ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Εποικοδομητικές Προσεγγίσεις Χωροθέτησης Στη Σχεδίαση VLSI

Κακάτσος Αντώνιος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Δαδαλιάρης Αντώνιος
Επίκουρος Καθηγητής

Λαμία 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Εποικοδομητικές Προσεγγίσεις Χωροθέτησης Στη Σχεδίαση VLSI

Κακάτσος Αντώνιος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Δαδαλιάρης Αντώνιος
Επίκουρος Καθηγητής

Λαμία 2022

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

Constructive Placement Approaches in VLSI Design

Kakatsos Antonios

FINAL THESIS

ADVISOR

Dadaliaris Antonios
Assistant Professor

Lamia 2022

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Χ

Ημερομηνία: 31/8/2022

Ο – Η Δηλ.

Κακάτσος Αντώνιος

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση
του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων
σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Η χωροθέτηση τυπικών κελιών (cells) είναι μέρος της φυσικής σχεδίασης ενός chip VLSI. Προκειμένου να επιτευχθεί υψηλή απόδοση, η περιοχή του chip και τα μήκη των καλωδίων που συνδέουν τα cells πρέπει να ελαχιστοποιηθούν. Στο βήμα της χωροθέτησης, ο στόχος είναι να τοποθετηθούν τα cells με τέτοιο τρόπο ώστε το συνολικό μήκος του καλωδίου να είναι όσο το δυνατόν μικρότερο, παράλληλα η χωροθέτηση πρέπει να γίνει με τέτοιο τρόπο ώστε να αποφευχθούν επικαλύψεις και υπερχειλίσεις των cells. Δεδομένου ότι αυτό το πρόβλημα είναι NP-hard, πρέπει να αναζητήσουμε ευοικοδομητικές προσεγγίσεις για την επίλυση του. Σε αυτή την εργασία θα εξετάσουμε κάποιους αλγορίθμους και τις παραλλαγές τους ώστε να το καταφέρουμε σε έναν καλό βαθμό. Θα χρησιμοποιηθούν τέσσερις διαφορετικοί αλγόριθμοι για τρία κυκλώματα.

ABSTRACT

Placement of standard cells is a part of physical VLSI chip design. In order to achieve high performance, area of the chip and lengths of wires connecting cells have to be minimized. In the placement step, the goal is to place cells in such a way that total wire-length is as short as possible, while the placement must be done in such a way that overlaps and overflows of cells are avoided. Since this problem is NP-hard, we need to seek constructive approaches to find a solution. In this paper we will look at some algorithms and their variations to achieve this to a good degree. Four different algorithms will be used for three circuits.

Table of Contents

Εισαγωγή

Η χωροθέτηση είναι ένα από τα σημαντικότερα στάδια της φυσικής σχεδίασης αφού επιδρά στην ποιότητα του τελικού αποτελέσματος του κυκλώματος. Σε μια τυπική ροή χωροθέτησης τα cells της σχεδίασης τοποθετούνται πάνω σε μια καθορισμένη ορθογώνια περιοχή, η οποία συγκροτεί και τον «πυρήνα» του κυκλώματος. Ταυτόχρονα, σκοπός της χωροθέτησης είναι να βελτιώσει την υπάρχουσα σχεδίαση με βάση κάποιες μετρικές, η πιο σημαντική από αυτές είναι το μήκος καλωδίου που χρησιμοποιείται (ημιπερίμετρος καλωδίου – half perimeter wirelength). Τις περισσότερες φορές, στο στάδιο της χωροθέτησης τοποθετούνται ή επανατοποθετούνται τα cells της σχεδίασης, και όχι άλλα δομικά στοιχεία των οποίων η θέση ορίστηκε από προηγούμενα στάδια της φυσικής σχεδίασης τα οποία είναι σταθερά κι αμετάκλητα.

Οι πλειοψηφία των αλγορίθμων χωροθέτησης έχουν σαν σκοπό τα προαναφερθέντα και λειτουργούν συχνά με διάφορες παραλλαγές πάνω σε μία ή και περισσότερες φάσεις της χωροθέτησης, οι οποίες είναι, η καθολική χωροθέτηση (global placement), η νομιμοποίηση (legalization) και η λεπτομερής χωροθέτηση (detailed placement). Κατά την πρώτη φάση σκοπός είναι το «σκόρπισμα» των cells στην περιοχή της σχεδίασης υλοποιώντας κάποιες τεχνικές τελειοποίησης, ενώ προκύπτουν κάποια σφάλματα τα οποία θα διαγραφθούν κατά τη φάση της νομιμοποίησης. Τέλος όταν συμβαίνει η λεπτομερής χωροθέτηση υλοποιούνται μικρές κινήσεις βελτίωσης ώστε τα αποτελέσματα των μετρικών τιμών που θα προκύψουν να είναι τα καλύτερα δυνατά.

Όλοι οι αλγόριθμοι χωροθέτησης προτού εκτελέσουν την βασική λειτουργία τους, πρέπει να περάσουν από μια προεργασία στην οποία «δημιουργείται» η εικόνα του πρωτότυπου κυκλώματος, όπου θα γίνουν μερικές αλλαγές. Η εικόνα ορίζεται συχνά από το πρότυπο βιβλιοθήκης (bookshelf format) σύμφωνα με το οποίο μια σχεδίαση παρομοιάζεται με μια βιβλιοθήκη και τα δομικά της στοιχεία με βιβλία τα οποία τοποθετούνται σε αυτή. Ωστόσο η απόδοση των αρχείων με τα δεδομένα του κυκλώματος σε αυτή την μορφή είναι στην κρίση του προγραμματιστή, αφού δεν υπάρχουν πολλά εργαλεία τα οποία να υλοποιούν αυτόματα αυτή τη λειτουργία. Οπότε, πρέπει να αναπτυχθούν σε κώδικα αυτές οι λειτουργίες αλλά και όποιες άλλες κριθούν απαραίτητες για να ολοκληρωθεί ο επιθυμητός αλγόριθμος.

Τα εργαλεία σχεδίασης με τη βοήθεια υπολογιστή καθιστούν πλέον δυνατή την αυτοματοποίηση ολόκληρης της διαδικασίας διάταξης που ακολουθεί τη φάση σχεδιασμού του κυκλώματος στη σχεδίαση VLSI. Αυτό κατέστη δυνατό κυρίως με τη χρήση της διάταξης πυλών και των σχεδιαστικών στυλ των standard cells, σε συνδυασμό με αποτελεσματικά πακέτα λογισμικού για αυτόματη χωροθέτηση και δρομολόγηση. Τα standard cells είναι λογικές μονάδες με προσχεδιασμένη εσωτερική διάταξη. Έχουν σταθερό ύψος αλλά διαφορετικό πλάτος, ανάλογα με τη λειτουργικότητα των μονάδων. Είναι

διατεταγμένα σε σειρές, με κανάλια δρομολόγησης ή κενά μεταξύ των σειρών που προορίζονται για τη διάταξη των διασυνδέσεων μεταξύ των στοιχείων του chip. Τα standard cells σχεδιάζονται συνήθως έτσι ώστε οι διασυνδέσεις ισχύος και γείωσης να διατρέχουν οριζόντια το επάνω και το κάτω μέρος των cells.

Βιβλιογραφική Επισκόπηση

Η χωροθέτηση αποτελεί θέμα διαφόρων ακαδημαϊκών ερευνών, καθώς είναι ένα σημαντικό κομμάτι της φυσικής σχεδίασης ολοκληρωμένων κυκλωμάτων. Το πρόβλημα χωροθέτησης cells VLSI είναι γνωστό ότι είναι NP-hard. Σε βιβλιογραφία υπάρχει ένα ευρύ ρεπερτόριο ευριστικών αλγορίθμων για την αποτελεσματική διάταξη των λογικών cells σε ένα chip VLSI. Σε αυτή την έρευνα παρουσιάζεται μια βασική επισκόπηση των αλγορίθμων χωροθέτησης που χρησιμοποιούνται για την αυτόματη χωροθέτηση των cells η οποία δίνει μια βελτιστοποιημένη σχεδίαση από την άποψη της περιοχής chip, της ταχύτητας και του κόστους. Συγκεκριμένα, θα δοθεί έμφαση στην πρόσφατη τάση χωροθέτησης με μήκος καλωδίου που περιλαμβάνει τους διαφορετικούς τρόπους εκτίμησης του μήκους καλωδίου της ημιπεριμέτρου, ολοκληρώνοντας έτσι έναν αποτελεσματικό τρόπο τακτοποίησης των λογικών cells στο chip VLSI.

Το στάδιο της καθολικής χωροθέτησης (global placement) είναι το αρχικό στην διαδικασία της χωροθέτησης, ο στόχος της είναι να τοποθετηθούν μαζικά τα δομικά στοιχεία της σχεδίασης επάνω στην προκαθορισμένη περιοχή του πυρήνα της ενώ ταυτόχρονα γίνονται ορισμένες κινήσεις ώστε να βελτιωθεί το τελικό αποτέλεσμα. Μερικοί γνωστοί αλγόριθμοι καθολικής χωροθέτησης αποτελούν πρότυπα για πολλούς άλλους καθώς και για ευριστικές μεθόδους. Κάποια παραδείγματα τέτοιων αλγορίθμων είναι ο Gordian [6], ο SimPL [7], ο FastPlace [8], ο Polar [9] και ο ePlace [10]. Επόμενο στάδιο είναι η νομιμοποίηση (legalization) στην οποία, όσα σχεδιαστικά λάθη δημιουργήθηκαν στο στάδιο της καθολικής χωροθέτησης διαγράφονται, με εφαρμογή διαφόρων κινήσεων των δομικών στοιχείων Η εμφάνιση υπερχειλίσεων, δηλαδή η προεξοχή των cells εκτός των ορίων της σχεδίασης και αλληλοεπικαλύψεων δηλαδή η τοποθέτηση ενός cell πάνω σε ένα άλλο είναι σημαντικά λάθη που μπορεί να εμφανιστούν. Παραδείγματα κλασσικών αλγορίθμων νομιμοποίησης είναι οι Tetris [11] και Abacus [12]. Η λεπτομερής χωροθέτηση (detailed placement) είναι το τελευταίο στάδιο, σκοπός της είναι να βελτιώσει κι άλλο το αποτέλεσμα εφαρμόζοντας διάφορες μικρές κινήσεις.

Στο πρώτο κεφάλαιο γίνεται μια γενική αναφορά στο κομμάτι της φυσικής σχεδίασης και τα στάδια που χρειάζονται για να επιτευχθεί, τα οποία είναι το partioning, το floorplanning, το placement, το routing και το compaction [5]. Το στάδιο που βεβαίως είναι πιο σημαντικό για την έρευνα είναι το κομμάτι του placement δηλαδή της χωροθέτησης, αλλά γίνεται μια γενική ανάλυση και στα υπόλοιπα στάδια.

Στο δεύτερο κεφάλαιο παραθέτεται ένα εργαλείο [2] που δημιουργήθηκε από έναν φοιτητή του πανεπιστημίου Θεσσαλίας. Γίνεται αναφορά σε αυτό το εργαλείο γιατί είναι μια πολύ καλοδουλεμένη μελέτη πάνω στο πρόβλημα της χωροθέτησης.

Στο τρίτο κεφάλαιο παρουσιάζονται οι αλγόριθμοι που χρησιμοποιήθηκαν για αυτήν την έρευνα, τα αποτελέσματα των οποίων βρίσκονται στο τέταρτο και τελευταίο κεφάλαιο της εργασίας.

ΚΕΦΑΛΑΙΟ 1: ΧΩΡΟΘΕΤΗΣΗ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ

Φυσική σχεδίαση

Ορισμός

Η φυσική σχεδίαση VLSI είναι ουσιαστικά η μελέτη των αλγορίθμων και των δομών δεδομένων που σχετίζονται με την διαδικασία της φυσικής σχεδίασης. Ο στόχος είναι να βρεθούν οι βέλτιστες καταχωρήσεις των cells μέσα σε ένα πεδίο (ένα chip για παράδειγμα) και η κατάλληλη διασύνδεση τους ώστε να επιτευχθεί η λειτουργικότητα που επιθυμούμε.

Το κομμάτι της φυσικής σχεδίασης τις περισσότερες φορές υποδιαιρείται σε αλλά δευτερεύοντα, τα οποία περιλαμβάνουν τον σχεδιασμό, την επαλήθευση και την επικύρωση της διάταξης του κυκλώματος. Αυτά στη συνέχεια υλοποιούνται μέσω της στρωμάτωσης κατάλληλων υλικών ώστε το κύκλωμα να λειτουργεί εύρυθμα.

Κύκλος φυσικής σχεδίασης

Η είσοδος στον κύκλο της φυσικής σχεδίασης είναι ένα διάγραμμα κυκλώματος και η έξοδος είναι η διάταξη του κυκλώματος. Αυτό επιτυγχάνεται σε διάφορα στάδια όπως:

- Partioning.
- Floorplanning.
- Placement.
- Routing.
- Compaction.

1. Partioning

Ένα chip μπορεί να περιέχει εκατομμύρια transistors. Συνεπώς, δεν γίνεται να διαχειριστεί η διάταξη ολόκληρου του κυκλώματος λόγω περιορισμένου χώρου μνήμης καθώς και της περιορισμένης υπολογιστικής ισχύος. Επομένως είναι φυσιολογικό να γίνει ο διαχωρισμός σε μπλοκ (πχ υποκυκλώματα). Η διαδικασία του διαχωρισμού υπολογίζει αρκετούς παράγοντες όπως το μέγεθος των μπλοκ, τον αριθμό τους και τον αριθμό των διασυνδέσεων μεταξύ τους. Το αποτέλεσμα του διαχωρισμού είναι ένα σετ από μπλοκ και οι διασυνδέσεις που απαιτούνται μεταξύ τους. Οι διασυνδέσεις που απαιτούνται ονομάζονται netlists. Σε μεγάλα κυκλώματα η διαδικασία του διαχωρισμού γίνεται ιεραρχικά.

2. Floorplanning και Placement

Αυτό το βήμα ασχολείται με την επιλογή καλών εναλλακτικών διατάξεων για κάθε μπλοκ, αλλά και για ολόκληρο το chip. Η θέση του κάθε μπλοκ μπορεί να υπολογιστεί μετά τον διαχωρισμό και βασίζεται στον αριθμό και τον τύπο των περιεχομένων του. Επιπλέον, η περιοχή διασύνδεσης που χρειάζεται μέσα στο μπλοκ πρέπει να ληφθεί υπόψιν. Το σχήμα παραλληλογράμμου του μπλοκ, το οποίο καθορίζεται από την αναλογία διαστάσεων, μπορεί να ποικίλει εντός μιας προκαθορισμένης εμβέλειας. Το βήμα του Floorplanning είναι κρίσιμο αφού εκεί μπαίνουν τα θεμέλια για ένα καλό διάγραμμα του κυκλώματος, ωστόσο είναι υπολογιστικά δύσκολο και συνήθως γίνεται χειροκίνητα. Αυτό μερικές φορές είναι απαραίτητο να συμβεί καθώς σημαντικά συστατικά του ολοκληρωμένου κυκλώματος πρέπει να τοποθετηθούν με βάση την ροή σήματος του chip.

Κατά την διάρκεια της χωροθέτησης, τα μπλοκ έχουν τοποθετηθεί πάνω στο chip. Ο στόχος της χωροθέτησης είναι να βρεθεί μια ελάχιστη διάταξη για τα μπλοκ, που να επιτρέπει την ολοκλήρωση των διασυνδέσεων αναμεσά τους, ενώ καλύπτονται οι περιορισμοί απόδοσης. Με αυτά τα δεδομένα, θέλουμε να αποφύγουμε μια χωροθέτηση που είναι δρομολογήσιμη, αλλά να μην επιτρέπει σε ορισμένα nets να συναντούν τους στόχους τους σχετικά με τον χρόνο. Η χωροθέτηση γίνεται τυπικά σε δυο στάδια. Στο πρώτο στάδιο, γίνεται μια αρχική χωροθέτηση. Στο δεύτερο στάδιο αξιολογείται η αρχική χωροθέτηση και γίνονται επαναληπτικά βελτιώσεις μέχρι να βρεθεί ένα ελάχιστο μήκος καλωδίου ή η βέλτιστη απόδοση μέσα στα πλαίσια της σχεδίασης. Επίσης, πρέπει να υπάρχει λίγος χώρος ανάμεσα στα μπλοκ ώστε να επιτρέπονται η διασυνδέσεις μεταξύ τους.

Η σημασία της χωροθέτησης δεν θα είναι εμφανής μέχρι την ολοκλήρωση του σταδίου της δρομολόγησης. Η χωροθέτηση μπορεί να οδηγήσει σε μη δρομολογήσιμη σχεδίαση (πχ λόγω του χώρου που παραθέτεται). Σε αυτήν την περίπτωση, χρειάζεται να γίνει κι άλλη επανάληψη. Για να μειωθεί ο αριθμός επαναλήψεων του αλγόριθμου χωροθέτησης, χρησιμοποιείται ένας εκτιμώμενος χώρος δρομολόγησης κατά την διάρκεια του σταδίου της χωροθέτησης. Η καλή απόδοση του κυκλώματος και της δρομολόγησης, βασίζεται πάρα πολύ σε έναν καλό αλγόριθμο χωροθέτησης. Αυτό οφείλεται στο γεγονός ότι μόλις βρεθεί η θέση του κάθε μπλοκ, πολύ λίγα μπορούν να γίνουν ώστε να βελτιωθεί η δρομολόγηση και η συνολική απόδοση του κυκλώματος.

3. Routing

Ο σκοπός του σταδίου δρομολόγησης είναι να ολοκληρωθούν οι διασυνδέσεις μεταξύ των μπλοκ σύμφωνα με την καθορισμένη netlist. Αρχικά, ο χώρος που δεν είναι κατειλημμένος από μπλοκ (καλείται χώρος δρομολόγησης) διαχωρίζεται σε περιοχές σε σχήμα παραλληλογράμμου που λέγονται κανάλια και switchboxes. Περιλαμβάνεται ο χώρος μεταξύ των μπλοκ όπως και ο χώρος πάνω από τα μπλοκ. Ο στόχος λοιπόν, του δρομολογητή είναι να ολοκληρωθούν όλες οι διασυνδέσεις του κυκλώματος χρησιμοποιώντας το ελάχιστο μήκος καλωδίου και μόνο τα κανάλια και τα switchboxes. Αυτό γίνεται συνήθως σε δυο στάδια, γνωστά και ως, Global Routing και Detailed Routing. Στο πρώτο από αυτά, οι συνδέσεις ολοκληρώνονται μεταξύ των κατάλληλων μπλοκ του κυκλώματος, χωρίς να λάβουμε υπ' όψη τις ακριβείς γεωμετρικές λεπτομέρειες του κάθε καλωδίου και του κάθε pin. Κάθε καλώδιο βρίσκει μια λίστα με κανάλια και switchboxes τα οποία πρόκειται να χρησιμοποιηθούν ως διάδρομος για αυτό το καλώδιο. Δηλαδή εξακριβώνει τις διαφορετικές περιοχές στον χώρο δρομολόγησης μέσω των οποίων πρέπει να δρομολογηθεί ένα καλώδιο. Ακολουθείται από λεπτομερή δρομολόγηση που ολοκληρώνει τις συνδέσεις σημείο προς σημείο, μεταξύ των μπλοκ και των pins. Έπειτα μετατρέπεται σε ακριβή δρομολόγηση όταν προσδιοριστούν οι γεωμετρικές πληροφορίες, όπως η τοποθεσία, η απόσταση καλωδίων και οι διατάξεις τους. Στο detailed routing περιλαμβάνεται η δρομολόγηση καναλιών και switchboxes και γίνεται για κάθε κανάλι και κάθε switchbox. Εφόσον σχεδόν όλα τα προβλήματα της δρομολόγησης είναι υπολογιστικά δύσκολα, οι ερευνητές έχουν αφοσιωθεί στους ευριστικούς αλγορίθμους. Ως αποτέλεσμα, η πειραματική αξιολόγηση έχει γίνει αναπόσπαστο κομμάτι όλων των αλγορίθμων και αρκετά σημεία αναφοράς έχουν τυποποιηθεί. Λόγω της φύσης της δρομολόγησης, σε πολλές περιπτώσεις δεν μπορεί να εγγυηθεί η ολοκληρωμένη δρομολόγηση όλων των συνδέσεων.

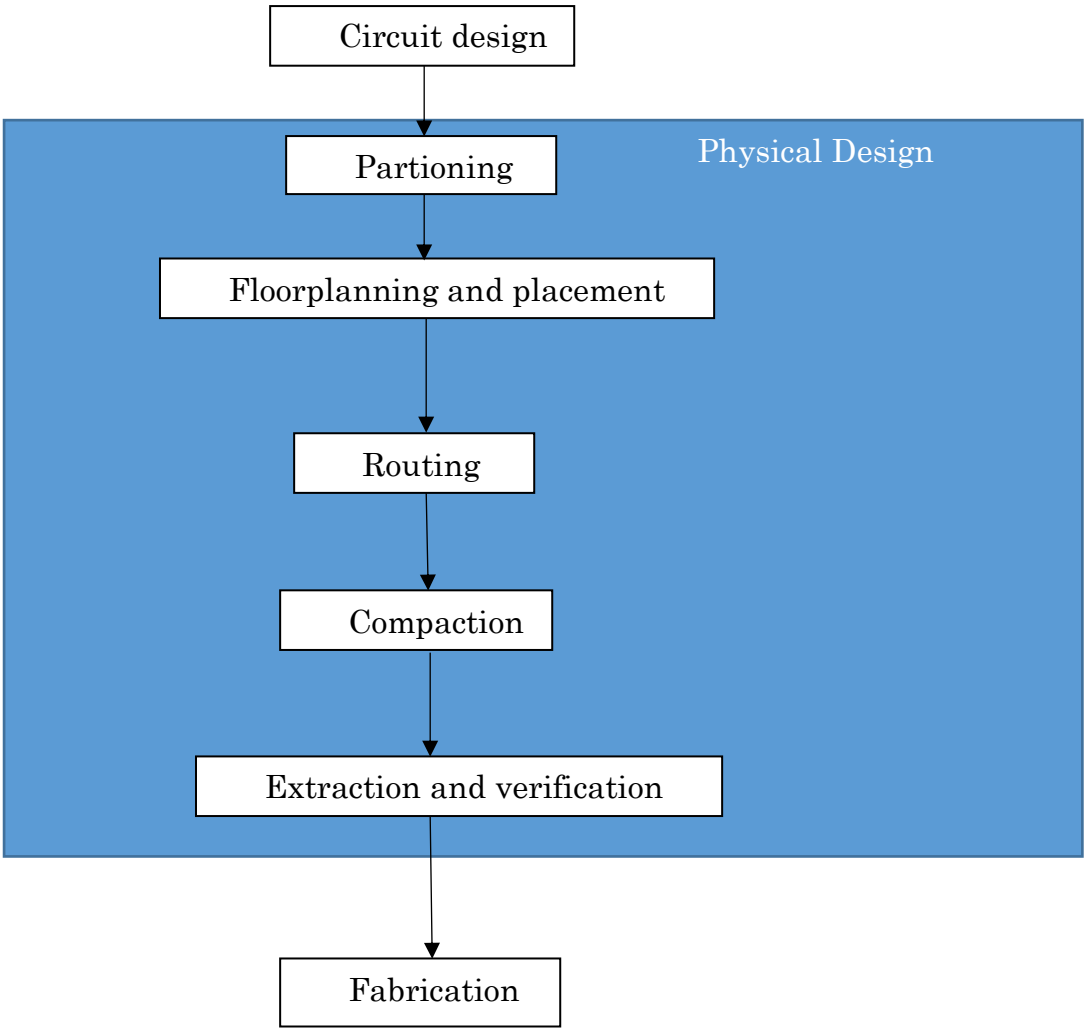
4. Compaction

Η συμπύκνωση είναι απλά η διαδικασία συμπίεσης της διάταξης σε κατευθύνσεις ώστε να μειωθεί η συνολική περιοχή του chip, κάνοντας το chip μικρότερο, τα μήκη καλωδίων μειώνονται με αποτέλεσμα να μειώνεται η καθυστέρηση σήματος μεταξύ των περιεχομένων του κυκλώματος. Συνεπώς, μια μικρότερη περιοχή μπορεί να σημαίνει ότι περισσότερα chips μπορούν να παραχθούν σε ένα δισκίο, με αποτέλεσμα να μειώνεται το κόστος παραγωγής. Κατά την συμπύκνωση πρέπει να βεβαιωθούμε ότι δεν έχουν παραβιαστεί κανόνες σχετικά με την σχεδίαση και την κατασκευή κατά την διάρκεια της διαδικασίας.

5. Extraction και Verification

Ο DRC (Design Rule Checking) είναι μια διαδικασία που επαληθεύει ότι όλα τα γεωμετρικά μοτίβα είναι σύμφωνα με τους κανόνες σχεδίασης που έχουν τεθεί από την διαδικασία κατασκευής. Για παράδειγμα ένας τυπικός κανόνας σχεδίασης είναι ο κανόνας διαχωρισμού του καλωδίου, όπου η διαδικασία σχεδίασης απαιτεί έναν συγκεκριμένο διαχωρισμό μεταξύ δυο γειτονικών καλωδίων. Ο DRC πρέπει να ελέγξει τα εκατομμύρια των καλωδίων που υπάρχουν στο chip. Αφού ελέγξει την διάταξη για παραβιάσεις κανόνων της σχεδίασης και τους διαγράψει, η λειτουργικότητα της διάταξης επαληθεύεται από την Εξαγωγή Σχεδίασης. Αυτή είναι μια αντιστροφή μηχανική διαδικασία που δημιουργεί την παρουσίαση του κυκλώματος μέσω της διάταξης. Η περιγραφή που εξάγεται συγκρίνεται με την περιγραφή του κυκλώματος για να επαληθεύσει την ορθότητα του. Η διαδικασία αυτή ονομάζεται Layout Versus Schematics (LVS) επαλήθευση. Οι γεωμετρικές πληροφορίες εξάγονται ώστε να υπολογιστεί η αντίσταση και η χωρητικότητα, αυτό επιτρέπει να υπολογιστεί με ακρίβεια ο χρονισμός κάθε στοιχείου συμπεριλαμβανομένης της διασύνδεσης, η διαδικασία αυτή ονομάζεται επαλήθευση απόδοσης. Οι εξαγόμενες πληροφορίες χρησιμοποιούνται και για τον έλεγχο αξιοπιστίας των πτυχών της διάταξης, η διαδικασία αυτή ονομάζεται επαλήθευση αξιοπιστίας.

Η φυσική σχεδίαση, όπως και η σχεδίαση VLSI, είναι επαναληπτική εκ' φύσεως και πολλά βήματα όπως το global routing και η δρομολόγηση καναλιών επαναλαμβάνονται αρκετές φορές ώστε να επιτευχθεί η βέλτιστη διάταξη. Επιπρόσθετα, η ποιότητα των αποτελεσμάτων που λαμβάνουμε σε ένα βήμα, βασίζεται στην ποιότητα της λύσης που πήραμε σε προηγούμενα βήματα. Για παράδειγμα μιας κακής ποιότητας χωροθέτηση δεν μπορεί να επιλυθεί από μιας καλής ποιότητας δρομολόγηση. Ως αποτέλεσμα, τα βήματα που εκτελέστηκαν νωρίτερα έχουν μεγαλύτερη επιρροή στην συνολική ποιότητα της λύσης. Επομένως τα προβλήματα της χωροθέτησης, του διαχωρισμού και του floorplanning, παίζουν σημαντικότερο ρολό στον καθορισμό της περιοχής και της απόδοσης του chip, συγκριτικά με την δρομολόγηση και την συμπύκνωση.



Εικόνα 1. Κύκλος φυσικής σχεδίασης

Χωροθέτηση

Η χωροθέτηση είναι ένα σημαντικό βήμα στον αυτοματισμό της ηλεκτρονικής σχεδίασης, αποτελεί το τμήμα της ροής της φυσικής σχεδίασης που αναθέτει ακριβείς θέσεις για διάφορα «συστατικά» του κυκλώματος εντός της περιοχής chip. Μια υποδεέστερη ανάθεση χωροθέτησης όχι μόνο επηρεάζει την απόδοση του chip, αλλά μπορεί επίσης να το καταστήσει μη κατασκευάσιμο με αποτέλεσμα να παραχθεί ένα υπερβολικό μήκος καλωδίου, το οποίο είναι πέρα από τους διαθέσιμους πόρους δρομολόγησης. Επομένως, ένας χωροθέτης (placer) πρέπει να εκτελέσει την ανάθεση βελτιστοποιώντας ταυτόχρονα κάποιους στόχους για να «βεβαιωθεί» ότι ένα κύκλωμα ανταποκρίνεται στις απαιτήσεις της απόδοσης του. Μαζί, τα βήματα χωροθέτησης και δρομολόγησης του σχεδιασμού IC (Integrated Circuit), που περιλαμβάνει την δημιουργία και την διασύνδεση διαφόρων ηλεκτρονικών υλικών πάνω σε έναν ημιαγωγό, είναι γνωστά ως place and route.

Ένας placer παίρνει μια δοσμένη netlist συνθετικού κυκλώματος μαζί με μια βιβλιοθήκη τεχνολογίας και παράγει μια έγκυρη διάταξη χωροθέτησης. Σύμφωνα με τους προηγούμενους στόχους η διάταξη έχει βελτιστοποιηθεί και είναι έτοιμη για αλλαγή μεγέθους των cells και αποθήκευση στην προσωρινή μνήμη, αυτό το βήμα είναι απαραίτητο για την ικανοποίηση του χρονισμού και της ακεραιότητας του σήματος. Σειρά έχουν η σύνθεση clock-tree και η δρομολόγηση (routing), ολοκληρώνοντας τη διαδικασία φυσικής σχεδίασης. Σε αρκετές περιπτώσεις, μέρη ή ολόκληρη η ροή φυσικής σχεδίασης επαναλαμβάνονται πολλές φορές μέχρι την τελειοποίηση της σχεδίασης.

Στην περίπτωση των ASIC (Application Specific Integrated Circuit) δηλαδή ολοκληρωμένων κυκλωμάτων για συγκεκριμένες εφαρμογές, το βασικό διάγραμμα του chip περιλαμβάνει έναν αριθμό σειρών σταθερού ύψους, είτε με λίγο ή καθόλου χώρο μεταξύ τους. Κάθε σειρά συγκροτείται από έναν αριθμό θέσεων που μπορούν να καλυφθούν από τα στοιχεία του κυκλώματος. Ένας ελεύθερος χώρος είναι ένας χώρος που δεν έχει καταληφθεί από κανένα στοιχείο. Τα στοιχεία του κυκλώματος είναι είτε τυπικά (standard) cells, μπλοκ μακροεντολών ή I/O pads. Τα standard cells έχουν σταθερό ύψος ίσο με το ύψος μιας σειράς, αλλά έχουν μεταβλητό πλάτος. Από την άλλη πλευρά, τα μπλοκ είναι συνήθως μεγαλύτερα από τα cells και έχουν μεταβλητά ύψη που μπορούν να τεντωθούν σε πολλές σειρές. Ορισμένα μπλοκ μπορεί να έχουν προκαθορισμένες θέσεις, από μια προηγούμενη διαδικασία floorplanning, η οποία περιορίζει την εργασία του placer στην εκχώρηση θέσεων μόνο για τα cells. Στην περίπτωση αυτή, τα μπλοκ αναφέρονται συνήθως ως σταθερά μπλοκ. Υπάρχει το ενδεχόμενο, ορισμένα ή όλα τα μπλοκ να μην έχουν προκαθορισμένες θέσεις. Σε αυτήν την περίπτωση, χρειάζεται να τοποθετηθούν με τα cells σε αυτό που συνήθως αναφέρεται ως χωροθέτηση μεικτού τρόπου λειτουργίας.

Εκτός από τα ASIC, η χωροθέτηση διατηρεί την πρωταρχική της σημασία σε δομές παράταξης πυλών, όπως στα FPGA (Field Programmable Gate Array). Στα FPGA, η χωροθέτηση χαρτογραφεί τα υποκυκλώματα του κυκλώματος σε προγραμματιζόμενα λογικά μπλοκ FPGA με τρόπο που εγγυάται την ολοκλήρωση του επόμενου σταδίου δρομολόγησης.

Στόχοι και Περιορισμοί

Η χωροθέτηση συχνά διατυπώνεται ως ένα πρόβλημα με λίγα περιθώρια βελτίωσης. Ο περιορισμός είναι να αφαιρεθούν οι επικαλύψεις μεταξύ όλων των παρουσιών στη λίστα δικτύου. Οι μετρικές που είναι δυνατό να βελτιωθούν είναι οι:

- **Total wire length (Συνολικό μήκος καλωδίου):** Η όσο το δυνατόν μεγαλύτερη μείωση του συνολικού μήκους καλωδίου, ή του αθροίσματος του μήκους όλων των καλωδίων της σχεδίασης αποτελεί τον πρωταρχικό στόχο των περισσότερων υπαρχόντων placers. Αυτό όχι μόνο βοηθά στην ελαχιστοποίηση του μεγέθους του chip και, ως εκ τούτου του κόστους, αλλά και ελαχιστοποιεί την ισχύ και την καθυστέρηση, τα οποία είναι ανάλογα με το μήκος του καλωδίου (Αυτό προϋποθέτει ότι τα μακριά καλώδια έχουν πρόσθετη προσωρινή αποθήκευση, όλες οι ροές μοντέρνας σχεδίασης το κάνουν αυτό.)

- **Timing (Χρονισμός):** Ο κύκλος ρολογιού ενός chip καθορίζεται από την καθυστέρηση της μεγαλύτερης διαδρομής του, η οποία συχνά αναφέρεται ως κρίσιμη διαδρομή. Δεδομένης μιας προδιαγραφόμενης απόδοσης, ένας placer πρέπει να «σιγουρευτεί» πως δεν υπάρχει διαδρομή με καθυστέρηση, η οποία υπερβαίνει τη μέγιστη καθορισμένη καθυστέρηση.

- **Congestion (Συμφόρηση):** Ενώ το μήκος καλωδίου πρέπει να ελαχιστοποιηθεί για την κάλυψη των συνολικών πόρων δρομολόγησης, πρέπει επίσης να πληρούνται οι πόροι δρομολόγησης σε διάφορες τοπικές περιοχές του πυρήνα του chip. Μια περιοχή με κυκλοφοριακή συμφόρηση είναι πιθανό να οδηγήσει σε υπερβολικές παρακάμψεις δρομολογήσεων ή ακόμα και να καταστήσει αδύνατη την ολοκλήρωση όλων των διαδρομών.

- **Power (Ισχύς):** Η ελαχιστοποίηση ισχύος συνήθως περιλαμβάνει την κατανομή των θέσεων των cells με σκοπό να μειωθεί η συνολική

κατανάλωση ενέργειας δηλαδή να μετριαστούν τα «καυτά» σημεία και να ομαλοποιηθούν οι διαβαθμίσεις θερμοκρασίας.

- **Χρόνος εκτέλεσης :** Η μείωση του είναι δευτερεύον στόχος της χωροθέτησης.

Βασικές τεχνικές χωροθέτησης

Η χωροθέτηση χωρίζεται σε καθολική χωροθέτηση, νομιμοποίηση και λεπτομερή χωροθέτηση. Η καθολική χωροθέτηση εισάγει σημαντικές αλλαγές κατανέμοντας όλες τις παρουσίες σε κατάλληλες τοποθεσίες σε καθολική κλίμακα με επιτρεπόμενες μικρές επικαλύψεις. Η λεπτομερής χωροθέτηση μετατοπίζει κάθε περίπτωση σε κοντινή «νόμιμη» τοποθεσία με πολύ μικρή αλλαγή διάταξης. Η χωροθέτηση και η συνολική ποιότητα σχεδίασης εξαρτάται περισσότερο από την συνολική απόδοση της καθολικής χωροθέτησης. Η νομιμοποίηση (legalization) είναι ένα από τα πιο κρίσιμα βήματα στις σύγχρονές σχεδιάσεις χωροθέτησης. Δεδομένου ότι αρκετοί στόχοι όπως το μήκος καλωδίου και η δυνατότητα δρομολόγησης έχουν ήδη βελτιστοποιηθεί στο στάδιο καθολικής χωροθέτησης, ο στόχος της νομιμοποίησης δεν είναι μόνο η ευθυγράμμιση των cells χωρίς επικάλυψη στις γραμμές, αλλά και η διατήρηση της λύσης της καθολικής χωροθέτησης, δηλαδή η μετατόπιση των cells πρέπει να ελαχιστοποιηθεί.

Σε πρώιμο χρόνο, η χωροθέτηση ολοκληρωμένων κυκλωμάτων αντιμετωπίζεται με συνδυαστικές προσεγγίσεις. Όταν ο σχεδιασμός IC ήταν κλίμακας χιλιάδων πυλών, οι μεθοδολογίες προσομοίωσης ανόπτησης όπως το TimberWolf[13] παρουσιάζουν την καλύτερη απόδοση. Καθώς ο σχεδιασμός IC εισήχθη σε κλίμακα εκατομμυρίων ενοποιήσεων, η χωροθέτηση επιτεύχθηκε με χρήση αναδρομικής κατάτμησης υπερ-γράφων όπως το Capo[14].

Το Quadratic placement αργότερα ξεπέρασε τις συνδυαστικές λύσεις τόσο σε ποιότητα όσο και σε σταθερότητα. Ο GORDIAN διατυπώνει το κόστος του μήκους καλωδίου ως τετραγωνική συνάρτηση, ενώ συνεχίζει να διασπείρει τα cells μέσω της αναδρομικής κατάτμησης. Η πυκνότητα χωροθέτησης μοντελοποιείται απ' τον αλγόριθμο ως γραμμικός όρος στη συνάρτηση τετραγωνικού κόστους , με αυτό τον τρόπο λύνεται το πρόβλημα χωροθέτησης με καθαρό τετραγωνικό προγραμματισμό. Η πλειονότητα των σύγχρονων τετραγωνικών placers (KraftWerk[15], FastPlace, SimPL) ακολουθεί αυτό το πλαίσιο, ο καθένας χρησιμοποιεί διαφορετικά ευρετικά για να προσδιορίσει την γραμμική δύναμη πυκνότητας.

Η μη γραμμική χωροθέτηση (Nonlinear placement) φαίνεται να έχει καλύτερη απόδοση σε σύγκριση με άλλες κατηγορίες αλγορίθμων. Η μη γραμμική χωροθέτηση αρχικά μοντελοποιεί το μήκος καλωδίου με εκθετικές (μη γραμμικές) συναρτήσεις και αντίστοιχα την πυκνότητα με τοπικές

τετραγωνικές, προκειμένου να επιτευχθεί καλύτερη ακρίβεια και επομένως βελτίωση της ποιότητας. Άλλες ακαδημαϊκές μελέτες περιλαμβάνουν το Aplace[16] και το NTUplace[17].

Το ePlace είναι ο πιο σύγχρονος καθολικός αλγόριθμος χωροθέτησης. Διαχωρίζει περιπτώσεις προσομοιώνοντας ένα ηλεκτροστατικό πεδίο, το οποίο παρουσιάζει την ελάχιστη ποιότητα επιβάρυνσης, έτσι επιτυγχάνεται η καλύτερη απόδοση.

ΚΕΦΑΛΑΙΟ 2: PyPUT

Εισαγωγή

Η χωροθέτηση ορίζεται ως το πρόβλημα της τοποθέτησης των cells μιας σχεδίασης σε μια καθορισμένη περιοχή του chip. Ο τελικός σχεδιασμός δεν πρέπει να περιέχει ούτε επικαλύψεις ούτε υπερχειλίσεις cells μεταξύ τους, επιπλέον, η συνάρτηση που εξετάζει την ποιότητα της σχεδίασης πρέπει να βελτιστοποιείται. Στην τυπική χωροθέτηση, όλα τα cells έχουν το ίδιο ύψος και ποικίλο πλάτος, η περιοχή του chip χωρίζεται σε σειρές ίσου ύψους με αυτό των cells και μπορεί να περιέχει εμπόδια ή φραγμένες περιοχές.

Ο διαχωρισμός της διαδικασίας της χωροθέτησης συμβαίνει συνήθως σε τρείς φάσεις:

- Global placement (καθολική χωροθέτηση)
- Legalization (νομιμοποίηση του κυκλώματος)
- Detailed placement (λεπτομερής χωροθέτηση)

Στην καθολική χωροθέτηση τα cells απλώνονται σε όλη την περιοχή του chip έτσι ώστε μία ή περισσότερες συναρτήσεις στόχου να βελτιστοποιούνται (συνήθως αφορά το συνολικό μήκος καλωδίου διασύνδεσης, την πυκνότητα ή τον χρονισμό). Χωροθέτηση των cells στη βέλτιστη θέση τους οδηγεί σε ένα μη έγκυρο σχέδιο που περιέχει επικαλύψεις ή/και υπερχειλίσεις και καθιστά αναγκαία την επακόλουθη φάση της νομιμοποίησης. Κατά την φάση του legalization, οι επικαλύψεις και υπερχειλίσεις εξαλείφονται και τα cells τοποθετούνται μέσα στις σειρές του chip. Ως τελευταίο βήμα (λεπτομερής χωροθέτηση), πραγματοποιούνται μικρές αλλαγές που βελτιώνουν τα μετρικά που αναφέρθηκαν.

Η χωροθέτηση έχει συγκεντρώσει αρκετό ερευνητικό ενδιαφέρον παλαιότερα, με πολλές μελέτες να έχουν δημοσιευθεί σχετικά με καθεμία από τις παραπάνω φάσεις. Όσο αφορά την καθολική χωροθέτηση, υπάρχουν προσεγγίσεις βασισμένες σε μορφοποίηση του προβλήματος σε τετραγωνικές εξισώσεις (quadratic formulation). Τέτοιες δημοσιεύσεις είναι οι Gordian, SimPL, FastPlace και Polar, ενώ οι Kraftwerk και ePlace βασίζονται σε force directed μεθόδους. Στο στάδιο της νομιμοποίησης υπάρχουν οι αλγόριθμοι Tetris και Abacus οι οποίοι παραμένουν να είναι αποδοτικοί σε όποια ροή placement με βάση τον χρόνο εκτέλεσης. Τέλος σχετικά με την λεπτομερή χωροθέτηση εφαρμόζονται κυρίως ευριστικές μέθοδοι και διαφοροποιήσεις των αλγορίθμων νομιμοποίησης.

Εργαλείο PyPUT

Το PyPUT [4] στοχεύει στα κυκλώματα αναφοράς ISPD [18], που χρησιμοποιούνται συνήθως για να αποδείξουν την αποτελεσματικότητα των προσεγγίσεων χωροθέτησης. Η παροχή αυτοματισμών που βοηθούν στην ανάπτυξη και την εφαρμογή αλγορίθμων σχετικών με την χωροθέτηση αποτελεί τον στόχο του εργαλείου.

Εκτός από τις βασικές βιβλιοθήκες της Python, χρησιμοποιήθηκε ένα σύνολο καλά καθιερωμένων πακέτων προκειμένου να διατηρηθεί η συμβατότητα (με ήδη υπάρχουσες ή και μελλοντικές εκδόσεις της γλώσσας), η σταθερότητα και η επεκτασιμότητα. Οι συναρτήσεις που εφαρμόστηκαν ήταν άμεσο αποτέλεσμα της αποσύνθεσης κάθε βήματος των προαναφερθέντων αλγορίθμων και μπορούν να κατηγοριοποιηθούν ως εξής:

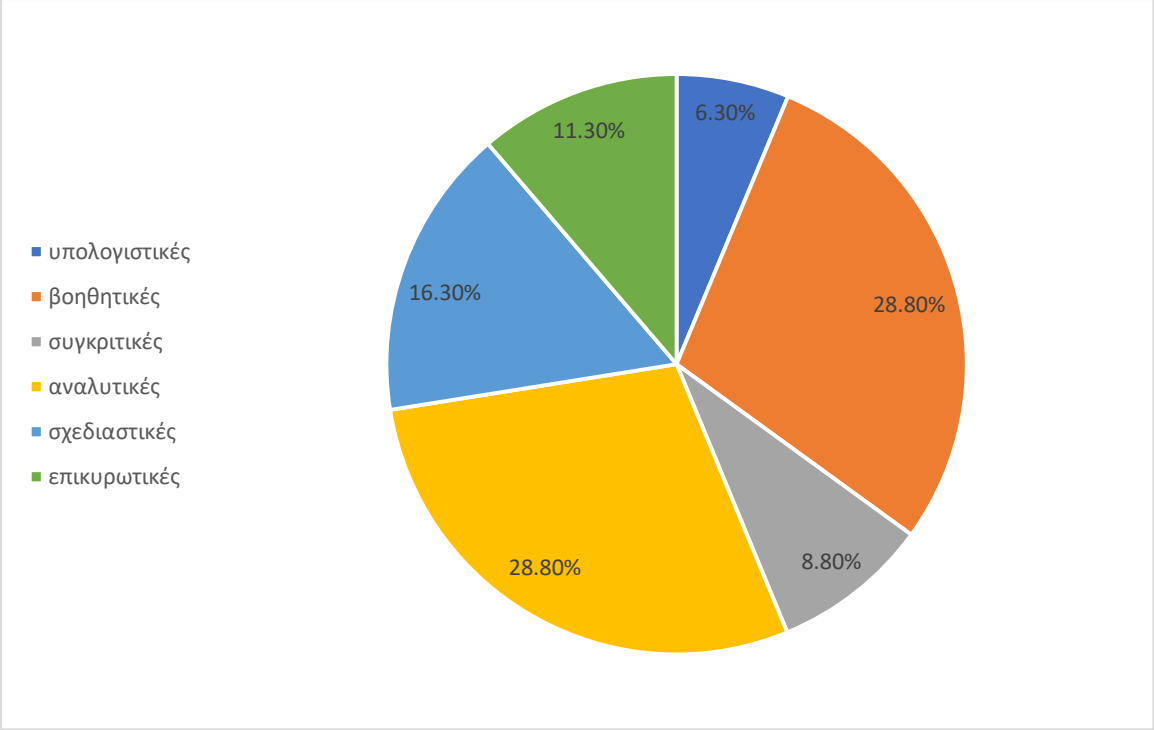
- Συγκριτικές
- Υπολογιστικές
- Αναλυτικές
- Σχεδιαστικές
- Βοηθητικές
- Επικυρωτικές

Στην πρώτη κατηγορία περιέχονται συναρτήσεις που μπορούν να χρησιμοποιηθούν για τη σύγκριση δύο ή περισσότερων προσεγγίσεων χωροθέτησης βάση των μετρήσεων που χρησιμοποιούνται συνήθως, σαν τον αριθμό υπερχειλίσεων, τον αριθμό επικαλύψεων, το συνολικό μήκος καλωδίου (hpwl) και την πυκνότητα.

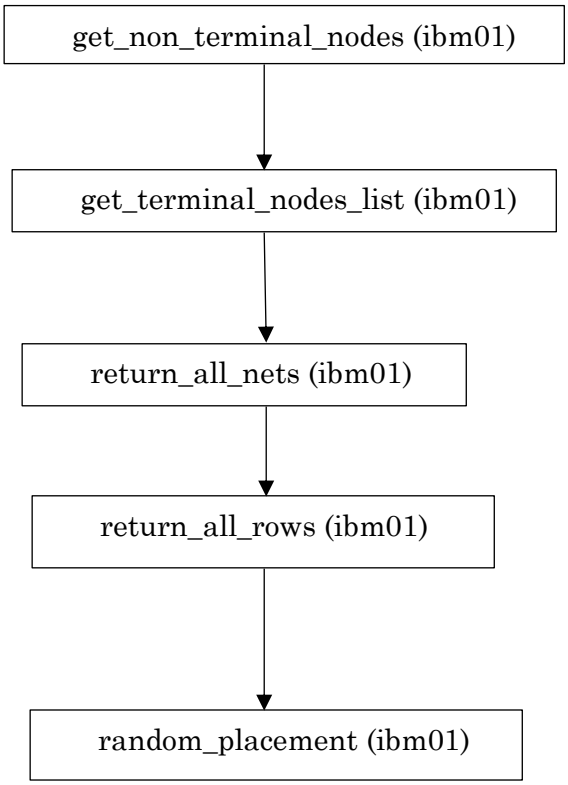
Η δεύτερη κατηγορία ενσωματώνει υπολογιστικές συναρτήσεις για τον υπολογισμό του μήκους καλωδίου και της πυκνότητας της συνολικής σχεδίασης ή ενός υποτιμήματος που ορίζεται από τον χρήστη.

Η λειτουργικότητα της αναλυτικής συνάρτησης μπορεί να είναι χρήσιμη, αφού κάθε προσπάθεια για την επίλυση του προβλήματος χωροθέτησης ξεκινά με την εξαγωγή των κατάλληλων δεδομένων από τα αρχεία αναφοράς. Υλοποιήθηκαν πολλαπλές εκδόσεις κάθε συνάρτησης ανάλογα με την έξοδο.

Στην κατηγορία επικύρωσης, εφαρμόστηκαν διάφοροι αυτοματοποιημένοι έλεγχοι, οι οποίοι μπορούν να βοηθήσουν τον προγραμματιστή να επαληθεύσει τυχόν ασυνέπειες στο σχεδιασμό. Πιο συγκεκριμένα, οι συναρτήσεις «check_move_validity», «check_move_hpwl» και «check_swap_hpwl», μπορούν να καθορίσουν εάν είναι δυνατόν να πραγματοποιηθεί μια μετακίνηση ή μια ανταλλαγή και ποιος θα είναι ο αντίκτυπος όσον αφορά το συνολικό μήκος καλωδίου.



Εικόνα 2. Κατανομή συναρτήσεων PyPUT



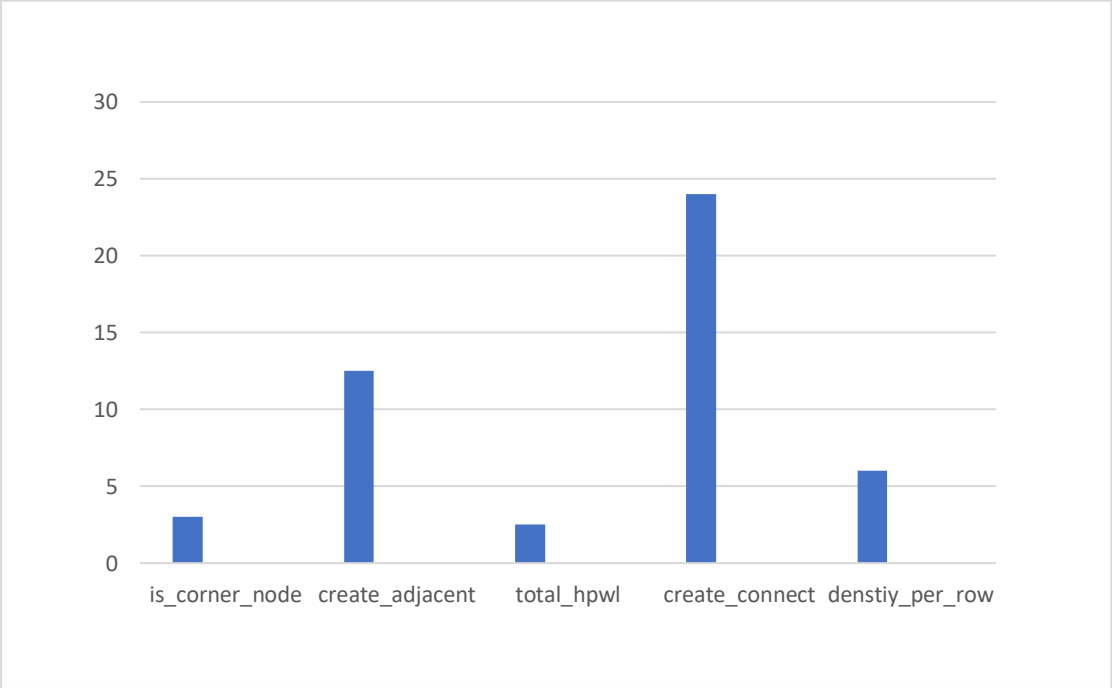
Εικόνα 3. Παράδειγμα χρήσης PyPUT

Παρέχονται επίσης δυνατότητες σχεδίασης. Τα plots μπορούν εύκολα να ενσωματωθούν σε οποιοδήποτε γραφικό εργαλείο, αλλά μπορούν να χρησιμοποιηθούν ως ένας τρόπος για να φανούν κρίσιμα μέρη μιας σχεδίασης, που πρέπει να εξεταστούν λεπτομερώς.

Τέλος, κάτω από το τμήμα της βιβλιοθήκης βοηθητικών συναρτήσεων, βρίσκεται μια μάλλον αποκλίνουσα ομάδα συναρτήσεων. Οι μέθοδοι ταξινόμησης μπορούν να χρησιμοποιηθούν σε προσεγγίσεις που βασίζονται σε ομαδοποίηση και να πραγματοποιηθούν κινήσεις των cells που γίνονται συνήθως, στις περιστροφές και εναλλαγές. Τα cells μπορούν να εντοπιστούν και να χαρακτηριστούν με βάση τη σημαντικότητα τους. Οι μετασχηματισμοί μπορούν να πραγματοποιηθούν γρήγορα με βάση τη διατύπωση του προβλήματος.

Αποτελέσματα

Συγκριτικά, οι συναρτήσεις που έχουν τον μεγαλύτερο χρόνο εκτέλεσης, είναι οι density_per_row(), is_corner_node(), total_hpwl(), create_adjacency_matrix(), create_connectivity_matrix(), αφού πρέπει να διαβάσουν πολλά αρχεία και να μαζέψουν πολύ περισσότερες πληροφορίες σε σχέση με τις άλλες συναρτήσεις. Σημασία έχει και η λειτουργικότητα των συγκεκριμένων συναρτήσεων. Στην πρώτη υπολογίζεται η πυκνότητα όλων των γραμμών της σχεδίασης, έτσι πρέπει να συγκεντρωθούν τα στοιχεία ολόκληρης της σχεδίασης (cells και γραμμές) και να υπολογίζεται η πυκνότητα. Παρόμοια ξεκινάει η εκτέλεση και της total_hpwl(), η οποία πρέπει να υπολογίσει αρχικά την ημιπερίμετρο καλωδίου για κάθε net ξεχωριστά ώστε έπειτα να υπολογιστεί το συνολικό μήκος καλωδίου. Τέλος η is_corner_node() βρίσκει αν ένα cell είναι γωνιακό, μέσα σε ένα ή περισσότερα nets στα οποία ανήκει, επομένως για τον υπολογισμό χρειάζεται για ένα cell, να ελεγχθούν όλα τα nets, και αντίστοιχα όλα τα υπόλοιπα cells τους.



Εικόνα 4. Χρόνος εκτέλεσης των πιο χρονοβόρων συναρτήσεων

Συμπεράσματα

Σε αυτό το κεφάλαιο παρουσιάστηκε ένα σύνολο εργαλείων βοηθητικών προγραμμάτων χωροθέτησης που βασίζεται σε Python. Η αρχική έκδοση της βιβλιοθήκης περιλαμβάνει πολλά χρήσιμα αποσπάσματα κώδικα που έχουν την δυνατότητα να χρησιμοποιηθούν ως δομικά στοιχεία για την κατασκευή αλγορίθμων που σχετίζονται με την χωροθέτηση. Διανέμεται ελεύθερα και μπορεί να βρεθεί στην ακόλουθη διεύθυνση URL: www.github.com/pyput.

Το PyPUT μπορεί εύκολα να επεκταθεί ως προς τη λειτουργικότητα. Λειτουργίες που υλοποιούν μονολιθικοί αλγόριθμοι ή μικρότερα μέρη τους μπορούν να προστεθούν χωρίς κόπο. Επιπλέον, με βάση τα αποτελέσματα, είναι σαφές ποιες λειτουργίες πρέπει να ελεγχθούν για περαιτέρω βελτιστοποίηση. Τέλος, λαμβάνοντας υπόψη τη συνεχή αύξηση του μεγέθους (όσον αφορά τον αριθμό cells) γίνεται προφανές ότι κάποιες τεχνικές παραλληλοποίησης θα πρέπει να εφαρμοστούν ώστε να μειωθεί ο χρόνος εκτέλεσης.

ΚΕΦΑΛΑΙΟ 3: ΑΛΓΟΡΙΘΜΗΚΕΣ ΠΡΟΣΕΓΓΙΣΕΙΣ

Οι αλγόριθμοι που σχεδιάστηκαν αναπτύχθηκαν σε γλώσσα προγραμματισμού C καθώς η γλώσσα είναι πιο απλή και πιο ευέλικτη από τις υπόλοιπες που θα μπορούσαν να χρησιμοποιηθούν (Python, Java κλπ). Το μόνο πρόβλημα που υπήρχε ήταν ότι λόγω της έλλειψης περιορισμών και τον ελλιπή έλεγχο λαθών που γίνεται, χρειάστηκε προσοχή στην σχεδίαση του κώδικα.

Ο κάθε κώδικας χωρίστηκε σε 3 βήματα, το κάθε ένα από αυτά θα αναλυθεί και αλγοριθμικά παρακάτω.

Στο πρώτο βήμα διαβάζονται τα αρχεία nodes,nets,scl και pl όπως προαναφέρθηκε και στην εισαγωγή. Τα στοιχεία τους αποθηκεύονται σε δυναμικούς πίνακες, εκτός από τα στοιχεία του αρχείου nets, τα οποία τοποθετούνται σε μια δυναμική λίστα και έπειτα γίνεται κατηγοριοποίηση των cells σε επίπεδα με βάση την λίστα που δημιουργήθηκε από το αρχείο nets.

Στο δεύτερο βήμα βρίσκουμε πιθανές θέσεις για τα cells μέσα στο chip. Αυτό γίνεται με 4 διαφορετικούς τρόπους για κάθε cell:

- Ξεκινάμε από το 1ο επίπεδο και επιλέγουμε το net στο οποίο ανήκει, με τον ελάχιστο αριθμό διασυνδέσεων (εξαιρούνται τα nets που έχουν pins).
- Ξεκινάμε από το τελευταίο επίπεδο και επιλέγουμε το net στο οποίο ανήκει, με τον ελάχιστο αριθμό διασυνδέσεων (εξαιρούνται τα nets που έχουν pins).
- Ξεκινάμε από το 1ο επίπεδο και επιλέγουμε το net στο οποίο ανήκει, με τον μέγιστο αριθμό διασυνδέσεων (εξαιρούνται τα nets που έχουν pins).
- Ξεκινάμε από το τελευταίο επίπεδο και επιλέγουμε το net στο οποίο ανήκει, με τον μέγιστο αριθμό διασυνδέσεων (εξαιρούνται τα nets που έχουν pins).

Στο τρίτο και τελευταίο βήμα ελέγχουμε κάθε cell αν μπορούμε να του αλλάξουμε θέση με οποιοδήποτε άλλο cell ώστε να μειωθεί το συνολικό hpwl. Ελέγχουμε μόνο μια φορά για πιθανές αλλαγές.

Αρχείο nodes

Το αρχείο nodes περιέχει το ύψος, το μήκος αλλά και το όνομα κάθε cell και κάθε pin που ανήκει στο netlist. Το ύψος κάθε cell είναι σε όλες τις σχεδιάσεις 16 μόνο το μήκος αλλάζει, ενώ όλα τα pins έχουν μήκος και ύψος ίσο με το μηδέν καθώς υπάρχουν σαν σημεία στην σχεδίαση.

Το στοιχείο που χρειάστηκε να αποθηκευτεί από το αρχείο σε δυναμικό πίνακα ήταν το ύψος του κάθε cell. Ταυτόχρονα μετρήθηκαν ποσά cells και pins υπήρχαν μέσα στο αρχείο.

```
step 1: read file using pointer c
step 2: if c is empty go to step 5
step 3: initialize dynamic array nodes and a counter
step 4: while c is not empty do
     if c doesn't point to a weight or the starting letter isn't "a" go to step 4
     nodes = c
     counter++
    else go to step 5
step 5: end
```

Εικόνα 5. Αλγόριθμος για το αρχείο nodes

Αρχείο nets

Το αρχείο nets περιέχει τις συνδέσεις των cells μέσα στον υπερ γράφο, το net degree δείχνει πόσα cells ή pins περιέχονται σε ένα net. Σε κάθε net υπάρχουν τουλάχιστον 2 συνδέσεις, τα cells μπορούν να συνδέονται μεταξύ τους αλλά και με pins, όλα τα pins πρέπει να συνδέονται αναγκαστικά με κάποιο cell και ποτέ με άλλα pins.

Απ' το αρχείο χρειάστηκε να αποθηκευτεί σε μια δυναμική απλά συνδεδεμένη λίστα, το κάθε net, αυτό που δημιουργήθηκε στο τέλος ήταν ένας πίνακας από δυναμικές λίστες, οπού η κάθε λίστα ήταν ένα net.

```
step 1: read file using pointer c
step 2: if c is empty go to step 5
step 3: initialize dynamic array of lists netlist
step 4: temp = 0
step 5: while c is not empty do
            if c = the number after net degree then temp = c
                    while temp>0 do
                            add the element to netlist
                    else go to step 5
            else go to step 5
        else go to step 6
step 6: end
```

Εικόνα 6. Αλγόριθμος για το αρχείο nets

Αρχείο scl

Το αρχείο scl περιέχει πληροφορίες για κάθε οριζόντια γραμμή της σχεδίασης όπως την συντεταγμένη, το πλάτος της θέσης, την διάσταση της θέσης της, τον προσανατολισμό της, την συμμετρία της θέσης της, την προέλευση της υποσμίας της και τον αριθμό των θέσεων της.

Τα προηγούμενα μεγέθη είναι όλα προκαθορισμένα εκτός από την συντεταγμένη, η οποία στην αρχή είναι 0 και αυξάνεται ανά 16 κάθε φορά και είναι το μοναδικό στοιχείο που πρέπει να κρατήσουμε από αυτό το αρχείο μαζί με τον αριθμό θέσεων, καθώς ο αριθμός των γραμμών θα πολλαπλασιαστεί με το 16 και το αποτέλεσμα της πράξης θα είναι το μέγιστο του άξονα Y ενώ ο αριθμός θέσεων θα είναι το μέγιστο του άξονα X. Αυτό χρησιμεύει στο να σχηματιστεί το περίγραμμα του chip μέσα στο οποίο θα τοποθετηθούν τα cells.

```
step 1: read file using pointer c
step 2: if c is empty go to step 5
step 3: initialize dynamic array scl
step 4: while c is not empty do
            if c = coordinates
                    scl = c
            else go to step 4
        else go to step 5
step 5: end
```

Εικόνα 7. Αλγόριθμος για το αρχείο scl

Αρχείο pl

Το αρχείο pl περιέχει τις συντεταγμένες του κάθε cell και του κάθε pin. Από αυτό το αρχείο είναι ευνόητο ότι χρειαζόμαστε τις συντεταγμένες αυτές, που με την σειρά τους χρειάζονται για τον υπολογισμό του καλωδίου της ημιπεριμέτρου (μόνο οι συντεταγμένες των cells).

```
step 1: read file using pointer c
step 2: if c is empty go to step 5
step 3: initialize 2 dynamic arrays pl1 and pl2
step 4: initialize 2 dynamic arrays pl3 and pl4
step 4: while c is not empty do
            if the letter before we reach the coordinates is “a”
                    if c points to coordinate X
                            pl1 = c
                    else if c points to coordinate Y
                            pl2 = c
                    else go to step 4
            else
                    if c points to coordinate X
                            pl3 = c
                    else if c points to coordinate Y
                            pl4 = c
                    else go to step 4
        else go to step 5
step 5: end
```

Εικόνα 8. Αλγόριθμος για το αρχείο pl

Levelization

Το levelization γίνεται εφόσον έχουμε διαβάσει και αποθηκεύσει τα προηγούμενα αρχεία ή τουλάχιστον μόλις γίνει η ανάγνωση του αρχείου nets. Πρόκειται για την κατάταξη των cells σε επίπεδα με βάση τα nets στα οποία ανήκουν.

Αρχικά στο πρώτο επίπεδο βάζουμε όλα τα pins, έπειτα βρίσκουμε τα cells με τα οποία συνδέεται το κάθε pin και τα τοποθετούμε στο δεύτερο επίπεδο και λειτουργούμε αντίστοιχα για τα υπόλοιπα επίπεδα που πρόκειται να δημιουργηθούν. Είναι σημαντικό να μην υπάρχουν διπλότυπα είτε στο ίδιο επίπεδο είτε σε διαφορετικά.

Η υλοποίηση του levelization έγινε με την χρήση δυναμικού πίνακα δυναμικών λιστών, οπού κάθε επίπεδο είναι μια λίστα και ο αριθμός των στοιχείων του πίνακα είναι ο αριθμός των επιπέδων.

```
step 1: start
step 2: create dynamic array of lists levels
step 3: level = 0
step 3: I = 0
step 4: add all pins to level
step 5: I = 0, level++
step 6: while I < number of pins do
            add all nodes connected to pins in level
            I++
        else go to step 7
step 7: I = 0
step 8: while I < number of nodes do
            level++
            add all nodes that are connected to the nodes from the
previous level in current level
            I++
        else go to step 9
step 9: end
```

Εικόνα 9. Αλγόριθμος για το levelization

Εύρεση θέσεων

Στο στάδιο της εύρεσης θέσεων θα ψάξουμε να βρούμε τις πιθανές θέσεις των cells πάνω στο chip. Όπως ειπώθηκε προηγουμένως στο αρχείο scl οι άξονες X,Y έχουν πάρει κάποιες μέγιστες τιμές με αποτέλεσμα να έχει δημιουργηθεί ένα νοητό περίγραμμα του chip. Μέσα σε αυτό το περίγραμμα θα προσπαθήσουμε να βάλουμε σε όσο καλύτερες θέσεις γίνεται τα cells.

Αρχικά ξεκινάμε από ένα επίπεδο, είτε το πρώτο είτε το τελευταίο, ανάλογα με την παραλλαγή που τρέχουμε και για κάθε cell ψάχνουμε να βρούμε το net που ανήκει, με τον μεγαλύτερο ή μικρότερο αριθμό διασυνδέσεων, πάλι ανάλογα με την παραλλαγή που τρέχουμε. Μόλις το βρούμε υπολογίζουμε την ημιπερίμετρο καλωδίου του, η οποία υπολογίζεται από την πρόσθεση του απολύτου της αφαίρεσης μεταξύ της μέγιστης τιμής με της ελάχιστης του Y μέσα στο net και του απολύτου της αντίστοιχης πράξης για το X. Δηλαδή:

|Ymax – Ymin| + |Xmax – Xmin|

Πρακτικά μετακινούμε το κάθε cell μέσα στο περίγραμμα και σε το τοποθετούμε στην θέση, οπού η ημιπερίμετρος καλωδίου έχει την μικρότερη τιμή. Είναι σημαντικό να σημειωθεί ότι μόλις τοποθετηθεί ένα cell η θέση αυτή σημειώνεται ως κατειλημμένη καθώς δεν μπορεί να τοποθετηθεί ένα cell πάνω σε ένα άλλο.

Αφότου ολοκληρωθεί η χωροθέτηση των cells πάνω στο chip θα βγει ένα πρώτο συνολικό μήκος καλωδίου το οποίο θα είναι αποτέλεσμα της πρόσθεσης όλων των μηκών καλωδίων της ημιπεριμέτρου του κάθε cell. Αυτό το συνολικό μήκος καλωδίου θα κοιτάξουμε να μειώσουμε στο επόμενο στάδιο του swap.

```
step 1: start
step 2: initialize X as num sites and Y as 16*num rows
step 3: J=0, K=0, total hpwl = 0
step 4: for I=1 to I=level do
            cell = (pl1, pl2)
             find the net with minimum connections and doesn't
include a pin
            calculate hpwl of said net
step 5:     while J < Y do
step 6:             while K < X do
                        cell = (K, J)
                        if a cell with the same coordinates exists go
to step 6
                        recalculate hpwl with the new coordinates
                        if new hpwl < old hpwl
                                cell = (K, J)
                        else go to step 6
                        K += width of the cell
                    else go to step 5
                    J += height of the cell
                    total hpwl += hpwl
            else go to step 4
step 7: end
```

Εικόνα 10. Αλγόριθμος για την εύρεση θέσεων

Swap

Το στάδιο του swap είναι το τελικό στάδιο του αλγορίθμου, στο οποίο θα ελέγξουμε ένα προς ένα τα cells για πιθανές ανταλλαγές οι οποίες θα αποσκοπήσουν στην μείωση του συνολικού μήκους καλώδιού.

Η ιδέα του swap είναι να συγκριθούν τα cells ένα προς ένα μεταξύ τους, δηλαδή να ανταλλάξουν θέσεις και να υπολογίσουμε εκ νέου το μήκος καλωδίου απ' τις νέες θέσεις τους. Εάν η ανταλλαγή αυτή φέρει μείωση του συνολικού μήκους καλωδίου τότε κρατάμε τις θέσεις όπως είναι αλλιώς επιστρέφουν στην αρχική.

Η σωστή υλοποίηση αυτού του σταδίου γίνεται κανονικά με πάνω από μια επαναλήψεις, καθώς πρέπει να φτάσει σε σημείο ο αλγόριθμος να μην επιδέχεται άλλες ανταλλαγές (swaps) μεταξύ των cells, αυτό θα επιτυγχανόταν μόλις το μήκος καλωδίου είχε φτάσει σε ένα σημείο, οπού ότι swap και να γινόταν δεν θα μπορούσε να μειωθεί. Λόγω του υπερβολικού χρόνου που θα χρειαζόταν για να συμβεί αυτό, στον αλγόριθμο αυτόν δεν έγινε καμμιά επανάληψη.

```
step 1: start
step 2: while I < number of cells
step 3:     while J < number of cells
                    if cell(I) has the same width as cell(J)
                            swap coordinates between cells
                            recalculate total hpwl from new coordinates
                            if new total hpwl < old total hpwl
                                    keep the swapped coordinates
                            else go to step 3
                    else go to step 3
              J++
              else go to step 2
            I++
           else go to step 4
step 4: end
```

Εικόνα 11. Αλγόριθμος για το swap

Σημειώσεις

Η υλοποίηση του αλγορίθμου ήταν σχετικά απλή, το στάδιο που ήταν πιο απαιτητικό ήταν το levelization. Όπως θα παρατηρήσετε και στο επόμενο κεφάλαιο που αναλύονται τα αποτελέσματα ο χρόνος εκτέλεσης είναι αρκετά μεγάλος, αυτό οφείλεται στην καθυστέρηση που συμβαίνει στο στάδιο του swap. Στο στάδιο αυτό η προσέγγιση που ακολουθήθηκε ήταν η brute force λογική καθώς ελέγχονται ένα προς ένα τα στοιχεία, αλλά εκτός του ότι συγκρίνονται όλα τα cells αναζητείται το net στο οποίο βρίσκονται, καθώς δεν έχουν μπει σε ένα struct με αποτέλεσμα να αναζητούνται τα nets του κάθε cell και η ημιπερίμετρος καλωδίου του κάθε net.

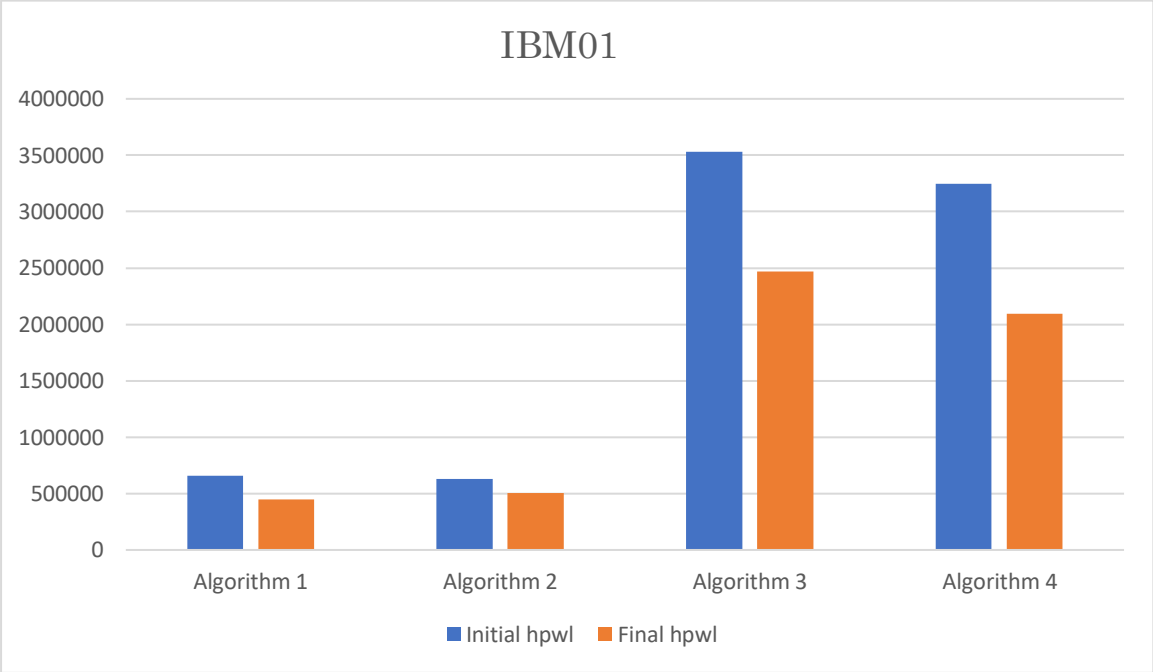
Κεφάλαιο 4: ΑΠΟΤΕΛΕΣΜΑΤΑ

Εισαγωγή

Σε αυτό το κεφάλαιο παρουσιάζονται τα αποτελέσματα από την εφαρμογή των αλγορίθμων πάνω στα κυκλώματα. Τα αποτελέσματα αυτά είναι το αρχικό μήκος καλωδίου (initial hpwl) σε σχέση με το τελικό μήκος καλωδίου (final hpwl), ο συνολικός αριθμός των ανταλλαγών που πραγματοποιήθηκαν (total swaps), η ελάττωση που έγινε στο μήκος καλωδίου σε ποσοστό επί τις εκατό και ο χρόνος εκτέλεσης (runtime).

Αρχικό και τελικό μήκος καλωδίου

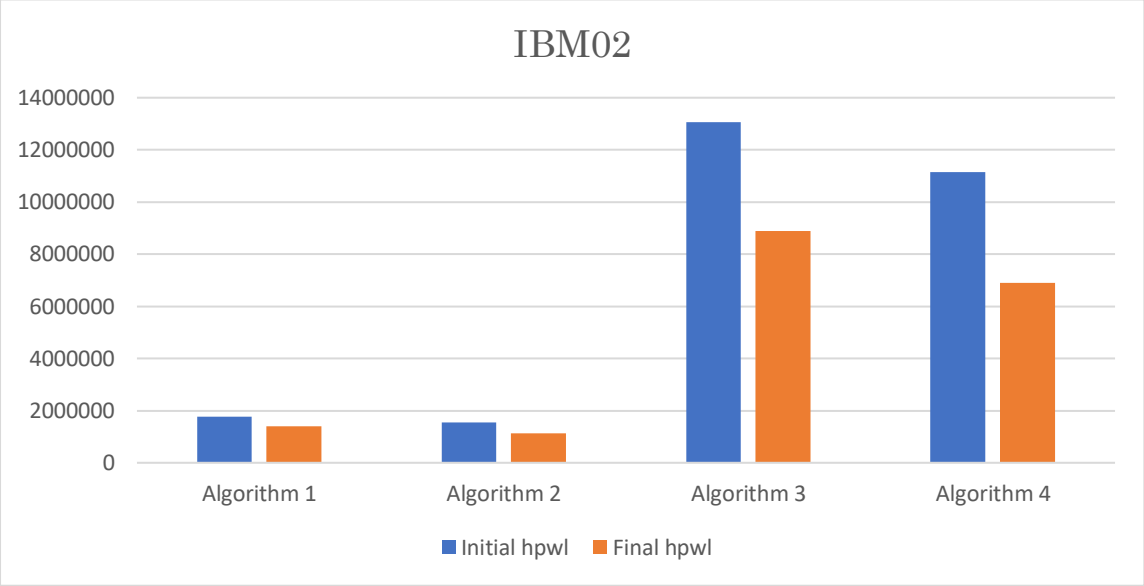
Στην παρακάτω παρουσιάζεται η διαφορά του αρχικού με το τελικό μήκος καλωδίου μετά την εφαρμογή των αλγορίθμων για τα κυκλώματα ibm01, ibm02 και ibm03, αφότου γίνουν οι κατάλληλες ανταλλαγές.



Εικόνα 12. Initial hpwl και final hpwl για το ibm01

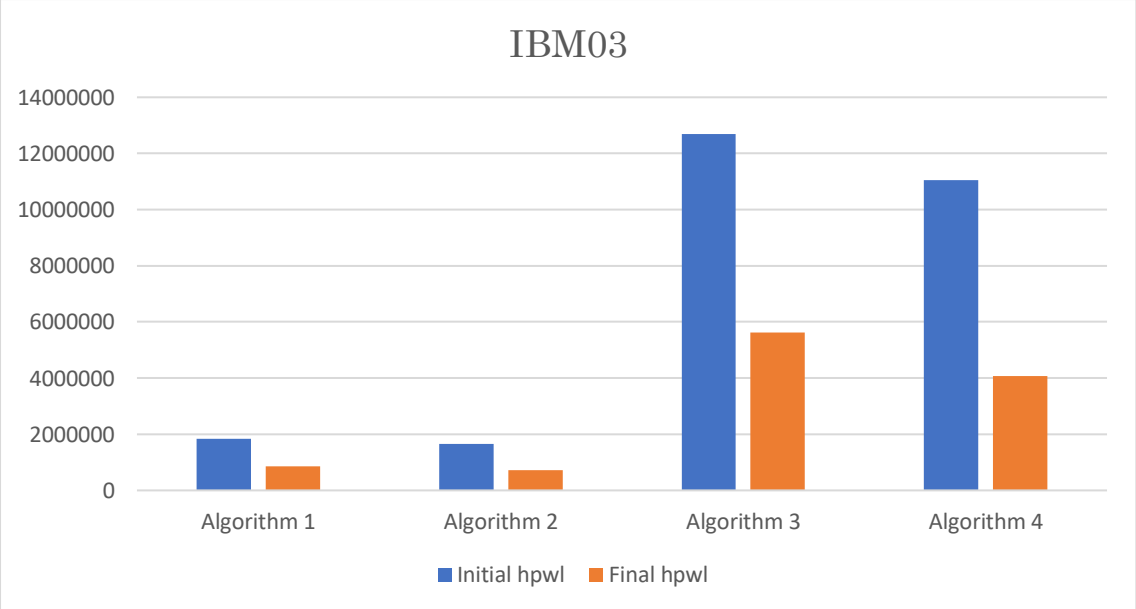
Στο ibm01 (Εικόνα 12) παρατηρείται η μεγαλύτερη ελάττωση στο συνολικό μήκος καλωδίου στον τέταρτο αλγόριθμο. Το αναμενόμενο είναι να συμβεί κάτι ανάλογο στον τρίτο και στον τέταρτο γιατί σε αυτούς τους δύο το αρχικό μήκος καλωδίου είναι πολύ μεγαλύτερο από τους δύο πρώτους, καθώς όπως εξηγήθηκε και στο προηγούμενο κεφάλαιο οι δύο τελευταίοι αναζήτησαν τα cells με το μεγαλύτερο net degree.

Στο ibm02 (Εικόνα 13) ο αλγόριθμος που παρουσιάζει την μεγαλύτερη μείωση είναι ο τέταρτος πάλι. Η διαφορά στην ελάττωση μεταξύ τρίτου και τέταρτου είναι μεγαλύτερη στο ibm02 από ότι στο ibm01 όπως θα φανεί και παρακάτω, όπου αυτή η μείωση παρουσιάζεται σε ποσοστό επί τις εκατό.



Εικόνα 13. Initial hpwl και final hpwl για το ibm02

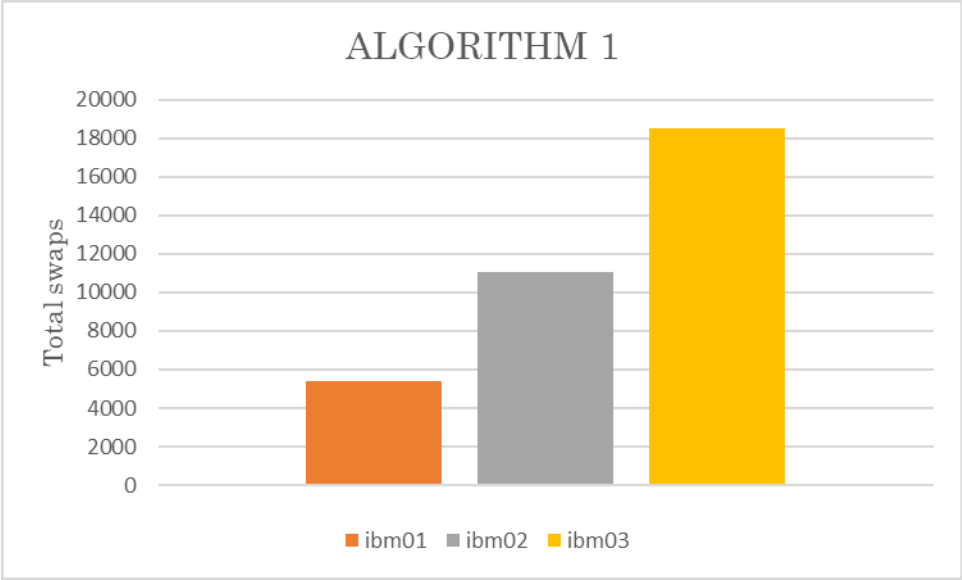
Στο ibm03 (Εικόνα 14), πάλι ο τέταρτος αλγόριθμος παρουσιάζει την μεγαλύτερη διαφορά στην ελάττωση, όπως και στους προηγούμενους.



Εικόνα 14. Initial hpwl και final hpwl για το ibm03

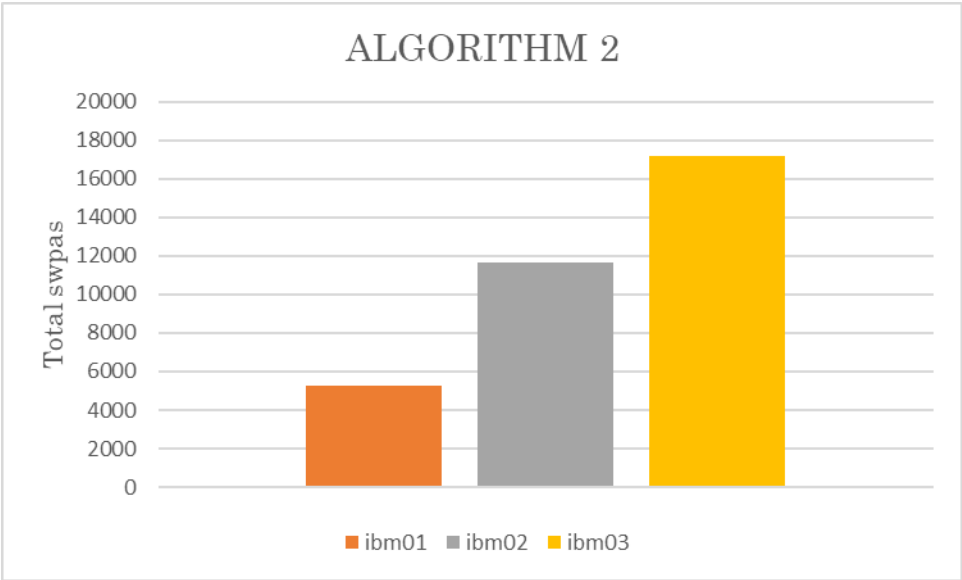
Αριθμός ανταλλαγών

Παρακάτω παρουσιάζεται ο συνολικός αριθμός ανταλλαγών που πραγματοποιήθηκε σε κάθε αλγόριθμο στα τρία ibms.



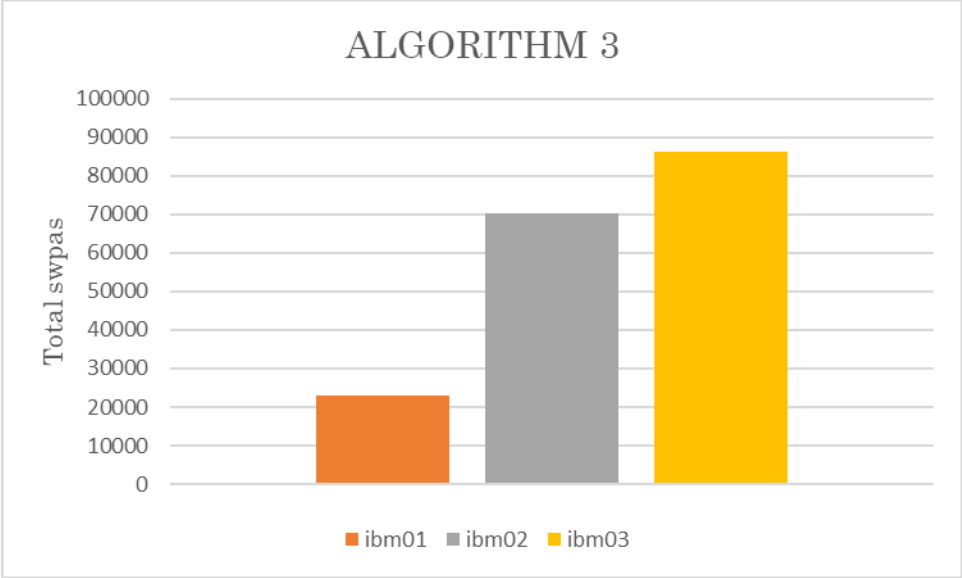
Εικόνα 15. Συνολικές ανταλλαγές του 1ου αλγορίθμου για τα 3 ibms

Στον πρώτο αλγόριθμο (Εικόνα 15) και στον δεύτερο (Εικόνα 16) παρατηρούμε ότι ο αριθμός των ανταλλαγών αυξάνεται αναλογικά με τον αριθμό των cells του κάθε ibm.

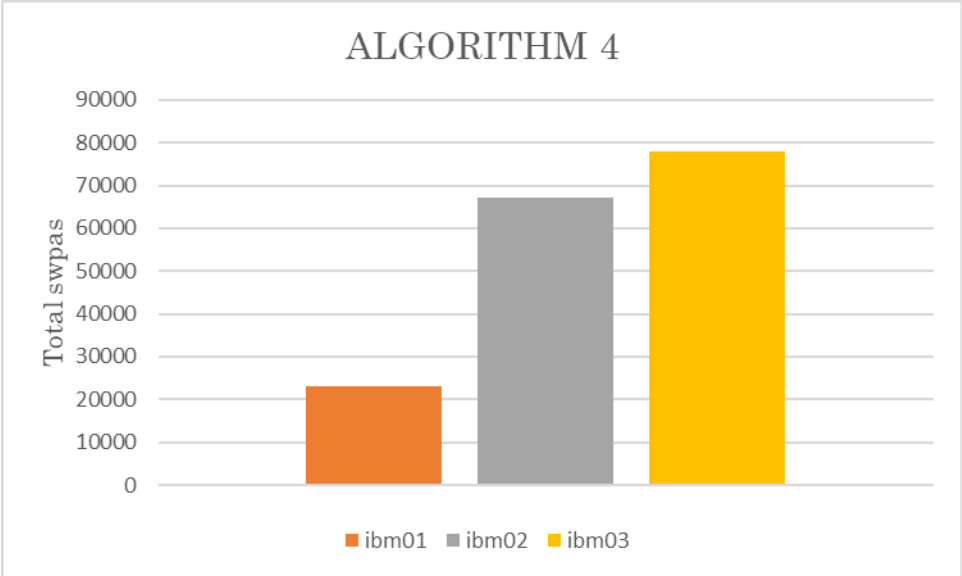


Εικόνα 16. Συνολικές ανταλλαγές του 2ου αλγορίθμου για τα 3 ibms

Στους δυο τελευταίους αλγορίθμους (Εικόνες 17 και 18) παρατηρούμε ότι ενώ η διαφορά στον αριθμό ανταλλαγών μεταξύ ibm01 και ibm03 είναι σχεδόν ίδια με τους δύο πρώτους, η διαφορά μεταξύ ibm02 και ibm03 είναι πολύ πιο μικρή σε σχέση με αυτούς. Επίσης στους δύο τελευταίους αλγορίθμους πραγματοποιούνται περισσότερες ανταλλαγές συγκριτικά με τους δύο πρώτους.



Εικόνα 17. Συνολικές ανταλλαγές του 3ου αλγορίθμου για τα 3 ibms



Εικόνα 18. Συνολικές ανταλλαγές του 4ου αλγορίθμου για τα 3 ibms

Υπόλοιπες μετρικές

Εδώ εμφανίζονται κάποια επιπλέον στοιχεία/μετρικές ο οποίες αφορούν την ποιότητα του συνολικού αποτελέσματος όπως είναι η ελάττωση επί τις εκατό και ο χρόνος εκτέλεσης του κάθε αλγορίθμου.

design	**ΔHPWL (%)**	**runtime**
ibm01	31,50%	90490 sec
ibm02	20,60%	317802 sec
ibm03	53%	745178 sec

Πίνακας 1. Υπόλοιπα αποτελέσματα του 1ου αλγορίθμου

design	**ΔHPWL (%)**	**runtime**
ibm01	19,80%	88934 sec
ibm02	27,10%	313523 sec
ibm03	56,90%	739581 sec

Πίνακας 2. Υπόλοιπα αποτελέσματα του 2ου αλγορίθμου

design	**ΔHPWL (%)**	**runtime**
ibm01	30,00%	91675 sec
ibm02	31,90%	318847 sec
ibm03	55,70%	747381 sec

Πίνακας 3. Υπόλοιπα αποτελέσματα του 3ου αλγορίθμου

design	**ΔHPWL (%)**	**runtime**
ibm01	35,50%	91039 sec
ibm02	37,90%	318026 sec
ibm03	63,20%	744125 sec

Πίνακας 4. Υπόλοιπα αποτελέσματα του 4ου αλγορίθμου

Από τα παραπάνω αποτελέσματα μπορούμε να διακρίνουμε ότι ο χρόνος εκτέλεσης για όλα τα ibms είναι πολύ μεγάλος και στους τέσσερις αλγορίθμους, όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο, ο λόγος είναι ότι πραγματοποιούνται πάρα πολλές ανταλλαγές. Ο ποιο γρήγορος αλγόριθμος φαίνεται να είναι ο δεύτερος για όλα τα ibms.

Η επόμενη μετρική είναι η ελάττωση του μήκους καλωδίου σε ποσοστό επί τις εκατό, η οποία οφείλεται στην φάση της ανταλλαγής των cells. Παρατηρούμε ότι και στους τέσσερις αλγορίθμους το τρίτο ibm εμφανίζει την μεγαλύτερη μείωση. Για τα άλλα δυο βλέπουμε ότι στον πρώτο αλγόριθμο υπάρχει μεγάλη διαφορά στην μείωση καθώς το πρώτο ibm εμφανίζει μεγαλύτερη με ποσοστό 31,5% απ' το δεύτερο που έχει μόλις 20,6%, στον δεύτερο αλγόριθμο γίνεται το ανάποδο, αλλά η διαφορά αυτήν τη φορά είναι μικρότερη. Στους δύο τελευταίους το δεύτερο εμφανίζει μεγαλύτερη μείωση απ' το πρώτο, αλλά με πολύ μικρότερη διαφορά από τους πρώτους δύο αλγορίθμους.

Συμπεράσματα

Από τα αποτελέσματα που παρουσιάστηκαν παραπάνω, φτάνουμε στο συμπέρασμα ότι το τρίτο ibm τα πάει καλυτέρα όσο αφορά την ελάττωση του μήκους καλωδίου, καθώς το ποσοστό είναι πάνω από 50% σε όλους τους αλγορίθμους, παρόλα αυτά έχει τον μεγαλύτερο χρόνο εκτέλεσης από τα άλλα δύο, κάτι που είναι αναμενόμενο καθώς έχει περισσότερα cells από αυτά.

Αυτός ο μεγάλος χρόνος εκτέλεσης οφείλεται στον μεγάλο αριθμό των cells που υπάρχουν στα κυκλώματα ibm, καθώς όλα τα cells πρέπει να ελεγχθούν ένα προς ένα για πιθανές ανταλλαγές, (με το συνολικό αριθμό ανταλλαγών να φτάνει μέχρι και τις 88.000 για το ibm03) με σκοπό να μειωθεί το συνολικό μήκος καλωδίου. Το πρόβλημα αυτό θα μπορούσε να λυθεί με χρήση δυναμικών δομών struct όπου για το κάθε cell θα αποθηκεύονται τα nets στα οποία ανήκει. Με λίγα λόγια το πρόβλημα είναι ότι ο αλγόριθμος αναγκάζεται να «ψάχνει» τα nets του κάθε cell αντί να τα παίρνει έτοιμα.

Ολοκληρώνοντας, η σημαντικότερη μετρική είναι η ελάττωση του μήκους καλωδίου, το ibm03 εμφανίζει την μεγαλύτερη ελάττωση σε όλους τους αλγορίθμους, ενώ μπορούμε να πούμε ότι το ibm02 έρχεται δεύτερο καθώς σε όλους τους αλγορίθμους εκτός από τον πρώτο, υπερτερεί του ibm01. Παρόλα αυτά το ibm01 υπερτερεί έναντι των υπολοίπων στο κομμάτι του χρόνου εκτέλεσης σε όλους τους αλγορίθμους, στον πρώτο αλγόριθμο φαίνεται να επιτυγχάνεται η καλύτερη επίδοση του, παρόλο που ο χρόνος εκτέλεσης είναι ο δεύτερος καλύτερος. Το ibm02 και το ibm03 πετυχαίνουν τις καλύτερες επιδόσεις τους στον τέταρτο αλγόριθμο, όσο αφορά την ελάττωση του μήκους καλωδίου, καθώς οι χρόνοι εκτέλεσης δεν είναι καλύτεροι από τους υπόλοιπους αλγορίθμους.

Σύνοψη

Η ροή της φυσικής σχεδίασης περιέχει ένα μεγάλο αριθμό σημαντικών σταδίων . Μέσα σε αυτά βρίσκεται και η χωροθέτηση που είναι ένα από τα πιο σημαντικά καθώς καθορίζει το τελικό αποτέλεσμα της σχεδίασης. Η χωροθέτηση χωρίζεται σε τρία στάδια, μέσα στα οποία σχεδιάζονται κι εφαρμόζονται διάφοροι αλγόριθμοι. Τα στάδια αυτά είναι η καθολική χωροθέτηση, η νομιμοποίηση και η λεπτομερής χωροθέτηση. Οι αλγόριθμοι χωροθέτησης σχεδιάζονται και λειτουργούνε πάνω σε ένα ή παραπάνω στάδια με τελικό στόχο την βελτιστοποίηση της σχεδίασης βάση κάποιας τιμής που τις περισσότερες φορές είναι το μήκος καλωδίου.

Για την δημιουργία ενός τέτοιου αλγορίθμου απαιτείται μια προεργασία κατά την οποία ο σχεδιαστής πρέπει να διαβάσει τα αρχεία του αρχικού κυκλώματος και να σχηματίσει τη σχεδίαση σε μια διαχειρίσιμη μορφή για τον αλγόριθμο που σχεδιάζει. Υπάρχει όμως έλλειψη ολοκληρωμένων εργαλείων που ανήκουν σε αυτήν την κατηγορία, γι' αυτό δημιουργήθηκε ο αλγόριθμος που παρουσιάστηκε σε αυτήν την μελέτη. Η εκτέλεση του αλγορίθμου αυτού έγινε πάνω σε τρία κυκλώματα και διαπιστώθηκε ότι λειτουργεί ομαλά. Παρόλα αυτά όπως φαίνεται στο κεφάλαιο 4 (με βάση τα αποτελέσματα) ο χρόνος εκτέλεσης είναι υπερβολικά μεγάλος, δεδομένου ότι στην φάση της ανταλλαγής δεν συμβαίνουν επαναλήψεις.

Η μεγάλη καθυστέρηση συμβαίνει στην φάση της ανταλλαγής, καθώς το net του κάθε cell δεν έχει αποθηκευτεί σε μια δομή struct και πρέπει ο αλγόριθμος να «ψάχνει» ένα προς ένα να βρει το net του κάθε cell ώστε να υπολογίσει την ημιπερίμετρο καλώδιού του. Επειδή λοιπόν τα κυκλώματα αποτελούνται από 12.000 έως 22.000 cells ο χρόνος εκτέλεσης κυμαίνεται από μια μέρα έως μια εβδομάδα.

Συμπερασματικά, οι αλγόριθμοι που σχεδιάστηκαν μπορούν και χρειάζεται να βελτιστοποιηθούν, βάση αυτού θα γίνει προσπάθεια τελειοποίησης του στο μέλλον.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://www.its.edu.in/RACE2016/RACE2014/finalpdfpapers/RACE1011.pdf

[2] https://en.wikipedia.org/wiki/Physical_design_(electronics)

[3] https://en.wikipedia.org/wiki/Placement_(electronic_design_automation)

[4] Georgios Kranas, George-Chris Tsalamagkakis, Panagiotis Oikonomou, Antonios N. Dadaliaris. “PyPUT: Python-based Placement Utilities Toolset.” SEEDA 2018, Kastoria Greece

[5] Naveed A. Sherwani. “Algorithms for VLSI Physical Design Automation.” 2012

[6] J. M. Kleinhans, G. Sigl, F. M. Johannes and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 3, pp. 356-365, March 1991

[7] M. C. Kim, D. J. Lee, and I. L. Markov, "SimPL: An effective placement algorithm," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 50-60, January 2012

[8] N. Viswanathan, and C. N. Chu, "FastPlace: efficient analytical placement using cell shifting, iterative local refinement, and a hybrid net model," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 722-733, May 2005

[9] T. Lin, C. Chu, J. R. Shinnerl, I. Bustany, and I. Nedelchev, "POLAR: placement based on novel rough legalization and refinement," In Proceedings of the International Conference on Computer-Aided Design, pp.357-362, November 2013

[10] J. Lu, P. Chen, C. C. Chang, L. Sha, D. J. Huang, C. C. Teng, and C. K. Cheng, "ePlace: Electrostatics-based placement using fast Fourier transform and Nesterov's method," ACM Transactions on Design Automation of Electronic Systems, p. 17, March 2015

[11] D. Hill, "Method and system for high-speed detailed placement of cells within an integrated circuit design," Patent US 6370673 B1, 2002

[12] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: fast legalization of standard cell circuits with minimal movement," In Proceedings of the 2008 international symposium on Physical design, pp. 47-53, April 2008

[13] C. Sechen and A. Sangiovanni-Vincentelli. TimberWolf3.2: A New Standard Cell Placement and Global Routing Package. In DAC, pages 432–439, 1986.

[14] Caldwell, A.E.; Kahng, A.B.; Markov, I.L. (June 2000). "Can recursive bisection alone produce routable placements? ". Proceedings of the 37th Design Automation Conference. pp. 477–482.

[15] P. Spindler, U. Schlichtmann, and F. M. Johannes. Kraftwerk2 - A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model. IEEE TCAD, 27(8):1398– 1411, 2008.

[16] A. B. Kahng, S. Reda, and Q. Wang, "Architecture and Details of a High Quality, Large Scale Analytical Placer", In ICCAD 2005, pp. 891-898.

[17] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang. NTUPlace3: An Analytical Placer for Large-Scale Mixed-Size Designs with Preplaced Blocks and Density Constraint. IEEE TCAD, 27(7):1228– 1240, 2008.

[18] http://vlsicad.eecs.umich.edu/BK/Slots/cache/www.public.iastate.edu/~nataraj/ISPD04_Bench.html