Μέρος Β'

Εργαστηριακές Ασκήσεις

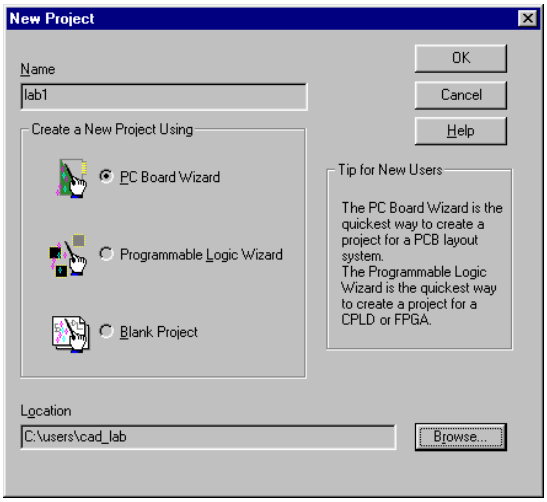
ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 1

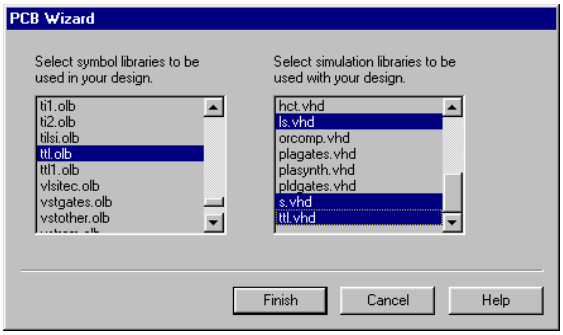
Σκοπός της εργαστηριακής άσκησης είναι η εξοικείωσή σας με την γραφική περιγραφή υλικού, την εξομοίωση ενός σχεδιασμού, την μέτρηση της καθυστέρησης διάδοσής του και τη διαδικασία εκσφαλμάτωσης. Στην άσκηση αυτή θα κληθείτε να εισάγετε γραφικά το σχηματικό ενός κυκλώματος μιας πύλης AND 74xx, 74Sxx, 74Asxx, 74LSxx και 74ALSxx.

Α. Εκκίνηση προγράμματος - Δημιουργία νέου σχεδιασμού

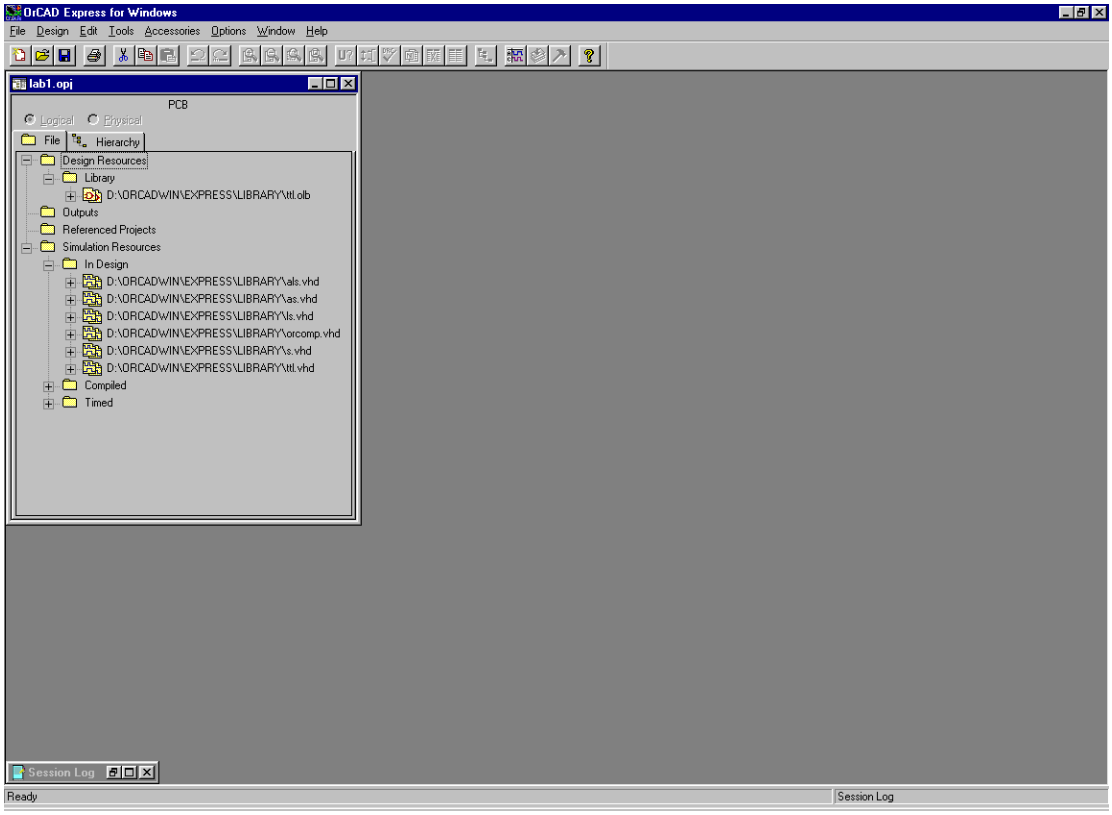
Εκτελούμε το πρόγραμμα Orcad Express for Windows. Το πρόγραμμα αυτό ακολουθεί την φιλοσοφία των σύγχρονων παραθυρικών εφαρμογών και πολλές λειτουργίες γίνονται όπως και στα άλλα προγράμματα που λειτουργούν κάτω από τα Microsoft Windows.

Όλα τα αρχεία που έχουν σχέση με ένα σχεδιασμό (σχηματικά, stimulus files, κοκ.) αποθηκεύονται από το Orcad με ενιαίο τρόπο με την μορφή project. Στην αρχική οθόνη που εμφανίζεται επιλέγουμε File -> New Project και δίνουμε το όνομα του project μας και το directory στο οποίο θέλουμε να βρίσκεται.





Από το μενού που προκύπτει επιλέγουμε PC Board Wizard. Με άλλα λόγια κατευθύνουμε το εργαλείο προς πλακέτα σαν πλατφόρμα υλοποίησης. Στην επόμενη οθόνη επιλέγουμε τις βιβλιοθήκες συμβόλων - symbol libraries που θέλουμε να χρησιμοποιήσουμε στον σχεδιασμό μας (στην συγκεκριμένη περίπτωση το ttl.old που περιέχει σχηματική πληροφορία για τις TTL πύλες) και τις βιβλιοθήκες εξομοίωσης - simulation libraries (στην συγκεκριμένη περίπτωση τις ttl.vhd, als.vhd, as.vhd, s.vhd, ls.vhd που περιέχουν τα μοντέλα χρονικής καθυστέρησης για τις οικογένειες standard TTL, ALS, AS, S και LS TTL αντίστοιχα).



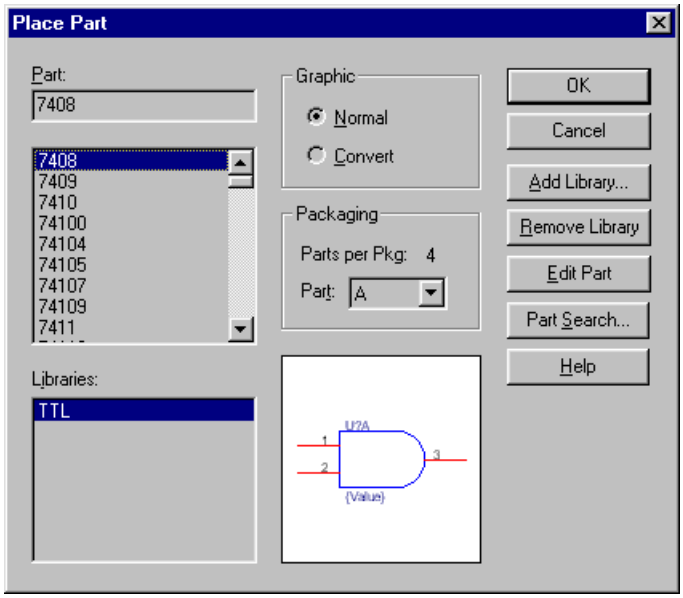
Στο αριστερό μέρος της οθόνης εμφανίζεται ο hierarchical browser. Σε αυτόν βρίσκεται συγκεντρωμένη όλη η πληροφορία που έχει σχέση με τον σχεδιασμό σας. Ο φάκελος Design Resources αποτελείται από τα αρχεία που περιέχουν την περιγραφή του κυκλώματος σε εσωτερική αναπαράσταση του πακέτου καθώς και τις βιβλιοθήκες συμβόλων που έχετε επιλέξει να χρησιμοποιήσετε σε αυτόν. Ο φάκελος Design Resources περιέχει μόνο σχηματική πληροφορία και πληροφορία συνδέσεων. Οι πληροφορίες για τα μοντέλα χρονικής καθυστέρησης βρίσκονται στον φάκελο Simulation Resources όπου θα βρείτε τρία directories (Compiled, In Design, Timed) που περιέχουν τις βιβλιοθήκες εξομοίωσης που είχατε επιλέξει προηγουμένως καθώς επίσης και την orcomp.vhd που χρησιμοποιείται εξ' ορισμού από το Orcad.

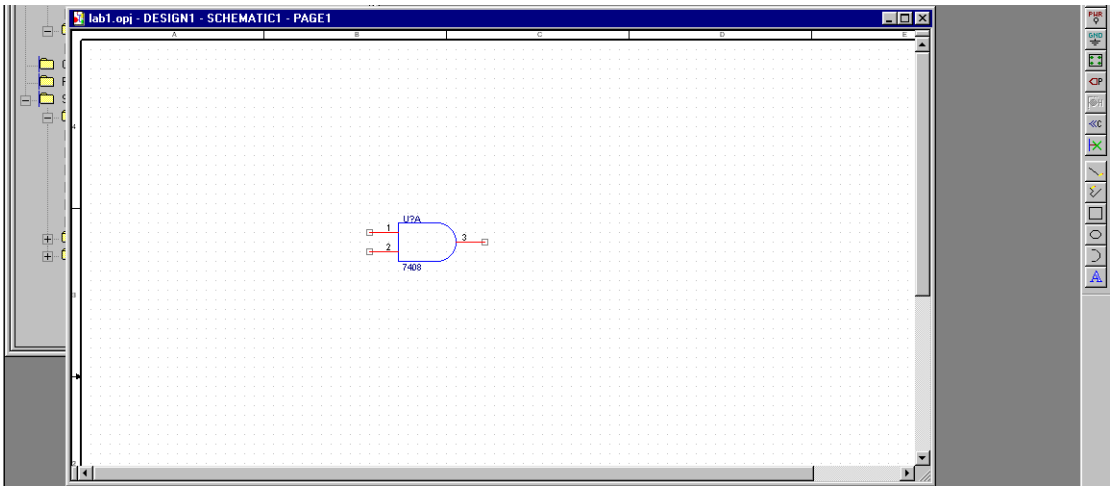
Β. Γραφική εισαγωγή σχεδιασμού

Προκειμένου να εισάγουμε το σχηματικό μας, επιλέγουμε το Design Resources και δίνουμε File -> New -> Design. Εμφανίζεται μία κενή σελίδα του graphic editor (αποτελεί την σελίδα 1 του σχηματικού 1 του σχεδιασμού μας) στην οποία θα σχηματίσουμε το κύκλωμα μας. Παρατηρήστε ότι ταυτόχρονα ενημερώνεται και ο hierarchical browser.

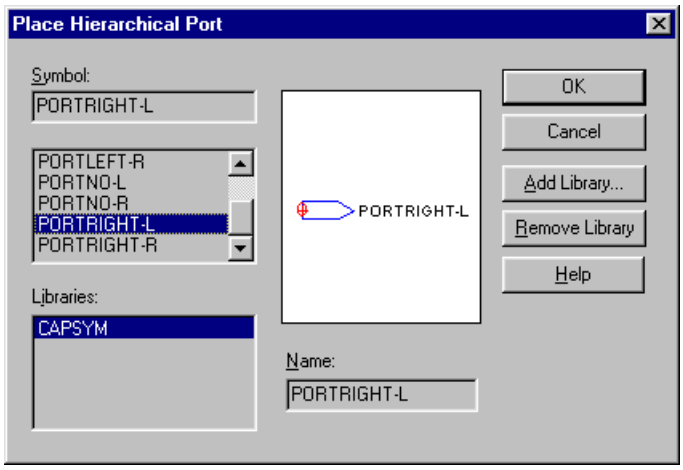
Κάθε κύκλωμα αποτελείται κυρίως από στοιχειώδη κομμάτια (parts), θύρες (ports) για τις εισόδους και τις εξόδους και καλώδια (wires) για τις συνδέσεις των παραπάνω. Τοποθετούμε τις πύλες μας στην σελίδα δίνοντας Place -> Part και επιλέγοντας την πύλη που θέλουμε. Στην συγκεκριμένη περίπτωση δίνουμε 7408. Επειδή κάθε part περιέχει 4

πύλες AND το εργαλείο αυτόματα ανακαλεί 1 πύλη AND και όχι ένα ολοκληρωμένο 7408. Για μεγαλύτερα στοιχειώδη κομμάτια (π.χ. 74244) κάθε part είναι και ένα αυτόνομο κομμάτι σχεδιασμού. Δίνοντας OK τοποθετούμε την πύλη AND στο σημείο που θέλουμε. Μπορούμε να τοποθετήσουμε όσες πύλες AND θέλουμε με αυτόν τον τρόπο στην σελίδα. Προκειμένου να τερματίσουμε την λειτουργία Place Part επιλέγουμε με το δεξί κουμπί του ποντικιού End Mode. Εάν κατά λάθος τοποθετήσαμε περισσότερες πύλες από αυτές που χρειαζόμαστε μπορούμε να επιλέξουμε με το ποντίκι τις περιττές πύλες και να τις διαγράψουμε με Delete. Παρατηρήστε ότι στο πάνω μέρος της πύλης εμφανίζεται το όνομα του ολοκληρωμένου –IC- U?A ενώ στις εισόδους και εξόδους της εμφανίζονται οι αριθμοί των ακροδεκτών του IC.

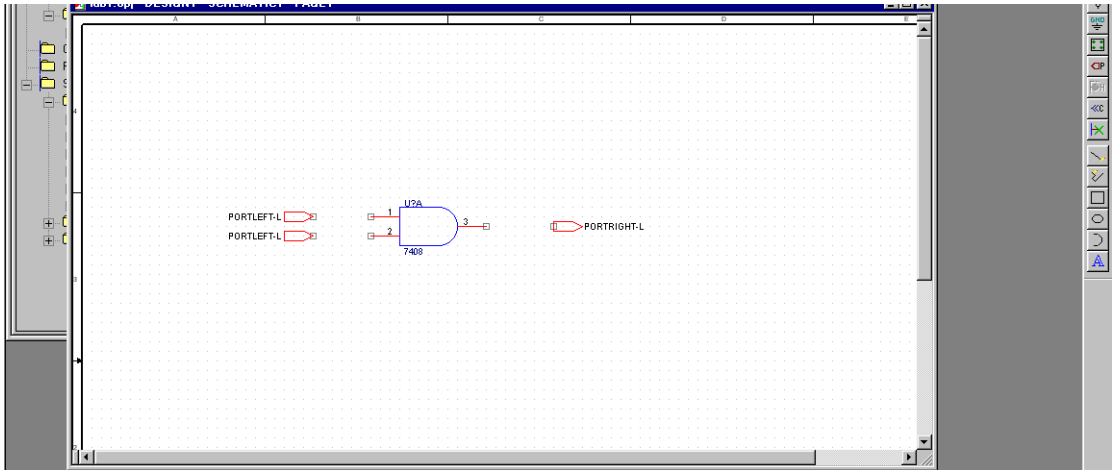




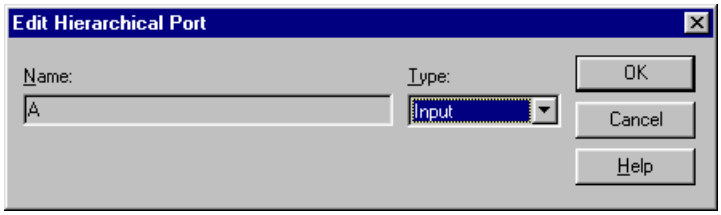
Στην συνέχεια δίνουμε Place -> Hierarchical Port για να τοποθετήσουμε τα I/O του κυκλώματός μας. Επιλέγουμε και τοποθετούμε το PORTRIGHT-L για την έξοδο και το PORTLEFT-L για τις δύο εισόδους. Με το δεξί κουμπί του ποντικιού και End Mode τερματίζουμε και αυτή την λειτουργία.



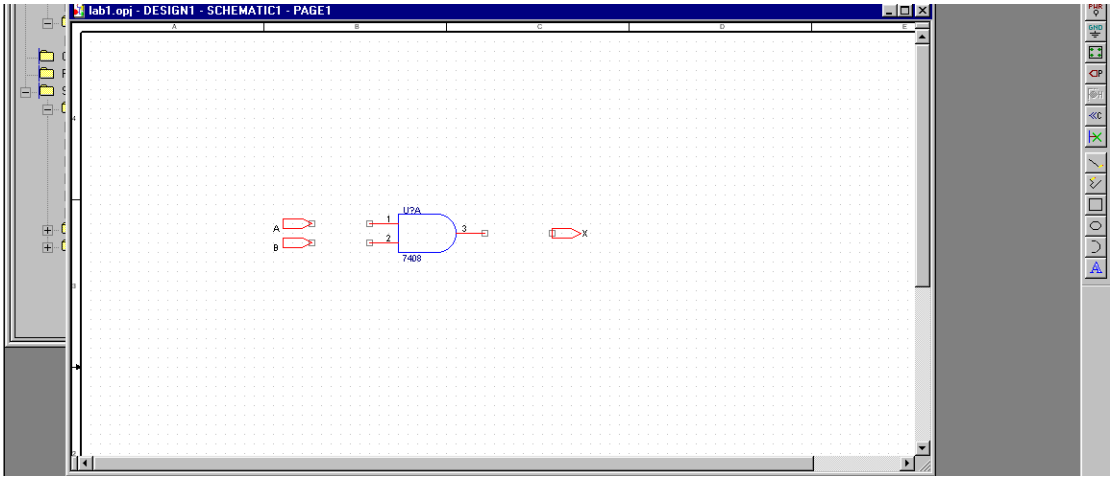
Προκειμένου να περιστρέψουμε οποιοδήποτε object στο σχήμα (port, part, wire) το επιλέγουμε και με το δεξί κουμπί του ποντικιού δίνουμε την αντίστοιχη λειτουργία (Rotate, Mirror Horizontally, Mirror Vertically).



Με διπλό κλικ πάνω στα ports δίνουμε όνομα A & B και τύπο Input για τις εισόδους και όνομα X και τύπο Output για την έξοδο.



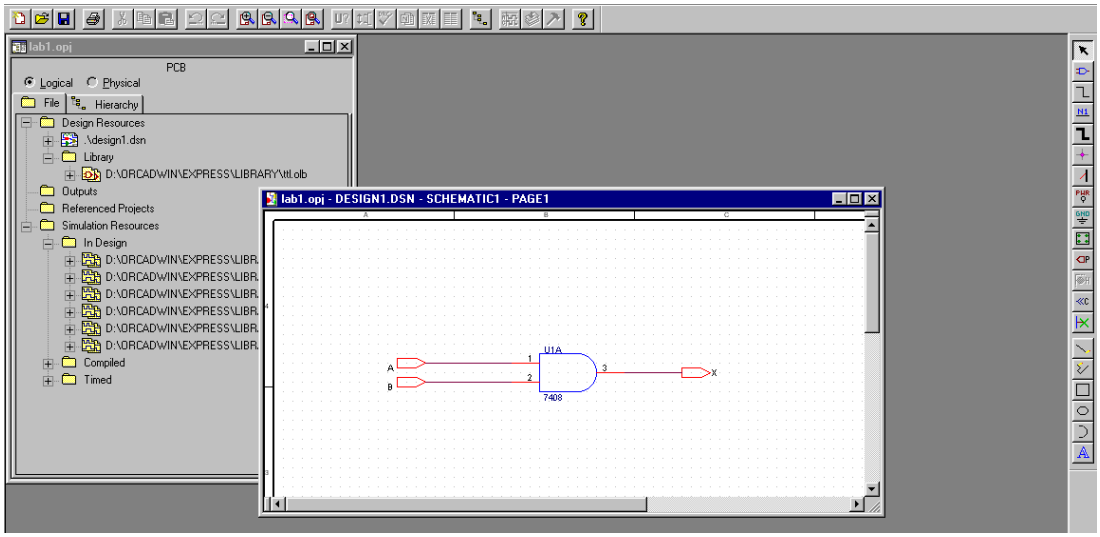
Το κύκλωμα μας θα πρέπει να φαίνεται πλέον όπως παρακάτω:



Στην συνέχεια συνδέουμε τα ports με την πύλη AND με wires. Δίνουμε Place-> Wire. Παρατηρήστε ότι στην σελίδα του σχηματικού μας ο cursor έχει μετατραπεί από βέλος σε σταυρό. Ενώνουμε τα άκρα που θέλουμε. Αυτό γίνεται κάνοντας κλικ με το ποντίκι εκεί όπου θέλουμε να ξεκινάει το καλώδιο και σύροντας το ποντίκι μέχρι εκεί που θέλουμε και κάνοντας κλικ για να τερματίσει. Τερματίζουμε την λειτουργία αυτή με End Wire (δεξί κουμπί). Εάν κατά λάθος δημιουργήσατε καλώδια που δεν χρειάζονται μπορείτε να τα επιλέξετε με το ποντίκι και να τα διαγράψετε με Delete. Οι παραπάνω λειτουργίες μπορούν να γίνουν εκτός από το μενού Place και με την βοήθεια των εικονιδίων στην εργαλειοθήκη που βρίσκεται στο δεξιό μέρος της οθόνης

Αφού τελειώσουμε τα παραπάνω, επιλέγουμε το Design Resources -> ./design1.dsn στον Hierarchical Browser και επιλέγουμε Tools -> Update Part References. Αφού δώσουμε OK και αποθηκεύσουμε το κύκλωμα και τις αλλαγές στο project μας παρατηρούμε ότι ενημερώνονται τα ονόματα των διαφόρων πυλών (π.χ. το U?A γίνεται U1A που σημαίνει η πρώτη πύλη από το A 7408 ολοκληρωμένο).

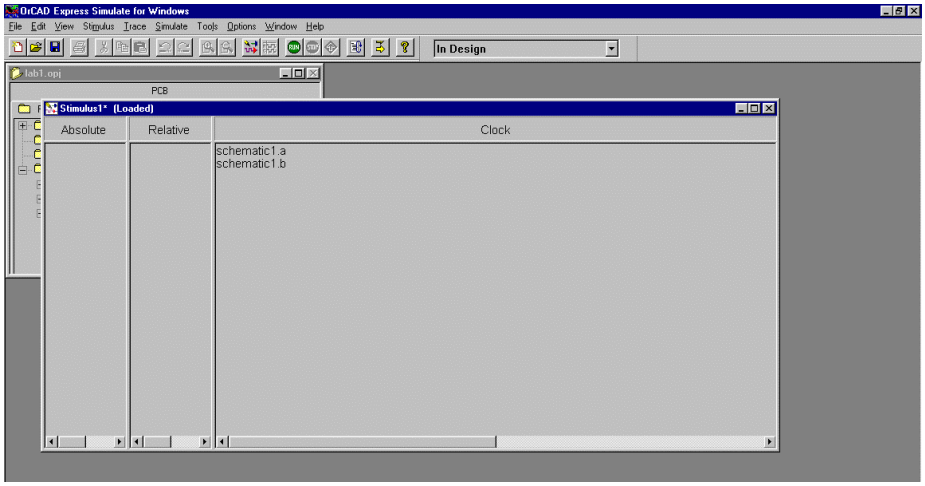
Το κύκλωμα πλέον θα πρέπει να φαίνεται όπως παρακάτω:



Γ. Εξομοίωση

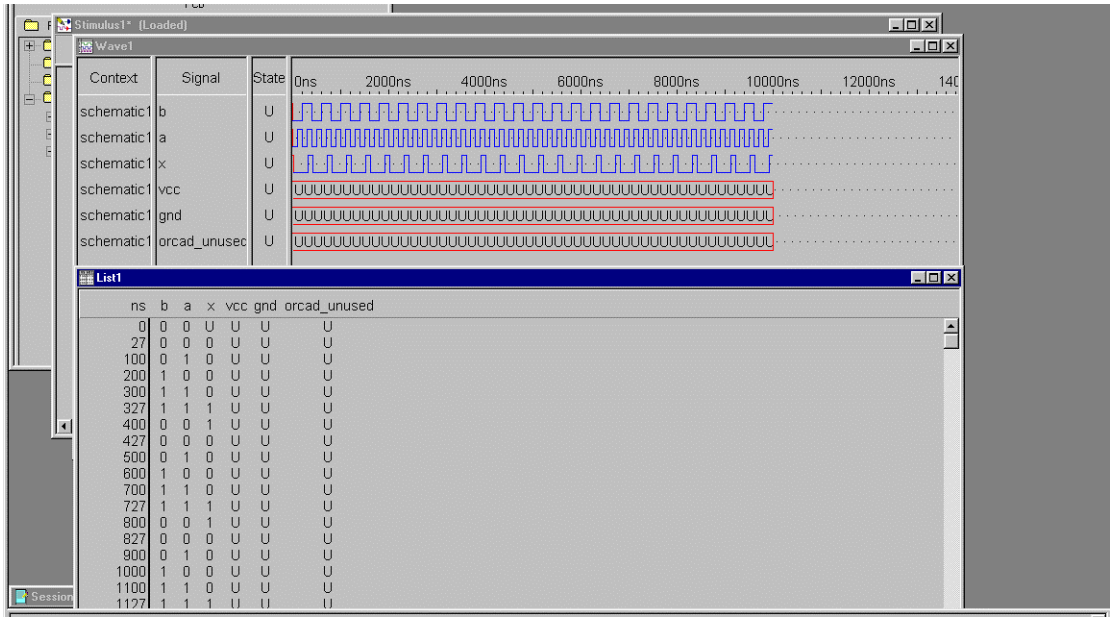
Αφού επιλέξουμε τον Hierarchical Browser δίνουμε Tools -> Simulate και αυτόματα εκτελείται ο Express Simulator. Επιλέγουμε In Design εξομοίωση και δίνουμε yes για να φορτωθεί ο σχεδιασμός μας. Στην προκειμένη περίπτωση και η επιλογή Compiled θα οδηγούσε στην ίδια εξομοίωση, αφού το κύκλωμά μας δεν περιλαμβάνει κανένα επίπεδο αφαίρεσης. Από το μενού Options->Preferences μπορούμε να επιλέξουμε το μοντέλο χρονικής καθυστέρησης (minimum, typical ή maximum). Επιλέξτε τυπική καθυστέρηση.

Πριν κάνουμε την εξομοίωση θα πρέπει να ετοιμάσουμε τις κυματομορφές των εισόδων μας. Αυτό γίνεται με την επιλογή Stimulus -> New Interactive. Μπορούμε να ορίσουμε τριών ειδών κυματομορφές: Absolute, Relative και Clock ανάλογα με το είδος της κυματομορφής εισόδου που θέλουμε. Οι κυματομορφές φαίνονται στο παράθυρο Stimulus1.

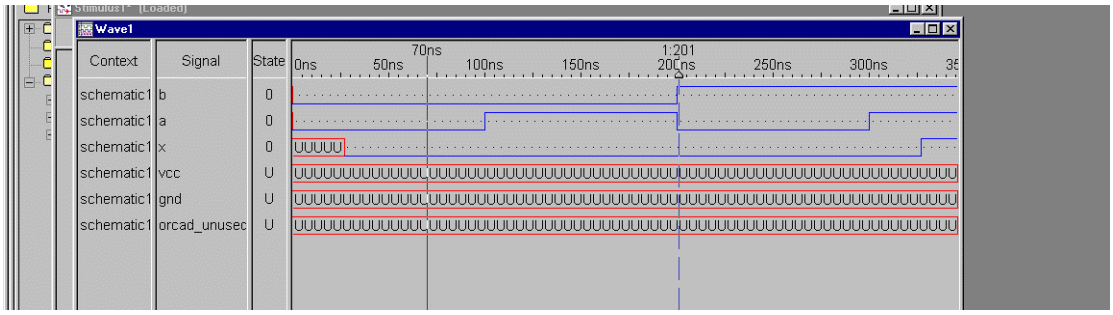


Επιλέγουμε το Clock Stimulus και αφού επιλέξουμε την είσοδο που θέλουμε π.χ. το A ορίζουμε κυματομορφή με 100ns low και 100ns high που επαναλαμβάνεται συνέχεια και δίνουμε Add. Αντίστοιχα για την είσοδο B δίνουμε 200ns low και 200ns high.

Αφού φορτώσουμε το stimulus file εκτελούμε την εξομοίωση για χρόνο 10000ns δίνοντας Simulate -> Run. Στο παράθυρο Wave1 εμφανίζονται τα αποτελέσματα σε μορφή κυματομορφής ενώ στο παράθυρο List1 εμφανίζονται οι χρόνοι στους οποίους γίνεται αλλαγή κατάστασης στα σήματα εισόδου ή εξόδου.



Με τις επιλογές View -> Zoom In και View -> Zoom Out μπορούμε να μεγεθύνουμε ή να μικρύνουμε την ορατή περιοχή για μεγαλύτερη λεπτομέρεια. Επιλέγουμε ένα σημείο της κυματομορφής και πατάμε το αριστερό κουμπί του ποντικιού. Εμφανίζεται ο marker που μας δείχνει σε ποιο χρονικό σημείο βρισκόμαστε κάθε φορά ενώ οι τιμές των σημάτων φαίνονται στην στήλη State. Επιλέγοντας τον marker, μπορούμε να τον σύρουμε σε όποιο χρονικό σημείο θέλουμε. Προκειμένου να μετρήσουμε την διαφορά μεταξύ δύο σημείων της κυματομορφής επιλέγουμε με το δεξί κουμπί του ποντικιού Add Delta Marker. Στο κάτω μέρος της οθόνης υπάρχει η απόσταση μεταξύ των δύο markers. Εάν θέλουμε μπορούμε να προσθέσουμε και δεύτερο Delta Marker. Οι Delta Markers μετράνε κάθε φορά την απόσταση από τον Marker.



Μετρήστε την καθυστέρηση της πύλης όταν μεταβαίνει στο λογικό μηδέν και όταν μεταβαίνει στο λογικό ένα και καταγράψτε την στον αντίστοιχο πίνακα.

Κλείστε τον εξομοιωτή κάνοντας Close και αφού αποθηκεύστε το stimulus file στο αρχείο stimulus.stm, το προσθέσετε στο project και μετά το κάνετε unload από τον εξομοιωτή.

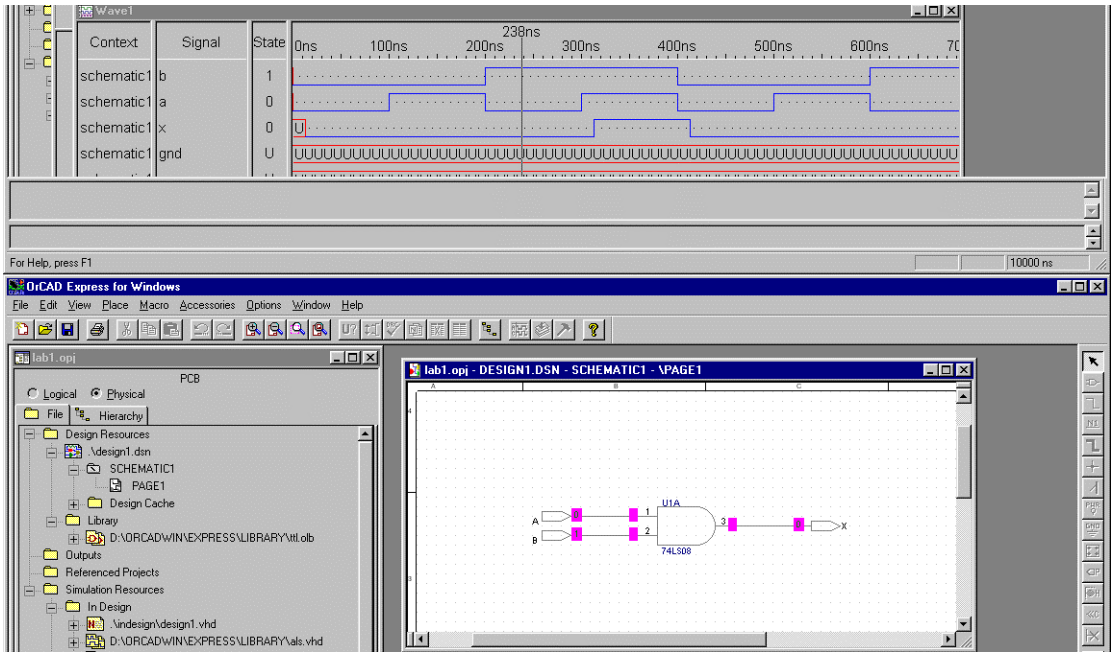
Επαναλάβατε την παραπάνω διαδικασία για τις πύλες 74LS08, 74S08, 74AS08 και 74ALS08 και μετρήστε την καθυστέρηση διάδοσης τους. Επιβεβαιώστε τους χρόνους που

μετρήσατε ανοίγοντας την αντίστοιχη βιβλιοθήκη εξομοίωσης και επιλέγοντας την πύλη που θέλετε μέσα από τον Hierarchical Browser. Συμπληρώστε τον παρακάτω πίνακα:

	7408	74ALS08	74AS08	74S08	74LS08
Χρόνος καθυστέρησης μετάβασης στο λογικό 0					
Χρόνος καθυστέρησης μετάβασης στο λογικό1					

Δ. Διαδικασία Εκσφαλμάτωσης

Προκειμένου να ελέγχουμε την ορθότητα του σχεδιασμού μας και να εντοπίζουμε εύκολα και γρήγορα λάθη, μπορούμε να χρησιμοποιήσουμε την διαδικασία της διαγώνιας έρευνας (Cross Probing). Προκειμένου να το πετύχουμε αυτό αφού εκτελέσουμε την εξομοίωση επιλέγουμε Window -> Split Screen ώστε να φαίνονται ταυτόχρονα στην οθόνη τόσο οι κυματομορφές όσο και το σχηματικό του κυκλώματός μας. Επίσης επιλέγουμε στον Hierarchical Browser την φυσική απεικόνιση (Physical) και όχι την λογική (Logical). Παρατηρούμε ότι κινώντας τον marker στα διάφορα σημεία της κυματομορφής εμφανίζονται οι τιμές των σημάτων εισόδου και εξόδου πάνω στο σχηματικό μας. Αντίστοιχα εάν επιλέξουμε κάποιο από τα σήματα στην κυματομορφή αυτόματα επιλέγεται το αντίστοιχο σήμα στο σχηματικό.



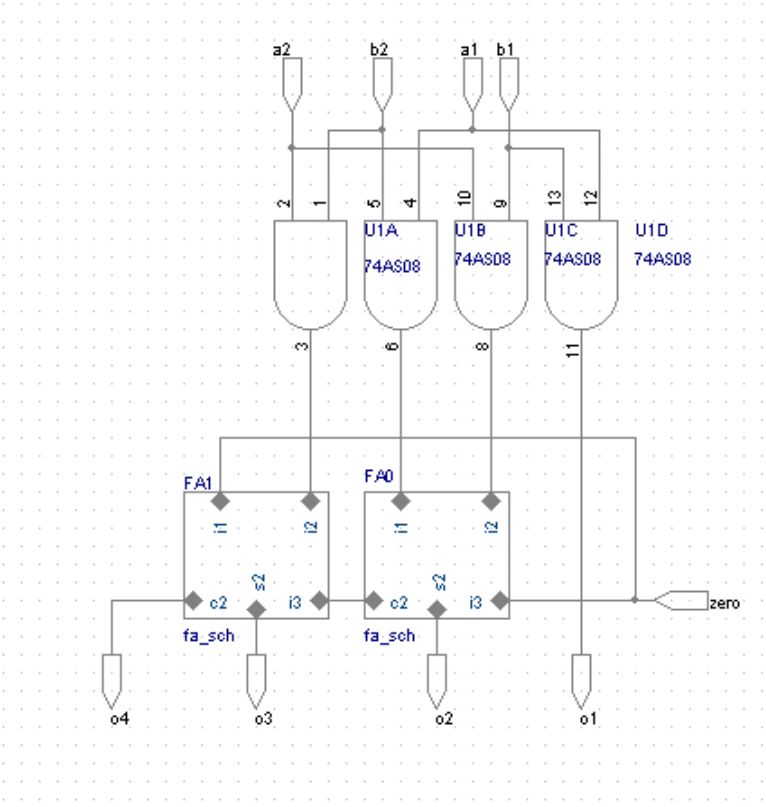
ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 2

Σκοπός της εργαστηριακής άσκησης είναι η εξοικείωσή σας με τις φιλοσοφίες σχεδιασμού top-down και bottom-up. Για τη μεν top-down φιλοσοφία θα σχεδιάσετε έναν carry-save πολλαπλασιαστή 2 δυαδικών ψηφίων, για τη δε bottom-up έναν καταχωρητή 2 δυαδικών ψηφίων.

A. Top-down design

Για τον σχεδιασμό σας θα χρησιμοποιήσετε την οικογένεια πυλών 74AS TTL και επομένως θα χρειαστείτε την ttl.old βιβλιοθήκη συμβόλων και την as.vhd βιβλιοθήκη εξομοίωσης.

Οπως ίσως γνωρίζετε από τις παραδόσεις Λογικού Σχεδιασμού, ένας carry-save πολλαπλασιαστής των δύο δυαδικών ψηφίων μπορεί να σχεδιαστεί με 2 πλήρεις αθροιστές και τέσσερις πύλες AND. Θεωρώντας προς το παρόν ότι ο πλήρης αθροιστής είναι ένα μαύρο κουτί με τρεις εισόδους (i1, i2, i3) και δύο εξόδους (c, s) για το κρατούμενο και το άθροισμα αντίστοιχα, δημιουργήστε την πρώτη σελίδα του σχηματικού σύμφωνα με το παρακάτω σχήμα. Τα blocks FA0 και FA1 αποτελούν hierarchical blocks.

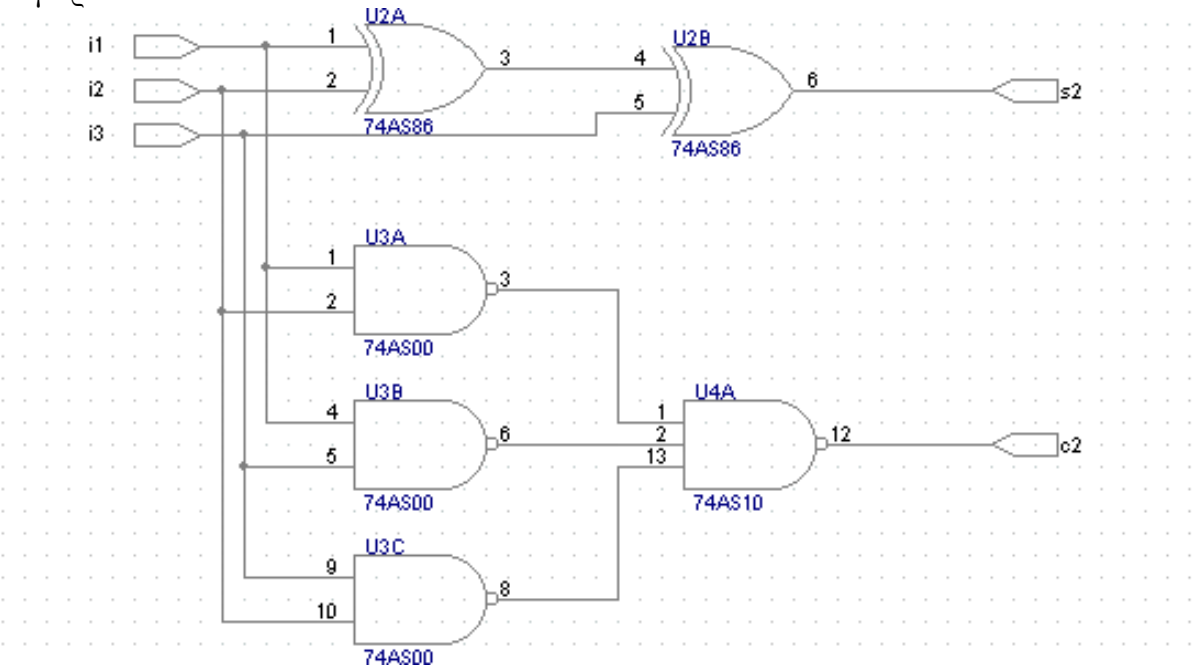


Τα hierarchical blocks αποτελούνται από:

- ένα μοναδικό όνομα αναφοράς μέσα στην σελίδα του σχηματικού (FA0 και FA1) για κάθε αντίγραφο,
- το όνομα του block (fa_sch) και

- τα σήματα εισόδου και εξόδου.

Τοποθετούνται στην σελίδα επιλέγοντας Place -> Hierarchical Block. Στο πεδίο reference το όνομα αναφοράς του block το οποίο θα πρέπει να είναι μοναδικό στην σελίδα μας (π.χ. FA0, FA1, FA2, …), στο πεδίο Implementation Type επιλέγουμε Schematic View (υπονοώντας ότι θα εισάγουμε τον πλήρη αθροιστή με γραφικό τρόπο) και στο Implementation Name δίνουμε το όνομα του block (π.χ fa_sch). Στην συνέχεια ορίζουμε με το ποντίκι τις διαστάσεις του block. Τα σήματα εισόδου και εξόδου του hierarchical block αποτελούν hierarchical pins. Εισάγονται επιλέγοντας με το ποντίκι το hierarchical block και στην συνέχεια με την εντολή Place -> Hierarchical Pins και αφού δώσουμε το όνομα του σήματος, το είδος και το πλάτος του. Η λειτουργία τερματίζεται με δεξί πλήκτρο του ποντικιού και End Mode.



Αφού τελειώσετε την εισαγωγή του πρώτου επιπέδου ιεραρχίας του σχεδιασμού σας, θα πρέπει να επεξηγήσετε την λειτουργία του hierarchical block fa_sch. Προσέξτε ότι εφόσον τα δύο blocks έχουν την ίδια λειτουργία, μόνο μία επεξήγηση χρειάζεται. Το σχηματικό που θα πρέπει να φτιάξετε για να επεξηγήσετε τον πλήρη αθροιστή φαίνεται στο παραπάνω σχήμα. Επιλέξτε λοιπόν ένα από τα δύο hierarchical blocks που έχετε ζωγραφίσει και με το δεξί πλήκτρο του ποντικιού επιλέξτε Descend Hierarchy. Το εργαλείο θα δημιουργήσει για εσάς μια κενή σελίδα μαζί με τους ακροδέκτες (I/Os) του hierarchical block. Εκεί εισάγετε το σχηματικό του πλήρους αθροιστή. Αφού ολοκληρώσετε την εισαγωγή του κυκλώματος, παρατηρείστε την δομή των Design Resources στον Hierarchical Browser.

Ακολουθήστε την διαδικασία που περιγράφεται στην προηγούμενη εργαστηριακή άσκηση για να εκτελέσετε την χρονική εξομοίωση ώστε να επαληθεύσετε την σωστή λειτουργία του σχηματικού σας, που είναι και το ζητούμενο του πρώτου μέρους.

B. Bottom-up design

Για τον σχεδιασμό σας θα χρησιμοποιήσετε την οικογένεια πυλών 74AS TTL και επομένως θα χρειαστείτε την ttl.old βιβλιοθήκη συμβόλων και την as.vhd βιβλιοθήκη εξομοίωσης.

Ενας καταχωρητής 2 δυαδικών ψηφίων αποτελείται από 2 D flip-flops. Παρότι το D flip-flop υπάρχει στις δεδομένες βιβλιοθήκες, εμείς θα δημιουργήσουμε ένα νέο σε μία δική μας βιβλιοθήκη, Στο νέο αυτό flip-flop θα δώσουμε όνομα d_ff.

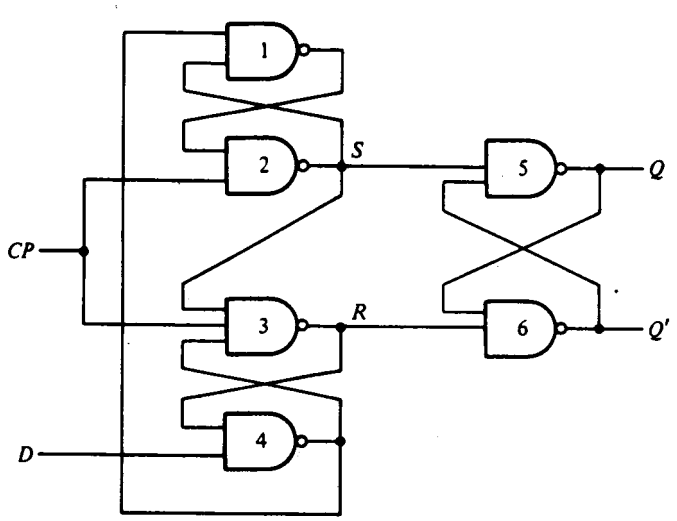
Προκειμένου να δημιουργήσουμε μία νέα βιβλιοθήκη επιλέγουμε : File -> New -> Library. Με το δεξί πλήκτρο του ποντικιού πάνω στο εικονίδιο της νέας βιβλιοθήκης μέσα στον hierarchical browser επιλέγουμε New Part. Δίνουμε το όνομα του νέου part (π.χ. d_ff) και στην επιλογή Attach implementation επιλέγουμε στο Type το Schematic View (το flip-flop θα επεξηγηθεί σχηματικά) ενώ στο Name δίνουμε το όνομα του σχηματικού που θα περιγράφει το part (π.χ. d_ff_schematic).

Στην σελίδα που εμφανίζεται δημιουργούμε το σχήμα του συμβόλου του νέου part (Place -> Rectangle, Place -> Ellipse, κ.ο.κ) και εισάγουμε πληροφορία για τα σήματα εισόδου και εξόδου του (Place -> Pin, Place -> Pin Array). Τελειώνοντας, αποθηκεύουμε τα σύμβολα της βιβλιοθήκης μας με κάποιο όνομα και τερματίζουμε το library project.

Προκειμένου να δημιουργήσουμε τον καταχωρητή των 2 δυαδικών ψηφίων δημιουργούμε ένα νέο project. Με το δεξί πλήκτρο του ποντικιού πάνω στο εικονίδιο Library μέσα στον hierarchical browser επιλέγουμε Add File και εισάγουμε την βιβλιοθήκη που δημιουργήσαμε νωρίτερα. Στην συνέχεια εισάγουμε το σχηματικό μας χρησιμοποιώντας αντίγραφα του νέου part που έχουμε δημιουργήσει. Με το δεξί κουμπί του ποντικιού πάνω στο σύμβολο του νέου part (d_ff) και την επιλογή Descend Hierarchy μπορούμε να εισάγουμε την σχηματική περιγραφή του. Προσέξτε ότι το εργαλείο θεωρεί ότι τα σύμβολα συνήθως δεν κρύβουν ιεραρχία πίσω τους, αλλά προέρχονται από τις βιβλιοθήκες κάποιου κατασκευαστή. Για να μπορέσετε συνεπώς να εκτελέσετε την Descend Hierarchy θα πρέπει πρώτα να αλλάξετε τις ιδιότητες των συμβόλων σε non-primitives. Παρατηρήστε ότι τα σήματα εισόδου και εξόδου είναι ήδη γνωστά από την περιγραφή του συμβόλου στην βιβλιοθήκη. Διάφοροι σχεδιασμοί μπορούν να χρησιμοποιούν αντίγραφα του νέου flip-flop, αρκεί να περιλαμβάνουν την βιβλιοθήκη, την οποία δημιουργήσαμε στα αρχικά στάδια του δεύτερου μέρους.

Εισάγετε το παρακάτω σχηματικό για το d_ff χρησιμοποιώντας πύλες της οικογένειας 74AS TTL. Τέλος, επαληθεύστε την ορθότητα του σχεδιασμού σας με την

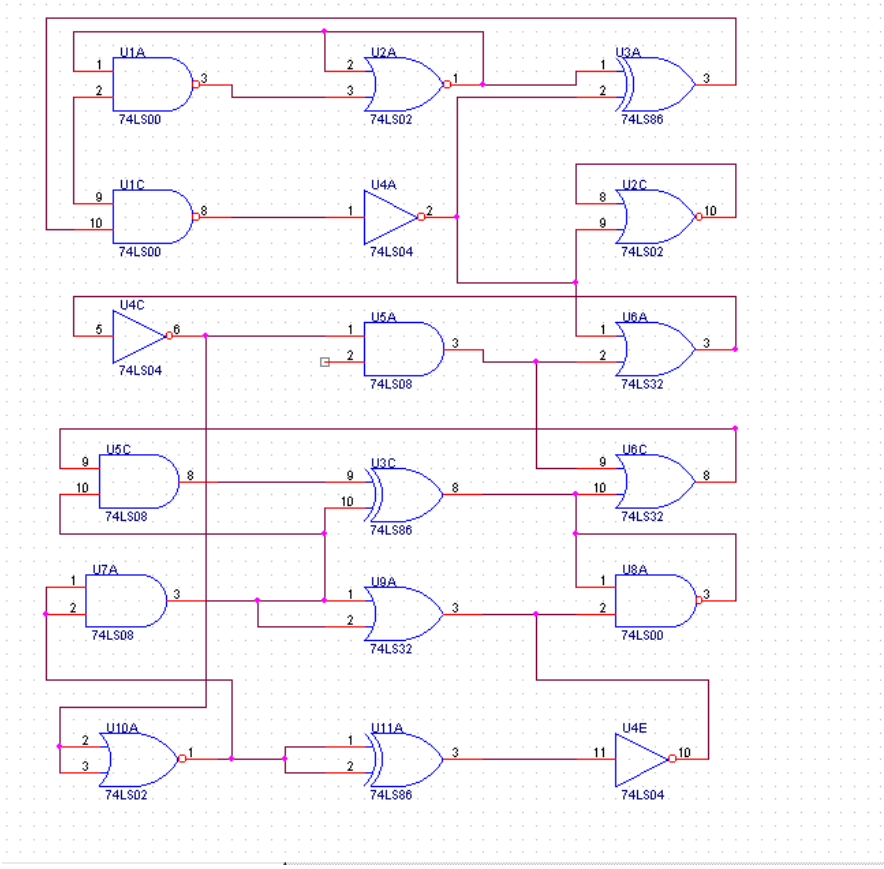
βοήθεια της χρονικής εξομοίωσης . Παραδοτέα για το 2ο μέρος είναι το stimulus που χρησιμοποιήσατε καθώς και τα αποτελέσματα της εξομοίωσης.



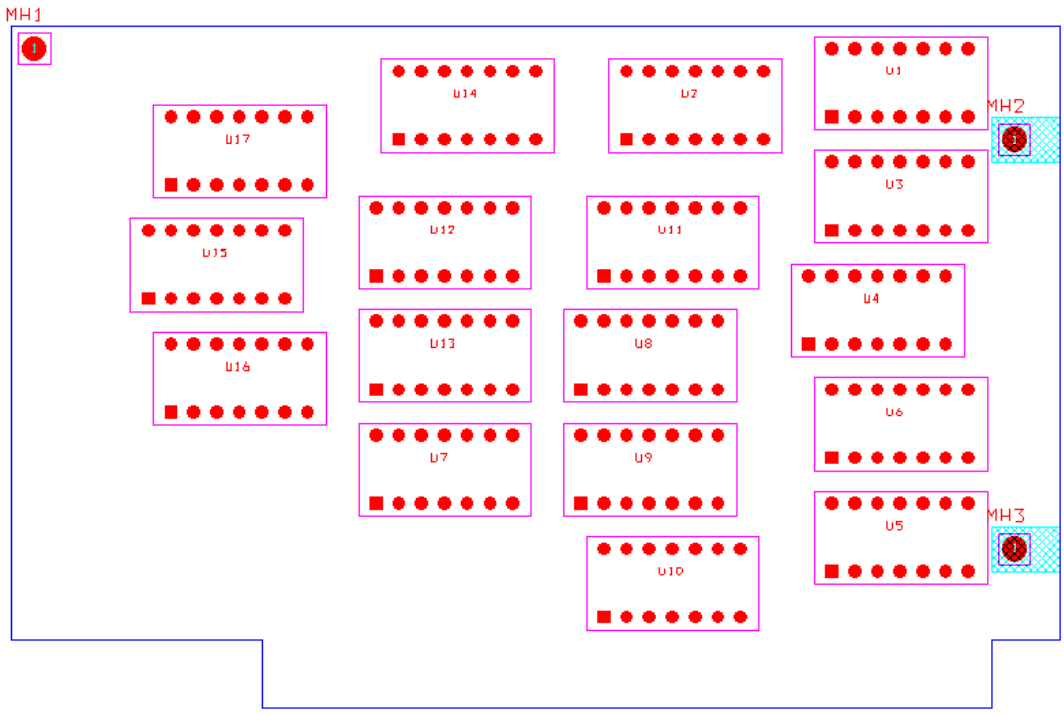
ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 3

Σκοπός της άσκησης είναι η εξοικείωση με τις διαδικασίες back-end σε τεχνολογία πλακέτας. Το τελικό αποτέλεσμα της εργαστηριακής άσκησης θα είναι ένας πλήρως τοποθετημένος στο χώρο και διασυνδεδεμένος σχεδιασμός.

Το πρώτο βήμα είναι η εισαγωγή του σχεδιασμού μας με γραφικό τρόπο και η δημιουργία του netlist. Έστω ότι θέλουμε να κατασκευάσουμε το PCB για κύκλωμα αποτελούμενο από 4 σχεδιαστικά κύτταρα κάθε ένα της μορφής της παραπάνω εικόνας. Αφού σχεδιάσετε το κύκλωμα στο Orcad Express, επιλέγετε Tools -> Update Part References και στην συνέχεια Tools -> Create Netlist. Επιλέγουμε το Layout και ενεργοποιούμε το User Properties are in inches. Το αποτέλεσμα αυτού είναι η δημιουργία του netlist σε ένα αρχείο με κατάληξη .MNL.



Στην συνέχεια τερματίζουμε το Orcad Express και εκτελούμε το Orcad Layout. Δίνοντας File -> New επιλέγουμε το design template που θέλουμε να ακολουθήσουμε (τα διάφορα design template που υπάρχουν ορίζουν παραμέτρους της πλακέτας όπως διαστάσεις, επίπεδα γραμμών κ.α.). Επιλέξτε το tutor.tch και στην συνέχεια ορίστε το αρχείο .MNL που δημιουργήσατε προηγουμένως. Τέλος δώστε το όνομα του αρχείου όπου θα αποθηκευτεί ο υλοποιημένος σχεδιασμός.



Στην οθόνη βλέπετε πλέον τις διαστάσεις της πλακέτας και τα ολοκληρωμένα κυκλώματα που πρέπει να εισαχθούν σε αυτή με τη μορφή footprints (αποτυπώματα). Το template tutor που χρησιμοποιούμε επιτρέπει μόνο τη χρήση DIP through hole ολοκληρωμένων.

Το επόμενο βήμα είναι η διάταξη στον χώρο των ολοκληρωμένων του σχεδιασμού. Η διαδικασία αυτή γίνεται αυτόματα επιλέγοντας Auto -> Batch Place. Εάν κάποια από τα ολοκληρωμένα δεν έχουν τοποθετηθεί στην επιθυμητή κατά την κρίση σας θέση μπορείτε να τα μετακινήσετε σύροντας τα με το ποντίκι. Παρατηρήστε ότι εκτός από τα ολοκληρωμένα κυκλώματα φαίνονται και οι γραμμές που ενώνουν τους διάφορους ακροδέκτες μεταξύ τους με τη μορφή ασύνδετων γραμμών (hairlines).Το επόμενο και τελευταίο βήμα είναι η διασύνδεση των ακροδεκτών των ολοκληρωμένων κυκλωμάτων. Αυτό μπορεί να γίνει αυτόματα με την επιλογή Auto -> Batch Route.

Η πλακέτα που κατασκευάσατε είναι δύο επιπέδων. Διασύνδεση των στοιχείων επιτρέπεται και στα δύο αυτά επίπεδα. Ανατρέχοντας στα manuals του εργαλείου (κάτω από το μενού help), απενεργοποιείστε το δεύτερο επίπεδο, έτσι ώστε η διαδρόμιση να γίνεται μόνο στο ένα επίπεδο. Επαναλάβετε την διαδικασία διαδρόμισης και εφόσον το εργαλείο δεν τα καταφέρει να σας παράγει έναν πλήρως διασυνδεδεμένο σχεδιασμό, πειραματιστείτε με διαφορετική διάταξη των ολοκληρωμένων, ή αλλάξτε το μέγεθος της πλακέτας. Παραδώστε τα σχηματικά των πλακετών ενός και δύο επιπέδων

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4

(Ιδέα & Υλοποίηση : Κ. Αδαός – Υπ. Διδάκτωρ ΤΜΗΥΠ)

Στην άσκηση αυτή θα φτιάξουμε ένα μικρό "κλειδωτήρι" (γνωστό ως hasp) για την προστασία του λογισμικού μας από πιθανούς αντιγραφείς. Ενα hasp διανέμεται μαζί με κάθε αντίγραφο του λογισμικού και τοποθετείται σε μια από τις θύρες του υπολογιστή μας. Το λογισμικό σε τακτά χρονικά διαστήματα, στέλνοντας και διαβάζοντας πληροφορίες μέσω της θύρας του υπολογιστή μας, ελέγχει την παρουσία ή την απουσία του hasp. Ετσι ο επίδοξος αντιγραφέας μπορεί μεν να αντιγράψει το λογισμικό, όμως αυτό δε θα μπορεί να λειτουργήσει σε άλλον υπολογιστή, παρά μόνο αν αντιγραφεί και το hasp.

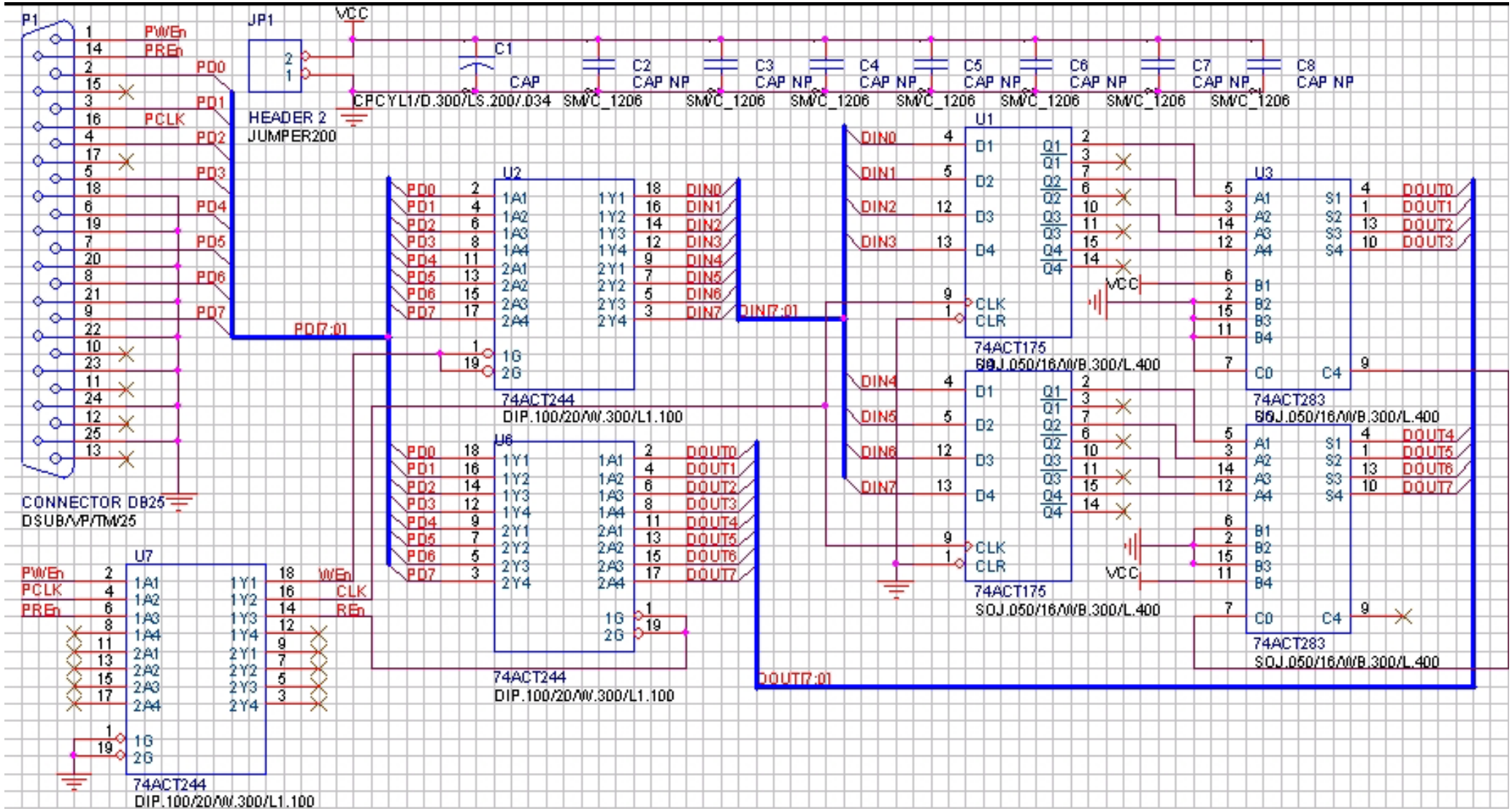
Για το παράδειγμά μας, υποθέτουμε :

- Οτι το hasp θα τοποθετηθεί σε μια παράλληλη θύρα του υπολογιστή μας και καμμία άλλη συσκευή δε θα χρησιμοποιεί τη συγκεκριμένη θύρα.
- Οτι το λογισμικό μας δεν ελέγχει απλά την παρουσία του hasp, αλλά του αναθέτει και κάποια μορφή επεξεργασίας της πληροφορίας. Υποθέτουμε ότι αυτή η επεξεργασία είναι η απεικόνιση κάθε byte σε κώδικα XS (Excess) 127.
- Το πρωτόκολλο επικοινωνίας με το hasp είναι το δυνατόν απλούστερο. Το λογισμικό αφού τοποθετήσει το byte στην παράλληλη θύρα, ενεργοποιεί ένα σήμα για να υποδείξει τη διαδικασία εγγραφής και παράγει ένα παλμό ρολογιού για το χρονισμό της εγγραφής. Το λογισμικό υποδεικνύει ανάγνωση των κωδικοποιημένων δεδομένων ενεργοποιώντας ένα σήμα ανάγνωσης.

Σκοπός της άσκησης είναι η εξοικείωσή σας με το χειρισμό αποκλειστικής πρόσβασης πάνω σε μια διαμοιραζόμενη αρτηρία, και της φάσης back-end..

Α. Εισαγωγή του σχεδιασμού

Η προτεινόμενη λύση για το παραπάνω πρόβλημα φαίνεται στο σχηματικό της επόμενης σελίδας. Τα σήματα PWEn, PCLK και PREn, είναι τα σήματα ελέγχου που μας παράγει το λογισμικό μας, ενώ η αρτηρία PD είναι τα παράλληλα δεδομένα που ανταλλάσονται μεταξύ του hasp και του λογισμικού μας. Προσέξτε ότι η αρτηρία PD είναι δύο κατευθύνσεων (bidirectional) και συνεπώς θα πρέπει να υπάρχει αυστηρός έλεγχος σχετικά με το πότε θα πρέπει το hasp να οδηγήσει αυτή την αρτηρία. Το hasp οδηγεί αυτή την αρτηρία ΜΟΝΟ όταν λάβει σήμα PREn (το n δείχνει σήμα αρνητικής λογικής), ενώ σε κάθε άλλη περίπτωση διαβάζει από αυτή την αρτηρία.


P1
JP1
VCC
C1
CAP
C2
CAP NP
C3
CAP NP
C4
CAP NP
C5
CAP NP
C6
CAP NP
C7
CAP NP
C8
CAP NP
CPCYL1/D.300/LS.200/.034
SMC_1206
HEADER 2
JUMPER200
PWEn
PREn
PCLK
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
PD[7:0]
CONNECTOR DB25
DSUB/VP/TMV25
U1
U2
U3
U6
U7
74ACT175
SOJ.050/16/W/B.300/L.400
74ACT244
DIP.100/20/W.300/L1.100
74ACT283
DIN0
DIN1
DIN2
DIN3
DIN4
DIN5
DIN6
DIN7
DIN[7:0]
DOUT0
DOUT1
DOUT2
DOUT3
DOUT4
DOUT5
DOUT6
DOUT7
DOUT[7:0]
CLK
CLR
VCC
WEn
REn

Αυτό το σχήμα διαιτησίας επιτυγχάνεται μέσω των ολοκληρωμένων ACT244 που είναι απομονωτές και οδηγοί γραμμών τριών καταστάσεων. Τα ολοκληρωμένα αυτά αντιγράφουν τα δεδομένα εισόδου τους (γραμμές A) στις αντίστοιχες εξόδους (γραμμές Y) μόνο όταν τα αντίστοιχα σήματα ενεργοποίησης G (αρνητικής λογικής) είναι στο λογικό 0. Αναφερόμενοι στο σχηματικό βλέπουμε ότι οι γραμμές ελέγχου επιτρέπεται διαρκώς να περνάνε στο hasp, ενώ τα δεδομένα εισόδου περνάνε στο hasp μόνο όταν ενεργοποιηθεί το σήμα PWEn. Το hasp οδηγεί με τα κωδικοποιημένα δεδομένα του την αρτηρία μόνο όταν το PEn είναι ενεργό (προσέξτε τον προσανατολισμό του U6).

Οταν νέα δεδομένα φτάσουν στο hasp, αυτά κλειδώνονται στα 175 (τετραπλά D flip flops) βάσει της ανοδικής ακμής του ρολογιού. Η ανοδική ακμή αυτή προκαλείται από το λογισμικό. Οι έξοδες των 175 μετατρέπονται σε XS127 κώδικα μέσω των δύο προσθετών (74ACT283) τεσσάρων δυαδικών ψηφίων που συνδέονται με ripple-carry σχήμα.

Στο όλο σχηματικό έχουν προστεθεί 7 decoupling πυκνωτές (C2 έως C8), ένας πυκνωτής ταντάλιου (C1) για τη σταθεροποίηση της τάσης και μια υποδοχή (header 2 pins) για τροφοδοσία και γή. Βεβαιωθείτε ότι όλα τα σύμβολα τροφοδοσίας και γής στο σχηματικό σας έχουν μετονομαστεί σε VCC και GND. Αποθηκεύστε το σχεδιασμό σας, ανανεώστε τις αναφορές των ολοκληρωμένων που χρησιμοποιήσατε, περάστε Design Rule Check … και πριν προχωρήσετε παρακάτω, αντιγράψτε το σχεδιασμό σας σε ένα νέο υποκατάλογο.

Β. Εξομοίωση του σχεδιασμού

Πριν την υλοποίηση του σχεδισμού, θα πρέπει να εξομοιώσετε αναλυτικά τη λειτουργία του. Απομακρύνετε τον connector P1, τους πυκνωτές και τον header, από το ένα αντίγραφο. Αντικαταστήστε τα σήματα του P1 με ιεραρχικά ports. Εξομοιώστε το σχεδιασμό σας γράφοντας τις κατάλληλες κυματομορφές εισόδου, οδηγώντας όμως την αρτηρία PD MONO όταν ενεργοποιείτε το σήμα εγγραφής. Σε όλες τις υπόλοιπες περιπτώσεις οδηγήστε την στην κατάσταση υψηλής εμπέδησης (Z). Σημειώστε τους χρόνους που χρειάζεται το σχηματικό σας για την κωδικοποίηση των δεδομένων. Αυτοί οι χρόνοι είναι σημαντικοί, μιας και θα πρέπει να γίνονται σεβαστοί από το λογισμικό ΑΝΕΞΑΡΤΗΤΑ από το πόσο γρήγορα ή αργά τρέχει το λογισμικό μας. Παραδοτέο αυτής της υποενότητας είναι το αρχείο των κυματομορφών εισόδου-εξόδου στο οποίο έχετε επιτύχει τη μέγιστη συχνότητα λειτουργίας του σχεδιασμού. Ποιά είναι αυτή η συχνότητα ? Σε περίπτωση που κάπου διαπιστώσατε λάθος στο σχεδιασμό σας, μη ξεχάσετε να ανανεώσετε και τα δύο αντίγραφα.

Γ. Φυσική υλοποίηση του σχεδιασμού

Επιλέξτε το αρχικό αντίγραφο του σχεδιασμού σας, δηλαδή εκείνο που έχει τους connectors και όχι τα ιεραρχικά ports. Για τη φυσική υλοποίηση του σχεδιασμού, χρειάζεται να δημιουργήσουμε ένα netlist του σχεδιασμού μας και κατόπιν να καλέσουμε το εργαλείο Layout. Ωστόσο, ακόμη δεν έχουμε ορίσει τα footprints των components που έχουμε χρησιμοποιήσει. Αυτό μπορεί να γίνει είτε μέσω του Layout, όταν φορτώνουμε το netlist είτε στο ίδιο το σχηματικό. Εμείς ακολουθούμε το δεύτερο τρόπο αν και μπορείτε να πειραματειστείτε με τον άλλο.

Καλέστε το εργαλείο Layout. Δώστε File->New και cancel για να μην επιλέξετε καμμία έτοιμη τεχνολογία. Ενώ είναι επιλεγμένο το κενό σχεδιαστικό φύλλο, επιλέξτε Tool ->Library Manager. Στο καινούργιο φύλλο που ανοίγει, βλέπετε αριστερά τις βιβλιοθήκες footprints που υπάρχουν. Επιλέγοντας κάποια βιβλιοθήκη, κάτω αριστερά εμφανίζονται όλα τα footprints που υπάρχουν στη συγκεκριμένη βιβλιοθήκη. Χτυπήστε σε κάποιο από αυτά για να το δείτε στο δεξί παράθυρο.

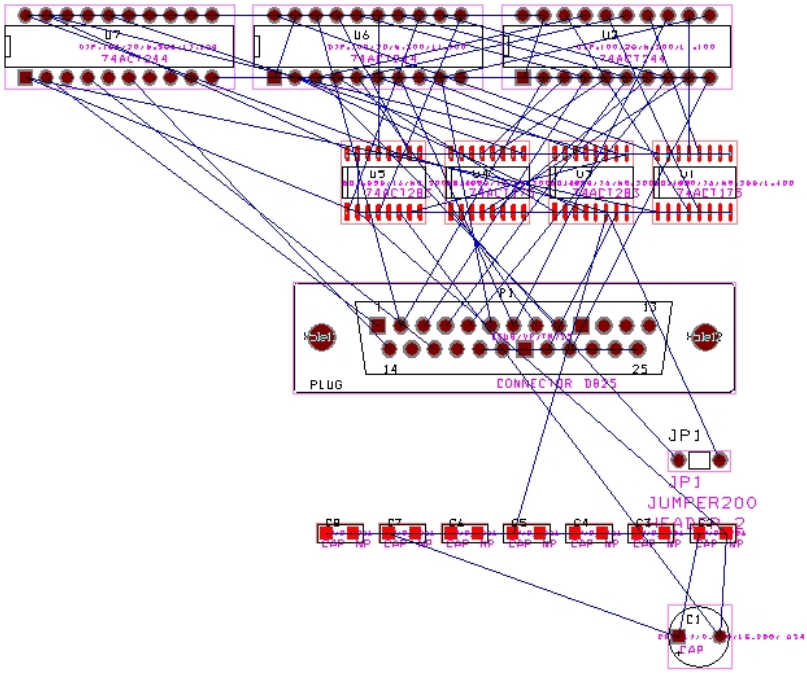
Ας γυρίσουμε στο σχεδιασμό μας, για να δούμε τι packages θα επιλέξουμε. Ξεκινώντας από τους πυκνωτές, βλέπουμε ότι θέλουμε through hole πυκνωτή για το C1, ενώ για τους C2 έως C8 μπορούμε να επιλέξουμε πυκνωτές surface mount για μείωση του εμβαδού της πλακέτας. Επιλέξτε συνεπώς στο Layout τη βιβλιοθήκη TM_CAP_P. Εκεί υπάρχουν δεκάδες footprints για διαφορετικούς πυκνωτές. Κάθε footprint ορίζει το σχήμα του πυκνωτή, την απόσταση μεταξύ των ακροδεκτών του, τις διαμέτρους των τρυπών κ.ο.κ. Επιλέγουμε ένα footprint που αντιστοιχεί σε κύλινδρικό πυκνωτή με μεγάλη απόσταση και μεγάλη διάμετρο τρυπών, π.χ. το CPCYL1/D.300/LS.200/.034. Προσέξτε ότι ο πυκνωτής έχει πολικότητα που υποδεικνύεται από τα +/- στο αποτύπωμα. Κάντε Control-C το όνομα του αποτυπώματος, γυρίστε στο σχεδιασμό και μέσω του Edit Properties, User Properties, PCB Footprint, New, Control-V, επισυνάψτε στο συγκεκριμένο component το συγκεκριμένο αποτύπωμα. Από τη βιβλιοθήκη SM (surface mount) επιλέξτε το αποτύπωμα για τους υπόλοιπους πυκνωτές (άνευ πολικότητας). Προτεινόμενο αποτύπωμα είναι το SM/C_1206. Ωστόσο μπορείτε να επιλέξετε οποιοδήποτε άλλο νομίζετε κατάλληλο. Αφού επισυνάψετε κι αυτό το αποτύπωμα στο σχηματικό σας, μπορείτε να αντιγράψετε τον πυκνωτή στις υπόλοιπες θέσεις ώστε να αποφύγετε την επανάληψη της διαδικασίας για κάθε πυκνωτή.

Με αντίστοιχο τρόπο επιλέγονται τα υπόλοιπα αποτυπώματα που φαίνονται στο σχηματικό. Προσέξτε ότι για τα 244 έχουμε επιλέξει through hole packages ενώ για τα υπόλοιπα surface mount packages. Η επιλογή αυτή δεν είναι τυχαία. Προσέξτε ότι αν για κάποιο λόγο δύο πηγές ταυτόχρονα προσπαθήσουν να οδηγήσουν την

διαμοιραζόμενη αρτηρία PD, τότε αυτό πιθανότατα να οδηγήσει σε "κάψιμο" των 244. Επιλέγοντας through hole packages μπορούμε να μη κολλήσουμε τα ολοκληρωμένα κατευθείαν πάνω στην πλακέτα, αλλά να κολλήσουμε βάσεις ολοκληρωμένων στη πλακέτα και να τα αλλάζουμε κάθε φορά που καίγονται. Η πιθανότητα να καούν τα υπόλοιπα ολοκληρωμένα είναι μικρή και συνεπώς γι' αυτά επιλέγουμε surface mount packages για τη μείωση του εμβαδού. Τέλος σημειώνεται η χρήση της οικογένειας ACT (Advanced CMOS – TTL compatible) σε όλο το σχεδιασμό. Η οικογένεια αυτή προσφέρει κατανάλωση τεχνολογίας CMOS με εξόδους συμβατές με TTL και καθυστέρηση αντίστοιχη της ALS TTL τεχνολογίας.

Αφού ολοκληρώσετε την επιλογή των αποτυπωμάτων για όλα τα components του σχεδιασμού σας, αποθηκεύστε τον, ανανεώστε τις αναφορές σε αυτά και όταν το Design Rule Check… σας υποδείξει ότι δεν υπάρχουν λάθη στο σχεδιασμό σας, δημιουργήστε ένα νέο netlist. Καλέστε το Layout.

Θα επιλέξουμε σαν Template υλοποίησης το Metric.TCH. Το template αυτό έχει ήδη οριστεί να παρέχει δύο επίπεδα διαδρόμισης και δύο επίπεδα για τροφοδοσία και γή. Οι γραμμές σε κάθε επίπεδο έχουν οριστεί να έχουν πλάτος 0,254mm. Σαν netlist επιλέξτε αυτό που δημιουργήσατε πιο πάνω. Μετά την ανάγνωση όλης της πληροφορίας, θα πάρετε ένα σχηματικό που έχει τη μορφή :



Στο παραπάνω σχηματικό φαίνονται τα αποτυπώματα που έχετε επιλέξει καθώς και οι γραμμές διασύνδεσης που πρέπει να διαδρομιστούν. Πριν από όλα όμως ας κάνουμε μια σύντομη γνωριμία με το Template METRIC.

Επιλέξτε window -> Database Spreadsheets και εκεί Layers. Παρατηρείστε ότι χρησιμοποιούνται δύο επίπεδα για διαδρόμιση και τα δύο επόμενα σαν επίπεδα τροφοδοσίας και γείωσης. Επιλέξτε τα δύο τελευταία επίπεδα και απενεργοποιείστε τα αλλάζοντας τον τύπο τους από Plane σε Unused. Στην ουσία έχετε πλέον ένα νέο Template με μόνο δύο επίπεδα.

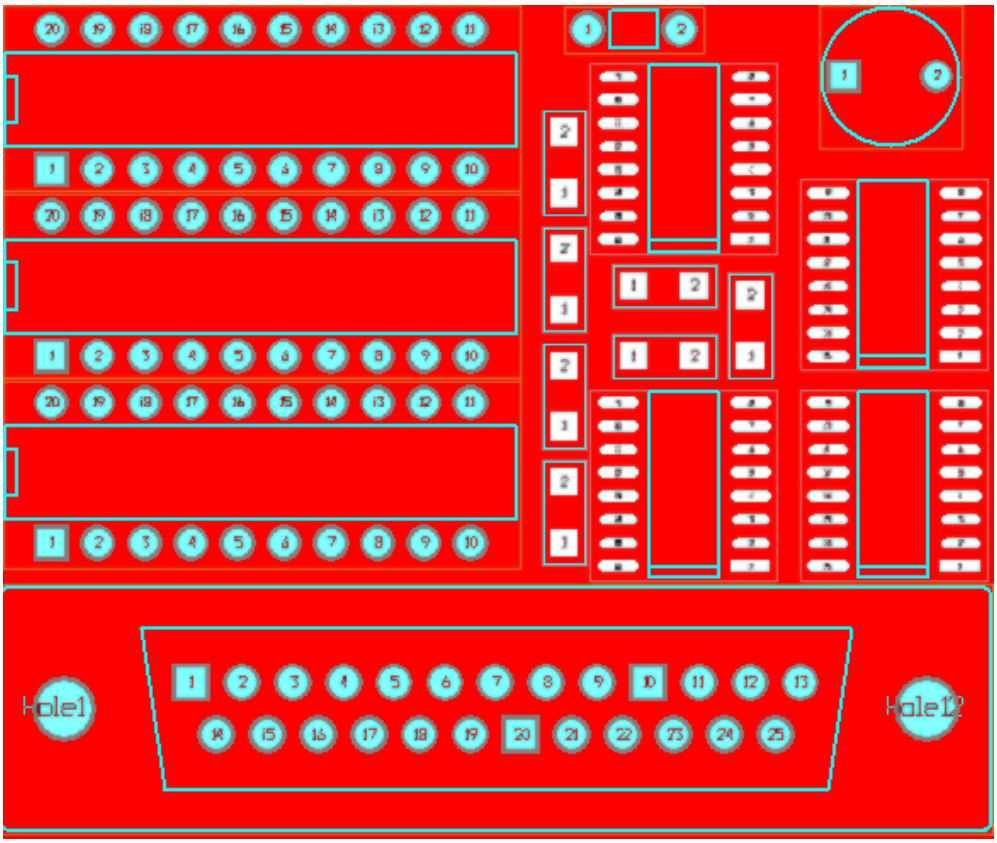
Επιλέγοντας Tool->Component, ξεκινείστε την τοποθέτηση του σχεδιασμού σας, έχοντας σα στόχους :

α) Το μικρότερο δυνατό εμβαδόν

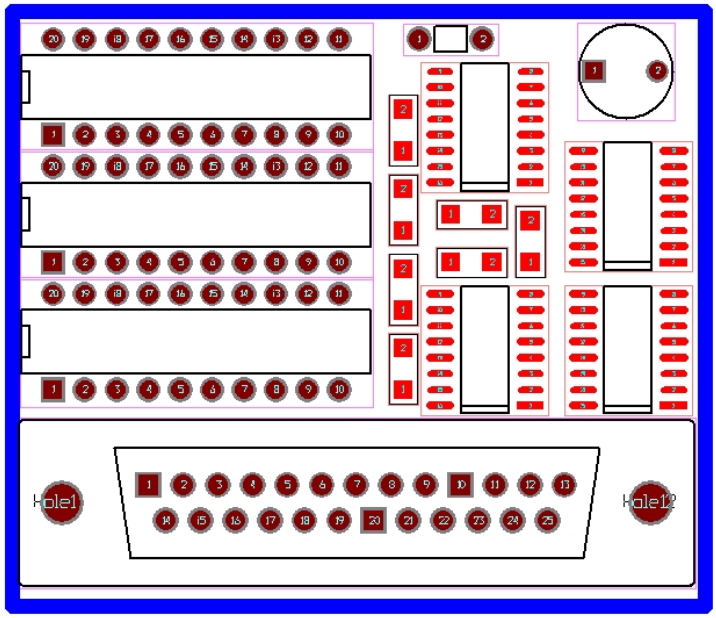
β) Οι decoupling πυκνωτές να είναι το δυνατόν κοντύτερα στα ολοκληρωμένα

γ) Ο ηλεκτρολυτικός πυκνωτής να είναι το δυνατόν κοντύτερα στο jumper τροφοδοσίας.

Μια πιθανή τοποθέτηση φαίνεται στο παρακάτω σχήμα :



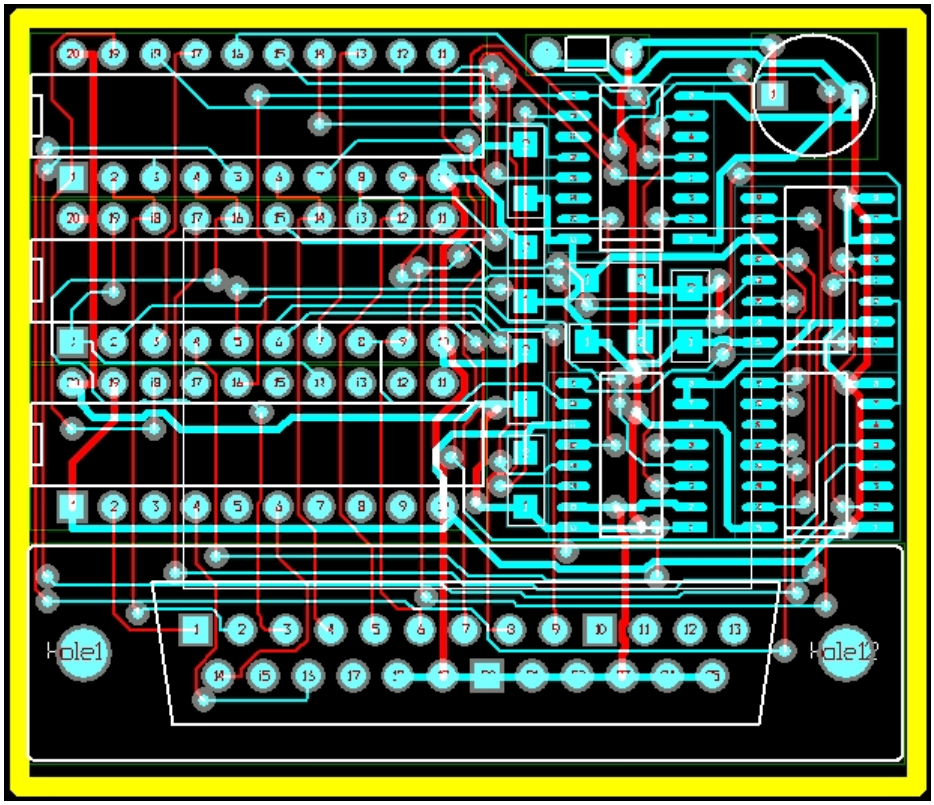
Πριν ξεκινήσουμε τη διαδρόμιση θα πρέπει να υποδείξουμε στο εργαλείο μας τα όρια της πλακέτας μας. Επιλέγουμε γι'αυτό Tool->Obstacle. Επίσης αλλάζουμε το επίπεδο που βλέπουμε στην οθόνη μας σε 0 (Global). Από το μενού που προκύπτει με δεξί κλίκ επιλέγουμε insert … και χαράζουμε τα όρια της πλακέτας μας, στα όρια των τοποθετημένων σχεδιασμών μας. Το σχηματικό μας θα πρέπει να είναι κάπως έτσι :



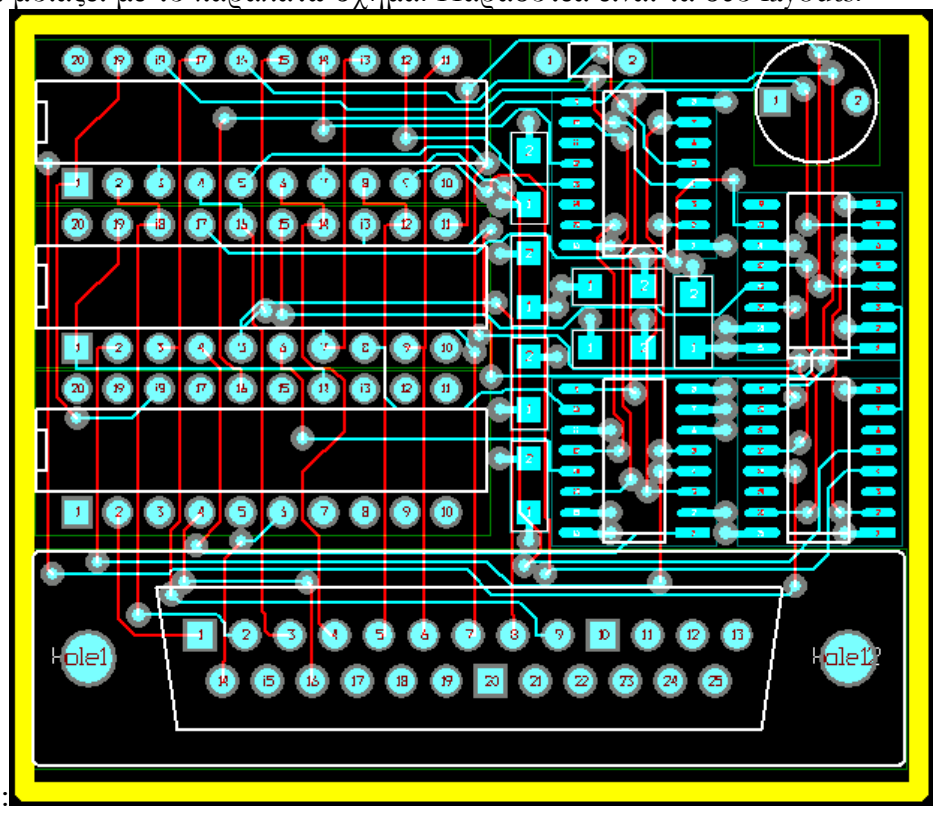
Ας ορίσουμε στη συνέχεια το πάχος των γραμμών που θέλουμε. Επιλέξτε Window->Database Spreadsheets και Nets. Από τα nets επιλέξτε τα σήματα VCC και GND και αλλάξτε το πλάτος τους σε .508 (δηλαδή το διπλό πλάτος). Επιλέξτε όλα τα σήματα χτυπώντας στη στήλη Net Name και αποεπιλέξτε το Routing Enabled. Επιλέξτε τα σήματα VCC και GND και αλλάξτε τα σε Routing Enabled, μιας και στο πρώτο πέρασμα θα κάνουμε διαδρόμιση μόνο των δύο σημάτων. Γυρνώντας στο σχηματικό, δείτε ότι μόνο αυτά τα δύο σήματα εμφανίζονται πλέον.

Διαδρομίστε αυτά τα δύο σήματα με το πλάτος που έχετε επιλέξει. Γυρίστε στον πίνακα των σημάτων, αλλάξτε τα δεδομένα διαδρόμισης έτσι ώστε να διαδρομιστούν όλα τα υπόλοιπα σήματα πλην των VCC και GND. Δώστε Auto->Route Window για να διαδρομιστούν τα υπόλοιπα σήματα. Πλέον θα πρέπει να έχετε έναν πλήρως τοποθετημένο και διαδρομημένο σχεδιασμό σαν τον παρακάτω (αν δεν μπορεί η διαδρόμιση να ολοκληρωθεί αυτόματα, επιλέξτε διαφορετική τοποθέτηση ή μεγαλώστε το μέγεθος της πλακέτας σας).

Αφού αποθηκεύσετε τον σχεδιασμό σας, επιλέξτε όλα τα nets από το spreadsheet και μετά clear tracks. Ετσι θα σβηστεί όλη η προηγούμενη διαδρόμιση. Σκοπός μας είναι πλέον να κάνουμε διαδρόμιση σε 4 επίπεδα, όπου τα δύο μεσαία επίπεδα θα είναι μόνο για τροφοδοσία και γή. Συνεπώς θα πρέπει να επιλέξουμε στα layers τα δύο επίπεδα σαν Planes. Στα nets επιλέγουμε μόνο το GND για routing και μέσω της επιλογής Net Layers αναθέτουμε αυτό το σήμα μόνο στο GND Plane Layer. Με αντίστοιχο τρόπο αναθέτουμε τη διαδρόμιση του VCC μόνο στο Power Plane Layer. Η διαδρόμιση σε αυτά τα επίπεδα γίνεται μέσω της εντολής Auto -> Fanout Board. Κάντε ορατά μόνο τα επίπεδα Power και GND (επίπεδα 3 και 4).



Εξηγείστε πως έχει γίνει η διαδρόμιση σε αυτά τα επίπεδα και γιατί είναι διαφορετική η τακτική για τα through hole και τα surface mount devices. Η διαδρόμιση στα υπόλοιπα επίπεδα γίνεται όπως προηγούμενα. Ο τελικός σχεδιασμός σας θα μοιάζει με το παρακάτω σχήμα. Παραδοτέα είναι τα δύο layouts.



ΕΡΓΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΣΕ FPGAs

Για τις ασκήσεις 5, 6, 7 και 8 κατά την τρέχουσα ακαδημαϊκή χρονιά, θα χρησιμοποιήσουμε τα εργαλεία :

(α) ModelSim 5.7f της Mentor Graphics και

(β) WebpackISE 13.1 της εταιρείας Xilinx,

καθώς και τις πλακέτες

(α) XSA-3S και

(β) XST-4 της εταιρείας Xess.com

Σκοπός όλων των ασκήσεων είναι το rapid system prototyping μέσω ανάπτυξης κώδικα HDL, σύνθεσης και προγραμματισμού ενός FPGA. Τόσο τα εργαλεία όσο και οι αναπτυξιακές πλακέτες έχουν τόσο πολλά χαρακτηριστικά που είναι αδύνατον να περιγραφούν πλήρως στα πλαίσια ενός μαθήματος. Παρακάτω συνεπώς χρησιμοποιούμε κάποια από εκείνα τα χαρακτηριστικά που είναι χρήσιμα για τις συγκεκριμένες ασκήσεις, αλλά θα πρέπει να είστε έτοιμοι να ανατρέξετε στα αντίστοιχα manuals ανά πάσα χρονική στιγμή.

Τα εργαλεία ModelSim και WebpackISE είναι διαθέσιμα για ακαδημαϊκούς λόγους από τα site www.model.com και www.xilinx.com οπότε μπορείτε μεγάλο μέρος των ασκήσεων να τις προετοιμάσετε σπίτι σας και απλά να τις επιβεβαιώνετε στις αναπτυξιακές πλακέτες. Ωστόσο λόγω του μεγέθους αυτών των εργαλείων, μπορείτε να τα προμηθευτείτε μαζί με τις εκπαιδευτικές άδειές τους από το γραφείο μου, προσκομμίζοντας ένα DVD.

Πριν από οποιαδήποτε εργασία σας στις πλακέτες, πρέπει οπωσδήποτε να διαβάσετε και να καταλάβετε τα αντίστοιχα manuals, που επισυνάπτονται. Το κόστος αυτών των πλακετών είναι ιδιαίτερα υψηλό και θα πρέπει να διαφυλάξετε την πλήρη λειτουργικότητά τους σας κόρη οφθαλμού. Επίσης, θα πρέπει κατά την επαφή σας με τις πλακέτες να είστε ιδιαίτερα προσεκτικοί, ώστε να μη βραχυκυκλώνετε με το άγγιγμά σας σημεία τα οποία δε θα πρέπει.

Στο εργαστήριο υπάρχει πάντα ένα ελεγμένο έτοιμο ζεύγος πλακετών προς χρήση, μαζί με το απαραίτητο τροφοδοτικό και το καλώδιο προγραμματισμού. Σε καμία περίπτωση δε θα πρέπει να προσπαθήσετε να διαχωρίσετε τις δύο αυτές πλακέτες, να αλλάξετε σχήμα τροφοδοσίας ή να αλλάξετε τρόπο προγραμαμτισμού του FPGA. Για όποιες ομάδες το επιθυμούν, μπορούν να παραχωρηθούν πλακέτες + τροφοδοτικά + καλώδια προγραμματισμού για το σπίτι με υποχρέωση καταβολής του κόστους (περίπου 650 Euro) σε περίπτωση επιστροφής κατεστραμένων υλικών. (Η κατάσταση μπορεί να ελεγχθεί άμεσα με ηλεκτρονικό τρόπο).

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 5

Σκοπός της εργαστηριακής άσκησης είναι μια πρώτη γνωριμία σας με την γλώσσα περιγραφής υλικού Verilog και την εξομοίωση. Στην άσκηση αυτή θα κληθείτε να περιγράψετε σε Verilog τόσο έναν μετρητή 8 δυαδικών ψηφίων όσο και τα διανύσματα εξομοίωσής του. (Στο εξής όλος ο κώδικας που παρατίθεται τις ασκήσεις, υπάρχει και στα αντίστοιχα subfolders του παρόντος).

Α. Περιγραφή του μετρητή

(αρχείο 5/counter.v)

```
module counter (clear, clock, load, start_stop, count, data);
 input [6:0] data;
 output [6:0] count;
 input start_stop;
 input load;
 input clock;
 input clear;
 reg [6:0] count;

 always @(posedge clock or posedge clear)
  if (clear) count <= 0;
   else if (load) count <= data;
          else if (start_stop) count <= count + 1;
endmodule
```

Ο κώδικας αυτός περιγράφει έναν μετρητή προς τα άνω των 7 δυαδικών ψηφίων με παράλληλη φόρτωση, είσοδο επίτρεψης και ασύγχρονη είσοδο καθαρισμού.

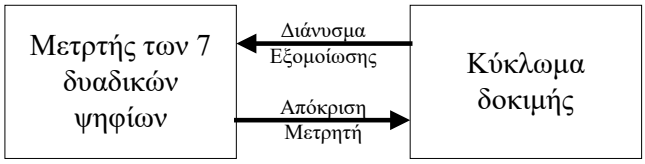
Β. Περιγραφή των διανυσμάτων εξομοίωσης

Για την εξομοίωση του κυκλώματος απαιτείται να περιγράψουμε και τα διανύσματα εξομοίωσης του μετρητή. Σε ένα καινούργιο αρχείο **(5/testcounter.v)** εισάγουμε τον ακόλουθο κώδικα :

```
module testcounter ();
reg [7:0] d;
wire [7:0] c;
reg s_s, l, clk, clr;

counter inst0 (clr, clk, l, s_s, c, d);
initial
 begin
  # 5 clk <= 0; clr <= 0; l <= 0;  s_s <= 0;  d <= 8'hF0;
 end
always # 40 clk <= !clk;
initial
  begin
   # 100 clr <= 1;
   #  50 clr <= 0;
  end
initial
 begin
  #  400 s_s <= 1;
  # 3000 s_s <= 0;
 end
initial
 begin
  # 3500 s_s <= 1;
  # 3700 l <= 1;
  #  200 l <= 0;
 end
endmodule
```

Ο κώδικας αυτός περιγράφει έναν tester για τον μετρητή σύμφωνα με το παρακάτω σχήμα :



Για τον σκοπό αυτό χρησιμοποιεί ένα αντίγραφο του μετρητή (inst0), κάποιους καταχωρητές (clr, clk, l, s_s,, d) για να εφαρμόσει το διάνυσμα δοκιμής και κάποια σήματα ώστε να διαβάσει την απόκριση του μετρητή (c). Ο υπόλοιπος κώδικας παράγει τα διανύσματα δοκιμής που ελέγχουν την λογική λειτουργία του μετρητή. Είμαστε πλέον έτοιμοι να προχωρήσουμε στην εξομοίωση του κυκλώματος.

Γ. Εξομοίωση του Σχεδιασμού

Για την διαδικασία εξομοίωσης θα χρησιμοποιήσουμε το εργαλείο ModelSim. Παρότι γνωρίζετε αρκετά τη χρήση του εργαλείου, τα βασικά βήματα επαναλαμβάνονται παρακάτω. Από το μενού File -> Change working directory, υποδείξτε στο εργαλείο το folder που περιέχει τα αρχεία σας. Από το μενού Library->Create New Library, δημιουργείστε μια καινούργια βιβλιοθήκη με όνομα π.χ. work. Σκοπός της βιβλιοθήκης είναι να κρατά όλα τα objects που θα δημιουργηθούν κατά την διαδικασία μετάφρασης του κώδικά σας. Με το πάνω αριστερά εικονίδιο της γραμμής εργαλείων καλείτε τον Verilog compiler, που όπως κάθε άλλος compiler θα ελέγξει το συντακτικό και την λογική του κώδικά σας και θα δημιουργήσει τα objects της βιβλιοθήκης. Υποδείξτε τα δύο αρχεία που θέλετε να μεταφραστούν και εκτελέστε την διαδικασία μετάφρασης. Οταν αυτή ολοκληρωθεί χωρίς λάθη είστε έτοιμοι να προχωρήσετε στην διαδικασία παραγωγής κυματομορφών. Σε περίπτωση λαθών, ο μεταφραστής υποδεικνύει την γραμμή λάθους και μπορείτε να διορθώσετε τα λάθη σας με το button edit source. Ο editor που θα παρουσιαστεί επιτρέπει την διαδικασία cross probing με τον μεταφραστή και είναι ειδικά διαμορφωμένος ώστε να αναλύει τα keywords και τις σταθερές που χρησιμοποιείτε και να τις παρουσιάζει με διαφορετικά χρώματα σε σχέση με τις μεταβλητές σας.

Εχοντας ολοκληρώσει την διαδικασία μετάφρασης, στην βιβλιοθήκη σας έχουν δημιουργηθεί τα δύο objects testcounter και counter όπως μπορείτε να διαπιστώσετε με την εντολή Library -> View Library Contents. Από το μενού File -> Load New Design επιλέξτε το object testcounter. Επειδή στον κώδικα αυτού του object είχατε συμπεριλάβει και ένα αντίτυπο του μετρητή, το εργαλείο θα φορτώσει αυτόματα και το object του μετρητή σας. Από το μενού View επιλέξτε structure και θα εμφανιστεί ένα παράθυρο που σας δείχνει την ιεραρχία του σχεδιασμού σας. Επιλέγοντας signals θα δείτε όλα τα σήματα που αναφέρονται στο συγκεκριμένο object της ιεραρχίας που έχετε επιλέξει καθώς και οι τιμές τους την τρέχουσα χρονική στιγμή. Εφόσον δεν έχετε τρέξει εξομοίωση όλες οι τιμές είναι σε x (άγνωστη) κατάσταση. Επιλέξτε όλα τα σήματα του testcounter και στο παράθυρο των σημάτων δώστε την εντολή View -> Wave -> Selected Signals.

Αυτόματα ανοίγει ένα παράθυρο κυματομορφών για τα επιλεγμένα σήματα. Για να τρέξετε εξομοίωση χρησιμοποιείστε το πρώτο από τα τέσσερα τελευταία buttons του παραθύρου κυματομορφών. Με αυτό θα πάρετε εξομοίωση 100 ns και μπορείτε να το χρησιμοποιείτε επαναληπτικά. Με το View -> Source μπορείτε να βλέπετε τον κώδικά σας. Ακριβώς όπως ένα debugging εργαλείο σας επιτρέπει να κάνετε step κατά την εκτέλεση του κώδικά σας, το ίδιο ισχύει και για τον συγκεκριμένο εξομοιωτή. Μπορείτε να επιλέξετε step με ένα από τα δύο buttons του παραθύρου κώδικα και το βέλος στα αριστερά δείχνει το επόμενο σημείο του κώδικα προς εκτέλεση. Επιπλέον το εργαλείο σας ανακοινώνει περί της χρονικής στιγμής που θα συμβεί το επόμενο γεγονός. Λόγω της παραλληλίας του υλικού, πιθανόν να υπάρχουν περισσότερα του ενός βέλη. Συνεχίστε την εξομοίωσή σας μέχρι να έχετε αποτελέσματα για μια πλήρη περιοχή τιμών του μετρητή (μέχρι να πάρει όλες τις 128 διαφορετικές τιμές του. Τι συμβαίνει μετά την τιμή 127 και γιατί;). Ελέγξτε τα αποτελέσματα της εξομοίωσης και βεβαιωθείτε ότι όλες οι λογικές λειτουργίες που προβλέφθηκαν από τα διανύσματα εξομοίωσης υλοποιούνται σωστά. Αν θέλετε μπορείτε να προσθέσετε ή να τροποποιήσετε τα διανύσματα εξομοίωσης μέσω κώδικα στο αρχείο testcounter.v. Δεν χρειάζεται να βγείτε από το εργαλείο για κάτι τέτοιο. Ωστόσο, αφού αλλάξετε τον κώδικα, χρειάζεται να αποθηκεύσετε τις αλλαγές και να ξανακάνετε μετάφραση. Ακολούθως απλά επιλέξτε από το File -> Restart για να ξεκινήσει η διαδικασία εξομοίωσης πάνω στον καινούργιο κώδικα.

Δ. Ζητούμενα της άσκησης

(1) Τροποποιείστε τον δεδομένο κώδικα ώστε να υλοποιεί έναν μετρητή των 4 δυαδικών ψηφίων με τα ίδια χαρακτηριστικά.
(2) Χρησιμοποιείστε δύο τέτοιους μετρητές όση λογική κρίνετε απαραίτητη για να υλοποιήσετε έναν μετρητή των 8 δυαδικών ψηφίων. Που θα πρέπει να συνδεθεί η είσοδος επίτρεψης του high order nibble ;
(3) Δημιουργείστε ένα σύνολο διανυσμάτων εξομοίωσης για τον μετρητή των 8 δυαδικών ψηφίων.

(4) Εξομοιώστε τον σχεδιασμό σας και επαληθεύστε την λογική λειτουργία του.
(5) Τροποποιείστε τον κώδικά σας ώστε ο μετρητής να έχει σύγχρονη είσοδο καθαρισμού και επαναλάβετε τα β, γ και δ.
(6) Εισάγετε μια καθυστέρηση 20 χρονικών στιγμών στη λειτουργία του μετρητή των 4 δυαδικών ψηφίων. Ποια είναι τότε η μέγιστη συχνότητα λειτουργίας του μετρητή των 8 δυαδικών ψηφίων ;

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 6

Σκοπός της εργαστηριακής άσκησης είναι μια πρώτη γνωριμία σας με τις διαδικασίες του rapid system prototyping. Θα περιγράψουμε ένα κύκλωμα σε HDL, θα πάρουμε την ισοδύναμη υλοποίησή του σε CLBs μέσω σύνθεσης και τέλος θα φτιάξουμε ένα πρωτότυπο αυτού του κυκλώματος στις αναπτυξιακές μας πλακέτες.

Πριν από οτιδήποτε άλλο, πρέπει να διαβάσετε (ακόμη κι αν το έχετε κάνει ήδη πρέπει να το ξανακάνετε) τα manuals :

(α) xsa-3S-manual-v1_1.pdf

(β) xst-manual-v4_0.pdf

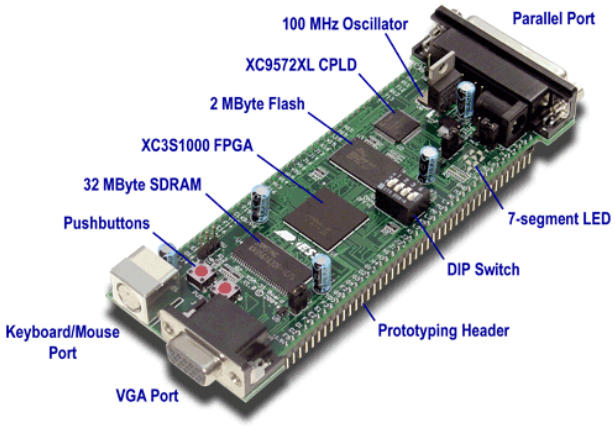
και να έχετε πάντοτε σε χρήση τα Excel αρχεία :

(α) XSA-3S-pins.xls

(β) XSA+XST4-pins.xls

Α. Σύντομη περιγραφή των αναπτυξιακών πλακετών XSA-3S και XSTend v.4.0

Για την πρωτοτυποποίηση των σχεδιασμών μας θα χρησιμοποιήσουμε ένα FPGA χωρητικότητας 1.000.000 ισοδύναμων πυλών, της σειράς SPARTAN 3 της εταιρείας Xilinx. Το FPGA αυτό για εργαστηρικούς λόγους διατίθεται και στη μορφή της αναπτυξιακής πλακέτας XSA-3S που έχει κατασκευάσει η εταιρεία XESS και που απεικονίζεται στην παρακάτω εικόνα :



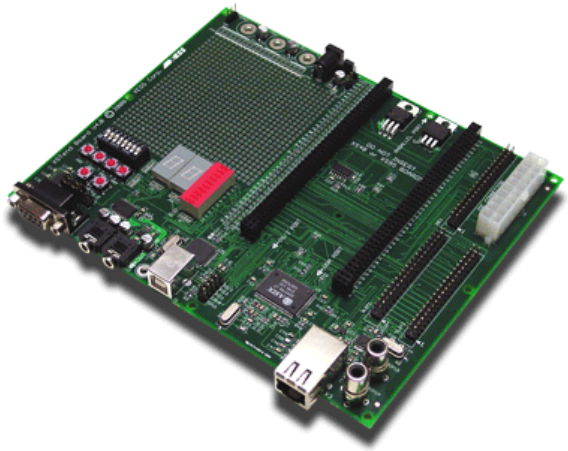
Στην πλακέτα αυτή προσφέρεται εκτός από το FPGA μια σειρά από interfaces διασύνδεσης (VGA, PS/2, Parallel Port) μαζί με τους απαραίτητους υποσχεδιασμούς τους καθώς και μια σειρά από σχεδιασμούς ελέγχου και παρατήρησης (dip switches, push buttons, 7 segment leds). Επίσης παρέχονται μνήμες :

(α) SDRAM, για bulk αποθήκευση ενδιάμεσων αποτελεσμάτων,

(β) Flash, για αποθήκευση κώδικα προγραμματισμού του ολοκληρωμένου ή για reconfiguration

και τέλος ένας προγραμματιζόμενος χρονιστής και ένα CPLD για τον έλεγχο των πηγών προγραμματισμού του FPGA και διάφορων άλλων λειτουργιών. Για όλες τις εργαστηριακές μας ασκήσεις καθώς και για την εξαμηνιαία εργασία σας, θα συνδέσουμε την πλακέτα αυτή με ένα προσωπικό υπολογιστή μέσω της παράλληλης θύρας και θα προγραμματίζουμε το FPGA με κώδικα που θα παράγεται στο PC και θα μεταβιβάζεται στο FPGA μέσω αυτής της σύνδεσης.

Παρότι η πλακέτα αυτή είναι standalone (χρειάζεται μόνο τροφοδοσία για να λειτουργήσει) μπορούμε να επεκτείνουμε επιπλέον τις δυνατότητές της προσκολλώντας την σε μια πλακέτα XSTend v.4.0 (XST4), η οποία μας δίνει επιπλέον interfaces (serial, audio USB, Ethernet, video), πάρα πολλούς ακροδέκτες παρατήρησης (headers) και σχεδιασμούς ελέγχου και παρατήρησης, χώρο για ανάπτυξη των δικών μας κυκλωμάτων και τη δυνατότητα τροφοδοσίας από ATX τροφοδοτικό. Η πλακέτα XST4 φαίνεται στην παρακάτω φωτογραφία:

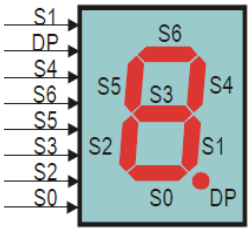


Στο εργαστήριο υπάρχει πάντα ένα ελεγμένο έτοιμο τέτοιο ζεύγος πλακετών προς χρήση, μαζί με το απαραίτητο τροφοδοτικό και το καλώδιο προγραμματισμού. Σε καμμία περίπτωση δε θα πρέπει να προσπαθήσετε να διαχωρίσετε τις δύο αυτές πλακέτες, να αλλάξετε σχήμα τροφοδοσίας ή να αλλάξετε τρόπο προγραμαμτισμού του FPGA.

Στην παρούσα εργαστηριακή άσκηση, θα χρησιμοποιήσουμε μόνο την πλακέτα XSA και θα χρησιμοποιούμε τη πλακέτα XST απλά για τροφοδοσία του συστήματός μας. Σκοπός του 1ου μέρους της άσκησης είναι να περιγράψουμε μια πύλη XOR 4 εισόδων, να τροφοδοτήσουμε τις εισόδους της από τα dip switches (υπενθυμίζεται : της πλακέτας XSA) και να βλέπουμε την έξοδο σαν αριθμό (0 ή 1) στο 7 segment.

Β. HDL περιγραφή του σχεδιασμού

Το κύκλωμα που θα χρειστεί να περιγράψουμε έχει 4 εισόδους (a, b, c, d που θα συνδεθούν στα dip switches) και 8 εξόδους (s[7:0] και dp που θα οδηγούν τα 8 led του 7 segment).



Θεωρούμε ότι η έξοδος s[i] οδηγεί το λαμπάκι Si της παραπάνω εικόνας και ότι η έξοδος dp το DP του 7 segment. Άρα για να βλέπουμε το 0, θα πρέπει να είναι s[0]=s[1]=s[2]=s[4]=s[5]=s[6] = 1 και s[3]=0, ενώ για να βλέπουμε το 1, θα πρέπει να είναι s[1]=s[4]=1 και όλα τα υπόλοιπα 0. Σε κάθε περίπτωση υποθέστε ότι dp=1.

(αρχείο 6/mlx.v)

```
module mlx (a, b, c, d, s, dp);
  input a, b, c, d;
  output [6:0] s;
  output       dp;
  wire o;

  assign o = a^b^c^d;
  assign s[6:0] = o ? 7'b0010010 : 7'b1110111;
  assign dp = 1;
endmodule
```

Γ. Περιγραφή των διανυσμάτων εξομοίωσης

Για την εξομοίωση του κυκλώματός μας, απλά βάζουμε όλες τις πιθανές τιμές στις εισόδους του :

(αρχείο 6/testmlx.v)

```
module testmlx ();
  reg a, b, c, d;
  wire [6:0] s;
  wire       dp;

  mlx i0 (a, b, c, d, s, dp);
  initial {a,b,c,d} <= 0;
```

```
always #100 {a,b,c,d} <= {a,b,c,d} + 1;
endmodule
```

Δ. UCF

Για τη σύνθεση του κυκλώματός μας, θα χρησιμοποιήσουμε το ISE της εταιρείας Xilinx (συγκεκριμένα την έκδοση WebPack, που είναι περιορισμένη ως προς τις σειρές FPGAs που υποστηρίζει, αλλά διατίθεται δωρεάν). Το εργαλείο αυτό είναι στην πράξη μια ολόκληρη πλατφόρμα εργαλείων (γραφικός editor + HDL editor + simulator + synthesis + place & route + bitgen), οπότε θα επικεντρωθούμε μόνο στα σημεία ενδιαφέροντος. Ενδεχόμενα να χρειαστεί να καταφύγετε στην online βοήθεια αρκετές φορές για να εντοπίσετε συγκεκριμένες ενέργειες του εργαλείου.

Για να έχουμε την ικανότητα υλοποίησης του κυκλώματός μας στην πλακέτα XSA θα πρέπει πέρα από τη περιγραφή του σχεδιασμού μας, να υποδείξουμε στο εργαλείο σύνθεσης, σε ποιους ακροδέκτες του FPGA θα συνδεθούν τα σήματα του σχεδιασμού μας. Αυτό είναι απαραίτητο, καθώς στην πλακέτα XSA μόνο συγκεκριμένα pins του FPGA συνδέονται με συγκεκριμένους άλλους σχεδιασμούς. Αυτή η διασύνδεση περιγράφεται αναλυτικά τόσο στο εγχειρίδιο της XSA όσο και στο αντίστοιχο Excel αρχείο.

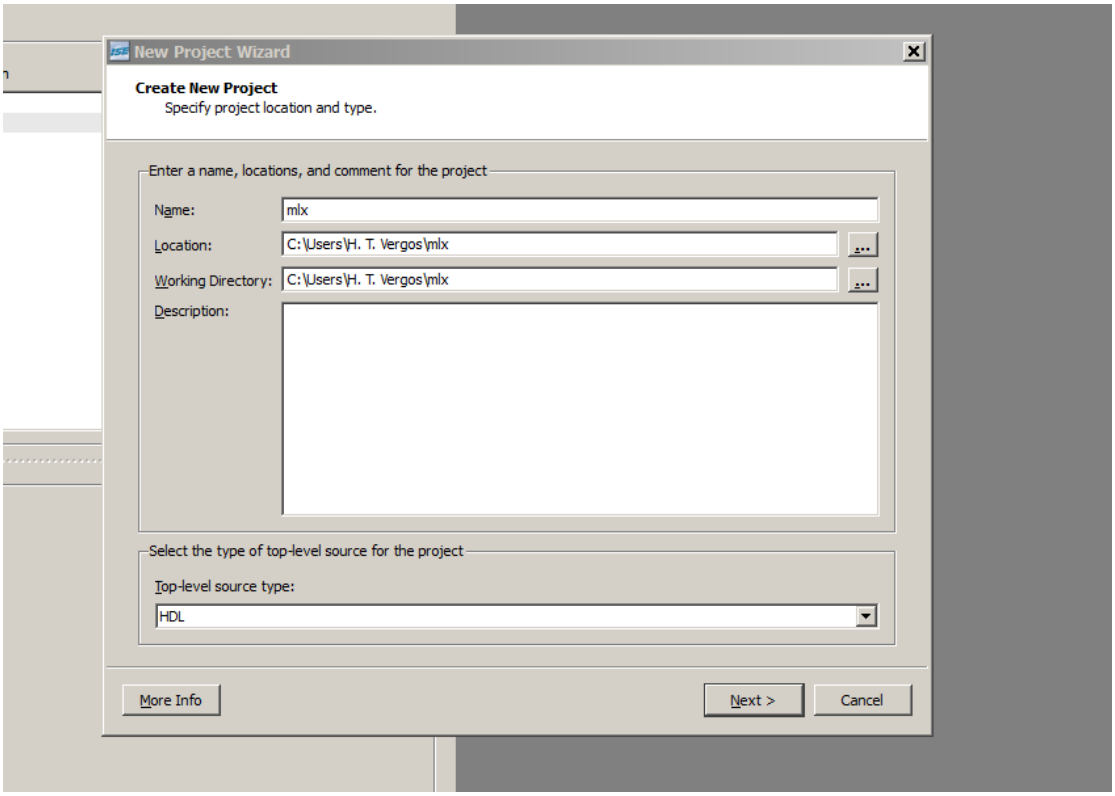
A	B	C	D	E	F	G	H	
Connections Between the FPGA, CPLD and other Components on the XSA-3								
Net Name	**FPGA Pin (U1)**	**CPLD Pin (U2)**	**Parallel Port**	**LEDs**	**Switch / Buttons**	**SDRAM (U4)**	**Flash (U3)**	**O**
FPGA-CCLK	T15	52						
FPGA-DIN-D0	M11	1		LED-C (S1)			DQ0 (29)	
FPGA-D1	N11	2		LED-DP			DQ1 (31)	
FPGA-D2	P10	4		LED-B (S4)			DQ2 (33)	
FPGA-D3	R10	5		LED-A (S6)			DQ3 (35)	
FPGA-D4	T7	7		LED-F (S5)			DQ4 (38)	
FPGA-D5	R7	9		LED-G (S3)			DQ5 (40)	
FPGA-D6	N6	10		LED-E (S2)			DQ6 (42)	
FPGA-D7	M6	11		LED-D (S0)			DQ7 (44)	
FPGA-DONE	R14	6						

Για παράδειγμα, στο απόσπασμα του Excel για την πλακέτα XSA της παραπάνω φωτογραφίας βρίσκουμε ότι το S0 led του 7 segment δυνδέεται με το ποδαράκι M6 του FPGA και συνεπώς πρέπει να ζητήσουμε από το εργαλείο της σύνθεσης να φτιάξει έτσι το κύκλωμά μας ώστε να συνδέσει το σήμα s[0] στο ποδαράκι M6. Όλοι αυτοί οι περιορισμοί που ζητάμε από το εργαλείο της σύνθεσης, αποτελούν ένα αρχείο περιορισμών (user constraints file) το οποίο θα πρέπει να επισυνάψουμε με το σχεδιασμό μας. Για το παράδειγμά μας, αυτό το αρχείο έχει ως εξής **(αρχείο 6/mlx.ucf)**

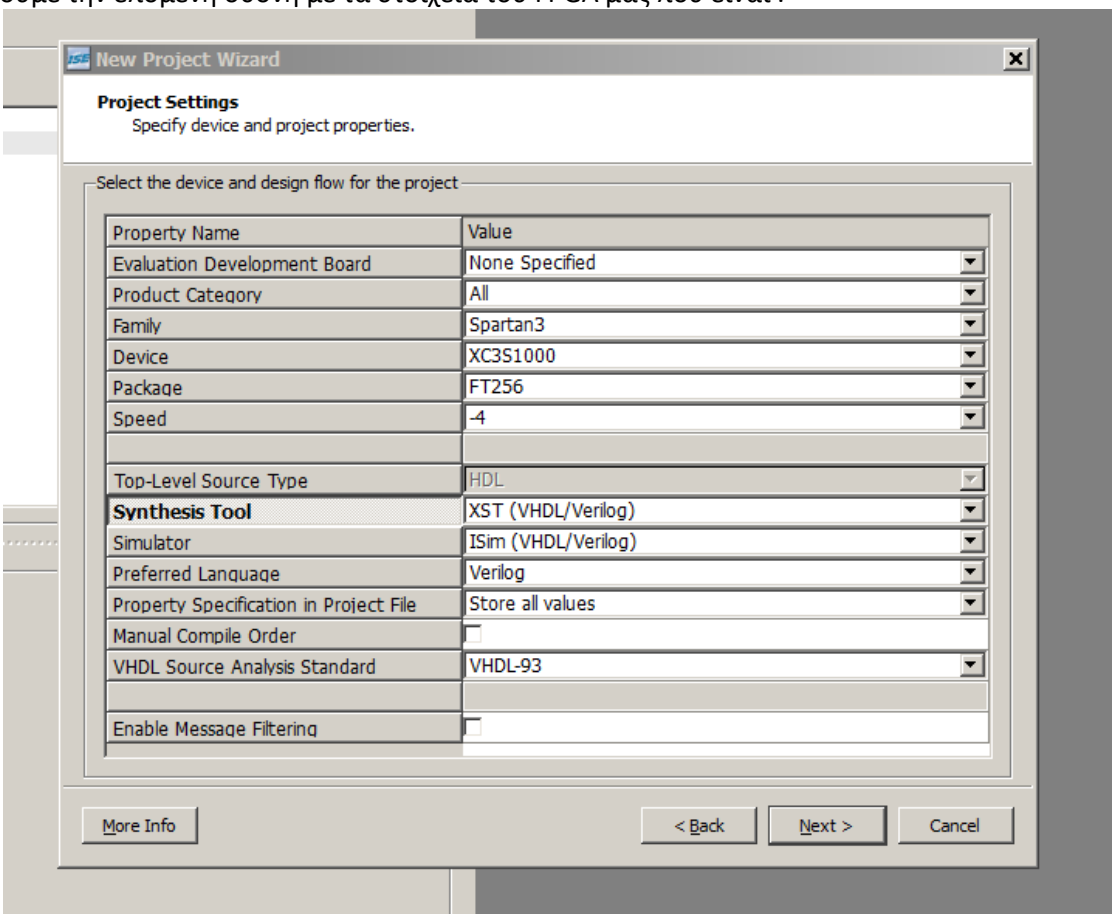
```
net a loc=k4;
net b loc=k3;
net c loc=k2;
net d loc=j4;
net s<6> loc=r10;
net s<5> loc=t7;
net s<4> loc=p10;
net s<3> loc=r7;
net s<2> loc=n6;
net s<1> loc=m11;
net s<0> loc=m6;
net dp loc=n11;
```

Ε. Σύνθεση του κυκλώματός μας και επισκόπηση των αποτελεσμάτων

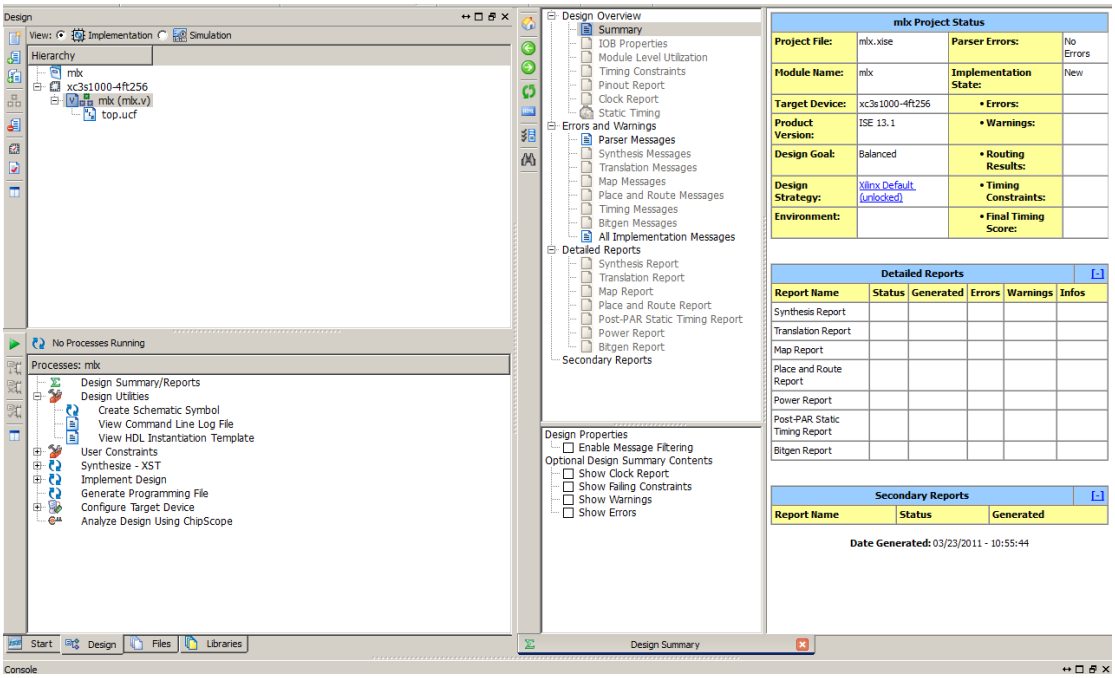
Ξεκινάμε το ISE Project Navigator που είναι ένα gui για όλο το φάσμα των εργαλείων του ISE. Επιλέγουμε New Project, δίνουμε ένα όνομα (π.χ. mlx) και έναν κατάλογο στον οποίο θα δημιουργηθεί ένα ολόκληρο folder με το παραπάνω όνομα για αυτό το project. Στο κάτω μέρος αφήνουμε τη default επιλογή ότι δηλαδή το κορυφαίο σχεδιαστικό μας κομμάτι θα είναι περιγεγραμμένο σε HDL (θυμηθείτε ότι το ISE έχει και γραφικό editor και πολλά άλλα).



Συμπληρώνουμε την επόμενη οθόνη με τα στοιχεία του FPGA μας που είναι :

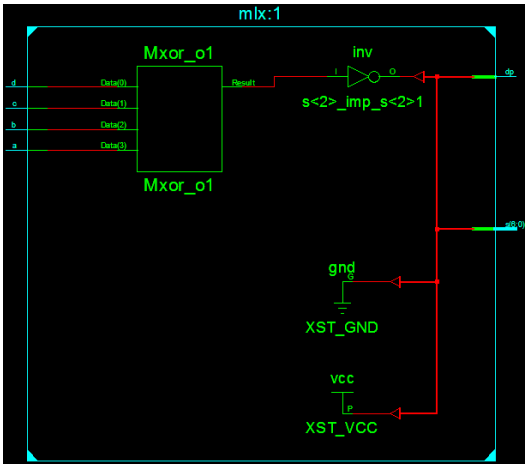


Λέμε δηλαδή στο εργαλείο ότι θα προγραμματίσουμε ένα XC3S1000 FPGA της Spartan 3 σειράς ταχύτητας 4 και σε tiny package 256 pins. Του λέμε επίσης ότι θα πρέπει να χρησιμοποιήσει το δικό του εργαλείο σύνθεσης και παρότι αφήνουμε το default εξομοιωτή, μπορούμε να του πούμε να διασυνδεθεί με την ήδη εγκατεστημένη έκδοση του ModelSim που τυχόν διαθέτουμε. Αφού αποδεχθούμε και την επόμενη διαλογική εικόνα, μεταφερόμαστε στο project navigator. Στη συνέχεια θα πρέπει να προσθέσουμε τα αρχεία μας στο project που μόλις δημιουργήσαμε μέσω του Project -> Add Source. Προσθέτουμε λοιπόν τόσο το mlx.v όσο και το mlx.ucf (όχι το testbench, αφού δεν περιγράφει τίποτε για τον σχεδιασμό μας) και έτσι καταλήγουμε στην εξής οθόνη :

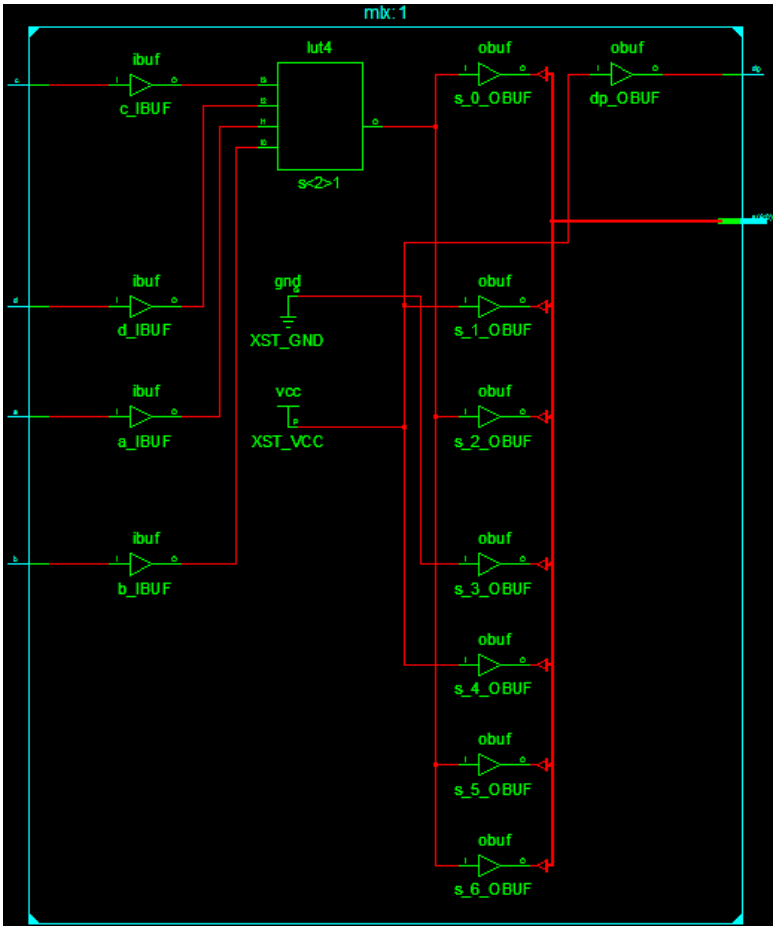


όπου πάνω αριστερά είναι η ιεραρχία του σχεδιασμού μας, κάτω αριστερά το τι μπορούμε να τρέξουμε ανάλογα με το τι θα επιλέξουμε από την ιεραρχία και δεξιά τα διάφορα reports που έχουν προκύψει.

Επιλέγουμε Process -> Implement Top module, και αυτόματα θα κληθούν τα εργαλεία μετάφρασης, σύνθεσης, απεικόνισης σε CLBs και Place & Route. Ας καλέσουμε κάποια από τα διαφορετικά εργαλεία για να δούμε τι ακριβώς μετατροπές έχουν γίνει στον σχεδιασμό μας. Επιλέγουμε Tools -> Schematic Viewer -> RTL και αφού αποδεχθούμε τις αρχικές επιλογές θα δούμε ένα σχηματικό παράθυρο του top level module. Χτυπώντας πάνω στο σύμβολο, θα δούμε την υλοποίησή του :

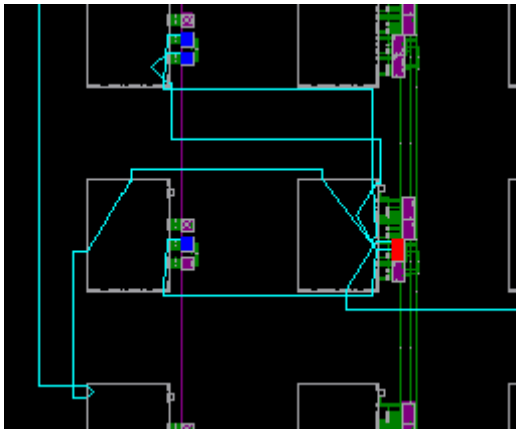


Κλείστε το παράθυρο του σχηματικού αυτού και επιλέξτε Tools -> Schematic Viewer -> Technology, αποδεχθείτε τις αρχικές επιλογές και πατήστε πάλι πάνω στο top level module του σχηματικού. Θα πάρετε έτσι το σχηματικό για τη συγκεκριμένη τεχνολογία που θα μιάζει με αυτό της εικόνας :

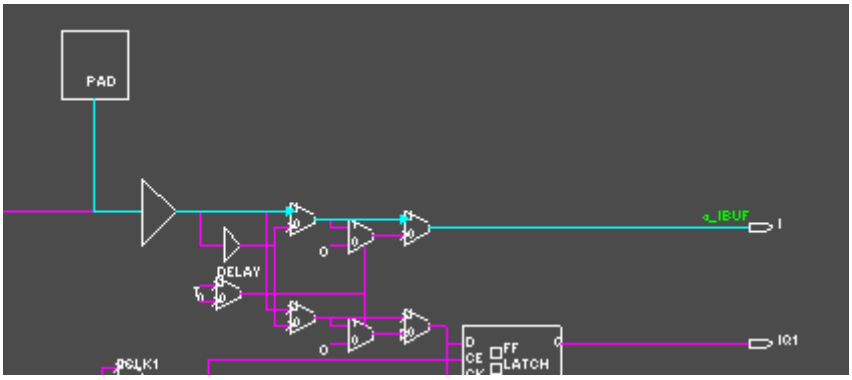


Δηλαδή, ο σχεδιασμός σας θα υλοποιηθεί με κάποιους input buffers, κάποιους output buffers και όλη σας η λογική με ένα LookUpTable 4 μεταβλητών. Ας δούμε στη συνέχεια, πως πράγματι αυτά μεταφράστηκαν στο πραγματικό κόσμο, εντός δηλαδή του FPGA που θέλαμε. Επιλέξτε Tools -> FPGA Editor -> Post Place & Route. Το εργαλείο που μας παρουσιάζεται, μας δείχνει την πραγματική υλοποίηση του κυκλώματός μας εντός του FPGA. Αφού μεγιστοποιήσετε το Array 1 υποπαράθυρο, από την πάνω πλευρά κάντε ορατά περισσότερα resources του FPGA κάνοντας τις εξής επιλογές :

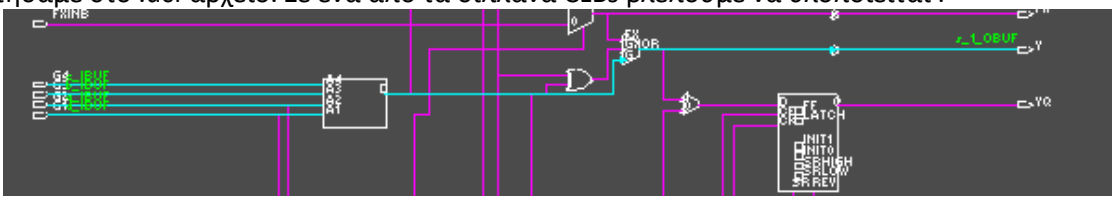
στα visibility options. Κάντε zoom in και προσπαθείστε να εντοπίσετε resources που χρησιμοποιούνται όπως αυτά της παρακάτω εικόνας στην αριστερή πλευρά του FPGA (τα πλέον αριστερά είναι IOB, ενώ το κόκκινο μέρος ενός CLB)



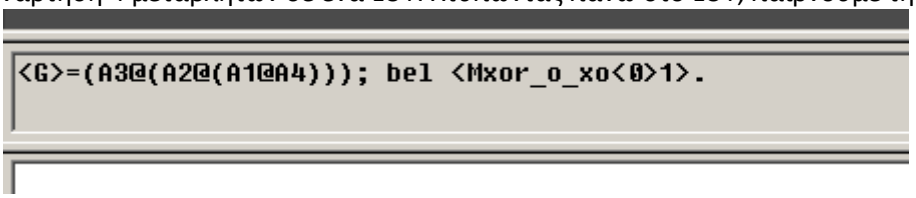
Χτυπώντας σε κάποιο από τα μπλε χρησιμοποιούμενα resources θα δείτε πως ακριβώς χρησιμοποιείται. Για παράδειγμα στην παρακάτω εικόνα :



φαίνεται ο σχεδιασμός που υλοποιείται εντός του ακροδέκτη k4. Υλοποιεί ένα pin εισόδου για το σήμα a όπως ακριβώς ζητήσαμε στο .ucf αρχείο. Σε ένα από τα διπλανά CLBs βλέπουμε να υλοποιείται :



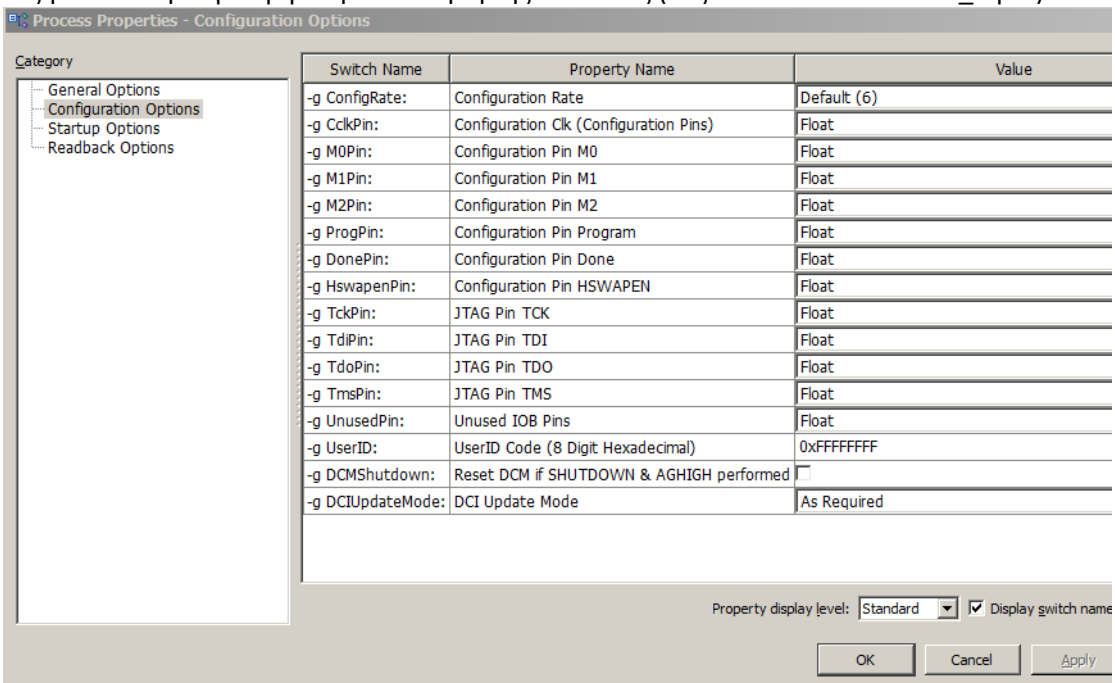
μια συνδυαστική συνάρτηση 4 μεταβλητών σε ένα LUT. Χτυπώντας πάνω στο LUT, παίρνουμε τη συνάρτησή του :



Αφού κλείσετε όλα τα εργαλεία τα οποία είδαμε ως τώρα, ας πάμε να προσδιορίσουμε τη συχνότητα λειτουργίας του σχεδιασμού μας. Θα χρησιμοποιήσουμε το Tools -> Timing Analyzer σε Post Place & Route mode. Στο παράθυρο που ανοίγει επεκτείνετε το Data Sheet report και επιλέξτε Pad to Pad. Εμφανίζονται τα χειρότερα μονοπάτια καθυστέρησης από κάθε είσοδο σε κάθε έξοδο (στο σχεδιασμό μας μόνο τα s[1] και s[4] αλλάζουν τιμές). Από εκεί προκύπτει ότι η χειρότερη καθυστέρηση του σχεδιασμού μας είναι 13.070ns.

ΣΤ. Υλοποίηση του κυκλώματός μας

Αφού ο σχεδιασμός μας έχει υλοποιηθεί, το επόμενο βήμα είναι η δημιουργία του αρχείου προγραμματισμού. Αφού κλείσετε όσα εργαλεία είναι ακόμη ανοιχτά, επιλέξτε στην ιεραρχία το mlx.v και στο κάτω αριστερά παράθυρο, κάντε δεξί κλικ στο Generate Programming File. Στο νέο παράθυρο που ανοίγει επιλέξτε Configuration Options και επιλέξτε να απομακρυνθούν όλες οι pull up αντιστάσεις που βάζει το εργαλείο στα configuration pins μιας και αυτό ζητείται στη συγκεκριμένη υλοποίηση της πλακέτας (δες το xsa-3S-manual-v1_1.pdf).



Αφού λοιπόν επιλέξουμε τις επιλογές που φαίνονται στην εικόνα, τις αποδεχόμαστε και κάνουμε διπλό κλικ στο Generate Programming file, οπότε και θα δημιουργηθεί ένα αρχείο mlx.bit στον κατάλογο του σχεδιασμού μας, που είναι το αρχείο προγραμματισμού. Ήρθε η ώρα να προγραμματίσουμε το FPGA μας.

Για την επικοινωνία της πλακέτας μας με το FPGA, θα χρησιμοποιήσουμε τα εργαλεία της XESS. Παρότι υπάρχουν αρκετά εμείς θα χρησιμοποιήσουμε μόνο το GXSLOAD. Δώστε λοιπόν τροφοδοσία στις πλακέτες σας και τρέξτε το GXSLOAD. Επιλέξτε Board Type XSA3S1000, Port επικοινωνίας το LPT1 και σύρτε το αρχείο mlx.bit εντός του χώρου FPGA/CPLD, για να δείξετε που θέλετε να αποθηκευτεί το αρχείο σας (με το ίδιο εργαλείο μπορούμε να προγραμματίσουμε και τις μνήμες της XSA). Πατείστε το Load και όταν ολοκληρωθεί η διαδικασία, έχετε πλέον ένα λειτουργικό πρωτότυπο !!! Χαρείτε το !!! Αλλάξτε κάποιο από τα dip switches (με τη βοήθεια της μύτης ενός μολυβιού) και δείτε πως αλλάζει και η ένδειξη στα leds. (Υπόδειξη: το λογικό 1 στα dip switches είναι μάλλον ανάποδα από ότι θα περιμένατε).

Ζ. Ζητούμενα

(1) Περιγράψτε ένα κύκλωμα το οποίο θα διαβάζει έναν δυαδικό αριθμό από τα dip switches και θα τον απεικονίζει στα led bars. Παραδοτέα : ο κώδικας περιγραφής, η μέγιστη συχνότητα λειτουργίας του κυκλώματός σας και το αρχείο προγραμματισμού.

(2) Περιγράψτε ένα κύκλωμα το οποίο θα πολλαπλασιάζει 2 αριθμούς, ο καθένας των 2 δυαδικών ψηφίων και θα απεικονίζει το αποτέλεσμά τους στο 7 segment. Εξετάστε 2 πιθανότητες : (α) οι αριθμοί να είναι μη προσημασμένοι και (β) οι αριθμοί να είναι σε παράσταση συμπληρώματος του 2. Χρησιμοποιείστε το DP του 7 segment για να υποδείξετε το πρόσημο του αποτελέσματος. Παραδοτέα για κάθε υποπερίπτωση : ο κώδικας περιγραφής, η μέγιστη συχνότητα λειτουργίας και το αρχείο προγραμματισμού.

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 7

Σκοπός της εργαστηριακής άσκησης είναι να φτιάξουμε ένα ακολουθιακό κύκλωμα χρησιμοποιώντας και τις 2 αναπτυξιακές μας πλακέτες ελαφρά μεγαλύτερης δυσκολίας από αυτό της προηγούμενης άσκησης. Αντικείμενο είναι αρχικά να φτιάξουμε έναν μετρητή δευτερολέπτων ο οποίος θα απεικονίζει τη μέτρησή του στα δύο 7 segment του XST board και κατόπιν να τον τροποποιήσετε ώστε να παρέχει επιπλέον δυνατότητες.

Α. Συχνότητα 1Hz

Η πλακέτα XSA έχει ενσωματωμένο έναν χρονιστή συχνότητας 100MHz. Συνεπώς για να πάρουμε την επιθυμητή συχνότητα του 1Hz θα πρέπει να διαιρέσουμε τη βασική με 10^8. Προφανώς αυτό μπορούμε να το πετύχουμε ανιχνεύοντας πότε ένας μετρητής με $\lceil \log_2 10^8 \rceil = 27$ δυαδικά ψηφία φθάνει σε μία συγκεκριμένη τιμή μέτρησης, αλλά κάτι τέτοιο θα απαιτούσε τη σύγκριση 27 ψηφίων, πράγμα που ενδεχόμενα να μας έριχνε σημαντικά τη μέγιστη συχνότητα λειτουργίας του σχεδιασμού μας και συνεπώς θα χάλαγε την ακρίβεια του ρολογιού μας. Συνεπώς θα φτιάξουμε μικρότερους μετρητές που η εξάντληση της ακολουθίας μέτρησης του ενός θα είναι η επίτρεψη μέτρησης του άλλου. Επειδή είναι $10^8 = 25^4 * 2^8$ θα φτιάξουμε ένα μετρητή mod25 και έναν 8 bit και θα χρησιμοποιήσουμε 4 από το πρώτο είδος και 1 από το δεύτερο σε μια αλυσίδα επίτρεψης.

Εδώ υπάρχει ένα κρυφό και ταυτόχρονα καίριο σημείο. Γιατί να φτιάξουμε μετρητές με είσοδο επίτρεψης και όχι απλούς μετρητές παίρνοντας τη πιο σημαντική έξοδο του ενός ως ρολόι του επόμενου ? Η απάντηση σε αυτό το ερώτημα έχει να κάνει με τους φυσικούς πόρους που διαθέτει το FPGA. Όπως εξηγήθηκε στο μάθημα, το FPGA διαθέτει ειδικούς buffers για σήματα ρολογιού, οι οποίοι όμως είναι περιορισμένοι. Αν συνεπώς χρησιμοποιήσουμε μόνο ένα ρολόι, θα χρησιμοποιηθεί γι' αυτό ο BUFGP (global buffer) με αποτέλεσμα να έχουμε το μικρότερο δυνατό skew ανάμεσα στο ρολόι που λαμβάνουν όλα τα flip flops του σχεδιασμού μας. Έτσι λοιπόν, ο κώδικας για τη συχνότητα 1 Hz είναι ο ακόλουθος :

(αρχείο 7/cnt25.v)

```
module cnt25 (reset, clk, enable, clkdiv25);
input reset, clk, enable;
output clkdiv25;
reg [5:0] cnt;

assign clkdiv25 = (cnt==5'd24);
always @(posedge reset or posedge clk)
  if (reset) cnt <= 0;
   else if (enable)
        if (clkdiv25) cnt <= 0;
          else cnt <= cnt + 1;
endmodule

module cnt8b (reset, clk, enable, clkdiv256);
input reset, clk, enable;
output clkdiv256;
reg [7:0] cnt;

assign clkdiv256 = (cnt==8'd255);
always @(posedge reset or posedge clk)
  if (reset) cnt <= 0;
   else if (enable) cnt <= cnt + 1;
endmodule


module OneHertz(reset, clk, en_nxt);
input clk, reset;
output en_nxt;
wire clk1Hz;
wire first, second, third, fourth;

cnt25 i0 (reset, clk, 1'b1, first);
cnt25 i1 (reset, clk, first, second);
```

```
cnt25 i2 (reset, clk, first & second, third);
cnt25 i3 (reset, clk, first & second & third, fourth);
cnt8b i4 (reset, clk, first & second & third & fourth, clk1Hz);
assign en_nxt = first & second & third & fourth & clk1Hz;
endmodule
```

Μπορούμε να επιβεβαιώσουμε τη λειτουργία του κυκλώματος παραγωγής της συχνότητας 1Hz, εξομοιώνοντάς το με τον ακόλουθο κώδικα :

```
module testOneHertz();
 reg reset, clk;
 wire en_nxt;

 OneHertz CUT (reset, clk, en_nxt);
 initial begin reset=0; clk = 0; # 10 reset = 1; # 9 reset = 0; end
 always #1 clk=~clk;
endmodule
```

Β. Μετρητής δευτερολέπτων

Για να μετρήσουμε τα δευτερόλεπτα, θα μπορούσαμε απλά να χρησιμοποιήσουμε έναν δυαδικό μετρητή 6 δυαδικών ψηφίων με αναδίπλωση μετά την τιμή 59. Ωστόσο για την απεικόνιση αυτής της λύσης στα δύο 7-segments θα έπρεπε μετά να υπολογίζουμε το πηλίκο και το υπόλοιπο της μέτρησης ως προς 10. Μια εναλλακτική πρόταση είναι να χρησιμοποιήσουμε 2 εναλλακτικούς μετρητές έναν για τις μονάδες των δευτερολέπτων και έναν για τις δεκάδες. Ο πρώτος είναι ένας δεκαδικός μετρητής, ενώ ο δεύτερος ένας μετρητής modulo 6. Ο κώδικας γι' αυτή τη δεύτερη λύση, είναι ο ακόλουθος (προσέξτε ότι εξακολουθούμε να χρησιμοποιούμε παντού τη λογική του ενός ρολογιού) :

(αρχείο 7/secondcounter.v)

```
module singleseconds (reset, clk, enable, ss, nxt);
 input reset, clk, enable;
 output [3:0] ss;
 output      nxt;
 reg    [3:0] ss;

 assign nxt= (ss == 4'd9);
 always @(posedge clk or posedge reset)
  if (reset) ss <= 4'd0;
    else if (enable)
         if (nxt) ss <= 0;
          else ss <= ss + 1;
endmodule

module tenthsofseconds (reset, clk, enable, ts);
 input reset, clk, enable;
 output [2:0] ts;
 wire        again;
 reg    [2:0] ts;

 assign again = (ts == 3'd5);
 always @(posedge clk or posedge reset)
  if (reset) ts <= 4'd0;
    else if (enable)
         if (again) ts <= 0;
          else ts <= ts + 1;
endmodule

module secondcounter (reset, clk, enable, ts, ss);
 input reset, clk, enable;
 output [2:0] ts;
```

```
output [3:0] ss;
wire       ent;

singleseconds   i0 (reset, clk, enable, ss, ent);
tenthsofseconds i1 (reset, clk, enable & ent, ts);
endmodule
```

του οποίου μπορείτε να επαληθεύσετε την ορθή λειτουργία με τον ακόλουθο κώδικα εξομοίωσης :

```
module test();
 reg reset, clk;
 wire [2:0] ts;
 wire [3:0] ss;

 secondcounter CUT (reset, clk, 1'b1, ts, ss);

 initial begin reset =0; clk = 0; #10 reset = 1; #9 reset = 0; end
 always #4 clk = ~clk;
endmodule
```

Γ. Απεικόνιση στα Leds

Χρειαζόμαστε τέλος ένα συνδυαστικό κύκλωμα που θα παίρνει στην είσοδο τη δυαδική μέτρηση και θα βγάζει στην έξοδο εξόδους για τα 7 segments. Ένα τέτοιο κύκλωμα είναι το εξής :

(αρχείο 7/bin_2_7.v)

```
module bin_2_7 (x, s);
 input  [3:0] x;
 output [6:0] s;

 assign s = (x == 4'd0 ) ? 7'b1111110 :  (x == 4'd1 ) ? 7'b0110000 :
        (x == 4'd2 ) ? 7'b1101101 : (x == 4'd3 ) ? 7'b1111001 :
        (x == 4'd4 ) ? 7'b0110011 : (x == 4'd5 ) ? 7'b1011011 :
        (x == 4'd6 ) ? 7'b1011111 : (x == 4'd7 ) ? 7'b1110010 :
        (x == 4'd8 ) ? 7'b1111111 : (x == 4'd9 ) ? 7'b1111011 :
        (x == 4'd10) ? 7'b1110111 : (x == 4'd11) ? 7'b0011111 :
        (x == 4'd12) ? 7'b1001110 : (x == 4'd13) ? 7'b0111101 :
        (x == 4'd14) ? 7'b1001111 : 7'b1000111 ;
endmodule
```

Δ. Top Level Module

Το τελευταίο module που χρειαζόμαστε είναι αυτό που ενώνει όλους τους υπόλοιπους σχεδιασμούς και είναι το κορυφαίο στην ιεραρχία μας. Αυτό θα περιέχει τους διαιρέτες συχνότητας, τον μετρητή δευτερολέπτων και 2 αντίγραφα του κυκλώματος απεικόνισης. Το κύκλωμα αυτό λαμβάνει μόνο reset και ρολόι και βγάζει τις εξόδους των δύο 7 segments. Ο κώδικας συνεπώς που χρειάζεται **(αρχείο 7/tops.v)** είναι :

```
module tops (reset, clk, left, right);
 input reset, clk;
 output [6:0] left, right;
 wire [2:0] ts;
 wire [3:0] ss;

 OneHertz      i0 (reset, clk, en_nxt);
 secondcounter i1 (reset, clk, en_nxt, ts, ss);
 bin_2_7       lt ({1'b0,ts}, left);
 bin_2_7       rt (ss, right);
endmodule
```

και όλο το κύκλωμά μας μπορεί να ελεγχθεί με το ακόλουθο testbench :

```
module test();
 reg reset, clk;
```

```
wire [6:0] left, right;

tops CUT (reset, clk, left, right);

initial
  begin
     reset = 0; clk =0;
   #2 reset = 1;
   #2 reset = 0;
  end

 always #5 clk = ~clk;
endmodule
```

Πλέον έχουμε ολοκληρώσει το σχεδιασμό μας και μπορούμε να προχωρήσουμε στην υλοποίησή του.

Ε. Αρχείο φυσικών παραμέτρων

Στη συνέχεια μπορούμε να προχωρήσουμε στον ορισμό των pins που θα μας χρειαστούν. Προσοχή χρειάζεται στο ότι πρέπει να χρησιμοποιήσουμε το Excel αρχείο του XST+XSA. Για το reset θα επιλέξουμε το 1ο dip switch της XSA πλακέτας και για ρολόι το pin t9 που οδηγείται από το CPLD (CLKA). Έτσι το αρχείο μας **(αρχείο 7/tops.ucf)** διαμορφώνεται στο εξής :

```
net clk loc=t9;
net reset loc=k4;
net left<6> loc=h14;
net left<5> loc=m4;
net left<4> loc=p1;
net left<3> loc=n3;
net left<2> loc=m15;
net left<1> loc=h13;
net left<0> loc=g16;
net right<6> loc=e2;
net right<5> loc=e1;
net right<4> loc=f3;
net right<3> loc=f2;
net right<2> loc=g4;
net right<1> loc=g3;
net right<0> loc=g1;
```

Ζ. Ζητούμενα

(1) Εφαρμόστε όλη τη ροή σύνθεσης και υλοποίησης του κυκλώματος όπως αυτή αναπτύχθηκε στην εργαστηριακή άσκηση 7 ώστε να καταλήξετε σε ένα λειτουργούν πρωτότυπο. Φωνάξτε τον επιβλέποντα, για να του το δείξετε.

(2) Επαυξήστε τις δυνατότητες του σχεδιασμού σας με τα ακόλουθα :

- Λειτουργία stop-watch : Η αλλαγή κάποιου διακόπτη παγώνει την ένδειξη δευτερολέπτων, αλλά το ρολόι σας συνεχίζει να μετράει κανονικά. Με την επαναφορά του διακόπτη, η ένδειξη συνεχίζει κανονικά από την πραγματική τιμή των δευτερολέπτων.
- Μέτρηση δεκάτων : Πέρα από δευτερόλεπτα το ρολόι σας είναι ικανό να μετρά και δέκατα του δευτερολέπτου. Η απεικόνιση γίνεται στα led bars δίπλα από τα 7 segment, όπου ακολουθείται η κωδικοποίηση θερμομέτρου, δηλαδή ο αριθμός των αναμένων leds αυξάνει προς κάποια διεύθυνση για κάθε δέκατο του δευτερολέπτου που περνά.

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 8

Σκοπός της εργαστηριακής άσκησης είναι να χρησιμοποιήσουμε ένα από τα πολλά interfaces τα οποία παρέχουν οι αναπτυξιακές μας κάρτες για επικοινωνία με κάποια περιφερειακή συσκευή. Συγκεκριμένα θα χρησιμοποιήσουμε ένα πληκτρολόγιο PS/2 που θα αποτελέσει συσκευή εισόδου στο σύστημά μας. Θα πρέπει να διαβάζουμε κάποιους συγκεκριμένους χαρακτήρες και να τους απεικονίζουμε στα 7 segment. Αρχικά θα μελετήσουμε το πρωτόκολλο PS/2 για επικοινωνία με το πληκτρολόγιο.

Α. Πρωτόκολλο πληκτρολογίου

Περιγράφεται παρακάτω η ανάγνωση δεδομένων από ένα πληκτρολόγιο τεχνολογίας AT, η οποία είναι ακριβώς ίδια με αυτήν του PS/2 πρωτοκόλλου. Η περιγραφή περιορίζεται μόνο στα απαραίτητα για το project στοιχεία. Ο ενδιαφερόμενος αναγνώστης μπορεί να ανατρέξει και στα :

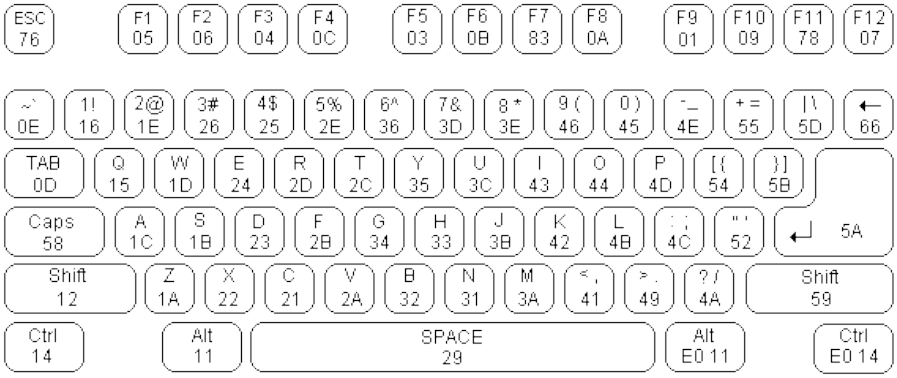
http://www.beyondlogic.org/keyboard/keybrd.htm

http://www.barcodeman.com/altek/mule/scandoc.php3

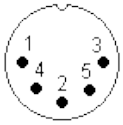
Σε κάθε πλήκτρο αντιστοιχεί ένας κωδικός 1 ή 2 ή 3 bytes που ονομάζεται κώδικας σάρωσης (scan code). Τα scan codes που μας ενδιαφέρουν στο παρόν project είναι :

Πλήκτρο	Scan Code (in HEX)
0	45
1	16
2	1E
3	26
4	25
5	2E
6	36
7	3D
8	3E
9	46

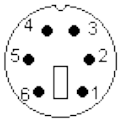
Οι κώδικες σάρωσης για ένα πληκτρολόγιο AT φαίνονται στη παρακάτω εικόνα.



Το πάτημα ενός πλήκτρου, ισοδυναμεί με τη δημιουργία και τη μεταφορά του αντίστοιχου κωδικού σάρωσης. Ο κωδικός σάρωσης θα συνεχίσει να παράγεται και να μεταδίδεται μέχρι να αφεθεί το πλήκτρο οπότε παράγεται ένας κωδικός αποτελούμενος από 2 bytes : το $F0_{16}$ και τον κωδικό σάρωσης. Συνεπώς ένα απλό πάτημα του "2", ισοδυναμεί με τη μεταφορά της πληροφορίας 1E F0 1E. Η μεταφορά των scan codes στο υπόλοιπο σύστημα γίνεται πάνω από μια σειριακή αρτηρία μέσω των γραμμών πληροφορίας (γραμμή KBD Data) όσο και ενός ρολογιού κωδικοποίησης (KBD Clock), μικρής συχνότητας (20 – 30 KHz). Τα σήματα που χρησιμοποιούνται σε 5 Pin DIN και PS/2 προσαρμογείς είναι όπως παρακάτω :



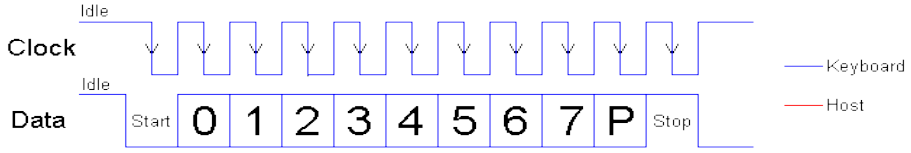
1. KBD Clock
2. KBD Data
3. N/C
4. GND
5. +5V (VCC)



1. KBD Clock
2. GND
3. KBD Data
4. N/C
5. +5V (VCC)
6. N/C

Το σειριακό πρωτόκολλο ανταλλαγής δεδομένων στην περίπτωση του πληκτρολογίου ακολουθεί τους εξής κανόνες :

- Οταν δεν ανταλλάσσονται δεδομένα η γραμμή δεδομένων (Data) είναι ανενεργή (στο λογικό 1).
- Για την αποστολή ενός byte δεδομένων απαιτείται η αποστολή ενός πακέτου 11 bits με χρονισμό που φαίνεται στην παρακάτω εικόνα.



- Το πρώτο δυαδικό ψηφίο της γραμμής δεδομένων είναι ένα ψηφίο εκκίνησης (start bit), ακολουθούμενο από τα 8 δυαδικά ψηφία του κώδικα σάρωσης, ένα ψηφίο περιττής ισοτιμίας και ένα δυαδικό ψηφίο τερματισμού (stop bit).
- Το πρώτο δυαδικό ψηφίο πληροφορίας που αποστέλλεται είναι το λιγότερο σημαντικό (τόσο για τον κώδικα σάρωσης, όσο και για το $F0_{16}$).
- Ο παραλήπτης της πληροφορίας για τη δειγματοληψία θα πρέπει να χρησιμοποιήσει την αρνητική ακμή του Clock. (Προσοχή αυτό δε σημαίνει ότι θα πρέπει στο σχεδιασμό σας να έχετε 2 ρολόγια, ένα του υπόλοιπου κυκλώματος και ένα για το πληκτρολόγιο).

Β. Υλοποίηση

Πριν τη συγγραφή του κώδικα, χρειάζεται να αναλυθούν περαιτέρω κάποια θέματα :

(1) Ο σχεδιασμός μας φαίνεται να έχει 2 ρολόγια, αυτό της αναπτυξιακής πλακέτας και αυτό του πληκτρολογίου. Ωστόσο, κάτι τέτοιο είναι απολύτως ανεπιθύμητο (μεγαλώνει το skew, χρησιμοποιούνται περισσότερα resources, είναι δύσκολο να προσδιοριστεί η συχνότητα λειτουργίας και είναι δύσκολος ο ακριβής συγχρονισμός μεταξύ υποσχεδιασμών με διαφορετικό χρονισμό). Επίσης, μιας και υπάρχει τάξεις μεγέθους διαφορά μεταξύ των συχνοτήτων αυτών των ρολογιών, υπάρχει περίπτωση ένα glitch στο αργό ρολόι να διαρκέσει δεκάδες κύκλους του γρήγορου. Για την αποφυγή όλων των παραπάνω, στον κώδικά μας πρέπει να έχουμε ένα μόνο ρολόι, αυτό της πλακέτας, το οποίο θα χρησιμοποιήσουμε και για δειγματοληψία του αργού (του πληκτρολογίου).

(2) Όπως αναλύθηκε παραπάνω, το γρήγορο πάτημα ενός χαρακτήρα, δημιουργεί την ακολουθία <scancode> <F0> <scancode>, ενώ το παρατεταμένο πάτημά του την ακολουθία <scancode><scancode> ... <scancode> <F0> <scancode>. Μια μεθοδολογία που μπορούμε να ακολουθήσουμε στο σχεδιασμό μας είναι να ανιχνεύουμε το <scancode> μόνο αμέσως το <F0>, έτσι ώστε να απεμπλακούμε από το φόρτο να κοιτάμε για πόσο πατήθηκε ένας χαρακτήρας ή να απορρίπτουμε τα 2 τελευταία σειριακά πακέτα κάθε πληκτρολόγησης.

(3) Το τελευταίο θέμα είναι τι γίνεται με τα υπόλοιπα πλήκτρα. Παρακάτω υποθέτουμε ότι γι' αυτά στην οθόνη μας όλα τα leds του 7 segment σβήνουν.

Μετά τα παραπάνω, ο κώδικας που χρειάζεται αρχικά να δοκιμάσουμε είναι ο :

(αρχείο 8/eight.v)

```
module kbd_protocol (reset, clk, ps2clk, ps2data, scancode);
 input       reset, clk, ps2clk, ps2data;
 output [7:0] scancode;
 reg    [7:0] scancode;
 // Synchronize ps2clk to local clock and check for falling edge;
 reg    [7:0] ps2clksamples; // Stores last 8 ps2clk samples
 always @(posedge clk or posedge reset)
  if (reset) ps2clksamples <= 8'd0;
   else ps2clksamples <= {ps2clksamples[7:0], ps2clk};
 wire fall_edge; // indicates a falling_edge at ps2clk
 assign fall_edge = (ps2clksamples[7:4] == 4'hF) & (ps2clksamples[3:0] == 4'h0);
 reg    [9:0] shift;   // Stores a serial package, excluding the stop bit;
 reg    [3:0] cnt;     // Used to count the ps2data samples stored so far
 reg          f0;      // Used to indicate that f0 was encountered earlier
/* A simple FSM is implemented here. Grab a whole package, check its parity validity and output it in
the scancode only if the previous read value of the package was F0 that is, we only trace when a
button is released, NOT when it is pressed. */
 always @(posedge clk or posedge reset)
  if (reset)
   begin
    cnt    <= 4'd0; scancode <= 8'd0; shift    <= 10'd0; f0 <= 1'b0;
   end
  else if (fall_edge)
     begin
```

```
            if (cnt == 4'd10) // we just received what should be the stop bit
             begin
              cnt <= 0;
              if ((shift[0] == 0) && (ps2data == 1) && (^shift[9:1]==1)) // A well received serial packet
               begin
                if (f0) // following a scancode of f0. So a key is released !
                 begin
                  scancode <= shift[8:1]; f0 <= 0;
                 end
                else if (shift[8:1] == 8'hF0) f0 <= 1'b1;
               end // All other packets have to do with key presses and are ignored
             end
            else
             begin
              shift <= {ps2data, shift[9:1]}; // Shift right since LSB first is transmitted
              cnt <= cnt+1;
             end
           end
endmodule

            module scan_2_7seg (scan, ss);
             input  [7:0] scan;
             output [7:0] ss;

             assign ss =     (scan == 8'h45) ? 8'b01111110 :
                             (scan == 8'h16) ? 8'b00110000 :
                             (scan == 8'h1E) ? 8'b01101101 :
                             (scan == 8'h26) ? 8'b01111001 :
                             (scan == 8'h25) ? 8'b00110011 :
                             (scan == 8'h2E) ? 8'b01011011 :
                             (scan == 8'h36) ? 8'b01011111 :
                             (scan == 8'h3D) ? 8'b01110010 :
                             (scan == 8'h3E) ? 8'b01111111 :
                             (scan == 8'h46) ? 8'b01111011 : 8'b10000000 ;
            endmodule

            module eight (reset, clk, ps2clk, ps2data, left);
             input      reset, clk;
             input      ps2clk, ps2data;
             output [7:0] left;
             wire   [7:0] scan;
             kbd_protocol kbd (reset, clk, ps2clk, ps2data, scan);
             scan_2_7seg  lft (scan, left);
            endmodule
```

Μερικές επιπλέον διευκρινίσεις για τον παραπάνω κώδικα. Ο σχεδιασμός μας αποτελείται από 2 υποσχεδιασμούς. Ο δεύτερος (scan_2_7seg) απλά διαβάζει ένα scancode και το μετατρέπει σε πληροφορία απεικόνισης σε 7 segment (θα χρησιμοποιήσουμε το αριστερό 7 segment της XST).

Ο πρώτος (kbd_protocol) υλοποιεί όλο το πρωτόκολλο ανάγνωσης από το πληκτρολόγιο. Χρησιμοποιούμε :

- τον καταχωρητή ps2clksamples για να κρατήσουμε τα τελευταία 8 δείγματα του ρολογιού του πληκτρολογίου. Παίρνουμε ένδειξη αρνητικής ακμής μόνον όταν αυτός έχει τη τιμή 11110000.
- έναν καταχωρητή ολίσθησης (shift) ο οποίος σε κάθε τέτοια αρνητική ακμή διαβάζει ένα δυαδικό ψηφίο δεδομένων.
- έναν μετρητή (cnt) για να διαπιστώσουμε πότε διαβάσαμε 11 ψηφία και εάν είναι όπως αναμενόταν (σωστά start – stop bit και σωστή ισοτιμία) τότε μόνο είναι είναι ένα σωστό σειριακό πακέτο με scancode υποψήφιο για έξοδο.

- ένα flag (f0) για να θυμόμαστε αν ο προηγούμενος scancode που ανιχνεύσαμε ήταν το f0. Μόνο σε αυτή την περίπτωση ανανεώνουμε την έξοδό μας.

Γ. Αρχείο φυσικών παραμέτρων

Μπορούμε πλέον να προχωρήσουμε στον ορισμό των pins που θα μας χρειαστούν. Προσοχή χρειάζεται στο ότι πρέπει να χρησιμοποιήσουμε το Excel αρχείο του XST+XSA. Για το reset θα επιλέξουμε το 1ο dip switch της XSA πλακέτας και για ρολόι το pin t9 που οδηγείται από το CPLD (CLKA). Έτσι το αρχείο μας διαμορφώνεται στα αναγραφόμενα στο **(αρχείο 8/eight.ucf)**

Δ. Ζητούμενα

(1) Εφαρμόστε όλη τη ροή σύνθεσης και υλοποίησης του κυκλώματος όπως αυτή αναπτύχθηκε στην εργαστηριακή άσκηση 7 ώστε να καταλήξετε σε ένα λειτουργούν πρωτότυπο. Φωνάξτε τον επιβλέποντα, για να του το δείξετε. Κρατήστε συνεχώς ένα αριθμητικό πλήκτρο πατημένο και δείτε τι συμβαίνει. Εξηγήστε το γιατί.

(2) Επαυξήστε τις δυνατότητες του σχεδιασμού σας με τα ακόλουθα :

- Λειτουργία προηγούμενου πλήκτρου : Χρησιμοποιείστε και το δεύτερο 7-segment της XSA για να απεικονίζετε το προηγούμενο πατηθέν πλήκτρο.
- Λειτουργία calculator : Δώστε τη δυνατότητα ανίχνευσης και των συμβόλων των αριθμητικών πράξεων με τους ακόλουθους scancodes (εισάγονται από το αριθμητικό πληκτρολόγιο πλην του /):

Πλήκτρο	Scan Code (in HEX)
+	79
-	7B
*	7C
/	4A

Όταν ανιχνευθεί ένας τέτοιος χαρακτήρας, εκτελείται η πράξη <πρόσφατο στοιχείο> <πράξη><προηγούμενο στοιχείο> και το αποτέλεσμα (για την πράξη * μόνο το least significant digit, για την / μόνο το ακέραιο μέρος) εμφανίζεται στο 7 segment της XSA.