ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ»

«Ευριστικοί Αλγόριθμοι Νομιμοποίησης της Χωροθέτησης Κελιών στη Φυσική Σχεδίαση»

“Legalization Heuristics for Physical Design”

ΝΕΡΑΝΤΖΑΚΗ ΕΥΑΓΓΕΛΙΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
Υπεύθυνος
ΣΤΑΜΟΥΛΗΣ ΓΕΩΡΓΙΟΣ

Λαμία, 2016

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ
ΚΑΤΕΥΘΥΝΣΗ

«ΠΛΗΡΟΦΟΡΙΚΗ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΑΣΦΑΛΕΙΑ, ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΟΥ ΟΓΚΟΥ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΠΡΟΣΟΜΟΙΩΣΗ»

«Ευριστικοί Αλγόριθμοι Νομιμοποίησης της Χωροθέτησης Κελιών στη Φυσική Σχεδίαση»

“Legalization Heuristics for Physical Design”

ΝΕΡΑΝΤΖΑΚΗ ΕΥΑΓΓΕΛΙΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων
ΣΤΑΜΟΥΛΗΣ ΓΕΩΡΓΙΟΣ

Λαμία, 2016

«Υπεύθυνη Δήλωση μη λογοκλοπής και ανάληψης προσωπικής ευθύνης»

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, και γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπεύθυνα και ενυπογράφως ότι η παρούσα εργασία με τίτλο «Ευριστικοί Αλγόριθμοι Νομιμοποίησης της Χωροθέτησης Κελιών στη Φυσική Σχεδίαση» αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές από τις οποίες χρησιμοποίησα δεδομένα, ιδέες, φράσεις, προτάσεις ή λέξεις, είτε επακριβώς (όπως υπάρχουν στο πρωτότυπο ή μεταφρασμένες) είτε με παράφραση, έχουν δηλωθεί κατάλληλα και ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Η ΔΗΛΟΥΣΑ

Ημερομηνία

Υπογραφή

«Ευριστικοί Αλγόριθμοι Νομιμοποίησης της Χωροθέτησης Κελιών στη Φυσική Σχεδίαση»

“Legalization Heuristics for Physical Design”

ΝΕΡΑΝΤΖΑΚΗ ΕΥΑΓΓΕΛΙΑ

Τριμελής Επιτροπή:

ΣΤΑΜΟΥΛΗΣ ΓΕΩΡΓΙΟΣ

ΛΟΥΚΟΠΟΥΛΟΣ ΑΘΑΝΑΣΙΟΣ

ΕΥΜΟΡΦΟΠΟΥΛΟΣ ΝΕΣΤΩΡ

Επιστημονικός Σύμβουλος:
ΔΑΔΑΛΙΑΡΗΣ ΑΝΤΩΝΙΟΣ

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον καθηγητή του τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων και βασικό επιβλέποντα αυτής της πτυχιακής κ. Γεώργιο Σταμούλη που μου έδωσε την ευκαιρία να πραγματοποιήσω αυτή την εργασία. Η υποστήριξή του, η αμέριστη συμπαράστασή του, αλλά και οι διαρκείς και εύστοχες υποδείξεις του βοήθησαν στην έγκαιρη ολοκλήρωση αυτής της εργασίας.

Επιπρόσθετα, θα ήθελα να ευχαριστήσω τον διδάσκοντα του τμήματος Πληροφορικής κ. Δαδαλιάρη Αντώνιο για την προγενέστερη δουλειά του πάνω στον τομέα καθώς και την μεταλαμπάδευση της γνώσης αυτής.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου, που μου συμπαραστάθηκαν καθ' όλη την διάρκεια των σπουδών μου.

Περιεχόμενα

Εισαγωγή

Η παρούσα μελέτη αφορά τον αλγόριθμο Tetris, έναν από τους αλγορίθμους που χρησιμοποιούνται για την επίλυση του προβλήματος της νομιμοποίησης των κελιών, σε μία Φυσική Σχεδίαση καθώς επίσης και τις διαφορετικές προσεγγίσεις του που υλοποιήθηκαν.

Αρχικά γίνεται μία περιγραφή των βημάτων που ακολουθούνται κατά τη σχεδίαση ενός ολοκληρωμένου κυκλώματος με μεγαλύτερη ανάλυση στο στάδιο της Φυσικής Σχεδίασης. Στη συνέχεια προσδιορίζεται με μεγαλύτερη λεπτομέρεια το πρόβλημα που επιλύεται με τον αλγόριθμο Tetris, ενώ ακολουθεί η ανάλυση του ίδιου του αλγορίθμου.

Αμέσως μετά αναλύονται οι υλοποιήσεις που πραγματοποιήθηκαν καθώς επίσης και τα αποτελέσματα που επιτεύχθηκαν. Στο παράρτημα παραθέτονται πληροφορίες για τα κυκλώματα που χρησιμοποιήθηκαν καθώς επίσης και όλα τα αποτελέσματα των πειραμάτων που γίνανε.

Περίληψη

Κατά το στάδιο της Φυσικής Σχεδίασης σε μία ροή σχεδίασης, ένα από τα βασικά στάδια που ακολουθούνται είναι αυτό του placement. Επιλύοντας το πρόβλημα της χωροθέτησης με την εφαρμογή του Gordian Placement Algorithm είχαμε να αντιμετωπίσουμε το πρόβλημα της νομιμοποίησης (legalization) των κελιών. Η εργασία αυτή επικεντρώνεται στην υλοποίηση διαφορετικών παραλλαγών του βασικού αλγορίθμου για legalization Tetris.

Η εφαρμογή διαφορετικών προσεγγίσεων την βασικής ιδέας φαίνεται να επιφέρει εντυπωσιακά αποτελέσματα. Πιο συγκεκριμένα, υλοποιήθηκαν έξι παραλλαγές του βασικού Tetris για legalization και μία για detailed placement επικεντρώνοντας μεγαλύτερη σημασία στη μείωση του συνολικού μήκους καλωδίου για τη σύνδεση των κελιών. Στην συντριπτική τους πλειοψηφία όλες οι νέες προσεγγίσεις βελτιώνουν τα αποτελέσματα του βασικού αλγορίθμου.

Abstract

Placement and legalization are two of the most important steps in the Physical Design phase of any ASIC design flow. A relatively simple way to tackle the placement problem is the application of the Gordian algorithm which computes the optimal positions of the cells, based on an objective function (usually concerning the total interconnect wire length) but produces a global placement with overlaps. Cell overlaps are then eliminated with the use of legalization techniques and algorithms, which is the focus of this thesis.

More specifically, a set of approaches and heuristics for the classic Tetris legalization algorithm and a detailed placement technique are presented that target the overall design wire length, producing better in comparable runtime.

Κεφάλαιο 1: Φυσική Σχεδίαση

1.1 Ροή Σχεδίασης Ολοκληρωμένων Κυκλωμάτων

Η διαδικασία που ακολουθείται κατά την σχεδίαση ενός ολοκληρωμένου κυκλώματος αποτελείται από πολλαπλά βήματα, με βασικότερα των οποίων τα ακόλουθα:

- *Feasibility Study:* Ανάλυση δυνατότητας ολοκλήρωσης του απώτερου στόχου.
- *Die Size Estimation:* Εκτίμηση του απαιτούμενου χώρου.
- *Functional Verification:* Επαλήθευση της λογικής και της λειτουργικότητας της σχεδίασης.
- *RTL Design:* Περιγραφή της λειτουργίας του κυκλώματος σε επίπεδο καταχωρητών (register-transfer level) .
- *RTL Simulation:* Προσομοίωση της λειτουργίας του κυκλώματος βάσει του RTL Design.
- *Logic Simulation:* Προσομοίωση της λειτουργίας της σχεδίασης.
- *Floor Planning:* Σχηματική αναπαράσταση.
- *Layout:* Αναπαράσταση του ολοκληρωμένου κυκλώματος με μια σειρά γεωμετρικών σχημάτων.
- *Static Timing Analysis:* Μελέτη του χρονισμού του ολοκληρωμένου κυκλώματος.
- *Layout Review:* Επανεξέταση του layout που έχουμε σχεδιάσει.
- *Design For Test:* Χρήση συγκεκριμένων τεχνικών σχεδίασης οι οποίες προσδίδουν ιδιαίτερα χαρακτηριστικά στο κύκλωμα.
- *Automatic Test Pattern Generation:* Εύρεση της κατάλληλης αλληλουχίας εισόδων.
- *Design For Manufacturability:* Εφαρμογή μιας σειράς τεχνικών οι οποίες τροποποιούν κατάλληλα το κύκλωμα έτσι ώστε η υλοποίησή του σε βιομηχανικό περιβάλλον να καθίσταται ευκολότερη.
- *Mask Data Preparation:* Μετάφραση της layout περιγραφής σε καταλληλότερη μορφή για έναν photomask writer.
- *Wafer Fabrication:* Η διαδικασία κατά την οποία δημιουργείται το ολοκληρωμένο κύκλωμα.
- *Packaging:* Το κύκλωμα «συσκευάζεται» σε κάποιο κεραμικό ή πλαστικό υλικό προκειμένου να αποφευχθεί η φθορά του.
- *Device Characterization:* Η διαδικασία κατά την οποία συγκεντρώνουμε και παρουσιάζουμε τα ιδιαίτερα χαρακτηριστικά της τελικής υλοποίησης.

- *Yield Analysis:* Συλλογή και ανάλυση των κατάλληλων δεδομένων που απαιτούνται για τον εντοπισμό και την διόρθωση αστοχιών που προκύπτουν κατά την λειτουργία του ολοκληρωμένου.

Τα παραπάνω βήματα χωρίζονται σε τρεις επιμέρους φάσεις

- System Level Design
- RTL Design
- Physical Design

Στην παρούσα εργασία θα επικεντρωθούμε στην τρίτη φάση, τη Φυσική Σχεδίαση, (Physical Design/Back-End Design).
Πιο αναλυτικά, στο στάδιο αυτό ακολουθούνται τα εξής βήματα:

- *Design Netlist:* Ένα netlist σε επίπεδο πυλών (gate level netlist) είναι κατ' ουσίαν το αποτέλεσμα που προκύπτει από την διαδικασία της σύνθεσης (synthesis) ενός ψηφιακού κυκλώματος.
- *Floor Planning:* Στο βήμα αυτό κάνουμε μια πρώτη εκτίμηση του συνολικού χώρου που απαιτείται για τις βασικές δομικές μονάδες του chip και καθορίζουμε τις σχετικές τους θέσεις εντός του προκείμενου χώρου.
- *Partitioning:* Στο βήμα αυτό ο σχεδιαστής καλείται να βρει τον κατάλληλο τρόπο διαίρεσης της περιοχής, που καταλαμβάνει το chip, σε μικρότερες και ευκολότερα διαχειρίσιμες περιοχές.
- *Placement:* Η χωροθέτηση πραγματοποιείται σε τέσσερα βήματα, που απώτερο στόχο έχουν την βέλτιστη τοποθέτηση των κελιών της σχεδίασης στον προκαθορισμένο χώρο.
- Pre-placement Optimization, In-Placement Optimization, Post-Placement Optimization, Post-Placement Optimization after Clock Tree Synthesis
- *Clock Tree Synthesis:* Ο στόχος του clock tree synthesis είναι να ελαχιστοποιήσουμε το skew και το insertion delay.
- *Routing:* Υπάρχουν δύο τύποι routing, το global routing και το detailed routing. Το global routing τοποθετεί τα routing resources τα οποία χρησιμοποιούνται για τις συνδέσεις μεταξύ των κελιών, ενώ το detailed routing αναθέτει συγκεκριμένα μονοπάτια (routes) σε συγκεκριμένα επίπεδα μετάλλου.
- *Physical Verification*: Στο τελευταίο στάδιο της φυσικής σχεδίασης ελέγχουμε την ορθότητα του layout που έχει παραχθεί.

Στην παρακάτω εικόνα, εμφανίζεται μία τυπική ροή σχεδίασης με μεγαλύτερη έμφαση στη φάση της Φυσικής Σχεδίασης.

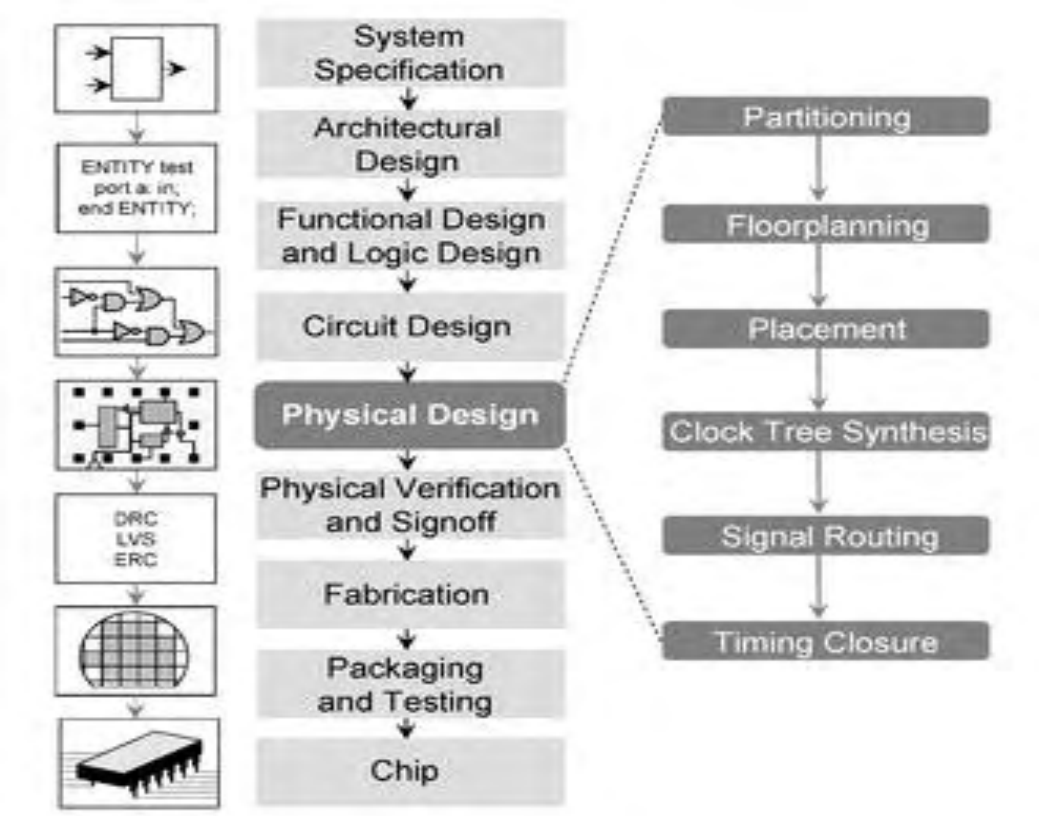


Figure 1 - VLSI Circuits Design Flow

Αξίζει να τονίσουμε ότι κατά τη διαδικασία σχεδίασης ενός ολοκληρωμένου κυκλώματος ανάλογα με τη σημαντικότητα και τη βαρύτητα που δίνεται σε κάθε ένα από τα παραπάνω βήματα ακολουθείται διαφορετική ροή σχεδίασης. Άλλες φορές πιο λεπτομερής και άλλες πιο γενικευμένη.

Η παρουσίαση της παρακείμενης ροής σχεδίασης, με χρήση των συγκεκριμένων εργαλείων, γίνεται λόγω της ευρείας χρήσης της από την πλειοψηφία των, σχετιζόμενων με τον κλάδο, του σχεδιασμού ολοκληρωμένων κυκλωμάτων, εταιρειών.

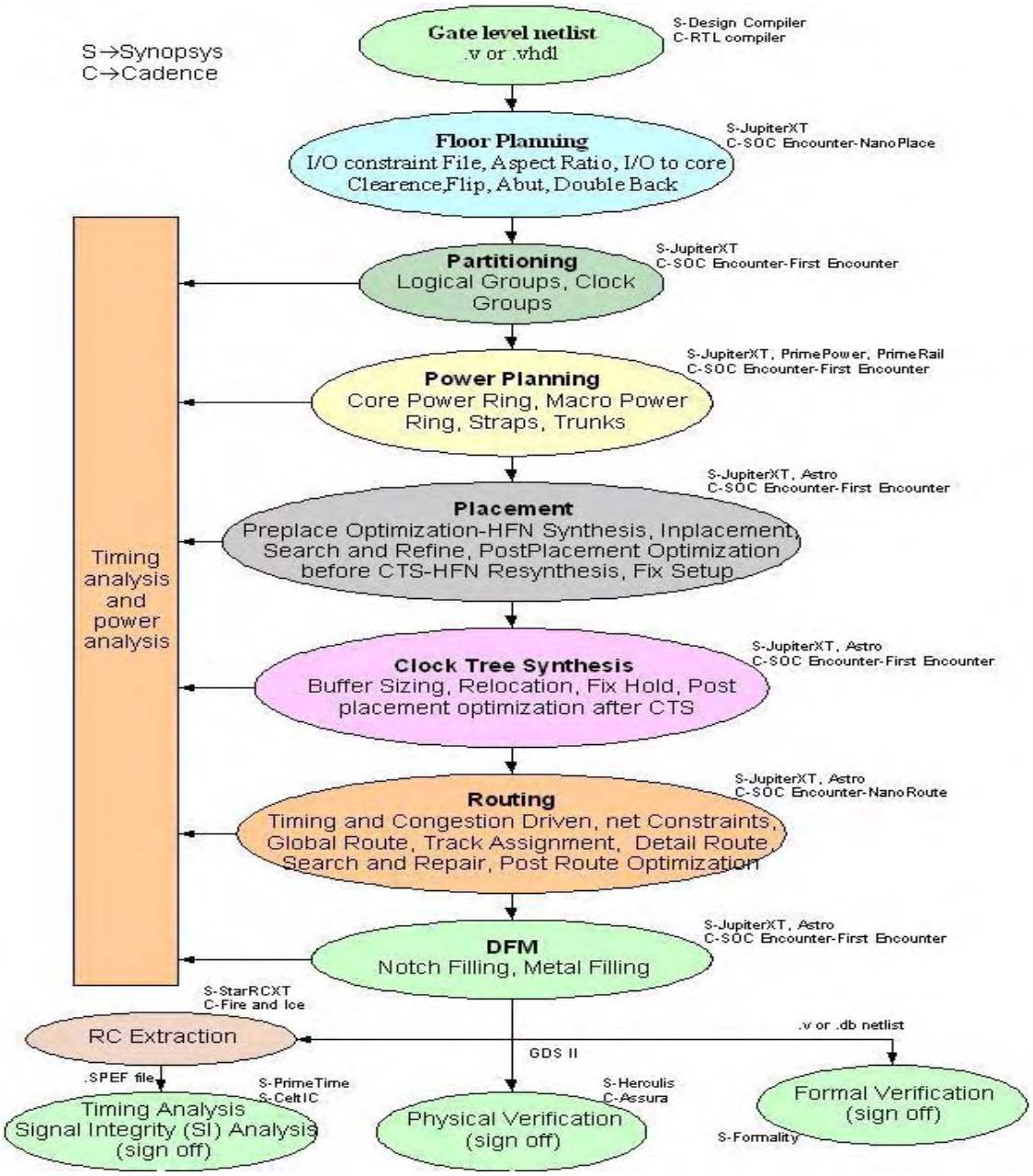


Figure 2 - Design Flow & CAD Tools

Κεφάλαιο 2: Χωροθέτηση Ολοκληρωμένων Κυκλωμάτων

2.1 Ορισμός του προβλήματος

Ως πρόβλημα χωροθέτησης ορίζεται η διαδικασία εύρεσης βέλτιστου τρόπου τοποθέτησης μιας ομάδας αντικειμένων σε προκαθορισμένο χώρο με στόχο την ελαχιστοποίηση της τιμής μίας αντικειμενικής συνάρτησης. Στη συγκεκριμένη περίπτωση την ομάδα αντικειμένων αποτελούν κελιά ενώ την τιμή της αντικειμενικής συνάρτησης αποτελεί το συνολικό μήκος καλωδίου που χρησιμοποιείται για τη σύνδεση των κελιών.

2.2 Χωροθέτηση Ολοκληρωμένων Κυκλωμάτων

Η διαδικασία του Placement αποτελεί την επίλυση του προβλήματος χωροθέτησης των κελιών ενός κυκλώματος. Το τελικό αποτέλεσμα αποτυπώνεται σε μορφή layout. Στην συγκεκριμένη εργασία χρησιμοποιήθηκε το standard cell layout, δηλαδή, όλα τα κελιά έχουν το ίδιο ύψος και διαφέρουν ως προς το μήκος. Τα κελιά τοποθετούνται σε σειρά πάνω σε προκαθορισμένες γραμμές που απέχουν σταθερή απόσταση μεταξύ τους.

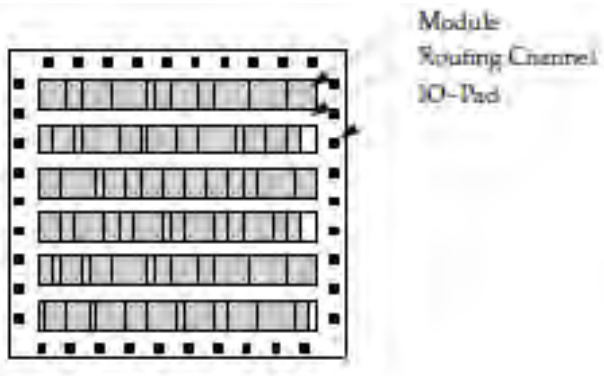
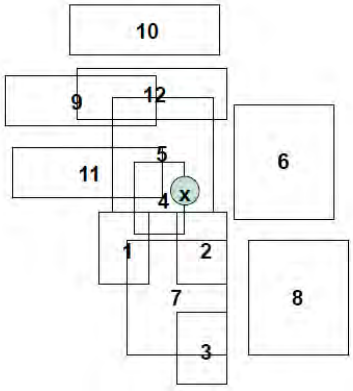


Figure 3 - Standard cell layout model

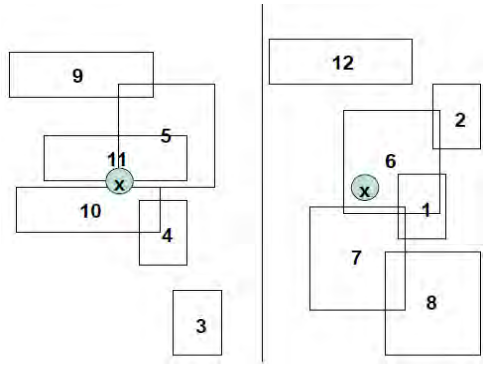
Ο αλγόριθμος που χρησιμοποιήθηκε για την επίλυση του προβλήματος χωροθέτησης των κελιών είναι ο Gordian. Βασικά του χαρακτηριστικά είναι η εφαρμογή του global optimization σε κάθε βήμα και η δημιουργία ορθογώνιων περιοχών στο στάδιο του partitioning. Πιο αναλυτικά,

- Τοποθετούνται όλα τα κελιά εντός της προκαθορισμένης περιοχής
- Δημιουργούνται ομάδες κελιών με χρήση κατάλληλων μεθόδων τμηματοποίησης τηρώντας τους κανόνες καθολικής βελτιστοποίησης του έχουν οριστεί.
- Όταν οι ομάδες που έχουν δημιουργηθεί αποτελούνται από αριθμό κελιών μικρότερο από αυτόν που έχει ορισθεί στην αρχή της εκτέλεσης του αλγορίθμου, τότε τοποθετούνται καταλλήλως τα κελιά στο διαθέσιμο χώρο.

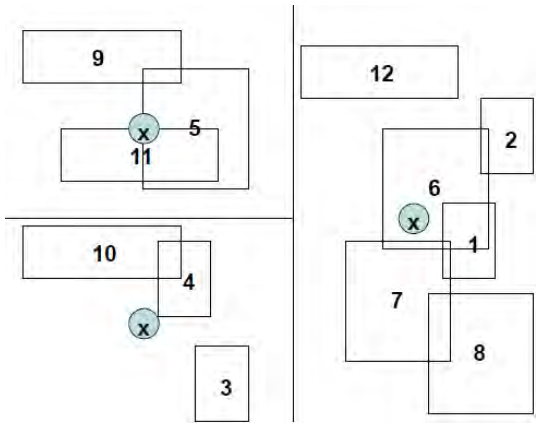
Ακολουθεί ένα παράδειγμα εφαρμογής του αλγορίθμου Gordian σε ένα κύκλωμα έντεκα πυλών με ελάχιστο αριθμό κελιών ανά εξεταζόμενη περιοχή ίσο με τρία.



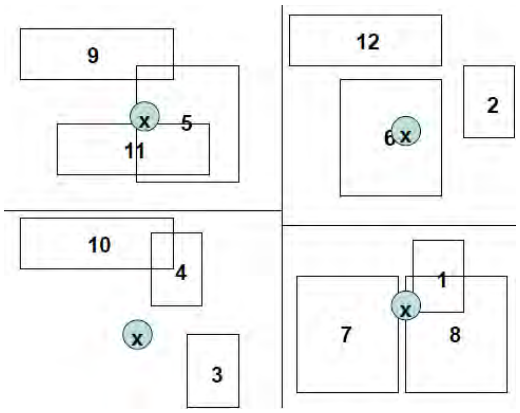
Τυχαία αρχική τοποθέτηση των κελιών



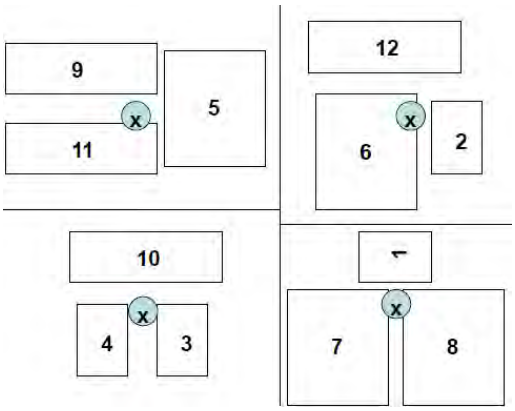
1ο partitioning και αναπροσαρμογή των θέσεων



2ο partitioning και αναπροσαρμογή των θέσεων



3ο partitioning και αναπροσαρμογή των θέσεων



4ο partitioning, αναπροσαρμογή των θέσεων και legalization

Το global placement έχει ως αποτέλεσμα ένα αρχικό placement με αλληλεπικαλύψεις κελιών. Μπορεί να εφαρμοσθεί πολλές φορές για να υπάρξει βελτίωση στο τελικό αποτέλεσμα αλλά για να επιλυθούν όλες οι επικαλύψεις (overlaps) θα πρέπει να εφαρμοσθεί κάποιος αλγόριθμος νομιμοποίησης (legalization).

2.3 Νομιμοποίηση Ολοκληρωμένων Κυκλωμάτων

Η εφαρμογή του global placement οδηγεί σε μία σχεδίαση με κελιά επικαλυπτόμενα μεταξύ τους ή κελιά τοποθετημένα εκτός της προκαθορισμένης περιοχής (υπερχείλιση/overflow). Για να μπορέσουμε να έχουμε σαν αποτέλεσμα μία νομιμοποιημένη σχεδίαση πρέπει να εφαρμόσουμε κάποιο αλγόριθμο ώστε να επιλυθούν οι όποιες επικαλύψεις ή υπερχειλίσεις υπάρχουν.

Η εργασία αυτή βασίστηκε στον αλγόριθμο νομιμοποίησης Tetris ώστε να υλοποιηθούν επιπλέον παραλλαγές του.

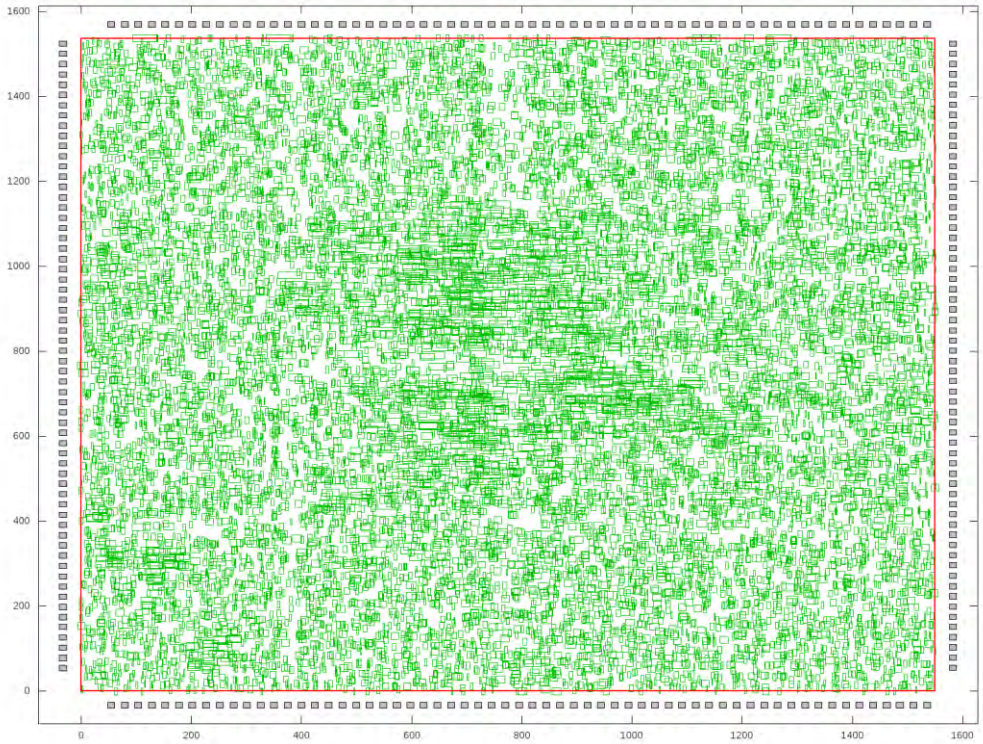


Figure 4 - Αρχική κατάσταση, μετά την εφαρμογή του Gordian Algorithm

Figure 5 - Τελική κατάσταση, μετά την εφαρμογή μιας παραλλαγής του Tetris Algorithm

2.4 Λεπτομερής Χωροθέτηση Ολοκληρωμένων Κυκλωμάτων

Μετά την εφαρμογή αλγορίθμων νομιμοποίησης στο κύκλωμα, έχουμε σαν αποτέλεσμα μία legalized σχεδίαση. Ωστόσο, η σχεδίαση είναι δυνατό να επιδέχεται επιπλέον βελτιώσεις. Η λεπτομερής χωροθέτηση αποτελεί αυτό ακριβώς το στάδιο. Η υλοποίηση μικρών αλλαγών μπορεί να οδηγήσει σε επιπλέον βελτιστοποιήσεις, όσον αφορά το συνολικό μήκος καλωδίου για την σύνδεση των κελιών, την πυκνότητα επιμέρους περιοχών του κυκλώματος, τον χρονισμό του ή οποιαδήποτε άλλη μετρική έχει καθοριστεί.

Κεφάλαιο 3: Αλγόριθμος Tetris

3.1 Βασικός Αλγόριθμος

Ο αλγόριθμος Tetris για legalization προτάθηκε από τον Hill το 2002, και είναι γρήγορος και αποτελεσματικός για εύκολα προβλήματα legalization. Μπορεί να χρησιμοποιηθεί για σχεδιάσεις που αποτελούνται από standard cells, δηλαδή τα κελιά έχουν σταθερό ύψος ενώ ποικίλουν ως προς το μήκος, αλλά εάν τροποποιηθεί μπορεί να χρησιμοποιηθεί ώστε να δέχεται σαν είσοδό του σχεδιάσεις με mixed – sized cells ή macroblocks ή obstacles. Στην παρούσα εργασία χρησιμοποιήθηκαν μόνο σχεδιάσεις από standard cells.

Ο αλγόριθμος λειτουργεί ως εξής: πρώτο βήμα αποτελεί η δημιουργία μιας λίστας με όλα τα κελιά της σχεδίασης, διατεταγμένα ως προς την συντεταγμένη x. Στη συνέχεια για κάθε ένα από τα κελιά της λίστας υπολογίζεται το κόστος μετακίνησης σε κάθε μια από τις σειρές. Το κόστος είναι η ευκλείδεια απόσταση μεταξύ αρχικής θέσης και προορισμού, ενώ η πρώτη διαθέσιμη θέση σε κάθε σειρά του κυκλώματος είναι γνωστή. Ακολουθεί η τοποθέτηση του κελιού στην πιο συμφέρουσα σειρά (μικρότερο κόστος), και τελευταίο βήμα είναι η ανανέωση της τιμής της πρώτης διαθέσιμης θέσης στην συγκεκριμένη σειρά. Όταν τοποθετηθεί ένα κελί, δεν εξετάζεται ξανά. Η διαδικασία επαναλαμβάνεται για όλα τα κελιά.

Εάν περιστρέψουμε το κύκλωμα κατά 90Ο αριστερά, ο τρόπος που τοποθετούνται τα κελιά θυμίζει το παιχνίδι Tetris γι’ αυτό και ο αλγόριθμος ονομάστηκε έτσι.

Ένα κομμάτι ψευδοκώδικα που περιγράφει την παραπάνω διαδικασία είναι το ακόλουθο:

```
{C} = All cells to be legalized;
Sort the cells in C by their X-coordinate to get Ls;
Ij = left-most position of each row rj;
for i=1 to the number of cells do
   best = limsup;
   for j=1 to the number of rows do
         cost=displacement of moving cell i in Ls to Ij;
         if cost <= best then
               best=cost;
               best_row = j;
         end if
   end for
```

```
   Move cell i in Ls to the row best_row;
   Ibest_row = Ibest_row + widthi;
end for
```

Ο συγκεκριμένος αλγόριθμος είναι απλός και εύκολα υλοποιήσιμος. Πραγματοποιώντας μικρές τροποποιήσεις στην λογική του αλγορίθμου, μπορούν να υλοποιηθούν εύκολα πολλές παραλλαγές, οι οποίες παρουσιάζουν σημαντικά αποτελέσματα ως προς τις βασικές μετρικές των αλγορίθμων νομιμοποίησης. Επιπρόσθετα, ο προκείμενος αλγόριθμος αποτελεί τη βάση των περισσοτέρων αλγορίθμων νομιμοποίησης, ενώ χρησιμοποιείται ως βασικό μέτρο σύγκρισης για όλους τους νέους legalizers. Είναι γρήγορος και αποτελεσματικός για απλές σχεδιάσεις.

Ωστόσο, παρουσιάζει κάποια μειονεκτήματα. Δεν λαμβάνει υπόψη του τη διασύνδεση των κελιών με αποτέλεσμα να εξαρτάται άμεσα από την ποιότητα του global placement. Το οποίο με τη σειρά του ενδέχεται να οδηγήσει σε βίαιες μετακινήσεις των κελιών. Τέλος, εξαιτίας του παραπάνω λόγου το συνολικό μήκος καλωδίου που μπορεί να χρειαστεί, ίσως και να υπερβαίνει το μέσο όρο σε σύγκριση με τους υπόλοιπους αλγορίθμους legalization.

3.2 LR Tetris

Η πρώτη από τις παραλλαγές που υλοποιήθηκαν ονομάστηκε Left-Right Tetris (LR Tetris). Το ιδιαίτερο γνώρισμα αυτής της παραλλαγής είναι ότι χωρίζεται στη μέση το κύκλωμα ως προς τον άξονα x και εφαρμόζεται ο αλγόριθμος Tetris μία φορά στο αριστερό τμήμα και μία στο δεξιό τμήμα. Το κάθε κελί ανήκει είτε στο αριστερό είτε στο δεξί τμήμα σύμφωνα με τη συντεταγμένη x. Ο αλγόριθμος εφαρμόζεται από αριστερά στα δεξιά για το αριστερό τμήμα και από τα δεξιά προς τα αριστερά για το δεξί τμήμα.

Ένα κομμάτι ψευδοκώδικα που περιγράφει την παραπάνω διαδικασία είναι το ακόλουθο:

```
{C} = All cells to be legalized;
Sort the cells in C by their X-coordinate to get Ls;
Divide area in the middle
Ij = left-most position of each row rj;
Gj = right-most position of each row rj;
for i=1 to the number of cells in Left area do
```

```
   best = limsup;
   for j=1 to the number of rows do
         cost=displacement of moving cell i in Ls to Ij;
         if cost <= best then
               best=cost;
               best_row = j;
         end if
   end for
   Move cell i in Ls to the row best_row;
   Ibest_row = Ibest_row + widthi;
end for
for k=n to the number of cells in Right area do
   best = limsup;
   for j=1 to the number of rows do
         cost=displacement of moving cell k in Ls to Gj;
         if cost <= best then
               best=cost;
               best_row = j;
         end if
         k--;
   end for
   Move cell k in Rs to the row best_row;
   Ibest_row = Ibest_row + widthi;
end for
```

Σκοπός της παραλλαγής αυτής, είναι να μειωθεί το κόστος μεταφοράς των κελιών μεταξύ των γραμμών και ως επακόλουθο το μήκος καλωδίου που απαιτείται για τη διασύνδεσή τους. Ταυτόχρονα, παρ' όλο που αλγόριθμος εκτελείται δύο φορές, χρειάζεται λιγότερο χρόνο για να νομιμοποιήσει τις θέσεις των κελιών αφού εκτελείται σε μικρότερο χώρο με μικρότερο πλήθος κελιών. Ένας πολύ απλός τρόπος για να βελτιώσουμε την ταχύτητα εκτέλεσης του LR Tetris, διατηρώντας τα ίδια αποτελέσματα ως προς το μήκος καλωδίου, είναι η παραλληλοποίηση της εκτέλεσης του Tetris στα δύο επιμέρους τμήματα της διαθέσιμης περιοχής.

3.3 Restricted Lines

Η δεύτερη από τις παραλλαγές που υλοποιήθηκαν ονομάστηκε Restricted Lines Tetris. Η ταχύτητα εκτέλεσης του αλγόριθμου Tetris εξαρτάται άμεσα από το βάθος αναζήτησης για την εξεύρεση της καταλληλότερης γραμμής στην οποία θα τοποθετηθεί το εκάστοτε κελί. Επιπλέον, στην πλειοψηφία των περιπτώσεων η ελάχιστη μετατόπιση ενός κελιού ταυτίζεται με τη μετακίνησή του σε κάποια από τις γειτονικές γραμμές. Τα παραπάνω προβλήματα μπορούν να αντιμετωπιστούν μεταλλάσσοντας των αλγόριθμο Tetris έτσι ώστε η αναζήτηση της καταλληλότερης θέσης ενός κελιού να γίνεται σε περιορισμένο πλήθος γραμμών. Ο αριθμός των γραμμών που καθορίζει τη γειτονιά αναζήτησης είναι μεταβλητός και καθορίζεται από το χρήστη κατά την εκτέλεση του αλγορίθμου. Στα αποτελέσματα που παρουσιάζονται σε επόμενο κεφάλαιο πραγματοποιήθηκαν πειράματα με μέγεθος αναζήτησης κυμαινόμενο μεταξύ 10% - 40% του συνολικού αριθμού γραμμών του εκάστοτε κυκλώματος.

Ένα κομμάτι ψευδοκώδικα που περιγράφει την παραπάνω διαδικασία είναι το ακόλουθο:

```
{C} = All cells to be legalized;
Sort the cells in C by their X-coordinate to get Ls;
Ij = left-most position of each row rj;
for i=1 to the number of cells do
   best = limsup;
   for j=1 to the number of x% of rows do
        cost=displacement of moving cell i in Ls to Ij;
        if cost <= best then
             best=cost;
             best_row = j;
        end if
   end for
   Move cell i in Ls to the row best_row;
   Ibest_row = Ibest_row + widthi;
end for
```

3.4 Area k-Cut

Η επόμενη υλοποίηση ονομάστηκε Area k-Cut. Βασική ιδέα της παραλλαγής αυτής είναι η τμηματοποίηση του κυκλώματος σε k ορθογώνια παραλληλόγραμμα και η εφαρμογή του βασικού αλγορίθμου Tetris σε κάθε ένα από αυτά. Αποτέλεσμα είναι να δημιουργηθούν πολλά μικρά τμήματα με διαφορετικό πλήθος κελιών. Αυτό έχει ως επακόλουθο οι μετακινήσεις των κελιών να κοστίζουν λιγότερο και να υπολογίζονται γρηγορότερα. Στα αποτελέσματα που παρουσιάζονται σε επόμενο κεφάλαιο πραγματοποιήθηκαν πειράματα με δημιουργία 4,9,16 και 25 ορθογώνιων παραλληλόγραμμων.

Ένα κομμάτι ψευδοκώδικα που περιγράφει την παραπάνω διαδικασία είναι το ακόλουθο:

```
{C} = All cells to be legalized;
Sort the cells in C by their X-coordinate to get Ls;
Cut Area in k partitions
Ij = left-most position of each row rj; // total number of rows =
   total number of rows * total columns
for k=1 to the number of partitions do
   for i=1 to the number of cells do
         best = limsup;
         for j=1 to the number of rows do
               cost=displacement of moving cell i in Ls to Ij;
               if cost <= best then
                     best=cost;
                     best_row = j;
               end if
         end for
         Move cell i in Ls to the row best_row;
         Ibest_row = Ibest_row + widthi;
   end for
end for
```

3.5 Line Density

Στην παραλλαγή Line Density υλοποιήθηκε μία διαφορετική ιδέα. Είναι βασισμένη στον απλό αλγόριθμο Tetris με τη διαφορά ότι στο στάδιο της τοποθέτησης των κελιών σε κάθε γραμμή δεν λαμβάνεται υπόψη ολόκληρη η γραμμή αλλά ένα ποσοστό αυτής. Υπάρχει ένα άνω όριο σύμφωνα με το οποίο γεμίζει η κάθε γραμμή. Με αυτό τον τρόπο προσπαθούμε να επιτύχουμε μικρή πυκνότητα σε κάθε γραμμή. Ως αποτέλεσμα έχει να μην υπάρχουν γραμμές με πυκνότητα κοντά στο 100% που θα μπορούσανε να παρουσιάσουνε προβλήματα στη δρομολόγηση καθώς επίσης και την αποφυγή σημείων Hotspot, δηλαδή σημεία που μπορεί να παρουσιάσουν θερμικά φαινόμενα υψηλής έντασης.

Ένα κομμάτι ψευδοκώδικα που περιγράφει την παραπάνω διαδικασία είναι το ακόλουθο:

```
{C} = All cells to be legalized;
Sort the cells in C by their X-coordinate to get Ls;
Ibest_row = width – 10% width;
Ij = left-most position of each row rj;
for i=1 to the number of cells do
   best = limsup;
   for j=1 to the number of rows do
         cost=displacement of moving cell i in Ls to Ij;
         if cost <= best then
               best=cost;
               best_row = j;
         end if
   end for
   Move cell i in Ls to the row best_row;
   Ibest_row = Ibest_row + widthi;
end for
```

3.6 Cell k-Cut

Η επόμενη παραλλαγή που υλοποιήθηκε ονομάστηκε Cell k-Cut. Σε αυτή την προσέγγιση, αντί να χωρίζουμε σε τμήματα τη συνολική διαθέσιμη περιοχή τοποθέτησης των κελιών, χωρίζουμε σε ομάδες το συνολικό πλήθος των κελιών. Στη συνέχεια,

χωρίζουμε σε στήλες τη διαθέσιμη περιοχή σύμφωνα με το εμβαδό του συνόλου των κελιών της κάθε ομάδας και εφαρμόζουμε τον βασικό αλγόριθμο Tetris σε κάθε τμήμα. Λόγω της ανομοιομορφίας του μήκους των κελιών, οι στήλες που δημιουργούνται δεν είναι όμοιες μεταξύ τους. Με αυτό τον τρόπο προσπαθούμε να μειώσουμε ακόμη περισσότερο τις μετακινήσεις των κελιών (displacement) καθώς επίσης και το συνολικό μήκος καλωδίου, αφού το τμήμα στο οποίο τρέχει ο αλγόριθμος έχει πιο στενά όρια. Στα αποτελέσματα που παρουσιάζονται σε επόμενο κεφάλαιο πραγματοποιήθηκαν πειράματα με δημιουργία 2,4,6,8 και 10 ομάδων κελιών.

Ένα κομμάτι ψευδοκώδικα που περιγράφει την παραπάνω διαδικασία είναι το ακόλουθο:

```
{C} = All cells to be legalized;
Sort the cells in C by their X-coordinate to get Ls;
Group cells in k groups, cut area in k columns according to the
   total area of cell-group
Ij = left-most position of each row rj; // total number of rows =
   total number of rows * total columns
for k=1 to the number of columns do
   for i=1 to the number of cells do
         best = limsup;
         for j=1 to the number of rows do
               cost=displacement of moving cell i in Ls to Ij;
               if cost <= best then
                     best=cost;
                     best_row = j;
               end if
         end for
         Move cell i in Ls to the row best_row;
         Ibest_row = Ibest_row + widthi;
   end for
end for
```

3.7 Net Displacement

Η τελευταία παραλλαγή που υλοποιήθηκε ονομάστηκε Net Displacement. Βασικό χαρακτηριστικό της είναι ότι λαμβάνεται υπόψη το net στο οποίο ανήκει το κάθε κελί. Πιο

αναλυτικά, ο αλγόριθμος αποτελείται από την παρακάτω διαδικασία, έχοντας ταξινομήσει τη λίστα των κελιών σύμφωνα με τη συντεταγμένη x, ξεκινάμε να εφαρμόζουμε τον βασικό αλγόριθμο Tetris για το πρώτο κελί. Μόλις μετατοπιστεί το κελί στην τελική του θέση, μετατοπίζονται όλα τα κελιά που ανήκουν στο ίδιο net κατά το ίδιο μέγεθος ως προς τον άξονα x. Στη συνέχεια αφαιρώντας το τρέχον κελί από τη λίστα, την ταξινομούμε ξανά σύμφωνα με τις νέες τιμές ως προς τον άξονα x και εφαρμόζουμε Tetris για το επόμενο κελί. Η διαδικασία επαναλαμβάνεται μέχρι να εξετασθούνε όλα τα κελιά. Αξίζει να σημειωθεί ότι όταν κάποιο κελί μετακινηθεί λόγω του ότι ανήκει στο ίδιο net και όχι επειδή είναι το προς εξέταση κελί, δεν μετατοπίζεται ξανά εκτός αν αποτελεί το προς εξέταση κελί.

Σε αυτή την προσέγγιση δίνεται βαρύτητα στη συνδεσιμότητα των κελιών (connectivity) με αποτέλεσμα να μειώνεται εντυπωσιακά το συνολικό μήκος καλωδίου που απαιτείται αφού οι μετατοπίσεις ελαττώνονται σημαντικά. Ωστόσο, λόγω την επαναληπτικής ταξινόμησης που εφαρμόζεται ο συνολικός χρόνος εκτέλεσης είναι μεγαλύτερος σε σύγκριση με τις υπόλοιπες μεθόδους.

Ένα κομμάτι ψευδοκώδικα που περιγράφει την παραπάνω διαδικασία είναι το ακόλουθο:

```
{C} = All cells to be legalized;
Sort the cells in C by their X-coordinate to get Ls;
Ij = left-most position of each row rj;
for i=1 to the number of cells do
   best = limsup;
   for j=1 to the number of rows do
         cost=displacement of moving cell i in Ls to Ij;
         if cost <= best then
               best=cost;
               best_row = j;
         end if
   end for
   x_distance = Move cell i in Ls to the row best_row;
   Move each cell k in net(i) to x_distance;
   Ibest_row = Ibest_row + widthi;
   Sort the cells in C-i by their X-coordinate to get Ls;
end for
```

3.8 Enhanced Tetris

Οι θέσεις των κελιών που έχουν υπολογιστεί κατά το στάδιο του global placement με χρήση του αλγορίθμου Gordian, είναι αρκετά κοντά στις βέλτιστες, όσον αφορά την ελαχιστοποίηση της εκάστοτε αντικειμενικής συνάρτησης, θέσεις. Καθίσταται προφανές ότι τα κελιά πρέπει να μετακινηθούν κατ' ελάχιστο στα στάδια του legalization και του detailed placement, προκειμένου να μην επηρεαστεί η παραπάνω ιδιότητα. Ένας απλός τρόπος για να επιτευχθεί αυτό, είναι η εφαρμογή του Tetris σε κάθε ένα από τα επιμέρους partitions που είχαν δημιουργηθεί κατά την εκτέλεση του Gordian. Τα αποτελέσματα που λαμβάνουμε από την παραπάνω διαδικασία ενδέχεται να παρουσιάζουν overlaps και/ή overflows, λόγω του γεγονότος ότι στον Gordian δεν είναι υποχρεωτικό να υπάρχει ο απαιτούμενος ελεύθερος χώρος σε κάθε partition για να χωρέσουν τα αντίστοιχα κελιά.

Στο σημείο αυτό, λοιπόν, όλα τα κελιά έχουν τοποθετηθεί εντός των γραμμών και απαιτείται μια επιπρόσθετη διαδικασία για την επίλυση του παραπάνω προβλήματος. Εφαρμόζουμε μια πλήρως αναδρομική διαδικασία, η οποία περιλαμβάνει την ανταλλαγή κελιών μεταξύ παρακείμενων σειρών. Το εύρος των σειρών εντός των οποίων ψάχνουμε για τις κατάλληλες ανταλλαγές είναι προκαθορισμένο κατά την αρχή της προκείμενης διαδικασίας, και αυξάνεται μετά από πιθανές αποτυχίες προκειμένου να βρεθούν κατάλληλες θέσεις για όλα τα "προβληματικά" κελιά. Η διαδικασία ολοκληρώνεται όταν εξαλειφθούν πλήρως όλα τα overlaps και τα overflows.

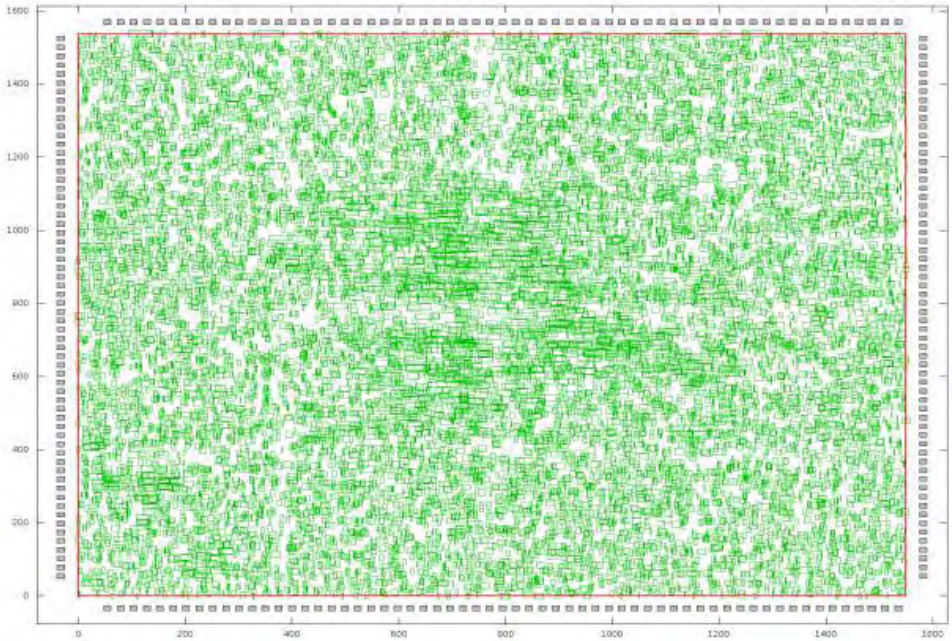


Figure 6 - Αρχική κατάσταση, μετά την εφαρμογή του Gordian Algorithm

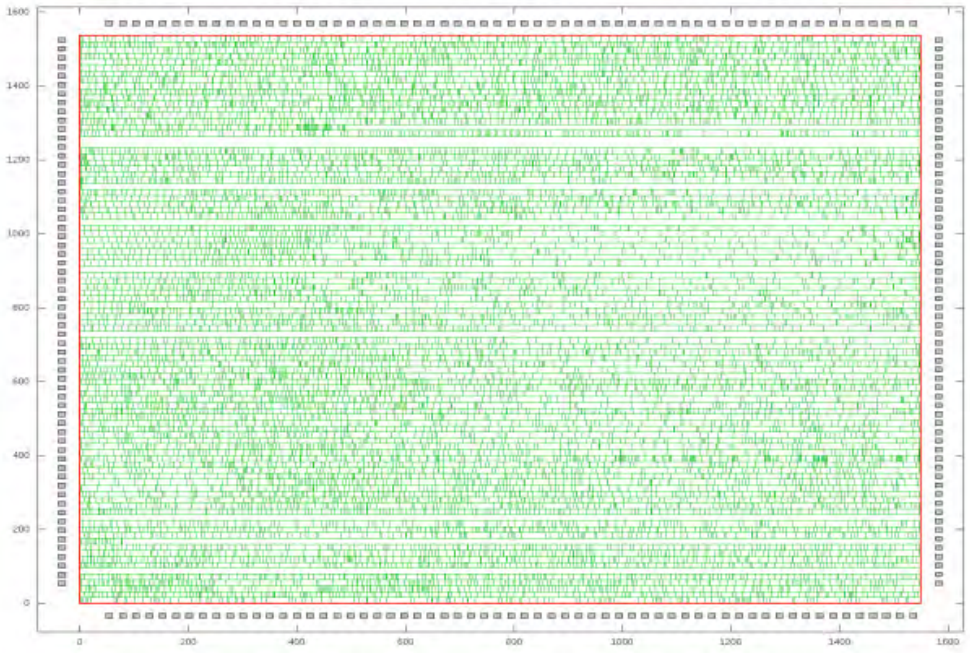


Figure 7 - Τελική κατάσταση, μετά την εφαρμογή του Tetris Algorithm

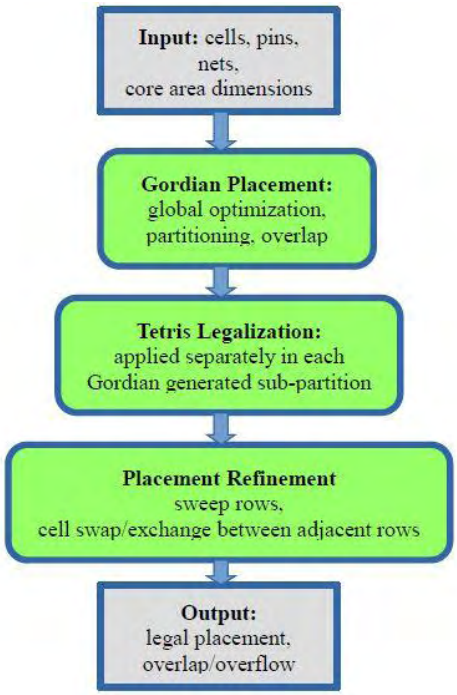


Figure 8 - Διάγραμμα ροής του Enhanced Tetris Algorithm

Κεφάλαιο 4: Αποτελέσματα

Στο παρόν κεφάλαιο θα παρουσιαστούν αναλυτικά τα αποτελέσματα της εφαρμογής των παραπάνω αλγορίθμων στα Circuit Benchmark ISPD98. Τα συγκεκριμένα κυκλώματα χρησιμοποιούνται από όλους τους ακαδημαϊκούς placers για την εξέταση των αποτελεσμάτων.

Όσον αφορά τις μετρικές που εξετάστηκαν στη συγκεκριμένη εργασία, συμβαδίζουν και αυτές με τη λογική των περισσοτέρων ακαδημαϊκών αλγορίθμων για placement και legalization. Στα γραφήματα που παρουσιάζονται στη συνέχεια γίνεται εκτενής αναφορά στο Half Perimeter Wire Length, στο Displacement και στο Runtime του κάθε αλγορίθμου.

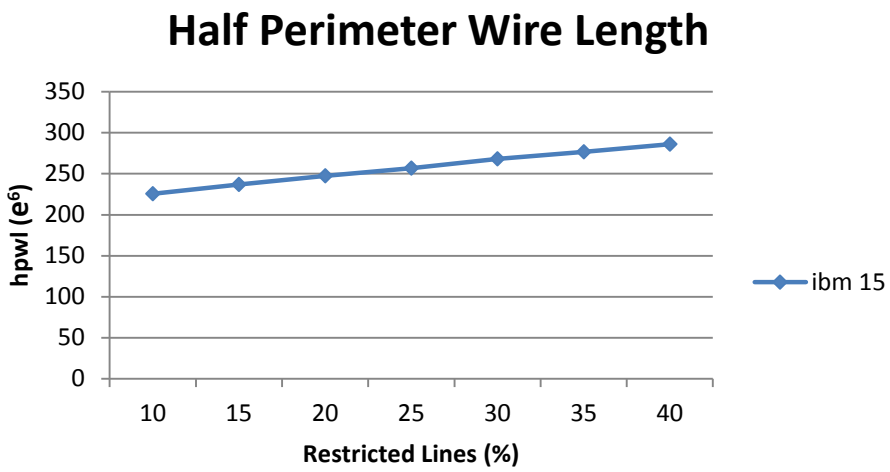
- Ως Half Perimeter Wire Length (hpwl) ορίζεται το συνολικό μήκος καλωδίου για τη σύνδεση των κελιών.
- Ως Displacement ορίζεται το άθροισμα του κόστος μετατόπισης όλων των κελιών που είναι η ευκλείδεια απόσταση μεταξύ αρχικής θέσης και προορισμού.
- Ως Runtime ορίζεται ο συνολικός χρόνος εκτέλεσης του αλγορίθμου.

Τα γραφήματα που ακολουθούν είναι από την εφαρμογή των αλγορίθμων στο ibm15 με 161187 κελιά, 303 γραμμές και 21174912 εμβαδό διαθέσιμης περιοχής τοποθέτησης των κελιών.

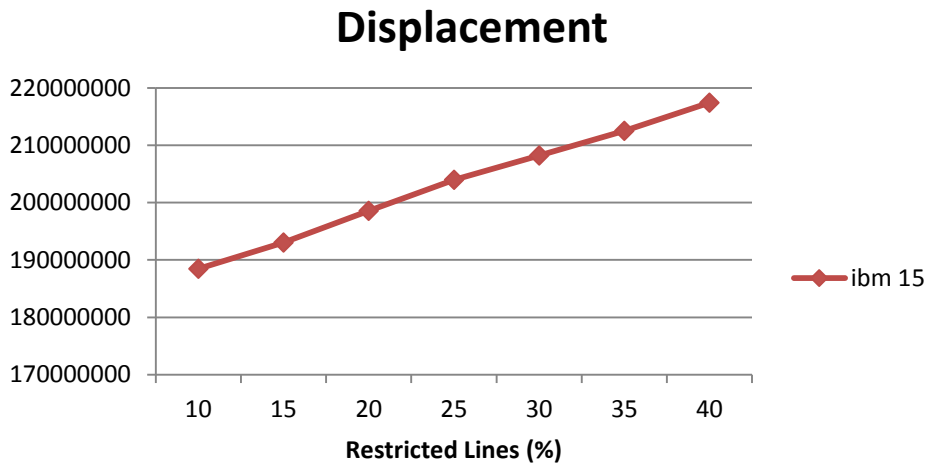
Πρώτα θα παρουσιάσουμε τα αποτελέσματα για την παραλλαγή Restricted Lines. Όπως αναφέρθηκε παραπάνω, ο αλγόριθμος εφαρμόστηκε για διάφορα μεγέθη αναζήτησης. Το βάθος εξέτασης γραμμών ήταν το 10%,15%,20%,25%,30%,35% ή 40% του συνολικού αριθμού γραμμών του κάθε κυκλώματος σε κάθε πείραμα.

Όπως ήταν αναμενόμενο τα αποτελέσματα παρουσίαζαν βελτίωση όσο συρρικνώναμε το προς εξέταση «παράθυρο» γραμμών. Αυτό οφείλεται στο γεγονός ότι ο global placer -που έχει προηγηθεί η εφαρμογή του- έχει τοποθετήσει τα κελιά στη βέλτιστη θέση όσον αφορά το wire length. Τρέχοντας τον αλγόριθμο σε μικρότερο βάθος αναζήτησης, τα κελιά τοποθετούνται πολύ κοντά στην βέλτιστη θέση τους και αποτέλεσμα έχει οι μετακινήσεις να είναι πολύ μικρές.

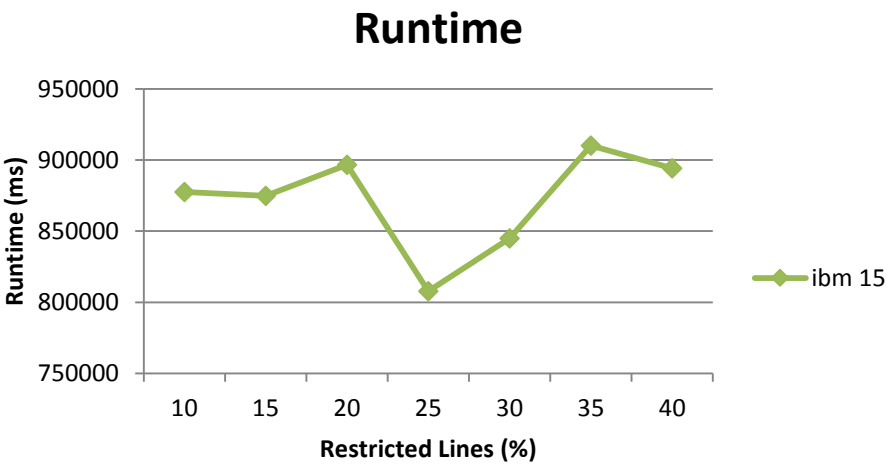
- Restricted Lines, Half Perimeter Wire Length



- Restricted Lines, Displacement

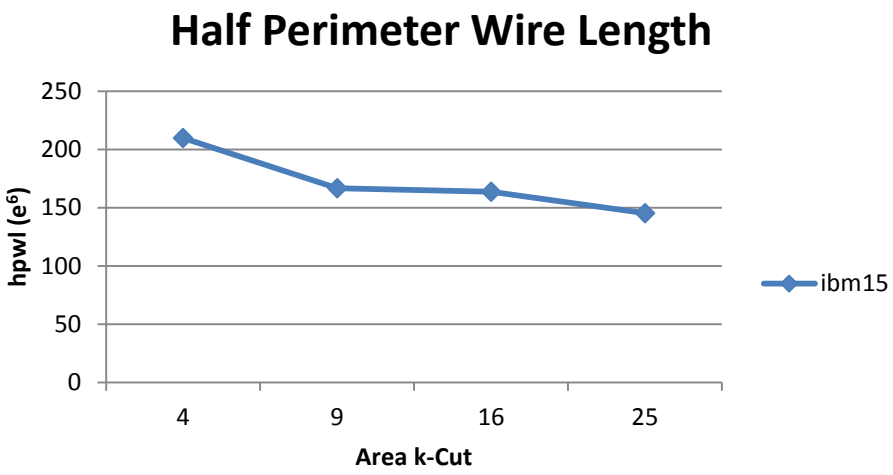


- Restricted Lines, Runtime



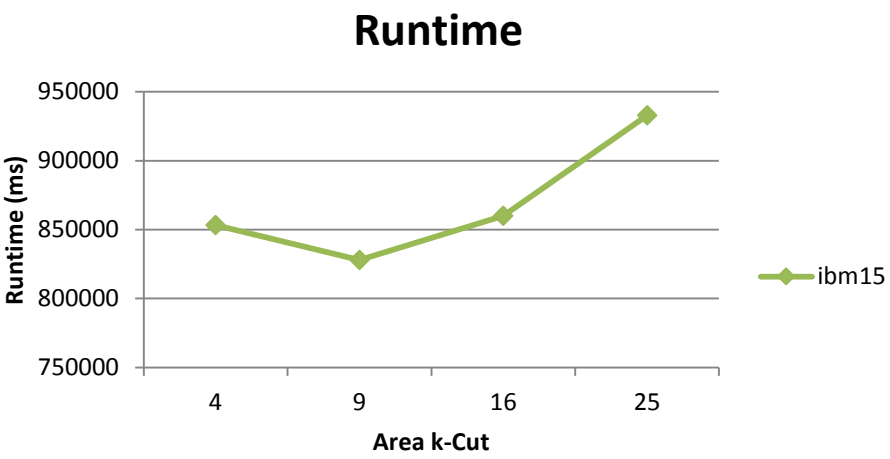
Στη συνέχεια παρουσιάζονται τα αποτελέσματα για τον Area k-Cut αλγόριθμο. Τη συγκεκριμένη παραλλαγή δοκιμάσαμε χωρίζοντας την αρχική περιοχή σε k = 4,9,16 και 25 τμήματα. Ακολουθώντας την ίδια λογική, παρατηρούμε ότι δημιουργώντας όσο το δυνατόν μικρότερα partitions ουσιαστικά περιορίζουμε το διαθέσιμο χώρο εντός του οποίου μπορούν να κινηθούν τα κελιά. Από τη στιγμή που περιορίζουμε την κίνηση σε μικρότερα τμήματα οι μετατοπίσεις και το μήκος καλωδίου έχουν αισθητά μικρότερες τιμές.

- Area k-Cut, Half Perimeter Wire Length



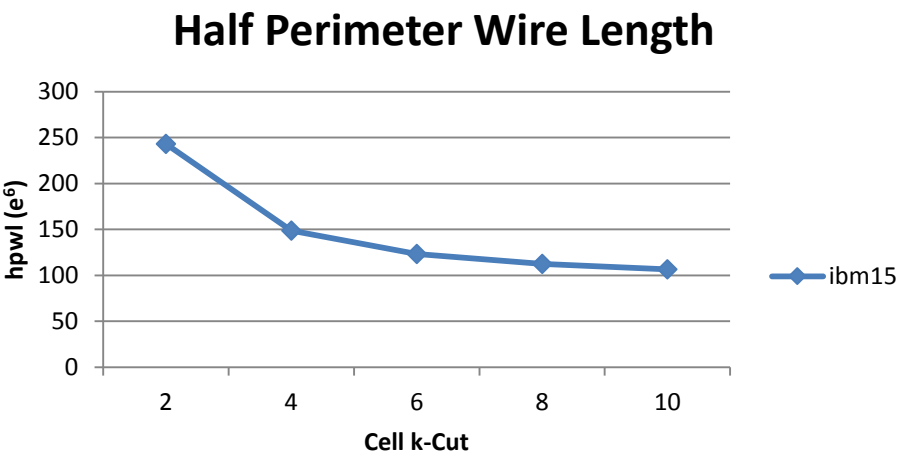
- Area k-Cut, Displacement

- Area k-Cut, Runtime

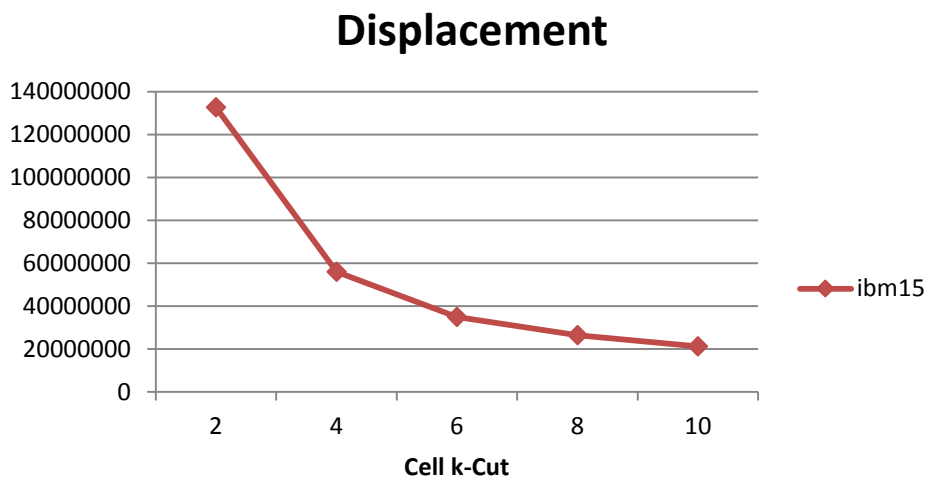


Όμοια συμπεριφορά παρατηρούμε και με τον Cell k-Cut αλγόριθμο. Έχοντας δοκιμάσει να ομαδοποιήσουμε τα κελιά σε k = 2,4,6,8 και 10 ομάδες, βασισμένος στην ίδια λογική με τις προηγούμενες παραλλαγές, παρατηρούμε ότι σε όσο περισσότερες ομάδες χωρίζουμε τα κελιά τόσο καλύτερα τα αποτελέσματα που παίρνουμε. Ο περιορισμός στο χώρο μετακίνησης φαίνεται να δρα αποτελεσματικά στην ελαχιστοποίηση του συνολικού wire length καθώς και του displacement.

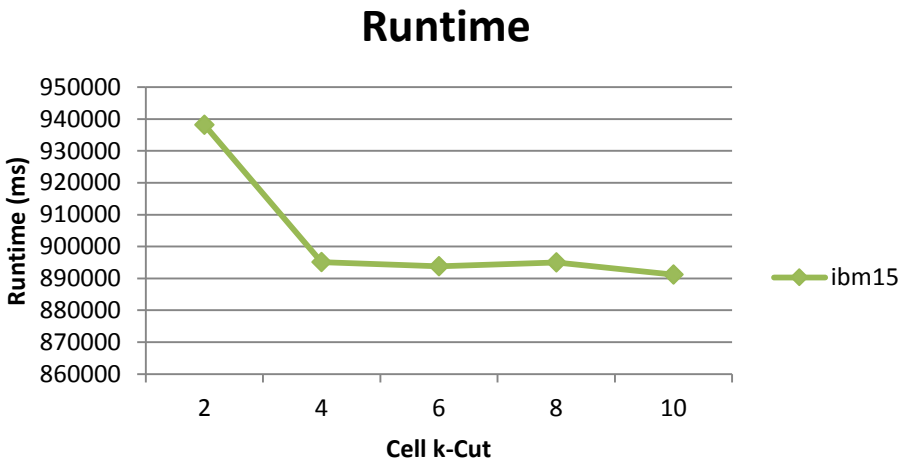
- Cell k-Cut, Half Perimeter Wire Length



- Cell k-Cut, Displacement

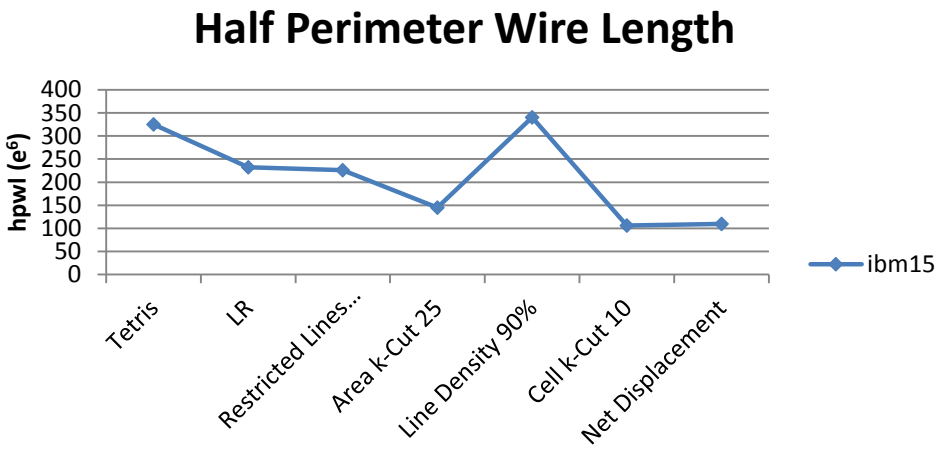


- Cell k-Cut, Runtime

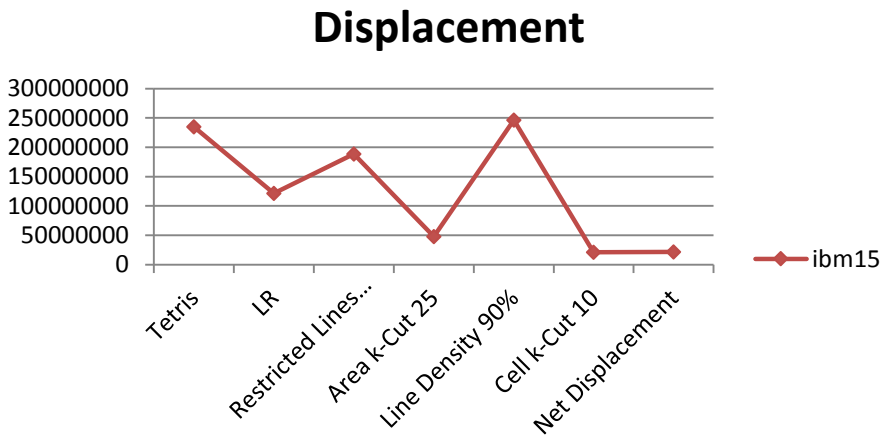


Στο σύνολό τους οι παραλλαγές έχουν καλύτερα αποτελέσματα από τον βασικό αλγόριθμο Tetris, ο οποίος και αποτελεί το βασικό μέτρο σύγκρισης, με εξαίρεση τον Line Density Tetris στην περίπτωση που καλούμαστε να κρατήσουμε την πυκνότητα της κάθε γραμμής στο 90%. Στην περίπτωσή του λόγω της σειράς με την οποία εξετάζουμε τα κελιά, υπάρχει το ενδεχόμενο κάποιο, μεγάλο σε μέγεθος κελί να παραβιάζει τους περιορισμούς που έχουμε θέσει, με αποτέλεσμα το συγκεκριμένο κελί ή κάποια από τα κελιά που το ακολουθούν να μετακινούνται σε μεγαλύτερες αποστάσεις.

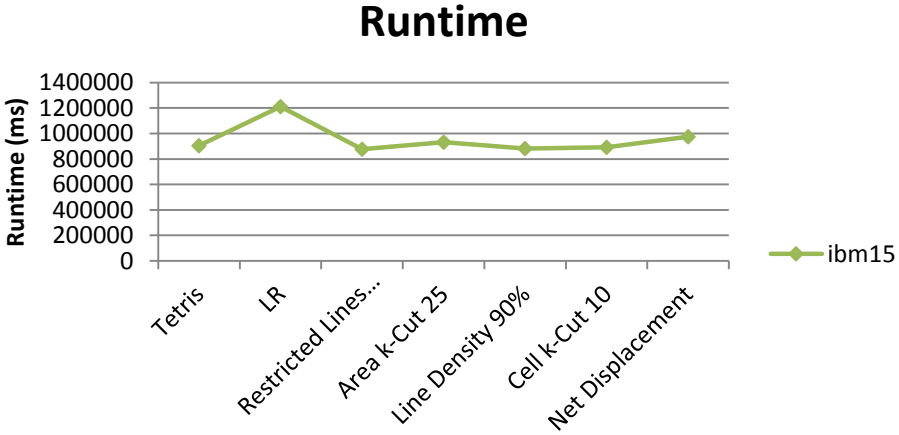
- Tetris/LR/Restricted Lines 10%/Area k-Cut 25/Line Density 90%/Cell k-Cut 10/Net Displacement, Half Perimeter Wire Length



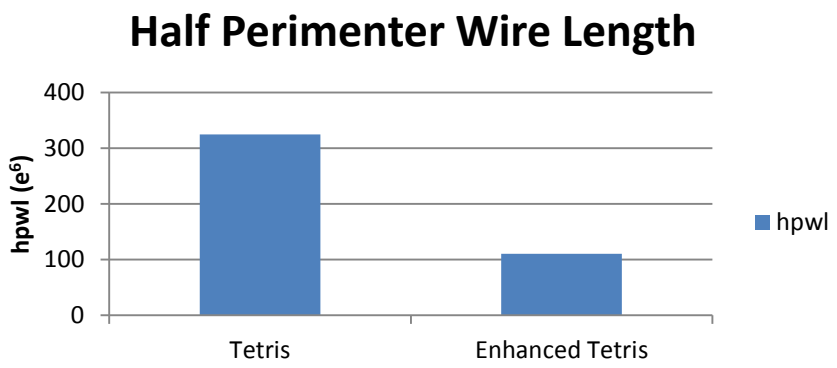
- Tetris/LR/Restricted Lines 10%/Area k-Cut 25/Line Density 90%/Cell k-Cut 10/Net Displacement, Displacement

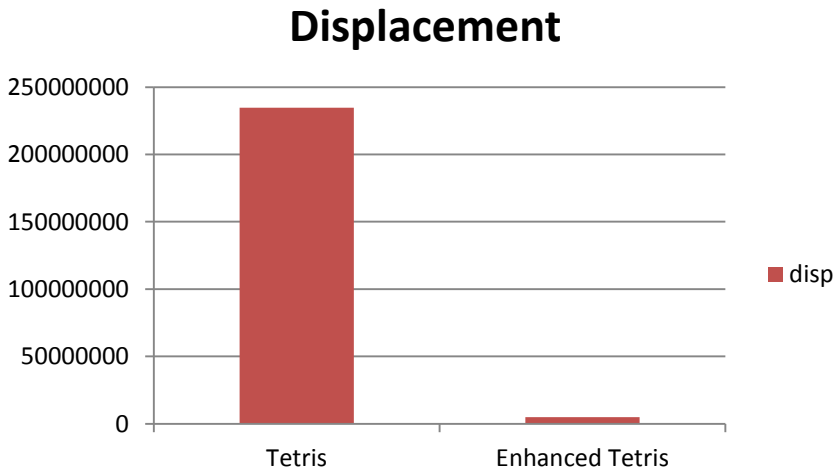


- Tetris/LR/Restricted Lines 10%/Area k-Cut 25/Line Density 90%/Cell k-Cut 10/Net Displacement, Runtime



- Enhanced Tetris





Κεφάλαιο 5: Μελλοντικές Επεκτάσεις

Στη συγκεκριμένα εργασία προαπαιτούμενο για να εφαρμόσουμε τις νέες υλοποιήσεις ήταν να έχει εκτελεστεί ο Gordian αλγόριθμος για να έχουμε ένα ορθά χωροθετημένο κύκλωμα. Στόχος μας για το μέλλον είναι η εφαρμογή των παραλλαγών αυτών έχοντας εφαρμόσει διαφορετικούς Global Placement αλγορίθμους πέρα από τον Gordian.

Επιπλέον περιορισμός που υπήρχε, ήταν ότι όλες οι σχεδιάσεις αποτελούνται από standard cells. Ως επέκταση, θα θέλαμε να τροποποιήσουμε καταλλήλως τους αλγορίθμους ώστε να μπορούνε να εφαρμοστούνε σε σχεδιάσεις που περιέχουν macro blocks ή/και obstacles.

Ταυτόχρονα, η παραλληλοποίηση του κώδικα, όπου είναι δυνατό αποτελεί ακόμη ένα μέρος της μελλοντικής μας δουλειάς, με σκοπό τη μεγαλύτερη βελτίωση του runtime που απαιτείται για την εκτέλεση του εκάστοτε αλγορίθμου.

Τέλος, ο συνδυασμός των heuristics που χρησιμοποιήσαμε σε συνδυασμό με άλλους legalization algorithms αποτελεί άλλη μία επέκταση της τρέχουσας εργασίας που θα θέλαμε να επιτύχουμε.

Παράρτημα

circuit	#cells	#rows	area
ibm01	12506	96	2141920
ibm02	19342	109	2757536
ibm03	22853	121	3375872
ibm04	27220	136	4303296
ibm05	28146	139	4471520
ibm06	32332	126	3695856
ibm07	45639	166	6422048
ibm08	51023	170	6663264
ibm09	53110	183	7755072
ibm10	68685	234	12664384
ibm11	70152	208	10010816
ibm12	70439	242	13603392
ibm13	83709	224	11592576
ibm14	147088	305	21534208
ibm15	161187	303	21174912
ibm16	182980	347	27836128
ibm17	184752	379	33185728
ibm18	210341	361	30189376

	Tetris			LR			Restricted Lines 10%		
circuit	hpwl	disp	time	hpwl	disp	time	hpwl	disp	time
ibm01	8.758	5954671	1552	6.063	2666750	2237	6.613	4871326	1563
ibm02	17.679	11308402	3750	12.534	5317104	5519	13.877	9142507	3762
ibm03	26.524	15900396	5189	17.429	6524377	7936	20.432	12827978	5270
ibm04	27.809	19213494	8123	21.935	10329712	11230	23.088	16018711	7646
ibm05	40.673	23427376	8078	30.111	10242034	11961	33.297	18536791	8104
ibm06	31.451	23507953	10993	20.086	11121502	17842	22.281	18748531	11289
ibm07	57.456	42702124	27293	36.627	18603046	40252	41.385	34359379	27792
ibm08	51.182	43915373	39537	36.649	21301011	57773	36.406	35876424	37598
ibm09	69.617	51426880	45259	45.203	25641338	65136	49.212	41949239	46488
ibm10	109.581	76839074	104524	74.744	35541111	146688	79.512	61873296	105569
ibm11	132.689	86859069	109175	82.563	40898153	158850	98.123	70618326	113450
ibm12	135.517	88391205	111554	96.372	42879361	160995	105.335	72623033	114928
ibm13	120.652	88692855	181895	85.96	46046100	233461	83.354	72326007	190477
ibm14	215.516	1.65E+08	742553	177.723	1.03E+08	1029550	150.363	1.36E+08	706606
ibm15	325.044	2.35E+08	905255	231.969	1.21E+08	1210374	225.75	1.88E+08	877516
ibm16	420.721	3.41E+08	1133088	276.613	1.7E+08	1636191	294.53	2.79E+08	1151365
ibm17	580.285	3.89E+08	1179373	374.629	1.74E+08	1657617	433.838	3.13E+08	1146045
ibm18	433.38	3.81E+08	1400903	279.939	1.88E+08	2169052	298.234	3.14E+08	1492656

	Restricted Lines 15%			Restricted Lines 20%			Restricted Lines 25%		
circuit	hpwl	disp	time	hpwl	disp	time	hpwl	disp	time
ibm01	6.814	4992015	1501	6.969	5109364	1549	7.15	5103086	1546
ibm02	14.355	9327251	3666	14.619	9590899	3765	15.063	9983439	3712
ibm03	21.134	13158060	5199	21.999	13643552	5285	22.652	14032843	5281
ibm04	23.621	16640648	7450	24.214	16895336	7873	24.63	17166685	7549
ibm05	34.001	19090337	7974	35.101	19791576	8375	36.075	20333638	9577
ibm06	23.04	19270621	11219	24.231	20043526	11184	25.324	20499025	12350
ibm07	43.472	35733376	30087	45.44	36769892	30798	47.119	37484665	36705
ibm08	38.384	36715414	40453	40.263	37797024	40907	41.79	38794089	53676
ibm09	51.143	42967075	45968	53.972	43944389	47400	56.612	45661656	37954
ibm10	83.194	63849773	104592	86.683	65392909	105853	90.175	67597728	108080
ibm11	103.493	73532168	113458	107.891	75683828	114575	112.73	77895882	115088
ibm12	107.583	74176364	113028	110.365	76274270	115724	113.363	77894573	113313
ibm13	88.109	73736400	175774	91.939	75839744	183915	96.732	78048491	190394
ibm14	158.214	1.39E+08	728777	164.771	1.42E+08	755065	170.59	1.45E+08	676264
ibm15	237.012	1.93E+08	874938	247.271	1.99E+08	896562	256.802	2.04E+08	807764
ibm16	313.726	2.87E+08	1165102	329.62	2.96E+08	1210468	342.99	3.03E+08	1167171
ibm17	451.215	3.23E+08	1170030	469.184	3.32E+08	1176608	486.436	3.41E+08	1093306
ibm18	318	3.22E+08	1493180	333.754	3.31E+08	1535842	351.156	3.39E+08	1485296

Restricted Lines 30%				Restricted Lines 35%			Restricted Lines 40%		
circuit	hpwl	disp	time	hpwl	disp	time	hpwl	disp	time
ibm01	7.328	5258016	1542	7.613	5445130	1558	7.827	5552285	1558
ibm02	15.496	10111854	3681	15.944	10433836	3860	16.304	10524427	3718
ibm03	23.317	14367839	5297	24.21	14786529	5273	24.778	15004187	5204
ibm04	25.159	17499446	8158	25.924	17811625	8148	25.842	18110187	7680
ibm05	37.133	21134792	8099	37.654	21509858	8536	38.411	21688952	8287
ibm06	26.538	20888927	12478	27.388	21569572	12256	28.591	22038760	12443
ibm07	48.932	38380590	37855	50.713	39525394	41791	51.919	39985813	27634
ibm08	43.685	39713537	41619	44.876	40572753	63547	45.893	41503404	41122
ibm09	58.598	46537709	46407	60.852	47551702	47814	62.454	48366233	47215
ibm10	93.674	69332982	106007	95.807	70419007	161220	98.605	71966925	106669
ibm11	116.227	79832858	96971	119.539	81465944	114260	122.227	83295820	102732
ibm12	116.398	79570352	111443	121.02	81860986	115210	124.312	83259386	115156
ibm13	101.205	79960756	183053	104.722	81333322	167028	106.74	83622788	191116
ibm14	176.02	1.47E+08	727378	181.413	1.51E+08	755478	189.551	1.54E+08	737110
ibm15	268.034	2.08E+08	844823	276.721	2.13E+08	910137	286.196	2.17E+08	894281
ibm16	356.341	3.11E+08	1079838	368.731	3.18E+08	1172741	381.028	3.24E+08	1154667
ibm17	504.045	3.5E+08	1161259	517.663	3.59E+08	1108717	529.955	3.66E+08	1103583
ibm18	363.844	3.48E+08	1443086	377.343	3.55E+08	1539199	388.921	3.63E+08	1493659

	Area K-Cut 4			Area K-Cut 9			Area K-Cut 16		
circuit	hpwl	disp	time	hpwl	disp	time	hpwl	disp	time
ibm01	5.277	2784458	6650	6.154	3185155	5493	5.785	2493761	6627
ibm02	9.385	3375369	8626	10.613	3735091	7331	10.094	3193088	8526
ibm03	16302	5962480	9673	14.19	4199054	8872	13.52	3586689	9506
ibm04	18.503	8802343	12294	15.339	4815572	10546	15.435	4407108	11094
ibm05	28.822	11010634	13973	25.167	6404939	11210	25.297	6363192	13115
ibm06	18.164	11121704	15767	15.897	7387875	16074	14.949	6601541	15629
ibm07	32.538	15722415	33219	27.35	11096362	31913	28.364	10344004	31865
ibm08	31.345	19759007	44725	26.332	12333658	43899	25.841	10961483	42441
ibm09	38.02	23232132	52980	31.895	14323169	50777	30.832	12859015	48539
ibm10	63.961	31858842	111446	61.062	22798196	111036	61.988	21459202	109603
ibm11	71.252	35410601	118426	62.552	25931446	121201	56.649	19612026	116274
ibm12	83.54	38243705	121221	72.756	25316352	120457	70.288	22047512	121732
ibm13	70.002	41093596	197678	62.806	28086503	194858	58.742	22628140	196140
ibm14	150.073	89649970	760057	129.021	61538921	758197	121.645	53938693	760946
ibm15	209.856	1.19E+08	853320	166.712	73452412	827881	163.565	64059659	859969
ibm16	247.06	1.62E+08	1222619	184.159	97401201	1195951	168.781	77761110	1138783
ibm17	335.626	1.58E+08	1199096	281.556	99626056	1161566	282.624	96296427	1249548
ibm18	265.094	1.81E+08	1575016	186.157	98705594	1607972	187.135	94608357	1630440

	Area K-Cut 25			Line Density 90%			Cell K-Cut 2		
circuit	hpwl	disp	time	hpwl	disp	time	hpwl	disp	time
ibm01	5.371	1586275	5417	8.898	6011352	1522	5.688	3007890	1561
ibm02	9.768	2677141	7363	17.417	11446021	3716	12.631	5936740	3743
ibm03	14.561	3184228	9924	25.865	16179908	5439	14.168	4788190	5336
ibm04	15.229	4099703	11868	27.486	20003815	8193	15.905	6348730	7825
ibm05	23.665	4814149	12551	38.664	23723418	8563	30.591	11966300	8105
ibm06	14.543	5295326	16165	30.231	23636187	11770	19.91	11981500	11151
ibm07	26.245	8400907	34343	57.141	43533636	32217	29.929	13018200	29393
ibm08	23.756	8391590	44645	52.293	45848949	26501	26.535	12938200	42329
ibm09	28.707	8440764	48610	68.867	53839306	69385	41.6	24890100	47341
ibm10	54.07	12316042	108648	110.512	81171008	102176	56.87	22447000	108856
ibm11	54.488	17344505	117172	126.576	88711515	108148	77.898	39080100	115040
ibm12	66.661	17288561	119260	141.058	90805615	104811	73.884	28079300	116989
ibm13	54.978	18839236	194430	123.441	93782457	183352	80.387	46294400	193315
ibm14	110.723	39632393	765007	242.908	1.87E+08	740315	123.81	59525100	765432
ibm15	145.131	48039761	932924	340.431	2.46E+08	882503	243.258	1.33E+08	938191
ibm16	154.325	62062750	1122368	416.503	3.51E+08	1082269	277.82	1.78E+08	1158682
ibm17	252.984	69164229	1189156	567.761	3.95E+08	1156606	299.68	1.14E+08	1196237
ibm18	161.832	68700374	1651041	439.439	4.04E+08	1495438	262.666	1.76E+08	1629717

	Cell K-Cut 4			Cell K-Cut 6			Cell K-Cut 8		
circuit	hpwl	disp	time	hpwl	disp	time	hpwl	disp	time
ibm01	4.553	1580570	1538	4.072	1144010	1508	3.815	960940	1526
ibm02	9.186	2559700	3697	8.277	1747150	3746	7.816	1372970	3705
ibm03	11.953	2910320	5215	11.314	2344590	5259	10.661	1800630	5296
ibm04	12.005	2323310	7504	11.079	1591320	7706	10.833	1401610	7507
ibm05	23.197	5205130	8041	21.346	3556210	8209	20.559	2880820	8057
ibm06	13.078	5275200	11183	11.353	3651520	11051	10.625	2940590	11119
ibm07	22.995	7081530	28151	19.486	4215130	26873	19.028	3694770	28339
ibm08	20.549	7079790	40693	17.851	4399020	40614	17.234	3752090	39973
ibm09	27.453	11024900	45354	23.076	7163130	46368	21.298	5537900	45541
ibm10	47.888	15002500	105666	43.296	10975700	104516	41.377	8993630	102510
ibm11	53.572	18489100	110027	46.079	12462000	109730	42.093	9256930	110948
ibm12	60.044	15891400	111528	54.095	11509700	110679	50.971	8669340	111998
ibm13	53.721	20867300	184801	45.386	13955800	181645	40.705	10627400	202969
ibm14	96.403	32558500	731498	85.411	21234300	728915	79.924	16532200	728581
ibm15	148.555	55996900	895104	123.322	34966800	893746	112.634	26549600	895041
ibm16	168.707	77871900	1180991	136.622	50963500	1182082	121.311	38338000	1181105
ibm17	241.17	68833100	1195590	214.888	47699800	1213547	200.518	36280300	1203710
ibm18	172.306	81504800	1595249	140.638	49583100	1605976	126.651	37669600	1592388

	Cell K-Cut 10			Wirelength Driven			Net Displacement		
circuit	hpwl	disp	time	hpwl	disp	time	hpwl	disp	time
ibm01	3.741	885914	1545	11.199	12724314	253205	3.68	1258300	1613
ibm02	7.597	1173060	3732	19.458	22294544	668946	7.455	2180550	3844
ibm03	10.631	1692940	5249	27.869	29242992	1036397	11.039	2259930	5400
ibm04	10.77	1332860	7666	35.298	39052687	1532911	11.482	2269230	7724
ibm05	20.114	2451170	8366	40.371	41383399	1852762	20.217	3936420	8306
ibm06	10.079	2574400	11995	35.685	43355399	2048327	10.256	2835500	11356
ibm07	18.371	3156210	28075	67.957	80063081	5207026	19.554	4543380	34164
ibm08	16.791	3151590	55527	70.162	90913018	7556111	17.568	6296170	45004
ibm09	20.628	4764950	44971	89.481	1.02E+08	9224490	22.747	6343660	50120
ibm10	39.442	7668010	104242	149.59	1.7E+08	23373911	41.234	9793420	122464
ibm11	40.054	7649540	111895	140.421	1.54E+08	21491876	40.273	8313340	136935
ibm12	49.534	7459640	113098	164.35	1.8E+08	49368837	54.161	11618300	135255
ibm13	38.383	8446660	185324	178.922	1.97E+08	67392172	44.608	11470100	217020
ibm14	77.012	13748700	733105	323.189	3.37E+09	118417971	86.219	15334400	802869
ibm15	106.559	21219800	891147	414.731	3.66E+09	122768828	109.501	21664500	975378
ibm16	111.985	30682800	1175578	451.083	5.63E+09	147312900	116.026	41987300	1298587
ibm17	192.07	29747300	1203523	668.522	5.7E+09	148739549	186.722	38363800	1328155
ibm18	117.603	28795200	1613269	519.864	7.21E+09	173176912	114.962	43677362	1753616

	Total HPWL Improvement (e6)			Runtime Overhead (ms)			Displacement Overhead
Circuit	Tetris	Enhanced Tetris	Improvement	Tetris	Enhanced Tetris	Improvement	Difference
ibm01	8.758	3.842	-56.10%	1552	262	16.90%	4.51%
ibm02	17.679	7.914	-55.20%	3750	16	0.40%	4.56%
ibm03	25.524	11.033	-58.40%	5189	1691	32.60%	3.69%
ibm04	27.809	13.121	-52.80%	8123	13	0.20%	4.11%
ibm05	40.673	19.546	-51.90%	8078	326	4.00%	4.00%
ibm06	31.451	10.947	-65.20%	10993	2771	25.20%	3.24%
ibm07	57.456	20.436	-64.40%	27293	35	0.10%	3.07%
ibm08	51.182	19.451	-62.00%	39537	255	0.60%	2.39%
ibm09	69.617	24.28	-65.10%	45259	937	2.00%	2.62%
ibm10	109.581	44.312	-59.60%	104524	625	0.60%	3.24%
ibm11	132.689	43.061	-67.50%	109175	154	0.10%	3.19%
ibm12	135.517	54.281	-59.90%	111554	1844	1.70%	3.27%
ibm13	120.652	46.995	-61.00%	181895	207	0.10%	2.42%
ibm14	215.516	91.623	-57.50%	742553	4240	0.60%	3.58%
ibm15	325.044	110.503	-66.00%	905255	538	0.10%	2.10%
ibm16	420.721	119.536	-71.60%	1133088	1796	0.20%	1.55%
ibm17	580.285	204.629	-64.70%	1179373	323	0.00%	2.59%
ibm18	433.38	137.383	-68.30%	1400903	1467	0.10%	2.34%

Βιβλιογραφία

[1] C. J. Alpert , D. P. Mehta , S. S. Sapatnekar - Handbook of Algorithms for Physical Design Automation

[2] A. Sarkar - VLSI Design and EDA Tools

[3] A. B. Kahng - VLSI Physical Design: From Graph Partitioning to Timing Closure

[4] S. K. Ioannidis, D. Ntioudis, C. Antoniadis, A. N. Dadaliaris, P. Tsompanopoulou, N. E. Evmorfopoulos, G. I. Stamoulis: “Optimization of an Integrated Circuit Placement Algorithm in a Parallel Environment”, CSCESM 2014

[5] A. N. Dadaliaris, G. Dimitriou, G. I. Stamoulis: “VDA-Place: Voltage-Drop-Aware Standard Cell Placement”, CSCESM 2014

[6] A. N. Dadaliaris, P. Oikonomou, N. Evmorfopoulos, G. Dimitriou, G. I. Stamoulis: “VDA Place+: Voltage-Drop-Aware Standard Cell Placer”, PACET 2015

[7] P. Oikonomou, T. Loukopoulos, A. N. Dadaliaris, M. Koziri, G. I. Stamoulis: “On Formulating and Tackling Integrated Circuit Placement as a Scheduling Problem”, PCI 2015

[8] A. N. Dadaliaris, E. Nerantzaki, M. Koziri, P. Oikonomou, T. Loukopoulos, G. I. Stamoulis: “Performance Evaluation of Tetris-based Legalization Heuristics”, PCI 2016

[9] A. N. Dadaliaris, E. Nerantzaki, P. Oikonomou, Y. Hatzaras, A. Troumpoulou, I. Arvanitakis, G. I. Stamoulis: “Enhanced Tetris Legalization”, SEEDA-CECNSM 2016

[10] P. Oikonomou, M. G. Koziri, A. N. Dadaliaris, Y. Hatzaras, E. Nerantzaki, G. I. Stamoulis: “Heuristics for Iterative Detailed Standard Cell Placement”, SEEDA-CECNSM 2016

[11] Kleinhaus, G. Sigl, F. Johannes, K. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization", IEEE Trans. on CAD, pp 356-365, 1991.

[12] G. Sigl, K. Doll and F.M. Johannes, "Analytical placement: A Linear or Quadratic Objective Function?", Proc. DAC, pp 427-432, 1991.

[13] J. M. Kleinhans, G. Sigl, F. M. Johannes, K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization".

[14] M. C. Kim, D. J. Lee, I. L. Markov, “SimPL: An Effective Placement Algorithm”.

[15] N. Viswanathan, C. Chu, "FastPlace: efficient analytical placement using cell shifting, iterative local refinement, and a hybrid net model".

[16] N. Viswanathan, M. Pan, C. Chu, "FastPlace3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control".

[17] T. C. Chen, Z. W. Jiang, T. C. Hsu, Y. W. Chang, "NTUplace3: An Analytical Placer for Large-Scale Mixed-Size Designs With Preplaced Blocks and Density Constraints".

[18] H. Eisenmann, F. M. Johannes, "Generic global placement and floorplanning".

[19] P. Spindler, U. Schlichtmann, F. M. Johannes, "Kraftwerk2 - A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model".

[20] T. Taghavi, X. Yang, B. Choi, "Dragon2005".

[21] C. Ababei, N. Selvakkumaran, K. Bazargan, G. Karypis, "Multi-objective circuit partitioning for cutsize and path-based delay minimization".

[22] T. F. Chan, J. Cong, M. Romesis, J. R. Shinnerl, K. , M. Xie, "mPL6: a robust multilevel mixed-size placement engine".

[23] D. Hill, "Method and system for high speed detailed placement of cells within an integrated circuit design".

[24] M. Cho, H. Ren, H. Xiang, R. Puri, "History-based VLSI legalization using network flow".

[25] U. Brenner, "VLSI legalization with minimum perturbation by iterative augmentation".

[26] U. Brenner, "BonnPlace Legalization: Minimizing Movement by Iterative Augmentation".

[27] U. Brenner, J. Vygen, "Legalizing a placement with minimum total movement".

[28] M. Sarrafzadeh, M. Wang, "NRG: global and detailed placement".

[29] N. Selvakkumaran, G. Karypis, “THETO: A Fast and High-Quality Partitioning Driven Global Placer”.

[30] P. Spindler, U. Schlichtmann, F. M. Johannes, "Abacus: fast legalization of standard cell circuits with minimal movement".

[31] C. C. Huang, C. H. Chiou, K. H. Tseng, "Detailed-Routing-Driven analytical standard-cell placement".

[32] T. Luo, D. Z. Pan, "DPlace2.0: A stable and efficient analytical placement based on diffusion".