ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΤΕ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αριθμός :1404

ΠΡΟΣΑΡΜΟΓΗ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ ΤΟΥ ΜΑΘΗΜΑΤΟΣ "ΨΗΦΙΑΚΗ ΣΧΕΔΙΑΣΗ" ΣΤΟ ΠΕΡΙΒΑΛΛΟΝ QUARTUS ΤΗΣ ALTERA

ΣΠΟΥΔΑΣΤΕΣ:

ΤΡΙΑΝΤΟΣ ΒΑΣΙΛΕΙΟΣ

ΠΕΤΡΟΣ ΜΠΑΡΚΟΥΛΑΣ

ΕΙΣΗΓΗΤΕΣ:

ΓΑΛΕΤΑΚΗΣ ΕΜΜΑΝΟΥΗΛ , ΕΡΓ. ΣΥΝΕΡΓΑΤΗΣ

ΒΛΑΧΟΠΟΥΛΟΣ ΠΕΤΡΟΣ , ΚΑΘΗΓΗΤΗΣ

ΠΑΤΡΑ 2014

ΠΕΡΙΕΧΟΜΕΝΑ

Πίνακας Εικόνων

Πίνακας Σχημάτων

Πίνακας Πινάκων

1. ΕΙΣΑΓΩΓΗ

Η παρούσα σειρά των εργαστηριακών ασκήσεων σε VHDL ασχολείται με την σχεδίαση και υλοποίηση απλών και πιο σύνθετων ψηφιακών κυκλωμάτων με την χρήση της γλώσσας VHDL, στο περιβάλλον του Quartus II της Altera.

Θεωρείται δεδομένη η γνώση βασικών εννοιών των Ψηφιακών Ηλεκτρονικών και μια ελάχιστη εξοικείωση με την χρήση ηλεκτρονικού υπολογιστή. Σε κάθε άσκηση δίνονται τα βασικά θεωρητικά στοιχεία όσον αφορά τα κυκλώματα και τις ψηφιακές έννοιες που διαπραγματεύονται. Για πιο ολοκληρωμένη εμβάθυνση στις λειτουργίες της γλώσσας προτείνεται η χρήση κάποιου βιβλίου πάνω στην VHDL, από τα προτεινόμενα στην βιβλιογραφία. της κάθε άσκησης.

Πρέπει να τονιστεί ότι σκοπός της σειράς δεν είναι η πλήρης εκμάθηση της γλώσσας VHDL. Κάτι τέτοιο δεν θα ήταν εφικτό να γίνει μέσα σε 7 εργαστηριακές ασκήσεις 2-3 ωρών η κάθε μία. Αυτό που ελπίζεται να πετύχει η σειρά είναι μια καλή εισαγωγή στις βασικές έννοιες της VHDL αλλά και η εξοικείωση με το περιβάλλον του Quartus II καθώς και η σχεδίαση ψηφιακών κυκλωμάτων με την χρήση ηλεκτρονικού υπολογιστή.

2. Εισαγωγή στα FPGA

2.1. Συσκευές προγραμματιζόμενης λογικής

Η προγραμματιζόμενη λογική είναι ο τρόπος με τον οποίο ένα μεγάλο τμήμα των μηχανικών εφαρμόζει τη προσαρμοσμένη λογική τους, εάν εκείνη η λογική είναι μια απλή I/O θύρα ή μια σύνθετη μηχανή. Η περισσότερη προγραμματιζόμενη λογική εφαρμόζεται με κάποιο τύπο του HDL που ελευθερώνει το μηχανικό από το να πρέπει να παράγει και να ελαχιστοποιεί λογικές εκφράσεις (Boolean) κάθε φορά που σχεδιάζεται μια νέα λογική σχέση. Τα πλεονεκτήματα της προγραμματιζόμενης λογικής περιλαμβάνουν τη γρήγορη προσαρμογή με σχετικά περιορισμένη δαπάνη που επενδύεται στα εργαλεία και την υποστήριξη.

Η ευρεία διαθεσιμότητα των ευέλικτων προγραμματιζόμενων προϊόντων λογικής έχει φέρει τις ικανότητες σχεδίου προσαρμοσμένης λογικής σε πολλά άτομα και μικρότερες επιχειρήσεις που χωρίς αυτά δεν θα είχαν τους οικονομικούς πόρους και το προσωπικό για να παράγουν ένα πλήρως προσαρμοσμένο IC. Αυτές οι συσκευές είναι διαθέσιμες σε ένα ευρύ φάσμα μεγεθών, των λειτουργικών τάσεων, και των ταχυτήτων, το οποίο κάθε άλλο παρά εγγυάται ότι για οποιαδήποτε εφαρμογή μπορεί να αντιστοιχηθεί μια συσκευή. Επιλέγοντας αυτή τη συσκευή απαιτείται κάποια έρευνα, επειδή κάθε κατασκευαστής έχει ελαφρώς διαφορετική ειδικότητα και γκάμα προϊόντων.

Η τεχνολογία της προγραμματιζόμενης λογικής εξελίσσεται γρήγορα, και οι κατασκευαστές προσφέρουν συνεχώς συσκευές με αυξημένες ικανότητες και ταχύτητες.

2.2. Προσαρμοσμένη και προγραμματιζόμενη λογική

Πέρα από τη χρήση διακριτών 7400 ICs, η προσαρμοσμένη λογική εφαρμόζεται σε μεγαλύτερα ICs που είτε παράγονται με προσαρμοσμένες μάσκες σε ένα εργοστάσιο ή έχουν προγραμματιστεί με συγκεκριμένα δεδομένα σε διαφορετικά σημεία μετά την κατασκευή. Προσαρμοσμένα ICs, ή (ASIC), είναι η πιο ευέλικτη επιλογή διότι, όπως με κάθε τι προσαρμόσιμο υπάρχουν λιγότεροι περιορισμοί σχετικά με το πώς η εκάστοτε εφαρμογή εφαρμόζεται.

Επειδή υπάρχουν προσαρμοσμένα ICs για κάθε τύπου εφαρμογή, επιτυγχάνεται σχέση χαμηλής αξίας με υψηλές ταχύτητες πυρήνα. Τα μειονεκτήματα των προσαρμοσμένων ICs είναι ο μεγάλος και δαπανηρός κύκλος ανάπτυξης και η αδυναμία άμεσων αλλαγών στη λειτουργία. Η ανάπτυξη των προσαρμοσμένων IC’s είναι χρονοβόρα, γιατί πριν την παραγωγή και την τελική υλοποίηση πρέπει να έχει γίνει σχεδίαση της πλήρης λογικής λειτουργίας του προσαρμοσμένου πλέον κυκλώματος. Από τη στιγμή που το προσαρμοσμένο IC παραχθεί λειτουργικές αλλαγές επηρεάζουν την αρχική σχεδίαση του κυκλώματος επομένως το κόστος αγγίζει αυτό του κόστους παραγωγής.

Συσκευές προγραμματιζόμενης λογικής (PLDs) είναι μια εναλλακτική των ASICs. Το PLD αποτελείται από δεδομένα λογικής γενικής χρήσης τα οποία μπορούν να χρησιμοποιηθούν με πολλές παραλλαγές, σύμφωνα με τη λογική του σχεδιασμού ενός μηχανικού. Αυτή η προγραμματιζόμενη συνδεσιμότητα λειτουργεί ως πρόσθετο με κρυμμένο κώδικα που αναλαμβάνει αυτού του τύπου τις συνδέσεις μέσα στο κύκλωμα. Το κύριο όφελος της τεχνολογίας PLD είναι ότι ένα σχέδιο μπορεί γρήγορα να φορτωθεί σε μια

συσκευή PLD, παρακάμπτοντας τη χρονοβόρα και δαπανηρή διαδικασία παραγωγής ενός προσαρμοσμένου IC. Αυτό έχει σαν αποτέλεσμα τη διόρθωση ενός κολλήματος η ενός λάθους στον κώδικα να μπορεί να επιτευχθεί πολύ πιο γρήγορα και πολλές φορές μέσα και στην ίδια την συσκευή του PLD. Μερικά PLDs είναι μια φορά προγραμματίσημα, και άλλα μπορούν να αναπρογραμματιστούν.

Το μειονέκτημα των PLDs είναι το αντίτιμο του κρυμμένου κώδικα λογικής που επιτυγχάνει τις λογικές συνδέσεις (όπως αναφέρθηκε παραπάνω) το οποίο έχει σαν αποτέλεσμα υψηλότερο κόστος παραγωγής, χαμηλότερες ταχύτητες και την αυξημένη κατανάλωση ενέργειας. Καθυστερήσεις στη διάδοση δεδομένων είναι αναπόφευκτες σε όλες τις δομές πυριτίου και κυρίως στις πιο σύνθετες από αυτές (όσο πιο σύνθετο είναι το μονοπάτι που μια πληροφορία πρέπει να διανύσει τόσο μεγαλύτερη είναι και η καθυστέρηση). Επομένως, μια προγραμματιζόμενη συσκευή θα είναι πιο αργή από μια προσαρμοσμένη, γιατί η πρώτη αποτελείται από πιο σύνθετες δομές σύνδεσης που προκαλούν χρονοκαθυστέρηση. Το ίδιο ισχύει και για την κατανάλωση ενέργειας.

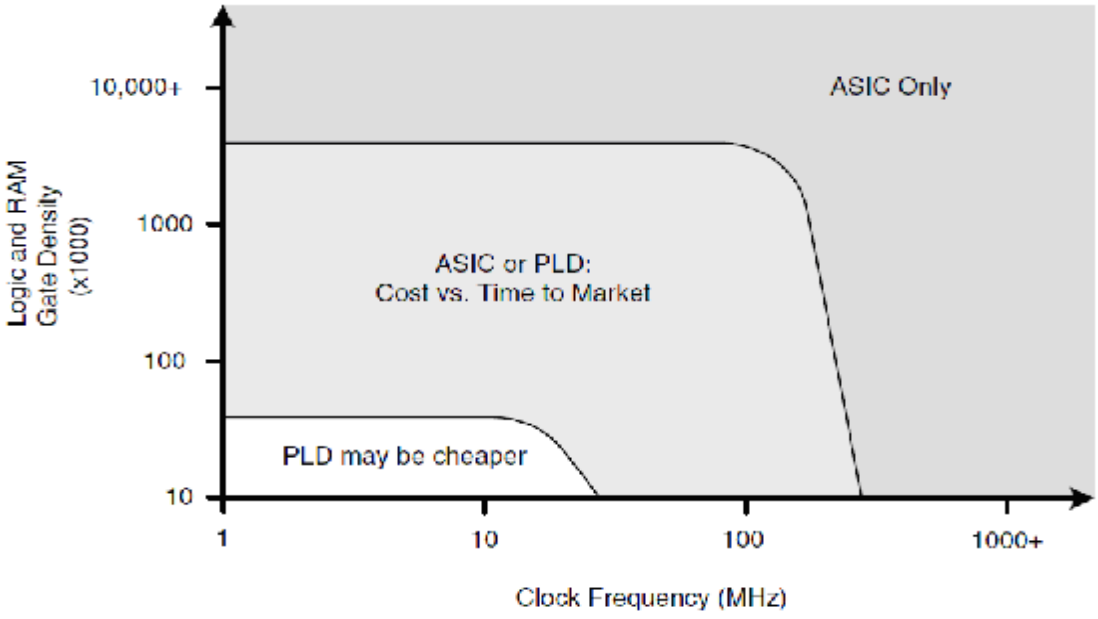
Παρά το μειονέκτημα της προγραμματιζόμενης λογικής, η τεχνολογία στο σύνολό της έχει προχωρήσει σημαντικά και είναι εξαιρετικά δημοφιλής, ως αποτέλεσμα της ανταγωνιστικής τιμολόγησης, υψηλά επίπεδα απόδοσης, και ιδίως, γρήγορο χρόνο διάθεσης στην αγορά. Χρόνος διάθεσης στην αγορά είναι μια ιδιότητα που είναι δύσκολο να ποσοτικοποιηθεί, αλλά εκτιμάται ως κρίσιμη για την επιτυχία. Τα PLDs συνεισφέρουν σε ένα συντομότερο κύκλο ανάπτυξης, γιατί σχέδια μπορούν να κατασκευαστούν γρήγορα, τα σφάλματα να επιλυθούν ευκολότερα και τα προϊόντα φτάνουν στον τελικό χρήστη πριν από τις τεχνολογίες ASIC που είναι ακόμη στην παραγωγή. Ακόμα καλύτερα, αν ένα πρόβλημα βρεθεί μπορεί να επιδιορθωθεί με σημαντικά λιγότερο κόστος και χρόνο. Κατά τους πρώτους καιρούς της προγραμματιζόμενης λογικής, τα PLDs δεν μπορούσαν να αναπρογραμματιστούν, που σημαίνει ότι ένα πρόβλημα θα μπορούσε να οδηγήσει στην απόσυρση του προϊόντος που έχει ήδη αποσταλεί. Πολλά σύγχρονα επαναπρογραμματιζόμενα PLDs επιτρέπουν σφάλματα υλικού να επιδιορθωθούν με αναβάθμιση λογισμικού από τον τελικό χρήστη.

Το κόστος και οι επιδόσεις είναι πιθανότατα οι πλέον πολυσυζητημένοι συμβιβασμοί που εμπλέκονται στη χρήση προγραμματιζόμενων λογικών. Το πλήρες φάσμα των εφαρμογών στις οποίες τα PLDs ή τα ASICs μπορούν να χρησιμοποιηθούν χωρίζονται σε τρεις κατηγορίες, όπως φαίνεται στην Σχήμα 21. Για τεχνολογίες αιχμής η χρήση της ASIC είναι η μόνη δυνατή λύση, λόγο χρονοδιαγράμματος και απαιτήσεων. Σε μέσες εφαρμογές, η συχνότητα λειτουργίας και η σύνθετη λογική είναι τέτοιες ώστε τα PLD να είναι σε θέση να επιλύουν το πρόβλημα, αλλά σε υψηλότερο κόστος ανά μονάδα από ένα ASIC. Εδώ, η απόφαση πρέπει να γίνει μεταξύ ευελιξίας και χρόνου διάθεσης στην αγορά σε σχέση με το χαμηλότερο κόστος ανά μονάδα προϊόντος. Στις πιο απλές εφαρμογές οι συχνότητες λειτουργίας και οι απαιτήσεις της λογικής του κυκλώματος είναι αρκετά κάτω από την τρέχουσα κατάσταση της τεχνολογίας πυριτίου που τα PLD’s μπορούν να ανταποκριθούν ή έστω να υπερτερήσουν του κόστους ενός ASIC.

Μπορεί να ακούγεται παράξενο ότι ένα PLD μπορεί να είναι λιγότερο ακριβό από ένα προσαρμοσμένο τσιπ. Οι λόγοι για αυτό είναι ένας συνδυασμός του μεγέθους του πυριτίου και την τιμολόγηση του όγκου. Δύο από τους σημαντικότερους παράγοντες του κόστους παραγωγής πυριτίου είναι το μέγεθός και η απόδοση κατασκευής. Καθώς γίνεται μικρότερο, πιο πολλά από αυτά μπορούν να κατασκευαστούν ταυτόχρονα στον ίδιο αριθμό πλακιδίων με χρήση των ίδιων πόρων. Η διαδικασία παραγωγής ολοκληρωμένου κυκλώματος (IC) έχει ορισμένους περιορισμούς με το σημαντικότερο να είναι το ποσοστό των λειτουργικών

κυκλωμάτων σε σχέση με τη συνολική παρτίδα που έχει παραχθεί. Ορισμένα αναπτύσσουν μικροσκοπικά ελαττώματα κατά τη διάρκεια της κατασκευής που τα καθιστούν άχρηστα. Απόδοση είναι συνάρτηση πολλών μεταβλητών, όπως η αξιοπιστία της ομοιόμορφης κατασκευής μιας σύνθετης δομής λόγω της επικρατούσας κατάστασης της τεχνολογίας τη συγκεκριμένη στιγμή. Από αυτά τα δύο σημεία, συνάγεται ότι ένα τσιπ πυριτίου θα είναι λιγότερο ακριβό για την παραγωγή αν είναι μικρό και χρησιμοποιεί μια διαδικασία τεχνολογίας που είναι ώριμη και έχει υψηλή απόδοση.

Στις απλές εφαρμογές ένα μικρό PLD και ένα μικρό ASIC μπορούν να έχουν την ίδια τεχνολογία και τα ίδια χαρακτηριστικά απόδοσης με την τελευταία να τα διαφοροποιεί σε θέμα κόστους παραγωγής.



Σχήμα 21 Συχνότητα και πυκνότητα πυλών σε ASIC

Επιπλέον, οι ακατέργαστες δαπάνες συσκευασίας είναι πιθανό να είναι συγκρίσιμες λόγω της ωρίμανσης των σταθερών υλικών συσκευασίας και των τεχνολογιών. Η διαφορά στο κόστος προκύπτει από τη λύση που απαιτεί μικρότερο κύκλωμα καθώς επίσης μικρότερο κόστος παραγωγής αλλά και διανομής των τσιπ.

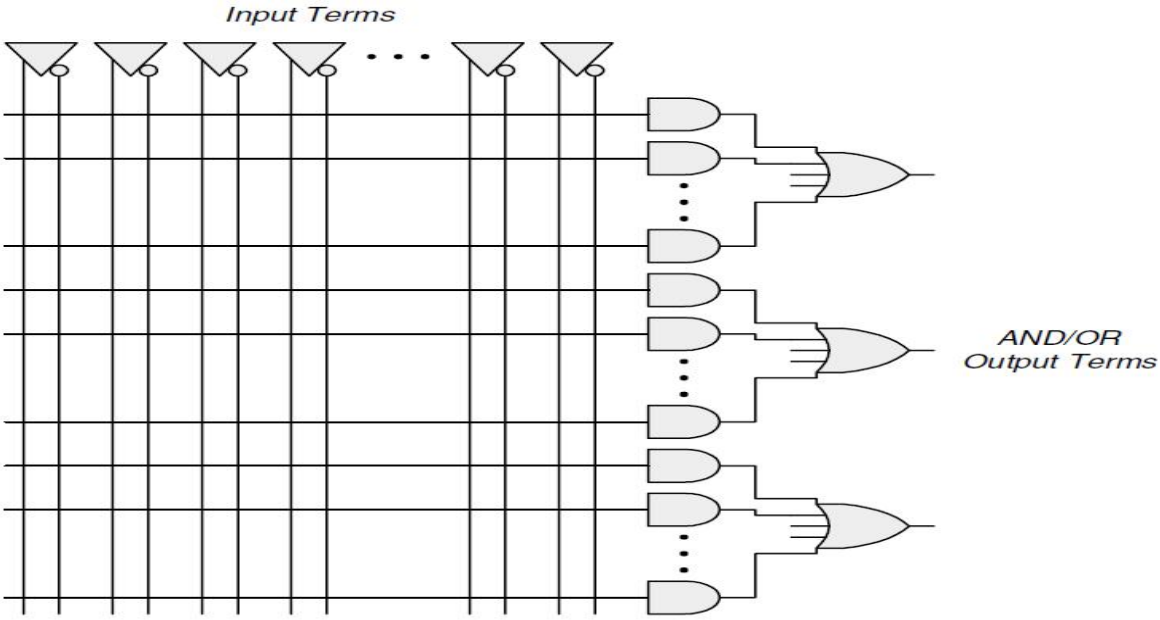
Το μέγεθος του κυκλώματος αποτελείται από δύο χαρακτηριστικά: πόση λογική απαιτείται και πόσα I/O pins απαιτούνται. Ενώ το μέγεθος των πυλών λογικής έχει μειωθεί με την πάροδο του χρόνου, το μέγεθος των I/O δεν έχει αλλάξει στον ίδιο βαθμό. Τα I/O τοποθετούνται συχνά στην περίμετρο ενός κυκλώματος. Εάν ο αριθμός των απαιτούμενων I/O δεν μπορεί να τοποθετηθεί κατά μήκος της υπάρχουσας περιμέτρου του κυκλώματος, το κύκλωμα πρέπει να διευρυνθεί ακόμα κι αν δεν απαιτείται από τη λογική λειτουργίας. Ένα ισορροπημένο σχέδιο είναι βέλτιστο, επειδή η περιοχή πυριτίου χρησιμοποιείται αποτελεσματικά από τη λογική και τις δομές κυκλωμάτων. Ένα ολοκληρωμένο κύκλωμα με περιορισμό στη λογική λειτουργίας χρησιμοποιεί σχεδόν όλη την περιοχή πυριτίου από τις εσωτερικές δομές λογικής λειτουργίας. Ο περιορισμός του μεγέθους του κυκλώματος είναι μεγαλύτερης σημασίας λόγου του ότι το τσιπ πρέπει να έχει ένα ελάχιστο μέγεθος έτσι ώστε να υποστηρίζει τον αντίστοιχο αριθμό των pins.

Πολλές απλές εφαρμογές παύουν να έχουν περιορισμό μεγέθους λόγο ότι η τεχνολογία παραγωγής τσιπ πυριτίου έχει προχωρήσει σε τέτοιο βαθμό έτσι ώστε

περισσότερες πύλες μπορούν να χωρέσουν σε μικρότερες περιοχές. Η λογική μπορεί να απαιτεί λιγότερο χώρο, αλλά το I/O όχι. Όταν επέλθει περιορισμός χώρου σε ένα ολοκληρωμένο κύκλωμα τα ASIC και CPLD μπορούν να χρησιμοποιούν το ίδιο μέγεθος κυκλώματος μειώνοντας το κόστος. Μεταξύ των πιο βασικών τύπων PLDs είναι οι συσκευές Generic Array Logic (GAL). Τα GAL's θεωρούνται παραλλαγές της παλαιότερης αρχιτεκτονικής (PAL) που είναι τώρα ουσιαστικά ξεπερασμένη. Ο όρος PAL χρησιμοποιείται ευρέως μέχρι σήμερα, αλλά συνήθως αναφέρεται στις συσκευές GAL ή σε άλλες παραλλαγές PLD. Τα PAL's ξεπεράστηκαν, επειδή τα GALs μπορούν να εκτελέσουν όλες τις λειτουργίες των PAL's με τη διαφορά ότι είναι φτηνότερα, μικρότερα και ευρέως διαθέσιμα.

Μπορεί να αποδειχθεί μέσω της άλγεβρας Boole ότι οποιαδήποτε λογική έκφραση μπορεί να αναπαρασταθεί ως ένα σύνθετο άθροισμα προϊόντων. Επομένως, παρέχοντας ένα προγραμματιζόμενο πίνακα με πύλες AND ή OR, η λογική μπορεί να εφαρμοστεί για να εξυπηρετήσει μια συγκεκριμένη εφαρμογή. Οι συσκευές GAL παρέχουν μια εκτενή επιλογή από πύλες AND, όπως φαίνεται στο παρακάτω Σχήμα 22. Τόσο οι κανονικές όσο και οι ανεστραμμένες είσοδοι είναι διαθέσιμες προς κάθε πύλη AND. Οι έξοδοι των AND πυλών (προϊόντα) καταλήγουν σε OR πύλες (αθροίσματα) για να παραχθούν οι ορισμένες από το χρήστη Boolean εκφράσεις.

Κάθε τομή μιας οριζόντιας πύλης AND και ενός κάθετου όρου εισαγωγής είναι μια προγραμματίσημη σύνδεση.



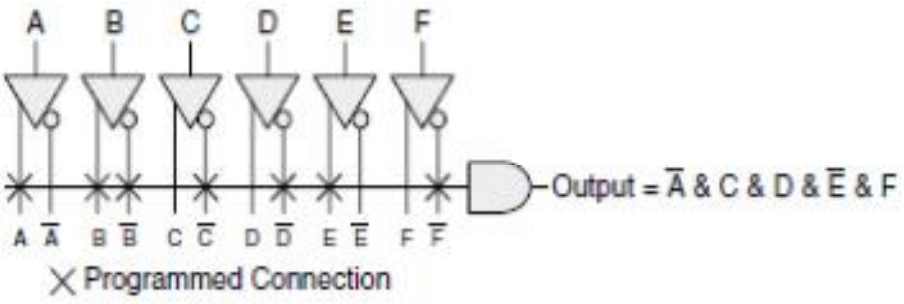
Σχήμα 22 GAL/PAL AND /OR δομή

Οι σημερινές συσκευές συνήθως βασίζονται στην τεχνολογία E-EPROM και CMOS διακόπτες για να επιτρέψουν το αμετάβλητο επαναπρογραμματισμό. Η αρχική διαμόρφωση μιας σύνδεσης εξομοιώνει μια κοινή ασφάλεια (ρελλέ) όπου η είσοδος συνδέεται με την είσοδο της πύλης AND. Όταν η σύνδεση διακοπεί η είσοδος της πύλης AND αποσυνδέεται από την είσοδο της συσκευής και παίρνει την τιμή 1 για να απομονώσει την εν λόγο είσοδο από τη λογική έκφραση Boolean. Η διαμόρφωση και ενός προγραμματιζόμενου GAL κυκλώματος με πύλες AND φαίνεται στο Σχήμα 23.

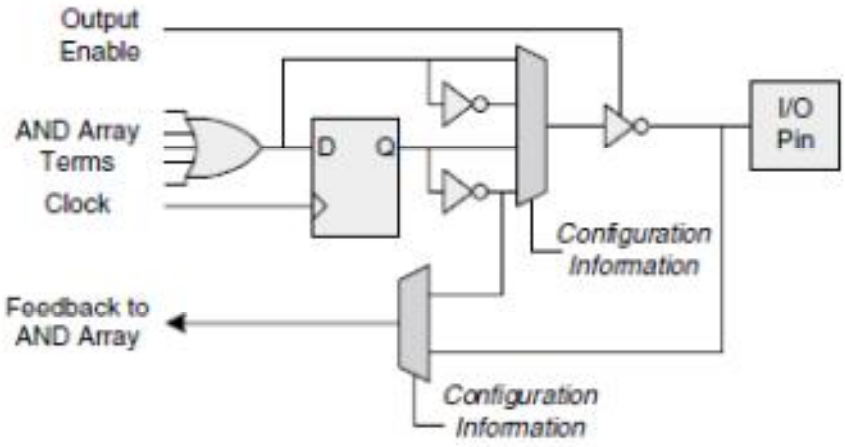
Με τη δυνατότητα του πλήρους προγραμματισμού των διατάξεων με πύλες AND, οι συνδέσεις μεταξύ των πυλών OR μπορούν να υλοποιηθούν δύσκολα. Κάθε GAL συσκευή

τροφοδοτεί έναν διαφορετικό αριθμό πυλών AND στις εισόδους των πυλών OR. Εάν μια ή περισσότερες από τις πύλες AND δεν χρειάζεται σε μια έκφραση Boolean μπορεί να απομονωθεί οδηγώντας την έξοδο της σε λογικό 0. Αυτό επιτυγχάνεται αφήνοντας την μια είσοδο της πύλης AND απρογραμμάτιστη. Υπενθυμίζουμε ότι οι είσοδοι στο κύκλωμα με πύλες AND παρέχονται τόσο στην κανονική όσο και στην συμπληρωματική τους μορφή. Όταν όλες οι συνδέσεις των πυλών AND παραμένουν ανέπαφες, πολλαπλές εκφράσεις της μορφής A&Ā=0 προκύπτουν με αποτέλεσμα οι έξοδοι των πυλών να οδηγούνται στο 0 χωρίς να επηρεάζουν την λειτουργία των πυλών OR.

Το μεγαλύτερο μέρος μιας προσαρμοσμένης λογικής ενός GAL κυκλώματος επιτυγχάνεται προγραμματίζοντας την διάταξη των πυλών AND. Εντούτοις, η επιλογή flip-flops, πολικοτήτων OR/NOR και η διαμόρφωση εισόδων/εξόδων επιτυγχάνεται προγραμματίζοντας ένα διαμορφώσιμο I/O και η τελική δομή ονομάζεται macrocell. Η βασική ιδέα πίσω από ένα macrocell είναι να καθορίζει το πώς διαχειρίζονται οι εκφράσεις Boolean (AND και OR) και το πώς τα pin του I/O κυκλώματος λειτουργούν. Μια σχηματική άποψη ενός GAL macrocell παρουσιάζεται στο Σχήμα 24. Οι διαμορφωτές καθορίζουν την πολικότητα του τελικού όρου OR/NOR, ανεξάρτητα από το εάν αυτός ο όρος καταχωρείται και εάν η ανατροφοδότηση του σήματος οδηγείται κατευθείαν στην έξοδο του flop ή στο pin. Διαμορφώνοντας την έξοδο του macrocell καθορίζεται και η συμπεριφορά του αντίστοιχου pin.



Σχήμα 23Προγραμματισμός εισαγμένων όρων AND



Σχήμα 24 GAL macrocell

Υπάρχουν δύο κοινές συσκευές GAL, το 16V8 και το 22V10, αν και υπάρχουν και άλλες παραλλαγές επίσης. Περιέχουν οκτώ και δέκα macrocells το κάθε ένα, αντίστοιχα. Το 16V8 παρέχει μέχρι και 10 εισαγωγές που τροφοδοτούν την διάταξη AND, ενώ το 22V10

παρέχει 12 εισαγωγές. Μια εισαγωγή από το 22V10 χρησιμεύουν επίσης ως ένα σφαιρικό ρολόι για οποιαδήποτε Flop που ενεργοποιούνται στα macrocells.

Η λογική εξόδου σε ένα 22V10 αξιολογείται ανεξάρτητα για κάθε macrocell μέσω μιας καθορισμένης εντολής AND. Το 16V8 είναι κάπως λιγότερο ευέλικτο, επειδή δεν μπορεί να ανατροφοδοτήσει αυθαίρετα όλες τις εξόδους του macrocell ανάλογα με τις ρυθμίσεις της συσκευής. Επιπλέον, όταν διαμορφώνονται για τον καταχωρημένο τρόπο όπου τα Flops των macrocell είναι χρησιμοποιήσιμα, δύο καθορισμένα pin εισαγωγής χάνονται στις λειτουργίες του ρολογιού και της εξόδου.

Τα GALs είναι αρκετά χαμηλής πυκνότητας PLDs σύμφωνα με τα σύγχρονα πρότυπα, αλλά τα πλεονεκτήματα χαμηλού κόστους και υψηλής ταχύτητας προέρχονται από το μικρό μέγεθός τους. Η εφαρμογή λογικής σε ένα GAL ακολουθεί αρκετά βασικά βήματα. Κατ' αρχάς, η λογική αναπαρίσταται είτε με γραφικό σχηματικό διάγραμμα είτε σε μορφή κειμένου (HDL). Αυτή η αναπαράσταση μετατρέπεται σε μια netlist χρησιμοποιώντας ένα εργαλείο μεταφράσεων ή σύνθεσης. Τέλος, η netlist προσαρμόζεται στη συσκευή που θέλουμε με τη χαρτογράφηση μεμονωμένων λειτουργιών των πυλών στη προγραμματίσιμη διάταξη AND. Λαμβάνοντας υπόψη τη συγκεκριμένη δομή των AND/OR ενός GAL, το λογισμικό σχεδιάζεται για να εκτελεί βελτιστοποιήσεις και μεταφράσεις λογικής για να μετατρέπει τις αυθαίρετες Boolean εκφράσεις σε ένα σύνολο εκφράσεων του προϊόντος. Το αποτέλεσμα αυτής της διαδικασίας είναι μια εικόνα προγραμματισμού που καθορίζει ακριβώς ποιες συνδέσεις πρόκειται να προγραμματιστούν και ποιες πρόκειται να μείνουν στην αρχική τους κατάσταση. Η εικόνα προγραμματισμού περιέχει επίσης άλλες πληροφορίες όπως η διαμόρφωση του macrocell και άλλες ιδιότητες για την συγκεκριμένη συσκευή.

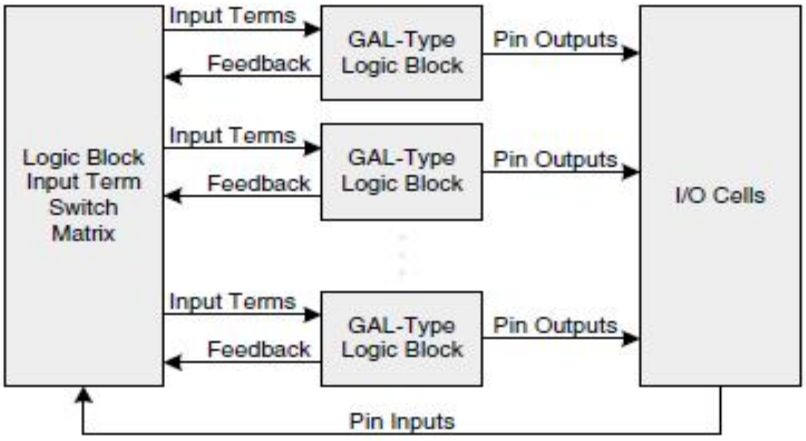
Το σύγχρονο λογισμικό ανάπτυξης PLD επιτρέπει τη σύνθεση GAL οπίσθιου μέρους και τη διαδικασία συναρμολογήσεων χωρίς χειρωνακτική επέμβαση στις περισσότερες περιπτώσεις. Η απλή ροή λογικής μέσω της προγραμματίσημης διάταξης AND μειώνει τις μεταλλαγές για το πώς μια δεδομένη Boolean έκφραση μπορεί να εφαρμοστεί και έχει ως αποτέλεσμα μια πολύ προβλέψιμη λογική συναρμολόγησης. Ένα σήμα εισόδου διαδίδει μέσω ενός pin και μιας δομής, άμεσα στην διάταξη AND μέσω μόλις δύο πυλών, και μπορεί έπειτα είτε να τροφοδοτήσει ένα Flop του macrocell είτε να οδηγηθεί κατευθείαν έξω μέσω ενός I/O pin.Τα στοιχεία λογικής μέσα σε ένα GAL είναι κοντά το ένα στο άλλο ως αποτέλεσμα του μικρού μεγέθους του GAL, το οποίο συμβάλλει στις χαμηλές εσωτερικές καθυστερήσεις διάδοσης. Αυτά τα χαρακτηριστικά επιτρέπουν στην αρχιτεκτονική GAL να αποδίδουν πολύ γρήγορες προδιαγραφές χρονισμού, επειδή τα σήματα ακολουθούν καθορισμένες διαδρομές με χαμηλές καθυστερήσεις διάδοσης.

Το GALs είναι μια τεχνολογία εφαρμογής λογικής με πολύ προβλέψιμες ικανότητες. Εάν η επιθυμητή λογική δεν μπορεί να ενσωματωθεί μέσα στο GAL, μπορεί να μην υπάρξουν και πολλά τα οποία μπορούν να γίνουν χωρίς τη βελτιστοποίηση του αλγορίθμου ή το διαχωρισμό του σχεδίου σε πολλαπλές συσκευές. Εάν η λογική ταιριάζει, αλλά δεν πληροί το διάγραμμα χρονισμού, η λογική πρέπει να βελτιστοποιηθεί, ή μια ταχύτερη συσκευή πρέπει να βρεθεί. Λόγω της βασικής διαδικασίας ενσωμάτωσης και της αρχιτεκτονικής της GAL, δεν υπάρχει η ίδια δυνατότητα για μικροαλλαγές στο προϊόν, όπως μπορεί να γίνει με πιο πολύπλοκα PLDs. Αυτό δεν πρέπει να εκλαμβάνεται ως έλλειψη ευελιξίας εκ μέρους της GAL. Αντίθετα, η GAL κάνει ακριβώς αυτό που λέει ότι κάνει και από εκεί και πέρα είναι στο χέρι του μηχανικού να εφαρμόσει σωστά την τεχνολογία για να λύσει το πρόβλημα. Η απλότητα στης αρχιτεκτονικής του GAL είναι και η μεγαλύτερη δύναμή του.

Σύνθετο PLD, ή CPLD (Complex PLD), είναι ένα macrocell βασισμένο στην επικρατούσα τάση των PLDs στη βιομηχανία σήμερα, παρέχοντας δυνατότητες πολύ πέρα

από εκείνες μιας συσκευής GAL. Τα GALs είναι ευέλικτα για το μέγεθος τους λόγω των μεγάλων διατάξεων προγραμματίσημων AND που καθορίζουν τις λογικές συνδέσεις μεταξύ των εισόδων και των εξόδων. Παρόλα αυτά, αυτό καθιστά την αρχιτεκτονική δαπανηρή. Για κάθε macrocell που προστίθεται, και οι δύο διαστάσεις της διάταξης αυξάνονται επίσης. Οι προμηθευτές CPLD επιδιώκουν να παρέχουν μια πιο γραμμική κλίμακα των πόρων συνδεσιμότητας στα macrocells εφαρμόζοντας μια αρχιτεκτονική με πολλαπλά GAL καθορισμένου μεγέθους που διασυνδέονται μέσω μιας κεντρικής διάταξης διακοπτών όπως φαίνεται στο Σχήμα 25. Όπως το GAL, τα CPLDs συνήθως κατασκευάζονται με την διαμόρφωση αποθήκευσης E-EPROM, που καθιστά τη λειτουργία τους αμετάβλητη. Μετά από τον προγραμματισμό, ένα CPLD θα διατηρήσει τη διαμόρφωσή του και θα είναι έτοιμο για λειτουργία όταν δοθεί ρεύμα στο σύστημα.

Κάθε μεμονωμένο block λογικής είναι παρόμοιο με αυτό του GAL και περιέχει τη δική του προγραμματισημη AND/OR διάταξη και τα δικά του macrocells. Αυτή η προσέγγιση είναι εξελικτική, επειδή οι προγραμματίσημες AND/OR διατάξεις παραμένουν σταθερές στο μέγεθος και αρκετά μικρές για να κατασκευαστούν οικονομικά. Όσο περισσότερα macrocells ενσωματώνονται επάνω στο ίδιο chip, περισσότερα block λογικής τοποθετούνται επάνω στο chip αντί να αυξάνεται το μέγεθος των ίδιων των block και να γεμίζουν τις διατάξειςAND/OR.Το CPLD αυτού του τύπου κατασκευάζεται από επιχειρήσεις συμπεριλαμβανομένου της Altera, της Cypress, της Lattice και της Xilinx.



Σχήμα 25 Χαρακτηριστική αρχιτεκτονική CPLD

Τα I/O pin γενικής χρήσης είναι αμφίδρομα και μπορούν να διαμορφωθούν ως είσοδοι, ως έξοδοι ή και τα δύο. Αυτό είναι σε αντίθεση με τα καθορισμένα pin ρεύματος και δοκιμής που είναι απαραίτητα για τη λειτουργία. Υπάρχουν τόσα πολλά πιθανά I/O pin χρήστη όσα είναι και τα macrocells, αν και κάποιο CPLD μπορεί να τοποθετηθεί σε συσκευασίες που δεν έχουν αρκετά φυσικά pin για να συνδεθεί με όλα τα σημεία I/O του chip.

Επειδή το μέγεθος κάθε λογικού block της διάταξης AND είναι καθορισμένο, το block έχει έναν σταθερό αριθμό πιθανών εισόδων. Λογισμικό το οποίο παρέχεται από τον προμηθευτή πρέπει να καθορίσει ποιες λογικές λειτουργίες τοποθετούνται σε ποια block και πώς η διάταξη των διακοπτών συνδέει την ανατροφοδότηση και τα pin εισόδου στο κατάλληλο block. Η διάταξη των διακοπτών δεν αυξάνεται γραμμικά όσο περισσότερα block λογικής προστίθενται. Κάθε CPLD οικογένεια παρέχει ένα διαφορετικό αριθμό από εισόδους σε κάθε λογικό block.

Τα block λογικής έχουν πολλά κοινά χαρακτηριστικά με ένα GAL, όπως φαίνεται στο

Σχήμα 26. Περιορίζοντας τον αριθμό των χαρακτηριστικών του προϊόντος σε κάθε λογικό block μειώνεται η πολυπλοκότητα και το κόστος της συσκευής. Μερικοί προμηθευτές παρέχουν μόνο πέντε χαρακτηριστικά ανά macrocell. Για να ισορροπηθεί αυτός ο περιορισμός, ο οποίος θα μπορούσε να έρθει σε σύγκρουση με τη χρησιμότητα ενός CPLD, η διανομή πόρων του προϊόντος επιτρέπει σε ένα macrocell να δανειστεί χαρακτηριστικά από γειτονικά macrocells. Αυτός ο δανεισμός έρχεται συνήθως σε μια μικρή καθυστέρησης διάδοσης αλλά παρέχει την απαραίτητη ευελιξία στο χειρισμό σύνθετων εκφράσεων Boole με πολλά χαρακτηριστικά του προϊόντος. Ένας macrocell block περιέχει flip-flop και διάφορες επιλογές διαμόρφωσης όπως η πολικότητα και ο έλεγχος ρολογιών. Ως αποτέλεσμα της υψηλότερης πυκνότητας λογικής τους, τα CPLDs περιέχουν πολλαπλά ρολόγια από τα οποία τα macrocells μπορούν να επιλέξουν, όπως και τη δυνατότητα να παράγουν ρολόγια και από την ίδια την λογική διάταξη.

Η Xilinx είναι προμηθευτής των προϊόντων CPLD και κατασκευάζει μια οικογένεια γνωστή ως XC9500. Λογικά block, ή block λειτουργίας στην ορολογία Xilinx, κάθε ένα περιέχει 18 macrocells, τα αποτελέσματα των οποίων ανατροφοδοτούν την διάταξη των διακοπτών και οδηγούν τα I/O pin’s επίσης. Τα XC9500 CPLDs περιέχουν πολλαπλάσια των 18 macrocells στις πυκνότητες από 36 έως 288 macrocells. Κάθε block λειτουργίας παίρνει 54 όρους εισόδου από τη διάταξη διακοπτών. Αυτοί οι όροι εισόδων μπορεί να είναι οποιοσδήποτε συνδυασμός I/O εισόδων pin’s και ανατροφοδότηση από άλλα block λειτουργίας macrocells.

Όπως ένα GAL, ο συγχρονισμός CPLD είναι πολύ προβλέψιμος λόγω της αποφασιστικής φύσης των λογικών block AND διάταξης και των χαρακτηριστικών εισόδων διάταξης των διακοπτών. Το Xilinx XC9536xv-3 εμφανίζει μια μέγιστη καθυστέρηση διάδοσης pin – to – pin3.5 ns και χρόνο tco = 2.5 ns. * Η εσωτερική λογική μπορεί να τρέξει με 277 MHZ με τις καθυστερήσεις από την ανατροφοδότηση συμπεριλαμβανόμενες, αν και οι σύνθετες εκφράσεις Boole πιθανών μειώνουν τη μέγιστη συχνότητα λόγω διαμοιρασμού των χαρακτηριστικών του προϊόντος και των καθυστερήσεων διάδοσης της ανατροφοδότησης μέσω πολλαπλών macrocell.

Το λογισμικό του CPLD συνήθως παρέχεται από τον κατασκευαστή, διότι οι αρχιτεκτονικές του είναι ιδιόκτητες και δεν αποκαλύπτονται με επαρκείς λεπτομέρειες για το σχεδιασμό των απαραίτητων αλγορίθμων από τρίτους. Αυτά τα εργαλεία δέχονται μια netlist από ένα σχηματικό εργαλείο ή μια σύνθετη HDL και αυτόματα χωρίζουν τη λογική μεταξύ των macrocells και των λογικών block.

Σχήμα 26 Block λογικής CPLD

Η διαδικασία εφαρμογής είναι πιο πολύπλοκη απ'ό, τι για ένα GAL. Δεν μπορεί κάθε όρος μέσα στο CPLD να τροφοδοτηθεί σε κάθε macrocell λόγω της κωδικοποίησης της δομής της AND διάταξης. Η διανομή των χαρακτηριστικών του προϊόντος τοποθετεί περιορισμούς στα γειτονικά macrocells όταν οι Boolean εκφράσεις υπερβαίνουν τον αριθμό των χαρακτηριστικών του προϊόντος που συνδέονται άμεσα με κάθε macrocell. Το λογισμικό

μειώνει αρχικά τη netlist σε ένα σύνολο εκφράσεων Boole σε μορφή που μπορεί να χαρτογραφηθεί στο CPLD και μετά τροποποιεί τις λειτουργίες των macrocells για να παρέχει σε κάθε ένα τα απαιτούμενα χαρακτηριστικά του προϊόντος. Η επιθυμητή συχνότητα λειτουργίας επηρεάζει την τοποθέτηση της λογικής λόγω καθυστέρησης διάδοσης των χαρακτηριστικών του προϊόντος στα macrocells. Αυτές οι ανταλλαγές εμφανίζονται σε τόσο χαμηλό επίπεδο που καθιστά συχνά αδύνατη την ανθρώπινη επέμβαση.

Τα CPLD’s έχουν έρθει για να προσφέρουν πλεονεκτήματα ευελιξίας πέρα από την απλή εφαρμογή λογικής. Καθώς τα ψηφιακά συστήματα γίνονται πιο σύνθετα, οι τάσεις στα ολοκληρωμένα κυκλώματα (IC) λογικής αρχίζουν να πολλαπλασιάζονται. Παλιότερα τα περισσότερα συστήματα έτρεχαν με 5V. Μετά ακολούθησαν τα συστήματα με 3.3V, και τώρα μπορούν εύκολα να βρεθούν συστήματα που λειτουργούν σε πολλαπλάσιες τάσεις όπως 3.3V, 2.5V, 1.8V, και 1.5V. Τα CPLD’s σχεδιάζονται σε περιβάλλοντα με διαφορετικές τάσεις λειτουργίας για τους σκοπούς των interfaces που κάνουν τις μετατροπές αλλά και την διαχείριση των καναλιών διαμεταγωγής δεδομένων. Για να ικανοποιήσει αυτές τις ανάγκες, πολλά CPLD’s υποστηρίζουν περισσότερα από ένα I/O τάσης επάνω το ίδιο chip συγχρόνως. Τα I/O pin διαχωρίζονται σε γκρουπ, και κάθε γκρουπ μπορεί να επιλεχτεί ανεξάρτητα για διαφορετική I/O τάση.

Τα περισσότερα CPLD’s είναι μικρής χωρητικότητας λογικού προγραμματισμού κυρίως της επιθυμίας για υψηλές ταχύτητες λειτουργίας με συγκεκριμένους χρονισμούς σε λογικό κόστος.

2.3. FPGAs

Τα CPLD’s ταιριάζουν στις εφαρμογές που περιλαμβάνουν έλεγχο λογικής, τις μηχανές βασικών λειτουργιών, και μικρές ομάδες καταχωρητών read/write. Αυτές οι εφαρμογές ελέγχου συνήθως χρησιμοποιούν ένα μικρό αριθμό από Flop’s. Όταν όμως μια διεργασία απαιτήσει πολλές εκατοντάδες ή χιλιάδες Flop’s, τότε τα CPLD’s γίνονται γρήγορα αδύνατο να χρησιμοποιηθούν. Οι σύνθετες εφαρμογές που χειρίζονται και αναλύουν μεγάλες ποσότητες δεδομένων απαιτούν συχνά μεγάλες ποσότητες από Flop’s για να χρησιμεύσουν ως καταχωρητές διαμεταγωγής δεδομένων, καταχωρητές προσωρινής αποθήκευσης δεδομένων και μετρητές. Τα ενσωματωμένα Block μνήμης είναι ζωτικής σημασίας για εφαρμογές που απαιτούν πολλαπλά FiFo και buffers αποθήκευσης δεδομένων. Τα FPGA (Field Programmable gate arrays) ταξινομούν άμεσα αυτές τις εφαρμογές δεδομένων.

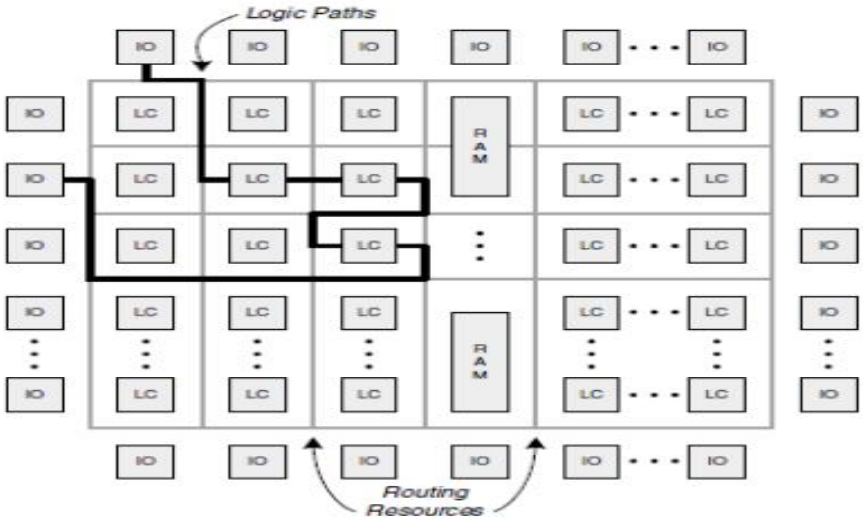
Τα FPGA’s είναι διαθέσιμα σε πολλές παραλλαγές με ποικιλία από σετ χαρακτηριστικών. Εντούτοις, το κύριο χαρακτηριστικό τους είναι η ιδιαίτερα λεπτομερής αρχιτεκτονική τους που αποτελείται από μια σειρά μικρών κελιών λογικής, κάθε ένα από αυτά αποτελείται από ένα Flop, ένα μικρό πίνακα lookup (LUT), και μια επιπλέον λογική υποστήριξης για την επιτάχυνση των συνηθισμένων λειτουργιών όπως, πολύπλεξη και αποθήκευσης αριθμών για τις διευθύνσεις και τους μετρητές. Οι Boolean εκφράσεις αξιολογούνται από τα LUT, τα οποία εφαρμόζονται συνήθως ως μικρές διατάξεις SRAM. Οποιαδήποτε συνάρτηση τεσσάρων μεταβλητών, παραδείγματος χάριν, μπορεί να εφαρμοστεί σε ένα 16 × 1 SRAM όταν οι τέσσερις μεταβλητές λειτουργούν ως δείκτες στη μνήμη. Δεν υπάρχουν διατάξεις AND/OR όπως σε ένα CPLD ή GAL. Όλες οι Boolean λειτουργίες εφαρμόζονται μέσα στα κελιά λογικής. Τα κελιά ταξινομούνται σε ένα πλέγμα δρομολόγησης δεδομένων που μπορούν να κάνουν τις συνδέσεις μεταξύ αυθαίρετων κελιών για να δημιουργήσουν μονοπάτια λογικής όπως στο σχήμα τις πορείες λογικής όπως φαίνεται στο Σχήμα 27. Ανάλογα με τον τύπο FPGA,δομές ειδικής λειτουργίας τοποθετούνται στη

διάταξη. Συχνότερα, αυτά είναι διαμορφώσιμα BlockRAM και στοιχεία διανομής ρολογιών. Γύρω από την περιφέρεια του τσιπ υπάρχουν κελιά I/O, τα οποία περιέχουν συνήθως ένα ή περισσότερα Flop’s για να επιτρέψουν την υψηλής απόδοσης σύγχρονων interfaces.

Η τοποθέτηση Flop μέσα στα I/O κελιά βελτιώνει τα χαρακτηριστικά χρονισμού ελαχιστοποιώντας την απόσταση και ως εκ τούτου και την καθυστέρηση μεταξύ κάθε Flopκαι του αντίστοιχουpin. Αντίθετα με τα CPLD’s, τα περισσότερα FPGA’s είναι βασισμένα στην τεχνολογία SRAM, που καθιστά ευκολότερο τον χειρισμό τους. Ένα τυπικό FPGA πρέπει να αναπρογραμματιστεί κάθε φορά που θέτουμε σε λειτουργία ένα σύστημα. Στους κυριότερους κατασκευαστές FPGA’s περιλαμβάνονται οι Actel, Altera, Atmel, Lattice, QuickLogic, και Xilinx.

Οι πολύ υψηλές πυκνότητες λογικής επιτυγχάνονται κλιμακώνοντας το μέγεθος του δισδιάστατου πίνακα των κελιών λογικής. Ο αρχικός περιοριστικός παράγοντας στην απόδοση του FPGA γίνεται ρουτίνα λόγω της μη αποφασιστικής φύσης των πολλαπλών διαδρομών πλέγματος που διασυνδέουν το σύστημα. Οι διαδρομές μεταξύ των κελιών λογικής μπορούν να ακολουθήσουν τις πολλαπλές διαδρομές, μερικές από τις οποίες μπορούν να δώσουν πανομοιότυπες καθυστερήσεις διάδοσης. Εντούτοις, οι πόροι δρομολόγησης είναι πεπερασμένοι, και προκύπτουν γρήγορα συγκρούσεις μεταξύ των ανταγωνιστικών διαδρομών για τα ίδια κανάλια δρομολόγησης. Όπως με τα CPLD’s, οι κατασκευαστές των FPGA παρέχουν ιδιόκτητα εργαλεία λογισμικού για την μετατροπή μιας netlist σε μια τελική εικόνα προγραμματισμού.

Ανάλογα με την πολυπλοκότητα του σχεδίου (π.χ., ταχύτητα και πυκνότητα), η καθοδήγηση ενός FPGA μπορεί να πάρει μερικά λεπτά ή πολλές ώρες. Αντίθετα από τα CPLD με συγκεκριμένους πόρους διασύνδεσης, ο συγχρονισμός ενός FPGAμπορεί να ποικίλει εντυπωσιακά, ανάλογα με την ποιότητα της τοποθέτησης λογικής. Τα μεγάλα, γρήγορα σχέδια απαιτούν επαναληπτικούς αλγορίθμους δρομολόγησης και τοποθέτησης.



Σχήμα 27Πίνακας λογικής FPGA

Η ανθρώπινη επέμβαση μπορεί να είναι κρίσιμη για την επιτυχή δρομολόγηση και το συγχρονισμό ενός σύνθετου σχεδίου FPGA. Floor planning είναι η διαδικασία από την οποία ένας μηχανικός χωρίζει χειροκίνητα τη λογική σε ομάδες και κατόπιν τοποθετεί αυτές τις ομάδες σε τμήματα ενός πίνακα με κελιά λογικής. Ο χειροκίνητος εντοπισμός μεγάλων τμημάτων λογικής δεσμεύει το λογισμικό δρομολόγησης στη βελτιστοποίηση της

τοποθέτησης του κώδικα λογικής και μειώνει τον αριθμό αλλαγών που χρειάζονται ώστε να έχει ένα επιτυχές αποτέλεσμα.

Η αρχιτεκτονική κελιών λογικής κάθε κατασκευαστή διαφέρει κάπως, αλλά κυρίως στο πώς η υποστήριξη λειτουργεί όπως στη πολύπλεξη και την εφαρμογή αριθμητικών όρων. Για το μεγαλύτερο μέρος, οι μηχανικοί δεν χρειάζεται να εξετάζουν τις μικρές λεπτομέρειες κάθε δομής κελιών λογικής, επειδή η μετατροπή της λογικής σε λογικά κελιά γίνεται από έναν συνδυασμό του εργαλείου σύνθεσης HDL και του ιδιόκτητου λογισμικού χαρτογράφησης του κατασκευαστή. Σε ακραίες καταστάσεις, όπου μια πολύ συγκεκριμένη εφαρμογή λογικής είναι απαραίτητη να συμπιέσει την μέγιστη απόδοση από ένα συγκεκριμένο FPGA, η βελτιστοποίηση της λογικής και της αρχιτεκτονικής για μια δεδομένη δομή κελιών λογικής μπορεί να έχει οφέλη. Η συμμετοχή σε αυτό το επίπεδο της τεχνολογικά-συγκεκριμένης βελτιστοποίησης, εντούτοις, μπορεί να είναι πολύ απρόβλεπτη και να οδηγήσει σε ένα σενάριο όπως το σπίτι από τραπουλόχαρτα στο οποίο όλα είναι τέλεια και ισορροπημένα για μια στιγμή, και έπειτα ένα νέο χαρακτηριστικό γνώρισμα προστίθεται που ανατρέπει ολόκληρο το σχέδιο. Εάν ένα σχέδιο είναι τόσο επιθετικό ώστε να απαιτήσει ακριβής βελτιστοποίηση και γρηγορότερες συσκευές δεν μπορούν να χρησιμοποιηθούν, μπορεί να είναι προτιμότερο να τροποποιηθεί η αρχιτεκτονική για να επιτραπούν περισσότερες αφαιρετικές μεθόδους σχεδίασης.

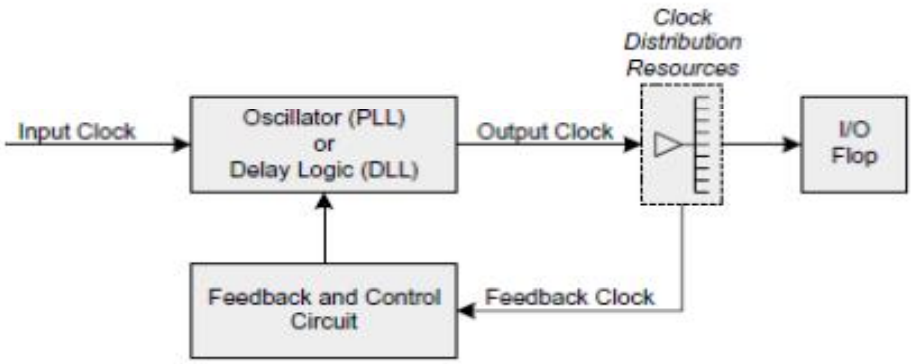
Παρά τα προηγούμενα σχόλια, υπάρχουν υψηλού επιπέδου διαφορές χαρακτηριστικών γνωρισμάτων μεταξύ των FPGAs που πρέπει να αξιολογηθούν πριν επιλέγει μια συγκεκριμένη συσκευή. Φυσικά, είναι απαραίτητο να επιλεχτεί ένα FPGA που έχει ικανοποιητική λογική και τα I/O pinγια να ικανοποιήσει τις ανάγκες της προοριζόμενης εφαρμογής. Αλλά δεν είναι όλα τα FPGAs δημιουργημένα ίσα, παρά το ότι έχουν παρόμοιες ποσότητες λογικής. Ενώ τα οφέλη από μια δομή λογικής σε μια άλλη μπορούν να συζητηθούν, η παρουσία ή η απουσία κρίσιμων πόρων σχεδίασης μπορεί να καταστήσει την εφαρμογή ενός συγκεκριμένου σχεδίου πιθανή ή αδύνατη. Αυτοί οι πόροι είναι στοιχεία διανομής χρονισμού, ενσωματωμένη μνήμη, ενσωματωμένοι πυρήνες τρίτων, και πολυχρηστικών I/O κελιών.

Η διανομή του χρονισμού σε ένα σύστημα σύγχρονης μετάδοσης πρέπει να γίνεται με ελάχιστες κινήσεις για να επιτύχει χρονισμούς σε αποδεκτά επίπεδα. Κάθε κελί λογικής μέσα σε ένα FPGA χρησιμοποιεί έναFlopτο οποίο απαιτεί δικό του χρονισμό, συνεπώς έναFPGAπρέπει να παρέχει τουλάχιστον ένα γενικό σήμα χρονισμού το οποίο να διανέμεται σε κάθε κελί λογικής με τις λιγότερες δυνατές κινήσεις μέσα στη συσκευή. Ένα ρολόι (χρονισμός) είναι ανεπαρκές για τα περισσότερα μεγάλα ψηφιακά συστήματα λόγω του πολλαπλασιασμού των διαφορετικών διεπαφών, των μικροεπεξεργαστών, και των περιφερειακών μονάδων. Συχνά τα FPGAs παρέχουν από 4 έως 16 ρολόγια γενικού χρονισμού με σχετικά σύντομες διαδρομές διανομής. Τα περισσότερα FPGAs επιτρέπουν στους χρονισμούς να δρομολογηθούν χρησιμοποιώντας τρόπους γενικής δρομολόγησης που συνήθως μεταφέρονται μέσα από λογικά σήματα, ωστόσο αυτές οι δρομολογήσεις συνήθως είναι ανεπαρκείς για να επιτύχουν τις σύντομες διαδρομές διανομής του χρονισμού με αποτέλεσμα να μην επιτρέπουν υψηλές ταχύτητες λειτουργίας.

Ορισμένα FPGAs υποστηρίζουν ένα μεγάλο αριθμό των ρολογιών, αλλά με τον περιορισμό ότι όλα τα ρολόγια να μην χρησιμοποιηθούν ταυτόχρονα στο ίδιο τμήμα του τσιπ. Τέτοιου είδους περιορισμοί μειώνουν την πολυπλοκότητα της διανομής ρολογιών για τον πωλητή των FPGA, διότι, ενώ ολόκληρο το chip υποστηρίζει ένα μεγάλο αριθμό των ρολογιών στο σύνολο, ξεχωριστά τμήματα του chip υποστηρίζουν ένα μικρότερο αριθμό. Για παράδειγμα, ένα FPGA μπορεί να υποστηρίξει 16 γενικά ρολόγια με τον περιορισμό ότι κάθε

ένα τεταρτημόριο μπορεί να υποστηρίξει μόνο 8 ρολόγια. Αυτό σημαίνει ότι υπάρχουν 16 ρολόγια διαθέσιμα, και κάθε τεταρτημόριο μπορεί να επιλέξει τα μισά από αυτά για αυθαίρετη χρήση. Αντί να παρέχει 16 ρολόγια για κάθε κελί λογικής, μόνο 8 πρέπει να είναι συνδεδεμένα, απλουστεύοντας κατά συνέπεια το κύκλωμα του FPGA.

Τα περισσότερα FPGAs παρέχουν phase locked loops (PLLs) ή delay locked loops (DLL) που ενεργοποιούν την εκ προθέσεως στρέβλωση, η διάσπαση, και τον πολλαπλασιασμό των εισερχόμενων σημάτων των ρολογιών. ΤαPLLs είναι εν μέρει αναλογικά κυκλώματα, ενώ τα DLL είναι πλήρως ψηφιακά κυκλώματα. Έχουν σημαντική αλληλεπίδραση στις λειτουργίες που μπορούν να προσφέρουν σε ένα FPGA, αν και οι μηχανικοί μπορούν να διαφωνήσουν για τα πλεονεκτήματα του ενός έναντι του άλλου. Το θεμελιώδες πλεονέκτημα ενός PLL ή DLL μέσα σε ένα FPGA είναι η ικανότητά του να βελτιώνει τον χρονισμό των I/O, με την αποτελεσματική άρση της καθυστέρησης διάδοσης μεταξύ του pin του ρολογιού εισόδου και του pin εξόδου του σήματος, ικανότητα επίσης γνωστή και ως deskewing. Όπως φαίνεται στο Σχήμα 28, το PLL ή το DLL ευθυγραμμίζει το εισερχόμενο ρολόι με ένα ρολόι ανατροφοδότησης με την ίδια καθυστέρηση που παρατηρείται και στα I/O flops. Με αυτό τον τρόπο, μεταθέτει το εισερχόμενο ρολόι έτσι ώστε η αιτιώδης άκρη που παρατηρείται από τα I/O flops να εμφανίζεται σχεδόν στον ίδιο χρόνο όπως όταν εισέρχεται στο pin εισαγωγής ρολογιού του FPGA.



Σχήμα 28 PLL/DLL λειτουργία ρολογιών deskew μέσα σε FPGA

Τα πρόσθετα στοιχεία κυκλώματος επιτρέπουν σε κάποια PLLs και DLLs να εκπέμψουν ένα ρολόι που σχετίζεται με τη συχνότητα εισαγωγής από μια προγραμματίσημη αναλογία. Η δυνατότητα να πολλαπλασιαστούν και να διαιρεθούν τα ρολόγια είναι ένα όφελος σε μερικά σχέδια συστημάτων. Μια εξωτερική board-level διεπαφή μπορεί να τρέξει σε μια πιο αργή συχνότητα για να κάνει την εφαρμογή στο κύκλωμα ευκολότερη, αλλά μπορεί να επιδιωχτεί να τρέξει την εσωτερική λογική FPGA ως ένα γρήγορο πολλαπλάσιο εκείνου του ρολογιού για λόγους απόδοσης επεξεργασίας. Ανάλογα με την εφαρμογή, πολλαπλασιασμός ή διαίρεση μπορεί να βοηθήσει με αυτό το σχέδιο.

Τα block της RAM που ενσωματώνονται μέσα σε ένα λογικό πίνακα κελιών είναι κρίσιμο χαρακτηριστικό γνώρισμα για πολλές εφαρμογές.

Τα FIFOs και τα μικρά buffers βρίσκονται σε ποικιλία στις αρχιτεκτονικές επεξεργασίας. Χωρίς RAM στο chip, οι πολύτιμοι I/O πόροι και οι ποινές ταχύτητας θα σταματούσαν για να χρησιμοποιούν συσκευές με την εκτός chip μνήμη. Για να ταιριάξουν ένα ευρύ φάσμα εφαρμογών, οι RAM πρέπει να είναι ιδιαίτερα διαμορφώσιμες και ευέλικτες. Ένα τυπικό blockRAM FPGA είναι βασισμένο σε μια ορισμένη πυκνότητα και μπορεί να χρησιμοποιηθεί σε αυθαίρετες πλάτους/βάθους διαμορφώσεις όπως φαίνεται στο Σχήμα 29

που χρησιμοποιεί το παράδειγμα ενός block RAM 4 KB. Πολλές εφαρμογές, συμπεριλαμβανομένου των FIFOs, ωφελούνται από τη RAM διπλής εισόδου για να επιτρέψει την ταυτόχρονη ανάγνωση και το γράψιμο της μνήμης από διαφορετικά block λογικής. Μια μηχανή μπορεί να γράφει δεδομένα σε μια RAM, και άλλη μπορεί να διαβάζει κάποια δεδομένα συγχρόνως. Τα RAM blocks μπορούν να έχουν σύγχρονες ή ασύγχρονες διεπαφές και μπορούν να υποστηρίξουν ένα ή δύο ρολόγια στη σύγχρονη κατάσταση. Η υποστήριξη δύο ρολογιών στις σύγχρονες καταστάσεις διευκολύνει τα σχέδια FIFO για την κίνηση των στοιχείων μεταξύ διαφορετικών περιοχών ρολογιών.

Σχήμα 29 Διαμορφώσιμο FPGA RAM 4 KB block

Κάποια FPGAs επιτρέπουν επίσης στα λογικά κελιά LUTs να χρησιμοποιηθούν ως γενική RAM σε ορισμένες διαμορφώσεις. Ένα 16 × 1 τετραπλής εισαγωγής LUT μπορεί να λειτουργήσει ως μια 16 × 1 RAM εάν υποστηρίζεται από την αρχιτεκτονική του FPGA. Είναι πιο αποδοτικό να χρησιμοποιήσει τα RAM block για τις μεγάλες δομές μνήμης, επειδή το υλικό βελτιστοποιείται για να παρέχει μια ουσιαστική ποσότητα μνήμης σε μια μικρή περιοχή πυριτίου. Εντούτοις, η RAM βασισμένη σε LUT είναι σημαντική όταν ένα σχέδιο απαιτεί πολλές ρηχές δομές μνήμης (π.χ., ένα μικρό FIFO) και όλα τα μεγάλα RAM block χρησιμοποιούνται ήδη. Μαζί με τη λογική ελέγχου, 32 τετραπλής εισαγωγής LUTs μπορούν να χρησιμοποιηθούν για να κατασκευαστεί ένα 16 × 32 FIFO.

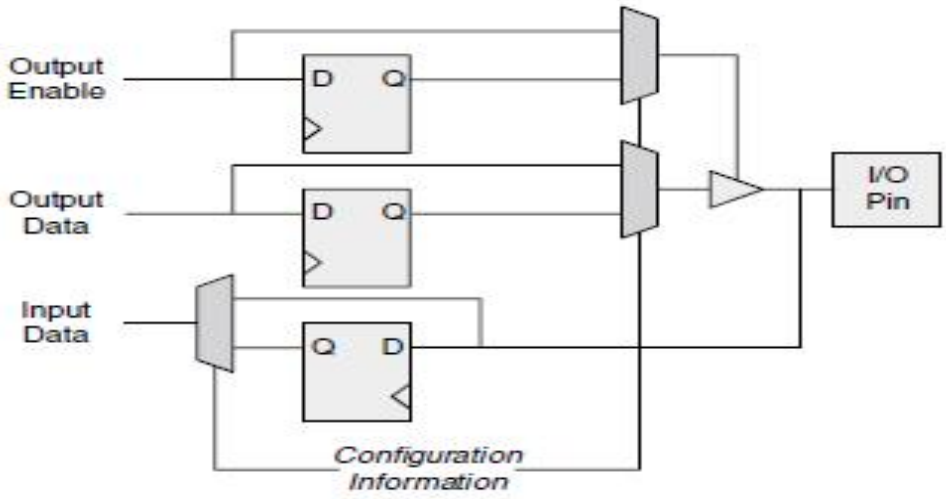
Η ενσωμάτωση των πυρήνων λογικής τρίτων είναι ένα χαρακτηριστικό γνώρισμα που μπορεί να είναι χρήσιμο για μερικά σχέδια, και μη χρήσιμο καθόλου για άλλα. Ένα μειονέκτημα των FPGAs είναι το υψηλότερο κόστος τους ανά πύλη από την τεχνολογία ASIC. Ο κύριος λόγος που οι μηχανικοί είναι πρόθυμοι να πληρώσουν αυτό το κόστος είναι για τη δυνατότητα να εφαρμόσουν διαμορφώσιμη λογική σε μια χαμηλού κινδύνου αναπτυξιακή διαδικασία. Μερικές εφαρμογές περιλαμβάνουν ένα μίγμα από διαμορφώσιμη και προσχεδιασμένη λογική που μπορεί να αγοραστεί από έναν τρίτο. Τα παραδείγματα αυτού περιλαμβάνουν την αγορά ενός σχεδίου μικροεπεξεργαστών ή ενός τυποποιημένου ελεγκτή διαύλων (π.χ. PCI) και την ενσωμάτωση του με τη διαμορφώσιμη λογική στο ίδιο τσιπ. Συνήθως, το κόστος ανά πύλη της λογικής τρίτων θα ήταν το ίδιο με αυτό της διαμορφώσιμης λογικής σας. Συν τοις άλλοις το κόστος είναι το τέλος αδείας που χρεώνεται από τρίτους. Μερικοί προμηθευτές FPGA έχουν αποφασίσει ότι υπάρχει ικανοποιητική ζήτηση για μερικούς τυποποιημένους πυρήνες λογικής για να προσφέρουν συγκεκριμένα FPGAs που ενσωματώνουν αυτούς τους πυρήνες στο chip σε μια σταθερή διαμόρφωση. Το όφελος κάνοντας αυτό είναι να πέσει το κόστος ανά πύλη του πυρήνα σχεδόν σε αυτό ενός διαμορφώσιμου ASIC, επειδή ο πυρήνας είναι συνδεμένος με καλώδιο και δεν απαιτεί τίποτα από τις γενικές διαμορφώσεις του FPGA.

Τα FPGAs με τους ενσωματωμένους πυρήνες λογικής μπορεί να κοστίσουν περισσότερο για να αντισταθμιστούν οι δαπάνες χορήγησης αδειών των πυρήνων, αλλά η ιδέα είναι ότι το γενικό κόστος στον πελάτη θα μειωθεί μέσω της αποδοτικότητας της εφαρμογή πυρήνων.

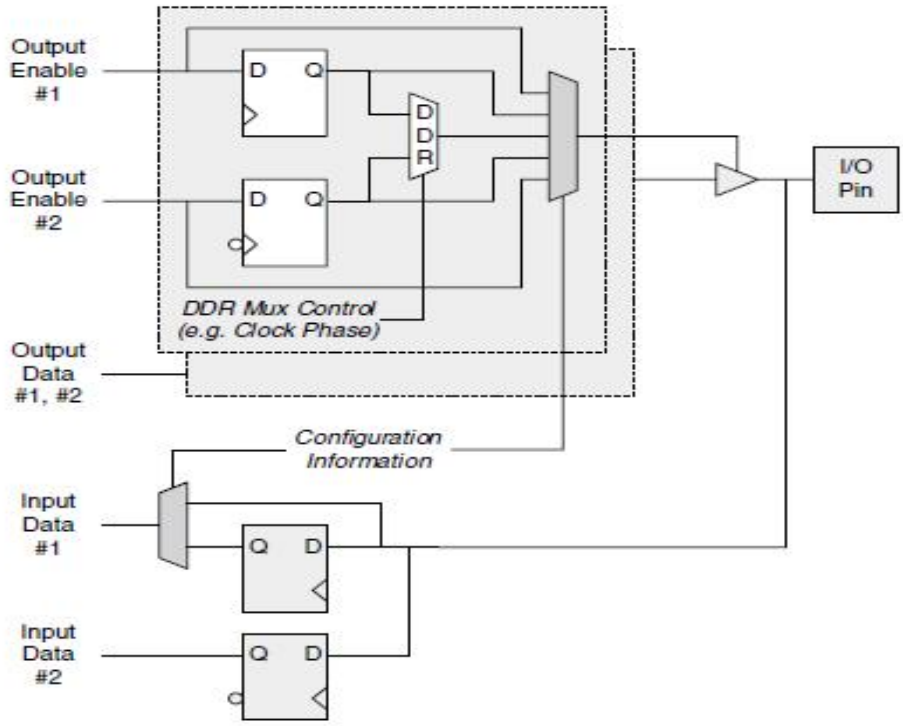
Η I/O αρχιτεκτονική κελιών μπορεί να έχει έναν σημαντικό αντίκτυπο στους τύπους επιπέδου πίνακα διεπαφών που το FPGA μπορεί να υποστηρίξει. Τα ζητήματα περιστρέφονται γύρω από δύο μεταβλητές: σύγχρονη λειτουργία και τα επίπεδα σε τάση / ρεύμα. ΤαFPGAs υποστηρίζουν I/O κελιά που μπορούν να διαμορφωθούν για εισαγωγή μόνο, έξοδο μόνο, ή αμφίδρομη λειτουργία με την ικανότητα παραγωγής buffer τριών καταστάσεων. Για να επιτύχουν τον καλύτερο I/O συγχρονισμό, τα flop και από τις τρεις καταστάσεις εισόδου, εξόδου, και ανατροφοδότησης πρέπει να περιληφθούν μέσα στο I/O κελί όπως φαίνεται στο Σχήμα 210. Η βελτίωση χρονισμού που παρατηρείται με τον εντοπισμό αυτών των τριών flop στα I/O κελιά είναι σημαντική. Η εναλλακτική λύση θα ήταν να χρησιμοποιηθούν λογικά κελιά flop και μονοπάτια από συστοιχία λογικών κελιών απευθείας στα pin του I/O κυκλώματος, αυξάνοντας τη καθυστέρηση διάδοσης του I/O. Κάθε μια από τις τρείς I/O λειτουργίες παρέχεται και ως καταχωρημένη και ως μη καταχωρημένη επιλογή χρησιμοποιώντας πολυπλέκτες για να παρέχει πλήρη ευελιξία στην εφαρμογή της λογικής.

Πιο προηγμένοι δίαυλοι επικοινωνίας τρέχουν με διπλή ταχύτητα ρυθμού δεδομένων, που απαιτούν πιο προηγμένη I/O δομή κελιών για την επίτευξη των απαραίτητων χρονοδιαγραμμάτων των προδιαγραφών τους.

Νεότερα FPGAs είναι διαθέσιμα με I/O κελιά τα οποία στηρίζουν τις διασυνδέσεις DDR με την ενσωμάτωση δύο σετ από flop, ένα για κάθε ρολόι, όπως φαίνεται στην εικόνα. 2,11. Όταν είναι ρυθμισμένη για λειτουργία DDR, κάθε μια από τις τρεις I/O λειτουργίες οδηγείται από ένα ζευγάρι flop, και ένας πολυπλέκτης επιλέγει το κατάλληλο flop εξόδου ανάλογα με τη φάση του ρολογιού. Μια διεπαφή DDR τρέχει εξωτερικά του FPGA και στις δύο άκρες του ρολογιού με ένα συγκεκριμένο πλάτος. Εσωτερικά, το interface τρέχει στο διπλάσιο του εξωτερικού πλάτους μόνο σε μία άκρη για την ίδια συχνότητα ρολογιού. Άρα το I/Oχρησιμεύει ως 2:1 πολυπλέκτης για τις εξόδους και ένα 1:2 αποπολυπλέκτης για τις εισόδους όταν λειτουργεί σε κατάσταση DDR.



Σχήμα 210 Δομή κελιών FPGA I/O



Σχήμα 211 Δομή κελιών FPGA DDR I/O

Εκτός από τη σύγχρονη λειτουργία, η συμβατότητα με διάφορες I/O τάσεις και ρεύματα στα πρότυπα των δίσκων είναι ένα κύριο χαρακτηριστικό για τα σύγχρονα και ευέλικτα FPGAs. Όπως τα CPLDs που υποστηρίζουν πολλαπλές I/O τράπεζες, κάθε μια από τις οποίες μπορεί να οδηγήσει ένα διαφορετικό επίπεδο τάσης, τα FPGAs χωρίζονται συνήθως σε I/O τράπεζες επίσης, για τον ίδιο σκοπό. Σε αντίθεση με τα CPLDs, πολλά FPGAs υποστηρίζουν μια ευρύτερη ποικιλία I/O προτύπων για μεγαλύτερη σχεδιαστική ευελιξία.

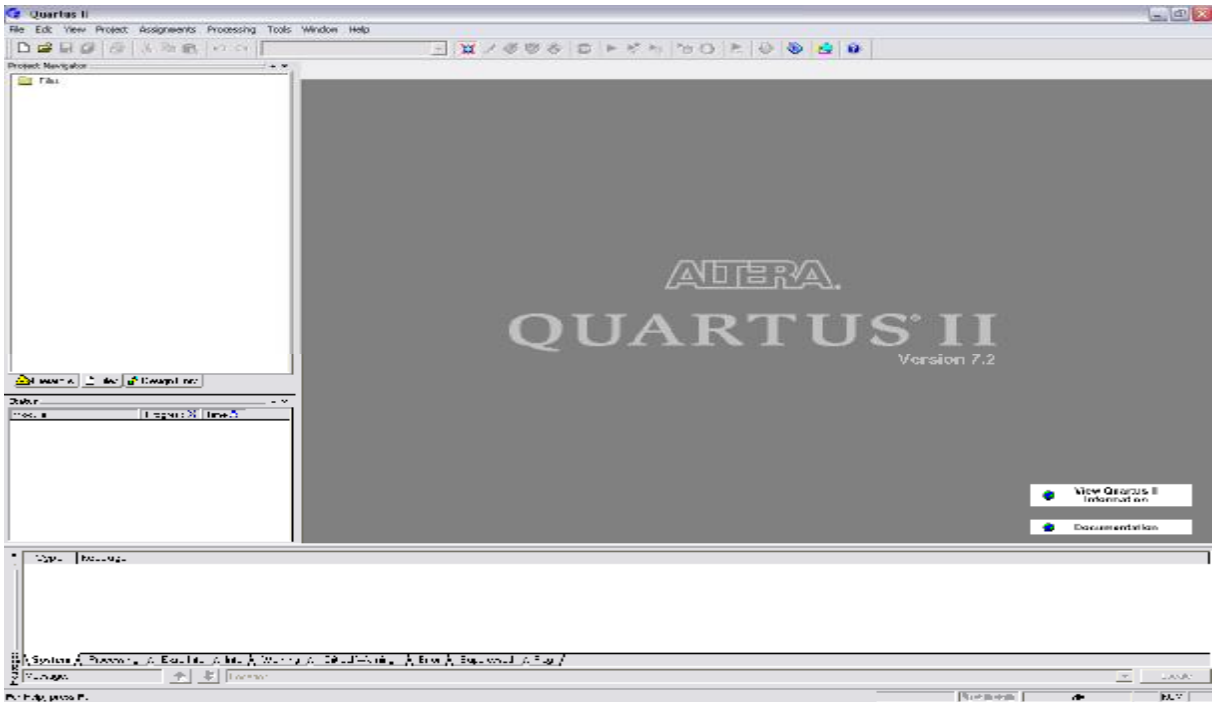
3. Σχεδίαση Κυκλωμάτων Με Το Λογισμικό QUARTUS II

3.1. Ξεκινώντας

Η σχεδίαση κυκλωμάτων σε VHDL μπορεί να γίνει σε έναν οποιονδήποτε text editor. Επειδή όμως είναι διαθέσιμο το εργαλείο Quartus II της Altera, πριν την εμβάθυνση στις βασικές έννοιες της VHDL, θα δειχτεί πως γίνεται η σχεδίαση και η εξομοίωση ενός κυκλώματος με τη βοήθεια του Quartus II.

Κάθε λογικό κύκλωμα, ή υποκύκλωμα, που σχεδιάζεται στο Quartus II εντάσσεται σε ένα project. Το πρόγραμμα δουλεύει πάνω σε ένα project κάθε φορά και κρατά όλες τις πληροφορίες γι' αυτό το project σε ένα φάκελο στο file system του υπολογιστή. Για να ξεκινήσει κανείς το σχεδιασμό ενός νέου λογικού κυκλώματος, το πρώτο βήμα είναι να δημιουργήσει ένα φάκελο (folder) όπου θα αποθηκευτούν όλα τα σχετικά αρχεία. Η τοποθεσία και το όνομα του φακέλου δεν έχει καμία σημασία, οπότε ο αναγνώστης μπορεί να χρησιμοποιήσει οποιοδήποτε φάκελο επιθυμεί (καλό είναι το path που θα βρίσκεται ο φάκελος αλλά και το όνομα αυτού να είναι γραμμένα στα αγγλικά). Για να αποθηκευτούν τα αρχεία που θα προκύψουν δημιουργήθηκε ο φάκελος C:\QuartusII\tutorial.

Μετά τη δημιουργία του φακέλου γίνεται εκκίνηση του Quartus II. Το παράθυρο που θα εμφανιστεί είναι παρόμοιο με αυτό στην Εικόνα 31και αποτελείται από άλλα παράθυρα, τα οποία παρέχουν πρόσβαση σε όλα τα χαρακτηριστικά (features) του Quartus II τα οποία ο χρήστης μπορεί να επιλέξει με το ποντίκι του υπολογιστή του.



Εικόνα 31 Κεντρικό παράθυρο του προγράμματος QuartusII

Οι περισσότερες από τις εντολές του Quartus II μπορούν να εκτελεστούν μέσω των menus που βρίσκονται στο επάνω μέρος του παραθύρου του Manager, ακριβώς κάτω από την μπάρα του τίτλου.(όπως δείχνει και η παρακάτω Εικόνα 32).

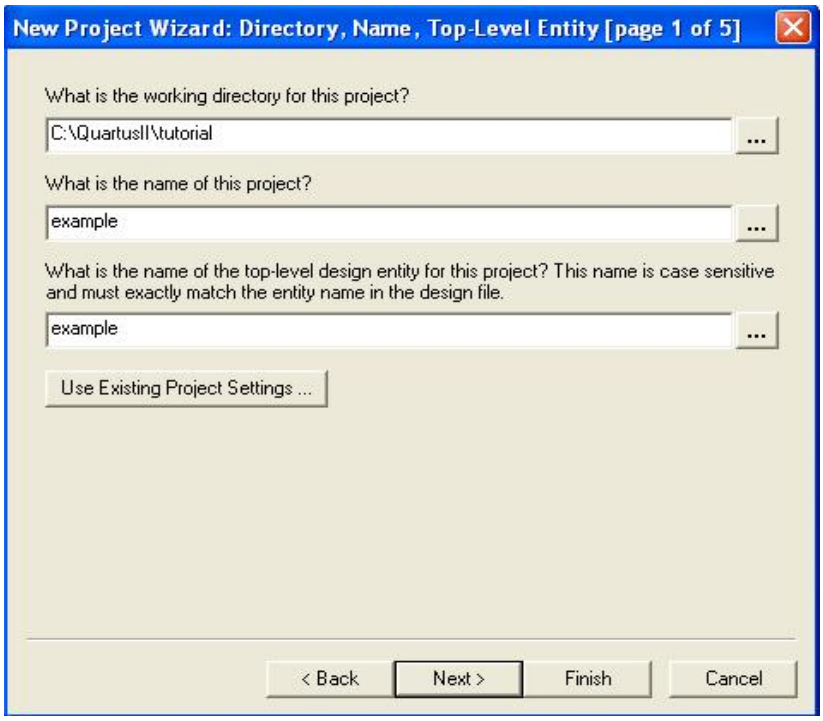
Εικόνα 32 Κεντρικό menu του προγράμματος Quartus II

Πολλές εντολές του Quartus II έχουν ένα αντίστοιχο εικονίδιο, το οποίο παρουσιάζεται πάνω στην γραμμή εργαλείων. Για να εμφανιστεί η λίστα των διαθέσιμων γραμμών εργαλείων επιλέγεται Tools | Customize | Toolbars. Μόλις ανοίξει μια γραμμή εργαλείων, μπορεί να μεταφερθεί με το ποντίκι όπως επίσης και τα εικονίδια μπορούν να μεταφερθούν από μια γραμμή εργαλείων σε μια άλλη. Για να παρουσιαστεί η εντολή του Quartus II, η οποία σχετίζεται με ένα εικονίδιο, τοποθετείται ο δείκτης του ποντικιού πάνω στο εικονίδιο και αμέσως εμφανίζεται ένα μήνυμα το οποίο περιγράφει το όνομα της εντολής.

3.2. Δημιουργία Νέου Project

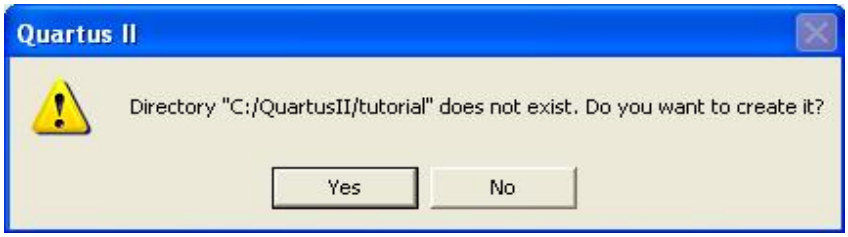
Σαν πρώτο βήμα πρέπει να προσδιοριστεί το όνομα του project. Το Quartus II παρέχει τη δυνατότητα ενός wizard. Επιλέγοντας File | New Project Wizard ανοίγει ένα παράθυρο το οποίο παρουσιάζει τις δυνατότητες που παρέχει ο wizard. Επιλέγοντας Next γίνεται μετάβαση στο παράθυρο που παρουσιάζεται στη Εικόνα 32.

Είναι απαραίτητο να προσδιοριστεί η θέση του φακέλου όπου θα αποθηκευτούν τα αρχεία που θα δημιουργηθούν για το project. Για αυτό το παράδειγμα, ο φάκελος που θα χρησιμοποιηθεί είναι ο c:\Quartus II\tutorial. Το project πρέπει να έχει ένα όνομα, το οποίο μπορεί να είναι ίδιο με το όνομα του directory. Γίνεται επιλογή του ονόματος example. Το Quartus II αυτόματα προτείνει το ίδιο όνομα (example) και ως όνομα του top-level κώδικα στην οντότητα του project. Ο χρήστης μπορεί να επιλέξει αν θα διατηρήσει το όνομα ή να επιλέξει ένα άλλο. Επιλέγοντας Next (εφόσον δεν έχει δημιουργηθεί ακόμη το directoryc:\Quartus II\tutorial), το Quartus II εμφανίζει το pop-up κουτί, όπως φαίνεται στο Εικόνα 33, με την ερώτηση αν πρέπει να δημιουργήσει το επιθυμητό directory. Επιλέγοντας Yes, εμφανίζεται το παράθυρο στη Εικόνα 33



Εικόνα 33 Παράθυρο για την δημιουργία καινούργιου project

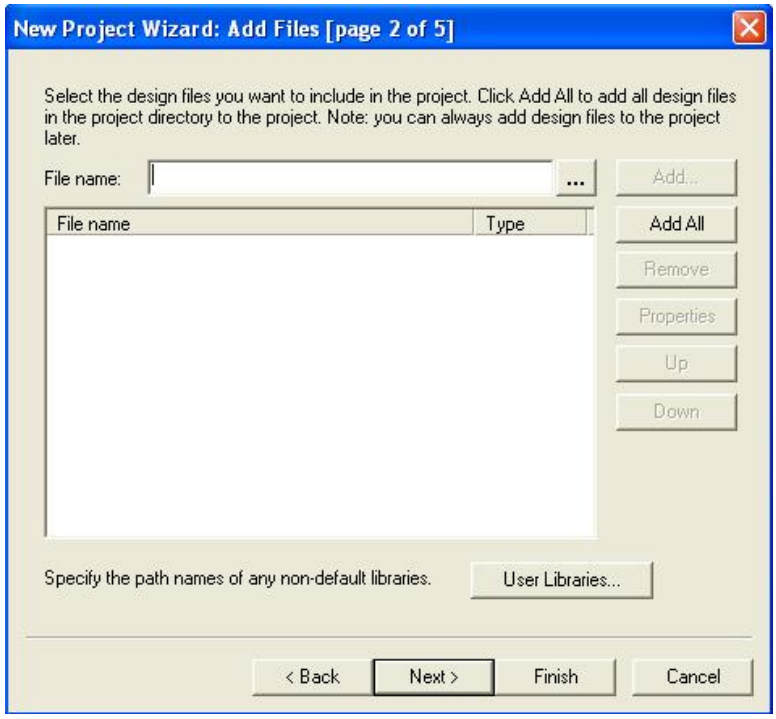
Σε αυτό το παράθυρο ο χρήστης μπορεί να προσδιορίσει ποια αρχείο επιθυμεί να συμπεριληφθούν στο project. Δεν υπάρχουν αρχεία, γι' αυτό η επιλογή είναι Next.



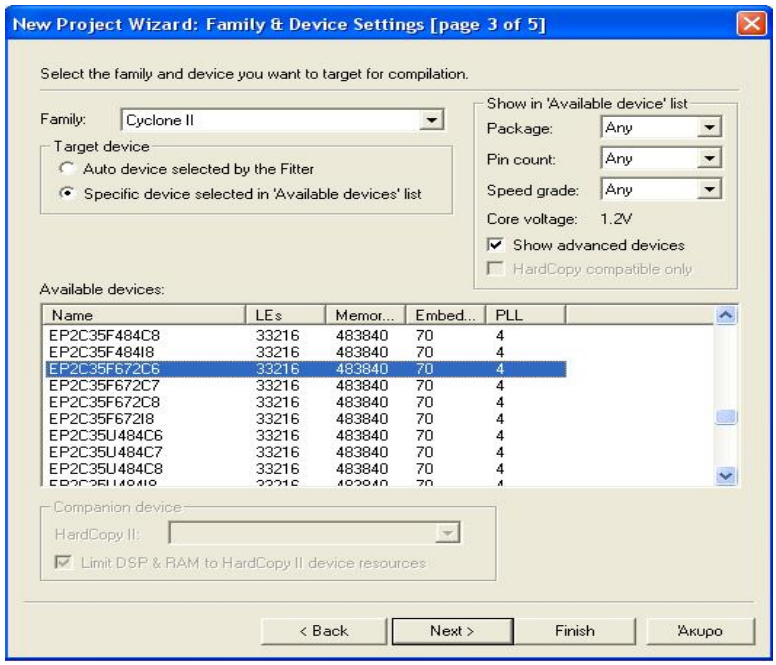
Εικόνα 34 Παράθυρο για επικύρωση της δημιουργίας καινούργιου project

Εμφανίζεται το παράθυρο του Εικόνα 35 το οποίο επιτρέπει στο χρήστη να καθορίσει τον τύπο του ολοκληρωμένου στο οποίο θα υλοποιηθεί το κύκλωμα που σχεδιάζει. Στη συγκεκριμένη περίπτωση επιλέγουμε την οικογένεια CycloneII και στο πεδίο **target device** επιλέγουμε **specificdeviceselectedin"available device"list**.

Τέλος στο πεδίο **Availabledevices** κάνουμε επιλογή του **EP2C35F672C6** και επιλέγουμε Next.

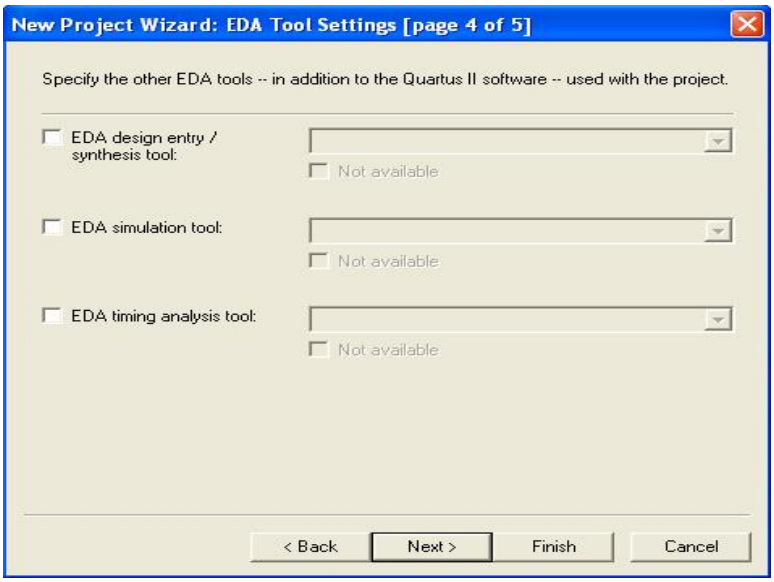


Εικόνα 35 Καθορισμός του τύπου του ολοκληρωμένου



Εικόνα 36 Επιλογή ολοκληρωμένου

Επιλέγοντας Next εμφανίζεται το παράθυρο της Εικόνα 37, στο οποίο προσδιορίζονται επιπλέον CAD εργαλεία σε περίπτωση που επιθυμούμε να τα χρησιμοποιήσουμε.

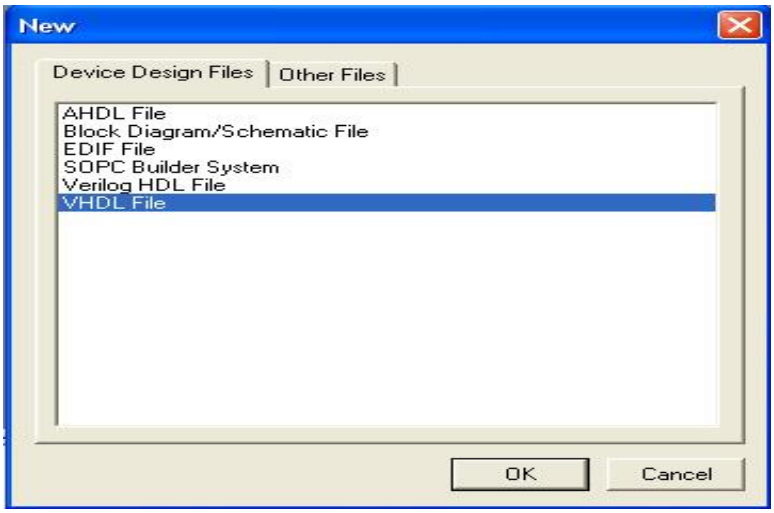


Εικόνα 37 Επιλογή εργαλείων

Πατώντας Finish, επανεμφανίζεται το κύριο παράθυρο του Quartus II της Εικόνα 31 με το example ως το νέο project.

3.3. Text Editor

Είναι ο βασικός κειμενογράφος ο οποίος χρησιμοποιείται για την πληκτρολόγηση της περιγραφής του κυκλώματος (προγράμματος) σε VHDL. Το πλεονέκτημά του σε σχέση με έναν απλό κειμενογράφο είναι το γεγονός ότι εμφανίζει τις δεσμευμένες λέξεις της VHDL με διαφορετικό χρώμα. Με αυτόν τον τρόπο γίνεται πιο εύκολη η ανάγνωση του προγράμματος αλλά και η εύρεση και αντιμετώπιση λαθών. Το παράθυρο του Text Editor της Εικόνα 38 ανοίγει επιλέγοντας File | New και στη συνέχεια επιλέγοντας VHDL File και OK.

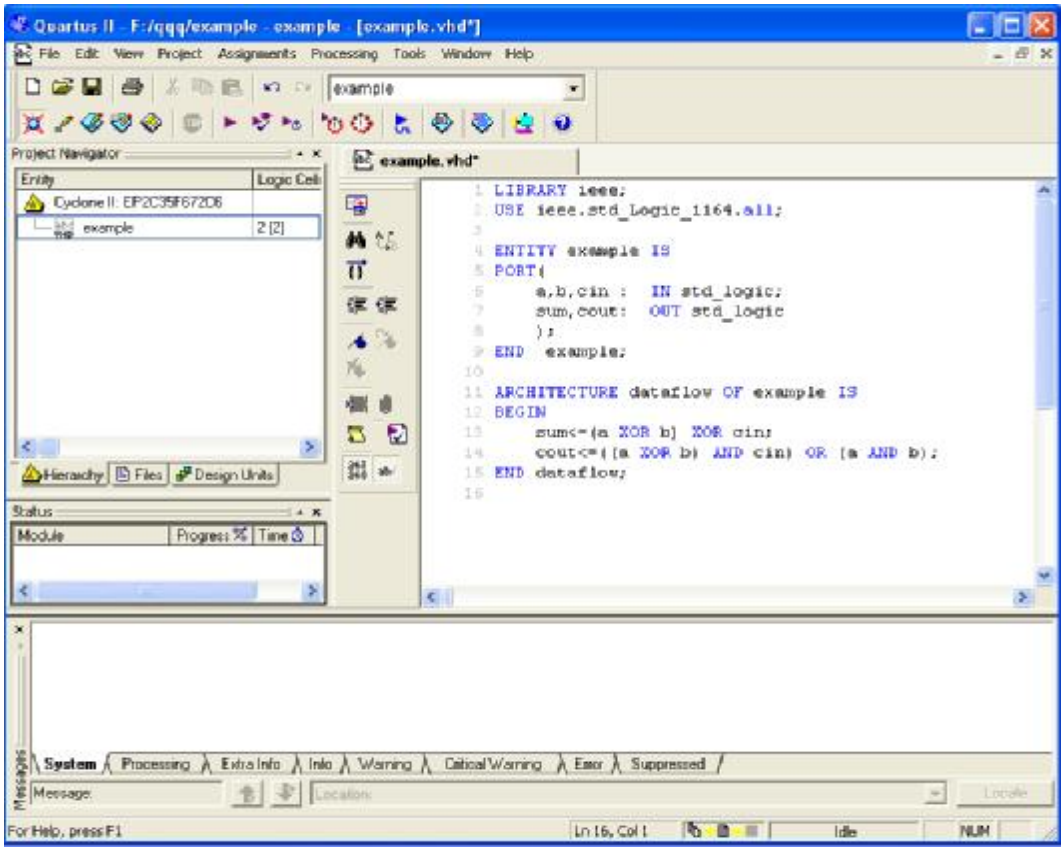


Εικόνα 38 Παράθυρο του TextEditor

Ο κώδικας σε VHDL για το παράδειγμα example.vhd φαίνεται στη Εικόνα 39. Περιγράφει έναν πλήρη αθροιστή. Σημαντική παρατήρηση είναι ότι το όνομα της entity είναι ***example***. Είναι απαραίτητο το όνομα του project, το όνομα του αρχείου και το όνομα της entity να είναι ακριβώς τα ίδια. Να τονίσουμε εδώ ότι η γλώσσα VHDL έχει αρκετές δηλώσεις και μπορεί να είναι δύσκολο για κάποιον (και ιδιαιτέρως έναν αρχάριο) να θυμάται τον τρόπο με τον οποίο η κάθε δήλωση συντάσσεται. Για να βοηθήσει ο TextEditor έχει έναν αριθμό από VHDL templates. Τα templates παρέχουν παραδείγματα δομών της VHDL, όπως δήλωση entity, architecturebody, δήλωση for και πολλά άλλα. Επιλέγοντας Edit | InsertTemplate | VHDL εμφανίζεται μια λίστα με τα templates υπάρχουν από την οποία μπορεί κανείς να επιλέξει κάποιο για να το εισάγει στον κώδικά του.

Συνθέτοντας Ένα Κύκλωμα Από Κώδικα Σε VHDL

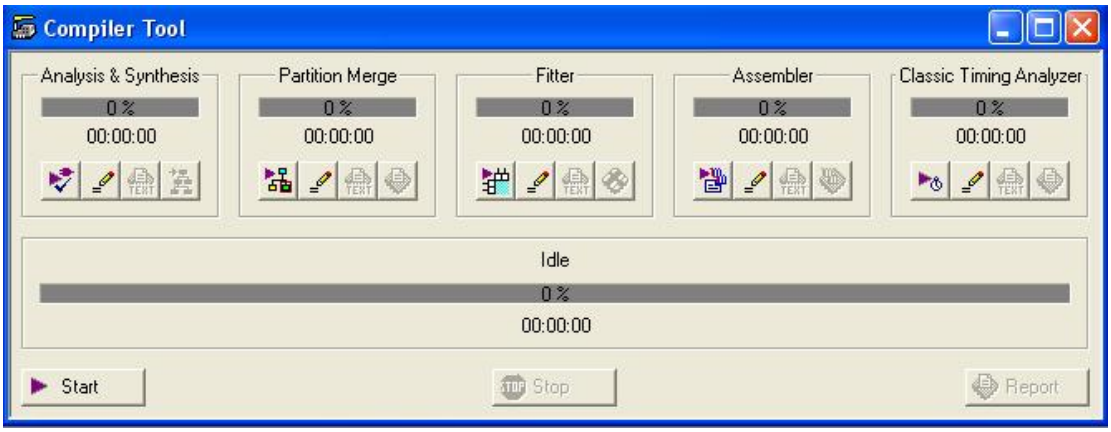
Το εργαλείο σύνθεσης μεταφράζει τον κώδικα σε λογικές εκφράσεις. Στη συνέχεια κάθε λογική έκφραση υλοποιείται στα λογικά στοιχεία που παρέχονται από το ολοκληρωμένο.



Εικόνα 39 Βασικό παράθυρο του Quartus II με κώδικα VHDL

Τα CAD εργαλεία που παρέχονται από το Quartus II χωρίζονται σε έναν αριθμό μονάδων. Με την επιλογή Processing | CompilerTool ανοίγει το παράθυρο της Εικόνα 310, στο οποίο παρουσιάζονται πέντε από τις κύριες μονάδες. Η μονάδα ανάλυσης και σύνθεσης (Analysis&Synthesis) χρησιμοποιείται στο στάδιο σύνθεσης του Quartus II. Παράγει ένα κύκλωμα λογικών στοιχείων, όπου κάθε στοιχείο υλοποιείται άμεσα στο ολοκληρωμένο.

Η μονάδα εφαρμογής (Fitter) καθορίζει την ακριβή θέση στο ολοκληρωμένο, στην οποία υλοποιούνται τα στοιχεία που παράγονται από το στάδιο της σύνθεσης.



Εικόνα 310Εργαλεία compiler

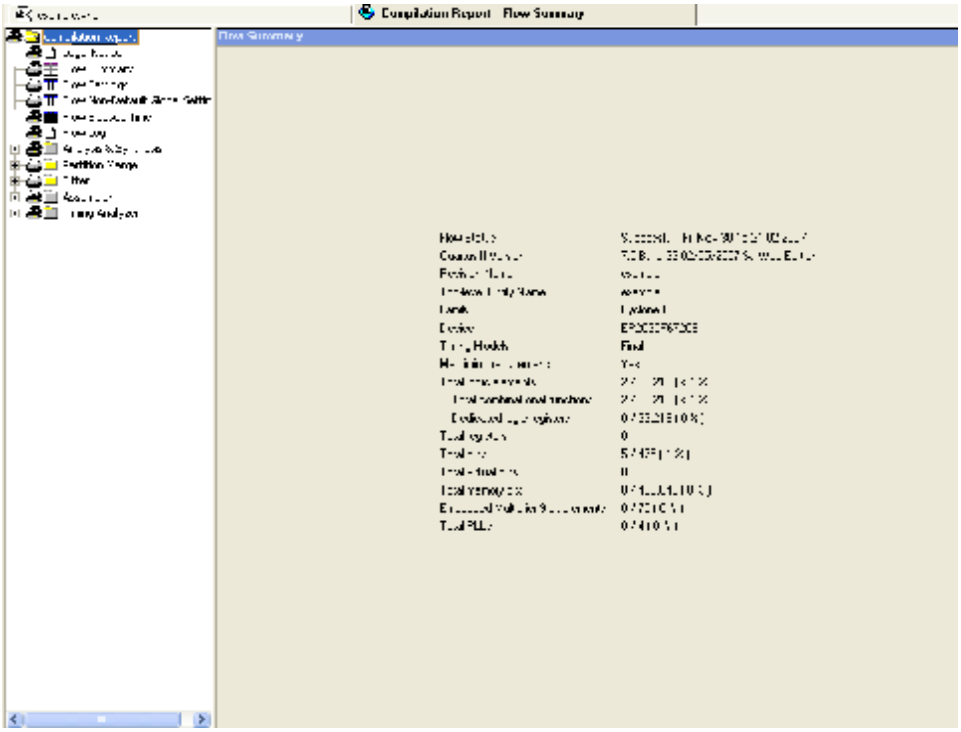
Οι μονάδες του QuartusII ελέγχονται από ένα πρόγραμμα εφαρμογής, τον compiler. Ο compiler μπορεί να χρησιμοποιηθεί για την εκτέλεση της λειτουργίας μιας μόνο μονάδας κάθε φορά, ή για την εκτέλεση πολλαπλών μονάδων στη σειρά. Υπάρχουν πολλοί τρόποι πρόσβασης του compiler στο QuartusII. Όπως φαίνεται στη Εικόνα 310 επιλέγοντας το αριστερό κουμπί κάτω από το Analysis&Synthesis θα εκτελεστεί αυτή η μονάδα. Ομοίως, η μονάδα Fitter θα εκτελεστεί επιλέγοντας το αριστερό κουμπί της εικόνας. Πατώντας το κουμπί Start εκτελούνται όλες οι μονάδες Εικόνα 311 κατά σειρά.

Μια άλλη εναλλακτική μέθοδος πρόσβασης του εξομοιωτή είναι με τη χρήση του μενού Processing | Start. Για να εκτελεστεί η μονάδα σύνθεσης επιλέγεται το Processing | Start | StartAnalysis&Synthesis. Ένα μέρος της σύνθεσης μπορεί να εκτελεστεί επιλέγοντας εναλλακτικά το Processing | Start | StartAnalysis&Elaboration. Αυτή η εντολή εκτελεί μόνο το αρχικό στάδιο της μεταγλώττισης, στο οποίο ελέγχεται το project για λάθη σύνταξης και καθορίζονται τα βασικά ονόματα του σχεδιασμού, τα οποία περιλαμβάνονται στο project. Η εντολή Processing | Start | Compilation είναι ισοδύναμη με το πάτημα του κουμπιού Start στη Εικόνα 310. Υπάρχει επίσης ένα εικονίδιο στην γραμμή εργαλείων, το οποίο μοιάζει με ένα μοβ τρίγωνο, το οποίο είναι ίδιο με την εντολή αυτή.

Ένας εύκολος τρόπος χρήσης των εργαλείων του CAD είναι η εκτέλεση μόνο όσων μονάδων είναι απαραίτητες σε μια συγκεκριμένη φάση της διαδικασίας σχεδίασης. Έτσι μειώνεται ο χρόνος που απαιτείται για τη σύνθεση ενός αρκετά μεγάλου project.

Με την επιλογή Processing | Start | StartAnalysis&Synthesis ή του αντίστοιχου εικονιδίου ανοίγει ο compiler. Καθώς η διαδικασία ξεκινά, η πρόοδός της αναφέρεται στην κάτω δεξιά γωνία του παραθύρου του QuartusII καθώς και στο παράθυρο κατάστασης στην αριστερή πλευρά (αν το παράθυρο δεν είναι ανοιχτό μπορεί να ανοιχτεί με την επιλογή View | UtilityWindows | Status). Αν η μεταγλώττιση είναι επιτυχής (ή ανεπιτυχής) αναφέρεται σε ένα pop-up κουτί.

Επιλέγοντας OK ο χρήστης αγνοεί το κουτί και μπορεί να εξετάσει την αναφορά της μεταγλώττισης όπως φαίνεται και στο Εικόνα 311.



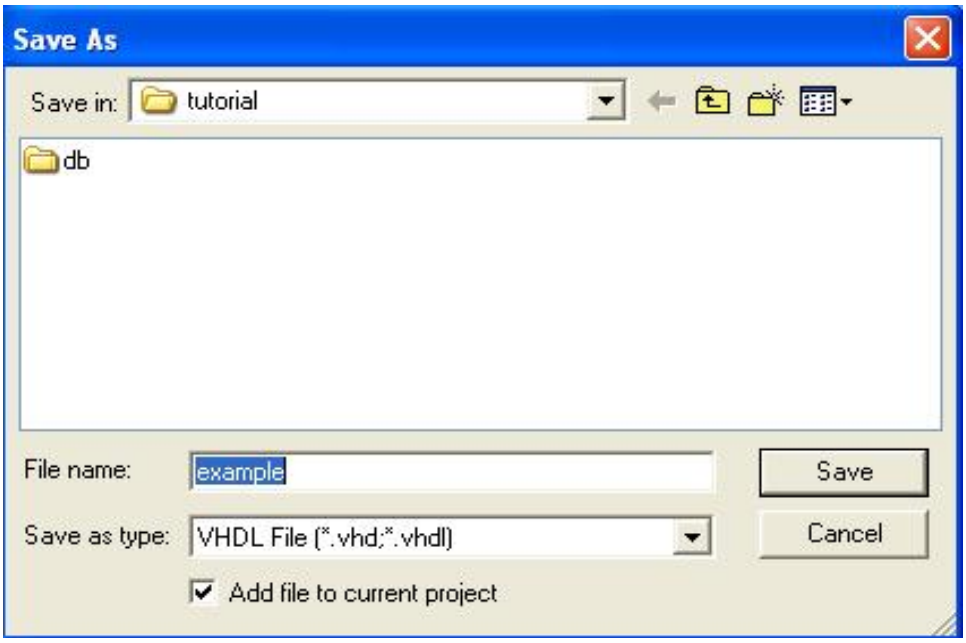
Εικόνα 311 Παράθυρο αναφοράς compiler

Αν η αναφορά δεν είναι ανοιχτή μπορεί να ανοιχτεί επιλέγοντας το εικονίδιο Report στο παράθυρο του εξομοιωτή, χρησιμοποιώντας το αντίστοιχο εικονίδιο της γραμμής εργαλείων, το οποίο μοιάζει με ένα λευκό φύλλο πάνω από ένα μπλε ολοκληρωμένο ή επιλέγονταςProcessing | CompilationReport. Η αναφορά παρουσιάζει ότι το κύκλωμα που μεταγλωττίστηκε χρησιμοποιεί μόνο 5 ακροδέκτες και 2 λογικά στοιχεία του CycloneIIFPGA.

Η αναφορά παραθέτει ένα σύνολο πληροφοριών οι οποίες είναι χρήσιμες για το σχεδιαστή. Το QuartusII παρουσιάζει μηνύματα κατά τη διάρκεια της μεταγλώττισης στο παράθυρο μηνυμάτων. Το παράθυρο αυτό βρίσκεται στο κάτω μέρος του QuartusII, όπως φαίνεται στη Εικόνα 31. Αν ο κώδικας είναι σωστός αναφέρεται μόνο ένα μήνυμα, το οποίο δηλώνει ότι η μεταγλώττιση ήταν επιτυχής και ότι δεν υπάρχουν λάθη ή προειδοποιήσεις.

3.3.1. Σώσιμο Αρχείου

Επιλέγοντας File | Save as… ανοίγει το παράθυρο που φαίνεται στην Εικόνα 312. Στο κουτί Saveastype επιλέγεται VHDLFile. Στο κουτί FileName πρέπει να πληκτρολογηθεί το όνομα του αρχείου. Το κουτί που είναι μαρκαρισμένο στο κάτω μέρος της εικόνας, παραμένει ως έχει (Addfiletocurrentproject). Έτσι το QuartusII πληροφορείται ότι το νέο αρχείο είναι τμήμα του τρέχον ανοιχτού project. Επιλέγεται Save και το αρχείο αποθηκεύεται.



Εικόνα 312 Παράθυρο αποθήκευσης

3.3.2. Άνοιγμα Project

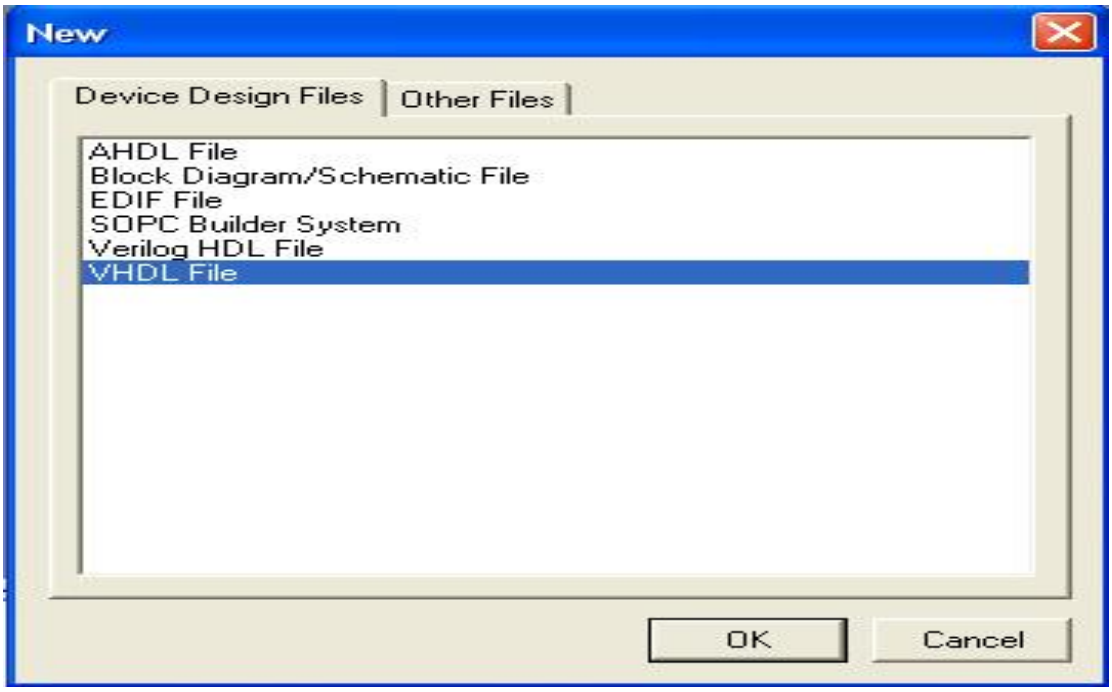
Για να ανοίξετε ένα project πηγαίνετε πάλι στο μενού File και επιλέξτε Open ή πατήστε τη συντόμευση Ctrl-O. Ακολουθώντας αυτή τη διαδικασία μαζί με το project θα ανοίξουν και όλα τα σχετικά αρχεία που περιέχονται σε αυτό.

3.3.3. WaveformEditor

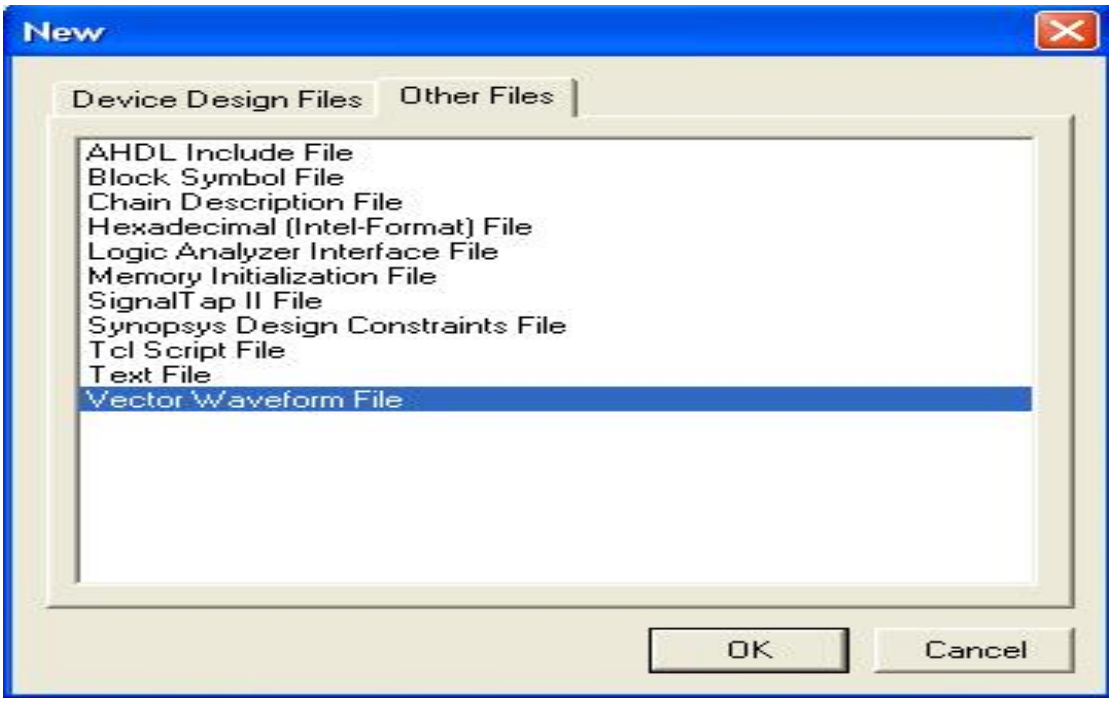
Το QuartusII περιλαμβάνει ένα εργαλείο το οποίο μπορεί να χρησιμοποιηθεί για την εξομοίωση της συμπεριφοράς του κυκλώματος. Πριν την εξομοίωση του κυκλώματος είναι απαραίτητη η δημιουργία απαιτούμενων κυματομορφών, οι οποίες ονομάζονται testvectors και οι οποίες αναπαριστούν τα σήματα εισόδου. Οι κυματομορφές θα σχεδιαστούν με τη χρήση του QuartusIIWaveformEditor.

Το παράθυρο του WaveformEditorανοίγει επιλέγοντας File | New, όπως φαίνεται στην Εικόνα 313.Πατώντας την επιλογή OtherFiles ανοίγει το παράθυρο του

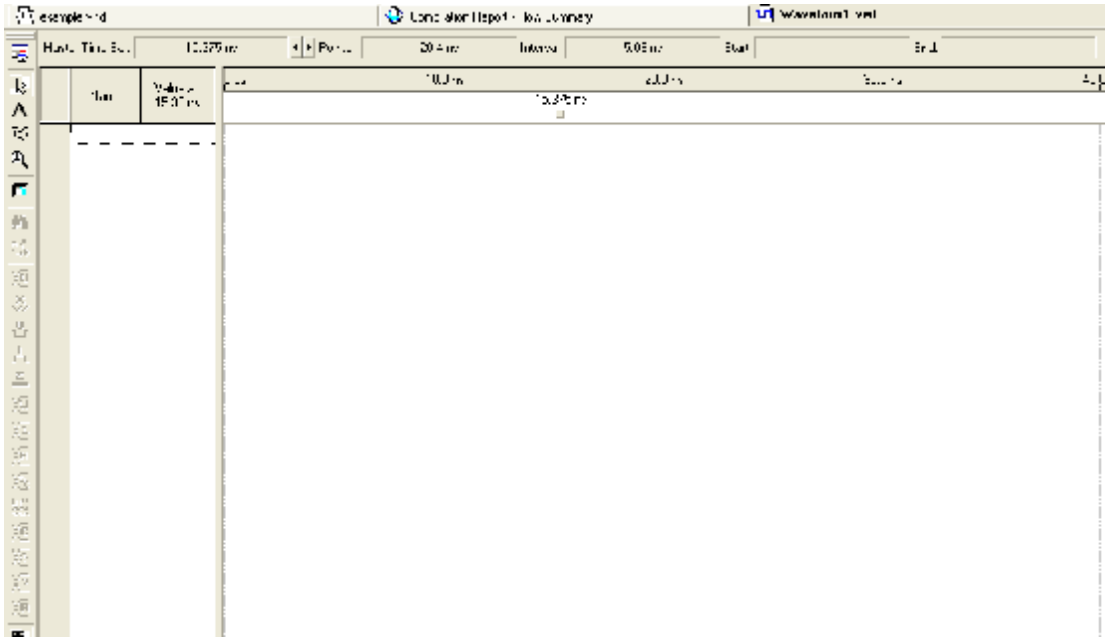
Εικόνα 314 και επιλέγοντας VectorWaveformFile και πατώντας OK ανοίγει το παράθυρο.



Εικόνα 313 Παράθυρο για έναν νέο WaveformEditor

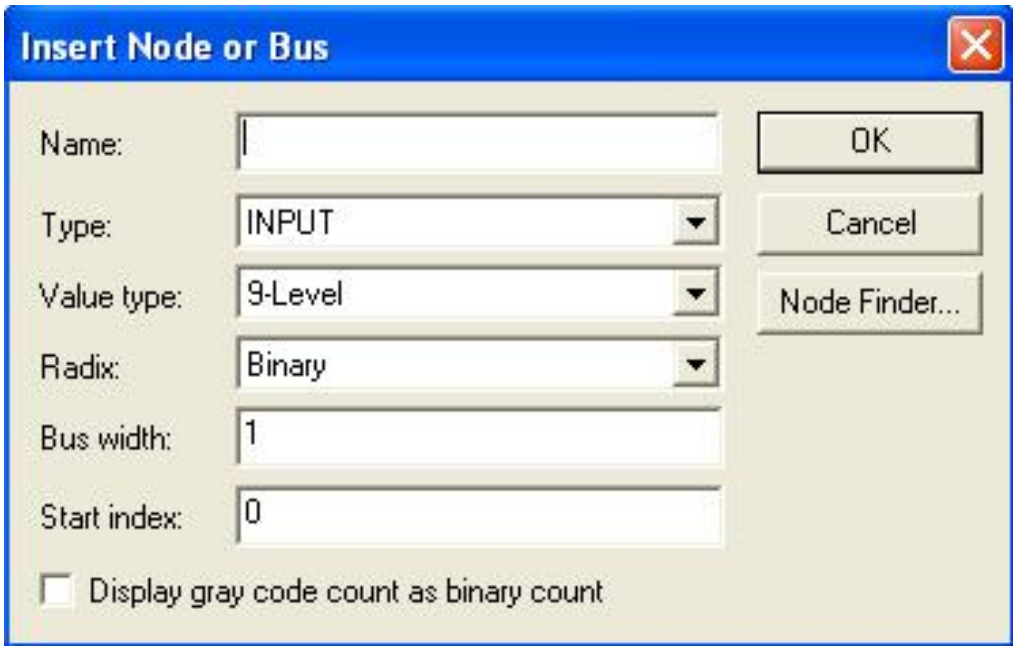


Εικόνα 314 Παράθυρο για την επιλογή του WaveformEditor

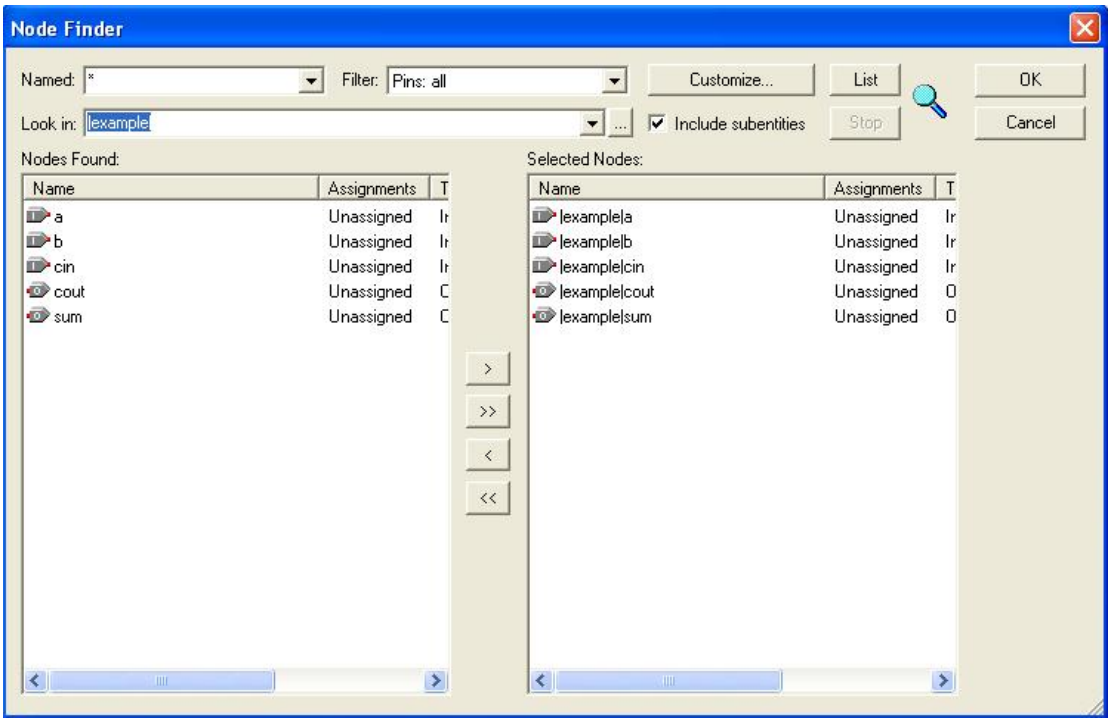


Εικόνα 315 Παράθυρο του WaveformEditor

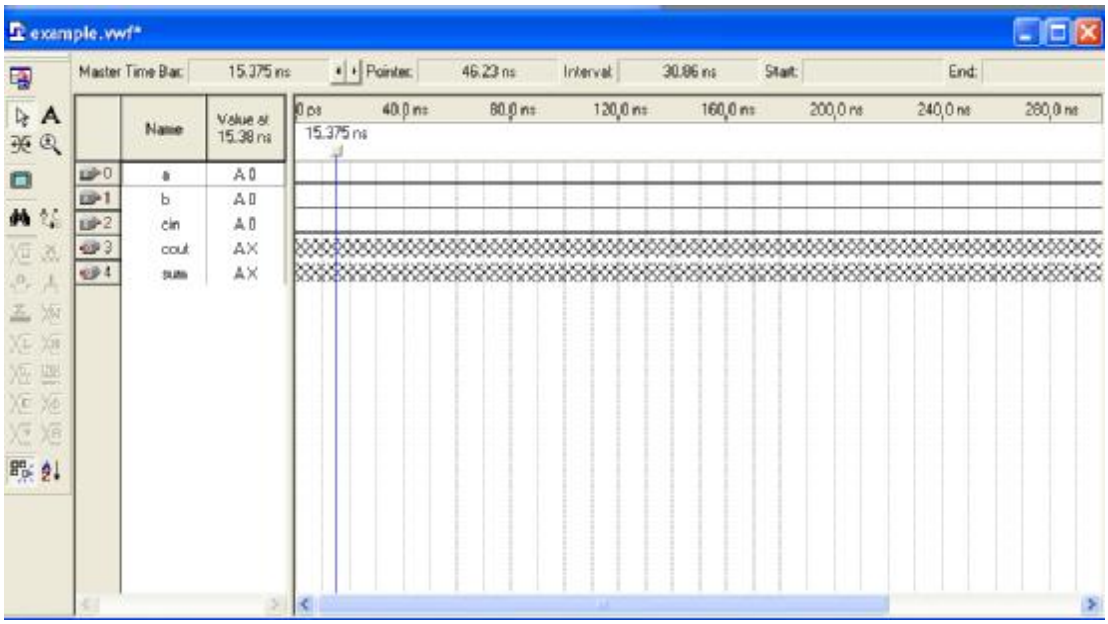
Επιλέγοντας Edit | InsertNodeorBus ανοίγει το παράθυρο της Εικόνα 316. Είναι δυνατό να γραφτεί το όνομα ενός σήματος (ακροδέκτη) στο κουτί Name, αλλά είναι πιο εύκολο να ανοίξει το παράθυρο του Εικόνα 317. Με τον ίδιο τρόπο προστίθενται και τα υπόλοιπα Εικόνα 318 πατώντας το κουμπί NodeFinder. Ο NodeFinder έχει ένα φίλτρο το οποίο χρησιμοποιείται για τον καθορισμό του τύπου του node που είναι προς αναζήτηση. Καθώς η αναζήτηση βασίζεται στους ακροδέκτες εισόδου και εξόδου, το φίλτρο (filter) επιλέγεται ως Pins:all. Πατώντας στο κουμπί List βρίσκονται όλοι οι ακροδέκτες εισόδου και εξόδου. Ο NodeFinder εμφανίζει στο αριστερό μέρος του παραθύρου του ακροδέκτες a, b, cin, cout και sum. Επιλέγοντας τον b και πατώντας το σύμβολο > το σήμα προστίθεται στο κουτί SelectedNodes στα δεξιά του σχήματος.



Εικόνα 316 Εισαγωγή ονόματος ενός σήματος (ακροδέκτη)



Εικόνα 317 Εύρεση και επιλογήσήματος (ακροδέκτη)



Εικόνα 318Παράθυρο του WaveformEditor με τα σήματα.

3.3.4. Simulator

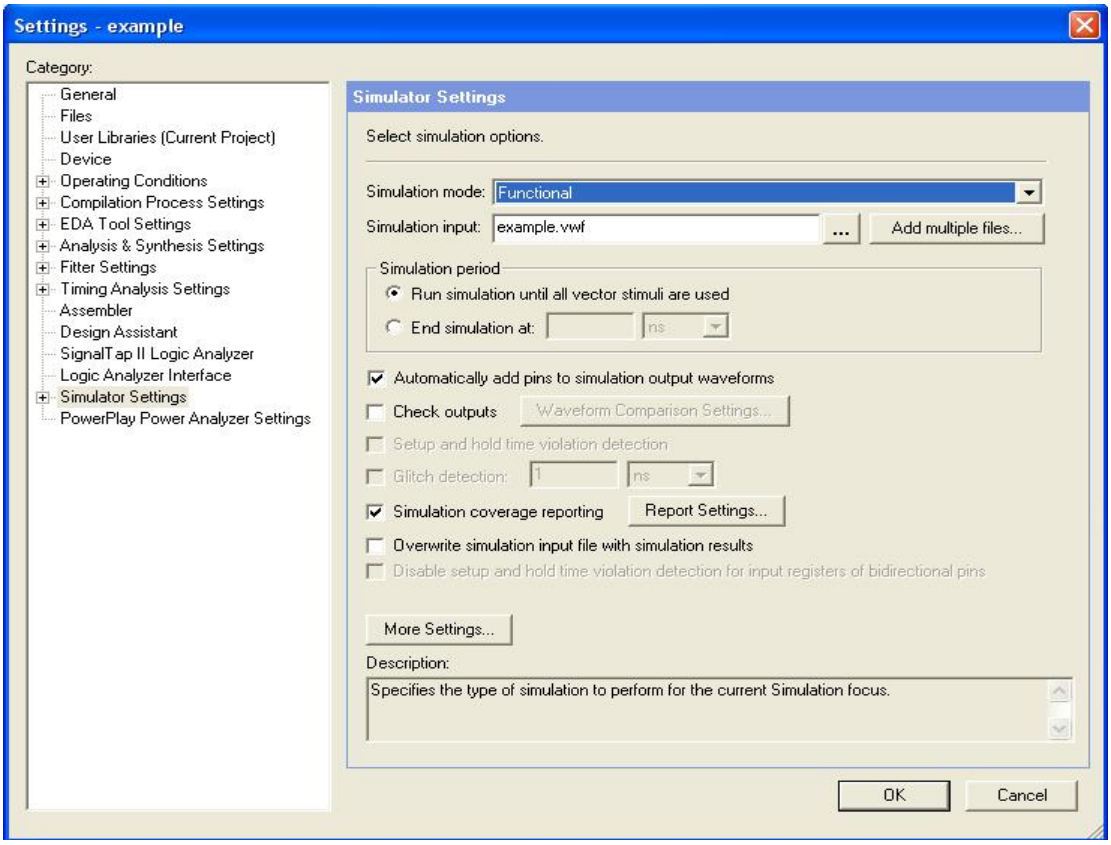
Ένα κύκλωμα μπορεί να εξομοιωθεί με δυο τρόπους. Ο πιο απλός τρόπος είναι να θεωρηθεί ότι τα λογικά στοιχεία και τα καλώδια διασύνδεσης είναι τέλεια, έτσι ώστε να μην προκαλείται καμιά καθυστέρηση στη διάδοση των σημάτων μέσα στο κύκλωμα. Αυτή ονομάζεται functional (λειτουργική) εξομοίωση. Σύμφωνα με τον δεύτερο τρόπο λαμβάνονται υπόψη όλες οι καθυστερήσεις, γεγονός που οδηγεί σε μια timing (χρονική) εξομοίωση. Τυπικά, η λειτουργική εξομοίωση χρησιμοποιείται για να διαπιστωθεί κατά πόσο είναι σωστό το κύκλωμα που σχεδιάστηκε. Η λειτουργική εξομοίωση απαιτεί λιγότερο χρόνο, επειδή η εξομοίωση χρησιμοποιεί μόνο τις λογικές εκφράσεις οι οποίες καθορίζουν το κύκλωμα.

Για την εκτέλεση της λειτουργικής εξομοίωσης επιλέγεται AssignmentsSettings για να ανοίξει το παράθυρο των ρυθμίσεων. Στην αριστερή πλευρά του παραθύρου πατώντας πάνω στο Simulator ανοίγει το παράθυρο της Εικόνα 319από το οποίο γίνεται η επιλογή Functional ως τρόπου εξομοίωσης. Για την ολοκλήρωση της προετοιμασίας του εξομοιωτή επιλέγεται η εντολή Processing | GenerateFunctionalSimulationNetlist. Ο εξομοιωτής του QuartusII παίρνει τα σήματα εισόδου και παράγει τις εξόδους οι οποίες καθορίζονται στο αρχείο example.vwf. Ο εξομοιωτής ξεκινά επιλέγοντας Processing |StartSimulation, ή χρησιμοποιώντας το εικονίδιο που βρίσκεται στη γραμμή εργαλείων το οποίο μοιάζει με ένα μπλε τρίγωνο με ένα τετράγωνο κύμα κάτω από αυτό. Στο τέλος της εξομοίωσης, το QuartusII αναγνωρίζει αν η εξομοίωση ήταν επιτυχής με την εμφάνιση μιας αναφοράς, όπως φαίνεται και στη Εικόνα 320, όπως φαίνεται και στη Εικόνα 320, ο εξομοιωτής παράγει μια κυματομορφή για κάθε έξοδο.

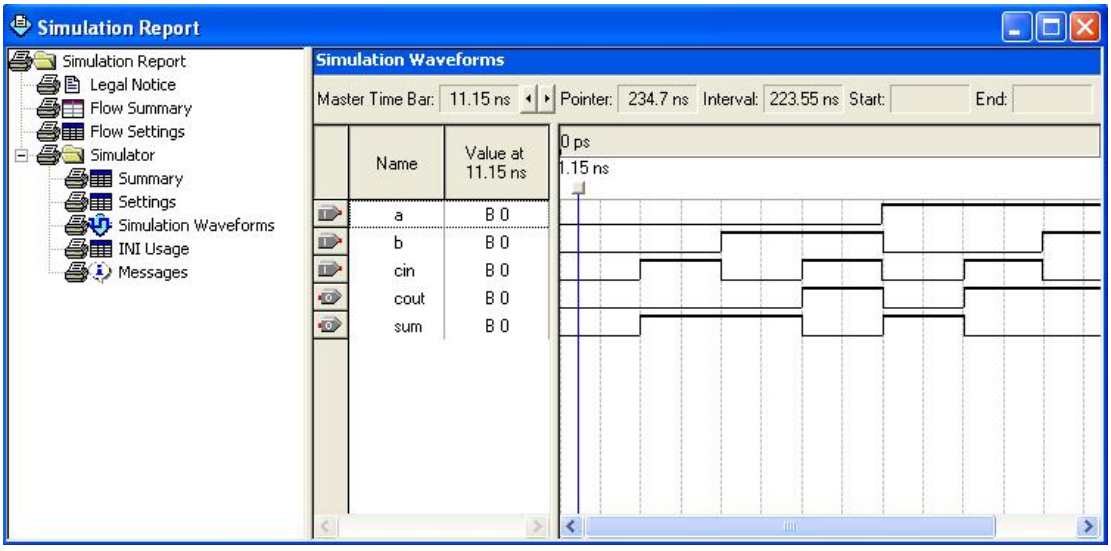
3.3.5. Εισαγωγή Κυματομορφών Στο WORD

Στις εργασίες στις οποίες σας ζητείται να συμπεριλάβετε τις κυματομορφές που προκύπτουν από την εξομοίωση ακολουθήστε την εξής διαδικασία: πατήστε το πλήκτρο **PrintScreen** (βρίσκεται πάνω δεξιά στο πληκτρολόγιο), στη συνέχεια ανοίξτε ένα σχεδιαστικό πρόγραμμα (όπως το *paint* των Windows ή το *Photoshop*) και κάντε **paste** σε ένα νέο **document**. Από εκεί επιλέξτε το μέρος της οθόνης που περιέχει τις κυματομορφές,

αντιγράψτε τις (*copy*) και επικολλήστε τις (*paste*) στο αντίστοιχο πλαίσιο της εκάστοτε εργασίας.



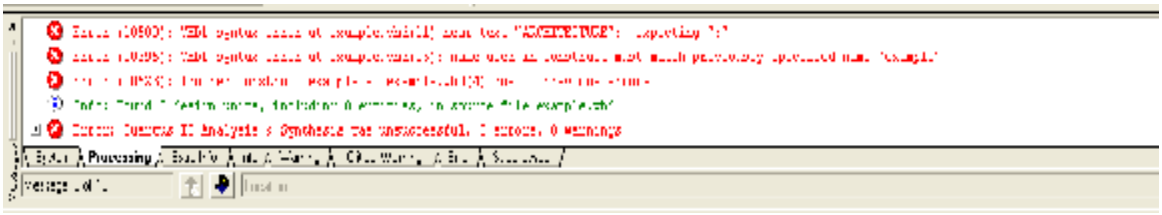
Εικόνα 319 Παράθυρο ρυθμίσεων προσομοιωτή



Εικόνα 320 Κυματομορφές εξομοιωτή

3.3.6. Χρησιμοποιώντας Τον MessageProcessor Για Τον Εντοπισμό Και Τη Διόρθωση Λαθών.

Ο Compiler θα πρέπει να εμφανίζει ένα μήνυμα που να λέει ότι το project έγινε compile με επιτυχία και να δείχνει 0 προειδοποιήσεις και 0 λάθη. Σε περίπτωση λάθους θα εμφανιστεί μια εικόνα παρόμοια με την ακόλουθη:

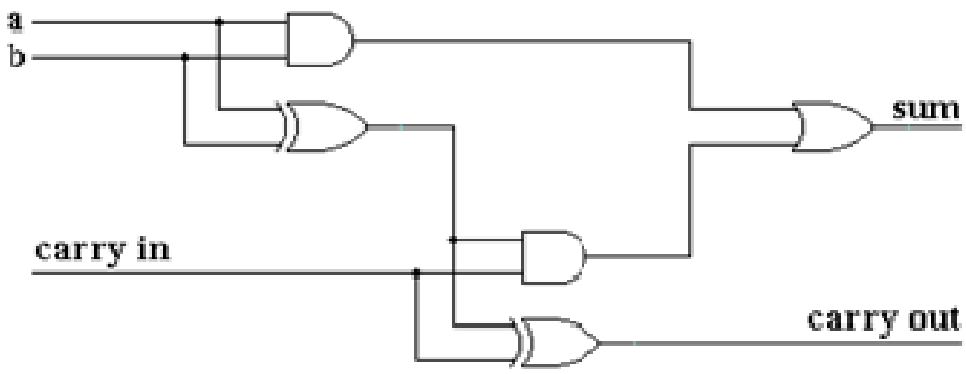


Εικόνα 321 Λάθη εξομοίωσης

Επιλέγοντας ένα μήνυμα λάθους και διπλό-πατώντας πάνω σε αυτό εντοπίζεται το σημείο του κώδικα VHDL που αντιστοιχεί στο μήνυμα. Το παράθυρο του TextEditor αυτόματα εμφανίζει τη γραμμή αυτή τονισμένη.

3.3.7. Σχεδιασμός με τη χρήση σχηματικού διαγράμματος

Στην προηγούμενη άσκηση παρουσιάστηκε ο τρόπος με τον οποίο μπορεί να χρησιμοποιηθεί το Quartus II για να περιγραφεί ένα λογικό κύκλωμα σε VHDL και να εξομοιωθεί. Εδώ θα χρησιμοποιηθεί γραφική μέθοδος για να κατασκευαστεί το ίδιο κύκλωμα (ένας full adder), το οποίο φαίνεται στο Σχήμα 31..



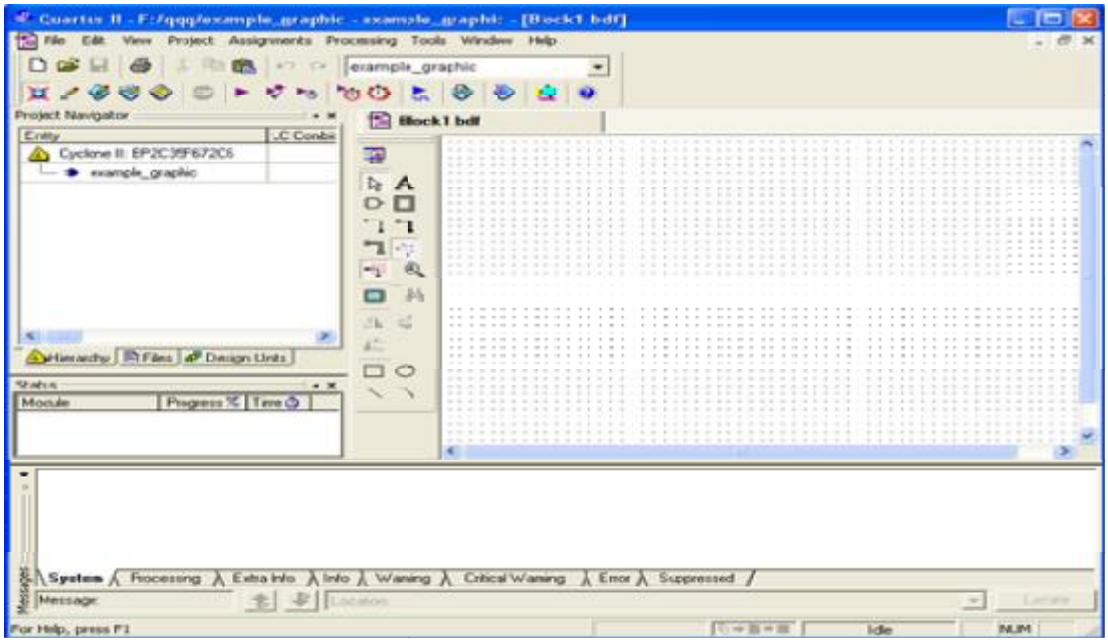
Σχήμα 31 Το κύκλωμα ενός full adder

3.3.8. Προσδιορίζοντας το όνομα του project

Για το νέο project χρησιμοποιείται το directory c:\QuartusII\tutorial1. Χρησιμοποιώντας το New Project Wizard δημιουργείται το νέο project, όπως παρουσιάστηκε στην προηγούμενη άσκηση. Το project παίρνει το όνομα example_graphic, ενώ επιλέγεται η ίδια οικογένεια ολοκληρωμένων FPGA για υλοποίηση.

3.3.9. Χρησιμοποιώντας τον Block Editor

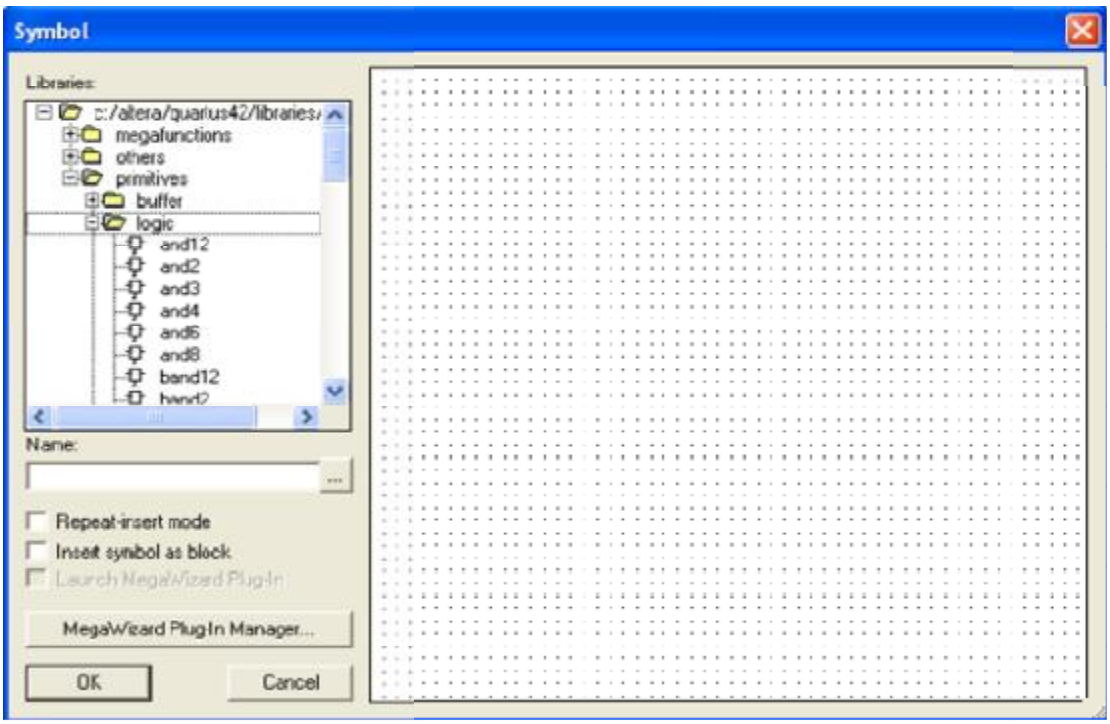
Το επόμενο βήμα είναι η σχεδίαση του κυκλώματος. Επιλέγοντας File | New | Block Diagram / Schematic File και πατώντας OK ανοίγει το παράθυρο του Block Editor το οποίο φαίνεται στην Εικόνα 322. Σχεδιάζοντας το κύκλωμα σε αυτό το παράθυρο παράγεται το επιθυμητό αρχείο του διαγράμματος.



Εικόνα 322 Το παράθυρο του block editor

3.3.10. Εισάγοντας σύμβολα λογικών πυλών

Ο Block Editor παρέχει αρκετές βιβλιοθήκες, οι οποίες περιλαμβάνουν κυκλωματικά στοιχεία που μπορούν να χρησιμοποιηθούν σε ένα σχηματικό. Για το παράδειγμά του full adder, όπου χρειάζονται 5 απλές πύλες, θα χρησιμοποιηθεί η βιβλιοθήκη που ονομάζεται Primitives, και η οποία περιέχει τις βασικές λογικές πύλες. Κάνοντας διπλό click στο κενό χώρο του Block Editor ή επιλέγοντας Edit | Insert Symbol ή κάνοντας click στο εικονίδιο της AND λογικής πύλης στην γραμμή εργαλείων θα εμφανιστεί το παράθυρο της Εικόνα 323. Στο σχήμα, το κουτί με την ονομασία Libraries περιλαμβάνει διάφορες βιβλιοθήκες του Quartus II. Η λίστα μπορεί να επεκταθεί κάνοντας click στο μικρό σύμβολο + δίπλα στο c:\altera\72\quartus\libraries, κάνοντας click στην συνέχεια στο + δίπλα στο primitives και τέλος κάνοντας click στο + δίπλα στο logic. Κάνοντας διπλό click στο σύμβολο AND 2 εισάγεται στο σχηματικό AND 2 (ή, εναλλακτικά, κάνοντας απλό click και μετά πατώντας OK). Έτσι θα εμφανιστεί στο παράθυρο του Block Editor μια πύλη AND 2 εισόδων. Χρησιμοποιώντας το mouse, μπορεί να μετακινηθεί το σύμβολο στη θέση που επιλέχθηκε και να τοποθετηθεί εκεί απλά κάνοντας ένα click με το mouse.



Εικόνα 323 Επιλογή λογικών συμβόλων

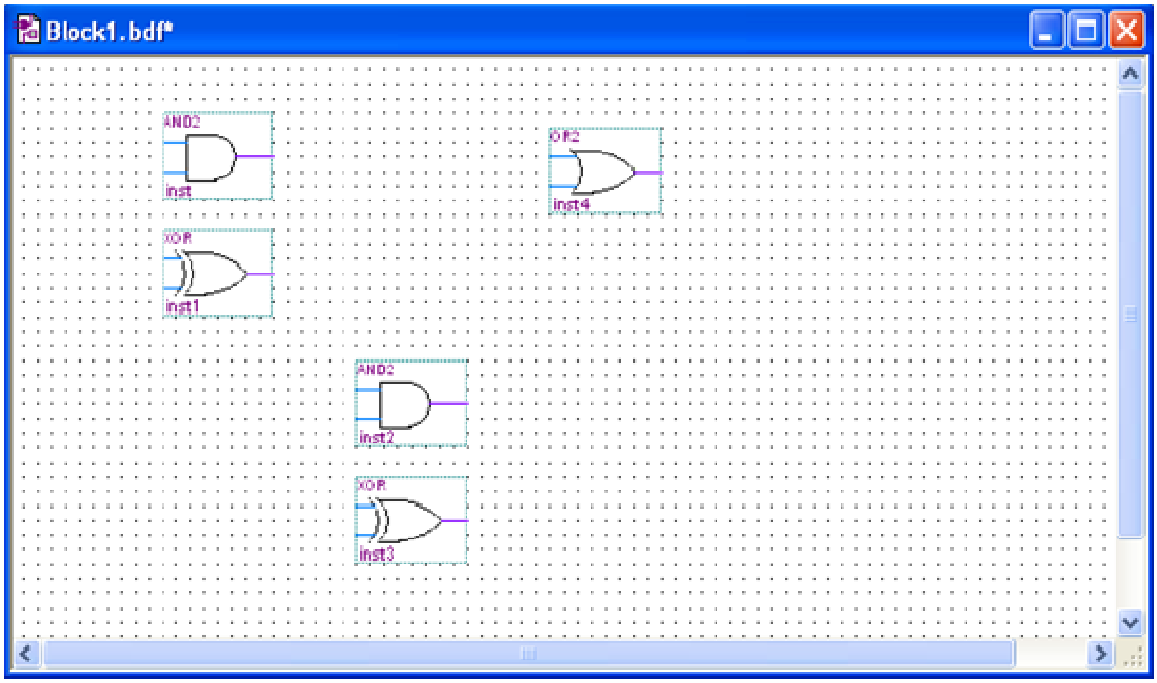
Κάθε σύμβολο σε ένα schematic μπορεί να επιλεγεί τοποθετώντας το δείκτη του ποντικιού πάνω από το σύμβολο και κάνοντας click. Γύρω από το επιλεγμένο σύμβολο εμφανίζεται ένα έντονο πλαίσιο. Για να αποεπιλεγεί, αρκεί ο χρήστης να κάνει click οπουδήποτε αλλού στο παράθυρο του Graphic Editor. Ένα σύμβολο μπορεί να μετακινηθεί επιλέγοντάς το και κρατώντας το κουμπί του ποντικιού πατημένο. Οι guidelines που υπάρχουν στο παράθυρο του Block Editor βοηθούν στο να τοποθετηθούν τα σύμβολα στην επιφάνεια εργασίας. Εμφανίζονται επιλέγοντας View | Show Guidelines.

Ένας πλήρης αθροιστής απαιτεί 2 πύλες AND , 2 πύλες XOR και 1 πύλη OR (όλες με 2 εισόδους). Στον Block Editor έχει ήδη εισαχθεί μια πύλη AND . Μπορεί να δημιουργηθεί ένα αντίγραφο αυτής της πύλης επιλέγοντας την και μετά, κρατώντας πατημένο το πλήκτρο Ctrl, σύροντας το ποντίκι σε κάποιο άλλο σημείο της επιφάνειας εργασίας. Με αυτόν τον τρόπο εισαγωγής στοιχείων μπορεί κανείς να δημιουργεί αντίγραφα πυλών αλλά ακόμα και κυκλωμάτων σε ένα schematic.

Για την εισαγωγή μιας πύλης XOR η διαδικασία είναι ίδια. Το κύκλωμα του πλήρη αθροιστή απαιτεί δυο πύλες XOR 2 εισόδων. Με τον τρόπο που περιγράφηκε, μπορεί να δημιουργηθεί ένα αντίγραφο της πύλης XOR που ήδη εισήχθη. Τέλος, με τον ίδιο τρόπο, εισάγεται και μια πύλη OR 2 εισόδων.

Τα σύμβολα στο schematic μπορούν να μετακινηθούν επιλέγοντάς τα ένα-ένα όπως περιγράφηκε πιο πριν. Περισσότερα από ένα σύμβολα μπορούν να επιλεγούν ταυτόχρονα εάν κάνοντας click και drag με το ποντίκι και περιβάλλοντάς τα με ένα ορθογώνιο πλαίσιο. Κατόπιν, μετακινώντας κάποιο από τα σύμβολα μετακινούνται όλα μαζί. Εάν μια επιλογή (ένα ή περισσότερα σύμβολα) πρέπει να περιστραφεί αυτό γίνεται επιλέγοντας Edit | Rorate by Degrees. Στο menu που θα εμφανιστεί υπάρχουν οι επιλογές Flip Horizontal, Flip Vertical και Rotate | 90^o, 180^o, 270^o.

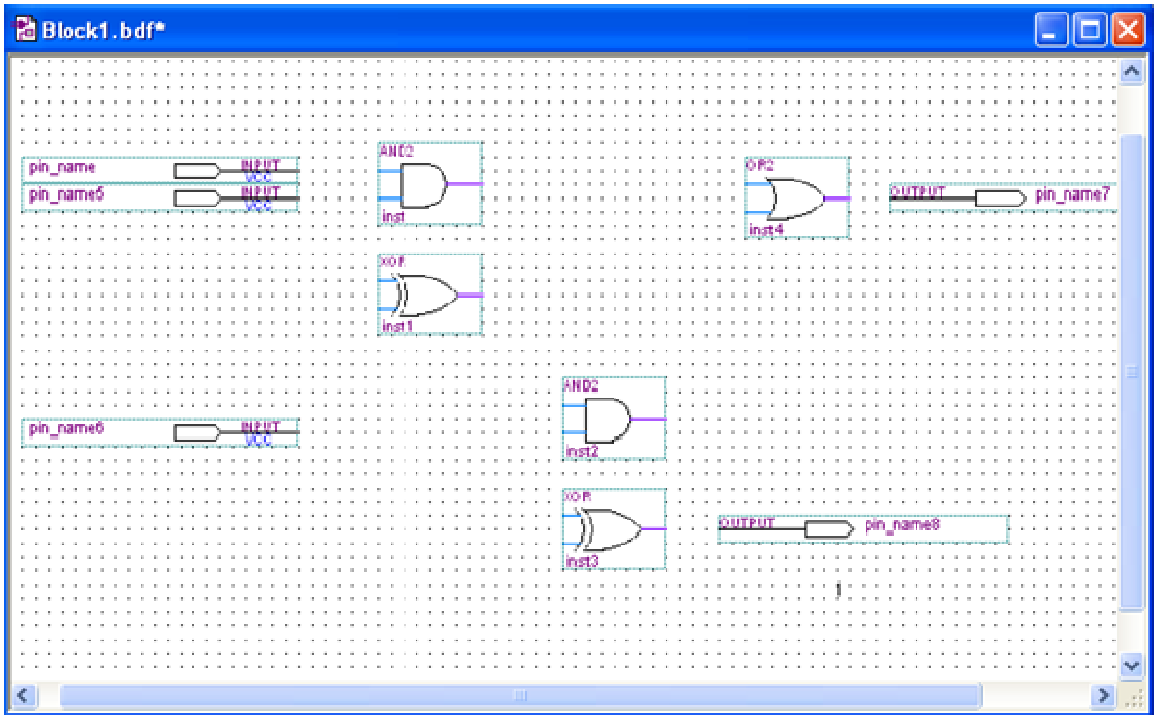
Πειραματιζόμενος κανείς με αυτές τις λειτουργίες του Graphic Editor μπορεί να αποκτήσει ευχέρεια στο σχεδιασμό schematics. Για το παράδειγμα του full adder, τα σύμβολα διευθετήθηκαν με τον τρόπο που φαίνεται στη Εικόνα 324.



Εικόνα 324 Διευθέτηση των πυλών

3.3.11. Εισάγοντας σύμβολα εισόδου και εξόδου

Τώρα που οι λογικές πύλες έχουν εισαχθεί στο schematic, είναι απαραίτητο να εισαχθούν σύμβολα που να αναπαριστούν τις εισόδους και εξόδους του κυκλώματος. Αυτά τα σύμβολα βρίσκονται στη βιβλιοθήκη Primitives. Ανοίγοντας τη βιβλιοθήκη και κατεβαίνοντας με την βοήθεια της μπάρας κύλισης μετά τις πύλες εμφανίζεται η ένδειξη pins. Από αυτά εισάγεται το σύμβολο input στο σχηματικό. Απαιτούνται ακόμα δυο σύμβολα input καθώς και δυο σύμβολα output. Η διευθέτηση των συμβόλων έγινε με τον τρόπο που φαίνεται στη Εικόνα 325.



Εικόνα 325 Εισαγωγή συμβόλων εισόδου/εξόδου

3.3.12. Προσδίδοντας ονόματα στα σύμβολα εισόδου και εξόδου

Ο προσδιορισμός του ονόματος του συμβόλου εισόδου στην πάνω αριστερή γωνία του schematic γίνεται κάνοντας διπλό click στη λέξη pin_name. Έτσι επιλέγεται το όνομα του ακροδέκτη επιτρέποντας την πληκτρολόγηση ενός νέου ονόματος. Πληκτρολογώντας ένα όνομα, π.χ. a και πατώντας Enter επιλέγεται όνομα του ακριβώς από κάτω ακροδέκτη. Ο ακροδέκτης αυτός είναι ο b ενώ ο επόμενος είναι ο cin. Τέλος οι δυο ακροδέκτες εξόδου είναι οι cout (ο πάνω) και sum (ο κάτω).

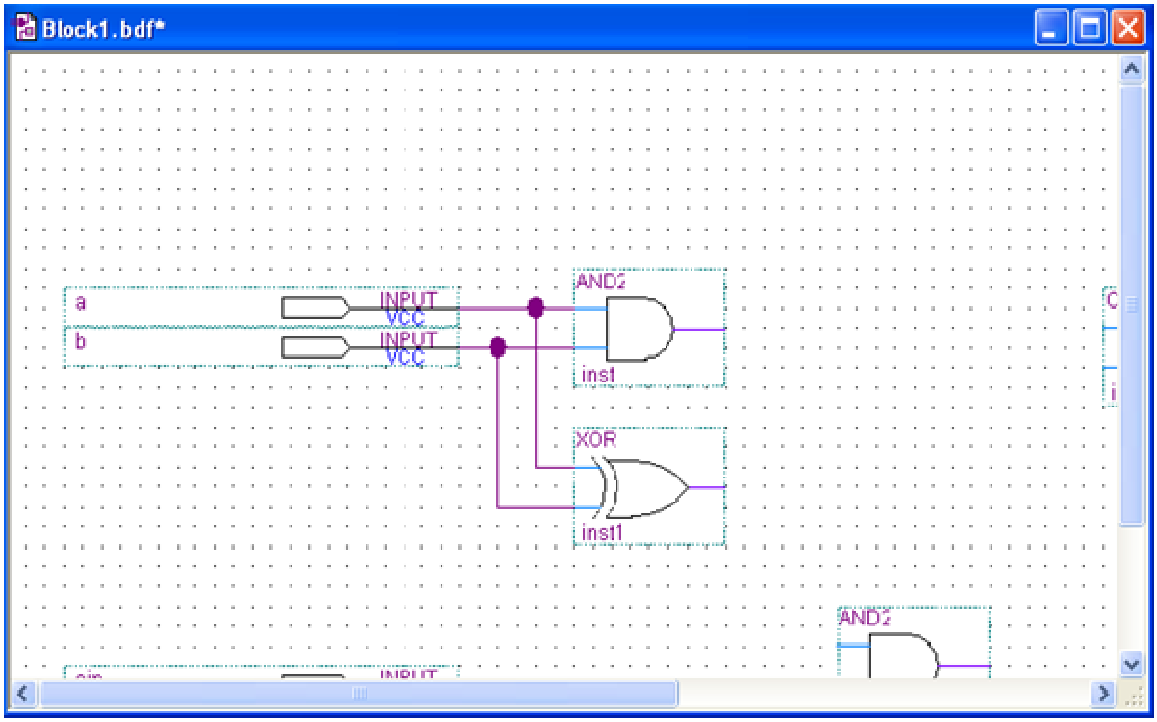
3.3.13. Συνδέοντας κόμβους με καλώδια

Το επόμενο βήμα είναι ο σχεδιασμός γραμμών (καλώδια) για τη διασύνδεση των συμβόλων στο σχηματικό. Το εργαλείο σχεδιασμού επιλέγεται κάνοντας click στην εικόνα που μοιάζει σαν αιχμή βέλους στην κάθετη γραμμή εργαλείων. Αυτό το εικονίδιο ονομάζεται Selection AND Smart Drawing tool, και επιτρέπει στον Block Editor να αλλάζει αυτόματα κατάσταση ανάλογα με τη δυνατότητα επιλογής ενός συμβόλου στην οθόνη ή της σχεδίασης καλωδίων για τη σύνδεσή τους. Η κατάλληλη κατάσταση επιλέγεται ανάλογα με το σημείο που δείχνει ο δείκτης του ποντικιού.

Μετακινώντας το δείκτη του ποντικιού πάνω από το σύμβολο εισόδου a και συγκεκριμένα πάνω από τη δεξιά άκρη του, ο δείκτης μετατρέπεται σε σταυρό. Όσο έχει το σχήμα της αιχμής βέλους μπορεί να χρησιμοποιηθεί για την επιλογή του συμβόλου κάνοντας click. Όσο βρίσκεται πάνω από τη λεπτή γραμμή στη δεξιά άκρη του συμβόλου (ονομάζεται pinstub) και έχει τη μορφή του σταυρού επιτρέπει, κάνοντας click και drag, να σχεδιαστούν καλώδια που θα το συνδέσουν με άλλα pinstubs στο schematic. Μια σύνδεση μεταξύ δύο ή περισσοτέρων pinstubs ονομάζεται κόμβος (node).

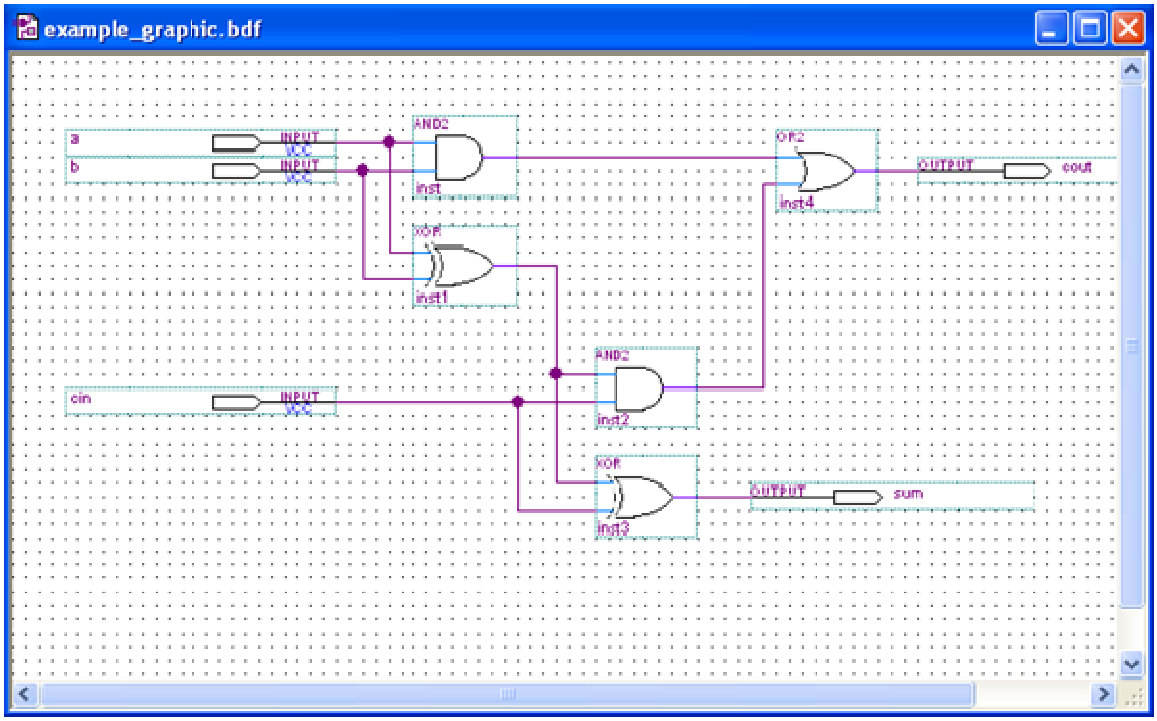
Με τον ίδιο τρόπο μπορεί να συνδεθεί και το σύμβολο εισόδου b με τον άλλο ακροδέκτη της πύλης AND . Κατόπιν πρέπει να σχεδιαστεί μια γραμμή από τον πάνω ακροδέκτη της πύλης XOR μέχρι την πρώτη γραμμή που σχεδιάστηκε. Στο σημείο που

συνδέονται οι γραμμές εμφανίστηκε μια τελεία. Αυτό σημαίνει ότι τα τρία pinstubs που συνδέθηκαν αποτελούν ένα κόμβο. Στη Εικόνα 326 δείχνει μια μεγέθυνση του κυκλώματος που περιέχει τις συνδέσεις που περιγράφηκαν μέχρι τώρα. Για τη μεγέθυνση ή σμίκρυνση του κομματιού του κυκλώματος που εμφανίζεται στην οθόνη χρησιμοποιούνται τα δυο εικονίδια που μοιάζουν με μεγεθυντικούς φακούς, στην κάθετη γραμμή εργαλείων.



Εικόνα 326 Μεγένθυνση των συνδέσεων στις πρώτες πύλες

Για να ολοκληρωθεί το schematic πρέπει να συνδεθούν και τα υπόλοιπα στοιχεία του κυκλώματος όπως φαίνεται και στη Εικόνα 327. Εάν κατά τη σύνδεση των συμβόλων γίνει κάποιο λάθος οι λανθασμένες γραμμές διαγράφονται επιλέγοντάς τις με το ποντίκι και πατώντας Delete ή επιλέγοντας Edit | Delete. Τέλος με File | Save (Ctrl+S) αποθηκεύεται το τελικό σχηματικό.



Εικόνα 327 Το ολοκληρωμένο κύκλωμα του full adder

Αναδιοργανώνοντας το κύκλωμα, επιλέγοντας μια πύλη και μετακινώντας την, τα καλώδια αναδιαμορφώνονται αυτόματα. Αυτό γίνεται επειδή το Quartus II έχει ένα χαρακτηριστικό το οποίο ονομάζεται rubberb AND ing και το οποίο ενεργοποιείται όταν επιλεγεί το Selection AND Smart Drawing tool. Υπάρχει ένα εικονίδιο για το rubberb AND ing το οποίο βρίσκεται στην γραμμή εργαλείων και μοιάζει με ένα καλώδιο σε σχήμα L με μικρά σημάδια στην άκρη του.

Το παράδειγμα του πλήρη αθροιστή είναι αρκετά απλό και οι συνδέσεις στο σχηματικό έγιναν χωρίς να δημιουργηθεί ένα μπερδεμένο διάγραμμα. Σε μεγαλύτερα σχηματικά όμως κάποιοι κόμβοι ή pinstubs που πρέπει να συνδεθούν μπορεί να είναι πολύ μακριά ο ένας από τον άλλον και ο σχεδιασμός καλωδίων μεταξύ τους μπορεί να είναι αρκετά δύσκολος και να δημιουργεί μπερδέματα. Σε αυτές τις περιπτώσεις οι κόμβοι ενώνονται, αποδίδοντας ονόματα σε αυτούς.

3.3.14. Συνθέτοντας ένα κύκλωμα από το σχηματικό

Όπως έχουμε ήδη περιγράψει, επιλέγοντας Processing | Start | Start Analysis AND Synthesis (ή Ctrl-k) ο Compiler ξεκινά τη σύνθεση του κυκλώματος από το σχηματικό. Αν το σχηματικό είναι σωστό ο Compiler θα εμφανίσει ένα μήνυμα το οποίο δηλώνει ότι δεν έχουν βρεθεί λάθη ή προειδοποιήσεις.

Αν ο Compiler εμφανίσει λάθη, τότε έχουν γίνει λάθη κατά την ένωση των καλωδίων. Σε αυτή την περίπτωση παρουσιάζεται ένα μήνυμα για κάθε ένα από τα λάθη. Κάνοντας διπλό click πάνω στο μήνυμα λάθους το καλώδιο το οποίο παρουσιάζει το λάθος θα γίνει έντονο. Ο χρήστης μπορεί να μάθει περισσότερες πληροφορίες για ένα λάθος ή μια προειδοποίηση επιλέγοντας το αντίστοιχο μήνυμα και πατώντας το κουμπί F1.

3.3.15. Λειτουργική εξομοίωση

Η λειτουργική εξομοίωση για το σχηματικό γίνεται ακριβώς με τον ίδιο τρόπο που γίνεται και στην περίπτωση του κώδικα σε VHDL

4. Γενικά Για Την Πλακέτα DE2 Της ALTERA

4.1. Εισαγωγή

Η χαμηλού κόστους οικογένεια Cyclone 2 FPGA της Alteraείναι βασισμένη σε μια 1.2 – V, 90 – nm SRAM διαδικασία με πυκνότητες πάνω από 68K λογικά στοιχεία (LEs) και έως 1.1 Mbits ενσωματωμένης RAM. Με χαρακτηριστικά όπως ενσωματωμένους 18χ18 πολλαπλασιαστές για να υποστηρίζουν υψηλής απόδοσης DSPεφαρμογές,phaselockedloops (PLLs)για τη διαχείριση του ρολογιού του συστήματος και διεπαφή εξωτερικής μνήμης υψηλής ταχύτητας για SRAM και DRAMσυσκευές. Οι συσκευές Cyclone 2είναι μια οικονομική λύση για εφαρμογές υψηλών απαιτήσεων. Οι συσκευέςCyclone 2υποστηρίζουν διαφορικά και μονοτερματικά I/Oπρότυπα συμπεριλαμβάνοντας LVDSσε ρυθμό δεδομένων μέχρι και 805 megabits /sec (Mbps)για τον δέκτη και 640 Mbpsγια τον πομπό και 64 bit, 66 MHz PCI και PCI-X για διασύνδεση με επεξεργαστές και ASSPκαι ASICσυσκευές. Η Alteraεπίσης προσφέρει χαμηλού κόστους συσκευές σειριακής εγκατάστασης για την ρύθμιση των συσκευών Cyclone 2. Η οικογένειαCyclone 2FPGAπροσφέρει εμπορική ποιότητα, βιομηχανική ποιότητα και συσκευές χωρίς μόλυβδο.

Η οικογένεια συσκευώνCyclone 2 προσφέρει τα ακόλουθα χαρακτηριστικά :

- Υψηλής πυκνότητας αρχιτεκτονική με 4,608 έως 68,416 LEs
- M4Kενσωματωμένο blockμνήμης.
- Ενσωματωμένους πολλαπλασιαστές.
- Εξελιγμένη υποστήριξη I/O.
- Ευέλικτη κυκλική διαχείριση ρολογιού.
- Διαμόρφωση συσκευών.
- Πνευματική ιδιοκτησία.

Ο Πίνακας 41παραθέτει χαρακτηριστικά της οικογένειας συσκευώνCyclone 2. Ο Πίνακας 42 παραθέτει τα πλεονεκτήματα των συσκευών τηςCyclone 2και τα μέγιστα I/O pinτου χρήστη.

Table 1–1. Cyclone II FPGA Family Features

Feature	EP2C5	EP2C8	EP2C20	EP2C35	EP2C50	EP2C70
LEs	4,608	8,256	18,752	33,216	50,528	68,416
M4K RAM blocks (4 Kbits plus 512 parity bits	26	36	52	105	129	250
Total RAM bits	119,808	165,888	239,616	483,840	594,432	1,152,000
Embedded multipliers *(1)*	13	18	26	35	86	150
PLLs	2	2	4	4	4	4
Maximum user I/O pins	158	182	315	475	450	622

Πίνακας 41 Χαρακτηριστικά της οικογένειας συσκευώνCyclone 2

Table 1–2. Cyclone II Package Options & Maximum User I/O Pins *Note (1)*

Device	144-Pin TQFP *(2)*	208-Pin PQFP *(3)*	240-Pin PQFP	256-Pin FineLine BGA	484-Pin FineLine BGA	484-Pin Ultra FineLine BGA	672-Pin FineLine BGA	896-Pin FineLine BGA
EP2C5 *(6)*	89	142		158 *(5)*				
EP2C8 *(6)*	85	138		182				
EP2C20 *(6)*			142	152	315			
EP2C35 *(6)*					322	322	475	
EP2C50 *(6)*					294	294	450	
EP2C70 *(6)*							422	622

Πίνακας 42 Επιλογές των συσκευών της Cyclone 2και τα μέγιστα I/O pin

Οι συσκευές Cyclone 2υποστηρίζουν κάθετες εναλλαγές μέσα στο ίδιο πακέτο (για παράδειγμα, μπορεί να γίνει εναλλαγή μεταξύ των EP2C35,EPC50 και EP2C70 συσκευών μέσα στο πακέτο 672 pinFineLineBGA). Η εξαίρεση στην υποστήριξη της κάθετης εναλλαγής στην οικογένεια της Cyclone 2 αναφέρεται στον Πίνακας 43.

Table 1–3. Total Number of Non-Migratable I/O Pins for Cyclone II Vertical Migration Paths

Vertical Migration Path	144-Pin TQFP	208-Pin PQFP	256-Pin FineLine BGA *(1)*	484-Pin FineLine BGA *(2)*	484-Pin Ultra FineLine BGA	672-Pin FineLine BGA *(3)*
EP2C5 to EP2C8	4	4	1 *(4)*			
EP2C8 to EP2C20			30			
EP2C20 to EP2C35				16		
EP2C35 to EP2C50				28	*(5)*	28
EP2C50 to EP2C70						28

Πίνακας 43 Η εξαίρεση στην υποστήριξη της κάθετης εναλλαγής στην οικογένεια της Cyclone 2

Όταν μετακινείσαι από μια πυκνότητα σε μια μεγαλύτερη πυκνότητα, I/O pins συχνά χάνονται λόγω του μεγάλου αριθμού των pin της τάσης και της γείωσης που χρειάζονται για να υποστηριχτεί η επιπλέων λογική στην μεγαλύτερη συσκευή. Η εναλλαγή των I/O pin μεταξύ των πυκνοτήτων πρέπει να ξεκαθαριστεί χρησιμοποιώντας το pin-out της συσκευής που είναι σχεδιασμένο για όλες τις πυκνότητες ενός δεδομένου τύπου πακέτου για να αναγνωριστεί πια I/O pin μπορούν να μετατραπούν.

Για να εξασφαλιστεί ότι υποστηρίζονται εναλλασσόμενες πυκνότητες μέσα στο πακέτο, ενεργοποιούμε την κάθετη εφαρμόσιμη εναλλαγή μέσω του Quartus 2λογισμικού (πάμε στο Assignmentsμενού, μετά Device, μετά κάνουμε κλικ στο MigrationDevices). Μετά την ολοκλήρωση, ελέγχουμε το μήνυμα για την ολοκληρωμένη λίσταI/O,DQ,LVDS και άλλων pinπου δεν είναι διαθέσιμα λόγω του επιλεγμένου μονοπατιού εναλλαγής. Ο Πίνακας 43 παραθέτει τα προτερήματα της συσκευής Cyclone 2και δείχνει τον συνολικό αριθμό των μη εναλλασσόμενωνI/O Pin καθώς εναλλάσσονται από μια συσκευή πυκνότητας σε μια μεγαλύτερη συσκευή πυκνότητας.

Οι συσκευές της Cyclone 2 είναι διαθέσιμες σε τρεις βαθμίδες ταχυτήτων : -6, -7 και -8 με την -6 να είναι η ταχύτερη. Ο Πίνακας 44 δείχνει τα πλεονεκτήματα των ταχυτήτων των συσκευών της Cyclone 2.

Table 1–4. Cyclone II Device Speed Grades

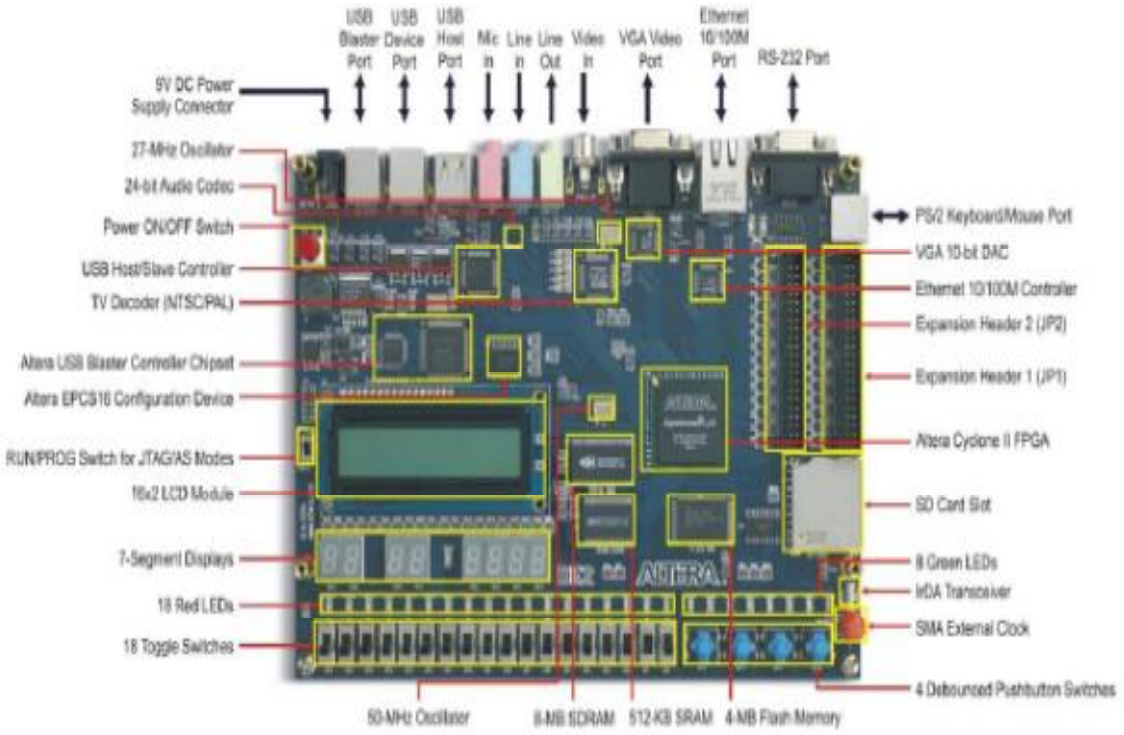
Device	144-Pin TQFP	208-Pin PQFP	240-Pin PQFP	256-Pin FineLine BGA	484-Pin FineLine BGA	484-Pin Ultra FineLine BGA	672-Pin FineLine BGA	896-Pin FineLine BGA
EP2C5	-6, -7, -8	-7, -8		-6, -7, -8				
EP2C8	-6, -7, -8	-7, -8		-6, -7, -8				
EP2C20			-8	-6, -7, -8	-6, -7, -8			
EP2C35					-6, -7, -8	-6, -7, -8	-6, -7, -8	
EP2C50					-6, -7, -8	-6, -7, -8	-6, -7, -8	
EP2C70							-6, -7, -8	-6, -7, -8

Πίνακας 44 Οι ταχύτητες των συσκευών της Cyclone 2

4.2. Τεχνικά Χαρακτηριστικά της Πλακέτας DE2

4.2.1. Τα σχεδιαστικά χαρακτηριστικά του DE2 board

Μια φωτογραφία του DE2 boardφαίνεται στο Εικόνα 41. Απεικονίζει τη διάταξη του boardκαι υποδεικνύει την τοποθεσία των συνδέσεων και των στοιχείων.



Εικόνα 41 DE2 Board

Το DE2 Board έχει πολλά χαρακτηριστικά που επιτρέπουν στον χρήστη να εισάγει ένα ευρύ φάσμα από σχεδιασμένα κυκλώματα, από απλά κυκλώματα έως διάφορα ολοκληρωμένα project.

Το παρακάτω hardware παρέχεται στο DE2 board :

• Altera Cyclone® II 2C35 FPGA device

• Altera Serial Configuration device - EPCS16

• USB Blaster (on board) for programming AND user API control; bothJTAG AND Active Serial

(AS) programming modes are supported

• 512-Kbyte SRAM

• 8-Mbyte SDRAM

• 4-Mbyte Flash memory (1 Mbyte on some boards)

• SD Card socket

• 4 pushbutton switches

• 18 toggle switches

• 18 red user LEDs

• 9 green user LEDs

• 50-MHz oscillator AND 27-MHz oscillator for clock sources

• 24-bit CD-quality audio CODEC with line-in, line-out, AND microphone-in jacks

• VGA DAC (10-bit high-speed triple DACs) with VGA-out connector

• TV Decoder (NTSC/PAL) AND TV-in connector

• 10/100 Ethernet Controller with a connector

• USB Host/Slave Controller with USB type A AND type B connectors

• RS-232 transceiver AND 9-pin connector

• PS/2 mouse/keyboard connector

• IrDA transceiver

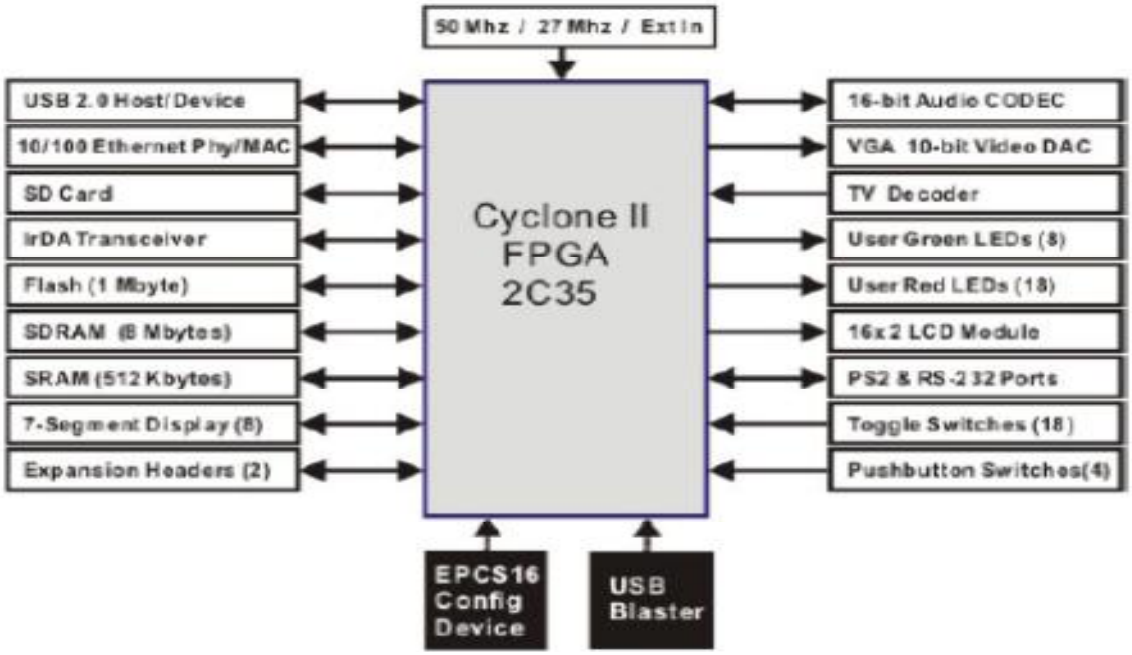
• Two 40-pin Expansion Headers with diode protection

Επιπροσθέτως σε αυτά τα χαρακτηριστικά του hardware, το DE2 boardέχει υποστήριξη λογισμικού για τις τυπικές I/O διαπάλες και μια εγκατάσταση πίνακα ελέγχου για την πρόσβαση σε διάφορα εξαρτήματα. Επίσης, λογισμικό παρέχεται για ένα αριθμό επιδείξεων που παρουσιάζουν τα πλεονεκτήματα των ικανοτήτων του DE2 board.

Για να χρησιμοποιήσει το DE2 board, ο χρήστης πρέπει να είναι εξοικειωμένος με το λογισμικό του Quartus 2.Η απαραίτητη γνώση μπορεί να αποκτηθεί διαβάζοντας τα εκπαιδευτικά GettingStartedwithAltera’s DE2 Board AND Quartus 2 Introduction (το όποιο υπάρχει σε τρεις εκδοχές βασισμένο στη μέθοδο σχεδιασμού που έχει χρησιμοποιηθεί, δηλαδή Verilog,VHDL ή schematicentry). Αυτά τα εκπαιδευτικά παρέχονται στον κατάλογο DE2_tutorials στο DE2 System CD-ROM που συνοδεύει το DE2 board και μπορεί επίσης να βρεθεί στην ιστοσελίδα της Altera DE2.

Block διάγραμμα του DE2 Board

Το Σχήμα 41 μας δίνει το block διάγραμμα του DE2 board. Για την παροχή μεγίστης ευελιξίας στον χρήστη όλες οι συνδέσεις είναι φτιαγμένες μέσα από τη συσκευή Cyclone 2 FPGA.Έτσι, ο χρήστης μπορεί να ρυθμίσει το FPGAγια να εφαρμόσει κάθε σχεδιασμό του συστήματος.



Σχήμα 41 Blockδιάγραμμα τουDE2 board

Παρακάτω υπάρχουν πιο λεπτομερείς πληροφορίες σχετικά με ταblocks στο Σχήμα 41:

Cyclone II 2C35 FPGA

• 33,216 LEs
• 105 M4K RAM blocks
• 483,840 συνολικά RAM bits
• 35 ενσωματωμένους πολλαπλασιαστές
• 4 PLLs
• 475 user I/O pins
• FineLine BGA 672-pin package

Serial Configuration device AND USB Blaster circuit

• Altera'sEPCS16 Σειριακές συσκευές διαμορφώσεις
• On-board USB Blaster για προγραμματισμό και έλεγχο χρηστή API
• JTAGAND AS προγραμματίσιμες μέθοδοι υποστηρίζονται.

SRAM

• 512-Kbyte Static RAM chip μνήμης
• Οργανωμένη ως 256Kx 16 bits
• Πρόσβαση ως μνήμη τουNiosII επεξεργαστή και από τον πίνακα έλεγχου του DE2

SDRAM

• 8-Mbyte Single Data Rate Σύγχρονης δυναμικής RAM chip μνήμης
• Οργανωμένη ως 1Mx 16 bitsx 4 banks
• Πρόσβαση ως μνήμη για το NiosII επεξεργαστή και από τον πίνακα ελέγχου του DE2

Flash memory

• 4-Mbyte NOR Flash μνήμη (1 Mbyte σε μερικά boards)
• 8-bitdatabus
• Προσβάσιμη ως μνήμη για το NiosII επεξεργαστή και από τον πίνακα ελέγχου του DE2

SD card socket

• Παρέχει SPI μορφή για SD Card access

• Προσβάσιμη ως μνήμη για το NiosII επεξεργαστή μαζί με το DE2 SDCardDriver

Pushbuttonswitches

• 4πιεζοδιακοπτες

• Έναρξη από ένα Schmitt κύκλωμα έναυσης

• Normallyhigh; παράγει ένα active-low παλμό όταν ο διακόπτης πατηθεί

Toggleswitches

• 18 toggle διακόπτες για inputs του χρήστη

• Ένας διακόπτης προκαλεί λογικό 0 όταν είναι στην ΚΑΤΩ (πλησιέστερα στην άκρη του DE2 board) θέση

Και λογικό 1 όταν είναι στην ΠΑΝΩ θέση

Clockinputs

• 50-MHz oscillator

• 27-MHz oscillator

• SMA εξωτερική είσοδο ρολογιού

Audio CODEC

• Wolfson WM8731 24-bit sigma-delta audio CODEC

• Line-level είσοδο, line-level έξοδο και είσοδο μικροφώνου

• Sampling frequency: 8 to 96 KHz

• Εφαρμογές για MP3 players AND recorders, PDAs, smart phones, recorders φωνής, κ.τ.λ.

VGA output

• Χρησιμοποιείτο ADV7123 240-MHz τριπλό 10-bit high-speed video DAC

• Με 15-pin υψηλής πυκνότητας D-sub συνδέσεις

• Υποστηρίζει μέχρι και 1600 x 1200 at 100-Hz refresh rate

• Μπορεί να χρησιμοποιηθεί με το Cyclone II FPGA για να ενσωματώσει υψηλής απόδοσηςTV Encoder

NTSC/PAL TV decoder circuit

• Χρησιμοποιεί ADV7181B Multi-format SDTV Video Decoder

• Υποστηρίζει NTSC-(M,J,4.43), PAL-(B/D/G/H/I/M/N), SECAM

• Ενσωματώνει τρεις 54-MHz 9-bit ADCs

• Χρονισμένο από μια27-MH zoscillator είσοδο

• Υποστηρίζει σύνθετο Video (CVBS) RCA jack input.

• Υποστηρίζει ψηφιακή έξοδο(8-bit/16-bit): ITU-RBT.656 YCrCb 4:2:2 output + HS, VS,

AND FIELD

• Εφαρμογές: DVD recorders, LCD TV, Set-top boxes, Digital TV, Φορητές video συσκευές

10/100 Ethernet controller

• Ενσωματωμένο MAC και PHY με μια γενική διεπαφή επεξεργαστή

• Υποστηρίζει 100Base-TAND 10Base-T εφαρμογές

• Υποστηρίζει full-duplex λειτουργία at 10 Mb/s AND 100 Mb/s, with auto-MDIX

• Πλήρης Συμβατότητα με IEEE 802.3u Χαρακτηριστικά

• Υποστηρίζει IP/TCP/UDP checksum generation AND checking

• Υποστηρίζει back-pressure μορφή για half-duplex μορφή ελέγχου ροής

USB Host/Slave controller

• Πλήρης συμβατότητα με Universal Serial Bus Χαρακτηριστικά Rev. 2.0

• Υποστηρίζει μεταφορά δεδομένων full-speed και low-speed

• Υποστηρίζει USB θύρα και συσκευή

• Δυο USB θύρες (μια τύπου Ακαι μια τύπου Β)

• Παρέχει high-speed παράλληλες διεπαφές στους πιο διαθέσιμους επεξεργαστές

Υποστηρίζει NiosII με ένα Terasicdriver
• Υποστηρίζει προγραμματισμένες I/O (PIO) AND Direct Memory Access (DMA)
Serial ports
• ΜιαRS-232 θύρα
• ΜιαPS/2 θύρα
• DB-9 σειριακή σύνδεση για RS-232 θύρα
• PS/2 σύνδεση για την σύνδεση ενός PS2 ποντικιού η πληκτρολογίου στο DE2 board
IrDAtransceiver
• Περιέχει ένα 115.2-kb/s υπέρυθρο πομποδέκτη
• 32 mA LED drive
• ΕνσωματωμένηEMIασπίδα
• ICE825-1 Class 1 eye safe
• Ανίχνευση εισόδου αιχμής
Two 40-pin expansion headers
• 72 Cyclone II I/O pins, όπως επίσης και 8 γραμμές ρεύματος και γείωσης, έρχονται σε δυο 40-pin συνδέσεις επέκτασης
• 40-pin κεφαλή είναι σχεδιασμένη να δέχεται ένα τυπικό 40-pin καλώδιο κορδέλας που χρησιμοποιείται για τους σκληρούς δίσκους IDE
• Προστασία για διόδους και αντιστάσεις παρέχεται.

4.2.2. Ενεργοποίηση του DE2 board

Το DE2 boardέρχεται με προεγκατεστημένη ρύθμιση ροής bit για να επειδίξει κάποια χαρακτηριστικά του board. Αυτή η ροή των bitεπίσης επιτρέπει στον χρήστη να δει γρήγορα αν το board λειτουργεί σωστά. Για την ενεργοποίηση του board εκτελέστε τα παρακάτω βήματα:

- Συνδέστε το παρεχόμενο καλώδιο USBαπό τον υπολογιστή στοUSBblaster στοDE2 board. Για επικοινωνία μεταξύ του υπολογιστή και του DE2 board είναι απαραίτητη η εγκατάσταση του λογισμικού οδήγησης του USB blaster της Altera. Αν αυτός ο οδηγός δεν είναι ήδη εγκατεστημένος στον υπολογιστή μπορεί να εγκατασταθεί όπως εξηγείται στο εκπαιδευτικό GettingStarted with Altera’s DE2 Board. Αυτό το εκπαιδευτικό είναι διαθέσιμο στο DE2 System CD-ROMκαι από την ιστοσελίδα του DE2τηςAltera.

- Συνδέστε τον αντάπτορα 9Vστο DE2 board.

- Συνδέστε μια VGA οθόνη στηνVGAθύρα του DE2 board.

- Συνδέστε τα ακουστικά στη lineoutaudio θύρα τουDE2 board.

- Γυρίστε το run/prog διακόπτη στην αριστερή άκρη τουDE2 boardστη θέσηRun.HPROG θέση χρησιμοποιείται μόνο για AS μορφή προγραμματισμού.

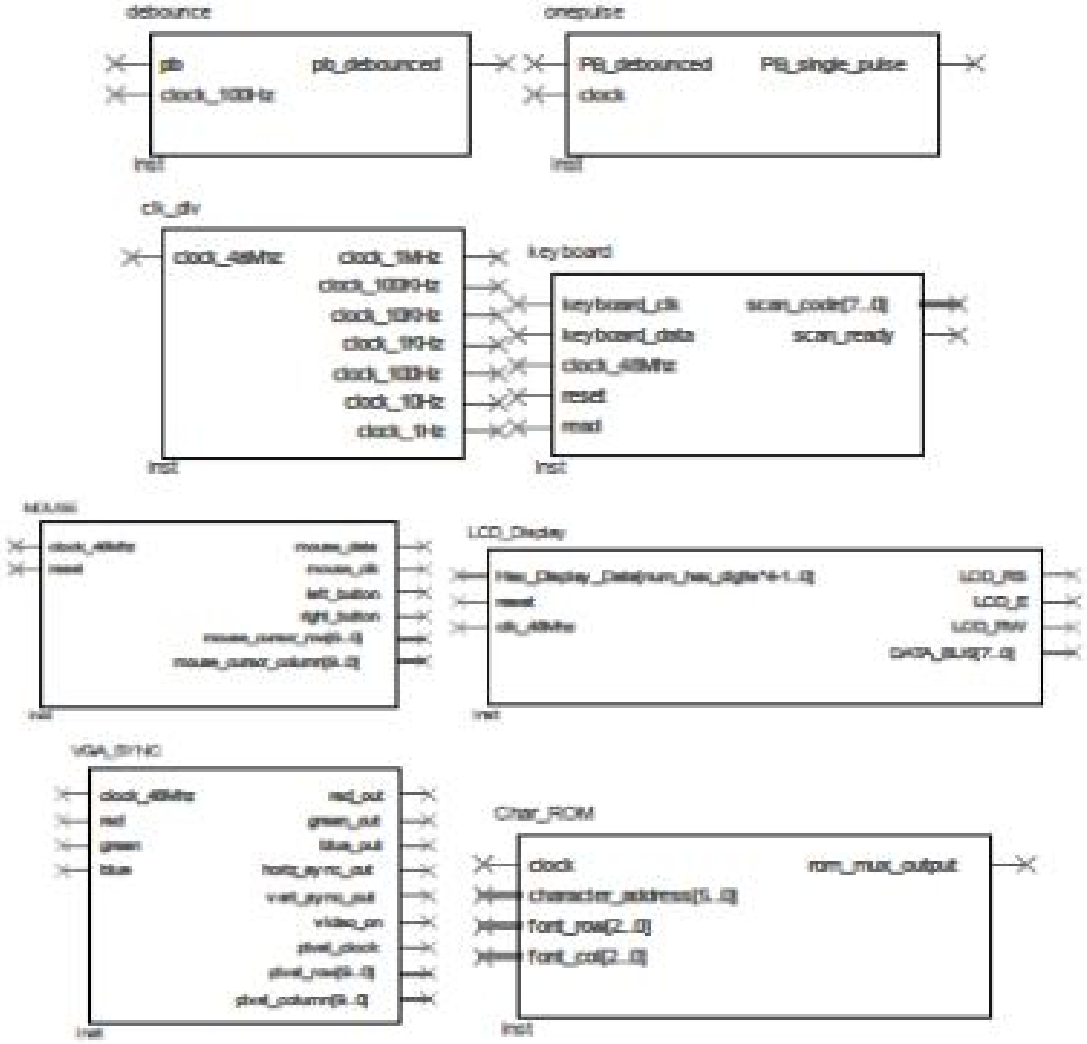
- Βάλτε σε λειτουργία το DE2 boardπιέζοντας τον διακόπτη ON/OFF.

Σε αυτό το σημείο θα πρέπει να παρατηρήσετε τα ακόλουθα :

1. Όλα τα LEDτου χρήστη αναβοσβήνουν.
   - Όλα τα 7-segmentdisplay γυρίζουν από το 0 ως το F.
   - Η οθόνη LCD δείχνει Welcome to Altera DE2 board.

- Η VGA οθόνη δείχνει αυτό που φαίνεται στην
- Εικόνα **42**.
- Ρύθμιση του διακόπτη SW17 στην ΚΑΤΩ θέση. Θα πρέπει να ακούσετε ένα ήχο 1KHz.
- Ρυθμίστε τον διακόπτη SW17 στην ΠΑΝΩ θέση και συνδέστε την έξοδο από ένα audio playerστο line inconnectortουDE2 board.
- Στα ακουστικά σας θα πρέπει να ακούτε την μουσική που παίζει το audio player(MP3,PC,IPod κ.τ.λ.).
- Μπορείτε επίσης να συνδέσετε ένα μικρόφωνο στο microphone in connector του DE2 boardκαι η φωνή σας θα μιξαριστεί με τη μουσική από το audio player.

Εικόνα 42 Η τυπική έξοδο VGA



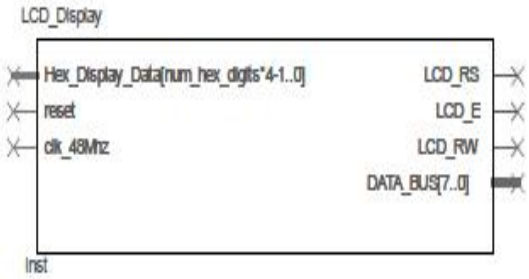
Σχήμα 42 Λειτουργίες βιβλιοθήκης πυρήνα FPGA

Στα σύνθετα ιεραρχικά σχέδια, οι πυρήνες πνευματικής ιδιοκτησίας (IP) χρησιμοποιούνται συχνά. Ένας πυρήνας IP είναι ένα συνδυαστικό σχέδιο υλικού από πριν ανεπτυγμένο που παρέχει μια ευρέως χρησιμοποιημένη λειτουργία. Οι εμπορικά εξουσιοδοτημένοι πυρήνες IP περιλαμβάνουν τις λειτουργίες όπως οι μικροεπεξεργαστές, οι μικροελεγκτές, οι διεπαφές, τα πολυμέσα οι λειτουργίες DSP, και οι ελεγκτές επικοινωνιών. Οι πυρήνες IP προωθούν την επαναχρησιμοποίηση σχεδίου και μειώνουν το χρόνο ανάπτυξης με την παροχή των κοινών λειτουργιών υλικού για τη χρήση σε ένα νέο σχέδιο. Οι λειτουργίες πυρήνα FPGA που απαριθμούνται στον Πίνακας 45 έχουν ως σκοπό να απλοποιήσουν τη χρήση των μπουτόν του πίνακα FPGA, του πληκτρολογίου, του ποντικιού, της οθόνης LCD, των seven segment LEDs και διαθέτει έξοδο βίντεο. Μπορούν να χρησιμοποιηθούν σε σχηματική σύλληψη VHDL, Verilog η με βάση τα σχέδια. Ο πλήρης πηγαίος κώδικας παρέχεται στο DVD.

FPGAcore Name	Description
LCD_Display	Displays ASCII Characters and Hex Data on an LCD Panel
DEC_7SEG	Display Hex Data on a seven-segment LED Display
Debounce	Pushbutton Debounce Circuit
OnePulse	Pushbutton Single Pulse Circuit
Clk_Div	Clock Prescaler with 7 slower frequency outputs (1MHz to 1hz)
VGA_Sync	VGA Sync signal generator for FPGA that outputs pixel addresses
Video_PLL	Used by VGA Sync to generate the video pixel clock using a PLL
Char_ROM	Small Character Font ROM for video character generation
Keyboard	Reads keyboard scan codes from the board's PS/2 connector
Mouse	Reads PS/2 mouse data and outputs cursor row and column address

Πίνακας 45Οι λειτουργίες πυρήνα FPGA

Οι πυρήνες FPGA μπορούν να χρησιμοποιηθούν ως σύμβολα από τη βιβλιοθήκη FPGA, να προσεγγιστούν μέσω μιας συσκευασίας VHDL, ή να χρησιμοποιηθούν ως συστατικό σε άλλα VHDL ή αρχεία Verilog. Η χρήση της συσκευασίας VHDL FPGA pack σώζει την εκ νέου δακτυλογράφηση των μεγάλων συστατικών δηλώσεων για τις λειτουργίες πυρήνων σε κάθε VHDL-βασισμένο σχέδιο. Αυτό το τμήμα περιέχει μια μονοσέλιδη περίληψη κάθε επαφής πυρήνα FPGA. Ο κώδικας πηγής VHDL παρέχεται για όλο το FPGA στο DVD. Για τη σωστή λειτουργία των λειτουργιών πυρήνα FPGA, αντιστοιχία των I/O pin θα πρέπει να γίνεται όπως φαίνεται στην περιγραφή της κάθε λειτουργίας πυρήνα FPGA. Οι εισαγωγές ρολογιών απαιτούνται επίσης σε αρκετές από τις λειτουργίες πυρήνα FPGA. Το Clk_Div FPGA παρέχει τα πιο αργά σήματα ρολογιών που απαιτούνται από μερικές από τις λειτουργίες πυρήνων. Ο πηγαίος κώδικας για τις λειτουργίες FPGA πρέπει να είναι στον κατάλογο προγράμματος ή στην πορεία αναζήτησης βιβλιοθηκών του χρήστη. Οι λειτουργίες πυρήνων VGA_Sync, Video_PLL, και LCD_Display τροποποιούνται συχνά από το χρήστη για να υποστηρίξουν διαφορετικές αναλύσεις οθόνης και επιλογές μηνυμάτων για κάθε σχεδιασμό. Να σιγουρευτείτε ότι θα επιλέξετε τη σωστή έκδοση των εν λόγω λειτουργιών όταν θα κάνετε προσθήκη αρχείων σε ένα νέο έργο.



Σχήμα43FPGAcoreLCD_Display: LCD Panel Character Display

Ο πυρήνας LCD_Display χρησιμοποιείται στους στατικούς χαρακτήρες ASCII επίδειξης και τις μεταβαλλόμενες τιμές δεκαεξαδικού από το υλικό στο DE2 ή το UP3 16 από την επιτροπή επίδειξης 2 γραμμών CD. Ο κώδικας VHDL του πυρήνα μπορεί να διαμορφωθεί εσωτερικά από το χρήστη στους διαφορετικούς τομείς σειρών ASCII επίδειξης και στοιχείων δεκαεξαδικού. Οι οδηγίες μπορούν να βρεθούν στα σχόλια στον κώδικα VHDL του πυρήνα. Ένα γενικό, Num_Hex_Digits, χρησιμοποιείται για τη ρύθμιση του μεγέθους της εισόδου Hex_Display_Data (δηλαδή, κάθε δεκαεξαδικό ψηφίο που εμφανίζεται απαιτεί ένα 4-bit σήμα). Το δελτίο ελεγκτών LCD περιέχει τις πληροφορίες για τους χαρακτήρες της γραφικής παράστασης και τις εντολές A. Μια μηχανή χρησιμοποιείται για να στείλει τα στοιχεία και τις εντολές στον ελεγκτή LCD και για να παραγάγει τα απαραίτητα σήματα.

```
COMPONENT LCD_Display
  PORT(Hex_Display_Data:   IN    STD_LOGIC_VECTOR
                           ((Num_Hex_Digits*4)-1 DOWNTO 0);
       reset, clock_48MHz: IN STD_LOGIC;
       LCD_RS, LCD_E:      OUT    STD_LOGIC;
       LCD_RW:             INOUT STD_LOGIC;
       DATA_BUS:           INOUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END COMPONENT;
```

Σχήμα44VHDLComponentDeclaration

Inputs

Το Hex_Display_Data περιέχει τιμές των 4-bit των σημάτων του δεκαεξαδικού υλικού για να τα μετατρέψετε σε ASCII hex ψηφία και να τα αποστέλλουν στην οθόνη LED. Γενικά το Num_Hex_Digits, ρυθμίζει το μέγεθος των δεδομένων hex εισόδου. Μπορεί να οριστεί μια αξία σε ένα αρχείο HDL ή με την ανάθεση παραμέτρου ενός block σε μια σχηματική αναπαράσταση. Σε ένα σχηματικό, χρησιμοποιήστε Προβολή **à** εκχώρησης παραμέτρων για να δείτε τη γενική αξία και την ετικέτα παραμέτρων των ιδιοτήτων του συμβόλου για να το θέσετε.

Outputs

Οι έξοδοι ελέγχουν ένα 8-bit tri-state αμφίδρομο δίαυλο δεδομένων στην οθόνη LCD. Οι γραμμές χειραψίας χρησιμοποιούνται για τη μεταφορά δεδομένων ASCII στην οθόνη. Την αντιστοιχία των ακίδων για τη μονάδα LCD για την de2 και UP3 πίνακες παρατίθενται παρακάτω. DE1,UP2, και UP1πίνακες δεν έχουν μια ενσωματωμένη ενότητα επίδειξης LCD.

Pin Name	DE1	DE2	UP3	UP2, UP1	Pin Type	Function of Pin
LCD_E	-	K3	50	-	Output	LCD Enable line
LCD_RW	-	K4	73	-	Output	LCD R/W control line
LCD_RS	-	K1	108	-	Output	LCD Register Select Line
LCD_DATA[0]	-	J1	94	-	Bidir.	LCD Data Bus
LCD_DATA[1]	-	J2	96(133)	-	Bidir.	LCD Data Bus
LCD_DATA[2]	-	H1	98	-	Bidir.	LCD Data Bus
LCD_DATA[3]	-	H2	100	-	Bidir.	LCD Data Bus
LCD_DATA[4]	-	J4	102(108)	-	Bidir.	LCD Data Bus
LCD_DATA[5]	-	J3	104	-	Bidir.	LCD Data Bus
LCD_DATA[6]	-	H4	106	-	Bidir.	LCD Data Bus
LCD_DATA[7]	-	H3	113	-	Bidir.	LCD Data Bus

Πίνακας 46 Αντιστοιχία των ακίδων για τη μονάδα LCD

Σχήμα 45 FPGAcoreDEC_7SEG: Hex to Seven-segmentDecoder

Το FPGA Dec_7seg που παρουσιάζεται στο Σχήμα 45είναι ένα δεκαεξαδικό στον αποκωδικοποιητή επίδειξης seven segment με ενεργά χαμηλά αποτελέσματα. Αυτή η λειτουργία χρησιμοποιείται στους αριθμούς δεκαεξαδικού επίδειξης στις επιδείξεις των οδηγήσεων seven-segment ενός πίνακα FPGA.

```
COMPONENT dec_7seg
        PORT( hex_digit         : IN    STD_LOGIC_VECTOR ( 3 DOWNTO 0 );
              seg_a, seg_b, seg_c,
              seg_d, seg_e, seg_f,
              seg_g             : OUT   STD_LOGIC );
END COMPONENT;
```

Εικόνα 43VHDL Component Declaration

Inputs

Το Hex_digit είναι η δεκαεξαδική αξία 4 bit που αποστέλλεται στην οθόνη led. Το Hex_digit [3] είναι το πιο σημαντικό κομμάτι.

Outputs

Οι κατηγορίες A έως G είναι σε active low και θα πρέπει να συνδεθούν με pins εξόδου στο αντίστοιχο pin σε ένα sevensegment display. Ο Πίνακας 47 απαριθμεί τις αναθέσεις των pin για τις πρώτες δύο επιδείξεις δεκαεξαδικού στους πίνακες FPGA. Ο UP3 δεν περιέχει τις επιδείξεις των sevensegment display’s, αλλά έχει μια ενότητα LCD. Περισσότερες από δύο sevensegment display’s είναι διαθέσιμες μαζί με τα DE1 και DE2,δείτε τα εγχειρίδια χρήσης για μια πλήρη λίστα των pin.

Pin Name	DE1	DE2	UP3	UP2, UP1	Pin Type	Function of Pin
HEX0[0]	J2	AF10	-	6	Output	Seven Segment Display 0 LED Segment A (0=on)
HEX0[1]	J1	AB12	-	7	Output	Seven Segment Display 0 LED Segment B (0=on)
HEX0[2]	H2	AC12	-	8	Output	Seven Segment Display 0 LED Segment C (0=on)
HEX0[3]	H1	AD11	-	9	Output	Seven Segment Display 0 LED Segment D (0=on)
HEX0[4]	F2	AE11	-	11	Output	Seven Segment Display 0 LED Segment E (0=on)
HEX0[5]	F1	V14	-	12	Output	Seven Segment Display 0 LED Segment F (0=on)
HEX0[6]	E2	V13	-	13	Output	Seven Segment Display 0 LED Segment G (0=on)
HEX1[0]	E1	V20	-	17	Output	Seven Segment Display 1 LED Segment A (0=on)
HEX1[1]	H6	V21	-	18	Output	Seven Segment Display 1 LED Segment B (0=on)
HEX1[2]	H5	W21	-	19	Output	Seven Segment Display 1 LED Segment C (0=on)
HEX1[3]	H4	Y22	-	20	Output	Seven Segment Display 1 LED Segment D (0=on)
HEX1[4]	G3	AA24	-	21	Output	Seven Segment Display 1 LED Segment E (0=on)
HEX1[5]	D2	AA23	-	23	Output	Seven Segment Display 1 LED Segment F (0=on)
HEX1[6]	D1	AB24	-	24	Output	Seven Segment Display 1 LED Segment G (0=on)

Πίνακας 47 Αναθέσεις των pin για τις πρώτες δύο επιδείξεις δεκαεξαδικού στους πίνακες FPGA

Σχήμα 46 FPGA Debounce: Pushbutton Debounce

Ο πυρήνας FPGA Debounce που παρουσιάζεται στο Σχήμα 46 είναι ένα κύκλωμα debounce μπουτόν. Αυτή η λειτουργία χρησιμοποιείται για να φιλτράρει τη μηχανική αναπήδηση επαφών σε έναν διακόπτη ή ένα μπουτόν. Ένας κατάλογος μετατόπισης χρησιμοποιείται για να φιλτράρει την αναπήδηση επαφών διακοπτών. Ο κατάλογος μετατόπισης παίρνει αρκετό χρόνο για να χωρίσει κατά διαστήματα τα δείγματα του διακόπτη που εισάγονται και αλλάζει την παραγωγή μόνο αφού τα διάφορα διαδοχικά δείγματα έχουν την ίδια αξία.

```
COMPONENT debounce
        PORT( pb, clock_100Hz       : IN        STD_LOGIC;
              pb_debounced          : OUT       STD_LOGIC;
END COMPONENT;
```

Σχήμα 47 VHDL Component Declaration

Inputs

PB είναι η πρώτη είσοδος μπουτόν. Πρέπει να δεθεί σε ένα pin εισαγωγής που συνδέεται με έναν μπουτόν ή συρόμενο διακόπτη. Το ρολόι είναι ένα σήμα ρολογιού περίπου 100Hz που χρησιμοποιείται για τα εσωτερικά 50ms κυκλώματα φίλτρων debounce διακοπτών. Τα UP3, UP2, και UP1 board μπουτόν δεν είναι debounced στο hardware, αλλά τα DE2 και DE1 boards είναι.

Outputs

PB debounced είναι η debounced εξόδου μπουτόν. Η έξοδος θα παραμείνει low μέχρι το κουμπί να απελευθερωθεί. Αν ένας παλμός πρέπει να έχει περίοδο ίση μόνο με αυτή ενός ρολογιού προσθέστε τη OnePulse λειτουργία στην έξοδο του debounced διακόπτη.

Pin Name	DE1	DE2	UP3	UP2, UP1	Pin Type	Function of Pin
KEY0	R22	G26	48	28 PB1	Input	Pushbutton KEY0 (debounced, 0 = button hit)
KEY1	R21	N23	49	29 PB2	Input	Pushbutton KEY1 (debounced, 0 = button hit)
KEY2	T22	P23	57	-	Input	Pushbutton KEY2 (debounced, 0 = button hit)
KEY3	T21	W26	62	-	Input	Pushbutton KEY3 (debounced, 0 = button hit)

Πίνακας 48 Τα pin διεργασίας των πινάκων FPGA

Σχήμα 48 FPGAcore OnePulse: Pushbutton Single Pulse

Η λειτουργία πυρήνα FPGA OnePulse που φαίνεται στο Σχήμα 48είναι ένα μπουτόν για ένα μόνο παλμό στο κύκλωμα. Η έξοδος από το μπουτόν είναι high για ένα μόνο κύκλο ρολογιού χωρίς να έχει σημασία πόσο καιρό το κουμπί είναι πατημένο. Αυτή η λειτουργία είναι χρήσιμη στις μηχανές που διαβάζουν από εξωτερικές εισόδους μπουτόν. Γενικά, απαιτούνται λιγότερες καταστάσεις όταν είναι γνωστό ότι οι είσοδοι ενεργοποιούνται μόνο για έναν κύκλο ρολογιού. Εσωτερικά, ένα άκρο-πυροδοτούμενο flip - flop χρησιμοποιείται για να χτίσει μια απλή μηχανή καταστάσεων.

```
COMPONENT onepulse
        PORT( PB_debounced, clock  : IN   STD_LOGIC;
              PB_single_pulse      : OUT  STD_LOGIC );
END COMPONENT;
```

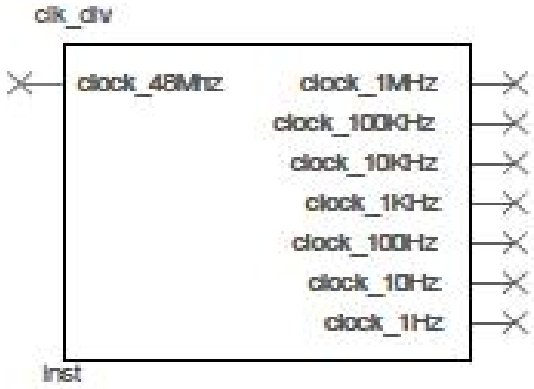
Σχήμα49 VHDL Component Declaration

Inputs

PB debounced είναι το debounced μπουτόν εισόδου. Θα πρέπει να συνδέεται σε ένα debounced μπουτόν. Ρολόι είναι το ρολόι της μηχανής καταστάσεων του χρήστη. Μπορεί να είναι οποιασδήποτε συχνότητας. Σε ορισμένα σχέδια, ο χρήστης μπορεί να θέλει να επεξεργαστεί τον κώδικα VHDL για να προσθέσει μια είσοδο επαναφοράς.

Outputs

PB_single_pulse είναι η έξοδος, η οποία είναι high για ένα μόνο κύκλο ρολογιού όταν πατήσουμε το μπουτόν.



Σχήμα410 FPGA core Clk_Div: Clock Divider

Το FPGA Clk_Div που παρουσιάζεται στο Σχήμα410χρησιμοποιείται για να παρέχει σήματα ρολογιών πιο αργά από τον on–board ταλαντωτή ρολογιών. Τα σήματα εξόδου λαμβάνονται διαιρώντας το σήμα εισόδου του ρολογιού. Πολλαπλές έξοδοι παρέχουν συχνότητες ρολογιών σε δυνάμεις του δέκα.

```
COMPONENT clk_div
        PORT(   clock_48MHz     : IN    STD_LOGIC;
                clock_1MHz      : OUT   STD_LOGIC;
                clock_100kHz    : OUT   STD_LOGIC;
                clock_10kHz     : OUT   STD_LOGIC;
                clock_1kHz      : OUT   STD_LOGIC;
                clock_100Hz     : OUT   STD_LOGIC;
                clock_10Hz      : OUT   STD_LOGIC;
                clock_1Hz       : OUT   STD_LOGIC );
END COMPONENT;
```

Σχήμα 411 VHDL ComponentDeclaration

Inputs

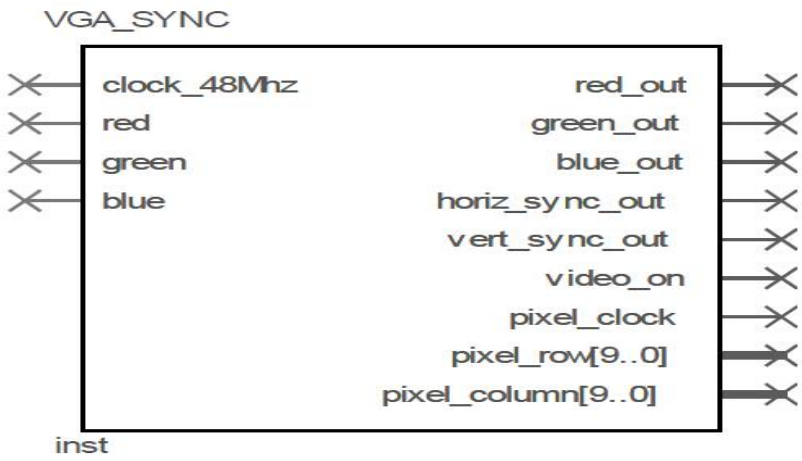
Ένα ρολόι διαφορετικής συχνότητας εισαγωγής χρησιμοποιείται σε διαφορετικά FPGA boards, έτσι κάθε board έχει μια ελαφρώς διαφορετική έκδοση αυτής της λειτουργίας. Η UP3 έκδοση παρουσιάζεται. Το Clock_48MHz είναι ένα pin εισαγωγής που πρέπει να συνδεθείστο UP3 onboardUSB ρολόι 48MHz. Ο αριθμός pin για το UP3 USB ρολόι 48MHz είναι 29.Βεβαιωθείτε ότι το jumperJP3 επιλέγει το 48MHz ρολόιUSB (προεπιλεγμένη ρύθμιση).

Outputs

Ρολόι 1MHz μέχρι 1Hz παρέχει σήματα εξόδου από την καθορισμένη συχνότητα. Με βάση ένα ταλαντωτή κρυστάλλου, η πραγματική συχνότητα είναι 1,007±0,005% φορές την αναγραφόμενη τιμή.

Pin Name	DE1	DE2	UP3	UP2, UP1	Pin Type	Function of Pin
CLOCK	L1 50Mhz	N2 50Mhz	153 48Mhz	91 25Mhz	Input	25-50MHz Clock

Πίνακας 49 Χρονισμοί ωρολογίου



Σχήμα 412 FPGAcore VGA_Sync: παραγωγή βίντεο Sync VGA

Το FPGAcore VGA_Sync που παρουσιάζεται στο Σχήμα 412 παρέχει τα οριζόντια και κάθετα σήματα sync για να παραχθούν τα 8-color640x480 pixel από τη εικόνα βίντεο VGA. Ένας πίνακας της κοινής ανάλυσης της οθόνης και ανανέωσης των ποσοστών μαζί με τα απαραίτητα clock για pixel και τις τιμές αρίθμησης των sync μπορεί να βρεθεί στο τέλος του κώδικα πηγής πυρήνων VGA_Sync IP. Όταν αλλάζουν τα ποσοστά ανανέωσης και κοινής ανάλυσης, πρέπει να χρησιμοποιήσουμε το MegaWizard για να προσαρμόσουμε τον κώδικα video_pll.vhd για την παραγωγή ενός διαφορετικού ποσοστού pixel clock και να αλλάξουμε τα οριζόντια και κάθετα αντίθετα όρια sync στις 6 νέες τιμές που βρίσκονται στον πίνακα.

Τα Video_pll.vhd πρέπει να είναι παρόντα για να συντάξουν VGA_Sync δεδομένου ότι χρησιμοποιεί αυτό το συστατικό για το clock.

```
COMPONENT vga_sync
   PORT( clock_48MHz, red, green, blue: IN   STD_LOGIC;
   red_out, green_out, blue_out,
   horiz_sync_out, vert_sync_out:        OUT STD_LOGIC;
   pixel_row, pixel_column:              OUT STD_LOGIC_VECTOR( 9 DOWNTO 0 ) );
END COMPONENT;
```

Σχήμα 413 VHDL ComponentDeclaration

Inputs

Το clock είναι ένα pin εισαγωγής που πρέπει να συνδεθεί με το on-board clock. Ένας από τους FPGA's Phase Locked Loops (PLL) χρησιμοποιείται για να παραγάγει το ακριβές ποσοστό βίντεο clock που απαιτείται στον κώδικα video_pll.vhd. Τα κόκκινα, πράσινα και μπλε inputs παρέχουν τις πληροφορίες χρώματος για το σήμα του video. Η εξωτερική λογική χρηστών πρέπει να παραγάγει τα RGB σήματα εισαγωγής. Ένα από τα PLLs του FPGA χρησιμοποιείται για να παραγάγει τα απαιτούμενα pixel clock σε όλες τις πλακέτες, εκτός από την UP2 και UP1, οι οποίες παρέχουν μόνο ένα clock 25Mhz για 640x480. Ένα εξωτερικό clock αναφοράς 48 ή 50MHz χρησιμοποιείται από το PLL για το input clock στους άλλους πίνακες FPGA.

Outputs

• Το Horiz_sync είναι ένα output pin που πρέπει να συνδεθεί στο οριζόντιο synctης VGA.

• Το Vert_sync είναι ένα output pin που πρέπει να συνδεθεί στο κάθετο sync της VGA.

• Το Red_out είναι ένα output pin που πρέπει να συνδεθεί στο κόκκινο RGB σήμα.

• Το Green_out είναι ένα output pin που πρέπει να συνδεθεί στο πράσινο RGB σήμα.

• Το Blue_out είναι ένα output pin που πρέπει να συνδεθεί στο μπλε RGB σήμα

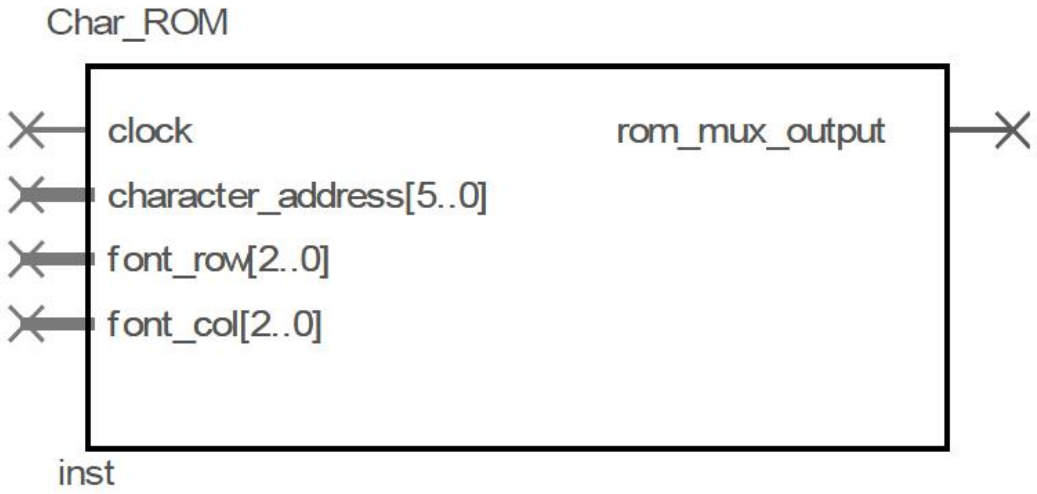
Ένα κύκλωμα διεπαφών των πινάκων UP2 & UP3 μετατρέπει τα ψηφιακά κόκκινα, πράσινα, και μπλε σήματα χρωμάτων βίντεο στην κατάλληλη αναλογική τάση για την οθόνη.

Στο UP3, καθορίστε τονjumperJP3 σε μικρά pin 3-4 για το clock των 48Mhz. Οκτώ χρώματα είναι πιθανόν να χρησιμοποιούνται τρία ψηφιακά σήματα χρώματος. Ένα σήμα 4 bit χρησιμοποιείται στο DE1 για συνολικά 4096 χρώματα και ο DE2 έχει 10 bit ανά χρώμα. Απλά συνδέοντας ένα σήμα σε 2 από τα υψηλά bit χρωμάτων στους πίνακες με περισσότερα χρώματα έχουμε ένα σταθερό χρώμα για απλά σχέδια τα οποία χρησιμοποιούν μόνο λίγα χρώματα.

Pixel_clock,pixel_row και pixel_column είναι outputs που παρέχουν το τρέχον pixel clock και την διεύθυνση pixel. Το Video_on υποδεικνύει ότι τα δεδομένα pixel παρουσιάζονται και ότι δεν συμβαίνει προς το παρόν ένας κύκλος επανεντοπισμού. Αυτά τα outputs χρησιμοποιούνται από λογικούς χρήστες για να παράγουν RGB δεδομένα input χρώματος.

Pin Name	DE1	DE2	UP3	UP2, UP1	Pin Type	Function of Pin
CLOCK	L1 50Mhz	N2 50Mhz	153 48Mhz	91 25Mhz	Input	25-50MHz Clock - PLL input on DEx and UP3
VGA_RED	B7	E10	228	236	Output	VGA Red Video Signal (highest bit)
VGA_GREEN	A8	D12	122	237	Output	VGA Green Video Signal (highest bit)
VGA_BLUE	B10	B12	170	238	Output	VGA Blue Video Signal (highest bit)
VGA_VSYNC	B11	D8	226	239	Output	VGA Connector Vertical Sync Signal
VGA_HSYNC	A11	A7	227	240	Output	VGA Connector Horizontal Sync Signal

Πίνακας 410 Ονόματα ακροδεκτών και οι συναρτήσεις των ακροδεκτών



Σχήμα 414 FPGAcore Char_ROM: ROM παραγωγής χαρακτήρα

Το FPGAcore Char_ROM που παρουσιάζεται στο Σχήμα 414είναι ένα ROM γενιάς χαρακτήρα που χρησιμοποιείται για να παραγάγει το κείμενο σε μια οθόνη βίντεο. Κάθε χαρακτήρας αντιπροσωπεύεται από ένα pixel μεγέθους 8 επί 8. Τα δεδομένα του μεγέθους περιέχονται στο αρχείο εκκίνησης μνήμης tcgrom.mif. Ένας φραγμός μνήμης κυκλώνων M4K απαιτείται για το ROM που περιέχει τα δεδομένα μεγέθους.

```
COMPONENT char_rom
        PORT( clock                 : IN STD_LOGIC;
              character_address     : IN STD_LOGIC_VECTOR( 5 DOWNTO 0 );
              font_row, font_col    : IN STD_LOGIC_VECTOR( 2 DOWNTO 0 );
              rom_mux_output        : OUT STD_LOGIC);
END COMPONENT;
```

Σχήμα 415 VHDL Component Declaration

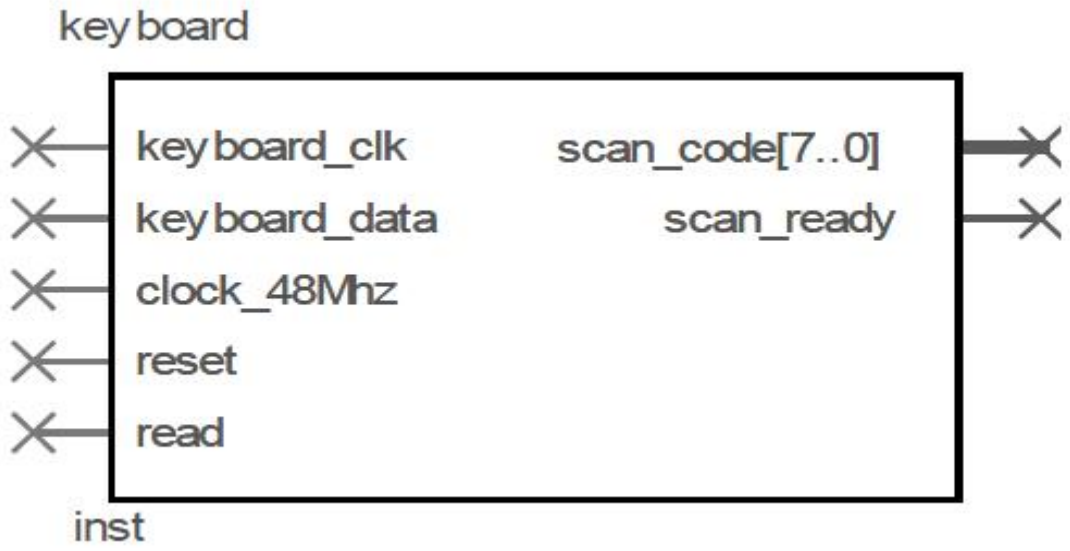
Inputs

Το Character_address είναι η διεύθυνση του αλφανουμερικού χαρακτήρα στην οθόνη. Τα Font_row και font_col χρησιμοποιούνται μέσω ενός μεγέθους8 επί 8 για την διεύθυνση

των μοναδικών pixel που χρειάζονται για την δημιουργία βίντεο σήματος. Τα clock που φορτώνουν την διεύθυνση καταγράφονται και πρέπει να συνδεθούν με τα pixel_clock του βίντεο.

Outputs

Το Rom_mux_output είναι η τιμή του μεγέθους τουpixel που περιέχεται στην διεύθυνση input. Χρησιμοποιείται από λογικούς χρήστες για να παραγάγει τα δεδομένα pixel RGB για το σήμα του βίντεο.



Σχήμα 416 FPGAcore πληκτρολόγιο: Διαβάστε τον κώδικα ανίχνευσης πληκτρολογίου

Το FPGAcore πληκτρολόγιο που παρουσιάζεται στο Σχήμα 416χρησιμοποιείται για να διαβάσει τον κώδικα ανίχνευσης του πληκτρολογίου PS/2από ένα πληκτρολόγιο προσκολλημένο σε αυτό που συνδέει το FPGA του πίνακα PS/2. Αυτή η λειτουργία μετατρέπει τα σειριακά δεδομένα από το πληκτρολόγιο σε ένα παράλληλο format για να παράγει το output του κώδικα ανίχνευσης.

```
COMPONENT keyboard
        PORT( keyboard_clk, keyboard_data, clock_48MHz ,
              reset, read      : IN    STD_LOGIC;
              scan_code        : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0 );
              scan_ready       : OUT STD_LOGIC);
END COMPONENT;
```

Σχήμα 417 VHDL Component Declaration

Inputs

• Το Clock_48MHz είναι ένα input pin το οποίο πρέπει να συνδεθεί με τον ταλαντωτήon-board clock.

• Το Keyboard_clk και το keyboard_data είναι δεδομένα input PS/2 από το πληκτρολόγιο.

• Τα pin διεργασίας των πινάκων FPGA αναγράφονται στον Πίνακας 411.

Η αυξανόμενη άκρη του σήματος ανάγνωσης ξεκαθαρίζει τα έτοιμα σήματα ανίχνευσης. Η επανεκκίνηση είναι ένα input που ξεκαθαρίζει τους εσωτερικούς χρήστες και τις σημαίες που χρησιμοποιούνται για την μετατροπή από σειριακό σε παράλληλο.

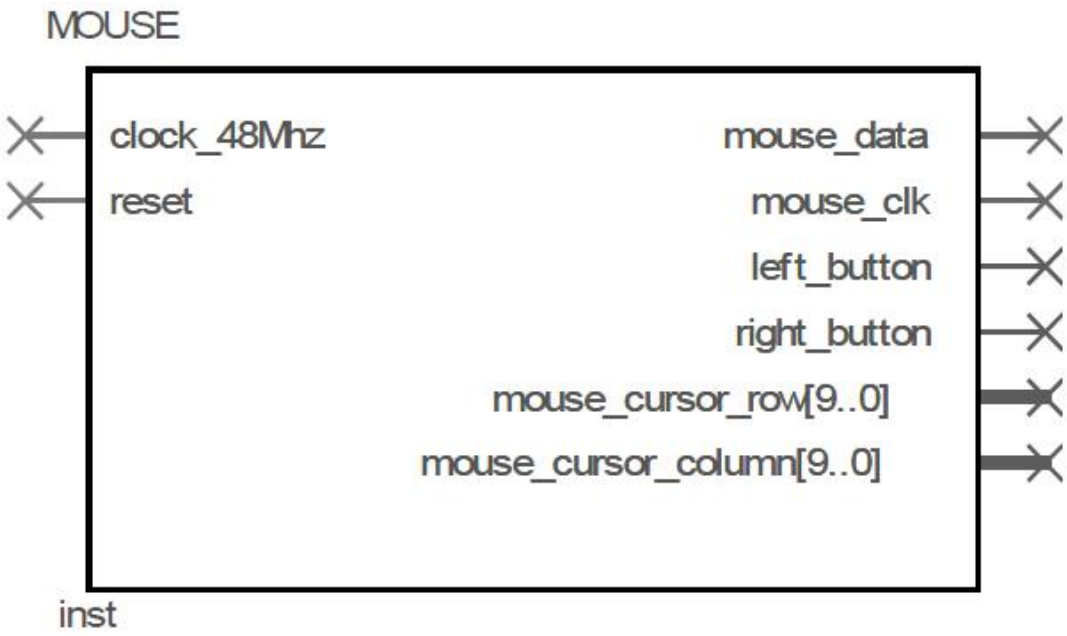
Pin Name	DE1	DE2	UP3	UP2, UP1	Pin Type	Function of Pin
PS2_CLK	H15	D26	12	30	Bidir.	PS2 Connector
PS2_DATA	J14	C24	13	31	Bidir.	PS2 Connector

Πίνακας 411 Τα pin διεργασίας των πινάκων FPGA

Outputs

Το Scan_code περιέχει τα byte που διαβιβάζονται από το πληκτρολόγιο όταν πιέζεται ένα πλήκτρο ή απελευθερώνεται. Οι κώδικες ανίχνευσης για ένα πλήκτρο είναι μια συχνότητα ορισμένων byte. Ένας κώδικας δημιουργίας στέλνεται όταν πατιέται ένα πλήκτρο και ένας κώδικας στέλνεται οποτεδήποτε ένα πλήκτρο απελευθερώνεται. Όταν πληκτρολογούμε είναι φυσιολογικό να έχουμε ορισμένα πλήκτρα στο πληκτρολόγιο πιεσμένα την ίδια στιγμή από διαφορετικά δάκτυλα. Επίσης μερικοί χαρακτήρες απαιτούν το να πιέζεις πολλαπλά πλήκτρα ώστε μερικοί κωδικοί δημιουργίας να μπορούν να ιδωθούν πριν τον κωδικό διακοπής για ένα συγκεκριμένο πλήκτρο.

Το Scan_ready είναι ένα σήμα output το οποίο πηγαίνει στο “High” όταν ένας νέος κωδικός ανίχνευσης στέλνεται από το πληκτρολόγιο. Το input της ανάγνωσης ξεδιαλύνει το scan_ready. Το scan_ready θα έπρεπε να χρησιμοποιείται για να διαβεβαιώνουμε ότι ένας νέος κώδικας ανίχνευσης διαβάζεται μόνο μία φορά.



Σχήμα 418 FPGAcore ποντίκι: Δρομέας Ποντικιού

Το FPGAcore ποντίκι που παρουσιάζεται στο Σχήμα 418 χρησιμοποιείται για να διαβάσει τα δεδομένα της θέσης από ένα ποντίκι προσκολλημένο σε αυτό που συνδέει το PS/2 του UP3. Αυτό βγάζει μια διεύθυνση μιας σειράς και μιας στήλης του κέρσορα για να χρησιμοποιηθεί σε εφαρμογές βίντεο. Το ποντίκι πρέπει να είναι συνδεδεμένο με τον πίνακα FPGA πρωτού το κατεβάσουμε για την σωστή εκκίνηση.

```
COMPONENT mouse
        PORT( clock_48MHz, reset    : IN     STD_LOGIC;
              mouse_data            : INOUT  STD_LOGIC;
              mouse_clk             : INOUT  STD_LOGIC;
              left_button, right_button : OUT  STD_LOGIC;
              mouse_cursor_row      : OUT STD_LOGIC_VECTOR( 9 DOWNTO 0 ) );
              mouse_cursor_column   : OUT STD_LOGIC_VECTOR( 9 DOWNTO 0 ) );
END COMPONENT;
```

Σχήμα 419 VHDL Component Declaration

Inputs

Το Clock_48MHz είναι ένα input pin που πρέπει να είναι συνδεδεμένο με το on-board clock. Στους D1 και D2 πίνακες χρησιμοποιείται ένα clock 50Mhz. Το Mouse_clk και το mouse_data είναι αμφίδρομες γραμμές σήματος PS/2του ποντικιού.

Pin Name	DE1	DE2	UP3	UP2, UP1	Pin Type	Function of Pin
PS2_CLK	H15	D26	12	30	Bidir.	PS2 Clk Connector
PS2_DATA	J14	C24	13	31	Bidir.	PS2 Data Connector

Πίνακας 412 Τα pin διεργασίας των πινάκων FPGA

Outputs

Το Mouse_cursor_row και mouse_cursor_column είναι outputs που περιέχουν την τρέχουσα διεύθυνση του κέρσορα του ποντικιού στην περιοχή της οθόνης με μέγεθος 640 αποτελέσματα που περιέχουν την τρέχουσα διεύθυνση του δρομέα ποντικιών στα 640X480. Ο κέρσορας ξεκινάει στο κέντρο της οθόνης. Το Left_button και right_button outputs είναι “High” όταν το αντίστοιχο κουμπί του ποντικιού πιέζεται.

4.3. Για επιπρόσθετες πληροφορίες

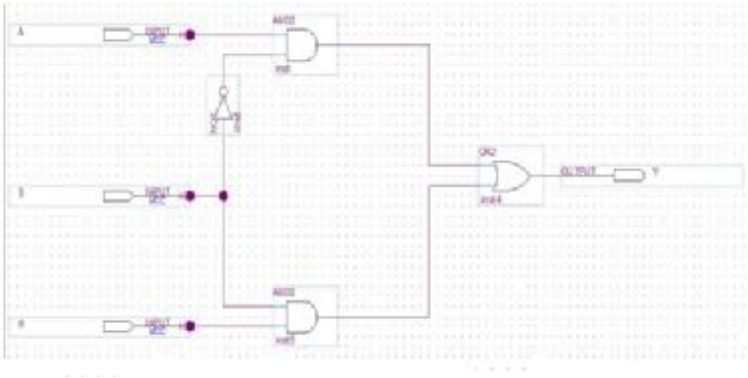
Οι πυρήνες FPGA που συνοψίζονται σε αυτό το κεφάλαιο χρησιμοποιούνται εκτενώς στα παραδείγματα σχεδίου του εγχειριδίου, και συμπληρώνουν τον κώδικα πηγής που παρέχεται στο DVD. Παρέχονται για να υποστηρίξουν οποιαδήποτε νέα σχέδια FPGA που μπορεί να αναπτυχθούν. Ο εκτενείς κατάλογος πιο σύνθετων εμπορικών πυρήνων IP 3 τμημάτων είναι διαθέσιμα και μπορούν να βρεθούν στις σημαντικότερες FPGA ιστοσελίδες,www.altera.com και www.xilinx.com. Το να εκτιμήσουμε τους εμπορικούς πυρήνες μπορεί να είναι ακριβό και η πρόσβαση στο κωδικό πηγής μπορεί να μην παρέχετε. Μια πρόσβαση σε δωρεάν ανοικτή πηγή IP πυρήνων για FPGAs είναι διαθέσιμη στο www.opencores.org.

5. ΑΣΚΗΣΕΙΣ

5.1. ΑΣΚΗΣΗ 1 Απλά συνδυαστικά κυκλώματα.

Το αντικείμενο της άσκησης αυτής που είναι η πρώτη επαφή με το λογισμικό Quartus II της Altera είναι η σχεδίαση και την εξομοίωση με διάφορους τρόπους, απλών ψηφιακών κυκλωμάτων. Μια γενική περιήγηση και μερικές βοήθειες είναι πάντα απαραίτητες σε κάθε νέο πακέτο.

Α) Να σχεδιαστεί με σχηματικά διαγράμματα βασικών ψηφιακών στοιχείων (Graphical User Interface GUI) το ακόλουθο:



Α.1) Μετά τη επιτυχή συμβολομετάφραση (compiler) δημιουργήστε το κατάλληλο περιβάλλον στον Waveform Editor ώστε να γίνει η εξομοίωση του κυκλώματος για όλες τις πιθανές τιμές των εισόδων.

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

Α.2) Δείτε και εξετάσατε την χρονική συμπεριφορά του κυκλώματος και βρείτε την μέγιστη συχνότητα ορθής λειτουργίας.(Timing Analyzer)

Γράψτε εδώ την απάντησή σας:

Α.3) Το πιο πάνω κύκλωμα σε κώδικα περιγραφής υλικού VHDL.

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY example1 IS
        PORT ( A, B, S          : IN     BIT ;
                                Y        : OUT  BIT ) ;
END example1 ;
ARCHITECTURE LogicFunc OF example1 IS
BEGIN
        Y <= (B AND  S) OR (NOT S AND  A) ;
END LogicFunc ;
```

Κώδικας 51 Για τηνασκησης 1.Α

Με την βοήθεια του Text Editor γράψατε τον πιο πάνω κώδικα σε αντίστοιχο αρχείο και μετά την συμβολομετάφραση και εξομοίωση συγκρίνατε τα αποτελέσματα της εξομοίωσης με τους δυο τρόπους δημιουργίας του ψηφιακού κυκλώματος.

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

Γράψτε εδώ την απάντησή σας:

Β) Με αντίστοιχο τρόπο και βήματα σχεδιάστε έναν πολυπλέκτη 4σε1.

Β.1) Να σχεδιαστεί με σχηματικά διαγράμματα βασικών ψηφιακών στοιχείων

(Graphical User Interface GUI) ο πολυπλέκτης 4 σε 1:

Γράψτε εδώ την απάντησή σας:

Σχήμα 2.1

Β.1.α) Μετά τη επιτυχή συμβολομετάφραση (compiler) δημιουργήστε το κατάλληλο περιβάλλον στον Waveform Editor ώστε να γίνει η εξομοίωση του κυκλώματος για όλες τις πιθανές τιμές των εισόδων.

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

Β.1.β) Δείτε και εξετάσατε την χρονική συμπεριφορά του κυκλώματος και βρείτε την μέγιστη συχνότητα ορθής λειτουργίας.(Timing Analyzer)

Γράψτε εδώ την απάντησή σας:

Β.1.γ) Δώστε το κύκλωμα του πολυπλέκτη 4 σε 1 σε κώδικα περιγραφής υλικού VHDL.

Γράψτε εδώ την απάντησή σας:

Κώδικας της Άσκησης 1.Β

Με την βοήθεια του Text Editor γράψατε τον πιο πάνω κώδικα σε αντίστοιχο αρχείο και μετά την συμβολομετάφραση και εξομοίωση συγκρίνατε τα αποτελέσματα της εξομοίωσης με τους δυο τρόπους δημιουργίας του ψηφιακού κυκλώματος.

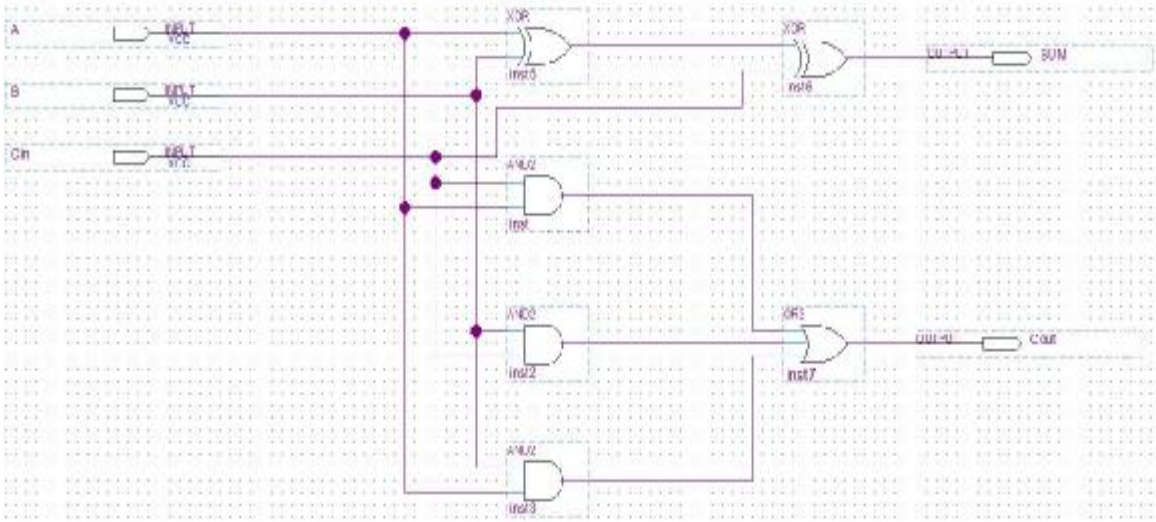
Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

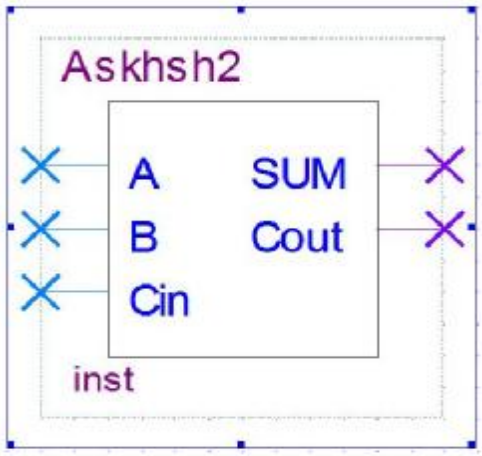
Γράψτε εδώ την απάντησή σας:

5.2. ΑΣΚΗΣΗ 2 Συνδυαστικά κυκλώματα

Α)Πλήρης αθροιστής: Σχεδιάστε με την γνωστή διαδικασία (*Graphical User Interface GUI*)το ακόλουθο κύκλωμα του πλήρη αθροιστή με εισόδους *A,B, Cin,* και εξόδους*SUM* και *Cout.*

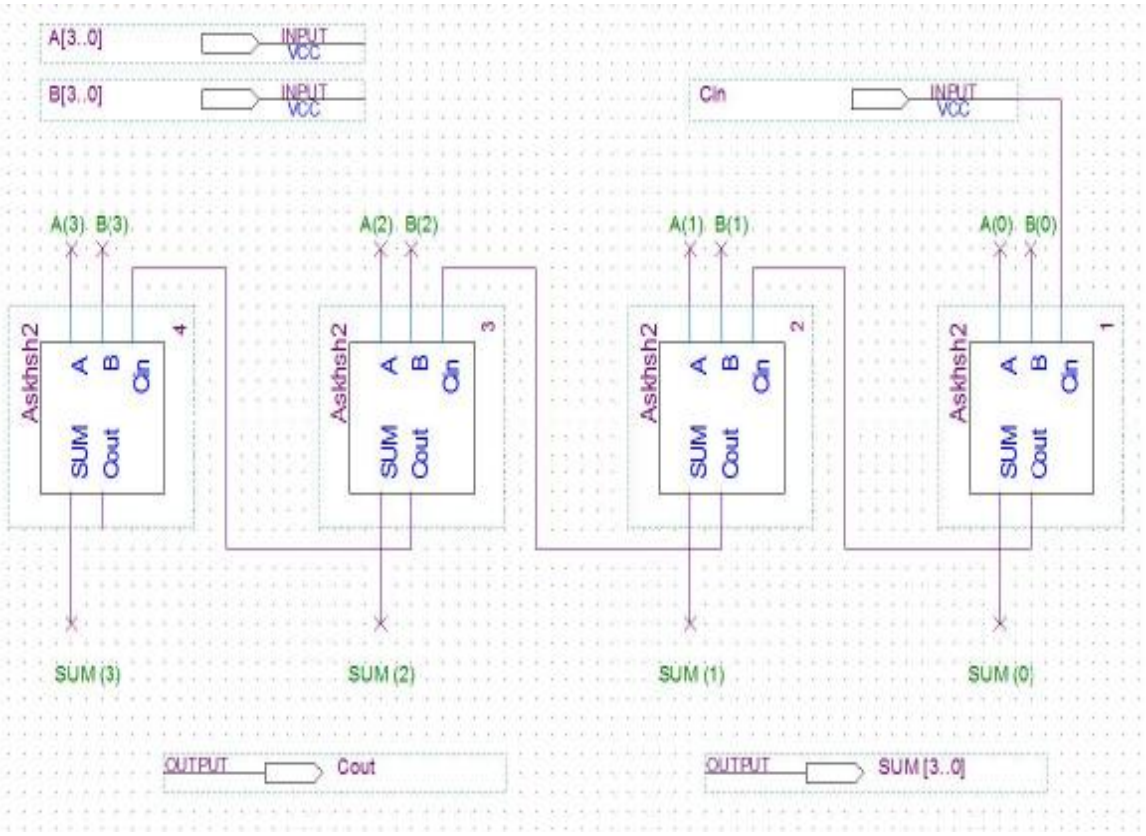


Α.1) Δημιουργήστε ένα σύμβολο για το πιο πάνω κύκλωμα



Β) Παράλληλος 4*bit* αθροιστής: Με τη χρήση του πιο πάνω συμβόλου σχεδιάστε σε επίπεδο χονδρικού διαγράμματος έναν παράλληλο αθροιστή 4 ψηφίων με διάδοση κρατούμενου.

Δώστε προσοχή στην δημιουργία των διαδρόμων εισόδων και εξόδων, *A[3..0], B[3..0], SUM[3..0].*

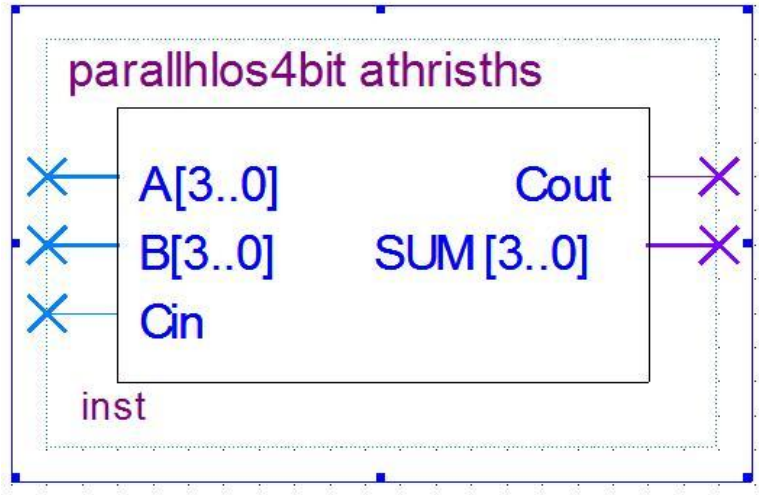


Β.1) Επεξεργαστείτε (*compiler*)και εξομοιώστε το κύκλωμα με αριθμητικά δεδομένα στις εισόδους και ελέγξτε την ορθότητα των αποτελεσμάτων.

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

Β.2) Από το κύκλωμα αυτό του 4 *bit* αθροιστή εύκολα προκύπτει το επόμενο σύμβολο



Γ) Περιγραφή του ιδίου κυκλώματος με *VHDL*. Ο πλήρης αθροιστής:

Γ.1) Δημιουργήστε ένα φάκελο μέσα στον οποίον θα δημιουργήσετε τα διάφορα αρχεία της τρέχουσας σχεδίασης.

Ο κώδικας περιγραφής του πλήρη αθροιστή:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY fulladd IS
PORT ( A, B, Cin      : IN     STD_LOGIC ;
SUM, Cout     : OT     STD_LOGIC ) ;
END fulladd ;
ARCHITECTURE LogicFunc OF fulladd IS
BEGIN
SUM <= A XOR B XOR Cin ;
Cout <= (A AND  B) OR (A AND  Cin) OR (B AND  Cin) ;
END LogicFunc ;
```

Κώδικας 52 Για την Άσκηση 2.Α

Γ.2) Μέσα στον ίδιο φάκελο εργασίας δημιουργήστε ένα αρχείο ***fulladd_package*** το οποίο θα αποτελέσει το εξάρτημα (***component***) για την περιγραφή στη συνέχεια του 4 ***bit*** παράλληλου αθροιστή.

```
        LIBRARYieee ;
        USEieee.std_logic_1164.all ;
        PACKAGEfulladd_packageI
S
        COMPONENTfulladd
        PORT ( A, B, Cin :
INSTD_LOGIC ;
        Sum, Cout: OUT
STD_LOGIC ) ;
        END COMPONENT ;
        END fulladd_package;
```

Κώδικας 53 Για την Άσκηση 2.Β

Γ.3) Ο κώδικας περιγραφής του 4 ***bit*** παράλληλου αθροιστή.

Δώστε προσοχή στην χρήση της βιβλιοθήκης ***work.fulladd*** που περιέχει το ήδη έτοιμο κώδικα του στοιχείου (***component***) του πλήρη αθροιστή :

```
        LIBRARY ieee ;
        USE ieee.std_logic_1164.all ;
        USE work.fulladd_package.all ;
        ENTITY adder IS
        PORT (          Cin     : IN    STD_LOGIC ;
         A, B   : IN    STD_LOGIC_VECTOR(3 DOWNTO 0)
;
        SUM    : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) ;
        Cout    : OUT  STD_LOGIC ) ;
        END adder ;
        ARCHITECTURE Structure OF adder IS
         SIGNAL C : STD_LOGIC_VECTOR(1 TO 3) ;
        BEGIN
        stage0: fulladd PORT MAP ( Cin, A(0), B(0), SUM(0),
```

```
C(1) ) ;

        stage1: fulladd PORT MAP ( C(1), A(1), B(1), SUM(1),
C(2) ) ;

        stage2: fulladd PORT MAP ( C(2), A(2), B(2), SUM(2),
C(3) ) ;

        stage3: fulladd PORT MAP ( C(3), A(3), B(3), SUM(3),
Cout ) ;

        END Structure ;
```

Κώδικας 54 Για την Άσκηση 2.Γ

Γ.4) Επεξεργαστείτε (*compiler*)και εξομοιώστε το κύκλωμα με αριθμητικά δεδομένα στις εισόδους, ελέγξτε την ορθότητα των αποτελεσμάτων και συγκρίνατε την Σχεδίαση με ιεραρχικό πρόγραμμα στην *VHDL*, με την ιεραρχική σχεδίαση με Σχηματικά Διαγράμματα.

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

Γράψτε εδώ την απάντησή σας:

Δ) Με αντίστοιχο τρόπο και βήματα σχεδιάστε έναν παράλληλοαφαιρέτη 4 ψηφίων.

Δ.1). .Σχεδιάστε με την γνωστή διαδικασία (*Graphical User Interface GUI*)το ακόλουθο κύκλωμα του παράλληλουαφαιρέτη 4 ψηφίων.

Γράψτε εδώ την απάντησή σας:

Σχήμα 2.1.1

Δ.1.α) Δημιουργήστε ένα σύμβολο για το πιο πάνω κύκλωμα.

Τοποθετήστε εδώ το σύμβολό σας :

Σχήμα 2.1.2

Με τη χρήση του πιο πάνω συμβόλου σχεδιάστε σε επίπεδο χονδρικού διαγράμματος έναν παράλληλοαφαιρέτη 4 ψηφίων.

Τοποθετήστε εδώ τον παράλληλο αφαιρέτη :

Δ.2.α) Επεξεργαστείτε (*compiler*)και εξομοιώστε το κύκλωμα με αριθμητικά δεδομένα στις εισόδους και ελέγξτε την ορθότητα των αποτελεσμάτων.

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

Δ.2.β) Από το κύκλωμα αυτό προκύπτει το επόμενο σύμβολο :

Τοποθετήστε εδώ το σύμβολό σας :

Σχήμα 2.1.4

Δ.3) Περιγραφή του ιδίου κυκλώματος με *VHDL*: παράλληλος αφαιρέτης 4 ψηφίων

Δ 3.α) Δημιουργήστε ένα φάκελο μέσα στον οποίον θα δημιουργήσετε τα διάφορα αρχεία της τρέχουσας σχεδίασης.

Τοποθετήστε εδώ τον κώδικας περιγραφής του παράλληλου αφαιρέτη 4 ψηφίων:

Κώδικας 2.1.1

Δ 3.β) Μέσα στον ίδιο φάκελο εργασίας δημιουργήστε ένα αρχείο *''παράλληλου αφαιρέτη''*το οποίο θα αποτελέσει το εξάρτημα (***component***) για την περιγραφή του.

Τοποθετήστε εδώ τον κώδικα :

Κώδικας 2.1.2

Δ.3.γ) Ο κώδικας περιγραφής παράλληλου αφαιρέτη 4 ψηφίων.

Δώστε προσοχή στην χρήση της βιβλιοθήκης ***full_minor***που περιέχει το ήδη έτοιμο κώδικα του στοιχείου (***component***) του παράλληλου αφαιρέτη 4 ψηφίων.

Τοποθετήστε εδώ τον κώδικα :

Κώδικας 2.3

Δ.3.δ) Επεξεργαστείτε (***compiler***)και εξομοιώστε το κύκλωμα μεαριθμητικά δεδομένα στις εισόδους, ελέγξτε την ορθότητα των αποτελεσμάτων και συγκρίνατε την Σχεδίαση με ιεραρχικό πρόγραμμα στην *VHDL*, με την ιεραρχική σχεδίαση με Σχηματικά Διαγράμματα.

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

5.3. ΑΣΚΗΣΗ 3 Εφαρμογή στο αναπτυξιακό σύστημα.

5.3.1. Εφαρμογήστο αναπτυξιακό σύστημα.

Σύμφωνα με την καθιερωμένη πρακτική, η ανάλυση και σχεδίαση κάθε ψηφιακού συστήματος δεν έχει τελειώσει, αν το σχεδιαζόμενο σύστημα δεν υλοποιηθεί και λειτουργήσει σε πραγματικές συνθήκες με επιτυχία, σύμφωνα με τις αρχικές προδιαγραφές και τους στόχους της σχεδίασης.

Στην άσκηση αυτή εκτός από τις τελικές ***software*** εξομοιώσεις που θα γίνουν με το Quartus II στο προσωπικό υπολογιστή, οι ρεαλιστικοί, δηλαδή οι σε πραγματική εφαρμογή έλεγχοι ορθής λειτουργίας, θα πρέπει να συνεχιστούν σε ένα ολοκληρωμένο κύκλωμα στο οποίο θα έχει «κατέβει-*Download*» (προγραμματισμός), ο *VHDL* κώδικας του σχεδιαζόμενου κυκλώματος. Την δυνατότητα αυτή προσφέρουν το λογισμικό Quartus II της Altera και η αναπτυξιακή κάρτα *DE2* της *Altera.*

Η αναπτυξιακή κάρτα *DE2* διαθέτει το ολοκληρωμένο κυκλώμα προγραμματιζόμενης λογικής, Cyclone® II 2C35 FPGA. Η κάρτα αυτή διαθέτει επίσης και μια σειρά κυκλωμάτων – εξαρτημάτων υποστήριξης όπως *clock, leds* απεικόνισης,*7segmentdisplay,push – buttons*κλπ.(Εισαγωγή).

Ο συνδυασμός της κάρτας με τους κώδικες προγραμματισμού δικαιολογούν απόλυτα τον όρο «αναπτυξιακό σύστημα» για την κάρτα *DE2*, γιατί δίνουν τη δυνατότητα στον χρήστη να υλοποιεί και να παρακολουθεί την λειτουργία ιδιαίτερα σύνθετων ψηφιακών συστημάτων.

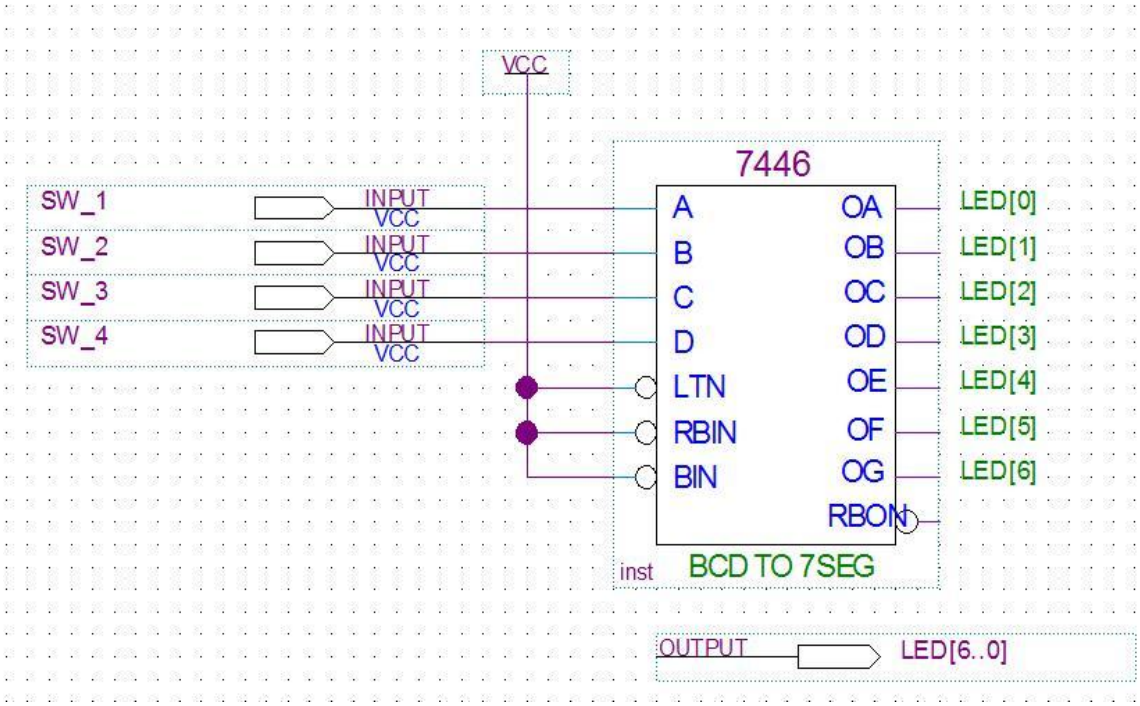
Α) Αποκωδικοποιητής BCD/ 7 segmentdisplay.

Στόχος της παρούσας άσκησης είναι να σχεδιαστεί ένας αποκωδικοποιητής (decoder) BCD/ 7 segment display. Οι είσοδοι του θα συνδεθούν με εξωτερικούς αγωγούς αγωγούς στους 4 διακόπτες (pushbutton switches) του DE2 και οι έξοδοί του στον ενδείκτη υγρών κρυστάλλων (16×2 LCD Module).

Η σχεδίαση του συγκεκριμένου αποκωδικοποιητήBCD/ σε ενδείκτη 7 τμημάτων θα γίνει με την χρήση του σχηματικού διαγράμματος του ολοκληρωμένου κυκλώματος 74246 που ως γνωστό περιέχει έναν decoder BCD/ 7 segment display. Το σχηματικό αυτό διάγραμμα βρίσκεται στην βιβλιοθήκη quartus\libraries\others\maxplus2*.

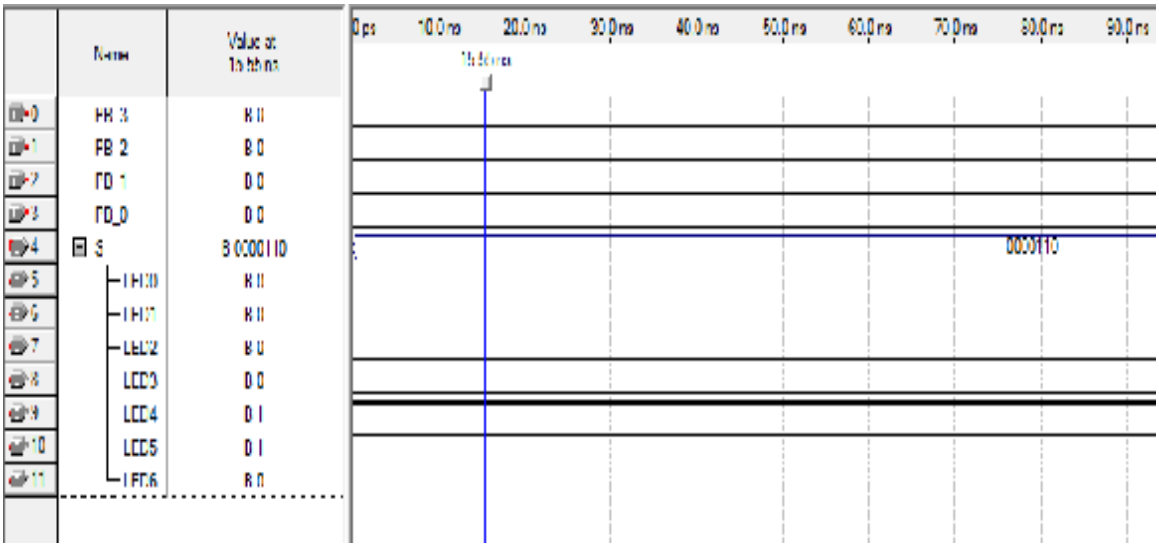
Ο προγραμματισμός του Cyclone® II 2C35 FPGA θα δώσει την δυνατότητα μιας ολοκληρωμένης λειτουργίας αφού κατά τον τελικό έλεγχο του κυκλώματος οι διαφορετικές τιμές των εισόδων θα απεικονίζονται στο ενδείκτη των επτά τμημάτων.

Α.1) Σχεδίαση του κυκλώματος



Σχήμα 3.1

Α.2) Συμβολομετάφραση και εξομοίωση.



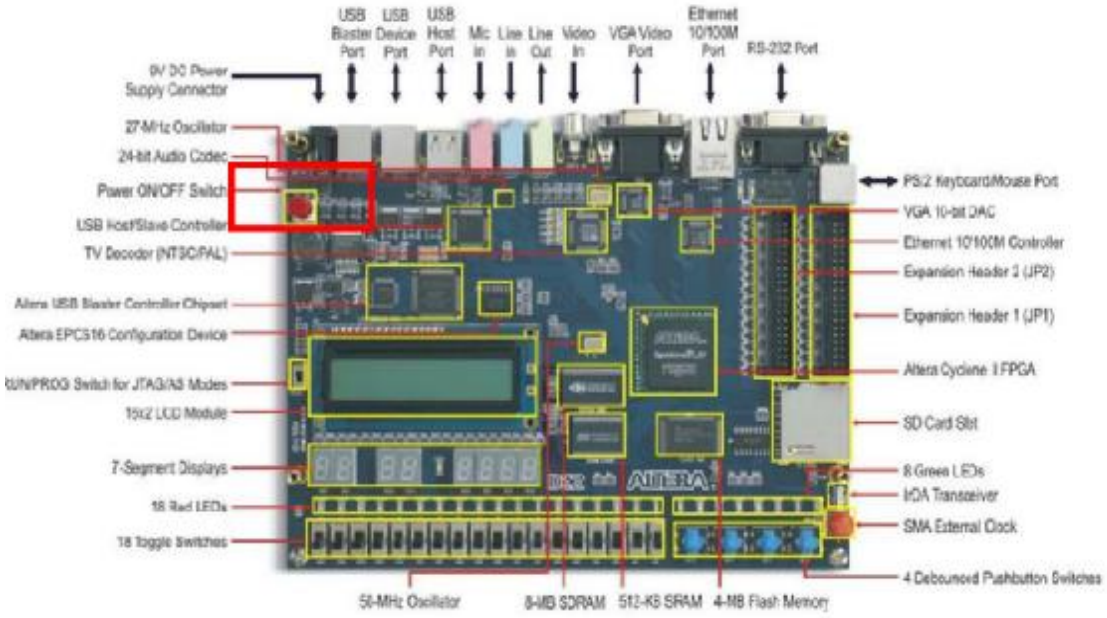
Σχήμα 3.2

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

Α.3) Ορισμός συσκευής:

Παρακάτω εχουμε την φωτογραφία του DE2 board φαίνεται στο σχέδιο 3.3.



Εικόνα 51 DE2 Board

Είναι πολύ σημαντικό να έχουμε τον διακόπτη Run/Prog στην θέση Run και να μην αλλάξουμε την θέση του

Στην επόμενη εικόνα έχουμε την απόδοση των I/O Pins στα εξωτερικά Pins. Στόχος είναι να ελέγξουμε το σχέδιο μας και γι’ αυτό συνδέομαι τα εξωτερικά Pins με τα ενδεικτικά LED. Το DE2 board της Alteraέχει μια σειρά από διακόπτες και δύο σειρές από LEDs διαφορετικού χρώματος τα οποία εμφανίζονται στους παρακάτω πίνακές.

Signal Name	FPGA Pin No.	Description
SW0	PIN_N25	Toggle Switch[0]
SW1	PIN_N26	Toggle Switch[1]
SW2	PIN_P25	Toggle Switch[2]
SW3	PIN_AE14	Toggle Switch[3]
SW4	PIN_AF14	Toggle Switch[4]
SW5	PIN_AD13	Toggle Switch[5]
SW6	PIN_AC13	Toggle Switch[6]
SW7	PIN_C13	Toggle Switch[7]
SW8	PIN_B13	Toggle Switch[8]
SW9	PIN_A13	Toggle Switch[9]
SW10	PIN_N1	Toggle Switch[10]
SW11	PIN_P1	Toggle Switch[11]
SW12	PIN_P2	Toggle Switch[12]
SW13	PIN_T7	Toggle Switch[13]
SW14	PIN_U3	Toggle Switch[14]
SW15	PIN_U4	Toggle Switch[15]
SW16	PIN_V1	Toggle Switch[16]
SW17	PIN_V2	Toggle Switch[17]

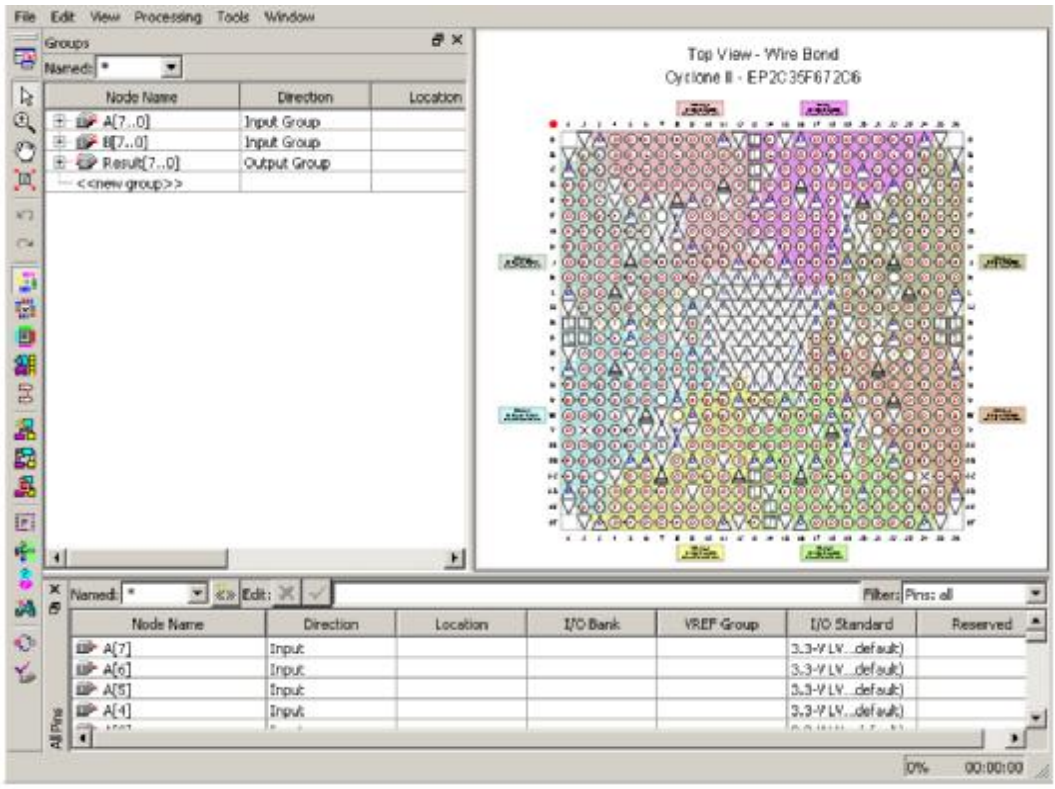
Signal Name	FPGA Pin No.	Description
LEDR0	PIN_AE23	LED Red[0]
LEDR1	PIN_AF23	LED Red[1]
LEDR2	PIN_AB21	LED Red[2]
LEDR3	PIN_AC22	LED Red[3]
LEDR4	PIN_AD22	LED Red[4]
LEDR5	PIN_AD23	LED Red[5]
LEDR6	PIN_AD21	LED Red[6]
LEDR7	PIN_AC21	LED Red[7]
LEDR8	PIN_AA14	LED Red[8]
LEDR9	PIN_Y13	LED Red[9]
LEDR10	PIN_AA13	LED Red[10]
LEDR11	PIN_AC14	LED Red[11]
LEDR12	PIN_AD15	LED Red[12]
LEDR13	PIN_AE15	LED Red[13]
LEDR14	PIN_AF13	LED Red[14]
LEDR15	PIN_AE13	LED Red[15]
LEDR16	PIN_AE12	LED Red[16]
LEDR17	PIN_AD12	LED Red[17]
LEDG0	PIN_AE22	LED Green[0]
LEDG1	PIN_AF22	LED Green[1]
LEDG2	PIN_W19	LED Green[2]
LEDG3	PIN_V18	LED Green[3]
LEDG4	PIN_U18	LED Green[4]
LEDG5	PIN_U17	LED Green[5]
LEDG6	PIN_AA20	LED Green[6]
LEDG7	PIN_Y18	LED Green[7]
LEDG8	PIN_Y12	LED Green[8]

Πίνακας 51 Οι διακόπτες και τα LEDτου DE2 Board

Α.4) Από την τεκμηρίωση της Alteraμπορούμε να τα αντιστοιχίσουμε. Για παράδειγμα στον πίνακα 10 το Pin AE23 αντιστοιχεί στο LEDRed[0].

Βήμα 1Βεβαιωθείτε ότι το αρχείο με το διάγραμμα η την περιγραφή του projectσε VHDL / Verilogείναι ενεργό και περιλαμβάνεται στο project τέλος το project πρέπει να έχει συνταχτεί με επιτυχία.

Βήμα 2Από το μενού Assignments επιλέγουμε Pin Planner και εμφανίζεται η παρακάτω εικόνα 3.4



Εικόνα 52Pin Planner

Μπορούμε να δούμε τις θύρες που ορίζονται από τον σχεδιασμό μας σαν είσοδοι A και B και σαν εξόδους με την ένδειξη Result

Βήμα 3ρυθμίζουμε κατάλληλα τα pin

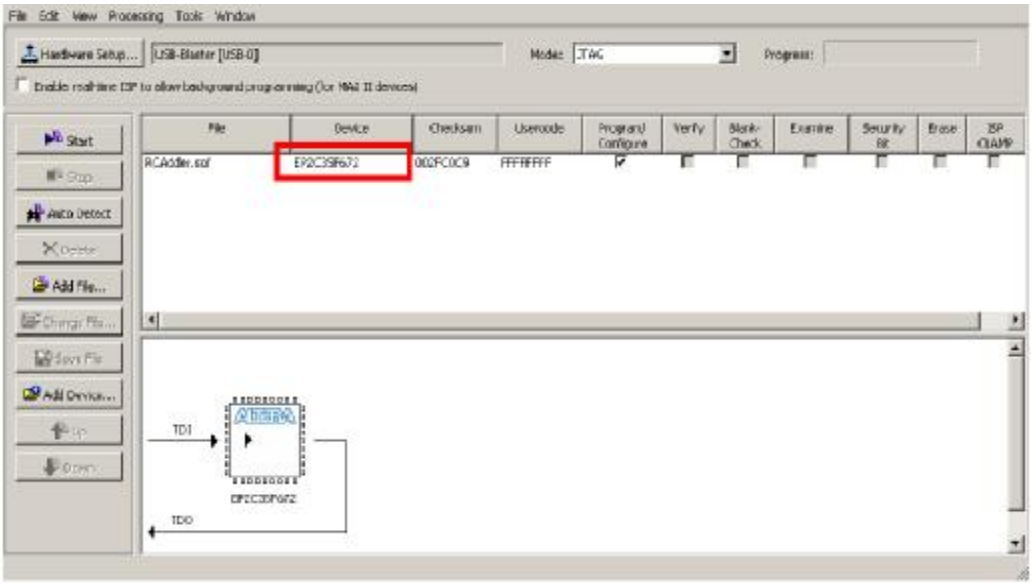
Ονομα κόμβου	**Κατεύθυνση**	**Θέση**
SW_1	Input	PIN_C13
SW_2	Input	PIN_AC13
SW_3	Input	PIN_AD13
SW_4	Input	PIN_AF14
LED[0]	Output	PIN_AC21
LED[1]	Output	PIN_AD21
LED[2]	Output	PIN_AD23
LED[3]	Output	PIN_AD22
LED[4]	Output	PIN_AC22
LED[5]	Output	PIN_AB21
LED[6]	Output	PIN_AF23

Πίνακας 52 Ρύθμιση εισόδων – εξόδων

Α.5) Ορισμός Ολοκληρωμένου Κυκλώματος

Βήμα 1 Από το μενού Assignments → Device ενεργοποιούμε το παράθυρο Settings βεβαιωθείτε ότι το ολοκληρωμένο EP2C35F672C6 είναι επιλεγμένο και κάνουμε click στο OK

Βήμα2 Click στο Tools → Programmer ενεργοποιούμε το παράθυρο programmer:

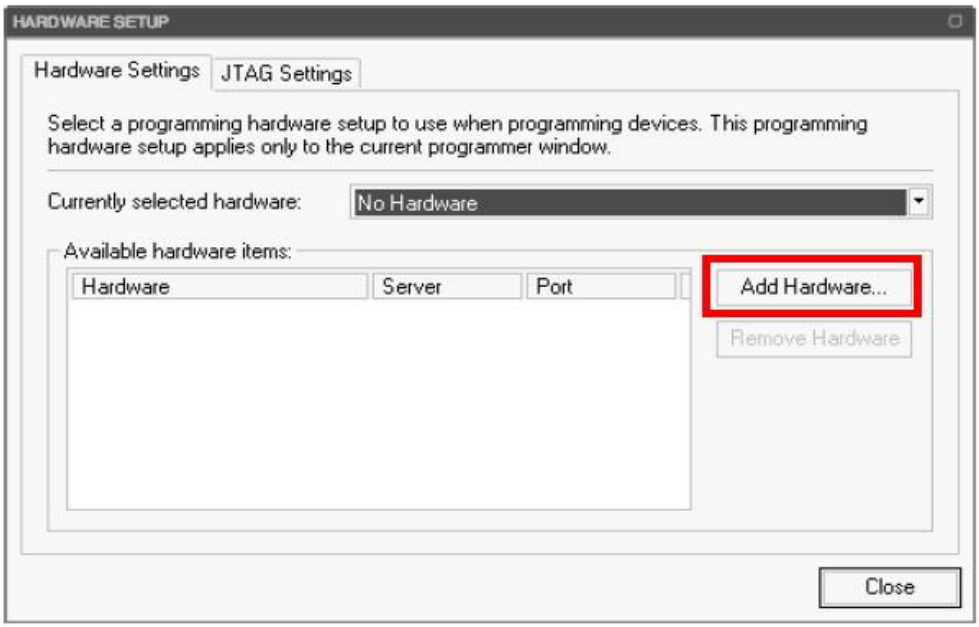


Εικόνα 53 Παράθυρο επιλογής ολοκληρωμένου

Βήμα 3Ενεργοποιήσετε το DE2 board και συνδέσετε USB cable αν το DE2 board δεν έχει δηλωθεί το δηλώνουμε με το Found New Hardware Wizard

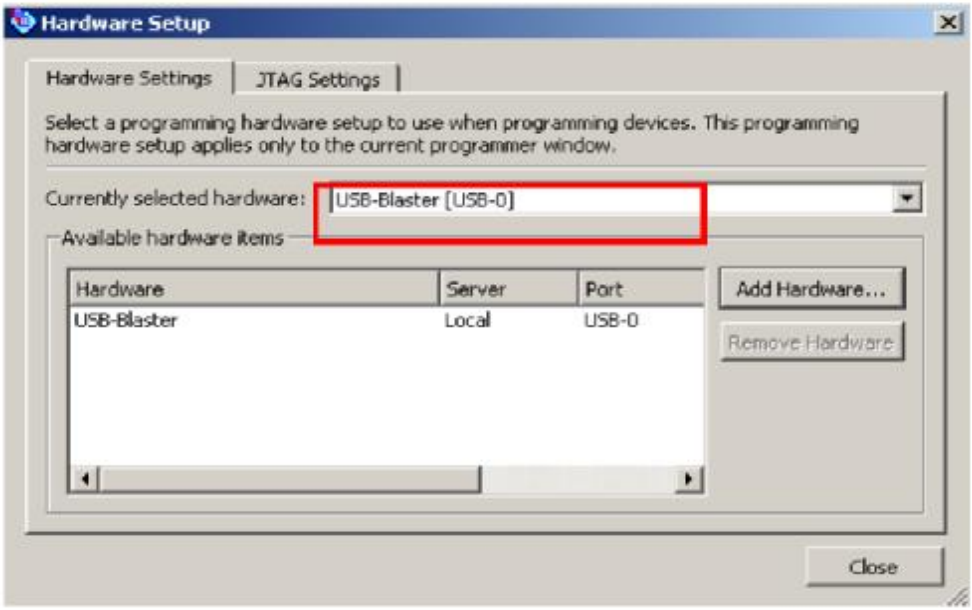
Εικόνα 54 Παράθυρο εύρεσης υλικού

Βήμα4: Click στο Hardware Setup ενεργοποιούμε το παράθυρο Hardware Setup



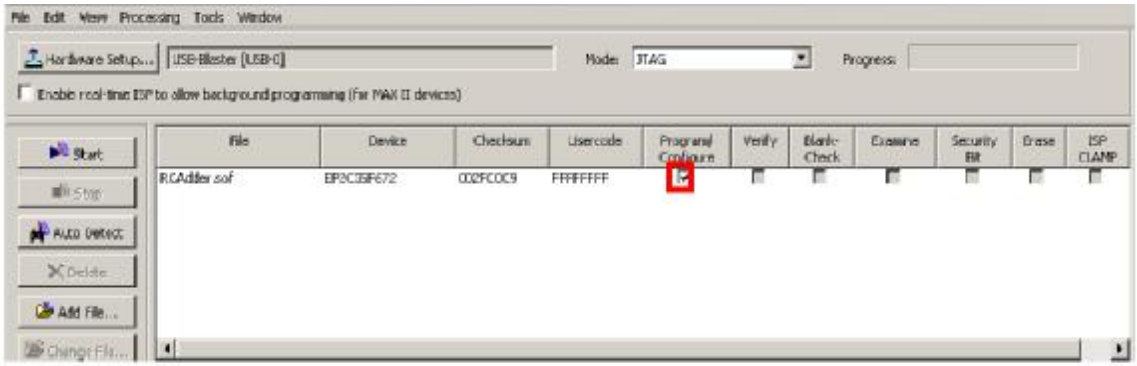
Εικόνα 55 Παράθυρο αρχικοποίησης υλικού

Βεβαιωθείτε ότι είναι επιλεγμένο το υλικόUSB-Blaster (USB-#)



Εικόνα56 Παράθυρο επιλογής υλικού

Click on Program/Configure checkbox:



Εικόνα57 Παράθυρο προγραμματισμού

Βήμα 5: ClickστοStartProgramming:

Εικόνα 58 Μενού προγραμματισμού

Και ξεκινάμε τον προγραματισμό

Γράψτε εδώ την απάντησή σας:

Α.6) Λειτουργία του συστήματος:

Συνδέστε με εξωτερικά μικρά καλώδια τους ακροδέκτες 12,11,9,8 με τους διακόπτες *PUSHBUTTON4, PUSHBUTTON3, PUSHBUTTON2, PUSHBUTTON1* αντίστοιχα.

Δίνοντας όλες τις πιθανές *BCD* τιμές στους παραπάνω διακόπτες ελέγξτε την ορθή απεικόνιση τους στον ενδείκτη υγρών κρυστάλλων (16×2 LCD Module).

Β) Μετά την επιτυχημένη απεικόνιση των *BCD* αριθμών στον ενδείκτη *LSB* του Quartus II εμπλουτίστε την σχεδίαση με τα απαραίτητα κατά την γνώμη σας κυκλώματα ούτως ώστε να απεικονίζονται ταυτόχρονα στον ενδείκτη *MSB* του LCD Module οι αντίστοιχες τιμές του *BCD* κώδικα σε *excess-3*κώδικα.

Ακολουθήστε την προτεινόμενη από το βήμα 1 πορεία.

Β.1.α) Σχεδίαση του κυκλώματος:

Σχήμα 3.1.1

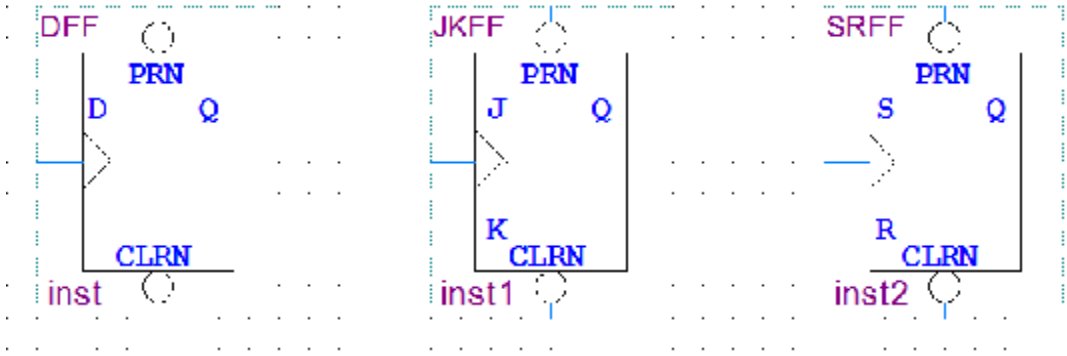
Β.1.β) Συμβολομετάφραση και εξομοίωση.

Σχήμα3.1.2

5.4. ΑΣΚΗΣΗ 4 Απλά ακολουθιακά κυκλώματα.

Τα δομικά στοιχεία των ακολουθιακών κυκλωμάτων είναι οι διάφοροι τύποι των *flip-flop*. Τα *flip-flop* εισάγονται σε κάθε σχεδίαση είτε με σχηματικά διαγράμματα είτε με κώδικα *VHDL*.

Μερικά χαρακτηριστικά σχηματικά παραδείγματα *flip –flop* που υπάρχουν στην βιβλιοθήκη *c:\altera\13.0sp1\quartus\libraries\storage** είναι τα ακόλουθα:



Σχήμα 4.1

Χαρακτηριστικά παραδείγματα *flip –flop*σε κώδικα*VHDL*είναι:

Α) Απλό *Dff*ενεργοποιούμενο σε ανερχόμενο μέτωπο παλμού χρονισμού:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
        PORT ( D, Clock : IN STD_LOGIC ;
        Q : OUT STD_LOGIC ) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
        PROCESS ( Clock )
        BEGIN
                IF Clock'EVENT AND  Clock = '1' THEN
```

```
                Q <= D ;
            END IF ;
        END PROCESS ;
    ENDBehavior ;
```

Κώδικας 55 Κώδικας VHDL για flip –flop

Β) *Dff* με ασύγχρονη είσοδο μηδενισμού (*Reset*):

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
    PORT (      D, Resetn, Clock    : IN    STD_LOGIC ;
                Q                   : OUT   STD_LOGIC ) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
    PROCESS ( Resetn, Clock )
    BEGIN
        IF Resetn = '0' THEN
            Q <= '0' ;
        ELSIF Clock'EVENT AND  Clock = '1' THEN
            Q <= D ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Κώδικας 56 Με ασύγχρονη είσοδο με μηδενισμό

Καταχωρητές,μετρητές όλων των τύπων και δυνατοτήτων αλλά και άλλα ακολουθιακά κυκλώματα είναι δυνατό να εισαχθούν στις διάφορες σχεδιάσεις.

Γ) Καταχωρητής *nbits* με ασύγχρονη είσοδο μηδενισμού (*Reset*):

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY regn IS
        GENERIC ( n : INTEGER := 4 ) ;
        PORT (  D          : IN     STD_LOGIC_VECTOR(n-1 DOWNTO 0) ;
        Resetn, Clock      : IN     STD_LOGIC ;
        Q                           : OUT    STD_LOGIC_VECTOR(n-1 DOWNTO 0) ) ;
END regn ;

ARCHITECTURE Behavior OF regn IS
BEGIN
        PROCESS ( Resetn, Clock )
        BEGIN
                IF Resetn = '0' THEN
                        Q <= (OTHERS => '0') ;
                ELSIF Clock'EVENT AND  Clock = '1' THEN
                        Q <= D ;
                END IF ;
        END PROCESS ;
END Behavior ;
```

Κώδικας 57 Καταχωρητής *n bits* με ασύγχρονη είσοδο μηδενισμού (*Reset*)

Δ) Καταχωρήτης ολίσθησης μήκους τεσσάρων *bits:*

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY shift4 IS
        PORT (  w, Clock           : IN     STD_LOGIC ;
                        Q                  : OUT    STD_LOGIC_VECTOR(1 TO 4) ) ;
END shift4 ;
```

```
ARCHITECTURE Behavior OF shift4 IS
        SIGNAL Sreg : STD_LOGIC_VECTOR(1 TO 4) ;
BEGIN
        PROCESS ( Clock )
        BEGIN
                IF Clock'EVENT AND  Clock = '1' THEN
                        Sreg(4) <= w ;
                        Sreg(3) <= Sreg(4) ;
                        Sreg(2) <= Sreg(3) ;
                        Sreg(1) <= Sreg(2) ;
                END IF ;
        END PROCESS ;
        Q <= Sreg ;
END Behavior ;
```

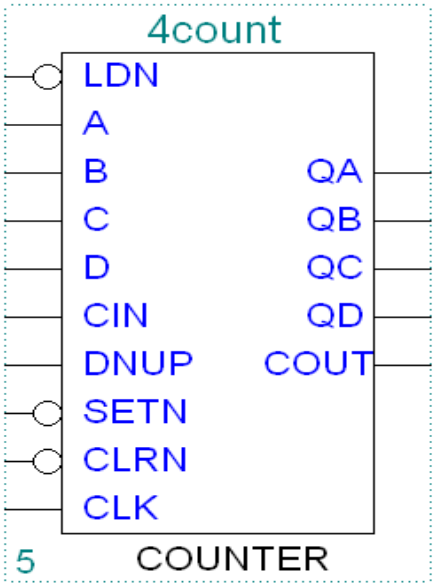
Ε) Μετρητής τεσσάρων *bits* αύξουσας μέτρησης και ασύγχρονου μηδενισμού:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;

ENTITY count4 IS
 PORT (  Resetn          : IN    STD_LOGIC ;
                E, Clock        : IN    STD_LOGIC ;
                        Q       : OUT   STD_LOGIC_VECTOR (3 DOWNTO 0) ) ;
END count4 ;

ARCHITECTURE Behavior OF count4 IS
        SIGNAL Count : STD_LOGIC_VECTOR (3 DOWNTO 0) ;
BEGIN
        PROCESS ( Clock, Resetn )
        BEGIN
                IF Resetn = '0' THEN
```

```
                    Count <= "0000" ;
                ELSIF (Clock'EVENT AND  Clock = '1') THEN
                    IF E = '1' THEN
                        Count <= Count + 1 ;
                    END IF ;
                END IF ;
            END PROCESS ;
            Q<= Count ;
        ENDBehavior ;
```

Κώδικας 58 Καταχωρήτης ολίσθησης μήκους τεσσάρων *bits*

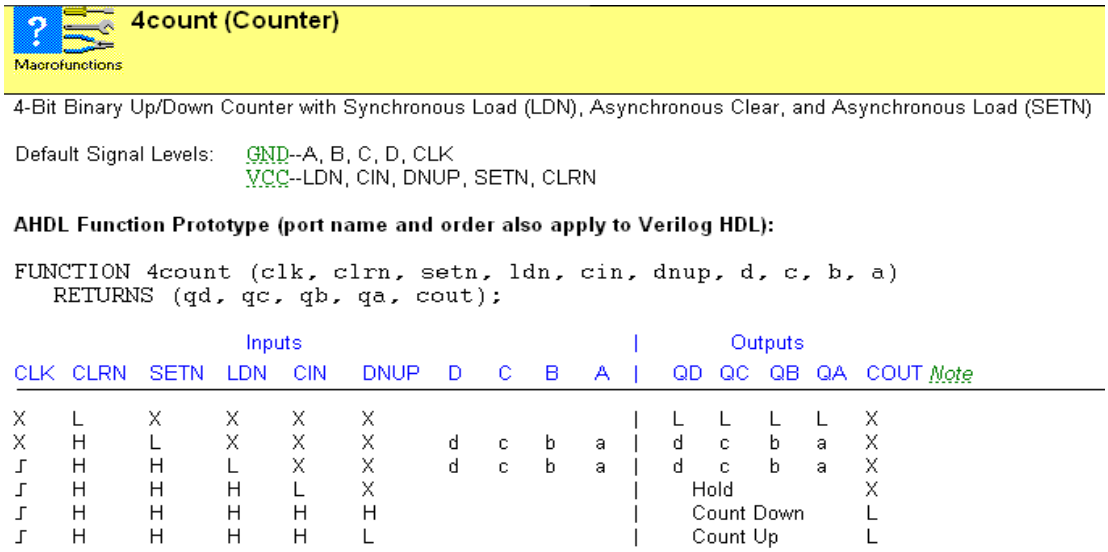
ΣΤ) Μετρητής τεσσάρων *bits* από την βιβλιοθήκη c:\altera\13.0sp1\quartus\libraries\others\maxplus2* με δυνατότητα αύξουσας ή φθίνουσας μέτρησης αλλά και προτοποθέτησης.



Σχήμα 51 Μετρητής τεσσάρων *bits*

Πληροφορίες για τις δυνατότητες και την λειτουργία των διαφόρων κυκλωμάτων από τις βιβλιοθήκες του Quartus II εμφανίζονται με την χρήση του εικονιδίου ? και κλικ

στο σχήμα του κυκλώματος πχ.



4count (Counter)

Macrofunctions

4-Bit Binary Up/Down Counter with Synchronous Load (LDN), Asynchronous Clear, and Asynchronous Load (SETN)

Default Signal Levels: GND--A, B, C, D, CLK
VCC--LDN, CIN, DNUP, SETN, CLRN

AHDL Function Prototype (port name and order also apply to Verilog HDL):

```
FUNCTION 4count (clk, clrn, setn, ldn, cin, dnup, d, c, b, a)
   RETURNS (qd, qc, qb, qa, cout);
```

Inputs										Outputs				
CLK	CLRN	SETN	LDN	CIN	DNUP	D	C	B	A	QD	QC	QB	QA	COUT *Note*
X	L	X	X	X	X					L	L	L	L	X
X	H	L	X	X	X	d	c	b	a	d	c	b	a	X
↑	H	H	L	X	X	d	c	b	a	d	c	b	a	X
↑	H	H	H	L	X					Hold				X
↑	H	H	H	H	H					Count Down				L
↑	H	H	H	H	L					Count Up				L

Εικόνα 59 Παράθυρο με πληροφορίες διαφόρων κυκλωμάτων από τις Βιβλιοθήκες του Quartus II

Β.1) Συντάξτε κώδικα *VHDL* για ένα *Dff* ενεργοποιούμενο σε κατερχόμενο μέτωπο παλμού χρονισμού με ασύγχρονες εισόδους μηδενισμού (*Reset*) και προτοποθέτησης (*Preset*).

Γράψτε εδώ την απάντησή σας:

Συντάξτε κώδικα *VHDL* για έναν καταχωρητή ολίσθησης8 *bits* με δυνατότητα ασύγχρονης προτοποθέτησης.

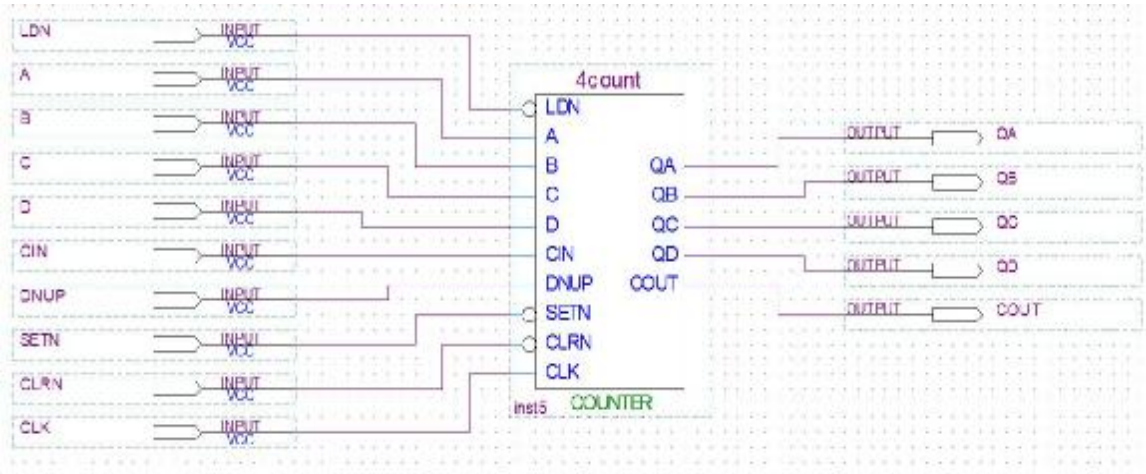
Γράψτε εδώ την απάντησή σας:

Β.2) Με την χρήση του *WaveformEditor* δείξτε την λειτουργία του καταχωρητή.

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

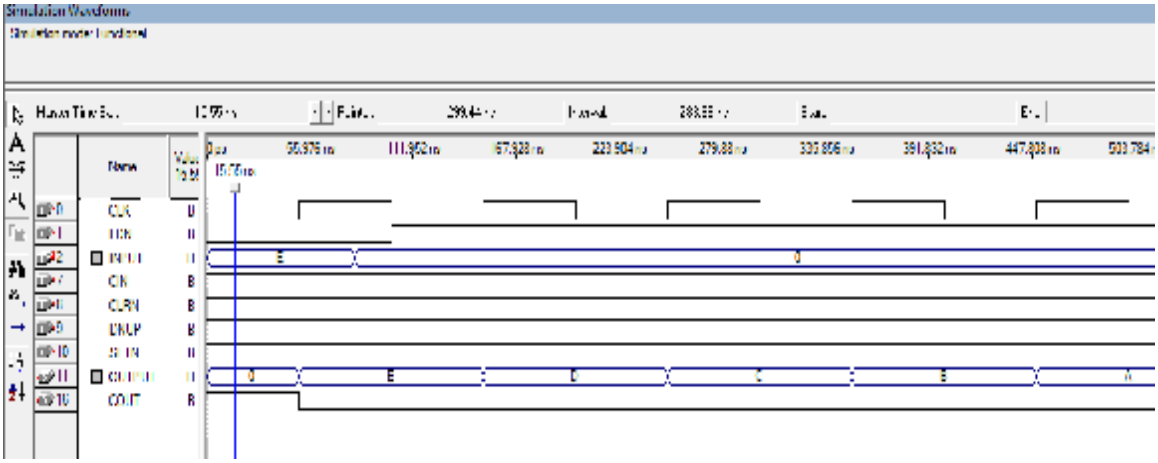
Β.3)Στον μετρητή *4count (Counter)* από την βιβλιοθήκη c:\altera\13.0sp1\quartus\libraries\others\maxplus2* συνδέστε με τον *Graphical User Interface GUI* εισόδους – εξόδους για να δημιουργήσετε το ακόλουθο σχέδιο.



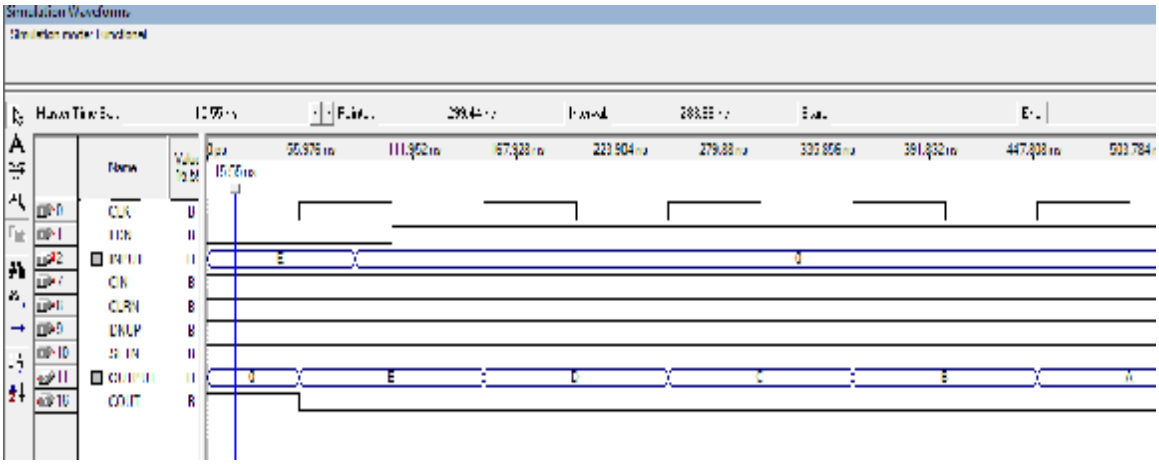
Σχήμα 52 Μετρητής τεσσάρων bits στο Quartus II

Γ.1) Με τον*WaveformEditor* δείξτε την λειτουργία του μετρητή στις ακόλουθες περιπτώσεις

Γ.2) Προτοποθέτηση του αριθμού *Hex*E και φθίνουσα δεκαεξαδική μέτρηση, (πχ.



Σχήμα 53)



Σχήμα 53 Παραθύρο του Waveform Editor

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

Γ.3) Προτοποθέτηση τουαριθμού *Hex*1και αύξουσαδεκαεξαδική μέτρηση.

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

Γ.4)Προτοποθέτηση του αριθμού *Hex*1και αύξουσα δεκαεξαδική μέτρηση, ανά δεύτερο παλμό χρονισμού (Χρήση δυνατότητας *Hold*)

Τοποθετείστε εδώ τις κυματομορφές σας:

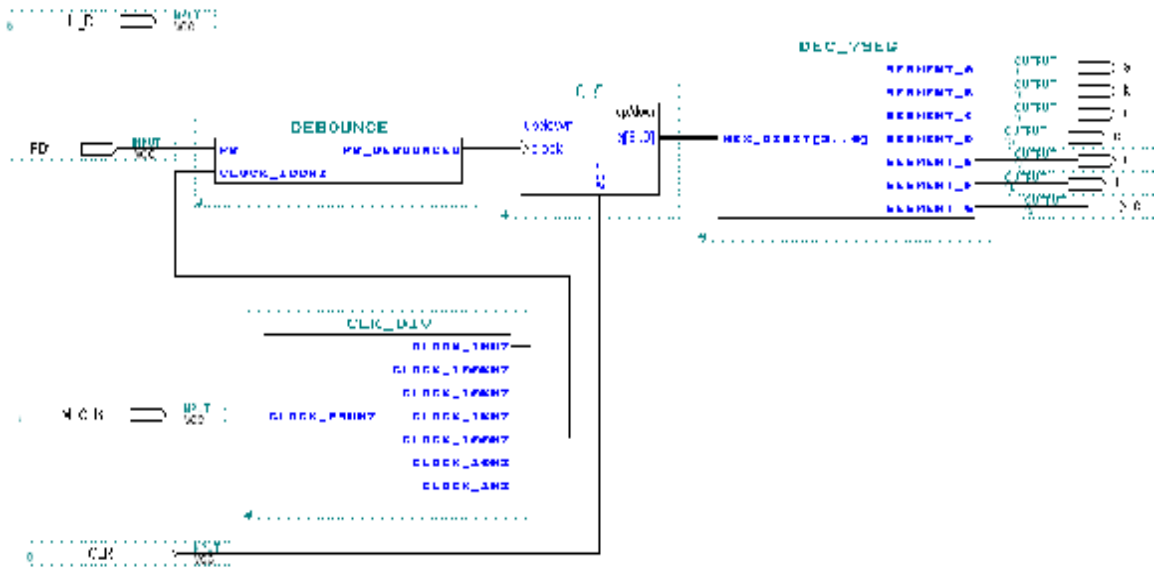
Κυματομορφές της εξομοίωσης

***** Δώστε προσοχή στον πίνακα αλήθειας του μετρητή (Εικόνα 59) και στην ομαδοποίηση των ακροδεκτών εισόδου και εξόδου σε δεκαεξαδική μορφή (*Hex*).

5.5. ΑΣΚΗΣΗ 5 Σύνθετα ακολουθιακά κυκλώματα.

Στην άσκηση αυτή θα σχεδιαστεί ένα **σχετικά** σύνθετο ακολουθιακό κύκλωμα το οποίο θα προγραμματισθεί στο Cyclone® II 2C35 FPGA και στη συνέχεια θα λειτουργήσει σε συνεργασία με τα εξαρτήματα εισόδου – εξόδου της αναπτυξιακής κάρτας *DE2*.Ο πυρήνας της σχεδίασης είναι ένας σύνθετος μετρητής ο οποίος δέχεται σαν είσοδο επεξεργασμένα πατήματα του πλήκτρου *PUSH BUTTON 1 (DebouncePushbutton)*_του οποίου οι έξοδοι του θα επηρεάζουν τους ενδείκτες *7-SEG DIPLAY.*_Με αυτό τον τρόπο θα απαριθμούνται και θα απεικονίζονται στους ενδείκτες τα ‘’πατήματα’’ του πλήκτρου *PUSH BUTTON 1* . Το πλήκτρο *PUSH BUTTON 2*_θα δίνει την δυνατότητα μηδενισμού του μετρητή και η είσοδος *U_D* συνδεδεμένη στον διακόπτη *TOGGLE SWITCH 1*_θα δίνει την δυνατότητα ανερχόμενης ή κατερχόμενης μέτρησης. Η καινοτομία στην άσκηση αυτή είναι η χρήση έτοιμων κυκλωμάτων από τις παραμετροποιημένες βιβλιοθήκες *Megafunctions* του λογισμικού Quartus II καθώς και η χρήση του Cyclone® II 2C35 FPGA για λειτουργική εξομοίωση της σχεδίασης.

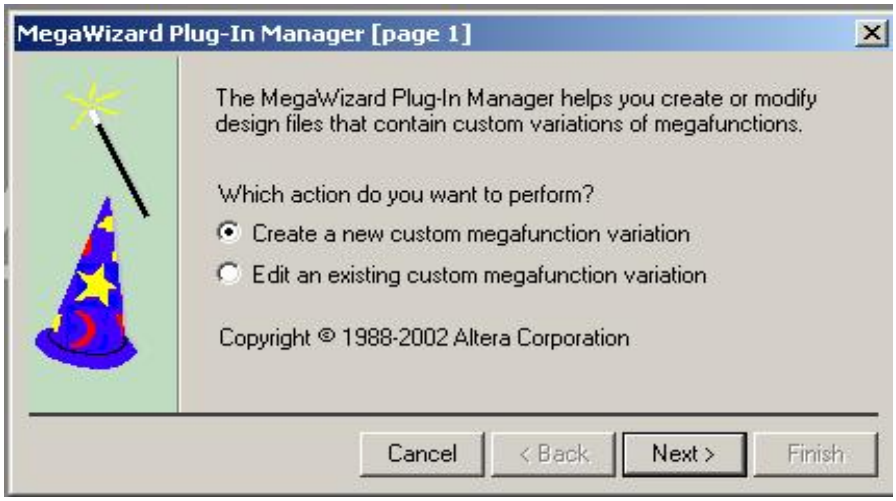
Πιο αναλυτικά η σχεδίαση έχει ως εξής:



Σχήμα 54 σύνθετος μετρητής

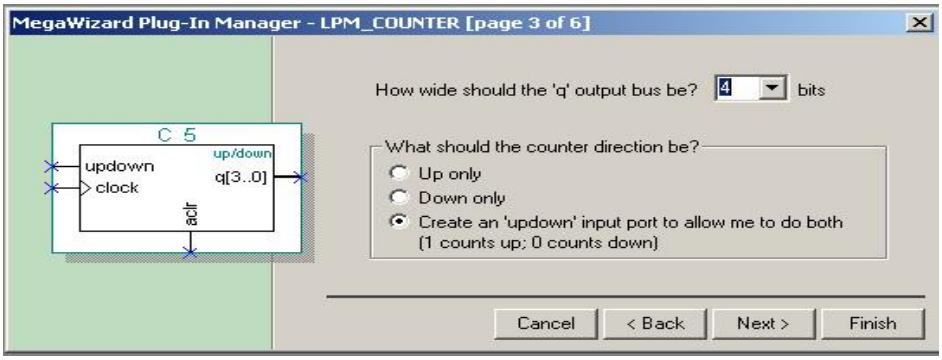
Α.1) Ο πυρήνας της σχεδίασης*C_5* έχει σχεδιαστεί με τον *MegaWizard Plug-In Manager*. Είναι ένας ανερχόμενος / κατερχόμενος μετρητής 4 ψηφίων με δυνατότητα ασύγχρονου μηδενισμού.

Αρχικά από το *FileMenu* επιλέγεταιο*MegaWizard*



Εικόνα 510 Παράθυρο του *MegaWizard*

και ακολουθώντας τα βήματα καταλήγει στην δημιουργία της επιθυμητής μονάδας:



Εικόνα 511 Παράθυρο δημιουργίας επιθυμητής μονάδας

Οι μονάδες DEBOUNCE,CLK_DIVκαιDEC_7SEG, υπάρχουν σε κώδικα VHDLστην βιβλιοθήκηUP1 LIBRARY.

Α.2*Debounce Pushbutton,*Αποφυγή Αναπήδησης Ηλεκτρομηχανικών Διακοπτών.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;

        -- Debounce Pushbutton: Filters out mechanical switch bounce for around 40Ms.
ENTITY debounce IS
        PORT(pb, clock_100Hz    : IN      STD_LOGIC;
                pb_debounced            : OUT    STD_LOGIC);
```

```
END debounce;


ARCHITECTURE a OF debounce IS
	SIGNAL SHIFT_PB		: STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
-- Debounce clock should be approximately 10ms or 100Hz
	PROCESS
	BEGIN
		WAIT UNTIL (clock_100Hz'EVENT) AND  (clock_100Hz = '1');
		-- Use a shift register to filter switch contact bounce
		SHIFT_PB(2 DOWNTO 0) <= SHIFT_PB(3 DOWNTO 1);
		SHIFT_PB(3) <= NOT PB;
		IF SHIFT_PB(3 DOWNTO 0)="0000" THEN
			PB_DEBOUNCED <= '0';
		ELSE
			PB_DEBOUNCED <= '1';
		END IF;
	END PROCESS;
END a;
```

Κώδικας 59*Debounce Pushbutton*

A.3)*Clk_div*Διαίρεση Συχνότητας Χρονισμού

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;


ENTITY clk_div IS


	PORT
	(
		clock_25Mhz				: IN	STD_LOGIC;
```

```
                clock_1MHz                      : OUT   STD_LOGIC;
                clock_100KHz                    : OUT   STD_LOGIC;
                clock_10KHz                     : OUT   STD_LOGIC;
                clock_1KHz                      : OUT   STD_LOGIC;
                clock_100Hz                     : OUT   STD_LOGIC;
                clock_10Hz                      : OUT   STD_LOGIC;
                clock_1Hz                       : OUT   STD_LOGIC);

END clk_div;

ARCHITECTURE a OF clk_div IS

        SIGNAL  count_1Mhz: STD_LOGIC_VECTOR(4 DOWNTO 0);
        SIGNAL  count_100Khz, count_10Khz, count_1Khz : STD_LOGIC_VECTOR(2 DOWNTO 0);
        SIGNAL  count_100hz, count_10hz, count_1hz : STD_LOGIC_VECTOR(2 DOWNTO 0);
        SIGNAL  clock_1Mhz_int, clock_100Khz_int, clock_10Khz_int, clock_1Khz_int: STD_LOGIC;
        SIGNAL  clock_100hz_int, clock_10Hz_int, clock_1Hz_int : STD_LOGIC;
BEGIN
        PROCESS
        BEGIN
-- Divide by 25
                WAIT UNTIL clock_25Mhz'EVENT AND  clock_25Mhz = '1';
                        IF count_1Mhz < 24 THEN
                                count_1Mhz <= count_1Mhz + 1;
                        ELSE
                                count_1Mhz <= "00000";
                        END IF;
                        IF count_1Mhz < 12 THEN
                                clock_1Mhz_int <= '0';
                        ELSE
```

```
                    clock_1Mhz_int <= '1';
                END IF;

        -- Ripple clocks are used in this code to save prescalar hardware
        -- Sync all clock prescalar outputs back to master clock signal
                clock_1Mhz <= clock_1Mhz_int;
                clock_100Khz <= clock_100Khz_int;
                clock_10Khz <= clock_10Khz_int;
                clock_1Khz <= clock_1Khz_int;
                clock_100hz <= clock_100hz_int;
                clock_10hz <= clock_10hz_int;
                clock_1hz <= clock_1hz_int;
    END PROCESS;

        -- Divide by 10
    PROCESS
    BEGIN
        WAIT UNTIL clock_1Mhz_int'EVENT AND  clock_1Mhz_int = '1';
            IF count_100Khz /= 4 THEN
                count_100Khz <= count_100Khz + 1;
            ELSE
                count_100khz <= "000";
                clock_100Khz_int <= NOT clock_100Khz_int;
            END IF;
    END PROCESS;

        -- Divide by 10
    PROCESS
    BEGIN
        WAIT  UNTIL  clock_100Khz_int'EVENT  AND   clock_100Khz_int =
'1';
            IF count_10Khz /= 4 THEN
```

```
                count_10Khz <= count_10Khz + 1;
            ELSE
                count_10khz <= "000";
                clock_10Khz_int <= NOT clock_10Khz_int;
            END IF;
    END PROCESS;

    -- Divide by 10
    PROCESS
    BEGIN
        WAIT UNTIL clock_10Khz_int'EVENT AND  clock_10Khz_int = '1';
            IF count_1Khz /= 4 THEN
                count_1Khz <= count_1Khz + 1;
            ELSE
                count_1khz <= "000";
                clock_1Khz_int <= NOT clock_1Khz_int;
            END IF;
    END PROCESS;

    -- Divide by 10
    PROCESS
    BEGIN
        WAIT UNTIL clock_1Khz_int'EVENT AND  clock_1Khz_int = '1';
            IF count_100hz /= 4 THEN
                count_100hz <= count_100hz + 1;
            ELSE
                count_100hz <= "000";
                clock_100hz_int <= NOT clock_100hz_int;
            END IF;
    END PROCESS;

    -- Divide by 10
```

```
        PROCESS
        BEGIN
                WAIT UNTIL clock_100hz_int'EVENT AND  clock_100hz_int = '1';
                        IF count_10hz /= 4 THEN
                                count_10hz <= count_10hz + 1;
                        ELSE
                                count_10hz <= "000";
                                clock_10hz_int <= NOT clock_10hz_int;
                        END IF;
        END PROCESS;


                -- Divide by 10
        PROCESS
        BEGIN
                WAIT UNTIL clock_10hz_int'EVENT AND  clock_10hz_int = '1';
                        IF count_1hz /= 4 THEN
                                count_1hz <= count_1hz + 1;
                        ELSE
                                count_1hz <= "000";
                                clock_1hz_int <= NOT clock_1hz_int;
                        END IF;
        END PROCESS;
END a;
```

Κώδικας510 Διαίρεση Συχνότητας Χρονισμού

Α.4)Hexadecimal to 7 Segment Decoder for LED Display,Αποκωδικοποιτής Hex /ενδείκτη 7τμήματων.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;


-- Hexadecimal to 7 Segment Decoder for LED Display
```

```
ENTITY dec_7seg IS

        PORT(hex_digit                          :                       IN
STD_LOGIC_VECTOR(3 DOWNTO 0);
                segment_a, segment_b, segment_c, segment_d, segment_e,
                segment_f, segment_g                    : OUT std_logic);
END dec_7seg;

ARCHITECTURE a OF dec_7seg IS
        SIGNAL segment_data : STD_LOGIC_VECTOR(6 DOWNTO 0);

BEGIN
        PROCESS (Hex_digit)
                -- HEX to 7 Segment Decoder for LED Display
        BEGIN
                -- Hex-digit is the four bit binary value to display in hexadecimal
         CASE Hex_digit IS
  WHEN "0000" =>
   segment_data <= "1111110";
  WHEN "0001" =>
   segment_data <= "0110000";
  WHEN "0010" =>
   segment_data <= "1101101";
  WHEN "0011" =>
   segment_data <= "1111001";
  WHEN "0100" =>
   segment_data <= "0110011";
          WHEN "0101" =>
  segment_data <= "1011011";
          WHEN "0110" =>
  segment_data <= "1011111";
```

```
            WHEN "0111" =>
        segment_data <= "1110000";
            WHEN "1000" =>
        segment_data <= "1111111";
            WHEN "1001" =>
        segment_data <= "1111011";
       WHEN "1010" =>
        segment_data <= "1110111";
            WHEN "1011" =>
        segment_data <= "0011111";
           WHEN "1100" =>
        segment_data <= "1001110";
            WHEN "1101" =>
                    segment_data <= "0111101";
            WHEN "1110" =>
             segment_data <= "1001111";
            WHEN "1111" =>
             segment_data <= "1000111";
                    WHEN OTHERS =>
            segment_data <= "0111110";
            END CASE;

           END PROCESS;

                    -- extract segment data bits AND  invert
                    -- LED driver circuit is inverted
           segment_a <=      NOT segment_data(6);
           segment_b <=      NOT segment_data(5);
           segment_c <=      NOT segment_data(4);
           segment_d <=      NOT segment_data(3);
           segment_e <=      NOT segment_data(2);
           segment_f <=      NOT segment_data(1);
```

```
                segment_g <=      NOT segment_data(0);


        END a;
```

Κώδικας 511 Hexadecimal to 7 Segment Decoder for LED Display

Α.5 Εκτελέστε τα βήματα Α.1, Α.2, Α.3, Α.4 ώστε να δημιουργηθούν τα ανάλογα σύμβολα.

Γράψτε εδώ τις απαντήσεις σας:

Α.6)Με τον *SymbolEditor* σχεδιάστε το κύκλωμα όπως στο Σχήμα 54 και κάντε τον απαραίτητο έλεγχο για την ορθή του σύνταξη/σχεδίαση *(compiler).*

Γράψτε εδώ την απάντησή σας:

Α.7) Με την εντολή *Assign* ορίσατε τους ακροδέκτες εισόδου και εξόδου ως ακολούθως:

"CLR" INPUT_PIN= PIN_G26

"M_CLK" INPUT_PIN= PIN_D13

"PB1" INPUT_PIN= PIN_N23

"U_D" INPUT_PIN= PIN_P23

"a" OUTPUT_PIN = PIN_AF10

"b" OUTPUT_PIN = PIN_AB10

"c" OUTPUT_PIN = PIN_AC10

"d" OUTPUT_PIN = PIN_AD10

"e" OUTPUT_PIN = PIN_AE10

"f" OUTPUT_PIN = PIN_V14

"g" OUTPUT_PIN = PIN_V13

Τοποθετείστε εδώ τις κυματομορφές σας:

Τοποθετείστε εδώ τις κυματομορφές σας:

Εικόνα 512 Παράθυρο προγραμματιστή

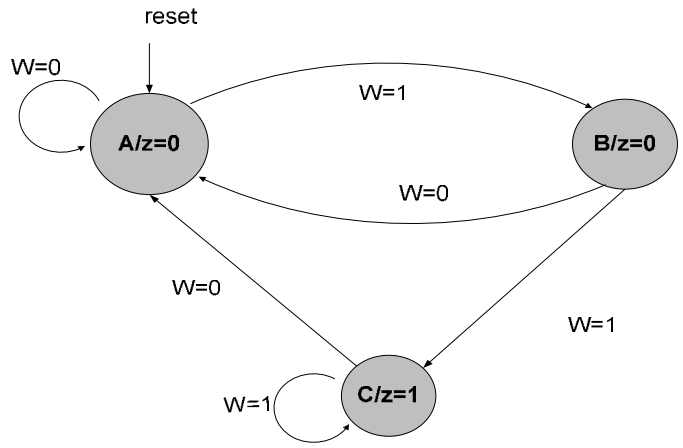
Α.9)Αν όλη η διαδικασία εξελιχθεί σωστά στην αναπτυξιακή κάρτα *DE2* θα λειτουργεί η σχεδίαση με όλες τις δυνατότητες, (*PB1 PB2, FLEXSWITCH 1,LSD_a…LSD_g*)τις οποίες να ελέγξετε και να λειτουργήσετε.

Α.10)Σχεδιάστε έναν σύστημα με δυνατότητα μέτρησης και απεικόνισης έως 99 πατημάτων ηλεκτρομηχανικού πλήκτρου, δυνατότητα ασύγχρονου μηδενισμού και σύγχρονης προτοποθέτησης.

5.6. ΑΣΚΗΣΗ 6Μηχανές καταστάσεων.

Τα κυκλώματα που χαρακτηρίζονται σαν Μηχανές Πεπερασμένων Καταστάσεων *(FSMFiniteStateMachine)* είναι σύγχρονα κυκλώματα που συχνά αποτελούν την μονάδα ελέγχου *(ControlUnit)* μεγαλύτερων σχεδιάσεων. Οι πιο συχνές χρήσεις των αφορούν μηχανές παραγωγής και αναγνώρισης ψηφιολέξεων, μηχανές αναγνώρισης προτύπων, σύνθετοι μετρητές κλπ. Οι πλέον βασικές μορφές *FSM* είναι αυτές τύπου *Moore* και*Mealy*. Στα κυκλώματα τύπου *Moore* η έξοδος ή οι έξοδοι εξαρτώνται αποκλειστικά από την **τρέχουσα** κατάσταση του κυκλώματος ενώ σε αυτά τύπου*Mealy* η έξοδος ή οι έξοδοι εξαρτώνται από την **τρέχουσα** κατάσταση του κυκλώματος και την **τρέχουσα** τιμή της ή των **κυρίων** εισόδων.

Α.1 Ένα χαρακτηριστικό παράδειγμα μοντέλου*Moore*είναι ο ακόλουθος VHDL κώδικας όπου περιγράφει ένα κύκλωμα του οποίου η έξοδοςzγίνεται ίση με ΄1΄ όταν η είσοδος *w* δεχτεί δύο τουλάχιστον διαδοχικά ΄1΄ διαφορετικά έχει την τιμή ΄0΄. Ο κώδικας αυτός αναφέρεται στο ακόλουθο διάγραμμα καταστάσεων.



Σχήμα 55 Διάγραμμα καταστάσεων

Όπου *A,B,C* οι καταστάσεις της μηχανής.

Ο *VHDL* κώδικας:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY MOORE IS
        PORT (  Clock, Resetn, w : IN       STD_LOGIC ;
                                z           : OUT    STD_LOGIC ) ;
END MOORE ;
ARCHITECTURE Behavior OF MOORE IS
TYPE State_type IS (A, B, C) ;
```

```
        SIGNAL y : State_type ;
BEGIN
        PROCESS ( Resetn, Clock )
        BEGIN
                IF Resetn = '0' THEN
                        y <= A ;
                ELSIF (Clock'EVENT AND  Clock = '1') THEN
                        CASE y IS
                                WHEN A =>
                                        IF w = '0' THEN
                                                y <= A ;
                                        ELSE
                                                y <= B ;
                                        END IF ;
                                WHEN B =>
                                        IF w = '0' THEN
                                                y <= A ;
                                        ELSE
                                                y <= C ;
                                        END IF ;
                                WHEN C =>
                                        IF w = '0' THEN
                                                y <= A ;
                                        ELSE
                                                y <= C ;
                                        END IF ;
                        END CASE ;
                END IF ;
        END PROCESS ;
                z <= '1' WHEN y = C ELSE '0' ;
END Behavior ;
```

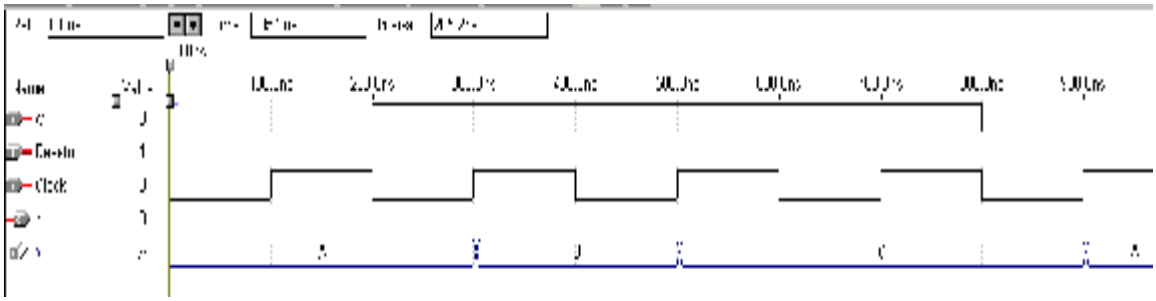
Κώδικας 512 Μηχανή πεπερασμένων καταστάσεων

Δώστε προσοχή στην εντολή δημιουργίας των καταστάσεων

TYPE State_type IS (A, B, C) ; και στην απόδοση της τιμής της εξόδου που εξαρτάται από την κατάσταση της μηχανής**,z <= '1' WHEN y = C ELSE '0'**.

Γράψτε εδώ την απάντησή σας:

Α.2)Εξομοιώσατε τον ανωτέρω κώδικα και με τον WaveformEditor εξετάσατε την ορθή και σύμφωνα με τις προδιαγραφές λειτουργία του (π.χ. Σχήμα 6.2).Δώστε προσοχή στην εναλλαγή των καταστάσεων σύμφωνα με τις προδιαγραφές της μηχανής.

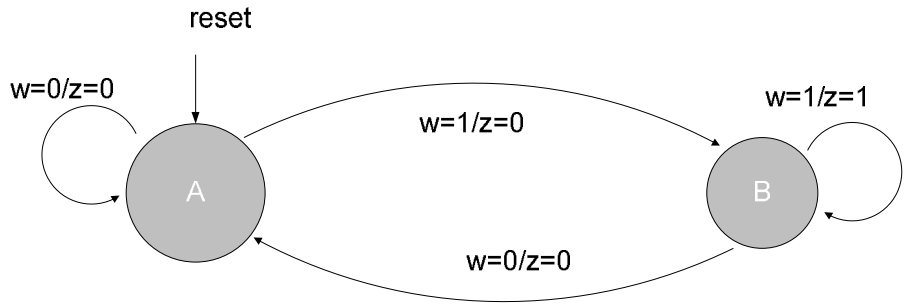


Εικόνα 513 Κυματομορφές μηχανής πεπερασμένων καταστάσεων

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

Α.3)Το μοντέλο μιας μηχανής τύπου*Mealy* βασισμένο σε μια παραλλαγή του παραδείγματος *Moore* 6.1 όπου η τιμή της εξόδου zδημιουργείται με βάση την παρούσα κατάσταση της μηχανής και την τρέχουσα τιμή της εισόδου, μπορεί να έχει το ακόλουθο διάγραμμα καταστάσεων.



Σχήμα 56 Μηχανή τύπου *Mealy*

Ο *VHDL* κώδικας:

```
LIBRARYieee ;
USE ieee.std_logic_1164.all ;
ENTITY  mealy IS
PORT (          Clock, Resetn :   IN STD_LOGIC ;
                                  W :   IN STD_LOGIC ;
                                  Z :   OUT     STD_LOGIC ) ;
END mealy ;

ARCHITECTURE Behavior OF mealy IS
                                  TYPE State_type IS (A, B) ;
                                  SIGNAL y : State_type ;
BEGIN
                                  PROCESS ( Resetn, Clock )
                                  BEGIN
                                      IF Resetn = '0' THEN
                                              y <= A ;
                                      ELSIF (Clock'EVENT AND  Clock = '1') THEN
                                              CASE y IS
                                                      WHEN A =>
                                                              IF w = '0' THEN y <= A ;
```

```
                                        ELSE y <= B ;
                                        END IF ;
                                WHEN B =>
                                        IF w = '0' THEN y <= A ;
                                        ELSE y <= B ;
                                        END IF ;
                        END CASE ;
                END IF ;
        END PROCESS ;
        PROCESS ( y, w )
        BEGIN
                CASE y IS
                        WHEN A =>
                                z <= '0' ;
                        WHEN B =>
                                z <= w ;
                END CASE ;
        END PROCESS ;
END Behavior ;
```

Δώσατε προσοχή στην εντολή ***CASE y IS*** μέσω της οποίας αποδίδεται η τιμή στην έξοδο *z* ανάλογα την κατάσταση στη οποία ευρίσκεται η μηχανή και την τιμή της εισόδου *w*.

Γράψτε εδώ την απάντησή σας:

Α.4Εξομοιώσατε τον ανωτέρω κώδικα και με τον WaveformEditor και εξετάσατε την ορθή και σύμφωνα με τις προδιαγραφές λειτουργία του.

Τοποθετείστε εδώ τις κυματομορφές σας:

Κυματομορφές της εξομοίωσης

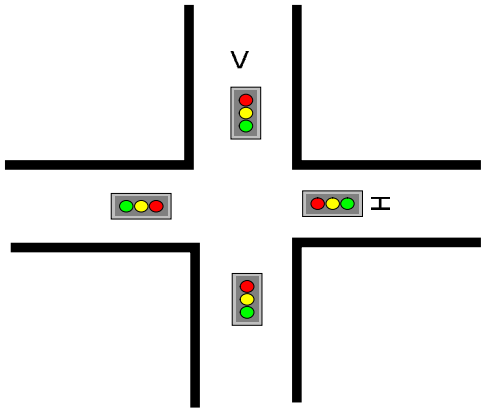
Α.5Να σχεδιαστεί μηχανή που θα παράγει τις ψηφιολέξεις :001,010,100,101, όταν η είσοδός του I=0, ενώ θα παραμένει στην ψηφιολέξη 000 όταν η I=1.

Γράψτε εδώ τις απαντήσεις σας:

5.7. ΑΣΚΗΣΗ 7 Σχεδίαση ψηφιακού συστήματος Ι.

Α.1)Ψηφιακό κύκλωμα ελέγχου φαναριών απλής διασταύρωσης.

Να σχεδιαστεί το ψηφιακό κύκλωμα για τον έλεγχο των φαναριών διασταύρωσης δύο ισοδύναμων σε κυκλοφορία οδών την κάθετη (*V*) και οριζόντια (*H*).



Σχήμα 57 Φανάρια απλής διασταύρωσης

Το ζητούμενο κύκλωμα θα ελέγχει ένα κλασικό και απλό σύστημα φαναριών που εξυπηρετεί την διασταύρωση δύο ισοδύναμων σε κυκλοφορία και μέγεθος δρόμων.

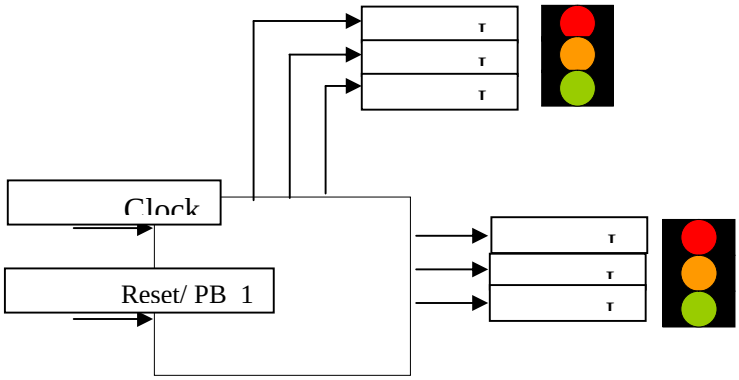
Περιλαμβάνει, ένα φανάρι για κάθε δρόμο και το σύστημα ελέγχου.

Σε κάθε φανάρι ανάβει περιοδικά πράσινο για 1 λεπτό, κίτρινο για 5 δευτερόλεπτα, κόκκινο για ένα λεπτό και 5 δευτερόλεπτα και αντίθετα στο άλλο.

Η όλη σχεδίαση θα προγραμματίσει το CYCLONE II του DE2 Board .

Οι ακροδέκτες που θα χρησιμοποιηθούν φαίνονται και στην εισαγωγή του εργαστηριακού βοηθήματος.

Τμήμα ή Πλήκτρο	*Pin του CYCLON*
A	PIN_AE23
G	PIN_AF23
D	PIN_AE22
PB1	PIN_N23
PB2	PIN_P23
Clock	PIN_D13



Σχήμα 58 Κύκλωμα για τον έλεγχο των φαναριών διασταύρωσης

Για την σχεδίαση του ζητούμενου συστήματος χρησιμοποιήστε μηχανές *FSM* και την γνωστή μονάδα υποδιαίρεσης συχνότητας *clk_div.vhd.*

Για την καλλίτερη λειτουργικότητα της σχεδίασης ας υπάρχει και ακροδέκτης μηδενισμού*Reset*Σχήμα 58.

Γράψτε εδώ τις απαντήσεις σας:

Α.2)Ψηφιακό κύκλωμα ελέγχου φαναριών διασταύρωσης πρωτεύουσας με δευτερεύουσα οδό και διάβαση πεζών.

Αναλύστε και μελετήστε την πιο προχωρημένη έκδοση της ίδιας άσκησης για οδούς διαφορετικής κυκλοφορίας και διάβαση πεζών.

Γράψτε εδώ τις απαντήσεις σας:

6. Βιβλιογραφία

M. ManoM., *Ψηφιακή Σχεδίαση***,** Εκδόσεις Παπασωτηρίου, 1992.

Θ. Λ. Δεληγιάννης, *Ηλεκτρονικά Ψηφιακά Κυκλώματα*, Εκδόσεις Πανεπιστημίου Πατρών, 2001.

Ε. Ζυγούρης, *Σχεδίαση ψηφιακών κυκλωμάτων και συστημάτων με την χρήση της VHDL*, Πάτρα 2002.

R. K. Dueck, *Digital Design with CPLD Applications AND VHDL*, Delmar part of Thomson Learning Publications, 2005, 2nd Edition.

P. J.Ashenden, *The VHDL Cookbook – First Edition*, University of Adelaide,1990.

Α.Ν. Σκόδρας, Ψηφιακή Λογική: Ακολουθιακά Κυκλώματα, Πάτρα 2001.

http://www.altera.comΑποτο sitetης altera πήραμεπ ληροφορίες σε μορφή pdf: Rapid_Prototyping_of_Digital_Systems_SOPC_Edition, Complete Digital Design, DE2_UserManual, cyc2_cii5v1_01, intro_to_quartus2, tut_quartus_intro_schem, tut_quartus_intro_vhdl, tut_simulation_vhdl, tut_timing_vhdl.