ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

**ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**

Εργαστήριο Μικροεπεξεργαστών και υλικού

Διπλωματική Εργασία

«ΜΕΛΕΤΗ ΚΑΙ ΠΕΙΡΑΜΑΤΙΚΗ ΔΙΑΤΑΞΗ ΓΙΑ ΥΨΗΛΗΣ ΤΑΧΥΤΗΤΑΣ ΣΕΙΡΙΑΚΗ ΕΠΙΚΟΙΝΩΝΙΑ ΜΕ FPGA»

Βασιλόπουλος Δημήτριος

Επιτροπή:
Δόλλας Απόστολος, Καθηγητής (Επιβλέπων)
Πνευματικάτος Διονύσιος, Αναπληρωτής Καθηγητής
Παπαευσταθίου Ιωάννης, Επίκουρος Καθηγητής

Χανιά 2009

Περίληψη

Στις μέρες μας, ένας σημαντικός παράγοντας που περιορίζει την πλήρη αξιοποίηση των δυνατοτήτων που μας προσφέρει η σύγχρονη τεχνολογία συσκευών αναδιατασσόμενης λογικής, είναι η έλλειψη ενός γρήγορου και αξιόπιστου τρόπου μεταφοράς δεδομένων στο επίπεδο του χρήστη μεταξύ ηλεκτρονικού υπολογιστή (Η/Υ) και FPGA (field-programmable gate array).

Στα πλαίσια αυτής της διπλωματικής μελετήσαμε και αναπτύξαμε μίας σύνδεσης υψηλής ταχύτητας για την μεταφορά δεδομένων, που χρησιμοποιεί το πρωτόκολλο Gigabit Ethernet. Για τον σκοπό ορίσαμε ένα απλό πρωτόκολλο μεταφοράς δεδομένων το MHL Transfer Protocol (MTP), στο επιπέδου εφαρμογής, και σχεδιάσαμε την αρχιτεκτονική πρωτοκόλλων χαμηλότερου επιπέδου για την υλοποίηση του τόσο σε υλικό όσο και σε λογισμικό.

Υλοποιήσαμε τη σχεδίαση σε μια αναπτυξιακή πλατφόρμα της οικογένειας Virtex-5 και επιδείξαμε τη λειτουργία της μεταφέροντας μεγάλο όγκο δεδομένων με ταχύτητα ως και 776.67 Mbps.

Περιεχόμενα

Περιεχόμενα 5

Κατάλογος Σχημάτων 7

Κατάλογος Πινάκων 8

1. Εισαγωγή 9

1.1 Σκοπός και συνεισφορά της εργασίας 9

1.2 Οργάνωση της εργασίας 10

2. Θεωρητικό Υπόβαθρο 13

2.1 Η στοίβα πρωτοκόλλων διαδικτύου 13

2.2 Επισκόπηση του πρωτοκόλλου UDP 15

2.3 Επισκόπηση του πρωτοκόλλου IPv4 16

2.4 Επισκόπηση του πρωτοκόλλου Ethernet 18

2.5 Διεπαφή φυσικού επιπέδου 19

3. Σχετική Έρευνα 21

3.1 Συναφείς Εργασίες 21

3.2 Ethernet-MAC πυρήνες για τα αναπτυξιακά συστήματα της Xilinx 21

4. Αρχιτεκτονική Ολοκλήρωσης Συστήματος 25

4.1 Ποια αναπτυξιακά συστήματα αφήσαμε 25

4.1.1 Virtex-2 Pro ML310 - XC2VP30 25

4.1.2 Virtex-4 ML410 - XC4VFX60 26

4.1.3 Virtex-4 ML401 - XC4VLX25 26

4.2 Το XUP Virtex-5 ML509 XC5VLX110T 26

4.2.1 Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC 27

4.3 Αρχιτεκτονική πρωτοκόλλων της σχεδίασης μας 29

4.3.1 Διαμόρφωση του Tri-Mode Ethernet MAC 29

4.3.2 Επιλογή πρωτοκόλλων για τα επίπεδα μεταφοράς και δικτύου 29

4.4 Το πρωτόκολλο επιπέδου εφαρμογής MHL Transfer Protocol 32
4.4.1 Η έκδοση MTP1.0 33
4.4.2 Η έκδοση MTP2.0 35
5. Σχεδίαση, Υλοποίηση και Επιβεβαίωση Λειτουργίας 39
5.1 Σχεδίαση 39
5.1.1 Το MTP1.0 39
5.1.2 Το MTP2.0 45
5.2 Υλοποίηση 57
5.2.1 Βασικά υποσυστήματα των δύο σχεδιάσεων 57
5.2.2 Αξιοποίηση πόρων της FPGA 58
5.2.3 Εφαρμογή ηλεκτρονικού υπολογιστή 59
5.3 Επιβεβαίωση ορθής λειτουργίας 61
5.3.1 Επιβεβαίωση λειτουργίας του Embedded Tri-Mode Ethernet MAC 61
5.3.2 Επιβεβαίωση σωστής λειτουργίας του MTP1.0 61
5.3.3 Επιβεβαίωση σωστής λειτουργίας του MTP2.0 62
6. Πειραματικά Αποτελέσματα 65
6.1 Πειραματικά αποτελέσματα του MTP1.0 65
6.2 Πειραματικά αποτελέσματα του MTP2.0 66
7. Συμπεράσματα και Μελλοντική Εργασία 69
7.1 Συμπεράσματα 69
7.2 Μελλοντική Εργασία 69
Βιβλιογραφία 71

Κατάλογος Σχημάτων

Σχήμα 1: Δομή πακέτου UDP 15

Σχήμα 2: Δομή πακέτου IPv4 16

Σχήμα 3: Δομή πλαισίου Ethernet 18

Σχήμα 4: Virtex-5 Embedded TEMAC υποστηριζόμενα επίπεδα OSI 23

Σχήμα 5: Το παράδειγμα σχεδίασης του Core Generator 27

Σχήμα 6: Local Link Interface Block Diagram 28

Σχήμα 7: Μεταφορά πλαισίου πάνω από διεπαφή Local Link (Xilinx) 29

Σχήμα 8: Η αρχιτεκτονική πρωτοκόλλων της σχεδίασης μας 31

Σχήμα 9: Δομή πακέτου MTP 33

Σχήμα 10: Παράδειγμα συνόδου MTP1.0 34

Σχήμα 11: Παράδειγμα συνόδου MTP2.0 36

Σχήμα 12: MTP1.0 Block Diagram 39

Σχήμα 13: Το mtp_module 41

Σχήμα 14: Η μονάδα ελέγχου RX control FSM 43

Σχήμα 15: Η κατάσταση MtpDataPacket_st 45

Σχήμα 16: MTP2.0 Block Diagram 45

Σχήμα 17: Το mtp_rx_module 46

Σχήμα 18: Το RX_Hold 48

Σχήμα 19: Timeout 49

Σχήμα 20: Η μονάδα ελέγχου RX control FSM 50

Σχήμα 21: Η κατάσταση WrOutFIFO_st 51

Σχήμα 22: Το mtp_tx_module 52

Σχήμα 23: Το Checksum Calculator 54

Σχήμα 24: Η μονάδα ελέγχου TX control FSM 55

Σχήμα 25: Η κατάσταση WrTxFIFO_st 56

Σχήμα 26: Τα διαγράμματα ροής των προγραμμάτων 60

Κατάλογος Πινάκων

Πίνακας 1: Τα αναπτυξιακά του εργαστηρίου [28-32] 24

Πίνακας 2: Η διεπαφή ανάγνωσης της OutFIFO 40

Πίνακας 3: Περιεχόμενα καταχωρητή κατάστασης 42

Πίνακας 4: Τα περιεχόμενα της μνήμης RX_ROM 47

Πίνακας 5: Περιεχόμενα καταχωρητή κατάστασης 49

Πίνακας 6: Τα περιεχόμενα της μνήμης TX_RAM_48x8 53

Πίνακας 7: Χρησιμοποίηση πόρων της FPGA από τη σχεδίαση MTP1.0 59

Πίνακας 8: Χρησιμοποίηση πόρων της FPGA από τη σχεδίαση MTP2.0 59

Πίνακας 9: Πειραματικές Μετρήσεις Ταχύτητας Μεταφοράς Δεδομένων του MTP1.0 65

Πίνακας 10: Εκτίμηση της Ταχύτητα Μεταφοράς Δεδομένων του MTP2.0 67

Κεφάλαιο 1

Εισαγωγή

Στο πρώτο κεφάλαιο αναφέρεται ο σκοπός και η συνεισφορά της παρούσας διπλωματικής εργασίας και περιγράφεται η δομή της.

1.1 Σκοπός και συνεισφορά της εργασίας

Η ανάπτυξη της τεχνολογίας αναδιατασσόμενης λογικής έχει προσφέρει στους μηχανικούς ένα εργαλείο για την ταχύτερη επεξεργασία δεδομένων. Ένα σημαντικό πρόβλημα που περιορίζει την αξιοποίηση των δυνατοτήτων που αυτής της τεχνολογίας είναι η έλλειψη ενός γρήγορου και αξιόπιστου τρόπου μεταφοράς δεδομένων στο επίπεδο του χρήστη μεταξύ ηλεκτρονικού υπολογιστή (Η/Υ) και FPGA (field-programmable gate array).

Στόχος της αυτής της εργασίας είναι η ανάπτυξη μίας σύνδεσης υψηλής ταχύτητας για την μεταφορά δεδομένων από τον Η/Υ στην FPGA, η οποία αξιοποιεί την τεχνολογία Gigabit Ethernet που υποστηρίζουν τα αναπτυξιακά συστήματα (development boards) της οικογένειας Virtex-5. Το σύστημα που θα υλοποιεί τη σύνδεση αυτή πρέπει να καταναλώνει όσο το δυνατών λιγότερους πόρους της FPGA και να μπορεί να συνδεθεί εύκολα με σχεδιάσεις άλλων χρηστών.

Για τον σκοπό αυτό χρησιμοποιήσαμε τον πυρήνα (core) V5 Embedded Tri-Mode Ethernet Media Access Controller (TEMAC), ο οποίος αποτελεί την βάση της σχεδίασης μας. Στη συνέχεια, μετά από μελέτη της στοίβας πρωτοκόλλων διαδικτύου (Internet stack) και με δεδομένη την λειτουργικότητα επιπέδου ζεύξης (Link layer), που μας παρείχε ο ελεγκτής της Xilinx, επιλέξαμε το πρωτόκολλο IPv4 στο επίπεδο δικτύου (Network layer) και το πρωτόκολλο UDP στο επίπεδο μεταφοράς (Transport layer). Στο επίπεδο εφαρμογής (Application layer), σχεδιάσαμε το πρωτόκολλο MTP (MHL Transfer Protocol) που αξιοποιεί τις υπηρεσίες των πρωτόκολλων UDP και IPv4 για να πραγματοποιήσει την μεταφορά δεδομένων.

Για την υλοποίηση του συστήματος μας το πρωτόκολλο MTP καθώς και η απαραίτητη λειτουργικότητα των πρωτοκόλλων UDP και IPv4 αποτυπώθηκαν σε υλικό και αναπτύχθηκαν κατάλληλες εφαρμογές λογισμικού για τον Η/Υ.

Το γεγονός ότι στο εργαστήριο δεν υπήρχε πρότερη εμπειρία σε αυτόν τον τομέα καθώς και η έλλειψη επαρκούς υποστήριξης για τα αναπτυξιακά συστήματα που είχαμε αρχικά στα χεριά μας από την εταιρία κατασκευής, δημιούργησαν σημαντικά εμπόδια καθώς ακόμα και για προβλήματα των οποίων η λύση ήταν σχετικά απλή καταβλήθηκε έντονη προσπάθεια και καταναλώθηκε πολύς χρόνος.

1.2 Οργάνωση της εργασίας

Στη συνέχεια η εργασία περιλαμβάνει ακόμα έξη κεφάλαια.

Στο δεύτερο κεφάλαιο περιγράφεται η στοίβα πρωτοκόλλων διαδικτύου, η οποία αποτέλεσε τη βάση της αρχιτεκτονικής πρωτοκόλλων της σχεδίασης μας. Επίσης, παρουσιάζεται μία επισκόπηση των πρωτοκόλλων UDP, IPv4 και Ethernet που χρησιμοποιήσαμε στην υλοποίηση μας. Τέλος, δίνεται μια περιληπτική περιγραφή των διεπαφών φυσικού επιπέδου-επίπεδο ζεύξης που είχαμε στη διάθεση μας.

Στο τρίτο κεφάλαιο γίνεται αναφορά σε συναφείς εργασίες και παρουσιάζονται συνοπτικά οι πυρήνες (IP-Intellectual property) της εταιρίας Xilinx που υποστηρίζουν οι οικογένειες αναπτυξιακών συστημάτων Virtex-5, Virtex-4 και Virtex-2 Pro.

Στο τέταρτο κεφάλαιο απαριθμούνται οι προσπάθειες που καταβάλαμε για να εγκαθιδρύσουμε επικοινωνία, με χρήση του πρωτοκόλλου Ethernet, μεταξύ του Η/Υ και των αναπτυξιακών συστημάτων που είχαμε στη διάθεση μας και εξηγούνται οι λόγοι που τελικά χρησιμοποιήσαμε το αναπτυξιακό σύστημα XUP Virtex-5. Στη συνέχεια τεκμηριώνονται οι σχεδιαστικές επιλογές των πρωτοκόλλων UDP για το επίπεδο μεταφοράς και IPv4 για το επίπεδο δικτύου και αναφέρεται η λειτουργικότητα που μας προσφέρουν το λειτουργικό σύστημα του υπολογιστή και ο Embedded Tri-Mode Ethernet MAC της Xilinx. Τέλος, περιγράφονται οι δύο εκδόσεις του πρωτοκόλλου, επιπέδου εφαρμογής, MTP που αναπτύξαμε για τη μεταφορά δεδομένων από τον υπολογιστή στο αναπτυξιακό.

Στο πέμπτο κεφάλαιο παρουσιάζεται το κομμάτι της σχεδίασης μας που υλοποιήθηκε στην FPGA και αναλύονται επιμέρους υποσυστήματα και υλοποιηστικές λεπτομέρειες. Παρουσιάζονται ακόμη τα εργαλεία που χρησιμοποιήσαμε για την ανάπτυξη και υλοποίηση της σχεδίασης μας καθώς και η διαδικασία που ακολουθήσαμε για την επιβεβαίωση της σωστής λειτουργίας της.

Στο έκτο κεφάλαιο παρουσιάζονται τα πειραματικά αποτελέσματα και απόδοση του συστήματος μας. Τέλος, στο έβδομο κεφάλαιο διατυπώνονται συμπεράσματα καθώς και προτάσεις για μελλοντική βελτίωση και επέκταση της σχεδίασης μας.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό περιγράφεται η στοίβα πρωτοκόλλων διαδικτύου. Επίσης, παρουσιάζονται συνοπτικά τα πρωτόκολλα UDP, IPv4 και Ethernet, που χρησιμοποιήσαμε στην υλοποίηση μας, και γίνεται αναφορά στις κύριες διεπαφές φυσικού επιπέδου-επίπεδο ζεύξης που .

2.1 Η στοίβα πρωτοκόλλων διαδικτύου

Η στοίβα πρωτοκόλλων διαδικτύου [1,3,34] είναι μια συλλογή πρωτοκόλλων επικοινωνίας στα οποία βασίζεται το Διαδίκτυο αλλά και μεγάλο ποσοστό των εμπορικών δικτύων. Αυτή η συλλογή πρωτοκόλλων είναι οργανωμένη σε επίπεδα το καθένα από τα οποία επιλύει συγκεκριμένα προβλήματα μεταφοράς δεδομένων και παρέχει καθορισμένες υπηρεσίες. Οι υπηρεσίες αυτές μπορεί να είναι μία ή περισσότερες από τις παρακάτω:

- *Έλεγχος σφαλμάτων (Error control),* που κάνει πιο αξιόπιστο το λογικό κανάλι ανάμεσα στην πλευρά μετάδοσης και την πλευρά λήψης ενός επιπέδου.

- *Έλεγχος ροής (Flow control),* που αποφεύγει τον κατακλυσμό ενός κόμβου του δικτύου με δεδομένα.

- *Τεμαχισμό και ανασύνθεση (Segmentation and reassembly),* που στην πλευρά μετάδοσης διαίρει μεγάλες μονάδες δεδομένων σε μικρότερα κομμάτια και στην πλευρά λήψης ανασυνθέτει τα μικρότερα κομμάτια στην αρχική μεγάλη ομάδα.

- *Πολυπλεξία (Multiplexing),* που επιτρέπει σε αρκετές συνόδους υψηλότερου επιπέδου να μοιράζονται μια μόνο σύνδεση χαμηλότερου επιπέδου.

- *Αποκατάσταση σύνδεσης (Connection setup),* που παρέχει δυνατότητα χειραψίας (handshaking) μεταξύ της πλευράς αποστολής και της πλευράς λήψης.

Ένα πρωτόκολλο ψηλότερου επιπέδου χρησιμοποιεί τις υπηρεσίες των πρωτοκόλλων χαμηλότερου επιπέδου για να λειτουργήσει. Τα υψηλότερα επίπεδα είναι πιο κοντά στη λογική του χρήστη και εξετάζουν πιο αφηρημένα δεδομένα,

χρησιμοποιώντας τις υπηρεσίες των πρωτοκόλλων χαμηλότερων επιπέδων για να μεταφράσουν τα δεδομένα σε μορφές που μπορούν να διαβιβαστούν με φυσικά μέσα.

Η στοίβα πρωτοκόλλων διαδικτύου αποτελείται από τέσσερα επίπεδα:

Επίπεδο Εφαρμογής (Application Layer)

Το επίπεδο εφαρμογής είναι υπεύθυνο για την υποστήριξη εφαρμογών δικτύου. Οι εφαρμογές παραδίδουν τα δεδομένα σε μορφή που ορίζουν οι ίδιες. Στο επίπεδο αυτό υλοποιούνται οι λειτουργίες παρουσίασης και συνόδου οι οποίες διευκολύνονται με τη χρήση βιβλιοθηκών (Network Libraries).

Επίπεδο Μεταφοράς (Transport Layer)

Το επίπεδο μεταφοράς παρέχει την υπηρεσία μεταφοράς μηνυμάτων του επιπέδου εφαρμογής, ανεξαρτήτως του υποκείμενου δικτύου, με έλεγχο σφαλμάτων, κατάτμηση και ρύθμιση ροής. Στο διαδίκτυο υπάρχουν δύο πρωτόκολλα του επιπέδου μεταφοράς το TCP (Transmission Control Protocol) [10] και το UDP (User Datagram Protocol) [8]. Το TCP παρέχει μια συνδεσιστρεφή (connection-oriented) υπηρεσία που περιλαμβάνει εγγυημένη παράδοση μηνυμάτων επιπέδου εφαρμογής και έλεγχο ροής. Το TCP επίσης τεμαχίζει μεγάλα μηνύματα σε μικρότερα τμήματα και παρέχει ένα μηχανισμό ελέγχου συμφόρησης. Το UDP παρέχει μια ασυνδεσιστρεφή (connectionless) υπηρεσία μεταφοράς μηνυμάτων επιπέδου εφαρμογής. Χρησιμοποιείται κυρίως σε εφαρμογές όπου η έγκαιρη άφιξη των δεδομένων είναι πιο σημαντική από την ακεραιότητα τους.

Επίπεδο Δικτύου (Network Layer)

Το επίπεδο δικτύου είναι υπεύθυνο για τη δρομολόγηση πακέτων δεδομένων μεταξύ της πλευράς αποστολής και της πλευράς λήψης. Αυτό προϋποθέτει συνήθως την δρομολόγηση πακέτων διαμέσου ενός δικτύου δικτύων ή διαδικτύου. Το επίπεδο δικτύου έχει δύο συστατικά. Έχει το πρωτόκολλο IP [9] που ορίζει τα πεδία στο πακέτο IP και το πώς τα τερματικά συστήματα και οι δρομολογητές (routers) ενεργούν στα πεδία αυτά. Έχει ακόμη πρωτόκολλα δρομολόγησης τα οποία καθορίζουν τις διαδρομές που ακολουθούν τα δεδομένα από την πηγή στον προορισμό.

Επίπεδο Ζεύξης (Link Layer)

Το επίπεδο ζεύξης δρομολογεί ένα πακέτο μέσω μιας σειράς δρομολογητών ανάμεσα στην προέλευση και στον προορισμό. Για τη μεταφορά ενός πακέτου από ένα κόμβο

στον επόμενο το επίπεδο δικτύου χρησιμοποιεί τις υπηρεσίες του επιπέδου ζεύξης. Καθώς τα πακέτα συνήθως πρέπει να διασχίζουν αρκετές ζεύξεις για να μεταβούν από την πηγή στον προορισμό, ένα πακέτο μπορεί να τύχει χειρισμού από διάφορα πρωτόκολλα επιπέδου ζεύξης.

2.2 Επισκόπηση του πρωτοκόλλου UDP

Το UDP [8] είναι ένα απλό και ελαφρύ πρωτόκολλο μεταφοράς, με ένα μινιμαλιστικό μοντέλο υπηρεσίας. Το UDP είναι ασυνδεσιστρεφές -δεν υπάρχει διαδικασία χειραψίας πριν αρχίσουν δύο διεργασίες να επικοινωνούν- και παρέχει μια αναξιόπιστη υπηρεσία μεταφοράς επομένως ένα δεδομενόγραμμα μπορεί να φτάσει στον προορισμό εκτός σειράς, δύο φορές ή να χαθεί χωρίς ειδοποίηση. Επίσης, το UDP δεν περιλαμβάνει κάποιον μηχανισμό ελέγχου συμφόρησης οπότε μια διεργασίας μπορεί να τροφοδοτεί με δεδομένα ένα UDP socket [1,4] με όποιο ρυθμό θέλει. Χρονικά ευαίσθητες εφαρμογές (πχ Voice over IP) χρησιμοποιούν συχνά UDP η απόρριψη πακέτων είναι προτιμότερη από την αναμονή καθυστερημένα πακέτα.

Δομή πακέτου UDP

Η δομή του πακέτου UDP φαίνεται στο Σχήμα 1. Τα δεδομένα της εφαρμογής καταλαμβάνουν το πεδίο δεδομένων στο πακέτο UDP. Η κεφαλίδα UDP έχει μόνο τέσσερα πεδία το καθένα μήκους 16 bits.

bits	0 - 15	16 - 31
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

Σχήμα 1: **Δομή πακέτου UDP** (wikipedia)

- *Θύρα προέλευσης (Source port).* Το πεδίο αυτό προσδιορίζει τη θύρα της διεργασίας αποστολής και όταν έχει νόημα είναι η θύρα να σταλεί απάντηση αν χρειαστεί. Όταν δεν χρησιμοποιείται πρέπει να έχει την τιμή μηδέν.

- *Θύρα προορισμού (Destination port).* Το πεδίο αυτό προσδιορίζει την θύρα της διεργασίας του παραλήπτη.

- *Μήκος (Length).* Το πεδίο αυτό καθορίζει το μήκος σε bytes ολόκληρου του πακέτου (κεφαλίδα + δεδομένα). Το ελάχιστο μήκος είναι 8 bytes δεδομένου ότι είναι το μήκος της κεφαλίδας.

- *Άθροισμα ελέγχου (Checksum).* Το άθροισμα ελέγχου χρησιμοποιείται για τον έλεγχο λαθών της κεφαλίδας και των δεδομένων. Ο αλγόριθμος για τον υπολογισμό του αθροίσματος ελέγχου είναι διαφορετικός για μεταφορά πακέτων UDP χρησιμοποιώντας τα πρωτόκολλα IPv4 και IPv6.

2.3 Επισκόπηση του πρωτοκόλλου IPv4

Το IPv4 (Internet Protocol version 4) [9,35] είναι ένα ασυνδεσιστρεφές πρωτόκολλο που έχει σχεδιαστεί να χρησιμοποιείται με δίκτυα μεταγωγής πακέτων του επιπέδου ζεύξης. Είναι ένα αναξιόπιστο πρωτόκολλο υπό τη έννοια ότι δεν εξασφαλίζει την παράδοση, την ορθή αλληλουχία ή την αποτροπή της διπλής παράδοσης των μεταφερόμενων πακέτων. Οι προηγούμενες λειτουργίες συμπεριλαμβανομένης της ακεραιότητας των δεδομένων επαφίενται κάποιο από τα ανώτερα επίπεδα της στοίβας πρωτοκόλλων. Το IPv4 εγγυάται μόνο την ακεραιότητα της κεφαλίδας η οποία επιτυγχάνεται με ένα άθροισμα ελέγχου. Τέλος, το πρωτόκολλο IPv4 υλοποιεί την λειτουργικότητα κατακερματισμού των πακέτων IPv4 σε μικρότερα και στη συνέχεια ανασύνθεσης τους στον τελικό προορισμό.

Δομή πακέτου IPv4

Η δομή του πακέτου IPv4 φαίνεται στο Σχήμα 2. Τα κύρια πεδία του πακέτου είναι τα παρακάτω:

bit offset	0–3	4–7	8–15	16–18	19–31
0	Version	Header length	Differentiated Services	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	Options				
160 or 192+	Data				

Σχήμα 2: **Δομή πακέτου IPv4** (wikipedia)

- *Αριθμός έκδοσης (Version).* Αυτά τα τέσσερα bit καθορίζουν την έκδοση του πρωτοκόλλου IP του πακέτου.

- *Μήκος κεφαλίδας (Header length).* Αυτά τα τέσσερα bit καθορίζουν που ακριβώς αρχίζουν τα δεδομένα μέσα στο πακέτο.

- *Τύπος υπηρεσίας (Type of service - TOS).* Τα bit του τύπου υπηρεσίας περιελήφθησαν στην κεφαλίδα του IPv4 για να επιτρέπουν τη διάκριση των διαφόρων τύπων πακέτων IPv4 (πχ πακέτα πραγματικού χρόνου).

- *Μήκος πακέτου (Total length).* Αυτό είναι το συνολικό μήκος του πακέτου IP (κεφαλίδα συν δεδομένα), μετρούμενο σε bytes.

- *Ταυτότητα (Identification), ενδείξεις (flags), μετατόπιση κατακερματισμού (fragment offset).* Αυτά τα τρία πεδία καθορίζουν τον κατακερματισμό των πακέτων IPv4.

- *Διάρκεια ζωής (Time to live - TTL).* Το πεδίο διάρκεια ζωής περιλαμβάνεται για να σιγουρέψει ότι τα πακέτα δεν κυκλοφορούν για πάντα μέσα στο δίκτυο. Η τιμή του πεδίου αυτού μειώνεται κάθε φορά που γίνεται επεξεργασία του πακέτου από ένα δρομολογητή. Αν το πεδίο TTL λάβει την τιμή μηδέν, τότε το πακέτο πρέπει να απορριφθεί.

- *Πρωτόκολλο (Protocol).* Η τιμή του πεδίου αυτού δηλώνει το πρωτόκολλο του επιπέδου μεταφοράς στο οποίο τα δεδομένα αυτού του πακέτου πρέπει να μεταβιβαστούν.

- *Άθροισμα ελέγχου κεφαλίδας (Header checksum).* Το άθροισμα ελέγχου κεφαλίδας χρησιμοποιείται για την ανίχνευση σφαλμάτων στην κεφαλίδα του πακέτου IPv4.

- *Διευθύνσεις IP προέλευσης και προορισμού (Source and destination IP addresses).* Αυτά τα πεδία περιέχουν τις διευθύνσεις IP (32 bit) της πηγής και του τελικού προορισμού για το συγκεκριμένο πακέτο IPv4.

- *Επιλογές (Options).* Αυτά τα πεδία επιτρέπουν σε μία κεφαλίδα IP να επεκταθεί. Οι επιλογές κεφαλίδας σχεδιάστηκαν να χρησιμοποιούνται σπάνια.

- *Δεδομένα (Data).* Το πεδίο αυτό περιέχει τα δεδομένα που μεταφέρει το πακέτο IPv4.

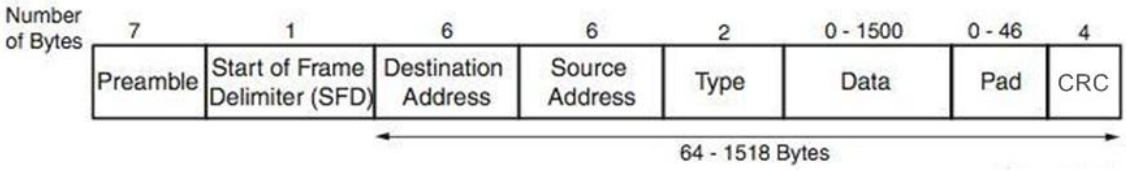
2.4 Επισκόπηση του πρωτοκόλλου Ethernet

Το πρωτόκολλο Ethernet (IEEE 802.3) [2,5,36] είναι μια οικογένεια τεχνολογιών δικτύωσης υπολογιστών βασισμένων στη μεταγωγή πλαισίων (frames). Καθορίζει μια σειρά από πρότυπα καλωδίωσης και σηματοδοσίας για το φυσικό επίπεδο (physical layer) του μοντέλου OSI μέσω του Media Access Control (MAC) [37] του επιπέδου ζεύξης δεδομένων (data link layer) καθώς και ένα κοινής μορφοποίησης διευθύνσεων. Σήμερα είναι το επικρατών πρότυπο για τα ενσύρματα δίκτυα τοπικής περιοχής (Local Area Networks - LAN).

Οι κύριες τεχνολογίες του Ethernet είναι: 10 Mbps (Ethernet), 100 Mbps (Fast Ethernet), 1 Gbps (Gigabit Ethernet), 10 Gbps (10-Gigabit Ethernet) και 100 Gbps (100-Gigabit Ethernet) – σε φάση ανάπτυξης. Οι παραπάνω τεχνολογίες δεν διαφέρουν μόνο στο εύρους ζώνης που παρέχουν αλλά και στο φυσικό μέσο που χρησιμοποιούν (καλώδια ομοαξονικά / συνεστραμμένων ζευγών / οπτικών ινών).

Δομή πλαισίου Ethernet

Υπάρχουν αρκετές παραλλαγές των πλαισίων Ethernet. Η πιο διαδεδομένη παραλλαγή σήμερα είναι η Ethernet II frame [38] (γνωστή και ως DIX Ethernet) επειδή χρησιμοποιείται από την στοίβα πρωτοκόλλων διαδικτύου. Το Σχήμα 2-3 παρουσιάζει την δομή του πλαισίου Ethernet II.



Σχήμα 3: **Δομή πλαισίου Ethernet**

- *Προοίμιο (Preamble).* Κάθε πλαίσιο Ethernet αρχίζει με ένα πεδίο προοιμίου μήκος 7 bytes που έχουν τιμή 10101011. Το πεδίο βοηθούν τον δέκτη να αναγνωρίσει το πακέτο και να συγχρονίσει το ρολόι του με αυτό του δέκτη.

- *Δείκτης αρχής πλαισίου (Start of frame delimiter - SFD).* Προσδιορίζει την αρχή του πλαισίου Ethernet.

- *Διεύθυνση MAC προορισμού (Destination MAC address).* Το πεδίο αυτό περιέχει την φυσική διεύθυνση του προορισμού.

- *Διεύθυνση MAC πηγής (Source MAC address).* Το πεδίο αυτό περιέχει την φυσική διεύθυνση της πηγής που εκπέμπει το πλαίσιο.

- *Τύπος (Type).* Το πεδίο αυτό επιτρέπει στο Ethernet να κάνει πολύπλεξη πρωτοκόλλων επιπέδου δικτύου.

- *Δεδομένα (Data), zero pad.* Αυτό το πεδίο μεταφέρει το πακέτο IP. Η μέγιστη μονάδα μεταφοράς του Ethernet είναι 1500 bytes. Αυτό σημαίνει ότι αν το πακέτο IP είναι μεγαλύτερο από 1500 bytes πρέπει να κατακερματιστεί. Το ελάχιστο μέγεθος του πεδίου δεομένων είναι 46 bytes επομένως να το πακέτο IP είναι μικρότερο από 46 bytes το πεδίο pad γεμίζεται με μηδενικά ώστε το πεδίο δεδομένων συν το πεδίο zero pad να φτάσουν στα 64 bytes.

- *Κυκλικός έλεγχος πλεονασμού (Cyclic Redundancy Checking - CRC).* Σκοπός του πεδίου αυτού είναι να επιτρέπει την ανίχνευση σφαλμάτων στο μεταδιδόμενο πλαίσιο.

- *Διάκενο μεταξύ των πλαισίων (Inter Frame Gap - IFG).* Μεταξύ της αποστολής διαδοχικών πλαισίων πρέπει να παρεμβάλλεται ένα κενό μεγέθους τουλάχιστο 12 bytes.

2.5 Διεπαφή φυσικού επιπέδου

Για τη διασύνδεση του επιπέδου ζεύξης με το φυσικό επίπεδο έχουν οριστεί μια σειρά από διεπαφές οι πιο σημαντικές από τις οποίες αναφέρονται παρακάτω.

Η **Media Independent Interface (MII)** [36,39] είναι μία τυποποιημένη παράλληλη διεπαφή που χρησιμοποιείται για τη διασύνδεση του υποεπιπέδου MAC ενός Fast Ethernet (100 Mbps) ή ενός Ethernet (10 Mbps) ελεγκτή με το φυσικό επίπεδο (PHY). Η μεταφορά δεδομένων γίνεται χρησιμοποιώντας λέξεις εύρους 4 bit σε κάθε κατεύθυνση και ρολόι 25 MHz για ταχύτητα 100 Mbps και 2.5 MHz για ταχύτητα 10 Mbps. Η ονομασία «ανεξάρτητο του μέσου» (media independent) υποδηλώνει ότι διαφορετικά είδη υλικού μπορούν να χρησιμοποιηθούν χωρίς να απαιτείται επανασχεδίαση ή αντικατάσταση υλικού του block MAC.

Η **Reduced Media Independent Interface (RMII)** [40] είναι μία πρότυπη διεπαφή που υλοποιεί τη διασύνδεση του υποεπιπέδου MAC ενός Fast Ethernet

(100 Mbps) ή ενός Ethernet (10 Mbps) πομποδέκτη με το φυσικό επίπεδο (PHY). Συγκρινόμενο με το MII, το RMII μειώνει τον αριθμό των pin που απαιτούνται για τη διασύνδεση από 16 (στο MII) σε 6 ως 10 μεταφέροντας λέξεις εύρους 2 bit με ρολόι 50 MHz.

Η **Gigabit Media Independent Interface (GMII)** [36,41] είναι μια διασύνδεση μεταξύ του υποεπιπέδου MAC και του φυσικού επιπέδου (PHY). Η διεπαφή χρησιμοποιεί 24 pin για λέξεις εύρους 8 bit σε κάθε κατεύθυνση, ρολόι 125 MHz και υποστηρίζει ταχύτητα ως και 1000 Mbps. Το GMII είναι συμβατό με το MII και μπορεί να λειτουργήσει σε υποβαθμισμένες ταχύτητες 100 / 10 Mbps.

Η **Reduced Gigabit Media Independent Interface (RGMII)** [42] είναι μια διασύνδεση του υποεπιπέδου MAC και του φυσικού επιπέδου (PHY) που υποστηρίζει ταχύτητές 1000/100/10 Mbps. Σε σύγκριση με το GMII το RGMII χρησιμοποιεί τα μισά pin (12 έναντι 24). Η μείωση αυτή επιτυγχάνεται με τον συγχρονισμό των δεδομένων τόσο στη θετική όσο και στην αρνητική ακμή του ρολογιού καθώς και με την απαλοιφή των μη ουσιωδών σημάτων (carrier-sense and collision-indication).

Η **Serial Gigabit Media Independent Interface (SGMII)** [43] είναι μια τυποποιημένη διεπαφή για την διασύνδεση του υποεπιπέδου MAC μιας σύνδεσης Ethernet με το φυσικό επίπεδο PHY. Υποστηρίζει ταχύτητες 1000/100/10 Mbps και αξιοποιεί διαφορικά ζεύγη για τα σήματα ρολογιού και δεδομένων τα οποία χρησιμοποιούνται και για αποστολή και για λήψη δεδομένων. Διαφέρει από το GMII επειδή έχει χαμηλότερη κατανάλωση ισχύος και μικρότερο αριθμό pin (8 συνολικά).

Η **10 Gigabit Media Independent Interface (XGMII)** [44] είναι μια πρότυπη διεπαφή για πλήρως αμφίδρομη σύνδεση (full-duplex) 10 Gigabit Ethernet. Αποτελείται από δύο διαδρομές δεομένων (πομπός και δέκτης) εύρους 32 bit και δύο διαδρομές ελέγχου ροής εύρους 4 bit που λειτουργούν σε συχνότητα ρολογιού 156,25 MHz.

Κεφάλαιο 3

Σχετική Έρευνα

Στο κεφάλαιο αυτό γίνεται αναφορά σε συναφείς εργασίες και παρουσιάζονται συνοπτικά οι πυρήνες (IP-Intellectual property) της εταιρίας Xilinx που υποστηρίζουν οι οικογένειες αναπτυξιακών συστημάτων Virtex-5, Virtex-4 και Virtex-2 Pro.

3.1 Συναφείς Εργασίες

Οι H. Fallside και M. Smith [6] προχώρησαν στη σχεδίαση μιας πλατφόρμας που συνδέει άμεσα ένα σύστημα αναδιατασσόμενης λογικής στο διαδίκτυο. Χαρακτηριστικό της πλατφόρμας αυτής είναι η υλοποίηση πρωτοκόλλων δικτύου (TCP/IP) που επιτρέπουν την μεταφορά από και προς την FPGA τόσο δεδομένων εφαρμογής όσο και δεδομένων διαμόρφωσης μέσω του διαδικτύου. Η εργασία αυτή απέδειξε ότι είναι δυνατό να αντικατασταθούν τα συμβατικά συστήματα μικροεπεξεργαστή, που υλοποιούν εφαρμογές δικτύου, από σχεδιάσεις σε αναδιατασσόμενη λογική.

Οι Α. Δόλλας, Ι. Ερμής, Ι. Κοϊδής και οι φοιτητές του [7] προχώρησαν στην ανάπτυξη ενός πυρήνα ανοιχτού κώδικα που υλοποιεί σχεδόν ολόκληρο πρωτόκολλο TCP/IP σε VHDL. Η αρχική σχεδίαση χρησιμοποιήθηκε για την ανάπτυξη τεχνογνωσίας στην βελτίωση της απόδοσης πολύπλοκων πρωτοκόλλων όπως το TCP/IP και λειτούργησε σε ταχύτητα 700 Mbps. Ο πυρήνας μπορεί να τροποποιηθεί ώστε να δουλεύει σε ταχύτητες της τάξης των Gbps με την εφαρμογή κατάλληλης διασύνδεσης φυσικού επιπέδου-επιπέδου ζεύξης.

3.2 Ethernet-MAC πυρήνες για τα αναπτυξιακά συστήματα της Xilinx

Οι οικογένειες αναπτυξιακών συστημάτων Virtex-2 Pro, Virtex-4 και Virtex-5 υποστηρίζουν τα παρακάτω IP.

Το **10 Gigabit Ethernet Media Access Controller (10GEMAC)** [14,15] προορίζεται για την ανάπτυξη εφαρμογών Gigabit Ethernet. Είναι συμβατό με την διεπαφή φυσικού επιπέδου – υποεπιπέδου MAC Ten Gigabit Media Independent Interface (XGMII) και παρέχει bandwidth μέχρι 10Gbps.

Το **Gigabit Ethernet Media Access Controller (GEMAC)** [18,19] προορίζεται για την ανάπτυξη εφαρμογών επικοινωνιών που απαιτούν ταχύτητες Gigabit Ethernet. Το GEMAC είναι συμβατό με τις διεπαφές φυσικού επιπέδου – υποεπιπέδου 8-bit Gigabit Media Independent Interface (GMII), 4-bit Reduced Gigabit Media Independent Interface (RGMII) και Serial Gigabit Media Independent Interface (SGMII).

Το **Tri-Mode Ethernet MAC (TEMAC)** [22,23] προορίζεται για την ανάπτυξη εφαρμογών επικοινωνιών που απαιτούν ταχύτητες Gigabit Ethernet. Επίσης, έχει την δυνατότητα να λειτουργεί σε ταχύτητες 100 (Fast Ethernet) και 10 Mbps (Ethernet). Είναι συμβατό με τις διεπαφές GMII/MII, RGMII και SGMII.

Το **XAUI** [16,17] υποστηρίζει τα αναπτυξιακά Virtex-5, Virtex-4FX και Virtex-2 Pro. Προορίζεται για εφαρμογές που απαιτούν ρυθμούς μεταφοράς δεδομένα 10 Gbps, για επικοινωνία chip-to-chip και για διασύνδεση συσκευών (devices) 10 Gigabit Ethernet MAC.

Το **Ethernet 1000BASE-X PCS/PMA** [20,21] υποστηρίζει τα αναπτυξιακά Virtex-5LXT, Virtex-4FX και Virtex-2 Pro. Προορίζεται για εφαρμογές που απαιτούν τη μεταφορά πλαισίων Ethernet σε μέσω διαφορετικό από το Ethernet, για διεπαφή Ethernet με άλλες πρότυπες διεπαφές και για chip-to-chip επικοινωνία. Είναι συμβατό με τις διεπαφές 1000BASE-X PCS και 1000BASE-X PCS/PMA.

Οι FPGA της οικογένειας Virtex-4FX διαθέτουν δύο ενσωματωμένους Tri-Mode Ethernet MAC ελεγκτές οι οποίοι μπορούν να διαμορφωθούν με το IP **Embedded Tri-mode Ethernet MAC Wrapper** [24,25]. Τα βασικά χαρακτηριστικά του είναι:

- Παρέχει στον χρήστη διαμορφώσιμη διεπαφή φυσικού επιπέδου για τον ενσωματωμένο Tri-mode Ethernet MAC ελεγκτή.
- Απλοποιεί τη διασύνδεση μεταξύ του Ethernet MAC και του PHY.
- Προσδιορίζει την διαμόρφωση των EMAC0/EMAC1 με βάση τις επιλογές του χρήστη.

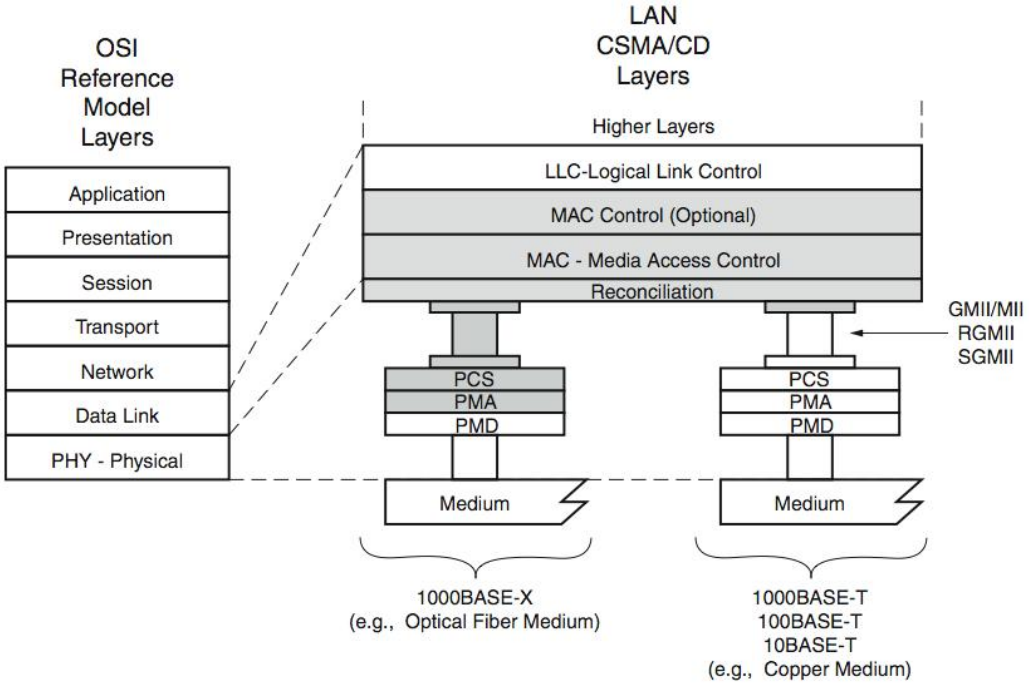
- Είναι συμβατό με τις διεπαφές MII, GMII, RGMII SGMII, και 1000BASE-X PCS / PMA και παράγει τα απαραίτητα ρολόγια και λογική για την σωστή λειτουργία τους.

- Δημιουργεί αρχεία VHDL ή Verilog για την ενσωμάτωση του IP σε μεγαλύτερες σχεδιάσεις καθώς και τα απαραίτητα αρχεία testbench για την προσομοίωση και το εργαλείο Modelsim.

- Παράγει παραδείγματα αρχείων UCF.

Οι FPGA της οικογένειας Virtex-5 διαθέτουν δύο ενσωματωμένους Tri-Mode Ethernet MAC ελεγκτές οι οποίοι μπορούν να διαμορφωθούν με το IP **Virtex-5 Embedded Tri-mode Ethernet MAC Wrapper** [26,27]. Τα βασικά χαρακτηριστικά του και λειτουργίες του Virtex-5 Embedded Tri-mode Ethernet MAC Wrapper είναι όμοια με αυτά του αντίστοιχου για την οικογένεια Virtex-4.

Στο Σχήμα 4 δείχνει τη σχέση μεταξύ του μοντέλου αναφοράς OSI και του πρωτοκόλλου Ethernet MAC, όπως ορίζεται στην προδιαγραφή IEEE 802.3. Με γκρι απεικονίζονται τα επίπεδα και η λειτουργικότητα που χειρίζεται ο Embedded Tri-mode Ethernet MAC ελεγκτής.



Σχήμα 4: **Virtex-5 Embedded TEMAC υποστηριζόμενα επίπεδα OSI** (Xilinx)

Τα αναπτυξιακά συστήματα που διαθέτει το εργαστήριο

Στον Πίνακα 1 παρουσιάζονται συνοπτικά τα IP Ethernet-MAC που υποστηρίζουν τα αναπτυξιακά της εταιρίας Xilinx που έχουμε στο εργαστήριο.

Τεχνολογίες EMAC[1]	**ML310 (2VP30)**	**ML401 (4VLX25)**	**ML410 (4VFX60)**	**ML509 (5VLX110T)**
10GEMAC	*	●	●	●
XAUI	○	○	●	●
GEMAC	**	●	●	●
1000BASE-X PCS/PMA	**	○	●	●
TEMAC	●	●	●	●
V4-Embedded TEMAC	○	○	●	○
V5-Embedded TEMAC	○	○	○	●

Πίνακας 1: **Τα αναπτυξιακά του εργαστηρίου** [28-32]

Το Virtex-2 Pro ML310 - XC2VP30 [29] έχει έναν Fast Ethernet PHY που υποστηρίζει πλήρως αμφίδρομη λειτουργία καθώς και λειτουργίες autonegotiation και παράλληλης ανίχνευσης σε ταχύτητες 10 Mbps και 100 Mbps. Υποστηρίζει τη διασύνδεση MII με τον 10/100 Media Access Controller που υλοποιείται στην FPGA.

Το Virtex-4 ML401 - XC4VLX25 [30] ένα PHY που λειτουργεί σε ταχύτητες 10/100/1000 Mbps και υποστηρίζει τις διασυνδέσεις MII, GMII και RGMII. Ένας κρύσταλλος συχνότητας 25 MHz παράγει το σήμα ρολογιού για τη λειτουργία του PHY.

Το Virtex-4 ML410 - XC4VFX60 [31] έχει δύο Ethernet PHY που λειτουργούν σε ταχύτητες 10/100/1000 Mbps. Το πρώτο υποστηρίζει τις διασυνδέσεις MII και RGMII και το δεύτερο τη διασύνδεση SGMII. Κάθε PHY διαθέτει κρύσταλλο ρολογιού συχνότητας 25 MHz.

Το XUP Virtex-5 ML509 XC5VLX110T [32] έχει PHY που λειτουργεί σε ταχύτητες 10/100/1000 Mbps και υποστηρίζει τις διασυνδέσεις MII, GMII, RGMII και SGMII.

---

[1]

● Υποστηρίζεται.
○ Δεν υποστηρίζεται.
* Απαιτείται το 10-Gigabit fibre-optic Personality Module PM101 [39].
** Απαιτείται το Gigabit Ethernet Personality Module PM102 [39].

Κεφάλαιο 4

Αρχιτεκτονική Ολοκλήρωσης Συστήματος

Στο κεφάλαιο αυτό αναφέρονται συνοπτικά οι αρχικές μας προσπάθειες να εγκαθιδρύσουμε επικοινωνία στο επίπεδο ζεύξης, με χρήση του πρωτοκόλλου Ethernet, μεταξύ του Η/Υ και των αναπτυξιακών συστημάτων που είχαμε στη διάθεση μας μέχρι να φτάσουμε στο XUP Virtex-5 που χρησιμοποιήσαμε τελικά. Στη, συνέχεια, αναφέρεται η λειτουργικότητα που μας προσφέρουν ο πυρήνας της Xilinx και το λειτουργικό σύστημα του Η/Υ και τεκμηριώνονται οι σχεδιαστικές να χρησιμοποιηθούν τα πρωτόκολλα IPv4 και UDP. Τέλος, περιγράφονται οι δύο εκδόσεις του πρωτοκόλλου, επιπέδου εφαρμογής, MTP που αναπτύξαμε για τη μεταφορά δεδομένων από τον υπολογιστή στο αναπτυξιακό.

4.1 Ποια αναπτυξιακά συστήματα αφήσαμε

4.1.1 Virtex-2 Pro ML310 - XC2VP30

Προσέγγιση από πάνω προς τα κάτω

Η αρχική μας προσέγγιση ήταν να χρησιμοποιήσουμε τις βιβλιοθήκες δικτύων (Network Libraries) του λειτουργικού συστήματος για μικροελεγκτές uClinux [45] με σκοπό να αναπτύξουμε ένα πρωτόκολλο μεταφοράς δεδομένων στο επίπεδο εφαρμογής που θα υλοποιεί την επικοινωνία του υπολογιστή με την FPGA. Σαν βάση για τη σχεδίαση μας χρησιμοποιήσαμε το project MicroBlaze uClinux [46] του κατασκευαστή του αναπτυξιακού Digilent Inc. στο οποίο η έκδοση του πυρήνα (kernel) του uClinux δεν έχει ενσωματωμένες τις απαραίτητες βιβλιοθήκες. Η προσπάθεια μας δεν είχε επιτυχία επειδή είχαμε προβλήματα στη διαδικασία της μεταγλώττισης (cross-compile) του πυρήνα συμπεριλαμβάνοντας τις βιβλιοθήκες δικτύων.

Προσέγγιση από κάτω προς τα πάνω

Η επόμενη προσπάθεια μας ήταν να αποκαταστήσουμε επικοινωνία μεταξύ του υπολογιστή και της FPGA στο επίπεδο ζεύξης χρησιμοποιώντας έναν Ethernet_MAC ελεγκτή. Επειδή στις διάφορες εκδόσεις των εργαλείων της Xilinx (7.1i, 9.1i, 10.1) που είχαμε στη διάθεση μας η χρήση των IP που υλοποιούν Ethernet_MAC ελεγκτές απαιτούσε άδεια, την οποία δεν διαθέταμε, εξετάσαμε αν μπορούσε να χρησιμοποιηθεί κάποια σχεδίαση ελεγκτή ανοιχτού κώδικα (open source).

Μετά από σχετική έρευνα στο διαδίκτυο βρήκαμε δύο υποψήφιους ελεγκτές στην ιστοσελίδα της κοινότητας ανοιχτού κώδικα για υλικό (hardware) opencores.org. Η πρώτη σχεδίαση που δοκιμάσαμε ήταν η 10_100_1000 Mbps tri-mode ethernet MAC [47] και η δεύτερη η Ethernet MAC 10/100 Mbps [48]. Ατυχώς, καμία από τις δύο σχεδιάσεις δεν μπορούσε να περάσει τη διαδικασία της σύνθεσης (synthesis) με το εργαλείο Xilinx ISE Design Suite 10.1.

4.1.2 Virtex-4 ML410 - XC4VFX60

Μετά τις ανεπιτυχείς προσπάθειες με τη Virtex-2 Pro στραφήκαμε στο αναπτυξιακό σύστημα Virtex-4 ML410. Στην τεχνολογία αυτή τα μοντέλα FX έχουν τον Virtex-4 Embedded Tri-Mode Ethernet Media Access Controller (MAC) τον οποίο μπορούμε να ενεργοποιήσουμε και να διαμορφώσουμε με το IP Embedded Tri-mode Ethernet MAC Wrapper. Ατυχώς για εμάς στη συγκεκριμένη έκδοση των εργαλείων της Xilinx οι βιβλιοθήκες του ολοκληρωμένου XC4VFX60 είχαν κάποιο πρόβλημα –το οποίο διορθώθηκε αργότερα όταν είχαμε αλλάξει πια αναπτυξιακό- με αποτέλεσμα να μην είναι δυνατό να προγραμματίσουμε την παραπάνω FPGA.

4.1.3 Virtex-4 ML401 - XC4VLX25

Το επόμενο αναπτυξιακό σύστημα με το οποίο δουλέψαμε ήταν το Virtex-4 ML401. Με αυτό είχαμε την πρώτη επιτυχημένη προσπάθεια επικοινωνίας της FPGA με τον υπολογιστή. Επρόκειτο για το παράδειγμα σχεδίασης (application note) της Xilinx xapp433 [33] μία σχεδίαση ενός Web Server ο οποίος τρέχει στον προγραμματιζόμενο (Soft) επεξεργαστή MicroBlaze. Στη συνέχεια εξετάσαμε πως μπορούμε να αξιοποιήσουμε τη βιβλιοθήκη LightWeight IP (lwIP) [50], μια υλοποίηση ανοικτού κώδικα της στοίβας πρωτοκόλλων διαδικτύου για ενσωματωμένα συστήματα, προσπάθεια που διακόπηκε επειδή αλλάξαμε αναπτυξιακό και προχωρήσαμε σε τεχνολογία Virtex 5.

4.2 Το XUP Virtex-5 ML509 XC5VLX110T

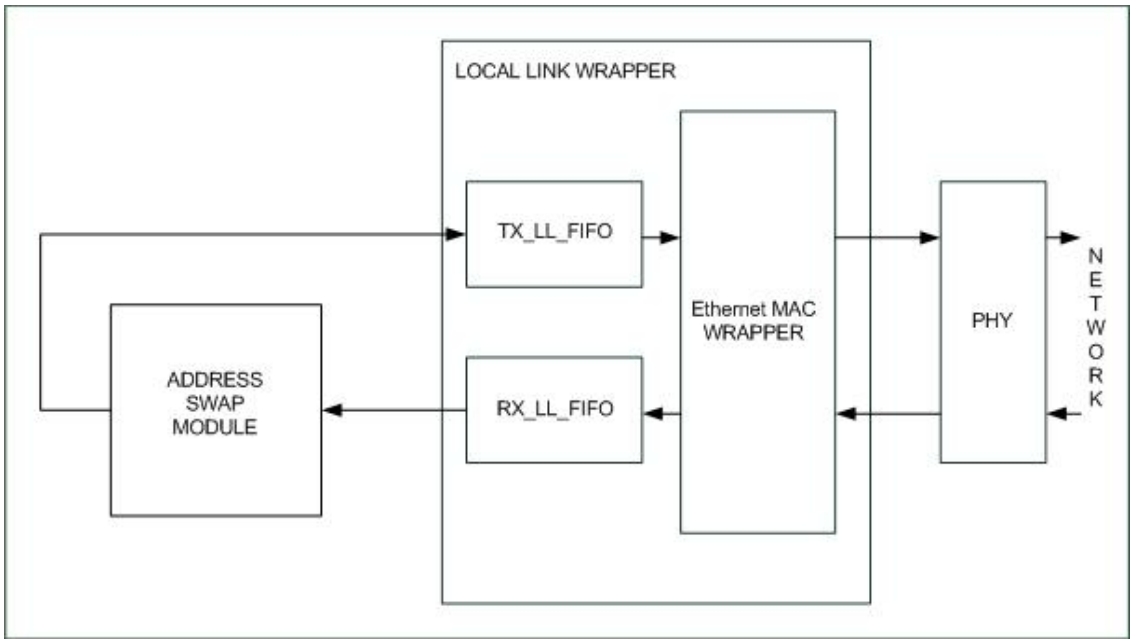
Το αναπτυξιακό σύστημα που αποτέλεσε τελικά την πλατφόρμα πάνω στην οποία υλοποιήσαμε, δοκιμάσαμε και επιβεβαιώσαμε τη σχεδίαση μας ήταν το XUP Virtex-5 ML509 XC5VLX110T. Η τεχνολογία Virtex-5 επιλέχθηκε επειδή διαθέτει χαρακτηριστικά χρήσιμα για τη γρήγορη ανάπτυξη σχεδιάσεων που απαιτούν σύνδεση με Ethernet. Επίσης η Xilinx έχει δημοσιεύσει μια σειρά από παραδείγματα σχεδιάσεων με σκοπό να διευκολύνει -σε σχέση με προγενέστερες

τεχνολογίες- τις προσπάθειες για την ανάπτυξη σχεδιάσεων που περιλαμβάνουν επικοινωνία μέσω Ethernet. Τέλος, στην επιλογή μας συνετέλεσε και το γεγονός ότι άλλες σχεδιάσεις που αναπτύσσονται από συναδέλφους στο εργαστήριο και χρησιμοποιούν αναπτυξιακά συστήματα της οικογένειας Virtex-5 πιθανώς να ενσωματώσουν στο μέλλον τη σχεδίαση μας για είσοδο/έξοδο (Input/Output).

4.2.1 Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC

Οι FPGA της οικογένειας Virtex-5 διαθέτουν τον V5 Embedded Tri-Mode Ethernet Media Access Controller (MAC) ο οποίος μας παρέχει την απαραίτητη λειτουργικότητα για να χειριστούμε το φυσικό και το επίπεδο ζεύξης μέχρι το υποεπίπεδο MAC για δύο συνδέσεις Ethernet οι οποίες είναι δυνατό να διαμορφωθούν ανεξάρτητα.

Για να ενεργοποιήσουμε και να διαμορφώσουμε τον Embedded TEMAC ελεγκτή χρησιμοποιήσαμε τον Embedded Tri-Mode Ethernet MAC Wrapper ο οποίος παράγει τα αρχεία VHDL που απαιτούνται καθώς και ένα παράδειγμα σχεδίασης (example design).



Σχήμα 5: **Το παράδειγμα σχεδίασης του Core Generator**

Το παράδειγμα που παράγει ο Core Generator (Σχήμα 5) αποτέλεσε τη βάση πάνω στην οποία χτίσαμε τη σχεδίαση μας. Το παράδειγμα αποτελείται από τρία επίπεδα:

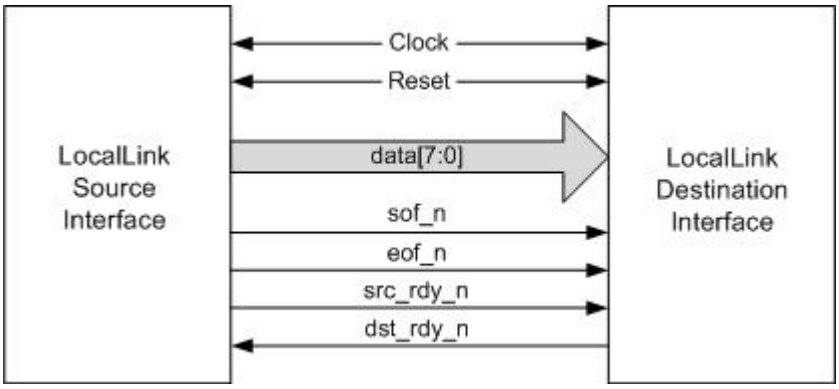
- Τον **block-level wrapper** που υλοποιεί (instantiates) τον Ethernet MAC wrapper και τη διεπαφή για κάθε επιλεγμένη διεπαφή φυσικού επιπέδου (Physical Layer Interface).

- Τον **LocalLink wrapper** που συνδέει τη διεπαφή του πομπού (Transmitter) και του δέκτη (Receiver), για κάθε επιλεγμένο Ethernet MAC ελεγκτή, σε μία LocalLink FIFO.

- Στο υψηλότερο ιεραρχικά επίπεδο (Top Level) το παράδειγμα συνδέει τις FIFO ώστε τα δεδομένα που λαμβάνονται στο δέκτη να κατευθύνονται στον πομπό. Μία μονάδα αλλαγής διεύθυνσης (Address Swap Module) ανταλλάσει τις MAC διευθύνσεις πηγής και προορισμού των εισερχομένων πλαισίων Ethernet.

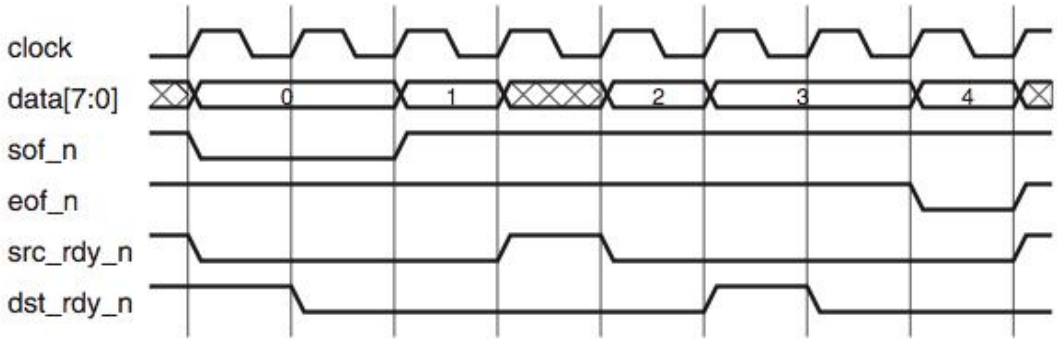
Η διεπαφή Local Link

Η διεπαφή Local Link [13] της Xilinx ορίζει ένα σύνολο των σημάτων (Σχήμα 4-2) που επιτρέπει τον έλεγχο της μεταφοράς πλαισίων Ethernet με ένα τρόπο ανεξάρτητο των ρυθμίσεων και των παραμέτρων του Ethernet MAC ελεγκτή.



Σχήμα 6: **Local Link Interface Block Diagram**

Για να έχουμε μεταφορά δεδομένων (Σχήμα 7) τα σήματα ελέγχου src_rdy_n (πηγή έτοιμη για αποστολή) και dst_rdy_n (προορισμός έτοιμος για λήψη) πρέπει να είναι ενεργοποιημένα ταυτόχρονα. Σε περίπτωση που κάποιο από τα δύο αυτά σήματα δεν είναι ενεργοποιημένο η μεταφορά δεδομένων αναστέλλεται (pause). Το σήμα sof_n ενεργοποιείται στην αρχή κάθε πλαισίου και αντίστοιχα το σήμα eof_n ενεργοποιείται στο τέλος κάθε πλαισίου.



Σχήμα 7: **Μεταφορά πλαισίου πάνω από διεπαφή Local Link** (Xilinx)

4.3 Αρχιτεκτονική πρωτοκόλλων της σχεδίασης μας

Σκοπός της σχεδίασης μας είναι η μεταφορά δεδομένων από τον υπολογιστή στην FPGA εκμεταλλευόμενοι την θύρα Ethernet του αναπτυξιακού μας.

Για τη φυσική σύνδεση του αναπτυξιακού με τον υπολογιστή είναι απαραίτητο ο δεύτερος να διαθέτει μία Gigabit Ethernet κάρτα δικτύου. Επιπλέον, χρειαζόμαστε ένα cross καλώδιο Ethernet το οποίο συνδέεται στις αντίστοιχες θύρες του αναπτυξιακού και της κάρτας δικτύου του υπολογιστή.

4.3.1 Διαμόρφωση του Tri-Mode Ethernet MAC

Ο Embedded Tri-Mode Ethernet MAC (TEMAC) ελεγκτής που χρησιμοποιείται στη σχεδίαση μας έχει διαμορφωθεί να αξιοποιεί τη μία από τις δύο συνδέσεις Ethernet που είναι διαθέσιμες. Η ταχύτητα έχει ρυθμιστεί στο 1 Gbps (Gigabit Ethernet), η επιλογή αύτη έγινε επειδή η επιθυμητή απόδοση της σύνδεσης FPGA-υπολογιστή που σχεδιάσαμε δεν ήταν ικανοποιητική με ταχύτητες 100 Mbps (Fast Ethernet) ή 10 Mbps (Ethernet). Η διεπαφή φυσικού επιπέδου με το επίπεδο ζεύξης δεδομένων που χρησιμοποιήθηκε είναι η Serial Gigabit Media Independent Interface (SGMII) επειδή χρησιμοποιεί λιγότερα pin (8) για είσοδο/έξοδο σε σχέση με άλλες επιλογές που δοκιμάσαμε. Ο πομπός και ο δέκτης του ελεγκτή έχουν ρυθμιστεί να λειτουργού πλήρως αμφίδρομα (full-duplex) και έχει ενεργοποιηθεί η λειτουργία autonegotiation για την αποκατάσταση επικοινωνίας μεταξύ αναπτυξιακού και υπολογιστή στο φυσικό επίπεδο. Τέλος, ρυθμίσαμε τον ελεγκτή ώστε να ελέγχει το πεδίο CRC του πλαισίου Ethernet και να απορρίπτει τα πλαίσια που έχουν σφάλματα.

4.3.2 Επιλογή πρωτοκόλλων για τα επίπεδα μεταφοράς και δικτύου

Από την πλευρά του υπολογιστή, εκμεταλλευόμενοι τις βιβλιοθήκες δικτύου του λειτουργικού συστήματος έχουμε τη δυνατότητα να αναπτύξουμε την εφαρμογή, η

οποία παράγει τα προς αποστολή πακέτα, στο επίπεδο εφαρμογής. Με δεδομένη τη λειτουργικότητα που μας παρέχει ο Ethernet ελεγκτής της Xilinx, η σχεδίαση μας θα πρέπει να χειρίζεται τα επίπεδα μεταφοράς και δικτύου.

Με στόχο να απλοποιήσουμε τη σχεδίαση μας, επιλέξαμε να υλοποιήσουμε σε υλικό μόνο την απαραίτητη λειτουργικότητα των επιπέδων μεταφοράς και δικτύου για την μεταφορά πακέτων από τον υπολογιστή στο αναπτυξιακό και αντίστροφα.

Η παραπάνω προσέγγιση μας οδήγησε στο να επιλέξουμε το UDP ως πρωτόκολλο του επιπέδου μεταφοράς αντί για το πρωτόκολλο TCP. Η επιλογή αυτή έγινε επειδή το UDP είναι πολύ πιο απλό πρωτόκολλο σε σχέση με το TCP. Το δεύτερο υλοποιεί αξιόπιστη αποστολή και λήψη δεδομένων με επιβεβαιώσεις (Acknowledgment), εξασφαλίζει τη μεταφορά των δεδομένων χωρίς λάθη μεταξύ του επιπέδου δικτύου και του επιπέδου εφαρμογής και τέλος, εγγυάται την παράδοση των δεδομένων στο επίπεδο εφαρμογής στη σωστή σειρά. Αντίθετα, το πρωτόκολλο UDP δεν εγγυάται αξιόπιστη μεταφορά δεδομένων ούτε και την παράδοση τους στη σωστή σειρά. Στο σύστημα μας η σύνδεση σημείο-προς-σημείο (point to point) του υπολογιστή με την FPGA μας εξασφαλίζει την εν σειρά μεταφορά δεδομένων. Η αξιόπιστη μεταφορά δεδομένων και η αποστολή επιβεβαιώσεων ανατίθεται στο πρωτόκολλο που σχεδιάσαμε για το επίπεδο εφαρμογής.

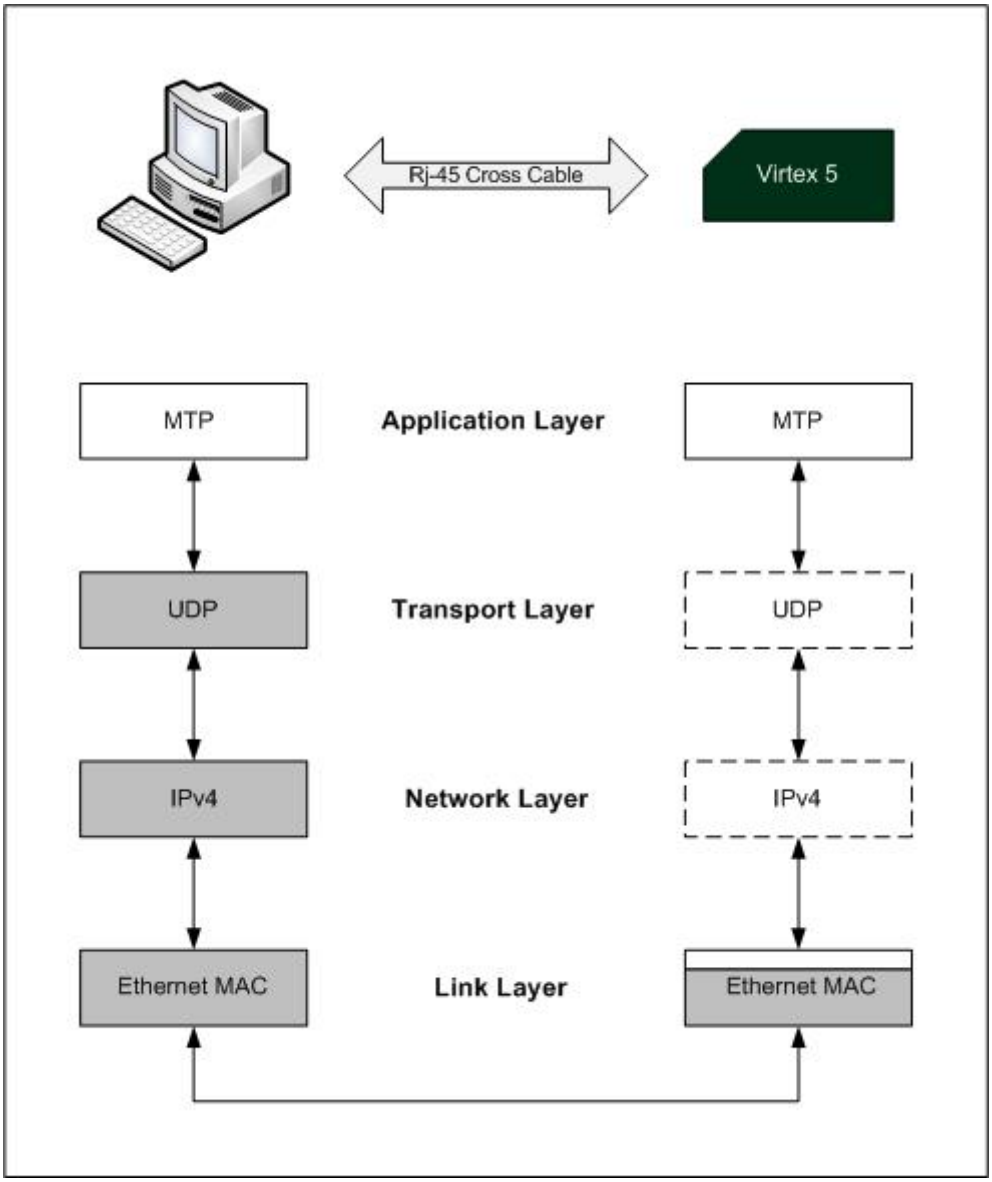
Στο επίπεδο δικτύου επιλέξαμε το πρωτόκολλο IPv4, το οποίο είναι η έκδοση του IP που χρησιμοποιείται περισσότερο σήμερα. Η σχεδίαση μας δεν υλοποιεί τη λειτουργία κατακερματισμού του IPv4 επομένως το πακέτο IPv4 πρέπει να μπορεί να ενθυλακωθεί σε ένα πλαίσιο Ethernet (μέγεθος πακέτου IPv4<1500 bytes). Επιπλέον, η σχεδίαση μας υποστηρίζει μέγεθος κεφαλίδας του IPv4 20 bytes (δεν επιτρέπονται τα πεδία επιλογών).

Τέλος, η σύνδεση σημείο-προς-σημείο (point to point) του υπολογιστή με το αναπτυξιακό μαζί με τον περιορισμό ως προς το μέγεθος των πακέτων IPv4 μας επιτρέπουν να χρησιμοποιήσουμε μόνο τον έλεγχο σφαλμάτων του Ethernet για όλα τα επίπεδα.

Ρυθμίσεις δικτύου του λειτουργικού συστήματος

Για να καταστεί δυνατή η επικοινωνία του υπολογιστή με το αναπτυξιακό μας πρέπει να έχουν διευθύνσεις IP οι οποίες να είναι στο ίδιο υποδίκτυο (subnet) –

στα πειράματα μας χρησιμοποιήσαμε την διεύθυνση 1.2.3.5 για την FPGA, τη διεύθυνση 1.2.3.9 και subnet mask 255.255.255.0 για τον υπολογιστή. Επιπλέον, η ταχύτητα και ο τύπος ζεύξης (full ή half duplex) πρέπει να έχουν ρυθμιστεί στην επιλογή auto και έχει επιλεχθεί η χρήση του πρωτοκόλλου IPv4. Τέλος, επειδή η σχεδίαση μας δεν υποστηρίζει το πρωτόκολλο ARP (Address Resolution Protocol) [47] πρέπει να δηλώσουμε στον πίνακα APR τη συσχέτιση της διεύθυνσης IP της FPGA με την διεύθυνση της MAC. Αυτό γίνετε με την εντολή arp -s hostname ether_addr (ίδια σύνταξη σε περιβάλλον MS-DOS και UNIX).



Σχήμα 8: **Η αρχιτεκτονική πρωτοκόλλων της σχεδίασης μας**

Αποστολή Πακέτων από τον υπολογιστή στην FPGA

Για την αποστολή δεδομένων από τον υπολογιστή στην FPGA, το πρόγραμμα μας που υλοποιεί το πρωτόκολλο του επιπέδου εφαρμογής δημιουργεί ένα UDP socket (τύπου client) και στέλνει πακέτα UDP στη διεύθυνση IP του αναπτυξιακού μας και στο port που ορίζεται από το πρωτόκολλο μας (στα πειράματα μας χρησιμοποιήσαμε το port 5000). Από την πλευρά της FPGA η σχεδίαση μας λαμβάνει το πλαίσιο Ethernet, που περιέχει το δεδομενόγραμμα που στείλαμε, και εξετάζει συγκεκριμένα πεδία των κεφαλίδων (MAC διεύθυνση προορισμού, IP διεύθυνση προορισμού, port προορισμού) για να διαλέξει αν θα απορρίψει το πακέτο ή θα το αξιοποιήσει.

Αποστολή Πακέτων από την FPGA στον υπολογιστή

Για να λάβει πακέτα από την FPGA το πρόγραμμα που υλοποιεί το επίπεδο εφαρμογής δημιουργεί ένα δεύτερο UDP socket (τύπου server) το οποίο περιμένει να λάβει πακέτα από τη διεύθυνση IP του αναπτυξιακού στο port που ορίζεται από το πρωτόκολλο μας (στα πειράματα μας χρησιμοποιήσαμε το port 5005). Στην FPGA, η σχεδίαση μας δημιουργεί πλαίσια Ethernet σταθερού μήκους τα οποία έχουν ενθυλακωμένα πακέτα IPv4, UDP και το πακέτο του πρωτοκόλλου εφαρμογής. Τα πεδία των κεφαλίδων IPv4 και UDP παίρνουν σταθερές τιμές εκτός από τα πεδία ταυτότητας και αθροίσματος ελέγχου κεφαλίδας του πακέτου IPv4 καθώς και το πεδίο αθροίσματος ελέγχου του δεδομενογράμματος UDP τα οποία υπολογίζονται.

Το Σχήμα 8 περιγράφει την αρχιτεκτονική πρωτοκόλλων της σχεδίασης μας. Με γκρι χρώμα παρουσιάζονται οι λειτουργικότητες που μας προσφέρουν το λειτουργικό σύστημα του υπολογιστή και ο Ethernet ελεγκτής τη Xilinx. Με διακεκομμένες γραμμές παρουσιάζονται τα πρωτόκολλα IPv4 και UDP τα οποία δεν υλοποιούνται ολόκληρα στην FPGA.

4.4 Το πρωτόκολλο επιπέδου εφαρμογής MHL Transfer Protocol

Για την υλοποίηση της μεταφοράς δεδομένων από τον υπολογιστή στο αναπτυξιακό Virtex-5 ML509 αναπτύξαμε το πρωτόκολλο επιπέδου εφαρμογής MTP (MHL Transfer Protocol). Το MTP εκμεταλλεύεται τη λειτουργικότητα του επίπεδου μεταφοράς που μας προσφέρει το πρωτόκολλο UDP όπως περιγράφηκε στην

προηγούμενη ενότητα, επομένως πρέπει να καθορίζει μηχανισμούς αξιόπιστης μεταφοράς δεδομένων και εγκαθίδρυσης συνόδου (Session).

Δομή πακέτου MTP

Τα πακέτα του πρωτοκόλλου MTP διακρίνονται σε πακέτα ελέγχου (control packets) και πακέτα δεδομένων (data packets). Το Σχήμα 9 δείχνει τη δομή του πακέτου. Τα προς μεταφορά δεδομένα καταλαμβάνουν το πεδίο Data το οποίο μπορεί να έχει μέγεθος ως 1467 bytes ώστε μπορεί το δεδομενόγραμμα να ενθυλακωθεί σε ένα μόνο πλαίσιο Ethernet. Το πεδίο Data υπάρχει μόνο στα πακέτα δεδομένων. Το πεδίο Type καθορίζει τον τύπο και το πεδίο Number καθορίζει τον αριθμό του πακέτου (unsigned integer - 2^{32} πακέτα).

Σχήμα 9: **Δομή πακέτου MTP**

4.4.1 Η έκδοση MTP1.0

Η έκδοση 1.0 του πρωτοκόλλου MTP δεν εξασφαλίζει αξιόπιστη μεταφορά δεδομένων. Το πρόγραμμα που υλοποιεί το πρωτόκολλο χρησιμοποιεί ένα UDP socket για αν μεταδώσει αριθμημένα πακέτα και η σχεδίαση μας, στην πλευρά του αναπτυξιακού, τα λαμβάνει και ελέγχει αν έχουμε απώλεια δεδομένων. Στην τελευταία περίπτωση η μεταφορά τερματίζεται.

Τα πακέτα του MTP1.0

- Πακέτο έναρξης συνόδου (Start of Session). Το πεδίο Type παίρνει την τιμή S (0x53) και το πεδίο Number παίρνει τιμή 0.

- Πακέτο δεδομένων (Data Packet). Το πεδίο Type παίρνει την τιμή D (0x44) και στο πεδίο Number ανατίθεται ο αριθμός του πακέτου (0 ως 2^{32}). Το πεδίο Data έχει μπορεί να έχει μέγεθος 1467 bytes έτσι ώστε το δεδομενόγραμμα να μπορεί να ενθυλακωθεί σε ένα πλαίσιο Ethernet.

- Πακέτο λήξης συνόδου (End Session). Το πεδίο Type παίρνει τιμή E (0x45) και το πεδίο Number παίρνει τιμή του τελευταίου πακέτου δεδομένων αυξημένη κατά ένα.

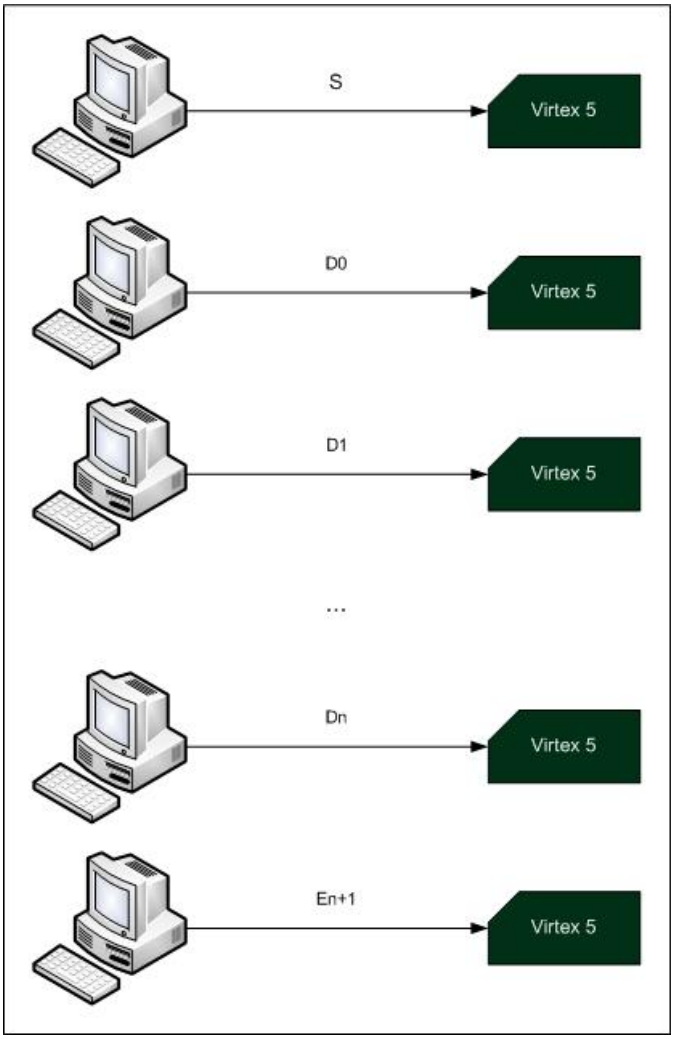
Σύνοδος MTP1.0

Στο Σχήμα 10 παρουσιάζεται ένα παράδειγμα συνόδου του πρωτοκόλλου MTP1.0:

1. Αρχικά, ο υπολογιστής στέλνει στο αναπτυξιακό ένα πακέτο έναρξής συνόδου. Όταν η σχεδίαση μας, στη FPGA, λάβει το πακέτο αρχικοποιεί (reset) τους μετρητές και τον καταχωρητή κατάστασης (status register) του δέκτη.

2. Στη συνέχεια, ο υπολογιστής στέλνει αριθμημένα πακέτα δεδομένων. Η σχεδίαση μας λαμβάνει τα πακέτα και ελέγχει αν καταφθάνουν με τη σειρά. Σε περίπτωση που χαθεί κάποιο πακέτο η σχεδίαση μας σταματά να δίνει δεδομένα στη σχεδίαση του χρήστη δίνει στο bit λάθους του καταχωρητή κατάστασης την τιμή 1 και περιμένει το επόμενο πακέτο έναρξης συνόδου.

3. Στο τέλος της συνόδου ο υπολογιστής στέλνει ένα πακέτο λήξης συνόδου και κλείνει το UDP socket. Η σχεδίαση μας όταν λάβει το πακέτο ελέγχει το πεδίο Number και αν δεν έχουμε απώλεια δεδομένων δίνει στο bit επιτυχούς ολοκλήρωσης του καταχωρητή κατάστασης την τιμή 1.



Σχήμα 10: **Παράδειγμα συνόδου MTP1.0**

Απαραίτητη προϋπόθεση για να χρησιμοποιήσει μια σχεδίαση το MTP1.0 είναι αυτή να μπορεί να απορροφήσει τα δεδομένα με ρυθμό μεγαλύτερο ή ίσο από τον ρυθμό που της τα παρέχει η σχεδίαση μας. Σε αντίθετη περίπτωση οι FIFO που χρησιμοποιούνται στη σχεδίαση μας θα υπερχειλίσουν με αποτέλεσμα το σύστημα να καθίσταται ασταθές.

4.4.2 Η έκδοση MTP2.0

Η έκδοση 2.0 του πρωτοκόλλου MTP υποστηρίζει την αξιόπιστη μεταφορά δεδομένων. Το πρόγραμμα που υλοποιεί το πρωτόκολλο δημιουργεί δύο νήματα (threads). Το πρώτο μέσω ενός UDP socket στέλνει αριθμημένα πακέτα σε παράθυρα. Το δεύτερο περιμένει πακέτα επιβεβαίωσης από την FPGA μέσω ενός δεύτερου UDP socket. Από την πλευρά της FPGA η σχεδίαση μας στέλνει πακέτα επιβεβαίωσης για κάθε παράθυρο πακέτων δεδομένων που λαμβάνει.

Τα πακέτα του MTP2.0

Τα πακέτα που στέλνονται από τον υπολογιστή στην FPGA είναι:

- Πακέτο έναρξης συνόδου (Start of Session). Το πεδίο Type παίρνει την τιμή S (0x53) και το πεδίο Number παίρνει τιμή 0.
- Πακέτο δεδομένων (Data Packet). Το πεδίο Type παίρνει την τιμή D (0x44) και στο πεδίο Number ανατίθεται ο αριθμός του πακέτου (0 ως 2^{32}-1). Το πεδίο Data έχει μπορεί να έχει μέγεθος 1467 bytes έτσι ώστε το δεδομενόγραμμα να μπορεί να ενθυλακωθεί σε ένα πλαίσιο Ethernet.
- Πακέτο λήξης συνόδου (End Session). Το πεδίο Type παίρνει τιμή E (0x45) και το πεδίο Number παίρνει τιμή του τελευταίου πακέτου δεδομένων αυξημένη κατά ένα.

Τα πακέτα που στέλνονται από την FPGA στον υπολογιστή είναι:

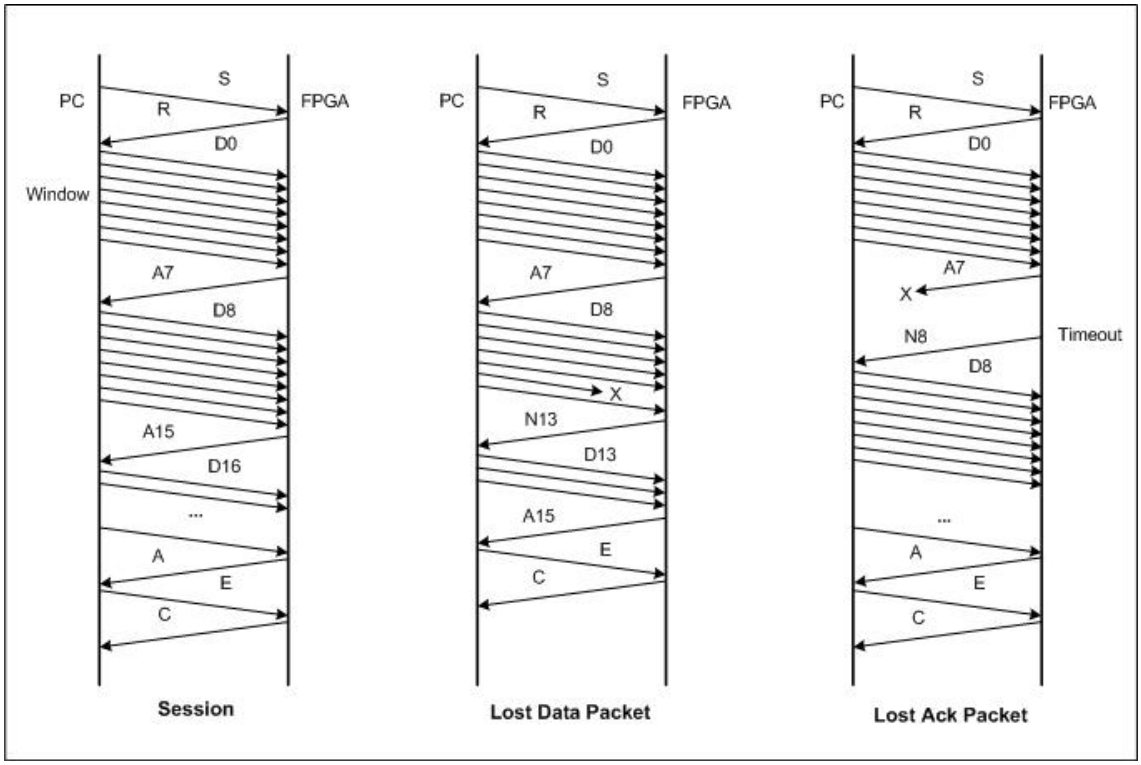
- Πακέτο αναμονής λήψης πακέτων δεδομένων (Ready to receive). Το πεδίο Type παίρνει την τιμή R (0x52) και το πεδίο Number παίρνει τιμή 0.
- Πακέτο θετικής επιβεβαίωσης (Acknowledgment). Το πεδίο Type παίρνει την τιμή A (0x41) και το πεδίο Number παίρνει ως τιμή τον αριθμό του πακέτου που παραλήφτηκε τελευταίο.

- Πακέτο αρνητικής επιβεβαίωσης (Negative Acknowledgment). Το πεδίο Type παίρνει την τιμή N (0x4E) και το πεδίο Number παίρνει ως τιμή τον αριθμό του πακέτου που αναμένεται.

- Πακέτο επιβεβαίωσής λήξης συνόδου (Close socket). Το πεδίο Type παίρνει την τιμή C (0x43) και το πεδίο Number παίρνει τιμή 0.

Σύνοδος MTP2.0

Στο Σχήμα 11 παρουσιάζεται ένα παράδειγμα συνόδου του πρωτοκόλλου MTP2.0



Σχήμα 11: **Παράδειγμα συνόδου MTP2.0**

1. Αρχικά ο υπολογιστής στέλνει στην FPGA ένα πακέτο έναρξης συνόδου. Όταν η σχεδίαση μας λάβει το πακέτο αρχικοποιεί τους μετρητές και τους καταχωρητές του πομπού και του δέκτη.

2. Στη συνέχεια η σχεδίαση μας στέλνει στον υπολογιστή ένα πακέτο αναμονής λήψης πακέτων δεδομένων ολοκληρώνοντας τη διαδικασία της χειραψίας για την έναρξη της αποστολής πακέτων δεδομένων.

3. Ο υπολογιστής στέλνει παράθυρα από πακέτα δεδομένων και η σχεδίαση μας τα λαμβάνει, τα ελέγχει και στέλνει στον υπολογιστή πακέτα επιβεβαίωσης για κάθε παράθυρο. Στην περίπτωση που χαθεί κάποιο

πακέτο δεδομένων τότε η σχεδίαση μας στέλνει ένα πακέτο αρνητικής επιβεβαίωσης. Τέλος αν χαθεί κάποιο πακέτο επιβεβαίωσης η σχεδίαση μας στέλνει ένα πακέτο αρνητικής επιβεβαίωσης μετά από ένα ορισμένο χρονικό διάστημα (Timeout).

4. Όταν ολοκληρωθεί η αποστολή των πακέτων δεδομένων ο υπολογιστής στέλνει ένα πακέτο λήξης συνόδου. Η σχεδίαση μας όταν λάβει το πακέτο ελέγχει το πεδίο Number και αν έχουμε απώλεια δεδομένων στέλνει ένα πακέτο αρνητικής επιβεβαίωσης.

5. Αν η σχεδίαση μας λάβει πακέτο λήξης συνόδου χωρίς να έχουμε απώλεια δεδομένων στέλνει ένα πακέτο επιβεβαίωσης λήξης συνόδου. Όταν το πρόγραμμα, που υλοποιεί το πρωτόκολλο στον υπολογιστή, λάβει την επιβεβαίωση λήξης συνόδου κλείνει τα δύο UDP socket.

Κεφάλαιο 5

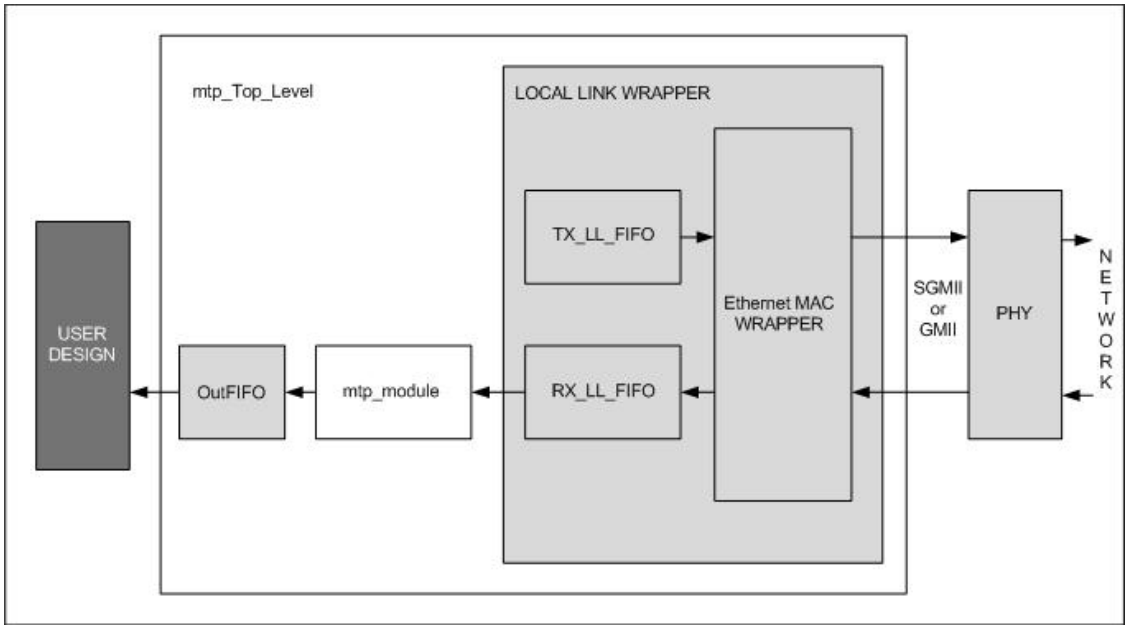
Σχεδίαση, Υλοποίηση και Επιβεβαίωση Λειτουργίας

Στο κεφάλαιο αυτό παρουσιάζεται το κομμάτι εκείνο της σχεδίασης μας που αποτυπώθηκε σε υλικό και αναλύονται οι επιμέρους λειτουργικές μονάδες και λεπτομέρειες της υλοποίησης. Επιπλέον, περιγράφεται η διαδικασία που ακολουθήθηκε για την επιβεβαίωση της σωστής λειτουργίας της,

5.1 Σχεδίαση

Στην ενότητα αυτή περιγράφεται το κομμάτι της σχεδίασης μας που υλοποιεί την αρχιτεκτονική πρωτοκόλλων (Σχήμα 8) του συστήματος μας σε υλικό.

5.1.1 Το MTP1.0



Σχήμα 12: **MTP1.0 Block Diagram**

Το Σχήμα 12 δείχνει μία αφηρημένη άποψη της υλοποίησης της σχεδίασης μας η οποία υποστηρίζει την έκδοση 1.0 του πρωτοκόλλου MTP. Σε αυτό απεικονίζονται οι κύριες λειτουργικές μονάδες (modules) καθώς και οι κύριες συνδέσεις μεταξύ τους. Με γρι παρουσιάζονται οι λειτουργικές μονάδες που δημιουργήσαμε με το εργαλείο Xilinx Core Generator 10.1 και με άσπρο αυτές για της οποίες γράψαμε κώδικα VHDL. Η αναπαράσταση αυτή ισχύει για όλα τα block διαγράμματα μέχρι το τέλος του κεφαλαίου.

Τα Ethernet πλαίσια που στέλνει ο υπολογιστής λαμβάνονται από τον ενσωματωμένο Ethernet MAC ελεγκτή της Xilinx, ο οποίος έχει διαμορφωθεί όπως περιγράφηκε στη παράγραφο 4.3.1. Τα πλαίσια γράφονται στη Local Link FIFO του δέκτη (RX_LL_FIFO) η οποία δεν τα καθιστά ορατά στην έξοδο της μέχρι να ολοκληρωθεί ο υπολογισμός του πεδίου CRC από τον Ethernet MAC ελεγκτή. Αν εντοπιστεί κάποιο σφάλμα σε ένα πλαίσιο τότε αυτό αφαιρείται από τη RX_LL_FIFO. Στη συνέχεια η λειτουργική μονάδα mtp_module διαβάζει τα Ethernet πλαίσια, μέσω της διεπαφής Local Link, ελέγχει αν αυτά έχουν ενθυλακωμένα πακέτα MTP. Αν τα πλαίσια περιέχουν ένα έγκυρο πακέτο δεδομένων του πρωτοκόλλου MTP το mtp_module γράφει το πεδίο δεδομένων του πακέτου στην FIFO του χρήστη (OutFIFO). Τέλος, η σχεδίαση του χρήστη μπορεί να διαβάσει τα δεδομένα που μεταφέρθηκαν μέσω της διεπαφής ανάγνωσης της OutFIFO (Πίνακας 2).

Σήμα	**Τύπος**	**Εύρος (bits)**	**Περιγραφή**
mtp_rd_en_in	είσοδος	1	Αν η OutFIFO περιέχει δεδομένα ενεργοποίηση αυτού του σήματος προκαλεί την ανάγνωση δεδομένων
mtp_data_out	έξοδος	8	Η θύρα δεδομένων
mtp_empty_out	έξοδος	1	Αυτό σήμα δηλώνει αν η OutFIFO είναι άδεια
mtp_full_out	έξοδος	1	Αυτό σήμα δηλώνει αν η OutFIFO είναι πλήρης
mtp_overflow_out	έξοδος	1	Αυτό το σήμα δηλώνει ότι υπήρξε μια αποτυχημένη εγγραφή
mtp_valid_out	έξοδος	1	Αυτό το σήμα δηλώνει αν διαβάζονται έγκυρα δεδομένα

Πίνακας 2: **Η διεπαφή ανάγνωσης της OutFIFO**

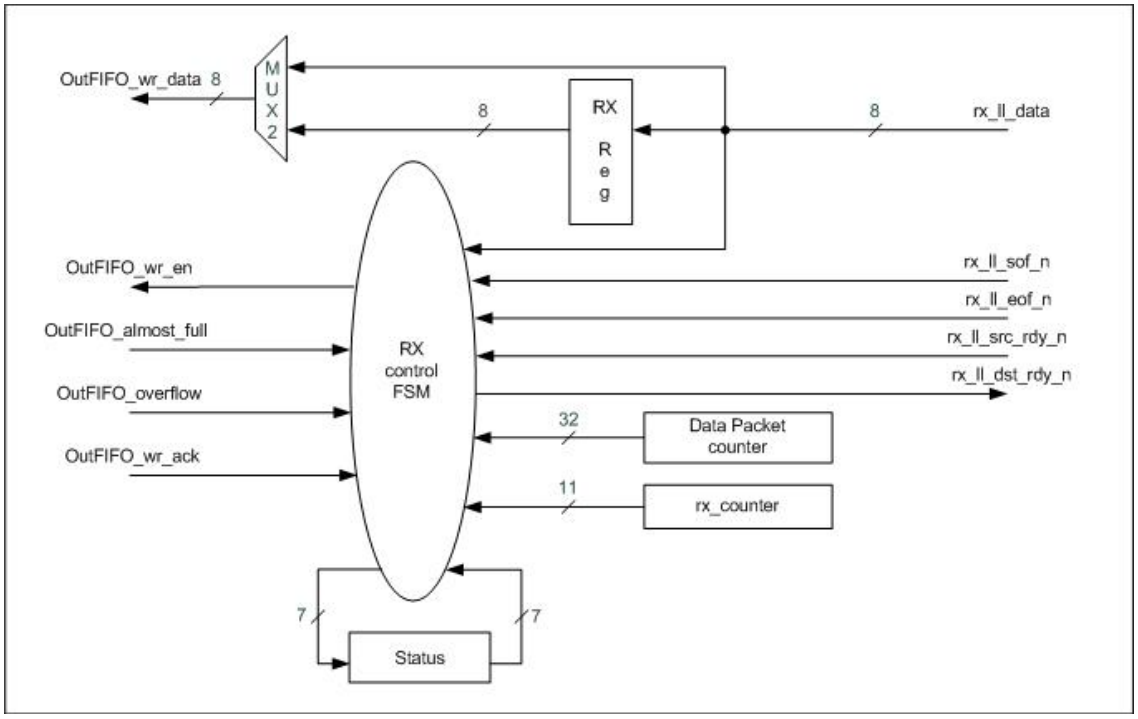
Το mtp_module

Η λειτουργική μονάδα mtp_module είναι το κομμάτι της σχεδίασης μας το οποίο υλοποιεί σε υλικό την απαραίτητη λειτουργικότητα των επιπέδων δικτύου (πρωτόκολλο IPv4) και μεταφοράς (πρωτόκολλο UDP) καθώς και το πρωτόκολλο MTP1.0 του επιπέδου εφαρμογής.

Το Σχήμα 13 δείχνει την βασική υλοποίηση του mtp_module. Στο σχήμα παρουσιάζονται οι λειτουργικές μονάδες και οι κύριες συνδέσεις τους. Παρουσιάζονται επίσης η διεπαφή (Local Link) του mtp_module με την RX_LL_FIFO από όπου διαβάζει τα εισερχόμενα πλαίσια Ethernet καθώς και η

διεπαφή με την OutFIFO όπου γράφει τα δεδομένα που προορίζονται για τη σχεδίαση του χρήστη.

Η διαδρομή δεδομένων (datapath) του mtp_module αποτελείται από έναν καταχωρητή εύρους 8 bit (**RX Reg**) που κρατά τα δεδομένα εσόδου για ένα κύκλο, για την περίπτωση που δεν είναι δυνατή η εγγραφή στην OutFIFO, και έναν πολυπλέκτη 2 σε 1 που με τον οποίο επιλέγουμε αν θα δώσουμε στην έξοδο τα δεδομένα που μας ήρθαν σε αυτόν ή τον προηγούμενο κύκλο.



Σχήμα 13: **Το mtp_module**

Το mtp_module διαθέτει ένα μετρητή εύρους 11 bit (**rx_counter**) που μετρά τον αριθμό των bytes του εισερχόμενου πλαισίου Ethernet (ως 1514 bytes). Με βάση την τιμή του rx_counter διακρίνουμε τα πεδία των πακέτων που ενθυλακώνονται σε κάθε πλαίσιο. Διαθέτει επίσης, έναν δεύτερο μετρητή εύρους 32 bit (**Data Packet counter**) ο οποίος προσδιορίζει τον αριθμό (πεδίο Number) του επόμενου MTP πακέτου δεδομένων που αναμένεται. Εκτός από τους δύο παραπάνω μετρητές

υπάρχει και ένας καταχωρητής κατάστασης (**Status Register**), εύρους 7 bit, που κρατά πληροφορία για την κατάσταση (status) της συνόδου MTP (Πίνακας 3).

Bit	Κατάσταση
0	Σύνοδος σε εξέλιξη
1	Επιτυχής ολοκλήρωση συνόδου
2	Λήψη πακέτου δεδομένων εκτός συνόδου
3	Λήψη πακέτου δεδομένων εκτός σειράς
4	Λήψη πακέτου λήξης συνόδου εκτός σειράς
5	Λήψη μη έγκυρου MTP πακέτου
6	Υπερχείλιση της OutFIFO

Πίνακας 3: **Περιεχόμενα καταχωρητή κατάστασης**

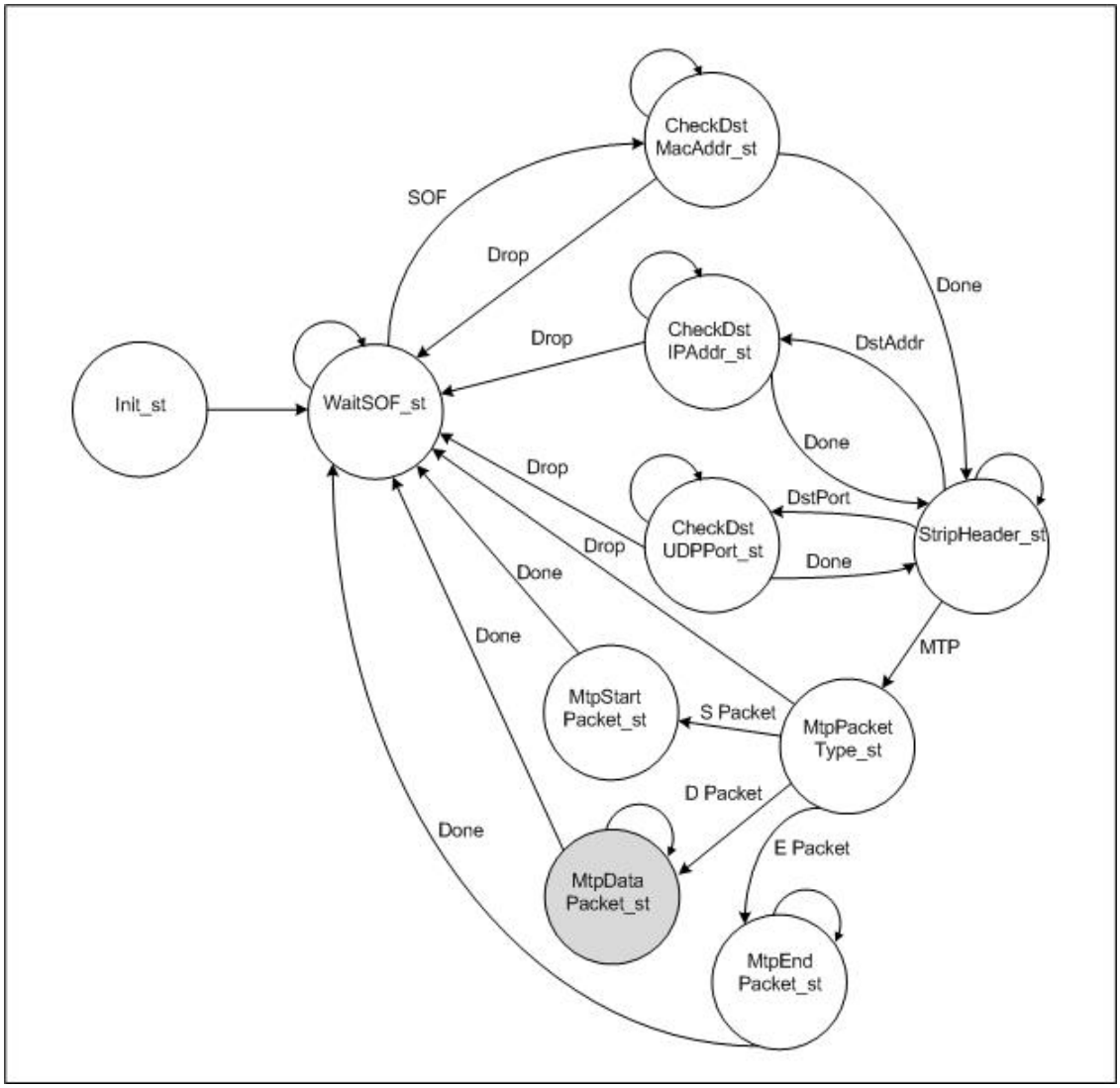
Τέλος, το mtp_module διαθέτει μια μονάδα ελέγχου, την **RX control FSM**, η οποία υλοποιείται σαν μια μηχανή πεπερασμένων καταστάσεων τύπου mealy και παράγει τα σήματα για τον έλεγχο των λειτουργικών μονάδων και των διεπαφών για την ανάγνωση των πλαισίων Ethernet από την RX_LL_FIFO και την εγγραφή των δεδομένων στην OutFIFO.

Το Σχήμα 14 δείχνει το διάγραμμα καταστάσεων της μονάδας ελέγχου (RX control FSM) του mtp_module.

Στην κατάσταση **Init_st**, η ποία είναι η αρχική κατάσταση της μηχανής, αρχικοποιούνται (reset) οι καταχωρητές και οι μετρητές του mtp_module.

Από την κατάσταση Init_st η μηχανή μας μεταβαίνει στην κατάσταση **WaitSOF_st** και ενεργοποιεί το σήμα rx_ll_dst_rdy_n για την ανάγνωση της RX_LL_FIFO. Όταν διαβαστεί το πρώτο byte ενός πλαισίου Ethernet (rx_ll_sof_n='0') ενεργοποιείται ο μετρητής rx_counter και η μηχανή μεταβαίνει στην κατάσταση CheckDstMacAddr_st.

Στην κατάσταση **CheckDstMacAddr_st** ελέγχεται η MAC διεύθυνση προορισμού του εισερχόμενου πλαισίου Ethernet. Αν αυτή δεν είναι η διεύθυνση που έχουμε ορίσει για το αναπτυξιακό μας (στα πειράματα μας χρησιμοποιήσαμε την τιμή 66:66:66:66:66:66) το πλαίσιο απορρίπτεται και η μηχανή μεταβαίνει πίσω στην κατάσταση WaitSOF_st αναμένοντας το επόμενο πλαίσιο, διαφορετικά η μηχανή μεταβαίνει στην κατάσταση **StripHeader_st**, η οποία αφαιρεί τα πεδία των κεφαλίδων του Ethernet και του IPv4 που δεν ελέγχουμε.



Σχήμα 14: **Η μονάδα ελέγχου RX control FSM**

Όταν φτάσει στο πεδίο της διεύθυνσης προορισμού της κεφαλίδας IPv4 η μηχανή μεταβαίνει στην κατάσταση **CheckDstIPAddr_st** και εξετάζει την IP διεύθυνση προορισμού. Όπως και προηγουμένως, αν η διεύθυνση προορισμού του πακέτου IPv4 είναι αυτή που έχουμε ορίσει για το αναπτυξιακό μας (στα πειράματα μας χρησιμοποιήσαμε την IP 1.2.3.5) η μηχανή μεταβαίνει πάλι στην κατάσταση StripHeader_st για να αφαιρέσει τα πεδία της κεφαλίδας UDP που δεν ελέγχουμε. Στην περίπτωση που το πακέτο έχει διαφορετική IP προορισμού από αυτή του αναπτυξιακού μας απορρίπτεται και η μηχανή μεταβαίνει στην κατάσταση WaitSOF_st.

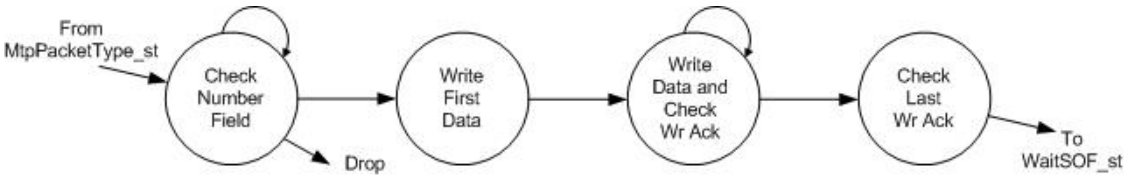
Η ίδια διαδικασία ακολουθείται και για τον έλεγχο του UDP Port προορισμού. Η μηχανή μεταβαίνει στην κατάσταση **CheckDstUDPPort_st**, και ελέγχει το port προορισμού της κεφαλίδας UDP. Αν αυτό είναι ίδιο με το port της σχεδίασης μας (στα περάματα μας χρησιμοποιήσαμε το Port 5000) η μηχανή μεταβαίνει στην κατάσταση StripHeader_st για να αφαιρέσει τα υπόλοιπα πεδία της κεφαλίδας UDP, διαφορετικά το πακέτο απορρίπτεται και η μηχανή μεταβαίνει στην κατάσταση WaitSOF_st.

Όταν φτάσει στο πεδίο δεδομένων του πακέτου UDP η μηχανή μεταβαίνει στην κατάσταση **MtpPacketType_st** η οποία ελέγχει το πεδίο Type του πακέτου MTP. Ανάλογα με την τιμή του πεδίου Type η μηχανή μεταβαίνει είτε στην κατάσταση MtpStartPacket_st αν έχουμε πακέτο έναρξης συνόδου είτε στην κατάσταση MtpDataPacket_st αν έχουμε πακέτο δεδομένων είτε στην κατάσταση MtpEndPacket_st αν έχουμε πακέτο λήξης συνόδου. Τέλος, αν η τιμή του πεδίου Type δεν προοδιορίζει κάποια από τις παραπάνω περιπτώσεις το πακέτο απορρίπτεται και η μηχανή μεταβαίνει στην κατάσταση WaitSOF_st.

Στην κατάσταση **MtpStartPacket_st** αρχικοποιούνται (reset) ο μετρητής rx_counter και ο μετρητής Data Packet counter και ενεργοποιείτε στον καταχωρητή κατάστασης το bit ενεργής συνόδου (τιμή “0000001” - Σύνοδος σε εξέλιξη χωρίς λάθη). Στη συνέχεια η μηχανή μεταβαίνει στην κατάσταση WaitSOF_st και περιμένει τα επόμενα πλαίσια Ethernet.

Η κατάσταση **MtpDataPacket_st** που είναι σκιασμένη στο Σχήμα 5-3 είναι μια απλοποιημένη απεικόνιση του τμήματος της μηχανής πεπερασμένων καταστάσεων που φαίνεται στο Σχήμα 15. Το κομμάτι αυτό ελέγχει αν το πεδίο Number του πακέτου δεδομένων είναι ίσο με την τιμή του Data Packet counter και αν ναι τότε

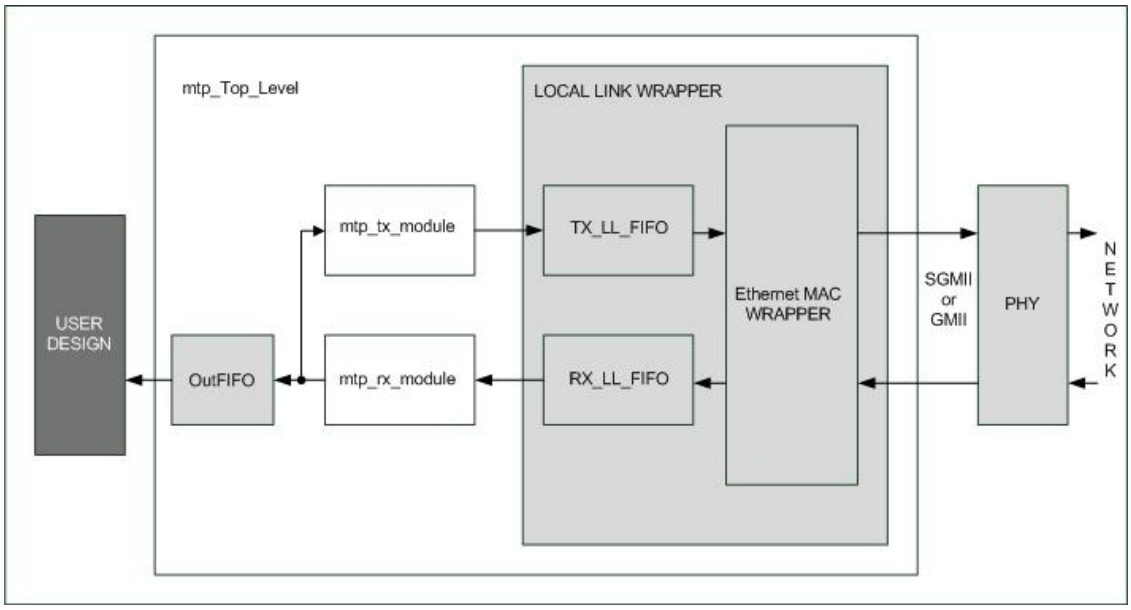
παράγει τα απαραίτητα σήματα για την εγγραφή του πεδίου δεδομένων στην OutFIFO.



Σχήμα 15: **Η κατάσταση MtpDataPacket_st**

Στην κατάσταση **MtpEndPacket_st** εξετάζεται αν το πεδίο Number του πακέτου MTP είναι ίσο με την τιμή του Data Packet Counter και ανάλογα με το αποτέλεσμα ενεργοποιεί στον καταχωρητή κατάστασης το bit επιτυχούς ολοκλήρωση συνόδου ή το bit λήψης πακέτου λήξης συνόδου εκτός σειράς.

5.1.2 Το MTP2.0



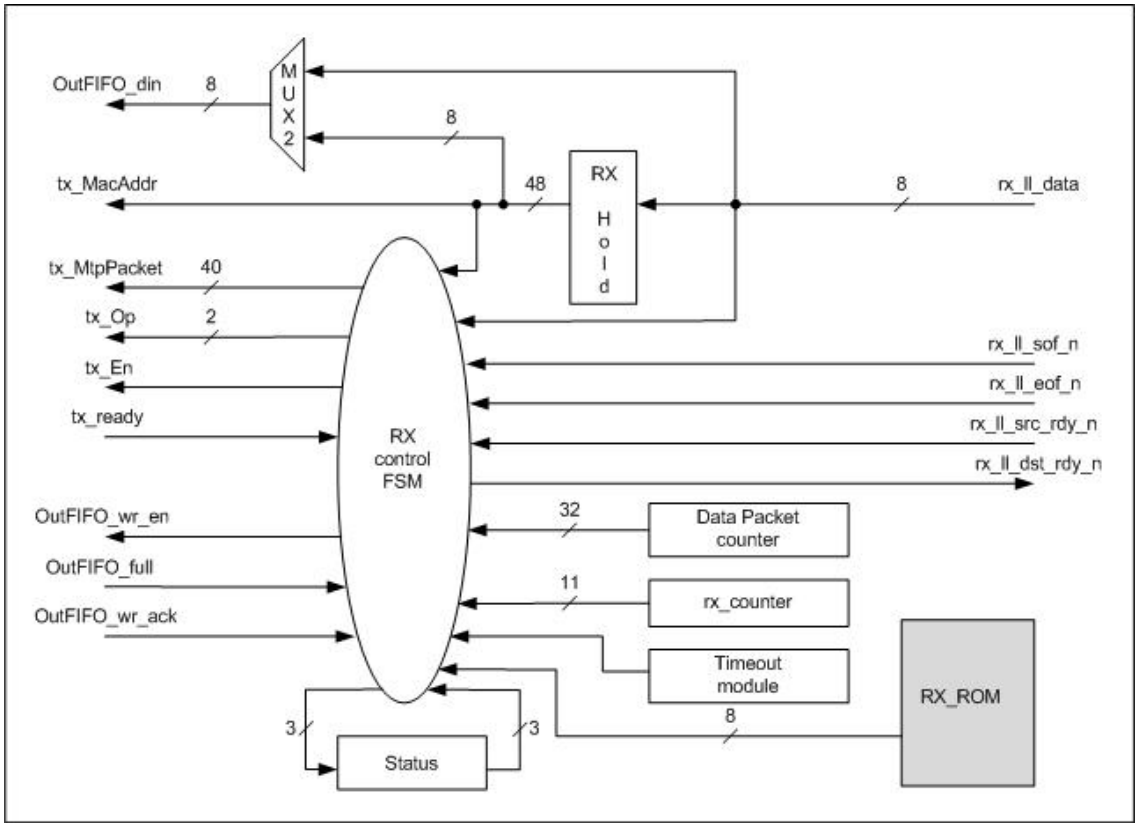
Σχήμα 16: **MTP2.0 Block Diagram**

Το Σχήμα 16 δείχνει μία αφηρημένη άποψη της υλοποίησης της σχεδίασης μας η οποία υποστηρίζει την έκδοση 2.0 του πρωτοκόλλου MTP. Σε αυτό απεικονίζονται οι κύριες λειτουργικές μονάδες (modules) καθώς και οι κύριες συνδέσεις μεταξύ τους.

Όπως και στη σχεδίαση που υποστηρίζει το MTP1.0, ο ενσωματωμένος Ethernet MAC ελεγκτής λαμβάνει τα πλαίσια Ethernet που στέλνει ο υπολογιστής, εκτελεί έλεγχο σφαλμάτων και τα γράφει στην RX_LL_FIFO. Στη συνέχεια, η λειτουργική μονάδα mtp_rx_module διαβάζει τα εισερχόμενα πλαίσια από την RX_LL_FIFO,

εξετάζει αν ενθυλακώνουν κάποιο πακέτο του πρωτοκόλλου MTP και αν ναι τότε γράφει το πεδίο Data στη FIFO του χρήστη (OutFIFO). Το mtp_rx_module παράγει επίσης τα πακέτα επιβεβαίωσης του πρωτοκόλλου MTP τα οποία και μεταβιβάζει στη λειτουργική μονάδα mtp_tx_module. Το mtp_tx_module αναλαμβάνει να παράξει τα πλαίσια Ethernet, που έχουν ενθυλακωμένα τα πακέτα IPv4 , UDP και MTP, και στη συνέχεια να τα γράψει στην TX_LL_FIFO για να τα μεταδώσει ο πομπός του Ethernet ελεγκτή. Τέλος, η σχεδίαση του χρήστη διαβάζει τα δεδομένα που έχουν μεταφερθεί μέσω της διεπαφής ανάγνωσης της OutFIFO (Πίνακας 2).

Το mtp_rx_module



Σχήμα 17: **Το mtp_rx_module**

Το Σχήμα 17 δείχνει την βασική υλοποίηση της λειτουργικής μονάδας mtp_rx_module η οποία αποτελεί επέκταση της λειτουργικής μονάδας mtp_module του MTP1.0. Στο σχήμα παρουσιάζονται οι λειτουργικές μονάδες και οι κύριες συνδέσεις τους. Παρουσιάζονται επίσης η διεπαφή (Local Link) του mtp_rx_module με την RX_LL_FIFO από όπου διαβάζει τα εισερχόμενα πλαίσια Ethernet, η διεπαφή με την OutFIFO όπου γράφει τα δεδομένα που προορίζονται για τη σχεδίαση του χρήστη και η διεπαφή με την λειτουργική μονάδα

mtp_tx_module στην οποία γράφονται τα πακέτα επιβεβαίωσης (Ready to Receive, Acknowledgment, Negative Acknowledgment, Close Socket) και η διεύθυνση MAC του υπολογιστή.

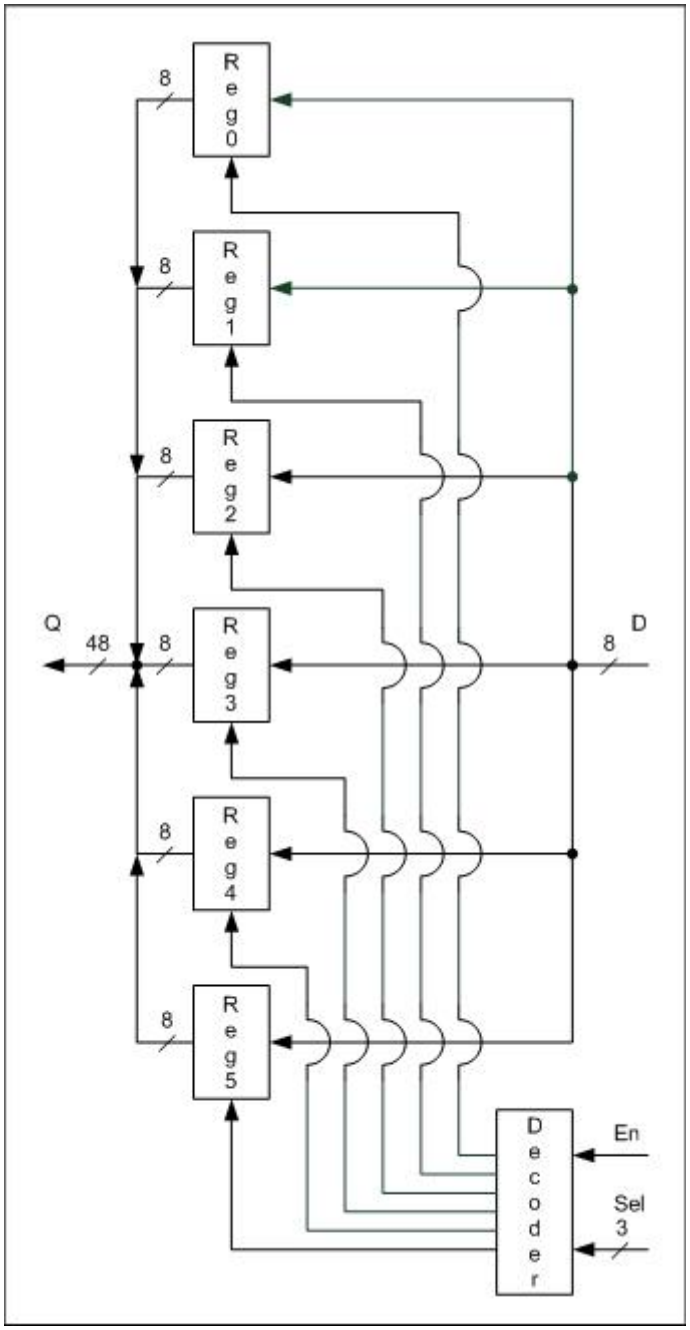
Η διαδρομή δεδομένων του mtp_rx_module αποτελείται από τον μετρητή **rx_counter**, η τιμή του οποίου προσδιορίζει τα πεδία των πακέτων που είναι ενθυλακωμένα σε κάθε πλαίσιο Ethernet, καθώς και τον μετρητή **Data Packet counter** ο οποίος προσδιορίζει τον αριθμό (πεδίο Number) του επόμενου MTP πακέτου δεδομένων που αναμένεται.

Η μνήμη **RX_ROM**, περιέχει τις διευθύνσεις MAC και IP καθώς και το Port που έχουμε ορίσει για το αναπτυξιακό μας (Πίνακας 4).

Θέση	**Τιμή (Hex)**	**Πεδίο**
0	66	Διεύθυνση MAC
1	66	
2	66	
3	66	
4	66	
5	66	
6	01	Διεύθυνση IP
7	02	
8	03	
9	05	
10	13	UDP Port
11	88	

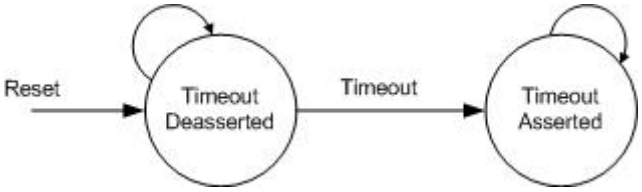
Πίνακας 4: **Τα περιεχόμενα της μνήμης RX_ROM**

Η λειτουργική μονάδα **RX_Hold** (Σχήμα 18) είναι μια συστοιχία έξι καταχωρητών που χρησιμοποιείται για την προσωρινή αποθήκευση της διεύθυνσης MAC πηγής του εισερχόμενου πλαισίου Ethernet (διεύθυνση MAC του υπολογιστή). Η διεύθυνση αυτή χρησιμοποιείται ως διεύθυνση προορισμού των πλαισίων Ethernet που αποστέλλονται στον υπολογιστή και ενθυλακώνουν τα πακέτα επιβεβαίωσης του πρωτοκόλλου MTP. Χρησιμοποιείται επίσης για την προσωρινή αποθήκευση του πεδίου Number των πακέτων MTP, ώστε αυτό να συγκριθεί με τον Data Packet counter, καθώς και για την καθυστέρηση των δεδομένων εσόδου για ένα κύκλο στην περίπτωση που δεν είναι δυνατή η εγγραφή στην OutFIFO



Σχήμα 18: **Το RX_Hold**

Η λειτουργική μονάδα **timeout_module** είναι ένας μετρητής που μετρά του κύκλους ρολογιού που μεσολαβούν μεταξύ της λήψης δύο διαδοχικών πλαισίων Ethernet τα οποία έχουν ενθυλακωμένα πακέτα MTP. Στην περίπτωση που ο αριθμός των κύκλων ρολογιού είναι μεγαλύτερος από ένα καθορισμένο όριο (timeout) το σήμα Timeout ενεργοποιείται και σχεδίαση μας στέλνει στον υπολογιστή ένα MTP πακέτο αρνητικής επιβεβαίωσης με τον αριθμό του πακέτου δεδομένων που περιμένουμε (Data Packet counter). Το Σχήμα 19 δείχνει την μηχανή πεπερασμένων καταστάσεων που ενεργοποιεί το σήμα Timeout.



Σχήμα 19: **Timeout**

Ο καταχωρητής κατάστασης (**Status Register**) χρησιμοποιείται, όπως και στο mtp_module, για να κρατάει πληροφορία σχετικά την κατάσταση της συνόδου MTP. Ο Πίνακας 5 δείχνει τα bit του καταχωρητή κατάστασης και την πληροφορία που αυτά αναπαριστούν.

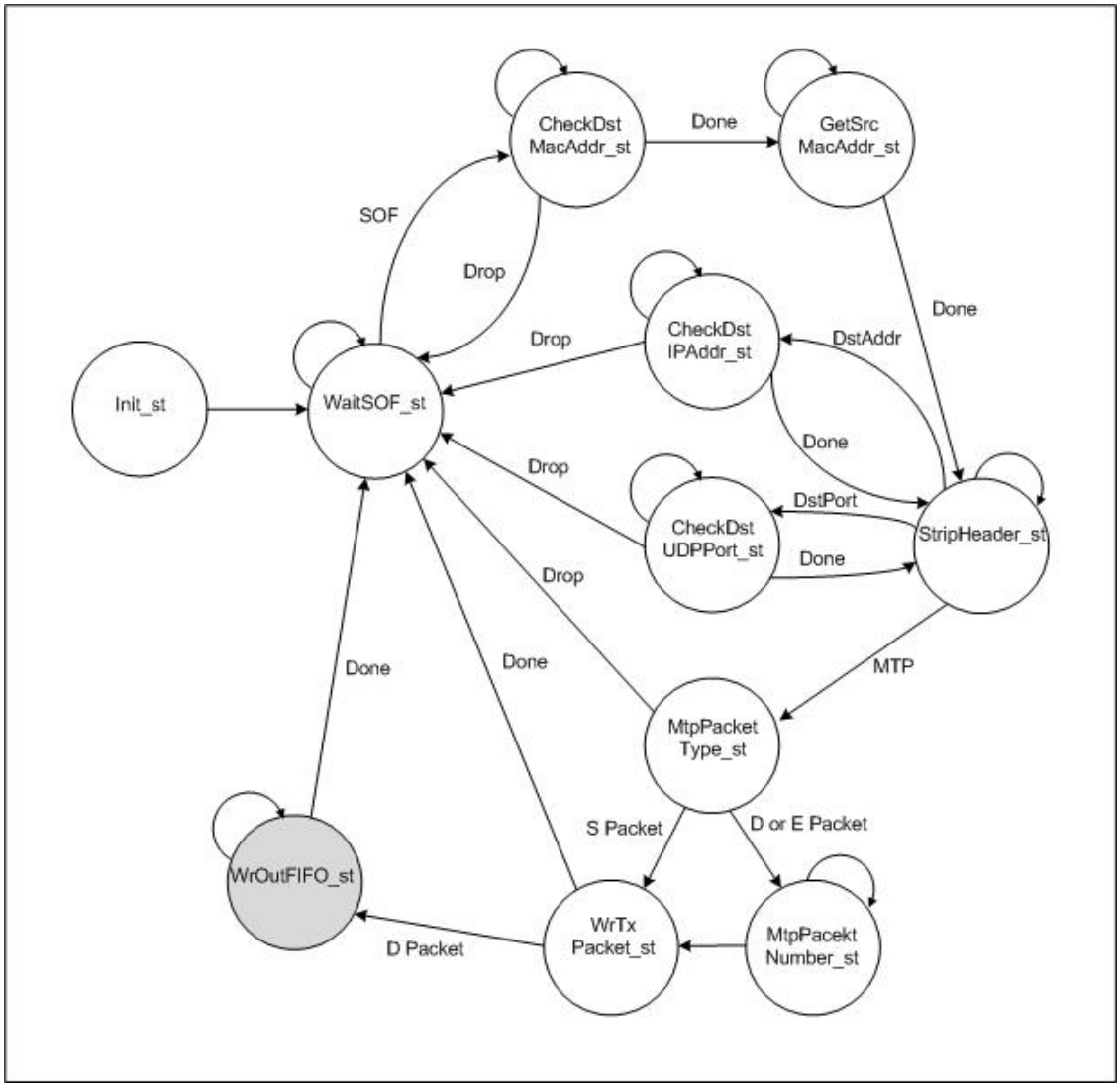
Bit	Κατάσταση
0	Σύνοδος σε εξέλιξη
1	MTP πακέτο δεδομένων
2	MTP πακέτο λήξης συνόδου

Πίνακας 5: **Περιεχόμενα καταχωρητή κατάστασης**

Τέλος, το mtp_rx_module διαθέτει την μονάδα ελέγχου **RX control FSM**, η οποία βασίζεται στην αντίστοιχη μονάδα ελέγχου του mtp_module του MTP1.0, και παράγει τα σήματα για τον έλεγχο των λειτουργικών μονάδων και των διεπαφών για την ανάγνωση των πλαισίων Ethernet από την RX_LL_FIFO, την εγγραφή των δεδομένων στην OutFIFO και την εγγραφή των πακέτων επιβεβαίωσης MTP στην λειτουργική μονάδα mtp_tx_module.

Το Σχήμα 20 δείχνει το διάγραμμα καταστάσεων της μονάδας ελέγχου του mtp_rx_module. Οι καταστάσεις Init_st, WaitSOF_st, StripHeader_st, CheckDstMacAddr_st, CheckDstIPAddr_st και CheckUDPPort_st προέρχονται από την μηχανή πεπερασμένων καταστάσεων RX control FSM του mtp_module.

Στην κατάσταση **Init_st** αρχικοποιούνται (reset) οι καταχωρητές και οι μετρητές του mtp_rx_module. Από την κατάσταση Init_st η μηχανή μεταβαίνει στην κατάσταση **WaitSOF_st** και διαβάζει την RX_LL_FIFO έως ότου ληφθεί το πρώτο byte ενός πλαισίου Ethernet. Στη συνέχεια, η μηχανή μεταβαίνει στην κατάσταση **CheckDstMacAddr_st** στην οποία συγκρίνει την διεύθυνση MAC προορισμού με την διεύθυνση που είναι αποθηκευμένη στην μνήμη RX_ROM.



Σχήμα 20: **Η μονάδα ελέγχου RX control FSM**

Από την κατάσταση CheckDstMacAddr_st η μηχανή μεταβαίνει στην κατάσταση **GetSrcMacAddr_st** στην οποία η διεύθυνση MAC του υπολογιστή αποθηκεύεται προσωρινά στη λειτουργική μονάδα RX_Hold.

Στη συνέχεια η μηχανή μεταβαίνει διαδοχικά στις καταστάσεις **StripHeader_st**, όπου αφαιρούνται τα πεδία των κεφαλίδων (του Ethernet, του IPv4 και του UDP) που δεν ελέγχουμε, την **CheckDstIPAddr_st** στην οποία ελέγχεται η IP διεύθυνση προορισμού και την **CheckDstUDPPort_st** όπου ελέγχεται το port προορισμού.

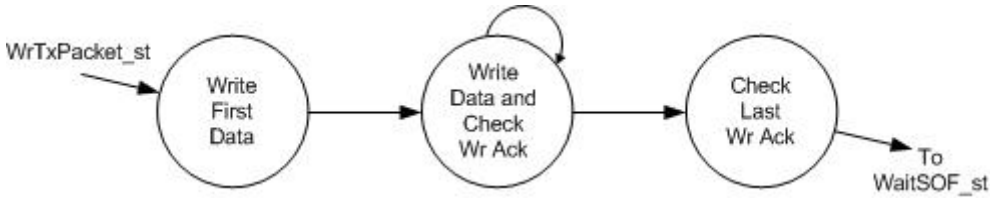
Όταν φτάσει στο πεδίο δεδομένων του πακέτου UDP η μηχανή μεταβαίνει στην κατάσταση **MtpPacketType_st** όπου ελέγχεται το πεδίο Type του πακέτου MTP. Ανάλογα με την τιμή του πεδίου Type η μηχανή μεταβαίνει είτε στην κατάσταση

WrTxPacket_st αν έχουμε πακέτο έναρξης συνόδου είτε στην κατάσταση MtpPacketNumber_st αν έχουμε πακέτο δεδομένων ή πακέτο λήξης συνόδου. Τέλος, αν η τιμή του πεδίου Type δεν προσδιορίζει κάποια από τις παραπάνω περιπτώσεις το πακέτο απορρίπτεται και η μηχανή μεταβαίνει στην κατάσταση WaitSOF_st.

Στην κατάσταση **MtpPacketNumber_st** ελέγχεται το πεδίο Number του πακέτου MTP. Αν το πεδίο είναι ίσο με την τιμή του μετρητή Data Packet counter τότε το πακέτο έχει ληφθεί εν σειρά διαφορετικά αν η τιμή του πεδίου Number είναι μεγαλύτερη από την τιμή του μετρητή Data Packet counter έχουμε απώλεια δεδομένων. Στη συνέχεια η μηχανή μεταβαίνει στην κατάσταση WrTxPacket_st.

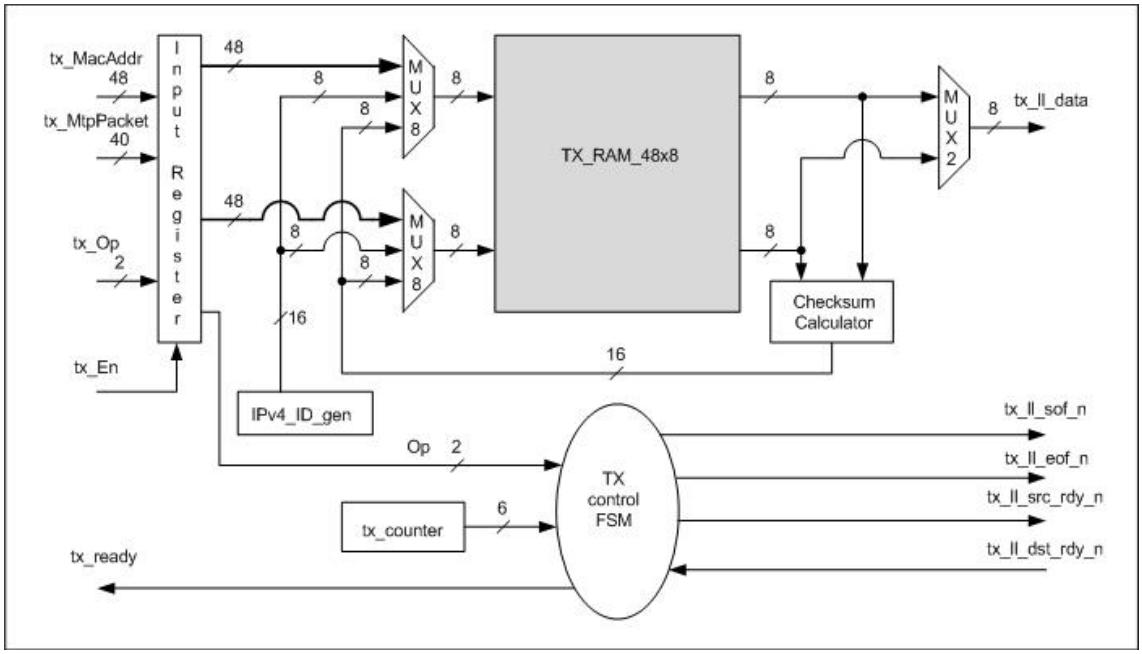
Στην κατάσταση **WrTxPacket_st** παράγονται τα πακέτα επιβεβαίωσης σύμφωνα με το πρωτόκολλο MTP2.0. Παράγει επίσης, τα απαραίτητα σήματα για την εγγραφή των πακέτων αυτών καθώς και της διεύθυνσης MAC του υπολογιστή στη λειτουργική μονάδα mtp_tx_module. Αν το πακέτο που λάβαμε είναι ένα έγκυρο πακέτο δεδομένων η μηχανή μεταβαίνει στην κατάσταση WrOutFIFO_st διαφορετικά μεταβαίνει στην κατάσταση WaitSOF_st αναμένοντας το επόμενο πλαίσιο Ethernet.

Η κατάσταση **WrOutFIFO_st** που είναι σκιασμένη στο Σχήμα 20 είναι μια απλοποιημένη απεικόνιση του τμήματος της μηχανής πεπερασμένων καταστάσεων που φαίνεται στο Σχήμα 21. Το κομμάτι αυτό παράγει τα απαραίτητα σήματα για την εγγραφή του πεδίου δεδομένων του MTP πακέτου στην OutFIFO.



Σχήμα 21: **Η κατάσταση WrOutFIFO_st**

Το mtp_tx_module



Σχήμα 22: **Το mtp_tx_module**

Το Σχήμα 22 δείχνει την βασική υλοποίηση της λειτουργικής μονάδας mtp_tx_module. Στο σχήμα παρουσιάζονται οι λειτουργικές μονάδες και οι κύριες συνδέσεις τους. Παρουσιάζονται επίσης η διεπαφή (Local Link) του mtp_tx_module με την TX_LL_FIFO, στην οποία γράφει τα πλαίσια Ethernet που αποστέλλονται στον υπολογιστή, και η διεπαφή με τη λειτουργική μονάδα mtp_rx_module από όπου διαβάζει τα πακέτα επιβεβαίωσης του πρωτοκόλλου MTP καθώς και την MAC διεύθυνση του υπολογιστή.

Η διαδρομή δεδομένων του mtp_tx_module έχει χτιστεί γύρω από την μνήμη **TX_RAM_48x8**. Η μνήμη αυτή έχει αρχικοποιηθεί με το πρότυπο των πλαισίων Ethernet τα οποία ενθυλακώνουν τα πακέτα επιβεβαίωσης του πρωτοκόλλου MTP2.0 που αποστέλλονται στον υπολογιστή. Το πρότυπο καθορίζει τις τιμές που ανατίθενται στατικά, στο σύστημα μας, στα πεδία των κεφαλίδων των πρωτοκόλλων Ethernet, IPv4 και UDP (Πίνακας 6). Τα πεδία που η τιμή τους δεν καθορίζονται στατικά στο σύστημα μας είναι η διεύθυνση MAC προορισμού της κεφαλίδας του πλαισίου Ethernet, τα πεδία Identification και Checksum της κεφαλίδας του πακέτου IPv4 και του πεδίο Checksum της κεφαλίδας του δεδομενογράμματος UDP.

Θέση	Τιμή (Hex)	Πεδίο	Θέση	Τιμή (Hex)	Πεδίο
0	--	PC MAC Addr.	24	--	IP Checksum
1	--		25	--	
2	--		26	01	FPGA IP Addr.
3	--		27	02	
4	--		28	03	
5	--		29	05	
6	66	FPGA MAC Addr.	30	01	PC IP Addr.
7	66		31	02	
8	66		32	03	
9	66		33	09	
10	66		34	00	Source UDP Port
11	66		35	00	
12	08	Type Field	36	13	Destination UDP Port
13	00		37	8D	
14	45	Version, IHL	38	00	Datagram Length
15	00	Type of Services	39	0D	
16	00	Total Length	40	--	UDP Checksum
17	21		41	--	
18	--	Identification Field	42	--	MTP Packet Type
19	--		43	--	MTP Packet No.
20	00	Flags/ Frag. Offset	44	--	
21	00		45	--	
22	80	Time To Live	46	--	
23	11	Protocol	47	00	Zero Pad

Πίνακας 6: **Τα περιεχόμενα της μνήμης TX_RAM_48x8**

Ο καταχωρητής εισόδου (**Input Register**) χρησιμοποιείται για την αποθήκευση του πακέτου επιβεβαίωσης MTP (εύρους 40 bits) και της διεύθυνση MAC του υπολογιστή (εύρους 48 bits), τα οποία παράγονται από την λειτουργική μονάδα mtp_rx_mdule.

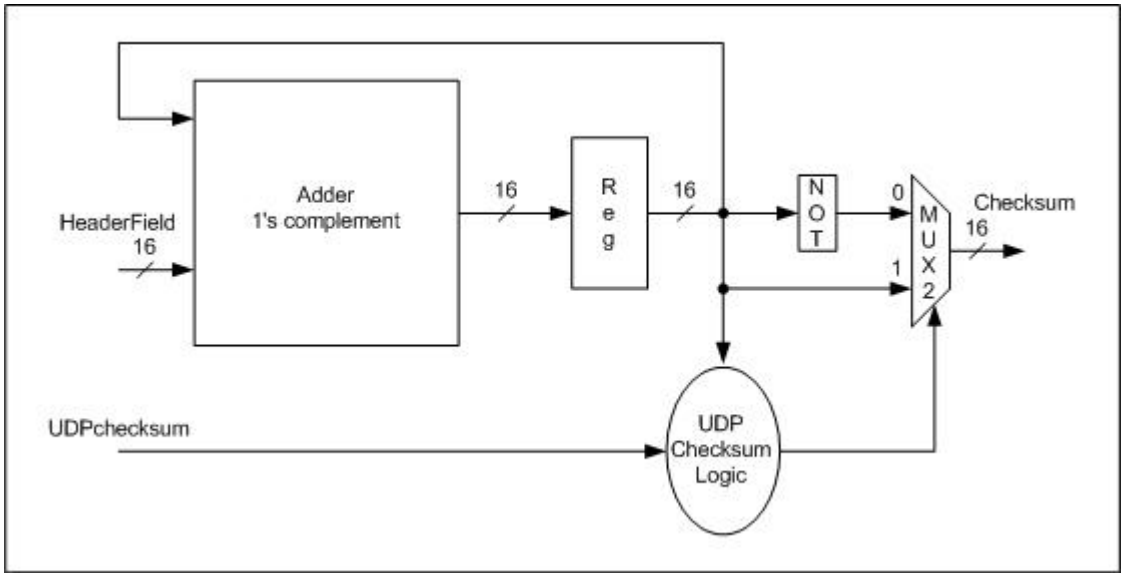
Η λειτουργική μονάδα **IPv4_ID_gen** παράγει την τιμή του πεδίου Identification της κεφαλίδας του πακέτου IPv4. Το πεδίο Identification μαζί με τα πεδία IP διεύθυνσης προορισμού και αποστολέα καθορίζουν μοναδικά την ταυτότητα του πακέτου IPv4. Διαδοχικά πακέτα IPv4, από έναν αποστολέα προς ένα παραλήπτη, έχουν διαδοχικές τιμές στο πεδίο Identification.

Η λειτουργική μονάδα **Checksum Calculator** (Σχήμα 23) χρησιμοποιείται για να υπολογιστούν τα πεδία αθροίσματος ελέγχου (Checksum) των κεφαλίδων IPv4 και UDP [12]. Το Checksum Calculator αποτελείται από έναν αθροιστή εύρους 16 bits, ο οποίος προσθέτει δυαδικούς αριθμούς που αναπαρίστανται στη μορφή του συμπληρώματος ως προς 1, και έναν καταχωρητή όπου αποθηκεύεται το μερικό

άθροισμα στη διάρκεια των υπολογισμών. Η λειτουργική μονάδα διαθέτει επίσης την απαραίτητη λογική για την επιλογή ως έξοδο είτε του αθροίσματος είτε του συμπληρώματος του ως προς 1 ανάλογα με το πρωτόκολλο (IPv4, UDP) και την τιμή του αθροίσματος.

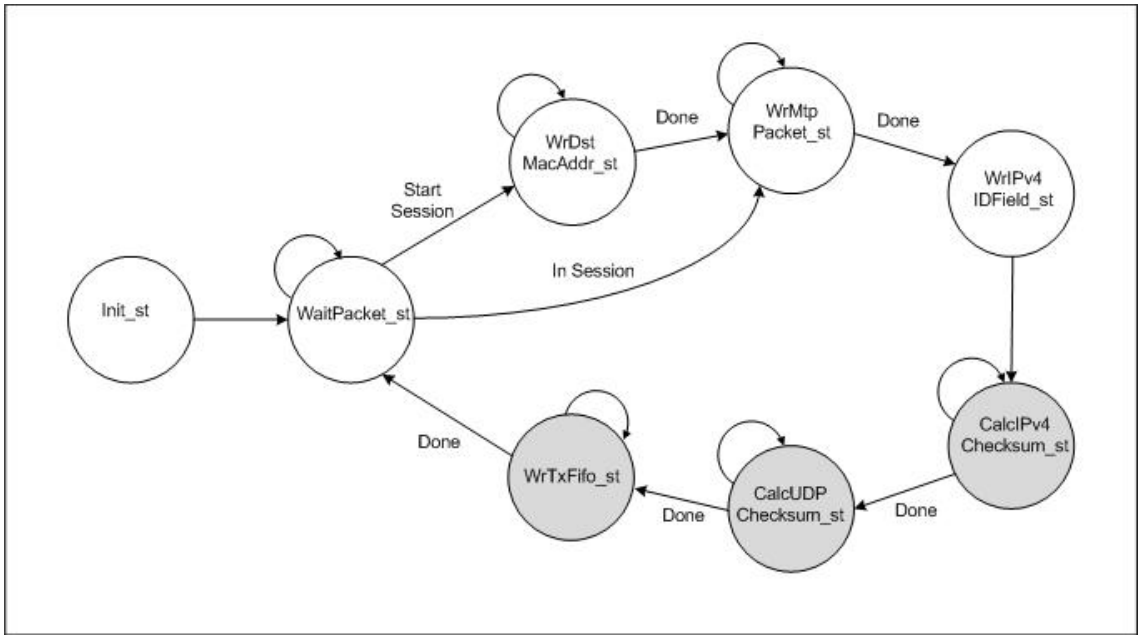
Το πεδίο αθροίσματος ελέγχου της κεφαλίδας του πακέτου IPv4 υπολογίζεται λαμβάνοντας το συμπλήρωμα ως προς 1 (one's complement) του αθροίσματος ως προς 1 όλων των λέξεων εύρους 16 bit της κεφαλίδας.

Το πεδίο αθροίσματος ελέγχου της κεφαλίδας UDP υπολογίζεται λαμβάνοντας το συμπλήρωμα ως προς 1 του αθροίσματος ως προς 1 των λέξεων εύρους 16 bit μιαw ψευδό κεφαλίδας με πληροφορίες από την κεφαλίδα IPv4, της κεφαλίδας και των δεδομένων του δεδομενογράμματος UDP. Αν το αποτέλεσμα του υπολογισμού της τιμής του αθροίσματος ελέγχου είναι μηδέν (όλα τα 16 bits 0) τότε χρησιμοποιείται το συμπλήρωμα του ως προς 1 (όλα τα 16 bits 1).



Σχήμα 23: **Το Checksum Calculator**

Η διαδρομή δεδομένων του mtp_tx_module διαθέτει τον μετρητή **tx_counter** εύρους 6 bits η τιμή του οποίου χρησιμοποιείται από τη μονάδα ελέγχου **TX control FSM**. Η μονάδα ελέγχου του mtp_tx_module είναι που υλοποιημένη σαν μια μηχανή πεπερασμένων καταστάσεων τύπου mealy παράγει τα σήματα για τον έλεγχο των λειτουργικών μονάδων και των διεπαφών για την επικοινωνία με το mtp_rx_module και την εγγραφή των προς αποστολή πλαισίων Ethernet στη TX_LL_FIFO.



Σχήμα 24: **Η μονάδα ελέγχου TX control FSM**

Το Σχήμα 24 δείχνει το διάγραμμα καταστάσεων της μονάδας ελέγχου (TX control FSM) του mtp_tx_module.

Στην κατάσταση **Init_st**, η ποία είναι η αρχική κατάσταση της μηχανής, αρχικοποιούνται (reset) οι καταχωρητές και οι μετρητές του mtp_tx_module.

Από την κατάσταση Init_st η μηχανή μεταβαίνει στην κατάσταση **WaitPacket_st** και ενεργοποιεί το σήμα tx_ready. Στην κατάσταση αύτη παραμένει έως ότου το mtp_rx_module γράψει στον καταχωρητή εισόδου (Input Resgister). Στη συνέχεια, εάν ο καταχωρητής εισόδου περιέχει μία έγκυρη διεύθυνση MAC (tx_Op = “11”), η μηχανή μεταβαίνει στην κατάσταση WrDstMacAddr_st διαφορετικά (tx_Op = ”01”) η μηχανή μεταβαίνει στην κατάσταση WrMtpPacket_st.

Στην κατάσταση **WrDstMacAddr_st** διαβάζεται η διεύθυνση MAC του υπολογιστή από τον καταχωρητή εισόδου και μέσω των δύο πολυπλεκτών 8 σε 1 (MUX8) γράφεται στη μνήμη TX_RAM_48x8 (πεδίο PC MAC Addr). Όταν ολοκληρωθεί η εγγραφή της διεύθυνσης MAC προορισμού η μηχανή μεταβαίνει στην κατάσταση WrMtpPacket_st.

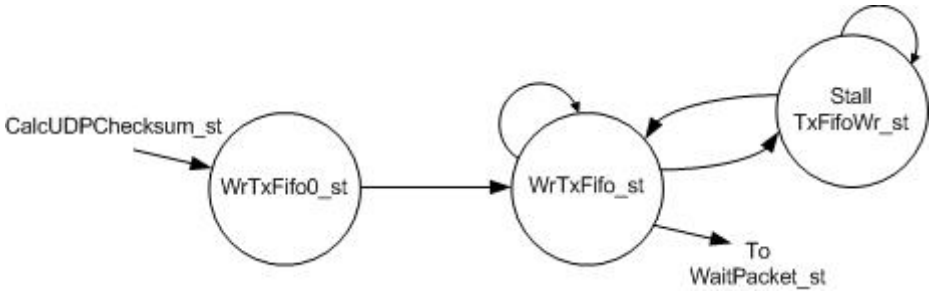
Στην κατάσταση **WrMtpPacket_st** διαβάζεται το πακέτο επιβεβαίωσης MTP από τον καταχωρητή εισόδου και γράφεται στη μνήμη TX_RAM_48x8 (πεδία MTP Packet Type και MTP Packet No).

Από την κατάσταση WrMtpPacket_st η μηχανή μεταβαίνει την κατάσταση **WrIPv4IDField_st** στην οποία γράφεται στη μνήμη η τιμή του πεδίου Identification που παράγεται στη λειτουργική μονάδα IPv4_ID_gen.

Στη συνέχεια, η μηχανή μεταβαίνει στην κατάσταση **CalcIPv4Checksum_st** στην οποία τα πεδία της κεφαλίδας IPv4 διαβάζονται από την μνήμη TX_RAM_48x8 και δίνονται στη λειτουργική μονάδα Checksum Calculator για να υπολογιστεί το άθροισμα ελέγχου της κεφαλίδας. Όταν υπολογιστεί το άθροισμα ελέγχου γράφεται στη μνήμη (πεδίο IP Checksum) και η μηχανή μεταβαίνει στην κατάσταση CalcUDPChecksum_st.

Στην κατάσταση **CalcUDPChecksum_st** υπολογίζεται το άθροισμα ελέγχου της κεφαλίδας UDP και γράφεται στη μνήμη TX_RAM_48x8 (πεδίο UDP Checksum). Η κατάσταση CalcUDPChecksum_st όπως και η CalcIPv4Checksum_st που φαίνονται σκιασμένες στο Σχήμα 24 είναι απλοποιημένες απεικονίσεις περισσότερων καταστάσεων που υλοποιούν τη λειτουργικότητα που περιγράφεται. Η παραπάνω κατάτμηση των καταστάσεων έγινε για να μπορεί η σχεδίαση μας να λειτουργεί στη συχνότητα ρολογιού των 125 MHz που λειτουργεί ο ενσωματωμένος Ethernet MAC ελεγκτής της Xilinx που χρησιμοποιούμε.

Όταν ολοκληρωθεί η δημιουργία του πλαισίου Ethernet που θα σταλεί στον υπολογιστή, η μηχανή μεταβαίνει στην κατάσταση **WrTxFifo_st**. Η κατάσταση αυτή είναι μία απλοποιημένη απεικόνιση του τμήματος της μηχανής πεπερασμένων καταστάσεων που φαίνεται στο Σχήμα 25. Το κομμάτι αυτό παράγει τα απαραίτητα σήματα για την εγγραφή πλαισίου Ethernet στην TX_LL_FIFO.



Σχήμα 25: **Η κατάσταση WrTxFIFO_st**

5.2 Υλοποίηση

5.2.1 Βασικά υποσυστήματα των δύο σχεδιάσεων

Όπως αναφέραμε στην ενότητα 4.2, το αναπτυξιακό που χρησιμοποιήσαμε για να υλοποιήσουμε και να δοκιμάσουμε την σχεδίαση μας ήταν το XUP ML509 XC5VLX110T της οικογένειας Virtex 5. Για την ανάπτυξη του κομματιού της σχεδίασης μας που αντιστοιχεί σε υλικό χρησιμοποιήθηκαν γλώσσα περιγραφής υλικού VHDL και τα εργαλεία Xilinx ISE Design Suite 10.1 και Xilinx EDK 10.1

Διαμόρφωση του V5 Embedded Tri-Mode Ethernet MAC

Τα απαραίτητα αρχεία VHDL για την διαμόρφωση του ενσωματωμένου Ethernet MAC ελεγκτή δημιουργήθηκαν με το εργαλείο Virtex-5 Embedded Tri-Mode Ethernet MAC Wrapper v1.6 του Xilinx Core Generator 10.1.

Στα πλαίσια της μελέτης για την κατανόηση της λειτουργίας και του τρόπου αξιοποίησης του ελεγκτή παράχθηκαν τρεις διαμορφώσεις του. Σε όλες τις περιπτώσεις το παράδειγμα σχεδίασης, που παράγει ο Core Generator, προστέθηκε σαν περιφερειακό σε μία σχεδίαση του EDK, του παρασχέθηκαν τα απαραίτητα σήματα ρολογιού και τροποποιήθηκε κατάλληλα το αρχείο User Constrain File (UCF).

Στην πρώτη διαμόρφωση [51] που δοκιμάστηκε ο ελεγκτής ρυθμίστηκε να λειτουργεί σε ταχύτητες 1000/100/10 Mbps οι οποίες επιτυγχάνονται με συχνότητα διαφορικού ρολογιού 125 MHz που υποβαθμίζεται με χρήση του Digital Clock Manager (DCM) σε 12.5 MHz και 1.25 MHz. Η διασύνδεση φυσικού επιπέδου-υποεπιπέδου MAC που υλοποιήθηκε ήταν η SGMII.

Στην δεύτερη διαμόρφωση, ο ελεγκτής ρυθμίστηκε να λειτουργεί σε ταχύτητα 1 Gbps με συχνότητα διαφορικού ρολογιού 125 MHz. Η διασύνδεση φυσικού επιπέδου-υποεπιπέδου MAC που υλοποιήθηκε ήταν πάλι η SGMII.

Στην τρίτη διαμόρφωση που δοκιμάστηκε ο ελεγκτής ρυθμίστηκε να λειτουργεί σε ταχύτητα 1 Gbps με συχνότητα ρολογιού 125 MHz και συχνότητα ρολογιού αναφοράς 200 MHz. Η διασύνδεση φυσικού επιπέδου-υποεπιπέδου MAC που υλοποιήθηκε ήταν η SGMII.

Τέλος, η προσπάθεια να διαμορφωθεί ο ελεγκτής έτσι ώστε να χρησιμοποιεί τη διασύνδεση RGMII συγκαλύφθηκε καθώς η ρύθμιση των ρολογιών λειτουργίας και

αναφοράς ήταν αρκετά απαιτητική και προτιμήθηκε να συνεχίσουμε στα επόμενα στάδια της εργασίας.

Όπως έχει ήδη αναφερθεί, στη συνέχεια τη εργασίας μας χρησιμοποιήθηκε η δεύτερη διαμόρφωση που περιγράφηκε. Η επιλογή αυτή έγινε επειδή η συγκεκριμένη διαμόρφωση υποστηρίζει την λειτουργικότητα που χρειάζεται η σχεδίαση μας προβάλλοντας μικρότερες απαιτήσεις διαχείρισης ρολογιού σε σχέση με τις άλλες δύο.

Δημιουργία της OutFIFO

Η OutFIFO που χρησιμοποιείται για την μεταβίβαση των δεδομένων που έρχονται από τον υπολογιστή στη σχεδίαση του χρήστη δημιουργήθηκε με το εργαλείο FIFO Generator v4.4 του Xilinx Core Generator 10.1.

Η FIFO συνίσταται από 4 block RAM και έχει εύρος ανάγνωση/εγγραφής 8 bit και βάθος 8192. Το μέγεθος της συγκεκριμένης FIFO είναι καθοριστικός παράγοντας για την απόδοση του πρωτοκόλλου μας, αφού επιτρέπει στη σχεδίαση του χρήστη να καταναλώνει τα δεδομένα με ρυθμό και τρόπο που αυτή καθορίζει. Η επιλογή μας ήταν αυθαίρετη εξ' αιτίας του γεγονότος ότι δεν υπήρχαν στη διάθεση μας κάποιου χρήστη. Παρά ταύτα, είναι εύκολο να προσαρμοστεί το μεγέθος της OutFIFO ανάλογα με τις απαιτήσεις της εκάστοτε σχεδίασης χρήστη.

Δημιουργία των RX_ROM και TX_RAM_48x8

Οι μνήμες RX_ROM και TX_RAM_48x8, που χρησιμοποιούνται στη σχεδίαση που υλοποιεί το MTP2.0, δημιουργήθηκαν με το εργαλείο Block Memory Generator v2.8 του Core Generator 10.1. Η RX_ROM είναι μια μνήμη μόνο για ανάγνωση (Read Only Memory) 12 θέσεων η οποία διαθέτει μια θύρα ανάγνωσης και μία θύρα εγγραφής εύρους 8 bit. Η TX_RAM_48x8 έχει υλοποιηθεί ως μία δίπορτη μνήμη τυχαίας προσπέλασης και έχει 48 θέσεις εύρους 8 bit.

5.2.2 Αξιοποίηση πόρων της FPGA

Οι δύο σχεδιάσεις που υλοποιήθηκαν (MTP1.0 και MTP2.0) λειτουργούν σε συχνότητα ρολογιού 125 MHz η οποία είναι η συχνότητα λειτουργίας του ενσωματωμένου Ethernet MAC ελεγκτή της Xilinx που χρησιμοποιήθηκε.

Στον Πίνακα 7 παρουσιάζονται οι πόροι της FPGA που καταναλώνει η σχεδίαση του MTP1.0.

Device utilization summary	Used	Available	Utilization
Number of Slice Registers	574	69120	0 %
Number of Slice LUTs	569	69120	0 %
Number of fully used LUT-FF pairs	338	805	41 %
Number of bonded IOBs	6	640	0 %
Number of Block RAM/FIFO	4	148	2 %
Number of BUFG/BUFGCTRLs	1	32	3 %

Πίνακας 7: **Χρησιμοποίηση πόρων της FPGA από τη σχεδίαση MTP1.0**

Στον Πίνακα 8 παρουσιάζονται οι πόροι της FPGA που καταναλώνει η σχεδίαση του MTP2.0.

Device utilization summary	Used	Available	Utilization %
Number of Slice Registers	803	69120	1 %
Number of Slice LUTs	952	69120	1 %
Number of fully used LUT-FF pairs	517	805	41 %
Number of bonded IOBs	6	640	0 %
Number of Block RAM/FIFO	4	148	2 %
Number of BUFG/BUFGCTRLs	1	32	3 %

Πίνακας 8: **Χρησιμοποίηση πόρων της FPGA από τη σχεδίαση MTP2.0**

5.2.3 Εφαρμογή ηλεκτρονικού υπολογιστή

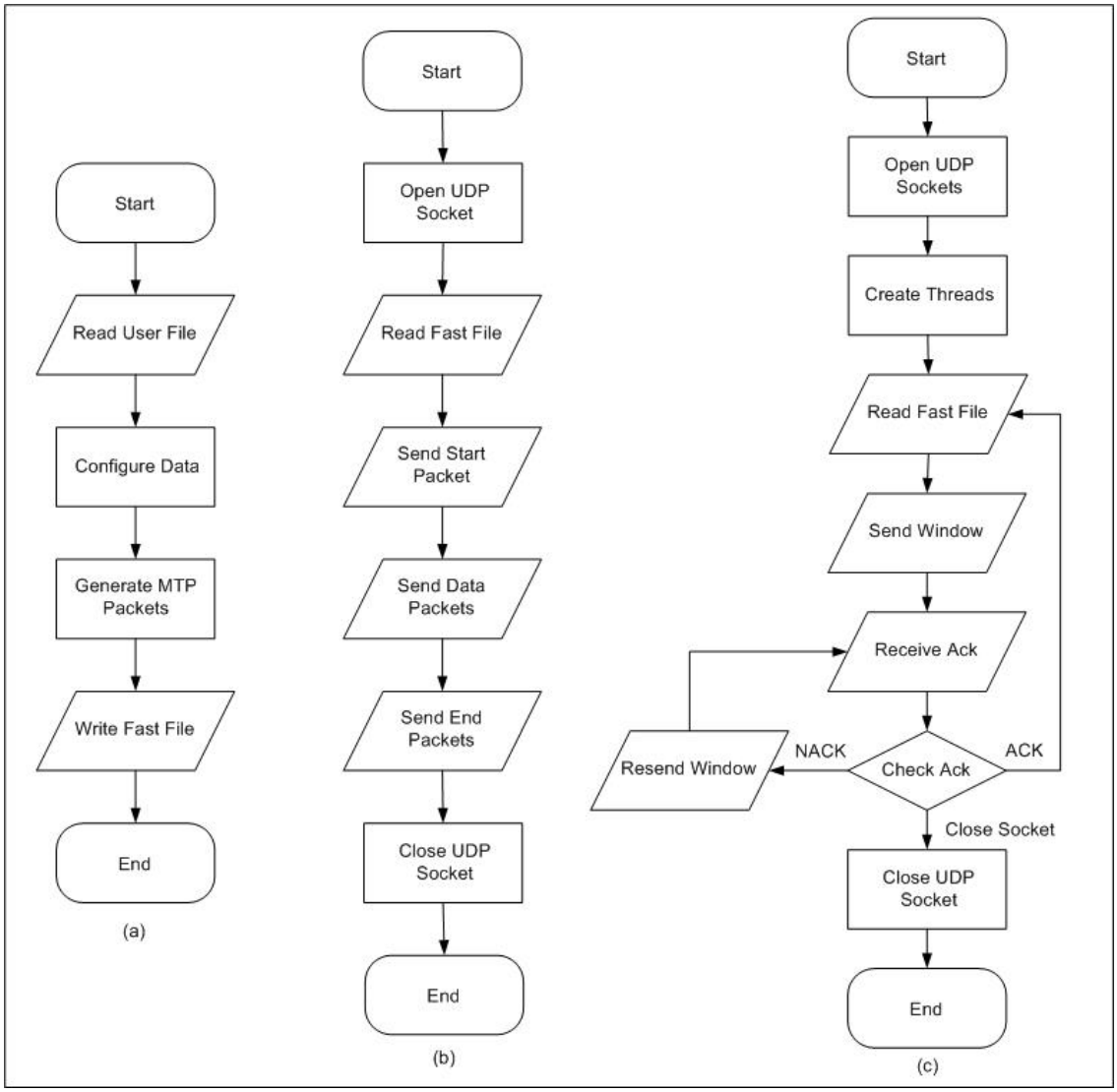
Για την υλοποίηση των δύο εκδόσεων του πρωτοκόλλου MTP στον υπολογιστή αναπτύξαμε το απαιτούμενο λογισμικό χρησιμοποιώντας τις γλώσσες προγραμματισμού Python και C.

Πριν την αποστολή των δεδομένων προηγείται ένα στάδιο off-line επεξεργασίας, που έχει υλοποιηθεί σε Python, στο οποίο παράγονται τα MTP πακέτα δεδομένων και αποθηκεύονται σε ένα ενδιάμεσο αρχείο (.fast). Σκοπός αυτής της διαδικασίας είναι να αφαιρέσει τον φόρτο παρασκευής των MTP πακέτων από τη διαδικασία αποστολής.

Στη συνέχεια, εφ' όσων χρησιμοποιείται το MTP1.0 το πρόγραμμα που το υλοποιεί ανοίγει ένα UDP socket, διαβάζει τα MTP πακέτα από το αρχείο .fast και τα στέλνει στο αναπτυξιακό. Όταν ολοκληρωθεί η παραπάνω διαδικασία το πρόγραμμα κλείνει το UDP socket και τερματίζει. Το συγκεκριμένο πρόγραμμα, υλοποιήθηκε αρχικά σε γλώσσα Python και μπορεί να εκτελεστεί σε λειτουργικά συστήματα Windows και Unix. Αργότερα, υλοποιήθηκε σε γλώσσα C (μόνο για Unix) με στόχο να επιτευχθεί αύξηση της ταχύτητας μεταφοράς.

Εάν χρησιμοποιείται το MTP2.0, το πρόγραμμα που το υλοποιεί -σε γλώσσα Python- ανοίγει δύο UDP sockets (ένα για αποστολή δεδομένων και ένα για λήψη επιβεβαιώσεων) και δημιουργεί δύο νήματα. Το πρώτο νήμα διαβάζει το .fast αρχείο και στέλνει τα πακέτα δεδομένων σε παράθυρα στην FPGA. Το δεύτερο νήμα λαμβάνει τα πακέτα επιβεβαίωσης από την σχεδίαση μας στο αναπτυξιακό, αν είναι Ack δίνει εντολή στο πρώτο νήμα να στείλει το επόμενο παράθυρο πακέτων ενώ αν είναι Nack δίνει εντολή στο πρώτο νήμα να ξαναστείλει το προηγούμενο παράθυρο από το πακέτο που υποδεικνύει το Nack και μετά.

Στο Σχήμα 26 παρουσιάζονται τα διαγράμματα ροής των προγραμμάτων ofline επεξεργασίας (a), MTP1.0 (b) και MTP2.0 (c).



Σχήμα 26: **Τα διαγράμματα ροής των προγραμμάτων ofline επεξεργασίας (a), MTP1.0 (b) και MTP2.0 (c)**

5.3 Επιβεβαίωση ορθής λειτουργίας

Για την επιβεβαίωση της ορθής λειτουργίας του συστήματος μας χρησιμοποιήσαμε το εργαλείο Xilinx ISE Simulator, με το οποίο προσομοιώθηκε η λειτουργία των επιμέρους λειτουργικών υπομονάδων, το εργαλείο Xilinx Chipscope Pro 10.1, με το οποίο έγινε δειγματοληψία των σημάτων της σχεδίασης μας στην FPGA, και το πρόγραμμα Wireshark, με το οποίο παρακολουθούσαμε την κίνηση των πακέτων στην κάρτα δικτύου του υπολογιστή.

5.3.1 Επιβεβαίωση λειτουργίας του Embedded Tri-Mode Ethernet MAC

Οι τρεις διαμορφώσεις του ενσωματωμένου Ethernet-MAC ελεγκτή που παράχθηκαν ελέγχτηκαν αν ήταν δυνατό να αποκαταστήσουν επικοινωνία Ethernet με τον υπολογιστή του εργαστηριού. Για τον σκοπό αυτό προγραμματίσαμε την FPGA με το παράδειγμα σχεδίασης που παρήγαγε ο Core Generator (Loop-back), την συνδέσαμε με την κάρτα δικτύου του υπολογιστή και στείλαμε πακέτα ping τα οποία και είδαμε, να μεταδίδονται και να λαμβάνονται, με το εργαλείο Wireshark.

Στη συνέχεια υπολογίσαμε την ταχύτητα μεταφοράς δεδομένων των τριών διαμορφώσεων. Για τον υπολογισμό της ταχύτητας ανοίξαμε ένα UDP socket και στείλαμε στην FPGA συνολικά 1 GByte ASCII χαρακτήρες σε UDP πακέτα μεγέθους 1 KByte. Ο συνολικός αριθμός κύκλων ρολογιού που χρειάστηκε για την μεταφορά και ο αριθμός των Byte δεδομένων μετρήθηκαν με μετρητές, που ενσωματώσαμε στο παράδειγμα σχεδίασης, την τιμή του οποίου διαβάσαμε μέσω της θύρας RS-232 του αναπτυξιακού. Με βάση τις παραπάνω μετρήσεις, καταλήξαμε στο συμπέρασμα ότι οι τρεις διαμορφώσεις του Ethernet MAC ελεγκτή επιτυγχάνουν παρόμοιες τιμές ταχύτητας.

5.3.2 Επιβεβαίωση σωστής λειτουργίας του MTP1.0

Για την διαπίστωση της σωστής λειτουργίας του συστήματος (υλικό και λογισμικό) που υλοποιεί την έκδοση MTP1.0 ακολουθήσαμε την διαδικασία που περιγράφεται παρακάτω.

Αρχικά, υλοποιήσαμε μια δική μας «σχεδίαση χρήστη» την οποία συνδέσαμε στην OutFIFO. Η σχεδίαση αυτή διαβάζει την OutFIFO και μετράει τον αριθμό των Bytes δεδομένων που στέλνει ο υπολογιστής στην FPGA. Με αυτόν τον τρόπο ελέγξαμε ότι δεν υπήρχε απώλεια πληροφορίας, κατά τη διάρκεια μιας συνόδου MTP1.0, που να μην μπορεί να ανιχνευτεί από τους ελέγχους του πρωτοκόλλου.

Στη συνέχεια, προγραμματίσαμε την FPGA με τις σχεδιάσεις του MTP1.0 και την σχεδίαση χρήστη. Με το πρόγραμμα που υλοποιεί το MTP1.0 στον υπολογιστή στείλαμε πακέτα στην FPGA, τα οποία μαζί με τα σήματα ελέγχου της σχεδίασης MTP1.0 τα δειγματοληπτήσαμε με το εργαλείο Xilinx Chipscope pro.

Χρησιμοποιώντας την παραπάνω διάταξη πραγματοποιήσαμε ελεγχόμενες συνόδους MTP, στις ποίες καλύψαμε μια σειρά πιθανών σφαλμάτων, και μπορέσαμε να επιβεβαιώσουμε την λειτουργία της σχεδίασης σύμφωνα τις προδιαγραφές του πρωτοκόλλου MTP1.0. Τα πειράματα μας περιελάμβαναν συνόδους που είχαμε απώλεια πακέτων δεδομένων, απώλεια πακέτου έναρξης συνόδου και διπλών (αντίγραφα) πακέτων δεδομένων.

Στο σημείο αυτό πρέπει να επισημάνουμε ότι σε όλα τα πειράματα που κάναμε με το MTP1.0 παρατηρήσαμε απώλεια πακέτων λόγω υπερχείλισης των FIFO, όταν η σχεδίαση χρήστη είχε ρυθμιστεί να καταναλώνει τα δεδομένα με ρυθμό μικρότερο από αυτόν που κατέφθαναν. Αντίθετα, δεν παρατηρήσαμε απώλεια πακέτων λόγω σφαλμάτων που αυτά είχαν αν και πραγματοποιήσαμε συνόδους που μεταφέρθηκε μεγάλος όγκος δεδομένων.

Τέλος, ο διδακτορικός φοιτητής κ. Ευριπίδης Σωτηριάδης δοκίμασε τη σχεδίαση του MTP1.0 και τη σύνδεσε στη δική του σχεδίαση του Blast.

5.3.3 Επιβεβαίωση σωστής λειτουργίας του MTP2.0

Για την επιβεβαίωση της σωστής λειτουργίας του συστήματος (υλικό και λογισμικό) που υλοποιεί την έκδοση MTP2.0 ακολουθήσαμε την διαδικασία που περιγράφεται παρακάτω.

Αρχικά, υλοποιήσαμε μία σχεδίαση η οποία διαβάζει «έτοιμα» δεδομενογράμματα UDP από μια μνήμη BRAM και τα γράφει την TX_LL_FIFO για να σταλούν στον Η/Υ. Από την πλευρά του υπολογιστή, ανοίξαμε ένα UDP socket το οποίο λαμβάνει τα δεδομενογράμματα που στέλνει η FPGA, παράλληλα με το εργαλείο Wireshark παρακολουθούσαμε τα εισερχόμενα πλαίσια Ethernet στην κάρτα δικτύου του υπολογιστή. Με τη διάταξη αυτή επιβεβαιώσαμε ότι είναι δυνατό να στείλουμε πακέτα επιβεβαίωσης του πρωτοκόλλου MTP από την FPGA στον Η/Υ.

Στη συνέχεια, συνδέσαμε την λειτουργική μονάδα mtp_rx_module στις RX_LL_FIFO και OutFIFO και προγραμματίσαμε την FPGA. Χρησιμοποιήσαμε το πρόγραμμα που υλοποιεί το MTP1.0 στον υπολογιστή για να στείλουμε πακέτα

στην FPGA και με το εργαλείο Xilinx Chipscope Pro δειγματοληπτήσαμε τα εισερχόμενα πακέτα, τα σήματα ελέγχου και τα πακέτα επιβεβαίωσης που παράγονται από το mtp_rx_module.

Με την παραπάνω διάταξη πραγματοποιήσαμε μεγάλο αριθμό ελέγχων και επιβεβαιώσαμε την σωστή λειτουργία του mtp_tx_module. Πιο συγκεκριμένα διαπιστώσαμε ότι όλοι οι έλεγχοι των πλαισίων Ethernet εκτελούνταν σωστά και κάθε φορά παραγόταν το κατάλληλο πακέτο επιβεβαίωσης σύμφωνα με το πρωτόκολλο MTP2.0.

Προβλήματα

Η σχεδίαση που υλοποιεί το MTP2.0 δεν κατέστη πλήρως λειτουργική εξ' αιτίας προβλημάτων που αντιμετωπίσαμε στην ανάπτυξη της λειτουργικής μονάδας mtp_tx_module. Τα προβλήματα εντοπίζονται στην παραγωγή των πακέτων IPv4 και UDP που ενθυλακώνουν τα πακέτα επιβεβαίωσης MTP.

Πιο συγκεκριμένα, ένα σχεδιαστικό πρόβλημα που εντοπίσαμε αργά κατά τη διαδικασία της αποσφαλμάτωσης του κώδικα μας οφείλεται στην μη πλήρη υλοποίηση των πρωτόκολλων IPv4 και UDP σε υλικό. Η προσέγγιση μας όσον αφορά τα πεδία των κεφαλίδων των παραπάνω πρωτοκόλλων είναι να αναθέτουμε σε αυτά στατικές τιμές και να υπολογίζουμε κάθε φορά τα αθροίσματα ελέγχου. Όμως, το πεδίο ταυτότητας μαζί με τα πεδία IP διεύθυνσης προορισμού και αποστολέα καθορίζουν μοναδικά την ταυτότητα του κάθε πακέτου IPv4. Επομένως, διαδοχικά πακέτα IPv4, από έναν αποστολέα προς ένα παραλήπτη, πρέπει να έχουν διαδοχικές τιμές στο πεδίο ταυτότητας. Το πρόβλημα αυτό λύθηκε με τη σχεδίαση και ενσωμάτωση στο mtp_tx_module της λειτουργικής μονάδας IPv4_ID_gen η οποία παράγει μία έγκυρη τιμή του πεδίου ταυτότητας για κάθε πακέτο IPv4.

Αυτή τη στιγμή, τα σημαντικότερα προβλήματα εντοπίζονται στην μονάδα ελέγχου TX control FSM, η οποία δεν λειτουργεί συμφώνα με τις προδιαγραφές.

Κεφάλαιο 6

Πειραματικά Αποτελέσματα

Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα των πειραματικών μετρήσεων της ταχύτητας μεταφοράς του πρωτοκόλλου MTP1.0 καθώς και μια εκτίμηση για την αντίστοιχη του MTP1.0.

6.1 Πειραματικά αποτελέσματα του MTP1.0

Η ταχύτητα μεταφοράς δεδομένων του MTP1.0 μετρήθηκε με τη βοήθεια σχεδίασης χρήστη που συνδέθηκε στην OutFIFO. Η σχεδίαση καταναλώνει τα δεδομένα με ρολόι 125 MHz και χρησιμοποιεί μετρητές για να μετρήσει τον αριθμό κύκλων που διήρκησε μια σύνοδος. Οι τιμές των μετρητών διαβάστηκαν με το εργαλείο Xilinx Chipscope Pro. Για την επιβεβαίωση των αποτελεσμάτων, μετρήθηκε και ο χρόνος εκτέλεσης του προγράμματος που υλοποιεί την σύνοδο MTP1.0. Οι δύο μέθοδοι είχαν ταυτόσημα αποτελέσματα.

Στα πειράματα μας χρησιμοποιήθηκε ένα φορητός Η/Υ με επεξεργαστή Intel Core 2 Duo 2.1 GHz, 1 GByte RAM με Ethernet PHY 1000/100/10 Mbps και λειτουργικό σύστημα Mac OS X 10.5.8. Δεδομένα όγκου 500 MByte στάλθηκαν στο αναπτυξιακό τόσο με την έκδοση σε Python όσο και με την έκδοση σε C του προγράμματος που υλοποιεί το MTP1.0 και μετρήθηκε η ταχύτητα μεταφοράς για μέγεθος πεδίου δεδομένων του πακέτου MTP από 512 ως 1467 Bytes.

Data Field (+Headers) Bytes	**2.1GHz, 1GB RAM, Mac OS X, Python**	**2.1GHz, 1GB RAM, Mac OS X, C**	**2.4GHz, 4GB RAM, Windows XP, Python**
512 (545)	188.75 Mbps	279.35 Mbps	343.60 Mbps
768 (801)	280.26 Mbps	406.91 Mbps	505.29 Mbps
991 (1024)	359.41 Mbps	548.87 Mbps	--
1024 (1057)	356.43 Mbps	532.54 Mbps	660.76 Mbps
1280 (1313)	444.84 Mbps	654.72 Mbps	--
1467 (1500)	522.18 Mbps	**776.67 Mbps**	--

Πίνακας 9: **Πειραματικές Μετρήσεις Ταχύτητας Μεταφοράς Δεδομένων για το MTP1.0**

Στον Πίνακα 9 παρουσιάζοντα οι πειραματικές μετρήσεις της ταχύτητας μεταφοράς δεδομένων. Η πρώτη στήλη δείχνει το μέγεθος του πεδίου δεδομένων του πακέτου MTP και σε παρένθεση το μέγεθος του πεδίου δεδομένων του πλαισίου Ethernet

(MTP Data + MTP Header + UDP Header + IPv4 Header). Η δεύτερη και η τρίτη στήλη περιέχουν τις πειραματικές μετρήσεις που κάναμε. Τέλος η τέταρτη στήλη δείχνει μετρήσεις που έκανε ο κ. Σωτηριάδης με ένα φορητό Η/Υ με επεξεργαστή Intel Core 2 Duo P8600 2.4 GHz, 4GByte RAM με Ethernet PHY Intel 82567LM και λειτουργικό σύστημα Windows XP SP3.

Αποτίμηση Απόδοσης

Παρατηρώντας τα πειραματικά αποτελέσματα του Πίνακα 9 βλέπουμε ότι η υλοποίηση του προγράμματος του MTP1.0 σε γλώσσα προγραμματισμού C επιτυγχάνει γενικά 1.47 φορές πιο γρήγορη μεταφορά δεδομένων σε σχέση με την υλοποίηση σε Python. Επίσης, παρατηρούμε ότι η εκτέλεση του προγράμματος Python στον «ταχύτερο» υπολογιστή επιτυγχάνει 1.82 φορές πιο γρήγορη μεταφορά δεδομένων σε σχέση με τον υπολογιστή που εκτελέσαμε τα πειράματα μας.

Όσον αφορά το μέγεθος πακέτου, παρατηρούμε ότι όσο αυξάνεται η πληροφορία που ενθυλακώνεται αυξάνεται και η ταχύτητα μεταφοράς δεδομένων. Εξαίρεση από τον παραπάνω κανόνα αποτελεί η περίπτωση που το πεδίο δεδομένων του πλαισίου Ethernet έχει μέγεθος στην περιοχή γύρω από τα 1024 Bytes (σειρές 3 και 4 του Πίνακα 9). Η διαφοροποίηση αυτή πιθανώς να οφείλεται στον τρόπο υπολογισμού του πεδίου κυκλικού ελέγχου πλεονασμού του πλαισίου Ethernet στην κάρτα δικτύου του υπολογιστή [52].

Τέλος, από τα πειράματα που εκτελέσαμε διαπιστώσαμε ότι κρίσιμος παράγοντας για επιτευχθεί η καλύτερη δυνατή ταχύτητα, όταν μεταφέρεται μεγάλος όγκος δεδομένων, πρέπει ο υπολογιστής που χρησιμοποιείται να διαθέτει επάρκεια σε μνήμη RAM. Σε διαφορετική περίπτωση θα έχουμε σφάλμα σελίδας (Page Fault) κατά τη διάρκεια της μετάδοσης των δεδομένων, γεγονός που θα έχει σημαντικές επιπτώσεις στην ταχύτητα μεταφοράς.

6.2 Πειραματικά αποτελέσματα του MTP2.0

Δεδομένου ότι η σχεδίαση που υλοποιεί σε υλικό το MTP2.0 τελικά δεν λειτούργησε δεν έχουμε πειραματικές μετρήσεις για την ταχύτητα μεταφοράς δεδομένων της δεύτερης έκδοσης του πρωτοκόλλου MTP.

Ο Πίνακας 10 δείχνει μια εκτίμηση της ταχύτητας μεταφοράς δεδομένων του MTP2.0. Οι μετρήσεις έγιναν με τη σχεδίαση που υλοποιεί το MTP1.0 προσομοιώνοντας τον επιπλέον φόρτο της εφαρμογής στον Η/Υ λόγο της αρχικής

χειραψίας των δύο πλευρών και του ελέγχου των πακέτων επιβεβαίωσης. Οι υποθέσεις που κάναμε είναι ότι το πεδίο τιμών των πακέτων MTP είναι 1467 Bytes και ότι δεν έχουμε απώλεια πακέτων κατά τη διάρκεια της συνόδου. Χρησιμοποιήθηκε ο ίδιος φορητός Η/Υ με επεξεργαστή Intel Core 2 Duo 2.1 GHz, 1 GByte RAM με Ethernet PHY 1000/100/10 Mbps και λειτουργικό σύστημα Mac OS X 10.5.8 όπως και πριν.

Window Size	**2.1GHz, 1GB RAM, Mac OS X, Python**
16	419.02 Mbps
32	443.24 Mbps
64	452.82 Mbps
128	461.58 Mbps

Πίνακας 10: **Εκτίμηση της Ταχύτητα Μεταφοράς Δεδομένων του MTP2.0**

Όπως ήταν αναμενόμενο, παρατηρούμε ότι η εκτιμώμενη ταχύτητα μεταφοράς δεδομένων του MTP2.0 είναι μικρότερη από την αντίστοιχη του MTP1.0 για ίδιο μέγεθος πακέτου. Επίσης, παρατηρούμε ότι απόδοση του πρωτοκόλλου αυξάνει όσο μεγαλώνει το μέγεθος του παραθύρου. Βέβαια, αν το τελευταίο λάβει πολύ μεγάλες τιμές το κόστος που πληρώνουμε όταν υπάρχει απώλεια δεδομένων αυξάνεται σημαντικά.

Κεφάλαιο 7

Συμπεράσματα και Μελλοντική Εργασία

Στο τελευταίο αυτό κεφάλαιο αναφέρονται τα συμπεράσματα και μελλοντικές επεκτάσεις του συστήματος μας.

7.1 Συμπεράσματα

Σε αυτήν την διπλωματική εργασία μελετήσαμε την δυνατότητα γρήγορης μεταφοράς δεδομένων από τον Η/Υ μια αναδιατασσόμενη συσκευή της οικογένειας Virtex-5, με χρήση του πρωτοκόλλου Gigabit Ethernet. Στα πλαίσια αυτά, σχεδιάσαμε τις δύο εκδόσεις του πρωτοκόλλου επιπέδου εφαρμογής MTP και αναπτύξαμε το απαραίτητο υλικό και λογισμικό για την υλοποίηση του. Τέλος, επιβεβαιώσαμε τη σωστή λειτουργία του MTP1.0 και συνδέσαμε την σχεδίαση μας σε ένα μεγαλύτερο σύστημα.

7.2 Μελλοντική Εργασία

Σαν μελλοντική εργασία προτείνεται η ολοκλήρωση της υλοποίησης του MTP2.0 και η εύρεση των βέλτιστων παραμέτρων του (Timeout, Window, μέγεθος πακέτου) με στόχο να μεγιστοποιηθεί ο ρυθμός μεταφοράς δεδομένων.

Προτείνεται ακόμη η επέκταση του πρωτοκόλλου MTP ώστε να υποστηρίζεται η μεταφορά δεδομένων από την FPGA στον υπολογιστή.

Τέλος, προτείνεται η υλοποίηση του πρωτοκόλλου GoBackN από την πλευρά του υπολογιστή (στην πλευρά της FPGA απαιτείται απλώς ρύθμιση να στέλνεται πακέτο επιβεβαίωσης για κάθε πακέτο δεδομένων) και η σύγκριση του με το MTP.

Βιβλιογραφία

Βιβλία

[1] J. F. Kurose, K. W. Ross, *Δικτύωση Υπολογιστών, Προσέγγιση από Πάνω προς τα Κάτω με Έμφαση στο Διαδίκτυο,* Δεύτερη Έκδοση, Αθήνα: Εκδόσεις Μ. Γκιούρδας, 2004.

[2] J. Walrand, *Δίκτυα Επικοινωνιών*, Πρώτη Έκδοση, Αθήνα: Εκδόσεις Παπασωτηρίου, 1997.

[3] A. S. Tanenbaum, *Δίκτυα υπολογιστών,* Τέταρτη Αμερικάνικη Έκδοση, Αθήνα: Εκδόσεις Κλειδάριθμος, 2003.

[4] M. J. Donahoo, K. L. Calvert, *TCP/IP SOCKETS IN C, Practical Guide for Programmers,* Second Edition, Morgan Kaufmann Publishers, Inc., 2009.

Δημοσιεύσεις

[5] R. M. Metcalfe, D. R. Boggs, *"Ethernet: Distributed Packet Switching for Local Computer Networks",* Communications of the ACM, 19 (7), pp 395-402, July 1976,

[6] H. Fallside, M. J. S. Smith, *Internet Connected FPL*, Proceedings of the The Roadmap to Reconfigurable Computing, 10th International Workshop on Field-Programmable Logic and Applications 2000 pp. 48-57.

[7] A. Dollas, I. Ermis, I. Koidis, et al., "An Open TCP/IP Core for Reconfigurable Logic", Proceedings, IEEE Symposium on Field Programmable Computing Machines (FCCM), pp. 297-298, Napa Valley, USA, April, 2005.

Πρότυπα

[8] J. Postel, *User Datagram Protocol, RFC 768,* August 1980, (http://www.ietf.org/rfc/rfc0768.txt)

[9] Information Sciences Institute University of Southern California, *Internet Protocol, RFC 791,* September 1981, (http://www.ietf.org/rfc/rfc791.txt)

[10] Information Sciences Institute University of Southern California, *Transmission Control Protocol, RFC 793,* September 1981, (http://www.ietf.org/rfc/rfc793.txt)

[11] David C. Plummer, *An Ethernet Address Resolution Protocol, RFC 826,* November 1982, (http://www.ietf.org/rfc/rfc826.txt)

[12] B. Braden, D. Borman, and C. Partridge, *Computing the Internet Checksum, RFC 1071,* September 1988, (http://tools.ietf.org/html/rfc1071)

[13] Xilinx, Inc., *Xilinx SP006 LocalLink Interface Specification, Specification*

Τεχνικά Εγχειρίδια

[14] Xilinx, Inc., *Xilinx DS201 10-Gigabit Ethernet MAC v9.3, Data Sheet,* (http://www.xilinx.com/support/documentation/ip_documentation/ten_gig_eth_mac_ds201.pdf)

[15] Xilinx, Inc., *Xilinx UG148 LogiCORE™ IP 10-Gigabit Ethernet MAC v9.3, User guide,* (http://www.xilinx.com/support/documentation/ip_documentation/ten_gig_eth_mac_ug148.pdf)

[16] Xilinx, Inc., *Xilinx DS266 XAUI v9.1, Data Sheet*, (http://www.xilinx.com/support/documentation/ip_documentation/xaui_ds266.pdf)

[17] Xilinx, Inc., *Xilinx UG150 LogiCORE™ IP XAUI v9.1, User Guide,* (http://www.xilinx.com/support/documentation/ip_documentation/xaui_ug150.pdf)

[18] Xilinx, Inc., *Xilinx DS200 1-Gigabit Ethernet MAC v8.5, Data Sheet*, (http://www.xilinx.com/support/documentation/ip_documentation/gig_eth_mac_ds200.pdf)

[19] Xilinx, Inc., *Xilinx UG144 LogiCORE™ IP 1-Gigabit Ethernet MAC v8.5, User Guide*, (http://www.xilinx.com/support/documentation/ip_documentation/gig_eth_mac_ug144.pdf)

[20] Xilinx, Inc., *Xilinx DS264 Ethernet1000BASE-XPCS/PMA or SGMII v10.3, Data Sheet*, (http://www.xilinx.com/support/documentation/ip_documentation/gig_eth_pcs_pma_ds264.pdf)

[21] Xilinx, Inc., *Xilinx UG155 LogiCORE™ IP Ethernet 1000BASE-X PCS/PMA or SGMII v10.3, User Guide*, (http://www.xilinx.com/support/documentation/ip_documentation/gig_eth_pcs_pma_ug155.pdf)

[22] Xilinx, Inc., *Xilinx DS297 Tri-Mode Ethernet MAC v4.3, Data Sheet*, (http://www.xilinx.com/support/documentation/ip_documentation/tri_mode_eth_mac_ds297.pdf)

[23] Xilinx, Inc., *Xilinx UG138 LogiCORE™ IP Tri-Mode Ethernet MAC v4.3, User Guide,* (http://www.xilinx.com/support/documentation/ip_documentation/tri_mode_eth_mac_ug138.pdf)

[24] Xilinx, Inc., *Xilinx DS307 Virtex-4 Tri-Mode Embedded Ethernet MAC Wrapper v4.7, Data Sheet*, (http://www.xilinx.com/support/documentation/ip_documentation/v4_emac_ds307.pdf)

[25] Xilinx, Inc., *Xilinx UG074 Virtex-4 FPGA Embedded Tri-Mode Ethernet MAC, User Guide*, (http://www.xilinx.com/support/documentation/user_guides/ug074.pdf)

[26] Xilinx, Inc., *Xilinx DS550 Virtex-5FPGA Embedded Tri-Mode Ethernet MAC Wrapper v1.6, Data Sheet*, (http://www.xilinx.com/support/documentation/ip_documentation/v5_emac_ds550.pdf)

[27] Xilinx, Inc., *Xilinx UG194 Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC, User Guide*, (http://www.xilinx.com/support/documentation/user_guides/ug194.pdf)

[28] Xilinx, Inc., *Xilinx UG069 Xilinx University Program Virtex-II Pro Development System, Hardware Reference Manual*, (http://www.xilinx.com/univ/XUPV2P/Documentation/ug069.pdf)

[29] Xilinx, Inc., *Xilinx UG080 ML401/ML402/ML403 Evaluation Platform, User Guide*, (http://www.xilinx.com/support/documentation/boards_and_kits/ug080.pdf)

[30] Xilinx, Inc., *Xilinx UG085 ML410 Embedded Development Platform, User Guide*, (http://www.xilinx.com/support/documentation/boards_and_kits/ug085.pdf)

[31] Xilinx, Inc., *Xilinx UG347 ML505/ML506/ML507 Evaluation Platform, User Guide*, (http://www.xilinx.com/support/documentation/boards_and_kits/ug347.pdf)

[32] Xilinx, Inc., *Xilinx XAPP443 Ethernet Cores Hardware Demonstration Platform, Application Note*, (http://www.xilinx.com/support/documentation/application_notes/xapp443.pdf)

[33] Xilinx, Inc., *Xilinx XAPP433 Web Server Design Using MicroBlaze Soft Processor, Application Note*, (http://www.xilinx.com/support/documentation/application_notes/xapp433.pdf)

Διαδίκτυο

[34] Wikipedia, the free encyclopedia, *TCP/IP model, article* (http://en.wikipedia.org/wiki/TCP/IP_model)

[35] Wikipedia, the free encyclopedia, *IPv4, article*, (http://en.wikipedia.org/wiki/IPv4)

[36] IEEE Working Group, *IEEE 802.3 Ethernet,* (http://www.ieee802.org/3/)

[37] Wikipedia, the free encyclopedia, *Media Access Control, article,* (http://en.wikipedia.org/wiki/Media_Access_Control)

[38] Wikipedia, the free encyclopedia, *Ethernet II framing, article,* (http://en.wikipedia.org/wiki/Ethernet_II_framing)

[39] Wikipedia, the free encyclopedia, *Media Independent Interface, article,* (http://en.wikipedia.org/wiki/Media_Independent_Interface)

[40] Wikipedia, the free encyclopedia, *Reduced Media Independent Interface, article,* (http://en.wikipedia.org/wiki/Reduced_Media_Independent_Interface)

[41] Wikipedia, the free encyclopedia, *Gigabit Media Independent Interface, article,* (http://en.wikipedia.org/wiki/Gigabit_Media_Independent_Interface)

[42] Wikipedia, the free encyclopedia, *Reduced Gigabit Media Independent Interface, article,* (http://en.wikipedia.org/wiki/Reduced_Gigabit_Media_Independent_Interface)

[43] Wikipedia, the free encyclopedia, *Serial Gigabit Media Independent Interface, article,* (http://en.wikipedia.org/wiki/Serial_Gigabit_Media_Independent_Interface)

[44] Wikipedia, the free encyclopedia, *10 Gigabit Media Independent Interface, article*, (http://en.wikipedia.org/wiki/10_Gigabit_Media_Independent_Interface)

[45] uClinux, *Embedded Linux/Microcontroller Project*, (http://www.uclinux.org/)

[46] Digilent Inc, *MicroBlaze uClinux, porject*, (http://www.digilentinc.com/Data/Products/XUPV2P/uclinux-xupv2p_rev_1_1.zip)

[47] opencores.org, *10_100_1000 Mbps tri-mode Ethernet MAC*, (http://www.opencores.org/project,ethernet_tri_mode,overview)

[48] opencores.org, *Ethernet MAC 10/100 Mbps*, (http://www.opencores.org/project,ethmac,overview)

[49] Xilinx, Inc., *Xilinx XAPP433 Web Server Design Using MicroBlaze Soft Processor, Application Note*, (http://www.xilinx.com/support/documentation/application_notes/xapp433.pdf)

[50] savannah.nongnu.org, *lwIP - A Lightweight TCP/IP stack*, (http://savannah.nongnu.org/projects/lwip/)

[51] fpgadeveloper.com, edk tutorials for xilinx fpgas, *Tri-mode Ethernet Mac, Tutorial*, (http://www.fpgadeveloper.com/2008/10/tri-mode-ethernet-mac.html)

[52] Wikipedia, the free encyclopedia, *Computation of CRC, article,* (http://en.wikipedia.org/wiki/Computation_of_CRC)