Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Υλοποίηση Δυναμικά Επαναδιαμορφούμενου Ενσωματωμένου Συστήματος σε τεχνολογία FPGA**
Ονοματεπώνυμο Φοιτητή	**Λεβέντης Απόστολος του Παναγιώτη**
Αριθμός Μητρώου	**ΠΜΣΠ/08010**
Κατεύθυνση	**Τεχνολογία Ενσωματωμένων Υπολογιστικών Συστημάτων**
Επιβλέπων	**Ψαράκης Μιχάλης, Λέκτορας**

Ημερομηνία Παράδοσης **Οκτώβριος 2010**

ПЕЧАТНО ПЕРА

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ

Τριμελής Εξεταστική Επιτροπή

.	.	.
Μιχάλης Ψαράκης Λέκτορας	Δημήτριος Γκιζόπουλος Αναπληρωτής Καθηγητής	Χαράλαμπος Κωνσταντόπουλος Λέκτορας

ПЕЧАТНО ПЕРА

Περίληψη

Διανύοντας την τρίτη δεκαετία της ζωής τους, τα FPGA αποτελούν σήμερα μια τεχνολογικά ώριμη πλατφόρμα η οποία όχι μόνο χρησιμοποιείται στην ταχεία ανάπτυξη πρωτοτύπων αλλά επιπλέον είναι δυνατό να αντικαταστήσει τα ASIC ολοκληρωμένα κυκλώματα που χρησιμοποιούνται σε εμπορικά προϊόντα. Σημαντικό ρόλο στην υιοθέτησή των FPGA συνετέλεσε η βελτίωσή τους στους βασικούς παράγοντες όπου η τεχνολογία των ASIC υπερτερούσε, δηλαδή στη χωρητικότητα σε πύλες, στην ταχύτητά τους καθώς και στην κατανάλωση ισχύος. Παρόλο τα FPGA εξακολουθούν να υπολείπονται των ASICs, η χρήση τους έφερε στο προσκήνιο τα πλεονεκτήματά τους, όπως το σχεδόν μηδενικό κόστος αρχικής υλοποίησης (NRE), την ταχύτητα στην παραγωγή λειτουργικού συστήματος καθώς και τη δυνατότητα επαναπρογραμματισμού τους. Η τελευταία είναι πολύ σημαντική καθώς επιτρέπει εκ των υστέρων αλλαγές και τροποποιήσεις σε ένα κύκλωμα, κάτι που στα ASIC δεν είναι εφικτό.

Προχωρώντας ένα βήμα παραπέρα, κάποια FPGA προσφέρουν τη δυνατότητα να γίνεται μεταβολή μέρους της λειτουργικότητάς τους κατά τη διάρκεια της λειτουργία τους έτσι ώστε κάποιο τμήμα του FPGA να αλλάζει λειτουργικότητα χωρίς να επηρεάζεται το υπόλοιπο. Η δυνατότητα αυτή ονομάζεται *Δυναμική Επαναδιαμόρφωση* και δημιουργεί νέο πεδίο εφαρμογών στα FPGA επιτρέποντας για παράδειγμα τη δημιουργία συστημάτων που προσαρμόζονται δυναμικά στις συνθήκες του περιβάλλοντός τους ή διορθώνουν μόνα τους προβλήματα που εμφανίζονται κατά τη λειτουργία τους.

Στην παρούσα εργασία μελετάται η Δυναμική Επαναδιαμόρφωση και παρουσιάζεται η διαδικασία υλοποίησής της σε ένα Ενσωματωμένο Σύστημα. Στόχος είναι να φανούν τα συγκριτικά πλεονεκτήματα που μπορεί να προσφέρει η χρήση δυναμικής επαναδιαμόρφωσης σε σχέση με τη συνηθισμένη στατική υλοποίηση αλλά και να εντοπισθούν τα μειονεκτήματα της συγκεκριμένης διαδικασίας.

Abstract

At their third decade, not only are FPGAs a technologically mature platform being used for rapid system prototyping but they are also a viable replacement option for ASICs in commercial product implementations. A significant factor for the adoption of FPGAs was their improvement in the areas of ASIC technology superiority, i.e. the capacity in logic gates, the speed and power consumption. While still behind ASICs, FPGAs' use uncovered some of their inherent advantages such as their almost zero Non-Recurring Engineering cost, their fast implementation cycle and their capability for reprogramming. This last characteristic is very important as it allows for changes and modifications in a finalized circuit, something impossible with ASIC technology implementation.

Some FPGAs go further, offering the possibility to modify the device functionality during its operation, thus allowing a part of the FPGA device to change its functionality without destructing the rest of it. This functionality is called *Dynamic Reconfiguration* and opens a new application area to FPGAs allowing, for example, the creation of systems that dynamically adapt to the conditions of their operating environment or have self-healing capabilities.

In this thesis, the Dynamic Reconfiguration is studied and the process of implementing it in an FPGA-based embedded system is presented. The objective is to identify the advantages offered by a dynamic system implementation in contrast to the default static one and, furthermore, to expose the drawbacks of the dynamic reconfiguration procedure.

Εισαγωγή

Η εξέλιξη της τεχνολογίας των Επιτόπου Προγραμματιζόμενων Πινάκων Πυλών (Field Programmable Gate Arrays – FPGAs) έχει επιτρέψει την ευρεία χρήση τους ακόμα και σε εφαρμογές όπου μέχρι πρότινος κυριαρχούσαν τα Ολοκληρωμένα Κυκλώματα Ειδικού Σκοπού (Application Specific Integrated Circuits – ASICs). Βασική αιτία για αυτήν την αλλαγή ήταν η αύξηση της χωρητικότητας σε πύλες που έχουν τα σύγχρονα FPGAs σε συνδυασμό με την δυνατότητα που παρέχουν για άμεση υλοποίηση ενός κυκλώματος. Κατ' αυτόν τον τρόπο, σε πάρα πολλές περιπτώσεις, τα όποια οικονομικά οφέλη παρουσιάζει η χρήση ενός ASIC σε κάποιο προϊόν ισοσκελίζονται από την ταχύτερη είσοδό του στην αγορά λόγω του μειωμένου χρόνου κατασκευής που προσφέρει η χρήση FPGA. Επιπλέον, μιας και η λειτουργικότητα ενός FPGA ορίζεται μέσω ενός αρχείου διαμόρφωσης (configuration file), τα FPGAs παρέχουν τη δυνατότητα για επιτόπιες αναβαθμίσεις στη λειτουργικότητα του προϊόντος προκειμένου να παρακαμφθούν σχεδιαστικές παραλείψεις ή να προστεθεί επιπλέον λειτουργικότητα σε ένα υπάρχον σύστημα.

Υπάρχουν διάφορες τεχνολογίες κατασκευής FPGAs (όπως SRAM, Antifuse, Fuse, Flash, EPROM/EEPROM). Η βασική διαφορά των SRAM-based FPGAs σε σχέση με τα υπόλοιπα είδη είναι ότι η διακοπή παροχής ενέργειας στο FPGA συνεπάγεται απώλεια της υλοποιούμενης λειτουργικότητας. Για το λόγο αυτό είναι απαραίτητη η χρήση μιας εξωτερικής μνήμης (configuration memory) όπου αποθηκεύεται το αρχείο διαμόρφωσης προκειμένου να φορτώνεται στο FPGA κατά την εκκίνηση του συστήματος. Αυτό το χαρακτηριστικό σε πολλές εφαρμογές αποτελεί κατασκευαστικό πλεονέκτημα μιας και επιτρέπει για παράδειγμα αναβαθμίσεις στη λειτουργικότητα ενός προϊόντος. Επιπλέον προσφέρει τη δυνατότητα να υπάρχουν για κάποιο σύστημα πολλαπλές λειτουργικότητες αποθηκευμένες στην εξωτερική μνήμη και αναλόγως με τις τρέχουσες απαιτήσεις να γίνεται η κατάλληλη διαμόρφωση του FPGA. Αν μπορεί να επαναδιαμορφωθεί κατά τη διάρκεια της λειτουργίας μόνον ένα τμήμα του FPGA ενώ το υπόλοιπο παραμένει ενεργό και ανεπηρέαστο, τότε έχουμε *Δυναμική Επαναδιαμόρφωση* (ΔΕ). Για να επιτευχθεί η ΔΕ πρέπει κατ' αρχάς να υποστηρίζεται κατασκευαστικά από την αρχιτεκτονική του FPGA αλλά και να υπάρχουν διαθέσιμα εργαλεία λογισμικού που να επιτρέπουν υλοποίηση της συγκεκριμένης διαδικασίας.

Μια τυπική εφαρμογή της ΔΕ επιτρέπει τη χρονική αλληλοεπικάλυψη τμημάτων του FPGA τα οποία δεν είναι ταυτόχρονα ενεργά. Σε μια τέτοια περίπτωση τα τμήματα αυτά μπορούν να «φορτώνονται» και να «ξεφορτώνονται» δυναμικά στο κύκλωμα του FPGA όταν είναι απαραίτητη η ύπαρξή τους κατ' αντιστοιχία με τον τρόπο που ένα λειτουργικό σύστημα «φορτώνει» και «ξεφορτώνει» τμήματα ενός προγράμματος από τη μνήμη ενός υπολογιστικού συστήματος. Έτσι, σε ένα σύστημα εγγραφής και αναπαραγωγής ήχου που χρησιμοποιεί ΔΕ, τα τμήματα που σχετίζονται με την εγγραφή και την αναπαραγωγή του ήχου (π.χ. codecs) δε χρειάζεται να είναι ταυτόχρονα ενεργά αλλά μπορούν να ενεργοποιούνται ανάλογα με το αν γίνεται εγγραφή ή αναπαραγωγή ήχου τη δεδομένη χρονική στιγμή. Ομοίως σε ένα σύστημα μετάδοσης ροής εικόνας μέσω δικτύου (streaming video) ο αλγόριθμος συμπίεσης εικόνας μπορεί να αλλάζει ανάλογα με την ζητούμενη ποιότητα ή το διατιθέμενο εύρος ζώνης του δικτύου. Αντίστοιχα, σε ένα σύστημα SDR (Software Defined Radio) που βασίζεται σε FPGA, μέσω της ΔΕ το FPGA μπορεί να αναδιατάσσεται προκειμένου να υποστηρίξει διαφορετικά πρωτόκολλα επικοινωνίας.

Στην παρούσα εργασία γίνεται μια μελέτη της διαδικασίας Δυναμικής Επαναδιαμόρφωσης στα FPGAs της Xilinx. Παρουσιάζεται μια μεθοδολογία δημιουργίας εναλλασσόμενων δυναμικών τμημάτων, ενώ γίνεται σύγκριση της συγκεκριμένης υλοποίησης με μια στατική σχεδίαση. Τα παραπάνω εφαρμόζονται στην περίπτωση μιας κρυπτογραφικής μηχανής στην οποία οι υποστηριζόμενοι αλγόριθμοι κρυπτογράφησης υλοποιούνται σε υλικό που μπορεί να αναδιαταχθεί. Έτσι μπορεί να αλλάζει κατά τη λειτουργία της τον αλγόριθμο κρυπτογράφησης δεδομένων προσφέροντας δυνατότητα για επιτάχυνση (hardware acceleration) διαφορετικών αλγόριθμων.

Για την πειραματική επαλήθευση των ανωτέρω χρησιμοποιήθηκε μια αναπτυξιακή κάρτα ML403 της Xilinx η οποία περιλαμβάνει το FPGA 4VFX12 της οικογένειας Virtex-4. Δημιουργήθηκε ένα σύστημα βασισμένο στον επεξεργαστή PowerPC ο οποίος υπάρχει εντός του FPGA και αναπτύχθηκαν περιφερειακά τα οποία εκτελούν επιτάχυνση δύο αλγορίθμων κρυπτογράφησης (Triple DES και RTEA).

Η διαδικασία ελέγχεται μέσω εντολών που παρέχονται στον ενσωματωμένο στο FPGA επεξεργαστή PowerPC 405 από μια σειριακή θύρα. Στην ίδια θύρα αποστέλλονται τα δεδομένα που κρυπτογραφούνται και αποκρυπτογραφούνται από το FPGA. Παράλληλα ο αλγόριθμος κρυπτογράφησης εκτελείται και σε λογισμικό από τον επεξεργαστή PowerPC προκειμένου να ελεγχθεί η ορθότητα της λειτουργίας του υλικού αλλά και να φανεί η επιτάχυνση που προσφέρει το υλικό στη διαδικασία κρυπτογράφησης & αποκρυπτογράφησης.

Το κείμενο της παρούσας εργασίας οργανώνεται ως εξής:

Στο Κεφάλαιο 1 περιγράφεται η λειτουργία της Δυναμικής Επαναδιαμόρφωσης, οι δυνατότητες που παρέχει στη σχεδίαση ενός συστήματος καθώς και οι εφαρμογές που μπορεί να βρει σε πραγματικές υλοποιήσεις.

Στο Κεφάλαιο 2 γίνεται μια συνοπτική παρουσίαση της αρχιτεκτονικής του Virtex-4 FPGA, καθώς και των τρόπων προγραμματισμού του. Επίσης περιγράφεται η κάρτα ML403 η οποία χρησιμοποιήθηκε σαν πλατφόρμα για τις δοκιμές μας, ενώ γίνεται και μια σύντομη ανασκόπηση της μεθοδολογίας σχεδίασης συστημάτων με τη χρήση του εργαλείου EDK της Xilinx. Τέλος παρουσιάζονται τα επιπλέον εργαλεία που χρησιμοποιούνται κατά τη σχεδίαση ενός συστήματος που χρησιμοποιεί Δυναμική Επαναδιαμόρφωση.

Στο Κεφάλαιο 3 παρουσιάζεται η διαδικασία επαναδιαμόρφωσης υλικού στα FPGA της Xilinx ενώ αναλύεται η διαδικασία υλοποίησης της δυναμικής επαναδιαμόρφωσης. Η ανάλυση εστιάζεται στην οικογένεια Virtex-4 η οποία χρησιμοποιήθηκε για την πειραματική υλοποίηση.

Στο Κεφάλαιο 4 περιγράφεται η αρχιτεκτονική του συστήματος που αναπτύχθηκε και παρουσιάζονται τα βήματα για την υλοποίηση ενός δυναμικά επαναδιαμορφούμενου συστήματος με τη χρήση των εργαλείων λογισμικού που παρέχει η Xilinx για την ανάπτυξη σε FPGA.

Στο Κεφάλαιο 5 περιγράφονται η εφαρμογή ελέγχου που εκτελείται στον επεξεργαστή PowerPC καθώς και τα βήματα που απαιτούνται για την υλοποίηση του λογισμικού ενώ παρουσιάζεται η λειτουργία της πειραματικής διάταξης.

Τέλος, στο Κεφάλαιο 6 παρατίθενται τα συμπεράσματα και κάποιες προτάσεις για μελλοντικές επεκτάσεις πάνω στη συγκεκριμένη εργασία. Επιπλέον παρέχονται και μετρήσεις απόδοσης του συστήματος του FPGA με και χωρίς την ύπαρξη δυναμικά επαναδιαμορφούμενων τμημάτων.

Στο Παράρτημα 1 περιγράφεται η διαδικασία εγκατάστασης των εργαλείων της Xilinx σε Ubuntu Linux προκειμένου να υποστηριχθεί η δυναμική επαναδιαμόρφωση. Στο Παράρτημα 2 περιγράφεται η Διαφορική Μερική Επαναδιαμόρφωση, μια τεχνική η οποία χρησιμοποιείται για μικρές αλλαγές στη λειτουργικότητα του συστήματος. Στο Παράρτημα 3 καταγράφονται τα περιεχόμενα του συνοδευτικού δίσκου, ενώ στο Παράρτημα 4 δίνεται μια συνοπτική περιγραφή των αλγορίθμων κρυπτογράφησης που χρησιμοποιήθηκαν.

ПЕЧАТНО ПЕРА

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Μιχάλη Ψαράκη για την εμπιστοσύνη που μου έδειξε καθώς και την καθοδήγηση που μου προσέφερε κατά τη διάρκεια εκπόνησης της παρούσας εργασίας. Επιπλέον τους συναδέλφους μου και ιδιαίτερα τον Άρη Νικολογιάννη για τις συμβουλές του ως προς την υλοποίηση του πρακτικού μέρους. Και βέβαια τη μητέρα μου για την αμέριστη συμπαράστασή της σε όλη τη διάρκεια του μεταπτυχιακού προγράμματος.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ

ПЕЧАТНО ПЕРА

Πίνακας Περιεχομένων

Πίνακας Σχημάτων

Συντμήσεις

ASIC	Application Specific Integrated Circuit
CLB	Complex Logic Block
CPLD	Complex Programmable Logic Device
DCM	Digital Clock Manager
DCR	Device Control Register
DES	Data Encryption Standard
DLL	Delay Lock Loop
DR	Dynamic Reconfiguration
EAPR	Early Access Partial Reconfiguration
EDK	Embedded Development Kit
EEPROM	Electrical Erasable Programmable Read Only Memory
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
ICAP	Internal Configuration Access Port
IP	Intellectual Property
IPIF	Intellectual Property Interface
ISE	Integrated Software Environment
JTAG	Joint Test Action Group
LUT	Look-Up Table
MUX	Multiplexer
OPB	On-Chip Peripheral Bus
PCB	Printed Circuit Board
PLB	Processor Local Bus
PR	Partial Reconfiguration
PRM	Partially Reconfigurable Module
PRR	Partially Reconfigurable Region
SEU	Single Event Upset
VHDL	Very high speed integrated circuit Hardware Description Language

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ

ПЕЧАТНО ПЕРА

1

Επισκόπηση της Δυναμικής Επαναδιαμόρφωσης

Στο κεφάλαιο αυτό περιγράφεται η λειτουργία της Δυναμικής Επαναδιαμόρφωσης, οι δυνατότητες που παρέχει στη σχεδίαση ενός συστήματος καθώς και οι εφαρμογές που μπορεί να βρει σε πραγματικές υλοποιήσεις.

1.1 FPGAs και Επαναδιαμόρφωση

Επαναδιαμόρφωση ονομάζεται η δυνατότητα ενός υλικού (hardware) να μεταβάλει τις λογικές συναρτήσεις τις οποίες εκτελεί εσωτερικά. Τη δυνατότητα αυτή έχουν ορισμένα από τα FPGAs τα οποία για την διαμόρφωσή τους χρησιμοποιούν SRAM μνήμες, παρόλο που υπάρχουν και άλλα συστήματα [1] πέρα από τα FPGAs με αυτή τη λειτουργικότητα. Η *Δυναμική Επαναδιαμόρφωση* προσφέρει τη δυνατότητα να αλλάξει κατά τη διάρκεια της λειτουργίας του FPGA η υλοποιούμενη λειτουργικότητα σε κάποιο τμήμα του, ενώ το υπόλοιπο να συνεχίσει να λειτουργεί κανονικά.

Η υλοποίηση της επαναδιαμόρφωσης δεν υποστηρίζεται απευθείας από την τυπική διαδικασία υλοποίησης κυκλώματος σε FPGA. Υπάρχουν διάφορες μεθοδολογίες που έχουν προταθεί κατά καιρούς προκειμένου να υλοποιηθούν συστήματα που εφαρμόζουν δυναμική επαναδιαμόρφωση μιας και η διαδικασία έχει ιδιαίτερες σχεδιαστικές απαιτήσεις. Παρόλα αυτά ο σχεδιαστής πρέπει να λάβει διάφορες αποφάσεις κατά τη διάρκεια της υλοποίησης χωρίς να είναι πάντα ξεκάθαρο αν το αποτέλεσμα θα είναι λειτουργικό και πολύ περισσότερο αν θα είναι το βέλτιστο.

Τα FPGAs που υποστηρίζουν δυναμική επαναδιαμόρφωση προέρχονται κατά βάση από τη Xilinx [2]. Υπάρχουν και κάποια προϊόντα από την Atmel [3] τα οποία υποστηρίζουν δυναμική επαναδιαμόρφωση αλλά η υποστήριξη από τα εργαλεία λογισμικού είναι ανύπαρκτη αφήνοντας τα πάντα στο σχεδιαστή.

1.2 Πλεονεκτήματα & Μειονεκτήματα της Δυναμικής Επαναδιαμόρφωσης

Η χρήση ΔΕ στη σχεδίαση ενός συστήματος είναι μια απόφαση που θα πρέπει να ληφθεί αφού αξιολογηθούν διάφορες παράμετροι έτσι ώστε να αποσαφηνισθεί αν όντως είναι απαραίτητη ή όχι. Βασικά πλεονεκτήματα της χρήσης ΔΕ στη σχεδίαση ενός συστήματος είναι:

- Οικονομία στους πόρους του FPGA, επομένως δυνατότητα χρήσης μικρότερης και συνεπώς φθηνότερης συσκευής.
- Δυνατότητα χρήσης FPGA μικρότερης συσκευασίας με λιγότερους ακροδέκτες, κάτι που οδηγεί σε πιο εύκολη κατασκευή του τυπωμένου κυκλώματος του τελικού συστήματος και επιτάχυνση της διαδικασίας συναρμολόγησης - ελέγχου - πιστοποίησής του.
- Χαμηλότερη κατανάλωση ενέργειας λόγω χρήσης μικρότερης συσκευής αλλά και του ότι δεν υπάρχουν τμήματα του FPGA τα οποία λειτουργούν ενώ δεν είναι απαραίτητο.
- Αυξημένες επιδόσεις του υλοποιούμενου κυκλώματος αφού τα διάφορα τμήματά του βρίσκονται τοποθετημένα πιο κοντά μεταξύ τους, οπότε μειώνονται οι καθυστερήσεις διάδοσης των ενδιαμέσων σημάτων.
- Μικρότερος χρόνος αρχικής διαμόρφωσης του FPGA λόγω του ότι χρησιμοποιείται μικρότερη συσκευή οπότε το FPGA είναι διαθέσιμο στο συνολικό σύστημα ταχύτερα

Αντίθετα, η χρήση ΔΕ σε ένα σύστημα έχει διάφορα μειονεκτήματα, όπως:

- Η ΔΕ δεν είναι πάντοτε εφικτή λόγω της απαίτησης για εφαρμογή της σε τμήματα του κυκλώματος τα οποία δεν είναι *ταυτόχρονα* ενεργά.
- Η σχεδίαση ενός κυκλώματος που χρησιμοποιεί ΔΕ απαιτεί επιπλέον βήματα στην υλοποίηση.
- Το «φόρτωμα» μιας νέας διαμόρφωσης απαιτεί κάποιον χρόνο. Αντίθετα σε μια συμβατική στατική υλοποίηση μετά τη διαμόρφωσή του FPGA όλη η λειτουργικότητά του είναι ανά πάσα στιγμή διαθέσιμη.

1.3 Εφαρμογές Δυναμικής Επαναδιαμόρφωσης

Η ΔΕ είναι ενεργό πεδίο έρευνας την τελευταία δεκαετία ενώ πολλές μελέτες αφορούν τις εφαρμογές εκείνες που μπορεί να αποκομίσουν τα περισσότερα οφέλη από αυτήν. Υπάρχουν τρεις κατευθύνσεις που ερευνώνται ως πιθανά πεδία εφαρμογής της ΔΕ:

i. Βελτίωση της αξιοποίησης του χώρου που καταλαμβάνει σε ένα FPGA κάποιο δεδομένο σύστημα
ii. Αντιστάθμιση δυσλειτουργιών ενός συστήματος σε πραγματικό χρόνο
iii. Εξ' αποστάσεως αναβάθμιση της λειτουργικότητας ενός συστήματος

Η βελτίωση της αξιοποίησης του χώρου που καταλαμβάνει σε κάποιο FPGA ένα υποσύστημα είναι η πιο συνηθισμένη χρήση της ΔΕ. Η ιδέα είναι να μπορούν να εναλλαχθούν στο FPGA τμήματα τα οποία υλοποιούν λειτουργίες που δεν είναι ταυτόχρονα ενεργές. Ταυτόχρονα τα υπόλοιπα τμήματα του FPGA να συνεχίζουν απρόσκοπτα τη λειτουργία τους.

Σε σχέση με τη βελτίωση της αξιοποίησης χώρου, οι εφαρμογές που θα μπορούν να επωφεληθούν στην πράξη από τη ΔΕ πρέπει να έχουν κάποια χαρακτηριστικά:

- Πρέπει να επιδέχονται παραλληλισμό και αμοιβαίο αποκλεισμό λειτουργιών. Επίσης η απόδοσή τους όταν υλοποιούνται σε υλικό να είναι πολύ ανώτερη από την αντίστοιχη υλοποίηση με χρήση κάποιου επεξεργαστή η οποία κατά τεκμήριο οδηγεί σε απλούστερη σχεδίαση.
- Πρέπει οι απαιτήσεις του τελικού συστήματος σε επιφάνεια πάνω στην πλακέτα (PCB) ή σε κόστος υλικών να επιτάσσουν τη χρήση του FPGA με τις μικρότερες δυνατές διαστάσεις. Σε αντίθετη περίπτωση ίσως είναι προτιμότερη η χρήση ενός μεγαλύτερου FPGA.
- Τα τμήματα του συστήματος που θα εναλλάσσονται δυναμικά εντός του FPGA θα πρέπει να έχουν τέτοιες διαστάσεις που όταν φορτώνεται στο FPGA η διαμόρφωση εκείνη που απαιτεί τον περισσότερο χώρο, να μπορεί να χωρέσει σε FPGA μικρότερο από εκείνο που θα απαιτούσε η πλήρως στατική υλοποίηση.

- Οι επαναδιαμορφώσεις του FPGA να γίνονται σπάνια συγκριτικά με το χρόνο στον οποίο κάθε διαμόρφωση παραμένει ενεργή. Σε αντίθετη περίπτωση η καθυστέρηση που εισάγεται από την αλλαγή διαμόρφωσης θα δημιουργεί πρόβλημα στη λειτουργία του συστήματος, πέραν της αυξημένης κατανάλωσης ισχύος.
- Το επιπλέον κόστος σε χρήμα, σε τυχόν αύξηση της επιφάνειας της πλακέτας (PCB) του συστήματος και σε κατανάλωση ισχύος για τη μεγαλύτερη μνήμη αποθήκευσης των bitstreams να αντισταθμίζεται από τη χρήση μικρότερου FPGA.
- Το κύκλωμα που ελέγχει την επαναδιαμόρφωση απαιτεί κάποιους πόρους του FPGA. Θα πρέπει τα οφέλη από την επαναδιαμόρφωση να συνυπολογιστούν με τις απώλειες εξαιτίας των πόρων αυτών.

Με βάση τα παραπάνω αποδεικνύεται ότι στην πράξη περιορίζονται οι εφαρμογές όπου πραγματικά έχει νόημα να χρησιμοποιηθεί η ΔΕ. Ωστόσο, λαμβάνοντας υπόψη ότι η τιμή και η κατανάλωση ενός FPGA είναι σημαντικοί παράγοντες για τη χρήση του, μέχρι η πρόοδος της τεχνολογίας να επιφέρει μείωση σε αυτούς η ΔΕ μπορεί να αποτελέσει εφικτή σχεδιαστική προσέγγιση για φορητά συστήματα χαμηλού κόστους.

Μια άλλη κατεύθυνση χρήσης της ΔΕ είναι για την αντιστάθμιση δυσλειτουργιών ενός συστήματος υπό λειτουργία. Αυτό γίνεται σε εφαρμογές εντός περιβάλλοντος επιβεβαρυμμένου από σωματίδια τα οποία μπορεί να προκαλέσουν δυσλειτουργία (όπως λειτουργία στο διάστημα, σε πυρηνικούς σταθμούς κλπ). Σε τέτοιες περιπτώσεις προσβολή από σωματίδια μπορεί να οδηγήσει σε SEUs (Single Event Upsets) τα οποία μπορεί να αλλοιώσουν την κατάσταση ενός flip-flop εντός του FPGA.

Τέλος, η ΔΕ μπορεί να εφαρμοστεί για εξ' αποστάσεως αναβάθμιση της λειτουργικότητας ενός συστήματος με χρήση κάποιο ασύρματου καναλιού. Τέτοιου είδους εφαρμογές θα μπορούσαν να είναι σε συστήματα εντός δορυφόρων που βρίσκονται σε τροχιά στο διάστημα και γενικότερα όπου είναι εκ των πραγμάτων δύσκολο (ή ακόμη και ανέφικτο) να γίνει αντικατάσταση ενός συστήματος κατά τη διάρκεια λειτουργίας του.

1.4 Παραδείγματα συστημάτων που χρησιμοποιούν Δυναμική Επαναδιαμόρφωση

Στη βιβλιογραφία υπάρχουν πάρα πολλά παραδείγματα συστημάτων που χρησιμοποιούν ΔΕ στα πλαίσια ερευνητικών εργασιών τα οποία καλύπτουν ευρύ φάσμα εφαρμογών. Εμπορικά συστήματα που χρησιμοποιούν ΔΕ είναι, προφανώς, πιο δύσκολο να εντοπιστούν. Ένα παράδειγμα χρήσης της ΔΕ έχουμε στο Μεγάλο Επιταχυντή Αδρονίων (LHC) [4] του ινστιτούτου CERN στη Γενεύη. Στην υλοποίηση του συστήματος υπάρχει μια συστοιχία FPGA [5] η οποία υπόκειται σε συνεχή ΔΕ προκειμένου να αντισταθμίσει σφάλματα SEU από σωματίδια εντός του επιταχυντή [6].

Η Xilinx έχει παρουσιάσει διάφορα παραδείγματα ολοκληρωμένων συστημάτων [7], [8] που χρησιμοποιούν ΔΕ και θα μπορούσαν να χρησιμοποιηθούν σε εμπορικές υλοποιήσεις προϊόντων. Στο [7] παρουσιάζεται ένα σύστημα βιομετρικού ελέγχου για ανάλυση δακτυλικών αποτυπωμάτων το οποίο χρησιμοποιεί ΔΕ προκειμένου να υλοποιεί τα διαφορετικά στάδια επεξεργασίας του συστήματος. Στη συγκεκριμένη εφαρμογή η ΔΕ είναι ιδιαίτερα αποδοτική μιας και η διαδικασία αναγνώρισης έχει πολλά ανεξάρτητα στάδια που εκτελούνται διαδοχικά. Στο [9] παρουσιάζεται μια εφαρμογή που προσφέρει στον οδηγό ενός αυτοκινήτου οπτική υποβοήθηση ώστε να αντιλαμβάνεται έγκαιρα πιθανά επικίνδυνες καταστάσεις. Η ΔΕ χρησιμοποιείται προκειμένου να φορτώνονται κάθε φορά διαφορετικοί αλγόριθμοι επεξεργασίας εικόνας ανάλογα με τις συνθήκες του περιβάλλοντος (μέρα, νύχτα, βροχή, ηλιοφάνεια, ομίχλη κλπ).

ПЕЧАТНО ПЕРА

2

Πλατφόρμα Ανάπτυξης Υλικού & Λογισμικού

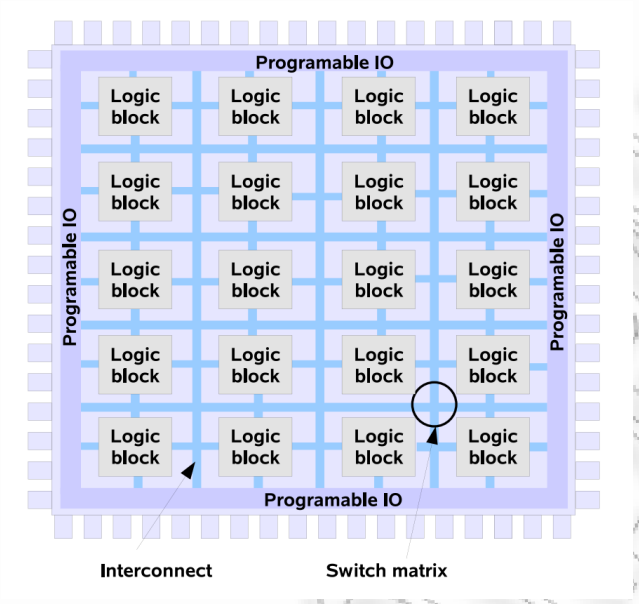
Στο κεφάλαιο αυτό δίνεται μια συνοπτική παρουσίαση των FPGAs και της αρχιτεκτονικής τους με έμφαση στην οικογένεια Virtex-4 της Xilinx η οποία χρησιμοποιήθηκε στην παρούσα εργασία. Ακόμη παρουσιάζονται τα εργαλεία λογισμικού που είναι απαραίτητα για την υλοποίηση ενός συστήματος σε FPGA καθώς και τα επιπλέον εργαλεία που απαιτούνται προκειμένου να σχεδιαστεί ένα σύστημα που επιτρέπει ΔΕ.

2.1 FPGAs

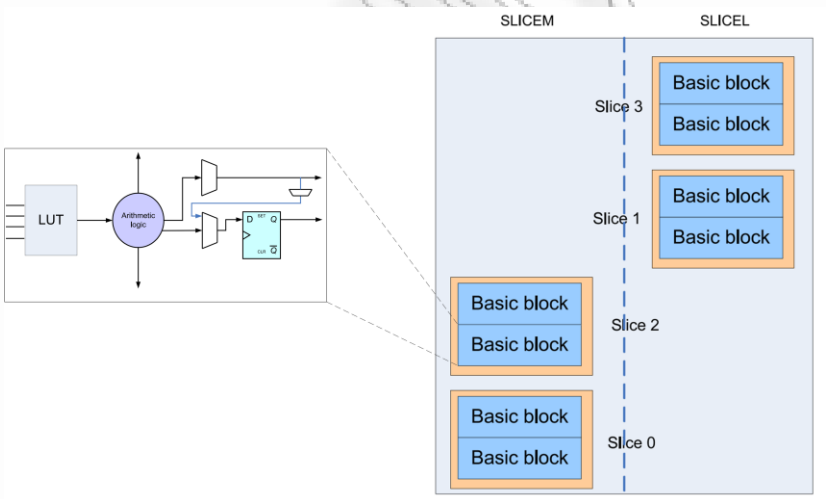
Τα FPGAs είναι πολλαπλά επαναπρογραμματιζόμενες συσκευές που μπορεί να υλοποιούν κυκλώματα συνδυαστικής και ακολουθιακής λογικής. Ο προγραμματισμός τους γίνεται επιτόπου ("Field Programmable"), κάτι που επιτρέπει αλλαγές λειτουργικότητας ακόμα και σε έτοιμα συστήματα. Η ταχύτητά τους είναι αρκετά υψηλή (άνω των 500MHz) ενώ προσφέρουν ένα σύνολο πόρων με τη βοήθεια των οποίων μπορεί να υλοποιηθούν εφαρμογές επεξεργασίας δεδομένων, όπως συμπίεση video, κρυπτογράφηση δεδομένων, επεξεργασία σημάτων. Η ανάπτυξη εφαρμογών FPGA συνήθως γίνεται με χρήση γλωσσών περιγραφής υλικού, με συνηθέστερες τις VHDL, Verilog και SystemC/HandelC.

2.2 Αρχιτεκτονική FPGA

Τα FPGAs αποτελούνται από τμήματα *Σύνθετων Μονάδων Λογικής (Complex Logic Blocks, CLBs)* τα οποία είναι τοποθετημένα σε διάταξη μήτρας. Τα CLBs μπορούν να προγραμματιστούν ανεξάρτητα προκειμένου να εκτελέσουν διάφορες πράξεις συνδυαστικής και ακολουθιακής λογικής. Επιπλέον τα FlipFlops μπορούν να χρησιμοποιηθούν ως στοιχεία μνήμης. Η διασύνδεση μεταξύ τους καθώς και με τους ακροδέκτες του ολοκληρωμένου κυκλώματος είναι και αυτή προγραμματιζόμενη μέσω ενός μεγάλου αριθμού διασυνδέσεων. Το Σχήμα 1 δείχνει μια απλουστευμένη άποψη της βασικής δομής ενός FPGA ενώ το Σχήμα 2 δείχνει τη δομή ενός CLB.



Σχήμα 1: Βασική Δομή ενός FPGA

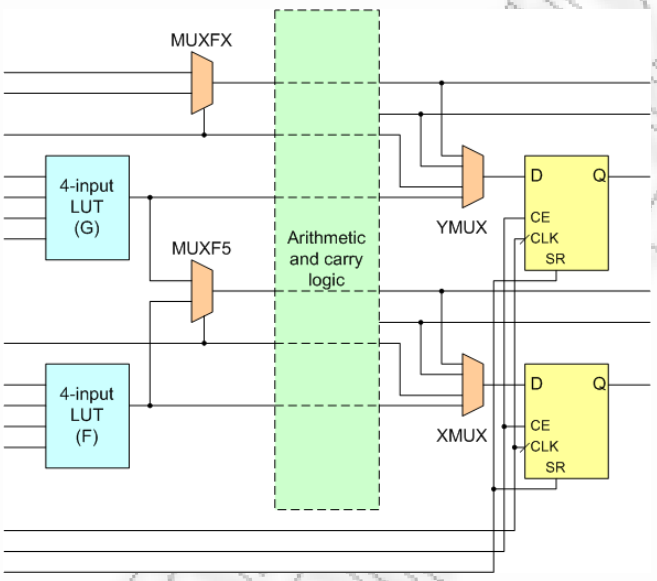


Σχήμα 2: Δομή CLB

Ένα FPGA περιλαμβάνει από εκατοντάδες έως αρκετές χιλιάδες CLBs. Οι σύγχρονες υλοποιήσεις CLBs είναι ιδιαίτερα πολύπλοκες και μπορεί να περιέχουν επιπλέον μνήμη, καταχωρητές ολίσθησης και πολυπλέκτες. Πέραν των CLB όμως, στα σύγχρονα FPGA υπάρχουν τμήματα που περιέχουν επιπλέον μνήμη, μονάδες που υλοποιούν πρόσθεση και πολλαπλασιασμό, μονάδες παραγωγής σημάτων ρολογιού (με πολλαπλασιαστές, διαιρέτες) (CLKDLL, DCM) ή και ολοκληρωμένα υποσυστήματα όπως επεξεργαστές (PowerPC, ARM) καθώς και ελεγκτές δικτύου Ethernet (10/100/1000Mbit). Αποτέλεσμα όλων αυτών είναι η δυνατότητα σχεδίασης σύνθετων διατάξεων που καλύπτουν ευρύ φάσμα εφαρμογών, στις οποίες το μεγαλύτερο μέρος της λειτουργικότητας (αν όχι ολόκληρη) υποστηρίζεται από το FPGA. Έτσι ενώ αρχικά χρησιμοποιούνταν για διασύνδεση ετερογενών συσκευών (*"glue logic"*) πλέον μπορούν να υλοποιήσουν πλήρη κυκλώματα ψηφιακής επεξεργασίας σημάτων, φίλτρα, επεξεργαστές ή μονάδες για συμπίεσης δεδομένων. Άλλα παραδείγματα αφορούν τηλεπικοινωνιακά συστήματα για ψηφιακή διαμόρφωση σημάτων αλλά και μεταγωγή πακέτων και έλεγχο δικτύων. Επίσης χρησιμοποιούνται σε υλοποίηση νευρωνικών δικτύων, γενετικών αλγορίθμων κ.ο.κ.

2.3 Η Οικογένεια Virtex-4

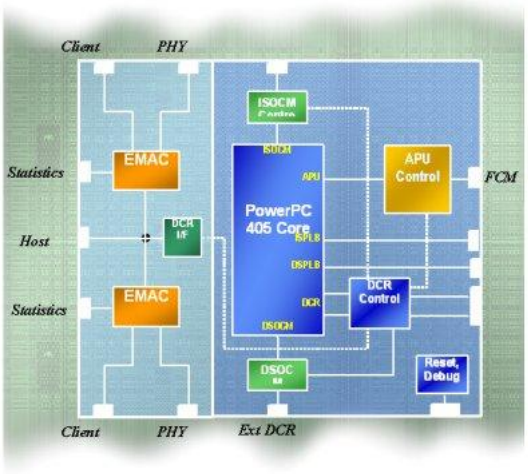
Στα πλαίσια εκπόνησης της παρούσας εργασίας χρησιμοποιήθηκε το FPGA FX12 της οικογένειας Virtex-4, του οποίου το CLB αποτελείται από τέσσερα τμήματα που ονομάζονται *slices*. Κάθε slice περιλαμβάνει 2 look-up tables (LUTF, LUTG) που μπορούν να υλοποιήσουν οποιαδήποτε λογική πράξη με 4 εισόδους, 2 πολυπλέκτες (MUXFX, MUXF5), μια αριθμητική μονάδα και 2 καταχωρητές 1-bit που μπορούν να χρησιμοποιηθούν είτε σαν flip-flop είτε σαν latch. Η είσοδός τους επιλέγεται από 2 πολυπλέκτες (XMUX, YMUX). Στο Σχήμα 3 παρουσιάζεται ένα απλοποιημένο διάγραμμα της δομής ενός slice [10].



Σχήμα 3: Απλοποιημένο διάγραμμα slice της οικογένειας Virtex-4

Το FPGA Virtex4-FX12 που χρησιμοποιήθηκε περιέχει 5472 slices που αντιστοιχούν σε περίπου 12000 λογικές πύλες. Επίσης περιλαμβάνει 36 στοιχεία ξεχωριστής μνήμης SRAM (BlockRAM), που το καθένα έχει χωρητικότητα 18Kbit. Ακόμη περιέχει 32 DSP slices, καθένα αποτελούμενο από έναν πολλαπλασιαστή 18x18, έναν αθροιστή και έναν καταχωρητή. Επιπλέον περιέχει 4 ελεγκτές ρολογιού (Digital Clock Managers). Αυτοί προσφέρουν ακριβές σήμα ρολογιού με χαμηλό θόρυβο και ολίσθηση ενώ μπορούν να κάνουν πολλαπλασιασμό και διαίρεση συχνότητας προκειμένου να παραχθούν εσωτερικά στο FPGA οι όποιες απαιτούμενες συχνότητες χρονισμού. Τέλος το Virtex4-FX12 περιλαμβάνει δύο ελεγκτές Ethernet (10/100/1000Mbps) καθώς και έναν επεξεργαστή PowerPC (hard-core macros).

Στο Σχήμα 4 φαίνεται το διάγραμμα του PowerPC και των ελεγκτών Ethernet [11].

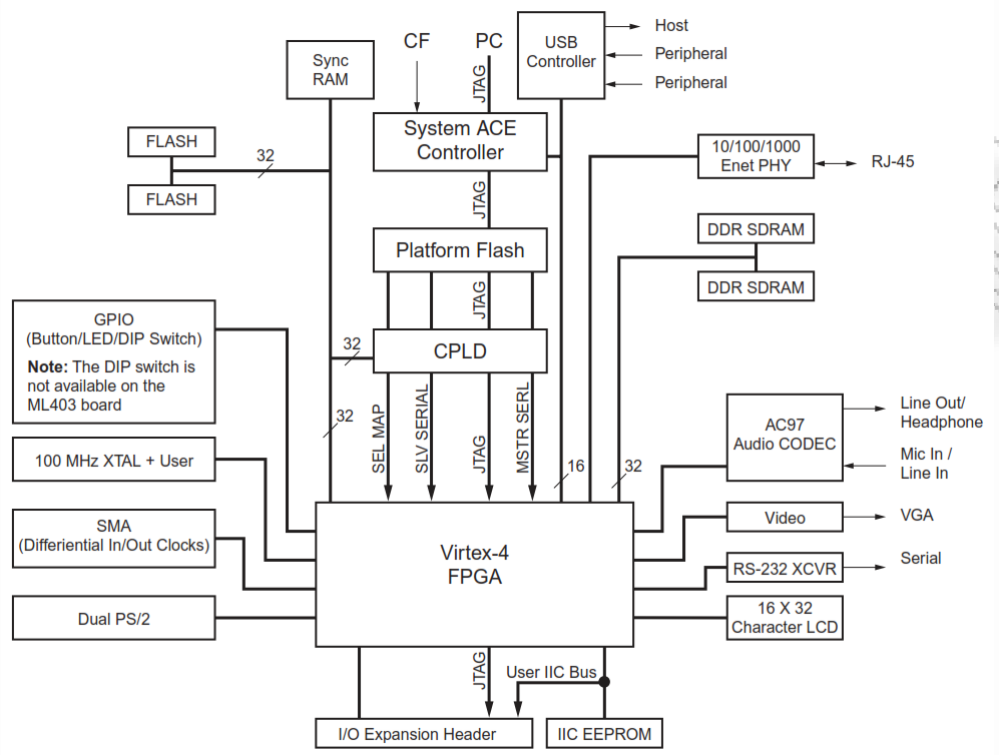


Σχήμα 4: Διάγραμμα του PowerPC και των Ethernet MAC στο Virtex-4 FX12

2.4 Το Αναπτυξιακό Σύστημα ML403

Το αναπτυξιακό σύστημα ML403 το οποίο χρησιμοποιήθηκε στην παρούσα εργασία περιλαμβάνει ένα πλήθος περιφερειακών συσκευών και επιτρέπει την ανάπτυξη πολλών διαφορετικών εφαρμογών που κάνουν χρήση του FPGA. Το δομικό διάγραμμα της ML403 φαίνεται στο Σχήμα 5 [12] ενώ τα βασικά χαρακτηριστικά της είναι τα ακόλουθα :

- XC4VFX12-FF668-10 Virtex-4 FPGA
- 64-MB DDR SDRAM, 32-bit interface για ρυθμό δεδομένων μέχρι 266MHz
- Ένα ζευγάρι εισόδου διαφορικού ρολογιού και ένα ζευγάρι εξόδου διαφορικού ρολογιού με SMA συνδετήρες.
- Έναν clock oscillator 100MHz
- LEDs και πλήκτρα επιλογής (push buttons)
- Κεφαλές επέκτασης (expansion headers) με 32 single-ended I/O, 16 διαφορικά κανάλια συμβατά με LVDS, 14 διαθέσιμα I/Os που μοιράζονται με πλήκτρα και LEDs, τροφοδοσία, δυνατότητα επέκτασης της JTAG αλυσίδας και επέκταση του IIC δίαυλου.
- Stereo AC97 audio codec με υποδοχές line-in, line-out, ακουστικά 50mW και είσοδο μικροφώνου
- Σειριακή θύρα RS-232
- LCD οθόνη 2x16 χαρακτήρων
- Μια 4-Kb IIC EEPROM
- Έξοδο VGA με 140 MHz / 15-bit video DAC
- PS/2 θύρες για ποντίκι και πληκτρολόγιο
- Ελεγκτή διαμόρφωσης System ACE CompactFlash με Type I/II CompactFlash θύρα.
- ZBT σύγχρονη SRAM 8 Mb σε 32-bit δίαυλο δεδομένων χωρίς bit ισοτιμίας
- Intel StrataFlash (ή συμβατά) linear flash chips (8MB)
- 10/100/1000Mbps Ethernet PHY
- USB interface chip (Cypress CY7C67300) με θύρες host και device
- Xilinx XC95144XL CPLD για να επιτρέπει linear flash chips να χρησιμοποιούνται για τη διαμόρφωση του FPGA
- Xilinx XCF32P Platform Flash για αποθήκευση αρχείου διαμόρφωσης
- JTAG είσοδο
- Τροφοδοτικά επί της πλακέτας για όλες τις απαραίτητες τάσεις

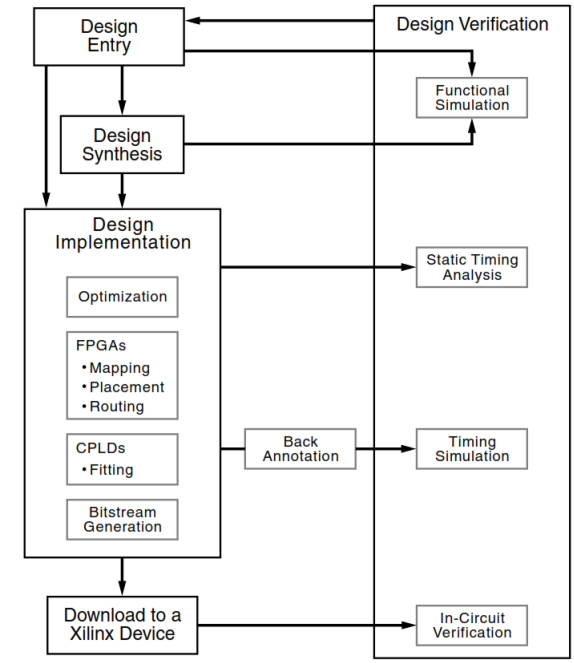


Σχήμα 5: Δομικό Διάγραμμα Αναπτυξιακής Πλακέτας ML403

2.5 Λογισμικό Ανάπτυξης Εφαρμογών FPGA

Προκειμένου να γίνει η ανάπτυξη μιας εφαρμογής σε FPGA, χρησιμοποιείται μια γλώσσα περιγραφής υλικού με συνηθέστερες τις VHDL και Verilog και σπανιότερα τις SystemC/HandelC προκειμένου να γίνει η περιγραφή του κυκλώματος (Design Entry). Κατόπιν ακολουθείται μια διαδικασία προκειμένου η περιγραφή του υλικού να καταλήξει στο κύκλωμα που θα υλοποιηθεί με το συγκεκριμένο FPGA. Η διαδικασία αυτή περιλαμβάνει τα στάδια της Σύνθεσης (*Design Synthesis*), όπου η αρχική περιγραφή της λειτουργικότητας μεταφράζεται σε γενικό κύκλωμα πυλών που περιλαμβάνει τα διαθέσιμα δομικά στοιχεία του FPGA. Κατόπιν ακολουθεί η φάση της Υλοποίησης (*Implementation*) όπου το αρχείο εξόδου της σύνθεσης περνά διαδοχικά τα στάδια του Mapping, Placement και Routing. Εκεί καθορίζονται ποιοι από τους διαθέσιμους πόρους χρησιμοποιούνται, πού ακριβώς βρίσκονται οι πόροι αυτοί και πώς διασυνδέονται. Το αποτέλεσμα είναι ένα αρχείο διαμόρφωσης (*configuration file*) το οποίο και χρησιμοποιείται για να προγραμματίσει το FPGA.

Παράλληλα με τις διαδικασίες Σύνθεσης και Υλοποίησης εκτελείται και η διαδικασία της επαλήθευσης, όπου το αποτέλεσμα κάθε βήματος ελέγχεται προκειμένου να φάνει κατά πόσον είναι σύμφωνο με τις απαιτήσεις της εφαρμογής, σε ότι αφορά την υλοποιούμενη λειτουργικότητα, την ταχύτητα υλοποίησης καθώς και το χώρο που καταλαμβάνει στο FPGA. Ο έλεγχος γίνεται με εργαλεία εξομοίωσης όπως Mentor Graphics ModelSim [13], Xilinx ISim [14] καθώς και με εργαλεία ανάλυσης αποτελεσμάτων (Static Timing Analysis Tools). Το Σχήμα 6 [15] δείχνει τη διαδικασία υλοποίησης ενός συστήματος σε Xilinx FPGA.



Σχήμα 6: Διαδικασία Υλοποίησης Συστήματος σε Xilinx FPGA

2.5.1 Περιβάλλον ISE

Ολόκληρη η διαδικασία Σύνθεσης και Υλοποίησης υποστηρίζεται από το εργαλείο ISE (Integrated Software Environment). Αυτό είναι ένα GUI (Graphics User Interface), το οποίο ανάλογα με το στάδιο στο οποίο βρίσκεται κάθε φορά η υλοποίηση καλεί τα αντίστοιχα εργαλεία γραμμής εντολών (όπως partgen, ngdbuild, map, par κ.α.) δίνοντάς τους τις κατάλληλες παραμέτρους. Επίσης μπορεί να καλεί άλλα εργαλεία, όπως π.χ. τον Constraints Editor ο οποίος δημιουργεί το αρχείο περιορισμών της υλοποίησης (constraints file) ή τον εξομοιωτή κυκλώματος (π.χ. ModelSim) με τη βοήθεια του οποίου μπορούμε να εξετάσουμε κατά πόσον η υλοποιούμενη λειτουργικότητα είναι η απαιτούμενη.

2.5.2 Περιβάλλον EDK

Το EDK (Embedded Development Kit) είναι ένα εργαλείο με το οποίο περιγράφονται συστήματα βασισμένα στον επεξεργαστή PowerPC ο οποίος βρίσκεται ενσωματωμένος σε διάφορες οικογένειες FPGA της Xilinx. Επίσης το EDK μπορεί να χρησιμοποιηθεί για συστήματα που περιέχουν τον microBlaze, τον 32bit επεξεργαστή της Xilinx ο οποίος είναι διαθέσιμος σε μορφή soft-core macro, δεν αποτελεί δηλαδή συγκεκριμένο πόρο του FPGA αλλά μπορεί να υλοποιηθεί εκ των υστέρων. Στο EDK ορίζεται ακριβώς η διαμόρφωση του συστήματος του επεξεργαστή, τα περιφερειακά του, οι δίαυλοι διασύνδεσης, η μνήμη προγράμματος και δεδομένων, η λανθάνουσα μνήμη αλλά και η επικοινωνία με τον έξω κόσμο.

Όταν οριστεί το σύστημα στο EDK μπορούν πλέον να κληθούν τα εργαλεία που θα κάνουν την υλοποίηση του συστήματος αντίστοιχα με τον τρόπο που καλούνται μέσα από το ISE. Στη γενική περίπτωση, η σχεδίαση με το EDK είναι αυτόνομη και δε χρειάζεται να χρησιμοποιηθεί καθόλου το ISE. Ωστόσο, μπορεί να γίνει και μικτή σχεδίαση, όταν το σύστημα που αναπτύσσεται με το EDK δεν είναι αυτόνομο αλλά αποτελεί τμήμα ενός μεγαλύτερου συστήματος. Σε αυτήν την περίπτωση το ISE μπορεί να εισαγάγει τα αρχεία που παρήγαγε το EDK και η υλοποίηση να ολοκληρωθεί από το περιβάλλον του ISE.

Επιπλέον, το EDK χρησιμοποιείται και για την ανάπτυξη της εφαρμογής που εκτελείται στον επεξεργαστή δημιουργώντας τα κατάλληλα αρχεία για το linker και καλώντας τον C-compiler, τον

assembler και τον linker. Όταν ολοκληρωθούν όλα τα βήματα υλοποίησης, δημιουργείται ένα συνολικό αρχείο διαμόρφωσης το οποίο αρχικοποιεί το FPGA.

2.5.3 PlanAhead

Το PlanAhead είναι ένα εργαλείο που χρησιμοποιείται μετά τη σύνθεση ενός κυκλώματος και παρέχει στο σχεδιαστή τη δυνατότητα να ελέγξει την τοποθέτηση των τμημάτων ενός ιεραρχικού συστήματος επί της επιφάνειας του FPGA και να ορίσει περιορισμούς (constraints). Επιπλέον επιτρέπει την κλήση των υπολοίπων εργαλείων υλοποίησης του FPGA οπότε μπορεί η υλοποίηση να ολοκληρωθεί μέσα από αυτό το περιβάλλον χωρίς ο χρήστης να χρειαστεί να επανέλθει στο ISE. Παρόλο που δεν είναι απαραίτητο να χρησιμοποιηθεί στην ανάπτυξη μιας εφαρμογής, η χρήση του PlanAhead μπορεί να οδηγήσει σε ένα τελικό σύστημα υψηλότερης απόδοσης. Ωστόσο τη σχεδίαση με Δυναμική Επαναδιαμόρφωση και σύμφωνα με την τρέχουσα μεθοδολογία η οποία και χρησιμοποιήθηκε στην παρούσα εργασία, η χρήση του PlanAhead είναι επιβεβλημένη.

2.5.4 Εργαλεία για υποστήριξη της Δυναμικής Επαναδιαμόρφωσης

Μέχρι την έκδοση 12.1 των εργαλείων της Xilinx δεν υπήρχε εγγενής υποστήριξη για ΔΕ. Ωστόσο στα πλαίσια ενός προγράμματος προεπισκόπησης της διαδικασίας ΔΕ (Partial Reconfiguration Early Access Program), η Xilinx διέθετε δωρεάν σε εξουσιοδοτημένους χρήστες μέχρι τον Ιούλιο του 2010 υποστήριξη ΔΕ με επιπλέον εγκατάσταση λογισμικού (software patch) το οποίο τροποποιεί την κανονική έκδοση των εργαλείων.

Οι εκδόσεις για τις οποίες παρείχετο υποστήριξη ήταν οι 8.1, 9.1 και 9.2. Πλέον η υποστήριξη δεν είναι διαθέσιμη από την ιστοσελίδα [16] αφού η διαδικασία ΔΕ υποστηρίζεται επίσημα από την τρέχουσα έκδοση (12.x) του ISE.

Για την εκπόνηση της παρούσας εργασίας χρησιμοποιήθηκε η έκδοση ISE 9.1_SP2 σε συνδυασμό με την EDK9.1_SP1. Η ISE9.1_SP2 ενημερώθηκε με την έκδοση PR10 των εργαλείων ΔΕ.

Πέραν της ενημερωμένης έκδοσης 9.1 του ISE για υποστήριξη ΔΕ, απαιτείται η χρήση του PlanAhead, έκδοση 10.1 ή νεώτερη.

Το λειτουργικό σύστημα που χρησιμοποιήθηκε ήταν το Ubuntu 9.1 και όλη η ανάπτυξη έγινε σε περιβάλλον Linux το οποίο λειτουργούσε σε εικονική μηχανή μέσα από το περιβάλλον του Virtual Box [17]. Αυτό διότι τα εργαλεία της Xilinx για Windows πλατφόρμα χρησιμοποιούν το περιβάλλον Cygwin [18] το οποίο όμως δε λειτουργεί σωστά εάν εγκατασταθεί πάνω από μια φορά σε ένα μηχάνημα. Επειδή υπάρχουν διάφορες εφαρμογές Windows που χρησιμοποιούν το Cygwin ώστε να εξομοιώσουν κλήσεις λειτουργικού Unix (όπως Xilinx ISE/EDK, TinyOS, εφαρμογές X-Server κλπ) πολλές φόρες δημιουργούνται προβλήματα, όπως συνέβη και στο μηχάνημα που χρησιμοποιήθηκε πριν αναγκαστούμε να προχωρήσουμε σε χρήση πλατφόρμας εικονικής μηχανής.

ПЕЧАТНО ПЕРА

3

Διαμόρφωση και Επαναδιαμόρφωση Υλικού

Στο κεφάλαιο αυτό περιγράφεται η διαδικασία επαναδιαμόρφωσης υλικού στα FPGA της Xilinx και αναλύεται η διαδικασία υλοποίησής της. Η ανάλυση εστιάζεται στην οικογένεια Virtex-4 αν και με μικρές διαφορές ανάλογα με κάποια εγγενή χαρακτηριστικά τους, η διαδικασία είναι αντίστοιχη και στις υπόλοιπες οικογένειες Virtex (Virtex-II, Virtex-5, Virtex-6).

3.1 Προγραμματισμός FPGA

Ο προγραμματισμός ενός FPGA επιτυγχάνεται με τη βοήθεια ενός αρχείου διαμόρφωσης (configuration file ή bitstream) το οποίο προκύπτει μετά την εκτέλεση της διαδικασία ανάπτυξης εφαρμογής. Το αρχείο διαμόρφωσης αποθηκεύεται σε μια εξωτερική μνήμη ώστε να μπορεί να φορτωθεί σε κάθε εκκίνηση του συστήματος. Όταν φορτωθεί στο FPGA προγραμματίζει κατάλληλα την SRAM μνήμη διαμόρφωσης και το FPGA εκτελεί την απαιτούμενη λειτουργικότητα.

3.2 Ορισμοί

Στα παρακάτω δίνονται κάποιοι ορισμοί με σκοπό να διαφοροποιηθούν τα είδη διαμόρφωσης σε ένα FPGA.

Διαμόρφωση (Configuration)

Ο όρος έχει διπλή σημασία: αφορά το αρχείο προγραμματισμού του FPGA (bitstream) αλλά και τη λειτουργικότητα που αποκτά ένα FPGA μετά τη φόρτωση σε αυτό ενός bitstream. Μπορεί να είναι *Στατική* ή *Δυναμική*.

- ***Στατική ή Πλήρης (Static/Full Configuration)***: Ολόκληρο το FPGA προγραμματίζεται και η διαμόρφωση παραμένει σταθερή σε όλη τη διάρκεια της λειτουργίας του.
- ***Δυναμική (Dynamic)***: Τμήματα της λειτουργικότητας του FPGA αλλάζουν δυναμικά κατά τη διάρκεια της λειτουργίας του. Τότε έχουμε τη λειτουργία της *επαναδιαμόρφωσης*.

Επαναδιαμόρφωση (Reconfiguration)

Η διαδικασία φόρτωσης μιας νέας λειτουργικότητας στο FPGA με κατάργηση της προηγούμενης. Παραβλέποντας την τετριμμένη περίπτωση της αρχικής διαμόρφωσης, η επαναδιαμόρφωση αφορά μόνο τα FPGA που χρησιμοποιούν SRAM για την αποθήκευση της διαμόρφωσης. Η επαναδιαμόρφωση συνήθως κατηγοριοποιείται με βάση το *«πότε», «πώς»* και *«πού»* συμβαίνει [19]. Έτσι σε σχέση με το:

- ***«πότε»:*** Η επαναδιαμόρφωση μπορεί να είναι *στατική (static)* ή *δυναμική (dynamic)*. Η στατική επαναδιαμόρφωση λαμβάνει χώρα όταν το FPGA δεν είναι ενεργό ενώ η δυναμική όταν το FPGA βρίσκεται σε κατάσταση λειτουργίας.
- ***«πώς»:*** Η επαναδιαμόρφωση μπορεί να είναι *πλήρης (full)* ή *μερική (partial)*. Η πλήρης διαμόρφωση αφορά ολόκληρο το FPGA ενώ η μερική ένα τμήμα του, κάτι που σημαίνει ότι το υπόλοιπο συνεχίζει κανονικά τη λειτουργία του.
- ***«πού»:*** Η επαναδιαμόρφωση μπορεί να είναι εξωτερική (external) ή εσωτερική (internal). Στην περίπτωση της εξωτερικής επαναδιαμόρφωσης, κάποια άλλη συσκευή επαναπρογραμματίζει το FPGA ενώ στην εσωτερική το FPGA φορτώνει από μόνο του το αρχείο διαμόρφωσης. Σε αυτήν την περίπτωση ένας εσωτερικός μηχανισμός θα πρέπει να αναλάβει τη διαδικασία. Για την περίπτωση αυτή υπάρχει και ο όρος *«αυτό-επαναδιαμόρφωση» (self-reconfiguration)* ή Run-Time Reconfiguration.

Δυναμική Περιοχή (Dynamic Region/ Partially Reconfigurable Region)

Περιοχή του FPGA η οποία δεσμεύεται προκειμένου να εφαρμοστεί δυναμική επαναδιαμόρφωση.

Στατική Περιοχή (Static Region)

Το σύνολο των μη δυναμικών περιοχών του FPGA των οποίων η λειτουργικότητα παραμένει σταθερή καθ' όλη τη διάρκεια της λειτουργίας του FPGA.

Δυναμικό Τμήμα (Partially Reconfigurable Module)

Μια από τις πολλαπλές διαμορφώσεις υλικού που μπορεί να εφαρμοσθούν σε μια δυναμική περιοχή.

Στην παρούσα εργασία μας απασχόλησε η δυναμική αυτό-επαναδιαμόρφωση (dynamic partial selfreconfiguration) ή απλούστερα Δυναμική Επαναδιαμόρφωση (ΔΕ).

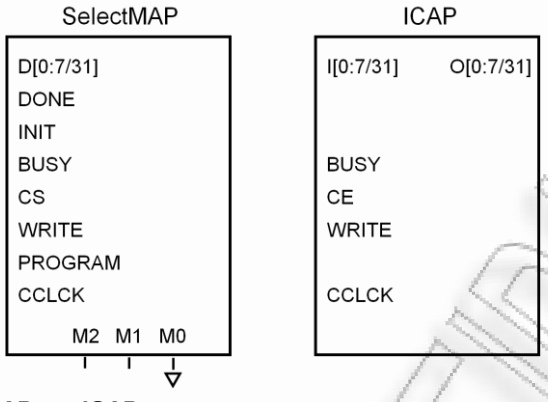
3.3 Υλοποίηση της Δυναμικής Επαναδιαμόρφωσης

Προκειμένου να μπορεί να υλοποιηθεί η ΔΕ πρέπει να υποστηρίζεται από το υλικό του FPGA, τόσο από πλευράς διαθέσιμων πόρων όσο και από πλευράς αρχιτεκτονικής σχεδίασης. Στη συνέχεια περιγράφονται οι σχεδιαστικές λύσεις που προσφέρονται από τα Xilinx FPGAs για να γίνει εφικτή η ΔΕ.

Θύρα ICAP (Internal Configuration Access Port)

Για να υποστηριχθεί η λειτουργικότητα της ΔΕ, στα Xilinx FPGA (ξεκινώντας από τα Virtex-II) υπάρχει ένας μηχανισμός πρόσβασης στη μνήμη διαμόρφωσης, το ICAP (Internal Configuration Access Port). Αυτό ελέγχεται εσωτερικά από το FPGA και χειρίζεται δεδομένα διαμόρφωσης χωρισμένα σε *frames*, που είναι η μικρότερη μονάδα κατάτμησης των δεδομένων διαμόρφωσης. Το ICAP ουσιαστικά είναι μια απλούστερη εκδοχή της θύρας SelectMap η οποία χρησιμοποιείται για τη διαμόρφωση του FPGA, όπως φαίνεται στο Σχήμα 7.

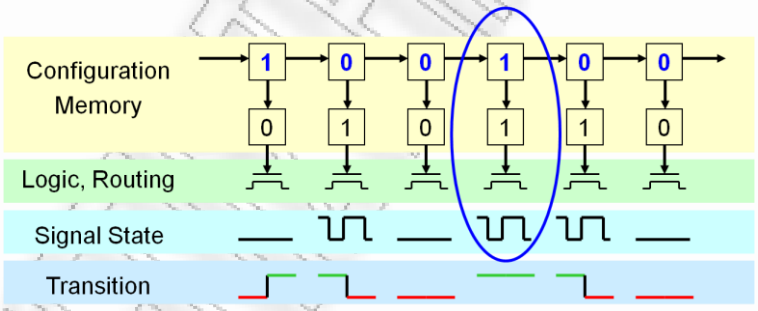
Η χρήση του ICAP είναι δυνατή μόνο όταν ολοκληρωθεί η αρχική διαμόρφωση του FPGA. Μέσω αυτού, δεδομένα μπορεί να φορτωθούν από κάποια εξωτερική μνήμη απ' ευθείας στη μνήμη διαμόρφωσης του FPGA.



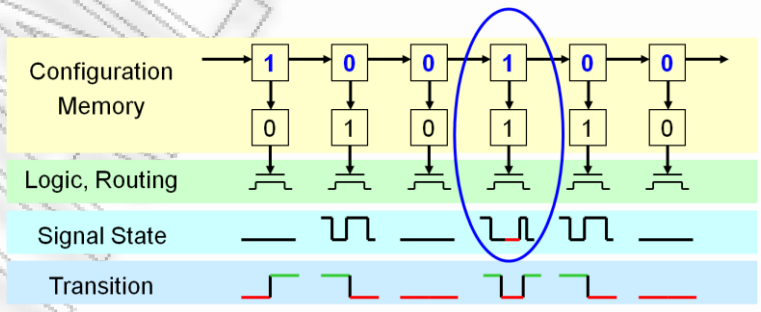
Σχήμα 7: Θύρες SelectMAP και ICAP

Μνήμη Διαμόρφωσης χωρίς αιχμές

Ένα άλλο σημαντικό χαρακτηριστικό της αρχιτεκτονικής των FPGA της σειράς Virtex-II και νεωτέρων είναι η δυνατότητα για προγραμματισμό της μνήμης διαμόρφωσης χωρίς αιχμές (*Glitch-less Memory Cells Configuration*). Αυτό σημαίνει ότι όταν η μνήμη διαμόρφωσης επανεγγράφεται με ίδια δεδομένα τα στοιχεία που διαμορφώνονται κρατούν σταθερή την κατάστασή τους χωρίς να δημιουργούνται αιχμές από τυχαίες μεταβάσεις. Το χαρακτηριστικό αυτό επιτρέπει τη δημιουργία δυναμικών περιοχών οι οποίες να μην εκτείνονται σε όλο το ύψος του FPGA. Στο Σχήμα 8 φαίνεται η συγκεκριμένη λειτουργικότητα των Virtex και νεωτέρων FPGA σε αντιδιαστολή με τη συμπεριφορά των Spartan & Virtex FPGA η οποία απεικονίζεται στο Σχήμα 9. Και στις δύο περιπτώσεις φαίνεται η λειτουργία της μνήμης διαμόρφωσης στην περίπτωση που ελέγχει transistors. Εάν η μνήμη είναι '1' το τρανζίστορ άγει, ενώ στην αντίθετη περίπτωση αποκόπτει. Παρατηρούμε ότι στην περίπτωση του Spartan, μια μετάβαση του bit διαμόρφωσης από '1' σε '1' και σε συνδυασμό με την είσοδο, μπορεί να προκαλέσει μια αιχμή λόγω στιγμιαίας μετάβασης στο '0'.



Σχήμα 8: Λειτουργία της Μνήμης Διαμόρφωσης χωρίς αιχμές για τα Virtex-II FPGAs



Σχήμα 9: Λειτουργία της Μνήμης Διαμόρφωσης με αιχμές για τα Spartan FPGAs

Bus Macros

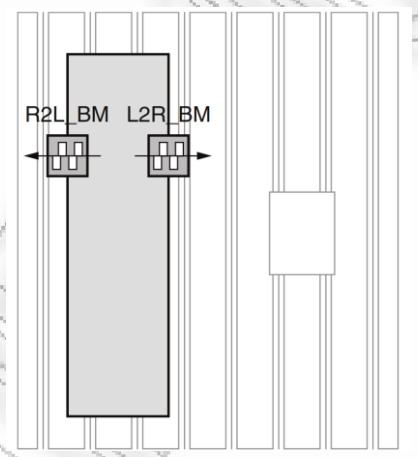
Τα bus macros χρησιμοποιούνται για τη διασύνδεση ανάμεσα στις στατικές και τις δυναμικές περιοχές ενός FPGA. Πρόκειται για hard-macros, δηλαδή εσωτερικά προ-τοποθετημένα υλικά (components) με καθορισμένες διασυνδέσεις (routed). Μπορούν να ορισθούν είτε στο ανώτερο επίπεδο ιεραρχίας είτε σε χαμηλότερο επίπεδο. Η τοποθέτησή τους γίνεται στο όριο μεταξύ στατικής και δυναμικής περιοχής έτσι ώστε το μισό τμήμα τους να είναι σε στατική περιοχή και το άλλο μισό σε δυναμική. Υπάρχουν διάφορες κατηγορίες bus macros τα οποία διαφοροποιούνται ανάλογα με:

- την κατεύθυνση προς την οποία επιτρέπουν την διέλευση του σήματος,
- το πλάτος τους (φυσικές διαστάσεις),
- το αν χρησιμοποιούν καταχωρητές ή όχι.

Ανάλογα με την κατεύθυνση διέλευσης του σήματος, τα bus macros διαχωρίζονται σε:

- ✓ *Αριστερα-προς-Δεξιά (l2r)*
- ✓ *Δεξιά-προς-Αριστερα (r2l)*
- ✓ *Πάνω-προς-Κάτω (t2b)*
- ✓ *Κάτω-προς-Πάνω (b2t)*

Έτσι σήματα που εισέρχονται από την αριστερή πλευρά μιας δυναμικής περιοχής απαιτούν ένα l2r bus macro ενώ, αντίστοιχα, είσοδοι που εισέρχονται από τη δεξιά πλευρά της δυναμικής περιοχής απαιτούν ένα r2l bus macro. Για τα Virtex-II FPGAs δεν ορίζονται t2b και b2t bus macros αφού σε αυτά τα FPGA οι δυναμικές περιοχές ορίζονται μόνο σε ολόκληρες στήλες. Το Σχήμα 10 δείχνει τη χρήση bus macros στην περίπτωση εξόδων μιας δυναμικής περιοχής.

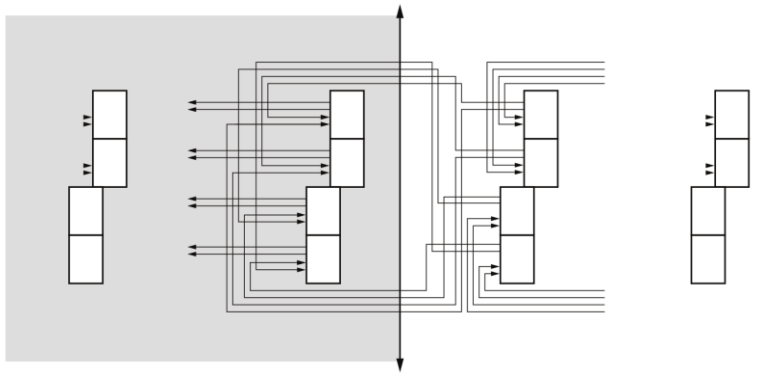


Σχήμα 10: Bus Macros στις εξόδους Δυναμικής Περιοχής

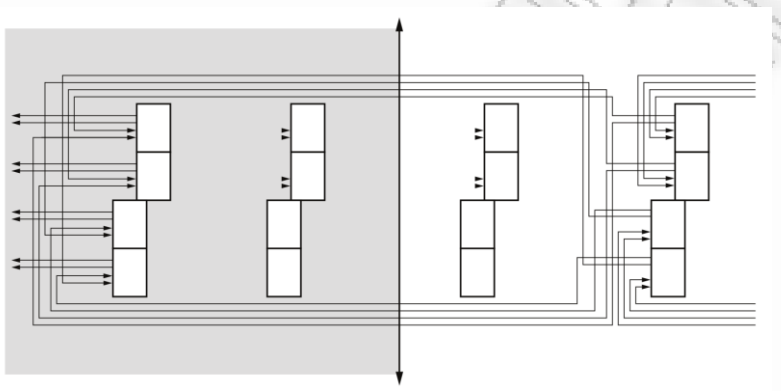
Ανάλογα με το πλάτος τους, τα bus macros διαχωρίζονται σε:

- ✓ *στενά (narrow)*
- ✓ *πλατιά (wide)*

Τα στενά συνδέουν διαδοχικά CLBs ενώ στα φαρδιά μεσολαβούν δύο ασύνδετα CLBs. Στο Σχήμα 11 παρουσιάζονται τα στενά bus macros ενώ στο Σχήμα 12 τα πλατιά.

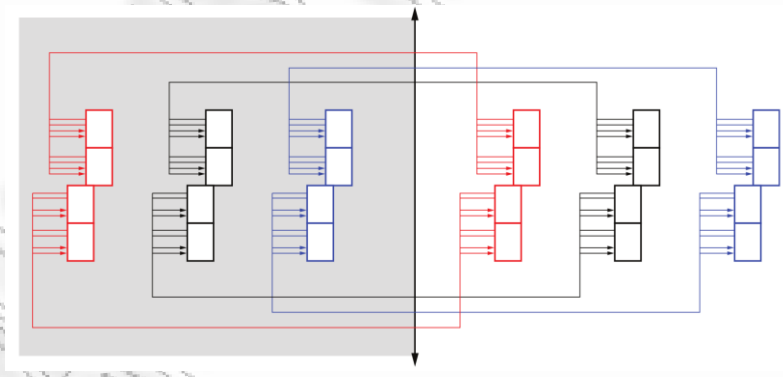


Σχήμα 11: Στενό (narrow) R2L Bus Macro



Σχήμα 12: Πλατύ (wide) R2L Bus Macro

Η βασική χρησιμότητα των wide bus macros είναι για να δημιουργούν σημεία απ' όπου μπορούν διέλθουν πολλά σήματα από και προς τις δυναμικές περιοχές. Έτσι τα wide bus macros μπορούν να αλληλεπικαλυφθούν, όπως φαίνεται στο Σχήμα 13 προκειμένου να χρησιμοποιηθεί μια μικρή περιοχή για να περάσουν περισσότερα σήματα.



Σχήμα 13: Τρία επικαλλυπτόμενα wide bus macros

Ανάλογα με το αν χρησιμοποιούν καταχωρητές, τα bus macros διαχωρίζονται σε:

- ✓ *σύγχρονα (synchronous)*
- ✓ *ασύγχρονα (asynchronous)*

Τα σύγχρονα περιλαμβάνουν επιπλέον έναν καταχωρητή και είσοδο ρολογιού ώστε να κρατούν την τιμή των δεδομένων στην είσοδο. Η χρήση τους επιτρέπει καλύτερη απόδοση του συνολικού κυκλώματος σε ότι αφορά την ταχύτητα.

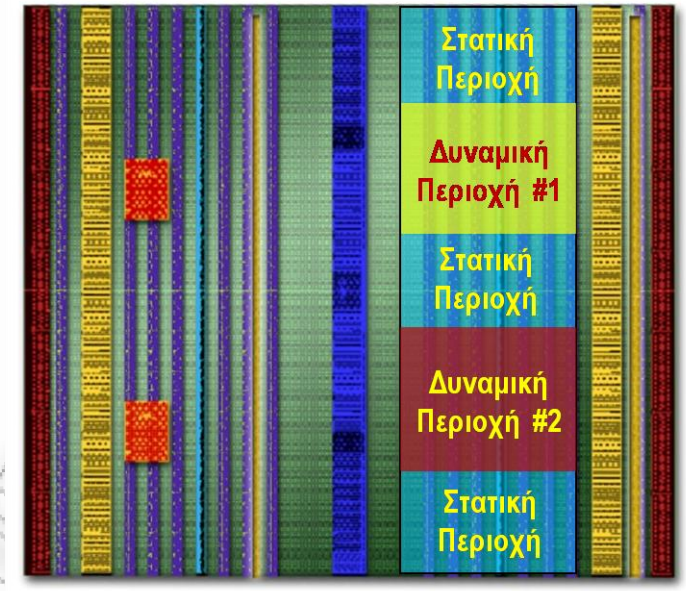
Δεν υπάρχει συγκεκριμένη μεθοδολογία σχετικά με το βέλτιστο σημείο τοποθέτησης των Bus Macros στην επιφάνεια του FPGA τα οποία, όπως περιγράφεται στην παράγραφο 4.6.4, τοποθετούνται

χειροκίνητα. Υπάρχουν κάποιες «συμβουλές» που προτείνονται από τη Xilinx όπως τοποθέτησή των εισόδων στη μια πλευρά της δυναμικής περιοχής και των εξόδων στην άλλη, ή ομαδοποίηση των σημάτων σε σχέση με το δίαυλο/τμήμα κυκλώματος από το οποίο πηγάζουν ή στο οποίο κατευθύνονται. Παρόλο που φαίνονται εύλογες, έχει αποδειχθεί ότι στην πράξη μάλλον σε χειρότερα αποτελέσματα καταλήγουν σε σχέση με άλλες τοποθετήσεις, ενώ τα βέλτιστα αποτελέσματα δε φαίνεται να ακολουθούν κάποιο συγκεκριμένο μοτίβο τοποθέτησης [9].

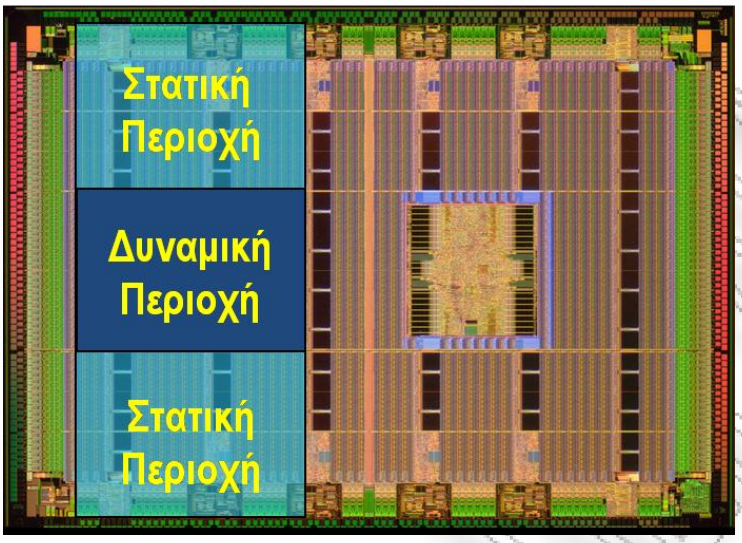
Διάσταση της Διαμόρφωσης

Μια άλλη παράμετρος που λαμβάνεται υπόψη στη ΔΕ είναι η διάσταση της διαμόρφωσης [20]. Ανάλογα με την οικογένεια μπορεί να έχουμε ΔΕ σε *μια διάσταση (1D)* ή *σε δύο διαστάσεις (2D)*. Στην περίπτωση της επαναδιαμόρφωσης μιας διάστασης, η οποία συναντάται στα Virtex-II, δεν μπορούν να τοποθετηθούν δύο δυναμικές περιοχές στην ίδια στήλη. Επίσης δεν μπορούν να υπάρχει ένα δυναμικό κομμάτι και στο υπόλοιπο τμήμα των στηλών που καταλαμβάνει να υπάρχουν στοιχεία μνήμης για το στατικό κομμάτι. Αυτό αποκλείει την συνύπαρξη look-up tables, RAM ή στοιχείων SRL16 του στατικού τμήματος στην ίδια στήλη με δυναμικό τμήμα.

Ο περιορισμός αυτός έχει ξεπεραστεί στα Virtex-4 και Virtex-5 όπου το frame περιέχει στοιχεία διαμόρφωσης για 16 CLBs μιας στήλης (ή 20 CLBs για το Virtex-5) και όχι για ολόκληρη τη στήλη όπως συνέβαινε στα Virtex-II. Στο Σχήμα 14 φαίνεται η δυνατότητα του Virtex-4 για δύο δυναμικές περιοχές σε αντίθεση με τη μια περιοχή που υποστηρίζουν τα Virtex-II (Σχήμα 15). Γι' αυτόν το λόγο τα Virtex-4 & Virtex-5 υποστηρίζουν ΔΕ δύο διαστάσεων μιας και μπορεί να οριστούν σε αυτά διαφορετικές δυναμικές περιοχές η μια πάνω από την άλλη.



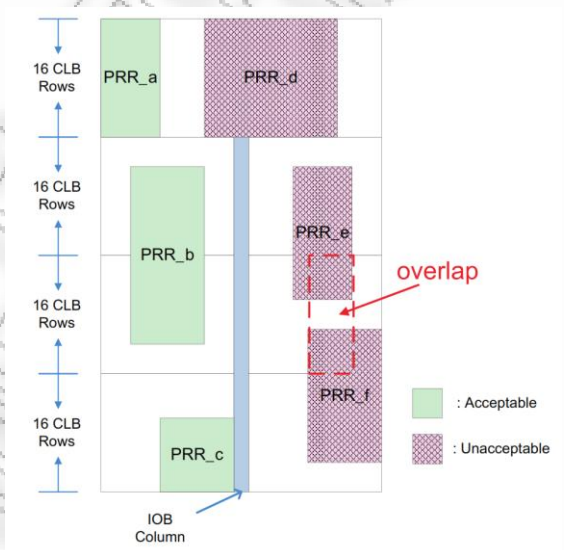
Σχήμα 14: Πολλαπλές Δυναμικές Περιοχές στο Virtex-4



Σχήμα 15: Δυναμική Περιοχή στο Virtex-II

Λόγω του ότι στα Virtex-4 ένα frame διαμόρφωσης περιέχει στοιχεία για 16 CLBs μιας στήλης, η επιφάνεια του FPGA χωρίζεται σε περιοχές που ονομάζονται *σελίδες (pages)*. Μια δυναμική περιοχή μπορεί να καταλαμβάνει μια ή περισσότερες σελίδες. Ωστόσο δεν μπορεί να περιέχει I/O Blocks (τα οποία βρίσκονται στο μέσο του Virtex-4) μιας και για τις δυναμικές περιοχές δεν επιτρέπεται πρόσβαση στους ακροδέκτες του FPGA. Επιπλέον, μια σελίδα μπορεί να περιέχει τμήματα μόνο μιας δυναμικής περιοχής.

Το Σχήμα 16 [20] αναφέρεται σε Virtex-4 FX12 FPGA. Σύμφωνα με τα παραπάνω, οι δυναμικές περιοχές PRR_a, PRR_b και PRR_c είναι σωστά τοποθετημένες ενώ αντίθετα οι PRR_d, PRR_e και PRR_f είναι τοποθετημένες σε μη αποδεκτές θέσεις.

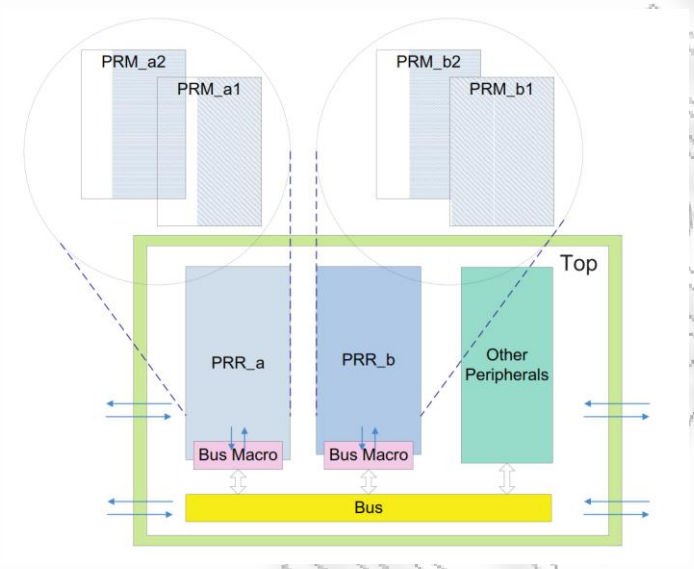


Σχήμα 16: Τοποθέτηση Δυναμικών Περιοχών σε Virtex-4 FX12 FPGA

Η ΔΕ συντονίζεται από έναν ελεγκτή, εξωτερικό ή εσωτερικό στο FPGA. Το ρόλο αυτό μπορεί να τον αναλάβει κάποιος επεξεργαστής ή κάποιο εξειδικευμένο κύκλωμα που θα επιτηρεί τη διαδικασία ελέγχοντας τα bus macros και τη μνήμη όπου βρίσκονται αποθηκευμένα τα αρχεία διαμόρφωσης.

3.4 Δημιουργία Δυναμικά Επαναδιαμορφούμενου Συστήματος

Με βάση τα παραπάνω μπορούμε πλέον να ορίσουμε την αρχιτεκτονική ενός Δυναμικά Επαναδιαμορφούμενου συστήματος σύμφωνα με το διάγραμμα που φαίνεται στο Σχήμα 17.



Σχήμα 17: Αρχιτεκτονική Δυναμικά Επαναδιαμορφούμενου Συστήματος

Προκειμένου να γίνει η υλοποίηση ενός τέτοιου συστήματος θα πρέπει πρώτα να ορισθεί η λειτουργικότητα που μπορεί να αλλάζει δυναμικά μέσα από συναρτήσεις αμοιβαία αποκλειόμενες. Αυτές οι συναρτήσεις μπορούν να ορισθούν σε δυναμικά τμήματα. Κατόπιν πρέπει να οριστεί η θέση της κάθε δυναμικής περιοχής στο FPGA. Έπειτα να οριστεί το είδος και να τοποθετηθούν τα bus macros.

Η υλοποίηση της ΔΕ σε ένα σύστημα επιβάλλει τη λογική του ιεραρχικού σχεδιασμού. Ετσι θα πρέπει να δημιουργηθεί ένα αρχείο που θα περιγράφει το ανώτερο επίπεδο (top-level) του συστήματος. Εκεί θα οριστούν τα σήματα εισόδου/εξόδου, τα ρολόγια, οι ελεγκτές ρολογιού (DCM), καθώς και τα bus macros (αν και υπάρχει τρόπος τα bus macros να ορισθούν και σε χαμηλότερα επίπεδα ιεραρχίας). Στα χαμηλότερα επίπεδα της ιεραρχίας θα ορισθεί η λειτουργικότητα των δυναμικών περιοχών και ενδεχομένως του στατικού τμήματος.

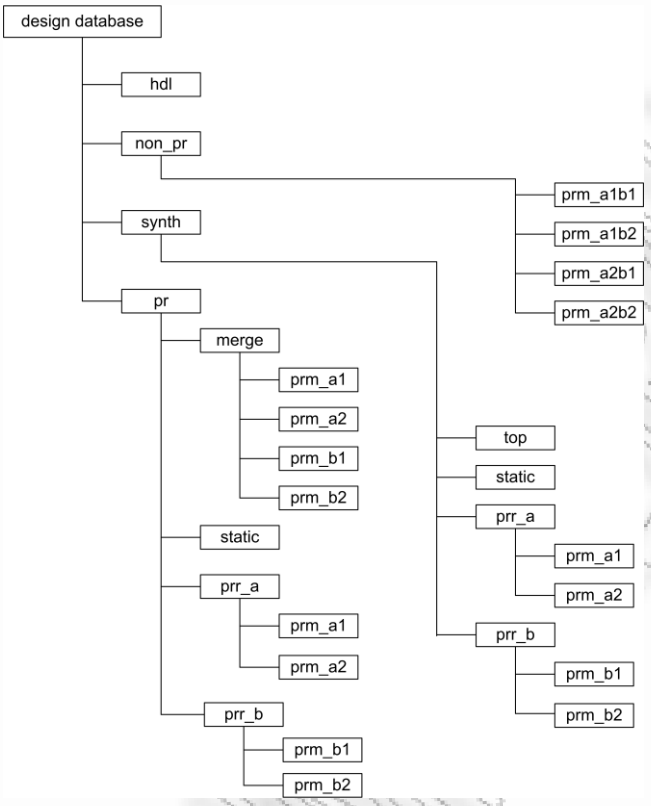
Όταν ορισθεί ολόκληρη η ιεραρχία του συστήματος μπορεί προαιρετικά (και συνιστάται) να γίνει μια στατική υλοποίηση για κάθε πιθανό συνδυασμό στατικών και δυναμικών περιοχών. Αυτό θα επιτρέψει έλεγχο ότι το αντίστοιχο στατικό σύστημα μπορεί να υλοποιηθεί και να λειτουργεί σωστά.

3.5 Early Access Partial Reconfiguration Flow

Η διαδικασία για δημιουργία συστημάτων με επαναδιαμορφούμενη λογική (μέχρι και την έκδοση ISE9.2) ονομάζεται *Early Access Partial Reconfiguration Flow* (EAPR). Η EAPR αντικατέστησε την προηγούμενη μέθοδο, *Modular Design Flow* [21].

Η EAPR εισήγαγε διάφορες βελτιώσεις στην υποστήριξη της ΔΕ. Έτσι πλέον, στα Virtex-II η δυναμική περιοχή δε χρειάζεται να καλύπτει ολόκληρες στήλες του FPGA αλλά μπορεί να είναι μια οποιαδήποτε ορθογώνια περιοχή (βλ. Σχήμα 15). Επίσης τα σήματα διασύνδεσης μιας στατικής περιοχής μπορούν να διέρχονται από μια δυναμική περιοχή χωρίς να μεσολαβούν bus macros, κάτι που βελτιώνει την ταχύτητα του τελικού συστήματος. Επιπλέον η EAPR εισάγει τα bus macros τα οποία πλεονεκτούν σε σχέση με τους τρι-σταθείς απομονωτές (Tri-State Buffers) που χρησιμοποιούσε η Modular Design Flow [22].

Η υλοποίηση ενός συστήματος με χρήση της EAPR διευκολύνεται εάν τηρηθεί μια συγκεκριμένη δομή καταλόγων και υποκαταλόγων όπου θα βρίσκονται τα αρχεία που περιγράφουν το σύστημα (υλοποίηση στατικής και δυναμικής περιοχής, αρχεία περιορισμών, κλπ). Αναφερόμενοι στο σύστημα που παρουσιάζεται στο Σχήμα 17, η προτεινόμενη δομή καταλόγου φαίνεται στο Σχήμα 18:



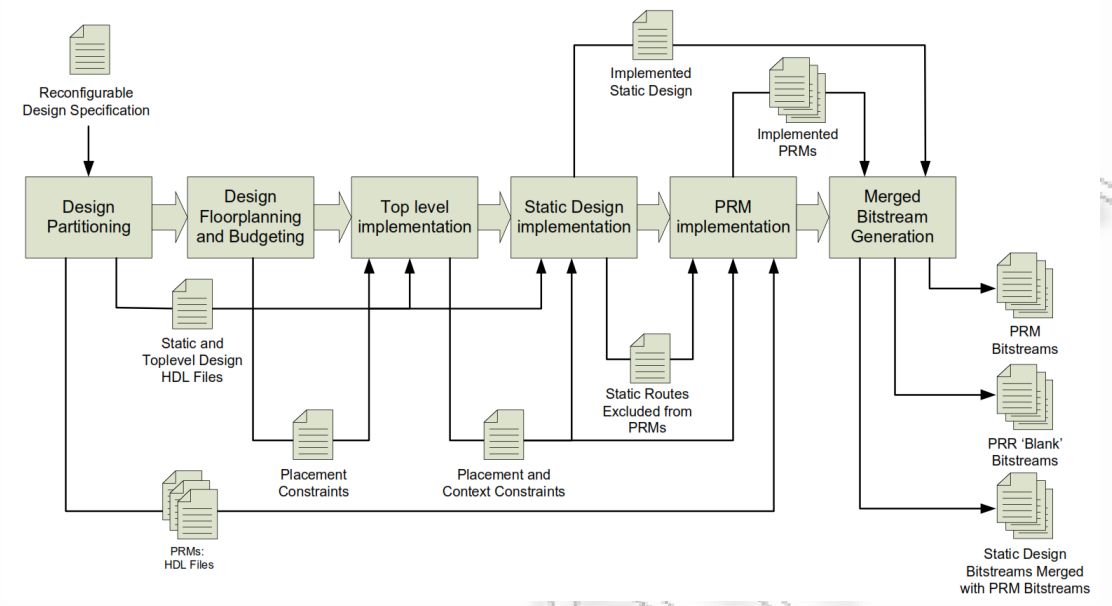
Σχήμα 18: Δομή καταλόγου για υλοποίηση EAPR

Η χρήση των υποκαταλόγων έχει ως εξής:

- Ο υποκατάλογος *hdl* χρησιμοποιείται για την αποθήκευση όλων των αρχείων hdl, περιλαμβανομένων του top-level, των στατικών και των δυναμικών τμημάτων.
- Στον υποκατάλογο *synth*, φυλάσσονται τα αποτελέσματα της σύνθεσης των στατικών και δυναμικών τμημάτων. Στον top βρίσκονται τα αρχεία σύνθεσης του ανωτέρου επιπέδου (top-level). Στον static τα αρχεία για τη σύνθεση του στατικού τμήματος, ενώ στους υποκαταλόγους των *prr_a* και *prr_b* βρίσκονται τα αποτελέσματα της σύνθεσης για τα αντίστοιχα δυναμικά τμήματα.
- Στον υποκατάλογο *non_pr* βρίσκονται τα αποτελέσματα της στατικής υλοποίησης του συστήματος για κάθε πιθανό συνδυασμό δυναμικών τμημάτων με σκοπό την επαλήθευση ότι όλες οι πιθανές διαμορφώσεις του τελικού συστήματος λειτουργούν όπως αναμένεται.
- Στον υποκατάλογο *pr* βρίσκονται τα αρχεία υλοποίησης της δυναμικής επαναδιαμόρφωσης. Τα αρχεία που προκύπτουν από την σύνθεση στατικών και δυναμικών τμημάτων τοποθετούνται στους αντίστοιχους υποκαταλόγους (static, prm_a1, …) ενώ στον υποκατάλογο *merge* αποθηκεύονται τελικά το στατικό και τα μερικά αρχεία διαμόρφωσης που προκύπτουν μετά την εκτέλεση της διαδικασίας υλοποίησης.

Η δομή αυτή μπορεί να προσαρμόζεται κάθε φορά με βάση τις ιδιαιτερότητες του καθενός υλοποιούμενου συστήματος.

Το διάγραμμα ροής της διαδικασίας υλοποίησης με βάση την EAPR φαίνεται στο Σχήμα 19.



Σχήμα 19: Διάγραμμα ροής της διαδικασίας υλοποίησης ΔΕ με βάση την EAPR μέθοδολογία

Στην πρώτη φάση (*Design Partitioning*) ο σχεδιαστής κάνει κατανομή των υλοποιούμενων υποσυστημάτων σε στατικά και δυναμικά. Η υλοποίηση των στατικών τμημάτων ακολουθεί τη λογική σχεδίασης ενός συστήματος χωρίς ΔΕ με μικρές διαφοροποιήσεις. Τα δυναμικά τμήματα (Partially Reconfigurable Modules - PRM) σχετίζονται με λειτουργίες που είναι αμοιβαίως αποκλειόμενες. Κάθε λειτουργία θα υλοποιηθεί σε ένα διαφορετικό δυναμικό τμήμα και όλα αυτά θα εναλλάσσονται εντός μιας δυναμικής περιοχής (Partially Reconfigurable Region - PRR) σε πραγματικό χρόνο. Ο σχεδιαστής θα πρέπει να ορίσει επακριβώς την επικοινωνία ανάμεσα σε μια δυναμική περιοχή και στο στατικό τμήμα. Η επικοινωνία γίνεται διαμέσου των bus macros τα οποία πρέπει να έχουν τον κατάλληλο προσανατολισμό. Κάθε επικοινωνία της δυναμικής περιοχής με τους ακροδέκτες εισόδου/εξόδου του FPGA γίνεται διαμέσου των bus macros. Στο τέλος της πρώτης φάσης, το σύστημα θα πρέπει να έχει οργανωθεί σε μια σειρά αρχείων hdl με μια συγκεκριμένη ιεραρχία, που παρουσιάζεται στο Σχήμα 18. Κατά το στάδιο της σύνθεσης η εισαγωγή στοιχείων IOBUF θα πρέπει να έχει απενεργοποιηθεί για όλα τα τμήματα πέραν του ανωτέρου στην ιεραρχία τμήματος (top module).

Στην επόμενη φάση (*Floorplanning & Budgeting*) γίνεται η αναγνώριση του μεγέθους και της θέσης της κάθε δυναμικής περιοχής. Το PlanAhead μπορεί να βοηθήσει σε αυτήν τη φάση λόγω του γραφικού περιβάλλοντος λειτουργίας του. Κάθε περιοχή (στατική ή δυναμική) αντιστοιχίζεται σε ένα physical block (pblock). Τα pblocks μπορούν να τοποθετηθούν εντός του FPGA. Κατόπιν τοποθετούνται τα bus macros στο όριο μεταξύ στατικής και δυναμικής περιοχής.

Η επόμενη φάση (*Top Level Implementation*) τα εργαλεία υλοποίησης (ngdbuild) δημιουργούν ένα ενδιάμεσο αρχείο που περιγράφει το ανώτερο επίπεδο ιεραρχίας του συστήματος (top module) με την επιπλέον πληροφορία για τη τοποθέτηση των bus macros, ελεγκτών ρολογιού, ακροδεκτών εισόδου/εξόδου και τα όρια στατικών και δυναμικών περιοχών.

Στη συνέχεια (*Static Design Implementation*) υλοποιείται το στατικό τμήμα του FPGA περνώντας από τη διαδικασία του Place-and-Route. Οι διασυνδέσεις που θα δημιουργηθούν μπορεί να διέρχονται από δυναμικές περιοχές προκειμένου να βελτιωθεί η απόδοση του στατικού τμήματος. Αυτές οι διαδρομές δε θα είναι διαθέσιμες για τα αντίστοιχα δυναμικά τμήματα που θα υλοποιηθούν στο επόμενο στάδιο

Στην επόμενη φάση (*PRM Implementation*) υλοποιεί κάθε δυναμικό τμήμα λαμβάνοντας υπόψη τις διασυνδέσεις του στατικού τμήματος που διέρχεται από αυτήν την περιοχή.

Το επόμενο στάδιο (*Merged Bitstream Generation)* είναι το τελικό στάδιο της διαδικασίας, όπου ένα πλήρες λειτουργικό σύστημα δημιουργείται από το στατικό τμήμα και κάθε συνδυασμό δυναμικών τμημάτων. Για κάθε συνδυασμό δημιουργείται το αντίστοιχο αρχείο διαμόρφωσης. Επίσης δημιουργείται

και ένα αρχείο διαμόρφωσης το οποίο δεν περιλαμβάνει κανένα δυναμικό τμήμα, παρά μόνο την υλοποίηση του στατικού τμήματος.

Η διαδικασία ολοκληρώνεται με τον έλεγχο των αρχείων διαμόρφωσης. Αυτός μπορεί να γίνει με φόρτωση στο FPGA του κάθε αρχείου διαμόρφωσης μέσω της JTAG διασύνδεσης από το εργαλείο Impact.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ

ПЕЧАТНО ПЕРА

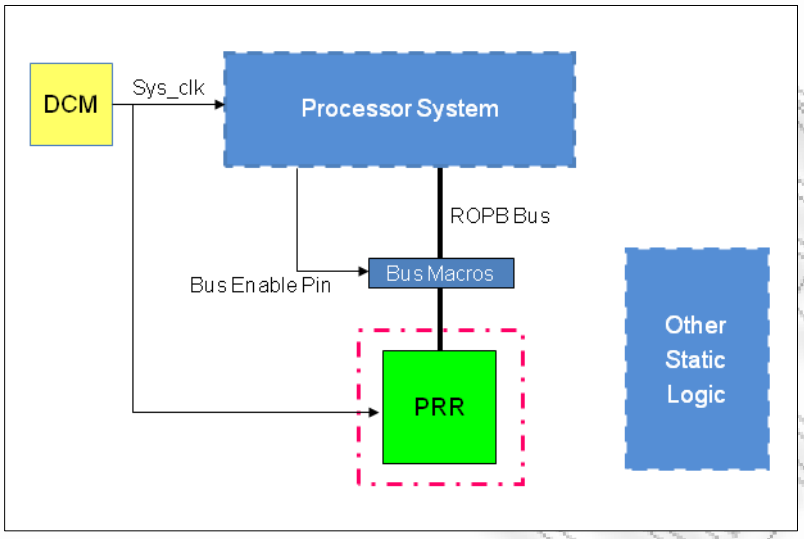
4

Μεθοδολογία Υλοποίησης Δυναμικά Επαναδιαμορφούμενου Συστήματος

Στο κεφάλαιο αυτό περιγράφεται το σύστημα που υλοποιήθηκε στα πλαίσια της παρούσας εργασίας προκειμένου να επιδειχθεί η δυναμική επαναδιαμόρφωση. Συγκεκριμένα παρουσιάζεται το δομικό διάγραμμα της διάταξης, αναλύεται η αρχιτεκτονική του και παρουσιάζονται τα μέρη του κυκλώματος στα οποία εφαρμόζεται η Δ.Ε. Επιπλέον παρουσιάζονται τα βήματα που ακολουθήθηκαν προκειμένου να υλοποιηθεί το σύστημα στην πλατφόρμα ML403 με τη χρήση των εργαλείων της Xilinx.

4.1 Αρχιτεκτονική Συστήματος

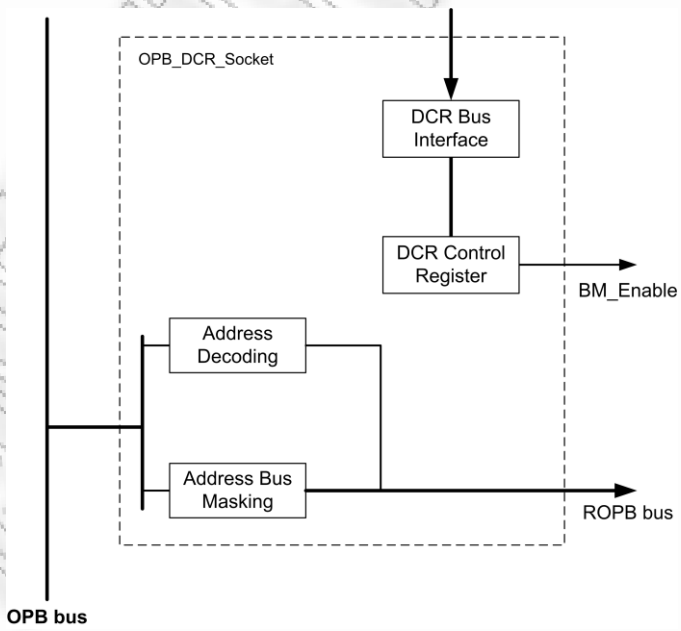
Το σύστημα είναι χτισμένο γύρω από τον PowerPC επεξεργαστή, ο οποίος πέραν των άλλων, συντονίζει και τη διαδικασία της επαναδιαμόρφωσης μέσω φόρτωσης των δυναμικών αρχείων διαμόρφωσης. Σε μια τυπική στατική υλοποίηση με το EDK, το Σύστημα Επεξεργαστή (Processor System) βρίσκεται στο ανώτερο επίπεδο ιεραρχίας της σχεδίασης. Στην περίπτωση της υλοποίησης με ΔΕ όμως, θα πρέπει να υπάρχει ένα επιπλέον ανώτερο επίπεδο στο οποίο το Σύστημα Επεξεργαστή θα διασυνδέεται με τον ελεγκτή ρολογιού και μέσω των bus macros με τη δυναμική περιοχή. Αυτή η αρχιτεκτονική παρουσιάζεται στο Σχήμα 20.



Σχήμα 20: Αρχιτεκτονική πλήρους συστήματος

Στη δυναμική περιοχή τοποθετούνται τα περιφερειακά που εκτελούν τους αλγορίθμους κρυπτογράφησης. Αυτά επικοινωνούν μέσω του OPB διαύλου (και διαμέσου των bus macros) με τον OPB δίαυλο του υποσυστήματος επεξεργαστή. Ακριβέστερα, η διασύνδεση δε γίνεται απ' ευθείας στον OPB δίαυλο, αλλά σε ένα απομονωμένο τμήμα του, το ROPB (Reconfiguration OPB).

Ο ROPB δίαυλος παράγεται από το OPB_DCR_Socket. Πρόκειται για ένα περιφερειακό που τοποθετείται στον OPB δίαυλο και μέσω μιας γέφυρας OPB-to-DCR (OPB2DCR_bridge) αφήνει επιλεκτικά να περάσουν τα σήματα του ROPB διαύλου από και προς τον OPB. Μέσω εντολών προς το OPB2DCR_bridge ο επεξεργαστής μπορεί να ελέγχει το OPB_DCR_Socket. Το λειτουργικό διάγραμμα του OPB_DCR_Socket φαίνεται στο Σχήμα 21.



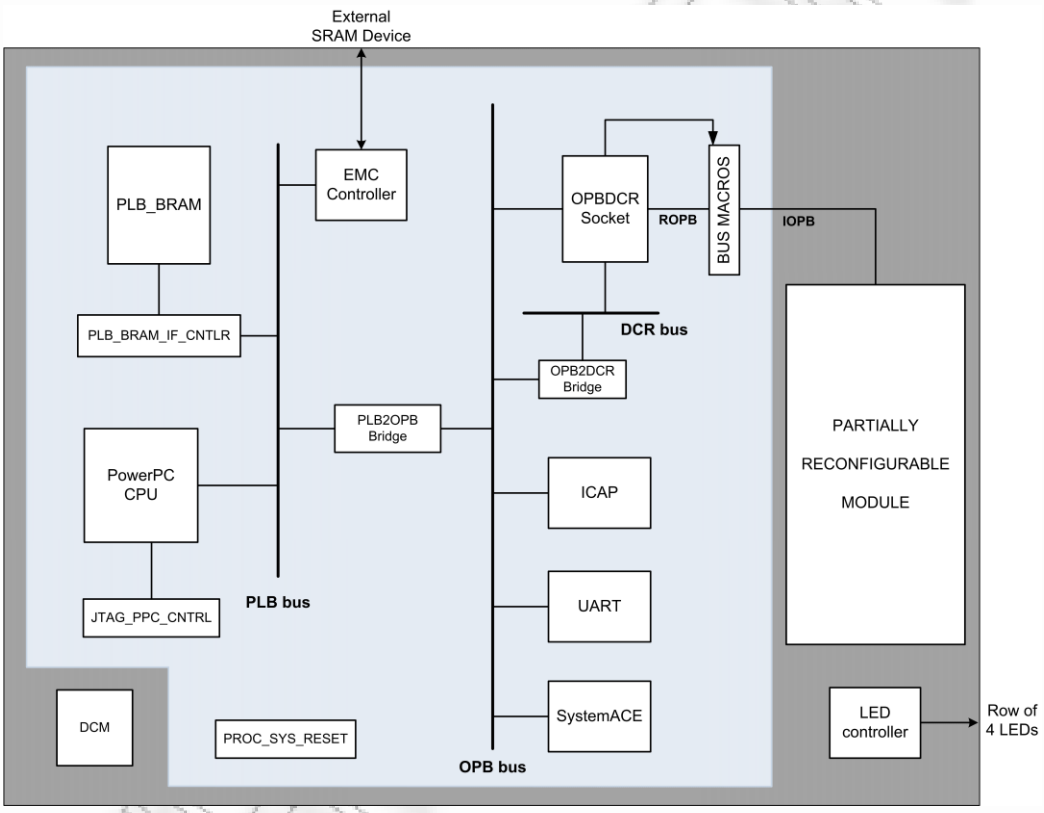
Σχήμα 21: Δομικό διάγραμμα OPB_DCR_Socket

Το υπόλοιπο της αρχιτεκτονικής φαίνεται στο είναι τυπικό για ένα σύστημα που ενσωματώνει PowerPC επεξεργαστή: ο επεξεργαστής χρησιμοποιεί ένα δίαυλο υψηλής ταχύτητας (PLB, Processor Local Bus) προκειμένου να επιτυγχάνεται ταχεία προσπέλαση της εσωτερικής μνήμης. Αυτή υλοποιείται με BlockRAM στοιχεία του FPGA. Μια γέφυρα (PLB-to-OPB bridge), επιτρέπει τη διασύνδεση του PLB

διαύλου με τον OPB δίαυλο στον οποίο βρίσκονται τα πιο «αργά» περιφερειακά, όπως ο SystemACE ελεγκτής, ο ελεγκτής σειριακής θύρας (UART) και ο ελεγκτής ICAP.

Στο Σχήμα 22 φαίνεται η αρχιτεκτονική του συνολικού συστήματος. Σα μνήμη προγράμματος για τον PowerPC χρησιμοποιείται η εξωτερική μνήμη SRAM που υπάρχει στην κάρτα ML403 επειδή οι περιορισμοί για την υλοποίηση της ΔΕ σε συνδυασμό με τις απαιτήσεις του επεξεργαστή δεν επέτρεψαν τη χρήση της εσωτερικής μνήμης BlockRAM του FPGA. Συγκεκριμένα το μέγεθος της δυναμικής περιοχής ήταν τέτοιο ώστε οι μνήμες BlockRAM που δε δεσμεύονταν από αυτή να μην επαρκούν για τις ανάγκες του PowerPC (64KB).

Επιπλέον στη στατική περιοχή υλοποιείται και ένα κύκλωμα ελέγχου των τεσσάρων leds που έχει η κάρτα ML403. Βάσει αυτού, ένα απλό μοτίβο εμφανίζεται στα leds συνεχώς, ανεξάρτητα από τις διαμορφώσεις που είναι φορτωμένες στη δυναμική περιοχή του FPGA.



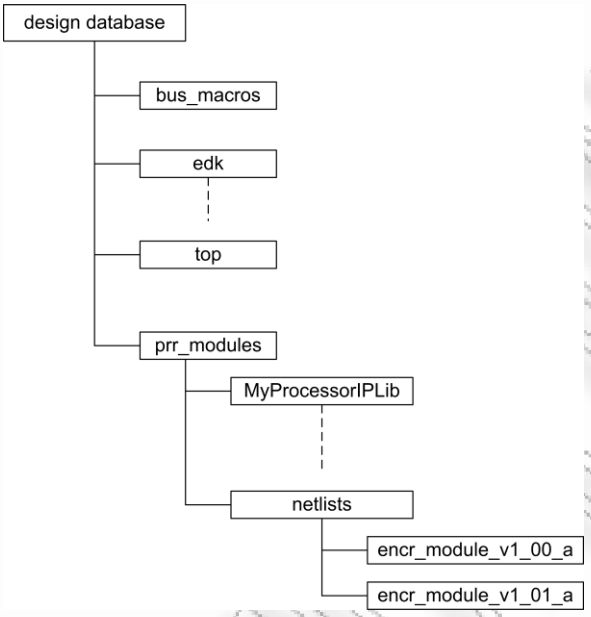
Σχήμα 22: Αρχιτεκτονική Πλήρους Συστήματος

4.2 Ορισμός δομής καταλόγου αρχείων υλοποίησης

Προκειμένου να υλοποιηθεί η διαδικασία EAPR όπως περιγράφεται στην παράγραφο 3.5, δημιουργήθηκε στο δίσκο η ιεραρχία καταλόγων που φαίνεται στο Σχήμα 23 και τοποθετήθηκαν τα διάφορα αρχεία ως εξής:

`bus_macros`	Περιέχει τα χρησιμοποιούμενα στο σύστημα bus macros
`edk`	Περιέχει την υλοποίηση του συστήματος επεξεργαστή όπως δημιουργείται από το EDK
`top`	Περιέχει το ανώτερο επίπεδο της ιεραρχίας καθώς και το στατικό τμήμα που ελέγχει τα leds
`prr_modules`	Περιέχει τα δυναμικά περιφερειακά

Στον υποκατάλογο `netlists` του `prr_modules` τοποθετούνται τα αρχεία που προκύπτουν μετά τη σύνθεση των δυναμικών περιφερειακών, όπως αυτή περιγράφεται στην παράγραφο

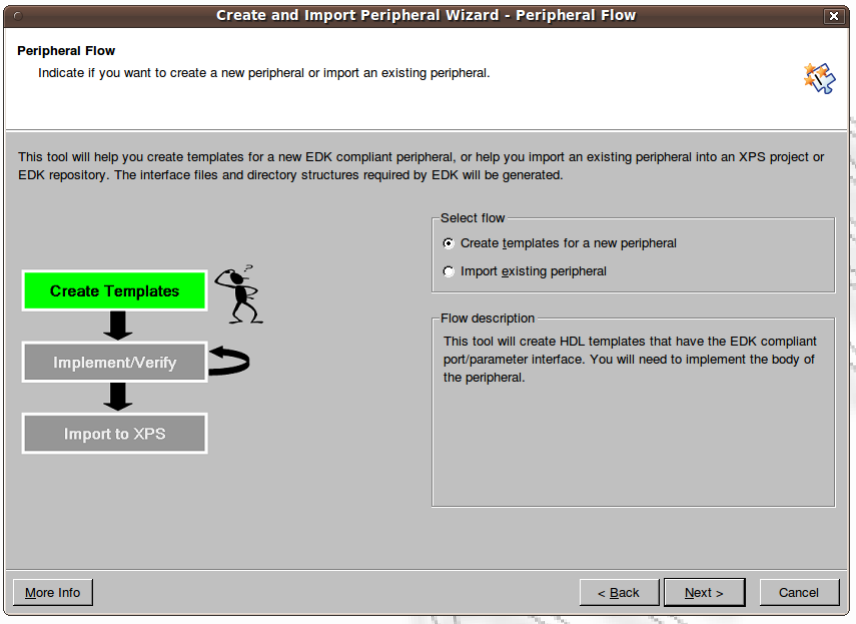


Σχήμα 23: Ιεραρχία Καταλόγων ΔΕ Συστήματος

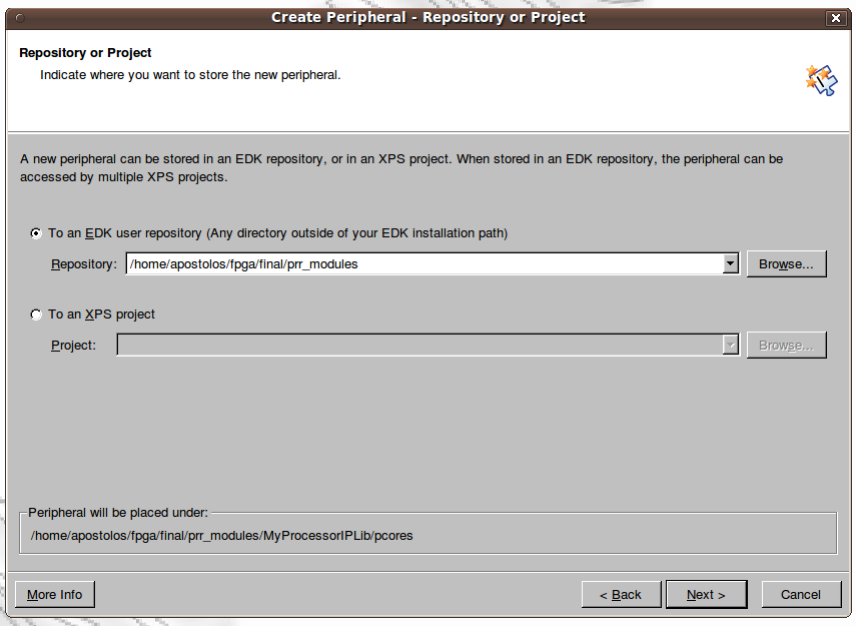
4.3 Δημιουργία Δυναμικών Περιφερειακών

Η δημιουργία των δυναμικών περιφερειακών έγινε με τη χρήση του εργαλείου *Create and Import Peripheral Wizard* το οποίο είναι μέρος του EDK. Η διαδικασία γίνεται σε δύο φάσεις, όπου στην πρώτη δημιουργείται το IPIF (IP interface) το οποίο περιγράφει πώς διασυνδέεται το περιφερειακό με τον OPB δίαυλο. Με βάση αυτήν την πληροφορία δημιουργείται ένα «κενό» περιφερειακό το οποίο στη δεύτερη φάση ενημερώνεται με τα αρχεία HDL που εκτελούν τη ζητούμενη λειτουργικότητα. Στο τέλος της διαδικασίας δημιουργείται ένα σετ VHDL αρχείων που υλοποιούν το περιφερειακό καθώς επίσης και ένα αρχείο τύπου .ise το οποίο μπορεί να φορτωθεί στο εργαλείο ISE. Αυτό δίνει τη δυνατότητα να κάνουμε εξομοίωση σε ολόκληρο το περιφερειακό καθώς επίσης και να εκτελέσουμε τα βήματα υλοποίησής του ξεχωριστά. Το τελευταίο είναι απαραίτητο για τη διαδικασία δημιουργίας του επαναδιαμορφούμενου περιφερειακού.

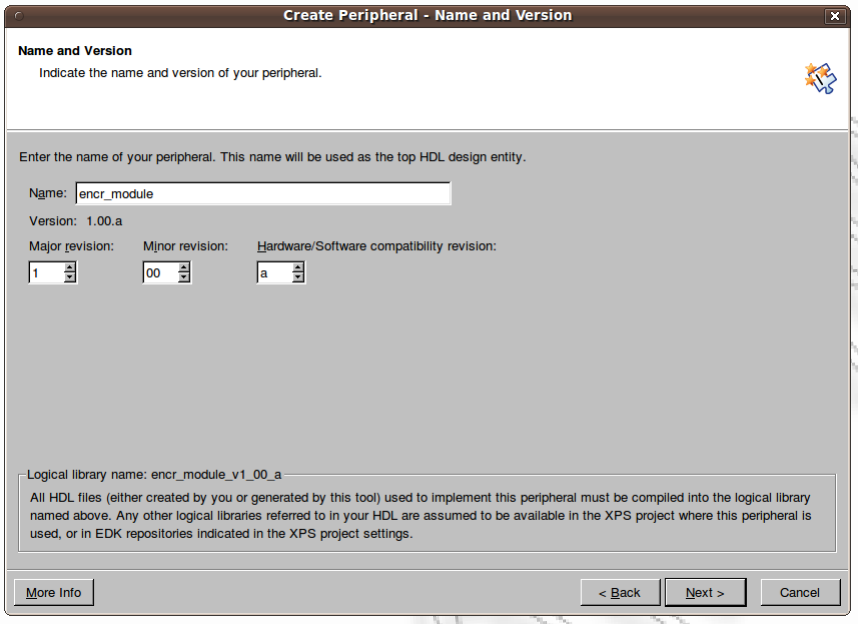
Τα σχήματα που ακολουθούν (Σχήμα 24 έως Σχήμα 32) δείχνουν την πρώτη φάση της διαδικασίας δημιουργίας ενός περιφερειακού. Τα σχήματα ελήφθησαν κατά τη δημιουργία του TripleDES περιφερειακού η οποία περιγράφεται αναλυτικά στην παράγραφο 4.3.1.



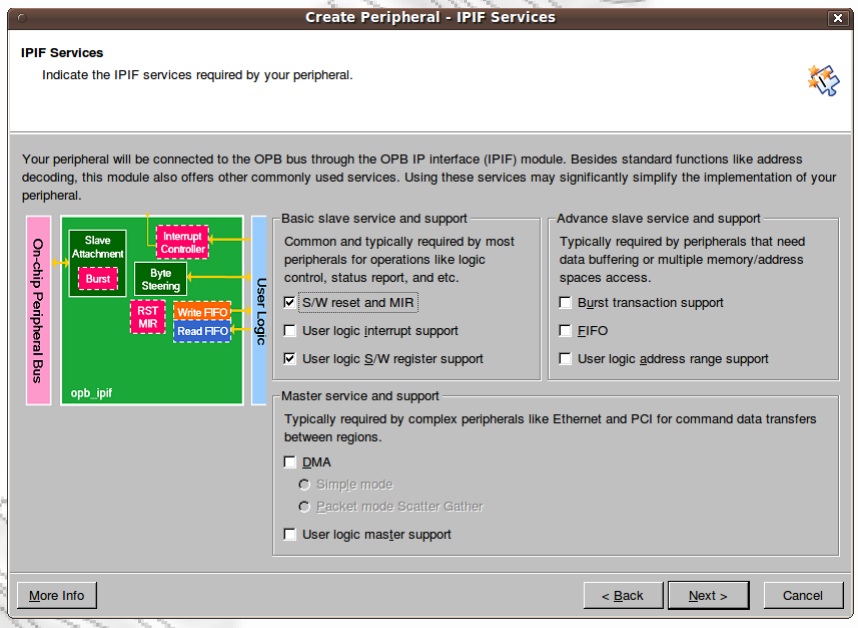
Σχήμα 24: Δημιουργία νέου Περιφερειακού



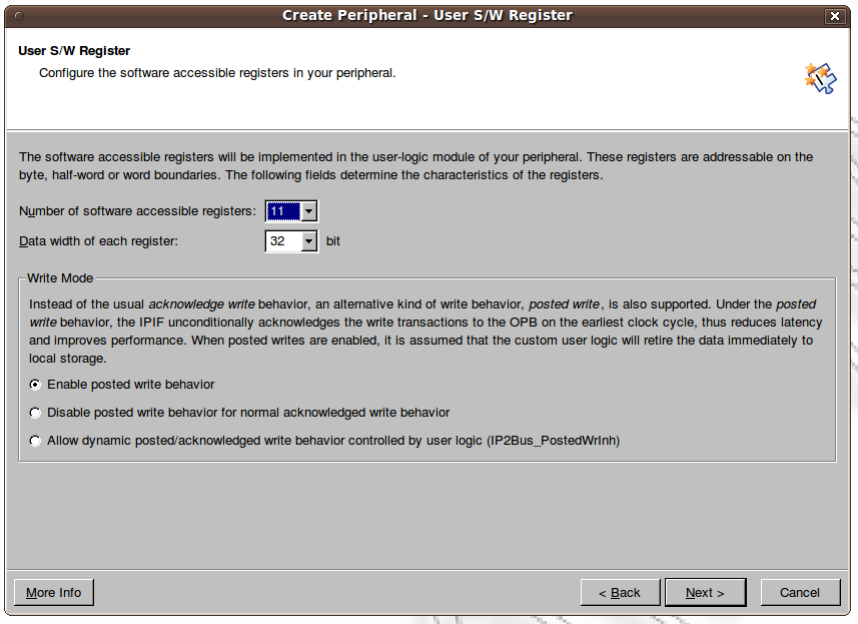
Σχήμα 25: Δήλωση θέσης περιφερειακού



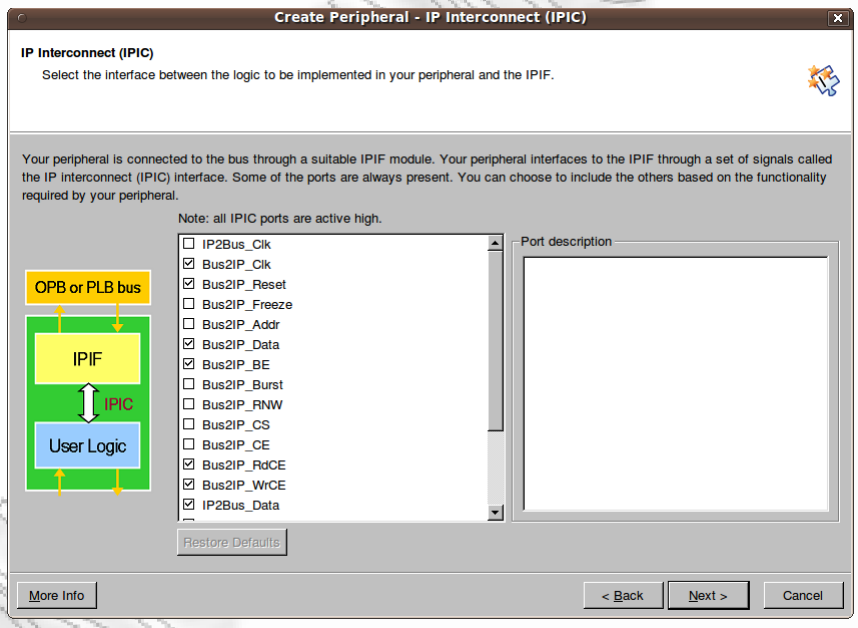
Σχήμα 26: Δήλωση ονομασίας περιφερειακού



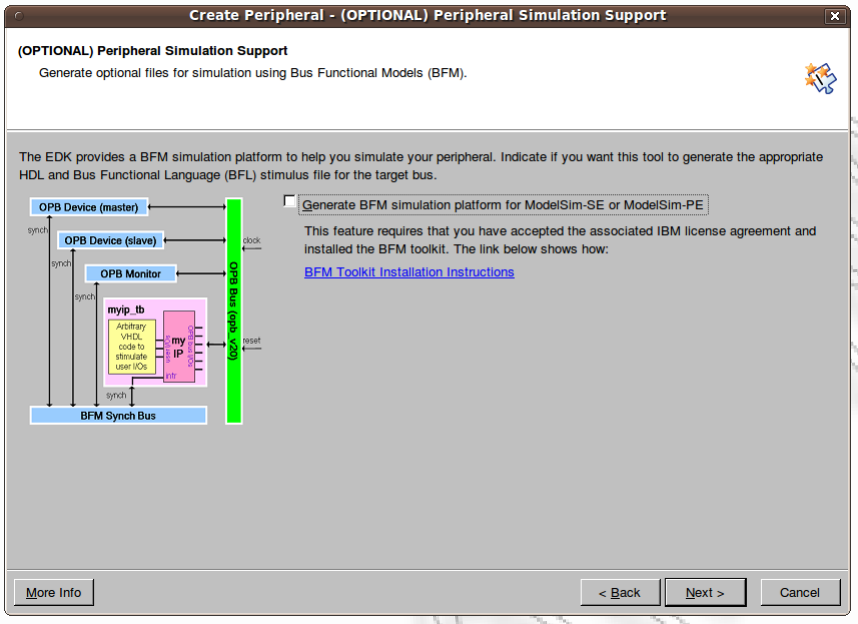
Σχήμα 27: Ορισμός IPIF interface



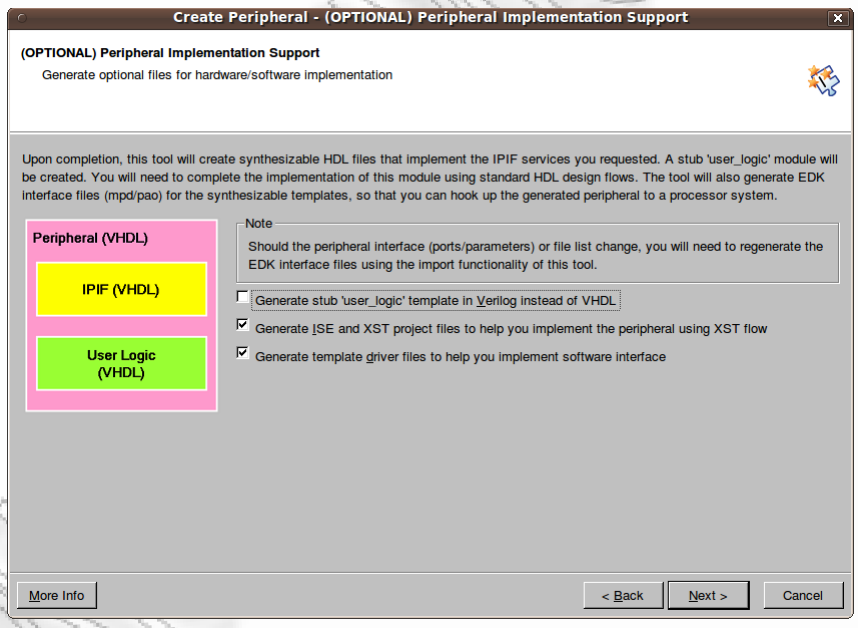
Σχήμα 28: Ορισμός Καταχωρητών



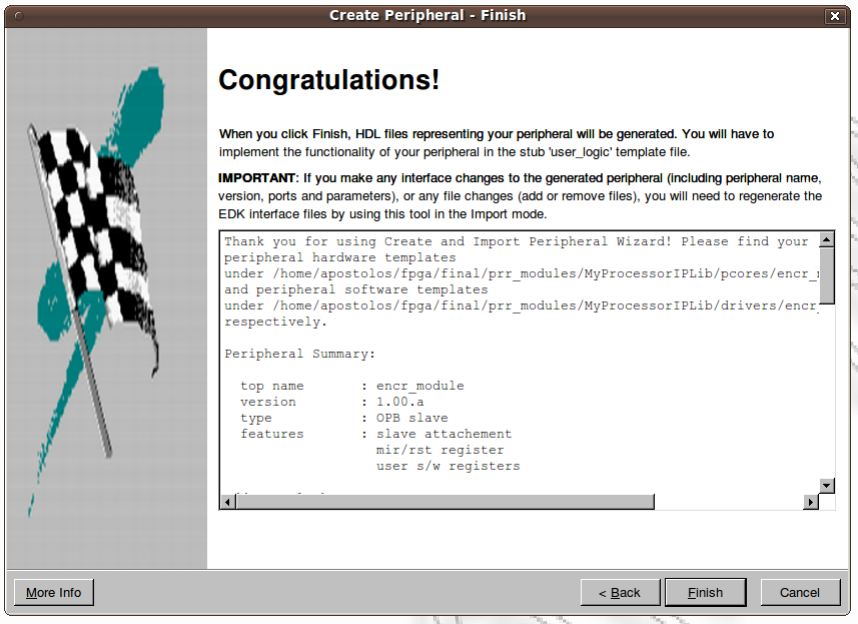
Σχήμα 29: Ορισμός διασύνδεσης με το IPIF Interface



Σχήμα 30: Δημιουργία αρχείων εξομοίωσης

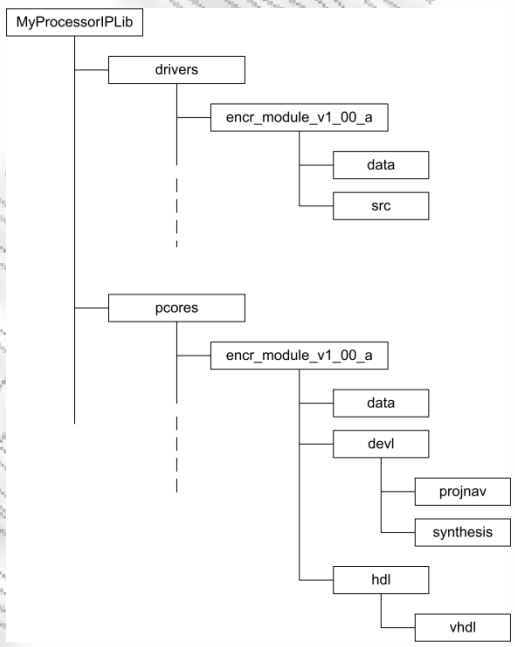


Σχήμα 31: Λοιπές επιλογές



Σχήμα 32: Τέλος διαδικασίας δημιουργίας περιφερειακού

Σε αυτό το σημείο έχει ολοκληρωθεί η πρώτη φάση της υλοποίησης του περιφερειακού και έχει δημιουργηθεί μια ιεραρχική δομή αρχείων κάτω από το σημείο `MyProcessorIPLib/`. Εκεί οι υποκατάλογοι `drivers/` και `pcores/` που περιέχουν δεδομένα για κάθε περιφερειακό που έχει υλοποιηθεί. Η δομή αυτή φαίνεται στο Σχήμα 33.



Σχήμα 33: Δομή αρχείων περιφερειακού

Ο κατάλογος `drivers/` περιέχει τους οδηγούς σε γλώσσα C για την προσπέλαση του περιφερειακού, ενώ ο `pcores/` την υλοποίηση σε vhdl του περιφερειακού.

Εστιάζοντας στην υλοποίηση του περιφερειακού, ο κατάλογος `data/` περιέχει ένα αρχείο με κατάληξη `.pao` (Peripheral Analysis Order) το οποίο περιγράφει τα hdl αρχεία που απαρτίζουν το περιφερειακό. Τα αρχεία αυτά πρέπει να βρίσκονται στον υποκατάλογο `hdl/vhdl/`. Αρχικά στο `.pao` αρχείο

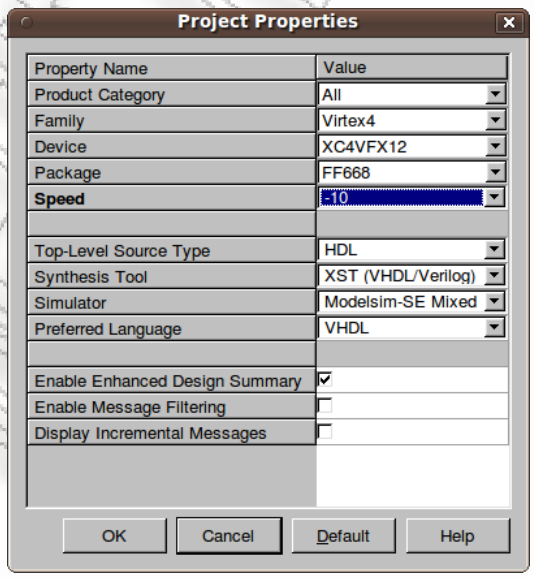
υπάρχουν εγγραφές μόνο για τα αρχεία `encr_module.vhd` και `user_logic.vhd` τα οποία δημιουργεί ο Wizard. Μετά την ολοκλήρωσή του, τα vhdl αρχεία που περιγράφουν το περιφερειακό πρέπει να τοποθετηθούν στον κατάλογο `hdl/vhdl/` και το `.pao` αρχείο να ενημερωθεί κατάλληλα. Στο Σχήμα 34 φαίνεται τμήμα του αρχείου `.pao` στο οποίο έχουν προστεθεί τα αρχεία υλοποίησης του περιφερειακού (παράδειγμα για τον TripleDES).

```
46 lib opb_ipif_v3_01_c write_buffer vhdl
47 lib opb_ipif_v3_01_c opb_bam vhdl
48 lib opb_ipif_v3_01_c opb_ipif vhdl
49 lib encr_module_v1_00_a add_key vhdl
50 lib encr_module_v1_00_a add_left vhdl
51 lib encr_module_v1_00_a block_top vhdl
52 lib encr_module_v1_00_a des_cipher_top vhdl
53 lib encr_module_v1_00_a des_top vhdl
54 lib encr_module_v1_00_a e_expansion_function vhdl
55 lib encr_module_v1_00_a key_schedule vhdl
56 lib encr_module_v1_00_a p_box vhdl
57 lib encr_module_v1_00_a s1_box vhdl
58 lib encr_module_v1_00_a s2_box vhdl
59 lib encr_module_v1_00_a s3_box vhdl
60 lib encr_module_v1_00_a s4_box vhdl
61 lib encr_module_v1_00_a s5_box vhdl
62 lib encr_module_v1_00_a s6_box vhdl
63 lib encr_module_v1_00_a s7_box vhdl
64 lib encr_module_v1_00_a s8_box vhdl
65 lib encr_module_v1_00_a s_box vhdl
66 lib encr_module_v1_00_a tdes_top vhdl
67 lib encr_module_v1_00_a user_logic vhdl
68 lib encr_module_v1_00_a encr_module vhdl
```

Σχήμα 34: Προσθήκη δηλώσεων για τα αρχεία του περιφερειακού στο αρχείο .pao

Κατόπιν πρέπει το αρχείο `hdl/vhdl/user_logic.vhd` να προσαρμοστεί κατάλληλα ώστε να συνδεθεί και να καλεί το περιφερειακό (στην περίπτωση του παραδείγματος στο `tdes_top`).

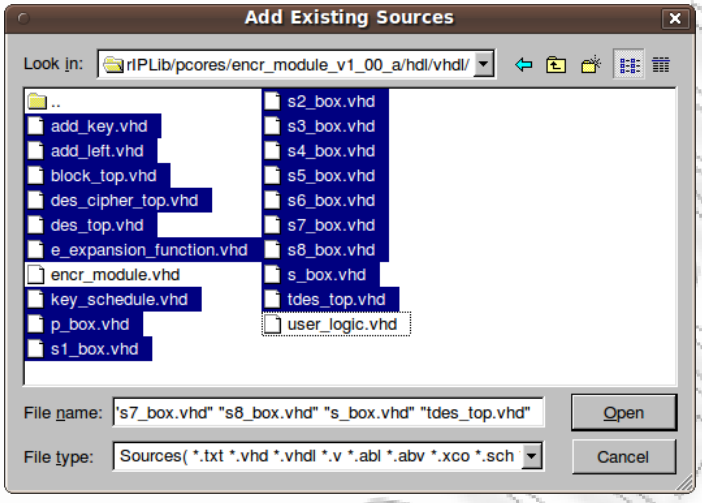
Στον κατάλογο `devl/` υπάρχει μια υλοποίηση ενός project για το ISE. Ανοίγοντας το project στο περιβάλλον του ISE, πρέπει να ρυθμιστούν οι παράμετροι ώστε να αντιστοιχούν στο χρησιμοποιούμενο FPGA, όπως φαίνεται στο Σχήμα 35.



Σχήμα 35: Παράμετροι του χρησιμοποιούμενου FPGA

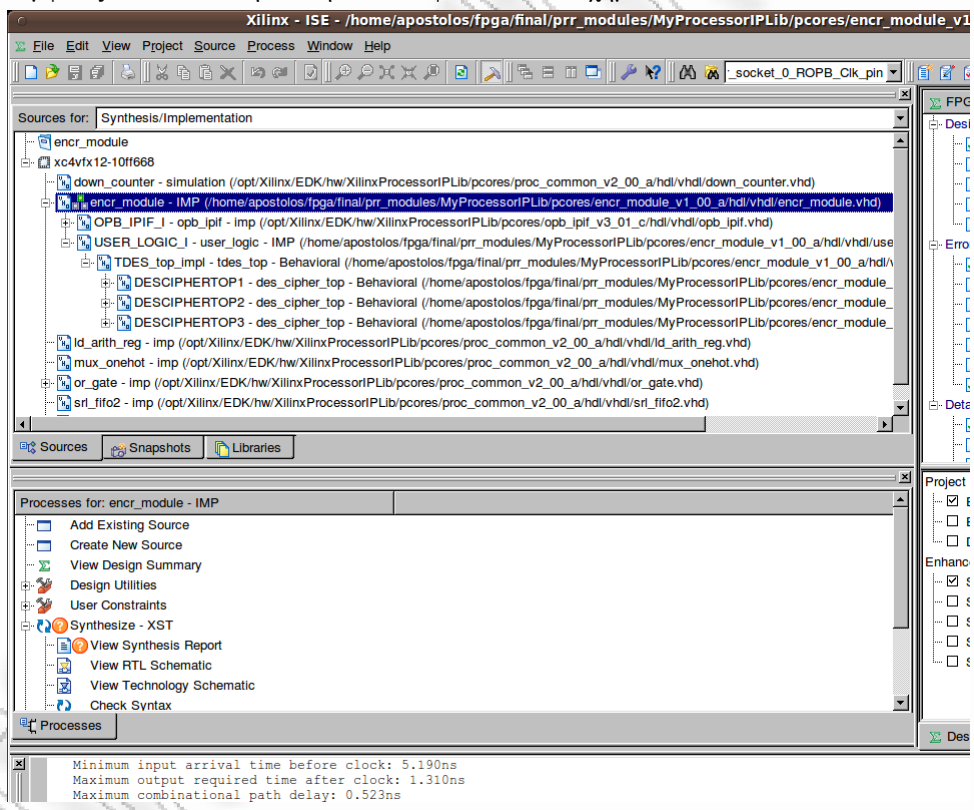
Σαν top-module το ISE αναγνωρίζει το `encr_module`, πρέπει όμως και πάλι να δηλωθούν τα vhdl αρχεία που υλοποιούν το περιφερειακό, μιας και το ISE δε χρησιμοποιεί το `.pao` αρχείο.

Χρησιμοποιώντας την επιλογή *Add Existing Sources* μπορούμε να δηλώσουμε τα αρχεία που περιέχει ο κατάλογος `/hdl/vhdl`, όπως φαίνεται στο Σχήμα 36.



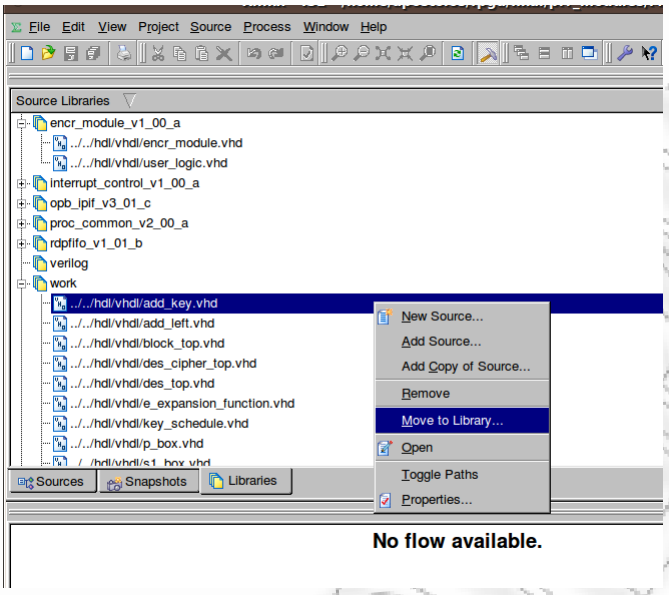
Σχήμα 36: Αρχεία του καταλόγου /hdl/vhdl

Τελικά εμφανίζεται στο ISE η οθόνη του που φαίνεται στο Σχήμα 37.



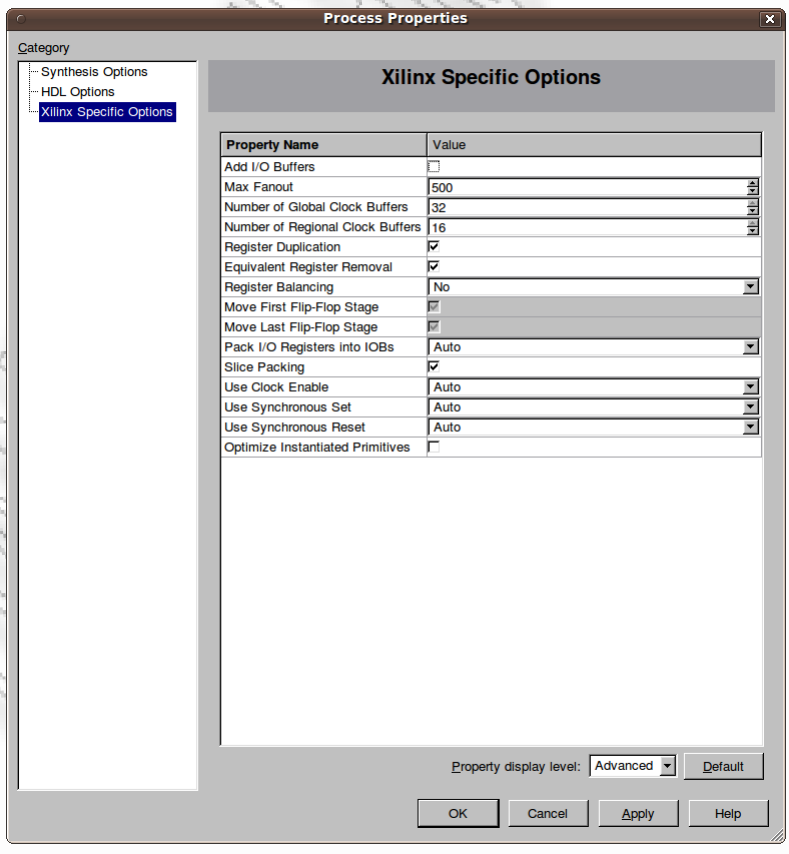
Σχήμα 37: Αρχική οθόνη περιβάλλοντος ISE

Σε αυτό το σημείο πρέπει να προσεχθεί ότι παρόλο που η ιεραρχία του περιφερειακού είναι σωστή, τα αρχεία που το απαρτίζουν βρίσκονται στη βιβλιοθήκη (Library) `work` του ISE και όχι στην `encr_module_v1_00_a` όπου βρίσκονται τα υπόλοιπα αρχεία, κάτι που εμποδίζει τα επόμενα στάδια της υλοποίησης. Προκειμένου να διορθωθεί, επιλέγουμε ξεχωριστά κάθε ένα από τα vhdl αρχεία και εκτελούμε την εντολή «*Move to Library… → encr_module_v1_00_a*», όπως φαίνεται στο Σχήμα 38.



Σχήμα 38: Αλλαγή θέσης βιβλιοθήκης

Μετά από αυτό ακολουθεί το βήμα της σύνθεσης. Επιλέγουμε *"Synthesize"* στο παράθυρο *"Processes"*. Επειδή όμως το design είναι εσωτερικό και δεν συνδέεται απ' ευθείας στις εισόδους/εξόδους του FPGA, πρέπει να μην τοποθετηθούν I/O buffers. Αυτό γίνεται επιλέγοντας στα *"Options"* της σύνθεσης τα *"Xilinx Specific Options"* και απενεργοποιώντας κατόπιν την επιλογή *"Add I/O Buffers"*, όπως φαίνεται στο Σχήμα 39.



Σχήμα 39: Απενεργοποίηση επιλογής "Add I/O Buffers"

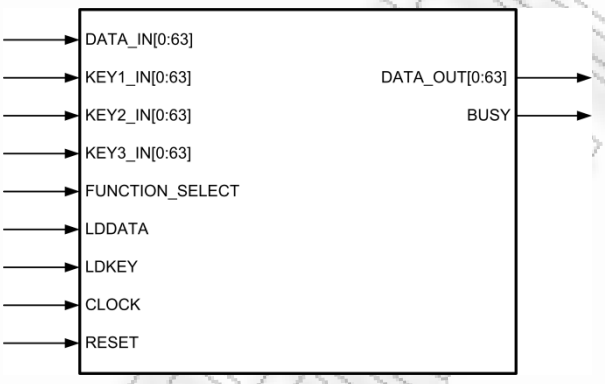
Το αποτέλεσμα της διαδικασίας είναι το αρχείο `encr_module.ngc` το οποίο θα χρησιμοποιηθεί σε επόμενη φάση και για το λόγο αυτό αποθηκεύεται στην αντίστοιχη θέση του υποκαταλόγου `prr_modules/netlists`, σύμφωνα με όσα περιγράφονται στην παράγραφο 4.2.

Η παραπάνω διαδικασία πρέπει να επαναληφθεί τόσες φορές όσα είναι και τα περιφερειακά που θέλουμε να υλοποιήσουμε. Είναι απαραίτητο τα περιφερειακά να έχουν το ίδιο όνομα, αφού μελλοντικά θα τοποθετηθούν στην ίδια θέση. Για να τα ξεχωρίζουμε χρησιμοποιείται ο αριθμός έκδοσης, έτσι ώστε το πρώτο περιφερειακό να ονομάζεται v1_00_a, το δεύτερο v1_01_a κ.ο.κ. (Σχήμα 26).

Πέραν του αρχείου σύνθεσης, σημαντική είναι και η σύνοψη των πόρων που απαιτεί το περιφερειακό και ειδικά των slices. Αυτό φαίνεται στο αρχείο `encr_module.syr` στην ενότητα "Device Utilization Summary".

4.3.1 Δημιουργία του TripleDES περιφερειακού

Για την δημιουργία του TripleDES αλγορίθμου σε OPB περιφερειακό, χρησιμοποιήθηκε η υλοποίηση που είναι διαθέσιμη στο [23]. Το διάγραμμα εισόδων-εξόδων φαίνεται στο Σχήμα 40.



Σχήμα 40: Διάγραμμα Εισόδων-Εξόδων του TripleDES Περιφερειακού

Η είσοδος δεδομένων είναι η DATA_IN[0:63] ενώ τα κλειδιά εισάγονται στις εισόδους KEY1_IN[0:63] KEY2_IN[0:63] και KEY3_IN[0:63]. Η είσοδος FUNCTION_SELECT επιλέγει λειτουργία κρυπτογράφησης ή αποκρυπτογράφησης και εξετάζεται κατά την αρχικοποίηση (όταν η είσοδος RESET είναι ενεργή). Η είσοδος LDDATA δείχνει την παρουσία δεδομένων στην DATA_IN και αντίστοιχα η LDKEY την παρουσία κλειδιών στις εισόδους KEYx_IN. Τα δεδομένα είναι διαθέσιμα στην έξοδο DATA_OUT[0:63] 17 κύκλους ρολογιού CLOCK αφότου φορτωθούν τα δεδομένα στην είσοδο, ενώ το σήμα OUT_READY σηματοδοτεί την παρουσία τους.

Χρησιμοποιώντας το εργαλείο *Create and Import Peripheral Wizard* δημιουργήσαμε ένα περιφερειακό που συνδέεται στον OPB δίαυλο και ενσωματώνει την παραπάνω υλοποίηση του TripleDES αλγορίθμου. Το περιφερειακό ονομάζεται `encr_module_v1_00_a` και είναι προσπελάσιμο μέσω 11 καταχωρητών πλάτους 32 bit. Ο Πίνακας 1 παρουσιάζει τη χρήση των καταχωρητών αυτών.

Καταχωρητές OPB περιφερειακού	Είσοδοι TripleDES Περιφερειακού
`slv_reg1(0:31) & slv_reg0(0:31)`	`data_in(0:63)`
`slv_reg3(0:31) & slv_reg2(0:31)`	`key1_in(0:63)`
`slv_reg5(0:31) & slv_reg4(0:31)`	`key2_in(0:63)`
`slv_reg7(0:31) & slv_reg6(0:31)`	`Key3_in(0:63)`
`slv_reg9(0:31) & slv_reg8(0:31)`	`data_out(0:63)`
`slv_reg10(0:31)`	`Command/Status register`

Πίνακας 1, Καταχωρητές TripleDES Περιφερειακού

Ο Πίνακας 2 παρουσιάζει τον καταχωρητή Command/Status ο οποίος μας επιτρέπει να δίνουμε εντολές προς το περιφερειακό καθώς και να λαμβάνουμε πληροφορίες για την κατάστασή του.

Bit	0	1 : 7	8:15	16:27	28	29	30	31
Τύπος	R/O	-	-	-	R/W	R/W	R/W	R/W
Λειτουργία	status	0...0	0xAB	0...0	reset_strobe	key_strobe	data_strobe	func_sel

Πίνακας 2, Καταχωρητής Command/Status

Όλοι οι καταχωρητές είναι προσπελάσιμοι από τον OPB Master προσθέτοντας στη διεύθυνση βάσης του περιφερειακού (base address) αντίστοιχο εκτόπισμα που αυξάνεται κατά 4 για κάθε καταχωρητή. Έτσι ο slv_reg0 είναι διαθέσιμος στη διεύθυνση (base address + 4 × 0), ο slv_reg1 είναι διαθέσιμος στη διεύθυνση (base address + 4 × 1) κ.ο.κ. Πρέπει να σημειωθεί ότι ενώ στο hardware το λιγότερο σημαντικό bit (bit 0) είναι το αριστερότερο, στο software είναι το δεξιότερο. Έτσι στην εφαρμογή που εκτελείται στον PowerPC το status bit θα βρίσκεται στο bit 31.

Όπως φαίνεται, όλες οι είσοδοι και έξοδοι του περιφερειακού είναι προσπελάσιμες από τον OPB δίαυλο και δε χρειάζεται να γίνουν επιπλέον συνδέσεις με το υποσύστημα επεξεργαστή.

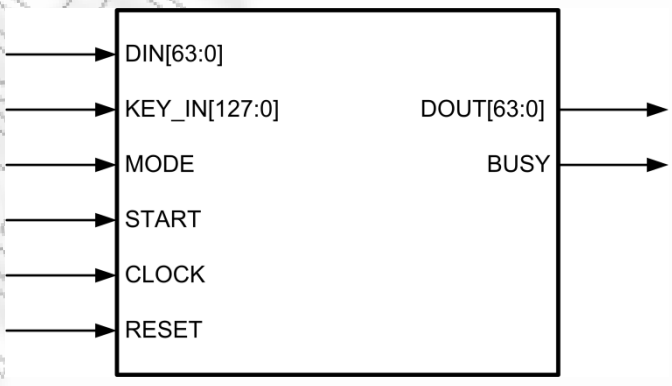
Η έξοδος `BUSY` μπορεί να χρησιμοποιηθεί και σαν έξοδος διακοπής προκειμένου να ειδοποιηθεί ο επεξεργαστής για την ύπαρξη αποτελέσματος στην έξοδο. Ωστόσο κάτι τέτοιο δεν έγινε δεδομένου ότι η όλη διαδικασία κρυπτογράφησης δεδομένων διαρκεί μόλις 17 κύκλους ρολογιού, οπότε μια απλή διαδικασία καθυστέρησης στην πλευρά του λογισμικού είναι αρκετή για να μπορέσει ο επεξεργαστής να πάρει το αποτέλεσμα για τα δεδομένα εισόδου. Στην πράξη δεν είναι απαραίτητη ούτε καν η ρουτίνα καθυστέρησης μιας και ο χρόνος που μεσολαβεί ανάμεσα στην εντολή γραψίματος του καταχωρητή command και στην ανάγνωση των δεδομένων εξόδου από τον καταχωρητή data_out είναι πολύ παραπάνω από 17 κύκλους.

Οι απαιτήσεις του περιφερειακού σε πόρους του FPGA είναι:

```
Number of Slices:                1904  out of   5472    34%
Number of Slice Flip Flops:      1621  out of  10944    14%
Number of 4 input LUTs:          2943  out of  10944    26%
Number of IOs:                    109
Number of bonded IOBs:              0  out of    320     0%
```

4.3.2 Δημιουργία του RTEA περιφερειακού

Για την υλοποίηση του RTEA περιφερειακού χρησιμοποιήθηκε το περιφερειακό που είναι διαθέσιμο στο [24]. Το διάγραμμα εισόδων-εξόδων του φαίνεται στο Σχήμα 41.



Σχήμα 41: Διάγραμμα Εισόδων-Εξόδων του RTEA περιφερειακού

Η είσοδος δεδομένων είναι η DIN[63:0] ενώ του κλειδιού η KEY_IN[127:0]. Η είσοδος MODE επιλέγει λειτουργία κρυπτογράφησης ή αποκρυπτογράφησης και πρέπει να παραμένει σταθερή κατά τη διάρκεια της διαδικασίας κρυπτογράφησης (αποκρυπτογράφησης). Όταν η είσοδος START οδηγηθεί, ξεκινά η

διαδικασία κρυπτογράφησης. Τα δεδομένα είναι διαθέσιμα στην έξοδο DOUT[63:0] 64 κύκλους ρολογιού CLK μετά την έναρξη της διαδικασίας. Η έξοδος BUSY είναι ενεργή (‘1’) για όσο χρονικό διάστημα η μονάδα εκτελεί εργασία κρυπτογράφησης.

Με τη βοήθεια του *Create and Import Peripheral Wizard* δημιουργήσαμε ένα περιφερειακό που συνδέεται στον OPB δίαυλο και ενσωματώνει την παραπάνω υλοποίηση του RTEA αλγορίθμου. Το περιφερειακό ονομάζεται `encr_module_v1_01_a` και είναι προσπελάσιμο μέσω 9 καταχωρητών πλάτους 32 bit. Ο Πίνακας 3 δείχνει τη χρήση των καταχωρητών αυτών.

Καταχωρητές OPB περιφερειακού	Είσοδοι RTEA Περιφερειακού
`slv_reg1(0:31) & slv_reg0(0:31)`	`din(0:63)`
`slv_reg5(0:31) & slv_reg4(0:31) & slv_reg3(0:31) & slv_reg2(0:31)`	`key_in(0:127)`
`slv_reg7(0:31) & slv_reg6(0:31)`	`dout(0:63)`
`slv_reg8(0:31)`	`Command/Status register`

Πίνακας 3, Καταχωρητές RTEA Περιφερειακού

Ο Πίνακας 4 δείχνει τον καταχωρητή Command/Status που μας επιτρέπει να δίνουμε εντολές προς το περιφερειακό και να λαμβάνουμε πληροφορίες για την κατάστασή του.

Bit	**0**	**1 : 7**	**8:23**	**24:29**	**30**	**31**
Τύπος	R/O	-	-	-	R/W	R/W
Λειτουργία	busy	0...0	0x0FEA	0...0	mode	start

Πίνακας 4, Καταχωρητής Command/Status

Και σε αυτήν την περίπτωση, όλες οι είσοδοι και έξοδοι του περιφερειακού είναι προσπελάσιμες από τον OPB δίαυλο και δε χρειάζεται να γίνουν επιπλέον συνδέσεις με το υποσύστημα επεξεργαστή.

Οι απαιτήσεις του περιφερειακού σε πόρους του FPGA είναι:

```
Number of Slices:                        547  out of   5472     9%
Number of Slice Flip Flops:              481  out of  10944     4%
Number of 4 input LUTs:                  841  out of  10944     7%
Number of IOs:                           109
Number of bonded IOBs:                     0  out of    320     0%
```

4.4 Δημιουργία Συστήματος Επεξεργαστή στο περιβάλλον EDK

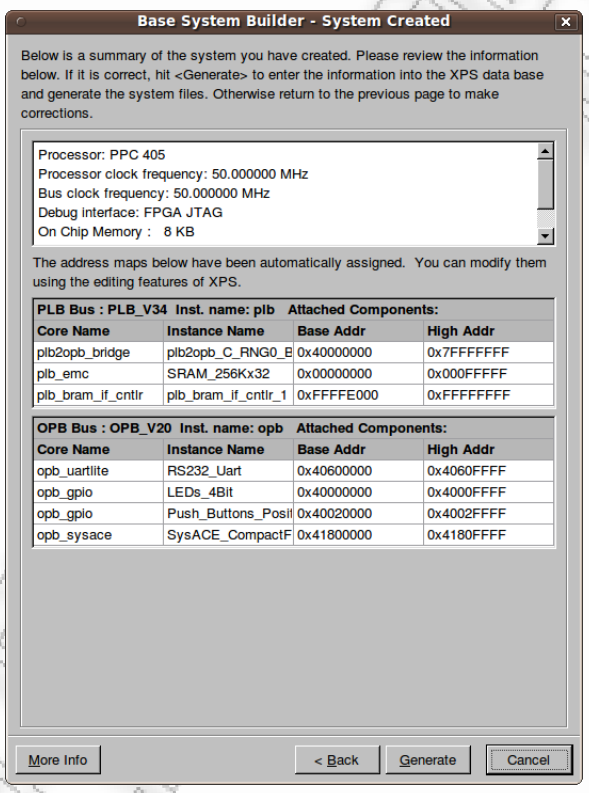
4.4.1 Δημιουργία Αρχικού Συστήματος

Για τη δημιουργία του συστήματος επεξεργαστή στο περιβάλλον του EDK δημιουργούμε ένα νέο project με τη βοήθεια του *Base System Builder*. Εκεί ορίζουμε τις παραμέτρους του συστήματος οι οποίες είναι οι εξής:

Board Vendor	Xilinx
Board name	Virtex-4 ML403 Platform (rev1)
Processor	PowerPC
Reference Clock Frequency	50MHz
Processor Clock Frequency	50MHz

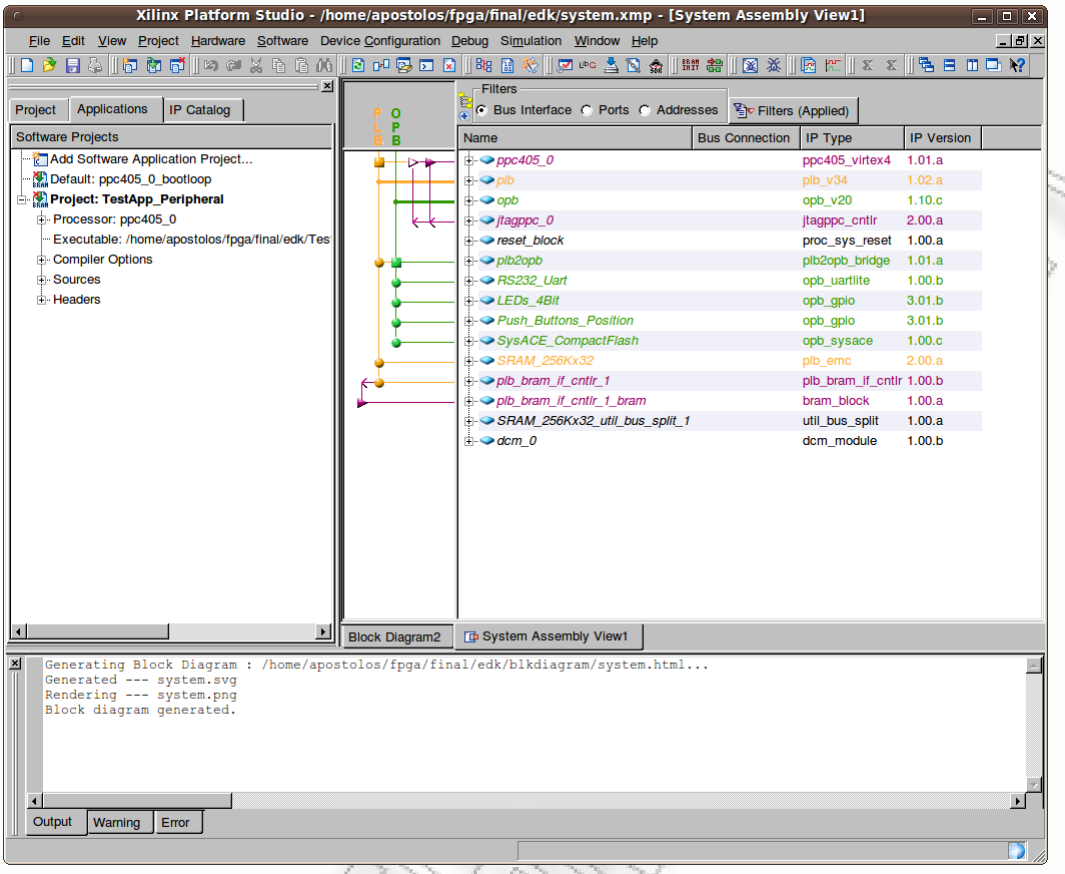
Bus Clock Frequency	50MHz
Debug I/F	FPGA JTAG
OnChip Memory (Program/Data)	None/None
Cache Enable	No
RS232_Uart	OPB UartLite (115200-8N1) - No Interrupt
LEDs_4Bit	OPB GPIO - No Interrupt
Push_Buttons_Position	OPB GPIO - No Interrupt
SysACE_CompactFlash	OPB SYSACE - No Interrupt
SRAM_256Kx32	PLB EMC
plb_bram_if_cntlr_1	8KB

Με την ολοκλήρωση του Base System Builder, παρουσιάζεται η σύνοψη του συστήματος όπως φαίνεται στο Σχήμα 42.



Σχήμα 42: Σύνοψη του συστήματος μετά την ολοκλήρωση του Base System Builder

Το σύστημα που έχει δημιουργηθεί περιέχει έναν ελεγκτή ρολογιού (dcm) όπως φαίνεται στο Σχήμα 43.



Σχήμα 43: Διασύνδεση του συστήματος στο EDK

Επειδή όμως, σύμφωνα με την αρχιτεκτονική που παρουσιάστηκε στην παράγραφο 4.1, το dcm πρέπει να ορίζεται στο ανώτερο επίπεδο ιεραρχίας του συστήματος (top-level), το σβήνουμε επιλέγοντας *"Delete instance and its internal ports"*. Κατόπιν πρέπει να επέμβουμε στο αρχείο διαμόρφωσης του συστήματος (`system.mhs`) ώστε να ορίσουμε τις εισόδους που σχετίζονταν με το dcm που πλέον έχει καταργηθεί. Έτσι:

- η θύρα συστήματος `dcm_clk_s` μετονομάζεται σε `sys_clk_s`
- ορίζεται μια νέα είσοδος, dcm_0_lock_pin η οποία θα συνδεθεί με το σήμα `lock` του dcm από το ανώτερο επίπεδο ιεραρχίας. Αυτό γίνεται με τη δήλωση:

```
PORT dcm_0_lock_pin = dcm_0_lock, DIR = I
```

- Στο component proc_sys_reset προσθέτουμε την είσοδο `dcm_locked` η οποία καταργήθηκε όταν σβήσαμε το dcm, κάτι που γίνεται με τη δήλωση:

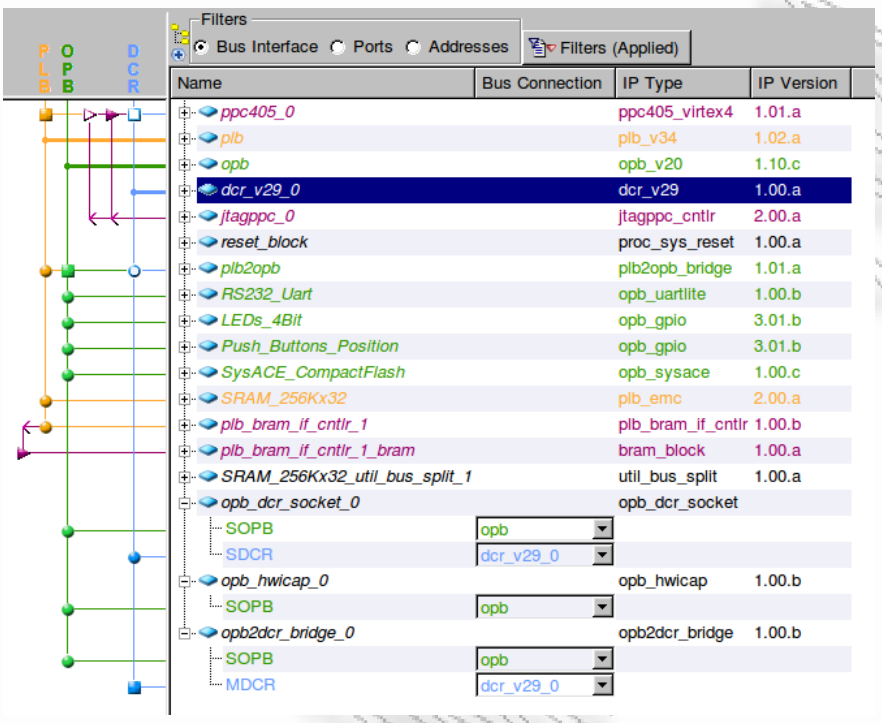
```
PORT dcm_locked = dcm_0_lock
```

Αφού ολοκληρωθούν οι αλλαγές στο `system.mhs` αρχείο μπορούν να οριστούν οι υπόλοιπες δομικές μονάδες του συστήματος. Νωρίτερα όμως θα πρέπει να βεβαιωθούμε ότι το EDK προβάλλει όλες τις υλοποιήσεις των IP cores οι οποίες είναι διαθέσιμες. Αυτό ορίζεται από την επιλογή *Edit → Preferences*. Στην κατηγορία "IP Catalog and IP Config Dialog" θα πρέπει στην κατηγορία "IP Catalog" να είναι ενεργοποιημένες οι επιλογές: *Display "Depricated" IP Cores in IP Catalog* καθώς και *Display "Early Access" IP Cores in IP Catalog*.

4.4.2 Προσθήκη IP Cores

Για να γίνει η υλοποίηση του συστήματος απαιτείται η χρήση του IP `opb_dcr_socket` το οποίο θα πρέπει να τοποθετηθεί στον κατάλογο `edk/pcores`. Κατόπιν με χρήση της επιλογής Project→Rescan User Repositories, το opb_dcr_socket γίνεται διαθέσιμο για χρήση στον κατάλογο της βιβλιοθήκης IP.

Μετά από αυτό γίνεται η υλοποίηση του συστήματος το οποίο θα χρησιμοποιεί το opb_dcr_socket, το opb_hwicap (v1.00.b), το dcr_v29 (v1.00.a), και το opb_to_dcr_bridge (v1.00.b). Αυτά πρέπει να διασυνδεθούν με το υπόλοιπο σύστημα όπως φαίνεται στο Σχήμα 44.



Σχήμα 44: Διασύνδεση περιφερειακών στους διαύλους του συστήματος

Μετά τον ορισμό των διασυνδέσεων, σειρά έχει ο ορισμός των διευθύνσεων. Από την επιλογή *Filters→Addresses* ορίζουμε 64KB χώρο για τα opb_hwicap και opb_dcr_socket περιφερειακά. Από την πλευρά του DCR, το opb_dcr_socket πρέπει να έχει 16 bytes χώρο διευθύνσεων ενώ το opb2dcr_bridge θα πρέπει να έχει 4KB χώρο. Η επιλογή *"Generate Addresses"* δημιουργεί το χώρο διευθύνσεων για τα περιφερειακά του συστήματος.

Προκειμένου να ορίσουμε τις θύρες εισόδου/εξόδου του συστήματος μπορούμε να χρησιμοποιήσουμε την επιλογή Filters→Ports ή εναλλακτικά να τις ορίσουμε απ' ευθείας στο αρχείο `system.mhs`. Σε αυτήν την περίπτωση θα πρέπει να προστεθούν οι παρακάτω δηλώσεις για τις επιπλέον θύρες:

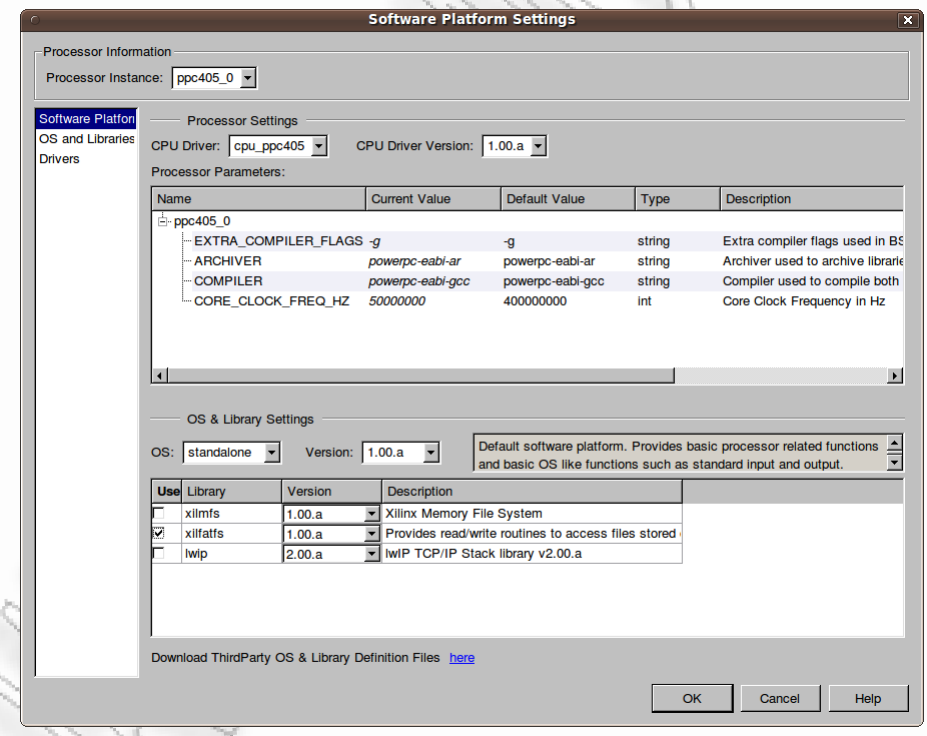
```
PORT opb_dcr_socket_0_ROPB_Clk = opb_dcr_socket_0_ROPB_Clk, DIR = O
PORT opb_dcr_socket_0_ROPB_Rst_pin = opb_dcr_socket_0_ROPB_Rst, DIR = O
PORT opb_dcr_socket_0_ROPB_ABus_pin = opb_dcr_socket_0_ROPB_ABus, DIR = O, VEC = [0:31]
PORT opb_dcr_socket_0_ROPB_BE_pin = opb_dcr_socket_0_ROPB_BE, DIR = O, VEC = [0:3]
PORT opb_dcr_socket_0_ROPB_RNW_pin = opb_dcr_socket_0_ROPB_RNW, DIR = O
PORT opb_dcr_socket_0_ROPB_select_pin = opb_dcr_socket_0_ROPB_select, DIR = O
PORT opb_dcr_socket_0_ROPB_seqAddr_pin = opb_dcr_socket_0_ROPB_seqAddr, DIR = O
PORT opb_dcr_socket_0_ROPB_DBus_pin = opb_dcr_socket_0_ROPB_DBus, DIR = O, VEC = [0:31]
PORT opb_dcr_socket_0_RSln_DBus_pin = opb_dcr_socket_0_RSln_DBus, DIR = I, VEC = [0:31]
PORT opb_dcr_socket_0_RSln_retry_pin = opb_dcr_socket_0_RSln_retry, DIR = I
PORT opb_dcr_socket_0_RSln_toutSup_pin = opb_dcr_socket_0_RSln_toutSup, DIR = I
PORT opb_dcr_socket_0_RSln_xferAck_pin = opb_dcr_socket_0_RSln_xferAck, DIR = I
PORT opb_dcr_socket_0_RSln_errAck_pin = opb_dcr_socket_0_RSln_errAck, DIR = I
PORT opb_dcr_socket_0_ROPB_BM_enable_pin = opb_dcr_socket_0_ROPB_BM_enable, DIR = O
```

Οι θύρες αυτές προέρχονται από το opb_dcr_socket περιφερειακό, στο οποίο θα πρέπει επίσης να δηλωθούν. Επομένως θα πρέπει το opb_dcr_socket να έχει τις δηλώσεις που φαίνονται παρακάτω:

```
PORT ROPB_Clk = opb_dcr_socket_0_ROPB_Clk
PORT ROPB_Rst = opb_dcr_socket_0_ROPB_Rst
PORT ROPB_ABus = opb_dcr_socket_0_ROPB_ABus
PORT ROPB_BE = opb_dcr_socket_0_ROPB_BE
PORT ROPB_RNW = opb_dcr_socket_0_ROPB_RNW
PORT ROPB_select = opb_dcr_socket_0_ROPB_select
PORT ROPB_seqAddr = opb_dcr_socket_0_ROPB_seqAddr
PORT ROPB_DBus = opb_dcr_socket_0_ROPB_DBus
PORT RSln_DBus = opb_dcr_socket_0_RSln_DBus
PORT RSln_errAck = opb_dcr_socket_0_RSln_errAck
PORT RSln_retry = opb_dcr_socket_0_RSln_retry
PORT RSln_toutSup = opb_dcr_socket_0_RSln_toutSup
PORT RSln_xferAck = opb_dcr_socket_0_RSln_xferAck
PORT ROPB_BM_enable = opb_dcr_socket_0_ROPB_BM_enable
```

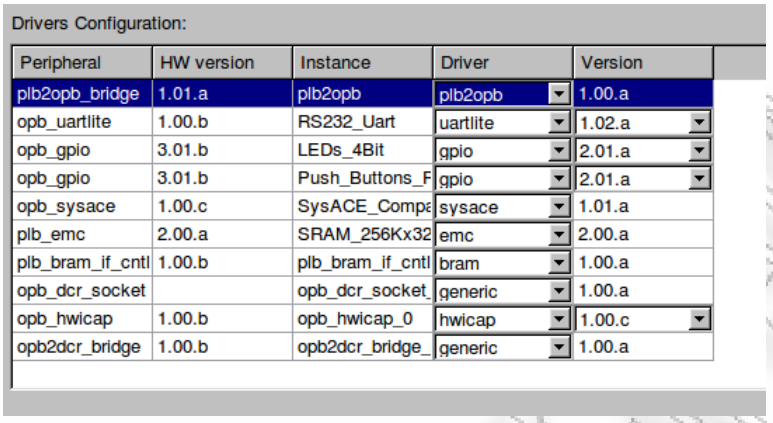
4.4.3 Ρυθμίσεις Συστήματος

Το τελικό στάδιο πριν την ολοκλήρωση του υλικού (hardware) τμήματος είναι οι ρυθμίσεις συστήματος. Προκειμένου να μπορεί το τελικό σύστημα να προσπελάσει την εξωτερική Compact Flash κάρτα στην οποία θα αποθηκεύονται τα αρχεία διαμόρφωσης, θα πρέπει να υποστηριχθεί το σύστημα αρχείων FAT. Αυτό γίνεται με προσθήκη του οδηγού xilfat, μέσω της επιλογής *Software →Software Platform Settings*, όπως φαίνεται στο Σχήμα 45.



Σχήμα 45: Επιλογές υποστήριξης λογισμικού

Τέλος πρέπει το opb_hwicap περιφερειακό να χρησιμοποιεί την έκδοση v1.00.c του οδηγού. Αυτή παρέχεται ξεχωριστά και πρέπει προηγουμένως να έχει τοποθετηθεί στον υποκατάλογο `edk/drivers` ώστε να γίνει διαθέσιμη προς χρήση από το EDK. Στο Σχήμα 46 φαίνεται η φόρμα ορισμού των drivers.



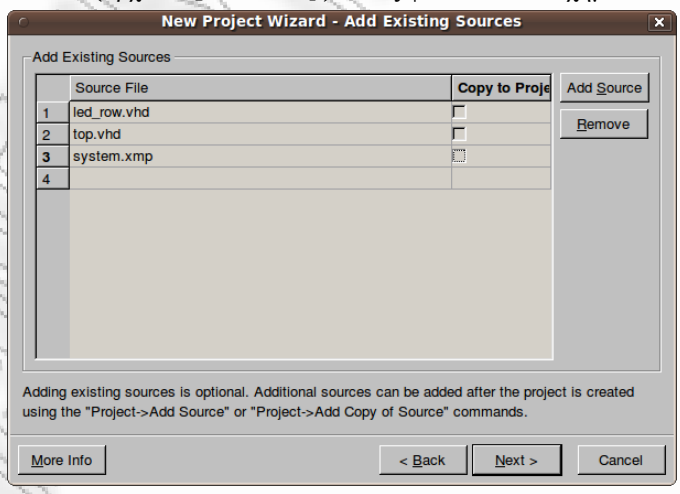
Σχήμα 46: Επιλογή έκδοσης για οδηγούς συσκευών

Έχοντας ολοκληρωθεί αυτή η φάση, μπορεί πλέον να δημιουργηθεί netlist από την επιλογή *Hardware→Generate Netlist.*

Σε αυτό το σημείο δεν έχει γίνει ακόμη η ανάπτυξη της εφαρμογής ελέγχου του συστήματος η οποία θα εκτελείται από τον PowerPC. Αυτή η διαδικασία είναι η τυπική που εφαρμόζεται σε ανάπτυξη με το EDK και δε σχετίζεται με το υπόλοιπο της υλοποίησης του επαναδιαμορφούμενου συστήματος. Κατάληξή της είναι η δημιουργία του εκτελέσιμου για τον επεξεργαστή PowerPC, το οποίο τυπικά έχει το όνομα `executable.elf`. Μια συνοπτική περιγραφή της διαδικασίας υλοποίησης της εφαρμογής παρέχεται στην παράγραφο 5.5.

4.5 Υλοποίηση κυκλώματος για το ανώτερο επιπέδου ιεραρχίας (top-level)

Για την υλοποίηση του top-level κυκλώματος, δημιουργούμε ένα νέο project στο ISE. Αυτό θα περιλαμβάνει τα αρχεία στατικής υλοποίησης (`top.vhd` και `led_row.vhd`) καθώς και το σύστημα που έχει δημιουργηθεί από το EDK (αρχείο `system.xmp`), όπως φαίνεται στο Σχήμα 47.



Σχήμα 47: Δημιουργία ανώτερου επιπέδου ιεραρχίας

Τα αρχεία (και ειδικά το `system.xmp`) δεν πρέπει να αντιγραφούν στον υποκατάλογο που θα δημιουργηθεί έτσι ώστε να υπάρχει σύνδεση ανάμεσα στο ISE και στο EDK. Κατά συνέπεια η επιλογή *Copy To Project* (Σχήμα 47) δε χρησιμοποιείται.

Για τη δημιουργία του `top.vhd` πολύ χρήσιμο είναι το αρχείο `edk/hdl/system_stub.vhd` το οποίο περιέχει την περιγραφή του συστήματος που δημιούργησε το EDK. Με βάση αυτό μπορεί να δημιουργηθεί εύκολα το `top.vhd` προσθέτοντας την επιπλέον λειτουργικότητα της στατικής περιοχής καθώς και τις κλήσεις στα bus macros και στο DCM ελεγκτή ρολογιού. Πρέπει εδώ να σημειωθεί ότι επειδή κατά τη δημιουργία του συστήματος στο EDK ορίσαμε τη συχνότητα του συστήματος στα 50MHz αντί για 100MHz, προκειμένου να χρησιμοποιήσουμε τον ταλαντωτή (oscillator) της ML403 ο οποίος είναι συχνότητας 100MHz, θα πρέπει στο DCM να δηλωθεί η παράμετρος `CLKIN_DIVIDE_BY_2 => TRUE`. Αυτό μπορεί να γίνει είτε στο VHDL κώδικα είτε στη φάση υλοποίησης με το εργαλείο PlanAhead.

Αφού ολοκληρωθεί η δημιουργία του project, πρέπει το κύκλωμα να περάσει από τη διαδικασία της σύνθεσης. Το αποτέλεσμα θα είναι το αρχείο `top.ngc` το οποίο θα χρησιμοποιηθεί στην επόμενη φάση από το εργαλείο PlanAhead.

4.6 Εισαγωγή δεδομένων στο εργαλείο PlanAhead

Έχοντας πραγματοποιήσει τη σύνθεση των στατικών και δυναμικών περιοχών του συστήματος η επόμενη φάση είναι η εισαγωγή των αποτελεσμάτων της σύνθεσης στο PlanAhead. Πριν προχωρήσουμε στην περιγραφή της διαδικασίας θα γίνει μια καταγραφή των αρχείων που θα χρησιμοποιηθούν στη συνέχεια, καθώς και της θέσης στην οποία βρίσκονται στο σύστημα αρχείων.

4.6.1 Αρχεία εισόδου του PlanAhead

Τα αρχεία που θα εισαχθούν στο PlanAhead είναι:

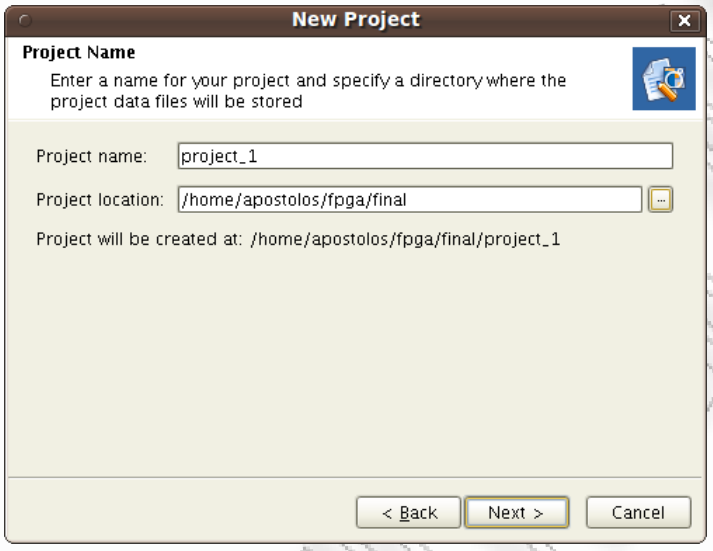
- Το `top/top.ngc`, αποτέλεσμα της σύνθεσης του top-module
- Το `pcores/encr_module_v1_00_a/devl/projnav/encr_module.ngc`, αποτέλεσμα σύνθεσης της πρώτης εκδοχής του δυναμικού περιφερειακού (TripleDES)
- Το `pcores/encr_module_v1_01_a/devl/projnav/encr_module.ngc`, αποτέλεσμα σύνθεσης της δεύτερης εκδοχής του δυναμικού περιφερειακού (RTEA)
- Το `edk/data/system.ucf`, User Constraints File για το σύστημα που δημιουργήθηκε στο EDK. Με ελάχιστες αλλαγές, αυτό θα αποτελέσει το αρχείο περιορισμών του συνολικού συστήματος.
- Το `edk/implementation/system_stub.bmm` αρχείο το οποίο περιγράφει πώς τα BlockRAMs που χρησιμοποιεί ο επεξεργαστής PowerPC διασυνδέονται μεταξύ τους ώστε να δημιουργήσουν το χώρο μνήμης του.
- Τα περιεχόμενα του καταλόγου `bus_macros/` όπου βρίσκονται τα αρχεία των bus macros που καλούνται από το top-module.

Οι αλλαγές που πρέπει να γίνουν στο `edk/data/system.ucf` αρχείο σχετίζονται με όσα σημεία του τελικού συστήματος διαφοροποιούνται σε σχέση με το σύστημα που δημιουργήθηκε στο EDK από την άποψη εισόδων και εξόδων. Έτσι για το συγκεκριμένο σύστημα οι αλλαγές είναι οι εξής:

- ✓ αλλαγή στο `sys_clk_pin` το οποίο πρέπει να οδηγείται από το pin AE14 (*System Clock*) αντί του AD12 (*User Clock*). Αυτήν η αλλαγή προέκυψε όταν κατά τη δημιουργία του συστήματος στο EDK ορίσαμε τη συχνότητα του συστήματος στα 50MHz αντί για 100MHz.
- ✓ αλλαγή στα 4 leds τα οποία οδηγούνται από το στατικό τμήμα (μέσα από το `top.vhd`) και όχι από το σύστημα που δημιουργήθηκε στο EDK.

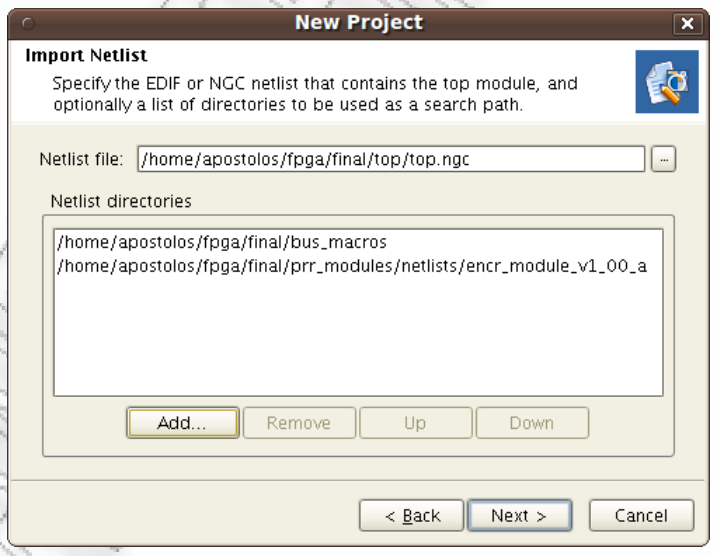
4.6.2 Δημιουργία PlanAhead project

Αφού εκκινήσουμε τη διαδικασία δημιουργίας νέου project μέσω του PlanAhead Project Wizard προσδιορίζουμε τη θέση των αρχείων εισόδου, όπως φαίνεται στο Σχήμα 48.



Σχήμα 48: Ορισμός project στο PlanAhead

Αφού επιλέξουμε ότι το project θα περιλαμβάνει netlist που έχουν περάσει σύνθεση, ορίζουμε το netlist του top-entity (`top/top.ngc`) καθώς και τους καταλόγους όπου βρίσκονται τα netlists τα οποία καλούνται από το top-module. Σε αυτό το σημείο το PlanAhead δεν έχει ακόμα πληροφορία αν το τελικό σύστημα θα εκτελεί δυναμική επαναδιαμόρφωση, οπότε αρκεί να προσδιοριστεί το netlist ενός δυναμικού περιφερειακού μόνο, όπως φαίνεται στο Σχήμα 49.



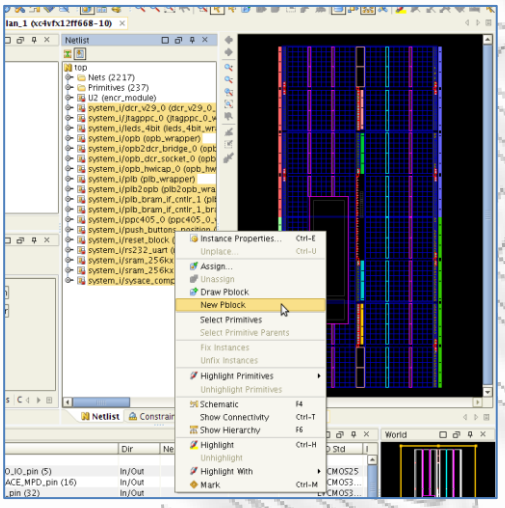
Σχήμα 49: Ορισμός αρχείων του συστήματος

Αφού φορτωθούν τα netlists και προσδιοριστεί ο τύπος του FPGA που χρησιμοποιείται φορτώνονται τα αρχεία περιορισμών (constraints files). Στην περίπτωση του συστήματός μας πρόκειται για το αρχείο `edk/data/system.ucf` στο οποίο έχουν γίνει οι αλλαγές που περιγράφονται στην παράγραφο 4.6.1.

Όταν ολοκληρωθεί η διαδικασία εισαγωγής, στο περιβάλλον του PlanAhead ορίζουμε ότι το σύστημα χρησιμοποιεί ΔΕ, από την επιλογή File → Set PR Project έτσι ώστε να ενεργοποιηθούν οι αντίστοιχες λειτουργίες.

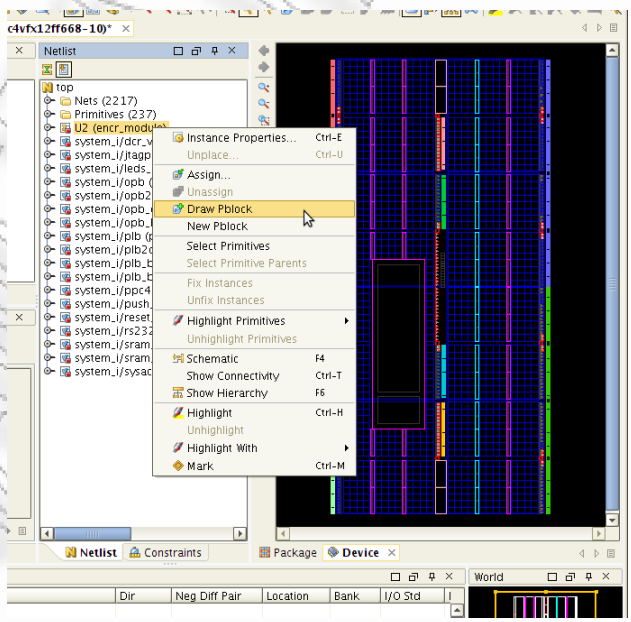
4.6.3 Ορισμός δυναμικών περιοχών και περιορισμών υλοποίησης

Το επόμενο βήμα της διαδικασίας είναι ο ορισμός των τμημάτων του συστήματος τα οποία ανήκουν στο στατικό κομμάτι. Αυτό γίνεται επιλέγοντας στο παράθυρο *Netlist* όλα τα τμήματα που ανήκουν στο στατικό τμήμα του συστήματος και δημιουργώντας ένα νέο Pblock, με όνομα π.χ. `static`, όπως φαίνεται στο Σχήμα 50.



Σχήμα 50: Δημιουργία Pblock για το στατικό τμήμα

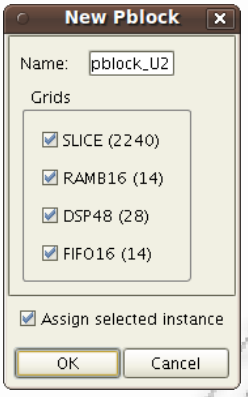
Κατόπιν ορίζουμε τα τμήματα που θα αντιστοιχηθούν σε κάθε δυναμική περιοχή. Στην υλοποίηση αυτή έχουμε μια δυναμική περιοχή, αυτή στην οποία θα φορτώνονται το component U2 (encr_module). Για να την ορίσουμε, επιλέγουμε το U2 και δίνουμε την εντολή "*Draw Pblock*", όπως φαίνεται στο Σχήμα 51. Στο παράθυρο *Device* φαίνεται η κάτοψη του FPGA. Έτσι μπορούμε να σχεδιάσουμε με το ποντίκι τη δυναμική περιοχή.



Σχήμα 51: Ορισμός Pblock για το δυναμικό τμήμα

Η δυναμική περιοχή πρέπει να είναι τόσο μεγάλη ώστε να μπορεί να υλοποιηθεί εντός αυτής το μεγαλύτερο από τα δυναμικά τμήματα. Αφού τη σχεδιάσουμε με το ποντίκι, θα εμφανιστεί το μήνυμα

που φαίνεται στο Σχήμα 52 , στο οποίο φαίνεται ποιοι είναι οι διαθέσιμοι πόροι της περιοχής που επιλέξαμε. Γνωρίζοντας από το στάδιο της σύνθεσης των δυναμικών τμημάτων ποιες είναι οι απαιτήσεις του κάθε τμήματος, μπορούμε να εκτιμήσουμε το ζητούμενο μέγεθος για τη δυναμική περιοχή.



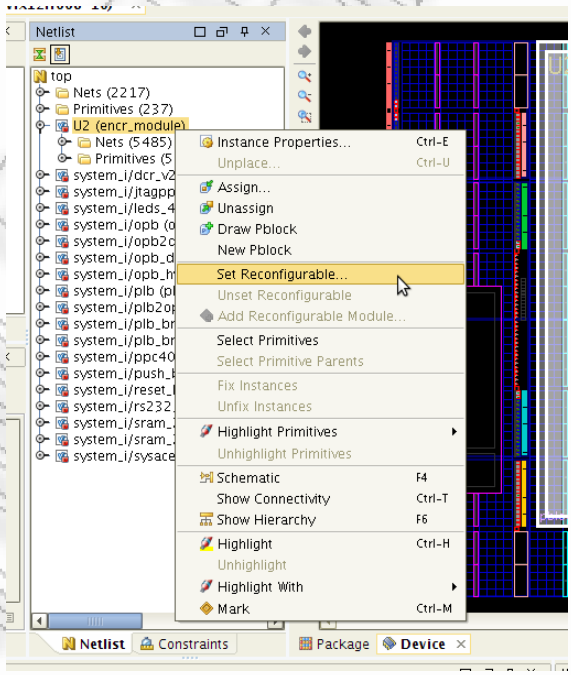
Σχήμα 52: Πόροι που περικλείει η δυναμική περιοχή

Στην περίπτωση των δύο περιφερειακών που αναπτύχθηκαν, η υλοποίηση του TripleDES αλγορίθμου απαιτεί 1904 slices ενώ η υλοποίηση του RTEA απαιτεί 547 slices. Η περιοχή που εκτείνεται από το slice X26Y16 έως το X45Y127 περιέχει 2240 slices οπότε είναι αρκετή για να χωρέσει το TripleDES περιφερειακό.

Εκτός από τους αναγκαίους πόρους, η περιοχή που θα οριστεί ως δυναμική πρέπει να είναι καλύπτει και άλλους περιορισμούς. Έτσι για το Virtex-4:

- Οι συντεταγμένες του κάτω αριστερά ορίου πρέπει να είναι *άρτιος* αριθμός
- Οι συντεταγμένες του άνω δεξιά ορίου πρέπει να είναι *περιττός* αριθμός

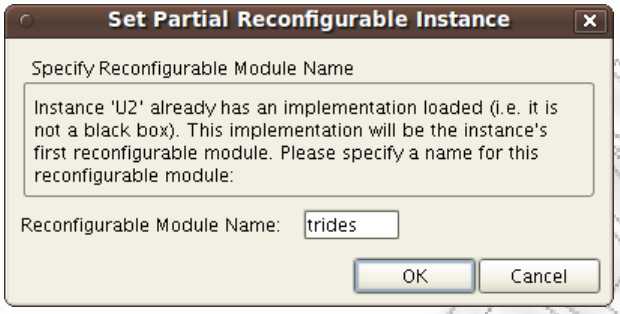
Με το πέρας της οριοθέτησης της περιοχής, πρέπει να οριστεί στο PlanAhead ότι πρόκειται για δυναμική περιοχή. Αυτό γίνεται με την επιλογή *"Set Reconfigurable"*, όπως φαίνεται στο Σχήμα 53 .



Σχήμα 53: Ορισμός της Δυναμικής Περιοχής

Κατόπιν θα πρέπει να δώσουμε ένα όνομα στην περιοχή. Εφόσον κατά την εισαγωγή δεδομένων στο PlanAhead χρησιμοποιήσαμε τον υποκατάλογο όπου βρισκόταν το netlist για την υλοποίηση του

TripleDES, δηλώνουμε `trides` σαν όνομα για το δυναμικό περιφερειακό που βρίσκεται στην περιοχή, όπως φαίνεται στο Σχήμα 54.



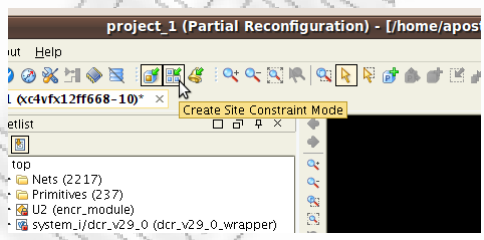
Σχήμα 54: Ορισμός ονόματος 1ου Δυναμικού Περιφεριακού

Στη συνέχεια θα πρέπει να ορίσουμε και το netlist για το άλλο δυναμικό περιφερειακό, το οποίο υλοποιεί τον RTEA. Αυτό γίνεται με την επιλογή *"Add Reconfigurable"*. Τότε πρέπει να δηλώσουμε το όνομα του δυναμικού τμήματος (`rtea`) καθώς και ποιο είναι το netlist που το ορίζει (`pcores/encr_module_v1_01_a/devl/projnav/encr_module.ngc`).

Η διαδικασία αυτή επαναλαμβάνεται μέχρι να εισαχθούν τα netlists για όλα τα δυναμικά τμήματα.

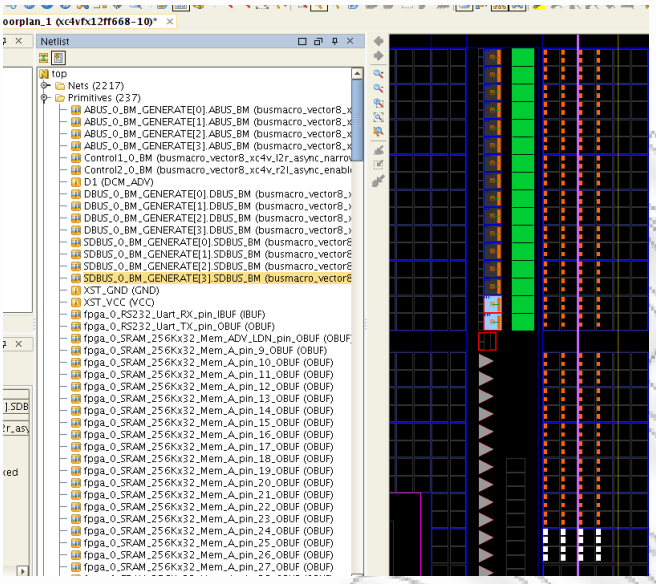
4.6.4 Ορισμός περιορισμών τοποθέτησης (Placement Constraints)

Με την ολοκλήρωσή της θα πρέπει να οριστούν οι θέσεις για τα bus macros, τα dcm, τις μνήμες BlockRAM καθώς και άλλα δομικά στοιχεία του FPGA. Αυτό γίνεται με την εντολή *"Create Site Constraint Mode"* (Σχήμα 55) οπότε και επιτρέπεται η τοποθέτηση στοιχείων πάνω στην επιφάνεια του FPGA.



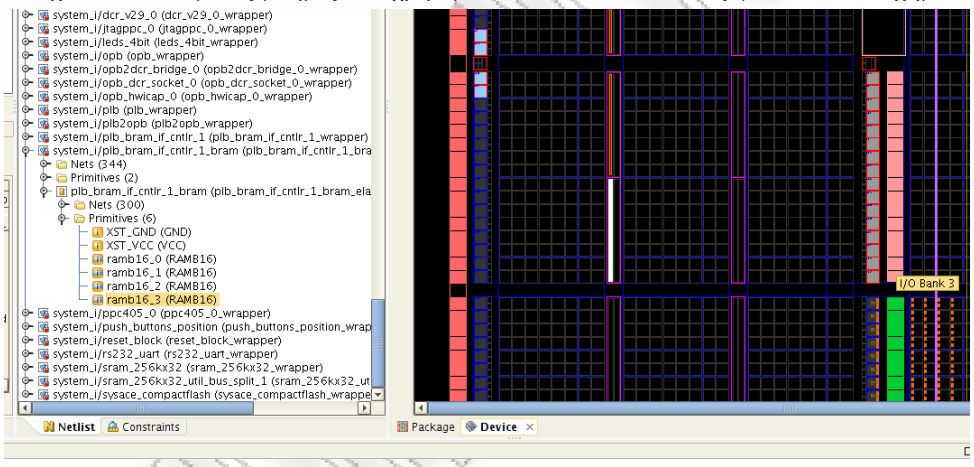
Σχήμα 55: Ενεργοποίηση εντολής για τοποθέτηση στοιχείων

Κατόπιν τοποθετούμε όλα τα bus macros στο όριο μεταξύ στατικής και δυναμικής περιοχής, όπως φαίνεται στο Σχήμα 56. Λεπτομέρειες σχετικά με την τοποθέτηση υπάρχουν στο [9].



Σχήμα 56: Τοποθέτηση των bus macros

Αντίστοιχα τοποθετούμε τις μνήμες που χρησιμοποιεί ο PowerPC, όπως φαίνεται στο Σχήμα 57



Σχήμα 57: Τοποθέτηση των μνημών

Η τοποθέτηση των μνημών BlockRAM γίνεται έξω από τη δυναμική περιοχή και συγκεκριμένα στις θέσεις `RAMB16_X012` έως `RAMB16_X015`.

Τέλος τοποθετούμε τον ελεγκτή ρολογιού στη θέση `DCM_ADV_X0Y1` και τον οδηγό διανομής ρολογιού `sys_clk_bufg` στη θέση `BUFGCTRL_X0Y0`.

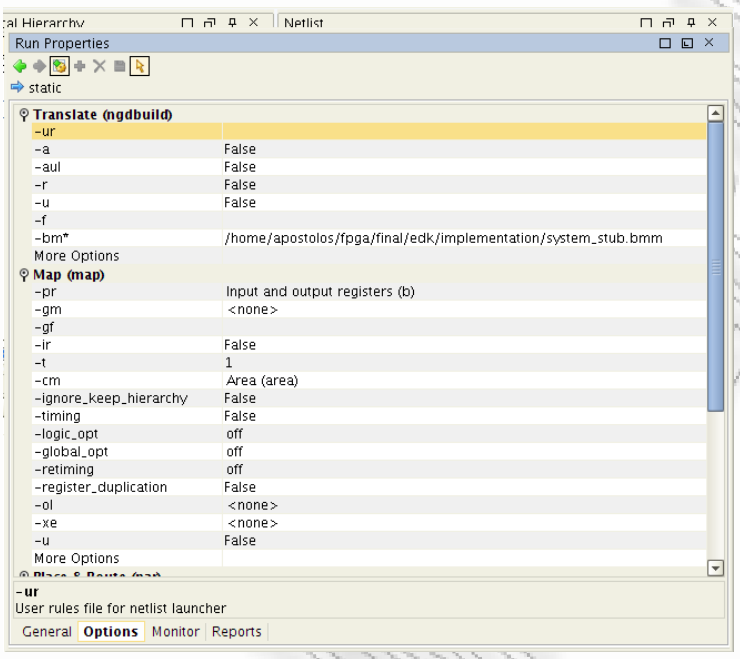
Πρέπει να σημειωθεί ότι όλοι οι παραπάνω περιορισμοί μπορούν και να οριστούν σε ένα αρχείο περιορισμών (`.ucf`) το οποίο θα εισαχθεί στο PlanAhead με την εντολή *File → Import Constraints*.

Πριν προχωρήσει η διαδικασία, θα πρέπει να γίνει ένας έλεγχος για συμμόρφωση με τους κανόνες υλοποίησης (Design Rules Check, DRC). Αυτό γίνεται επιλέγοντας *Tools → DRC* οπότε αφού γίνουν οι έλεγχοι θα παρουσιαστεί η αντίστοιχη αναφορά. Στην περίπτωση της υλοποίησης Επαναδιαμορφούμενου Συστήματος, εμφανίζονται προειδοποιήσεις ότι η απόδοση του συστήματος μπορεί να αυξηθεί με χρήση σύγχρονου σχεδιασμού. Οι προειδοποιήσεις αναφέρονται στα bus macros και μπορούν να αγνοηθούν.

4.6.5 Εκτέλεση της ροής ΔΕ

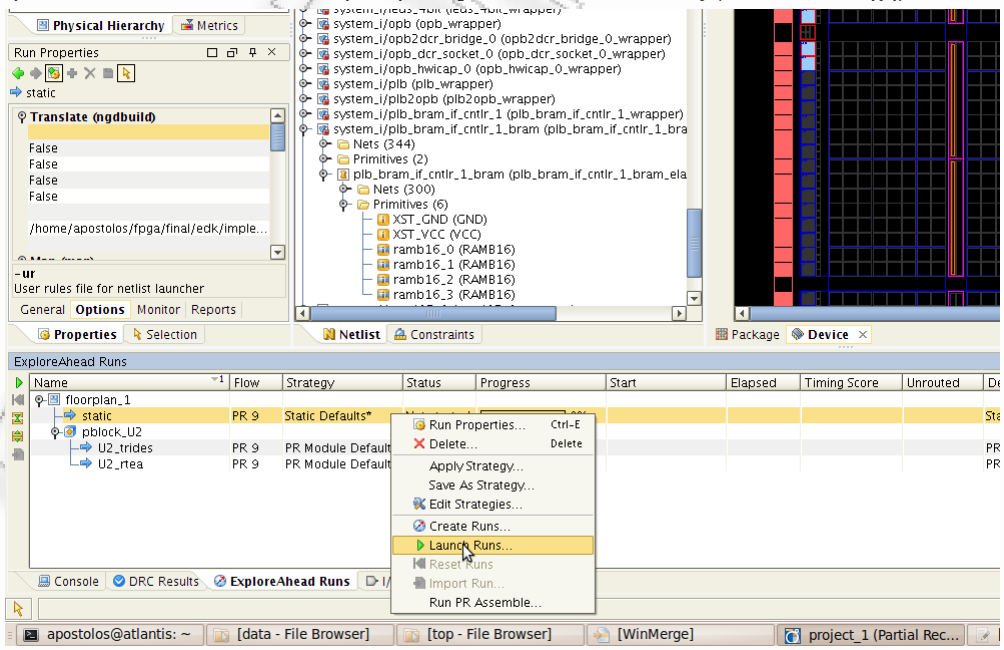
Έχοντας ορίσει και τους περιορισμούς τοποθέτησης, το τελευταίο βήμα πριν γίνει η υλοποίηση του συστήματος είναι να ορισθεί η θέση του αρχείου `system_stub.bmm`. Το αρχείο αυτό περιγράφει πώς τα

BlockRAMs που χρησιμοποιεί ο επεξεργαστής PowerPC δημιουργούν το χώρο μνήμης του και βρίσκεται στον υποκατάλογο `edk/implementation`. Ο ορισμός γίνεται επιλέγοντας *static* στο παράθυρο *ExploreAhead Runs*. Τότε στο παράθυρο *Run Properties* → *Options* → *Translate (ngdbuild)* ορίζουμε στην επιλογή `-bm` τη θέση `edk/implementation/system_stub.bmm` όπως φαίνεται στο Σχήμα 58 και κατόπιν επιλέγουμε *Apply* ώστε η αλλαγή να γίνει αποδεκτή.



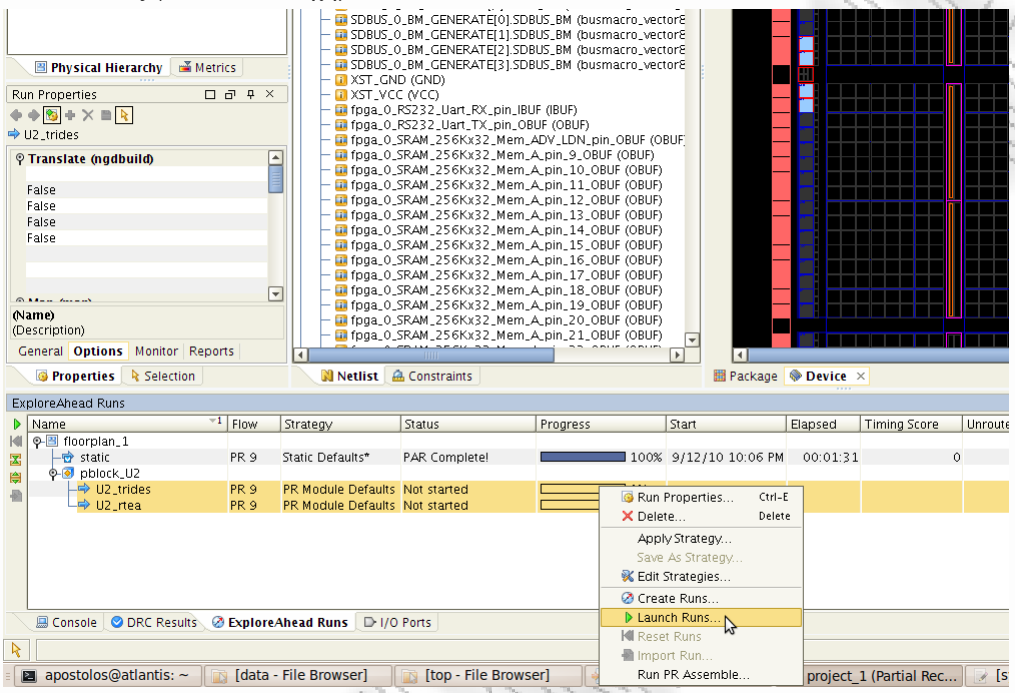
Σχήμα 58: Ορισμός θέσης του αρχείου system_stub.bmm

Πρώτο βήμα της εκτέλεσης της ροής ΔΕ είναι η υλοποίηση του στατικού τμήματος. Αυτό γίνεται με την επιλογή *Static* → *Launch Runs* στο παράθυρο *ExploreAhead Runs* όπως φαίνεται στο Σχήμα 59.



Σχήμα 59: Υλοποίηση του στατικού τμήματος

Τότε θα αρχίσουν να καλούνται διαδοχικά τα εργαλεία υλοποίησης (ngdbuild, map, par) προκειμένου να υλοποιήσουν το κύκλωμα που βρίσκεται στην στατική περιοχή. Μετά το πέρας της υλοποίησης θα πρέπει να γίνει η αντίστοιχη διαδικασία υλοποίησης για τα δυναμικά τμήματα. Αυτό γίνεται επιλέγοντας από κοινού όλα τα δυναμικά τμήματα στο παράθυρο *ExploreAhead Runs* και κατόπιν δίνοντας την εντολή *Launch Runs*, όπως φαίνεται στο Σχήμα 60.



Σχήμα 60: Διαδικασία υλοποίησης για τα δυναμικά τμήματα

Αντίστοιχα με πριν, καλούνται τα εργαλεία υλοποίησης και στο τέλος δημιουργούνται τα αρχεία που περιγράφουν την υλοποίηση της κάθε δυναμικής περιοχής ξεχωριστά. Στη διαδικασία υλοποίησης κάθε δυναμικού περιφερειακού λαμβάνεται υπόψη το σύνολο των διαδρομών που ανήκουν στο στατικό τμήμα αλλά διέρχονται από τη δυναμική περιοχή. Προφανώς αυτές οι διαδρομές δε μπορούν να χρησιμοποιηθούν στην υλοποίηση των δυναμικών τμημάτων. Αυτός είναι και ο λόγος που εάν αλλάξει το στατικό τμήμα του συστήματος τα δυναμικά τμήματα πλέον δε μπορούν να χρησιμοποιηθούν.

Τελικό βήμα της εκτέλεσης είναι η διαδικασία συνένωσης του στατικού τμήματος με κάθε συνδιασμό δυναμικών τμημάτων. Αυτό γίνεται επιλέγοντας στο παράθυρο *ExploreAhead Runs* ένα από τα δυναμικά τμήματα και κατόπιν δίνοντας την εντολή *Run PR Assemble*. Αφού ζητηθεί να ορίσουμε το δυναμικό τμήμα που θα είναι φορτωμένο κατά την εκκίνηση του συστήματος, θα δημιουργηθούν τα εξής αρχεία διαμόρφωσης:

- `pblock_U2_blank.bit`, το οποίο είναι η υλοποίηση του στατικού τμήματος χωρίς κανένα δυναμικό
- `static_full.bit`, το οποίο είναι η υλοποίηση του στατικού τμήματος με το δυναμικό τμήμα που δηλώθηκε κατά την εκτέλεση της εντολής PR Assemble
- `u2_rtea_partial.bit`, το οποίο είναι η διαμόρφωση του ενός δυναμικού τμήματος
- `u2_trides_partial.bit`, το οποίο είναι η διαμόρφωση του άλλου δυναμικού τμήματος

Αυτά τα αρχεία μπορεί να χρησιμοποιηθούν απ' ευθείας για τη διαμόρφωση του FPGA εάν φορτωθούν μέσω του JTAG debugger. Προκειμένου να μπορεί ο PowerPC να τα φορτώσει από την CompactFlash μνήμη θα πρέπει να συνενωθεί το αρχείο `static_full.bit` με το αρχείο `executable.elf` που περιέχει το εκτελέσιμο της εφαρμογής του PowerPC, μια διαδικασία που περιγράφεται στην ενότητα 4.6.6.

4.6.6 Δημιουργία αρχείων διαμόρφωσης

Η δημιουργία των αρχείων διαμόρφωσης γίνεται σε δύο στάδια:

- Πρώτα συνενώνεται το αρχείο `static_full.bit` με το αρχείο `executable.elf`. Αυτό γίνεται με εκτέλεση της εντολής:

```
data2mem -bm edk/implementation/system_stub_bd.bmm
         -bt edk/implementation/static_full.bit
         -bd application_dir/executable.elf
         tag ppc405_0
         -o b edk/implementation/download.bit
```

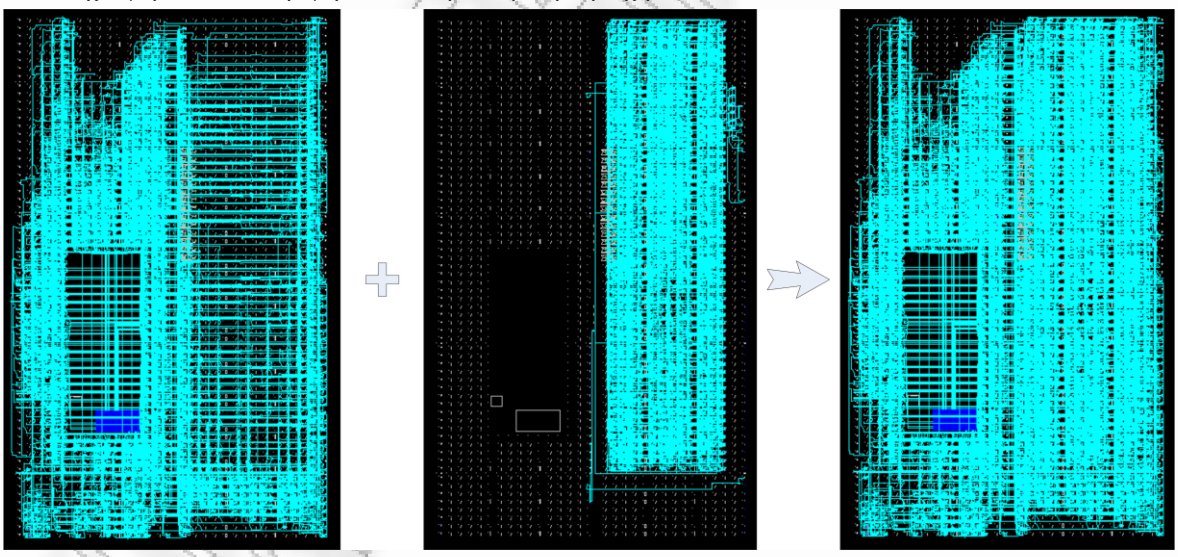
- Στη συνέχεια δημιουργείται το αρχείο `system.ace` το οποίο μπορεί να αποθηκευτεί στην Compact Flash και να κάνει τη διαμόρφωση του FPGA, με την εντολή:

```
xmd -tcl genace.tcl -jprog -target ppc_hw -board ml403
    -hw edk/implementation/download.bit
    -elf application_dir/executable.elf
    -ace system.ace
```

Το αρχείο `system.ace` τοποθετείται στην μνήμη Compact Flash μαζί με τα αρχεία διαμόρφωσης του κάθε δυναμικού τμήματος (`u2_rtea_partial.bit` και `u2_trides_partial.bit`) καθώς και το αρχείο που δεν περιλαμβάνει κανένα δυναμικό τμήμα (`pblock_U2_blank.bit`).

4.7 Επισκόπηση δημιουργηθέντων αρχείων

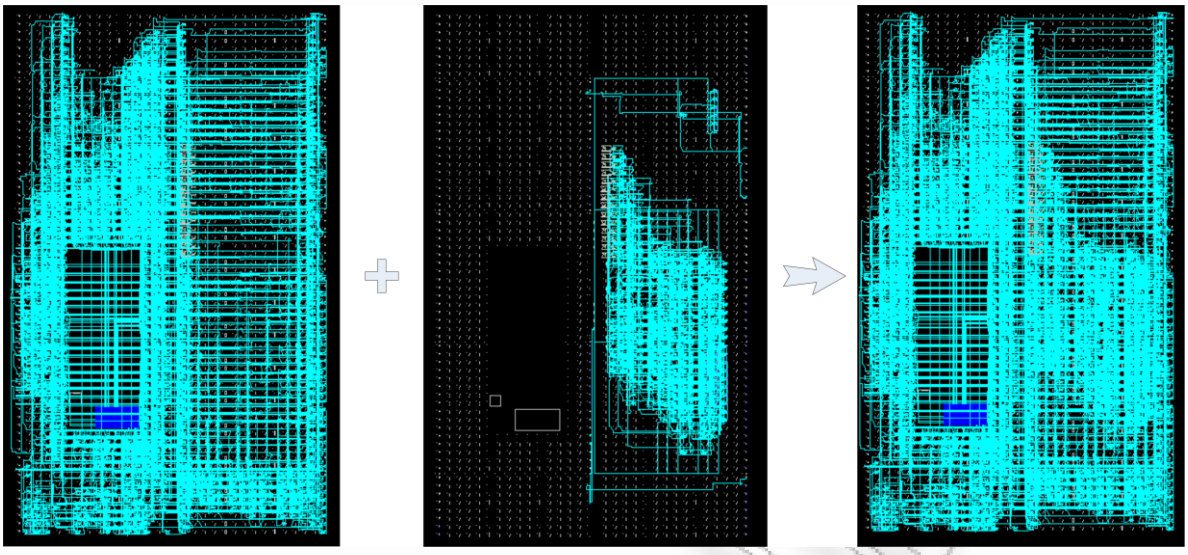
Με την χρήση του FPGA Editor μπορέσαμε να δούμε την υλοποίηση κάθε διαμόρφωσης που δημιουργήθηκε μετά την ολοκλήρωση της εκτέλεσης της ροής ΔΕ από το PlanAhead. Στο Σχήμα 61 φαίνεται η υλοποίηση του στατικού τμήματος, του TripleDES περιφερειακού καθώς και της υλοποίησης όταν έχει φορτωθεί το περιφερειακό στη δυναμική περιοχή.



Σχήμα 61: Επισκόπηση της υλοποίησης του TripleDES περιφερειακού

Αντίστοιχα, στο Σχήμα 61 φαίνεται η υλοποίηση του στατικού τμήματος, του RTEA περιφερειακού καθώς και της υλοποίησης όταν έχει φορτωθεί το περιφερειακό στη δυναμική περιοχή.

Και στις δύο περιπτώσεις διακρίνονται τα bus macros ενώ είναι εμφανής η χρήση διαδρομών μέσα στη δυναμική περιοχή για την υλοποίηση του στατικού τμήματος.



Σχήμα 62: Επισκόπηση της υλοποίησης του RTEA περιφερειακού

5

Ανάπτυξη Πειραματικής Διάταξης

Στο κεφάλαιο αυτό παρουσιάζεται η πειραματική διάταξη που αναπτύχθηκε προκειμένου να γίνει επίδειξη της διαδικασίας ΔΕ. Γίνεται ανάλυση της λειτουργικότητας της εφαρμογής ελέγχου του επαναδιαμορφούμενου συστήματος η οποία εκτελείται στον επεξεργαστή PowerPC και επιτρέπει την φόρτωση των δυναμικών τμημάτων. Επιπλέον παρουσιάζονται εικόνες από την εκτέλεση της εφαρμογής

5.1 Περιγραφή Λειτουργικότητας

Η εφαρμογή ελέγχου είναι γραμμένη σε γλώσσα C και χρησιμοποιεί τη σειριακή θύρα του αναπτυξιακού συστήματος ML403 προκειμένου να λαμβάνει εντολές ώστε να φορτώνει τα διάφορα δυναμικά τμήματα στη δυναμική περιοχή του FPGA. Αυτό γίνεται μέσα από ένα μενού επιλογών το οποίο καθοδηγεί το χρήστη ώστε είτε να διαλέξει τη φόρτωση κάποιου δυναμικού περιφερειακού (ή την εκ-φόρτωση όλων) ή να εκτελέσει τη διαδικασία ελέγχου. Κατά τη διαδικασία ελέγχου, η οποία είναι κοινή για τα δύο περιφερειακά, τυχαία δεδομένα:

- κρυπτογραφούνται από το υλικό και αποκρυπτογραφούνται από το λογισμικό,
- κρυπτογραφούνται και αποκρυπτογραφούνται από το υλικό,
- κρυπτογραφούνται και αποκρυπτογραφούνται από το λογισμικό

ενώ οι αντίστοιχοι χρόνοι εκτέλεσης συγκρίνονται προκειμένου να φανεί η διαφορά στην ταχύτητα εκτέλεσης αλλά και η σωστή λειτουργία των περιφερειακών.

5.2 Φόρτωση Δυναμικών Περιφερειακών

Η φόρτωση των δυναμικών περιφερειακών γίνεται με χρήση της συνάρτησης `XHwIcap_CF2Icap()`. Αυτή χρησιμοποιεί τον οδηγό xilfat προκειμένου να προσπελάσει αρχεία που είναι αποθηκευμένα στη (μορφοποιημένη κατά FAT16) Compact Flash και να τα φορτώσει στο FPGA μέσω της θύρας ICAP. Πριν τη φόρτωση θα πρέπει τα bus macros να έχουν απομονώσει τη δυναμική περιοχή. Ο έλεγχος των bus macros γίνεται από το OPB2DCR_Socket διαμέσου της γέφυρας OPB2DCR. Όπως και κάθε είδους επικοινωνία με περιφερειακά που βρίσκονται στον OPB δίαυλο η C εφαρμογή μπορεί να ελέγξει τα bus macros με χρήση των συναρτήσεων `XIo_Out32()` και `XIo_In32()`. Στο Σχήμα 63 φαίνεται ένα απόσπασμα κώδικα το οποίο φορτώνει το TripleDES περιφερειακό.

```
case '3': // 3DES
        xil_printf("\r\n      Performing reconfiguration for 3DES module");
              XIo_Out32(XPAR_OPB_DCR_SOCKET_0_DCR_BASEADDR,0x00000000);  // disable
        xil_printf("\r\n      Existing PRR disabled, will now load module ");
              XHwIcap_CF2Icap(&MyIcap, "trides.bit");
        xil_printf("\r\n      Bitstream loaded, will now enable PRR ");
              XIo_Out32(XPAR_OPB_DCR_SOCKET_0_DCR_BASEADDR,0x00000001);  // enable
        xil_printf("\r\n      3DES module enabled, back to menu \n\r");
        menu();
        break;
```

Σχήμα 63: Φόρτωση του TripleDES περιφερειακού

Με τη χρήση της συνάρτησης `XTime_GetTime()` μπορέσαμε να προσπελάσουμε τον εσωτερικό 64bit μετρητή κύκλων μηχανής του PowerPC ώστε να μετρήσουμε το χρόνο εκτέλεσης της διαδικασίας φόρτωσης των δυναμικών περιφερειακών στο FPGA κατά την εκτέλεση της `XHwIcap_CF2Icap()`.

5.3 Λειτουργία κώδικα ελέγχου TripleDES περιφερειακού

Ο κώδικας ελέγχου του TripleDES περιφερειακού χρησιμοποιεί το αρχείο `triple_des.h` που δημιουργήθηκε αυτόματα από τον *Create and Import Peripheral Wizard*. Αυτό περιέχει ορισμούς και μακροεντολές με τις οποίες απλοποιείται η συγγραφή κώδικα για προσπέλαση του περιφερειακού. Με βάση αυτούς δημιουργήθηκε η συνάρτηση `des_data_new()` η οποία εκτελεί διαδοχικά τρεις διαδικασίες που αναφέρθηκαν στην παράγραφο 5.1. Το κλειδί για τον TripleDES αλγόριθμο είναι σταθερά ορισμένο στον κώδικα μιας και στην παρούσα εφαρμογή δε μας ενδιαφέρει η χρησιμοποίηση διαφορετικών κλειδιών αλλά η λειτουργία του αλγορίθμου σε υλικό και λογισμικό.

Για την προσπέλαση του TripleDES περιφερειακού χρησιμοποιούνται και πάλι οι συναρτήσεις `XIo_Out32()` και `XIo_In32()`. Με τη χρήση της συνάρτησης `XTime_GetTime()` μετρήσαμε το χρόνο εκτέλεσης της κάθε διαδικασίας.

Στην περίπτωση που η κρυπτογράφηση και η αποκρυπτογράφηση γίνονται με το TripleDES περιφερειακό, η λειτουργία του προγράμματος έχει ως εξής:

Η συνάρτηση εκτελεί ένα βρόχο 10 επαναλήψεων όπου:

- Δημιουργεί ένα τυχαίο διάνυσμα εισόδου (plaintext) με τη βοήθεια της συνάρτησης `rand()`
- Προγραμματίζει το περιφερειακό για κρυπτογράφηση
- Φορτώνει το κλειδί
- Φορτώνει το διάνυσμα εισόδου στο TripleDES περιφερειακό
- Διαβάζει το αποτέλεσμα
- Προγραμματίζει το περιφερειακό για αποκρυπτογράφηση
- Φορτώνει το αποτέλεσμα σαν είσοδο στο TripleDES περιφερειακό
- Διαβάζει το αποτέλεσμα
- Ελέγχει αν το τελικό αποτέλεσμα είναι ίδιο με το αρχικό διάνυσμα εισόδου που προέκυψε από την γεννήτρια τυχαίων αριθμών

Πρέπει να σημειωθεί ότι το TripleDES περιφερειακό απαιτεί να φορτωθεί το κλειδί κάθε φορά που αλλάζει ο τρόπος λειτουργίας του (κρυπτογράφηση ή αποκρυπτογράφηση) [23], ακόμη κι αν αυτό παραμείνει ίδιο. Η χρήση του περιφερειακού γίνεται βάσει των καταχωρητών που είναι ορατοί από τον OPB δίαυλο με βάση τα όσα αναφέρονται στην παράγραφο 4.3.1. Επιπλέον, το περιφερειακό χρειάζεται 55 κύκλους ρολογιού για να έχει έτοιμο το αποτέλεσμα και δεδομένου ότι ο χρόνος που μεσολαβεί ανάμεσα σε διαδοχικές εκτελέσεις των εντολών `XIo_Out32()` και `XIo_In32()` είναι πολύ μεγαλύτερος από 55 κύκλους (περίπου 280 κύκλοι), με το που θα γίνει η εγγραφή της εντολής για εκκίνηση της διαδικασίας κρυπτογράφησης, στην επόμενη εντολή του C-κώδικα το αποτέλεσμα θα είναι έτοιμο προς ανάγνωση. Για το λόγο αυτό δε χρειάζεται να γίνει έλεγχος στο bit BUSY του καταχωρητή status πριν την ανάγνωση του αποτελέσματος.

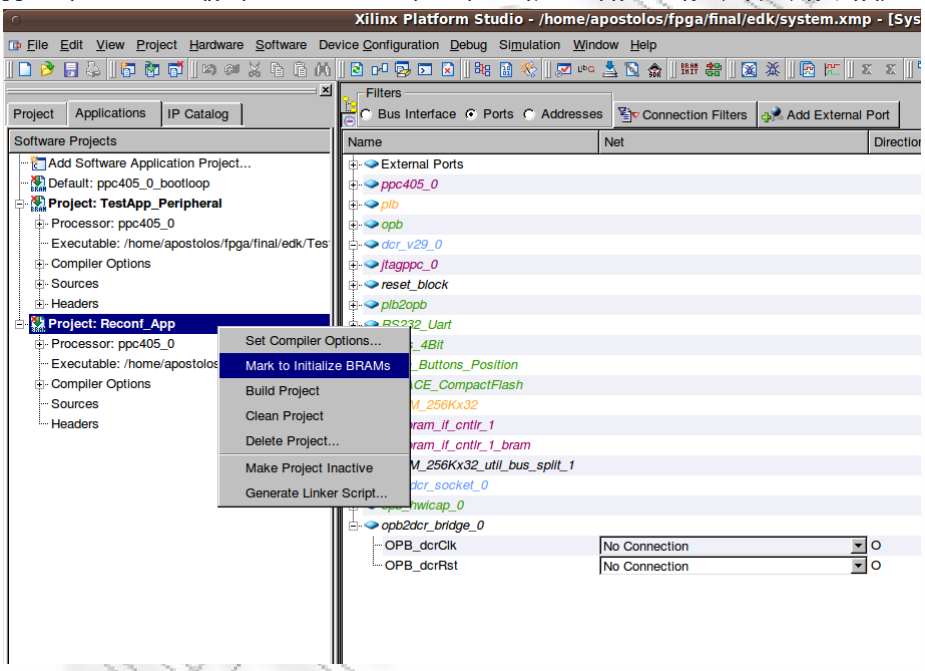
Αναφορικά με την υλοποίηση του TripleDES σε λογισμικό, αυτή βασίστηκε στον ανοικτό κώδικα που είναι διαθέσιμος στην ιστοσελίδα `http://sourceforge.net` [25] ο οποίος χρησιμοποιήθηκε με μικρές τροποποιήσεις ώστε να συνεργάζεται με την υπόλοιπη εφαρμογή.

5.4 Λειτουργία κώδικα ελέγχου RTEA περιφερειακού

Η λειτουργικότητα του RTEA περιφερειακού είναι ακριβώς ίδια με του TripleDES. Η συνάρτηση υποστήριξης του RTEA, `rtea_data_new()`, εκτελεί διαδοχικά τις τρεις διαδικασίες που περιγράφονται στην παράγραφο 5.1. Η μόνη διαφοροποίηση σε σχέση με τον TripleDES είναι ότι το κλειδί δε χρειάζεται να φορτωθεί εκ νέου όποτε αλλάζει η λειτουργικότητα (κρυπτογράφηση/αποκρυπτογράφηση). Όπως και στην περίπτωση του TripleDES χρησιμοποιούμε τη συνάρτηση `XTime_GetTime()` για να μετρήσουμε το χρόνο εκτέλεσης των υλοποιούμενων διαδικασιών.

5.5 Υλοποίηση της Εφαρμογής στο περιβάλλον XPS

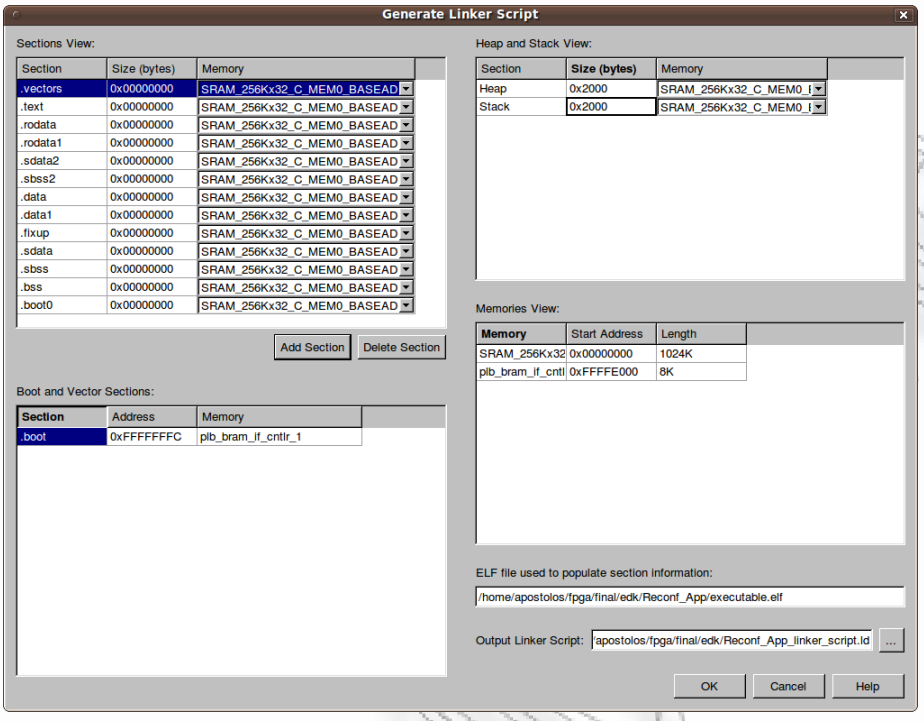
Η υλοποίηση της εφαρμογής έγινε στο περιβάλλον του XPS. Εκεί δημιουργήθηκε η εφαρμογή Reconf_App στην οποία εισήχθησαν όλα τα απαραίτητα πηγαία αρχεία της εφαρμογής (Σχήμα 64).



Σχήμα 64: Δημιουργία εφαρμογής PowerPC

Κατόπιν με την εντολή *"Generate Linker Script"* δημιουργήθηκε το αντίστοιχο αρχείο για τη διασύνδεση των τμημάτων, όπως φαίνεται στο Σχήμα 65. Στο αρχείο αυτό όλα τα τμήματα του προγράμματος τοποθετήθηκαν στην εξωτερική μνήμη μιας και ο χώρος που είναι διαθέσιμος από τις εσωτερικές μνήμες BlockRAM είναι πολύ μικρός. Ενδεχομένως θα μπορούσε να τοποθετηθεί στη BlockRAM η στοίβα (stack) ή ο σωρός (heap) προκειμένου να αυξηθεί η ταχύτητα προσπέλασης στα αντίστοιχα δεδομένα.

Σημαντικό για την σωστή λειτουργία του προγράμματος είναι να οριστεί αρκετός χώρος στη στοίβα και στο σωρό. Για το λόγο αυτό το μέγεθος και των δύο ορίστηκε σε 0x2000 θέσεις μνήμης.

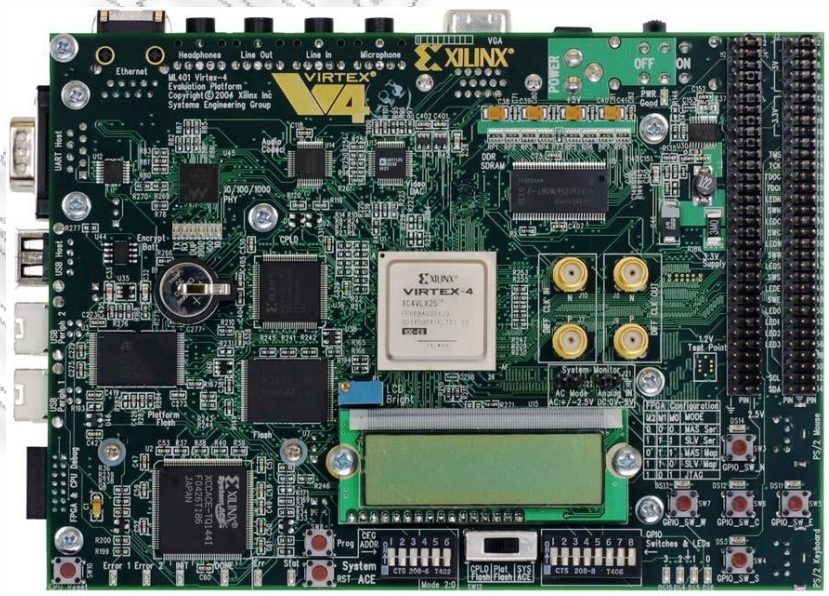


Σχήμα 65: Δημιουργία αρχείου διασύνδεσης

Έπειτα με την εντολή *"Mark to Initialize BRAMs"* ορίστηκε ότι τμήματα του εκτελέσιμου θα πρέπει να τοποθετηθούν στην εσωτερική μνήμη BlockRAM ενώ με την εντολή *"Build Project"* δημιουργήθηκε το εκτελέσιμο της εφαρμογής, `executable.elf` Τα βήματα αυτά γίνονται μετά το στάδιο που περιγράφεται στην παράγραφο 4.4.3.

5.6 Περιγραφή της πειραματικής διάταξης

Η πειραματική διάταξη η οποία χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής αποτελείται από την κάρτα ML403, η οποία φαίνεται στο Σχήμα 66.

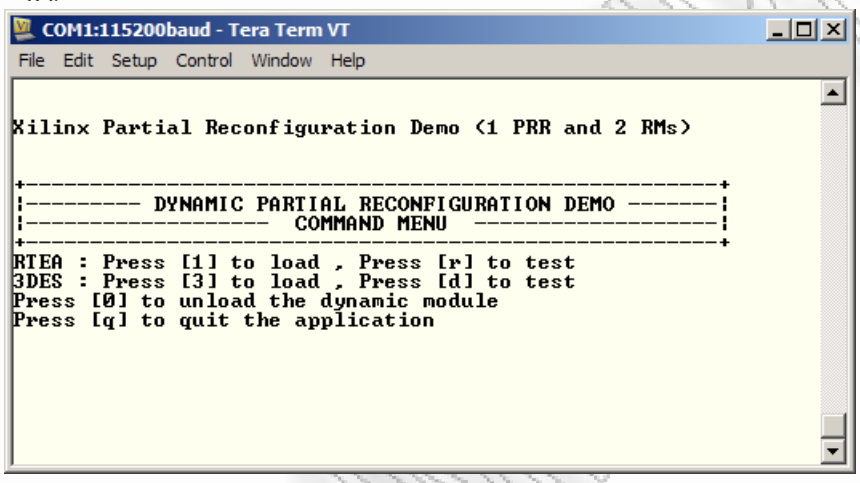


Σχήμα 66: Αναπτυξιακή Κάρτα ML403

Στο κάτω μέρος της κάρτας τοποθετείται η μνήμη Compact Flash η οποία χρησιμοποιείται για την αποθήκευση των αρχείων διαμόρφωσης. Στη σειριακή θύρα (UART Host) συνδέεται ένα τερματικό ή μέσω καλωδίου null modem η σειριακή θύρα ενός υπολογιστή (PC). Οι παράμετροι της επικοινωνίας είναι: Baudrate 115200bps, 8 data bits, 1 stop bit, No Parity (115200-8N1)

5.7 Εκτέλεση της εφαρμογής

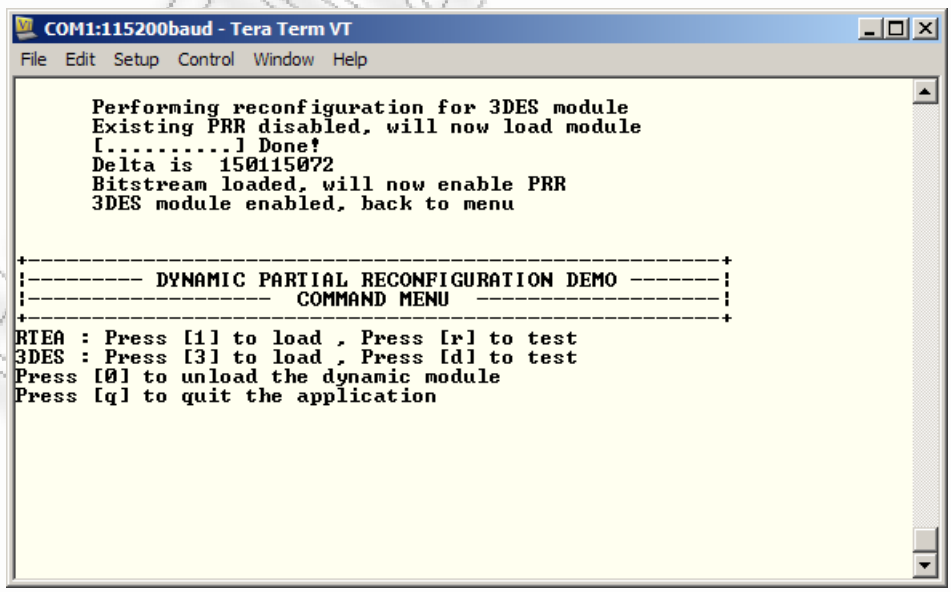
Το προγράμματος εξομοίωσης τερματικού Tera Term v4.67 [26] χρησιμοποιήθηκε για την επικοινωνία του PC με την κάρτα ML403. Ξεκινώντας το σύστημα παρουσιάζεται το μενού επιλογής εντολών σύμφωνα με το Σχήμα 67.



Σχήμα 67: Μενού Επιλογής Εντολών

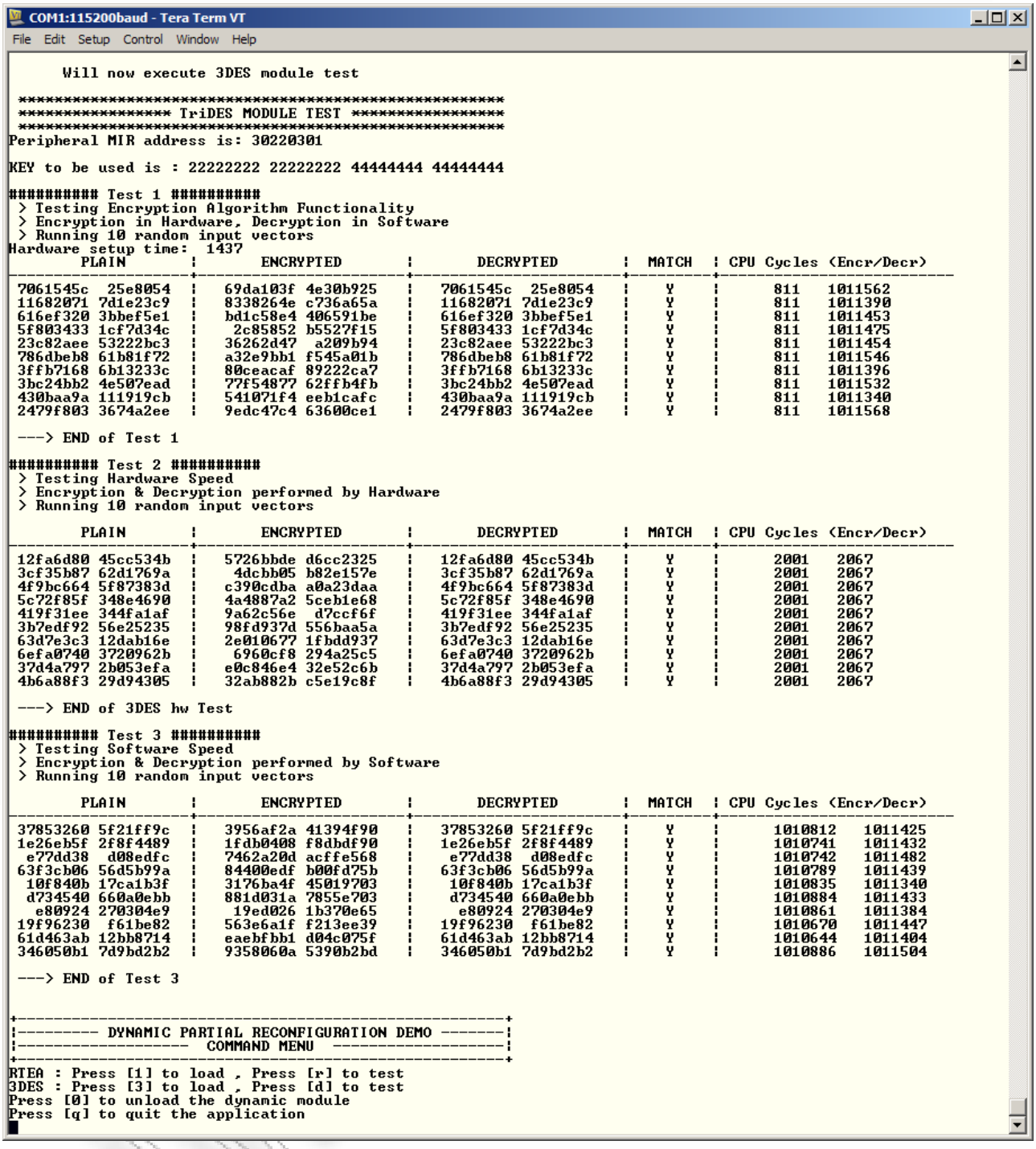
Σε αυτό το σημείο είναι ήδη φορτωμένο στο FPGA ένα δυναμικό περιφερειακό, αφού στη φάση *"merge"* της υλοποίησης με το PlanAhead πρέπει να προσδιοριστεί ποιο από τα δυναμικά τμήματα θα εγγραφεί στο αρχείο διαμόρφωσης το οποίο ξεκινά το FPGA. Στην περίπτωση αυτής της υλοποίησης έχει χρησιμοποιηθεί το περιφερειακό που υλοποιεί τον RTEA αλγόριθμο.

Επιλέγοντας **3** φορτώνεται το περιφερειακό που εκτελεί τον TripleDES αλγόριθμο (Σχήμα 68). Ο χρόνος φόρτωσης (Delta), είναι 150115072 κύκλοι του PowerPC εμφανίζεται μετά τη φόρτωση του αντίστοιχου περιφερειακού.



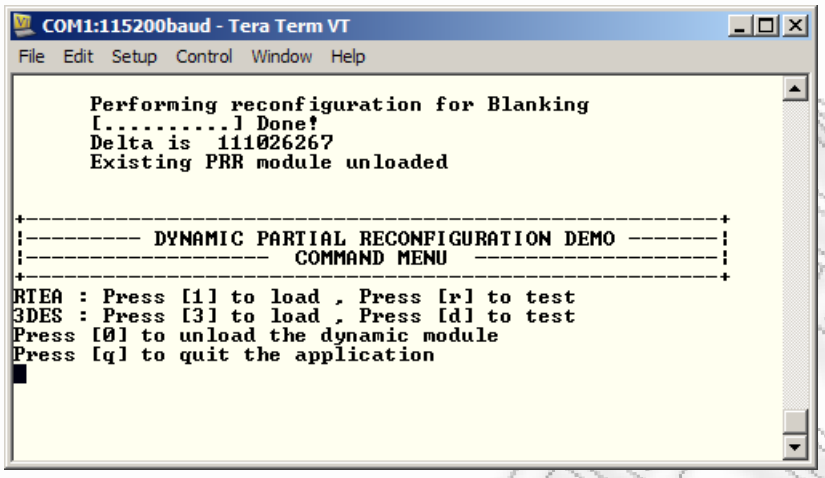
Σχήμα 68: Φορτωση του TripleDES περιφερειακού

Μετά τη φόρτωση οι δοκιμές εκτελούνται πατώντας το πλήκτρο **d** όπως φαίνεται στο Σχήμα 69.



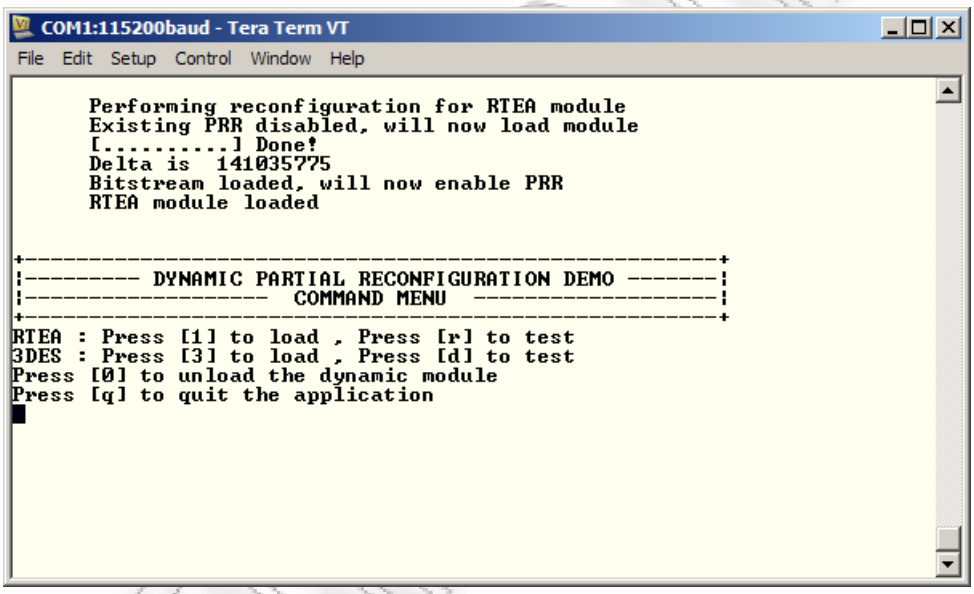
Σχήμα 69: Δοκιμές με το TripleDES περιφερειακό

Κατόπιν μπορεί να φορτωθεί το άλλο περιφερειακό ή να «ξεφορτωθεί» τελείως η δυναμική περιοχή. Αυτό γίνεται με την επιλογή **0**, όπως φαίνεται στο Σχήμα 70.



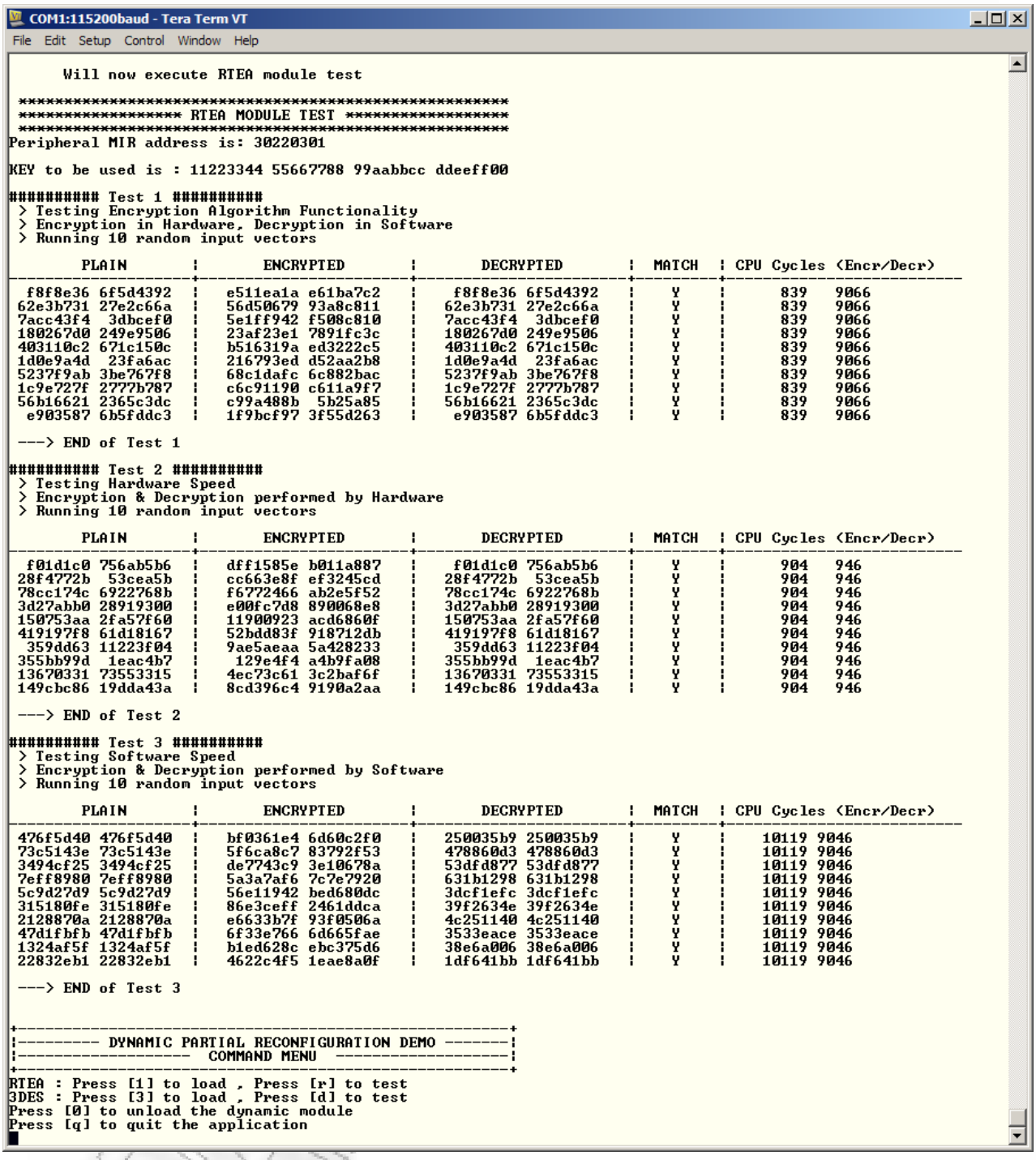
Σχήμα 70: Ξεφόρτωμα περιφερειακών απο τη δυναμική περιοχή

Για να φορτωθεί το περιφερειακό που εκτελεί τον RTEA επιλέγουμε **1**, όπως φαίνεται στο Σχήμα 71.



Σχήμα 71: Φόρτωση του RTEA περιφερειακού

Για να εκτελεστούν οι δοκιμές πατάμε το πλήκτρο **r**, σύμφωνα με το Σχήμα 72.



Σχήμα 72: Δοκιμές με το RTEA περιφερειακό

6

Συμπεράσματα - Επεκτάσεις

6.1 Συμπεράσματα

Σκοπός της παρούσας εργασίας ήταν να μελετηθεί η Δυναμική Επαναδιαμόρφωση και να δημιουργηθεί μια ολοκληρωμένη εφαρμογή που να την αξιοποιεί. Αυτό επετεύχθη με την ανάπτυξη μιας κρυπτογραφικής μηχανής η οποία χρησιμοποιεί αναδιατασσόμενο υλικό για την υλοποίηση των αλγορίθμων κρυπτογράφησης. Το βασικό πλεονέκτημά της είναι ότι χρησιμοποιεί δυναμικά περιφερειακά προκειμένου να υποστηρίξει την υλοποίηση πολλαπλών εναλλασσομένων αλγορίθμων. Η χρήση υλικού στην υλοποίηση ενός αλγορίθμου σε σχέση με την υλοποίησή του σε λογισμικό προσφέρει επιτάχυνση η οποία φαίνεται ιδιαίτερα στον αλγόριθμο TripleDES (επιτάχυνση x1250). Στην υλοποίηση του RTEA η διαφορά δεν είναι αντίστοιχα μεγάλη (επιτάχυνση x10) μιας και ο RTEA είναι σχεδιασμένος για να εκτελείται γρήγορα σε λογισμικό και μάλιστα σε επεξεργαστές πολύ χαμηλότερων επιδόσεων από τον PowerPC.

Από την άλλη πλευρά, η χρήση της ΔΕ οδήγησε σε ένα σύστημα που μπορεί να αξιοποιήσει καλύτερα τους διαθέσιμους πόρους του FPGA. Αυτό φαίνεται και από τη σύγκριση της παρούσας υλοποίησης με μια καθαρά στατική υλοποίηση του συστήματος. Στην υλοποίηση αυτή συνυπάρχουν τα δύο περιφερειακά κρυπτογράφησης ενώ δεν έχουν συμπεριληφθεί όποια είναι απαραίτητα μόνο για τη λειτουργία της ΔΕ (opb_dcr_socket, opb_hwicap, dcr_v29 και opb_to_dcr_bridge). Ο Πίνακας 5 παρουσιάζει τις απαιτήσεις σε πόρους του FPGA που έχει το δυναμικό καθώς και το πλήρως στατικό σύστημα.

	Τμήμα	*Virtex-4 Slices*	*Χρήση FPGA*
Υλοποίηση Δυναμικού Συστήματος	RTEA περιφερειακό	755	13%
	TripleDES περιφερειακό	2173	39%
	Στατικό τμήμα	1930	35%
	Μέγιστη χρήση (Στατικό τμήμα + TripleDES)	4103	74%
Στατική Υλοποίηση	Πλήρες Σύστημα	4437	81%

Πίνακας 5: Απαιτήσεις σε πόρους FPGA για τις δύο υλοποιήσεις

Συγκρίνοντας τα αποτελέσματα από τις δύο υλοποιήσεις παρατηρούμε ότι το όφελος σε πόρους του FPGA από τη χρήση της ΔΕ είναι περίπου 7%. Αν και η διαφορά αυτή δεν είναι ιδιαίτερα μεγάλη, είναι εμφανές ότι εάν η υλοποίηση του RTEA αλγορίθμου είχε απαιτήσεις σε χώρο αντίστοιχες με αυτές του TripleDES (~35% αντί για 13%) η στατική υλοποίηση θα ήταν ανέφικτη (~110%), αντίθετα με τη δυναμική. Έτσι, όχι μόνο θα εξακολουθούσε να υπάρχει χώρος στο FPGA διαθέσιμος για χρήση (~25%), αλλά στη δυναμική περιοχή θα μπορούσαν να τοποθετηθούν και άλλα περιφερειακά που να υλοποιούν διαφορετικούς αλγορίθμους.

Προκειμένου να συγκρίνουμε τις επιδόσεις των δύο υλοποιήσεων, χρησιμοποιήσαμε το εργαλείο Timing Analyzer. Ο Πίνακας 6 δείχνει τα αποτελέσματα που προέκυψαν.

	Τμήμα	*Περίοδος*	*Συχνότητα*
Υλοποίηση Δυναμικού Συστήματος	RTEA περιφερειακό	14,55 ns	68,69 MHz
	TripleDES περιφερειακό	15,11 ns	66,16 MHz
	Στατικό τμήμα	1,64 ns	609,01 MHz
	Στατικό τμήμα + RTEA περιφερειακό	16,88 ns	59,22 MHz
	Στατικό τμήμα + TripleDES περιφερειακό	17,63 ns	56,70 MHz
Στατική Υλοποίηση	Πλήρες Σύστημα	13,085ns	76,42 MHz

Πίνακας 6: Επιδόσεις για τις δύο υλοποιήσεις

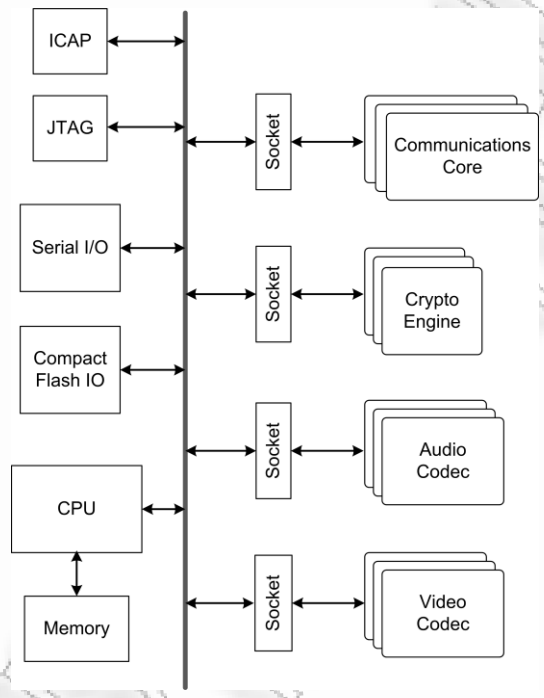
Από τα στοιχεία προκύπτει πτώση ταχύτητας κατά 26% ανάμεσα στα δύο συστήματα, στη χειρότερη περίπτωση (TripleDES). Πιθανές αιτίες για αυτό θεωρούμε τους περιορισμούς γεωμετρίας της δυναμικής περιοχής αλλά και την επιλογή των σημείων τοποθέτησης των bus macros. Η αρχιτεκτονική του FPGA που χρησιμοποιήθηκε επέβαλλε συγκεκριμένη περιοχή τοποθέτησης του δυναμικού τμήματος καθώς και συγκεκριμένο λόγο διαστάσεων των πλευρών του. Η τοποθέτηση των bus macros έχει αποδειχθεί πως επιφέρει σημαντικές αλλαγές στην απόδοση μιας υλοποίησης [9]. Επιπλέον θα είχαμε καλύτερα αποτελέσματα εάν καθοδηγούσαμε τα εργαλεία στην πορεία της υλοποίησης μέσω περιορισμών θέσης (location constraints) των διαφόρων μονάδων του συστήματος αλλά και συχνότητας ρολογιού (timing constraints).

Προσπαθώντας να συσχετίσουμε τα αποτελέσματα της υλοποίησής μας με τα πλεονεκτήματα της ΔΕ όπως αυτά περιγράφονται στην παράγραφο 1.2 παρατηρούμε ότι απεδείχθη στην πράξη η οικονομία χώρου που προσφέρει η ΔΕ. Αναφορικά με τις επιδόσεις παρατηρήθηκε πτώση ταχύτητας κατά 26% σε σχέση με την πλήρως στατική σχεδίαση αν και αυτό το νούμερο θα μπορούσε να βελτιωθεί με αλλαγές στην υλοποίηση (μετακίνηση δυναμικής περιοχής, θέσης των bus macros) και ορισμό κατάλληλων περιορισμών. Δυστυχώς η κάρτα ML403 δεν παρέχει τη δυνατότητα μέτρησης του ρεύματος που καταναλώνει το FPGA οπότε δε στάθηκε δυνατό να μετρήσουμε την κατανάλωση ισχύος έτσι ώστε να διαπιστωθούν οι διαφορές στατικής και δυναμικής υλοποίησης σε ό,τι αφορά την καταναλισκόμενη ισχύ.

Όπως προαναφέραμε, η χρήση του συγκεκριμένου FPGA (4VFX12) δημιούργησε περιορισμούς στη σχεδίαση. Έτσι, η ύπαρξη του ενσωματωμένου επεξεργαστή PowerPC ο οποίος είναι υλοποιημένος (hard macro) και τοποθετημένος στο αριστερό τμήμα του FPGA μαζί με τα Ethernet περιφερειακά επηρέασε τη θέση που δεσμεύτηκε για την δυναμική περιοχή και επέβαλλε τη χρήση εξωτερικής μνήμης με το αντίστοιχο τίμημα στην ταχύτητα εκτέλεσης της εφαρμογής λογισμικού. Αντιθέτως η χρήση ενός FPGA χωρίς τον ενσωματωμένο PowerPC (για παράδειγμα του αντίστοιχης χωρητικότητας 4VLX15 από την ίδια οικογένεια) θα προσέφερε μεγαλύτερη ευελιξία στη σχεδίαση αφού, αντί για τον PowerPC, θα επέτρεπε τη χρήση του επεξεργαστή microBlaze ο οποίος δεν είναι προ-τοποθετημένος.

6.2 Επεκτάσεις

Ένα σύστημα ασφαλούς επικοινωνίας που χρησιμοποιεί ΔΕ, θα μπορούσε να βασιστεί στο σύστημα που παρουσιάστηκε μετά από κάποιες τροποποιήσεις και επεκτάσεις. Για παράδειγμα θα μπορούσε να προστεθεί υποστήριξη για επιπλέον αλγορίθμους, ενώ θα μπορούσε να χρησιμοποιηθεί σειριακή γραμμή ή σύνδεση δικτύου (π.χ. Ethernet) για τη μετάδοση των δεδομένων. Επιπλέον, με χρήση audio και video codecs θα μπορούσε να δημιουργηθεί ένα σύστημα ασφαλούς τηλεδιάσκεψης. Το δομικό διάγραμμα μιας τέτοιας διάταξης φαίνεται στο Σχήμα 73. Σε μια τέτοια υλοποίηση, πέρα από τα τμήματα των αλγορίθμων κρυπτογράφησης, θα μπορούσαν να είναι δυναμικά και τα τμήματα που υλοποιούν τους codecs ή τους αλγορίθμους επικοινωνίας. Έτσι το τελικό σύστημα θα είναι προσαρμόσιμο σε διαφορετικές συνθήκες του περιβάλλοντος.



Σχήμα 73: Δομικό διάγραμμα επαναδιαμορφούμενου συστήματος επικοινωνίας

6.3 Η Δυναμική Επαναδιαμόρφωση ως τεχνολογία

Η υλοποίηση EAPR των εργαλείων της Xilinx για υποστήριξη δυναμικής επαναδιαμόρφωσης η οποία χρησιμοποιήθηκε έχει διάφορες απαιτήσεις και περιορισμούς στη σχεδίαση, κάτι που πολλές φορές κάνει τη χρήση της ιδιαίτερα χρονοβόρα. Επίσης αφήνει πολλές επιλογές υλοποίησης στο χρήστη χωρίς να του δίνει καμία καθοδήγηση πράγμα που μπορεί να καταλήξει σε μέτριας απόδοσης υλοποιήσεις. Σε πολλές περιπτώσεις κάποια από αυτά τα μειονεκτήματα εξαλείφονται στην τελευταία έκδοση του λογισμικού (v12.3) στην οποία η δυναμική επαναδιαμόρφωση υποστηρίζεται πλέον κανονικά, αν και απαιτεί επιπλέον άδεια σε σχέση με το υπόλοιπο πακέτο ISE.

Η δυναμική επαναδιαμόρφωση αναμένεται στο μέλλον να γίνει πολύ πιο κοινή σε εμπορικά συστήματα ξεφεύγοντας από τα όρια της έρευνας. Αυτός είναι και ο λόγος που, πέρα από τη Xilinx η οποία έχει πολυετή ενασχόληση με το αντικείμενο, άλλοι κατασκευαστές όπως η Altera [27] εισέρχονται στο συγκεκριμένο χώρο [28] προσφέροντας προϊόντα που υποστηρίζουν Δυναμική Επαναδιαμόρφωση.

Τέτοιες κινήσεις από τους κατασκευαστές είναι αναμενόμενες διότι πέρα από τα όποια προβλήματα έχει σήμερα στην υλοποίησή της, η Δυναμική Επαναδιαμόρφωση μπορεί να προσφέρει στα FPGA νέο πεδίο εφαρμογών καθώς και δυνατότητες διείσδυσης σε νέες αγορές. Για παράδειγμα η χρήση της Δυναμικής Επαναδιαμόρφωσης κάνει εφικτές εφαρμογές στις οποίες τόσο ο κατασκευαστής αλλά και ο τελικός χρήστης θα έχουν γνώση ενός διαφορετικού τμήματος του τελικού συστήματος και θα μπορούν πλέον να χρησιμοποιούν συστήματα με FPGAs χωρίς τον κίνδυνο διαρροής ευαίσθητης πληροφορίας.

Σε κάθε περίπτωση όμως, προκειμένου η ΔΕ να είναι εύχρηστη ως σχεδιαστική λύση θα πρέπει να βελτιωθούν τα εργαλεία σχεδίασης ώστε να οδηγούν σε καλύτερα αποτελέσματα χωρίς να υπάρχουν τόσες πολλές απαιτήσεις από την πλευρά του σχεδιαστή. Ήδη έχουν γίνει βήματα προς αυτήν την κατεύθυνση και στην τρέχουσα έκδοση των εργαλείων Xilinx έχει διευκολυνθεί πολύ η σχεδίαση με ΔΕ. Έτσι, για παράδειγμα, τα bus macros πλέον έχουν καταργηθεί ενώ το PlanAhead είναι πλέον πολύ πιο φιλικό προς το σχεδιαστή και υποστηρίζει και τη διαδικασία σύνθεσης. Παρόλα αυτά όμως υπάρχουν ακόμη πολλά περιθώρια βελτίωσης προκειμένου η ΔΕ να θεωρηθεί ώριμη τεχνολογία σχεδίασης συστημάτων με FPGAs.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Βιβλιογραφία

1. PACT: Reconfiguration on XPP-III Processors. (2006)
2. In: Xilinx Inc. Available at: http://www.xilinx.com
3. In: ATMEL Corp. Available at: http://www.atmel.com
4. Large Hadron Collider. In: Wikipedia. Available at: http://en.wikipedia.org/wiki/LHC
5. Xilinx honored for enabling technology in the ALICE experiment at CERN. In: EETimes News. Available at: http://www.eetimes.com/electronics-news/4183239/Xilinx-honored-for-enabling-technology-in-the-ALICE-experiment-at-CERN
6. Troger, G., Alme, J., Campagnolo, R., Kebschull, U., Lindenstruth, V., Musa, L., Roed, K., Rohich, D.: FPGA Dynamic Reconfiguration in ALICE and beyond. 11th Workshop on Electronics for LHC and future Experiments (2005)
7. Fons, F., Fons, M.: Making Biometrics the Killer App of FPGA Dynamic Partial Reconfiguration. Xilinx Xcell Journal(72) (Q3 2010)
8. Claus, C., Altenried, F., Stechele, W.: Dynamic Partial Reconfiguration of Xilinx FPGAs Lets Systems Adapt on the Fly. Xilinx Xcell Journal (Q1 2010)
9. Carver, J., Pittman, R., Forin, A.: Automatic bus macro placement for partially reconfigurable FPGA designs. In : International Symposium on Field Programmable Gate Arrays - FPGA'09, Monterey, California, USA , pp.269-272 (2009)
10. FPGA Logic Cells Comparison. In: Core Technologies. Available at: http://www.1-core.com/library/digital/fpga-logic-cells/
11. C-Language techniques for FPGA acceleration of embedded software. In: EETimes. Available at: http://www.embedded.com/columns/technicalsights/184419800?_requestid=216904
12. Xilinx: ML401/ML402/ML403 Evaluation Platform User Guide v2.5. (2006)
13. ModelSim, Advanced Simulation and Debugging. Available at: http://model.com/
14. ISE Simulator (ISim). In: Xilinx. Available at: http://www.xilinx.com/tools/isim.htm
15. Xilinx: Development System Reference Guide, v9.1i. (2006)
16. Xilinx: Partial Reconfiguration Early Access Lounge. In: Partial Reconfiguration Early Access Lounge. Available at: http://www.xilinx.com/support/prealounge/protected/index.htm
17. In: Virtual Box. Available at: http://www.virtualbox.org/
18. Cygwin. Available at: http://www.cygwin.com
19. Tumeo, A., Monchiero, M., Palermo, G., Ferrandi, F., Sciuto, D.: A Self Reconfigurable Implementation of the JPEG Encoder. Proceedings of the 18th International Conference on Application-specific Systems, Architectures and Processors, 24-29 (2007)
20. Hsiung, P.-A., Santambrogio, M., Huang, C.-H.: Reconfigurable System Design and Verification 1st edn. CRC Press (2009)
21. Xilinx, X.: XAPP290, Difference Based Partial Reconfiguration, Application Note. (2007)
22. Lysaght, P., Blodget, B., Mason, J., Young, J., Bridgford, B.: Enhanced Architectures, Design Methodologies and CAD Tools for Dynamic Reconfiguration of Xilinx FPGAs. In Press, I., ed. : Proceedings of the 16th IEEE International Conference on Field Programmable Logic & Applications (FPL2006), pp.1-6 (August 2006)
23. Coretex Systems, L.: Opencores.org TripleDES project. Available at: http://opencores.org/project,3des_vhdl
24. Belousov, O. In: OpenCores.org. Available at: http://opencores.org/project,rtea
25. Easy Triple-DES Sourceforge Project. In: Sourceforge. Available at: http://easy-triple-des.sourceforge.net/
26. SourceForge TeraTerm. In: Sourceforge.jp. Available at:

http://sourceforge.jp/projects/ttssh2/downloads/48772/teraterm-4.67.exe/

27. Altera Corp. In: Altera. Available at: http://www.altera.com
28. Altera to offer partial reconfiguration at 28-nm. In: EETimes. Available at: http://www.eetimes.com/electronics-products/fpga-pld-products/4114942/Altera-to-offer-partial-reconfiguration-at-28-nm
29. Tiny Encryption Algorithm. In: WikiPedia. Available at: http://en.wikipedia.org/wiki/Tiny_Encryption_Algorithm
30. Bobda, C.: Introduction to Reconfigurable Computing - Architectures, Algorithms and Applications. Springer, Dordrecht, The Netherlands (2007)

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ

Παράρτημα 1 - Εγκατάσταση εργαλείων Xilinx με υποστήριξη Δυναμικής Επαναδιαμόρφωσης

Η εγκατάσταση της έκδοσης 9.1 των εργαλείων της Xilinx σε περιβάλλον Ubuntu Linux με υποστήριξη ΔΕ απαιτεί μια σειρά βημάτων, τα οποία παρατίθενται στη συνέχεια.

1. Εγκατάσταση ISE 9.1 με το Service Pack 2, πχ στη θέση `/opt/Xilinx/ise91`
2. Στο αρχείο `.bashrc` κάθε χρήστη που θα επιτρέπεται να εκτελέσει το ISE 9.1 θα πρέπει να προστεθεί η δήλωση `source /opt/xilinx/ise91/settings.sh`
3. Εγκατάσταση EDK 9.1 με το Service Pack 1. Πριν την εγκατάσταση ο χρήστης root θα πρέπει να εκτελέσει την εντολή `source /opt/xilinx/ise91/settings.sh`
4. Το EDK χρησιμοποιεί την εφαρμογή gmake η οποία δεν είναι πάντοτε διαθέσιμη. Προκειμένου να αποφευχθούν προβλήματα στη φάση δημιουργίας εφαρμογών για PowerPC ή microBlaze, θα πρέπει να δημιουργηθεί ένας σύνδεσμος προς την εφαρμογή make:
   ```
   $ ln -s /usr/bin/make /usr/bin/gmake
   ```
5. Πριν την εγκατάσταση των αρχείων για υποστήριξη της δυναμικής επαναδιαμόρφωσης θα πρέπει να ελεγχθεί εάν η εφαρμογή xilperl μπορεί να εκτελεστεί. Προβλήματα μπορεί να εμφανιστούν εάν λείπει το αρχείο `libdb-4.1.so`. Σε αυτήν την περίπτωση, θα πρέπει να εγκατασταθεί η εφαρμογή libdb4.2 και να δημιουργηθεί ένας συνδεσμος προς αυτήν:
   ```
   $ apt-get install libdb4.2
   $ ln -sf /usr/lib/libdb-4.2.so /usr/lib/libdb-4.1.so
   ```
   Μετά από αυτά τα βήματα η εντολή `xilperl –help` θα πρέπει να μπορεί να εκτελεστεί κανονικά.
6. Αντίστοιχα προς την εγκατάσταση του ISE, υποθέτοντας ότι το EDK έχει εγκατασταθεί στη θέση `/opt/Xilinx/EDK` στο αρχείο .bashrc κάθε χρήστη που θα μπορεί να εκτελέσει το EDK 9.1 θα πρέπει να προστεθεί η δήλωση `source /opt/xilinx/EDK/settings.sh` αμέσως μετά την αντίστοιχη δήλωση για το ISE 9.1.
7. Τα εργαλεία για τη ΔΕ θα πρέπει να εγκατασταθούν με χρήση της εντολής:
   ```
   $ xilperl PRinstall.pl PRfiles.txt
   ```
8. Το PlanAhead θα πρέπει να εγκατασταθεί στη συνέχεια αλλά προηγουμένως θα πρέπει να έχουν τρέξει τα δύο αρχεία `settings.sh` ώστε να έχουν αρχικοποιηθεί οι μεταβλητές συστήματος που χρησιμοποιεί το ISE και το EDK. Αυτό μπορεί να επαληθευτεί με τη χρήση της εντολής:
   ```
   $ printenv | grep -i xilinx
   ```
9. Η εγκατάσταση του PlanAhead θα πρέπει να γίνει στη θέση `/opt/Xilinx/PlanAhead` ενώ στο αρχείο `.bashrc` κάθε χρήστη που θα επιτρέπεται να εκτελέσει το PlanAhead θα πρέπει να προστεθεί η εντολή `source /opt/xilinx/PlanAhead/PlanAhead/settings.sh`
10. Το PlanAhead δημιουργεί scripts τα οποία χρησιμοποιούν το πρόγραμμα csh. Εάν αυτό δεν είναι εγκατεστημένο στο σύστημα, η εκτέλεση θα αποτύχει και το PlanAhead θα σταματήσει απότομα κατά το βήμα *"Run PR Assemble ..."*. Για το λόγο αυτό θα πρέπει να εγκατασταθεί και το csh, κάτι που γίνεται με την εντολή:
    ```
    $ apt-get install csh
    ```

Παράρτημα 2 - Υλοποίηση Διαφορικής Μερικής Επαναδιαμόρφωσης (Difference-Based PR)

Η *Διαφορική Μερική Επαναδιαμόρφωση* (ΔΜΕ) επιτρέπει να γίνουν μικρές αλλαγές στη λειτουργικότητα ενός FPGA και να φορτωθούν αυτές και μόνο (αντί για το συνολικό bitstream) ενώ το υπόλοιπο FPGA παραμένει σε λειτουργία. Οι αλλαγές αυτές γίνονται συνήθως με τον FPGA Editor (ή εναλλακτικά με το εργαλείο Data2MEM όταν πρόκειται για αλλαγές στα περιεχόμενα των BlockRAMs). Κατόπιν εκτελείται το BitGen που δημιουργεί ένα διαφορικό bitstream το οποίο περιέχει μόνο τις αλλαγές ανάμεσα στο αρχικό και στο τελικό κύκλωμα.

Οι αλλαγές που μπορεί να γίνουν σε ένα κύκλωμα μπορεί να περιλαμβάνουν αλλαγές:

- στην εξίσωση που υλοποιεί ένας LUT
- στα περιεχόμενα μιας BlockRAM μνήμης
- στο I/O standard που έχει ορισθεί για κάποιον ακροδέκτη
- στις τιμές αρχικοποίησης ενός flip-flop
- στις αντιστάσεις πρόσδεσης (pullup/pulldown) που συνδέονται σε κάποιον ακροδέκτη

Η δημιουργία αρχείων διαμόρφωσης (bitstreams) για ΔΜΕ δε μπορεί να γίνει από το πρόγραμμα ISE, αλλά πρέπει να κληθεί το πρόγραμμα bitgen από την γραμμή εντολών. Για την διαδικασία της ΔΜΕ, απαιτείται η χρήση συγκεκριμένων παραμέτρων κατά κλήση του bitgen για την δημιουργία του bitstream. Συγκεκριμένα πρέπει να τεθούν οι παράμετροι:

`-g ActiveReconfig:Yes` έτσι ώστε το fpga να παραμένει σε λειτουργία κατά τη φόρτωση ενός (μερικού) bitstream. Σε αντίθετη περίπτωση η φόρτωση του bitstream θα έχει σαν αποτέλεσμα τη διακοπή της λειτουργίας του FPGA (οπότε και η έξοδος DONE απενεργοποιείται)

`-g Persist:Yes` (για το αρχικό bitstream) με την οποία δεν απενεργοποιείται το SelectMAP interface μετά τον αρχικό προγραμματισμό, έτσι ώστε να μπορεί να προγραμματιστούν και άλλες διαμορφώσεις

`-g Security:none` μιας και δεν υποστηρίζονται (έκδοση 9.1) ασφαλή bitstreams

Ο διακόπτης `-r` χρησιμοποιείται για την δημιουργία διαφορικών bitstreams.

Παράδειγμα χρήσης του bitgen έχει ως εξής:

```
bitgen -g ActiveReconfig:Yes -g Persist:Yes -r and_test.bit and_test2.ncd
and_test2_partial.bit
```

Στην περίπτωση αυτή το `and_test2_partial.bit` περιέχει τις αλλαγές προκειμένου να φορτωθεί το `and_test2` στη θέση του `and_test`. Για να δημιουργηθεί το αρχείο που θα επαναφέρει τη λειτουργικότητα του and_test πρέπει να εκτελεστεί η εξής εντολή:

```
bitgen -g ActiveReconfig:Yes -g Persist:yes -r and_test2.bit and_test.ncd
and_test_partial.bit
```

Περεταίρω λεπτομέρειες για τη διαδικασία της ΔΜΕ παρέχονται στο [21].

Παράρτημα 3 - Περιεχόμενα συνοδευτικού δίσκου

Στο συνοδευτικό δίσκο υπάρχουν οι εξής κατάλογοι:

`bus macros`	Περιέχει τα χρησιμοποιούμενα από το σύστημα bus macros
`edk`	Περιέχει την υλοποίηση του συστήματος επεξεργαστή όπως δημιουργείται από το EDK
`Reconf_App`	Περιέχει τα αρχεία για την εφαρμογή ελέγχου που εκτελείται στον PowerPC επεξεργαστή
`project_1`	Περιέχει τα αρχεία υλοποίησης του συστήματος στο PlanAhead
`top`	Περιέχει το ανώτερο επίπεδο της ιεραρχίας καθώς και το στατικό τμήμα που ελέγχει τα leds
`prr_modules`	Περιέχει τα δυναμικά περιφερειακά
`MyProcessorIPLib`	Περιέχει τα δυναμικά περιφερειακά που προέκυψαν από τον *Create & Import Peripherals Wizard*
`netlists`	Εκεί αντιγράφονται τα netlists μετά τη σύνθεση των δυναμικών περιφερειακών
`resources`	Περιέχει διάφορα βοηθητικά αρχεία που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος
`cf_image`	Περιέχει ένα bash script που εκτελεί τις εφαρμογές data2mem και xmd και δημιουργεί τα αρχεία για προγραμματισμό της Compact Flash
`drivers`	Περιέχει τον οδηγό για το hwicap (hwicap_v1_00_c) από το συνοδευτικό υλικό της Xilinx
`pcores`	Περιέχει το opb_dcr_socket περιφερειακό από το συνοδευτικό υλικό της Xilinx

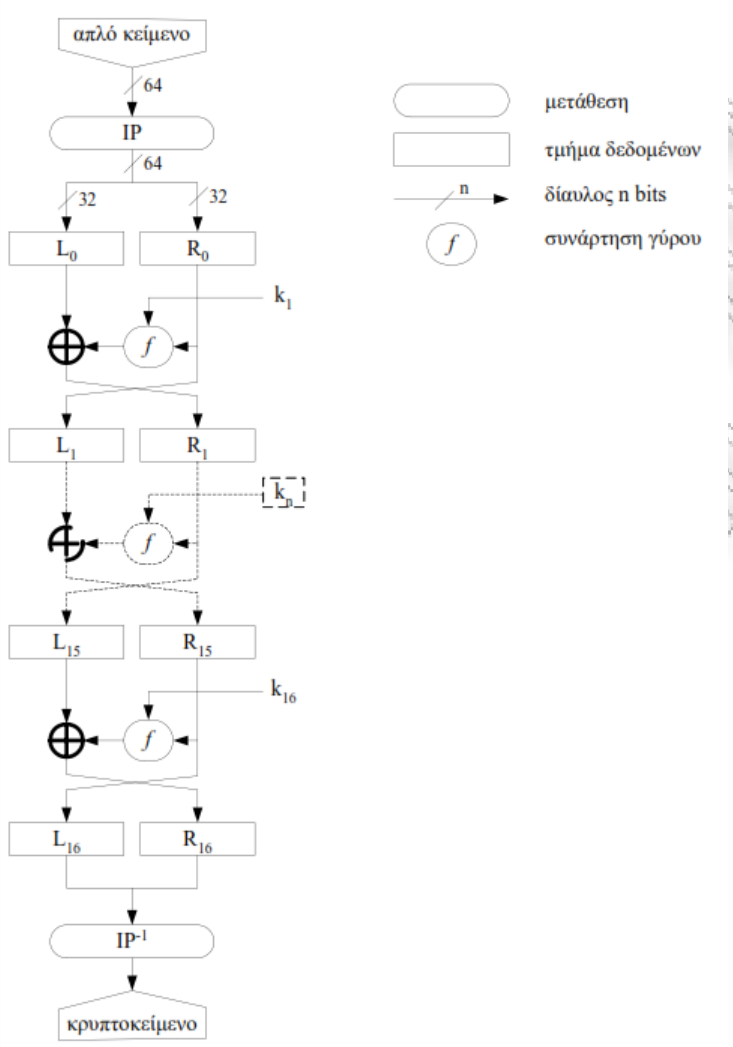
Παράρτημα 4 - Περιγραφή Αλγορίθμων Κρυπτογράφησης

Παρακάτω περιγράφονται οι δύο αλγόριθμοι κρυπτογράφησης που χρησιμοποιούνται από την παρούσα εφαρμογή, ο TripleDES και ο RTEA. Και στις δύο περιπτώσεις χρησιμοποιήθηκαν έτοιμες υλοποιήσεις ανοικτού κώδικα, τόσο για την εκδοχή του hardware (σε γλώσσα vhdl) όσο και για αυτή του software (σε γλώσσα C).

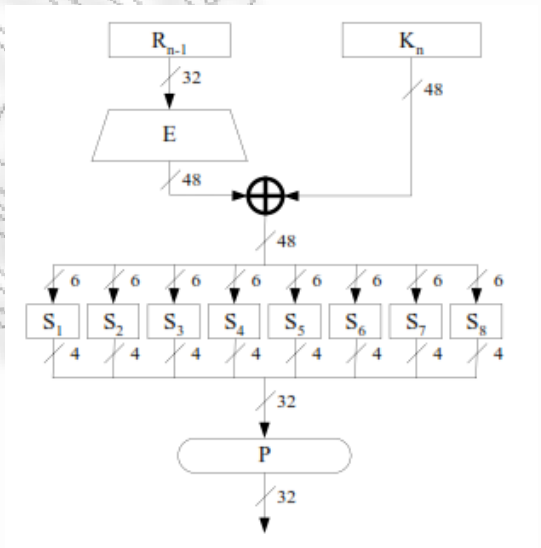
Π4.1 Ο αλγόριθμος TripleDES

Ο αλγόριθμος TripleDES βασίζεται στον DES τον οποίο και χρησιμοποιεί διαδοχικά τρεις φορές προκειμένου να αυξήσει το παρεχόμενο επίπεδο ασφαλείας. Ο DES αλγόριθμος προτάθηκε αρχικά το 1974 ως μέθοδος κρυπτογράφησης κυβερνητικών εγγράφων στις Η.Π.Α. από την IBM ενώ αργότερα διάφοροι κρατικοί οργανισμοί και τράπεζες τον υιοθέτησαν προκειμένου να κρυπτογραφήσουν εσωτερική πληροφορία.

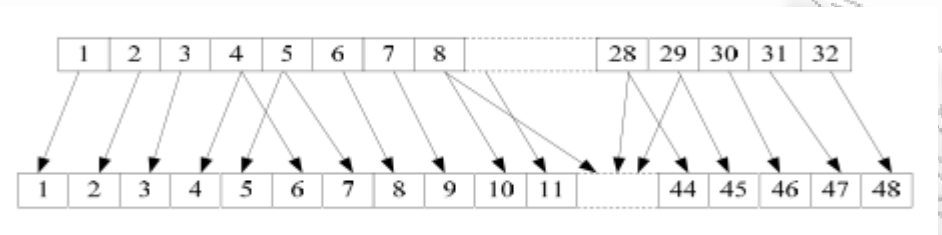
Ο αλγόριθμος DES είναι συμμετρικός αλγόριθμος κρυπτογράφησης που επεξεργάζεται δεδομένα σε ομάδες των 64 bits με χρήση δομών Feistel. Για την λειτουργία της κρυπτογράφησης ο αλγόριθμος DES χρησιμοποιεί «κλειδιά» κρυπτογράφησης επίσης μεγέθους 64 bits. Στην περίπτωση που τα δεδομένα εισόδου είναι λιγότερα από 64 bits, στο τέλος τους προστίθενται τόσα μηδενικά ώστε να συμπληρωθούν τα 64 bit (τεχνική zero-padding), ενώ αν είναι περισσότερα από 64 bits τότε αυτό χωρίζεται σε ομάδες των 64 bits. Αν είναι απαραίτητο, στην τελευταία ομάδα χρησιμοποιείται zero-padding ώστε να αποτελείται και αυτή από 64 bits. Μία γενική σχηματική περιγραφή του αλγορίθμου DES φαίνεται στο Σχήμα 74 ενώ στο Σχήμα 75 παρουσιάζεται η δομή των Feistel συναρτήσεων (F) που αποτελούν βασικό δομικό χαρακτηριστικό του αλγορίθμου. Στο Σχήμα 76 παρουσιάζεται η συνάρτηση επέκτασης ενώ στο Σχήμα 77 το πρόγραμμα του κλειδιού.



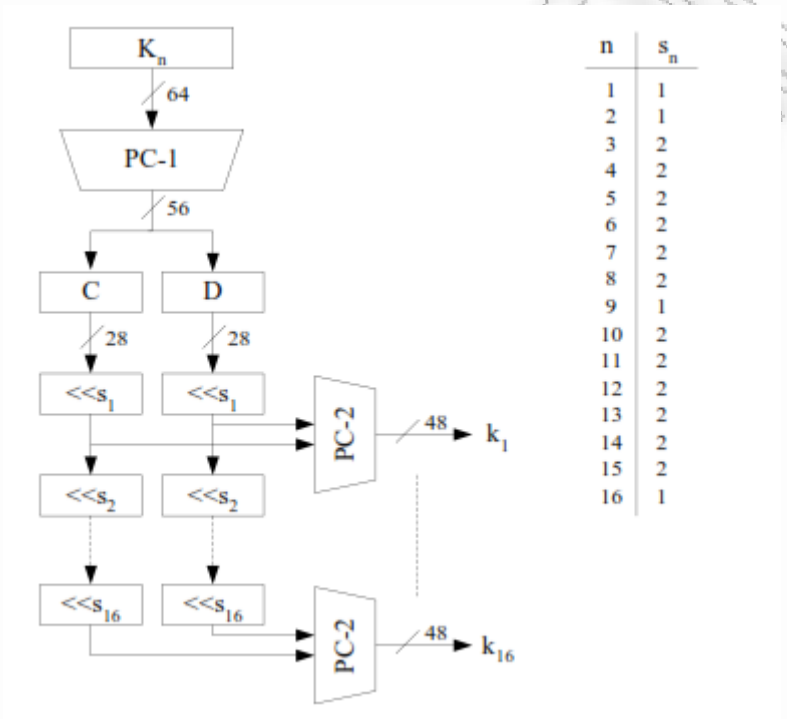
Σχήμα 74: Δομή του Αλγορίθμου DES



Σχήμα 75: Δομή των Συναρτήσεων Feistel



Σχήμα 76: Συνάρτηση Επέκτασης



Σχήμα 77: Πρόγραμμα του Κλειδιού

Συνοπτικά, ο DES αλγόριθμος λειτουργεί ως εξής:

Στην αρχική είσοδο εφαρμόζεται η αρχική μετάθεση (IP), μια ακολουθία 16 γύρων τύπου Feistel, και η τελική μετάθεση (IP^{-1}). Μιας και ο αλγόριθμος είναι συμμετρικός, η ίδια διαδικασία εφαρμόζεται και για την αποκρυπτογράφηση, με μόνη διαφορά ότι το πρόγραμμα κλειδιού της αποκρυπτογράφησης παράγει την αντίστροφη ακολουθία από αυτήν του προγράμματος κλειδιού της κρυπτογράφησης.

Η αρχική μετάθεση είναι η αντιστροφή της τελικής μετάθεσης και υλοποιείται με χρήση ενός πίνακα αναζήτησης (look-up table). Η συνάρτηση γύρου F αποτελείται από μια συνάρτηση επέκτασης E, οκτώ κουτιά αντικατάστασης S και μια τελική συνάρτηση μετάθεσης P των 32 bits, όπως φαίνεται στο Σχήμα 75. Η συνάρτηση επέκτασης είναι μια μετάθεση στην οποία ορισμένα bits της εισόδου εμφανίζονται σε περισσότερες από μια θέσεις στην έξοδο, όπως φαίνεται στο Σχήμα 76. Κατ' αυτόν τον τρόπο ένα διάνυσμα 32 bit επεκτείνεται στα 48 bit. Τα οκτώ κουτιά αντικατάστασης S αποτελούν το ακρογωνιαίο λίθο του DES καθώς η κρυπτογραφική δύναμη του κρυπταλγόριθμου εξαρτάται άμεσα από αυτά μιας και εισάγουν μη γραμμικότητα στην κατασκευή.

Η είσοδος της συνάρτησης γύρου απαιτεί 32 bits δεδομένων και 48 bits του κλειδιού. Σε κάθε γύρο το κλειδί προκύπτει από το πρόγραμμα κλειδιού όπως φαίνεται στο Σχήμα 77. Το πρόγραμμα κλειδιού αποτελείται από δύο συναρτήσεις μετάθεσης επιλογής (permuted choice, PC-1 και PC-2), καθώς και από

καταχωρητές ολίσθησης. Η μετάθεση επιλογής είναι μια μετάθεση κατά την οποία ορισμένα bits της εισόδου αγνοούνται και δεν εμφανίζονται στην έξοδο. Η PC-1 ξεχωρίζει όλα τα bits του αρχικού κλειδιού που θα συμμετάσχουν στο πρόγραμμα κλειδιού για τη δημιουργία των επιμέρους κλειδιών. Έτσι το κάθε όγδοο bit του αρχικού κλειδιού αγνοείται, με αποτέλεσμα να διατίθενται για την κρυπτογράφηση 64 – 8 = 56 bits. Στη συνέχεια, τα 56 bits μοιράζονται σε δύο λέξεις των 28 bits και τροφοδοτούνται σε καταχωρητές ολίσθησης. Ανάλογα με τον γύρο, πραγματοποιείται μια ολίσθηση η προκαθορισμένων θέσεων όπως δείχνει ο πίνακας στο Σχήμα 77. Κατά την κρυπτογράφηση η ολίσθηση πραγματοποιείται προς την αριστερή φορά, ενώ κατά την αποκρυπτογράφηση η ολίσθηση εκτελείται προς τη δεξιά φορά. Ο αριθμός των ολισθήσεων έχει ως αποτέλεσμα το κλειδί να επανέλθει στην αρχική του θέση. Έτσι κατά την αποκρυπτογράφηση εκτελούνται οι αντίστροφες (δεξιές) ολισθήσεις, με αρχική τη μηδενική ολίσθηση, δηλαδή: 0, 1, 2, 2, …, 1.

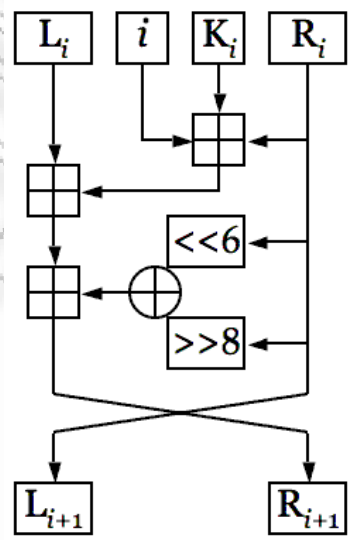
Κατά την ολοκλήρωση της κάθε ολίσθησης επιλέγονται 48 από τα 56 bits που βρίσκονται στον καταχωρητή ολίσθησης σύμφωνα με την μετάθεση επιλογής PC-2. Οι μεταθέσεις PC-1 και PC-2 γίνονται βάσει πινάκων αναζήτησης.

Ο TripleDES αλγόριθμος αποτελείται από τρεις διαδοχικές επαναλήψεις του DES, όπου κατά την κρυπτογράφηση το αρχικό κείμενο κρυπτογραφείται βάσει ενός κλειδιού K1, κατόπιν αποκρυπτογραφείται βάσει ενός κλειδιού K2 και κατόπιν κρυπτογραφείται και πάλι βάσει ενός κλειδιού K3, ενώ η αντίστροφη διαδικασία ακολουθείται κατά την αποκρυπτογράφηση. Υπάρχουν 3 πιθανές υλοποιήσεις του TripleDES σε σχέση με τα κλειδιά K1, K2, K3:

- Εάν K1=K2=K3 καταλήγουμε σε υλοποίηση συμβατή με το DES αλγόριθμο
- Εάν K1=K3, K1≠K2 έχουμε υλοποίηση με 112-bit κλειδί
- Εάν K1≠K2≠K3 έχουμε την πιο ισχυρή εκδοχή, με 168-bit κλειδί.

Π4.2 Ο αλγόριθμος RTEA

Ο αλγόριθμος RTEA (Ruptor's TEA ή Repaired TEA) [24] είναι ένας κρυπτογραφικός αλγόριθμος που αναπτύχθηκε σαν επέκταση του αλγορίθμου TEA (Tiny Encryption Algorithm) και ακολουθεί την ίδια φιλοσοφία σχεδίασης ενός γρήγορου αλγορίθμου που θα μπορεί να εκτελεστεί αποδοτικά ακόμα και σε έναν μικροεπεξεργαστή 8-bit. Βασίζεται σε δομή Feistel και για τη λειτουργία του χρησιμοποιεί λέξεις 32bit. Είναι πολύ απλούστερος και ταχύτερος από τους TEA και XTEA ενώ δεν έχει το πρόβλημα ασφάλειας που παρουσιάζει ο TEA [29]. Σε κάθε επανάληψη του RTEA υλοποιείται η συνάρτηση που φαίνεται στο Σχήμα 78.



Σχήμα 78: Κύκλος επανάληψης του αλγορίθμου RTEA