Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Συγκριτική μελέτη αρχιτεκτονικών και εργαλείων σχεδίασης προγραμματιζόμενων διατάξεων λογικής**
Ονοματεπώνυμο Φοιτητή	**Ιωάννης Βενετικίδης**
Πατρώνυμο	**Αβραάμ**
Αριθμός Μητρώου	**ΠΜΣΠΣΠ/ 08002**
Επιβλέπων	**Μιχάλης Ψαράκης, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Μήνας Έτος**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Μιχάλης Ψαράκης, Επίκουρος Καθηγητής	Δημήτριος Γκιζόπουλος, Αναπληρωτής Καθηγητής	Παναγιώτης Κοτζανικολάου, Λέκτορας

Περιεχόμενα

Περίληψη

Η τεχνολογία των συσκευών FPGA αποτελεί μια πολύ σπουδαία εξέλιξη στο τομέα των ενσωματωμένων υπολογιστικών συστημάτων. Σε αυτή τη μεταπτυχιακή διατριβή ασχολούμαστε με αυτή την τεχνολογία και προσπαθούμε να δούμε τις εξελίξεις, όπως αυτές φαίνονται μέσα από κάποια παραδείγματα. Αυτά αποτελούν συσκευές FPGA διαφορετικών εταιριών και συγκεκριμένα των Altera, Xilinx και Atmel. Εστιάζουμε σε οικογένειες συσκευών όπως Cyclone, Spartan-3 και AT40K και βλέπουμε μέσα από τις αρχιτεκτονικές τους δομές, τις διαφορές αλλά και τις ομοιότητες τους. Παράλληλα χρησιμοποιώντας τα αντίστοιχα εργαλεία σχεδίασης των εταιριών αυτών, Quartus II, ISE και IDS – Figaro, σχεδιάζουμε και συγκρίνουμε τις τεχνολογικές δομές και υποδομές. Ο έλεγχος και η σχεδίαση γίνεται μέσα από έναν αριθμό μετροπρογραμματων, συγκεκριμένα 29, τα οποία “τρέχουν” σε κάθε εργαλείο σχεδίασης. Τα αποτελέσματα των προσομοιώσεων δεν μπορούμε να πούμε ότι μας εξέπληξαν αλλά ούτε μας άφησαν αδιάφορους. Μέσα από την έρευνα - μελέτη αυτής της μεταπτυχιακής διατριβής, πιστεύουμε ότι βγήκαν χρήσιμα συμπεράσματα τα οποία κάλυψαν μεν κάποια ερωτηματικά, αλλά “γέννησαν” δε νέα πιο ενδιαφέροντα.

Summary

The technology of FGPA devices is a very important development in the field of embedded computing systems. In this graduate thesis we are dealing with this technology and trying to see the developments, as seen through some examples. These devices are different FPGA companies namely Altera, Xilinx and Atmel. We focus on families of devices such as Cyclone, Spartan-3 and AT40K and see, through their architectural structures, their differences and similarities. At the same time, using the corresponding tools of these companies, Quartus II, ISE and IDS - Figaro, we are designing and comparing the technological structures and infrastructures. The control and design is done through a number of benchmarks, namely 29, which "run" in each design tool. We cannot say that the results of the simulations surprised us neither left us indifferent. Through research of this postgraduate thesis, we believe that useful conclusions emerged that on the one hand covered some questions, but on the other, some new, more important were born.

1. Εισαγωγή

Η ραγδαία ζήτηση αλλά και η ανάπτυξη των πληροφορικών και πληροφοριακών συστημάτων τη δεκαετία του 90 οδήγησε αναπόφευκτα στην εξέλιξη της τεχνολογίας των συστημάτων αυτών. Παράλληλα με αυτά υπήρξε και μια τεράστια ζήτηση για αποθήκευση αλλά και διαχείριση τεράστιων ποσοτήτων πληροφοριών. Το internet, αλλά και η δυνατότητα δημιουργίας δικτύων ικανών να διακινούν το τεράστιο μέγεθος των πληροφοριών αυτών, έθεσαν τις βάσεις στην εξέλιξη συστημάτων μικρότερου μεγέθους τα επονομαζόμενα *ενσωματωμένα πληροφορικά συστήματα* (*embedded systems*). Τα συστήματα αυτά δημιουργήθηκαν ή εξελίχθηκαν για την εξυπηρέτηση και τη διευκόλυνση του κοινού. Παραδείγματα τέτοιων μικρών συστημάτων είναι πχ ένας προσωπικός μίνι ηλεκτρονικός υπολογιστής (*Notebook*), ένα *σύστημα πλοήγησης* (*GPS, Global Positioning System)*, ένα *έξυπνο κινητό τηλέφωνο* (*smart phone*) ή ένας *ψηφιακός προσωπικός βοηθός* το γνωστό μας *PDA* (*Personal Digital Assistant*). Επιπλέον παραδείγματα τέτοιων μικρών συστημάτων είναι ένα υπολογιστικό σύστημα το οποίο μπορεί να βρίσκεται σε ένα αυτοκίνητο, σε ένα ψυγείο, σε μια αντλία καυσίμου κτλ. Είναι κοινή πεποίθηση όλων λοιπόν ότι διανύουμε την *Επανάσταση της Πληροφορίας*. Μέσα σε όλη αυτή την εξέλιξη ο τομέας των ενσωματωμένων πληροφορικών συστημάτων δεν μπορεί να μη μένει ανεπηρέαστος. Η παγκόσμια βιομηχανία των υπολογιστών αλλά και των εταιριών λογισμικού έχουν ρίξει το βάρος της στην ανάπτυξη των συστημάτων αυτών. Δεν είναι τυχαίο ότι η παγκόσμια αγορά των ενσωματωμένων συστημάτων εκτινάχτηκε το 2002 στο τεράστιο αριθμό των 1.122.000.000.000 υπολογιστών έναντι των 131.000.000 επιτραπέζιων υπολογιστών, όπως αναφέρουν στο βιβλίο τους οι Patterson και Hennessy [P&H]. Αυτός ο αριθμός πιστεύετε ότι είναι αστρονομικός τη χρονιά που διανύουμε. Παράλληλα μια ακόμη σημαντική εξέλιξη αλλά και ανάπτυξη παρατηρούμε και στο τομέα των προγραμμάτων και των λειτουργικών συστημάτων για ενσωματωμένα συστήματα. Όλο και περισσότερες εταιρίες λογισμικού και αυτόνομοι προγραμματιστές προσπαθούν να δημιουργούν *προγράμματα – εφαρμογές* (*applications*) για ενσωματωμένες συσκευές. Ένα πολύ πρόσφατο παράδειγμα στις μέρες μας είναι οι εφαρμογές που έχουν δημιουργηθεί για τη φημισμένη συσκευή της Apple το iPod.

Ένας πολύ μεγάλος αλλά και γρήγορα αναπτυσσόμενος κλάδος της βιομηχανίας των ενσωματωμένων συστημάτων είναι η τεχνολογία των συσκευών *FPGA* (*Field Programmable Gate Array*), σε ελεύθερη μετάφραση θα μπορούσαμε να πούμε ότι είναι οι *Επιτόπου Προγραμματιζόμενες Διατάξεις Πυλών*. Η παρούσα μεταπτυχιακή διατριβή ασχολείται με την τεχνολογία των συσκευών FPGA και θα προσπαθήσει να αναπτύξει και να παρουσιάσει αρχιτεκτονικές που έχουν αναπτυχθεί κατά καιρούς από διάφορες εταιρίες. Θα δούμε επίσης τα αντίστοιχα λογισμικά *σχεδίασης μέσω υπολογιστή* (*CAD,Computer Aided Design*) που χρησιμοποιούνται για το προγραμματισμό των συσκευών αυτών. Αυτές οι συσκευές έχουν ενσωματωμένους επεξεργαστές οι οποίοι σχεδιάζονται (προγραμματίζονται για την ακρίβεια) με χρήση *πυρήνων επεξεργαστών* (*processor cores*). Η σχεδίαση αυτή μπορεί να γίνει προγραμματίζοντας τις συσκευές αυτές με τη χρήση *γλώσσάς περιγραφής υλικού* (*HDL, Hardware Description Language*). Κατά καιρούς έχουν αναπτυχθεί διάφορες γλώσσες HDL από διαφορετικές εταιρίες προς εξυπηρέτηση των συσκευών αυτών, στο Πίνακα Α.1 στο παράρτημα Α βλέπουμε κάποιες από αυτές. Από τις πιο διαδεδομένες γλώσσες είναι η *Verilog* και *VHDL (Very Hardware Description Language*), εμείς από τη μεριά μας σε όλα τα παραδείγματα που θα ακολουθήσουν θα χρησιμοποιήσουμε κώδικα VHDL.

1.1. Σκοπός της μεταπτυχιακής διατριβής

Η παρούσα μεταπτυχιακή διατριβή αποτελείται από δυο μέρη, το θεωρητικό και το πειραματικό. Στο θεωρητικό κομμάτι θα μελετήσουμε και θα παρουσιάσουμε τις αρχιτεκτονικές δομές των συσκευών FPGA καθώς και τις μεταξύ τους διαφορές. Συγκεκριμένα θα ασχοληθούμε με τρεις διαφορετικές αρχιτεκτονικές δομές που βρίσκονται αυτή τη στιγμή στην αγορά και ειδικότερα με τις οικογένειες συσκευών *Spartan-3* [Xln1], *Cyclone* [Alt1] και *AT40K* [Atm1] των εταιριών *Xilinx* [Xln5], *Altera* [Alt4] και *Atmel* [Atm4] αντίστοιχα. Επίσης θα γίνει μια αναφορά στα εργαλεία CAD που θα χρησιμοποιηθούν για το προγραμματισμό της κάθε συσκευής, αυτά είναι : το *Altera Quartus II* το *Xilinx ISE* και το *Atmel IDS Figaro*. Στο πειραματικό κομμάτι θα παρουσιάσουμε τα αποτελέσματα και τις επιδόσεις των συσκευών αυτών έναντι των πρότυπων προγραμμάτων VHDL που θα χρησιμοποιηθούν.

Σκοπεύουμε να παρουσιάσουμε χρήσιμα συμπεράσματα από τα οποία να μπορέσει κανείς να αναδείξει τις διαφορές αλλά και τις ομοιότητες που μπορεί να υπάρχουν ανάμεσα σε διαφορετικές αρχιτεκτονικές συσκευών, οικογενειών και εταιριών. Επίσης πεποίθησή μας είναι ότι θα μπορέσουμε να δώσουμε μια καθαρή εικόνα για το πώς λειτουργούν αυτές οι συσκευές και ποιο σκοπό αποσκοπεί η παρουσία τους. Από το πειραματικό μέρος θα θέλαμε να επισημάνουμε τη διαφορά που μπορεί να

υπάρχει στη σχεδίαση από συσκευή σε συσκευή καθώς και τη διάφορα στο τρόπο λειτουργίας τους. Θα θέλαμε να σημειώσουμε επίσης ότι το τελευταίο στάδιο πριν την πραγματική δοκιμή των συσκευών, δηλαδή πριν το προγραμματισμό των συσκευών απευθείας στο υλικό, δεν υφίσταται σε αυτή τη διατριβή. Οι δόκιμες και οι έλεγχοι που γίνονται, ξεχωριστά για κάθε μια συσκευή, υλοποιούνται αποκλειστικά σε προσομοιωτή, αυτός ο προσομοιωτής αποτελεί μέρος των εργαλείων CAD.

1.2. Προαπαιτούμενα για την παρακολούθηση της μεταπτυχιακής διατριβής

Για να μπορέσει κάποιος να παρακολουθήσει την παρούσα διατριβή θα πρέπει να έχει παρακολουθήσει τα βασικά μαθήματα της κατεύθυνσης του υλικού (hardware) του προγράμματος σπουδών ενός τμήματος Πληροφορικής, όπως *Ψηφιακή Σχεδίαση* (*Digital Design*) και *Αρχιτεκτονική Υπολογιστών* (*Computer Architecture*).
Να είναι εξοικειωμένος με έννοιες όπως :

- λογικές πύλες (*Logic Gate*)
- αποκωδικοποιητής (*Decoder*)
- κωδικοποιητής (*Coder*)
- πολυπλέκτης (*MUX, Multiplexer*)
- μνήμες *RAM, ROM, SRAM, DRAM*
- το βασικό κύτταρο μνήμης, *Flip Flop*
- δίαυλος δεδομένων (*Data Bus*), δίαυλος διευθύνσεως (*Address Bus*)
- καθώς και τη *διάταξη τριών καταστάσεων* (*Three State Buffer*)

Επίσης θα πρέπει να έχει γνώση περί διασύνδεσης, τουλάχιστον τα στοιχειώδη, *λογικών κυκλωμάτων* (*Logic Circuit*) και *ολοκληρωμένων κυκλωμάτων, ΟΚ* (*IC,Integrate Circuit*) ή αλλιώς *τσιπ* (*Chip*).

Όλα τα παραπάνω αποτελούν βασική γνώση για την παρακολούθηση του θεωρητικού μέρους της διατριβής. Για την παρακολούθηση του πειραματικού μέρους θα λέγαμε ότι δεν χρειάζεται κάποιος να έχει γνώση της γλωσσάς VHDL. Όλες οι δοκιμές θα γίνονται αποκλειστικά μέσα από τα προγράμματα, κάθε κομμάτι κώδικα που θα πηγαίνει για υλοποίηση θα φορτώνεται αυτόματα και θα εκτελείται. Σκοπός των πειραμάτων είναι να δούμε πως κάθε συσκευή διευθετεί την κάθε σχεδίαση και ποιες είναι οι ανοχές της. Σίγουρα αν κάποιος έχει γνώση της γλώσσας VHDL θα έχει μια καλύτερη εικόνα ως προς το τι διαδραματίζεται μέσα στις συσκευές. Σε αυτό το σημείο εμείς λέμε και πάλι ότι δεν είναι προϋπόθεση ότι κάποιος πρέπει να ξέρει τη VHDL για να παρακολουθήσει το δεύτερο μέρος. Όλες οι αναφορές που θα δημιουργηθούν θα παρουσιάζονται σε πίνακες και διαγράμματα έτσι ώστε να είναι καλύτερη παρακολούθησή τους.

2. Προγραμματιζόμενες Λογικές Συσκευές

Πριν προχωρήσουμε στην παρουσίαση των FPGAs θα κάνουμε μια μικρή ιστορική ανάδρομη στο τι προϋπήρχε αλλά και πώς εξελίχθηκε η τεχνολογία των ολοκληρωμένων κυκλωμάτων και των προγραμματιζόμενων λογικών συσκευών. Παρακάτω αναλύουμε τους πιο διαδεδομένους τύπους ολοκληρωμένων κυκλωμάτων που υπήρξαν από το ξεκίνημα της τεχνολογίας αυτής μέχρι και τις σημερινές εξελίξεις. Στην Ενότητα 2.1 θα δούμε τα *τυπικά ολοκληρωμένα κυκλώματα* (*Standard Chips*) και πώς τα ολοκληρωμένα κυκλώματα κατατάσσονται με βάση την κλίμακα ολοκλήρωσης που έχουν. Στην Ενότητα 2.2 θα δούμε τις *διατάξεις προγραμματιζόμενης λογικής* (*PLDs, Programmable Logic Devices*), πώς αυτές δομούνται, καθώς και τα ειδή των διατάξεων αυτών. Τέλος στην Ενότητα 2.3 θα δούμε τα *ειδικά ολοκληρωμένα κυκλώματα* (*ASIC, Application Specific Integrated Circuit*) και συγκεκριμένα ποιος ο ρόλος τους και οποια η διαφορά τους με τα άλλα ΟΚ (από εδώ και στο εξής θα αναφέρουμε τα ολοκληρωμένα κυκλώματα με την ένδειξη ΟΚ, επιπλέον την ίδια φιλοσοφία θα ακολουθούμε και με άλλα ακρώνυμα μέσα στη διατριβή).

Στην αγορά υπάρχει μια πολύ μεγάλη ποικιλία ΟΚ τα οποία εκτελούν διάφορες λειτουργίες από τις πιο απλές μέχρι τις πιο πολύπλοκες. Έτσι υπάρχουν ΟΚ πολύ απλής λειτουργικότητας έως εξαιρετικά περίπλοκης. Από αυτά μπορούμε να συμπεράνουμε ότι κάποια από τα ΟΚ λόγω της ιδιαιτερότητας τους και λόγω διαφορετικών απαιτήσεων χρειάζονται να σχεδιαστούν από την αρχή. Σε αυτά τα προβλήματα δίνουν λύση τρεις μορφές ΟΚ: τα τυπικά ΟΚ, οι διατάξεις προγραμματιζόμενης λογικής και τα ειδικά ολοκληρωμένα κυκλώματα.

2.1. Τυπικά Ολοκληρωμένα Κυκλώματα

Τα λογικά κυκλώματα υλοποιούνται ηλεκτρονικά με τη χρήση τρανζίστορ που υπάρχουν στα ΟΚ. Στις πρώτες μορφές τα ΟΚ είχαν μερικές δεκάδες τρανζίστορ, στις μέρες μας έχουν φτάσει σε δεκάδες - εκατοντάδες εκατομμύρια τρανζίστορ με εμβαδό μικρότερο από 100 mm^2.

Τα ΟΚ ανάλογα με την πολυπλοκότητα που έχουν κατατάσσονται σε κατηγορίες, αυτές οι κατηγορίες ονομάζονται κλίμακες ολοκλήρωσης. Στην αγορά υπάρχουν :

- *Μικρής κλίμακας ολοκλήρωσης* (*SSI, Small Scale Integration*) που περιέχουν μέχρι 10 λογικές πύλες ανά ΟΚ. Πολύ γνωστή σειρά αυτών των ΟΚ είναι η 7400 της εταιρίας *Texas Instruments* [TI].
- *Μέσης κλίμακας ολοκλήρωσης* (*MSI, Medium Scale Integration*) που περιέχουν μεταξύ 10 έως και 200 λογικές πύλες ανά ΟΚ. Συνήθως αυτά τα ΟΚ υλοποιούν πολύπλοκες λογικές συναρτήσεις όπως αριθμητικές μονάδες (αθροιστές αφαιρέτες), κωδικοποιητές αποκωδικοποιητές κτλ.
- *Μεγάλης κλίμακας ολοκλήρωσης* (*LSI, Large Scale Integration*) που περιέχουν μεταξύ 200 έως και 200.000 λογικές πύλες ανά συσκευή. Αυτές οι συσκευές υλοποιούν περισσότερο πολύπλοκες λογικές συναρτήσεις από τα προηγούμενα όπως πχ μια μικρή λογική μονάδα, μια μονάδα ελέγχου, κύκλωμα μνημών κτλ.
- *Πολύ Μεγάλης κλίμακας ολοκλήρωσης* (V*LSI, Very Large Scale Integration*) που περιέχουν περισσότερες από 200.000 λογικές πύλες ανά συσκευή. Αυτές οι συσκευές υλοποιούν συνήθως ψηφιακά συστήματα πχ αυτόνομες μονάδες, μεγάλες μονάδες ελέγχου κτλ.
- *Υπερβολικά μεγάλης κλίμακας ολοκλήρωσης* (U*LSI, Ultra Large Scale Integration*) που περιέχουν εκατομμύρια λογικές πύλες ανά συσκευή. Αυτές οι συσκευές υλοποιούν συνήθως πλήρη ψηφιακά συστήματα πχ υπολογιστές SoC (System on Chip), δηλαδή ένα ΟΚ το οποίο να υλοποιεί ένα ολόκληρο υπολογιστικό σύστημα κτλ.

Τα μικρής κλίμακας ολοκλήρωσης (SSI) και τα μέσης κλίμακας ολοκλήρωσης (MSI) ΟΚ θα μπορούσαμε να τα αναφέρουμε και ως τυπικά ΟΚ. Τα υπόλοιπα αποτελούν πολύπλοκες συσκευές με ιδιαίτερα χαρακτηριστικά και δυνατότητες, τα οποία θα δούμε στη συνέχεια.

Κάθε ΟΚ πρέπει να ακολουθεί κάποια χαρακτηριστικά τα οποία να το κάνουν μοναδικό, να βρίσκονται δηλαδή σε συμφωνία με κάποια θεσπισμένα πρότυπα. Η παραγωγή διαφόρων τύπων ΟΚ με αυτά τα χαρακτηριστικά, αποτελούν μια οικογένεια (*family*) ΟΚ της εκάστοτε εταιρίας κατασκευής. Τα κύρια χαρακτηριστικά ενός ΟΚ είναι :

- *Ταχύτητα* (*speed*) ή μπορούμε να πούμε η μέγιστη συχνότητα λειτουργίας. Καθορίζεται από την *καθυστέρηση διάδοσης* (*Propagation Delay*) του σήματος μέσα από το ΟΚ, δηλαδή το χρονικό

διάστημα από την αλλαγή της κατάστασης της εισόδου μέχρι την αλλαγή της κατάστασης της εξόδου του ΟΚ. Να σημειωθεί εδώ ότι αυτή η καθυστέρηση παίζει πολύ σημαντικό ρόλο στην αξιολόγηση ενός ΟΚ.

- *Fan In – Fan Out.* Ονομάζονται το μεν πρώτο το πλήθος των εισόδων που διαθέτει ένα ΟΚ (ή μια πύλη) και το δεύτερο το πλήθος των εισόδων άλλων ΟΚ (ή πυλών), της ίδια οικογενείας, που μπορεί να τροφοδοτήσει άμεσα η έξοδος αυτού του ΟΚ (ή της πύλης).
- *Ανοχή σε Θόρυβο* (*Noise Immunity*). Είναι ο βαθμός αλλοίωσης που πρέπει να βρεθεί το ηλεκτρικό σήμα ώστε να δώσει λανθασμένο αποτέλεσμα (κατάσταση) στην έξοδο του ΟΚ (ή της πύλης).
- *Ρυθμός βλαβών* (*Failure Rate*) του κυκλώματος. Καθορίζει την αξιοπιστία του κυκλώματος και τη συχνότητα βλαβών.
- *Τάση τροφοδοσίας (Supply Voltage*) και *κατανάλωση ισχύος* (*Power Dissipation*). Αποτελούν δυο παράγοντες πολύ σημαντικούς για την αξιολόγηση ενός ΟΚ, ειδικά στις μέρες μας. Τα ενσωματωμένα συστήματα πολλές φορές αποτελούν αυτόνομα συστήματα, δηλαδή έχουν μπαταρίες για την τροφοδοσία τους, σκεφτείτε λοιπόν ένα τέτοιο ΟΚ να έχει μεγάλη κατανάλωση ισχύος και τάση τροφοδοσίας. Αυτό το ΟΚ θα αποτελέσει την καταστροφή του όποιου συστήματος στο οποίο είναι ενσωματωμένο. Έτσι όταν ένα ΟΚ έχει μηδαμινές καταναλώσεις, το σύστημα θα έχει μακροβιότερη λειτουργία. Ένας ακόμη παράγοντας μεγάλης κατανάλωσης είναι και η ταχύτητα ρολογιού, όσο μεγαλύτερη είναι τόσο μεγάλη κατανάλωση ισχύος έχουμε.
- Το *γινόμενο ταχύτητας ισχύος* (*Speed Power Product*), δηλαδή της καθυστέρησης διάδοσης επί την κατανάλωση ισχύος σε ένα ΟΚ (ή μια πύλη).
- Το κόστος των ΟΚ

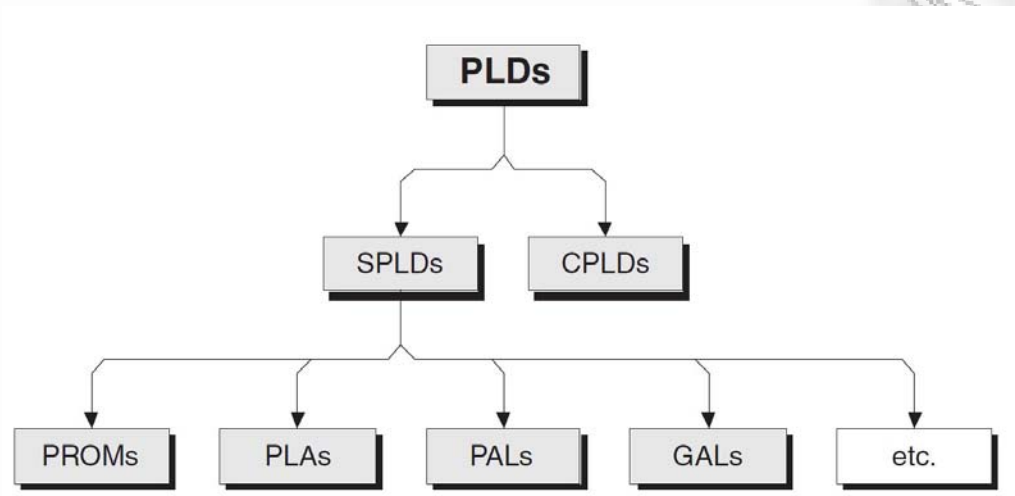
Τα τυπικά ΟΚ ήταν δημοφιλή για τη δημιουργία λογικών κυκλωμάτων έως της αρχές της δεκαετίας 1980. Όμως καθώς εξελισσόταν η τεχνολογία κατασκευής ΟΚ, κατέστη ανώφελη η διάθεση πολύτιμου χορού στις *πλακέτες τυπωμένου κυκλώματος* (*PCB,Printed Circuit Board*), για την τοποθέτηση ΟΚ με περιορισμένες δυνατότητες. Έτσι στράφηκαν σε λύσης οι οποίες να ωφελούν τις εξελίξεις, δηλαδή να υπάρχουν πολύ περισσότερα τρανζίστορ σε μια μικρή cm^2 επιφάνεια. Αυτό αυτόματα σημαίνει ότι πολύ περισσότερες λειτουργίες μπορούν να υλοποιούνται - εκτελούνται σε μια πλακέτας PCB. Επίσης ένα άλλο μειονέκτημα των τυπικών ΟΚ είναι η λειτουργία του καθενός από αυτά είναι συγκεκριμένη και δε μπορεί να μεταβληθεί κατά τη διάρκεια του χρόνου. Για να γίνει κάτι τέτοιο πρέπει να ξανασχεδιαστεί και αμέσως μετά να κατασκευαστεί, διαδικασία που είναι αρκετά χρονοβόρα.

2.2. Διατάξεις Προγραμματιζόμενης Λογικής PLDs

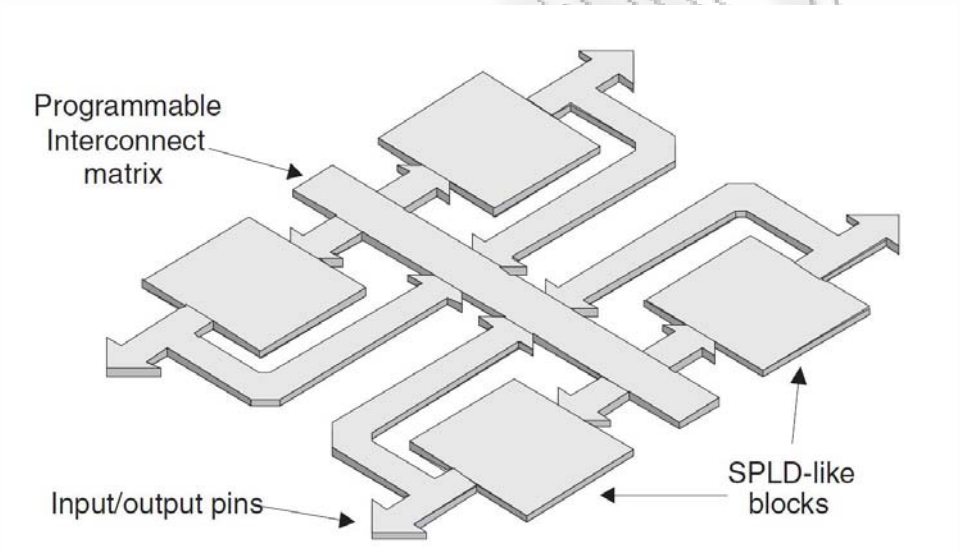
Σε αντίθεση με τα τυπικά ΟΚ που εκτελούν συγκεκριμένες λειτουργίες, υπάρχουν ΟΚ που μπορούν να σχεδιαστούν και να οργανωθούν από το χρήστη, έτσι ώστε να υλοποιούν συγκεκριμένα λογικά κυκλώματα. Αυτό πετυχαίνεται από ειδικά κυκλώματα γενικού σκοπού τα οποία έχουν μια γενική αρχιτεκτονική δομή και περιλαμβάνουν ένα σύνολο από *προγραμματιζόμενους λογικούς διακόπτες* (*PLS, Programmable Logic Switch*), αυτά είναι οι συσκευές LSI,VLSI και ULSI που είδαμε και παραπάνω. Οι διακόπτες αυτοί επιτρέπουν τα εσωτερικά κυκλώματα του ΟΚ να οργανώνονται με διάφορους τρόπους. Ο σχεδιαστής μπορεί να οργανώσει και να σχεδιάσει το λογικό κύκλωμα που θέλει επιλέγοντας την κατάλληλη οργάνωση των διακοπτών αυτών την ώρα της "κατασκευής" του ΟΚ. Οι διακόπτες αυτοί προγραμματίζονται από το τελικό χρήστη και όχι από την κατασκευάστρια εταιρία. Δηλαδή ο χρήστης αφού καταλήξει και σχεδιάσει το κύκλωμα που θέλει στο εργαλείο CAD, το εργαλείο προγραμματίζει τη συσκευή ενεργοποιώντας ή απενεργοποιώντας τους διακόπτες αυτούς. Ο προγραμματισμός μπορεί να γίνει όσες φορές χρειάζεται στις περισσότερες από τις συσκευές αυτές. Αν χρειαστεί για παράδειγμα ένα ΟΚ να εμπλουτιστεί με μια νέα λειτουργία, ή να επαλειφθούν κάποιες παλιές λειτουργίες από αυτό, γίνεται πολύ εύκολα και όσες φορές απαιτείται. Αυτό είναι ίσως το μεγάλο πλεονέκτημα των συσκευών αυτών που ονομάζονται *διατάξεις προγραμματιζόμενης λογικής* (*PLDs, Programmable Logic Devices*). Υπάρχουν πολλά PLDs στην αγορά σε διαφορετικά μεγέθη για να καλύπτουν όλες τις ανάγκες για ΟΚ. Το πλεονέκτημα τους στο να επαναπρογραμματίζονται έναντι των τυπικών ΟΚ, τα έχει κάνει πολύ δημοφιλή και ιδιαίτερα χρησιμοποιημένα στις μέρες μας.

Τα PLDs χωρίζονται σε δυο κατηγορίες τα *απλά PLDs* (*SPLDs, Simple Programmable Logic Devices*) και τα *πολύπλοκα* PLDs (*CPLDs, Complexity Programmable Logic Devices*), Εικόνα 2.1. Πολλοί δεν

καταλαβαίνουν τη διάφορα αλλά η αιτία του διαχωρισμού είναι η πολυπλοκότητα τους. Τα SPLDs είναι αυτόνομα ΟΚ τα οποία προγραμματίζονται το καθένα ξεχωριστά, ενώ τα CPLDs αποτελούνται από πολλά αυτόνομα SPLDs τα οποία προγραμματίζονται όλα μαζί, δείτε την Εικόνα 2.2 για να καταλάβετε τη διαφορά τους. Το CPLD περιέχει περισσότερα από ένα SPLD, τα οποία αποτελούν μέρος της αρχιτεκτονικής του δομής.



Εικόνα 2.1 [Max] Ιεραρχικές ομάδες PLDs



Εικόνα 2.2 [Max] Η γενική δομή ενός CPLD

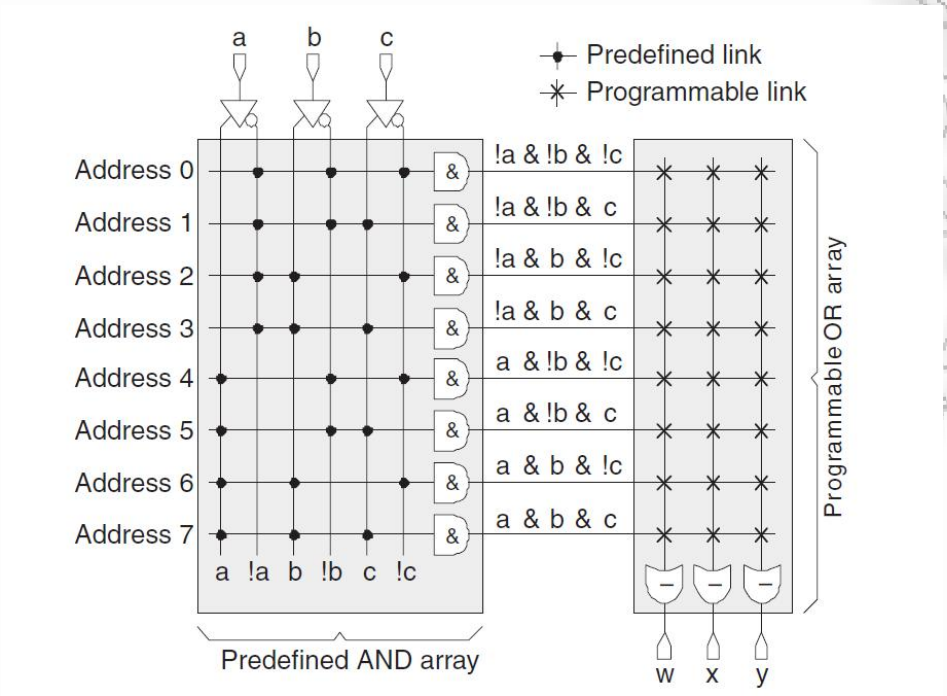
Παρακάτω βλέπουμε τα σημαντικότερα από τα PLDs.

- Μνήμη PROM
- Προγραμματιζόμενη λογική διάταξη (PLA)
- Προγραμματιζόμενη διάταξη λογικής (PAL)
- Πολύπλοκες προγραμματιζόμενες λογικές διάταξης (CPLD)
- Επιτόπου προγραμματιζόμενες διατάξεις πυλών (FPGA).

2.2.1. Μνήμη PROM

Η πρώτη διάταξη SPLD είναι η *προγραμματιζόμενη μνήμη ROM (PROM, Programmable Read Only Memory*) και εμφανίστηκε το 1970. Όπως μπορείτε να δείτε και από την Εικόνα 2.3, η συγκεκριμένη μνήμη PROM αποτελείται από τρεις εισόδους όπου κάθε είσοδος έχει και το συμπλήρωμά της κάνοντας χρήση της πύλης NOT. Υπάρχει επίσης μια διάταξη από προκαθορισμένες από το κατασκευαστή πύλες AND και μια διάταξη από προγραμματιζόμενες πύλες OR. Ο χρήστης μπορεί να προγραμματίσει μόνο

τις πύλες OR και όχι τις πύλες AND, αυτός ο προγραμματισμός (των πυλών OR) γίνεται μόνο μια φορά και δεν μπορεί να επαναληφθεί, δηλαδή αυτή η μνήμη δεν είναι επαναπρογραμματιζόμενη. Όταν λέμε ότι προγραμματίζεται ένας διακόπτης, ουσιαστικά αυτό που γίνεται είναι να "καεί" σε αυτή τη θέση η ασφάλεια του διακόπτη.

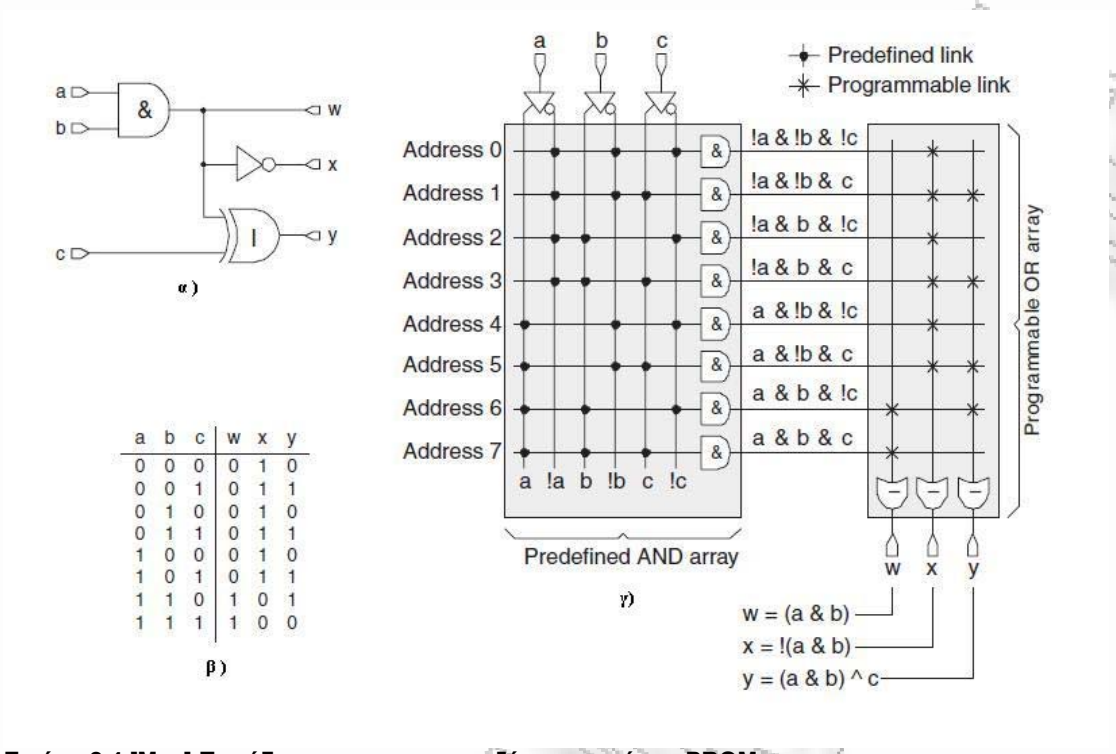


Εικόνα 2.3 [Max] Προγραμματιζόμενη μνήμη PROM

Οι μνήμες PROM παρέχουν ευελιξία και είναι βολικές, το οποίο δεν ισχύει στις κοινές μνήμες ROM. Οι τελευταίες είναι οικονομικά ελκυστικές αλλά το κόστος παρασκευής μια μάσκας, η οποία απαιτείται για την αποθήκευση ενός συγκεκριμένου πληροφοριακού προτύπου σε μια μνήμη ROM, τις καθιστά αρκετά ακριβές όταν μόνο ένα μικρό πλήθος από αυτές θα χρειαστεί να κατασκευαστεί. Αυτό το μειονέκτημα έρχεται να εκμεταλλευτεί μια μνήμη PROM, η οποία παρέχει ένα γρηγορότερο και αρκετά φθηνότερο τρόπο κατασκευής, επειδή ακριβώς μπορούν να προγραμματιστούν απευθείας από το χρήστη.

Οι μνήμες PROM μπορούν να χρησιμοποιηθούν σαν μνήμες γενικού σκοπού αλλά και σαν *πίνακες αναζήτησης* (*Lookup Table*), όπου μπορεί ο χρήστης να υλοποιήσει απλές λογικές λειτουργίες - συναρτήσεις. Υπάρχουν μνήμες πολλών εισόδων πολλών εξόδων και γενικά μπορούμε να υλοποιήσουμε Μ συναρτήσεις των Ν εισόδων.

Ας δούμε για παράδειγμα πως λειτουργεί αυτή η μνήμη. Έχουμε ένα κύκλωμα τριών εισόδων a, b, c όπως αυτό που βλέπουμε στην Εικόνα 2.4 στο α). Το κύκλωμα αυτό αποτελείται από μία πύλη AND μια XOR και μία NOT, επίσης έχει τρείς εξόδους τα w,x,y. Ο πίνακας αληθείας του κυκλώματος φαίνεται στην ίδια εικόνα στο β). Στο γ) βλέπουμε ότι για κάθε έξοδο η μνήμη PROM έχει ενεργοποιήσει τους κατάλληλους διακόπτες ώστε να δώσει τα σωστά αποτελέσματα. Παράδειγμα : για την έξοδο w ενεργοποιήθηκαν οι δυο τελευταίοι διακόπτες από την πρώτη στήλη ξεκινώντας από πάνω. Ενώ για την έξοδο x ενεργοποιήθηκαν όλοι οι διακόπτες της στήλης εκτός από τα δύο τελευταία. Όποιος και να είναι ο συνδυασμός των τριών εισόδων της μνήμης PROM θα ενεργοποιηθεί η κατάλληλη έξοδος. Μπορούμε να καθορίσουμε απευθείας το πίνακα αληθείας των συναρτήσεων που θέλουμε να υλοποιήσουμε στη μνήμη PROM, χωρίς να προβούμε σε καμία απλοποίηση.



Εικόνα 2.4 [Max] Παράδειγμα προγραμματιζόμενης μνήμης PROM

Μεταγενέστερη εξέλιξη των μνημών PROM είναι η *επαναπρογραμματιζόμενη μνήμη PROM* ή *EPROM* (*Erasable Programmable Read Only Memory*). Ο επαναπρογραμματισμός της μνήμης EPROM γίνεται με την απομάκρυνση των φορτίων που βρίσκονται στα τρανζίστορ. Αυτό γίνεται με την απόσπαση του ΟΚ από το φυσικό κύκλωμα και την έκθεση του σε υπεριώδες φως. Αυτό το μειονέκτημα οδήγησε στη δημιουργία επαναπρογραμματιζόμενων μνημών που λειτουργούν ηλεκτρονικά, τα επονομαζόμενα *EEPROM (Electrical Erasable Programmable Read Only Memory)*. Για να απαλειφθούν τα περιεχόμενα τους, οι μνήμες EEPROM δε χρειάζονται να αποσπαστούν από το φυσικό κύκλωμα στο οποίο ανήκουν, γιατί αυτό γίνεται τοπικά. Ένα ακόμη πλεονέκτημα είναι ότι μπορούμε να κάνουμε επιλεκτική διαγραφή των πληροφοριών που θέλουμε και όχι ολόκληρης της μνήμης όπως υφίσταται στη μνήμη EPROM.

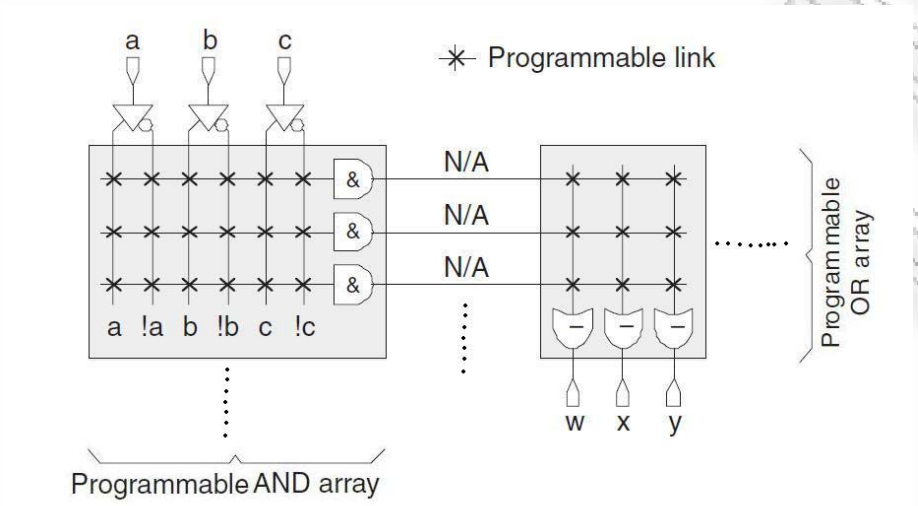
Μια προσέγγιση παρεμφερής με την τεχνολογία μνημών EEPROM, έχει οδηγήσει στην ανάπτυξη συσκευών μνήμης *flash* (*flash memory*). Υπάρχουν ομοιότητες αλλά και διαφορές μεταξύ μια μνήμης EEPROM και μια μνήμης flash. Στη μνήμη EEPROM μπορεί να γίνει ανάγνωση και εγγραφή μίας κυψελίδας (ενός bit). Ενώ σε μια μνήμη flash, καθίσταται μεν δυνατή η ανάγνωση μίας κυψελίδας όπως και σε μια EEPROM, αλλά η εγγραφή μπορεί να γίνει μόνο σε επίπεδο τμήματος του πίνακα των κυψελίδων και όχι καθεμία ξεχωριστά. Πριν την εγγραφή τα προηγούμενα περιεχόμενα του τμήματος απαλείφονται. Οι συσκευές flash έχουν μεγαλύτερη πυκνότητα κάτι που τις οδηγεί σε αυξημένη χωρητικότητα και χαμηλότερο κόστος ανά bit. Απαιτούν μία και μοναδική τάση ρεύματος και καταναλώνουν λιγότερη ενέργεια κατά τη λειτουργία τους. Η χαμηλή κατανάλωση ενέργειας των μνημών flash τις καθιστά ιδιαίτερα ελκυστικές σε φορητές συσκευές οι οποίες ρευματοδοτούνται με μπαταρίες. Κατασκευές της τεχνολογίας flash είναι οι κάρτες flash τα γνωστά μας *USB flash* και οι δίσκοι flash εσωτερικής η εξωτερικής χρήσης με πολύ μεγάλη χωρητικότητα.

Όλες οι παραπάνω τεχνολογίες που είδαμε EPROM, EEPROM και flash αποτελούν επέκταση της τεχνολογίας PROM άρα και επέκταση της ομάδας SPLD.

2.2.2. Προγραμματιζόμενη λογική διάταξη PLA

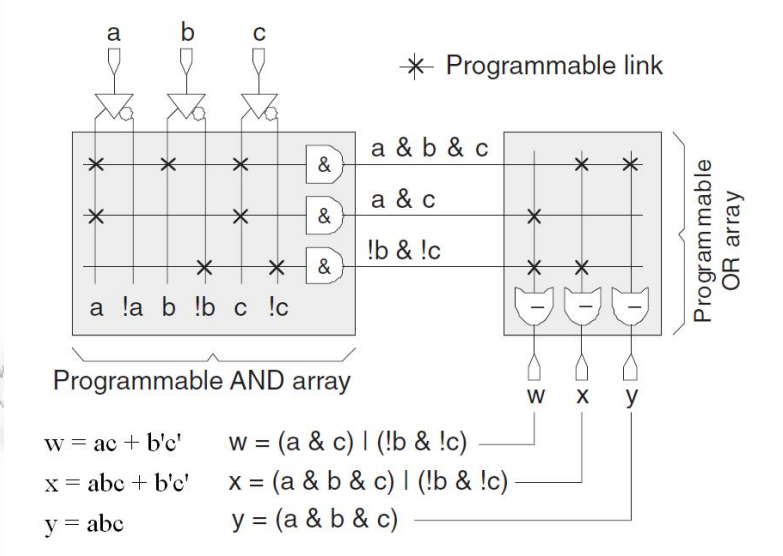
Προγραμματιζόμενη λογική διάταξη (*PLA, Programmable Logic Array*), πρωτοεμφανίστηκε το 1975. Στηριζόμενο στο γεγονός ότι οι λογικές συναρτήσεις μπορούν να υλοποιηθούν με τη μορφή του αθροίσματος γινομένων, το PLA αποτελείται από ένα σύνολο προγραμματιζόμενων πυλών AND που τροφοδοτούν ένα σύνολο προγραμματιζόμενων πυλών OR, δηλαδή και τα δυο σύνολα μπορούν να προγραμματιστούν, Εικόνα 2.5. Ο αριθμός των συναρτήσεων που υλοποιούνται στη συστοιχία των

πυλών AND είναι ανεξάρτητος από τον αριθμό των εισόδων της διάταξης. Το ίδιο συμβαίνει και με τις συναρτήσεις που υλοποιούνται στη συστοιχία των πυλών OR, δηλαδή είναι ανεξάρτητες από τον αριθμό των συναρτήσεων που υλοποιούνται στη συστοιχία των πυλών AND και των εισόδων της διάταξης. Η συστοιχία πυλών AND παράγει ένα αριθμό όρων γινομένου έστω P_1, P_2,….P_k, τα γινόμενα αυτά ενεργούν ως είσοδοι ενός επιπέδου OR, το οποίο παράγει τις εξόδους. Η κάθε έξοδος μπορεί να οργανωθεί έτσι ώστε να αποτελεί άθροισμα οποιονδήποτε όρων του συνόλου P_1, P_2,….P_k και έτσι προκύπτει η λειτουργία του αθροίσματος γινομένων που επιτελεί η διάταξη PLA. Από τα παραπάνω προκύπτει ότι το μέγεθος ενός PLA εξαρτάται από τη μορφή του αθροίσματος γινομένων της εισόδου της διάταξης.



Εικόνα 2.5 [Max] Προγραμματιζόμενη λογική διάταξη PLA

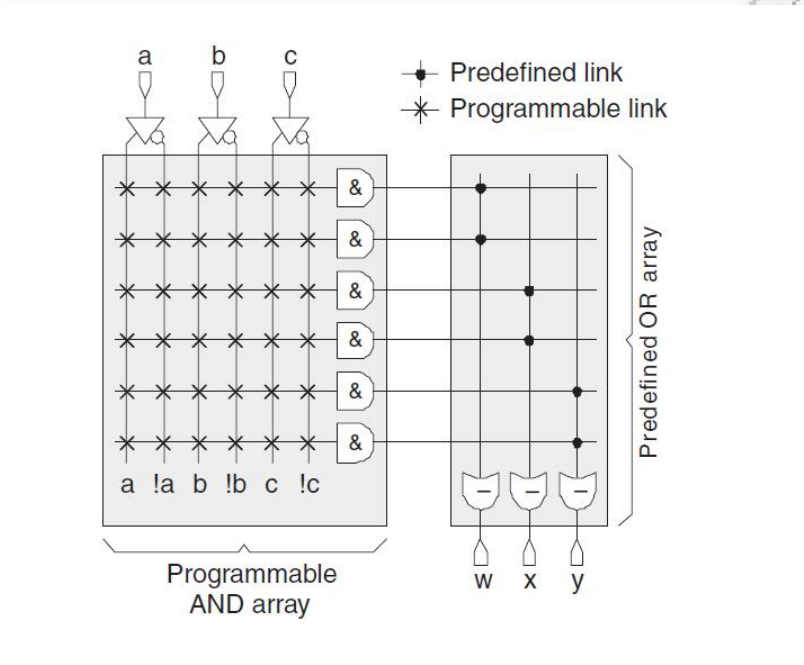
Ας δούμε ένα παράδειγμα προγραμματισμού μιας διάταξης PLA, έστω ότι έχουμε τα αθροίσματα γινομένων της Εικόνας 2.6. Όπως μπορούμε να διαπιστώσουμε από τη διαμόρφωση που έχει γίνει, η συστοιχία των πυλών AND έχει διαμορφώσει τους κατάλληλους διακόπτες ώστε να υλοποιούν τα γινόμενα *abc*, *ac* και *b'c'*. Κατόπιν τα εισάγει στη συστοιχία των πυλών OR και εκεί γίνεται πάλι διαμόρφωση σε κατάλληλους διακόπτες ώστε υλοποιούν τα αθροίσματα των συναρτήσεων. Όπως μπορούμε να δούμε και από την έξοδο w έχουν ενεργοποιηθεί οι δυο τελευταίοι διακόπτες από την πρώτη στήλη της συστοιχίας των πυλών OR, όπου υλοποιούν τη συνάρτηση $w = ac + b'c'$. Κατά παρόμοιο τρόπο έχουν διαμορφωθεί και οι άλλες δυο έξοδοι.



Εικόνα 2.6 [Max] Παράδειγμα προγραμματιζόμενης λογικής διάταξης PLA

2.2.3. Προγραμματιζόμενη διάταξη λογικής PAL

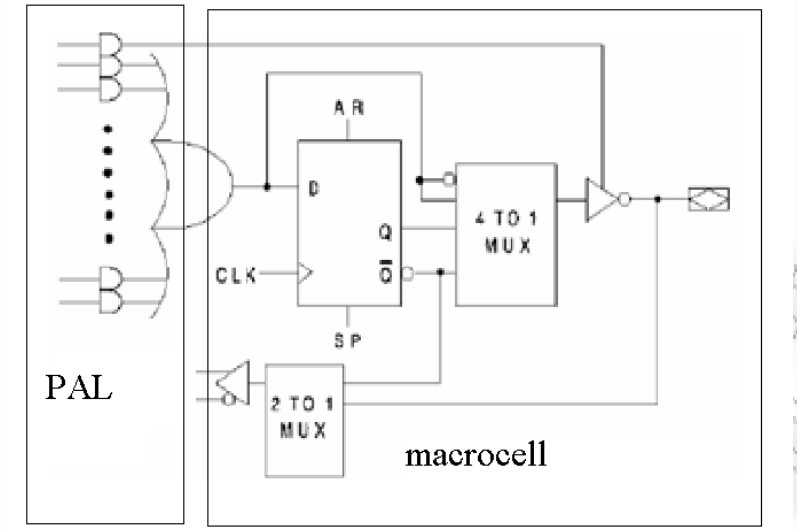
Μια άλλη διάταξη SPLD είναι η *Προγραμματιζόμενη διάταξη λογικής* PAL, η οποία πρωτοεμφανίστηκε και αυτή το 1970 όπως και οι μνήμες PROM. Οι διατάξεις PAL ουσιαστικά επιτελούν την αντίθετη λειτουργία από τις μνήμες PROM. Όπως οι μνήμες PROM έτσι και οι διατάξεις PAL είναι δύο επιπέδων με συστοιχίες πυλών AND και OR, στις μεν μνήμες PROM ο χρήστης μπορεί να προγραμματίσει τη συστοιχία πυλών OR, στις διατάξεις PAL δε ο χρήστης μπορεί να προγραμματίσει τη συστοιχία των πυλών AND και να μείνει ανεπηρέαστη η συστοιχία των πυλών OR η οποία είναι προκαθορισμένη από το κατασκευαστή, Εικόνα 2.7. Μπορούμε να υλοποιήσουμε συναρτήσεις χρησιμοποιώντας τους διαθέσιμους *ελαχιστόρους* (*minterms*). Οι διατάξεις GAL (Generic Array Logic) είναι ένας άλλος τύπος διάταξης PAL.



Εικόνα 2.7 [Max] Προγραμματιζόμενη διάταξη λογικής PAL

Οι διατάξεις PLA είναι προγραμματιζόμενες και στα δυο πεδία συστοιχιών των πυλών AND και OR, αυτό είχε σαν αποτέλεσμα να παρουσιαστούν δυο μειονεκτήματα, το πρώτο έχει να κάνει με τη δυσκολία κατασκευής τους με ακρίβεια και το δεύτερο προκαλούσε μείωση της συνολικής ταχύτητας του κυκλώματος στο οποίο συνδέονταν. Αυτά τα μειονεκτήματα οδήγησαν στην εξέλιξη των διατάξεων PAL και τα έκαναν ιδιαίτερα δημοφιλή τόσο για την ακρίβεια κατασκευής όσο και για τη γρήγορη ταχύτητα τους (αυτό συνεπάγεται και καλύτερη απόδοση). Επιπλέον είναι πολύ οικονομικότερα από τις διατάξεις PLA και γι' αυτό χρησιμοποιούνται ευρέως σε πρακτικές εφαρμογές. Μια πολύ σημαντική αίτια της ταχύτητας των PAL έναντι των PLA, είναι ότι στις πρώτες προγραμματίζεται μόνο μια συστοιχία πυλών, ενώ αντίθετα στις δεύτερες πρέπει να προγραμματιστούν δυο συστοιχίες πυλών.

Σε πολλά κυκλώματα PAL τοποθετούνται επιπρόσθετα κυκλώματα στις εξόδους των πυλών OR ώστε να παρέχεται επιπλέον ευελιξία. Συνηθίζεται να χρησιμοποιούμε τον όρο *μακροκυψέλη* (*macrocell*) για να αναφερόμαστε σε πύλες OR με επιπλέον κυκλώματα, Εικόνα 2.8. Αυτό ουσιαστικά είναι και ένα σημείο αναφοράς για την εξέλιξη των macrocell σε *λογικά στοιχεία* (*LE, Logic Element*) τα οποία χρησιμοποιούνται και αποτελούν το κύριο λογικό στοιχείο μιας αρχιτεκτονικής FPGA, για περισσότερες λεπτομερές για τα λογικά στοιχεία (LE) θα σας παραπέμψουμε στο Κεφάλαιο 3.

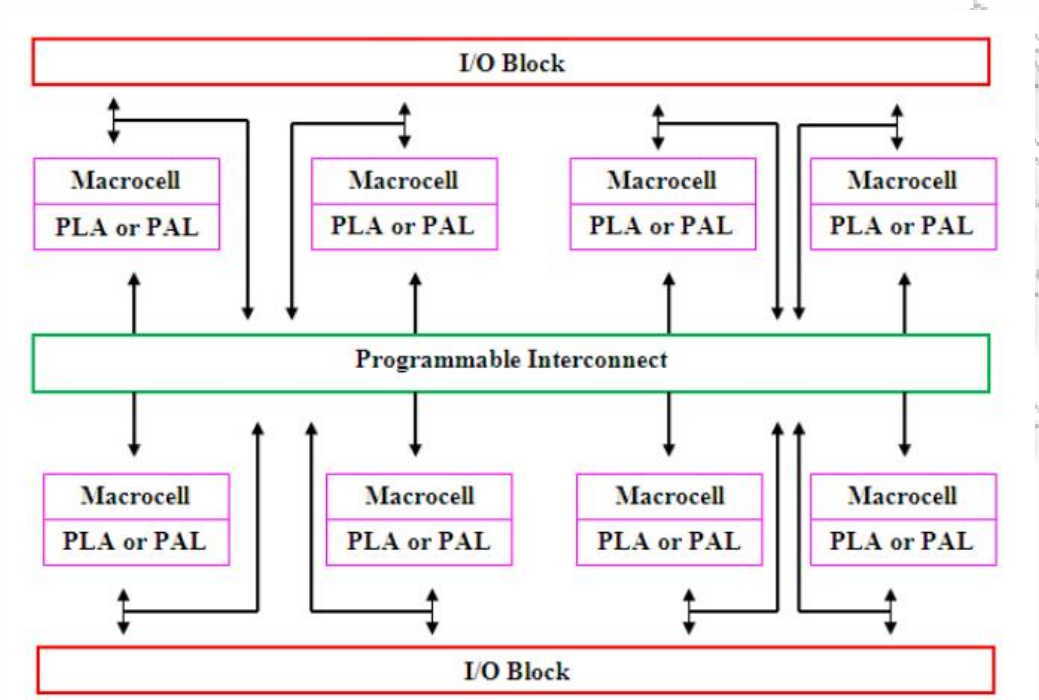


Εικόνα 2.8 Παράδειγμα μιας μακροκυψέλης (macrocell) σε διάταξη PAL

2.2.4. Πολύπλοκες προγραμματιζόμενες λογικές διάταξης CPLDs

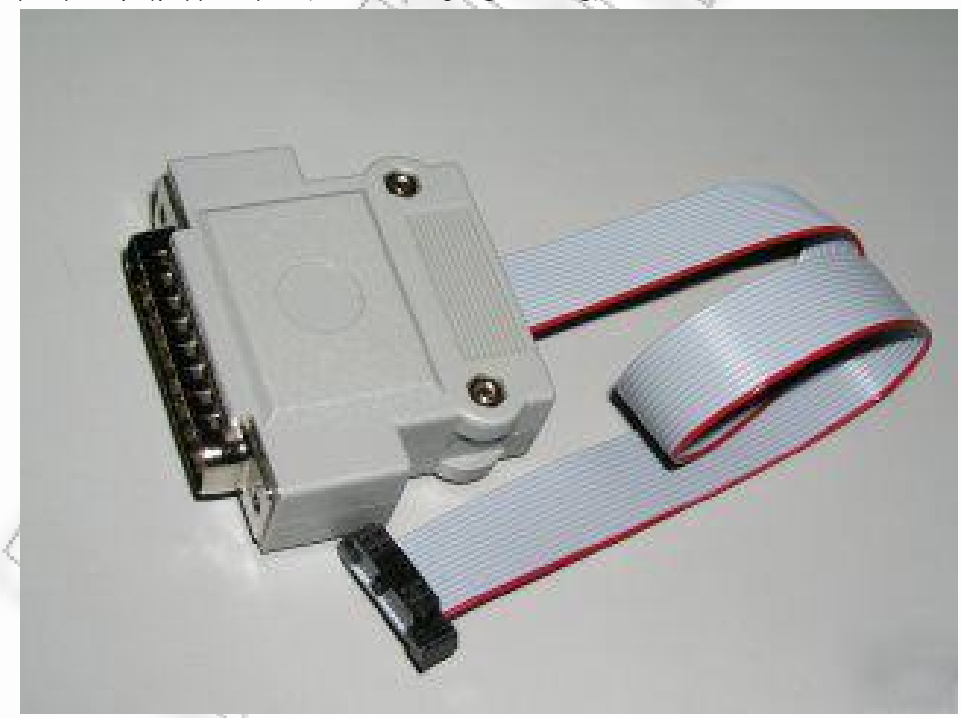
Μέχρι στιγμής έχουμε δει μόνο ΟΚ SPLD, όπως αναφέραμε το χαρακτηριστικό τους είναι ότι είναι μικρά και αυτόνομα ΟΚ, αν θελήσουμε ένα ΟΚ αρκετά πολύπλοκο ώστε να περιέχει περισσότερα από δύο SPLD, τότε πρέπει να αναφερθούμε στις *πολύπλοκες προγραμματιζόμενες λογικές διατάξεις* (*CPLDs, Complexity Programmable Logic Devices*), οι διατάξεις αυτές πρωτοεμφανίστηκαν μεταξύ του τέλους της δεκαετίας του ‘70 και αρχές δεκαετίας ‘80.

Ένα CPLD αποτελείται από πολλές βαθμίδες κυκλωμάτων που βρίσκονται μέσα στο ίδιο ΟΚ και συνδέονται μεταξύ τους με εσωτερικές καλωδιώσεις. Κάθε βαθμίδα κυκλώματος μοιάζει με ένα SPLD, συνήθως είναι PAL η PLA. Κάθε τέτοια βαθμίδα συνδέεται επίσης με ένα υποκύκλωμα, που ονομάζεται *βαθμίδα εισόδου / εξόδου* (*I/O Block*) και προσαρμόζεται σε ένα αριθμό ακροδεκτών εισόδου και εξόδου του ΟΚ, Εικόνα 2.9. Οι εσωτερικές καλωδιώσεις περιέχουν *προγραμματιζόμενους διακόπτες διασύνδεσης* (*PIS*, *Programmable Interconnect Switches*), οι οποίοι χρησιμοποιούνται για να διασυνδέσου μεταξύ τους τις βαθμίδες SPLD. Στο εμπόριο τα CPLD χρησιμοποιούνται για την υλοποίηση πολλών ειδών ψηφιακών κυκλωμάτων και έχουν μεγέθη που κυμαίνονται μεταξύ δυο βαθμίδων SPLD και περισσοτέρων από εκατό SPLD.



Εικόνα 2.9 [int3] Γενική αρχιτεκτονική δομή ενός CPLD

Τα κυκλώματα της διάταξης CPLD προγραμματίζονται μέσω μια θύρας που ονομάζεται *JTAG* (*JTAG port, Joint Test Action Group*), Εικόνα 2.10. Αυτή η διαδικασία προγραμματισμού έχει προτυποποιηθεί από τον οργανισμό *IEEE* [I3E] τόσο για συσκευές CPLD όσο και για συσκευές FPGA. Εφόσον προγραμματιστεί ένα CPLD, διατηρεί μόνιμα την πληροφορία που έχει τοποθετηθεί σε αυτήν ακόμη και αν διακοπεί η τροφοδοσία του ΟΚ. Η διαδικασία αυτή ονομάζεται *μη- πτητικός προγραμματισμός* ή *μόνιμος προγραμματισμός* (*non-volatile programming*).

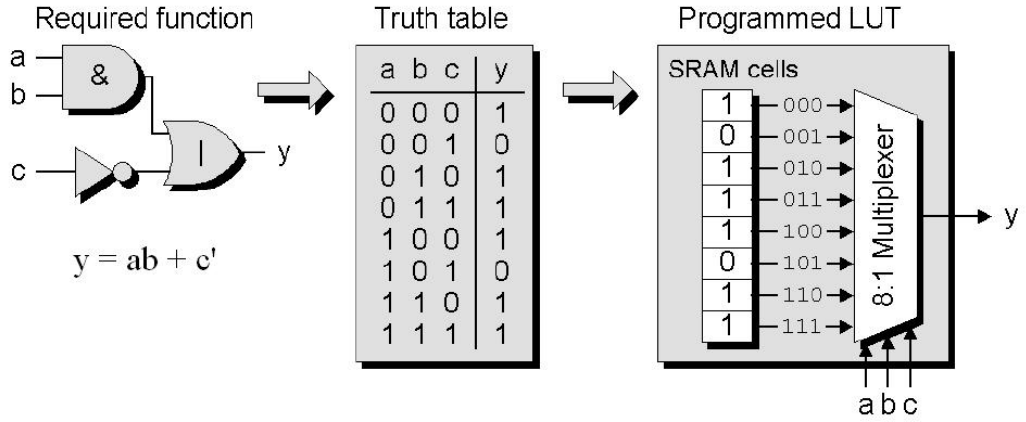


Εικόνα 2.10 Θύρα JTAG της εταιρίας Xilinx για CPLD και FPGA

2.2.5. Επιτόπου προγραμματιζόμενες διατάξεις πυλών FPGAs

Οι μορφές ΟΚ που έχουν περιγράφει έως αυτό το σημείο, η οικογένεια 7400 και τα SPLD και CPLD, είναι χρήσιμες για την υλοποίηση ενός μεγάλου αριθμού λογικών κυκλωμάτων. Εκτός από τα CPLD, τα παραπάνω ΟΚ είναι εν γένει μικρού μεγέθους και είναι κατάλληλα μόνο για σχετικά απλές εφαρμογές. Ακόμη και στην περίπτωση των CPLD, μόνο μετρίως μεγάλα λογικά κυκλώματα μπορούν να χωρέσουν σε ένα ΟΚ. Για την υλοποίηση μεγαλύτερων κυκλωμάτων είναι βολικό να χρησιμοποιήσουμε ένα διαφορετικό είδος ΟΚ που έχει μεγαλύτερη χωρητικότητα. Μια από τις πλέον εξελιγμένες εκδοχές των PLDs είναι οι *Επιτόπου Προγραμματιζόμενες Διατάξεις Πυλών FPGAs* (*Field Programmable Gate Arrays*). Τα FPGAs είναι διατάξεις προγραμματιζόμενης λογικής και υποστηρίζουν την υλοποίηση μεγάλων κυκλωμάτων και επιτρέπουν πλήρη ελευθερία όσον αφορά τη διαδικασία σχεδίασης. Οι συσκευές FPGAs είναι συσκευές γενικής χρήσης, οι οποίες εμπεριέχουν ένα μεγάλο αριθμό *λογικών στοιχείων* (*LE, Logic Element*), καλωδίων διασύνδεσης και διακοπτών. Η συγκεκριμένη διάταξη δεν φαίνεται στην Εικόνα 2.1, αλλά αποτελεί μέλος της οικογενείας PLDs όπως θα δείτε και από τα παρακάτω.

Η αρχιτεκτονική δομή των FPGAs διαφέρει σημαντικά από αυτή των διατάξεων SPLD και CPLD επειδή δεν περιέχουν πύλες AND και OR, αλλά περιέχει *λογικές βαθμίδες* για την υλοποίηση των ζητούμενων συναρτήσεων. Η πιο ευρέως χρησιμοποιημένη λογική βαθμίδα είναι ο *πίνακας αναζήτησης* (*LUT, Lookup Table*), Εικόνα 2.11, ο οποίος περιέχει *κυψέλες αποθήκευσης* (*storage cells*) που χρησιμοποιούνται για την υλοποίηση μιας μικρής συνάρτησης. Κάθε κυψέλη είναι μια μνήμη τύπου *SRAM* (*Static Random Access Memory*) και μπορεί να αποθηκεύσει μια λογική τιμή, 0 ή 1. Η αποθηκευμένη τιμή μεταφέρεται στην έξοδο της κυψέλης αποθήκευσης. Μπορούν να αναπτυχθούν πίνακες LUT σε διάφορα μεγέθη, όπου το μέγεθος ορίζεται από τον αριθμό των εισόδων. Ένα LUT n-εισόδων μπορεί να υλοποιήσει όλες τις πιθανές συνδυαστικές συναρτήσεις των n-εισόδων, προσθέτοντας μία ακόμη είσοδο είναι δυνατή η αναπαράσταση πιο σύνθετων συναρτήσεων. Μελέτες έχουν δείξει ότι τα LUT 4-εισόδων είναι μία "καλή" λύση [PsM1].



Εικόνα 2.11 [Max] Πίνακας αναζήτησης (LUT, Lookup Table) αρχιτεκτονικής FPGA

Οι κυψέλες αποθήκευσης των πινάκων LUT είναι *πτητικές* ή *μη μόνιμες* (*volatile*), αυτό συνεπάγεται ότι χάνουν τα δεδομένα που περιέχουν εάν διακοπεί η τροφοδοσία του ΟΚ. Δεδομένου αυτού του προβλήματος, θα έπρεπε το FPGA να προγραμματίζεται κάθε φορά που θα εφαρμόζονταν η τροφοδοσία στο κύκλωμα. Για την αποφυγή αυτού του προβλήματος, συμπεριλαμβάνεται συχνά μαζί με τη μητρική πλακέτα του FPGA και ένα μικρό ΟΚ μη-πτητικής μνήμης. Αυτή η μνήμη είναι της μορφής PROM ή EPROM ή EEROM ή οτιδήποτε άλλο. Η μνήμη αυτή διατηρεί σε μόνιμη βάση τα δεδομένα της διάταξης τα οποία έχουν εισέλθει μέσω της θύρας JTAG. Κάθε φορά που γίνεται τροφοδοσία στη συσκευή FPGA, διαβάζεται η μνήμη και διαμορφώνεται στη συσκευή η συγκεκριμένη λειτουργία για την οποία σχεδιάστηκε. Για περισσότερες λεπτομέρειες γύρω από την αρχιτεκτονική δομή ενός FPGA θα σας παραπέμψουμε στο Κεφάλαιο 3.

2.3. Ειδικά Ολοκληρωμένα Κυκλώματα ASICs

Ένα πολύ μεγάλο μειονέκτημα των συσκευών PLDs είναι ότι οι προγραμματιζόμενοι διακόπτες που διαθέτουν, καταλαμβάνουν πολύτιμο χώρο πάνω στο ΟΚ και περιορίζουν την ταχύτητα λειτουργίας των κυκλωμάτων που υλοποιούνται με αυτά. Επομένως σε μερικές περιπτώσεις τα PLDs ενδέχεται να μη πληρούν τις προδιαγραφές επίδοσης που επιθυμεί ο χρήστης ή το κόστος αγοράς τους να μην είναι επιτρεπτό. Στις περιπτώσεις αυτές θα ήταν προτιμότερο να κατασκευαστεί ένα ΟΚ από την αρχή, ώστε να μπορέσει να εκπληρώσει τα επιθυμητά αποτελέσματα φθηνότερα αλλά και αποτελεσματικότερα. Τέτοια κυκλώματα που προορίζονται για χρήση σε συγκεκριμένες εφαρμογές ονομάζονται *ειδικά ολοκληρωμένα κυκλώματα εφαρμογής* (*ASICs, Application-Oriented Integrated Circuit*). Η προσέγγιση που γίνεται είναι *ειδική σχεδίαση* (*custom design*) για συγκεκριμένη λειτουργία και αυτά τα ΟΚ ονομάζονται *ειδικά ΟΚ* (*custom chips*) γιατί επιτελούν μια ειδική λειτουργία.

Το κύριο πλεονέκτημα των ASICs είναι ότι η σχεδίαση τους μπορεί να βελτιστοποιηθεί προς την κατεύθυνση κάποιας λειτουργίας και αυτό οδηγεί συνήθως σε καλύτερες επιδόσεις. Επιπλέον το κόστος παραγωγής τέτοιων ΟΚ είναι υψηλό, αλλά συνήθως το συγκεκριμένο προϊόν όταν πρόκειται να πουληθεί σε μεγάλες ποσότητες, το κόστος ανά τεμάχιο μπορεί να μειωθεί πολύ και να γίνει μικρότερο από το κόστος ενός εμπορικού τεμαχίου που εκτελεί την ίδια λειτουργία. Επιπρόσθετα, εάν μπορεί να χρησιμοποιηθεί για την εκτέλεση μιας λειτουργίας ένα ΟΚ αντί για δυο ή περισσότερα, εξοικονομείται χώρος στην πλακέτα (PCB) του τελικού προϊόντος και έτσι το κόστος μειώνεται ακόμη περαιτέρω.

Το μεγάλο μειονέκτημα των συσκευών ASICs είναι ότι, ακριβώς όπως προαναφέραμε, εκτελούν μια συγκεκριμένη λειτουργία. Αν κατά τη χρήση της συγκεκριμένης συσκευής προκύψουν νέες λειτουργίες τότε η συσκευή ASICs δεν θα μπορέσει να ανταποκριθεί στις νέες προκλήσεις. Θα πρέπει να γίνει μια νέα σχεδίαση, να συμπεριληφθούν οι νέες λειτουργίες και στη συνέχεια να κατασκευαστεί εκ νέου. Αυτή η διαδικασία είναι χρονοβόρα, διαρκεί 6 με 8 μήνες, οπότε είναι μη λειτουργικό να χρησιμοποιείς συσκευές ASICs σε καταστάσεις που δεν είναι σταθερές. Αυτό όμως είναι το μεγάλο πλεονέκτημα των συσκευών PLDs, δηλαδή ότι μπορούν να επαναπρογραμματίζονται με μηδαμινό κόστος.

3. Αρχιτεκτονικές FPGAs

Στην Ενότητα 2.2.5 έχουμε κάνει μια πολύ μικρή εισαγωγή για τις διατάξεις των FPGAs, σε αυτό το κεφάλαιο θα ασχοληθούμε διεξοδικά με την αρχιτεκτονική δομή των διατάξεων αυτών, ξεκινώντας από την Ενότητα 3.1 με μια ιστορική και μια γενική περιγραφή. Κατόπιν θα αναλύσουμε τρεις διαφορετικές αρχιτεκτονικής δομές FPGAs από τρεις διαφορετικές εταιρίες. Η πρώτη που θα δούμε είναι στην Ενότητα 3.2 και είναι η οικογένεια συσκευών *Cyclone* [Alt1] της εταιρίας *Altera* [Alt4]. Αμέσως μετά, στην Ενότητα 3.3 θα δούμε την οικογένεια συσκευών *Spartan-3* [Xln1] της εταιρίας *Xilinx* [Xln5]. Στην Ενότητα 3.4 θα δούμε την οικογένεια συσκευών *AT40K* [Atm1]της εταιρίας *Atmel* [Atm4]. Και τέλος στην Ενότητα 3.5 θα δούμε τις διαφορές των τριών αυτών αρχιτεκτονικών δομών και θα καταλήξουμε σε κάποια συμπεράσματα.

Στο σημείο αυτό θα πρέπει να σας αναφέρουμε ότι η επιλογή των συγκεκριμένων οικογενειών δεν έγινε τυχαία. Αρχική μας σκέψη ήταν σε αυτή τη μεταπτυχιακή διατριβή να συγκρίνουμε τρεις διαφορετικές αρχιτεκτονικές δομές FPGAs αλλά και τη δυνατότητα των εργαλείων σχεδίασης CAD των εταιριών αυτών, να διευθετήσουν όσον το δυνατό καλύτερα τα προτεινόμενα *σχεδιαστικά πρότυπα* ή αλλιώς *μέτρο-προγράμματα* (*benchmark programs*). Είχε γίνει επιλογή κάποιων συσκευών - διατάξεων με βάση τα λογικά στοιχεία (LE) που διαθέτει η κάθε οικογένεια. Η επιλογή έγινε έτσι ώστε και οι τρεις διατάξεις να έχουν περίπου τα ίδια λογικά στοιχειά (LE), λέμε περίπου διότι δεν μπορούσε να γίνει συμφωνία στο μέγεθος των λογικών στοιχείων λόγω της διαφορετικότητας των αρχιτεκτονικών δομών. Αυτές οι συσκευές ήταν:

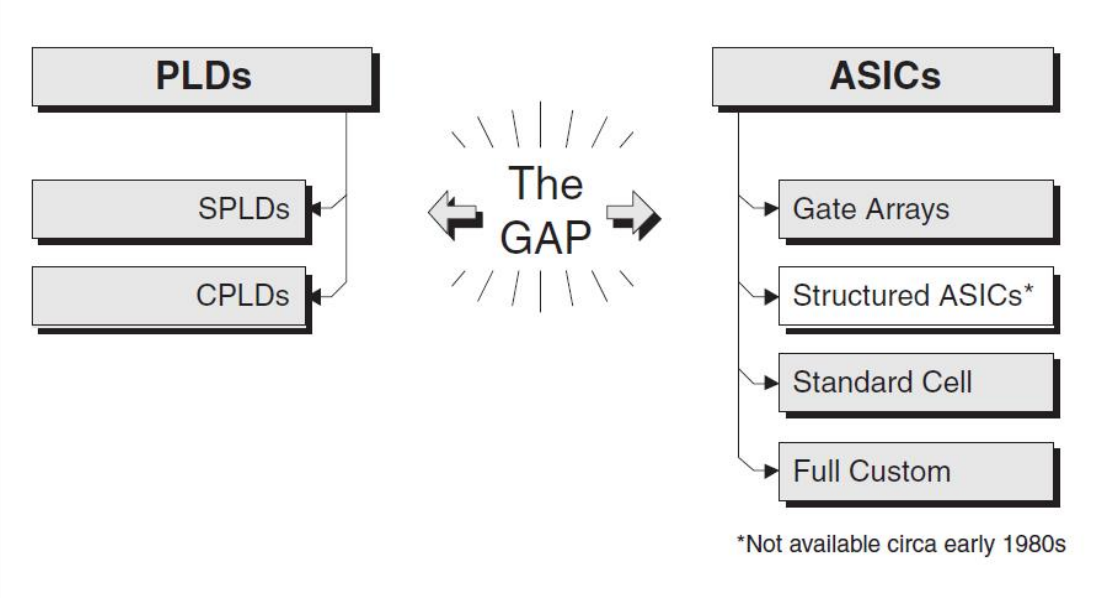
1. Η συσκευή EP1C3 της εταιρίας Altera με 2,910 λογικά στοιχεία.
2. Η συσκευή XC3S50 της εταιρίας Xilinx με 1,728 λογικά στοιχεία.
3. Η συσκευή AT40K40 της εταιρίας Atmel με 2,304 λογικά στοιχεία

Όμως τα προβλήματα που παρουσιάστηκαν δεν μας έδωσαν τη δυνατότητα προσομοίωσης και των τριών αρχιτεκτονικών δομών αλλά μόνο των δυο. Στη συνέχεια της διατριβής εξηγούνται οι λόγοι αυτής της επιλογής.

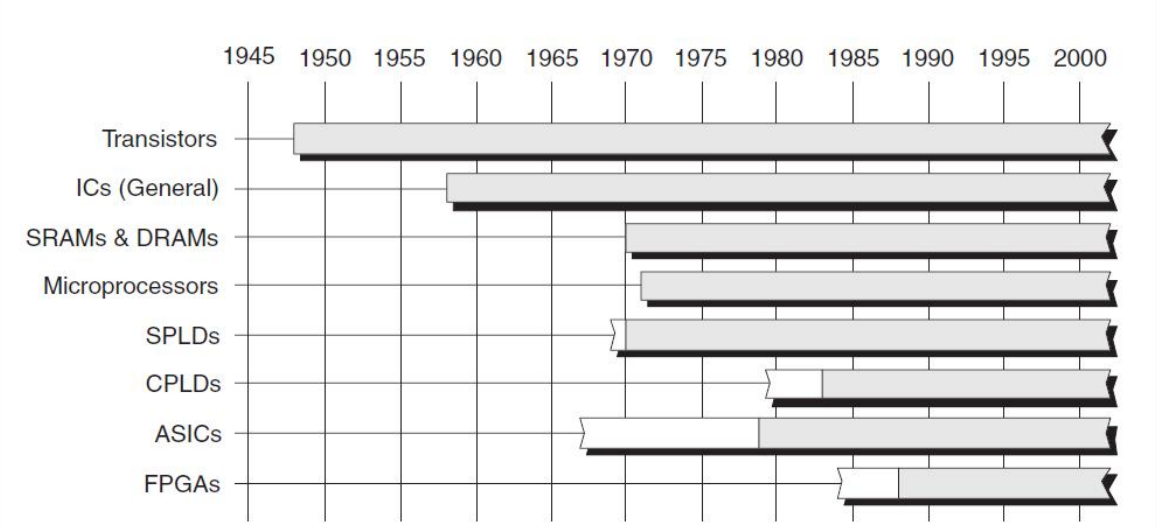
3.1. Γενική Αρχιτεκτονική Δομή FPGAs

3.1.1. Αρχιτεκτονική δομή

Η πρώτη εμφάνιση των διατάξεων FPGAs έγινε αρχές της δεκαετίας του ’80 και συγκεκριμένα το 1984 όταν η εταιρία Xilinx θέλησε να μετριάσει το χάσμα που επικρατούσε μεταξύ των διατάξεων PLDs και ASICs, Εικόνα 3.1. Αποτέλεσμα αυτής της προσπάθειας ήταν η πρώτη διάταξη FPGA, που όμως άργησε αρκετά να απορροφηθεί από την αγορά μιας και οι μηχανικοί δεν τα είχαν ακόμη σε εκτίμηση, όλα αυτά όμως μέχρι τις αρχές του 1990. Στη σημερινή αγορά υπάρχει πληθώρα από διαφορετικές ποικιλίες των συσκευών αυτών, όπου ολοένα αναπτύσσονται και εξελίσσονται με γρήγορους ρυθμούς. Παράλληλα βλέπουμε ένα ιδιαίτερο ενδιαφέρον από ακαδημαϊκής πλευράς στη μελέτη αυτής της τεχνολογίας μιας και αποτελεί τη σύγχρονη μορφή των ψηφιακών συστημάτων και της ψηφιακής σχεδίασης. Στην Εικόνα 3.2 βλέπουμε την εξέλιξη των ΟΚ, από τις πρώτες μορφές των τρανζίστορ μέχρι τις τελευταίες αυτές των FPGA.

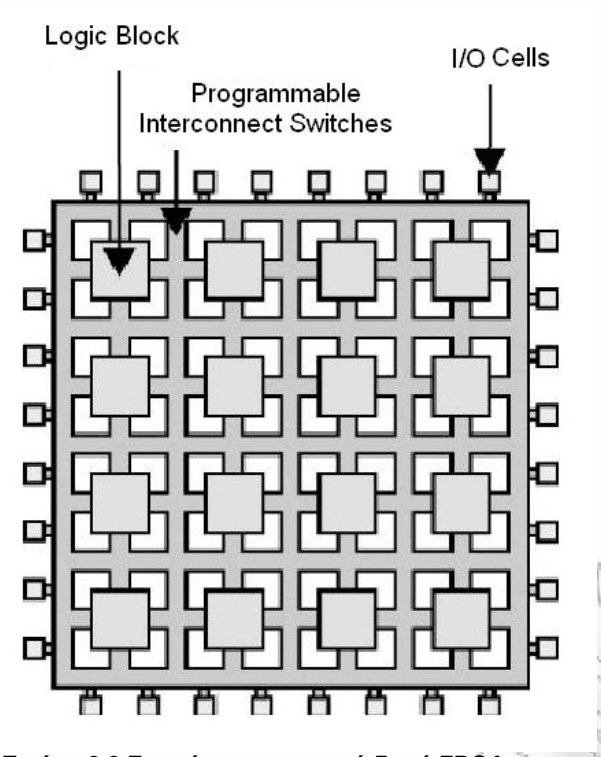


Εικόνα 3.1 [Max] Το χάσμα μεταξύ διατάξεων PLDs και ASICs



Εικόνα 3.2 [Max] Χρονική εξέλιξη της τεχνολογίας των ολοκληρωμένων κυκλωμάτων, (οι ημερομηνίες είναι κατά προσέγγιση)

Η γενική αρχιτεκτονική δομή των FPGAs είναι αυτή της Εικόνας 3.3, όπου μπορεί κανείς να παρατηρήσει ότι αποτελείται από τρεις χαρακτηριστικές ομάδες. Η πρώτη ομάδα αποτελείται από τους *ακροδέκτες εισόδου εξόδου* (*pins*, *I/O Cells*) τα οποία είναι και η διασύνδεση της συσκευής με τον έξω κόσμο, αποτελούν δηλαδή την είσοδο και την έξοδο του FPGA. Η δεύτερη ομάδα αποτελείται από τις *λογικές βαθμίδες* (*LB*, *Logic Block*) όπου το καθένα από αυτά περιέχει ένα σύνολο από λογικά στοιχεία (ο αριθμός διαφέρει αν αρχιτεκτονική). Και τέλος η τρίτη ομάδα με τους *προγραμματιζόμενους διακόπτες διασύνδεσης* (*PIS*, *Programmable Interconnect Switches*) ή αλλιώς *κανάλια δρομολόγησης* (*Routing Channels*). Η τελευταία ομάδα είναι αυτή που διασύνδεει τις λογικές βαθμίδες (LB) μεταξύ τους όπως και με τους ακροδέκτες (I/O). Όπως καταλαβαίνετε η ποσότητα αυτών των χαρακτηριστικών (I/O, LE, LB,PIS κτλ) διαφέρει από αρχιτεκτονική σε αρχιτεκτονική, όμως η γενική δομή παραμένει η ίδια.



Εικόνα 3.3 Γενική αρχιτεκτονική δομή FPGA

Επιπλέον ενσωματωμένα χαρακτηριστικά τα οποία δε φαίνονται στην Εικόνα 3.3 είναι τα παρακάτω:

- Μνήμες (*Memories*) RAM
- Αριθμητικές μονάδες (*Arithmetic Units*)
- Επεξεργαστές (*Microprocessors*)
- Διευθυντές ρολογιού (*CLKm,Clock Manager*)
- Λογικό στοιχείο (*LE, Logic Element*)
- Γρήγορες αλυσίδες κρατουμένου (*Fast Carry Chains*)

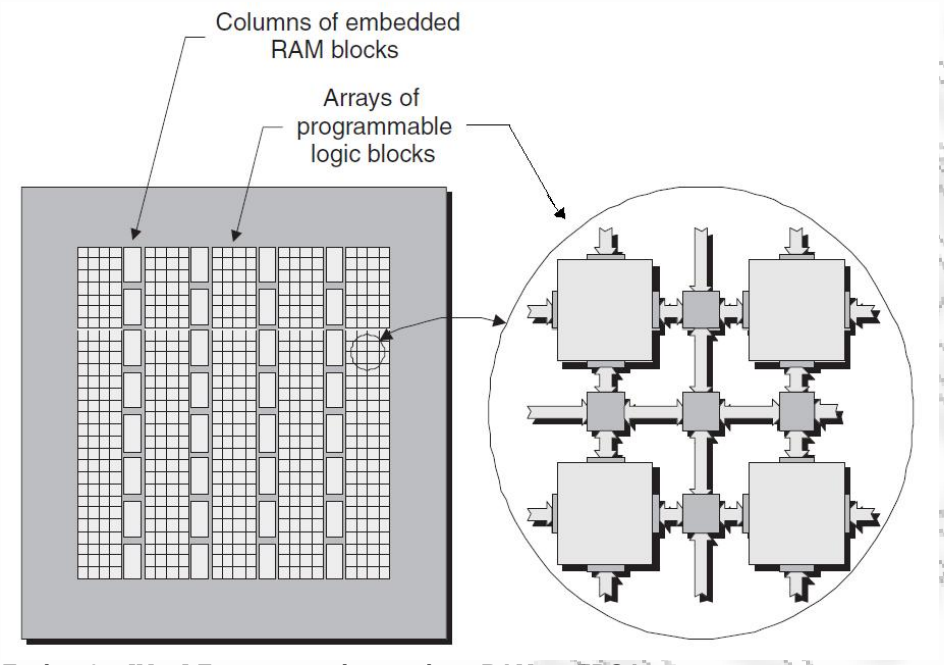
Όλα τα χαρακτηριστικά θα αναλυθούν εκτενέστερα παρακάτω, εκτός αυτά των ακροδεκτών και των διασυνδέσεων μιας και αυτά διαφέρουν κατά πολύ ανά αρχιτεκτονική.

3.1.2. Μνήμες (Memories) RAM

Μνήμες (*Memories*) ή *βαθμίδες μνημών* (*MB, Memory Block*), μπορεί να αποτελούν μέρος της αρχιτεκτονικής δομής, ανάμεσα σε λογικές βαθμίδες (LB) η περιφερειακά της συσκευής FPGA (βλέπε Εικόνα 3.4). Αυτές οι βαθμίδες μνημών (MB) εξυπηρετούν τοπικές ανάγκες του κυκλώματος παρέχοντας επιπλέον μνήμη στις λογικές βαθμίδες. Είναι συνήθως μνήμες RAM και λόγω της ανάγκης για μεγάλη διάθεση μνήμης, πολλές από τις συσκευές FPGA ενσωματώνουν βαθμίδες από RAM (*RAM Block*). Κάθε βαθμίδα RAM μπορεί να χρησιμοποιηθεί ανεξάρτητα ή να συνδυαστούν πολλές μαζί ώστε να υλοποιήσουν μία μεγαλύτερη μνήμη. Επίσης οι μνήμες αυτές μπορούν να χρησιμοποιηθούν σε ποικιλία εφαρμογών όπως

- Single Port Ram
- Dual Port Ram
- FIFO, κτλ

Στις αρχιτεκτονικές που θα αναπτύξουμε παρακάτω θα δείτε τη δομή αλλά και τη θέση που καταλαμβάνουν στην αρχιτεκτονική της εκάστοτε οικογένειας.

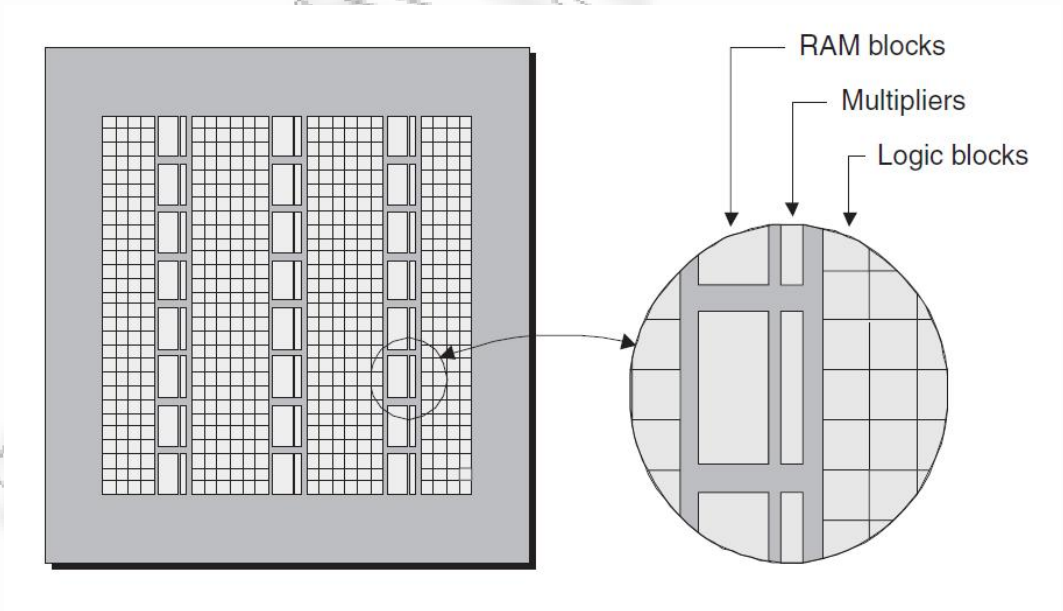


Εικόνα 3.4 [Max] Ενσωματωμένες μνήμες RAM σε FPGA

3.1.3. Αριθμητικές μονάδες (Arithmetic Units)

Αριθμητικές μονάδες (*Arithmetic Units*), κάποιες αριθμητικές συναρτήσεις, όπως ο πολλαπλασιασμός, είναι αργές όταν υλοποιηθούν από ένα αριθμό από λογικά στοιχειά. Για αυτό το λόγο ενσωματώνονται αριθμητικές μονάδες στην αρχιτεκτονική δομή των FPGAs, έτσι ώστε να επιταχύνουν τη διαδικασία, Εικόνα 3.5. Πολλές διατάξεις FPGA ενσωματώνουν τέτοιες αριθμητικές μονάδες όπως

- Πολλαπλασιαστές (multipliers)
- Αθροιστές (Adders)
- Πολλαπλασιαστές - Συσσωρευτές (MAC, Multiple-Accumulator)



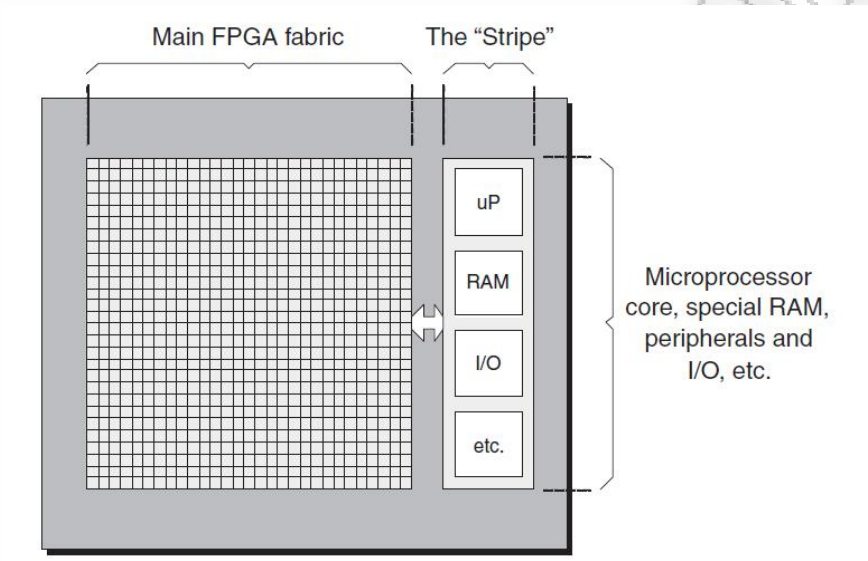
Εικόνα 3.5 [Max] Ενσωματωμένες αριθμητικές μονάδες σε FPGA

Οι ενσωματωμένες αριθμητικές μονάδες μαζί με τις ενσωματωμένες μνήμες κάνουν ιδανική τη χρήση των FPGAs σε εφαρμογές *ψηφιακής επεξεργασίας σήματος* (*DSP, Digital Signal Processing*). Άλλωστε

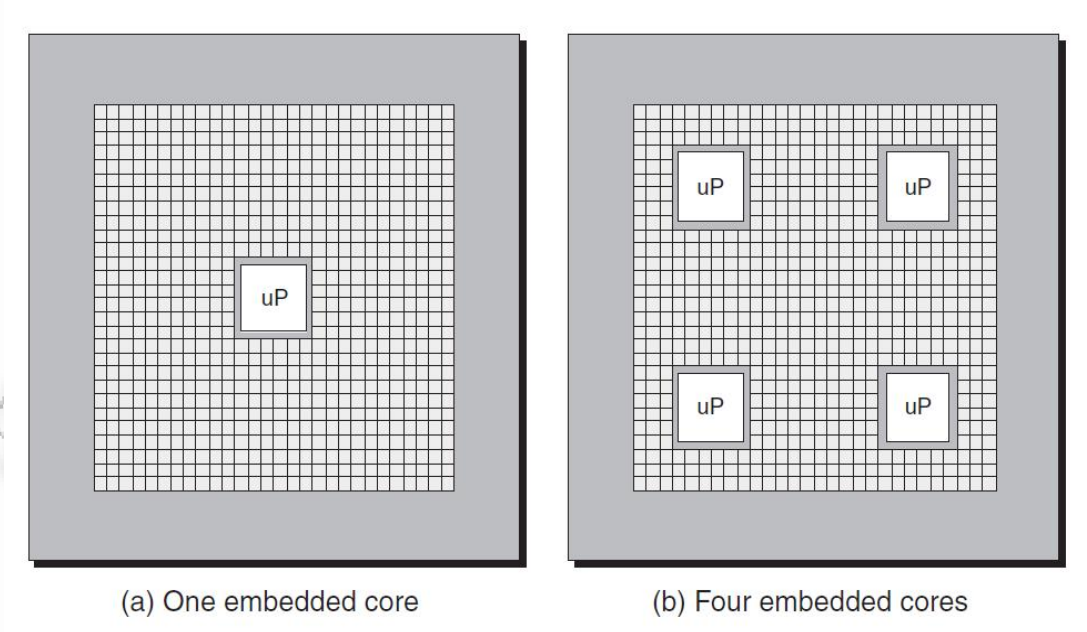
δεν είναι τυχαίο ότι ένα μεγάλο μερίδιο της αγοράς των FPGAs, αποτελεί προϊόν για οπτικοακουστικούς καταναλωτές.

3.1.4. Επεξεργαστές (Microprocessors)

Επεξεργαστές (*Microprocessors*), οι προηγμένες διατάξεις FPGAs ενσωματώνουν *πυρήνες επεξεργαστών* (*Microprocessors Cores*), για χρήση σε πολύ απαιτητικές λειτουργίες όπου χρειάζεται ένας γρήγορος επεξεργαστής. Υπάρχουν δύο τύποι ενσωματωμένων επεξεργαστών, οι *σκληρού πυρήνα* (*Hard Core*) και οι *μαλακού πύρινα* (*Soft Core*). Οι επεξεργαστές μαλακού πύρινα υλοποιούνται μέσω των λογικών στοιχείων της διάταξης και μπορούν να τροποποιηθούν και να προσαρμοστούν στις ανάγκες του χρήστη. Οι σκληρού πυρήνα ενσωματώνονται στη διάταξη σαν ένα επιπλέον OK και δεν είναι δυνατόν να τροποποιηθεί από το χρήστη, Εικόνα 3.6. Ο αριθμός των επεξεργαστών που μπορούν να ενσωματωθούν εξαρτάται από την αρχιτεκτονική, στην Εικόνα 3.7 βλέπουμε κάποια από αυτά.



Εικόνα 3.6 [Max] Σκληρός πυρήνας (Hard Core) ενός FPGA έξω από την κύρια αρχιτεκτονική δομή του



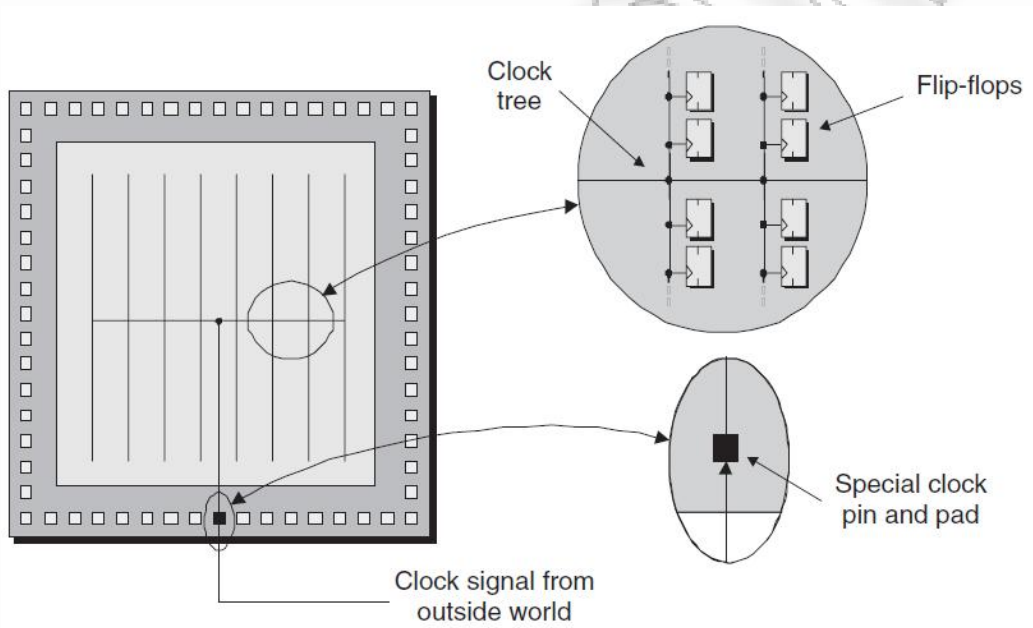
Εικόνα 3.7 [Max] Ενσωματωμένοι επεξεργαστές σε διαφορετικές αρχιτεκτονικές δομές

3.1.5. Διευθυντές ρολογιού (CLKm, Clock Managers)

Διευθυντές ρολογιού (*CLKm, Clock Managers*), ένα από τα σημαντικότερα χαρακτηριστικά σε ένα FPGA. Οι διευθυντές ρολογιού (CLKm) παρέχουν σε όλη την αρχιτεκτονική δομή του FPGA το σήμα του ρολογιού ή σήματα διαφορετικών ρολογιών. Σκοπός της παρουσία τους είναι:

- Αποφυγή του φαινομένου *Clock Skew*, το οποίο είναι η μη σωστή και ταυτόχρονη άφιξη του σήματος του ρολογιού σε όλα τα λογικά στοιχειά της διάταξης.
- Αφαίρεση *παραμόρφωσης χρονισμού* (*jitter removal*).
- *Σύνθεση συχνότητας* (*frequency synthesis*), υπάρχουν ποικιλίες συχνοτήτων που μπορούν να διαμορφωθούν με έναν διευθυντή ρολογιού (CLKm).
- *Ολίσθηση φάσης* (*phase shifting*).
- *Διόρθωση αυτό - απόκλισης* (*auto - skew correction*).

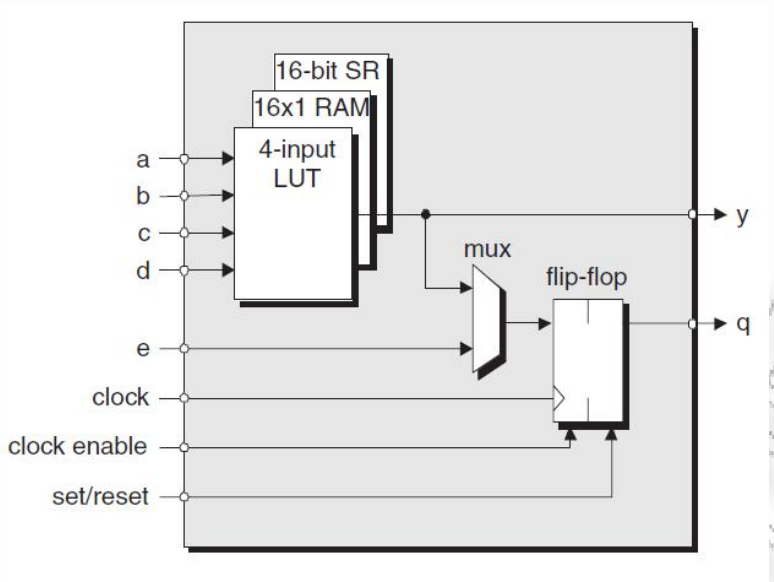
Ο τρόπος αποφυγής μερικών από τα παραπάνω προβλήματα επιτυγχάνεται με την τοποθέτηση *δέντρων ρολογιού* (*clock trees*), πολλαπλών *ακροδεκτών* (*clock pins*) και πολλαπλών *πεδίων ρολογιού* (*clock domains*), Εικόνα 3.8. Όπως θα δούμε στις παρακάτω αρχιτεκτονικές, σε μερικές περιπτώσεις, υπάρχουν περισσότερα από ένα ή δυο πεδία ρολογιού όπου το καθένα από αυτά έχει το δικό του δέντρο και τους δικούς του ακροδέκτες.



Εικόνα 3.8 [Max] Αναπαράσταση ενός δέντρου ρολογιού

3.1.6. Λογικό στοιχείο (LE, Logic Element)

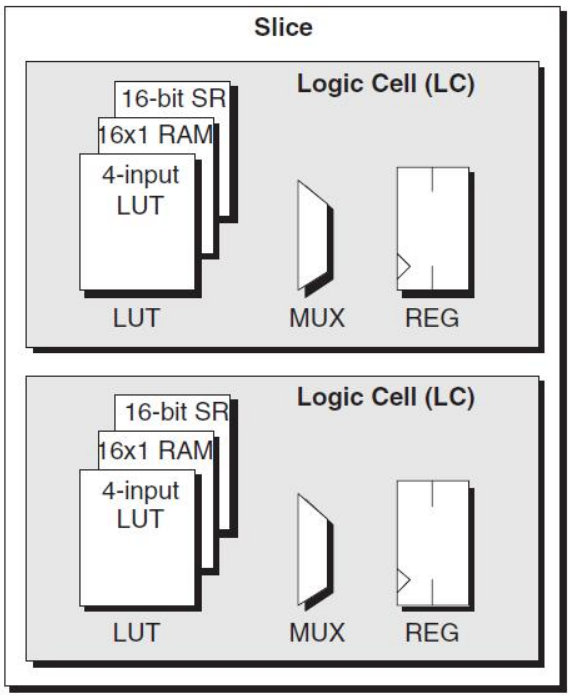
Λογικό στοιχείο (*LE, Logic Element*), σε αυτό το σημείο θα πρέπει να αναλύσουμε λίγο την ύπαρξη του λογικού στοιχείου. Το λογικό στοιχείο αποτελεί την ελάχιστη λογική μονάδα στην οποία μπορεί να αποθηκευτεί μια συνάρτηση ή ένα δεδομένο. Στη βιβλιογραφία όπως και στην αγορά υπάρχει μία πληθώρα ονομασία αυτών των στοιχείων. Μπορεί κανείς να τα δει ως *λογικό κύτταρο* (*logic cell*), ως λογικό στοιχείο (logic element), ως *κύτταρο* (*cell*) κτλ. Εμείς σε όλη τη διατριβή θα αναφερόμαστε σε αυτή τη λογική μονάδα ως λογικό στοιχείο (LE). Με εξαίρεση όταν θα γίνει παρουσίαση των αρχιτεκτονικών των τριών εταιριών, θα αναφερόμαστε στα λογικά στοιχεία όπως τα αναφέρει ο κατασκευαστής τους. Ένα λογικό στοιχείο αποτελείται από ένα πίνακα αναζήτησης (Lookup Table, LUT), ο οποίος όμως μπορεί να διαφέρει στον αριθμό των εισόδων, Εικόνα 2.11. Επιπλέον ένα λογικό στοιχείο περιέχει πληθώρα από λογικές πύλες, κυκλώματα όπως και κύτταρα μνήμης flip flop, Εικόνα 3.9. Είναι αναμενόμενο λοιπόν η αρχιτεκτονική κάθε λογικού στοιχείο σε διαφορετικές εταιρίες και οικογένειες να διαφέρει.



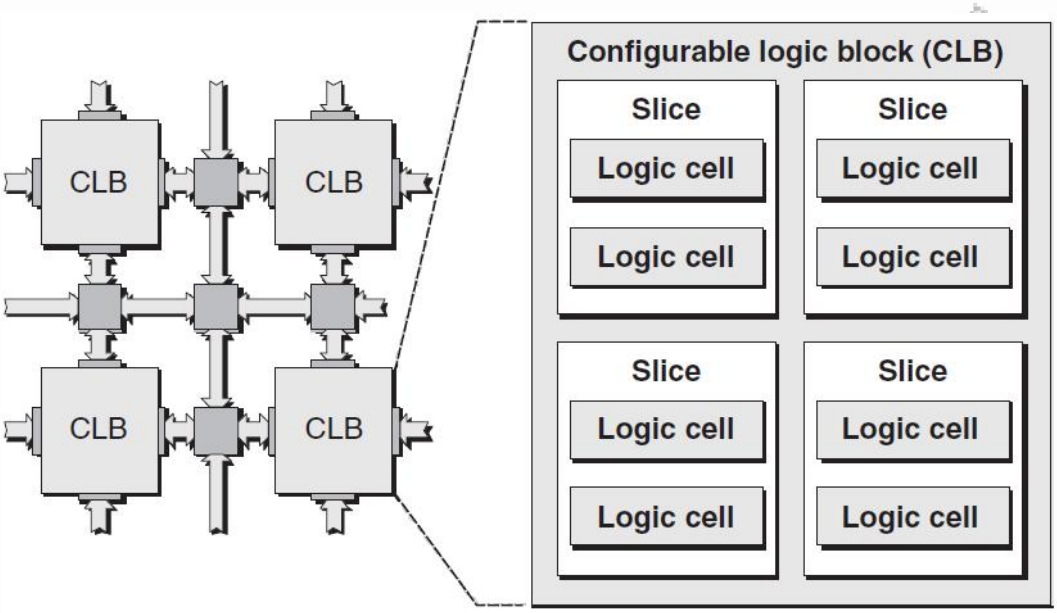
Εικόνα 3.9 [Max] Λογικό στοιχείο της εταιρίας Xilinx

3.1.7. Διαμορφωμένη λογική βαθμίδα (CLB, Configurable Logic Block)

Η αμέσως επόμενη μεγαλύτερη λογική μονάδα είναι ένα σύνολο από λογικά στοιχεία. Υπάρχουν διαφορετικές αρχιτεκτονικές όπου κάποιες από αυτές περιλαμβάνουν δυο λογικά στοιχεία και αποτελούν μία "φέτα" (slice), Εικόνα 3.10. Κάποιες άλλες απλώς έχουν πολύ περισσότερα λογικά στοιχεία, γύρω στις 8 με 10 και αποτελούν μια *διαμορφωμένη λογική βαθμίδα* (*CLB, Configurable Logic Block*), Εικόνα 3.11. Τις διαμορφωμένες λογικές βαθμίδες (CLB) τις έχουμε δει και ως λογικές βαθμίδες (LB), αλλά μπορούμε να τις δούμε και ως *συστοιχία λογικών βαθμίδων* (*LAB, Logic Array Block*). Εμείς θα αναφερόμαστε σε αυτές τις μονάδες ως διαμορφωμένες λογικές βαθμίδες (CLB).



Εικόνα 3.10 [Max] Ένα slice που περιέχει δυο λογικά στοιχεία της εταιρίας Xilinx



Εικόνα 3.11 [Max] Διαμορφωμένη λογική βαθμίδα (CLB, Configurable Logic Block) της εταιρίας Xilinx

Ένα άλλο χαρακτηριστικό που έχει να κάνει με τα λογικά στοιχεία και ειδικά με τους πίνακες αναζήτησης, είναι ότι συγκεκριμένα κάποια από αυτά μπορούν να λειτουργήσουν ως μνήμη ή ως καταχωρητές ολίσθησης. Λέμε κάποια επειδή υπάρχουν εταιρίες που δεν ενσωματώνουν αυτή την παραλλαγή στην αρχιτεκτονική τους. Όπως είδατε και στην Εικόνα 3.9 ένα πίνακα αναζήτησης 4-εισόδων μπορεί να λειτουργήσει ως μνήμη 16×1 RAM ή ως 16-bit καταχωρητής ολίσθησης.

3.1.8. Γρήγορες αλυσίδες κρατουμένου (fast carry chains)

Γρήγορες αλυσίδες κρατουμένου (*fast carry chains*), είναι μία ακόμη πρόσθετη λογική που υλοποιούν τα λογικά στοιχεία. Παράδειγμα, από τα προηγούμενα που έχουμε αναφέρει, για να επεκτείνουμε το μέγεθος τους μπορούμε να συνδέσουμε τις αλυσίδες δυο λογικών στοιχείων (LE) μεταξύ τους, στη συνέχεια δυο "φέτες" (slices) και τέλος δύο διαμορφωμένες λογικές βαθμίδες (CLB). Σε κάθε περίπτωση αυτή η σειρά σύνδεσης των αλυσίδων κρατουμένου, εξαρτάται από την αρχιτεκτονική δομή του εκάστοτε FPGA. Οι γρήγορες αλυσίδες κρατουμένου βελτιώνουν την απόδοση αριθμητικών κυκλωμάτων όπως είναι οι αθροιστές κτλ.

Σε αυτό το σημείο να αναφέρουμε ότι κάποια επιπλέον χαρακτηριστικά μπορούν να υπάρξουν μεμονωμένα και ανεξάρτητα σε κάποιες αρχιτεκτονικές δομές FPGA. Αυτά δεν αποτελούν γενικό χαρακτηριστικό αλλά ειδικό, κάποια από αυτά θα παρουσιαστούν στις παρακάτω αρχιτεκτονικές.

3.2. Αρχιτεκτονική Δομή Altera Cyclone FPGA

3.2.1. Αρχιτεκτονική και γενικά χαρακτηριστικά διασύνδεσης

Η πρώτη συσκευή που θα δούμε αφορά την οικογένεια *Cyclone* της εταιρίας *Altera* [Alt1]. Για αρχή θα παρουσιάσουμε πολύ γρήγορα κάποια χαρακτηριστικά που έχουν να κάνουν με το υλικό, όπως είναι η λειτουργία ισχύος της συσκευής η οποία είναι στα *1,5-V, 0.13-μm*. Όλες οι SRAM αποτελούνται από στρώματα χαλκού, η πυκνότητα είναι μέχρι 20.060 λογικά στοιχεία (LEs) και η μνήμη RAM έως 294,912 bits. Επιπλέον χαρακτηριστικά είναι: κλείδωμα φάσης ρολογιού σε βρόγχους (*PLLs, Phase Locked Loops*), ειδική μνήμη διπλής ταχύτητας δεδομένων (*DDR, Double Data Rate*), μονής ταχύτητας δεδομένων (*SDR, Single Data Rate),* γρήγορου κύκλου ρολογιού μνήμης RAM (*FCRAM, Fast Cycle RAM*). Υποστηρίζει διάφορα πρότυπα εισόδου εξόδου (I/O), συμπεριλαμβανομένων σημάτων διαφορετικών χαμηλών τάσεων (*LVDS, Low-Volt Differential Signals*) με ρυθμούς δεδομένων μέχρι και 640 Mbits ανά δευτερόλεπτο (Mbps), 66 - και 33-MHz, 64-bit και 32-bit περιφερειακή διασύνδεση (*PCI, Peripheral Component Interconnect*), για την υποστήριξη διασύνδεσή τους με συσκευές ASIC κτλ.

Παρακάτω βλέπουμε κάποια από αυτά αλλά και άλλα επιπλέον. Στο Πίνακα 3.1 μπορούμε να δούμε τις συσκευές που παρέχονται από το κατασκευαστή για την οικογένεια συσκευών Cyclone.

Η συσκευές της οικογένειας Cyclone προσφέρουν τις ακόλουθες δυνατότητες:

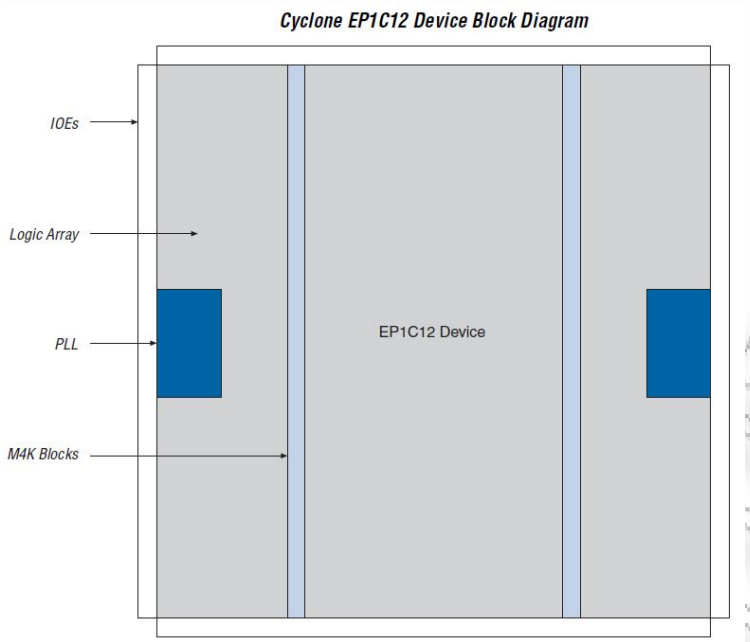
- 2910 εώς 20.060 λογικά στοιχεία (LEs).
- Έως 294.912 RAM bits (36864 bytes).
- Υποστηρίζει διαμόρφωση, μέσω χαμηλού κόστους σειριακής διαμορφωμένης συσκευής.
- Υποστηρίζει πρότυπα για LVTTL, LVCMOS, SSTL-2, και SSTL-3 I / O.
- Υποστηρίζει πρότυπα για 66 - και 33-MHz, 64 - και 32-bit PCI.
- Υποστήριξη υψηλής ταχύτητας (640 Mbps) LVDS I / O.
- Υποστήριξη χαμηλής ταχύτητας (311 Mbps) LVDS I / O.
- Υποστήριξη 311-Mbps RSDS I / O.
- Μέχρι δύο PLLs ανά συσκευή, παρέχει πολλαπλασιασμό ρολόι και μετατόπιση φάσης.
- Μέχρι οκτώ γενικά ρολόγια, έξι πηγές ρολογιού διαθέσιμες ανά λογική μπλοκ (LAB).
- Υποστήριξη για εξωτερική μνήμη, συμπεριλαμβανομένης της DDR SDRAM (133 MHz), FCRAM και (SDR) SDRAM.
- Υποστήριξη για πολλαπλούς πυρήνες, συμπεριλαμβανομένων Altera ® Megacore ® λειτουργίες και Altera Megafunctions Partners Program (AMPPSM).

Cyclone Device Features					
Feature	**EP1C3**	**EP1C4**	**EP1C6**	**EP1C12**	**EP1C20**
LEs	2,910	4,000	5,980	12,060	20,060
M4K Ram bloks (128 x 36 bits)	13	17	20	52	64
Total RAM bits	59,904	78,336	92,160	239,616	294,912
PLLs	1	2	2	2	2
Maximum user I/O pins *(1)*	104	301	185	249	301

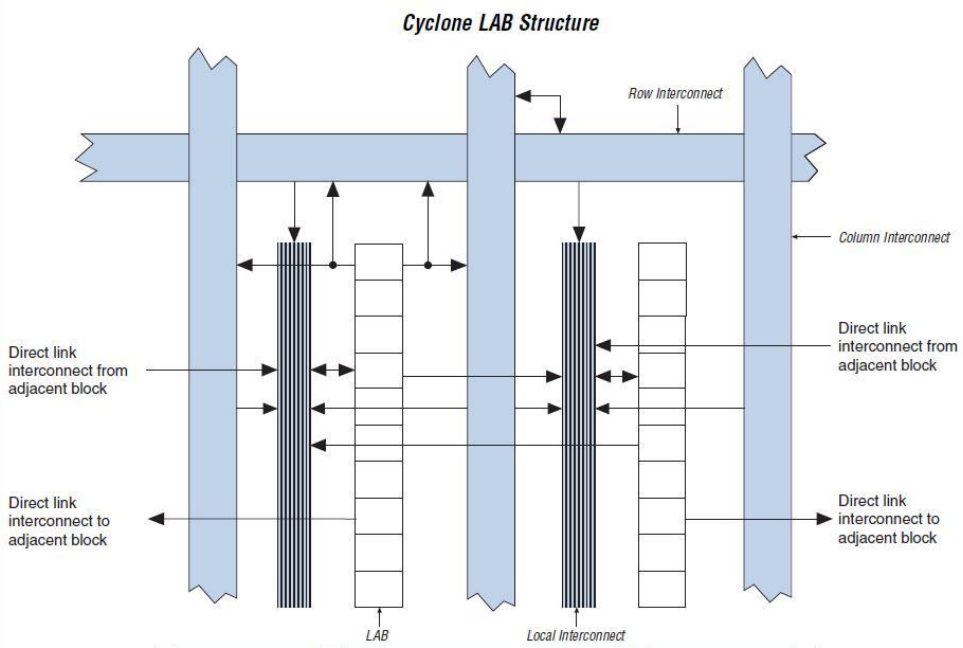
(1) Αυτή η παράμετρος περιλαμβάνει global clock ακροδέκτες

Πίνακας 3.1 [Alt1] Οικογένεια συσκευών FPGA Cyclone της εταιρίας Altera, γενικά χαρακτηριστικά συσκευής

Η γενική αρχιτεκτονική δομή μιας συσκευής Cyclone είναι αυτή που απεικονίζεται στην Εικόνα 3.12. Όπως μπορούμε να δούμε από τη συγκεκριμένη συσκευή *EP1C12,* παρέχονται δύο πεδία ρολογιού PLL και δύο στήλες με ενσωματωμένες μνήμες. Οι συσκευές Cyclone περιέχουν δισδιάστατες γραμμές και στήλες διασύνδεσης που βασίζονται στην αρχιτεκτονική προσαρμοσμένης λογικής. Οι στήλες και οι γραμμές διασυνδέονται με μία ποικιλία σημάτων διαφορετικών ταχυτήτων που χρησιμεύουν στη διασύνδεση των *συστοιχιών λογικής βαθμίδας* (*LAB, Logic Array Block*) και των ενσωματωμένων *βαθμίδων μνήμης* (*memory block*), Εικόνα 3.13.



Εικόνα 3.12 [Alt1] Γενική αρχιτεκτονική δομή συσκευών FPGA Cyclone και ιδιαιτέρα της συσκευής EP1C12



Εικόνα 3.13 [Alt1] Δομή καναλιών διασύνδεσης μεταξύ LABs και βαθμίδων μνημών (memory block) της οικογένειας FPGA Cyclone

Όπως βλέπουμε από την Εικόνα 3.13 υπάρχουν δυο διαφορετικά δίκτυα διασύνδεσης, το ένα είναι για εσωτερική διασύνδεση του κάθε LAB και ονομάζεται τοπική διασύνδεση (Local *Interconnect*) και το άλλο για διασύνδεση γειτονικών LAB και μνημών και αποτελούν μία γενική διασύνδεση (*Global Interconnect*). Η αρχιτεκτονική των συσκευών Cyclone τα ονομάζει *C4 Interconnect* και *R4 Interconnect,* για τις στήλες και τις γραμμές αντίστοιχα. Και τα δυο δίκτυα επικοινωνούν μεταξύ τους και υλοποιούν μαζί τα κανάλια δρομολόγησης (*Routing Channels*) της συσκευής.

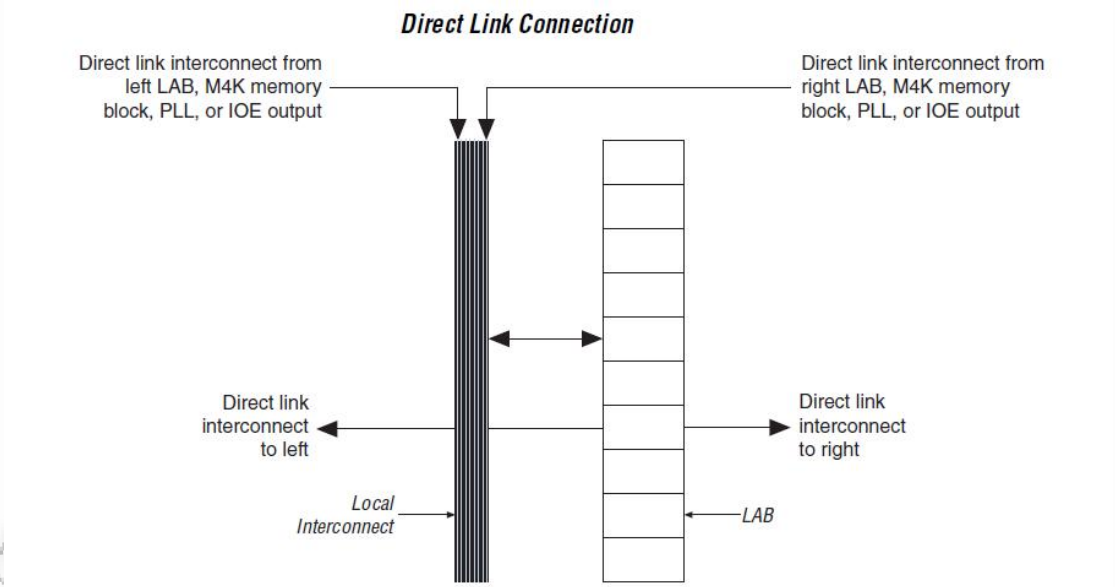
Στον Πίνακα 3.2, βλέπουμε τον αριθμό των χαρακτηριστικών κάθε συσκευής της οικογένειας Cyclone. Όπως μπορούμε να παρατηρήσουμε, όλες οι συσκευές έχουν δυο μονάδες PLL εκτός την

EP1C3 που έχει μία. Επιπροσθέτως, όταν οι βαθμίδες μνήμης είναι περισσότερες από 20 (τουλάχιστον οι διπλάσιες) η αρχιτεκτονική δομή περιλαμβάνει δυο στήλες μνημών και όχι μια.

Cyclone Device Features					
Device	M4K RAM		PLLs	LAB Columns	LAB Rows
	Columns	Blocks			
EP1C3	1	13	1	24	13
EP1C4	1	17	2	26	17
EP1C6	1	20	2	32	20
EP1C12	2	52	2	48	26
EP1C20	2	64	2	64	32

Πίνακας 3.2 [Alt1] Αριθμός χαρακτηριστικών κάθε συσκευής της οικογένειας FPGA Cyclone

Συστοιχία λογικών βαθμίδων (*LAB, Logic Array Block*), αποτελεί τη λογική βαθμίδα (LB) στις συσκευές Cyclone. Όπως προαναφέραμε το κάθε LAB οργανώνεται (γκρουπάρεται) μέσα σε γραμμές και στήλες από τα δίκτυα διασύνδεσης για να μπορεί να διασταυρώνεται με άλλα LAB της συσκευής. Κάθε LAB αποτελείται από 10 λογικά στοιχεία (LE) τα οποία μπορεί ο χρήστης να προγραμματίσει το καθένα ξεχωριστά έτσι ώστε να επιτελεί μια συγκεκριμένη λειτουργία. Επιπλέον έχει αλυσίδες κρατουμένου σε λογικά στοιχεία (*LE carry chains*), σήματα ελέγχου LAB (*control signal*), αλυσίδα κρατουμένου για πίνακες αναζήτησης LUT (L*ookup Table Chain*) και γραμμές καταχωρητών για αλυσίδες διασύνδεσης. Η τοπική διασύνδεση γίνεται μεταξύ των LE του κάθε LAB, μπορεί δηλαδή κάθε LUT να συνδέεται με κάποιο άλλο LUT ενός άλλου LE στο ίδιο LAB. Το ίδιο συμβαίνει και με τη διασύνδεση των καταχωρητών μεταξύ δύο ή περισσοτέρων LE ενός LAB. Κάθε LE μπορεί να επικοινωνήσει με άλλα 30 LE μέσω των τοπικών δίαυλων διασύνδεσης, Εικόνες 3.13 και 3.14. Η λογική διασύνδεση γίνεται μέσω του εργαλείου CAD της εταιρία Altera το οποίο είναι το Quartus II και θα το αναλύσουμε στο Κεφάλαιο 4.

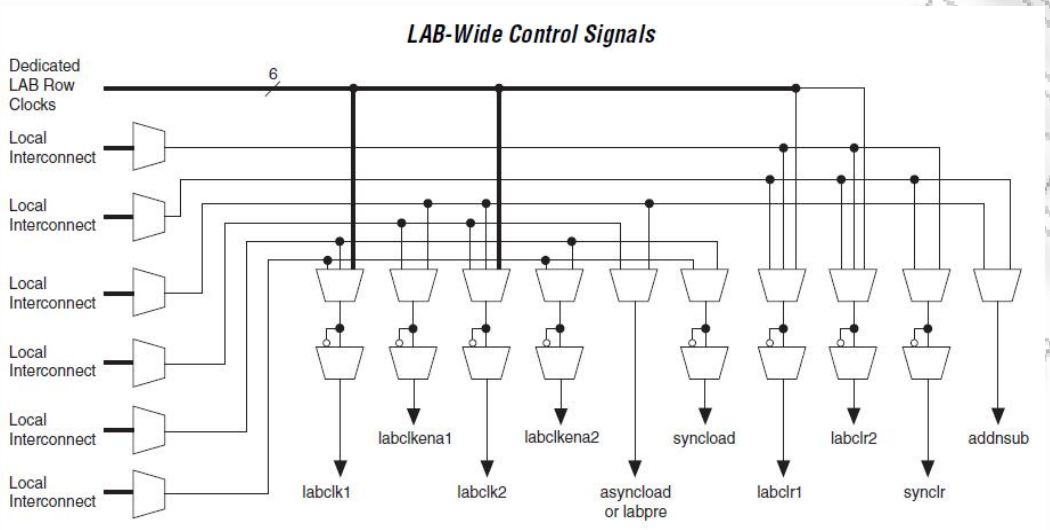


Εικόνα 3.14 [Alt1] Τοπική διασύνδεση λογικών στοιχείων LE μιας συστοιχίας λογικών βαθμίδων LAB της οικογένειας FPGA Cyclone

3.2.2. Σήματα αρχικοποίησης

Τα σήματα ελέγχου (*control signal*), σε ένα LAB είναι 10, δύο ρολόγια, δύο σήματα ενεργοποίησης, δύο ασύγχρονες εκκαθαρίσεις (*clear*), μία σύγχρονη εκκαθάριση, ασύγχρονη φόρτωση / αρχικοποίηση (*load/preset*), σύγχρονη φόρτωση και τέλος σήμα πρόσθεσης αφαίρεσης (*add / subtract*). Τα σήματα

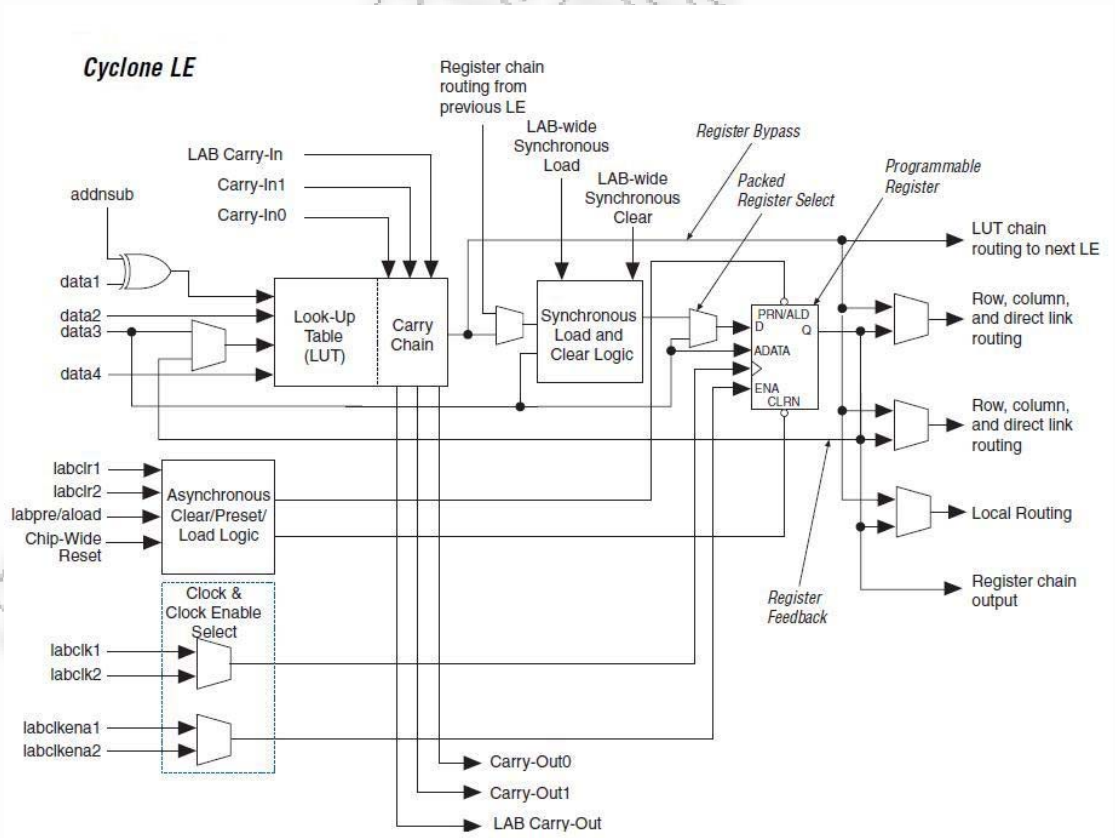
αυτά συνήθως χρησιμοποιούνται για την υλοποίηση μετρητών (*counters*) ή για οποιαδήποτε άλλη λειτουργία, Εικόνα 3.15.



Εικόνα 3.15 [Alt1] Τοπικά σήματα διασύνδεση ενός LAB της οικογένειας FPGA Cyclone

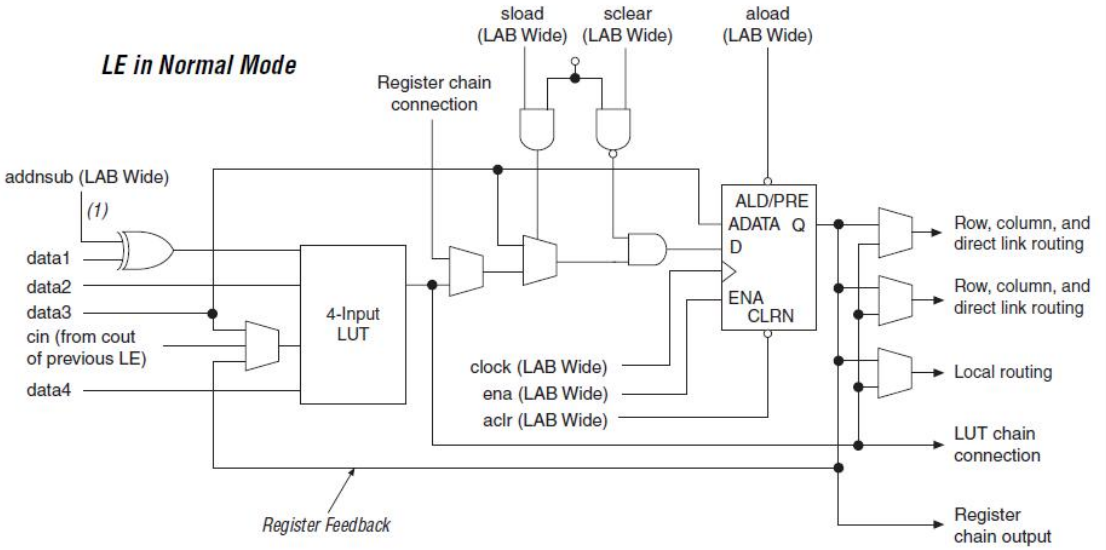
3.2.3. Λογικό στοιχείο

Κάθε λογικό στοιχείο LE περιέχει LUT τεσσάρων εισόδων (4-Input LUT), είναι η μικρότερη λογική μονάδα στην αρχιτεκτονική δομή της οικογένειας Cyclone, η οποία μπορεί να υλοποιήσει οποιαδήποτε λογική συνάρτηση - λειτουργία που παράγεται από είσοδο 4 μεταβλητών. Επιπρόσθετα όπως έχουμε αναφέρει, κάθε LE περιέχει προγραμματιζόμενους καταχωρητές, αλυσίδες κρατουμένου με δυνατότητα επιλογής κρατουμένου όπως και σήματα ελέγχου, Εικόνα 3.16. Το εύρος των LE σε μία συσκευή Cyclone κυμαίνεται από 2910 μέχρι 20060 στοιχεία.



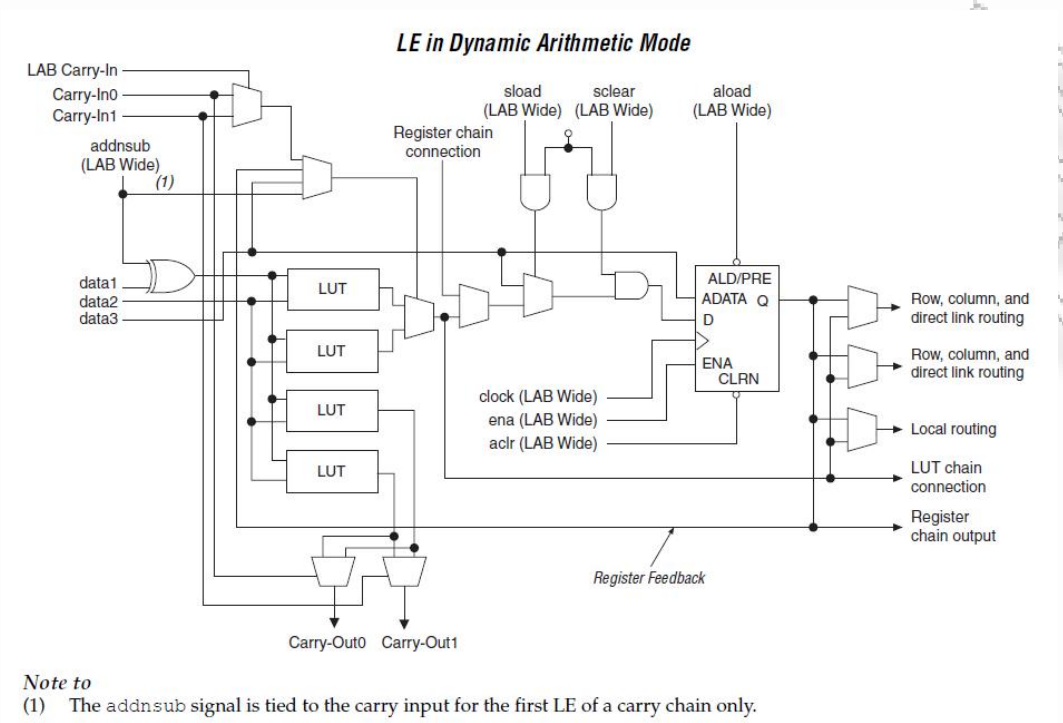
Εικόνα 3.16 [Alt1] Αρχιτεκτονική δομή λογικού στοιχείου LE της οικογένειας FPGA Cyclone

Μπορούν να διαμορφώσουν όλων των ειδών τα flip flop όπως D,T, SR και JK. Κάθε LE μπορεί να βρεθεί σε δύο καταστάσεις, στην *κανονική κατάσταση* – λειτουργία (*Normal Mode*) και στη *δυναμική αριθμητική κατάσταση – λειτουργία* (*Dynamic Arithmetic Mode*). Εάν απαιτείται μπορεί να δημιουργηθεί ειδικού σκοπού λειτουργία όπου θα διευκρινίζονται ποια LE θα επιτελέσουν ποια λειτουργία με σκοπό τη βέλτιστη της απόδοσης. Η κανονική κατάσταση είναι κατάλληλη για γενικές εφαρμογές και συνδυαστική λογική, κάθε LE έχει στη διάθεσή του ένα LUT τεσσάρων εισόδων (4-input LUT) για τη διευθέτηση αυτής της λειτουργίας, Εικόνα 3.17. Η δυναμική κατάσταση είναι κατάλληλη για την υλοποίηση αθροιστών, μετρητών, συσσωρευτών, συγκριτών κτλ και γενικά όπου χρειαζόμαστε αριθμητικές πράξεις. Αυτό επιτυγχάνεται με τη χρησιμοποίηση τεσσάρων LUT των δύο εισόδων (2-input LUT) σε κάθε LE, Εικόνα 3.18. Η επιτυχία αυτής της αρχιτεκτονικής οφείλεται στην πολύ καλή δομή που υπάρχει στις γεννήτριες διάδοσης κρατουμένου ανάμεσα στα LE σε κάθε LAB. Η υψηλές ταχύτητες που επιτυγχάνονται στη διάδοση των κρατουμένων βοηθούν στην υλοποίηση αριθμητικών βαθμίδων.



Note to
(1) This signal is only allowed in normal mode if the LE is at the end of an adder/subtractor chain.

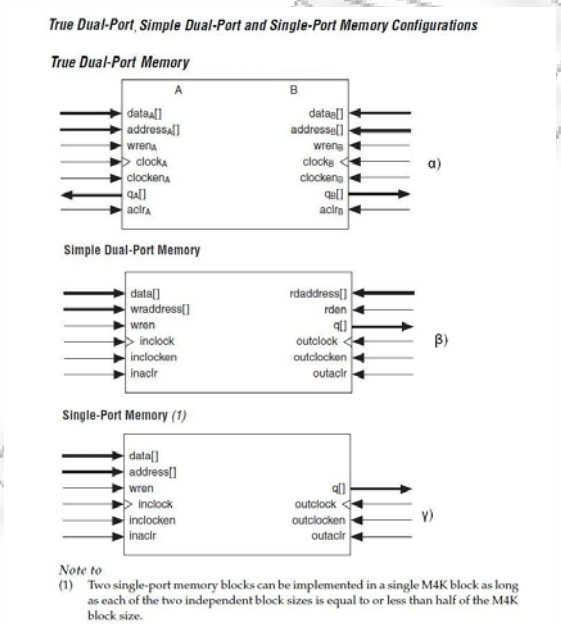
Εικόνα 3.17 [Alt1] Κανονική λειτουργία (Normal Mode) ενός λογικού στοιχείου LE της οικογένειας FPGA Cyclone



Εικόνα 3.18 [Alt1] Δυναμική αριθμητική λειτουργία (Dynamic Arithmetic Mode) ενός λογικού στοιχείου LE της οικογένειας FPGA Cyclone

3.2.4. Μνήμη RAM

Η μνήμη αποτελείται από M4K RAM block, φτάνοντας τα 4608bits μαζί με το bit ισοτιμίας. Οι μνήμες αυτές μπορούν να έχουν τις παρακάτω λειτουργίες, *αληθής διπλής θύρας* προσπέλασης (*True Dual Port*), *απλής διπλής θύρας* προσπέλασης (*Simple Dual Port*) και *μονής θύρας* προσπέλασης (*Single Port*), Εικόνα 3.19.



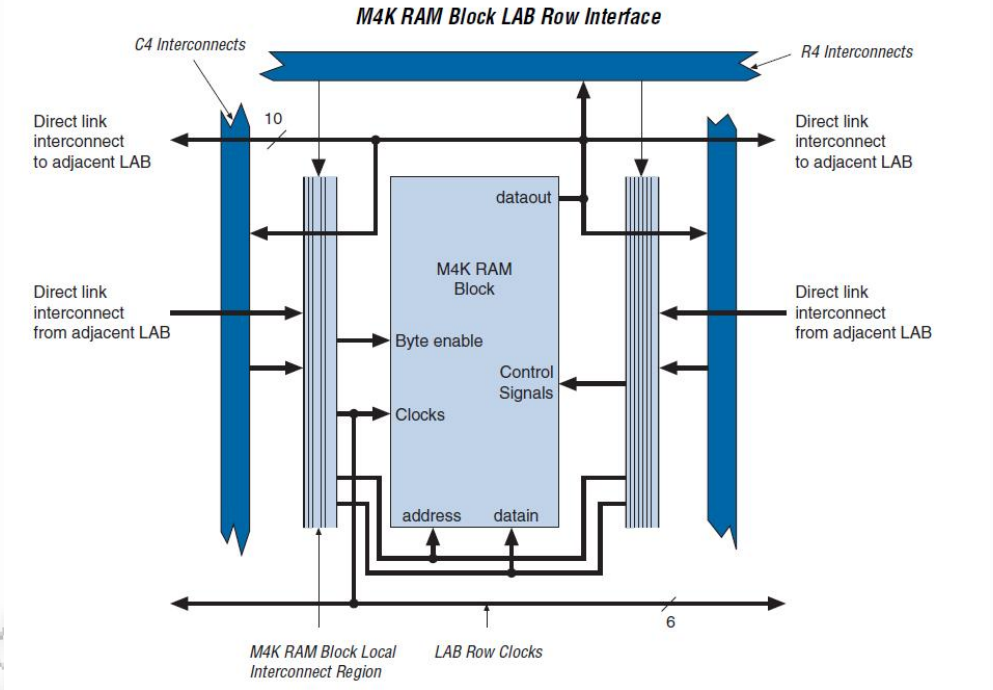
Εικόνα 3.19 [Alt1] Οι μνήμες True Dual Port, Simple Dual Port και Single Port της οικογένειας FPGA Cyclone

Αναλυτικότερα η μνήμη RAM μπορεί να χρησιμοποιηθεί ως:

- Στη *true dual port* μπορούμε να έχουμε ταυτόχρονη προσπέλαση σε δύο εγγραφές και δυο αναγνώσεις στον ίδιο παλμό ρολογιού. Ή μπορούμε να έχουμε ταυτόχρονη προσπέλαση σε μια εγγραφή και μια ανάγνωση, αλλά σε διαφορετικές συχνότητες ρολογιού.
- Στη *simple dual port* έχουμε ταυτόχρονη προσπέλαση εγγραφής και ανάγνωσης με τον ίδιο παλμό ρολογιού. Η μνήμη M4K έχει δύο διαφορετικά σήματα, ένα για την εγγραφή (write enable) και ένα για την ανάγνωση (read enable).
- Τέλος στη *single port* δεν υποστηρίζεται η ταυτόχρονη προσπέλαση εγγραφής και ανάγνωσης. Μπορεί επίσης τα M4K να διαιρεθούν στο μισό και να χρησιμοποιηθούν ως δύο ανεξάρτητα single port RAM block. Το Quartus II αυτόματα ενεργοποιεί αυτή τη λειτουργία αν διαπιστωθεί ότι η χρησιμοποιημένη μνήμη RAM είναι ελάχιστη.
- Καταχωρητής ολίσθησης (*shift register*), μπορούν να συνδυαστούν πολλά LE ώστε να διαμορφώσουν ένα μεγαλύτερο καταχωρητή ολίσθησης. Αυτό το χαρακτηριστικό χρησιμοποιείται από *DSP* (*Digital Signal Processing*) εφαρμογές. Αυτή η λειτουργία θα μπορούσε να υλοποιηθεί από τα LE αλλά δόθηκε μια επιπλέον εναλλακτική λύση με τη χρησιμοποίηση της μνήμης.
- Στη λειτουργία FIFO δεν υποστηρίζεται ταυτόχρονη εγγραφή και ανάγνωση.
- Ψηφία ισοτιμίας (*parity bit*), υποστηρίζει ένα προαιρετικό bit ισοτιμίας για κάθε byte δεδομένων. Από τα 4608 bit που είναι διαθέσιμα στη M4K μνήμη τα 512 είναι για την ισοτιμία. Τα bit ισοτιμίας μαζί με τη λογική που εφαρμόζεται σε κάθε LE, διευκολύνει τη μέθοδο ανεύρεσης σφαλμάτων και παρέχει σε αυτές την ακεραιότητα των δεδομένων. Τα επιπλέον bit ισοτιμίας μπορούν να χρησιμοποιηθούν και ως έξτρα bit για δεδομένα, δηλαδή θα μπορούσαν να είναι της μορφής 9, 18, 36 bit.

Επιπλέον υποστηρίζει *byte enable* λειτουργία, διόρθωσης σφαλμάτων και πληθώρα άλλες αναπτυγμένες λειτουργίες.

Στην Εικόνα 3.20 βλέπουμε πως η μνήμη RAM M4K συνδέεται με τα δίκτυα διασύνδεσης.

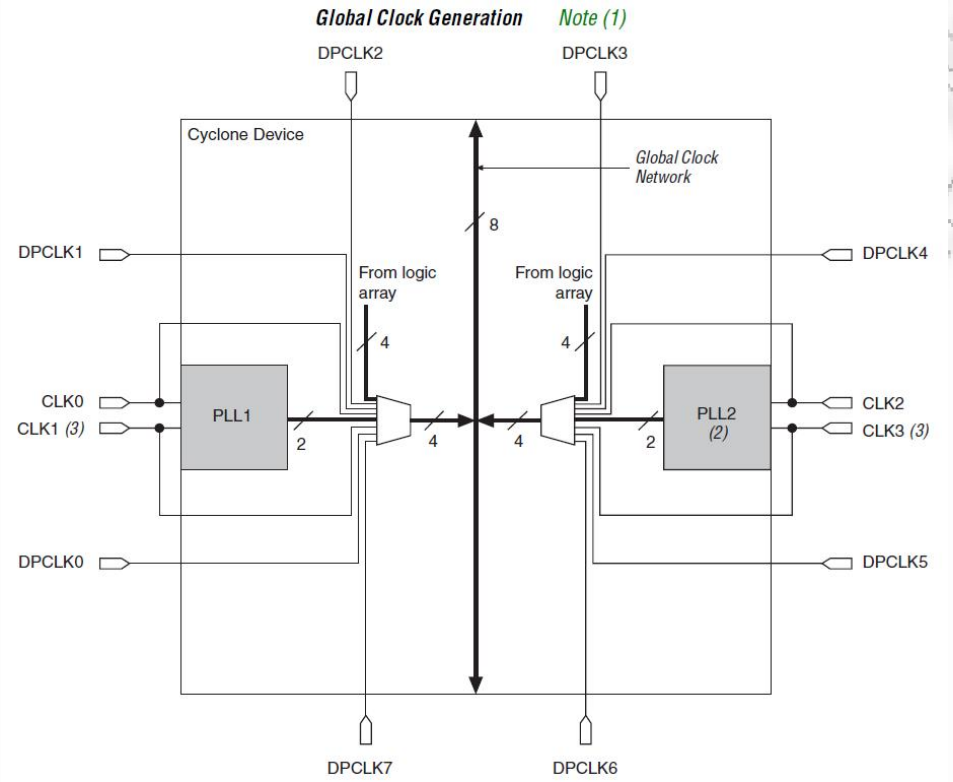


Εικόνα 3.20 [Alt1] Η μνήμη M4K και η διασύνδεση της με τα κανάλια δρομολόγησης της οικογένειας FPGA Cyclone

3.2.5. Ρολόι

Η αρχιτεκτονική δομή της οικογένειας Cyclone διαθέτει ένα διευθυντή ρολογιού (*Clock Manager*) με δύο PLL (*Phase Locked Loop*), εκτός από τη συσκευή EP1C3 που έχει ένα PLL, όπως είδαμε και από το Πίνακα 3.2. Ο διευθυντής ρολογιού παρέχει 8 ρολόγια γενικού σκοπού DPCLK[7..0] δύο για κάθε

πλευρά, χρησιμοποιεί ένα δίκτυο με οκτώ γραμμές που τροφοδοτούν τα ΟΚ της διάταξης σαν έξοδοι ρολογιού του ΟΚ, Εικόνα 3.21. Παρέχει επιπλέον 4 γραμμές εισόδου ρολογιού τα CLK[1..3] τα οποία είναι για εισαγωγή σημάτων ρολογιού. Οι γραμμές αυτές μπορούν να χρησιμοποιηθούν και σαν γραμμές σημάτων ελέγχου, όπως είναι το σήμα ενεργοποίησης ρολογιού αλλά και σημάτων σύγχρονης και ασύγχρονης εκκαθάρισης. Τα PLL παρέχουν επίσης πολλαπλασιασμό και μετατόπιση φάσης καθώς και εξωτερικές εξόδους για χρήση από υψηλών ταχυτήτων I/O. Το PLL συγχρονίζει το εξωτερικό ρολόι με το εσωτερικό. Το εσωτερικό ρολόι συνήθως λειτουργεί σε μεγαλύτερες συχνότητες από ότι το εξωτερικό. Αποτέλεσμα αυτού του συγχρονισμού είναι να ελαχιστοποιηθεί η *καθυστέρηση ρολογιού* (*clock delay*) καθώς και η *απόκλιση ρολογιού* (*clock skew*). Επιπλέον μειώνει ή προσαρμόζει τους χρόνους clock-to-out (t_{CO}) και set-up (t_{SU}).

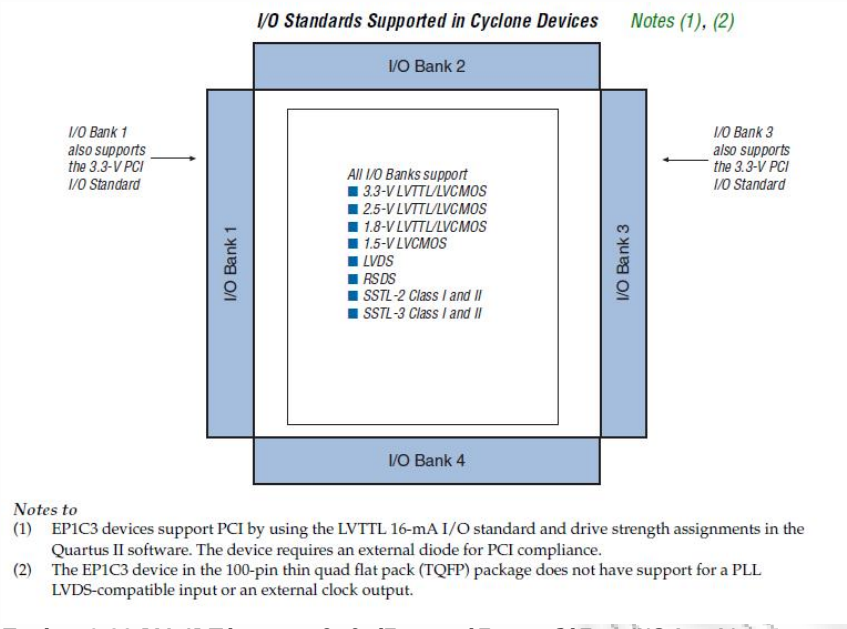


Notes to
(1) The EP1C3 device in the 100-pin TQFP package has five DPCLK pins (DPCLK2, DPCLK3, DPCLK4, DPCLK6, and DPCLK7).
(2) EP1C3 devices only contain one PLL (PLL 1).
(3) The EP1C3 device in the 100-pin TQFP package does not have dedicated clock pins CLK1 and CLK3.

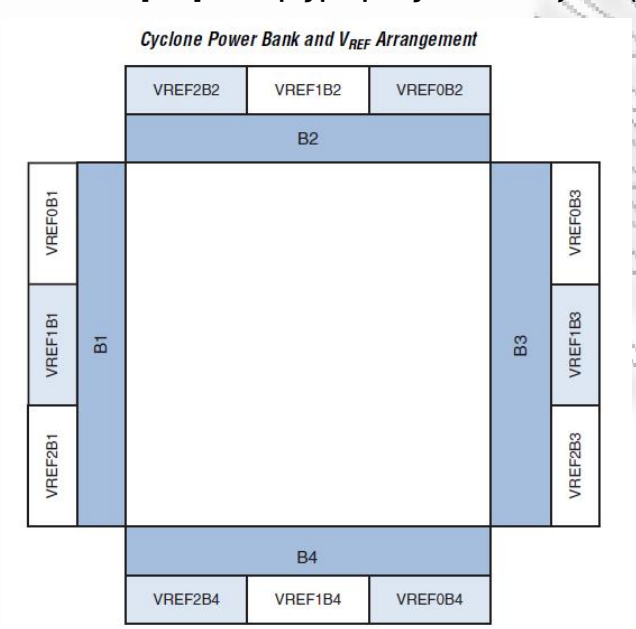
Εικόνα 3.21 [Alt1] Δίκτυο ρολογιού με δύο PLL (Phase Locked Loop) της οικογένειας FPGA Cyclone

3.2.6. Ακροδέκτες εισόδου εξόδου (I/O Bank) και η διασύνδεση τους

Οι ακροδέκτες εισόδου - εξόδου έχουν ομαδοποιηθεί σε τέσσερις *βαθμίδες εισόδου - εξόδου* (*I/O bank*) Εικόνα 3.22, όπου το καθένα έχει ένα ξεχωριστό ισχυρό δίαυλο επικοινωνίας, Εικόνες 3.24 και 3.25. Αυτό επιτρέπει στο σχεδιαστή να έχει ευελιξία στο ποιες εισόδους - εξόδους θα διαλέξει. Κάθε I/O bank αποτελείται από τρείς VRFF ακροδέκτες όπως φαίνεται στην Εικόνα 3.23, τα οποία μπορούν να χρησιμοποιηθούν ως κανονική ακροδέκτες εισόδου - εξόδου (I/O).

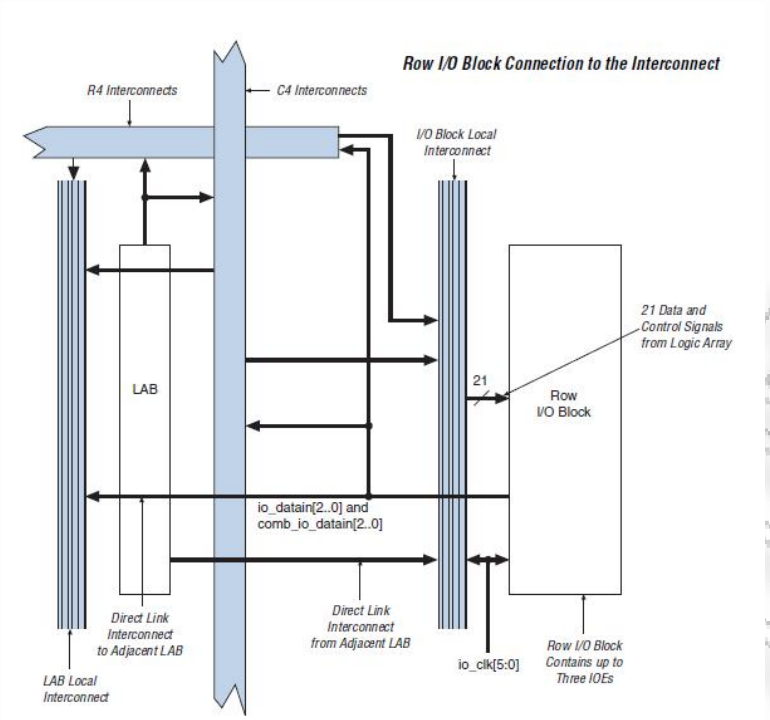


Εικόνα 3.22 [Alt1] Τέσσερις βαθμίδες εισόδου - εξόδου (I/O bank) της οικογένειας FPGA Cyclone

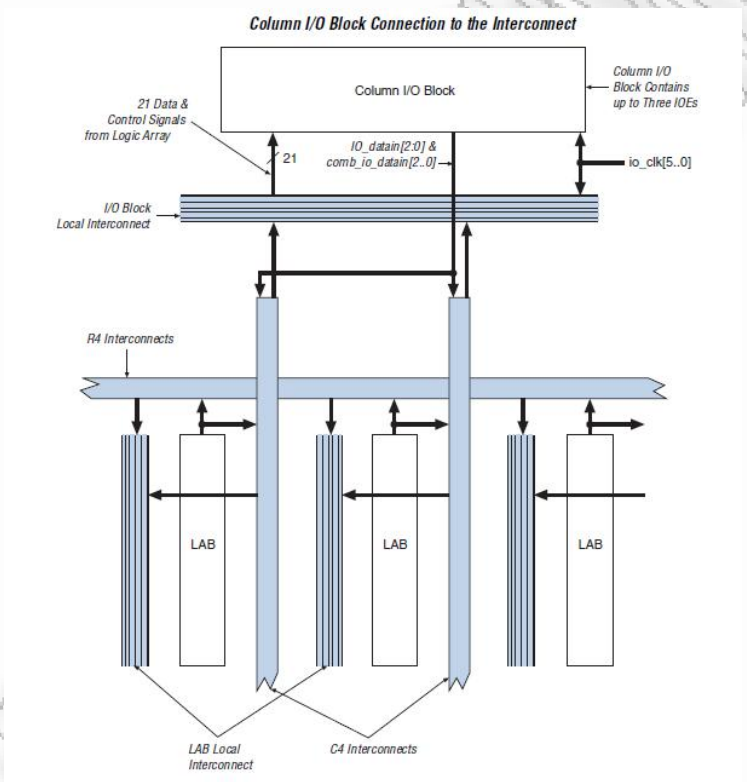


Εικόνα 3.23 [Alt1] VRFF ακροδέκτες στις βαθμίδες εισόδου – εξόδου (I/O bank) της οικογένειας FPGA Cyclone

Στις Εικόνες 3.24 και 3.25 βλέπουμε μια πιο λεπτομερή απεικόνιση της διασύνδεσης που γίνεται μεταξύ ακροδεκτών και υπόλοιπης αρχιτεκτονικής της οικογένειας Cyclone.



Εικόνα 3.24 [Alt1] Διασύνδεση γραμμής ακροδεκτών με την υπόλοιπη αρχιτεκτονική της οικογένειας FPGA Cyclone



Εικόνα 3.25 [Alt1] Διασύνδεση στήλης ακροδεκτών με την υπόλοιπη αρχιτεκτονική της οικογένειας FPGA Cyclone

Για περισσότερη λεπτομέρεια για όσα έχουμε αναφέρει παραπάνω σε σχέση με την αρχιτεκτονική δομή της οικογένειας FPGA Cyclone της εταιρίας Altera, θα σας παραπέμψουμε στο οδηγό χρήσης *Cyclone Device Handbook* [Alt1].

3.3. Αρχιτεκτονική Δομή Xilinx Spartan-3 FPGA

3.3.1. Αρχιτεκτονική και γενικά χαρακτηριστικά

Η δεύτερη αρχιτεκτονική δομή που θα δούμε είναι αυτή της οικογένειας συσκευών *Spartan-3* FPGA της εταιρίας *Xilinx*. Οι επεκτάσεις αυτής της οικογένειας είναι οι ομάδες συσκευών *Spartan-3A* και *Spartan-3E*. Εμείς θα εξετάσουμε τις συσκευές Spartan-3 άλλα σε κάποιους από τους πίνακες και τις εικόνες που θα δούμε, θα γίνεται αναφορά και σε κάποιες συσκευές των επεκτάσεων Spartan-3A και Spartan-3E. Κατασκευαστικά οι περισσότερες συσκευές Spartan σχετίζονται στενά με κάποιο άλλο προϊόν της Xilinx.

Κάποια σύντομα χαρακτηριστικά της συσκευής είναι τα εξής: τα σήματα της συσκευής κυμαίνονται ανάμεσα σε 1,14V και 3,465V, μονοί και διαφορικοί ακροδέκτες, 26 πρότυπα εισόδου-εξόδου (I/O), 8 βαθμίδες εισόδου-εξόδου (I/O bank), μέχρι 1916928 bits μνήμη RAM και μέχρι 74,880 λογικά στοιχεία κτλ. Επιπλέον χαρακτηριστικά φαίνονται παρακάτω. Στο Πίνακα 3.3 φαίνονται οι συσκευές που παρέχονται από τον κατασκευαστή για την οικογένεια Spartan-3.

Η συσκευές της οικογένειας Spartan-3 προσφέρουν τις ακόλουθες δυνατότητες:

- Χαμηλό κόστος, υψηλής απόδοσης λύσεις για δεδομένα υψηλού όγκου, εφαρμογές προσανατολισμένες προς τον καταναλωτή.
  - Πυκνότητα έως 74.880 λογικά στοιχεία
- Επιλογή I/O διασύνδεσης.
  - Μέχρι 633 I / O ακροδέκτες.
  - 622 + Mb/s ταχύτητα μεταφοράς δεδομένων ανά I/O.
  - 18 single - ended πρότυπα σήματα.
  - 8 διαφορετικά I/O πρότυπα, συμπεριλαμβανομένων LVDS, RSDS.
  - Σήματα που κυμαίνονται από 1.14V εώς 3.465V.
  - Υποστήριξη Double Data Rate (DDR).
  - Υποστήριξη DDR, DDR2 SDRAM έως και 333 Mbps.
- Λογικοί πόροι.
  - Συγκέντρωση των λογικών στοιχείων με δυνατότητα ολίσθησης καταχωρητή.
  - Ευρύς, γρήγορους πολυπλέκτες.
  - Γρήγορη πρόβλεψης κρατουμένου.
  - Αφιερωμένους 18 x 18 πολλαπλασιαστές.
  - JTAG λογική συμβατή με IEEE 1149.1/1532.
- Επιλογή RAM ιεραρχικής μνήμης.
  - Έως και 1.872 Kbits του συνόλου μπλοκ RAM.
  - Έως και 520 Kbits διανέμονται της συνολικής RAM.
- Ψηφιακό ρολόι διαχείρισης (έως και τέσσερις DCMs).
  - Εξάλειψη φαινομένου Clock Skew.
  - Σύνθεση συχνοτήτων.
  - Υψηλής ανάλυσης μετατόπιση φάσης.
- Οκτώ γενικές γραμμές ρολογιού για δρομολόγηση.
- Πλήρης υποστήριξη από την Xilinx ISE ® και WebPACK ™ λογισμικά συστήματα ανάπτυξης
- MicroBlaze ™ και PicoBlaze ™ επεξεργαστή, PCI ®, PCI Express ® PIPE Endpoint, και αλλους πυρήνες IP.

Summary of Spartan-3 FPGA Attributes

Device	System Gates	Equivalent Logic Cells	CLB Array (One CLB = Four Slices)			Distributed RAM Bits (K=1024)	Block RAM Bits (K=1024)	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs						
XC3S50	50K	1,728	16	12	192	12K	72K	4	2	124	56
XC3S200	200K	4,320	24	20	480	30K	216K	12	4	173	76

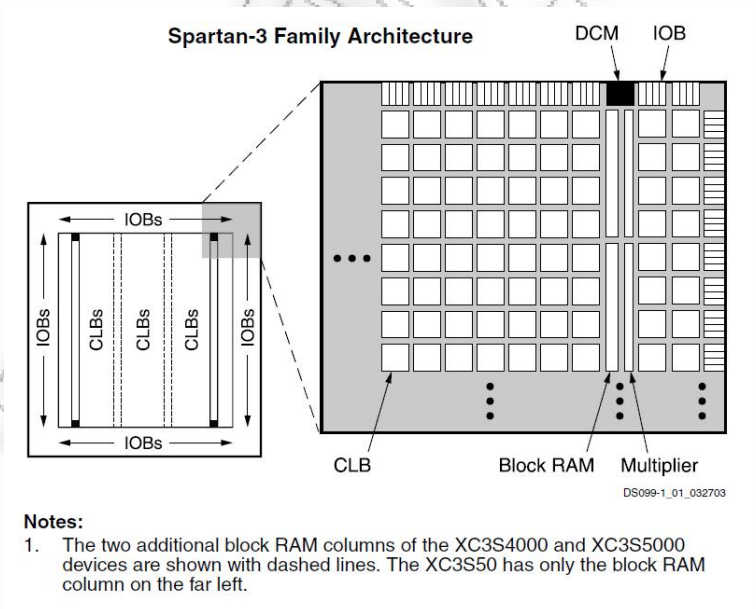
XC3S400	400K	8,064	32	28	896	56K	288K	16	4	264	116
XC3S1000	1M	17,280	48	40	1,920	120K	432K	24	4	391	175
XC3S1500	1.5M	29,952	64	52	3,328	208K	576K	32	4	487	221
XC3S2000	2M	46,080	80	64	5,120	320K	720K	40	4	565	270
XC3S4000	4M	62,208	96	72	6,912	432K	1,728K	96	4	633	300
XC3S5000	5M	74,880	104	80	8,320	520K	1,872K	104	4	633	300

Πίνακας 3.3 [Xln1] Οικογένεια συσκευών Spartan-3 και τα χαρακτηρίστηκα κάθε μιας

Η Spartan-3 οικογένεια αποτελείται από πέντε βασικά προγραμματιζόμενα λειτουργικά στοιχεία:

- Οι διαμορφωμένες λογικές βαθμίδες (*CLBs, Configurable Logic Blocks*) περιέχουν μνήμες RAM που βασίζονται σε πίνακες αναζήτησης (LUTs). Τα CLBs μπορούν να προγραμματιστούν έτσι ώστε να εκτελούν μια ευρεία ποικιλία από λογικές λειτουργίες, εκτελώντας τις λογικές συναρτήσεις που παρουσιάζονται, καθώς και για αποθήκευση δεδομένων.
- Λογικά σύνολα εισόδου - εξόδου (*IOBs, Input / Output Blocks*) ελέγχουν τη ροή των δεδομένων μεταξύ των ακροδεκτών εισόδου εξόδου (I/O pins) και την εσωτερική λογική της συσκευής. Κάθε IOB υποστηρίζει αμφίδρομη ροή δεδομένων συν τρισταθής λειτουργία, δηλαδή παρέχεται και μία προγραμματιζόμενη διάταξη τριών καταστάσεων (*Three - State*). Είναι διαθέσιμα επιπλέον είκοσι έξι διαφορετικά πρότυπα σήματος (I/O), μεταξύ των οποίων οκτώ διαφορικά πρότυπα υψηλής απόδοσης.
- Οι βαθμίδες μνήμης (*RAM Block*) παρέχουν αποθηκευτικό χώρο για τα δεδομένα, είναι της μορφής 18Kbit διπλής θύρας (Dual Port).
- Οι βαθμίδες πολλαπλασιαστών (*Multipliers Block*) δέχονται δύο 18bit δυαδικούς αριθμούς ως είσοδο και υπολογίζουν το αποτέλεσμα βγάζοντας ένα 36bit δυαδικό αριθμό ως έξοδο.
- Ο διευθυντής ρολογιού (DCM, Digital Clock Manager) παρέχει αυτόματη βαθμονόμηση, πλήρες ψηφιακές λύσεις για διανομή, καθυστέρηση, πολλαπλασιασμό, διαίρεση καθώς και μετατόπιση φάσης, του σήματος ρολογιού.

Η γενική αρχιτεκτονική που ακλουθείται είναι της μορφής της Εικόνας 3.26. Τα παραπάνω στοιχεία οργανώνονται όπως μπορούμε να διακρίνουμε σε, διαμορφωμένες λογικές βαθμίδες (CLB), βαθμίδες μνημών RAM, βαθμίδες πολλαπλασιαστών (Multipliers) καθώς και βαθμίδες εισόδου εξόδου (IOB). Η οικογένεια Spartan-3 διαθέτει ένα πλούσιο δίκτυο ιχνών και διακοπτών για τη διασύνδεση των πέντε παραπάνω λειτουργικών στοιχείων. Κάθε λειτουργικό στοιχείο έχει ένα σχετικό πίνακα διακοπτών που επιτρέπει πολλαπλές συνδέσεις στο δίκτυο δρομολόγησης.

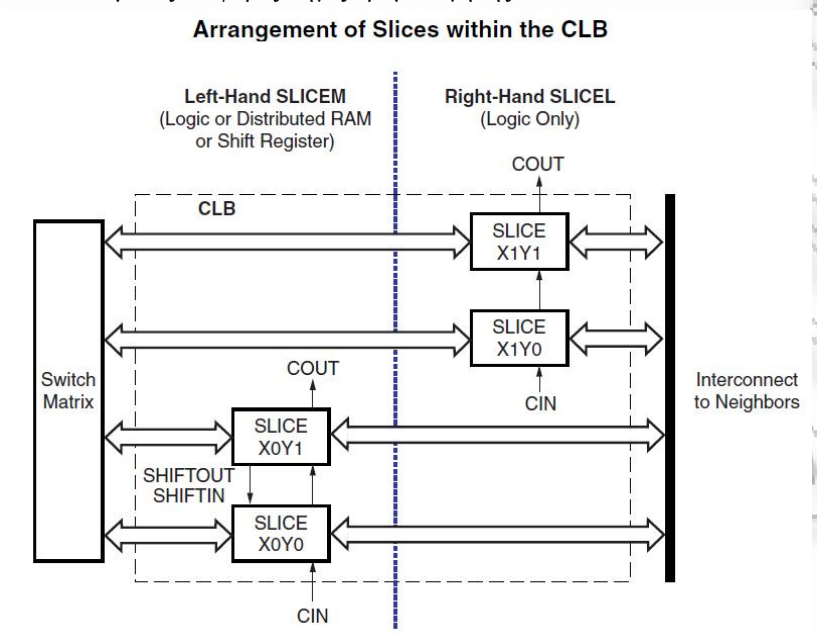


Notes:
1. The two additional block RAM columns of the XC3S4000 and XC3S5000 devices are shown with dashed lines. The XC3S50 has only the block RAM column on the far left.

Εικόνα 3.26 [Xln1] Γενική αρχιτεκτονική δομή των συσκευών Spartan-3

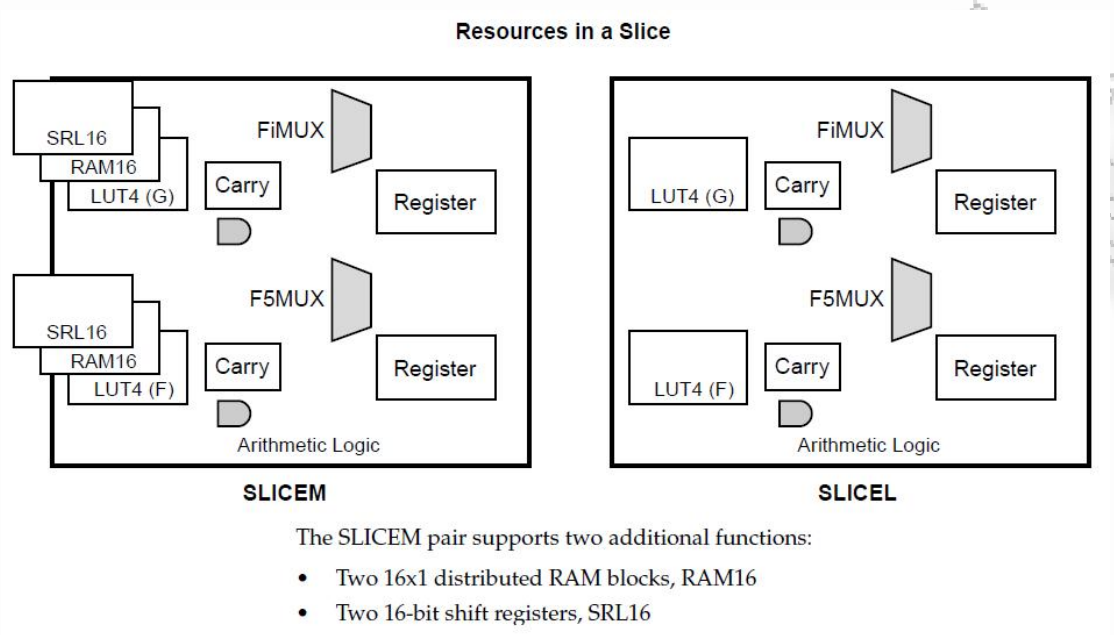
3.3.2. Διαμορφωμένες λογικές βαθμίδες (CLBs)

Οι διαμορφωμένες λογικές βαθμίδες (CLBs) αποτελούν την κύρια πηγή λογικής για την εφαρμογή σύγχρονων και συνδυαστικών κυκλωμάτων. Κάθε CLB περιέχει τέσσερα λογικά κυκλώματα τα οποία ονομάζονται ”φέτες” (Slices), Εικόνα 3.27. Το κάθε CLB περιέχει 2 *SLICEL* τα οποία μπορούν να υλοποιήσουν μόνο λογική και 2 *SLICEM* τα οποία μπορούν να υλοποιήσουν λογική, μνήμη και ολίσθηση. Στην Εικόνα 3.27 βλέπουμε επίσης τη μήτρα μεταγωγής (*Switch Matrix*) με την οποία το CLB συνδέεται με τις διάφορες πηγές δρομολόγησης.



Εικόνα 3.27 [Xln1] Δομή μιας διαμορφωμένης λογικής βαθμίδας (CLBs) της οικογένειας FPGA Spartan-3

Κάθε Slice περιέχει δύο πίνακες αναζήτησης (LUT) των 4 εισόδων (4-Input LUT) για την εφαρμογή οποιασδήποτε λογικής συνάρτησης και δύο ειδικά αποθηκευτικά στοιχεία που μπορούν να χρησιμοποιηθούν ως flip flops ή μανδαλωτές (Latches). Τα LUT είναι δυνατόν να χρησιμοποιηθούν ως 16×1 μνήμη (RAM16) ή ως 16-bit καταχωρητής ολίσθησης (SRL16, Shift Register), αλλά μόνο στα SLICEM και όχι στα SLICEL. Επιπλέον οι πολυπλέκτες και η λογική κρατουμένου (*Carry Logic*), απλοποιούν τις αριθμητικές λειτουργίες. Η λογική διαμόρφωση αντιστοιχίζεται αυτόματα στα περισσότερα από τα γενικής χρήσης σχεδία λογικής. Στην Εικόνα 3.28, βλέπουμε πως μπορούν να διαμορφωθούν τα SLICEM και SLICEL και παράλληλα πώς μπορούν αυτά να χρησιμοποιηθούν σε ένα CLB.

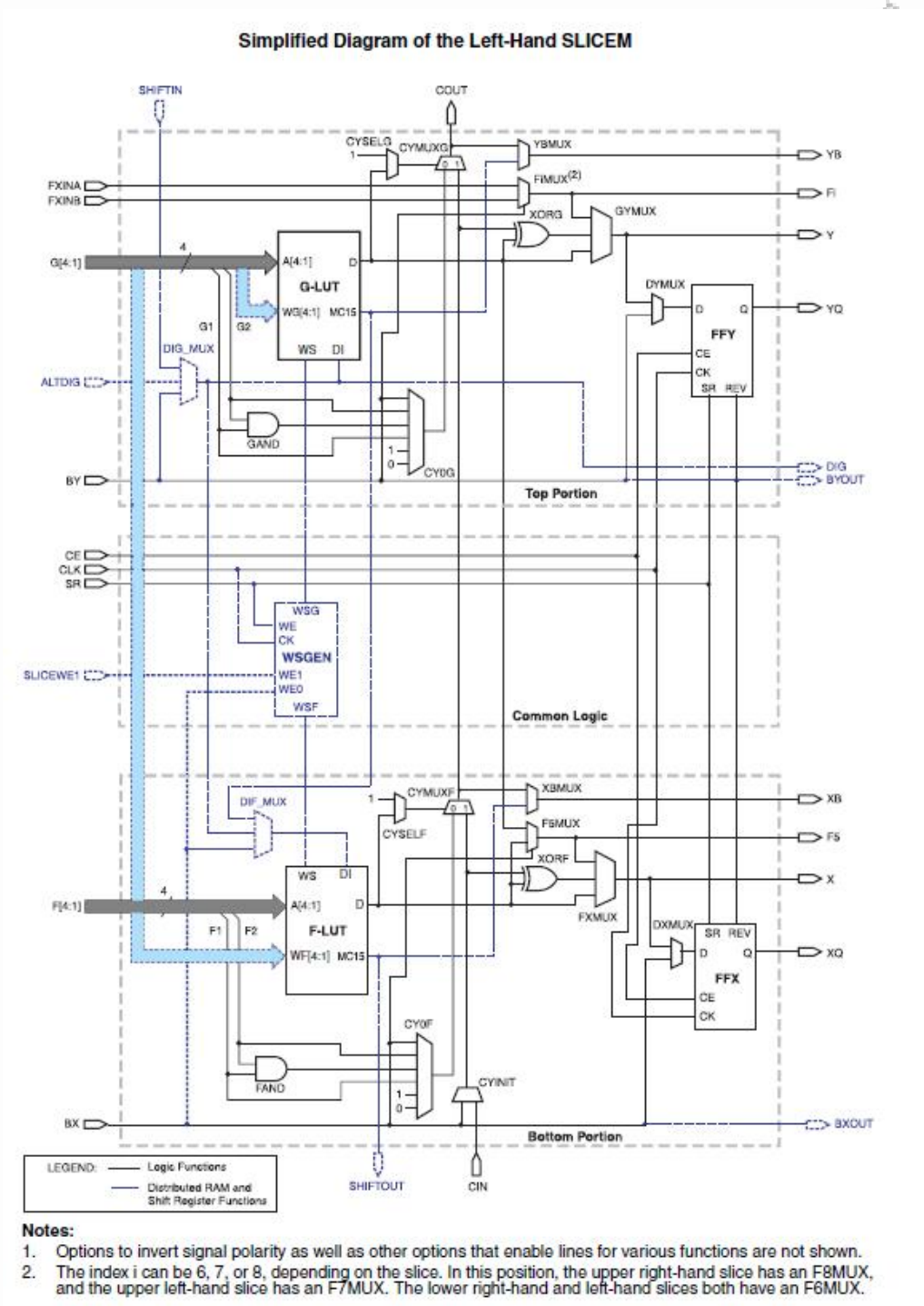


Εικόνα 3.28 [XIn2] Δομή ενός SLICEM και ενός SLICEL μέσα σε ένα CLB, της οικογένειας FPGA Spartan-3

Στην Εικόνα 3.29, βλέπουμε σε μεγαλύτερη λεπτομέρεια τη δομή ενός SLICEM, η δομή ενός SLICEL είναι πανομοιότυπη απλά δεν χρησιμεύουν τα LUT ως μνήμη ή ως καταχωρητής ολίσθησης. Παρέχονται επίσης, γρήγορη αριθμητική λογική, λογική πολλαπλασιαστή, λογική πολυπλέκτη, set ή reset, κανονικές ή αντεστραμμένες είσοδοι. Η μνήμη RAM που μπορεί να υλοποιηθεί από ένα LUT σε ένα SLICEM, μπορεί να είναι μονής ή διπλής θύρας (single and dual port RAM) και συνδέοντας πολλά LUTs αυξάνεται το μέγεθός της. Η εγγραφή είναι μόνο σύγχρονη και η ανάγνωση μπορεί να είναι σύγχρονη ή ασύγχρονη. Μπορούμε να έχουμε έναν μεγάλο καταχωρητής ολίσθησης συνδέοντας πολλά LUTs. Το κάθε CLB περιέχει έως και 64 bits μίας θύρας μνήμη RAM ή 32 bits της διπλής θύρας μνήμη RAM. Αυτές η μνήμες RAM είναι κατανεμημένα σε όλη την αρχιτεκτονική της συσκευής FPGA και ονομάζεται κοινώς "*κατανεμημένη μνήμη RAM*", ώστε να διαχωρίζεται από το 18Kbit μνήμης RAM (RAM Block). Η κατανεμημένη μνήμη RAM είναι επίσης γνωστή και ως *LUT RAM*.

3.3.3. Λογικό στοιχείο

Το λογικό στοιχειό που έχουμε δεις έως τώρα, στη Xilinx αποτελεί το λογικό κύτταρο “Logic Cell”, το οποίο δεν είναι άλλο από τη λογική που περιβάλει ένα LUT. Ο συνδυασμός ενός LUT και του αποθηκευτικού στοιχείο είναι γνωστό ως «Logic Cell". Τα πρόσθετα χαρακτηριστικά σε ένα slice, όπως είναι ο πολυπλέκτης, η λογική κρατουμένου, αλλά και οι αριθμητικές πύλες προσθέτουν και αυξάνουν τη χωρητικότητα ενός slice. Αν δεν υπήρχαν αυτές οι λογικές μονάδες θα χρειαζόταν επιπλέον πρόσθεση ενός ακόμη LUT.

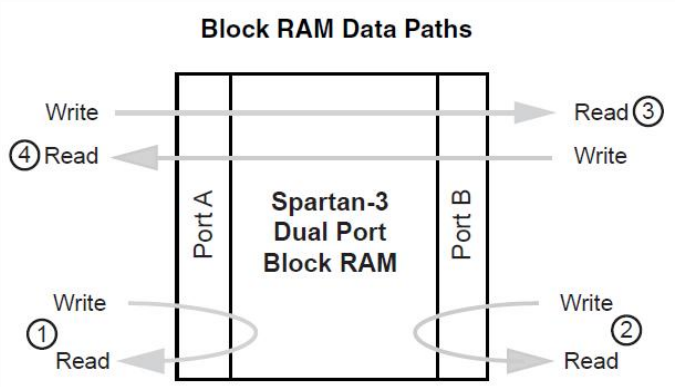


Notes:
1. Options to invert signal polarity as well as other options that enable lines for various functions are not shown.
2. The index i can be 6, 7, or 8, depending on the slice. In this position, the upper right-hand slice has an F8MUX, and the upper left-hand slice has an F7MUX. The lower right-hand and left-hand slices both have an F6MUX.

Εικόνα 3.29 [Xln1] Λεπτομερέστερη δομή ενός SLICEM, της οικογένειας FPGA Spartan-3

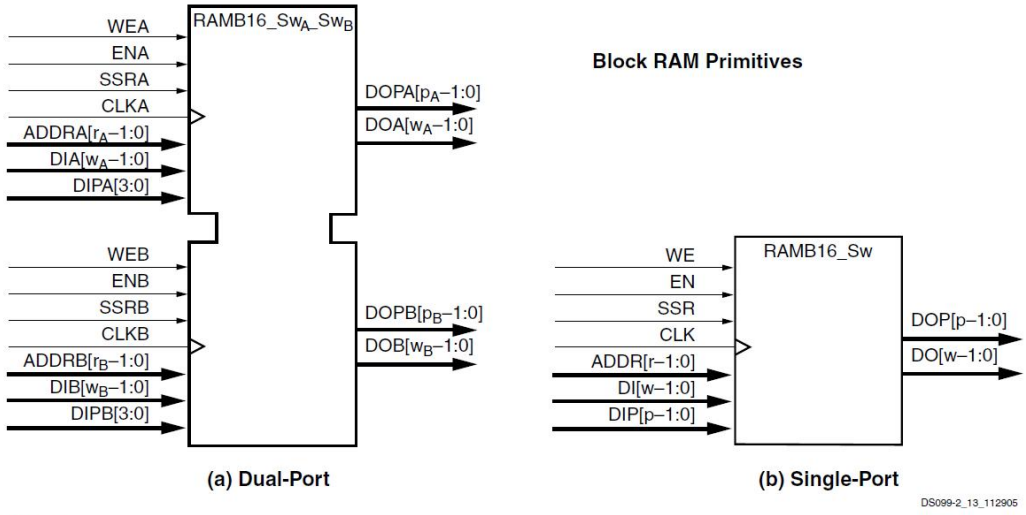
3.3.4. Μνήμη RAM

Οι ενσωματωμένες βαθμίδες μνήμης (RAM Block), οργανώνονται σε μία διαμορφωμένη μνήμη των 18Kbit. Υπάρχουν ειδικές περιπτώσεις όπου κάποιες μνήμες, ανάλογα με τη λειτουργία που εκτελούν, διαθέτουν τα 16K για τα δεδομένα και τα 2K για την ισοτιμία. Η μνήμη RAM αποθηκεύει σχετικά μεγαλύτερες ποσότητες δεδομένων πιο αποτελεσματικά από την κατανεμημένη μνήμη RAM που περιγράψαμε παραπάνω. Είναι ιδανική για υλοποίηση διαφορετικών τύπων μνημών όπως διπλής και μονής θύρας. Στην Εικόνα 3.30, βλέπουμε τη χρήση της μνήμης RAM ως διπλής θύρας (Dual Port) προσπέλασης.



Εικόνα 3.30 [XIn1] Μνήμη RAM με χρήση διπλής θύρας (Dual Port), της οικογένειας FPGA Spartan-3

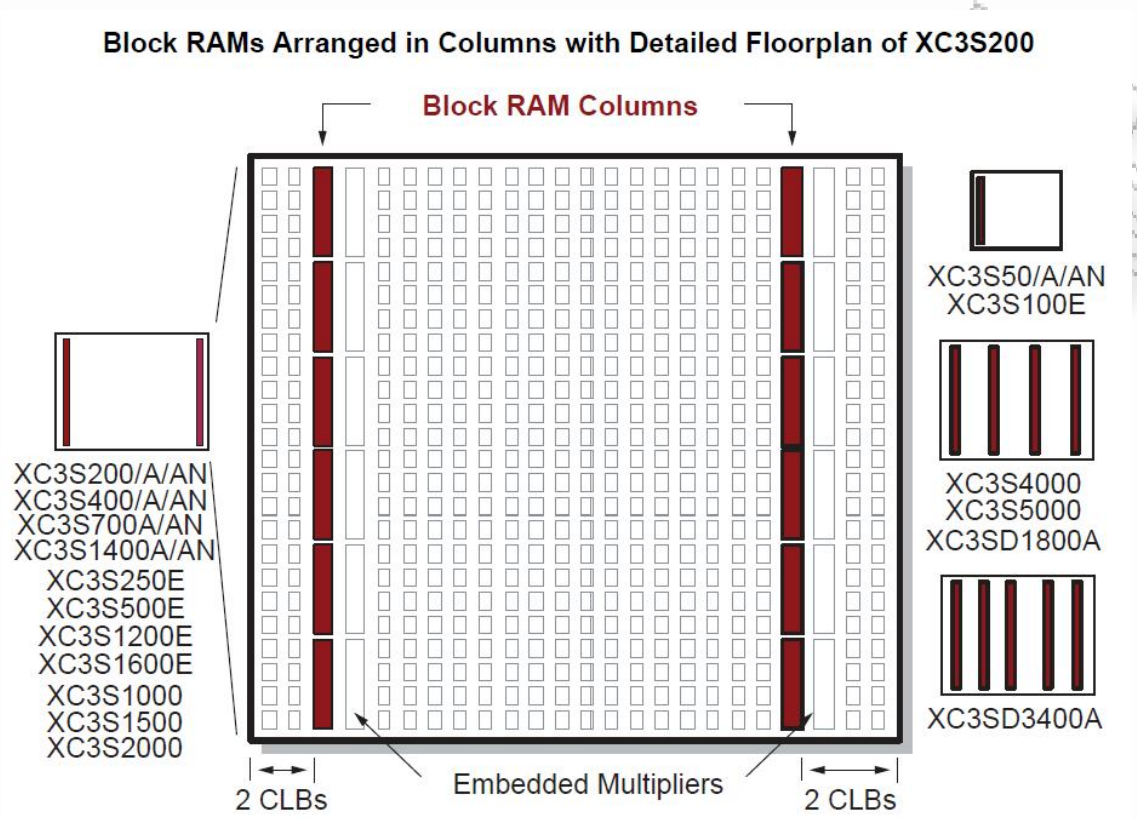
Στην Εικόνα 3.31, βλέπουμε μια λεπτομερέστερη απεικόνιση των σημάτων ελέγχου και διασύνδεσης με χρήση λειτουργίας διπλής θύρας και μονής θύρας. Όπως μπορείτε να παρατηρήσετε υπάρχει ανεξάρτητη πρόσβαση και διευθέτηση των θυρών. Μια διπλής θύρας RAM μπορεί να διαχωριστεί και να λειτουργεί ως δυο ανεξάρτητες μνήμες μονής θύρας, το μειονέκτημα με αυτή την επιλογή είναι ότι η μνήμη πέφτει στο μισό της μέγεθος. Η κάθε θύρα συγχρονίζεται με το δικό της ανεξάρτητο ρολόι. Επιπλέον αυτές οι μνήμες μπορούν αν αρχικοποιηθούν και να χρησιμοποιηθούν ως ROM.



Notes:
1. w_A and w_B are integers representing the total data path width (i.e., data bits plus parity bits) at ports A and B, respectively.
2. p_A and p_B are integers that indicate the number of data path lines serving as parity bits.
3. r_A and r_B are integers representing the address bus width at ports A and B, respectively.
4. The control signals CLK, WE, EN, and SSR on both ports have the option of inverted polarity.

Εικόνα 3.31 [XIn1] Τα σήματα ελέγχου και διασύνδεσης της μνήμης RAM με χρήση διπλής και μονής θύρας (Dual - Single Port), της οικογένειας FPGA Spartan-3

Στην Εικόνα 3.32, βλέπετε τη δομή που έχει η μνήμη RAM σε κάθε μία από τις συσκευές της οικογένειας συσκευών FPGA Spartan-3, αλλά και των επεκτάσεών της Spartan-3A και Spartan-3E.



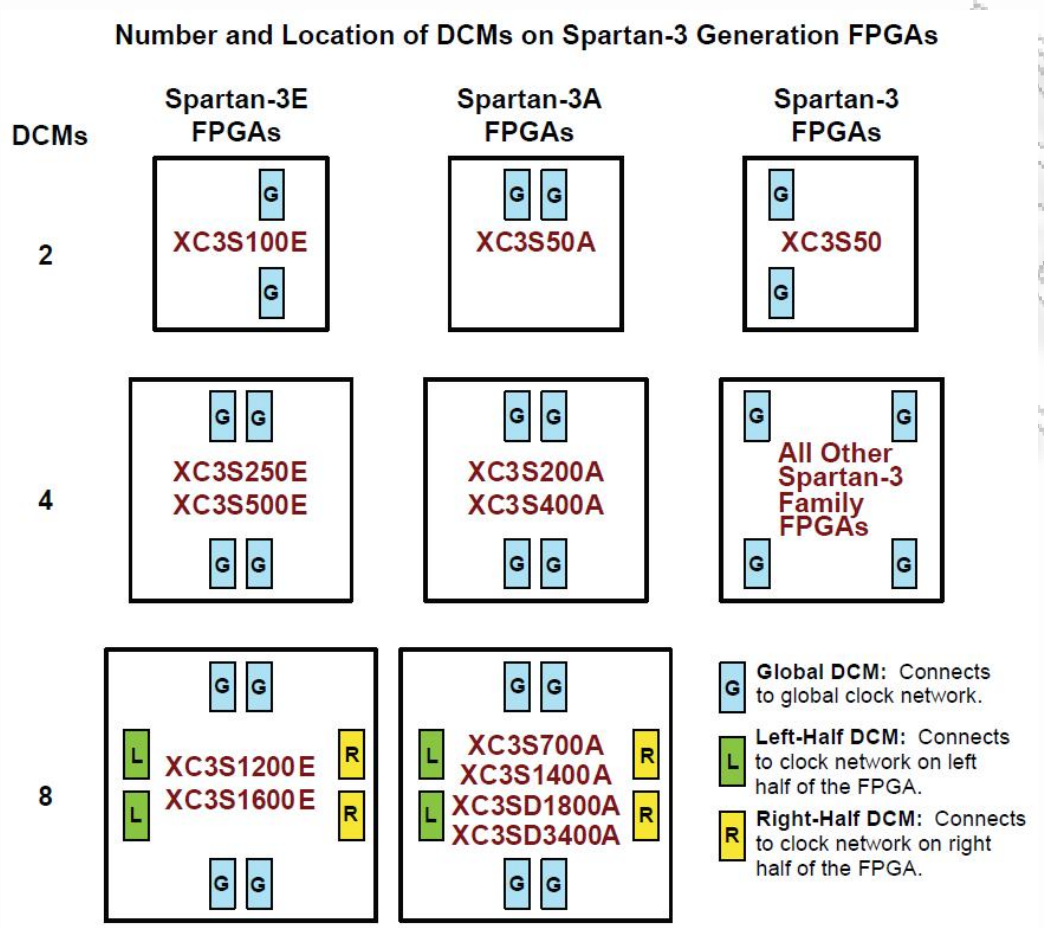
Εικόνα 3.32 [Xln2] Δομή μνήμης RAM για κάθε μία από τις συσκευές της οικογένειας FPGA Spartan-3 και των επεκτάσεων της, Spartan-3A και Spartan-3E

Από όλα τα παραπάνω χαρακτηριστικά που είδαμε βλέπουμε ότι χρησιμοποιώντας διάφορες ρυθμίσεις και επιλογές, μπορούμε στις επιλεγόμενες μνήμες RAM να δημιουργήσουμε RAM, ROM, FIFOs, μεγάλους πίνακες αναζήτησης LUT, καταχωρητές ολίσθησης και μετατρέποντας το πλάτος των δεδομένων αποθήκευσης να υποστηρίξουμε μία ποικιλία εισαγωγής δεδομένων ανεξάρτητα του μεγέθους τους.

3.3.5. Ρολόι

Κάθε Spartan-3 FPGA συσκευή παρέχει οκτώ υψηλής ταχύτητας πηγές παραγωγής σημάτων ρολογιού, με χαμηλή απόκλιση (low skew) για να βελτιστοποιήσουν την απόδοση. Οι πηγές αυτές χρησιμοποιούνται αυτόματα από τα εργαλεία Xilinx. Ακόμη και αν ο ρυθμός του ρολογιού είναι σχετικά αργός, εξακολουθεί να είναι σημαντικό η χρησιμοποίηση των πηγών αυτών για τη σωστή δρομολόγηση. Με αυτό το τρόπο πετυχαίνουμε να εξαλειφθεί ο κάθε πιθανός κινδύνους που μπορεί να υπάρξει σε μία συσκευή όπως η έλλειψη σήματος ρολογιού. Είναι πολύ σημαντικό να καταλάβουμε πώς πρέπει να καθορίσουμε ή καλύτερα πως να επωφεληθούν από αυτούς τους πόρους.

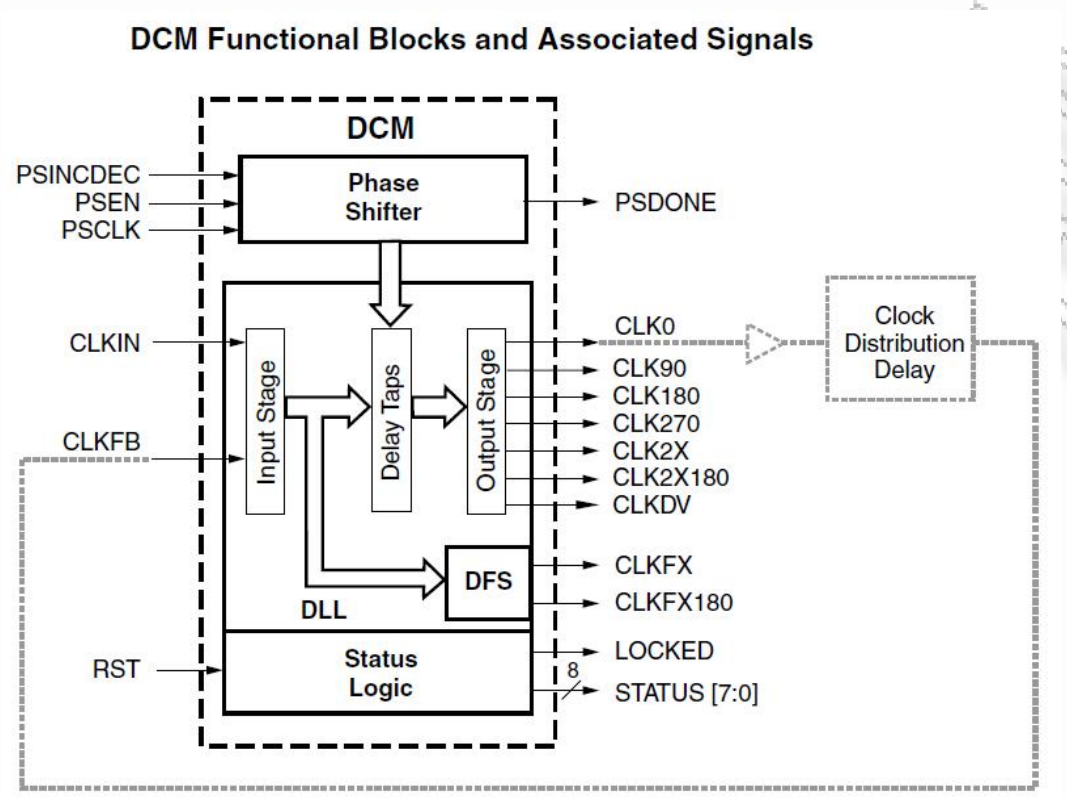
Κάθε συσκευή Spartan-3 έχει περισσότερα από δυο διευθυντές ρολογιού (DCM). Μία λεπτομερή αναπαράσταση της δομής των διευθυντών ρολογιού σε κάθε συσκευή της οικογένειας Spartan-3 και των επεκτάσεων της Spartan-3A και Spartan-3E, φαίνεται στην Εικόνα 3.33.



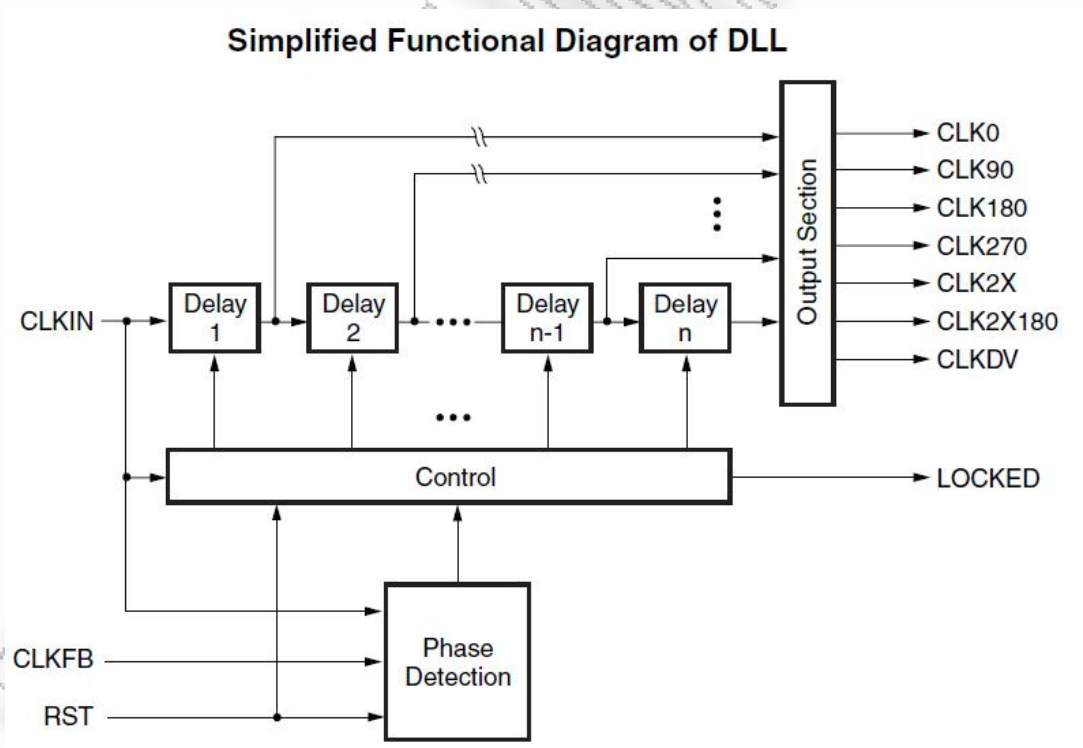
Εικόνα 3.33 [Xln2] Δομή των διευθυντών ρολογιού (DCM) για κάθε μία από τις συσκευές της οικογένειας FPGA Spartan-3 και των επεκτάσεων της, Spartan-3A και Spartan-3E

Ο κάθε διευθυντής ρολογιού παρέχει, περιορισμό της απόκλισης ρολογιού (clock skew), ολίσθηση φάσης (DPS, Digital Phase Shift), σύνθεση συχνοτήτων (DFS, Digital Frequency Synthesis), η δομή ενός DCM φαίνεται στην Εικόνα 3.34. Για να επιτευχθούν τα παραπάνω ο DCM χρησιμοποιεί ένα *βρόχο σταθερής καθυστέρησης* (*DLL, Delay Locked Loop*) το οποίο κλειδώνει στην καθυστέρηση. Είναι ένα πλήρες ψηφιακό σύστημα ελέγχου που χρησιμοποιεί την ανατροφοδότηση του κύριου ρολογιού για να διατηρήσει τα χαρακτηριστικά του σήματος ρολογιού σε υψηλό βαθμό ακρίβειας, Εικόνα 3.35. Κάποια κύρια χαρακτηριστικά του DLL είναι:

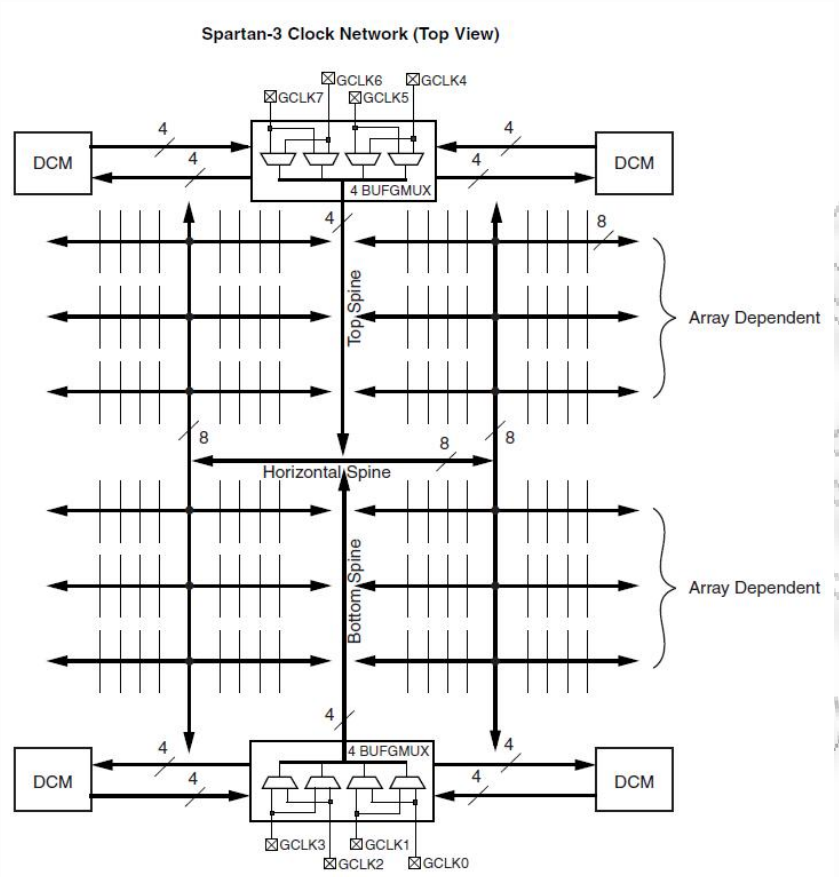
- Εισάγει καθυστέρηση στο δίκτυο του ρολογιού έως ότου η ανοδική ακμή του ρολογιού εισόδου (*CLKIN*) είναι σε φάση με την ανοδική ακμή του ρολογιού ανάδρασης (*CLKFB*). Δηλαδή το εξωτερικό ρολόι και το εσωτερικό να είναι «ευθυγραμμισμένα», Εικόνα 3.34.
- Με ένα καλοσχεδιασμένο δίκτυο διανομής του ρολογιού, οι ακμές του ρολογιού «φτάνουν» ταυτόχρονα σε όλα τα σημεία της συσκευής μαζί με την άφιξη της ακμής του ρολογιού εισόδου του DLL, Εικόνα 3.36.
- Χρήση ρολογιών με διαφορετικές φάσεις 0^o, 90^o, 180^o και 270^o ώστε να αυξηθεί η απόδοση, είναι κατάλληλο για διασύνδεση εξωτερικών μνημών.
- Πολλαπλασιασμό συχνότητας $\times 2$, διαίρεση συχνότητας από 1.5 έως 16.



Εικόνα 3.34 [XIn1] Δομή ενός διευθυντή ρολογιού (DCM), της οικογένειας FPGA Spartan-3



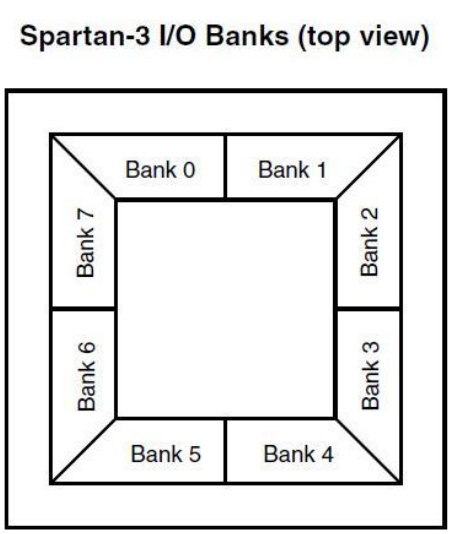
Εικόνα 3.35 [XIn1] Δομή ενός βρόχου σταθερής καθυστέρησης (DLL, Delay Locked Loop), της οικογένειας FPGA Spartan-3



Εικόνα 3.36 [XIn1] Δομή δικτύου διανομής του ρολογιού της οικογένειας FPGA Spartan-3

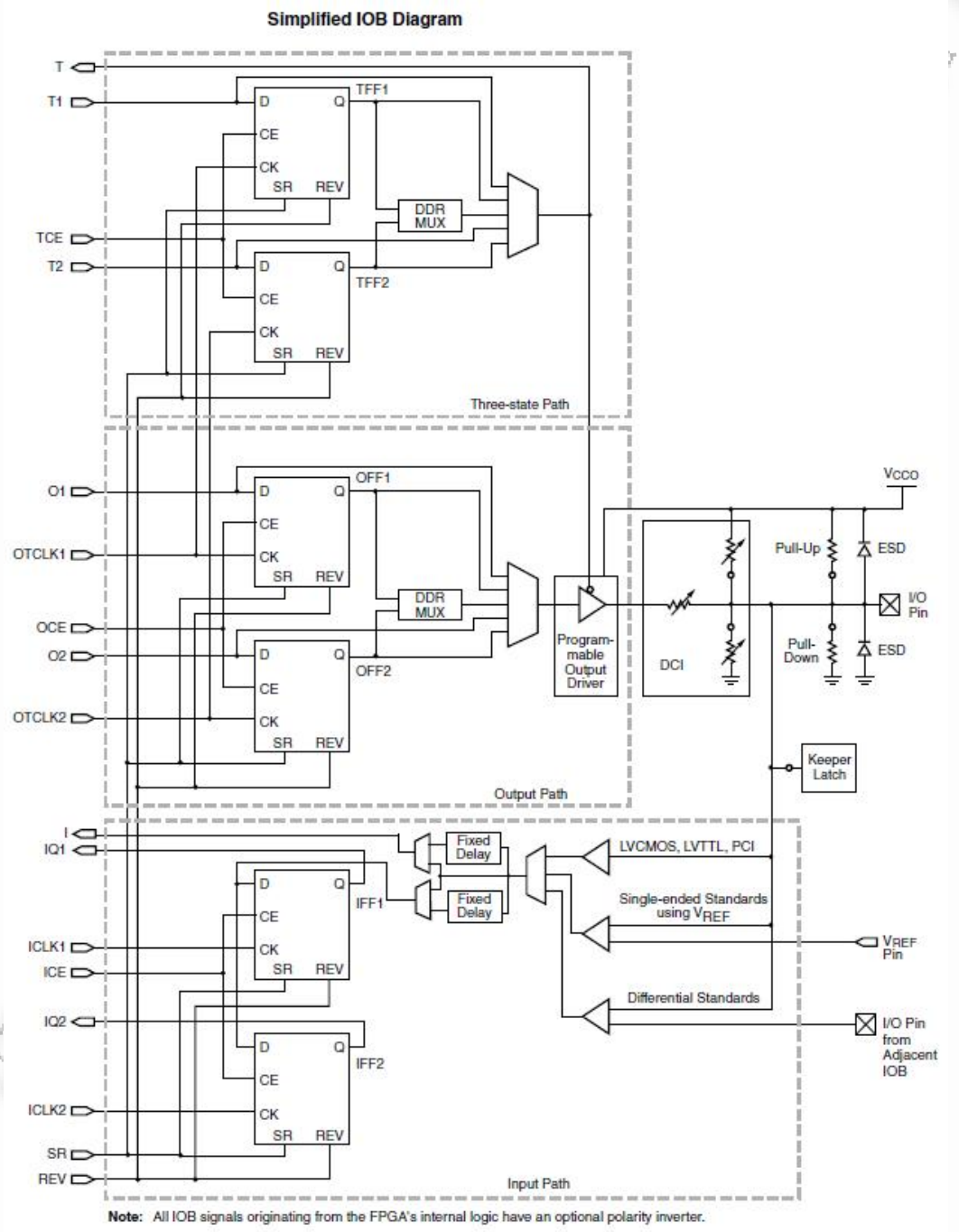
3.3.6. Ακροδέκτες εισόδου εξόδου (I/O Bank)

Οι ακροδέκτες εισόδου εξόδου (I/O) στην αρχιτεκτονική της οικογένειας Spartan-3 οργανώνονται σε οκτώ βαθμίδες (I/O Bank), Εικόνα 3.37. Κάθε βαθμίδα περιέχει ένα σύνολο από ακροδέκτες (IOB, I/O Block) το οποίο εξαρτάται από τη δομή της κάθε συσκευής.



Εικόνα 3.37 [XIn1] Δομή των βαθμίδων (I/O Bank) των ακροδεκτών στην αρχιτεκτονική της οικογένειας FPGA Spartan-3

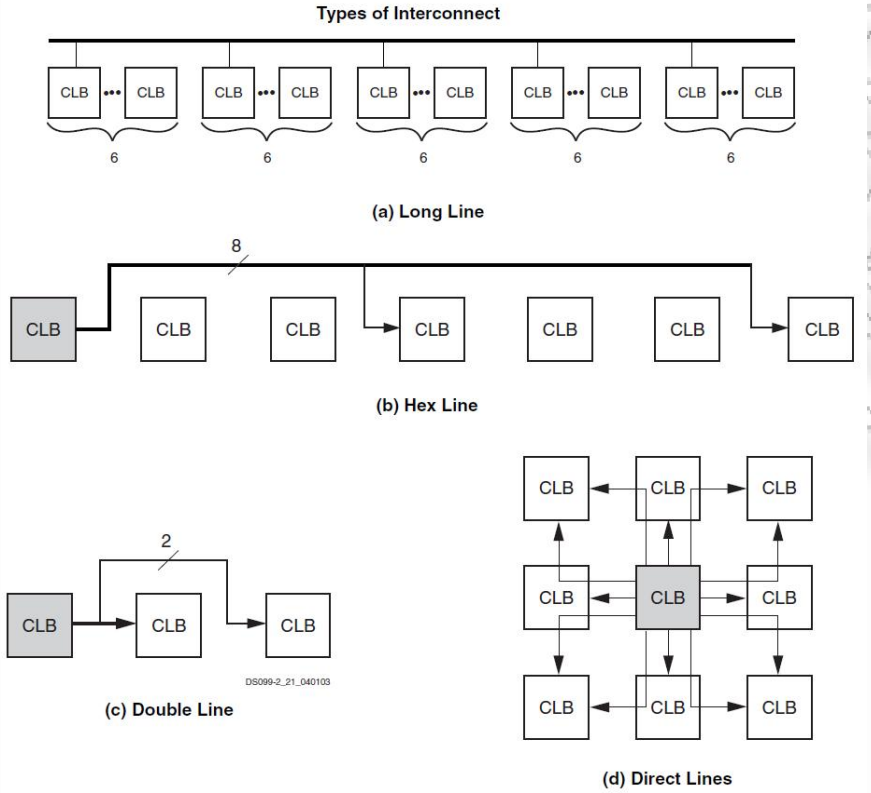
Το σύνολο ακροδεκτών IOB παρέχει μια προγραμματιζόμενη μονής ή διπλής κατεύθυνσης διεπαφή μεταξύ ενός πακέτου ακροδεκτών και της εσωτερικής λογικής του FPGA, υποστηρίζοντας μια μεγάλη ποικιλία τυποποιημένων διεπαφών. Το σύνολο των χαρακτηριστικών περιλαμβάνει προγραμματιζόμενο έλεγχο της εξόδου και της εισόδου, καθυστέρηση της εισόδου, αποθήκευση των αποτελεσμάτων στην έξοδο ή στην είσοδο, με τους ειδικούς καταχωρητές διπλού ρυθμού (DDR, Double Data Rate). Η δομή ενός IOB φαίνεται στην Εικόνα 3.38. Προαναφέραμε κάποια χαρακτηριστικά όπως, μονοί και διαφορικοί ακροδέκτες, πχ όταν έχει 784 μονούς ακροδέκτες αυτοί μπορούν να γίνουν 344 διαφορικά ζεύγη, 26 πρότυπα εισόδου-εξόδου (I/O) όπου περισσότερα πρότυπα επιτρέπουν περισσότερες δυνατότητες ολοκλήρωσης στο σύστημα [PsM1]. Επιπλέον, σύνδεση Chip με Chip LVDS, LVCMOS, LVTTL, διασύνδεση με πλακέτα βάσης (balkplane) GTL, GTL+, PCI, BLVDS, διασύνδεση με μνήμη υψηλής ταχύτητας HSTL, SSTL, υποστήριξη PCI 32/33 και 64/33.



Εικόνα 3.38 [Xln1] Δομή ενός συνόλου ακροδεκτών (I/O Block) στην αρχιτεκτονική της οικογένειας FPGA Spartan-3

3.3.7. Διασύνδεση

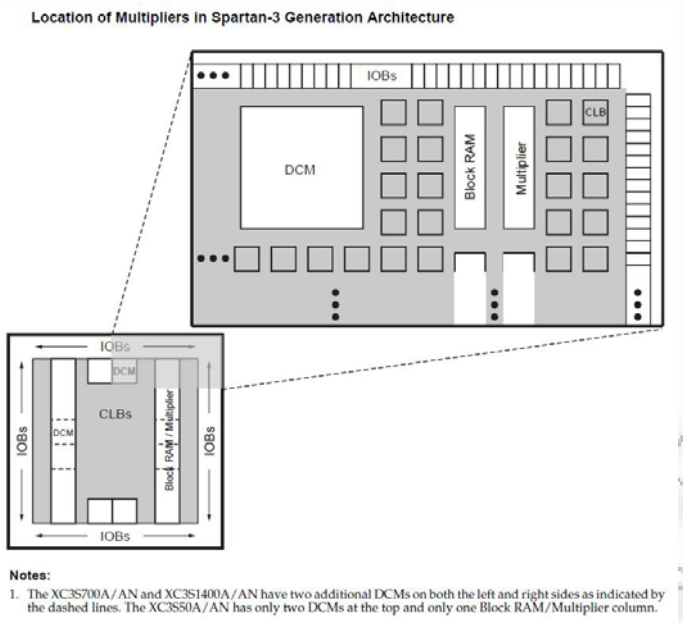
Ο τρόπος διασύνδεσης των CLB μεταξύ τους φαίνεται στην Εικόνα 3.39, όπου στο a) *Long Line* έχουμε διασύνδεση κάθε CLB ανά έξι, στο b) *Xex Line* έχουμε διασύνδεση κάθε CLB ανά 3, στο c) *Double Line* έχουμε διασύνδεση κάθε CLB ανά 2 και τέλος στο d) *Direct Line* έχουμε άμεση διασύνδεση κάθε CLB με τα γειτονικά του.



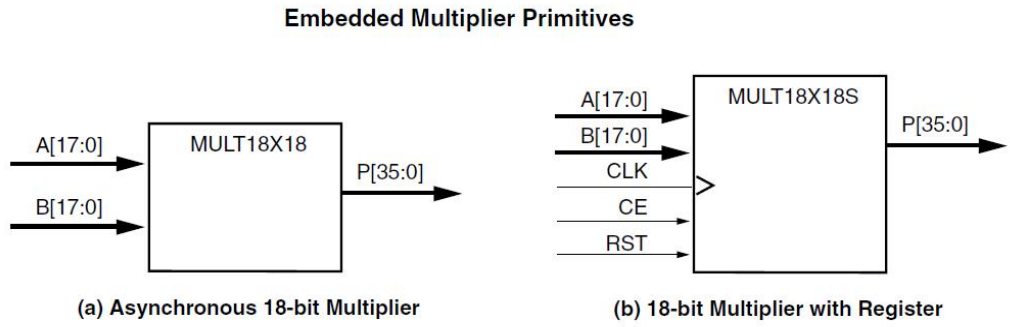
Εικόνα 3.39 [XIn1] Δομή της διασύνδεσης των CLB στην αρχιτεκτονική της οικογένειας FPGA Spartan-3

3.3.8. Πολλαπλασιαστές

Μία επιπλέον λογική μονάδα που έχει η αρχιτεκτονική δομή της οικογένειας Spartan-3 είναι οι *Πολλαπλασιαστές* (*Multipliers*). Ο κάθε πολλαπλασιαστής βρίσκεται δίπλα σε μια βαθμίδα μνήμης (RAM Block) έτσι ώστε να διευκολύνει τον υπολογισμό των αριθμητικών πράξεων και των δεδομένων, Εικόνα 3.40. Αυτοί οι πολλαπλασιαστές δέχονται δύο 18bit λέξεις ως είσοδο και παράγουν ως έξοδο μια λέξη των 36bit. Υπάρχουν δύο εκδόσεις, μία με την ονομασία MULT18X18 η οποία είναι ασύγχρονη και μια έκδοση με καταχωρητή, με την ονομασία MULT18X18S, Εικόνα 3.41 a) και b).



Εικόνα 3.40 [Xln2] Δομή πολλαπλασιαστών (Multipliers) στην αρχιτεκτονική της οικογένειας FPGA Spartan-3



Εικόνα 3.41 [Xln1] Οι δύο εκδόσεις των πολλαπλασιαστών (Multipliers) της οικογένειας FPGA Spartan-3

Για περισσότερη λεπτομέρεια για όσα έχουμε αναφέρει παραπάνω σε σχέση με την αρχιτεκτονική δομή της οικογένειας FPGA Spartan-3 της εταιρίας Xilinx, θα σας παραπέμψουμε στους οδηγούς χρήσης *Spartan-3 FPGA Family Data Sheet* [Xln1] και *Spartan-3 Generation FPGA User Guide* [Xln2].

3.4. Αρχιτεκτονική Δομή Atmel AT40K FPGA

3.4.1. Αρχιτεκτονική και γενικά χαρακτηριστικά

Η τελευταία αρχιτεκτονική δομή που θα εξετάσουμε είναι αυτή της οικογένειας FPGA *AT40K* της εταιρίας *Atmel* [Atm1]. Πολύ γρήγορα κάποια χαρακτηριστικά είναι τα εξής: η λειτουργία ισχύος της συσκευής κυμαίνεται στα *3V/5V, 0.13-μm*, έχει πυκνότητα μέχρι 2304 λογικά στοιχεία (LEs), μνήμη RAM από 2,048 έως 18,432 bits και μπορεί να λειτουργήσει ως μόνη ή διπλή θύρα (Single / Dual Port). Επιπλέον έχει οκτώ ανεξάρτητα ελεγχόμενο ρολόγια, σήματα εκκαθάρισης και περιβάλλεται από προγραμματιζόμενους ακροδέκτες εισόδου εξόδου (I/O). Η ταχύτατα του συστήματος κυμαίνεται στα 100 MHz. Παρακάτω βλέπουμε κάποια από τα παραπάνω χαρακτηριστικά αλλά και άλλα ακόμη.

Η συσκευές της οικογένειας AT40K προσφέρουν τις ακόλουθες δυνατότητες:

- Εξαιρετικά υψηλές επιδώσεις.
  - Ταχύτητες Σύστημα στα 100 MHz.
  - Διατάξεις Πολλαπλασιαστών > 50 MHz.

- 10 ns Ευέλικτη SRAM.
- Εσωτερική Three-state δυνατότητα σε κάθε κύτταρο.

- RAM
  - Ευέλικτη, Single/Dual Port, Σύγχρονη/Ασύγχρονη 10 ns SRAM.
  - 2.048 - 18.432 Bits κατανεμημένης SRAM, ανεξάρτητα από τα λογικά κύτταρα.
- 128 - 384 PCI συμβατότητα I/O ακροδέκτες.
  - Δυνατότητα 3V/5V.
  - Προγραμματιζόμενη μονάδα εξόδου.
  - Γρήγορη, ευέλικτη πρόσβαση όπου διευκολύνει την ασφάλιση των ακροδεκτών.
  - Ακροδέκτες συμβατοί με XC4000, XC5200 FPGAs.
- 8 Γενικά Ρολόγια
  - Γρήγορα, χαμηλή απόκλισης ρολογιού.
  - Προγραμματιζόμενες Rising / Falling μεταβάσεις ακμών.
  - Κατανεμημένη δυνατότητα ρολογιού για χαμηλή διαχείρισης ενέργειας.
  - Ασύγχρονη επαναφορά των επιλογών.
  - 4 επιπλέον εξειδικευμένα Ρολόγια PCI.
- Λογική Cache ® Dynamic πλήρης ή μερικής εκ νέου διαμόρφωσης In-System.
  - Επαναπρογραμματισμός μέσω σειριακής ή παράλληλης λειτουργίας.
  - Δίνει τη δυνατότητα Adaptive σχεδίαση.
  - Επιτρέπει τη γρήγορη ανανέωση διανυσματικού πολλαπλασιαστής.
  - QuickChange εργαλεία για γρήγορη και εύκολη αλλαγή σχεδιασμού.
- Ακροδέκτες με συμβατή επιλογή πακέτων.
  - Πλαστικά Μολυβδούχος Carriers Chip (PLCC).
  - Λεπτό, πλαστικό Quad Flat πακέτο (LQFP, TQFP, PQFP).
  - Πίνακες Ball Grid (BGAs).
- Η βιομηχανικού πρότυπα εργαλεία σχεδίασης
  - Ανεπαίσθητη ενσωμάτωση (βιβλιοθήκες, Interface, Full Back-σχολιασμός) και με Concept ®, Everest, Exemplar ™, Mentor ®, OrCAD ®, Synario ™, Synopsys ®, Verilog ®, Veribest ®, Viewlogic ®, Synplicity ®.
  - Timing Driven τοποθέτηση και δρομολόγηση.
  - Αυτόματη / Αλληλεπιδραστική πολλαπλών ψηφίδων διαμέρισης.
  - Γρήγορη, αποτελεσματική σύνθεση.
  - Πάνω από 75 αυτόματες γεννήτριες δημιουργία 1000s από επαναχρησιμοποιούμενες, πλήρως ντετερμινιστικής λογικής και λειτουργίες μνήμης RAM.
- Πυρήνες
  - Fir φίλτρα, UARTs, PCI, FFT και άλλες λειτουργίες επιπέδου συστήματος.
- Εύκολη μετανάστευση από τις διατάξεις θυρών της Atmel για μεγαλύτερης παραγωγής διάταξη.
- Τάση τροφοδοσίας 5V για AT40K, και 3.3V για AT40KLV.

Η οικογένεια *AT40K* παρέχει τις συσκευές που φαίνονται στο Πίνακα 3.4. Όπως μπορείτε να διαπιστώσετε οι συσκευές αυτές δεν έχουν πολύ μεγάλη πυκνότητα σε *λογικά κύτταρα* (*Logic Cells*), όπως και μνήμη RAM, το γεγονός αυτό όμως δεν τις καθιστά υποδιαιρέστερες έναντι των άλλων συσκευών.

AT40K/AT40KLV Family				
Device	**AT40K05 AT40K05LV**	**AT40K10 AT40K10LV**	**AT40K20 AT40K20LV**	**AT40K40 AT40K40LV**
Usable Gates	**5K - 10K**	**10K - 20K**	**20K - 30K**	**40K - 50K**
Rows x Columns	**16 x 16**	**24 x 24**	**32 x 32**	**48 x 48**
Cells	**256**	**576**	**1,024**	**2,304**
Registers	**256**	**576**	**1,024**	**2,304**
RAM Bits	**2,048**	**4,608**	**8,192**	**18,432**
I/O (Maximum)	**128**	**192**	**256**	**384**

Πίνακας 3.4 [Atm1] Οικογένεια συσκευών AT40K

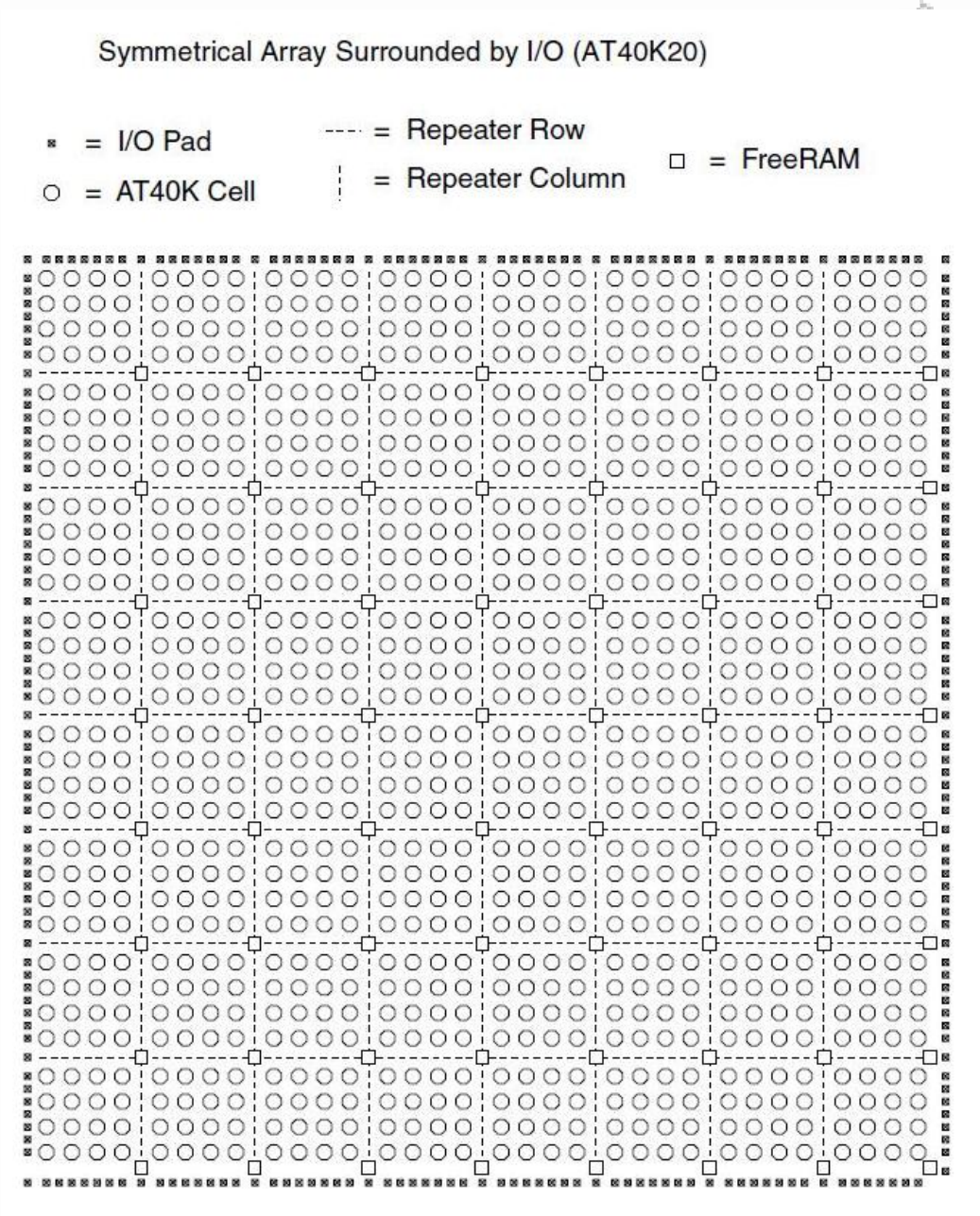
Θα δούμε, από τη δομή της οικογένειας AT40K, ότι υπάρχει μια πολύ διαφορετική αρχιτεκτονική σχεδίαση σε σχέση με τις άλλες δύο οικογένειες FPGA που έχουμε προαναφέρει. Δυστυχώς δεν κατέστη δυνατόν να δούμε και τα πειραματικά αποτελέσματα αυτής της πρωτότυπης σχεδίασης στην πράξη. Η αιτία αυτού του προβλήματος ήταν ότι για την πλήρη χρήση του εργαλείου σχεδίασης (CAD) ISD 7.6 Figaro, με το οποίο μπορεί κανείς να προγραμματίσει ή να σχεδιάσει σε μια συσκευή της εταιρίας Atmel, απαιτούνταν να προσκομίσουμε άδεια χρήσης λογισμικού την οποία και δεν διαθέταμε. Έχοντας γνώση αυτής της λεπτομέρειας και πάρα τα προβλήματα που προέκυψαν, θεωρήσαμε σκόπιμο από μέρους μας να συμπεριλάβουμε στη μεταπτυχιακή διατριβή τόσο τη διάταξη AT40K όσο και το εργαλείο σχεδίασης ISD 7.6 Figaro.

Η πατενταρισμένη αρχιτεκτονική σειρά AT40K απασχολεί ένα συμμετρικό πλέγμα μικρών όμως ισχυρών *λογικών κύτταρων* (*Logic Cells*) που συνδέονται με ένα ευέλικτο δίκτυο διασύνδεσης Η δομή της διασύνδεσης είναι κάθετη, οριζόντια αλλά και διαγώνια έτσι ώστε να παρέχει μία εξαιρετική επικοινωνία των λογικών κυττάρων (Logic Cells) μεταξύ τους. Αυτή η δομή μπορεί και εφαρμόζει εξαιρετικά γρήγορους πολλαπλασιαστές χωρίς να κάνει χρήση κάποιων άλλων διαύλων. Παρέχει επίσης τη δυνατότητα χρήσης μεγάλου αριθμού συντελεστών και μεταβλητών, έτσι ώστε η υλοποίηση τους να γίνεται σε μια πολύ μικρή ποσότητα υλικού. Αυτή η δυνατότητα επιτρέπει να γίνει μεγάλη βελτίωση της ταχύτητας του συστήματος με κόστος πολύ χαμηλότερο από ότι στα συμβατικά.

Οι συσκευές AT40K μπορούν να χρησιμοποιηθούν ως συνεπεξεργαστές για υψηλής ταχύτητας (DSP / processorbased) σχεδίαση, για την εφαρμογή διαφόρων υπολογισμών και αριθμητικές λειτουργίες. Αυτές μπορεί να περιλαμβάνουν για παράδειγμα, προσαρμοστικά πεπερασμένα κρουστικής απόκρισης φίλτρα (*FIR, Finite Impulse Response*), γρήγορους μετασχηματισμούς Fourier *(FFT, Fast Fourier Transforms*), διακριτού συνημιτόνου μετασχηματισμούς (*DCT, Discrete Cosine Transforms*) τα οποία απαιτούνται για τη συμπίεση και αποσυμπίεση βίντεο, κρυπτογράφηση, συνέλιξη όπως και άλλες πολυμεσικές (*multimedia*) εφαρμογές.

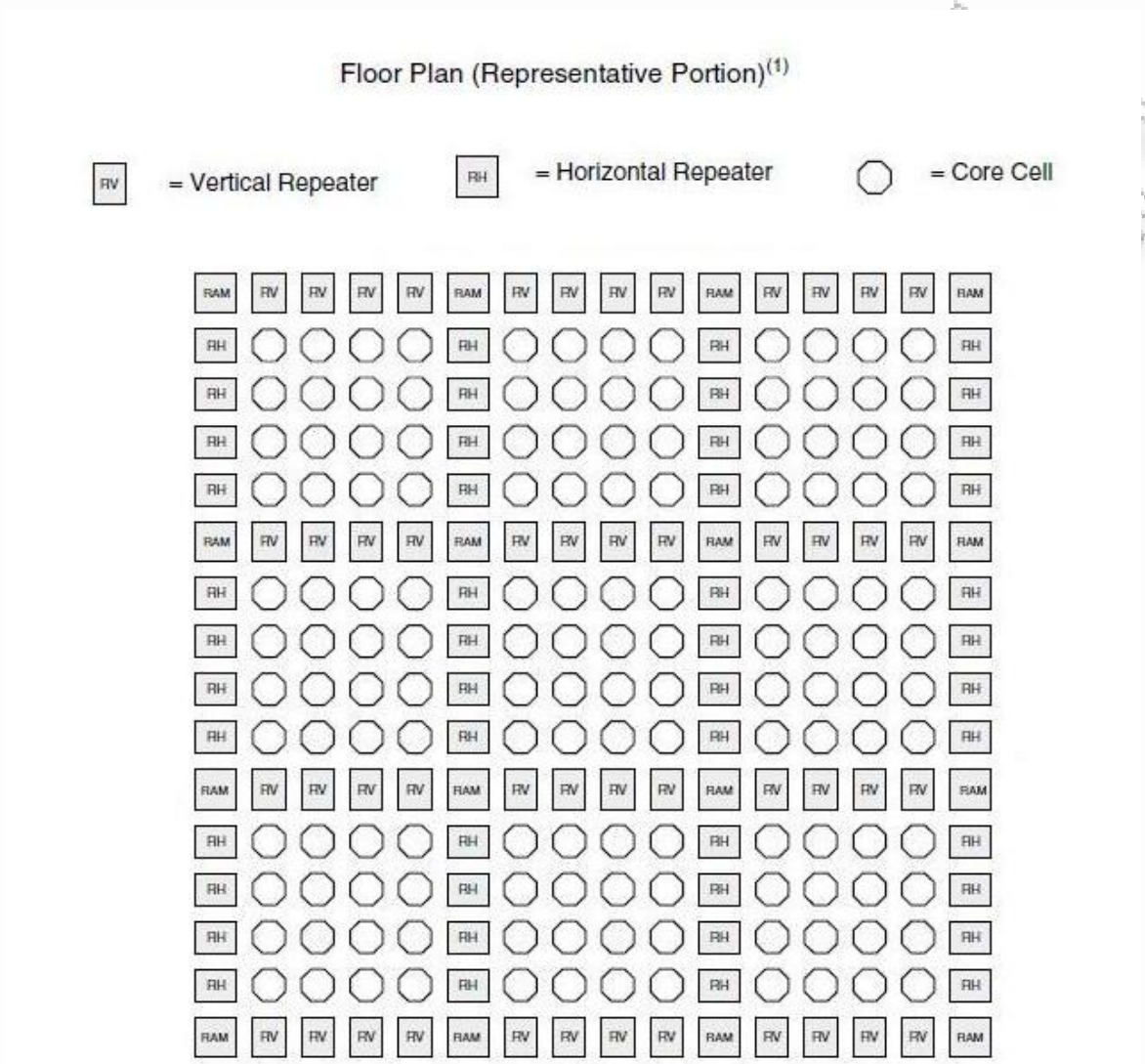
Προσφέρει μια μνήμη SRAM με μία ιδική πατέντα όπου μπορεί να διανεμηθεί η πληροφορία μέσα σε 10 ns. Αυτό συνεπάγεται ότι η μνήμη RAM μπορεί να χρησιμοποιηθεί εξολοκλήρου πολύ καλά, χωρίς τη χρήση άλλων λογικών πόρων. Επίσης παρέχει ανεξάρτητους πολλαπλασιαστές, έχει λειτουργία σύγχρονη ή ασύγχρονη, είναι διπλής θύρας ή μίας θύρας λειτουργίας RAM (FIFO, κ.λπ.). Όλα αυτά μπορούν να δημιουργηθούν χρησιμοποιώντας μια γεννήτρια παραγωγής μονάδων λογικής (*macro generator toll*) από το εργαλείο σχεδίασης CAD. Αυτή η γεννήτρια λειτουργεί απρόσκοπτα με βιομηχανικά πρότυπα όπως σχηματικής αλλά και σύνθεσης σχεδίαση. Ο σκοπός είναι να δημιουργήσουν την ταχύτερη και πιο αποτελεσματική διαθέσιμη σχεδίαση.

Η γενική αρχιτεκτονική δομή της οικογένειας AT40K είναι αυτή που φαίνεται στην Εικόνα 3.42. Όπως μπορείτε να διαπιστώσετε ακολουθεί τη γενική σχεδίαση των αρχιτεκτονικών FPGA με ενδιάμεσες μονάδες όπως είναι οι μνήμες RAM και οι διακόπτες διασύνδεσης.



Εικόνα 3.42 [Atm1] Γενική αρχιτεκτονική δομή της οικογένειας FPGA AT40K

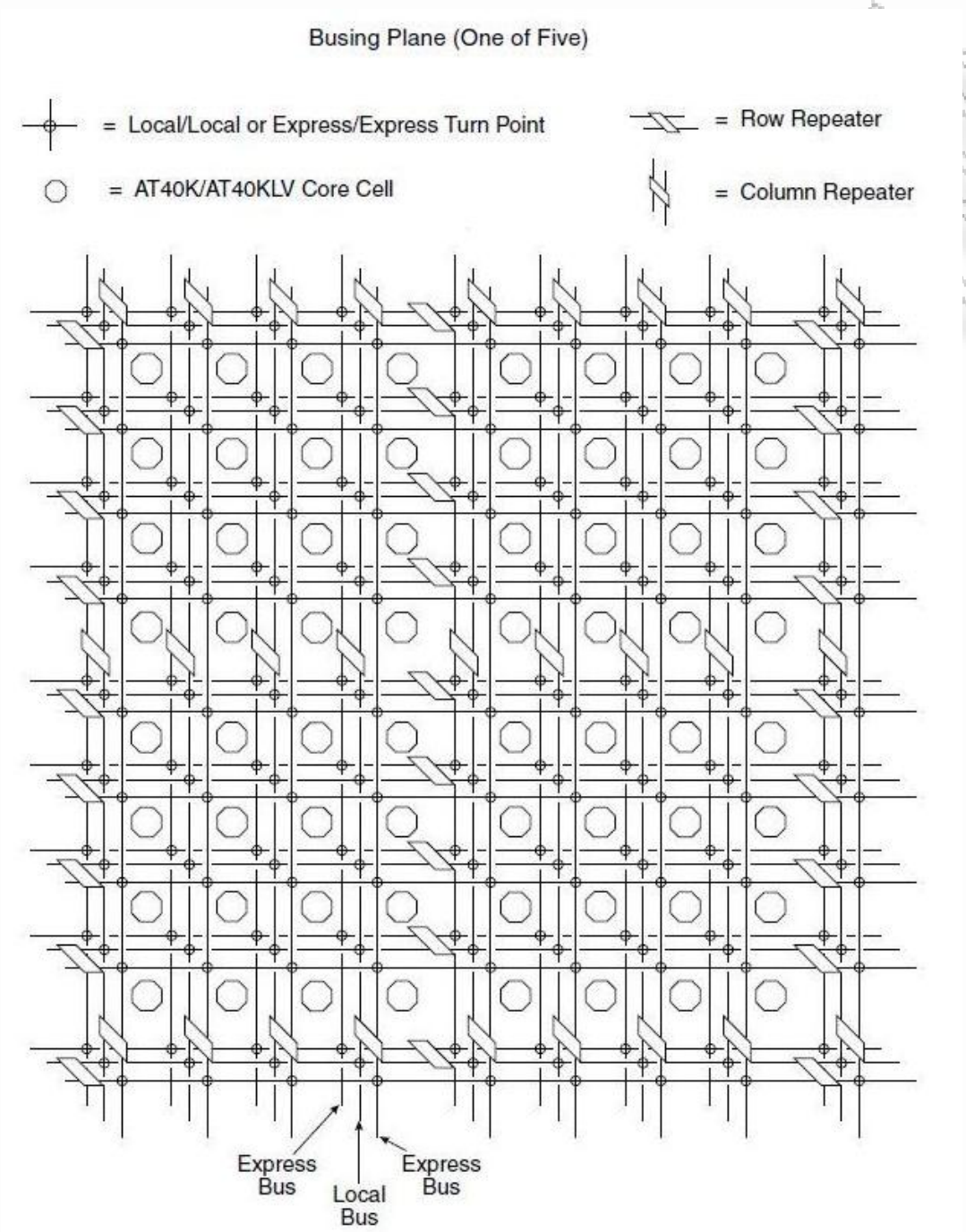
Η καρδιά της αρχιτεκτονικής Atmel είναι μια συμμετρική σειρά πανομοιότυπων κυττάρων, η συστοιχία αυτή είναι συνεχής από το ένα άκρο της διάταξης στο άλλο, εκτός από τους αναμεταδότες - επαναλήπτες (*Repeaters*) που είναι κατανεμημένοι ανά τέσσερα λογικά κύτταρα, Εικόνα 3.43. Στη διασταύρωση του κάθε επαναλήπτη της κάθε γραμμής και στήλης υπάρχει μια βαθμίδα μνήμης 32 x 4 RAM προσβάσιμη από παρακείμενους διαύλους.



Εικόνα 3.43 [Atm1] Δομή των μονάδων στην αρχιτεκτονική της οικογένειας FPGA AT40K

3.4.2. Διασύνδεση

Η Εικόνα 3.44 απεικονίζει ένα από τα πέντε πανομοιότυπα επίπεδα διασύνδεσης (*Busing Plane*). Κάθε επίπεδο έχει τρεις διαύλους, ο μεσαίος είναι για την τοπική διασύνδεση (*local bus*) και οι άλλοι δύο (που βρίσκονται στα άκρα) είναι οι γρήγοροι δίαυλοι (*express bus*). Όλοι οι δίαυλοι συνδέονται μέσω των επαναληπτών, κάθε επαναλήπτης έχει συνδέσεις σε δύο γειτονικά τμήματα διαύλου. Κάθε τοπικός δίαυλος εκτείνεται σε ένα τμήμα τεσσάρων κελιών και συνδέεται με διαδοχικούς επαναλήπτες. Κάθε γρήγορος δίαυλος εκτείνεται σε ένα τμήμα των οκτώ λογικών κύτταρων και παρακάμπτει κάποιους επαναλήπτες.



Εικόνα 3.44 [Atm1] Ένα από τα πέντε επίπεδα διασύνδεσης (Busing Plane), αρχιτεκτονική δομή της οικογένειας FPGA AT40K

Οι επαναλήπτες αναγεννούν το σήμα και μπορεί να συνδέσουν έναν οποιοδήποτε δίαυλο με έναν άλλο οποιοδήποτε δίαυλο (όλες οι οδοί είναι νόμιμες) στο ίδιο επίπεδο. Αν και δεν φαίνεται, ένας τοπικός δίαυλος μπορεί να παρακάμψει έναν επαναλήπτη μέσω μιας προγραμματιζόμενης διάταξης τριών καταστάσεων (*Three - State*). Μερικοί από αυτούς τους δίαυλους χρησιμοποιούνται και ως πηγές διπλής λειτουργίας. Ο Πίνακας 3.5 δείχνει ποιοι δίαυλοι και ποιού επιπέδου χρησιμοποιούνται σε κατάσταση διπλής λειτουργίας.

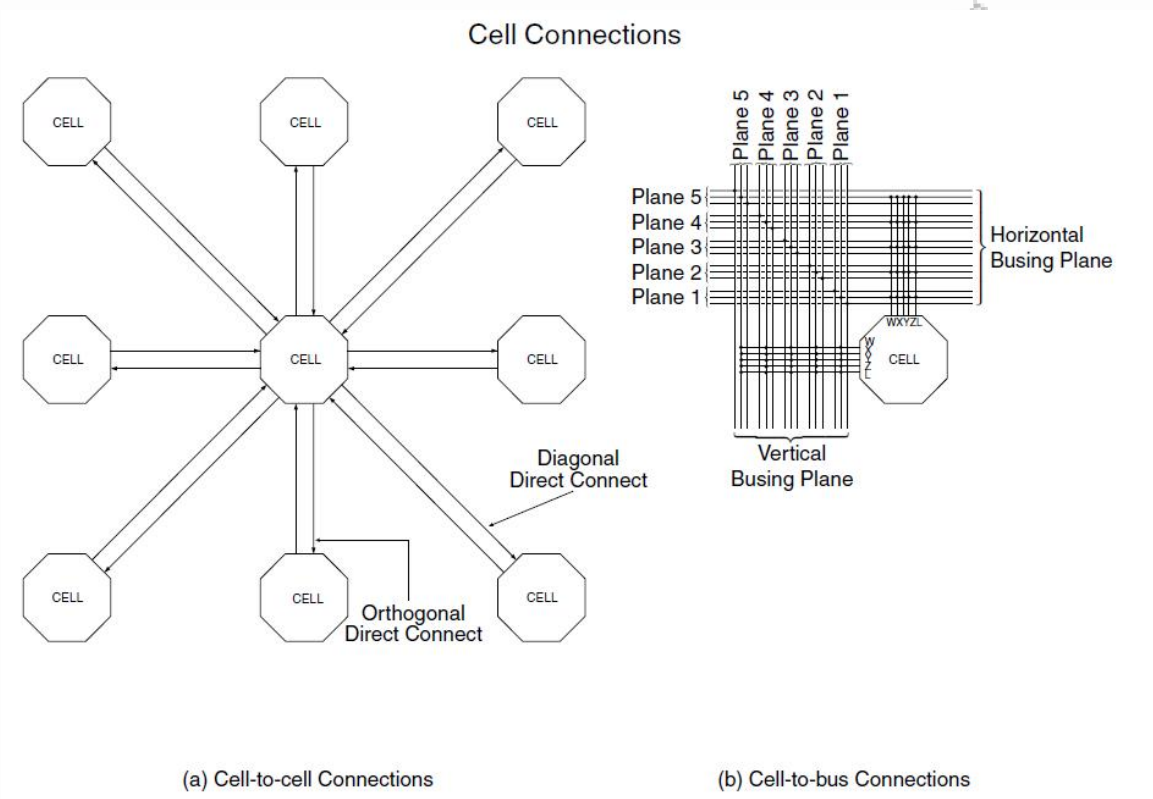
Dual-function Buses

Function	Type	Plane(s)	Direction	Comments
Cell Output Enable	Local	5	Horizontal and Vertical	
RAM Output Enable	Express	2	Vertical	Bus full length at array edge Bus in first column to left of RAM block
RAM Write Enable	Express	1	Vertical	Bus full length at array edge Bus in first column to left of RAM block
RAM Address	Express	1 - 5	Vertical	Buses full length at array edge Buses in second column to left of RAM block
RAM Data In	Local	1	Horizontal	Data In connects to local bus plane 1
RAM Data Out	Local	2	Horizontal	Data out connects to local bus plane 2
Clocking	Express	4	Vertical	Bus half length at array edge
Set/Reset	Express	5	Vertical	Bus half length at array edge

Πίνακας 3.5 [Atm1] Απεικόνιση διαύλων που χρησιμοποιούνται σε κατάσταση διπλής λειτουργίας, αρχιτεκτονική δομή της οικογένειας FPGA AT40K

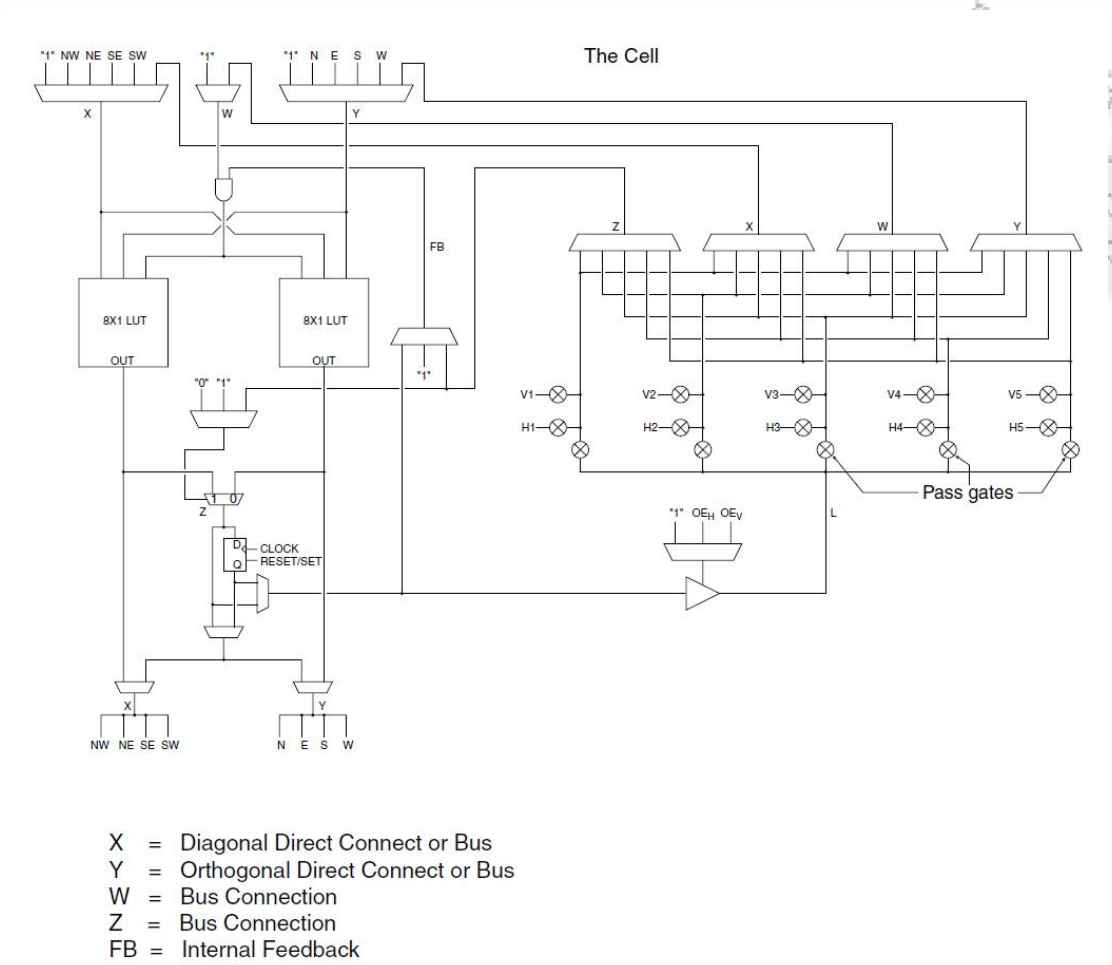
3.4.3. Λογικό στοιχείο

Όπως προαναφέραμε η αρχιτεκτονική δομή της διάταξης AT40K είναι ιδιαίτερης μορφής. Παρέχει μία διασύνδεση μεταξύ των λογικών κυττάρων (Logic Cells) διαφορετική σε σχέση με τα δυο παραπάνω που έχουμε δει, όπως και με της γενικής μορφής διασύνδεσης των FPGA. Στην Εικόνα 3.45 a) μπορούμε να δούμε αυτή τη δομή όπου απεικονίζεται η διασύνδεση ενός λογικού κυττάρου με τα οκτώ γειτονικά του (εκτός αυτών που βρίσκονται στα άκρα), ενώ στην Εικόνα 3.45 b) βλέπουμε τη διασύνδεση ενός λογικού κυττάρου με τους διαύλους οριζόντιας και κάθετης διασύνδεσης. Κάθε δίαυλος από τους πέντε που βλέπουμε αποτελεί (είτε στην οριζόντια είτε στην κάθετη) τη διασύνδεση των πέντε επιπέδων (Busing Plane) που υπάρχουν στο σύστημα.



Εικόνα 3.45 [Atm1] Διασύνδεση λογικών κυττάρων (Logic Cells), αρχιτεκτονική δομή της οικογένειας FPGA AT40K

Η δομή ενός λογικού κυττάρου φαίνεται στην Εικόνα 3.46, όπου κάνεις μπορεί να παρατηρήσει την ύπαρξη δύο LUT των τριών εισόδων. Η λειτουργία τους μπορεί να είναι είτε δύο LUT των τριών εισόδων (*3-input*) είτε να συνδυαστούν και να παράγουν ένα LUT των τεσσάρων εισόδων (*4-input*). Αυτό σημαίνει ότι κάθε κύτταρο πυρήνας μπορεί να εφαρμόσει δύο λειτουργίες των 3 εισόδων ή μία λειτουργία των 4 εισόδων. Επιπλέον όταν τα LUT λειτουργούν ως 3-input LUT μπορούν να λειτουργήσουν και σαν μια μνήμη ROM 8×1. Επιπλέον η διασύνδεση γίνεται με τα κανάλια V_n ($V_1..V_5$) και H_n ($H_1..H_5$) που φαίνονται στην εικόνα.

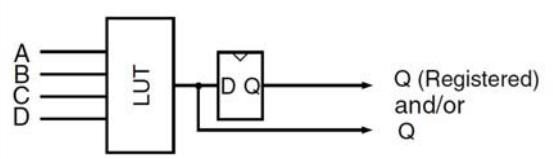


Εικόνα 3.46 [Atm1] Δομή ενός λογικού κυττάρου (Logic Cells), αρχιτεκτονική δομή της οικογένειας FPGA AT40K

Έχοντας επιπλέον λογικά κυκλώματα μέσα στη δομή, όπως πολυπλέκτες, σήματα χρονισμού, flip flop κτλ, κάθε λογικό κύτταρο μπορεί να υλοποιήσεις διαφορετικές λειτουργίες, δηλαδή μπορεί να βρεθεί σε διαφορετικές λογικές καταστάσεις.
Κάποιες από αυτές είναι οι παρακάτω :

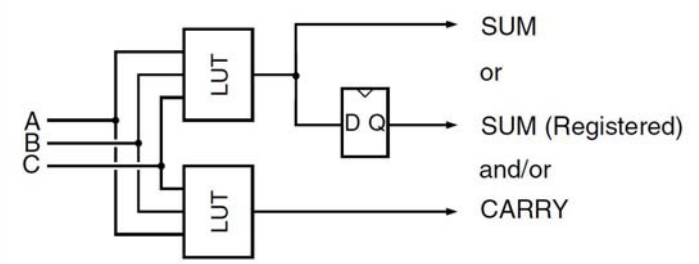
- Κατάσταση σύνθεσης (*Synthesis Mode*)
- Αριθμητική κατάσταση (*Arithmetic Mode*)
- Κατάσταση επεξεργαστή- επεξεργασίας (*DSP/ Multiplier Mode*)
- Κατάσταση Μετρητή (*Counter Mode*)
- Κατάσταση επιλογής (*Three - State/Mux Mode*)

Κατάσταση σύνθεσης (*Synthesis Mode*), η λειτουργία αυτή είναι ιδιαίτερα σημαντική για η χρήση VHDL / Verilog σχεδιασμό. Η σύνθεση με VHDL / Verilog κώδικα γενικά παράγει σαν έξοδο μεγάλο αριθμό από τυχαίες λογικές λειτουργίες. Έχοντας μια δομή 4-input LUT παρέχεται μια αποτελεσματική βελτιστοποίηση της λογικής, χωρίς τις καθυστερήσεις που μπορεί να σχετίζονται με μεγαλύτερες δομές LUT. Η έξοδος από κάθε κύτταρο μπορεί να καταχωρηθεί και οι λογικές πύλες όπως AND/OR μπορούν να ανατροφοδοτούν το βασικό κύτταρο, Εικόνα 3.47



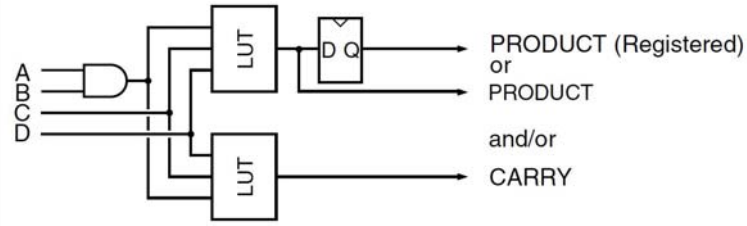
Εικόνα 3.47 [Atm1] Κατάσταση σύνθεσης (Synthesis Mode) ενός λογικού κυττάρου (Logic Cells), αρχιτεκτονική δομή της οικογένειας FPGA AT40K

Αριθμητική κατάσταση (*Arithmetic Mode*) είναι η συχνότερη χρησιμοποιούμενη στα περισσότερα σχέδια. Όπως φαίνεται στο σχήμα, το λογικό κύτταρο μπορεί να εφαρμόσει έναν πλήρη αθροιστή των 2-εισόδων με κρατούμενο εισόδου (Carry in) και κρατούμενο εξόδου (Carry Out). Η έξοδος από κάθε κύτταρο μπορεί να καταχωρηθεί και οι λογικές πύλες όπως AND/OR μπορούν να ανατροφοδοτούν το βασικό κύτταρο, Εικόνα 3.48.



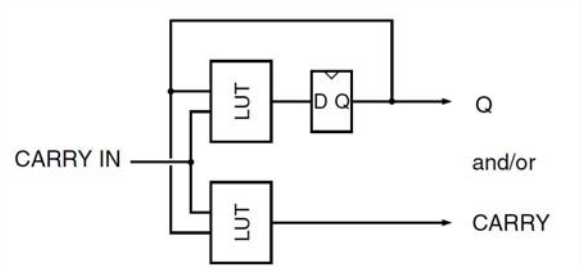
Εικόνα 3.48 [Atm1] Αριθμητική κατάσταση (Arithmetic Mode) ενός λογικού κυττάρου (Logic Cells), αρχιτεκτονική δομή της οικογένειας FPGA AT40K

Κατάσταση επεξεργαστή – επεξεργασίας (*DSP/ Multiplier Mode*), αυτή η λειτουργία χρησιμοποιείται για την αποτελεσματική εφαρμογή πολλαπλασιασμού πινάκων. Μια συστοιχία από πολλαπλασιαστές είναι ένας πίνακας από bit με πολύ καλή δομή από πολλαπλασιαστές όπου το καθένα υλοποιείται ως ένας πλήρης αθροιστής με μία πύλη AND. Χρησιμοποιώντας την πύλη AND και τη διαγώνια διασύνδεση μεταξύ των κυττάρων, η δομή της συστοιχία των πολλαπλασιαστών ταιριάζει πολύ καλά στην αρχιτεκτονική AT40K. Η έξοδος από κάθε κύτταρο μπορεί να καταχωρηθεί και οι λογικές πύλες όπως AND/OR μπορούν να ανατροφοδοτούν το βασικό κύτταρο, Εικόνα 3.49.



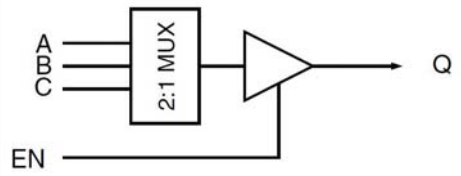
Εικόνα 3.49 [Atm1] Κατάσταση επεξεργαστή - επεξεργασίας (DSP/ Multiplier Mode) ενός λογικού κυττάρου (Logic Cells), αρχιτεκτονική δομή της οικογένειας FPGA AT40K

Κατάσταση Μετρητή (*Counter Mode*), οι μετρητές είναι θεμελιώδους σημασίας για όλα σχεδόν τα ψηφιακά σχέδια. Αποτελούν τη βάση των μηχανών κατάστασης (*state machines*), των αλυσίδων χρόνου και βασικό στοιχειό για τους διαιρέτες ρολογιού. Ένας μετρητής είναι ουσιαστικά μια προσαύξηση σε μία λειτουργία (πχ σε ένα αθροιστή). Μία μέτρηση ενός bit μπορεί να εφαρμοστεί σε ένα πυρήνα λογικού κυττάρου, με εισόδους όπως και με τις εξόδους ως ανατροφοδότηση από ένα προηγούμενο στάδιο. Και πάλι η έξοδος από κάθε κύτταρο μπορεί να καταχωρηθεί και οι λογικές πύλες όπως AND/OR μπορούν να ανατροφοδοτούν το βασικό κύτταρο, Εικόνα 3.50.



Εικόνα 3.50 [Atm1] Κατάσταση Μετρητή (Counter Mode) ενός λογικού κυττάρου (Logic Cells), αρχιτεκτονική δομή της οικογένειας FPGA AT40K

Κατάσταση επιλογής (*Three - State/Mux Mode*), αυτή η λειτουργία χρησιμοποιείται σε πολλές τηλεπικοινωνιακές εφαρμογές όπου τα δεδομένα πρέπει να δρομολογούνται μέσω περισσότερων από μία δυνατές διαδρομές. Η έξοδος του λογικού κυττάρου εξαρτάται από την λειτουργία της προγραμματιζόμενης διάταξης τριών καταστάσεων (*Three -State*), Εικόνα 3.51.

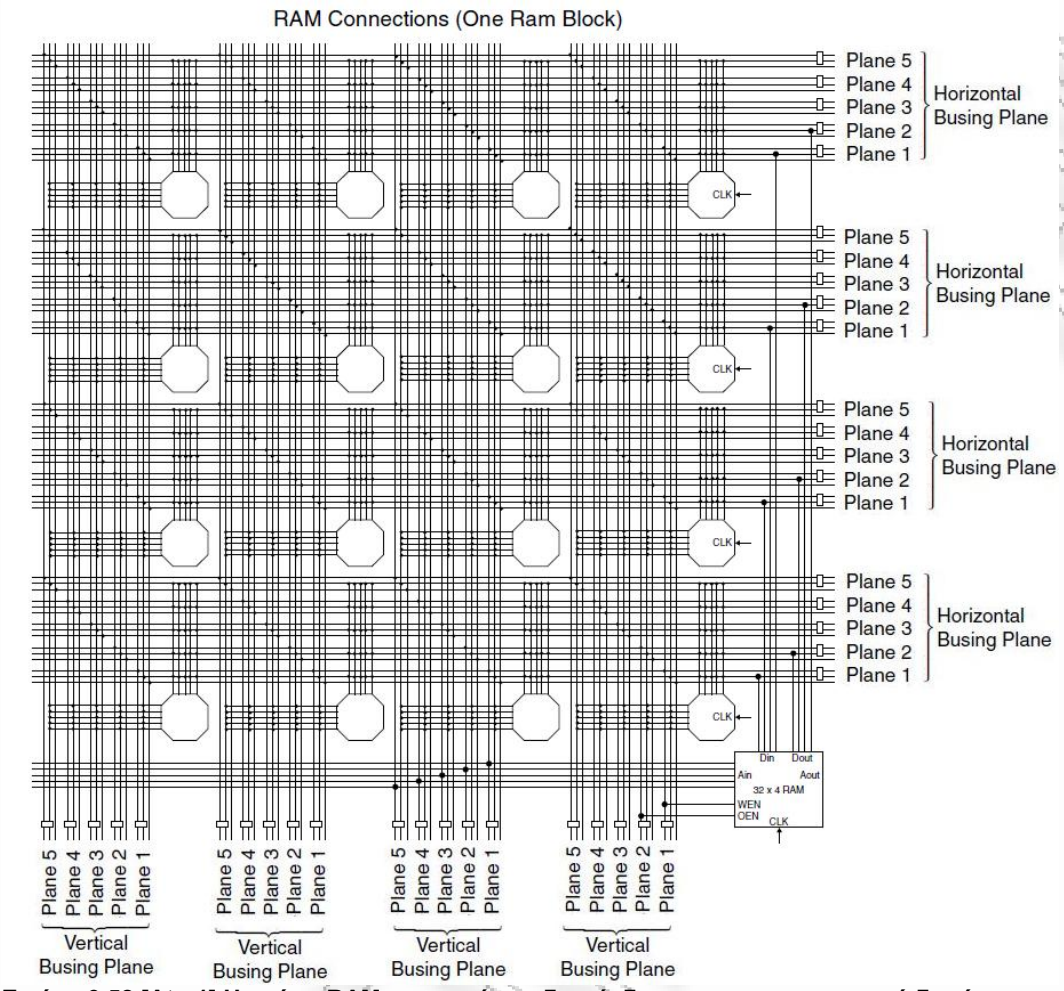


Εικόνα 3.51 [Atm1] Κατάσταση επιλογής (Three - State/Mux Mode) ενός λογικού κυττάρου (Logic Cells), αρχιτεκτονική δομή της οικογένειας FPGA AT40K

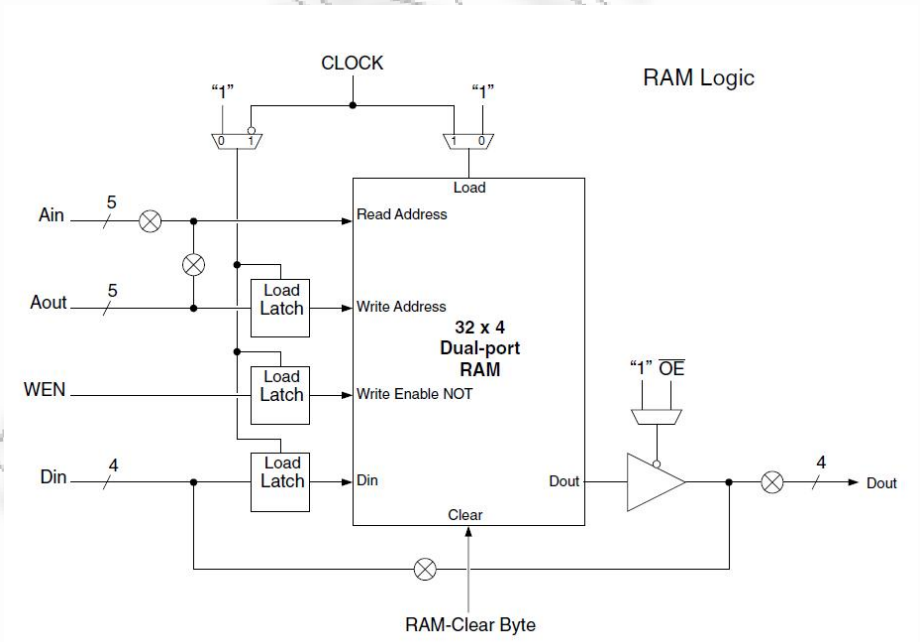
3.4.4. Μνήμη RAM

Η μνήμη RAM 32×4 διπλής θύρας (Dual Port) είναι διασκορπισμένη σε όλο τη διάταξη, Εικόνα 3.52. Ο 4bit δίαυλος εισόδου δεδομένων (*Input Data*) D_{in} συνδέεται με τέσσερις οριζόντιους τοπικούς διαύλους από τις γραμμές διασύνδεσης του επιπέδου 1 (plane 1). Ο 4bit δίαυλος εξόδου δεδομένων (*Output Data*) D_{out} συνδέεται με τέσσερις οριζόντιους τοπικούς διαύλους από τις γραμμές διασύνδεσης του επιπέδου 2 (plane 2). Ο 5bit δίαυλος εισόδου διευθύνσεων (*Input Address*) A_{in} συνδέεται με πέντε κάθετους γρήγορους διαύλους από τις γραμμές διασύνδεσης κάθε επιπέδου (plane1..plane5) στην ίδια κάθετη στήλη διασύνδεσης. Το ίδιο συμβαίνει και με τον 5bit δίαυλο εξόδου διευθύνσεων (Out*put Address*) A_{out} όπου και αυτά είναι συνδεδεμένα με τους πέντε κάθετους γρήγορους διαύλους από τις γραμμές διασύνδεσης κάθε επιπέδου στην ίδια κάθετη στήλη διασύνδεσης. Τα σήματα WEN και OEN συνδέονται με γρήγορους διαύλους στην ίδια κάθετη στήλη διασύνδεσης. Οι μνήμες RAM που βρίσκονται στα άκρα λειτουργούν ως μονής θύρας μνήμη (single port) αν κάποιο από τα A_{in} ή A_{out} δεν συνδέεται με κάποιον δίαυλο.

Η ανάγνωση και η εγγραφή σε μία μνήμη 32×4 διπλής θύρας (Dual Port) RAM γίνεται σε 10 ns και είναι ανεξάρτητες μεταξύ τους. Ανάγνωση από μία 32×4 διπλής θύρας RAM είναι εντελώς ασύγχρονη. Στην εγγραφή υπάρχουν 3 στοιχεία που ονομάζονται Latch τα οποία χρησιμοποιούνται για τον έλεγχο των δεδομένων και των διευθύνσεων που εισέρχονται στη μνήμη, Εικόνα 3.53. Επιπλέον στο ρολόι της εικόνας, αν η είσοδος είναι 1 τότε η μνήμη λειτουργεί ασύγχρονα, αν γίνει 0 τότε λειτουργεί σύγχρονα.

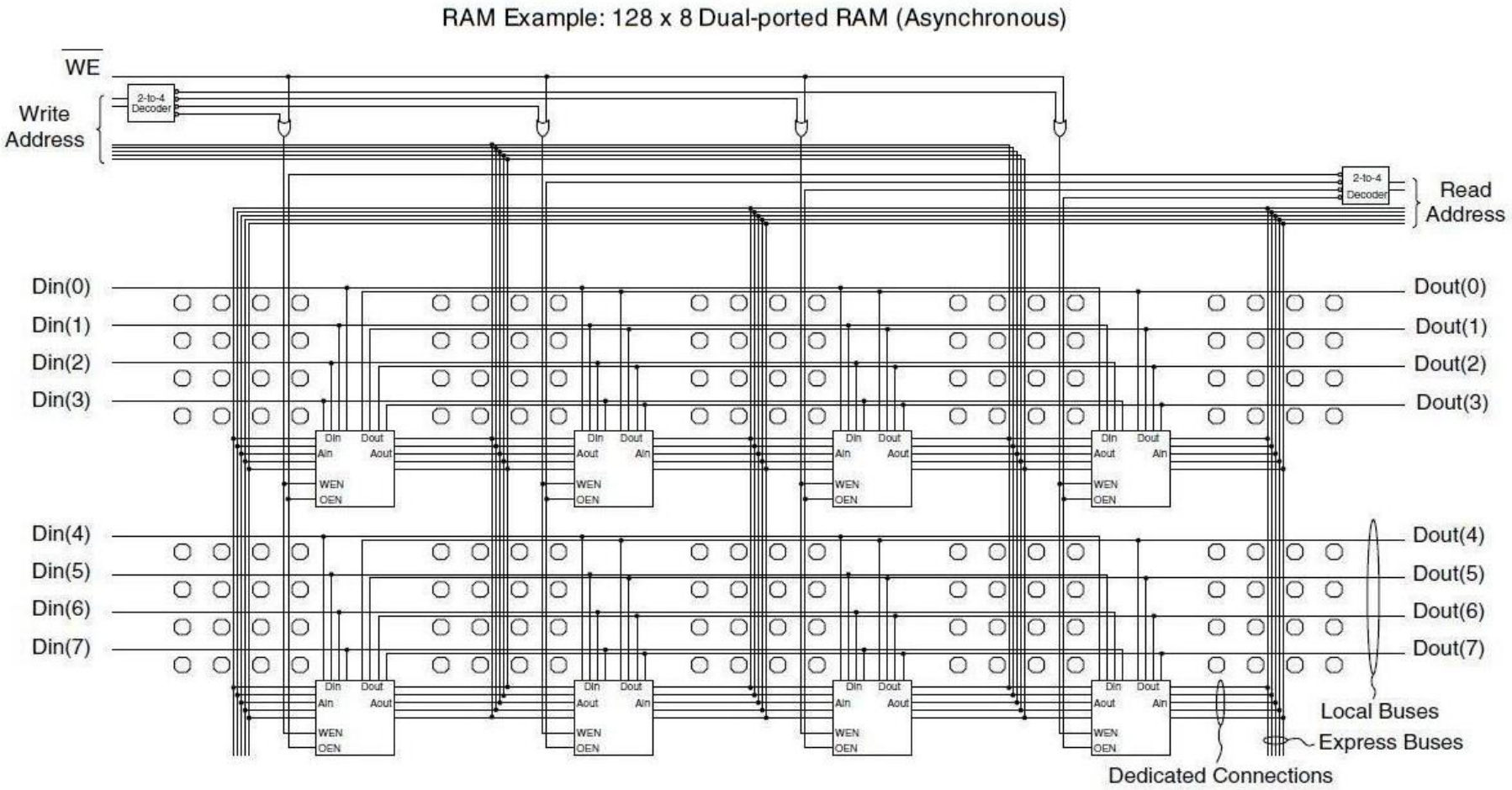


Εικόνα 3.52 [Atm1] Η μνήμη RAM και ο τρόπος διασύνδεσης της, αρχιτεκτονική δομή της οικογένειας FPGA AT40K



Εικόνα 3.53 [Atm1] Η μνήμη RAM και ο τρόπος λειτουργίας της σε κατάσταση διπλής θύρας (Dual Port), αρχιτεκτονική δομή της οικογένειας FPGA AT40K

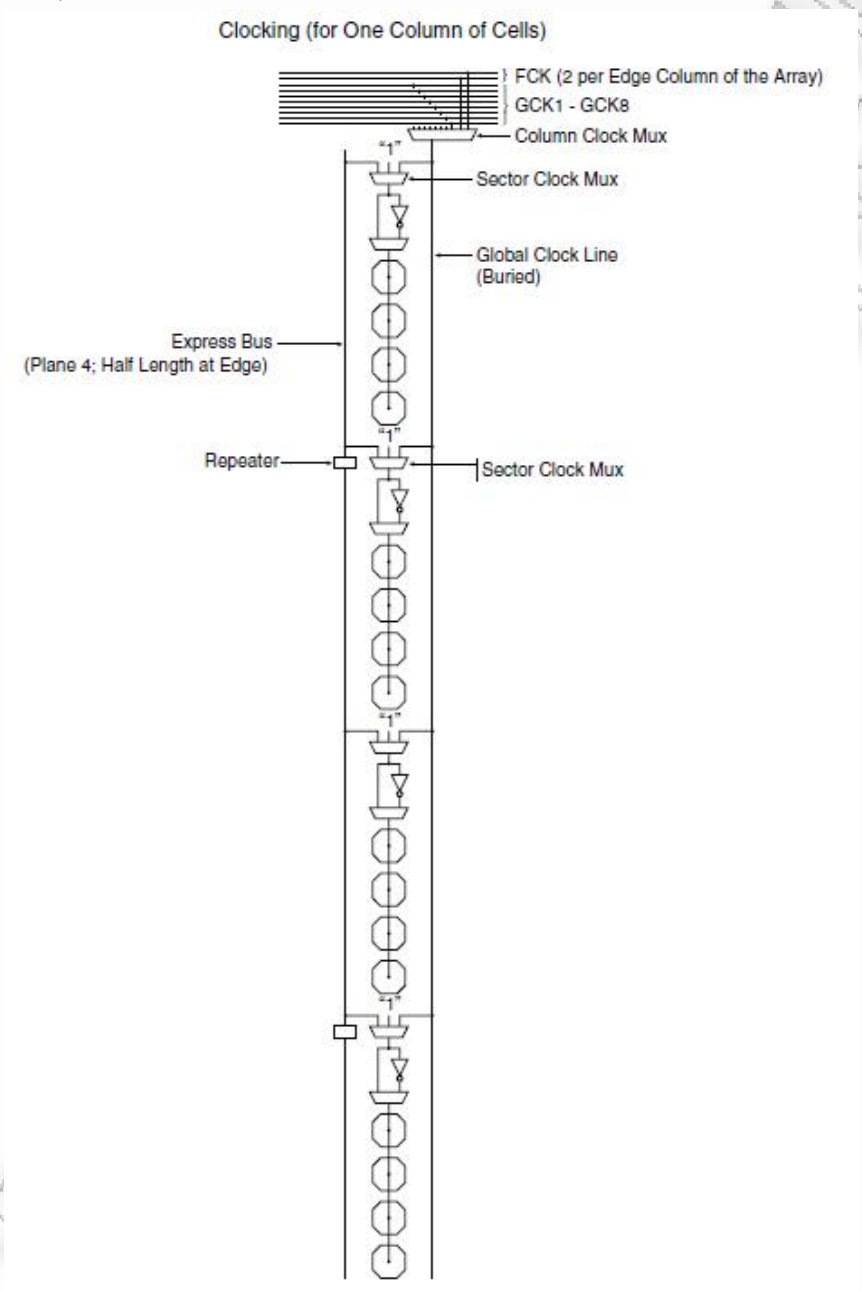
Παράλληλα θα μπορούσε να δημιουργηθεί μία ασύγχρονη μνήμη RAM 128×4 Dual Port. Χρειάζεται μόνο ένα μικρό εξάρτημα εξωτερικής λογικής για την αποκωδικοποίηση της διεύθυνσης, Εικόνα 3.54.



Εικόνα 3.54 [Atm1] Η μνήμη RAM σε μορφή 128 x 4 διπλής θύρας (Dual Port), αρχιτεκτονική δομή της οικογένειας FPGA AT40K

3.4.5. Ρολόι και σήματα αρχικοποίησης

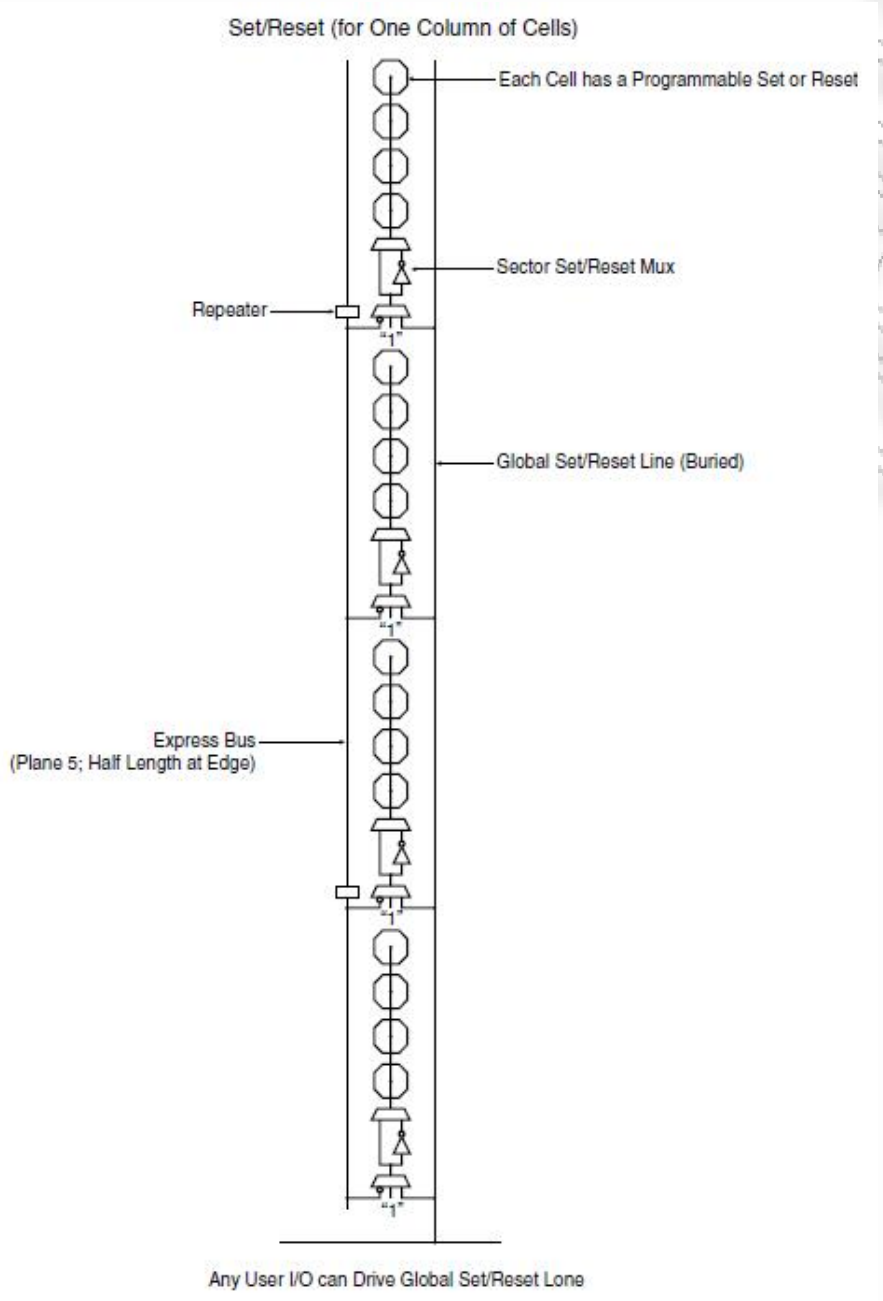
Όπως έχουμε προαναφέρει η οικογένεια AT40K έχει οκτώ γενικά ρολόγια GCK[1..8], το καθένα από αυτά συνδέονται με έναν από τους διπλούς γενικούς ωρολογιακούς ακροδέκτες (dual-use Global clock pins). Επιπρόσθετα υπάρχουν 4 επιπλέον γρήγορα ρολόγια (fast clock) τα FCK[1..4] δύο για κάθε άκρο, όπου βρίσκονται σε κάθε στήλη για την προδιαγραφή εξυπηρέτησης της περιφερικής διασύνδεσης PCI (*Peripheral Component Interconnect*), Εικόνα 3.55. Η ωρολογιακή δομή έχει σχεδιαστεί έτσι ώστε να είναι δυνατή και αποτελεσματική η χρήση πολλαπλών ρολογιών με χαμηλή απόκλιση ρολογιού (clock skew).



Εικόνα 3.55 [Atm1] Ωρολογιακή διάταξη στήλης, αρχιτεκτονική δομή της οικογένειας FPGA AT40K

Οι καταχωρητές ενεργοποιούνται να λειτουργούν από το κατασκευαστή με τη θετική ακμή του ρολογιού (θετικά ακμοπυροδοτούμενα), μπορεί όμως ο χρήστης να διαμορφώσει αυτή την επιλογή σύμφωνα με τις ανάγκες του.

Η αρχιτεκτονική των σημάτων Set / Reset είναι πανομοιότυπη με αυτή του ρολογιού. Παρέχεται μόνο ένα γενικό σήμα εκκαθάρισης (Reset), Εικόνα 3.56. Το πρόγραμμα σχεδίασης και διαχείρισης της διάταξης τοποθετεί αυτόματα την επιλογή Set / Reset στην πιο χρησιμοποιημένη από τις γενικές χρήσης διαύλους διασύνδεσης. Και πάλι όμως δίνεται η δυνατότητα στο χρήστη να επέμβει και να τροποποιήσει αυτή τη λειτουργία σύμφωνα με τις ανάγκες του.

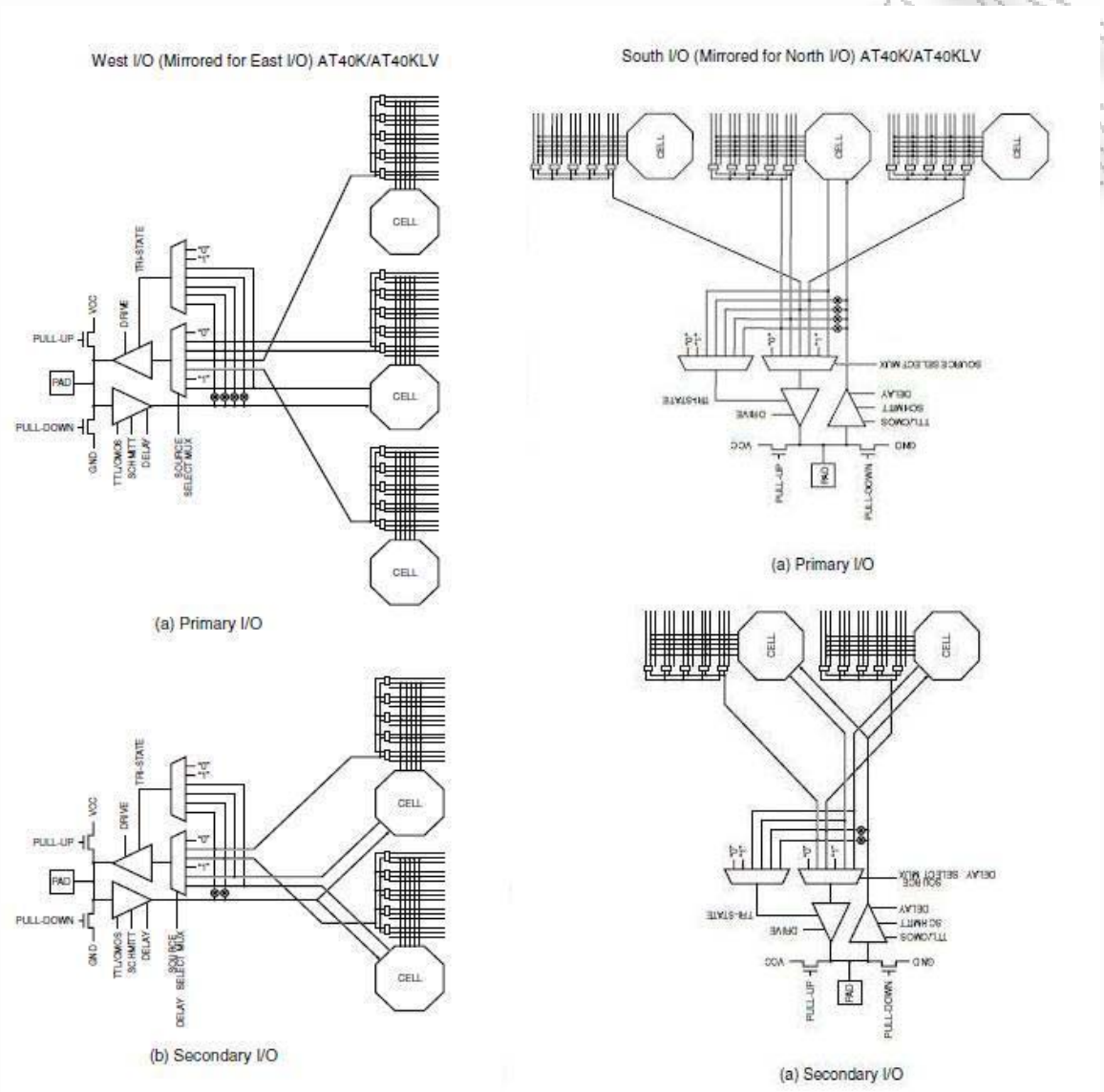


Εικόνα 3.56 [Atm1] Η αρχιτεκτονική δομή των σημάτων Set / Reset στήλης, αρχιτεκτονική δομή της οικογένειας FPGA AT40K

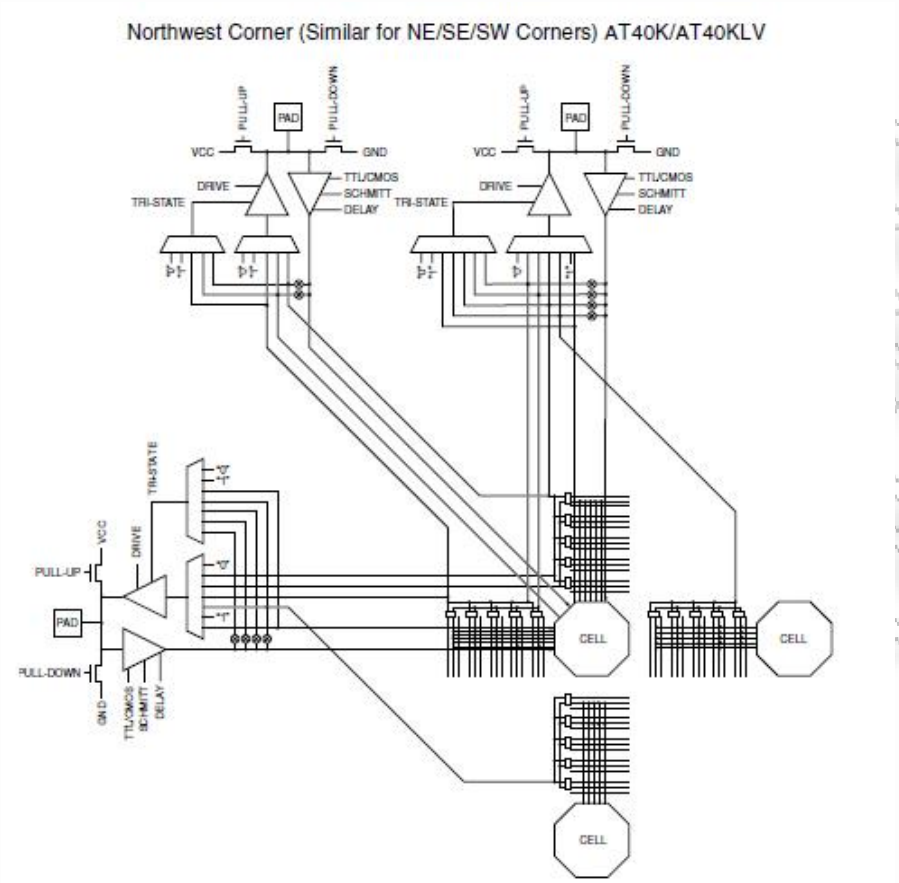
3.4.6. Ακροδέκτες εισόδου εξόδου (I/O Pad)

Οι ακροδέκτες εισόδου εξόδου (I/O Pad) είναι τα στοιχεία που επιτρέπουν την επικοινωνία της διάταξης AT40K με τον έξω κόσμο. Οι ακροδέκτες pad είναι τριών ειδών τα οποία είναι τα πρωτεύοντα (Primary I/O) τα δευτερεύοντα (Secondary I/O) και τα γωνιακά (Corner I/O). Τα πρωτεύοντα έχουν άμεση επικοινωνία με τα λογικά κύτταρα με οριζόντια και κάθετη διασύνδεση. Τα δευτερεύοντα έχουν

επικοινωνία με τα λογικά κύτταρα, με διαγώνιους διαύλους επικοινωνίας και τέλος τα γωνιακά έχουν συνδυασμό κάθετων, οριζόντιων και διαγώνιων δίαυλων, Εικόνες 3.57 και 3.58.



Εικόνα 3.57 [Atm1] Πρωτεύοντες (Primary I/O PAD) και δευτερεύοντες (Secondary I/O PAD) ακροδέκτες, αρχιτεκτονική δομή της οικογένειας FPGA AT40K



Εικόνα 3.58 [Atm1] Γωνιακοί (Corner I/O PAD) ακροδέκτες, αρχιτεκτονική δομή της οικογένειας FPGA AT40K

Για περισσότερη λεπτομέρεια για όσα έχουμε αναφέρει παραπάνω σε σχέση με την αρχιτεκτονική δομή της οικογένειας FPGA AT40K της εταιρίας Atmel, θα σας παραπέμψουμε στο οδηγό χρήσης *AT40K Device Datasheets* [Atm1].

3.5. Διαφορές των αρχιτεκτονικών

Είδαμε τρεις διαφορετικές οικογένειες συσκευών FPGA με την κάθε μία να αναδεικνύει τα προτερήματά της. Σε γενικές γραμμές είδαμε αρχιτεκτονικές ταυτισμένες με τη γενική δομή και φιλοσοφία που είδαμε στην Ενότητα 3.1. Σε αυτή την ενότητα θα δούμε τις κύριες γενικές διαφορές που διαπιστώσαμε, όπου ευελπιστούμε να τις έχετε διαπιστώσει και εσείς, μεταξύ των αρχιτεκτονικών δομών των συσκευών Cyclone, Spartan-3 και AT40K. Είναι λογικό να μην υπάρχει 100% ταύτιση των αρχιτεκτονικών δομών μεταξύ τους, για τον απλούστατο λόγο της ύπαρξης της εμπορικής ανταγωνιστικότητας μεταξύ των εταιριών. Δεν θα επισέλθουμε σε διαφορές που έχουν να κάνουν με το κατασκευαστικό κομμάτι, αλλά θα αναφερθούμε σε βασικές αρχιτεκτονικές διαφορές. Τις διαφορές αυτές θα τις δούμε με μια πιο αφαιρετική ματιά, με τη ματιά του πελάτη – αγοραστή.

Μαζί με την παρουσίαση των χαρακτηριστικών που διαφέρουν θα αναφερθούμε και σε κάποιες ομοιότητες που υπάρχουν, με απώτερο σκοπό να δούμε καλύτερα τις διαφορές. Για να εξηγήσουμε τις διαφορές και να διευκρινίσουμε τις λεπτομέρειες κάθε αρχιτεκτονικής στη συνέχεια της ενότητας θα χρειαστεί να ανατρέξουμε σε κάποιες από τις εικόνες που έχουν παρουσιαστεί σε προηγούμενες ενότητες. Οι σημαντικότερες διαφορές που παρατηρήσαμε είναι οι παρακάτω :

Δομή διασύνδεσης

- Η δομή διασύνδεσης μεταξύ των λογικών στοιχείων (Cell) στην περίπτωση της αρχιτεκτονικής της συσκευής AT40K, Εικόνα 3.45 είναι εμφανής! Η γενική διασύνδεση των FPGA έχει να κάνει με οριζόντιες και κάθετες διασυνδέσεις των λογικών στοιχείων μεταξύ τους αλλά και των λογικών βαθμίδων LAB και CLB, Εικόνα 3.13 και 3.39 αντίστοιχα. Σε αυτό το κομμάτι η

εταιρία Atmel πρωτοτύπησε! Βλέπουμε ότι η αρχιτεκτονική της συσκευής AT40K εκμεταλλεύεται το πολύ καλό σύστημα διασύνδεσης της (Εικόνα 3.44), έτσι ώστε να παρέχει (πέραν της κάθετης και οριζόντιας διασύνδεσης) και διαγώνια επικοινωνία των λογικών στοιχείων μεταξύ τους. Οι άλλες αρχιτεκτονικές απλά συνδέουν τα λογικά στοιχεία μεταξύ τους με οριζόντιες και κάθετες διασυνδέσεις.

➢

Λογικά Στοιχεία

- Το λογικό στοιχείο (Cell) της αρχιτεκτονικής της συσκευής AT40K περιέχει δύο πίνακες αναζήτησης (LUT) των 3 εισόδων (3 Input LUT) με δυνατότητα επέκτασης σε έναν πίνακα αναζήτησης των 4 εισόδων (4 Input LUT), Εικόνα 3.46. Οι δύο άλλες αρχιτεκτονικές στα λογικά τους στοιχεία έχουν από ένα πίνακα αναζήτησης των 4 εισόδων (4 Input LUT), Εικόνα 3.16 και 3.29.
- Το λογικό στοιχείο (Logic Cell) της αρχιτεκτονικής της συσκευής Spartan-3 δεν υφίσταται σαν αυτούσια λογική μονάδα όπως των άλλων δυο αρχιτεκτονικών, (Logic Element και Cell). Έχει την υπόσταση της σαν ένα σχήμα λογικής που περιβάλει ένα πίνακα αναζήτησης (LUT). Αυτή η λογική υπόσταση έχει βρει δομή στις "φέτες" (Slices) που είναι της μορφής SLICEM και SLICEL. Το καθένα από αυτά έχει υποτίθεται 2 λογικά στοιχεία, έχει όμως στην πραγματικότητα δύο πίνακες αναζήτησης LUT, με αυτό το τρόπο υλοποιείται η λογική, Εικόνα 3.29. Θα ισχυριζόταν κανείς να πει ότι μια φέτα θα μπορούσε να αποτελεί ένα λογικό στοιχείο, αλλά αυτό δεν υφίσταται όπως είδαμε παραπάνω, άλλωστε αυτό είναι τεκμηριωμένο και από την κατασκευάστρια εταιρία στα εγχειρίδιά της [Xln1], [Xln2].
- Οι συσκευές AT40K και Cyclone έχουν αριθμητικές και λογικές λειτουργίες, για αριθμητικές λειτουργίες χρησιμοποιούν τις δομές που φαίνονται στις Εικόνες 3.29 και 3.48. Για λογικές (κανονικές λειτουργίες) χρησιμοποιούν τις δομές που φαίνονται στις Εικόνες 3.17 και 3.47. Η αρχιτεκτονική Spartan-3 αυτές τις δύο λειτουργίες, τις έχει προσαρμόσει στις "φέτες" (Slices) με τα πρότυπα SLICEM και SLICEL. Το μεν πρώτο SLICEM για αριθμητικές και άλλες υπολογιστικές λειτουργίες και το δεύτερο SLICEL για την υλοποίηση των λογικών συναρτήσεων, Εικόνα 3.29. Δεν υπάρχει μεγάλη διαφορά στη λογική της υλοποίησης αυτών των δομών, απλά η Xilinx προχώρησε λίγο περισσότερο στο κομμάτι αυτό σε σύγκριση με τις άλλες δύο εταιρίες.
- Η αρχιτεκτονική της συσκευής AT40K δεν έχει λογική βαθμίδα (LB) όπως οι άλλες δύο εταιρίες τα LAB και τα CLB, Εικόνα 3.14 και 3.27 αντίστοιχα. Η δομή της σε σχέση με την ιδιόμορφη διασύνδεση της, την καθιστά από μόνη της μια πολύπλοκη λογική βαθμίδα, Εικόνα 3.45. Παράλληλα αν κοιτάξει κανείς και τις δομές διασύνδεσης των ακροδεκτών της συσκευής με τα λογικά στοιχεία (Cells) θα καταλήξει στο ίδιο συμπέρασμα, Εικόνες 3.57 και 3.58.

Τροφοδοσία

- Η τάση τροφοδοσίας των συσκευών είναι, για AT40K 3V/5V, για Spartan-3 είναι 1,4V μέχρι 3,4V (υπάρχει ποικιλία) και η συσκευή Cyclone λειτουργεί στα 1,5V.

Αριθμητικές Μονάδες

- Πολλαπλασιαστές (Multipliers) στην αρχιτεκτονική των συσκευών Spartan-3 είναι επιπλέον λογικές μονάδες, Εικόνες 3.40 και 3.41. Είναι ανεξάρτητες λειτουργικά σε σχέση με τους πολλαπλασιαστές που επιτελούν τα SLICEM. Καμία άλλη αρχιτεκτονική δεν έχει κάποια αντίστοιχη λογική μονάδα όπως η αρχιτεκτονική της Spartan-3. Οι άλλες δύο αρχιτεκτονικές υλοποιούν πολλαπλασιαστές στις αριθμητικές λειτουργίες των λογικών στοιχείων. Θα πρέπει εδώ να αναφέρουμε ότι η εταιρία Atmel ισχυρίζεται ότι με την ιδιόμορφη διασύνδεση και λογική που παρέχει δεν χρειάζεται να υπάρχει επιπλέον λογική μονάδα πολλαπλασιαστή, Εικόνα 3.49. Η διασύνδεση που προσφέρει στην αρχιτεκτονική της, υπόσχεται γρήγορους και αξιόπιστους υπολογισμούς.

Μνήμες

- Όλες οι αρχιτεκτονικές επιτελούν λειτουργίες μονής και διπλή θύρας (Single and Dual Port) προσπέλαση στη μνήμη RAM. Αλλά μόνο η αρχιτεκτονική της Cyclone υπόσχεται και αληθής διπλής θύρας προσπέλαση (True Dual Port), Εικόνες 3.19, 3.31 και 3.53. Όπως προαναφέραμε η λειτουργία True Dual Port επιτελεί ταυτόχρονη προσπέλαση σε δύο εγγραφές και δυο αναγνώσεις στον ίδιο παλμό ρολογιού. Ή μπορούμε να έχουμε ταυτόχρονη προσπέλαση σε μια

εγγραφή και μια ανάγνωση, αλλά σε διαφορετικές συχνότητες ρολογιού. Επίσης όλες οι αρχιτεκτονικές χρησιμοποιούν τα λογικά στοιχεία τους ως επιπλέον μνήμη.

Διαχείριση Ρολογιών

- Και στις τρεις αρχιτεκτονικές παρέχονται 8 γενικά ρολόγια για παροχή και διανομή σε όλη τη δομή. Η συσκευή AT40K παρέχει 4 επιπλέον γρήγορα ρολόγια (fast clock) τα FCK[1..4] δύο για κάθε άκρο.
- Στην παροχή διευθυντών ρολογιού (DCM, Digital Clock Manager) υπάρχει μια διαφορά. Η αρχιτεκτονική Spartan-3 παρέχει από δυο μέχρι τέσσερις διευθυντές ρολογιού, Εικόνα 3.36. Η αρχιτεκτονική Cyclone παρέχει έναν αλλά λόγω της διαρρύθμισης της με ένα η δύο PLL Εικόνα 3.21, μπορεί να πει κάνεις η να χαρακτηρίσει τα PLL και ως διευθυντές ρολογιού. Η αρχιτεκτονική δομή AT40K έχει έναν διευθυντή ρολογιού. Η μεγάλη διαφορά οφείλεται βέβαια στο διαφορετικό πλήθος των λογικών στοιχείων που βρίσκονται σε κάθε συσκευή. Κατά την άποψη μας είναι αναμενόμενη αυτή η διαφορά και λογική, δεν μπορεί να πει κανείς λοιπόν ότι κάποια αρχιτεκτονική υστερεί επί τούτου.

Είδαμε λοιπόν κάποιες διαφορές των αρχιτεκτονικών σε σχέση με τη γενική δομή που διέπει μια αρχιτεκτονική FPGA. Σε καμία περίπτωση όμως δεν είδαμε αρχιτεκτονική που να έχει μια τελείως διαφορετική φιλοσοφία. Υπήρξαν λοιπόν αναμενόμενες διαφορές με πολύ μικρές εκπλήξεις.

Κλείνοντας το κεφάλαιο αυτό θα θέλαμε να αναφέρουμε ότι είδαμε αρχιτεκτονικές δομές FPGA με ιδιαίτερο ενδιαφέρον και προσοχή. Θελήσαμε να μελετήσουμε τα κύρια σημεία αυτών χωρίς να εμβαθύνουμε πολύ κάνοντας τη μεταπτυχιακή διατριβή ένα τεχνικό εγχειρίδιο. Πιστεύουμε ότι καταφέραμε να αναδείξουμε κάποια καίρια σημεία των αρχιτεκτονικών δομών, τα οποία να σας έκαναν να διαπιστώσετε από μονοί σας τις όποιες διαφορές και τις όποιες ομοιότητες.

Στη συνέχεια των κεφαλαίων θα γίνει παρουσίαση των εργαλείων σχεδίασης CAD (Κεφάλαιο 4) των παραπάνω συσκευών και την εκτέλεση των πρότυπων προγραμμάτων VHDL σε κάθε ένα πρόγραμμα σχεδίασης CAD (Κεφάλαιο 5).

4. Εργαλεία Σχεδίασης CAD για FPGAs

Σε αυτό το κεφάλαιο θα δούμε πως μπορούμε να χρησιμοποιήσουμε τα εργαλεία σχεδίασης CAD των εταιριών Altera *Quartus II*, Xilinx *ISE* και Atmel *ISD – Figaro*, για να σχεδιάσουμε στις συσκευές Cyclone, Spartan-3 και AT40K αντίστοιχα. Τα συγκεκριμένα εργαλεία μπορούν να προγραμματίσουν όλη την ποικιλία των συσκευών FPGA που παρέχει η κάθε εταιρία ξεχωριστά. Στην Ενότητα 4.1 θα κάνουμε μία εισαγωγή στα εργαλεία CAD για το πώς αυτά λειτουργούν και πως αυτά χρησιμοποιούνται για το προγραμματισμό των συσκευών FPGA. Στην Ενότητα 4.2 θα εξετάσουμε το εργαλείο σχεδίασης Quartus II, στην Ενότητα 4.3 το εργαλείο σχεδίασης ISE και τέλος στην Ενότητα 4.4 το εργαλείο σχεδίασης ISD – Figaro.

Δεν θα επισέλθουμε σε λεπτομέρειες που αφορούν τη χρήση των εργαλείων σχεδίασης CAD και το πώς αυτά λειτουργούν, αλλά μόνο στο πως θα μπορούσαμε να διαχειριστούμε ένα κύκλωμα το οποίο έχει περιγραφεί σε γλώσσα VHDL. Για περισσότερες λεπτομέρειες μπορείτε να ανατρέξετε στις αναφορές, όπου παραθέτονται τα εγχειρίδια χρήσης των εργαλείων αυτών.

4.1. Εισαγωγή στα εργαλεία σχεδίασης CAD (Computer Aided Design)

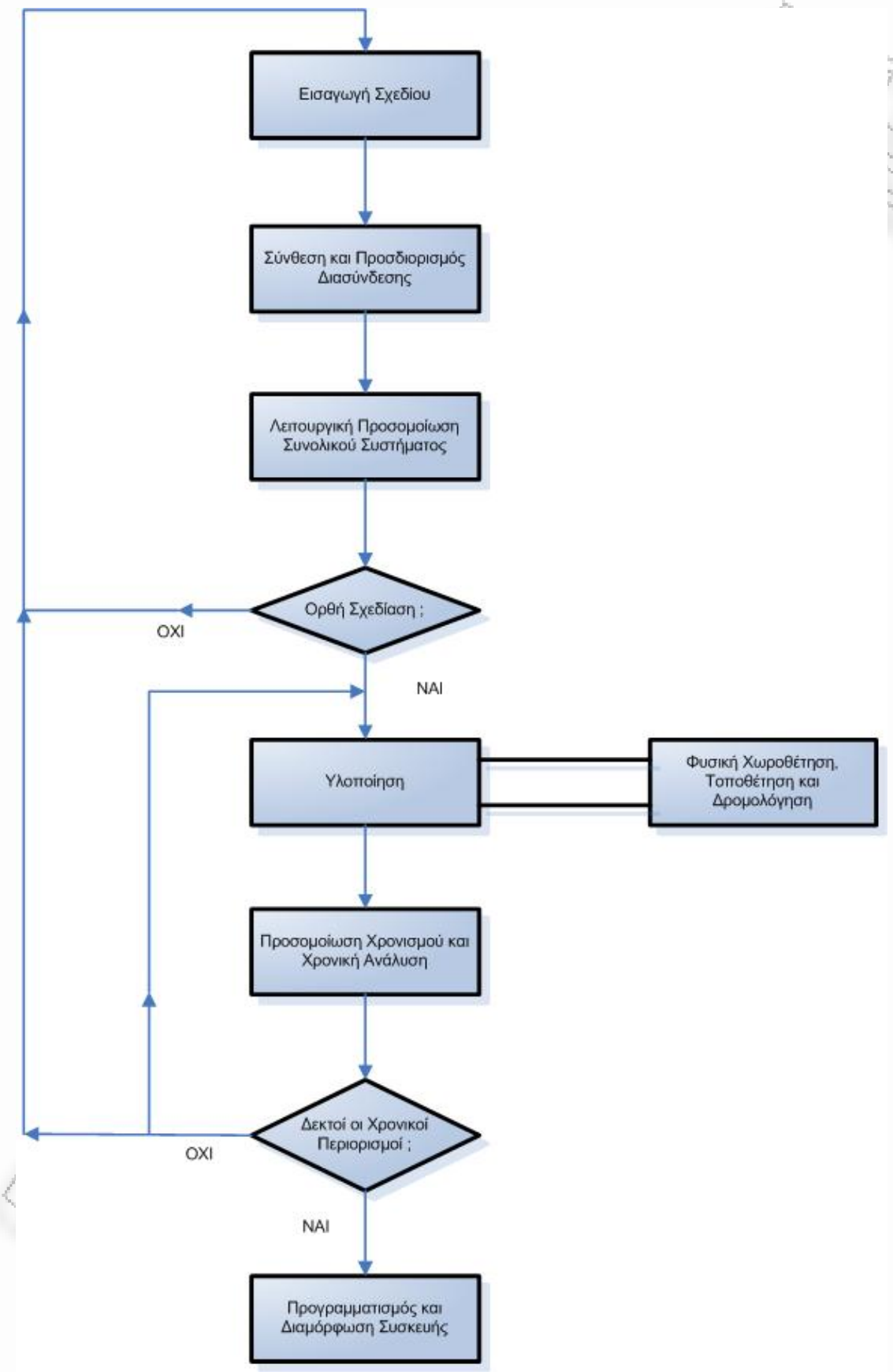
4.1.1. Τυπική σχεδίαση σε εργαλείο CAD για συσκευές FPGAs

Η τυπική ροή σχεδίασης ενός ψηφιακού κυκλώματος με τη χρήση εργαλείων σχεδίασης CAD για συσκευές FPGA φαίνεται στην Εικόνα 4.1. Τα βήματα είναι τα εξής :

- Εισαγωγή του σχεδίου (*Design Entry*), η οποία μπορεί να γίνει με διάφορους τρόπους τους οποίους θα αναφέρουμε παρακάτω.
- Σύνθεση και προσδιορισμός διασύνδεσης (*Synthesis*), σε αυτό το βήμα έχουμε τη σύνθεση των λογικών βαθμίδων που θέλουμε να υλοποιήσουμε. Αυτό μπορεί να γίνει από ένα ανεξάρτητο κατασκευαστή λογισμικού ή από τον κατασκευαστή του FPGA. Σε αυτό το στάδιο έχουμε την περιγραφή της συμπεριφοράς (*Behavioral Description*) της σχεδίασης η οποία μεταφράζεται σε περιγραφή δομής (*Structural Netlist*). Ο μεταφραστής (*Netlist Translators*) μεταφράζει την περιγραφή δομής σε μορφή που υποστηρίζει το εργαλείο του κατασκευαστή.
- Λειτουργική προσομοίωση συνολικού συστήματος (*Functional Simulation*), εδώ έχουμε την πιστοποίηση της ορθότητας της λογικής σχεδίασης πριν υλοποιηθεί. Εκτελείται πάντα στα πρώτα στάδια της σχεδίασης είτε σε περιγραφή συμπεριφοράς είτε σε περιγραφή δομής.
- Αν η σχεδίαση που επιθυμούμε είναι σωστή τότε συνεχίζουμε στα επόμενα βήματα, αν δεν είναι τότε επανερχόμαστε στο πρώτο στάδιο και ελέγχουμε την ορθότητα της σχεδίασης μας ή αναθεωρούμε τη σχεδίαση με μια νέα.
- Υλοποίηση (*Implementation*), σε αυτό το βήμα εκτελούνται τρεις επιμέρους λειτουργίες. Η μία έχει να κάνει με τη φυσική χωροθέτηση ή αντιστοίχιση (*Fitting* ή *Mapping*) όπου τεμαχίζονται οι συναρτήσεις στα λογικά στοιχεία, δηλαδή πως υλοποιούνται οι λογικές λειτουργίες στο εσωτερικό ενός λογικού στοιχείου. Η δεύτερη λειτουργία έχει να κάνει με την τοποθέτηση (*Placement*), δηλαδή που τοποθετούμε κάθε κομμάτι λογικής στη διάταξη των λογικών στοιχείων. Και τέλος η τρίτη λειτουργία έχει να κάνει με τη δρομολόγηση (*Routing*), τη διασύνδεση των λογικών στοιχείων μεταξύ τους, δηλαδή ποιες καλωδιώσεις χρησιμοποιούνται για να συνδεθούν τα λογικά στοιχεία μεταξύ τους καθώς και με τους ακροδέκτες.
- Προσομοίωση χρονισμού (*Timing Simulation*) και Χρονική ανάλυση (*Timing Analysis*), το μεν πρώτο εκτελείται μετά την υλοποίηση και πιστοποιεί ότι η σχεδίαση λειτουργεί στην επιθυμητή ταχύτητα. Καθορίζει τα κρίσιμα μονοπάτια κάτω από τις χειρότερες συνθήκες και ανιχνεύει χρονικές παραβάσεις (setup time and hold time violations). Το δε δεύτερο εκτελείται και αυτό μετά την υλοποίηση και υπολογίζει την καθυστέρηση των μονοπατιών του κυκλώματος. Πιστοποιεί ότι η σχεδίαση ικανοποιεί τις χρονικές προδιαγραφές, πληροφορεί το σχεδιαστή για «αργά» μονοπάτια και προσφέρει υλικό για την τεκμηρίωση της σχεδίασης.
- Αν η σχεδίαση πληρεί τους χρονικούς περιορισμούς και τις χρονικές προδιαγραφές τότε συνεχίζουμε στο τελευταίο βήμα, αν όχι τότε : ή επανερχόμαστε στο πρώτο στάδιο και ελέγχουμε την ορθότητα της σχεδίασης μας, ακόμη και να αναθεωρήσουμε τη σχεδίαση με μια

νέα, ή επανερχόμαστε στο στάδιο της υλοποίησης και επιδιώκουμε επαναδιαμόρφωση και νέα υλοποίηση. Το τελευταίο μπορεί να γίνει είτε αυτόματα από το εργαλείο σχεδίασης είτε χειροκίνητα από τον ίδιο το χρήστη.

- Προγραμματισμός και διαμόρφωση συσκευής (*Programming and Configuration*), το τελευταίο βήμα είναι αυτό που θα παράγει το αρχείο BitStream με το οποίο θα διαμορφωθεί και θα προγραμματιστεί η συσκευή FPGA.



Εικόνα 4.1 Τυπική μεθοδολογία σχεδίασης με χρήση εργαλείων CAD για συσκευές FPGA

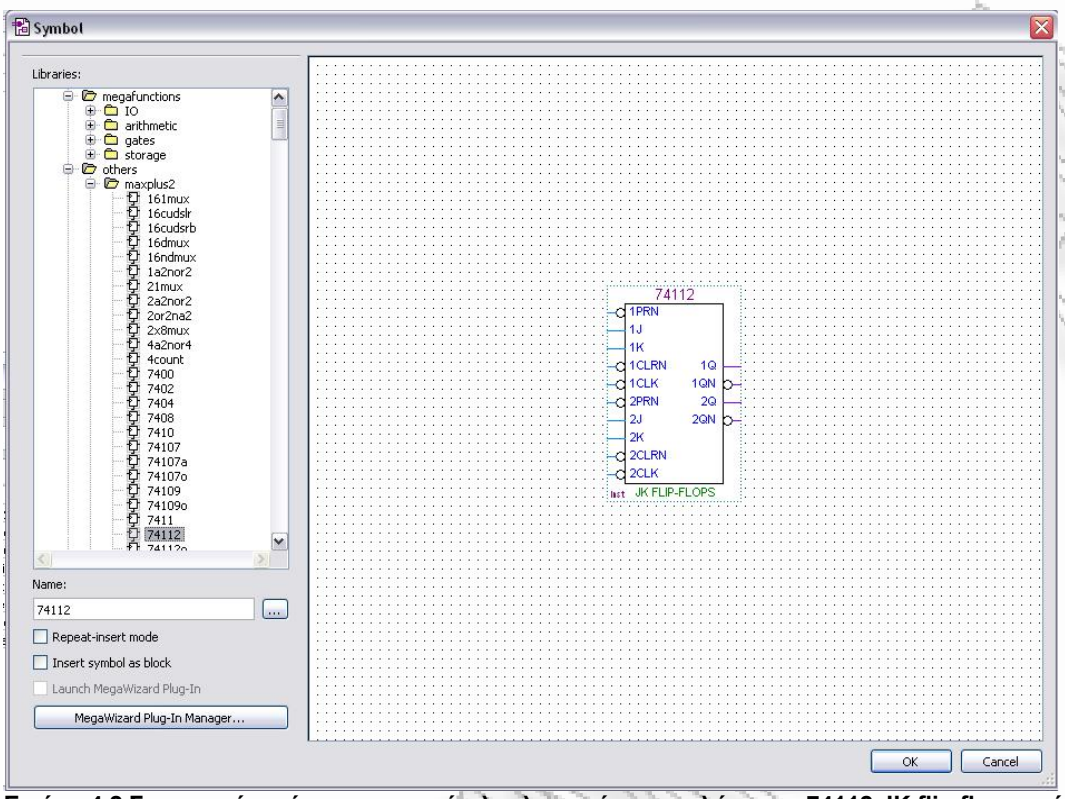
Θεωρούμε ότι η διαδικασία ανάπτυξης στοχεύει στην παραγωγή ενός προϊόντος που πληρεί κάποιες προδιαγραφές. Οι πιο προφανείς απαιτήσεις είναι ότι το προϊόν θα πρέπει να λειτουργεί σωστά και ότι θα πρέπει να έχει κάποιο συγκεκριμένο βαθμό απόδοσης. Για να γίνει ένας πιο λεπτομερής έλεγχος των προδιαγραφών αυτών δεν αρκεί μόνο η μεθοδολογία σχεδίασης που είδαμε παραπάνω. Θα πρέπει μετά το προγραμματισμό και τη διαμόρφωση της συσκευής, να γίνει έλεγχος και της συσκευής FPGA σε πραγματικές συνθήκες αν όντως πληρεί της προδιαγραφές αυτές. Αυτή η διαδικασία ονομάζεται *πιστοποίηση στο κύκλωμα* (*in–circuit verification*) και πιστοποιεί ότι η σχεδίαση λειτουργεί σωστά στην τελική εφαρμογή, κάτω από πραγματικές συνθήκες λειτουργίας. Οι κατασκευαστές παρέχουν εργαλεία για να βοηθήσουν την πιστοποίηση στο κύκλωμα.

4.1.2. Τρόποι σχεδίασης στα εργαλεία CAD

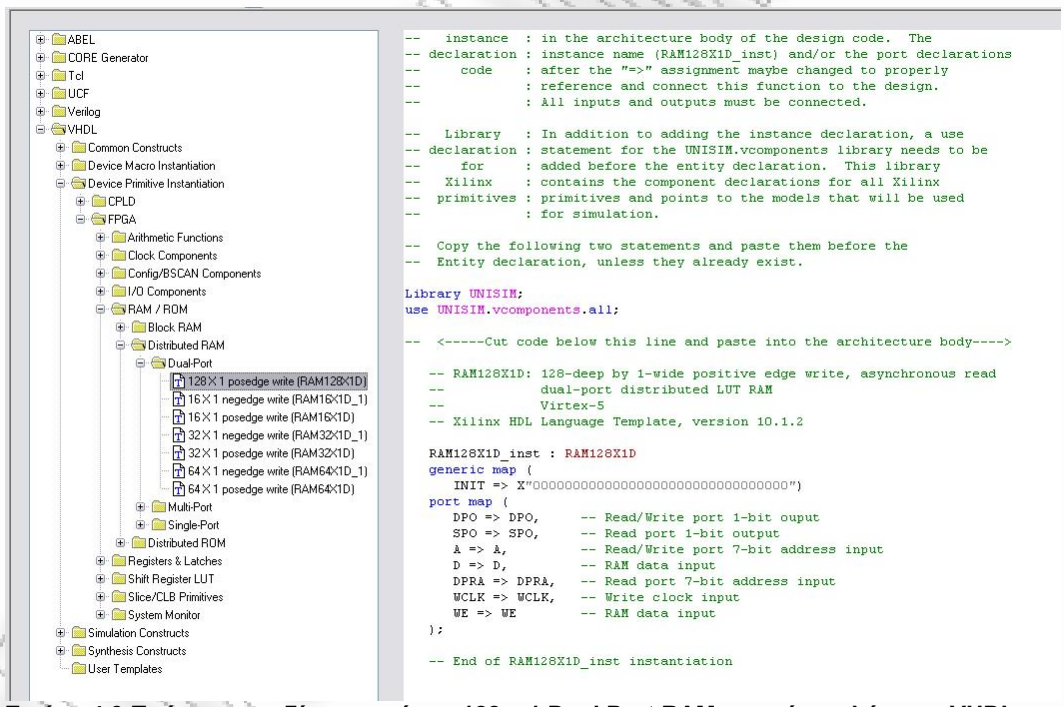
Σε κάθε εργαλείο σχεδίασης CAD για την εξυπηρέτηση του χρήστη υπάρχει μια ποικιλία από εναλλακτικούς τρόπους σχεδίασης, κάποια από αυτά είναι:

- Σύνθεση προγραμμάτων με χρήση έτοιμων σχηματικών προτύπων και ονομάζεται *σχηματική εισαγωγή* (*Schematic Entry*).
- Σύνθεση προγραμμάτων με χρήση γλωσσών HDL όπως Verilog και VHDL.
- Σύνθεση προγραμμάτων μέσω άλλων λογικών μονάδων που ήδη έχουν χρησιμοποιηθεί και κατασκευαστεί. Αυτά συνήθως τα δημιουργεί ο κάθε χρηστής ξεχωριστά σύμφωνα πάντα με τις ανάγκες του αλλά και την ειδίκευση κατασκευής που κάνει.

Και στα τρία πρότυπα υπάρχουν βιβλιοθήκες μέσα στις οποίες υπάρχουν έτοιμα σχεδιαστικά πρότυπα όπως πολυπλέκτες, λογικές πύλες, μνήμες RAM, flip flop και άλλα λογικά κυκλώματα, τα οποία μπορεί ο χρήστης να χρησιμοποιήσει και να δημιουργήσει το λογικό κύκλωμα της επιθυμίας του. Στο μεν σχηματικό υπάρχουν σχηματικά πρότυπα, Εικόνα 4.2, ενώ στη χρήση με γλώσσες HDL υπάρχουν πρότυπα σχεδίασης με χρήση κώδικα (προγράμματα), Εικόνα 4.3. Στην τελευταία μέθοδο πάλι, το κάθε λογισμικό παρέχει μια ξεχωριστή βιβλιοθήκη στην οποία μπορεί ο χρήστης να αποθηκεύει τα πρότυπα που ο ίδιος δημιουργεί. Στις δυο πρώτες μεθόδους τα πρότυπα των βιβλιοθηκών είναι δημιούργημα των εταιριών κατασκευής των λογισμικών CAD, ενώ στην τρίτη μέθοδο τα πρότυπα είναι δημιούργημα (κατασκευή) του εκάστοτε χρήστη. Και τα τρία εργαλεία που θα εξετάσουμε έχουν υποδομές για χρήση και των τριών μεθόδων σχεδίασης. Επιπλέον παρέχουν υποδομές για τη χρήση των γλωσσών Verilog και VHDL. Το κάθε λογικό κύκλωμα που θα δημιουργηθεί από αυτές τις τρεις μεθόδους, θα εκτελεστεί για να παράγει το αρχείο BitStream με το οποίο θα προγραμματιστεί, μέσω της θύρας JTAG, Εικόνα 2.10.



Εικόνα 4.2 Σχηματικό πρότυπο τυπικού ολοκληρωμένου κυκλώματος 74112 JK flip flop, από τη βιβλιοθήκη του εργαλείου σχεδίασης Quartus II της εταιρίας Altera



Εικόνα 4.3 Πρότυπο σχεδίασης μνήμης 128 x 1 Dual Port RAM με χρήση γλώσσας VHDL, από τη βιβλιοθήκη του εργαλείου σχεδίασης ISE της εταιρίας Xilinx

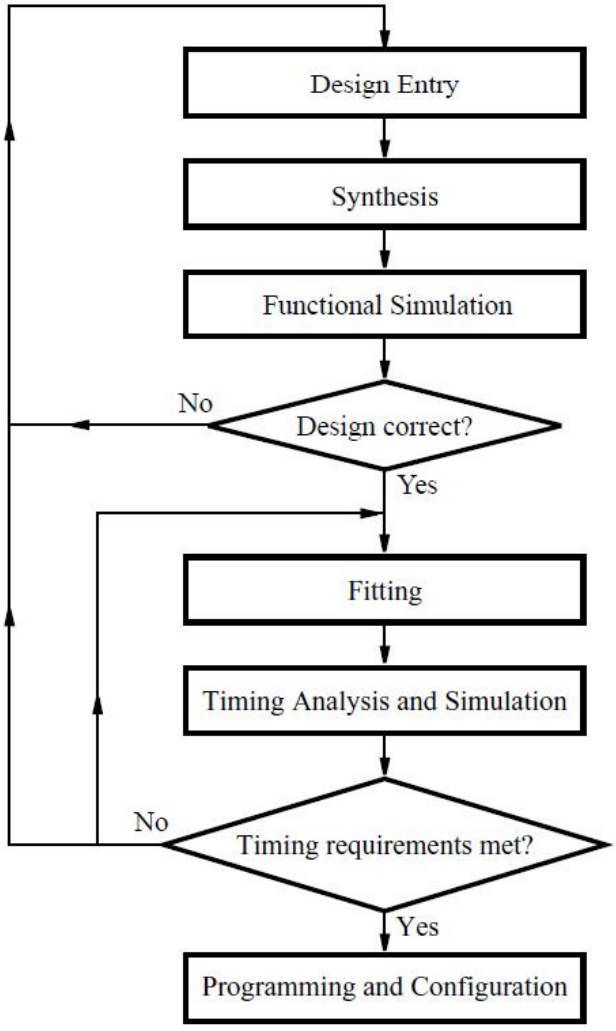
Εμείς στην ανάπτυξη των εργαλείων σχεδίασης CAD που θα κάνουμε θα περιοριστούμε στο τρόπο δημιουργίας αρχείων και έργων (project) με τη γλώσσα περιγραφής υλικού VHDL. Συγκεκριμένα θα δούμε πως μπορούμε να εισάγουμε μοντέλα VHDL, πως τα επεξεργαζόμαστε και πως δημιουργούμε το αρχείο BitStream.

4.2. Σχεδίαση με χρήση εργαλείου Altera Quartus II

4.2.1. Εισαγωγή στο Quartus II

Σε αυτή την ενότητα θα εξετάσουμε το εργαλείο σχεδίασης Quartus II και συγκεκριμένα την έκδοση 11.0. Το σενάριο που θα ακολουθήσουμε θα είναι το ίδιο και για τα 3 εργαλεία σχεδίασης, δηλαδή θα δημιουργήσουμε ένα αρχείο με τη γλώσσα VHDL και θα το επεξεργαστούμε μέχρι την παραγωγή του αρχείου bitstream. Θα πάρουμε για υλοποίηση προτεινόμενα προγράμματα VHDL που είναι ενσωματωμένα στα εγχειρίδια εκμάθησης (tutorial) του κάθε CAD.

Η γενική μεθοδολογία σχεδίασης που προτείνεται στο Quartus II είναι αυτό της Εικόνας 4.4. Όπως μπορείτε να δείτε η μεθοδολογία σχεδίασης ταιριάζει απόλυτα με αυτή που είδαμε στην Εικόνα 4.1 της Ενότητας 4.1. Στο σημείο *Fitting* της Εικόνας 4.4 έχουμε το στάδιο της υλοποίησης της Εικόνας 4.1.

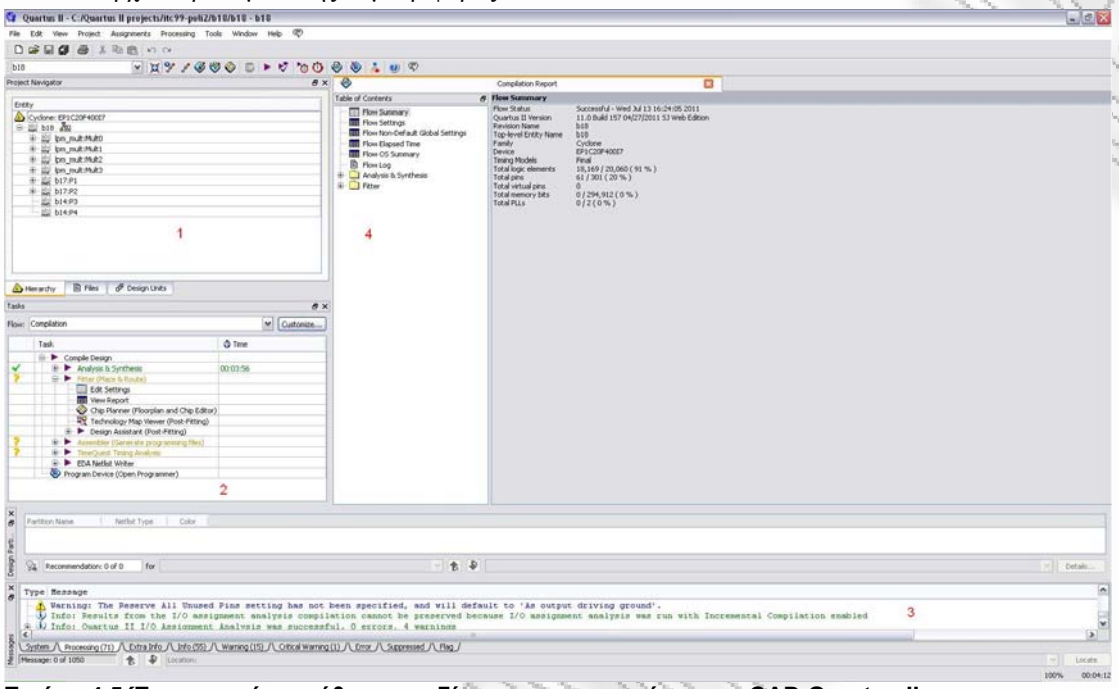


Εικόνα 4.4 [Alt2] Γενική μεθοδολογία σχεδίασης στο πρόγραμμα CAD Quartus II

Το περιβάλλον εργασίας του Quartus II είναι αυτό που εμφανίζεται στην Εικόνα 4.5 το οποίο αποτελείται από τέσσερα υποπαράθυρα τα οποία είναι τα εξής :

1. Το παράθυρο *Project Navigator*, το οποίο αποτελείται από τρεις ετικέτες που παρέχουν πληροφορίες για το χρηστή σε σχέση με τα αρχεία που χρησιμοποιούνται στο κάθε project.
2. Το παράθυρο *Tasks*, το οποίο περιέχει τις διαδικασίες τις οποίες μπορεί να εκτελέσει ο χρήστης και αλλάζει το περιεχόμενο της ανάλογα με το αρχείο που εκτελείται εκείνη τη στιγμή.
3. Το παράθυρο *Messages*, το οποίο μας δείχνει τα τυχόν λάθη ή μηνύματα που θα προκύψουν κατά τη διάρκεια της σχεδίασης

4. Και τέλος το τέταρτο παράθυρο είναι αυτό στη μέση το οποίο χρησιμοποιείται για ανάγνωση και εγγραφή σε ποικιλία αρχείων που βρίσκονται μέσα στο project όπως αναφορές, αρχεία VHDL, αρχεία προσομοίωσης συμπεριφοράς κτλ.



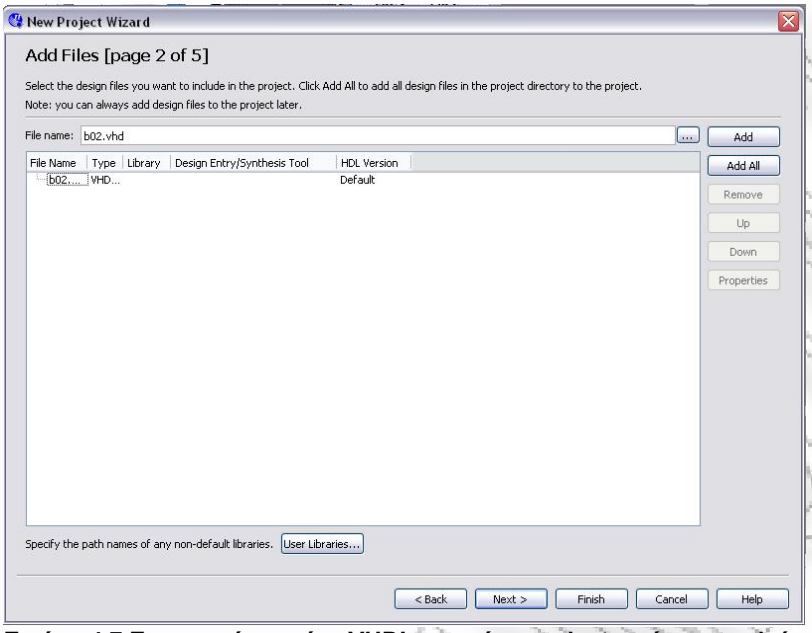
Εικόνα 4.5 Ένα τυπικό παράθυρο σχεδίασης του προγράμματος CAD Quartus II

4.2.2. Δημιουργία νέου Project

Το πρώτο που πρέπει να κάνουμε όταν θέλουμε να σχεδιάσουμε ένα πρόγραμμα είναι να δημιουργήσουμε ένα έργο (Project), αυτό το πετυχαίνουμε πηγαίνοντας *Open→New Project Wizard*, στο νέο παράθυρο διαλόγου πατάμε Next και στο επόμενο διαλέγουμε ένα όνομα για το νέο project που θέλουμε να δημιουργήσουμε. Έστω είναι το *addersubtractor*, αν μας προτρέψει το πρόγραμμα να διαλέξουμε νέο κατάλογο το κάνουμε από την πρώτη επιλογή, Εικόνα 4.6.

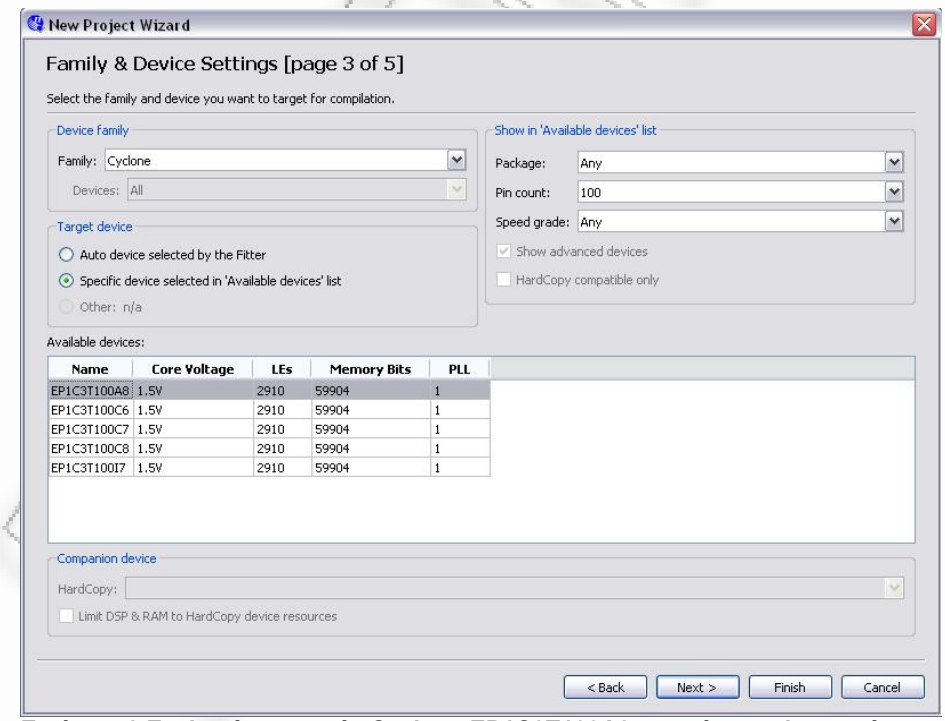
Εικόνα 4.6 Δημιουργία νέου project από το εργαλείο σχεδίασης Quartus II

Αφού διευθετήσουμε το όνομα και το κατάλογο προορισμού πατάμε Next. Στο επόμενο παράθυρο διαλόγου, αν θα θέλαμε να εισάγουμε ένα ήδη υπάρχον αρχείο με πρόγραμμα VHDL που έχουμε δημιουργήσει, θα επιλέξουμε να πατήσουμε το κουμπί Add, Εικόνα 4.7. Αν δεν έχουμε κάποιο έτοιμο αρχείο αλλά θέλουμε να δημιουργήσουμε ένα καινούργιο (όπως στην περίπτωση μας) αρκεί να πατήσουμε το κουμπί Next. Και στις δυο περιπτώσεις πατώντας το Next θα μας βγάλει το επόμενο παράθυρο διαλόγου το οποίο είναι παρόμοιο με αυτό της Εικόνας 4.8.



Εικόνα 4.7 Εισαγωγή αρχείου VHDL στο νέου project από το εργαλείο σχεδίασης Quartus II

Στο παράθυρο διαλόγου της Εικόνας 4.8 διαλέγουμε τη συσκευή της επιθυμίας μας. Εμείς θα επιλέξουμε Cyclone και συγκεκριμένα τη συσκευή *EP1C3T100A8* με ακροδέκτες εισόδου (*pin cou*nt) 100.

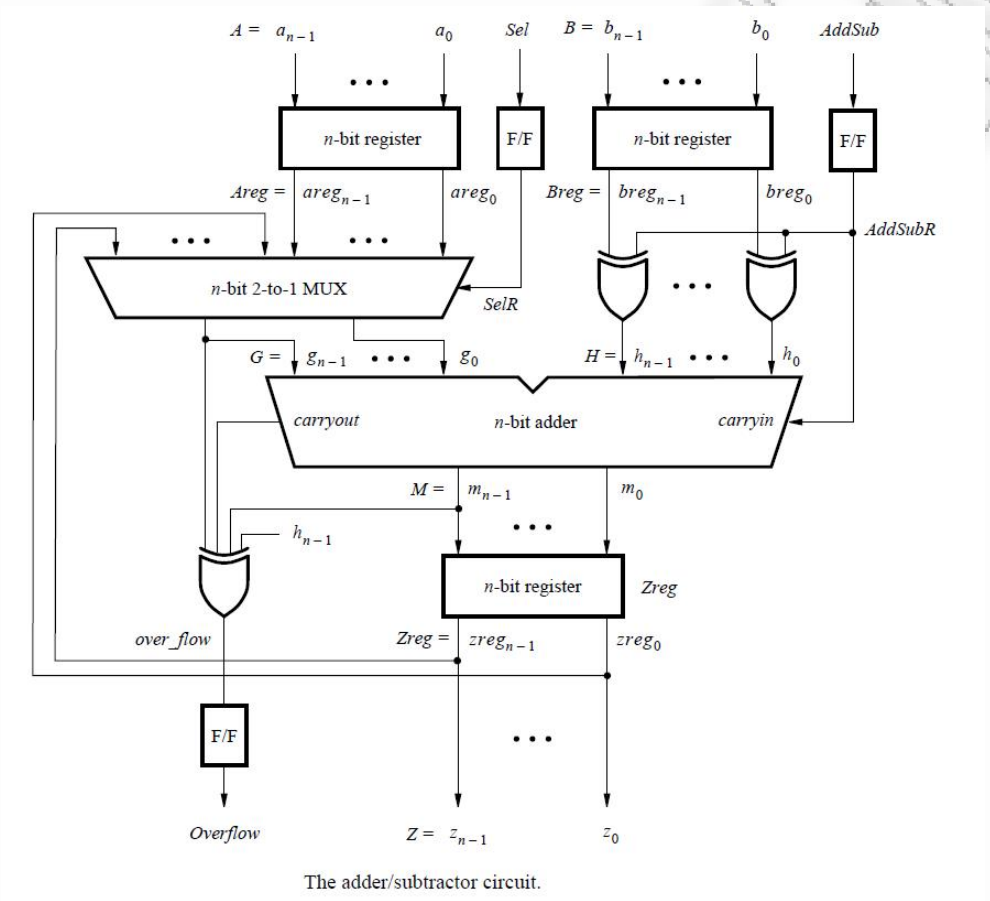


Εικόνα 4.8 Επιλογή συσκευής Cyclone EP1C3T100A8 στο νέου project από το εργαλείο σχεδίασης Quartus II

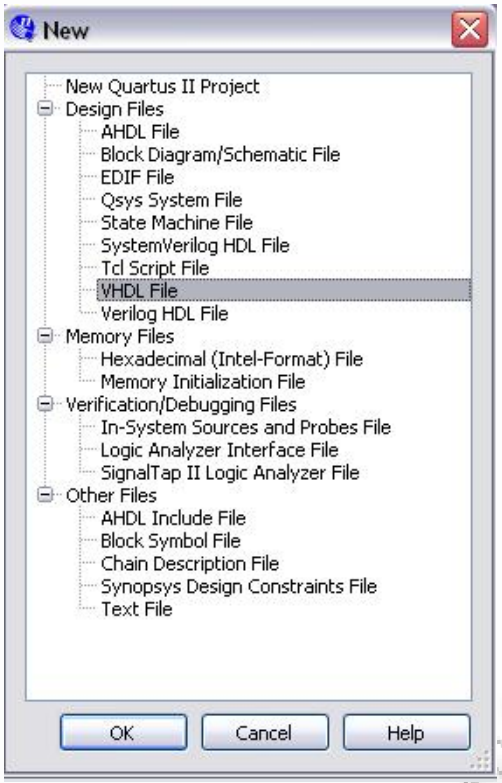
Στα επόμενα δυο παράθυρα πατάμε Next και Finish και το πρόγραμμα θα ανοίξει για να προχωρήσουμε στην υλοποίηση.

4.2.3. Σχεδίαση με γλώσσα VHDL

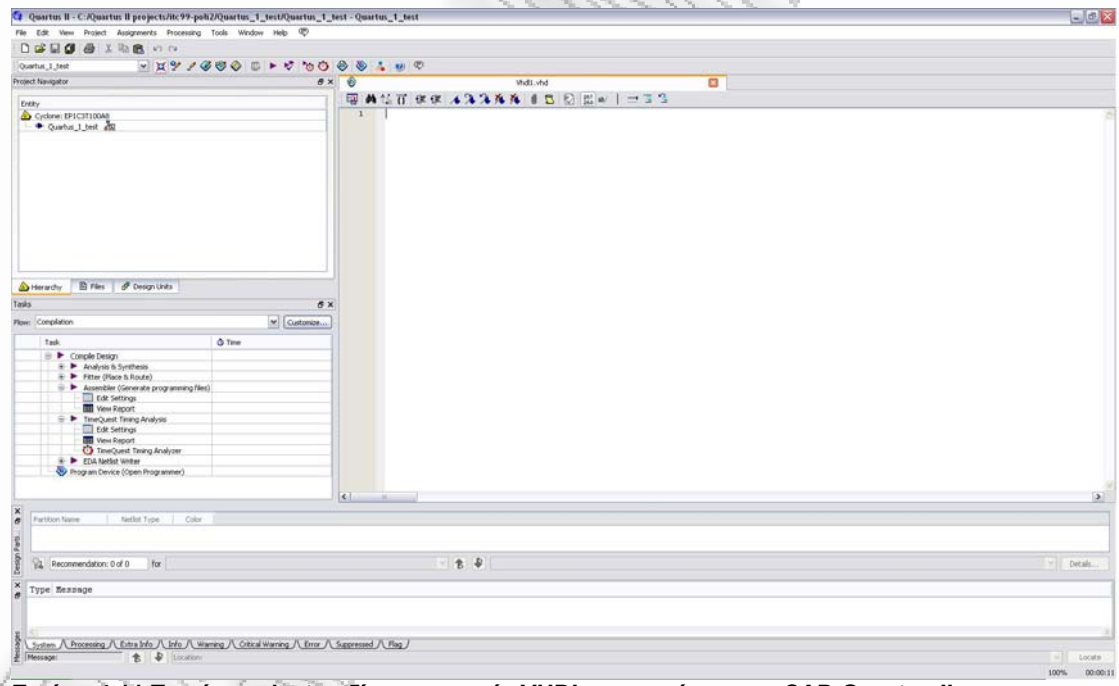
Όπως προαναφέραμε θα δημιουργήσουμε ένα αρχείο VHDL το οποίο και θα εκτελέσουμε. Σε αυτή την ενότητα θα ακολουθήσουμε το προτεινόμενο πρόγραμμα που βρίσκεται στον οδηγό χρήσης του εργαλείου Quartus II, το οποίο είναι ένας *αθροιστής / αφαιρέτης* (*adder / subtractor)* των 8bit ο οποίος φαίνεται σχηματικά στην Εικόνα 4.9. Για μπορέσομε να δημιουργήσουμε το αρχείο VHDL αρκεί να πάμε *File→New* και από το νέο παράθυρο διαλόγου να επιλέξουμε *VHDL File*, Εικόνα 4.10. Αφού επιλέξουμε τη δημιουργία του αρχείου θα πρέπει στο περιβάλλον εργασίας του Quartus να φαίνεται ένας κειμενογράφος στον οποίο θα μπορούμε να γράψουμε κώδικα VHDL, Εικόνα 4.11.



Εικόνα 4.9 [Alt2] Σχηματική αναπαράσταση του αθροιστή αφαιρέτη (adder / subtractor) των 8bit



Εικόνα 4.10 Δημιουργία αρχείου VHDL στο πρόγραμμα CAD Quartus II



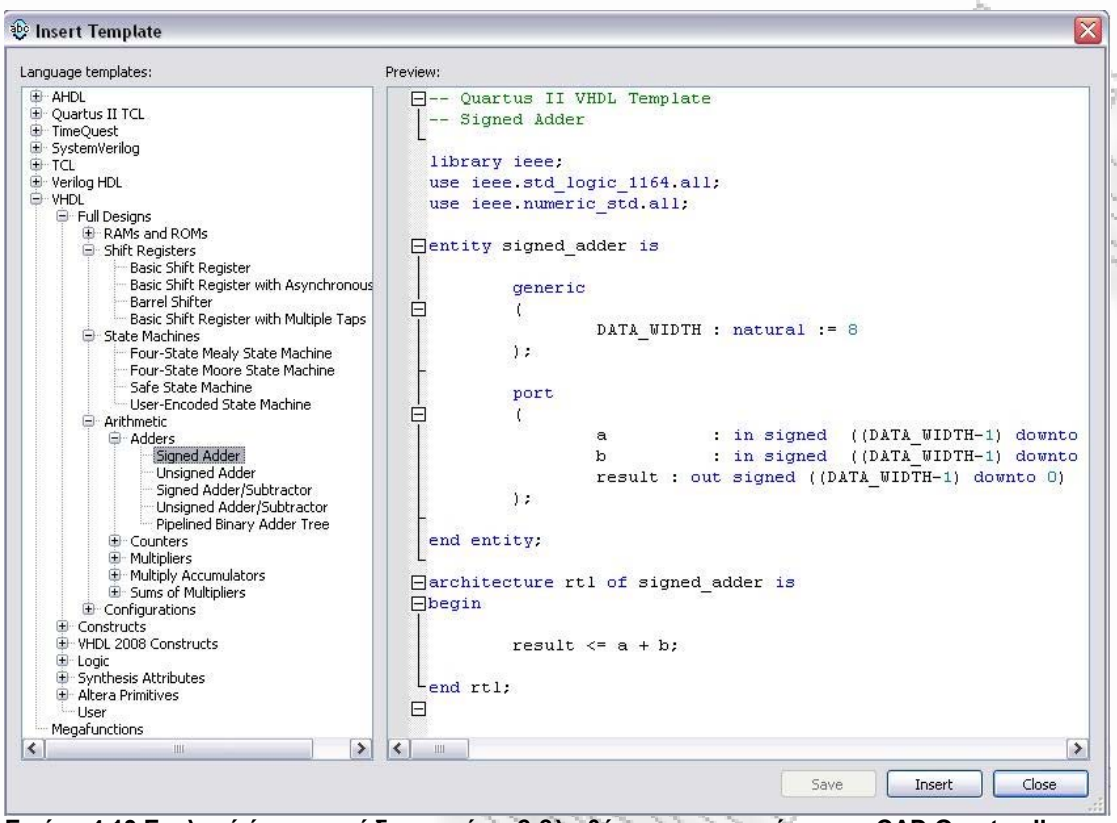
Εικόνα 4.11 Το νέο project μαζί με το αρχείο VHDL στο πρόγραμμα CAD Quartus II

Κατόπιν όλων αυτών πρέπει να γράψουμε το κώδικα που θα υλοποιήσει τον αθροιστή / αφαιρέτη. Ενδεικτικό κομμάτι του κώδικα φαίνεται στην Εικόνα 4.12 και ο πλήρης κώδικας είναι επισυναπτόμενος στο παράρτημα Β με την ονομασία *addersubtractor.vhd*.

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

-- Top-level entity
ENTITY addersubtractor IS
  GENERIC ( n :  INTEGER := 16 ) ;
  PORT ( A, B                      :  IN STD_LOGIC_VECTOR(n-1 DOWNTO 0) ;
         Clock, Reset, Sel, AddSub :  IN STD_LOGIC ;
         Z                         :  BUFFER STD_LOGIC_VECTOR(n-1 DOWNTO 0) ;
         Overflow                  :  OUT STD_LOGIC ) ;
END addersubtractor ;

ARCHITECTURE Behavior OF addersubtractor IS
  SIGNAL G, H, M, Areg, Breg, Zreg, AddSubR_n :  STD_LOGIC_VECTOR(n-1 DOWNTO 0) ;
  SIGNAL SelR, AddSubR, carryout, over_flow :  STD_LOGIC ;
  COMPONENT mux2to1
    GENERIC ( k :  INTEGER := 8 ) ;
    PORT ( V, W :  IN STD_LOGIC_VECTOR(k-1 DOWNTO 0) ;
           Sel  :  IN STD_LOGIC ;
           F    :  OUT STD_LOGIC_VECTOR(k-1 DOWNTO 0) ) ;
  END COMPONENT ;
  COMPONENT adderk
    GENERIC ( k :  INTEGER := 8 ) ;
    PORT ( carryin  :  IN STD_LOGIC ;
           X, Y     :  IN STD_LOGIC_VECTOR(k-1 DOWNTO 0) ;
           S        :  OUT STD_LOGIC_VECTOR(k-1 DOWNTO 0) ;
           carryout :  OUT STD_LOGIC ) ;
  END COMPONENT ;
BEGIN
  PROCESS ( Reset, Clock )
  BEGIN
    IF Reset = '1' THEN
      Areg <= (OTHERS => '0'); Breg <= (OTHERS => '0');
      Zreg <= (OTHERS => '0'); SelR <= '0'; AddSubR <='0'; Overflow <= '0';
    ELSIF Clock'EVENT AND Clock = '1' THEN
      Areg <= A; Breg <= B; Zreg <= M;
      SelR <= Sel; AddSubR <= AddSub; Overflow <= over_flow;
    END IF ;
  END PROCESS ;
```

Εικόνα 4.12 [Alt2] Ενδεικτικό κομμάτι κώδικα του αθροιστής / αφαιρέτης (adder / subtractor) των 8bit

Αφού αντιγράψουμε το κώδικα σώζουμε το αρχείο VHDl στο κατάλογο που έχουμε το project μας. Αν θα θέλαμε να εισάγουμε έτοιμα κομμάτια κώδικα από τη βιβλιοθήκη του Quartus θα έπρεπε να επιλέξουμε *Edit→Insert Template*. Θα μας εμφανίσει η βιβλιοθήκη από την οποία μπορούμε να επιλέξουμε έτοιμο κομμάτι κώδικα VHDL, Εικόνα 4.13 και να το εισάγουμε στη σχεδίαση μας. Υπάρχουν και άλλες σχεδιαστικές επιλογές που μπορούμε να διαλέξουμε όπως με χρήση της γλώσσας Verilog αλλά και τη σχηματική εισαγωγή.

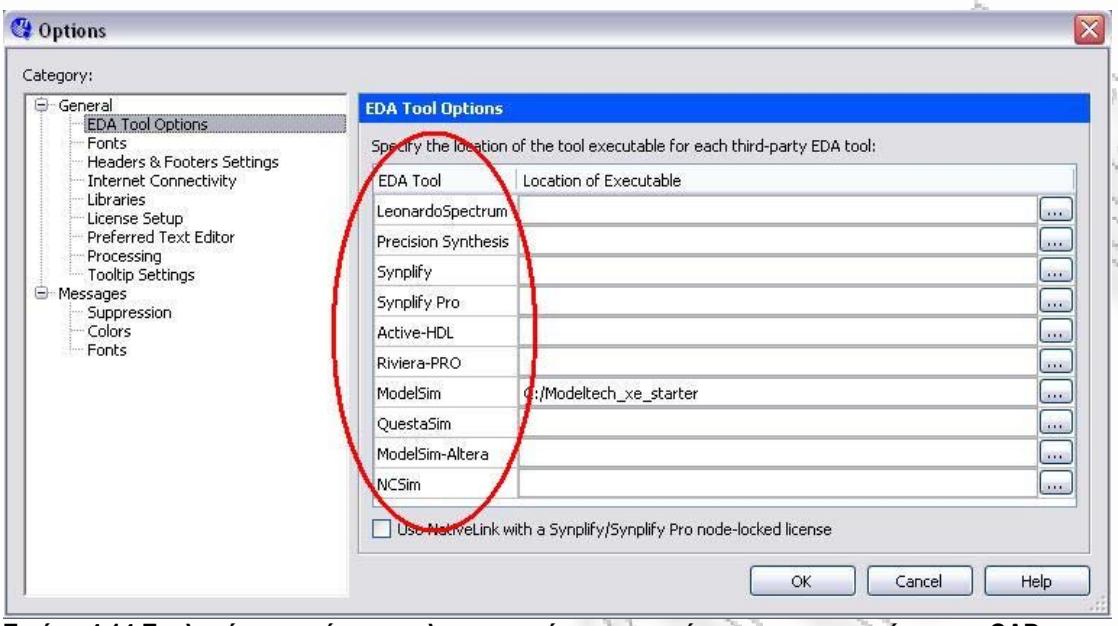


Εικόνα 4.13 Επιλογή έτοιμου κώδικα από τη βιβλιοθήκη του προγράμματος CAD Quartus II

Εμείς για της ανάγκες της παρουσίασης θα αρκεστούμε στην υλοποίηση του αθροιστή/αφαιρέτη που προαναφέραμε.

4.2.4. Λειτουργική Προσομοίωση

Για την λειτουργική προσομοίωση της σχεδίασης μας το Quartus II χρησιμοποιεί μια ποικιλία από άλλα προγράμματα όπως αυτά που φαίνονται στην Εικόνα 4.14. Στο Quartus II δεν μπορούμε να φτιάξουμε αρχείο προσομοίωσης όπως το πάγκο εργασίας (*Test Bench*) της εταιρίας Xilinx αλλά μπορούμε να το προσαρμόσουμε (αν υπάρχει από άλλη σχεδίαση) μέσα στη διαδικασία προσομοίωσης. Αυτή την παράμετρος δεν θα εξετάσουμε σε αυτή τη ενότητα διότι ξεφεύγει από τα πλαίσια της διατριβής. Θα υπάρξει αναφορά στο πάγκο εργασίας στην Ενότητα 4.3.4. Απλά να αναφέρουμε και πάλι ότι η λειτουργική προσομοίωση μας δείχνει τη συμπεριφορά της σχεδίασης μας, εμείς σε αυτό το παράδειγμα και στα άλλα που θα ακολουθήσουν παίρνουμε ως δεδομένο ότι τα προγράμματα που εξετάζουμε έχουν σωστή συμπεριφορά! Όλα αυτά γιατί αποτελούν κομμάτι εκμάθησης των σχεδιαστικών εργαλείων αλλά και το γεγονός της ενασχόλησης μας με όλα τα παραδείγματα πρωτύτερα.

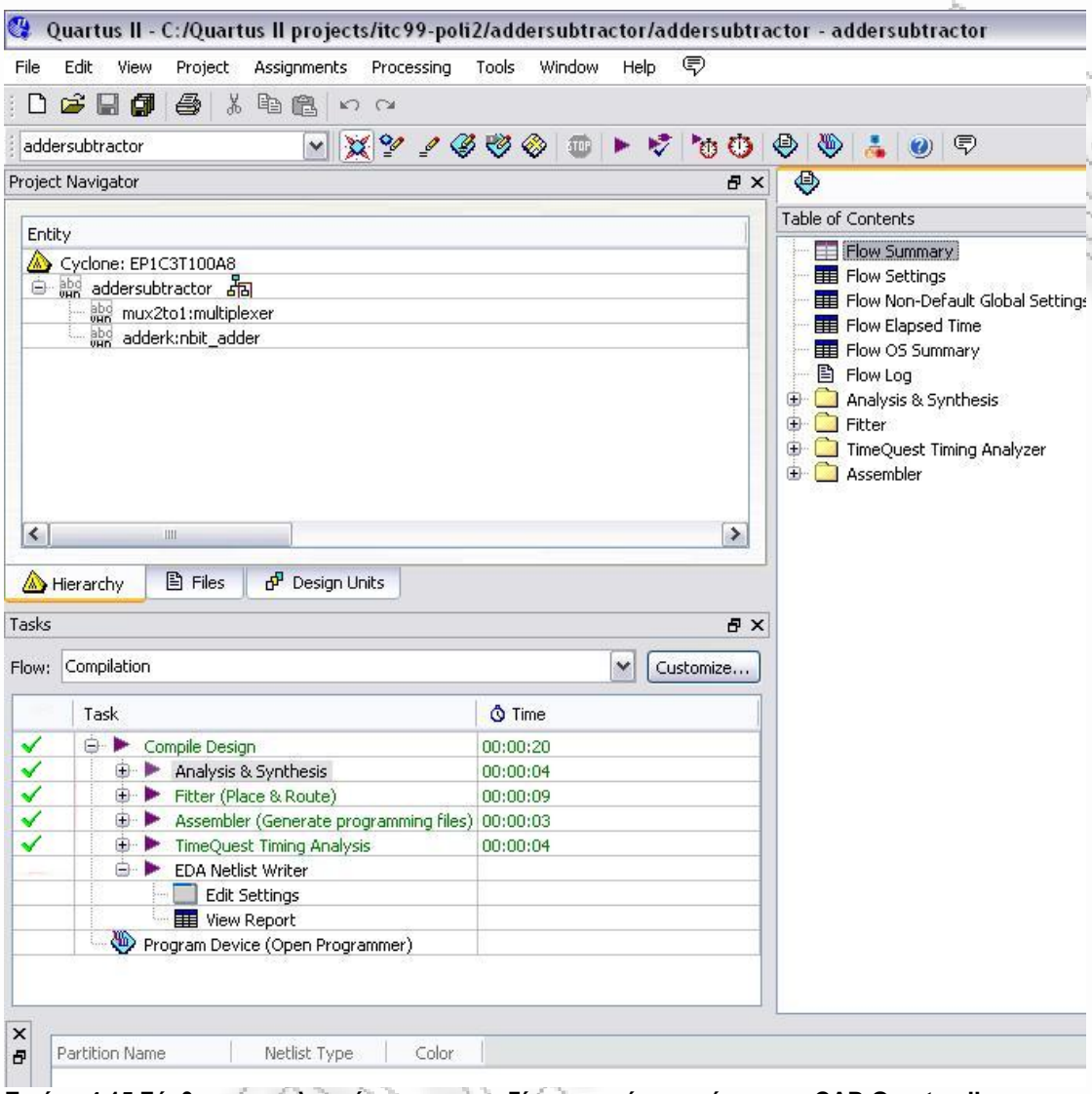


Εικόνα 4.14 Επιλογή προγράμματος λειτουργικής προσομοίωσης του προγράμματος CAD Quartus II

4.2.5. Σύνθεση και Υλοποίηση

Αφού δούμε ότι η λειτουργική προσομοίωση είναι σωστή προχωράμε στη σύνθεση της σχεδίασης μας πατώντας την επιλογή *Analysis & Synthesis* από το παράθυρο*Task*. Αν όλα πάνε καλά τότε αριστερά από την επιλογή θα εμφανιστεί μια πράσινη ένδειξη. Μετά μπορούμε να προχωρήσουμε στην υλοποίηση επιλέγοντας να εκτελέσουμε το Fitter. Όπως προαναφέραμε σε αυτή την επιλογή εκτελούνται οι λειτουργίες τις Τοποθέτησης (Place) και της Δρομολόγησης (Route). Κατόπιν αυτών των επιλογών ακολούθου δυο άλλες επιλογές, ή μια *Assempler* που έχουν να κάνουν με τη φυσική τοποθέτηση των μονάδων στη συσκευή και η επιλογή *TimeQuest Timing Analysis* η οποία έχει να κάνει με τους χρονικούς περιορισμούς που θέτουμε ή που έχουν παρουσιαστεί στη σχεδίαση μας. Λεπτομερή αναφορά για το τελευταίο θα έχουμε στη συνέχεια.

Αν θέλαμε τώρα να εκτελεστούν όλα μαζί χωρίς να το κάνουμε βήμα βήμα, επιλέγοντας *Processing→Start Compilation* θα είχαμε εκτέλεση όλων των επιλογών τη μια μετά την άλλη. Αν όλα πάνε καλά θα πρέπει να έχουμε το αποτέλεσμα της Εικόνας 4.15.



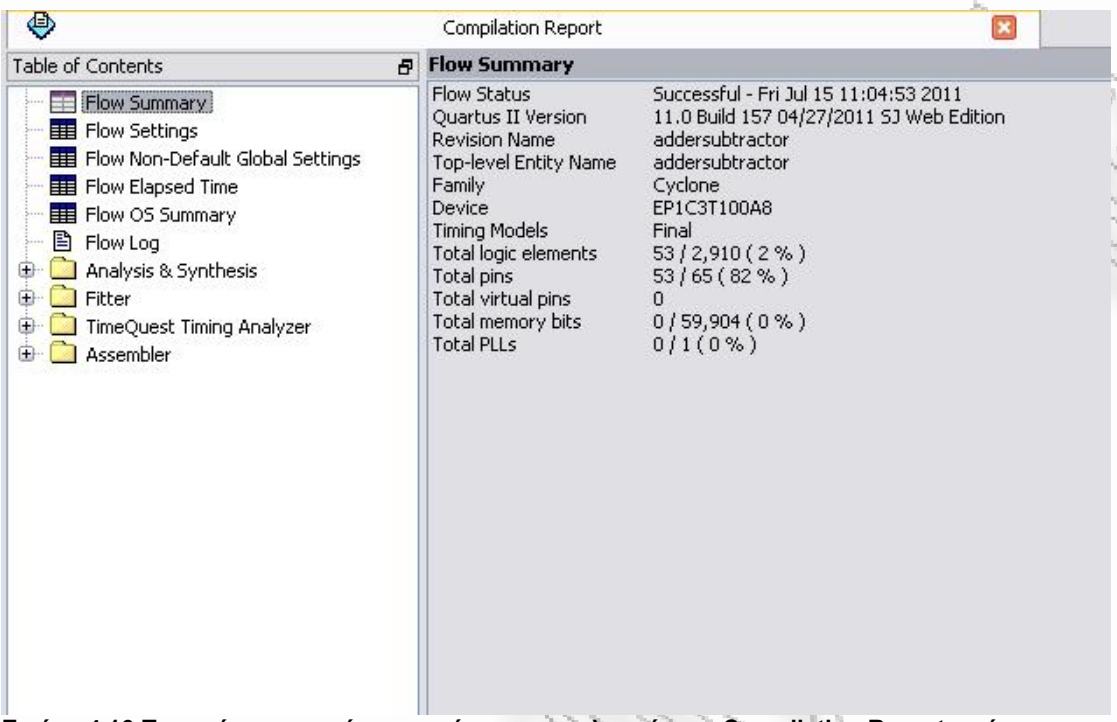
Εικόνα 4.15 Σύνθεση και υλοποίηση μιας σχεδίασης από το πρόγραμμα CAD Quartus II

4.2.6. Χρονικοί Περιορισμοί και αρχείο υλοποίησης Design Summary

Μετά την υλοποίηση το Quartus βγάζει μια αναφορά με όλα τα χαρακτηριστικά που διέπουν μια σχεδίαση. Από αυτή την αναφορά ο χρήστης (σχεδιαστής) μπορεί να καταλάβει αν η σχεδίαση του πληρεί της προδιαγραφές που έχουν ειπωθεί. Η αναφορά αυτή φαίνεται στο γενικό παράθυρο και έχει την ονομασία *Compilation Report*. Μέσα εκεί μπορεί κανείς να δει :

- Τα λογικά στοιχεία έχουν χρησιμοποιηθεί στη σχεδίαση.
- Το μέγεθος της μνήμης RAM που έχει χρησιμοποιηθεί.
- Οι ενεργοί ακροδέκτες που χρησιμοποιούνται και άλλα πολλά, Εικόνα 4.16

Για κάθε λειτουργία το Quartus βγάζει ξεχωριστή αναφορά με την οποία μπαίνει σε λεπτομέρεια ως προς τη σχεδίαση. Ένα από τα πιο σημαντικά είναι η αναφορά που βγάζει για τους χρονικούς περιορισμούς και αυτή φαίνεται από την επιλογή *TimeQuest Timing Analyzer*, Εικόνα 4.16.

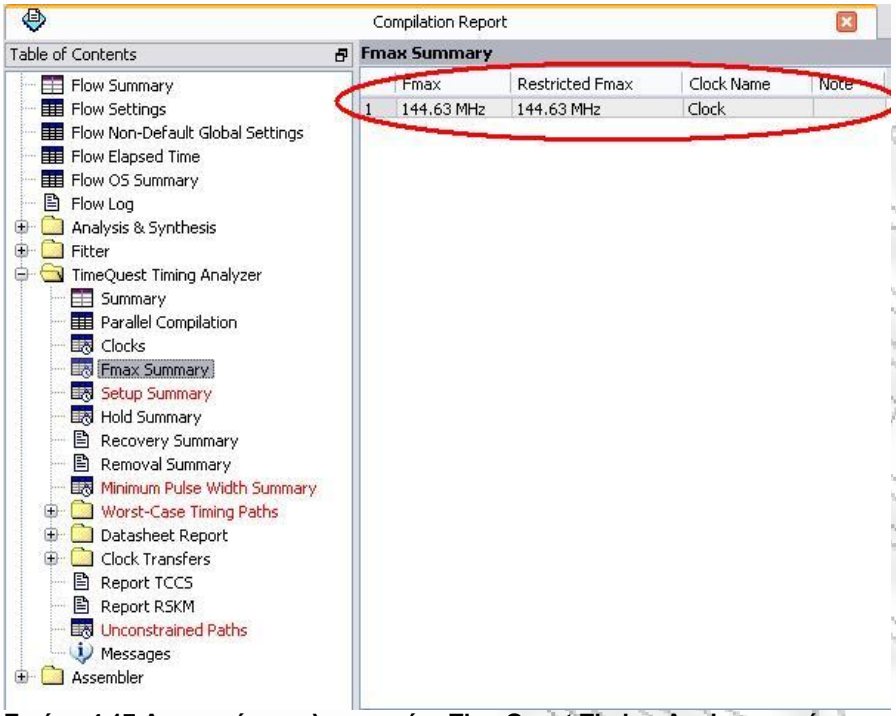


Εικόνα 4.16 Παραγόμενο αρχείο τεκμηρίωσης της υλοποίησης Compilation Report από το πρόγραμμα CAD Quartus II

Οι χρονικοί περιορισμοί είναι ένα από τα σημαντικά χαρακτηριστικά που πρέπει να ελέγχουμε σε μια σχεδίαση. Το Quartus έχει ένα πολύ αναλυτικό μηχανισμό ελέγχου και τεκμηρίωσης, ώστε να μπορεί ο χρήστης να διαπιστώνει τυχόν προβλήματα χρονισμού κατά τη σχεδίαση. Κάποια από τα πολύ σημαντικά είναι:

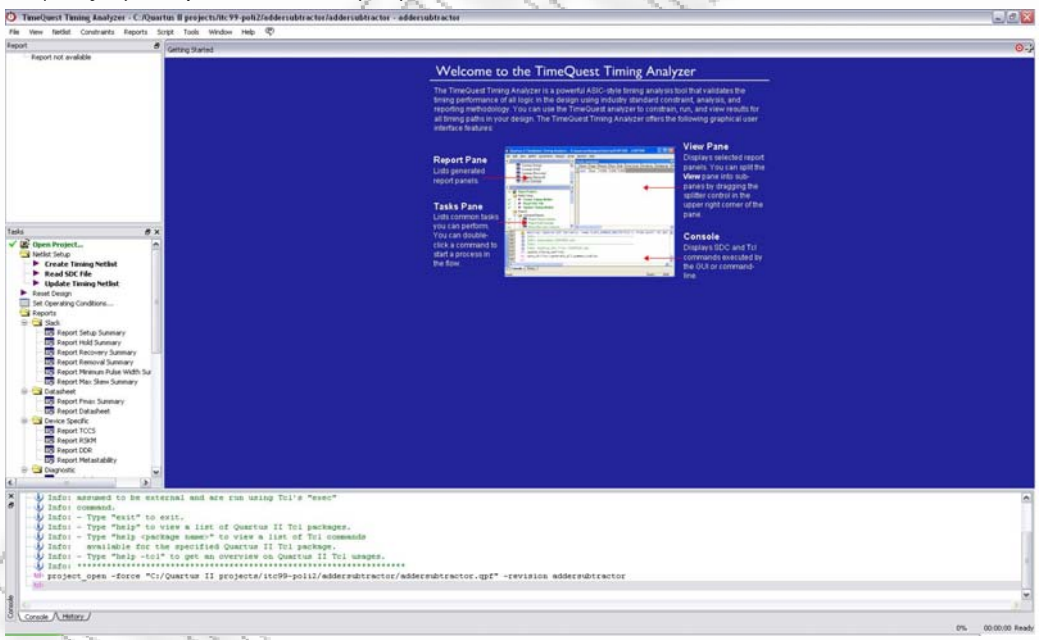
- Μέγιστη συχνότητα λειτουργίας (FMax) σχεδίασης, για να το δούμε αυτό αρκεί να επιλέξουμε από το κατάλογο *TimeQuest Timing Analyzer* την επιλογή *Fmax Summary*, Εικόνα 4.17. Αυτή η ταχύτητα λειτουργίας εξαρτάται από τη μέγιστη απόσταση (καθυστέρηση) μεταξύ καταχωρητών που χρησιμοποιούν το ίδιο ρολόι. Η λειτουργία FMax μας δείχνει ποια θα είναι η μέγιστη συχνότητας της σχεδίασης μας.
- Το Setup Summary, είναι το χρονικό διάστημα για το οποίο τα δεδομένα εισόδου σε ένα καταχωρητή θα πρέπει να είναι στους ακροδέκτες εισόδου του flip flop πριν η ενεργή ακμή του ρολογιού ενεργοποιήσει το flip flop. Αυτός ο χρόνος είναι απαραίτητος για να μπορέσει να γίνει η προπαρασκευή των δεδομένων πριν γίνει η δειγματοληψία από το καταχωρητή (flip flop).
- Το Hold Summary, είναι το ελάχιστο χρονικό διάστημα στο οποίο τα δεδομένα εισόδου πρέπει να είναι σταθερά ώστε να μπορέσει ο καταχωρητής να κάνει σωστή δειγματοληψία. Αν ο χρόνος αυτός δεν πληρεί τις προδιαγραφές του κατασκευαστή τότε υπάρχει πιθανότητα ο καταχωρητής να μη κάνει σωστή δειγματοληψία.

Επιπλέον υπάρχουν και άλλες αναφορές που παρέχουν χρήσιμες πληροφορίες όπως φαίνεται και από την Εικόνα 4.17.



Εικόνα 4.17 Αναφορές της λειτουργίας TimeQuest Timing Analyzer από το πρόγραμμα CAD Quartus II

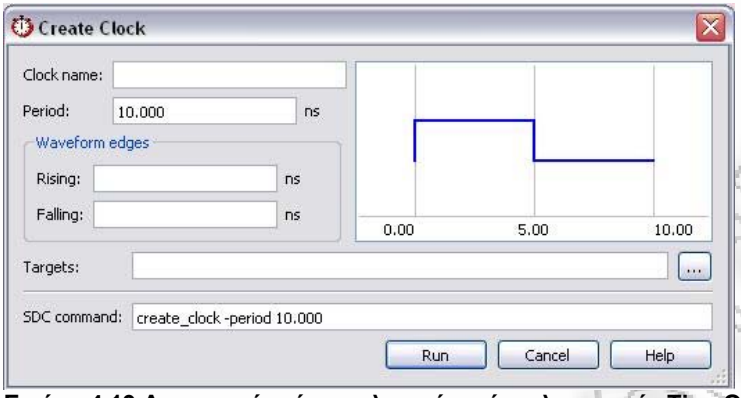
Αν κάποια αναφορά είναι κόκκινη τότε υποδηλώνει ότι τα συγκεκριμένα σημεία έχουν πρόβλημα χρονισμού ή κάποια καθυστέρηση στη λειτουργία. Αυτό σημαίνει ότι πρέπει να φτιάξουμε ένα ρολόι με συγκεκριμένα χαρακτηριστικά όπως τη συχνότητα αλλά και την περίοδο. Αυτό το πετυχαίνουμε επιλέγοντας *Tools→TimeQuest Timing Analyzer*, θα ανοίξει ένα νέο παράθυρο διαλόγου όπου μπορούμε να φτιάξουμε το ρολόι που επιθυμούμε, Εικόνα 4.18.



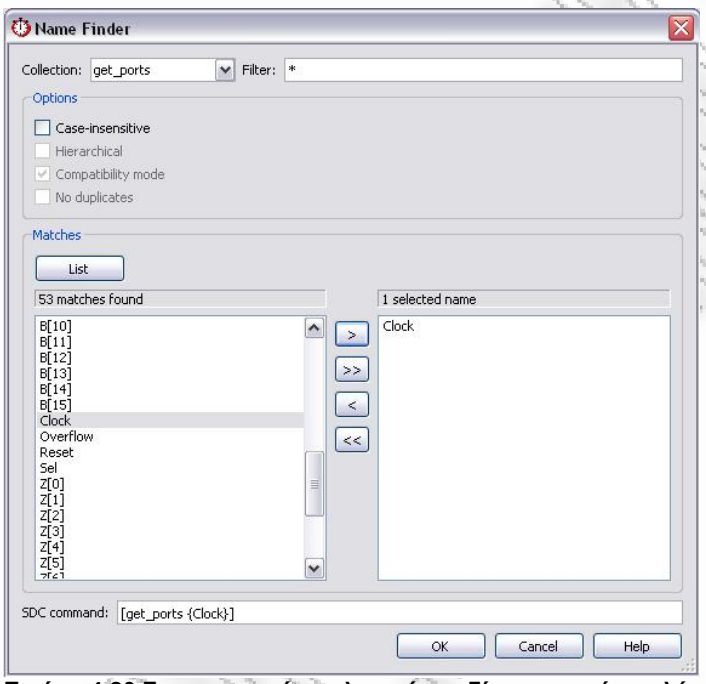
Εικόνα 4.18 Παράθυρο για τη δημιουργία ρολογιού στο TimeQuest Timing Analyzer από το πρόγραμμα CAD Quartus II

Για να δημιουργήσουμε ένα ρολόι αρκεί να κάνουμε διπλό κλικ στην επιλογή *Create Timing Netlist* και μετά Constraints→*Create Clock*, όπου το νέο ρολόι που θα δημιουργήσουμε έχει την κατάληξη *SDC* (*Synopsys Design Constraints*). Στο νέο παράθυρο που θα εμφανιστεί θα επιλέξουμε τα χαρακτηριστικά για το νέο ρολόι, Εικόνα 4.19. Αν επιλέξουμε *Targets* θα εμφανιστεί ένα νέο παράθυρο από το οποίο μπορούμε να πάρουμε το ρολόι της σχεδίασης μας και να το συγχρονίσουμε με αυτό που δημιουργούμε.

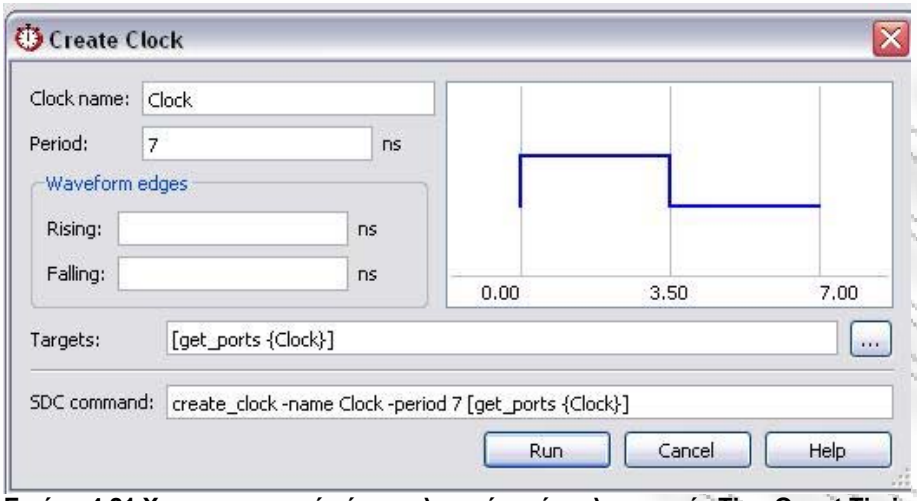
Αυτό μπορεί να γίνει επιλέγοντας από το κουμπί *List* και μετά επιλέγοντας το ρολόι μας να το μεταφέρουμε στο παράθυρο *Select Name* με το κουμπί >, Εικόνα 4.20. Πατώντας OK μεταφέρεται το ρολόι στο νέο που δημιουργούμε, Εικόνα 4.21. Τέλος πατάμε το κουμπί *Run* και αυτόματα εκτελεστικέ η καταχώρηση. Τώρα αν θέλουμε μπορούμε να αποθηκεύσουμε αυτό το ρολόι για μελλοντικές εφαρμογές πάμε *Constraints→Write SDC File* και στο νέο παράθυρο σώζουμε το αρχείο (συνήθως) στο φάκελο με το project που δημιουργήσαμε, Εικόνα 4.22.



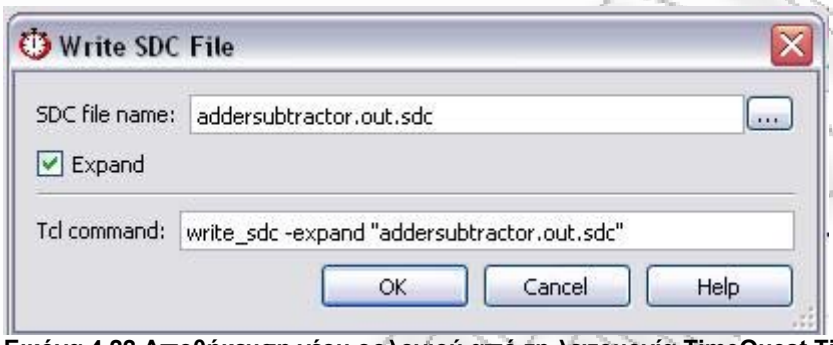
Εικόνα 4.19 Δημιουργία νέου ρολογιού από τη λειτουργία TimeQuest Timing Analyzer από το πρόγραμμα CAD Quartus II



Εικόνα 4.20 Συγχρονισμός ρολογιού σχεδίασης με νέο ρολόι από τη λειτουργία TimeQuest Timing Analyzer από το πρόγραμμα CAD Quartus II



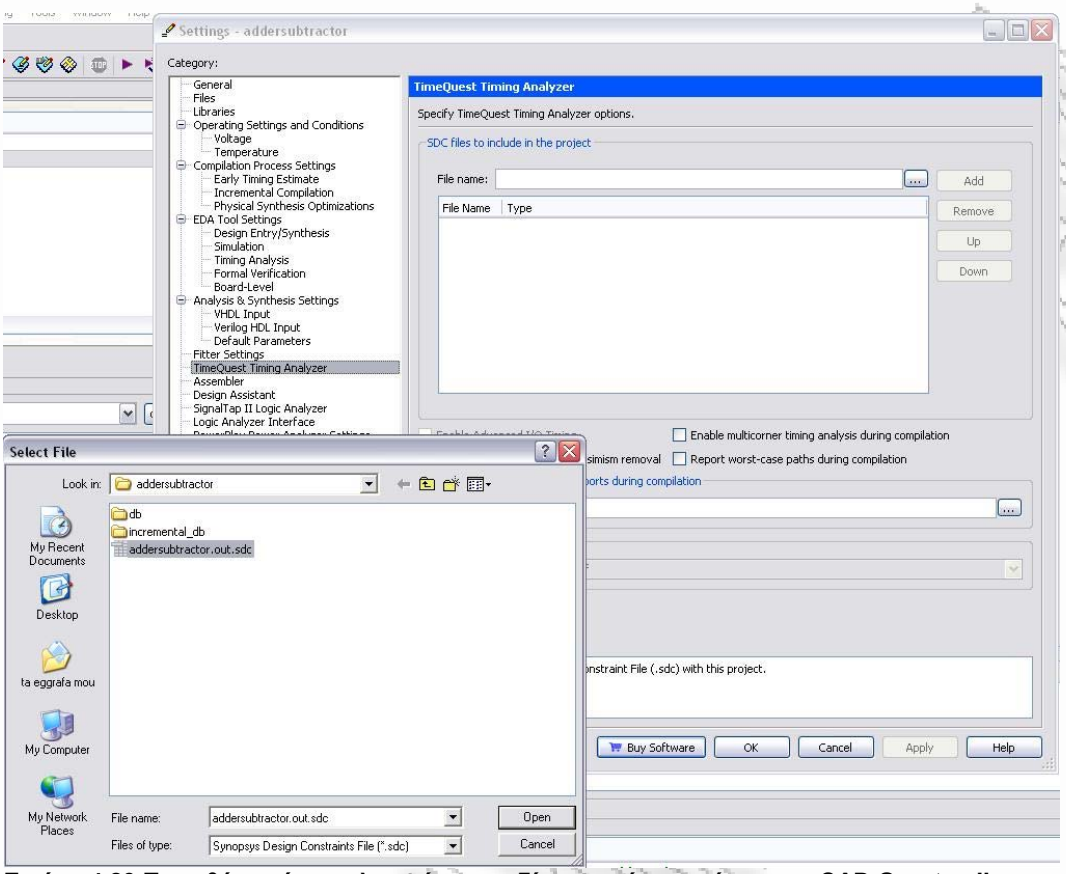
Εικόνα 4.21 Χαρακτηριστικά νέου ρολογιού από τη λειτουργία TimeQuest Timing Analyzer από το πρόγραμμα CAD Quartus II



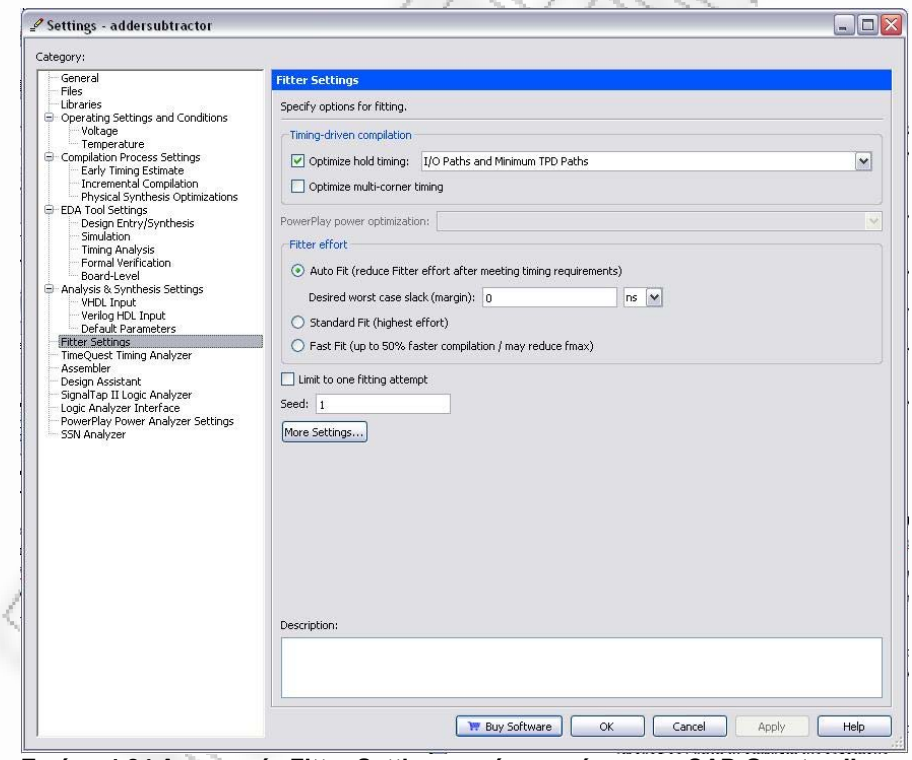
Εικόνα 4.22 Αποθήκευση νέου ρολογιού από τη λειτουργία TimeQuest Timing Analyzer από το πρόγραμμα CAD Quartus II

Για να μπορέσουμε τώρα να εισάγουμε το νέο ρολόι στη σχεδίαση μας θα πρέπει να κάνουμε πρώτα κάποιες ενέργειες. Αρχικά από την κεντρική οθόνη του Quartus επιλέγω *Assignments→Settings* και στο νέο παράθυρο επιλέγω *TimeQuest Timing Analyzer*, κατόπιν από το πεδίο *File Name* επιλέγω το ρολόι που δημιουργήσαμε, Open και OK, Εικόνα 4.23. Κατόπιν ανοίγω πάλι *Assignments→Settings* και επιλέγω τώρα *Fitter Settings*, μετά *Auto Fit* και OK, Εικόνα 4.24. Αφού τα κάνουμε όλα αυτά μπορούμε πλέον να εκτελέσουμε τη σχεδίαση μας με το νέο ρολόι.

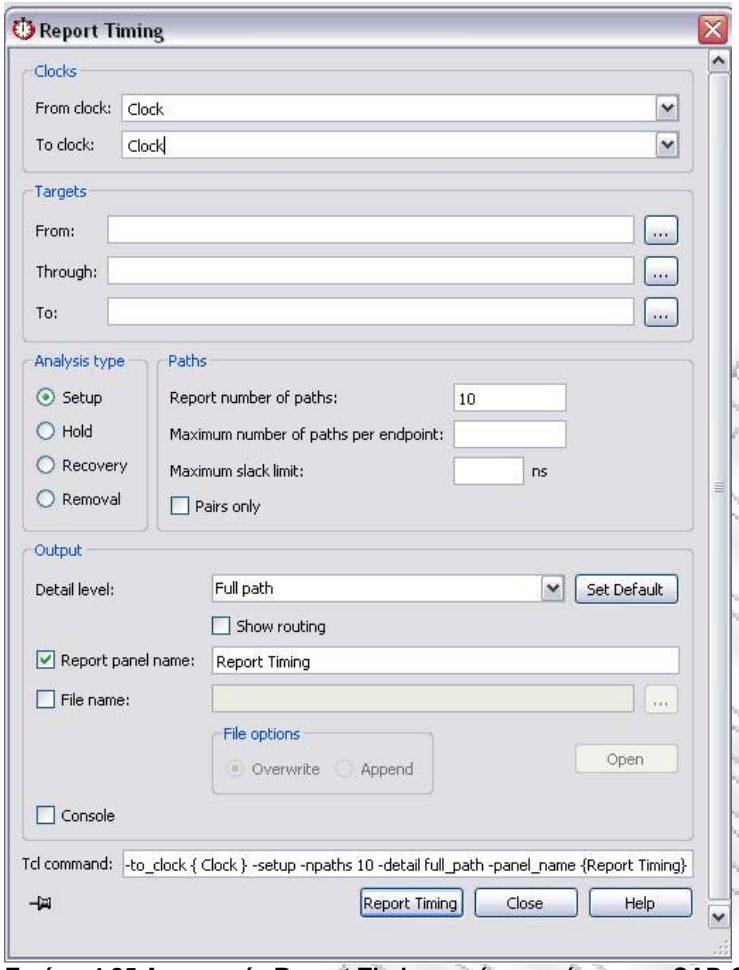
Αν τώρα θέλουμε να δούμε τα δέκα πιο κρίσιμα μονοπάτια της σχεδίασης μας, επιλέγουμε να ανοίξουμε τη λειτουργία *TimeQuest Timing Analyzer,* Εικόνα 4.17 και επιλέγοντας από το παράθυρο *Tasks* το κατάλογο *Custom Reports,* ανοίγουμε με διπλό κλικ το *Report Timing*. Στο νέο παράθυρο διαλόγου επιλέγω από τα πεδία *From Clock* και *To Clock* το *Clock* που έχουμε δημιουργήσει και στο πεδίο *Report Number of paths* βάζω τον αριθμό 10, Εικόνα 4.25. Πατώντας *Report Timing* θα εμφανιστούν στο TimeQuest Timing Analyzer τα δέκα πιο κρίσιμα μονοπάτια, Εικόνα 4.26.



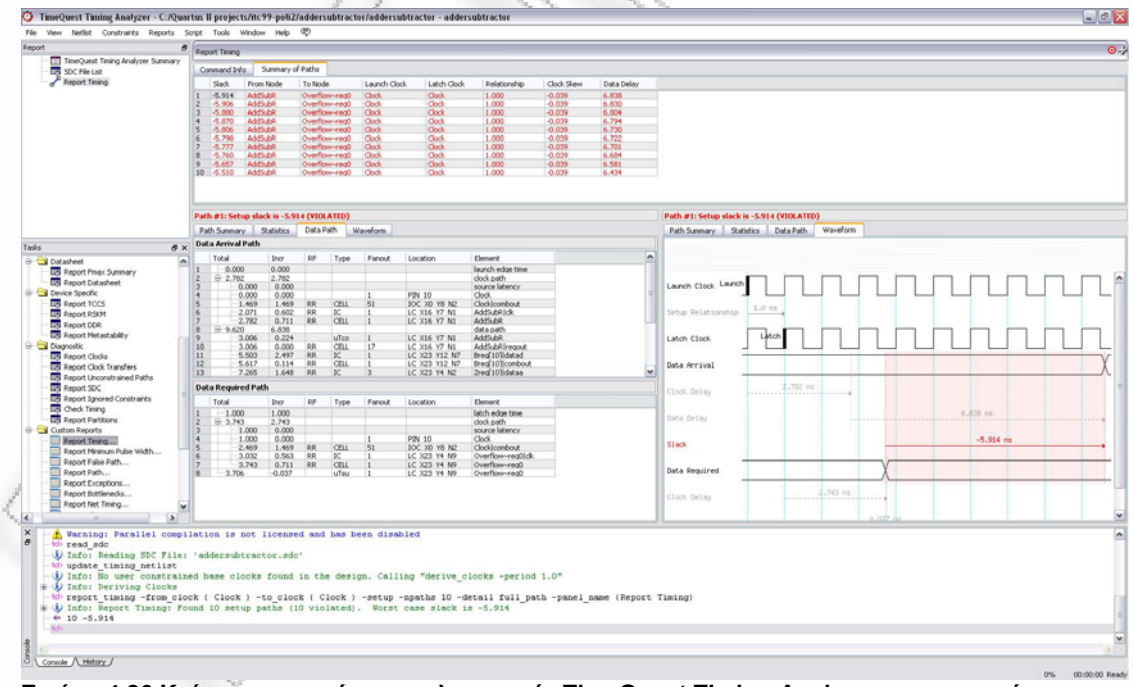
Εικόνα 4.23 Προσθήκη νέου ρολογιού σε σχεδίαση από το πρόγραμμα CAD Quartus II



Εικόνα 4.24 Λειτουργία Fitter Settings από το πρόγραμμα CAD Quartus II



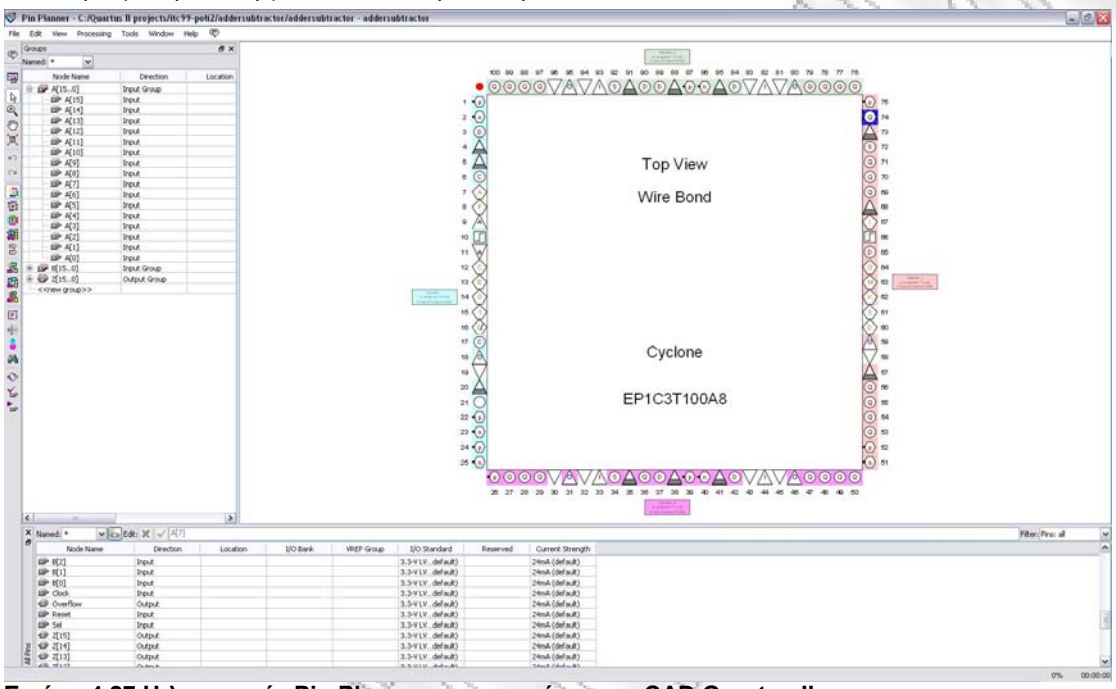
Εικόνα 4.25 Λειτουργία Report Timing από το πρόγραμμα CAD Quartus II



Εικόνα 4.26 Κρίσιμα μονοπάτια στη λειτουργία TimeQuest Timing Analyzer του προγράμματος CAD Quartus II

4.2.7. Ακροδέκτες εισόδου εξόδου και Pinout Report

Ένα τελευταίο χαρακτηριστικό που πρέπει να δούμε είναι αυτό της διαμόρφωσης των ακροδεκτών. Έναν πρώτο τρόπο είδαμε από την ενότητα με τις αναφορές (reports) που βγάζει το Quartus. Αυτές οι αναφορές είναι γενικές, εμείς αν θέλουμε να δούμε πόσοι ακροδέκτες έχουν χρησιμοποιηθεί για την εισαγωγή και πόσοι για την έξοδο αρκεί να επιλέξουμε *Assignments→Pin Planer*. Στο νέο παράθυρο μπορούμε να δούμε σε γραφική αναπαράσταση ποιοι ακροδέκτες χρησιμοποιούνται αλλά και ονομαστικά μέσω των δυο παράθυρων αριστερά και κάτω, Εικόνα 4.27. Από την περιγραφή *Direction* μπορούμε να καταλάβουμε τη λειτουργία του κάθε ακροδέκτη.



Εικόνα 4.27 Η λειτουργία Pin Planer του προγράμματος CAD Quartus II

Ο προγραμματισμός της συσκευής γίνεται εκτελώντας την επιλογή *Program Devise* από το παράθυρο Tasks, Εικόνα 4.15. Επειδή όμως αυτό δεν αποτελεί μέρος της διατριβής δεν θα το παρουσιάσουμε σε βάθος. Η παραγωγή του αρχείου Bitstream γίνεται αυτόματα κατά τη διαδικασία της υλοποίησης.

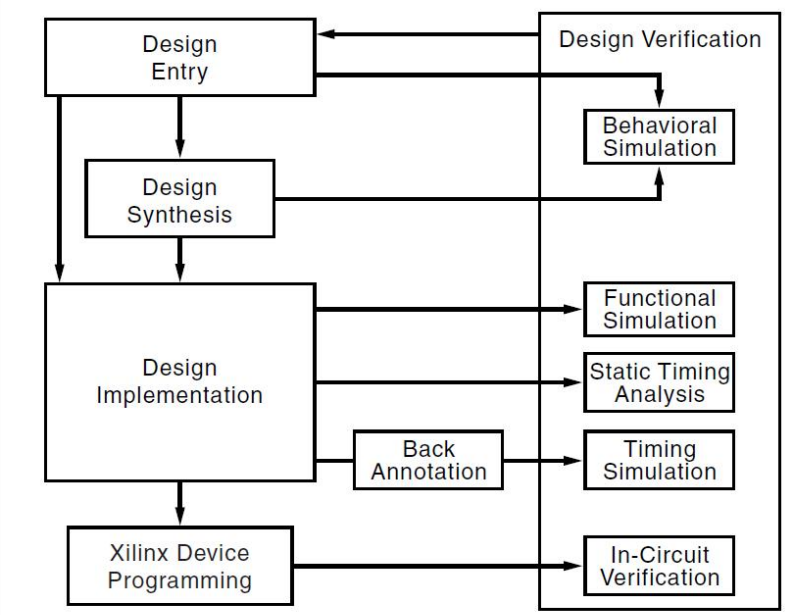
Σε αυτή την ενότητα είδαμε πως μπορούμε να σχεδιάσουμε και να δημιουργήσουμε αρχεία VHDL με το εργαλείο σχεδίασης Quartus II. Συγκεκριμένα είδαμε το τρόπο σχεδίασης με κώδικα VHDL, πως ελέγχουμε τη σχεδίαση μας αλλά και τη διαδικασία. Είδαμε το τρόπο που θέτουμε χρονικούς περιορισμούς σε μία σχεδίαση αλλά και την τελική σύνοψη της σχεδίασης μας σε μια ποικιλία από αναφορές σε παράθυρα.

Για περεταίρω μελέτη του εργαλείου σχεδίασης Quartus II της εταιρίας Altera δείτε τα αντίστοιχα έγγραφα *Quartus® II Introduction for VHDL Users* [Alt2] και το *Quartus II Handbook v11.0.0* [Alt3] τα οποία αναφέρονται στη βιβλιογραφία.

4.3. Σχεδίαση με χρήση εργαλείου Xilinx ISE

4.3.1. Εισαγωγή στο ISE

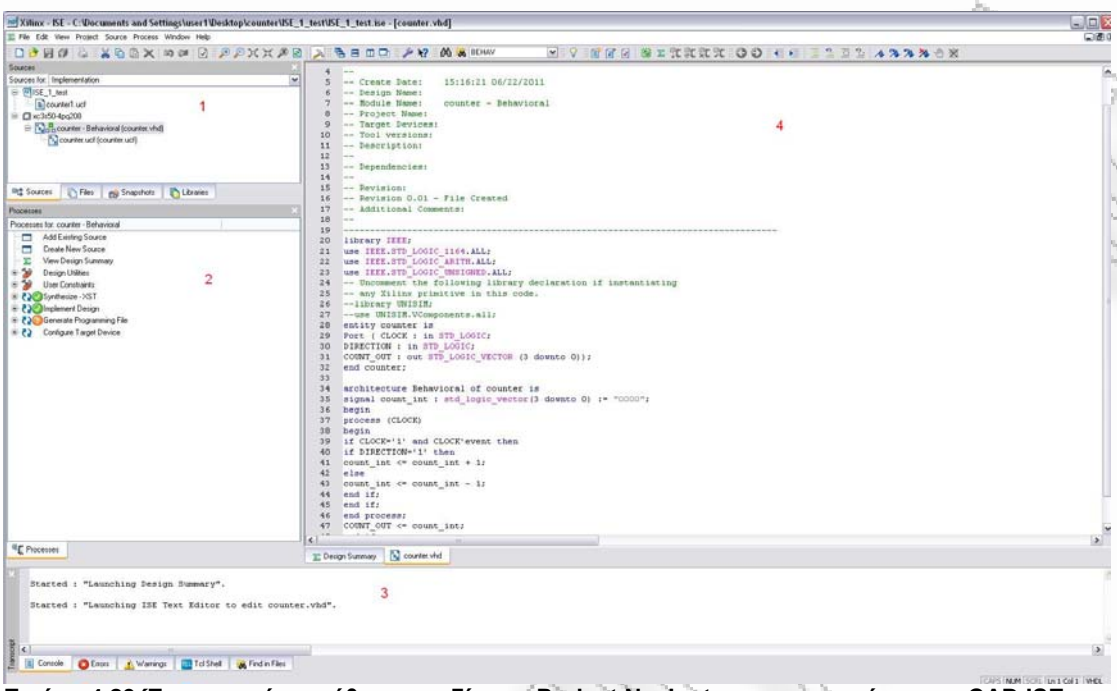
Το επόμενο εργαλείο σχεδίασης που θα δούμε είναι το ISE και συγκεκριμένα η έκδοση 10.1. Η γενική μεθοδολογία σχεδίασης που προτείνεται στο ISE είναι αυτό της Εικόνας 4.28. Όπως μπορείτε να δείτε έχει ομαδοποιηθεί η υλοποίηση (*Design Implementation*) η οποία περιλαμβάνει τα στάδια της λειτουργικής προσομοίωσης (*Functional Simulation*), της στατικής και χρονικής ανάλυσης (*Static Timing Analysis*) αλλά και αυτή της χρονικής προσομοίωσης (*Timing Simulation*). Να σημειωθεί εδώ ότι η Εικόνα που βλέπετε μπορεί να διαφέρει αρκετά με αυτό της Εικόνας 4.1, αλλά η ακολουθία των βημάτων είναι η ίδια με αυτή που περιγράψαμε στην Ενότητα 4.1. Άλλωστε θα το διαπιστώσετε και μόνοι σας από αυτά που θα ακολουθήσουν.



Εικόνα 4.28 [XIn4] Γενική μεθοδολογία σχεδίασης στο πρόγραμμα CAD ISE

Το περιβάλλον εργασίας του ISE είναι το Project Navigator το οποίο φαίνεται στην Εικόνα 4.29 το οποίο αποτελείται από τέσσερα υποπαράθυρα τα οποία είναι τα εξής :

1. Το παράθυρο *Source in Project*, το οποίο αποτελείται από τρεις ετικέτες που παρέχουν πληροφορίες για το χρηστή σε σχέση με τα αρχεία που χρησιμοποιούνται στο κάθε project.
2. Το παράθυρο *Processes for Current Source*, το οποίο έχει την ετικέτα *Process View* η οποίο περιέχει τις διαδικασίες στις οποίες μπορεί να προβεί για εκτέλεση ο χρήστης και αλλάζει το περιεχόμενο της ανάλογα το αρχείο που εκτελείται εκείνη τη στιγμή.
3. Το παράθυρο *Console*, το οποίο μας δείχνει τα τυχόν λάθη ή μηνύματα που θα προκύψουν κατά τη διάρκεια της σχεδίασης
4. Και τέλος το τέταρτο παράθυρο είναι αυτό στη μέση το οποίο χρησιμοποιείται για ανάγνωση και εγγραφή σε ποικιλία αρχείων που βρίσκονται μέσα στο project όπως αναφορές, αρχεία VHDL, αρχεία προσομοίωσης συμπεριφοράς κτλ.

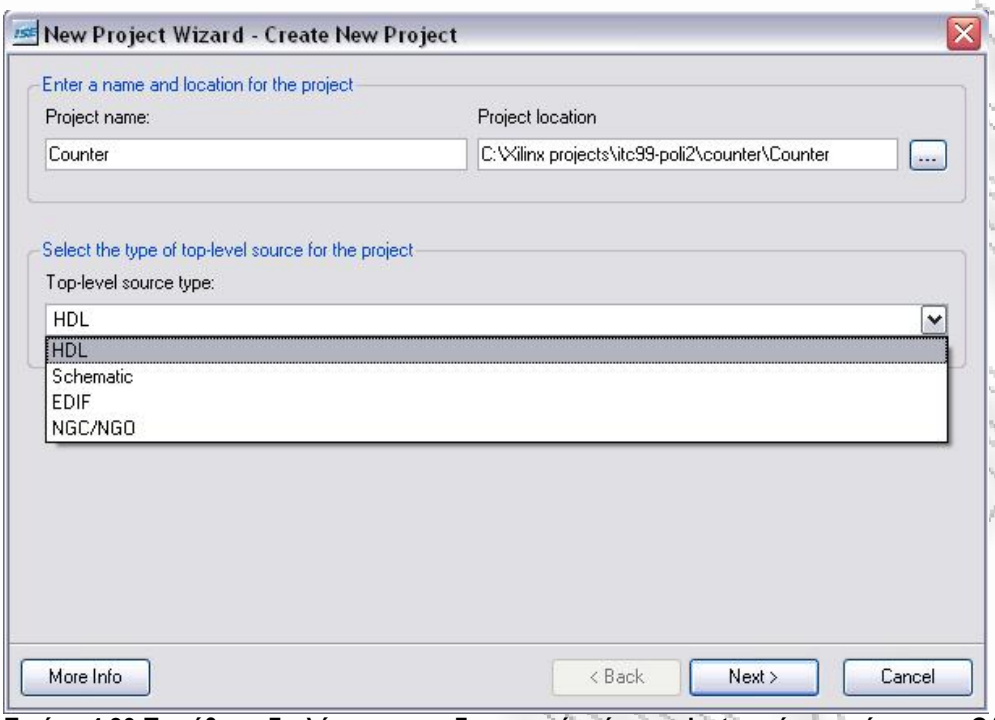


Εικόνα 4.29 Ένα τυπικό παράθυρο σχεδίασης Project Navigator του προγράμματος CAD ISE

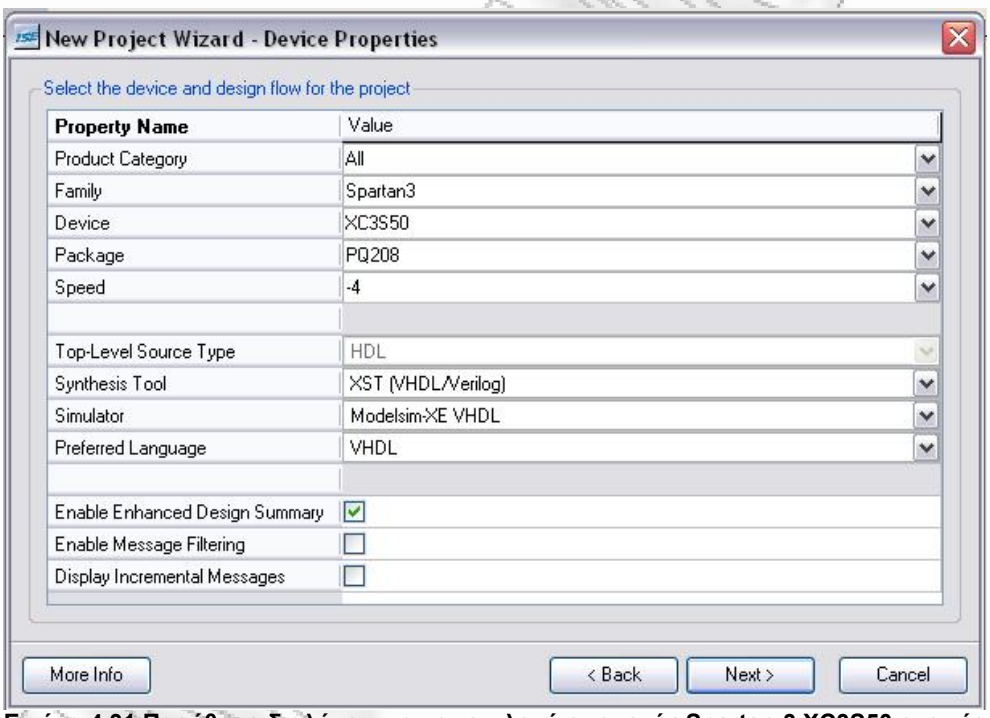
4.3.2. Δημιουργία νέου Project

Όπως και στην προηγούμενη ενότητα έτσι και τώρα θα δημιουργήσουμε ένα αρχείο VHDL το οποίο και θα εκτελέσουμε. Σε αυτή την ενότητα θα ακολουθήσουμε το προτεινόμενο πρόγραμμα που βρίσκεται στον οδηγό χρήσης του εργαλείου ISE, το οποίο είναι ένας μετρητής *Counter*.

Το πρώτο που θα κάνουμε είναι να δημιουργήσουμε ένα νέο project, πάμε *File→New Project*. Στο παράθυρο διαλόγου που εμφανίζεται επιλέγουμε το όνομα του νέου project έστω *Counter*, το κατάλογο προορισμού αλλά και τη σχεδίαση θέλουμε να κάνουμε, Εικόνα 4.30. Κατόπιν πατάμε next, στο νέο παράθυρο διαλόγου επιλέγουμε τη συσκευή της επιθυμίας μας, έστω *Spartan–3 XC3S50* αλλά και τα υπόλοιπα χαρακτηριστικά αν θέλουμε, Εικόνα 4.31 Αν σε περίπτωση η συσκευή που έχουμε επιλέξει αποδειχθεί λιγότερο ικανή (λόγω χαρακτηριστικών) για το project που θα έχουμε δημιουργήσει, τότε μπορούμε να την αλλάξουμε από τις ιδιότητες του project.



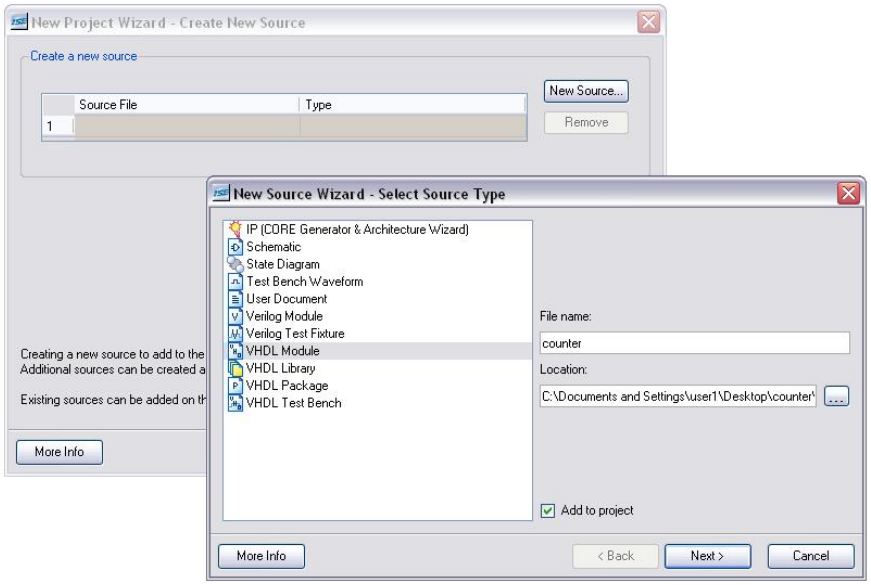
Εικόνα 4.30 Παράθυρο διαλόγου για τη δημιουργία νέου project, από το πρόγραμμα CAD ISE



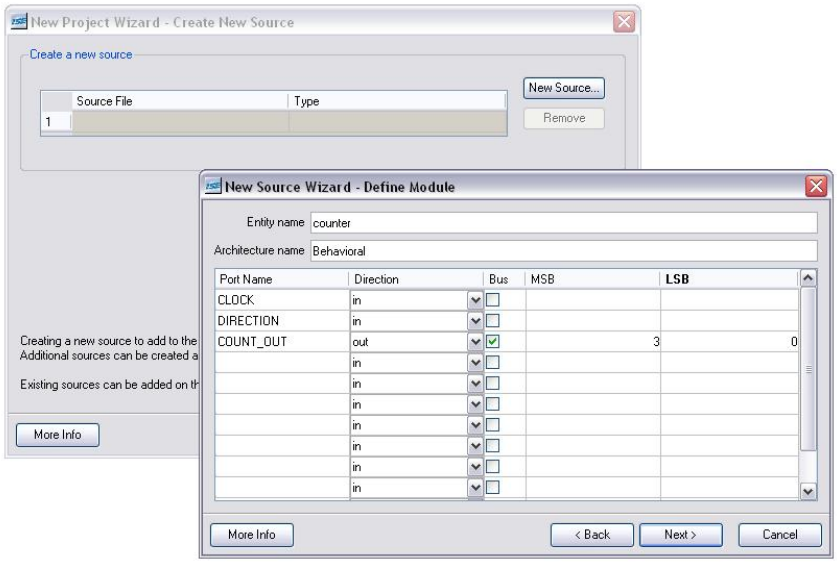
Εικόνα 4.31 Παράθυρο διαλόγου για την επιλογή συσκευής Spartan-3 XC3S50 στο νέο project, από το πρόγραμμα CAD ISE

Στη συνέχεια αφού επιλέξουμε τη συσκευή πατάμε Next. Στο νέο παράθυρο διαλόγου θα πρέπει να επιλέξου ένα αρχείο στο οποίο θα είναι η πηγή της εργασίας που θα κάνουμε. Επιλέγουμε *New Source*, από το νέο παράθυρο που θα εμφανιστεί επιλέγουμε το αρχείο στο οποίο θα γράψουμε το κώδικα μας. Επιλέγουμε *VHDL Module* και βάζουμε όνομα αρχείου Counter και μετά Next, Εικόνα 4.32. Έχουμε ένα νέο παράθυρο στο οποίο θα πρέπει να βάλουμε τις μεταβλητές εισόδου αλλά και εξόδου για τη νέα οντότητα (*Entity*) που δημιουργούμε, γράφουμε και επιλέγουμε ότι φαίνεται στην Εικόνα 4.33. Αφού γίνει και αυτό πατάμε Next και Finish, αφού επανέλθουμε στο αρχικό παράθυρο θα πρέπει να έχει

περιλάβει το νέο αρχείο που δημιουργήσαμε, Εικόνα 4.34. Μετά από όλα αυτά με δυο Next και Finish έχουμε δημιουργήσει το νέο project.



Εικόνα 4.32 Παράθυρο διαλόγου για την επιλογή VHDL Module στο νέο project, από το πρόγραμμα CAD ISE

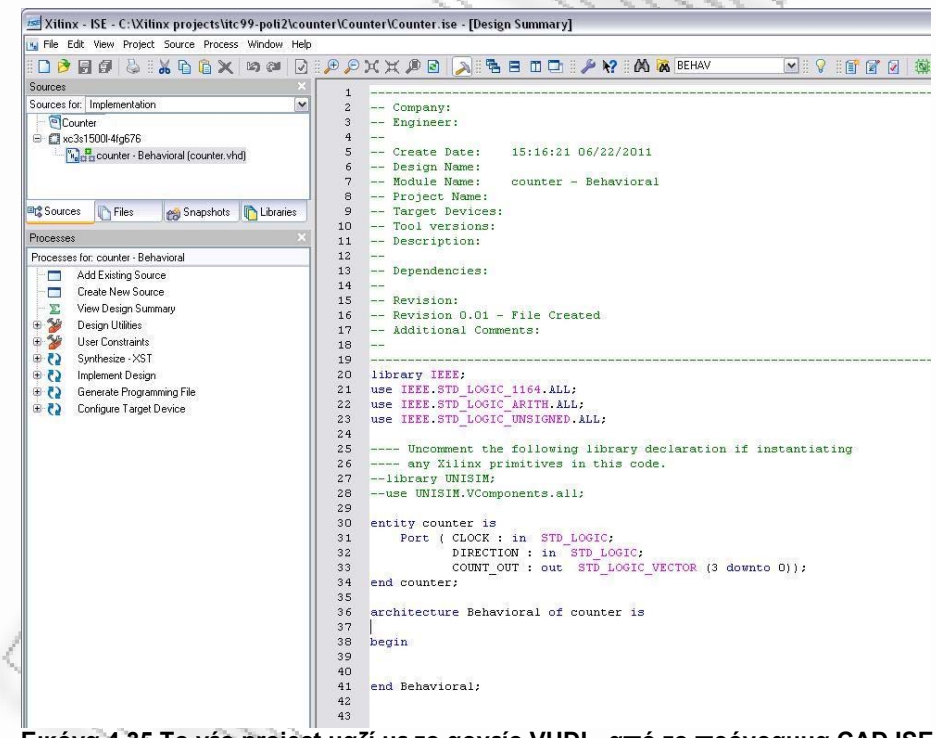


Εικόνα 4.33 Παράθυρο διαλόγου για τον ορισμό της νέας οντότητας (Entity) που δημιουργούμε στο project, από το πρόγραμμα CAD ISE

Εικόνα 4.34 Παράθυρο διαλόγου που περιλαμβάνει τη νέα οντότητα (Entity) στο νέο project, από το πρόγραμμα CAD ISE

4.3.3. Σχεδίαση με γλώσσα VHDL

Αν όλα πάνε καλά θα πρέπει στο εργαλείο ISE να υπάρχει το νέο project μαζί με το αρχείο VHDL όπως φαίνεται στην Εικόνα 4.35. Αν τώρα θα θέλαμε να εμπλουτίσουμε το αρχείο μας με νέες και έτοιμες μονάδες λογικής όπως είναι ένας απλός μετρητής, αρκεί να πάμε *Edit→Language Templates→VHDL→ Synthesis Constructs→Coding Examples→Counters→Binary→Up / Down Counters→Simple Counter*. Με αυτό το τρόπο μπορούμε να εισάγουμε έτοιμες μονάδες λογικής στη σχεδίαση μας. Στις βιβλιοθήκες αυτές υπάρχουν διαφόρων ειδών λογικές μονάδες χωρίς εμείς να χρειαστεί να δημιουργήσουμε τα περισσότερα από αυτά.



Εικόνα 4.35 Το νέο project μαζί με το αρχείο VHDL, από το πρόγραμμα CAD ISE

Στη συνέχεια, θα πρέπει να διαμορφώσουμε τη σχεδίαση μας σύμφωνα με το κώδικα που φαίνεται στην Εικόνα 4.36, αν το κάνουμε αυτό αποθηκεύουμε. Το κομμάτι αυτού του κώδικα βρίσκεται στο παράρτημα Β με το κωδικό όνομα *counter.vhd*.

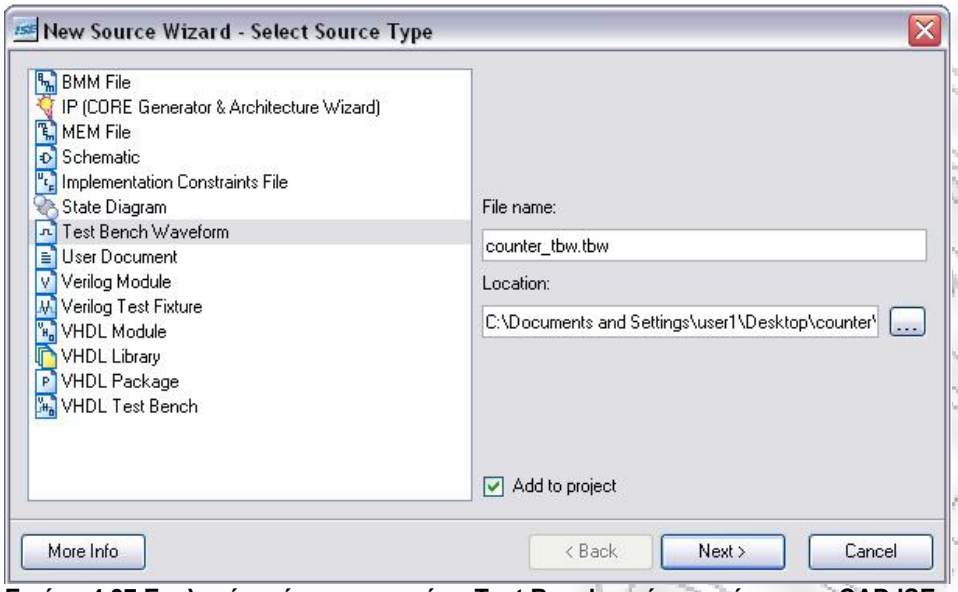
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitive in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity counter is
  Port ( CLOCK : in  STD_LOGIC;
    DIRECTION : in  STD_LOGIC;
    COUNT_OUT : out  STD_LOGIC_VECTOR (3 downto 0));
end counter;

architecture Behavioral of counter is
signal count_int : std_logic_vector(3 downto 0) := "0000";
begin
process (CLOCK)
begin
  if CLOCK='1' and CLOCK'event then
    if DIRECTION='1' then
      count_int <= count_int + 1;
    else
      count_int <= count_int - 1;
    end if;
  end if;
end process;
COUNT_OUT <= count_int;
end Behavioral;
```

Εικόνα 4.36 [Xln3] Κώδικας VHDL για την υλοποίηση απλού μετρητή Counter

4.3.4. Λειτουργική Προσομοίωση

Για να μπορέσουμε να δούμε αν η σχεδίαση μας είναι σωστή πρέπει να κάνουμε σύνθεση και λειτουργική προσημείωση, όπως είδαμε και στην Ενότητα 4.1. Πριν κάνουμε σύνθεση καλό είναι να δούμε τη λειτουργική προσομοίωση. Για να γίνει αυτό στο ISE πρέπει να δημιουργήσουμε έναν «*πάγκο εργασίας*» για την ακρίβεια να δημιουργήσουμε ένα αρχείο *Test Bench*. Υπάρχουν δύο ειδών αρχεία Test Bench που έχει το ISE, το πρώτο *Test Bench Waveform* με το οποίο παράγουμε κυματομορφές με χειροκίνητη εισαγωγή των τιμών εισόδου και το δεύτερο *VHDL Test Be*nch γράφοντας ένα αρχείο με τη γλώσσα VHDL. Στη δεύτερη περίπτωση ο χρήστης μπορεί να γράψει την κυματομορφή εισαγωγής που επιθυμεί με την τοποθέτηση μηδέν και ένα. Μέσω των παραγόμενων κυματομορφών μπορούμε να διαπιστώσουμε αν η σχεδίαση μας έχει τη σωστή συμπεριφορά. Εμείς θα δούμε την πρώτη εφαρμογή και μετά τη δεύτερη.

Για την πρώτη περίπτωση, το πετυχαίνουμε πηγαίνοντας *Project→New Source* και επιλέγοντας τη δημιουργία αρχείου *Test Bench WaveForm*, δηλαδή το αρχείο αυτό θα μας δείξει τις κυματομορφές που θα παράγει η σχεδίαση μας. Σώζουμε με το όνομα *counter_tbw.tbw* και πατάμε Next, Next και Finsh, Εικόνα 4.37.

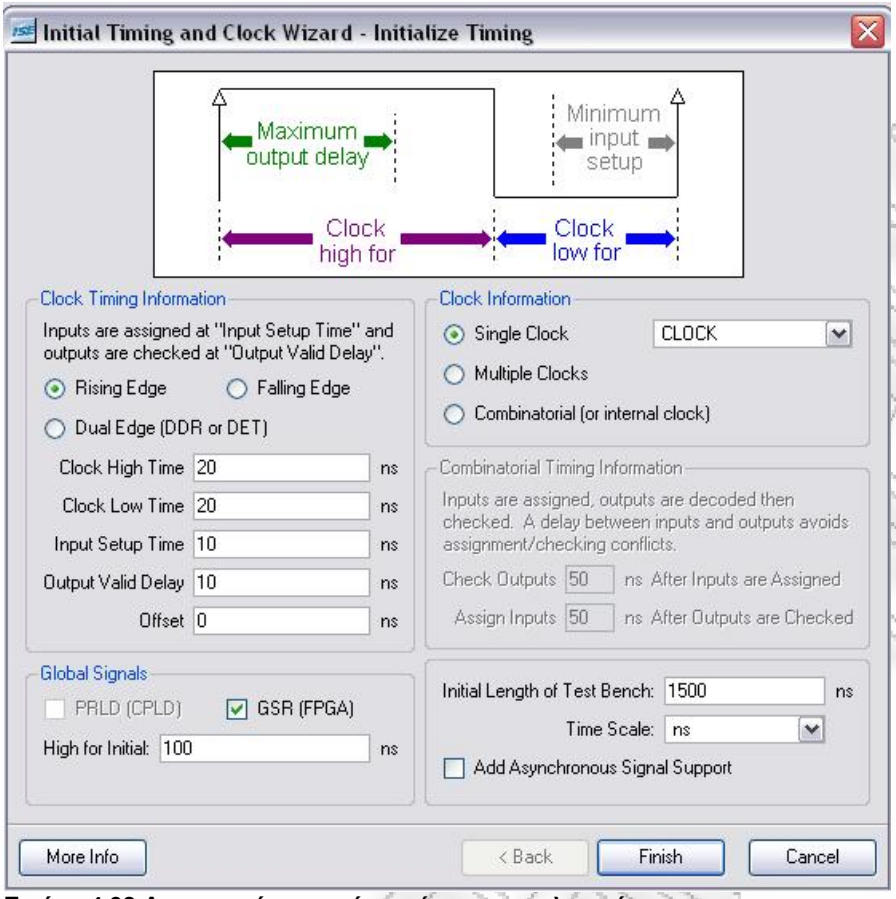


Εικόνα 4.37 Επιλογή «πάγκου εργασίας»Test Bench από το πρόγραμμα CAD ISE

Κατόπιν θα εμφανίσει ένα παράθυρο το *Initial Timing and Clock Wizard* στο οποίο θα πρέπει να τροποποιήσουμε κάποιες παραμέτρους όπως χρόνος διάρκειας προσομοίωσης, χρόνους ρολογιού (High and Low) κτλ. Αυτές οι παράμετροι χρησιμεύουν για να θέσουμε μερικές προδιαγραφές στο σύστημα μας και μέσω αυτών θα δούμε αν η σχεδίαση μας λειτουργεί σωστά. Έτσι επιλέγουμε να αλλάξουμε:

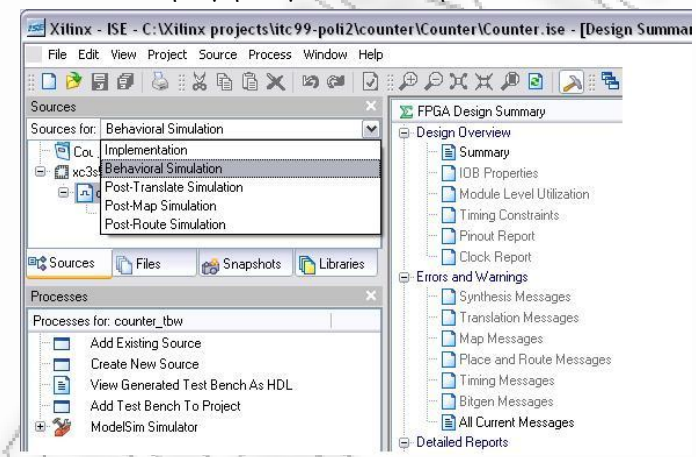
- Clock High Time: **20** ns
- Clock Low Time: **20** ns
- Input Setup Time: **10** ns
- Output Valid Delay: **10** ns
- Offset: **0** ns
- Global Signals: **GSR (FPGA)**
- High for initial: **100** ns
- Initial Length of Test Bench: **1500** ns

Οι νέες ρυθμίσεις που κάναμε θα πρέπει να φαίνονται όπως στην Εικόνα 4.38.

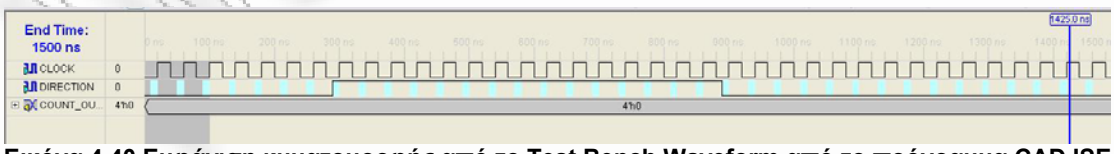


Εικόνα 4.38 Αρχικοποίηση τιμών χρόνου και ρολογιού

Από το παράθυρο Sources του εργαλείου ISE επιλέγουμε την προσομοίωση συμπεριφοράς *Behaviral Simulation* Εικόνα 4.39 και με διπλό click στο αρχείο *counter_tbw.tbw* θα εμφανιστεί η κυματομορφή Εικόνα 4.40, στην οποία μπορούμε να θέσουμε και τιμές εισαγωγής που επιθυμούμε. Εμείς θέσαμε στα 300ns ανοδική ακμή και μετά καθοδική στα 900ns.



Εικόνα 4.39 Επιλογή Behavioral Simulation από το πρόγραμμα CAD ISE

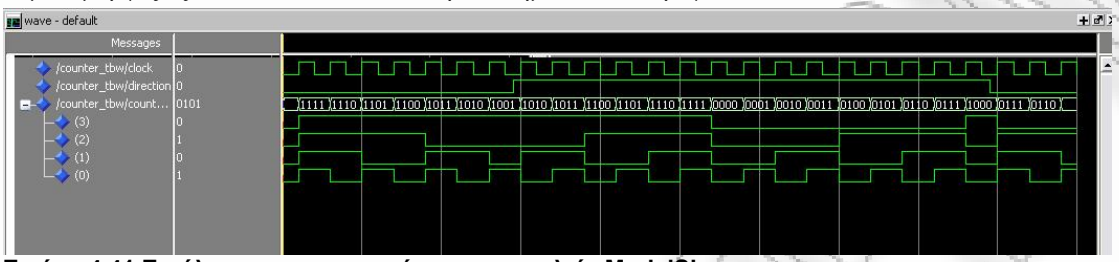


Εικόνα 4.40 Εμφάνιση κυματομορφής από το Test Bench Waveform από το πρόγραμμα CAD ISE

Όμως υπάρχει και μία επιπλέον βοήθεια και αυτή έρχεται από ένα άλλο πρόγραμμα προσομοίωσης το *ModelSim* [Mdls] το οποίο είναι συμβατό με τα αρχεία του ISE. Γενικά κάποιες εταιρίες κατασκευής

εργαλείων σχεδίασης CAD το περιλαμβάνουν μέσα στο πακέτο λογισμικού που προσφέρουν και άλλες το προτείνουν ως ένα επιπλέον λογισμικό για ελέγχους και προσομοιώσεις.

Επιλέγοντας από την καρτέλα *Processes* το "+" βρίσκεται το *Simulate Behavioral Model*, με διπλό click πάνω, ενεργοποιείται το ModelSim και εμφανίζεται η κυματομορφή εκτελούμενη με την είσοδο που της θέσαμε, Εικόνα 4.41. Με αυτόν το τρόπο μπορούμε να διαπιστώσουμε τις παραγόμενες κυματομορφές εξόδου αλλά και τα εσωτερικά σήματα που παράγονται.



Εικόνα 4.41 Εκτέλεση κυματομορφής με το εργαλείο ModelSim

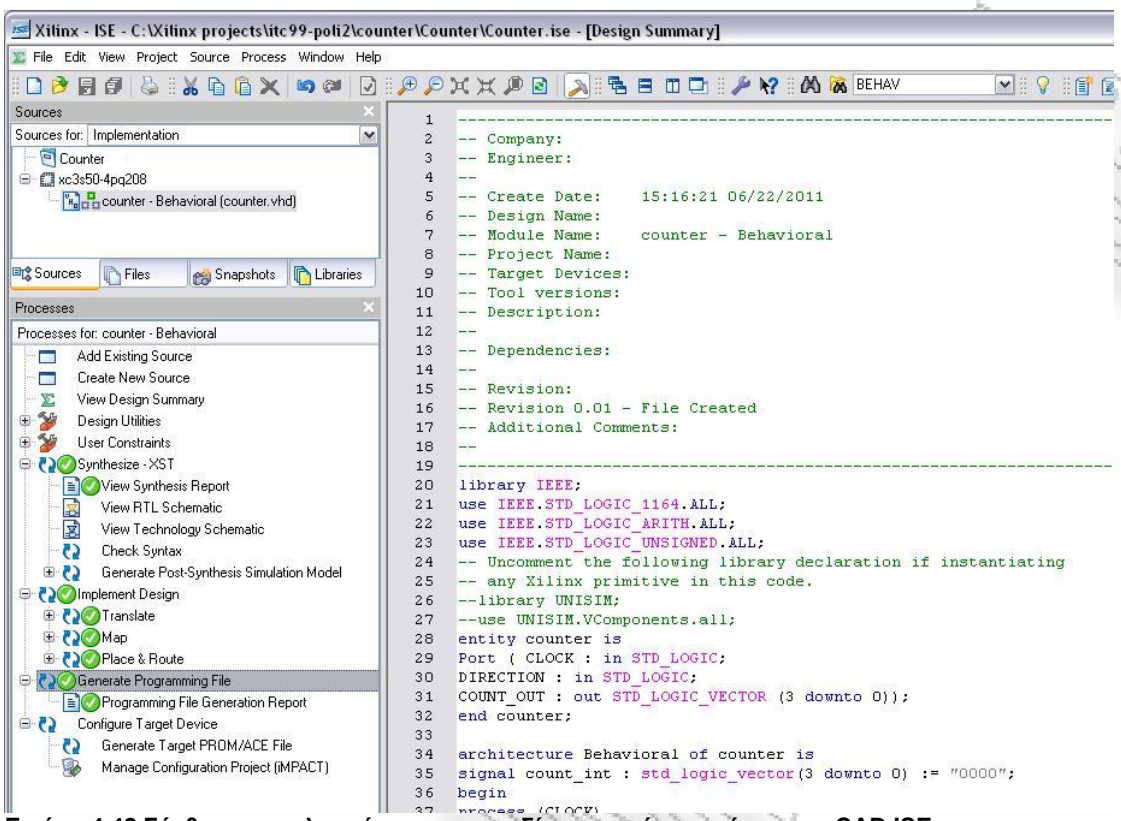
Ο δεύτερος τρόπος υλοποίησης της λειτουργικής προσομοίωσης είναι χρησιμοποιώντας πάγκο εργασίας με χρήση της γλώσσας VHDL. Απλά διαλέγουμε *Project→New Source→VHDL Test Bench*. Ένα ενδεικτικό αρχείο για την κατασκευή κυματομορφών με χρήση της γλώσσας VHDL είναι αυτό στο παράρτημα Β με το κωδικό όνομα *file even_par.vhd* [PsM2]. Κατά τα άλλα ακολουθείται η ίδια διαδικασία με τα παραπάνω για την παραγωγή των κυματομορφών εξόδου αλλά και των εσωτερικών σημάτων.

4.3.5. Σύνθεση και Υλοποίηση

Αν η λειτουργική προσομοίωση είναι σωστή τότε προχωράμε στη σύνθεση της σχεδίασης πατώντας με διπλό click την ένδειξη *Synthesize–XST*. Όταν το ISE εκτελέσει τη σύνθεση μπορούμε να προχωρήσουμε και στην υλοποίηση της σχεδίασης πατώντας με διπλό click την επιλογή *Implement Design*. Αν όλα πάνε καλά τότε θα εμφανιστεί μια πράσινη ένδειξη σε όλες τις επιλογές όπως φαίνεται στην Εικόνα 4.42.

Σημείωση 1 : Αν επιλέξουμε απευθείας να πατήσουμε την υλοποίηση (Implement Design) το ISE αυτόματα θα κάνει πρώτα τη σύνθεση (Synthesize) και μετά θα προχωρήσει στην υλοποίηση.

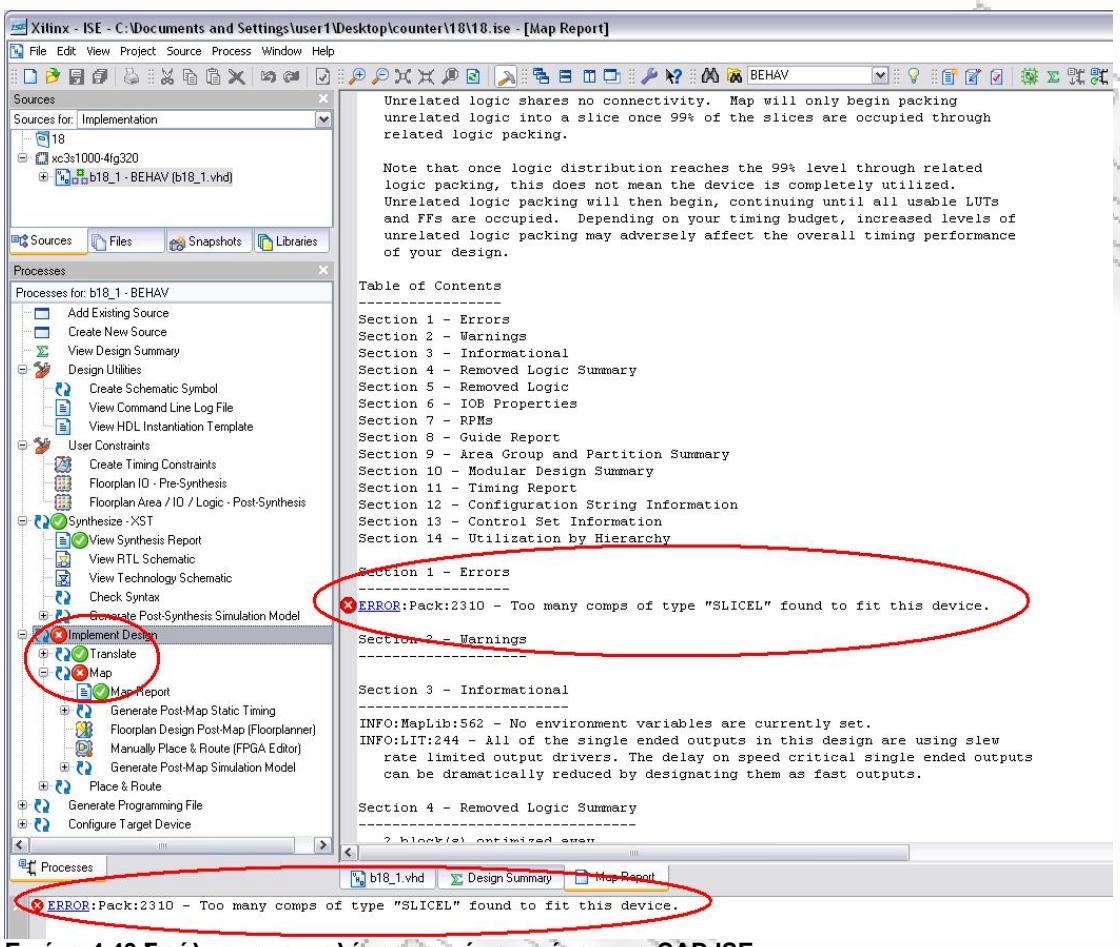
Σημείωση 2 : στη δεύτερη επιλογή Implement Design θα μπορούσε ο χρήστης να εκτελέσει μια μια τις επιλογές *Translate*, *Map* και *Place & Route* με διπλό click. Οι τρεις αυτές ενδείξεις εκτελούν τη χωροθέτηση την τοποθέτηση και τη δρομολόγηση αντίστοιχα όπως έχουμε δει και στην Ενότητα 4.1.



Εικόνα 4.42 Σύνθεση και υλοποίηση μιας σχεδίασης από το πρόγραμμα CAD ISE

Έχοντας ολοκληρώσει σωστά τις παραπάνω ενέργειες μπορούμε να υλοποιήσουμε και το αρχείο BitStream εκτελώντας την επιλογή *Generate Programming File*. Τέλος όταν δημιουργηθεί το αρχείο BitStream, πηγαίνοντας και εκτελώντας την επιλογή *Configure Target Device* φορτώνεται το πρόγραμμα που σχεδιάσαμε στη συσκευή FPGA. Εμείς σε αυτό το σημείο δεν θα προχωρήσουμε μιας και το τελευταίο δεν αποτελεί αντικείμενο μελέτης αυτής της μεταπτυχιακής διατριβής.

Αν σε περίπτωση κάποιο από τα παραπάνω είχε αστοχία στην εκτέλεση, το ISE προειδοποιεί με μηνύματα την ύπαρξη λάθους. Για παράδειγμα βλέπουμε στην Εικόνα 4.43 ένα σφάλμα στην υλοποίηση, η συγκεκριμένη αιτία είναι ότι η σχεδίαση είναι αρκετά μεγάλη για τη συσκευή που έχει προταθεί. Δεν αρκεί στην υφιστάμενη συσκευή ο χώρος για τη χωροθέτηση των λογικών μονάδων, οπότε θα πρέπει να μεταβούμε σε μια μεγαλύτερη συσκευή. Το σφάλμα και η αιτιολόγηση του φαίνονται στην Εικόνα 4.43.



Εικόνα 4.43 Σφάλμα και αιτιολόγηση από το πρόγραμμα CAD ISE

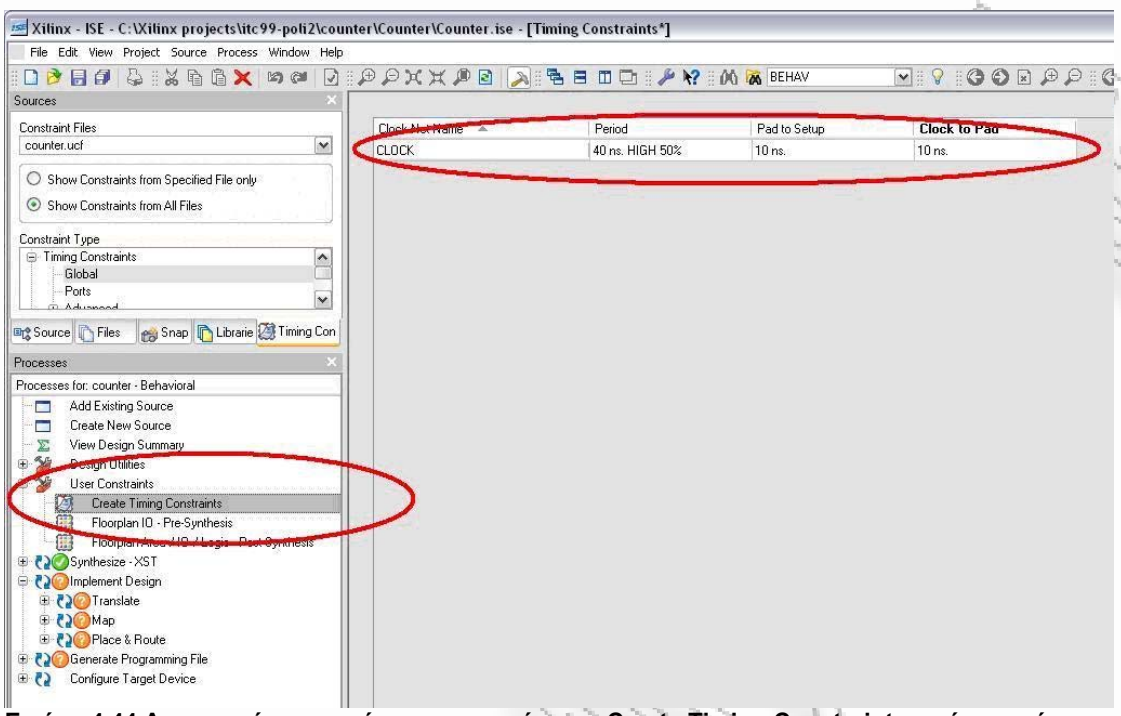
4.3.6. Χρονικοί Περιορισμοί και αρχείο υλοποίησης Design Summary

Μία άλλη σημαντική παράμετρος είναι η δημιουργία χρονικών περιορισμών, αυτοί οι περιορισμοί μπορούν υπάρχουν σε κάποιες συγκεκριμένες χρονικές στιγμές όπου σύμφωνα με τη σχεδίαση και τις αρχικές προδιαγραφές του συστήματος, θα πρέπει να λειτουργούν.

Αυτοί είναι :

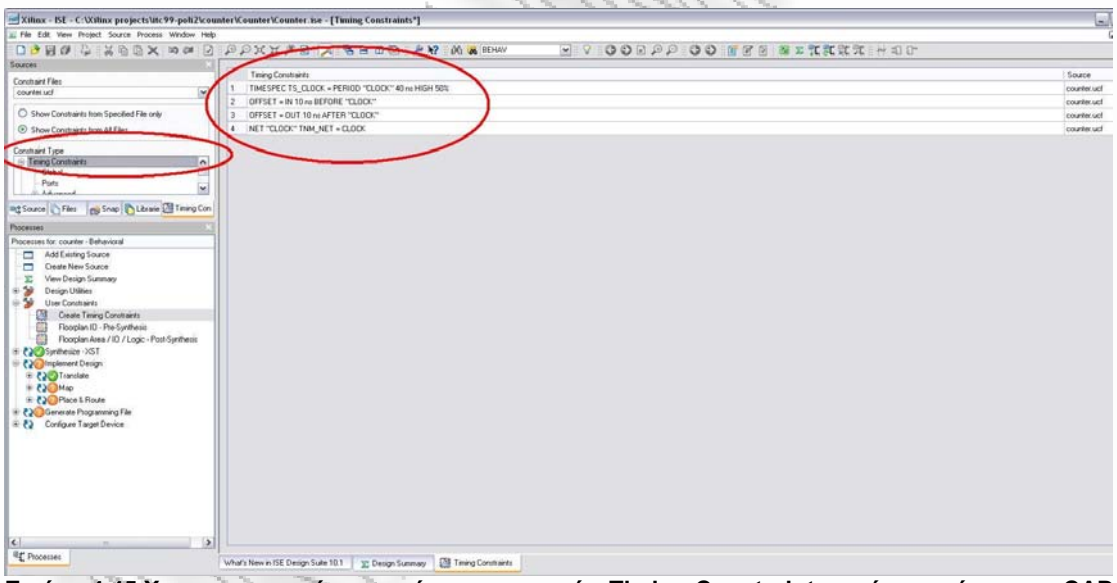
- να θέσουμε χρονικά όρια στην περίοδο του χρόνου ρολογιού,
- την αρχικοποίηση, δηλαδή το *Setup Time* αλλά και
- το χρόνο απόκρισης και διάδοσης των δεδομένων μέσα στους ακροδέκτες, δηλαδή το *Hold Time*.

Έχοντας χρονικούς περιορισμούς θα δούμε αν η σχεδίαση μας ανταποκρίνεται θετικά σε αυτά. Για να το πετύχουμε αυτό θα πρέπει να δημιουργήσουμε ένα αρχείο το *<arxeio>.ucf*, που τα αρχικά σημαίνουν *User Constraints File*. Για να το κάνουμε αυτό πηγαίνουμε στο παράθυρο *Source* επιλέγουμε από την πτυσσόμενη λίστα το *Implementation* και αμέσως μετά το αρχείο *counter.vhdl* (συνέχεια του παραδείγματος μας). Στη συνέχεια πάμε στο παράθυρο *Processes* και επιλέγουμε από το *User Constraints* το *Create Timing Constraints*, αυτόματα το εργαλείο θα ανοίξει το παράθυρο που φαίνεται στην Εικόνα 4.44 στο οποίο ο χρήστης συμπληρώνει τους χρόνους πού θέλει. Εμείς θα συμπληρώσουμε τους χρόνους όπως αυτοί φαίνονται στην Εικόνα 4.44 και στη συνέχεια θα πατήσουμε enter. Παράλληλα με το άνοιγμα του παραθύρου για τη συμπλήρωση των χρόνων, το εργαλείο ISE δημιουργεί το αρχείο *counter.ucf* το οποίο και θα συμπεριλάβει στο φάκελο με όλα τα αρχεία του project.



Εικόνα 4.44 Δημιουργία χρονικών περιορισμών στο Create Timing Constraints από το πρόγραμμα CAD ISE

Όταν τελειώσουμε και θέλουμε να δούμε αυτά τα χαρακτηριστικά πάμε στο *Constraint Type* και επιλέγουμε *Timing Constraints*, θα εμφανιστούν τα χαρακτηριστικά όπως φαίνονται στην Εικόνα 4.45.

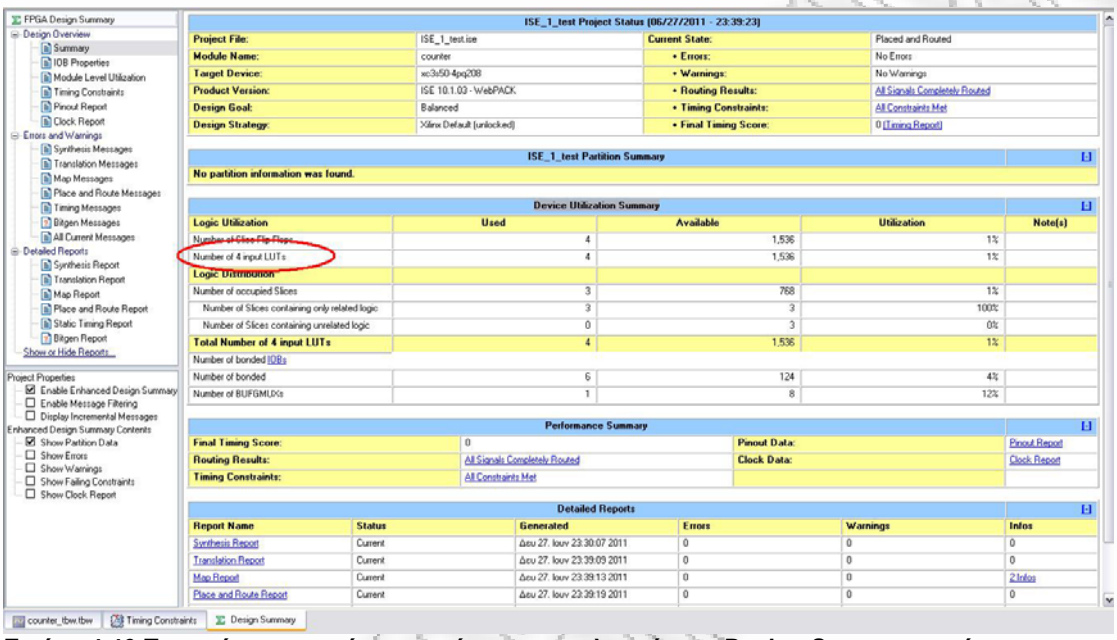


Εικόνα 4.45 Χαρακτηριστικά χρονικών περιορισμών Timing Constraints από το πρόγραμμα CAD ISE

Αφού θέσουμε τους περιορισμούς θα πρέπει να υλοποιήσουμε τη σχεδίαση μας. Εκτελώντας την όπως μάθαμε από τις παραπάνω υποδείξεις διαπιστώνουμε το αποτέλεσμα αυτής της υλοποίησης.

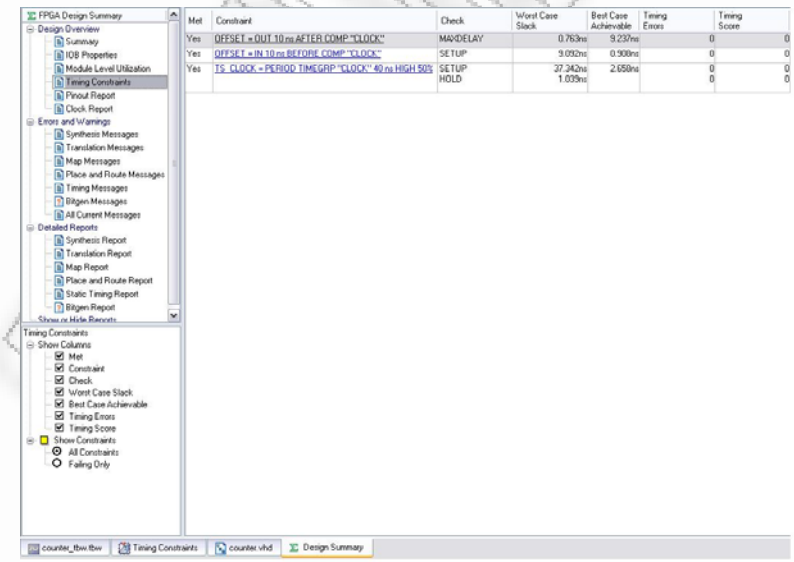
Το ISE (όπως και το Quartus), όταν αρχικά εκτελεί μία σχεδίαση δεν λαμβάνει υπόψη τους χρονικούς περιορισμούς, διότι δεν υπάρχει κάποιο αρχείο χρονικών περιορισμών. Αυτό που κάνει είναι να εκτελεί την αρχική σχεδίαση και μετά να βγάζει αναφορές σχετικά με τις όποιες καθυστερήσεις αλλά και την προτεινόμενη συχνότητα λειτουργίας της σχεδίασης. Το ISE βγάζει αναφορές για την περίοδο του ρολογιού από όπου κανείς μπορεί να βγάλει τη μέγιστη συχνότητα λειτουργίας, εν αντιθέσει του Quartus που το βγάζει απευθείας με το FMax. Επιπλέον το ISE βγάζει τους προτεινόμενους χρόνους Setup Time και Hold Time που είδαμε στην Ενότητα 4.2.6, κάτι το οποίο δεν κάνει το Quartus και το οποίο τα ενσωματώνει αυτόματα.

Το ISE σε κάθε υλοποίηση παράγει ένα αρχείο με όλα τα στατιστικά στοιχεία που διέπουν μια υλοποίηση. Για να δούμε το αρχείο αυτό και όλα τα χαρακτηριστικά αρκεί να πάμε στο παράθυρο Processes και να επιλέξουμε το *View Design Summary*. Στο παράθυρο που θα ανοίξει θα δούμε κάποια χαρακτηριστικά τα οποία έχουν ομαδοποιηθεί. Για παράδειγμα αν θέλουμε να δούμε πόσα LUT έχουν χρησιμοποιηθεί στη σχεδίαση μας αρκεί να δούμε στο *Device Utilization Summary* το *Number of 4-input LUT* Εικόνα 4.46, να θυμηθούμε ότι κάθε LUT αποτελεί ένα λογικό στοιχείο στην αρχιτεκτονική του Spartan-3. Παράλληλα μπορεί κανείς να δει και άλλα χαρακτηριστικά όπως ενεργοποιημένοι ακροδέκτες είσοδο εξόδου χρησιμοποιημένη μνήμη RAM κτλ κτλ. Πάντα όμως όλα έχουν σχέση με την εκάστοτε σχεδίαση και τη συσκευή που έχει επιλεγεί για υλοποίηση.



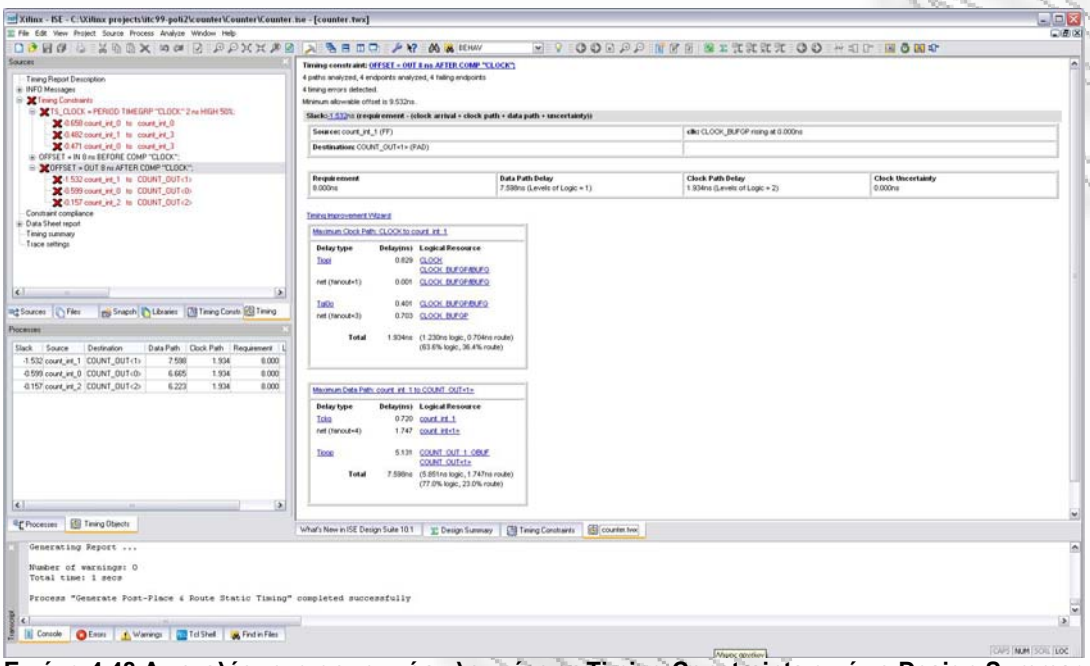
Εικόνα 4.46 Παραγόμενο αρχείο τεκμηρίωσης της υλοποίησης Design Summary από το πρόγραμμα CAD ISE

Ιδιαίτερη αναφορά πρέπει να γίνει για τους χρονικούς περιορισμούς που έχουμε βάλει από την παραπάνω διαδικασία. Στην επιλογή *Performance Summary* διαλέγοντας το *All Constraints Met* μπορούμε να δούμε τα αποτελέσματα. Στο νέο παράθυρο που θα εμφανιστεί μπορεί κανείς να δει τη μεγαλύτερη καθυστέρηση που σημειώθηκε στην ένδειξη *MAXDELAY*, τα *SETUP* και *HOLD TIME* Εικόνα 4.47. Η ίδια επιλογή μπορεί να ενεργοποιηθεί και από την επιλογή πάνω αριστερά *Design Overview* πατώντας την ένδειξη *Timing Constraints*.



Εικόνα 4.47 Αποτελέσματα υλοποίησης Timing Constraints από το Design Summary του προγράμματος CAD ISE

Αν τώρα κάποιος περιορισμός δεν είναι επιτρεπτός αυτό φαίνεται με την αρνητική τιμή που έχουν οι ενδείξεις MAXDELAY, τα SETUP και HOLD TIME. Για λεπτομέρεια μπορούμε να πατήσουμε πάνω σε κάποια ένδειξη και να εμφανιστεί το αρχείο **.twx* το οποίο αποτελεί μια αναφορά σε κάθε ένα από τα προαναφερόμενα. Για αρνητικές τιμές ή χρόνους που δεν καλύπτουν τους χρονικούς περιορισμού της σχεδίασης, απεικονίζονται με κόκκινο, Εικόνα 4.48.



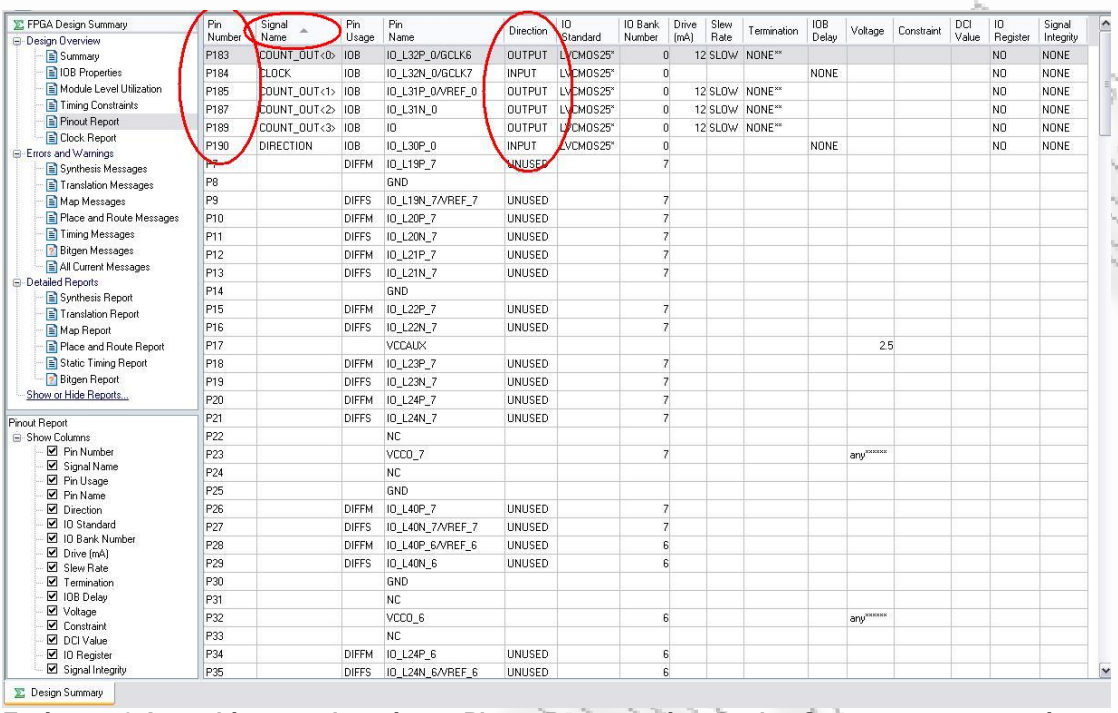
Εικόνα 4.48 Αποτελέσματα αρνητικής υλοποίησης Timing Constraints από το Design Summary του προγράμματος CAD ISE

Από τους όποιους περιορισμούς μπορούμε να συμπεράνουμε τη συχνότητα λειτουργίας αλλά και τους χρόνους Setup Time και Hold Time. Αυτές οι λειτουργίες μας ενδιαφέρουν πολύ γιατί θα αποτελέσουν αντικείμενο μελέτης και παρακολούθησης στο πειραματικό μέρος.

4.3.7. Ακροδέκτες εισόδου εξόδου και Pinout Report

Η χαρτογράφηση των ακροδεκτών εισόδου εξόδου δε θα παρουσιαστεί στη συνέχεια, μιας και το αντικείμενο μελέτης της μεταπτυχιακής διατριβής δεν έχει σχέση με αυτή τη λειτουργία. Ουσιαστικά μπορεί ο χρήστης να επιλέξει από μόνος του τους ακροδέκτες εισόδου και εξόδου αλλά και των άλλων χαρακτηριστικών όπως ρολόι κτλ.

Απλά αυτό που έχει ενδιαφέρον και πρέπει να δούμε είναι η αναφορά που βγάζει το ISE για την ενεργοποίηση των ακροδεκτών. Επιλέγουμε από το Design Summary το *Pinout Report*, το οποίο θα μας δείχνει ποιοι ακροδέκτες χρησιμοποιούνται σε μια σχεδίαση. Σε συνέχεια της σχεδίασης μας θα δούμε το Pinout Report του Counter. Για να δούμε ποιοι ακροδέκτες χρησιμοποιούνται πατάμε την επιλογή *Signal Name* και αυτόματα θα εμφανιστούν στην αρχή, παράλληλα στην επιλογή *Direction* περιγράφουν πια χρήση επιτελούν. Όπως βλέπουμε από το report χρησιμοποιούνται έξη ακροδέκτες δυο για Input και τέσσερις για Output, Εικόνα 4.49. Μια άλλη παρόμοια αλλά πιο σύντομη διαδικασία είναι να πατήσουμε την επιλογή IOB Properties η οποία εμφανίζει μόνο τους ακροδέκτες που χρησιμοποιούνται.



Pin Number	Signal Name	Pin Usage	Pin Name	Direction	IO Standard	IO Bank Number	Drive (mA)	Slew Rate	Termination	IOB Delay	Voltage	Constraint	DCI Value	IO Register	Signal Integrity
P183	COUNT_OUT<0>	IOB	IO_L32P_0/GCLK6	OUTPUT	LVCMOS25*	0	12	SLOW	NONE**					NO	NONE
P184	CLOCK	IOB	IO_L32N_0/GCLK7	INPUT	LVCMOS25*	0				NONE				NO	NONE
P185	COUNT_OUT<1>	IOB	IO_L31P_0/VREF_0	OUTPUT	LVCMOS25*	0	12	SLOW	NONE**					NO	NONE
P187	COUNT_OUT<2>	IOB	IO_L31N_0	OUTPUT	LVCMOS25*	0	12	SLOW	NONE**					NO	NONE
P189	COUNT_OUT<3>	IOB	IO	OUTPUT	LVCMOS25*	0	12	SLOW	NONE**					NO	NONE
P190	DIRECTION	IOB	IO_L30P_0	INPUT	LVCMOS25*	0				NONE				NO	NONE
P7		DIFFM	IO_L19P_7	UNUSED		7									
P8			GND												
P9		DIFFS	IO_L19N_7/VREF_7	UNUSED		7									
P10		DIFFM	IO_L20P_7	UNUSED		7									
P11		DIFFS	IO_L20N_7	UNUSED		7									
P12		DIFFM	IO_L21P_7	UNUSED		7									
P13		DIFFS	IO_L21N_7	UNUSED		7									
P14			GND												
P15		DIFFM	IO_L22P_7	UNUSED		7									
P16		DIFFS	IO_L22N_7	UNUSED		7									
P17			VCCAUX								2.5				
P18		DIFFM	IO_L23P_7	UNUSED		7									
P19		DIFFS	IO_L23N_7	UNUSED		7									
P20		DIFFM	IO_L24P_7	UNUSED		7									
P21		DIFFS	IO_L24N_7	UNUSED		7									
P22			NC												
P23			VCCO_7			7					any******				
P24			NC												
P25			GND												
P26		DIFFM	IO_L40P_7	UNUSED		7									
P27		DIFFS	IO_L40N_7/VREF_7	UNUSED		7									
P28		DIFFM	IO_L40P_6/VREF_6	UNUSED		6									
P29		DIFFS	IO_L40N_6	UNUSED		6									
P30			GND												
P31			NC												
P32			VCCO_6			6					any******				
P33			NC												
P34		DIFFM	IO_L24P_6	UNUSED		6									
P35		DIFFS	IO_L24N_6/VREF_6	UNUSED		6									

Εικόνα 4.49 Αποτελέσματα υλοποίησης Pinout Report από το Design Summary του προγράμματος CAD ISE

Σε αυτή την ενότητα είδαμε πως μπορούμε να σχεδιάσουμε και να δημιουργήσουμε αρχεία VHDL με το εργαλείο σχεδίασης ISE. Συγκεκριμένα είδαμε το τρόπο σχεδίασης με κώδικα VHDL, πως ελέγχουμε τη σχεδίαση μας αλλά και τη διαδικασία υλοποίησης για την παραγωγή του αρχείου Bitstream. Είδαμε το τρόπο που θέτουμε χρονικούς περιορισμούς σε μία σχεδίαση αλλά και την τελική σύνοψη της σχεδίασης μας σε ένα παράθυρο όπως και σε διάφορα αρχεία log.

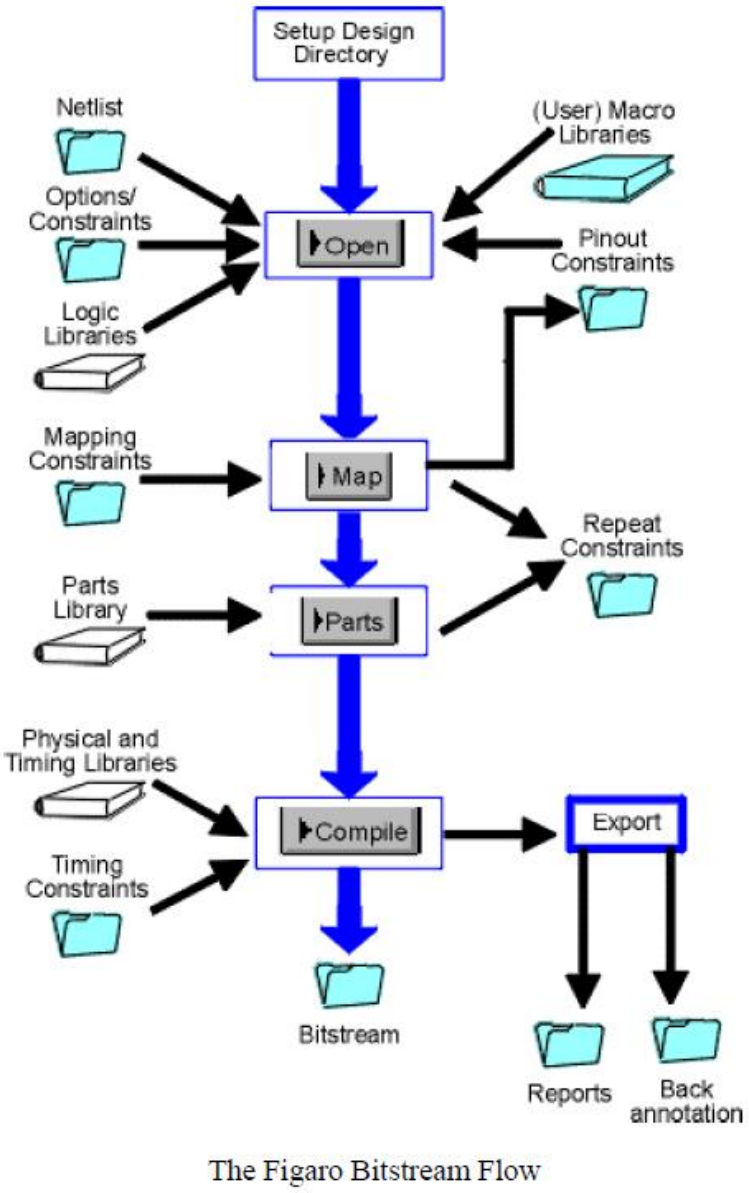
Για περεταίρω μελέτη του εργαλείου σχεδίασης ISE της εταιρίας Xilinx δείτε το αντίστοιχο έγγραφο *ISE 10.1 Quick Start Tutorial* [Xln3] το οποίο αναφέρεται στη βιβλιογραφία.

4.4. Σχεδίαση με χρήση εργαλείου Atmel IDS Figaro

Η τρίτη και τελευταία παρουσίαση έχει να κάνει με το εργαλείο σχεδίασης *Integrated Development System IDS* ή αλλιώς *Figaro* της εταιρίας Atmel και συγκεκριμένα θα δούμε την έκδοση 7.6. Όπως ήδη έχουμε προαναφέρει λόγω της μη ύπαρξης άδειας χρήσης δε μπορούμε να προβούμε σε μια ολοκληρωμένη και λεπτομερή παρουσίαση.

4.4.1. Εισαγωγή στο IDS Figaro

Το IDS Figaro όπως και τα άλλα σχεδιαστικά εργαλεία CAD παρέχει μια ποικιλία από σχεδιαστικά πρότυπα με τα οποία μπορεί κανείς να σχεδιάσει. Όπως και στα άλλα έτσι και εδώ θα δούμε τη σχεδίαση μέσω της γλώσσας VHDL. Όταν σχεδιάζουμε με το IDS Figaro μπορούμε να παράγουμε δύο αρχεία, το ένα είναι το εκτελέσιμο αρχείο Bitsream και το δεύτερο είναι η παραγωγή ενός αρχείου επονομαζόμενο *μακροεντολή* (UDMs, *User Define Macros*) το οποίο αποθηκεύεται και χρησιμοποιείται για μελλοντική χρήση. Όταν ο χρήστης φτάσει σε ένα σημείο επιλέγει τι από τα δυο θέλει να κάνει. Οι δυο χρήσεις εκτελούν διαφορετικές ενέργειες στο IDS Figaro, στην Εικόνα 4.50 βλέπουμε πως παράγεται το αρχείο Bitstream. Εμείς θα αναπτύξουμε την πρώτη περίπτωση μιας και είναι αυτή που μας ενδιαφέρει στην παρούσα διατριβή. Η δεύτερη περίπτωση απλά αναφέρεται για ενημερωτικούς λόγους.



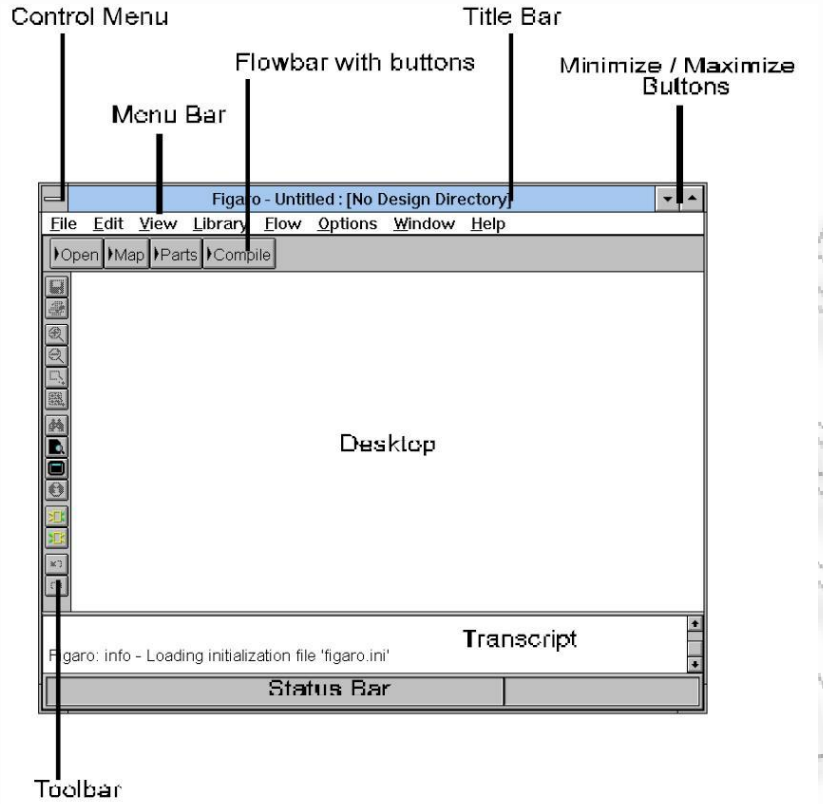
Εικόνα 4.50 [Atm2] Γενική μεθοδολογία σχεδίασης παραγωγής Bitstream αρχείου στο πρόγραμμα CAD IDS Figaro

Όπως προαναφέραμε η γενική μεθοδολογία σχεδίασης για την παραγωγή του αρχείου Bitstream είναι αυτή της Εικόνας 4.50, όπως μπορείτε να δείτε αποτελείται από τέσσερα απλά βήματα τα οποία είναι

1. *Setup and Design Directory*, εγκατάσταση και καθορισμός καταλόγου προορισμού.
2. *Open Design*, άνοιγμα αρχείου σχεδίασης της προτίμησης μας.
3. *Parts*, καθορισμός των μερών της σχεδίασης στη συσκευή που μπορεί να προστεθούν είτε από τη βιβλιοθήκη αυτόματα είτε από τη σχεδιαστική ικανότητα του χρήστη. Το βήμα αυτό περιλαμβάνει τη χωροθέτηση (*Mapping*) και το καθορισμό των ακροδεκτών εισόδου εξόδου (*Pinou*).
4. Και τέλος *Compile*, την εκτέλεση του σχεδίου για την παραγωγή του bitstream αρχείου.

Επιπλέον βλέπουμε στην Εικόνα 4.50 κάποιες πληροφορίες που φαίνονται και στις δύο πλευρές των βημάτων σχεδίασης, κάποιες από αυτές είναι προαιρετικές και κάποιες άλλες χρησιμοποιούνται αυτόματα από το εργαλείο IDS Figaro.

Το περιβάλλον εργασίας του Figaro είναι το Project Navigator το οποίο φαίνεται στην Εικόνα 4.51.



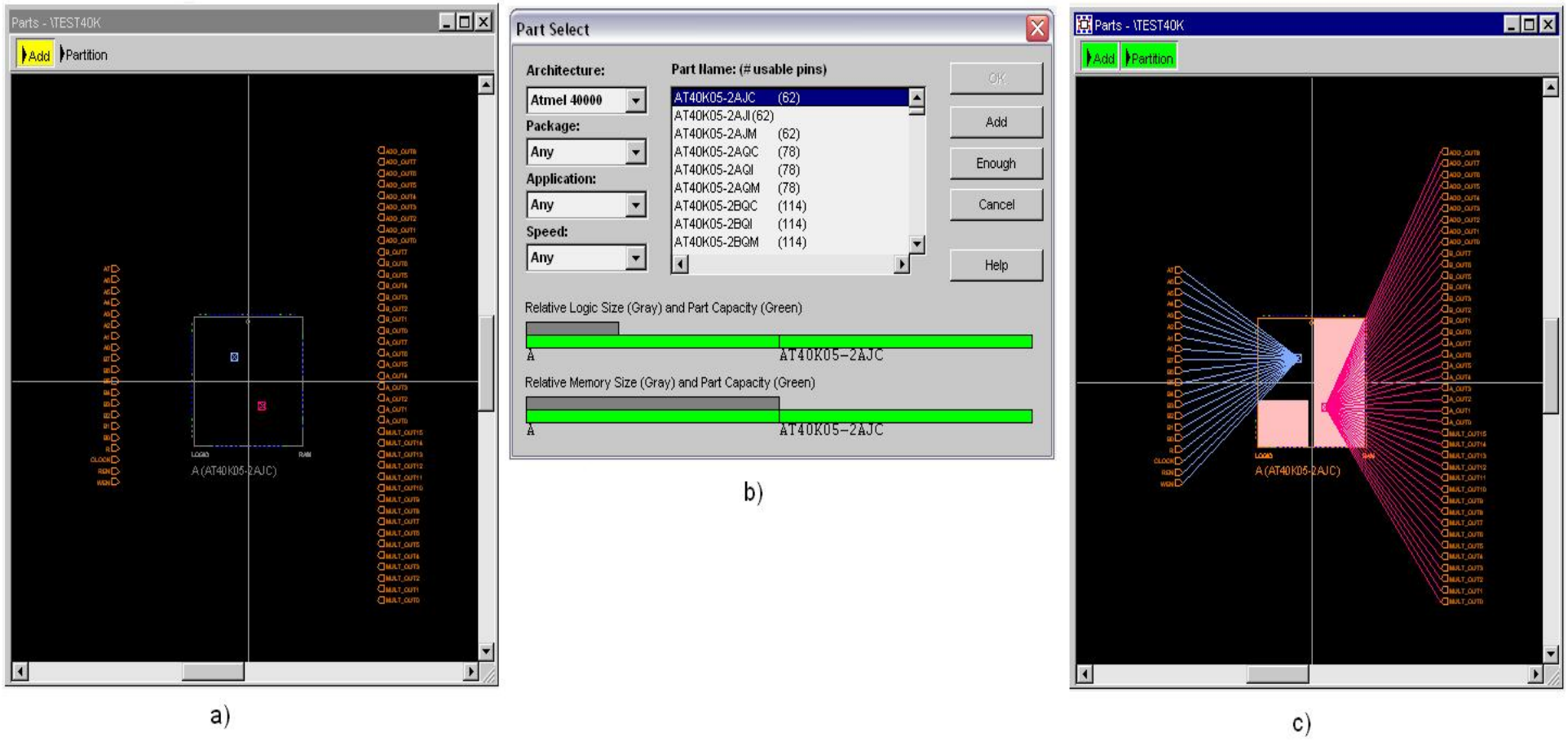
Εικόνα 4.51 [Atm2] Τυπικό παράθυρο σχεδίασης στο πρόγραμμα CAD IDS Figaro

Το Figaro έχει τέσσερα διαδραστικά εργαλεία – παράθυρα τα οποία είναι :

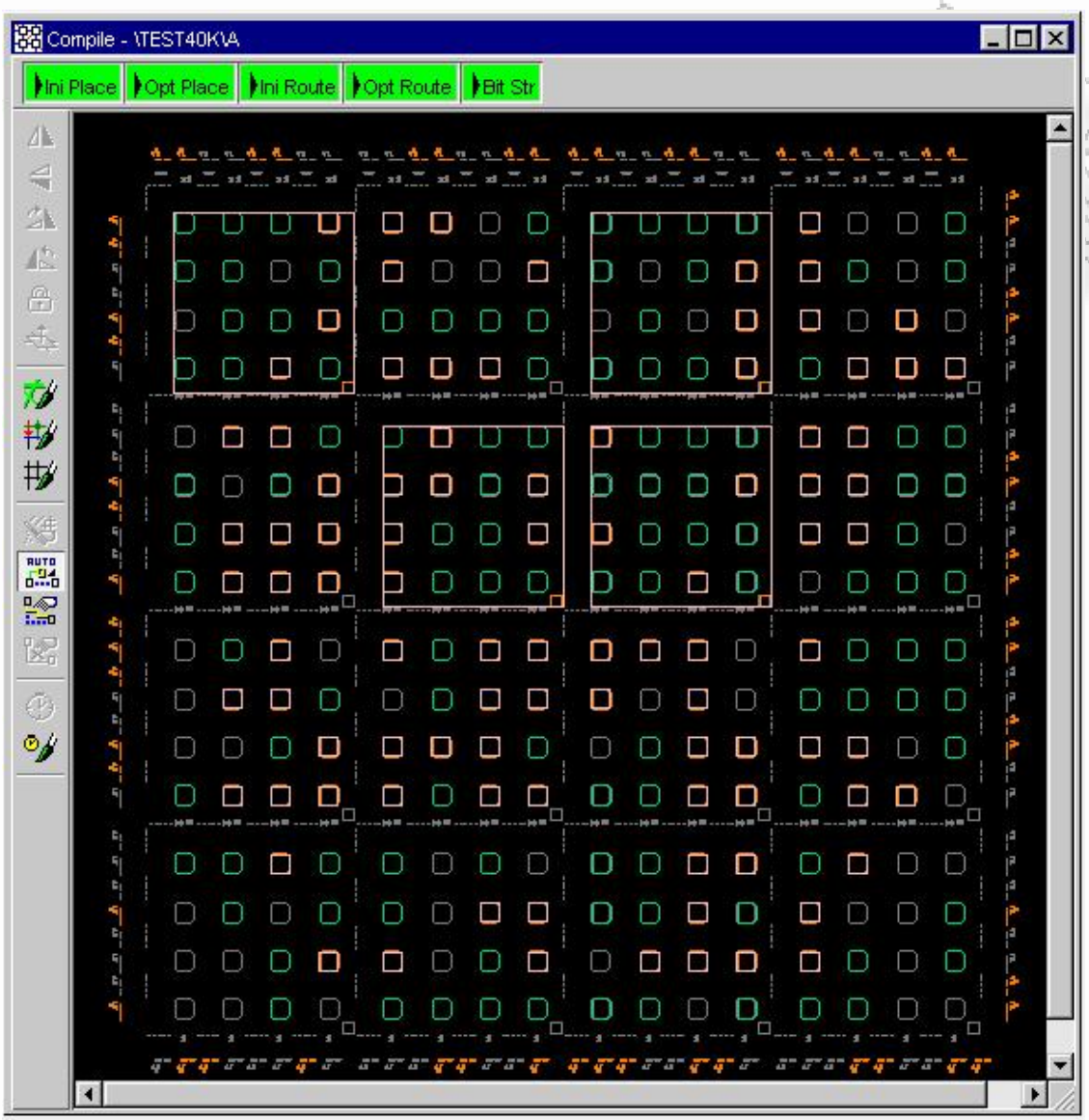
1. Το *Design Browser* το οποίο εμφανίζει τις λογικές μονάδες σχεδιασμού που εισάγονται στη σχεδίαση, ιεραρχημένες. Η ιεραρχική σχεδίαση φαίνεται μέσα στο Design Browser. Οποιαδήποτε λογική μονάδα έχει υλοποιηθεί βρίσκεται μέσα σε ένα ορθογώνιο και φαίνεται ως μια ετικέτα (label box), ως ένα ορθογώνιο. Αν η μονάδα αυτή δεν έχει τοποθετηθεί (placed) τότε το ορθογώνιο είναι κενό Εικόνα 4.52 το a), αν έχει τοποθετηθεί τότε είναι γεμάτο Εικόνα 4.52 το b).
2. Το *Map Browser* εμφανίζει τα λογικά στοιχεία και την ιεράρχηση της σχεδίασης μας. Επίσης κατά τη διάρκεια της χωροθέτησης (mapping) όποιες αλλαγές έχει υποστεί ένα λογικό στοιχείο, αυτές λαμβάνονται υπόψη και αποθηκεύονται από το Figaro. Οι αλλαγές αυτές είναι εμφανείς στα διάφορα ονόματα των βοηθητικών στοιχείων αλλά και στα προγράμματα περιήγησης. Το Map Browser χρησιμοποιεί τις ίδιες γραφικές συμβάσεις όπως και το Design Browser αλλά και την ίδια λειτουργία, δηλαδή αν η μονάδα αυτή δεν έχει τοποθετηθεί (placed) τότε το ορθογώνιο είναι κενό Εικόνα 4.53 το a), αν έχει τοποθετηθεί τότε είναι γεμάτο Εικόνα 4.53 το b).
3. Το *Parts Window* εμφανίζει τα πακέτα των συσκευών FPGAs που έχει διαθέσιμα, Εικόνα 4.54 b). Γίνεται επιλογή της συσκευής και «κλειδώνουμε» τους ακροδέκτες εξόδου (pinout), Εικόνα 4.54 a). Όταν γίνει και η εκτέλεση (compiling) της σχεδίαση τότε φαίνονται και οι διασυνδέσεις αυτών, Εικόνα 4.54 c).
4. Το *Compile Window*, Compilation είναι ο συνδυασμός της τοποθέτησης (Placed) και της δρομολόγησης (Routing). Το παράθυρο Compile δείχνει την αρχιτεκτονική της συσκευής καθώς και τα αποτελέσματα της εκτέλεσης, τα οποία είναι η τοποθέτηση των δυναμικών μονάδων στη συσκευή όπως και η δρομολόγηση (διασύνδεση) αυτών μεταξύ τους, Εικόνα 4.55. Όπως μπορείτε να διαπιστώσετε μπορεί ο χρήστης να εκτελέσει ένα ένα τα βήματα η απλός πατώντας το κουμπί Compile να εκτελεστούν όλα αυτόματα. Σε αυτό το βήμα επίσης παράγεται και το bitstream αρχείο. Επιπλέον αν έχουμε εισάγει χρονικούς περιορισμούς στη σχεδίαση μας μπορούμε να δούμε τις λεπτομέρειες στο δίκτυο σημάτων όπου το παράθυρο Compile μεταβαίνει σε κατάσταση χρονικής ανάλυσης (Timing Analysis).

Εικόνα 4.52 Παράθυρο Design Browser στο πρόγραμμα CAD IDS Figaro πριν την εκτέλεση (Compilation) και μετά

Εικόνα 4.53 Παράθυρο Map Browser στο πρόγραμμα CAD IDS Figaro πριν την εκτέλεση (Compilation) και μετά



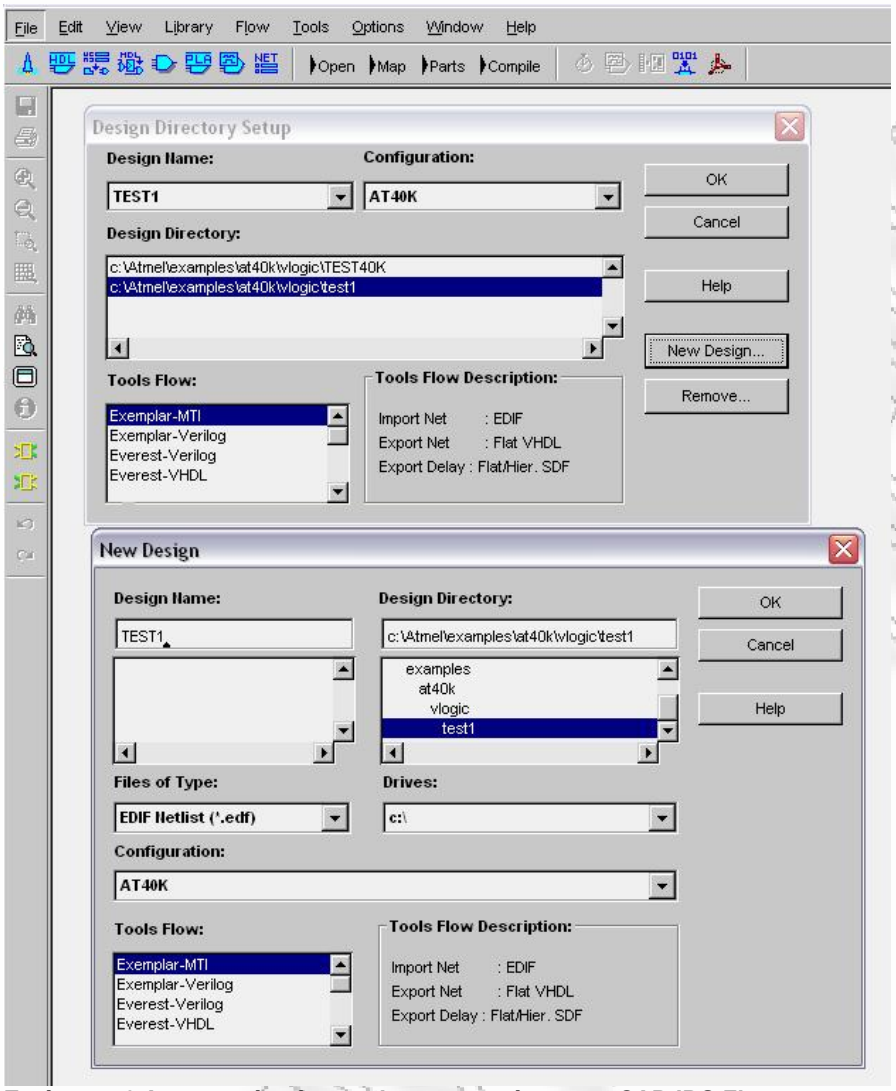
Εικόνα 4.54 Παράθυρο Parts στο πρόγραμμα CAD IDS Figaro πριν την εκτέλεση (Compilation) και μετά



Εικόνα 4.55 Παράθυρο Compile μετά την εκτέλεση (Compilation) της σχεδίασης στο πρόγραμμα CAD IDS Figaro

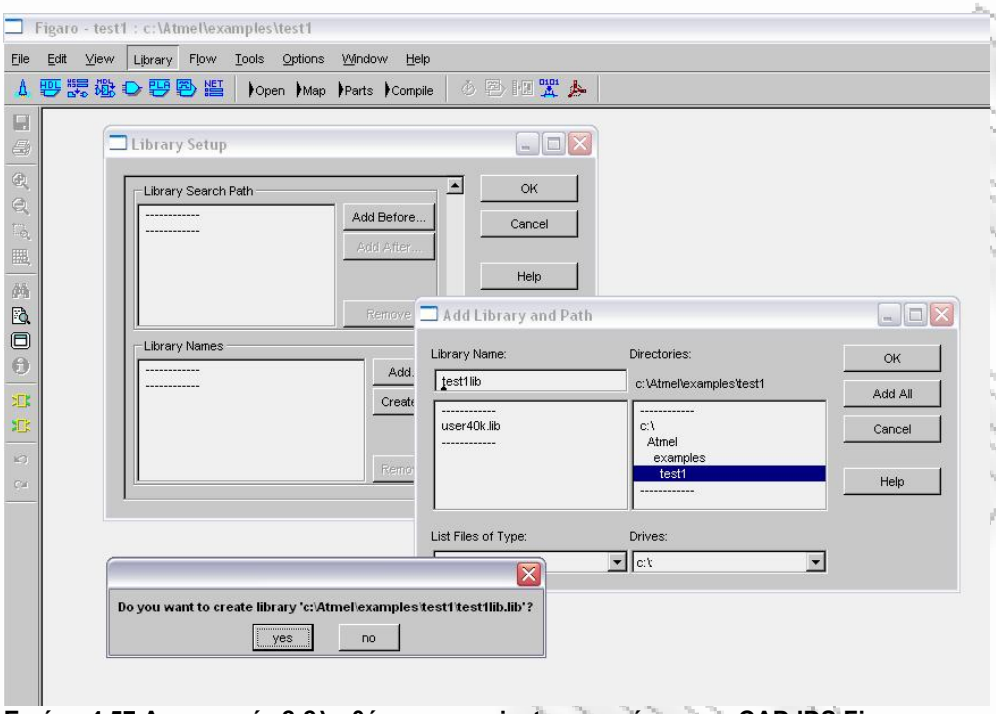
4.4.2. Δημιουργία νέου Project

Για την κατασκευή ενός αρχείου VHDL θα πρέπει να κάνουμε τα εξής. Χρησιμοποιώντας το λειτουργικό βοήθημα του Figaro το *HDL Planner* θα γράψουμε κώδικα. Για να γίνει αυτό αρκεί να πάμε *File→Design Setup* και στο παράθυρο που εμφανίζεται πατάμε *New Design*. Στο νέο παράθυρο καθορίζουμε το φάκελο προορισμού αλλά και το όνομα αρχείου έστω το ονομάζουμε *Test1*, επιλέγουμε *Examplar – MTI* και μετά OK και OK, Εικόνας 4.56.



Εικόνα 4.56 Δημιουργία νέου project στο πρόγραμμα CAD IDS Figaro

Κατόπιν θα πρέπει να επιλέξουμε που θα είναι η βιβλιοθήκη μας, για να γίνει αυτό πάμε *Library→Library Setup*. Στο παράθυρο διαλόγου που θα εμφανιστεί επιλέγουμε *Add Before*, στο νέο παράθυρο διαλόγου επιλέγουμε το φάκελο στο οποίο έχουμε το project μας όπως και το όνομα για τη βιβλιοθήκη που θα δημιουργήσουμε. Με δυο OK επιστρέφουμε στην αρχική οθόνη, ίσως υπάρξει ένα μήνυμα που θα μας ζητήσει να δημιουργήσει τη βιβλιοθήκη, σε κάθε περίπτωση πατάμε OK, Εικόνα 4.57.

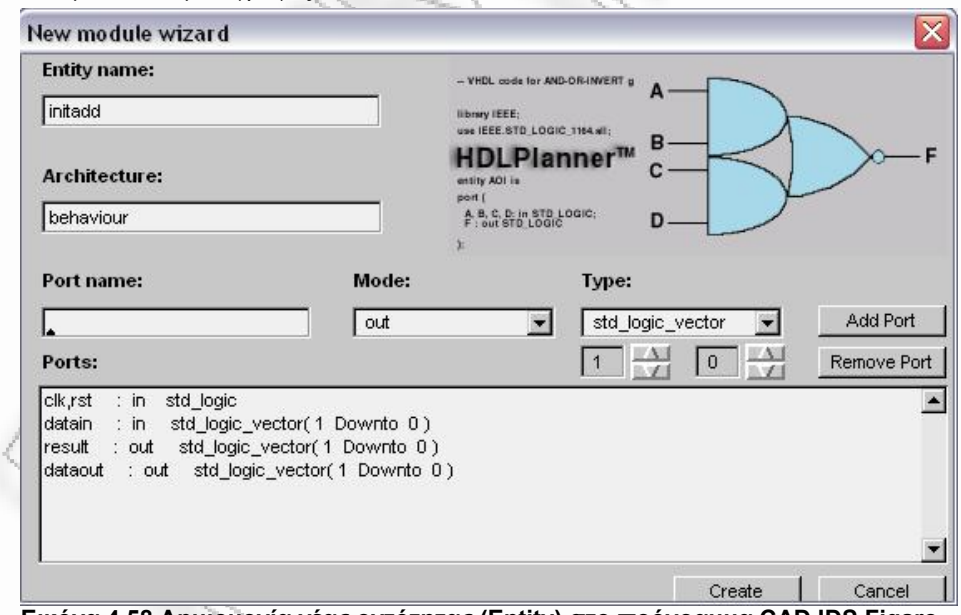


Εικόνα 4.57 Δημιουργία βιβλιοθήκης σε project στο πρόγραμμα CAD IDS Figaro

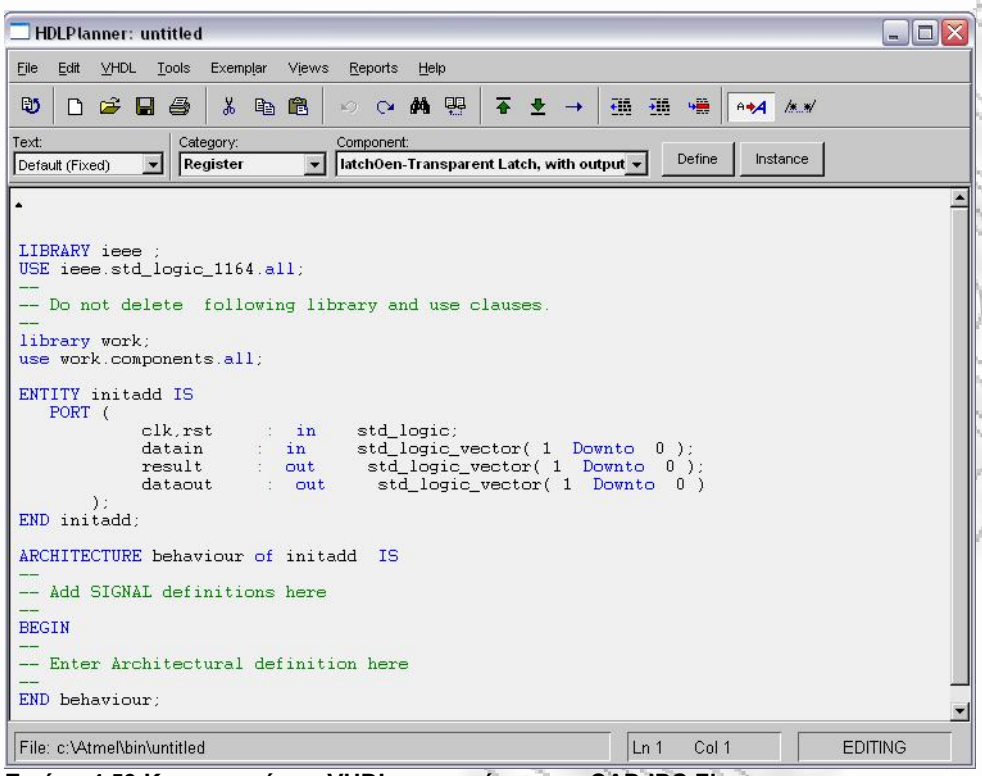
Με την παραπάνω διαδικασία καταφέραμε να δημιουργήσουμε τις κατάλληλες προϋποθέσεις ώστε να αρχίσουμε να γράφουμε το κώδικα που θέλουμε.

4.4.3. Σχεδίαση με γλώσσα VHDL

Έχοντας ολοκληρώσει με επιτυχία την προηγούμενη ενότητα μπορούμε να ξεκινήσουμε τη σχεδίαση μας με τη γλώσσα VHDL. Για να γίνει αυτό επιλέγουμε *Tool→HDL Planer* όπου εμφανίζεται ένα παράθυρο στο οποίο κανείς μπορεί δημιουργήσει μια οντότητα (Entity) με τις εισόδους και εξόδους της, Εικόνα 4.58. Αφού ορίσουμε τις εισόδους και τις εξόδους της νέας οντότητας, πατώντας Create θα ανοίξει αυτόματα ο κειμενογράφος VHDL, Εικόνα 4.59.

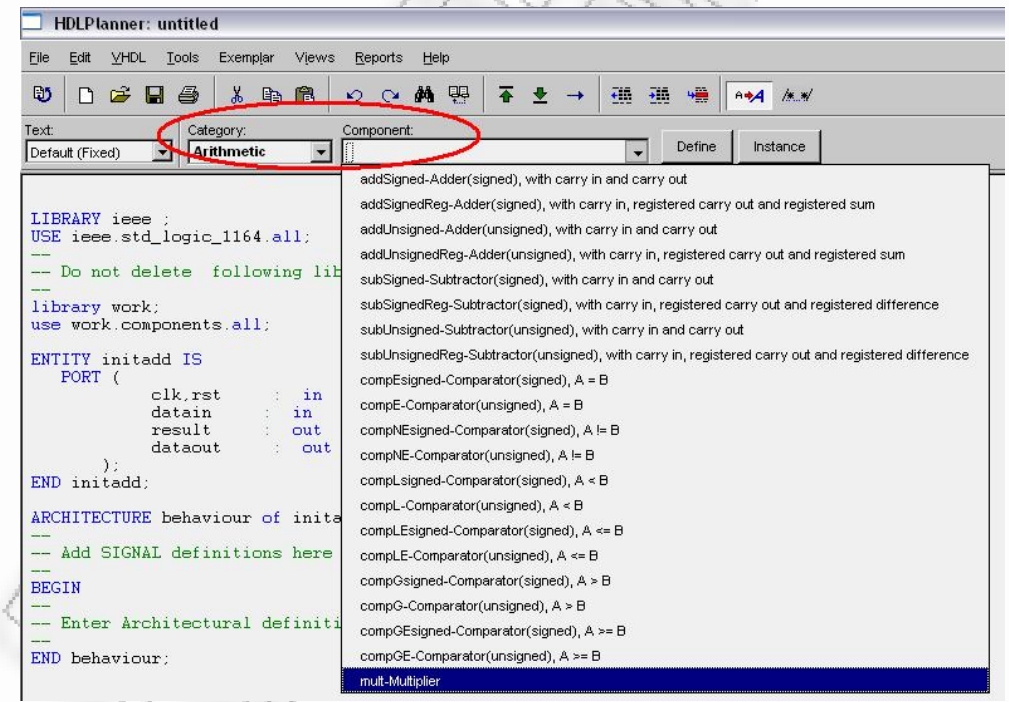


Εικόνα 4.58 Δημιουργία νέας οντότητας (Entity) στο πρόγραμμα CAD IDS Figaro



Εικόνα 4.59 Κειμενογράφος VHDL στο πρόγραμμα CAD IDS Figaro

Από το κειμενογράφο μπορούμε να εισάγουμε έτοιμες μονάδες λογικής που έχουν υλοποιηθεί από το κατασκευαστή για τη διευκόλυνση μας. Αυτό γίνεται αν επιλέξουμε από το *Category* το τύπο που επιθυμούμε και μετά από το *Component* τη μονάδα της επιλογής μας, Εικόνα 4.60.



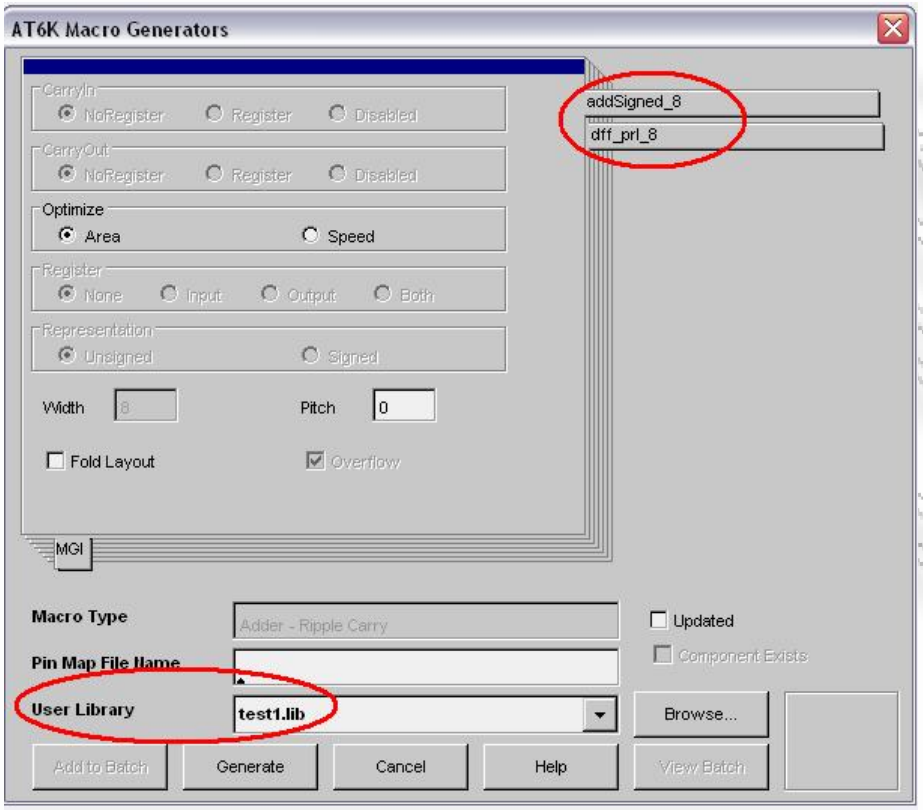
Εικόνα 4.60 Επιλογή έτοιμης λογικής μονάδος από βιβλιοθήκη στο πρόγραμμα CAD IDS Figaro

Εμείς το μόνο που θα κάνουμε είναι να γράψουμε το κώδικα που εμφανίζεται στην Εικόνα 4.61, αυτός ο κώδικας υλοποιεί έναν αθροιστή των οκτώ bit. Ο κώδικας αυτός παρουσιάζεται στο παράρτημα Β και έχει ονομασία *initadd.vhd*.

```
-- Do not delete following library and use clauses.
library atmel;
use atmel.components.all;
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY initadd IS
   generic(WIDTH : integer :=8);
      PORT (
         clk, rst : IN std_logic;
         datain : IN std_logic_vector(WIDTH-1 downto 0);
         dataout : OUT std_logic_vector(WIDTH-1 downto 0);
         result : OUT std_logic_vector(WIDTH downto 0)
      );
END initadd;

ARCHITECTURE behaviour OF initadd IS

SIGNAL inta,temp_dataout : std_logic_vector(WIDTH-1 downto 0);
SIGNAL zero : std_logic;
SIGNAL unused : std_logic;

BEGIN
zero <= '0';
-- HDLPlanner Instance dff_prl
-- Do not **DELETE** previous line
u1 : dff_prl
   GENERIC MAP (WIDTH => 8)
      PORT MAP(
         DATA => datain,
         RN => rst,
         CLK => clk,
         Q => inta
      );

-- HDLPlanner Instance dff_prl
-- Do not **DELETE** previous line
u2 : dff_prl
   GENERIC MAP (WIDTH => 8)
      PORT MAP(
         DATA => inta,
         RN => rst,
         CLK => clk,
         Q => temp_dataout
      );

-- HDLPlanner Instance addSigned
-- Do not **DELETE** previous line
u3 : addSigned
   GENERIC MAP (WIDTH => 8)
      PORT MAP (
         DATAA => inta,
         DATAB => temp_dataout,
         CIN => zero,
         SUM => result(WIDTH-1 downto 0),
         COUT => result(WIDTH),
         OVERFLOW => unused
      );
dataout <= temp_dataout;
END behaviour;
```

Εικόνα 4.61 [Atm2] Κώδικας VHDL για την υλοποίηση αθροιστής των 8 bit σε γλώσσα VHDL

Αφού τροποποιήσουμε τον κώδικά μας μπορούμε να σώσουμε το αρχείο που μόλις δημιουργήσαμε. Μετά από όλα αυτά θα μπορούσαμε, αν θέλαμε, να αποθηκεύσουμε τη σχεδίαση μας για μελλοντική χρήση, αυτό το πετυχαίνουμε με την επιλογή *Tools→Invoke Macro Generators*. Θα εμφανιστεί το παράθυρο της Εικόνας 4.62 στο οποίο βλέπουμε τα συστατικά από τα οποία αποτελείται η σχεδίαση μας αλλά και σε ποια βιβλιοθήκη προτείνεται να αποθηκευθεί.



Εικόνα 4.62 Δημιουργία Macro Generator στο πρόγραμμα CAD IDS Figaro

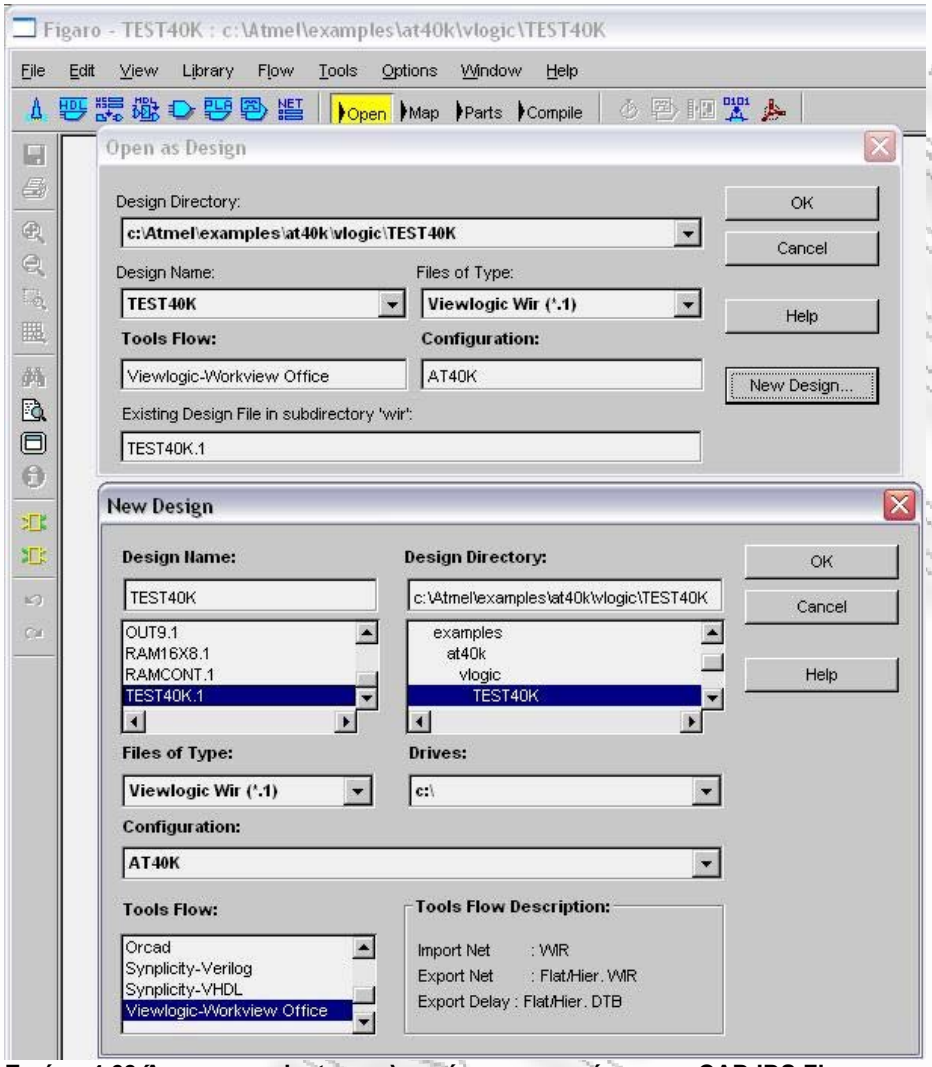
Στην επόμενη ενότητα για την παρουσίαση των δυνατοτήτων της υλοποίησης του εργαλείου IDS Figaro δανειζόμαστε ένα άλλο αρχείο δοκιμής το οποίο εγκαταστάθηκε μαζί με το πρόγραμμα IDS Figaro.

4.4.4. Λειτουργική Προσομοίωση, Σύνθεση και Υλοποίηση

Λόγω προβλημάτων με το εργαλείο IDS Figaro δεν καταφέραμε να δημιουργήσουμε το αρχείο **.edf* η αλλιώς *EDIF netlist*. Έχοντας αυτό το αρχείο θα μπορούσαμε να προχωρήσουμε στη λειτουργική προσομοίωση αλλά και στα περεταίρω βήματα που χρειάζονται για την παραγωγή του αρχείου Bitstream. Επειδή υπήρξε αυτό το πρόβλημα εμείς θα σας δείξουμε πως λειτουργούν αυτά τα βήματα με άλλο πρόγραμμα το οποίο προτείνεται στο εγχειρίδιο χρήσης. Αυτό το πρόγραμμα εγκαταστάθηκε σε συγκεκριμένο φάκελο μαζί με την εγκατάσταση του εργαλείου. Το αρχείο είναι το *TEST40K.1* και το φορτώνουμε με το παρακάτω τρόπο.

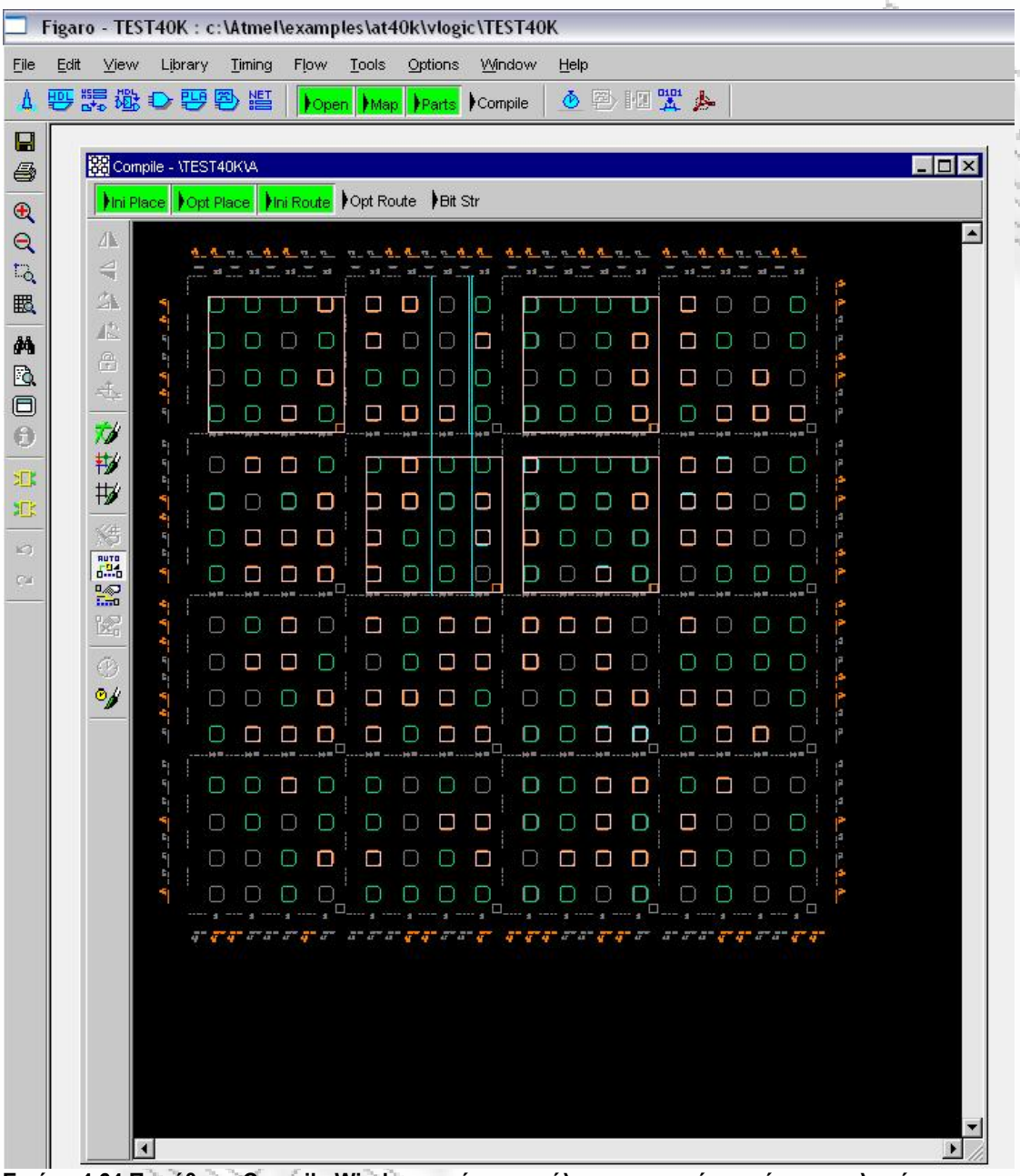
Σημείωση : Η διαδικασία που θα ακολουθηθεί είναι η ίδια για κάθε project στο οποίο έχει δημιουργηθεί και έχει παραχθεί το αρχείο **.edf*. Δηλαδή και αν εμείς καταφέρναμε και παράγαμε το αρχείο **.edf* στην προηγούμενη ενότητα, την ίδια διαδικασία θα ακολουθούσαμε.

Το πρώτο που πρέπει να κάνουμε είναι να φορτώσουμε το πρόγραμμα που θέλουμε να υλοποιήσουμε. Επιλέγουμε *File→Open as Design* στο παράθυρο που θα εμφανιστεί επιλέγουμε *New Design* και στο νέο παράθυρο βρίσκουμε το project που θέλουμε να υλοποιήσουμε, Εικόνα 4.63.



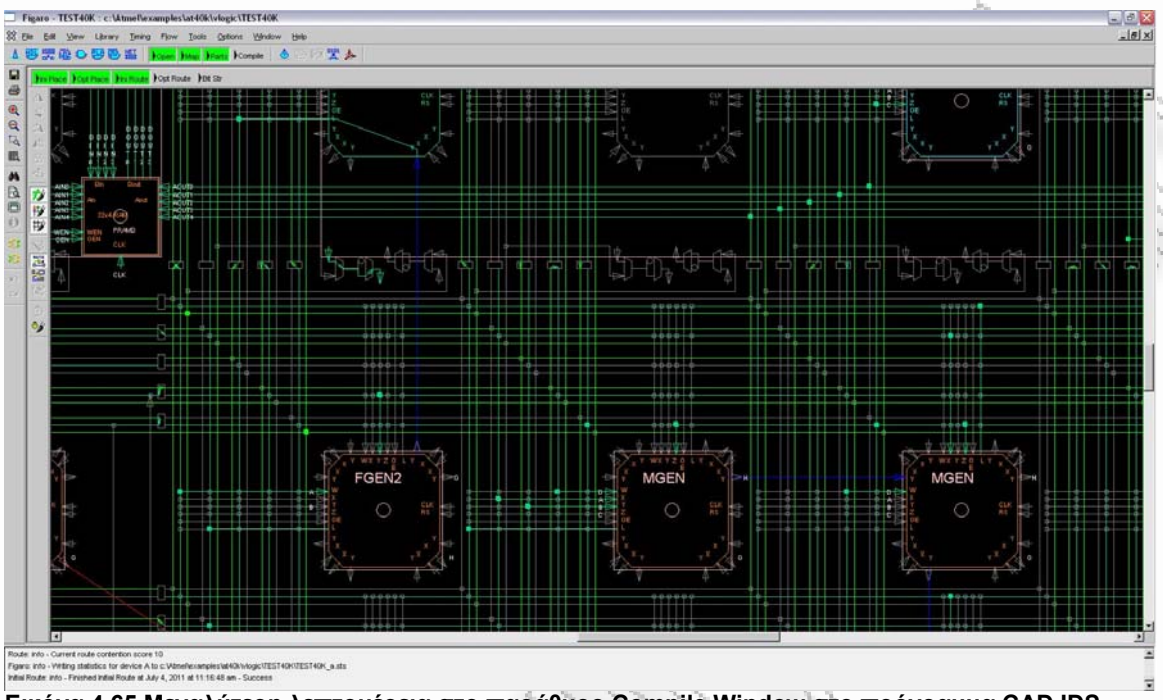
Εικόνα 4.63 Άνοιγμα project για υλοποίηση στο πρόγραμμα CAD IDS Figaro

Όταν φορτωθεί το πρόγραμμα θα εκτελεστεί αυτόματα το *Design Browser*, τα *Map Browser* και *Parts Window* μπορούμε να τα εκτελέσουμε χειροκίνητα, οι ενδεικτικές εικόνες αυτών των λειτουργιών έχουν παρουσιαστεί και είναι οι Εικόνες 4.52, 4.53 και 4.54. Τώρα θα μπορούσε κάνεις να εκτελέσει απευθείας τη λειτουργία Compile, αλλά αν κάνεις επιθυμούσε να δει τα βήματα της εκτελέσεις αυτής, δεν έχει από το να επιλέξει *Window→New Compile Window* και θα εμφανιστεί το *Compile Window*, Εικόνα 4.64. Στο παράθυρο αυτό μπορεί κανείς να δει, πέραν της δομής της συσκευής και πέντε επιλογές που καθορίζουν όλα τα βήματα που θα επιτελούσε αυτόματα το εργαλείο IDS Figaro. Αυτά είναι η τοποθέτηση (Place) η δρομολόγηση (Rout) και η παραγωγή του αρχείου Bitsream, κάθε ένα από δυο πρώτα έχει δυο επιλογές που το καθένα κάνει πρώτη επιλογή και μετά βελτιστοποίηση. Για την τοποθέτηση έχουμε τις επιλογές *ini Place* και *Opt Place* ενώ για τη δρομολόγηση έχουμε το *Ini Route* και το *Opt Route*. Στο τέλος επιλέγοντας και το κουμπί Bitsream θα καταλήξουμε, αν όλα πάνε καλά, στην τελική Εικόνα 4.55.



Εικόνα 4.64 Παράθυρο Compile Window μετά την εκτέλεση των τριών πρώτων επιλογών στο πρόγραμμα CAD IDS Figaro

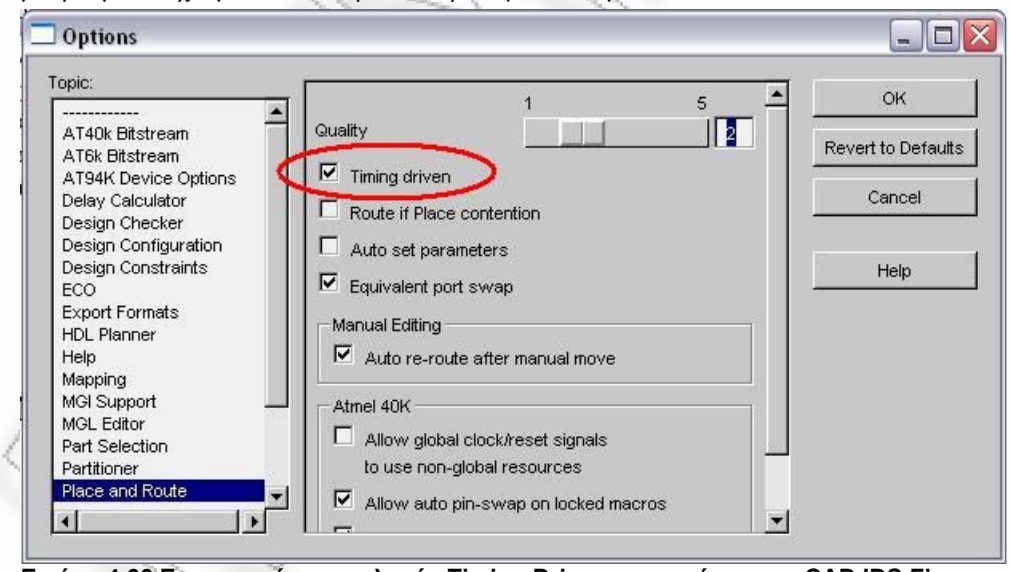
Κάνοντας μεγαλύτερη εστίαση στη δομή της σχεδίασης μπορεί κανείς να δει σε λεπτομέρεια πως γίνεται η επικοινωνία μεταξύ των λογικών στοιχείων μεταξύ τους όπως και ποια από αυτά είναι σε λειτουργία και ποια όχι, Εικόνα 4.65.



Εικόνα 4.65 Μεγαλύτερη λεπτομέρεια στο παράθυρο Compile Window στο πρόγραμμα CAD IDS Figaro

4.4.5. Χρονικοί Περιορισμοί

Για να θέσουμε χρονικούς περιορισμούς πρέπει να έχουμε φορτώσει ένα πρόγραμμα στο IDS Figaro. Εδώ θα χρησιμοποιήσουμε το ίδιο πρόγραμμα με αυτό που είχαμε χρησιμοποιήσει στην προηγούμενη ενότητα το *TEST40K.1*. Κατόπιν αυτού θα πρέπει να επιλέξουμε *Options→Options→Place and Route→Timing Driven* και OK. Επιλέγοντας αυτή τη λειτουργία το Figaro προσπαθεί να βελτιστοποιήσει τις συχνότητες ρολογιού για όλα τα πρωτοβάθμια και δευτεροβάθμια παράγωγά του, Εικόνα 4.66. Το πρωτεύον ρολόι ορίζεται στην εισαγωγή του σχεδίου ενώ το δευτερεύον αναφέρεται στο πρωτεύον, μπορούμε να έχουμε όσα δευτερεύοντα ρολόγια θέλουμε.



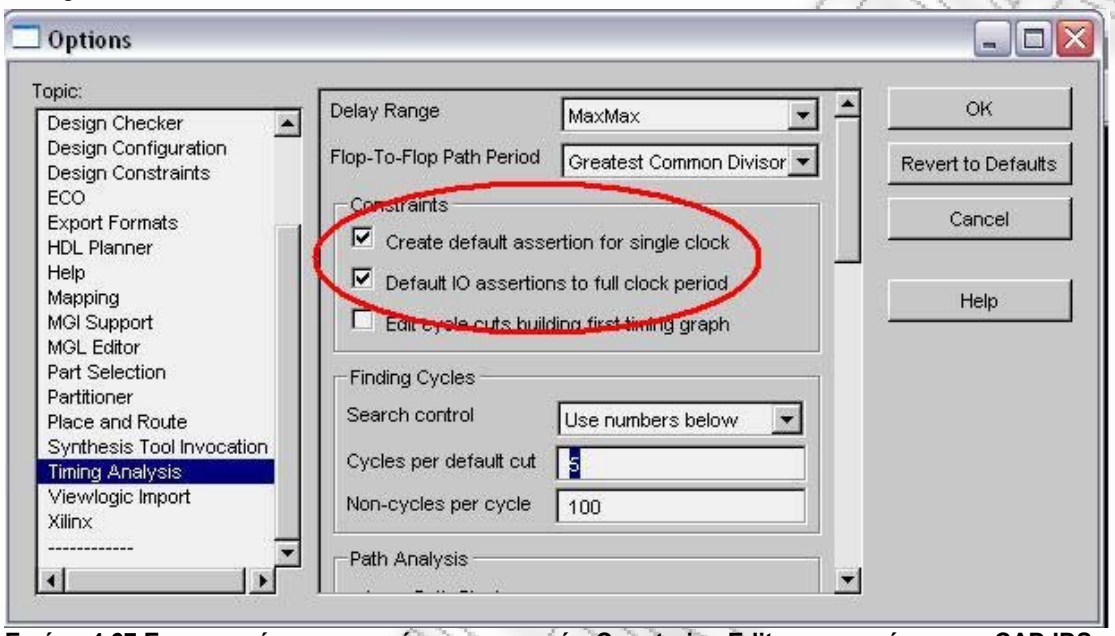
Εικόνα 4.66 Ενεργοποίηση επιλογής Timing Driven στο πρόγραμμα CAD IDS Figaro

Και τα πρωτεύοντα αλλά και τα δευτερεύοντα ρολόγια ευθυγραμμίζονται με το εξωτερικό ρολόι για την αποφύγει το φαινόμενο clock skew που έχουμε προαναφέρει στην Ενότητα 3.1.5.

Αφού ενεργοποιήσουμε το Timing Driven μπορούμε να προχωρήσουμε στην εισαγωγή των χρονικών περιορισμών. Η εισαγωγή των περιορισμών μπορεί να γίνει με δυο τρόπους :

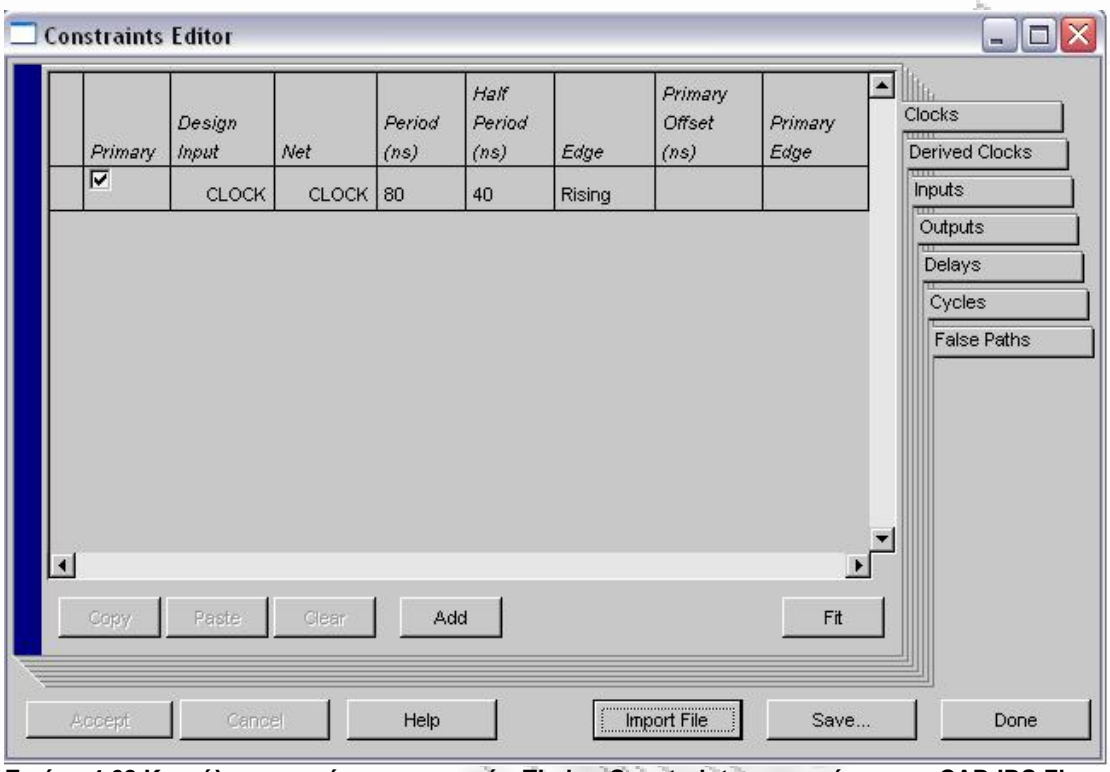
- Διαδραστικά, αλληλεπιδρώντας ο χρήστης με το Figaro και
- Εισάγοντας ένα αρχείο χρονικού περιορισμού **.TMG* το οποίο μπορεί να παραχθεί πολύ νωρίτερα χρησιμοποιώντας το "κειμενογράφο" περιορισμού *Constrains Editor*.

Εμείς σε αυτή την ενότητα θα χρησιμοποιήσουμε τη δεύτερη περίπτωση, πριν γίνει όμως αυτό θα πρέπει πρώτα ενεργοποιήσουμε πρώτα κάποιες επιλογές, οπότε πάμε *Options→Options→Timing Analysis* και ενεργοποιούμε τα *Create default assertion for single clock* και *Default IO assertion to full clock period* και OK, Εικόνα 4.67.



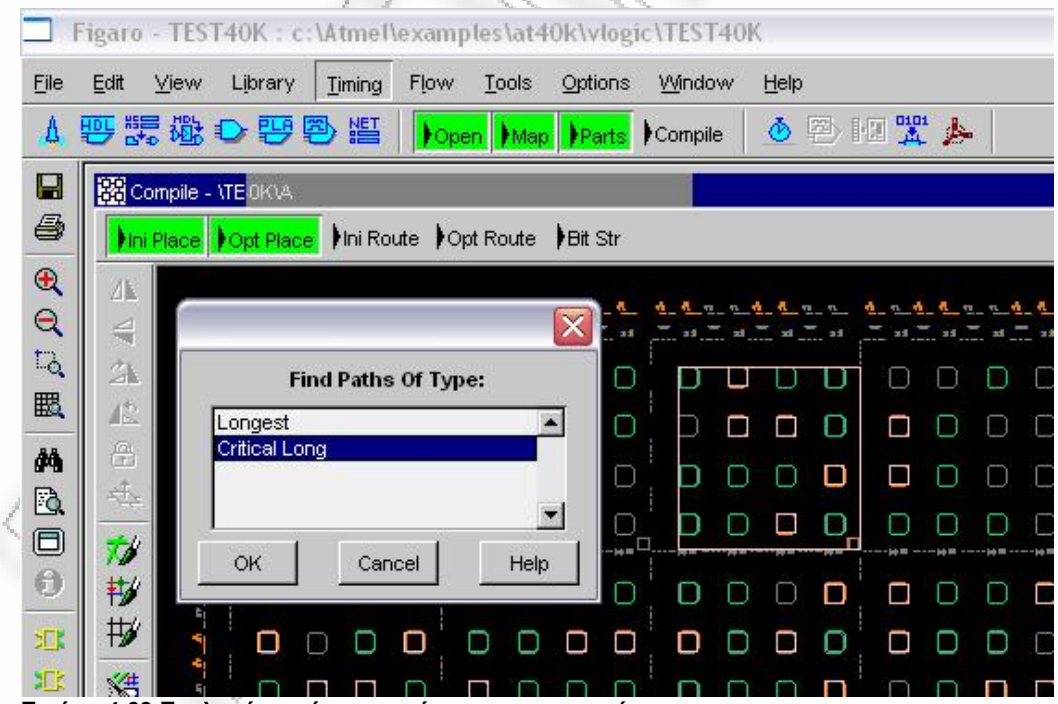
Εικόνα 4.67 Ενεργοποίηση χρονικών περιορισμών Constrains Editor στο πρόγραμμα CAD IDS Figaro

Αφού γίνουν όλα αυτά μπορούμε να ανοίξουμε την καρτέλα για να θέσουμε τους χρονικούς περιορισμούς που θέλουμε. Επιλέγουμε *Edit→Timing Constraints* και εμφανίζεται το παράθυρο της Εικόνας 4.68 όπου κάνεις μπορεί να δει επιπλέον καρτέλες με διάφορες επιλογές που σχετίζονται με τα πρωτεύοντα και δευτερεύοντα ρολόγια. Στην εικόνα βλέπουμε την ύπαρξη ενός μόνο ρολογιού διότι ένα μόνο έχουμε μέσα στη σχεδίαση μας.

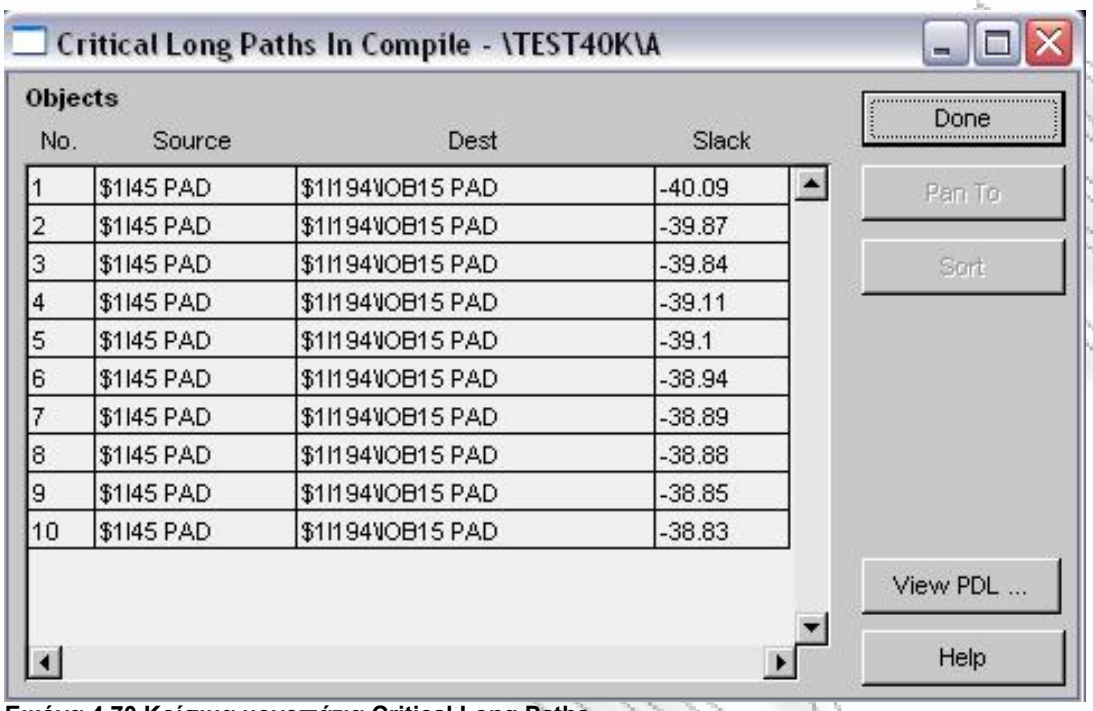


Εικόνα 4.68 Καρτέλα χρονικών περιορισμών Timing Constraints στο πρόγραμμα CAD IDS Figaro

Αν θελήσουμε τώρα να δούμε τα κρίσιμα μονοπάτια μέσα στη σχεδίαση μας πρέπει πρώτα να ανοίξουμε το παράθυρο Compile Window επιλέγοντας *Window→New Compile Window* και εκτελώντας το βήμα της τοποθέτησης (placement). Κατόπιν επιλέγοντας *Timing→Show Analyzed Paths* εμφανίζεται ένα παράθυρο στο οποίο επιλέγουμε Critical Long, Εικόνα 4.69. Επιλέγοντας *Critical Long* και πατώντας OK θα εμφανιστεί το νέο παράθυρο με τα δέκα ποιο κρίσιμα μονοπάτια μέσα στη σχεδίαση μας, Εικόνα 4.70.



Εικόνα 4.69 Επιλογή εμφάνισης κρίσιμων μονοπατιών



No.	Source	Dest	Slack
1	$1I45 PAD	$1I194\OB15 PAD	-40.09
2	$1I45 PAD	$1I194\OB15 PAD	-39.87
3	$1I45 PAD	$1I194\OB15 PAD	-39.84
4	$1I45 PAD	$1I194\OB15 PAD	-39.11
5	$1I45 PAD	$1I194\OB15 PAD	-39.1
6	$1I45 PAD	$1I194\OB15 PAD	-38.94
7	$1I45 PAD	$1I194\OB15 PAD	-38.89
8	$1I45 PAD	$1I194\OB15 PAD	-38.88
9	$1I45 PAD	$1I194\OB15 PAD	-38.85
10	$1I45 PAD	$1I194\OB15 PAD	-38.83

Εικόνα 4.70 Κρίσιμα μονοπάτια Critical Long Paths

Μαζί με τα μονοπάτια φαίνονται και οι τιμές που παίρνει το καθένα από αυτά. Αυτές οι τιμές είναι λογικές καθυστερήσεις που έχει η συγκεκριμένη τοποθέτηση, δηλαδή γίνεται μια εκτίμηση με τις πιθανόν καθυστερήσεις που μπορεί να προέλθουν κατά τη δρομολόγηση (Routing). Στη στήλη *Slack* φαίνεται το χρονικό περιθώριο που υπάρχει κατά τη δρομολόγηση. Μια θετική τιμή σημαίνει ότι το μονοπάτι μπορεί να δρομολογηθεί εντός των τρεχόντων χρονικών περιορισμών. Μια αρνητική τιμή σημαίνει ότι το Figaro δεν θα μπορέσει να εκτελέσει τη συγκεκριμένη σχεδίαση βάση των τρεχόντων χρονικών περιορισμών που υπάρχουν. Εμείς το μόνο που μπορούμε να κάνουμε για βελτιστοποιήσουμε τους χρόνους είναι να τροποποιούμε τις τιμές της περιόδου και το μισό της από την Εικόνα 4.68.

Παρόλο αυτά μπορούμε να προχωρήσουμε στη δρομολόγηση άσχετα αν υπάρχουν αυτές οι καθυστερήσεις μιας και το Figaro εξετάζει τη βέλτιστη επιλογή πάντα. Όταν γίνει η δρομολόγηση μπορούμε χειροκίνητα να τροποποιήσουμε τις όποιες καθυστερήσεις υπάρξουν. Ένα επιπλέον χαρακτηριστικό είναι ότι μπορούμε να δούμε τις όποιες καθυστερήσεις που παράγονται σε ένα έγγραφο επιλέγοντας *Window→New Viewer→Path Analysis Report*. Στο έγγραφο αυτό αναλύονται όλα τα κρίσιμα μονοπάτια λεπτομερέστατα, έχει την κατάληξη *<name>.PDL*.

4.4.6. Στατιστικό αρχείο υλοποίησης

Το στατιστικό αρχείο που παράγεται από την όποια σχεδίαση έχει την ονομασία *<design>.sts* και μπορούμε να το δούμε επιλέγοντας *Window→New Viewer→Design Statistics File*. Στο παράθυρο διαλόγου που θα εμφανιστεί επιλέγουμε το αρχείο μας (αυτό που εργαζόμαστε) και εμφανίζεται το στατιστικό αρχείο, Εικόνα 4.71. Με τον ίδιο τρόπο μπορούμε να δούμε και άλλα log αρχεία τα οποία παράγοντα κατά την υλοποίηση της σχεδίασης όπως είναι το *Log File*, το *Net Delay Table* κτλ.

Εικόνα 4.71 Στατιστικά αποτελέσματα σχεδίασης Design Statistics File στο πρόγραμμα CAD IDS Figaro

Σε αυτή την ενότητα είδαμε πως μπορούμε να σχεδιάσουμε και να δημιουργήσουμε αρχεία VHDL με το εργαλείο σχεδίασης IDS Figaro. Συγκεκριμένα είδαμε το τρόπο σχεδίασης με κώδικα VHDL, πως ελέγχουμε τη σχεδίαση μας αλλά και τη διαδικασία υλοποίησης για την παραγωγή του αρχείου Bitstream. Είδαμε το τρόπο που θέτουμε χρονικούς περιορισμούς σε μία σχεδίαση αλλά και την τελική σύνοψη της σχεδίασης μας σε ένα παράθυρο όπως και σε αρχεία log.

Για περεταίρω μελέτη του εργαλείου σχεδίασης IDS Figaro της εταιρίας Atmel δείτε τα αντίστοιχα έγγραφα *Integrated Development System – Figaro Tutorial* [Atm2] και *Integrated Development System – Figaro User Guide* [Atm3], τα οποία αναφέρονται στη βιβλιογραφία.

5. Πειραματικά Αποτελέσματα

Ολοκληρώσαμε την παρουσίαση των αρχιτεκτονικών δομών στο Κεφάλαιο 3 και την παρουσίαση των εργαλείων σχεδίασης CAD στο Κεφάλαιο 4, σε αυτό το κεφάλαιο θα προβούμε σε κάποια πειράματα έχοντας ως υλικό αξιολόγησης μέτροπρογράμματα (*Benchmarks Program*) τα οποία χρησιμοποιούνται για αυτού του είδους τις δοκιμές και τις συγκρίσεις. Σκοπός μας είναι να δούμε για κάθε μετροπρόγραμμα πως ανταποκρίνεται το αντίστοιχο εργαλείο σχεδίασης αλλά και η αρχιτεκτονική δομή των συσκευών.

Στην Ενότητα 5.1 θα κάνουμε μια εισαγωγή για το πώς θα προχωρήσουμε στα πειράματα και ποια θα είναι τα προγράμματα προς υλοποίηση, θα αναφέρουμε το τρόπο συμπεριφοράς τους αλλά και τι αναμένουμε από αυτά. Στην Ενότητα 5.2 θα προχωρήσουμε σε υλοποίηση των μετροπρογραμμάτων με το εργαλείο σχεδίασης Quartus II της εταιρίας Altera και στην Ενότητα 5.3 την υλοποίηση τους με το εργαλείο ISE της εταιρίας Xilinx. Τέλος στην Ενότητα 5.4 θα συγκρίνουμε τα αποτελέσματα των δοκιμών και θα βγάλουμε κάποια χρήσιμα συμπεράσματα.

Όπως ήδη έχουμε προαναφέρει δοκιμές με το πρόγραμμα IDS Figaro της εταιρίας Atmel δεν είναι δυνατόν.

5.1. Εισαγωγή

5.1.1. Μετροπρογράμματα ITC'99

Για να μπούμε στη διαδικασία να συγκρίνουμε περισσότερα από δυο εργαλεία θα πρέπει το μέτρο σύγκρισης να είναι το ίδιο, δηλαδή με το ίδιο πρόγραμμα να γίνουν δοκιμές σε διαφορετικά εργαλεία και από τα αποτελέσματα που θα παραχθούν θα βγάλουμε τα συμπεράσματα μας. Εμείς για τις ανάγκες των δοκιμών μας έχουμε δανειστεί μερικά μετροπρογράμματα σε μορφή *.vhd από την Πολυτεχνική σχολή του Τορίνο (*Politecnico di Torino*, [PdT]) τα οποία βρίσκονται στην ηλεκτρονική σελίδα ITC'99 [ITC]. Παρέχονται είκοσι ένα προγράμματα για εκτέλεση, κάποια από αυτά έχουν και παραλλαγές και φτάνουν σε αριθμό τα είκοσι εννέα. Εμείς και στα δυο εργαλεία θα προβούμε σε υλοποίηση και των είκοσι εννέα προγραμμάτων, δηλαδή θα γίνουν πενήντα οκτώ υλοποιήσεις και από τα δυο εργαλεία. Τα είκοσι εννέα αυτά μετροπρογράμματα φαίνονται, το καθένα από αυτά, αναλυτικά στο παράρτημα Γ. Τα όνομα των μετροπρογραμματων αλλά και η λειτουργία που επιτελούν φαίνονται στο Πίνακα 5.1. Υλοποιούν διάφορα κυκλώματα από τα οποία κάποια από αυτά συσχετίζονται μεταξύ τους, δηλαδή κάποια αποτελούν δομικά στοιχεία κάποιων άλλων.

Όνομα	Λειτουργία
b01	FSM συγκρίνει σειριακή ροή
b02	FSM αναγνωρίζει αριθμούς - στοιχειά BCD
b03	Πόρος διαιτητεύσεις (Resource arbiter)
b04	Υπολογίζει το μέγιστο και το ελάχιστο (min,max)
b05	Επεξεργάζεται τα περιεχόμενα της μνήμης
b06	Διακοπή ρουτίνας εξυπηρέτησης
b07	Μέτρηση σημείων σε μια ευθεία γραμμή
b08	Εύρεση στοιχείων σε ακολουθία αριθμών
b09	Σειριακός μετατροπέας
b10	Εκλογικό σύστημα
b11	Ανακάτεμα αλφαριθμητικού με κρυπτογραφημένη μεταβλητή
b12	Ενός παίκτη παιχνίδι (μάντεψε την ακολουθία)
b13	Διεπαφή για μετεωρολογικούς αισθητήρες
b14	Viper επεξεργαστής (υποσύνολο)
b15	80386 επεξεργαστής (υποσύνολο)
b17	Τρία αντίγραφα από το b15
b18	Δύο αντίγραφα από το b14 και δύο από το b17
b19	Δύο αντίγραφα από το b14 και δύο από το b17
b20	Αντίγραφο από το b14 και μια τροποποιημένη έκδοση του b14

b21	Δύο αντίγραφα από το b14
b22	Αντίγραφο από το b14 και δυο τροποποιημένες εκδόσεις του b14

Πίνακας 5.1 [ITC] Τα μετροπρογράμματα ITC'99 και η λειτουργία που επιτελούν

5.1.2. Επιδώσεις Συστήματος

Για να έχουμε σωστά αποτελέσματα και τεκμηριωμένα θα προβούμε σε παρατήρηση των χαρακτηριστικών που παρουσιάζονται μετά την υλοποίηση του κάθε μετροπρογράμματος. Τα πεδία που θα παρατηρήσουμε φαίνονται στο Πινάκα 5.2.

Όνομα	Περιγραφή
Name VHDL	Το όνομα κάθε μετροπρογράμματος θα ξεκινήσει από την κωδική λέξη **b** και στη συνέχεια η αρίθμηση του 01, 02…22. Για τις παραλλαγές κάποιων κυκλωμάτων που έχουμε προαναφέρει, θα χρησιμοποιείτε ενδιάμεσα η κάτω παύλα _ , δηλαδή θα είναι της μορφής **b17_1**.
Device	Η συσκευή FPGA που χρησιμοποιείτε για τη σχεδίαση
Line	Οι γραμμές κώδικα που υπάρχουν στο αρχείο VHDL.*
Process	Αριθμός των διαδικασιών (process) που υπάρχουν μέσα στο μετροπρογραμμα.*
Fmax	Maximum Frequency, Υποδηλώνει τη μέγιστη προτεινόμενη συχνότητα στην οποία μπορεί να δουλέψει η συσκευή FPGA. Η μονάδα μέτρησης είναι τα MHz (Mega Hertz), όπου 1 MHz = 10^6 Hz η ακρίβεια θα είναι δύο δεκαδικών ψηφίων.
PLL - DCM	Διευθυντές ρολογιού που χρησιμοποιούνται κατά την υλοποίηση
LE	Logic Elements, Λογικά στοιχεία που χρησιμοποιούνται κατά την υλοποίηση.
LUT 4-input	Look up table, Πίνακες αναζήτησης τεσσάρων εισόδων που χρησιμοποιούνται κατά την υλοποίηση
Pin In	Ακροδέκτες εισόδου που χρησιμοποιούνται κατά την υλοποίηση
Pin Out	Ακροδέκτες εξόδου που χρησιμοποιούνται κατά την υλοποίηση
Global Clocks	Γενικά ρολόγια που χρησιμοποιούνται κατά την υλοποίηση
BRAM	Block RAM, βαθμίδες μνήμης RAM που χρησιμοποιούνται κατά την υλοποίηση
Flip Flop	Number of Flip Flop, Αριθμός από flip flop (καταχωρητές) που χρησιμοποιούνται κατά την υλοποίηση

* Οι αριθμοί αυτοί παίρνονται από το κατασκευαστή των μετροπρογραμμάτων.

Πινάκας 5.2 Πεδία παρατήρησης των συστημάτων κατά την υλοποίηση

Από την παρατήρηση των πεδίων θα βγάλουμε κάποια χρήσιμα συμπεράσματα σε σχέση με την ικανότητα του εργαλείου σχεδίασης CAD να διευθετεί με αποδοτικό τρόπο τους χώρους και τους πόρους της συσκευής FPGA. Επίσης θα δούμε αν η αρχιτεκτονική δομής της συσκευής αυτής, δίνει στο εργαλείο σχεδίασης CAD την ευχέρεια να τη διαχειριστεί σωστά.

Τα χαρακτηριστικά των χρόνων Setup Time και Hold Time δε θα παρατηρηθούν μιας και αποτελούν ιδιαίτερο (υλικό) χαρακτηριστικό κάθε συσκευής και ξεφεύγουν από το πλαίσιο της διατριβής. Για αυτό το λόγο θα θέσουμε σε όλα τα πειράματα που θα διεξαχθούν, τους χρόνους που προτείνουν τα εργαλεία CAD κατά τη σχεδίαση.

5.1.3. Ακολουθούμενη Διαδικασία Πειράματος

Η διαδικασία που θα ακολουθηθεί στα πειράματα και για τα δυο εργαλεία θα είναι η εξής:

1. Θα φορτώσουμε το πρότυπο πρόγραμμα VHDL σε ένα νέο project.
2. Θα κάνουμε σύνθεση και υλοποίηση.
3. Θα κάνουμε καταμέτρηση των χαρακτηριστικών του Πίνακα 5.2 και θα ελέγξουμε ποια είναι η προτεινόμενη συχνότητα λειτουργίας ώστε να την καταγράψουμε.

Οι παραπάνω ενέργειες θα γίνουν κατά τη διάρκεια της υλοποίηση και εμείς θα καταγράψουμε τα αποτελέσματα, για κάθε ένα μετροπρόγραμμα, στο Πίνακα 5.3.

Μια ακόμη πολύ σημαντική λεπτομέρεια είναι ότι από το βήμα 2 θα προκύψει και ποια συσκευή μπορεί να χρησιμοποιηθεί για την υλοποίηση της όποιας σχεδίασης. Εμείς στο Πίνακα 5.3 στο πεδίο "**Device**" θα γράφουμε ποια συσκευή υλοποιεί τη συγκεκριμένη σχεδίαση.

Name VHDL	Device	Line	Process	Fmax	PLL - DCM	LE	LUT 4-input	Pin In	Pin Out	Global Clocks	BRAM	Flip Flop

Πίνακας 5.3 Προτεινόμενος πίνακας για τα αποτελέσματα των δοκιμών μετροπρογράμματος

5.2. Υλοποίηση μετροπρογραμματων VHDL στο εργαλείο σχεδίασης CAD Altera Quartus II 11.0

Από τις υλοποιήσεις που εκτελέσαμε στα μετροπρογράμματα με το εργαλείο σχεδίασης Quartus II, προέκυψαν οι συσκευές που φαίνονται στο Πίνακα 5.4. Τα συνολικά αποτελέσματα των υλοποιήσεων για κάθε ένα μετροπρόγραμμα φαίνονται στο Πίνακα 5.5.

Device	PLL	LE	Pin In - Out	Global Clocks	Total RAM (bits)	Flip Flop
EP1C3T100A8	**1**	**2910**	**65**	**8**	**59904**	**3099**
EP1C3T144A8	**1**	**2910**	**104**	**8**	**59904**	**3210**
EP1C6T144C6	**2**	**5980**	**98**	**8**	**92160**	**6262**
EP1C6Q240C6	**2**	**5980**	**185**	**8**	**92160**	**6523**
EP1C12Q240C6	**2**	**12060**	**173**	**8**	**239616**	**12567**
EP1C20F324C6	**2**	**20060**	**233**	**8**	**294912**	**20747**

Πίνακας 5.4 Συσκευές FPGA που χρησιμοποιήθηκαν κατά την υλοποίηση στο εργαλείο σχεδίασης CAD Quartus II

Παρακάτω κάποιες πρώτες παρατηρήσεις που βγήκαν με μια πρώτη ματιά από τα αποτελέσματα των υλοποιήσεων.

Σημείωση 1: Στα πεδία PLL – DCM και BRAM δεν καταγράφηκαν χαρακτηριστικά για κανένα από τα μετροπρογράμματα.

Σημείωση 2: Σε όλα τα μετροπρογράμματα χρησιμοποιήθηκαν 2 γενικά ρολόγια.

Σημείωση 3: Για τα μετροπρόγραμματα b19 και b19_1 δεν καταφέραμε να κάνουμε υλοποίηση σε κάποια συσκευή FPGA διότι η σχεδίαση απαιτούσε σχεδόν 35000 λογικά στοιχειά, ενώ η αρχιτεκτονική των συσκευών Cyclone φτάνει μέχρι 20060 λογικά στοιχεία.

Name VHDL	Device	Line	Process	Fmax	PLL - DCM	LE (percent)	LUT 4-input	Pin In	Pin Out	Global Clocks (percent)	BRAM	Flip Flop - Registers (percent)
b01	EP1C3T100A8	110	1	407,00 MHz	-	12 (1%)	5	4	2	2 (25%)	-	5 (1%)
b02	EP1C3T100A8	70	1	505,82 MHz	-	6 (1%)	3	3	1	2 (25%)	-	4 (1%)
b03	EP1C3T100A8	141	1	197,12 MHz	-	36 (1%)	5	6	4	2 (25%)	-	30 (1%)
b04	EP1C3T100A8	102	1	69,75 MHz	-	192 (7%)	29	13	8	2 (25%)	-	66 (2%)
b05	EP1C3T100A8	332	3	87,15 MHz	-	187 (6%)	74	3	36	2 (25%)	-	34 (1%)
b06	EP1C3T100A8	128	1	442,87 MHz	-	13 (1%)	8	4	6	2 (25%)	-	9 (1%)
b07	EP1C3T100A8	92	1	115,51 MHz	-	106 (4%)	20	3	8	2 (25%)	-	43 (1%)
b08	EP1C3T100A8	89	1	191,5 MHz	-	48 (1%)	21	11	4	2 (25%)	-	21 (1%)
b09	EP1C3T100A8	103	1	152,95 MHz	-	45 (2%)	30	3	1	2 (25%)	-	28 (1%)
b10	EP1C3T100A8	167	1	114,29 MHz	-	75(3%)	52	13	6	2 (25%)	-	17 (1%)
b11	EP1C3T100A8	118	1	85,06 MHz	-	211 (7%)	78	9	6	2 (25%)	-	31 (1%)
b12	EP1C3T100A8	569	4	111,38 MHz	-	460 (16%)	244	7	6	2 (25%)	-	119(4%)
b13	EP1C3T100A8	296	5	136,17 MHz	-	99 (3%)	44	12	10	2 (25%)	-	53 (2%)
b14	EP1C3T144A8	509	1	38,9 MHz	-	1100 (35%)	452	34	54	2 (25%)	-	216 (7%)
b14_1	EP1C3T144A8	509	1	43,03 MHz	-	1072 (34%)	423	34	54	2 (25%)	-	216 (7%)

b15	EP1C6Q240C6	671	3	75,92 MHz	-	2343 (39%)	1200	38	70	2 (25%)	-	419 (6%)
b15_1	EP1C6Q240C6	671	3	68,67 MHz	-	2343 (39%)	1200	38	70	2 (25%)	-	419 (6%)
b17	EP1C12Q240C6	810	15	71,96 MHz	-	7229 (60%)	3582	39	97	2 (25%)	-	1325 (11%)
b17_1	EP1C12Q240C6	810	15	71,13 MHz	-	7229 (60%)	3582	39	97	2 (25%)	-	1325 (11%)
b18	EP1C20F324C6	1424	33	59,81 MHz	-	18169 (89%)	8749	38	23	2 (25%)	-	3037 (15%)
b18_1	EP1C20F324C6	1424	33	56,68 MHz	-	18131 (89%)	8679	38	23	2 (25%)	-	3037 (15%)
b19	-	1491	10	-	-	-	-	-	-	-	-	-
b19_1	-	1491	10	-	-	-	-	-	-	-	-	-
b20	EP1C3T100A8	1085	3	45,08 MHz	-	2262 (72%)	885	34	22	2 (25%)	-	431 (14%)
b20_1	EP1C3T100A8	1085	3	44, 84 MHz	-	2233 (71%)	855	34	22	2 (25%)	-	431 (14%)
b21	EP1C3T100A8	1089	3	47,56 MHz	-	2265 (72%)	761	34	22	2 (25%)	-	431 (14%)
b21_1	EP1C3T100A8	1089	3	46,94 MHz	-	22237 (71%)	736	34	22	2 (25%)	-	431 (14%)
b22	EP1C6T144C6	1613	4	56,03 MHz	-	3353 (52%)	1253	34	22	2 (25%)	-	614 (10%)
b22_1	EP1C6T144C6	1613	4	54,71 MHz	-	3325 (52%)	1228	34	22	2 (25%)	-	614 (10%)

Πίνακας 5.5 Αποτελέσματα δοκιμών μετροπρογράμματων στο εργαλείο CAD Altera Quartus II

5.3. Υλοποίηση μετροπρογραμματων VHDL στο εργαλείο σχεδίασης CAD Xilinx ISE 10.1

Από τις υλοποιήσεις που εκτελέσαμε στα μετροπρογράμματα με το εργαλείο σχεδίασης ISE, προέκυψαν οι συσκευές που φαίνονται στο Πίνακα 5.6. Τα συνολικά αποτελέσματα των προσομοιώσεων φαίνονται στο Πίνακα 5.7 για κάθε ένα μετροπρόγραμμα.

Device	DCM	LE	Pin In - Out	Global Clocks	Total RAM (bits)	Flip Flop
XC3S50-4VQ100	2	1536	63	8	73728	1536
XC3S200-4TQ144	4	3840	97	8	221184	3840
XC3S200-4PQ208	4	3840	141	8	221184	3840
XC3S400-4TQ144	4	7168	97	8	294912	7168
XC3S1000-4FG320	4	15360	221	8	442368	15360
XC3S1500-4FG320	4	26624	221	8	589824	26624

Πίνακας 5.6 Συσκευές FPGA που χρησιμοποιήθηκαν κατά την υλοποίηση στο εργαλείο σχεδίασης CAD ISE

Παρακάτω κάποιες πρώτες παρατηρήσεις που βγήκαν με μια πρώτη ματιά από τα αποτελέσματα των υλοποιήσεων.

Σημείωση 1: Στα πεδία PLL – DCM και BRAM δεν καταγράφηκαν χαρακτηριστικά για κανένα από τα μετροπρογράμματα.

Σημείωση 2: Σε όλα τα μετροπρογράμματα χρησιμοποιήθηκε 1 γενικό ρολόι για τη σχεδίαση.

Σημείωση 3: Για τα μετροπρόγραμματα b19 και b19_1 δεν καταφέραμε να κάνουμε υλοποίηση σε κάποια συσκευή FPGA διότι η σχεδίαση απαιτούσε σχεδόν 41000 λογικά στοιχειά, κάτι που η αρχιτεκτονική των συσκευών Spartan–3 φτάνει μέχρι 29952 λογικά στοιχεία. Από το Πίνακα 3.3 φαίνεται ότι υπάρχουν συσκευές με περισσότερα από 40000 λογικά στοιχεία, αλλά αυτές οι συσκευές δεν είναι συμβατές με το πρόγραμμα σχεδίασης ISE. Οπότε η υλοποίηση σε μια συσκευή FPGA είναι αδύνατη.

Name VHDL	Device	Line	Process	Fmax	PLL - DCM	LE (percent)	LUT 4-input	Pin In	Pin Out	Global Clocks (percent)	BRAM	Flip Flop - Registers (percent)
b01	XC3S50-4VQ100	110	1	399,84 MHz	-	9 (1%)	9	4	2	1 (12%)	-	5 (1%)
b02	XC3S50-4VQ100	70	1	488,38 MHz	-	4 (1%)	4	3	1	1 (12%)	-	4 (1%)
b03	XC3S50-4VQ100	141	1	182,58 MHz	-	76 (4%)	76	6	4	1 (12%)	-	35 (2%)
b04	XC3S50-4VQ100	102	1	145,62 MHz	-	143 (9%)	143	13	8	1 (12%)	-	66 (4%)
b05	XC3S50-4VQ100	332	3	82,25 MHz	-	235 (15%)	235	3	36	1 (12%)	-	46 (2%)
b06	XC3S50-4VQ100	128	1	327,55 MHz	-	9 (1%)	9	4	6	1 (12%)	-	8 (1%)
b07	XC3S50-4VQ100	92	1	151,12 MHz	-	149 (9%)	149	3	8	1 (12%)	-	47 (3%)
b08	XC3S50-4VQ100	89	1	171,55 MHz	-	37 (2%)	37	11	4	1 (12%)	-	21 (1%)
b09	XC3S50-4VQ100	103	1	146,47 MHz	-	47 (3%)	47	3	1	1 (12%)	-	28 (1%)
b10	XC3S50-4VQ100	167	1	209,51 MHz	-	54 (3%)	54	13	6	1 (12%)	-	24 (1%)
b11	XC3S50-4VQ100	118	1	116,73 MHz	-	145 (9%)	142	9	6	1 (12%)	-	37 (2%)
b12	XC3S50-4VQ100	569	4	131,87 MHz	-	385 (25%)	384	7	6	1 (12%)	-	141(9%)
b13	XC3S50-4VQ100	296	5	163,58 MHz	-	60 (3%)	59	12	10	1 (12%)	-	49 (3%)
b14	XC3S200-4TQ144	509	1	39,789 MHz	-	2470 (64%)	2440	34	54	1 (12%)	-	218 (5%)
b14_1	XC3S200-4TQ144	509	1	43,94 MHz	-	2564 (66%)	2535	34	54	1 (12%)	-	217 (5%)

b15	XC3S200-4PQ208	671	3	63,50 MHz	-	2796(72%)	2591	38	70	1 (12%)	-	442 (11%)
b15_1	XC3S200-4PQ208	671	3	65,61 MHz	-	2834 (73%)	2598	38	70	1 (12%)	-	436 (11%)
b17	XC3S1000-4FG320	810	15	62,85 MHz	-	8352 (54%)	7738	39	97	1 (12%)	-	13935 (9%)
b17_1	XC3S1000-4FG320	810	15	58,88 MHz	-	8642 (56%)	7935	39	97	1 (12%)	-	1376 (8%)
b18	XC3S1500-4FG320	1424	33	44,80 MHz	-	21513 (80%)	20202	38	23	1 (12%)	-	3105 (11%)
b18_1	XC3S1000-4FG320	1424	33	44,64 MHz	-	21960 (82%)	20394	38	23	1 (12%)	-	3105 (11%)
b19	-	1491	10	-	-	-	-	-	-	-	-	-
b19_1	-	1491	10	-	-	-	-	-	-	-	-	-
b20	XC3S400-4TQ144	1085	3	44,47 MHz	-	4740 (66%)	4696	34	22	1 (12%)	-	435 (6%)
b20_1	XC3S400-4TQ144	1085	3	41,86 MHz	-	4872 (67%)	4840	34	22	1 (12%)	-	435 (6%)
b21	XC3S400-4TQ144	1089	3	40,77 MHz	-	5196 (72%)	5118	34	22	1 (12%)	-	435 (6%)
b21_1	XC3S400-4TQ144	1089	3	43,91 MHz	-	5355 (74%)	5299	34	22	1 (12%)	-	436 (6%)
b22	XC3S400-4TQ144	1613	4	41,79 MHz	-	6574 (91%)	6480	34	22	1 (12%)	-	620 (8%)
b22_1	XC3S400-4TQ144	1613	4	41,60 MHz	-	6673 (93%)	6613	34	22	1 (12%)	-	621 (8%)

Πίνακας 5.7. Αποτελέσματα δοκιμών μετροπρογράμματων στο εργαλείο CAD Xilinx ISE

5.4. Σύγκριση αποτελεσμάτων και εκτίμηση τους

Οι πρώτες παρατηρήσεις έγιναν στην προηγούμενη ενότητα, εδώ θα δούμε λίγο πιο σχολαστικά κάποια ιδιαίτερα χαρακτηριστικά όπως είναι η συχνότητα λειτουργίας, τα λογικά στοιχεία, τους πίνακες αναζήτησης τεσσάρων εισόδων (4 input LUT) και τέλος τους καταχωρητές (flip flop) που καταλαμβάνει μια σχεδίαση.

Να σημειωθεί ότι στα πεδία Pin in και Pin Out υπήρχε πλήρης ταύτιση στην εκτίμηση και από τα δύο εργαλεία, η μόνη διαφορά ήταν στο ποσοστό (τοις εκατό) μιας και το καθένα από αυτά έχει διαφορετικό αριθμό ακροδεκτών.

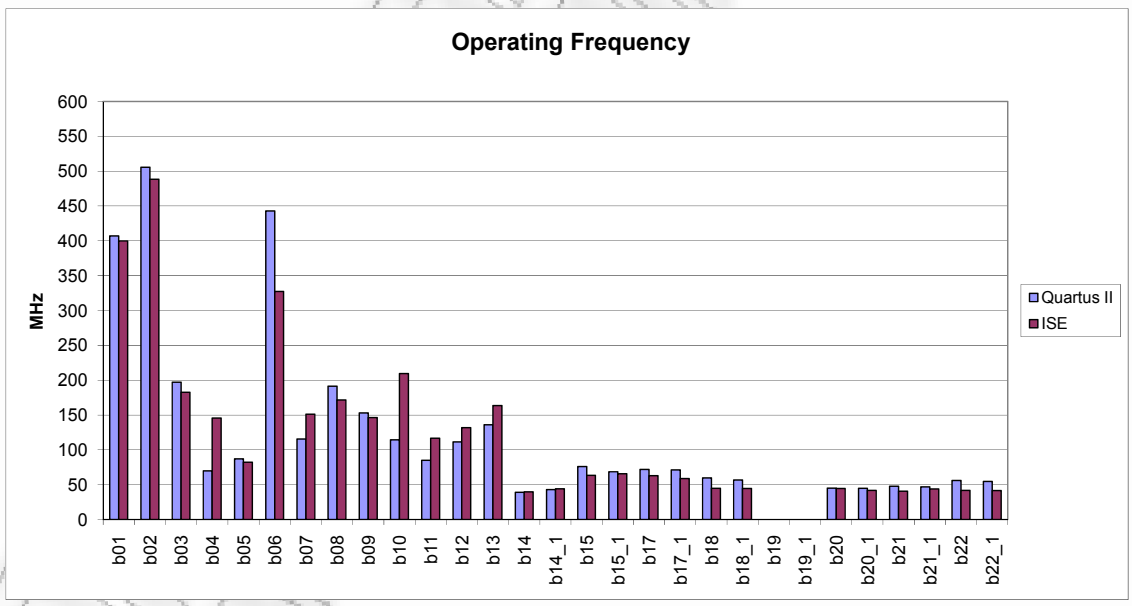
Επίσης σε όλες τις σχεδιάσεις των μετροπρογραμματων το πρόγραμμα σχεδίασης Quartus II έχει χρησιμοποιήσει δύο από τα οκτώ γενικά ρολόγια (Global Clocks) ενώ το πρόγραμμα ISE ένα από τα οκτώ.

Και ένα τελευταίο και τα δύο εργαλεία σχεδίασης όπως και οι αντίστοιχες αρχιτεκτονικές δεν κατάφεραν να υλοποιήσουν τη σχεδίαση των μετροπρογράμματων b19 και b19_1, τα μηδενικά αποτελέσματα φαίνονται και στα παρακάτω διαγράμματα που θα παρακολουθήσετε.

5.4.1. Συχνότητες λειτουργίας

Στην Εικόνα 5.1 φαίνεται το διάγραμμα με τις συχνότητες λειτουργίας των μετροπρογραμματων που έχουν επιτύχει τα δύο CAD εργαλεία. Όπως μπορείτε να παρατηρήσετε δεν διαφέρουν και πολύ αν εξαιρέσουμε τις περιπτώσεις των μετροπρογραμματων b04 και b10, όπου το ISE φαίνεται να έχει προτείνει σχεδόν διπλάσια συχνότητα από ότι αυτή του Quartus II. Ενώ στο μετροπρόγραμμα b06 το Quartus προτείνει μεγαλύτερη συχνότητα.

Μία ακόμη παρατήρηση μπορούμε να κάνουμε σε σχέση με τις συχνότητες της σχεδίασης είναι, ότι από το μετροπρόγραμμα b14_1 και μετά, το Quartus II έχει σταθερά προτείνει μεγαλύτερη συχνότητα λειτουργίας από ότι το ISE. Εδώ θα θέλαμε να σας ενημερώσουμε ότι η πολυπλοκότητα των μετροπρογραμμάτων αυτών αυξάνει κατά πολύ, στα παρακάτω διαγράμματα θα δείτε αυτή τη διαφορά.



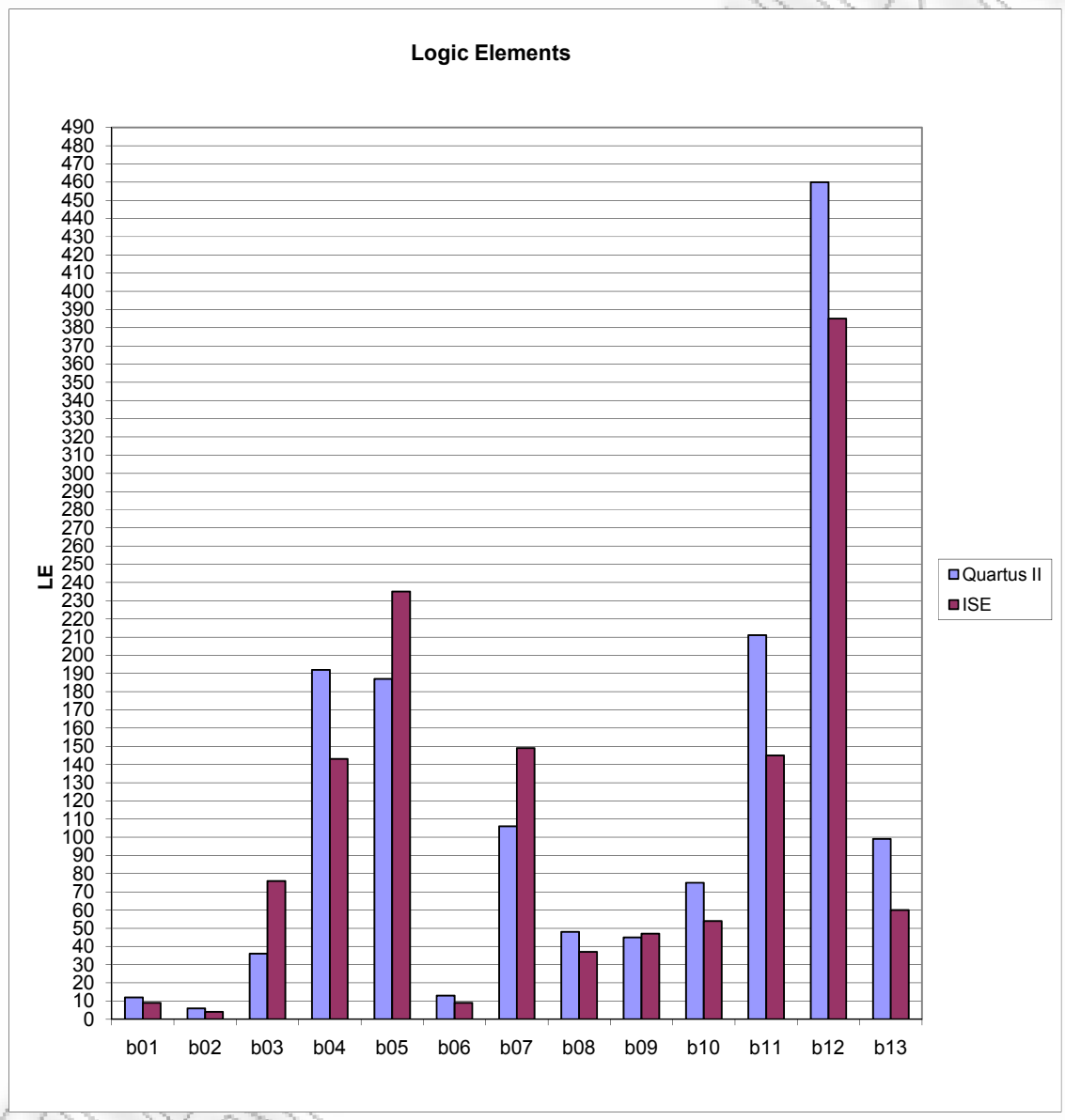
Εικόνα 5.1 Διάγραμμα συχνοτήτων λειτουργίας των μετροπρογραμμάτων, από τα εργαλεία σχεδίασης CAD Quartus II και ISE

5.4.2. Λογικά στοιχεία σχεδίασης

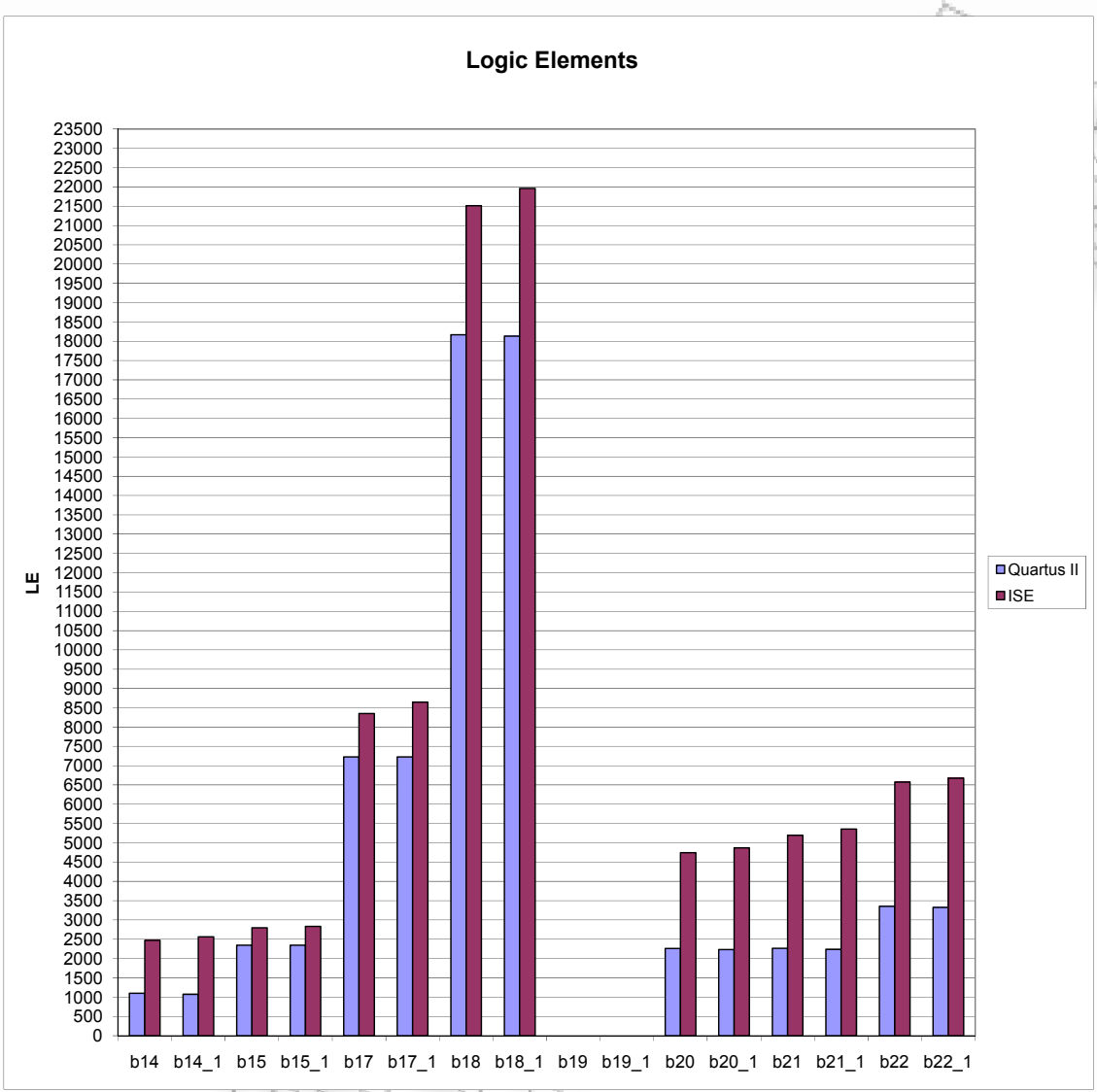
Στην Εικόνα 5.2 και 5.3 βλέπουμε τα λογικά στοιχεία που καταλαμβάνει το σχέδιο κάθε μετροπρογράμματος. Έχουμε χωρίσει σε δύο διαγράμματα τα αποτελέσματα για καλύτερη παρουσίαση αλλά και εξαγωγή συμπερασμάτων. Στην Εικόνα 5.2 βλέπουμε τα μετροπρογράμματα από το b01μέχρι το b13, αυτά τα μετροπρογράμματα έχουν το πολύ 500 λογικά στοιχεία. Στην Εικόνα 5.3 βλέπουμε τα

υπόλοιπα από το b14 μέχρι το b22_1, αυτά έχουν περισσότερα από 500 λογικά στοιχεία. Οι δύο εικόνες μας δείχνουν και τη διαφορά στην πολυπλοκότητα των σχεδίων των μετροπρογραμμάτων, στη μεν πρώτη έχουμε τα απλά σχέδια και στη δε δεύτερη τα πολύπλοκα.

Όπως μπορείτε να διαπιστώσετε στην πρώτη ομάδα η κατανομή των λογικών στοιχείων μπορεί να χαρακτηριστεί φυσιολογική. Στη δεύτερη, η κατανομή των λογικών στοιχείων από μέρους του εργαλείου ISE είναι σε όλες τις περιπτώσεις, μεγαλύτερη! Εδώ θα θέλαμε να σας υπενθυμίσουμε ότι η αρχιτεκτονική Spartan-3 δεν έχει αυτούσια λογικά στοιχεία, αλλά τα τεσσάρων εισόδων πίνακες αναζήτησης (4-input LUT) τα οποία και χαρακτηρίζει ως «λογικά στοιχεία». Θεωρούμε λοιπόν φυσιολογική αυτή την αύξηση λόγω της ιδιόρρυθμης αρχιτεκτονικής της οικογένειας Spartan-3.



Εικόνα 5.2 Διάγραμμα λογικών στοιχείων που καταλαμβάνει η σχεδίαση των μετροπρογραμμάτων b01 .. b13, από τα εργαλεία σχεδίασης CAD Quartus II και ISE



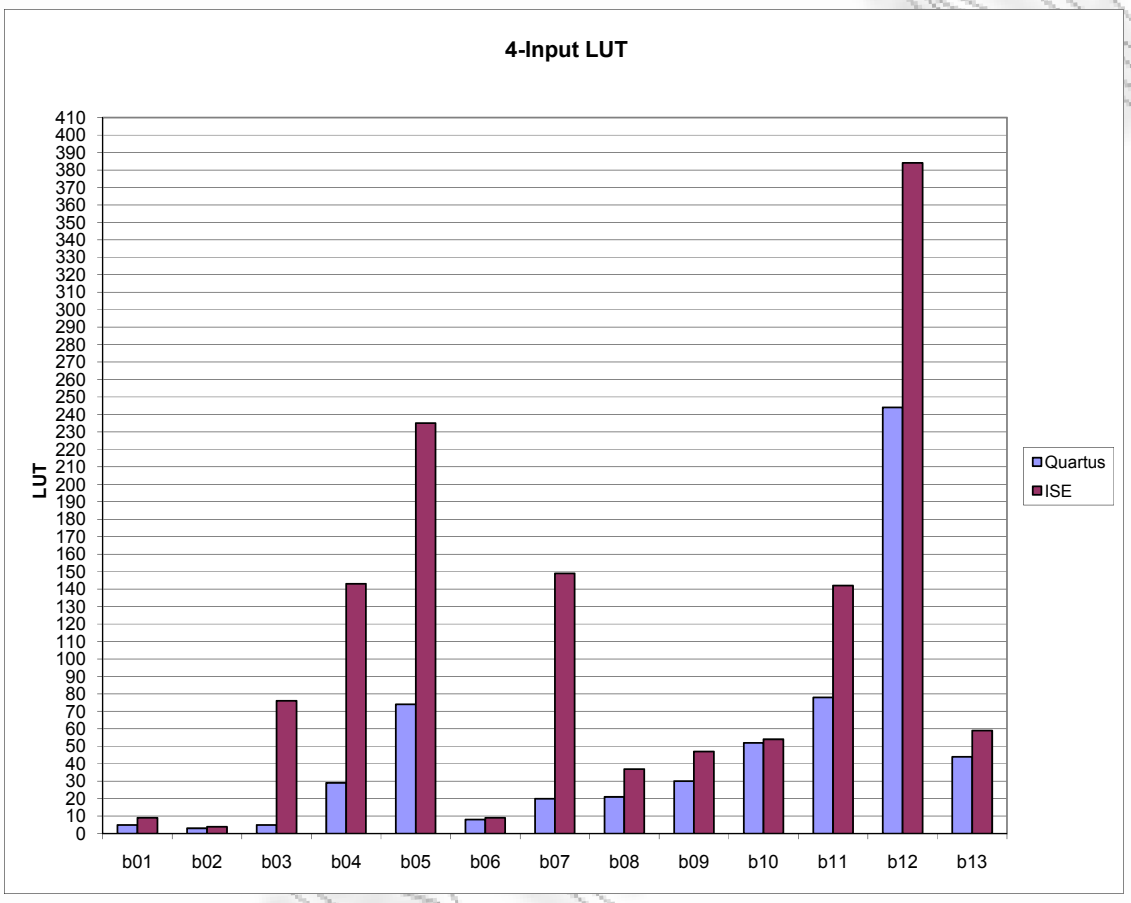
Εικόνα 5.3 Διάγραμμα λογικών στοιχείων που καταλαμβάνει η σχεδίαση των μετροπρογραμμάτων b14 .. b22_1, από τα εργαλεία σχεδίασης CAD Quartus II και ISE

Θα θέλαμε επίσης να παρατηρήσετε τις Εικόνες 5.1 και 5.3 για να δείτε την αύξηση της συχνότητας λειτουργία από μέρους του εργαλείου Quartus II. Όσο περισσότερα λογικά στοιχεία υπάρχουν σε μια σχεδίαση τόσες περισσότερες διασυνδέσεις υπάρχουν. Άρα οι καθυστερήσεις μεταξύ καταχωρητών και διασυνδέσεων είναι αναπόφευκτη. Αυτή η θεώρηση είναι για σχεδιάσεις που περιέχουν περισσότερα από 500 λογικά στοιχεία και αυτό γιατί στην περίπτωση του μετροπρογράμματος b07 Εικόνα 5.1 και 5.2, βλέπουμε ότι ενώ η σχεδίαση έχει περισσότερα λογικά στοιχεία η συχνότητα λειτουργίας του ISE είναι και πάλι μεγαλύτερη!

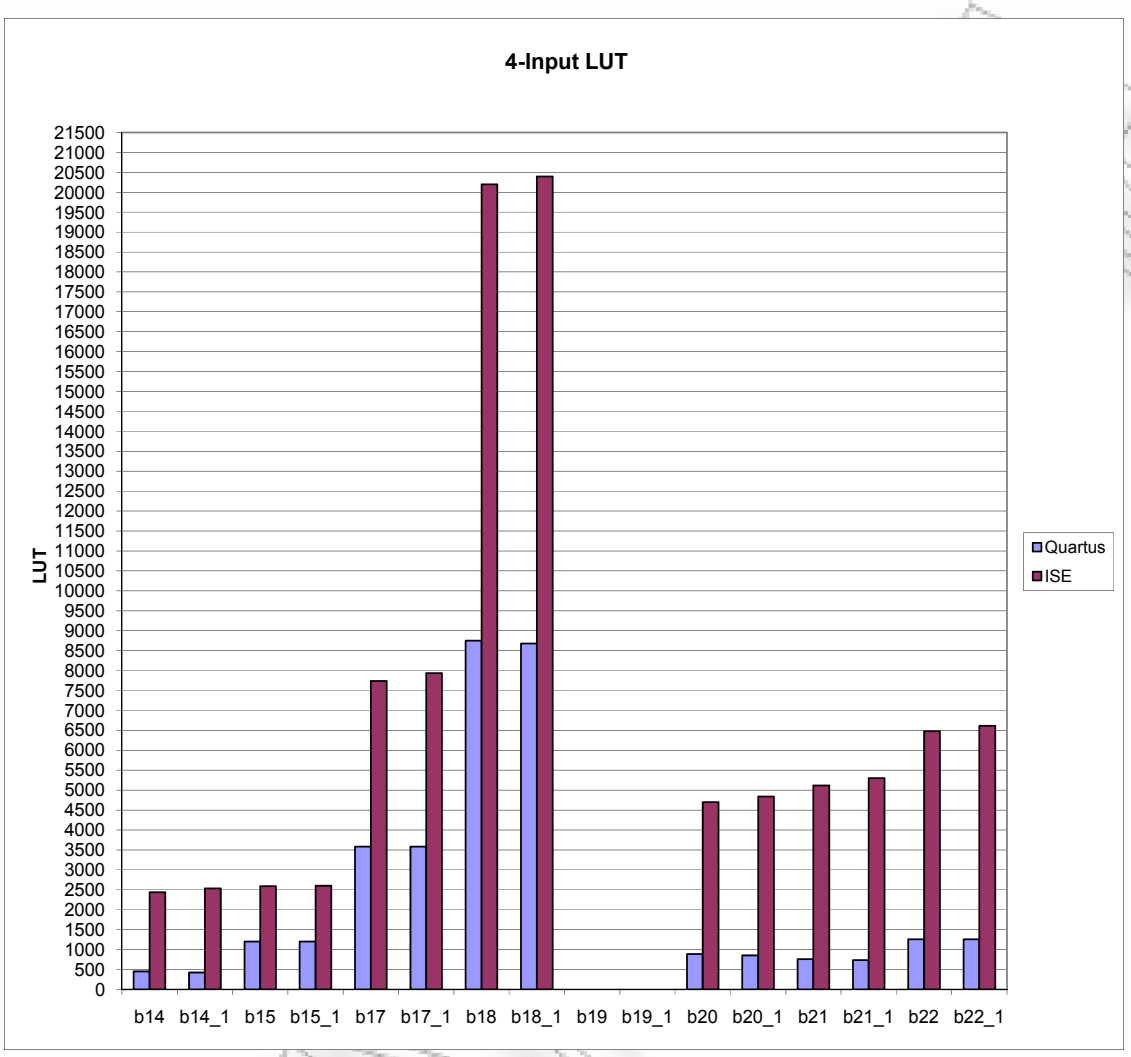
5.4.3. Πίνακες αναζήτησης LUT

Σε αυτή την ενότητα θα δούμε τους τεσσάρων εισόδων πίνακες αναζήτησης (4-input LUT) που χρησιμοποιούνται για κάθε σχεδίαση μετροπρογραμμάτων. Και εδώ χωρίσαμε σε δύο διαγράμματα τα αποτελέσματα όπως και στην προηγούμενη ενότητα για να έχουμε καλύτερη απεικόνιση αλλά και ευκολότερη εξαγωγή συμπερασμάτων. Έχουμε τα διαγράμματα των Εικόνων 5.4 και 5.5. Στην Εικόνα 5.4 βλέπουμε μια φυσιολογική κατανομή των LUT και στην Εικόνα 5.5 μια αύξηση από μεριάς του εργαλείου σχεδίασης ISE. Και πάλι εδώ φέρει ευθύνη η αρχιτεκτονική της συσκευής Spartan-3. Ενδεικτικά να σας αναφέρουμε ότι η αρχιτεκτονική Cyclone περιλαμβάνει χρήση και υποδιαιρέστερων LUT των τριών, δύο και μιας εισόδου. Για παράδειγμα ας δούμε τα αποτελέσματα του μετροπρογράμματος b18, βλέπουμε μια πολύ μεγάλη διαφορά μεταξύ του Quartus και του ISE. Στην

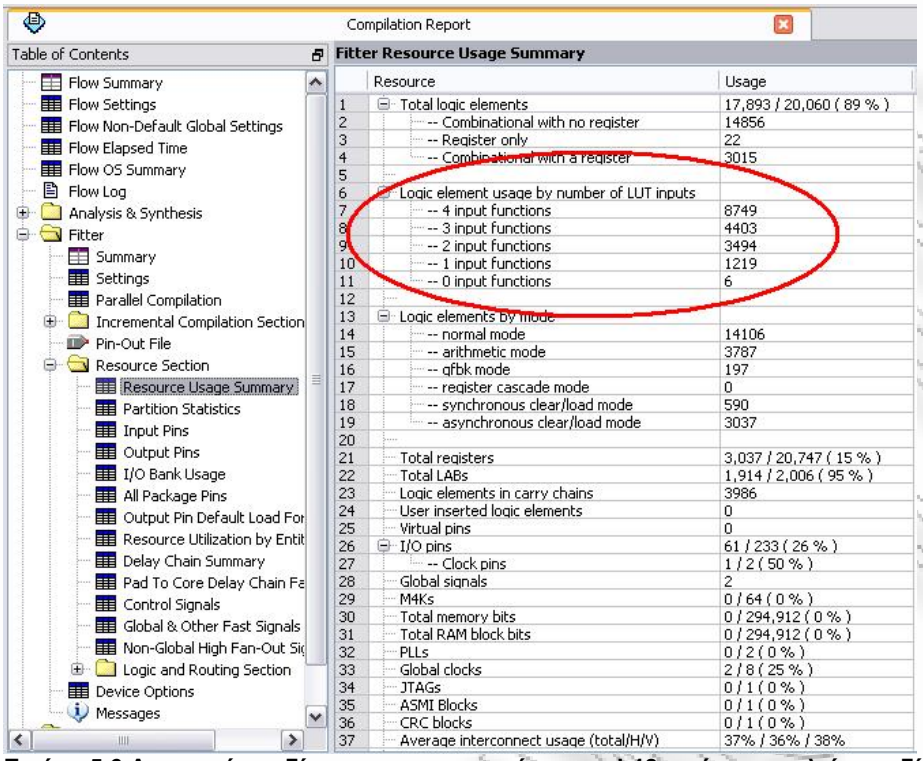
αναφορά που έβγαλε το εργαλείο Quartus II είχαμε ότι τα 17893 λογικά στοιχεία χρησιμοποιήθηκαν σε λειτουργίες από πίνακες αναζήτησης με μεγέθη, 8749 LUT των τεσσάρων εισόδων, 4403 LUT των τριών εισόδων, 3494 LUT των δύο εισόδων,1219 LUT της μίας εισόδου και τέλος 6 καμιάς εισόδου, Εικόνα 5.6. Σε αντίθεση το εργαλείο ISE τα μετράει όλα ως πίνακες αναζήτησης τεσσάρων εισόδων, άσχετα αν χρησιμοποιούνται λιγότερες είσοδοι, Εικόνα 5.7.



Εικόνα 5.4 Διάγραμμα τεσσάρων εισόδων πινάκων αναζήτησης (4-input LUT) που καταλαμβάνει η σχεδίαση των μετροπρογραμμάτων b01 .. b13, από τα εργαλεία σχεδίασης CAD Quartus II και ISE



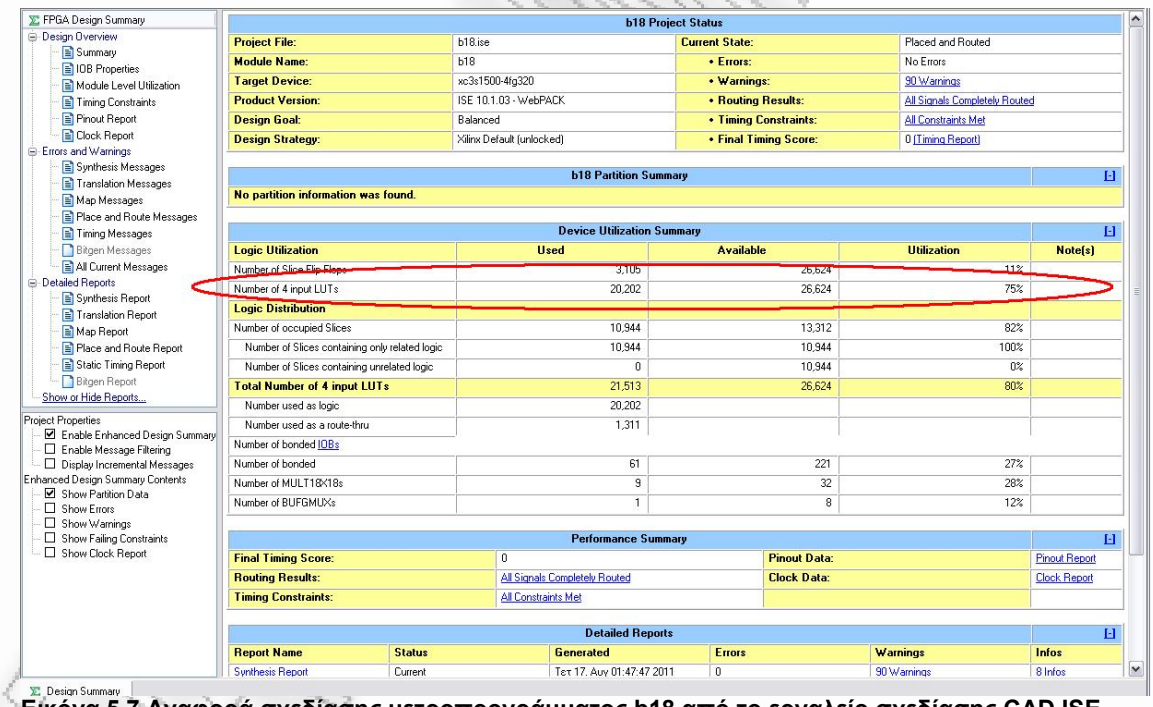
Εικόνα 5.5 Διάγραμμα τεσσάρων εισόδων πινάκων αναζήτησης (4-input LUT) που καταλαμβάνει η σχεδίαση των μετροπρογραμμάτων b14 .. b22_1, από τα εργαλεία σχεδίασης CAD Quartus II και ISE



Fitter Resource Usage Summary

	Resource	Usage
1	Total logic elements	17,893 / 20,060 (89 %)
2	-- Combinational with no register	14856
3	-- Register only	22
4	-- Combinational with a register	3015
5		
6	Logic element usage by number of LUT inputs	
7	-- 4 input functions	8749
8	-- 3 input functions	4403
9	-- 2 input functions	3494
10	-- 1 input functions	1219
11	-- 0 input functions	6
12		
13	Logic elements by mode	
14	-- normal mode	14106
15	-- arithmetic mode	3787
16	-- qfbk mode	197
17	-- register cascade mode	0
18	-- synchronous clear/load mode	590
19	-- asynchronous clear/load mode	3037
20		
21	Total registers	3,037 / 20,747 (15 %)
22	Total LABs	1,914 / 2,006 (95 %)
23	Logic elements in carry chains	3986
24	User inserted logic elements	0
25	Virtual pins	0
26	I/O pins	61 / 233 (26 %)
27	-- Clock pins	1 / 2 (50 %)
28	Global signals	2
29	M4Ks	0 / 64 (0 %)
30	Total memory bits	0 / 294,912 (0 %)
31	Total RAM block bits	0 / 294,912 (0 %)
32	PLLs	0 / 2 (0 %)
33	Global clocks	2 / 8 (25 %)
34	JTAGs	0 / 1 (0 %)
35	ASMI Blocks	0 / 1 (0 %)
36	CRC blocks	0 / 1 (0 %)
37	Average interconnect usage (total/H/V)	37% / 36% / 38%

Εικόνα 5.6 Αναφορά σχεδίασης μετροπρογράμματος b18 από το εργαλείο σχεδίασης CAD Quartus II



b18 Project Status

Project File:	b18.ise	**Current State:**	Placed and Routed
Module Name:	b18	**• Errors:**	No Errors
Target Device:	xc3s1500-4fg320	**• Warnings:**	90 Warnings
Product Version:	ISE 10.1.03 - WebPACK	**• Routing Results:**	All Signals Completely Routed
Design Goal:	Balanced	**• Timing Constraints:**	All Constraints Met
Design Strategy:	Xilinx Default (unlocked)	**• Final Timing Score:**	0 (Timing Report)

b18 Partition Summary

No partition information was found.

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	3,105	26,624	11%	
Number of 4 input LUTs	20,202	26,624	75%	
Logic Distribution				
Number of occupied Slices	10,944	13,312	82%	
Number of Slices containing only related logic	10,944	10,944	100%	
Number of Slices containing unrelated logic	0	10,944	0%	
Total Number of 4 input LUTs	21,513	26,624	80%	
Number used as logic	20,202			
Number used as a route-thru	1,311			
Number of bonded IOBs				
Number of bonded	61	221	27%	
Number of MULT18X18s	9	32	28%	
Number of BUFGMUXs	1	8	12%	

Performance Summary

Final Timing Score:	0	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

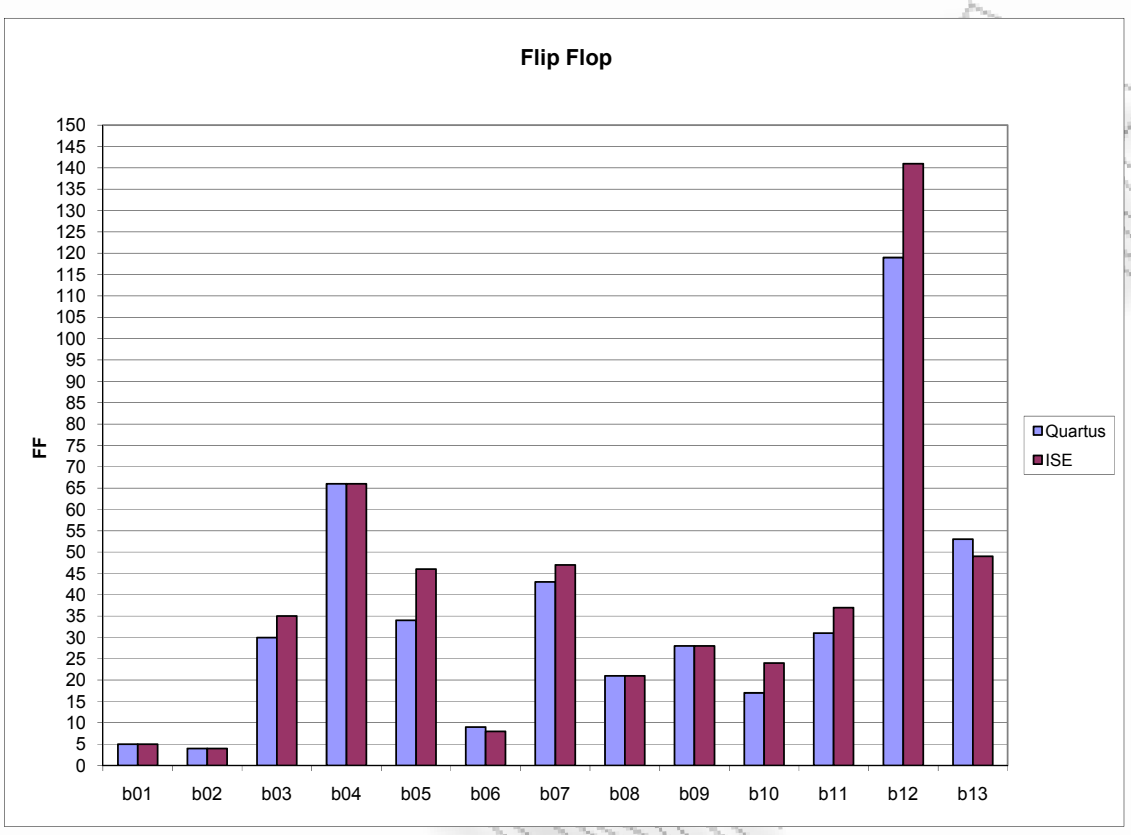
Detailed Reports

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Τετ 17. Αυγ 01:47:47 2011	0	90 Warnings	8 Infos

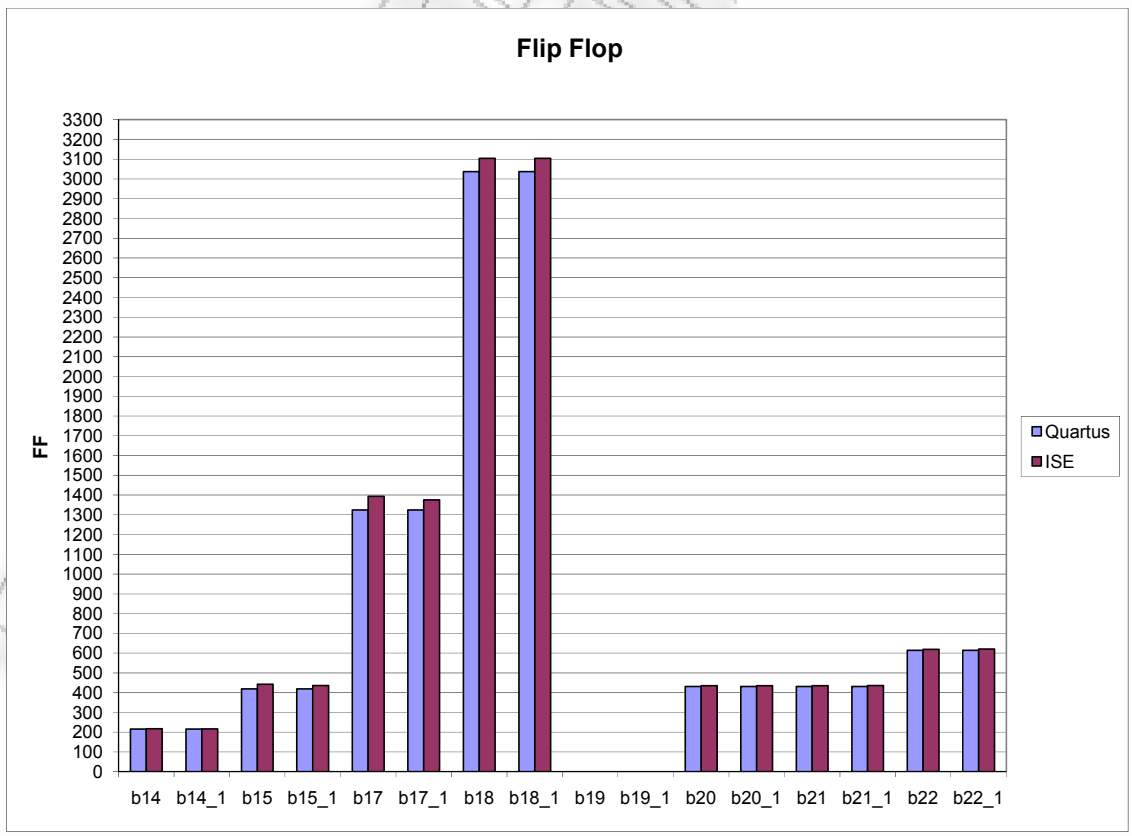
Εικόνα 5.7 Αναφορά σχεδίασης μετροπρογράμματος b18 από το εργαλείο σχεδίασης CAD ISE

5.4.4. Καταχωρητές

Και εδώ όπως και στις προηγούμενες ενότητες θα διαχωρίσουμε σε δυο διαγράμματα τα αποτελέσματα, τα οποία φαίνονται στις Εικόνες 5.8 και 5.9. Όπως μπορείτε να παρατηρήσετε υπάρχει φυσιολογική κατανομή των καταχωρητών εκτός αυτή των μετροπρογράμματων b05 και b12 όπου υπάρχει μια μικρή διαφορά. Αυτή η διαφορά δεν μπορούμε να πούμε ότι είναι πολύ μεγάλη τέτοια ώστε να χρίζει περαιτέρω μελέτης.



Εικόνα 5.8 Διάγραμμα καταχωρητών (registers - flip flop) που καταλαμβάνει η σχεδίαση των μετροπρογραμμάτων b01 .. b13, από τα εργαλεία σχεδίασης CAD Quartus II και ISE



Εικόνα 5.9 Διάγραμμα καταχωρητών (registers - flip flop) που καταλαμβάνει η σχεδίαση των μετροπρογραμμάτων b14 .. b22_1, από τα εργαλεία σχεδίασης CAD Quartus II και ISE

5.5. Παρατηρήσεις - Τελικά συμπεράσματα

Από τα πειράματα που διεξήχθησαν διαπιστώσαμε ότι υπάρχει μια μικρή απόκλιση (κάποιες φόρες λίγο μεγαλύτερη από όσο περιμέναμε) στις τιμές των αποτελεσμάτων, αλλά δεν μπορούμε να πούμε ότι υπήρξαν μεγάλες διαφορές. Οι διαφορές που υπάρχουν οφείλονται στη διαφορετικότητα των αρχιτεκτονικών δομών των συσκευών Cyclone και Spartan–3, αλλά και αυτές των προγραμμάτων σχεδίασης CAD Quartus II και ISE. Δεν είδαμε απόκλιση των τιμών σε μεγάλο αριθμό πειραμάτων τέτοια ώστε να μας κάνει σαφές την ύπαρξη κάποιου ανώτερου η κατωτέρου εργαλείου, αλλά και την ύπαρξη κάποιας ανώτερης η κατώτερης αρχιτεκτονικής.

Σαν τελικό συμπέρασμα διαπιστώνουμε ότι και τα δύο εργαλεία σχεδίασης CAD Quartus II και ISE έχουν τις ίδιες δυνατότητες σχεδίασης αλλά και ευχρηστίας στο να μπορεί κάνεις να σχεδιάζει με "εμπιστοσύνη". Και τα δύο παρέχουν δυνατότητες χειροκίνητης μορφοποίησης σε κάθε στάδιο σχεδίασης, τέτοια ώστε ο χρήστης να μπορεί να σχεδιάζει το έργο του σύμφωνα με τις απαιτήσεις του. Στα παραπάνω πειράματα που διεξήγαμε είχαμε καταφέρει χειροκίνητα να πετύχουμε καλύτερες συχνότητες λειτουργίας από αυτές που μας έδινε το εργαλείο σχεδίασης (και στα δυο), αλλά θέλαμε να διαπιστώσουμε τη δυνατότητα του εργαλείου σχεδίασης να πετύχει από μόνο του τη βέλτιστη σχεδίαση στη συσκευή FPGA.

Όσο για τις αρχιτεκτονικές FPGA που είδαμε και αυτές έδωσαν τη δυνατότητα στα αντίστοιχα εργαλεία σχεδίασης να αξιοποιήσουν τους πόρους που πρόσφεραν. Δεν είδαμε κατάχρηση πόρων αλλά και ούτε παραμέληση. Είδαμε υψηλά πόστα κάλυψης όπως αυτή του 89% των λογικών στοιχείων της συσκευής EP1C20F324C6, από το μετροπρόγραμμα b18 το οποίο έγινε με το εργαλείο Quartus II στην αρχιτεκτονική Cyclone. Όπως επίσης το ποσοστό κάλυψης 93% των λογικών στοιχείων της συσκευής XC3S400-4TQ144, από το μετροπρόγραμμα b22_1 με χρήση του εργαλείο ISE και την αρχιτεκτονική Spartan-3.

Ως τελικό συμπέρασμα μπορούμε να πούμε ότι οι αρχιτεκτονικές και τα εργαλεία σχεδίασης των δύο εταιρειών επιτυγχάνουν παρόμοιες επιδόσεις.

6. Μελλοντικές επεκτάσεις

Μια πρώτη σκέψη για μελλοντική επέκταση της διατριβής είναι να εκτελέσουμε το πειραματικό μέρος και στο εργαλείο σχεδίασης ώστε να μπορέσουμε να συγκρίνουμε την αρχιτεκτονική δομή των συσκευών AT40K με αυτές των Cylcone και Spartan-3.

Μια δεύτερη σκέψη για μελλοντικές έρευνες είναι η μελέτη πιο εξελεγμένων αρχιτεκτονικών δομών και συσκευών. Θα μπορούσε να είναι από τις υπάρχουσες εταιρίες αλλά θα μπορούσε να είναι και από άλλες διαφορετικές.

Μια τρίτη είναι να γίνει πειραματισμός με πιο πολύπλοκες σχεδιάσεις που να περιέχουν τη χρήση μνημών RAM αλλά και τη χρήση πυρήνων επεξεργαστή.

7. Αναφορές

7.1. Βιβλία

[Ale] Γεώργιος Αλεξίου. Μικροεπεξεργαστές, Ελληνικό Ανοικτό Πανεπιστήμιο, Πάτρα 2000. ISBN 960-538-197-4
[B&V] Stephen Brown, Zvonko Vranesic. Σχεδίαση Ψηφιακών Συστημάτων με τη Γλώσσα VHDL, Εκδόσεις Τζιόλα Θεσσαλονίκη 2001. ISBN 960-8050-50-2
[HVZ] Carl Hammacher, Zvonko Vranesic, Safwat Zaky. Οργάνωση και Αρχιτεκτονική Ηλεκτρονικών Υπολογιστών, Εκδόσεις Επίκεντρο, Θεσσαλονίκη 2006. ISBN 960-458-000-0
[Lin] Παναγιώτης Λινάρδης. Ψηφιακή σχεδίαση Ι, Ελληνικό Ανοικτό Πανεπιστήμιο, Πάτρα 2001. ISBN 960-538-195-8
[Man1] Morris Mano. Ψηφιακή σχεδίαση, 2 Έκδοση, Εκδόσεις Παπασωτηρίου, Αθήνα 1992. ISBN 960-7182-01-4
[**Man2]** Morris Mano. Ψηφιακή σχεδίαση, 3 Έκδοση, Εκδόσεις Παπασωτηρίου, Αθήνα 2003. ISBN 960-7530-63-2
[Nik] Δημήτριος Νικολός. Αρχιτεκτονική Υπολογιστών Ι, Ελληνικό Ανοικτό Πανεπιστήμιο, Πάτρα 2000. ISBN 960-538-196-6
[P&H] David A.Patterson, John L. Hennessy. Οργάνωση και Σχεδίαση Υπολογιστών, Διασύνδεση Υλικού και Λογισμικού, 3 Αμερικάνικη Έκδοση. Εκδόσεις Κλειδάριθμος 2006. ISBN 960-209-906-2
[Sko] Αθανάσιος Σκόδρας. Ψηφιακή σχεδίαση ΙΙ, Ελληνικό Ανοικτό Πανεπιστήμιο, Πάτρα 2008. ISBN 978-960-538-689-4
[Sta] William Staling. Οργάνωση και Αρχιτεκτονική Υπολογιστών, Εκδόσεις Τζιόλα Θεσσαλονίκη 2003. ISBN 960-418-008-8

7.2. Ηλεκτρονικά Βιβλία – Σημειώσεις – Εγχειρίδια

[Alt1] Cyclone Device Handbook, Altera, Copyright © 2008 Altera Corporation.
[Alt2] Quartus® II Introduction for VHDL Users, Altera, Copyright © 2010 Altera Corporation.
[Alt3] Quartus II Handbook v11.0.0, Altera, Copyright © 2011 Altera Corporation.
[Atm1] AT40K Device Datasheets, Atmel, Copyright © 2002 Atmel Corporation.
[Atm2] Integrated Development System – Figaro Tutorial, Atmel, Copyright © 2002 Atmel Corporation.
[Atm3] Integrated Development System – Figaro User Guide, Atmel, Copyright © 2002 Atmel Corporation.
[Giz1] Δημήτρης Γκιζόπουλος. Σημειώσεις / Διαφάνειες Διδασκαλίας, Μοντελοποίηση Ενσωματωμένων Υπολογιστικών Συστημάτων, Πανεπιστήμιο Πειραιώς. Τμήμα Πληροφορικής, Πρόγραμμα Μεταπτυχιακών Σπουδών «Προηγμένα συστήματα πληροφορικής», κατεύθυνση «Τεχνολογία Ενσωματωμένων Υπολογιστικών Συστημάτων». Πειραιάς 2009.
[Giz 2] Δημήτρης Γκιζόπουλος. Σημειώσεις / Διαφάνειες Διδασκαλίας, Σύγχρονοι Επεξεργαστές, Πανεπιστήμιο Πειραιώς. Τμήμα Πληροφορικής, Πρόγραμμα Μεταπτυχιακών Σπουδών «Προηγμένα συστήματα πληροφορικής», κατεύθυνση «Τεχνολογία Ενσωματωμένων Υπολογιστικών Συστημάτων». Πειραιάς 2009.
[Giz 3] Δημήτρης Γκιζόπουλος. Σημειώσεις / Διαφάνειες Διδασκαλίας, Λειτουργικά Συστήματα Πραγματικού Χρόνου, Πανεπιστήμιο Πειραιώς. Τμήμα Πληροφορικής, Πρόγραμμα Μεταπτυχιακών Σπουδών «Προηγμένα συστήματα πληροφορικής», κατεύθυνση «Τεχνολογία Ενσωματωμένων Υπολογιστικών Συστημάτων». Πειραιάς 2009.
[Har] Hardi Electronicks AB. VHDL handbook. Hardi Electronicks AB, Copyright © 1997-2000
[Max] Clive "Max" Maxfield. The Design Warrior's Guide to FPGAs, Copyright 2004 Mentor Graphics Corporation and Xilinx, Inc. Elsevier. ISBN 0-7506-7604-3
[PsM1] Μιχάλης Ψαράκης. Σημειώσεις / Διαφάνειες Διδασκαλίας, Προηγμένης Ψηφιακής Σχεδίασης, Πανεπιστήμιο Πειραιώς. Τμήμα Πληροφορικής, Πρόγραμμα Μεταπτυχιακών Σπουδών «Προηγμένα συστήματα πληροφορικής», κατεύθυνση «Τεχνολογία Ενσωματωμένων Υπολογιστικών Συστημάτων». Πειραιάς 2009.
[PsM2] Μιχάλης Ψαράκης. Εργαστηριακές Σημειώσεις: Σύντομος Οδηγός Εκμάθησης του Λογισμικού Xilinx ISE, Προηγμένης Ψηφιακής Σχεδίασης, Πανεπιστήμιο Πειραιώς. Τμήμα Πληροφορικής, Πρόγραμμα Μεταπτυχιακών Σπουδών «Προηγμένα συστήματα πληροφορικής», κατεύθυνση «Τεχνολογία Ενσωματωμένων Υπολογιστικών Συστημάτων». Πειραιάς 2009.

[PsM3] Μιχάλης Ψαράκης. Σημειώσεις / Διαφάνειες Διδασκαλίας, Τεχνικές Ανάπτυξης Ενσωματωμένου Λογισμικού, Πανεπιστήμιο Πειραιώς. Τμήμα Πληροφορικής, Πρόγραμμα Μεταπτυχιακών Σπουδών «Προηγμένα συστήματα πληροφορικής», κατεύθυνση «Τεχνολογία Ενσωματωμένων Υπολογιστικών Συστημάτων». Πειραιάς 2009.
[Xln1] Spartan-3 FPGA Family Data Sheet, Xilinx, Copyright © 2009 Xilinx Corporation.
[Xln2] Spartan-3 Generation FPGA User Guide, Xilinx, Copyright © 2010 Xilinx Corporation.
[Xln3] ISE 10.1 Quick Start Tutorial, Xilinx, Copyright © 2008 Xilinx Corporation.
[Xln4] Xilinx ISE Design Suite 10.1 Software Manuals, Xilinx, Copyright © 2008 Xilinx Corporation.

7.3. Internet

[Alt4] Altera Corporation, http://www.altera.com/
[Atm4] Atmel Corporation, http://www2.atmel.com/
[I3E] IEEE, http://www.ieee.org/index.html
[int1] Wikipedia, http://en.wikipedia.org/wiki/Hardware_description_language
[int2] Wikipedia, http://en.wikipedia.org/wiki/7400_series
[int3] http://www.ustudy.in/node/7586
[ITC] ITC'99 Benchmarks, http://www.cad.polito.it/downloads/tools/itc99.html
[Mdls] Mentor Graphics, http://model.com/content/modelsim-downloads
[PdT] Politecnico di Torino, http://www.polito.it/index.en.php
[TI] Texas Instruments, http://www.ti.com/
[Xln5] Xilinx Corporation, http://www.xilinx.com/

7.4. Προγράμματα – Υλικό

Για την υλοποίηση της μεταπτυχιακής διατριβής χρησιμοποιήθηκαν τα παρακάτω προγράμματα:

- Microsoft Office Word 2003
- Microsoft Office Excel 2003
- Microsoft Office Visio 2003
- ISE 10.1, Xilinx
- Quartus II 11.0.0, Altera
- Integrated Development System – Figaro, Atmel

Οι υλοποιήσεις έγιναν σε ηλεκτρονικό υπολογιστή με λειτουργικό σύστημα Microsoft Windows XP Professional Version 2002 Service Pack 3 και φέρει επεξεργαστή Intel Core2 CPU 4400@2.00GHz 2.01GHz, 1,97 GB RAM.

8. Παραρτήματα

8.1. Παράρτημα Α

Σύμβολο Γλώσσας	Όνομα Γλώσσας	Σημείωση
ABEL	Advanced Boolean Expression Language	
AHDL	Altera HDL	a proprietary language from Altera
AHPL	A Hardware Programing language	
Bluespec		high-level HDL originally based on Haskell, now with a SystemVerilog syntax
C-to-Verilog		Converter from C to Verilog
Confluence		a functional HDL; has been discontinued)
CoWareC		a C-based HDL by CoWare. Now discontinued in favor of SystemC
CUPL		a proprietary language from Logical Devices, Inc.
Handel-C		a C-like design language
HJJ	Hardware Join Java	based on Join Java
HML		based on SML
Hydra		based on Haskell
Impulse C	another C-like HDL	
ParC	Parallel C++	C++ extended with HDL style threading and communication for task-parallel programming
JHDL		based on Java
Lava		based on Haskell
M		A HDL from Mentor Graphics
MyHDL		based on Python
PALASM		for Programmable Array Logic (PAL) devices
ROCCC 2.0	Riverside Optimizing Compiler for Configurable Computing	Free and open-source C to HDL tool
RHDL		based on the Ruby programming language
Ruby (hardware description language)		
SystemC		a standardized class of C++ libraries for high-level behavioral and transaction modeling of digital hardware at a high level of abstraction, i.e. system-level
SystemVerilog		a superset of Verilog, with enhancements to address system-level design and verification
SystemTCL		SDL based on Tcl.
Verilog		most widely-used and well-supported HDL
VHDL	VHSIC HDL	most widely-used and well-supported HDL

Πίνακας Α.1 Είδη γλωσσών HDL που έχουν αναπτυχθεί κατά καιρούς

8.2. Παράρτημα Β

8.2.1. Κώδικας VHDL *adderSubtractor.vhd* [Alt2]

```
--------------------------------------------------------------------------------
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
-- Top-level entity
ENTITY addersubtractor IS
GENERIC (n : INTEGER := 16) ;
PORT (A, B : IN STD_LOGIC_VECTOR(n-1 DOWNTO 0) ;
Clock, Reset, Sel, AddSub : IN STD_LOGIC ;
Z : BUFFER STD_LOGIC_VECTOR(n-1 DOWNTO 0) ;
Overflow : OUT STD_LOGIC) ;
END addersubtractor ;
ARCHITECTURE Behavior OF addersubtractor IS
SIGNAL G, H, M, Areg, Breg, Zreg, AddSubR_n : STD_LOGIC_VECTOR(n-1 DOWNTO 0) ;
SIGNAL SelR, AddSubR, carryout, over_flow : STD_LOGIC ;
COMPONENT mux2to1
GENERIC (k : INTEGER := 8) ;
PORT (V, W : IN STD_LOGIC_VECTOR(k-1 DOWNTO 0) ;
Sel : IN STD_LOGIC ;
F : OUT STD_LOGIC_VECTOR(k-1 DOWNTO 0)) ;
END COMPONENT ;
COMPONENT adderk
GENERIC (k : INTEGER := 8) ;
PORT (carryin : IN STD_LOGIC ;
X, Y : IN STD_LOGIC_VECTOR(k-1 DOWNTO 0) ;
S : OUT STD_LOGIC_VECTOR(k-1 DOWNTO 0) ;
carryout : OUT STD_LOGIC) ;
END COMPONENT ;
BEGIN
PROCESS (Reset, Clock)
BEGIN
IF Reset = '1' THEN
Areg <= (OTHERS => '0'); Breg <= (OTHERS => '0');
Zreg <= (OTHERS => '0'); SelR <= '0'; AddSubR <='0'; Overflow <= '0';
ELSIF Clock'EVENT AND Clock = '1' THEN
Areg <= A; Breg <= B; Zreg <= M;
SelR <= Sel; AddSubR <= AddSub; Overflow <= over_flow;
END IF ;
END PROCESS ;
nbit_adder: adderk
GENERIC MAP (k => n)
PORT MAP (AddSubR, G, H, M, carryout) ;
multiplexer: mux2to1
GENERIC MAP (k => n)
PORT MAP (Areg, Z, SelR, G) ;
AddSubR_n <= (OTHERS => AddSubR) ;
H <= Breg XOR AddSubR_n ; Z <= Zreg ;
over_flow <= carryout XOR G(n-1) XOR H(n-1) XOR M(n-1) ;
END Behavior;
-- k-bit 2-to-1 multiplexer
```

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY mux2to1 IS
GENERIC (k : INTEGER := 8) ;
PORT (V, W : IN STD_LOGIC_VECTOR(k-1 DOWNTO 0) ;
Sel : IN STD_LOGIC ;
F : OUT STD_LOGIC_VECTOR(k-1 DOWNTO 0)) ;
END mux2to1 ;
ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
PROCESS (V, W, Sel)
BEGIN
IF Sel = '0' THEN
F <= V ;
ELSE
F <= W ;
END IF ;
END PROCESS ;
END Behavior ;
-- k-bit adder
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_signed.all ;
ENTITY adderk IS
GENERIC (k : INTEGER := 8) ;
PORT (carryin : IN STD_LOGIC ;
X, Y : IN STD_LOGIC_VECTOR(k-1 DOWNTO 0) ;
S : OUT STD_LOGIC_VECTOR(k-1 DOWNTO 0) ;
carryout: OUT STD_LOGIC) ;
END adderk ;
ARCHITECTURE Behavior OF adderk IS
SIGNAL Sum : STD_LOGIC_VECTOR(k DOWNTO 0) ;
BEGIN
Sum <= ('0' & X) + ('0' & Y) + carryin ;
S <= Sum(k-1 DOWNTO 0) ;
carryout <= Sum(k) ;
END Behavior ;
```

8.2.2. Κώδικας VHDL *counter.vhd* [Xln3]

```
----------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 15:16:21 06/22/2011
-- Design Name:
-- Module Name: counter - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
```

```
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx primitive in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity counter is
Port (CLOCK : in STD_LOGIC;
DIRECTION : in STD_LOGIC;
COUNT_OUT : out STD_LOGIC_VECTOR (3 downto 0));
end counter;
architecture Behavioral of counter is
signal count_int : std_logic_vector(3 downto 0) := "0000";
begin
process (CLOCK)
begin
if CLOCK='1' and CLOCK'event then
if DIRECTION='1' then
count_int <= count_int + 1;
else
count_int <= count_int - 1;
end if;
end if;
end process;
COUNT_OUT <= count_int;
end Behavioral;
```

8.2.3. Κώδικας VHDL *file even_par.vhd* [PsM2]

```
----------------------------------------------------------------------------------
-- VHDL Test Bench Created from source file even_par.vhd -- 18:15:04 04/03/2007
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test. Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
ENTITY even_par_testbench_vhd_tb IS
END even_par_testbench_vhd_tb;
ARCHITECTURE behavior OF even_par_testbench_vhd_tb IS
COMPONENT even_par
PORT(
```

```
a : IN std_logic;
b : IN std_logic;
c : IN std_logic;
d : IN std_logic;
ep : OUT std_logic
);
END COMPONENT;
SIGNAL a : std_logic;
SIGNAL b : std_logic;
SIGNAL c : std_logic;
SIGNAL d : std_logic;
SIGNAL ep : std_logic;
BEGIN
uut: even_par PORT MAP(
a => a,
b => b,
c => c,
d => d,
ep => ep
);
-- *** Test Bench - User Defined Section ***
tb : PROCESS
BEGIN
-- All possible input combinations
a <= '0'; b <= '0'; c <= '0'; d <= '0';
wait for 20 ns;
a <= '0'; b <= '0'; c <= '0'; d <= '1';
wait for 20 ns;
a <= '0'; b <= '0'; c <= '1'; d <= '0';
wait for 20 ns;
a <= '0'; b <= '0'; c <= '1'; d <= '1';
wait for 20 ns;
a <= '0'; b <= '1'; c <= '0'; d <= '0';
wait for 20 ns;
a <= '0'; b <= '1'; c <= '0'; d <= '1';
wait for 20 ns;
a <= '0'; b <= '1'; c <= '1'; d <= '0';
wait for 20 ns;
a <= '0'; b <= '1'; c <= '1'; d <= '1';
wait for 20 ns;
a <= '1'; b <= '0'; c <= '0'; d <= '0';
wait for 20 ns;
a <= '1'; b <= '0'; c <= '0'; d <= '1';
wait for 20 ns;
a <= '1'; b <= '0'; c <= '1'; d <= '0';
wait for 20 ns;
a <= '1'; b <= '0'; c <= '1'; d <= '1';
wait for 20 ns;
a <= '1'; b <= '1'; c <= '0'; d <= '0';
wait for 20 ns;
a <= '1'; b <= '1'; c <= '0'; d <= '1';
wait for 20 ns;
a <= '1'; b <= '1'; c <= '1'; d <= '0';
wait for 20 ns;
a <= '1'; b <= '1'; c <= '1'; d <= '1';
```

```
wait; -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***
END;
```

8.2.4. Κώδικας VHDL *intiadd.vhd* [Atm2]

```
--------------------------------------------------------------------------------
-- Do not delete following library and use clauses.
--
library atmel;
use atmel.components.all;
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY initadd IS
generic(WIDTH : integer :=8);
PORT (
clk, rst : IN std_logic;
datain : IN std_logic_vector(WIDTH-1 downto 0);
dataout : OUT std_logic_vector(WIDTH-1 downto 0);
result : OUT std_logic_vector(WIDTH downto 0)
);
END initadd;
ARCHITECTURE behaviour OF initadd IS
SIGNAL inta,temp_dataout : std_logic_vector(WIDTH-1 downto 0);
SIGNAL zero : std_logic;
SIGNAL unused : std_logic;
BEGIN
zero <= '0';
-- HDLPlanner Instance dff_prl
-- Do not **DELETE** previous line
u1 : dff_prl
GENERIC MAP (WIDTH => 8)
PORT MAP(
DATA => datain,
RN => rst,
CLK => clk,
Q => inta
);
-- Do not **DELETE** next line
-- HDLPlanner End Instance dff_prl
-- HDLPlanner Instance dff_prl
-- Do not **DELETE** previous line
u2 : dff_prl
GENERIC MAP (WIDTH => 8)
PORT MAP(
DATA => inta,
RN => rst,
CLK => clk,
Q => temp_dataout
);
-- Do not **DELETE** next line
-- HDLPlanner End Instance dff_prl
-- HDLPlanner Instance addSigned
```

```
-- Do not **DELETE** previous line
u3 : addSigned
GENERIC MAP (WIDTH => 8)
PORT MAP (
DATAA => inta,
DATAB => temp_dataout,
CIN => zero,
SUM => result(WIDTH-1 downto 0),
COUT => result(WIDTH),
OVERFLOW => unused
);
-- Do not **DELETE** next line
-- HDLPlanner End Instance addSigned
dataout <= temp_dataout;
END behaviour;
```

8.3. Παράρτημα Γ

8.3.1. Μετροπρογράμματα ITC'99

Τα μετροπρογράμματα που έχουν χρησιμοποιηθεί στις υλοποιήσεις για κάθε εργαλείο σχεδίασης, είναι επισυναπτόμενα μέσα σε ένα οπτικό δίσκο που συνοδεύει τη μεταπτυχιακή διατριβή. Για κάθε εργαλείο υπάρχει ένας ξεχωριστός φάκελος με τα προγράμματα και τα παραγόμενα αρχεία.