ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Υλοποίηση και Ανάπτυξη Αλγορίθμων Λεπτομερούς Χωροθέτησης Ολοκληρωμένων Κυκλωμάτων

ΚΡΑΝΑΣ ΓΕΩΡΓΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Δαδαλιάρης Αντώνιος
Επίκουρος Καθηγητής

ΣΥΝΕΠΙΒΛΕΠΩΝ

Κοζύρη Μαρία
Επίκουρη Καθηγήτρια

Λαμία 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Υλοποίηση και Ανάπτυξη Αλγορίθμων Λεπτομερούς Χωροθέτησης Ολοκληρωμένων Κυκλωμάτων

ΚΡΑΝΑΣ ΓΕΩΡΓΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Δαδαλιάρης Αντώνιος
Επίκουρος Καθηγητής

ΣΥΝΕΠΙΒΛΕΠΩΝ

Κοζύρη Μαρία
Επίκουρη Καθηγήτρια

Λαμία 2021

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

Implementation and Development of Detailed Placement Algorithms for Integrated Circuits

KRANAS GEORGIOS

FINAL THESIS

ADVISOR

Dadaliaris Antonios
Assistant Professor

CO ADVISOR

Koziri Maria
Assistant Professor

Lamia 2021

«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.

Ημερομηνία: 24/03/2021

Ο Δηλ.

(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

Κατά τη διαδικασία της χωροθέτησης σε μια τυπική ροή σχεδίασης ολοκληρωμένων κυκλωμάτων τα δομικά στοιχεία της πρέπει να τοποθετηθούν εντός μιας ορισμένης περιοχής, στην οποία ενδέχεται να περιέχονται και εμπόδια. Η χωροθέτηση πρέπει να πραγματοποιηθεί με τέτοιο τρόπο ώστε να αποφεύγονται επικαλύψεις και υπερχειλίσεις κελιών, ενώ παράλληλα θα πρέπει να βελτιστοποιείται η υπάρχουσα σχεδίαση βάση κάποιων προκαθορισμένων μετρικών. Αν και στο ερευνητικό πεδίο της Φυσικής Σχεδίασης έχουν αναπτυχθεί και παρουσιαστεί αρκετοί αλγόριθμοι χωροθέτησης, είναι εμφανής η έλλειψη εργαλείων τα οποία μπορούν να βοηθήσουν στην ανάπτυξή τους. Στο πρώτο κομμάτι της εργασίας παρουσιάζεται ένα τέτοιο εργαλείο, το PyPUT (Python Placement Utilities Toolset), μια συλλογή συναρτήσεων με στόχο την υποβοήθηση κατά την υλοποίηση συγκεκριμένων λειτουργιών που χρησιμοποιούνται κατά την ανάπτυξη τέτοιων αλγορίθμων. Στο δεύτερο κομμάτι της εργασίας παρουσιάζεται ένας καινοτόμος γενετικός αλγόριθμος χωροθέτησης ο οποίος δεν επεμβαίνει στα κελιά της σχεδίασης αλλά επανατοποθετεί τα I/O Pads της κατά τέτοιο τρόπο ώστε να βελτιώνεται το συνολικό μήκος καλωδίου διασύνδεσης. Τέλος, παρουσιάζεται ένα ακόμη εργαλείο το οποίο δύναται να χρησιμοποιηθεί κατά την υλοποίηση τόσο του γενετικού αλγορίθμου όσο και γενικότερα τυπικών αλγορίθμων χωροθέτησης και αποτελεί την εξέλιξη του εργαλείου PyPUT.

ABSTRACT

In the process of placement in a typical integrated circuit design flow, its components must be placed within a certain area, which may also contain obstacles. Placement should be done in such a way as to avoid overlaps and overflows of cells, while at the same time the existing design should be optimized based on some predefined metrics. Although several placement algorithms have been developed and presented in the field of Physical Design, the lack of tools that can help in their development is obvious. The first part of this thesis presents such a tool, PyPUT (Python Placement Utilities Toolset), a collection of functions designed to assist in the implementation of specific functions used in the development of such algorithms. The second part of this work presents an innovative genetic algorithm that does not interfere with the design's cells but repositions its I/O Pads in such a way as to improve the overall length of the interconnection cable. In the last chapter, another tool is presented which can be used in the implementation of both the genetic algorithm and generally in standard placement algorithms and is the evolution of PyPUT.

Table of Contents

ΕΙΣΑΓΩΓΗ

Η χωροθέτηση αποτελεί ένα από τα σημαντικότερα βήματα της φυσικής σχεδίασης καθώς επηρεάζει σε μεγάλο βαθμό την ποιότητα του τελικού αποτελέσματος ενός κυκλώματος. Σε μια τυπική ροή χωροθέτησης τα κελιά της σχεδίασης, τοποθετούνται σε μια καλά καθορισμένη ορθογώνια περιοχή, η οποία αποτελεί και την περιοχή-πυρήνα του κυκλώματος. Παράλληλα στόχος της χωροθέτησης είναι η βελτιστοποίηση της υπάρχουσας σχεδίασης βάσης κάποιων μετρικών, με πιο σημαντική αυτή του μήκους καλωδίου που χρησιμοποιείται (ημιπερίμετρος καλωδίου – half perimeter wirelength). Συνήθως στο βήμα της χωροθέτησης τοποθετούνται ή επανατοποθετούνται τα κελιά της σχεδίασης και όχι άλλα δομικά στοιχεία των οποίων η θέση καθορίστηκε από προηγούμενα βήματα της φυσικής σχεδίασης, τα οποία θεωρούνται πλέον σταθερά κι αμετακίνητα.

Οι περισσότεροι αλγόριθμοι χωροθέτησης έχουν ως στόχο τα παραπάνω και λειτουργούν συνήθως με διάφορες παραλλαγές πάνω σε μία ή και παραπάνω φάσεις της χωροθέτησης, την καθολική χωροθέτηση (global placement), την νομιμοποίηση (legalization) και την λεπτομερή χωροθέτηση (detailed placement). Κατά την πρώτη φάση στόχος είναι η διασπορά των κελιών στον χώρο της σχεδίασης εφαρμόζοντας κάποιες τεχνικές βελτιστοποίησης ενώ παρουσιάζονται σφάλματα τα οποία θα εξαλειφθούν κατά τη φάση της νομιμοποίησης. Τέλος κατά την λεπτομερή χωροθέτηση εφαρμόζονται μικρές κινήσεις βελτιστοποίησης ώστε να επιτευχθούν καλύτερα αποτελέσματα μετρικών τιμών.

Κάθε αλγόριθμος χωροθέτησης πριν την εκτέλεση της κύριας λειτουργίας του, πρέπει να περάσει από μια προεργασία κατά την οποία «δημιουργεί» την εικόνα του πρωτότυπου κυκλώματος, στο οποίο θα κάνει κάποιες αλλαγές. Η εικόνα αυτή συνήθως ορίζεται από το πρότυπο βιβλιοθήκης ή bookshelf format κατά το οποίο μια σχεδίαση παρομοιάζεται με βιβλιοθήκη, ενώ τα δομικά της στοιχεία με ορθογώνια (βιβλία) τα οποία τοποθετούνται σε αυτή. Η απόδοση όμως των αρχείων με τα δεδομένα του κυκλώματος σε αυτή την μορφή, αφήνεται στην κρίση του προγραμματιστή καθώς δεν υπάρχει μεγάλη ποικιλία εργαλείων τα οποία να αυτοματοποιούν αυτή τη λειτουργία. Έτσι ο προγραμματιστής σε συνεργασία με τον σχεδιαστή πρέπει να αναπτύξουν σε μορφή κώδικα αυτές τις λειτουργίες καθώς και όσες άλλες κρίνονται απαραίτητες για την ολοκλήρωση του επιθυμητού αλγόριθμου.

Με αφορμή το παραπάνω γεγονός, στο πρώτο κομμάτι της εργασίας παρουσιάζεται ένα εργαλείο, το PyPUT, το οποίο δημιουργήθηκε με σκοπό την αυτοματοποίηση αυτών των λειτουργιών. Στο εργαλείο, το οποίο αναπτύχθηκε σε γλώσσα Python περιέχονται οι απαραίτητες συναρτήσεις οι οποίες σχηματίζουν την προαναφερθείσα «εικόνα» σε διάφορες δομές δεδομένων, ιδανικές για την ανάπτυξη ενός αλγορίθμου χωροθέτησης. Επιπλέον περιέχονται και άλλες λειτουργίες που βοηθούν στην εξέταση αποτελεσμάτων και την αποσφαλμάτωση των αλγορίθμων αυτών, όπως λειτουργίες εκτύπωσης του αποτελέσματος σε εικόνες ή επιστροφή διάφορων μετρικών (πυκνότητα του κυκλώματος, ημιπερίμετρος καλωδίου κ.α.).

Κατά το βήμα του floorplanning, τοποθετούνται στο κύκλωμα τα σημεία εισόδου/εξόδου – I/O Pads, τα οποία θεωρούνται αμετακίνητα κομμάτια της σχεδίασης για τα υπόλοιπα βήματα μιας τυπικής ροής σχεδίασης. Αν και οι περισσότεροι αλγόριθμοι χωροθέτησης προσπαθούν να βελτιστοποιήσουν σχεδιάσεις τηρώντας αυτή τη συνθήκη, υπάρχουν περιπτώσεις στις οποίες αν εφαρμοστούν

κατάλληλα βήματα μπορούν να προκύψουν αρκετά καλά αποτελέσματα. Ένας τέτοιος αλγόριθμος χωροθέτησης παρουσιάζεται στο δεύτερο κομμάτι της εργασίας. Η ιδιαιτερότητά του βέβαια δεν είναι μόνο στη μετακίνηση των I/O Pads, αλλά και στο γεγονός πως αποτελεί έναν γενετικό αλγόριθμο, ο οποίος εκτελείται τηρώντας τα τυπικά βήματα οποιουδήποτε αλγόριθμου της ίδιας κατηγορίας, παρουσιάζοντας δηλαδή τις αλλαγές – βελτιστοποιήσεις με τον ίδιο τρόπο που εμφανίζονται στην φυσική εξέλιξη κάθε βιολογικού οργανισμού.

Η ενασχόληση και η προσπάθεια ανάπτυξης διάφορων μεθοδολογιών ή/και αλγορίθμων χωροθέτησης έκαναν εμφανή τη σημασία ύπαρξης ενός εργαλείου όπως το PyPUT, το οποίο διευκολύνει σημαντικά τον χρήστη παρέχοντας έτοιμες λύσεις οι οποίες θα μπορούσαν να είναι τετριμμένες αλλά ταυτόχρονα χρονοβόρες διαδικασίες που αποσπούν από το στόχο και τη μελέτη του αντικειμένου. Όπως κάθε εργαλείο όμως στην πρώιμη μορφή του, έτσι κι αυτό έχει αρκετά περιθώρια βελτίωσης. Αν και λειτουργικά αποτελεί μια ολοκληρωμένη βιβλιοθήκη συναρτήσεων η οποία εξυπηρετεί το σκοπό της, υστερεί στην τυποποίηση και την ευκολία που συναντάει κανείς σε άλλα μεγαλύτερα πακέτα της γλώσσας Python. Έτσι δημιουργήθηκε ένα εργαλείο το οποίο καλύπτει αυτές τις ελλείψεις και κατά μία έννοια μπορεί να αποτελέσει μια νέα και βελτιωμένη έκδοση του εργαλείου PyPUT με την προσθήκη και την προσαρμογή των είδη υπαρχόντων συναρτήσεων. Το εργαλείο που παρουσιάζεται στο τελευταίο κομμάτι της εργασίας ανήκει στην κατηγορία των parser εργαλείων, και αποτυπώνει το πρότυπο της βιβλιοθήκης σε αντικειμενοστραφή μορφή, απεικονίζοντας μάλιστα όλη τη σχεδίαση με την κλήση μίας μόνο μεθόδου, αποθηκεύοντας όλες τις απαραίτητες πληροφορίες σε μία μόνο μεταβλητή/αντικείμενο. Περιέχει επίσης ένα κομμάτι των συναρτήσεων που παρέχονται από το εργαλείο PyPUT οι οποίες είναι απαραίτητες για τη χρήση και τη λειτουργία του. Επιπρόσθετα η σχεδίασή του το καθιστά εξαιρετικά εύκολο στην εκμάθηση και κατ' επέκταση στην τροποποίησή του, αν το επιθυμεί ο χρήστης.

ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΕΠΙΣΚΟΠΙΣΗ

Η χωροθέτηση είναι σημαντικό κομμάτι της ροής φυσικής σχεδίασης ολοκληρωμένων κυκλωμάτων και αποτελεί ένα επίκαιρο ερευνητικό θέμα με πλήθος εργασιών και δημοσιεύσεων τις τρείς τελευταίες δεκαετίες. Η πλειοψηφία των δημοσιεύσεων αφορά αλγορίθμους χωροθέτησης, οι οποίοι, τυπικά, εμπεριέχουν ένα ή και παραπάνω από τα βήματά της.

Το πρώτο βήμα της χωροθέτησης είναι η καθολική χωροθέτηση (global placement), κατά την οποία, στόχος είναι η μαζική τοποθέτηση των δομικών στοιχείων της σχεδίασης πάνω στην ήδη καθορισμένη περιοχή του πυρήνα της ενώ παράλληλα εφαρμόζονται κινήσεις βελτιστοποίησης του τελικού αποτελέσματος. Υπάρχουν αρκετά παραδείγματα γνωστών αλγορίθμων καθολικής χωροθέτησης οι οποίοι έχουν αποτελέσει πρότυπα για πολλούς άλλους αλγόριθμους και ευριστικές μεθόδους. Μεταξύ άλλων είναι ο Gordian [1], ο SimPL [2], ο FastPlace [3], ο Polar [4] και ο ePlace [5]. Το δεύτερο βήμα είναι αυτό της νομιμοποίησης (legalization), όπου όσα σχεδιαστικά λάθη δημιουργήθηκαν κατά το πρώτο βήμα εξαλείφονται εφαρμόζοντας διάφορες κινήσεις των δομικών στοιχείων. Τα λάθη που μπορούν να εμφανιστούν είναι οι υπερχειλίσεις, δηλαδή η προεξοχή των κελιών εκτός των ορίων της σχεδίασης καθώς και οι αλληλεπικαλύψεις κατά τις οποίες κελιά έχουν τοποθετηθεί το ένα πάνω στο άλλο. Κλασικά παραδείγματα αλγορίθμων νομιμοποίησης είναι οι Tetris [6] και Abacus [7]. Το τρίτο και τελευταίο βήμα είναι η λεπτομερής χωροθέτηση (detailed placement), της οποίας στόχος είναι η επιπλέον βελτιστοποίηση του αποτελέσματος με την εφαρμογή διαφόρων μικρών κινήσεων (γι' αυτό και οι περισσότερες προτάσεις σε αυτή την κατηγορία είναι ευριστικές μέθοδοι είτε κάποιες παραλλαγές αλγορίθμων νομιμοποίησης όπως το [8]).

Οι προαναφερθέντες αλγόριθμοι έχουν ως κοινό σημείο αναφοράς και πειραματισμού πρότυπα κυκλώματα τα οποία ακολουθούν το bookshelf format. Έτσι ως κοινή αρχή, κάθε σχεδιαστής πρέπει να «διαβάσει» και να «αποδώσει» τα περιεχόμενα των αρχείων αυτού του format ώστε να σχηματιστεί μια κατάλληλη εικόνα του κυκλώματος, με βάση την οποία θα εκτελεστεί ο αλγόριθμος. Η έλλειψη της ποικιλίας εργαλείων που αναλαμβάνουν την διαχείριση αυτών των προεργασιών οδήγησε στην υλοποίηση του PyPUT, το οποίο αποτελεί ένα σύνολο συναρτήσεων που διευκολύνει και αυτοματοποιεί τις διαδικασίες αυτές για τον σχεδιαστή, ενώ παράλληλα παρέχει εργαλεία που βοηθούν στην διαδικασία ανάπτυξης των αλγορίθμων χωροθέτησης.

Πριν το βήμα της χωροθέτησης, κατά τη φυσική σχεδίαση, προέχει το βήμα του floorplanning, κατά το οποίο τοποθετούνται στην περιφέρεια της σχεδίασης τα σημεία εισόδου και εξόδου ή αλλιώς I/O Pads, τα οποία κατά το τέλος του βήματος θεωρούνται αμετακίνητα στοιχεία. Συνήθως οι αλγόριθμοι χωροθέτησης λαμβάνουν αυτά τα σημεία ως σταθερές και δεν αλλάζουν την θέση τους, όμως υπάρχουν αρκετές περιπτώσεις στις οποίες η ριζική αλλαγή μιας σχεδίασης είναι αναπόφευκτη, ώστε να υπάρξουν βελτιώσεις μεγαλύτερου βαθμού. Συνήθως τέτοιες αλλαγές εφαρμόζονται σε σχετικά καινούριες τεχνολογίες που θεωρούν πως οι παλιότερες σχεδιάσεις, πλέον, αποτυγχάνουν να ανταπεξέλθουν σε νέες απαιτήσεις απόδοσης και λειτουργικότητας. Οι πιο πρόσφατες προσπάθειες εφαρμογής αλλαγών πάνω στα I/O Pads έχουν ως στόχο τον ίδιο μετρικό στόχο με τους κλασικούς αλγορίθμους

χωροθέτησης, την ελαχιστοποίηση του μήκους καλωδίου – hpwl. Στον αλγόριθμο που παρουσιάζεται στη δημοσίευση [9] ακολουθείται μια μέθοδος γραμμικής χωροθέτησης των I/O pads στην περιφέρεια του πυρήνα, λαμβάνοντας υπόψη τους περιορισμούς της σχεδίασης ως σχέσεις εγγύτητας με τα pads. Στο [10] εφαρμόζεται ένας αλγόριθμος παραλληλισμού επιπέδου εντολής – Instruction Level Parallelism – ILP για τον ορισμό των θέσεων των pads. Στον αλγόριθμο [11] εφαρμόζεται μια διαδικασία δύο φάσεων, η οποία λειτουργεί κάνοντας αλλαγές πάνω σε pads εντός κι εκτός της σχεδίασης, δημιουργώντας κανόνες συσχέτισης μέσω κόμβων, διατηρώντας την αρχική ιεραρχία της σχεδίασης στην πρώτη φάση. Στην δεύτερη υπολογίζονται και εφαρμόζονται οι βέλτιστες θέσεις των pads ώστε να βρίσκονται σε νόμιμες τοποθεσίες ενώ παράλληλα ελαχιστοποιείται το hpwl.

Με παρόμοια λογική σχεδιάστηκε και λειτουργεί ο πρότυπος γενετικός αλγόριθμος που παρουσιάζεται στο κεφάλαιο 3. Στόχος είναι η επανατοποθέτηση των I/O pads στην περιφέρεια της σχεδίασης, μειώνοντας το μήκος καλωδίου, ακολουθώντας τα πρότυπα και των σχεδιασμό των περισσότερων γενετικών αλγορίθμων.

Τέλος στο κεφάλαιο 4, παρουσιάζεται ένας parser κατάλληλα σχεδιασμένος για να λειτουργεί πάνω σε πρότυπα κυκλώματα που ακολουθούνε το bookshelf format. Το συγκεκριμένο εργαλείο χρησιμοποιήθηκε για την ανάπτυξη και τον σχεδιασμό του γενετικού αλγορίθμου του κεφ.3 μαζί με την προσθήκη και την προσαρμογή πάνω σε αυτό συναρτήσεων του εργαλείου PyPUT. Αποτελεί εν δυνάμει μία νέα βελτιωμένη έκδοση του PyPUT αν σε αυτό προστεθούν όλες οι συναρτήσεις του εργαλείου ενώ ακολουθεί το αντικειμενοστραφές πρότυπο προγραμματισμού κάνοντάς το πιο εύκολο στην ανάγνωση, την επέκταση και τη χρήση του από άλλους σχεδιαστές που επιθυμούν να το χρησιμοποιήσουν.

ΚΕΦΑΛΑΙΟ 1: ΧΩΡΟΘΕΤΗΣΗ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ

Φυσική Σχεδίαση

Ορισμός

Η Φυσική Σχεδίαση (Physical Design) [12] αποτελεί ένα βασικό βήμα οποιασδήποτε τυπικής ροής σχεδίασης ολοκληρωμένων κυκλωμάτων (Integrated Circuit Design) και ακολουθεί αυτό της σχεδίασης του κυκλώματος (Circuit Design). Σε αυτό το βήμα, οι αναπαραστάσεις των δομικών στοιχείων της σχεδίασης συσκευές και διασυνδέσεις, που ως τώρα βρίσκονται σε μορφή κυκλώματος, μετατρέπονται σε γεωμετρικά σχήματα, τα οποία, όταν στη συνέχεια κατασκευαστούν στα αντίστοιχα στρώματα υλικών, θα εξασφαλίσουν την εύρυθμη λειτουργία τους. Το βήμα της φυσικής σχεδίασης, συνήθως υποδιαιρείται σε άλλα δευτερεύοντα, που συμπεριλαμβάνουν σχεδιασμό, επαλήθευση και επικύρωση της διάταξης του κυκλώματος τα οποία στη συνέχεια θα υλοποιηθούν μέσω της στρωμάτωσης κατάλληλων υλικών προκειμένου να εξασφαλιστεί η εύρυθμη λειτουργία του κυκλώματος.

Η σχεδίαση ολοκληρωμένων κυκλωμάτων δύναται να χωριστεί σε Front-End Design και Back-End Design. Το Front-End βασίζεται στη χρήση γλωσσών περιγραφής υλικού (Hardware Description Languages – HDLs) και στοχεύει στην επιτυχημένη σύνθεση της σχεδίασης, ενώ το Back-End Design ταυτίζεται με την έννοια της φυσικής σχεδίασης. Τα δεδομένα που εισάγονται στην Φυσική Σχεδίαση για την εκτέλεσή της είναι:

i. Ένα netlist που περιγράφει τα κελιά που συμμετέχουν στην σχεδίαση και τη διασύνδεσή τους.
ii. Μια τεχνολογική βιβλιοθήκη (technology library) για τις βασικές υπομονάδες της σχεδίασης και
iii. ένα αρχείο που ορίζει τους κατασκευαστικούς περιορισμούς.

Η Φυσική Σχεδίαση ολοκληρώνεται με τροποποιήσεις και προσθήκες στην διάταξη του κυκλώματος, μια διαδικασία που είναι γνωστή και ως Layout Post-Processing.

Κάθε μία από τις προαναφερθείσες φάσεις έχει σχετικές ροές σχεδιασμού, οι οποίες καθορίζουν την διαδικασία που επιτελούν καθώς και το πλαίσιο λειτουργίας τους. Η ροή της φυσικής σχεδίασης χρησιμοποιεί τις τεχνικές πληροφορίες που βρίσκονται στις παραπάνω τεχνολογικές βιβλιοθήκες και παρέχονται από τους κατασκευαστές. Οι πληροφορίες αυτές αφορούν τον τύπο του κομματιού πυριτίου (silicon wafer) που χρησιμοποιείται για την κατασκευή, τα κελιά (cells) της σχεδίασης, καθώς και τους κανόνες διάταξης.

Τυπικά, η Φυσική Σχεδίαση κατηγοριοποιείται σε πλήρως-προσαρμόσιμη (full-custom) και μερικώς προσαρμόσιμη (semi-custom) σχεδίαση. Στην πρώτη ο σχεδιαστής έχει πλήρη ελευθερία στην σχεδίαση της διάταξης, ενώ δεν χρησιμοποιούνται προκαθορισμένα κελιά. Στην δεύτερη, χρησιμοποιούνται προκαθορισμένα κελιά, ορισμένα από βιβλιοθήκες, ενώ ο σχεδιαστής έχει ευελιξία

στην τοποθέτηση (placement) των κελιών και στη δρομολόγηση (routing). Για πλήρως-προσαρμόσιμες σχεδιάσεις χρησιμοποιούνται ολοκληρωμένα κυκλώματα ειδικής εφαρμογής (Application Specific Integrated Circuit - ASIC) και για μερικώς-προσαρμόσιμες γίνεται χρήση επαναπρογραμματιζόμενων μονάδων υλικού (Field Programmable Gate Arrays - FPGAs). Αυτό γίνεται διότι στα ASIC δεν υπάρχει περιορισμός στον σχεδιασμό και την τροποποίηση των μπλοκ σχεδίασης από τους προμηθευτές ενώ αυτή η ελευθερία κινήσεων λείπει από τα FPGAs.

Ροή Φυσικής Σχεδίασης ASIC

Τα βήματα της ροής φυσικής σχεδίασης ASIC είναι:

- Design Netlist.
- Floorplanning.
- Partitioning.
- Placement.
- Clock-tree Synthesis (CTS).
- Routing.
- Physical Verification.
- Layout Post Processing with Mask Data Generation.

Design Netlist

Η Φυσική Σχεδίαση είναι βασισμένη σε ένα netlist το οποίο είναι το τελικό παράγωγο της διαδικασίας της σύνθεσης. Η σύνθεση μετατρέπει τον σχεδιασμό επιπέδου register-transfer (RTL), που αποτελεί μια αφαιρετική σχεδίαση της λογικής του κυκλώματος και συνήθως περιγράφεται με γλώσσες περιγραφής υλικού (π.χ. VHDL, Verilog) σε περιγραφή επιπέδου πυλών που μπορεί να γίνει κατανοητή από διάφορα εργαλεία σύνθεσης. Το netlist περιέχει πληροφορίες σχετικά με τα κελιά που χρησιμοποιούνται, την μεταξύ τους διασύνδεση, τον χώρο που καταλαμβάνουν κι άλλες λεπτομέρειες. Η διαδικασία της σύνθεσης υπόκειται σε περιορισμούς προκειμένου να διασφαλιστούν οι προδιαγραφές της σχεδίασης (λειτουργικότητα, ταχύτητα κ.λ.π.). Το netlist προωθείται σε επόμενα στάδια της ροής φυσικής σχεδίασης μόνο αφού έχει επαληθευτεί η λειτουργικότητά του και η συμμόρφωσή του ως προς τους προαναφερθέντες περιορισμούς.

Βήματα Φυσικής Σχεδίασης: 1. Partitioning

Ο διαμερισμός (partitioning) είναι η διαδικασία διαίρεσης του chip σε μικρότερα μπλοκ. Αυτό γίνεται ώστε να διαχωρίζονται μεταξύ τους τα μπλοκ που έχουν διαφορετική λειτουργία και να γίνεται πιο εύκολα η χωροθέτηση (placement) και η δρομολόγηση (routing). Το partitioning μπορεί να γίνει κατά το σχεδιασμό RTL όταν ο σχεδιαστής χωρίζει τη σχεδίαση σε υπο-μπλοκ και σχεδιάζει κάθε μονάδα της. Αυτές οι μονάδες συνδέονται μεταξύ τους σε μία κύρια που ονομάζεται Μονάδα Υψηλού Επιπέδου (Top Level Module). Αυτό το είδος partitioning αναφέρεται και ως Λογικός Διαμερισμός (Logical Partitioning). Στόχος αυτού του βήματος είναι η διαίρεση τους κυκλώματος ώστε να ελαχιστοποιείται ο αριθμός των συνδέσεων μεταξύ των διαμερισμάτων (partitions).

Βήματα Φυσικής Σχεδίασης: 2. Floorplanning

Το δεύτερο βήμα της φυσικής σχεδίασης είναι το floorplanning. Κατά το floorplanning γίνεται εντοπισμός στοιχείων που πρέπει να τοποθετηθούν κοντά καθώς και η κατανομή χώρου για αυτά. Η κατανομή γίνεται με τέτοιο τρόπο ώστε να λαμβάνονται υπόψη διάφορες συγκρούσεις για το διαθέσιμο χώρο, η απαιτούμενη απόδοση και σαφώς η επιθυμία τα στοιχεία να βρίσκονται όσο το δυνατόν πιο κοντά μεταξύ τους.

Το floorplanning λαμβάνει υπόψη ακίνητα δομικά στοιχεία (macros) της σχεδίασης, την μνήμη, πυρήνες IP (semiconductor Intellectual Property cores) και τους περιορισμούς τοποθέτησής τους, τις δυνατότητες δρομολόγησης κι επίσης την περιοχή ολόκληρης της σχεδίασης ενώ καθορίζει τη δομή εισόδων-εξόδων (Input/Output, I/O) και την αναλογία διαστάσεων του σχεδιασμού. Ένα κακό floorplan οδηγεί σε σπατάλη περιοχής και σε πιθανή συμφόρηση (routing congestion).

Σε πολλές μεθοδολογίες σχεδιασμού, η περιοχή και η ταχύτητα αποτελούν δύο αντιστρόφως ανάλογα μεγέθη. Αυτό οφείλεται στους περιορισμένους πόρους δρομολόγησης, αφού όσο περισσότεροι πόροι χρησιμοποιούνται τόσο πιο αργή γίνεται η λειτουργία. Η προσπάθεια βελτιστοποίησης της περιοχής με σκοπό την ελαχιστοποίησή της επιτρέπει στη σχεδίαση να κάνει χρήση λιγότερων πόρων και σε μεγαλύτερη γειτνίαση των τμημάτων της. Αυτό οδηγεί σε μικρότερες αποστάσεις διασύνδεσης, μικρότερη χρήση των αντίστοιχων πόρων, γρηγορότερες διαδρομές σημάτων από άκρο σε άκρο και ακόμη πιο αξιόπιστους χρόνους θέσης και διαδρομής. Ένα σωστό floorplanning μπορεί να οδηγήσει στην εξάλειψη των παραπάνω προβλημάτων.

Γενικά, τα τμήματα της σχεδίασης που επωφελούνται περισσότερο από αυτό το βήμα, είναι η διαδρομή δεδομένων (datapath), ενώ οι υπομονάδες random-logic (λογικές πύλες κ.α.), οι μηχανές καταστάσεων και άλλες μη δομημένες μονάδες λογικής μπορούν να αφεθούν στο λογισμικό χωροθέτησης και δρομολόγησης.

Οι διαδρομές δεδομένων είναι συνήθως οι περιοχές της σχεδίασης όπου γίνεται παράλληλη επεξεργασία πολλαπλών bits που τροποποιούνται με τον ίδιο τρόπο με την πιθανότητα να επηρεάζονται από γειτονικά bits. Τυπικά παραδείγματα τέτοιων μονάδων είναι οι αθροιστές, οι αφαιρέτες, οι μετρητές, οι καταχωρητές και οι πολυπλέκτες.

Βήματα Φυσικής Σχεδίασης: 3. Placement

Πριν την έναρξη της βελτιστοποίησης χωροθέτησης, όλα τα μοντέλα φόρτου καλωδίων (Wire Load Models) αφαιρούνται. Γίνεται χρήση τιμών RC από την VR (Virtual Router) για τον υπολογισμό του χρονισμού. Η VR είναι η μικρότερη απόσταση μεταξύ δύο pins (απόσταση Manhattan). Οι τιμές RC από την VR είναι πιο ακριβείς από αυτές ενός WLM.
Το placement πραγματοποιείται σε τέσσερις φάσεις βελτιστοποίησης:

1. Pre-placement βελτιστοποίηση.
2. Βελτιστοποίηση εντός placement.
3. Βελτιστοποίηση post-placement (PPO), πριν γίνει το βήμα Clock Tree Synthesis.
4. PPO βελτιστοποίηση μετά το βήμα CTS.

- Η PPO βελτιστοποιεί το netlist πριν το βήμα του placement, τα High Fanout Nets συμπτύσσονται, ενώ μπορεί να μειώσεις επίσης το μέγεθος των κελιών.
- Στη βελτιστοποίηση εντός placement, γίνεται ξανά βελτιστοποίηση της λογικής βασισμένης στο VR. Εδώ μπορεί να γίνουν διάφορες ενέργειες όπως, αλλαγή μεγέθους κελιών, μετακίνηση και παράκαμψη κελιών, διαχωρισμός πλέγματος (net), διπλασιασμός πυλών, εισαγωγή buffer, και ανάκτηση περιοχής. Η βελτιστοποίηση κάνει επαναλήψεις καθορισμού ρυθμίσεων, σταδιακού χρονισμού και placement με βάση τη συμφόρηση (congestion).
- Η PPO πριν το CTS βελτιστοποιεί το netlist χρησιμοποιώντας ιδανικά ρολόγια. Μπορεί να διορθώσει παραβάσεις ρυθμίσεων, αναμονής και παραβάσεις max trans/cap, ενώ επαναλαμβάνει την HFN σύνθεση.
- Η PPO μετά το CTS βελτιστοποιεί τον χρονισμό με τη χρήση μη-ιδεατού ρολογιού διάδοσης (propagated clock), προσπαθώντας να διατηρήσει τη διαφορά χρονισμού (clock skew), το φαινόμενο δηλαδή όπου ένα σήμα από το ίδιο ρολόι φτάνει σε διαφορετικά συστατικά του chip σε διαφορετικέ χρονικές στιγμές.

Βήματα Φυσικής Σχεδίασης: 3. Clock tree synthesis (CTS)

Η διαδικασία του Clock Tree Synthesis αφορά την υλοποίηση των κατάλληλων διασυνδέσεων που αφορούν τα σήματα ρολογιού, προκειμένου αυτά να διαμοιραστούν στα δομικά στοιχεία όπου απαιτούνται.

Ο απώτερος στόχος της διαδικασίας είναι η αποφυγή συμφόρησης των συνδέσεων και η ελαχιστοποίηση του clock skew και των καθυστερήσεων εισόδου.

Βήματα Φυσικής Σχεδίασης: 4. Routing

Υπάρχουν δύο τρόποι δρομολόγησης στο physical design, η καθολική δρομολόγηση (global routing) και η λεπτομερής δρομολόγηση (detailed routing). Η πρώτη διαθέτει πόρους δρομολόγησης που χρησιμοποιούνται για τις συνδέσεις ενώ επίσης παρακολουθεί την ανάθεση για ένα συγκεκριμένο net.

Η λεπτομερής δρομολόγηση πραγματοποιεί τις πραγματικές συνδέσεις. Διάφοροι περιορισμοί λαμβάνονται υπόψη, μεταξύ άλλων και οι κανόνες σχεδίασης Design Rule Check - DRC, το μήκος καλωδίου και ο χρονισμός.

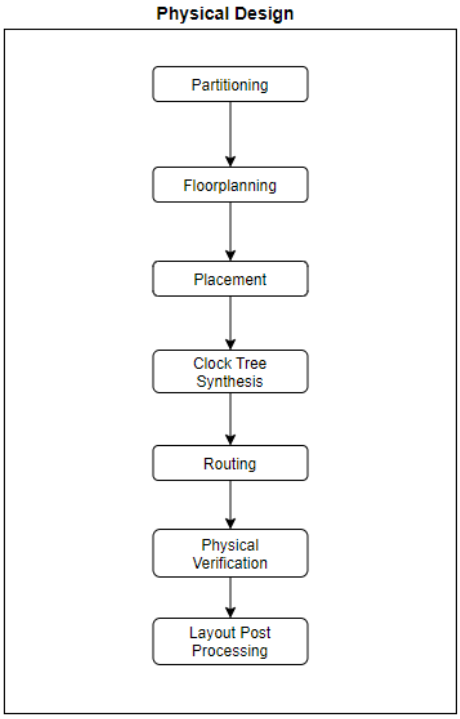
Βήματα Φυσικής Σχεδίασης: 5. Physical Verification

Το physical verification επαληθεύει την ορθότητα της σχεδίασης που παράχθηκε. Αυτό σημαίνει πως η διάταξη:

- Είναι συμβατή με όλες τις απαιτήσεις της τεχνολογίας – Design Rule Checking (DRC).
- Είναι σύμφωνη με το αρχικό netlist – Layout vs Schematic (LVS).
- Επαληθεύεται η πυκνότητα σε επίπεδο chip.
- Συμμορφώνεται με όλες τις ηλεκτρικές απαιτήσεις – Electrical Rule Checking (ERC).

Βήματα Φυσικής Σχεδίασης: 6. Layout Post Processing

Το Layout Post Processing, επίσης γνωστό με την ονομασία προετοιμασία δεδομένων μάσκας (mask data preparation), ολοκληρώνει συχνά τη Φυσική Σχεδίαση και την επαλήθευσή της. Μετατρέπει τη φυσική διάταξη (πολύγωνα) σε δεδομένα μάσκας, δηλαδή πραγματικά στοιχεία έτοιμα για εκτύπωση σε γκοφρέτες πυριτίου.



Εικόνα 1: Βήματα Physical Design

Placement - Χωροθέτηση

Ορισμός

Η Χωροθέτηση (Placement) [13] είναι ένα από τα βασικά βήματα στην αυτοματοποίηση της ηλεκτρονικής σχεδίασης, στο κομμάτι της ροής της Φυσικής Σχεδίασης και αναλαμβάνει την ανάθεση ακριβών τοποθεσιών για διάφορα δομικά στοιχεία του κυκλώματος, εντός της περιοχής του πυρήνα του chip. Ένα κακό Placement δεν θα επηρεάσει απλά την απόδοση του chip, αλλά μπορεί και να το καταστήσει μη-κατασκευάσιμο παράγοντας περιττό μήκος καλωδίου (wirelength), πέρα της δυνατότητας των διαθέσιμων πόρων Δρομολόγησης. Κατά συνέπεια, ένας placer, θα πρέπει να κάνει την ανάθεση ενώ παράλληλα εφαρμόζει μεθόδους βελτιστοποίησης σε ένα αριθμό στόχων ώστε να διασφαλίσει ότι το κύκλωμα συναντά τις απαιτήσεις απόδοσης που έχουν τεθεί. Τα βήματα του Placement και Routing μαζί είναι γνωστά και ως place & route (P&R) στον σχεδιασμό IC.

Ένας placer δέχεται ως είσοδο το netlist του κυκλώματος που έχει ήδη περάσει την διαδικασία της σύνθεσης, μαζί με μια βιβλιοθήκη τεχνολογίας (technology library), και παράγει μια έγκυρη διάταξη. Η διάταξη είναι βελτιστοποιημένη σύμφωνα με τους στόχους που προαναφέρθηκαν και είναι έτοιμη για πιθανές αλλαγές του μεγέθους των κελιών και την εισαγωγή buffers, βήματα απαραίτητα για την επίτευξη του χρονισμού και την ακεραιότητα του σήματος. Ακολουθούν τα βήματα του Clock-Tree Synthesis και του Routing, συμπληρώνοντας έτσι τη διαδικασία του Physical Design. Σε πολλές περιπτώσεις, τμήματα ή ολόκληρη η ροή του Physical Design επαναλαμβάνονται αρκετές φορές ώστε να επιτευχθεί η ολοκλήρωση της σχεδίασης.

Στην περίπτωση των ολοκληρωμένων κυκλωμάτων για συγκεκριμένες εφαρμογές (ASIC), η περιοχή της διάταξης καταλαμβάνει έναν αριθμό σειρών (rows), σταθερού ύψους, με κάποιο ή καθόλου κενό μεταξύ τους. Κάθε σειρά αποτελείται από έναν αριθμό τοποθεσιών που μπορούν να καταληφθούν από τα στοιχεία του κυκλώματος. Μια ελεύθερη τοποθεσία είναι αυτή που δεν καταλαμβάνεται από κανένα τέτοιο στοιχεία. Τα στοιχεία ενός κυκλώματος είναι είτε κανονικά κελιά (cells), τερματικά κελιά (macro cells), ή I/O pads (pins). Τα cells έχουν ένα προκαθορισμένο ύψος που δεν αλλάζει, ίσο με το ύψος των γραμμών, αλλά έχουν πλάτος που διαφέρει. Το πλάτος ενός cell είναι στην ουσία ένας ακέραιος αριθμός από τοποθεσίες. Σε αντίθεση με τα cells, τα macro blocks έχουν ύψος που διαφέρει και μπορεί να εκτείνονται σε ένα πλήθος γραμμών. Κάποια blocks έχουν προκαθορισμένες τοποθεσίες στη σχεδίαση, συνήθως παράγωγο από το προηγούμενο βήμα Floorplanning, γεγονός που περιορίζει την διεργασία του placer στην ανάθεση τοποθεσιών μόνο για τα cells. Σε αυτή την περίπτωση, τα block παίρνουν το προσωνύμιο προκαθορισμένα (fixed blocks). Ειδάλλως, κάποια ή/και όλα τα macro blocks δεν έχουν προκαθορισμένες θέσεις. Εδώ, θα πρέπει να τοποθετηθούν μαζί με τα cells σε μια διαδικασία που ονομάζεται μικτή χωροθέτηση (mixed placement).

Στόχοι και Περιορισμοί

Το Placement συνήθως διατυπώνεται ως πρόβλημα βελτιστοποίησης με περιορισμούς (constraint optimization problem). Οι μετρικές που δύναται να βελτιστοποιηθούν είναι οι ακόλουθες:

- Total wirelength – Συνολικό μήκος καλωδίου διασύνδεσης: Η ελαχιστοποίηση του συνολικού μήκους όλων των καλωδίων της σχεδίασης, είναι ο συνηθέστερος στόχος σχεδόν όλων των βιομηχανικών και ακαδημαϊκών placers. Ο περιορισμός του καλωδίου διασύνδεσης ελαττώνει το κόστος υλοποίησης, την καταναλισκόμενη ισχύ και τις επιμέρους καθυστερήσεις (μεγέθη ανάλογα του μήκους καλωδίου).

- Timing – Χρονισμός: Ο κύκλος ρολογιού ενός chip καθορίζεται από την καθυστέρηση του μεγαλύτερου μονοπατιού-διαδρομής, που συνήθως αναφέρεται ως κρίσιμη διαδρομή (critical path). Δεδομένων των προδιαγραφών απόδοσης, ένας placer πρέπει να διασφαλίζει ότι δεν υπάρχει μονοπάτι με καθυστέρηση μεγαλύτερη της μέγιστης που αναφέρεται στις προδιαγραφές.

- Congestion – Συμφόρηση: Αν και είναι απαραίτητο να ελαχιστοποιηθεί το συνολικό μήκος καλωδίου, ώστε οι συνολικοί πόροι δρομολόγησης να είναι επαρκείς, είναι επίσης απαραίτητο να κατανεμηθούν σωστά οι πόροι και εντός των διάφορων τοπικών περιοχών του chip. Μια περιοχή στην οποία παρουσιάζεται συμφόρηση μπορεί να οδηγήσει σε υπερβολικές παρακάμψεις δρομολόγησης, ή μπορεί να καταστήσει αδύνατη την ολοκλήρωση όλων των διαδρομών.

- Power – Ισχύς: Η ελαχιστοποίηση ισχύος περιλαμβάνει συνήθως τη διανομή θέσεων των cells έτσι ώστε να μειωθεί η συνολική κατανάλωση ενέργειας, να αραιώσουν τα hot spots, που είναι σημεία με μεγάλη πυκνότητα δομικών στοιχείων, και να επιτευχθούν ομαλές θερμοκρασίες λειτουργίας στη σχεδίαση.

- Ένας δευτερεύον στόχος είναι η μείωση του χρόνου εκτέλεσης της διαδικασίας του placement.

Βήματα και βασικές τεχνικές Χωροθέτησης

Η διαδικασία του Placement μπορεί να διαχωριστεί σε τρία επιμέρους βήματα, την καθολική χωροθέτηση (global placement), την λεπτομερή χωροθέτηση (detailed placement), και το ενδιάμεσο βήμα της νομιμοποίησης (legalization). Η καθολική χωροθέτηση δημιουργεί σημαντικές αλλαγές, βάζοντας όλα τα δομικά στοιχεία σε κατάλληλες/ιδανικές θέσεις, σε καθολική κλίμακα, επιτρέποντας επικαλύψεις μεταξύ των στοιχείων, σε μικρό βαθμό. Κατά τη νομιμοποίηση οι διάφορες ανομίες της σχεδίασης που παραβαίνουν τους περιορισμούς που έχουν τεθεί, όπως οι επικαλύψεις που αναφέρθηκαν, εξαλείφονται, μέσω της εκτέλεσης διαφόρων διαδικασιών. Η λεπτομερής χωροθέτηση μετατοπίζει κάθε δομικό στοιχείο της σχεδίασης σε κοντινή νόμιμη τοποθεσία, δηλαδή τοποθεσία που υπακούει στους περιορισμούς της σχεδίασης, έχοντας ως στόχο την βελτιστοποίηση της σχεδίασης, προσπαθώντας να ικανοποιήσει τους προαναφερθέντες στόχους. Η χωροθέτηση και η συνολική ποιότητα σχεδιασμού εξαρτώνται κυρίως από την απόδοση της καθολικής χωροθέτησης.

Σε πρώιμα στάδια, η διαδικασία της χωροθέτησης των ολοκληρωμένων κυκλωμάτων γινόταν με τη χρήση διάφορων συνδυαστικών μεθόδων. Όταν ο σχεδιασμός ήταν της κλίμακας χιλίων-πυλών, την καλύτερη απόδοση και λύση έδιναν μεθοδολογίες προσομοιωμένης ανόπτησης (simulated annealing) όπως ο αλγόριθμος TimberWolf [14]. Καθώς ο σχεδιασμός IC έφτασε στην κλίμακα των εκατομμυρίων-πυλών, για την επιτυχή χωροθέτηση χρησιμοποιήθηκαν αλγόριθμοι επαναληπτικού διαχωρισμού υπερ-γράφου (hyper-graph partitioning), όπως ο Capo [15].

Ακολούθως, το quadratic placement ξεπέρασε τις συνδυαστικές λύσεις τόσο ως προς την ποιότητα του τελικού αποτελέσματος όσο και ως προς τη σταθερότητα στην αντιμετώπιση διαφορετικών περιπτώσεων. Χαρακτηριστικό παράδειγμα της κατηγορίας αυτής αποτελεί ο αλγόριθμος Gordian [1] ο οποίος διαμορφώνει το κόστος μήκους καλωδίου ως τετραγωνική συνάρτηση ενώ εξακολουθεί να απομακρύνει τα cells μεταξύ τους χρησιμοποιώντας επαναληπτική χωροθέτηση (recursive partitioning). Ο αλγόριθμος [16] μοντελοποιεί την πυκνότητα χωροθέτησης ως γραμμικό όρο στην τετραγωνική συνάρτηση κόστους και λύνει το πρόβλημα της χωροθέτησης με καθαρό τετραγωνικό προγραμματισμό (quadratic programming). Η πλειοψηφία των μοντέρνων quadratic placers (KraftWerk [17], FastPlace [3], SimPL [2]) ακολουθούν αυτό το πλαίσιο, ο καθένας με διαφορετικές ευριστικές μεθόδους για τον προσδιορισμό της δύναμης της γραμμικής πυκνότητας (linear density force).

Η μη γραμμική χωροθέτηση παρουσιάζει καλύτερη απόδοση σε σχέση με τις άλλες κατηγορίες αλγορίθμων. Η προσέγγιση στον αλγόριθμο [18] πρώτα μοντελοποιεί το μήκος καλωδίου με εκθετικές συναρτήσεις και την πυκνότητα με τοπικές τετραγωνικές συναρτήσεις πάνω στα δομικά συστατικά της σχεδίασης, έτσι ώστε να βελτιώσει την ακρίβεια και κατ' επέκταση την ποιότητα. Άλλες ακαδημαϊκές μελέτες περιλαμβάνουν κυρίως τους APlace [19] και NTUPlace [20]. Ο ePlace [5] είναι ένας προηγμένος αλγόριθμος καθολικής τοποθέτησης. Ξεχωρίζει στιγμιότυπα μεταξύ τους, δημιουργώντας ένας ηλεκτροστατικό πεδίο, το οποίο εγκαθιδρύει την έννοια της ελάχιστης ποιότητας επιβάρυνσης (minimum quality overhead), επιτυγχάνοντας έτσι την καλύτερη απόδοση.

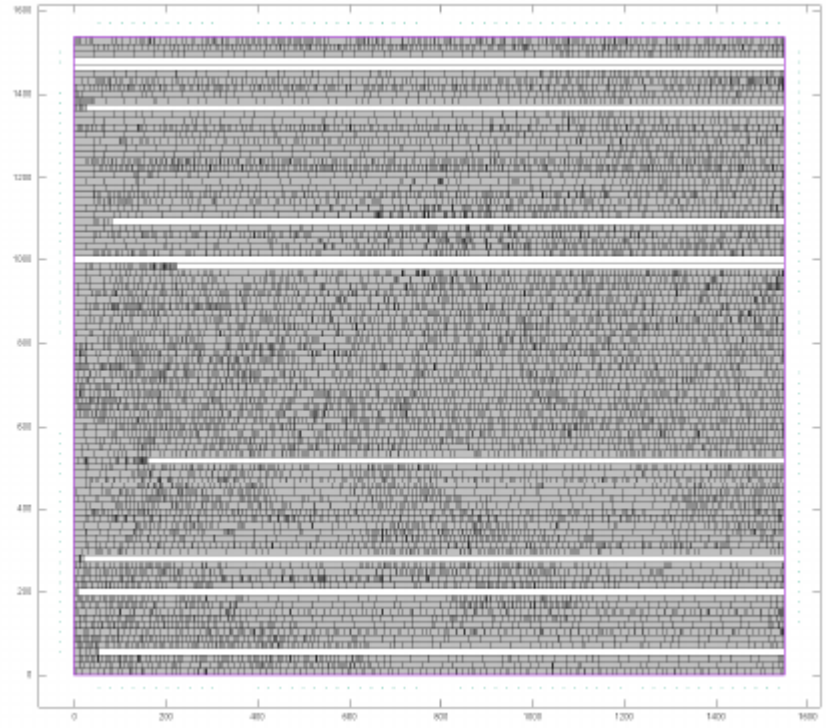
ΚΕΦΑΛΑΙΟ 2: PyPUT, Python-based Placement Utilities Toolset

Εισαγωγή

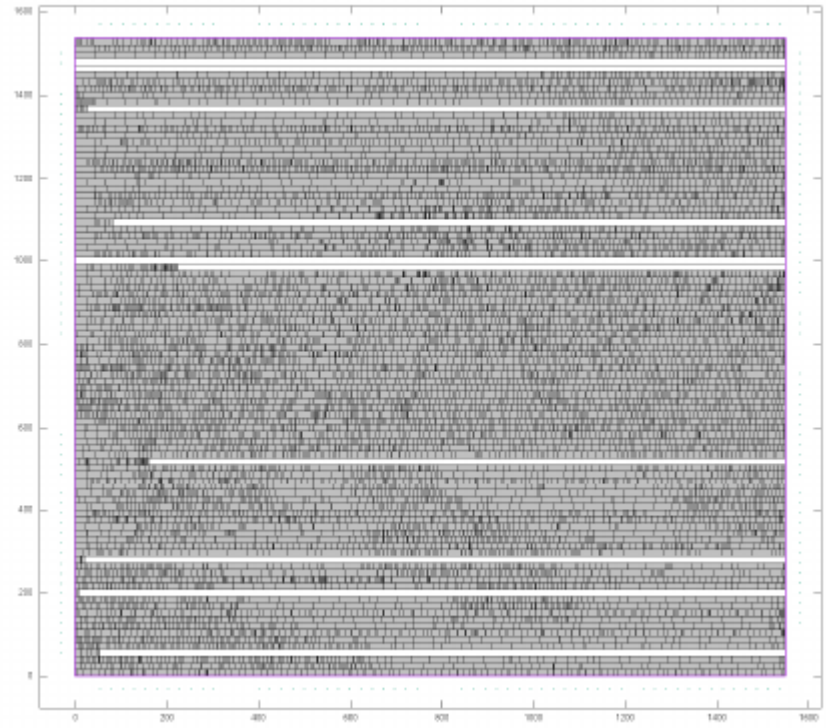
Η Χωροθέτηση μπορεί να οριστεί ως το πρόβλημα της τοποθέτησης των κελιών (cells) μιας σχεδίασης σε μία ορισμένη περιοχή, την περιοχή του chip. Η τελική μορφή της σχεδίασης δεν θα πρέπει να περιέχει επικαλύψεις (overlaps) κελιών μεταξύ τους, ούτε και υπερχειλίσεις (overflows) εκτός του κυκλώματος και η συνάρτηση η οποία εξετάζει την ποιότητα της σχεδίασης που έχει παραχθεί θα πρέπει να βελτιστοποιείται. Σε μία τυπική Χωροθέτηση, όλα τα κελιά θα πρέπει να έχουν το ίδιο ύψος και πλάτος που διαφέρει, η περιοχή του chip χωρίζεται σε σειρές ίδιου ύψους ίσου με αυτού των κελιών, και μπορεί να περιέχει εμπόδια ή φραγμένες περιοχές (fence regions).

Ο διαχωρισμός της διαδικασίας της χωροθέτησης γίνεται συνήθως σε τρείς φάσεις, το Global Placement, Legalization και Detailed Placement, που αναφέρθηκαν στο προηγούμενο κεφάλαιο.

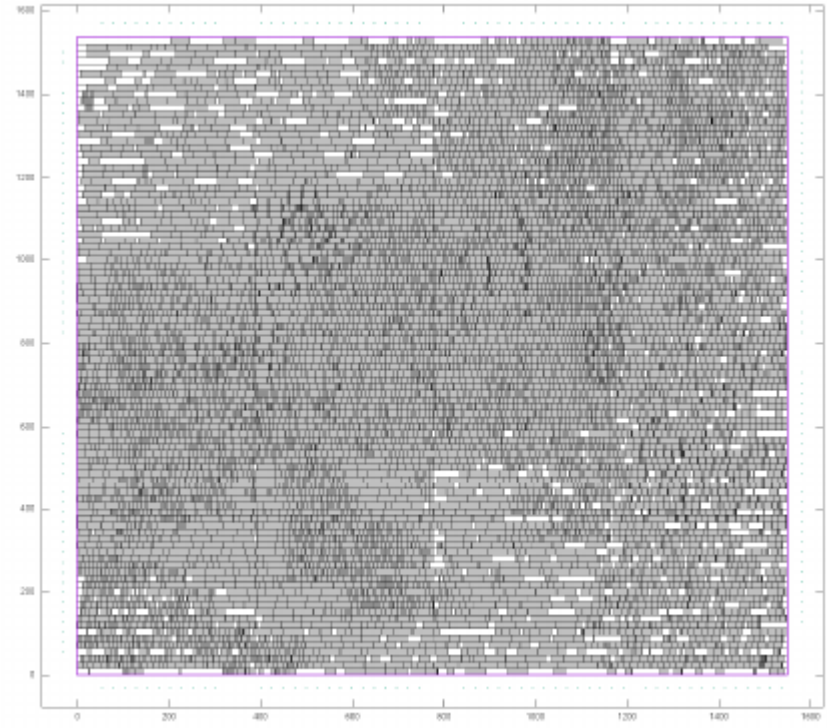
Στο global placement ή καθολική χωροθέτηση, τα κελιά διαμοιράζονται σε όλη την περιοχή του chip ώστε ένας ή περισσότεροι στόχοι να βελτιστοποιούνται (συνήθως το συνολικό μήκος καλωδίου των συνδέσεων, η πυκνότητα του chip ή ο χρονισμός. Αν τα κελιά τοποθετηθούν στις καλύτερες δυνατές θέσεις πάνω στο chip τότε η σχεδίαση θα είναι λάθος καθώς θα περιέχει υπερκαλύψεις μεταξύ των κελιών ή υπερχειλίσεις κι έτσι κρίνεται απαραίτητη η επόμενη φάση της νομιμοποίησης του κυκλώματος (Legalization). Κατά τη φάση της νομιμοποίησης ή legalization οι υπερκαλύψεις και οι υπερχειλίσεις εξαλείφονται και τα κελιά τοποθετούνται σωστά πλέον, μέσα στις γραμμές του chip. Στο τελικό βήμα του Detailed Placement, πραγματοποιούνται μικρές αλλαγές που βελτιώνουν τα μετρικά που αναφέρθηκαν. Στις παρακάτω εικόνες φαίνονται, το αποτέλεσμα της φάσης global placement, του legalization step και το τελικό αποτέλεσμα μετά το detailed placement.



Εικόνα 2: Global Placement instance



Εικόνα 3: Legalization instance



Εικόνα 4: Detailed Placement instance

Η Χωροθέτηση έχει συγκεντρώσει αρκετό ερευνητικό ενδιαφέρον στο παρελθόν, με πολλές προσπάθειες να έχουν δημοσιευθεί σχετικά με καθεμία από τις παραπάνω φάσεις. Συγκεκριμένα για το global placement, υπάρχουν προσεγγίσεις βασισμένες σε μορφοποίηση του προβλήματος σε τετραγωνικές εξισώσεις (quadratic formulation). Τέτοιες δημοσιεύσεις είναι οι, Gordian [1], SimPL [2], FastPlace[3] και Polar[4], ενώ οι Kraftwerk [17] και ePlace [5] βασίζονται σε force directed μεθόδους. Στο βήμα του legalization υπάρχουν οι αλγόριθμοι Tetris [6] και Abacus [7] οι οποίοι παραμένουν να είναι αποδοτικοί σε οποιαδήποτε ροή placement με βάση το χρόνο εκτέλεσης. Τέλος σχετικά με το detailed placement εφαρμόζονται κυρίως ευριστικές μέθοδοι και διαφοροποιήσεις των αλγορίθμων legalization.

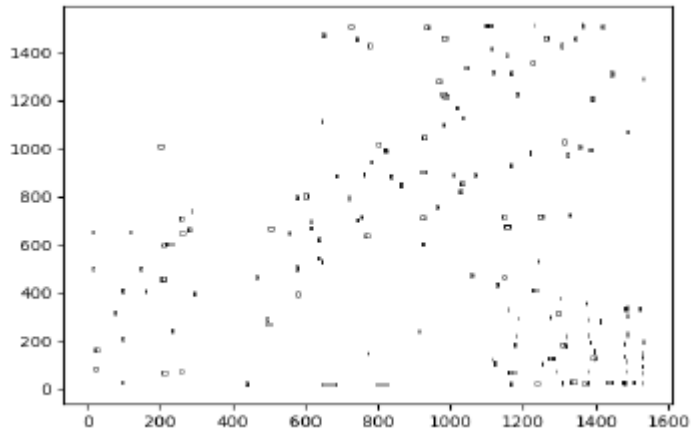
PyPUT

Το εργαλείο στοχεύει στα ISPD πρότυπα (benchmark) κυκλώματα [21][22], τα οποία χρησιμοποιούνται ευρέως σε δοκιμές ώστε να αποδειχθεί η αποδοτικότητα διαφόρων μεθόδων placement. Στόχος του εργαλείου είναι να παρέχει αυτοματισμούς που βοηθούν στην ανάπτυξη και την εφαρμογή αλγορίθμων σχετικών με το placement.

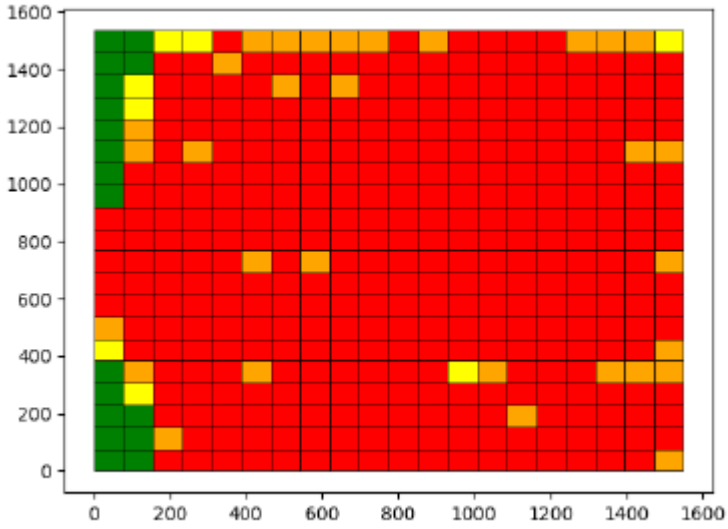
Εκτός των βασικών βιβλιοθηκών της Python, ένα πλήθος αρκετά γνωστών πακέτων χρησιμοποιήθηκαν ώστε το εργαλείο να μπορεί να λειτουργεί σε ήδη υπάρχουσες καθώς και σε μελλοντικές εκδόσεις της γλώσσας (backwards & forwards compatibility), παράλληλα κάνοντάς το πιο σταθερό και πιο εύκολα επεκτάσιμο (scalability). Οι συναρτήσεις που έχουν υλοποιηθεί και είναι κομμάτι του εργαλείου αποτελούν άμεσο παράγωγο του διαχωρισμού των βημάτων που αναφέρονται πιο πάνω και μπορούν να κατηγοριοποιηθούν στις εξής κατηγορίες.

- Συγκριτικές – Comparative: Αποτελείται από συναρτήσεις που μπορούν να χρησιμοποιηθούν για να συγκρίνουν δύο ή περισσότερες προσεγγίσεις placement με βάση τα μετρικά που χρησιμοποιούνται συχνότερα, όπως ο αριθμός των υπερχειλίσεων, των αριθμό των υπερκαλύψεων το άθροισμα της ημιπεριμέτρου καλωδίου (total half perimeter wirelength) και η πυκνότητα.
- Υπολογιστικές – Computational: Σε αυτή την κατηγορία υπάρχουν συναρτήσεις που επιτελούν διάφορες υπολογιστικές διαδικασίες όπως ο υπολογισμός της ημιπεριμέτρου καλωδίου, της πυκνότητας όλης της σχεδίασης είτε μικρότερου μέρους αυτής καθορισμένης από το χρήστη.
- Αναλυτικές – Parsing: Η κατηγορία parsing είναι ιδιαιτέρως χρήσιμη καθώς κάθε προσπάθεια στην προσέγγιση λύσης του προβλήματος της χωροθέτησης ξεκινάει με την εξαγωγή των απαραίτητων δεδομένων από τα αρχεία της πρότυπης σχεδίασης (benchmark). Έχουν υλοποιηθεί πολλαπλές εκδόσεις κάθε συνάρτησης ώστε να παράγονται τα δεδομένα με διαφορετικό τρόπο και να καλύπτεται ένα ευρύτερο φάσμα αναγκών του χρήστη.
- Σχεδιαστικές – Plotting: Το εργαλείο παρέχει μεταξύ άλλων, και συναρτήσεις σχεδίασης. Τα σχέδια μπορούν να χρησιμοποιηθούν για πολλούς λόγους και μπορούν παραχθούν από διάφορα εργαλεία. Οι συναρτήσεις του PyPUT

στοχεύουν σε λεπτομέρειες της σχεδίασης ώστε να παρέχεται στον χρήστη μια πιο καθαρή εικόνα αυτής και περισσότερη λεπτομέρεια.



Εικόνα 5: Υπερκαλύψεις της σχεδίασης



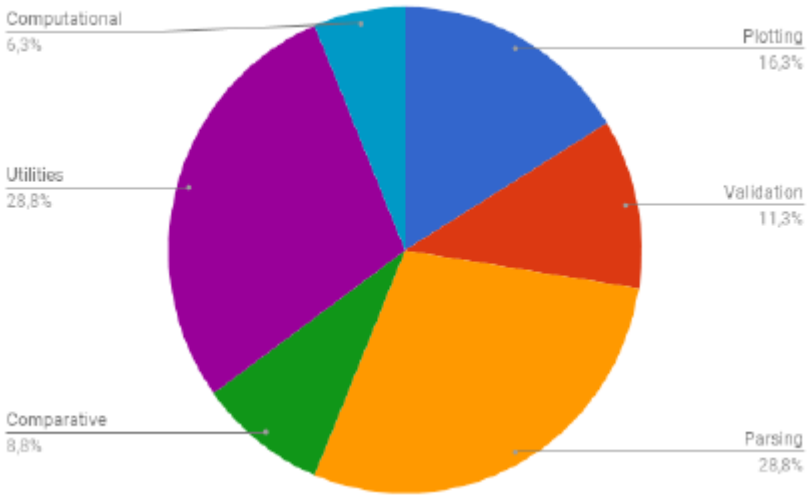
Εικόνα 6: Χάρτης πυκνότητας σχεδίασης

Στην εικόνα 5 παρουσιάζεται το σχέδιο που παράγεται από μια εκ των συναρτήσεων της κατηγορίας και παρουσιάζει τις υπερκαλύψεις (overlaps) που δημιουργούνται μεταξύ των cells μιας σχεδίασης. Στην εικόνα 6 η παρουσιάζεται άλλο ένα σχέδιο κατά το οποίο η σχεδίαση έχει χωριστεί σε πλέγμα-grid και χρωματίζεται ανάλογα με την πυκνότητα, με τις πιο πυκνές περιοχές να είναι οι κόκκινες.

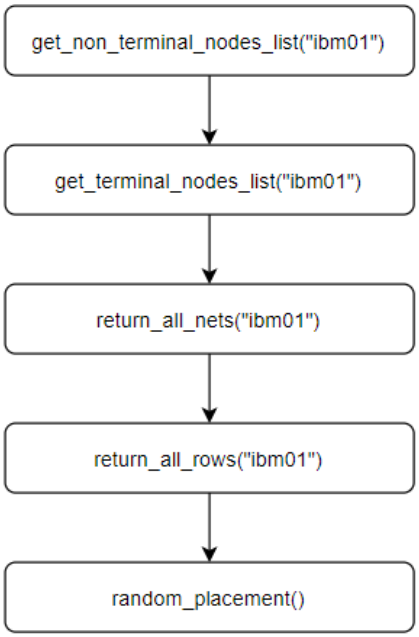
- Βοηθητικές – Utilities: Εδώ περιλαμβάνεται ένα πλήθος συναρτήσεων που αποκλίνουν αρκετά μεταξύ τους και παρέχουν μια μεγάλη γκάμα λειτουργιών. Οι συναρτήσεις κατηγοριοποίησης που υπάρχουν μπορούν να χρησιμοποιηθούν σε προσεγγίσεις ομαδοποίησης (clustering). Τέτοιες συναρτήσεις μπορούν να δώσουν πληροφορία για την τοποθεσία συγκεκριμένων cells, καθώς και των χαρακτηριστικών τους βάση της σημασίας τους για το κύκλωμα, π.χ. γωνιακά cells που καθορίζουν την περίμετρο ενός πλέγματος (net). Επίσης υπάρχουν συναρτήσεις που μπορούν να μετακινήσουν cells, να τα ανταλλάξουν μεταξύ τους και να τα

περιστρέψουν ενώ ως σύνολο μπορούν να πετύχουν ένα πλήθος αρκετών αλλαγών γρήγορα, βάση πάντα της προσέγγισης που έχει ακολουθηθεί για τη λύση του προβλήματος.

- Επαλήθευσης – Validation: Στην τελευταία κατηγορία εντάσσονται συναρτήσεις που επιτελούν διάφορους αυτοματισμούς ελέγχου, που μπορούν να βοηθήσουν στον εντοπισμό διαφόρων προβλημάτων στη σχεδίαση. Πιο συγκεκριμένα, υπάρχουν συναρτήσεις που ελέγχουν την εγκυρότητα μιας κίνησης που επιλέγει να κάνει ο αλγόριθμος ή τον υπολογισμό της ημιπεριμέτρου καλωδίου που θα προκύψει μετά από μια κίνηση ή ανταλλαγή cells (αν είναι μεγαλύτερη από την αρχική ή όχι). Τέτοιες συναρτήσεις επαληθεύουν αν μπορεί να γίνει μια κίνηση ή μια ανταλλαγή cell και τί φόρτο θα επιφέρει στο κύκλωμα και στο άθροισμα της ημιπεριμέτρου καλωδίου.



Εικόνα 3: Κατανομή συναρτήσεων PyPUT

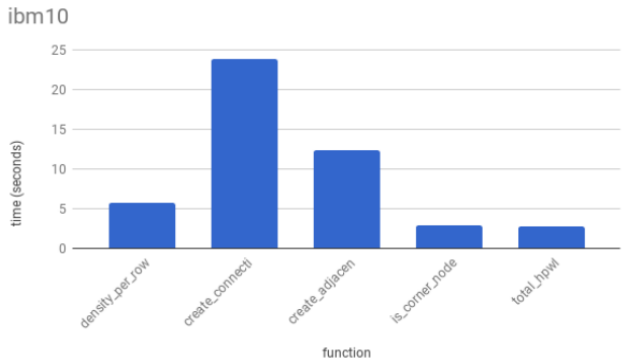


Εικόνα 4: Παράδειγμα χρήσης PyPUT

Παράδειγμα χρήσης

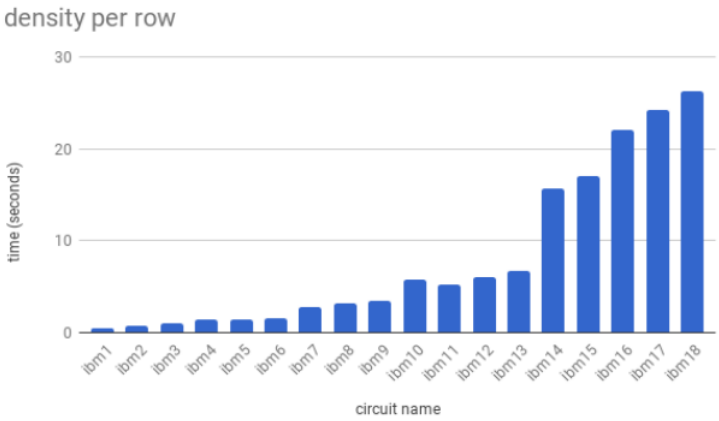
Στο παραπάνω σχήμα παρουσιάζεται η σειρά κλήσης κάποιων απαραίτητων στοιχείων του εργαλείου ώστε να γίνει πλήρης χρήση των δυνατοτήτων του. Οι δύο πρώτες συναρτήσεις επιστρέφουν λίστες που περιέχουν όλα τα μη-τερματικά και τα τερματικά cells, δηλαδή τα δομικά στοιχεία μίας σχεδίασης. Έπειτα με τη χρήση της τρίτης συνάρτησης επιστρέφεται στο χρήστη ένα dictionary, στο οποίο περιέχονται όλα τα nets καθώς και μια λίστα με τα cells που περιέχουν, ώστε να γίνει γνωστή η συνδεσμολογία των δομικών στοιχείων. Παρομοίως με τη συνάρτηση return_all_rows() επιστρέφονται όλες οι σειρές της σχεδίασης μαζί με τη λίστα των διαστάσεων και συντεταγμένων τους. Πλέον όλα τα απαραίτητα στοιχεία μίας σχεδίασης για το placement έχουν επιστραφεί στο χρήστη για οποιαδήποτε χρήση. Στο συγκεκριμένο σχήμα επιλέγεται η συνάρτηση της τυχαίας χωροθέτησης, δηλαδή τυχαία διανομή των δομικών στοιχείων πάνω στη σχεδίαση. Προφανώς έναντι της πέμπτης συνάρτησης μπορεί να γίνει χρήση άλλων, αναλόγως με το σκοπό του χρήστη, παρόλα αυτά, η random_placement() είναι κατάλληλη για την ανάλυση της απόδοσης αλγορίθμων placement, χωρίς να χρειάζεται να εκτελεστεί global placement, που είναι αρκετά χρονοβόρο.

Αποτελέσματα



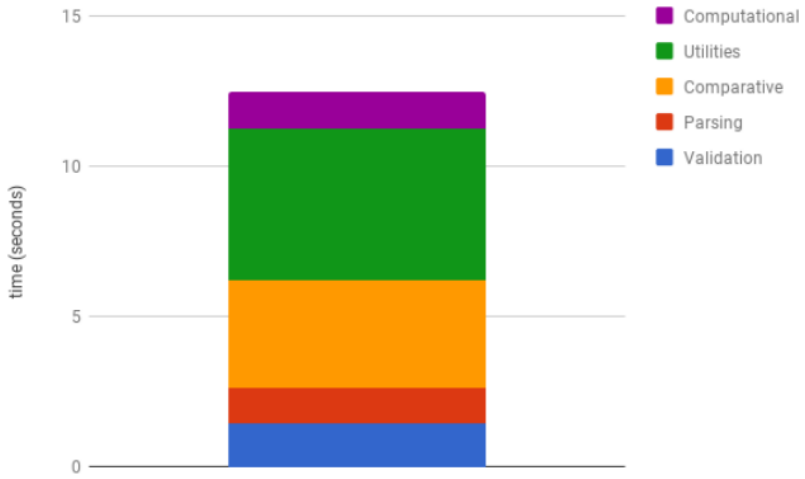
Εικόνα 5: Χρόνος εκτέλεσης των πιο χρονοβόρων συναρτήσεων

Συγκριτικά, οι πιο χρονοβόρες συναρτήσεις ως προς τον χρόνο εκτέλεσής τους, είναι οι density_per_row(), total_hpwl(), is_corner_node(), create_connectivity_matrix(), create_adjacency_matrix(), αφού πρέπει να διαβάσουν πολλαπλά αρχεία και να ανακτήσουν πολύ περισσότερες πληροφορίες σε σχέση με τις υπόλοιπες συναρτήσεις. Βεβαίως σημασία έχει και η λειτουργικότητα των συγκεκριμένων συναρτήσεων. Η πρώτη υπολογίζει την πυκνότητα όλων των γραμμών της σχεδίασης, έτσι πρέπει να συγκεντρώσει τα στοιχεία ολόκληρης της σχεδίασης (κελιά, γραμμές) και να υπολογίσει την πυκνότητα. Παρομοίως αρχίζει να εκτελείται και η total_hpwl(), η οποία πρέπει να υπολογίσει αρχικά την ημιπερίμετρο καλωδίου για κάθε net ξεχωριστά ώστε στη συνέχεια να υπολογιστεί το συνολικό μήκος καλωδίου. Τέλος is_corner_node() βρίσκει αν ένα cell είναι γωνιακό μέσα σε ένα ή περισσότερα net στα οποία ανήκει, έτσι για τον υπολογισμό χρειάζεται για ένα cell να ελέγξει όλα τα net, και αντιστοίχως όλα τα υπόλοιπα cell τους.



Εικόνα 6: Ενδεικτικός χρόνος εκτέλεσης σε διάφορα πρότυπα κυκλώματα

Στην εικόνα 10 παρουσιάζεται η διαφορά του χρόνου εκτέλεσης της συνάρτησης density_per_row() σε διάφορα πρότυπα κυκλώματα. Ο χρόνος εκτέλεσης που παρουσιάζεται είναι αναμενόμενος αν λάβουμε υπόψη πως τα κυκλώματα αυτά έχουν διαφορετικό μέγεθος, με το μεγαλύτερο να βρίσκεται στα δεξιά (ibm18). Έτσι ο χρόνος εκτέλεσης είναι ανάλογος του αριθμού των cells που βρίσκονται στη σχεδίαση.



Εικόνα 7: Σχετικός χρόνος εκτέλεσης συναρτώσεων σε κάθε κατηγορία

Τέλος στο παραπάνω γράφημα παρουσιάζεται ο σχετικός χρόνος εκτέλεσης, μοιρασμένος σε κάθε κατηγορία συναρτήσεων αν υποθέσουμε ότι κάποιος θα εκτελέσει όλες τις συναρτήσεις του εργαλείου, χωρίς να συμπεριλαμβάνονται οι συναρτήσεις γραφημάτων (plotting), αφού εξαρτώνται κυρίως από τις υπορουτίνες που χρησιμοποιεί η βιβλιοθήκη matplotlib και όχι μόνο από τον κώδικα του εργαλείου.

Συμπεράσματα και Μελλοντικές προεκτάσεις

Το συγκεκριμένο κεφάλαιο αφορά το εργαλείο PyPUT και την αντίστοιχη δημοσίευσή του στο συνέδριο SEEDA το 2018 [23], όπου παρουσιάστηκε το εργαλείο PyPUT, μια βιβλιοθήκη συναρτήσεων με σκοπό να λειτουργήσει ως βοήθημα στο πεδίο της χωροθέτησης. Η πρώτη έκδοση της βιβλιοθήκης συμπεριλαμβάνει μια πληθώρα από κομμάτια κώδικα που μπορούν να γίνουν οι ακρογωνιαίοι λίθοι για την κατασκευή αλγορίθμων σχετικών με τη χωροθέτηση.

Το PyPUT μπορεί εύκολα να επεκταθεί σε θέματα λειτουργικότητας. Είναι δυνατό να υλοποιηθούν διάφοροι μονολιθικοί αλγόριθμοι placement ή μικρότερα μέρη τους. Επίσης, βάση των αποτελεσμάτων, είναι εμφανές ότι συγκεκριμένες συναρτήσεις μπορούν να βελτιστοποιηθούν. Λαμβάνοντας υπόψη και το γεγονός πως το μέγεθος κάθε σχεδίασης είναι αυξανόμενο, θα μπορούσαν να υλοποιηθούν τεχνικές παραλληλοποίησης για να μειωθεί ο χρόνος εκτέλεσης. Τέλος θα μπορούσε να γίνει χρήση και υλοποίηση διαφόρων δομών δεδομένων ώστε να γίνει πιο εύκολη και αποδοτική η χρήση του εργαλείου, αφού στη αρχική του μορφή ακολουθεί το πρότυπο της σειριακής/συναρτησιακής μορφής προγραμματισμού.

ΚΕΦΑΛΑΙΟ 3: A Novel Genetic Algorithm for I/O Pad Planning Retaining Former Cell Positions

Εισαγωγή

Το Floorplanning και το Placement θεωρούνται τα πιο σημαντικά βήματα της Φυσικής Σχεδίασης, αφού επηρεάζουν σε μεγαλύτερο βαθμό την ποιότητα του τελικού αποτελέσματος της παραγόμενης σχεδίασης, σύμφωνα με την τρέχουσα τεχνολογία υλοποίησης. Ένας λειτουργικά μέτριος floorplanner ή placer μπορεί να αποδειχθεί επιζήμιος για την σχεδίαση. Μετά το βήμα του floorplanning τα I/O pads (σημεία εισόδου/εξόδου) θεωρούνται μόνιμα τοποθετημένα και πλέον αμετακίνητα. Έτσι, τα I/O pads καταλήγουν να γίνονται επιπλέον περιορισμοί τους οποίους πρέπει να συμπεριλάβει στην επίλυση ο placer που έχει αναλάβει τη διαδικασία της χωροθέτησης των κελιών της σχεδίασης στις θέσεις τους. Παρόλα αυτά, υπάρχουν περιπτώσεις που η εκ-νέου εφαρμογή της διάταξης της σχεδίασης είναι αναπόφευκτη. Συνήθως αυτό είναι παράγωγο των νεότερων τεχνολογιών πυριτίου, οι οποίες θεωρούν τα ήδη κατασκευασμένα chip ως απαρχαιωμένα, αφού αποτυγχάνουν να φτάσουν την αναμενόμενη απόδοσή τους ή/και τη λειτουργικότητά τους. Θεωρώντας πως το προηγούμενο σενάριο είναι αληθινό, μία απλοϊκή προσέγγιση θα ήταν να γίνει επανασχεδιασμός όλου του κυκλώματος από την αρχή, κάτι που θα ήταν εξαιρετικά χρονοβόρο και δύσκολο ως διαδικασία. Εναλλακτικά, δεδομένης μίας ήδη χωροθετημένης σχεδίασης, κι ενός συνόλου από στόχους με σκοπό της βελτίωση της απόδοσης, μπορούν να εφαρμοστούν ένα σύνολο από μικρές αλλαγές πάνω στην ήδη υπάρχουσα σχεδίαση. Σχετικά με το συνολικό μήκος καλωδίου, μία γρήγορη αναδιάταξη των I/O pads μπορεί να οδηγήσει σε βελτίωση της μετρικής, δεδομένου ότι το πλήθος τους είναι αρκετά μικρότερο από το πλήθος των κελιών της σχεδίασης. Η συγκεκριμένη διαδικασία μπορεί να θεωρηθεί είτε ως βήμα floorplanning είτε ως ένα ξεχωριστό βήμα το οποίο εφαρμόζεται παράλληλα με τη ροή σχεδίασης και χαρακτηρίζει cells σε συγκεκριμένες περιπτώσεις ως μη-μετακινούμενες οντότητες.

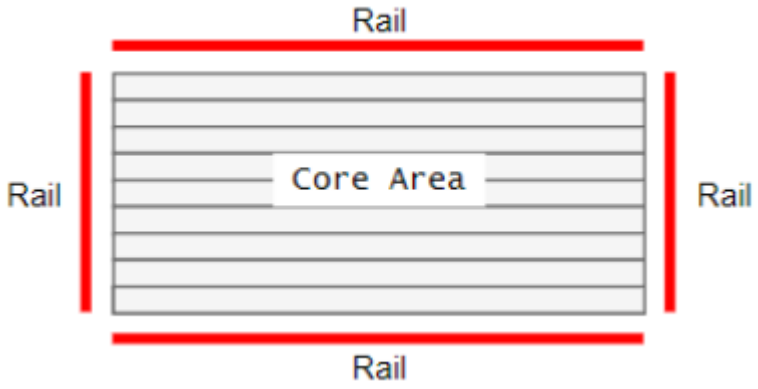
Οι περισσότερες προσεγγίσεις που αφορούν την αναδιάταξη των I/O pads σε σύγχρονες έρευνες στοχεύουν στην ελαχιστοποίηση του μήκος καλωδίου. Στο άρθρο [24] η δομή της σχεδίασης και οι περιορισμοί των καθυστερήσεων συνδυάζονται σε σχέσεις εγγύτητας μεταξύ των I/O pads, τα οποία πρέπει να τοποθετούνται στην περιφέρεια της περιοχής του πυρήνα, έτσι ώστε να είναι δυνατή η εφαρμογή της τεχνικής της γραμμικής τοποθέτησης, ή οποία τα τοποθετεί στις κατάλληλες θέσεις πριν περάσει η σχεδίαση σε έναν placer για να συνεχίσει η ροή. Στο [25], ο αλγόριθμος για την τοποθέτηση των pads λειτουργεί σαν ένα προ-επεξεργαστικό βήμα ενός placer. Ένας ILP αλγόριθμος προτείνεται στο [26] για τον καθορισμό των pads που λύνει το πρόβλημα της χωροθέτησης ενώ στο [27] παρουσιάζεται μια συνολική μελέτη η οποία σημειώνει τις επιπτώσεις της χωροθέτησης των I/O pads σε μια γραμμική χωροθέτηση (row-based placement). Επιπλέον, ένας αλγόριθμος δύο φάσεων, ο οποίος εξετάζει τα pads που βρίσκονται πάνω και εκτός του chip, παρουσιάζεται στο [28]. Στη φάση ομαδοποίησης (clustering) του, δημιουργούνται οι σχετικές ομάδες κόμβων (node clusters), που διατηρούν την αρχική ιεραρχία, ενώ

η φάση σχεδιασμού του προσπαθεί να αναθέσει τις βέλτιστες δυνατές θέσεις σε αυτές τις ομάδες έτσι ώστε οι κύριες είσοδοι και έξοδοι του κυκλώματος να βρίσκονται σε νόμιμες θέσεις ενώ παράλληλα ελαχιστοποιείται το συνολικό μήκος καλωδίου. Τέλος οι συγγραφείς του [29] παρουσιάζουν μια μέθοδο ανάθεσης θέσεων σε pads που στοχεύει στην ελαχιστοποίηση του φαινομένου crosstalk και του σύγχρονου θορύβου/καθυστερήσεων εναλλαγών.

Σε αυτή την έρευνα, προτείνεται ένας πρότυπος γενετικός αλγόριθμος ανάθεσης I/O pads που μειώνει το συνολικό μήκος καλωδίου το οποίο έχει υπολογιστεί στην προηγούμενη κατάσταση της σχεδίασης (πριν την εκτέλεσή του) και μπορεί να γίνει κομμάτι οποιουδήποτε floorplanner ή/και placer.

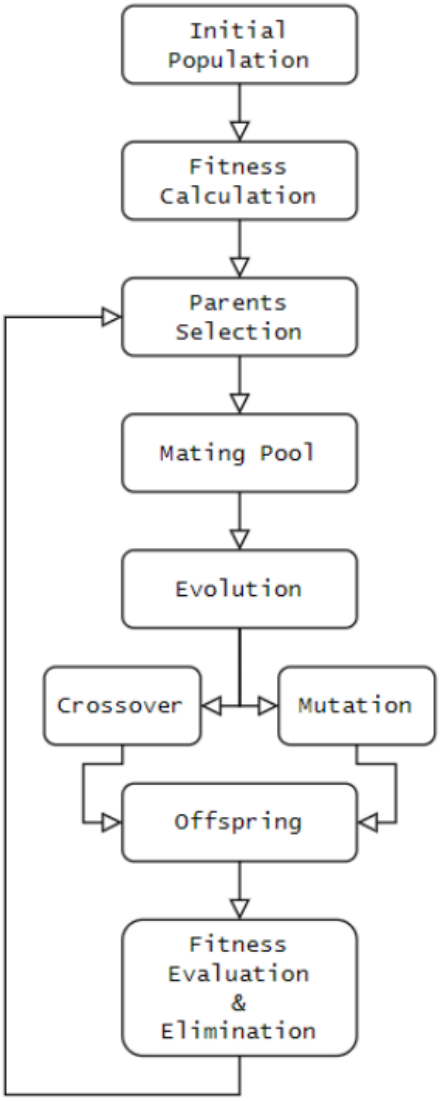
Διατύπωση του προβλήματος

Ο αλγόριθμος στοχεύει σε standard cell σχεδιάσεις. Κάθε σχεδίαση αποτελείται από μια λίστα κόμβων. Οι τερματικοί κόμβοι, είναι είτε τα τερματικά κελιά (cells) είτε τα I/O pins της σχεδίασης, δηλαδή μονάδες οι οποίες δεν μετακινούνται, ενώ οι μη-τερματικοί κόμβοι είναι τα κελιά τα οποία τοποθετούνται στον πυρήνα της σχεδίασης. Κάθε αποτέλεσμα το οποίο θα παράγεται, πρέπει να υπακούει στους εξής κανόνες. Αρχικά κάθε τερματικό cell μπορεί να τοποθετηθεί οπουδήποτε πάνω στην περιοχή των I/O pads. Έπειτα, δύο διακριτά I/O cells δεν μπορούν να βρίσκονται στο ίδιο σημείο. Τέλος οι ράγες (rails) που φαίνονται στο παρακάτω σχήμα, είναι αυτές που περιέχουν όλες τις νόμιμες I/O pad τοποθεσίες, και σχηματίζονται από την αρχική μορφή της σχεδίασης πριν την εκτέλεση του αλγορίθμου. Στόχος είναι να βρεθούν με ακρίβεια οι τοποθεσίες που παράγουν σχεδίαση με αρκετά μειωμένο μήκος καλωδίου.



Εικόνα 8: Διαθέσιμες τοποθεσίες I/O pad

Προτεινόμενος Αλγόριθμος



Εικόνα 9: Ροή αλγορίθμου

Στην παραπάνω εικόνα παρουσιάζεται η ροή του προτεινόμενου αλγορίθμου, από την αρχική δημιουργία του πληθυσμού έως την εξέλιξη και την παραγωγή νέων γενεών που θα οδηγήσει στην τελική λύση.

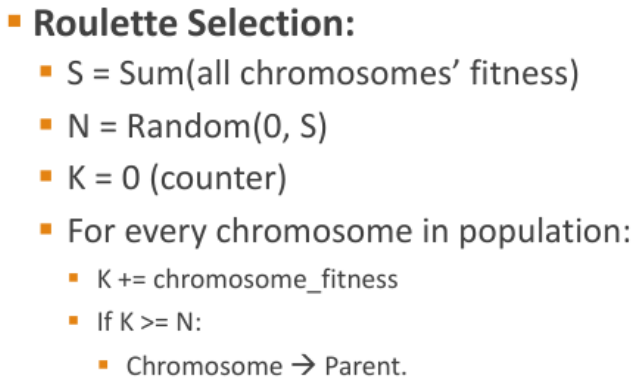
Πριν την αναλυτική περιγραφή της λειτουργίας της προτεινόμενης μεθόδου, πρέπει να γίνει μια σύντομη εισαγωγή στην ορολογία που χρησιμοποιείται. Ένα I/O pin που μπορεί να τοποθετηθεί οπουδήποτε σε μια από τις διαθέσιμες pad τοποθεσίες, θεωρείται ως ένα γονίδιο (gene). Όπως ένα χρωμόσωμα αποτελείται από πολλά γονίδια, έτσι κι εδώ το αντίστοιχο χρωμόσωμα αποτελείται από όλα τα pins της σχεδίασης που έχουν τοποθετηθεί πάνω στα pads. Στόχος είναι να αναγνωρίσουμε το καλύτερο (fittest) χρωμόσωμα, κρίνοντάς το με βάση του

συνολικού μήκος καλωδίου, αφού έχουν περάσει αρκετά στάδια αλλεπάλληλων βημάτων εξέλιξης από τον αρχικό πληθυσμό.

Στο πρώτο βήμα του αλγορίθμου ένας αρχικός και άδειος πληθυσμός δημιουργείται. Για να γεμίσουμε αυτή τη λίστα πληθυσμού, δημιουργούμε ένα στατικό αριθμό από τυχαία χρωμοσώματα των οποίων το fitness αμέσως υπολογίζεται και αξιολογείται. Πρακτικά ένα χρωμόσωμα είναι μια λίστα που απεικονίζει όλες τις διαθέσιμες τοποθεσίες pad. Όταν ένα pin λαμβάνει μια θέση τότε, το όνομά του αποθηκεύεται σε αυτή τη θέση. Οι άδειες θέσεις λαμβάνουν την τιμή -1. Για παράδειγμα το χρωμόσωμα {-1, -1, 0, -1, 2, 1, -1, 3} μπορεί να μας δώσεις τις εξής πληροφορίες. Αρχικά οι συνολικές διαθέσιμες θέσεις pad είναι 8. Λαμβάνοντας υπόψη πως έχουμε 4 ράγες στην περιφέρεια της σχεδίασης ερχόμαστε στο συμπέρασμα πως κάθε πλευρά θα έχει 2 διαθέσιμες θέσεις στις οποίες μπορούν να τοποθετηθούν pins. Διαβάζοντας το χρωμόσωμα, διαπιστώνουμε πως η πρώτη πλευρά ή το πρώτο rail δεν έχει πάνω του κανένα pin, αφού και οι δύο θέσεις έχουν την τιμή -1. Στην πρώτη θέση του δεύτερου rail έχουμε το pin με αναγνωριστικό 0, ενώ η δεύτερή του θέση είναι άδεια. Ομοίως διαβάζουμε και το υπόλοιπο χρωμόσωμα. Σημαντικό είναι να τονίσουμε πως η συγκεκριμένη μορφή χρωμοσώματος είναι αρκετά βολική καθώς εύκολα μπορεί κανείς να αποφύγει την τοποθέτηση 2 pins πάνω στο ίδιο σημείο ελέγχοντας αν υπάρχει άλλο αναγνωριστικό κάποιου pin σε αυτή τη θέση της λίστας, τηρώντας πάντα τον δεύτερο κανόνα. Επίσης κάθε θέση αυτής της λίστας αντιπροσωπεύει μια πραγματική συντεταγμένη πάνω στη σχεδίαση, έχοντας ως δεδομένο πως όλα τα pins έχουν διαστάσεις ίσες με 1 σε μήκος και πλάτος. Έτσι αν για παράδειγμα στην πρώτη θέση αντιστοιχούν οι συντεταγμένες x=0, y=0, τότε στη δεύτερη θέση της λίστας έχουμε τη θέση x=0, y=1 ανεβαίνοντας κατακόρυφα μια θέση στο αριστερό rail. Η τρίτη και τέταρτη θέση αντιστοιχούν στο δεύτερο rail της σχεδίασης που είναι τοποθετημένο πάνω από την κεντρική περιοχή. Έτσι διαβάζουμε τις συντεταγμένες y=max_height που έχει υπολογιστεί από την αρχική μορφή της σχεδίασης. Το ίδιο μοτίβο ακολουθείται και για τα υπόλοιπα rails. Η τυχαιότητα του αρχικού πληθυσμού, σημαίνει πως τα pins τοποθετήθηκαν τυχαία πάνω στις διαθέσιμες θέσεις για κάθε χρωμόσωμα, τηρώντας πάντα τους κανόνες που τέθηκαν και τις βασικές πρακτικές των γενετικών αλγορίθμων. Ο υπολογισμός της τιμής fitness γίνεται αφού τοποθετηθούν όλα τα pins πάνω σε κάθε χρωμόσωμα και για κάθε χρωμόσωμα μπορεί να υπολογιστεί το μήκος καλωδίου. Ο τρόπος του υπολογισμού αυτής της τιμής θα αναλυθεί στη συνέχεια.

Στο επόμενο βήμα, ακολουθώντας την ροή της σχεδίασης αρχίζει η διαδικασία της αναπαραγωγής του πληθυσμού και της εξέλιξης. Αρχικά γίνεται η επιλογή των υποψήφιων γονέων μέσω ενός παραλλαγμένου αλγορίθμου ρουλέτας [30], στο οποίο έχουν γίνει κάποιες τροποποιήσεις ώστε να λειτουργεί με βάση τον αλγόριθμο. Συνήθως ο αλγόριθμος ρουλέτας εφαρμόζεται σε προβλήματα μεγιστοποίησης τιμής, αλλά στην περίπτωση του συγκεκριμένου αλγορίθμου έχουμε ως στόχο την ελαχιστοποίησης του συνολικού μήκους καλωδίου της σχεδίασης. Για να επιτευχθεί αυτός ο στόχος, γίνεται τροποποίηση της τιμής fitness. Στις συνήθεις πρακτικές των γενετικών αλγορίθμων το fitness είναι συνήθως μια τιμή η οποία όσο μεγαλύτερη γίνεται, τόσο καταλληλότερο θεωρείται το χρωμόσωμα. Η αλλαγή που πραγματοποιήθηκε εδώ είναι πως πλέον η τιμή fitness υπολογίζεται ως διαφορά της μέγιστης τιμής που συναντάται σε κάθε πληθυσμό αφαιρώντας κάθε φορά την εκάστοτε τιμή του χρωμοσώματος που εξετάζουμε. Έτσι το χρωμόσωμα με το αρχικό

μέγιστο fitness, που είναι και το χρωμόσωμα με την χειρότερη/μέγιστη τιμή μήκους καλωδίου, θα αποκτήσει την τιμή 0 αφού θα αφαιρέσει από το fitness του την ίδια τιμή. Ως αποτέλεσμα το καλύτερο χρωμόσωμα, αυτό δηλαδή με τη μικρότερη τιμή μήκους καλωδίου θα αποκτήσει την μεγαλύτερη τιμή διότι η διαφορά αυτής της τιμής από το μέγιστο θα είναι η μεγαλύτερη όλων. Σημαντικό είναι να τονιστεί και η διαδικασία ενός αλγορίθμου ρουλέτας, καθώς έτσι γίνεται εμφανής ο λόγος αλλαγής της τιμής fitness από την αρχικά ελάχιστη τιμή ως καλύτερη, στη μέγιστη. Ο αλγόριθμος συνοπτικά λειτουργεί με τον εξής τρόπο. Αρχικά υπολογίζεται το άθροισμα-S όλων των τιμών fitness. Έπειτα επιλέγεται ένας τυχαίος αριθμός-N στο όριο (0, S) και αρχικοποιείται ένας μετρητής k=0. Στη συνέχεια για κάθε χρωμόσωμα προσθέτουμε στην τιμή k το fitness του και εξετάζουμε αν η ανανεωμένη τιμή k είναι μεγαλύτερη ή ίση της τιμής N. Αν είναι τότε το χρωμόσωμα επιλέγεται ως γονέας. Γίνεται επανάληψη για όλα τα χρωμοσώματα.



Εικόνα 10: Αλγόριθμος επιλογής ρουλέτας

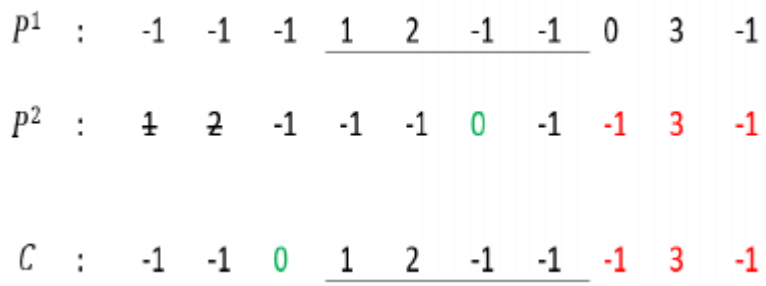
Ο αλγόριθμος επιλογής ρουλέτας είναι σημαντικός καθώς οι πιθανότητες επιλογής ενός χρωμοσώματος γίνονται περισσότερες αν το fitness του έχει μεγάλο αριθμό σε αντίθεση με ένα άλλο που έχει μικρότερο, αφού το φράγμα N μπορεί να σπάσει ευκολότερα αν στην τιμή k προστεθεί μεγάλος αριθμός. Βεβαίως αφού πρόκειται για τυχαίο αριθμό είναι πιθανό να επιλεγεί και χρωμόσωμα με μικρή τιμή fitness διατηρώντας την τυχαιότητα που χρειάζεται ένας γενετικός αλγόριθμος. Στη συγκεκριμένη επιλογή ένα χρωμόσωμα είναι δυνατό να επιλεγεί πάνω από μία φορές. Συνήθως αυτό συμβαίνει για τα χρωμοσώματα με μεγαλύτερες τιμές fitness και βοηθά στη σύγκλιση του αποτελέσματος.

Μετά την επιλογή των διαθέσιμων γονέων αρχίζει η διαδικασία της αναπαραγωγής/crossover από την οποία θα παράγονται δύο παιδιά-χρωμοσώματα τα οποία θα γίνουν κομμάτι του πληθυσμού και της διαδικασίας επιλογής στην επόμενη γενιά αφού γίνει υπολογισμός τις τιμής fitness τους.

Στη συνέχεια θα γίνει η ανάλυση της διαδικασίας αναπαραγωγής, η οποία γίνεται με τη χρήση ενός crossover operator. Το crossover ως μεθοδολογία είναι η πιο απλή και πιο διαδεδομένη μέθοδος αναπαραγωγής για έναν γενετικό αλγόριθμο. Πρακτικά τα δύο επιλεγμένα χρωμοσώματα σπάνε σε δύο μέρη το καθένα, με κοινό το σημείο διαχωρισμού. Έπειτα το πρώτο μέρος του πρώτου χρωμοσώματος ενώνεται με το δεύτερο μέρος του δεύτερου χρωμοσώματος δημιουργώντας ένα παιδί. Ομοίως συνδυάζονται τα άλλα δύο υπόλοιπα για την παραγωγή ενός δεύτερου. Η διαδικασία

αναπαραγωγής και ο operator που χρησιμοποιείται σε αυτό τον αλγόριθμο χρησιμοποιεί αυτή τη βασική αρχή με αρκετές παραλλαγές. Αρχικά για κάθε επιλεγμένο γονέα από τη διαδικασία επιλογής δημιουργούνται ζευγάρια {γονέας1, γονέας2}, {γονέας2, γονέας3} κ.λ.π. Σημειώνεται πως σε αυτή τη διαδικασία είναι πιθανό να έχουμε ζευγάρι που αποτελείται από το ίδιο χρωμόσωμα καθώς όπως αναφέρθηκε πριν, η επιλογή ρουλέτας μπορεί να επιλέξει πάνω από μία φορές το ίδιο χρωμόσωμα. Επίσης ένας γονέας χρησιμοποιείται σε δύο ζεύγη ώστε να δημιουργηθεί το επιθυμητό πλήθος απογόνων. Στη συνέχεια για κάθε ζευγάρι γονέων, επαναληπτικά δημιουργούνται δύο τυχαία σημεία διαχωρισμού τα οποία θα είναι κοινά για κάθε χρωμόσωμα, θα χωρίζουν δηλαδή τα χρωμοσώματα στο ίδια σημεία/δείκτες λίστας. Έτσι οι δύο γονείς θα χωριστούν σε τρία κομμάτια ενώ παράλληλα θα γίνει η αρχικοποίηση ενός άδειου χρωμοσώματος που θα πάρει το ρόλο του απογόνου. Στη συνέχεια παίρνουμε αυτούσιο το μεσαίο κομμάτι του πρώτου γονέα και το αντιγράφουμε στο παιδί, στην ίδια θέση. Εξετάζουμε τα υπόλοιπα γονίδια στο δεξιό κομμάτι του δεύτερου γονέα με τον εξής τρόπο. Αν το γονίδιο υπάρχει ήδη στο παιδί τότε το αντίστοιχο γονίδιο στην ίδια θέση παίρνει την τιμή -1, ειδάλλως του ανατίθεται το ίδιο γονίδιο. Συνεχίζουμε εξετάζοντας τα υπόλοιπα γονίδια του δεύτερου γονέα. Αν δεν βρίσκονται ήδη μέσα στο παιδί τότε μεταφέρονται στην πρώτη κενή θέση είτε στο αριστερό είτε στο δεξί μέρος του παιδιού. Η συγκεκριμένη μέθοδος αναπαραγωγής εξασφαλίζει τη μοναδικότητα κάθε γονιδίου, δεδομένου πως δεν γίνεται να υπάρχουν δύο ίδια pins στην ίδια σχεδίαση. Επίσης είναι σίγουρο πως δεν θα λείψει κάποιο pin από το τελικό αποτέλεσμα και κατ' επέκταση από τη σχεδίαση.

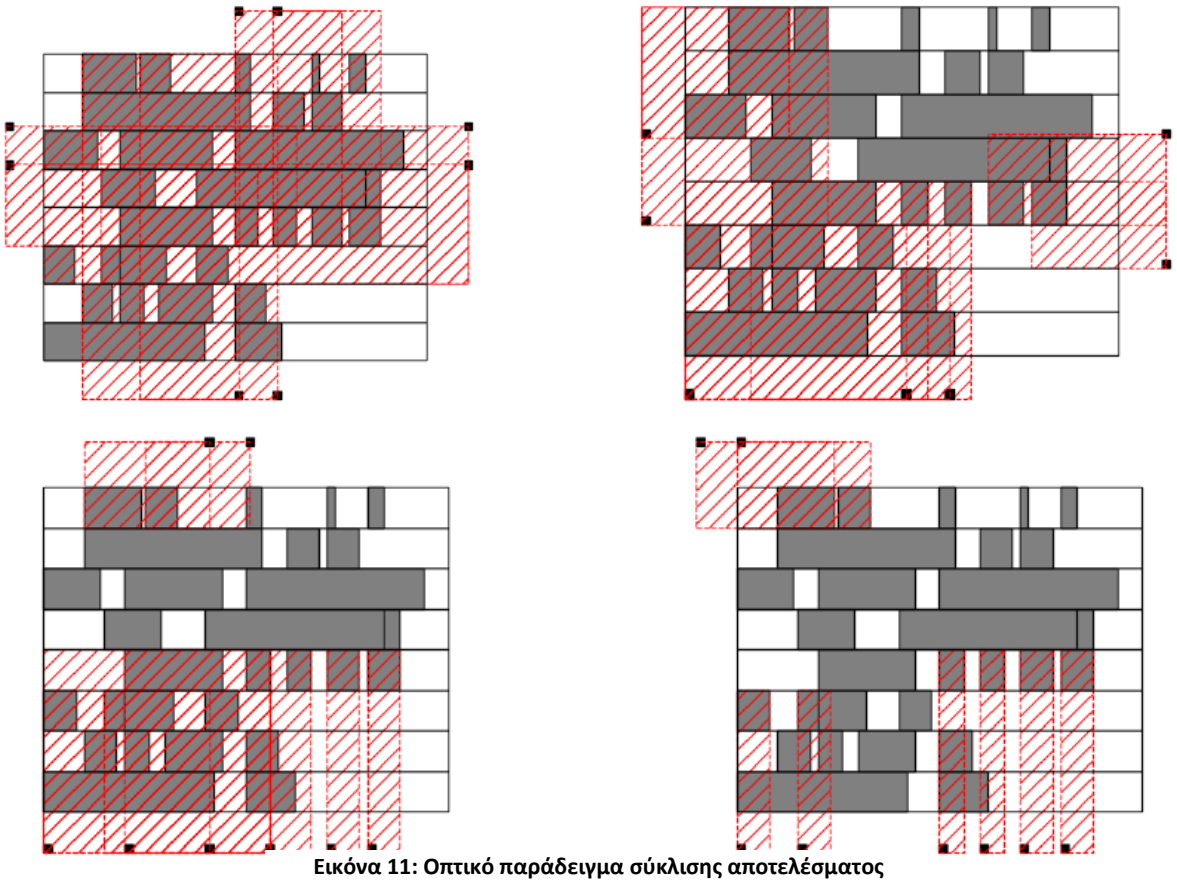
Στο παράδειγμα που ακολουθεί γίνεται πιο κατανοητός ο τρόπος αναπαραγωγής.



Ακολουθώντας την μεθοδολογία που αναφέρθηκε οι δύο γονείς χωρίζονται νοητά στα ίδια σημεία έχοντας πλέον ο καθένας τρία μέρη. Το επιλεγμένο και μεσαίο μέρος του πρώτου γονέα μεταφέρεται αυτούσιο στο ίδιο κομμάτι του παιδιού. Έπειτα ελέγχεται η αριστερά πλευρά του δεύτερου γονέα. Εφόσον τα δύο υπάρχοντα γονίδια βρίσκονται ήδη μέσα στο μεσαίο μέρος του παιδιού οι αντίστοιχες θέσεις παίρνουνε την τιμή -1. Τέλος ελέγχεται το κεντρικό και το δεξί μέρος του δεύτερου γονέα. Το pin με αναγνωριστικό 0 δεν υπάρχει στο παιδί έτσι μεταφέρεται στην πρώτη διαθέσιμη θέση του απογόνου, ενώ το pin με αναγνωριστικό 3 μεταφέρεται στην αντίστοιχη θέση αφού δεν υπάρχει κάποιο άλλο εκεί. Χρησιμοποιώντας το ίδιο ζεύγος γονέων, αντιστρέφοντας τη θέση τους, είναι δυνατό να παραχθεί ένα νέο χρωμόσωμα. Κάνοντας χρήση αυτής της τακτικής με το ίδιο ζεύγος γονέων μπορούν να παραχθούν δύο διαφορετικοί απόγονοι. Στο συγκεκριμένο αλγόριθμο μετά τη λήξη της διαδικασίας αναπαραγωγής ο πληθυσμός θα διπλασιαστεί.

Στο επόμενο βήμα εφαρμόζεται μετάλλαξη των γονιδίων στους απογόνους που δημιουργήθηκαν. Η μετάλλαξη έχει ως στόχο την αύξηση της τυχαιότητας στον πληθυσμό και γι' αυτό εφαρμόζεται μόνο αν ένας τυχαίος αριθμός/πιθανότητα που παράγεται για κάθε υποψήφιο προς μετάλλαξη είναι μεγαλύτερος από έναν αριθμό ορισμένο από τον χρήστη. Το όριο που θέτει ο χρήστης έχει σημασία καθώς επηρεάζει αναλόγως τη σύγκλιση αποτελέσματος. Πρακτικά η μετάλλαξη που εφαρμόζεται εδώ, είναι μια απλή ανταλλαγή γονιδίων. Δηλαδή σε ένα χρωμόσωμα, επιλέγονται δύο τυχαία γονίδια/pins και αλλάζουν θέσεις μεταξύ τους. Αυτό βεβαίως συμπεριλαμβάνει και άδειες θέσεις οι οποίες έχουν την τιμή -1.

Τέλος, μετά από κάθε γενιά ο πληθυσμός έχει διπλασιαστεί. Αυτό όμως δημιουργεί αρκετά προβλήματα καθώς αν ένας πληθυσμός αρχικοποιηθεί με 100 χρωμοσώματα, τότε στο πέρασμα 3 γενεών θα έχει 800 χρωμοσώματα (2x100 = 200, 200x2 = 400, 400x2 = 800). Συνήθως τα πειράματα γενετικών αλγορίθμων ορίζονται με επαναλήψεις άνω των 1000 γενεών, το οποίο με συνεχή αυξανόμενο πληθυσμό δημιουργεί προβλήματα μνήμης καθώς και ταχύτητας εκτέλεσης του αλγορίθμου και σύγκλισης του αποτελέσματος. Έτσι για να διατηρηθεί ο πληθυσμός σε ένα στατικό αριθμό, ο οποίος έχει οριστεί πριν την εκτέλεση του αλγορίθμου και θα είναι ίσος με τον αρχικό πληθυσμό, επέρχεται η διαδικασία αποκλεισμού (elimination). Σε αυτό το βήμα, ο πληθυσμός ταξινομείται με φθίνουσα σειρά βάση της τιμής fitness και ο μισός πληθυσμός στη βάση της λίστας διαγράφεται, κρατώντας έτσι τα πιο κατάλληλα χρωμοσώματα του πληθυσμού και διατηρώντας τον αριθμό τους στατικό. Ο αλγόριθμος τερματίζει μετά το πέρας ενός ορισμένου αριθμού επαναλήψεων/γενεών από το χρήστη.



Εικόνα 11: Οπτικό παράδειγμα σύκλισης αποτελέσματος

Στην παραπάνω εικόνα δίνεται ένα παράδειγμα της σύγκλισης του αλγορίθμου, όπως εκτυπωμένο κάνοντας χρήση των συναρτήσεων εκτύπωσης του εργαλείου PyPUT. Για ένα οπτικά εύκολο αποτέλεσμα έγινε χρήση ενός πρότυπου κυκλώματος πολύ μικρού μεγέθους το οποίο αποτελείται από 8 pins και 32 τυπικά cells. Τα I/O pins είναι οι ορθογώνιες τελείες στην περιφέρεια του κυκλώματος, τα cells είναι τα γκρίζα ορθογώνια στο εσωτερικό της σχεδίασης ενώ τα κόκκινα πλέγματα είναι τα nets που περικλείουν τα pins και τα αντίστοιχα cells. Στην αριστερά πάνω εικόνα παρουσιάζεται ο αρχικός πληθυσμός, τα pins στην περιφέρεια του κυκλώματος καθώς και τα net στα οποία ανήκου. Στην πάνω δεξιά εικόνα φαίνεται το στιγμιότυπο κάποιες γενιές μετά την αρχικοποίηση του πληθυσμού, ενώ ήδη είναι εμφανής η πρώτη προσπάθεια μείωσης του μήκος καλωδίου. Ομοίως σε ένα στιγμιότυπο πιο κοντά στη σύγκλιση, στην κάτω αριστερά εικόνα, κάποια pins έχουν λάβει ήδη τις καταλληλότερες θέσεις τους πάνω στη σχεδίαση. Στην τελευταία εικόνα φαίνεται η σύγκλιση του αποτελέσματος καθώς τα pins έχουν λάβει πλέον θέσεις που μειώνουν σε πολύ μεγάλο βαθμό την περιοχή των nets τους και κατ' επέκταση το συνολικό μήκος καλωδίου σε σχέση με την πρώτη εικόνα και τον αρχικό πληθυσμό.

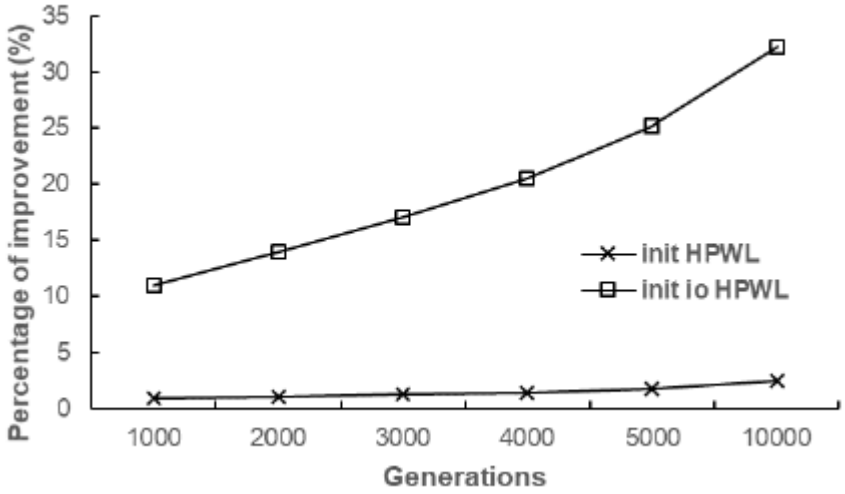
Αποτελέσματα

Ο γενετικός αλγόριθμος αποτελεί δημοσιευμένη εργασία [31] κι έχει υλοποιηθεί στη γλώσσα Python χωρίς τη χρήση επιπλέον βιβλιοθηκών. Τα πειράματα εκτελέστηκαν πάνω σε πρότυπα κυκλώματα τα οποία έχουν γίνει globally placed χρησιμοποιώντας το εργαλείο NTUPlace [20] και έγιναν legalized χρησιμοποιώντας το [8].

design	*#cells*	*#I/O pins*	*init_wl_impr*	*init_io_wl_impr*
ibm01	12505	246	3.92%	28.71%
ibm02	19341	259	2.56%	30.84%
ibm03	22852	283	2.83%	34.18%
ibm04	27219	287	3.55%	46.21%
ibm05	28145	1201	1.8%5	10.45%
ibm06	32331	166	1.18%	28.78%
ibm07	45638	287	0.61%	10.4%
ibm08	51022	286	3.22%	55.78%
ibm09	53109	285	2.85%	52.54%
ibm10	68684	744	4.79%	40.56%
ibm11	70151	406	1.68%	28.47%
ibm12	70438	637	3.33%	40.89%
ibm13	83708	490	2.25%	34.04%
ibm14	147087	517	3.24%	57.63%
ibm15	161186	383	1.04%	30.91%
ibm16	182979	504	0.94%	22.55%
ibm17	184751	743	0.91%	17.76%
ibm18	210340	273	1.02%	42,12%

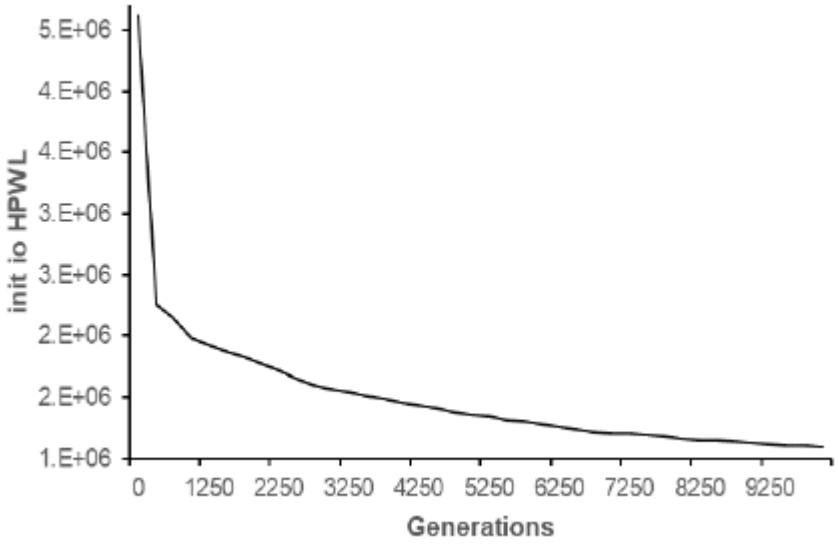
Εικόνα 12: Στοιχεία πρότυπων κυκλωμάτων και αποτελέσματα

Στον παραπάνω πίνακα φαίνονται τα αποτελέσματα του αλγορίθμου. Στις στήλες #cells και #I/O pins αναγράφονται ο αριθμός των κελιών κάθε benchmark καθώς και ο αριθμός των I/O pins, ενώ στις στήλες init_wl_impr() και init_io_wl_impr() φαίνονται οι ποσοστιαίες βελτιώσεις στο συνολικό μήκος καλωδίου καθώς και στο μήκος καλωδίου που αφορά μόνο τα I/O pins μετά το πέρας 10000 γενεών. Είναι εμφανές στις περισσότερες περιπτώσεις ότι υπάρχει αρκετά μεγάλη βελτίωση στο μήκος καλωδίου.



Εικόνα 13: Μέση βελτίωση απόδοσης στο αρχικό HPWL και pins HPWL

Στο προηγούμενο γράφημα παρουσιάζεται η απόδοση του αλγορίθμου σε σχέση με την το μετρικό της ημιπεριμέτρου καλωδίου (HPWL) ανά διάφορες γενεές σε ποσοστιαία κλίμακα, ενώ φαίνεται πως η βελτίωση απόδοσης του γενετικού αλγορίθμου γίνεται μεγαλύτερη με την αύξηση του αριθμού των γενεών, γεγονός που επικυρώνει την καλή λειτουργία του γενετικού αλγορίθμου.



Εικόνα 14: Σύγκλιση αποτελέσματος για το benchmark ibm10

Στο τελευταίο γράφημα φαίνεται η σύγκλιση του καλύτερου χρωμοσώματος ανά τις γενεές εκτέλεσης του αλγορίθμου πάνω στο πρότυπο κύκλωμα ibm10. Μετά από μια απότομη βελτίωση στην χιλιοστή γενιά, το αποτέλεσμα συγκλίνει σταδιακά προς την τελική λύση.

Συμπεράσματα και μελλοντικές προεκτάσεις

Ο γενετικός αλγόριθμος που παρουσιάστηκε σε αυτό το κεφάλαιο μπορεί να εφαρμοστεί είτε ως ένα βήμα πριν την έναρξη της χωροθέτησης, είτε ως ένα ξεχωριστό βήμα της σχεδίασης. Ο αλγόριθμος βρίσκει πιο κατάλληλες θέσεις τοποθέτησης για τα I/O pins βάση του μετρικού του συνολικού μήκους καλωδίου, ενώ διατηρεί τις προηγούμενες θέσεις των cells που έχουν τοποθετηθεί στην κεντρική περιοχή της σχεδίασης, αποφεύγοντας έτσι χρονοβόρες διαδικασίες re-placement. Τα αποτελέσματα των πειραμάτων δείχνουν πως μπορεί να επιτευχθεί βελτίωση έως και 3% σε μερικές περιπτώσεις.

Παραλλάσσοντας την διαδικασία αναπαραγωγής (crossover operator), είτε χρησιμοποιώντας άλλες πιο σύνθετες, ενώ παράλληλα γίνει προσθήκη παραπάνω μεταβλητών που αφορούν τη διαδικασία ανάθεσης θέσεων I/O pads, είναι βήματα που ενδεχομένως να βελτιώσουν τον ρυθμό σύγκλισης κάνοντας τον αλγόριθμο μια συμβατή λύση για λεπτομερή χωροθέτηση (detailed placement). Επίσης η εφαρμογή μεθόδων παραλληλοποίησης είναι ικανή να μειώσει τον χρόνο εκτέλεσης καθώς και η εφαρμογή άλλων πιο εκλεπτυσμένων μεθόδων απόδοσης του προβλήματος σε μορφή εκτελέσιμου κώδικα.

ΚΕΦΑΛΑΙΟ 4: Bookshelf Format Parser

Εισαγωγή

Η επαφή με διάφορες εργασίες και προσπάθειες πάνω σε διάφορα προβλήματα στο χώρο της Φυσικής Σχεδίασης και πιο συγκεκριμένα στο βήμα της Χωροθέτησης, δημιουργήσαν την ανάγκη για τον αυτοματισμό κάποιων διαδικασιών που ενώ θεωρούνται αμελητέες αποδεικνύονται αρκετά χρονοβόρες. Η συνήθης μορφή των πρότυπων αρχείων που χρησιμοποιούνται ως η βάση των πειραμάτων ονομάζεται ως «μορφή βιβλιοθήκης» ή "Bookshelf Format" διότι η εμφάνιση των κύριων υπομονάδων της σχεδίασης cells αποδίδεται με ορθογώνιο σχεδιασμό ενώ η συνολική σχεδίαση είναι ένα παραλληλόγραμμο χωρισμένο με οριζόντιες γραμμές, τις γραμμές/rows της σχεδίασης. Έτσι όταν τα cells τοποθετηθούν πάνω στη σχεδίαση, αυτή θυμίζει το σχήμα βιβλιοθήκης.

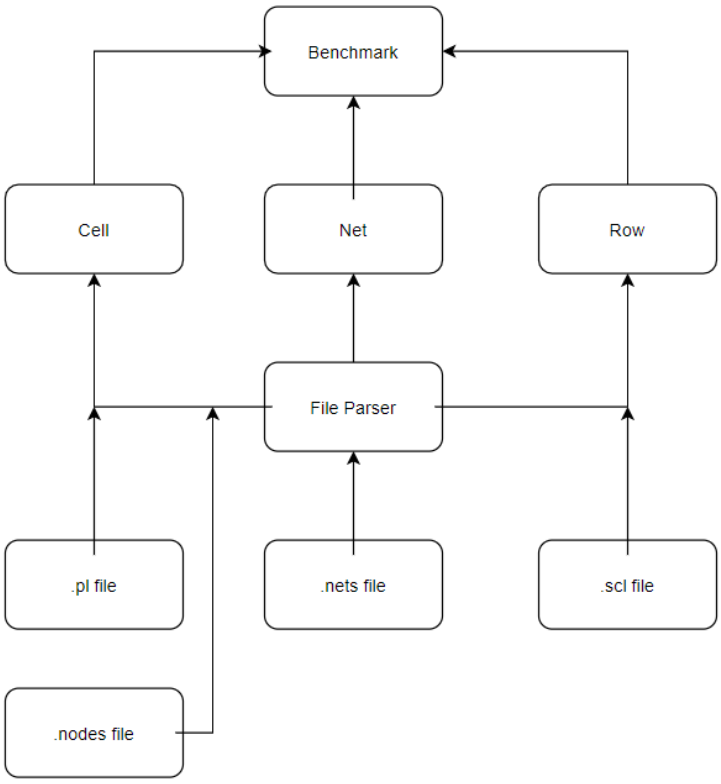
Στη συγκεκριμένη απόδοση, το κύκλωμα αποδίδεται σε διάφορα αρχεία κειμένου τα οποία περιέχουν τις απαραίτητες πληροφορίες για την μεταφορά της σχεδίασης στον υπολογιστή και σε μορφή βολική για κάποιον που ασχολείται με το σχεδιασμό και το placement. Αναλυτικότερα, η τυπική αυτή μορφή περιέχει τα αρχεία με καταλήξεις .pl, .nets, .nodes, .scl, .aux, .wts. Αντιστοίχως τα αρχεία αυτά περιέχουν τις συντεταγμένες κάθε cell και pin της σχεδίασης, τα nets τα οποία υπάρχουν στη σχεδίαση καθώς και τα περιεχόμενα cells και pins τους, οι διαστάσεις των cells και pins, οι διαστάσεις και οι συντεταγμένες των γραμμών που θα σχηματίσουν τον πυρήνα της σχεδίασης και τέλος δύο αρχεία τα οποία δεν είναι τόσο κρίσιμα όσο τα άλλα καθώς περιέχουν με τη σειρά τη λίστα όλων των προηγούμενων αρχείων που παρουσιάστηκαν, και ένα σύνολο με τα βάρη των cells της σχεδίασης. Αν και το διάβασμα των αρχείων σε μορφή προγράμματος είναι μια τετριμμένη διαδικασία για κάθε προγραμματιστή, γίνεται εξαιρετικά χρονοβόρο σε κάποιες περιπτώσεις καθώς σε αυτό πρέπει να συμπεριληφθεί το γεγονός πως κάθε ένα από αυτά τα αρχεία θα πρέπει όχι μόνο να διαβάζεται αλλά και να αποδίδεται με κατάλληλο τρόπο σε δομές που θα είναι βολικές για χρήση στην πραγματική κεντρική ροή κάθε προσέγγισης στο πρόβλημα της χωροθέτησης, συνυπολογίζονται ότι τις περισσότερες φορές οι σημαντικές πληροφορίες είναι κάποια μεταδεδομένα που υπολογίζονται με βάση αυτές αφού γίνει το διάβασμά τους, έχοντας ως παράδειγμα μία από τις βασικότερες τιμές σε κάθε πείραμα, το συνολικό μήκος καλωδίου που υπολογίζεται βρίσκοντας κάθε ακρογωνιαίο cell σε κάθε net κι έπειτα υπολογίζοντας την ημιπερίμετρο για κάθε ένα από αυτά τα νοητά ορθογώνια nets. Το άθροισμα όλων αυτών είναι η τιμή της ημιπεριμέτρου καλωδίου HPWL.

Η πρώτη προσπάθεια αυτοματοποίησης αυτών των διαδικασιών έγινε με το εργαλείο PyPUT του δευτέρου κεφαλαίου. Αν και περιέχει σχεδόν όλες τις διαδικασίες αυτές, αποδίδοντας τα αποτελέσματα σε πολλές μορφές δεδομένων προς χρήση εξακολουθεί να είναι στο χέρι του χρήστη η επιλογή όλων αυτών των συναρτήσεων αφού είναι γραμμένες σε μορφή συναρτησιακού κώδικα. Επίσης κάθε φορά που θα εκτελείται ένα πείραμα ή κάποιος αλγόριθμος θα πρέπει αυτές οι συναρτήσεις να τρέξουν ως προεργασία πριν την αρχή του, κάτι που θα συμπεριληφθεί στο χρόνο εκτέλεσης και μπορεί να είναι χρονοβόρο ανάλογα με το μέγεθος της σχεδίασης.

Με βάση το PyPUT γράφτηκε ένα εργαλείο parsing το οποίο ακολουθεί την αντικειμενοστραφή μορφή προγραμματισμού και περιέχει όλες τις απαραίτητες πληροφορίες σε μία μεταβλητή/αντικείμενο.

Σχεδιασμός και υλοποίηση

Όπως αναφέρθηκε πριν, το εργαλείο ακολουθεί την μορφή αντικειμενοστραφούς προγραμματισμού, την απόδοση δηλαδή διαφόρων μεταβλητών που περιέχουν σχεδόν όμοια δομή και χρήση σε γενικότερες κλάσεις οι οποίες θα περιέχουν χαρακτηριστικά που θα αποδίδονται σε κάθε αντικείμενό τους. Για παράδειγμα δύο ξεχωριστά άτομα μπορούν να περιγραφούν ως άνθρωποι, και οι δύο έχουν ένα συγκεκριμένο ύψος και βάρος. Αν λοιπόν δημιουργήσουμε έναν άνθρωπο, αυτομάτως θα έχει και τα χαρακτηριστικά του ύψους και του βάρους στα οποία μπορούμε να προσδώσουμε μια τιμή στη συνέχεια. Ο συγκεκριμένος παραλληλισμός είναι αρκετά σημαντικός, αφού η τελική σχεδίαση ως σύνολο, αποτελεί ένα κύκλωμα που θα έχει ως χαρακτηριστικά τα nets, τα cells, τα pins, γραμμές, διαστάσεις και συντεταγμένες, καταλήγοντας έτσι σε μία μοναδική μεταβλητή που θα συμπεριλαμβάνει όλες τις απαραίτητες πληροφορίες. Σημαντική είναι επίσης η δομή της γλώσσας Python για την ανάπτυξη του εργαλείου, αφού στη βάση της είναι αντικειμενοστραφής και όχι συναρτησιακή, κάνοντας την χρήση αντικειμένων πιο αποδοτική σε χρήση μνήμης και ταχύτητας ανάκλησης δεδομένων σε αντίθεση με τη χρήση άλλων απλών δομών, όπως απλές λίστες και λεξικά.



Εικόνα 15: Σχεδιασμός του εργαλείου

Τη βάση του εργαλείου αποτελεί η κλάση του File Parser ο οποίος περιέχει συναρτήσεις του εργαλείου PyPUT και διαβάζοντας τα αρχεία του σχήματος, παράγει τις κατάλληλες πληροφορίες για την δημιουργία των υπο-κλάσεων που θα σχηματίσουν τη συνολική σχεδίαση. Πιο αναλυτικά η συγκεκριμένη κλάση έχει ως μόνο χαρακτηριστικό τη διαδρομή του φακέλου που βρίσκεται το σύνολο αυτών των αρχείων. Έπειτα με τη χρήση των μεθόδων read_cells(), read_nets() και read_rows() επιστρέφονται, το σύνολο των πληροφοριών για το σχηματισμό κάθε cell, net και row, τα κύρια συστατικά κάθε σχεδίασης.

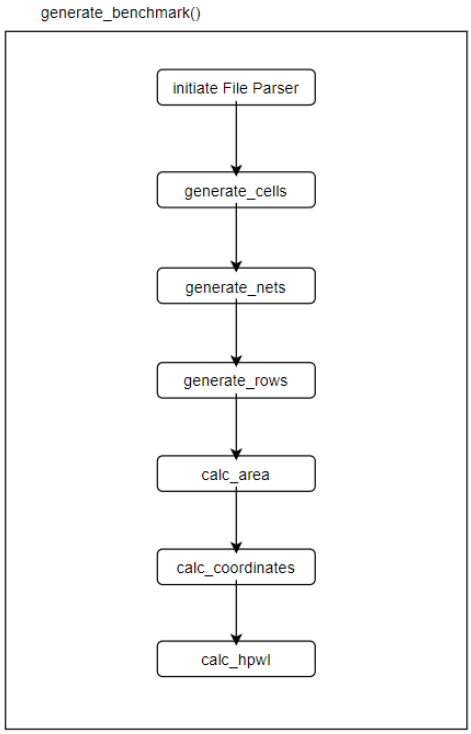
Η πρώτη από τις τρείς κλάσεις που αναπαριστούν τα κύρια συστατικά της σχεδίασης είναι η κλάση Cell. Όπως φανερώνει το όνομά της η κλάση αυτή περιγράφει τα κελιά του κυκλώματος και έχει τα εξής χαρακτηριστικά. Αρχικά υπάρχει ένα σετ από χαρακτηριστικά-μεταβλητές τα οποία διαβάζονται από τα αρχεία με κατάληξη .pl και .nodes. Αυτά τα χαρακτηριστικά είναι το όνομα, η χαμηλότερη οριζόντια συντεταγμένη low_y καθώς και η αριστερότερη κατακόρυφη συντεταγμένη left_x. Επιπλέον δίνονται το πλάτος και το ύψος και το δεδομένο αν το κελί μετακινείται ή όχι, κοινώς αν είναι μετακινήσιμο-cell ή αν είναι τερματικό-pin ή κελί που δεν μετακινείται (macro). Στη συνέχεια με βάση τα προηγούμενα χαρακτηριστικά υπολογίζονται η μέγιστη οριζόντια συντεταγμένη high_y και η δεξιότερη κατακόρυφη συντεταγμένη right_x. Δίνεται τέλος ένα αναγνωριστικό id το οποίο έχει τον αύξων αριθμό αντικειμένων και υπάρχει για λόγους αποσφαλμάτωσης. Την κλάση ολοκληρώνουν οι μέθοδοι που επιτελούν στον υπολογισμό των προηγούμενων συντεταγμένων καθώς και η μέθοδος που αυτοματοποιεί την δημιουργία του ενός αντικειμένου της κλάσης αυτής.

Η δεύτερη κλάση είναι αυτή που περιγράφει τα nets του κυκλώματος. Τα χαρακτηριστικά είναι παρόμοια με αυτά της προηγούμενης κλάσης με τη προσθήκη των cells, μια λίστα που περιέχει αντικείμενα της προηγούμενης κλάσης και δείχνει ποια cells υπάρχουν μέσα σε ένα net. Η μεταβλητή net_degree που δείχνει τον αριθμό των cells καθώς η μεταβλητή hpwl που είναι η ημιπερίμετρος καλωδίου μόνο για ένα net. Η συγκεκριμένη είναι αρκετά σημαντική καθώς αν στην αρχικοποίηση κάθε net αυτοματοποιείται ο υπολογισμός αυτής της τιμής τότε για το συνολικό μήκος καλωδίου μένει μόνο να γίνει το άθροισμα αυτών των μεταβλητών. Σημαντική είναι η μέθοδος calculate_net_corners(), η οποία είναι συνδυασμός συναρτήσεων του εργαλείου PyPUT. Η μέθοδος προσπελαύνει όλα τα cells τα οποία ανήκουν σε ένα net, βρίσκει αυτά που ορίζουν τις γωνίες τους και πρακτικά διαμορφώνουν τις τέσσερις άκρες του νοητού ορθογωνίου. Οι συντεταγμένες αυτές είναι απαραίτητες για την επόμενη μέθοδο, calculate_hpwl() που υπολογίζει τη ημιπερίμετρο για ένα net. Τέλος υπάρχει επίσης μια μέθοδος δημιουργίας ενός net, αυτοματοποιώντας την κλήση των παραπάνω μεθόδων σε κλήση μόνο μίας, δημιουργώντας παράλληλα το στιγμιότυπο ενός net.

Η τελευταία υποκλάση, είναι η Row και διαμορφώνει τις γραμμές της σχεδίασης. Προφανώς τα χαρακτηριστικά της είναι οι συντεταγμένες που διαμορφώνουν την περιοχή κάθε γραμμής, μια μεταβλητή που ορίζει τον αριθμό των τοποθετημένων cells σε αυτή καθώς και η τιμή που ορίζει το κενό μεταξύ αυτών των διαδοχικών cells. Οι μέθοδοι που υπάρχουν στην κλάση υπολογίζουν τις υπόλοιπες συντεταγμένες που δεν υπάρχουν στο αντίστοιχο αρχείο ως δεδομένα και αυτοματοποιούν την δημιουργία ενός στιγμιότυπου.

Η κύρια κλάση του εργαλείου ονομάζεται Benchmark και είναι αυτή η οποία συνδυάζει όλες τις υπόλοιπες, υπολογίζοντας δεδομένα και πληροφορίες τα οποία

μπορεί να αποδειχθούν χρήσιμα για οποιονδήποτε το χρησιμοποιεί καθώς και μετατρέπει όλη τη σχεδίαση σε μορφή μίας μεταβλητής η οποία περιέχει όλες τις απαραίτητες πληροφορίες της σχεδίασης. Αρχικά η κλάση έχει τις συντεταγμένες και τις διαστάσεις του κυκλώματος ως χαρακτηριστικά καθώς και λίστες/λεξικά (dictionaries) που περιέχουν τα cells, nets, rows και pins του κυκλώματος. Τέλος υπάρχει η μεταβλητή hpwl που μετράει το συνολικό μήκος καλωδίου. Η δημιουργία ενός αντικειμένου benchmark γίνεται καλώντας τις μεθόδους αυτής της κλάσης, ως εξής. Αρχικά μέσω της μεθόδου generate_benchmark_cells() χρησιμοποιώντας τα δεδομένα του parser, δημιουργούνται αντικείμενα της κλάσης cell ίσα σε πλήθος με τον αριθμό που αναγράφεται στο αντίστοιχο αρχείο για κάθε πρότυπο κύκλωμα. Ομοίως κατασκευάζονται τα nets και τα rows της σχεδίασης με τις κλήσεις των μεθόδων generate_benchmark_nets() και generate_benchmark_rows(). Έπειτα υπολογίζονται η περιοχή και οι συντεταγμένες του κυκλώματος. Τέλος αφού σε κάθε net υπολογίστηκε με την δημιουργία του, το hpwl, δημιουργείται μια ρουτίνα υπολογισμού του συνολικού hpwl που ουσιαστικά αθροίζει τις προαναφερθείσες τιμές. Σημαντικό είναι να αναφερθεί πως ο υπολογισμός του μήκους καλωδίου σε αυτό το εργαλείο είναι εμφανώς ταχύτερος από αυτόν της συνάρτησης στο εργαλείο PyPUT. Αυτό παρατηρείται για δύο λόγους. Πρώτον, στην αντικειμενοστραφή μορφή ένα net έχει ήδη αποθηκευμένη μια λίστα με τα cells του. Η λίστα αυτή δεν θα αλλάξει ακόμη κι αν αλλάξουν οι συντεταγμένες ενός cell, ενώ ενημερώνεται αυτόματα καθώς οι τιμές σε επίπεδο αντικειμένου είναι συνδεδεμένες με τη μνήμη και ανανεώνονται αυτόματα. Επιπρόσθετα ο parser καλείται μόνο μία φορά σε αντίθεση με τη συναρτησιακή μορφή που θα έπρεπε να καλείται κάθε φορά που χρειάζεται ο υπολογισμός του hpwl μαζί με τις συναρτήσεις για τη δημιουργία cells και nets, φόρτο αρκετά βαρύ αν σκεφτεί κανείς πως είναι συναλλαγές σε επίπεδο σκληρού δίσκου. Δεύτερον, όπως αναφέρθηκε στην εισαγωγή, η Python είναι στη βάση της γλώσσα αντικειμενοστραφούς προγραμματισμού, ως αποτέλεσμα η διαχείριση αντικειμένων σε επίπεδο μνήμης είναι καλύτερη και πιο αποδοτική ακόμη κι αν αυτά περιέχουν λίστες ή/και λεξικά, σε σχέση με απλά λεξικά/λίστες στο κύριο επίπεδο μνήμης. Προφανώς αυτό είναι πιο κατανοητό αν γνωρίζει κάποιος τη διασύνδεση κύριας και κρυφής μνήμης σε επίπεδο αρχιτεκτονικής υπολογιστών, αλλά το συγκεκριμένο θέμα δεν είναι προς ανάλυση στην παρούσα εργασία.



Εικόνα 16: Ροή της μεθόδου generate_benchmark()

Οι διαδικασίες και μέθοδοι που αναλύθηκαν στην προηγούμενη παράγραφο καλούνται μέσω της μεθόδου generate_benchmark() η οποία αυτοματοποιεί όλα όσα αναφέρθηκαν μέχρι στιγμής και παράγει ένα και μοναδικό αντικείμενο το οποίο θα περιέχει τα πάντα. Η διαδικασία αυτή φαίνεται στη ροή της παραπάνω εικόνας. Έτσι πλέον ο χρήστης χρειάζεται να γράψει 4 σειρές προγράμματος για την αρχικοποίηση οποιουδήποτε κυκλώματος σε μορφή bookshelf.

```
from c_benchmark import *

path = "file_path/benchmark_name ... NO EXTENSION"
m = Benchmark(path)
m.generate_benchmark()
```

Στην πρώτη γραμμή είναι απαραίτητη η κλήση του εργαλείου μέσω της εντολής import της Python και του αρχείου c_benchmark.py. Στη συνέχεια ο χρήστης εισάγει το μονοπάτι του φακέλου όπου βρίσκονται τα αρχεία του πρότυπου κυκλώματος, χωρίς να χρειάζεται να τα αναφέρει αναλυτικά. Τέλος ορίζει μια μεταβλητή-αντικείμενο, δημιουργώντας ένα άδειο στιγμιότυπο της κλάσης benchmark και καλεί τη μέθοδο generate_benchmark() για την αυτοματοποίηση της διαδικασίας. Πλέον όλες οι πληροφορίες βρίσκονται στο αντικείμενο m. Αν για παράδειγμα ο χρήστης καλέσει τη εντολή print(m.cells.keys()) θα εμφανιστεί μια λίστα με όλα τα ονόματα των cells που υπάρχουν στο κύκλωμα ή αν χρειαστεί το hwpl καλεί το m.hpwl. Η σύνδεση των αντικειμένων καθώς και όλα τα

χαρακτηριστικά τους είναι πιο εμφανής οπτικά αν κάποιος χρησιμοποιήσει κάποιο εξειδικευμένο IDE με τη χρήση debugger.

Συμπεράσματα και μελλοντικές προσθήκες

Η χρήση του εργαλείου προσπέλασης αρχείων της μορφής bookshelf αποδείχθηκε αναγκαία σε διάφορες προσπάθειες υλοποίησης αλγορίθμων για Χωροθέτηση. Σε αντίθεση με το εργαλείο PyPUT, ένας χρήστης μπορεί πολύ πιο εύκολα να κατανοήσει τον τρόπο με τον οποίο θα προσαρμόσει το εργαλείο σε δικό του πρόγραμμα ενώ δεν χρειάζεται να εκτελέσει μια σειρά από συναρτήσεις ως προεργασία, γλιτώνοντας έτσι χρόνο από τη διαδικασία υλοποίησης καθώς και εκτέλεσης. Όμως το PyPUT αποτελεί μια ολοκληρωμένη σουίτα συναρτήσεων και παρέχει λειτουργίες που δεν έχουν υλοποιηθεί στον parser. Σε μια μελλοντική επέκταση, ο parser θα μπορούσε να αποτελέσει τη βάση του εργαλείου PyPUT, ενώ οι συναρτήσεις αυτού θα υλοποιηθούν σε μορφή μεθόδων αντικειμενοστραφούς προγραμματισμού. Έτσι το PyPUT θα αποκτήσει την μορφή που έχουν όλα τα πακέτα της γλώσσας Python, ενώ παράλληλα θα είναι ακόμη πιο εύκολο να γίνουν προσθήκες επιπλέον λειτουργιών, όχι μόνο από τους δημιουργούς, αλλά και από άλλους χρήστες καθώς αποτελεί πρόγραμμα ανοιχτού λογισμικού (Open Source).

ΣΥΝΟΨΗ

Η ροή της φυσικής σχεδίασης περιέχει ένα μεγάλο αριθμό διακριτών βημάτων. Μεταξύ αυτών, η χωροθέτηση είναι ένα από τα πιο σημαντικά καθώς καθορίζει το τελικό αποτέλεσμα μια σχεδίασης. Η χωροθέτηση χωρίζεται σε τρία διακριτά βήματα, μέσα στα οποία έχουν σχεδιαστεί κι εφαρμοστεί διάφοροι αλγόριθμοι. Τα βήματα αυτά είναι η καθολική χωροθέτηση, η νομιμοποίηση και η λεπτομερής χωροθέτηση. Οι αλγόριθμοι χωροθέτησης έχουν σχεδιαστεί και λειτουργούνε πάνω σε ένα ή και παραπάνω από τα βήματα αυτά με απώτερο στόχο την βελτιστοποίηση της σχεδίασης βάση κάποιας τιμής που συνήθως είναι το μήκος καλωδίου.

Για τη δημιουργία όμως ενός τέτοιου αλγορίθμου, πέραν της σχεδίασης της δικής του λειτουργίας, απαιτείται μια προεργασία κατά την οποία ο σχεδιαστής πρέπει να διαβάσει τα αρχεία του πρότυπου κυκλώματος και να «σχηματίσει» τη σχεδίαση σε μια διαχειρίσιμη μορφή για τον αλγόριθμο που σχεδιάζει. Λόγω της έλλειψης ολοκληρωμένων εργαλείων που εμπίπτουν σε αυτή την κατηγορία, δημιουργήθηκε το PyPUT, το οποίο έχει ως βασικό στόχο τη βοήθεια και τον αυτοματοποίηση αυτών των διεργασιών, προσφέροντας επιπλέον και άλλες λειτουργίες που μπορεί να φανούν χρήσιμες στον σχεδιασμό και την εκτέλεση αλγορίθμων χωροθέτησης. Η εκτέλεση πειραμάτων χρήσης του εργαλείου πάνω σε πρότυπα κυκλώματα οδήγησε στη διαπίστωση της ομαλής λειτουργίας του (σχετίζοντας τον χρόνο εκτέλεσης και το μέγεθος όγκου των εκάστοτε σχεδιάσεων όπως φανερώνουν και τα γραφήματα του εκάστοτε κεφαλαίου).

Οι τυπικοί αλγόριθμοι χωροθέτησης πραγματοποιούν αλλαγές στα δομικά στοιχεία του κυκλώματος τα οποία θεωρούνται «μετακινήσιμα» και πρόκειται για τα κελιά – cells της κάθε σχεδίασης. Όμως υπάρχουν περιπτώσεις στις οποίες γίνεται αλλαγή σε μη – μετακινήσιμα στοιχεία για την επίτευξη καλύτερου αποτελέσματος. Αυτή τη λογική ακολουθεί ο πρότυπος γενετικός αλγόριθμος που αναλύθηκε στο τρίτο κεφάλαιο. Στόχος του αλγορίθμου είναι η μετακίνηση των I/O Pads μιας ήδη νόμιμης σχεδίασης με «φυσικό» τρόπο. Δηλαδή τα σημεία εισόδου/εξόδου της σχεδίασης, τα οποία βρίσκονται στην περιφέρεια, μετακινούνται και αλλάζουν θέσεις μεταξύ τους, έχοντας ως στόχο τη βελτίωση της ημιπερίτρου καλωδίου. Ο σχεδιασμός και η εκτέλεση του αλγορίθμου, έγινε ακολουθώντας τα πρότυπα των γενετικών αλγορίθμων και την προσαρμογή των στοιχείων μιας σχεδίασης πάνω σε αυτά ώστε να επιτευχθεί η σύγκλιση ενός αποτελέσματος. Όπως δείχνουν τα γραφήματα, τα αποτελέσματα εκτέλεσης του αλγορίθμου πάνω στα πρότυπα κυκλώματα είναι ευνοϊκά, ενώ είναι σχετικά μικρός και ο χρόνος εκτέλεσης ανά γενεές του αλγορίθμου, πράγμα που σημαίνει πως η μελέτη και η εφαρμογή πιο εκλεπτυσμένων μεθόδων γενετικών αλγορίθμων θα επιφέρει ακόμη καλύτερα αποτελέσματα.

Ως παράγωγο του γενετικού αλγορίθμου, δημιουργήθηκε και το εργαλείο του τελευταίου κεφαλαίου, ένας parser ο οποίος αυτοματοποιεί εξολοκλήρου τη δημιουργία μιας συνολικής εικόνας για μια σχεδίαση που ακολουθεί το πρότυπο bookshelf. Το εργαλείο απαρτίζεται από τη βάση του PyPUT αλλάζοντας όμως τον τρόπο σχεδίασης από συναρτησιακό σε αντικειμενοστραφή. Έτσι επιτυγχάνεται ή κατά μία έννοια συμπίεση του κυκλώματος σε αντικείμενα τα οποία περιέχουν όλες τις απαραίτητες πληροφορίες ενώ αυτό καταλήγει με τη σειρά στην απεικόνιση όλου

του κυκλώματος σε μία μόνο μεταβλητή. Η συγκεκριμένη μεταβλητή – αντικείμενο δίνει στον σχεδιαστή τον έλεγχο όλου του κυκλώματος με περιεχόμενες τις απαραίτητες μεθόδους για τις αλλαγές και τους υπολογισμούς που ενδέχεται να χρειαστούν κατά τον σχεδιασμό και την εκτέλεση ενός αλγορίθμου χωροθέτησης. Τέλος, το συγκεκριμένο εργαλείο αν και διαθέτει μόνο τα απολύτως απαραίτητα για τη απεικόνιση μια σχεδίασης καθώς σχεδιάστηκε με γνώμονα το μικρό μέγεθος και τον πολύ μικρό χρόνο εκτέλεσης, μπορεί εύκολα να δεχθεί οποιαδήποτε προσθήκη, ενώ με την προσαρμογή όλων των συναρτήσεων που περιέχονται στο PyPUT θα αποτελέσει μια νέα και βελτιωμένη μορφή του.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 356-365, March 1991

[2] 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

[3] 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

[4] 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

[5] 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

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

[7] 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

[8] A. N. Dadaliaris, P. Oikonomou, M. G. Koziri, E. Nerantzaki, T. Loukopoulos, and G. I. Stamoulis, "A connectivity-based legalization scheme for standard cell placement," Circuits and Systems, p. 191, August 2017

[9] M. Pedram, K. Chaudhary, and E. S. Kuh, "I/o pad assignment based on the circuit structure," in 1991 IEEE International Conference on Computer Design: VLSI in Computers and Processors, pp. 314–315, Citeseer, 1991

[10] J. Xiong, Y.-C. Wong, E. Sarto, and L. He, "Constraint driven i/o planning and placement for chip-package co-design," in Asia and South Pacific Conference on Design Automation, 2006., pp. 6–pp, IEEE, 2006

[11] J. Wang, K. K. Muchherla, and J. G. Kumar, "A clustering based area i/o planning for flip-chip technology," in International Symposium on Signals, Circuits and Systems. Proceedings, SCS 2003.(Cat. No. 03EX720), pp. 196–201, IEEE, 2004.

[12] General information on Physical Design, https://en.wikipedia.org/wiki/Physical_design_(electronics)

[13] General information on Placement, https://en.wikipedia.org/wiki/Placement_(electronic_design_automation)

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

[15] 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.

[16] H. Eisenmann and F. M. Johannes. Generic Global Placement and Floorplanning. In DAC, pages 269–274, 1998.

[17] 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.

[18] W. C. Naylor, R. Donelly, and L. Sha. Non-Linear Optimization System and Method for Wire Length and Delay Optimization for an Automatic Electric Circuit Placer. In US Patent 6301693, 2001.

[19] 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.

[20] 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.

[21] ISPD98 benchmark circuits, Available at: http://vlsicad.eecs.umich.edu/BK/Slots/cache/www.public.iastate.edu/~nataraj/ISPD04_Bench.html

[22] ISPD04 benchmark circuits. Available at: http://vlsicad.eecs.umich.edu/BK/Slots/cache/www.public.iastate.edu/~nataraj/ISPD04_Bench.html

[23] Georgios Kranas, George-Chris Tsalamagkakis, Panagiotis Oikonomou, Antonios N. Dadaliaris. “PyPUT: Python-based Placement Utilities Toolset”. SEEDA 2018, Kastoria Greece, (DOI: 10.23919/SEEDA-CECNSM.2018.8544926).

[24] M. Pedram, K. Chaudhary, and E. S. Kuh, “I/o pad assignment based on the circuit structure,” in 1991 IEEE International Conference on Computer Design: VLSI in Computers and Processors, pp. 314–315, Citeseer, 1991.

[25] M. Chrzanowska-Jeske and S.-K. Her, “I/o pad assignment for force-directed placement algorithms,” International Journal of Electronics, vol. 77, no. 4, pp. 467–479, 1994.

[26] J. Xiong, Y.-C. Wong, E. Sarto, and L. He, “Constraint driven i/o planning and placement for chip-package co-design,” in Asia and South Pacific Conference on Design Automation, 2006., pp. 6–pp, IEEE, 2006.

[27] A. Caldwell, A. Kahng, S. Mantik, and I. Markov, “Implications of area-array i/o for row-based placement methodology,” in Proceedings. 1998 IEEE Symposium on IC/Package Design Integration (Cat. No. 98CB36211), pp. 93–98, IEEE, 1998.

[28] J. Wang, K. K. Muchherla, and J. G. Kumar, “A clustering based area i/o planning for flip-chip technology,” in International Symposium on Signals, Circuits and Systems. Proceedings, SCS 2003.(Cat. No. 03EX720), pp. 196–201, IEEE, 2004.

[29] K.-Y. Chao and D. Wong, “Signal integrity optimization on the pad assignment for high-speed vlsi design,” in Proceedings of IEEE International Conference on Computer Aided Design (ICCAD), pp. 720–725, IEEE, 1995.

[30] Roulette Selection, https://en.wikipedia.org/wiki/Fitness_proportionate_selection

[31] George K. Kranas, Taxiarchis G. Kouskouras, Vasileios Dimitriadis, Michael Dossis, Panagiotis Oikonomou, Antonios N. Dadaliaris. “A Novel Genetic Algorithm for I/O Pad Planning Retaining Former Cell Positions”. SEEDA 2020, Corfu Greece, (DOI: 10.1109/SEEDA-CECNSM49515.2020.9221794)