ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

"ΑΛΓΟΡΙΘΜΟΙ ΗΛΕΚΤΡΟΝΙΚΟΥ ΑΥΤΟΜΑΤΙΣΜΟΥ ΤΟΠΟΘΕΤΗΣΗΣ STANDARD CELL ΣΕ ΜΙΚΡΟΗΛΕΚΤΡΟΝΙΚΑ ΚΥΚΛΩΜΑΤΑ"

"ELECTRONIC AUTOMATION ALGORITHMS FOR STANDARD CELL PLACEMENT IN MICROELECTRONIC CIRCUITS"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Κ. Σκετόπουλος

Επιβλέποντες:
Δρ. Σωτηρίου Χρήστος, Αναπληρωτής Καθηγητής, Πανεπιστήμιο Θεσσαλίας
Δρ. Τσομπανοπούλου Παναγιώτα, Επίκουρος Καθηγήτρια, Πανεπιστήμιο Θεσσαλίας

ΒΟΛΟΣ 2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΛΓΟΡΙΘΜΟΙ ΗΛΕΚΤΡΟΝΙΚΟΥ ΑΥΤΟΜΑΤΙΣΜΟΥ ΤΟΠΟΘΕΤΗΣΗΣ STANDARD CELL ΣΕ ΜΙΚΡΟΗΛΕΚΤΡΟΝΙΚΑ ΚΥΚΛΩΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Κ. Σκετόπουλος

Επιβλέποντες:
Δρ. Σωτηρίου Χρήστος, Αναπληρωτής Καθηγητής, Πανεπιστήμιο Θεσσαλίας
Δρ. Τσομπανοπούλου Παναγιώτα, Επίκουρος Καθηγήτρια, Πανεπιστήμιο Θεσσαλίας

Εγκρίθηκε από τη διμελή εξεταστική επιτροπή την $22^{η}$ Σεπτεμβρίου 2015

.....	.....
Σωτηρίου Χρήστος	Τσομπανοπούλου Παναγιώτα
Αναπληρωτής Καθηγητής	Επίκουρος Καθηγήτρια

Διπλωματική διατριβή για την απόκτηση του Διπλώματος του Μηχανικού Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας, στα πλαίσια του Προγράμματος Προπτυχιακών Σπουδών του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας.

Ευχαριστίες

Με την περάτωση της παρούσας διατριβής, θα ήθελα να ευχαριστήσω θερμά τους επιβλέποντες της διπλωματικής εργασίας κ. Σωτηρίου Χρήστο και κα. Τοομπανοπούλου Παναγιώτα για την εμπιστοσύνη που έδειξαν στο πρόσωπό μου, τη συνεχή καθοδήγηση, τις υποδείξεις και παρεμβάσεις, που διευκόλυναν την εκπόνηση της διπλωματικής διατριβής.

Επιπλέον, οφείλω ένα μεγάλο ευχαριστώ στην οικογένειά μου και τους φίλους μου για την αμέριστη υποστήριξη και την ανεκτίμητη βοήθεια που μου παρείχαν καθ' όλη τη διάρκεια των σπουδών μου.

Σκετόπουλος Νικόλαος
Βόλος, 2015

ΠΕΡΙΕΧΟΜΕΝΑ

1 Εισαγωγή στο EDA **20**

1.1 Εισαγωγή 20

1.2 Ιστορία του EDA 21

1.3 Ροή Σχεδίασης VLSI Κυκλωμάτων 22

1.4 Χωροθέτηση 26

1.5 Τοποθέτηση 27

1.5.1 Γενική Τοποθέτηση 28

1.5.2 Λεπτομερής Τοποθέτηση 29

1.5.3 Παράμετροι Τοποθέτησης 30

1.6 Διασύνδεση 33

1.7 Αντικείμενο και Στόχοι της Διπλωματικής Διατριβής 35

2 Επισκόπηση Νομιμοποίησης Στοιχείων (Legalization) **37**

2.1 Εισαγωγή 37

2.2 Tetris Legalizer 39

2.3 Abacus Legalizer 40

2.3.1 Αλγόριθμος Abacus 40

2.3.2 Συνάρτηση PLACEROW 41

2.3.3 Παράδειγμα Abacus 45

2.4 Σύνοψη Κεφαλαίου 49

3 Υλοποίηση και Επέκταση Αλγορίθμου Νομιμοποίησης **50**

3.1 Εισαγωγή 50

3.2 Συνάρτηση Κόστους Μετακίνησης 51

3.3 Έλεγχος Χωρητικότητας Γραμμων 53

3.4 Εξαντλητική και Περιοριστική Εξερεύνηση Γραμμών 56

3.5 Διατάξεις Επιλογής Στοιχείων 56

3.6 Προσέγγιση Επίπεδων και Ιεραρχικών Κυκλωμάτων 57

3.7 Κυκλώματα με Εμπόδια 59

3.7.1 Νομιμοποίηση με Εμπόδια 61

4 Αποτελέσματα **66**

4.1 Αποτελέσματα Κόστους Μετακίνησης και Χρόνου εκτέλεσης μεταξύ των F_{TD} και F_{MD} 67

4.2 Αποτελέσματα Εξαντλητικής (ES) και Βέλτιστης (BS) Εξερεύνησης Γραμμών . . 69

4.3 Αποτελέσματα Διατάξεων Επιλογής Στοιχείων 70

4.4 Επίπεδη και Ιεραρχική Υλοποίηση 72

5 Συμπεράσματα και Μελλοντική Ανάπτυξη **75**

ΒΙΒΛΙΟΓΡΑΦΙΑ **77**

ΚΑΤΆΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

1.1 Ο νόμος του Moore για την αύξηση του πλήθους των στοιχείων ανά κύκλωμα. . . 20
1.2 Ιστορική εξέλιξη των εργαλείων EDA [2]. 22
1.3 Τα βήματα σχεδίασης κυκλωμάτων με επίκεντρο την φυσική σχεδίαση και τα επιμέρους βήματά της [7]. 23
1.4 Ροή Σχεδίασης σε φυσικό επίπεδο [9]. 25
1.5 Χωροθέτηση του μικροεπεξεργαστή Intel Pentium 4 [14]. 26
1.6 Χωροθέτηση ενός κυκλώματος, δεδομένου ενός πλήθους από modules και ενός RTL netlist [7]. 27
1.7 Τοποθέτηση στοιχείων ενός κυκλώματος [7]. 27
1.8 Παραδείγματα τοποθέτησης για τα κυκλώματα (a) ibm01 και (b) adapetc5 [14]. . 28
1.9 Επικάλυψη στοιχείων. 29
1.10 Ευθυγράμμιση στοιχείου σε γραμμές και στήλες. 29
1.11 Κύβος πολυπλοκότητας τοποθέτησης [7]. 31
1.12 Cost Function [7]. 32
1.13 Algorithms [7]. 32
1.14 Netlist Granularity [7]. 33
1.15 Layout Coarseness [7]. 33
1.16 Παράδειγμα διασύνδεσης. 34
1.17 Γενική και λεπτομερής διασύνδεση [14]. 34

2.1 Ευθυγράμμιση στοιχείων σε γραμμές [7]. 38
2.2 Παράδειγμα νομιμοποίησης μίας γενικής τοποθέτησης [11]. 38
2.3 Ψευδοκώδικας Tetris [10]. 39
2.4 Νομιμοποίηση Tetris και κόστος μετακίνησης [11]. 40
2.5 Αλγόριθμος Abacus Legalizer [11]. 41
2.6 Ιδιότητες στοιχείων προς νομιμοποίηση [11]. 42
2.7 Επαναληπτικός υπολογισμός ιδιοτήτων στοιχείων προς νομιμοποίηση [11]. . . . 43
2.8 Κώδικας PlaceRow [11]. Τοποθετεί τα στοιχεία μίας γραμμής σε βέλτιστες θέσεις. 44
2.9 Γενική τοποθέτηση. 45
2.10 Δοκιμαστική τοποθέτηση του Cell 1. 45
2.11 Τελική τοποθέτηση του Cell 1 στη δεύτερη γραμμή με κόστος 1.95. 46
2.12 Δοκιμαστική τοποθέτηση του Cell 5. 46
2.13 Συγχώνευση επικαλυπτόμενων στοιχείων σε ένα group. 47
2.14 Δοκιμαστική και τελική τοποθέτηση του Cell 5. 47
2.15 Δοκιμαστική τοποθέτηση του Cell 3 στη πρώτη γραμμή. 47
2.16 Δοκιμαστική τοποθέτηση του Cell 3 στη δεύτερη γραμμή. 48
2.17 Ευθυγράμμιση του group των στοιχείων Cell 4, Cell 2 και Cell 3 στη δεύτερη γραμμή. 48
2.18 Δοκιμαστική και τελική τοποθέτηση του Cell 3. 48
2.19 Τελική τοποθέτηση όλων των στοιχείων. 49

3.1 Παράδειγμα επίπεδου κυκλώματος. 51
3.2 Παράδειγμα ιεραρχικού κυκλώματος. 51
3.3 Παράδειγμα κυκλώματος με εμπόδια. 52
3.4 Ευκλείδεια απόσταση δύο σημείων. 52
3.5 Διαδικασία υπολογισμού συνολικού κόστους νομιμοποίησης. 53

3.6 Τυχαία τοποθέτηση. 54
3.7 Εξέταση του στοιχείου $CellX$ προς νομιμοποίηση. 54
3.8 Τοποθέτηση εκτός περιθωρίων. 54
3.9 Εκτέλεση συνάρτησης Collapse. 55
3.10 Εξερεύνηση τρίτης γραμμής. 55
3.11 Εμβέλεια εξερεύνησης γραμμών. 56
3.12 Διατάξεις επιλογής στοιχείων. 57
3.13 Επίπεδο και Ιεραρχικό Κύκλωμα. 58
3.14 Ιεραρχική χωροθέτηση. 59
3.15 Νομιμοποίηση ενός υπό-κυκλώματος. 60
3.16 Παράδειγμα κυκλώματος με εμπόδια. 60
3.17 Εύρεση κοντινότερης θέσης για νομιμοποίηση. 61
3.18 Διάταξη σε κυκλώματα με εμπόδια. 62
3.19 Προσέγγιση μετατόπισης στοιχείων. 63
3.20 Αλλαγή διάταξης για νομιμοποίηση με μετατόπιση στοιχείων. 64
3.21 Αλγόριθμος νομιμοποίησης κυκλωμάτων με εμπόδια. 64
3.22 Αλγόριθμος αλυσιδωτής μετατόπισης αριστερά. 65

4.1 Παρδείγματα τυχαίας τοποθέτησης. 67
4.2 Διάγραμμα προσεγγίσεων διπλωματικής εργασίας 67
4.3 Κλάσμα κόστους μετακίνησης $\frac{C_{F_{TD}}}{C_{F_{MD}}}$. 68
4.4 Κλάσμα χρόνου εκτέλεσης των $\frac{F_{TD}}{F_{MD}}$. 69
4.5 Κλάσμα χρόνου εκτέλεσης $\frac{ES}{BS}$. 70
4.6 Κλάσματα κόστους μετακίνησης $\frac{C_{IO}}{C_{DO}}$, $\frac{C_{IO}}{C_{MNO}}$ και $\frac{C_{DO}}{C_{MNO}}$, για τα Κυκλώματα 1 ως 6. 71
4.7 Ποσοστά επιλογής των IO, DO και MNO. 71

4.8 Κλάσμα κόστους μετακίνησης $\frac{C_E}{C_H}$. 72
4.9 Χωροθέτηση του κυκλώματος Κύκλωμα5. 73
4.10 Γενική τοποθέτηση και ιεραρχική νομιμοποίηση του κυκλώματος Κύκλωμα5. . . 73
4.11 Διαφορετικές χωροθετήσεις για το Κύκλωμα5. 74
4.12 Κλάσμα χρόνου εκτέλεσης $\frac{E}{H}$. 74

Περίληψη

Τα σημερινά εργαλεία EDA πραγματοποιούν την τοποθέτηση των στοιχείων τόσο με συνδυαστικούς μεθόδους, όσο και με αναλυτικούς. Ωστόσο, οι αναλυτικοί μέθοδοι καθιστούν την τοποθέτηση μη νόμιμη. Το φαινόμενο αυτό παρουσιάζεται διότι οι αναλυτικοί μέθοδοι μεταχειρίζονται τα στοιχεία ως σημεία στο χώρο. Έτσι, μετά την τοποθέτηση παρατηρούνται επικαλύψεις μεταξύ των στοιχείων και επιπλέον τα στοιχεία λαμβάνουν μη ευθυγραμμισμένες θέσεις στο πλέγμα του κυκλώματος. Για το λόγο αυτό πραγματοποιείται η νομιμοποίηση των στοιχείων εξαλείφοντας τις επικαλύψεις και ευθυγραμμίζοντας τα στοιχεία στο πλέγμα του κυκλώματος. Η διαδικασία αυτή πραγματοποιείται από τους νομιμοποιητές. Στόχος των νομιμοποιητών είναι η ελαχιστοποίηση της μετακίνησης των στοιχείων από τις βέλτιστες θέσεις που έλαβαν από την γενική τοποθέτηση.

Στη παρούσα διπλωματική διατριβή παρουσιάζεται η υλοποίηση, η βελτιστοποίηση και η αξιολόγηση ενός αλγορίθμου για τη νομιμοποίηση των στοιχείων ενός κυκλώματος. Ο νομιμοποιητής που επιλέχθηκε προς υλοποίηση είναι ο Abacus [11] και η επιλογή του πραγματοποιήθηκε λόγω των καλών επιδόσεων σε ό,τι αφορά την ελάχιστη τροποποίηση της βέλτιστης λύσης. Ωστόσο, ο κλασικός αλγόριθμος υποστηρίζει τη νομιμοποίηση μόνο επίπεδων κυκλωμάτων και χωρίς εμπόδια. Επομένως, μελετήθηκαν και υλοποιήθηκαν προσεγγίσεις για την υποστήριξη ιεραρχικών κυκλωμάτων και κυκλωμάτων με εμπόδια.

Επιπλέον, υλοποιήθηκαν διαφορετικές προσεγγίσεις για τα επιμέρους στάδια της νομιμοποίησης, όπως για παράδειγμα του υπολογισμού του κόστους μετακίνησης των στοιχείων. Βελτιστοποιήθηκε ο χρόνο εκτέλεσης του αλγορίθμου χρησιμοποιώντας ευριστικούς αλγόριθμους και μελετήθηκε η υποστήριξη κυκλωμάτων με εμπόδια. Λόγω περιορισμένου χρόνου δεν κατέστη δυνατή η υλοποίηση σε γλώσσα C των προσεγγίσεων για την υποστήριξη εμποδίων που μελετήθηκαν και παρουσιάζονται στη συγκεκριμένη διπλωματική διατριβή. Έπειτα, πραγματοποιήθηκαν πειράματα και έγινε σύγκριση των αποτελεσμάτων σε ό,τι αφορά το κόστος μετακίνησης και τον χρόνο εκτέλεσης της νομιμοποίησης. Τέλος, ο νομιμοποιητής ενσωματώθηκε σε ένα υπό ανάπτυξη βιομηχανικό εργαλείο EDA, λαμβάνοντας υπόψιν τους βιομηχανικούς περιορισμούς.

Abstract

Nowadays EDA tools use both combinatorial and analytical methods to place circuits' components. However, analytical methods render placement illegal. This phenomenon occurs because analytical methods use components as physical points. Thus, after global placement components may overlap each other and receive non-aligned positions in the circuit's grid. For this reason, legalization is used by eliminating components' overlapping and aligning them in the circuit's grid. The aim of legalizers is to minimize the components' movement from the positions which they had at global placement.

In this thesis, the implementation, optimization and evaluation of a legalizer are presented. Abacus legalizer [11] has been chosen to be implemented because of its great performance in terms of minimizing the perturbation of the optimal solution. However, the fundamental algorithm supports legalization only at flat circuits and without blockages. In this way, the fundamental legalization algorithm has been extended to support not only flat circuits, but also hierarchical and circuits with blockages.

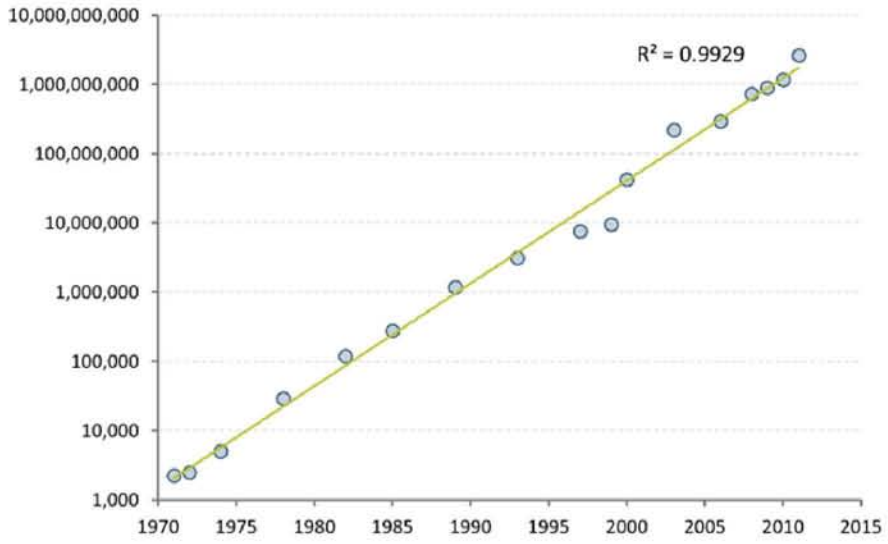
Additionally, different approaches for the individual stages of legalization have been implemented, such as the calculation of the components' movement cost. Execution time of legalization has been optimized by using heuristic algorithms and legalization with blockages has also been studied. Due to time constraints, it has not been possible to develop an algorithm in C language to support blockages, so theoretical approaches have been presented. Following this, experiments have been conducted and the results, in terms of movement cost along with the execution time of legalization have been compared. Finally, the legalizer has been integrated in developing an industrial EDA tool [13], taking into account its restrictions.

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή στο EDA

1.1 Εισαγωγή

Στη σημερινή εποχή οι ολοένα και αυξανόμενες ανάγκες των ανθρώπων για ηλεκτρονικά συστήματα με πολύπλοκες λειτουργίες οδήγησαν στην κατασκευή κυκλωμάτων με δεκάδες διεκατομμύρια τρανζίστορ. Χαρακτηριστικά, σύμφωνα με το νόμο του Moore, το πλήθος των τρανζίστορ ανά ολοκληρωμένο κύκλωμα διπλασιάζεται κάθε 18 μήνες (Σχήμα 1.1).



Σχήμα 1.1: Ο νόμος του Moore για την αύξηση του πλήθους των στοιχείων ανά κύκλωμα.

Η τεράστια αύξηση του πλήθους των στοιχείων σε ένα κύκλωμα κατέστησε ανέφικτη τη χειροκίνητη σχεδίαση τους. Γι' αυτό το λόγο, ακόμα και από τη δεκαετία του '60, ξεκίνησε η δημιουργία εργαλείων για την αυτοματοποίηση της διαδικασίας κατασκευής κυκλωμάτων.

Τα εργαλεία αυτοματοποιημένης σχεδίασης και ελέγχου ηλεκτρικών κυκλωμάτων (EDA) αποτελούν μία κατηγορία λογισμικών, τα οποία χρησιμοποιούνται για τη σχεδίαση κυκλωμάτων

όπως ολοκληρωμένα κυκλώματα (ICs) και τυπωμένα σε πλακέτα κυκλώματα (PCBs). Τα κυκλώματα αυτά, αποτελούν μέρος των περισσοτέρων ηλεκτρικών κυκλωμάτων που κυκλοφορούν στην αγορά όπως τα GPS, PDAs, συστήματα ασφάλειας, mp3 players κτλ. Ωστόσο, μεγαλύτερο ενδιαφέρον παρουσιάζεται στην όλο και αναπτυσσόμενη αγορά των ηλεκτρονικών υπολογιστών και των έξυπνων τηλεφώνων, τα οποία χρησιμοποιούνται καθημερινά από δισεκατομμύρια ανθρώπους.

Ακολουθούν ορισμένα ιστορικά στοιχεία για την εξέλιξη του EDA από τα πρώτα χρόνια δημιουργίας τους έως και σήμερα.

1.2 Ιστορία του EDA

Η ιστορία του EDA ξεκινά μετά την εμφάνιση των PCBs στα μέσα του 1960. Στα πρώιμα αυτά χρόνια, της μετάβασης στην αυτοματοποιημένη σχεδίαση, έγινε σημαντική προσπάθεια δημιουργίας λογισμικών για την επιτάχυνση της διαδικασίας σχεδίασης.

Ωστόσο, δεν ήταν μέχρι τα μέσα της δεκαετίας του '70, όπου αναπτύχθηκαν λογισμικά ικανά για την σχεδίαση τόσο PCBs όσο και ICs με χρήση πιο αποτελεσματικών αλγορίθμων, παρέχοντας παράλληλα περισσότερες δυνατότητες στους σχεδιαστές και τους ελεγκτές των κυκλωμάτων [7]. Κατά τη διάρκεια αυτής της περιόδου, πραγματοποιήθηκε τόσο έρευνα όσο και ανάπτυξη στα εργαλεία EDA από διάφορες εταιρίες. Χαρακτηριστικό παράδειγμα η Bell Labs, Hewlett Packard, IBM, Intel, και Tektronix.

Τα πρώτα εργαλεία αναπτύχθηκαν για την προσομοίωση (π.χ. SPICE) και την επαλήθευση των κυκλωμάτων. Τα εργαλεία προσομοίωσης ελέγχουν αν η λογική σχεδίαση του κυκλώματος (logic design) ικανοποιεί τις προδιαγραφές που έχουν οριστεί πριν τη διαδικασία σχεδίασης του κυκλώματος. Από την άλλη, τα εργαλεία επαλήθευσης λαμβάνουν ως είσοδο τα μοντέλα προσομοίωσης και αν η φυσική σχεδίαση του κυκλώματος συνάδει με τους κανόνες σχεδίασης παράγει γραφικές αναπαραστάσεις της φυσικής υπόστασης του κυκλώματος. Παρόλα αυτά, τα παραπάνω αποτελούν μόνο εργαλεία ελέγχου και όχι επιτάχυνσης της διαδικασίας σχεδίασης. Έτσι, δημιουργήθηκαν εργαλεία για τη σχεδίαση σε επίπεδο πυλών καθώς και για την τοποθέτηση και διασύνδεση των στοιχείων του κυκλώματος (place and route - P&R). Επιπλέον, αναπτύχθηκαν γλώσσες προγραμματισμού για την ευκολότερη περιγραφή του υλικού (HDL) όπως η Verilog και η VHDL.

Παρά το γεγονός ότι ένα μεγάλο μέρος της πρόωρης έρευνας και ανάπτυξης του EDA έγινε στις εταιρίες το 1960 και το 1970, κορυφαία πανεπιστήμια δημιούργησαν μεγάλες ερευνητικές ομάδες. Αυτό σηματοδοτεί επίσης το χρονικό πλαίσιο εντός του οποίου το EDA ξεκίνησε πλέον ως μια βιομηχανία με εταιρείες όπως η Daisy Systems, Mentor Graphics και Valid Logic Systems στις αρχές του 1980.

Στις επόμενες δεκαετίες η βιομηχανία του EDA συνέχισε τη ανοδική πορεία με την εισαγωγή επιπλέον δυνατοτήτων στα εργαλεία (π.χ. σχεδίαση 3D κυκλωμάτων, δημιουργία εργαλείων για επαλήθευση (ATPG)) και βοηθώντας τους χρήστες τους να διασφαλίσουν ταχύτερη και εγκυρότερη σχεδίαση και παραγωγή κυκλωμάτων [14]. Ακολουθεί το χρονοδιάγραμμα εξέλιξης του EDA σε ό,τι αφορά την φυσική σχεδίαση.

Time Period	Circuit and Physical Design Process Advancements
1950-1965	Manual design only.
1965-1975	Layout editors, e.g., place and route tools, first developed for printed circuit boards.
1975-1985	More advanced tools for ICs and PCBs, with more sophisticated algorithms.
1985-1990	First performance-driven tools and parallel optimization algorithms for layout; better understanding of underlying theory (graph theory, solution complexity, etc.).
1990-2000	First over-the-cell routing, first 3D and multilayer placement and routing techniques developed. Automated circuit synthesis and routability-oriented design become dominant. Start of parallelizing workloads. Emergence of physical synthesis.
2000-now	Design for Manufacturability (DFM), optical proximity correction (OPC), and other techniques emerge at the design-manufacturing interface. Increased reusability of blocks, including intellectual property (IP) blocks.

Σχήμα 1.2: Ιστορική εξέλιξη των εργαλείων EDA [2].

1.3 Ροή Σχεδίαοης VLSI Κυκλωμάτων

Η διαδικαοία οχεδίαοης μεγάλων κυκλωμάτων (VLSI) είναι εξαιρετικά πολύπλοκη και μπορεί να διαχωριστεί οε διακριτές λειτουργίες. Η αλυοιδωτή σύνδεοη αυτών των λειτουργιών θα παράξει το τελικό κύκλωμα το οποίο θα πληροί τις προδιαγραφές και θα είναι λειτουργικό.

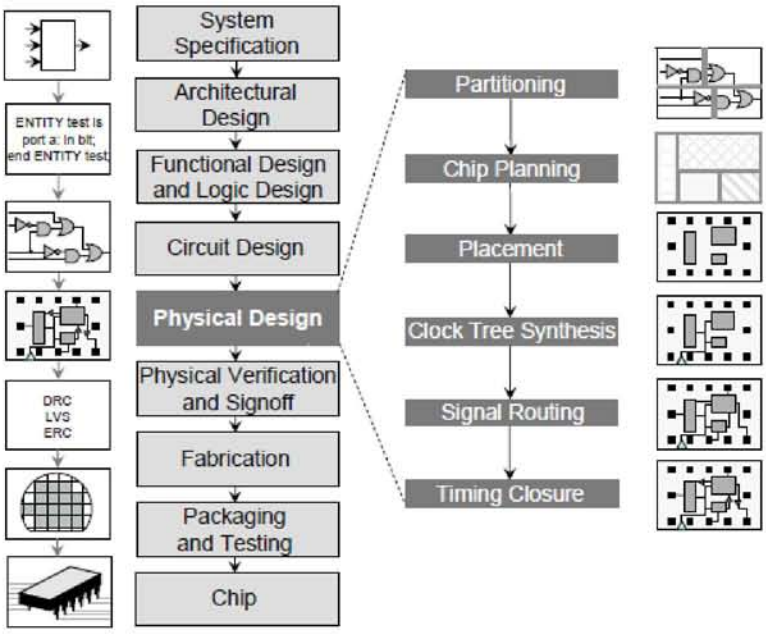
Στο οημείο αυτό παρουοιάζονται τα βήματα οχεδίαοης VLSI κυκλωμάτων όπως απεικονίζονται στο Σχήμα 1.3 [7].

- **Περιορισμοί Συστήματος (System specification)**.

  Οι αρχιτέκτονες και οχεδιαστές κυκλωμάτων, οι μάνατζερ προϊόντων καθώς και οι οχεδιαστές βιβλιοθηκών ορίζουν τους περιορισμούς και τους στόχους που επιθυμούν να πληροί το κύκλωμα που θα καταοκευαστεί. Οι στόχοι και οι απαιτήοεις καλύπτουν τη λειτουργικότητα, τις επιδόοεις, τις φυοικές διαστάοεις και την τεχνολογία παραγωγής.

- **Αρχιτεκτονική Σχεδίαοη (Architectural design)**.

  Η αρχιτεκτονική οχεδίαοη, πρέπει να πληροί τους περιοριομούς ουοτήματος και ορίζει τις αποφάοεις, μεταξύ των οποίων της ανάπτυξης αναλογικών block, της διαχείριοης μνήμης και των απαιτήοεων ιοχύος.



Σχήμα 1.3: Τα βήματα οχεδίαοης κυκλωμάτων με επίκεντρο την φυοική οχεδίαοη και τα επιμέρους βήματά της [7].

- **Λογική Σχεδίαση (Functional and Logic Design)**.

  Μετά από την αρχιτεκτονική οχεδίαοη, ορίζεται η λειτουργικότητα και η διαούνδεοη των modules του κυκλώματος. Να οημειωθεί οε αυτό το οημείο ότι ένα module είναι ένα ούνολο από λογικά οτοιχεία του κυκλώματος. Η λογική οχεδίαοη πραγματοποιείται οε επίπεδο τρανζίστορ (register-transfer level - RTL) χρηοιμοποιώντας γλώοοες περιγραφής υλικού (HDL). Τα HDL modules πρέπει να είναι ορισμένα πλήρως καθώς και να έχει επαληθευτεί η ορθότητά τους.

  Στο οημείο αυτό αρχίζει να γίνεται ξεκάθαρη η ουμβολή των εργαλείων EDA οτη οχεδίαοη κυκλωμάτων. Τα εργαλεία μετατρέπουν με αυτόματο τρόπο τον κώδικα HDL (με τη χρήοη βιβλιοθηκών) οε χαμηλού επιπέδου κύκλωμα. Το κύκλωμα μπορεί να οπτικοποιηθεί και να εξαχθεί η λειτουργικότητά του υπό μορφή οημάτων (waveform).

- **Σχεδίαση Επιπέδου Κυκλώματος (Circuit Design)**.

  Για το μεγαλύτερο μέρος της ψηφιακής λογικής οτο chip, το εργαλείο λογικής ούνθεοης μετατρέπει αυτόματα Boolean εκφράοεις οε επίπεδο πυλών ως netlist. Ωοτόοο, ορισμένα οτοιχεία του κυκλώματος πρέπει να οχεδιαοτούν οε επίπεδο τρανζίστορ. Η διαδικαοία αυτή ονομάζεται οχεδίαοη οε επίπεδο κυκλώματος. Παραδείγματος χάριν, οτοιχεία που οχεδιάζονται οε επίπεδο κυκλώματος είναι: RAM blocks, I/O, αναλογικά κυκλώματα και high-speed functions. Η ορθότητα των κυκλωμάτων οε αυτό το επίπεδο πραγματοποιείται με προοομοιωτές κυκλωμάτων όπως το SPICE.

- **Σχεδίαση σε Φυσικό Επίπεδο (Physical Design)**.

  Κατά τη φυσική σχεδίαση, τα στοιχεία του κυκλώματος αποκτούν γεωμετρικές αναπαραστάσεις. Έτσι, οι πύλες (gates), τα τρανζίστορ και τα υπόλοιπα στοιχεία, αναπαρίστανται με συγκεκριμένα σχήματα, μεγέθη και συνδέσεις με άλλα στοιχεία. Το αποτέλεσμα της φυσικής σχεδίασης είναι ένα σύνολο προδιαγραφών κατασκευής του κυκλώματος, οι οποίες πρέπει να εξεταστούν ως προς την ορθότητά τους.

  Η απόδοση, το εμβαδόν, η αξιοπιστία καθώς και η ισχύς του κυκλώματος επηρεάζονται άμεσα από τη φυσική σχεδίαση. Για παράδειγμα, η απόδοση επηρεάζεται από τα μήκη των καλωδίων λόγω της καθυστέρησης τους και το μεγάλο πλήθος vias μειώνει την αξιοπιστία.

  Λόγω της μεγάλης πολυπλοκότητας, η σχεδίαση σε φυσικό επίπεδο χωρίζεται σε βήματα τα οποία παρουσιάζονται στο Σχήμα 1.4.

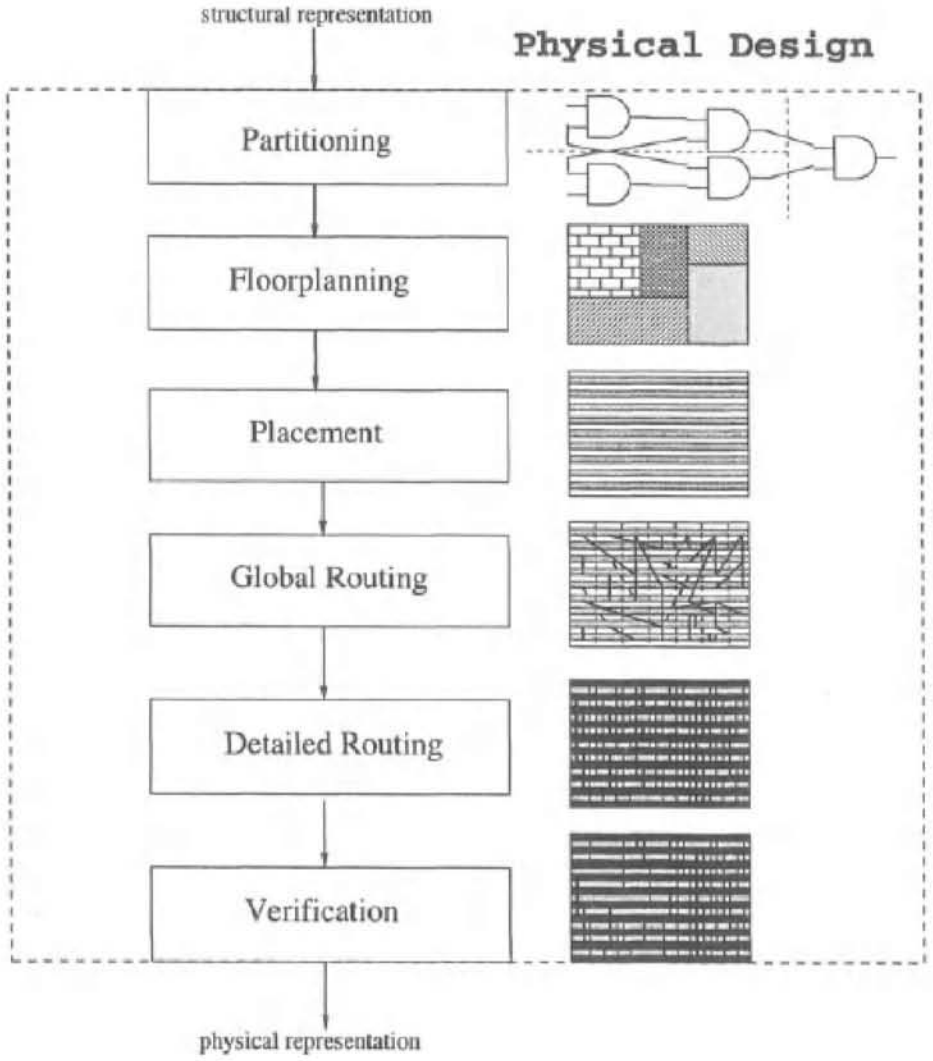
- **Τμηματοποίηση (Partitioning)** του κυκλώματος σε υπό-κυκλώματα ή modules τα οποία μπορούν να σχεδιαστούν και να αναλυθούν ανεξάρτητα.
- **Χωροθέτηση (Floorplanning)** του κυκλώματος καθορίζοντας τα σχήματα των υπό-κυκλωμάτων, τα μεγέθη τους, καθώς και τις θέσεις των εξωτερικών ports και IPs. Ακόμα ορίζονται τα δίκτυα παροχής ρεύματος και γείωσης (VDD και GND). Περισσότερη ανάλυση για τη χωροθέτηση παρουσιάζεται στην Ενότητα 1.4.
- **Τοποθέτηση (Placement)** των στοιχείων του κυκλώματος σε συγκεκριμένες θέσεις. Οι θέσεις επιλέγονται έτσι ώστε να ικανοποιούνται όλες οι προδιαγραφές που έχουν οριστεί στα προηγούμενα βήματα της ροής σχεδίασης. Περισσότερη ανάλυση για την τοποθέτηση παρουσιάζεται στην Ενότητα 1.5.
- **Γενική Διασύνδεση (Global Routing)**, για τη διασύνδεση μονοπατιών διασύνδεσης. Περισσότερη ανάλυση για τη γενική διασύνδεση παρουσιάζεται στην Ενότητα 1.6.
- **Λεπτομερής Διασύνδεση (Detailed Routing)**, ορίζοντας τις συνδέσεις στα διαφορετικά επίπεδα μετάλλων και των μονοπατιών από τη γενική διασύνδεση. Περισσότερη ανάλυση για τη λεπτομερή διασύνδεση παρουσιάζεται στην Ενότητα 1.6.
- **Βελτιστοποίηση (Timing Closure)** της απόδοση του κυκλώματος μετά την τοποθέτηση και τη διασύνδεση.

- **Επαλήθευση (Physical verification)**.

  Μετά τη φυσική σχεδίαση, το κύκλωμα πρέπει να ελεγχθεί για την ορθότητά του τόσο σε λογικό όσο και σε ηλεκτρικό επίπεδο. Ορισμένα προβλήματα που παρουσιάζονται κατά τη φυσική σχεδίαση, μπορούν να παραλειφθούν ή και να διορθωθούν. Ωστόσο, οι αλλαγές που πραγματοποιούνται κατά τη διόρθωση δεν πρέπει να δημιουργήσουν επιπλέον προβλήματα.

- **Κατασκευή (Fabrication)**.

  Στο σημείο αυτό, το κύκλωμα είναι στη διαδικασία φυσικής κατασκευής. Σε ειδικά κατασκευασμένους χώρους (fabs) και με χρήση φωτολιθογραφίας αποτυπώνονται τα σχεδιασμένα κυκλώματα σε δισκία (wafers) με τη χρήση μασκών.



Σχήμα 1.4: Ροή Σχεδίαοης οε φυοικό επίπεδο [9].

Τα ολοκληρωμένα κυκλώματα επανελέγχονται και χαρακτηρίζονται ως λειτουργικά ή ελλαττωματικά. Τα ελαττωματικά μπορεί να μην τηρούν ορισμένες προδιαγραφές (π.χ. συχνότητα λειτουργίας) αλλά να μην παρουσιάζουν κάποιο λειτουργικό πρόβλημα. Έτσι, μπορούν να χρησιμοποιηθούν οε διαφορετικής φύσης μηχανήματα απ' ότι προορίζονταν.

- **Συσκευασία και Τελικός Έλεγχος (Packaging and Testing)**.

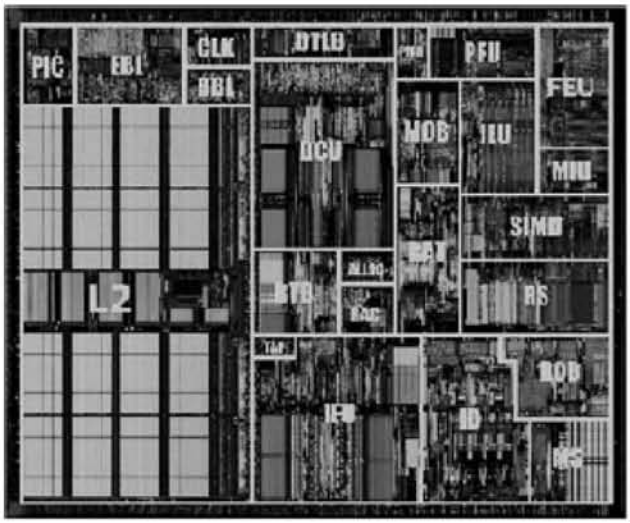
  Η διαδικασία ουοκευαοίας αποτελεί το τελικό στάδιο καταοκευής ενός chip. Η διαδικαοία αυτή, πραγματοποιείται έτοι ώοτε το κύκλωμα να είναι προοτατευμένο από διάφορους εξωγενείς παράγοντες που μπορούν να το καταοτήοουν μη λειτουργικό. Μετά την τοποθέτηοη του κυκλώματος οτο πακέτο (package) γίνεται η ούνδεοη των pins του κυκλώματος με τα pins του πακέτου και το πακέτο οφραγίζεται. Μετά τη ουοκευαοία, το τελικό προϊόν

πρέπει να ελεγχθεί για να εξασφαλιστεί ότι πληροί τις απαιτήσεις του σχεδιασμού, όπως η λειτουργία του (σχέση εισόδου/εξόδου), ο χρονισμός ή η κατανάλωση ισχύος του.

1.4 Χωροθέτηση

Η χωροθέτηση μπορεί να θεωρηθεί το πρώτο, καίριας σημασίας στάδιο, κατά τη διαδικασία φυσικής σχεδίασης ενός VLSI κυκλώματος.

Δεδομένου ενός συνόλου από στοιχεία, είτε με σταθερό σχήμα (hard blocks) είτε με μεταβαλλόμενο (soft blocks), και ενός RTL netlist, η χωροθέτηση καθορίζει το σχήμα των blocks και ορίζει το μέγεθος του chip. Η επιλογή των μεγεθών πραγματοποιείται με γνώμονα την ελαχιστοποίηση ορισμένων συνθηκών, όπως του μήκους καλωδίου (wirelength) ή της συνολικής επιφάνειας που καταλαμβάνει το chip (chip area) [14].



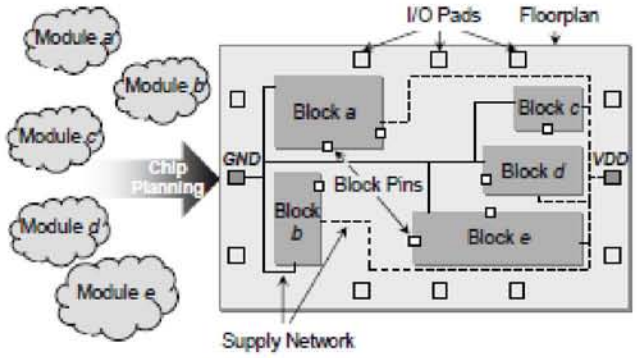
Σχήμα 1.5: Χωροθέτηση του μικροεπεξεργαστή Intel Pentium 4 [14].

Η χωροθέτηση αποτελεί σημαντικό στάδιο στη φυσική σχεδίαση, καθώς στο στάδιο αυτό πραγματοποιούνται εκτιμήσεις σχετικά με το μέγεθος του chip, την καθυστέρηση (delay), τη συμφόρηση των καλωδίων στο κύκλωμα (congestion) και τη θέση των blocks στο chip. Έτσι, ακόμα και από αυτό το αρχικό στάδιο, μπορούν να πραγματοποιηθούν αλλαγές στη σχεδίαση του κυκλώματος για να πληρούνται οι προδιαγραφές.

Στο στάδιο αυτό, πραγματοποιείται και η εύρεση των θέσεων των pins στο κύκλωμα αλλά και η δημιουργία του δικτύου παροχής ενέργειας για την τροφοδότηση του κάθε block [7].

Η χωροθέτηση, αναδεικνύεται όλο και πιο καίριας σημασίας λόγω του όλο και αυξανόμενου πλήθους στοιχείων των κυκλωμάτων. Για να αντιμετωπιστεί η μεγάλη πολυπλοκότητα των σημερινών κυκλωμάτων, χρησιμοποιούνται ιεραρχικά μοντέλα σχεδίασης κυκλωμάτων. Έτσι, στην παραγωγή κυκλωμάτων χρησιμοποιείται η χωροθέτηση για να διαχωρίσει το πρόβλημα του μεγάλου πλήθους στοιχείων σε μικρότερα με την τεχνική ¨Διαίρει και Βασίλευε¨. Γι' αυτό το λόγο,

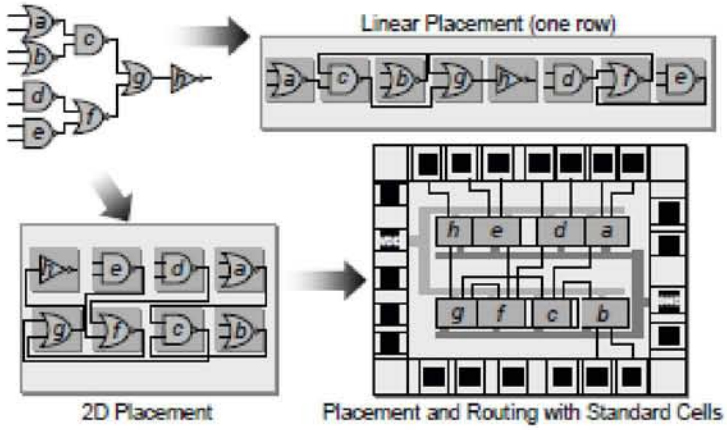
σήμερα, η χωροθέτηση αποτελεί ένα από τα σημαντικότερα βήματα στη διαδικασία σχεδίασης κυκλωμάτων παρά ποτέ.



Σχήμα 1.6: Χωροθέτηση ενός κυκλώματος, δεδομένου ενός πλήθους από modules και ενός RTL netlist [7].

1.5 Τοποθέτηση

Η τοποθέτηση αποτελεί, αν όχι το βασικότερο ένα από τα βασικότερα βήματα στη οχεδίαση κυκλωμάτων. Είναι το σημείο στο οποίο τα στοιχεία του κυκλώματος τοποθετούνται, δηλαδή αποκτούν συγκεκριμένη θέση, πάνω στο chip.



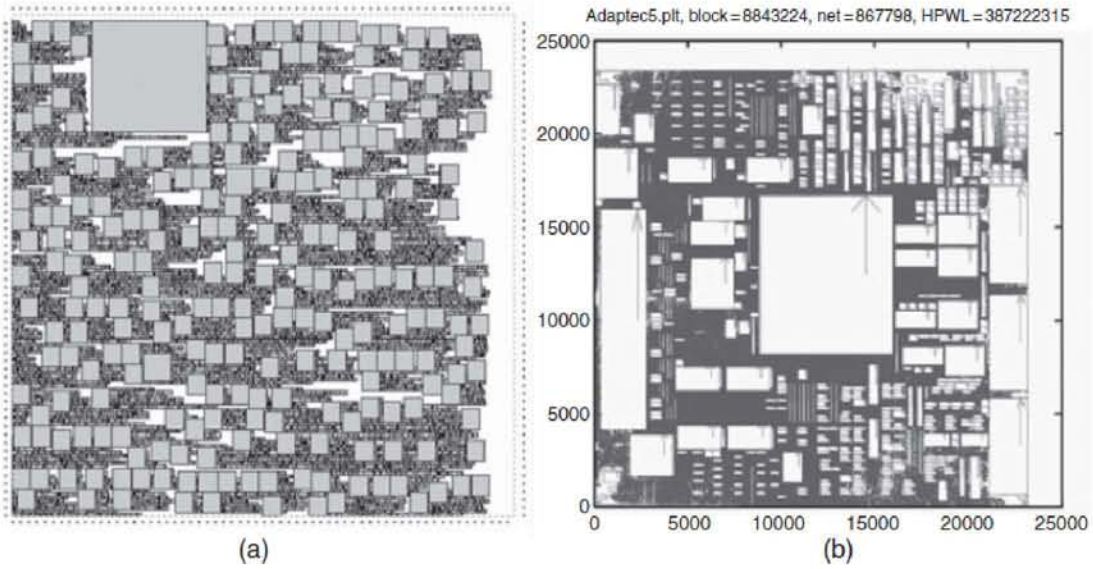
Σχήμα 1.7: Τοποθέτηση στοιχείων ενός κυκλώματος [7].

Τα στοιχεία του κυκλώματος (π.χ. πύλες, standard cells και macro blocks) αποκτούν ορθογώνια σχήματα και αντιπροσωπεύονται σαν κόμβοι ενώ οι διασυνδέσεις τους (nets) ως ακμές (Σχήμα 1.7). Ορισμένα στοιχεία μπορεί να έχουν προκαθορισμένες και σταθερές θέσεις ενώ

άλλα να μπορούν να μετακινούνται (placeable). Ακόμα, η τοποθέτηση χωρίζεται σε δύο βήματα. Τη γενική (global) τοποθέτηση και τη λεπτομερή (detailed).

1.5.1 Γενική Τοποθέτηση

Στη γενική τοποθέτηση τα στοιχεία τοποθετούνται έτσι ώστε όχι μόνο να πληρούνται οι προδιαγραφές που έχουν οριστεί στα προηγούμενα βήματα της σχεδίασης, αλλά και να ελαχιστοποιούνται ορισμένες συναρτήσεις. Οι συναρτήσεις αφορούν τη συνολική βελτιστοποίηση της λειτουργίας του κυκλώματος και ονομάζονται συναρτήσεις κόστους (cost functions).

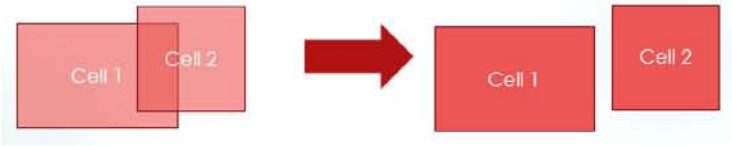


Σχήμα 1.8: Παραδείγματα τοποθέτησης για τα κυκλώματα (a) ibm01 και (b) adapetc5 [14].

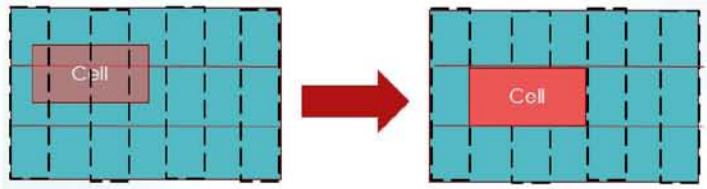
Οι συναρτήσεις κόστους και η τοποθέτηση των στοιχείων υλοποιούνται είτε με συνδυαστικές, είτε με μαθηματικές ή αναλυτικές τεχνικές. Οι συνδυαστικές, τοποθετούν ένα στοιχείο τη φορά, προσπαθώντας να ελαχιστοποιήσουν τις συναρτήσεις κόστους για τα στοιχεία που έχουν ήδη τοποθετηθεί. Στης τεχνικές αυτές, τα στοιχεία τοποθετούνται σε θέσεις ευθυγραμμισμένες στις γραμμές και δεν εμφανίζουν επικάλυψη μεταξύ τους.

Οι αναλυτικές μέθοδοι, προσπαθούν να βρουν μια θέση για κάθε στοιχείο του κυκλώματος, λύνοντας συναρτήσεις τοποθέτησης με κατάλληλους περιορισμούς. Τέτοιες τεχνικές είναι για παράδειγμα η αριθμητική ανάλυση και ο γραμμικός προγραμματισμός. Αυτές οι μέθοδοι απαιτούν συγκεκριμένες υποθέσεις για την επίλυση των συστημάτων, όπως η υπόθεση ότι τα στοιχεία αντιπροσωπεύονται ως σημεία, χωρίς διαστάσεις. Ωστόσο, όταν χρησιμοποιούνται τέτοιοι αλγόριθμοι, τα στοιχεία του κυκλώματος μπορεί να τοποθετηθούν αρκούντως κοντά δημιουργώντας επικαλύψεις μεταξύ των στοιχείων ή και αποδίδοντας μη ευθυγραμμισμένες συντεταγμένες στις γραμμές και τις στήλες που έχουν οριστεί για το κύκλωμα κατά τη σχεδίαση [7]. Για τη διόρθωση

αυτού του προβλήματος, πρέπει να υπάρξει ένα ακόμα επίπεδο στον αλγόριθμο τοποθέτησης. Το επίπεδο αυτό ονομάζεται λεπτομερής τοποθέτηση.



Σχήμα 1.9: Επικάλυψη στοιχείων.



Σχήμα 1.10: Ευθυγράμμιση στοιχείου σε γραμμές και στήλες.

1.5.2 Λεπτομερής Τοποθέτηση

Η λεπτομερής τοποθέτηση είναι το σημείο στο οποίο τα στοιχεία θα λάβουν τις νόμιμες θέσεις τους (η διαδικασία αυτή ονομάζεται legalization). Οι νόμιμες θέσεις πρέπει να ικανοποιούνται όλες τις προδιαγραφές, βελτιώνοντας παράλληλα την ποιότητα της τοποθέτησης, η οποία πραγματοποιήθηκε κατά τη γενική τοποθέτηση. Επομένως, η λεπτομερής τοποθέτηση πραγματοποιείται μετά από τη γενική, στην οποία έχουν βρεθεί οι προσεγγιστικές θέσεις των στοιχείων.

Υπάρχουν δύο προσεγγίσεις για την υλοποίηση της λεπτομερής τοποθέτησης. Στην πρώτη προσέγγιση, τα στοιχεία νομιμοποιούνται και ταυτόχρονα βελτιώνεται η τοποθέτηση, πραγματοποιώντας ορισμένες αλλαγές για κάθε στοιχείο που τοποθετείται (πχ. περιστροφή). Στη δεύτερη προσέγγιση, η διαδικασία χωρίζεται σε δύο διακριτά τμήματα, τη νομιμοποίηση των στοιχείων και τη βελτίωση έγκυρης τοποθέτησης.

Κατά συνέπεια, αναπτύχθηκαν μεθοδολογίες και για τις δύο προσεγγίσεις. Για την πρώτη προσέγγιση, η λεπτομερής τοποθέτηση πραγματοποιείται με τους λεπτομερείς τοποθετητές και για τη δεύτερη οι νομιμοποίητες με ένα επίπεδο βελτιστοποίησης. Μετά από την εκτέλεση ενός νομιμοποιητή πραγματοποιούνται οι τυχόν βελτιστοποιήσεις της νόμιμης πλέον τοποθέτησης. Τόσο για την πρώτη προσέγγιση, όσο και για τη δεύτερη, χρησιμοποιήθηκαν διαφορετικές τεχνικές για την επίλυση του προβλήματος και παρουσιάζονται στο Πίνακα 2.1.

Για παράδειγμα, ο Domino [4] είναι ένας από τους περισσότερο χρησιμοποιημένους λεπτομερείς τοποθετητές. Ο τοποθετητής αυτός, λαμβάνει μια τοποθέτηση με επικαλύψεις και επαναληπτικά βελτιώνει την τοποθέτηση επιλύοντας τοπικά, σε μικρές περιοχές, το πρόβλημα

Stage	Technique
Legalization	greedy moves to free locations
	ripple cell movement
	diffusion partial differential equation
	dynamic programming
	computational geometry
	network flow
	linear programming
	top-down opt. & clustering
Detailed Placement	branch-and-bound
	network flow
	simulated annealing
	mixed ILP
	single-row optimization
	cell-to-slot matching
	cell swapping
	clustering
	dynamic programming
	global-placer integration

Πίνακας 1.1: Τεχνικές που χρησιμοποιούνται για την νομιμοποίηση στοιχείων [8].

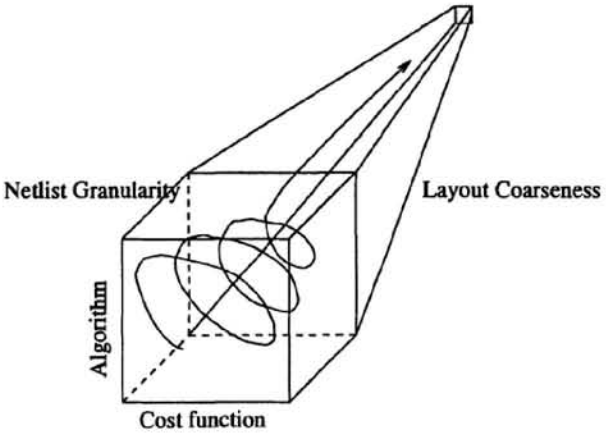
της επικάλυψης. Παράλληλα λαμβάνει τις αποφάσεις τοποθέτησης λαμβάνοντας υπόψιν το μήκος καλωδίων. Οι Sun και Sechen [12], πρότειναν μία τεχνική η οποία λαμβάνει υπόψιν το εκτιμώμενο μήκος καλωδίων. Η συγκεκριμένη τεχνική χρησιμοποιείται και στο TimberWolf. Ο Caldwell [3] χρησιμοποίησε τμηματοποίηση και βασίστηκε στην τεχνική branch-and-bound.

Από την άλλη, για τη δεύτερη προσέγγιση, ο αλγόριθμος που χρησιμοποιείται ευρέως είναι ο Tetris [5]. Ο συγκεκριμένος αλγόριθμος παρουσιάζεται στο Κεφάλαιο 2. Αποτελεί το θεμελιώδη νομιμοποιητή, διότι οι μεταγενέστερες προσεγγίσεις βασίζονται σε αυτόν. Ο Tetris είναι απλός και γρήγορος νομιμοποιητής. Παραλλαγές του αλγορίθμου αυτού, όπως ο Abacus χρησιμοποιούνται κατά κόρον στη βιομηχανία.

1.5.3 Παράμετροι Τοποθέτησης

Το πρόβλημα της τοποθέτησης στοιχείων σε ένα chip έχει μελετηθεί περισσότερο από τρείς δεκαετίες. Ωστόσο, υπάρχουν θεμελιώδη ερωτήματα που παραμένουν αναπάντητα. Το Σχήμα 1.11 παρουσιάζει το πλαίσιο του προβλήματος τοποθέτησης.

Ο κύβος πολυπλοκότητας έχει τέσσερις διαστάσεις που αντιπροσωπεύουν τα εξής στοιχεία: συνάρτηση κόστους (cost function), απόδοση αλγορίθμου, αφαιρετικότητα netlist (Netlist Granularity) και αφαιρετικότητα σχεδίασης (Layout Coarseness) [7]. Ένας αποτελεσματικός τοποθετητής θα πρέπει να λαμβάνει υπόψιν όλα τα παραπάνω στοιχεία για να επιτύχει το καλύ-



Σχήμα 1.11: Κύβος πολυπλοκότητας τοποθέτησης [7].

τερο δυνατό αποτέλεσμα.

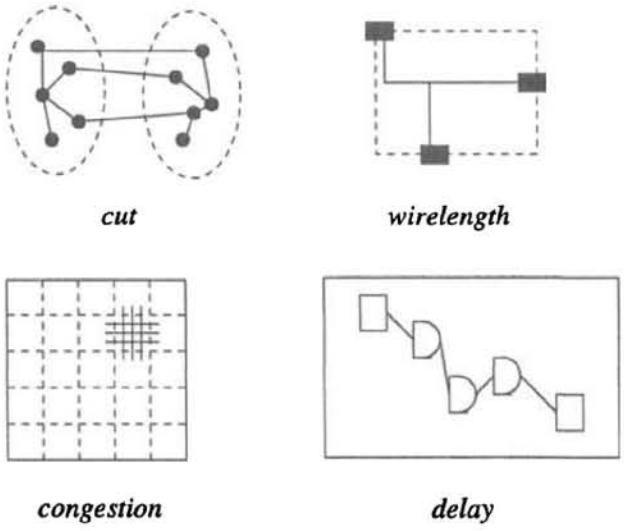
Cost Function (Σχήμα 1.12): Περιλαμβάνει το πουοστό επίτευξης των στόχων του placement. Μία χαρακτηριστική συνάρτηση κόστους που επιδιώκεται να ελαχιστοποιηθεί είναι το συνολικό μήκος καλωδίων του κυκλώματος. Η ελαχιστοποίηση αυτή, επιφέρει λιγότερη συμφόρηση στα καλώδια ή και επίτευξη των χρονικών περιορισμών του κυκλώματος. Ωστόσο, ο υπολογισμός του ελάχιστου μήκους καλωδίων είναι κοστοβόρος και επιβαρύνει υπολογιστικά το σύστημα. Το πρόβλημα αυτό μπορεί να προσπεραστεί καθώς στο placement δεν ενδιαφερόμαστε να υπολογίσουμε την ακριβή τιμή της συνάρτησης ελαχιστοποίησης και επαρκεί ο υπολογισμός μίας προσέγγισης [7].

Μία ακόμα συνάρτηση κόστους είναι η συμφόρηση των καλωδίων. Έτσι, επιδιώκεται να ελαχιστοποιηθεί το πλήθος των καλωδίων που διέρχονται από τις γραμμές του κυκλώματος. Η ελαχιστοποίηση της καθυστέρησης αποτελεί μια επιπλέον συνάρτηση κόστους η οποία διαφέρει από το είδος του κάθε κυκλώματος. Η συνάρτηση επιδιώκει την ελαχιστοποίηση της καθυστέρηση μεταξύ των διαφόρων μονοπατιών του κυκλώματος.

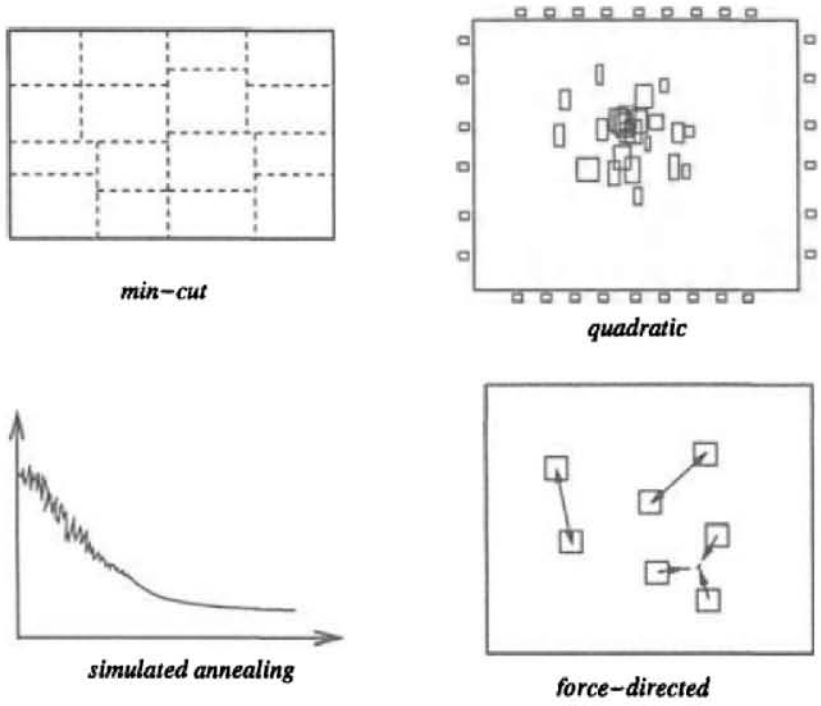
Η ελαχιστοποίηση μίας συνάρτησης από τις παραπάνω δε θα επιφέρει την επίτευξη όλων των στόχων που έχουν τεθεί για το placement. Έτσι, οι σημερινοί placers συνδυάζουν διαφορετικές συναρτήσεις κόστους για την επίτευξη καλύτερων αποτελεσμάτων.

Algorithms (Σχήμα 1.13): Για την επίλυση του προβλήματος τοποθέτησης, έχουν προταθεί αρκετοί αλγόριθμοι (π.χ. αναλυτικοί, ευριστικοί, κτλ.). Καθένας από τους οποίους έχει και θετικά και αρνητικά. Επομένως, η επιλογή του κατάλληλου αλγορίθμου εξαρτάται κάθε φορά από τους στόχους που έχουν τεθεί καθώς και από τους περιορισμούς του κάθε προβλήματος.

Netlist Granularity (Σχήμα 1.14): Το netlist granularity παρουσιάζει το επίπεδο αφαιρετικότητας των διασυνδέσεων του κυκλώματος. Για μεγάλα κυκλώματα, είναι αρκετά ασύμφορο να εφαρμόσει κανείς τους αλγορίθμους σε ολόκληρο το netlist. Ορισμένες μορφές αφαίρεσης, για παράδειγμα η κατάτμηση ή ομαδοποίηση (partitioning or clustering), είναι απαραίτητες για να επιλυθεί αποτελεσματικά το πρόβλημα τοποθέτησης.

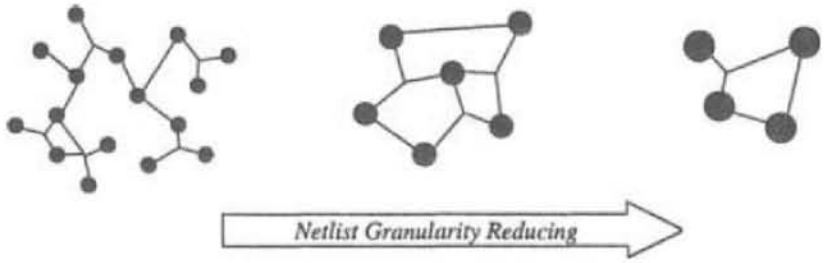


Σχήμα 1.12: Cost Function [7].



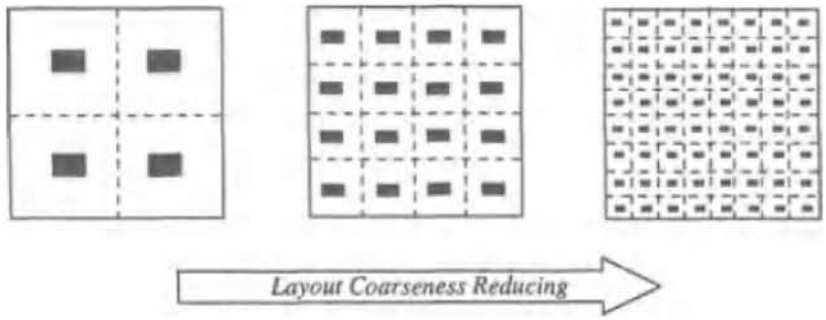
Σχήμα 1.13: Algorithms [7].

Layout Coarseness (Σχήμα 1.15): Το layout coarseness σχετίζεται με την από πάνω προς τα κάτω (top-down) ροή τοποθέτησης η οποία είναι η πιο διαδεδομένη για τη σχεδίαση VLSI



Σχήμα 1.14: Netlist Granularity [7].

κυκλωμάτων. Στη μεθοδολογία αυτή, το κύκλωμα διαιρείται σε υπό-κυκλώματα και καθώς η ροή μεγαλώνει το μέγεθός τους μειώνεται. Έτσι, το layout coarseness αναδεικνύει το επίπεδο της top-down ροής ή το επίπεδο της τοποθέτησης. Αξίζει να σημειωθεί ότι η αφαιρετικότητα σχεδίασης δεν είναι το ίδιο με την αυτή του netlist. Το πλήθος των περιοχών κατά την τοποθέτηση των κόμβων δεν είναι απαραίτητα ίδιο.



Σχήμα 1.15: Layout Coarseness [7].

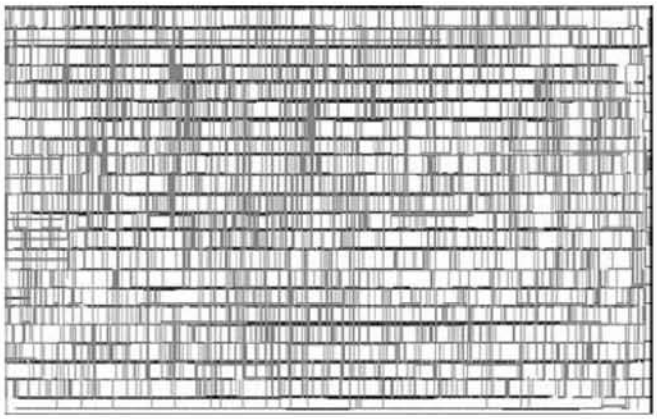
Στη διαδικασία τοποθέτησης, δεν υπάρχει μόνο μία συνάρτηση ή ένας αλγόριθμος που να εγγυάται την επιτυχία. Οπότε, είναι σημαντικό να επιλεγεί ο κατάλληλος αλγόριθμος που να ελαχιστοποιεί τις αντίστοιχες συναρτήσεις κόστους. Αυτό προϋποθέτει την πλήρη κατανόηση των διαφορετικών πτυχών του προβλήματος τοποθέτησης.

1.6 Διασύνδεση

Μετά τη διαδικασία τοποθέτησης, ακολουθεί η διασύνδεση των μονοπατιών του κυκλώματος (routing). Στο στάδιο αυτό, γίνεται η διασύνδεση των στοιχείων που συνδέονται για την ανταλλαγή ηλεκτρικών σημάτων από τα pin του ενός προς τα pin των υπολοίπων στοιχείων που συνδέονται.

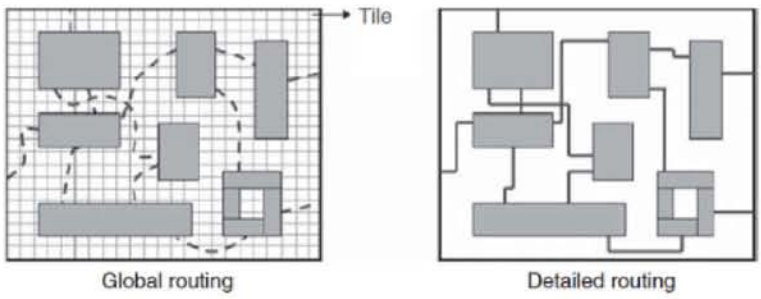
Το Σχήμα 1.16 παρουσιάζει ένα παράδειγμα της διασύνδεσης. Μετά τη διασύνδεση, πρέπει

να πραγματοποιηθεί μία σειρά από ελέγχους (rule checking, performance checking και reliability checking) έτοι ώστε να διαοφαλιστεί ότι το κύκλωμα ουνεχίζει να πληροί τις προδιαγραφές.



Σχήμα 1.16: Παράδειγμα διαούνδεοης.

Ουοιαστικά, η διαούνδεοη αποτελεί μία αρκετά πολύπλοκη διαδικαοία και ουνήθως χωρίζεται οε δύο οτάδια [14]. Τη γενική και τη λεπτομερή διαούνδεοη. Το πρώτο, κάνει μία μορφή κατάτμηοης του chip οε μικρά κομμάτια και αποφαοίζει τις διαδρομές ανάμεοα οε αυτές τις περιοχές, ενώ ταυτόχρονα προοπαθεί να βελτιοτοποιήοει ορισμένα οτοιχεία του κυκλώματος, όπως το μήκος των καλωδίων. Στη ουνέχεια, η λεπτομερής διαούνδεοη καθορίζει τις πραγματικές και νόμιμες διαουνδέοεις και τα μονοπάτια για όλα τα δίκτυα του κυκλώματος βάοη του αποτελέοματος της γενικής διαούνδεοης (Σχήμα 1.17).



Σχήμα 1.17: Γενική και λεπτομερής διαούνδεοη [14].

1.7 Αντικείμενο και Στόχοι της Διπλωματικής Διατριβής

Στόχος αυτής της διπλωματικής διατριβής ήταν η υλοποίηση ενός νομιμοποιητή για τη νομιμοποίηση των στοιχείων ενός κυκλώματος. Αρχικά, πραγματοποιήθηκε έρευνα σχετικά με τους νομιμοποιητές που υπάρχουν στη βιβλιογραφία και έγινε η επιλογή του Abacus. Ο συγκεκριμένος νομιμοποιητής επιλέχθηκε διότι επιτυγχάνει τη μικρότερη δυνατή τροποποίηση της λύσης, σε αντίθεση με τους υπόλοιπους νομιμοποιητές (πχ. Tetris [5]). Επιπρόσθετα, ο Abacus, δεν είναι μόνο ένας εύκολα υλοποιήσιμος νομιμοποιητής, αλλά παρέχει και εύκολη κατανόηση του προβλήματος νομιμοποίησης. Για τους παραπάνω λόγους αποτελεί μια πολύ καλή επιλογή νομιμοποιητή για την ενσωμάτωση του σε έναν λεπτομερή τοποθετητή, η οποία αποτελεί και έναν από τους μελλοντικούς στόχους της παρούσας εργασίας.

Οι επιμέρους στόχοι της διπλωματικής εργασίας ήταν :

- Βελτίωση βασικού αλγορίθμου
- Αλλαγή συνάρτησης υπολογισμού κόστους μετακίνησης
- Φραγμός/Όριο θέσεων που εξερευνώνται
- Διάταξη επιλογής στοιχείων
- Νομιμοποίηση σε ιεραρχικά κυκλώματα
- Νομιμοποίηση κυκλωμάτων με εμπόδια
- Ενσωμάτωση σε βιομηχανικό εργαλείο

Για να τη βελτίωση του θεμελιώδους αλγορίθμου εισαγάγαμε επιπλέον ελέγχους για την τοποθέτηση των στοιχείων. Οι έλεγχοι αφορούν την χωρητικότητα της κάθε γραμμής. Έτσι, αν ένα στοιχείο δε χωράει να τοποθετηθεί σε μία γραμμή, τότε η τοποθέτηση στη συγκεκριμένη γραμμή δεν πραγματοποιείται.

Επιπρόσθετα, δοκιμάσαμε να αλλάξουμε την συνάρτηση υπολογισμού της μετακίνησης. Ο κλασικός αλγόριθμος υπολογίζει το κόστος μετακίνησης των στοιχείων με την αθροιστική συνάρτηση μετακίνησης (Total Displacement Function - F_{TD}). Η F_{TD} αποτελεί μία αντικειμενική συνάρτηση υπολογισμού κόστους διότι η μεγάλη αύξηση ή μείωση μίας παραμέτρου επιφέρει συνολικά μεγάλη αύξηση ή μείωση αντίστοιχα. Εμείς, προσαρτήσαμε και την συνάρτηση μέσης μετακίνησης (Mean Displacement Function - F_{MD}), η οποία υπολογίζει την μέση μετακίνηση των στοιχείων. Ο λόγος που επιλέξαμε να εξετάσουμε και την F_{MD} ήταν γιατί η συγκεκριμένη συνάρτηση είναι λιγότερο άπληστη. Έτσι, μπορεί να επιλέξει μια μετακίνηση η οποία προσωρινά να μην είναι η μικρότερη δυνατή, αλλά μακροπρόθεσμα να επιφέρει μικρότερη συνολική μετακίνηση.

Σε ό,τι αφορά την εξερεύνηση των γραμμών, η κλασική υλοποίηση πραγματοποιεί εξαντλητική αναζήτηση (Exhaustive Search - ES) των γραμμών και έπειτα επιλέγει τη γραμμή με τη

μικρότερη μετακίνηση των στοιχείων. Εν τούτοις, η εξερεύνηση των γραμμών μπορεί να περιοριστεί εισάγοντας ορισμένους φραγμούς, χωρίς ωστόσο να επηρεαστεί η μετακίνηση των στοιχείων. Γι' αυτό το λόγο εισαγάγαμε εμβέλειες για την εξερεύνηση των γραμμών (Range - R), χρησιμοποιώντας το κόστος της μετακίνησης των στοιχείων της γραμμής και επιτυγχάνοντας την βέλτιστη εξερεύνηση γραμμών (Bounded Search - BS).

Ακόμα, ο θεμελιώδης αλγόριθμος επιλέγει τα στοιχεία προς νομιμοποίηση βάσει της συντεταγμένης x που έχουν κατά τη γενική τοποθέτηση. Χαρακτηρίστηκα, αναφέρεται ότι η σειρά με την οποία επιλέγονται τα στοιχεία έχει αντίκτυπο στη συνολική μετακίνηση των στοιχείων. Έτσι, πρέπει να δοκιμαστούν διαφορετικές διατάξεις και να επιλέγεται εκείνη με την μικρότερη μετακίνηση. Γι' αυτό το λόγο επιλέξαμε να εξετάσουμε τρείς διαφορετικές μεθόδους επιλογής, την αύξουσα (Increasing Order - IO) και την φθίνουσα διάταξη (Decreasing Order - DO), τι οποίες αναφέρει ο κλασικός αλγόριθμος, καθώς και την από το κέντρο προς το κοντινότερο (Middle to Nearest Order - MNO). Στην συγκεκριμένη προσέγγιση επιλέγουμε αρχικά το μεσαίο στοιχείο της διάταξης και έπειτα επιλέγουμε το στοιχείο που βρίσκεται πιο κοντά του μέχρι να τελειώσουν τα στοιχεία.

Έπειτα, τροποποιήσαμε την υλοποίηση του αλγορίθμου για την υποστήριξη ιεραρχικών κυκλωμάτων, μιας και ο Abacus δεν λαμβάνει υπόψιν την ιεραρχία. Η ιεραρχία των κυκλωμάτων που υποστηρίζουμε προέρχεται από την ιεραρχία της Verilog. Έτσι μετά την διαδικασία της χωροθέτησης πραγματοποιείται, ξεχωριστά σε κάθε ένα επίπεδο της ιεραρχίας, η τοποθέτηση και η νομιμοποίηση των στοιχείων.

Η υποστήριξη κυκλωμάτων με εμπόδια αποτέλεσε έναν επιπλέον στόχο της συγκεκριμένης διατριβής. Η κλασική υλοποίηση του αλγορίθμου για να αντιμετωπίσει το πρόβλημα της νομιμοποίησης κυκλωμάτων με εμπόδια, χωρίζει τις γραμμές σε υπό-γραμμές και πραγματοποιεί νομιμοποίηση σε αυτές. Ωστόσο, η προσέγγιση αυτή χειρίζεται όλα τα στοιχεία σα να έχουν την ίδια σημασία για το κύκλωμα. Έτσι, δεν έχει καλά αποτελέσματα για κυκλώματα που μας ενδιαφέρουν περισσότερο ορισμένα μονοπάτια και συνδέσεις απ' ότι από άλλες. Γι' αυτό το λόγο προτείνουμε και μια επιπλέον προσέγγιση, η οποία διατηρεί την αρχική διάταξη των στοιχείων και πραγματοποιεί μετακίνηση των στοιχείων που βρίσκονται στις υπό-γραμμές. Η υλοποίηση της μεθόδου αυτής είναι υπό ανάπτυξη και γι' αυτό δε διαθέτουμε πειραματικά αποτελέσματα για να συγκρίνουμε τις δύο μεθόδους.

Τέλος, προσαρμόσαμε το νομιμοποιητή σε ένα υπανάπτυξη βιομηχανικό εργαλείο EDA [13] και ως συνέπεια, ο νομιμοποιητής πληροί τις βιομηχανικές προϋποθέσεις και περιορισμούς.

ΚΕΦΑΛΑΙΟ 2

Επισκόπηση Νομιμοποίησης Στοιχείων (Legalization)

2.1 Εισαγωγή

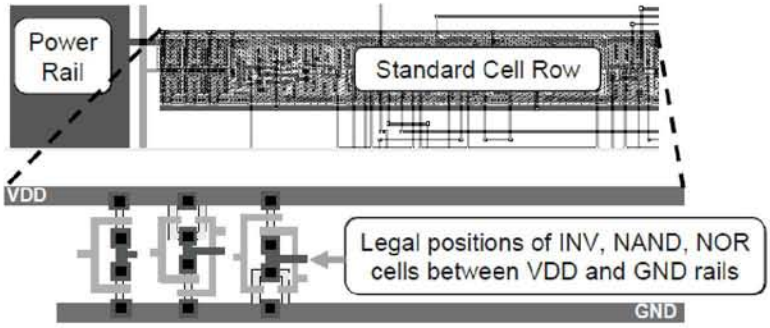
Η γενική τοποθέτηση αναθέτει συντεταγμένες στα standard cells και στα modules (π.χ. macroblocks) του κυκλώματος. Η εύρεση των κατάλληλων συντεταγμένων πραγματοποιείται κατά κύριο λόγο με χρήση αναλυτικών μεθόδων [1]. Ωστόσο, οι συντεταγμένες που ανατίθενται κατά τη γενική τοποθέτηση δεν είναι ευθυγραμμισμένες στις γραμμές τροφοδοσίας (power rails ή rows) και έχουν πραγματικές τιμές και μη κβαντισμένες στις στήλες (collums) του κυκλώματος. Επιπλέον, δημιουργούνται περιοχές οι οποίες είναι εξαιρετικά πυκνές προκαλώντας για παράδειγμα αύξηση της θερμοκρασίας.

Ο λόγος εμφάνισης του συγκεκριμένου προβλήματος είναι γιατί στις αναλυτικές (μαθηματικές) μεθόδους δε μπορούν να μοντελοποιηθούν οι γραμμές τροφοδοσίας, οι συντεταγμένες και τα φυσικά μεγέθη των στοιχείων ως μαθηματικοί περιορισμοί.

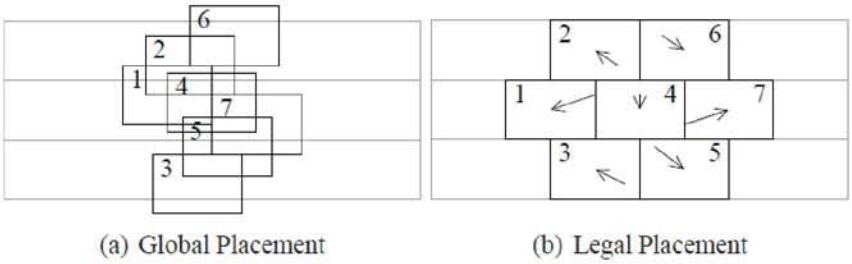
Για την αντιμετώπιση των παραπάνω φαινομένων, η γενική τοποθέτηση πρέπει να νομιμοποιηθεί. Σε ό,τι αφορά τις επιτρεπτές θέσεις τοποθέτησης των στοιχείων κατά τον κάθετο και τον οριζόντιο άξονα, υπολογίζονται ως ο κβαντισμός της εκάστοτε τιμής σε πολλαπλάσια του ύψους των γραμμών και του πλάτους των στηλών του κυκλώματος. Οι δύο αυτοί παράμετροι ορίζονται στο στάδιο των περιορισμών συστήματος (Ενότητα 1.3) και εντοπίζονται συγκεκριμένα στα αρχεία βιβλιοθηκών (Σχήμα 2.1).

Η διαδικασία νομιμοποίησης επιδιώκει τον εντοπισμό μίας νόμιμης θέσης για κάθε στοιχείο του κυκλώματος, ελαχιστοποιώντας την επίδραση παραμέτρων, όπως για παράδειγμα το μήκος καλωδίων (Σχήμα 2.2). Για πυκνά κυκλώματα, η νομιμοποίηση μπορεί να είναι μία αρκετά δύσκολη και χρονοβόρα διαδικασία. Έτσι, αναπτύχθηκαν διαφορετικές μέθοδοι για την επίτευξη ταχύτερων αποτελεσμάτων.

Επιπλέον, η νομιμοποίηση είναι απαραίτητη όχι μόνο μετά από τη γενική τοποθέτηση, αλλά και μετά από κάθε τροποποίηση του κυκλώματος, όπως η αλλαγή του μεγέθους κάποιων στοι-



Σχήμα 2.1: Ευθυγράμμιση στοιχείων σε γραμμές [7].



Σχήμα 2.2: Παράδειγμα νομιμοποίησης μίας γενικής τοποθέτησης [11].

χείων ή η εισαγωγή των buffers κατά την φυσική σύνθεση του κυκλώματος (physical synthesis).

Για κυκλώματα που περιέχουν και standard cells και macroblocks, υπάρχουν δύο προσεγγίσεις. Η πρώτη τοποθετεί αρχικά τα macroblocks στις σταθερές νόμιμες θέσεις τους και μετά τοποθετεί τα standard cells στον υπολειπόμενο ελεύθερο χώρο. Η δεύτερη προσέγγιση τοποθετεί και τα standard cell και τα macroblock ταυτόχρονα σε νόμιμες θέσεις.

Υπάρχουν αρκετοί δημοσιευμένοι αλγόριθμοι για τη νομιμοποίηση στοιχείων. Για παράδειγμα, ο Domino [4], ο οποίος χωρίζει τις γραμμές σε περιοχές και τμηματοποιεί τα στοιχεία σε υπό-στοιχεία τα οποία έχουν το ίδιο ύψος και πλάτος, τοποθετώντας τα με μια μέθοδο ελαχιστοποίησης κόστους. Ο Mongrel [6], χρησιμοποιεί ένα άπληστο ευριστικό αλγόριθμο για τη μετακίνηση των στοιχείων, με μορφή κύματος, από πυκνές περιοχές σε λιγότερο πυκνές. Ωστόσο, ο αλγόριθμος που χρησιμοποιείται ευρέως στη βιομηχανία, αυτούσιος ή με παραλλαγές, είναι ο αλγόριθμος Tetris [5] ο οποίος αναλύεται στην Ενότητα 2.2. Ο Tetris χρησιμοποιεί έναν γρήγορο και άπληστο ευριστικό αλγόριθμο, για την εύρεση των νόμιμων συντεταγμένων με τη μικρότερη δυνατή μετακίνηση [2].

Το βασικό πρόβλημα των γρήγορων και απλών στην υλοποίηση νομιμοποιητών, είναι ότι ορισμένα στοιχεία μπορεί να μετακινηθούν σε μεγάλη απόσταση από τη βέλτιστη θέση που υπολογίστηκε από τον αναλυτικό αλγόριθμο κατά τη γενική τοποθέτηση. Η επίδραση αυτής της μετακίνησης είναι καίριας σημασίας μίας και η αύξηση του μήκους των καλωδίων συνεπάγεται περισσότερη καθυστέρηση στο κύκλωμα.

Μόλις όλα τα στοιχεία νομιμοποιηθούν, εξαλείφοντας κάθε επικάλυψη και ευθυγραμμίζοντας

τα στις γραμμές και τις στήλες, πραγματοποιείται η λεπτομερής τοποθέτηση. Στο σημείο αυτό θα πραγματοποιηθούν βελτιστοποιήσεις οι οποίες έχουν δραματική επίδραση στο μήκος καλωδίων.

2.2 Tetris Legalizer

Ο αλγόριθμος Tetris είναι εξαιρετικά απλός στην κατανόηση και στην υλοποίησή του και παρέχει καλή διαίσθηση για το πρόβλημα νομιμοποίησης. Ωστόσο, η μέθοδος αυτή μπορεί για ορισμένα κυκλώματα να επιτύχει ικανοποιητικά αποτελέσματα και για κάποια να αποτύχει δραματικά. Για παράδειγμα ένα μειονέκτημα είναι ότι τα στοιχεία μπορεί να τοποθετηθούν στη μία πλευρά του κυκλώματος αυξάνοντας το συνολικό μήκος καλωδίων και την καθυστέρηση, δεδομένου ότι υπάρχουν σταθερά pins στην άλλη μεριά του κυκλώματος [7].

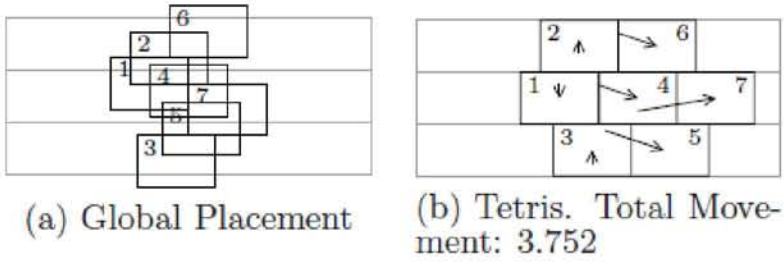
Ο αλγόριθμος στο Σχήμα 2.3 περιγράφει τη μέθοδο Tetris. Αρχικά, τα στοιχεία διατάσσονται σύμφωνα με τη συντεταγμένη x που έλαβαν κατά τη γενική τοποθέτηση (γραμμή 1). Ακολουθεί η νομιμοποίηση όλων των στοιχείων, ανά ένα τη φορά (γραμμή 2-13). Η νομιμοποίηση ενός στοιχείου i πραγματοποιείται με τη μετακίνηση του στοιχείου σε κάθε γραμμή και σε κάθε νόμιμη θέση x (γραμμή 4 και 5). Αν για το στοιχείο βρεθεί μια νόμιμη θέση υπολογίζεται το κόστος αυτής της θέσης (γραμμή 7). Η συνάρτηση κόστους αφορά συνήθως τη συνολική μετακίνηση από τη θέση που είχε δοθεί κατά τη γενική τοποθέτηση και την τελικής. Μετά την εξερεύνηση όλων των γραμμών, επιλέγονται οι συντεταγμένες (γραμμή 12) με το μικρότερο δυνατό κόστος (γραμμή 8).

```
1  Sort modules according to x-position;
2  foreach module i do
3  |  c_best ← ∞;
4  |  foreach x do
5  |  |  foreach y do
6  |  |  |  if module i fits at (x, y) then
7  |  |  |  |  Determine cost c;
8  |  |  |  |  if c < c_best then c_best = c, x_best = x, y_best = y;
9  |  |  |  end
10 |  |  end
11 |  end
12 |  Place module i to (x_best, y_best);
13 end
```

Σχήμα 2.3: Ψευδοκώδικας Tetris [10].

Το βασικό μειονέκτημα του Tetris είναι ότι όταν βρεθεί μια νόμιμη θέση και τοποθετηθεί το στοιχείο σε αυτή, το στοιχείο δεν μπορεί να ξανά μετακινηθεί. Το πρόβλημα παρουσιάζεται στο Σχήμα 2.4(b) όπου τα στοιχεία 1,2 και 3, τα οποία τοποθετούνται πρώτα, θα έχουν μικρές

μετακινήσεις ενώ τα υπόλοιπα θα έχουν μεγάλες καθιστώντας τη συνολική μετακίνηση αρκετά μεγάλη.



(a) Global Placement (b) Tetris. Total Movement: 3.752

Σχήμα 2.4: Νομιμοποίηση Tetris και κόστος μετακίνησης [11].

2.3 Abacus Legalizer

Ο Abacus αποτελεί μία βελτιστοποίηση του Tetris και είναι και αυτός ένας γρήγορος αλγόριθμος νομιμοποίησης κυκλωμάτων με standard cell. Στόχος του είναι η ελάχιστη μετακίνηση των στοιχείων. Όπως και ο Tetris, έτσι και ο Abacus, βαοίζεται στη διάταξη των στοιχείων βάοει της θέσης που έχουν μετά τη γενική τοποθέτηση και νομιμοποιεί ένα στοιχείο τη φορά. Η νομιμοποίηση ενός στοιχείου πραγματοποιείται με τη διάτρεξη όλων των γραμμών της οχεδίαοης και επιλέγοντας τη γραμμή στην οποία η μετακίνηση είναι η ελάχιστη δυνατή.

Στο οημείο αυτό έγκειται και η διαφοροποίηση του αλγορίθμου οε οχέοη με αυτόν του Tetris. Τα στοιχεία που προϋπάρχουν στη γραμμή, πλέον μπορούν να μετακινούνται έτοι ώστε να επιτευχθεί η ελάχιστη συνολική μετακίνηση των στοιχείων που έχουν τοποθετηθεί μέχρι αυτό το οημείο. Ο υπολογιομός των καινούριων ουντεταγμένων πραγματοποιείται με χρήοη δυναμικού προγραμματιομού. Όοο για τις ουντεταγμένες αυτές, υπολογίζεται μόνο η οριζόντια θέοη και όχι η κάθετη, μιας και τα ήδη τοποθετημένα στοιχεία δεν μπορούν να αλλάξουν γραμμή. Ως ουνέπεια, ο Abacus πετυχαίνει οημαντικά μικρότερη μετακίνηση των στοιχείων, αλλά υστερεί οε χρόνο εκτέλεοης. Αυτό ουμβαίνει γιατί πρέπει να πραγματοποιήοει τους υπολογιομούς των νέων ουντεταγμένων όλων των ουντεταγμένων της γραμμής και όχι μόνο του τελευταίου που τοποθέτηοε.

2.3.1 Αλγόριθμος Abacus

Ο αλγόριθμος ξεκινά με την υπόθεοη ότι τα χαρακτηριστικά όπως το μήκος των καλωδίων, το μήκος του κρίοιμου μονοπατιού κτλ. έχουν βελτιστοποιηθεί κατά τη γενική τοποθέτηοη. Έτοι, στόχος του αλγορίθμου είναι να μετακινήοει όοο το λιγότερο δυνατόν τα στοιχεία ώστε να είναι οε νόμιμες θέοεις και να προοεγγίζουν τις βέλτιστες θέοεις. Δηλαδή, να μην έχουν επικαλύψεις και να είναι ευθυγραμμιομένα οτις γραμμές και τις οτήλες. Επιπροοθέτως, θεωρείται ότι οι

γραμμές έχουν οριζόντια κατεύθυνση και τα στοιχεία έχουν το ίδιο ύψος. Αν το κύκλωμα περιέχει macroblocks, τότε αυτά πρέπει να έχουν τοποθετηθεί εκ των προτέρων σε νόμιμες θέσεις. Αλλιώς οι γραμμές υποδιαιρούνται σε νέες "υπο-γραμμές" έτσι ώστε να μην περιέχουν macroblocks [11].

Το Σχήμα 2.5 παρουσιάζει τον ψευδοκώδικα υλοποίησης του Abacus Legalizer.

```
1  Sort cells according to x-position;
2  foreach cell i do
3  |   c_best ← ∞;
4  |   foreach row r do
5  |   |   Insert cell i into row r;
6  |   |   PlaceRow r (trial);
7  |   |   Determine cost c;
8  |   |   if c < c_best then c_best = c, r_best = r;
9  |   |   Remove cell i from row r;
10 |   end
11 |   Insert Cell i to row r_best;
12 |   PlaceRow r_best (final);
13 end
```

Σχήμα 2.5: Αλγόριθμος Abacus Legalizer [11].

Όμοια με τον αλγόριθμο Tetris, τα στοιχεία αρχικά διατάσσονται σύμφωνα με τη συντεταγμένη x (γραμμή 1). Έπειτα, τα στοιχεία νομιμοποιούνται ένα τη φορά (γραμμή 2-13). Ένα στοιχείο νομιμοποιείται όταν εξερευνώντας όλες τις γραμμές (γραμμή 4-10) εισάγοντας το στοιχείο (γραμμή 5) σε κάθε μία σύμφωνα με τη συντεταγμένη x. Στο σημείο αυτό παρουσιάζεται η διαφορά του παραπάνω αλγορίθμου σε σχέση με τον αλγόριθμο Tetris. Η συνάρτηση PlaceRow (γραμμή 6 και 12) μετακινεί τα απαραίτητα στοιχεία της γραμμής για την επίτευξη μικρότερης συνολικής μετακίνησης (γραμμή 7). Στο τέλος της επανάληψης, το στοιχείο απομακρύνεται από τη γραμμή (γραμμή 9) για να συνεχίσει την εξερεύνηση των υπολοίπων γραμμών. Μετά την εξερεύνηση όλων των γραμμών, το στοιχείο εισάγεται στη γραμμή (γραμμή 11 - 12) με τη βέλτιστη μετακίνηση (γραμμή 8).

2.3.2 Συνάρτηση PLACEROW

Η συνάρτηση PlaceRow υπολογίζει και βελτιστοποιεί τη συνολική μετακίνηση των στοιχείων μίας γραμμής. Η συνάρτηση αυτή, αποτελεί τη σημαντικότερη διαφορά ανάμεσα στους αλγορίθμου Tetris και Abacus.

Υποθέτουμε ότι η γραμμή περιέχει N_r στοιχεία, τα οποία είναι διατεταγμένα σύμφωνα με τη συντεταγμένη x. Το Σχήμα 2.6 παρουσιάζει τις ιδιότητες του στοιχείου i. Έτσι, δεδομένων των συντεταγμένων x' και y' από τη γενική τοποθέτηση, του πλάτους του στοιχείου και ενός συντελεστή βαρύτητας, υπολογίζονται οι καινούριες νόμιμες συντεταγμένες του στοιχείου x και y (Σχέση 2.2). Ο συντελεστής βαρύτητας μπορεί να προκύψει για παράδειγμα από το εμβαδόν ή

από το πλήθος των pin του στοιχείου. Συγκεκριμένα, η συντεταγμένη y υπολογίζεται δεδομένου ότι γνωρίζουμε σε ποια γραμμή θα τοποθετηθεί το στοιχείο.

Property	Explanation
$x'(i), y'(i)$	Position in global placement
$x(i), y(i)$	Position in legal placement
$w(i)$	Width
$e(i)$	Weight (e.g., number of pins)

Σχήμα 2.6: Ιδιότητες στοιχείων προς νομιμοποίηση [11].

Επιπλέον, αν θεωρηθεί ότι τα στοιχεία είναι διατεταγμένα έτσι ώστε $x'(i) \geq x'(i-1)$, ο υπολογισμός των νέων συντεταγμένων x θα προέλθει από την επίλυση του ακόλουθου τετραγωνικού συστήματος (quadratic program).

$$min \sum_{i=1}^{Nr} e(i)[x(i) - x'(i)]^2 \tag{2.1}$$

$$s.t. \quad x(i) - x(i-1) \geq w(i-1) \quad i = 2, ..., Nr \tag{2.2}$$

Η Σχέση 2.1 περιγράφει τη συνολική, βεβαρημένη και τετραγωνική μετακίνηση των στοιχείων μεταξύ του $x'(i)$ και του $x(i)$. Η Σχέση 2.2 εγγυάται ότι τα στοιχεία της γραμμής δεν παρουσιάζουν επικαλύψεις μεταξύ τους και διατηρεί τη διάταξη των στοιχείων σύμφωνα με τη διάταξη κατά τη γενική τοποθέτηση.

Ωστόσο, η επίλυση τετραγωνικών προγραμμάτων με περιορισμούς της μορφής "$\geq$" είναι αρκετά χρονοβόρες. Για το λόγο αυτό, στον αλγόριθμο πραγματοποιήθηκε μία εικασία. Το πρόβλημα μεταοχηματίστηκε σε γραμμική εξίσωση μετατρέποντας το "$\geq$" σε "$=$". Η μετατροπή αυτή προϋποθέτει την εικασία ότι τα στοιχεία της γραμμής εφάπτονται μεταξύ τους, δηλαδή δεν υπάρχει καθόλου ελεύθερος χώρος μεταξύ δύο στοιχείων. Έτσι, η Σχέση 2.2 μεταοχηματίζεται στη Σχέση 2.3:

$$x(i) = x(1) + \sum_{k=1}^{i-1} w(k) \quad i = 2, ..., Nr \tag{2.3}$$

και εισάγοντας τη Σχέση 2.3 στη Σχέση 2.1 προκύπτει η Σχέση 2.4, η οποία εξαρτάται μόνο από το $x(1)$ και για την ελαχιστοποίηση της, την εξισώνουμε με το μηδέν.

$$\sum_{i=1}^{Nr} e(i)x(1) - \left[e(1)x'(1) + \sum_{i=2}^{Nr} e(i) \left[x'(i) - \sum_{k=1}^{i-1} w(k) \right] \right] = 0 \tag{2.4}$$

Το Σχήμα 2.7 παρουσιάζει τον επαναληπτικό υπολογισμό των ιδιοτήτων του κάθε στοιχείου της γραμμής, οι οποίες είναι εξαρτημένες μόνο ως προς το $x'(i)$, το $w(i)$ και το $e(i)$.

Κατά συνέπεια, μετά τον επαναληπτικό υπολογισμό των παραπάνω ιδιοτήτων για τα Nr στοιχεία της γραμμής, υπολογίζεται η βέλτιοτη συντεταγμένη x για το πρώτο στοιχείο, δηλαδή

Init	Iteration $(i = 1, 2, ..., N_r)$
$e_c = 0$	$e_c \leftarrow e_c + e(i)$
$q_c = 0$	$q_c \leftarrow q_c + e(i)\,[x'(i) - w_c]$
$w_c = 0$	$w_c \leftarrow w_c + w(i)$

Σχήμα 2.7: Επαναληπτικός υπολογισμός ιδιοτήτων στοιχείων προς νομιμοποίηση [11].

το $x(1)$, σύμφωνα με τη Σχέση 2.5. Όπου e_c και w_c το συνολικό βάρος και πλάτος των στοιχείων της γραμμή και τέλος ο υπολογισμός του $x(i)$ γίνεται σύμφωνα με τη Σχέση 2.3.

$$x(1)e_c - q_c = 0 \Leftrightarrow x(1) = \frac{q_c}{e_c} \tag{2.5}$$

Ωστόοο, για να χρησιμοποιηθούν οι παραπάνω εξισώσεις, έγινε η υπόθεση ότι όλα τα στοιχεία της γραμμής εφάπτονται. Η εικαοία αυτή, μπορεί να απλοποιεί σημαντικά τη συνάρτηση του προβλήματος εύρεσης των συντεταγμένων x, αλλά δεν είναι ρεαλιστική. Σε μία γραμμή δεν είναι απαραίτητο τα στοιχεία να εφάπτονται. Γι' αυτό το λόγο, προστίθεται στον αλγόριθμο η έννοια της "ομάδας" (cluster ή group) από στοιχεία.

Πλέον, οι γραμμές περιέχουν groups από στοιχεία. Τα στοιχεία του κάθε group εφάπτονται μεταξύ τους αλλά όχι και τα groups. Έτοι, για τον υπολογισμό της τοποθέτησης ενός group χρησιμοποιείται η Σχέση 2.5. Μετά τον εντοπισμό της τοποθεσίας του group, δηλαδή του πρώτου στοιχείου του group, ο υπολογισμός της συντεταγμένης των υπολοίπων στοιχείων του group είναι τετριμμένη καθώς τα στοιχεία εφάπτονται (Σχέση 2.3). Το Σχήμα ;; παρουσιάζει τις αντίστοιχες ιδιότητες του Σχήματος 2.6 αλλά πλέον για groups στοιχείων και όχι για μεμονωμένα στοιχεία.

Property	Explanation
$n_{first}(c)$	First cell of cluster
$n_{last}(c)$	Last cell of cluster
$N_c(c)$	Number of cells in cluster
$x_c(c)$	Optimal position (lower left corner)
$e_c(c)$	Total weight
$w_c(c)$	Total width
$q_c(c)$	Optimal position

Πίνακας 2.1: Τεχνικές που χρησιμοποιούνται για την νομιμοποίηση στοιχείων.

Ακολουθεί η περιγραφή της υλοποίησης της συνάρτηση PlaceRow (Σχήμα 2.8) με χρήση δυναμικού προγραμματισμού.

Ο αλγόριθμος ξεκινάει με την επαναληπτική ομαδοποίηση (clustering ή grouping) των στοιχείων και υπολογίζει τη βέλτιστη θέση κάθε group (γραμμή 1-13). Στο συγκεκριμένο ψευδοκώ-

```
   // Determine clusters and their optimal positions x_c(c):
 1 for i = 1, ..., N_r do
 2 |   c ← Last cluster;
   |   // First cell or cell i does not overlap with last
   |      cluster:
 3 |   if i = 1 or x_c(c) + w_c(c) ≤ x'(i) then
 4 |   |   Create new cluster c;
 5 |   |   Init e_c(c), w_c(c), q_c(c) to zero;
 6 |   |   x_c(c) ← x'(i);
 7 |   |   n_first(c) ← i;
 8 |   |   AddCell(c, i);
 9 |   else
10 |   |   AddCell(c, i);
11 |   |   Collapse(c);
12 |   end
13 end
   // Transform cluster positions x_c(c) to cell positions
      x(i):
14 i ← 1;
15 for all clusters c do
16 |   x = x_c(c);
17 |   for i ≤ n_last(c) do
18 |   |   x(i) ← x;
19 |   |   x ← x + w(i);
20 |   end
21 end

22 Function AddCell(c, i):
23 n_last(c) ← i;
24 e_c(c) ← e_c(c) + e(i);
25 q_c(c) ← q_c(c) + e(i) · (x'(i) − w_c(c));
26 w_c(c) ← w_c(c) + w(i);

27 Function AddCluster(c, c'):
28 n_last(c) ← n_last(c');
29 e_c(c) ← e_c(c) + e_c(c');
30 q_c(c) ← q_c(c) + q_c(c') − e_c(c') · w_c(c);
31 w_c(c) ← w_c(c) + w_c(c');

32 Function Collapse(c):
   // Place cluster c:
33 x_c(c) ← q_c(c)/e_c(c);
   // Limit position between x_min and x_max − w_c(c)
34 if x_c(c) < x_min then x_c(c) = x_min;
35 if x_c(c) > x_max − w_c(c) then x_c(c) = x_max − w_c(c);
   // Overlap between c and its predecessor c'?:
36 c' ← Predecessor of c;
37 if c' exists and x_c(c') + w_c(c') > x_c(c) then
   |   // Merge cluster c to c':
38 |   AddCluster(c', c);
39 |   Remove cluster c;
40 |   Collapse(c');
41 end
```

Σχήμα 2.8: Κώδικας PlaceRow [11]. Τοποθετεί τα στοιχεία μίας γραμμής σε βέλτιστες θέσεις.

δικα, τα στοιχεία έχουν διαταχθεί με αύξουσα σειρά σύμφωνα με τις συντεταγμένες που έλαβαν κατά τη γενική τοποθέτηση (γραμμή 1), αλλά η φορά της διάταξης δεν επηρεάζει τη συνολική εικόνα του αλγορίθμου.

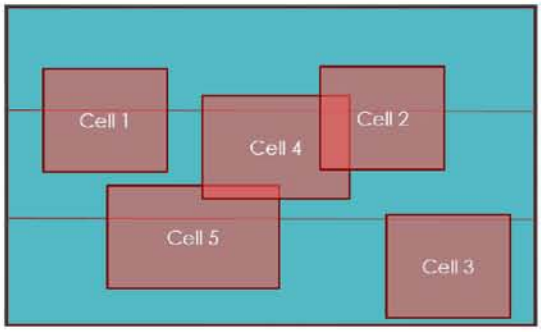
Αν το στοιχείο i είναι το πρώτο που τοποθετείται στη γραμμή, τότε δεν θα επικαλύπτεται με κάποιο άλλο (γραμμή 3) και έτσι δημιουργείται το πρώτο group της γραμμής (γραμμή 4-8). Όμοια, αν το στοιχείο δεν έχει επικάλυψη με κανένα άλλο από τη γραμμή (γραμμή 3), τότε δημιουργείται ένα καινούριο group το οποίο περιλαμβάνει το στοιχείο i (γραμμή 4-8). Σε αντίθετη περίπτωση, το στοιχείο i εισάγεται στο τελευταίο group (γραμμή 10) και το group συνενώνεται με το προηγούμενο group αν έχουν επικάλυψη (γραμμή 11 και 37-41 αντιστοίχως). Ο υπολογισμός των τιμών $e_c(c)$, $w_c(c)$ και $q_c(c)$ πραγματοποιείται επαναληπτικά μέσω της συνάρτησης "AddCell" (γραμμή 24-26) και της "AddCluster" (γραμμή 29-31). Ο υπολογισμός αυτός είναι παρόμοιος με αυτόν του Σχήματος 2.3 με τη διαφορά ότι πλέον κάνουμε τη διαδικασία αυτή για τα στοιχεία του κάθε group και όχι της κάθε γραμμής. Ο υπολογισμός της βέλτιστης τοποθέτησης του group $x_c(c)$ υπολογίζεται στη γραμμή 33 όπου είναι όμοια με τη Σχέση 2.5.

Στις επόμενες γραμμές του κώδικα (γραμμές 34-35), ελέγχεται αν η νέα τοποθεσία είναι έγκυρη, δηλαδή βρίσκεται ανάμεσα στα όρια που έχουν τεθεί ($xmin$ και $xmax$). Αν η νέα τοποθεσία δεν ικανοποιεί αυτά τα όρια, τότε πρέπει να ευθυγραμμιστεί σε αυτά. Στις γραμμές 14-21, αναθέτονται οι θέσεις στα στοιχεία του κάθε group της γραμμής ξεκινώντας από το πρώτο και συνεχίζοντας την ανάθεση των συντεταγμένων σύμφωνα με τη Σχέση 2.5. Μετά το τέλος της συνάρτησης PlaceRow, το τετραγωνικό πρόγραμμα των Σχέσεων 2.1 και 2.4 έχει επιλυθεί

και ακολουθεί η επαναληπτική διαδικασία όπως αναφέρθηκε στην Ενότητα 2.3.1 (Σχήμα 2.5, γραμμή 4) μέχρι τον εντοπισμό της τοποθεσίας με τη μικρότερη μετακίνηση.

2.3.3 Παράδειγμα Abacus

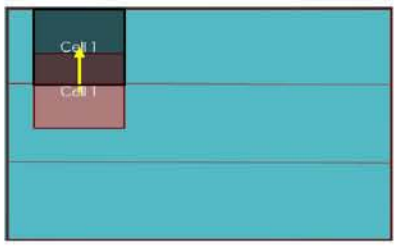
Στην ενότητα αυτή παρουσιάζουμε ένα απλό παράδειγμα της εκτέλεσης του Abacus για τη νομιμοποίηση πέντε στοιχείων τα οποία έχουν τοποθετηθεί κατά τη γενική τοποθέτηση σε ένα design με τρείς γραμμές (Σχήμα 2.9).



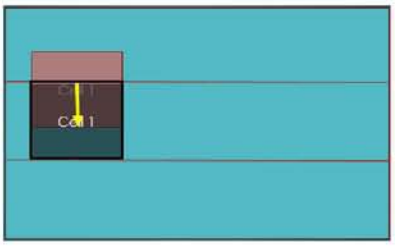
Σχήμα 2.9: Γενική τοποθέτηση.

Το πρώτο βήμα του αλγορίθμου είναι η διάταξη των στοιχείων βάση της συντεταγμένης που έλαβαν κατά τη γενική τοποθέτηση. Έτσι, στο συγκεκριμένο παράδειγμα θα έχουμε {Cell 1, Cell 5, Cell 4, Cell 2, Cell 3}.

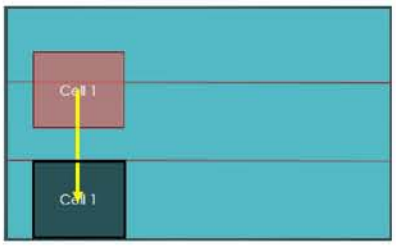
Για κάθε διατεταγμένο στοιχείο θα υπολογιστεί, για κάθε γραμμή, το κόστος μετακίνησης και θα επιλεγεί η μικρότερη δυνατή μετακίνηση. Για παράδειγμα για το στοιχείο Cell 1 θα έχουμε τις τρείς επιλογές που παρουσιάζονται στο Σχήμα 2.10. Και έτσι, θα επιλεγεί η μετακίνηση στη δεύτερη γραμμή (Σχήμα 2.11) εφόσον επιτυγχάνεται η μικρότερη μετακίνηση.



(α') Δοκιμαστική τοποθέτηση του Cell 1 στην πρώτη γραμμή με κόστος 2.02.



(β') Δοκιμαστική τοποθέτηση του Cell 1 στη δεύτερη γραμμή με κόστος 1.95.

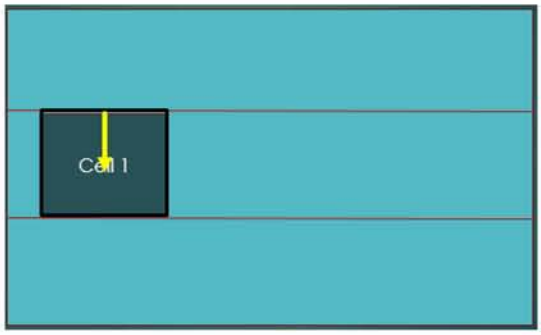


(γ') Δοκιμαστική τοποθέτηση του Cell 1 στην τρίτη γραμμή με κόστος 5.09.

Σχήμα 2.10: Δοκιμαστική τοποθέτηση του Cell 1.

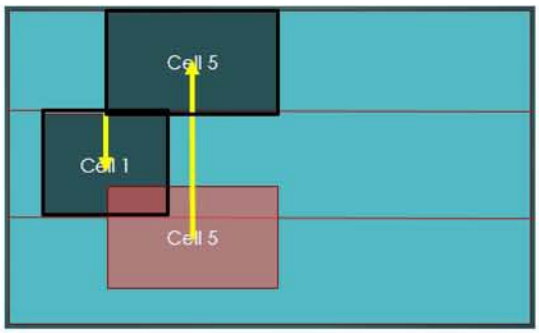
(τα κόστη που παρουσιάζονται είναι εικονικά και δεν αποτελούν πραγματικά αποτελέσματα)

Έτσι, θα επιλεγεί η μετακίνηση στη δεύτερη γραμμή (Σχήμα 2.11) μιας και επιτυγχάνεται η μικρότερη μετακίνηση.

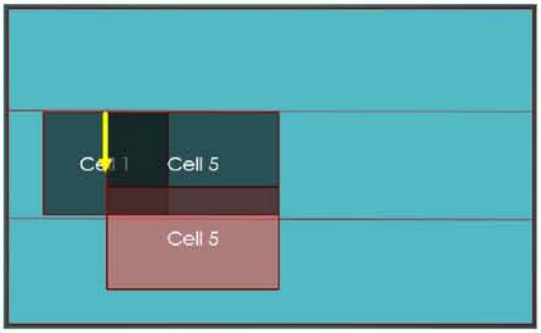


Σχήμα 2.11: Τελική τοποθέτηση του Cell 1 στη δεύτερη γραμμή με κόστος 1.95.

Ο αλγόριθμος συνεχίζει για το δεύτερο κατά τη διάταξη στοιχείο, δηλαδή για το Cell 5.



(α') Δοκιμαστική τοποθέτηση του Cell 5 στην πρώτη γραμμή.



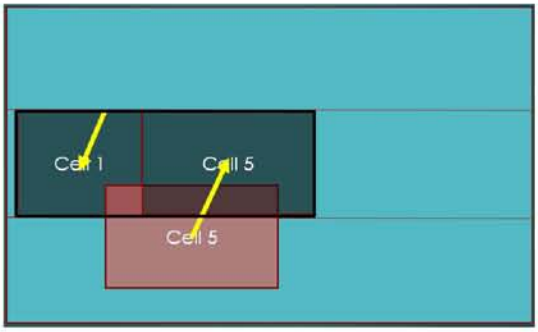
(β') Δοκιμαστική τοποθέτηση του Cell 5 στη δεύτερη γραμμή.

Σχήμα 2.12: Δοκιμαστική τοποθέτηση του Cell 5.

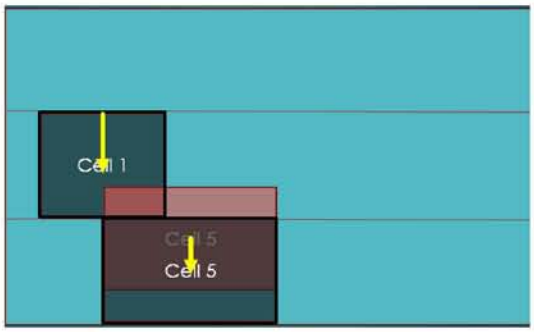
Στο Σχήμα 2.12β' παρουσιάζεται επικάλυψη με το στοιχείο Cell 1 το οποίο είναι προ τοποθετημένο. Έτσι, θα πραγματοποιηθεί συνένωση των δύο στοιχείων σε ένα group και θα υπολογιστεί η συνολική μετακίνηση όλων των στοιχείων του group (Σχήμα 2.13). Τέλος, ακολουθεί ο έλεγχος για την εισαγωγή του στοιχείου στην τελευταία γραμμή (Σχήμα 2.14α') η οποία θα είναι και αυτή με τη μικρότερη μετακίνηση (Σχήμα 2.14β').

Η διαδικασία συνεχίζει με παρόμοιο τρόπο μέχρι τη νομιμοποίηση του στοιχείου Cell 3. Όπου για τον έλεγχο της πρώτης γραμμή (Σχήμα 2.15) δεν προκύπτει κάποια μή νόμιμη κατάσταση.

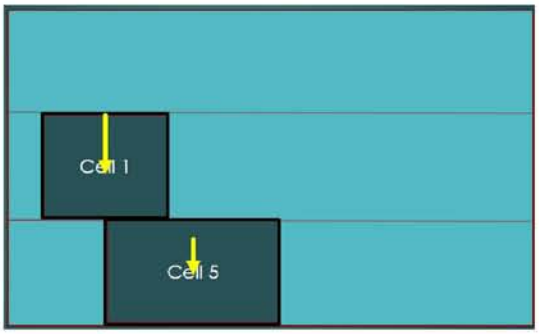
Ωστόσο, για την τοποθέτηση στη δεύτερη γραμμή θα προκύψει η κατάσταση που απεικονίζεται στο Σχήμα 2.16. Η κατάσταση αυτή αντιμετωπίζεται στον ψευδοκώδικα με τη συνάρτηση Collapse (γραμμή 34-35), στην οποία γίνεται έλεγχος αν τα group που δημιουργούνται δεν τηρούν τα όρια του κυκλώματος. Σε κάθε τέτοια περίπτωση, τα groups ευθυγραμμίζονται στα όρια του κυκλώματος (Σχήμα 2.17).



Σχήμα 2.13: Συγχώνευση επικαλυπτόμενων στοιχείων σε ένα group.

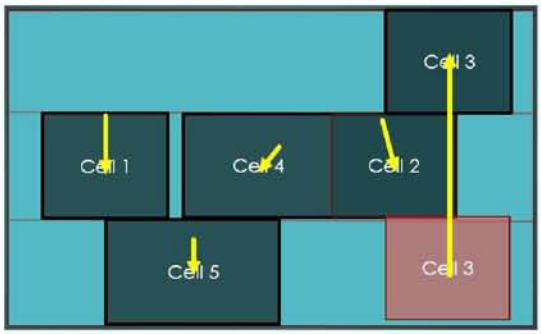


(α') Δοκιμαστική τοποθέτηση του Cell 5 στην τρίτη γραμμή



(β') Τελική τοποθέτηση του Cell 5 στη δεύτερη γραμμή

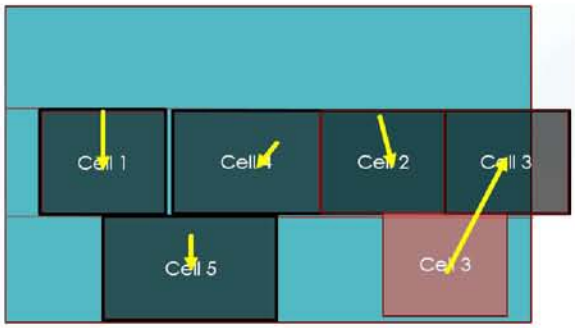
Σχήμα 2.14: Δοκιμαστική και τελική τοποθέτηση του Cell 5.



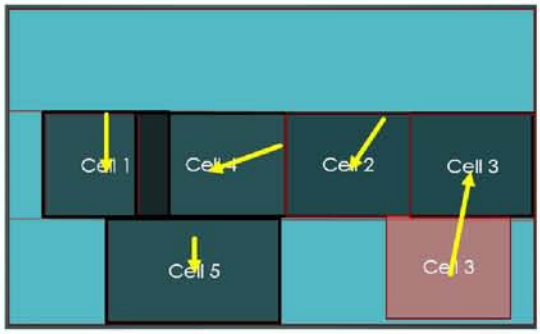
Σχήμα 2.15: Δοκιμαστική τοποθέτηση του Cell 3 στη πρώτη γραμμή.

Μετά την ευθυγράμμιση, παρατηρείται ότι υπάρχει επικάλυψη μεταξύ δύο group, το group που περιέχει το Cell 1 και αυτό που περιέχει τα Cell 4, Cell 2 και Cell 3 (Σχήμα 2.17). Έτσι, πραγματοποιείται συνένωση των δύο παραπάνω group (Σχήμα 2.18α'). Θα ακολουθήσει και ο έλεγχος για την τοποθέτηση στην τελευταία γραμμή, η οποία είναι και η επιλαχούσα (Σχήμα 2.18β').

Έτσι, η τελική νόμιμη τοποθέτηση όλων των στοιχείων παρουσιάζεται στο Σχήμα 2.19 με



Σχήμα 2.16: Δοκιμαστική τοποθέτηση του Cell 3 στη δεύτερη γραμμή.



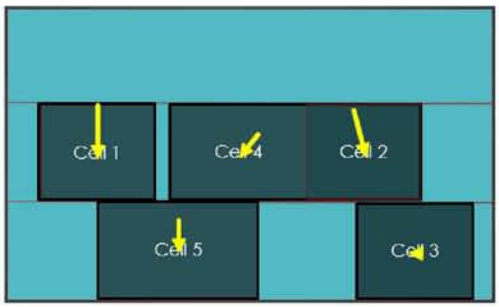
Σχήμα 2.17: Ευθυγράμμιση του group των στοιχείων Cell 4, Cell 2 και Cell 3 στη δεύτερη γραμμή.

(α') Συγχώνευση δύο group

(β') Δοκιμαστική τοποθέτηση του Cell 3 στην τρίτη γραμμή

Σχήμα 2.18: Δοκιμαστική και τελική τοποθέτηση του Cell 3.

κόστος όσο και το άθροισμα των μετακινήσεων των όλων των στοιχείων (άθροισμα μήκους κίτρινων γραμμών στο Σχήμα 2.19).



Σχήμα 2.19: Τελική τοποθέτηση όλων των στοιχείων.

2.4 Σύνοψη Κεφαλαίου

Στο παρόν κεφάλαιο παρουσιάστηκε διεξοδικά το πρόβλημα της νομιμοποίησης, η οποία αποτελεί σημαντικό βήμα στη διαδικασία τοποθέτησης των στοιχείων. Η διαδικασία έπεται της γενικής τοποθέτησης και προηγείται της λεπτομερούς τοποθέτησης.

Παρουσιάστηκε ακόμα ο αλγόριθμος Tetris. Ο Tetris, καθώς και οι παραλλαγές του, είναι ο αλγόριθμος νομιμοποίησης που χρησιμοποιείται κατά κόρον για τη νομιμοποίηση στη βιομηχανία. Ο λόγος της επιλογής του είναι η εύκολη υλοποίηση και περιγραφή του προβλήματος. Ωστόσο, τα αποτελέσματα του βασικού αλγορίθμου δεν είναι αξιοσημείωτα. Γι' αυτό το λόγο αναπτύχθηκαν μεταοχηματισμοί του αλγορίθμου για την επίτευξη καλύτερων αποτελεσμάτων.

Έτσι, παρουσιάστηκε μία παραλλαγή του Tetris, ο αλγόριθμος Abacus. Ο Abacus, σε αντίθεση με τον Tetris, μετακινεί τα ήδη τοποθετημένα στοιχεία των γραμμών για τη βελτίωση των αποτελεσμάτων. Ως αποτέλεσμα, ο Abacus επιτυγχάνει ελάχιστη συνολική μετακίνηση των στοιχείων, σε αντίθεση με τον Tetris. Ο αλγόριθμος χρησιμοποιεί δυναμικό προγραμματισμό για τον υπολογισμό των νόμιμων θέσεων των στοιχείων.

Αξίζει να σημειωθεί ότι τα στοιχεία, τόσο στον Tetris όσο και στον Abacus, δε χρειάζεται να είναι διατεταγμένα σε αύξουσα σειρά, όπως παρουσιάστηκε στις προηγούμενες ενότητες. Θα πρέπει να δοκιμαστούν διαφορετικές διατάξεις και να επιλεγεί η καλύτερη. Έχει παρατηρηθεί μέσω πειραμάτων ότι η διαφορά στη συνολική μετακίνηση των στοιχείων, ανάλογα αν επιλέχθηκε αύξουσα ή φθίνουσα διάταξη, μπορεί να αγγίξει και το 0.5% [11].

Ένα επιπλέον συμπέρασμα είναι ότι δε είναι απαραίτητο ένα στοιχείο να πρέπει να διατρέξει όλες τις γραμμές. Έτσι, τοποθετώντας το στοιχείο αρχικά και μετά δοκιμάζοντας τις κοντινές γραμμές ο αλγόριθμος μπορεί να κερδίσει χρόνο εκτέλεσης.

ΚΕΦΑΛΑΙΟ 3

Υλοποίηση και Επέκταση Αλγορίθμου Νομιμοποίησης

3.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζεται η υλοποίηση του αλγορίθμου Abacus που αποτέλεσε το αντικείμενο αυτής της διπλωματικής διατριβής. Θα ακολουθήσει η επεξήγηση των διαφορετικών προσεγγίσεων που υλοποιήθηκαν και των προβλημάτων που παρουσιάστηκαν για τη νομιμοποίηση των στοιχείων ενός κυκλώματος.

Αρχικά, μελετήθηκε η νομιμοποίηση στοιχείων σε επίπεδο κύκλωμα ή σε κύκλωμα το οποίο το θεωρήσαμε εμείς επίπεδο. Επίπεδο χαρακτηρίζεται ένα κύκλωμα όταν δεν έχει ή δε λαμβάνεται υπόψιν η ιεραρχική δομή του κυκλώματος. Έτσι, όλα τα στοιχεία ανήκουν στο ίδιο επίπεδο και τόσο η τοποθέτηση, όσο και η νομιμοποίηση, γίνεται σε ολόκληρο το κύκλωμα (Σχήμα 3.1) και όχι σε επιμέρους περιοχές.

Αντίθετα, το ιεραρχικό μοντέλο χωρίζει το συνολικό κύκλωμα σε περιοχές (Bounding Boxes) (Σχήμα 3.2) και η τοποθέτηση και η νομιμοποίηση πραγματοποιείται σε κάθε μια περιοχή ξεχωριστά. Οι ιεραρχικές προσεγγίσεις είναι πιο δύσκολο να υλοποιηθούν συγκριτικά με τις επίπεδες, αλλά μπορούν να παραλληλοποιηθούν. Η παραλληλοποίηση μπορεί να πραγματοποιηθεί διότι η τοποθέτηση και νομιμοποίηση αποτελούν ανεξάρτητα βήματα μεταξύ των διαφόρων περιοχών του κυκλώματος. Έτσι, με την παράλληλη υλοποίηση, ο χρόνος εκτέλεσης της νόμιμης τοποθέτησης μπορεί να μειωθεί σημαντικά.

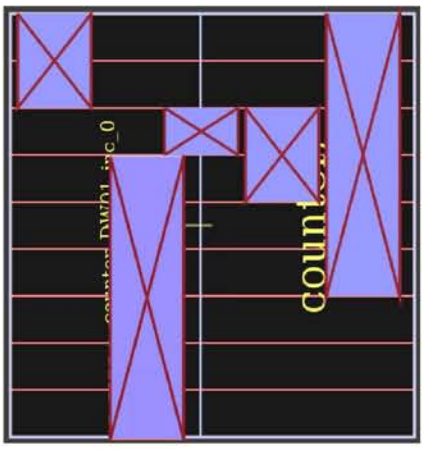
Τέλος, μελετήθηκε η νομιμοποίηση σε κυκλώματα τα οποία περιέχουν εμπόδια (Σχήμα 3.3). Τα εμπόδια μπορεί να είναι για παράδειγμα μνήμες ή αναλογικά κυκλώματα. Τα εμπόδια δεν επιτρέπεται να μετακινηθούν και τα υπόλοιπα στοιχεία πρέπει να τοποθετηθούν σε νόμιμες θέσεις στον υπολειπόμενο χώρο. Νόμιμη θέση εξακολουθεί να είναι εκείνη στην οποία τα στοιχεία είναι ευθυγραμμισμένα στις γραμμές και τις στήλες και δεν επικαλύπτονται με κανένα άλλο στοιχείο. Η επικάλυψη αφορά και τα εμπόδια, τα οποία αναπαρίστανται ως στοιχεία τα οποία δεν δύναται να μετακινηθούν.

Σχήμα 3.1: Παράδειγμα επίπεδου κυκλώματος.

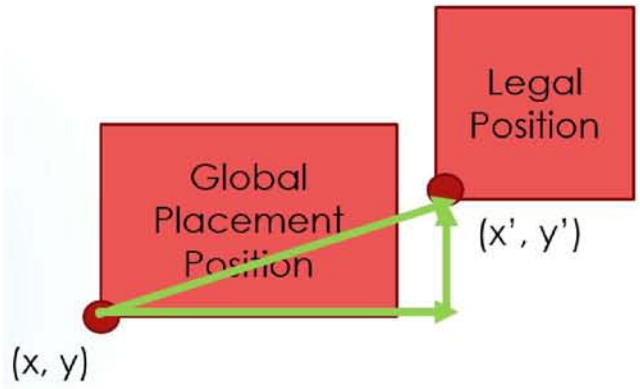
Σχήμα 3.2: Παράδειγμα ιεραρχικού κυκλώματος.

3.2 Συνάρτηση Κόστους Μετακίνησης

Ο υπολογισμός του κόστους μετακίνησης (αντίστοιχη γραμμή 7 στο Σχήμα 2.5) πραγματοποιήθηκε με δύο τρόπους. Ο πρώτος, με χρήση του αθροίσματος των μετακινήσεων (Total Displacement Function - F_{TD}) και ο δεύτερος με τον υπολογισμό της μέσης μετακίνησης των στοιχείων (Mean Displacement Function - F_{MD}). Ως μετακίνηση και για τις δύο προσεγγίσεις υπονοείται η Ευκλείδεια απόσταση μεταξύ της αρχικής και της τελικής θέσης του στοιχείου (Σχήμα 3.4). Η αρχική θέση είναι η μη νόμιμη θέση που είχε το στοιχείο μετά τη γενική τοποθέτηση.



Σχήμα 3.3: Παράδειγμα κυκλώματος με εμπόδια.



Σχήμα 3.4: Ευκλείδεια απόσταση δύο σημείων.

Έτσι, η F_{TD} υπολογίζει το συνολικό κόστος της Ευκλείδειας μετακίνησης ($C_{F_{TD}}$) σύμφωνα με τη Σχέση 3.1, για όλα τα στοιχεία, n, του κυκλώματος.

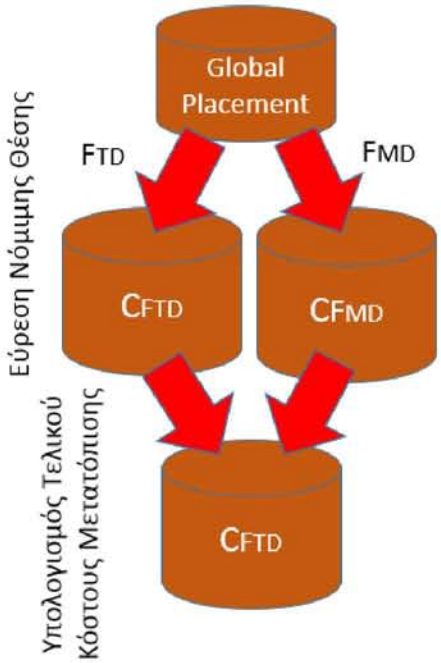
$$C_{F_{TD}} = \sum_{i=0}^{n} \sqrt{(y' - y)^2 + (x' - x)^2} \tag{3.1}$$

Όπου x', y' οι δοκιμαστικές νόμιμες συντεταγμένες και x, y οι συντεταγμένες κατά τη γενική τοποθέτηση. Οι συντεταγμένες χαρακτηρίζονται δοκιμαστικές γιατί πρέπει να εξερευνηθούν και οι υπόλοιπες γραμμές (Σχήμα 2.5, γραμμή 4).

Σε αντίθεση με τον κλασικό αλγόριθμο, ο οποίος χρησιμοποιεί τη F_{TD}, υλοποιήθηκε και η F_{MD}. Η προσέγγιση F_{MD} υπολογίζει το κόστος μετακίνησης, βρίσκοντας τη μέση τιμή της Σχέσης 3.1 ($C_{F_{MD}}$), δηλαδή όπως παρουσιάζει η Σχέση 3.2.

$$C_{F_{MD}} = \frac{C_{F_{TD}}}{n} \tag{3.2}$$

Το Σχήμα 3.5 παρουσιάζει την διαδικασία υπολογισμού του κόστους νομιμοποίησης. Δεδομένης μιας γενικής τοποθέτησης, ο αλγόριθμος επιλέγει την νόμιμη θέση του στοιχείου βάσει του μικρότερου κόστους μετατόπισης των στοιχείων του κυκλώματος. ο υπολογισμός του κόστους πραγματοποιείται είτε με τη χρήση της F_{TD} είτε με της F_{MD}. Η διαδικασία επαναλαμβάνεται μέχρι να νομιμοποιηθούν όλα τα στοιχεία του κυκλώματος. Τέλος για να βρεθεί το συνολικό κόστος νομιμοποίησης χρησιμοποιείται μόνο η F_{TD}, για να υπολογιστεί η πραγματική μετατόπιση των στοιχείων από την αρχική προς την τελική νόμιμη θέση.



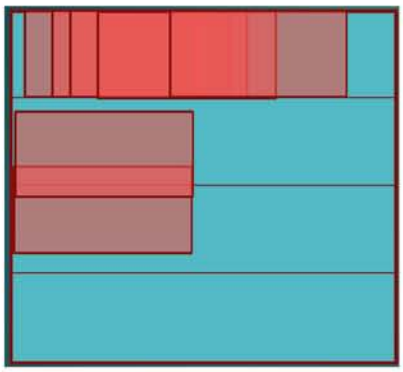
Σχήμα 3.5: Διαδικασία υπολογισμού συνολικού κόστους νομιμοποίησης.

3.3 Έλεγχος Χωρητικότητας Γραμμων

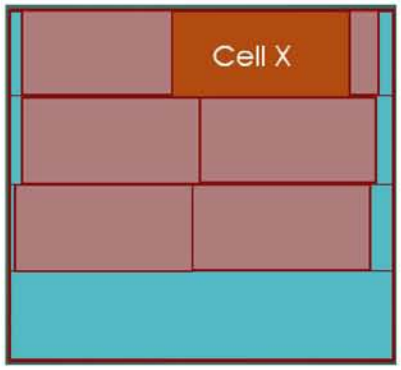
Κατά την άμεση μετατροπή του ψευδοκώδικα (Σχήμα 2.5 και 2.8) σε κώδικα C παρατηρήθηκε ότι ο ψευδοκώδικας δεν λαμβάνει υπόψιν του μία σημαντική παράμετρο. Δεν εξετάζει αν υπάρχει επαρκής χώρος σε μία γραμμή για να τοποθετήσει ένα στοιχείο. Ακολουθεί ένα παράδειγμα για να γίνει κατανοητό το πρόβλημα.

Στο Σχήμα 3.6 παρατηρούμε μία τοποθέτηση από τη γενική τοποθέτηση, όπου τα στοιχεία δεν είναι τοποθετημένα σε νόμιμες θέσεις.

Έτσι, μετά την εφαρμογή του αλγορίθμου για οριομένα στοιχεία, θα προκύψει η ακόλουθη κατάσταση (Σχήμα 3.7). Έστω ότι υπολείπεται να τοποθετηθεί το $CellX$ και θα δοκιμάσουμε να το τοποθετήσουμε σε κάθε γραμμή για να βρούμε την τοποθέτηση με την ελάχιστη συνολική μετακίνηση.

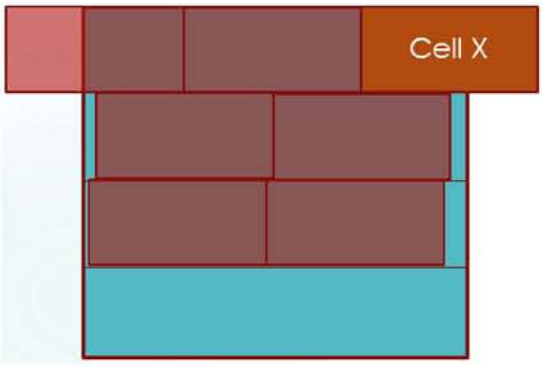


Σχήμα 3.6: Τυχαία τοποθέτηση.



Σχήμα 3.7: Εξέταση του στοιχείου $CellX$ προς νομιμοποίηση.

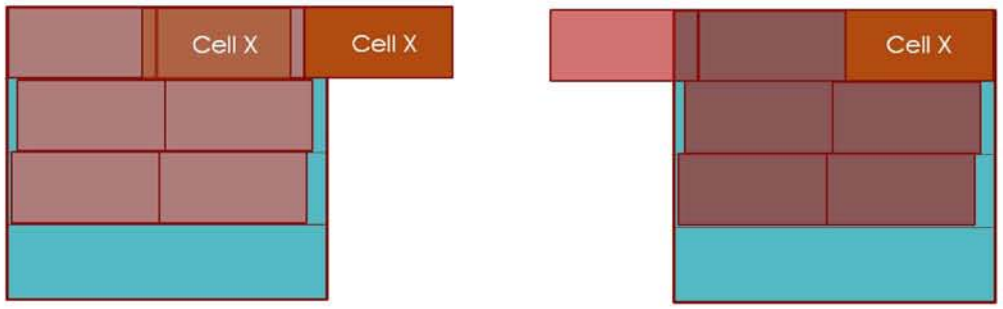
Για την πρώτη (και τη δεύτερη γραμμή ομοίως) θα έχουμε το εξής πρόβλημα. Θα τοποθετήσουμε το στοιχείο στη γραμμή, αλλά αυτό θα έχει επικάλυψη με το τελευταίο group της γραμμής. Έτσι, θα κληθεί η συνάρτηση Collapse (Σχήμα 2.8, γραμμή 32-41) η οποία θα συνενώσει και θα βρει τις συντεταγμένες του νέου group (Σχήμα 3.8).



Σχήμα 3.8: Τοποθέτηση εκτός περιθωρίων.

Ακόμα, η συνάρτηση ελέγχει αν οι συντεταγμένες που υπολόγισε είναι εντός ορίων του κυ-

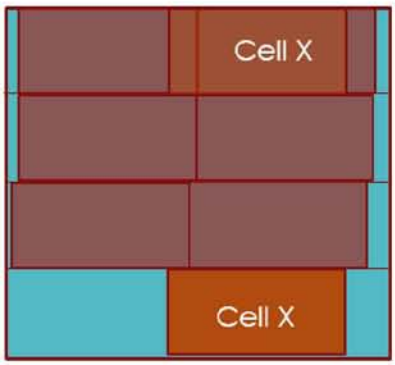
κλώματος. Αρχικά, ελέγχει αν το group βγήκε έξω από το αριστερό όριο (Σχήμα 2.8, γραμμή 34). Αν αυτό ισχύει (Σχήμα 3.8) τότε το ευθυγραμμίζει στο αριστερό όριο (Σχήμα 3.9α΄). Έπειτα ακολουθεί ο έλεγχος για το δεξί όριο (Σχήμα 2.8, γραμμή 35). Ομοίως, αν ισχύει το ευθυγραμμίζει στο δεξί όριο (Σχήμα 3.9β΄). Τέλος, θα υπολογιστεί η μετακίνηση των στοιχείων για την τοποθέτηση στην πρώτη γραμμή (ομοίως και στη δεύτερη) και θα ελεγχθεί η τρίτη γραμμή (Σχήμα 3.10).



(α΄) (β΄)

Σχήμα 3.9: Εκτέλεση συνάρτησης Collapse.

Στην τρίτη γραμμή δεν παρουσιάζεται κάποιο περίεργο φαινόμενο. Ωστόσο, έστω ότι η συνολική ελάχιστη μετακίνηση εκ των τριών γραμμών είναι η πρώτη. Τότε ο αλγόριθμος θα επιλέξει την πρώτη γραμμή, όπου θα καταστήσει την τοποθέτηση λανθασμένη (Σχήμα 3.9β΄).



Σχήμα 3.10: Εξερεύνηση τρίτης γραμμής.

Το πρόβλημα παρουσιάζεται όταν η συνολική χωρητικότητα της γραμμής είναι μικρότερη από το συνολικό μέγεθος των στοιχείων που επιχειρείται να τοποθετηθούν σε αυτή. Η λύση του προβλήματος είναι απλή. Πριν την τοποθέτηση του κάθε στοιχείου ελέγχεται η χωρητικότητα της γραμμής. Αν το στοιχείο χωράει στη γραμμή τότε επιχειρείται να τοποθετηθεί, αλλιώς γίνεται δοκιμή οε επόμενη γραμμή.

3.4 Εξαντλητική και Περιοριστική Εξερεύνηση Γραμμών

Σύμφωνα με τον ψευδοκώδικα που παρουσιάζεται στη δημοσίευση του Abacus [11], για την τοποθέτηση ενός στοιχείου πρέπει να γίνει έλεγχος όλων των γραμμών (Σχήμα 2.5, γραμμή 4) και να επιλεγεί η γραμμή, στην οποία η συνάρτηση μετακίνησης ελαχιστοποιείται. Ωστόσο, η προσέγγιση αυτή είναι αρκετά αργή, ειδικά σε designs με πολλές γραμμές. Ο λόγος του μεγάλου χρόνου εκτέλεσης είναι η εξαντλητική εξερεύνηση (Exhaustive Search - ES) των γραμμών.

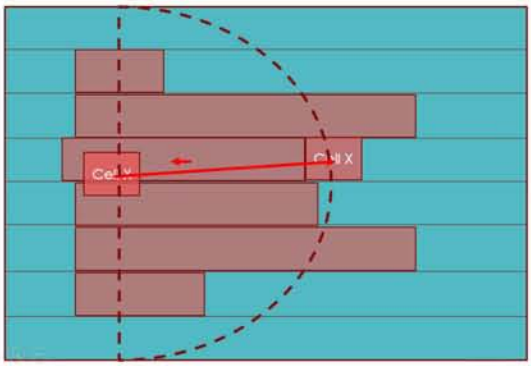
Για το λόγο αυτό, υλοποιήθηκε ο αλγόριθμος περιοριστικής εξερεύνησης γραμμών (Bounded Search - BS), όπου περιορίζει την εξερεύνηση όλων των γραμμών. Η βασική φιλοσοφία της συγκεκριμένης προσέγγισης είναι η χρήση εμβελειών R, οι οποίες υπολογίζονται είτε βάσει του $C_{F_{TD}}$, είτε του $C_{F_{MD}}$. Έτσι έστω row_h το ύψος της κάθε γραμμής του κυκλώματος τότε για τη F_{TD} ισχύει η Σχέση 3.3:

$$R = C_{F_{TD}}/row\_h \tag{3.3}$$

και για τη F_{MD} ισχύει η Σχέση 3.4

$$R = C_{F_{MD}}/row\_h. \tag{3.4}$$

Δεδομένης της R, ο αλγόριθμος ξεκινάει την εξερεύνηση από την παρούσα γραμμή και ελέγχει σε εμβέλεια R γραμμών (Σχήμα 3.11).



Σχήμα 3.11: Εμβέλεια εξερεύνησης γραμμών.

3.5 Διατάξεις Επιλογής Στοιχείων

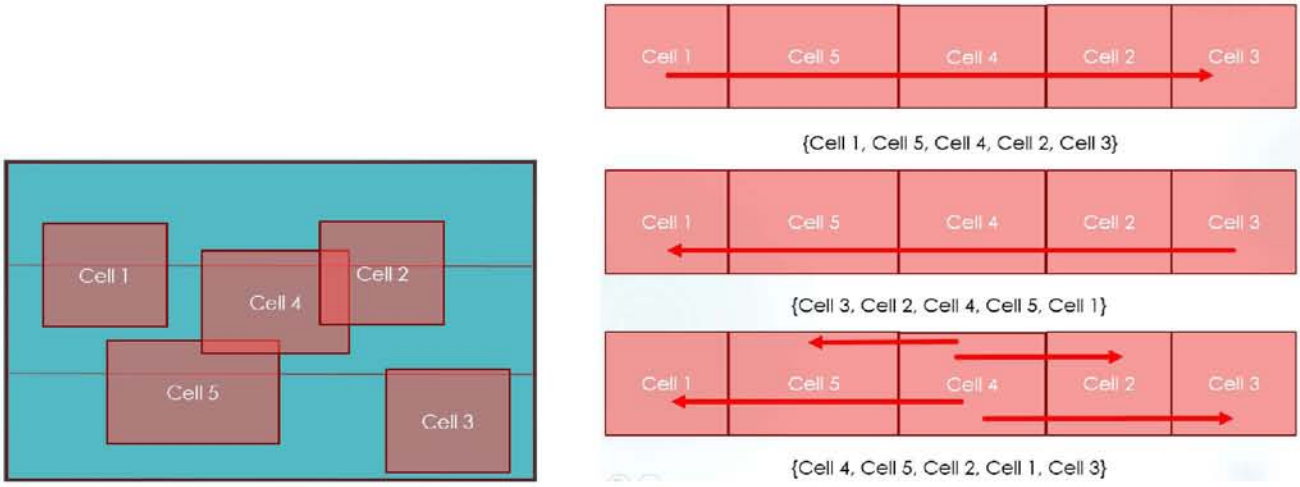
Στην ενότητα αυτή θα παρουσιάσουμε το πλαίσιο επιλογής των στοιχείων προς επιλογή. Τα στοιχεία διατάσσονται βάση της συντεταγμένης x που έλαβαν κατά τη γενική τοποθέτηση. Έπειτα, επιλέγεται ένα τη φορά και για να βρεθεί η νόμιμη τοποθέτηση. Η διάταξη εγγυάται ότι αν ένα στοιχείο a βρίσκεται αριστερά από ένα στοιχείο b κατά τη γενική τοποθέτηση, τότε το a

θα βρίσκεται αριστερά από το b και μετά τη νομιμοποίηση τους στην ίδια γραμμή. Η εγγυημένη διάταξη δεν ισχύει για στοιχεία τα οποία νομιμοποιούνται σε διαφορετικές γραμμές.

Έτσι, όπως αναφέρθηκε και στην Ενότητα 2.4 και στην αναφορά [11] η επιλογή των διατεταγμένων στοιχείων δεν είναι υποχρεωτικό ότι θα γίνει κατά αύξουσα τιμή του x. Στην πραγματικότητα πρέπει να δοκιμαστούν διαφορετικές προσεγγίσεις και να επιλεγεί η καλύτερη, δηλαδή εκείνη με τη μικρότερη μετακίνηση όλων των στοιχείων.

Σύμφωνα με τα παραπάνω, υλοποιήσαμε τρείς προσεγγίσεις. Στην πρώτη τα στοιχεία επιλέγονται με αύξουσα τιμή της συντεταγμένης x (Increasing Order - IO), στη δεύτερη με φθίνουσα (Decreasing Order - DO) και στην τελευταία από τη μέση (μεσαίο στοιχείο της διάταξης) προς το κοντινότερο στοιχείο (Mean to Nearest Order - MNO). Η μόνη διαφοροποίηση των τριών αυτών προσεγγίσεων είναι η σειρά επιλογής των στοιχείων από μία διατεταγμένη λίστα στοιχείων. Χαρακτηριστικά, στην τελευταία προσέγγιση, νομιμοποιείται το μεσαίο στοιχείο και μετά επιλέγεται το στοιχείο του οποίου η συντεταγμένη x είναι πιο κοντά στη συντεταμένη x του μεσαίου στοιχείου και δεν έχει επιλεγεί ακόμα.

Το Σχήμα 3.12 παρουσιάζει μία τυχαία τοποθέτηση και για τις τρείς διαφορετικές προσεγγίσεις.



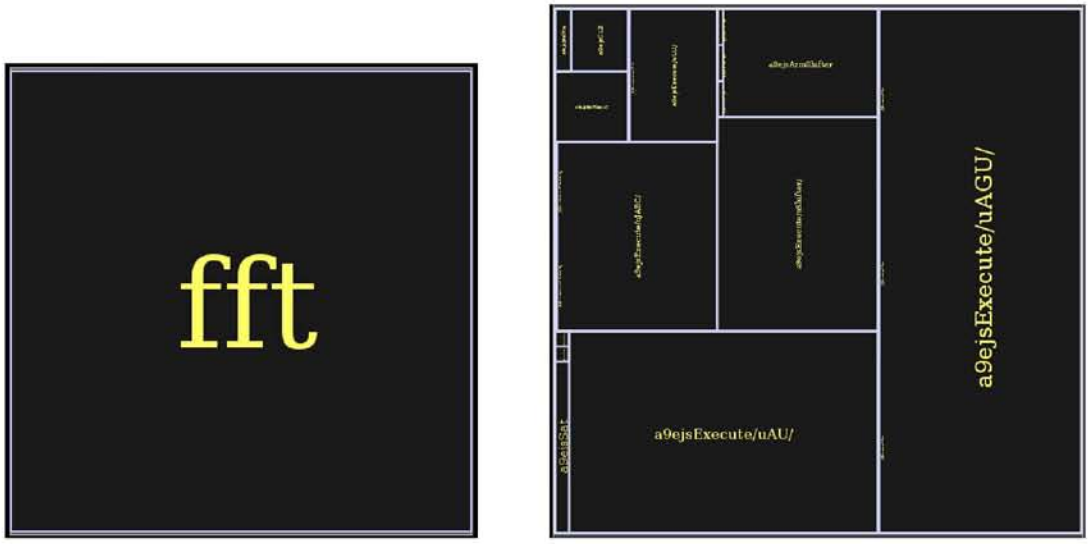
(α') Γενική τοποθέτηση. (β') Διατάξεις IO, DO, MNO.

Σχήμα 3.12: Διατάξεις επιλογής στοιχείων.

3.6 Προσέγγιση Επίπεδων και Ιεραρχικών Κυκλωμάτων

Στη συγκεκριμένη ενότητα θα παρουσιάσουμε τις προσεγγίσεις για τα δύο είδη κυκλωμάτων που αναλύθηκαν στην εισαγωγή του κεφαλαίου. Τα κυκλώματα χωρίζονται σε επίπεδα και ιεραρχικά. Στα επίπεδα κυκλώματα όλα τα στοιχεία ανήκουν στο ίδιο ¨επίπεδο¨ και η περιοχή νομιμοποίησης είναι η συνολική περιοχή που υπολογίστηκε για να τοποθετηθούν όλα τα στοιχεία

του κυκλώματος (Σχήμα 3.13α'). Αντίθετα, στα ιεραρχικά κυκλώματα, το κύκλωμα χωρίζεται σε υπό-περιοχές και η νομιμοποίηση πραγματοποιείται σε κάθε περιοχή ξεχωριστά (Σχήμα 3.13β'). Οι περιοχές αυτές καθορίζονται από τη διαδικασία χωροθέτησης.



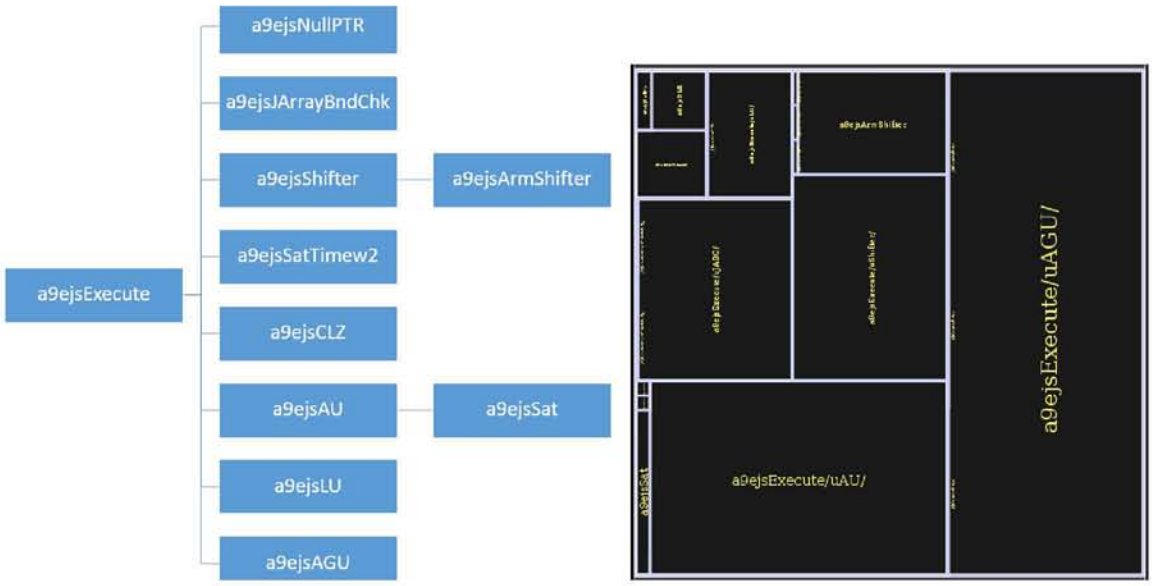
(α') Παράδειγμα επίπεδου κυκλώματος. (β') Παράδειγμα ιεραρχικού κυκλώματος.

Σχήμα 3.13: Επίπεδο και Ιεραρχικό Κύκλωμα.

Η χωροθέτηση, για τα παραδείγματα που παρουσιάζονται στη συγκεκριμένη διπλωματική εργασία, πραγματοποιήθηκε στα πλαίσια ενός προς ανάπτυξη βιομηχανικού EDA. Το συγκεκριμένο εργαλείο πραγματοποιεί την ιεραρχική χωροθέτηση βάσει της ιεραρχίας των modules της Verilog. Δεδομένου των στοιχείων που είναι οριομένα στην Verilog δημιουργείται μία ιεραρχική δομή δηλώσεων στοιχείων και submoduls. Έτσι, δεδομένου ενός πηγαίου κώδικα ενός κυκλώματος και το αντίστοιχο δέντρο ιεραρχίας Σχήμα 3.14α', τότε η χωροθέτηση θα είναι για παράδειγμα όπως αυτή στο Σχήμα 3.14β'. Είναι προφανές ότι σε κάθε επίπεδο της ιεραρχίας μπορεί να υπάρχουν είτε στοιχεία, είτε submodules, είτε και τα δύο.

Μετά τη χωροθέτηση ακολουθεί η γενική τοποθέτηση. Σε ό,τι αφορά την ιεραρχική προσέγγιση, τα στοιχεία του κάθε επιπέδου τοποθετούνται ανεξάρτητα από αυτά των άλλων επιπέδων. Παρόμοια εκτελείται και η νομιμοποίηση τους.

Η προσέγγιση της νομιμοποίησης για ιεραρχικά κυκλώματα, σε αυτή τη διπλωματική, πραγματοποιήθηκε με επαναληπτικό τρόπο. Έτσι, για το κάθε επίπεδο, πλέον, εκτελείται η νομιμοποίηση σαν να μην υπάρχουν τα υπόλοιπα υπό-κυκλώματα. Η βασική διαφορά στον αλγόριθμο υλοποίησης είναι τα όρια της τοποθέτησης και της νομιμοποίησης. Για παράδειγμα στο Σχήμα 3.15, το υπό-κύκλωμα $a9ejsExecute/uShifter$ καταλαμβάνει συγκεκριμένες γραμμές και έτσι ο αλγόριθμος θα πρέπει να εξετάσει μόνο αυτές. Επιπλέον, το υπό-κύκλωμα έχει και συγκεκριμένο πλάτος και έτσι η συνάρτηση Collapse πρέπει να ευθυγραμμίζει τα στοιχεία του υπό-κυκλώματος στα αντίστοιχα όρια του.



(α') Δένδρο ιεραρχίας βάσει της ιεραρχείας του κώδικα HDL

(β') Παράδειγμα ιεραρχικής χωροθέτησης

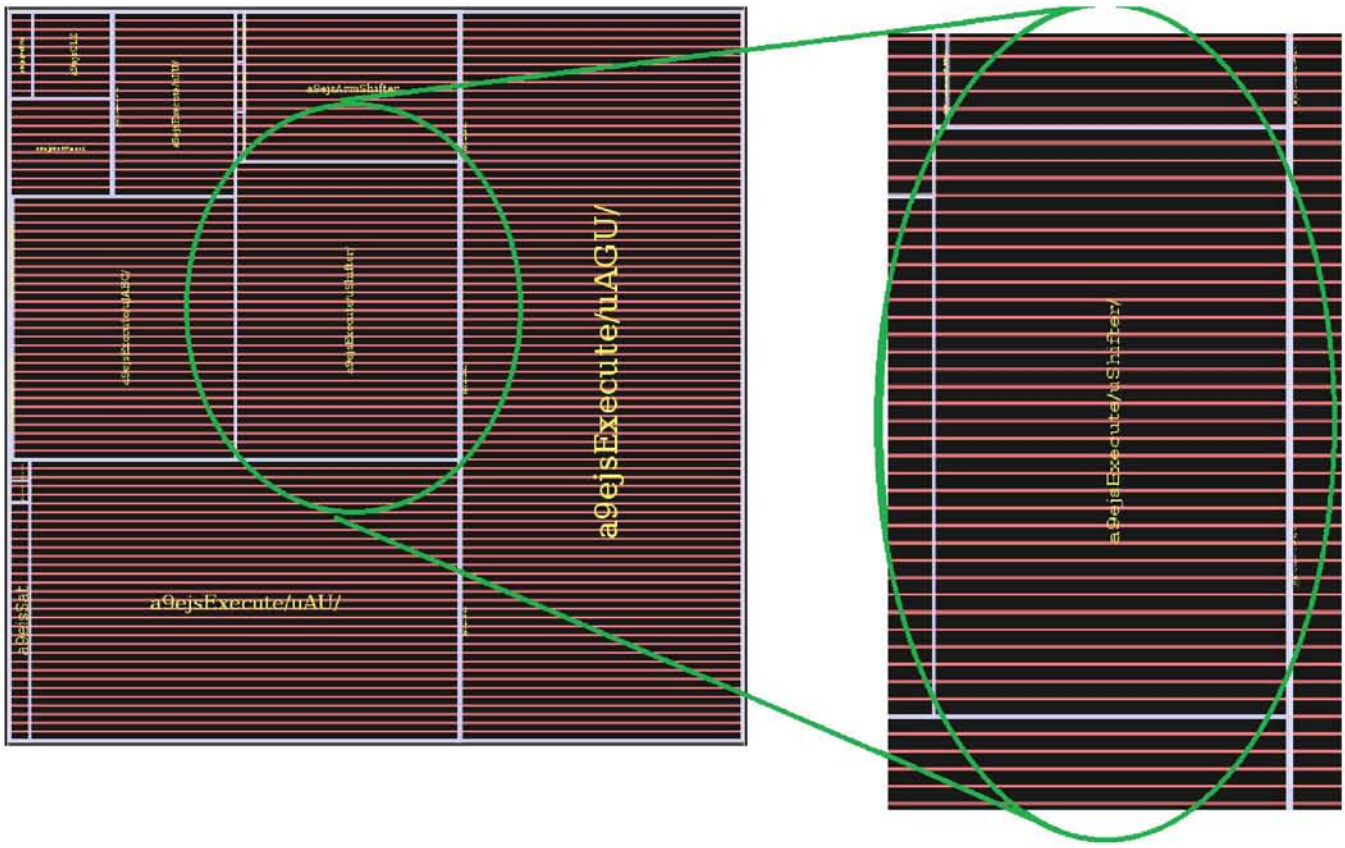
Σχήμα 3.14: Ιεραρχική χωροθέτηση.

3.7 Κυκλώματα με Εμπόδια

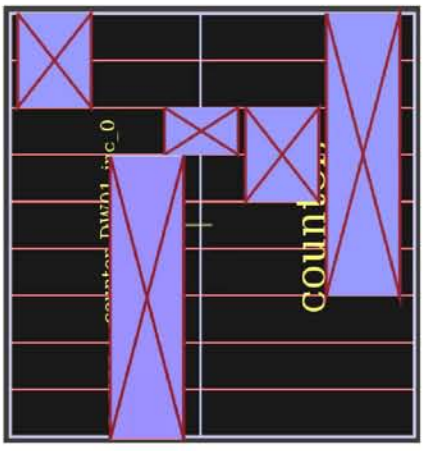
Τα κυκλώματα εκτός από τα στοιχεία τα οποία μπορούν να μετακινηθούν, περιέχουν και σταθερά στοιχεία. Τέτοια στοιχεία για παράδειγμα είναι οι μνήμες και τα αναλογικά κυκλώματα. Τα σταθερά στοιχεία μπορεί είτε να τοποθετηθούν πριν είτε μαζί με τα στοιχεία που επιτρέπεται να μετακινούνται. Στη δικιά μας υλοποίηση τα σταθερά στοιχεία ή αλλιώς ¨εμπόδια¨ είναι τοποθετημένα εκ των προτέρων. Έτοι, τα υπόλοιπα στοιχεία πρέπει να νομιμοποιηθούν στον υπολειπόμενο ελεύθερο χώρο.

Επιπλέον, τα σταθερά στοιχεία μπορεί να καταλαμβάνουν περισσότερες από μία γραμμές. Αξίζει να τονιστεί ότι τα standard cells, έχουν το ίδιο ύψος και αυτό είναι ίσο με το ύψος μίας γραμμής του κυκλώματος. Οι τιμές αυτές είναι καθοριομένες από τις βιβλιοθήκες οι οποίες σχεδιάζονται πριν τη σχεδίαση του κυκλώματος.

Το Σχήμα 3.16 παρουσιάζει ένα κύκλωμα, το οποίο έχει εμπόδια και αυτά είναι τοποθετημένα πριν την τοποθέτηση των υπολοίπων στοιχείων. Παρατηρείται επίσης, ότι τα εμπόδια μπορεί να καταλαμβάνουν περισσότερες γραμμές από μία, αλλά το ύψος τους κβαντίζεται σε πολλαπλάσιο γραμμών. Σε φυσικό επίπεδο, το εμπόδιο μπορεί να μην έχει ακριβώς κβαντισμένο ύψος, αλλά το εργαλείο EDA πρέπει να το απεικονίσει ώστε να υπονοείται ότι είναι κβαντισμένο στο ύψος των γραμμών.



Σχήμα 3.15: Νομιμοποίηση ενός υπό-κυκλώματος.

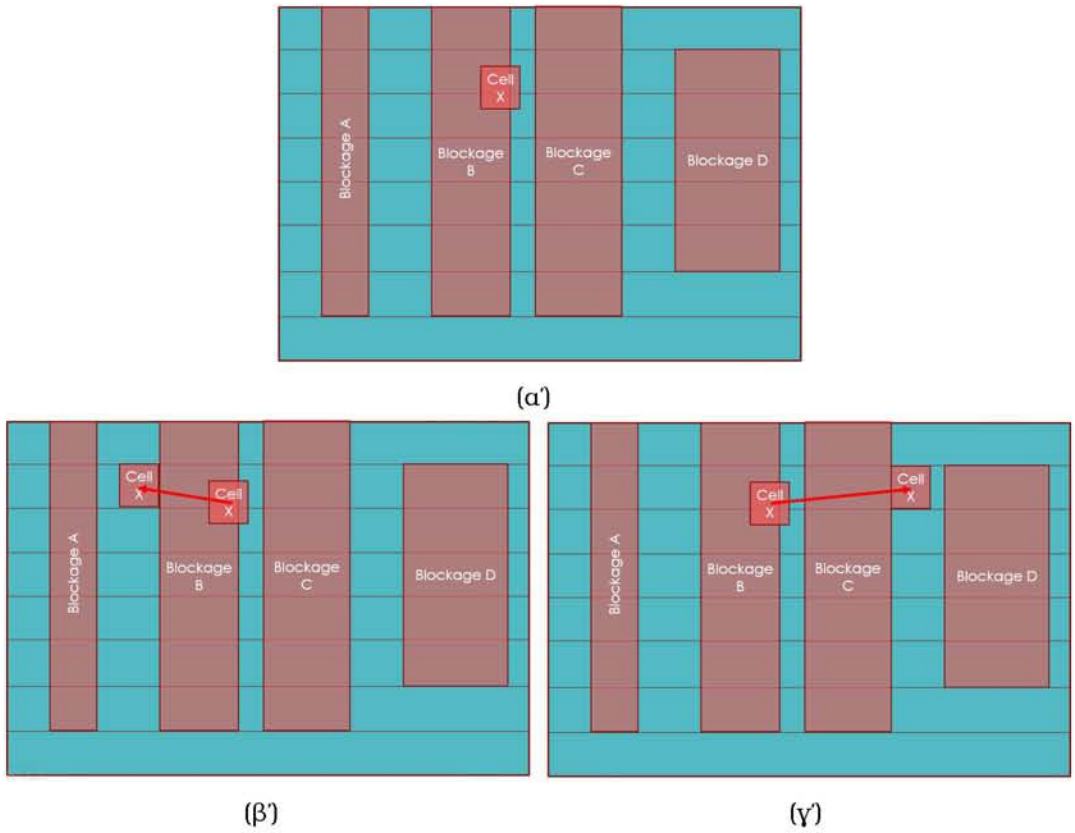


Σχήμα 3.16: Παράδειγμα κυκλώματος με εμπόδια.

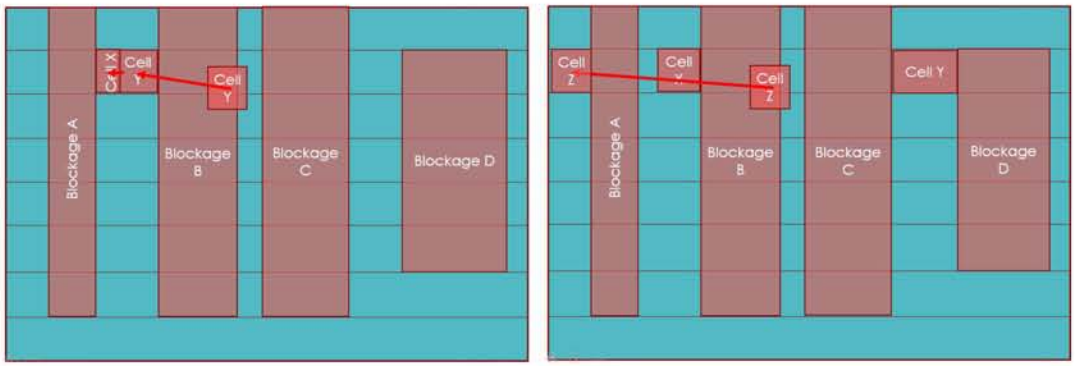
3.7.1 Νομιμοποίηση με Εμπόδια

Στα πλαίσια της διατριβής αυτής μελετήθηκαν δύο προσεγγίσεις για τη νομιμοποίηση κυκλωμάτων με εμπόδια, οι οποίες δεν υλοποιήθηκαν λόγο έλλειψης χρόνου. Η πρώτη είναι η εύρεση κενού (Find Free Space - FFS) και η δεύτερη η εύρεση κενού με μετακίνηση (Find Free Space with Movement - $FFSM$). Και στις δύο προσεγγίσεις οι γραμμές διασπώνται σε υπογραμμές ή περιοχές και πραγματοποιείται η νομιμοποίηση σε κάθε υπογραμμή. Οι περιοχές αυτές δεν έχουν απαραιτήτως το ίδιο μήκος.

Η FFS αναζητά την πιο κοντινή νόμιμη θέση, όχι για κάθε γραμμή αλλά για κάθε περιοχή. Έτσι, έστω το στοιχείο $CellX$ προς νομιμοποίηση (Σχήμα 3.17α'), τότε τα σχήματα 3.17β') και 3.17β') δείχνουν τις δύο πιο κοντινές νόμιμες θέσεις. Με τη συγκεκριμένη προσέγγιση τα στοιχεία που τοποθετούνται σε μία περιοχή εξακολουθούν να διατηρούν τη διάταξη που είχαν (Σχήμα 3.18α') κατά τη γενική τοποθέτηση (Σχήμα 3.17α'). Ωστόσο, για στοιχεία που είναι σε διαφορετικές περιοχές δεν ισχύει η διάταξη (Σχήμα 3.18β'), όπως συμβαίνει και στον κλασικό αλγόριθμο για τα στοιχεία που ανήκουν σε διαφορετικές γραμμές. Επιπλέον, αν ένα στοιχείο τοποθετηθεί σε μία περιοχή τότε δεν μπορεί να μετακινηθεί από αυτή την περιοχή.



Σχήμα 3.17: Εύρεση κοντινότερης θέσης για νομιμοποίηση.



(α') Στην ίδια περιοχή διατηρείται η διάταξη $\{X, Y\}$.

(β') Σε διαφορετικές περιοχές δεν διατηρείται η διάταξη $\{X, Y, Z\}$.

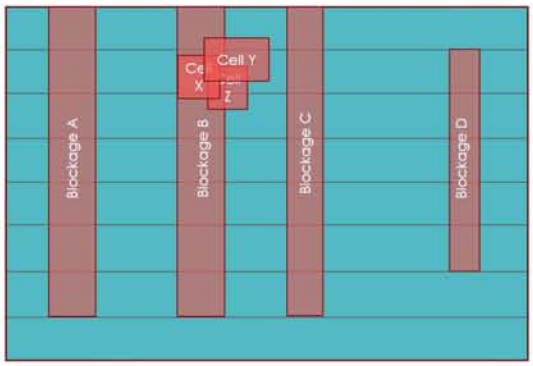
Σχήμα 3.18: Διάταξη σε κυκλώματα με εμπόδια.

Στη $FFSM$ αρχικά, γίνεται ο έλεγχος για τη νομιμοποίηση ενός στοιχείου σε μία περιοχή. Αν το στοιχείο χωράει στην περιοχή τότε τοποθετείται σύμφωνα με τον κλασικό τρόπο για τη δεδομένη περιοχή (Σχήμα 3.19δ'). Αν το στοιχείο δε χωράει στην περιοχή επειδή υπάρχουν προ-νομιμοποιημένα στοιχεία, τότε το νέο στοιχείο εξωθεί ορισμένα προ-υπάρχοντα στοιχεία να νομιμοποιηθούν σε προηγούμενες περιοχές της συγκεκριμένης γραμμής (Σχήμα 3.20). Με τη συγκεκριμένη μέθοδο, μπορεί να υπάρξει αλυσιδωτή μετακίνηση στοιχείων και έτσι η συνολική μετακίνηση των στοιχείων να γίνει αρκετά μεγάλη. Ακολουθεί ο ψευδοκώδικας και η ανάλυση της προσέγγισης αυτής.

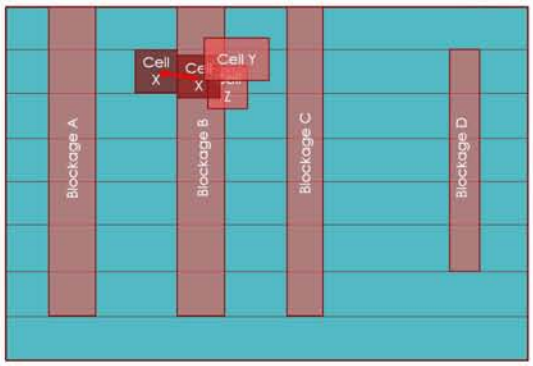
Έστω c το στοιχείο προς νομιμοποίηση και R το σύνολο των περιοχών σε μία συγκεκριμένη γραμμή. Επιπλέον, Gr το σύνολο των groups της περιοχής r και CGr το σύνολο των στοιχείων του group G της περιοχής r. Τέλος, έστω M το σύνολο των στοιχείων προς μετακίνηση.

Αρχικά εντοπίζεται η αρχική περιοχή cr στην οποία επιδιώκουμε να νομιμοποιήσουμε το στοιχείο c (ψευδοκώδικας Abacus_with_blockages, γραμμή 6-8). Ακολουθεί ο έλεγχος αν το στοιχείο μπορεί να τοποθετηθεί στην περιοχή αυτή χωρίς να χρειαστεί να μετακινήσουμε τα υπόλοιπα στοιχεία της περιοχής σε άλλη περιοχή (γραμμή 10). Με τον όρο "χωράει" υπονοούμε ότι η συνολική ελεύθερη χωρητικότητα της περιοχής cr είναι μεγαλύτερη από το μέγεθος του στοιχείου c. Στην πραγματικότητα αρκεί να ελεγχθεί μόνο το μήκος, διότι και το ύψος των περιοχών (ομοίως των γραμμών) και των στοιχείων είναι ίσο. Έπειτα, ακολουθεί ο έλεγχος για το αν το στοιχείο c επικαλύπτεται με τα στοιχεία της περιοχής (γραμμή 12-19), όπως πραγματοποιούνται και στον αλγόριθμο χωρίς εμπόδια.

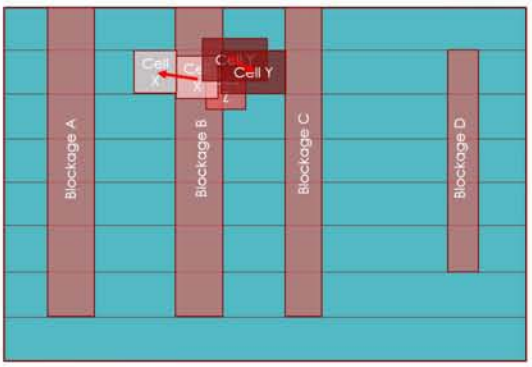
Αν το στοιχείο c είναι το πρώτο της περιοχής ή δεν επικαλύπτεται με το τελευταίο group, δημιουργείται ένα καινούριο group και τοποθετείται στα groups της περιοχής (γραμμή 14). Αν υπάρχει επικάλυψη του c με προϋπάρχον group τότε το c εισάγεται στα στοιχεία του τελευταίου group και υπολογίζονται οι καινούριες συντεταγμένες των στοιχείων του group (γραμμές 17-18). Αντιθέτως, αν το c δεν χωράει στην περιοχή cr τότε δοκιμάζεται α) να τοποθετηθεί στην αμέσως δεξιά περιοχή που χωράει (γραμμή 22) και β) να μετακινηθούν ορισμένα στοιχεία της περιοχής σε προηγούμενες, ώστε να χωρέσει στην περιοχή cr (γραμμή 25). Τέλος, επιλέγεται η



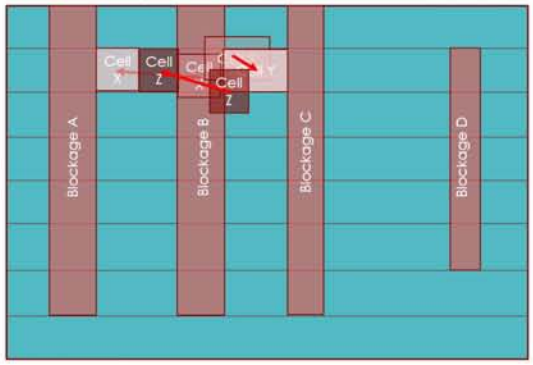
(α') Γενική τοποθέτηση με εμπόδια.



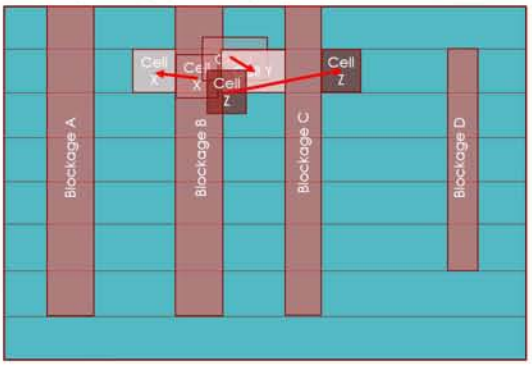
(β') Νομιμοποίηση στοιχείου X.



(γ') Νομιμοποίηση στοιχείου Y.



(δ') Αναζήτηση αριστερά για νομιμοποίηση στοιχείου Z.

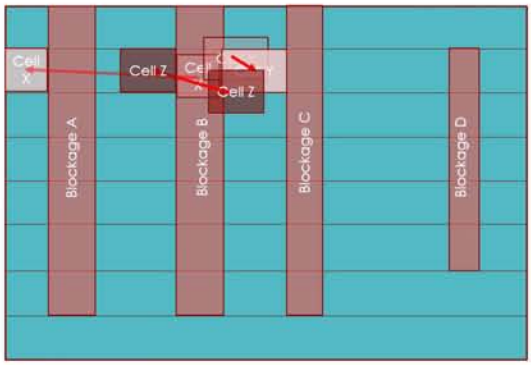


(ε') Αναζήτηση Δεξιά για νομιμοποίηση στοιχείου Z.

Σχήμα 3.19: Προσέγγιση μετατόπισης στοιχείων.

τοποθέτηση με τη μικρότερη μετακίνηση (γραμμή 27).

Ο έλεγχος για τη νομιμοποίηση σε δεξιότερη περιοχή είναι τετριμμένος, καθώς απλά γίνεται αναζήτηση της περιοχής που βρίσκεται δεξιά της cr και χωράει το c. Στην περίπτωση αυτή μπορεί να μην ισχύει η αρχική διάταξη καθώς μπορεί στοιχεία με μεγάλο πλάτος να νομιμοποιούνται σε περιοχές rl αρκετά δεξιά και μικρότερα σε πλάτος στοιχεία να τοποθετούνται σε περιοχές r, όπου $rc < r < r1$. Ωστόσο, η συγκεκριμένη περίπτωση δεν επηρεάζει την ορθότητα του αλγορίθμου.



Σχήμα 3.20: Αλλαγή διάταξης για νομιμοποίηση με μετατόπιση στοιχείων.

```
1  Ci = {};  // i group's components
2  M = {};   // moveable group
3  Gr = {};  // groups in region r
4
5  function Abacus_with_blockages(C, M, Gr, R, c) {
6    foreach region r in R // find current region
7      if c in r
8            cr = r;
9
10   if (c fits in cr)     // if c fits to current region
11     last_group = Gcr[|Gcr|]; // Gcr[|Gcr|]: last group in cr
12     if (|Gcr| == 0 || c has no overlap with last_group)
13       // add new group in the region
14       Gcr = Gcr U Gc; // Gc new group with c as its only component
15     else {
16       // add component to the last region's group
17       CGcr[|Gcr|] = CGcr[|Gcr|] U c; // CGcr[|Gcr|] components in Gcr[|Gcr|]
18       collapse(cr); // collapse region's groups
19     }
20   else {
21     // try to fit component on the right regions of the blockage
22     result = try_to_fit_component_on_the_right();
23     M = M U c;
24     // try to fit component on the left regions of the blockage
25     result = try_to_fit_M_on_the_left(cr_prev); // cr_prev: previous region of cr
26     // chose best movement of component (left or right)
27     choose_best();
28   }
```

Σχήμα 3.21: Αλγόριθμος νομιμοποίησης κυκλωμάτων με εμπόδια.

Όσον αφορά τη δεύτερη επιλογή, με τη μετατόπιση προς τα δεξιά στοιχείων, αυτή είναι πιο πολύπλοκη. Η πολυπλοκότητα έγκειται στην αλυσιδωτή μετατόπιση στοιχείων. Αρχικά, το c τοποθετείται στη λίστα των προς μετακίνηση στοιχείων (γραμμή 23 του ψευδοκώδικα Abacus_with_blockages) και έπειτα τοποθετείται στο τελευταίο group, μιας και υπάρχει επικάλυψη (ψευδοκώδικας try_to_fit_M_on_the_left, γραμμή 2). Έπειτα, υπολογίζονται τα στοιχεία που πρέπει να μετακινηθούν αριστερά, τα οποία αποτελούν το νέο σύνολο M (γραμμή 3-6). Αν δεν υπάρχει κάποιο στοιχείο προς μετακίνηση, δηλαδή το M είναι κενό, τότε η τοποθέτηση είναι έγκυρη (γραμμή 7-8). Αν υπάρχουν στοιχεία προς μετακίνηση, αλλά δεν υπάρχουν άλλες πε-

ριοχές για να τοποθετηθούν τότε η προσπάθεια τοποθέτησης του c στην περιοχή cr ή σε περιοχή αριστερά της είναι άκυρη (γραμμή 9-11). Τέλος, αν υπάρχουν στοιχεία προς μετακίνηση και υπάρχουν και περιοχές για να δοκιμαστεί η τοποθέτηση τους σε αυτές, εκτελείται η αναδρομική κλήση της συνάρτησης μετακίνησης στοιχείων προς τα αριστερά (γραμμή 12-19).

```
1  function try_to_fit_M_on_the_left(cr, M) {
2    Gcr[|Gcr|] = Gcr[|Gcr|] U M; // add moveable group to current region's groups
3    M = {};                      // reset moveable group
4    foreach ci in each Gcr // ci: component that does not fit to cr
5      M = M U ci; // insert component ci to moveable group
6    }
7    if (|M| == 0) // there is no components to move
8      return 1;
9    else if (|M| > 0 && cr_prev != NULL) // cr_prev: previous region of cr
10     // there is components to move and there are no regions
11     return 0;
12   else
13     // there is components to move and there are more regions
14     result = try_to_fit_M_on_the_left(cr);
15     if (result == 0)
16       return 0;
17     else
18       return 1;
19 }
```

Σχήμα 3.22: Αλγόριθμος αλυσιδωτής μετατόπισης αριστερά.

ΚΕΦΑΛΑΙΟ 4

Αποτελέσματα

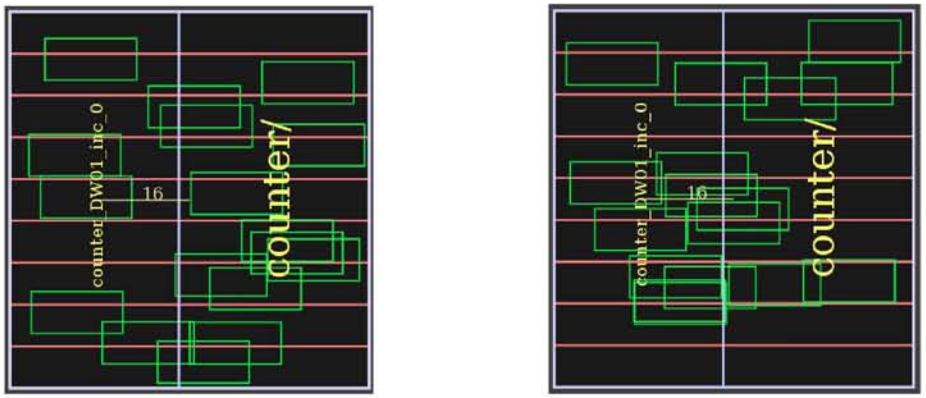
Στο παρόν κεφάλαιο παρουσιάζονται τα αποτελέσματα των πειραμάτων που πραγματοποιήθηκαν στα πλαίσια της διπλωματικής διατριβής. Τα αποτελέσματα πραγματεύονται συγκρίσεις του κόστους μετακίνησης και του χρόνου εκτέλεσης των διαφορετικών προσεγγίσεων που αναλύθηκαν στο Κεφάλαιο 4. Τα αποτελέσματα προέρχονται από συνολικά 200 πειράματα σε έξι κυκλώματα ποικίλου πλήθους στοιχείων και γραμμών. Επιπλέον, τα κυκλώματα που εξετάστηκαν ήταν τόσο επίπεδα όσο και ιεραρχικά. Ο Πίνακας 4.1 περιέχει το πλήθος στοιχείων και γραμμών, καθώς και το είδος των κυκλωμάτων που χρησιμοποιήθηκαν για τα πειράματα.

Όνομα κυκλώματος	Κύκλωμα1	Κύκλωμα2	Κύκλωμα3	Κύκλωμα4	Κύκλωμα5	Κύκλωμα6
Πλήθος Στοιχείων	17	382	546	718	2346	32281
Πλήθος Γραμμών	9	24	41	52	82	171
Είδος Κυκλώματος	Ιεραρχικό	Επίπεδο	Ιεραρχικό	Επίπεδο	Ιεραρχικό	Επίπεδο

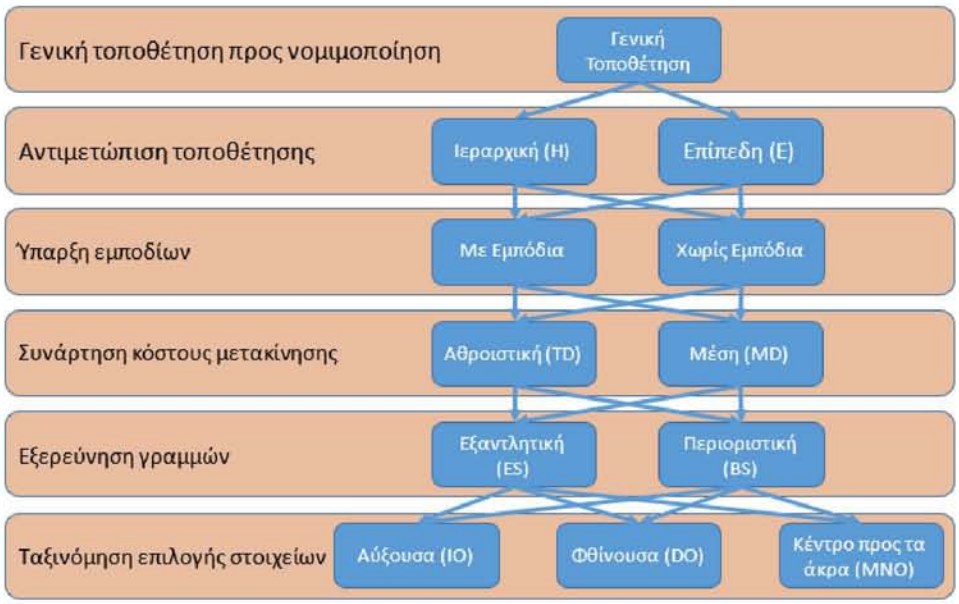
Πίνακας 4.1: Κυκλώματα Πειραμάτων

Έτσι, το πρώτο βήμα για τη νομιμοποίηση ήταν να δημιουργηθεί ένας τοποθετητής που να τοποθετεί τα στοιχεία ενός κυκλώματος σε τυχαίες θέσεις, οι οποίες ωστόσο πληρούν ορισμένες βασικές προδιαγραφές των τοποθετητών, όπως την τοποθέτηση των στοιχείων εντός της προδιαγεγραμμένης περιοχής του κυκλώματος. Τα Σχήματα 4.1 παρουσιάζουν δύο διαφορετικές τοποθετήσεις για το ίδιο κύκλωμα από τον τυχαίο αυτό τοποθετητή. Παρατηρείται ότι οι δύο τοποθετήσεις δεν είναι νόμιμες καθώς τα στοιχεία δεν είναι ευθυγραμμισμένα και έχουν επικαλύψεις, κάτι που θα το περιμέναμε και σε ένα αναλυτικό αλγόριθμο τοποθέτησης.

Το Σχήμα 4.2 απεικονίζει τις διαφορετικές προσεγγίσεις για τα πειράματα που πραγματοποιήσαμε. Τα αποτελέσματα των πειραμάτων παρουσιάζονται τμηματικά στις επόμενες ενότητες.



Σχήμα 4.1: Παρδείγματα τυχαίας τοποθέτησης.

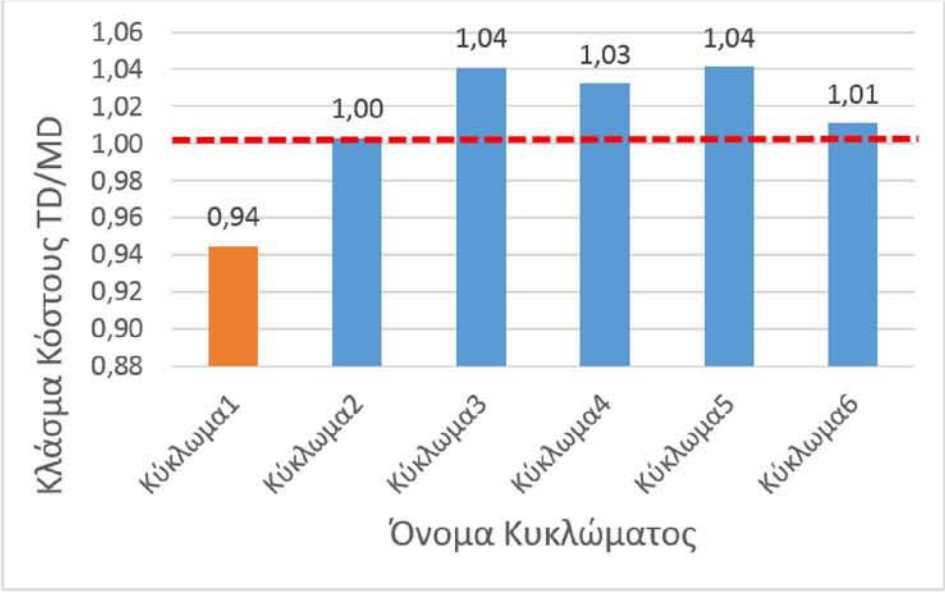


Σχήμα 4.2: Διάγραμμα προσεγγίσεων διπλωματικής εργασίας

4.1 Αποτελέσματα Κόστους Μετακίνησης και Χρόνου εκτέλεσης μεταξύ των F_{TD} και F_{MD}

Η επιλογή της συνάρτησης υπολογισμού κόστους μετακίνησης επηρεάζει όχι μόνο το κόστος μετακίνησης αλλά και τον χρόνο εκτέλεσης της νομιμοποίησης. Στα πειράματα που πραγματοποιήθηκαν χρησιμοποιήθηκαν οι συναρτήσεις υπολογισμού κόστους μετακίνησης F_{TD} και F_{MD} (Ενότητα 3.2).Το Σχήμα 4.3 παρουσιάζει το κλάσμα του κόστους μετακίνησης της F_{TD} προς τη F_{MD}, δηλαδή το $\frac{C_{F_{TD}}}{C_{F_{MD}}}$. Στο σημείο αυτό πρέπει να τονιστεί ότι οι δύο προσεγγίσεις χρησιμο-

ποιούνται μόνο κατά τη διάρκεια της νομιμοποίησης. Το συνολικό κόστος που υπολογίζουμε και αφορά την τελική μετακίνηση των στοιχείων υπολογίζεται μόνο με την F_{TD}. Έτσι, μπορούμε να συγκρίνουμε την επίδραση της κάθε προσέγγισης στη διαδικασία νομιμοποίησης. Μπορούμε να παρατηρήσουμε ότι για τα κυκλώματα: Κύκλωμα 2 ως 6, το κόστος μετακίνησης μειώνεται αν χρησιμοποιηθεί η συνάρτηση μέσου κόστους μετακίνησης. Ωστόσο, για το Κύκλωμα1 δεν ισχύει το ίδιο, όπου παρατηρούμε ότι το κόστος αυξάνεται με την επιλογή της F_{MD} έναντι της F_{MD}.

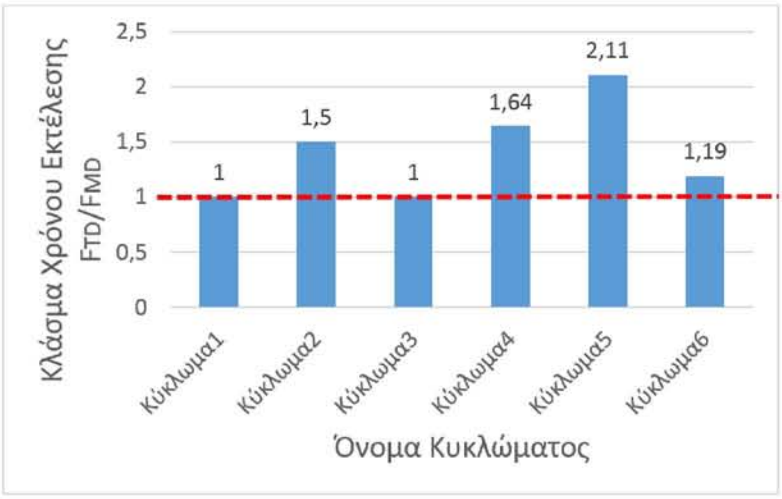


Σχήμα 4.3: Κλάσμα κόστους μετακίνησης $\frac{C_{F_{TD}}}{C_{F_{MD}}}$.

Επιπλέον, το Σχήμα 4.4 παρουσιάζει το κλάσμα του χρόνου εκτέλεσης της F_{TD} προς της F_{MD}. Παρατηρούμε, ότι η συνάρτηση μέσης μετακίνησης πραγματοποιεί γρηγορότερα τη νομιμοποίηση, μιας και ο λόγος είναι μεγαλύτερος της μονάδας. Κατά μέσο όρο, ο χρόνος εκτέλεσης με χρήση της F_{MD}, μειώνεται κατά 24% σε σχέση με τον της F_{TD}.

Η διαφοροποίηση τόσο του χρόνου εκτέλεσης, όσο και του κόστους μετακίνησης έγκειται στο γεγονός ότι το κόστος επηρεάζει την εξερεύνηση των λύσεων νομιμοποίησης. Όπως παρουσιάστηκε και στην Ενότητα 3.4, η περιοριστική εξερεύνηση γραμμών (BS) πραγματοποιείται βάση του κόστους μετακίνησης. Έτσι, όταν επιλέγεται η F_{MD} δεν είναι βέβαιο ότι το κάθε στοιχείο θα τοποθετηθεί σε νόμιμη θέση με τη μικρότερη δυνατή μετακίνηση. Υπάρχει περίπτωση να απορριφθούν έλεγχοι σε γραμμές, στις οποίες η νομιμοποίηση θα έχει μικρότερο κόστος όπως παρουσιάστηκε και στην Ενότητα 3.4. Αυτό προκύπτει γιατί το $C_{F_{MD}}$ θα είναι μικρότερο του $C_{F_{TD}}$. Κατά συνέπεια, η εμβέλεια, R, εξερεύνησης γραμμών θα είναι μικρότερη στη F_{MD} απ' ότι στην F_{TD}.

Ωστόσο, αν και η F_{MD} μπορεί να απορρίψει κάποιες βέλτιστες επιλογές, μπορεί μακροπρόθεσμα να μειώσουν τη συνολική μετακίνηση. Γι' αυτό το λόγο παρατηρείται και η βελτίωση στο



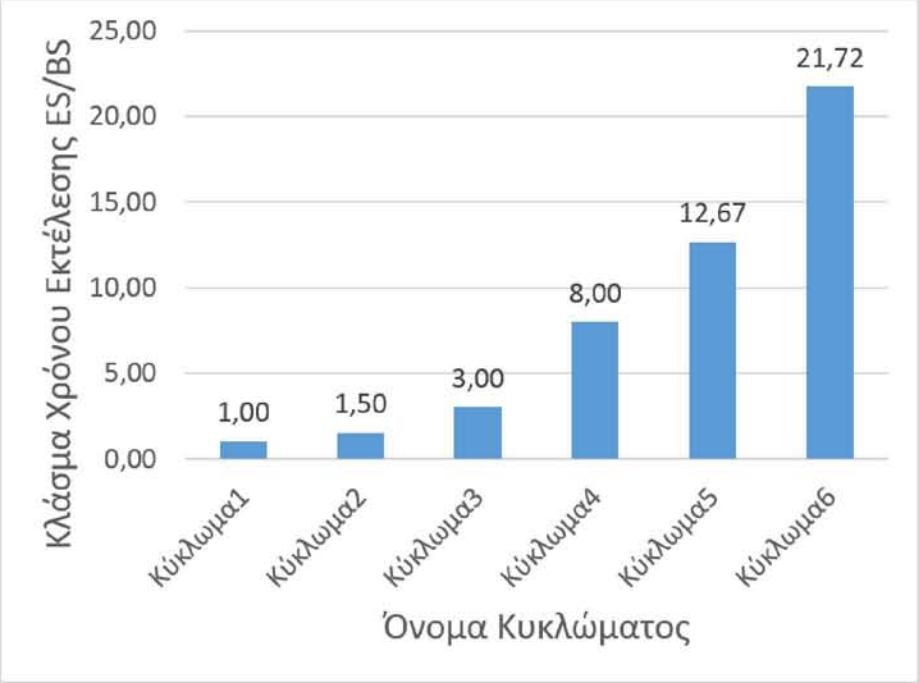
Σχήμα 4.4: Κλάσμα χρόνου εκτέλεσης των $\frac{F_{TD}}{F_{MD}}$.

συνολικό κόστος μετακίνησης (Σχήμα 4.4). Από την άλλη, ο χρόνος εκτέλεσης μειώνεται γιατί εξερευνώνται λιγότερες γραμμές σε σχέση με την F_{TD}, μιας και εξερευνώνται λιγότερες γραμμές (Σχήμα 4.3).

4.2 Αποτελέσματα Εξαντλητικής (ES) και Βέλτιστης (BS) Εξερεύνησης Γραμμών

Όπως αναφέραμε στην Ενότητα 3.4 υλοποιήθηκαν δύο προσεγγίσεις σχετικά με την εξερεύνηση γραμμών κατά τη νομιμοποίηση ενός στοιχείου. Η εξαντλητική (ES) και η περιοριστική (BS) εξερεύνηση. Στην ενότητα αυτή παρουσιάζουμε τα αποτελέσματα των πειραμάτων μεταξύ του χρόνου εκτέλεσης των εξερευνήσεων ES και BS. Το Σχήμα 4.5 παρουσιάζει το κλάσμα του χρόνου εκτέλεσης της ES προς την BS.

Όπως γίνεται κατανοητό από το Σχήμα 4.5, ο χρόνος εκτέλεσης μειώνεται δραματικά όταν δεν ελέγχονται όλες οι γραμμές. Ακόμα παρατηρούμε ότι όσο περισσότερες είναι οι γραμμές του κυκλώματος και τα στοιχεία που θέλουμε να νομιμοποιήσουμε, τόσο μεγαλύτερη επιτάχυνση επιτυγχάνουμε. Χαρακτηριστικό παράδειγμα το Κύκλωμα6, για το οποίο επιτυγχάνουμε επιτάχυνση με λόγο $21,72$. Στο σημείο αυτό αξίζει να υπενθυμίσουμε και να τονίσουμε ότι η επιλογή της BS σε σχέση με την ES δεν επιφέρει καμία αλλαγή στο συνολικό κόστος μετακίνησης.



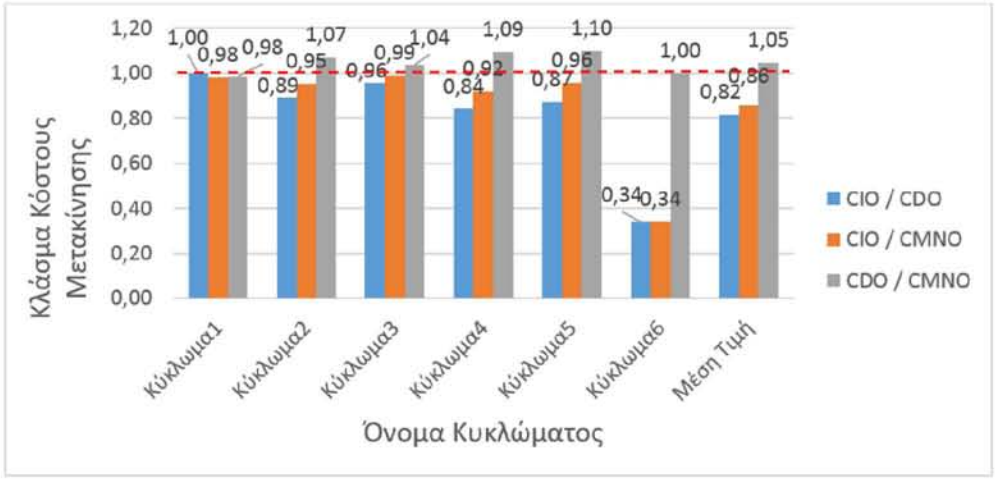
Σχήμα 4.5: Κλάσμα χρόνου εκτέλεσης $\frac{ES}{BS}$.

4.3 Αποτελέσματα Διατάξεων Επιλογής Στοιχείων

Ο Abacus, για να πραγματοποιήσει τη νομιμοποίηση, επιλέγει τα στοιχεία βάσει της συντεταγμένης x κατά τη γενική τοποθέτηση (Ενότητα 3.5). Η επιλογή της διάταξης επιφέρει διαφορετικά αποτελέσματα στο συνολικό κόστος μετακίνησης. Γι' αυτό το λόγο, ο νομιμοποιητής, πρέπει να δοκιμάζει διαφορετικές διατάξεις και να επιλέγει αυτή με τη μικρότερη μετακίνηση. Οι διατάξεις που δοκιμάστηκαν είναι κατά αύξουσα τιμή του x (Increasing Order - IO), κατά φθίνουσα (Decreasing Order - DO) και από στοιχείο με το μέσο x προς το κοντινότερο στοιχείο (Mean to Nearest Order - MNO) (Ενότητα 3.5).

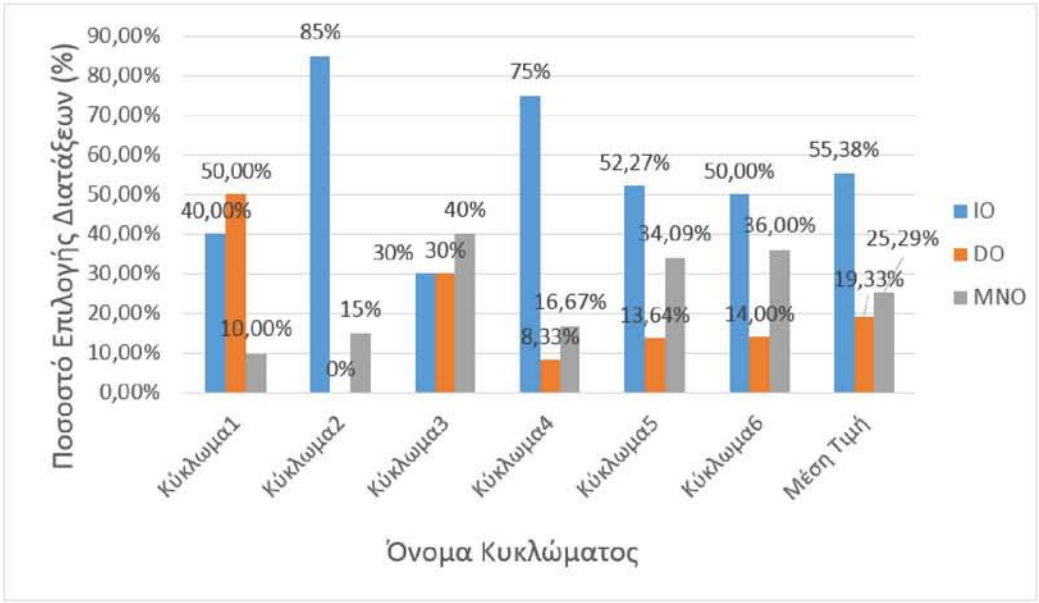
Σε ό,τι αφορά τη διαφορά στο κόστος μετακίνησης ανάλογα με τις προσεγγίσεις IO, DO, και MNO, παρατηρούμε στο Σχήμα 4.6, ότι η προσέγγιση IO επιφέρει μικρότερη μετακίνηση για το μέσο κόστος μετακίνησης των Κυκλωμάτων 1 ως 6. Αυτό γίνετε κατανοητό αν παρατηρήσει κανείς τα κλάσματα $\frac{IO}{DO}$ και $\frac{IO}{MNO}$ για τη μέση τιμή των αποτελεσμάτων, όπως παρουσιάζονται στο Σχήμα 4.6. Βλέπουμε ότι ο λόγος είναι 0.82 και 0.86 αντίστοιχα, άρα η IO επιφέρει μικρότερο κόστος. Αξίζει να σημειωθεί επιπλέον ότι η προσέγγιση MNO για τα Κυκλώματα 2 ως 5 μειώνει τη μετακίνηση των στοιχείων σε σχέση με την προσέγγιση DO.

Το Σχήμα 4.7, παρουσιάζει τα ποσοστά επιλογής της IO, της DO και της MNO, ως την προσέγγιση με τη μικρότερη μετακίνηση, για τα Κυκλώματα 1 ως 6. Παρατηρούμε ότι η προ-



Σχήμα 4.6: Κλάσματα κόστους μετακίνησης $\frac{C_{IO}}{C_{DO}}$, $\frac{C_{IO}}{C_{MNO}}$ και $\frac{C_{DO}}{C_{MNO}}$, για τα Κυκλώματα 1 ως 6.

σέγγιση IO επιφέρει κατά $55,38\%$ μικρότερη συνολική μετακίνηση, σε αντίθεση με το $19,33\%$ και $25,29\%$ των προσεγγίσεων DO και MNO αντίστοιχα.



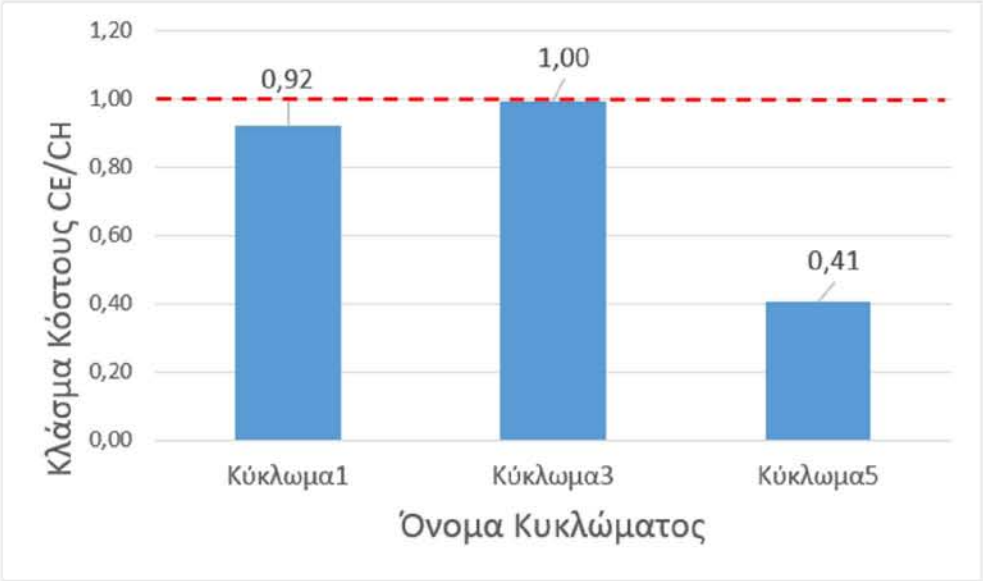
Σχήμα 4.7: Ποσοστά επιλογής των IO, DO και MNO.

4.4 Επίπεδη και Ιεραρχική Υλοποίηση

Όπως αναφέραμε και στην Ενότητα 3.6, τα κυκλώματα χωρίζονται σε δύο είδη. Τα επίπεδα (E) και τα ιεραρχικά (H). Στα επίπεδα κυκλώματα, η τοποθέτηση και η νομιμοποίηση χειρίζεται ταυτόχρονα όλα τα στοιχεία και ολόκληρη την επιφάνεια του κυκλώματος. Αντίθετα στα ιεραρχικά, τόσο τα στοιχεία όσο και η επιφάνεια του κυκλώματος χωρίζονται σε μικρότερα υποσύνολα. Ως επακόλουθο, μπορεί να πραγματοποιηθεί η τοποθέτηση και η νομιμοποίηση ανεξάρτητα σε καθένα από αυτά τα κομμάτια. Η διαδικασία αυτή μπορεί να παραλληλοποιηθεί μιας και η λύση του κάθε προβλήματος τοποθέτησης είναι ανεξάρτητη από τα υπόλοιπα.

Στα πειράματα που πραγματοποιήθηκαν χρησιμοποιήθηκαν τρία ιεραρχικά κυκλώματα, Κύκλωμα1, Κυκλώμα3 και Κύκλωμα5, και τρία επίπεδα. Τα ιεραρχικά κυκλώματα, αντιμετωπίστηκαν και ως ιεραρχικά και ως επίπεδα. Έτσι, έγινε η σύγκριση των δύο προσεγγίσεων και εξαγάγαμε τα συμπεράσματα που παρουσιάζονται στην ενότητα αυτή. Στα πειράματα που πραγματοποιήθηκαν η ιεραρχική δομή των κυκλωμάτων εξάγεται κατά τη χωροθέτηση βάσει της ιεραρχίας της Verilog (Ενότητα 3.6).

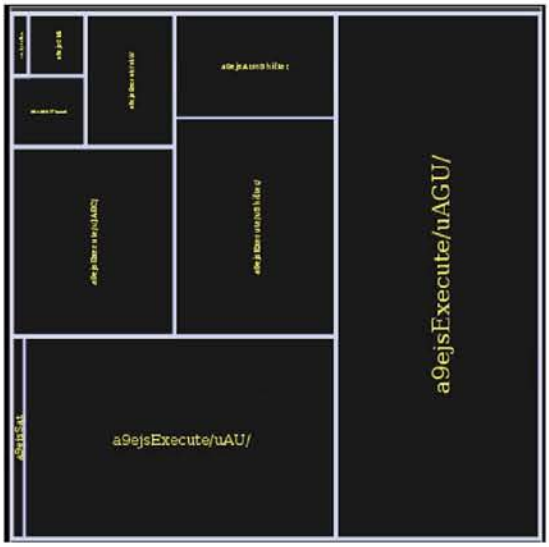
Σε ό,τι αφορά τη σύγκριση του κόστους μετακίνησης αν επιλεγεί η ιεραρχική προσέγγιση αντί της επίπεδης, παρατηρούμε ότι υπάρχει σημαντική αύξηση. Το Σχήμα 4.8 περιγράφει το κλάσμα $\frac{C_E}{C_H}$ του κόστους μετακίνησης της επίπεδης προς την ιεραρχική προσέγγιση για τα κυκλώματα Κύκλωμα1, Κύκλωμα3 και Κύκλωμα5.



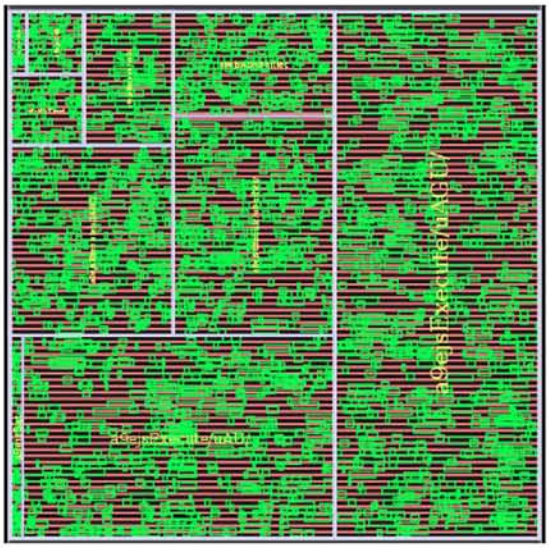
Σχήμα 4.8: Κλάσμα κόστους μετακίνησης $\frac{C_E}{C_H}$.

Παρατηρούμε για το Κύκλωμα5 ότι η ιεραρχική προσέγγιση, της οποίας η χωροθέτηση απεικονίζεται στο Σχήμα 4.9, παρουσιάζει δραματική αύξηση του κόστους μετακίνησης, διότι ο

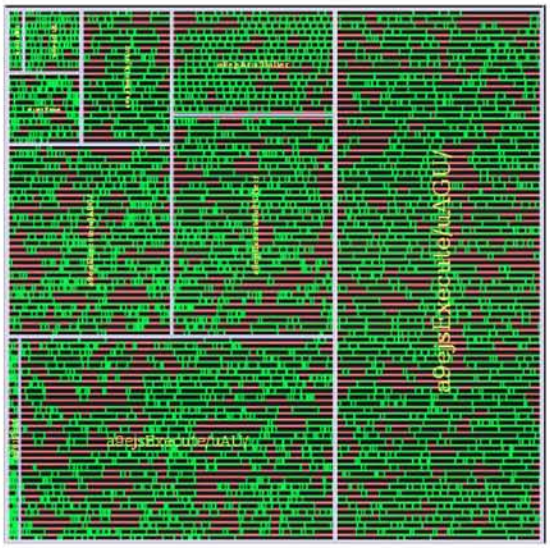
λόγος του κλάσματος $\frac{C_E}{C_H}$ είναι $0,41$. Αυτό συμβαίνει γιατί περιέχει όχι μόνο αρκετά στοιχεία προς νομιμοποίηση, αλλά και αρκετές περιοχές. Το Σχήμα 4.10α' παρουσιάζει ένα παράδειγμα γενικής τοποθέτησης και το Σχήμα 4.10β' τη νομιμοποίηση του κυκλώματος.



Σχήμα 4.9: Χωροθέτηση του κυκλώματος Κύκλωμα5.



(α') Γενική τοποθέτηση του κυκλώματος Κύκλωμα5



(β') Ιεραρχική νομιμοποίηση της τοποθέτησης του Σχήματος 4.10α'

Σχήμα 4.10: Γενική τοποθέτηση και ιεραρχική νομιμοποίηση του κυκλώματος Κύκλωμα5.

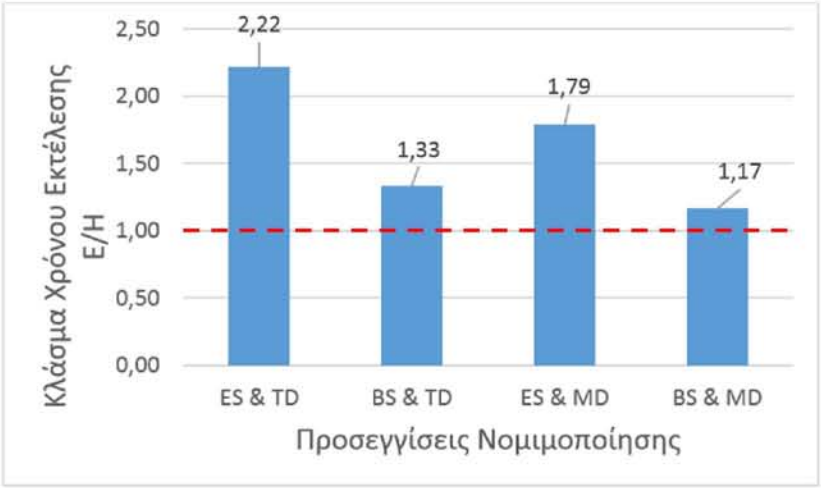
Το πρόβλημα της ραγδαίας αύξησης του κόστους μπορεί να αντιμετωπιστεί αλλάζοντας τη

χωροθέτηση του κυκλώματος (πχ. Σχήμα 4.11), έτσι ώστε να μειωθεί ο αριθμός των εξεταζόμενων περιοχών.

(α') (β')

Σχήμα 4.11: Διαφορετικές χωροθετήσεις για το Κύκλωμα5.

Το Σχήμα 4.12 παρουσιάζει το κλάσμα του χρόνου εκτέλεσης $\frac{E}{H}$. Για τον υπολογισμό του χρόνου νομιμοποίησης στα ιεραρχικά κυκλώματα, δεν πραγματοποιήθηκε καθόλου παραλληλισμός. Μια περιοχή νομιμοποιείται σειριακά μετά από τις υπόλοιπες. Ακόμα και χωρίς παραλληλισμό, παρατηρούμε ότι η ιεραρχική προσέγγιση είναι ταχύτερη έως και 2.22 φορές από την επίπεδη (στην περίπτωση που χρησιμοποιούμε την εξαντλητική εξερεύνηση γραμμών και την αθροιστική συνάρτηση κόστους). Αυτό συμβαίνει γιατί ο αλγόριθμος χειρίζεται, σε κάθε υπό-περιοχή, λιγότερα στοιχεία.



Σχήμα 4.12: Κλάσμα χρόνου εκτέλεσης $\frac{E}{H}$.

ΚΕΦΑΛΑΙΟ 5

Συμπεράσματα και Μελλοντική Ανάπτυξη

Στη συγκεκριμένη διπλωματική διατριβή, υλοποιήθηκε ο νομιμοποιητής Abacus. Η υλοποίηση μας, σε αντίθεση με την κλασική, υποστηρίζει όχι μόνο επίπεδα αλλά και ιεραρχικά κυκλώματα. Παράλληλα ελαχιστοποιεί το χρόνο εκτέλεσης της νομιμοποίησης θέτοντας φραγμούς στην εξερεύνηση των νόμιμων θέσεων. Τέλος, μελετήθηκε η υποστήριξη κυκλωμάτων με εμπόδια, αλλά λόγο περιορισμένου χρόνου δεν κατέστη δυνατή η υλοποίηση της σε γλώσσα C.

Η επιλογή του συγκεκριμένου τοποθετητή πραγματοποιήθηκε διότι είναι εύκολος στην κατανόηση, γρήγορος και στοχεύει στο να επιτύχει την ελάχιστη τροποποίηση της βέλτιστης λύσης. Σε αντίθεση με τους υπόλοιπους γρήγορους και άπληστους νομιμοποιητές της βιβλιογραφίας, ο Abacus επιτυγχάνει ελαχιστοποίηση της μετακίνησης.

Μέσω πειραμάτων, παρατηρήσαμε ότι ο χρόνος εκτέλεσης της νομιμοποίησης για την ιεραρχική προσέγγιση είναι σημαντικά μικρότερος απ' ότι για την επίπεδη, χωρίς καν την παραλληλοποίηση που κώδικα σε πολλαπλά νήματα ή διεργασίες. Ο λόγος της επιτάχυνσης αυτής είναι ο μικρότερος όγκος στοιχείων για την επίλυση του κάθε υπό προβλήματος. Ωστόσο, η μετακίνηση των στοιχείων είναι μεγαλύτερη στην ιεραρχική προσέγγιση και αυτό διότι ο χώρος τοποθέτησης είναι περιορισμένος. Έτσι τα στοιχεία πρέπει να μετακινηθούν μακριά από την βέλτιστη τοποθέτηση.

Επιπρόσθετα, υλοποιήθηκαν διαφορετικές προσεγγίσεις και ευριστικοί αλγόριθμοι για τη βελτίωση και τη σύγκριση αποτελεσμάτων. Έτσι, υπολογίστηκαν δύο συναρτήσεις υπολογισμού του κόστους μετακίνησης, η αθροιστική (F_{TD}) και η μέση (F_{MD}). Με τη χρήση της F_{MD}, η οποία επιδέχεται και μη βέλτιστες μετακινήσεις, παρατηρήθηκε ότι και το κόστος αλλά και ο χρόνος εκτέλεσης μπορεί να μειωθεί σημαντικά. Κατά συνέπεια, μία λιγότερο άπληστη επιλογή μετακίνησης μπορεί να επιφέρει μακροπρόθεσμα μικρότερη συνολική μετακίνηση.

Μια ακόμα ανάλυση που πραγματοποιήσαμε ήταν αυτή της διάταξης επιλογής των στοιχείων. Η επιλογή των στοιχείων προς νομιμοποίηση μπορεί να μειώσει σημαντικά τη συνολική μετακίνηση. Η διατάξεις που υλοποιήσαμε ήταν η αύξουσα (IO), η φθίνουσα (DO) και η από το κέντρο προς τα ακραία (MNO).

Συν τοις άλλης, έγινε μελέτη σχετικά με την υποστήριξη κυκλωμάτων με εμπόδια. Χαρακτηριστικά, προτείνουμε δύο προσεγγίσεις. Την FFS και την $FFMS$. Και στις δύο οι γραμμές

τμηματοποιούνται σε υπό γραμμές και πραγματοποιείται νομιμοποίηση σε κάθε υπό γραμμή. Η FFS αναζητά την πρώτη ελεύθερη και νόμιμη θέση για να τοποθετήσει το κάθε στοιχείο. Η προσέγγιση αυτή δεν διαφέρει από την φιλοσοφία της νομιμοποίησης που ακολουθεί η κλασική υλοποίηση. Ωστόσο, η μόνη διαφορά είναι ότι πλέον τα στοιχεία της κάθε γραμμής δε θα διατηρήσουν την αρχική τους διάταξη. Η $FFMS$ διατηρεί την διάταξη των στοιχείων. Για να το επιτύχει αυτό μετακινεί τα στοιχεία που έχουν τοποθετηθεί σε μία υπό γραμμή σε προηγούμενες. Η μετακίνηση πραγματοποιείται ώστε να δημιουργηθεί ικανός χώρος για την τοποθέτηση του εξεταζόμενου στοιχείου, σεβόμενοι πάντα την αρχική διάταξη. Η προσέγγιση αυτή μπορεί να τοποθετεί τα στοιχεία σύμφωνα με την αρχική διάταξη, αλλά η συνολική μετακίνηση των στοιχείων και ο χρόνος νομιμοποίησης θα είναι αυξημένοι σε σχέση με την προσέγγιση FFS. Η επιλογή της κάθε προσέγγισης έχει να κάνει με την φύση του κυκλώματος και τους στόχους του σχεδιαστή του κυκλώματος.

Τέλος, ο νομιμοποιητής προσαρμόστηκε σε ένα υπό ανάπτυξη βιομηχανικό εργαλείο [13] το οποίο περιέχει ολόκληρη τη ροή σχεδίασης κυκλωμάτων. Ως επακόλουθο, ο νομιμοποιητής υλοποιήθηκε για να πληροί τις βιομηχανικές απαιτήσεις και τους περιορισμούς.

Μελλοντικές επεκτάσεις της συγκεκριμένης εργασίας θα μπορούσαν να είναι οι :

- Υλοποίηση και σύγκριση μεθόδων για τη νομιμοποίηση κυκλωμάτων με εμπόδια
- Ενσωμάτωση νομιμοποιητή σε λεπτομερή τοποθετητή
- Σύγκριση λεπτομερούς τοποθετητή με τους υπόλοιπους ακαδημαϊκούς και βιομηχανικούς τοποθετητές
- Εξέταση μεγαλυτέρων κυκλωμάτων και περισσοτέρων πειραμάτων
- Δημιουργία αναλυτικού αλγορίθμου γενικής τοποθέτησης

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Charles J Alpert, Dinesh P Mehta, and Sachin S Sapatnekar. *Handbook of algorithms for physical design automation*. CRC press, 2008.

[2] José L Ayala, David Atienza Alonso, and Ricardo Reis. *VLSI-SoC: Forward-Looking Trends in IC and Systems Design: 18th IFIP WG 10.5/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2010, Madrid, Spain, September 27-29, 2010, Revised Selected Papers*, volume 373. Springer, 2012.

[3] Andrew E Caldwell, Andrew B Kahng, and Igor L Markov. Optimal partitioners and end-case placers for standard-cell layout. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 19(11):1304–1313, 2000.

[4] Konrad Doll, Frank M Johannes, and Kurt J Antreich. Iterative placement improvement by network flow methods. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 13(10):1189–1200, 1994.

[5] Dwight Hill. Method and system for high speed detailed placement of cells within an integrated circuit design, April 9 2002. US Patent 6,370,673.

[6] Sung Woo Hur and John Lillis. Mongrel: hybrid techniques for standard cell placement. In *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, pages 165–170. IEEE Press, 2000.

[7] Andrew B Kahng, Jens Lienig, Igor L Markov, and Jin Hu. *VLSI physical design: from graph partitioning to timing closure*. Springer Science & Business Media, 2011.

[8] Igor L Markov, Jin Hu, and Myung-Chul Kim. Progress and challenges in vlsi placement research. In *Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on*, pages 275–282. IEEE, 2012.

[9] Majid Sarrafzadeh, Maogang Wang, and Xiaojian Yang. *Modern placement techniques*. Springer Science & Business Media, 2013.

[10] Peter Spindler. *Efficient Quadratic Placement of VLSI Circuits*. PhD thesis, Universität München, 2008.

[11] Peter Spindler, Ulf Schlichtmann, and Frank M Johannes. Abacus: fast legalization of standard cell circuits with minimal movement. In *Proceedings of the 2008 international symposium on Physical design*, pages 47–53. ACM, 2008.

[12] Wern-Jieh Sun and Carl Sechen. Efficient and effective placement for very large circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 14(3):349–359, 1995.

[13] A S P (Automated Structured Placement) Tool.

[14] Laung-Terng Wang, Yao-Wen Chang, and Kwang-Ting Tim Cheng. *Electronic design automation: synthesis, verification, and test.* Morgan Kaufmann, 2009.