης παρουσιάζονται στο Διάγραμμα 6.4.



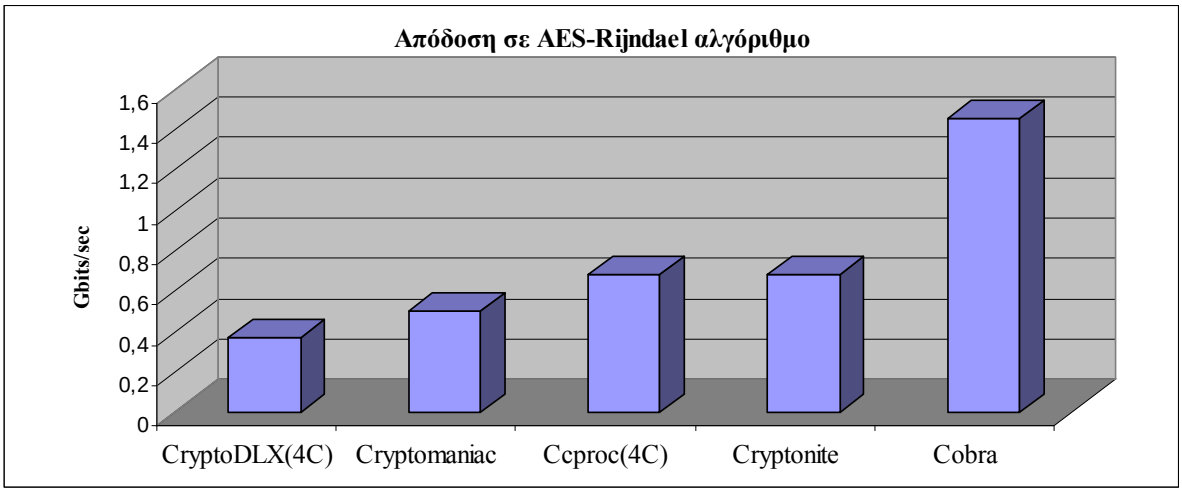
Διάγραμμα 6.4 Σύγκριση της απόδοσης του CryptoDLX για τον AES-Rijndael αλγόριθμο με τις αποδόσεις διάφορων συνεπεξεργαστών κρυπτογράφησης

Πρέπει να σημειωθεί σε αυτό το σημείο ότι ο CryptoDLX και Ccproc υλοποιούνται σε συσκευές FPGA σε αντίθεση με τους άλλους τρεις συνεπεξεργαστές που έχουν υλοποιηθεί σαν ολοκληρωμένα κυκλώματα ASIC.

Ο Δ. Θεοδωρόπουλος στην [2] τοποθετεί 4 αντίγραφα του Ccproc στην xc4vlx200 τετραπλασιάζοντας το ρυθμό ολοκλήρωσης. Παρατηρώντας τον Πίνακα 6.1 είναι φανερό ότι το ίδιο μπορεί να γίνει και με τον CryptoDLX. Το Διάγραμμα 6.5 παρουσιάζει τις

επιδόσεις των τετραπύρηνων CryptoDLX και Ccproc σε σχέση με τους τρεις άλλους συνεπεξεργαστές κρυπτογράφησης .

Ο τελικός ρυθμός ολοκλήρωσης του AES-Rijndael αλγόριθμου στον CryptoDLX είναι 381Mbits/sec που είναι συγκρίσιμο νούμερο με τους ρυθμούς ολοκλήρωσης που δίνουν οι συνεπεξεργαστές κρυπτογράφησης. Το τελευταίο είναι αρκετά σημαντικό αν αναλογιστεί κανείς ότι οι συνεπεξεργαστές κρυπτογράφησης έχουν σχεδιαστεί από το μηδέν με βασικότερο στόχο την αποδοτικότερη επεξεργασία των κρυπτογραφικών αλγορίθμων.



Διάγραμμα 6.5 Σύγκριση της απόδοσης των τετραπύρηνων CryptoDLX και Ccproc με τις αποδόσεις διάφορων συνεπεξεργαστών κρυπτογράφησης

7. Συμπεράσματα και Μελλοντική Δουλειά

Όπως έχει αναλυθεί και στο εισαγωγικό κεφάλαιο αυτής της διπλωματικής η ανάπτυξη του διαδικτύου είναι ραγδαία. Ανάλογη είναι και η ανάπτυξη εφαρμογών που βασίζονται σε ενσωματωμένα συστήματα μικροεπεξεργαστών. Συνεπώς είναι πολλή σημαντική η επίτευξη ασφαλής μετάδοσης δεδομένων για να αποφευχθεί μόλυνση από ιούς, απώλεια προσωπικών πληροφοριών καθώς και κάθε είδους ηλεκτρονικό έγκλημα από κακόβουλους χρήστες.

Από τα παραπάνω γίνεται αντιληπτό πόσο σημαντικό θέμα είναι αυτό της κρυπτογραφίας και ότι πρέπει να λαμβάνεται υπόψη στον σχεδιασμό κάθε εφαρμογής που έχει να κάνει με ανταλλαγή δεδομένων. Η διπλωματική αυτή αρχικά εστίασε στην υλοποίηση ενός επεξεργαστή τύπου DLX και στη συνέχεια στην επέκταση της αρχιτεκτονικής του έτσι ώστε να επιτευχθεί μια σημαντική επιτάχυνση στην απόδοση του κατά την επεξεργασία συμμετρικών κρυπτογραφικών αλγορίθμων. Το αποτέλεσμα ήταν ο CryptoDLX επεξεργαστής που πέτυχε επιτάχυνση μέχρι και 7.54x σε σχέση με τον απλό DLX και απόδοση συγκρίσιμη με υλοποιήσεις συνεπεξεργαστών κρυπτογράφησης. Μερικά από τα χαρακτηριστικά του CryptoDLX είναι τα εξής:

- Πρόκειται για VLIW επεξεργαστή που εκδίδει εντολές των 128 bit, για να γίνει η επεξεργασία τους από 4 DLX πυρήνες κατάλληλα τροποποιημένους.
- Επιτάχυνση μέχρι και 5.52x σε σχέση με τον απλό DLX σε συσκευή FPGA της Xilinx.
- Η αρχιτεκτονική συνόλου εντολών είναι αρκετά ευέλικτη έτσι ώστε να υποστηρίξει αρκετούς συμμετρικούς κρυπτογραφικούς αλγόριθμους.
- Υποστηρίζει το σύνολο των αλγορίθμων που το NIST επέλεξε ως πρότυπα AES.
- Πετυχαίνει ρυθμό ολοκλήρωσης 381Mbits/sec στα 83,724MHz με ECB τρόπο λειτουργίας σε μια υλοποίηση που περιέχει 4 πυρήνες CryptoDLX.
- Μπορεί να υποστηρίζει το ασύρματο πρωτόκολλο 801.11g καθώς και το πρωτόκολλο Ethernet 802.3y 100 Mbits/sec.

Η ανάπτυξη του CryptoDLX ήταν μια διαδικασία που απαιτούσε συνεχή μελέτη της αρχιτεκτονικής του DLX όπως διαμορφωνόταν μετά από την προσθήκη κάθε νέας εντολής και αναζήτησης εναλλακτικών δομών για την υλοποίηση της επόμενης εντολής.

Το κόστος των εναλλακτικών δομών υπολογιζόταν είτε θεωρητικά είτε πρακτικά με την βοήθεια των εργαλείων της Xilinx και επιλεγόταν αυτή που θα έδινε καλύτερα στατιστικά απόδοσης στο τελικό σύστημα.

Τα στατιστικά απόδοσης του CryptoDLX είναι αρκετά ικανοποιητικά ωστόσο, όπως σε κάθε πρώτη έκδοση ενός επεξεργαστή, είναι εφικτή και περαιτέρω βελτίωση. Κάποιες από τις βελτιώσεις που θα μπορούσαν να γίνουν είναι οι παρακάτω:

- Η μνήμη δεδομένων και το αρχείο καταχωρητών που υπάρχουν σε κάθε τροποποιημένο DLX πυρήνα του CryptoDLX επεξεργαστή έχουν μέγεθος πολύ μεγαλύτερο από αυτό που απαιτούν οι συμμετρικοί κρυπτογραφικοί αλγόριθμοι για να δουλέψουν. Η σχεδίαση μικρότερων εκδόσεων των παραπάνω μνημών, που θα μπορούν όμως να προστεθούν εύκολα στην παρούσα αρχιτεκτονική συνόλου εντολών, θα οδηγούσε σε χαμηλότερο ποσοστό χρησιμοποίησης πόρων και υψηλότερη συχνότητα λειτουργίας. Το χαμηλότερο ποσοστό χρησιμοποίησης πόρων είναι πιθανό να επιτρέψει την τοποθέτηση ακόμα περισσότερων πυρήνων CryptoDLX σε μια συσκευή FPGA εκτοξεύοντας τον ρυθμό ολοκλήρωσης.
- Αύξηση του ρυθμού ολοκλήρωσης των εντολών που βρίσκονται στην ομοχειρία. Όπως έχει αναλυθεί και σε προηγούμενες ενότητες υπάρχουν κάποιες περιπτώσεις που στην ομοχειρία εισάγεται ανάσχεση λόγω κινδύνων δεδομένων που δεν μπορούν να επιλυθούν με προώθηση δεδομένων. Επίσης έχει αναλυθεί και το θέμα εκμετάλλευσης της «καθυστερημένης διακλάδωσης». Η κατάλληλη δρομολόγηση του κώδικα έτσι ώστε να αποφεύγονται όσο το δυνατόν περισσότερες εξαρτήσεις δεδομένων που οδηγούν σε ανασχέσεις αλλά και η επιλογή της εντολής που θα εκτελεστεί με την «καθυστερημένη διακλάδωση» είναι μια διαδικασία επίπονη και χρονοβόρα. Θα ήταν πολύ χρήσιμο να προστεθεί στον Python assembler μια λειτουργία που να κάνει τις παραπάνω δρομολογήσεις παράγοντας αποδοτικότερο κώδικα.

Από τα αρχαία χρόνια η κρυπτογραφία εξελίσσεται συνεχώς με διαχρονικό σκοπό την προστασία σημαντικών πληροφοριών. Ανέκαθεν ο στόχος ήταν να υπάρχει αντίσταση σε κάθε είδους επίθεση που μπορεί να προκύψει. Έτσι και στις μέρες μας στόχος των αρχιτεκτόνων υπολογιστών θα πρέπει να είναι η ανάπτυξη υλοποιήσεων που να έχουν την ευελιξία να αντισταθούν σε ένα συνεχώς αυξανόμενο σύνολο από τύπους ηλεκτρονικών επιθέσεων.

8. Αναφορές

[1] Joan Daemen, Vincent Rijmen, "*AES Proposal: Rijndael*", Document Version 2, March 9, 1999.

[2] Theodoropoulos Dimitris, "*CCproc: A custom VLIW cryptography co-processor for symmetric key ciphers*", October 2005.

[3] Intel Corporation, "*Intel Pentium Pro family*", 1996.

[4] Intel Corporation, "*Intel Pentium II processor at 350MHz, 400Mhz, 450 MHz*", August 1998.

[5] Intel Corporation, "*Intel Pentium III processor based on 0,13 micron process up to 1.33 GHz*", December 2001.

[6] Intel Corporation, "*Intel Pentium 4 processor with 512-KB L2 cache on 0.13 micron process*", February 2004.

[7] R.E. Kessler, E.J. McLellan, D.A. Webb, "*The Alpha 21264 microprocessor architecture*", Compaq Computer Corporation, January 1999.

[8] AMD Corporation, "*AMD Athlon processor model 4 datasheet*", November 2001.

[9] Alireza Hodjat, Ingrid Verbauwhede, "*A 21.54 Gbits/s Fully Pipelined AES Processor on FPGA*", IEEE Symposium on Field-Programmable Custom Computing Machines, April 2004.

[10] Xilinx Corporation, "*Virtex-II Pro and Virtex-II Pro X Platform FPGAs*", March 2005.

[11] Maire McLoone, John McCanny, "*Rijndael FPGA Implementations Utilizing Look-up tables*" , Journal of VLSI signal processing 34, 261-275, 2003.

[12] Xilinx Corporation, "*Virtex-E 1.8V Field programmable gate arrays*", July 2002.

[13] P. Chodowiec, P. Khuon, and K. Gaj, "*Fast Implementations of Secret-Key Block Ciphers Using Mixed Inner- and Outer-Round Pipelining*", In Symposium on Field Programmable Gate Arrays – FPGA 2001, pages 94–102. ACM Press, 2001.

[14] Xilinx Corporation, "*Virtex 2.5V Field programmable gate arrays*", April 2001.

[15] Maman Abdurohman, Sarwono Sutikno "*DLX Processor Enhancement for AES Rijndael crypto algorithm*", 2006.

[16] J. Burke, J. McDonald, T. Austin, "*Architectural Support for Fast Symmetric-Key Cryptography*", ASPLOS 2000.

[17] A. Murat Fiskiran and Ruby B. Lee, "*On-Chip Lookup Tables for Fast Symmetric-Key Encryption*", Proceedings of the IEEE 16th International Conference on Application-Specific Systems, Architectures and Processors (ASAP), pp. 356-363, July 23-25, 2005.

[18] Lisa Wu, Chris Weaver, and Todd Austin, "*Cryptomaniac: A Fast Flexible Architecture for Secure Communication*", ISCA 2001, June 2001.

[19] D. Oliva, R. Buchty, and N. Heintze, "*AES and the Cryptonite Crypto Processor*", Proc. Int. Conf. Compiler, Architectures and Synthesis for Embedded Systems, pp. 198-209, Oct. 2003.

[20] A. J. Elbirt, C. Paar, "*An Instruction-Level Distributed Processor for Symmetric-Key Cryptography*", IEEE Transactions on Parallel and Distributed Systems, 16(5), pp. 468-480, May 2005.

[21] John L. Hennessy, David A. Patterson, "*Computer Organization & Design . The Hardware-Software interface*", second edition 1997 .

[22] David A. Patterson, John L. Hennessy, "*Computer Architecture: A quantitative approach: Third edition*", Morgan Kaufman Publishers, Inc, 2003.

Διαδικτυακοί Τόποι:

[I1] **http://www.internetworldstats.com**

[I2] **http://www.codesandciphers.org.uk/enigma/**

[I3] **http://csrc.nist.gov/CryptoToolkit/aes/round1/round1.htm#algorithms**

[I4] **http://www.model.com**

[I5] **http://www.xilinx.com/**

Παράρτημα Α: Πλήρες Σύνολο Εντολών της DLX Υλοποίησης

Μορφή	Εντολή	Σύνταξη	Περιγραφή
R	add	add rd,rs,rt	Προσθέτει rs , rt αποθηκεύει στον rd
	sub	sub rd,rs,rt	Αφαιρεί τον rt απο τον rs αποθηκεύει στον rd
	or	or rd,rs,rt	Λογικό Ή μεταξύ rs , rt αποθηκεύει στον rd
	and	and rd,rs,rt	Λογικό ΚΑΙ μεταξύ rs , rt αποθηκεύει στον rd
	xor	xor rd,rs,rt	Αποκλειστικό Ή μεταξύ rs , rt αποθηκεύει στον rd
	shr	shr rd,rs,rt	Ολίσθηση του rs κατά rt θέσεις δεξιά, αποθηκεύει στον rd
	shl	shl rd,rs,rt	Ολίσθηση του rs κατά rt θέσεις αριστερά, αποθηκεύει στον rd
	ror	ror rd,rs,rt	Κυκλική ολίσθηση του rs κατά rt θέσεις δεξιά, αποθηκεύει στον rd
	rol	rol rd,rs,rt	Κυκλική ολίσθηση του rs κατά rt θέσεις αριστερά, αποθηκεύει στον rd
I	addi	addi rt,rs,Imm	Προσθέτει rs , Imm αποθηκεύει στον rt
	subi	subi rt,rs,Imm	Αφαιρεί το Imm απο τον rs αποθηκεύει στον rt
	andi	andi rt,rs,Imm	Λογικό ΚΑΙ μεταξύ rs , Imm αποθηκεύει στον rt
	ori	ori rt,rs,Imm	Λογικό Η μεταξύ rs , Imm αποθηκεύει στον rt
	xori	xori rt,rs,Imm	Αποκλειστικό Ή μεταξύ rs, Imm αποθηκεύει στον rt
	beqz	beqz rs,Imm	Αν ο rs είναι μηδέν διακλάδωση σε PC+4+επέκταση προσήμου(Imm)
	bnez	bnez rs,Imm	Αν ο rs δεν είναι μηδέν διακλάδωση σε PC+4+επέκταση προσήμου(Imm)
	lw	lw rt,Imm(rs)	Φόρτωση του rt από την Imm+rs διεύθυνση της μνήμης δεδομένων
	sw	sw rt,Imm(rs)	Αποθήκευση του rt στην Imm+rs διεύθυνση της μνήμης δεδομένων
	shri	shri rd,rs, Imm	Ολίσθηση του rs κατά Imm[5-0] θέσεις δεξιά, αποθηκεύει στον rd
	shli	shli rd,rs, Imm	Ολίσθηση του rs κατά Imm[5-0] θέσεις αριστερά, αποθηκεύει στον rd
	rori	rori rd,rs, Imm	Κυκλική ολίσθηση του rs κατα Imm[5-0] θέσεις δεξιά, αποθηκεύει στον rd
	roli	roli rd,rs, Imm	Κυκλική ολίσθηση του rs κατά Imm[5-0] θέσεις αριστερά, αποθηκεύει στον rd
	lui	lui rt,Imm	Φόρτωση των 16 msb του rt με το Imm
J	j	j label	Άλμα στη διεύθυνση «label»
	jr	jr rs	Άλμα στη διεύθυνση που περιέχει ο rs

Παράρτημα Β: Πλήρες Σύνολο Εντολών του CryptoDLX

Μορφή	Εντολή	Σύνταξη	Περιγραφή
R	add	add rd,rs,rt	Προσθέτει rs , rt αποθηκεύει στον rd
	sub	sub rd,rs,rt	Αφαιρεί τον rt απο τον rs αποθηκεύει στον rd
	or	or rd,rs,rt	Λογικό Ή μεταξύ rs , rt αποθηκεύει στον rd
	and	and rd,rs,rt	Λογικό ΚΑΙ μεταξύ rs , rt αποθηκεύει στον rd
	xor	xor rd,rs,rt	Αποκλειστικό Ή μεταξύ rs , rt αποθηκεύει στον rd
	shr	shr rd,rs,rt	Ολίσθηση του rs κατά rt θέσεις δεξιά, αποθηκεύει στον rd
	shl	shl rd,rs,rt	Ολίσθηση του rs κατά rt θέσεις αριστερά, αποθηκεύει στον rd
	ror	ror rd,rs,rt	Κυκλική ολίσθηση του rs κατά rt θέσεις δεξιά, αποθηκεύει στον rd
	rol	rol rd,rs,rt	Κυκλική ολίσθηση του rs κατά rt θέσεις αριστερά, αποθηκεύει στον rd
	addadd	addadd rd,rs,rt,rs2	Προσθέτει rs, rt, rs2, αποθηκεύει στον rd
	subadd	subadd rd,rs,rt,rs2	Αφαιρεί τον rt απο τον rs, προσθέτει τον rs2 στο αποτέλεσμα της αφαίρεσης και αποθηκεύει στον rd
	addsub	addsub rd,rs,rt,rs2	Προσθέτει rs και rt, αφαιρεί τον rs2 από το αποτέλεσμα της πρόσθεσης, αποθηκεύει στον rd
	subsub	subsub rd,rs,rt,rs2	Αφαιρεί τον rt απο τον rs, αφαιρεί τον rs2 από το αποτέλεσμα της αφαίρεσης και αποθηκεύει στον rd
	xoradd	xoradd rd,rs,rt,rs2	Λογικό xor μεταξύ rs, rt, προσθέτει τον rs2 στο αποτέλεσμα της λογικής πράξης, αποθηκεύει στον rd
	xorsub	xorsub rd,rs,rt,rs2	Λογικό xor μεταξύ rs, rt, αφαιρεί τον rs2 από το αποτέλεσμα της λογικής πράξης, αποθηκεύει στον rd
	addxor	addxor rd,rs,rt,rs2	Προσθέτει rs και rt, λογικό xor μεταξύ του rs2 και του αποτελέσματος της πρόσθεσης, αποθηκεύει στον rd
	subxor	subxor rd,rs,rt,rs2	Αφαιρεί τον rt απο τον rs, λογικό xor μεταξύ του rs2 και του αποτελέσματος της αφαίρεσης, αποθηκεύει στον rd
	xorxor	xorxor rd,rs,rt,rs2	Λογικό xor μεταξύ rs,rt,rs2
	mmult	mmult rd,rs,rt	Πολλαπλασιάζει rs με rt και αποθηκεύει στον rd
	gfm	gfm rd,rs	Πολλαπλασιασμός στο πεδίο GF(2^8) μεταξύ του rs κα του τελεστή x του GF, αποθήκευση στον rd
	ldgfopx	ldgfopx rs	Φόρτωση του τελεστή x του GF με το περιεχόμενο του καταχωρητή rs
	ldgfmr	ldgfmr rs	Φόρτωση του καταχωρητή που κρατάει το πολυώνυμο για την modulo πράξη με το περιεχόμενο του rs
	r2c/c2r	r2c/c2r rd rs	Μετατροπή των στηλών σε γραμμές και τον ανάποδο σε μια 128 bit ποσότητα δεδομένων που θεωρούνται 4 επί 4 πίνακας

Μορφή	Εντολή	Σύνταξη	Περιγραφή
R	aesX	aesX rd,rs	Πρόσβαση του rs στο Sbox κατά την κρυπτογράφηση ή αποκρυπτογράφηση (X=E,D) με το αποτέλεσμα να αποθηκεύεται στον rd
	marsX	marsX rd,rs	Πρόσβαση του rs στο Sbox σε κάποια μορφή λειτουργίας (X=F,BE) με το αποτέλεσμα να αποθηκεύεται στον rd
	serX	serX rd,rs	Πρόσβαση του rs στο Sbox κατά την κρυπτογράφηση ή αποκρυπτογράφηση (X=E,D) με το αποτέλεσμα να αποθηκεύεται στον rd
	tX	tsld rs,rs / tsbox rd,rs	Φορτώνει S0 και S1 με rsa και rsb αντίστοιχα / Πρόσβαση του rs στο Sbox με το αποτέλεσμα να αποθηκεύεται στον rd
	krfaddrz	krfaddrz	Μηδενίζει την διεύθυνση πρόσβασης στο αρχείο καταχωρητών των κλειδιών
	addkey	addkey rd,kr,rt	Προσθέτει kr , rt αποθηκεύει στον rd
	subkey	subkey rd,kr,rt	Αφαιρεί τον rt απο τον kr αποθηκεύει στον rd
	orkey	orkey rd,kr,rt	Λογικό Ή μεταξύ kr , rt αποθηκεύει στον rd
	andkey	andkey rd,kr,rt	Λογικό ΚΑΙ μεταξύ kr , rt αποθηκεύει στον rd
	xorkey	xorkey rd,kr,rt	Αποκλειστικό Ή μεταξύ kr , rt αποθηκεύει στον rd
	shrkey	shrkey rd,kr,rt	Ολίσθηση του kr κατά rt θέσεις δεξιά, αποθηκεύει στον rd
	shlkey	shlkey rd,kr,rt	Ολίσθηση του kr κατά rt θέσεις αριστερά, αποθηκεύει στον rd
	rorkey	rorkey rd,kr,rt	Κυκλική ολίσθηση του kr κατά rt θέσεις δεξιά, αποθηκεύει στον rd
	rolkey	rolkey rd,kr,rt	Κυκλική ολίσθηση του kr κατά rt θέσεις αριστερά, αποθηκεύει στον rd
	addaddkey	addaddkey rd,kr,rt,rs2	Προσθέτει kr, rt, rs2, αποθηκεύει στον rd
	subaddkey	subaddkey rd,kr,rt,rs2	Αφαιρεί τον rt απο τον kr, προσθέτει τον rs2 στο αποτέλεσμα της αφαίρεσης, αποθηκεύει στον rd
	addsubkey	addsubkey rd,kr,rt,rs2	Προσθέτει kr και rt, αφαιρεί τον rs2 από το αποτέλεσμα της πρόσθεσης, αποθηκεύει στον rd
	subsubkey	subsubkey rd,kr,rt,rs2	Αφαιρεί τον rt απο τον kr, αφαιρεί τον rs2 από το αποτέλεσμα της αφαίρεσης, αποθηκεύει στον rd
	xoraddkey	xoraddkey rd,kr,rt,rs2	Λογικό xor μεταξύ kr, rt, προσθέτει τον rs2 στο αποτέλεσμα της xor πράξης, αποθηκεύει στον rd
	xorsubkey	xorsubkey rd,kr,rt,rs2	Λογικό xor μεταξύ kr, rt, αφαιρεί τον rs2 από το αποτέλεσμα της xor πράξης, αποθηκεύει στον rd
	addxorkey	addxorkey rd,kr,rt,rs2	Προσθέτει kr και rt, λογικό xor μεταξύ του rs2 και αποτελέσματος της πρόσθεσης, αποθηκεύει στον rd
	subxorkey	subxor rd,kr,rt,rs2	Αφαιρεί τον rt απο τον kr, λογικό xor μεταξύ του rs2 και του αποτελέσματος της αφαίρεσης, αποθηκεύει στον rd
	xorxorkey	xorxorkey rd,kr,rt,rs2	Λογικό xor μεταξύ kr,rt,rs2 αποθήκευση στον rd

R	keyadd	keyadd kr,rs,rt	Προσθέτει rs , rt αποθηκεύει στον kr
	keysub	keysub kr,rs,rt	Αφαιρεί τον rt απο τον rs αποθηκεύει στον kr
	keyor	keyor kr,rs,rt	Λογικό Ή μεταξύ rs , rt αποθηκεύει στον kr
	keyand	keyand kr,rs,rt	Λογικό ΚΑΙ μεταξύ rs , rt αποθηκεύει στον kr
	keyxor	keyxor kr,rs,rt	Αποκλειστικό Ή μεταξύ rs , rt αποθηκεύει στον kr
	keyshr	keyshr kr,rs,rt	Ολίσθηση του rs κατά rt θέσεις δεξιά, αποθηκεύει στον kr
	keyshl	keyshl kr,rs,rt	Ολίσθηση του rs κατά rt θέσεις αριστερά, αποθηκεύει στον kr
	keyror	keyror kr,rs,rt	Κυκλική ολίσθηση του rs κατά rt θέσεις δεξιά, αποθηκεύει στον kr
	keyrol	keyrol kr,rs,rt	Κυκλική ολίσθηση του rs κατά rt θέσεις αριστερά, αποθηκεύει στον kr
	keyaddadd	keyaddadd kr,rs,rt,rs2	Προσθέτει rs, rt, rs2, αποθηκεύει στον kr
	keysubadd	keysubadd kr,rs,rt,rs2	Αφαιρεί τον rt απο τον rs, προσθέτει τον rs2 στο αποτέλεσμα της αφαίρεσης, αποθηκεύει στον kr
	keyaddsub	keyaddsub kr,rs,rt,rs2	Προσθέτει rs και rt, αφαιρεί τον rs2 από το αποτέλεσμα της πρόσθεσης, αποθηκεύει στον kr
	keysubsub	keysubsub kr,rs,rt,rs2	Αφαιρεί τον rt απο τον rs, αφαιρεί τον rs2 από το αποτέλεσμα της αφαίρεσης, αποθηκεύει στον kr
	keyxoradd	keyxoradd kr,rs,rt,rs2	Λογικό xor μεταξύ rs, rt, προσθέτει τον rs2 στο αποτέλεσμα της xor πράξης, αποθηκεύει στον kr
	keyxorsub	keyxorsub kr,rs,rt,rs2	Λογικό xor μεταξύ rs, rt, αφαιρεί τον rs2 από το αποτέλεσμα της xor πράξης, αποθηκεύει στον kr
	keyaddxor	keyaddxor kr,rs,rt,rs2	Προσθέτει rs και rt, λογικό xor μεταξύ του rs2 και του αποτελέσματος της πρόσθεσης, αποθηκεύει στον kr
	keysubxor	keysubxor kr,rs,rt,rs2	Αφαιρεί τον rt απο τον rs, λογικό xor μεταξύ του rs2 και του αποτελέσματος της αφαίρεσης, αποθηκεύει στον kr
	keyxorxor	keyxorxor kr,rs,rt,rs2	Λογικό xor μεταξύ rs,rt,rs2 αποθήκευση στον kr
I	addi	addi rt,rs,Imm	Προσθέτει rs , Imm αποθηκεύει στον rt
	subi	subi rt,rs,Imm	Αφαιρεί το Imm απο τον rs αποθηκεύει στον rt
	andi	andi rt,rs,Imm	Λογικό ΚΑΙ μεταξύ rs , Imm αποθηκεύει στον rt
	ori	ori rt,rs,Imm	Λογικό Η μεταξύ rs , Imm αποθηκεύει στον rt
	xori	xori rt,rs,Imm	Αποκλειστικό Ή μεταξύ rs, Imm αποθηκεύει στον rt
	lw	lw rt,Imm(rs)	Φόρτωση του rt από την Imm+rs διεύθυνση της μνήμης δεδομένων
	sw	sw rt,Imm(rs)	Αποθήκευση του rt στην Imm+rs διεύθυνση της μνήμης δεδομένων
	lui	lui rt,Imm	Φόρτωση των 16 msb του rt με το Imm
	ldlc	ldlc Imm	Φορτώνει τον καταχωρητή lc με την τιμή Imm

Μορφή	**Εντολή**	**Σύνταξη**	**Περιγραφή**
I	shri	shri rd,rs, Imm	Ολίσθηση του rs κατά Imm[5-0] θέσεις δεξιά, αποθηκεύει στον rd
	shli	shli rd,rs, Imm	Ολίσθηση του rs κατά Imm[5-0] θέσεις αριστερά, αποθηκεύει στον rd
	rori	rori rd,rs, Imm	Κυκλική ολίσθηση του rs κατα Imm[5-0] θέσεις δεξιά, αποθηκεύει στον rd
	roli	roli rd,rs, Imm	Κυκλική ολίσθηση του rs κατά Imm[5-0] θέσεις αριστερά, αποθηκεύει στον rd
R	movex	movex<instr> rd,rs,rt,rs2	Ο πυρήνας στέλνει δεδομένα στο στάδιο πρόσβασης μνήμης από τον x πυρήνα (x=a, b, c, d). Το <instr> θα γίνει μεταξύ rs, rt και rs2,αλλά το αποτέλεσμα δεν θα αποθηκευτεί στον rd. Μπορεί όμως να αποθηκευτεί το αποτέλεσμα από άλλους πυρήνες .Το <instr> μπορεί να πάρει τις ακόλουθες τιμές: add sub shr shl or rol and ror xor addadd addsub addxor subadd subsub subxor xoradd xorsub xorxor
J	loop	loop label	Διακλάδωση στη διεύθυνση που δείχνει το label