ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Ψηφιακή Σχεδίαση Παραμετροποιημένου Ψηφιακού Κυκλώματος "Game of Life" Σε Τεχνολογία 90nm.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Αντώνιου Σπυρόπουλου

Βόλος 2011

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Ψηφιακή Σχεδίαση Παραμετροποιημένου Κυκλώματος "Game of Life" Σε Τεχνολογία 90nm.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΝΤΩΝΙΟΥ ΣΠΥΡΟΠΟΥΛΟΥ

Επιβλέπων : Νέστωρ Ευμορφόπουλος
Λέκτορας Τ.Μ.Η.Υ.Τ.Δ

Εγκρίθηκε από την διμελή εξεταστική επιτροπή την 18η Οκτώβριου 2011.

(Υπογραφή)	*(Υπογραφή)*
.....	.....
Νέστωρ Ευμορφόπουλος	Γεώργιος Δημητρίου
Λέκτορας Παν. Θεσσαλίας	Διδάσκων Παν. Θεσσαλίας

Βόλος 2011

.....

ΑΝΤΩΝΙΟΣ ΣΠΥΡΟΠΟΥΛΟΣ

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων Πανεπιστημίου Θεσσαλίας

Περίληψη

Ο σκοπός αυτής της διπλωματικής εργασίας ήταν η μελέτη και ανάπτυξη μεθοδολογίας για την ροή σχεδίασης ενός ψηφιακού ολοκληρωμένου κυκλώματος ειδικού σκοπού (ASIC) με τεχνολογία CMOS VLSI και με την χρήση σύγχρονων αυτοματοποιημένων εργαλείων σχεδίασης.

Στην μελέτη αυτή χρησιμοποιήθηκε ως κύκλωμα το Game Of Life σε γλώσσα περιγραφής υλικού Verilog καθώς και τα εργαλεία αυτοματοποιημένης σχεδίασης (EDA Tools) , ModelSim για προσομοίωση της λογικής σχεδίασης, Design Compiler για την λογική σύνθεση της σχεδίασης, SoC Encounter για την τοποθέτηση και διασύνδεση, Cadence ICFB για την οριστικοποίηση και εξαγωγή του κυκλώματος προς κατασκευή.

Η μεθοδολογία της ροής σχεδίασης που παρατηθείτε μπορεί να γίνει οδηγός για την υλοποίηση οποιουδήποτε ψηφιακού κυκλώματος. Η γενικότητα της μεθοδολογίας έγκειται στο γεγονός ότι έχουν καταγραφεί όλα τα βήματα και στάδια της ροής σχεδίασης ψηφιακών κυκλωμάτων.

Λέξεις Κλειδιά: << Ροή Ψηφιακή Σχεδίασης, ASIC, CMOS, Game Of Life, EDA Tools >>

Ευχαριστίες

Αυτή η εργασία θα ήταν αδύνατο να ολοκληρωθεί χωρίς την συμπαράσταση και ανιδιοτελή βοήθεια πρώτα απ' όλων των καθηγητών μου, κ. Νέστορα Ευμορφόπουλου, κ. Γεώργιου Δημητρίου και κ. Γεώργιου Σταμούλη για την έμπνευση και την γνώση που μου μετέφεραν σε αυτόν τον τομέα του Υλικού των Υπολογιστών.

Επίσης θα ήθελα να ευχαριστήσω τους συναδέλφους από το εργαστήριο Ε5 για την υπομονή τους και την γνώση που μου μετέδωσαν καθ' όλη την διάρκεια της ακαδημαϊκής μου πορείας.

Τέλος θέλω να ευχαριστήσω την οικογένεια μου για την στήριξη και παρότρυνση σε όλα αυτά τα χρόνια.

Πίνακας περιεχομένων

1 *Εισαγωγή*

1.1 Σχεδίαση Ολοκληρωμένων κυκλωμάτων

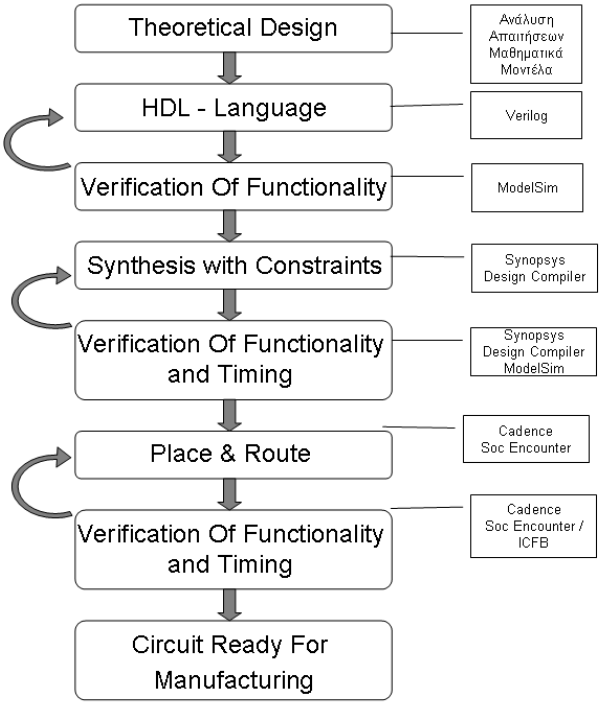
Στην διάρκεια των περασμένων δεκαετιών η τεχνολογία CMOS (Complementary Metal Oxide Semiconductor) έπαιξε σημαντικό ρόλο στην ανάπτυξη της παγκόσμιας βιομηχανίας ολοκληρωμένων κυκλωμάτων. Οι πρώτες ιδέες για την υλοποίηση των τρανζίστορ εμφανίστηκαν το 1930, όμως προβλήματα που σχετίζονταν με την διαχείριση των υλικών δεν επέτρεψαν την ολοκλήρωση των προσπαθειών, μέχρι την ανακάλυψη της επίπεδης επεξεργασίας πυριτίου γύρω στο 1960. Αρχικά χρησιμοποιούνταν στοιχεία μίας πολικότητας p-MOS, μέχρι την στιγμή που παρουσιάστηκε η τεχνολογία πυριτίου n- MOS. Μετά τις ανακαλύψεις αυτές άρχισαν να υλοποιούνται οι τεχνολογίες CMOS που χαρακτηριστικό τους ήταν οι τάξεις μεγέθους χαμηλότερη κατανάλωση ισχύος.

Η τεχνολογία CMOS είναι το βασικό δομικό στοιχείο για την δημιουργία ολοκληρωμένων κυκλωμάτων ειδικού σκοπού ASIC’s (Application Specific Integrated Circuits).

Η εξέλιξη αυτής της τεχνολογίας στις μέρες μας οδήγησε στην κατασκευή κυκλωμάτων πολύ μεγάλης κλίμακας VLSI (Very Large Scale Integration). Τα μοντέρνα ολοκληρωμένα κυκλώματα μπορούν να περιέχουν έως και 1 δισεκατομμύριο τρανζίστορ κάνοντας την ψηφιακή σχεδίαση εξαιρετικά πολύπλοκη, τους κανόνες σχεδίασης να πολλαπλασιάζονται και την κατασκευή κυκλωμάτων να απαιτεί περισσότερα από 600 βήματα.

Η πολυπλοκότητα των ολοκληρωμένων κυκλωμάτων καθώς και η διαρκής απαίτηση της αγοράς για γρηγορότερα, αποδοτικότερα και μικρότερα κυκλώματα έχει κάνει επιτακτική την ανάγκη χρησιμοποίηση εργαλείων αυτοματοποιημένης σχεδίασης, EDA tools (Electronic Design Automation), για γρηγορότερη και αποτελεσματικότερη παραγωγή. Σημαντικό για την βιομηχανία σχεδίασης ολοκληρωμένων κυκλωμάτων είναι η επιλογή των κατάλληλων εργαλείων σχεδίασης, λόγω του μεγάλου κόστους που έχουν σαν προγράμματα αλλά και

εύρεσης σχεδιαστών με τα απαραίτητα προσόντα για την λειτουργία τους. Τα εργαλεία σχεδίασης από διαφορετικούς προμηθευτές συνεργάζονται μεταξύ τους σε μία ροή σχεδίασης για την επίτευξη των απαιτήσεων και των περιορισμών για την επιτυχή παραγωγή ολοκληρωμένων κυκλωμάτων.



Διάγραμμα Ροής Ψηφιακής Σχεδίασης.

Η βιομηχανία παραγωγής ψηφιακών ολοκληρωμένων συνεχώς αναπτύσσεται αναζητώντας μικρότερες διαστάσεις στους αγωγούς πολυπυριτίου, πλέον οι τεχνολογίες σχεδίασης αναφέρονται στα 32nm και 45nm, χαμηλότερη κατανάλωση ισχύος λόγω της εκτεταμένης εφαρμογής που έχουν στις φορητές συσκευές καθώς και σε εύρεση νέων υλικών πέρα του πυριτίου που πλέον φθάνει στα φυσικά όρια αντοχής του.

1.2 Αντικείμενο διπλωματικής

Στον τομέα της σχεδίασης ολοκληρωμένων κυκλωμάτων στο διαδίκτυο υπάρχουν πολλοί οδηγοί χρήσης για τα εργαλεία αυτοματοποιημένης σχεδίασης οι οποίοι δεν καλύπτουν γενικότερες περιπτώσεις ψηφιακής σχεδίασης.

Η εργασία αυτή προσπαθεί να συγκεντρώσει και να παρουσιάσει όλες τις αναγκαίες πληροφορίες για μία πλήρη ροή ψηφιακής σχεδίασης και να τις παρουσιάσει ως οδηγό για επαναχρησιμοποίηση για οποιοδήποτε ολοκληρωμένο κύκλωμα. Η χρησιμοποίηση του

παιχνιδιού Game Of Life ως παράδειγμα κυκλώματος, προσπαθεί να προσδώσει την γενικότητα που χρειάζεται στην εργασία αυτή ώστε να καλύπτονται όλες οι σχεδιάσεις.

Η εργασία προσπαθεί να λύσει τα θέματα σχεδίασης σε επίπεδο αρχιτεκτονικής και περιγραφής λειτουργίας χρησιμοποιώντας γλώσσες περιγραφής υλικού και εργαλείων προσομοίωσης στο επίπεδο καταχωρητή. Στο λογικό επίπεδο προσπαθεί να παρουσιαστεί μεθοδολογία για την λογική σύνθεση με βιομηχανικά εργαλεία σύνθεσης, ικανοποιώντας σχεδιαστικούς κανόνες και περιορισμούς. Στο κυκλωματικό επίπεδο δίνονται μεθοδολογία για την τοποθέτηση και διασύνδεση σε αυτοματοποιημένο εργαλείο ώστε η φυσική σχεδίαση που θα εξαχθεί να έχει όσο το δυνατόν καλύτερο χρονισμό, πυκνότητα σχεδίασης, κατανάλωση ενέργειας. Οι παραπάνω ενέργειες καλύπτουν το Front – End Design ενός ολοκληρωμένου κυκλώματος. Τέλος αναφέρονται οι ενέργειες που πρέπει να ακολουθήσει ο σχεδιαστής ώστε το κύκλωμα να προετοιμαστεί και να αποσταλεί για κατασκευή σε εργοστάσια παραγωγής. Η κατασκευή σε εργοστάσιο παραγωγής καλύπτει την Back – End Design ενός ολοκληρωμένου κυκλώματος.

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής βασίζεται κυρίως στην ανάπτυξη μεθοδολογίας για την ψηφιακή σχεδίαση ολοκληρωμένων κυκλωμάτων με χρήση αυτοματοποιημένων εργαλείων σχεδίασης και συνοψίζεται ως εξής:

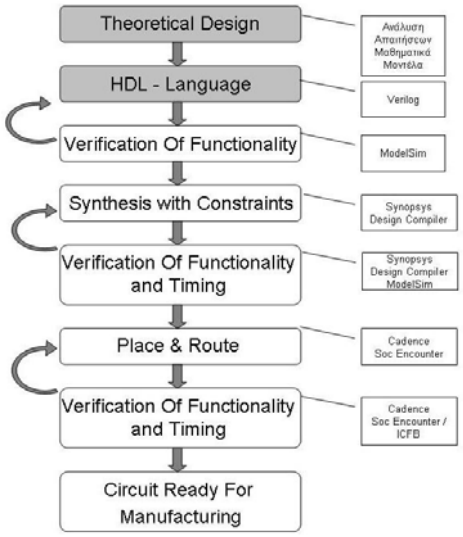
1. Μελετήθηκε η γλώσσα περιγραφής υλικού Verilog.
2. Υλοποιήθηκε παραμετροποίηση του κώδικα του παιχνιδιού GoL.
3. Παρουσιάστηκε προσομοίωση σχεδίασης στο εργαλείο ModelSim.
4. Μελετήθηκε και παρουσιάστηκε μεθοδολογία για την ροή σχεδίασης της λογικής σύνθεσης στον Design Compiler .
5. Δόθηκε μεθοδολογία για την αυτοματοποιημένη τοποθέτηση και διασύνδεση σε φυσική σχεδίαση με το εργαλείο SoC Encounter.
6. Παρουσιάστηκε ροή για την οριστικοποίηση και εξαγωγή του κυκλώματος προς κατασκευή.

1.3 Οργάνωση κειμένου

Εργασίες σχετικές με το επίπεδο αρχιτεκτονικής, σχεδιασμού, περιγραφής λειτουργίας και προσομοίωσης της σχεδίασης παρουσιάζονται στο Κεφάλαιο 2 . Το Κεφάλαιο 3 αναφέρεται στην λογική σχεδίαση ενός ολοκληρωμένου κυκλώματος. Στο Κεφάλαιο 4 αναπτύσσεται μεθοδολογία για την τοποθέτηση και διασύνδεση. Στο Κεφάλαιο 5 παρουσιάζονται τα βήματα για την οριστικοποίηση του κειμένου. Στο κεφάλαιο 6 βρίσκεται ο επίλογος της εργασίας και στο Κεφάλαιο 7 η βιβλιογραφία. Τέλος υπάρχει παράρτημα όπου παρουσιάζονται οι εντολές που χρησιμοποιήθηκαν για την λειτουργία των αυτοματοποιημένων εργαλείων σχεδίασης.

2 *Θεωρητική Σχεδίαση - Αρχιτεκτονική*

Σε αυτό το κεφάλαιο αναλύονται τα τρία πρώτα στάδια της ροής σχεδίασης ενός ψηφιακού κυκλώματος. Στην εργασία αυτή χρησιμοποιήθηκε ως κύκλωμα το παιχνίδι του John Conway Game Of Life (GoL). Το GoL περιγράφηκε σε γλώσσα περιγραφής υλικού Verilog καθώς και ο κώδικας παραμετροποιήθηκε ώστε να μπορούν να γίνουν δοκιμές σε οποιαδήποτε διάσταση επιθυμεί ο χρήστης. Έπειτα προσομοιώθηκε η RTL (Register Transfer Level) σχεδίαση για την επαλήθευση της σωστής λειτουργίας.



2.1 Περιγραφή του Game Of Life

Το GoL επινοήθηκε από τον μαθηματικό John Conway το 1970. Ως μαθηματικό μοντελο ανήκει στην κατηγορία των κυψελοειδών αυτόματων (cellular automata) και είναι το χαρακτηριστικότερο παράδειγμα στον τομέα αυτό.

Το κυψελοειδές αυτόματο πρωτοπαρουσιάστηκε από τον John Von Neumann σαν πιθανό μοντέλο για την βιολογία και ορίζεται: « *Ένα διακριτό μοντέλο το οποίο βρίσκει εφαρμογή σε πολλούς επιστημονικούς τομείς. Αποτελείται από ένα πλέγμα κυψελών, πεπερασμένου αριθμού,*

οι οποίες έχουν πεπερασμένο αριθμό καταστάσεων (π.χ. On ή Off) και εξελίσσεται σύμφωνα με ένα σύνολο κανόνων, το οποίο βασίζεται στην κατάσταση των γειτονικών κυψελών. »

Το Game of Life είναι ένα παιχνίδι μηδενικής κατάστασης (zero- state) καθώς η εξέλιξη του εξαρτάται μόνο από την αρχική του είσοδο. Γρήγορα έγινε διάσημο για τους απλούς κανόνες του και λόγω του εντυπωσιακού τρόπου που τα πρότυπα που δέχεται σαν αρχική κατάσταση, εξελίσσονται σε κάθε επανάληψη.

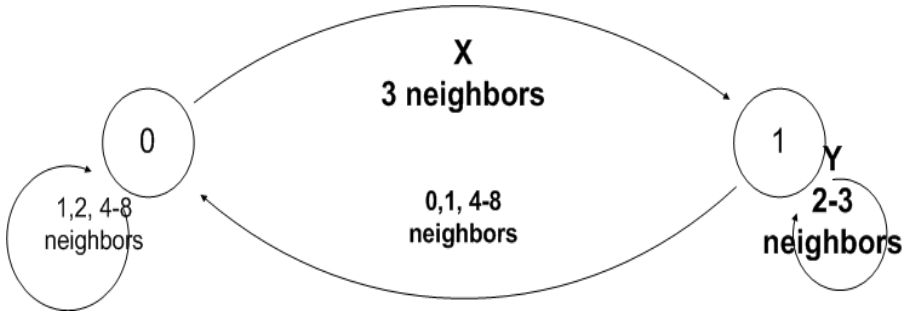
Οι κανόνες του παιχνιδιού είναι

- Σε ένα κατειλημμένο κελί (populated cell)
  - Αν αυτό δεν έχει κανένα γειτονικό κατειλημμένο πεθαίνει, από μοναξιά
  - Αν έχει 4 ή περισσότερους γείτονες πεθαίνει, από υπερπληθυσμό
  - Αν έχει 2 ή 3 γείτονες παραμένει ζωντανό
- Σε ένα μη κατειλημμένο κελί (unpopulated cell)
  - Κάθε κελί με 3 κατειλημμένους γείτονες γεννιέται

2.2 Σχεδιασμός Μοντελοποίηση Συστήματος

2.2.1 Μηχανή Πεπερασμένων Καταστάσεων

Το panel του συστήματος χωρίζεται σε κυψέλες, οι οποίες αλλάζουν την κατάσταση τους στο πέρασμα του χρόνου ανάλογα με την κατάσταση των γειτονικών. Αυτή η συμπεριφορά στα ψηφιακά κυκλώματα μπορεί να μοντελοποιηθεί με μια μηχανή πεπερασμένων καταστάσεων (Final State Machine). Βάσει των κανόνων του παιχνιδιού έχουμε την παρακάτω μηχανή πεπερασμένων καταστάσεων.



2.2.2 Πίνακας Αληθείας

Από την παραπάνω μηχανή πεπερασμένων καταστάσεων μπορεί να εξαχθεί ο πίνακας αληθείας για την σχεδίαση μιας κυψέλης του συστήματος.

X=around(7:0)								**C**
Upleft C1	**Up C2**	**Upright C3**	**Left C4**	**Right C5**	**Downleft C6**	**Down C7**	**Downright C8**	**C0**
0	0	0	0	0	1	1	1	1
0	0	0	0	1	0	1	1	1
0	0	0	0	1	1	0	1	1
0	0	0	0	1	1	1	0	1
0	0	0	1	0	0	1	1	1
0	0	0	1	0	1	0	1	1
0	0	0	1	0	1	1	0	1
0	0	0	1	1	0	0	1	1
0	0	0	1	1	0	1	0	1
0	0	0	1	1	1	0	0	1
0	0	1	0	0	0	1	1	1
0	0	1	0	0	1	0	1	1
0	0	1	0	0	1	1	0	1
0	0	1	0	1	0	0	1	1
0	0	1	0	1	0	1	0	1
0	0	1	0	1	1	0	0	1

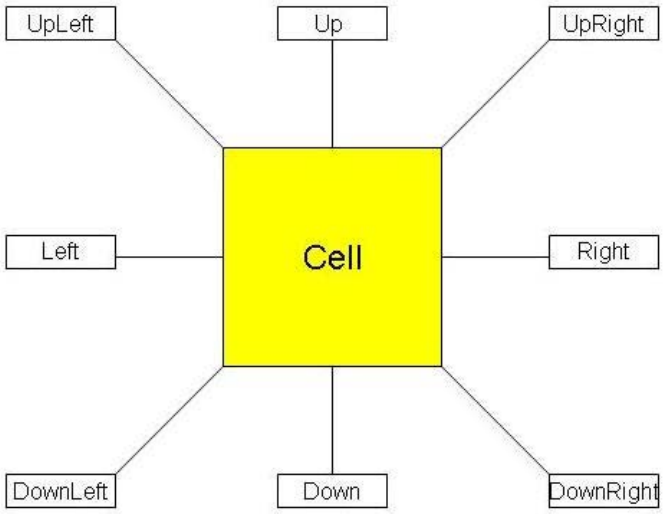
Παρατηρείται στο απόσπασμα του πίνακα ότι η έξοδος της κυψέλης είναι 1 μόνο όταν 2 ή 3 γειτονικές κυψέλες είναι 1.

2.3 Μοντελοποίηση Κυκλώματος

2.3.1 Μοντέλο Κυψέλης

Το κύκλωμα αποτελείται από ένα πάνελ κυψελών. Κάθε κυψέλη έχει ως είσοδο την κατάσταση των γειτονικών και ως έξοδο την δική της κατάσταση.

Κάθε κελί επομένως έχει 8 εισόδους , 1 έξοδο, 1 σήμα ανάθεσης της αρχική κατάστασης καθώς και σαν ακολουθιακό κύκλωμα απαιτείται είσοδος ρολογιού και σήματος SET/RESET. Το μοντέλο αυτό είναι το χαμηλότερο ιεραρχικά στοιχείο της σχεδίασης (component).



Μοντέλο μίας Κυψέλης

2.3.2 Μοντέλο Πάνελ Κυψελών

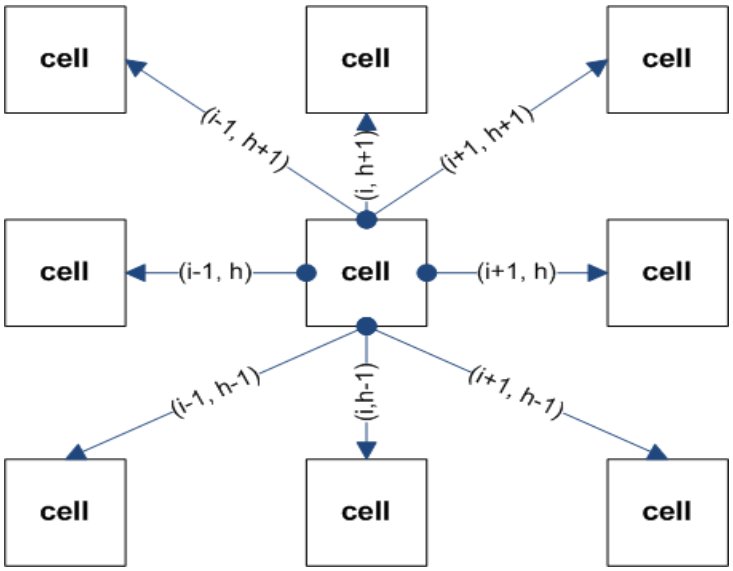
Μετά τη μοντελοποίηση μιας κυψέλης πρέπει υψηλότερο ιεραρχικά επίπεδο να δοθεί κατάλληλος τρόπος διασύνδεσης μεταξύ των κυψελών. Επίσης η σχεδίαση πρέπει να είναι παραμετροποιημένη ώστε ο χρήστης να μπορεί εύκολα να καθορίζει το μέγεθος του πάνελ για NxN διάσταση.

Κάθε κελί έχει σαν είσοδο τα γειτονικά του. Στο Panel για κάθε κελί i θα έχει σαν εισόδους κελία με συντεταγμένες

UpLeft: (i-1, N-1)	Up: (i, N-1)	UpRight: (i+1, N-1)
Left: (i-1, N)	*Cell[i]*	Right: (i-1, N)
DownLeft: (i-1, N+1)	Down: (i , N+1)	DownRight: (i+1, N-1)

Στο πάνελ κυψελών διακρίνονται ειδικές περιπτώσεις για

1. Τις γωνιακές κυψέλες, καθώς πρέπει δύο πλευρές αυτών να βρίσκονται συνεχώς στην κατάσταση Off
2. Την πρώτη και τελευταία γραμμή καθώς η πάνω και κάτω πλευρά αντίστοιχα, πρέπει να βρίσκονται στην κατάσταση Off
3. Την πρώτη και τελευταία σειρά καθώς η αριστερή και δεξιά πλευρά αντίστοιχα, πρέπει να βρίσκονται στην κατάσταση Off



Μοντέλο ενός Πάνελ

2.4 Υλοποίηση του μοντέλου σε γλώσσα Verilog

Σε αυτό το επίπεδο της σχεδίασης πρέπει να περιγραφεί η λειτουργία και συμπεριφορά του παραπάνω μοντέλου σε λογική σχεδίαση. Η συμπεριφορά αυτή πρέπει να οριστεί σε όρους σημάτων εισόδου-εξόδου καθώς να οριστούν και οι λογικές πράξεις που εφαρμόζονται σε αυτά τα σήματα. Το επίπεδο αυτό ονομάζεται Register Transfer Level (RTL) και ο κώδικας που θα προκύψει (RTL netlist) πρέπει να είναι ικανός να προσομοιωθεί για την επαλήθευση

της σωστής λειτουργίας καθώς και να μετατραπεί σε κώδικα επιπέδου πυλών με κάποιο εργαλείο σύνθεσης.

Η Verilog είναι μια πανίσχυρη και ευρέως διαδεδομένη γλώσσα περιγραφής υλικού (Hardware Description Language). Οι HDL' s δίνουν την δυνατότητα στον σχεδιαστή δίνοντας μια περιγραφή της συμπεριφοράς του κυκλώματος να μεταφράσουν αυτή την συμπεριφορά σε κυκλωματικό επίπεδο χρησιμοποιώντας τα κατάλληλα εργαλεία.

2.4.1 Ορισμός Κυψέλης (Cell)

Η κυψέλη αποτελεί την δομική μονάδα ενός πάνελ. Έχει 11 εισόδους και μία έξοδο, σε γλώσσα Verilog ορίζεται :

```
module cell_logic   (   gsres, clock,
                        P0,
                        C1, C2, C3,
                        C4,       C5,
                        C6, C7, C8,
                        C0
                        );
input gsres;                 //General set-reset
input clock;                //System clock
input P0;         // initial state
input   C1,   C2,   C3,   C4,   C5,   C6,   C7,   C8;
//neighbor's status
output C0;             // output
```

Στον πίνακα αληθείας του κυκλώματος παρατηρήθηκε πως όταν υπάρχουν 2 ή 3 γειτονικές κυψέλες σε κατάσταση On τότε και η κυψέλη θα είναι On. Αυτή η συμπεριφορά μεταφράζεται σε γλώσσα Verilog

```
always @(posedge clock or negedge gsres)
    if (~gsres)      C0 <= P0; //Load initial state
    else
        begin
            if  (      (C0==1'b1  &&  (composition_C0  ==2  ||
composition_C0 ==3 ) )  ||
                    (C0==1'b0 && composition_C0 ==3)    )
                C0 <= 1'b1;
            else
C0 <= 1'b0;
        end
```

2.4.2 Κεντρική Σχεδίαση (Πάνελ)

Μετά τον ορισμό της κυψέλης πρέπει να ορισθεί το πάνελ κυψελών λαμβάνοντας υπ' όψιν τις παρατηρήσεις για τις ειδικές περιπτώσεις που δόθηκαν προηγούμενα. Το πάνελ θα αποτελεί και την υψηλότερη ιεραρχικά σχεδίαση.

```
module top_module (  gsres, clock,
                    Panel,
                    C );
parameter N;
parameter N_2=N*N;

input gsres;                     //General set-
reset
input clock;                    //System clock
input [0:N_2-1] Panel;   //initial input
output [0:N_2-1] C;      //output
```

Στην ανώτερη ιεραρχικά σχεδίαση προσθέτουμε μια παράμετρο N την οποία ο χρήστης μπορεί να μεταβάλει ώστε να δώσει οποιαδήποτε διάσταση N x N επιθυμεί στο πάνελ.

Μετά την συγγραφή του κώδικα σε Verilog χρειάζεται να επαλήθευση της ορθότητας και σωστής λειτουργίας αυτού, μέσω εργαλείων προσομοίωσης (Simulation).

2.5 Προσομοίωση Σχεδίασης

Η προσομοίωση χρησιμοποιείται για τον έλεγχο της ορθότητας της σωστής λειτουργίας μιας λογικής σχεδίασης. Είναι ένα πολύ σημαντικό στάδιο για την ψηφιακή σχεδίαση ενός ολοκληρωμένου κυκλώματος διότι ο σχεδιαστής μπορεί να επαληθεύσει εάν η λογική σχεδίαση πληροί τις προδιαγραφές που έχουν δοθεί . Ο σχεδιαστής σε αυτό το στάδιο έχει την δυνατότητα να αλληλεπιδράσει με την σχεδίαση μελετώντας την συμπεριφορά του κυκλώματος καθώς και την ορθότητα των αποτελεσμάτων στις εξόδους του κυκλώματος. Αυτό δίνει την δυνατότητα αποσφαλμάτωσης τις σχεδίασης σε πρώιμο στάδιο της ροής ψηφιακής σχεδίασης καθώς η διόρθωση σφάλματος στο επίπεδο συμπεριφοράς του κυκλώματος είναι πολύ πιο εύκολη απ' ότι στα επόμενα στάδια της σχεδίασης.

2.5.1 Προετοιμασία για προσομοίωση

Για την προσομοίωση ενός ψηφιακού κυκλώματος χρειάζεται η δημιουργία ενός αρχείου δοκιμής (Testbench). Για το παράδειγμα αυτό υλοποιήθηκε ένα testbench χρησιμοποιώντας κάποιο από τα πρότυπο (pattern) του GOL 10- Cell Row για το οποίο γνωρίζουμε την εξέλιξη του σε κάθε βήμα του.

2.5.2 Εργαλείο Προσομοίωσης

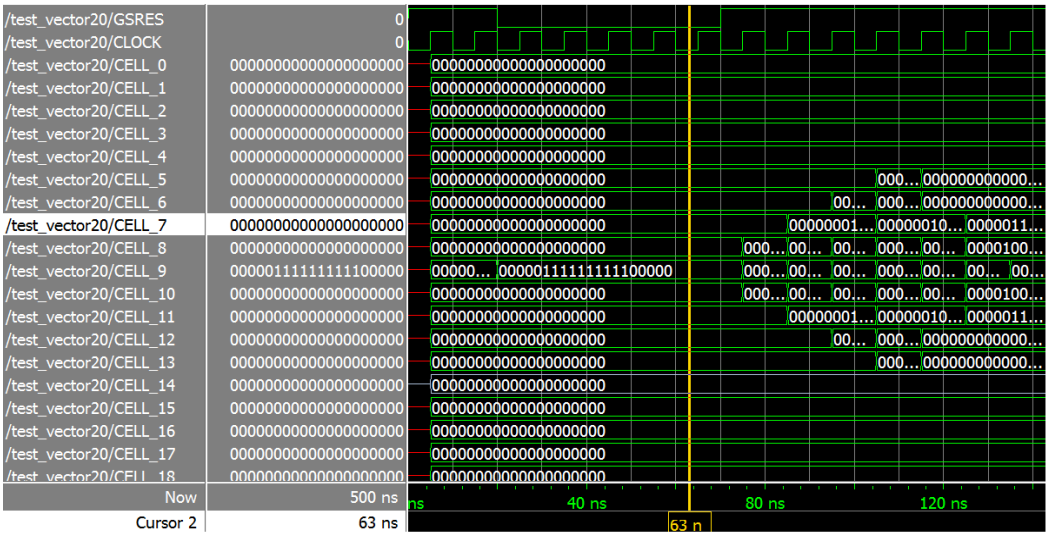
Στην εργασία αυτή χρησιμοποιήθηκε το εργαλείο προσομοίωσης ModelSim της Mentor Graphics. Είναι ένα από τα δημοφιλέστερα βιομηχανικά εργαλεία προσομοίωσης και αυτό το οφείλει στην καλή διαχείριση πόρων που έχει μιας και η προσομοίωση είναι αρκετά δαπανηρή σε υπολογιστικούς πόρους καθώς και στο ότι μπορεί να προσομοιώσει σχεδιάσεις μικτών αρχείων Verilog & VHDL.

Στο εργαλείο προσομοίωσης χρησιμοποιούνται συνηθέστερα αρχεία μακροεντολών (do files) στο παράδειγμα μας χρησιμοποιήθηκε

```
vlib work // set working library
vmap work work // set working directory
vlog test_vector20.v // compile HDL files
vsim -novopt test_vector20 // simulate Design
add wave sim:/test_vector20/* // Add signals to waveform
window
run 500 // Run 500 ns
```

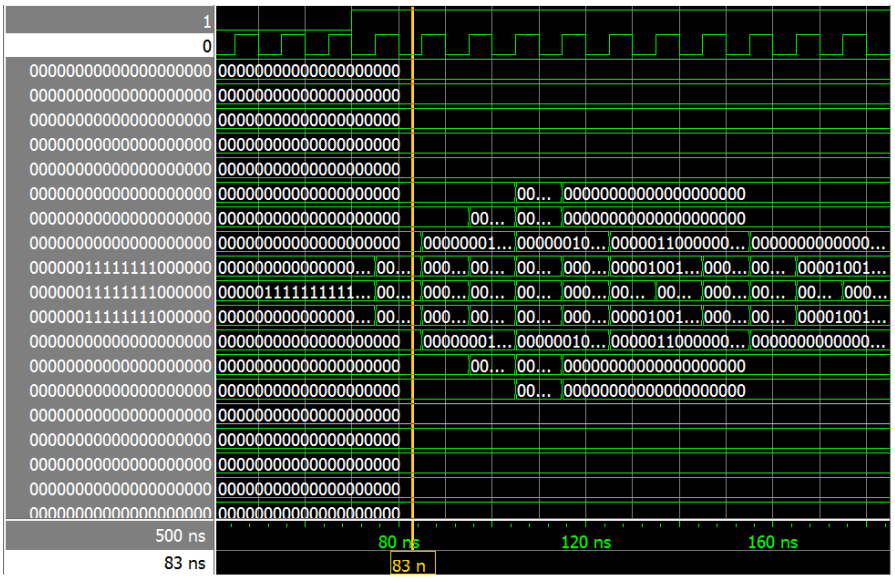
2.6 *Αποτελέσματα Προσομοίωσης*

Μετά την φόρτωση της σχεδίασης πρέπει να ελεγχθούν οι κυματομορφές της σχεδίασης για την επαλήθευση της σωστής λειτουργίας του κυκλώματος



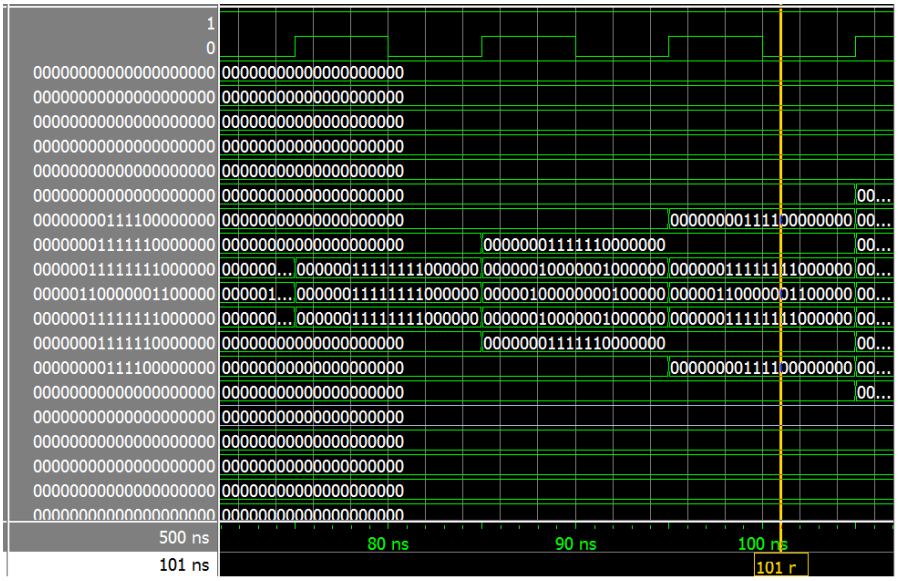
Στιγμιότυπο 1

Φαίνεται στο στιγμιότυπο 1 ότι κατά την διάρκεια που το σήμα GSRES είναι 0 φορτώνεται ο πίνακας με το πρότυπο σαν είσοδος του κυκλώματος. Η εξέλιξη του θα ξεκινήσει όταν πλέον το σήμα γίνει 1.



Στιγμιότυπο 2

Στο στιγμιότυπο 2 φαίνεται η εξέλιξη του παιχνιδιού μετά από μία επανάληψη.



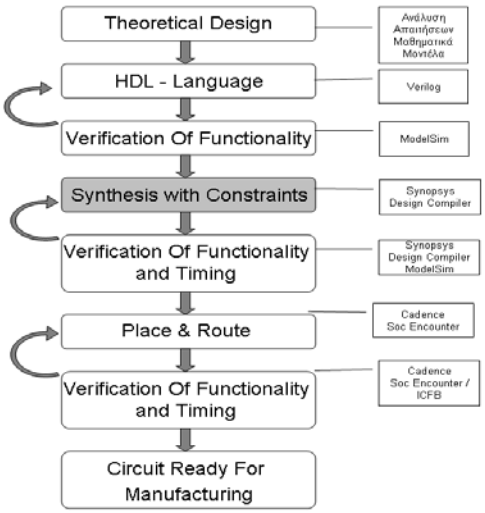
Στιγμιότυπο 3. Η εξέλιξη του παιχνιδιού μετά από 4 επαναλήψεις.

Μετά το τέλος της προσομοίωσης και εφόσον η σχεδίαση ανταποκρίνεται στις απαιτήσεις λειτουργίας που ορίστηκαν παραπάνω είναι έτοιμη για το επόμενο στάδιο τη ψηφιακής σχεδίασης ολοκληρωμένου κυκλώματος την λογική σύνθεση.

3 *Λογική Σύνθεση Σχεδίασης*

(Logic Synthesis)

Στο στάδιο αυτό της σχεδίασης ψηφιακού κυκλώματος χρειάζεται να μετατραπεί η λογική σχεδίαση που δόθηκε στο προηγούμενο κεφάλαιο σε σχεδίαση επιπέδου πυλών (gate – level net list), εξαρτημένη από τεχνολογική βιβλιοθήκη (technology library ή standard cell library), ικανοποιώντας περιορισμούς για τον χρονισμό του κυκλώματος, την επιφάνεια και την κατανάλωση ισχύος. Αυτό γίνεται με τα εργαλεία σύνθεσης, στην εργασία αυτή χρησιμοποιήθηκε το βιομηχανικό εργαλείο Design Compiler της Synopsys.



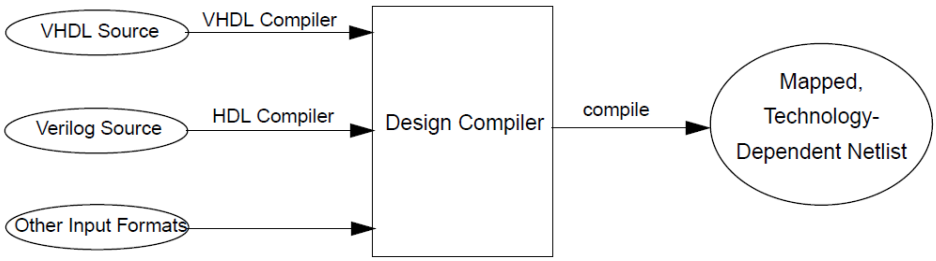
Πριν τα αυτοματοποιημένα εργαλεία σύνθεσης γίνονταν όλοι οι υπολογισμοί γίνονταν χειροκίνητα. Γινόταν κυρίως με ελαχιστοποιήσεις πινάκων Karnaugh και αργότερα οι ελαχιστοποιήσεις γίνονταν με την μέθοδο Quine – McCluskey. Κύριος στόχος των σχεδιαστών ήταν η παραγωγή βέλτιστης τεχνολογικά εξαρτημένης σχεδίασης που ικανοποιεί τους περιορισμούς. Αυτό ήταν τρομερά χρονοβόρο και οι σχεδιάσεις, ειδικότερα οι μεγαλύτερες, ήταν επιρρεπής σε σφάλματα. Η αποσφαλμάτωση και η οποιαδήποτε βελτιστοποίηση ήταν ιδιαίτερα επίπονη καθώς η διαδικασία έπρεπε να επαναληφθεί σε μεγάλο μέρος της.

Τα εργαλεία σύνθεσης έχουν βελτιώσει δραματικά το χρόνο σχεδίασης και έχουν αυξήσει την παραγωγικότητα των σχεδιαστών. Χάρη σε αυτά ο σχεδιαστής μπορεί να δημιουργήσει σχεδιάσεις σε υψηλό αφαιρετικό επίπεδο και να μεταφραστούν σε σχεδιάσεις τεχνολογικά εξαρτημένες είτε αυτές είναι ακολουθιακές (sequential) είτε συνδυαστικές (combinational). Η

σχεδίαση πλέον γίνεται εύκολα επαναχρησιμοποιήσιμη καθώς το πρόβλημα εύρεσης βέλτιστου netlist και το πρόβλημα ικανοποίησης των περιορισμών λύνεται αυτόματα από τα εργαλεία σύνθεσης. Επιπλέον το ενδιαφέρον του σχεδιαστή στρέφεται στον ακριβή προσδιορισμό των περιορισμών και στην βελτιστοποίηση της τεχνολογικής βιβλιοθήκης.

3.1 Προετοιμασία για Σύνθεση

Για την προετοιμασία της σύνθεσης της σχεδίασης χρειάζονται ο κώδικας HDL καθώς και η τεχνολογική βιβλιοθήκη.



3.1.1 Τεχνολογική βιβλιοθήκη

Μια τεχνολογική βιβλιοθήκη είναι μια συλλογή λογικών συναρτήσεων χαμηλού επιπέδου όπως NAND, NOT, NOR, flip – flop, latches, buffers είτε και πιο σύνθετων συναρτήσεων όπως αθροιστών, πολυπλεκτών κ.α. . Αυτές οι λογικές συναρτήσεις ονομάζονται βασικά κύτταρα ή τυποποιημένα κύτταρα (Standard Cells) και είναι τα βασικά δομικά στοιχεία για ASIC's (application – specific intergraded circuits) σχεδιάσεις. Τα standard cells είναι σχεδιάσεις με προκαθορισμένο ύψος και μεταβαλλόμενο πλάτος ώστε να μπορούν να τοποθετηθούν σε σειρές για την δημιουργία αυτοματοποιημένης φυσικής σχεδίασης (layout).

Μία τυπική τεχνολογική βιβλιοθήκη πρέπει να περιέχει

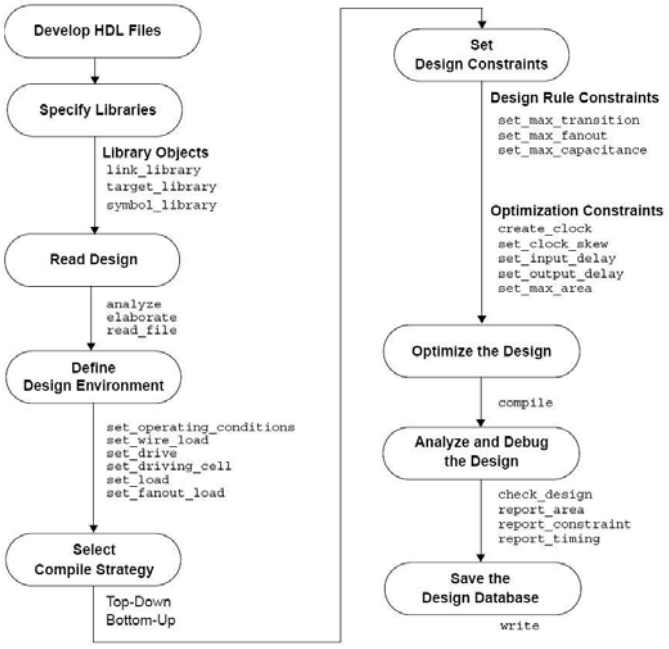
- Βάση Δεδομένων Βιβλιοθήκης (Library Database) : Περιέχει την περιγραφή των Standard Cells σε φυσική σχεδίαση, σχηματικό, σύμβολο κ.α. όψεις αναγκαίες για προσομοίωση και σχεδιασμό. Συνηθέστερα περιγράφεται σε .lef αρχείο (Cadence Library Exchange Format) είτε σε .db αρχείο (Synopsys Milkyway Format).
- Περίληψη Χρονισμού (Timing Abstract) : Είναι αρχεία .lib (Liberty Format), τα οποία περιέχουν πληροφορίες για τον χρονισμό, τον θόρυβο και την κατανάλωση ισχύος των Standard Cells.
- Φυσικές Σχεδιάσεις των Standard Cells

- Μοντέλα SPICE
- Περιγραφή σε Verilog ή VHDL Vital
- Κανόνες σχεδίασης : DRC (Design Rule Check) και LVS (Layout Vs Schematic)

Για την εργασία αυτή χρησιμοποιήθηκε η βιβλιοθήκη GSCLib (Generic Standard Cell Library) η οποία δημιουργήθηκε από την Cadence κυρίως για εκπαιδευτικούς και δοκιμαστικούς σκοπούς. Η βιβλιοθήκη βασίζεται στην τεχνολογία gpdk (Generic Process Design Kit) και είναι διαθέσιμη σε μεγέθη 180nm, 90nm και 45nm[11]. Είναι μια πλήρης βιβλιοθήκη καθώς περιέχει όλα τα παραπάνω χαρακτηριστικά.

Στο Design Compiler δέχεται μόνο .db αρχεία οπότε πρέπει να μετατραπεί το .lef αρχείο της βιβλιοθήκης. Αυτό γίνεται με το ενσωματωμένο εργαλείο του Design Compiler, Design Library.

3.1 Ροή Λογικής Σύνθεσης



Διάγραμμα Ροής Σύνθεσης του Design Compiler

Στον DC χρησιμοποιούνται συνήθως αρχεία Tcl τα οποία περιέχουν το σύνολο των εντολών για την χρήση του προγράμματος. Παρακάτω περιγράφεται η ροή της λογικής σύνθεσης της σχεδίασης.

1. Αρχικά ορίζονται σταθερές μεταβλητές για την σχεδίαση όπως το που θα αποθηκευθούν οι αναφορές, που βρίσκεται τα αρχεία HDL, τους καταλόγους απ' όπου θα διαβαστούν οι τεχνολογικές βιβλιοθήκες, την περίοδο του ρολογιού κ.α.

```
# set some per design variables
set TOPLEVEL  "top_module"
set LOG_PATH  "log1/"
set GATE_PATH  "lib/"
set RTL_PATH  "src/"
set STAGE  "gsclib"
set CLK  "clock"
set RST  "gsres"
set CLK_PERIOD 2.50; # 10ns
set CLK_UNCERTAINTY  0.3; # 200ps
```

2. Έπειτα ορίζουμε τις link και target βιβλιοθήκες. Σε αυτές καθορίζονται οι διάφορες τεχνολογικές πληροφορίες (cell names, cell pin names, delay arcs, pin loading, design rules, operating conditions)

```
set link_library   $GATE_PATH /GSCLib_2.0_prelim3.db
set target_library $GATE_PATH /GSCLib_2.0_prelim3.db
```

3. Εφόσον έχουν οριστεί οι βιβλιοθήκες πρέπει να διαβαστούν τα αρχεία που περιέχουν την λογική σχεδίαση του κυκλώματος. Στην περίπτωση που υπάρχει gate – level netlist χρησιμοποιείται η εντολή **read_file**, διαφορετικά σε περίπτωση όπου υπάρχει RTL netlist χρησιμοποιείται ο συνδυασμός εντολών analyze – elaborate. Η εντολή **analyze** ελέγχει την ορθότητα του κώδικα και η **elaborate** εντολή δημιουργεί μια ενδιάμεση τεχνολογικά ανεξάρτητη σχεδίαση απαλείφοντας τις αριθμητικές παραμέτρους.

```
analyze -library WORK -format verilog $RTL_PATH /cell_logic.v
analyze -library WORK -format verilog $RTL_PATH/my.v
elaborate $TOPLEVEL -library WORK
 # Set design top
current_design $TOPLEVEL
```

4. Οι περιορισμοί που τίθενται παρακάτω (Design Rule Constraints) είναι ορισμένοι από την τεχνολογική βιβλιοθήκη και δεν μπορούν να επηρεαστούν από τον χρήστη.

```
set_operating_conditions -library GSCLib_2.0
set_wire_load_model -library GSCLib_2.0
set_timing_ranges -library GSCLib_2.0
```

5. Με την εντολή **create_clock** δημιουργείται ένα φυσικό ρολόι με την επιθυμητή περίοδο για την σχεδίαση το οποίο αντιστοιχεί στην είσοδο του κυκλώματος. Επίσης προστίθεται και μια ‘αβεβαιότητα’ για το ρολόι σε περίπτωση που υπάρξουν blocks της σχεδίασης που χρειάζονται μεγαλύτερη περίοδο από την δηλωμένη. Με την εντολή **set_dont_touch_network** εφόσον το ρολόι μεταφραστεί επιτυχώς δεν επιδέχεται πλέον βελτιστοποίησης.

```
create_clock $CLK -period $CLK_PERIOD
set_clock_uncertainty $CLK_UNCERTAINTY [get_clocks $CLK]
set_dont_touch_network [get_clocks $CLK]
```

6. Το σήμα general set reset που υπάρχει στην σχεδίαση πρέπει να μεταφραστεί ως ξεχωριστή οντότητα στη σχεδίαση, με απεριόριστη ικανότητα οδήγησης δεν βελτιστοποιείται καθώς δεν ενδιαφέρει η απόδοση του.

```
set_drive 0 $RST
set_dont_touch_network $RST
```

7. Οι εντολές **set_input_delay**, **set_load**, **set_max_delay** και **set_max_area** είναι εντολές βελτιστοποίησης ορισμένες από τον χρήστη (Optimization constraints) που καθορίζουν τους στόχους της σχεδίασης σε θέματα χρονισμού και μεγέθους του κυκλώματος. Κατά την διάρκεια της σύνθεσης και ειδικότερα κατά την διαδικασία της βελτιστοποίησης, ο DC προσπαθεί να καλύψει όλες αυτές τις απαιτήσεις αλλά ποτέ δεν παραβιάζει τους κανόνες σχεδίασης. Για να βελτιστοποιηθεί μια σχεδίαση κατά τον καλύτερο τρόπο πρέπει να τεθούν ρεαλιστικές παράμετροι βελτιστοποίησης.

```
set_input_delay 0.5 -clock $CLK [remove_from_collection [remove_from_collection [all_inputs] [get_port $CLK]] [get_port $RST]]
set_load 0.003 [remove_from_collection [remove_from_collection [all_inputs] [get_port $CLK]] [get_port $RST]]
set_load  0.003 [all_outputs]
set_max_delay 0.5 -to [all_outputs]
set_max_area 0
```

8. Η σχεδίαση περιέχει πολλά όμοια κελιά για να λυθούν όποιες συγκρούσεις πρέπει το καθένα να γίνει μοναδικό. Αυτό γίνεται με την εντολή **uniquify**. Με την **link** δημιουργούνται όλες οι συνδέσεις μεταξύ των κελιών.

```
# Link all blocks and uniquify them
 link
 uniquify
```

9. Με την εντολή **compile** ο DC ξεκινά την διαδικασία σύνθεσης και βελτιστοποίησης της σχεδίασης. Οι παράμετροι έχουν τεθεί στο υψηλότερο επίπεδο προσπάθειας.

```
set_ultra_optimization -f
compile -map_effort high -area_effort high -incremental_mapping
```

10. Μετά την επιτυχημένη σύνθεση χρειάζεται να παραχθούν οι αναφορές για την σχεδίαση. Τέτοιες είναι ο τεχνολογικά εξαρτημένος κώδικας σε γλώσσα Verilog που θα χρησιμοποιηθεί στο επόμενο βήμα της τοποθέτησης και διασύνδεσης με την εντολή **write -hierarchy**, ένα αρχείο .SDF (Standar Delay Format). Σημαντικό επίσης αρχείο για την διαδικασία τοποθέτησης και διασύνδεσης είναι το .SDC (Synopsis Design Constraint). Ο DC δίνει την δυνατότητα παραγωγής αναφορών για την λειτουργία του κυκλώματος μετά την σύνθεση. Σημαντικές αναφορές είναι αυτές για τον χώρο που καταλαμβάνει το κύκλωμα, οι αναφορές για τα κρίσιμα μονοπάτια και τον χρονισμό του κυκλώματος, την κατανάλωση ισχύος του κυκλώματος, αναφορά για τυχόν παραβιάσεις των προδιαγραφών που έχουν τεθεί και αναφορά για την ιεραρχική δομή του κυκλώματος .

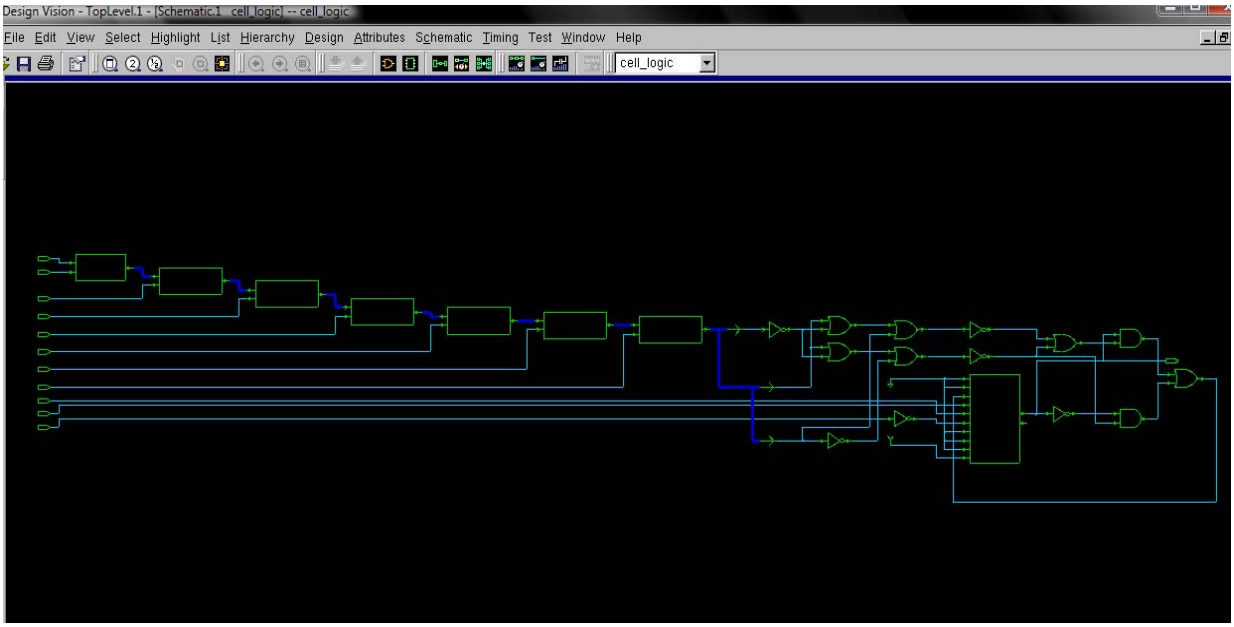
```
write -hierarchy -format verilog -output $LOG_PATH/$TOPLEVEL-$STAGE.v
write_sdf    $LOG_PATH/$TOPLEVEL-$STAGE.sdf
write_sdc    $LOG_PATH/$TOPLEVEL-$STAGE.sdc
check_design        > $LOG_PATH/$TOPLEVEL-$STAGE-check_design.log
report_area               > $LOG_PATH/$TOPLEVEL-$STAGE-area.log
report_timing -nworst 10       > $LOG_PATH/$TOPLEVEL-$STAGE-timing.log
report_hierarchy           > $LOG_PATH/$TOPLEVEL-$STAGE-hierarchy.log
report_cell               > $LOG_PATH/$TOPLEVEL-$STAGE-cell.log
report_power              > $LOG_PATH/$TOPLEVEL-$STAGE-power.log
report_constraint -all_violators -verbose    > $LOG_PATH/$TOPLEVEL-
$STAGE-constraint.log
```

3.2 Αποτελέσματα Σύνθεσης

3.2.1 Σχηματικό (Schematic)

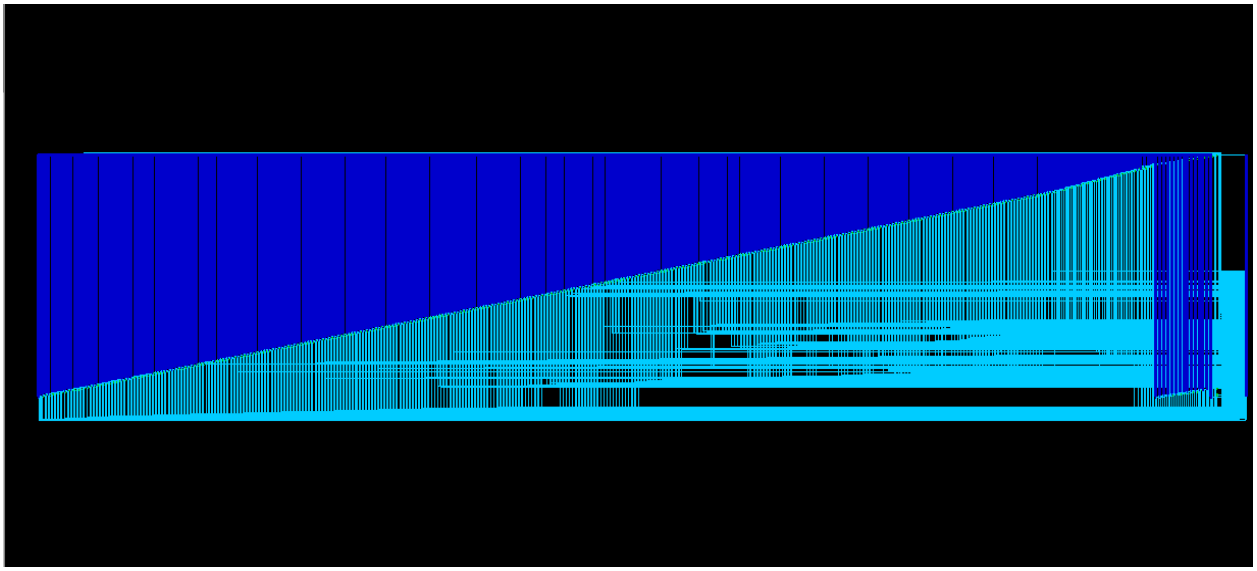
Η σχεδίαση αποτελείται από δομικά στοιχεία τις κυψέλες. Το εργαλείο σύνθεσης Design Vision, η έκδοση του DC με γραφικό περιβάλλον μας δίνει μια απεικόνιση της σχεδίασης σε επίπεδο πυλών. Αυτή η απεικόνιση ονομάζεται σχηματικό.

Η κάθε κυψέλη θα αποτελείται από βασικά κύτταρα της τεχνολογικής βιβλιοθήκης, όπως αυτά έχουν οριστεί από την σύνθεση της σχεδίασης.

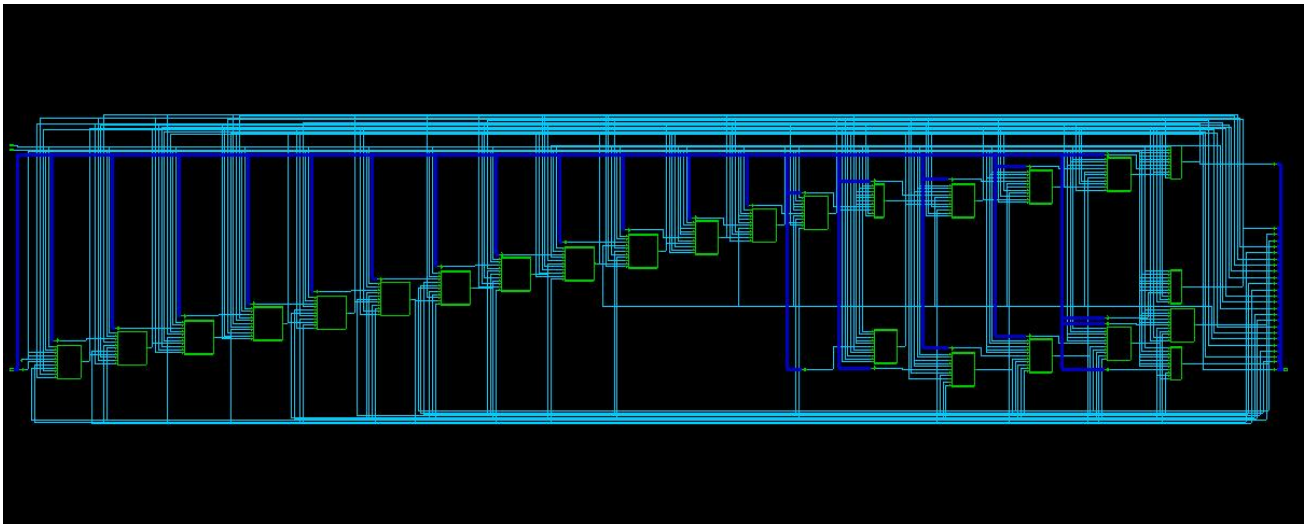


Σχηματικό κυψέλης.

Η κάθε κυψέλη γίνεται δομικό στοιχείο για την κεντρική σχεδίαση συνθέτοντας το κεντρικό πάνελ. Αυτή είναι και η υψηλότερη ιεραρχικά σχεδίαση. Παρακάτω δίδεται το σχηματικό μια σχεδίασης του παραδείγματος 20X20, καθώς και μια σχεδίαση 5X5 για λόγους ευανάγνωσης.



Ένα πάνελ 20X20.



Ένα πάνελ 5X5.

3.2.2 Τεχνολογικά εξαρτημένος Κώδικας

Από τα σημαντικότερα αποτελέσματα της λογικής σύνθεσης είναι η παραγωγή τεχνολογικά εξαρτημένου κώδικα από κώδικα λογικής περιγραφής υψηλότερου επιπέδου.

```
top_module
    cell_logic_0                    r
        ADDFX1              GSCLib_2.0
        ADDHX1              GSCLib_2.0
        DFFSRX1             GSCLib_2.0
        INVX1               GSCLib_2.0
        OAI33X1             GSCLib_2.0
        OR2X1               GSCLib_2.0
    cell_logic_1                    r
        ADDFX1              GSCLib_2.0
        ADDHX1              GSCLib_2.0
        DFFSRX1             GSCLib_2.0
        INVX1               GSCLib_2.0
        OAI33X1             GSCLib_2.0
        OR2X1               GSCLib_2.0
```

Ιεραρχία κεντρικής σχεδίασης και υποσχεδιάσεων.

3.2.3 Αναφορές Σύνθεσης Σχεδίασης – Αποτελέσματα

- Κατανάλωση Ισχύος

```
Global Operating Voltage = 3
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW
  Cell Internal Power  = 403.1745 mW   (70%)
  Net Switching Power  = 170.2716 mW   (30%)
                         ---------
Total Dynamic Power    = 573.4461 mW  (100%)

Cell Leakage Power     =   1.1647 uW
```

- Επιφάνεια κυκλώματος

```
Library(s) Used:

    GSCLib_2.0 (File: /home/adadalia/adspirop/my/lib/GSCLib_2.0_prelim3.db)

Number of ports:              802
Number of nets:               803
Number of cells:              400
Number of references:         400

Combinational area:       76768.031250
Noncombinational area:    18539.708984
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:          95307.835938
Total area:                 undefined
```

3.3 Επαλήθευση αποτελεσμάτων Σύνθεσης

Μετά την σύνθεση του RTL κώδικα σε τεχνολογικά εξαρτημένο πρέπει να επιβεβαιωθεί η σωστή λειτουργία του. Αυτό μπορεί να γίνει είτε με λειτουργική επαλήθευση είτε με τυπική απόδειξη ορθής λειτουργίας.

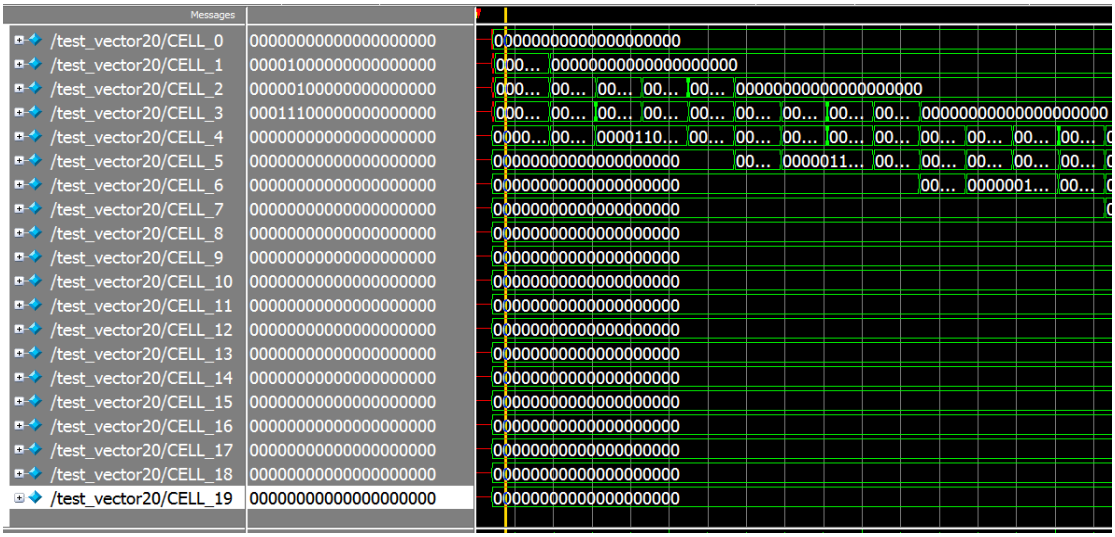
Λειτουργική Επαλήθευση: Κατά την διαδικασία αυτή εξετάζονται όλα τα πιθανά σενάρια εισόδων στο κύκλωμα ώστε να επαληθευτεί η έξοδος. Λόγω των ιδιαιτεροτήτων και της πολυπλοκότητας των κυκλωμάτων μια τέτοια διαδικασία είναι αρκετά πολύπλοκη και χρονοβόρα. Αυτή η διαδικασία μπορεί να αντικατασταθεί με αρκετά καλή ακρίβεια χρησιμοποιώντας τις μεθόδους της λογική προσομοίωσης όπως αναλύθηκαν στο προηγούμενο κεφάλαιο.

Τυπική Απόδειξη: Με την μέθοδο αυτή χρησιμοποιούνται πολύπλοκα μαθηματικά μοντέλα για την απόδειξη της ορθότητας της λειτουργίας του κυκλώματος. Η αύξηση της πολυπλοκότητας των ψηφιακών κυκλωμάτων έχει ωθήσει την βιομηχανία παραγωγής υλικού να στραφεί προς αυτή μέθοδο επαλήθευσης, καθώς η λογική προσομοίωση δεν μπορεί να καλύψει όλες τις πιθανές περιπτώσεις και τα λάθη σε ένα ολοκληρωμένο κύκλωμα έχουν σοβαρό εμπορικό αντίκτυπο για τις βιομηχανίες.

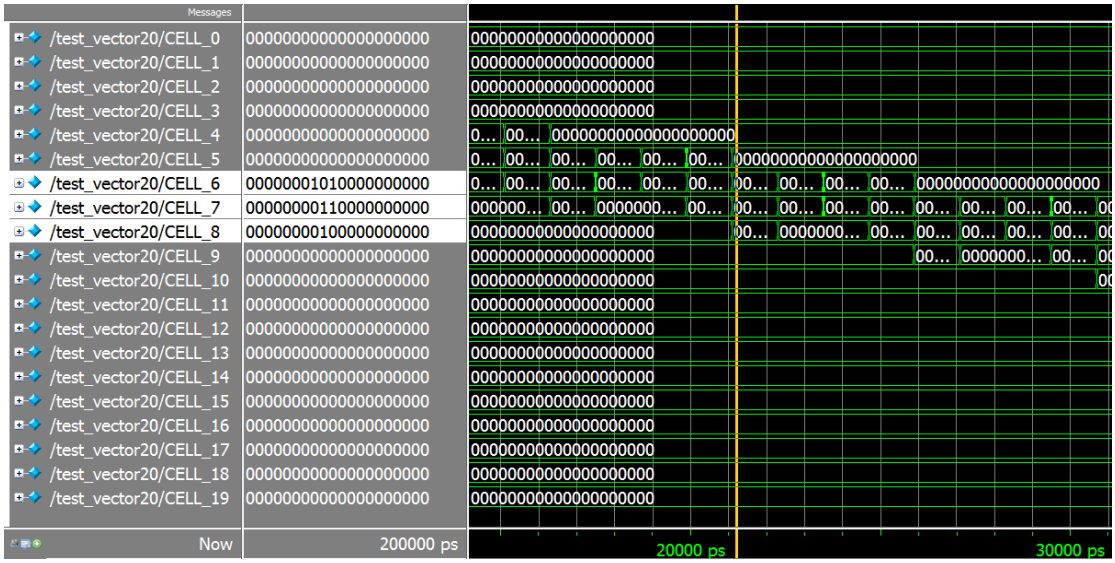
3.3.1 Επαλήθευση με προσομοίωση

Στο παράδειγμα αυτό η ορθότητα του τεχνολογικά εξαρτημένου κώδικα θα επαληθευτεί με λογική προσομοίωση στο εργαλείο ModelSim. Για να γίνει προσομοίωση χρειάζεται το αρχείο Verilog που έχει παράγει ο DC και η περιγραφή των βασικών κυττάρων σε γλώσσα περιγραφής υλικού, κάτι που παρέχεται στην τεχνολογική βιβλιοθήκη. Για την προσομοίωση χρησιμοποιείται επίσης και το αρχείο καθυστερήσεων .sdf.

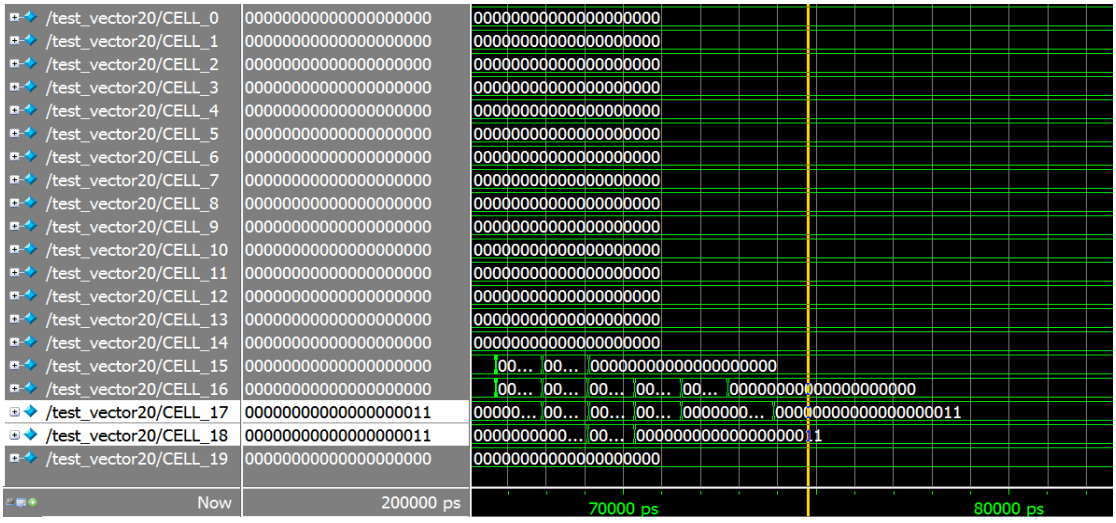
Στην συνέχεια τροποποιείται το αρχείο δοκιμής περιλαμβάνει τα δύο νέα αρχεία. Επίσης χρειάζεται να αυξηθεί και η περίοδος ρολογιού καθώς το κύκλωμα έχει μεγαλώσει αρκετά σε σχέση με την προηγούμενη περίπτωση προσομοίωσης. Σε αυτή τη περίπτωση χρησιμοποιείται το πρότυπο Glider το οποίο μπορεί να μετακινηθεί από την πάνω αριστερή γωνία έως την κάτω δεξιά.



Αρχική Κατάσταση πανελ.



Μετά από μερικές επαναλήψεις.

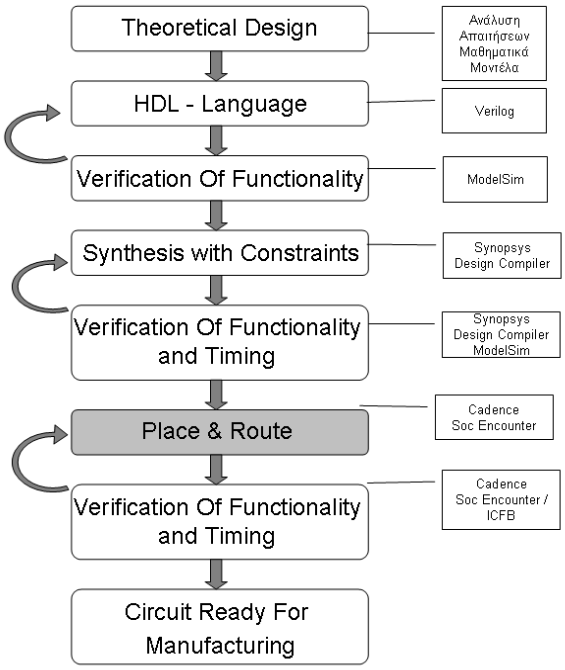


Τέλος του παιχνιδιού.

4 *Τοποθέτηση & Διασύνδεση (Place & Route)*

Μετά την δημιουργία του βελτιστοποιημένου τεχνολογικά εξαρτημένου κώδικα της σχεδίασης και την επαλήθευση της σωστής λειτουργίας, ακολουθεί η τοποθέτηση και η διασύνδεση των δομικών στοιχείων σε φυσική σχεδίαση. Η διαδικασία αυτή όπως δείχνει και το όνομα της αποτελείται από 2 στάδια

1. Τοποθέτηση (Place): Σε αυτό το στάδιο τα βασικά κύτταρα της σχεδίασης τοποθετούνται στις θέσεις τους πάνω στην φυσική σχεδίαση με σκοπό να ελαχιστοποιηθεί η επιφάνεια και χρονισμός του κυκλώματος. Η σχεδίαση έχει προκαθορισμένες διαστάσεις που δηλώνονται από τον σχεδιαστή καθώς και από τις προδιαγραφές της τεχνολογικής βιβλιοθήκης. Σκοπός της τοποθέτησης είναι η εκπλήρωση των χρονικών περιορισμών, η όσο το δυνατόν πυκνότερη τοποθέτηση και η αποσυμφόρηση των καλωδιώσεων.



2. Διασύνδεση (Route): Εφόσον τα κελιά έχουν τοποθετηθεί στην κατάλληλη θέση σκοπός του εργαλείου είναι η διασύνδεση των τυποποιημένων κυττάρων με καλωδιώσεις μεταξύ τους, καθώς και η διασύνδεση των ειδικών δικτύων της

σχεδίασης (δίκτυα τροφοδοσίας και γείωσης). Σκοπός του διασυνδετή είναι η χρήση όσο το δυνατών λιγότερων επιπέδων μετάλλων και στις μικρότερες διαστάσεις ώστε να μην γίνεται παράβαση των κανόνων DRC που ορίζει η τεχνολογική βιβλιοθήκη.

Η διαδικασία τοποθέτησης και διασύνδεσης εξαρτάται από πολλούς παράγοντες. Οι χρονικοί περιορισμοί παίζουν σημαντικό ρόλο για κυκλώματα υψηλής απόδοσης. Ο ορισμός της χωροθέτησης του ολοκληρωμένου κυκλώματος (floorplan) είναι σημαντικός παράγοντας καθώς ορίζει την θέση των τυποποιημένων κυττάρων όπως επίσης ο ορισμός και η σχεδίαση του δέντρου ρολογιού (Clock Tree Synthesis) και των ειδικών δικτύων τάσης - γείωσης (Power Planning).

Οι διαδικασίες αυτές γίνονται από αυτοματοποιημένα εργαλεία σχεδίασης. Στην εργασία αυτή χρησιμοποιήθηκε το βιομηχανικό, και πλέον διαδεδομένο, εργαλείο SoC Encounter της Cadence.

4.1 Προετοιμασία – Συλλογή Πληροφοριών

Η διαδικασία P&R χρειάζεται ένα αρκετά μεγάλο αριθμό πληροφοριών που πρέπει να συγκεντρωθούν.

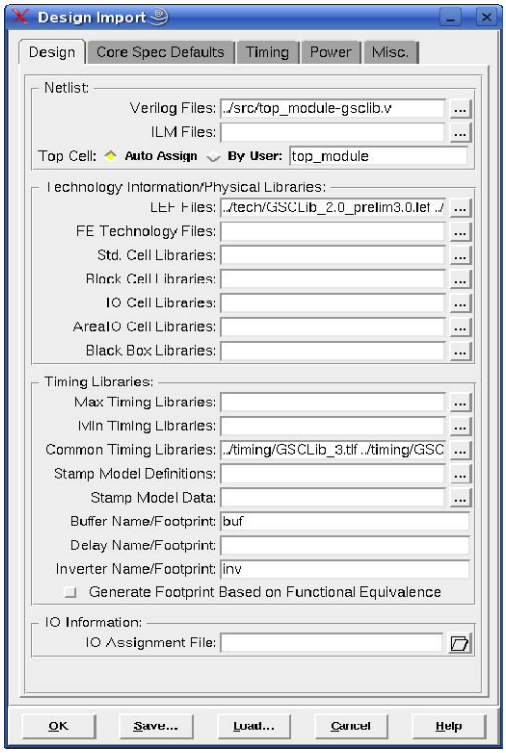
1. Φυσική Περιγραφή των Κανόνων Σχεδίασης και Απεικόνιση της φυσικής μορφής των Βασικών Κυττάρων και Έτοιμων Τμημάτων
    - Απαιτείται ένα αρχείο τύπου .LEF (Library Exchange Format) το οποίο να ορίζει τους φυσικούς κανόνες, λ.χ. το ελάχιστο πλάτος μετάλλου, την ελάχιστη απόσταση δυο ειδών μετάλλου, κτλ.
    - Επιπλέον το αρχείο τύπου .LEF, ορίζει φυσικά χαρακτηριστικά για είτε τα βασικά κύτταρα είτε για έτοιμες μονάδες, λ.χ. μνήμες SRAM
    - Μπορεί να υπάρχουν πάνω από ένα LEF αναλόγως με την χρήση έτοιμων μονάδων
2. Αρχείο Οργάνωσης Επιθεμάτων Εισόδων-Εξόδων
    - Απαιτείται ένα αρχείο περιγραφής Εισόδων-Εξόδων (.io αρχείο) που να ορίζει την θέση του κάθε επιθέματος στον χώρο γύρω από τον πυρήνα
    - Το Αρχείο αυτό περιγράφει τον προσανατολισμό και την θέση του κάθε επιθέματος στις τέσσερις πλευρές του πυρήνα (Βόρεια, Νότια, Ανατολικά και Δυτικά)
    - Εισάγει επιπλέον τα ειδικά επιθέματα για την Τάση/Γείωση και οι τέσσερις Γωνίες
    - Τα επιθέματα και οι γωνίες πρέπει να κλείνουν το πλαίσιο, με ειδικά γεμίσματα (fillers), για να διοχετεύεται το ρεύμα σε όλα τα επιθέματα και τελικά στον πυρήνα

3. Αρχείο Περιγραφής Χρονισμού Βιβλιοθήκης
    - Η μορφή .lib, που χρησιμοποιείται και στην λογική σύνθεση, είναι απαραίτητη για την εισαγωγή του χρονισμού των κυττάρων και την διαδικασία χρονικής ανάλυσης
    - Υποστηρίζονται και επιπλέον μορφές, όπως .tlf (timing library file) ή .ctlf (compiled timing library file)
4. Η σχεδίαση σε μορφή Verilog ως τεχνολογικά εξαρτημένος κώδικας (Netlist) στην βιβλιοθήκη υλοποίησης
5. Χρονικοί περιορισμοί για το σχέδιο σε μορφή .SDC (Synopsys Design Constraints)

4.2 Ροή Σχεδίασης

4.2.1 Εισαγωγή Σχεδίασης στον SoC Encounter

Όλες οι παραπάνω πληροφορίες που έχουν συγκεντρωθεί πρέπει να εισαχθούν στο εργαλείο σχεδίασης. Η εισαγωγή γίνεται κυρίως από αρχείο ορισμού παραμέτρων (Configuration File .conf). Σε αυτό το αρχείο δηλώνονται οι κατάλογοι όπου βρίσκονται οι σχεδιάσεις, τα αρχεία περιορισμών καθώς και τα αρχεία περιγραφής των τεχνολογικών βιβλιοθηκών. Στο εργαλείο όλες οι ενέργειες μπορούν να γίνουν είτε δίνοντας εντολές στην κονσόλα της εφαρμογής είτε χρησιμοποιώντας το γραφικό περιβάλλον είτε χρησιμοποιώντας αρχεία δέσμης ενεργειών Tcl.



Design Import στο γραφικό περιβάλλον του εργαλείου.

Επίσης σημαντικές παράμετροι που πρέπει να ορισθούν με την είσοδο της σχεδίασης στη μνήμη του προγράμματος είναι το αρχείο χρονικών (.sdc) περιορισμών που έχει δημιουργηθεί κατά την διάρκεια της σύνθεσης, ο ορισμός των ονομάτων των δικτύων τάσης – γείωσης όπως αναφέρονται στο αρχείο περιγραφής (LEF) της τεχνολογικής βιβλιοθήκης και τα αρχεία περιγραφής επιθεμάτων εισόδου εξόδου.

Αντίστοιχες εντολές για την εισαγωγή σχεδίασης

```
loadConfig  $CONF_FILE_NAME.conf
commitConfig
```

4.2.2 Χρονική ανάλυση Καλύτερης Περίπτωσης

Μετά την είσοδο της σχεδίασης στην μνήμη μπορεί να γίνει χρονική ανάλυση. Αυτή θεωρείται ως ανάλυση καλύτερης περίπτωσης καθώς ακόμη δεν έχουν υπολογιστεί οι καθυστερήσεις των αγωγών και διασυνδέσεων.

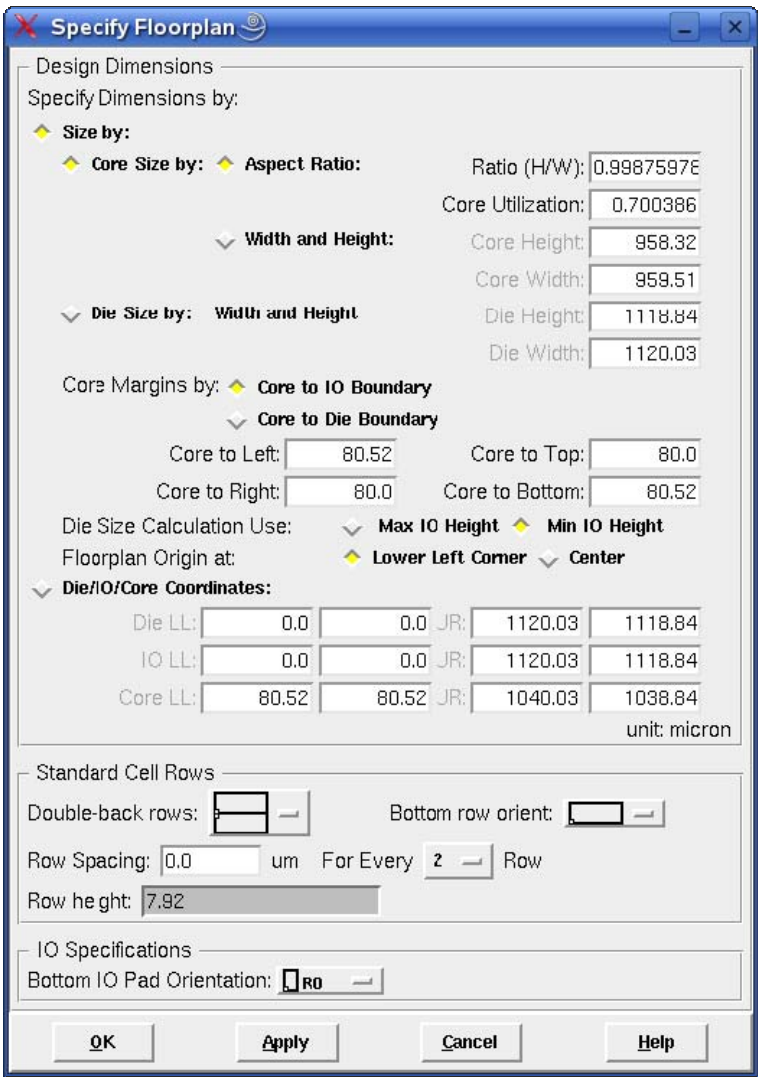
```
setCteReport
setAnalysisMode -setup -async -skew -autoDetectClockTree
buildTimingGraph -ignoreNetLoad
```

Με τις εντολές ενεργοποιούμε την Common Time Engine (CTE) του εργαλείου για την χρονική ανάλυση της σχεδίασης. Με τις παραμέτρους -setup, ελέγχεται το κύκλωμα για παραβιάσεις ανάθεσης (setup), `-async`, ελέγχεται για περιορισμούς set/reset, `-skew`, ελέγχεται στην χρονική ανάλυση η απόκλιση του δέντρου ρολογιού, `-autoDetectClockTree` δημιουργείται το δέντρο ρολογιού για την χρονική ανάλυση. Με αυτόν τον τρόπο εξάγονται τα μονοπάτια με τις μεγαλύτερες καθυστερήσεις και χρησιμοποιούνται ως συγκριτικό μετρό για τα επόμενα βήματα της σχεδίασης. Παρακάτω αναφέρεται ένα κρίσιμο μονοπάτι για την αρχική χρονική ανάλυση της σχεδίασης το οποίο καλύπτει τους χρονικούς περιορισμούς..

```
Path 1: MET External Delay Assertion
Endpoint:   C[0]          (v) checked with  leading edge of 'clock'
Beginpoint: cl_0/C0_reg/Q (v) triggered by leading edge of 'clock'
Other End Arrival Time     0.000
- External Delay           0.000
+ Path Delay               0.500
- Uncertainty              0.300
= Required Time            0.500
- Arrival Time             0.391
= Slack Time               0.109
     Clock Rise Edge                 0.000
     = Beginpoint Arrival Time       0.000
     +-------------------------------------------------------------------------+
     |      Pin       |     Arc     |    Cell    | Delay | Arrival |  Slew |  Load |
     |                |             |            |       |  Time   |       |       |
     |----------------+-------------+------------+-------+---------+-------+-------|
     | clock          | clock ^     |            |       |   0.000 | 0.000 |       |
     | cl_0/C0_reg/CK |             | DFFSRX1    | 0.000 |   0.000 | 0.000 |       |
     | cl_0/C0_reg/Q  | CK ^ -> Q v | DFFSRX1    | 0.105 |   0.105 | 0.036 |       |
     | cl_0/U8/A      |             | INVX1      | 0.000 |   0.105 | 0.036 | 0.006 |
     | cl_0/U8/Y      | A v -> Y ^  | INVX1      | 0.103 |   0.208 | 0.134 |       |
     | cl_0/U3/A      |             | INVX8      | 0.000 |   0.208 | 0.134 | 0.060 |
     | cl_0/U3/Y      | A ^ -> Y v  | INVX8      | 0.183 |   0.391 | 0.166 |       |
     | C[0]           |             | top_module | 0.000 |   0.391 | 0.166 | 0.589 |
     +-------------------------------------------------------------------------+
```

4.2.3 Χωροθέτηση – Floorplaning

Μετά την εισαγωγή και μια πρώτη χρονική ανάλυση της σχεδίασης ακολουθεί ο ορισμός των διαστάσεων του πυρήνα (core) και της ψηφίδας (die).



Σε αυτό το στάδιο καθορίζεται η αναλογία των διαστάσεων (aspect ratio), συνήθως είναι 1 για να τετράγωνος ο πυρήνας. Ο τετράγωνος πυρήνας έχει μεγαλύτερη ωφελιμότητα για την διασύνδεση των βασικών κυττάρων. Η ωφελιμότητα του πυρήνα (Core Utilization), συνήθως κυμαίνεται από 0.5 έως 0.75, σε αυτό το παράδειγμα η ωφελιμότητα 0.70 που σημαίνει ότι το 70% του εμβαδού του πυρήνα θα χρησιμοποιηθεί για την τοποθέτηση των βασικών κυττάρων και το υπόλοιπο 30% για την διασύνδεση αυτών. Καθορίζονται οι διαστάσεις για το ύψος και το πλάτος (Core Width, Height) του πυρήνα και τα περιθώρια του από την άκρη της ψηφίδας (Core to I/O Boundaries). Το ύψος των σειρών που τοποθετούνται τα βασικά κύτταρα (standard cell row height) είναι προκαθορισμένο από την τεχνολογία που χρησιμοποιείται.

Ο ορισμός των διαστάσεων μπορεί να οριστεί επίσης με την εντολή

```
floorplan -d <dieW> <dieH> <coreToLeft> <coreToBottom> <coreToRight> <coreToTop>
```

Κατά την διαδικασία της χωροθέτησης σε μια ιεραρχικά δομημένη σχεδίαση οι υποσχεδιάσεις (blocks, στοιχεία μνήμης κτλ), μπορούν να τοποθετηθούν από τον σχεδιαστή στις επιθυμητές θέσεις. Αυτή η διαδικασία ονομάζεται ιεραρχική χωροθέτηση με τμήματα (chip Partition). Στο παράδειγμα αυτής της εργασίας η σχεδίαση είναι επίπεδη (flat).

4.2.4 Δημιουργία Δικτύου Τάσης – Γείωσης

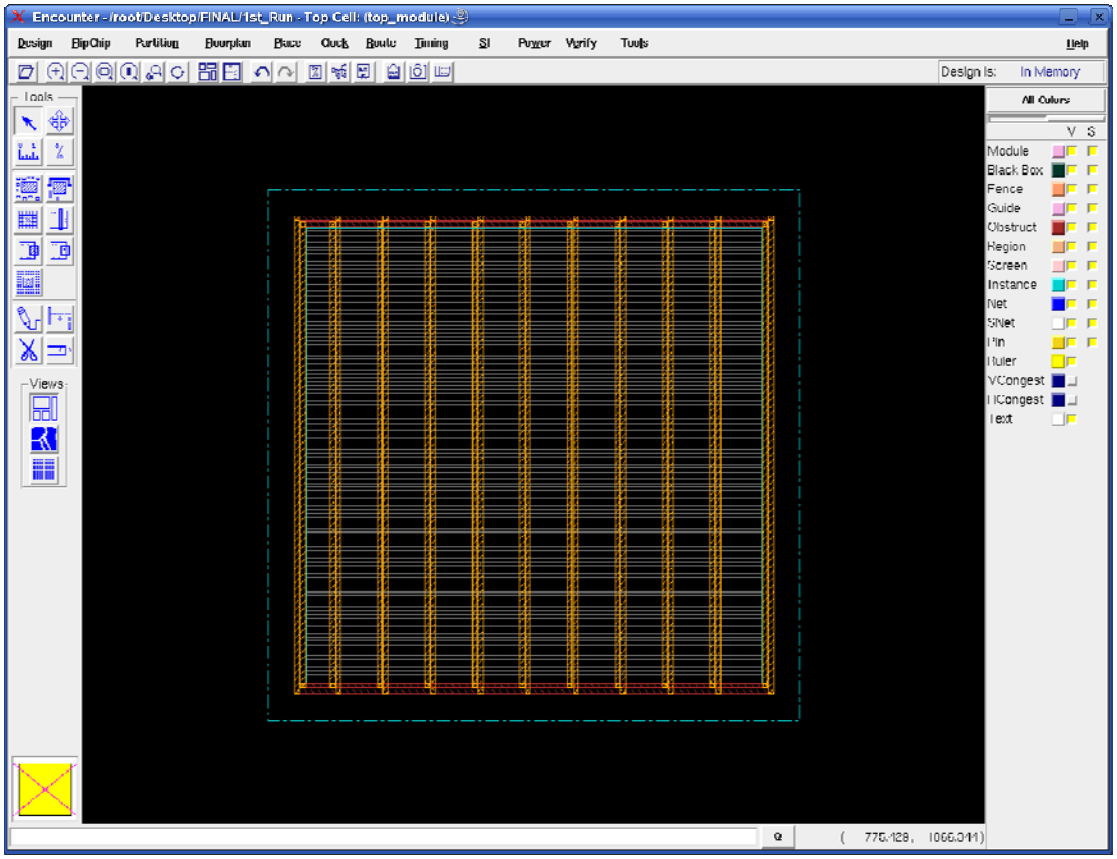
Μετά την χωροθέτηση έπεται η δημιουργία του ειδικού δικτύου τάσης – γείωσης. Το δίκτυο αποτελείται από ένα δακτύλιο γύρω από τον πυρήνα και από κάθετες λωρίδες μετάλλου μέσα στον πυρήνα.

```
addRing -spacing_bottom 3 -width_left 10 -width_bottom 10 -width_top 10 -spacing_top 3 -layer_bottom Metal5 -stacked_via_top_layer Metal6 -width_right 10 -offset_bottom 2 -layer_top Metal5 -offset_left 2 -spacing_right 3 -spacing_left 3 -offset_right 2 -offset_top 2 -layer_right Metal6 -stacked_via_bottom_layer Metal1 -layer_left Metal6 -nets {VDD VSS }
```

Η εντολή προσθέτει 2 δακτυλίους έναν για την τάση και έναν για την γείωση. Σε αυτούς ορίζονται οι αποστάσεις από τον πυρήνα με την παράμετρο `–offset`, η απόσταση μεταξύ των δακτυλίων ορίζεται με την παράμετρο `–spacing,` η παράμετρος `–width` ορίζει το πάχος των μετάλλων. Με την παράμετρο `–layer` ορίζεται ο τύπος μετάλλου. Στους οριζόντιους αγωγούς έχει επιλεγεί Metal 5 καθώς σύμφωνα με τους κανόνες σχεδίασης της τεχνολογίας πρέπει να έχει οριζόντιο προσανατολισμό. Ομοίως και για τους κάθετους αγωγούς έχει επιλεγεί Metal 6. Η παράμετρος `-nets { VDD VSS }` δίνει τα ονόματα στους αγωγούς σύμφωνα με τις προδιαγραφές της τεχνολογίας για τα ονόματα των δικτύων τάσης – γείωσης.

```
addStripe -set_to_set_distance 100 -stacked_via_top_layer Metal6 -spacing 2 -xleft_offset 50 -merge_stripes_value 0.33 -layer Metal6 -width 10 -stacked_via_bottom_layer Metal1 -nets { VDD VSS }
```

Η εντολή προσθέτει κάθετες παροχές στον πυρήνα . Η παράμετροι παρομοίως δηλώνουν τα μέταλλα που θα χρησιμοποιηθούν, το μέγεθος τους, τις αποστάσεις μεταξύ τους, καθώς και την απόσταση από πυρήνα.



Η σχεδίαση μετά από τη δημιουργία του δικτύου.

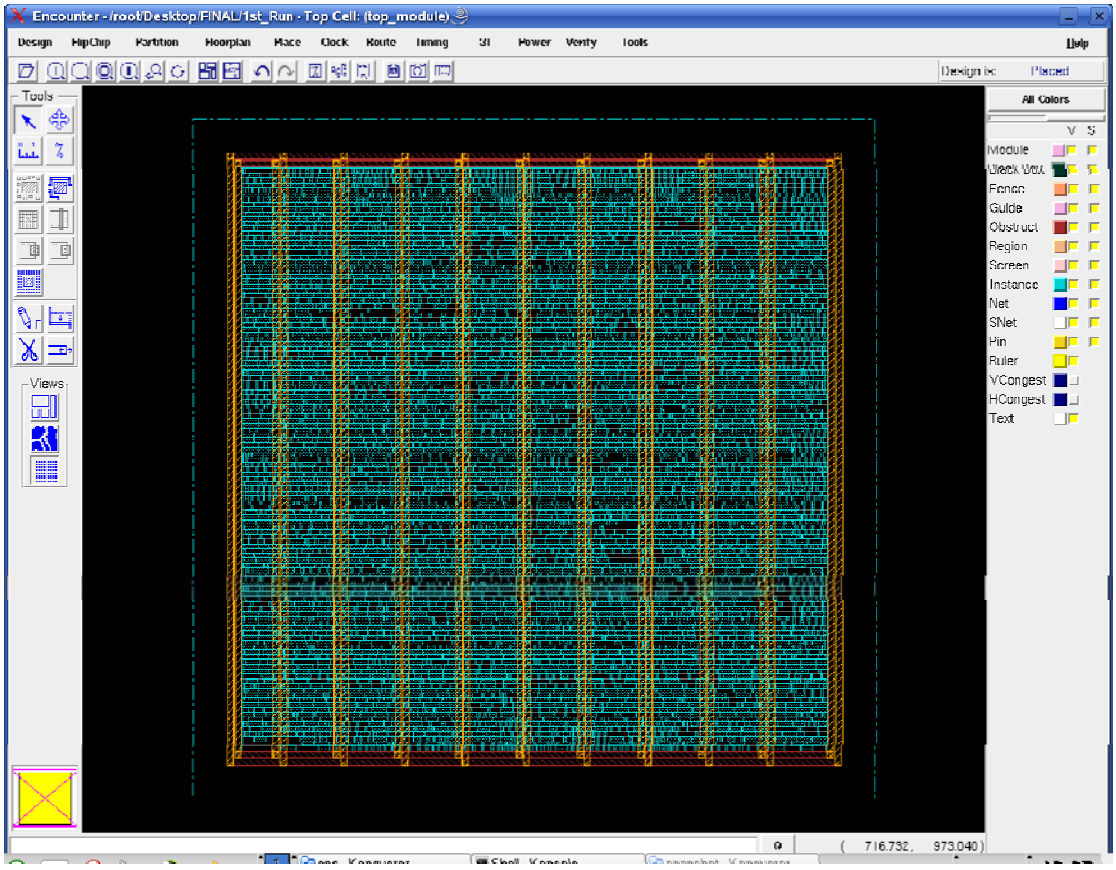
4.2.5 Τοποθέτηση Βασικών κυττάρων – Placement

Μετά τον ορισμό της επιφάνειας του ολοκληρωμένου και την τοποθέτηση των αγωγών τάσης και γείωσης πρέπει να τοποθετηθούν τα βασικά κύτταρα στις καθορισμένες σειρές που έχουν δημιουργηθεί. Σκοπός είναι να τοποθετηθούν τα βασικά κύτταρα όσον το δυνατόν πυκνότερα και να ικανοποιηθούν όσο περισσότερο οι χρονικοί περιορισμοί και να μειωθεί όσο το δυνατόν η συμφόρηση των αγωγών για την καλύτερη διασύνδεση σε επόμενο βήμα. Η τοποθέτηση είναι μια επαναληπτική διαδικασία στην οποία αρχικά τοποθετούνται τα βασικά κύτταρα στις θέσεις τους και έπειτα με επαναλήψεις αλλάζουν θέσεις μεταξύ τους ώστε να καλυφθούν οι περιορισμοί που έχουν τεθεί.

Δίνοντας την εντολή

amoebaPlace `-highEffort -timingdriven`

Καλείται το πρόγραμμα να τοποθετήσει τα βασικά κύτταρα με την παράμετρο –`highEffort` το εργαλείο προσπαθεί να καταβάλει όσες περισσότερες επαναλήψεις χρειάζονται ώστε να καλυφθούν οι περιορισμοί. Με την παράμετρο `–timingdriven` το εργαλείο δίνει έμφαση στην τοποθέτηση των βασικών κυττάρων που αποτελούν κρίσιμο μονοπάτι, ώστε να καλύπτουν τις χρονικές προδιαγραφές που έχουν δοθεί, ενώ στα λιγότερο κρίσιμα δίνει έμφαση στην αποφυγή συμφόρησης για τον δρομολογητή.



Η σχεδίαση με τοποθετημένα τα βασικά κύτταρα.

Κατά την τοποθέτηση παράγονται αναφορές στο αρχείο καταγραφής του εργαλείου για την πυκνότητα του κυκλώματος, το προβλεπόμενο μήκος των αγωγών διασύνδεσης . Αναφορές για τα στοιχεία που έχουν αλλάξει θέσεις και τον χρόνο ολοκλήρωσης της διαδικασίας.

```
Estimated cell power/ground rail width = 0.990 um
Average module density = 0.700.
Density for the design = 0.700.
       = stdcell_area 123137 (643662 um^2) / alloc_area 175813 (919010 um^2)
Statistics of distance of Instance movement in detailed placement:
```

```
  maximum (X+Y) =        59.40 um
  inst (gen_cell_logic_4[241].inner[244].cl_4/add_25_7_DP_OP_44_7899_1/U3)
with max move: (595.32, 832.92) -> (654.72, 832.92)
  mean    (X+Y) =         8.43 um
Total instances moved : 2653
*** cpu=0:00:00.1   mem=77.7M  mem(used)=0.0M***
Total net length = 7.236e+05 (4.383e+05 2.853e+05) (ext = 2.428e+05)
*** End of Placement (cpu=0:00:20.7, real=0:00:21.0, mem=77.7M) ***

Begin checking placement ...
*info: Placed = 6016
*info: Unplaced = 0
Placement Density:70.04%
```

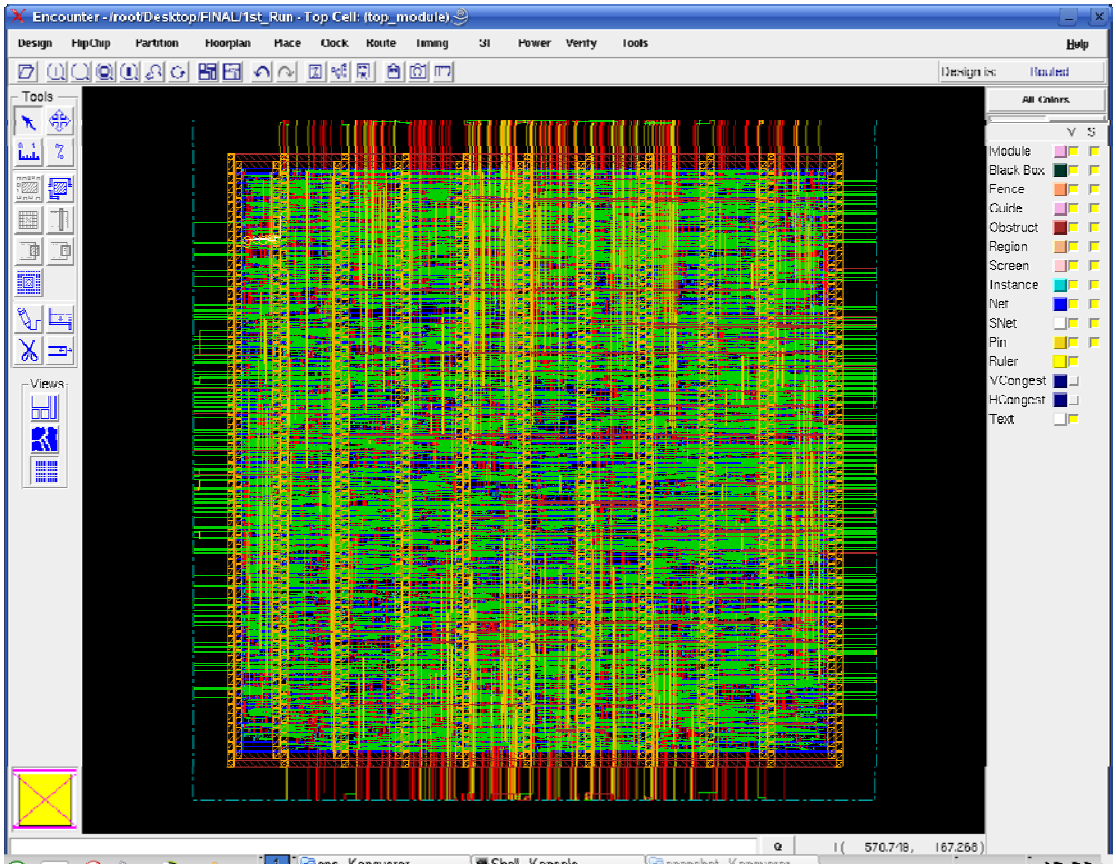
Μετά την τοποθέτηση πρέπει να γίνει χρονική ανάλυση της σχεδίασης για να εντοπιστούν οι μεταβολές στον χρονισμό του κυκλώματος σε σχέση με την αρχική ιδανική περίπτωση. Πριν την χρονική ανάλυση πρέπει να διασυνδεθούν τα βασικά κύτταρα. Μια ικανοποιητική και λιγότερο απαιτητική διαδικασία σε σχέση με την λεπτομερή διασύνδεση είναι η δοκιμαστική διασύνδεση (Trial Route).

4.2.6 Δοκιμαστική Διασύνδεση – Trial Route

Με την δοκιμαστική διασύνδεση μπορεί να γίνει μια εκτίμηση της συμφόρησης των αγωγών του κυκλώματος επίσης μια εκτίμηση των παρασιτικών καθυστερήσεων. Δημιουργεί κανονικούς αγωγούς διασύνδεσης ώστε να μπορεί να γίνει ανάλυση καθυστέρησης RC. Με την δοκιμαστική διασύνδεση παρέχεται μια εκτίμηση της λειτουργίας και της εκπλήρωσης των σχεδιαστικών στόχων σε πρώιμο στάδιο της ροής σχεδίασης.

Δίνοντας την εντολή

`trialRoute` `-lowEffort`



Το κύκλωμα μετά την δοκιμαστική διασύνδεση.

Στην κονσόλα του εργαλείου εμφανίζονται αναφορές για το εκτιμώμενο μήκος των αγωγών μετάλλου, το μήκος των αγωγών κάθε επιπέδου μετάλλου και της εκτιμώμενης συμφόρησης του κυκλώματος καθώς και των αριθμό των διασυνδέσεων μεταξύ των μετάλλων.

```
Est net length = 7.915e+05um = 4.742e+05H + 3.173e+05V
.
…
…
Usage: (14.2%H 9.1%V) = (5.061e+05um 4.906e+05um) = (148723 62680)
Overflow: 0 = 0 (0.00% H) + 0 (0.00% V)
..
….
*** Memory Usage v0.70 (Current mem = 82.211M, initial mem = 51.500M) ***
Phase 1l route (0:00:00.5 78.2M):
Cleanup 3349 multi-gpin terms
Prep phase2 (0:00:00.0 77.7M):
```

```
Total length: 7.911e+05um, number of vias: 46905
M1(H) length: 0.000e+00um, number of vias: 22326
M2(V) length: 2.378e+05um, number of vias: 23212
M3(H) length: 4.515e+05um, number of vias: 1113
M4(V) length: 7.495e+04um, number of vias: 213
M5(H) length: 2.231e+04um, number of vias: 41
M6(V) length: 4.594e+03um
*** Completed Phase 1 route (0:00:01.1 77.7M) ***

Phase 2a route (0:00:00.3 77.8M)
Phase 2b route (0:00:00.5 77.8M)

Total length: 8.042e+05um, number of vias: 47171
M1(H) length: 0.000e+00um, number of vias: 22592
M2(V) length: 2.489e+05um, number of vias: 23212
M3(H) length: 4.538e+05um, number of vias: 1113
M4(V) length: 7.460e+04um, number of vias: 213
M5(H) length: 2.237e+04um, number of vias: 41
M6(V) length: 4.548e+03um
*** Completed Phase 2 route (0:00:00.9 77.8M) ***

*** xroute (cpu=0:00:02.0 mem=77.8M) ***
```

Μετά την δοκιμαστική διασύνδεση μπορούν να εξαχθούν οι παρασιτικές καθυστερήσεις του κυκλώματος.

```
Nr. Extracted Resistors     : 84884
Nr. Extracted Ground Cap.   : 66147
Nr. Extracted Coupling Cap. : 13736

##############################################################################################
# FirstEncounter Capacitance File                                                            #
#    Format: <netName> <wireCap> <pinCap> <totalCap> <netLength> <wireCapPerUnitLen>
<nrFanout> #
#   The wireCap, pinCap and totalCap are in the unit of pF and                               #
#   the netLength is in the unit micron.                                                     #
#   the capPerUnitLen is in pF per micron.                                                   #
#                     scaling                     factor:                      1.000000.
#
```

```
#################################################################################################
#
#   Print out the nets with capacitance in range: 0.000000 - 1000.000000 pf.
#
Panel[0]       0.013015       0.015536       0.028551       94.060  1.384e-04      2
Panel[1]       0.014746       0.015536       0.030282       105.940 1.392e-04      2
Panel[2]       0.013973       0.015536       0.029509       101.980 1.370e-04      2
Panel[3]       0.014537       0.015536       0.030073       105.940 1.372e-04      2
Panel[4]       0.013242       0.015536       0.028778       97.360  1.360e-04      2
……
……
…….
C[326]  0.084492       0.121856       0.206348       611.490 1.382e-04      10
C[327]  0.117804       0.121856       0.239660       853.050 1.381e-04      10
C[328]  0.142868       0.121856       0.264724       1022.010 1.398e-04     10
C[329]  0.087868       0.121856       0.209724       626.010 1.404e-04      10
C[330]  0.074226       0.121856       0.196082       529.650 1.401e-04      10
……..
……..
```

Μετά την εξαγωγή των παρασιτικών μπορεί να γίνει πάλι χρονική ανάλυση του κυκλώματος και να συγκριθεί ο χρονισμός με την αρχική εκτίμηση καλύτερης περίπτωσης.

Στον παρακάτω πίνακα φαίνονται η ολική καθυστέρηση και η χειρότερη καθυστέρηση ενός μονοπατιού του κυκλώματος και μια κατανομή αυτών.

```
*** Reported constrained paths (0:00:00.3) ***
    (src|dest)          *      clock
    ----------+----------+----------
 0:          *|    -0.263|    -0.263
 1:      clock|    -0.263|    -0.263

--------------------  -------    -------
 Slack Range (ns)      Count        Sum
--------------------  -------    -------
( -0.263 ~  -0.200]       14         14
( -0.200 ~   0.000]      277        291
--------------------  -------    -------
Total negative slacks(TNS)=-25.577
Worst negative slacks(WNS)=-0.263
```

Δίνεται το χειρότερο μονοπάτι και ο χρονισμός του αναλυτικά.

```
Path 1: VIOLATED External Delay Assertion
Endpoint:   C[130]                                          (v) checked with
leading edge of 'clock'
Beginpoint: \gen_cell_logic_4[121].inner[130].cl_4 /C0_reg/Q (v) triggered by
leading edge of 'clock'
Other End Arrival Time     0.000
- External Delay           0.000
+ Path Delay               0.500
- Uncertainty              0.300
= Required Time            0.500
- Arrival Time             0.763
= Slack Time              -0.263
Clock Rise Edge                 0.000
= Beginpoint Arrival Time       0.000
+-------------------------------------------------------------------------------
--------------------------+
|                       Pin                       |     Arc     |    Cell    | Delay
| Arrival |  Slew |  Load |
|                                                 |             |            |
|  Time   |       |       |
|-------------------------------------------------+-------------+------------+------
-+---------+-------+-------|
| clock                                           | clock ^     |            |
|   0.000 | 0.000 |       |
| \gen_cell_logic_4[121].inner[130].cl_4 /C0_reg/CK |             | DFFSRX1    | 0.000
|   0.000 | 0.000 |       |
| \gen_cell_logic_4[121].inner[130].cl_4 /C0_reg/Q  | CK ^ -> Q v | DFFSRX1    | 0.715
|   0.715 | 0.734 |       |
| C[130]                                          |             | top_module | 0.048
|   0.763 | 0.734 | 0.409 |
+-------------------------------------------------------------------------------
--------------------------+
```

Παρατηρείται ότι σε αυτή την περίπτωση υπάρχουν μονοπάτια τα οποία δεν καλύπτουν τους αρχικούς χρονικούς περιορισμούς. Οι παραβάσεις αυτές μπορούν να επιλυθούν με προσθήκη ενισχυτών (buffers) είτε με αλλαγή του μεγέθους των πυλών.

4.2.7 Σύνθεση Δέντρου Ρολογιού – Clock Tree Synthesis

Με την σύνθεση του δέντρου ρολογιού το εργαλείο προσπαθεί να αναλύσει το ρολόι του κυκλώματος ώστε προσθέτοντας μονάδες καθυστέρησης buffers ή inverters , να μειώσει ή να εξαλείψει την χρονική απόκλιση του ρολογιού. Το δέντρο ρολογιού είναι μια σχεδίαση η οποία εκτός της χρονικής απόκλισης παρουσιάζει και άλλα σχεδιαστικά προβλήματα. Τέτοια είναι, το ρολόι είναι μια αρκετά ενεργοβόρα σχεδίαση, παρουσιάζει συχνά το φαινόμενο της ηλεκτρομετανάστευσης, έχει μεγάλο φορτίο στους αγωγούς αυξάνοντας την συνολική καθυστέρηση. Η σύνθεση του ρολογιού είναι πολύ σημαντική διαδικασία για αυτό το στάδιο της ψηφιακής σχεδίασης ολοκληρωμένων κυκλωμάτων.

Αρχικά δημιουργείται ένα αρχείο με τις προδιαγραφές του ρολογιού (Clock Tree Specification file). Σε αυτό το αρχείο ορίζονται το όνομα του ρολογιού, η μεγαλύτερη και μικρότερη καθυστέρηση στο σήμα του ρολογιού, τον στόχο για την εξάλειψη της απόκλισης και τον μέγιστο χρόνο μετάβασης των εισόδων ρολογιού των βασικών κυττάρων. Το αρχείο αυτό δημιουργείται με την εντολή

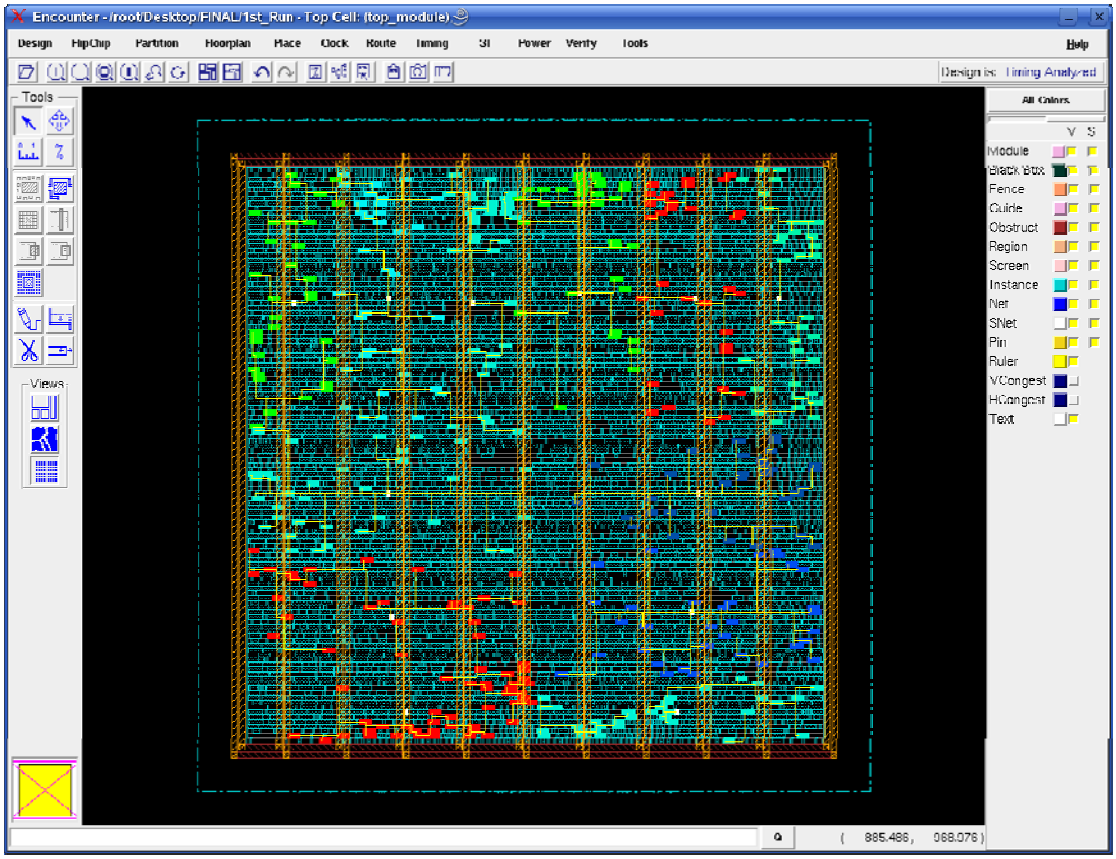
```
createClockTreeSpec – output $top.ctstch
–bufFootprint buf
–invFootprint inv
```

Εφόσον δημιουργηθεί το αρχείο περιγραφής προδιαγραφών ο σχεδιαστής μπορεί να το τροποποιήσει αναλόγως με τις απαιτήσεις της σχεδίασης

```
specifyClockTree -clkfile $top.ctstch
checkUnique
```

Μετά τον ορισμό των αρχείων περιγραφής το εργαλείο είναι έτοιμο να σχεδιάσει το δέντρο του ρολογιού και να παράγει τις αναφορές του.

```
ckSynthesis –rguide report/$top.rguide
-report report/$top.ctsrpt
-macromodel report/$top.ctsmdl
-fix_added_buffers
```



Τα χρωματιστά κελιά είναι τα στοιχεία που προσθέτει η σχεδίαση του ρολογιού

Αναφορά ρολογιού.

```
**** Clock Tree clock Stat ****
Total Clock Level      : 3
***** Top Nodes *****
clock  delay[0(ps)  0(ps)]  (     clock__L1_I3/A     clock__L1_I2/A     clock__L1_I1/A
clock__L1_I0/A )
Level 3 (Total=400     Sink=400)
Level 2 (Total=12      Sink=0  INVX8=12)
Level 1 (Total=4       Sink=0  INVX8=4)
Total Sinks            : 400

********** Clock clock Pre-Route Timing Analysis **********
Nr. of Subtrees                : 1
Nr. of Sinks                   : 400
Nr. of Buffer                  : 16
Nr. of Level (including gates) : 2
Max  trig.  edge  delay  at  sink(R):  gen_cell_logic_4[361].inner[373].cl_4/C0_reg/CK
391(ps)
Min  trig.  edge  delay  at  sink(R):  gen_cell_logic_4[21].inner[23].cl_4/C0_reg/CK
300.4(ps)
                                (Actual)              (Required)
```

```
Rise Phase Delay              : 300.4~391(ps)         0~10000(ps)
Fall Phase Delay              : 312.7~404.9(ps)       0~10000(ps)
Trig. Edge Skew               : 90.6(ps)              300(ps)
Rise Skew                     : 90.6(ps)
Fall Skew                     : 92.2(ps)
Max. Rise Buffer Tran.        : 155.6(ps)             400(ps)
Max. Fall Buffer Tran.        : 155.6(ps)             400(ps)
Max. Rise Sink Tran.          : 197.5(ps)             400(ps)
Max. Fall Sink Tran.          : 178(ps)               400(ps)
```

Εφόσον σχεδιαστεί το δέντρο ρολογιού πρέπει αυτό να διασυνδεθεί με την υπόλοιπη σχεδίαση και πάλι να εξαχθούν αναφορές για τον χρονισμό και τις παρασιτικές καθυστερήσεις του κυκλώματος.

Στην αναφορά χρονικής ανάλυσης παρατηρείται ότι η καθυστέρηση του κυκλώματος έχει αυξηθεί κάτι το οποίο αναμενόταν λόγω της πρόσθεσης επιπλέον κυττάρων στο κύκλωμα από την σύνθεση του δέντρου ρολογιού.

```
*** Reported constrained paths (0:00:00.3) ***
    (src|dest)          *      clock
    ----------+----------+----------
 0:          *|    -0.760|    -0.760
 1:      clock|    -0.760|    -0.760
--------------------  -------    -------
 Slack Range (ns)      Count        Sum
--------------------  -------    -------
( -0.760 ~  -0.600]       71         71
( -0.600 ~  -0.400]      261        332
( -0.400 ~  -0.200]       67        399
( -0.200 ~   0.000]        1        400
--------------------  -------    -------
Total negative slacks(TNS)=-200.434
Worst negative slacks(WNS)=-0.760
```

4.2.8 Επιτόπια Βελτιστοποίηση – In Place Optimization (IPO)

Μετά την τοποθέτηση και μετά και την διασύνδεση των αγωγών το σχέδιο έχει αλλάξει σημαντικά τον χρονισμό του λόγω της φυσικής πια διάστασης των πραγμάτων, όπου οι αποστάσεις μεταξύ των πυλών, μονάδων ή και επιθεμάτων συμβάλουν σημαντικά στην καθυστέρηση των μονοπατιών του κυκλώματος. Όλες αυτές οι φυσικές πληροφορίες δεν ήταν διαθέσιμες κατά την διαδικασία της Λογικής Σύνθεσης, συνεπώς είναι προφανές ότι το σχέδιο δεν είναι βελτιστοποιημένο ως προς την φυσική του μορφή.

Για την βελτίωση του χρονισμού μπορούμε να εκτελέσουμε μια διαδικασία Επιτόπιας Βελτιστοποίησης (In-Place Optimization), όπου τα μεγέθη των πυλών ή και η δομή τους αλλάζει για να βελτιωθεί η οδήγηση τους και να μειωθούν οι καθυστερήσεις λόγω απόστασης. Δίνοντας την εντολή με τις προεπιλεγμένες ρυθμίσεις.

```
setIPOMode -nofixFanoutLoad -noRestruct -noSwapPin -highEffort -FixDrc -addPortAsNeeded -incrTrialRoute -targetSlack 0.0 -maxDensity 0.95
```

Η IPO είναι μια επαναληπτική διαδικασία. Αρχικά αλλάζει το μέγεθος στα βασικά κύτταρα επαναληπτικά. Στο τέλος κάθε επανάληψης γίνεται χρονική ανάλυση για το να καλύπτονται οι περιορισμοί. Εφόσον δεν καλυφθούν με τις αλλαγές μεγέθους των κυττάρων η διαδικασία προχωρά στο επόμενο βήμα.

```
*** Starting new resizing for area recovery ***
density before resizing = 67.832%
11 instances have been resized
density change = -0.007%
*summary:     11 instances changed cell type
*      :     11 instances changed cell type from 'INVX2' to 'INVX1'
density after resizing = 67.825%
```

Στο επόμενο βήμα προστίθενται ενισχυτές (buffers) στα μονοπάτια ώστε να διορθωθεί ο χρονισμός στα επίπεδα των προδιαγραφών.

```
Summary:
* cell INVX1 (21um^2): 754 insts (area = 15765um^2)
* cell INVX2 (26um^2): 405 insts (area = 10585um^2)
* cell CLKBUFX1 (26um^2): 612 insts (area = 15995um^2)
* cell INVX4 (31um^2): 871 insts (area = 27317um^2)
* cell BUFX3 (31um^2): 112 insts (area = 3513um^2)
* cell CLKBUFX3 (31um^2): 230 insts (area = 7214um^2)
```

```
* cell BUFX1 (31um^2): 10 insts (area = 314um^2)
* cell CLKBUFX2 (31um^2): 3 insts (area = 94um^2)
* cell INVX8 (52um^2): 0 inst (area = 0um^2)
* ------------------------------ (area = 80797um^2)
2997 buffers added on 616 nets (with 23 drivers resized)

Density after buffering = 0.773186
```

Μετά την προσθήκη ενισχυτών γίνεται πάλι επαναληπτικά αλλαγή μεγέθους των κυττάρων είτε αλλαγή των θέσεων των εισόδων των βασικών κυττάρων είτε ακόμα και αναδιάρθρωση της τοποθέτησης. Μετά το τέλος της διαδικασίας εξάγεται η αναφορά χρονισμού.

```
*** Reported constrained paths (0:00:00.5) ***
    (src|dest)          *      clock
    ----------+----------+----------
 0:          *|    -0.563|    -0.563
 1:      clock|    -0.563|    -0.563
--------------------  -------    -------
 Slack Range (ns)      Count        Sum
--------------------  -------    -------
( -0.563 ~  -0.400]       31         31
( -0.400 ~  -0.200]      313        344
( -0.200 ~   0.000]       56        400
--------------------  -------    -------
Total negative slacks(TNS)=-113.739
Worst negative slacks(WNS)=-0.563
```

Παρατηρείται ότι η συνολική καθυστέρηση έχει βελτιστοποιηθεί κατά 43,25 % και η καθυστέρηση του χειρότερου μονοπατιού κατά 25,9%.

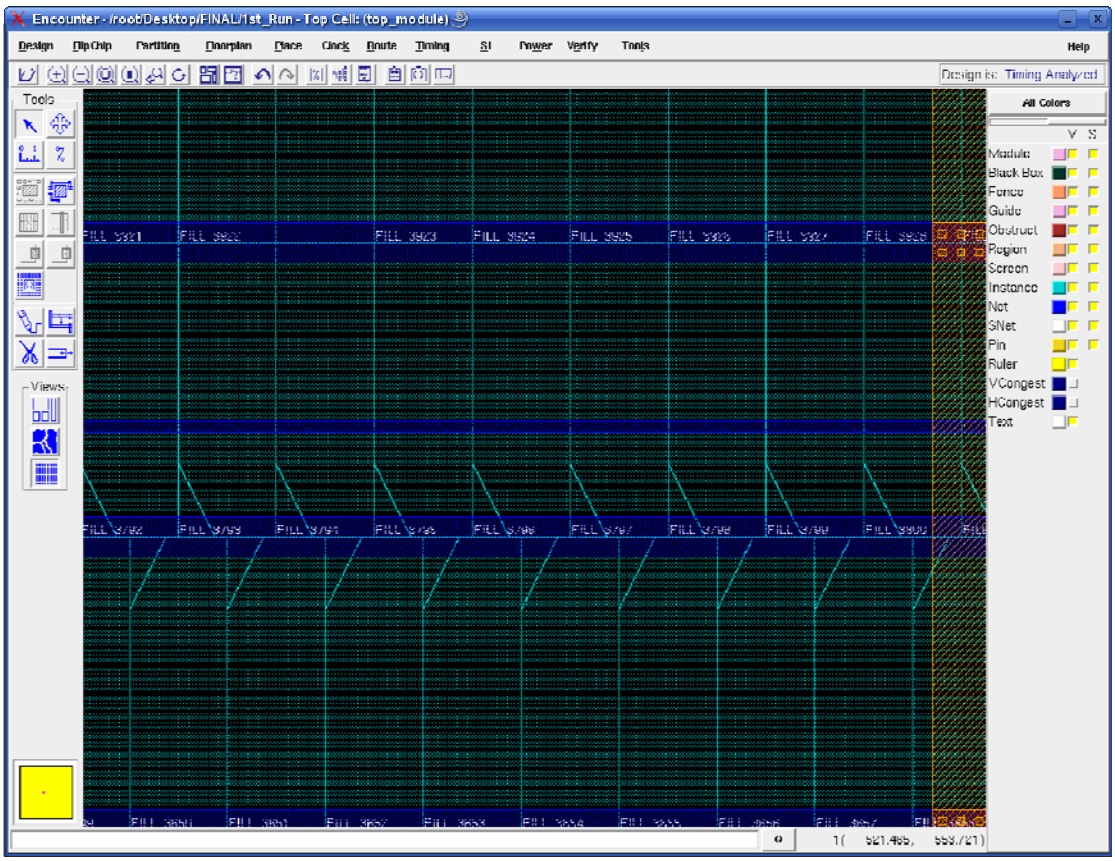
4.2.9 Προσθήκη Γεμισμάτων – Add Filler Cells

Το τελείωμα του σχεδίου είναι το τελικό στάδιο, όπου ο όποιος άδειος χώρος εσωτερικά του πυρήνα πρέπει να γεμίσει με γεμίσματα τα οποία περιέχουν επαφές στο υπόστρωμα, έτσι ώστε να μην δημιουργηθούν παρασιτικά μονοπάτια κατά μήκος του υποστρώματος σε κενά τμήματα.

Το γέμισμα επιτυγχάνεται με την παρακάτω εντολή:

```
addFiller -cell <filler cellname> -prefix <prefix name>
```

Οι βιβλιοθήκες διαθέτουν διάφορα μεγέθη γεμισμάτων, τα οποία γεμίζουν όλο τον κενό χώρο.



Τα γεμίσματα καλύπτουν τον κενό χώρο

4.2.10 Διασύνδεση Γραμμών τροφοδοσίας – Power Routing

Μετά την επιτυχή τοποθέτηση και βελτιστοποίηση όλων των βασικών κυττάρων πάνω στον πυρήνα αυτά πρέπει να συνδεθούν στις γραμμές τάσης – γείωσης.

```
sroute –noPadPins -noPadRings -nets {VDD VSS}
```

4.2.11 Καθολική Λεπτομερής Διασύνδεση – Global Detail Route

Το τελικό σχεδιαστικό βήμα είναι η καθολική διασύνδεση των βασικών κυττάρων του κυκλώματος. Αυτό γίνεται με το πρόγραμμα NanoRoute του SoC Encounter. Επίσης γίνεται και βελτιστοποίηση σε θέματα χρονισμού, θορύβου κ.α. Η NanoRoute έχει πολλές παραμέτρους που βοηθούν στην βελτιστοποίηση του κυκλώματος. Αναλύονται μερικές από τις βασικότερες

```
setNanoRouteMode -quiet drouteFixAntenna true
setNanoRouteMode -quiet routeInsertAntennaDiode false
setNanoRouteMode -quiet routeAntennaCellName default
```

Ζητείται από τον διασυνδετή να διορθωθούν αυτόματα όποια προβλήματα παρασιτικής κεραίας παρουσιαστούν. Τα προβλήματα παρασιτικής κεραίας μπορεί να εμφανιστούν λόγω του μεγάλου μήκους των αγωγών. Αυτά μπορούν να επιλυθούν είτε προσθέτοντας διόδους σε κάποιο σημείο του αγωγού είτε προσθέτοντας ενισχυτές είτε με σπάσιμο του αγωγού και μεταπήδηση του σε υψηλότερο επίπεδο μετάλλου. Το τελευταίο ζητείται από τον διασυνδετή για την επίλυση τέτοιου είδους σφαλμάτων.

```
setNanoRouteMode -quiet routeWithTimingDriven true
```

Ο διασυνδετής προσπαθεί να ελαχιστοποιήσει όποιες χρονικές παραβιάσεις υπάρχουν στην σχεδίαση αναλύοντας τους χρονισμούς όλων των μονοπατιών.

```
setNanoRouteMode -quiet routeWithTimingOpt false

setNanoRouteMode -quiet optimizeBi false

setNanoRouteMode -quiet optimizeGs false

setNanoRouteMode -quiet optimizeFixSetupTime true

setNanoRouteMode -quiet optimizeTargetSetupSlack 0.000000

setNanoRouteMode -quiet optimizeDontUseCellFile default
```

Οι παραπάνω ομάδα παραμέτρων ορίζει τον τρόπο που ο διασυνδετής θα εκτελέσει χρονική βελτιστοποίηση του κυκλώματος.

```
setNanoRouteMode -quiet envNumberProcessor 4
```

Ορίζεται ο αριθμός των νημάτων με τα οποία θα εκτελεστεί η διαδικασία.

Μετά τον ορισμό των παραμέτρων για την διασύνδεση δίνεται η εντολή για να ξεκινήσει η διαδικασία.

globalDetailRoute

Η διαδικασία χωρίζεται σε δυο στάδια την καθολική (global) και την λεπτομερή (detail) διασύνδεση. Η διασύνδεση είναι μια πολύ απαιτητική διεργασία σε υπολογιστικούς πόρους.

Κατά την καθολική διασύνδεση η σχεδίασης διαιρείται σε παραλληλόγραμμα (gcells) ώστε να εξακριβωθεί η συμφόρηση σε καθένα από αυτά ακολούθως αναθέτει στα gcells δίκτυα αγωγών σημάτων (nets). Σκοπός της καθολικής διασύνδεσης είναι να ελαχιστοποιηθεί η συμφόρηση του κυκλώματος καθώς και μειώσει τον αριθμό των gcells.

```
Total wire length = 874730 um.
#Total half perimeter of net bounding box = 874151 um.
#Total wire length on LAYER Metal1 = 11222 um.
#Total wire length on LAYER Metal2 = 333130 um.
#Total wire length on LAYER Metal3 = 457320 um.
#Total wire length on LAYER Metal4 = 34563 um.
#Total wire length on LAYER Metal5 = 37572 um.
#Total wire length on LAYER Metal6 = 923 um.
#Total number of vias = 55249
#Total number of vias on LAYER Metal1 to Metal2 = 27169
#Total number of vias on LAYER Metal2 to Metal3 = 25319
#Total number of vias on LAYER Metal3 to Metal4 = 1545
#Total number of vias on LAYER Metal4 to Metal5 = 1197
#Total number of vias on LAYER Metal5 to Metal6 = 19
#Cpu time = 00:00:09
#Elapsed time = 00:00:09
#Increased memory = 27.00 (Mb)
#Total memory = 139.00 (Mb)
#Peak memory = 139.00 (Mb)
```

Η λεπτομερής διασύνδεση είναι υπεύθυνη για την σύνδεση των ακροδεκτών (pins) των βασικών κυττάρων με τα δίκτυα αγωγών πού έχουν ορισθεί στο προηγούμενο βήμα της καθολικής διασύνδεσης. Έπειτα ο διασυνδετής είναι υπεύθυνος για την επίλυση όποιων παραβιάσεων των κανόνων σχεδίασης της τεχνολογικής βιβλιοθήκης.

```
#Start Detail Routing.
#start initial detail routing ...
#    number of violations = 297
#cpu time = 00:01:04, elapsed time = 00:01:13, memory = 138.00 (Mb)
#start 1st optimization iteration ...
#    number of violations = 117
#cpu time = 00:00:09, elapsed time = 00:00:10, memory = 132.00 (Mb)
#start 2nd optimization iteration ...
#    number of violations = 111
#cpu time = 00:00:02, elapsed time = 00:00:03, memory = 137.00 (Mb)
……
……
#start 6th optimization iteration ...
#    number of violations = 59
#cpu time = 00:00:07, elapsed time = 00:00:07, memory = 140.00 (Mb)
#start 7th optimization iteration ...
#    number of violations = 88
#cpu time = 00:00:05, elapsed time = 00:00:06, memory = 157.00 (Mb)
#start 8th optimization iteration ...
#    number of violations = 88
……
……
#start 15th optimization iteration ...
#    number of violations = 0
#cpu time = 00:00:08, elapsed time = 00:00:08, memory = 186.00 (Mb)
#Complete Detail Routing.
#Total wire length = 873143 um.
#Total half perimeter of net bounding box = 874151 um.
#Total wire length on LAYER Metal1 = 25302 um.
#Total wire length on LAYER Metal2 = 248321 um.
#Total wire length on LAYER Metal3 = 434737 um.
#Total wire length on LAYER Metal4 = 111125 um.
#Total wire length on LAYER Metal5 = 52081 um.
```

```
#Total wire length on LAYER Metal6 = 1577 um.
#Total number of vias = 72144
#Total number of vias on LAYER Metal1 to Metal2 = 29471
#Total number of vias on LAYER Metal2 to Metal3 = 33126
#Total number of vias on LAYER Metal3 to Metal4 = 7466
#Total number of vias on LAYER Metal4 to Metal5 = 2052
#Total number of vias on LAYER Metal5 to Metal6 = 29
#Total number of violations = 0
#Cpu time = 00:03:34
#Elapsed time = 00:04:13
#Increased memory = -7.00 (Mb)
#Total memory = 132.00 (Mb)
#Peak memory = 232.00 (Mb)
```

Μετά την ολοκλήρωση της διασύνδεσης εξάγουμε τις τελικές αναφορές για τον χρονισμό του κυκλώματος.

```
*** Reported constrained paths (0:00:00.4) ***
    (src|dest)          *      clock
    ----------+----------+----------
 0:          *|    -0.363|    -0.363
 1:      clock|    -0.363|    -0.363

--------------------  -------    -------
 Slack Range (ns)      Count        Sum
--------------------  -------    -------
( -0.363 ~  -0.200]       12         12
( -0.200 ~   0.000]      388        400
--------------------  -------    -------
Total negative slacks(TNS)=-43.355
Worst negative slacks(WNS)=-0.363
```

Η τελική χρονική ανάλυση δείχνει ότι ο διασυνδετής βελτίωσε την ολική καθυστέρηση κατά 61,88 % και την καθυστέρηση του χειρότερου μονοπατιού κατά 35 %. Παρόλο που οι καθυστερήσεις (slack) παραμένουν αρνητικές είναι σε πολύ χαμηλά επίπεδα πράγμα που κάνει κατάλληλο το κύκλωμα για κατασκευή.

4.2.12 Έλεγχος Κανόνων Σχεδίασης

Μετά την ολοκλήρωση της σχεδίασης, ελέγχεται για παραβιάσεις των κανόνων σχεδίασης, και για τυχόν προβλήματα συνδεσιμότητας.

Με την εντολή **`fillNotch`** γεμίζονται τα κενά μεταξύ αγωγών για αποφυγή σφαλμάτων same net κατά την επαλήθευση των κανόνων σχεδίασης.

Με την εντολή **`verifyConnectivity`** ελέγχεται η σχεδίαση για λάθη μεταξύ των συνδέσεων του κυκλώματος, π.χ. κάποιο pin ενός βασικού κυττάρου δεν είναι συνδεδεμένο. Επίσης είναι σημαντικός έλεγχος για τα ειδικά δίκτυα τάσης – γείωσης.

Με την εντολή **`verifyGeometry`** γίνεται έλεγχος των κανόνων σχεδίασης DRC. Επιβεβαιώνονται ότι οι αγωγοί έχουν σωστές αποστάσεις μεταξύ τους, ότι δεν υπάρχουν φαινόμενα παρασιτικής κεραίας.

4.2.13 Εξαγωγή Αρχείων – Στατιστικά Σχεδίασης - Αποθήκευση

Η ολοκλήρωση της αυτοματοποιημένης φυσικής σχεδίασης συνεχίζεται με το στάδιο της ολοκλήρωσης και εξαγωγής του ολοκληρωμένου κυκλώματος προς κατασκευή (Tape Out). Σημαντικά αρχεία προς εξαγωγή για το επόμενο στάδιο της ψηφιακής σχεδίασης είναι

1. **Αρχείο Παρασιτικών .spef (Standard Parasitic Exchange Format):** Τα αρχεία αυτά περιέχουν πληροφορίες για τα παρασιτικά δεδομένα των αγωγών διασύνδεσης (αντίσταση, χωρητικότητα, επαγωγή). Χρησιμοποιούνται για τον υπολογισμό καθυστέρησης ενός κυκλώματος.
2. **Αρχείο Ροής Σχεδίασης GDS II (Graphic Design System):** Τα αρχεία ροής είναι βάσεις δεδομένων με τα οποία μεταφέρονται μεταξύ των εργαλείων ψηφιακής σχεδίασης πληροφορίες για την φυσική σχεδίαση (layout) όπως γεωμετρικά σχήματα ταμπέλες κειμένου κ.α.
3. **Αρχείο DEF (Design Exchange Format):** Τα αρχεία DEF περιέχουν την λεπτομερή φυσική σχεδίαση και είναι η αναπαράσταση της ροής σχεδίασης στο αυτοματοποιημένο εργαλείο.
4. **Αρχείο LEF (Library Exchange Format):** Τα αρχεία LEF χρησιμοποιούνται για την δημιουργία βιβλιοθήκης βάση το κύκλωμα που σχεδιάστηκε. Χρησιμοποιείται σε συνδυασμό με το DEF αρχείο.

Ο Soc Encounter παρέχει στατιστικά στοιχεία για τον αριθμό και το είδος των πυλών που έχουν χρησιμοποιηθεί, το εμβαδό του κυκλώματος, το συνολικό μήκος των αγωγών

```
*** Statistics for net list top_module ***
Number of cells      = 18545
Number of nets       = 11827
Number of pins       = 29412
Number of i/os       = 802

*** 20 Primitives used:
Primitive INVX1 (2225 insts)
Primitive INVX2 (382 insts)
Primitive INVX4 (994 insts)
Primitive INVX8 (41 insts)
Primitive NAND3X1 (1 insts)
Primitive BUFX1 (31 insts)
Primitive NOR2X1 (2 insts)
Primitive NOR3X1 (2 insts)
Primitive OR2X1 (818 insts)
Primitive BUFX3 (112 insts)
Primitive FILL2 (1531 insts)
Primitive FILL4 (7989 insts)
Primitive AOI21X1 (17 insts)
Primitive OAI33X1 (380 insts)
Primitive DFFSRX1 (400 insts)
Primitive CLKBUFX1 (472 insts)
Primitive CLKBUFX3 (227 insts)
Primitive CLKBUFX2 (121 insts)
Primitive ADDHX1 (1200 insts)
Primitive ADDFX1 (1600 insts)
************
Summary report for Module: Top Cell
=============================================
 No. of Cells: 18545
 No. of IOs: 802
 Total Area: 1.253134e+06 um^2
================= Wire Info ==================
Total net length: 7.0853e+05 um 1.2437e+05 um
```

4.2.14 Αναφορές Κατανάλωσης Ισχύος

Ο Encounter δίνει την δυνατότητα εξαγωγής αναφορών για την κατανάλωση ισχύος για το δίκτυο παροχής ρεύματος.

```
updatePower -noRailAnalysis -postCTS -toggleFile top_module.tg -report
power.rpt -reportInstancePower instance.power VDD
#########################################################################
      Total-Power  Internal-Power Switching-Power   Leakage-Power
#########################################################################

# Sum: 2.507422e-01 1.186298e-01   1.321111e-01      1.373628e-06
# No. of instances: 9025
#########################################################################
```

4.1 Εισαγωγή Επιθεμάτων Εισόδου – Εξόδου (I/O Pads)

Τα επιθέματα εισόδου εξόδου είναι αυτά που συνδέονται με ακίδες στην βάση του ολοκληρωμένου κυκλώματος. Δηλαδή, το κάθε input / output γίνεται wire και οδηγείται είτε από επίθεμα εισόδου, είτε οδηγεί ένα επίθεμα εξόδου. Για τα επιθέματα εξόδου πρέπει να επιλέγεται το επίθεμα και με την σωστή οδηγητική ικανότητα κατάλληλη για εκτός του ολοκληρωμένου. Τα επιθέματα ορίζονται στην τεχνολογική βιβλιοθήκη και προστίθενται στον τεχνολογικά εξαρτημένο κώδικα με το χέρι. Πέρα των επιθεμάτων εισόδου / εξόδου υπάρχουν ειδικά επιθέματα για την τάση και γείωση, γωνιακά επιθέματα και επιθέματα γεμίσματος.

Στο παράδειγμα της σχεδίασης θα χρειαστούν

- 400 επιθέματα εισόδου για το διάνυσμα σημάτων Panel[] που εισάγει την αρχική κατάσταση του παιχνιδιού
- 400 επιθέματα εξόδου για το διάνυσμα σημάτων C[] που είναι η έξοδος του κυκλώματος
- 1 επίθεμα για την είσοδο του ρολογιού
- 1 επίθεμα για το σήμα SET/RESET
- 4 γωνιακά επιθέματα
- Επιθέματα για τα δίκτυα τάσης - γείωσης

4.1.1 Εισαγωγή επιθεμάτων στον κώδικα

Στον τεχνολογικά εξαρτημένο κώδικα δημιουργείται μια νέα μονάδα

```
module iopads(C,C_out, Panel, P_in,clock, clock_in, gsres, gsres_in);
```

Σε αυτή τη μονάδα οι έξοδοι του κυκλώματος γίνονται είσοδοι σε αυτή και οι είσοδοι έξοδοι προς το κύκλωμα.

```
input[0:399] C;
output [0:399] C_out;
input [0:399] P_in;
output [0:399] Panel;
input clock_in;
output clock;
input gsres_in;
output gsres;
```

Έπειτα ορίζονται τα επιθέματα σύμφωνα με την τεχνολογική βιβλιοθήκη GSCLib _IO στην οποία περιγράφονται τα επιθέματα εισόδου εξόδου

```
//clock gsres
PDUDGZ pad_gsres(.PAD(gsres_in), .Y(gsres));
PDUDGZ pad_clk(.PAD(clock_in), .Y(clock));
//pads for Panel inputs
PDUDGZ pad_P0(.PAD(P_in[0]), .Y(Panel[0]));
PDUDGZ pad_P1(.PAD(P_in[1]), .Y(Panel[1]));
PDUDGZ pad_P2(.PAD(P_in[2]), .Y(Panel[2]));
//pads for C outputs
PDO12CDG pad_C396(.A(C[396]), .PAD(C_out[396]));
PDO12CDG pad_C397(.A(C[397]), .PAD(C_out[397]));
PDO12CDG pad_C398(.A(C[398]), .PAD(C_out[398]));
PDO12CDG pad_C399(.A(C[399]), .PAD(C_out[399]));
```

Επόμενη κίνηση είναι να οριστούν τα ειδικά επιθέματα για τις γωνίες την τάση και την γείωση.

```
//corners fillers power ground
       PCORNER pad_corner4();
       PCORNER pad_corner3();
       PCORNER pad_corner2();
       PCORNER pad_corner1();
```

```
PVSSDGZ pad_vss4();
PVSSDGZ pad_vss3();
PVSSDGZ pad_vss2();
PVSSDGZ pad_vss1();
PVDDDGZ pad_vdd4();
PVDDDGZ pad_vdd3();
PVDDDGZ pad_vdd2();
PVDDDGZ pad_vdd1();
```

Δημιουργείται μία νέα μονάδα ως υψηλότερη ιεραρχικά στην οποία η προηγουμένως υψηλότερη ιεραρχικά μετατρέπεται σε υπομονάδα της.

```
module top_module(P_in,C_out, clock_in, gsres_in);

input [0:399] P_in;
output [0:399] C_out;
input clock_in;
input gsres_in;
wire [0:399] Panel;
wire [0:399] C;

iopads my_pads( .C(C), .C_out(C_out), .P_in(P_in), .Panel(Panel),
.clock(clock), .clock_in(clock_in), .gsres(gsres), .gsres_in(gsres_in));

top_module_core top_mod(.Panel(Panel), .C(C), .clock(clock), .gsres(gsres));

endmodule
```

4.1.2 Δημιουργία αρχείου επιθεμάτων (IO file)

Τα αρχείο επιθεμάτων περιέχει την σειρά και την θέση τους πάνω στην ψηφίδα. Η εισαγωγή στοιχείων ξεκινά από κάτω αριστερά με γωνιακό (SW) στοιχείο και συνεχίζει η τοποθέτηση σύμφωνα με τη φορά του ρολογιού

SW → W → NW → N → NE →E → SE.

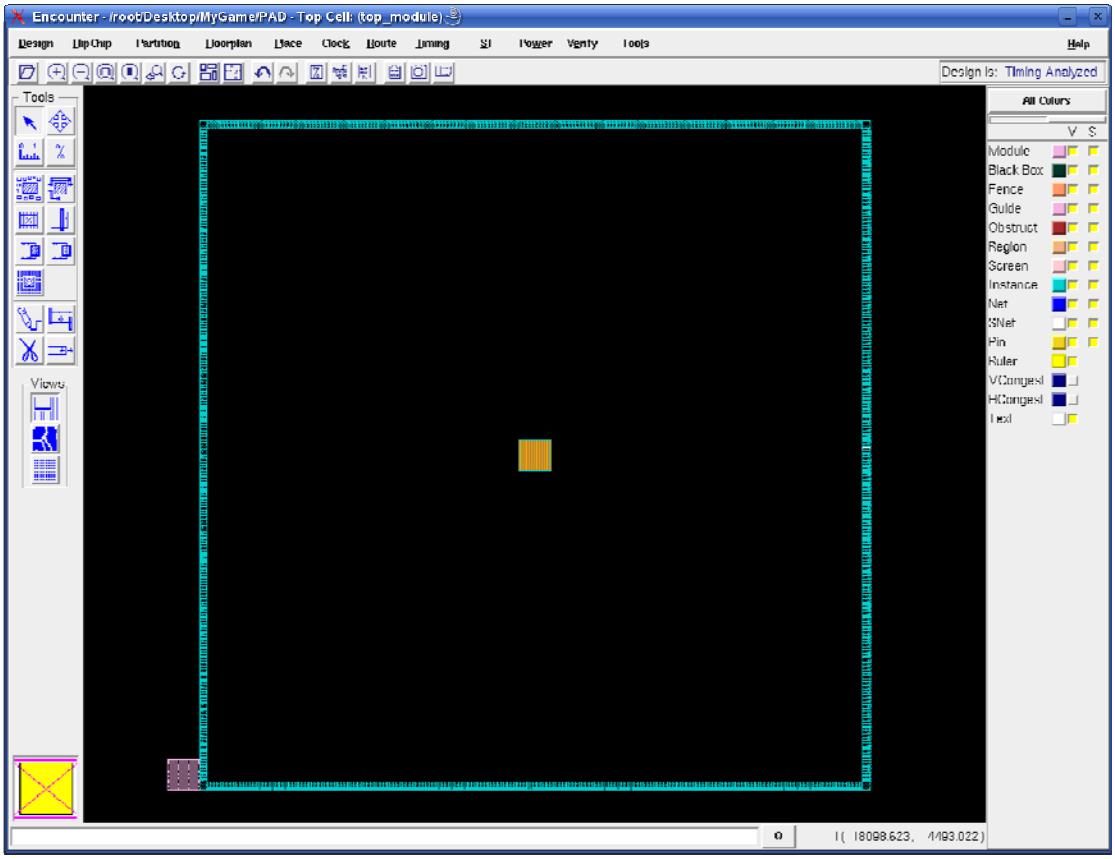
Σαν γενικότερες τακτικές σωστής σχεδίασης το επίθεμα του ρολογιού πρέπει να είναι δίπλα από ένα επίθεμα τάσης και τα επιθέματα τάσης γείωσης πρέπει να είναι στη μέση κάθε πλευράς.

```
PAD: my_pads/pad_corner1 SW
```

```
PAD: my_pads/pad_P0     W
PAD: my_pads/pad_P1     W
PAD: my_pads/pad_P2     W
PAD: my_pads/pad_P3     W
PAD: my_pads/pad_P4     W
PAD: my_pads/pad_P5     W3
PAD: my_pads/pad_P300   N
PAD: my_pads/pad_P301   N
PAD: my_pads/pad_clk     N
PAD: my_pads/pad_vdd1    N
PAD: my_pads/pad_vss1    N
PAD: my_pads/pad_gsres   N
PAD: my_pads/pad_P302   N
```

4.1.3 Ροή σχεδίασης

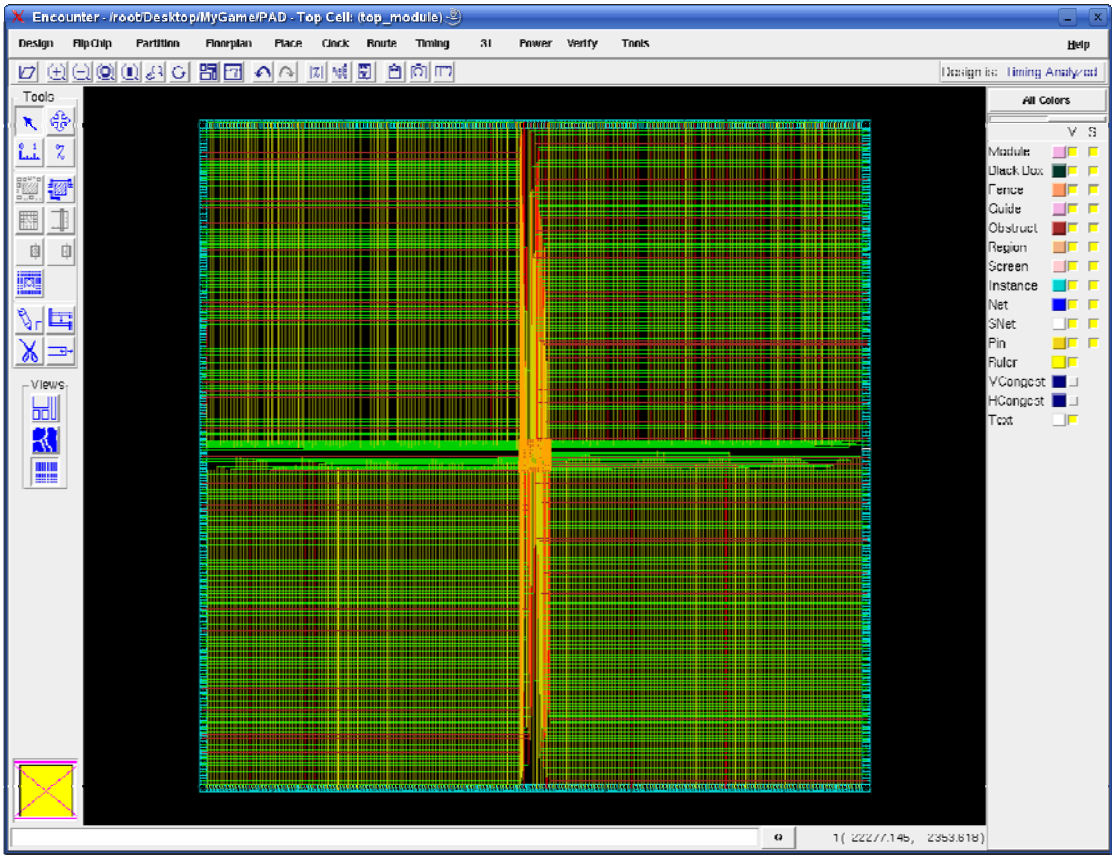
Η ροή σχεδίασης είναι όμοια με αυτήν που αναλύθηκε στην προηγούμενη παράγραφο. Εισαγωγή σχεδίασης στην μνήμη του εργαλείου



Σχέδιο κυκλώματος μετά την εισαγωγή

Παρατηρείται ότι λόγω των πολλών εισόδων και εξόδων του κυκλώματος δημιουργείται ένας πολύ μεγάλος δακτύλιος (Pad Ring) γύρω από τον πυρήνα του κυκλώματος (Pad Limited Die). Από την αρχή αυτό προϊδεάζει για μια πιο αργή σχεδίαση σε σχέση με την προηγούμενη. Η χρονική ανάλυση καλύτερης περίπτωσης μας δίνει καθυστέρηση για το χειρότερο μονοπάτι -0.577 ns.

Μετά την εισαγωγή και την ανάλυση καλύτερης περίπτωσης συνεχίζεται η ροή με τις διαδικασίες της τοποθέτησης, της σύνθεσης δέντρου ρολογιού, και της δοκιμαστικής διασύνδεσης. Μετά την χρονική ανάλυση αυτών των σταδίων η χρονική καθυστέρηση του χειρότερου μονοπατιού εκτιμάται στα - 61.685 ns. Μια αρκετά σημαντική καθυστέρηση που κυρίως οφείλεται στους μεγάλου μήκους αγωγούς από τον πυρήνα έως τον δακτύλιο επιθεμάτων. Στην παρακάτω εικόνα φαίνεται ο σχεδιασμός από το εργαλείο.

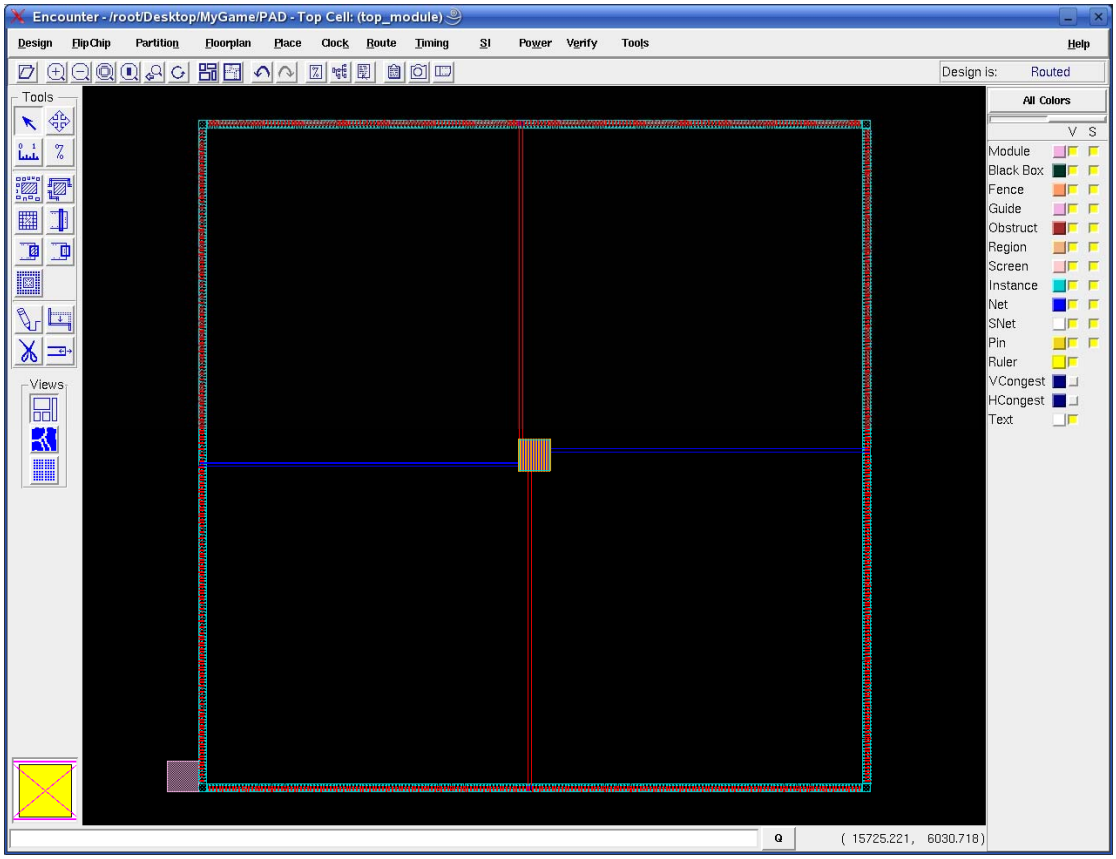


Η σχεδίαση μετά την δοκιμαστική διασύνδεση

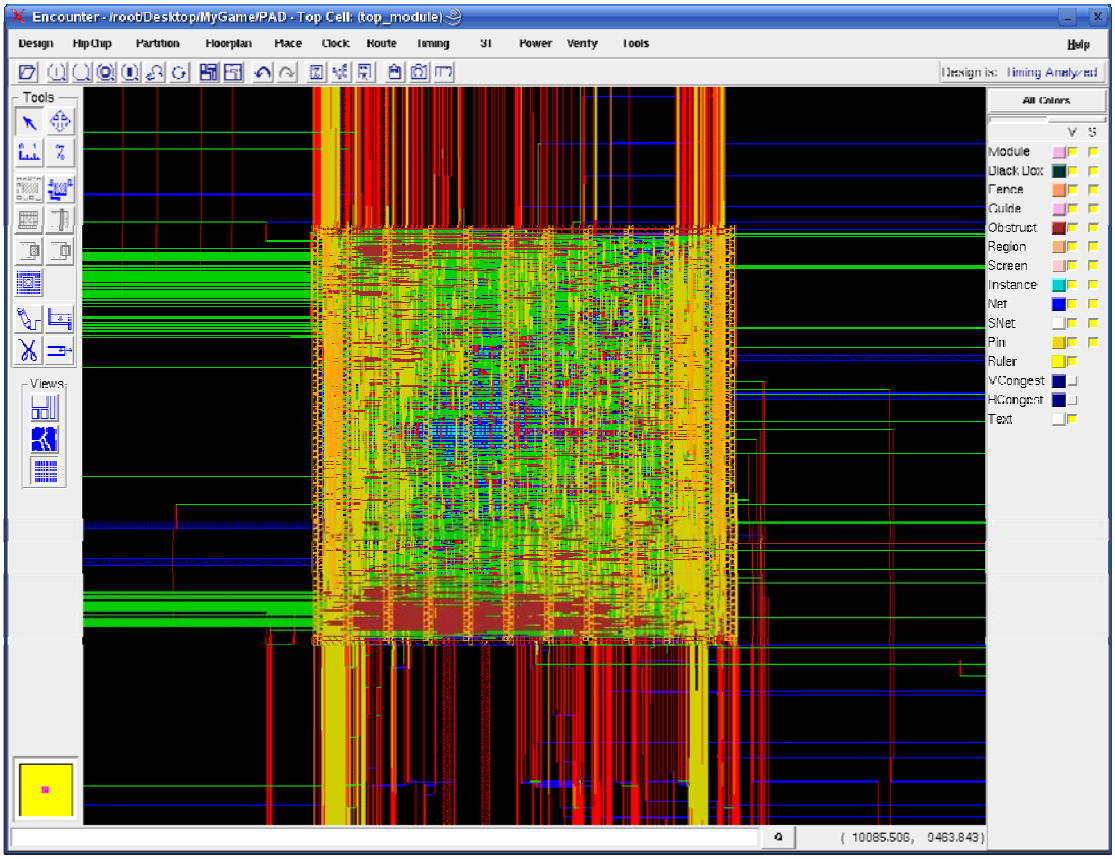
Αυτή η σχεδίαση με μεγάλους αγωγούς προς τα επιθέματα υποστηρίζεται από τα περισσότερα εργοστάσια κατασκευής ολοκληρωμένων κυκλωμάτων. Συνήθως η ψηφίδα καλύπτεται από αγώγιμο υλικό ώστε να μπορούν να τυπωθούν πάνω της αγωγοί και καλωδιώσεις.

Ακλουθούν τα βήματα, επιτόπιας βελτιστοποίησης, προσθήκη γεμισμάτων, διασύνδεσης παροχών ρεύματος και τελικής διασύνδεσης. Μετά κα την καθολική διασύνδεση παράγεται

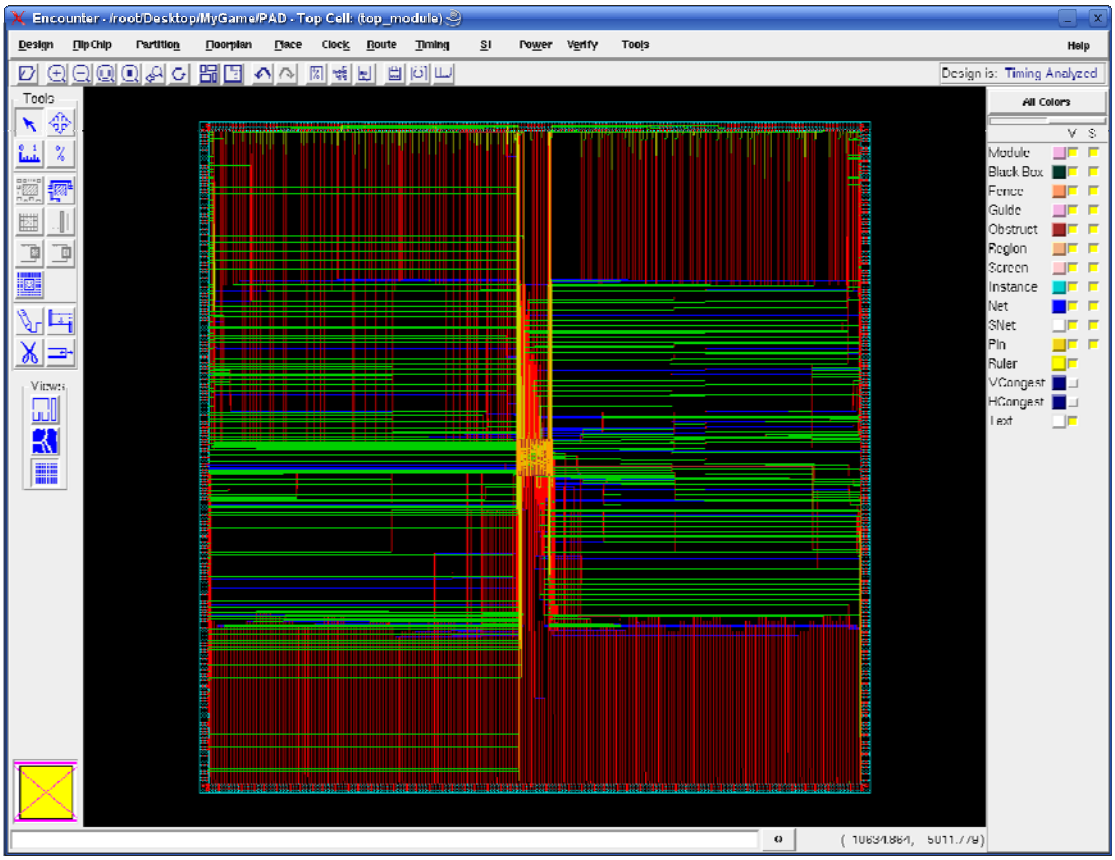
και η αναφορά χρονισμού δίνοντας χειρότερη καθυστέρηση μονοπατιού -38.766 ns βελτιωμένο κατά 37,15% σε σχέση με την δοκιμαστική διασύνδεση.



Διασύνδεση Παροχών ρεύματος



Ο πυρήνας μετά την καθολική και λεπτομερή διασύνδεση

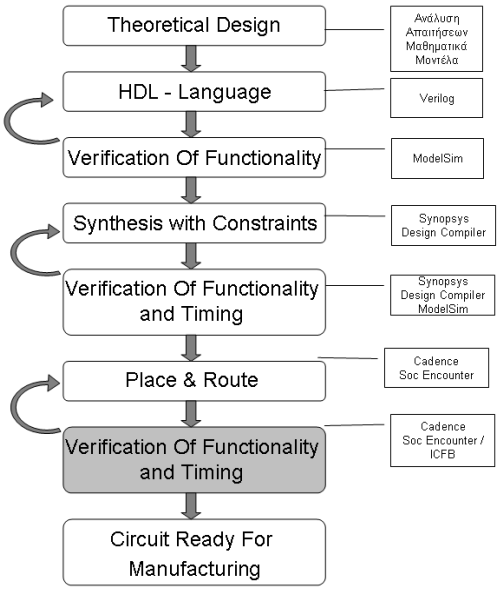


Πλήρες κύκλωμα προς εξαγωγή.

5 *Οριστικοποίηση Σχεδίασης – Εξαγωγή για κατασκευή*

5.1 Εισαγωγή

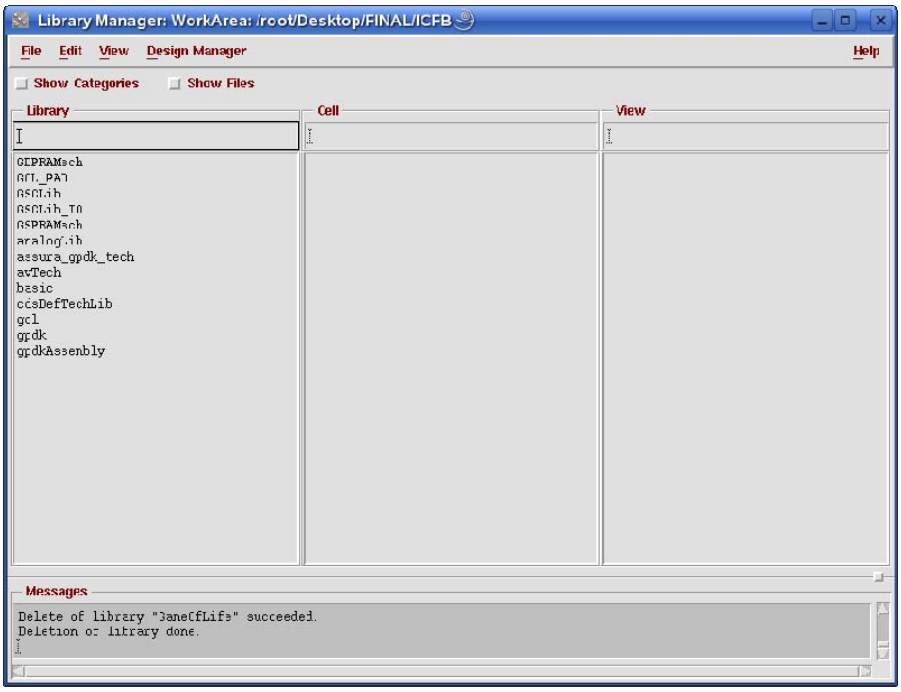
Το εργαλείο τοποθέτησης και διασύνδεσης δεν εξάγει λεπτομερής πληροφορίες για τις παρασιτικές καθυστερήσεις του κυκλώματος. Τα βασικά κύτταρα που εισάγονται στη σχεδίαση δεν περιέχουν όλα τα λειτουργικά τους επίπεδα διότι το εργαλείο έχεις ως σκοπό την αυτόματη διασύνδεση των εκατομμυρίων στοιχείων σε ικανοποιητικό χρόνο. Τα εργαλεία τελειοποίηση ψηφιακών σχεδιάσεων δίνουν την δυνατότητα εισάγοντας το αρχείο GDS II ή DEF που εξήχθη να το μετατρέψουν σε φυσική σχεδίαση με λεπτομερής σχεδίαση των βασικών κυττάρων για τον ακριβή υπολογισμό των παρασιτικών καθυστερήσεων RC. Επίσης στο εργαλείο μπορούν να διεξαχθούν έλεγχοι DRC και LVS για την τελική εξαγωγή του κυκλώματος. Στο στάδιο αυτό χρησιμοποιήθηκε η σχεδιαστική σουίτα της Cadence το οποίο είναι ευρέως διαδεδομένο σύνολο εργαλείων.



5.2 Δημιουργία Βιβλιοθήκης – Εισαγωγή

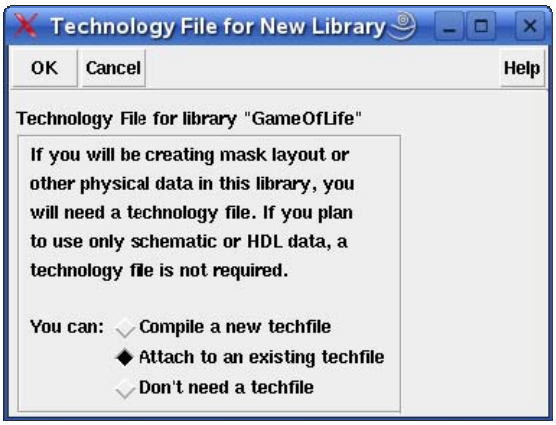
5.2.1 Δημιουργία βιβλιοθήκης Σχεδίασης

Στην κονσόλα δίνοντας την εντολή **icfb &** ξεκινά η σουίτα σχεδίασης της Cadence. Στο αρχικό παράθυρο ανοίγεται το παράθυρο του διαχειριστή βιβλιοθηκών (Tools → Library Manager)



Ο Library Manager του εργαλείου

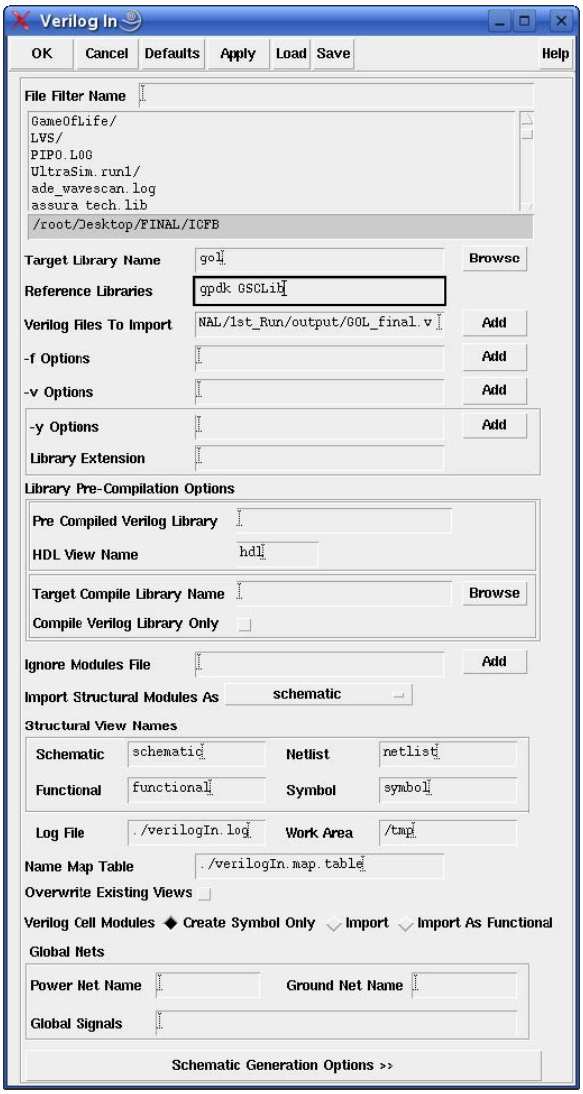
Δίνοντας File -> New -> Library ανοίγει παράθυρο διαλόγου όπου δίνεται το όνομα της βιβλιοθήκης που θα δημιουργηθεί. Σημαντικό βήμα είναι η αντιστοίχηση βιβλιοθήκης σε μια υπάρχουσα τεχνολογική βιβλιοθήκη.



Στο επόμενο βήμα επιλέγεται η βιβλιοθήκη της τεχνολογίας **gpdk.** Μετά την δημιουργία της βιβλιοθήκης εργασίας μπορούν να εισαχθούν τα αποτελέσματα από το προηγούμενο στάδιο.

5.2.2 Εισαγωγή Κώδικα Verilog

Στο τέλος της τοποθέτησης και διασύνδεσης παράγεται κώδικας Verilog της σχεδίασης μετά τις αλλαγές που γίνονται στο ολοκληρωμένο κύκλωμα. Αυτός ο κώδικας εισάγεται στο εργαλείο για την δημιουργία σχηματικού τεχνολογικά εξαρτημένου με την τεχνολογία. Σε αυτό το στάδιο τα βασικά κύτταρα έχουν όλες τις απαραίτητες πληροφορίες που χρειάζονται για την ανάλυση τους. Μέγεθος τρανζίστορ, μοντέλα SPICE των πυλών και μοντέλα κατάλληλα για τη διεξαγωγή των ελέγχων στο εργαλείο. Στο αρχικό παράθυρο δίνοντας File→ Import → Verilog… Ανοίγει το παράθυρο για την εισαγωγή κώδικα.

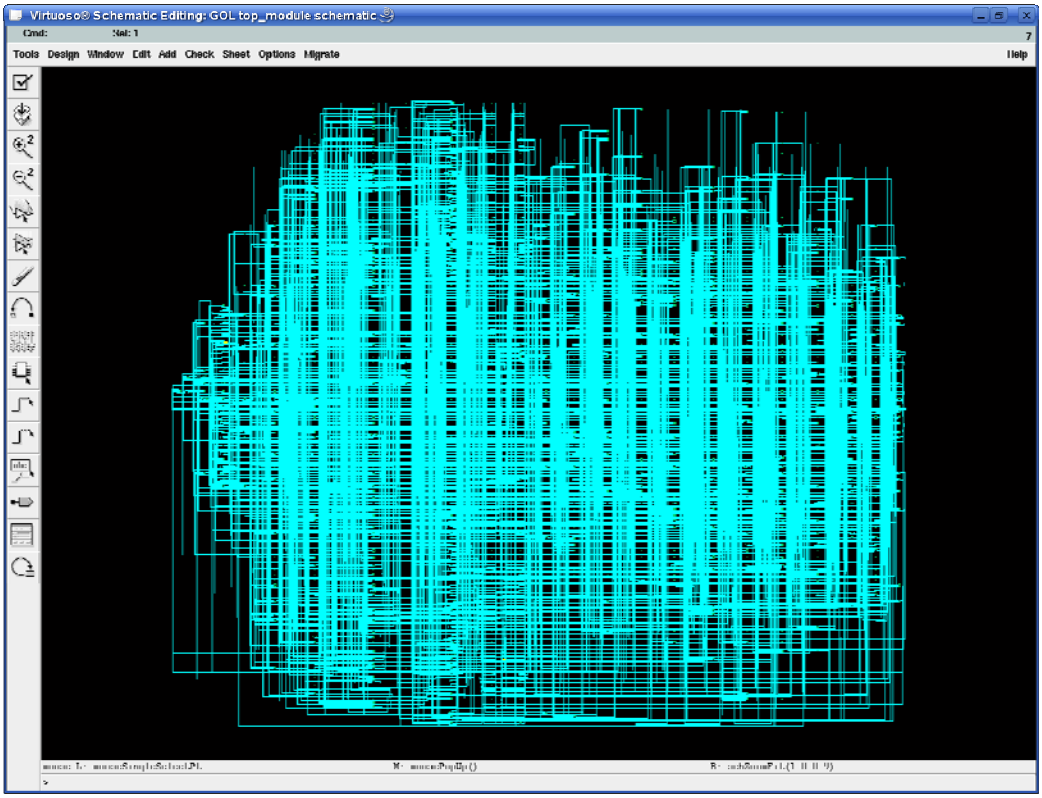


Εισάγουμε στα πεδία

- Target Library Name : Το όνομα της βιβλιοθήκης Εργασίας
- Reference Libraries: Τα ονόματα των τεχνολογικών βιβλιοθηκών που αντιστοιχούν στον κώδικα.

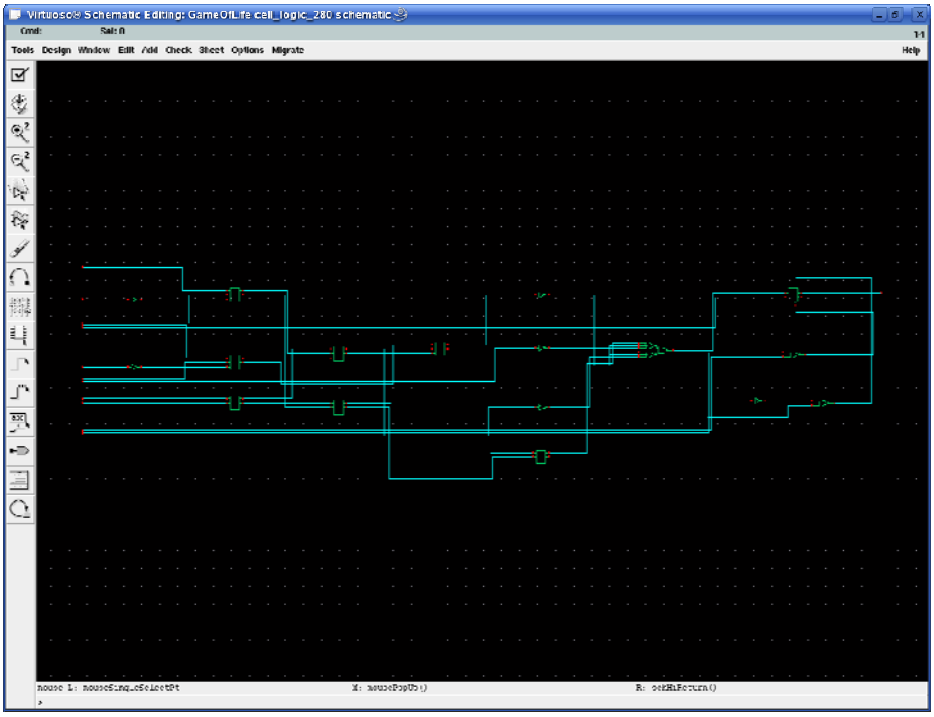
- GSCLib: στην οποία περιέχονται τα σχηματικά των βασικών κυττάρων, η φυσική σχεδίαση τους.
- gpdk: όπου περιέχονται οι πληροφορίες για τα μεγέθη των τρανζίστορ, τα παρασιτικά, κανόνες σχεδίασης,

Πηγαίνοντας στον Library Manager και επιλέγοντας το υψηλότερο ιεραρχικά κύτταρο, σε αυτό έχουν δημιουργηθεί όψεις (views) επιλέγεται αυτή του σχηματικού.



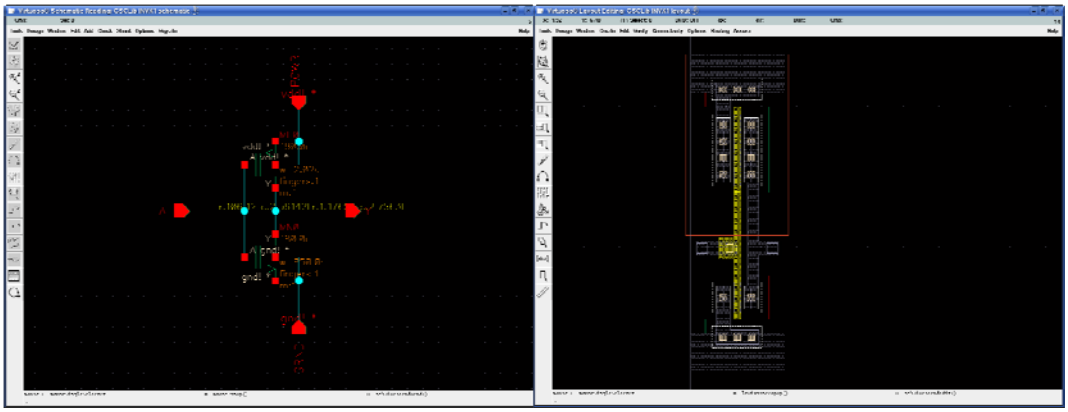
Το σχηματικό της σχεδίασης μετά την εισαγωγή.

Κάνοντας ζουμ στο σχηματικό πάνω σε μια κυψέλη Design → Hierarchy → Descend μπορεί να εμφανιστεί το σχηματικό του δομικού στοιχείου.



Σχηματικό κυψέλης

Όμοια σε κάθε βασικό κύτταρο του κελιού γίνεται να εμφανιστεί το σχηματικό του σε επίπεδο τρανζίστορ ή η φυσική του σχεδίαση. Αυτές οι όψεις παρέχονται από της βιβλιοθήκες αναφοράς που δόθηκαν κατά την εισαγωγή.

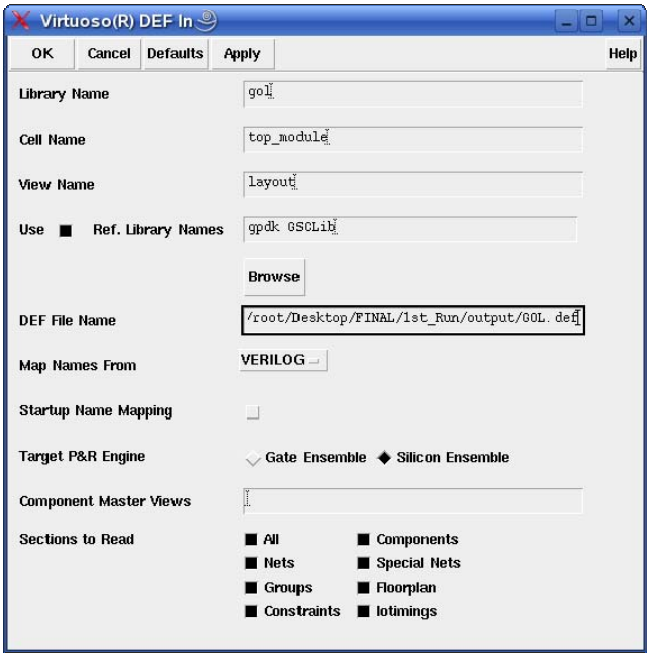


Σχηματικό και φυσική σχεδίαση ενός inverter.

5.2.3 Εισαγωγή Φυσικής Σχεδίασης

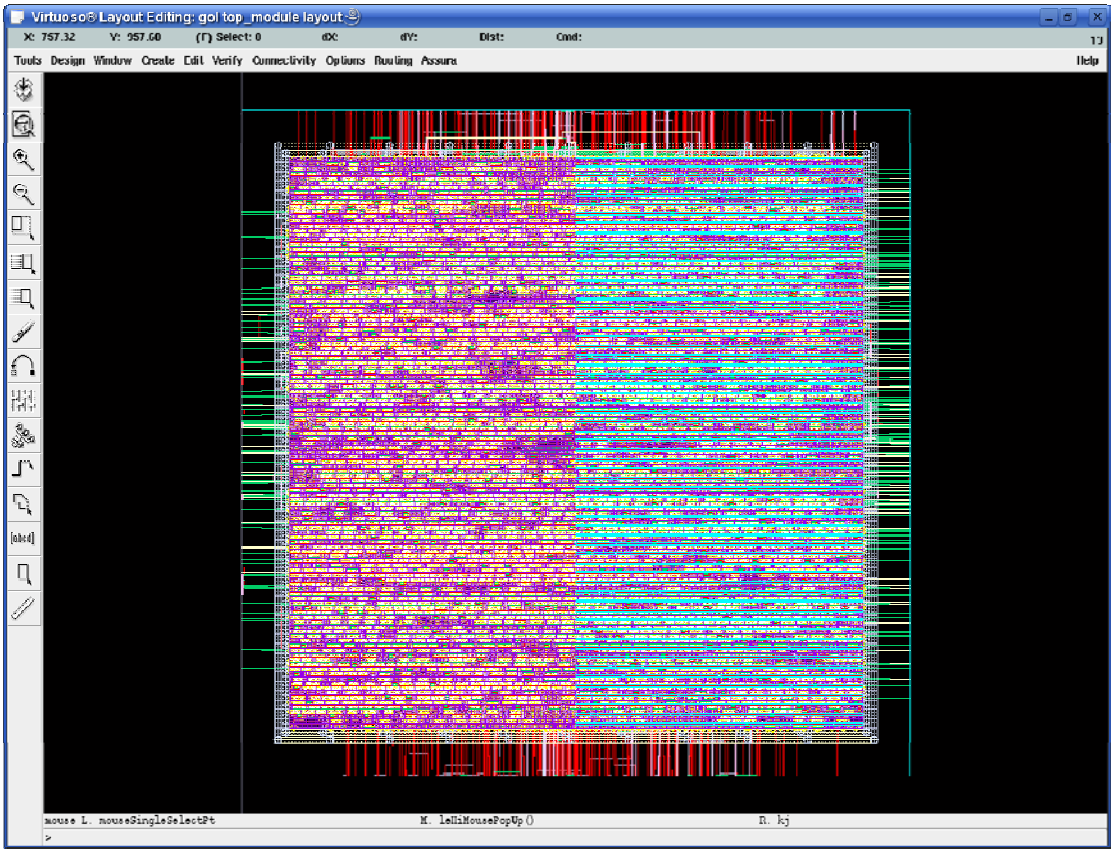
Τα αρχεία DEF περιέχουν τις τοποθεσίες των βασικών κυττάρων πάνω στον πυρήνα τις τοποθεσίες των αγωγών. Στο εργαλείο μπορούν να εισαχθούν αρχεία DEF για την λεπτομερή σχεδίαση του κυκλώματος. Στο αρχικό παράθυρο δίνοντας File→ Import → DEF… .

Εισαγωγή μπορεί να γίνει επίσης και με αρχεία GDS II τα οποία πάλι προέρχονται από την αυτοματοποιημένη διαδικασία φυσικής σχεδίασης.



Παράθυρο Διαλόγου εισαγωγής Αρχείων DEF

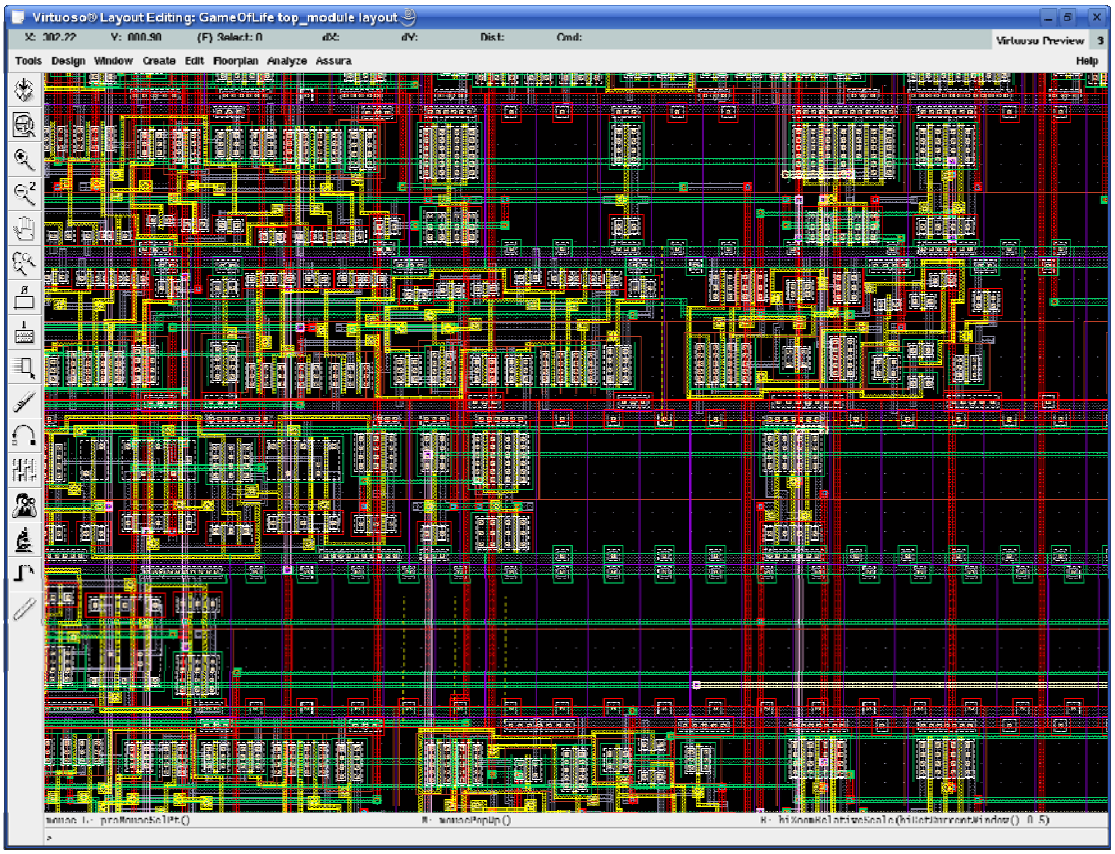
Στην βιβλιοθήκη εργασίας πλέον έχει δημιουργηθεί μια νέα όψη για την φυσική σχεδίαση



Εικόνα του layout.

Κατά την εισαγωγή του αρχείου τα βασικά κύτταρα τοποθετούνται σαν αφηρημένο σχέδιο (abstract view). Μπορεί η όψη τους να αλλάξει κάνοντας Floorplan→ Replace View και έπειτα επιλέγεται η επιθυμητή όψη. Μετά την αλλαγή αυτή αλλάζομε την λειτουργικότητα του Virtuoso από Preview σε Editor. Με την εισαγωγή λύνεται το πρόβλημα τοποθέτησης των βασικών κυττάρων σε πλήρη λειτουργικότητα και σχεδίαση.

Έπειτα συνεχίζεται η σχεδίαση κάνοντας τους ελέγχους DRC, LVS, ERC.



Πλήρης Φυσική Σχεδίαση

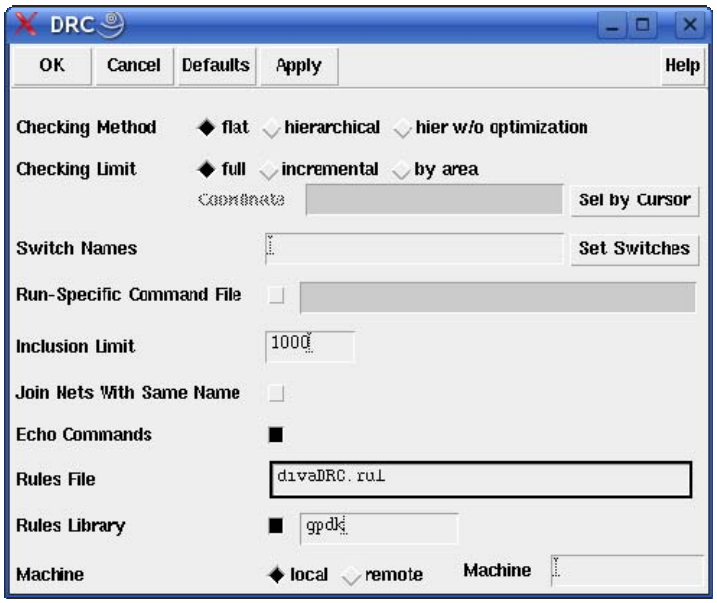
5.3 Έλεγχος Ολοκληρωμένου Κυκλώματος

Μετά την επιτυχή εισαγωγή της σχεδίασης το ολοκληρωμένο κύκλωμα είναι έτοιμο για την εκτέλεση των τελικών ελέγχων.

5.3.1 Έλεγχος Κανόνων Σχεδίασης – Design Rule Check

Σε αυτό τον έλεγχο το ολοκληρωμένο εξετάζεται για τυχόν παραβιάσεις στις αποστάσεις μεταξύ των αγωγών (spacing rules), έλεγχοι παραβίασης του ελάχιστου πλάτους αγωγών (minimum width rules), έλεγχοι για τις διασυνδέσεις (vias) μεταξύ των διαφορετικών επιπέδων μετάλλων, αποστάσεις μεταξύ των βασικών κυττάρων έλεγχοι για παρουσία

παρασιτικής κεραίας, γίνονται έλεγχοι για την σωστή σύνδεση των αγωγών. Υπάρχουν πάρα πολλοί ακόμη σχεδιαστικοί κανόνες που ορίζονται στις τεχνολογικές βιβλιοθήκες κάθε κατασκευαστή. Στην σουίτα της Cadence υπάρχουν τρία προγράμματα για την έλεγχο του ολοκληρωμένου κυκλώματος *Diva*, *Dracula* και *Assura*. Στο παράδειγμα αυτό χρησιμοποιήθηκε το πρόγραμμα *Diva*.



Παράθυρο διαλόγου Diva DRC.

Επειδή η φυσική σχεδίαση προέρχεται από αυτοματοποιημένο εργαλείο αναμένεται να μην παρουσιάσει σφάλματα στο τέλος του ελέγχου. Αν παρουσιαστούν σφάλματα θα πρέπει να επαναληφθεί η διαδικασία τοποθέτησης και διασύνδεσης.

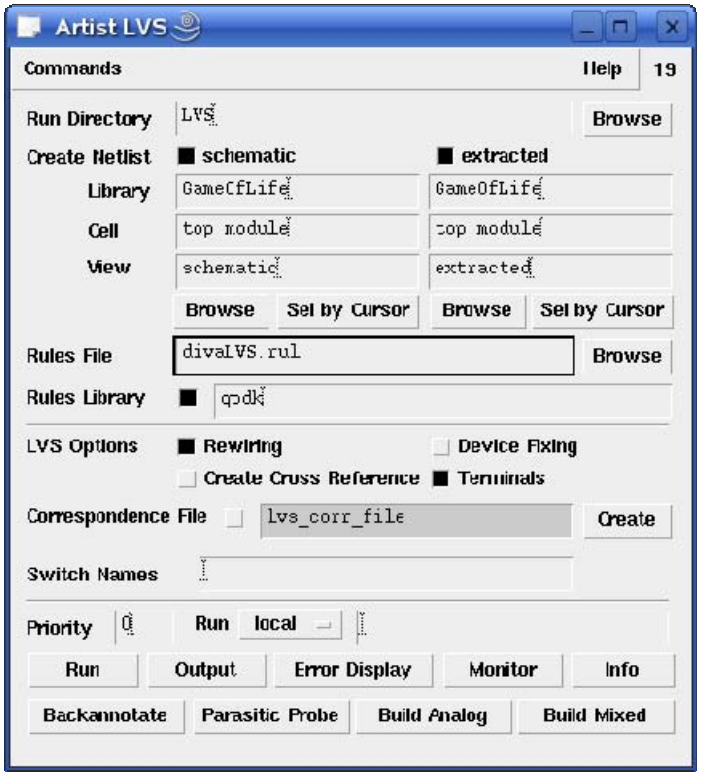
5.3.2 Φυσική Σχεδίαση Vs Σχηματικού – Layout Vs Schematic

Σε αυτό τον έλεγχο το πρόγραμμα ελέγχει αν η φυσική σχεδίαση του ολοκληρωμένου κυκλώματος ανταποκρίνεται στο σχηματικό του κυκλώματος. Κατά τον έλεγχο DRC ελέγχεται κατά πόσο είναι κατάλληλο για κατασκευή το κύκλωμα σε αυτόν τον έλεγχο ελέγχεται κατά πόσο η φυσική σχεδίαση ανταποκρίνεται σε αυτό που ζητείτε για κατασκευή.

Αρχικά μετά των έλεγχο των κανόνων σχεδίασης δημιουργείται μια νέα όψη του ολοκληρωμένου κυκλώματος Extracted view. Σε αυτή την όψη διαβάζονται όλα τα στοιχεία της τεχνολογίας, επίπεδα μετάλλων, διασυνδέσεις, βασικά κύτταρα και έπειτα διαβάζει την σχεδίαση για να αναγνωρίσει όλα αυτά τα στοιχεία. Επίσης αναγνωρίζει τους ακροδέκτες και τις τοποθεσίες τους πάνω στο κυκλώματος (εισόδους, εξόδους, τάση, γείωση). Αυτό είναι το στάδιο εξαγωγής (extraction).

Στη συνέχεια κατά την μείωση (reduction) δημιουργούνται και για τις δύο σχεδιάσεις γράφοι συνδεσιμότητας.

Τέλος γίνεται η σύγκριση των δύο τελευταίων γράφων και η εξαγωγή των αποτελεσμάτων.



Παράθυρο Διαλόγου ελέγχου LVS

5.3.3 Ηλεκτρικοί Κανόνες – Electrical Rule Check

Οι ηλεκτρικοί κανόνες είναι ιδιότητες που προσδιορίζουν το κύκλωμα από τη γεωμετρία της φυσικής σχεδίασης και τις συνδέσεις της χωρίς να ενδιαφέρονται για την λειτουργικότητα που προσδιορίζει το κύκλωμα.

5.4 Αποστολή Προς Κατασκευή

Μετά την ολοκλήρωση των ελέγχων και την πλήρη ανάλυση για τον πραγματικό χρονισμό του κυκλώματος, αυτό είναι έτοιμο για το τελικό στάδιο της ψηφιακής σχεδίασης την παραγωγή. Οι συνηθέστερες μορφές αρχείων σχεδίασης που δέχονται τα εργοστάσια κατασκευής (FAB's) είναι είτε αρχεία GDS II (Graphic Design System) CIF (Caltech Intermediate Format).

6 *Επίλογος*

6.1 Σύνοψη και συμπεράσματα

Στις σημερινές μέρες όπου τα προϊόντα έχουν μικρό χρόνο ζωής, ο χρόνος σχεδίασης και παραγωγής είναι πρωταρχικό μέλημα. Στον τομέα των ολοκληρωμένων κυκλωμάτων, τομέας που πλέον έχει κατακλύσει την αγορά και είναι κομμάτι της καθημερινότητας, τα εργαλεία αυτοματοποιημένης σχεδίασης δίνουν επαρκής λύσεις για την επίτευξη αυτών των στόχων. Οι ροές σχεδίασης συνεχώς εμπλουτίζονται με βήματα που εξυπηρετούν κάθε ανάγκη για επιπλέον έλεγχο ή εξυπηρέτηση ειδικών απαιτήσεων. Οι διαδικασίες που ακολουθήθηκαν σε αυτή την εργασία είναι εύκολα επαναχρησιμοποιήσιμες, αυτό άλλωστε είναι και ένα ακόμη πλεονέκτημα των εργαλείων.

Η θεωρητική σχεδίαση του κυκλώματος βασίστηκε σε προηγούμενες εργασίες θεωρητικά μοντέλα και υλοποιήσεις[1][3] . Παρατηρήθηκε ότι δεν είναι ικανά όλα τα προγράμματα σύνθεσης να ανταπεξέλθουν σε κώδικα υψηλού αφαιρετικού επιπέδου. Ο HDL Compiler του Design Compiler ανταποκρίθηκε στις απαιτήσεις. Το εργαλείο τοποθέτησης και διασύνδεσης ήταν το ποιο απαιτητικό σε θέματα μελέτης καθώς ο τεράστιος αριθμός επιλογών που παρέχει προσπαθούν να καλύψουν κάθε ανάγκη του σχεδιαστή και των απαιτήσεων που έχουν οριστεί, ανταπόκρισης στα επιθυμητά αποτελέσματα καθώς και στους χρόνους εκτέλεσης. Χαρακτηριστικά η σχεδίαση με επιθέματα εισόδου εξόδου απαιτούσε περίπου 3 ώρες για την περάτωση της. Η σουίτα σχεδίασης της Cadence είναι και αυτό ένα πολύ απαιτητικό εργαλείο απαιτεί βαθιές θεωρητικές γνώσεις στους τομείς των ηλεκτρονικών και της ψηφιακής σχεδίασης. Με την αυτοματοποιημένη διαδικασία που παρουσιάστηκε στην εργασία εξοικονομείται χρόνος για τον σχεδιαστή. Οι πιο απαιτητικές εργασίες που έγιναν στη σουίτα ήταν οι έλεγχοι DRC & LVS καθώς και οι προσομοιώσεις των όψεων τις σχεδίασης.

6.2 Μελλοντικές επεκτάσεις

Η εξέλιξη των εργαλείων αυτοματοποιημένης σχεδίασης CAD είναι ραγδαία. Συνεχώς οι προμηθευτές παρουσιάζουν καινούρια στοιχεία για την καλύτερη, αποτελεσματικότερη και γρηγορότερη υλοποίηση ολοκληρωμένων κυκλωμάτων. Βελτιστοποιήσεις μπορούν να γίνουν σε όλα τα βήματα της ροής σχεδίασης που παρουσιάστηκε, μιας και πλέον χρησιμοποιούνται όσον το δυνατόν μικρότερα και γρηγορότερα κυκλώματα. Η ροή σχεδίασης αυτής της εργασίας μπορεί να επαναληφθεί χρησιμοποιώντας τεχνολογία που παρέχεται από εργοστάσια κατασκευής για την παραγωγή του κυκλώματος. Το Game of Life λόγω της ιδιαιτερότητα της εξέλιξης του παιχνιδιού, η λειτουργία του κυκλώματος μπορεί να ξεκινήσει από την πάνω αριστερή γωνία και να καταλήξει σε μία άλλη μπορεί να δώσει σημαντικές πληροφορίες για αναλύσεις όπως ανάλυση πτώσης τάσης (IR Drop) είτε μελέτες για φαινόμενα ηλεκτρομετανάστευσης.

7 *Βιβλιογραφία*

[1] Apostolos Dollas, Kyprianos Papademetriou, Euripides Sotiriades, Dimitrios Theodoropoulos, Iosif Koidis, George Vernardos. *A Case Study on Rapid Prototyping of Hardware Systems: the Effect of CAD Tool Capabilities, Design Flows, and Design Styles*, 2004.

[2] http://www.vtvt.ece.vt.edu/vlsidesign/tutorials.php

[3] http://www.bitstorm.org/gameoflife/

[4] http://www.csd.uoc.gr/~hy523/2010-2011f/howtos/socencounterhowto_2010.html

[5] Encounter User Guide, Cadence Inc.

[6] Design Compiler User Guide. Synopsys Inc.

[7] Design Vision User Guide, Synopsys Inc.

[8] Introduction to Place and Route Designs in VLSI's, Patrick Lee

[9] ASICs… the website, http://iroi.seu.edu.cn/books/asics/ASICs.htm

[10] N.H. WESTE, K.ESHRAGHIAN, *ΣΧΕΔΙΑΣΗ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ CMOS VLSI*, Παπασωτηρίου 1η Ελληνική έκδοση, 1996

[11] http://pdk.cadence.com

[12] http://en.wikipedia.org

Παράρτημα

1. Κώδικας Verilog

a. Κυψέλη – Cell

```
module cell_logic   (   gsres, clock,
                        P0,
                        C1, C2, C3,
                        C4,     C5,
                        C6, C7, C8,
                        C0
                        );
//  | c1  c2  c3  |         | NW  N  NE  |
//  | c4 (c0) c5  |   ...   | W   C   E  |
//  | c6  c7  c8  |         | SW  S  SE  |
input gsres;                //General set-reset
input clock;                //System clock
input P0;
input C1, C2, C3, C4, C5, C6, C7, C8;
output C0;
wire [2:0] composition_C0;
assign composition_C0 = C1 + C2 + C3 + C4 + C5 + C6 + C7 + C8;
reg C0;
always @(posedge clock or negedge gsres)
    if (~gsres)      C0 <= P0;
    else
        begin
            if ((C0==1'b1 &&(composition_C0 ==2|| composition_C0 ==3 ))  ||
```

```
                    (C0==1'b0 && composition_C0 ==3)    )

C0 <= 1'b1;
            else
C0 <= 1'b0;
        end
endmodule //
```

b. Πάνελ

```
module top_module(  gsres, clock,
                    Panel,
                    C
                    //cellout
                    );

parameter N=20;
parameter N_2=N*N;
input gsres;                    //General set-reset
input clock;                    //System clock
input [0:N_2-1] Panel;
output [0:N_2-1] C;
/*
  C[i-11, i-10, i-9,
    i-1,        i+1,
    i+9, i+10, i+11]
*/
genvar i,j;









//----------------------------------------
//--Instantiate the 4 Cell-Corners of Panel  --
```

```
//----------------------------------------

cell_logic cl_0 (   .gsres(gsres), .clock(clock),
                    .P0(Panel[0]), //input signal initial value from Panel
                    //.P0(1'b0),
                    .C1(1'b0),      .C2(1'b0),      .C3(1'b0),
                    .C4(1'b0),                      .C5(C[1]),
                    .C6(1'b0),      .C7(C[N]),     .C8(C[N+1]),
                    .C0(C[0])                                 //output signal
synchronized with 1 clock
                    );

cell_logic cl_1 (   .gsres(gsres), .clock(clock),
                    .P0(Panel[N-1]), //input signal initial value from Panel
                    //.P0(1'b0),
                    .C1(1'b0),      .C2(1'b0),      .C3(1'b0),
                    .C4(C[N-2]),                      .C5(1'b0),
                    .C6(C[(N-1)+N-1]),    .C7(C[N-1+N]),      .C8(1'b0),
                    .C0(C[N-1])                               //output signal
synchronized with 1 clock
                    );

cell_logic cl_2 (   .gsres(gsres), .clock(clock),
                    .P0(Panel[N_2-N]),                         //input signal
initial value from Panel
                    //.P0(1'b0),
                    .C1(1'b0),          .C2(C[N_2-(2*N)]),         .C3(C[N_2-
(2*N)+1]),
                    .C4(1'b0),                      .C5(C[N_2-N +1]),
                    .C6(1'b0),      .C7(1'b0),      .C8(1'b0),
                    .C0(C[N_2-N])                             //output signal
synchronized with 1 clock
                    );




cell_logic cl_3 (   .gsres(gsres), .clock(clock),
                    .P0(Panel[N_2-1]),//input signal initialvalue from Panel
```

```
                    //.P0(1'b0),
                    .C1(C[N_2-N-2]),     .C2(C[N_2-N-1]),     .C3(1'b0),
                    .C4(C[N_2-2]),                    .C5(1'b0),
                    .C6(1'b0),      .C7(1'b0),      .C8(1'b0),
                    .C0(C[N_2-1])                             //output signal
synchronized with 1 clock
                    );
//1 to N-1 Horizontal first line
generate
for (i=1; i < N-1; i = i + 1)
begin: gen_cell_logic_5
cell_logic cl_5 (   .gsres(gsres), .clock(clock),
                    .P0(Panel[i]),  //input signal initial value from Panel
                    //.P0(1'b0),
                    .C1(1'b0),          .C2(1'b0),      .C3(1'b0),
                    .C4(C[i-1]),                        .C5(C[i+1]),
                    .C6(C[i+N-1]),    .C7(C[i+N]),  .C8(C[i+N+1]),
                    .C0(C[i])                                 //output signal
synchronized with 1 clock
                    );
end
endgenerate
//last line
generate
for (i=N_2 -N +1; i <N_2-1; i = i + 1)
begin: gen_cell_logic_6
cell_logic cl_6(   .gsres(gsres), .clock(clock),
                    .P0(Panel[i]),  //input signal initial value from Panel
                    //.P0(1'b0),
                    .C1(C[i-N-1]),          .C2(C[i-N]),      .C3(C[i-N+1]),
                    .C4(C[i-1]),                        .C5(C[i+1]),
                    .C6(1'b0),          .C7(C[1'b0]),   .C8(C[1'b0]),
                    .C0(C[i])                                 //output signal
synchronized with 1 clock
                    );
end
endgenerate
//first column
generate
```

```
for (i=N; i < N_2-(2*N)+1; i = i + N)
 begin: gen_cell_logic_7
cell_logic cl_7(   .gsres(gsres), .clock(clock),
                   .P0(Panel[i]),   //input signal initial value from Panel
                   //.P0(1'b0),
                   .C1(1'b0),          .C2(C[i-N]),  .C3(C[i-N+1]),
                   .C4(1'b0),                          .C5(C[i+1]),
                   .C6(1'b0),          .C7(C[i+N]),  .C8(C[i+N+1]),
                   .C0(C[i])                                 //output signal
synchronized with 1 clock
                   );
end
endgenerate


//last column
generate
for (i=2*N-1; i <N_2-N ; i = i + N)
begin: gen_cell_logic_8
cell_logic cl_8(   .gsres(gsres), .clock(clock),
                   .P0(Panel[i]),   //input signal initial value from Panel
                   //.P0(1'b0),
                   .C1(C[i- N-1]),    .C2(C[i-N]),  .C3(1'b0),
                   .C4(C[i-1]),                        .C5(1'b0),
                   .C6(C[i+N-1]),    .C7(C[i+N]),  .C8(1'b0),
                   .C0(C[i])                                 //output signal
synchronized with 1 clock
                   );
end
endgenerate

generate
for (i=N+1; i < N_2-N-1; i = i + N)
begin: gen_cell_logic_4
       for(j=i;j<N+i-2;j=j+1)
       begin:inner

cell_logic cl_4  (   .gsres(gsres), .clock(clock),
```

```
                    .P0(Panel[j]),    /input signal initial value from Panel
                    //.P0(1'b0),
                    .C1(C[j-N-1]),          .C2(C[j-N]),      .C3(C[j-N+1]),
                    .C4(C[j-1]),                        .C5(C[j+1]),
                    .C6(C[j+ N- 1]),    .C7(C[j+N]),  .C8(C[j+N+1]),
                    .C0(C[j])                                 //output signal
synchronized with 1 clock
                    );
                                end
end
endgenerate
endmodule
```

c. Testbench

```
module test_vector20;
parameter num_bytes = 19;
reg [19:0] mem[0:num_bytes];            //RAM with 8-bits words
//top_module Output signals
//output CELLS;
wire [0:399] CELLS;
//top_module Input signals
//input GSRES;
//input CLOCK;
reg GSRES;                      //global set-reset
reg CLOCK;                      //System clock
//----- System Clock
//--------------------
initial #0 CLOCK <= 1'b0;
always #5 CLOCK <= ~CLOCK;
//============================
//======== SIMULATION ========
//============================
//----- Configuration
//--------------------
initial begin
        GSRES <= 1'b1;
#20     GSRES <= 1'b0;
```

```
#50     GSRES <= 1'b1;
end
//--- Read MEMORY
//-----------------
initial begin
    $readmemb("memory20.mem",mem,0,num_bytes);   //reading file FIFO.txt and
put data into Verilog memory
end
wire [0:399] PANEL;                       //data in from MEMORY
assign PANEL[0:19]   =   mem[0];
assign PANEL[20:39] =   mem[1];
assign PANEL[40:59] =   mem[2];
assign PANEL[60:79] =   mem[3];
assign PANEL[80:99] =   mem[4];
assign PANEL[100:119] =   mem[5];
assign PANEL[120:139] =   mem[6];
assign PANEL[140:159] =   mem[7];
assign PANEL[160:179] =   mem[8];
assign PANEL[180:199] =   mem[9];
assign PANEL[200:219] =   mem[10];
assign PANEL[220:239] =   mem[11];
assign PANEL[240:259] =   mem[12];
assign PANEL[260:279] =   mem[13];
assign PANEL[280:299] =   mem[14];
assign PANEL[300:319] =   mem[15];
assign PANEL[320:339] =   mem[16];
assign PANEL[340:359] =   mem[17];
assign PANEL[360:379] =   mem[18];
assign PANEL[380:399] =   mem[19];
wire [0:19] CELL_0, CELL_1, CELL_2, CELL_3, CELL_4, CELL_5, CELL_6, CELL_7,
CELL_8,                                                              CELL_9,
CELL_10,CELL_11,CELL_12,CELL_13,CELL_14,CELL_15,CELL_16,CELL_17,CELL_18,CELL
_19;
assign CELL_0   =   CELLS[0:19];
assign CELL_1   =   CELLS[20:39];
assign CELL_2   =   CELLS[40:59];
assign CELL_3   =   CELLS[60:79];
assign CELL_4   =   CELLS[80:99];
assign CELL_5   =   CELLS[100:119];
```

```
assign CELL_6   =   CELLS[120:139];
assign CELL_7   =   CELLS[140:159];
assign CELL_8   =   CELLS[160:179];
assign CELL_9   =   CELLS[180:199];
assign CELL_10   =   CELLS[200:219];
assign CELL_11   =   CELLS[220:239];
assign CELL_12   =   CELLS[240:259];
assign CELL_13   =   CELLS[260:279];
assign CELL_14   =   CELLS[280:299];
assign CELL_15   =   CELLS[300:319];
assign CELL_16   =   CELLS[320:339];
assign CELL_17   =   CELLS[340:359];
assign CELL_18   =   CELLS[360:379];
assign CELL_19   =   CELLS[380:399];
top_module topmod(  .gsres(GSRES), .clock(CLOCK),
                    .Panel(PANEL),
                    .C(CELLS)
                    );
endmodule //
```

d. ModelSim do file

```
vlib work // set working library
vmap work work // set working directory
vlog test_vector20.v // compile HDL files
vsim -novopt test_vector20 // simulate Design
add wave sim:/test_vector20/* // Add signals to waveform window
run 500 // Run 500 ns
```

2. Logic Synthesis Tcl Script

```
#set some per design variables
```

```
set TOPLEVEL  "top_module"
set LOG_PATH  "log1/"
set GATE_PATH  "lib/"set RTL_PATH  "src/"
set STAGE  "gsclib"
set CLK  "clock"
set RST  "gsres"
set CLK_PERIOD 2.50; # 10ns
set CLK_UNCERTAINTY  0.3; # 200ps
# Set search path for verilog include files set search_path [concat $RTL_PATH $GATE_PATH]
set link_library   /home/adadalia/adspirop/my/lib/GSCLib_2.0_prelim3.db
set target_library /home/adadalia/adspirop/my/lib/GSCLib_2.0_prelim3.db
#set symbol_library /home/adadalia/adspirop/my/lib/umce13h210t3.sdb
# RTL Synthesis
analyze -library WORK -format verilog /home/adadalia/adspirop/my/src/cell_logic.v
analyze -library WORK -format verilog /home/adadalia/adspirop/my/src/my.v
#synthesize netlist
elaborate top_module  -library WORK
# Set design top
current_design top_module
# check the design for errors such as missing module definisions
check_design  > $LOG_PATH/$TOPLEVEL-$STAGE-check_design.log
set_operating_conditions -library GSCLib_2.0
set_wire_load_model -library GSCLib_2.0
set_timing_ranges -library GSCLib_2.0
create_clock $CLK -period $CLK_PERIOD
set_clock_uncertainty $CLK_UNCERTAINTY [get_clocks $CLK]
set_dont_touch_network [get_clocks $CLK]
remove_driving_cell $RST
set_drive 0 $RST
set_dont_touch_network $RST
set_ideal_network [get_port clock]
set_input_delay 0.5 -clock $CLK [remove_from_collection [remove_from_collection [all_inputs] [get_port $CLK]] [get_port $RST]]
set_load 0.003 [remove_from_collection [remove_from_collection [all_inputs] [get_port $CLK]] [get_port $RST]]
set_load  0.003 [all_outputs]
set_max_delay 0.5 -to [all_outputs]
```

```
set_max_area 0
# Link all blocks and uniquify them
 Link
 Uniquify
set_ultra_optimization –f
compile -map_effort high -area_effort high -incremental_mapping
remove_unconnected_ports -blast_buses [get_cells -hierarchical *]
# Save current design using both db and verilog formats
write -hierarchy -format verilog -output $LOG_PATH/$TOPLEVEL-$STAGE.v
write_sdf $LOG_PATH/$TOPLEVEL-$STAGE.sdf
write_sdc $LOG_PATH/$TOPLEVEL-$STAGE.sdc
# Basic reports
report_area                      > $LOG_PATH/$TOPLEVEL-$STAGE-area.log
report_timing -nworst 10        > $LOG_PATH/$TOPLEVEL-$STAGE-timing.log
report_hierarchy                > $LOG_PATH/$TOPLEVEL-$STAGE-hierarchy.log
report_cell                > $LOG_PATH/$TOPLEVEL-$STAGE-cell.log
report_power               > $LOG_PATH/$TOPLEVEL-$STAGE-power.log
report_resources                > $LOG_PATH/$TOPLEVEL-$STAGE-resources.log
report_constraint  -all_violators  -verbose                                    >
$LOG_PATH/$TOPLEVEL-$STAGE-constraint.log
```

3. Place & Route Tcl Scripts

a. Configuration file

```
################################################
#                                              #
#  FirstEncounter Input configuration file     #
#                                              #
################################################
global rda_Input
set cwd .
set rda_Input(ui_netlist) "../src/top_module-gsclib.v"
set {rda_Input(ui_netlisttype)} {Verilog}
set {rda_Input(ui_ilmlist)} {}
set {rda_Input(ui_settop)} {0}
set {rda_Input(ui_topcell)} {$TOP}
set {rda_Input(ui_celllib)} {}
set {rda_Input(ui_iolib)} {}
set {rda_Input(ui_areaiolib)} {}
set {rda_Input(ui_blklib)} {}
set {rda_Input(ui_kboxlib)} {}
set rda_Input(ui_timelib) "../timing/$LIB.tlf"
set {rda_Input(ui_smodDef)} {}
set {rda_Input(ui_smodData)} {}
set {rda_Input(ui_dpath)} {}
set {rda_Input(ui_tech_file)} {}
set rda_Input(ui_io_file) ""
set rda_Input(ui_timingcon_file) "../sdc/$TOP.sdc"
set {rda_Input(ui_buf_footprint)} {buf}
set {rda_Input(ui_delay_footprint)} {}
set {rda_Input(ui_inv_footprint)} {inv}
set rda_Input(ui_leffile) "../tech/$LIB.0.lef”
set {rda_Input(ui_core_cntl)} {aspect}
set {rda_Input(ui_aspect_ratio)} {1.0}
set {rda_Input(ui_core_util)} {0.70}
set {rda_Input(ui_core_height)} { }
set {rda_Input(ui_core_width)} { }
set {rda_Input(ui_core_to_left)} {80}
set {rda_Input(ui_core_to_right)} {80}
```

```
set {rda_Input(ui_core_to_top)} {80}
set {rda_Input(ui_core_to_bottom)} {80}
set {rda_Input(ui_delay_limit)} {1000}
set {rda_Input(ui_net_delay)} {1000.0ps}
set {rda_Input(ui_net_load)} {0.5pf}
set {rda_Input(ui_in_tran_delay)} {120.0ps}
set {rda_Input(ui_captbl_file)} {}
set {rda_Input(ui_cap_scale)} {1.0}
set {rda_Input(ui_xcap_scale)} {1.0}
set {rda_Input(ui_res_scale)} {1.0}
set {rda_Input(ui_shr_scale)} {1.0}
set {rda_Input(ui_time_unit)} {none}
set {rda_Input(ui_cap_unit)} {}
set {rda_Input(ui_pwrnet)} {VDD}
set {rda_Input(ui_gndnet)} {VSS}
set {rda_Input(flip_first)} {0}
set {rda_Input(double_back)} {1}
set rda_Input(ui_pg_connections) [list \
                        {PIN:POWR::} \
                        {TIEHI::} \
                        {PIN:GRND::} \
                        {TIELO::} \

                              ]
set {rda_Input(PIN:POWR::)} {VDD}
set {rda_Input(TIEHI::)} {VDD}
set {rda_Input(PIN:GRND::)} {VSS}
set {rda_Input(TIELO::)} {VSS}
```

b. init.tcl

```
# Setup Design
# Set top level module
set TOP top_module
# Reports Directory
set rpt_dir report/
# Incoming SDC Constraints Directory
set cts_dir cts/
# Encounter Command Scripts Directory
```

```
set cmd_dir cmd/
# Final Encounter Output Directory
# for TLF, LEF, DEF etc.
set out_dir output/
# Read in libraries and verilog code
loadConfig $TOP.conf
commitConfig
# Set Encounter in CTE mode
# necessary for checkTA, reportTA
setCteReport
```

c. timing.tcl

```
# Initial Timing Check
# No wires yet, so this is the best case check
# Setup Timing Analysis
setAnalysisMode -setup -async -skew -autoDetectClockTree
buildTimingGraph -ignoreNetLoad
setCteReport
# Check Constraints for consistency
checkTA -verbose >  report/checkTA
# Create Timing Reports
reportTA -max_paths 100 -outfile report/reportTa.max.paths
reportTA -format {hpin arc cell delay arrival slew load} -late -max_points 100 -net -summary > report/reportTA.initial.summary
reportTA -format {hpin arc cell delay arrival slew load} -late -max_points 100 -net > report/reportTA.initial.slack
```

d. Floorplan.tcl

```
floorplan -r 1.0 0.70 100 100 100 100
```

e. Power.tcl

```
# Add Power Rings to design
addRing -spacing_bottom 3 -width_left 10 -width_bottom 10 -width_top 10 -spacing_top 3 -layer_bottom Metal5 -stacked_via_top_layer Metal6 -width_right 10 -around core -offset_bottom 2 -layer_top Metal5 -threshold 0.33 -offset_left 2 -spacing_right 3 -spacing_left 3 -offset_right 2 -offset_top 2 -layer_right Metal6 -stacked_via_bottom_layer Metal1 -layer_left Metal6 -nets { VDD VSS }
```

```
addStripe    -set_to_set_distance  100  -stacked_via_top_layer  Metal6  -
spacing 2 -xleft_offset 50 -merge_stripes_value 0.33 -layer Metal6 -width
10 -stacked_via_bottom_layer Metal1 -nets { VDD VSS }
```

f. Place.tcl

```
# This Script places the cells,
# runs verification and changes to placement view
# Place standard cells
amoebaPlace -highEffort -timingdriven
# Check Placement
checkPlace
# Change to Amoeba View
setDrawMode place
# Now perform Congestion Analysis
# Route Design
trialRoute -lowEffort
# Provide Congestion
puts " Writing Congestion Information to File"
dumpCongestArea  report/congestionmap.txt
# Extract RC
setExtractRCMode -detail -assumeMetFill
extractRC
# Perform Timing Analysis
setAnalysisMode -setup -async -skew -autoDetectClockTree
buildTimingGraph
reportTA  -format  {hpin  arc  cell  delay  arrival  slew  load}  -late  -
max_points 100 -net -summary > report/reportTA.placed.summary

reportTA  -format  {hpin  arc  cell  delay  arrival  slew  load}  -late  -
max_points 100 -net  > report/reportTA.placed.slack
```

g. Cts.tcl

```
# Clock Tree Synthesis
# Load Clock Tree Configuration File
createClockTreeSpec  -output  ../timing/$TOP.ctstch  -bufFootprint  buf  -
invFootprint inv
specifyClockTree -clkfile ../timing/$TOP.ctstch
checkUnique
```

```
# Synthesise Clock Tree
ckSynthesis -rguide report/$TOP.rguide -report report/$TOP.ctsrpt -macromodel report/$TOP.ctsmdl -fix_added_buffers
saveClockNets -output report/clock_nets.ctsntf
# Post CTS Timing Analysis
# Extract after CTS
trialRoute -highEffort -guide report/$TOP.rguide
extractRC
# Write Clock Tree reports (as html)
reportClockTree -postRoute -localSkew -report report/gol.post_troute_local.ctsrpt
reportClockTree -postRoute -report report/gol.post_troute.ctsrpt
# Timing Analysis post Clock Tree
setAnalysisMode -setup -async -skew -autoDetectClockTree
buildTimingGraph
reportTA -format {hpin arc cell delay arrival slew load} -late -max_points 100 -net -summary > report/reportTA.cts.summary
 reportTA -format {hpin arc cell delay arrival slew load} -late -max_points 100 -net > report/reportTA.cts.slack
```

h. IPO.tcl

```
setIPOMode -nofixFanoutLoad -noRestruct -noSwapPin -highEffort -FixDrc -addPortAsNeeded -incrTrialRoute -targetSlack 0.0 -maxDensity 0.95
```

i. Addfiller.tcl

```
# Add Filler cells to the design
puts "### Adding Filler Cells"
addFiller -cell FILLCELL_X4 -prefix FILLER
addFiller -cell FILLCELL_X2 -prefix FILLER
```

j. Routepower.tcl

```
# Route Power Nets
sroute –noPadPins -noPadRings -nets {VDD VSS}
```

k. Route.tcl

```
# Setting NanoRoute Options"
set critNet {@clock}
setAttribute -net $critNet -weight 100
setAttribute -net $critNet -avoid_detour true
setAttribute -net $critNet -bottom_preferred_routing_layer 8
setAttribute -net $critNet -top_preferred_routing_layer 9
```

```
#setAttribute -net $critNet -shield_net VSS
setNanoRouteMode -quiet drouteFixAntenna true
setNanoRouteMode -quiet routeInsertAntennaDiode false
setNanoRouteMode -quiet routeAntennaCellName default
setNanoRouteMode -quiet routeWithTimingDriven true
setNanoRouteMode -quiet routeWithTimingOpt false
setNanoRouteMode -quiet optimizeBi false
setNanoRouteMode -quiet optimizeGs false
setNanoRouteMode -quiet optimizeFixSetupTime true
setNanoRouteMode -quiet optimizeTargetSetupSlack 0.000000
setNanoRouteMode -quiet optimizeFixMaxCap false
setNanoRouteMode -quiet optimizeFixHoldTime false
setNanoRouteMode -quiet optimizeTargetHoldSlack 0.000000
setNanoRouteMode -quiet optimizeFixMaxTran false
setNanoRouteMode -quiet optimizeDontUseCellFile default
setNanoRouteMode -quiet routeWithSiDriven true
setNanoRouteMode -quiet routeSiEffort normal
setNanoRouteMode -quiet siNoiseCTotalThreshold 0.050000
setNanoRouteMode -quiet siNoiseCouplingCapThreshold 0.005000
setNanoRouteMode -quiet routeWithSiPostRouteFix false
setNanoRouteMode -quiet drouteAutoStop true
setNanoRouteMode -quiet routeSelectedNetOnly false
setNanoRouteMode -quiet routeFixPrewire false
setNanoRouteMode -quiet drouteStartIteration default
setNanoRouteMode -quiet envNumberProcessor 4
setNanoRouteMode -quiet routeTopRoutingLayer default
setNanoRouteMode -quiet drouteEndIteration default


# Global and Detail Route with NanoRoute
globalDetailRoute
# Analyze Timing after Routing
setExtractRCMode -detail -noReduce -extendedCapTbl
extractRC
buildTimingGraph
reportTA -format {hpin arc cell delay arrival slew load} -late -max_points
100 -net -summary >  report/reportTA.routed.summary
```

```
reportTA -format {hpin arc cell delay arrival slew load} -late -max_points 100 -net >  report/reportTA.routed.slack
```

l. Verify.tcl

```
# Perform Sign-off and verification
fillNotch -reportfile report/fillnotch.rpt
verifyConnectivity -reportfile  report/connectivity.rpt
# Verify Geometry
# nosameNet comes from Cadence Flow Manual
# noShorts because Filler cells report 1000s of shorts
verifyGeometry -reportfile  report/geometry.rpt -noSameNet -noShorts
```

m. Write.tcl

```
#Write all the results to disk
# Output SPEF File and Report Final Timing
setExtractRCMode -detail -noReduce -extendedCapTbl
extractRC
rcOut -spef  output/$TOP.spef
buildTimingGraph
reportTA -format {hpin arc cell delay arrival slew load} -late -max_points 100 -net -summary >  report/reportTA.final.summary
reportTA -format {hpin arc cell delay arrival slew load} -late -max_points 100 -net >  report/reportTA.final.slack
# Create DEF
defOut -placement -routing -floorplan  output/$TOP.def
# Create LEF
lefOut output/$TOP.lef
# Create GDSII
streamOut output/$TOP.gds -libName DesignLib -stripes 1 -units 2000 -mode ALL
# Create Netlist
saveNetlist -excludeLeafCell output/$TOP.final.v
# Write Constraints for Primetime
writeTimingCon -pt -filePrefix output/$TOP
# Save Encounter Database
saveDesign output/$TOP.enc
```