ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

Σχεδίαση κυκλωμάτων επικοινωνίας με απλές οθόνες, με τη γλώσσα VHDL και υλοποίηση στις αναπτυξιακές πλακέτες LP-2900 και DE2.

Πτυχιακή Εργασία της

Βουρδόγλου Γεωργίας (3431)

Επιβλέπων: Δρ. Καλόμοιρος Ιωάννης, Επίκ. Καθηγητής

ΣΕΡΡΕΣ, ΙΟΥΝΙΟΣ 2016

Υπεύθυνη Δήλωση : Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής Τ.Ε του Τ.Ε.Ι. Κεντρικής Μακεδονίας.

Περίληψη

Η VHDL είναι μια από τις πλέον διαδεδομένες γλώσσες περιγραφής υλικού (HDL) και χρησιμοποιείται για την πρότυπη περιγραφή της συμπεριφοράς απλών ελεγκτών, ψηφιακών υπολογιστικών κυκλωμάτων και μεγάλων συστημάτων σε ψηφίδα (SoC). Τα κυκλώματα αυτά συνδέονται συχνά με οθόνες απεικόνισης λογικών καταστάσεων, απλών ψηφίων ή και χαρακτήρων. Τέτοιες οθόνες είναι οι απλές συστοιχίες LEDs, οι απεικονίσεις επτά τομέων (SSD) και οι οθόνες LCD.

Σκοπός της πτυχιακής εργασίας είναι η μελέτη του υλισμικού (hardware) που σχετίζεται με το καθένα από τα παραπάνω κυκλώματα απεικόνισης, καθώς και των αναγκών σε σήματα οδήγησης που έχουν οι παραπάνω διατάξεις. Γι' αυτό αναπτύχθηκε κώδικας για τη δημιουργία κατάλληλων ελεγκτών για την οδήγηση των οθονών, με τη βοήθεια του Quartus II. Οι ελεγκτές λαμβάνουν είσοδο από μία διάταξη διακοπτών και προβάλλουν το αντίστοιχο αριθμητικό αποτέλεσμα σε μία συστοιχία LEDs (δυαδικό), σε μία απεικόνιση επτά τομέων (δεκαεξαδικό) και σε μία οθόνη LCD (αλφαριθμητικό). Τέλος, γίνεται αναφορά στις αναπτυξιακές πλακέτες LP-2900 και DE2, οι οποίες χρησιμοποιούνται στην εργασία, και αναπτύσσονται τα χαρακτηριστικά τους.

Περιεχόμενα

Ευρετήριο Σχημάτων

Πρόλογος

Σκοπός της πτυχιακής εργασίας που ακολουθεί είναι η σχεδίαση σε γλώσσα VHDL (γλώσσα περιγραφής υλικού) κυκλωμάτων επικοινωνίας με οθόνες απεικόνισης λογικών καταστάσεων και η υλοποίησή τους σε FPGA (Διατάξεις Πυλών Προγραμματιζόμενες στο Πεδίο) μέσω των αναπτυξιακών πλακέτων LP-2900 και DE2. Σε αυτές τις οθόνες ανήκουν οι απλές συστοιχίες LEDs, οι απεικονίσεις επτά τομέων (SSD) και οι οθόνες LCD.

Αρχικά, στο κεφάλαιο 1 θα γίνει μία αναφορά στον ορισμό και την ιστορία της γλώσσας περιγραφής υλικού VHDL. Καθώς και μία γενική εισαγωγή στη διαδικασία της σχεδίασης ψηφιακών συστημάτων περιγράφοντας τα βασικά βήματα της σχεδίασης και πραγματοποιώντας τα βέλτιστα για την παραγωγή ενός επιθυμητού τελικού κυκλώματος χρησιμοποιώντας τα εργαλεία CAD. Ακόμη, περέχεται μία πλήρη σύνοψη των χαρακτηριστικών της γλώσσας VHDL.

Στη συνέχεια, το κεφάλαιο 2 δίνει μία αναλυτική παρουσίαση των σύγχρονων ακολουθιακών κυκλωμάτων (μηχανές πεπερασμένων καταστάσεων) και εξηγεί τη συμπεριφορά των κυκλωμάτων αυτών. Τα κατηγοριοποιεί και περιγράφει το σχεδιασμό τους.

Στο κεφάλαιο 3 περιγράφονται οι λογικές διατάξεις πυλών προγραμματιζόμενες στο πεδίο (FPGA). Αναλύεται η γενική δομή μιας διάταξης FPGA, τα μέρη από τα οποία αποτελείται (ακροδέκτες εισόδου / εξόδου, λογικές βαθμίδες, γραμμές διασύνδεσης) καθώς και ο προγραμματισμός ενός FPGA.

Στα δύο επόμενα κεφάλαια γίνεται μία αναφορά στις αναπτυξιακές πλακέτες LP-2900 και DE2, οι οποίες χρησιμοποιούνται για την υλοποίηση της πτυχιακής εργασίας, περιγράφοντας το σκοπό τους αλλά και από τι αποτελείται η καθεμιά.

Στο κεφάλαιο 6 ακολουθεί μια αναλυτική περιγραφή των οθονών απεικόνισης λογικών καταστάσεων. Περιγράφεται τόσο η γενική δομή τους, όσο και ο τρόπος λειτουργίας της καθεμιάς ξεχωριστά.

Τέλος, στο κεφάλαιο 7 παρουσιάζεται ο κώδικας για την οδήγηση οθονών και η επεξήγησή του. Όπως, επίσης, περιγράφεται βήμα-βήμα η πρόσομοίωσή του με τη βοήθεια του εργαλείου σχεδίασης Quartus II της εταιρείας ALTERA.

Κεφάλαιο 1 :
Χαρακτηριστικά της γλώσσας VHDL

1.1 Εισαγωγή στη γλώσσα VHDL

Η VHDL είναι μία γλώσσα περιγραφής υλικού που χρησιμοποιείται στον αυτόματο σχεδιασμό ηλεκτρονικών σχεδιάσεων για την περιγραφή και ανάπτυξη ολοκληρωμένων ψηφιακών κυκλωμάτων και συστημάτων. Ο όρος VHDL είναι συντόμευση των λέξεων VHSIC Hardware Description Language, όπου VHSIC σημαίνει Very High Speed Integrated Circuit. Αρχικά σκοπός της ήταν να αποτελέσει μια γλώσσα κειμένου, η οποία να περιγράφει τη δομή περίπλοκων ψηφιακών κυκλωμάτων και να παρέχει δυνατότητες μοντελοποίησης της συμπεριφοραστων ψηφιακων κυκλωμάτων. Έτσι μπορούσε να χρησιμοποιηθεί ως είσοδος σε προγράμματα λογισμικού που προσομοίωναν τη συμπεριφορά των ψηφιακών κυκλωμάτων. Αργότερα, πέρα από τη χρήση της για καταγραφή και προσομοίωση, η γλώσσα VHDL χρησιμοποιήθηκε επίσης ως εργαλείο σχεδίασης σε συστήματα σχεδίασης CAD (Computer Aided Design). Κατά τη σύνθεση, ο μεταγλωτιστής συνθέτει ένα πραγματικό κύκλωμα που ανταποκρίνεται με ακρίβεια στη λογική και χρονική περιγραφή, την οποία μοντελοποιεί ο κώδικας.

Η γλώσσα VHDL δημιουργήθηκε στα πλαίσια της προσπάθειας ανάπτυξης μιας προτότυπης σχεδίασης ψηφιακών κυκλωμάτων κατά τη δεκαετία του 1980 με χρηματοδότηση από το υπουργείο Άμυνας των ΗΠΑ. Το αρχικό πρότυπο της γλώσσας υιοθετήθηκε το 1987 από το ινστιτούτο IEEE και ονομάστηκε IEEE 1076. Μια αναθεωρημένη εκδοχή του υιοθετήθηκε το 1993 και ονομάστηκε IEEE 1164. Εκτός από το πρότυπο 1164 εμφανίστηκαν πολλά θυγατρικά πρότυπα που επέκτειναν τη λειτουργηκότητα της γλώσσας. Αργότερα ακολούθησαν και νεότερες αναβαθμίσεις του προτύπου με μικρές αλλαγές, όπως η IEEE-2002 και η IEEE-2008.

Ο κώδικας VHDL αποτελεί τον βασικό τύπο αρχείου που δέχονται ως είσοδο τα λογισμικά ψηφιακής σχεδίασης κυκλωμάτων (Computer Aided Design ή CAD), για τη δημιουργία σύνθετων ολοκληρωμένων κυκλωμάτων. Η γλώσσα VHDL χρησιμοποιείται ευρύτατα για την περιγραφή και υλοποίηση ψηφιακών συστημάτων σε προγραμματιζόμενες λογικές διατάξεις, τύπου CPLDs (Complex Programmable Logic Devices-Σύνθετες προγραμματιζόμενες λογκές διατάξεις) και FPGAs (Field

Programmable Gate Arrays-(ιατάξεις πυλών προγραμματιζόμενες στο πεδίο). Επίσης, έχει καθιερωθεί σαν ένα πρότυπο (standard) στη σχεδίαση ηλεκτρονικών κυκλωμάτων ASICs (Application Specific Integrated Circuits). Η ευρεία χρήση και καθιέρωση της γλώσσας αυτής, ως πρότυπο, μας βεβαιώνει ότι και οι επόμενες εκδόσεις εργαλείων σχεδίασης θα υποστηρίζουν το πρότυπο αυτό. Έτσι, μελλοντικά θα είναι δυνατή η μεταφερσιμότητα σε νέα εργαλεία σχεδίασης, με ελάχιστες ή καθόλου αλλαγές ενός κυκλώματος που περιγράφηκε και αναπτύχθηκε με τα σημερινά εργαλεία σχεδίασης.

Ας αναφερθεί ότι εκτός από τη γλώσσα VHDL, πρότυπο του ινστιτούτου IEEE αποτελεί και η γλώσσα περιγραφής υλικού Verilog. Η οποία έχει επίσης ευρύτατη χρήση και αποδοχή από τη βιομηχανία διεθνώς.

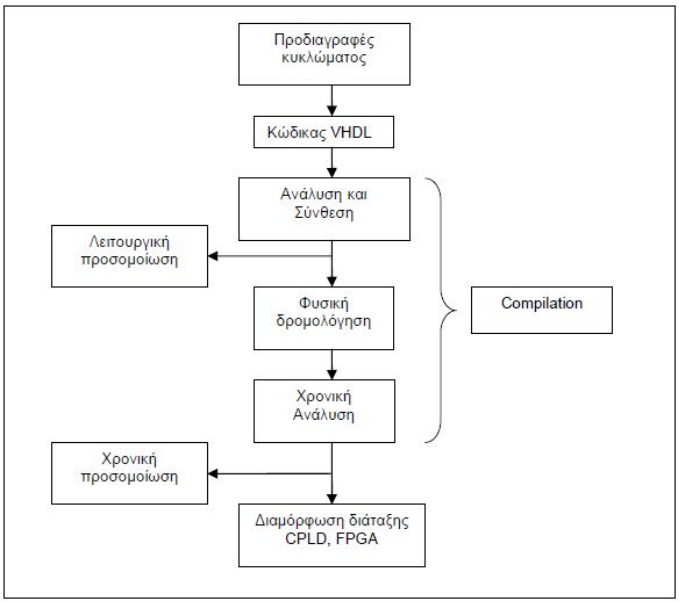
1.2 Εισαγωγή σχεδίασης

Για τη σχεδίαση ενός λογικού κυκλώματος χρειάζεται ένας αριθμός εργαλείων CAD, που βρίσκονται συνήθως μαζί με τη μορφή ενός συστήματος σχεδίασης CAD. Αυτό περιλαμβάνει εργαλεία για την εκτέλεση των λειτουργιών της εισαγωγής σχεδίασης, της σύνθεσης και βελτιστοποίησης, της προσομοίωσης και της φυσικής σχεδίασης. Το σημείο έναρξης της σχεδίασης ενός λογικού κυκλώματος είναι η κατανόηση της λειτουργίας που θα πρέπει να επιτελεί και η μορφοποίηση της γενικής δομής του. Αρχικά η διαδικασία αυτή περιλαμβάνει την εισαγωγή της περιγραφής του κυκλώματος που πρόκειται να σχεδιαστεί στο πρόγραμμα CAD. Το στάδιο αυτό ονομάζεται εισαγωγή σχεδίασης (entry design). Υπάρχουν δύο μέθοδοι εισαγωγής σχεδίασης: η μέθοδος σχηματικών διαγραμμάτων και η μέθοδος γραφης πηγαίου κώδικα.

Η εισαγωγή σχεδίασης με τη μέθοδο σχηματικού διαγράμματος αναφέρεται στη δημιουργία λογικού κυκλώματος σχεδιάζοντας λογικές πύλες και συνδέοντάς τες με καλώδια. Η λέξη σχηματικό αναφέρεται σε ένα διάγραμμα κυκλώματος στο οποίο τα στοιχεία του κυκλώματος, όπως οι λογικές πύλες, απεικονίζονται σαν γραφικά σύμβολα και οι συνδέσεις μεταξύ των κυκλωματικών στοιχείων σχεδιάζονται με γραμμές. Το εργαλείο σχηματικού διαγράμματος χρησιμοποιεί τις δυνατότητες γραφικής απεικόνισης ώστε να σχεδιάσει ένα σχηματικό διάγραμμα. Για την εισαγωγή λογικών πυλών στο διάγραμμα, το εργαλείο παρέχει μια συλλογή από

γραφικά σύμβολα που παριστάνουν λογικές πύλες διαφόρων τύπων με διάφορους αριθμούς εισόδων. Η συλλογή αυτή συμβόλων ονομάζεται βιβλιοθήκη (library) και το εργαλείο παρέχει ένα γραφικό τρόπο σύνδεσης μεταξύ τους, ώστε να δημιουργηθεί ένα λογικό κύκλωμα. Τα υποκυκλώματα που έχουν δημιουργηθεί μπορούν να παρασταθούν ως γραφικά σύμβολα και να συμπεριληφθούν στο τρέχον σχηματικό διάγραμμα. Η τεχνική αυτή ονομάζεται ιεραρχική σχεδίαση (hierarchical design) και αποτελεί έναν καλό τρόπο διαχείρισηςτων πολύπλοκων κυκλωμάτων.

Άλλος τρόπος εισαγωγής σχεδίασης είναι η μέθοδος γραφής πηγαίου κώδικα. Μία Γλώσσα Περιγραφής Κυκλωμάτων (Hardware Description Language, HDL) μοιάζει με μία γλώσσα προγραμματισμού υπολογιστών, με τη διαφορά ότι χρησιμοποιείται για να περιγράψει κύκλωμα. Σήμερα υπάρχουν πολλές γλώσσες HDL όμως η γλώσσα που υποστηρίζεται από όλες τις κατασκευαστικές εταιρείες που παρέχου ψηφιακό εξοπλισμό και υιοθετείται επίσημα από το Ινστιτούτο Ηλεκτρολόγων Μηχανικών (Institute of Electrical and Electronic Engineers, IEEE) ως πρότυπο είναι η γλώσσα VHDL. Η εισαγωγή σχεδίασης ενός λογικού κυκλώματος πραγματοποιείται γράφοντας κάποιο πρόγραμμα, ή όπως λέγεται κάποιο κώδικα (code) στη γλώσσα VHDL. Στο Σχήμα 1.1 φαίνεται η πλήρης ροή των εργασιών κατά τη σχεδίαση με κώδικα VHDL.



Σχήμα 1.1 Διάγραμμα ροής της σχεδίασης με κώδικα VHDL

Για τη συγγραφή του κώδικα χρησιμοποιείται συνήθως ο ASCII Editor που ενσωματώνεται στα εργαλεία σχεδίασης. Ο κώδικας αποθηκεύεται με επέκταση *.vhd*.

Τα σήματα που διαβιβάζονται στα κυκλώματα παριστάνονται ως ματαβλητές στον πηγαίο αυτό κώδικα και οι λογικές συναρτήσεις εκφράζονται ως δηλώσεις τιμών σε αυτές τις μεταβλητές. Ο πηγαίος κώδικας έχει τη μοφή απλού κειμένου και έτσι υπάρχει η δυνατότητα πρόσθεσης συνοδευτικών κειμένων για την καλύτερη κατανόηση της λειτουργίας του. Για τη σχεδίαση μικρών κυκλωμάτων η περιγραφή αποτελείται από ένα αρχείο, ενώ για μεγαλύτερα, από πολλά αρχεία (design units). Έτσι και στη γλώσσα VHDL η εγγραφή του κώδικα γίνεται με τέτοιο τρόπο, ώστε η σχεδίαση να είναι ιεραρχική. Το ανώτερο ιεραρχικά αρχείο ονομάζεται «οντότητα ανώτερου επιπέδου» (top-level entity), ενώ τα υπόλοιπα αρχεία συνδέονται με αυτό μέσω κατάλληλων δηλώσεων και αναφορών. Το σύνολο των αρχείων σχεδίασης ενός συστήματος, όπως και αυτά που δημιουργούνται αργότερα για την προσομοίωση και τη διαμόρφωση (configuration), αποτελούν ένα ολοκληρωμένο project μέσα στο περιβάλλον σχεδίασης και αποθηκεύονται σε κοινό φάκελο εργασίας.

Μετά τη συγγραφή του κώδικα ακολουθεί η διαδικασία της μετάφρασης ή μεταγλώττισης (compiling) κώδικα VHDL που αποτελεί μέρος της σύνθεσης. Το πρώτο στάδιο της μεταγλώττισης είναι η ανάλυση. Κατά την οποία γίνεται επεξεργασία του κώδικα για συντακτικά λάθη και επιστρέφει σχόλια που οδηγούν το χρήστη στη διόρθωσή τους. Επίσης ο compiler ελέγχει για τη συμβατότητα του με τμήματα του κώδικα από τον οποίο εξαρτάται.

Στη συνέχεια ακολουθεί η σύνθεση (synthesis) κατά την οποία ο compiler δημιουργεί ένα λογικό κύκλωμα από μία αρχική προδιαγραφή που δίνεται είτε με τη μορφή σχηματικού διαγράμματος είτε με τη μορφή κώδικα. Η σύνθεση αποτελεί την πιο σημαντική διαδικασία στη συνολική ροή. Το αποτέλεσμα της σύνθεσης διαφοροποιείται ανάλογα με το κύκλωμα που έχει να διαμορφώσει διότι κάθε κύκλωμα περιέχει τις δικές του διαφορετικές βαθμίδες για τη δημιουργία λογικών συναρτήσεων. Ένα κύκλωμα CPLD θα πρέπει να υλοποιήσει τις λογικές συναρτήσεις με βάση λογικούς πίνακες πυλών AND και OR (Logic Arrays), ενώ ένα κύκλωμα FPGA χρησιμοποιεί τους λεγόμενους πίνακες αναφοράς (Look-up Tables-LUTs). Η σύνθεση έχει ως αποτέλεσμα τη λειτουργική προσομοίωση κατά την οποία το κύκλωμα που παρίσταται με τη μορφή λογικών εκφράσεων μπορεί να προσομοιωθεί για να επαληθευτεί ότι θα λειτουργήσει όπως αναμένεται. Η λειτουργία του κυκλώματος ελέγχεται ανεξάρτητα από το χρόνο. Τόσο οι καθυστερήσεις στις πύλες όσο και όλες οι υπόλοιπες παράμετροι του χρόνου θεωρούνται μηδέν.

Μετά τη λογική σύνθεση το επόμενο βήμα στο διάγραμμα ροής της σχεδίασης είναι να καθοριστεί με ακρίβεια το πως θα υλοποιηθεί το κύκλωμα σε ένα δεδομένο ολοκληρωμένο κύκλωμα. Το βήμα αυτό είναι η διαδικασία της προσαρμογής (fitting), η οποία λέγεται αλλιώς και δρομολόγηση (place and route). Κάθε λογική δομή η οποία έχει παραχθεί από τη σύνθεση βρίσκει τη φυσική της αντιστοίχηση σε μια λογική βαθμίδα μέσα στη διάταξη. Αυτό έχει ως αποτέλεσμα τον υπολογισμό των χρονικών καθυστερήσεων των πυλών καθώς και τις χρονικές απαιτήσεις σειριακών κυκλωμάτων, όπως τα flip-flops. Έτσι το επόμενο στάδιο μετά την φυσική δρομολόγηση είναι η χρονική προσομοίωση (timing simulation) της σχεδίασης.

Αφού εξακριβωθεί ότι το σχεδιασμένο κύκλωμα ικανοποιεί τις απαιτήσεις των προδιαγραφών, εφόσον τα παραπάνω στάδια έχουν υλοποιηθεί με επιτυχία, διαμορφώνεται η διάταξη-στόχος , καταφορτώνοντας το αρχείο διαμόρφωσης που προκύπτει από το τελικό στάδιο της. Μετά τη διαμόρφωση (configuration) η διάταξη μπορεί να δεχτεί σήματα εισόδου και να παράγει τα αναμενόμενα σήματα εξόδου. Το στάδιο αυτό ονομάζεται οργάνωση ολοκληρωμένου κυκλώματος ή προγράμματος.

1.3 Εργαλεία λογισμικού για σχεδίαση με τη γλώσσα VHDL

Υπάρχουν πολλά εργαλεία (Electronic Design Automation-EDA-tools) λογισμικού για τη σχεδίαση με γλώσσα VHDL. Αυτά που χρησιμοποιούνται όμως πιο συχνά στην ηλεκτρονική βιομηχανία για την αυτοματοποίηση της σχεδίασης λογικών κυκλωμάτων είναι:

- Quartus II της Altera, που είναι ολοκληρωμένο εργαλείο σύνθεσης, γραφικής προσομοίωσης και προγραμματισμού
- ISE της Xilinx., που αποτελείται από το XST για σύνθεση και το IDE Simulator για
  προσομοίωση
- Leonardo Spectrum της Mentor Graphics, για σύνθεση
- ModelSim της Mentor Graphics για προσομοίωση

1.4 Βασικά χαρακτηριστικά της γλώσσας VHDL

Η VHDL διαφέρει από τις συμβατικές γλώσσες κατά το ότι δεν προορίζεται να περιγράψει λειτουργίες που εκτελούνται σειριακά, η μία μετά την άλλη. Έτσι καθε πρόταση ή τμήμα του κώδικα που παράγει κάποια λειτουργία, το κάνει σε συγχρονισμό με άλλες λειτουργίες. Αυτό σημαίνει ότι κάθε τμήμα του κώδικα μπορεί να γραφεί σε οποιοδήποτε σημείο του ανεξαρτήτως της σειράς του διότι ο compiler θα συνθέσει κύκλωμα βάσει ολόκληρου του προγράμματος.

Επίσης, είναι μια γλώσσα που επιτρέπει την ιεραρχική σχεδίαση κυκλωμάτων εξαιτίας της δομής του κώδικα, που διακρίνει ανάμεσα στην περιγραφή του κυκλώματος ως βαθμίδα (block) και στη λειτουργική του περιγραφή. Τα δύο αυτά μέρη του κώδικα αναφέρονται ως «οντότητα» και «αρχιτεκτονική». Έτσι ο χρήστης μπορεί να περιγράψει ένα σύνθετο κύκλωμα με βάσει τα υποκυκλώματα που το αποτελούν. Τα υποκυκλώματα που συμπεριλαμβάνονται σε μια σχεδίαση ονομάζονται δομικά στοιχεία (components) και συμπεριλαμβάνονται σε μια ανώτερη οντότητα (top-level entity). Αυτά είναι μικρότερα υποκυκλώματα, σε VHDL, που έχουν ήδη περιγραφεί και μπορούν να κληθούν ως αρχεία βιβλιοθήκης.

Η δήλωση ενός δομικου στοιχείου μπορει να γίνει ή στην περιοχή δήλωσης της αρχιτεκτονικής ή σε μία δήλωση πακέτου. Στη συνέχεια, στο κύριο σώμα της αρχιτεκτονικής πρέπει να δημιουργηθούν στιγμιότυπα (instants) για τα δομικά στοιχεία που έχουν δηλωθεί, τα οποία λειτουργούν ως υποκυκλώματα της σχεδίασης. Η δυνατότητα αυτή, της επανάληψης των στιγμιοτύπων συνιστά ένα χαρακτηριστικό της γλώσσας VHDL που καθιστά την περιγραφή του υλικού επαναχρησιμοποιήσιμη (reusable).

Ακόμη, η VHDL είναι μία γλώσσα η οποία μπορεί να συντάξει κώδικα τόσο με συγχονες δομές (concurrent code) όσο με ακολουθιακές δομές (sequential code). Αυτό συμβαίνει διότι μπορει να περιγράψει και συνδιαστικά κυκλώματα, στα οποία οι έξοδοί τους σε κάθε χρονική στιγμή εξαρτώνται μόνο από τις τιμές των εισόδων τους εκείνη τη στιγμή και όχι απο την κατάσταση του κυκλώματος πριν αυτές εφαρμοσθούν (π.χ. πολυπλέκτες). Αλλά και ακολουθιακά κυκλώματα, στα οποία οι τιμές των εξόδων σε κάθε χρονική στιγμή δεν εξαρτώνται μόνο από τις τιμές που έχου οι είσοδοι τους αλλά και από τις τιμές των εξόδων σε προηγούμενες χρονικές στιγμές (π.χ καταχωρητές).

1.5 Αντικείμενα Δεδομένων στη VHDL

1.5.1 Αντικείμενα δεδομένων και τα ονόματά τους

Στην VHDL οτιδήποτε μπορεί να έχει μία τιμή είναι ένα αντικείμενο. Κάθε αντικείμενο έχει ένα όνομα και έναν τύπο δεδομένων (data type), ο οποίος δηλώνει τις τιμές που μπορεί να πάρει το αντικείμενο. Ο τύπος κάθε αντικειμένου μπορεί να καθοριστεί στατικά και δεν μπορεί να αλλάξει κατά την διάρκεια της εκτέλεσης του. Τα αντικείμενα δεδομένων (data objects) που χρησιμοποιεί η VHDL είναι:

- Σήματα (signals): αποδίδουν τιμές στα καλώδια του κυκλώματος και αντιπροσωπεύουν τις διασυνδέσεις ανάμεσα σε μονάδες του κυκλώματος. Μπορούν να χρησιμοποιηθούν τόσο σε τμήματα κώδικα που περιλαμβάνουν σύγχρονες εντολές, όσο και σε ακολουθιακά τμήματα κώδικα.
- Μεταβλητές (variables): χρησιμοποιούνται για την προσωρινή αποθήκευση τιμών που προκύπτουν από την τέλεση αριθμητικών πράξεων. Μια μεταβλητή δηλώνεται και χρησιμοποιείται μόνο σε τμήματα του κώδικα που περιλαμβάνουν «ακολουθιακές εντολές». Ένα τέτοιο τμήμα κώδικα δηλώνεται συνήθως ως ΔΙΕΡΓΑΣΙΑ (PROCESS). Εκτός από τις διεργασίες, άλλα τμήματα κώδικα που περιγράφονται με ακολουθιακές εντολές είναι τα λεγόμενα υποπρογράμματα (FUNCTIONS και PROCEDURES).
- Σταθερές (constants): λαμβάνουν τιμή κατά τη δήλωσή τους και η τιμή αυτή παραμένει στη συνέχεια σταθερή.

Για τον ορισμό των ονομάτων των αντικειμένων μπορούν να χρησιμοποιηθεί οποιοσδήποτε αλφαριθμητικός χαρακτήρας, καθώς και ο χαρακτήρας "_". Υπάρχουν όμως τέσσερις περιορισμοί. Ένα όνομα δεν είναι δυνατόν να είναι κάποια λέξη-κλειδί της γλώσσας, όπως ENTITY, πρέπει να ξεκινάει από ένα γράμμα, δεν είναι δυνατό να τελειώνει με το χαρακτήρα "_" και τέλος, δεν είναι δυνατό να έχει δύο διαδοχικούς χαρακτήρες.

1.5.2 Δήλωση και τύποι Αντικειμένων δεδομένων

Τα αντικείμενα δεδομένων για να χρησιμοποιηθούν μέσα σε κώδικα πρεπει πρώτα να δηλωθούν, δηλαδή να τους δοθεί ένα όνομα και να οριστεί ο τύπος τους. Σε κάθε αντικείμενο η δήλωση γίνεται διαφορετικά.

Τα αντικείμενα δεδομένων τύπου **σήματος (SIGNAL)** αντιπροσοπευουν λογικά σήματα στα καλώδια ενός κυκλωματος και υπάρχουν τρεις θέσεις στις οποίες μπορούν να δηλωθούν σε ένα πρόγραμμα: σε μια δήλωση οντότητας (entity declaration), στο τμήμα δηλώσεων της αρχιτεκτονικήςδομής και στο τμήμα δηλώσεων ενός πακέτου. Ένα σήμα πρέπει να δηλώνεται μέσω του τύπου που έχει:

SIGNAL όνομα_σήματος : τύπος_σήματος

Μία **σταθερά (CONSTANT)** είναι ένα αντικείμενο του οποίου η τιμή δεν είναι δυνατό να αλλάξει. Σε αντίθεση με ένα αντικείμενο τύπου SIGNAL, ένα αντικείμενο τύπου CONSTANT δεν αντιπροσοπεύει κάποιο καλώδιο. Η γενική μορφή της δήλωσής του είναι:

CONSTANT όνομα_σταθεράς := τιμή σταθεράς ;

Τα αντικείμενα τύπου **μεταβλητής (VARIABLE)** χρησιμοποιούνται μερικές φορές για να συγκρατήσουν τα αποτελέσματα των υπολογισμών και για της μεταβλητές δείκτη στους βρόγχους. Η δήλωση γίνεται ως εξής:

VARIABLE όνομα μεταβλητής : τύπος μεταβλητής;

Τα σήματα, οι μεταβλητές όπως και οι σταθερές μεταφέρουν αριθμητική πληροφορία, την οποία υποστηρίζει η VHDL με τη μορφή ακεραίων και δυαδικών τιμών. Οι τύποι ακεραίων στην VHDL μπορούν να περιορίσουν το εύρος τιμών τους με την δήλωση range και οι δυαδικές τιμές παριστάνονται ανάμεσα σε μονά ή διπλά εισαγωγικά, αν είναι τιμές του ενός ή περισσοτέρων bits, αντίστοιχα. Επίσης, οι ακέραιοι αριθμοί μπορούν να γραφούν με δεκαδική μορφή και χωρίς τη χρήση εισαγωγικών.

Στη VHDL διακρίνουμε τους προκαθορισμένους τύπους δεδομένων (predefined data types) και αυτούς που δημιουργούν οι χρήστες (user defined data types). Οι προκαθορισμένοι τύποι είναι προτυποποιημένοι και οι βιβλιοθήκες τους περιλαμβάνονται στα εργαλεία σχεδίασης, οπότε η χρήση τους μπορεί να γίνει με απλή αναφορά σε έτοιμα πακέτα, που περιγράφουν αυτούς τους τύπους.

1.6 Προκαθορισμένοι τύποι δεδομένων (πακέτο standard)

Κάποιοι βασικοί τύποι δεδομένων ανήκουν στην αρχική προτυποποίηση της γλώσσας (πακέτο standard της βιβλιοθήκης std) και λειτουργούν χωρίς να χρειάζεται αναφορά σε βιβλιοθήκες για να χρησιμοποιηθούν σε πρόγραμμα. Τέτοιοι τύποι είναι:

- **Τύποι BIT και BIT_VECTOR**

Οι τύποι αυτοί προσδιορίζονται στα Πρότυπα IEEE 1076 και IEEE 1164. Τα αντικείμενα που είναι τύπου BIT μπορούν να έχουν τις τιμές '0' ή '1'. Ένα αντικείμενο τύπου BIT_VECTOR είναι μία γραμμική σειρά αντικειμένων τύπου BIT και η αλληλουχία των bits αποτελεί ένα διάνυσμα (vector) με οκτώ στοιχεία.

- **Τύπος INTEGER**

Ένα σήμα τύπου INTEGER χρησιμοποιείται με αριθμητικούς τελεστές και αντιπροσωπεύει ένα δυαδικό αριθμό. Το πρόγραμμα δεν καθορίζει τον αριθμό bits του σήματος . Ένα τέτοιο σήμα έχει μήκος 32 bits και μπορεί να αντιπροσωπεύει οποιοδήποτε αριθμό από $-(2^{31}-1)$ έως $(2^{31}-1)$. Το εύρος αυτό είναι κατά έναν αριθμό μικρότερο από το αντίστοιχο εύρος των αριθμών σε συμπλήρωμα ως προς 2 και ο λόγος για αυτό είναι ότι το πρότυπο της γλώσσας VHDL προβλέπει ίσο αριθμό θετικών και αρνητικών αριθμών. Μπορούν επίσης να δηλωθούν ακέραιοι αριθμοί με λιγότερα bits με τη βοήθεια της λέξης-κλειδί RANGE.

- **Τύπος NATURAL**

Είναι υποτύπος του integer και περιλαμβάνει μη αρνητικούς ακεραίους (από 0 μέχρι και το άνω όριο των ακεραίων). Υποστηρίζει τις ίδιες πράξεις με τον τύπο integer.

- **Τύπος POSITIVE**

Είναι υποτύπος του integer και περιλαμβάνει μόνον θετικούς ακεραίους.

- **Τύπος BOOLEAN**

Ένα αντικείμενο τύπου BOOLEAN μποορεί να έχει τις τιμές TRUE και FALSE, όπου TRUE ισοδυναμεί με την τιμή 1 και FALSE με την τιμή 0.

- **Τύπος CHARACTER**

Σήματα αυτού του τύπου μπορούν να πάρουν τιμές από ένα σύνολο 256 συμβόλων των 8-bit. Τα σύμβολα αντιπροσωπεύουν το σύνολο χαρακτήρων ISO 8859-1, ενώ τα πρώτα 128 σύμβολα ανήκουν στον κοινό κώδικα ASCII. Αν και κάθε χαρακτήρας έχει εύρος 8 bits, ο τύπος διαχειρίζεται κάθε φορά ένα χαρακτήρα, οπότε είναι βαθμωτός.

- **Τύπος TIME**

Αυτός ο τύπος δεδομένων προορίζεται μόνο για προσομοίωση (όχι για σύνθεση). Με τη βοήθειά του ορίζονται ακέραιοι, που παριστάνουν χρονικές στιγμές στον χρόνο προσομοίωσης.

1.7 Τύποι δεδομένων που δημιουργούνται από το χρήστη(user defined data types)

Εκτός από τους προκαθορισμένους τύπους, υπάρχουν και τύποι δεδομένων οι οποίοι δημιουργούνται από τον ίδιο το χρήστη. Η δήλωση του τύπου δεδομένων μπορεί να γίνει στο σώμα δηλώσεων της αρχιτεκτονικής ή σε ξεχωριστό πακέτο. Η δήλωση ενός τύπου δεδομένων από το χρήστη γίνεται ως εξής:

TYPE όνομα_τύπου IS περιγραφή τύπου ;

Οι τύποι δεδομένων που ορίζει ο χρήστης ανήκουν στις ίδιες κατηγορίες, όπως οι προκαθορισμένοι. Μία κατηγορία τύπων είναι οι «βαθμωτοί» (scalar), όπου τα αντικείμενα λαμβάνουν μία τιμή από ένα σύνολο, ενώ μία άλλη κατηγορία είναι οι «σύνθετοι» (composite), που περιλαμβάνουν πολλές τιμές. Μια μορφή σύνθετου τύπου είναι οι πίνακες στους οποίους τα στοιχεία προσπελάζονται με τη βοήθεια δεικτών. Επίσης, ένας σύνθετος τύπος μπορεί να είναι RECORD (εγγραφή), οπότε περιλαμβάνει συλλογές δεδομένων, που μπορεί να ανήκουν και σε διαφορετικούς τύπους.

1.8 Τύπος απαρίθμησης (ENUMERATION)

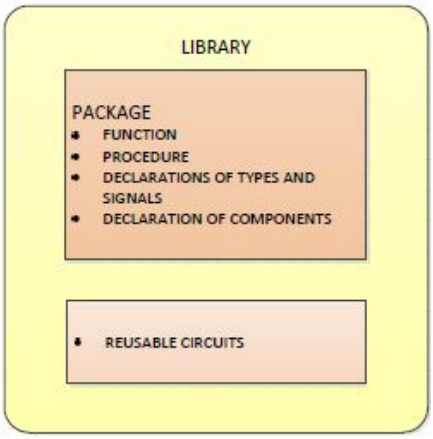
Ένα σήμα τύπου απαρίθμησης (ENUMERATION) είναι ένα σήμα του οποίου οι δυνατές τιμές καθορίζονται από το χρήστη. Η γενική μορφή του είναι:

TYPE όνομα_τύπου_emumeration IS (όνομα {,όνομα});

Τα άγκιστρα δηλώνουν ότι μπορούν να περιληφθούν περισσότερα από ένα ονόματα. Το πιο κοινό παράδειγμα χρήσης του τύπου ENUMERATION είναι για να καθορίζουμε τις καταστάσεις μιας μηχανής πεπερασμένων καταστάσεων.

1.9 Βιβλιοθήκες και πακέτα της γλώσσας VHDL

Μια βιβλιοθήκη (library) VHDL αποτελεί το χώρο όπου ο VHDL compiler αποθηκεύει πληροφορίες για μια σχεδίαση ενός project, συμπεριλαμβάνοντας και ενδιάμεσα αρχεία που χρησιμοποιούνται κατά την ανάλυση, προσομοίωση και σύνθεση του κυκλώματος. Ο σκοπός μιας βιβλιοθήκης είναι να συγκεντρώνει σχεδιάσεις που θα χρησιμοποιηθούν ξανά από τους ίδιους ή άλλους χρήστες. Μια βιβλιοθήκη είναι οργανωμένη σε έναν φάκελο αρχείων, όπου συγκεντρώνονται όλοι οι κώδικες της συλλογής. Στο Σχήμα 1.2 φαίνονται τα βασικά μέρη μιας βιβλιοθήκης.



Σχήμα 1.2 Βασικά μερη μιας βιβλιοθήκης VHDL

Ένα πακέτο της γλώσσας VHDL λειτουργεί όπως μια αποθήκη. Χρησιμοποιείται για να συγκρατει προγράμματα της γλώσσας VHDL που είναι γενικής χρήσης, όπως τα προγράμματα που ορίζουν έναν τύπο. Το πακέτο μπορεί να περιληφθεί για χρήση σε οποιδήποτε αριθμό αρχείων προγραμμάτων, τα οποία μπορούν να χρησιμοποιούν τους ορισμούς που αναφέρονται στο πακέτο.Όπως μια αρχιτεκτονική έτσι και ένα πακέτο μπορεί να έχει δύο κύρια μέρη: τη δήλωση πακέτου (package declaration) και το σώμα του πακέτου (package body). Μια χρήση του σώματος του πακέτου είναι να ορίζει συναρτήσεις της γλώσσας VHDL. Οι ορισμοί που υπάρχου σε αυτό το πακέτο, όπως ο ορισμός κάποιου τύπου, μπορούν να χρησιμοποιηθούν σε οποιοδήποτε αρχείο προγράμματος που περιέχει τις εντολές:

LIBRARY όνομα_βιβλιοθήκης ;

USE όνομα_βιβλιοθήκης.όνομα πακέτου.all ;

Το όνομα_βιβλιοθήκης αντιπροσωπεύει τη θέση στο σύστημα αρχείων του υπολογιστή, όπου αποθηκεύεται το πακέτο. Μία βιβλιοθήκη μπορεί να παρέχεται είτε

ως τμήμα ενός συστήματος CAD, οπότε ονομάζεται βιβλιοθήκη συστήματος (system library), είτε να δημιουργηθεί από το χρήστη, οπότε λέγεται βιβλιοθήκη χρήστη (user library). Ένα παράδειγμα βιβλιοθήκης συστήματος είναι η βιβλιοθήκη ieee, η οποία περιέχει τα πακέτα std_logic_1164, std_logic_signed, std_logic_unsigned και std_logic_arith.

Στο πακέτο std_logic_1164 περιέχονται οι τύποι STD_LOGIC και STD_LOGIC_VECTOR. Ο τύπος STD_LOGIC προστέθηκε στο Πρότυπο IEEE 1164 της VHDL και παρέχει περισσότερη ευελιξία από τον τύπο BIT. Για να χρησιμοποιήθεί ο τύπος αυτός πρέπει να συμπεριληφθούν δύο εντολές:

LIBRARY ieee;

USE ieee.std_logic_1164.all;

Για ένα αντικείμενο του τύπου STD_LOGIC οι παρακάτω τιμές είναι έγκυρες: 0, 1, Z, -, L, H, U, X και W. Μόνο οι τέσσερις πρώτες τιμές είναι χρήσιμες για τη σύνθεση λογικών κυκλωμάτων. Η τιμή Z αντιπροσωπεύει την κατάσταση υψηλής σύνθετης αντίστασης και η τιμή "_" αντιπροσωπεύει τις αδιάφορες λογικές καταστάσεις. Η τιμή L σημαίνει "ασθενής τιμή 0", η τιμή H σημαίνει "ασθενής τιμή 1", η τιμή U σημαίνει "μη-μηδενισμένη τιμή", η τιμή X σημαίνει "αγνωστη τιμή" και η τιμή V σημαίνει "ασθενές άγνωστη τιμή". Ο τύπος STD_LOGIC_VECTOR αντιπροσωπεύει μία σειρά αντικειμένων τύπου STD_LOGIC και μπορεί να πάρει τις ίδιες τιμές με αυτά.

Για να είναι δυνατή η χρήση αριθμητικών με τους τύπους STD_LOGIC και STD_LOGIC_VECTOR πρέπει να δηλωθεί επιπλέον ένα από τα πακέτα std_logic_unsigned ή std_logic_signed. Τα πακέτα αυτά χρησιμοποιούν ένα άλλο πακέτο που ονομάζεται std_logic_arith. Το πακέτο αυτό ορίζει δύο τύπους σημάτων, τους SIGNED και UNSIGNED. Σκοπός τους είναι να επιτραπεί στο χρήστη να δείξει στο πρόγραμμα που αναπτύσσει τι είδου παράσταση αριθμών χρησιμοποιεί. Ο τύπος SIGNED χρησιμοποιείται σε προγράμματα κυκλωμάτων που χειρίζονται προσημασμένους αριθμούς (δηλαδή γραμμένους στο συμπλήρωμα ως προς 2) και ο τύπος UNSIGNED χρησιμοποιείται σε προγράμματα που χειρίζονται μη-προσημασμένους αριθμούς.

1.10 Δομή προγράμματος VHDL

Η δομή του πρόγραμματος που περιγράφει ένα κύκλωμα στη γλώσσα VHDL χωρίζεται σε δύο μέρη, την οντότητα (entity) και την αρχιτεκτονική (architecture). Πριν το τμήμα της οντότητα δηλώνονται κάποια πακέτα βιβλιοθηκών, τα οποία επιτρέπουν τη χρήση τμημάτων κώδικα που έχουν ήδη δημιουργηθεί στο παρελθόν και χρησιμοποιούνται συχνά, ανάλογα με τους τύπου δεδομένων που χρειάζεται να περιγράψουν ή τα υποκυκλώματα που χρειαζεται να χρησιμοποιήσουν. Το τυποποιημένο πακέτο, το οπίο χρησιμοποιείται πάρα πολυ συχνά είναι το std_logic_1164 της βιβλιοθήκης ieee, το οποίο περιγράφει τον τύπο δεδομένων std_logic. Στο Σχήμα 1.2 φαίνεται η γενική δομή ενός αρχείου VHDL.

Σχήμα 1.3 Γενική δομή ενός αρχείου VHDL

Η οντότητα (entity) περιγράφει το κύκλωμα ως βαθμίδα, με εισόδους και εξόδους. Τα σήματα εισόδου και εξόδου της καθορίζονται με τη βοήθεια της δήλωσης οντότητας, η οποία γίνεται με τη λέξη-κλειδί ENTITY και με τη βοήθεια της λέξη-κλειδί PORT (θύρα). Εάν μία θύρα είναι είσοδος, έξοδος ή διπλής κατεύθυνσης καθορίζεται από τηκ κατάσταση (mode) της θύρας. Η γενική μορφή της δήλωσης μιας οντότητα είναι:

ENTITY όνομα_οντότητας IS

PORT(όνομα σήματος1 : τρόπος_λειτουργίας τύπος_σήματος1;

όνομα σήματος2 : τρόπος_λειτουργίας τύπος_σήματος2

όνομα σήματοςN : τρόπος_λειτουργίας τύπος_σήματοςN);

END όνομα_οντότητας ;

Οι δυνατές καταστάσεις που μπορούν να παρουσιάσουν τα σήματα εισόδου-εξόδου είναι:

- **IN**: Χρησιμοποιείται για ένα σήμα που αποτελεί είσοδο σε μια οντότητα.
- **OUT**: Χρησιμοποιείται για ένα σήμα που αποτελεί έξοδο σε μια οντότητα. Η τιμή του σήματος δεν μπορεί να χρησιμοποιηθεί μέσα στην οντότητα.
- **INOUT**: Χρησιμοποιείται για ένα σήμα που αποτελεί είσοδο αλλά και έξοδο σε μια οντότητα.
- **BUFFER**: Χρησιμοποιείται για ένα σήμα που αποτελεί έξοδο σε μια οντότητα. Η τιμή του σήματος δεν μπορεί να χρησιμοποιηθεί μέσα στην οντότητα αλλά μπορεί να διαβαστεί στο εσωτερικό της.

Εάν η κατάσταση μιας θύρας δεν καθορίζεται τότε θεωρείται θύρα εισόδου (IN).

Η αρχιτεκτονική (architecture) μιας οντότητας παρέχει λεπτομέρειες του κυκλώματος της. Αυτή αποτελείται από δύο κύρια μέρη: την περιοχή δήλωσης (declaration region) και το σώμα της αρχιτεκτονικής (architecture body). Η περιοχή δήλωσης εμφανίζεται πριν από τη λέξη-κλειδι BEGIN. Μπορεί να χρησιμοποιηθεί για τη δήλωση σημάτων, σινιστωσών, τύποι ορισμένοι από το χρήστη και σταθερές. Η λειτουργία μιας οντότητας καθορίζεται στο σώμα της αρχιτεκτονικής, το οποίο έπεται της λέξης BEGIN. Το τμήμα αυτό περιλαμβάνει εντολές που καθορίζουν τις λογικές συναρτήσεις που εκτελεί το κύκλωμα. Η αχιτεκτονική ενός κυκλώματος περιγράφεται παρακάτω:

ARCHITECTURE όνομα_αρχιτεκτονικής OF όνομα_οντότητας IS
[Δηλώσεις επιπλέον μεταβλητών]
BEGIN
Εντολές που περιγράφουν λογικές λειτουργίες και αναθέτουν τιμές σε σήματα.
END [ARCHITECTURE] [όνομα_αρχιτεκτονικής];

1.11 Εντολή γενικών δηλώσεων (GENERIC)

Οι γενικές δηλώσεις αφορούν σε στεθερές και στόχος τους είναι η παραμετροποίηση του κυκλώματος ώστε να μπορεί να επαναχρησιμοποιηθεί σε περισσότερες περιπτώσεις. Για τις δηλώσεις αυτές χρησιμοποιείται η δεσμευμένη λέξη GENERIC, η οποία τοποθετείται μέσα στην οντότητα, αμέσως πριν τη δήλωση PORT. Με τον τρόπο αυτό, οι γενικές σταθερές (generic constants) είναι πραγματικά καθολικές

(global) και μπορούν να επηρεάσουν ακόμη και τις προδιαγραφές των θυρών. Η σύνταξή της γίνεται ως εξής:

GENERIC (όνομα_παραμέτρου : τύπος_δεδομένων := τιμή{;
όνομα_παραμέτρου : τύπος_δεδομένων := τιμή});

Η βασική δομή που χρησιμοποιείται στο σώμα της αρχιτεκτονικής είναι η FOR…GENERATE. Πρόκειται για μια πολύ ευέλικτη δομή επανάληψης, που επιτρέπει να λαμβάνουν τιμές σήματα τύπου πίνακα.

1.12 Είδη εντολών στη VHDL

Ένα ψηφιακό σύστημα περιλαμβάνει και συνδυαστικά και ακουθιακά κυκλώματα. Η περιγραφή και προσομοίωση ενός κυκλώματος σε VHDL γίνεται με δύο τύπους εντολών, τις σύγχρονες (concurrent) και τις ακολουθιακές (sequential). Ένα συνδυαστικό κύκλωμα μεταβάλει τα σήματα εξόδου παράλληλα με τις εισόδους, με άμεσο τρόπο, χωρίς φαινόμενα μνήμης. Ένα ακολουθιακό κύκλωμα περιέχει βρόχους ανάδρασης, ώστε οι τιμές των εξόδων επηρεάζονται από προηγούμενες καταστάσεις του συστήματος. Οι αλλαγές των εξόδων γίνονται σε διατεταγμένα βήματα, με τη βοήθεια μια ακολουθίας παλμών, που ελέγχει πότε γίνονται οι αλλαγές των καταστάσεων.

Οι προτάσεις SELECT, WHEN και GENERATE θεωρούνται καθαρά σύγχρονες και τοποθετούνται αποκλειστικά έξω από διεργασίες και υποπρογράμματα. Αντίστοιχα,οι προτάσεις IF, WAIT, LOOP και CASE θεωρούνται καθαρά ακολουθιακές και τοποθετούνται μόνον σε διεργασίες και υποπρογράμματα.

1.12.1 Η εντολή SELECT

Μια βασική σύγχρονη εντολή είναι η SELECT, που συντάσσεται ως εξής:

WITH αναγνωριστικό SELECT
Έκφραση_ανάθεσης_τιμής WHEN τιμή
Ανάθεση_τιμής WHEN τιμή
Ανάθεση_τιμής WHEN τιμή
….;

Το «αναγνωριστικό» μπορεί να είναι το όνομα ενός σήματος. Επειδή η εντολή SELECT απαιτεί να καλύπτονται όλες οι δυνατές τιμές που λαμβάνει το «αναγνωριστικό», η παραπάνω δομή κώδικα κλείνει με την έκφραση WHEN OTHERS; στην οποία περιλαμβάνονται όλες οι δυνατές τιμές του αναγνωριστικού που δεν έχουν αναφερθεί παραπάνω. Η εντολή SELECT προφανώς υλοποιεί κυκλώματα με εισόδους επιλογής, όπως οι πολυπλέκτες, οι αποκωδικοποιητές και οι κωδικοποιητές.

1.12.2 Εντολή Διεργασίας (PROCESS)

Η εντολή διεργασίας (PROCESS) αναγράφεται μέσα στο σώμα της αρχιτεκτονικής και περικλείει στο εσωτερικό της άλλες εντολές. Οι εντολές IF, CASE και LOOP μπορούν να υπάρχουν μόνο μέσα σε μια διεργασία και χρησιμοποιούνται για να περιγράψουν είτε συνδιαστικά είτε ακολουθιακά κυκλώματα. Η γενική μορφή της εντολής είναι η εξής:

PROCESS [(λίστα ευαισθησίας)] [IS]

[τμήμα δηλώσεων]

BEGIN

Ακολουθιακές προτάσεις

END PROCESS;

Η δομή της είναι κάπως παρόμοια με αυτή της αρχιτεκτονικής. Μια διεργασία συνοδεύεται τις περισσότερες φορές από μια λίστα ευαισθησίας. Αυτή, περιλαμβάνει τα σήματα, που η αλλαγή της κατάστασής τους έχει ως αποτέλεσμα την εκτέλεση της διεργασίας. Ένα πολύ συνηθισμένο σήμα στη λίστα ευαισθησίας είναι το clock. Τα αντικείμενα δεδομένων VARIABLE μπορουν να δηλωθούν μόνο μέσα στη διεργασία. Οποιαδήποτε μεταβλητή που δηλώνεται μπορεί να χρησιμοποιηθεί μόνο από το πρόγραμμα που περιέχεται στη διεργασία. Για να χρησιμοποιηθεί μια τέτοια εντολή έξω από τη διεργασία θα πρέπει η τιμή της να αντιστοιχηθεί σε κάποιο σήμα.

1.12.3 Εντολή IF

Η εντολή Αν είναι εντολή διακλάδωσης υπό συνθήκη και μπορεί να γραφεί μόνον μέσα σε ακολουθιακό τμήμα κώδικα (διεργασία PROCESS). Ελέγχει αν μια συνθήκη είναι αληθής, οπότε εκτελεί το μέρος του κώδικα που ακολουθεί, αλλιώς ελέγχει μια νέα σειρά συνθηκών και εκτελεί το αντίστοιχο μέρος του κώδικα. Η πλήρης περιγραφή της IF έχει ως εξής:

IF συνθήκη THEN
Προτάσεις ανάθεσης;
ELSIF συνθήκη THEN
Προτάσεις ανάθεσης
ELSE
Προτάσεις ανάθεσης;
END IF;

Σε περίπλοκες περιπτώσεις μπορεί να υπάρχουν πολλαπλές ELSIF. Η IF μπορεί να διατυπωθεί και απλά:

IF συνθήκη THEN
προτάσεις ανάθεσης;
END IF;

1.12.4 Εντολή CASE

Η εντολή CASE έχει παρόμοια λειτουργία με την IF, αφού κι αυτή επιλέγει ανάμεσα σε διαφορετικές ομάδες ακολουθιακών εντολών με βάση την ισχύ μιας συνθήκης. Πρέπει να αναγραφούν όλοι οι δυνατοί συνδιασμοί τιμών έκφρασης που χρησιμοποιείται στις εντολές WHEN και γι'αυτό είναι αναγκαία η χρήση της λέξης-κλειδί OTHERS. Η διαφορά είναι ότι η IF εξετάζει διαδοχικά την ισχύ μιας σειράς λογικών συνθηκών, προκειμένου να επιλέξει την ομάδα εντολών που θα εκτελέσει, ενώ η CASE επιλέγει με βάση την τιμή που λαμβάνει μια μοναδική έκφραση. Η σύνταξή της είναι ως εξής:

CASE έκφραση IS
WHEN τιμή => αναθέσεις;
WHEN τιμή => αναθέσεις;

....

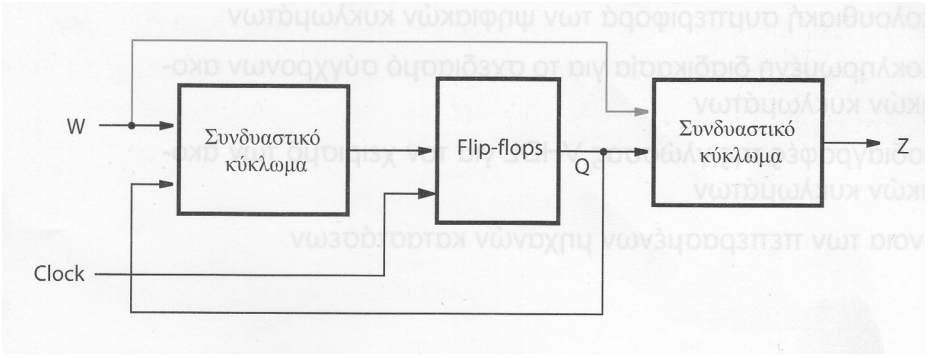
END CASE;

Η εντολή CASE είναι, επίσης, πολύ χρήσιμη στην περιγραφή μηχανών πεπερασμένων καταστάσεων (Finite State Machines).

Κεφάλαιο 2 :
Μηχανές πεπερασμένων καταστάσεων
(Finite State Machines)

2.1 Εισαγωγή

Τα ακολουθιακά κυκλώματα (sequential circuits) είναι μια γενική κατηγορία κυκλωμάτων, στα οποία οι έξοδοι εξαρτώνται από την προηγούμενη συμπεριφορά του κυκλώματος, καθώς και από την τρέχουσα τιμή των εισόδων του. Στο Σχήμα 2.1 φαίνεται η γενική μορφή ενός ακολουθιακού κυκλώματος. Στις περισσότερες περιπτώσεις υπάρχει ένα ωρολογιακό σήμα που ελέγχει τη λειτουργία ενός ακολουθιακού κυκλώματος και ένα τέτοιο κύκλωμα λέγεται σύγχρονο ακολουθιακό κύκλωμα (synchronous sequential circuits). Η άλλη περίπτωση, στην οποία δεν χρησιμοποιείται ωρολογιακό σήμα ονομάζεται ασύγχρονο ακολουθιακό κύκλωμα (asynchronous sequential circuits). Τα σύγχρονα κυκλώματα σχεδιάζονται ευκολότερα και χρησιμοποιούνται σε ένα τεράστιο εύρος πρακτικών εφαρμογών.



Σχήμα 2.1 Γενική μορφή ενός ακολουθιακού κυκλώματος

Μία μηχανή πεπερασμένων καταστάσεων (FSM) είναι μία ειδική τεχνική μοντελοποίηση ακολουθιακών κυκλωμάτων που υλοποιείται με τη βοήθεια συνδιαστικής λογικής και ενόςή περισσότερων Flip-Flops. Αυτή η προσσέγγιση μπορεί να είναι πολύ χρήσιμη στο σχεδιαδμό κυκλωμάτων που εκτελούν μια καθορισμένη ακολουθία λειτουργιών. Δηλαδή, τα έργα των οποίων αποτελούν μια καλά καθορισμένη λίστα, που περιέχει όλες τις πιθανές καταστάσεις του συστήματος και τις αναγκαίες προϋποθέσεις για το σύστημα να κινηθεί από τη μία κατάσταση

στην άλλη, καθώς και τις τιμές εξόδου που πρέπει να παράγει η κάθε κατάσταση του συστήματος. Οι ψηφιακοί ελεγκτές ή οι μονάδες ελέγχου είναι από τα κλασσικά παραδείγματα των κυκλωμάτων που εμπίπτουν σε αυτή την κατηγορία. Υπάρχουν δύο θεμελιώδεις αναπαραστάσεις για τις FSMs, η μία σχετίζεται με τις προδιαγραφές (διάγραμμα καταστάσεων) και η άλλη σχετίζεται με το υλικό (συνδυαστική ενάντια ακολουθιακής λογικής).

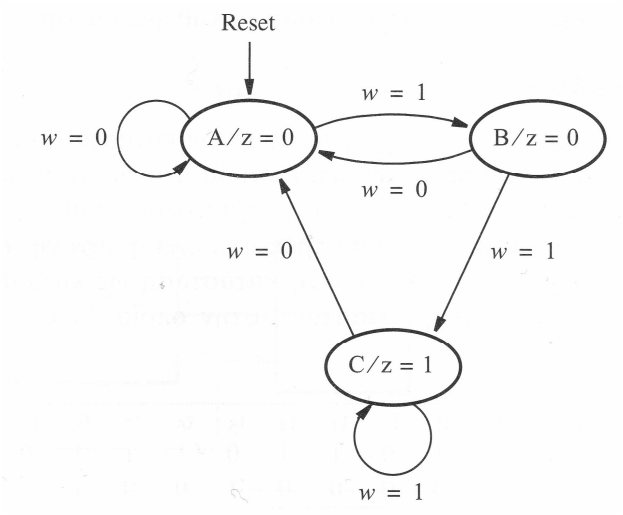
Πιο συγκεκριμένα, μια μηχανή πεπερασμένων καταστάσεων (FSM) αποτελείται απο:

FSM=(S , I , O , f , g , So)

- Ένα πεπερασμένο σύνολο καταστάσεων S.
- Ένα πεπερασμένο σύνολο απο γράμματα εισόδου I.
- Ένα πεπερασμένο σύνολο απο γράμματα εξόδου O.
- Μια συνάρτηση μετάβασης f ,η οποία αναθέτει νέα κατάσταση σε κάθε ζεύγος κατάστασης και εισόδου.
- Μια συνάρτηση εξόδου g ,η οποία αναθέτει μια έξοδο σε κάθε ζεύγος κατάστασης και εισόδου.
- Ένα ειδικό στοιχείο,του συνόλου S,που ονομάζεται αρχική κατάσταση So.

2.2 Διάγραμμα καταστάσεων (state diagram)

Ο πιο απλός τρόπος να περιγραφεί η συμπεριφορά ενός ακολουθιακού κυκλώματος είναι να χρησιμοποιηθεί μία εικονική αναπαράσταση με τη μορφή ενός διαγράμματος καταστάσεων (state diagram). Το οποίο είναι ένα διάγραμμα που απεικονίζει τις καταστάσεις ενός κυκλώματος ως κύκλους (δηλαδή κόμβους) και τις μεταβάσεις μεταξύ αυτών ως κατευθυνόμενα τόξα. Το διάγραμμα καταστάσεων καθορίζει τη συμπεριφορά που αντιστοιχεί στις προδιαγραφές του κυκλώματος. Οι καταστάσεις A, B και C απεικονίζονται ως κόμβοι στο διάγραμμα του Σχήματος 2.2, το οποίο είναι ένα παράδειγμα ακολουθιακού κυκλώματος.



Σχήμα 2.2 Διάγραμμα καταστάσεων ενός απλού ακολουθιακού κυκλώματος

Ο κόμβος Α αντιπροσωπεύει την αρχική κατάσταση, η οποία είναι η κατάσταση στην οποία οδηγείται το κύκλωμα όταν εφαρμόζεται στην είσοδο w ένα λογικό 0. Στην κατάσταση αυτή η έξοδος z θα πρέπει να ισούται με μηδέν και αυτό φαίνεται από την έκφραση "A/z=0" που γράφεται στον κόμβο. Το κύκλωμα πρέπει να μείνει στην κατάσταση Α για όσο χρόνο θα είναι w=0, κάτι που δηλώνεται από ένα τόξο που ξεκινά και τερματίζεται στον κόμβο και γράφει την επιγραφή "w=0". Η πρώτη εμφάνιση εισόδου w ίσης με 1 καταγράφεται με τη μετάβαση από την κατάσταση Α σε μία άλλη Β. Αυτό δηλώνεται από ένα τόξο που ξεκινά από την Α και τερματίζει στη Β, με την επιγραφή "w=1". Στην κατάσταση Β η έξοδος παραμένει στην τιμή 0 και γι'αυτό γράφεται στον κόμβο η έκφραση "B/z=0".

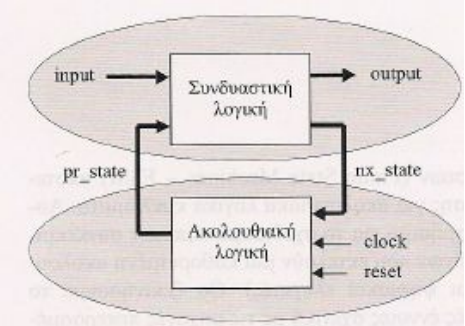
Όταν το κύκλωμα βρίσκεται στην κατάσταση Β, θα μεταβεί στνη κατάσταση C μόνο όταν η είσοδος w εξακολουθεί να παραμένει ίση με 1 κατά το επόμενο ενεργό μέτωπο του ωρολογιακού σήματος. Εάν η είσοδος w παραμένει στην τιμή 1 κατά τη διάρκεια διαδοχικών ωρολογιακών παλμών, το κύκλωμα θα παραμείνει στην κατάσταση C και θα εξακολουθεί να παράγει έξοδο z=1. Όταν, όμως, η είσοδος w γίνει ίση με 0 όταν το κύκλωμα βρίσκεται είτε στην κατάσταση Β είτε στην κατάσταση C,το επόμενο μέτωπο των ωρολογιακών παλμών θα οδηγήσει το σύστημα στην κατάσταση Α.

Η είσοδος Reset χρησιμοποιείται για να θέσει το κύκλωμα στην κατάσταση Α, πράγμα το οποίο είναι δυνατό ανεξάρτητα από την κατάσταση στην οποία βρίσκεται το κύκλωμα. Εφόσον οι καταστάσεις σε μια μηχανή πεπερασμένων καταστάσεων υλοποιούνται με τη χρήση flip-flops και αυτά έχουν τη δυνατότητα επαναφοράς,

μπορούμε να υποθέσουμε ότι η είσοδος Reset χρησιμοποιείται για να θέσει όλα τα flip-flops στο 0 με τη χρήση αυτής της δυνατότητας.

2.3 Αναπαράσταση βασισμένη σε υλικό (Hardware-Based Representation)

Η αναπαράστασηδομικού διαγράμματος μιας μηχανής πεπερασμένων καταστάσεων (FSM) φαίνεται στο Σχήμα 2.3 όπου δείχει ένα σύστημα χωρισμένο σε δύο τμήματα.



Σχήμα 2.3 Αναπαράσταση δομικού διαγράμματος FSM

Το κατώτερο τμήμα είναι το ακολουθιακό ((sequential) περιέχει τα flip-flops), ενώ το ανώτερο τμήμα είναι το συνδιαστικό ((combinational) περιέχει τα συνδιαστικά κυκλώματα). Στο τμήμα της συνδιαστικής λογικής, το σήμα της τρέχουσας κατάστασης αποθηκεύεται στα flip-flops και ονομάζεται μεταβλητή παρούσας ή τρέχουσας κατάστασης (pr_state), ενώ αυτό που θα αποθηκευτεί στον επόμενο (ας πούμε θετικό) ωρολογιακό παλμό λέγεται μεταβλητή επόμενης κατάστασης (nx_state). Ακόμη, διαθέτει εκτός από την pr_state ως είσοδο και την input η οποία είναι η κύρια εξωτερική είσοδος και εκτός από την nx_state ως έξοδο και την output η οποία είναι η κύρια έξοδος.

Στο τμήμα της ακολουθιακής λογικής περιλαμβάνει τρεις εισόδους: την clock ή clk η οποία είναι το ρολόι,την reset ή rst η οποία είναι απαραιτητη για το σήμα επαναφοράς και την nx_state η οποία μας δείχνει την επόμενη κατάσταση, όπως αναφέραμε και παραπάνω. Τέλος, διαθέτει και μια έξοδο: την pr_state η οποία είναι για την τρέχουσα κατάσταση.

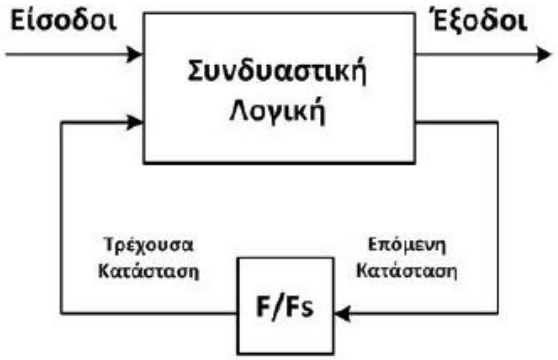
Ένας περιορισμός αυτής της αρχιτεκτονικής είναι ότι η έξοδος παράγεται από συνδυαστικά κυκλώματα και υπάρχει περίπτωση να υπάρξουν δυσλειτουργίες. Αν

όμως στην εφαρμογή αυτή, οι δυσλειτουργίες δεν είναι αποδεκτές, τότε κάποια λύση πρέπει να παρέχεται.

Ακόμη, οι μηχανές πεπερασμένων καταστάσεων ταξινομούνται σε δύο είδη, στο μοντέλο καταστάσεων του Mealy ή στο μοντέλο καταστάσεων του Moore αναλόγως από την εξάρτηση που έχουν τα σήματα των εισόδων τους.

2.4 Μοντέλο καταστάσεων του Mealy

Οι μηχανές καταστάσεων που ανήκουν στο μοντέλο αυτό, είναι γνωστές σαν μηχανές Mealy, επειδή μελετήθηκαν για πρώτη φορά απο τον G.H.Mealy το 1955. Χαρακτηριστικό των συγκεκριμένων μηχανών είναι ότι οι έξοδοι αντιστοιχούν σε μεταβάσεις μεταξύ καταστάσεων, δηλαδή με άλλα λόγια οι έξοδοι δημιουργούνται με βάση παρούσα κατάσταση του κυκλώματος και τις τρέχουσες τιμές των εισόδων. Στο Σχήμα 2.4 απεικονίζεται το διάγραμμα μηχανών πεπερασμένων καταστάσεων του μοντέλου Mealy.

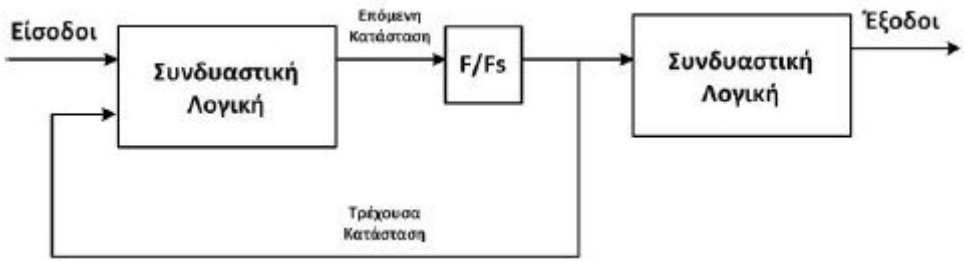


Σχήμα 2.4 Διάγραμμα FSM του μοντέλου Mealy

2.5 Μοντέλο καταστάσεων του Moore

Οι συγκεκριμένες μηχανές πεπερασμένης κατάστασης έγιναν γνωστές σαν μηχανές Moore, διότι τις είχε παρουσιάσει ο E.F.Moore το 1956 έναν χρόνο αργότερα από την εμφάνιση των Mealy μηχανών. Χαρακτηριστικό αυτών των μηχανών είναι ότι η έξοδος προσδιορίζεται μόνο από την τρέχουσα κατάσταση, δηλαδή πιο απλά οι έξοδοι είναι συνάρτηση μόνο της τρέχουσας κατάστασης. Ιδιαίτερο χαρακτηριστικό των μηχανών Moore (Moore FSMs) είναι ότι απαιτούν έναν κύκλο ρολογιού

παραπάνω σε σχεση με τις μηχανές Mealy (Mealy FSMs) δηλαδή συνολικά οι μηχανές Moore απαιτούν δύο κύκλους ρολογιού, έναντι ενός των μηχανών Mealy. Οι κύκλοι ρολογιού είναι δύο, ένας κύκλος υπολογισμού της κατάστασης και ένας κύκλος για τον υπολογισμό της εξόδου. Στο Σχήμα 2.5 φαίνεται το διάγραμμα μηχανών πεπερασμένων καταστάσεων του μοντέλου Moore.



Σχήμα 2.5 Διάγραμμα FSM του μοντέλου Moore

2.6 Οι μηχανές πεπερασμένων καταστάσεων (FSMs) στη γλώσσα VHDL

Μία μηχανή πεπερασμένων καταστάσεων (FSMs) μπορεί να περιγραφεί πολύ εύκολα από μία Διεργασία (PROCESS), η οποία περιλαμβάνει το σήμα ρολογιού στη λίστα ευαισθησίας. Ακόμη, τα προγράμματα σύνθεσης (τύπου ISE) καταλαβαίνουν ότι πρόκειται για περιγραφές FSMs αν ακολουθούνται κάποιοι απλοί κανόνες. Οι μεταβάσεις καταστάσεων (state transitions) μπορούν να περιγραφούν σε μια Διεργασία (PROCESS), με χρήση clock και ασύγχρονου reset και οι έξοδοι περιγράφονται χρησιμοποιώντας απλούς κανόνες σχεδιασμού συνδυαστικών κυκλωμάτων π.χ. παράλληλες δηλώσεις ή χρήση process με δήλωση όλων των εισόδων στη λίστα ευαισθησίας.

2.6.1 Περιγραφή σχεδιασμού μοντέλου Moore

Στο σχεδιασμό του μοντέλου Moore, το τμήμα της ακολουθιακής λογικής διαχωρίζεται πλήρως από του τμήματος συνδυαστικής λογικής και όλες οι καταστάσεις της μηχανής δηλώνονται ρητά με χρήση τύπου δεδομένων απαρίθμησης.

Στο ακουλουθιακό τμήμα του έχει τρεις εισόδους: clock, reset και το σήμα της επόμενης κατάστασης (nx_state) ενώ στην έξοδο έχει μόνο το σήμα της τρέχουσας κατάστασης (pr_state). Τέλος, για τον προγραμματισμό του τμήματος θα απαιτηθεί μια process και εντολές ακολουθιακής λογικής όπως οι If, Wait, Case και Loop). Ακολουθεί παράδειγμα κώδικα για το τμήμα ακολουθιακής λογικής:

```
Process (clock, reset)
Begin
   If (reset= '1') then
        pr_state <= state0 ;
   Elseif (clock'Enent and clock= '1') then
        pr_state <= nx_state ;
    End If;
End Process;
```

Το παρακάτω παράδειγμα κώδικα απεικονίζει το τμήμα συνδιαστικής λογικής:

```
Process (input, pr_state)
Begin
   Case pr_state Is
        when state0 =>
                If (input=....) then
                        output <= <value>;
                        nx_state <= state1;
                Else…
                End If;
        when state1 =>
                If (input=....) then
                        output <= <value>;
                        nx_state <= state2;
                Else…
                End If;
          ....
    End Case;
End Process;
```

2.6.2 Περιγραφή σχεδιασμού μοντέλου Mealy

Στο σχεδιασμό του μοντέλου Mealy, δεν αποθηκεύεται μόνο το σήμα τρέχουσας κατάστασης (pr_state) αλλά και η κατάσταση της εισόδου. Σε ένα τέτοιο μοντέλο η έξοδος αλλάζει κάθε φορά που αλλάζει και η είσοδος (ασύγχρονη έξοδος). Σε πολλές περιπτώσεις τα σήματα πρέπει να είναι σύγχρονα, δηλαδή η έξοδος να ενημερώνεται όταν εμφανίζεται η σωστή ακμή του ρολογιού. Για να γίνουν οι Mealy μηχανές σύγχρονες θα πρέπει να αποθηκευτεί και η έξοδος της.

Η σύνταξη του κώδικα για το τμήμα ακολουθιακής λογικής στο μοντέλο αυτό απεικονίζεται στο παρακάτω παράδειγμα:

```
Process (reset, clock)
Begin
   If (reset= '1') then
        pr_state <= state0 ;
   Elseif (clock'Enent and clock= '1') then
        output <= temp ;
        pr_state <= nx_state ;
    End If;
End Process;
```

Το τμήμα της συνδιαστικής λογικής του φαίνεται στο ακόλουθο παράδειγμα κώδικα:

```
Process (pr_state)
Begin
   Case pr_state Is
        when state0 =>
                temp <= <value>;
                If (συνθήκη) then nx_state <= state1;
                .....
                End If;
        when state1 =>
                temp <= <value>;
                If (συνθήκη) then nx_state <= state2;
                .....
                End If;
```

```
        when state2 =>
                temp <= <value>;
                If (συνθήκη) then nx_state <= state3;
                .....
                End If;
        ....
    End Case;
End Process;
```

2.7 Μορφές κωδικοποίησης μηχανής πεπερασμένων καταστάσεων

Οι καταστάσεις ενός FSM μπορούν να κωδικοποιηθούν με διάφορους τρόπους, οι οποίοι περιγράφονται παρακάτω χρησιμοποιώντας το ακόλουθα δεδομένα αριθμητικού τύπου:

```
TYPE state IS (A, B, C, D, E);
```

Ακολουθιακή κωδικοποίηση (Sequential encoding): Ο ελάχιστος αριθμός των bits χρησιμοποιείται και οι καταστάσεις κωδικοποιούνται σε αύξουσα σειρά των δεκαδικών τιμών. Με N bits (N flip-flops), 2^N καταστάσεις μπορούν να κωδικοποιηθούν. Ο παραπάνω τύπος θα έχει τα ακόλουθα ως αποτέλεσμα: A= "000" (=0 decimal), B= "001" (= 1), C= "010" (= 2), D= "011" (= 3) και E= "100" (= 4).

Κωδικοποίηση Gray (Gray encoding): Και πάλι, ο ελάχιστος αριθμός των bits χρησιμοποιείται, με τις καταστάσεις να κωδικοποιούνται χρησιμοποιώντας τον κώδικα Gray. Έτσι οι γειτονικές κωδικοποιημένες λέξεις διαφέρουν κατά ένα μόνο bit. Ο παραπάνω τύπος θα έχει τα ακόλουθα ως αποτέλεσμα: A= "000", B= "001", C= "011", D= "010" και E= "110".

Κωδικοποίηση Johnson (Johnson encoding): Όπως και στον κώδικα Gray, οι γειτονικές κωδικοποιημένες λέξεις διαφέρουν κατά ένα μόνο bit. Ωστόσο, με N bits (N flip-flops), μόνο 2^N καταστάσεις μπορούν να κωδικοποιηθούν. Κάθε νέα κωδικοποιημένη λέξη λαμβάνεται με κυκλική μετατόπιση της προηγούμενης προς τα δεξιά κατά μία θέση, με το νέο MSB (Most Significant Bit) να ισούται με το

αντίστροφο του προηγούμενου LSB (Least Significant Bit). Ο παραπάνω τύπος θα έχει τα ακόλουθα ως αποτέλεσμα: A= "000", B= "100", C= "110", D= "111" και E= "011".

Κωδικοποίηση one-hot (One-hot encoding): Για την κωδικοποιήση N καταστάσεων, χρειάζονται N flip-flops. Κάθε κωδικοποιημένη λέξη περιέχει μόνο ένα bit, το οποίο διαφέρει από τα υπόλοιπα (δηλαδή, όλα τα bits είναι '0' εκτός από ένα ή το αντίστροφο). Ο παραπάνω τύπος θα έχει τα ακόλουθα ως αποτέλεσμα: A= "00001", B= "00010", C= "00100", D= "01000" και E= "10000".

Κωδικοποίηση που ορίζεται από το χρήστη (User-defined encoding): Αυτή η κωδικοποίηση περιλαμβάνει οποιοδήποτε άλλο σύστημα κωδικοποίησης, το οποίο καθορίζεται από το σχεδιαστή.

Η μορφή κωδικοποίησης one-hot χρησιμοποιείται συχνά σε εφαρμογές όπου υπάρχουν άφθονα flip-flops, όπως στα FPGAs, ενώ η ASIC υλοποιείται συχνά με ακολουθιακή κωδικοποίηση (sequential encoding). Στο Σχήμα 2.6 απεικονίζονται οι κύριες μορφές κωδικοποίησης FSM και οι αντίστοιχες κωδικοποιημένες λέξεις με 4 bits.

FSM encodings (for 4-bit systems)					
Sequential		Gray		Johnson	One-hot
0000	1000	0000	1100	0000	0001
0001	1001	0001	1101	1000	0010
0010	1010	0011	1111	1100	0100
0011	1011	0010	1110	1110	1000
0100	1100	0110	1010	1111	---
0101	1101	0111	1011	0111	---
0110	1110	0101	1001	0011	---
0111	1111	0100	1000	0001	---
Total=16 states		Total=16 states		Total=8 states	Total=4 states

Σχήμα 2.6 Κύριες μορφές κωδικοποίησης FSM και οι αντίστοιχες κωδικοποιημένες λέξεις με 4 bits

2.8 Βασικά βήματα σχεδίασης μιας μηχανής πεπερασμένων καταστάσεων

Τα βήματα τα οποία πραγματοποιούνται κατά τη σχεδίαση μιας μηχανής πεπερασμένων καταστάσεων είναι τα εξής:

1. Λήψη των προδιαγραφών του ζητούμενου κυκλώματος.
2. Δημιουργία των καταστάσεων της μηχανής επιλέγοντας καταρχάς μία αρχική κατάσταση. Στη συνέχεια με βάση τις προδιαγραφές του κυκλώματος, θεωρούμε όλους τους δυνατούς συνδιασμούς τιμών εισόδου που μπορεί να λάβει το κύκλωμα και δημιουργούμε νέες καταστάσεις, στις οποίες θα οδηγηθεί η μηχανή με βάση αυτούς τους συνδιασμούς τιμών. Για την παρακολούθηση των καταστάσεων που δημιουργούνται κατασκευάζουμε ένα διάγραμμα καταστάσεων. Όταν αυτό ολοκληρωθεί, το διάγραμμα αυτό θα περιέχει όλες τις καταστάσεις της μηχανής και θα δίνει τις συνθήκες υπό τις οποίες το κύκλωμα θα μεταβαίνει από τη μία κατάσταση στην άλλη.
3. Δημιουργία του πίνακα καταστάσεων μέσω του διαγράμματος κατάστασης. Ενδέχεται να είναι δυνατή η δημιουργία του πίνακα αυτού χωρίς τη μεσολάβηση του διαγράμματος καταστάσεων.
4. Σύνηθες φαινόμενο είναι να χειριζόμαστε κυκλώματα που έχουν μεγάλο αριθμό καταστάσεων και στις περιπτώσεις αυτές είναι απίθανο να λάβουμε τα βέλτιστα αποτελέσματα με την πρώτη κατασκευή του πίνακα καταστάσεων. Αντίθετα, είναι σχεδόν σίγουρο ότι θα κατασκευάσουμε περισσότερες καταστάσεις από όσες είναι αναγκαίο. Η κατάσταση αυτή μπορεί να διορθωθεί από μία διαδικασία η οποία ελαχιστοποιεί τον αριθμό τον καταστάσεων. Η ελαχιστοποίηση του αριθμού των καταστάσεων είναι πολύ σημαντική, επειδή μπορεί να απαιτουνται λιγότερα flip-flops για την παράσταση των καταστάσεων και επομένως η πολυπλοκότητα του τελικού συνδιαστικού κυκλώματος που θα χρησιμοποιηθεί μπορεί να μειωθεί.
5. Αποφασίζουμε για τον αριθμό των μεταβλητών κατάστασης που απαιτούνται για την αντιπροσώπευση όλων των καταστάσεων και εκτελούμε την αντιστοίχηση καταστάσεων. Υπάρχουν πολλοί δυνατοί τρόποι αντιστοίχησης σε κάποιο ακολουθιακό κύκλωμα και μερικοί τρόποι είναι καλύτεροι από άλλους.
6. Γίνεται επιλογή για το είδος των flip-flops που θα χρησιμοποιηθούν στο κύκλωμα. Δημιουργούμε τις λογικές συναρτήσεις επόμενης κατάστασης για τον έλεγχο των εισόδων των flip-flops και στη συνέχεια δημιουργούμε λογικές εκφράσεις για τις εξόδους του κυκλώματος.
7. Υλοποίηση του κυκλώματος με τον τρόπο που δεικνύεται από τις λογικές εκφράσεις.

Κεφάλαιο 3 :
Διατάξεις Πυλών Προγραμματιζόμενων στο Πεδίο (FPGA)

3.1 Εισαγωγή

Τα FPGAs (Field Programmable Gate Array- Διατάξεις Πυλών Προγραμματιζόμενων στο Πεδίο) είναι ψηφιακά ολοκληρωμένα κυκλώματα τα οποία περιέχουν προγραμματιζόμενα μπλοκ ψηφιακής λογικής. Αυτά τα μπλοκ συνδέονται μεταξύ τους με την βοήθεια προγραμματιζόμενων διασυνδέσεων. Τα FPGAs προγραμματίζονται είτε από τον καταναλωτή είτε από τον σχεδιαστή μετά την κατασκευή τους. Ο προγραμματισμός τους πραγματοποιείται κυρίως με τη χρήση μίας γλώσσας περιγραφής υλικού (HDL- Hardware Description Language), δηλαδή είτε VHDL, είτε AHDL, είτε Verilog.

Η ύπαρξη των FPGAs οφείλεται στο κενό που παρατηρήθηκε ότι υπάρχει στο φάσμα των ψηφιακών ολοκληρωμένων κυκλωμάτων στις αρχές της δεκαετίας του '90. Από την μία πλευρα υπήρχαν τα PLDs (Programmable Logic Devices), τα οποία ήταν σε πολύ μεγάλο βαθμό προγραμματιζόμενα και απαιτούσαν πολύ μικρούς χρόνους σχεδίασης (όσον αφορά την διαδικασία σχεδίαση – προγραμματισμός – έλεγχος – διορθώσεις – επαναπρογραμματισμός) αλλά αδυνατούσαν να υλοποιήσουν πολύ μεγάλες ή και πολύπλοκες λογικές συναρτήσεις. Από την άλλη πλευρά, ήταν τα ASICs (Application-Specific Integrated Circuits), τα οποία μπορούσαν να υλοποιήσουν πολύ μεγάλες και πολύπλοκες λογικές συναρτήσεις αλλά ο χρόνος και το κόστος σχεδίασής τους ήταν εξαιρετικά μεγάλα. Επίσης, στα ASICs η τελική σχεδίαση έχει παραμείνει ακόμη στο πυρίτιο. Για να καλυφθεί αυτό το κενό στο φάσμα των ψηφιακών ολοκληρωμένων κυκλωμάτων η εταιρεία Xilinx ανέπτυξε μία νέα κατηγορία ψηφιακών ολοκληρωμένων κυκλωμάτων, τα FPGAs, τα οποία πρωτοεμφανίστηκαν στην αγορά το 1984.

Τα FPGA έχουν κάποια ιδιαίτερα χαρακτηριστικά τα οποία δεν διαθέτουν τα άλλα προγραμματιζόμενα ολοκληρωμένα ψηφιακά κυκλώματα. Αρχικά, στα FPGA όταν διακόπτεται η τροφοδοσία τους, τότε χάνεται και ο προγραμματισμός τους. Για αυτό το λόγο χρειάζονται εξωτερικούς επεξεργαστές ή μνήμες για να μπορούν να

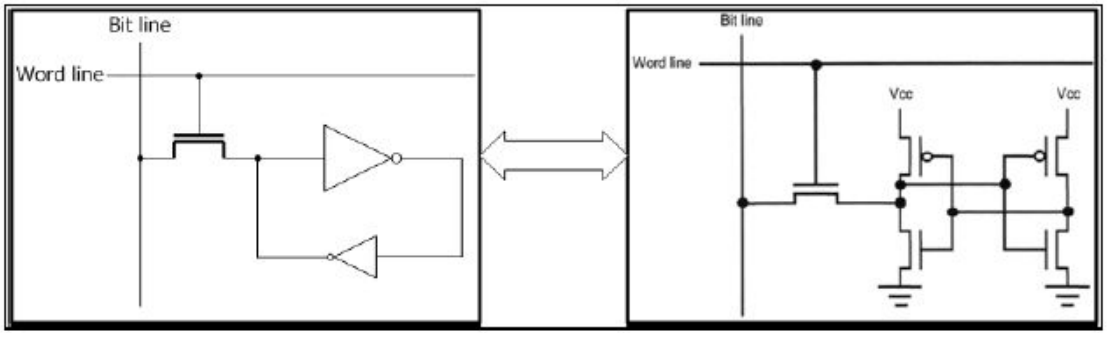
επανέρχονται αυτόματα κάθε φορά που θα επανέρχεται η τροφοδοσία. Ακόμη, ο προγραμματισμός τους μπορεί να αλλάζει κάθε φορά που τροποποιείται το λογισμικό του επεξεργαστή ή τα δεδομένα της μνήμης που τα ελέγχει. Τέλος, μπορεί να επαναπρογραμματιστεί άπειρες φορές.

3.2 Λειτουργία του FPGA

Για την υλοποίηση των FPGAs υπάρχουν πολλές τεχνολογίες. Μερικές από αυτές τις τεχνολογίες είναι οι ακόλουθες: Antifuse, EEPROM, FLASH, SRAM. Η πιο διαδεδομένη από τις τεχνολογίες είναι η SRAM προγραμματιζόμενων στοιχείων (SRAM-based programmable cells), γιατί αυτή χρησιμοποιείται ως επί το πλείστον στις τυπικές εφαρμογές και γιατί η αρχιτεκτονική των FPGAs είναι, σε μεγάλο βαθμό, ανεξάρτητη από την τεχνολογία υλοποίησης.

3.2.1 Τεχνολογία βασισμένη σε SRAM-στοιχεία

Το SRAM-στοιχείο μνήμης, το οποίο χρησιμοποιείται ως δομικό στοιχείο των μνημών SRAM, φαίνεται στο Σχήμα 3.1.

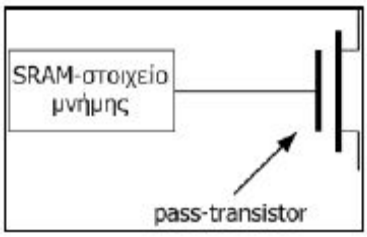


Σχήμα 3.1 Ένα SRAM - στοιχείο μνήμης και η υλοποίηση του με τρανζίστορ

Αυτό το στοιχείο μνήμης μπορεί να αποθηκεύσει πληροφορία του ενός bit. Για να εισαχθεί πληροφορία στο SRAM-στοιχείο μνήμης θα πρέπει να ενεργοποιηθεί η γραμμή Word line για ένα ορισμένο χρονικό διάστημα, τέτοιο ώστε η τιμή που βρίσκεται στην γραμμή Bit line να προλάβει να διαδοθεί μέσω των δύο αντιστροφέων

που ορίζουν τον βρόχο. Η τιμή που θα αποθηκευτεί θα μείνει εκεί για πάντα εκτός αν αντικατασταθεί με καινούργια τιμή. Επιπλέον όταν η τροφοδοσία διακοπεί η τιμή του SRAM-στοιχείου μνήμης θα χαθεί.

Το SRAM-στοιχείο διασύνδεσης(pass-transistor ή SRAM switch) αποτελείται από δύο στοιχεία, το SRAM-στοιχείο μνήμης και το τρανζίστορ(pass transistor). Η λειτουργία του pass-transistor ως ανοικτός ή κλειστός διακόπτης γίνεται ανάλογα την τιμή που έχει αποθηκευτεί στο SRAM-στοιχείο μνήμης. Στο Σχήμα 3.2 απεικονίζεται ένα SRAM-στοιχείο διασύνδεσης (SRAM switch).

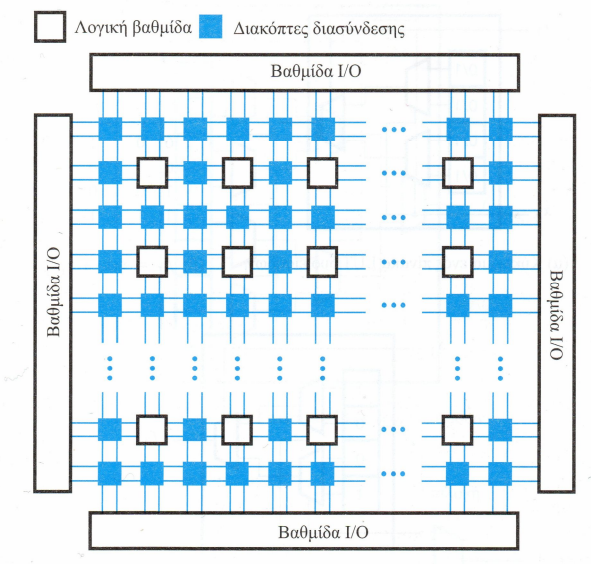


Σχήμα 3.2 SRAM-στοιχείο διασύνδεσης (SRAM switch)

Η τεχνολογία SRAM έχει και μειονεκτήματα αλλά και πλεονεκτήματα. Ένα από τα μειονεκτήματα είναι το μέγεθός του, το οποίο καλύπτει μεγάλο χώρο πάνω στο ολοκληρωμένο. Ο λόγος είναι ότι αποτελείται από πέντε SRAM –στοιχεία μνήμης και ένα pass-transistor. Ακόμα, τα SRAM-στοιχεία μνήμης μπορούν να χρησιμοποιηθούν ως latches, flip-flop ή και ως διαδεδομένη μνήμη μέσα στο ολοκληρωμένο. Επειδή, τα στοιχεία που περιέχονται στην SRAM-στοιχείων μνήμης χάνονται αν η τροφοδοσία διακοπεί, το FPGA θα πρέπει να προγραμματίζεται κάθε φορά που του παρέχεται τροφοδοσία. Βέβαια, με την βοήθεια μίας εξωτερικής μνήμης EEPROM ή FLASH, η οποία κρατάει μόνιμα το επιθυμητό "πρόγραμμα", η διαδικασία προγραμματισμού του FPGA (αφού προγραμματιστεί η εξωτερική μνήμη) διαρκεί περίπου 1 με 2 δευτερόλεπτα, δίνοντας την αίσθηση ότι το FPGA είναι μόνιμα προγραμματισμένο. Με λίγα λόγια ένα SRAM-based FPGA έχει την δυνατότητα να προγραμματίζεται γρήγορα και επαναλαμβανόμενα. Άρα αυτός είναι ο λόγος που το κάνει περιζήτητο για την σχεδίαση νέων εφαρμογών, όπου απαιτούνται επαναλαμβανόμενες δοκιμές και διορθώσεις.

3.3 Δομή της διάταξης FPGA

Η γενική δομή της διάταξης FPGA περιέχει τρία είδη πόρων: λογικές βαθμίδες, βαθμίδες εισόδου / εξόδου για τη σύνδεση με τους ακροδέκτες της συσκευασίας και διακόπτες και γραμμές εσωτερικής διασύνδεσης. Στο Σχήμα 3.3 φαίνεται η κάτοψη της γενικής αρχιτεκτονικής FPGA.



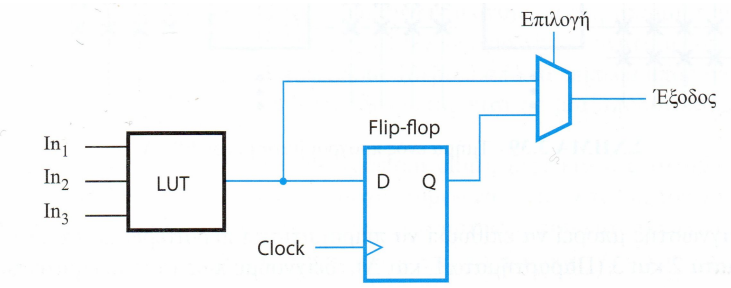
Σχήμα 3.3 Κάτοψη της γενικής αρχιτεκτονικής FPGA

Οι λογικές βαθμίδες οργανώνονται με τη μορφή δισδιάστατης σειράς και οι γραμμές διασύνδεσης οργανώνονται ως οριζόντια και κατακόρυφα κανάλια δρομολόγησης (routing channels) ανάμεσα στις γραμμές και τις στήλες των λογικών βαθμίδων. Τα κανάλια αυτά εμπεριέχουν καλώδια και προγραμματιζόμενους διακόπτες που επιτρέπουν τις λογικές βαθμίδες να διασυνδέονται με πολλούς τρόπους.

Παρατηρώντας το παραπάνω σχήμα, παρουσιάζονται δύο θέσεις προγραμματιζόμενων διακοπτών: Τα τετράγωνα που βρίσκονται δίπλα στις λογικές βαθμίδες περιέχουν διακόπτες που συνδέουν τους ακροδέκτες εισόδου και εξόδου των λογικών βαθμίδων με τα καλώδια διασύνδεσης και τα τετράγωνα που βρίσκονται διαγώνια μεταξύ των λογικών βαθμίδων συνδέουν ένα καλώδιο διασύνδεσης με ένα άλλο (όπως ένα οριζόντιο με ένα κατακόρυφο). Υπάρχουν επίσης προγραμματιζόμενες συνδέσεις ανάμεσα στις βαθμίδες εισόδου / εξόδου και τα καλώδια διασύνδεσης. Ο ακριβής αριθμός προγραμματιζόμενων διακοπτών και καλωδίων σε μία διάταξη FPGA λαμβάνει διάφορες τιμές στα ολοκληρωμένα κυκλώματα του εμπορίου.

Κάθε λογική βαθμίδα ενός FPGA έχει τυπικά ένα μικρό αριθμό εισόδων και μία έδοδο. Σήμερα υπάρχει στο εμπόριο μία σειρά προϊόντων FPGA, τα οποία διαθέτουν διάφορα είδη λογικών βαθμίδων. Η πιο ευρέως χρησιμοποιημένη λογική βαθμίδα είναι ο πίνακας αναζήτησης ή αναφοράς (Lookup table, LUT), ο οποίος περιέχει κυψέλες αποθήκευσης (storage cells) που χρησιμοποιούνται για την υλοποίηση μίας μικρής λογικής συνάρτησης. Κάθε κυψέλη μπορεί να κρατήσει μία λογική τιμή 0 ή 1. Η αποθηκευμένη τιμή μεταφέρεται στην έξοδο της κυψέλης αποθήκευσης. Μπορούν να αναπτυχθούν πίνακες LUT σε διάφορα μεγέθη, όπου το μέγεθος ορίζεται από τον αριθμό των εισόδων.

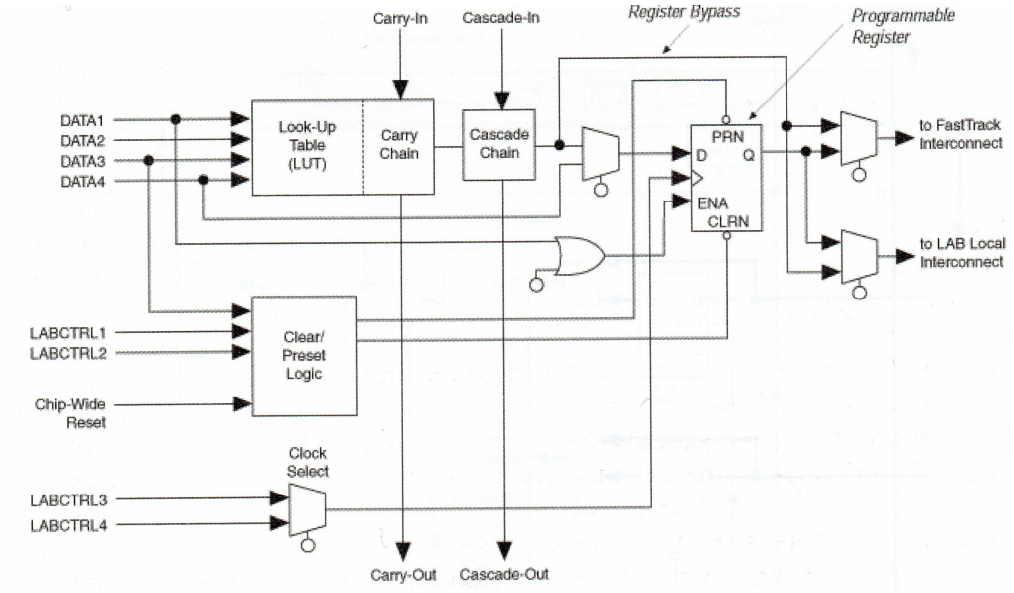
Μία κλασσική λογική βαθμίδα αποτελείται από ένα πίνακα αντιστοίχισης τριών εισόδων (3-input lookup table-LUT) και ένα flip-flop, όπως αναπαρίσταται στο Σχήμα 3.4.



Σχήμα 3.4 Λογική βαθμίδα FPGA με τοποθέτηση flip-flop

Υπάρχει μόνο μία έξοδος, η οποία μπορεί να είναι είτε η εκταμιευμένη (registered) είτε ή μη εκταμιευμένη (unregistered) έξοδος του LUT. Η λογική βαθμίδα έχει 3 εισόδους για τον πίνακα και μία είσοδο ρολογιού. Τα σήματα ρολογιού (και αρκετές φορές και αρκετά άλλα high-fanout σήματα) συνήθως δρομολογούνται μέσα από ειδικά δίκτυα δρομολόγησης (DCM–Digital Clock Management).

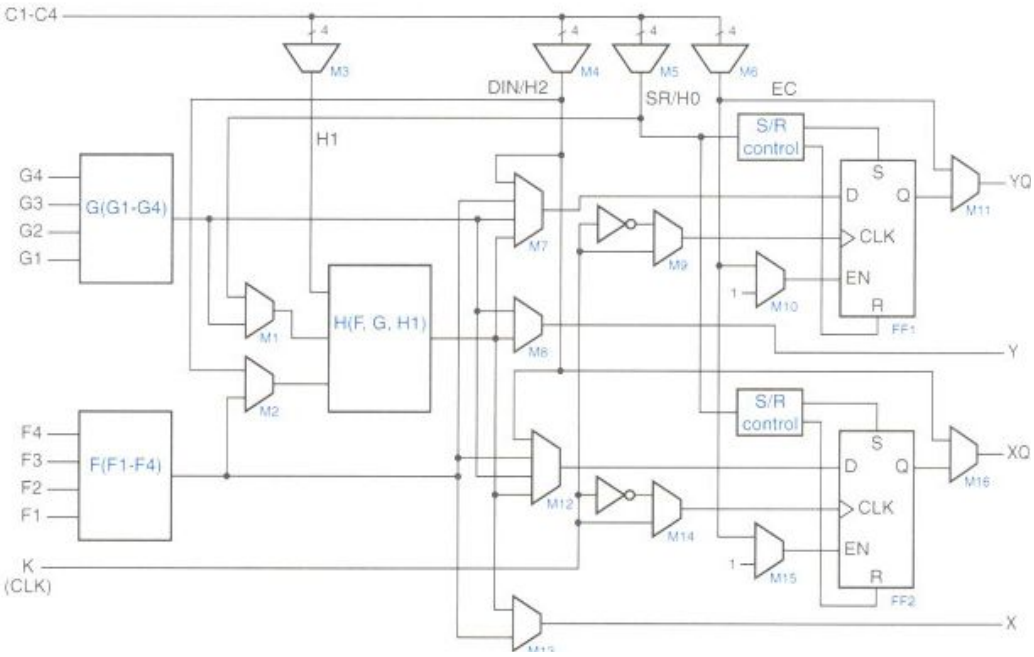
Στο Σχήμα 3.5 απεικονίζεται η δομή ενός λογικού στοιχείου της διάταξης FLEX 10K της εταιρείας Altera, το οποίο διαθέτει έναν πίνακα αναφοράς τεσσάρων εισόδων και ένα flip-flop, το οποίο μπορεί να παρακαμφθεί.



Σχήμα 3.5 Δομή λογικού στοιχείου της διάταξης FLEX 10K της εταιρείας Altera

Για την υλοποίηση των αριθμητικών αθροιστών, οι πίνακες LUT τεσσάρων εισόδων μπορούν να υλοποιήσουν δύο συναρτήσεις τριών εισόδων και ειδικότερα τις συναρτήσεις αθροίσματος και κρατουμένου εξόδου ενός πλήρη αθροιστή.

Η αντίστοιχη λογική βαθμίδα σε διατάξεις FPGA της εταιρείας Xilinx ονομάζεται διαμορφούμενο λογικό μπλοκ (Configurable Logic Block - CBL) ή αλλιώς στην βιβλιογραφία αναφέρεται και ως slice. Η απεικόνισή της φαίνεται παρακάτω στο Σχήμα 3.6.



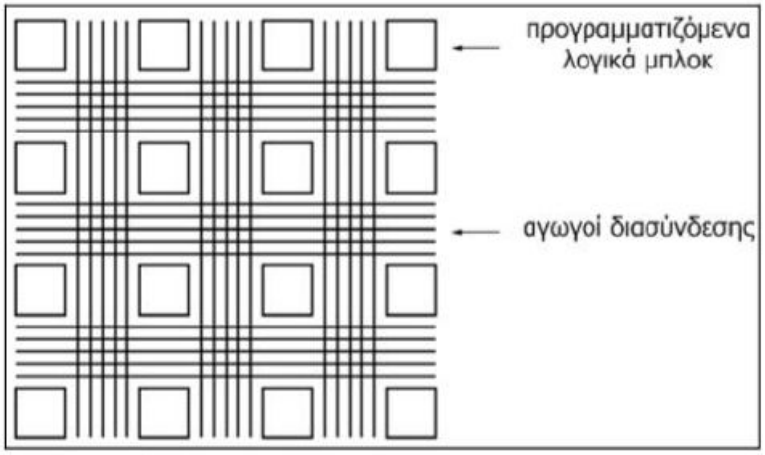
Σχήμα 3.6 Διαμορφούμενη λογική βαθμίδα (CLB) της εταιρείας Xilinx

Όταν ένα κύκλωμα υλοποιείται με τη βοήθεια ενός FPGA τότε οι λογικές βαθμίδες προγραμματίζονται για να υλοποιούν τις αναγκαίες συναρτήσεις και τα κανάλια δρομολόγησης προγραμματίζονται για να εκτελούν τις απαραίτητες διασυνδέσεις ανάμεσα στις λογικές βαθμίδες. Τα FPGA οργανώνονται με τη βοήθεια της μεθόδου προγραμματισμού μέσα στο σύστημα (in-system programming, ISP) δηλαδή εκτελούν τον προγραμματισμό όταν το ολοκληρωμένο κύκλωμα βρίσκεται πάνω στη μητρική πλακέτα. Οι κυψέλες αποθήκευσης των πινάκων LUT των FPGAs είναι πτητικές (ή μη μόνιμες, Volatile) που σημαίνει ότι χάνουν τα δεδομένα που περιέχουν εάν διακοπεί η τροφοδοσία του ολοκληρωμένου κυκλώματος. Επομένως τα FPGA πρέπει να προγραμματίζονται κάθε φορά που εφαρμόζεται τροφοδοσία στο κύκλωμα. Συχνά περιλαμβάνεται στη μητρική πλακέτα μαζί με το FPGA και ένα μικρό ολοκληρωμένο κύκλωμα μνήμης, το οποίο διατηρεί μόνιμα τα δεδομένα της διάταξης και ονομάζεται προγραμματιζόμενη μνήμη μόνον ανάγνωσης (programmable read-only memory, PROM). Οι κυψέλες αποθήκευσης του FPGA φορτώνονται αυτόματα από τη μνήμη PROM, όταν εφαρμόζεται τροφοδοσία στο κύκλωμα.

3.3.1 Αρχιτεκτονική διασύνδεσης

Η αρχιτεκτονική διασύνδεσης είναι ένα πολύ βασικό κομμάτι του FPGA. Σύμφωνα με αυτή γίνεται η τοποθέτηση των προγραμματιζόμενων διασυνδέσεων και των αγωγών πάνω στο FPGA κατά την κατασκευή του. Οι υπηρεσίες που προσφέρει η αρχιτεκτονική διασύνδεσης είναι ο μεγάλος βαθμός διασύνδεσης, μεγάλη ταχύτητα μετάδοσης σημάτων και ακόμα περιλαμβάνει ικανοποιητικό αριθμό αγωγών.

Το ψευδώνυμο που έχει δοθεί στην γενική αρχιτεκτονική των σύγχρονων FPGAs είναι το Island-Style. Τα ``νησιά΄΄ είναι τα προγραμματιζόμενα λογικά μπλοκ τα οποία ``επιπλέουν΄΄ στη ``θάλασσα΄΄ διασυνδέσεων. Στο Σχήμα 3.7 φαίνονται οι αγωγοί διασύνδεσης, οι οποίοι ομαδοποιούνται σε κανάλια και διασχίζουν το ολοκληρωμένο σε οριζόντια και κάθετη μορφή.



Σχήμα 3.7 Η απλοποιημένη μορφή της αρχιτεκτονικής island-style

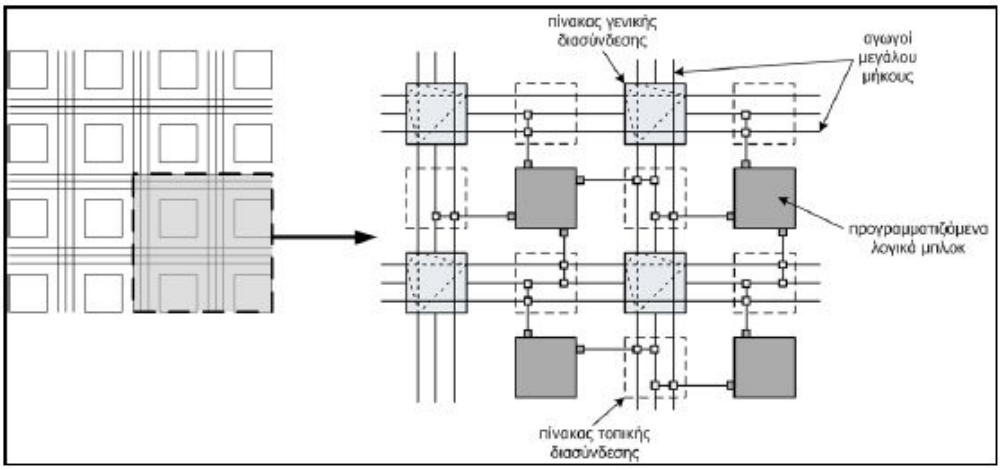
Η αρχιτεκτονική διασύνδεσης διαφέρει λίγο σε σχέση με τα παλαιότερα FPGAs και τα σύγχρονα FPGAs. Τα σύγχρονα FPGAs έχουν σαν χαρακτηριστικό την ιεραρχία και την ποικιλία στο μήκος των αγωγών διασύνδεσης. Ο αγωγός διασύνδεσης αποτελείται από ένα συνεχές αγώγιμο τμήμα που στη αρχή και στο τέλος του υπάρχουν προγραμματιζόμενες διασυνδέσεις. Οι προγραμματιζόμενες διασυνδέσεις υλοποιούνται κυρίως με SRAM-στοιχεία διασύνδεσης και σε κάποιες περιπτώσεις με βαθμίδες απομόνωσης τριών καταστάσεων.

Ο λόγος που το μήκος των αγωγών διασύνδεσης δεν είναι σταθερό είναι διοτι η διάδοση των σημάτων κατά την διασύνδεση απομακρυσμένων προγραμματιζόμενων μπλοκ θα γινόταν με πολύ μεγάλη καθυστέρηση. Οι τύποι αγωγών που υπάρχουν σε κάθε κανάλι είναι οι ακόλουθοι:

- αγωγοί με μήκος ικανό για την διασύνδεση γειτονικών προγραμματιζόμενων λογικών μπλοκ.
- αγωγοί για την διασύνδεση ενός προγραμματιζόμενου λογικού μπλοκ με το μπλοκ που ακολουθεί μετά το γειτονικό και βρίσκεται στην ίδια στήλη ή στην ίδια σειρά των προγραμματιζόμενων λογικών μπλοκ.
- αγωγοί για την διασύνδεση ενός προγραμματιζόμενου λογικού μπλοκ με το τρίτο ή το έκτο μπλοκ της ίδιας σειράς ή στήλης.
- αγωγοί που διατρέχουν όλο το ολοκληρωμένο για διασύνδεση κάθε προγραμματιζόμενου λογικού μπλοκ με κάποιο ή κάποια άλλα προγραμματιζόμενα λογικά μπλοκ, τα οποία ανήκουν στην ίδια σειρά ή στήλη.

Οι προαναφερόμενες διασυνδέσεις υλοποιούνται μέσω των πινάκων τοπικής και γενικής διασύνδεσης.

Εκτός από τους αγωγούς διασύνδεσης, η αρχιτεκτονική διασύνδεσης περιλαμβάνει ακόμα δύο δομές. Τους πίνακες τοπικής και γενικής διασύνδεσης, οι οποίοι απεικονίζονται στο Σχήμα 3.8. Οι πίνακες τοπικής διασύνδεσης βρίσκονται συνήθως στις τέσσερις πλευρές κάθε προγραμματιζόμενου λογικού μπλοκ και περιέχουν προγραμματιζόμενες διασυνδέσεις μέσω των οποίων συνδέουν τους ακροδέκτες ενός γειτονικού προγραμματιζόμενου λογικού μπλοκ με αγωγούς που οδηγούν σε πίνακες γενικής διασύνδεσης. Ο πίνακας γενικής διασύνδεσης βρίσκεται στις διασταυρώσεις των οριζόντιων και κάθετων καναλιών συνδέοντας με προγραμματιζόμενες διασυνδέσεις αγωγούς των οποίων τα άκρα καταλήγουν εκεί. Κάθε αγωγός που ανήκει σε ένα κανάλι μπορεί να συνδεθεί μόνο σε ένα υποσύνολο των αγωγών του κάθετου καναλιού. Αυτό συμβαίνει λόγω του μεγάλου μεγέθους και της μεγάλης χωρητικότητας C των προγραμματιζόμενων διασυνδέσεων.



Σχήμα 3.8 Διασύνδεση των προγραμματιζόμενων λογικών μπλοκ

Κεφάλαιο 4 :
Αναπτυξιακή Πλακέτα LP-2900

4.1 Εισαγωγή στην αναπτυξιακή πλακέτα LP-2900

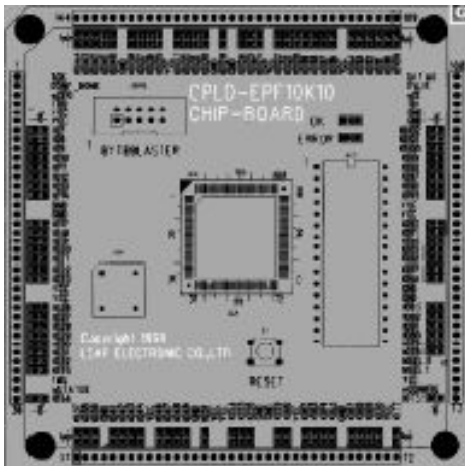
Η πρώτη αναπτυξιακή πλακέτα που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής μας είναι η LP-2900 της εταιρείας Leap Electronic Co. Το αναπτυξιακό κύκλωμα LP-2900 βασίζεται στο ολοκληρωμένο FPGA EPF10K10TC144-4, το οποίο ανήκει στην οικογένεια FLEX10K της εταιρείας Altera. Διαθέτει 144 pins εισόδου/εξόδου από τα οποία τα 102 είναι ελεύθερα για χρήση και αποτελείται από 576 λογικά στοιχεία (logic elements) και 61444 bits μνήμης. Στο Σχήμα 4.1 που ακολουθεί φαίνεται το αναπτυξιακό κύκλωμα LP-2900, το οποίο είναι κατάλληλο για ανάπτυξη και επαλήθευση απλών πρωτότυπων ψηφιακών κυκλωμάτων.

Σχήμα 4.1 Η αναπτυξιακή πλακέτα LP-2900

Το αναπτυξιακό κύκλωμα LP-2900 χωρίζεται στα εξής μέρη: στο μέρος που περιέχεται το FPGA, στο μέρος της τροφοδοσίας, στις συσκευές εισόδου/εξόδου και στη θύρα επικοινωνίας με τον υπολογιστή. Επίσης, είναι εφοδιασμένο με μια σειρά περιφερειακών, όπως:

- 4 σετ από κόκκινα, κίτρινα και πράσινα leds κοινής καθόδου
- ενδείκτες επτά τομέων (7 segment displays) κοινής ανόδου
- 1 buzzer
- 2 ηλεκτρονικά ζάρια (dices) κοινής καθόδου
- 8 διακόπτες τύπου push button
- 2 X 8 διακόπτες τύπου dip switches
- 4 διακόπτες παλμών
- 1 8 X 8 dot matrix display
- 1 πληκτρολόγιο 4 X 3
- 1 οθόνη LCD
- 1 μετατροπέα σήματος από αναλογικό σε ψηφιακό (A / D και D / A)
- 1 ολοκληρωμένο κύκλωμα 8051 (microprocessor)

Στο υποκύκλωμα όπου είναι τοποθετημένο το FPGA (Chipboard), το οποίο απεικονίζεται στο Σχήμα 4.2, εκτός από τη διάταξη FPGA υπάρχει μία EPROM (Erasable Programmable Read-Only Memory), ένας διακόπτης reset και 144 μικρά leds που είναι συνδεδεμένα στους ακροδέκτες του FPGA ώστε να μπορούμε να δούμε ποιοί από αυτους χρησιμοποιούνται αλλά και ποιά είναι η τιμή τους (1 ή 0).



Σχήμα 4.2 Chipboard της πλακέτας LP-2900

Η σύνδεση της αναπτυξιακής πλακέτας με τον υπολογιστή γίνεται μέσω ενός καλωδίου όπου το ένα άκρο του το συνδέουμε στη θύρα του αναπτυξιακού και το άλλο στην παράλληλη θύρα του υπολογιστή. Τίθεται αναγκαία η εγκατάσταση των οδηγών BYTE BLASTER της Altera για να εγκατασταθεί στο Quartus II η κατάλληλη θύρα προγραμματισμού (LPT1). Στο παρακάτω Σχήμα 4.3 φαίνεται το διάγραμμα βαθμίδων του αναπτυξιακού κυκλώματος LP-2900.

Σχήμα 4.3 Διάγραμμα βαθμίδων του αναπτυξιακού κυκλώματος LP-2900

4.2 Συσκευές εισόδου/εξόδου που χρησιμοποιήθηκαν στην εφαρμογή

Το αναπτυξιακό κύκλωμα αν και διαθέτει πολλές συσκευές εισόδου/εξόδου θα αναφερθούμε μόνο σε αυτές που χρησιμοποιήθηκαν στην εφαρμογή. Είναι οι ακόλουθες:

- **LED κοινής καθόδου**

Code	L1	L2	L3	L4	L5	L6	L7	L8
Device	Red	Yellow	Green	Red	Yellow	Green	Red	Yellow
Pin	Pin 7	Pin 8	Pin 9	Pin 10	Pin 11	Pin 12	Pin 13	Pin 14

Code	L9	L10	L11	L12	LED_COM
Device	Green LED	Red LED	Yellow LED	Green LED	Common cathode of LED1～LED12
Pin	Pin 17	Pin 18	Pin 19	Pin 20	Pin 141

Επειδή τα leds είναι κοινής καθόδου, για να δούμε τα αποτελέσματα σε αυτά θα πρέπει να φέρουμε την κοινή κάθοδο των leds στο δυναμικό της γης. Η κοινή κάθοδος συνδέεται μέσω ενός αντιστροφέα με τον ακροδέκτη 141 του FPGA.

Εκτός από τα leds L1 εώς L12 υπάρχει στην πλακέτα του FPGA (Chipboard) ένας μεγάλος αριθμός από SMD led (102 συνολικά). Η κάθοδος των led αυτών είναι μόνιμα συνδεδεμένη στο δυναμικό της γης και ανάβουν στον αντίστοιχο ακροδέκτη του led λογικό 1. Ορισμένα τέτοια led που χρησιμοποιούνται είναι αντιστοιχισμένα στα pin 7 εώς και 14, 17 εώς και 23, 26 εώς και 33 και 95 εώς και 102.

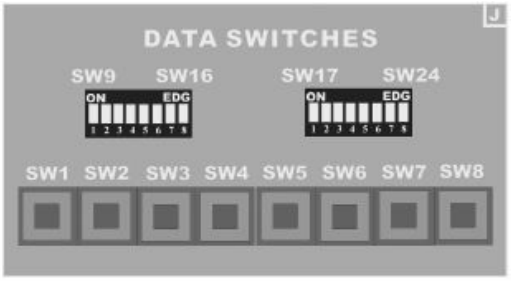
- **Ενδέκτες 7 τομέων (7 Segment Display)**

Code	A	B	C	D	E	F	G	DP
Device	7 Segment Display							
Pin	Pin 23	Pin 26	Pin 27	Pin 28	Pin 29	Pin 30	Pin 31	Pin 32

Code	DE1	DE2	DE3	—	—	—	—	—
Device	74138			—	—	—	—	—
Pin	Pin33	Pin36	Pin37	—	—	—	—	—

Η επιλογή για το ποιός ενδείκτης επτά τομέων (7 Segment Display) θα εμφανίσει τα αποτελέσματα γίνεται από έναν αποκωδικοποιητή 3 προς 8 (74HCT138) του οποίου οι είσοδοι (DE1, DE2, DE3) συνδέονται με τους ακροδέκτες 33, 36 και 37.

- **Διακόπτες Δεδομένων**



Code	SW1	SW2	SW3	SW4	SW5	SW6	SW7	SW8
Device	Push Button							
Pin	Pin 47	Pin 48	Pin 49	Pin 51	Pin 59	Pin 60	Pin 62	Pin 63

Code	SW9	SW10	SW11	SW12	SW13	SW14	SW15	SW16
Device	Dip Switch							
Pin	Pin 64	Pin 65	Pin 67	Pin 68	Pin 69	Pin 70	Pin 72	Pin 73

Code	SW17	SW18	SW19	SW20	SW21	SW22	SW23	SW24
Device	Dip Switch							
Pin	Pin 78	Pin 79	Pin 80	Pin 81	Pin 82	Pin 83	Pin 86	Pin 87

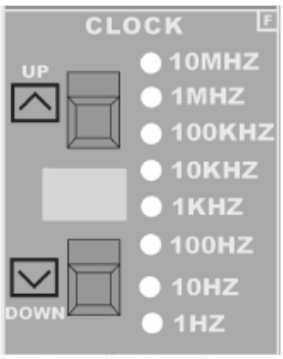
Οι διακόπτες SW1 εώς SW8 είναι διακόπτες τύπου push-button, ενώ οι διακόπτες SW9 ως SW24 αντιστοιχούν σε διακόπτες τύπου dip switches.

- **Απεικόνιση σε οθόνη υγρών κρυστάλλων (LCD Display)**

Code	EN	RS	RW	D0	D1	D2	D3	D4
Device	LCD							
Pin	Pin 130	Pin 122	Pin 128	Pin 131	Pin 132	Pin 133	Pin 135	Pin 136

Code	D5	D6	D7	—	—	—	—	—
Device	LCD			—	—	—	—	—
Pin	Pin 137	Pin 138	Pin 140	—	—	—	—	—

- **7 Clock 10MHz**



Code	L27	L28	L29	L30	L31	L32	L33	L34
Device	Yellow LED							
Pin	Pin 23	Pin 26	Pin 27	Pin 28	Pin 29	Pin 30	Pin 31	Pin 32

Code	DE1	DE2	DE3
Device	74138		
Pin	Pin 33	Pin 36	Pin 37

Code	OSC	UP	DOWN
Device	OSC	Button	Button
Pin	Pin 55	Pin 121	Pin 125

Στην αναπτυξιακή πλακέτα υπάρχει ένας ενσωματωμένος ταλαντωτήςστα 10MHz που συνδέεταιο σαν είσοδος (OSC) στο Pin 55 και μπορεί να χρησιμοποιηθεί σαν Clock στα ακολουθιακά κυκλώματα. Ο χρήστης μπορεί να πάρει από τον ταλαντωτή αυτό μικρότερες συχνότητες υλοποιώντας σε VHDL διαιρέτες συχνότητας.

Υπάρχει, επίσης και μία σειρά από 8 πορτοκαλί leds (L23 εώς L34) για την ένδειξη της συχνότητας και δύο διακόπτες (UP, DOWN) για την αύξηση ή μείωση της συχνότητας.

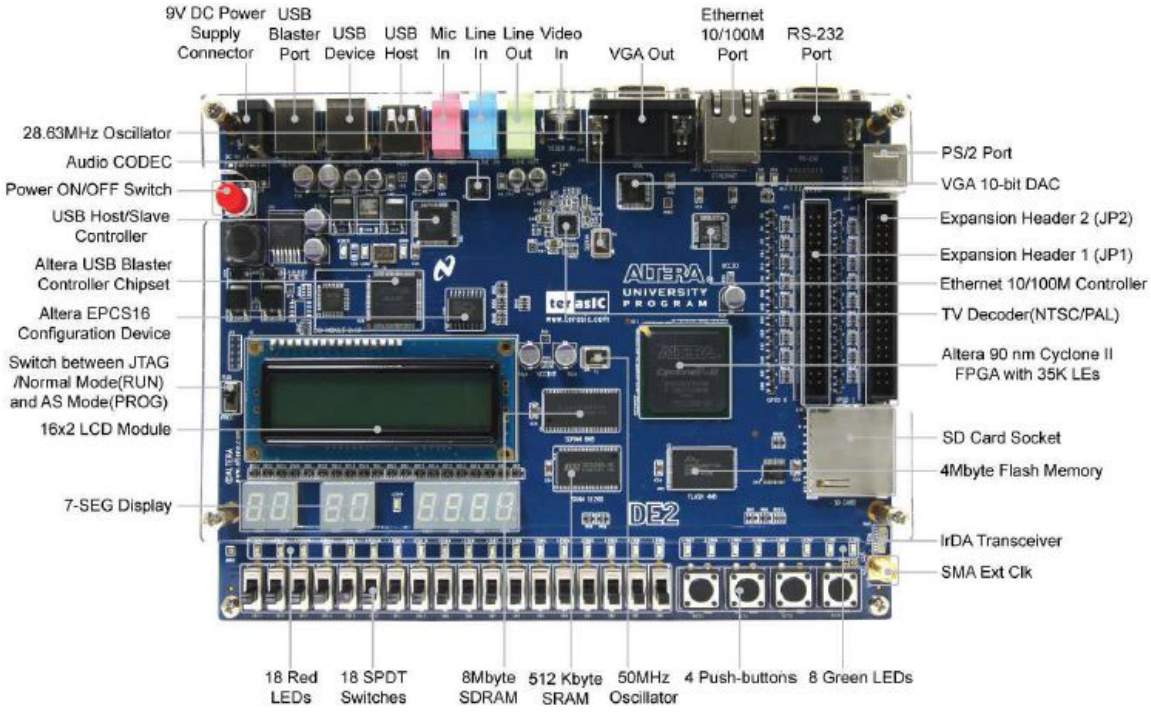
Οι κοινές κάθοδοι των led L23 εώς L34 συνδέονται στην άξοδο Y6 ενός αποκωδικοποιητή 3 προς 8 (74HCT138). Οι είσοδοι του αποκωδικοποιητή είναι σήματα DE1, DE2, DE3 και για την ενεργοποίηση της εξόδου Y6 θα πρέπει να έχουν τιμή DE1=0, DE2=1, DE3=1.

Κεφάλαιο 5 :

Αναπτυξιακή Πλακέτα DE2 της ALTERA

5.1 Εισαγωγή στην αναπτυξιακή πλακέτα DE2

Η αναπτυξιακή και εκπαιδευτική πλακέτα DE2 της Altera είναι η δεύτερη πλακέτα που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής μας, η οποία απεικονίζεται στο Σχήμα 5.1, έχει αναπτυχθεί για την εκμάθηση της ψηφιακής λογικής και την υπολογιστική οργάνωση σε εργαστηριακό περιβάλλον. Σχεδιάστηκε για να χρησιμοποιείται σε εργαστηριακά ακαδημαϊκά μαθήματα που έχουν ως στόχο τον σχεδιασμό ψηφιακών λογικών κυκλωμάτων και την οργάνωση του υπολογιστή. Επιπλέον, η πλακέτα DE2 εκτός από την υλοποίηση πολυπλοκων και απαιτητικών κυκλωμάτων, παρέχει τη δυνατότητα επικοινωνίας με μία πληθώρα περιφρειακών.



Σχήμα 5.1 Αναπτυξιακή Πλακέτα DE2 της ALTERA

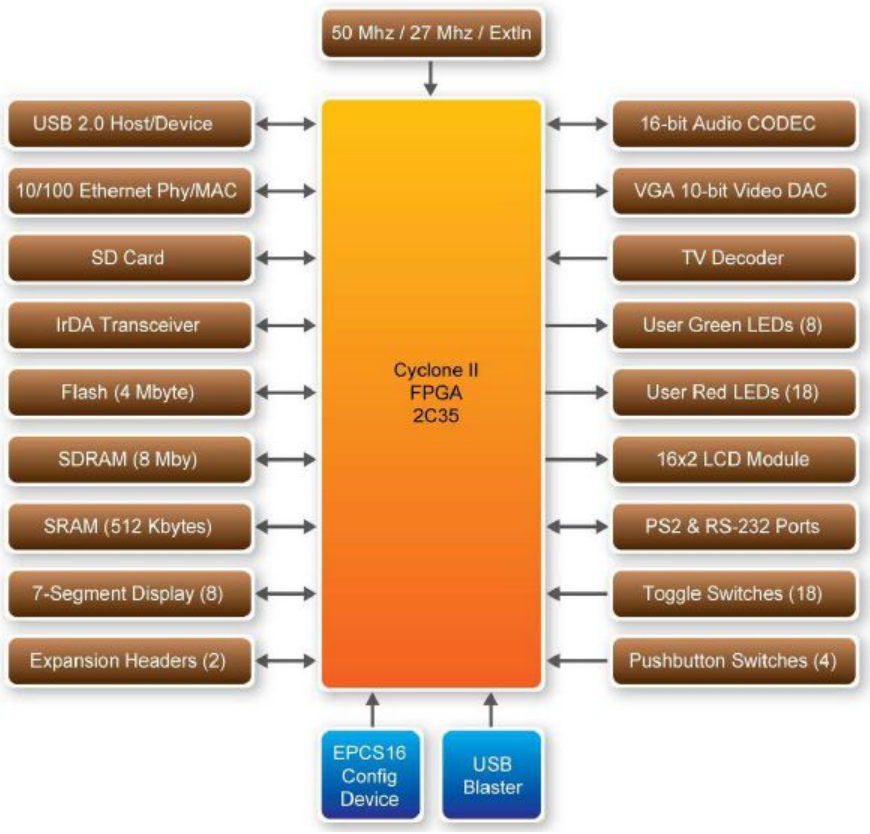
5.2 Περιγραφή της αναπτυξιακής πλακέτας DE2

Η διάταξη της πλακέτα DE2, η οποία φαίνεται παραπάνω, αποτελείται από πολλές διασυνδέσεις και υπσυστήματα. Όμως η καρδιά του συστήματος αυτού, είναι το Cyclone II FPGA chip. Όλα τα σημαντικά επιμέρους υποσυστήματα στην πλακέτα συνδέονται με τους ακροδέκτες του τσιπ, επιτρέποντας στο χρήστη τη ρύθμιση των διασυνδέσεων μεταξύ των διαφόρων υποσυστημάτων όπως επιθυμεί.

Για τη διασύνδεση και τη χρήση της πλακέτας είναι απαραίτητη η σύνδεση με υπολογιστή μέσω της θύρας USB. Γι'αυτό χρειάζεται, συνήθως, η εγκατάσταση στον υπολογιστή των USB-Blaster drivers κατά την εγκατάσταση του προγράμματος Quartus II, μέσω του οποίου γίνεται ο σχεδιασμός συστημάτων. Οι USB-Blaster drivers χρησιμοποιούνται για τον προγραμματισμό των FPGA και αυτό μπορεί να γίνει με δύο τρόπους:

1) Απευθείας από Quartus II: Σε αυτήν την περίπτωση, ο διακόπτης πρέπει να είναι στη θέση RUN, έτσι ώστε τα δεδομένα διαμόρφωσης να πηγαίνουν απευθείας στην εσωτερική μνήμη διαμόρφωσης SRAM του FPGA του. Ο τύπος του αρχείου που χρησιμοποιείται σε αυτή την περίπτωση ονομάζεται SOF (SRAM αρχείο αντικειμένου). Δεδομένου ότι η SRAM είναι πτητική, η διαμόρφωση χάνεται όταν η πλακέτα δεν τροφοδοτείται.

2) Με μια εξωτερική μνήμη διαμόρφωσης: Σε αυτήν την περίπτωση, ο διακόπτης πρέπει να είναι στη θέση PROG, έτσι ώστε τα δεδομένα διαμόρφωσης να αποστέλλονται σε μια εξωτερική μη πτητική μνήμη. Η τροφοδοσία πρέπει να απενεργοποιηθεί και στη συνέχεια να ενεργοποιηθεί για να προκαλέσει τα δεδομένα διαμόρφωσης να ανακτηθούν αυτόματα από την εν λόγω μνήμη από τον ελεγκτή του FPGA. Ο τύπος του αρχείου που χρησιμοποιείται στην περίπτωση αυτή ονομάζεται POF (προγραμματιστής αρχείου αντικειμένου).



Σχήμα 5.2 Block διάγραμμα των περιεχόμεων της αναπτυξιακής πλακέτας DE2

Στο παραπάνω block διάγραμμα του Σχήματος 5.2, απεικονίζονται τα περιεχόμενα της αναπτυξιακής πλακέτας DE2, τα οποία αναλυτικά είναι τα εξής:

- **Cyclone II 2C35 FPGA:** περιέχει 33.216 λογικά στοιχεία, 105 M4K RAM blocks, συνολικά 483.840 bits RAM, 35 ενσωματωμένους πολλαπλασιαστές, 4 PLLs, 475 I/O pins (ακροδέκτες εισόδου/εξόδου) για τον χρήστη και πακέτο FineLine BGA 672 pins.
- **Σειριακή διάταξη μορφοποίησης και κύκλωμα USB Blaster:** Η πλακέτα υποστηρίζει παράλληλη και σειριακή (JTAG και AS) μέθοδο προγραμματισμού, εμπεριέχει σειριακή διάταξη μορφοποίησης, την EPCS16 της Altera καθώς και USB Blaster, εγκατεστημένο πάνω στην αναπτυξιακή πλακέτα για προγραμματισμό και για έλεγχο χρήστη API.
- **Μνήμη Static RAM (SRAM):** μεγέθους 512-Kbyte, οργανωμένη ως 256K x 16 bits.

- **Synchronous Dynamic RAM (SDRAM):** δυναμική μνήμη RAM των 8Mbyte, οργανωμένη ως 1M x 16 bits x 4 banks.

- **Μνήμη Flash:** είναι μεγέθους 4Mbyte (1 Mbyte σε μερικές πλακέτες) με 8bit δίαυλο δεδομένων.
  Όλες οι μνήμες είναι διαθέσιμες από τον επεξεργαστή Nios II και από το DE2 Control Panel.
- **Υποδοχη κάρτας SD:** παρέχει SPI και 4bit SD λειτουργία για πρόσβαση σε SD κάρτες και είναι προσβάσιμη ως μνήμη για τον επεξεργαστή Nios II, μέσω της εγκατάστασης του DE2 SD Card Driver.

- **Pushbutton switches:** Στην πλακέτα παρέχονται 4 διακόπτες pushbutton που βρίσκονται κανονικά σε υψηλή στάθμη (λογικό 1). Όταν όμως πιέζονται τα pushbuttons, παράγεται παλμός σε χαμηλή στάθμη, λογικό 0 (active-low).

- **Toggle switches:** Επίσης υπάρχουν 18 διακόπτες εναλλαγής για είσοδο από τον χρήστη. Ένας διακόπτης παράγει λογικό 0 όταν είναι στη θέση "down" (πιο κοντά στην άκρη της πλακέτας) και λογικό 1 όταν είναι στη θέση "up".

- **Οθόνη LCD:** μεγέθους 16 x 2 για την προβολή του κειμένου.

- **LEDs:** Για τις ανάγκες εξόδου υπάρχουν φωτεινοί ενδείκτες (LEDs). Συγκεκριμένα 8 πράσινα LEDs και 18 κόκκινα.

- **Είσοδοι ρολογιού:** Υπάρχουν δύο κρύσταλλοι χρονισμού, 50MHz ταλάντωσης και 27MHz πηγή ρολογιού και μέσω SMA έχει και εξωτερική πηγή ρολογιού.

- **Audio CODEC:** Για τον ήχο υπάρχει ένα audio CODEC (Wolfson WM8731) των 24 bits sigma-delta, με γραμμή εισόδου, γραμμή εξόδου και υποδοχές για μικρόφωνο. Η συχνότητα δειγματοληψίας κυμαίνεται από 8 έως 96KHz.

- **Έξοδος VGA:** χρησιμοποιείται ο VGA μετατροπέας, ADV7123, των 140MHz με triple 10bit υψηλής ταχύτητας video DAC. Με υποδοχή D-sub με 15 ακροδέκτες υψηλής πυκνότητας, υποστηρίζει αναλύσεις έως και 1600 x 1200 με ρυθμό ανανέωσης 100HZ. Μπορεί να χρησιμοποιηθεί με το Cyclone

II FPGA για την υλοποίηση ενός υψηλής απόδοσης κωδικοποιητή τηλεόρασης.

- **Αποκωδικοποιητής τηλεόρασης NTSC/PAL:** χρησιμοποιείται ο αποκωδικοποιητής ADV7180 Multi-format και υποστηρίζει παγκοσμίως NTSC/PAL/SECAM έγχρωμη αποδιαμόρφωση. Επίσης, υποστηρίζει υποδοχή σύνθετου σήματος Video (CVBS) και ψηφιακές μορφές εξόδου (8bit/ 16bit).

- **10/100 Ethernet ελεγκτής:** υποστηρίζει πλήρη αμφίδρομη επικοινωνία με ταχύτητες 10Mb/s και 100Mb/s και είναι συμβατό με την προδιαγραφή IEEE 802.3u

- **USB Host/Slave ελεγκτής:** που είναι συμβατό με το Universal Serial Bus Specification Rev. 2.0, υποστηρίζει μεταφορά δεδομένων σε υψηλή αλλά και χαμηλή ταχύτητα και μπορεί να λειτουργήσει και ως USB host και ως συσκευή. Υπάρχουν δύο θύρες USB που παρέχουν υψηλής ταχύτητας παράλληλη διασύνδεση στους περισσότερους επεξεργαστές και υποστηρίζει προγραμματισμένη είσοδο/έξοδο (PIO) και άμεση προσπέλαση μνήμης (DMA).

- **Σειριακές θύρες:** Υπάρχουν δύο σειριακές θύρες. Μία σειριακή θύρα είναι η RS-232, για την οποία υπάρχει ο σειριακός σύνδεσμος DB-9, και η δεύτερη είναι η PS/2, στην οποία συνδέονται σειριακά ένα ποντίκι ή πληκτρολόγιο PS2 πάνω στην αναπτυξιακή πλακέτα DE2.

- **Πομποδέκτης IrDA:** Είναι ένας πομποδέκτης υπερύθρων με ρυθμό μετάδοσης δεδομένων 115,2 kb/s, με μετάδοση ρεύματος στα LED στα 32mA, ολοκληρωμένη ασπίδα EMI, IEC825-1 Class 1 eye safe και είσοδο Edge detection.

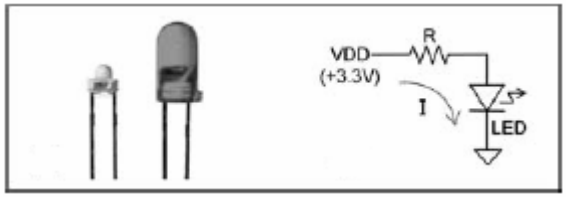
- **Δύο κεφαλές επέκτασης των 40-ακροδεκτών (pins):** 72 ακροδέκτες εισόδου εξόδου (I/O) της Cyclone II, καθώς και 8 γραμμές τροφοδοσίας και γείωσης, εξέρχονται σε δύο υποδοχές επέκτασης με 40 pins. Κεφαλή με 40 pins είναι σχεδιασμένη για να δεχτεί μία τυπική καλωδιοταινία των 40 pin που χρησιμοποιείται για IDE σκληρούς δίσκους.

Κεφάλαιο 6 :
Οθόνες απεικόνισης λογικών καταστάσεων

Η γλώσσα περιγραφής υλικού VHDL χρησιμοποιείται για την περιγραφή συμπεριφοράς απλών ελεγκτών, ψηφιακών υπολογιστικών κυκλωμάτων και μεγάλων κυκλωμάτων σε ψηφίδες, τα οποία συνδέονται συχνά με οθόνες απεικόνισης λογικών καταστάσεων, απλών ψηφίων ή και χαρακτήρων. Τέτοιες οθόνες απεικονίσεις είναι: οι απλές συστοιχίες LEDs, οι απεικονίσεις επτά τομέων (SSD) και οι οθόνες LCD.

6.1 Απλές συστοιχίες LEDs

Η Δίοδος Εκπομπής Φωτός, (LED, Light Emitting Diode) είναι ένας ημιαγωγός, ο οποίος εκπέμπει φωτεινή ακτινοβολία στενού φάσματος όταν του παρέχεται μία ηλεκτρική τάση κατά τη φορά ορθής πόλωσης. Το χρώμα του φωτός που εκπέμπεται εξαρτάται από την χημική σύσταση του ημιαγώγιμου υλικού που χρησιμοποιείται, και μπορεί να είναι υπεριώδες, ορατό ή υπέρυθρο. Στο Σχήμα 6.1 που ακολουθεί, παρουσιάζονται παραδείγματα των LEDs που υπάρχουν στο εμπόριο.

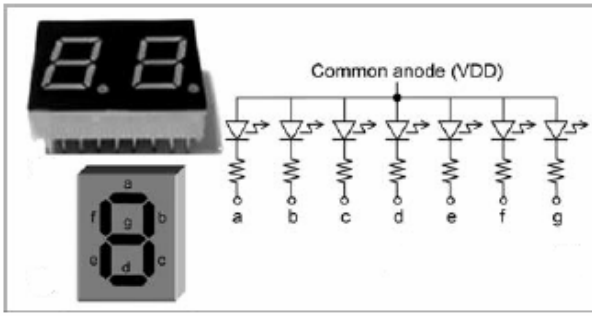


Σχήμα 6.1 Παραδείγματα των LEDs του εμπορίου

6.2 Ενδείκτης επτά τομέων (SSD)

Ο ενδείκτης επτά τομέων (SSD, seven-segment-display) μετατρέπει ένα δεκαδικό ψηφίο σε σήματα που οδηγούν τις διόδους φωτοεκπομπής (LEDs) του ενδείκτη. Οι δίοδοι αυτοί συμβολίζονται με γράμματα a εως g. Ο μετατροπέας αυτός αντιστοιχίζει κάθε συνδιασμό των εισόδων στον αντίστοιχο αριθμό του δεκαδικού συστήματος. Θέτει δηλαδή σε λειτουργία τα απαραίτητα LEDs ώστε να σχηματιστεί ο αριθμός.

Στο Σχήμα 6.2 απεικονίζεται ο ενδείκτης επτά τομέων καθώς και ο πίνακας, ο οποίος δείχνει τα τμήματα που πρέπει να φωτίζονται από 0 εώς F.



char	abcdefg	char	abcdefg
0	"0000001" (1)	8	"0000000" (0)
1	"1001111" (79)	9	"0000100" (4)
2	"0010010" (18)	A	"0001000" (8)
3	"0000110" (6)	b	"1100000" (96)
4	"1001100" (76)	C	"0110001" (49)
5	"0100100" (36)	d	"1000010" (66)
6	"0100000" (32)	E	"0110000" (48)
7	"0001111" (15)	F	"0111000" (56)

Σχήμα 6.2 Ενδείκτης επτά τομέων

6.3 Οθόνη υγρών κρυστάλλων (LCD)

Η οθόνη υγρών κρυστάλλων (LCD, Liquid Crystal Display) περιέχει δύο γραμμές των 16 χαρακτήρων η καθεμία. Στο πίσω μέρος της οθόνης υπάρχει ένας ελεγκτής (HD44780U, της Hitachi) που ενεργεί ως διεπαφή μεταξύ της οθόνης LCD και του εξωτερικού κόσμου. Αυτός ο ελεγκτής μπορεί να προσεγγιστεί μέσω 16 ακίδων, οι οποίες περιλαμβάνουν τροφοδοσία, γραμμές αντίθεσης, ελέγχου και δεδομένων.

6.3.1 Περιγραφή του ελεγκτή (controller) της οθόνης LCD

Οι οθόνες LCD(απεικόνισης υγρών κρυστάλλων) διαθέτουν δύο γραμμές με 16 χαρακτήρες η καθεμιά από αυτές. Μπορούν να χωριστούν σε δύο ομάδες: σε αυτές που έχουν έναν ενσωματωμένο ελεγκτή και έναν οδηγό τσιπ και σε αυτές που έχουν μόνο ένα οδηγό τσιπ. Οι οθόνες που δεν έχουν ελεγκτή χρησιμοποιούνται τυπικά με ισχυρο hardware, στη δική μας περίπτωση όμως η LCD οθόνη έχει ενσωματωμένο στο πίσω μέρος της έναν ελεγκτή, τον HD44780U (Hitachi), ο οποίος διαθέτει 16 pins και είναι υπεύθυνος για τη σωστή απεικόνιση των χαρακτήρων και τη λειτουργία της. Τα τρία πρώτα pins παρέχουν ρεύμα στη μονάδα LCD. Το pin1 είναι το GND και θα πρέπει να στηρίζεται στην παροχή ρεύματος. Το pin2 είναι το Vcc και θα πρέπει να συνδεθεί με ρεύμα 3V-5V. Το pin3 είναι αυτό που ρυθμίζει την αντίθεση της οθόνης. Τα τρία επόμενα pins είναι οι γραμμές ελέγχου της LCD. Το pin4 είναι η RS (Register Select) , η οποία δείχνει εάν θέλουμε να επιλέξουμε εντολές του

καταχωρητή του ελεγκτή, όταν βρίσκεται στην κατάσταση ‘0’ ή να επιλέξουμε δεδομένα του καταχωρητή , όταν βρίσκεται στην κατάσταση ‘1’. Το pin5 είναι η R/W (Read/Write), η οποία δείχνει εάν θέλουμε να διαβάσουμε, όταν βρίσκεται σε κατάσταση ‘1’ ή να γράψουμε όταν βρίσκεται σε κατάσταση ‘0’, στον καταχωρητη του ελεγκτή. Το pin6 είναι η E (Enable), η οποία πρέπει να είναι επιλεγμένη εάν θέλουμε να διαβάσουμε ή να γράψουμε οτιδήποτε. Τα pins7-14 είναι οι γραμμές δεδομένων, οι οποίες αποτελούνται από διαύλους των 8-bit, των οποίων τα περιεχόμενα (δεδομένα ή εντολές) είναι γραμμένα ή διαβάζονται από τον καταχωρητή του ελεγκτή ανάλογα με την κατάσταση του R/W. Τα παραπάνω είναι σήματα τα οποία στέλνονται στον ελεγκτή. Το κυριότερο από αυτά είναι η Busy Flag, το οποίο παρέχεται στον ελεγκτή μέσω του bit DB7 του διαύλου δεδομένων. Όταν η BF είναι ‘1’, τότε ο ελεγκτής είναι σε κατάσταση εσωτερικής λειτουργίας και η επόμενη εντολή δεν θα γίνει δεκτή. Η επόμενη εντολή θα γραφεί όταν εξασφαλιστεί ότι η BF έχει γίνει ‘0’. Τέλος, τα pins15,16 ασχολούνται με το οπίσθιο φωτισμό (backlight) της πηγής (Vcc) και της γείωσης(GND), αντίστοιχα. Στο Σχήμα 6.3 απεικονίζονται οι ακροδέκτες του ελεγκτή της οθόνης LCD.

Pin#	Name		In/Out/Pwr
1	GND	Ground	Power
2	VCC	LCD Controller Power (+3 to +5V)	Power
3	VLCD	LCD Display Bias (+5 to -5V *see text)	Analog
4	RS	Register Select: H: Data L: Command	Input
5	R/W	H: Read L: Write	Input
6	E	Enable (Data strobe, active high)	Input
7	DB0	Data LSB	I/O
8	DB1	Data	I/O
9	DB2	Data	I/O
10	DB3	Data	I/O
11	DB4	Data	I/O
12	DB5	Data	I/O
13	DB6	Data	I/O
14	DB7	Data MSB	I/O
15	A	LED Backlight Anode (optional)	Power
16	K	LED Backlight Cathode (optional)	Power

Σχήμα 6.3 Απεικόνιση ακροδεκτών του ελεγκτή της οθόνης LCD

6.3.2 Βήματα για εγγραφή στην LCD οθόνη

Για να επιτευχθεί η εγγραφή στην οθόνη lcd, πρέπει πρώτα να πραγματοποιηθεί η αρχικοποιησή της. Η αρχικοποίηση αποτελείται από πέντε διαφορετικές καταστάσεις, οι οποίες μας δίνουν διαφορετικό αποτέλεσμα στους διαύλους δεδομένων (DataBus). Στιστρεις πρώτες καταστάσεις ο DB παίρνει την τιμή "00110000", λειτουργώντας δηλαδή σαν να έχει μηδενική είσοδο. Η τέταρτη κατάσταση έχει διαφορετικό αποτέλεσμαστον DB, το οποίο είναι "00110001". Έπειτα συνεχίζουν οι καταστάσεις ClearDisplay, DisplayControl και EntryMode. Η ClearDisplay έχει ως αποτέλεσμα στον DB την τιμή "00000001", καθαρίζει τη διεύθυνση μνήμης και επιστρέφει την οθόνη στην αρχική της κατάσταση. Η DisplayControl έχει ως αποτέλεσμα στον DB "00001100" ενώ η EntryMode "00000110". Στη συνέχεια, γίνεται η εγγραφή δεδομένων στην lcd, η οποία αποτελείται από δύο καταστάσεις. Η πρώτη κατάσταση είναι η WriteData μέσω της οποίας γίνεται η εγγραφή της εισόδου στη μνήμη της lcd και απεικονίζεται στον DB. Στο Σχήμα 6.4, που ακολουθεί, παρουσιάζονται οι πιθανές περιπτώσεις εισόδου στην lcd και το αντίστοιχο αποτέλεσμα στους διαύλους δεδομένων (DB).

Είσοδος	**DataBus (DB)**
0 ("0000")	"00110000"
1 ("0001")	"00110001"
2 ("0010")	"00110010"
3 ("0011")	"00110011"
4 ("0100")	"00110100"
5 ("0101")	"00110101"
6 ("0110")	"00110110"
7 ("0111")	"00110111"
8 ("1000")	"00111000"
9 ("1001")	"00111001"
A ("1010")	"01000001"
b ("1011")	"01000010"
C ("1100")	"01000011"
d ("1101")	"01000100"

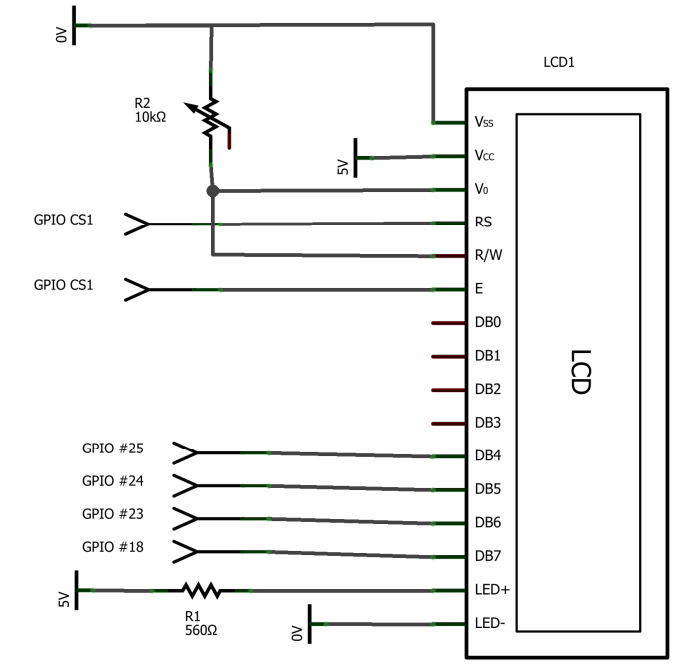
E (“1110”)	“01000101”
F (“1111”)	“01000110”

Σχήμα 6.4 Περιπτώσεις εισόδου στην οθόνη lcd και τα αποτελέσματα στουςδιαύλους δεδομένων (DB)

Έπειτα, ακολουθεί η ReturnHome, η οποία μηδενίζει τη μνήμη κ επιστρέφει την οθόνη στην αρχική της κατάσταση, βγάζοντας ως αποτέλεσμα στον DB “10000000”. Οι δύο αυτές καταστάσεις επαναλαμβάνονται συνεχώς.

6.3.3 Καταχωρητές του ελεγκτή (controller) της οθόνης LCD

Ο HD44780U έχει δύο καταχωρητές των 8-bit, ένα καταχωρητή εντολών (IR) και ένα καταχωρητή δεδομένων (DR).Ο καταχωρητή εντολών (IR) αποθηκεύει κώδικες εντολών (όπως η display clear και cursor shift) και πληροφορίες διεύθυνσης για τη RAM απεικόνισης δεδομένων (DDRAM) και τη RAM γεννήτριας χαρακτήρων (CGRAM). Ο καταχωρητής αυτός μπορεί να γραφεί μόνο από τον MPU.Από την άλλη, ο καταχωρητής δεδομένων (DR) αποθηκεύει προσωρινά δεδομένα για να γραφούν ή να διαβαστούν από την DDRAM ή την CGRAM . Τα δεδομένα γράφονται στον DR από το MPU και είναι αυτόματα γραμμένα στην DDRAM ή στηνCGRAM από μια εσωτερική λειτουργία. Ο DR χρησιμοποιείται επίσης για την αποθήκευση δεδομένων κατά την ανάγνωση των δεδομένων από την DDRAM ή την CGRAM. Όταν οι πληροφορίες διεύθυνσης γράφονται στον IR, τα δεδομένα διαβάζονται και στη συνέχεια αποθηκεύονται στον DR από την DDRAM ή την CGRAM από μια εσωτερική λειτουργία. Όταν ο MPU διαβάζει τον DR σημαίνει ότι η μεταφορά δεδομένων μεταξύ του MPU έχει ολοκληρωθεί. Μετά την ανάγνωση, τα δεδομένα της επόμενης διεύθυνσης της DDRAM ή της CGRAM αποστέλλονται στον DR για την επόμενη ανάγνωση από τον MPU. Από τον καταχωρητη επιλογής σήματος (RS), αυτοί οι δύο καταχωρητές μπορούν να επιλεχθούν. Στο Σχήμα 6.4 παρουσιάζεται το διάγραμμα της αρχιτεκτονικής του ελεγκτή.



Σχήμα 6.5 Διάγραμμα της αρχιτεκτονικής του ελεγκτή

Οι χωροι αποθηκευσης του Display Data RAM (DDRAM) απεικονιζουν τα δεδομενα σε κωδικούς χαρακτήρων 8-bit. Παρατείνει την χωρητικότητα σε 80 x 8 bits, ή 80 χαρακτήρες. Η περιοχή στη μνήμη RAM δεδομένων οθόνη (DDRAM) που δεν χρησιμοποιείται για απεικονιση μπορεί να χρησιμοποιηθεί ως γενική RAM δεδομένων.

- 1-line απεικονισης (N = 0)

Όταν υπάρχουν λιγότεροι από 80 απεικονίσεις χαρακτήρων, η απεικόνιση αρχίζει στην θέση της κεφαλής. Για παράδειγμα, αν χρησιμοποιείται μόνο ο HD44780, οι 8 χαρακτήρες εμφανίζονται. Όταν η λειτουργία μετατόπισης της οθόνης εκτελείται, μετατοπίζεται η διεύθυνση DDRAM.

- 2-line απεικονισης (N = 1)

Περίπτωση 1: Όταν ο αριθμός των χαρακτήρων της οθόνης είναι μικρότερος από 40 x 2 γραμμές, οι δύο γραμμές απεικονιζονται από την κεφαλή. Σημειώνεται ότι το τέλος της διεύθυνσης στην πρώτη γραμμή δεν είναι συνεχής με την αρχή της διεύθυνσης στην δεύτερη γραμμή. Για παράδειγμα, όταν χρησιμοποιείται ο HD44780, οι 8 χαρακτήρες x 2 γραμμές εμφανίζονται. Όταν η λειτουργία μετατόπισης της οθόνης εκτελείται, μετατοπίζεται η διεύθυνση DDRAM.

Περίπτωση 2: Για μια οθόνη με 2 γραμμές και 16 χαρακτήρες, ο HD44780 μπορεί να επεκταθεί με τη χρήση ενός 40-εξόδου οδηγού επέκτασης. Όταν η λειτουργία μετατόπισης της οθόνης εκτελείται, μετατοπίζεται η διεύθυνση DDRAM.

Στη γεννήτρια χαρακτήρων RAM (Character Generator RAM, CGRAM) ο χρήστης μπορεί να ξαναγράψει τα μοτιβα χαρακτήρων με πρόγραμμα. Για 5 x 8 κουκκίδες, οκτώ μοτίβα χαρακτήρων μπορούν να γραφούν, και για 5 x 10 κουκκίδες, μπορούν να γραφούν τέσσερα μοτίβα χαρακτήρων. Οι περιοχές που δεν χρησιμοποιούνται για απεικονιση μπορεί να χρησιμοποιηθούν ως γενική RAM δεδομένων.

6.3.4 Βασικές εντολές του ελεγκτή της LCD

Clear Display

Η Clear display γράφει κωδικα κενων 20H σε όλες τις διευθύνσεις DDRAM. Στη συνέχεια, μηδενίζει τη διεύθυνση της DDRAM στον address counter, και επιστρέφει την οθόνη στην αρχική της κατάσταση αν είχε μετατοπίστει. Με άλλα λόγια, η προβολη εξαφανίζεται και ο δρομέας πηγαίνει το αριστερό άκρο της οθόνης (στην πρώτη γραμμή αν 2 γραμμές εμφανίζονται). Θέτει, επίσης, το I / D σε 1 (λειτουργία προσαύξηση) στη λειτουργία εισόδου και αφήνει το S έτσι όπως είναι, δεν το αλλάζει.

Return Home

Επιστρέφει την οθόνη στην αρχική της κατάσταση και μηδενίζει τη διεύθυνση DDRAM στον address counter. Το περιεχόμενο της DRAM δεν αλλάζει. Ο δρομέας πάει στο αριστερό άκρο της οθόνης (στην πρώτη γραμμή αν 2 γραμμές εμφανίζονται).

Entry Mode Set

Ορίζει την κατεύθυνση του κέρσορα και την απεικόνιση της μετατόπισης κατά τη διάρκεια της ανάγνωσης ή της εγγραφής.

I/D: Αυξήση (I / D = 1) ή μειώση (I / D = 0) της διεύθυνσης DDRAM κατα 1 όταν ένας κωδικός χαρακτήρας εχει γραφει εντός ή διαβάζεται από την DDRAM.
Ο δρομέας μετακινείται προς τα δεξιά όταν αυξάνεται κατά 1 και προς τα αριστερά, όταν μειώνεται κατά 1.Το ίδιο ισχύει και για την εγγραφή και την ανάγνωση της CGRAM
S: Μετακινεί ολόκληρη την οθόνη είτε προς τα δεξιά (I / D = 0) ή προς τα αριστερά (I/ D = 1) όταν το s είναι 1. Η οθόνη δεν μετατοπιζεται αν S είναι 0. Αν S είναι 1, θα φαίνεται σαν να μην κινειται ο κέρσορας, αλλά η οθόνη το κάνει. Η οθόνη δεν αλλάζει όταν διαβαζουμε από την DDRAM. Επίσης, η εγγραφή και η ανάγνωση της CGRAM δεν θα αλλάξει την οθόνη.

Display On/Off Control

D: Η οθόνη είναι ενεργοποιημένη όταν D είναι σε κατάσταση 1 και σβήνει όταν D είναι 0. Όταν σβηνει τα δεδομένα της οθόνης παραμένουν στην DDRAM, αλλά μπορούν να εμφανιστούν άμεσα θέτοντας το D σε 1.
C: Ο δρομέας εμφανίζεται όταν C είναι 1 και δεν εμφανίζεται όταν το C είναι 0. Ακόμη και αν ο δρομέας εξαφανίστει, η λειτουργία του I / D ή άλλες προδιαγραφές δεν θα αλλάξουν κατά τη διάρκεια της εγγραφής δεδομένων της οθόνης.
B: Ο χαρακτήρας υποδεικνύεται από τον δρομέα που αναβοσβήνει όταν το B είναι 1. Η ένδειξη που αναβοσβήνει εμφανίζεται ως εναλλαγή μεταξύ όλων των κενων κουκκιδων και των χαρακτήρων που εμφανίζονται με ταχύτητα διαστήματων 409.6ms όταν Fcp ή Fosc είναι 250 kHz. Ο δρομέας και το blinking μπορούν να ρυθμιστούν για να εμφανιζονται ταυτόχρονα. (Η συχνοτήτα του blinking αλλαζει σύμφωνα με την Fosc ή το αντίστροφο του Fcp.

Cursor or Display Shift

Δρομέας ή μετατόπιση οθόνης μετατοπίζει τη θέση του δρομέα ή της οθόνης προς τα δεξιά ή προς τα αριστερά χωρίς εγγραφή ή ανάγνωση εμφάνισης δεδομένων. Αυτή η λειτουργία χρησιμοποιείται για να διορθώσει ή να ψαξει την οθόνη. Σε μια οθόνη 2 γραμμών, ο δρομέας κινείται προς τη δεύτερη γραμμή όταν περνά το 40ο ψηφίο της πρώτης γραμμής. Σημειώστε ότι η πρώτη και η δεύτερη γραμμή της οθόνης μετατοπιζονται ταυτόχρονα.

Όταν τα εμφανιζόμενα δεδομένα μετατοπίζονται κατ 'επανάληψη κάθε γραμμή κινείται μόνο οριζόντια. Η δεύτερη γραμμή εμφανισης δεν μετατοπίζεται στην πρώτη θέση της γραμμής. Το περιεχόμενο του address counter (AC) δεν θα αλλάξει αν η μόνη ενέργεια που εκτελείται είναι μια μετατόπιση της οθόνης.

S/C=1 μετακίνηση απεικόνισης S/C=0 μετακίνηση δρομέα

R/L=1 μετακίνηση στα δεξιά R/L=0 μετακίνηση στα αριστερά

Function Set

DL: Ορίζει το μέγεθος διαύλου. Τα δεδομένα αποστέλλονται ή λαμβάνονται σε μήκη 8-bit (DB7-DB0) όταν DL είναι 1, και σε μήκη 4-bit(DB7-DB4) όταν DL είναι 0. Όταν ειναι επιλεγμένο το μήκος 4-bit, τα δεδομένα πρέπει να αποστέλλονται ή να λαμβανονται δύο φορές**.**

N: Ορίζει τον αριθμό των γραμμών της οθόνης. Όταν N είναι 0 τότε υπάρχει 1-γραμμή λειτουργίας ενώ όταν N είναι 1 τότε υπάρχουν 2-γραμμές λειτουργίας.

F: Ορίζει τη γραμματοσειρά χαρακτήρων. Όταν F είναι 0 τότε 5x8 dots ενώ ότα F είναι 1 τότε 5x10 dots.

Set CGRAM Address

Ορίζει την διεύθυνση CGRAM σε δυαδικό AAAAAA στον address counter.

Τα δεδομένα στη συνέχεια εγγραφονται ή διαβάζονται από τον MPU για την CGRAM.

Set DDRAM Address

Ορίζει την διεύθυνση DDRAM σε δυαδικό AAAAAAA στον address counter.

Τα δεδομένα στη συνέχεια εγγραφονται ή διαβάζονται από τον MPU για την DDRAM. Ωστόσο, όταν το N είναι 0 (οθόνη 1 γραμμής), AAAAAA μπορεί να είναι 00H-4PH. Όταν το N είναι 1 (οθόνη 2 γραμμών), AAAAAAA μπορεί να είναι 00H-27H για την πρώτη γραμμή, και 40H-67H για τη δεύτερη γραμμή.

Read Busy Flag and Address

Διαβάζει την busy flag (BF) και τον μετρητή διεύθυνσης (AC) που δείχνει ότι το σύστημα είναι σε εσωτερική λειτουργία. Αν BF είναι 1, η εσωτερική λειτουργία είναι σε εξέλιξη. Η επόμενη εντολή δεν θα γίνει δεκτη μέχρι η BF να επανέλθει στο 0. Γινεται ελεγχος του BF πριν από την επόμενη λειτουργία εγγραφής. Την ιδια στιγμη, η τιμή του μετρητή διεύθυνσης(address counter) σε δυαδικό AAAAAA διαβάζεται. Αυτός ο μετρητής διεύθυνση(address counter) χρησιμοποιείται τόσο απο την CG και την DDRAM διευθυνση, και η αξία τους καθορίζεται από την προηγούμενη εντολή. Τα περιεχόμενα της διεύθυνσης είναι ίδια όπως και για τις οδηγίες διεύθυνσης CGRAM και της διεύθυνση DDRAM.

Write Data to CG or DDRAM

Γράφει 8-bit δυαδικων δεδομένων DDDDDDDD στην CGRAM ή στην DDRAM.
Για την εγγραφη στην CGRAM ή στην DDRAM αυτό καθορίζεται από την προηγούμενη προδιαγραφή της ρύθμιση διεύθυνσης της CGRAM ή DDRAM. Μετά από μια εγγραφή, η διεύθυνση αυτόματα αυξάνεται ή μειώνεται κατά 1 σύμφωνα με η λειτουργία εισαγωγής. Η λειτουργία εισαγωγής καθορίζει επίσης τη μετατόπιση της οθόνης.

Read Data from CG or DDRAM

Διαβάζει 8-bit δυαδικων δεδομένων DDDDDDDD από την CGRAM ή την DDRAM.
Η προηγούμενη περιγραφη καθορίζει εάν η CGRAM ή η DDRAM είναι να διαβαστεί. Πριν από την είσοδο αυτής της εντολης ανάγνωσης, είτε η CGRAM είτε η DDRAM διεύθυνση οριζει την εντολη που πρέπει να εκτελεστεί. Εάν δεν εκτελεστει, τα πρώτα δεδομένα αναγνωσης θα είναι άκυρα. Όταν η εντολη αναγνωσης εκτελεστει σειριακα, η επόμενη διεύθυνση δεδομενων διαβάζεται κανονικά από τη δεύτερη ανάγνωση. Αυτη η διευθυνση οριζει ποιες εντολες δεν χρειάζεται να εκτελεστούν πριν από αυτή την εντολή ανάγνωσης όταν μετατόπιζεται ο κερσορας με την εντολή μετατόπισης κερσορα (όταν διαβάζει την DDRAM). Η λειτουργία της εντολης μετατόπισης του κερσορα είναι η ίδια με την εντολή της ρυθμισης διευθυνσης της

DDRAM. Μετά από μια ανάγνωση, η λειτουργία εισαγωγής αυτόματα αυξάνει ή μειώνει την διεύθυνση κατα 1. Ωστόσο,η μετατόπιση οθόνης δεν εκτελείται ανεξάρτητα από τον τρόπο εισαγωγής.

Κεφάλαιο 7 :
Εφαρμογή οδήγησης οθονών και προσομοίωση

Σε αυτό το κεφάλαιο απεικονίζεται ο κώδικας οδήγησης οθονών με την ανάλυση του, όπως και η προσομοίωσή του με το κατάλληλο εργαλείο CAD (Computer-Aided Design) της εταιρείας ALTERA, το Quartus II, το οποίο χρησιμοποιείται για τη σχεδίαση ψηφιακών κυκλωμάτων.

7.1 Υλοποίηση κώδικα για απλές συστοιχίες LEDs, απεικόνισεις επτά τομέων (SSD) και οθόνης LCD

```
1.  LIBRARY ieee;
2.  USE ieee.std_logic_1164.all;
3.  ENTITY led_ssd_lcd_driver IS
4.      GENERIC (clk_divider: POSITIVE :=50000);
5.       PORT(clk: IN std_logic;
6.            switches: IN std_logic_vector(3 DOWNTO 0);
7.            leds: OUT std_logic_vector(3 DOWNTO 0);
8.            ssd: OUT std_logic_vector(6 DOWNTO 0);
9.            RS, RW, LCD_ON, BKL_ON: OUT std_logic;
10.           E: BUFFER std_logic;
11.           DB: OUT std_logic_vector(7 DOWNTO 0));
12. END led_ssd_lcd_driver;
13. ---------------------------------------------------------------------------------
14. ARCHITECTURE Behaviour OF led_ssd_lcd_driver IS
15.     TYPE STATE IS( FunctionSet1, FunctionSet2, FunctionSet3,
16.                    FunctionSet4, ClearDisplay, DisplayControl,
17.                    EntryMode, WriteData,  ReturnHome);
18.      SIGNAL pr_state, nx_state: state;
19.      SIGNAL input_lcd: std_logic_vector(7 DOWNTO 0);
20. BEGIN
21.       ---------Part1: LED driver----------------------------------------
```

```
22.         leds<= switches;
23.         ---------Part2: SSD driver----------------------------------------
24.         WITH switches SELECT
25.            ssd<= “0000001” WHEN “0000”,
26.                     “1001111” WHEN “0001”,
27.                     “0010010” WHEN “0010”,
28.                     “0000110” WHEN “0011”,
29.                     “1001100” WHEN “0100”,
30.                     “0100100” WHEN “0101”,
31.                     “0100000” WHEN “0110”,
32.                     “0001111” WHEN “0111”,
33.                     “0000000” WHEN “1000”,
34.                     “0000100” WHEN “1001”,
35.                     “0001000” WHEN “1010”,
36.                     “1100000” WHEN “1011”,
37.                     “0110001” WHEN “1100”,
38.                     “1000010” WHEN “1101”,
39.                     “0110000” WHEN “1110”,
40.                     “0111000” WHEN OTHERS;
41.         ---------Part3: LCD driver (FSM-based)-----------------------------
42.         WITH switches SELECT
43.             input_lcd<= “00110000” WHEN “0000”,
44.                              “00110001” WHEN “0001”,
45.                              “00110010” WHEN “0010”,
46.                               “00110011” WHEN “0011”,
47.                              “00110100” WHEN “0100”,
48.                               “00110101” WHEN “0101”,
49.                               “00110110” WHEN “0110”,
50.                               “00110111” WHEN “0111”,
51.                               “00111000” WHEN “1000”,
52.                               “00111001” WHEN “1001”,
53.                               “01000001” WHEN “1010”,
54.                               “01000010” WHEN “1011”,
55.                               “01000011” WHEN “1100”,
```

```
                    “01000100” WHEN “1101”,
                    “01000101” WHEN “1110”,
                    “01000110” WHEN OTHERS;
    -----Ενεργοποίηση οθόνης LCD και του οπίσθιου φωτισμού της-----
    LCD_ON<=’1’; BKL_ON<=’1’;
    -----Ρύθμιση γεννήτριας ρολογιού για την LCD (E=500Hz)-----------
    PROCESS(clk)
      VARIABLE count: INTEGER RANGE 0 TO clk_divider;
    BEGIN
      IF (clk’EVENT AND clk=’1’) THEN
          count:=count+1;
          IF (count = clk_divider ) THEN
              E<=NOT E;
              count:=0;
          END IF;
      END IF;
    END PROCESS;
    -----Κατώτερο τμήμα μηχανής πεπερασμένων καταστάσεων(FSM)----
    PROCESS(E)
    BEGIN
      IF (E’EVENT AND E=’1’) THEN
          pr_state<= nx_state;
      END IF;
    END PROCESS;
    -----Ανώτερο τμήμα μηχανής πεπερασμένων καταστάσεων(FSM)------
    PROCESS( pr_state, input_lcd)
    BEGIN
      CASE pr_state IS
      ---------Αρχικοποίηση της LCD---------
          WHEN FunctionSet1=>
              RS<=’0’; RW<=’0’;
              DB<= “0011XX00”;
               nx_state<= FunctionSet2;
          WHEN FunctionSet2=>
```

```
90.                     RS<='0'; RW<='0';
91.                     DB<= "0011XX00";
92.                     nx_state<= FunctionSet3;
93.                 WHEN FunctionSet3=>
94.                     RS<='0'; RW<='0';
95.                     DB<= "0011XX00";
96.                     nx_state<= FunctionSet4;
97.                 WHEN FunctionSet4=>
98.                     RS<='0'; RW<='0';
99.                     DB<= "00111000";
100.                    nx_state<= ClearDisplay;
101.                WHEN ClearDisplay =>
102.                    RS<='0'; RW<='0';
103.                    DB<= "00000001";
104.                    nx_state<= DisplayControl;
105.                WHEN DisplayControl =>
106.                    RS<='0'; RW<='0';
107.                    DB<= "00001100";
108.                    nx_state<= EntryMode;
109.                WHEN EntryMode =>
110.                    RS<='0'; RW<='0';
111.                    DB<= "00000110";
112.                    nx_state<= WriteData;
113.                -------Εγγραφή δεδομένων στην LCD------
114.                WHEN WriteData =>
115.                    RS<='1'; RW<='0';
116.                    DB<= input_lcd;
117.                    nx_state<= ReturnHome;
118.                WHEN ReturnHome =>
119.                    RS<='0'; RW<='0';
120.                    DB<= "10000000";
121.                    nx_state<= WriteData;
122.            END CASE;
123.        END PROCESS;
```

124. END Behaviour;

7.1.1 Επεξήγηση κώδικα

Στον παραπάνω κώδικα παρουσιάζεται η απεικόνιση του αποτελέσματος εισόδου 4bit σε απλές συστοιχίες LED, σε απεικόνιση επτά τομέων (SSD) και σε οθόνη LCD. Η οντότητα έχει το όνομα led_ssd_lcd_driver(γραμμή 3) με εισόδους clk και switches (γραμμές 5-6) και εξόδους leds (γραμμή 7 για τον LED driver), ssd (γραμμή 8 για τον SSD driver) και τέλος RS, RW, LCD_ON, BLK_ON, E, DB (γραμμές 9-11 για τον LCD driver). Στη συνέχεια περιγράφεται η αρχιτεκτονική με το όνομα Behaviour (γραμμή 14), στην αρχή της οποίας δηλώνεται μια μεταβλητή με δεδομένα αριθμητικού τύπου, που ονομάζεται state (γραμμή 15-17). Η οποία περιέχει όλα τα στάδια του ανώτερου τμήματος της μηχανής καταστάσεων (FSM) που χρησιμοποιείται για την αρχικοποίηση της LCD οθόνης. Επίσης, τα σήματα pr_state και nx_state (γραμμή 18), τα οποία έχουν δηλωθεί σύμφωνα με τον ίδιο τύπο δεδομένων και το σήμα input_lcd (γραμμή 19). Η αρχιτεκτονική χωρίζεται σε τρία κομμάτια. Στο πρώτο απεικονίζεται ο LED driver (γραμμή 22), ο οποίος χρειάζεται μόνο μια εκχώρηση, στο δεύτερο ο SSD driver (γραμμή 24-40) στον οποίο εισάγεται ένα σήμα των 4bit από τους διακόπτες και μετατρέπεται σε ένα σήμα των 7bit για την έξοδο από το SSD. Στο τρίτο κομμάτι περιγράφεται ο LCD driver (γραμμές 42-123) ο οποίος είναι πιο περίπλοκος από τους προηγούμενους διότι πρέπει να επικοινωνήσει με τον LCD μικροελεγκτή του. Στην αρχή εισάγεται σήμα των 4bit από τους διακόπτες και εξέρχεται σήμα 8bit δεδομένων (γραμμές 42-58), τα οποία παρέχει ο μικροελεγκτής. Όπως και 3bit ελέγχου (RS, RW, E) και επιπλέον δύο πρόσθετα σήματα (LCD_ON, BKL_ON) για την ενεργοποίηση της οθόνης LCD και του οπίσθιου φωτισμού της (γραμμή 60). Στη συνέχεια ρυθμίζει τη γεννήτρια ρολογιού για την LCD οθόνη έτσι ώστε να είναι στα 500Hz(γραμμές 62-72) και ακολουθούν τα δύο τμήματα της μηχανής πεπερασμένων καταστάσεων (FSM). Το κατώτερο (ακολουθητικό) τμήμα του FSM, όπου το E (Enable) παίζει το ρόλο του ρολογιού μέσα στη διεργασία(γραμμές 74-79) και το ανώτερο(συνδιαστικό) τμήμα του σε άλλη διεργασία, όπου γίνεται η αρχικοποίηση της LCD (γραμμές 81-123). Τα αρχικά τέσσερα στάδια (FunctionSet1-4) προετοιμάζουν τον μικροελεγκτή για να λειτουργήσει με ένα δίαυλο των 8bit και σε κατάσταση 2 γραμμών(γραμμές 85-100).

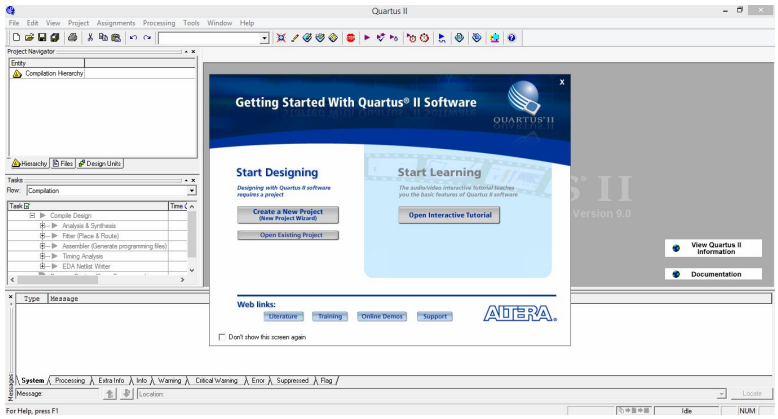
Το πέμπτο στάδιο (ClearDisplay) προκαλεί τον καθαρισμό της απεικόνισης και μηδενίζει τη διεύθυνση της μνήμης(γραμμές 101-104). Στο έκτο στάδιο (DisplayControl) η απεικόνιση είναι ενεργοποιημένη ενώ ο κέρσορας και το blink είναι απενεργοποιημένα (γραμμές 105-108). Στο έβδομο στάδιο (EntryMode) η διεύθυνση της μνήμης μπαίνει σε κατάσταση αύξησης(γραμμές 109-112). Στο όγδοο στάδιο (WriteData) εγγράφει τα δεδομένα στη μνήμη, η οποία έχει καθοριστεί από προηγούμενη εντολή (γραμμές 114-117). Και τέλος, το ένατο στάδιο (ReturnHome) επιστρέφει την οθόνη στην αρχική της κατάσταση και μηδενίζει τη διεύθυνση μνήμης (γραμμές 118-121).

Στα πρώτα επτά στάδια ο RS είναι μηδέν για να είναι επιλεγμένος ο καταχωρητής εντολών και ο RW είναι μηδέν διότι οι πληροφορίες πρέπει πάντα να γράφονται στον μικροελεγκτή και όχι να διαβάζονται από αυτόν.

7.2 Προσομοίωση με το Quartus II της εταιρείας ALTERA

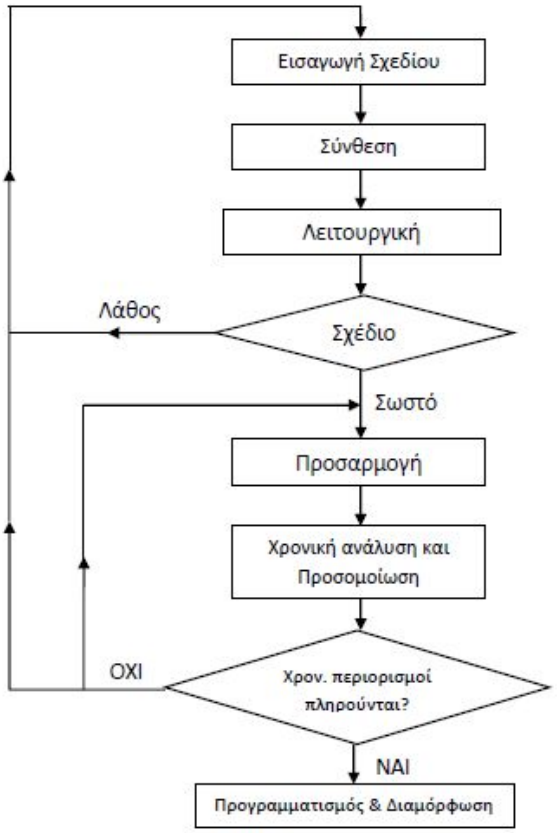
Το λογισμικό Quartus II είναι ένα από τα γνωστότερα προγράμματα σχεδίασης CAD (Computer-Aided Design) και είναι πνευματική ιδιοκτησία της ALTERA. Το Quartus II χρησιμοποιείται για την ανάπτυξη και τον προγραμματισμό όλων των αναπτυξιακών κυκλωμάτων της εταιρείας ALTERA, δηλαδή των διατάξεων CPLDs και FPGAs που κατασκευάζει η εταιρεία.

Στην ιστοσελίδα της ALTERA υπάρχουν πολλές διαθέσιμες εκδόσεις του Quartus II, για την προσομοίωση του κώδικά μου επέλεξα την έκδοση Quartus II 9.0. Στο Σχήμα 7.1 που ακολουθεί, απεικονίζεται το παράθυρο του λογισμικού Quartus II αμέσως μετά το άνοιγμά του.



Σχήμα 7.1 το παράθυρο του λογισμικού Quartus II 9.0

Επίσης, υπάρχουν και πολλές νεότερες εκδόσεις του, με πιο πρόσφατη την έκδοση Quartus II 16.0. Ωστόσο για την αναπτυξιακή πλακέτα DE2 η νεότερη έκδοση που μπορούμε να χρησιμοποιήσουμε είναι η 13, διότι οι νεότερες της δεν υποστηρίζουν το Cyclone II FPGA που περιέχεται σε αυτήν. Στο Σχήμα 7.2 που ακολουθεί, φαίνεται η ροή διεργασιών στο Quartus II.



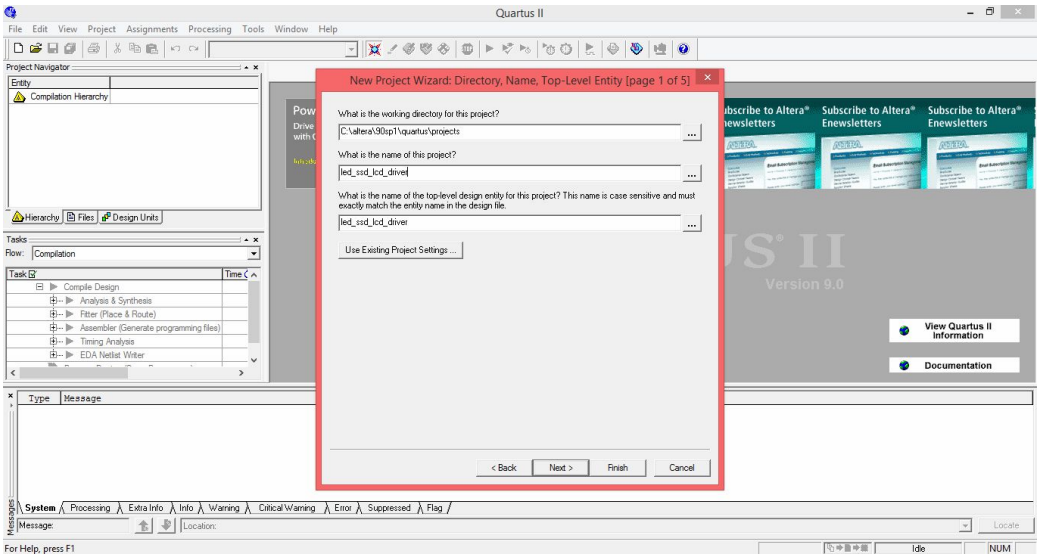
Σχήμα 7.2 Ροή διεργασιών στο Quartus II

7.2.1 Ορισμός του σχεδίου (Project)

Ανοίγοντας το σχεδιαστικό περιβάλλον του Quartus II μας ζητείται η ονομασία του project. Η έννοια του project περιλαμβάνει το σύνολο των αρχείων που δημιουργούμε, καθώς και τα αρχεία που δημιουργεί το πρόγραμμα για να εκτελέσει τις διάφορες λειτουργίες του. Δηλαδή, σαν project θεωρούμε το σύνολο των αρχείων που δημιουργούνται για μια εφαρμογή. Η δημιουργία γίνεται από το **Menu File/New Project Wizard.**

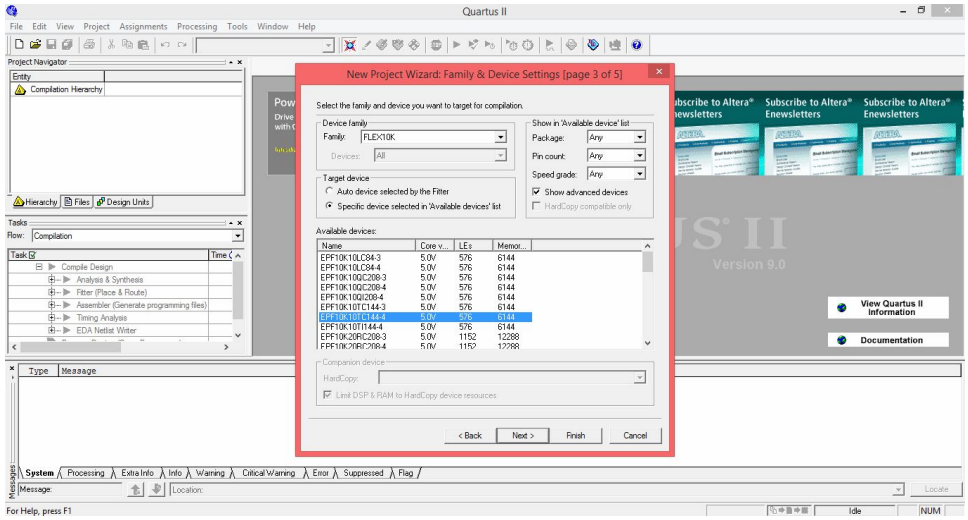
Στη συνέχεια, ανοίγει ένα παράθυρο όπου δηλώνουμε το όνομα του Project, καθώς και το όνομα του φακέλου που θα το περιέχει. Στη δική μας περίπτωση, το

όνομα του project είναι **led_ssd_lcd_driver** και αποθηκεύεται στο φάκελο projects, ο οποίος βρίσκεται στο ακόλουθο path, c:\altera\90sp1\quartus. Όπως απεικονίζεται και στο Σχήμα 7.3.



Σχήμα 7.3 Ονομασία και φάκελος αποθήκευσης του Project

Τέλος, πατώντας Next στην ίδια καρτέλα, το πρόγραμμα μας ζητά να επιλέξουμε από μία λίστα την συγκεκριμένη διάταξη FPGA που θα χρησιμοποιήσουμε. Το FPGA που χρησιμοποιούμε στην πρώτη περίπτωση, ανήκει στην οικογένεια **FLEX10K** και είναι το **EPF10K10TC144-4**. Ενώ στη δεύτερη περίπτωση χρησιμοποίησα το **EP2C35F672C6** της οικογένειας **Cyclone II**, όπως φαίνεται και στο Σχήμα 7.4.

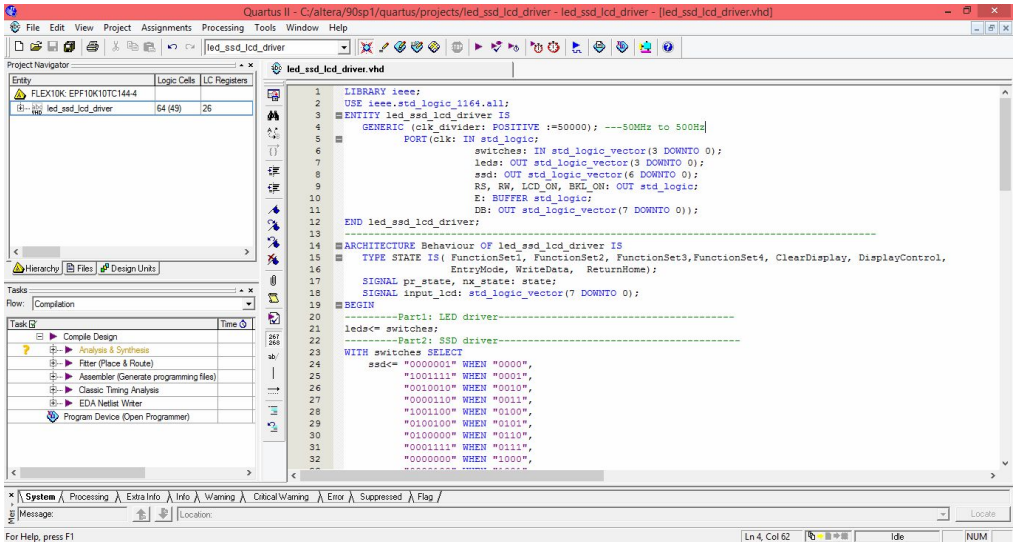


Σχήμα 7.4 Επιλογή διάταξης FPGA

7.2.2 Εισαγωγή αρχείου

Εφόσον έχουμε δώσει όνομα στο Project μπορούμε να συνεχίσουμε κάνοντας την εισαγωγή του κυκλώματος με όποιον τρόπο επιθυμούμε. Η επιλογή αυτή γίνεται από την καρτέλα **New** στο **Design Files**. Μπορούμε να δημιουργήσουμε ένα κύκλωμα είτε περιγράφοντάς το σχηματικά, είτε με κάποια γλώσσα περιγραφής. Στην περίπτωσή μας, η περιγραφή θα γίνει με τη γλώσσα περιγραφής υλικού VHDL, διαλέγοντας την επιλογή **VHDL File.**

Στη συνέχεια, γράφουμε τον κώδικα στο νέο παράθυρο που ανοίγει, όπως φαίνεται στο Σχήμα 7.5 και αφού τελειώσουμε, σώζουμε το αρχείο με το ίδιο όνομα που είχαμε δώσει στο Project, δηλαδή **led_ssd_lcd_driver.** Ο κώδικας παρουσιάζεται αναλυτικά στην παραπάνω υποενότητα 7.1, ο οποίος χρησιμοποιείται για την υλοποίηση στο ανάπτυγμα DE2. Για την υλοποίηση στο ανάπτυγμα LP-2900 χρησιμοποιείται ο ίδιος κώδικας με μία μόνο αλλαγή στο clk_divider, το οποίο θέτουμε στα 10MHz διότι αυτή είναι η συχνότητα του κρυστάλλου του.

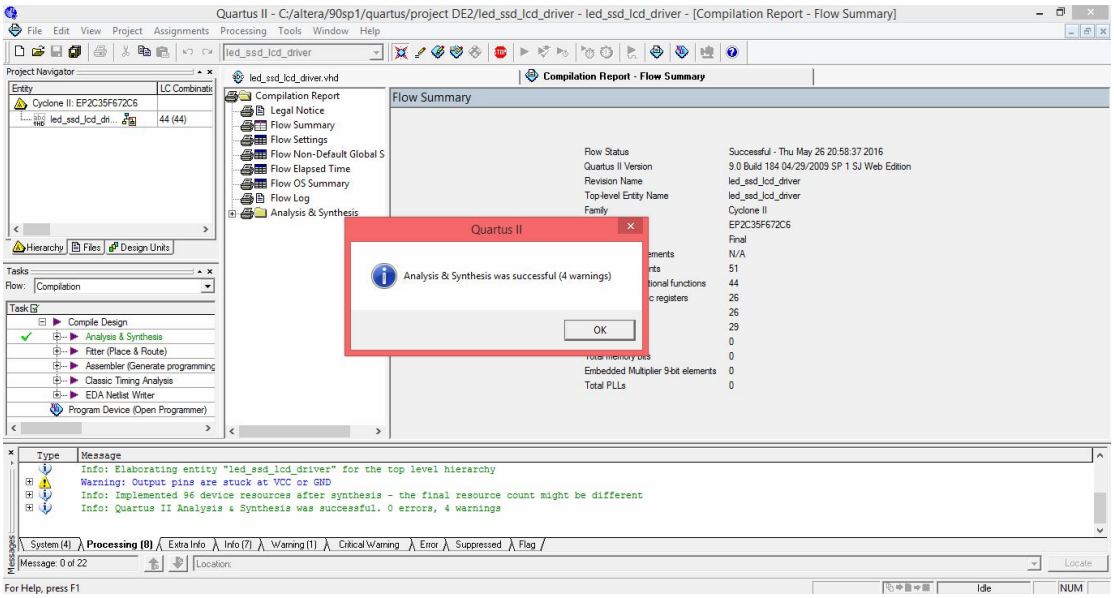


Σχήμα 7.5 Εισαγωγή του κυκλώματος με γλώσσα περιγραφής υλικού VHDL

7.2.3 Ανάλυση και Σύνθεση

Μετά την ολοκλήρωση της περιγραφής του κυκλώματος και της αποθήκευσής του, ακολουθεί η διαδικασία της Ανάλυσης και Σύνθεσης. Η διαδικασία αυτή γίνεται από το **Menu/ Processing/ Start/ Start Analysis & Synthesis**. Μετά το τέλος αυτής της

διαδικασίας εμφανίζεται μία αναφορά, η οποία μας ενημερώνει και πρέπει να εμφανίσει το μήνυμα του Σχήματος 7.6, έτσι ώστε να μπορέσουμε να συνεχίσουμε.

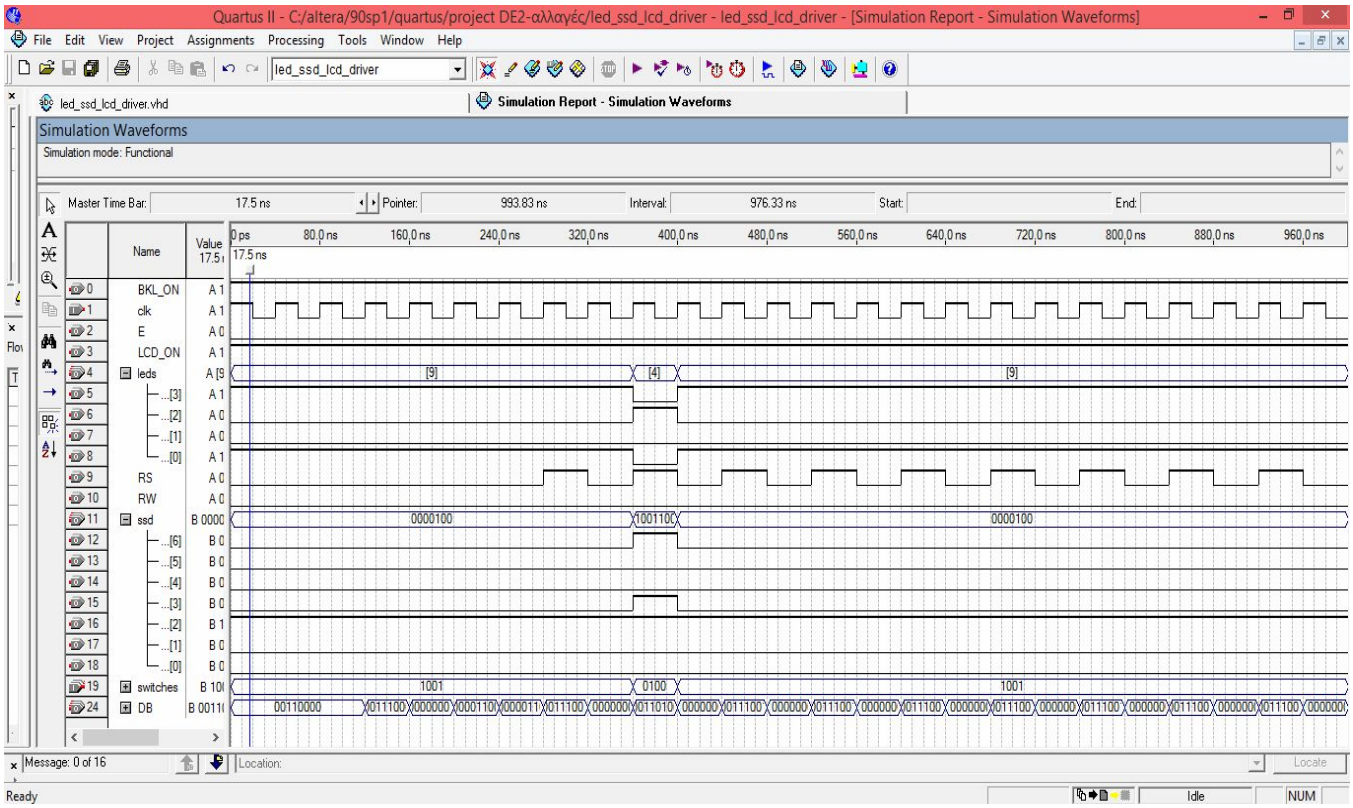


Σχήμα 7.6 Η διαδικασία της Ανάλυσης & Σύνθεσης

7.2.4 Η διαδικασία της προσομοίωσης

Η διαδικασία αυτή γίνεται μέσα από τον Waveform Editor, με τον οποίο εισάγουμε τις κατάλληλες κυματομορφές εισόδου (waveform vectors) που είναι απαραίτητες για την προσομοίωση. Για να εμφανίσουμε τον Waveform Editor επιλέγουμε από το **Menu/ File/ New/ Verification/Debugging Files/ Vector Waveform File**.

Στην επιφάνεια του Quartus εμφανίζεται ένα παράθυρο, όπου επιλέγουμε τις εισόδους και τις εξόδους του κυκλώματος που επιθυμούμε να προσομοιώσουμε. Πριν τους σώσουμε τιμές επιλέγουμε από το **Menu/ Processing/ Generate Functional Simulation Netlist** ώστε να δημιουργηθεί ένα αρχείο λειτουργικής περιγραφής του κυκλώματος. Έπειτα, επιλέγουμε από το **Menu/ Assignments/ Settings/ Simulator Setting** έτσι ώστε να επιλέξουμε τι είδους προσομοίωση θέλουμε να κάνουμε. Σε αθτή την περίπτωση επιλέγουμε **Functional** και θέτουμε τη συνολική διάρκεια της προσομοίωσης ίση με 1us. Στη συνέχεια δίνουμε τιμές στις εισόδους και τέλος, εκτελούμε την προσομοίωση πηγαίνοντας στο **Menu/ Processing/ Start Simulation**. Στο τέλος της προσομοίωσης, εμφανίζονται οι τιμές των εξόδων για τις τιμές των εισόδων που σχεδιασαμε, οι οποίες απεικονίζονται στο Σχήμα 7.7.



Σχήμα 7.7 Παράθυρο προσομοίωσης με τιμές εισόδου και εξόδου

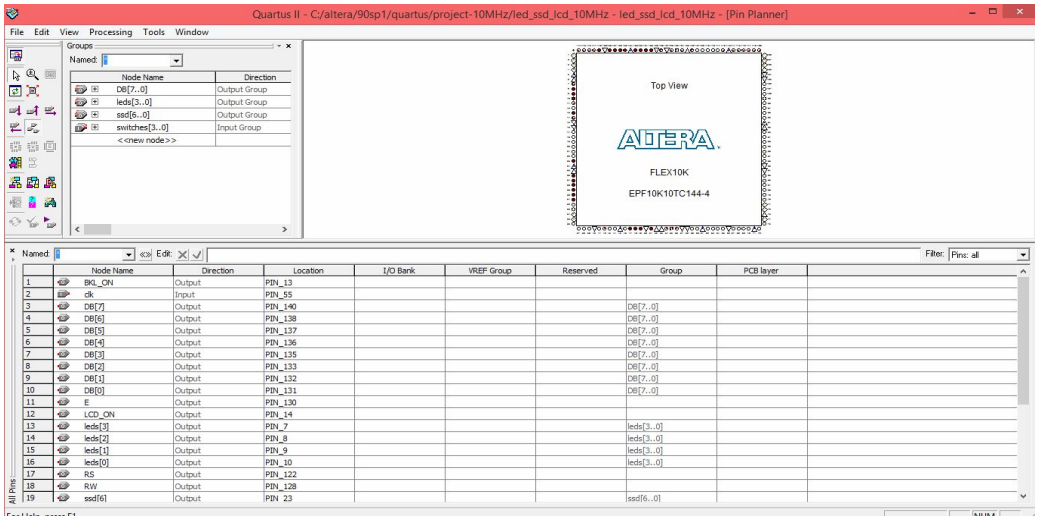
Στην προσομοίωση που πραγματοποιήσαμε, χρησιμοποιήσαμε στα switches μια συγκεκριμένη είσοδο ώστε να μπορέσουμε να καταγράψουμε τις αλλαγές που θα προκληθούν στους διαύλους δεδομένων της οθόνης lcd σε κάθε περίπτωση, στα leds και στην οθόνη επτά τομέων. Έχουμε βάλει ως είσοδο στα switches, αρχικά, τον αριθμό εννέα (“1001”), έτσι ξεκινά και η αρχικοποίηση της οθόνης lcd με βάση τις τιμές του σήματος DB. Αυτό περνά από πέντε διαφορετικές καταστάσεις, όπως έχουμε εξηγήσει πιο αναλυτικά στο Κεφάλαιο 6.3.2: τις καταστάσεις 1-3 που έχουν ως αποτέλεσμα “00110000” στους διαύλους δεδομένων, την κατάσταση 4 η οποία έχει διαφορετικό αποτέλεσμαστους διαύλους δεδομένων απ’ότι οι υπόλοιπες καταστάσεις, “00110001”. Έπειτα συνεχίζουν οι καταστάσεις ClearDisplay (“0000001”), DisplayControl (“00001100”) και EntryMode (“00000110”) οι οποίες επηρεάζουν η καθεμιά με το δικό τους τρόπο τους διαύλους δεδομένων. Στη συνέχεια γίνεται η εγγραφή δεδομένων στην lcd, η οποία αποτελείται από δύο καταστάσεις, την WriteData και την ReturnHome που επαναλαμβάνονται συνεχώς. Μέσω της πρώτης γίνεται η εγγραφή της εισόδου στη μνήμη της lcd (RS=1) και απεικονίζεται στους διαύλους δεδομένων. Σε αυτή την περίπτωση, που έχουμε ως είσοδο τον αριθμό εννέα (“1001”) στον DB έχουμε την τιμή “00111001”, η οποία απεικονίζει

τον αριθμό αυτό μετατρέποντάς τον σε σήμα των 8 bit. Έπειτα, ακολουθεί η κατάσταση ReturnHome, η οποία μηδενίζει τη μνήμη και επιστρέφει την οθόνη στην αρχική της κατάσταση, εμφανίζοντας στον DB την τιμή “10000000”. Όλη αυτή η διαδικασία διαρκεί εννέα παλμούς ρολογιού, κατά την οποία στις εξόδους των leds και της οθόνης επτά τομέων απεικονίζεται ο αριθμός εννέα. Στην έξοδο των leds ακριβώς όπως εισάχθηκε (“1001”) ενώ στην έξοδο της οθόνης επτά τομέων έχει μετατραπεί σε σήμα των 7 bit (“0000100”).

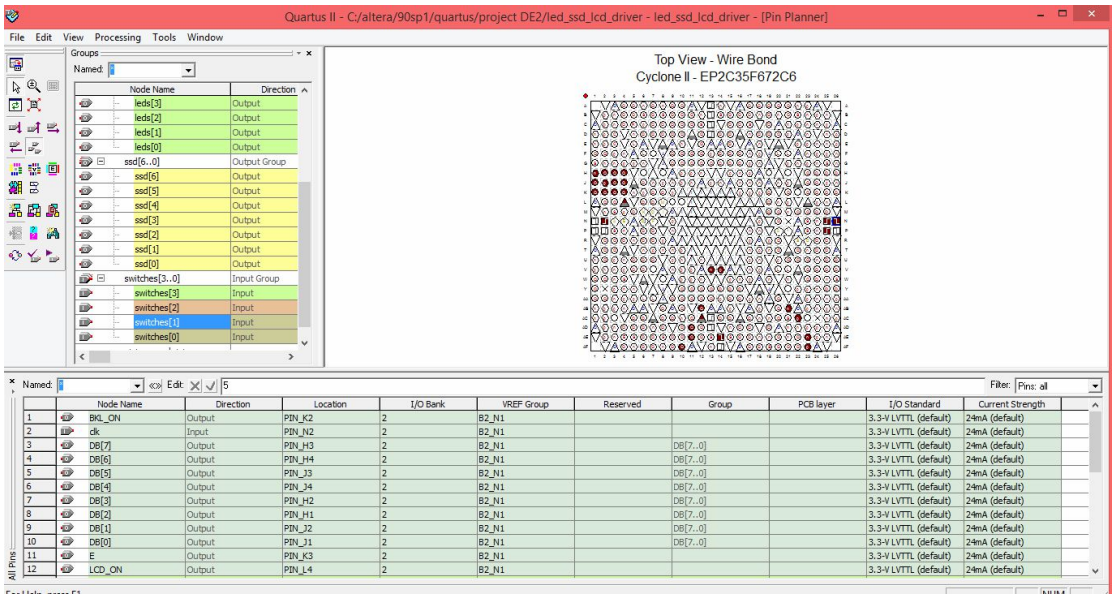
Στη συνέχεια, εισάγουμε τον αριθμό τέσσερα (“0100”) και παρατηρούμε ότι εγγράφεται κατευθείαν μετά την κατάσταση ReturnHome, εμφανίζοντας στον DB την τιμή “00110100”. Μετά την εγγραφή ακολουθεί ξανά η κατάσταση ReturnHome και εναλλάσσεται με τον αριθμό που δίνουμε στη συνέχεια μέχρι το τέλος της προσομοίωσης. Στην έξοδο των leds και της οθόνης επτά τομέων απεικονίζονται οι αριθμοί που δίνονται ως είσοδοι με τον τρόπο που εξηγήσαμε πιο πάνω.

7.2.5 Ορισμός ακροδεκτών (pin assigments)

Σε αυτό το βήμα γίνεται αντιστοίχιση των εισόδων και εξόδων του κυκλώματος με συγκεκριμένους ακροδέκτες της διάταξης που πρόκειται να προγραμματίσουμε. Έχουμε ήδη ορίσει ότι η διάταξή μας είναι ένα FPGA της εταιρείας ALTERA, διαφορετικής οικογένειας για καθεμιά από τις δύο περιπτώσεις. Για να ορίσουμε σε ποιούς ακροδέκτες του τσιπ θα συνδεθούν οι είσοδοι και οι έξοδοι του κυκλώματος επιλέγουμε **Menu/ Assignments/ Pins**, όπως φαίνεται και στο Σχήμα 7.8 και 7.9.



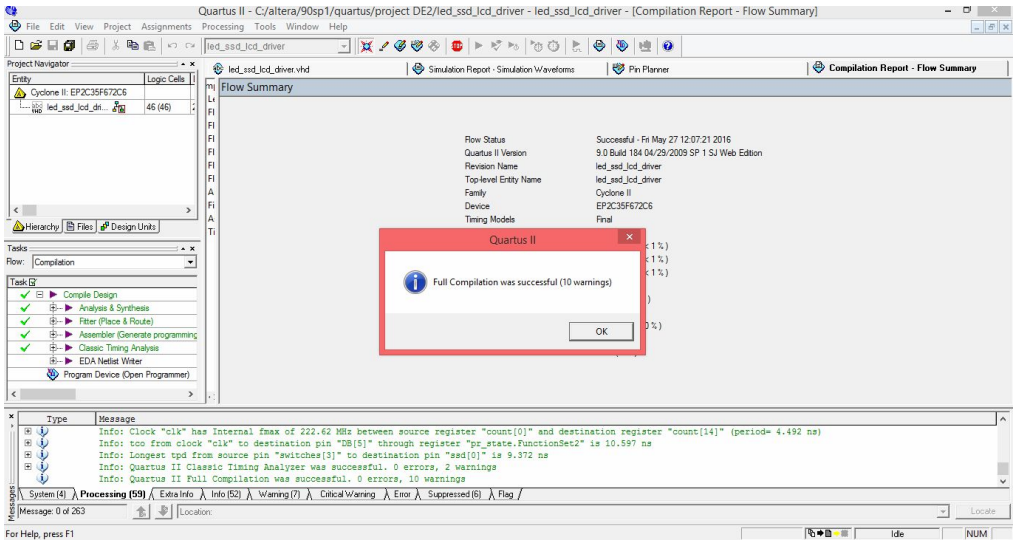
Σχήμα 7.8 Παράθυρο ορισμού ακροδεκτών για το FLEX10K



Σχήμα 7.9 Παράθυρο ορισμού ακροδεκτών για το Cyclone II

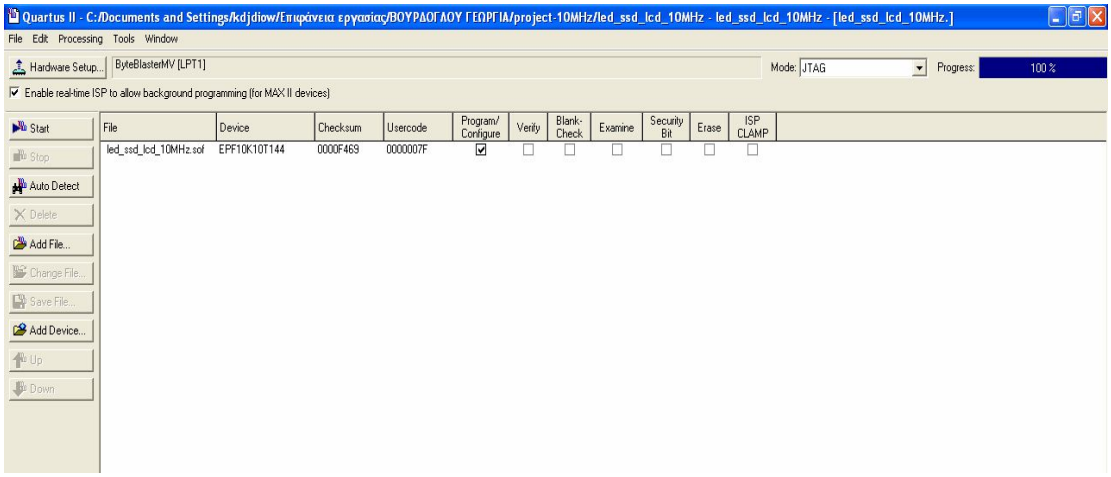
7.2.6 Προγραμματισμός (διαμόρφωση) του κυκλώματος

Αφού ολοκληρωθεί ο ορισμός ακροδεκτών και πριν πραγματοποιηθεί η διαμόρφωση του FPGA θα πρέπει να γίνει Μετάφραση (Compilation) ώστε να δημιουργηθεί το αρχείο *.sof, το οποίο περιέχει όποιες πληροφορίες χρειάζονται για τη διαμόρφωση του FPGA. Η Μετάφραση γίνεται από το **Menu/ Processing/ Start Compilation**. Η Μετάφραση περιλαμβάνει την Ανάλυση & Σύνθεση, τη Χρονική Ανάλυση και την παραγωγή των αρχείων προγραμματισμού του FPGA. Μόλις ολοκληρωθεί η Μετάφραση πρέπει να εμφανιστεί το μήνυμα του Σχήματος 7.10, για να μπορέσουμε να συνεχίσουμε.

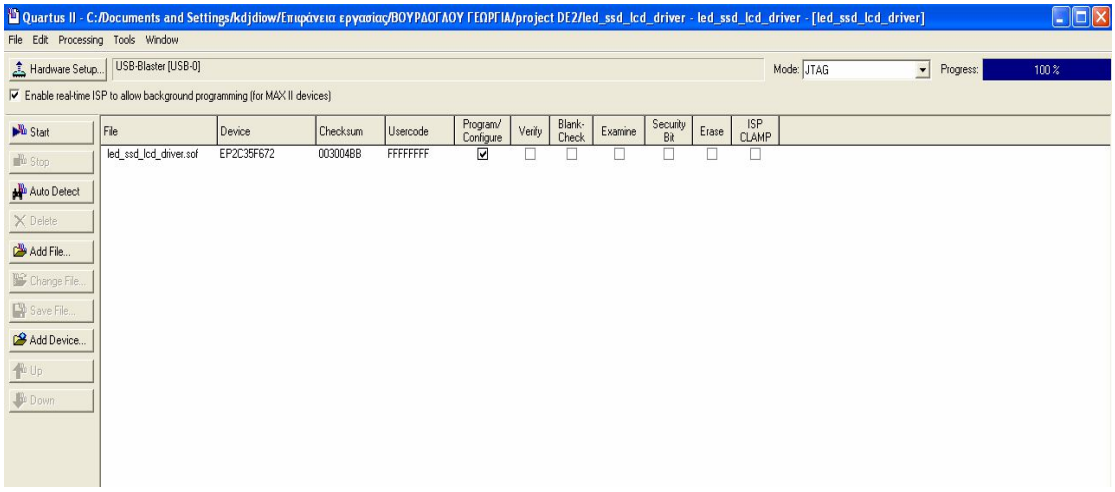


Σχήμα 7.10 Μετάφραση (Compilation) του VHDL αρχείου

Μετά την επιτυχή ολοκλήρωση της Μετάφρασης μπορεί να γίνει η διαμόρφωση του FPGA με τον Programmer, ο οποίος βρίσκεται στο Menu/ Tools/ Programmer. Στο παράθυρο του Programmer πρέπει να επιλέξουμε το κατάλληλο υλικό (Hardware Setup). Οι επιλογές του Hardware Setup είναι ο οδηγός Byte Blaster για προγραμματισμό μέσω παράλληλης θύρας, τον οποίο χρησιμοποιούμε στην αναπτυξιακή πλακέτα LP-2900 (που βασίζεται στην οικογένεια FLEX10K) και ο USB Blaster για προγραμματισμό μέσω της θύρας USB, τον οποίο χρησιμοποιούμε στην αναπτυξιακή πλακέτα DE2 (που βασίζεται στην οικογένεια Cyclone II). Αν δεν είναι ήδη επιλεγμένο κα πρέπει να επιλέξουμε το τελικό αρχείο διαμόρφωσης (.sof) που δημιουργήθηκε από την μετάφραση. Κατόπιν, επιλέγοντας Start αρχίζει η διαδικασία διαμόρφωσης. Η πρόοδος της διαδικασίας φαίνεται στην μπάρα πάνω δεξιά της οθόνης, η οποία απεικονίζεται στο Σχήμα 7.11 και στο Σχήμα 7.12.



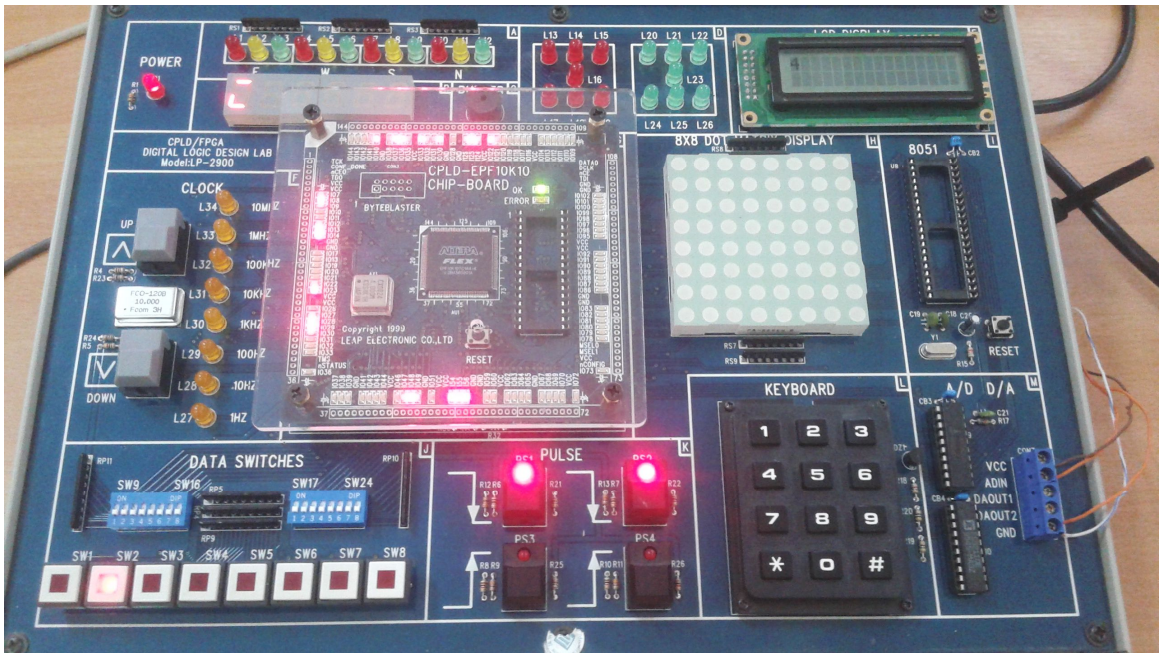
Σχήμα 7.11 Το παράθυρο του Programmer για την αναπτυξιακή πλακέτα LP-2900



Σχήμα 7.12 Το παράθυρο του Programmer για την αναπτυξιακή πλακέτα DE2

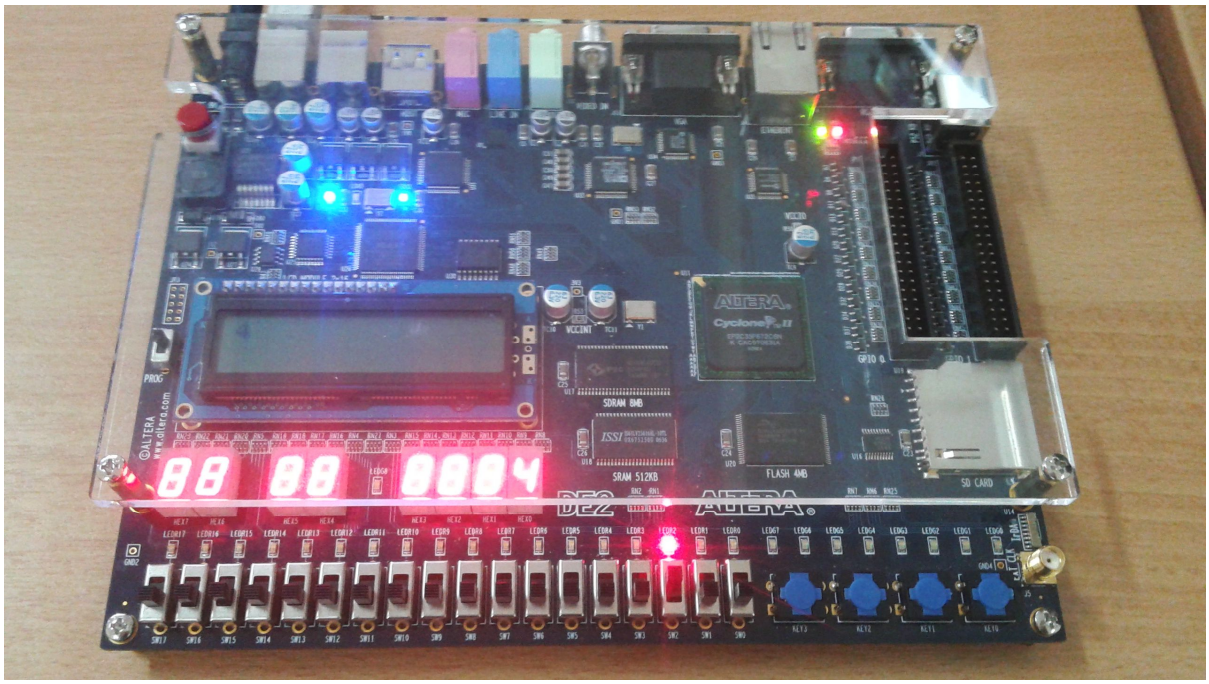
Αφού έχει γίνει η διαμόρφωση του συστήματος τα δεδομένα μας πηγαίνουν κατευθείαν στο FPGA. Τέλος, για να επιβεβαιώσουμε ότι το κύκλωμά μας λειτουργεί σωστά πρέπει να το δοκιμάσουμε δίνοντας κατάλληλες εισόδους 0 ή 1 από τους διακόπτες. Τα αποτελέσματα των εξόδων εμφανίζεται στα SMD leds, στην οθόνη επτά τομέων και στην οθόνη lcd στην αναπτυξιακή πλακέτα LP-2900, όπως και στο ανάπτυγμα DE2, με μόνη διαφορά στην έξοδο των leds, η οποία γίνεται από τα κόκκινα leds.

Στο Σχήμα 7.13 όπου απεικονίζεται η λειτουργία του κυκλώματος στην πλακέτα LP-2900, έχουμε ορίσει τις εισόδους $Switches_{0,1,3}$ = 0 ($SW_{4,3,1}$) και $Switch_2$=1 (SW_2). Άρα σύμφωνα με τη λειτουργία των leds, της οθόνης επτά τομέων και της lcd οθόνης που εξηγήσαμε σε προηγούμενο κεφάλαιο, διαπιστώνουμε ότι θα πρέπει να ανάβει μόνο το SMD led_8 , στην οθόνη επτά τομέων τα $LEDs_{23,28,29}$ και στην οθόνη LCD να απεικονίζεται ο αριθμός τέσσερα.



Σχήμα 7.13 Λειτουργία του κυκλώματος στην πλακέτα LP-2900

Στο Σχήμα 7.14 παρουσιάζεται η λειτουργία του κυκλώματος στο ανάπτυγμα DE2, έχουμε ορίσει τις εισόδους $Switches_{0,1,3}$ = 0 ($SW_{0,1,3}$) και $Switch_2$=1 (SW_2). Έτσι έχουμε ως αποτέλεσμα να ανάβει μόνο το $LEDR_2$ και στην οθόνη επτά τομέων, όπως και στην οθόνη lcd να απεικονίζεται ο αριθμός τέσσερα.



Σχήμα 7.14 Λειτουργία του κυκλώματος στην πλακέτα DE2

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην εργασία αυτή περιγράφηκαν λεπτομερώς οι διαδικασίες σχεδιασμού λογισμικού ελεγκτών για την οδήγηση απλών οθονών, όπως οι συστοιχίες LEDs, οι απεικονίσεις επτά τομέων (SSD) και οι οθόνες LCD. Για να επιτευχθεί αυτό, μελετήθηκε σε βάθος η γλώσσα περιγραφής υλικού VHDL, στην οποία γράφηκε ο κώδικας, όπως και η σχεδίαση μηχανών πεπερασμένων καταστάσεων, στην οποία είναι βασισμένη η περιγραφή της οθόνης LCD. Η σχεδίαση του λογισμικού αποδόθηκε με το λογισμικό Quartus II. Σε αυτό το εργαλείο σχεδίασης πραγματοποιήθηκε και η αντίστοιχη προσομοίωση δίνοντας στις εισόδους του κυκλώματος μερικές τιμές, έτσι ώστε να δούμε τις τιμές που θα πάρουμε ως αποτελέσματα στις εξόδους για να επιβεβαιωθεί η σωστή λειτουργία του κυκλώματος.

Οι βασικοί στόχοι επιτεύχθηκαν μέσω της υλοποίησης του κυκλώματος σε FPGA. Χρησιμοποιήθηκε η εκπαιδευτική αναπτυξιακή πλακέτα LP-2900 που βασίζεται στη διάταξη EPF10K10TC144-4 της οικογένειας FLEX10K της εταιρείας ALTERA, με την οποία είμαστε εξοικειωμένοι από το εργαστήριο των Προηγμένων Ψηφιακών Συστημάτων του Τομέα της Αρχιτεκτονική Υπολογιστών & Βιομηχανικών Εφαρμογών του τμήματος Μηχανικών Πληροφορικής Τ.Ε. Ακόμη, υλοποιήθηκε στην αναπτυξιακή πλακέτα DE2 που βασίζεται, σε αντίθεση με την LP-2900, στη διάταξη EP2C35F672C6 της οικογένειας Cyclone II, επίσης, της εταιρείας ALTERA.

Η πτυχιακή αυτή εργασία έγινε στο τμήμα Μηχανικών Πληροφορικής Τ.Ε. το ακαδημαϊκό έτος 2015-2016 με επιβλέποντα καθηγητή τον κ. Καλόμοιρο Ιωάννη, τον οποίο θα ήθέλα να ευχαριστήσω θερμά τόσο για την καθοδήγηση κατά τη διάρκεια της εκπόνησής της, αλλά και για όλες τις γνώσεις που μου μεταλαμπάδευσε κατά τη διάρκεια των σπουδών μου.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Ιωάννης Καλόμοιρος, «*Εισαγωγή στη VHDL*», Έκδοση 1.2, Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κεντρικής Μακεδονίας, Σέρρες, 2012

2. Ιωάννης Καλόμοιρος, «*Διαφάνειες για το μάθημα Προηγμένα Ψηφιακά Συστήματα*», Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κεντρικής Μακεδονίας, Σέρρες, 2008

3. Ιωάννης Καλόμοιρος, «*Σημειώσεις εργαστηρίου Προηγμένων Ψηφιακών Συστημάτων*», Έκδοση 2η, Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κεντρικής Μακεδονίας, Σέρρες, 2010

4. Volnei A. Pedroni, «*Circuit Design and Simulation with VHDL*», second edition, MIT Press, 2010

5. Volnei A. Pedroni, «*Digital Electronics and design with VHDL*», Elsevier, 2008

6. https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

7. http://web.stanford.edu/class/ee281/handouts/lcd_tutorial.pdf

8. Stephen Brown and Zvonko Vranesic, «*Σχεδίαση Ψηφιακών Συστημάτων Με Την Γλώσσα VHDL*», Εκδόσεις Τζιόλας, 2002

9. https://www.altera.com/

10. Altera, «*DE2 Development and Education Board User Manual*», Version 1.4, 2006

11. Παρασκευάς Κίτσος, «*Σχεδιασμός Ολοκληρωμένων Κυκλωμάτων*», Τμήμα Μηχανικών Πληροφορικής ΤΕ, Αντίρριο, 2016

12. http://www.ekenrooi.net/lcd/lcd.shtml