Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή εργασία

Ανάπτυξη Δομικών Στοιχείων Δρομολογητών για Δίκτυα σε Τσιπ (Network on Chip - NoC)

Φίλιππος Γεώργιος Κολυμπιανάκης (ΑΜ: 2172)

Επιβλέπων καθηγητής : Κορνάρος Γεώργιος

Επιτροπή Αξιολόγησης :

Κορνάρος Γεώργιος, Βλησίδης Ανδρέας,
Λυμπινάκης Ανδρέας

Ημερομηνία παρουσίασης : 26/10/2012

Ευχαριστίες

Η παρούσα πτυχιακή εργασία εκπονήθηκε από το φοιτητή Κολυμπιανάκη Φίλιππο –Γεώργιο του Τμήματος Εφαρμοσμένης Πληροφορικής και Πολυμέσων του ΤΕΙ Ηρακλείου το ακαδημαϊκό έτος 2011 – 2012 υπό την επίβλεψη του καθηγητή εφαρμογών του τμήματος Κ. Γεώργιο Κορνάρο. Στον Κ. Κορνάρο οφείλω τις θερμές μου ευχαριστίες για την καθοδήγηση και την υποστήριξη του καθ ‘όλη τη διάρκεια διεκπεραίωσης της παρούσας πτυχιακής.

Abstract

In a few years it will be possible for designers to have more than 50 processors and memories of various types in a single chip. A new model for the design of such a system on chip is based upon Network-on-chip (NoC) where a network based on flexible routers is used for the communication between processors in the chip, using a fixed packet size. Very important issues arise in the designing of such scalable network structures. This thesis concerns the development of key components (routers) and communication interfaces for cores within a chip. The development will be done in VHDL and schematics while a test prototype of the communication system will be developed on an FPGA.

Σύνοψη

Σε λίγα χρόνια θα είναι εφικτό για τους σχεδιαστές να έχουν περισσότερους από 50 επεξεργαστές καθώς και μνήμες διαφόρων τύπων σε ένα μοναδικό τσιπ. Ένα νέο μοντέλο για την σχεδίαση τέτοιων συστημάτων σε τσιπ στηρίζεται σε Δίκτυα-σε-τσιπ (NoC) όπου ένα δίκτυο με βάση ευέλικτους δρομολογητές χρησιμοποιείται για την επικοινωνία των επεξεργαστών μέσα στο τσιπ, με χρήση πακέτων δεδομένων σταθερού μεγέθους. Πολύ σημαντικά θέματα δημιουργούνται για την σχεδίαση τέτοιων επεκτάσιμων δομών δικτύου. Η πτυχιακή εργασία αφορά την δημιουργία βασικών δομικών στοιχείων (δρομολογητών) επικοινωνίας και διασύνδεσης για πυρήνες μέσα σε ένα τσιπ. Η ανάπτυξη θα γίνει σε VHDL και σχηματικά και θα αναπτυχθεί δοκιμαστικό πρωτότυπο του συστήματος επικοινωνίας πάνω σε FPGA.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Λίστα Πινάκων

1 Εισαγωγή

Ένα ολοκληρωμένο Σύστημα σε Τσιπ (System on Chip - SoC) είναι, όπως υποδηλώνει το όνομα, ένα σύστημα το οποίο αποτελείτε από μικρότερα υποσυστήματα στο ίδιο το τσιπ. Τα υποσυστήματα αυτά μπορεί να είναι οτιδήποτε από μονάδες επεξεργασίας γενικού σκοπού μέχρι μονάδες μνήμης και διασυνδέσεις. Επειδή τα υποσυστήματα μπορούν να προσαρμοστούν είτε σε μια συγκεκριμένη εφαρμογή είτε σε μια πολύ γενική και ευρεία, τα SoC μπορούν να καλύψουν ένα μεγάλο φάσμα εφαρμογών και προϊόντων με χαμηλής ισχύος ενσωματωμένα συστήματα που έχουν πολύ αυστηρούς περιορισμούς σε έκταση και ενέργεια και μπορούν να χειριστούν ένα ευρύ φάσμα εφαρμογών.

Με την εισαγωγή υποσυστημάτων, τα μεμονωμένα εξαρτήματα μπορούν να σχεδιαστούν και να επαληθευτούν πάρα πολύ γρήγορα κάνοντας έτσι την διαδικασία δημιουργίας των SoC ευκολότερη. Επιπλέον, η προσέγγιση με SoC επιτρέπει την δημιουργία συστημάτων Intellectual Property Cores (IP Cores) και την επαναχρησιμοποίηση αυτών σε διαφορετικά SoC. Λόγο της μεγάλης επαναχρησιμοποίησης των IP Core μειώνεται σημαντικά ο χρόνος που απαιτείται για να κατασκευαστούν ακόμα και μεγάλα και πολύπλοκα SoC.

Η προσέγγιση SoC για το σχεδιασμό ενός συστήματος, επιτρέπει στο σχεδιαστή να συνδέσει πολλά IP Cores μέσω διασύνδεσης (interconnect) σε μια κοινή διεπαφή (interface). Σε πολλές περιπτώσεις η διασύνδεση αυτή είναι μια τοπολογία δίαυλου ή σημείου προς σημείο. Όσο αυξάνεται ο αριθμός των IP Core που είναι συνδεμένα σε ένα SoC τόσο αυξάνεται και η ανάγκη για μεγαλύτερη χωρητικότητα στους διαδρόμους (busses) επικοινωνίας. Για να διατηρηθεί ή ακόμα και να αυξηθεί η απόδοση της επικοινωνίας έχουν ερευνηθεί εναλλακτικές υποδομές διασύνδεσης σε ένα μεγάλο SoC.

Τα τελευταία χρόνια αυξάνεται συνεχώς το ενδιαφέρον για τα Δίκτυα σε Τσιπ (Network on Chip – NoC) ως προσέγγιση για το χειρισμό της on-chip επικοινωνίας στα μεγάλης κλίμακας SoC. Τα NoC παρέχουν λύσεις για την αυξημένη επικοινωνία που δημιουργείται από το μεγάλο αριθμό των IP cores που είναι συνδεδεμένα σε ένα SoC. Πολλές λύσεις μπορούν να υιοθετηθούν από έρευνες που έχουν ήδη γίνει στον τομέα των δικτύων μεγάλης κλίμακας, όπως του διαδικτύου. Νέες προκλήσεις και δυνατότητες προκύπτουν από τα NoC, όπως είναι το εύρος ζώνης και η καθυστέρηση επικοινωνίας μεταξύ των πυρήνων IP. Καθώς όλοι οι πυρήνες IP είναι συνδεμένοι σε δίκτυο, το NoC επιτρέπει λογικές συνδέσεις σημείου προς σημείο μεταξύ των πυρήνων αυξάνοντας έτσι την ευελιξία του SoC.

1.1 Περίληψη

Λόγο της ανάπτυξης στον τομέα ερευνάς των NoC, έχουν διερευνηθεί διάφορες πτυχές σχεδίασης τόσο στον ακαδημαϊκό κόσμο όσο και στη βιομηχανία. Η έννοια των NoC μοιράζεται πολλές έννοιες και λειτουργιές με μεγάλης κλίμακας δίκτυα όπως εκείνα που έχουν δημιουργηθεί για την επικοινωνία μεταξύ υπολογιστών. Μια από τις μεγάλες διαφορές είναι ότι ένα NoC είναι στατικό, σε αντίθεση με τα περισσότερα δίκτυα μεγάλης κλίμακας που έχουν να αντιμετωπίσουν συνεχώς μεταβαλλόμενα δρομολόγια και διαφορετικό αριθμό συνδεδεμένων χρηστών. Αυτό επηρεάζει τις επιλογές σχεδιασμού, όπως το πρωτόκολλο μετάδοσης και ο έλεγχος ροής. Επιπλέον, επειδή ένα NoC είναι ένα στενά συνδεδεμένο περιβάλλον η απώλεια και η αλλοίωση των δεδομένων μπορεί στις περισσότερες περιπτώσεις να αγνοηθεί.

Στην πτυχιακή αυτή θα αναλύσουμε βασικές έννοιες και θα εξετάσουμε τα ιδιαίτερα χαρακτηριστικά των NoC. Στην συνεχεία θα αναφερθούμε τοπολογίες και πρωτόκολλα καθώς και το τρόπο σχεδίασης των NoC. Τέλος θα αναπτυχτεί ένα ολοκληρωμένο NoC Router επικοινωνία μεταξύ πέντε θυρών εισόδου και πέντε θυρών εξόδου όπου και θα εξεταστεί ο τρόπος λειτουργίας του. Μέσω προσομοιώσεων λειτουργιάς και συλλογής αποτελεσμάτων θα μπορούμε να είμαστε σε θέση να αξιολογήσουμε το NoC που κατασκευάστηκε όπως επίσης και την ποιότητα των υπηρεσιών που παρέχει (Quality of Service – QoS).

1.2 Κίνητρο για την Διεξαγωγή της Εργασίας

Τα NoC είναι ένα αναδυόμενο παράδειγμα για την επικοινωνία σε μεγάλα συστήματα VLSI (Very large scale integration - Πολύ μεγάλης κλίμακας ενσωμάτωση) που υλοποιούνται σε ένα τσιπ πυριτίου. Σε ένα σύστημα NoC, ενότητες όπως πυρήνες επεξεργαστή, μνήμες και εξειδικευμένα IP blocks ανταλλάσουν δεδομένα χρησιμοποιώντας ένα δίκτυο ως "μέσο μαζικής μεταφοράς". Ένα NoC είναι κατασκευασμένο από πολλαπλές συνδέσεις σημείου προς σημείο οι οποίες διασυνδέονται με μεταγωγείς (switches) ή διαφορετικά με δρομολογητές (routers) έτσι ώστε τα μηνύματα να αναμεταδίδονται από οποιαδήποτε μονάδα προέλευσης σε οποιαδήποτε μονάδα προορισμού παίρνοντας αποφάσεις δρομολόγησης.

Ένα NoC είναι παρόμοιο με ένα σύγχρονο δίκτυο τηλεπικοινωνιών, κάνοντας χρήση μεταγωγής πακέτων πάνω από πολυπλεκτικές συνδέσεις. Τα καλώδια στους συνδέσμους ενός NoC μοιράζονται πολλά σήματα. Ένα υψηλό επίπεδο παραλληλισμού επιτυγχάνεται, διότι όλες οι συνδέσεις του NoC μπορούν να λειτουργούν ταυτόχρονα με διαφορετικά πακέτα δεδομένων. Κατά συνέπεια, όπως η πολυπλοκότητα των ολοκληρωμένων συστημάτων συνεχίζει να αυξάνεται, ένα NoC προσφέρει βελτιωμένες επιδόσεις (όπως throughput) και επεκτασιμότητα σε σύγκριση με τις προηγούμενες αρχιτεκτονικές επικοινωνίας (π.χ. ειδικά καλώδια σήματος σημείου προς σημείο, διαμοιραζόμενους διαδρόμους και διαδρόμους με γέφυρες). Φυσικά, οι αλγόριθμοι πρέπει να σχεδιάζονται με τέτοιο τρόπο ώστε να προσφέρουν μεγάλο παραλληλισμό και κατά συνέπεια να μπορούν να χρησιμοποιήσουν το δυναμικό του NoC.

Παραδοσιακά, τα ολοκληρωμένα κυκλώματα έχουν σχεδιαστεί με ειδικές συνδέσεις σημείου προς σημείο, ένα καλώδιο αφιερωμένο σε κάθε σήμα. Για μεγάλα σχέδια, από φυσικής άποψης, αυτό έχει αρκετούς περιορισμούς. Τα καλώδια καταλαμβάνουν μεγάλο μέρος της περιοχής του τσιπ, οι διασυνδέσεις ρίχνουν την απόδοση και καταναλώνουν περισσότερη ισχύ ενώ η διάδοση του σήματος σε όλα τα καλώδια του τσιπ απαιτεί αρκετούς κύκλους ρολογιού.

Οι συνδέσεις με NoC μπορούν να αυξήσουν την ταχύτητα, και την αξιοπιστία μειώνοντας παράλληλα την πολυπλοκότητα, το θόρυβο και την ενέργεια κατανάλωσης εξαιτίας της καλά ελεγχόμενης δομής τους. Από την άποψη του σχεδιασμού του συστήματος, με την εμφάνιση των συστημάτων με πολυπύρηνους επεξεργαστές, ένα δίκτυο μέσα στο τσιπ είναι μια φυσική αρχιτεκτονική επιλογή. Το NoC μπορεί να προσφέρει διαχωρισμό μεταξύ υπολογισμών και επικοινωνίας, την επαναχρησιμοποίηση των IP πυρήνων, μπορεί να χειριστεί θέματα συγχρονισμού, να χρησιμεύσει ως πλατφόρμα για δόκιμη συστήματος, και, συνεπώς, να αυξήσει την μηχανική παραγωγικότητα.

1.3 Σκοπός και Στόχοι Εργασίας

Οι στόχοι αυτής της εργασίας είναι οι εξής:

- Ανάπτυξη πρωτότυπου Network on Chip Router για επικοινωνία μεταξύ πέντε θυρών εισόδου και πέντε θυρών εξόδου.
- Υψηλού επίπεδου λειτουργικότητα, με όσο το δυνατόν μικρότερες καθυστερήσεις και χωρίς απώλειες πακέτων.
- Ανάπτυξη απαραιτήτων μονάδων για προσομοίωση του NoC που κατασκευάστηκε στο modelsim 6.5 και στο Spartan 3 Starter kit board.
- Παρουσίαση της λειτουργιάς του NoC Router 5x5 στο Spartan 3 Starter kit board.

1.4 Δομή εργασίας

Το υπόλοιπο αυτής της εργασίας είναι οργανωμένο ως εξής:

- Στο κεφάλαιο 2 δίνουμε μια σύντομη περιγραφή για τον τρόπο με τον οποίο θα αναπτυχτεί το NoC router καθώς επίσης και σε ποιες τεχνολογίες θα βασιστεί η κατασκευή του.

- Στο κεφάλαιο 3 κάνουμε μια αναφορά στις τελευταίες εξελίξεις της τεχνολογίας για την κατασκευή των NoC router στις οποίες θα βασιστούμε ώστε να κατασκευάσουμε ένα δικό μας πρωτότυπο.
- Στο κεφάλαιο 4 παρουσιάζουμε τα εργαλεία που θα χρησιμοποιήσουμε για την κατασκευή του NoC όπως επίσης και τα source files που απαιτούνται για την ολοκλήρωση του.
- Στο κεφάλαιο 5 που είναι και το κύριο μέρος αυτής της πτυχιακής εργασίας περιγράφεται αναλυτικά κάθε ένα από τα source files που αναπτύχτηκαν, καθώς επίσης και ο τρόπος με τον οποίο ενώθηκαν για τη δημιουργία ενός ολοκληρωμένου κυκλώματος.
- Στο κεφάλαιο 6 παρουσιάζονται τα αποτελέσματα και γίνεται αξιολόγηση του NoC router που κατασκευάστηκε. Τέλος, δίνουμε συμπεράσματα μας καθώς επίσης και κατευθύνσεις για μελλοντικές επεκτάσεις.

2 Μεθοδολογία Υλοποίησης

Στην παρούσα πτυχιακή εργασία θα αναπτυχτεί το πρωτότυπο ενός NoC router για την επικοινωνία μεταξύ πέντε θυρών εισόδου και πέντε θυρών εξόδου. Το NoC που θα κατασκευαστεί θα πρέπει να παρέχει υψηλού επίπεδου υπηρεσίες με όσο το δυνατόν μικρότερες καθυστερήσεις στην δρομολόγηση και χωρίς απώλειες πακέτων. Για την ανάπτυξη του NoC θα χρησιμοποιήσουμε το λογισμικό ISE 12.1 της εταιρίας Xilinx και για την προσομοίωση λειτουργίας του το modelsim 6.5. Το δοκιμαστικό πρωτότυπο θα υλοποιηθεί στο FPGA του Spartan 3 Starter kit board.

2.1 Μέθοδος Ανάλυσης & Ανάπτυξης Πτυχιακής

Το NoC router θα αναπτυχτεί με τη βοήθεια του λογισμικού ISE 12.1 της εταιρίας Xilinx. Το NoC θα διαθέτει πέντε θύρες εισόδου πακέτων, τα όποια θα δρομολογούνται σε πέντε θύρες εξόδου. Επίσης, θα διαθέτει πέντε buffers (τύπου FIFO) για την προσωρινή αποθήκευση των πακέτων. Κάθε θύρα δεν διαθέτει το προσωπικό της buffer αλλά μπορεί να προωθήσει τα πακέτα σε οποιοδήποτε από τα διαθέσιμα, αρκεί αυτό να μην χρησιμοποιείται εκείνη τη χρονική στιγμή από κάποιου άλλη θύρα και να μην έχει γεμίσει με πακέτα προς μετάδοση το συγκεκριμένο buffer. Από τα buffer τα πακέτα προωθούνται στην θύρα εξόδου χρησιμοποιώντας ένα αλγόριθμο δρομολόγησης, όπως θα αναφέρουμε παρακάτω. Τέλος, θα αναπτυχτούν ακόμα κάποιες μονάδες που κατασκευάζουν τυχαία πακέτα για κάθε θύρα εισόδου, καθώς επίσης και μια μονάδα για να μπορέσουμε να απεικονίζουμε τα πακέτα σε μια οθόνη επτά τμημάτων που διαθέτει το Spartan 3 Starter kit board.

2.1.1 Αλγόριθμοι

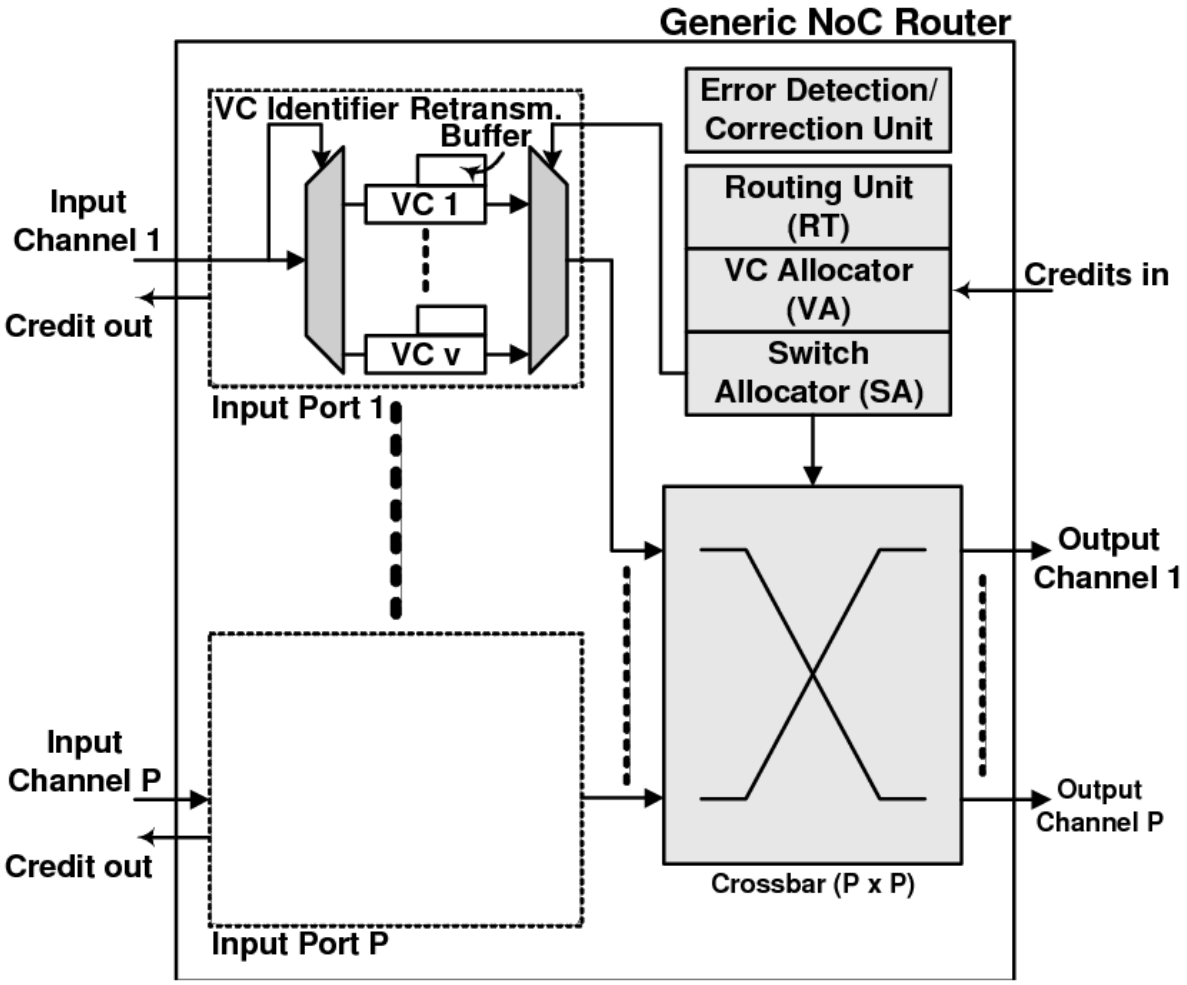
Για τη σωστή επικοινωνία στο εσωτερικό του NoC router χρησιμοποιούμε δυο αλγόριθμους δρομολόγησης. Ο πρώτος εκτελείτε μεταξύ των θυρών εισόδου και των buffers και ασχολείται με την κατάσταση των FIFO, δηλαδή ποιες από αυτές είναι απασχολημένες, πότε και αν κάποια από αυτές είναι άδεια ή γεμάτη και αν μπορεί να γραφτεί κάτι σε αυτές. Ο δεύτερος εκτελείτε μεταξύ των buffers και των θυρών εξόδου και ασχολείται με την προώθηση των πακέτων από τις FIFO προς τις θύρες εξόδου. Συνεπώς, αυτός είναι ο αλγόριθμος δρομολόγησης που αποφασίζει πότε (χρονικά) και από ποια FIFO θα γίνει ανάγνωση πακέτου και προώθηση του πακέτου αυτού στη θύρα εξόδου. Ο αλγόριθμος αυτός στηρίζει τη λειτουργία του στον **αλγόριθμο χρονοπρογραμματισμού εξυπηρέτησης εκ περιτροπής** (**Round - Robin scheduling algorithm**). Ο αλγόριθμος αυτός επιλέχτηκε επειδή είναι ένας από τους παλαιότερους, πιο δίκαιους και πιο διαδεδομένους αλγορίθμους χρονοπρογραμματισμού.

2.1.2 Θεωρίες

Η λογική του **Round - Robin scheduling** είναι απλή. Σε κάθε FIFO δίνεται ο ίδιος χρόνος για ανάγνωση πακέτων διάρκειας ίσης με το χρόνο ανάγνωσης ενός πακέτου (4 clock cycles). Με μια μηχανή καταστάσεων τύπου Moore ο arbiter ‘’τσεκάρει’’ αν μια FIFO έχει πακέτα προς προσπέλαση, αν αυτό ισχύει τότε θα προωθήσει στην έξοδο ένα πακέτο και θα προχωρήσει στην επόμενη FIFO κάνοντας την ίδια εργασία. Αυτό γίνεται και για τις πέντε FIFO αντιμετωπίζοντας τις όλες το ίδιο (ίδια δικαιώματα). Ο αλγόριθμος δεν προκαλεί στέρηση (starvation) και δεν υπάρχουν επικαλύψεις ή απώλειες πακέτων.

2.1.3 Μοντέλα

Η μεθοδολογία κατασκευής του NoC router θα βασίζεται στους αλγόριθμους δρομολόγησης και σε γενικές αρχιτεκτονικές σχεδίασης που περιγράφονται παραπάνω. Στόχος μας είναι η κατασκευή ενός ολοκληρωμένου συστήματος το οποίο θα παρέχει επικοινωνία μεταξύ πέντε θυρών εισόδου και πέντε θυρών εξόδου χωρίς απώλειες πακέτων και με όσο το δυνατόν μικρότερες καθυστερήσεις.



Εικόνα 2.1: Γενική Αρχιτεκτονική NoC Router

3 Σχέδιο Δράσης για την εκπόνηση της εργασίας

Η υλοποίηση του NoC router θα πραγματοποιηθεί ακλουθώντας τις τελευταίες εξελίξεις της τεχνολογίας στην κατασκευή ολοκληρωμένων συστημάτων για επικοινωνία μέσα σε ένα τσιπ, όπως περιγράφεται στο παρακάτω υποκεφάλαιο State of the Art.

3.1 State of the Art

Ένα τυπικό NoC router είναι υπεύθυνο για τη μετακίνηση των πακέτων που λαμβάνονται από τα buffer εισόδου, συμφώνα τους αλγόριθμους διαιτησίας (arbitration algorithms) και δρομολόγησης (routing algorithms), στις θύρες εξόδου. Οι αποφάσεις που παίρνει ο δρομολογητής (router) βασίζονται σε πληροφορίες που συλλέγονται από το δίκτυο. Συγκεντρωτικές αποφάσεις αναφέρονται στη λήψη αποφάσεων με βάση τις πληροφορίες που συγκεντρώθηκαν από ολόκληρο το δίκτυο. Κατανεμημένες αποφάσεις αναφέρονται στη λήψη αποφάσεων με βάση τα στοιχεία που προκύπτουν από τον τοπικό δρομολογητή ή τους κοντινούς δρομολογητές. Η κατανεμημένη δρομολόγηση επιτρέπει στο NoC να μεγαλώσει σε μέγεθος, χωρίς να χρειάζεται να ανησυχούμε για την αυξανομένη πολυπλοκότητα μέσα σε μια συγκεντρωτική μονάδα δρομολόγησης. Μια κεντρική δρομολόγηση εξαρτάται από αποφάσεις δρομολόγησης που λαμβάνονται από συγκεντρωτικές πληροφορίες, ως εκ τούτου, χρειάζεται περισσότερα buffers, πινάκες δρομολόγησης και μηχανισμούς διαιτησίας σε κάθε κόμβο.

Υπάρχουν κάποιοι κατανεμημένοι αλγόριθμοι δρομολόγησης που στηρίζονται μόνο σε τοπικές πληροφορίες. Οι αλγόριθμοι αυτοί έχουν προταθεί ως πάρα πολύ αποτελεσματικοί παρόλο που εξακολουθούν να διατηρούν χαμηλή επιβάρυνση και υψηλή επεκτασιμότητα. Οι αλγόριθμοι αυτής της κατηγορίας περιλαμβάνουν προσδιοριστικούς (deterministic) και προσαρμοστικούς (adaptive) αλγόριθμους. Συμφώνα με ρεαλιστικά μοντέλα κίνησης όπου τίθεται το πρόβλημα μεγάλου φόρτου κυκλοφορίας σε κάποια σημεία, η προσδιοριστική δρομολόγηση απέτυχε να απαγάγει τα σημεία αυτά οδηγώντας έτσι σε υψηλό μέσο όρο καθυστερήσεων. Η προσαρμοστική δρομολόγηση οδηγεί το router να αντιδράσει στα σημεία όπου δημιουργείται μεγάλος φόρτος, από διαφορά σχέδια κυκλοφορίας, επιτρέποντας σε ένα πακέτο στο buffer εισόδου να ζητήσει περισσότερες από μια θύρες εξόδου ή κατευθύνσεις.

Οι γειτονικοί κόμβοι στέλνουν ενδείξεις για χρήση προσαρμοστικής δρομολόγησης όταν τα buffer τους διαθέτουν παραπάνω πακέτα από ένα προκαθορισμένο όριο. Υπό αυτές τις συνθήκες ο δρομολογητής επιβάλει στα πακέτα να δρομολογούνται προς την κατεύθυνση με τις περισσότερες διαθέσιμες υποδοχές buffer. Ο προσαρμοστικός αυτός αλγόριθμος χρησιμοποιείται με την παρουσία αυξημένου φόρτου κυκλοφορίας. Με την έγκαιρη ανίχνευση του επίπεδου συμπλήρωσης των buffer μπορεί να αποφευχθεί καλυτέρα η συμφόρηση. Εκτός από τη δρομολόγηση με ανίχνευση του επιπέδου συμπλήρωσης των buffer, το άλλο μέρος λήψης αποφάσεων του router είναι η διαιτησία των πακέτων. Όταν πολλαπλά πακέτα εισόδου προορίζονται σε ίδιες θύρες εξόδου , αλγόριθμοι δρομολόγησης και διαιτησίας όπως ο round-robin και ο first-come-first-serve έχουν προταθεί για την επίλυση μεγάλης συμφόρησης.

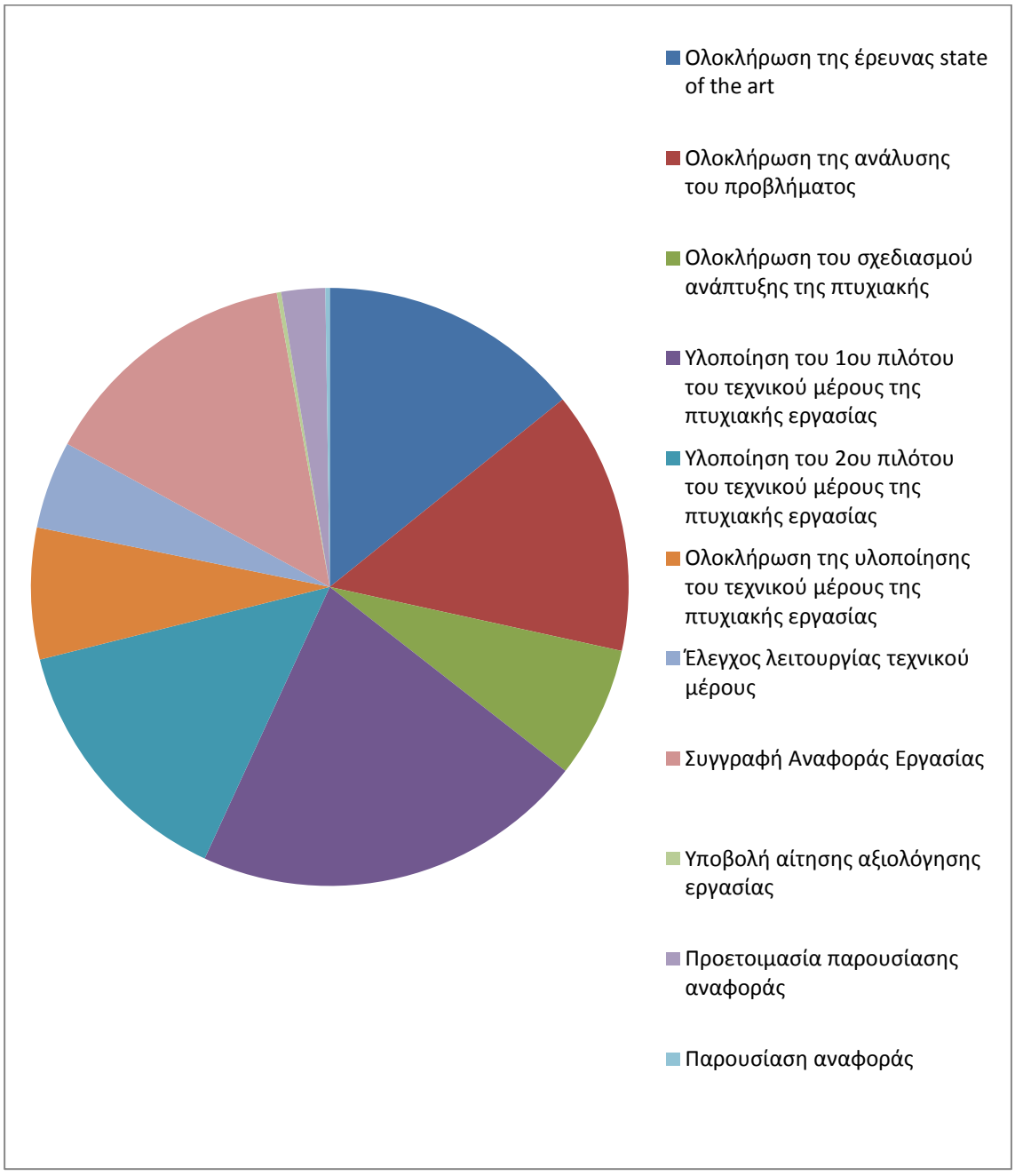
Μερικές φορές κατηγοριοποιούμε τους δυο παραπάνω αλγόριθμους ως αλγόριθμους αποφυγής συμφόρησης. Με αλλά αποφεύγεται να αποστέλλονται πακέτα προς την κυκλοφοριακή συμφόρηση έτσι ώστε να μην επιδεινώνεται. Τα τελευταία χρόνια έχουν προταθεί περισσότερα έργα και νέοι αλγόριθμοι για την αντιμετώπιση του μεγάλου φόρτου κυκλοφορίας και της σύνθετης δρομολόγησης, παρ’ όλα αυτά δεν θα αναφερθούμε σε περεταίρω τεχνολογίες καθώς επιλέξαμε τον Round Robin ως αλγόριθμο δρομολόγησης της παρούσας πτυχιακής εργασίας.

3.2 Σημαντικοί Στόχοι Για την Ολοκλήρωση της Πτυχιακής

Ολοκλήρωση της έρευνας state of the art	60
Ολοκλήρωση της ανάλυσης του προβλήματος	60
Ολοκλήρωση του σχεδιασμού ανάπτυξης της πτυχιακής	30
Υλοποίηση του 1ου πιλότου του τεχνικού μέρους της πτυχιακής εργασίας	90

Υλοποίηση του 2ου πιλότου του τεχνικού μέρους της πτυχιακής εργασίας	60
Ολοκλήρωση της υλοποίησης του τεχνικού μέρους της πτυχιακής εργασίας	30
Έλεγχος λειτουργίας τεχνικού μέρους	20
Συγγραφή Αναφοράς Εργασίας	60
Υποβολή αίτησης αξιολόγησης εργασίας	1
Προετοιμασία παρουσίασης αναφοράς	10
Παρουσίαση αναφοράς	1

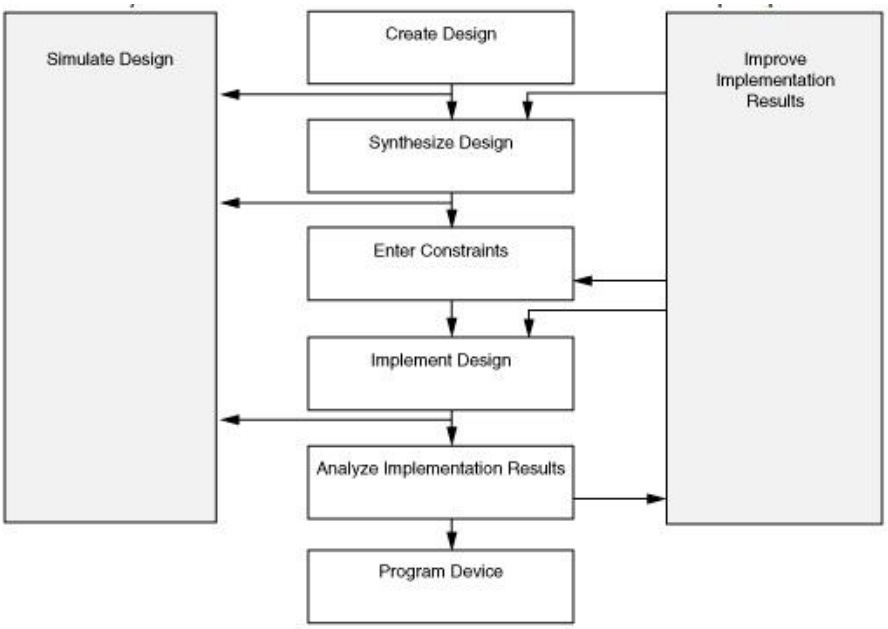
3.3 Χρονοδιάγραμμα Εργασίας



Εικόνα 3.1: Χρονοδιάγραμμα Εργασίας

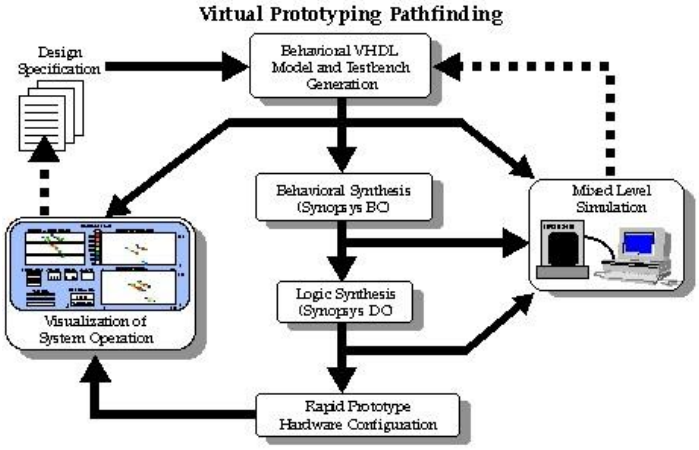
4 Περιβάλλον Σχεδίασης – Εργαλεία Προσομοίωσης

Για την ανάπτυξη του NoC router χρησιμοποιούμε ένα πρόγραμμα σχεδίασης, το **ολοκληρωμένο περιβάλλον λογισμικού (ISE - Integrated Software Environment)** της εταιρίας Xilinx έκδοση 12.1, ένα πρόγραμμα προσομοίωσης, το Modelsim 6.5, καθώς και το Spartan 3 starter kit board για την προσομοίωση της σωστής λειτουργίας του κυκλώματος . Το project αναπτύχτηκε με μια **γλώσσα περιγραφής υλικού (hardware description language)**, την VHDL, την οποία εισάγουμε στον επεξεργαστή κειμένου του ISE. Το project στη συνέχεια γίνεται σύνθεση (synthesis), προσομοίωση (simulation) στο modelsim και τοποθέτηση (placement) στο FPGA του Spartan 3 SE .



Εικόνα 4.1: Ροή Σχεδίασης

Μετά τη σχεδίαση και τη συγγραφή του κώδικα θα γίνει **σύνθεση (synthesize)** του κώδικα αυτού και είναι το αντίστοιχο του μεταγλωττιστή (compile) για τις κοινές γλώσσες προγραμματισμού. Ο synthesizer μετατρέπει το VHDL κώδικα σε λογικές πύλες επιπέδου netlist. Από προεπιλογή το ISE της Xilinx χρησιμοποίει τον ενσωματωμένο synthesizer XST και οι λογικές πύλες net list αποθηκεύονται σε NGC μορφή. Στο σημείο αυτό μπορεί να γίνει προσομοίωση λειτουργίας με το modelsim για να διαπιστωθεί αν λειτουργεί σύμφωνα με τις απαιτήσεις που έχουν καθοριστεί.

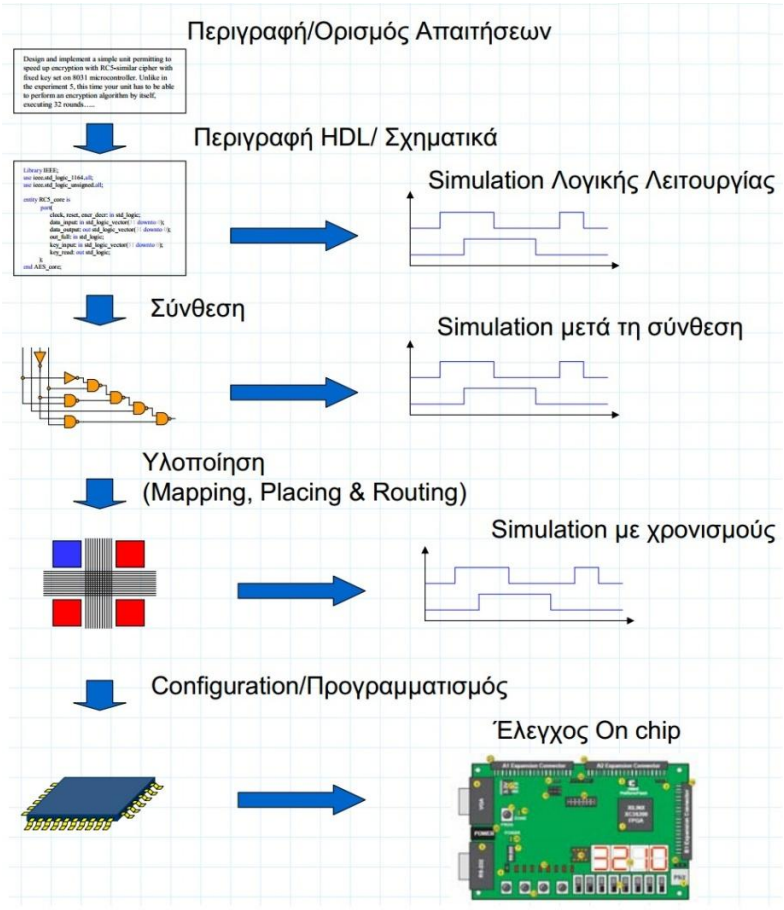


Εικόνα 4.2: Μεθοδολογία Σύνθεσης

Με την επιτυχημένη ολοκλήρωση της σύνθεσης έρχεται το στάδιο της **υλοποίησης (Implementation)** στο οποίο οι λογικές πύλες net list μεταφράζονται σε ένα σχέδιο FPGA. Το στάδιο αυτό έχει τρία βήματα:

- **Μετάφραση (Translate)**
  Στη φάση της μετάφρασης οι NGC net list μετατρέπονται σε NGD net list. Η διαφορά τους είναι ότι τα NGC net list βασίζονται σε UNISIM βιβλιοθήκες, σχεδιασμένα για προσομοίωση συμπεριφοράς (behavioural simulation), ενώ τα NGD net list βασίζονται σε SIMPRIM βιβλιοθήκες.
- **Χαρτογράφηση (Mapping)**
  Στη φάση της χαρτογράφησης τα NGD net list αντιστοιχίζονται σε συγκεκριμένους πόρους συσκευής (device resources), όπως LUTs, flip-flops, BRAMs κ.α. , δηλαδή στοιχειά λογικής για το FPGA.
- **Τοποθέτηση και Δρομολόγηση (Place & Route)**
  Η φάση της τοποθέτησης και της δρομολόγησης είναι το ποιο σημαντικό και χρονοβόρο βήμα της υλοποίησης κατά το οποίο ορίζεται ο τρόπος με τον οποίο οι πόροι συσκευής (device resources) βρίσκονται και διασυνδέονται μέσα στο FPGA. Η τοποθέτηση της λογικής σε συγκεκριμένα λογικά Blocks μέσα σε ένα FPGA πρέπει να είναι τέτοια ώστε οι καθυστερήσεις (wiring delay) να είναι αποδεκτές. Η τοποθέτηση είναι ακόμη ποιο σημαντική από τη δρομολόγηση, επειδή μια λάθος τοποθέτηση θα καταστήσει αδύνατη μια σωστή δρομολόγηση.

Μετά από μια ολοκληρωμένη υλοποίηση το επόμενο στάδιο είναι η **δημιουργία του αρχείου προγραμματισμού (Generate Programming File)** κατά το οποίο τρέχει το BitGen, ένα πρόγραμμα παραγωγής bit stream, που δημιουργεί ένα αρχείο bitstream (*.bit) το οποίο μετά θα τοποθετηθεί στη συσκευή προορισμού (FPGA) Spartan 3 Starter kit.

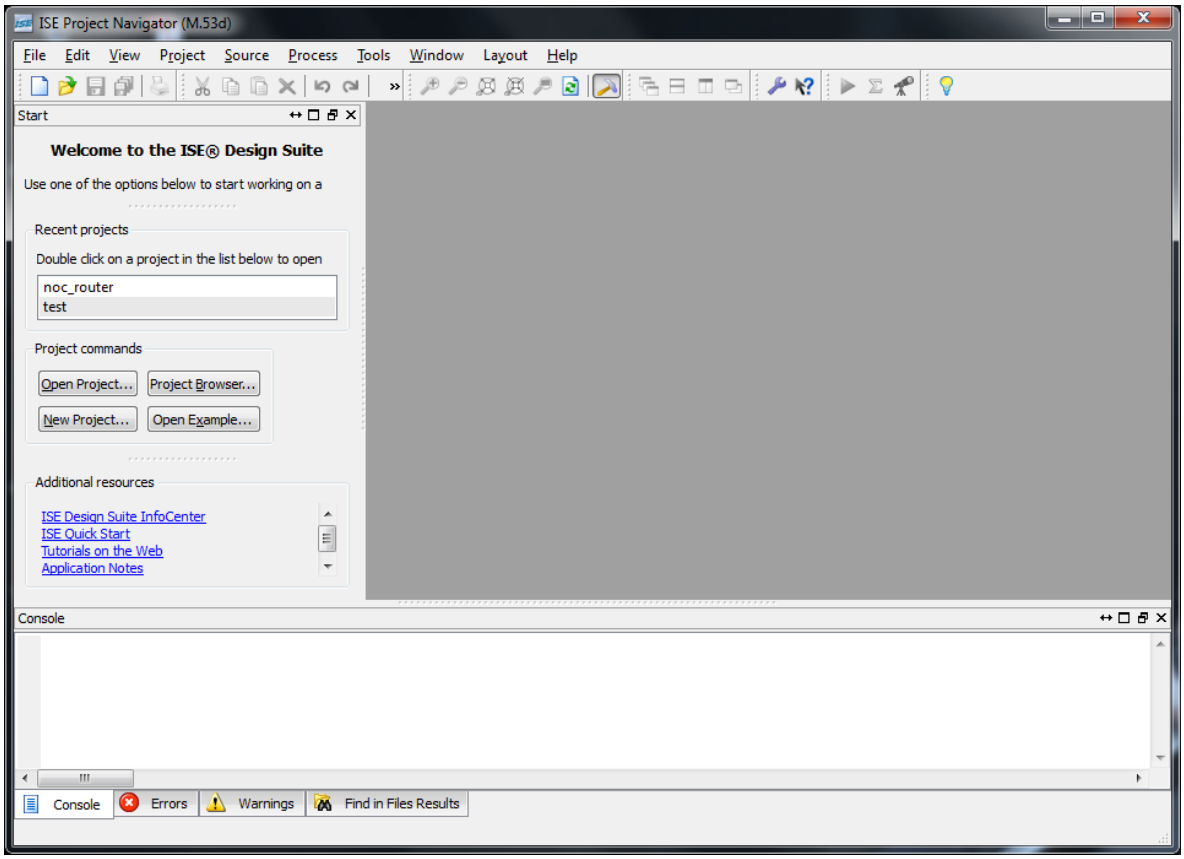


Εικόνα 4.3: Στάδια ολοκλήρωσης project

4.1 Xilinx ISE 12.1

Το **Xilinx ISE (Integrated Software Environment)** είναι ένα εργαλείο λογισμικού που παράγεται από την εταιρία Xilinx για τη σύνθεση και την ανάλυση σχεδίων HDL, που επιτρέπει στον προγραμματιστή να συνθέσει (synthesize – compile) τα σχέδια του, να εκτελέσει την ανάλυση χρονισμού, να εξετάζει RTL διαγράμματα, να προσομοιώσει την αντίδραση ενός σχεδίου σε διαφορετικά σενάρια και να ρυθμίσει με προγραμματισμό τη συσκευή προορισμού (FPGA) Spartan 3 Starter kit.

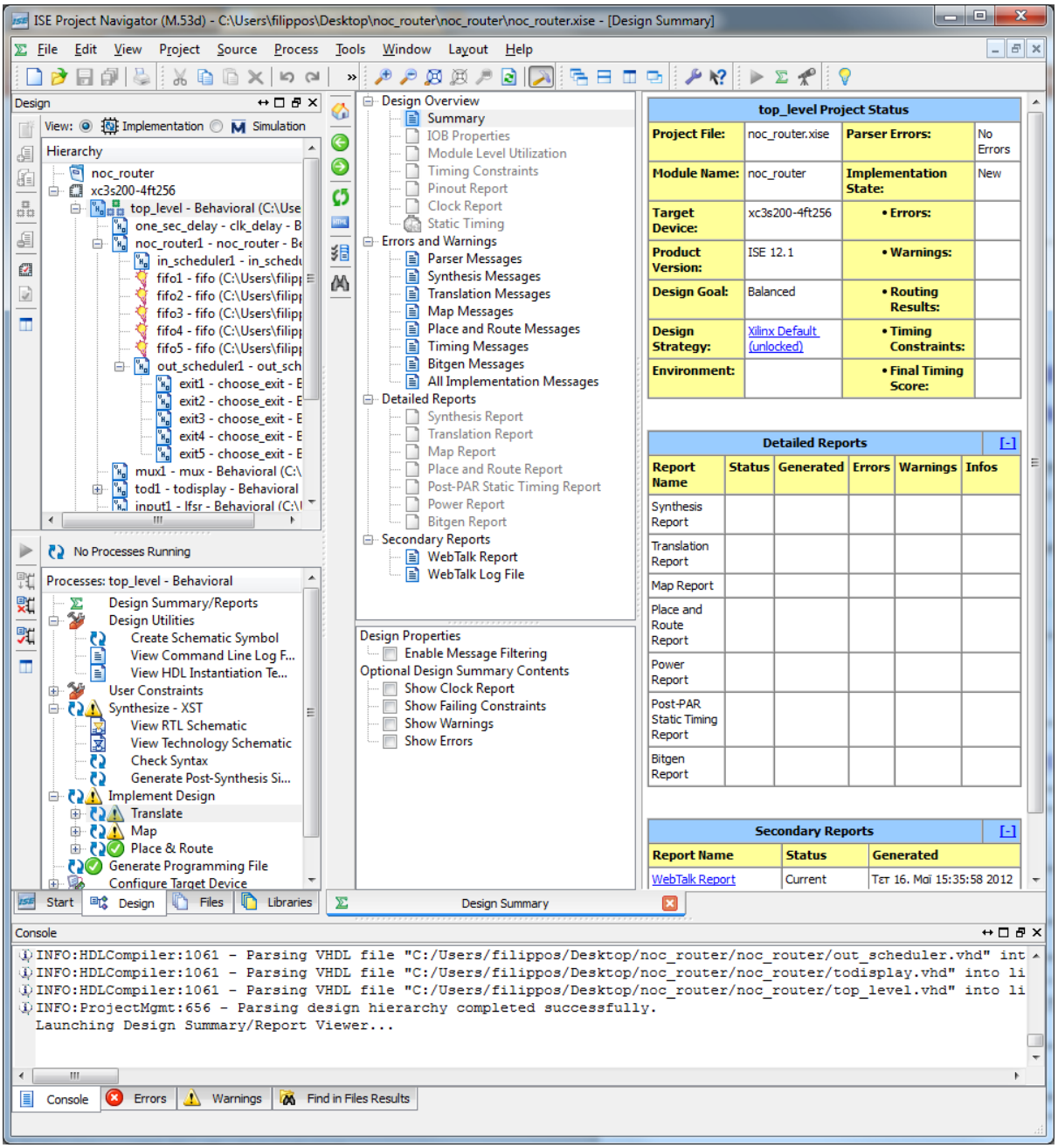
Το λογισμικό ISE ελέγχει όλες τις πτυχές της ροής του σχεδιασμού. Μέσω του περιβάλλοντος του προγράμματος πλοήγησης (Project Navigator) μπορούμε να αποκτήσουμε πρόσβαση σε όλα τα design entry και τα design implementation tools. Επίσης, μπορούμε να έχουμε πρόσβαση σε αρχεία και έγγραφα που σχετίζονται με το project μας.



Εικόνα 4.4: Περιβάλλον Σχεδίασης Xilinx ISE 12.1 – Βασικό παράθυρο του Project Navigator

Εξ' ορισμού, το περιβάλλον του Project Navigator χωρίζεται σε τέσσερα επιμέρους παράθυρα. Πάνω αριστερά είναι τα πάνελ Start, Design, Files και Libraries, τα οποία περιλαμβάνουν προβολή και πρόσβαση στα αρχεία προέλευσης του project καθώς και πρόσβαση σε τρεχούμενες διεργασίες (Processes) για την επιλεγμένη πηγή. Το πάνελ Start παρέχει γρήγορη πρόσβαση για άνοιγμα project καθώς και συχνή πρόσβαση σε υλικό αναφοράς, έγγραφα και tutorials. Στο κάτω μέρος του Project Navigator είναι τα πάνελ Console, Errors και Warnings, τα οποία εμφανίζουν μηνύματα κατάστασης, λάθη και προειδοποιήσεις.

Δεξιά του Project Navigator υπάρχει ένα Multidocument Interface (MDI) παράθυρο αποκαλούμενο ως χώρος εργασίας (Workspace). Ο χώρος εργασίας δίνει τη δυνατότητα να προβάλουμε αναφορές των project, αρχεία κειμένου, σχηματικά και κυματομορφές προσομοίωσης. Κάθε παράθυρο μπορεί να αλλάζει μέγεθος, να αποδεσμευτεί από το Project Navigator, να μεταφερθεί σε νέα θέση μέσα στο κύριο παράθυρο του Project Navigator ή να κλείσει.



Εικόνα 4.5: Project Navigator

4.1.1 Εκκίνηση Λογισμικού ISE

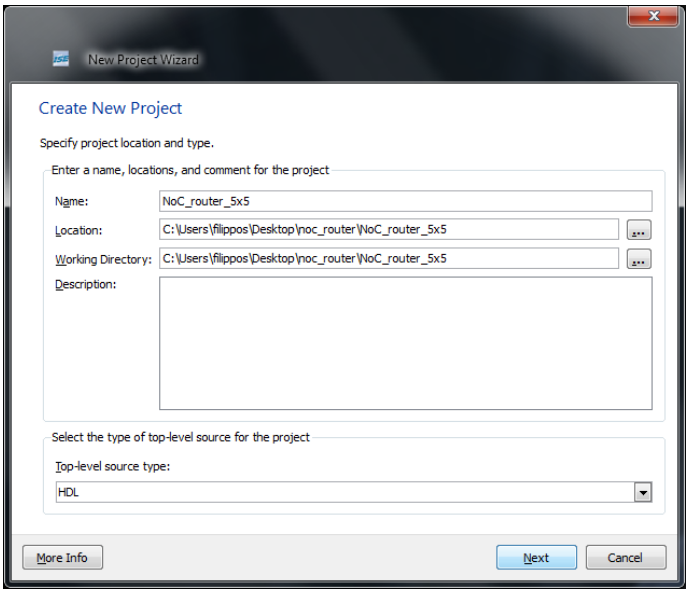
Για να ξεκινήσει το λογισμικό ISE, κάνουμε διπλό κλικ στο εικονίδιο "ISE Project Navigator" ή διαφορετικά ακλουθούμε τη διαδρομή **Έναρξη > Προγράμματα > Xilinx ISE Design Suite 12.1 > ISE Design Tools > Project Navigator**.

Εικόνα 4.6: Εικονίδιο Project Navigator

4.1.2 Δημιουργία Νέου Project

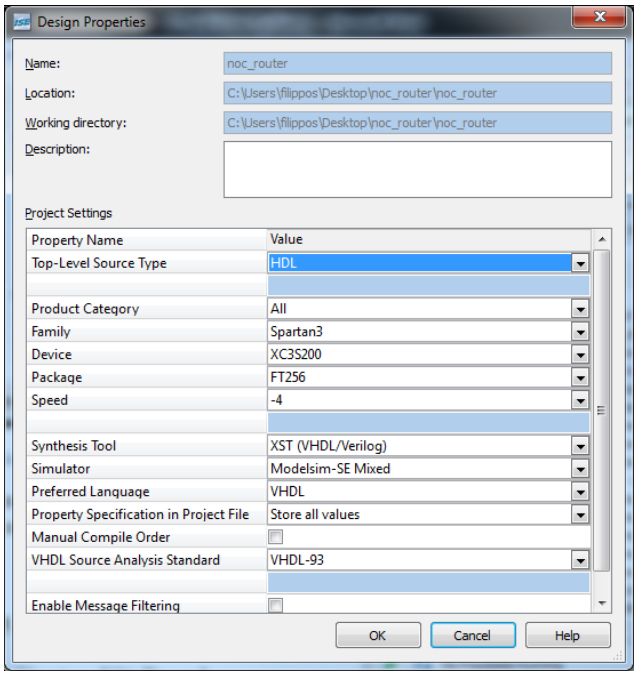
Για τη δημιουργία νέου project χρησιμοποιώντας τον οδηγό δημιουργίας νέου προγράμματος (New Project Wizard) κάνουμε τα εξής:

1. Από το Project Navigator, επιλέγουμε **File > New Project**. Εμφανίζεται ο οδηγός δημιουργίας νέου προγράμματος



Εικόνα 4.7: Οδηγός δημιουργίας νέου Προγράμματος (New Project Wizard)

2. Στο πεδίο **Name**, βάζουμε το όνομα που θέλουμε να έχει το project.
3. Στο πεδίο Location, βάζουμε την τοποθεσία στην οποία θέλουμε να αποθηκεύσουμε το project.
4. Στην επιλογή Top-**level source type**, επιλέγουμε **HDL** και πατάμε **Next**. Αμέσως μετά εμφανίζεται το παράθυρο των ιδιοτήτων της συσκευής.



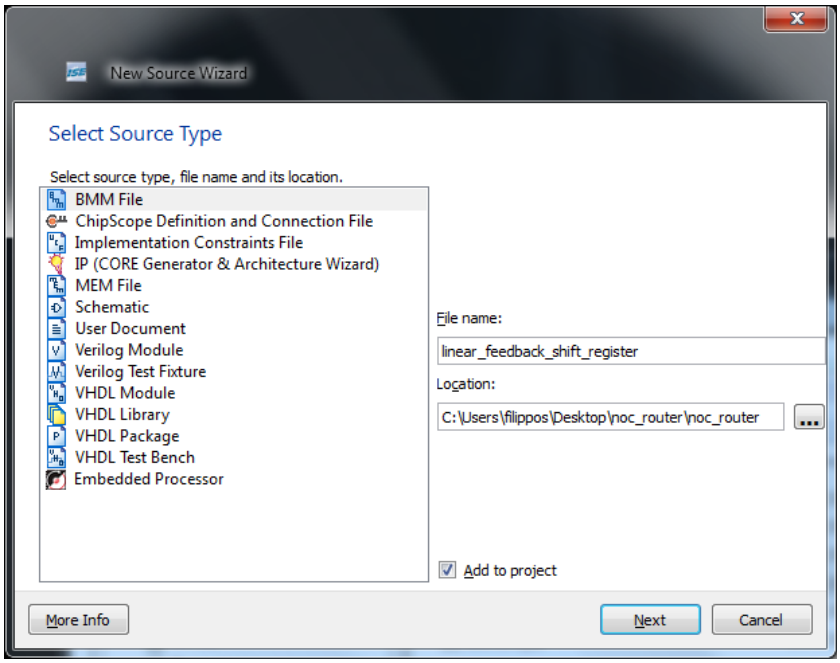
Εικόνα 4.8: Οδηγός δημιουργίας νέου Προγράμματος – παράθυρο ιδιοτήτων συσκευής

5. Στο παράθυρο ιδιοτήτων επειδή θα χρησιμοποιήσουμε το Spartan 3 Starter Kit Board της Xilinx, σαν γλώσσα περιγραφής υλικού την VHDL και το πρόγραμμα προσομοίωσης Modelsim επιλέγουμε τα παραπάνω όπως φαίνεται στην εικόνα 3.7 και πατάμε **OK**.
6. Στην συνέχεια πατάμε **Finish** για να ολοκληρωθεί η δημιουργία του project.

4.1.3 Αρχεία Προέλευσης (Source Files)

Για την υλοποίηση ενός project, στο οποίο θα μπορεί να γίνει σύνθεση, υπάρχουν διάφοροι τρόποι προέλευσης η δημιουργίας αρχείων προέλευσης (Source Files). Το Πρόγραμμα Προέλευσης (Project Navigator) μας επιτρέπει να προστεθεί στο project ένα είδη υπάρχον Source File επιλέγοντας **Project > Add Source**. Τα αρχεία προέλευσης μπορεί να βρίσκονται στον κατάλογο του project ή σε ένα απομακρυσμένο κατάλογο. Με το λογισμικό ISE, μπορούμε εύκολα να δημιουργήσουμε μανάδες-μοντέλα (modules) με VHDL κώδικα χρησιμοποιώντας τον επεξεργαστή κειμένου του ISE (ISE Text Editor). Ο κώδικας VHDL στη συνέχεια συνδέεται με ανώτατου επιπέδου VHDL σχεδίου αφού έχει γίνει αρχικοποίηση (instantiation) του module και έχει γίνει σύνθεση (ή αλλιώς να γίνει compile) με το υπόλοιπο project.

Για τη δημιουργία ενός νέου Source File επιλέγουμε **Project > New Source**. Στην εικόνα 2.8 φαίνεται η λίστα με τους υποστηριζόμενους τύπους αρχείων προέλευσης. Στη συνέχεια θα γίνει μια αναφορά για κάθε ένα από τα υποστηριζόμενα αρχεία προέλευσης.



Εικόνα 4.9: Οδηγός Δημιουργίας Νέου Αρχείου Προέλευσης (New Source File Wizard)

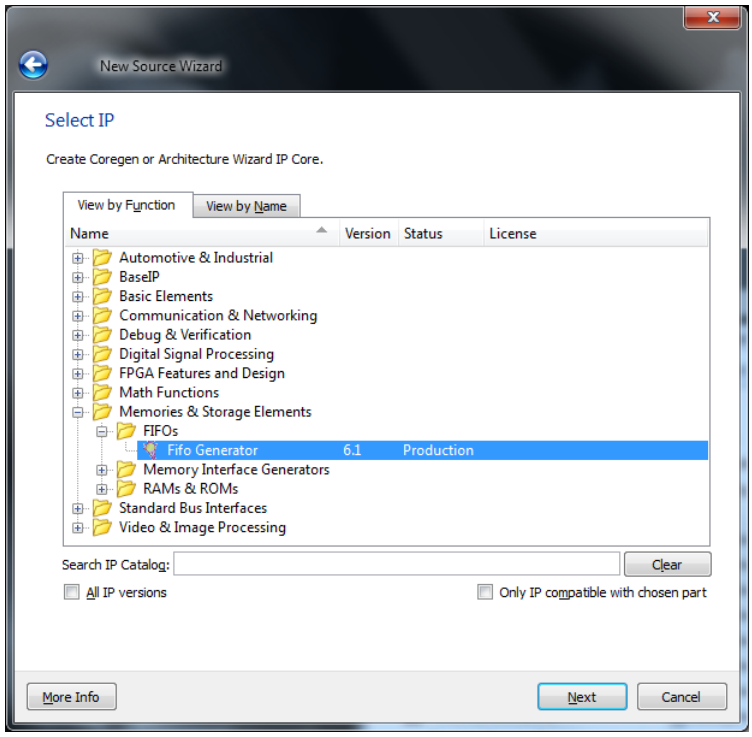
- **BMM File**, χρησιμοποιείται σε σχέδια επεξεργαστών PowerPC και Microblaze για να περιγράψει την οργάνωση σε ένα μπλοκ μνήμης RAM. Μόνο ένα αρχείο BMM επιτρέπεται να χρησιμοποιηθεί σε ένα project.
- **ChipScope Definition and Connection File**, περιέχει γενικές πληροφορίες για τις θύρες ενεργοποίησης και τις θύρες δεδομένων του πυρήνα (core) ChipScope.
- **Implementation Constraints File**, περιέχει λογικούς περιορισμούς που ορίζονται από το χρήστη.
- **IP (CORE Generator & Architecture Wizard)**, περιέχει προκαθορισμένες λογικές λειτουργιές καθώς και λειτουργιές που διαμορφώνουν τα χαρακτηριστικά της αρχιτεκτονικής ή των μονάδων.
- **MEM File**, χρησιμοποιείται για να καθορίσει τα περιεχόμενα της μνήμης. Μόνο ένα αρχείο MEM επιτρέπεται να χρησιμοποιηθεί σε ένα project.

- **Schematic**, περιέχει ένα σχέδιο από σχηματικά σύμβολα.
- **User Document**, περιέχει πληροφορίες του χρήστη, για παράδειγμα, συνοδευτικά έγγραφα.
- **Verilog Module**, περιέχει κώδικα γραμμένο σε γλώσσα Verilog,
- **Verilog Test Fixture**, καθορίζει τις τιμές των θυρών εισόδου για ένα αρχείο προέλευσης Verilog ώστε να εκτελεστεί μια προσημείωση λειτουργίας του αρχείου αυτού.
- **VHDL Module**, περιέχει κώδικα γραμμένο σε γλώσσα VHDL.
- **VHDL Library**, περιέχει μια συλλογή από VHDL πακέτα (VHDL packages).
- **VHDL Package**, περιέχει ορισμούς, μακροεντολές, υπορουτίνες, συμπληρωματικούς τύπους, υπότυπους, σταθερές, συναρτήσεις και άλλα αρχεία.
- **VHDL Test Bench**, καθορίζει τις τιμές των θυρών εισόδου για ένα αρχείο προέλευσης VHDL ώστε να εκτελεστεί μια προσομοίωση λειτουργίας του αρχείου αυτού.
- **Embedded Processor**, αρχείο Ενσωματωμένου μικροεπεξεργαστή που δημιουργήθηκε με τη πλατφόρμα Xilinx Platform Studio.

Από τα αναφερόμενα παραπάνω αρχεία προέλευσης θα χρησιμοποιήσουμε μόνο αρχεία IP Core (CORE Generator & Architecture Wizard), VHDL Modules, VHDL Test Benches και τέλος, ένα Implementation Constraints File για τη σύνδεση των εισόδων/εξόδων στα pins του FPGA.

4.1.4 Δημιουργία αρχείου IP (CORE Generator & Architecture Wizard)

Το λογισμικό CORE Generator της Xilinx δημιουργεί παραμετροποιημένες εκδόσεις προκαθορισμένων IP Cores βελτιστοποιημένα για FPGAs της Xilinx. Το IP CORE Generator περιλαμβάνει μνήμες και FIFOs και Ψηφιακής Επεξεργασίας Σήματος (DSP), μαθηματικές συναρτήσεις, standard bus interface, standard logic και λειτουργιές δικτύωσης. Στην παρούσα πτυχιακή εργασία θα χρησιμοποιήσουμε πέντε FIFOs του IP CORE Generator. Για τη δημιουργία IP CORE ανοίγουμε τον οδηγό δημιουργίας νέου αρχείου προέλευσης και επιλέγουμε IP (CORE Generator & Architecture Wizard).

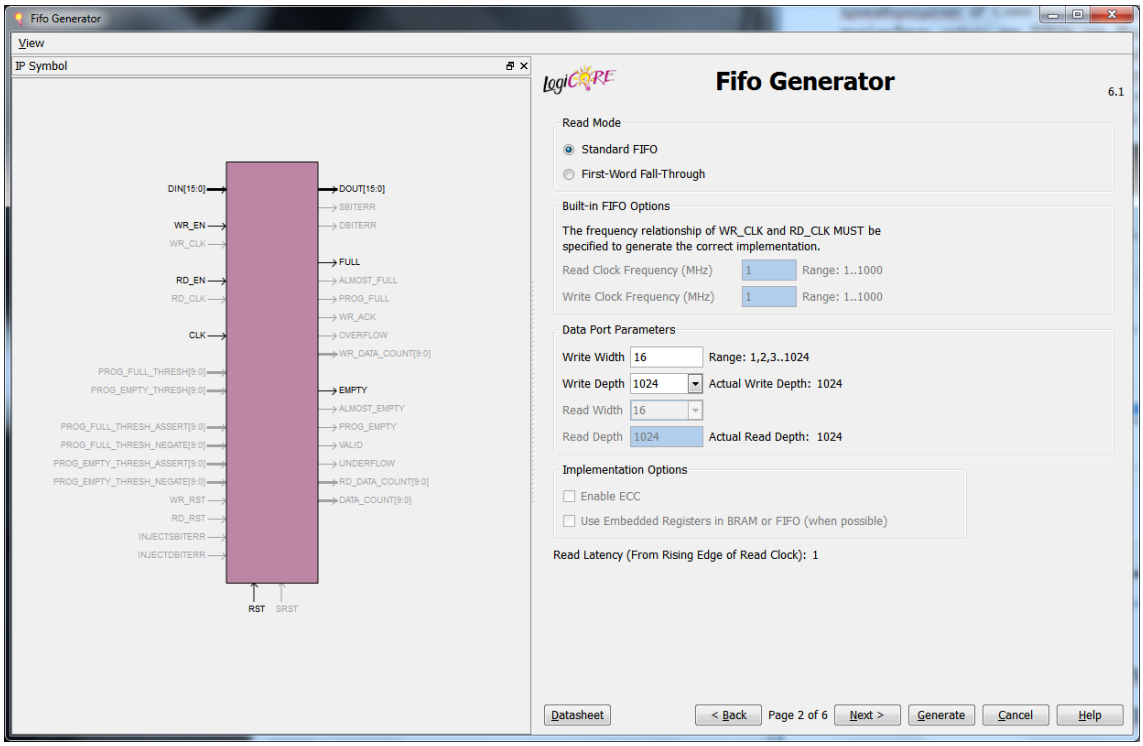


Εικόνα 4.10: Επιλογή του FIFO Generator από τον οδηγό νέου αρχείου προέλευσης

Επιλέγουμε το FIFO Generator από τον οδηγό δημιουργίας νέου αρχείου προέλευσης θα ανοίξει το παράθυρο επιλογής παραμέτρων για τη δημιουργία μιας μνήμης FIFO. Κατάλληλο μέγεθος

για τις FIFO που θα χρησιμοποιήσουμε θα είναι 1024 θέσεις μνήμης, με μέγεθος κάθε θέσης μνήμης 16 bit. Αφού εισάγουμε τις παραμέτρους πατάμε **Generate** θα δημιουργηθεί ένα πρότυπο μνήμης FIFO, το οποίο θα αρχικοποιηθεί (instantiate) πέντε φορές, για τις πέντε FIFOs που χρησιμοποιήσουμε στο project.

Στην εικόνα που ακλουθεί φαίνεται η εισαγωγή παραμέτρων στον οδηγό δημιουργίας νέας FIFO καθώς και η αρχικοποίηση των FIFOs που θα χρησιμοποιηθούν στο project.



Εικόνα 4.11: FIFO Generator

Όταν δημιουργηθεί ένας IP Core, το λογισμικό δημιουργεί αυτόματα ένα πρότυπο συγκεκριμενοποιημένου αρχείου VHDL ή Verilog το οποίο χρησιμοποιείται για την **αρχικοποίηση** του IP Core στο δικό μας HDL project.

```
component fifo
   port (
   clk: IN std_logic;
   rst: IN std_logic;
   din: IN std_logic_VECTOR(15 downto 0);
   wr_en: IN std_logic;
   rd_en: IN std_logic;
   dout: OUT std_logic_VECTOR(15 downto 0);
   full: OUT std_logic;
   empty: OUT std_logic);
end component;
your_instance_name : fifo
      port map (
         clk => clk,
         rst => rst,
         din => din,
         wr_en => wr_en,
         rd_en => rd_en,
         dout => dout,
         full => full,
         empty => empty);
```

Εικόνα 4.12: Αρχικοποίηση μνήμης FIFO (παρόμοια και οι υπόλοιπες τέσσερις)

4.1.5 Δημιουργία αρχείου VHDL Module

Με το λογισμικό ISE, μπορούμε εύκολα να δημιουργήσουμε μονάδες με VHDL κώδικα με τη χρήση του ISE Text Editor. Στη συνέχεια ο VHDL κώδικας συνδέεται με το VHDL σχέδιο ανώτερου επιπέδου μέσω **συγκεκριμενοποίησης (instantiation)** και εφόσον έχει **μεταγλωττιστεί (compiled)** με το υπόλοιπο σχέδιο. Για τη δημιουργία ενός VHDL αρχείου, από τον οδηγό δημιουργίας νέου αρχείου προέλευσης επιλέγουμε το VHDL Module, δίνουμε ένα όνομα, πατάμε Next και στη συνέχεια αφού ανοίξει το ISE Text Editor μπορούμε να ξεκινήσουμε να γράφουμε τον VHDL κώδικα.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity clock_divider_1sec is
port(
   clk50m,rst : in std_logic;
   clkout : out  STD_LOGIC
   );
end clock_divider_1sec;

architecture Behavioral of clock_divider_1sec is

   signal count: integer;
   signal clk1 : STD_LOGIC;

begin

   process (rst,clk50m)
   begin
      if rst='1' then
         count <= 0;
         clk1 <= '0';
      elsif rising_edge(clk50m) then
         if (count >= 24999999) then
            clk1 <= not clk1;
            count <= 0;
         else
            count <= count+1;
         end if;
      end if;
   end process;

   clkout <= clk1;


end Behavioral;
```

Εικόνα 4.13: Παράδειγμα αρχείου VHDL Module

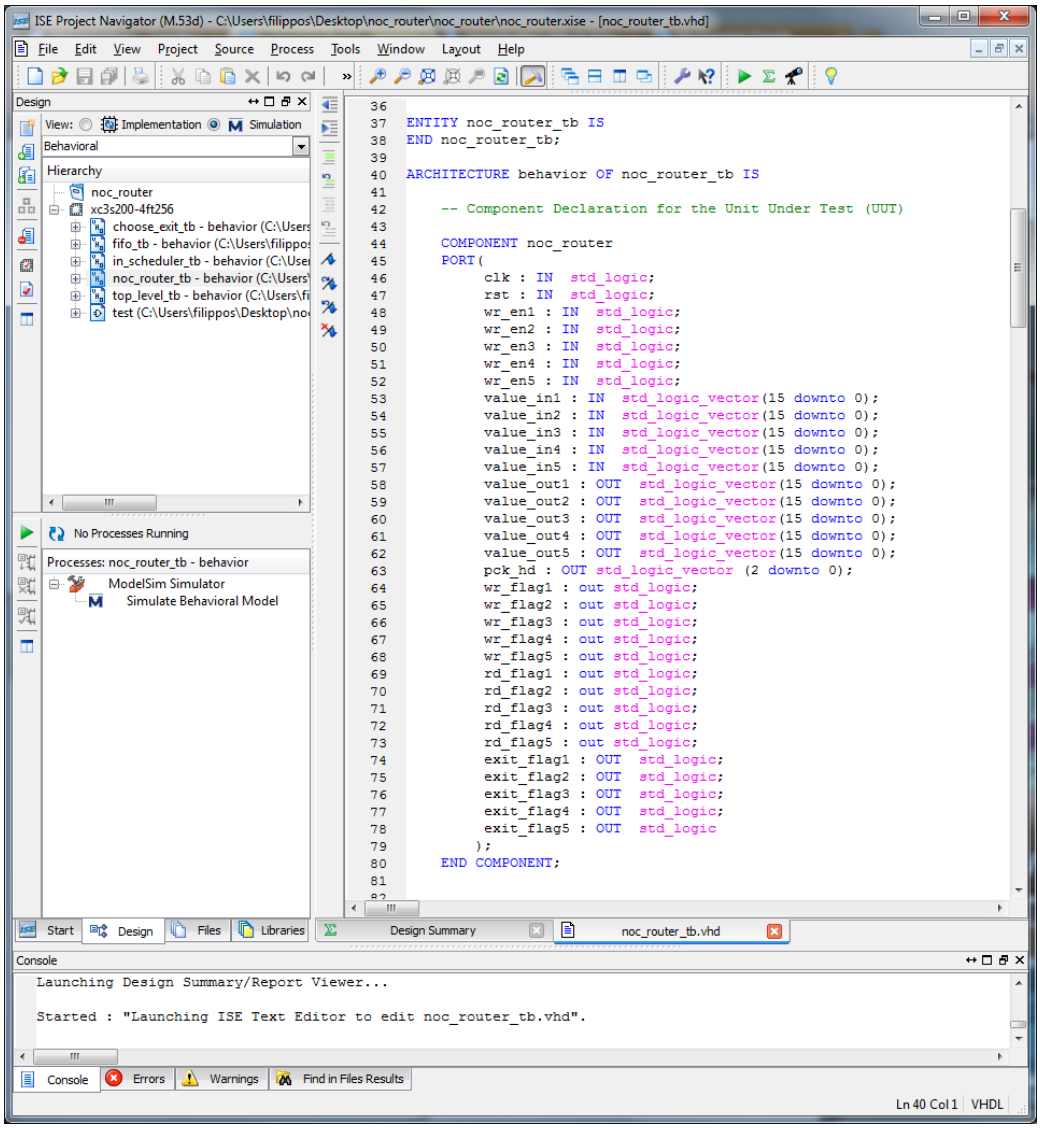
4.1.6 Δημιουργία αρχείου VHDL Test Bench

Μετά τη δημιουργία ενός αρχείου VHDL Module μπορούμε να προσομοιώσουμε τον τρόπο λειτουργίας του δημιουργώντας ένα αρχείο προέλευσης VHDL Test Bench από τον οδηγό δημιουργίας νέου αρχείου προέλευσης. Επιλεγέντας την δημιουργία ενός VHDL Test Bench θα ανοίξει ο οδηγός δημιουργίας νέου VHDL Test Bench και στο παράθυρο αυτό επιλέγουμε με ποιο VHDL Source File θέλουμε να συνδεθεί.

Σε αυτό το σημείο θα ανοίξει το ISE Text Editor και μπορούμε να προσπελάσουμε το test bench αρχείο που έχει δημιουργηθεί στο κύριο παράθυρο του programming window. Χρησιμοποιώντας το Text Editor του ISE μπορούμε να δώσουμε τιμές στις θύρες εισόδου του αρχείου εφαρμογής για να προσομοιώσουμε τη λειτουργία του. Η Xilinx χωρίζει τα αρχεία εφαρμογής από τα αρχεία προσομοίωσης, για το λόγο αυτό όταν θέλουμε να εμφανίζονται τα αρχεία προσομοίωσης

πρέπει να αλλάξουμε καρτέλα από "**Implementation**" σε "**Simulation**" και το αντίστροφο όταν θέλουμε να εμφανίζονται τα αρχεία εφαρμογής.

Επιλέγοντας την καρτέλα "**Simulation**" και αφού έχουμε δημιουργήσει ένα αρχείο test bench για κάποιο αρχείο προέλευσης μπορούμε να τρέξουμε την επιλογή "**Simulate Behavioural Model**". Πατώντας διπλό κλικ σε αυτή την επιλογή θα ξεκινήσει η προσομοίωση με το Modelsim



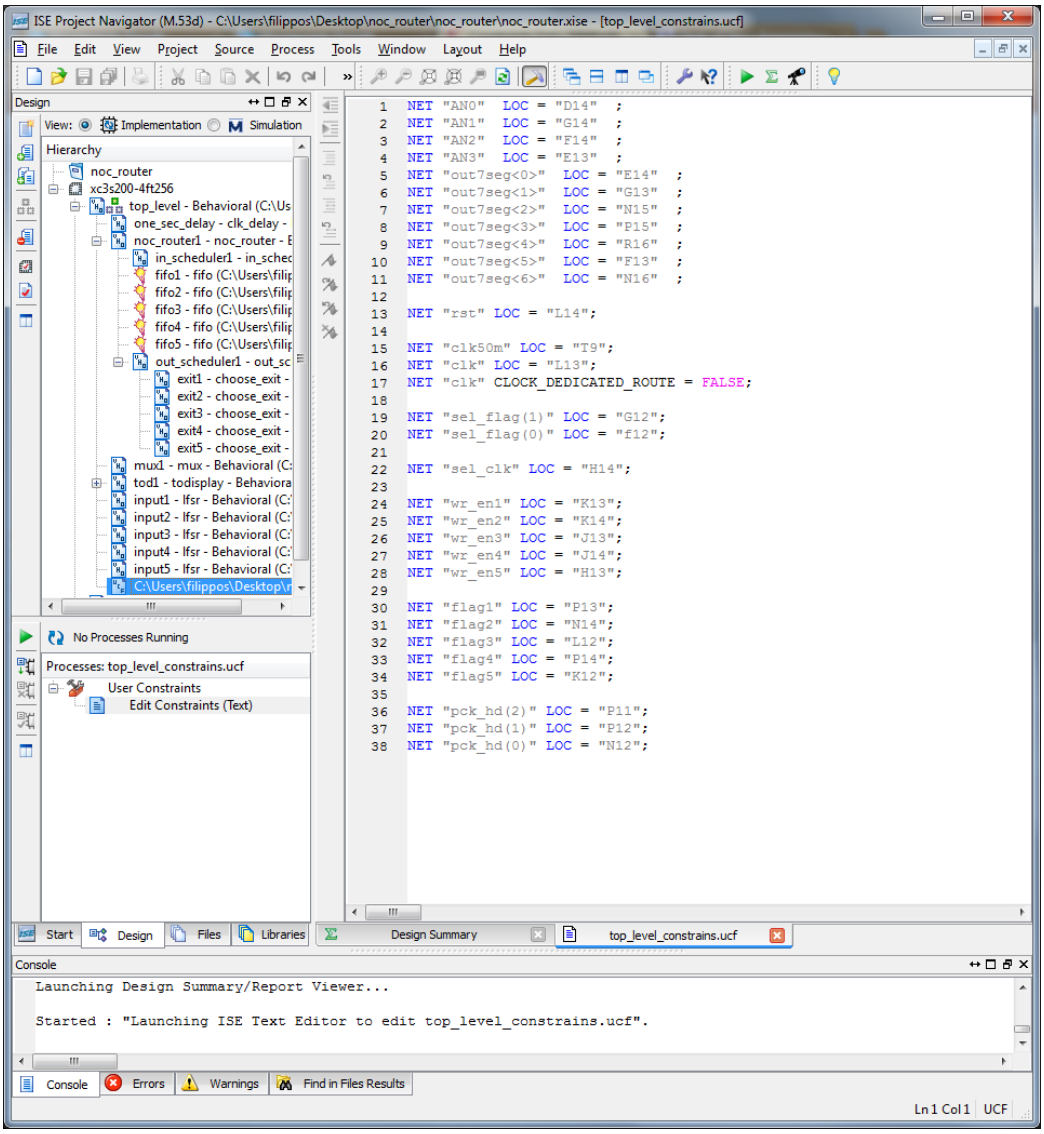
Εικόνα 4.14: Επιλογή καρτέλας "Simulation" (Αρχεία Προσομοίωσης)

Περισσότερα για το λογισμικό Modelsim θα ειπωθούν σε επόμενες παραγράφους.

4.1.7 Δημιουργία Implementation Constraints File

Τα Implementation Constraints Files ή αλλιώς User Constraints Files (UCF) χρησιμοποιούνται κατά τη διαδικασία υλοποίησης για εισαγωγή χρονικών περιορισμών και τοποθέτησης. Στη παρούσα πτυχιακή στο UCF αρχείο εισάγουμε τους περιορισμούς τοποθέτησης, τα pins εισόδων τα pins εξόδων για στο Spartan 3 Starter Kit board.

Ένα UCF αρχείο το δημιουργούμε στον οδηγό δημιουργίας νέου αρχείου προέλευσης. Επιλέγουμε Implementation Constraints File, χρησιμοποιούμε ένα όνομα και πατάμε OK. Τώρα που έχει δημιουργηθεί το UCF αρχείο μπορούμε να επεξεργαστούμε τους περιορισμούς (constraints) πατώντας πάνω στο UCF και επιλεγέντας τον **Text Editor** από το **User Constraints**, όπως φαίνεται στην εικόνα που ακλουθεί παρακάτω.

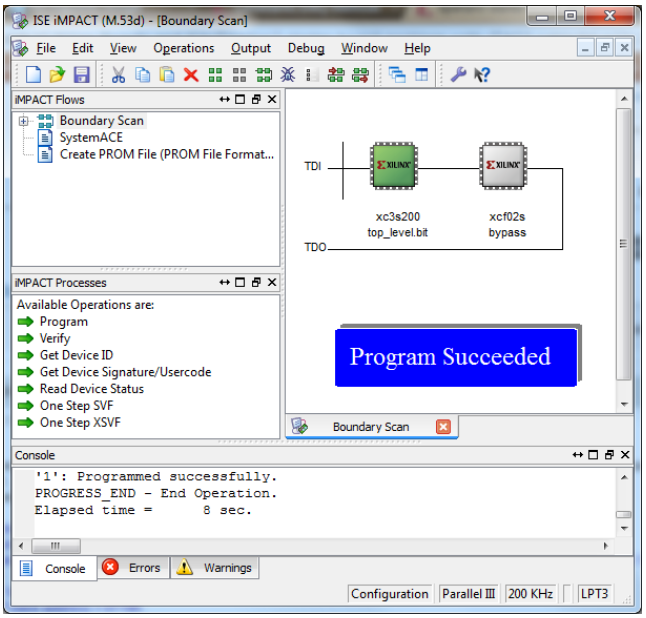


Εικόνα 4.15: Implementation Constraints File

4.1.8 Προγραμματισμός Συσκευής

Έχοντας ολοκληρώσει επιτυχημένα τα στάδια **Σύνθεσης (Synthesize)** και **Υλοποίησης (Implementation)**, τα επόμενα στάδια είναι η **Δημιουργία του Αρχείου Προγραμματισμού (Generate Programming File)** και η Ρύθμιση **της Συσκευής Προορισμού (Configure Target Device)**. Η διεργασία δημιουργίας του αρχείου προγραμματισμού τρέχει το BitGen (ένα πρόγραμμα της Xilinx που παράγει bit stream αρχεία) ώστε να δημιουργηθεί το bit stream αρχείο (.BIT) για τη ρύθμιση της συσκευής προορισμού της Xilinx. Η διεργασία αυτή τρέχει με διπλό κλικ στο **"Generate Programming File"** αφού έχει ολοκληρωθεί επιτυχημένα η. **Τοποθέτηση και Δρομολόγηση (Place & Route)**

Η διεργασία Ρύθμισης της Συσκευής Προορισμού χρησιμοποιεί το παραγόμενο αρχείο bit από Δημιουργία του Αρχείου Προγραμματισμού για να ρυθμίσει ή να προγραμματίσει τη συσκευή προορισμού. Η διεργασία αυτή τρέχει με διπλό κλικ στο " Configure Target Device " αφού έχει ολοκληρωθεί επιτυχημένα η "Δημιουργία του Αρχείου Προγραμματισμού". Εκτελώντας τη διεργασία Ρύθμισης της Συσκευής Προορισμού ξεκάνει το iMPACT, ένα εργαλείο με λειτουργιές που επιτρέπουν στο χρήστη να ρυθμίσει τη συσκευή προορισμού (ή διαφορετικά να προγραμματίσει τη συσκευή προορισμού).

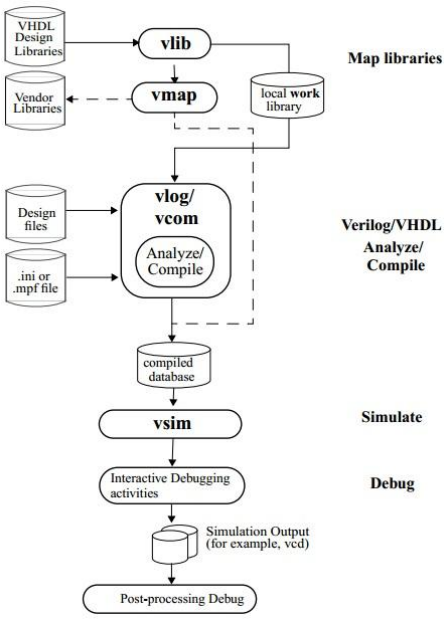


Εικόνα 4.16: Ρύθμιση της Συσκευής Προορισμού (Configure Target Device) - iMPACT - Επιτυχημένος Προγραμματισμός FPGA

Η συσκευή προορισμού είναι το FPGA του Spartan 3 Starter Kit board το οποίο επικοινωνεί με τον υπολογιστή του χρήστη μέσο JTAG καλωδίου συνδεμένο σε μια παράλληλη θύρα. Μετά από έναν επιτυχημένο προγραμματισμό το project τρέχει στο board σύμφωνα με τους περιορισμούς που έχουμε καθορίσει στο UCF αρχείο (Implementation Constraints File).

4.2 Modelsim 6.5

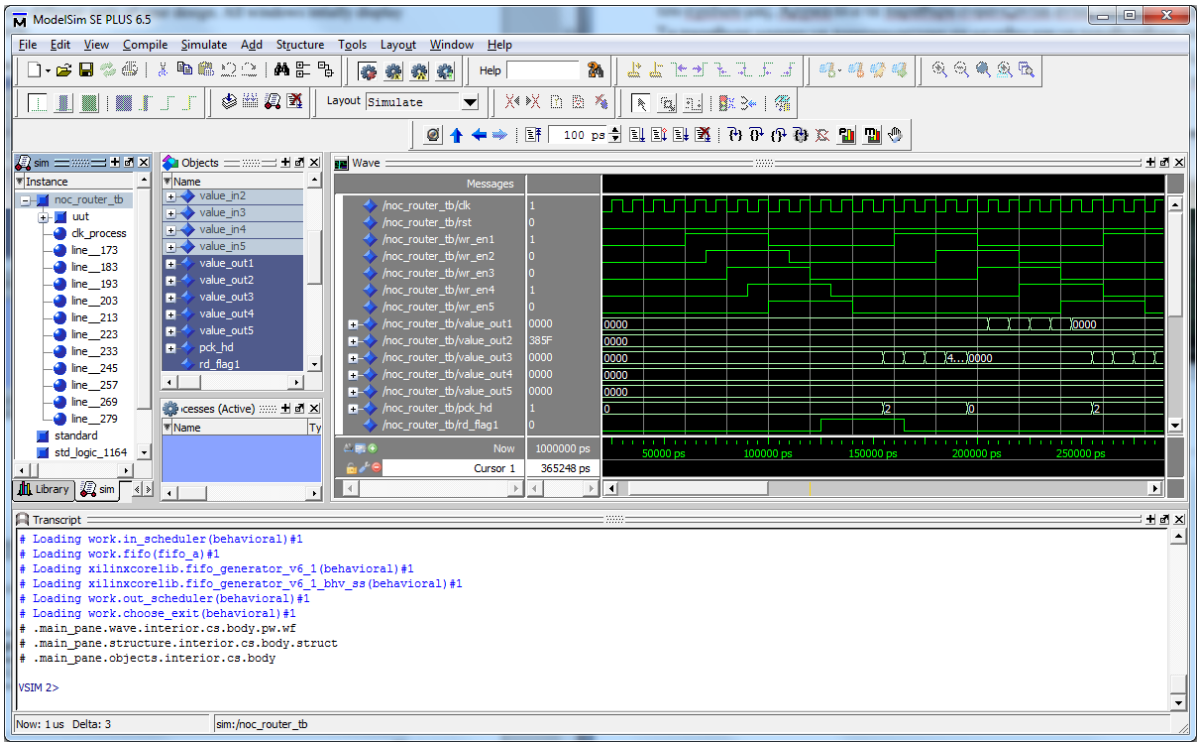
Το λογισμικό προσομοίωσης Modelsim ελέγχει τη λειτουργικότητα και το χρονοδιάγραμμα του project ή μέρος του project. Ο προσομοιωτής ερμηνεύει τον VHDL κώδικα σε λειτουργία κυκλώματος και εμφανίζει τα αποτελέσματα της λογικής που περιγράφεται στον HDL προσδιορισμό ώστε να διαπιστωθεί η σωστή λειτουργία του κυκλώματος. Η προσομοίωση μας επιτρέπει να δημιουργούμε και να επαληθεύουμε σύνθετες λειτουργίες σε ένα σχετικά μικρό χρονικό διάστημα.



Εικόνα 4.17: Επιχειρησιακή Δομή και Ροή ModelSim

Η προσομοίωση λαμβάνει χώρο σε διάφορα σημεία της ροής του project. Είναι ένα από τα πρώτα βήματα μετά την έναρξη του σχεδιασμού και ένα από τα τελευταία βήματα πριν την εφαρμογή, ως μέρος της επαλήθευσης της λειτουργικότητας και τέλος της απόδοσης του σχεδιασμού. Η προσομοίωση είναι μια επαναληπτική διαδικασία, η οποία μπορεί να εκτελείτε μέχρι να τηρείται η λειτουργικότητα και το χρονοδιάγραμμα του σχεδιασμού.

Το γραφικό περιβάλλον (GUI) του Modelsim παρέχει πρόσβαση σε πολλά εργαλεία εντοπισμού σφαλμάτων και παράθυρα που μας δίνουν τη δυνατότητα να αναλύσουμε διάφορα μέρη του σχεδίου μας. Αρχικά όλα τα παράθυρα εμφανίζονται εντός του κυρίου παραθύρου του Modelsim. Τα παράθυρα μπορούν να προσαρμοστούν σε μέγεθος και να τοποθετηθούν σε οποιοδήποτε μέρος της οθόνης μετά από επιλογή του χρήστη. Το Modelsim αποθηκεύει αυτές τις ρυθμίσεις και στις επόμενες συνεδρίες. Επαναφορά των ρυθμίσεων του Modelsim μπορεί να γίνει οποιαδήποτε στιγμή από το κύριο μενού.



Εικόνα 4.18: Παράθυρο Modelsim 6.5

Το κύριο παράθυρο του Modelsim έχει αρκετά παράθυρα και ο πινάκας που ακλουθεί συνοψίζει όλα αυτά τα διαθέσιμα παράθυρα.

Όνομα Παραθύρου	**Περιγραφή**
Main	Κεντρικό σημείο πρόσβασης GUI
Process	Εμφανίζει όλες τις διεργασίες που έχουν προγραμματιστεί να τρέχουν κατά τη διάρκεια του τρέχοντος κύκλου προσομοίωσης
Class Browsers	Εμφανίζει διαδραστικές σχέσεις τάξεων του συστήματος
Dataflow	Εμφανίζει τη φυσική συνδεσιμότητα και επιτρέπει την ανίχνευση γεγονότων
List	δείχνει κυματοειδές δεδομένα σε μορφή πίνακα
Locals	εμφανίζει τα αντικείμενα δεδομένων που είναι άμεσα ορατά στο τρέχον σημείο εκτέλεσης της επιλεγμένης διαδικασίας
Message Viewer	Επιτρέπει εύκολη πρόσβαση, οργάνωση και αναλύει προειδοποιήσεις, σφάλματα και άλλα μηνύματα που συντάσσονται κατά τη διάρκεια μια προσομοίωσης

Όνομα Παραθύρου	Περιγραφή
Memory	Παράθυρα που εμφανίζουν τις μνήμες και τα περιεχόμενα τους
Objects	Εμφανίζει όλα τα αντικείμενα δεδομένων που δηλώνονται στο τρέχον πεδίο
Source	Ένα πρόγραμμα επεξεργασίας κειμένου για προβολή και επεξεργασία αρχείων HDL, DO κ.ο.κ.
Structure (sim)	Εμφανίζει την ιεραρχική προβολή μιας εκτελούμενης προσομοίωσης
Transcript	Διατηρεί ένα τρέχον ιστορικό εντολών και μηνυμάτων και παρέχει ένα περιβάλλον γραμμής εντολών
Watch	Εμφανίζει σήματα και τιμές μεταβλητών σε μια τρέχουσα προσομοίωση
Wave	Εμφανίζει κυματομορφές

Πίνακας 4.1: Παράθυρα περιβάλλοντος Mode

Το Modelsim ξεκάνει μέσα από το ISE και τρέχει την προσομοίωση κάποιου VHDL Source File ή κάποιου Source File που έχει δημιουργηθεί από το IP Core Generator μετά τη δημιουργία του Test Bench. Το λογισμικό αυτό έχει πάρα πολλές δυνατότητες και επιλογές άλλα δεν θα αναφερθούμε περεταίρω διότι στη παρούσα πτυχιακή το εκτελούμε μόνο μέσα από το ISE και οι υπόλοιπες λειτουργιές του δεν μας αφορούν άμεσα.

4.3 Spartan 3 SE Starter Kit Board

Το Spartan 3 Starter Kit board της Xilinx παρέχει μια ισχυρή, αυτόνομη, χαμηλού κόστους, εύκολη στη χρήση πλατφόρμας ανάπτυξης και αξιολόγησης FPGA σχεδίων που στοχεύουν στη νέα γενιά των Spartan της Xilinx. Το board διαθέτει ένα FPGA (μοντέλο xc3s200-4ft256) 200.000 λογικών πυλών, ενσωματωμένες συσκευές εισόδου/εξόδου και μια ασύγχρονη μνήμη SRAM 1MB καθιστώντας την πλατφόρμα ιδανική για πειράματα με κάθε νέα σχεδίαση, από ένα απλό κύκλωμα μέχρι ένα ενσωματωμένο πυρήνα επεξεργαστή και είναι πλήρως συμβατό με όλες τις εκδόσεις των εργαλείων Xilinx ISE.

Εικόνα 4.19: Spartan 3 SE Starter Kit Board 1

Το Spartan 3 Starter Kit board περιλαμβάνει διάφορα εξαρτήματα (components). Μερικά από αυτά τα εξαρτήματα θα αναφερθούν σε επόμενες παραγράφους.

Το board διαθέτει μια συστοιχία (array) SRAM η οποία αποτελείτε είτε μια ενιαία 32 θέσεων x 256 Kbit ή δυο ανεξάρτητες συστοιχίες 16 θέσεων x 256Kbit καθεμία. Η διαμόρφωση 32 θέσεων x 256 Kbit είναι ιδανική για να αποθηκεύει Microblaze οδηγίες. Εναλλακτικά, ωστόσο, μπορεί να παρέχει μια υψηλής πυκνότητας αποθήκευση δεδομένων για μια ποικιλία εφαρμογών, όπως ψηφιακή επεξεργασία σήματος (DSP), μεγάλες FIFOs δεδομένων και buffers γραφικών.

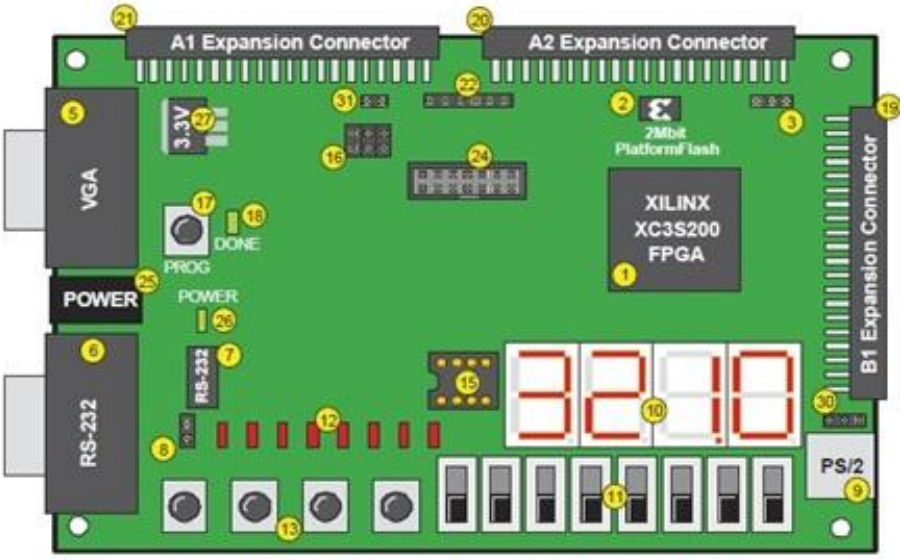
Το Spartan 3 Starter Kit board έχει οκτώ διακόπτες. Οι διακόπτες βρίσκονται κάτω δεξιά του board με ονομασίες sw0 εως sw7. Ένας διακόπτης που είναι συνδεμένος παράγει λογικό ‘ένα’ όταν είναι πάνω και λογικό ‘μηδέν’ όταν είναι κάτω, Επίσης διαθέτει και 4 Μπουτάν στιγμιαίας επαφής (push-button). Τα button αυτά βρίσκονται κάτω αριστερά του board με ονομασίες btn0 έως btn3. Ένα button παράγει λογικό ‘ένα’ όταν είναι πατημένο, διαφορετικά λογικό ‘μηδέν’. Πάνω από τα push-button βρίσκονται οκτώ δίοδοι εκπομπής φωτός (LED) με ονομασία led0 έως led7. Τα LED αυτά ενεργοποιούνται με λογικό ένα.

Το Spartan 3 έχει μια οθόνη επτά τμημάτων τεσσάρων χαρακτήρων η οποία ελέγχεται από το χρήστη με σήματα εισόδου/εξόδου. Κάθε ψηφίο μοιράζεται οκτώ κοινά σήματα έλεγχου που ενεργοποιούν επιμέρους τμήματα LED. Κάθε χαρακτήρας έχει μια ξεχωριστή είσοδο ελέγχου ενεργοποίησης. Κάθε μεμονωμένο LED της οθόνης επτά τμημάτων ενεργοποιείτε με λογικό μηδέν. Επειδή το component αυτό το χρησιμοποιούμε στην απεικόνιση πακέτων από τις θύρες εξόδου του NoC Router πάνω στο board θα αναλυθεί ο τρόπος λειτουργιάς του αναλυτικά στο επόμενο κεφάλαιο.

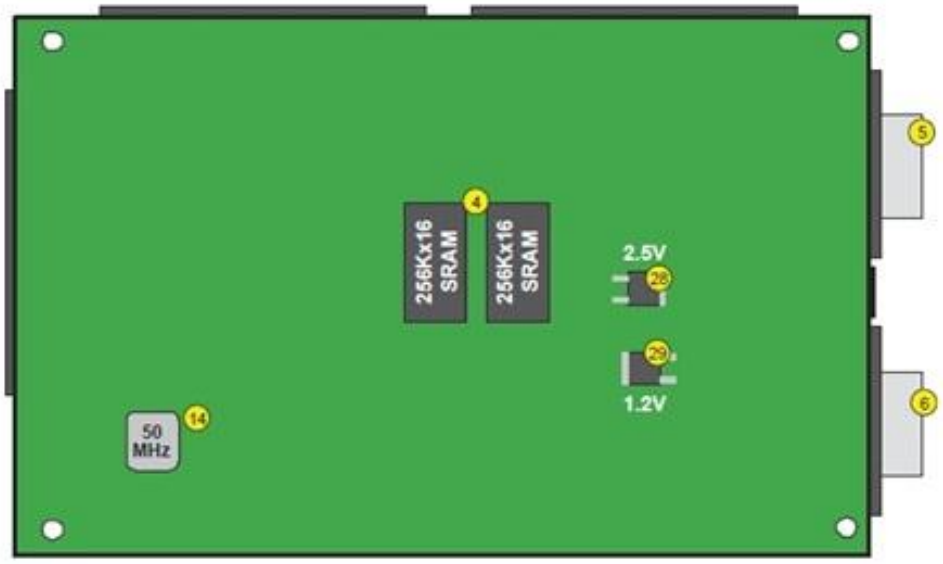
Το board περιλαμβάνει μια θύρα VGA για σύνδεση με οθόνη υπολογιστή. Το FPGA του Spartan 3 ελέγχει πέντε σήματα έλεγχου VGA, το κόκκινο (RED), πράσινο (GREEN), μπλε (BLUE), οριζόντιος συγχρονισμός (Horizontal Sync) και κάθετος συγχρονισμός (Vertical Sync) που είναι διαθέσιμα στην υποδοχή VGA. Χρησιμοποιώντας τα πέντε αυτά σήματα μπορούμε να αναπαράγουμε γραφικά στοιχεία μέσω του FPGA σε μια οθόνη υπολογιστή. Μια θύρα PS/2 είναι επίσης διαθέσιμη για σύνδεση ποντικιού ή πληκτρολογίου. Οι ρυθμίσεις των FPGA σημάτων είναι διαφορετικές αν θέλουμε να συνδέσουμε ποντίκι ή αν θέλουμε να συνδέσουμε πληκτρολόγιο.

Το Spartan 3 Starter Kit board διαθέτει μια σειριακή θύρα RS-232 για εκπομπή και λήψη σημάτων. Μέσω της θύρας RS-232 το board μπορεί να επικοινωνεί με ένα υπολογιστή ή με άλλες συσκευές (σκληρούς δίσκους, modems, εκτυπωτές κ.ο.κ.) ή ακόμα και με κάποια άλλη συσκευή της οικογένειας Xilinx. Διαθέτει ακόμα ένα ρολόι συχνοτήτων 50 MHz και μια προαιρετική υποδοχή για κάποια άλλη πηγή συχνοτήτων. Το ρολόι των 50 MHz μπορεί να χρησιμοποιηθεί ως έχει ή διαφορετικά να χρησιμοποιηθεί για να παραχθεί κάποια ποιο βολική συχνότητα. Στο επόμενο κεφάλαιο θα δούμε πως χρησιμοποιήθηκε το ρολόι αυτό σε διαφορετικές συχνότητες.

Όπως αναφέρθηκε και παραπάνω για το προγραμματισμό του FPGA του Spartan 3 χρησιμοποιούμε ένα JTAG καλώδιο στην υποδοχή που έχει πάνω το board συνδεμένο με τον υπολογιστή μέσω παράλληλης θύρας. Λειτουργεί με συνεχή τάση 5V παραγόμενη από μετασχηματιστή εναλλασσόμενου ρεύματος. Τέλος, το Spartan 3 Starter Kit board έχει τρεις υποδοχές επέκτασης (A1,A2 και B1) κάθε μια από τις οποίες έχει 40 pins για σύνδεση



Εικόνα 4.20: Πάνω όψη του Spartan 3 Starter Kit Board



Εικόνα 4.21: Κάτω όψη του Spartan 3 Starter Kit Board 1

5 Υλοποίηση NoC Router

Το NoC – router αποτελείτε από 5 μνήμες RAM τύπου FIFO στις οποίες φτάνουν πακέτα δεδομένων από πέντε πόρτες εισόδου και ένας arbiter επιλέγει σε ποια από τις 5 πόρτες εξόδου θα δρομολογηθούν. Δυο schedulers χρησιμοποιούνται για τη δρομολόγηση μέσα στο NoC router από την πόρτα εισόδου μέχρι την πόρτα εξόδου, ένας input scheduler που δρομολογεί τα πακέτα από την είσοδο στις FIFOόπου αποθηκεύονται προσωρινά και ένας output scheduler που δρομολογεί τα πακέτα από τις FIFO στις πόρτες εξόδου. Η λειτουργία των input και output scheduler θα γίνει πολύ αναλυτικά παρακάτω καθώς είναι τα σημαντικότερα μοντέλα για τη σωστή λειτουργία του router και την επικοινωνία ανάμεσα στα SoC.

Η επικοινωνία μέσα στο NoC router πρέπει να εξασφαλίζει ότι δεν θα υπάρχουν απώλειες πακέτων ούτε επικαλύψεις έτσι ώστε οποιοδήποτε πακέτο εισέρθει από μια πόρτα εισόδου να δρομολογηθεί με ασφάλεια και με τη λιγότερο δυνατή καθυστέρηση στην πόρτα εξόδου για την οποία προορίζεται.

Τα πακέτα δεδομένων που φτάνουν στις πόρτες εισόδου είναι της τάξεως των 64 bits ή 8 Bytes (8 λέξεις) το καθένα αλλά για λόγους ευκολίας αναπαράστασης και αποθήκευσης χωρίζονται σε 4 flits των 16 bits ή 2 Bytes (2 λέξεων) το καθένα. Κάθε ένα πακέτο δρομολογείτε στην πόρτα εξόδου με βάση τα 3 σημαντικότερα bits στην κεφαλίδα του, αναλυτικά: πακέτα με πρόθεμα ‘’000’’ και ‘’101’’ δρομολογούνται στην πρώτη πόρτα εξόδου, ‘’001’’ στην δεύτερη, ‘’010’’ και ‘’110’’ στη τρίτη, ‘’011’’ στην τέταρτη και τέλος ‘’100’’ και ‘’111’’ στην πέμπτη πόρτα.

Η σωστή επικοινωνία του NoC router εξασφαλίζεται λόγο του χρονοπρογραμματισμού που γίνεται στο εσωτερικό του. Στο εσωτερικό του NoC router χρησιμοποιούνται δυο διαιτητές (arbiter). Ένας arbiter που συνεργάζεται με τον input scheduler και του δίνει πληροφορίες για την κατάσταση των FIFO, δηλαδή ποιες από αυτές είναι απασχολημένες, πότε και αν κάποια από αυτές είναι άδεια ή γεμάτη και αν μπορεί να γραφτεί κάτι σε αυτές. Ο δεύτερος arbiter χρησιμοποιείτε στην ανάγνωση από τις FIFO και συνεργάζεται με τον output scheduler, είναι αυτός που αποφασίζει πότε (χρονικά) και από ποια FIFO θα γίνει ανάγνωση πακέτου.

Το δεύτερο μοντέλο του arbiter που χρησιμοποιείτε, όταν κάνει ανάγνωση πακέτων από τις FIFO στέλνει τα πακέτα αυτά στον output scheduler ώστε να αποφασιστεί σε ποια πόρτα εξόδου προορίζονται. Το μοντέλο arbiter αυτό στηρίζει τη λειτουργία του στον **αλγόριθμο χρονοπρογραμματισμού εξυπηρέτησης εκ περιτροπής** (**Round - Robin scheduling algorithm**). Ο αλγόριθμος αυτός επιλέχτηκε επειδή είναι ένας από τους παλαιότερους, πιο δίκαιους και πιο διαδεδομένους αλγορίθμους χρονοπρογραμματισμού.

Η λογική του **Round - Robin scheduling** είναι απλή. Σε κάθε FIFO δίνεται ο ίδιος χρόνος για ανάγνωση πακέτων διάρκειας ίσης με το χρόνο ανάγνωσης ενός πακέτου (4 clock cycles). Με μια μηχανή καταστάσεων τύπου Moore ο arbiter ‘’τσεκάρει’’ αν μια FIFO έχει πακέτα προς προσπέλαση, αν αυτό ισχύει τότε θα προωθήσει στην έξοδο ένα πακέτο και θα προχωρήσει στην επόμενη FIFO κάνοντας την ίδια εργασία. Αυτό γίνεται και για τις πέντε FIFO αντιμετωπίζοντας τις όλες το ίδιο (ίδια δικαιώματα). Ο αλγόριθμος δεν προκαλεί στέρηση (starvation) και δεν υπάρχουν επικαλύψεις ή απώλειες πακέτων.

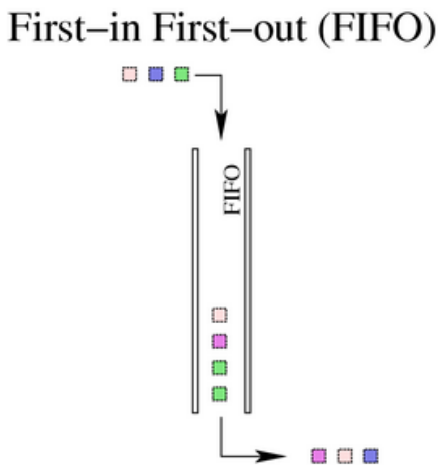
5.1 Αρχεία Προέλευσης (Source Files)

Για την υλοποίηση αυτής της πτυχιακής και για να εξασφαλιστεί η σωστή επικοινωνία μέσα στο NoC χρησιμοποιήθηκαν διάφορα αρχεία προέλευσης (source files), κάθε ένα από τα οποία αναφέρεται στις παρακάτω υποενότητες και δίνεται περιγραφή για τον τρόπο λειτουργίας τους.

5.1.1 Μνήμες FIFO (First In First Out)

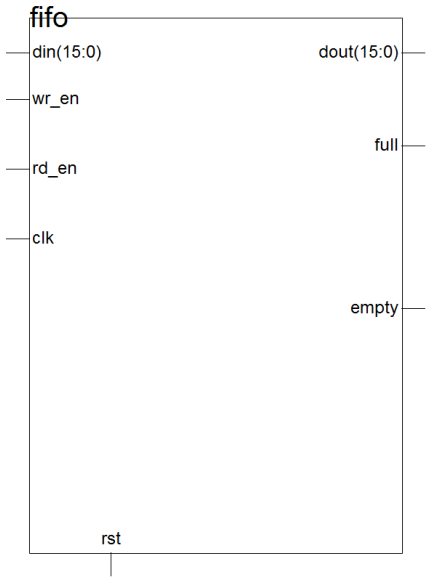
Σε μια μνήμη FIFO (First In First Out), ακρωνύμιο που δηλώνει ότι το πρώτο στοιχείο που μπαίνει στην ουρά είναι και το πρώτο που θα εξυπηρετηθεί, τα στοιχεία βγαίνουν με τη σειρά που μπήκαν. Τυχαία προσπέλαση ενός συγκεκριμένου στοιχείου δεν είναι δυνατή. Κάθε μια από τις μνήμες τύπου FIFO διαθέτει 1024 θέσεις για αποθήκευση 2 bytes (2 λέξεων) ή 16 bits, για την

περίπτωση μας όπου είναι και ο χώρος αποθήκευσης για καθένα από τα flit ενός πακέτου. Το module αυτό δημιουργείτε μέσω CORE Generator του ISE της Xilinx.



Εικόνα 5.1: Εισαγωγή και εξαγωγή δεδομένων από μνήμη FIFO

Μια FIFO στο σύνολο της διαθέτει πέντε εισόδους: reset, clock, data in, write enable και read enable και τρεις εξόδους: data out, empty και full. Το reset χρησιμοποιείτε για επανεκκίνηση της μονάδας. Αν κάποια στιγμή η είσοδος αυτή γίνει ενεργή θα διαγραφούν όλα τα δεδομένα που υπήρχαν αποθηκευμένα μέχρι εκείνη τη χρονική στιγμή και η μνήμη θα είναι άδεια. Επειδή όλο το project είναι συγχρονισμένο σε κοινό ρολόι κάθε FIFO διαθέτει είσοδο ρολογιού (clock). Οι είσοδοι write enable / read enable ενεργοποιούνται σε χρονικές στιγμές κατά τις οποίες θέλουμε να γράψουμε ή να διαβάσουμε, αντίστοιχα, δεδομένα στην ή από την μνήμη. Data in είναι η είσοδος των 16 bits που θέλουμε να αποθηκεύσουμε στη μνήμη αλλά για να γίνει αυτό πρέπει το write enable να είναι ενεργοποιημένο. Η έξοδος data out αφόρα την προσπέλαση των δεδομένων 16 bit από την πρώτη θέση της μνήμης όταν η είσοδος read enable ενεργοποιηθεί. Τα full και empty ενεργοποιούνται όταν η μνήμη γεμίσει ή αδειάσει κάποια χρονική στιγμή αντίστοιχα.



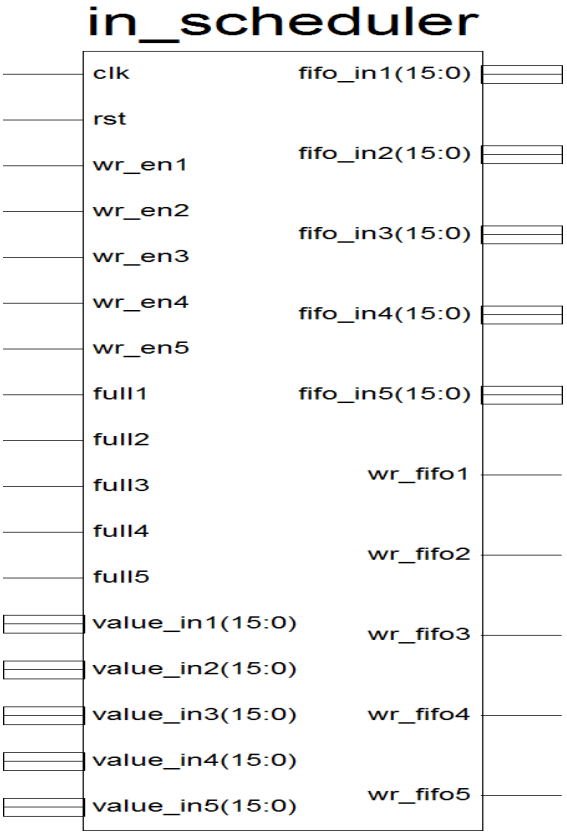
Εικόνα 5.2: Σχηματικό σύμβολο μνήμης FIFO, είσοδοι (αριστερά) και έξοδοι (δεξιά)

5.1.2 Χρονοδρομολογητής εισόδου (Input Scheduler)

Ο Χρονοδρομολογητής εισόδου βρίσκεται μεταξύ των θυρών εισόδου και των πέντε FIFO. Η εργασία του είναι να αποφασίζει σε ποια FIFO θα αποθηκευτεί ένα εισερχόμενο πακέτο από οποιαδήποτε θύρα εισόδου (υπάρχουν πέντε) και αν έχει φτάσει. Ο σκοπός του είναι να μην αφήσει κάποια από τις FIFO να γεμίσει με πακέτα δεδομένων τα οποία δεν έχουν φτάσει στην έξοδο, ενώ την ίδια χρονική στιγμή μπορεί να υπάρχει κάποια από τις FIFO η οποία μπορεί να είναι και άδεια.

Στο σημείο αυτό πρέπει να σημειωθεί ότι αν στην περίπτωση μια FIFO γεμίσει δεν μπορεί να αποθηκεύσει περεταίρω πακέτα αν δεν αποδεσμευτεί χώρος από τη μνήμη με την προώθηση στην έξοδο των ήδη υπαρχόντων πακέτων. Πακέτα δεδομένων τα οποία φτάνουν σε μια FIFO και αυτή είναι γεμάτη απορρίπτονται και αυτό είναι ανεπίτρεπτο γιατί υπάρχει απώλεια δεδομένων.

Ο Χρονοδρομολογητής εισόδου έχει 5 εισόδους και 5 εξόδους για πακέτα δεδομένων. Όταν ένα πακέτο φτάσει από μια θύρα εισόδου ο Scheduler εξ' ορισμού θα αποθηκεύσει το πακέτο στην αντίστοιχη FIFO της θύρας εισόδου (δηλ. 1 θύρα στην 1 FIFO, 2 θύρα στην 2 FIFO κ.ο.κ.) αλλά όχι πριν επαληθευτεί ότι δεν είναι γεμάτη ή ότι κάποιο άλλο πακέτο από κάποια άλλη θύρα εισόδου αποθηκεύεται εκείνη τη στιγμή. Όταν ένα πακέτο γράφεται σε μια FIFO τότε ενεργοποιείτε ένα flag ώστε να δείξει ότι η μνήμη αυτή χρησιμοποιείτε εκείνη χρονική στιγμή και δεν μπορεί να γραφτεί πακέτο από άλλη θύρα εισόδου. Αυτό εξασφαλίζει ότι δεν πρόκειται να υπάρξει υπερκάλυψη ή διαγραφή πακέτων κατά την είσοδο.



Εικόνα 5.3: Σχηματικό σύμβολο ενός Χρονοδρομολογητή εισόδου (Input Scheduler), είσοδοι (αριστερά) και έξοδοι (δεξιά)

Οι είσοδοι value_in1(15:0) έως value_in5(15:0) αφορούν τα flit που εισέρχονται από τις πέντε θύρες εισόδου και οι έξοδοι fifo_in1(15:0) έως fifo_in5(15:0) την προώθηση αυτών των flit προς τις πέντε FIFO. Τα rst και clk (reset,clock) χρησιμοποιούνται με τον ίδιο τρόπο όπως αναφέραμε παραπάνω (βλ. FIFO). Τα σήματα εισόδου (wr_en1 έως wr_en5) ενεργοποιούνται για να δείξουν από ποια θύρα έχουμε εισερχόμενα πακέτα ενώ τα σήματα εξόδου (wr_fifo1 έως wr_fifo5) ενεργοποιούν την αντίστοιχη FIFO στην οποία θα αποθηκευτούν τα εισερχόμενα πακέτα. Τέλος, τα σήματα

εισόδου full1 έως full5 ενεργοποιούνται για να δείξουν αν μια FIFO είναι γεμάτη έτσι ώστε να μην επιχειρήσει ο scheduler να αποθηκεύσει κάποιο πακέτο σε αυτήν.

Το μοντέλο αυτό είναι ένα αρχείο VHDL module και αναπτύχτηκε με VHDL κώδικα μέσω του text editor του λογισμικού Xilinx ISE (πολυπλοκότητα :167 γραμμές κώδικα).

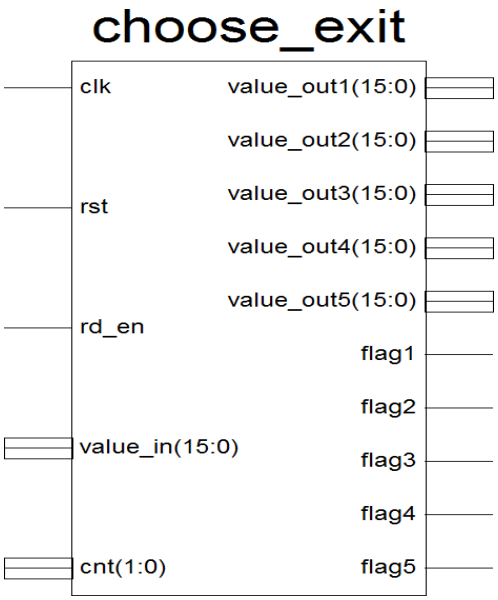
5.1.3 Χρονοδρομολογητής εξόδου (Output Scheduler)

Ο Χρονοδρομολογητής εξόδου βρίσκεται μεταξύ των FIFOs και των θυρών εξόδου. Η κύρια εργασία του είναι να προωθήσει τα εξερχόμενα πακέτα από τις FIFO στις θύρες εξόδου. Τα πακέτα αποθηκεύονται προσωρινά μέχρι προωθηθούν στην θύρα εξόδου λόγο έλεγχου που πρέπει να πραγματοποιηθεί προκειμένου κάθε πακέτο εξερχόμενο από μια FIFO να ακολουθήσει το σωστό μονοπάτι για τη θύρα εξόδου.

Στο εσωτερικό του output scheduler κάθε FIFO συνδέεται με ένα module το οποίο ονομάζεται **επιλογέας θύρας εξόδου (choose exit)**. Πέντε choose exit είναι αυτά που κάνουν όλη τη δουλειά μέσα στον output scheduler.

5.1.3.1 Επιλογέας Εξόδου (Choose Exit)

Όταν ένα πακέτο εξερχόμενο από μια FIFO εισέρχεται στο choose exit με το οποίο είναι συνδεμένη. Τότε γίνεται έλεγχος της κεφαλίδας (header) του πακέτου ώστε να διαπιστωθεί σε ποια θύρα εξόδου πρέπει να προωθηθεί. Υπενθυμίζεται ότι κάθε πακέτο είναι χωρισμένο σε τέσσερα flit. Ο έλεγχος αυτός πρέπει να γίνει μόνο στο πρώτο flit κάθε πακέτου και όχι και στα υπόλοιπα τρία. Για το λόγο αυτό χρησιμοποιούμε μετρητές (counters) των δυο bit κάθε ένας (μετράει απο 1 έως 4) σε όλο το μήκος από την στιγμή που το πακέτο θα εξέλθει από τη FIFO μέχρι να φτάσει στην θύρα εξόδου. Αυτό εξασφαλίζει ότι θα γίνεται έλεγχος μόνο όταν ο μετρητής ισούται με ‘άσσο’ (1) άρα και στο πρώτο flit και τα υπόλοιπα τρία θα επακολουθήσουν την διαδρομή του πρώτου.



Εικόνα 5.4: Σχηματικό σύμβολο της μονάδας επιλογέα εξόδου (choose exit), είσοδοι (αριστερά) και έξοδοι (δεξιά)

Όταν φτάσει ένα πακέτο (value_in(15:0)) στον επιλογέα εξόδου γίνεται έλεγχος της κεφαλίδας (header) του για να καθοριστεί σε ποια θύρα εξόδου προωθείται. Πακέτα με κεφαλίδα ‘’000’’ και ‘’101’’ αποθηκεύονται στην πρώτη έξοδο (value_out1(15:0)), ‘’001’’ στην δεύτερη (value_out2(15:0)), ‘’010’’ και ‘’110’’ στη Τρίτη (value_out3(15:0)), ‘’011’’ στην τέταρτη (value_out4(15:0)) και τέλος ‘’100’’ και ‘’111’’ στην πέμπτη έξοδο (value_out5(15:0)). Οι έξοδοι από κάθε choose exit είναι προσωρινοί καταχωρητές των πακέτων τα οποία θα φτάσουν στις τελικές θύρες

εξόδου. Οι είσοδοι rst,clk (reset, clock) στο block choose exit χρησιμοποιούνται για τον ίδιο λόγο που αναφέρεται στο παραπάνω υποκεφάλαιο (συγχρονισμένο & συνδυαστικό project).

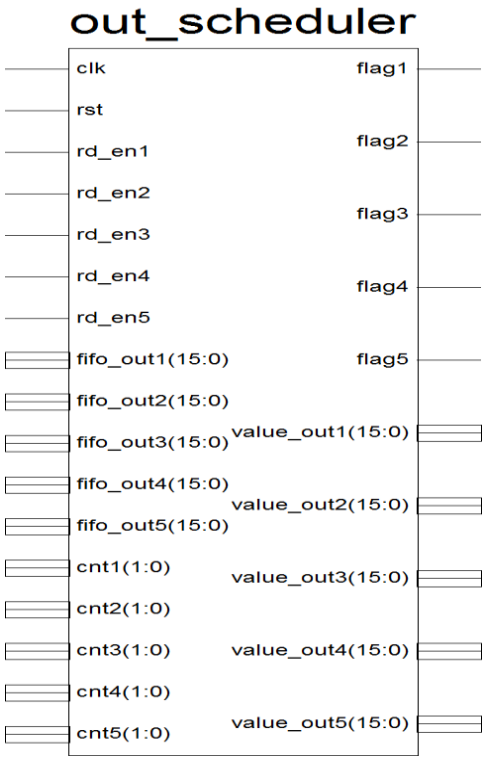
Κάθε εισερχόμενο πακέτο σε ένα choose exit έχει πέντε πιθανές διαδρομές (μια από τις πέντε θύρες εξόδου). Όταν γίνεται ανάγνωση από την μια FIFO τότε το πακέτο αποθηκεύεται προσωρινά σε μια από τις πέντε εξόδους του choose exit ανάλογα με το πρόθεμα του κάθε πακέτου όπως αναφέραμε παραπάνω. Ένα σήμα (rd_en) εισέρχεται στον επιλογέα εξόδου από μια FIFO μας ειδοποιεί ότι υπάρχει διαθέσιμο πακέτο προς αποστολή σε κάποια θύρα εξόδου. Όταν ενεργοποιηθεί αυτό το σήμα επιλέγεται η θύρα εξόδου και το πακέτο αποθηκεύεται εκεί προσωρινά.

Με την προσωρινή αποθήκευση κάποιου πακέτου στο choose exit ενεργοποιούνται τα σήματα εξόδου flag1, flag2, flag3, flag4, flag5 ως χρησιμοποιούνται σηματοδότες εξόδου (ανάλογα σε ποια από τις πέντε θύρες εξόδου προωθείται το πακέτο) και χρησιμοποιούνται σε όλη τη διαδρομή της ιεραρχίας. Τα flags αυτά διαβάζονται από τον arbiter για να επιλέξει πως θα φτάσουν τα πακέτα που είναι προσωρινά αποθηκευμένα στο choose exit στις τελικές θύρες εξόδου.

Η μονάδα choose exit είναι ένα VHDL source file το οποίο έχει περιγραφεί με VHDL κώδικα (100 γραμμές) και συγκεκριμενοποιείται (ή διαφορετικά αρχικοποιείται) πέντε φορές, μια για κάθε FIFO .

5.1.3.2 Λειτουργία Χρονοδρομολογητή εξόδου

Τα πακέτα των καταχωρητών από ένα choose exit θα φτάσουν στις τελικές θύρες εξόδου του output scheduler μετά από μερικούς κύκλους ρολογιού αφού έχει ανάγνωση από την FIFO με την οποία είναι συνδεμένο το choose exit. Σε όλο το σύνολο έχουμε 25 καταχωρητές από τα πέντε choose exit (πέντε FIFO επί πέντε choose exit). Για το λόγο του ότι δεν μπορεί να γίνει ανάγνωση σε πάνω από μια FIFO ανά χρονική στιγμή (θα εξηγήσουμε παρακάτω το λόγο – κεφάλαιο 3.2) ο arbiter χρησιμοποιεί τα flags των choose exit για να προωθήσει τα πακέτα των καταχωρητών στις θύρες εξόδου του output scheduler. Όλο αυτό γίνεται επειδή δεν μπορούμε να συνδέσουμε άμεσα όλες τις εξόδους από τα choose exit (σύνολο 25) με τις πέντε θύρες εξόδου του output scheduler. Το πρόβλημα αυτό αναφέρεται ως multiple drivers error και λύνεται με το τρόπο που περιγράφουμε.



Εικόνα 5.5: Σχηματικό σύμβολο ενός Χρονοδρομολογητής εξόδου (Output Scheduler), είσοδοι (αριστερά) και έξοδοι (δεξιά)

Παραπάνω φαίνεται το σχηματικό του output scheduler με τα σήματα εισόδου και τα σήματα εξόδου. Τα clk και rst (clock,reset) χρησιμοποιούνται για το ίδιο λόγο που αναφέρεται και στις παραπάνω παραγράφους. Τα fifo_out1, fifo_out2, fifo_out3, fifo_out4, fifo_out5 είναι τα πακέτα (flits των πακέτων) που προωθούνται στις θύρες εξόδου και τα οποία εξήλθαν από τις FIFO. Τα σήματα εισόδου rd_en1, rd_en2, rd_en3, rd_en4, rd_en5 μαζί με τους counters τους cnt1, cnt2, cnt3, cnt4, cnt5 όταν εισέρχονται block διαβάζονται από τον arbiter για να ξέρει από ποια FIFO διάβασε πακέτο κάθε χρονική στιγμή, σε ποιο flit του πακέτου βρίσκεται και χρησιμοποιούνται μέσα στον scheduler ως σήματα έλεγχου. Στη συνέχεια ο scheduler επιλέγει σε ποιες θύρες εξόδου θα στείλει τα πακέτα με τον τρόπο που αναφέραμε.

Τα σήματα εξόδου value_out1, value_out2, value_out3, value_out4, value_out5 του scheduler είναι τα πακέτα δεδομένων με αποφασισμένη πλέον τη διαδρομή εξόδου και προωθούνται στις τελικές θύρες εξόδου. Τα flags χρησιμοποιούνται όπως αναφέρεται παραπάνω και ακλουθούν τα πακέτα δεδομένων σηματοδοτώντας την έξοδο ώστε να μπορούμε να βλέπουμε σε ποια θύρα εξόδου υπάρχουν πακέτα.

Το μοντέλο αυτό είναι ένα αρχείο VHDL module και αναπτύχτηκε με VHDL κώδικα μέσω του text editor του λογισμικού Xilinx ISE (πολυπλοκότητα :420 γραμμές κώδικα).

5.2 NoC Router 5 x 5 θυρών

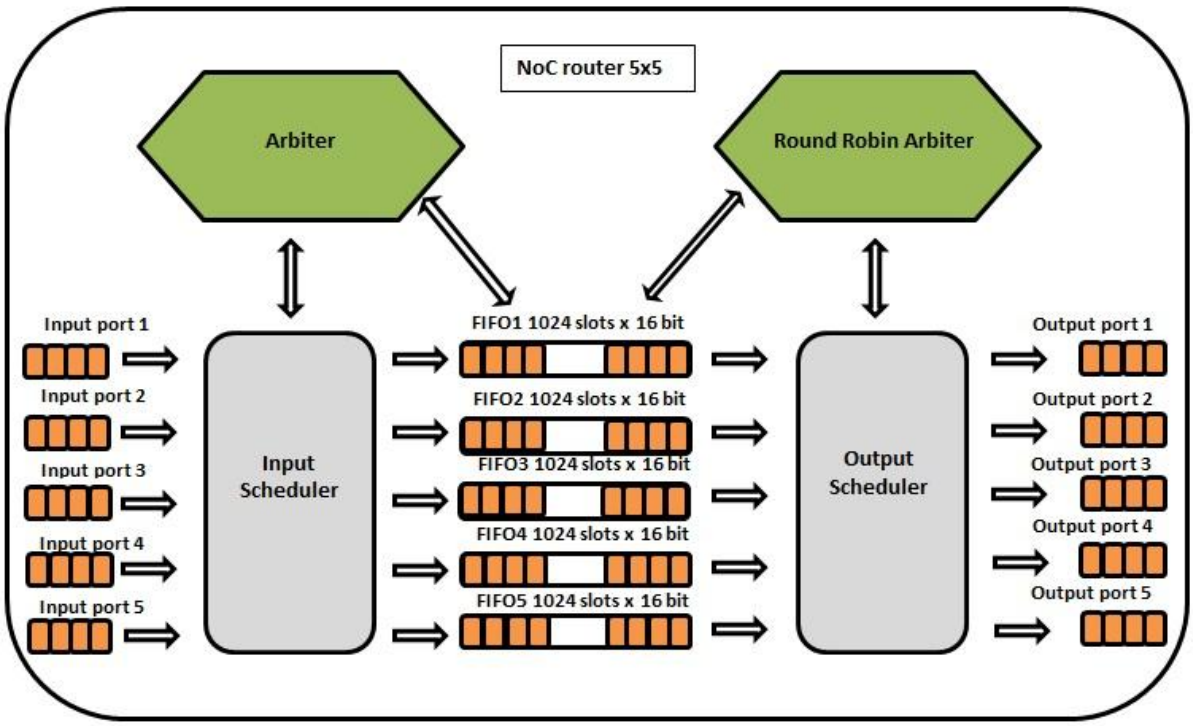
Στο σημείο αυτό περιγράφεται το κύριο μέρος του προγράμματος. Χρησιμοποιώντας τα αρχεία προέλευσης που περιγράψαμε παραπάνω μπορούμε πλέον να κατασκευάσουμε **το NoC Router**. Στο μοντέλο αυτό συνδυάζουμε τα παραπάνω μοντέλα και δυο μοντέλα διαιτητών (arbiters) που εξασφαλίζουν τη σωστή επικοινωνία μέσα στο NoC.

Για την περιγραφή αυτού του μοντέλου αρχικοποιηθήκαν (instantiated) ένας χρονοδρομολογητής εισόδου (input scheduler), ένας χρονοδρομολογητής εξόδου (output scheduler) και πέντε μνήμες FIFO. Επίσης αναπτύχτηκαν δυο μοντέλα arbiter που σε συνδυασμό με τους δυο schedulers εξασφαλίζουν ότι δεν υπάρχουν απώλειες ή υπερκαλύψεις πακέτων και κάθε θύρα εισόδου ή εξόδου θα έχει τα ίδια δικαιώματα για εγράφη ή ανάγνωση σε ή από κάποια FIFO.

Ο πρώτος arbiter παρακολουθεί τις FIFOs ώστε να ξέρει ποιες από αυτές είναι διαθέσιμες και ποιες από αυτές είναι απασχολημένες, ποιες από αυτές είναι άδειες και ποιες από αυτές είναι γεμάτες. Ο arbiter παρέχει τις πληροφορίες αυτές στον Χρονοδρομολογητή εισόδου όταν του ζητηθεί (όταν δηλαδή υπάρχει εισερχόμενο πακέτο) ώστε να γνωρίζει σε ποια FIFO θα προωθήσει το πακέτο.

Ο δεύτερος arbiter χρησιμοποιείτε κατά την ανάγνωση των πακέτων από τις FIFO και στηρίζει τη λειτουργία του αλγόριθμο round robin, όπως αναφέρεται παραπάνω. Εν συντομία, κάθε FIFO έχει τα ίδια δικαιώματα με όλες τις άλλες και μόνο από μια FIFO θα γίνεται ανάγνωση κάθε χρονική στιγμή. Με τον τρόπο αυτό αποφεύγονται οι υπερκαλύψεις και οι απώλειες πακέτων. Ο χρόνος που δίνεται σε κάθε FIFO για ανάγνωση ισοδυναμεί με το χρόνο ανάγνωσης ενός πακέτου, δηλαδή τέσσερις κύκλους ρολογιού (κάθε πακέτο αποτελείτε από τέσσερα flit επί ένα κύκλο ρολογιού για ανάγνωση ενός flit).

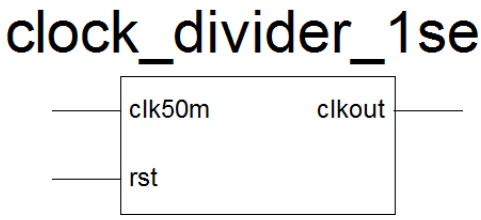
Όλα αυτά τα modules ενσωματώνονται σε ένα ενιαίο module, το NoC Router. Το NoC Router έχει επίσης ένα κοινό ρολόι και ένα κοινό reset για όλο το project. Επίσης, διαθέτει κάποια flags ώστε να γνωρίζουν οι arbiter που γράφεται και από πού διαβάζεται κάποιο πακέτο. Το μοντέλο NoC Router είναι ένα αρχείο VHDL module και αναπτύχτηκε με VHDL κώδικα μέσω του text editor του λογισμικού Xilinx ISE (πολυπλοκότητα :343 γραμμές κώδικα).



Εικόνα 5.6: Σχηματικό NoC Router 5 x 5 θυρών

Στην προσομοίωση λειτουργίας πάνω στο board θα έχουμε την δυνατότητα να επιλέγουμε το τύπο του ρολογιού που θέλουμε να χρησιμοποιήσουμε. Το Spartan 3S διαθέτει εξωτερικό ρολόι συχνοτήτων 50MHz, επειδή όμως η συχνότητα αυτή είναι πολύ μεγάλη έχουμε δυο τρόπους. Μπορούμε να δίνουμε είσοδο ρολόι χειροκίνητα με ένα button πάνω στο board ή διαφορετικά από το ρολόι του Spartan 3 στο οποίο όμως χρησιμοποιούμε ένα διαιρέτη συχνότητας.

Στη δεύτερη μέθοδο με το διαιρέτη συχνότητας αναπτύχθηκε ένα VHDL module (40 γραμμές κώδικα) το οποίο μετατρέπει τη συχνότητα των 50 MHz στη περιοχή του 1 Hz (ρολόι ενός δευτερολέπτου) . Αυτό μας δίνει το πλεονέκτημα ότι οι αλλαγές θα γίνονται ανά ένα δευτερόλεπτο και θα μπορούμε να τις βλέπουμε πάνω στο Spartan 3 Starter Kit board. Η λογική είναι απλή, το μοντέλο one second clock παίρνει ως είσοδο το ρολόι των 50 MHz και χρησιμοποιεί ένα μετρητή (0 έως 50 000 000) που αυξάνεται ανά ακμή ανόδου ρολογιού. Η έξοδος του μοντέλου αυτού είναι μηδέν (0) όταν ο μετρητής είναι 0 εως 25 000 000 και άσσος (1) όταν είναι 25 000 000 έως 50 000 000. Έτσι καταφέραμε με αυτό το διαιρετή συχνότητας να μειώσουμε τη συχνότητα από τα 50 MHz στο 1 Hz.



Εικόνα 5.7: Σχηματικό σύμβολο του διαιρέτη συχνότητας ενός δευτερολέπτου, είσοδοι (αριστερά) και έξοδος (δεξιά)

5.3 Προσομοίωση του NoC Router στο Spartan 3 Starter kit

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο για την εποπτεία της σωστής επικοινωνίας μέσα στο NoC Router γίνεται προσομοίωση της λειτουργίας του στο Spartan 3 Starter kit board. Το NoC Router διαθέτει πέντε θύρες εισόδου και πέντε θύρες εξόδου καθιστώντας αδύνατο να μπορούμε να παρατηρούμε και τις πέντε θύρες εξόδου ταυτόχρονα. Άλλο ένα θέμα που τίθεται είναι ο τρόπος με

τον όποιο δίνουμε εισόδους πακέτων στις θύρες εισόδου. Θα μπορούσαμε να χρησιμοποιήσουμε την σειριακή θύρα RS-232 και να δίνουμε τιμές εισόδων μέσω του υπολογιστή, αλλά αυτή η μέθοδος δεν μας επιτρέπει να δίνουμε τιμές σε παραπάνω από μια θύρα ανά χρονική στιγμή.

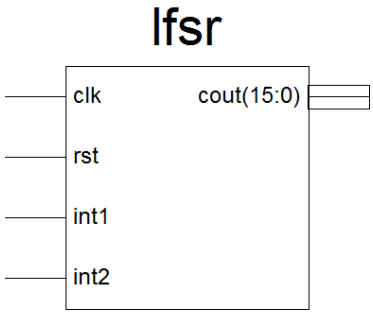
Για τους λόγους αυτούς δημιουργήθηκε μια μονάδα, η οποία βρίσκεται στην υψηλότερη θέση της ιεραρχίας, η οποία, αρχικοποιεί ένα NoC Router και στη συνεχεία δημιουργεί πακέτα για τις θύρες εισόδου. Επιπλέον, χρησιμοποιεί ένα πολυπλέκτη οδηγώντας τις θύρες εξόδου στην οθόνη επτά τμημάτων του Spartan 3 Starter Kit board. Η μονάδα αυτή ονομάζεται **Top Level** και ο τρόπος λειτουργίας της παρουσιάζεται παρακάτω.

5.3.1 Δημιουργία πακέτων για Προσομοίωση του NoC Router

Για την δημιουργία των πακέτων και τη προσομοίωση λειτουργίας στο board Spartan 3 Starter Kit board θα χρησιμοποιήσουμε πέντε linear feedback shift registers οι οποίοι δημιουργούν τυχαία πακέτα μεγέθους οκτώ Bytes (64 bit) ανά χρονικό διάστημα τεσσάρων κύκλων ρολογιού. Τα πακέτα αυτά θα χρησιμοποιούμε ως είσοδο δεδομένων στις πέντε πόρτες εισόδου του NoC router. Ο χρήστης με πέντε διακόπτες (switches) πάνω στο board Spartan 3S θα μπορεί να επιλεγεί σε ποια πόρτα εισόδου κάθε χρονική στιγμή θα μπορεί να δίνει ως είσοδο δεδομένων τα πακέτα που δημιουργούνται από τους linear feedback shift registers.

Κάθε ένας linear feedback shift register δημιουργεί ένα τυχαίο flit του πακέτου κάθε φορά που δέχεται μια ακμή ανόδου ρολογιού. Ο χρήστης με χρήση των διακοπτών επιλέγει σε ποιες θύρες εισόδου του NoC Router θέλει να δώσει δεδομένα και όταν ενεργοποιηθεί ένας διακόπτης η θύρα με την οποία αντιστοιχεί θα δεχτεί ως είσοδο τέσσερα συνεχόμενα flit που αποτελούν το πακέτο. Με το τρόπο αυτό και επειδή κάθε θύρα εισόδου αντιστοιχεί σε ένα διακόπτη έχουμε τη δυνατότητα να δίνουμε εισόδους πακέτων σε πολλαπλές θύρες εισόδου ανά χρονική στιγμή.

Ένας linear feedback shift register είναι ένα VHDL module (34 γραμμές κώδικα) το οποίο έχει περιγραφεί και αρχικοποιείται πέντε φορές (μια για κάθε θύρα εισόδου) με διαφορετικές παραμέτρους εισόδου (δυο τυχαίοι ακέραιοι αριθμοί διαφορετικοί για κάθε έναν linear feedback shift register για δημιουργία τυχαίου flit) δημιουργώντας τυχαία πακέτα, έτσι ώστε σε μια δεδομένη χρονική στιγμή να μην υπάρχουν ταυτόσημα εισερχόμενα πακέτα σε παραπάνω από μια θύρα εισόδου. Με το τρόπο αυτό είναι ευκολότερο στο χρήστη να παρατηρεί την κίνηση των πακέτων που εισέρχονται στο NoC και των πακέτων που εξέρχονται από αυτό. Οι linear feedback shift registers αρχικοποιούνται στο Top Level μοντέλο.



Εικόνα 5.8: Σχηματικό σύμβολο του διαιρέτη linear feedback shift register, είσοδοι (αριστερά) και έξοδος (δεξιά)

5.3.2 Απεικόνιση των πακέτων στην οθόνη επτά τμημάτων

Το Spartan-3 Starter Kit διαθέτει μια οθόνη επτά τμημάτων τεσσάρων ψηφίων το οποίο θα χρησιμοποιήσουμε για να παρατηρούμε τα πακέτα που έφτασαν σε μια θύρα εξόδου. Κάθε ψηφίο έχει επτά κοινά σήματα ελέγχου που ενεργοποιούν μεμονωμένα τμήματα LED. Κάθε χαρακτήρας έχει μια ξεχωριστή είσοδο ελέγχου ενεργοποίησης.

Τα σήματα ελέγχου που ενεργοποιούν τα LED κάθε χαρακτήρα είναι πολυπλεγμένα στο χρόνο (time multiplexed) με αποτέλεσμα όλα τα ψηφία να αναπαριστούν τον ίδιο χαρακτήρα. Λόγο επιμονής της όρασης, ο ανθρώπινος εγκέφαλος αντιλαμβάνεται τα τέσσερα ψηφία να

εμφανίζονται ταυτόχρονα, παρόμοια με τον τρόπο που ο εγκέφαλος αντιλαμβάνεται μια οθόνη τηλεόρασης.

Η παραπάνω τεχνική ονομάζεται ‘’σάρωση’’ και χρησιμοποιείτε σε πάρα πολλές συσκευές επειδή μειώνει τον αριθμό σημάτων εισόδων/εξόδων που απαιτούνται για την απεικόνιση των τεσσάρων ψηφίων. Αν για κάθε επιμέρους τμήμα ήταν αφιερωμένο ένα FPGA pin τότε θα χρειαζόμασταν 32 pin για να αναπαραστήσουμε τέσσερις χαρακτήρες στην οθόνη επτά τμημάτων. Με τη μέθοδο της πολυπλεξίας και την τεχνική της σάρωσης καταφέρνουμε να χρησιμοποιήσουμε μόνο 12 pins. Το μειονέκτημα αυτής της προσέγγισης είναι πρέπει να γίνεται συνεχώς σάρωση των ψηφίων, μικρό τίμημα για την εξοικονόμηση 20 επιπλέον σημάτων εισόδων/εξόδων.

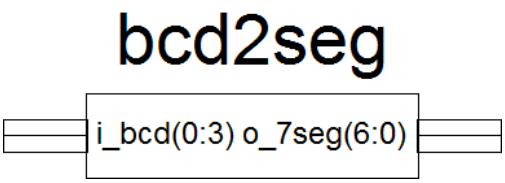
Για να είναι εφικτή η τεχνική της σάρωσης χρησιμοποιούμε ένα ρολόι συχνοτήτων που βρίσκεται πάνω στο Spartan 3 και είναι της τάξης των πενήντα μεγακύκλων (50 MHz). Η συχνότητα αυτή είναι πολύ μεγάλη για την αναπαράσταση τεσσάρων διαφορετικών χαρακτήρων και θα χρησιμοποιήσουμε ένα διαιρέτη συχνότητας, διαφορετικά θα βλέπουμε τον ίδιο χαρακτήρα και στα τέσσερα ψηφία.

Σε μια δεδομένη χρονική στιγμή θα μπορούμε να αναπαριστούμε ένα flit του πακέτου στην οθόνη επτά τμημάτων. Κάθε flit αυτό είναι της τάξεως των 16 bit το καθένα και η αναπαράσταση του θα είναι στη δεκαεξαδική μορφή με τέσσερις χαρακτήρες τεσσάρων bit ο κάθε ένας.

Εικόνα 5.9: Οθόνη επτά τμημάτων του Spartan 3 Starter kit

Επειδή έχουμε πέντε διαθέσιμες πόρτες εξόδου και μπορούμε να παρατηρούμε μόνο μια πόρτα σε μια δεδομένη χρονική στιγμή χρησιμοποιούμε ένα πολυπλέκτη (mux) με είσοδο ελέγχου τριών Bit και επιλεγούμε μια πόρτα εξόδου για την αναπαράσταση των δεδομένων της στην οθόνη επτά τμημάτων. Η επιλογή της εισόδου έλεγχου του πολυπλέκτη γίνεται αυτόματα από το block, ανάλογα σε ποια θύρα εξόδου υπάρχει πακέτο (χρησιμοποιώντας τα flags), χωρίς να χρειάζεται να δώσει ο χρήστης κάποια επιλογή. Ο πολυπλέκτης είναι ένα VHDL αρχείο το οποίο έχει περιγραφεί με VHDL κώδικα (40 γραμμές κώδικα)

Η παραπάνω διαδικασία επιτυγχάνεται χρησιμοποιώντας τέσσερα bcd2seg modules (ένα για κάθε χαρακτήρα του flit). Κάθε ένα από αυτά τα modules παίρνει ως είσοδο ένα ακέραιο αριθμό τεσσάρων δυαδικών ψηφίων (από 0 έως 15 στο δεκαδικό) και τον μετατρέπει σε ένα δυαδικό των επτά bit ώστε να μπορεί να απεικονιστεί στην οθόνη επτά τμημάτων. Ένα bcd2seg είναι ένα VHDL module (50 γραμμές κώδικα) το οποίο έχει περιγραφεί με VHDL και αρχικοποιείται τέσσερις φορές (ένα για κάθε χαρακτήρα) σε ένα module που ονομάζεται todisplay στο οποίο γίνονται και οι σχετικές ενέργειες ώστε να μπορέσουμε να απεικονίσουμε ένα πλήρες flit στην οθόνη επτά τμημάτων του Spartan 3.



Εικόνα 5.10: Σχηματικό σύμβολο ενός bcd2seg, είσοδοι (αριστερά) και έξοδοι (δεξιά)

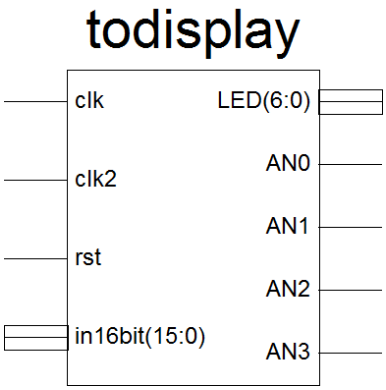
Όπως αναφέραμε προηγουμένως για να μπορέσουμε να απεικονίσουμε τέσσερις διαφορετικούς χαρακτήρες στην οθόνη επτά τμημάτων χρησιμοποιούμε την τεχνική της ‘’σάρωσης’’. Με την χρήση τεσσάρων bcd2seg modules έχουμε την μετατροπή ενός δεκαεξάμπιτου flit σε τέσσερις χαρακτήρες δεκαεξαδικής μορφής. Οι τέσσερις αυτοί χαρακτήρες θα απεικονίζονται ο καθένας στην

αντίστοιχη θέση στην οθόνη επτά τμημάτων. Το μειονέκτημα αυτής της μεθόδου είναι ότι δεν μπορούμε να βλέπουμε ολόκληρο το πακέτο αλλά με τη σειρά τα τέσσερα flit (ένα flit ανά χρονική στιγμή) από τα οποία αποτελείτε. Το γεγονός αυτό οφείλεται στο ότι στην οθόνη επτά τμημάτων δεν είναι εφικτό να εμφανιστούν μεγαλύτερου μεγέθους πληροφορίες.

Για να είναι εφικτό να βλέπουμε τους χαρακτήρες του flit θα συνδέσουμε στο module todisplay το ρολόι συχνοτήτων του Spartan 3 το οποίο είναι της τάξης των 50 MHz. Η συχνότητα αυτή είναι πάρα πολύ μεγάλη και το ανθρώπινο μάτι δεν μπορεί να ξεχωρίσει τι βλέπει σε κάθε τμήμα της οθόνης, γι' αυτό χρησιμοποιούμε ένα διαιρέτη συχνότητας ο οποίος δεν είναι τίποτα άλλο από ένα μετρητή ο οποίος ξεκινά από το μηδέν και σε κάθε ακμή ανόδου του ρολογιού αυξάνεται κατά 1. Όταν ο μετρητής αυτός φτάσει να γίνει 2^{18} (262 264 ακμές ανόδου ρολογιού 50 MHz) τότε σε εκείνη την ακμή του ρολογιού γίνεται ανανέωση της οθόνης και αναπαρίστανται οι τέσσερις χαρακτήρες στην οθόνη επτά τμημάτων.

Στην πραγματικότητα οι χαρακτήρες που εμφανίζονται είναι ίδιοι και στους τέσσερις χαρακτήρες της οθόνης επτά τμημάτων αλλά επειδή αλλάζουν με πολύ μεγάλη ταχύτητα το ανθρώπινο μάτι δεν αντιλαμβάνεται την κίνηση και έτσι μας φαίνονται ότι οι τέσσερις χαρακτήρες είναι διαφορετικοί καθιστώντας έτσι δυνατό να βλέπουμε όλο το flit .

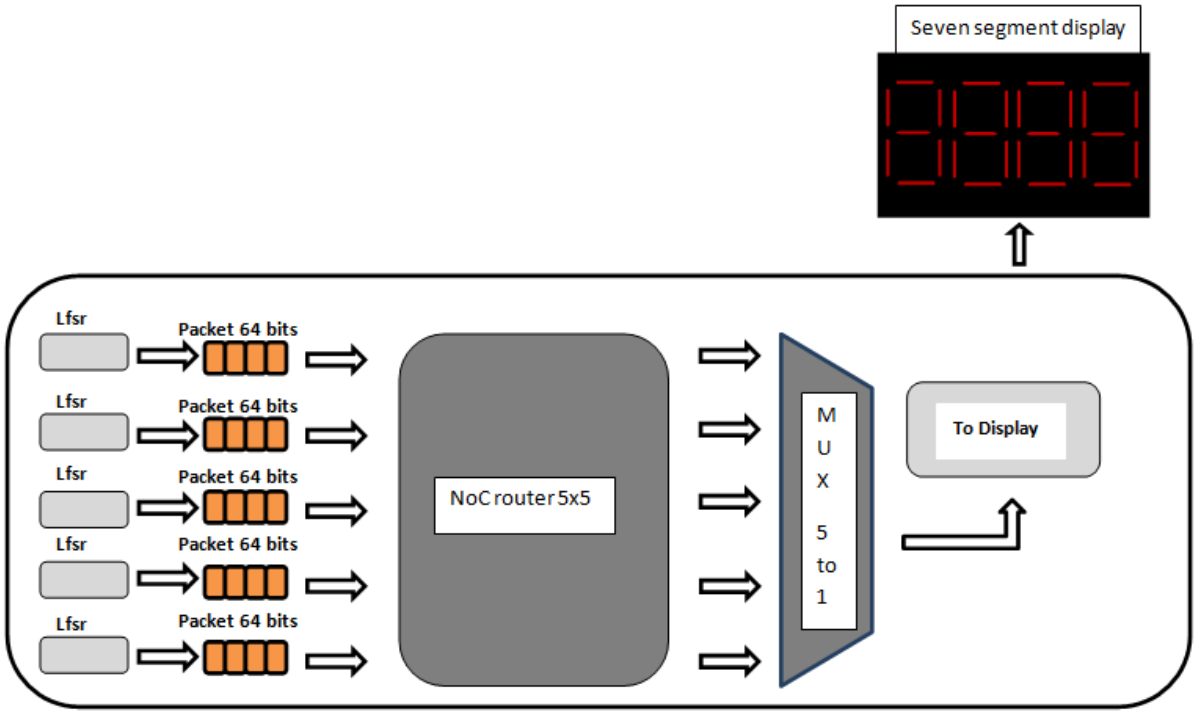
Το module todisplay είναι ένα VHDL αρχείο (105 γραμμές κώδικα) το οποίο παίρνει είσοδο για δυο ρολόγια, ένα με το οποίο είναι συνδεμένο με όλο το υπόλοιπο project (συγχρονισμένο) ενώ η δεύτερη είσοδος ρολογιού συχνοτήτων είναι αποκλειστικά για την περιοχή των 50 MHz επειδή διαιρεί τη συχνότητα σε μια ποιο ιδανική για το ανθρώπινο μάτι αυτή και καθιστά δυνατή την απεικόνιση των πακέτων στην οθόνη επτά τμημάτων. Επίσης διαθέτει μια είσοδο για επανεκκίνηση (reset) και η κύρια εργασία του είναι να παίρνει σαν είσοδο ένα flit 16 bit και να το μετατρέπει σε τέσσερις χαρακτήρες οι όποιοι θα απεικονιστούν στην οθόνη επτά τμημάτων.



Εικόνα 5.11: Σχηματικό σύμβολο μιας μονάδας todisplay, είσοδοι (αριστερά) και έξοδοι (δεξιά)

Εκτός από τα πακέτα που θα μπορούμε να παρατηρούμε στην οθόνη επτά τμημάτων, θα μπορούμε να βλέπουμε και την κεφαλίδα (header) του πακέτου μεγέθους τριών δυαδικών ψηφίων σε τρεις διόδους εκπομπής φωτός (light emitting diodes – LED). Πέντε ακόμα LED θα χρησιμοποιήσουμε ώστε να μπορούμε να βλέπουμε τα flags των πακέτων που μας ειδοποιούν για την παρουσία πακέτου. Το project διαθέτει τρία διαθέσιμα είδη flags, πέντε που μας πληροφορούν σε ποια θύρα εξόδου υπάρχει πακέτο (ένα για κάθε μια), πέντε για βλέπουμε πότε γράφεται πακέτο σε κάποια FIFO και πέντε για να βλέπουμε πότε διαβάζεται πακέτο από κάποια FIFO. Η επιλογή του είδους των flag που θέλουμε να βλέπουμε θα γίνεται από δυο διακόπτες πάνω στο Spartan 3 Starter Kit board.

Για το είδος του ρολογιού που θέλουμε να έχουμε χρησιμοποιούμε ένα διακόπτη για να επιλέγουμε να δίνουμε το ρολόι είτε με ένα κουμπί (button) είτε να χρησιμοποιήσουμε εκείνο του ενός δευτερόλεπτου που αναφερθήκαμε σε προηγούμενη παράγραφο. Τέλος, έχουμε ακόμα ένα κουμπί για επανεκκίνηση ολόκληρου του NoC router.



Εικόνα 5.12: Γενικό σχήμα του top level για προσομοίωση στο Spartan 3 Starter kit

Στο μοντέλο Top Level που χρησιμοποιείται για την προσομοίωση το NoC Router συνολικά συγκεκριμενοποιούνται πέντε Linear feedback shift registers για την δημιουργία των πακέτων εισόδου, ένα NoC Router 5x5 θυρών, ένα πολυπλέκτη πέντε προς ένα και μια μονάδα todisplay μαζί με τα τέσσερα bcd2seg που αρχικοποιεί για την απεικόνιση των πακέτων στην οθόνη επτά τμημάτων έτσι ώστε να είναι δυνατή η παρακολούθηση της λειτουργιάς του NoC.

Το μοντέλο Top Level είναι ένα αρχείο VHDL module και αναπτύχτηκε με VHDL κώδικα μέσω του text editor του λογισμικού Xilinx ISE. Στο επόμενο κεφάλαιο γίνονται αναφορές σε προσομοιώσεις λειτουργίας καθώς και σε μετρήσεις απόδοσης του NoC Router που αναπτύχτηκε (πολυπλοκότητα : 220 γραμμές κώδικα). Ο τρόπος με τον οποίο είναι συνδεμένοι οι είσοδοι και οι έξοδοι που διαθέτει το Spartan 3 περιγράφεται σε ένα αρχείο περιορισμών (User Constrains File – UCF – 38 γραμμές περιορισμών)

6 Αποτελέσματα – Αξιολόγηση NoC Router 5 x 5

Στο κεφάλαιο αυτό θα μελετήσουμε την απόδοση του NoC Router μέσω προσομοιώσεων λειτουργίας στο λογισμικό modelsim και στο board Spartan 3 Starter Kit. Αναφορές θα γίνουν επίσης για το χώρο αποθήκευσης που καταλαμβάνει, την ενέργεια που καταναλώνει, την απόδοση σε διάφορες ταχύτητες, οι καθυστερήσεις που προκαλούνται μέσω διαφορετικών σεναρίων λειτουργίας. Υπενθυμίζουμε στο σημείο αυτό ότι και από τις πέντε θύρες εισόδου πιθανός προορισμός μπορεί να είναι οποιαδήποτε από τις πέντε θύρες εξόδου, αυτό σημαίνει ότι ο καθένας μπορεί να στέλνει στον καθένα. Πιθανά σενάρια λειτουργίας μπορεί να περιλαμβάνουν μια θύρα εισόδου να στέλνει πακέτα σε όλες τις θύρες εξόδου ή οι πέντε θύρες εισόδου να στέλνουν πακέτα σε μια θύρα εξόδου και διάφορες παραλλαγές των σεναρίων αυτών.

Ένα ενδιαφέρων ίσως σενάριο θα ήταν να φτάνουν πακέτα σε όλες τις θύρες εισόδου με μεγάλη συχνότητα την ίδια στιγμή για μεγάλο χρονικό διάστημα με προορισμό όλες τις θύρες εξόδου. Στην περίπτωση αυτή κάποια στιγμή οι buffers θα γεμίσουν και από το σημείο αυτό τυχόν εισερχόμενα πακέτα θα απορρίπτονται οδηγώντας έτσι σε απώλειες πακέτων. Η απώλεια πακέτων δεν είναι αποδεκτή, για το λόγο αυτό κάθε μια από τις FIFO διαθέτει χώρο καταχώρησης 256 πακέτων (ή 1024 flit) και σε κανονικές συνθήκες λειτουργίας του NoC Router είναι αδύνατον να γεμίσουν. Για την πλήρη επίγνωση όμως της λειτουργίας του NoC θα δούμε μια προσομοίωση λειτουργίας του χρησιμοποιώντας μικρότερες FIFO (π.χ. τεσσάρων θέσεων) και πως κάτι τέτοιο οδηγεί σε απώλειες πακέτων. Στις παραγράφους που ακολουθούν θα αναλυθούν τα διάφορα σενάρια λειτουργίας του NoC Router και θα γίνουν συγκρίσεις σε θέματα χώρου αποθήκευσης και καθυστερήσεων.

6.1 Επισκόπηση του NoC

Μετά από μια επιτυχημένη σύνθεση του NoC Router μπορούμε, μέσα από την αναφορά HDL που παράγει το λογισμικό ISE της Xilinx, να παρατηρήσουμε τα λογικά στοιχεία που χρησιμοποιούνται για την υλοποίηση του. Τα λογικά στοιχεία που συμπεριλαμβάνει το project αποτυπώνονται στον παρακάτω πίνακα:

ROMs	**4**
-16x7-bit ROM	4
Counters	**12**
-19-bit up counter	1
-2-bit up counter	10
-32-bit up counter	1
Registers	**137**
-1-bit register	80
-16-bit register	46
-3-bit register	8
-5-bit register	2
-7-bit register	1
Comparators	**2**
-19-bit comparator greater	1
-32-bit comparator greatequal	1
Multiplexers	**20**
1-bit 16-to-1 multiplexer	10
1-bit 4-to-1 multiplexer	5
3-bit 8-to-1 multiplexer	5
XORs	**5**
-1-bit xor2	5

Πίνακας 6.1: Λογικά στοιχεία του NoC Router

Τα στοιχεία αυτά καταλαμβάνουν χώρο και λογικές πύλες πάνω στο FPGA του Spartan 3 όπως φαίνεται στον παρακάτω πίνακα:

Logic Utilization		
Number of Slice Flip Flops	1,188 out of 3,840	30%
Number of 4 input LUTs	1,221 out of 3,840	31%
Logic Distribution		
Number of occupied Slices -Number of Slices containing only related logic -Number of Slices containing unrelated logic	1,104 out of 1,920 1,104 out of 1,104 0 out of 1,104	57% 100% 0%
Total Number of 4 input LUTs -Number used as logic -Number used as a route-thru	1,333 out of 3,840 1,221 112	34%
Number of bonded IOBs	30 out of 173	17%
Number of BRAMs:	5 out of 12	41%
Number of GCLKs	2 out of 8	25%

Πίνακας 6.2: Αποθηκευτικός Χώρος που καταλαμβάνουν τα λογικά στοιχεία1

Τέλος, όσον αφορά την ταχύτητα απόκρισης του NoC Router για μια περίοδο ρολογιού 10,408 ns η συχνότητα λειτουργίας του είναι της τάξης των 96,084 MHz. Καθυστερήσεις παρατηρούνται μόνο κατά την είσοδο πακέτων στα buffers και κατά την έξοδο από αυτά καθώς αυτά δρομολογούνται από τις θύρες εισόδου στις θύρες εξόδου.

6.2 Αποτελέσματα του NoC Router

Πριν να ξεκινήσουμε να αναλύουμε κάθε σενάριο θα πρέπει να αναφερθούμε σε κάποια βασικά στοιχεία του NoC Router. Ένα πακέτο δεδομένων αποτελείτε από τέσσερα flit. Από την στιγμή που ένα πακέτο φτάσει σε μια θύρα εισόδου χρειάζεται τέσσερις κύκλους ρολογιού ώστε να αποθηκευτεί στο buffer (FIFO). Κατά την έξοδο κάποιου πακέτου από ένα buffer, αφού έχει αποφασιστεί η θύρα εξόδου για την οποία προορίζεται, απαιτούνται τέσσερις κύκλοι ρολογιού μέχρι να φτάσει στην θύρα εξόδου.

Για να μπορέσουμε να προσομοιώσουμε τη λειτουργία του NoC Router δημιουργούμε εμείς τα πακέτα μέσω lfsr (όπως αναφερθήκαμε στο προηγούμενο κεφάλαιο). Ο χρόνος δημιουργίας για ένα πακέτο ισούται με το χρόνο τεσσάρων κύκλων ρολογιού (ένα κύκλο για κάθε flit). Οπότε, οι περιπτώσεις λειτουργίας που θα δούμε περιλαμβάνουν διάφορα σενάρια που αφορούν την συχνότητα με την οποία γεννιούνται πακέτα και ποιες θύρες εισόδου και εξόδου αφορούν.

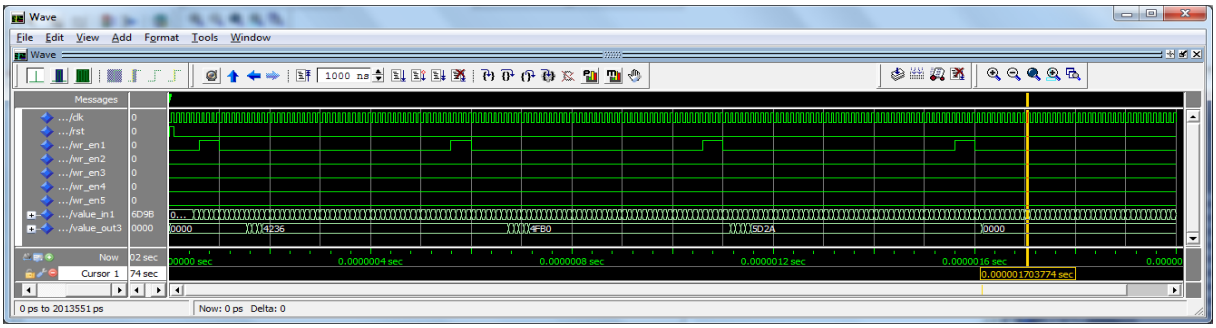
Στις παραγράφους που ακολουθούν θα γίνουν υποθέσεις σεναρίων λειτουργίας και προσομοιώσεις αυτών ώστε να γίνει σύγκριση αποτελεσμάτων. Μέσω αυτών των αποτελεσμάτων θα ήμαστε σε θέση πλέον να γνωρίζουμε πως το NoC Router θα λειτουργεί υπό κάθε συνθήκη. Δημιουργούμε τρεις περιπτώσεις προσομοίωσης λειτουργίας σύμφωνα με την συχνότητα με την οποία δημιουργούνται πακέτα (σε 1 πακέτο ανά 50 κύκλους ρολογιού, 1 πακέτο ανά 20 κύκλους και 1 πακέτο ανά 10 κύκλους) και σε διαφορετικά σενάρια κάθε περίπτωσης ανάλογα με το ποιες θύρες εισόδου/εξόδου αφορούν (από μια θύρα σε μια θύρα, από μια θύρα σε πολλές θύρες, από πολλές θύρες σε μια θύρα και από πολλές θύρες σε πολλές θύρες).

6.2.1 Περίπτωση 1η: Δημιουργία ενός πακέτου ανά πενήντα κύκλους ρολογιού

Η συχνότητα με την οποία τα πακέτα θα φτάνουν σε μια ή παραπάνω θύρες εισόδου είναι της τάξης του ενός πακέτου ανά πενήντα κύκλους ρολογιού. Θα δούμε διαφορετικά σενάρια της περίπτωσης αυτής απασχολώντας διαφορετικές θύρες εισόδου και εξόδου και πως επηρεάζεται η απόκριση του NoC σε κάθε σενάριο. Πρέπει να πούμε ότι στην περίπτωση αυτή ο φόρτος κυκλοφορίας είναι πολύ μικρός οπότε δεν περιμένουμε ιδιαίτερα μεγάλες καθυστερήσεις, παρ' όλα αυτά όμως είναι απαραίτητο να γνωρίζουμε πως θα συμπεριφέρεται το NoC.

6.2.1.1 Σενάριο 1ο: Από μια θύρα εισόδου σε μια θύρα εξόδου

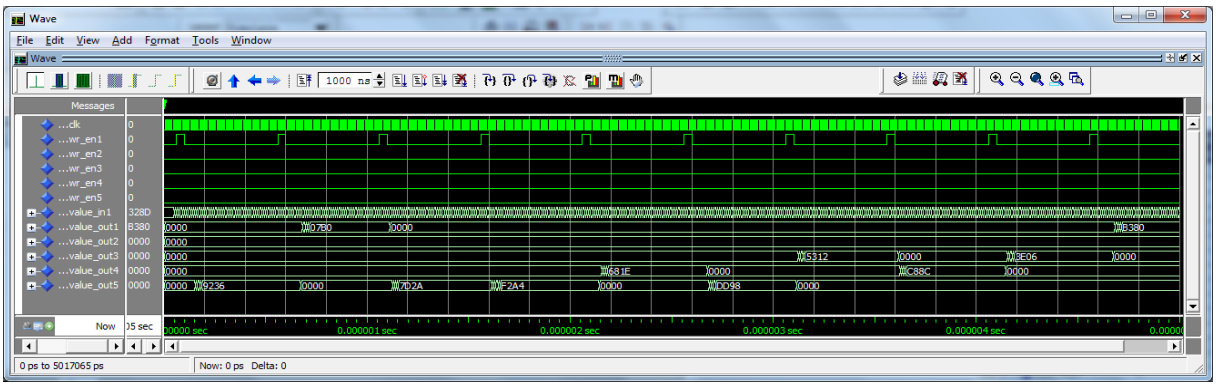
Έστω ότι δημιουργούμε ένα πακέτο ανά πενήντα κύκλους ρολογιού και το πακέτο αυτό από μια θύρα εισόδου θα προωθηθεί σε μια θύρα εξόδου. Επειδή η συχνότητα δημιουργίας πακέτων είναι πολύ μικρή δεν δημιουργούνται μεγάλες καθυστερήσεις και το buffer έχει αδειάσει μέχρι να φτάσει νέο πακέτο. Από τη στιγμή που ένα πακέτο φτάνει σε μια θύρα εισόδου απαιτούνται μόνο τρεις με έξι κύκλους ρολογιού μέχρι να φτάσει στην θύρα εξόδου (στη συγκεκριμένη περίπτωση το πακέτο προωθείτε στην τρίτη θύρα εξόδου αφού έχει κεφαλίδα πακέτου ‘’010’’). Η προσομοίωση με το modelsim φαίνεται παρακάτω.



Εικόνα 6.1: Προσομοίωση πρώτου σεναρίου πρώτης περίπτωσης

6.2.1.2 Σενάριο 2ο: Από μια θύρα εισόδου σε πολλές θύρες εξόδου

Το σενάριο αυτό δεν διαφέρει από το προηγούμενο απλούστατα επειδή δεν αλλάζει η συχνότητα με την οποία φτάνουν τα πακέτα σε μια θύρα εισόδου και η καθυστέρηση δρομολόγησης κάποιου πακέτου προς μια θύρα εξόδου θα είναι η ίδια με τη προηγούμενη (τρεις με έξι κύκλους ρολογιού), οποιαδήποτε θύρα εξόδου και αν αφορούν τα πακέτα. Η προσομοίωση λειτουργίας του σεναρίου αυτού φαίνεται παρακάτω.



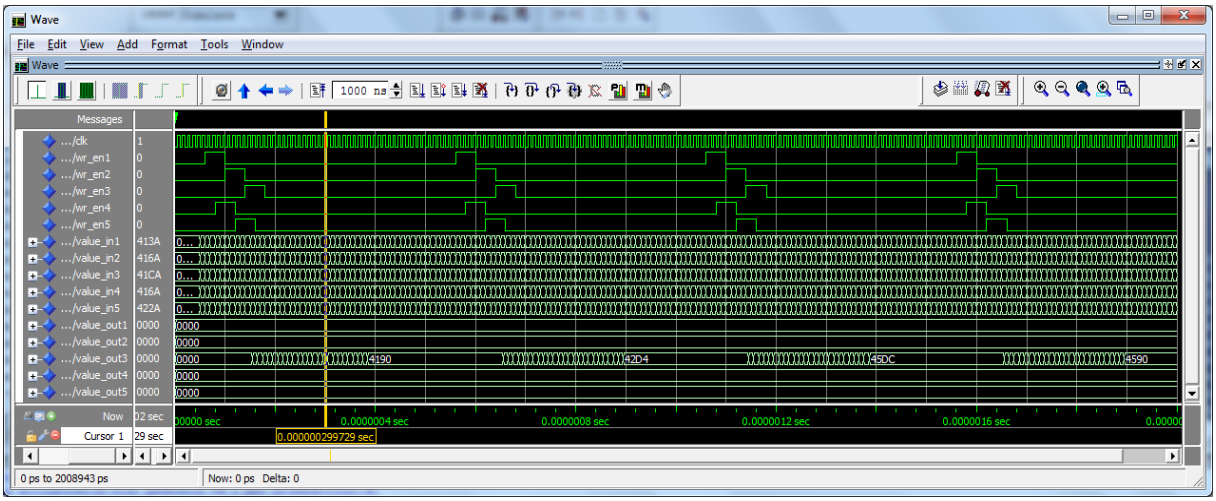
Εικόνα 6.2: Προσομοίωση δευτέρου σεναρίου πρώτης περίπτωσης

6.2.1.3 Σενάριο 3ο: Από πολλές θύρες εισόδου σε μια θύρα εξόδου

Αν και σπάνιο αυτό το σενάριο, τυχαίνει μερικές φορές να φτάνουν σε πολλές θύρες εισόδου πακέτα με προορισμό μια και μόνο θύρα εξόδου και πρέπει να γνωρίζουμε πως θα συμπεριφερθεί το NoC Router σε περίπτωση που συμβεί κάτι τέτοιο. Ένα τέτοιο σενάριο δεν δημιουργεί προβλήματα δρομολόγησης αφού κάθε εισερχόμενο πακέτο από διαφορετική θύρα αποθηκεύεται σε διαφορετικό buffer και το NoC ελέγχει διαδοχικά όλα buffer για τυχόν πακέτα προς δρομολόγηση .

Σύμφωνα με τον αλγόριθμο που λειτουργεί το NoC router τα πακέτα δεν φτάνουν στις θύρες εξόδου με την ίδια σειρά με την οποία έφτασαν στις θύρες εισόδου επειδή ο αλγόριθμος round robin που εκτελείτε ελέγχει διαδοχικά τα buffers για την ύπαρξη πακέτων προς δρομολόγηση. Ο έλεγχος αυτός εκτελείτε κυκλικά ελέγχοντας πρώτα το πρώτο buffer, μετά το δεύτερο κ.ο.κ.. Μετά τον έλεγχο

του πέμπτου buffer ξαναγυρνά στο πρώτο. Έτσι πακέτα που μπορεί έφτασαν αργότερα σε υψηλότερη FIFO τυχαίνει καμιά φορά να δρομολογηθούν νωρίτερα. Το γεγονός αυτό δεν παρουσιάζει πρόβλημα και είναι αποδεκτός σαν τρόπος λειτουργιάς εφόσον δεν υπάρχουν απώλειες.

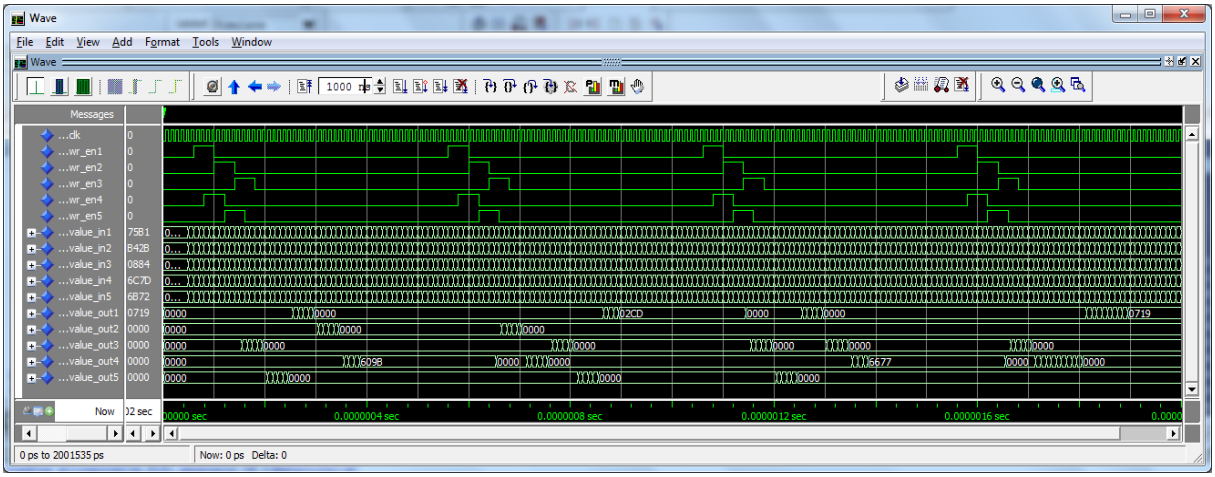


Εικόνα 6.3: Προσομοίωση τρίτου σεναρίου πρώτης περίπτωσης

Από την προσομοίωση που φαίνεται παραπάνω και για το λόγο που αναφέραμε προηγουμένως οι καθυστερήσεις δρομολόγησης των πακέτων είναι διαφορετικές για κάθε θύρα και κυμαίνονται στην περιοχή των τριών κύκλων ρολογιού μέχρι τους είκοσι κύκλους ρολογιού. Οι καθυστερήσεις αυτές δεν αλλάζουν στο χρονικό άξονα επειδή η συχνότητα δημιουργίας πακέτων είναι πολύ μικρή και τα buffers έχουν αδειάσει μέχρι να φτάσουν καινούρια πακέτα.

6.2.1.4 Σενάριο 4ο: Από πολλές θύρες εισόδου σε πολλές θύρες εξόδου

Το σενάριο αυτό είναι το ποιο σημαντικό διότι αναπαριστά μια συνθήκη λειτουργίας που είναι κοντά στην πραγματικότητα όπου πακέτα φτάνουν σε όλες τις θύρες εισόδου σε διαφορετικές χρονικές στιγμές και τα πακέτα αυτά προωθούνται σε διαφορετικές θύρες εξόδου. Λόγο του ότι η συχνότητα δημιουργίας πακέτων είναι πολύ μικρή δεν δημιουργούνται ιδιαίτερα μεγάλες καθυστερήσεις.



Εικόνα 6.4: Προσομοίωση τέταρτου σεναρίου πρώτης περίπτωσης

Όπως και στο προηγούμενο σενάριο τα πακέτα δεν φτάνουν στις θύρες εξόδου του NoC με την ίδια σειρά με την οποία έφτασαν στις θύρες εισόδου επειδή ο αλγόριθμος round robin που εκτελείτε ελέγχει διαδοχικά τα buffers και συμβαίνει μέχρι να ελέγξει τα τελευταία buffers να έχουν

ήδη εισέρθει πακέτα στα ποιο πάνω buffers. Για τον ίδιο αυτό λόγο, όπως και παραπάνω, οι καθυστερήσεις δρομολόγησης των πακέτων είναι διαφορετικές για κάθε θύρα και κυμαίνονται από τους τρεις μέχρι τους είκοσι κύκλους ρολογιού. Οι καθυστερήσεις αυτές δεν αλλάζουν στο χρονικό άξονα επειδή η συχνότητα δημιουργίας πακέτων είναι πολύ μικρή και τα buffers έχουν αδειάσει μέχρι να φτάσουν καινούρια πακέτα.

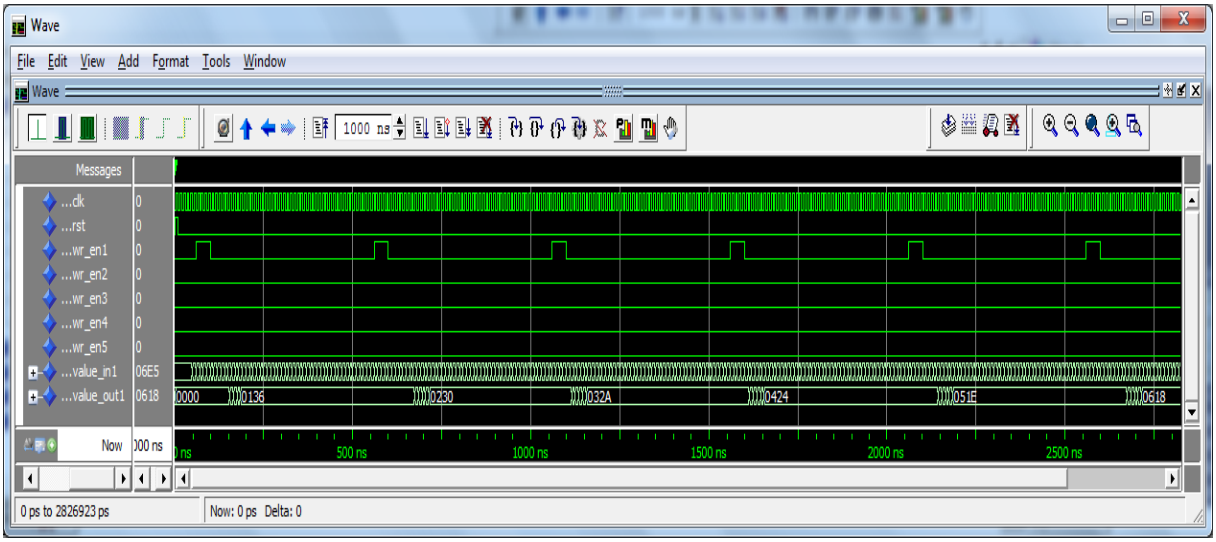
Σαν συμπέρασμα μπορούμε να πούμε ότι στα δυο τελευταία σενάρια οι καθυστερήσεις είναι οι ίδιες και τα πακέτα δρομολογούνται με την ίδια σειρά στις θύρες εξόδου. Επειδή το μοτίβο με το όποιο φτάνουν τα πακέτα στις θύρες εισόδου είναι το ίδιο και αποθηκεύονται σε διαφορετικά buffers, σύμφωνα με τον αλγόριθμο που λειτουργεί το NoC τα πακέτα θα φτάσουν στις θύρες εξόδου με τον ίδιο τρόπο και στα δυο σενάρια άσχετα σε ποια θύρα εξόδου προωθούνται.

6.2.2 Περίπτωση $2^{η}$: Δημιουργία ενός πακέτου ανά είκοσι κύκλους ρολογιού

Στην προηγούμενη περίπτωση είδαμε ότι δεν δημιουργούνται προβλήματα ούτε στην δρομολόγηση αλλά ούτε στις καθυστερήσεις. Τα buffers αδειάζουν πριν προλάβουν να αποθηκεύσουν νέα πακέτα με αποτέλεσμα οι καθυστερήσεις δρομολόγησης των πακέτων να είναι σχετικά σταθερές. Στην παρούσα περίπτωση έχουμε ένα ποιο ιδανικό περιβάλλον επικοινωνίας (μέσου φόρτου κυκλοφορίας) όπου τα πακέτα φτάνουν στις θύρες εισόδου ανά είκοσι κύκλους ρολογιού.

6.2.2.1 Σενάριο $1^{ο}$: Από μια θύρα εισόδου σε μια θύρα εξόδου

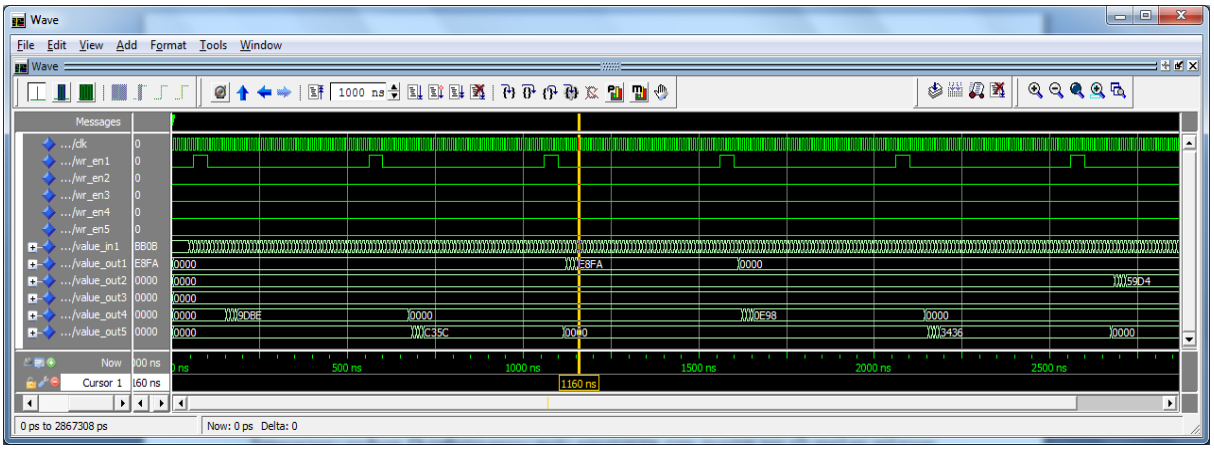
Το σενάριο αυτό δεν διαφέρει καθόλου από το πρώτο σενάριο της πρώτης περίπτωσης παραπάνω. Το μοναδικό που αλλάζει είναι η συχνότητα με την οποία δημιουργούνται τα πακέτα αλλά πάλι η καθυστέρηση δρομολόγησης είναι τρεις με έξι κύκλους ρολογιού όπως στην παραπάνω περίπτωση. Το buffer αδειάζει πριν φτάσει νέο πακέτο αρά και δεν δημιουργούνται επιπρόσθετες καθυστερήσεις. Η προσομοίωση φαίνεται παρακάτω.



Εικόνα 6.5: Προσομοίωση πρώτου σεναρίου δεύτερης περίπτωσης

6.2.2.2 Σενάριο $2^{ο}$: Από μια θύρα εισόδου σε πολλές θύρες εξόδου

Το σενάριο αυτό δεν διαφέρει από το προηγούμενο, απλούστατα επειδή δεν αλλάζει η συχνότητα με την οποία φτάνουν τα πακέτα στην συγκεκριμένη θύρα εισόδου και η καθυστέρηση προκειμένου να φτάσει το πακέτο στην θύρα εξόδου θα είναι η ίδια (τρεις με έξι κύκλους ρολογιού), οποιαδήποτε θύρα εξόδου και αν αφορούν τα πακέτα αυτά. Η προσομοίωση λειτουργίας του σεναρίου αυτού από το modelsim φαίνεται παρακάτω.

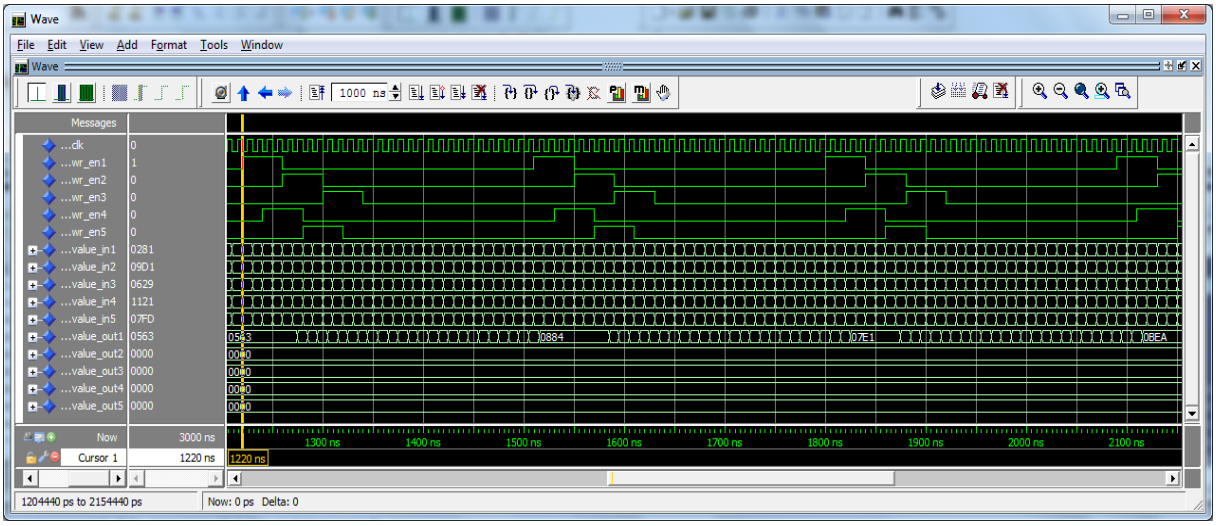


Εικόνα 6.6: Προσομοίωση δεύτερου σεναρίου δεύτερης περίπτωσης

Πρέπει ακόμα να προσθέσουμε ότι το σενάριο αυτό δεν διαφέρει από το αντίστοιχο σενάριο της πρώτης περίπτωσης αφού η καθυστέρηση δρομολόγησης είναι η ίδια. Το μοναδικό που αλλάζει είναι η συχνότητα δημιουργίας πακέτων, ωστόσο το buffer αδειάζει πριν φτάσουν νέα πακέτα και η καθυστέρηση είναι πάντα η ίδια.

6.2.2.3 Σενάριο 3ο: Από πολλές θύρες εισόδου σε μια θύρα εξόδου

Όπως αναφέραμε και στο αντίστοιχο σενάριο της πρώτης περίπτωσης το γεγονός να φτάνουν πακέτα από όλες τις θύρες εισόδου με προορισμό μια μόνο θύρα εξόδου για μεγάλο χρονικό διάστημα είναι κάπως σπάνιο παρόλα αυτά για μια ποιο σφαιρική αντίληψη πρέπει να γνωρίζουμε πως θα συμπεριφερθεί το NoC router. Η συχνότητα δημιουργίας πακέτων είναι ιδανική και δεν υπάρχουν απώλειες αλλά η θύρα εξόδου δέχεται συνεχώς πακέτα στον άξονα του χρόνου. Η προσομοίωση με το modelsim φαίνεται παρακάτω.



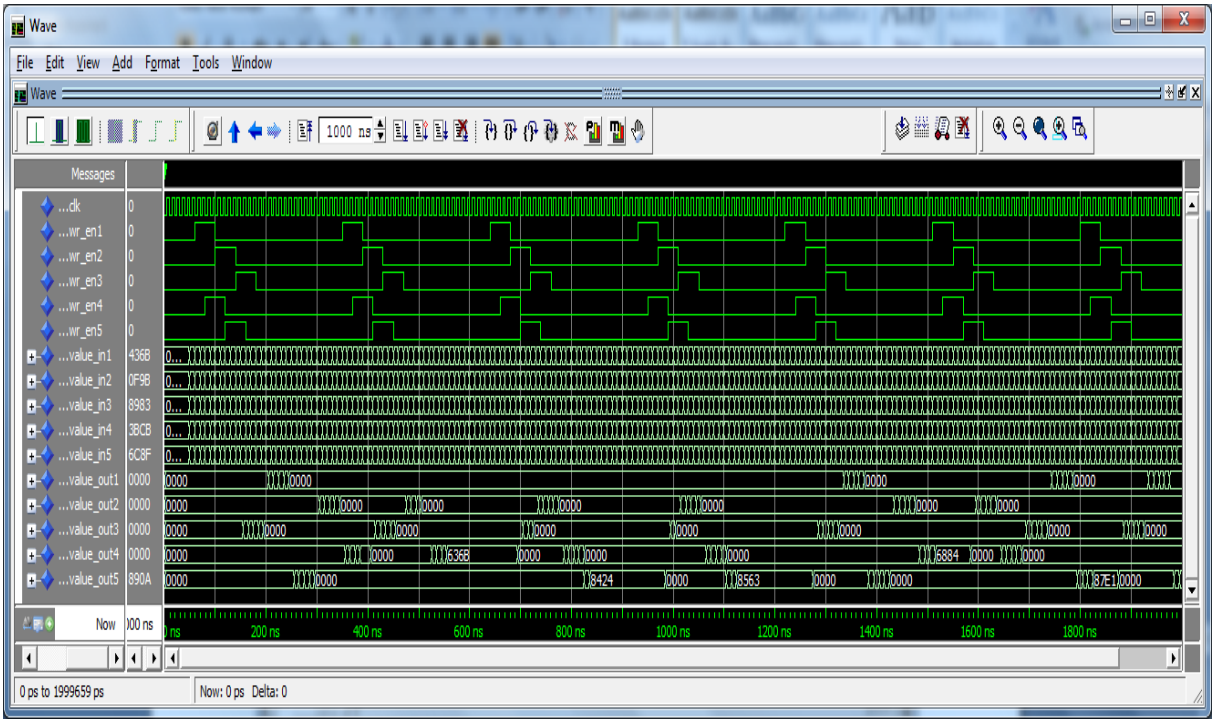
Εικόνα 6.7: Προσομοίωση τρίτου σεναρίου δεύτερης περίπτωσης

Το σενάριο αυτό δεν δημιουργεί προβλήματα δρομολόγησης αφού τα buffers αδειάζουν πριν φτάσουν νέα πακέτα στις θύρες εισόδου και οι καθυστερήσεις είναι σχετικά σταθερές. Τα πακέτα μπορεί να εμφανιστούν στις θύρες εξόδου με διαφορετική σειρά απ' ότι εισήρθαν λόγο του τρόπου με τον όποιο λειτουργεί ο αλγόριθμος round robin για τις FIFOs (όπως αναφέρεται παραπάνω). Οι καθυστερήσεις δρομολόγησης κυμαίνονται από τρεις έως είκοσι κύκλους ρολογιού και δεν αλλάζουν στο χρονικό άξονα όπως και στο αντίστοιχο σενάριο της πρώτης περίπτωσης.

6.2.2.4 Σενάριο 4ο: Από πολλές θύρες εισόδου σε πολλές θύρες εξόδου

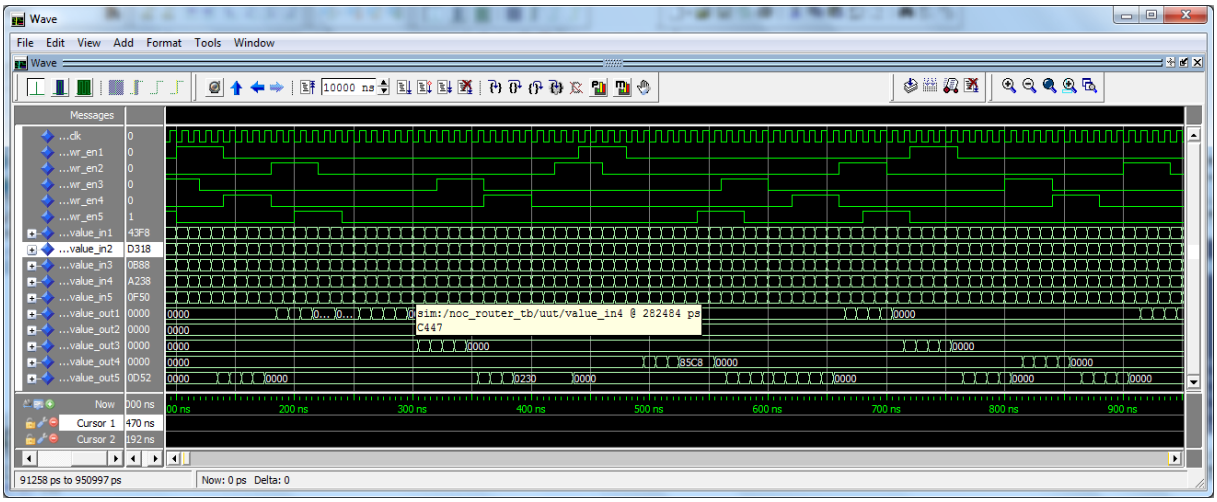
Το σενάριο αυτό όπως αναφέραμε παραπάνω είναι το σημαντικότερο από τα υπόλοιπα τρία και σε αυτή τη περίπτωση που η δημιουργία των πακέτων είναι ιδανική (ένα πακέτο ανά είκοσι κύκλους) θα δούμε δυο προσομοιώσεις λειτουργιάς. Στην πρώτη θα δούμε πως λειτουργεί με ένα σταθερό μοτίβο δημιουργίας πακέτων αλλά επειδή σε πραγματικές συνθήκες λειτουργιάς η άφιξη των πακέτων είναι πολύ τυχαία και οι θύρες για τις όποιες προορίζονται δεν μπορούμε να γνωρίζουμε ποιες μπορεί να είναι, θα τρέξουμε και μια δεύτερη ποιο τυχαία ώστε να δούμε τα αποτελέσματα.

Δημιουργώντας την πρώτη προσομοίωση με ένα σταθερό μοτίβο δημιουργίας πακέτων βλέπουμε ότι οι καθυστερήσεις και η σειρά με την οποία φτάνουν τα πακέτα δεν αλλάζει σε σχέση με το προηγούμενο σενάριο αλλά ούτε με το αντίστοιχο της προηγούμενης περίπτωσης. Τα πακέτα δεν φτάνουν στις θύρες εξόδου του NoC με την ίδια σειρά με την οποία έφτασαν στις θύρες εισόδου λόγο του αλγορίθμου round robin που εκτελείτε (όπως αναφέρεται παραπάνω). Για τον λόγο αυτό οι καθυστερήσεις δρομολόγησης των πακέτων είναι διαφορετικές για κάθε θύρα και κυμαίνονται από τους τρεις μέχρι τους είκοσι κύκλους ρολογιού. Οι καθυστερήσεις αυτές δεν αλλάζουν στο χρονικό άξονα αλλά η συχνότητα δημιουργίας πακέτων είναι ιδανική και τα buffers έχουν αδειάσει μέχρι να φτάσουν καινούρια πακέτα.



Εικόνα 6.8: Προσομοίωση τέταρτου σεναρίου δεύτερης περίπτωσης

Στην ιδανική αυτή περίπτωση και όπως φαίνεται παραπάνω μπορεί οι καθυστερήσεις δρομολόγησης να είναι σχετικά σταθερές αλλά κίνηση υπάρχει καθ' όλο το χρονικό άξονα. Στη συνέχεια θα δούμε και μια δεύτερη προσομοίωση με ποιο τυχαία δημιουργία πακέτων που θα είναι ποιο κοντά σε ένα πραγματικό περιβάλλον επικοινωνίας. Να υπενθυμίσουμε σε αυτό το σημείο ότι σε πραγματικές συνθήκες λειτουργιάς δεν υπάρχουν απώλειες πακέτων αλλά μόνο κάποιες επιπλέον καθυστερήσεις δρομολόγησης σε περίπτωση μεγάλου φόρτου κυκλοφορίας.



Εικόνα 6.9: Προσομοίωση λειτουργίας με τυχαίο μοτίβο δημιουργίας πακέτων υπό ιδανικές συνθήκες (περίπου ένα πακέτο ανά είκοσι κύκλους ρολογιού)

Η παραπάνω προσομοίωση αγγίζει πραγματικές καταστάσεις λειτουργίας κάτω από ιδανικές συνθήκες δημιουργίας πακέτων έτσι ώστε να μπορούμε να γνωρίζουμε πως συμπεριφέρεται. Σε αυτή τη προσομοίωση οι καθυστερήσεις είναι μικρότερες διότι οι χρονικές στιγμές στις οποίες φτάνουν πακέτα είναι στις περισσότερες περιπτώσεις διαφορετικές με αποτέλεσμα να δρομολογούνται με τη σειρά όπου έφτασαν. Οι καθυστερήσεις που δημιουργούνται είναι από τρεις έως δεκατέσσερις κύκλους ρολογιού ανάλογα το φόρτο. Ο χρόνος αυτός είναι αρκετά ικανοποιητικός όσο αφορά την απόδοση του NoC.

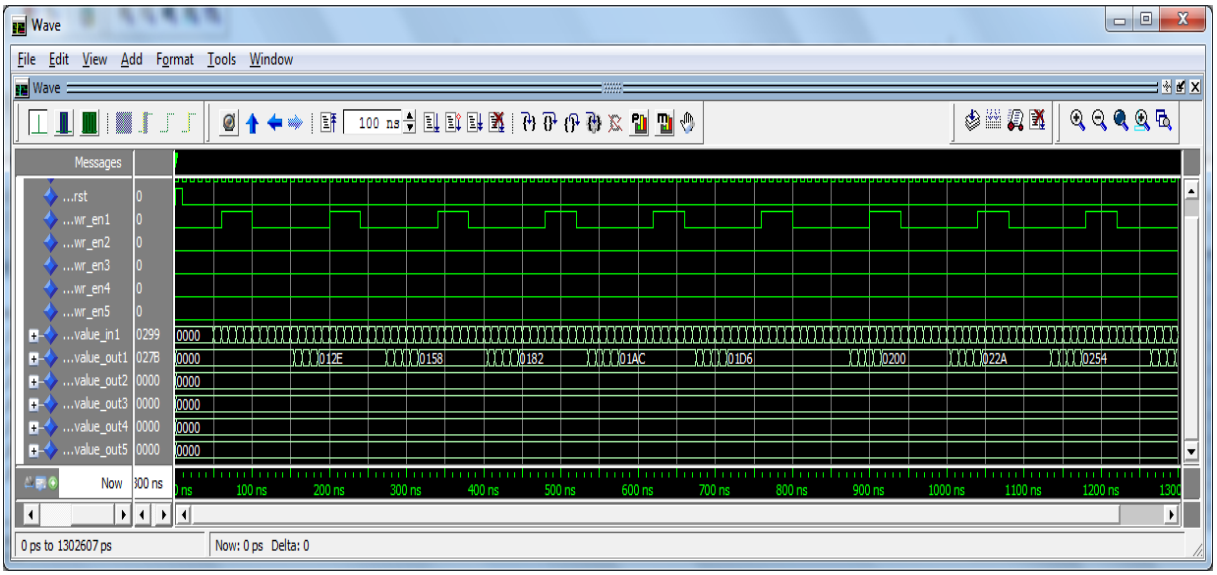
Αξίζει να συμπληρώσουμε ότι η παραπάνω προσομοίωση έτρεξε για 10^{48} κύκλους ρολογιού ώστε να διαπιστωθεί ότι λειτουργεί σωστά. Σε όλο αυτό το χρονικό διάστημα δεν διαπιστώθηκε δυσλειτουργία, ούτε απώλειες πακέτων. Σε κάποιες περιπτώσεις μόνο σημειωθήκαν κάποιες επιπλέον καθυστερήσεις αλλά τα buffers δεν γέμισαν ποτέ και όλα τα πακέτα δρομολογήθηκαν στη θύρα εξόδου για την οποία δημιουργήθηκαν. Δυστυχώς, στο έγγραφο αυτό δεν μπορούμε να δείξουμε όλη τη προσομοίωση διότι ο άξονας του χρόνου είναι παρά πολύ μεγάλος και δεν μπορούμε να ξεχωρίσουμε τα πακέτα που δρομολογούνται.

6.2.3 Περίπτωση 3η: Δημιουργία ενός πακέτου ανά δέκα κύκλους ρολογιού

Με τις παραπάνω περιπτώσεις έχουμε καλύψει όλα τα πιθανά σενάρια λειτουργίας του NoC router αλλά για μπορούμε να γνωρίζουμε τα πάντα γύρω από τον τρόπο επικοινωνίας του θα εξετάσουμε άλλη μια περίπτωση όπου θα υπάρχει πολύ μεγάλος φόρτος και θα δημιουργούνται πακέτα ανά δέκα κύκλους ρολογιού. Αυτό θα γίνει για να είμαστε σε θέση να γνωρίζουμε πότε θα αρχίσουν να υπάρχουν απώλειες πακέτων και πότε οι καθυστερήσεις δρομολόγησης θα γίνουν πάρα πολύ μεγάλες.

6.2.3.1 Σενάριο 1ο: Από μια θύρα εισόδου σε μια θύρα εξόδου

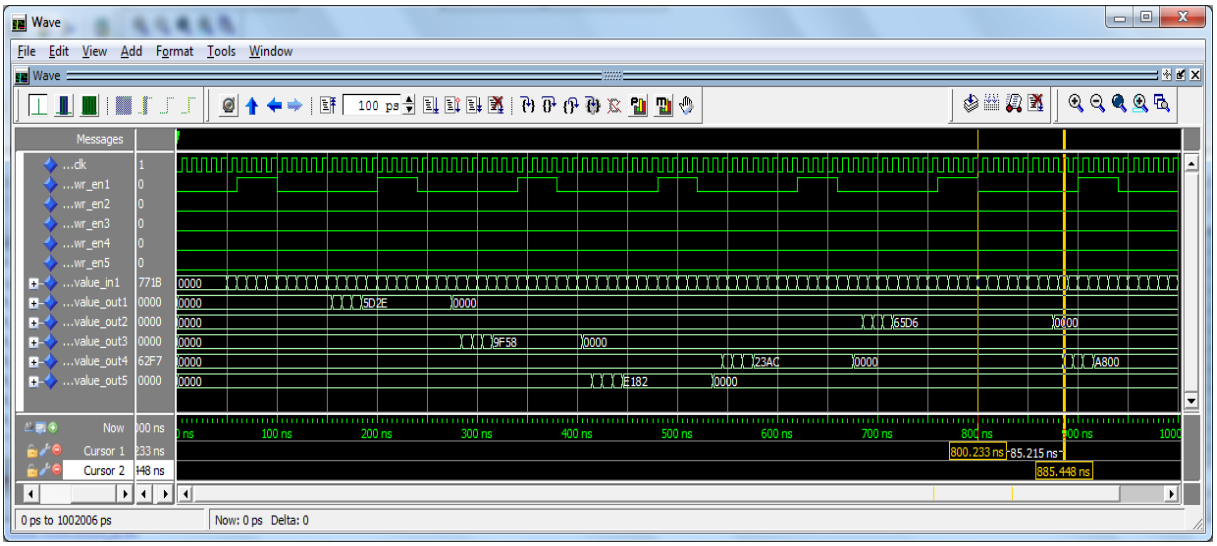
Το σενάριο αυτό έχει κάποιες διαφορές από τις δυο παραπάνω περιπτώσεις των αντίστοιχων σεναρίων. Λόγο του ότι φτάνουν περισσότερα πακέτα, κάποιες φορές το buffer δεν έχει αδειάσει μέχρι την άφιξη του νέου πακέτου και έτσι προκύπτουν κάποιες επιπλέον καθυστερήσεις μερικών κύκλων ρολογιού. Ενώ στις παραπάνω δυο περιπτώσεις οι καθυστερήσεις κυμαίνονταν μεταξύ τριών και έξι κύκλων ρολογιού, τώρα αυξάνονται λίγο οι καθυστερήσεις και θα κυμαίνονται μεταξύ τριών και εννέα κύκλων ρολογιού λόγο του μεγάλου φόρτου κυκλοφορίας πακέτων που αναφέραμε παραπάνω. Η προσομοίωση φαίνεται παρακάτω.



Εικόνα 6.10: Προσομοίωση πρώτου σεναρίου τρίτης περίπτωσης

6.2.3.2 Σενάριο 2ο: Από μια θύρα εισόδου σε πολλές θύρες εξόδου

Όπως και παραπάνω, λόγο του ότι φτάνουν περισσότερα πακέτα, κάποιες φορές το buffer δεν έχει αδειάσει μέχρι την άφιξη του νέου πακέτου και έτσι προκύπτουν κάποιες επιπλέον καθυστερήσεις μερικών κύκλων ρολογιού. Το σενάριο αυτό δεν διαφέρει από το προηγούμενο, απλούστατα επειδή δεν αλλάζει η συχνότητα με την οποία φτάνουν τα πακέτα στην συγκεκριμένη θύρα εισόδου και η καθυστέρηση για να φτάσει το πακέτο στην θύρα εξόδου θα είναι η ίδια (τρεις με εννέα κύκλους ρολογιού), οποιαδήποτε θύρα εξόδου και αν αφορούν τα πακέτα.

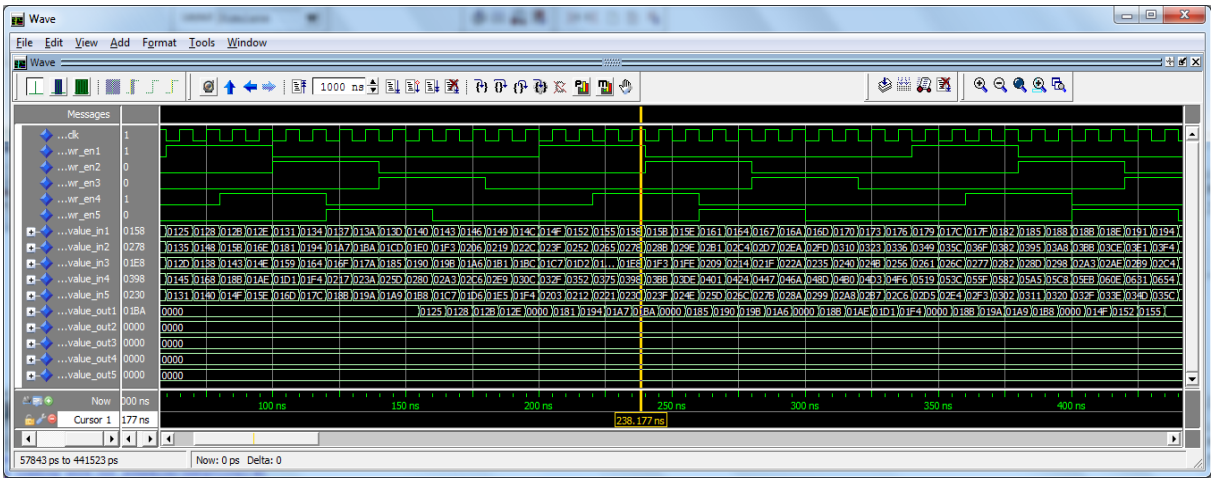


Εικόνα 6.11: Προσομοίωση δεύτερου σεναρίου τρίτης περίπτωσης

6.2.3.3 Σενάριο 3ο: Από πολλές θύρες εισόδου σε μια θύρα εξόδου

Στα αντίστοιχα σενάρια των δυο παραπάνω περιπτώσεων οι καθυστερήσεις ήταν σταθερές στον άξονα του χρόνου και σε καμιά περίπτωση δεν ξεπερνούσαν τους είκοσι κύκλους ρολογιού (κοιμούνταν μεταξύ τριών και είκοσι). Σε αυτό και το παρακάτω σενάριο όπου ο φόρτος κυκλοφορίας πακέτων είναι πολύ μεγάλος θα παρατηρήσουμε ότι οι καθυστερήσεις αυξάνονται στον άξονα του χρόνου κάθε φορά εισέρχονται νέα πακέτα. Τα buffers δεν αδειάζουν καθόλου με αποτέλεσμα όλο και

περισσότερα πακέτα να αποθηκεύονται και ο χρόνος αποδέσμευσης τους από τις μνήμες να αυξάνεται όσο φτάνουν νέα πακέτα.



Εικόνα 6.12: Προσομοίωση τρίτου σεναρίου τρίτης περίπτωσης

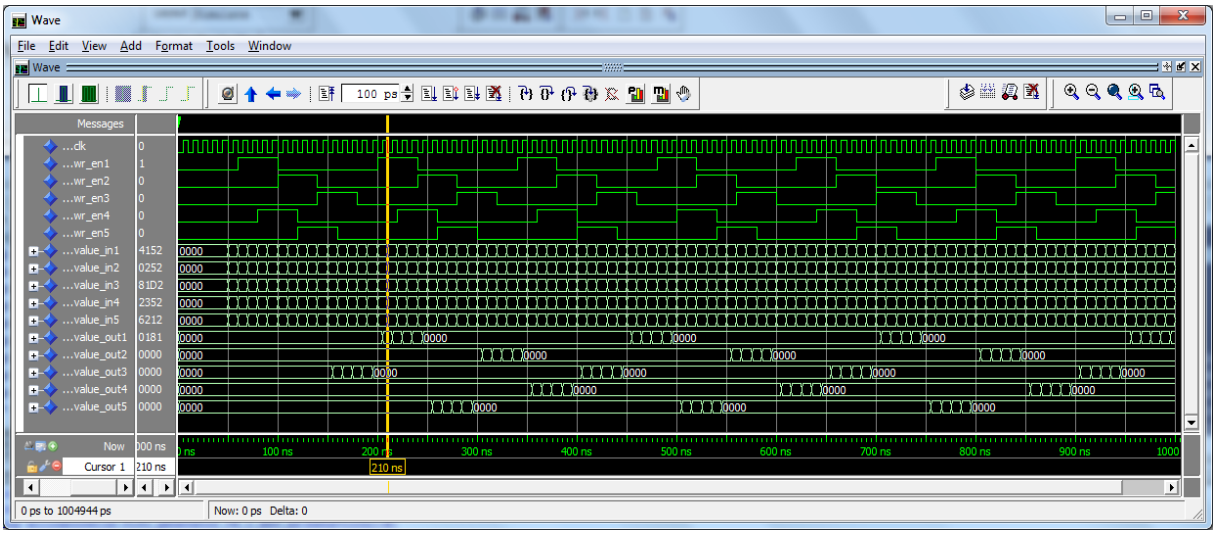
Μετά από την προσομοίωση του σεναρίου αυτού μπορούμε να παρατηρήσουμε ότι κατά την δρομολόγηση των πρώτων πακέτων που έφτασαν στις εισόδους οι καθυστερήσεις είναι οι ίδιες όπως και στα αντίστοιχα σενάρια των δυο παραπάνω περιπτώσεων. Παρ' όλα αυτά όμως λόγο του μεγάλου φόρτου οι καθυστερήσεις αυτές αυξάνονται ανά δέκα κύκλους ρολογιού κάθε φορά που φτάνουν νέα πακέτα καθώς ο ρυθμός άφιξης πακέτων διατηρείτε σταθερός (ένα πακέτο ανά δέκα κύκλους). Μπορούμε, εύκολα, να φανταστούμε ότι αν δεν αλλάξει ο ρυθμός άφιξης πακέτων οι καθυστερήσεις θα γίνουν πάρα πολύ μεγάλες μέχρι το σημείο όπου τα buffers θα γεμίσουν και από το σημείο αυτό και μετά θα έχουμε απώλειες πακέτων λόγο έλλειψης χώρου αποθήκευσης.

Δυστυχώς δεν μπορούμε να δείξουμε ολόκληρη την προσομοίωση διότι δεν θα ήταν δυνατό να φαίνονται τα πακέτα. Τέλος, όπως και στα αντίστοιχα σενάρια των δυο παραπάνω περιπτώσεων τα πακέτα ίσως να μην δρομολογούνται στις εξόδους με την ίδια σειρά με την οποία έφτασαν στις εισόδους. Αυτό γίνεται λόγο του τρόπου λειτουργίας του round robin αλγορίθμου που εκτελείτε (όπως αναφέρεται παραπάνω).

6.2.3.4 Σενάριο 4ο: Από πολλές θύρες εισόδου σε πολλές θύρες εξόδου

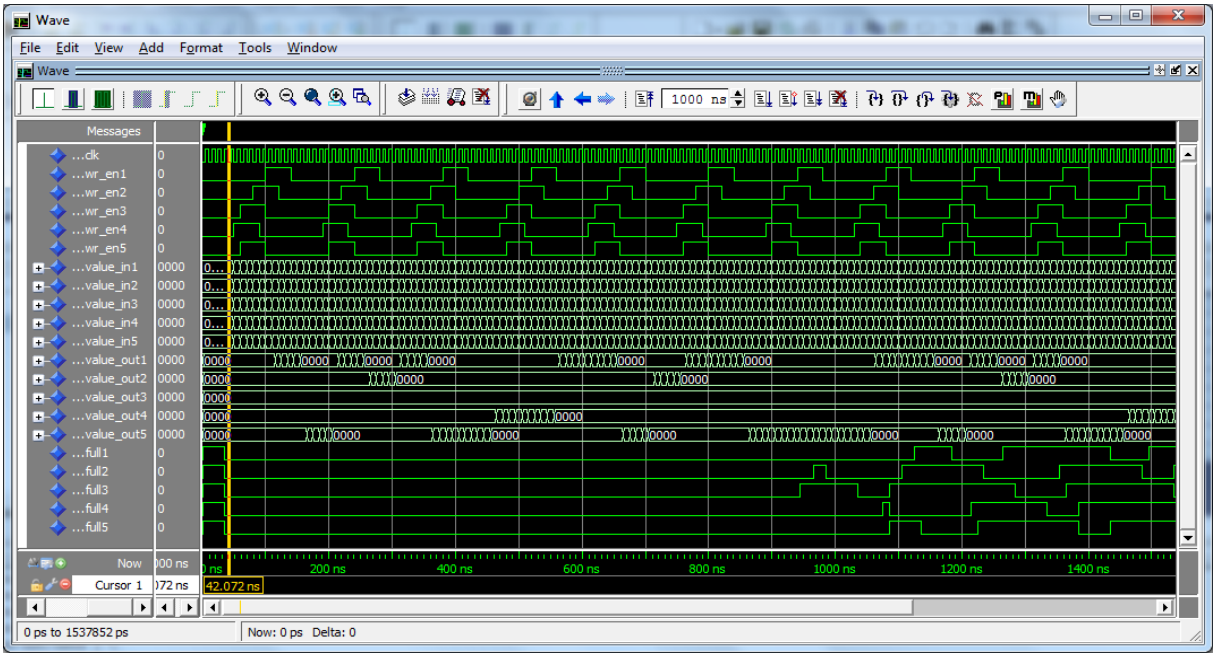
Φτάνοντας στο τέταρτο και σημαντικότερο σενάριο θα δούμε πως συμπεριφέρεται το NoC router στον πολύ μεγάλο φόρτο κυκλοφορίας για μεγάλο χρονικό διάστημα. Όταν τα buffers έχουν αρκετό χώρο για αποθήκευση πακέτων όπως στην περίπτωση μας (256 πακέτα για κάθε FIFO) ακόμα και αν συμβεί να φτάνουν τόσο πολλά πακέτα για αρκετό χρονικό διάστημα το μόνο πρόβλημα που θα έχουμε θα είναι ότι θα αυξηθούν οι καθυστερήσεις δρομολόγησης, χωρίς όμως να υπάρξουν απώλειες πακέτων. Για το λόγο αυτό θα κάνουμε και μια προσομοίωση χρησιμοποιώντας μικρότερες FIFOs των τεσσάρων θέσεων για να δούμε ότι πολύ σύντομα τα πακέτα θα αρχίσουν να χάνονται.

Σε αυτό και το προηγούμενο σενάριο όπου ο φόρτος κυκλοφορίας πακέτων είναι πολύ μεγάλος παρατηρούμε ότι οι καθυστερήσεις αυξάνονται ανά δέκα κύκλους ρολογιού στον άξονα του χρόνου κάθε φορά που φτάνουν νέα πακέτα ενώ ο ρυθμός άφιξης πακέτων να διατηρείτε σταθερός (ένα πακέτο ανά δέκα κύκλους). Τα buffers δεν αδειάζουν καθόλου με αποτέλεσμα όλο και περισσότερα πακέτα να αποθηκεύονται και ο χρόνος αποδέσμευσης τους από τις μνήμες να γίνεται πάρα πολύ μεγάλος.



Εικόνα 6.13: Προσομοίωση τέταρτου σεναρίου τρίτης περίπτωσης

Όπως και σε κάθε άλλη περίπτωση το σενάριο αυτό δεν έχει διαφορές με το προηγούμενο ούτε στις καθυστερήσεις των πακέτων αλλά ούτε στη σειρά με την οποία δρομολογούνται τα πακέτα όταν η συχνότητα άφιξης των πακέτων είναι η ίδια, σε όποια έξοδο και αν προωθούνται τα πακέτα. Τέλος, αν η συχνότητα άφιξης των πακέτων είναι πολύ μεγάλη για μικρό χρονικό διάστημα δεν ρισκάρουμε να έχουμε απώλειες πακέτων, ωστόσο, στην επομένη προσομοίωση θα χρησιμοποιήσουμε μικρότερες FIFO για να δούμε πως θα λειτουργούσε σε διαφορετική περίπτωση που θα θέλαμε το NoC να καταλαμβάνει μικρότερο χώρο στο FPGA.

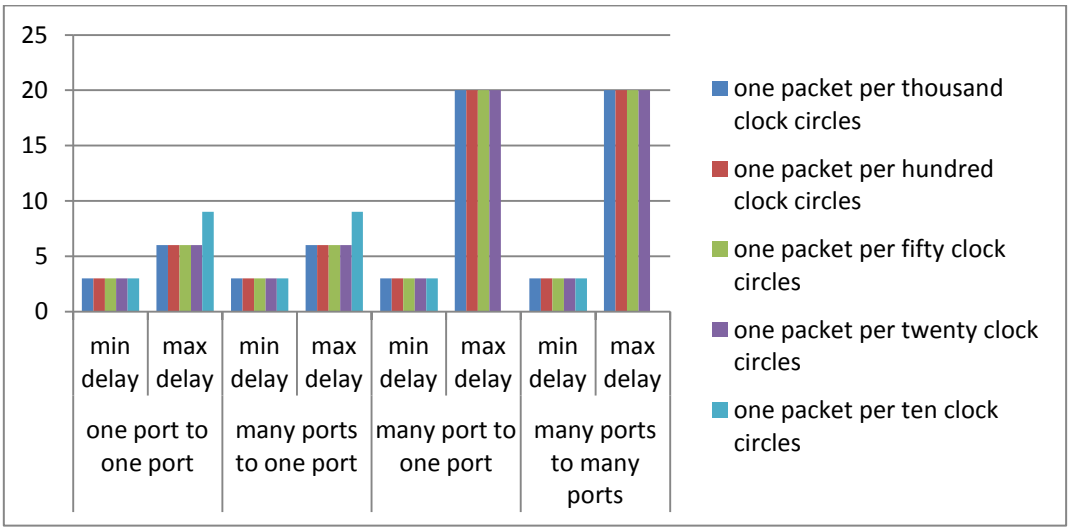


Εικόνα 6.14: Προσομοίωση τέταρτου σεναρίου τρίτης περίπτωσης με FIFO τεσσάρων θέσεων

Στην περίπτωση αυτή όπου χρησιμοποιήσουμε FIFOs τεσσάρων θέσεων ο χώρος τον όποιο καταλαμβάνει το NoC πάνω στο FPGA είναι περίπου ο μισός αλλά σε ένα σενάριο όπου έχουμε μεγάλο φόρτο κυκλοφορίας και όπως φαίνεται και από την προσομοίωση παραπάνω με τα λίγα πακέτα που έφτασαν, τα buffers γέμισαν μέσα σε πολύ μικρό χρονικό διάστημα. Αυτό έχει ως αποτέλεσμα όποιο πακέτο και να φτάσει μετά από αυτό το σημείο θα διαγράφει, κάτι που δεν είναι αποδεκτό. Πρέπει να υπενθυμίσουμε ότι η πτυχιακή αυτή αναπτύχτηκε με σκοπό να μην υπάρχουν απώλειες πακέτων υπό καμιά περίπτωση κατά την δρομολόγηση των πακέτων σε ένα NoC router.

6.3 Αξιολόγηση NoC Router

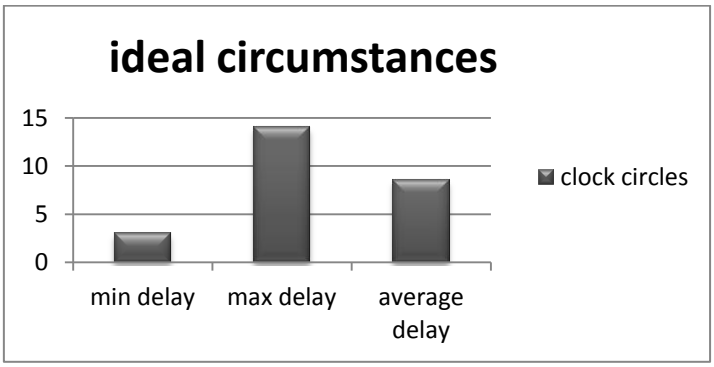
Τώρα που έχουμε μια πλήρη αντίληψη για τον τρόπο που λειτουργεί το NoC router μπορούμε να δούμε τα αποτελέσματα αυτά ως σύνολο και να αξιολογήσουμε την απόδοση του, καθώς επίσης και αν καλύπτει τους στόχους που προσδιορίστηκαν σε προηγούμενα κεφάλαια. Στη συνέχεια θα δούμε ένα γράφημα με το σύνολο των καθυστερήσεων δρομολόγησης των πακέτων που προκαλούνται μερικές περιπτώσεις που εξετάσαμε παραπάνω. Βλέποντας συγκεντρωτικά τα αποτελέσματα μπορούμε εύκολα να αξιολογήσουμε αν η λειτουργία του συστήματος καλύπτει τις προδιαγραφές που έχουν τεθεί σε προηγούμενα σταδία.



Εικόνα 6.15: Γράφημα καθυστέρησης δρομολόγησης πακέτων σε διάφορες περιπτώσεις

Στο παραπάνω γράφημα μπορούμε να δούμε τα συγκεντρωτικά αποτελέσματα των καθυστερήσεων που υφίστανται κατά την δρομολόγηση των πακέτων από τις εισόδους στις εξόδους σε όλες τις περιπτώσεις που εξετάσαμε παραπάνω. Παρατηρούμε ότι κατά την δρομολόγηση από πολλές θύρες προς μια και από πολλές προς πολλές κατά την άφιξη των πακέτων ανά δέκα κύκλους ρολογιού δεν υπάρχουν αποτελέσματα στην μεγίστη καθυστέρηση επειδή κάθε φορά που φτάνουν πακέτα αυξάνονται κατά δέκα κύκλους οι καθυστερήσεις. Για το λόγο αυτό δεν μπορούμε να δώσουμε μια σταθερή μέγιστη καθυστέρηση.

Βλέποντας τα αποτελέσματα μπορούμε να πούμε ότι είμαστε ικανοποιημένοι από την απόδοση του NoC router σύμφωνα με τις καθυστερήσεις που συμβαίνουν κατά τις δρομολογήσεις των πακέτων αν σκεφτούμε το γεγονός ότι δεν συμβαίνουν συγκρούσεις και δεν υπάρχουν απώλειες πακέτων. Ακόμα και στις τρεις περιπτώσεις φόρτου κυκλοφορίας που εξετάσαμε (μικρό, μέσο και μεγάλο φόρτο) τα αποτελέσματα είναι πάρα πολύ ικανοποιητικά, αφού οι στόχοι για την ανάπτυξη αυτής της πτυχιακής έχουν καλυφτεί (δρομολόγηση πακέτων σε NoC router χωρίς απώλειες πακέτων με μικρές καθυστερήσεις).



Εικόνα 6.16: Γράφημα απόδοσης υπό ιδανικές συνθήκες δρομολόγησης πακέτων

6.3 Μελλοντική Ανάπτυξη

Στην παρούσα πτυχιακή εργασία κατασκευάστηκε ένας πλήρως λειτουργικός δρομολογητής (router) για επικοινωνία μεταξύ IP Cores σε ένα Δίκτυο-σε-τσιπ (Network-on-Chip). Ως μελλοντική εργασία για το NoC router που κατασκευάστηκε θα μπορούσε να είναι η περεταίρω επέκταση του. Ορισμένες από τις εργασίες που θα μπορούσαν να γίνουν είναι οι παρακάτω.

- Θα μπορούσαμε να χρησιμοποιήσουμε ένα μεγάλο αριθμό από NoC Router που κατασκευάστηκαν για ένα μεγάλο σύστημα επικοινωνίας σε SoC και να τα ενσωματώσουμε σε ένα μεγάλο FPGA ώστε να μελετήσουμε τους παραμέτρους και την απόδοση του.
- Επίσης, θα μπορούσαμε να ενισχύσουμε το NoC router με πρόσθετα χαρακτηριστικά (features) όπως να χρησιμοποιήσουμε περισσότερες θύρες εισόδου/εξόδου, περισσότερες συνδέσεις (links) ανάμεσα σε θύρες εισόδου και εξόδου αναπτύσσοντας διαφορετικές τοπολογίες συνδέσεων, περισσότερο buffering ανά κάθε θύρα εισόδου για μεγαλύτερη ασφάλεια ώστε να αποφεύγονται οι απώλειες πακέτων σε περιπτώσεις μεγάλου φόρτου.
- Για μεγαλύτερη ασφάλεια στην επικοινωνία θα μπορούσαμε να χρησιμοποιήσουμε τεχνικές κρυπτογράφησης και αποκρυπτογράφησης σε κάθε σύνδεση ανάμεσα στις θύρες εισόδου και εξόδου με σκοπό την απόκρυψη του περιεχομένου των μηνυμάτων.
- Τέλος, θα μπορούσαμε να χρησιμοποιήσουμε πολλά εικονικά κυκλώματα (virtual circuits – VC) για τις συνδέσεις ανάμεσα στις θύρες. Ένα VC παρέχει υπηρεσίες επικοινωνίας σε ένα δίκτυο μεταγωγής πακέτων όπου δυο η περισσότερες συνδέσεις εκμεταλλεύονται, εκ περιτροπής, κοινόχρηστα buffers και εύρος ζώνης χρησιμοποιώντας ειδικές χρονοθυρίδες καταναλώνοντας έτσι λιγότερους πόρους.

Βιβλιογραφία

1. SoC Wikipedia:
   http://en.wikipedia.org/wiki/System_on_a_chip
2. NoC Wikipedia:
   http://en.wikipedia.org/wiki/Network_On_Chip
3. Structure and Design Methodologies:
   http://www.hindawi.com/journals/jece/2012/509465/
4. Xilinx ISE 12.1 tutorial:
   http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_2/ise_tutorial_ug695.pdf
5. Modelsim 6.5 tutorial:
   http://cseweb.ucsd.edu/classes/sp10/cse141L/lab1/modelsim_tut.pdf
6. Spartan 3 Starter Kit Board User Guide:
   http://www.xilinx.com/support/documentation/boards_and_kits/ug130.pdf
7. Virtual Circuits in NoC:
   http://www.google.gr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCIQFjAA&url=http%3A%2F%2Fwww2.imm.dtu.dk%2Fpubdb%2Fviews%2Fedoc_download.php%2F4878%2Fpdf%2Fimm4878.pdf&ei=Lg10UM-wHcaF4gShr4GoDg&usg=AFQjCNGVLxFUIA2EFiRQdDl58aZDvFQGEg

Παράρτημα

Παρουσίαση Πτυχιακής

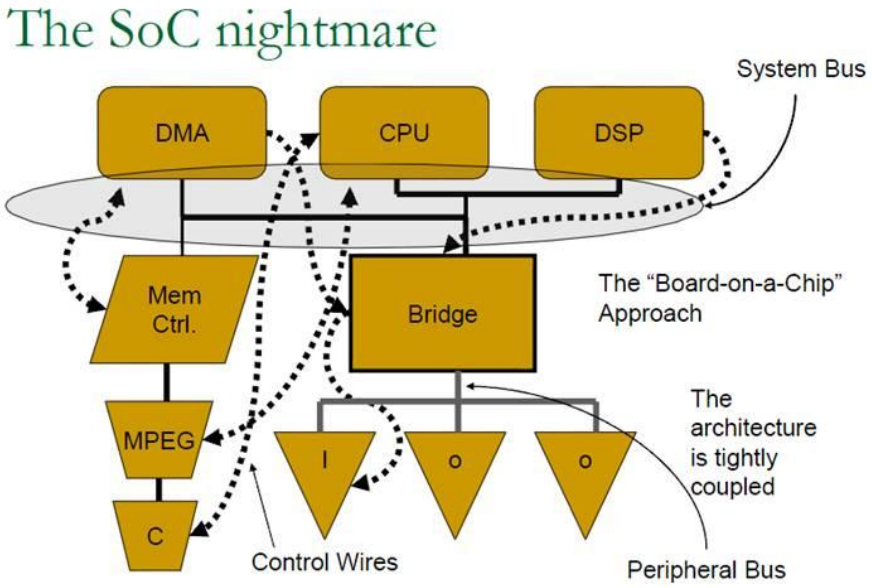
Ανάπτυξη Δομικών Στοιχείων Δρομολογητών για Δίκτυα σε Τσιπ (Network on Chip - NoC)

- Φίλιππος Γεώργιος Κολυμπιανάκης Α.Μ. 2172
- Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων
- Επιβλέπων καθηγητής : Κορνάρος Γεώργιος

Εισαγωγή

- System on Chip – Soc
  - Ενσωμάτωση όλων των εξαρτημάτων ενός ηλεκτρονικού υπολογιστή ή άλλων ηλεκτρονικών συστημάτων σε ένα ενιαίο τσιπ
  - Καλύπτουν ένα μεγάλο φάσμα εφαρμογών
  - Χαμηλή κατανάλωση ισχύος
  - Μικρή έκταση
  - Διασύνδεση πολλαπλών IP Cores
  - Συνδέσεις σημείου προς σημείο χρησιμοποιώντας busses
  - Αρκετούς περιορισμούς για μεγάλης έκτασης SoC

Εισαγωγή



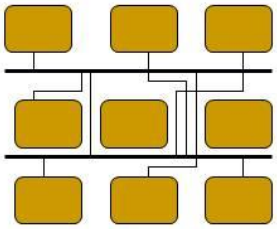
3

Network-on-Chip (NoC) Εισαγωγή

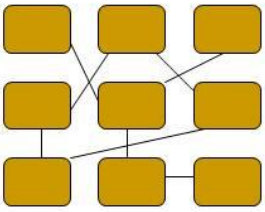
- Network on Chip – NoC
  - Ανάγκες για επικοινωνία σε ένα SoC
  - Με την εμφάνιση συστημάτων με πολυπύρηνους επεξεργαστές, ένα δίκτυο μέσα σε ένα τσιπ είναι μια φυσική αρχιτεκτονική επιλογή
  - Μετακίνηση πακέτων συμφώνα με αλγόριθμους διαιτησίας και δρομολόγησης (routing & arbitration algorithms)
  - Προσδιοριστικοί (deterministic) και Προσαρμοστικοί (adaptive) αλγόριθμοι
  - Κεντρικό arbitration και Κατανεμημένο

4

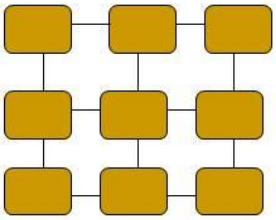
Bus vs NoC



Bus-based architectures



Irregular architectures

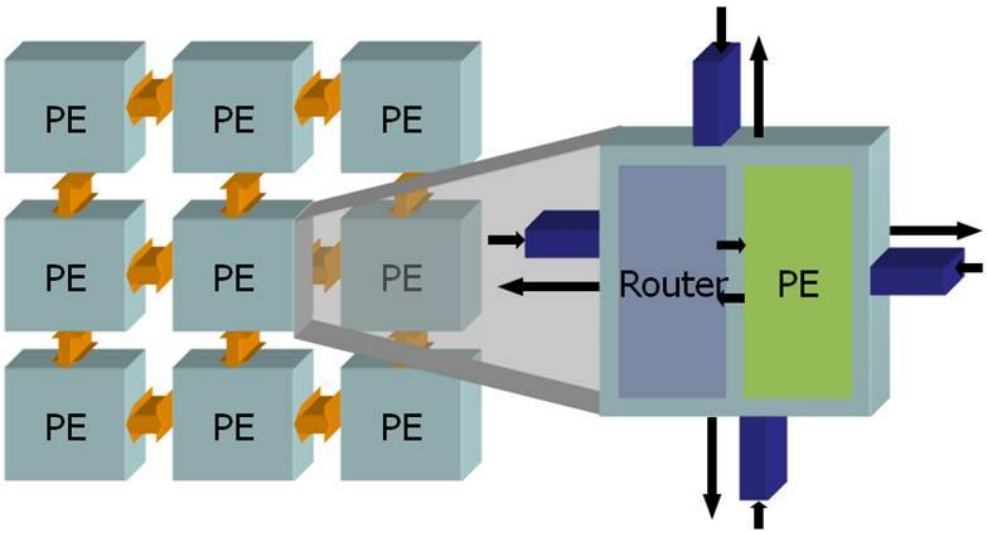


Regular Architectures

- Διασύνδεση με Bus
  - Μικρό κόστος
  - Ευκολία στην υλοποίηση
  - Ευέλικτο
- Networks on Chip
  - Προσέγγιση σε πολλά επίπεδα
  - Αντικατάσταση των busses με αρχιτεκτονικές δικτύων
    - Περισσότερες ιδιότητες σχεδίασης
    - Περισσότερο εύρος ζώνης
    - Εξοικονόμηση ενέργειας

5

Network on Chip



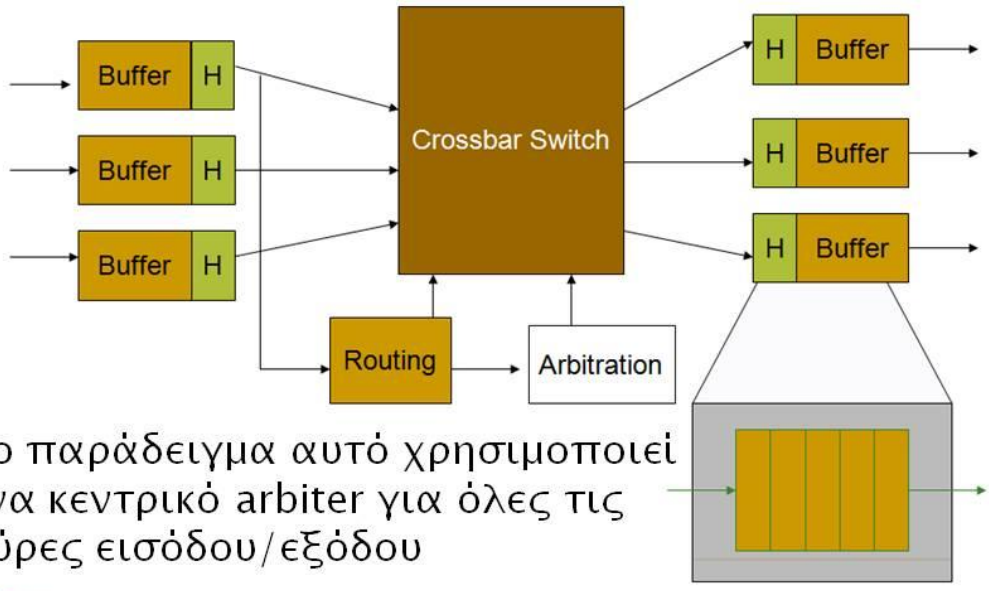
6

Network on Chip

- Πλεονεκτήματα
  - Νέες αρχιτεκτονικές σχεδίασης
  - Αντικατάσταση των καλωδίων με νέες υποδομές δικτύων
  - Σήματα και διάδρομοι αντικαταστάθηκαν με πακέτα
  - Ποιο αποτελεσματική διασύνδεση
  - Αύξηση παραγωγικότητας του συστήματος

7

Τυπικό Network on Chip



- Το παράδειγμα αυτό χρησιμοποιεί ένα κεντρικό arbiter για όλες τις θύρες εισόδου/εξόδου

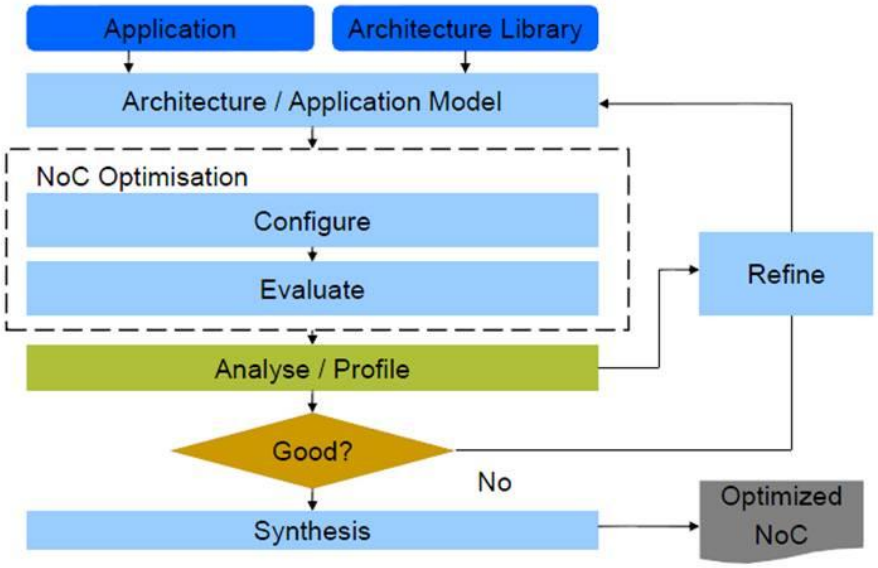
8

Αλγόριθμοι Δρομολόγησης

- Οι αλγόριθμοι δρομολόγησης των NoC πρέπει να είναι απλοί
  - Πολύπλοκα σχήματα καταναλώνουν περισσότερο χώρο στη συσκευή
  - Πρόσθετες καθυστερήσεις
- Αποφυγή αδιεξόδων (deadlocks)
  - Μπορούν να συμβούν όταν γεμίσουν τα buffers
  - Όλες οι διαδρομές είναι απασχολημένες
- Επιπλέον επιθυμητά χαρακτηριστικά
  - QoS, Ανοχή στα λάθη

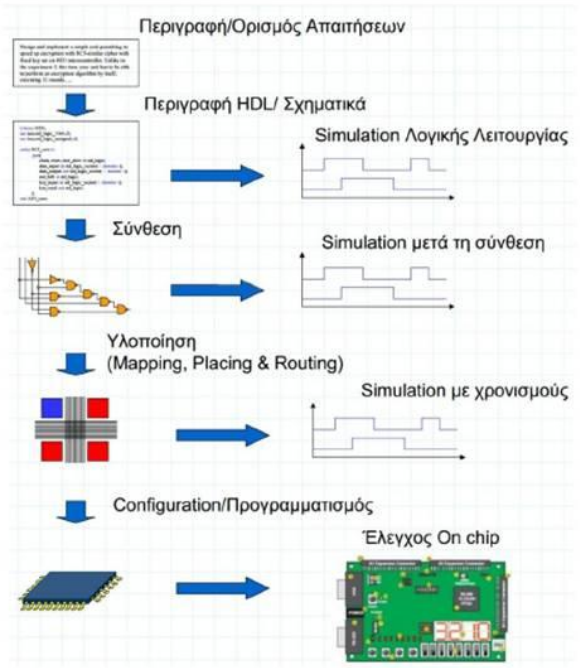
9

Διαδικασία σχεδίασης



10

Στάδια ολοκλήρωσης του Project



11

Στόχοι

- Σχεδίαση NoC Router 5x5 θυρών
- Υψηλού επίπεδου λειτουργικότητα με μικρές καθυστερήσεις
- Χωρίς απώλειες πακέτων
- Ανάπτυξη απαραίτητων μονάδων για σωστή λειτουργία
- Παρουσίαση της λειτουργίας σε FPGA του Spartan 3 Starter Kit board

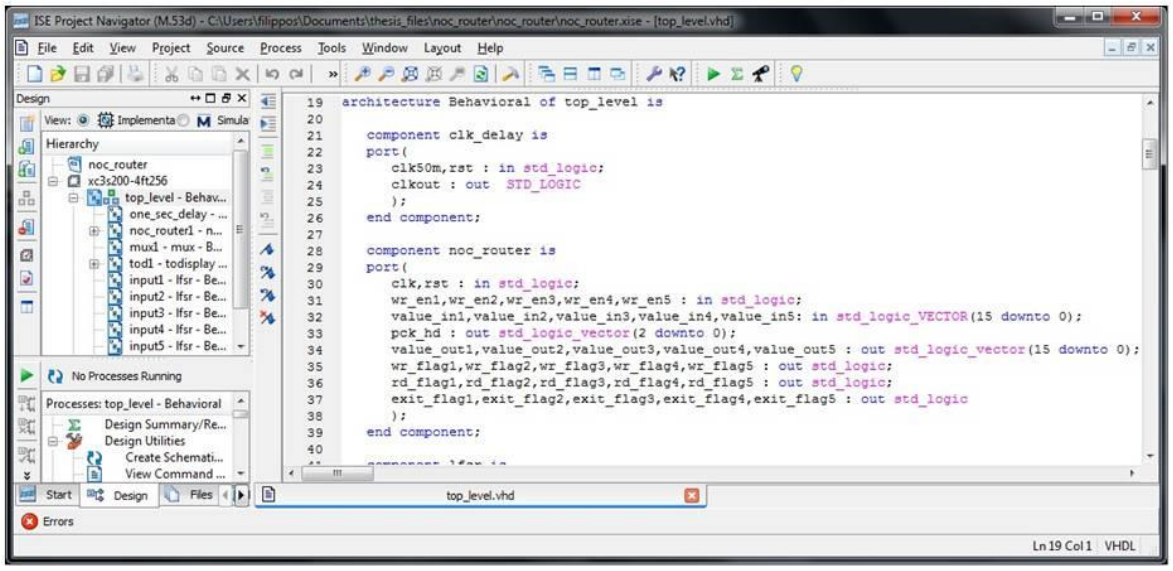
12

Εργαλεία Σχεδίασης και Προσομοίωσης

- Xilinx ISE 12.1
  - Σχεδίαση των source files με VHDL
- Modelsim 6.5
  - Προσομοίωση των source files και του NoC
- Spartan 3 Starter Kit Board
  - Προσομοίωση λειτουργίας του NoC στο FPGA του Spartan 3

13

Xilinx ISE 12.1



14

NoC Router

- Πέντε θύρες εισόδου
- Πέντε θύρες εξόδου
- Πέντε buffers (τύπου FIFO) χωρητικότητας 256 πακέτα (1024 flit)
- Ένας Χρονοδρομολογητής εισόδου (Input Scheduler)
- Ένας Χρονοδρομολογητής εξόδου (Output Scheduler)
- Δυο arbiters για έλεγχο εισόδου και εξόδου

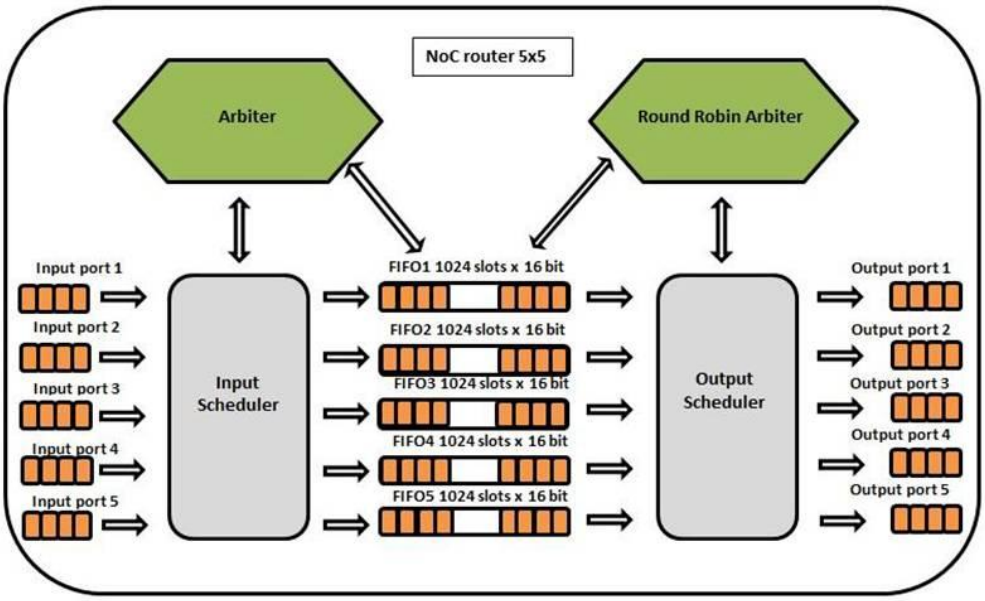
15

NoC Router

- Πακέτα δεδομένων
  - Μέγεθος πακέτου οκτώ Byte (64 bit)
  - Τέσσερα flit δυο Byte (16 bit)
  - Μέγεθος κεφαλίδας 3 bit (στο πρώτο flit κάθε πακέτου
- Δρομολόγηση πακέτων
  - Από όλες τις θύρες εισόδου προς όλες τις θύρες εξόδου
  - Έλεγχος των buffers από arbiter εισόδου
  - Λειτουργία του round robin αλγόριθμου από τον arbiter εξόδου

16

NoC Router 5 x 5 θυρών



17

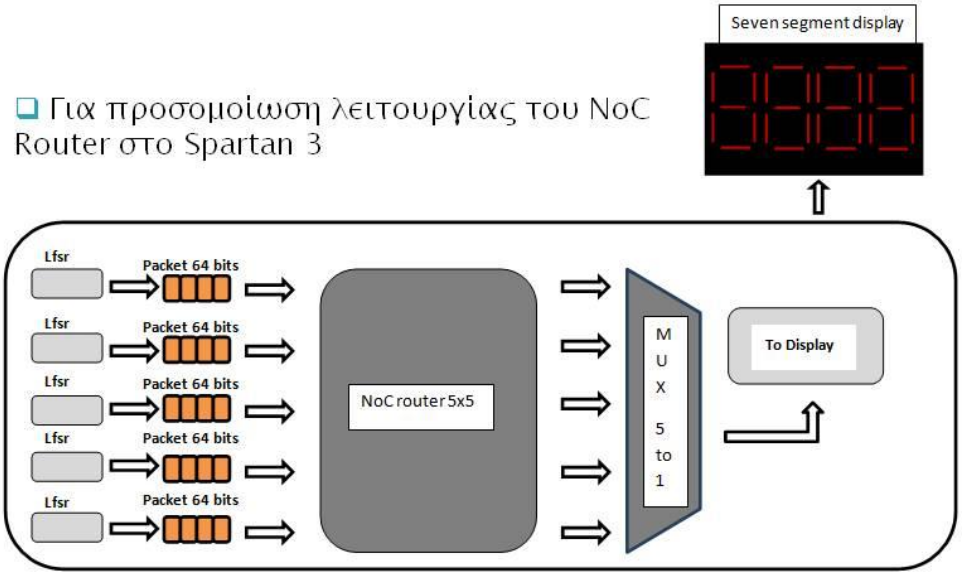
Προσομοίωση λειτουργίας του NoC Router

- linear feedback shift register
  - Δημιουργία τυχαίων πακέτων για προσομοίωση στο Spartan 3
  - Ένας LFSR για κάθε μια θύρα εισόδου
- Απεικόνιση πακέτων στην οθόνη επτά τμημάτων
  - Χρήση πολυπλέκτη πέντε εισόδων για αναπαράσταση εξόδου στην οθόνη επτά τμημάτων
  - Μετατροπή των χαρακτήρων κάθε flit των πακέτων σε δεκαεξαδική μορφή (To Display)
- One second clock divider

18

Top Level

- Για προσομοίωση λειτουργίας του NoC Router στο Spartan 3



19

Source Files

Source files hierarchy	Complexity –lines in VHDL
Top Level	220
One Second Clock	40
NoC Router	343
In Scheduler	167
5 x FIFOs	Core Generator
Out Scheduler	420
5 x Choose Exit	100
Mux	40
To Display	105
4 x bcd to seg	50
5 x LFSR	34
User Constrains	38

20

Χώρος που καταλαμβάνει το NoC στο FPGA

Logic Utilization		
Number of Slice Flip Flops	1,188 out of 3,840	30%
Number of 4 input LUTs	1,221 out of 3,840	31%
Logic Distribution		
Number of occupied Slices -Number of Slices containing only related logic -Number of Slices containing unrelated logic	1,104 out of 1,920 1,104 out of 1,104 0 out of 1,104	57% 100% 0%
Total Number of 4 input LUTs -Number used as logic -Number used as a route-thru	1,333 out of 3,840 1,221 112	34%
Number of bonded IOBs	30 out of 173	17%
Number of BRAMs:	5 out of 12	41%
Number of GCLKs	2 out of 8	25%

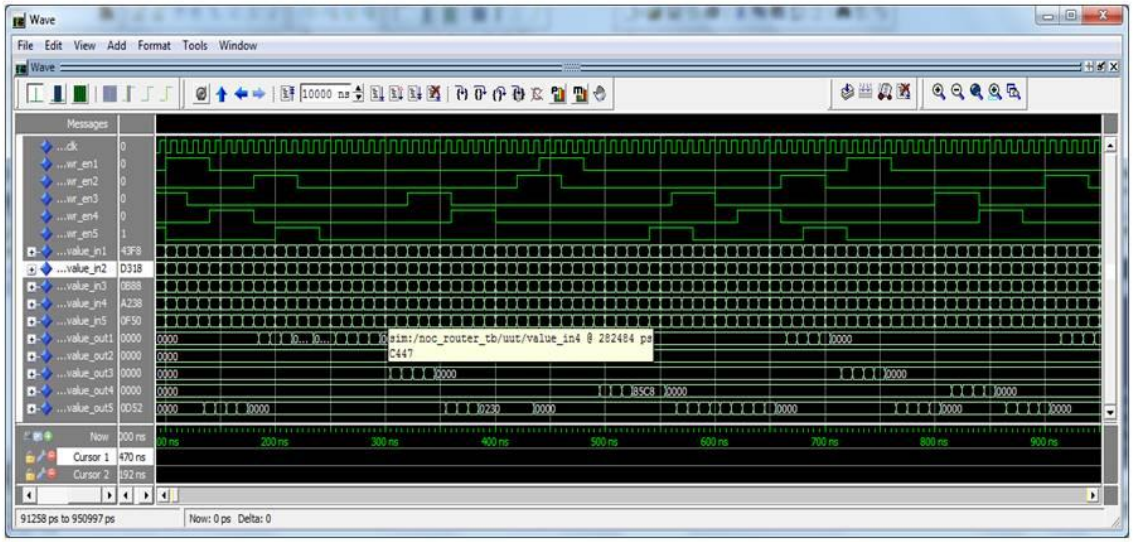
21

Test Benches

- Αρχεία VHDL
- Καθορισμός των τιμών εισόδου για κάποιο Source File με σκοπό την προσομοίωση λειτουργίας στο Modelsim
- Δημιουργία Test Bench για κάθε module
- Δημιουργία Test Bench για ολόκληρο το project
- Εκκίνηση Modelsim για προσομοίωση μέσα από ISE

22

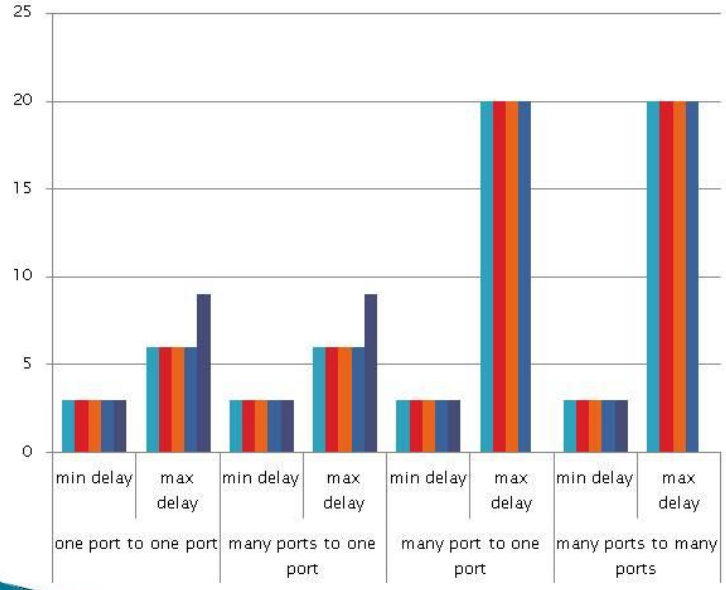
Test Bench



Test Benches

- Εξέταση σε τρεις περιπτώσεις
  - Ένα πακέτο / 50 κύκλους ρολογιού
  - Ένα πακέτο / 20 κύκλους ρολογιού
  - Ένα πακέτο / 10 κύκλους ρολογιού
- Με τέσσερα διαφορετικά σενάρια προσομοίωσης
  - Από μια θύρα σε μια θύρα
  - Από μια θύρα σε πολλές θύρες
  - Από πολλές θύρες σε μια θύρα
  - Από πολλές θύρες σε πολλές θύρες

Αποτελεσματα NoC Router



- Καθυστέρηση δρομολόγησης πακέτων σε κύκλους ρολογιού

- Για περίοδο ρολογιού 10 ns
  Συχνότητα λειτουργίας 96 MHz

25

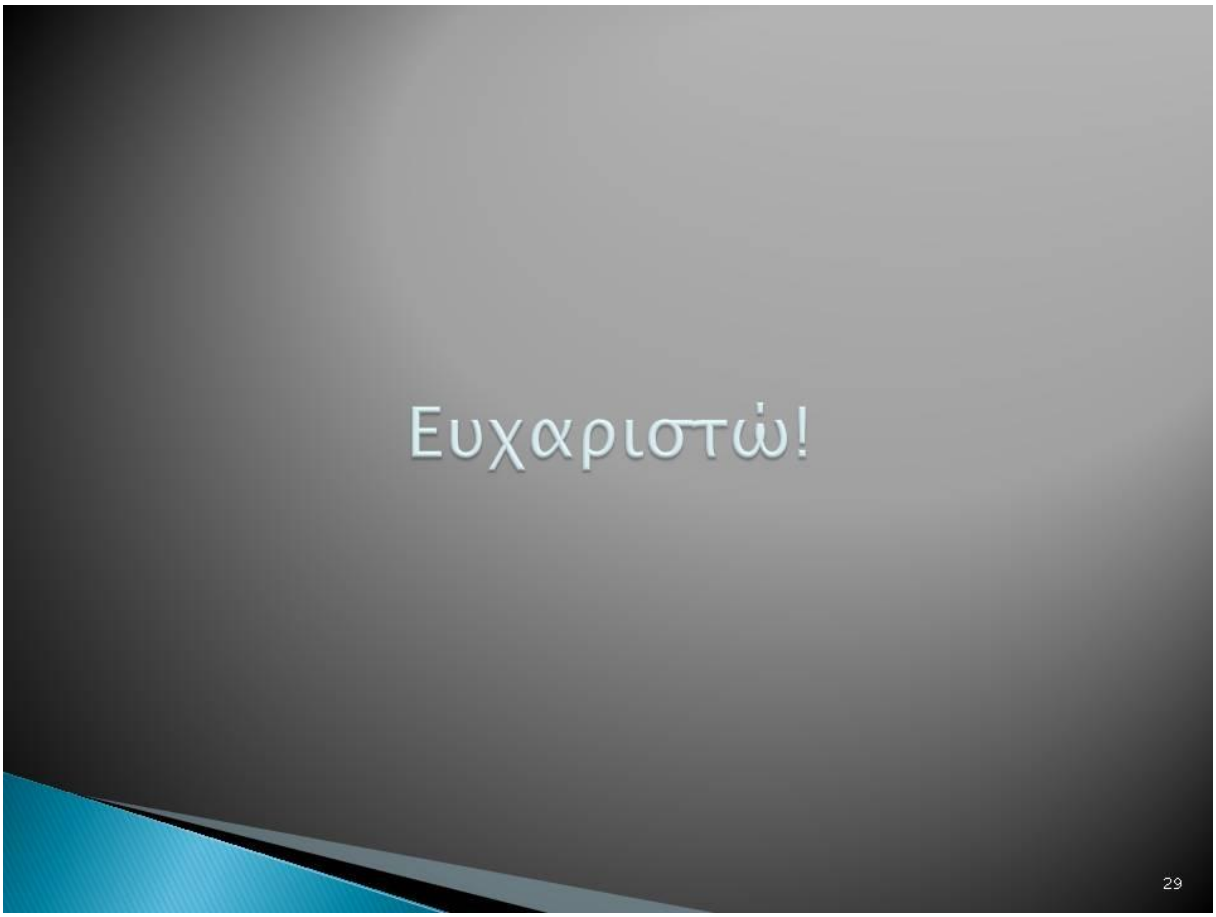
Συμπεράσματα

- Ικανοποιητική απόδοση
- Δεν συμβαίνουν συγκρούσεις πακέτων
- Δεν παρατηρούνται επικαλύψεις πακέτων
- Δεν έχουμε απώλειες πακέτων
- Μικρές σχετικά καθυστερήσεις
- Ικανοποιητική συχνότητα απόκρισης
- Μεγάλος βαθμός δυσκολίας

26

Μελλοντική ανάπτυξη

- Μεγάλο αριθμό από NoC σε ένα μεγάλο SoC
- Πρόσθετα χαρακτηριστικά
  - Περισσότερες θύρες
  - Διαφορετικές τοπολογίες
  - Περισσότερο buffering
- Τεχνικές κρυπτογράφησης και αποκρυπτογράφησης σε κάθε σύνδεση
- Περισσότερα εικονικά κυκλώματα ανάμεσα στις συνδέσεις


Ευχαριστώ!
29