Σχεδίαση ενός κυκλώματος πολύ υψηλής ολοκλήρωσης για την εκτέλεση του αλγορίθμου βιοπληροφορικής BLASTn

του

Μαστρογιαννόπουλου Δημήτρη

Διπλωματική εργασία

Επιτροπή

Απόστολος Δόλλας, επιβλέπων
Διονύσιος Πνευματικάτος
Ιωάννης Παπαευσταθίου

Πολυτεχνείο Κρήτης
Μάρτιος 2009

Πρόλογος

Η εργασία αυτή εκπονήθηκε με στόχο την εκπλήρωση των υποχρεώσεων του γράφοντος στο τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών (ΗΜΜΥ) του Πολυτεχνείου Κρήτης για την απόκτηση διπλώματος. Το θέμα της είναι η σχεδίαση ενός κυκλώματος πολύ υψηλής ολοκλήρωσης που εκτελεί το δημοφιλή αλγόριθμο βιοπληροφορικής BLASTn.

Σε αυτό το σημείο θέλω να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Απόστολο Δόλλα, τον κ. Ευριπίδη Σωτηριάδη, τον κ. Κυπριανό Παπαδημητρίου, τον κ. Διονύσιο Πνευματικάτο και τον κ. Χρήστο Σωτηρίου, καθώς χωρίς τη συμβολή τους, η διεκπεραίωση αυτής της εργασίας δε θα ήταν δυνατή.

Περιεχόμενα

1. Εισαγωγή

Σε μια πραγματικότητα όπως η σημερινή, είναι ίσως πιο σημαντική από ποτέ η ταχύτητα. Τώρα πια σε ό,τι κι αν κάνουμε δε μετρά μόνο το αποτέλεσμα, άλλα κι ο χρόνος που θα χρειαστούμε για να οδηγηθούμε σε αυτό. Ένας τομέας οπού ο χρόνος αποτελεί τροχοπέδη στην ταχύτερη ανάπτυξη του, είναι η βιολογία, που τυγχάνει να είναι και από τους πλέον σημαντικούς, καθώς μπορούμε να έχουμε τεράστια οφέλη στην παγκόσμια υγεία και όχι μόνο.

Εδώ και αρκετά χρόνια η ιδέα της εξερεύνησης του ανθρώπινου γονιδιώματος έχει κεντρίσει βιολόγους, ερευνητές ακόμα και απλούς ανθρώπους σε όλο τον κόσμο. Το κέρδος από την περάτωση ενός τέτοιου έργου προφανές και δεν περιορίζεται μόνο στη θεραπεία διάφορων, σήμερα ανίατων, ασθενειών. Δυστυχώς όμως είναι κάτι που απαιτεί ένα τεράστιο αριθμό υπολογισμών, συνεπώς ήταν μια ιδέα που μέχρι το δεύτερο μισό του εικοστού αιώνα παρέμενε στα συρτάρια. Από το 1970 και ύστερα με την σημαντική και ραγδαία εξέλιξη των υπολογιστών, η ανάλυση του ανθρώπινου γονιδιώματος επανήλθε και πάλι στο προσκήνιο καθώς άρχισε να γίνεται εφικτή.

Από την αρχή της χρήσης των υπολογιστών από βιολόγους, έχουν αναπτυχθεί πάρα πολλοί αλγόριθμοι βιοπληροφορικής. Ο πλέον δημοφιλής και χρήσιμος αλγόριθμος είναι ο BLAST. Δυστυχώς όμως ο αλγόριθμος αυτός είναι πολύ υπολογιστικά ακριβός, κάτι που σημαίνει πως ο χρόνος εκτέλεσης είναι αρκετά μεγάλος, ώστε να αρχίζει να έχει νόημα η υλοποίηση ενός System On Chip (SOC) για τον αλγόριθμο αυτό. Αυτό είναι και το κίνητρο αυτής της εργασίας, η δημιουργία ενός κυκλώματος πολύ υψηλής ολοκλήρωσης (VLSI) που να υλοποιεί τον αλγόριθμο BLASTn, την κυριότερη παραλλαγή του BLAST.

1.1 Οργάνωση της διπλωματικής εργασίας

Η εργασία αυτή αποτελείται από έξι κεφάλαια, με το παρόν να είναι το πρώτο. Στο δεύτερο κεφάλαιο αναφέρεται η σχετική έρευνα που έγινε στα πλαίσια αυτής της εργασίας τόσο για τον αλγόριθμο όσο και για τα εργαλεία που χρειάστηκαν στο τμήμα της υλοποίησης. Στο τρίτο κεφάλαιο παρουσιάζεται η αρχιτεκτονική που σχεδιάστηκε για την αποτελεσματική μεταφορά του αλγορίθμου σε ολοκληρωμένο. Στο τέταρτο κεφάλαιο αναλύεται η σχεδιαστική ροή που υιοθετήθηκε για την φυσική σχεδίαση του chip. Στο πέμπτο κεφάλαιο γίνεται μια γενική συζήτηση για ζητήματα με περιθώρια βελτίωσης τόσο στην σχεδίαση όσο και σε άλλα θέματα. Τέλος αναγράφεται η βιβλιογραφία και δίνονται παραρτήματα με πιο τεχνικές πληροφορίες.

1.2 Συνεισφορά της διπλωματικής

Σε αυτήν την εργασία ακολουθήθηκε μια αρκετά διαφορετική προσέγγιση για την δημιουργία chip, από αυτήν που διδάσκεται στο μάθημα «Σχεδιασμός κυκλωμάτων VLSI», του Πολυτεχνείου Κρήτης. Στο μάθημα αυτό ακολουθείται η παραδοσιακή μέθοδος σχεδίασης κυκλωμάτων με το χέρι, που αποτελεί απαραίτητη γνώση, για την μέθοδο που υιοθετήθηκε στην εργασία αυτή. Στην εργασία αυτή μελετήθηκαν οι παρακάτω διαδικασίες:

- Δημιουργία HDL περιγραφής ενός κυκλώματος, ικανής να περάσει από λογική σύνθεση.
- Λογική σύνθεση HDL περιγραφών και απεικόνισή τους στα κύτταρα μιας τεχνολογίας με χρήση κατάλληλων εργαλείων.
- Εφαρμογή περιορισμών για την επίτευξη επιθυμητών χαρακτηριστικών, όπως χρονισμός, κατανάλωση κ.α ενός netlist.
- Τοποθέτηση και διασύνδεση των κυττάρων ενός netlist για την παραγωγή ενός ολοκληρωμένου κυκλώματος.
- Δημιουργία απαραίτητων δομών για την λειτουργία ενός chip, όπως padframe, power plan κ.α.
- Βελτιστοποίηση ενός layout για την επίτευξη επιθυμητών χαρακτηριστικών όπως χρονισμός, κατανάλωση, διαστάσεις κ.α.
- Δημιουργία όλων των απαραίτητων δομών για την επαλήθευση της σχεδίασης.

2. Σχετική έρευνα

Ο αλγόριθμος BLASTn αναπτύχθηκε στα τέλη της δεκαετίας του 1980 με αρχές τις δεκαετίας του 1990 από τον οργανισμό NCBI [11]. Τα αρχικά του σημαίνουν Basic Local Alignment Search Tool. Σαν πρώτη ιδέα ο αλγόριθμος BLASTn είναι μια ευριστική μέθοδος αναζήτησης λέξεων μεγέθους τουλάχιστον W χαρακτήρων με score το λιγότερο T σε μία βάση δεδομένων, όπως αναφέρεται στο [11]. Αφού έχουμε να κάνουμε με ακολουθίες DNA, οι πιθανοί χαρακτήρες που μπορούμε να έχουμε είναι αδενίνη, θυμίνη, γουανίνη, κυτοσίνη. Το μέγεθος W είναι ο ελάχιστος αριθμός χαρακτήρων που αν ταυτίζονται με τη ροή της βάσης δεδομένων, θεωρούμε πως έχουμε ευστοχία. Το T είναι το ελάχιστο score μιας ευστοχίας.

Πιο συγκεκριμένα, έχουμε μία ακολουθία χαρακτήρων DNA, το λεγόμενο query και μία βάση δεδομένων από χαρακτήρες DNA. Στόχος του αλγορίθμου είναι να διερευνήσει την ύπαρξη του query στη βάση δεδομένων, αυτούσιου ή στο περίπου και να βαθμολογήσει κάθε εύρημα. Για να γίνει αυτό το query αναλύεται σε λέξεις προκαθορισμένου μεγέθους W και με βάση αυτές γίνεται η αναζήτηση στη βάση δεδομένων. Το score T είναι η βαθμολογία μιας ευστοχίας μήκους W.

Εκτός από τον BLASTn υπάρχουν και άλλες παραλλαγές του BLAST που φαίνονται συγκεντρωτικά στον παρακάτω πίνακα. Οι διαφορές βρίσκονται στον τρόπο ερμηνείας του query, τον τρόπο ερμηνείας της βάσης δεδομένων καθώς και τη μέθοδο βαθμολόγησης.

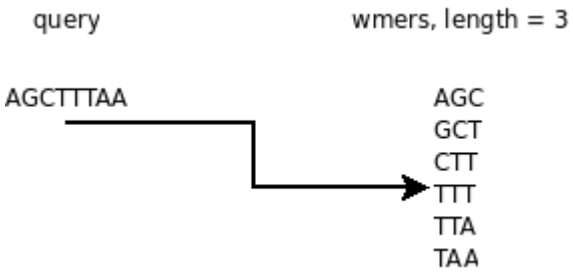
Πίνακας 1. Βασικές παραλλαγές του BLAST

Έκδοση αλγορίθμου	*Είσοδοι*	
	Query	*Database*
BLASTn	Νουκλεοτίδιο	Νουκλεοτίδιο
BLASTp	Αμινοξέα	Αμινοξέα
BLASTx	Μεταφρασμένη ακολουθία νουκλεοτιδίων	Αμινοξέα
TBLASTn	Αμινοξέα	Μεταφρασμένη ακολουθία νουκλεοτιδίων
TBLASTx	Μεταφρασμένη ακολουθία νουκλεοτιδίων	Μεταφρασμένη ακολουθία νουκλεοτιδίων

Ο αλγόριθμος αποτελείται από τρία βήματα. Το πρώτο είναι η δημιουργία των λέξεων μήκους W από το query. Το δεύτερο είναι η αναζήτηση λέξης μήκους W και το τρίτο η βαθμολόγηση της οποιασδήποτε ευστοχίας.

2.1 Τα τρία βήματα του BLASTn

Στο πρώτο βήμα έχουμε την αρχικοποίηση του συστήματος. Το query χωρίζεται σε υποακολουθίες που ονομάζονται wmers. Το μήκος της κάθε υποακολουθίας είναι σημαντική παράμετρος του αλγορίθμου καθώς ελέγχει την ευαισθησία του. Το τυπικό μέγεθος του κάθε wmer είναι έντεκα νουκλεοτίδια, ενώ στην πιο ευαίσθητη έκδοση του αλγορίθμου είναι πέντε. Υπάρχει και παραλλαγή με μεγαλύτερο wmer η MEGABLAST [12], τουλάχιστον δεκαέξι νουκλεοτίδια που είναι πιο γρήγορη και ταιριάζει καλύτερα σε μεγαλύτερα queries. Ο τρόπος με τον οποίο παράγουμε από το query wmers, φαίνεται στο Σχήμα 1.2. Λόγου χάρη αν έχουμε query ACTTATTGC και wmer size 3 τα wmers που προκύπτουν είναι (κινούμαστε από αριστερά προς τα δεξιά) : ACT, CTT, TTA, TAT, ATT, TTG και TGC.



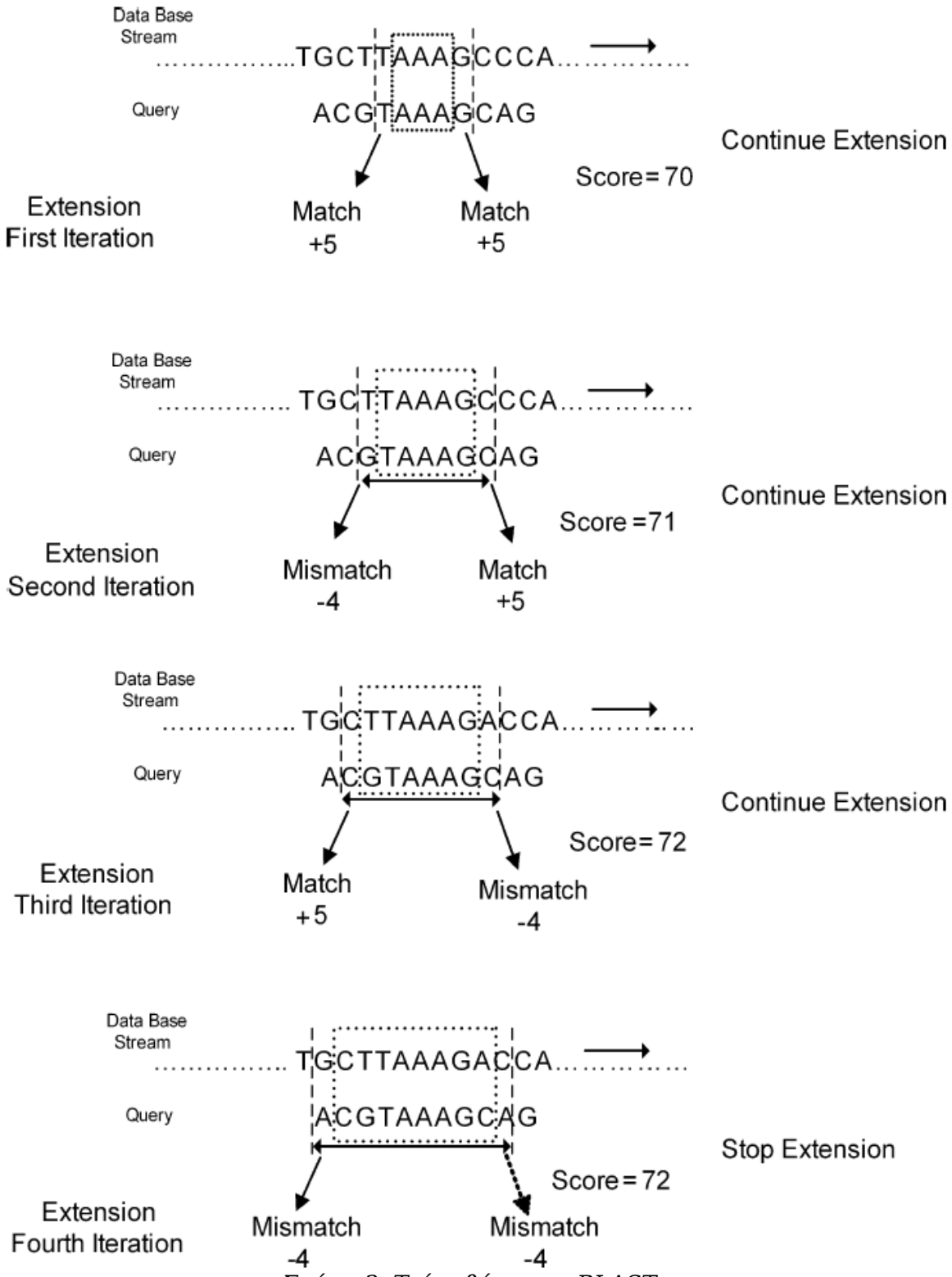
Σχήμα 1. Πρώτο βήμα του BLASTn

Το δεύτερο βήμα του αλγορίθμου είναι ο έλεγχος της βάσης δεδομένων για ευστοχίες. Έχουμε δηλαδή ένα ή παραπάνω «παράθυρα» εύρους όσο ένα wmer, στη ροή εισόδου της βάσης δεδομένων τα οποία ελέγχουμε αν ταυτίζονται με κάποιο wmer. Αν συμβαίνει κάτι τέτοιο τότε έχουμε ευστοχία και προχωράμε στο τρίτο βήμα του αλγορίθμου, που είναι ο υπολογισμός της αξίας (score) της ευστοχίας.

Σχήμα 2. Δεύτερο βήμα του BLASTn

Στο τρίτο βήμα του αλγορίθμου βαθμολογήται η ευστοχία χρησιμοποιώντας μία απλή μέθοδο. Γίνεται επέκταση εκατέρωθεν της ευστοχίας

κατά ένα χαρακτήρα τη φορά και ελέγχεται κατά πόσο ταυτίζεται με τον αντίστοιχο της ροής εισόδου. Για κάθε ταύτιση το score αυξάνει κατά πέντε ενώ για κάθε χαρακτήρα που διαφέρει μειώνεται κατά τέσσερα. Η επέκταση συνεχίζεται για όσο το score αυξάνει. Αυτό ουσιαστικά σημαίνει ότι η ευστοχία επεκτείνεται όσο τουλάχιστον ένας από τους δύο χαρακτήρες στα άκρα της είναι σωστός.



Σχήμα 3. Τρίτο βήμα του BLASTn

2.2 Έρευνα για τα εργαλεία

Ο στόχος αυτής της εργασίας είναι η σχεδίαση ενός chip για τον BLASTn. Δύο είναι οι τρόποι που μπορεί να γίνει κάτι τέτοιο, ο πρώτος είναι να σχεδιαστεί το ολοκληρωμένο με το χέρι και ο δεύτερος να γίνει χρήση εργαλείων Electronic Design Automatation (EDA) [15]. Βέβαια μπορεί να γίνει και συνδυασμός τους.

Η σχεδίαση ολοκληρωμένων στο χέρι προσφέρει στον μηχανικό τον ολοκληρωτικό έλεγχο επί του τελικού αποτελέσματος. Δυστυχώς όμως είναι μία προσέγγιση που περιορίζεται από το μέγεθος του προς σχεδίαση κυκλώματος, καθώς είναι πρακτικά αδύνατο να σχεδιαστούν αποτελεσματικά με το χέρι και εντός πιεστικών χρονικών ορίων, κυκλώματα μεγαλύτερα από μερικές εκατοντάδες ή χιλιάδες τρανζίστορ.

Η εναλλακτική λύση λοιπόν είναι τα εργαλεία EDA. Το μειονέκτημα τους είναι ότι παύει ο μηχανικός να έχει τον απόλυτο έλεγχο του αποτελέσματος και συνεπώς τίθεται ένα πιο ορατό άνω όριο στην απόδοση, από τις δυνατότητες του εκάστοτε εργαλείου. Τα EDA όμως είναι το μοναδικό ουσιαστικά μέσο που διαθέτουμε για την παραγωγή μεγάλων και πολύπλοκων κυκλωμάτων σε λογικά χρονικά πλαίσια και μάλιστα με εντυπωσιακή απόδοση. Πάλι βέβαια η παραδοσιακή μέθοδος μπορεί να χρησιμοποιηθεί για την σχεδίαση ιδιαίτερα απαιτητικών τμημάτων ενός ολοκληρωμένου, τον τελικό λόγο όμως στη σύγχρονη εποχή τον έχουν τα EDA.

2.2.1 Ιστορική αναδρομή

Πριν από τα EDA η σχεδίαση των κυκλωμάτων γινόταν εξ ολοκλήρου με το χέρι, εκτός από σπάνιες περιπτώσεις που χρησιμοποιούνταν ειδικό λογισμικό για τον σχεδιασμό ταινιών για τον Gerber photoplotter [23]. Πάλι όμως η διαδικασία βασιζόταν σε μηχανικά σχεδιασμένα τμήματα. Κυρίαρχος εκείνη την εποχή στο χώρο ήταν η εταιρία Calma [17], η οποία εισήγαγε το GDSII [16], που μεσουρανεί μέχρι σήμερα. Το GDSII είναι ένα δυαδικό αρχείο που περιέχει πληροφορία για γεωμετρικά χαρακτηριστικά, ετικέτες κειμένου κ.α, ικανό να αναπαράγει πλήρως ή εν μέρει ένα κύκλωμα. Το αρχείο αυτό είναι ο δημοφιλέστερος τρόπος στη βιομηχανία για την ανταλλαγή μασκών κυκλωμάτων.

Περί τα μέσα της δεκαετίας του '70, η ερευνητές άρχισαν να αυτοματοποιούν την διαδικασία της ίδιας της σχεδίασης και όχι απλά την δημιουργία γραφικών αναπαραστάσεων. Το 1980 αποτελεί χρονιά σταθμό, καθώς

δημοσιεύεται ένα από τα πλέον αναγνωρισμένα συγγράμματα, το «Introduction to VLSI systems» [1], των Carver Mead [13,14] και Lynn Conway [18]. Το βιβλίο αυτό εισήγαγε την ιδέα των γλωσσών προγραμματισμού οι οποίες συντάσσουν κυκλώματα. Αυτόματα λοιπόν έγινε δυνατή η αύξηση της πολυπλοκότητας των chip και ταυτόχρονα διευκολύνθηκε η σχεδίαση και επαλήθευση, καθώς δημιουργήθηκαν προσομοιωτές. Σχεδόν αμέσως έκαναν την εμφάνισή τους και τα πρώτα EDA αρχικά ακαδημαϊκής προέλευσης. Την ίδια εποχή ιδρύεται η MOSIS, ένας πάροχος υπηρεσιών κατασκευής κυκλωμάτων, μειωμένου κόστους, που απευθύνεται μεταξύ άλλων σε ακαδημαϊκά και ερευνητικά ιδρύματα.

Το 1981 είναι η χρονιά που τα EDA έγιναν βιομηχανικό προϊόν. Οι μεγάλες εταιρίες ημιαγωγών ήδη επεδίωκαν την δημιουργία τέτοιων εργαλείων εσωτερικά. Το 1986 η εταιρία Gateway δημιουργεί την Verilog, μία από τις πιο διαδεδομένες γλώσσες περιγραφής υλικού υψηλού επιπέδου και μία χρονιά αργότερα εμφανίζεται η VHDL χρηματοδοτούμενη από το αμερικανικό υπουργείο άμυνας. Λίγα χρόνια αργότερα εμφανίστηκαν προσομοιωτές και λογισμικό φυσικής σύνθεσης για τις δύο αυτές γλώσσες.

Σήμερα τα EDA κυριαρχούν στη βιομηχανία ημιαγωγών. Οι εταιρίες που μονοπωλούν στο χώρο είναι η Synopsys, η Cadence, η Magma και η Mentor Graphics [19-22]. Τα εργαλεία που χρησιμοποιήθηκαν σε αυτήν την εργασία είναι προϊόντα των δύο πρώτων εταιριών.

3. Αρχιτεκτονική του BLASTn για VLSI

Ένα από τα σημαντικότερα βήματα αυτής της εργασίας ήταν η ανάπτυξη ενός datapath ικανού να καλύψει τη λειτουργικότητα του αλγορίθμου και ταυτόχρονα να εκμεταλλευτεί τα πλεονεκτήματα και να αποφύγει τα μειονεκτήματα μιας υλοποίησης σε VLSI.

Γενικά ο αλγόριθμος θα μπορούσε να υλοποιηθεί με τρεις βασικές προσεγγίσεις. Η πρώτη που είναι η ευκολότερη και με τη μεγαλύτερη ευελιξία είναι σε επίπεδο λογισμικού. Όμως είναι η πιο αργή λύση, ειδικά την στιγμή που ο εν λόγω αλγόριθμος είναι υπολογιστικά ακριβός. Η δεύτερη προσέγγιση είναι με χρήση αναδιατασσόμενων πόρων σε συνεργασία ή όχι με λογισμικό. Αυτή η ιδέα έχει υιοθετηθεί ήδη από το Πολυτεχνείο Κρήτης με μεγάλη επιτυχία. Είναι μία λύση πολύ καλύτερη από πλευράς ταχύτητας, με αντάλλαγμα κάποια ευελιξία. Στην εργασία αυτή εξερευνάται η τρίτη προσέγγιση που είναι η σχεδίαση ενός chip, αποκλειστικά για τον BLASTn. Είναι η λύση που αναμένεται να είναι η πιο γρήγορη, χωρίς όμως την παραμικρή ευελιξία. Άπαξ και δημιουργηθεί ένα ολοκληρωμένο, δεν μπορεί να γίνει καμία παρέμβαση στη λειτουργικότητα του. Αλλαγές είναι εφικτές μόνο όσο το σχέδιο είναι σε προκατασκευαστικό στάδιο. Αυτός είναι και ο λόγος που πρέπει να δοθεί τόση σημασία στην σχεδίαση, διότι δεν υπάρχει καμία δυνατότητα μετέπειτα επέμβασης. Αν συνυπολογιστεί το ότι η δημιουργία των μασκών για ένα chip είναι υπόθεση πολλών χιλιάδων ευρώ, γίνεται αμέσως αντιληπτό ότι τα λάθη κοστίζουν ακριβά.

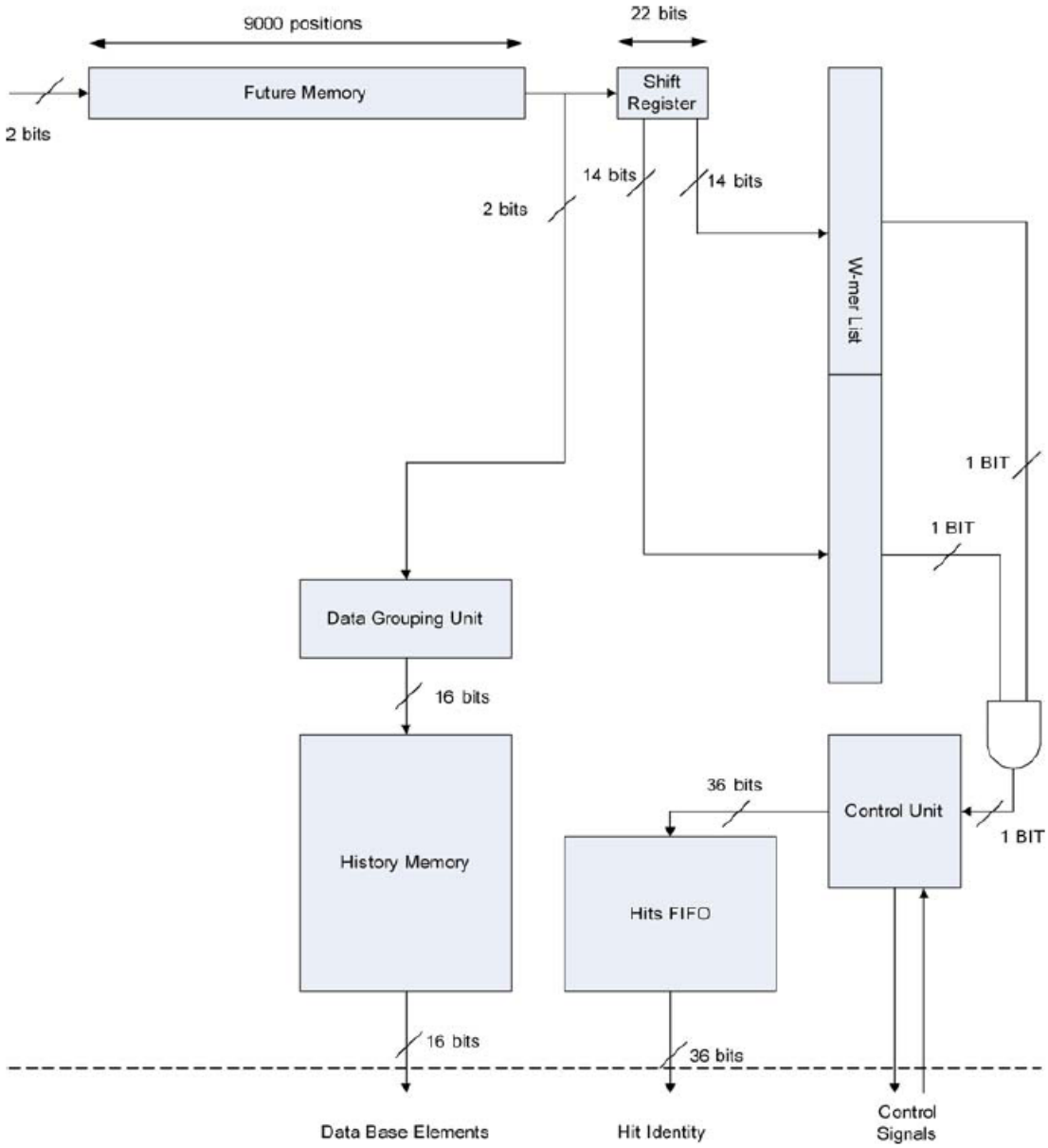
Στο κεφάλαιο αυτό παρουσιάζεται αρχικά η αρχιτεκτονική του Πολυτεχνείου Κρήτης [10] για Field Programmable Gate Arrays (FPGA) συσκευές της Xilinx και ύστερα η αντίστοιχη λύση για VLSI.

3.1 Αρχιτεκτονική για FPGA

Το συγκεκριμένο datapath παρουσιάζεται συνοπτικά σε αυτήν την εργασία διότι αποτελεί την σχεδιαστική αφετηρία του datapath για VLSI. Πρόκειται για μία αρχιτεκτονική που προορίζεται για απεικόνιση σε αναδιατασσόμενους πόρους ως ένα βαθμό και ύστερα σε λογισμικό.

Αυτή η επιλογή έγινε μετά από προσεκτική μελέτη του αλγορίθμου και των επιμέρους βημάτων που τον αποτελούν και αναλύθηκαν παραπάνω. Πριν κατασταλάξει το Πολυτεχνείο σε αυτήν την αρχιτεκτονική παρουσιάστηκαν και άλλες λύσεις χωρίς τη χρήση λογισμικού, όλες όμως η καθεμία για διαφορετικούς λόγους παρουσίαζαν παρόμοια απόδοση.

Σαν γενική ιδέα για την αποτελεσματική χρήση συσκευών FPGA, γίνεται profiling του οποιουδήποτε αλγορίθμου που στοχεύει σε μία συσκευή και εμφανίζονται τα πιο απαιτητικά κομμάτια της σχεδίασης. Εφόσον υπάρχουν περιορισμένοι πόροι FPGA, τα υπολογιστικά ακριβά τμήματα είναι τα υποψήφια για υλοποίηση στην FPGA. Τα υπόλοιπα υλοποιούνται λογισμικό και γίνεται χρήση των επεξεργαστών της εκάστοτε πλακέτας. Το τμήμα του αλγορίθμου που αποφασίστηκε να τρέχει σε λογισμικό είναι το τρίτο βήμα. Όπως αναφέρθηκε σε προγενέστερο κομμάτι της εργασίας, το βήμα αυτό εκτελείται μόνο όταν το δεύτερο βήμα εντοπίσει ευστοχία και στατιστικά οι ευστοχίες συμβαίνουν σπάνια. Το γεγονός αυτό οδήγησε σε σκέψεις για εκμετάλλευση των προσφερόμενων επεξεργαστών της συσκευής στόχου, ούτως ώστε να απελευθερωθούν αναδιατασσόμενοι πόροι και να χρησιμοποιηθούν πιο αποτελεσματικά. Το τελικό datapath φαίνεται στο Σχήμα 3.1.



Σχήμα 4. Αρχιτεκτονική Πολυτεχνείου Κρήτης για τον BLAST

Η παραπάνω αρχιτεκτονική είναι το υποσύνολο της ολόκληρης, η οποία υλοποιεί όλες τις παραλλαγές του BLAST και δείχνει μόνο τον τρόπο που εκτελείται ο BLASTn, το τμήμα που σχετίζεται με αυτήν την εργασία.

Το πρώτο βήμα του αλγορίθμου αποτελεί ουσιαστικά την αρχικοποίηση της παραπάνω διάταξης. Το δεύτερο βήμα (αναζήτηση ευστοχίας μεγέθους W) επιλέχθηκε να υλοποιηθεί με χρήση look up tables. Αυτό σημαίνει πως μία μνήμη διευθυνσιοδοτήται με το υποψήφιο wmer, και επιστρέφει λογικό 1 αν είναι wmer, αλλιώς 0. Στην παραπάνω διάταξη η ποσότητα που διευθυνσιοδοτεί τις δύο μνήμες είναι αυτή που περιέχεται στον shift register των 22bit(αφού το τυπικό μέγεθος wmer για τον BLASTn είναι 11 χαρακτήρες και έχουμε αλφάβητο τεσσάρων). Ο λόγος που υπάρχουν δύο μνήμες είναι ότι δεν υπήρχε στην πλακέτα αρκετά μεγάλη μνήμη για να χωρέσει όλη την πληροφορία. Χρειάζεται μνήμη 2^{11} λέξεων του ενός bit, για την πλήρη κάλυψη όλων των περιπτώσεων. Το τρίτο βήμα εκτελείται σε λογισμικό.

Όπως φαίνεται και στο block diagramm τα κύρια χαρακτηριστικά του datapath είναι μνήμες, καταχωρητές ολίσθησης, fifo, μηχανές πεπερασμένων καταστάσεων και απλή λογική. Σε κάθε κύκλο το κύκλωμα εξετάζει για ευστοχία μία εντεκάδα χαρακτήρων. Όλες οι ευστοχίες μπαίνουν σε μια fifo για την εκτέλεση του τρίτου βήματος στον PowerPC της πλακέτας.

3.2 Αρχιτεκτονική για VLSI

Το datapath για VLSI σχεδιάστηκε δανειζόμενο αρκετά χαρακτηριστικά από την παραπάνω αρχιτεκτονική. Έχει όμως κάποιες διαφορές οι οποίες προκύπτουν από περιορισμούς που θέτει η τεχνολογία υλοποίησης. Η αρχιτεκτονική για VLSI βασικά περιλαμβάνει και το τρίτο βήμα του αλγορίθμου σε υλικό καθώς έχουμε να κάνουμε με ένα αυτόνομο chip. Τα συστατικά του κυκλώματος γενικά είναι πολύ απλές δομές όπως καταχωρητές ολίσθησης, μνήμες RAM και μηχανές πεπερασμένων καταστάσεων, καθώς έγινε προσπάθεια να είναι όσο πιο απλή και σίγουρη γίνεται η σχεδίαση. Το datapath υποστηρίζει queries μέχρι και 10000 χαρακτήρων, κάτι που μεταφράζεται σε 20000 bits, αν λάβουμε υπόψη την 2 bit αναπαράσταση κάθε χαρακτήρα, αφού έχουμε τέσσερις πιθανούς.

Το πρώτο βήμα του αλγορίθμου είναι μέρος της αρχικοποίησης του κυκλώματος και σε αυτή την περίπτωση. Το query κρατείται σε έναν barrel shifter των 20000 bit, που γεμίζει με χαρακτήρες από αριστερά προς τα δεξιά και έναν-έναν. Βεβαίως υπάρχει και ο απαραίτητος έλεγχος για τυχόν εισαγωγή παραπάνω χαρακτήρων οπότε και το κύκλωμα σταματά. Κατά την εισαγωγή των

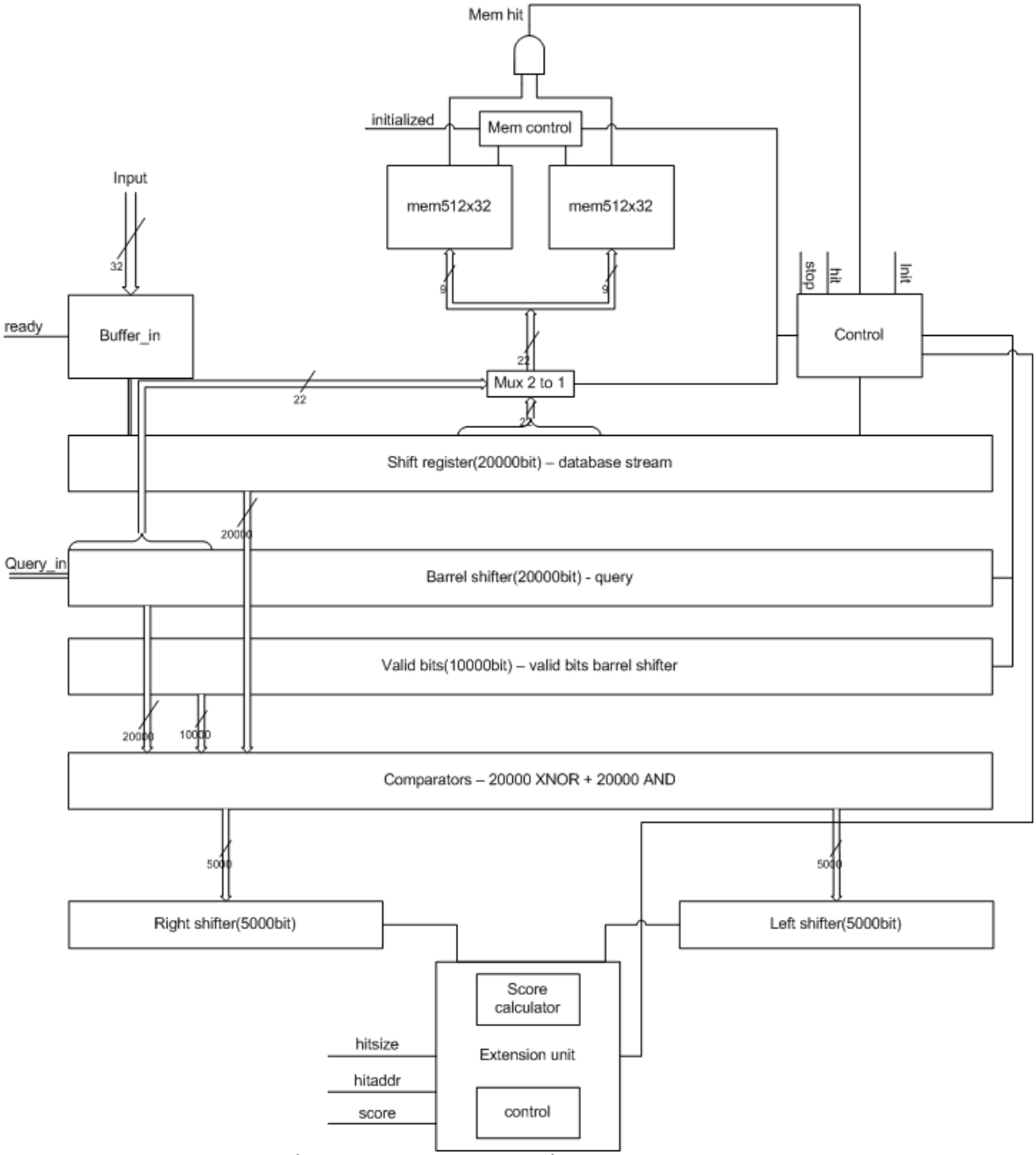
χαρακτήρων τα πρώτα 22 bit του barrel shifter από αριστερά προς τα δεξιά, που αποτελούν ουσιαστικά τα wmers, διευθυνσιοδοτούν μερικώς τις μνήμες, οι οποίες βρίσκονται σε λειτουργία εγγραφής. Η εγγραφή κρατά δύο κύκλους, καθώς στον πρώτο πρέπει να διαβαστεί η υπάρχουσα λέξη στην θέση μνήμης και στον δεύτερο να γίνει το αντίστοιχο bit της λέξης ένα. Αυτός είναι ο τρόπος που αρχικοποιούνται οι μνήμες σύμφωνα με τα wmers.

Το δεύτερο βήμα του αλγορίθμου υλοποιήται σχεδόν ακριβώς όπως και στην παραπάνω αρχιτεκτονική. Ιδανικά, για την αναζήτηση των hit των wmer, θα έπρεπε να υπάρχει μία μνήμη που να έχει 2^{11} λέξεις του ενός bit, ούτως ώστε να διευθυνσιοδοτήται με το πιθανό wmer και να δίνει αμέσως την απάντηση. Δυστυχώς όμως μία τέτοια μνήμη είναι πολύ μεγάλη και έχει δύσκολες αναλογίες κάτι που σημαίνει πως και να μπορούσε να δημιουργηθεί θα ήταν ασύμφορη από πλευράς χρόνου, ισχύος και ισομερούς χρήσης της περιόδου του ρολογιού. Συνεπώς για το δεύτερο βήμα επιλέχθηκαν δύο αρκετά μικρότερες μνήμες, η κάθε μία των 512 λέξεων των 32 bit. Αυτό βέβαια σημαίνει πως για να έχουμε πραγματικό hit, πρέπει η έξοδος και των δύο μνημών να είναι λογικό ένα. Η εντεκάδα των χαρακτήρων που διευθυνσιοδοτούν τις μνήμες αυτές, βρίσκεται στο κέντρο του shift register που περνά η ροή εισόδου.

Η εκτέλεση του τρίτου βήματος είναι η κύρια διαφορά με την αντίστοιχη αρχιτεκτονική FPGA. Υπάρχουν 10000 συγκριτές των δύο bit που συγκρίνουν κάθε χαρακτήρα του barrel shifter με τον αντίστοιχο του shift register που περνά η ροή εισόδου. Το αποτέλεσμα «φιλτράρεται» με ένα valid bit που αντιστοιχεί σε κάθε θέση του query και δείχνει το αν η πληροφορία που περιέχεται εκεί είναι χαρακτήρας ή κάτι άχρηστο. Έτσι δημιουργήται μία τελική πληροφορία 10000 bit που αποθηκεύεται σε δύο καταχωρητές ολίσθησης, έναν αριστερής και έναν δεξιάς, των 5000 bit έκαστος. Όταν στο δεύτερο βήμα εντοπιστεί ευστοχία και προχωρά η εκτέλεση στο τρίτο, οι δύο αυτοί καταχωρητές αρχίζουν να κάνουν ολίσθηση και εξετάζεται κάθε φορά το LSB αυτού με την δεξιά ολίσθηση και το MSB αυτού με την αριστερή. Αν και τα δύο είναι μηδέν τότε σταματά η επέκταση, διαφορετικά συνεχίζει και προστίθεται κάθε φορά στο υπάρχον score η ανάλογη τιμή. Αυτό το βήμα δεδομένου ότι υποστηρίζονται queries μέχρι 10000 χαρακτήρες μπορεί να κρατήσει έως και 5000 κύκλους ρολογιού.

Όλοι οι καταχωρητές που χρησιμοποιούνται έχουν ασύγχρονο reset το οποίο είναι active low. Για την αρχικοποίηση του κυκλώματος γίνονται δύο ενέργειες· μηδενίζονται όλοι οι καταχωρητές και αρχικοποιούνται οι μνήμες. Για την πρώτη χρειαζόμαστε ένα κύκλο, ενώ για την δεύτερη 512, αφού έχουμε μνήμες 512 λέξεων των 32 bit. Συνεπώς από τη στιγμή που θα γίνει χρήση του reset, το κύκλωμα θα είναι διαθέσιμο μετά από 512 κύκλους. Ακολουθεί το block diagramm μαζί με ένα βοηθητικό διάγραμμα του control για την αποσαφήνιση

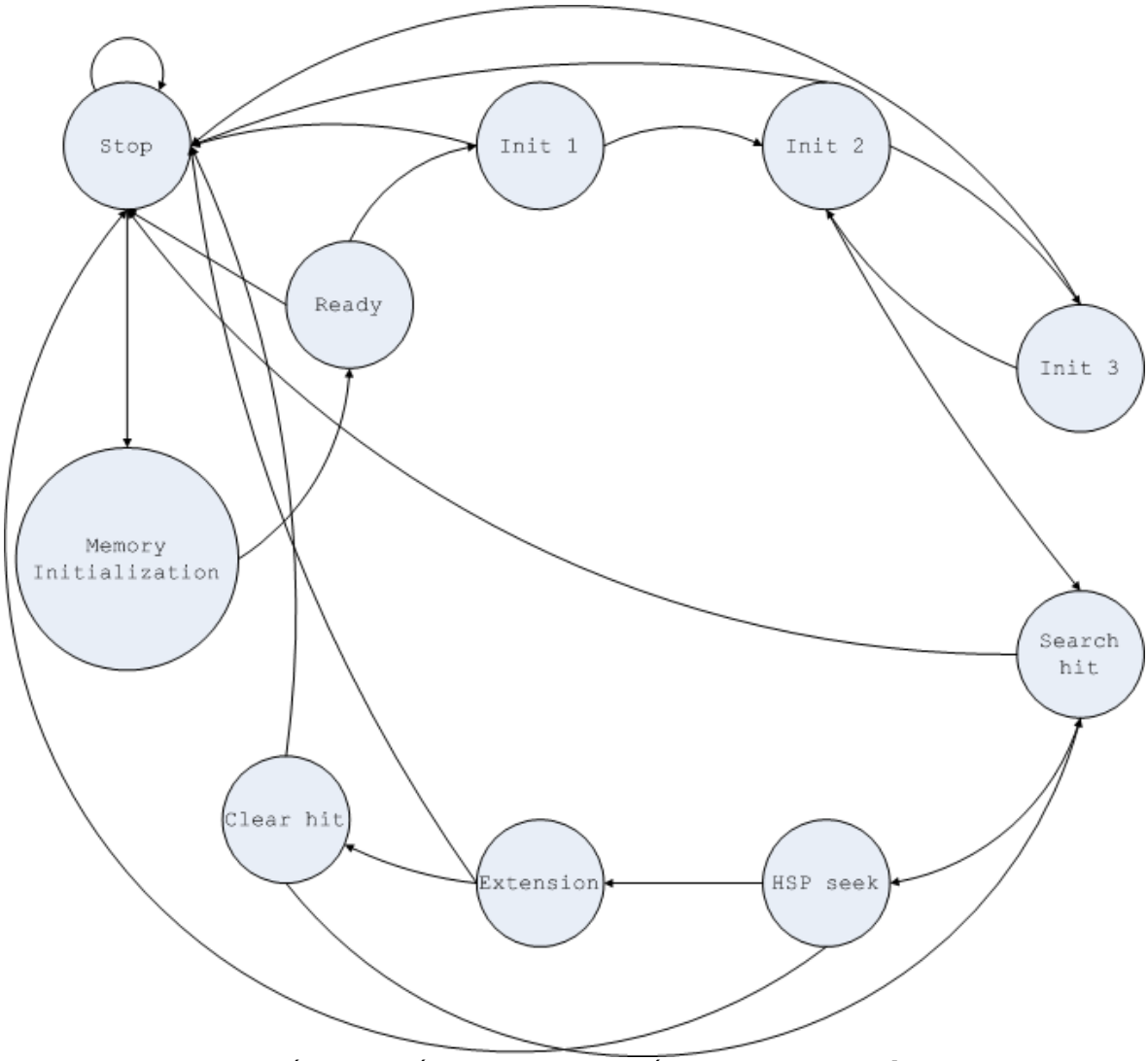
της λειτουργίας του datapath.



Σχήμα 5. Αρχιτεκτονική BLASTn για VLSI

Στο παραπάνω διάγραμμα φαίνονται οι δομικές μονάδες που απαρτίζουν το κύκλωμα. Επίσης φαίνονται οι κύριες συνδέσεις μεταξύ τους και σημειώνεται το εύρος των buses. Επίσης οι κύριοι έξοδοι του κυκλώματος αναγράφονται κοντά στις μονάδες που προέρχονται. Το interface του κυκλώματος παρουσιάζεται βεβαία αναλυτικά στην επόμενη ενότητα.

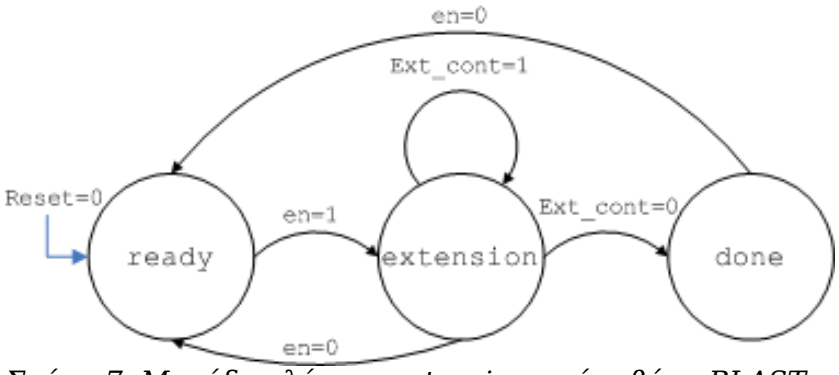
Για να δοθεί η πλήρη εικόνα της λειτουργικότητας ακολουθεί το state diagramm της μηχανής πεπερασμένων καταστάσεων του control. Σε αυτό υπάρχουν όλες οι πιθανές καταστάσεις του control και σημειώνονται όλες οι μεταβάσεις.



Σχήμα 6. Διάγραμμα καταστάσεων του control

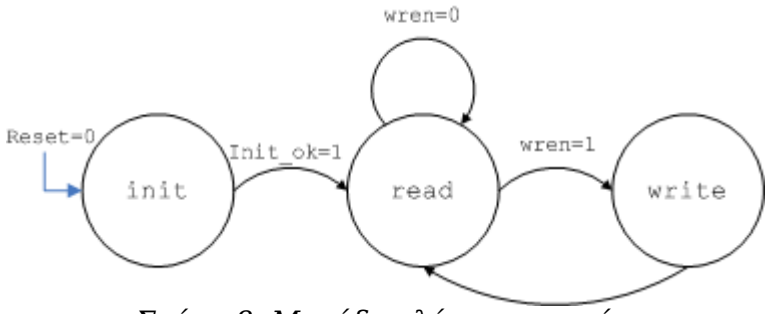
Οι πιθανές καταστάσεις είναι δέκα. Σε πρώτη φάση αρχικοποιείται η μνήμη με μηδενικά. Στην επόμενη κατάσταση (ready) μόλις έρθει το κατάλληλο σήμα ξεκινά το πρώτο βήμα του αλγορίθμου το οποίο πραγματοποιείται στις Init 1,2,3. Μόλις ολοκληρωθεί, ξεκινά να εκτελείται το δεύτερο βήμα στην Search hit. Αν βρεθεί ευστοχία ευθυγραμμίζεται ο barrel shifter του query με την ροή εισόδου και εκτελείται το τρίτο βήμα του αλγορίθμου στην κατάσταση Extension. Μόλις σταματήσει η επέκταση και καθαριστούν οι καταχωρητές συνεχίζει η εκτέλεση του δεύτερου βήματος. Όπως φαίνεται σχεδόν σε κάθε σημείο της εκτέλεσης μπορεί να διακοπεί η διαδικασία αν ο χρήστης το θέλει, χρησιμοποιώντας το σήμα stop, οπότε το κύκλωμα μεταβαίνει στην αντίστοιχη κατάσταση από την οποία βγαίνει με χρήση του reset. Η κωδικοποίηση που επιλέχθηκε για το control είναι one hot, ενώ οι fsm των μονάδων μνήμης και επέκτασης ακολουθούν κωδικοποίηση gray. Η επιλογή αυτή έγινε ύστερα από δοκιμές που έδειξαν ποια ήταν η καλύτερη για κάθε περίπτωση. Ακολουθούν και

τα διαγράμματα καταστάσεων των fsm της μονάδας μνήμης και επέκτασης(τρίτο βήμα αλγορίθμου).



Σχήμα 7. Μονάδα ελέγχου extension, τρίτο βήμα BLASTn

Όπως φαίνεται και στο σχήμα η μονάδα οδηγείται στην κατάσταση ready με το σήμα reset ή αν είναι σε κάποια άλλη κατάσταση και με την πτώση του enable. Το extension λαμβάνει χώρα στην ομώνυμη κατάσταση, ενώ όταν ολοκληρωθεί, το σύστημα μεταβαίνει στην κατάσταση done, μέχρι την πτώση του enable ή του reset. Δίνεται και το αντίστοιχο διάγραμμα για την μονάδα ελέγχου των μνημών.



Σχήμα 8. Μονάδα ελέγχου μνημών

Αρχικά πρέπει οι μνήμες να αρχικοποιηθούν σε μηδενικά. Η εργασία αυτή γίνεται στην κατάσταση init και διαρκεί 512 κύκλους, όσες είναι δηλαδή και οι λέξεις στη μνήμη. Κατά τη διάρκεια αυτών των κύκλων ένας αύξων μετρητής των 9 bit, διευθυνσιοδοτεί τις μνήμες, οι οποίες λειτουργούν σε κατάσταση εγγραφής, με δεδομένα 32 μηδενικά. Οποτεδήποτε πέσει το reset, η μονάδα επιστρέφει σε αυτήν την κατάσταση.

Οι αναγνώσεις γίνονται στην κατάσταση read. Οι εγγραφές γίνονται σε δύο φάσεις. Επειδή σύμφωνα με την αρχιτεκτονική του BLASTn για VLSI, είναι επιθυμητή η εγγραφή σε επίπεδο bit, διαβάζεται σε έναν κύκλο η λέξη που περιέχει το προς αλλαγή bit, το bit αυτό αλλάζει και στον επόμενο κύκλο γίνεται η εγγραφη της αλλαγμένης λέξης. Το σήμα wren των μνημών είναι active low, προς αποφυγή όμως σύγχυσης στην εικόνα, ισχύει το αντίθετο.

3.3 Διεπαφή του κυκλώματος

Στον παρακάτω πίνακα συνοψίζεται η διεπαφή του datapath. Ακολουθεί τα γενικά πρότυπα της αντίστοιχης αρχιτεκτονικής για FPGA.

Πίνακας 2. Διεπαφή αρχιτεκτονικής για τον BLASTn

Όνομα	*Είδος*	*Μέγεθος*	*Λειτουργία*
clk	in	1-bit	ρολόι
reset	in	1-bit	Ασύγχρονος μηδενισμός, active low
init	in	1-bit	'1' όταν ο χρήστης εισάγει το query
query_in	in	2-bit	Ο επόμενος χαρακτήρας του query(βήμα 1)
input	in	32-bit	Είσοδος χαρακτήρων στη ροή της βάσης δεδομένων
stop	in	1-bit	Σήμα για την παύση του κυκλώματος
hit	out	1-bit	'1' όταν το κύκλωμα έχει εντοπίσει ευστοχία
ready	out	1-bit	Στη θετική ακμή αυτού του σήματος ανανεώνουμε την είσοδο input
initialized	out	1-bit	'1' όταν η μνήμη είναι αρχικοποιημένη
score	out	16-bit	Το score της ευστοχίας
hitsize	out	16-bit	Το μέγεθος της ευστοχίας
hitaddr	out	32-bit	Πόσοι χαρακτήρες έχουν περάσει από την αρχή και μέχρι τον πρώτο χαρακτήρα της ευστοχίας

Το interface του datapath έχει τυπικά χαρακτηριστικά. Τα buses των εισόδων – εξόδων είναι όλα δυνάμεις του δύο(2-bit,16-bit, 32-bit) και το reset

ασύγχρονο και active low. Οι τελευταίες τρεις έξοδοι περιέχουν χρήσιμη πληροφορία μόνο όταν το σήμα hit είναι ενεργό.

3.4 Πρόχειρη εκτίμηση κατανάλωσης πόρων για τον BLASTn

Στον παρακάτω πίνακα φαίνεται η μεγεθοποίηση της αρχιτεκτονικής για τον BLASTn.

Πίνακας 3. Μεγεθοποίηση της αρχιτεκτονικής του BLASTn

Μονάδα	*Καταχωρητές*	*Λογικές πύλες*
Barrel shifter	20000	<20
Shift register(input stream)	20000	<20
Valid bits register	10000	<20
Control	<50	<50
Extension unit	<50	<50
Left shifter	5000	<20
Right shifter	5000	<20
Comparators	-	40000
Buffer in	32	<20
Memories unit	<20	<200
SRAMs	2 x 512 words x 32 bits	
I/O	37 είσοδοι και 68 έξοδοι	
Σύνολο	60000 καταχωρητές και 40000 πύλες	

Όπως φαίνεται και στον παραπάνω πίνακα το τελικό chip, αναμένεται να έχει ένα μέγεθος της τάξης των 100000 λογικών κυττάρων. Ακόμη ένας παράγοντας είναι οι διασυνδέσεις που απαιτούνται μεταξύ τους. Ο shift register της ροής εισόδου, ο valid bits register και ο barrel shifter του query επικοινωνούν με τη μονάδα comparators, η οποία δίνει το 10000 bit αποτέλεσμα της στους left και right shifters. Αυτό σημαίνει πως θα το σχέδιο θα είναι ιδιαίτερα απαιτητικό στο routing το οποίο μπορεί να θέσει ένα αυστηρό άνω όριο στην πυκνότητα.

4. Σχεδίαση σε VLSI

Σε αυτό το κεφάλαιο παρουσιάζεται η διαδικασία μεταφοράς της παραπάνω σχεδίασης σε ολοκληρωμένο, που είναι και το κύριο τμήμα της εργασίας.

Όπως αναφέρθηκε νωρίτερα, ένα ολοκληρωμένο μπορεί να παραχθεί είτε με το χέρι, είτε με χρήση EDA, είτε και με τα δύο. Στην παρούσα εργασία, ο δεύτερος τρόπος είναι αυτός που υιοθετήθηκε, η σχεδίαση έγινε εξ ολοκλήρου με χρήση EDA. Αυτό διότι αφενός ήταν αρκετά μεγάλη για να γίνει με το χέρι και αφετέρου οι τελευταίες γενιές εργαλείων EDA δίνουν αξιοθαύμαστα αποτελέσματα σε πολύ μεγάλες σχεδιάσεις, πόσο μάλλον στην συγκεκριμένη, η οποία θα μπορούσε να χαρακτηριστεί μικρή αν αναλογιστεί κανείς τις δυνατότητες των εργαλείων.

Συνοπτικά τρία είναι χονδρικά τα βήματα όλης της διαδικασίας, με το πρώτο να είναι το κύριο έργο του σχεδιαστή. Αρχικά το datapath περιγράφηκε με μια γλώσσα περιγραφής υλικού, συγκεκριμένα VHDL, ύστερα η περιγραφή αυτή απεικονίστηκε στις λογικές πύλες μίας τεχνολογίας κατασκευής και ύστερα έγινε η τοποθέτηση και διασύνδεση (place and route). Η VHDL επιλέχθηκε για την περιγραφή του datapath, λόγω οικειότητας και καλής γνώσης της.

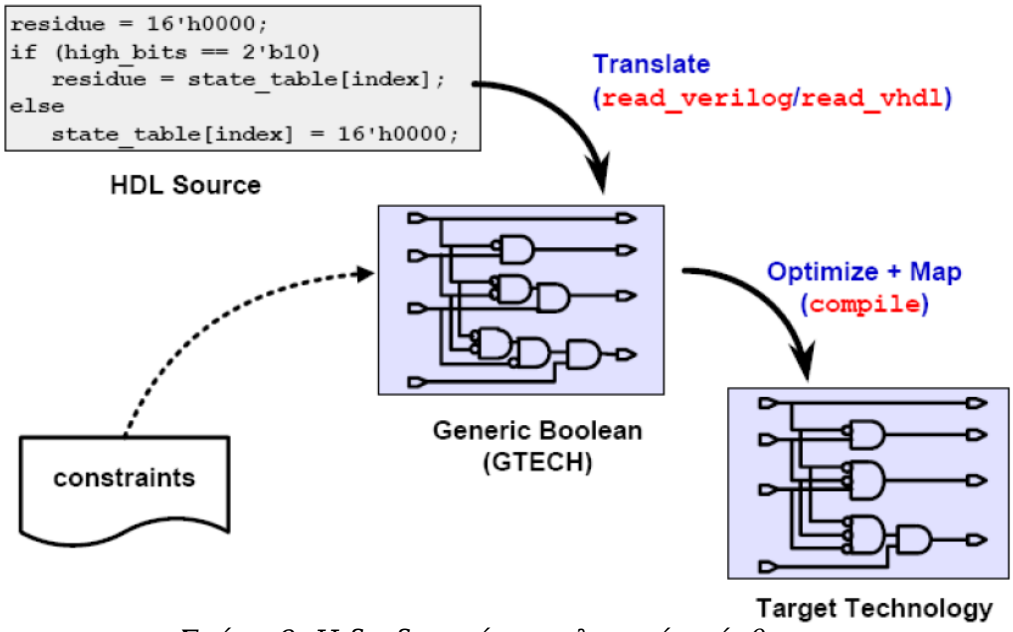
Τα εργαλεία που χρησιμοποιήθηκαν για αυτή τη ροή είναι προϊόντα της Synopsys και Cadence τα οποία διαθέτει το Πολυτεχνείο Κρήτης μέσω του προγράμματος Europractice.

4.1 Η λογική σύνθεση

Η λογική σύνθεση, είναι μια διαδικασία στην οποία RTL απεικονίζεται στα διαθέσιμα standard cells ή κύτταρα μιας δεδομένης τεχνολογίας. Τα κύτταρα συνήθως είναι απλές λογικές πύλες, σύνθετες πύλες, αθροιστές, flip flops κ.α.

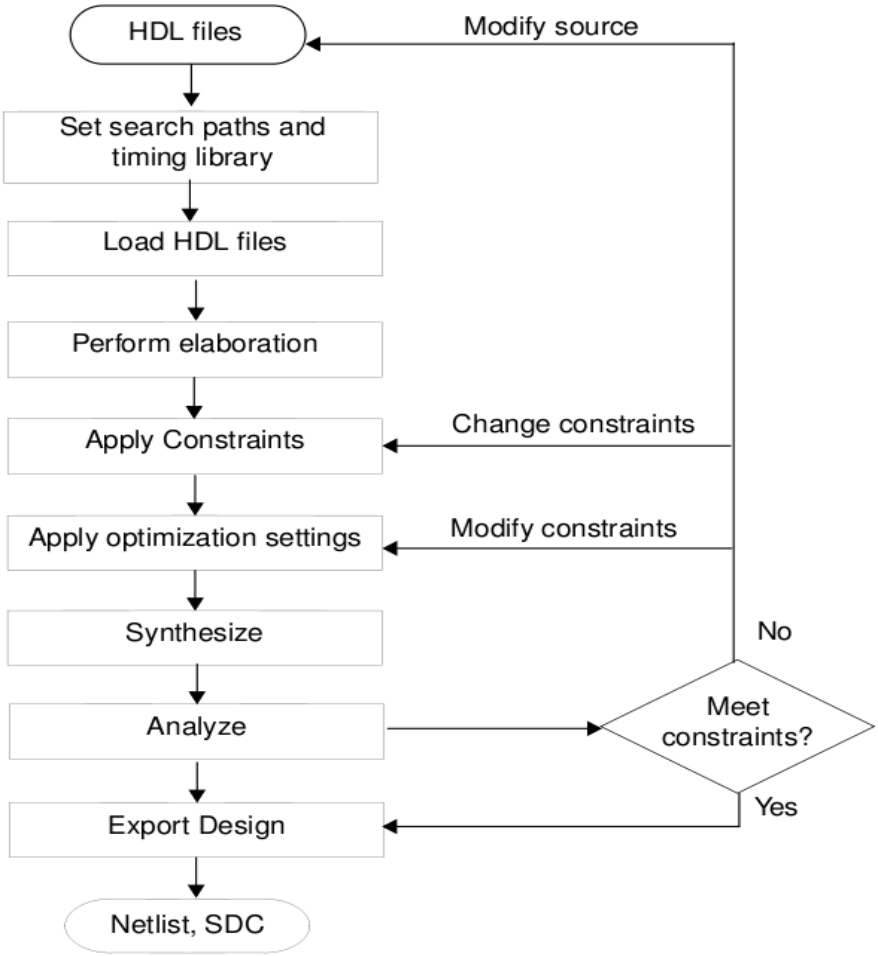
Αρχικά το εργαλείο που έχει επιλεγεί για να κάνει την σύνθεση διαβάζει την HDL (VHDL ή Verilog) που του δίνει ο σχεδιαστής και τη μεταφράζει σε generic πύλες, μετατρέπει τον κωδικά δηλαδή σε μία τυπική απεικόνιση. Αφού ο σχεδιαστής εισάγει περιορισμούς σχετικά με τον χρονισμό, το εμβαδό, την κατανάλωση και οποιαδήποτε άλλη παράμετρο κρίνει για το κύκλωμα, αρχίζει η διαδικασία της απεικόνισης στα κύτταρα της τεχνολογίας στόχου (mapping). Μόλις ολοκληρωθεί η διαδικασία ο μηχανικός έχει στα χέρια του το netlist και το αρχείο sdc που κυρίως χρειάζεται για να προχωρήσει στο επόμενο βήμα, την τοποθέτηση και διασύνδεση. Επίσης σε αυτό το σημείο είναι διαθέσιμη μια αρχική εκτίμηση του χρονισμού της κατανάλωσης και του εμβαδού της

σχεδίασης. Στο παρακάτω σχήμα φαίνεται διαισθητικά η διαδικασία της σύνθεσης [24].



Σχήμα 9. Η διαδικασία της λογικής σύνθεσης

Η γενική ροή για τη λογική σύνθεση φαίνεται παρακάτω.



Σχήμα 10. Ροή της φυσικής σύνθεσης

Τα εργαλεία που χρησιμοποιήθηκαν σε αυτή την εργασία για τη λογική σύνθεση είναι το Design Vision της Synopsys και το RTL Compiler της Cadence. Το πρώτο θεωρείται από τα πλέον αναγνωρισμένα στο χώρο και είναι το προϊόν για το οποίο η Synopsys είναι κυρίως γνωστή. Το δεύτερο είναι μια πολύ μεταγενέστερη εφαρμογή η οποία μαζί με το Build Gates είναι η απάντηση της Cadence στην κυριαρχία της Synopsys στο χώρο. Αρκετοί τελευταία υποστηρίζουν ότι τα εργαλεία της Cadence δίνουν καλύτερα αποτελέσματα, η βιομηχανία όμως δεν δείχνει να έχει πειστεί. Δεν ήταν αρχικά σχεδιασμένο να χρησιμοποιηθεί ο RTL Compiler, δεν γινόταν όμως αλλιώς, καθώς προέκυψε ζήτημα αδειών για το Design Vision. Η Cadence διαφημίζει τον RTL Compiler ως την βέλτιστη λύση για σύνθεση μεγάλων κυκλωμάτων διότι ενσωματώνει μία νέα προσέγγιση που η εταιρία ονομάζει global focus synthesis.

Σε αυτήν την εργασία χρησιμοποιήθηκαν κυρίως δύο τεχνολογίες υλοποίησης. Η πρώτη είναι μία της UMC στα 0,13μm και η δεύτερη της Faraday στα 90nm, η οποία αποκτήθηκε μέσω Europractice. Η δεύτερη τεχνολογία είναι και αυτή UMC ουσιαστικά, καθώς η Faraday είναι θυγατρική της UMC. Ο λόγος που χρησιμοποιήθηκαν δύο τεχνολογίες είναι οτί η πρώτη δυστυχώς παρουσίαζε σοβαρές ασυμβατότητες με το εργαλείο τοποθέτησης και διασύνδεσης, το encounter της Cadence. Ούτε όμως η δεύτερη ήταν χωρίς προβλήματα καθώς τα οι βιβλιοθήκες των pads παρουσίαζαν προβλήματα με το ίδιο εργαλείο.

4.1.1 Εισαγωγή αρχείων τεχνολογίας

Το πρώτο βήμα στη λογική σύνθεση είναι να καθοριστεί η τεχνολογία στόχος. Για να γίνει αυτό πρέπει να τεθούν οι αντίστοιχες μεταβλητές περιβάλλοντος του εργαλείου στα αρχεία χρονισμού της τεχνολογίας που έρχονται από το foundry για λογική σύνθεση. Στο στάδιο της σύνθεσης μόνο τα αρχεία χρονισμού χρειάζονται καθώς δεν χρησιμοποιείται άλλου είδους πληροφορία, εκτός ίσως από τις φυσικές διαστάσεις και την κατανάλωση κάθε κυττάρου. Η κάθε τεχνολογία έχει ξεχωριστές βιβλιοθήκες I/O και κυττάρων για τον πυρήνα του chip καθώς και ξεχωριστές βιβλιοθήκες χρονισμού των προκατασκευασμένων δομών όπως είναι οι μνήμες. Οι δομές αυτές πρέπει να ζητηθούν χωριστά.

Με κάθε τεχνολογία έρχονται συνήθως τρία αρχεία με στοιχεία χρονισμού των κυττάρων, τα οποία αντιστοιχούν σε τρεις ενδεικτικές καταστάσεις λειτουργίας. Κάθε δομή που υπάρχει μέσα σε οποιοδήποτε chip, χαρακτηρίζεται χρονικά και όχι μόνο στις χειρότερες, καλύτερες και τυπικές συνθήκες που αναμένεται να λειτουργεί. Αυτό μεταφράζεται σε τρεις διαφορετικές θερμοκρασίες και τάσεις λειτουργίας οι οποίες πιέζουν ή ευνοούν το προς

εξέταση κύκλωμα. Στις χειρότερες συνθήκες είναι οι μεγαλύτερες θερμοκρασίες με τις χαμηλότερες τάσεις και στις καλύτερες το αντίθετο.

Οι βιβλιοθήκες χρονισμού έρχονται συνήθως σε αρχεία liberty, με κατάληξη .lib δηλαδη, ή σε αρχεία με κατάληξη .tlf. Σχεδόν πάντα έρχονται και στην ειδική μορφή που απαιτούν οι synthesizers της Synopsys, τα αρχεία .db. Ανάλογα με το foundry της τεχνολογίας μπορεί να υπάρχουν και άλλες μορφές για άλλα εργαλεία.

Στην περίπτωση του Design Vision, η εισαγωγή μπορεί να γίνει είτε μέσω του γραφικού περιβάλλοντος είτε μέσω ενός ειδικού αρχείου με το όνομα .synopsys_dc.setup, που τοποθετήται στο home directory κάθε χρήστη ενός workstation με το εργαλείο εγκατεστημένο. Τρία αντίτυπα αυτού του αρχείου υπάρχουν· το πρώτο βρίσκεται μέσα στην εγκατάσταση του εργαλείου και έχει την υψηλότερη προτεραιότητα, δεν πανωγράφεται δηλαδή. Το δεύτερο είναι στο home directory κάθε χρήστη και το τρίτο είναι στο φάκελο που τρέχει το Design Vision και διαβάζεται τελευταίο. Για την εισαγωγή το μόνο που πρέπει να γίνει, είναι η αλλαγή των αντίστοιχων μεταβλητών, ώστε να δείχνουν στα αρχεία χρονισμού, στο δεύτερο ή τρίτο αντίτυπο. Τα αρχεία χρονισμού πρέπει να τεθούν ως link library και target library.

Για το RTL compiler δεν υπάρχει η δυνατότητα εισαγωγής μέσω γραφικού περιβάλλοντος. Πρέπει να τεθεί το αντίστοιχο attribute να δείχνει στα .lib αρχεία χρονισμού και να οριστεί το path με την τοποθεσία των αρχείων. Ένα ενδιαφέρον χαρακτηριστικό του RTL compiler είναι ότι μπορεί να κάνει και physical aware synthesis, δηλαδή να λάβει υπόψη και τα φυσικά στοιχεία των κυττάρων, αρκεί να τροφοδοτηθεί με τα ανάλογα αρχεία.

Σαν γενική ιδέα και τα δύο εργαλεία είναι σχεδιασμένα για εργασία με scripts. Αυτός είναι και ο πιο αποτελεσματικός τρόπος χρήσης τους, στην περίπτωση του RTL Compiler δε, ο μοναδικός καθώς το γραφικό του δεν προσφέρει σχεδόν καμία λειτουργικότητα.

4.1.2 Εισαγωγή αρχείων HDL

Αφού δοθούν στο εργαλείο τα αρχεία της τεχνολογίας, το αμέσως επόμενο βήμα είναι η εισαγωγή της HDL που προορίζεται για mapping. Αυτό επιτυγχάνεται με τις σχετικές εντολές που έχει κάθε εργαλείο για αυτήν την εργασία και δίνονται στο αντίστοιχο παράρτημα.

Στο Design Vision δύο είναι οι τρόποι για να γίνει αυτό το βήμα. Ο πρώτος είναι με την απλή εντολή που αναφέρεται παραπάνω και ο άλλος είναι να γίνει πρώτα analyze και ύστερα elaborate του κώδικα. Με τον πρώτο τρόπο διαβάζονται τα σχέδια χωρίς προσδιορισμό παραμέτρων και δημιουργείται

απευθείας ενα συμβολικό μοντέλο στην μνήμη του εργαλείου, ενώ η δεύτερη ακολουθία υπολογίζει τιμές παραμέτρων και προσθέτει επιπρόσθετες δομές, όπως φραγή ρολογιού. Στον RTL Compiler διαθέσιμος είναι μόνο ο δεύτερος τρόπος. Καλό είναι τα αρχεία να διαβαστούν με ιεραρχική σειρά.

4.1.3 Εισαγωγή περιορισμών

Μετά την εισαγωγή των αρχείων HDL της σχεδίασης ακολουθεί το πιο σημαντικό ίσως τμήμα της σύνθεσης, η εισαγωγή των περιορισμών. Από τους περιορισμούς αυτούς θα εξαρτηθεί η ποιότητα του αποτελέσματος της σύνθεσης. Σε κάθε περίπτωση στόχος είναι η δημιουργία ενός ρεαλιστικού netlist ούτως ώστε να ελαχιστοποιηθούν οι εκπλήξεις σε μετέπειτα στάδιο. Οι ελάχιστοι περιορισμοί που πρέπει να τεθούν, είναι η δήλωση των ρολογιών (όσων περιλαμβάνει η σχεδίαση), η δήλωση καθυστέρησης των εισόδων και των εξόδων ως προς το ρολόι, η εισαγωγή μιας τυπικής οδηγητικής ικανότητας των εισόδων και ενός τυπικού φορτίου των εξόδων και η επιλογή μιας εκ των τριών καταστάσεων λειτουργίας των κυττάρων της τεχνολογίας. Όλοι οι περιορισμοί ορίζονται από αντίστοιχες εντολές και καλό είναι να συγκεντρώνονται σε ένα script, για την διευκόλυνση του σχεδιαστή και την εξοικονόμηση χρόνου. Σε κάθε περίπτωση για τη βέλτιστη χρήση του εργαλείου ενθαρρύνεται η ανάγνωση των manual πρώτα, αν και είναι αρκετές εκατοντάδες σελίδες. Τόσο στον RTL Compiler όσο και στο Design Vision τα manual βρίσκονται στον φάκελο της εγκατάστασης, είτε σε μορφή pdf είται σε html.

Η δήλωση των ρολογιών είναι συνήθως το πρώτο που γίνεται. Πρέπει να δοθεί το όνομα του port που αντιστοιχεί σε κάθε ωρολογιακό σήμα καθώς και η επιθυμητή περίοδος. Ακόμα δίνεται η δυνατότητα να επιλεχθεί κυματομορφή, να οριστεί δηλαδή για πόσο μέσα σε μια περίοδο το ρολόι θα είναι high και για πόσο low, η προεπιλογή είναι να είναι ίσοι οι χρόνοι. Μπορούν να δηλωθούν ακόμα σαν ρολόγια και σήματα που γεννιούνται μέσα σε εσωτερικές μονάδες. Γενικά και τα δύο εργαλεία προσφέρουν παρόμοιες επιλογές.

Οι καθυστερήσεις των εισόδων και τον εξόδων προς το ρολόι πρέπει να δηλωθούν μετά το ρολόι, αλλιώς θα υπάρξει μήνυμα λάθους. Οι περιορισμοί αυτοί σχετίζονται άμεσα με τη φύση του σχεδίου (αν έχει pads κ.τ.λ) και πρέπει να δοθούν με προσοχή και με γνώμονα την επιθυμητή περίοδο. Χονδρικά η καθυστέρηση εισόδου είναι ο χρόνος που χρειάζονται οι είσοδοι για να φτάσουν τον πυρήνα του chip και η καθυστέρηση εξόδου ο χρόνος που χρειάζονται οι έξοδοι για να φτάσουν έξω. Μπορεί να τεθεί για όλες τις εισόδους ο ίδιος περιορισμός ή διαφορετικός. Το ίδιο ισχύει και για τις εξόδους. Σε κάθε περίπτωση είναι μια επιλογή που θα επηρεάσει το αποτέλεσμα της σύνθεσης

καθώς μειώνει τον πραγματικό χρόνο που έχει στη διάθεσή του το chip για να εκτελέσει εργασίες.

Για να είναι η σύνθεση ρεαλιστική, πρέπει να δοθεί μια οδηγητική ικανότητα στις εισόδους και ένα φορτίο στις εξόδους. Ο τρόπος που γίνεται αυτό είναι να τεθεί στις εισόδους η οδηγητική ικανότητα ενός τυπικού κυττάρου τις βιβλιοθήκης και αντίστοιχα στις εξόδους το φορτίο εξόδου ενός τυπικού κυττάρου. Οι δύο αυτοί αριθμοί μπορούν να βρεθούν μέσω απλών εντολών των εργαλείων σύνθεσης, ή μπορούν να βρεθούν στα datasheets της εκάστοτε τεχνολογίας.

Πριν το mapping πρέπει να δηλωθεί η επιθυμητή κατάσταση λειτουργίας των κυττάρων, η οποία θα υπαγορεύσει και τον χρονισμό τους βέβαια. Θεωρητικά πρέπει να χρησιμοποιείται η βιβλιοθήκη με τις χειρότερες συνθήκες και επειδή κάθε βιβλιοθήκη έχει συνήθως και υποπεριπτώσεις, πρέπει να χρησιμοποιήται η χειρότερη. Αυτό είναι αληθές όταν το chip προορίζεται για mission critical εφαρμογές όπως είναι οι στρατιωτικές, στις αερομεταφορές, στην ιατρική και γενικά οπουδήποτε απώλεια αξιοπιστίας σημαίνει απώλεια ζωής. Σε οποιαδήποτε άλλη όμως εφαρμογή που λόγου χάρη είναι απολύτως βέβαιο ότι το chip δε θα λειτουργεί στους 125℃ π.χ, είναι συχνή τακτική να χρησιμοποιούνται οι λίγο πιο ευνοϊκές βιβλιοθήκες, που διευκολύνουν τις επιδόσεις. Στην παρούσα εργασία υιοθετήθηκε το θεωρητικά σωστό.

Τέλος, υπάρχουν πάρα πολλές ακόμα ρυθμίσεις, κάποιες για πολύ έμπειρους σχεδιαστές που ξέρουν πως να πιέσουν τα εργαλεία και τη σχεδίαση τους για βέλτιστα αποτελέσματα.

4.1.4 Ρύθμιση παραμέτρων βελτιστοποίησης

Το τελευταίο βήμα πριν το mapping είναι η ρύθμιση των παραμέτρων βελτιστοποίησης. Συνήθως αυτό γίνεται με τη μορφή ορισμάτων των εντολών του mapping. Οι παράμετροι αυτοί μπορεί να είναι για βελτιστοποίηση χρονισμού, εμβαδού ή κατανάλωσης συνήθως. Κάποια στοιχεία μπορεί να βελτιωθούν και μετά το πρώτο mapping, καθώς υπάρχει δυνατότητα incremental mapping, δηλαδή επανακαθορισμός παραμέτρων και ξανά mapping μέχρι την επίτευξη ενός αποδεκτού αποτελέσματος. Το incremental mapping πέρνει σημαντικά λιγότερο χρόνο στην εκτέλεση καθώς το κύκλωμα δε διασπάται πάλι σε εξισώσεις από την αρχή, αλλά αφετηρία είναι ήδη μια καλή λύση.

Πρέπει όμως να τονιστεί ότι οι προσδοκίες πρέπει να είναι ρεαλιστικές. Το αποτέλεσμα της σύνθεσης ακολουθεί την καμπύλη pareto, όταν το εμβαδό είναι στόχος βελτιστοποίησης πάσχει ο χρονισμός και αντίστροφα. Συνεπώς είναι πολύ δύσκολο και σίγουρα παίρνει πολύ χρόνο το να φτάσουν όλοι οι

παράμετροι στα άκρα τους, τουλάχιστον στα άκρα των εργαλείων. Πολλά εξαρτώνται και από την τεχνολογία στόχο όμως καθώς μπορεί να είναι low power, οπότε η κατανάλωση δε θα είναι το μεγαλύτερο πρόβλημα, ή μπορεί να είναι high performance, όπου ο χρονισμός δε θα είναι το μεγαλύτερο πρόβλημα. Γενικά ανάλογα με την φύση του σχεδίου και τον προορισμό του γίνονται τα αντίστοιχα trade off. Για το chip του BLASTn οι τεχνολογίες που χρησιμοποιήθηκαν είναι standard performance οπότε υπάρχει μία ισορροπία μεταξύ χρονισμού και κατανάλωσης.

4.1.5 Mapping

Αφού ολοκληρωθούν όλες οι παραπάνω διαδικασίες είναι η σειρά του mapping. Πρόκειται για μία χρονοβόρα διαδικασία ανάλογα με το μέγεθος του σχεδίου βέβαια. Παρατηρήθηκε ότι σε σχέδια πιο structural με μικρές behavioral περιγραφές η διαδικασία ολοκληρώνεται σημαντικά πιο γρήγορα. Για παράδειγμα στην αρχιτεκτονική του BLASTn υπάρχουν shift registers των 20000 bit. Αν δινόντουσαν στο Design Vision γραμμένοι σε behavioral περιγραφές ο χρόνος που χρειαζόταν για σύνθεση ήταν τουλάχιστον δέκα φορές μεγαλύτερος από τον αντίστοιχο για μια πιο structural περιγραφή. Επίσης ο χρονισμός μπορούσε να επιτευχθεί με μεγαλύτερη άνεση. Παρόμοια συμπεριφορά είχε και ο RTL Compiler, παρότι ήταν αρκετά πιο γρήγορος από το Design Vision σε γενικές γραμμές. Επίσης ο RTL compiler έχει σημαντικά πιο επιθετική πολιτική βελτιστοποίησης η οποία μπορεί να δώσει εύκολα περίεργα αποτελέσματα. Στους ίδιους κώδικες που το Design Vision δεν έκανε καμία αλλαγή τρέχοντας σε ultra effort και δίνοντας εξαιρετικά αποτελέσματα ο RTL compiler αφαιρούσε εξαιρετικά πολλά instances δημιουργώντας ένα όχι και τόσο αξιόπιστο netlist.

Μετά το πέρας του mapping και αφού επαναληφθεί όσες φορές χρειαστεί η διαδικασία σε incremental mode, έχει παραχθεί το netlist του chip και το αρχείο περιορισμών sdc. Πριν όμως ξεκινήσει το επόμενο στάδιο της τοποθέτησης και διασύνδεσης υπάρχει κάτι ακόμα που πρέπει να γίνει· να δημιουργηθεί το padframe.

4.1.6 Η σύνθεση της αρχιτεκτονικής του BLASTn

Η αρχιτεκτονική του BLASTn παρουσίαζε κάποιες ιδιαιτερότητες στην σύνθεση. Αρχικά έπρεπε να γίνουν αρκετά πιο structural οι περιγραφές των μεγάλων shift register, καθώς τα εργαλεία αργούσαν πολύ σε αντίθετη περίπτωση. Επίσης μετά από δοκιμές προέκυψε πως η σύνθεση πρώτα των μικρών behavioral περιγραφών των κομματιών των shift registers έκανε την όλη

διαδικασία αρκετά πιο γρήγορη, χωρίς όμως αλλαγή του αποτελέσματος. Παρατηρήθηκε πως τα εργαλεία κάνοντας βελτιστοποιήσεις πέτυχαν ένα netlist με σχεδόν 10% λιγότερα κύτταρα από το αναμενόμενο όπως υπολογίστηκε στο προηγούμενο κεφάλαιο. Ο χρονισμός τέθηκε από τις μνήμες στα 300MHz περίπου και για τις δύο υλοποιήσεις.

4.2 Δημιουργία padframe

Σε αυτό το σημείο, ακριβώς πριν την τοποθέτηση και διασύνδεση, είναι η κατάλληλη στιγμή για να προστεθούν τα pads(τα οποία στις περισσότερες τεχνολογίες περιλαμβάνουν και pad drivers). Τα pads έρχονται σαν ξεχωριστές I/O βιβλιοθήκες. Συνήθως δεν είναι στην ίδια τεχνολογία με τα κύτταρα του πυρήνα και συχνά απαιτούν διαφορετική τάση τροφοδοσίας. Τα pads μπορεί να είναι in line ή staggered δηλαδή σε μια γραμμή η σε παραπάνω. Η τελευταία επιλογή απευθύνεται σε chip με μεγάλο αριθμό pads.

Δύο ενέργειες πρέπει να γίνουν για την δημιουργία padframe. Αρχικά πρέπει να προστεθούν με το χέρι στο netlist. Το επόμενο βήμα είναι η δημιουργία ενός ειδικού αρχείου κειμένου, του αρχείου I/O. Σε παλιότερες εκδόσεις του Design Vision, υπήρχε η επιλογή να προστεθούν αυτόματα pads με τη χρήση ενός attribute και μιας εντολής. Γενικά η χειρωνακτική προσθήκη είναι η απλούστερη και πιο σίγουρη μέθοδος, αν και για κυκλώματα με πολλά pads μπορεί να είναι κουραστική.

Οι I/O βιβλιοθήκες έχουν περισσότερα pads από όσα κάποιος θα περίμενε. Υπάρχουν συνήθως ξεχωριστές επιλογές για ωρολογιακά σήματα, τροφοδοσία και γείωση καθώς και για κάποια ειδικά σήματα. Συνεπώς η επιλογή του καταλληλότερου pad και ειδικά της καταλληλότερης οδηγητικής ικανότητας για το κάθε σήμα μπορεί να βοηθήσει σημαντικά στην επιτυχία του τελικού αποτελέσματος.

Εκτός από τα pads, το padframe περιέχει fillers και corner pads. Αυτές οι δομές δεν μπορούν να χρησιμοποιηθούν για I/O, αλλά χρησιμεύουν για το κλείσιμο του πλαισίου, το γέμισμα δηλαδή κάθε κενού μεταξύ των pad. Αυτό χρησιμεύει κυρίως στην συνέχεια της καλωδίωσης των pad, που περιλαμβάνει αγωγούς τροφοδοσίας και γείωσης. Έτσι χωρίς καμία ενέργεια από την πλευρά του σχεδιαστή οι δακτύλιοι τροφοδοσίας και γείωσης για το πλαίσιο είναι εξ ορισμού έτοιμοι. Στο netlist προστίθενται μόνο τα pads του top level entity, οι γωνίες, οι τροφοδοσίες και οι γειώσεις δηλώνονται στο αρχείο I/O.

Σε κάθε πλευρά του chip, αντιστοιχεί ένας προσανατολισμός, στην πάνω βόρεια, κάτω νότια, δεξιά ανατολικά και αριστερά δυτικά. Αυτή τη σημειογραφία χρησιμοποιεί το αρχείο I/O. Για να το συνταχθεί πρέπει να είναι γνωστό που

ακριβώς θα τοποθετηθεί το κάθε pad και πόσες τροφοδοσίες και γειώσεις χρειάζονται για το padframe και τον πυρήνα. Αυτή η τελευταία πληροφορία προκύπτει από τη μελέτη των εκάστοτε διαθέσιμων βιβλιοθηκών, γνωρίζοντας δηλαδή τη μέση ισχύ που χρειάζεται ο πυρήνας και με βάση κάποιους πίνακες που έχουν τα datasheets, υπολογίζεται ο αριθμός των power pads του πυρήνα. Ομοίως για το padframe, ανάλογα με τον αριθμό των pads, ειδικά των output επειδή αυτά έχουν μεγαλύτερο ρεύμα λόγω οδηγητικής ικανότητας, αποφασίζεται ο αριθμός των power pads του padframe. Η βιβλιοθήκη δίνει επίσης και το ελάχιστο απαιτούμενο spacing που πρέπει να έχουν τα pads. Εδώ πρέπει να δοθεί αρκετή προσοχή καθώς όσο περισσότερο κενό έχουν, τόσο περισσότερο μειώνεται το φαινόμενο που δημιουργήται όταν γειτονικά pads αλλάζουν ταυτόχρονα τιμή. Στο σχήμα 4.3 φαίνεται η σύνταξη του αρχείου I/O.

Η πρώτη τεχνολογία που χρησιμοποιήθηκε, η UMC 0,13μm, προσέφερε μεγάλο αριθμό pad με διαφορετικές οδηγητικές ικανότητες και χαρακτηριστικά. Η δεύτερη προσέφερε πολύ μικρότερο αριθμό, άλλα προγραμματιζόμενα pad, με ρυθμιζόμενη οδηγητική ικανότητα, pull up και pull down δυνατότητες καθώς και την επιλογή enable.

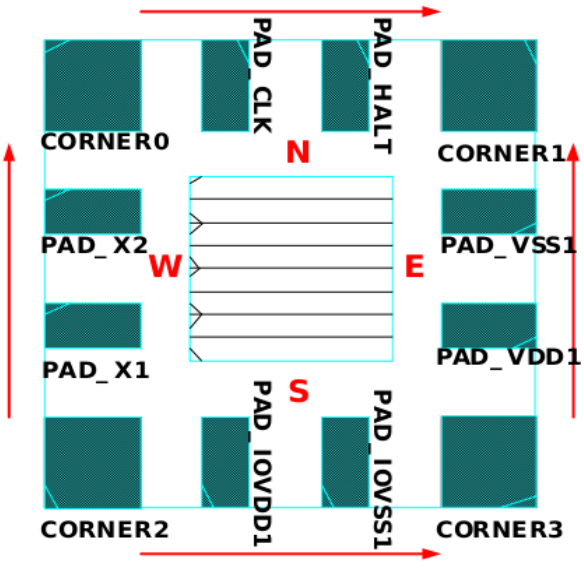
Στην πρώτη παραλλαγή του chip χρησιμοποιήθηκαν in line pads, καθώς το σχέδιο θα μπορούσε να χαρακτηριστεί core limited, δηλαδή ο κύριος παράγοντας που υπαγορεύει τις ελάχιστες διαστάσεις είναι ο πυρήνας, ενώ με τη δεύτερη τεχνολογία, χρησιμοποιήθηκαν staggered pads, καθώς οι διαστάσεις του πυρήνα μειώθηκαν λόγω τις μικρότερης τεχνολογίας και το σχέδιο έγινε pad limited. Τα pad είναι σχεδιασμένα έτσι ώστε να προστατεύουν τον πυρήνα από απότομα και ισχυρά ρεύματα, χρησιμοποιώντας κάποια ειδικά κυκλώματα πριν παραδώσουν το σήμα στον πυρήνα.

```
Version: 1
Pad: CORNER0 NW
Pad: PAD_CLK      N
Pad: PAD_HALT     N

Pad: CORNER1      NE
Pad: PAD_X1       W
Pad: PAD_X2       W

Pad: CORNER2      SW
Pad: PAD_IOVDD1   S
Pad: PAD_IOVSS1   S

Pad: CORNER3      SE
Pad: PAD_VDD1     E
Pad: PAD_VSS1     E
```



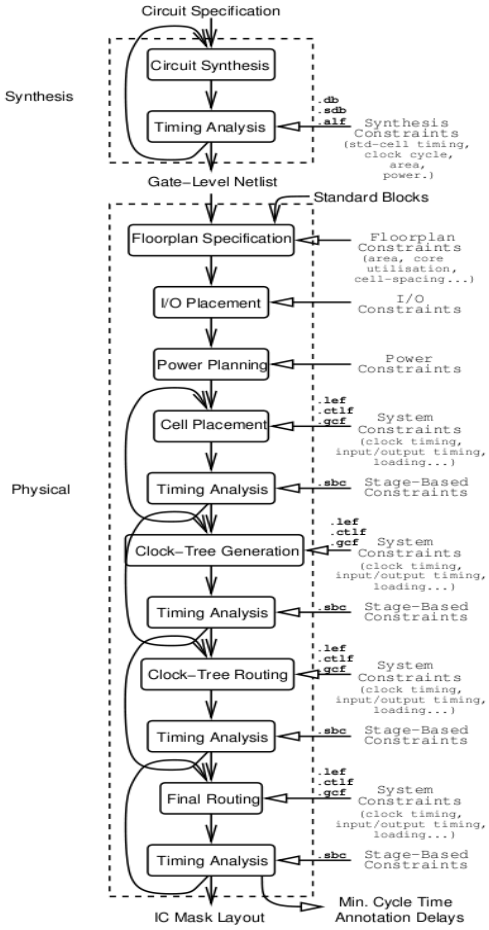
Σχήμα 11. Δημιουργία αρχείου IO

Υπάρχουν αναλυτικές οδηγίες για την σύνταξη του αρχείου IO στο manual του encounter. Υπάρχει η οδηγία spacing, η οποία είναι πολύ σημαντική καθώς καθορίζει την απόσταση που θα τοποθετηθούν τα pads. Χωρίς τη σωστή χρήση της είναι δύσκολο να δημιουργηθεί padframe ελεύθερο από DRC Violations.

Επίσης είναι πολύ σημαντική η θέση που θα επιλεγεί για κάποια pads, όπως είναι το ρολόι. Αν τοποθετηθεί στην άκρη θα είναι πολύ πιο δύσκολο να δημιουργηθεί ένα καλό δέντρο ρολογιών επηρεάζοντας σημαντικά την απόδοση του κυκλώματος. Προτείνεται να έχει κεντρική θέση, όπως και τα power-ground pads για τον πυρήνα και το padframe. Για τα τελευταία, καλό είναι να αποφεύγεται η τοποθέτηση δίπλα δίπλα. Όπως αναφέρθηκε και νωρίτερα οι γωνίες και τα power-ground pads, δηλώνονται μόνο στο αρχείο IO.

4.3 Τοποθέτηση και διασύνδεση

Σειρά μετά την εξαγωγή του Verilog netlist, του αρχείου sdc και την δημιουργία του padframe, έχει το τελικό βήμα της σχεδίασης που είναι η τοποθέτηση και διασύνδεση [25]. Αυτό είναι ίσως και το σημαντικότερο τμήμα αυτής της εργασίας. Ακολουθεί η γενική ροή της τοποθέτησης και διασύνδεσης.



Σχήμα 12. Ροή της τοποθέτησης και διασύνδεσης

Το εργαλείο που χρησιμοποιήθηκε για αυτή τη φάση είναι το SOC Encounter της Cadence. Η έκδοση που διαθέτει το Πολυτεχνείο Κρήτης είναι η 6.2 με τα updates. Πρόκειται για ένα βιομηχανικό εργαλείο με πολύ μεγάλες δυνατότητες που χαίρει εξαιρετικής αναγνώρισης στο χώρο του. Το εργαλείο στην τελευταία του έκδοση διαφημίζει ότι σε τεχνολογίες μέχρι και 32nm, μπορεί άνετα να χειριστεί σχέδια μεγαλύτερα από 55 εκατομμύρια πύλες.

Όπως και τα εργαλεία σύνθεσης έτσι και αυτό είναι προορισμένο για χρήση με scripts. Εδώ όμως, το γραφικό περιβάλλον προσφέρει σχεδόν την ίδια λειτουργικότητα με το scripting αλλά και πάλι είναι πολύ βολικότερη η δημιουργία script για κάθε βήμα της ροής. Καθώς το εργαλείο δημιουργεί logs με τις εντολές που δόθηκαν σε κάθε session, είναι δυνατό να χρησιμοποιήται το γραφικό περιβάλλον και ύστερα οι εντολές να παίρνονται από το log και να γίνονται script. Επίσης υπάρχει δυνατότητα σωσίματος του σχεδίου ανά πάσα στιγμή και μετέπειτα φορτώσεως του χωρίς κανένα πρόβλημα.

Όπως είναι φανερό και από το όνομα της διαδικασίας που θα ακολουθήσει, χονδρικά υπάρχουν δύο φάσεις· η τοποθέτηση και η διασύνδεση. Το πρώτο είναι η φυσική τοποθέτηση των κυττάρων της τεχνολογίας στόχου μέσα στο chip. Το δεύτερο είναι η μεταξύ τους διασύνδεση όπως υπαγορεύει το netlist, ούτως ώστε να προκύψει η λειτουργικότητα του netlist. Οι δύο αυτές διαδικασίες εκτελούνται σύμφωνα με αλγορίθμους του εργαλείου με γνώμονα τον χρονισμό την κατανάλωση και το congestion, τον «συνωστισμό» των κυττάρων στον πυρήνα. Εκτός από τα δύο αυτά κύρια βήματα υπάρχουν και άλλες απαραίτητες διαδικασίες που παρουσιάζονται παρακάτω.

Κάθε chip γενικά αποτελείται από τρεις περιοχές. Η πρώτη είναι το padframe, η δεύτερη ο πυρήνας και η τρίτη το μεταξύ τους κανάλι που προορίζεται για την δημιουργία δακτυλίων τροφοδοσίας και γείωσης. Στο σχεδιαστικό πακέτο κάθε τεχνολογίας, εκτός από τα αρχεία χρονισμού που χρειάζονται στη σύνθεση, περιέρχονται και τα απαραίτητα αρχεία για την τοποθέτηση και διασύνδεση, που περιέχουν φυσικές πληροφορίες για κάθε κύτταρο και γενικές πληροφορίες για την τεχνολογία στόχο. Υπάρχουν ξεχωριστά αρχεία για τις δύο κύριες περιοχές του chip, τα κύτταρα του πυρήνα και τα IO, όπως και στην σύνθεση. Επίσης ξεχωριστά είναι και τα αρχεία για τα έτοιμα blocks όπως οι μνήμες.

Η Faraday 90nm, έχει εννέα επίπεδα μετάλλου, τα πρώτα έξι με κανονικό πάχος, τα επόμενα δύο με διπλό και το τελευταίο με τετραπλό. Η UMC 0,13μm έχει οκτώ επίπεδα μετάλλου, τα πρώτα έξι με κανονικό πάχος και τα τελευταία δύο με διπλό. Ο κύριος λόγος για την αλλαγή του πάχους, είναι πως όσο ψηλότερα βρίσκεται το μέταλλο τόσο μεγαλύτερη αντοχή πρέπει να έχει για να μην σπάσει, καθώς τα ψηλότερα επίπεδα έχουν πιο ανώμαλες επιφάνειες, λόγω

των πολλών στρώσεων που υπάρχουν από κάτω.

Δυστυχώς η Faraday 90nm, έχει έναν κατασκευαστικό περιορισμό που δεν καλύπτεται από το encounter, το VIAFARM. Αυτός ο περιορισμός υποστηρίζεται μόνο από το Astro της Synopsys. Συνεπώς οποιοδήποτε σχέδιο κατατίθεται για κατασκευή πρέπει πρώτα να περάσει από την αυτοματοποιημένη διόρθωση VIAFARM, που διατίθεται από την εταιρία επί πληρωμή.

Σε οποιοδήποτε σημείο κριθεί απαραίτητο μπορεί να χρησιμοποιηθεί ο ενσωματωμένος Design Rule Check (DRC) ελεγκτής Verify Geometry για την εμφάνιση τυχόν violations.

4.3.1 Εισαγωγή αρχείων τεχνολογίας

Για να ξεκινήσει η διαδικασία, πρέπει να δοθούν στο εργαλείο όλα τα απαραίτητα αρχεία. Αρχικά πρέπει να δοθεί το netlist και το αρχείο περιορισμών sdc, τα οποία είναι τα αποτελέσματα της λογικής σύνθεσης. Ύστερα πρέπει να φορτωθούν τα αρχεία χρονισμού, όπως και στην σύνθεση, για τα κύτταρα του πυρήνα, τα pads, τα έτοιμα blocks και για τις τρεις καταστάσεις λειτουργίας(worst, typical και best). Πρέπει να δοθεί το αρχείο IO και τα αρχεία .lef, που περιέχουν τις φυσικές πληροφορίες των κυττάρων, των IO και των έτοιμων block. Τα αρχεία .lef συνήθως είναι αρκετά και μπορεί να προκληθούν λάθη αν δεν διαβαστούν με τη σωστή σειρά. Κατά πάσα πιθανότητα η σωστή σειρά αναγράφεται στα datasheets της τεχνολογίας, αλλά στην περίπτωση που δεν συμβαίνει κάτι τέτοιο, πρέπει να διαβαστούν πρώτα τα headers, ύστερα τα macros και στο τέλος τα antenna. Τα πρώτα περιέχουν γενικές πληροφορίες για την τεχνολογία στόχο, το δεύτερο τα μοντέλα των κυττάρων και το τρίτο πληροφορίες για το φαινόμενο antenna [27]. Πρέπει να δηλωθούν απαραιτήτως τα ονόματα των κόμβων τροφοδοσίας και γείωσης που υπάρχουν στα datasheets, καθώς και τα ονόματα των buffer, inverter και delay cells, που θα χρησιμοποιηθούν στην διαδικασία της επιτόπιας βελτιστοποίησης.

Όλα αυτά τα αρχεία και πληροφορίες μπορούν να εισαχθούν εντολή-εντολή, ή μέσω μιας βοηθητικής φόρμας του γραφικού περιβάλλοντος του encounter. Αφού συμπληρωθεί η φόρμα, μπορεί να σωθεί για μετέπειτα χρήση της με τη μορφή ενός script με την κατάληξη .conf. Αφου ολοκληρωθεί με επιτυχία αυτή η διαδικασία θα φανεί ο πυρήνας του chip με τα pads γύρω του και δίπλα θα υπάρχουν σαν κουτιά τα έτοιμα blocks και οι υπόλοιπες δομές του netlist.

4.3.2 Ορισμός floorplan

Η πρώτη ενέργεια που πρέπει να γίνει είναι να οριστούν οι διαστάσεις του

chip και το μέγεθος του καναλιού μεταξύ πυρήνα και IO. Πρόκειται για ένα πολύ σημαντικό βήμα καθώς θα επηρεάσει σε μεγάλο βαθμό την υπόλοιπη διαδικασία. Το μέγεθος του chip, μπορεί είτε να είναι δεδομένο, είτε να είναι στην ευχέρεια του σχεδιαστή ως ένα βαθμό. Ένας καλός παράγοντας για την εύρεση των κατάλληλων διαστάσεων είναι η χρήση του πυρήνα. Το βέλτιστο γενικά είναι μια χρήση 50%-60%. Σε μετέπειτα στάδια ο παραπάνω χώρος θα χρησιμοποιηθεί για την βελτιστοποίηση του σχεδίου με την εισαγωγή buffers. Αυτοί οι αριθμοί όμως δεν είναι δεσμευτικοί, πρέπει να συνυπολογιστεί και η φύση του σχεδίου, αν έχει έντονο routing κ.τ.λ. Στην παρούσα εργασία χρησιμοποιήθηκαν πυκνότητες από 55% μέχρι 60%.

Μεγάλο κομμάτι από το συνολικό εμβαδό καταλαμβάνουν τα pads. Έχει μεγάλη σημασία η σωστή επιλογή μεταξύ staggered και in-line pads. Σε σχέδια με μικρό αριθμό pad αναλογικά με το μέγεθος του πυρήνα, ταιριάζει καλύτερα η δεύτερη επιλογή και σε σχέδια με μεγάλο αριθμό η πρώτη. Η κύρια διαφορά τους σε επίπεδο floorplan είναι ότι τα staggered είναι πιο στενά, αλλά πιο μακριά. Μπορεί το εμβαδό τους να είναι παρόμοιο, υπάρχει μεγάλο κέρδος όμως στο τέλος, λόγο της δυνατότητας να υπάρχει σχεδόν διπλάσιος αριθμός σε κάθε πλευρά σε σχέση με τα in-line. Αν πρέπει να πιεστεί στα άκρα το σχέδιο, υπάρχει η δυνατότητα να χρησιμοποιηθούν και τα δύο είδη στο ίδιο chip, με χρήση ειδικών γωνιών που αλλάζουν τον τύπο του pad.

Έχοντας ολοκληρώσει τον ορισμό του floorplan το επόμενο βήμα είναι να κλείσει το padframe με fillers.

4.3.3 Γέμισμα του padframe

Τα κενά μεταξύ των pads πρέπει απαραιτήτως να γεμίσουν με ειδικά για αυτό το σκοπό κύτταρα, που διαθέτει κάθε IO βιβλιοθήκη και λέγονται filler ή empty cells. Τα κύτταρα αυτά έρχονται σε πολλά μεγέθη και μερικές φορές διαφημίζουν ότι μπορούν να γεμίσουν κάθε κενό. Η καλύτερη προσέγγιση είναι να επιλεχθεί κατάλληλο spacing των pads στο αρχείο IO, ούτως ώστε να χωράνε ακριβώς fillers και οι πλευρές του chip να είναι ακέραια πολλαπλάσια του grid της τεχνολογίας, της ελάχιστης μονάδας μήκους δηλαδή, του κάθε κατασκευαστή. Ίσως το πλαίσιο να μην κλείσει σωστά με την πρώτη απόπειρα και να χρειάζονται μικρές αλλαγές στο spacing στο αρχείο IO, ή στις διαστάσεις του chip, σε κάθε περίπτωση όμως το πλαίσιο πρέπει να κλείσει χωρίς violations. Αυτό διότι με αυτόν το τρόπο, δημιουργούνται οι δακτύλιοι τροφοδοσίας και γείωσης των pads και συνεπώς είναι εφικτή η λειτουργία τους.

Μετά την ολοκλήρωση αυτού του βήματος καλό είναι να σημειωθούν τα ονόματα των pins τροφοδοσίας και γείωσης των κυττάρων πυρήνα που θα

συνδεθούν αργότερα στην τροφοδοσία ή τη γείωση. Αυτό δεν είναι δεσμευτικό να γίνει σε αυτό το σημείο, πρέπει όμως να γίνει πριν το τελικό power routing που αναλύεται παρακάτω.

4.3.4 Δημιουργία power plan

Η δημιουργία ενός καλού power plan είναι ζωτικής σημασίας για ένα αξιόπιστο κύκλωμα. Συνεπώς είναι ένα από τα τμήματα της σχεδίασης που απαιτεί ιδιαίτερη προσοχή.

Το power plan αποτελείται από τους δακτυλίους τροφοδοσίας και γείωσης γύρω από τον πυρήνα, τους δακτυλίους τροφοδοσίας και γείωσης γύρω από τα έτοιμα blocks, τις κάθετες και οριζόντιες λωρίδες. Σε μετέπειτα βήμα ολοκληρώνεται με το τελικό power routing που συνδέει σε κάθε κύτταρο μετά την τοποθέτηση τροφοδοσία και γείωση.

Για ένα επιτυχημένο power routing, πρέπει να έχει γίνει μια καλή εκτίμηση του μέσου ρεύματος που χρειάζεται το κύκλωμα για τη συχνότητα που είναι επιθυμητό να λειτουργεί. Αυτό γίνεται είτε με κάποιες εξισώσεις που υπάρχουν στα datasheets της τεχνολογίας, είτε μέσω του encounter που υπολογίζει κατευθείαν με μεγάλη ακρίβεια αυτόν τον αριθμό. Όταν υπολογιστεί αυτός ο αριθμός τότε υπολογίζεται το πλάτος των δακτυλίων τροφοδοσίας και γείωσης. Είναι σημαντικό να έχουν αρκετό πλάτος αυτοί οι αγωγοί, καθώς κινδυνεύουν από υπερθέρμανση και το φαινόμενο της ηλεκτρομετανάστευσης [26]. Το φαινόμενο αυτό λαμβάνει χώρα όταν το πλάτος ενός αγωγού είναι πολύ μικρό για το ρεύμα που περνά από μέσα του στη μονάδα του χρόνου, οπότε αυξάνει η πυκνότητα του ρεύματος πάνω από ένα συγκεκριμένο όριο και κυριολεκτικά τα ηλεκτρόνια παρασύρουν τα μόρια του μετάλλου καταστρέφοντας τη δομή του.

Το power plan μπορεί να γίνει από τον σχεδιαστή ή από το εργαλείο. Στην παρούσα εργασία δοκιμάστηκαν και οι δύο μέθοδοι. Η πρώτη με την UMC 0,13μm και η δεύτερη με την Faraday 90nm. Παρατηρήθηκε πως το εργαλείο κάνει επιλογές που ταιριάζουν περισσότερο ίσως στο προφίλ κάθε τεχνολογίας. Επίσης παρατηρήθηκαν κάποιες ενδιαφέρουσες επιλογές που το νόημά τους δεν ήταν προφανές εξαρχής. Για παράδειγμα κάθε δακτύλιος αποτελείται από δύο επίπεδα μετάλλου με αποτέλεσμα να δημιουργήται έντονη χωρητικότητα. Καθώς η τελευταία είναι γενικά ανεπιθύμητη καθώς επηρεάζει αρνητικά τον χρονισμό, στους αγωγούς τροφοδοσίας και γείωσης είναι ευπρόσδεκτη, καθώς κάνει πιο σταθερή την τάση του κάθε αγωγού. Το εργαλείο χρησιμοποίησε stacked vias για να συνδέσει αυτούς τους αγωγούς. Ο μόνος ουσιώδης τρόπος για την εξέταση του καλύτερου power plan θα ήταν η δοκιμή του πραγματικού chip, οπότε θα

μείνει άγνωστο αφού το chip αυτής της εργασίας, είναι ένα ακαδημαϊκό προϊόν.

Γενικά ένα καλό power plan επιτρέπει την υγιή τροφοδοσία και γείωση του chip, χωρίς να θυσιάζονται σε μεγάλο βαθμό πόροι routing. Το κύριο τμήμα που πρέπει να δωθεί προσοχή είναι τα strippes. Τα τελευταία συνήθως μπαίνουν από το τέταρτο επίπεδο μετάλλου και πάνω, τόσο οριζόντια όσο και κάθετα. Πειραματικά έχει βρεθεί πως συχνά και λεπτά strippes, δουλεύουν καλύτερα από αραιά και χοντρά. Κάποια σχέδια έχουν τη δυνατότητα να διαθέσουν περισσότερους πόρους στο power plan λόγω χαμηλότερης πυκνότητας. Εκεί ακολουθείται διαφορετική προσέγγιση όπου τα δύο τελευταία επίπεδα μετάλλου ουσιαστικά αφιερώνονται στα strippes, καθώς είναι συχνά και πλατιά.

Αφού ολοκληρωθεί το power planning, το επόμενο βήμα είναι η χειρωνακτική τοποθέτηση των έτοιμων δομών και η αρχική τοποθέτηση των κυττάρων.

4.3.5 Τοποθέτηση

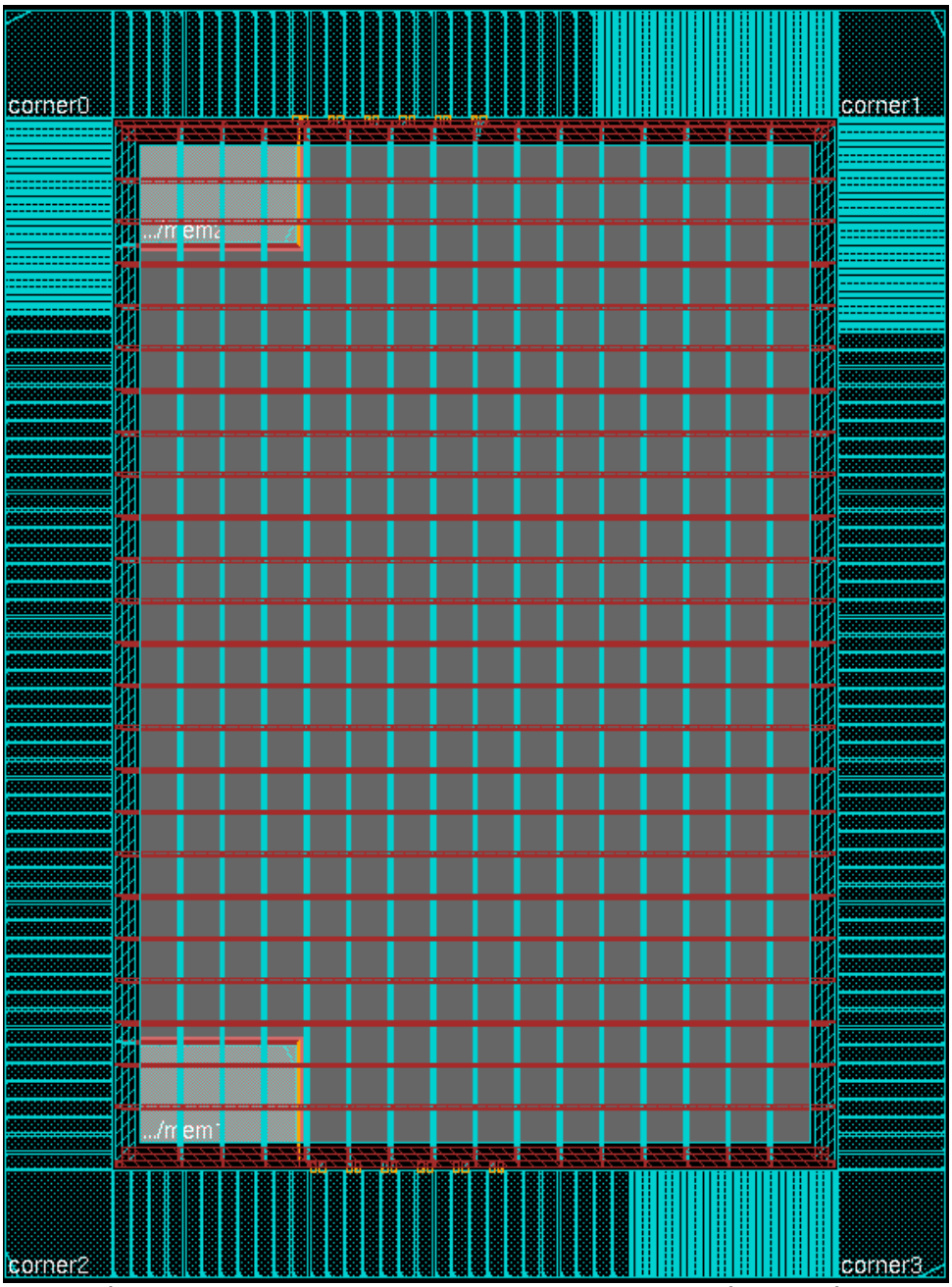
Η τοποθέτηση είναι ένα από τα δύο μεγάλα βήματα της όλης διαδικασίας. Το εργαλείο προσφέρει πάρα πολλές επιλογές για την εκτέλεσή της, οι οποίες καλύπτουν κάθε ανάγκη. Δυστυχώς δεν είναι multithreading όπως θα περίμενε κανείς, κάτι που την καθιστά με διαφορά την πιο χρονοβόρα διαδικασία όταν εκτελείται με την επιλογή της επιτόπιας βελτιστοποίησης.

Πριν αρχίσουν να τρέχουν οι αλγόριθμοι της τοποθέτησης θα πρέπει να έχουν τοποθετηθεί χειρωνακτικά οι έτοιμες δομές του σχεδίου. Προτείνεται η χειρωνακτική τοποθέτηση καθώς αν επιλέξει το εργαλείο τη θέση τους, κατά πάσα πιθανότητα θα δυσκολέψει πολύ το routing. Η προτεινόμενη θέση για αυτόν ακριβώς το λόγο είναι στα άκρα του chip και δη στις γωνίες. Αυτή η θέση εμποδίζει το λιγότερο δυνατό το routing, προσοχή όμως πρέπει να δοθεί στον προσανατολισμό της κάθε μονάδας, ώστε οι είσοδοι και έξοδοί της να είναι προς τον πυρήνα και όχι τα pads. Επίσης κάθε έτοιμη μονάδα πρέπει να έχει τους δικούς της δακτυλίους τροφοδοσίας και γείωσης. Για να γίνει αυτό πρέπει πρώτα να δεσμευτεί χώρος γύρω από τη μονάδα για το λόγο αυτό. Αν οι μονάδες είναι στις γωνίες, μπορεί να έχουν δικό τους δακτύλιο μόνο στις δύο πλευρές και στις άλλες να «δανείζονται» τους δακτυλίους του πυρήνα. Αυτή είναι και η στρατηγική που ακολουθήθηκε σε αυτήν την εργασία για την εξοικονόμηση χώρου στον πυρήνα.

Αν είναι επιθυμητό το εργαλείο μπορεί να τρέξει μία γρήγορη τοποθέτηση σε floorplan mode όπως λέγεται, για να φανεί πώς γενικά σκοπεύει να κατανείμει τις μονάδες στον πυρήνα. Βεβαίως αυτό το βήμα δεν είναι απαραίτητο και απλά δίνει μια εκτίμηση. Η κανονική τοποθέτηση είναι

καθοδηγούμενη από το χρονισμό και μπορεί να περιλαμβάνει βελτιστοποίηση πριν την τοποθέτηση αλλά και επιτόπια βελτιστοποίηση. Η τελευταία διαδικασία είναι προαιρετική σε αυτό το στάδιο, αλλά προτείνεται αν υπάρχει επιπλέον χώρος στον πυρήνα και ο επιθυμητός χρονισμός είναι ιδιαίτερα επιθετικός. Η επιτόπια βελτιστοποίηση γίνεται κατά κανόνα μετά τη σύνθεση των ρολογιών και είναι πολύ σημαντική καθώς βελτιώνει την απόδοση κατά 30%-40% ίσως και παραπάνω. Η τοποθέτηση είναι διαθέσιμη και σε incremental mode.

Παράμετροι της τοποθέτησης που πρέπει να ρυθμιστούν πριν την έναρξη είναι το effort level, το congestion effort level, αν θα γίνει post place congestion opimazition, αν χρειάζεται να διατηρηθεί το routing(π.χ μετά από σύνθεση ρολογιών στην επιτόπια βελτιστοποίηση) κ.α.



Σχήμα 13. Το power planning και οι τοποθετημένες μνήμες

Στην εικόνα φαίνεται η σχάρα που δημιουργούν οι οριζόντιες και κάθετες λωρίδες τροφοδοσίας και γείωσης. Πάνω αριστερά και κάτω αριστερά φαίνονται οι μνήμες με τους δικούς τους δακτυλίους στις δύο πλευρές. Αχνά φαίνεται και λίγος χώρος(ροζ χρώμα) από τον δεσμευμένο για δακτυλίους που έχει αφεθεί σκόπιμα για αποφυγή violations στο τελικό power routing. Ορατές ακόμα είναι και οι συνδέσεις(πορτοκαλί χρώμα) των power ground pads για τον πυρήνα με τους δακτυλίους.

4.3.6 Σύνθεση ρολογιών

Αμέσως μετά την τοποθέτηση γίνεται η δημιουργία του δέντρου των ρολογιών. Το ρολόι για να μεταδίδεται όσο καλύτερα γίνεται σε όλα τα μέρη του chip πρέπει να διαδίδεται μέσα από ειδικές δομές, τα δέντρα Η, που στόχο έχουν την ελαχιστοποίηση της παραμόρφωσης. Η ποιότητα της διάδοσης του ωρολογιακού σήματος θέτει ένα άνω όριο στον μέγιστο χρονισμό που μπορεί να επιτευχθεί. Η σύνθεση των ρολογιών αποτελεί ίσως το σημαντικότερο βήμα για την βελτίωση του χρονισμού και της συνολικής αξιοπιστίας του ολοκληρωμένου.

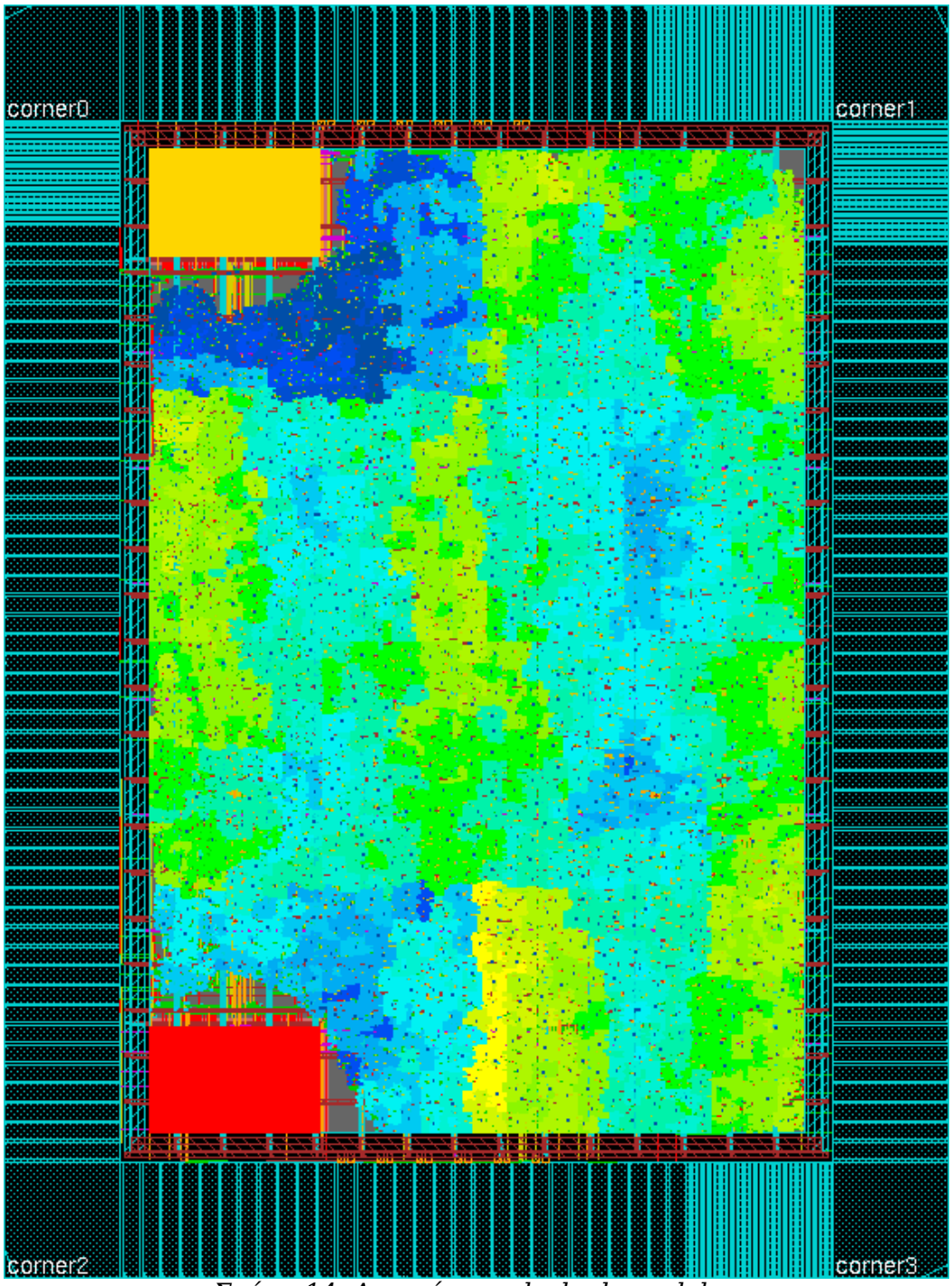
Η σύνθεση των ρολογιών γίνεται πριν από το routing, διότι έχει τη μεγαλύτερη προτεραιότητα και συνεπώς τον πρώτο λόγο στους διαθέσιμους πόρους διασύνδεσης. Υπάρχουν EDA, τα οποία εκτελούν μόνο σύνθεση ρολογιών. Μπορεί το σχέδιο να εξαχθεί από το encounter σε μορφή .def, να εισαχθεί στο οποιοδήποτε άλλο εργαλείο, να γίνει η σύνθεση ρολογιών και μετά να γυρίσει πάλι στο encounter. Δεν είναι αναγκαία μια τέτοια διαδικασία όμως καθώς τα τελευταία χρόνια το encounter δίνει εξαιρετικά αποτελέσματα σε αυτόν τον τομέα.

Δύο είναι τα εκτελέσιμα του εργαλείου που εκτελούν σύνθεση ρολογιών. Το πρώτο είναι το ckSynthesis και το δεύτερο το ckECO. Το πρώτο διαφημίζεται ώς καλύτερο(και πιο χρονοβόρο), αλλά παραδόξως το δεύτερο σε κάποια σχέδια δίνει καλύτερα αποτελέσματα(όπως συμβαίνει και με τον wroute, τον παλιό router). Έτσι λοιπόν η εταιρία διατηρεί και τα δύο εκτελέσιμα έτσι ώστε ο μηχανικός να επιλέγει το καταλληλότερο. Στην παρούσα εργασία αυτό είναι το ckSynthesis. Μπορούν όμως να χρησιμοποιηθούν και τα δύο με τη ρύθμιση των κατάλληλων μεταβλητών.

Για να γίνει η διαδικασία χρειάζεται ένα ειδικό αρχείο του οποίου η σύνταξη και οι δυνατότητες αναλύονται εκτενώς στα manual της cadence, μέσα στην εγκατάσταση του εργαλείου. Μπορεί να δημιουργηθεί αυτόματα από το εργαλείο με προεπιλεγμένα χαρακτηριστικά απαιτώντας από τον μηχανικό μόνο τα ονόματα των buffers, inverters και delay cells που κάθε βιβλιοθήκη διαθέτει αποκλειστικά για τη σύνθεση των ρολογιών. Για βέλτιστη όμως χρήση και

μεγιστοποίηση του αποτελέσματος χρειάζονται και κάποιες επεμβάσεις ανάλογα με την περίπτωση. Η διαδικασία αυτή θα δώσει εκτενείς αναφορές αν ζητηθεί.

Γενικά δύο είναι οι κύριες μέθοδοι για την βελτιστοποίηση του δέντρου. Ο πρώτος είναι η αλλαγή της οδηγητικής ικανότητας σε όποιους buffer κρίνει το εργαλείο σωστό. Το δεύτερο είναι η εισαγωγή νέων buffer ή inverter, από την προκαθορισμένη λίστα στο αρχείο σύνθεσης ρολογιών. Υπάρχει και η επιλογή να απενεργοποιηθούν και οι δύο αυτοί τρόποι. Το εργαλείο από προεπιλογή έχει μόνο τον πρώτο, για λόγους βελτιστοποίησης της κατανάλωσης κυρίως. Παρακάτω φαίνεται το δέντρο που επιτεύχθηκε με την Faraday 90nm.



Σχήμα 14. Απεικόνιση clock phase delay

Οι μπλέ περιοχές είναι αυτές με την ελάχιστη παραμόρφωση και οι κόκκινες το αντίθετο. Τα χρώματα είναι σε σχέση με το εύρος της παραμόρφωσης που έχει πετύχει το εργαλέιο, οπότε μπορεί στην εικόνα να είναι κόκκινες κάποιες περιοχές, η παραμόρφωσή τους όμως είναι καθόλα αποδεκτή.

4.3.7 Επιτόπια βελτιστοποίηση

Η διαδικασία της επιτόπιας βελτιστοποίησης λαμβάνει χώρα τυπικά μετά την σύνθεση των ρολογιών. Πρόκειται για μία διαδικασία που στόχο έχει την επίτευξη του επιθυμητού χρονισμού, με την εισαγωγή buffers, inverters, και delay cells, τα οποία έχουν καθοριστεί από το πρώτο κιόλας βήμα, την εισαγωγή δεδομένων στο εργαλείο.

Το βήμα αυτό βελτιώνει σημαντικά την απόδοση του κυκλώματος, καταναλώνοντας όμως χώρο στον πυρήνα. Αυτός είναι και ένας από τους κύριους λόγους που πρέπει η πυκνότητα να κρατείται σε λογικά επίπεδα, ούτως ώστε να μπορεί να γίνει αυτή η διαδικασία. Το εργαλείο αξιοποιεί τον χώρο όσο πιο καλά μπορεί και οι αλλαγές που συμβαίνουν στην πυκνότητα του πυρήνα πριν και μετά από αυτό το βήμα σπάνια ξεπερνούν το 3%-4%, τουλάχιστον στο κύκλωμα αυτής της εργασίας.

Η επιτόπια βελτιστοποίηση μπορεί να τρέξει και στην αρχική τοποθέτηση, βελτιώνοντας το έδαφος για την σύνθεση των ρολογιών, μόνο όμως όταν υπάρχει ο διαθέσιμος χώρος, γιατί πρέπει να εισαχθούν buffers για το ρολόι και να γίνει και μια δεύτερη επιτόπια βελτιστοποίηση. Επίσης υπάρχουν επιλογές για την διατήρηση του routing(δέντρο ρολογιών), την αφαίρεση routing που χάλασε από την βελτιστοποίηση κ.α.

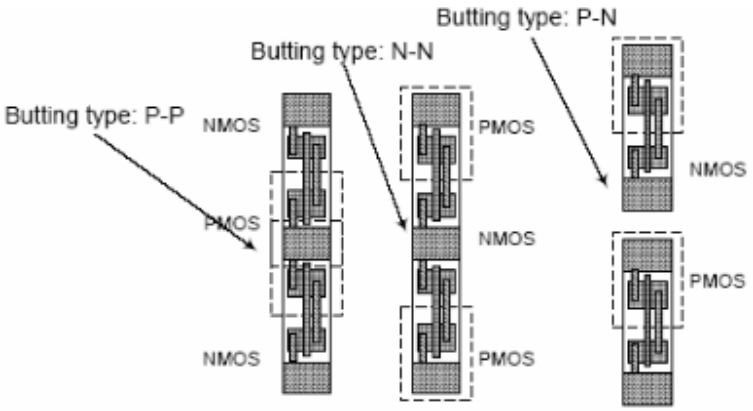
Αρχικά γίνεται ανάλυση του υπάρχοντος κυκλώματος για εξαγωγή καθυστερήσεων. Μελετούνται τα μονοπάτια με τη χειρότερη καθυστέρηση τα οποία μπαίνουν σε σειρά για βελτιώση. Τρεις είναι οι κύριοι τρόποι βελτίωσης, η εισαγωγή buffer, η αλλαγή του μεγέθους των υπαρχόντων και οι μικρές αλλαγές στη θέση διάφορων κυττάρων με σεβασμό στο υπάρχον routing.

Μόλις ολοκληρωθεί η επιτόπια βελτιστοποίηση, αν έχει επιτευχθεί ο ζητούμενος χρονισμός, το επόμενο βήμα είναι το τελικό power routing. Αν δεν έχει επιτευχθεί μπορεί να ξανατρέξει η βελτιστοποίηση, αλλιώς υπάρχουν σοβαρές πιθανότητες να μην μπορέσει να φτάσει το chip αυτόν το χρονισμό. Περιθώρια βελτίωσης υπάρχουν και από το nanoroute, άλλα αν από αυτό το βήμα το chip δεν τρέχει όσο γρήγορα χρειάζεται, τα πράγματα δυσκολεύουν.

Για την ολοκλήρωση της φυσική σχεδίασης μένουν να γίνουν με την σειρά που αναφέρονται, το τελικό power routing, το γενικό global detail routing και το finalization του chip με την εισαγωγή fillers.

4.3.8 Διασύνδεση τροφοδοσίας και γείωσης

Στο chip είναι ήδη έτοιμο το power plan, το μόνο που μένει είναι να δημιουργηθούν οι καλωδιώσεις που θα μεταφέρουν ισχύ από τις υπάρχουσες δομές(δακτυλίους, λωρίδες) στα κύτταρα του πυρήνα. Τα κύτταρα στον πυρήνα είναι τοποθετημένα σε τροχιές ή ράγες. Μία τεχνική που χρησιμοποιείται σχεδόν πάντα για την εξοικονόμηση χώρου, είναι οι σειρές των κυττάρων να εναλλάσσονται ώς προς τον άξονα των x. Αυτό διότι σαν κανόνας σε οποιοδήποτε κύτταρο, η τροφοδοσία και γείωση έχουν προκαθορισμένες θέσεις αγωγών, στο πάνω και στο κάτω μέρος του κυττάρου. Αν τοποθετηθούν όλες οι σειρές κανονικά, τότε θα πρέπει να διατηρηθεί μία ελάχιστη απόσταση μεταξύ δύο σειρών και συνεπώς θα χαθεί χώρος. Στην εικόνα φαίνεται αυτό διαισθητικά.



Σχήμα 15. Εναλλαγή σειρών κυττάρων πυρήνα

Με το power routing, αυτές οι ράγες αποκτούν αγωγούς τροφοδοσίας και γείωσης, μονού ή διπλού πάχους ανάλογα με το αν προορίζονται για μία ή για δύο σειρές, όπως φαίνεται πάνω στην εικόνα. Επίσης το power routing ενώνει τα pins τροφοδοσίας και γείωσης των έτοιμων blocks με τους δακτυλίους τους και τα pads τροφοδοσίας και γείωσης για τον πυρήνα, με τους δακτυλίους του πυρήνα.

Μετά από αυτό το βήμα έρχεται η δεύτερη και ουσιαστικά τελευταία διαδικασία της τοποθέτησης και διασύνδεσης· η καθολική λεπτομερής διασύνδεση(global detail routing).

4.3.9 Τελική διασύνδεση

Πρόκειται ίσως για το πιο υπολογιστικά ακριβό τμήμα της όλης διαδικασίας. Ευτυχώς το routing μπορεί να τρέξει multithreading, με ξεχωριστή άδεια να απαιτείται βέβαια ή ακόμα και superthreading, σε παραπάνω από έναν υπολογιστές.

Οι κλασσικοί αλγόριθμοι για διασύνδεση είναι αυτοί που ελαχιστοποιούν το συνολικό μήκος καλωδίωσης. Οι νεότεροι όμως και αυτοί που πλέον χρησιμοποιούνται είναι αυτοί που βελτιώνουν τον χρονισμό. Υπάρχουν αρκετές αλγοριθμικές προσεγγίσεις για το πρόβλημα του routing. Οι ισχυρότεροι routers τα τελευταία χρόνια είναι οι channel routers. Χονδρικά, αυτό που κάνουν είναι να χωρίζουν το κύκλωμα σε υποπεριοχές(channels) και να διασυνδέουν αυτές της περιοχές. Στο τέλος, θα έχει ολοκληρωθεί η διασύνδεση. Το κάθε επίπεδο μετάλλου μπορεί να διασυνδεθεί μόνο κάθετα ή οριζόντια, για την ελαχιστοποίηση της χωριτικότητας.

Η διασύνδεση, ειδικά στα μεγάλα κυκλώματα, θα γίνει κατά κανόνα περισσότερες από μία φορές. Αυτό διότι μετά το πρώτο routing το κύκλωμα θα έχει κάποια violations, τα οποία για να εξαληφθούν πρέπει να διαγραφούν και να ξανατρέξει ο router.

Στο encounter ενσωματώνεται ένας από τους πιο ισχυρούς routers στη βιομηχανία, ο nanoroute. Επίσης διατηρείται και ο παλιός του προηγούμενου εργαλείου, ο wroute, επειδή υπάρχουν ακόμα κάποια σχέδια που δίνει καλύτερα αποτελέσματα. Υπάρχουν πάρα πολλές επιλογές που μπορεί κανείς να κάνει πριν την εκκίνηση της διαδικασίας που μπορούν να φέρουν πιο κοντά στις ανάγκες του εκάστοτε σχεδίου, τη διασύνδεση. Για την διόρθωση των violations μετά τη διαγραφή τους, υπάρχει και η ελαφριά έκδοση του nanoroute, η eco. Μπορεί να ρυθμιστεί η ισορροπία μεταξύ χρονο-οδηγούμενων χαρακτηριστικών του αλγορίθμου και συμφόρησης στον πυρήνα, μπορεί να επιλεγεί ο τρόπος της αντιμετώπισης του φαινομένου antenna, ο αριθμός των επεξεργαστών που θα εκτελέσουν το routing, ο γράφος που θα χρησιμοποιηθεί για χρονική ανάλυση κ.α. Όταν όλες οι παράμετροι ρυθμιστούν μπορεί να ξεκινήσει το global detail routing.

4.3.10 Τελείωμα του chip

Μόλις ολοκληρωθεί η διασύνδεση με επιτυχία και δεν υπάρχουν άλλα violations, μπορεί να γίνει το λεγόμενο finalization του chip. Αυτό γίνεται με την εισαγωγή ειδικών κυττάρων της βιβλιοθήκης που γεμίζουν κάθε κενό στον πυρήνα. Ο κύριος λόγος που γίνεται αυτή η διαδικασία είναι η προστασία από το φαινόμενο latch up.

Το latch up, είναι το φαινόμενο κατά το οποίο η τροφοδοσία βραχυκυκλώνεται με τη γείωση. Αυτό γίνεται όταν κάποια ανεπιθύμητα διπολικά τρανζίστορ που σχηματίζονται μεταξύ του υποστρώματος και των συστατικών των τρανζίστορ, ξεκινούν να άγουν λόγω κάποιου δυνατού και απότομου ρεύματος. Αν αρχίσουν να άγουν τότε δεν υπάρχει επιστροφή καθώς λόγω

πόλωσης δεν μπορούν να σταματήσουν, συνεπώς το κύκλωμα καταστρέφεται.

Η λύση σε αυτό το φαινόμενο, είναι η εισαγωγή fillers. Επίσης σε κάθε κύτταρο υπάρχουν λεγόμενοι δακτύλιοι προστασίας που ο ρόλος τους είναι να έλκουν οποιαδήποτε δυνατά ρεύματα ώστε να μην ξεκινήσουν να άγουν τα διπολικά τρανζίστορ.

Οι fillers, όπως και στην περίπτωση του padframe, πρέπει να εισάγονται από τον μεγαλύτερο προς τον μικρότερο. Μετά από αυτή τη διαδικασία μπορεί να γραφτεί το τελικό netlist, το τελικό sdf και να εξαχθεί το αρχείο GDSII για την κατασκευή του κυκλώματος.

4.4 Πιστοποίηση

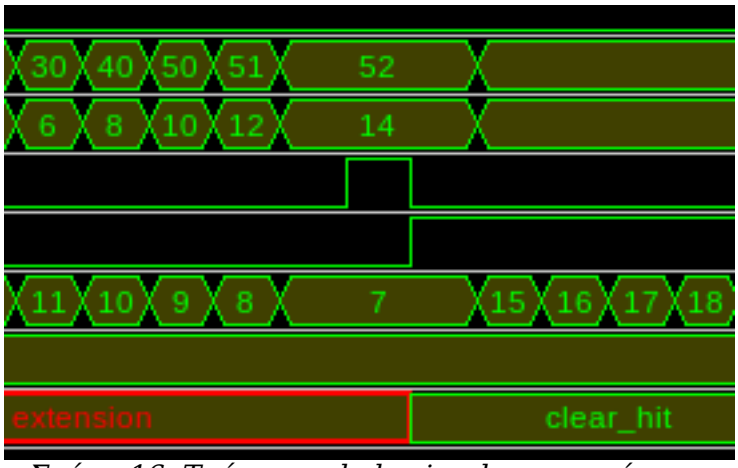
Ο σχεδιαστής πρέπει να έχει φροντίσει από πολύ πριν στείλει το σχέδιο για κατασκευή να υπάρχει τρόπος να ελεγχθεί η κατασκευαστική του ακεραιότητα. Για να γίνει αυτό πρέπει ιδανικά να ελεγχθεί κάθε κόμβος αν μπορεί να γίνει λογικό 1 και λογικό 0.

Οι μεγάλες εταιρίες παραγωγής κυκλωμάτων υιοθετούν ποικίλους τρόπους επαλήθευσης των κυκλωμάτων. Ο πρώτος είναι η μετατροπή των flip flop του κυκλώματος σε έναν ειδικό τύπο, τα scan flip flops. Με τη χρήση ενός ειδικού εξωτερικού σήματος είναι δυνατό να ελεγχθεί η έξοδος αυτών των flip flop και συνεπώς η τάση του κάθε κόμβου. Οι ακολουθίες εισόδου αυτού του ειδικού σήματος λέγονται scan chains. Στο στάδιο της λογικής σύνθεσης μπορούν να μετατραπούν τα flip flops του κυκλώματος σε scan. Επίσης υπάρχουν εργαλεία που δημιουργούν testbenches για την επαλήθευση το κυκλώματος. Ο δεύτερος τρόπος επαλήθευσης και τελευταίος, είναι η πραγματική δοκιμή του chip στην εφαρμογή για την οποία σχεδιάστηκε. Ακόμα και μετά από αυτό όμως το στάδιο μπορεί ένα chip να αποτύχει. Στην παρούσα εργασία δεν λήφθηκε υπόψη η κατασκευαστική δοκιμή, με το σκεπτικό ότι το chip δεν επρόκειτο να κατασκευαστεί.

Εκτός από την κατασκευαστική δοκιμή πρέπει να γίνουν και simulations, για την επαλήθευση της λειτουργικότητας. Simulation μπορεί να γίνει σχεδόν σε κάθε βήμα της σχεδιαστικής ροής. Πριν το στάδιο της σύνθεσης, γίνεται το behavioral simulation, η επαλήθευση δηλαδή της ορθής συμπεριφοράς του κώδικα. Μετά την σύνθεση μπορεί να γίνει ένα πιο ακριβές simulation, το gate level simulation. Τελευταίο στάδιο, είναι το simulation του τελικού netlist μετά το place and route, το οποίο λέγεται post place and route simulaton. Τα δύο τελευταία είδη, χρειάζονται και το αρχείο sdf, το οποίο περιέχει τις καθυστερήσεις των συνδέσεων.

Σαν γενικός κανόνας, πρέπει να γράφονται πλήρη testbenches, όσο μικρά γίνεται, χωρίς όμως να θυσιάζονται σημαντικές περιπτώσεις. Επίσης επειδή σε μεγάλα κυκλώματα η επαλήθευση βλέποντας της κυματομορφές δυσχεραίνει, καλό είναι να υπάρχουν μικρά scripts, που συγκρίνουν την έξοδο του testbench με την σωστή και απλά δίνουν μια απάντηση, επιτυχία ή αποτυχία.

Δυστυχώς μόνο behavioral simulation μπόρεσε να γίνει σε αυτήν την εργασία, καθώς τόσο με την πρώτη όσο και με την δεύτερη τεχνολογία υλοποίησης ο προσομοιωτής έβγαζε λάθος. Ενώ με την HDL περιγραφή και το ίδιο ακριβώς testbench το behavioral simulation ολοκληρωνόταν επιτυχώς, όταν απλά φορτωνόντουσαν τα simulation models της κάθε τεχνολογίας, το εργαλείο εμφάνιζε λάθη. Δοκιμάστηκαν διάφορες λύσεις από το διαδίκτυο χωρίς όμως αποτέλεσμα. Το μήνυμα λάθους πληροφορούσε για κάποιες παραμέτρους που λείπανε από τα simulation models. Ακολουθεί τμήμα της κυματομορφής από το behavioral simulation.



Σχήμα 16. Τμήμα της behavioral προσομοίωσης

Επιλέχθηκε μόνο αυτό το τμήμα της κυματομορφής, καθώς είναι το λεγόμενο σημείο pass-fail, όπως θα μπορούσε να χαρακτηριστεί. Στην εικόνα φαίνονται οι καταστάσεις του control, ο αριθμός 52 είναι το score του hit, το 14 είναι το μέγεθος του, το σήμα ακριβώς από κάτω είναι η έξοδος hit και ο αριθμός 7 είναι η διεύθυνση του hit, ο έβδομος χαρακτήρας από την έναρξη της εισόδου δηλαδή. Το σημαντικό είναι ότι οι μνήμες λειτούργησαν εντόπισαν την στοιχειώδη ευστοχία, ευθυγραμμίστηκε με επιτυχία ο barrel shifter με τη ροή εισόδου, η επέκταση λειτούργησε(το query περιελάμβανε όλους τους δυνατούς χαρακτήρες), υπολογίστηκε το σωστό αποτέλεσμα και το κύκλωμα αφού προσπέρασε την ευστοχία επέστρεψε στο δεύτερο βήμα του αλγορίθμου. Όπως είναι φανερό και από την εικόνα μόνο όταν το σήμα hit είναι ενεργό, οι εξόδοι περιέχουν χρήσιμες πληροφορίες. Το σήμα πάνω από τον αριθμό 7 είναι η έξοδος ready που υποδηλώνει πως το κύκλωμα μπορεί και πάλι να δεχτεί χαρακτήρες.

4.5 Γενικές πληροφορίες για τις δύο υλοποιήσεις

Συγκριτικά, τα βασικά στοιχεία των δύο chip παρουσιάζονται στον παρακάτω πίνακα.

Πίνακας 4. Βασικά στοιχεία των δύο υλοποιήσεων

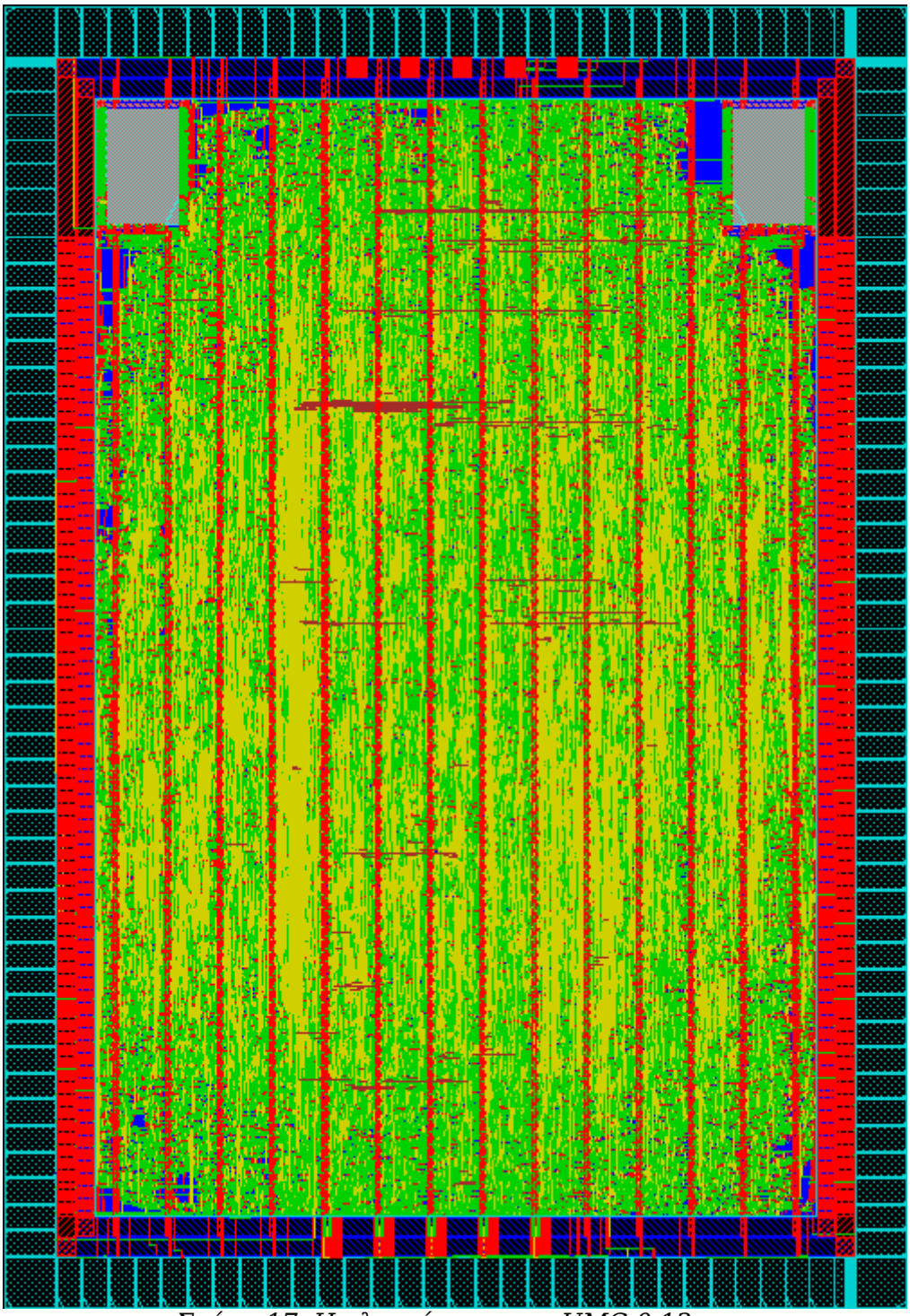
Βασικά στοιχεία	*Τεχνολογία υλοποίησης(foundry)*	
	UMC	Faraday/UMC
Ελάχιστο μέγεθος πύλης	0,13μm	90nm
Επίπεδα μετάλλων	8	9
Χρονισμός	300MHz	320MHz
Διαστάσεις σε μm	3775x2517	2496x1894
Κατανάλωση σε mW	120	240
Τάση τροφοδοσίας πυρήνα	1.2V	1V
Τάση τροφοδοσίας IO	3.3V	2.5V
Υπολογιζόμενο κόστος κατασκευής	$22,500.00	$25,000.00
Αριθμός κυττάρων πυρήνα	100000	92500
Αριθμός pads	133	149

Από τον παραπάνω πίνακα αίσθηση προκαλεί κυρίως ένα μέγεθος. Η κατανάλωση του chip με την Faraday τεχνολογία. Πρόκειται για μία standard performance βιβλιοθήκη όπως είναι και της UMC, οπότε θα περίμενε κανείς αφού είναι σε μικρότερη τεχνολογία υλοποίησης, να έχει μικρότερη κατανάλωση, πόσο μάλλον αφού κι ο χρονισμός είναι κοντά. Η κατανάλωση προέκυψε κατευθείαν μέσα από το encounter. Επίσης, η πρώτη έκδοση έχει περισσότερα κύτταρα στον πυρήνα λόγω εντονότερου buffering για την επίτευξη του χρονισμού. Η δεύτερη έχει περισσότερα pads, λόγω αυξημένη κατανάλωσης.

Οι τιμές που δίνονται εδώ για fabrication, είναι καθαρά ενδεικτικές, οι πραγματικές για το Πολυτεχνείο Κρήτης, μπορεί να απέχουν πάρα πολύ. Στο site της MTC online [28], αναφέρεται ότι λίγα αντίτυπα για ακαδημαϊκά ιδρύματα μη κερδοσκοπικού χαρακτήρα, είναι δωρεάν. Γενικά η Faraday τεχνολογία διατίθεται σε ανταγωνιστικές τιμές και τα διαθέσιμα design kits, σύμφωνα με την αντίστοιχη ιστοσελίδα [29] είναι δωρεάν σε όποιον αποκτήσει λογαριασμό, είτε δωρεάν, είτε επί πληρωμή.

Όσον αφορά τον χρονισμό των chip, κύριος παράγοντας που έθεσε άνω όριο είναι η ταχύτητα των μνημών. Εκτός από αυτό όμως δεν θα ήταν δυνατή η εντυπωσιακή τουλάχιστον αύξηση του χρονισμού καθώς και τα pads των δύο

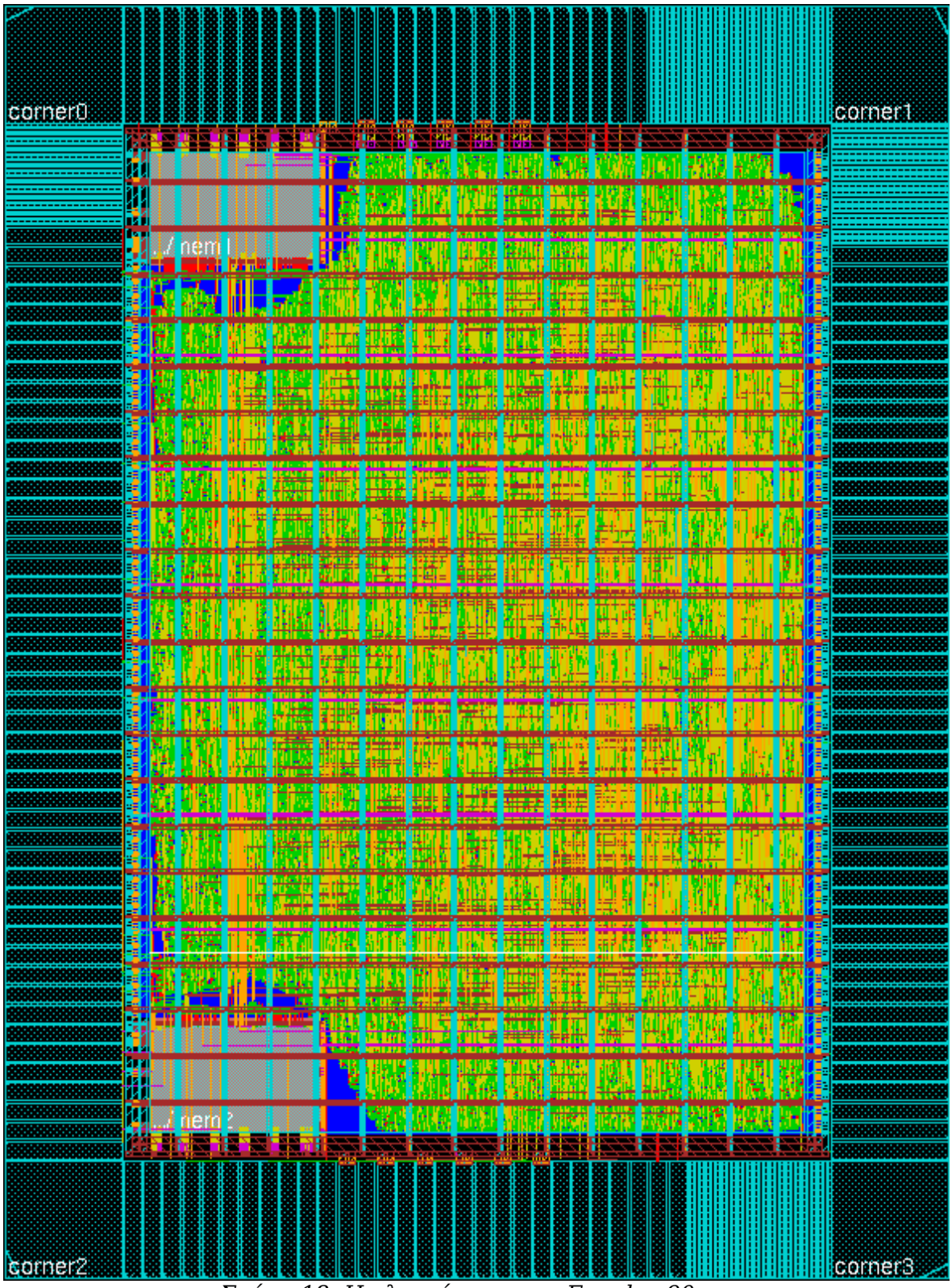
τεχνολογιών εισήγαγαν διόλου αμελητέες καθυστερήσεις. Ακολουθούν οι τελικές εικόνες των δύο υλοποιήσεων, πρώτα της UMC και ύστερα της Faraday. Αξίζει να σημειωθεί το διαφορετικό power plan της κάθε έκδοσης καθώς και η διαφορετική τοποθέτηση μνημών. Επίσης υπενθυμίζεται ότι η πρώτη υλοποίηση έχει in-line pads, ενώ η δεύτερη staggered.



Σχήμα 17. Η υλοποίηση με τη UMC 0,13μm

Αυτή είναι η τελική εικόνα του chip με την πρώτη τεχνολογία. Το power routing εδώ, έχει γίνει με το χέρι. Ορατές είναι κάποιες περιοχές του πυρήνα

κοντά στις μνήμες που παρέμειναν άδειες, ακόμα κι αν η χρήση του πυρήνα είναι στο 72% και οι ρυθμίσεις της τοποθέτησης είναι για μέγιστη μείωση του συνωστισμού. Ίσως μία χρήση 75%-76% να αξιοποιούσε καλύτερα τον πυρήνα. Ακολουθεί η εικόνα της δεύτερης υλοποίησης με την Faraday 90nm από το Europractice.



Σχήμα 18. Η υλοποίηση με τη Faraday 90nm

Συγκρίνοντας τις δύο υλοποιήσεις αμέσως φαίνεται η διαφορά μεγέθους στον πυρήνα. Διαφορά επίσης έχει και το πιο ώριμο power routing αυτής της έκδοσης που χρησιμοποιεί αποτελεσματικότερα τα διαθέσιμα μέταλλα της τεχνολογίας αφήνοντας χώρο για καλύτερο routing, χωρίς να θυσιάζει ασφάλεια.

5. Επίλογος

Η παρούσα διπλωματική ανήκει στην πρώτη γενιά τέτοιου είδους εργασιών που ανατίθενται στο Πολυτεχνείο Κρήτης. Είναι λογικό λοιπόν να συναντήθηκαν αρκετά προβλήματα, κυρίως χρήσης των εργαλείων. Επειδή τα EDA είναι κάπως εξεζητημένος τομέας, η εύρεση πληροφοριών είναι αρκετά πιο δύσκολη από ότι για άλλου είδους υλοποιήσεις, όπως είναι σε FPGA ή σε software. Αυτή ακριβώς η δυσκολία της εύρεσης επαρκών πληροφοριών για την υλοποίηση σε συνδυασμό με το γεγονός ότι δεν είχε ξαναγίνει κάτι παρόμοιο στο Πολυτεχνείο Κρήτης, χαρακτήρισαν αυτή τη διπλωματική.

Δυστυχώς τα EDA δεν είναι εύκολο πράγμα για αρχάριους σχεδιαστές. Ο κυριότερος όμως λόγος που αυτό ισχύει είναι ότι αφήνουν αρκετό έλεγχο στον σχεδιαστή, που εύκολα μπορεί να χαθεί μέσα στην πληθώρα των πληροφοριών και επιλογών με αποτέλεσμα να γίνει έρμαιο της απειρίας του. Δεν θα μπορούσε να γίνει αλλιώς όμως, καθώς το προϊόν που παράγουν αυτά τα εργαλεία, στοιχίζει πάρα πολλά χρήματα και είναι λογικό και επιθυμητό για μια επένδυση τέτοιου μεγέθους να έχει ο μηχανικός τον πρώτο λόγο. Η κύρια γνώση πάνω στα εργαλεία αποκτήθηκε σε ένα σεμινάριο για EDA που πραγματοποιήθηκε στο Ίδρυμα Τεχνολογίας και Έρευνας(ΙΤΕ) στο Ηράκλειο [30] το Σεπτέμβριο του 2008.

5.1 Σύγκριση με εναλλακτικές αρχιτεκτονικές

Για να είναι επιτυχημένη μία σύγκριση μεταξύ υλοποιήσεων, πρέπει να γίνει με βάση το έργο που διεκπεραιώθηκε σε ένα δεδομένο χρόνο για την ίδια είσοδο. Στη συγκεκριμένη περίπτωση αυτό δεν είναι εφικτό επειδή το σχεδιασμένο chip, δεν έχει υλοποιηθεί. Τα υπόλοιπα χαρακτηριστικά, όπως είναι η συχνότητα του ρολογιού, δεν αποτελούν αξιόπιστο μέτρο σύγκρισης.

Έτσι λοιπόν, μόνο υποθέσεις μπορούν να γίνουν σε αυτόν τον τομέα. Η πρώτη σύγκριση είναι πολύ λογικό να γίνει με την ήδη υπάρχουσα αρχιτεκτονική του BLAST για FPGA, του Πολυτεχνείου Κρήτης. Η αρχιτεκτονική της παρούσας εργασίας, βασίστηκε σε αυτή σε πολύ μεγάλο βαθμό. Είναι όμως δύο διαφορετικές αρχιτεκτονικές, καθώς αυτή για FPGA υποστηρίζει όλες τις εκδόσεις του BLAST για queries μέχρι και 200000 χαρακτήρες, ενώ η έκδοση για VLSI υποστηρίζει queries, μέχρι 10000 χαρακτήρες και εκτελεί μόνο το BLASTn.

Σίγουρα λοιπόν δεν μπορεί να γίνει πλήρης σύγκριση. Αν όμως ληφθούν υπόψη μόνο τα κοινώς υποστηριζόμενα στοιχεία, τότε αυτό αλλάζει. Οι περιοχή που είναι εφικτή η σύγκριση, είναι στο BLASTn, σε queries μέχρι και 10000

χαρακτήρες.

Οι δύο αρχιτεκτονικές αντιμετωπίζουν με διαφορετικό τρόπο το πρώτο βήμα του αλγορίθμου, αλλά αυτό δεν έχει σημασία, αφού εκτελείται μόνο μία φορά και δεν επηρεάζει το throghput. Ο κοινός παρονομαστής είναι το δεύτερο βήμα, που είναι και αυτό που εκτελείται στατιστικά περισσότερο, ενώ το τρίτο βήμα διαφέρει. Η αρχιτεκτονική για FPGA το αφήνει στο software, ενώ στην περίπτωση του VLSI εκτελείται σε hardware. Αυτό δίνει ένα προβάδισμα στο VLSI, αλλά θα μπορούσε κανείς να ισχυριστεί πως αν συσσωρευτούν μαζί πολλές ευστοχίες, στην πρώτη περίπτωση μπαίνουν σε μία fifo για υπολογισμό του score και το κύκλωμα συνεχίζει τη λειτουργία του, ενώ στη δεύτερη το κύκλωμα σταματά για τον υπολογισμό και συνεχίζει μετά το πέρας του. Παρόλα αυτά όμως ένας υπολογισμός σε software, είναι κατά πολύ αργότερος από τον αντίστοιχο σε hardware και μάλιστα σε ολοκληρωμένο. Επίσης η πλακέτα στόχος είναι στα 90nm υλοποιημένη όπως και η δεύτερη παραλλαγή για VLSI.

Λαμβάνοντας υπόψη τα παραπάνω και διαπιστώνοντας πως οι δύο αρχιτεκτονικές εκτελούν τις ίδιες εργασίες σε κάθε κύκλο, θα μπορούσε να ειπωθεί πως η αρχιτεκτονική για VLSI, είναι 2,3 φορές ταχύτερη από την αντίστοιχη για FPGA. Με βάση αυτήν την υπόθεση και λαμβάνοντας υπόψη το [10], προκύπτει ο παρακάτω πίνακας. Τονίζεται ξανά όμως ότι αποτελεί υπόθεση.

Πίνακας 5. Σύγκριση throughput(10^6 χαρακτήρες ανά δευτερόλεπτο)

Μέγεθος query	*Υλοποίηση*				
	TUC VLSI (εκτίμηση)	TUC FPGA	IBM multiprocessor	IBM single chip	Software, Pentium 4 @ 3,00GHz
1000	320	103	1201	187	588
2000					409
5000			159	14	218
10000		51,5			129

5.2 Παρατηρήσεις για βελτιώσεις στο μέλλον

Η πρώτη κίνηση για να μπορέσουν να παραχθούν βιώσιμα chip, είναι η εύρεση μιας τεχνολογίας που να υποστηρίζεται πλήρως από το SOC Encounter και μάλιστα από την εκάστοτε έκδοση που χρησιμοποιείται. Καμία από τις δύο τεχνολογίες που χρησιμοποιήθηκαν σε αυτήν την εργασία δεν υποστηρίζεται πλήρως από το εργαλείο. Σαν αποτέλεσμα αυτού του γεγονότος κανένα από τα δύο chip δεν θα μπορούσε να θεωρηθεί τελειωμένο.

ΚΕΦΑΛΑΙΟ 5 - ΕΠΙΛΟΓΟΣ

Από την αρχή της σχεδίασης αντιμετωπίστηκαν πολλά προβλήματα με τις βιβλιοθήκες, κάποια λύθηκαν γιατί ήταν εφικτό, κάποια άλλα όμως δεν λύθηκαν. Για παράδειγμα, με την UMC, χρειάστηκε να αλλαχτούν σε βάθος κάποια αρχεία lef των μνημών, μόνο και μόνο για να γίνει power routing. Παρόλα αυτά η διαδικασία και πάλι δεν προχώρησε γιατί μετά την τελική διασύνδεση παρέμεναν πολλά violations που οφείλονταν σε ασυμβατότητες της έκδοσης των lef αρχείων με το encounter, καθώς η τεχνολογία προοριζόταν για χρήση με το Silicon Ensemble.

Η Faraday από την άλλη μεριά εμφάνιζε εξαιρετική εικόνα μέχρι που το σχέδιο απέκτησε pads. Ενώ η βιβλιοθήκη του πυρήνα μπορούσε να συνθέσει κύκλωμα χωρίς κανένα πρόβλημα, μόλις έγινε το padframe, το σχέδιο δεν μπορούσε να κάνει την τελική διασύνδεση με τον nanoroute. Το εργαλείο εμφάνιζε μήνυμα πως δεν μπορεί να βρει κάποια pins των pads στα αρχεία lef. Δοκιμάστηκε εναλλακτικά ο wroute, αλλά το αποτέλεσμα που έδινε είχε λάθη στην περιοχή των pads. Συγκεκριμένα δεν υπήρχε επαφή πάνω από τα pins εξόδου των input pads και το μέταλλο σταματούσε αρκετά μακριά. Συνεπώς για να μπορέσουν να παραχθούν βιώσιμα chips, πρέπει να βρεθεί μία συμβατή τεχνολογία με το encounter, ή εναλλακτικά να χρησιμοποιηθεί το Astro της Synopsys.

Όσον αφορά το σχεδιαστικό τμήμα αυτής της εργασίας μια μελλοντική επέκταση της αρχιτεκτονικής για την υποστήριξη όλων των εκδόσεων του BLAST θα ήταν καλή ιδέα. Η υποστήριξη μεγαλύτερων queries αποτελεί μάλλον δευτερεύουσα επέκταση. Αυτό διότι το τρέχον υποστηριζόμενο μέγεθος των 10000 χαρακτήρων, καλύπτει πάνω από το 90% των περιπτώσεων, τη στιγμή που το 70% των queries είναι μέχρι 1000 χαρακτήρες.

6. Βιβλιογραφία

Δημοσιεύσεις

1. L. Conway and C. Mead, "Introduction to VLSI systems", Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA
2. P. G. Higgs and T. K. Attwood, "Bioinformatics And Molecular Evolution", Blackwell, 2005.
3. S. Altschul et al., "Basic Local Alignment Search Tool", J. Mol. Biol., vol. 215, 1990, pp. 403–410.
4. J. Lancaster, J. Buhler, R. Chamberlain, "Acceleration of Ungraphed Extension in Mercury BLAST", in 7th workshop on media and streaming processors, Barcelona, Spain, November 12, 2005.
5. S. B. Needleman and C. D. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins", J. Mol. Biol., vol. 48, 1970, pp. 443–453.
6. C. Chang, "BLAST Implementation on BEE2", University of California at Berkeley, California, 2004.
7. E. Sotiriades, C. Kozanitis and A. Dollas, "FPGA based Architecture of DNA Sequence Comparison and Database Search", in Proceedings 20th International Parallel and Distributed Processing Symposium, IPDPS 2006, p. 193., at the 13th Reconfigurable Architectures Workshop (RAW 2006), Rhodes, Greece, 25–29 April, 2006
8. E. Sotiriades, C. Kozanitis and A. Dollas, “Some Initial Results on Hardware BLAST Acceleration with a Reconfigurable Architecture”, in Proceedings 20th International Parallel and Distributed Processing Symposium, IPDPS 2006, p. 251, at the 5th IEEE International Workshop on High Performance Computational Biology (HiCOMB2006), Rhodes, Greece, 25–29 April, 2006.
9. E. Sotiriades, C. Kozanitis, G. Chrysos and A. Dollas, "Rapid Phototyping of a System-on-a-Chip for the BLAST Algorithm Implementation", in Proceedings, 17th International IEEE Workshop on Rapid System Prototyping RSP-2006, pp. 223– 229, Computer Society, Chania, Greece, 14–16 June, 2006.
10. Euripides Sotiriades and Apostolos Dollas, “A General Reconfigurable Architecture for the BLAST Algorithm”, Journal of VLSI Signal Processing 48, 189–208, 2007

Σελίδες στο διαδίκτυο

11. http://www.ncbi.nlm.nih.gov/
12. http://www.ncbi.nlm.nih.gov/blast/megablast.shtml
13. http://en.wikipedia.org/wiki/Carver_Mead
14. http://www.cs.caltech.edu/cspeople/faculty/mead_c.html
15. http://en.wikipedia.org/wiki/Electronic_design_automation
16. http://en.wikipedia.org/wiki/GDSII
17. http://en.wikipedia.org/wiki/Calma
18. http://www.csd.uoc.gr/~hy523/http://ai.eecs.umich.edu/people/conway/
19. http://www.mentor.com/
20. http://www.magma.com/
21. http://www.synopsys.com/home.aspx
22. http://www.cadence.com/us/pages/default.aspx
23. http://www.gerberscientific.com/about_gerber/history.htm
24. http://en.wikipedia.org/wiki/Logic_synthesis
25. http://en.wikipedia.org/wiki/Place_and_route
26. http://www.csl.mete.metu.edu.tr/Electromigration/emig.htm
27. http://en.wikipedia.org/wiki/Antenna_effect
28. http://www.mtc-online.be/index.asp
29. http://freelibrary.faraday-tech.com/AIPMain.php
30. http://www.forth.gr/index.php?l=g

Λίστα Σχημάτων

Λίστα Πινάκων

Παράρτημα Α - Εγκατάσταση των εργαλείων

Η πλειοψηφία των εργαλείων προορίζεται για Red Hat Enteriprise Linux και Solaris workstations. Κάποια έχουν σαν επιλογή και τα MS Windows, αλλά κατά κοινή παραδοχή καλό είναι να αποφεύγεται αυτός ο συνδυασμός. Τμήμα αυτής της εργασίας ήταν και η προετοιμασία ενός υπολογιστή για χρήση με EDA.

Τα διαθέσιμα εργαλεία για σύνθεση και place and route, είναι αντίστοιχα το Design Vision της Synopsys και το First Encounter της Cadence. Πρόκειται για δύο από τα πιο αναγνωρισμένα εργαλεία στο χώρο τους το καθένα. Αρχικά έγινε προσπάθεια να στηθούν σε λειτουργικό Ubuntu Linux, κυρίως λόγο οικειότητας με το λειτουργικό αλλά και ελεύθερης διάθεσης. Γρήγορα όμως έγινε αντιληπτό ότι το Ubuntu είναι μια από τις πλέον ακατάλληλες διανομές. Το ενδιαφέρον ύστερα στράφηκε σε ελεύθερα λειτουργικά συναφή με τα Red Hat, όπου και βρέθηκαν τα CentOS, που είναι φτιαγμένα, όπως δηλώνεται στην ιστοσελίδα τους, από τα πακέτα του Red Hat Enterprise Linux της αντίστοιχης έκδοσης κάθε φορά. Πράγματι η εγκατάσταση των εργαλείων έγινε απόλυτα εφικτή και ολοκληρώθηκε σε μία ώρα, τη στιγμή που στα Ubuntu έχει κρατήσει δέκα ημέρες και χωρίς αποτέλεσμα. Το μόνο που χρειάστηκε ήταν κάποια πακέτα, για να τρέξει ο InstallScape, ο installer των εργαλείων της Cadence.

Για τα εργαλεία της Cadence η γενική τακτική είναι να αντιγράψουμε ότι media έχουμε στο δίσκο του υπολογιστή και ύστερα να τρέξουμε τον installer από τα update media, αν έχουμε, αλλιώς από τα baseline της αντίστοιχης έκδοσης. Ο installer θα πρέπει να τρέξει με δικαιώματα super user, από τα αντίγραφα των media στο δίσκο. Από αυτό το σημείο και ύστερα η εγκατάσταση είναι αρκετά εύκολη και κατανοητή. Το synopsys εγκαθίσταται χωρίς προβλήματα ακόμα και σε ubuntu με τη διαδικασία να είναι αρκετά απλή.

Τα δύο αυτά εργαλεία τρέχουν παίρνοντας άδεια από κάποιο license server με χρήση license daemon. Συνεπώς είναι πολύ σημαντικό να υπάρχουν οι άδειες και να τρέχουν σε κάποιο μηχάνημα με γνωστή IP, που έχουμε πρόσβαση και μπορούμε να κατευθύνουμε τα εργαλεία με τις κατάλληλες μεταβλητές περιβάλλοντος, ώστε να μπορούμε να τα χρησιμοποιήσουμε. Οι άδειες συνήθως είναι για έναν υπολογιστή, κάτι που σημαίνει πως δεν μπορούν ταυτόχρονα δύο υπολογιστές να κάνουν χρήση του ίδιου εργαλείου.

Για να μπορέσουμε να τρέξουμε οποιοδήποτε από τα δύο αυτά εργαλεία, επιβάλλεται να φορτώσουμε πριν τις σωστές μεταβλητές περιβάλλοντος στο shell που θα χρησιμοποιήσουμε, στη συγκεκριμένη περίπτωση το csh. Αυτό γίνεται με την χρήση ενός cshrc αρχείου, που είναι ξεχωριστό για την κάθε εφαρμογή, το οποίο τρέχουμε στο csh shell, πριν ξεκινήσουμε τα εργαλεία με την

εντολή source. Για το encounter το αρχείο αυτό βρίσκεται στο φάκελο bin, του κεντρικού φακέλου της εγκατάστασης. Το μόνο που χρειάζεται να γίνει, είναι να ανοιχτεί το αρχείο και να συμπληρωθεί η IP του licence server και το installation root directory. Επίσης πρέπει να δημιουργηθεί ένα symbolic link με το όνομα tools στον κεντρικό φάκελο της εγκατάστασης, το οποίο θα παραπέμπει στο φάκελο tools.xxxxx που είναι επίσης στον κεντρικό φάκελο της εγκατάστασης. Το xxxxx είναι ο κωδικός της αρχιτεκτονικής για την οποία προορίζεται η συγκεκριμένη έκδοση του εργαλείου. Για το synopsys πρέπει να αντιγράψουμε στον φάκελο admin μέσα στην εγκατάσταση ένα αρχείο κειμένου που έχει να κάνει με την άδεια και να δημιουργήσουμε ένα cshrc που θα κάνουμε source και θα περιέχει τις ακόλουθες γραμμές:

```
setenv SYNOPSYS here_goes_installation_root_directory
set path=($SYNOPSYS/linux/syn/bin $path)
```

Εν κατακλείδι για να τρέξουμε τα εργαλεία πληκτρολογούμε:

Για το encounter της cadence:

```
csh
source installation_root_directory_here/bin/CSHRC
encounter
```

Για το design vision της synopsys:

```
csh
source directory_containing_our_cshrc/cshrc
design_vision
```

Όλα τα παραπάνω ισχύουν βέβαια στην περίπτωση που δουλεύουμε σε λειτουργικό linux. Τελευταίο εργαλείο, για το οποίο δεν έχει γίνει ακόμα λόγος είναι ο προσομοιωτής, το nclaunch. Για να το εγκαταστήσουμε, χρησιμοποιούμε τον ίδιο installer και την ίδια ακριβώς τεχνική με το encounter(πρέπει και εδώ να δημιουργηθεί symbolic link tools). Το nclaunch χρειάζεται κι αυτό το δικό του cshrc το οποίο δίνεται παρακάτω.

```
setenv IUS_ROOT installation_root_directory
set path = ($path $IUS_ROOT/tools/bin)
setenv CDS_LIC_FILE 5280@licence_server_ip
```

Παράρτημα Β - Scripts για σύνθεση

Δίνονται τα scripts που έγινε λογική σύνθεση με το Design Vision και τον RTL compiler. Επίσης δύνεται και το αρχείο .synth_init, που χρειάζεται ο τελευταίος για τον ορισμό των βιβλιοθηκών κ.α.

```
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/dflipflop.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/mux2to1.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/pipe_500.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/pipe_500_1.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/shift_reg_database.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/valid_bits_reg.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/barrel_shifter_query.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/left_reg.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/right_reg.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/comparisons_reg.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/is_equal.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/comparators.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/data_unit.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/score_calculator.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/buffer_in.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/wmers_unit.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/counter9.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/upcount_4.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/upcount_32.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/extension_unit.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/control.vhd}
read_file -format vhdl
{/home/mastrojohn/synthesis/final/codes/top_level_entity.vhd}

create_clock -name "clk" -period 3.34 -waveform { 0 1.67  }  { clk  }

set_drive 1 [all_inputs]
set_load 0.0005 [all_outputs]
set_input_delay 1.5 -clock {clk} [all_inputs]
```

```
set_output_delay 0.5 -clock {clk} [all_outputs]

set_operating_conditions -library umce13h210t3_wc_108V_125C WORST

current_design control
set_fsm_encoding_style one_hot

current_design extension_unit
set_fsm_encoding_style gray

current_design wmers_unit
set_fsm_encoding_style gray

current_design top_level_entity

compile

write_sdc sdc/sdc_file
write_sdf sdf/sdf_file
write -hierarchy -format verilog -output netlist/top_level_entity.v

quit
```

Το script για τον RTL compiler είναι το ακόλουθο:

```
read_hdl -vhdl shift_reg_database.vhd
read_hdl -vhdl valid_bits_reg.vhd
read_hdl -vhdl barrel_shifter_query.vhd
read_hdl -vhdl left_reg.vhd
read_hdl -vhdl right_reg.vhd
read_hdl -vhdl comparisons_reg.vhd
read_hdl -vhdl comparators.vhd
read_hdl -vhdl score_calculator.vhd
read_hdl -vhdl data_unit.vhd
read_hdl -vhdl buffer_in.vhd
read_hdl -vhdl wmers_unit.vhd
read_hdl -vhdl counter9.vhd
read_hdl -vhdl upcount_4.vhd
read_hdl -vhdl upcount_32.vhd
read_hdl -vhdl extension_unit.vhd
read_hdl -vhdl control.vhd
read_hdl -vhdl top_level_entity.vhd
elaborate

dc::create_clock [dc::get_ports clk]  -period 3  -waveform {0 1.5}
dc::set_input_delay 1.5 -clock {clk} [dc::all_inputs]
dc::set_output_delay 0.5 -clock {clk} [dc::all_outputs]
dc::set_load 0.001 [dc::all_outputs]
dc::set_drive 1.2 [dc::all_inputs]

synthesize -to_mapped

report timing > reports/chip_repo
report area >> reports/chip_repo
report power >> reports/chip_repo
write_sdc > sdc/chip_sdc
write_hdl > netlists/chip.v
write_sdf > sdf/chip_sdf
quit
```

ΠΑΡΑΡΤΗΜΑ Β – SCRIPTS ΓΙΑ ΣΥΝΘΕΣΗ

Το προαιρετικό αρχείο .synth_init που αρχικοποιεί τον RTL Compiler και τοποθετείται στον φάκελο .cadence στο home folder του οποιουδήποτε χρήστη έχει την παρακάτω ενδεικτική μορφή.

```
set_attribute lib_search_path {lib1_path lib2_path ..} /

set_attr hdl_search_path /home/mastrojohn/synthesis/final_rc/codes

set_attribute library { fsd0a_a_generic_core_1d08vwc.lib
SHAA90_512X32X1CM4_WC.lib} /
set_attribute lef_library { lef1_path lef2_path ... } / #προεραιτικό
set_attribute operating_conditions WCCOM
set_attr hdl_vhdl_read_version 1993
set_attribute hdl_max_loop_limit 10000 /
```

Παράρτημα Γ - Script για place and route

Ενδεικτικά δίνεται ένα script για place and route για ολόκληρη την διαδικασία, αν δεν χρησιμοποιούνται έτοιμα blocks και δεν υπάρχει padframe.

```
# Setup design and create floorplan
loadConfig ./example.conf
commitConfig

# Create Floorplan
floorplan -r 1.0 0.6 40.05 40.8 40.05 42

# Add supply rings around core
addRing -spacing_bottom 9.9 -width_left 9.9 -width_bottom 9.9
-width_top 9.9 -spacing_top 9.9 -layer_bottom metal1 -width_right 9.9
-around core -center 1 -layer_top metal1 -spacing_right 9.9
-spacing_left 9.9 -layer_right metal2 -layer_left metal2 -offset_top
9.9 -offset_bottom 9.9 -offset_left 9.9 -offset_right 9.9 -nets { gnd
vdd }

placeInstance $custom_cell  342 343 R0 -fixed

# Place standard cfells
amoebaPlace

# Route power nets
sroute -noBlockPins -noPadRings

# Perform trial route and get initial timing results
trialroute
buildTimingGraph
setCteReport
reportTA -nworst  10 -net > timing.rep.1.placed

# Run in-place optimization
# to fix setup problems
setIPOMode -mediumEffort -fixDRC -addPortAsNeeded
initECO ./ipo1.txt
fixSetupViolation
endECO
buildTimingGraph
setCteReport
reportTA -nworst  10 -net > timing.rep.2.ipo1

# Run Clock Tree Synthesis
createClockTreeSpec -output encounter.cts -bufFootprint buf
-invFootprint inv
specifyClockTree -clkfile encounter.cts
ckSynthesis -rguide cts.rguide -report report.ctsrpt -macromodel
report.ctsmdl -fix_added_buffers

# Output Results of CTS
trialRoute -highEffort -guide cts.rguide
extractRC
reportClockTree -postRoute -localSkew -report
```

```
skew.post_troute_local.ctsrpt
reportClockTree -postRoute -report report.post_troute.ctsrpt

# Run Post-CTS Timing analysis
setAnalysisMode -setup -async -skew -autoDetectClockTree
buildTimingGraph
setCteReport
reportTA -nworst  10 -net > timing.rep.3.cts

# Perform post-CTS IPO
setIPOMode -highEffort -fixDrc -addPortAsNeeded -incrTrialRoute
-restruct -topomap
initECO ipo2.txt
setExtractRCMode -default -assumeMetFill
extractRC
fixSetupViolation -guide cts.rguide

# Fix all remaining violations
setExtractRCMode -detail -assumeMetFill
extractRC
if {[isDRVClean -maxTran -maxCap -maxFanout] != 1} {
fixDRCViolation -maxTran -maxCap -maxFanout
}

endECO
cleanupECO

# Run Post IPO-2 timing analysis
buildTimingGraph
setCteReport
reportTA -nworst  10 -net > timing.rep.4.ipo2

# Add filler cells
addFiller -cell FILL -prefix FILL -fillBoundary

# Connect all new cells to VDD/GND
globalNetConnect vdd -type tiehi
globalNetConnect vdd -type pgpin -pin vdd -override

globalNetConnect gnd -type tielo
globalNetConnect gnd -type pgpin -pin gnd -override

# Run global Routing
globalDetailRoute

# Get final timing results
setExtractRCMode -detail -noReduce
extractRC
buildTimingGraph
setCteReport
reportTA -nworst  10 -net > timing.rep.5.final

# Output GDSII
streamOut final.gds -mapFile /a_path/example.map -stripes 1 -units
1000 -mode ALL
saveNetlist -excludeLeafCell final.v

# Output DSPF RC Data
rcout -spf final.dspf
```

```
# Run DRC and Connection checks
verifyGeometry
verifyConnectivity -type all
```