ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ & ΔΙΚΤΥΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΔΙΑΤΡΙΒΗ

Κουκάς Ιάσων

Α.Μ.: 1700016

Τίτλος Διατριβής:

«Βελτιστοποίηση κυκλωμάτων CMOS με τη μέθοδο ***Logical Effort***»

Επιβλέπων Καθηγητής:

Γεώργιος Σταμούλης

Βόλος, Σεπτέμβριος 2007

Περιεχόμενα:

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Γεώργιο Σταμούλη που δέχτηκε τη συνεργασία μαζί μου στα πλαίσια της διπλωματικής μου εργασίας καθώς και για την πολύτιμη επιστημονική και ηθική του βοήθεια καθ' όλη τη διάρκεια της φοίτησής μου, για το κίνητρο και το κουράγιο που μου έδωσε με την αμεσότητα και την εξυπηρετικότητά του ώστε να προσπαθώ για το καλύτερο.

Τον συμφοιτητή μου Αθανασόπουλο Παναγιώτη για την εποικοδομητική συνεργασία που είχαμε κατά τη διάρκεια του έτους 2003 αλλά και αργότερα, όταν η εργασία που κάναμε μαζί έγινε αφορμή για τη συνέχισή της στη διπλωματική μου εργασία.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για τη βοήθεια τους, κάθε είδους, όλα αυτά τα χρόνια.

Κεφάλαιο 1 - Εισαγωγή

Όπου κι αν ψάξουμε γύρω μας θα βρούμε ολοκληρωμένα κυκλώματα. Ολοκληρωμένα κυκλώματα βρίσκουμε στα ρολόγια που φοράμε, στις τηλεοράσεις, στις ηλεκτρικές συσκευές, στα οχήματα, στα κινητά τηλέφωνα και φυσικά στους προσωπικούς ηλεκτρονικούς υπολογιστές. Για παράδειγμα, ένας μικροεπεξεργαστής ενός ηλεκτρονικού υπολογιστή, είναι ένα ολοκληρωμένο κύκλωμα του οποίου ο σκοπός είναι η εκτέλεση πολύπλοκων υπολογισμών και η επεξεργασία ολόκληρης της πληροφορίας που αφορά σε έναν ηλεκτρονικό υπολογιστή. Αντιλαμβάνεται το πάτημα των κουμπιών στο πληκτρολόγιο καθώς και την κίνηση του ποντικιού από τον χρήστη. Μετράει αριθμούς, εκτελεί προγράμματα, παιχνίδια, και φυσικά το λειτουργικό σύστημα, το οποίο είναι το κυρίαρχο πρόγραμμα που «τρέχει» κάθε υπολογιστής μέσω του οποίου εκτελούνται όλα τα υπόλοιπα προγράμματα. Όμως τι είναι τελικά ένα ολοκληρωμένο κύκλωμα;

Ένα ολοκληρωμένο κύκλωμα δεν είναι τίποτα παραπάνω από ένα περισσότερο ανεπτυγμένο ηλεκτρικό κύκλωμα. Κάθε ηλεκτρικό κύκλωμα αποτελείται από διαφορετικά ηλεκτρικά στοιχεία όπως είναι τα τρανζίστορ, οι αντιστάτες, οι πυκνωτές, οι δίοδοι και άλλα στοιχεία, τα οποία είναι συνδεδεμένα μεταξύ τους με ποικίλους διαφορετικούς τρόπους. Κάθε ένα από αυτά τα στοιχεία έχει διαφορετική συμπεριφορά. Για παράδειγμα, ένα τρανζίστορ λειτουργεί με απλά λόγια ως ένας διακόπτης. Μπορεί να αφήσει να περάσει το ηλεκτρικό ρεύμα ή να το διακόψει, αλλά μπορεί και να το ενισχύσει. Είναι το βασικό συστατικό των περισσότερων ολοκληρωμένων κυκλωμάτων.

1.1. Ολοκληρωμένα Κυκλώματα

Υπάρχουν απεριόριστα είδη ολοκληρωμένων ηλεκτρικών κυκλωμάτων. Κάθε ένα σχεδιάζεται και κατασκευάζεται για κάποιον συγκεκριμένο σκοπό, προσπαθώντας να λύσει ένα συγκεκριμένο πρόβλημα. Για παράδειγμα η κάρτα γραφικών ενός ηλεκτρονικού υπολογιστή είναι ένα ολοκληρωμένο κύκλωμα όπως είναι και ο επεξεργαστής, με τη διαφορά ότι η κάρτα γραφικών έχει σχεδιαστεί αποκλειστικά για να εκτελεί ταχύτατα πράξεις που αφορούν σε μαθηματικά γραφικών, ενώ ο κύριος επεξεργαστής είναι σχεδιασμένος έτσι ώστε να εκτελεί πράξεις γενικού σκοπού οι οποίες συναντώνται στα περισσότερα προγράμματα που τρέχουμε.

Βιβλιογραφία [2]. Τα ολοκληρωμένα κυκλώματα χωρίζονται σε κλίμακες πολυπλοκότητας, ανάλογα με τον αριθμό των τρανζίστορ από τα οποία αποτελούνται. Το πρώτο τσιπ ημιαγωγού αποτελείτο από ένα και μόνο τρανζίστορ. Μέσα από διαδοχική εξέλιξη της επιστήμης των τρανζίστορ και των τρόπων διασύνδεσής τους, σε αυτά τα πρώτα τσιπ προστέθηκαν περισσότερα τρανζίστορ με σκοπό την αύξηση των λειτουργιών που θα μπορούσε να εκτελέσει ένα και μοναδικό τσιπ. Σιγά σιγά, οι ξεχωριστές λειτουργίες μετετράπησαν σε ολόκληρα συστήματα, κατασκευασμένα να εξυπηρετήσουν συγκεκριμένες πολύπλοκες λειτουργίες, όπως είναι για παράδειγμα μια κάρτα δικτύου η οποία εκτελεί γρήγορα, συγκεκριμένους υπολογισμούς που είναι απαραίτητοι για τη μεταφορά δεδομένων από χρήστη σε χρήστη μέσω του Ίντερνετ. Τα πρώτα κυκλώματα αποτελούνταν από λίγα μόνο στοιχεία, ίσως περίπου από *δέκα* (διόδους, τρανζίστορ, αντιστάτες, πυκνωτές), κάνοντας όμως δυνατή την κατασκευή μιας ή περισσότερων λογικών πυλών πάνω σε μια και μόνο συσκευή. Οι λογικές πύλες τελικά άνοιξαν το δρόμο για την εκτέλεση λογικών πράξεων στις συσκευές αυτές, πράγμα που εκτόξευσε τις δυνατότητες των ηλεκτρονικών υπολογιστών σε ύψη που ίσως δεν τα γνωρίζουμε ακόμη. Οι πρώτες αυτές συσκευές, σήμερα, όμως αναδρομικά, είναι γνωστές και ως συσκευές ολοκλήρωσης μικρής κλίμακας (SSI). Μέσα από βελτιώσεις στις τεχνικές κατασκευής και σχεδίασης, κατασκευάστηκαν συστήματα με *εκατοντάδες* λογικές πύλες (LSI), δηλαδή συστήματα με τουλάχιστον *χίλιες* λογικές πύλες. Η ίδια διαδικασία οδήγησε σε ολοκληρωμένα κυκλώματα (ICs) με χιλιάδες συσκευές, κάνοντάς τα LSI (μεγάλη κλίμακα ολοκλήρωσης).

Η σημερινή τεχνολογία έχει ξεπεράσει κατά πολύ εκείνο το σημείο και οι σημερινοί μικροεπεξεργαστές αποτελούνται από *εκατομμύρια* πυλών ή *εκατοντάδες εκατομμύρια* διαφορετικών τρανζίστορ. Πλέον είναι πραγματικότητα επεξεργαστές του ενός *δισεκατομμυρίου* τρανζίστορ οπότε και μιλάμε για VLSI (πολύ μεγάλης κλίμακας ολοκλήρωση).

Τα ηλεκτρικά κυκλώματα χωρίζονται σε δύο μεγάλες κατηγορίες: στα *αναλογικά κυκλώματα* και τα *ψηφιακά κυκλώματα*. Τα περισσότερα μεγάλης κλίμακας ολοκληρωμένα κυκλώματα (VLSI) αποτελούνται από υποκυκλώματα και των δύο κατηγοριών. Αυτός ο διαχωρισμός όπως θα γίνει κατανοητό στην επόμενη ενότητα γίνεται διότι τα κυκλώματα της κάθε κατηγορίας έχουν διαφορετική λειτουργικότητα αλλά και διαφορετική διαδικασία σχεδίασης.

1.1.1. Αναλογικά Κυκλώματα

Αναλογικά, λέμε τα ηλεκτρικά κυκλώματα στα οποία υπάρχει η ανάγκη τα ηλεκτρικά τους σήματα να παίρνουν συνεχείς τιμές ώστε να ανταποκρίνονται στην αναλογική πληροφορία που αναπαριστούν. Τέτοια κυκλώματα είναι οι ενισχυτές τάσης, ισχύος και πολλά ακόμα κυκλώματα, ενώ ενδεικτικά, περιέχονται στα ραδιόφωνα και στους ενισχυτές των ηχοσυστημάτων.

1.1.2. Ψηφιακά Κυκλώματα

Στην πραγματικότητα, και τα ψηφιακά κυκλώματα ανήκουν στην κατηγορία των αναλογικών κυκλωμάτων, όμως ένα αναλογικό κύκλωμα για να μπορεί να χαρακτηριστεί ως ψηφιακό, πρέπει να ακολουθεί ένα συγκεκριμένο πρότυπο λειτουργίας, δηλαδή η πληροφορία που αναπαριστά να μπορεί να εκληφθεί ως ψηφιακή. Ψηφιακή χαρακτηρίζεται η πληροφορία που λαμβάνει διακριτές τιμές για να αναπαραστήσει λογικές και αριθμητικές τιμές που αναπαριστούν με τη σειρά τους την χρήσιμη προς επεξεργασία πληροφορία. Τα σήματα λοιπόν που διαδίδονται στα ψηφιακά κυκλώματα πρέπει να μπορούν να εκληφθούν και να επεξεργαστούν ως διακριτά σήματα. Στα ψηφιακά κυκλώματα, τα τρανζίστορ χρησιμοποιούνται επί το πλείστον ως διακόπτες για να σχηματίσουν *λογικές πύλες*. Το πλεονέκτημα των ψηφιακών κυκλωμάτων είναι πως η λειτουργικότητά τους μπορεί να σχεδιαστεί με βάση τα αφαιρετικά μοντέλα των λογικών πυλών σε αντίθεση με τα αμιγώς αναλογικά κυκλώματα όπου συνήθως η σχεδίαση δεν μπορεί να γίνει αφαιρετικά με χρήση υψηλότερου επιπέδου δομικών στοιχείων αντίστοιχων με τις λογικές πύλες.

Αυτή η δυνατότητα αφαίρεσης στα ψηφιακά κυκλώματα σε συνδυασμό με πολλά μοντέλα επίλυσης των κάθε είδους προβλημάτων σχεδίασης έχουν βοηθήσει σημαντικά στην εκρηκτική διάδοση της ανάπτυξης και της χρήσης τους. Όπως θα δούμε όμως παρακάτω, δεν αρκεί μόνο η λογική ορθότητα ενός ψηφιακού κυκλώματος, αλλά αυτή χάνει την αξία της εφόσον δεν συνοδεύεται από παράλληλη ταχύτητα αλλά και χαμηλή κατανάλωση ισχύος. Για να ικανοποιηθούν όλες αυτές οι απαιτήσεις ταυτόχρονα θα πρέπει να αρχίσουμε να βλέπουμε ξανά τις πύλες ως συνδεδεμένα τρανζίστορ, όμως προτιμότερα μέσα από το πρίσμα συγκεκριμένης μεθοδολογίας η οποία επιτρέπει την μοντελοποίηση και άρα την αυτοματοποίηση διαδικασιών βελτιστοποίησης.

Αυτή η διατριβή, ασχολείται με το συγκεκριμένο κλάδο κυκλωμάτων, των ψηφιακών CMOS κυκλωμάτων (κυκλωμάτων που αποτελούνται από τρανζίστορ MOSFET), και αφορά στη βελτιστοποίησή τους ως προς την καθυστέρηση με αυτοματοποιημένο τρόπο μέσω λογισμικού. Πριν αρχίσει η παρουσίαση του μοντέλου του *logical effort* θα ακολουθήσει μια γενική εισαγωγή στα βήματα σχεδίασης ολοκληρωμένων κυκλωμάτων και στα εργαλεία αυτοματοποιημένης σχεδίασης (CAD Tools).

1.2. Σχεδίαση Ψηφιακών Ολοκληρωμένων Κυκλωμάτων

1.2.1. Βήματα Σχεδίασης

Βιβλιογραφία [3]. Σε γενικές γραμμές, η σχεδίαση ψηφιακών ολοκληρωμένων κυκλωμάτων, μπορεί να χωριστεί σε τρία μέρη:

1. Σχεδίαση **ESL**: Σε αυτό το βήμα, αποφασίζονται οι λειτουργικές προδιαγραφές. Ο σχεδιαστής μπορεί να χρησιμοποιήσει πλειάδα από γλώσσες και εργαλεία για να κατασκευάσει αυτή την περιγραφή. Παραδείγματα τέτοιων γλωσσών είναι οι SystemC, SystemVerilog Transaction Level Models, Simulink, κ.α.
2. Σχεδίαση **RTL**: Αυτό το βήμα μετατρέπει το αποτέλεσμα των προδιαγραφών του προηγούμενου βήματος (δηλαδή το τι θέλει ο σχεδιαστής να κάνει το κύκλωμά του), σε επίπεδο μεταφοράς - καταχωρητή (register transfer level). Η περιγραφή αυτή καθορίζει με μεγάλη λεπτομέρεια, επακριβώς, πως θα συμπεριφέρεται κάθε τμήμα του chip σε κάθε κύκλο ρολογιού.
3. **Φυσική** Σχεδίαση: Αυτό το βήμα σχεδίασης, χρησιμοποιεί τα αποτελέσματα της σχεδίασης RTL και μια βιβλιοθήκη από διαθέσιμες πύλες, και δημιουργεί την σχεδίαση του chip. Σε αυτό το βήμα καθορίζεται ποιες πύλες θα χρησιμοποιηθούν, η θέση των πυλών και η σύνδεση μεταξύ τους. Ας σημειωθεί πως το δεύτερο βήμα σχεδίασης (σχεδίαση RTL) είναι υπεύθυνο για την ορθή λειτουργία του chip. Η φυσική σχεδίαση δεν επηρεάζει καθόλου τη λειτουργικότητα (εάν γίνει σωστά) αλλά καθορίζει την ταχύτητα λειτουργίας του chip και το κόστος του.

1.2.2. Φυσική Σχεδίαση

Βιβλιογραφία [3]. Το βήμα σχεδίασης με το οποίο θα ασχοληθούμε είναι το τρίτο, δηλαδή το βήμα της **Φυσικής** Σχεδίασης. Η φυσική σχεδίαση περιλαμβάνει πλειάδα βημάτων τα οποία αναφέρουμε επιγραμματικά:

1. **Floorplanning**: Τα περισσότερα chip χωρίζονται σε περισσότερα από ένα εύχρηστα κομμάτια έτσι ώστε να μπορούν να μοιραστούν σε πολλούς διαφορετικούς σχεδιαστές, αλλά και για να μπορούν να αναλυθούν ως κομμάτια από εργαλεία αυτόματης σχεδίασης (CAD Tools). Κατά τη διαδικασία του floorplanning γίνεται εκτίμηση μήκους – καλωδίωσης, ενώ αποφασίζονται και οι στόχοι της φυσικής σχεδίασης.
2. **Λογική Σύνθεση**: Σε αυτό το στάδιο, το αποτέλεσμα της RTL σχεδίασης χαρτογραφείται σε ένα δίκτυο επιπέδου πυλών (netlist) της επιθυμητής τεχνολογίας του chip.
3. **Τοποθέτηση**: Οι πύλες στο δίκτυο πυλών, ανατίθενται σε μη επικαλυπτόμενες θέσεις στην περιοχή του καλουπιού.
4. **Λογική Βελτίωση / Βελτίωση τοποθέτησης**: Επαναληπτικοί λογικοί και μετασχηματισμοί τοποθέτησης για να ικανοποιηθούν οι περιορισμοί απόδοσης και ισχύος.
5. **Εισαγωγή Ρολογιού**: Ισοζυγισμένα δέντρα ρολογιού εισάγονται στη σχεδίαση
6. **Δρομολόγηση**: Εισάγονται τα καλώδια που συνδέουν τις πύλες στο δίκτυο πυλών.
7. **Βελτιστοποίηση μετά την Καλωδίωση**: Σε αυτό το στάδιο γίνεται προσπάθεια να ικανοποιηθούν οι εναπομείναντες ανικανοποίητοι περιορισμοί απόδοσης (χρονισμού), θορύβου (ακεραιότητα σήματος) και περιορισμοί σχετικοί με την διαδικασία κατασκευής.
8. **Σχεδίαση Κατασκευής**: Η σχεδίαση τροποποιείται όπου είναι αυτό δυνατόν έτσι ώστε να γίνει όσο το δυνατόν αποδοτικότερη η διαδικασία παραγωγής του.
9. **Τελικός Έλεγχος**: Επειδή τα λάθη είναι ακριβά, χρονοβόρα και δύσκολο να εντοπιστούν, ο εκτενής έλεγχος λαθών είναι κανόνας, επιβεβαιώνοντας πως η

χαρτογράφηση της λογικής έγινε σωστά, και πως ακολουθήθηκαν πιστά οι κανόνες κατασκευής.

10. **Η Σχεδίαση σε Τελική Μορφή**: Τα δεδομένα της σχεδίασης αντιγράφονται σε μέσα αποθήκευσης τα οποία αποστέλλονται στο χώρο κατασκευής για να αρχίσει η διαδικασία παραγωγής.

1.2.3. Απλοποιημένο Μοντέλο Σχεδίασης

Βιβλιογραφία [5]. Το Σχήμα 1.1 δείχνει μια πιο απλοποιημένη ροή σχεδίασης αναπαριστώντας τα στάδια της λογικής, κυκλωματικής και φυσικής σχεδίασης. Η σχεδίαση ξεκινά με το σύνολο προδιαγραφών, τυπικά σε μορφή κειμένου, καθορίζοντας τη λειτουργικότητα και τους στόχους απόδοσης του chip. Οι λογικοί σχεδιαστές γράφουν περιγραφές επιπέδου καταχωρητή (RTL) για κάθε κομμάτι, σε μια γλώσσα όπως είναι οι Verilog και VHDL και παράλληλα προσομοιώνουν αυτά τα μοντέλα μέχρι να πειστούν πως οι προδιαγραφές είναι σωστές. Βασιζόμενοι στην πολυπλοκότητα της RTL περιγραφής, οι σχεδιαστές εκτιμούν το μέγεθος για κάθε κομμάτι και κατασκευάζουν ένα floorplan, περιγράφοντας τη σχετική τοποθέτηση των κομματιών. Το floorplan επιτρέπει την εκτίμηση μήκους – καλωδίωσης και παρέχει στόχους για τη φυσική σχεδίαση.

Δεδομένης της RTL σχεδίασης και του floorplan, είναι μπορεί να αρχίσει η σχεδίαση του κυκλώματος. Υπάρχουν δύο γενικά στυλ σχεδίασης κυκλωμάτων: Η *αυτοματοποιημένη* και η *ειδική*: Η *ειδική* σχεδίαση απαιτεί περισσότερη ανθρώπινη εργασία με αντάλλαγμα την καλύτερη απόδοση. Σε μια ειδική μεθοδολογία, ο σχεδιαστής του κυκλώματος έχει την ευχέρεια να κατασκευάσει κελιά σε επίπεδο τρανζίστορ ή να επιλέξει από μια βιβλιοθήκη προκαθορισμένων κελιών. Ο σχεδιαστής πρέπει να πάρει πολλές αποφάσεις όσον αφορά στην οικογένεια πυλών που θα χρησιμοποιήσει, στην επιλογή της τοπολογίας που υλοποιεί με τον καλύτερο τρόπο τις συναρτήσεις που καθορίζονται στην RTL σχεδίαση, στην χρήση απλών ή πολύπλοκων πυλών κλπ. Τέλος, ο σχεδιαστής θα πρέπει να επιλέξει τα μεγέθη των τρανζίστορ της κάθε πύλης. Μια μεγαλύτερη πύλη οδηγεί το φορτίο της πιο γρήγορα, όμως επιδεικνύει μεγαλύτερη χωρητικότητα εισόδου στο προηγούμενο επίπεδο και καταλαμβάνει περισσότερη επιφάνεια ενώ καταναλώνει και μεγαλύτερη ισχύ. Όταν τα σχηματικά διαγράμματα ολοκληρωθούν, πρέπει να γίνουν λειτουργικοί και έλεγχοι χρονισμού για να επιβεβαιωθεί πως τα σχηματικά διαγράμματα υλοποιούν τις

προδιαγραφές της σχεδίασης RTL και τους περιορισμούς χρονισμού. Αν η απόδοση δεν είναι ικανοποιητική, ο σχεδιαστής θα προσπαθήσει να επαναπροσδιορίσει τα μεγέθη των πυλών για βελτιωμένη ταχύτητα, να αλλάξει σε οικογένεια πυλών μεγαλύτερης ταχύτητας, ή μπορεί να χρειαστεί να αλλάξει εντελώς την τοπολογία επιδιώκοντας παραλληλισμό ώστε να κατασκευαστούν ταχύτερες δομές με κόστος την επιπλέον επιφάνεια.

Η *αυτοματοποιημένη* σχεδίαση κυκλωμάτων χρησιμοποιεί εργαλεία σύνθεσης για να επιλέξει κυκλωματικές τοπολογίες και μεγέθη πυλών. Η αυτόματη σύνθεση απαιτεί πολύ λιγότερο χρόνο από την χειρωνακτική βελτιστοποίηση μονοπατιών και τη σχεδίαση σχηματικών διαγραμμάτων, όμως είναι γενικά περιορισμένη σε μια δεδομένη βιβλιοθήκη στατικών κελιών CMOS και παράγει αρκετά πιο αργά κυκλώματα από εκείνα που σχεδιάζονται από έναν έμπειρο μηχανικό. Από την άλλη, η εξέλιξη στις τεχνολογίες της σύνθεσης και της κατασκευής συνεχίζει να επεκτείνει το σύνολο των προβλημάτων που η σύνθεση μπορεί να λύσει ικανοποιητικά, αλλά για το προβλεπόμενο μέλλον, τα σχέδια αιχμής θα απαιτούν τουλάχιστον μερικά ειδικά κυκλώματα. Τα αυτόματα συντεθειμένα κυκλώματα είναι φυσιολογικά λογικώς ορθά εκ' κατασκευής, όμως εξακολουθεί να είναι απαραίτητη η επαλήθευση χρονισμού. Εάν η απόδοση δεν είναι επαρκής, ο σχεδιαστής του κυκλώματος μπορεί να δώσει κατευθυντήριες γραμμές για το εργαλείο σύνθεσης ώστε να βελτιώσει τα κρίσιμα μονοπάτια (τα πιο αργά μονοπάτια).

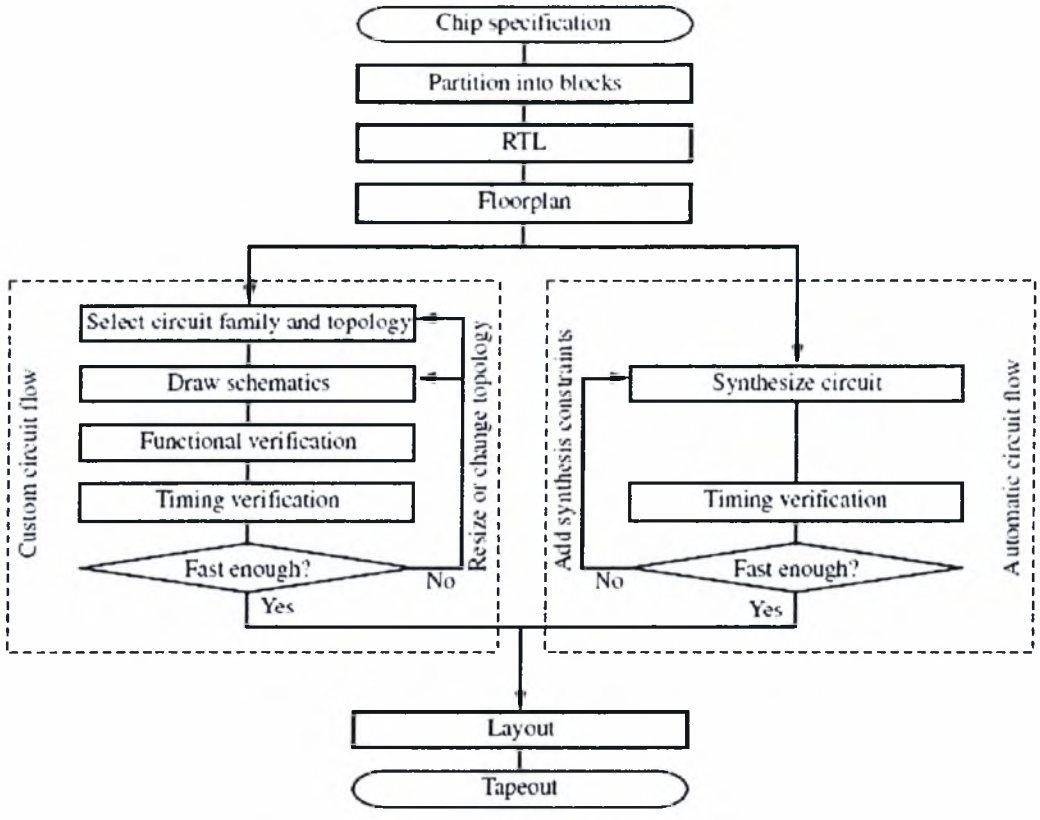


Figure 1.1 — Simplified chip design flow.

1.3. Αυτοματοποιημένη Σχεδίαση - Εργαλεία CAD

Βιβλιογραφία [4]. Ο όρος CAD κυριολεκτικά σημαίνει *Σχεδίαση Βοηθούμενη από τον Υπολογιστή*. Όταν αναφερόμαστε στη χρήση εργαλείων CAD για τη σχεδίαση ολοκληρωμένων κυκλωμάτων, τότε εκτός από τον όρο CAD χρησιμοποιούμε συχνά και τον όρο EDA ή αλλιώς *Αυτοματοποίηση Ηλεκτρονικού Σχεδιασμού*. Σε αυτή την κατηγορία, ανήκουν εργαλεία που χρησιμοποιούνται για το σχεδιασμό και την κατασκευή ολοκληρωμένων κυκλωμάτων.

Η σημαντικότητα των εργαλείων CAD έχει αυξηθεί αστραπιαία, ταυτόχρονα με την συνεχή κλιμάκωση της τεχνολογίας των ημιαγωγών. Πριν τη δημιουργία των πρώτων εργαλείων CAD, τα ολοκληρωμένα κυκλώματα σχεδιάζονταν τόσο ως προς τη λειτουργικότητα, όσο και ως προς την τοπολογική διάταξή τους με το χέρι. Στα μέσα της δεκαετίας του 70, οι προγραμματιστές άρχιζαν να αυτοματοποιούν το σχεδιασμό. Τότε αναπτύχθηκαν τα πρώτα εργαλεία τοποθέτησης και δρομολόγησης (place and route). Η επόμενη εποχή ξεκίνησε με τη δημοσίευση του «Εισαγωγή στα συστήματα VLSI» των *Carver Mead* και *Lynn Conway* το 1980. Αυτό το επαναστατικό σύγγραμμα συνηγόρησε στον σχεδιασμό chip μέσω γλωσσών προγραμματισμού των οποίων τα

προγράμματα έβγαζαν τελικά σαν έξοδο πραγματικό πυρίτιο. Το άμεσο αποτέλεσμα ήταν η κατά παράγοντες του 100 αύξηση στην πολυπλοκότητα των chip που μπορούσαν να σχεδιαστούν, με βελτιωμένη πρόσβαση στα εργαλεία επαλήθευσης σχεδίασης τα οποία χρησιμοποιούσαν λογική προσομοίωση. Όπως είπαμε και παραπάνω, ο αριθμός των τρανζίστορ που περιέχει ένα ολοκληρωμένο κύκλωμα σήμερα μπορεί να είναι περισσότερο από απαγορευτικά μεγάλος για να γίνει ολόκληρη η σχεδίαση χειρωνακτικά. Έτσι, οι σημερινές διαδικασίες σχεδίασης είναι εξαιρετικά παραμετροποιήσιμες. Τα εμπρόσθια μέρη παράγουν τυποποιημένες περιγραφές σχεδίασης οι οποίες μεταγλωττίζονται σε αφαιρετικά κελιά χωρίς εξάρτηση από την τεχνολογία κελιών. Τα κελιά υλοποιούν λογικές συναρτήσεις χρησιμοποιώντας μια συγκεκριμένη τεχνολογία ολοκληρωμένων κυκλωμάτων. Οι κατασκευαστές γενικά παρέχουν βιβλιοθήκες από στοιχεία για τη δική τους διαδικασία κατασκευής, τα οποία υποστηρίζουν μοντέλα προσομοίωσης που ταιριάζουν με τα πρότυπα εργαλεία προσομοίωσης.

Σήμερα υπάρχουν εργαλεία CAD για κάθε στάδιο σχεδίασης ενός ολοκληρωμένου κυκλώματος, από τη σχεδίαση ESL μέχρι τη φυσική σχεδίαση. Εμείς όπως έχουμε αναφέρει και νωρίτερα θα ασχοληθούμε με τη φυσική σχεδίαση, και ειδικότερα στο βήμα που εμπλέκεται ο υπολογισμός των μεγεθών των πυλών του εκάστοτε κυκλώματος ή όμοια ο υπολογισμός των μεγεθών των τρανζίστορ της κάθε πύλης (sizing). Ακολουθεί μια εισαγωγική ενότητα που αφορά στο μοντέλο και τη μέθοδο του *logical effort* τα οποία χρησιμοποίησα για να υλοποιήσω το λογισμικό βελτιστοποίησης.

Κεφάλαιο 2 - Η Μέθοδος του Logical Effort

Βιβλιογραφία [5],[6],[7]. Ο όρος *logical effort* επινοήθηκε από τους *Ivan Sutherland* και *Robert Sproull* το 1991. Η μέθοδος του logical effort είναι μια άμεση τεχνική που χρησιμοποιείται για την εκτίμηση της καθυστέρησης σε ένα κύκλωμα CMOS. Η κυκλωματική τοπολογία καθώς και η ανάθεση μεγεθών στις πύλες είναι οι κύριοι τομείς στους οποίους γίνεται χρήση της μεθόδου.

Η μέθοδος στηρίζεται πάνω σε ένα απλό μοντέλο για την καθυστέρηση διαμέσου μιας λογικής πύλης MOS. Το μοντέλο περιγράφει τις καθυστερήσεις που προκαλούνται από τη χωρητικότητα φορτίου που η πύλη οδηγεί, καθώς και από την τοπολογία της ίδιας της πύλης.

Η μοντελοποίηση αυτή της καθυστέρησης έχει πολλές εφαρμογές στη σχεδίαση ολοκληρωμένων κυκλωμάτων. Απαντά μεταξύ άλλων στα εξής ερωτήματα:

1. Δεδομένης μιας λογικής συνάρτησης, τι τύπου πύλες θα έπρεπε να χρησιμοποιηθούν;
2. Πως θα έπρεπε ένα συνδυαστικό λογικό δίκτυο να σχεδιαστεί ώστε να έχει την ελάχιστη καθυστέρηση;
3. Πόσα είναι τα σωστά επίπεδα λογικής για ένα κύκλωμα;
4. Ποιο είναι το ελάχιστο μέγεθος τρανζίστορ που απαιτείται για να οδηγήσει μια δεδομένη χωρητικότητα φορτίου;

2.1. Το Μοντέλο Καθυστέρησης

Η καθυστέρηση της κάθε πύλης εκφράζεται σε σχέση με μια βασική μονάδα καθυστέρησης, την $\tau = 3RC$, η οποία είναι η καθυστέρηση ενός αντιστροφέα που οδηγεί έναν ακριβώς ίδιο αντιστροφέα χωρίς καθόλου παρασιτική χωρητικότητα. Η απόλυτη καθυστέρηση ορίζεται απλά ως το γινόμενο της καθυστέρησης της πύλης d και του τ.

$$d_{abs} = d\tau$$

Επομένως η μονάδα καθυστέρησης μετράται σε σχέση με το τ. Στην τυπική τεχνολογία των 0.6μ το τ είναι περίπου ίσο με 50 ps. Για να υπολογίσουμε την d μιας πύλης τώρα, εκμεταλλευόμαστε το ότι μπορεί αυτή να εκφραστεί ως το άθροισμα

δύο βασικών όρων: της *παρασιτικής καθυστέρησης* (parasitic delay) *p* και της *προσπάθειας βαθμίδας* (stage effort) *f*. Επομένως,

$$d = f + p$$

Όμως, το stage effort μπορεί να διασπαστεί περαιτέρω στο γινόμενο δύο άλλων όρων: στο *logical effort g*, το οποίο περιγράφει τις εγγενείς ιδιότητες της κάθε πύλης, και στην *ηλεκτρική προσπάθεια* (electrical effort) *h*, η οποία περιγράφει το φορτίο. Το stage effort επομένως θα είναι:

$$f = gh$$

ενώ το electrical effort μπορεί να υπολογιστεί ως:

$$h = \frac{C_{out}}{C_{in}}$$

, όπου το C_{in} είναι η χωρητικότητα εισόδου της λογικής πύλης, και το C_{out} είναι η χωρητικότητα του εξωτερικού φορτίου που οδηγείται από την λογική πύλη. Συνδυάζοντας αυτές τις εξισώσεις μπορούμε να δούμε πως έχουμε μια εξίσωση για την καθυστέρηση μιας λογικής πύλης σε μονάδες του *τ*. Αυτή είναι η:

$$d = gh + p$$

Ποιοτικά, η τιμή του logical effort μιας πύλης, μας λέει πόσο χειρότερη είναι στο να παράγει ρεύμα εξόδου σε σχέση με έναν αντιστροφέα, δεδομένου ότι κάθε μια από τις εισόδους της παρουσιάζει την ίδια χωρητικότητα εισόδου με αυτή του αντιστροφέα. Ο υπολογισμός του *g* (logical effort) μιας πύλης είναι γενικά πολύπλοκος και δεν θα ασχοληθούμε περισσότερο με αυτόν. Θα θεωρήσουμε την τιμή του γνωστή για κάθε είδος πύλης καθώς είναι δυνατόν να υπολογιστεί. Για τις πιο ευρέως χρησιμοποιούμενες πύλες, οι τιμές του logical effort εξάγονται από πολύ απλούς τύπους, οι οποίοι για κάθε είδος πύλης εξαρτώνται μόνο από τον αριθμό των εισόδων της. Για παράδειγμα, για μια NAND *n* εισόδων, η τιμή του logical effort θα είναι *(n+2)/3*. Ο Πίνακας 2.1 μας δίνει τιμές του logical effort για τους αντιστροφείς, τις πύλες NAND, τις πύλες NOR, τους πολυπλέκτες και τις πύλες XOR.

	Αριθμός Εισόδων					
Πύλη	1	2	3	4	5	n
Αντιστροφέας	1					
NAND		4/3	5/3	6/3	7/3	(n+2)/3
NOR		5/3	7/3	9/3	11/3	(2n+1)/3
Πολυπλέκτης		2	2	2	2	2
XOR		4	12	32		

Πίνακας 2.1:

Logical effort για τους διάφορους τύπους πυλών

Για να υπολογίσουμε την τιμή της καθυστέρησης μιας πύλης και κατά συνέπεια ενός μονοπατιού, πλέον απομένει να γνωρίζουμε για κάθε πύλη την τιμή της παρασιτικής καθυστέρησης. Αυτό που απλοποιεί τη διαδικασία, είναι ότι η παρασιτική καθυστέρηση είναι σταθερή για κάθε πύλη, όπως είναι και το logical effort της κάθε πύλης, ανεξαρτήτως του πλάτους των τρανζίστορ που αυτή περιέχει. Εξαρτάται αποκλειστικά από τη λογική συνάρτηση που υλοποιεί η πύλη. Επειδή η παρασιτική καθυστέρηση προστίθεται σαν σταθερά στην συνολική καθυστέρηση της πύλης και άρα και του κάθε μονοπατιού, για δεδομένο κύκλωμα, ο μόνος τρόπος να μειώσουμε την καθυστέρηση του κρίσιμου μονοπατιού (του μονοπατιού του κυκλώματος με τη μεγαλύτερη καθυστέρηση) μέσω της μείωσης της παρασιτικής καθυστέρησης θα ήταν να αλλάξουμε την τοπολογία του κυκλώματος χρησιμοποιώντας πύλες με μικρότερη παρασιτική καθυστέρηση, αλλά διατηρώντας τη συνάρτηση που υλοποιεί το κύκλωμα σταθερή, δηλαδή μέσω μιας ισοδύναμης τοπολογίας με μικρότερη συνολική παρασιτική καθυστέρηση. Εμείς θα θεωρήσουμε πως η τοπολογία του κυκλώματος που παραλαμβάνουμε είναι η βέλτιστη οπότε δεν θα ασχοληθούμε με αλλαγές τοπολογίας. Παρ' όλ' αυτά, θα χρησιμοποιήσουμε στους υπολογισμούς των καθυστερήσεων και την παρασιτική καθυστέρηση, διότι εκτός του ότι αυτή μας δίνει μια εικόνα για την πραγματική καθυστέρηση ενός μονοπατιού σε πραγματικό χρόνο και όχι σχετικό, η παρασιτική καθυστέρηση είναι απαραίτητη στο να είμαστε βέβαιοι για το ποιο είναι το πραγματικά κρίσιμο μονοπάτι. Ο Πίνακας 2.2 που ακολουθεί, μας δίνει τις παρασιτικές καθυστερήσεις του ανστιστροφέα, του πολυπλέκτη και των πυλών NAND, NOR, XOR, XNOR. Όλες δίνονται ως πολλαπλάσια της παρασιτικής καθυστέρησης του αντιστροφέα, η οποία εξαρτάται από την τεχνολογία που χρησιμοποιείται.

Πύλη	*Παρασιτική Καθυστέρηση*
Αντιστροφέας	P_{inv}
Πύλη NAND *n*-εισόδων	$n\ P_{inv}$
Πύλη NOR *n*-εισόδων	$n\ P_{inv}$
Πολυπλέκτης *n*-διαδρομών	$2\ n\ P_{inv}$
XOR, XNOR	$4\ P_{inv}$

Πίνακας 2.2:

Παρασιτικές καθυστερήσεις για τους διάφορους τύπους πυλών

Έτσι, με δεδομένο ένα κύκλωμα, μπορούμε πια να υπολογίσουμε την καθυστέρηση της κάθε πύλης του, και άρα αθροίζοντας αυτές τις καθυστερήσεις των πυλών ενός μονοπατιού, μπορούμε να εξάγουμε την συνολική καθυστέρησή του. Με λίγα λόγια, μέσα από μεθόδους εύρεσης του πιο κρίσιμου μονοπατιού όπως είναι η μέθοδος *PERT*, μπορούμε να επικεντρώσουμε την προσοχή μας στη μείωση της καθυστέρησης σε αυτό το συγκεκριμένο κρίσιμο μονοπάτι, αυξάνοντας έτσι την συχνότητα λειτουργίας ολόκληρου του κυκλώματος.

2.2. Η Μέθοδος Βελτιστοποίησης Μονοπατιού

Η μέθοδος του logical effort αποκαλύπτει τον βέλτιστο αριθμό επιπέδων για ένα πολυεπίπεδο δίκτυο και πώς να αποκομίσουμε την ελάχιστη συνολική καθυστέρηση ζυγίζοντας την καθυστέρηση μεταξύ των επιπέδων. Οι έννοιες του logical effort γενικεύονται εύκολα από ξεχωριστές πύλες σε πολυεπίπεδα μονοπάτια.

Το *logical effort* ενός *μονοπατιού* σχηματίζεται πολλαπλασιάζοντας τα logical efforts όλων των πυλών κατά μήκος του μονοπατιού. Χρησιμοποιούμε το κεφαλαίο σύμβολο *G* για να δηλώσουμε το logical effort ενός μονοπατιού, ώστε να το ξεχωρίζουμε από το *g*, το logical effort μιας πύλης μέσα στο μονοπάτι. Έτσι έχουμε:

$$G = \prod g_i$$

Το *electrical effort* για ένα *μονοπάτι* θα είναι ο λόγος των χωρητικοτήτων εξόδου προς την χωρητικότητα εισόδου του μονοπατιού, δηλαδή η χωρητικότητα εξόδου της τελευταίας πύλης του μονοπατιού προς την χωρητικότητα εισόδου της πρώτης πύλης. Χρησιμοποιούμε το σύμβολο *H*. Έτσι θα έχουμε:

$$H = \frac{C_{out}}{C_{in}}$$

Τώρα πρέπει να εισάγουμε μια καινούρια έννοια που ονομάζεται *branching effort*, ή *προσπάθεια διακλάδωσης*, για να είμαστε σε θέση να εκτιμήσουμε το βαθμό διακλάδωσης της εξόδου μιας πύλης σε εισόδους άλλων πυλών. Μέχρι τώρα χρησιμοποιούσαμε την έννοια του electrical effort για το σκοπό αυτό, όμως σε επίπεδο μονοπατιού δεν αρκεί να χρησιμοποιήσουμε τη μεταβλητή *H* στη θέση του electrical effort διότι σε κάθε επίπεδο, ένα μέρος μεν του διαθέσιμου ρεύματος οδήγησης κατευθύνεται κατά μήκος του μονοπατιού, όμως το υπόλοιπο κατευθύνεται εκτός αυτού λόγω των διακλαδώσεων. Ορίζουμε το branching effort *b* στην έξοδο μιας λογικής πύλης ως:

$$b = \frac{C_{on-path} + C_{off-path}}{C_{on-path}} = \frac{C_{total}}{C_{useful}}$$

Δηλαδή το ορίζουμε ως τον λόγο της χωρητικότητας εξόδου προς την χωρητικότητα της προπορευόμενης πύλης στο μονοπάτι. Με άλλα λόγια, ως τον λόγο της συνολικής χωρητικότητας εντός και εκτός μονοπατιού, προς την χωρητικότητα εντός του μονοπατιού. Σε επίπεδο μονοπατιού τώρα, το branching effort θα είναι το γινόμενο όλων των branching effort κάθε επιπέδου στο μονοπάτι:

$$B = \prod b_i$$

Τώρα είμαστε έτοιμοι να εξάγουμε μια νέα έννοια που αφορά σε ένα μονοπάτι, που είναι απόρροια των εννοιών που εισάγαμε μέχρι τώρα. Αυτή είναι το *path effort* ή *προσπάθεια μονοπατιού*, *F*. Είναι το ανάλογο του *stage effort*, *f* μιας *πύλης*, σε επίπεδο μονοπατιού. Προκύπτει από το γινόμενο των logical, branching και electrical effort του μονοπατιού:

$$F = GBH$$

Παρότι το path effort δεν είναι ένα άμεσο μέτρο της καθυστέρησης του μονοπατιού, ωστόσο σε αυτό βρίσκεται το κλειδί για την ελαχιστοποίηση της καθυστέρησης του. Η *καθυστέρηση* του *μονοπατιού*, *D* είναι το άθροισμα των καθυστερήσεων σε κάθε επίπεδο λογικής στο μονοπάτι. Όπως και στην έκφραση της καθυστέρησης μίας μόνο πύλης, έτσι και σε επίπεδο μονοπατιού, θα διαχωρίσουμε την *effort delay μονοπατιού*, D_F από την *parasitic delay μονοπατιού*, *P*.

$$D = \sum d_i = D_F + P$$

όπου,

$$D_F = \sum g_i h_i$$

και

$$P = \sum p_i$$

Όπως είναι φανερό, για να ελαχιστοποιήσουμε την καθυστέρηση ενός μονοπατιού *N* επιπέδων, το μόνο που μπορούμε να κάνουμε είναι να μειώσουμε την D_F καθώς η παρασιτική καθυστέρηση για δεδομένο τύπο πυλών παραμένει σταθερή. Η ελαχιστοποίηση αυτή θα στηριχθεί σε μια αρχή την οποία θα δεχτούμε χωρίς απόδειξη: *Η καθυστέρηση ενός μονοπατιού είναι ελάχιστη, όταν κάθε επίπεδο του μονοπατιού έχει το ίδιο stage effort*. Επίσης, αυτή η ελάχιστη καθυστέρηση επιτυγχάνεται όταν το stage effort είναι ίσο με:

$$\hat{f} = g_i h_i = F^{1/N}$$

όπου *F* είναι το path effort του μονοπατιού. Συνδυάζοντας τις παραπάνω εξισώσεις έχουμε στη διάθεσή μας το κυρίαρχο αποτέλεσμα της μεθόδου του logical effort, το οποίο είναι μια έκφραση για την ελάχιστη καθυστέρηση που μπορεί να επιτευχθεί κατά μήκος ενός μονοπατιού:

$$\hat{D} = NF^{1/N} + P$$

Τώρα μένει να δούμε πως θα εφαρμόσουμε την παραπάνω αρχή, ώστε να έχουμε ελάχιστη καθυστέρηση στο μονοπάτι. Για να κάνουμε ίσο το stage effort σε κάθε επίπεδο του μονοπατιού πρέπει να επιλέξουμε κατάλληλα μεγέθη για τα τρανζίστορ σε κάθε επίπεδο λογικής κατά μήκος του μονοπατιού. Από την παραπάνω εξίσωση η οποία μας έδωσε το stage effort που αντιστοιχεί στην ελάχιστη καθυστέρηση, λύνοντας ως προς το electrical effort έχουμε την εξίσωση:

$$\hat{h}_i = \frac{F^{1/N}}{g_i}$$

Από αυτή τη σχέση, μπορούμε να καθορίσουμε τα μεγέθη των τρανζίστορ κατά μήκος του μονοπατιού. Αρχίζοντας από το τέλος του και προχωρώντας προς τα πίσω

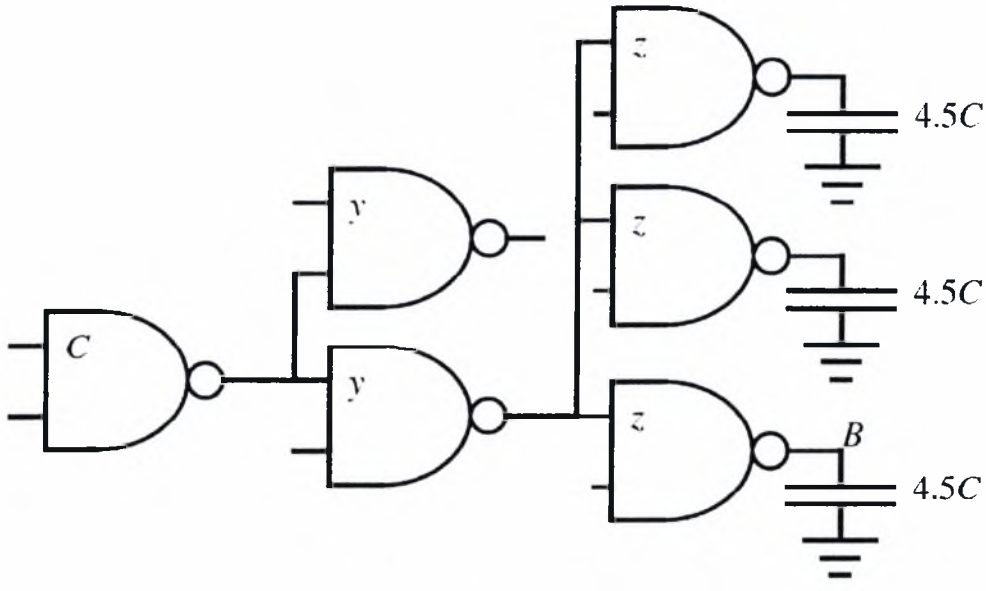
κατά μήκος του μονοπατιού, εφαρμόζουμε τον παρακάτω μετασχηματισμό χωρητικότητας:

$$C_{in_i} = \frac{g_i C_{out_i}}{f}$$

Ένας περιορισμός που πρέπει να ισχύει μετά την εφαρμογή της ελαχιστοποίησης, είναι ότι το *Cin* του μονοπατιού, δηλαδή η χωρητικότητα εισόδου της πρώτης πύλης του μονοπατιού πρέπει να είναι η ίδια με αυτήν πριν να γίνει η εφαρμογή. Αυτό είναι σημαντικό, διότι ένα δίκτυο λογικής, χαρακτηρίζεται από έναν αριθμό προδιαγραφών στις οποίες ανήκουν και οι χωρητικότητες των κόμβων εισόδου, άρα πρέπει να παραμένουν σταθερές μετά την εφαρμογή οποιουδήποτε αλγορίθμου βελτιστοποίησης. Για να ισχύει λοιπόν αυτός ο περιορισμός αποδεικνύεται πως πρέπει κατά τη διάρκεια της εφαρμογής του μετασχηματισμού βελτιστοποίησης, να παραμένουν τα branching efforts κάθε πύλης που ανήκει στο μονοπάτι σταθερά. Αυτό συνεπάγεται, πως η αλλαγή του *Cin* σε μια πύλη του μονοπατιού, συνεπάγεται και την κατά τον ίδιο παράγοντα αλλαγή της χωρητικότητας εισόδου όλων των πυλών που βρίσκονται στην ίδια διακλάδωση με την πύλη αυτή.

2.2.1. Ένα Απλό Παράδειγμα

Ακολουθεί ένα παράδειγμα για να γίνουν κατανοητά όλα τα παραπάνω όπου χάριν απλότητας κάθε πύλη είναι του ίδιου τύπου (NAND):



Σχήμα 2.1

Στο Σχήμα 2.1 βλέπουμε πως το *H* είναι 4.5 διότι τα *C* απαλείφονται. Το logical effort της πύλης NAND δύο εισόδων είναι 4/3, άρα το logical effort *G* του μονοπατιού θα είναι:

$$G = (4/3)^3$$

Το branching effort στην έξοδο του πρώτου επιπέδου, θα είναι *(y+y)/y* = 2, και όμοια, στην έξοδο του δευτέρου επιπέδου θα είναι *3z/z* = 3. Επομένως το branching effort του μονοπατιού (δεν μας ενδιαφέρει ποιο ακριβώς είναι το μονοπάτι διότι όλα τα μονοπάτια είναι ίδια λόγω συμμετρίας) θα είναι *B* = 2 x 3 = 6. Τώρα μπορούμε να υπολογίσουμε το path effort *F* το οποίο θα είναι *F* = *GBH* = 64. Επίσης, θα είναι:

$$\hat{D} = 3(64)^{1/3} + 3(2p_{inv}) = 18.0$$

Για να έχουμε ελάχιστη καθυστέρηση, πρέπει να εξισώσουμε σε κάθε επίπεδο το stage effort. Από τη στιγμή που το path effort είναι 64 και έχουμε 3 επίπεδα, το ιδανικό stage effort θα είναι 4. Αρχίζοντας από την έξοδο θα είναι, *z* = 4.5*C* x (4/3)/4 = 1.5*C*. Όμοια, το δεύτερο επίπεδο οδηγεί τρία αντίγραφα του τρίτου επιπέδου, άρα θα είναι, *y* = 3*z* x (4/3)/4 = *z* = 1.5*C*. Όπως μπορούμε να επαληθεύσουμε, το *Cin*

του μονοπατιού, δηλαδή της πρώτης πύλης, θα είναι ίσο με $2y \times (4/3)/4 = (2/3)y = C$, όπως ήταν στις προδιαγραφές του κυκλώματος.

Κεφάλαιο 3 - Το Λογισμικό Βελτιστοποίησης

Σε αυτό το κομμάτι θα ασχοληθούμε με το λογισμικό βελτιστοποίησης. Σκοπός του λογισμικού είναι η βελτίωση ενός δεδομένου κυκλώματος CMOS ως προς την καθυστέρηση, διατηρώντας την τοπολογία του και τις προδιαγραφές χωρητικότητας εισόδου για κάθε είσοδό του, αμετάβλητες. Η μέθοδος βελτιστοποίησης έχει σαν βάση τη μέθοδο του logical effort, ενώ για να βρεθούν τα *N* χειρότερα μονοπάτια χρησιμοποιείται ο Αλγόριθμος 7.1 από τη Βιβλιογραφία [10]. Το κύκλωμα εισόδου πρέπει να βρίσκεται σε μορφή αρχείων SPICE ενώ θεωρούμε πως η τοπολογία του είναι βέλτιστη καθώς το λογισμικό δεν «ψάχνει» για τοπολογική βελτιστοποίηση. Η έξοδος του προγράμματος είναι σε μορφή αρχείου HTML.

3.1. Τα Βήματα του Αλγορίθμου του Λογισμικού

Το πρόγραμμα δέχεται σαν παράμετρο το όνομα του αρχείου SPICE το οποίο περιγράφει το προς επεξεργασία κύκλωμα. Η διαδικασία βελτιστοποίησης χωρίζεται σε 7 βασικά βήματα:

1. Ανάγνωση από το Αρχείο SPICE
2. Αναπαράσταση του Κυκλώματος σε Δομή Δεδομένων
3. Κατηγοριοποίηση των Πυλών σε Επίπεδα Λογικής
4. Αλγόριθμος Βελτιστοποίησης
5. Αναίρεση Τελευταίων Αλλαγών
6. Κβάντιση των Μεγεθών των Τρανζίστορ
7. Εκτύπωση

Όπως θα φανεί και παρακάτω, το τέταρτο βήμα είναι πιο πολύπλοκο από τα υπόλοιπα, γι αυτό το λόγο στη συνέχεια το αναλύουμε σε περισσότερες φάσεις καθώς σε αυτό γίνεται εκτέλεση πλειάδας ανεξάρτητων μεταξύ τους αλγορίθμων κατά τη διαδικασία βελτιστοποίησης.

3.1.1. Ανάγνωση από το Αρχείο SPICE

Βιβλιογραφία [8]. Το πρώτο βήμα είναι η ανάγνωση από το αρχείο εισόδου, το οποίο περιέχει την περιγραφή του κυκλώματος σε μορφή SPICE. Το αρχείο πρέπει να περιγράφει το κύκλωμα σε επίπεδο πυλών, όχι τρανζίστορ. Για να γίνει κάτι τέτοιο, ένα αρχείο SPICE υποστηρίζει τον ορισμό υψηλότερων δομών όπως είναι οι πύλες μέσω της λέξης κλειδιού *.subckt* που αντιστοιχεί στην έννοια του υποκυκλώματος. Έτσι ορίζοντας όλα τα είδη πυλών που περιέχει το προς περιγραφή κύκλωμα ως υποκυκλώματα, αυτές μπορούν να χρησιμοποιηθούν επανειλημμένα και να θεωρηθεί πως το κύκλωμα αποτελείται από πύλες. Μέσα σε ένα block *.subckt* γίνεται η τοπολογική περιγραφή μιας πύλης ενώ καθορίζονται και τα μεγέθη των τρανζίστορ από τα οποία αποτελείται η πύλη. Από αυτά τα blocks, εμείς διαβάζουμε τα χρησιμοποιούμενα είδη πυλών καθώς και τα αρχικά μεγέθη χωρητικοτήτων εισόδου της κάθε πύλης. Ένα παράδειγμα block *.subckt* που περιγράφει την πύλη NOT είναι το ακόλουθο,

```
.subckt not1 2 1 9999
m1 2 1 9999 9999 modp l=2u w=8u as=8p ad=8p ps=12u pd=12u
m2 2 1 0 0 modn l=2u w=8u as=16p ad=16p ps=16u pd=16u
.ends not1
```

όπου τα πλάτη *w* περιγράφουν τα πλάτη των *p* και *n* τύπου τρανζίστορ της πύλης αντίστοιχα.

Στην αρχή του αρχείου SPICE, κωδικοποιούνται οι είσοδοι του κυκλώματος. Κάθε είσοδος αντιστοιχίζεται σε έναν αριθμητικό κωδικό και η περιγραφή της αρχίζει με ένα όνομα που αρχίζει με το γράμμα *v* ακολουθούμενο από τον αριθμητικό κωδικό ο οποίος θα χρησιμοποιηθεί στην περιγραφή της τοπολογίας του υπόλοιπου κυκλώματος. Παρακάτω βλέπουμε την περιγραφή ενός κυκλώματος με 7 εισόδους.

```
v1 1 0 pulse(0 5 15n)
v2 2 0 pulse(0 5 15n)
v3 3 0 pulse(0 5 15n)
v6 6 0 pulse(0 5 15n)
v7 7 0 pulse(0 5 15n)
```

Τα *vi* είναι τα ονόματα που υποδηλώνουν εισόδους, ενώ στην επόμενη στήλη, οι αριθμοί από το 1 ως το 7 ορίζουν τον αριθμητικό κωδικό της κάθε μιας εισόδου.

Η κυρίως περιγραφή του κυκλώματος βρίσκεται στο τελευταίο τμήμα του αρχείου SPICE, όταν έχουν οριστεί όλα τα στοιχεία που το απαρτίζουν. Εκεί, σε κάθε γραμμή ορίζεται μια συγκεκριμένη πύλη του κυκλώματος, ενώ η έξοδος και κάθε είσοδός της κωδικοποιούνται με ένα όνομα κόμβου. Αν ο κωδικός κόμβου εξόδου μιας πύλης

συμπίπτει με τον κωδικό ενός κόμβου εισόδου μιας άλλης, αυτό σημαίνει πως η δεύτερη είναι συνδεδεμένη σε μια είσοδό της με την έξοδο της πρώτης πύλης. Οι γραμμές στο αρχείο SPICE που αντιστοιχούν στην περιγραφή των πυλών του κυκλώματος αρχίζουν με το γράμμα *x*. Παρακάτω βλέπουμε ένα τέτοιο τμήμα του αρχείου:

```
x10 10 1 3 9999 nand2
x11 11 3 6 9999 nand2
x16 16 2 11 9999 nand2
x19 19 11 7 9999 nand2
x22 22 10 16 9999 nand2
x23 23 16 19 9999 nand2
```

Στην πρώτη γραμμή βλέπουμε πως ορίζεται μια πύλη nand2 (πύλη NAND δύο εισόδων) της οποίας η έξοδος είναι ο κόμβος 10 ενώ δέχεται ως εισόδους τους κόμβους 1 και 3 όπου όπως είδαμε στην προηγούμενη παράγραφο αποτελούν εισόδους του κυκλώματος.

Τέλος, επειδή στις προδιαγραφές του κυκλώματος ανήκουν και οι χωρητικότητες εξόδου του, για να μπορέσουμε να κάνουμε βελτιστοποίηση διαβάζουμε στο τέλος του αρχείου γραμμές με πρώτο γράμμα το *c* οι οποίες θέτουν πρόσθετες χωρητικότητες στους κόμβους που δηλώνουν. Αυτή η περιγραφή, εκτός από τις χωρητικότητες εξόδου του κάθε κυκλώματος, μπορεί να περιγράψει και τις χωρητικότητες διασύνδεσης μεταξύ των πυλών οι οποίες και στις δύο περιπτώσεις προστίθενται στη συνολική χωρητικότητα του κάθε κόμβου. Η σύνταξη είναι η ακόλουθη:

```
c1 22 0 3.0u
c2 23 0 3.0u
```

Εδώ, θέτουμε στους κόμβους 22 και 23 (που στο συγκεκριμένο κύκλωμα είναι κόμβοι εξόδου) χωρητικότητες μέτρου 3.0uF.

3.1.2. Αναπαράσταση του Κυκλώματος σε Δομή Δεδομένων

Κατά την ανάγνωση του αρχείου SPICE δημιουργείται ταυτόχρονα μια δομή δεδομένων που αναπαριστά το κύκλωμα. Πρώτα δημιουργούνται οι κόμβοι εισόδου, οι οποίοι τοποθετούνται σε μία αρχική λίστα. Ταυτόχρονα χρησιμοποιείται και μια δομή δυαδικού δένδρου αναζήτησης στο οποίο αποθηκεύονται όλοι οι κόμβοι με κλειδί αναζήτησης το όνομά τους. Ήδη έχουμε τοποθετήσει στο δένδρο αυτό τους κόμβους εισόδου του κυκλώματος.

Στη συνέχεια γίνεται ανάγνωση των blocks *.subckt* ώστε να δημιουργηθεί μια δομή από γεννήτορες της κάθε πύλης από τους οποίους θα δημιουργούνται αντίγραφα σε κάθε εμφάνιση μιας πύλης κατά την ανάγνωση του κυρίως κυκλώματος στο τρίτο μέρος του αρχείου SPICE.

Στο τρίτο μέρος, κάθε γραμμή αναπαριστά όπως είπαμε μια πύλη του κυκλώματος. Στην γραμμή αυτή περιγράφεται ο τύπος της πύλης, ο κόμβος εξόδου της και οι κόμβοι εισόδου της. Είπαμε παραπάνω, ότι κάθε κόμβος μπορεί να εμφανιστεί σε περισσότερες της μιας γραμμές καθώς σε μια γραμμή μπορεί να παίζει το ρόλο της εξόδου μιας πύλης, ενώ σε πολλές γραμμές μπορεί να παίζει το ρόλο της εισόδου πυλών. Με την ανάγνωση του κάθε κόμβου, γίνεται αναζήτηση στο δυαδικό δένδρο, και εφόσον δεν βρίσκεται ήδη ο κόμβος στο δένδρο (δηλαδή εφόσον είναι η πρώτη εμφάνισή του στην περιγραφή του κυκλώματος), δημιουργείται και τοποθετείται με βάση το όνομα – κλειδί του. Στον κόμβο εξόδου της περιγραφής, θέτουμε την αντίστοιχη πύλη ως είσοδό του, ενώ στους κόμβους εισόδου προσθέτουμε την πύλη στις εξόδους του κόμβου, οι οποίες είναι πολλαπλές λόγω της δυνατότητας διακλαδώσεων. Οι συνδέσεις είναι διπλής κατεύθυνσης, δηλαδή υπάρχει πρόσβαση και από τον κάθε κόμβο προς τις πύλες εκατέρωθέν του, και από την κάθε πύλη προς τους κόμβους εισόδου και εξόδου της.

Μετά το πέρας αυτής της διαδικασίας, έχουμε αναπαραστήσει πλήρως τον γράφο του κυκλώματος και γνωρίζουμε και τους κόμβους εισόδου του που έχουν τοποθετηθεί στην αρχική λίστα. Στο τελευταίο βήμα θέτουμε τις αρχικές χωρητικότητες των κόμβων που ορίζει το αρχείο SPICE, όπου σε κάθε περίπτωση συμπεριλαμβάνονται οι χωρητικότητες εξόδου του κυκλώματος, και εφόσον θέλουμε να συνυπολογιστούν οι χωρητικότητες διασύνδεσης στην περιγραφή του, ορίζονται και πρόσθετες χωρητικότητες για κάθε άλλο κόμβο του κυκλώματος.

Εδώ θα κάνουμε μια μικρή παρένθεση για να περιγράψουμε συνοπτικά τις διαφορές μεταξύ *συνδυαστικών* και *ακολουθιακών* κυκλωμάτων καθώς αυτός ο διαχωρισμός είναι απαραίτητος όσον αφορά στον τρόπο αναπαράστασης και επεξεργασίας τους.

3.1.2.1. Συνδυαστικά Κυκλώματα

Βιβλιογραφία [9]. *Συνδυαστικά ή κυκλώματα συνδυαστικής λογικής λέγονται τα ψηφιακά κυκλώματα των οποίων η έξοδος είναι μια καθαρή συνάρτηση της παρούσας εισόδου και μόνο αυτής.* Η συνδυαστική λογική χρησιμοποιείται για να εκτελεστούν

λογικές πράξεις (πράξεις της άλγεβρας *Bool*) πάνω σε σήματα και σε αποθηκευμένα δεδομένα.

3.1.2.2. Ακολουθιακά Κυκλώματα

Βιβλιογραφία [9]. Σε αντίθεση με τα συνδυαστικά, τα *ακολουθιακά ψηφιακά κυκλώματα είναι εκείνα των οποίων η έξοδός εξαρτάται όχι μόνο από την παρούσα είσοδο, αλλά επίσης και από την ιστορία των εισόδων που εφαρμόστηκαν στο παρελθόν.* Είναι δηλαδή κυκλώματα με μνήμη ενώ τα συνδυαστικά είναι κυκλώματα χωρίς μνήμη.

Όπως θα δούμε πιο συγκεκριμένα στην επόμενη ενότητα, τα ακολουθιακά κυκλώματα διαφοροποιούνται ελαφρώς στον τρόπο αναπαράστασής και επεξεργασίας τους από τα συνδυαστικά.

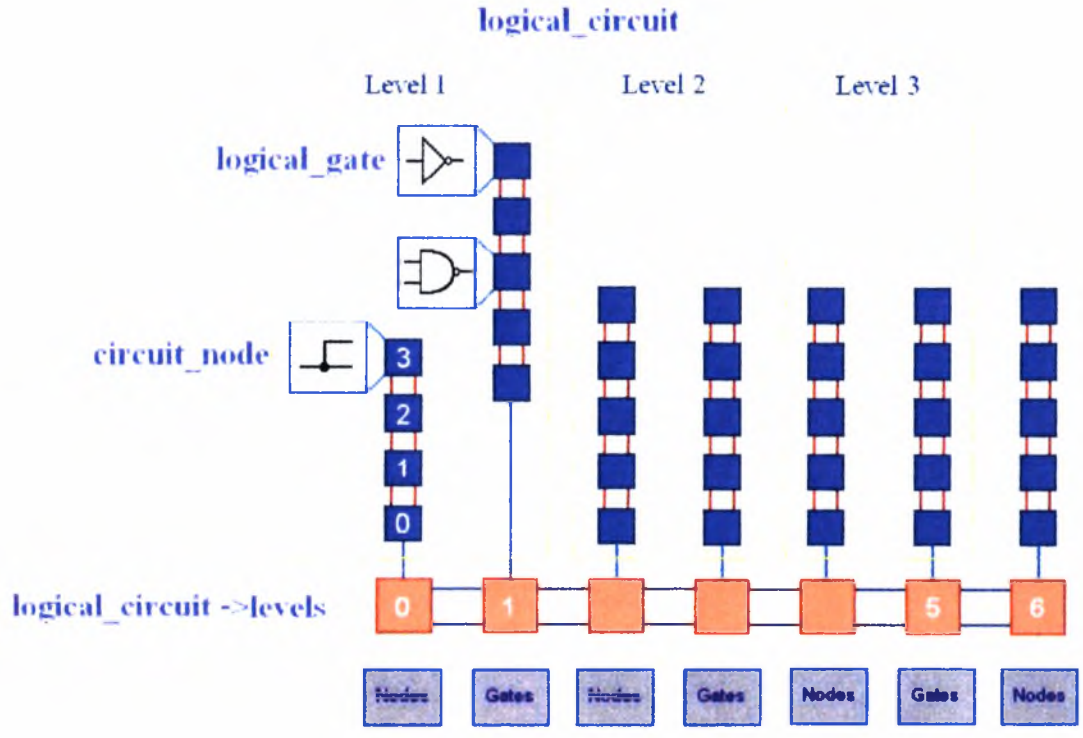
3.1.3. Κατηγοριοποίηση Πυλών σε Επίπεδα Λογικής

Όπως είπαμε νωρίτερα, κατά την ανάγνωση του αρχείου SPICE αποθηκεύουμε σε μια λίστα τους κόμβους εισόδου του κυκλώματος. Η λίστα αυτή αποτελεί το πρώτο επίπεδο κόμβων. Η έννοια του επιπέδου μιας πύλης ή ενός κόμβου, έχει να κάνει με την αλληλουχία διάδοσης των σημάτων εισόδου. Μια πύλη τελευταίου επιπέδου πρέπει να περιμένει τα σήματα να διασχίσουν το υπόλοιπο κύκλωμα για να φτάσουν στις εισόδους της. Αντίθετα, μια πύλη πρώτου επιπέδου έχει εξ' αρχής στη διάθεσή της τα σήματα εισόδου. Η έννοια του επιπέδου ορίζεται πιο αυστηρά από την παρακάτω αναδρομική πρόταση θεωρώντας πως οι κόμβοι εισόδου βρίσκονται στο επίπεδο 0 και ότι το επίπεδο των υπόλοιπων κόμβων του κυκλώματος συμπίπτει με το επίπεδο της πύλης που έχει σαν έξοδο τον εκάστοτε κόμβο. Έτσι: *Ο αριθμός του επιπέδου μιας πύλης ισούται με το μέγιστο των επιπέδων των κόμβων εισόδου της, αυξημένο κατά ένα.*

Χρειάζεται προσοχή όταν προσπαθούμε να κάνουμε ανάθεση επιπέδου στους κόμβους και τις πύλες των ακολουθιακών κυκλωμάτων. Τα ακολουθιακά κυκλώματα που συναντούμε, εκτός από τις γνωστές λογικές πύλες, περιέχουν στοιχεία μνήμης όπως είναι οι μανταλωτές από τους οποίους ξεκινούν βρόχοι ανάδρασης προς πύλες προηγουμένων επιπέδων. Για να μπορέσουμε να αποφύγουμε την αέναη επανάληψη με βάση τον παραπάνω ορισμό εύρεσης του επίπεδου κόμβων και πυλών, θα

θεωρήσουμε πως οι κόμβοι εξόδου των στοιχείων μνήμης αποτελούν και αυτοί κύριες εισόδους του κυκλώματος. Έτσι, μετά από αυτή την παραδοχή, σε κάθε ακολουθιακό κύκλωμα μπορεί να εφαρμοστεί το ίδιο απλά η διαδικασία της επιπεδοποίησης. Έτσι στο εξής θα χειριζόμαστε όλα τα κυκλώματα όπως τα συνδυαστικά.

Στο Σχήμα 3.1 βλέπουμε τον τρόπο αναπαράστασης του κυκλώματος σε επίπεδα πυλών και κόμβων μετά την εφαρμογή της διαδικασίας επιπεδοποίησης. Σε μια βασική λίστα, αποθηκεύουμε διαδοχικά λίστες κόμβων ακολουθούμενες από λίστες με πύλες όπου για κάθε εσωτερική λίστα, οι πύλες ή οι κόμβοι που περιέχει βρίσκονται στο ίδιο επίπεδο.



Σχήμα 3.1:

Αναπαράσταση του κυκλώματος σε επίπεδα πυλών και κόμβων

Σε αυτό το σημείο η αναπαράσταση του κυκλώματος μέσα από αυτή τη δομή δεδομένων είναι αρκετά εύχρηστη ώστε να προχωρήσουμε στην εφαρμογή του αλγόριθμου βελτιστοποίησης του κυκλώματος στην επόμενη ενότητα.

3.1.4. Αλγόριθμος Βελτιστοποίησης

Ο αλγόριθμος βελτιστοποίησης είναι ένας αλγόριθμος επανάληψης του ίδιου κύκλου εργασιών μέχρι να ικανοποιηθούν κάποια κριτήρια τερματισμού. Ποιοτικά, ο επαναληπτικός αλγόριθμος σε κάθε κύκλο κάνει τα εξής: Εντοπίζει το πιο αργό μονοπάτι του κυκλώματος και ελαχιστοποιεί την καθυστέρησή του μέσω της μεθόδου του logical effort. Μετά την ελαχιστοποίηση αυτή διακρίνουμε τις εξής δύο περιπτώσεις:

1. Η πρώτη είναι το μονοπάτι που βελτιστοποιήσαμε να έχει γίνει πιο γρήγορο ή να έχει παραμείνει το ίδιο αργό, και να εξακολουθεί να είναι το πιο αργό μονοπάτι του κυκλώματος. Σε αυτή την περίπτωση ο αλγόριθμος θα σταματήσει γιατί αυτό το μονοπάτι είναι αδύνατον να γίνει πιο γρήγορο καθώς έχει βελτιστοποιηθεί.
2. Η δεύτερη περίπτωση είναι το μονοπάτι που βελτιστοποιήσαμε να έχει γίνει πιο γρήγορο, ενώ έχει πάψει να είναι πια το πιο αργό, και τη θέση του έχει πάρει ένα άλλο μονοπάτι. Σε αυτή την περίπτωση, υπάρχουν άλλες δύο εκδοχές:
    a. Η πρώτη είναι το νέο χειρότερο μονοπάτι να είναι πιο αργό απ' ότι ήταν το κρίσιμο μονοπάτι πριν τη βελτιστοποίηση (αυτό είναι δυνατόν να συμβεί καθώς η βελτιστοποίηση ενός μονοπατιού μπορεί να κάνει ένα άλλο μονοπάτι πιο αργό). Και σε αυτή την περίπτωση σταματάει ο αλγόριθμος καθώς έκανε το συνολικό κύκλωμα πιο αργό απ' ότι ήταν στο προηγούμενο βήμα.
    b. Η δεύτερη εκδοχή είναι το νέο χειρότερο μονοπάτι να είναι πιο γρήγορο απ' ότι ήταν το κρίσιμο μονοπάτι πριν τη βελτιστοποίηση, το οποίο σημαίνει πως το κύκλωμα συνολικά έγινε πιο γρήγορο μετά από τον κύκλο που πέρασε και άρα συνεχίζουμε σε επόμενο κύκλο βελτιστοποίησης καθώς όπως είπαμε, κρίσιμο έχει γίνει ένα νέο μονοπάτι.

Πρακτικά, τα βήματα που εκτελούνται σειριακά σε κάθε κύκλο είναι τα ακόλουθα και θα αναλυθούν περισσότερο κάθε ένα ξεχωριστά:

1. Υπολογισμός των Βοηθητικών Ποσοτήτων
2. Εύρεση των *N* πιο Κρίσιμων Μονοπατιών

3. Εφαρμογή Logical Effort στα Μονοπάτια
4. Έλεγχος Κριτηρίων Τερματισμού

Ο λόγος που στο δεύτερο βήμα βρίσκουμε τα *N* χειρότερα μονοπάτια αντί του ενός χειρότερου, είναι ότι στατιστικά σε ορισμένα κυκλώματα έχουμε καλύτερα αποτελέσματα αν βελτιστοποιούμε αντί για ένα, και τα *N* χειρότερα μονοπάτια σε έναν κύκλο για κάποιες τιμές του *N*. Θα αναφερθούμε με μεγαλύτερη λεπτομέρεια στη βελτιστοποίηση των *N* μονοπατιών παρακάτω.

Μετά από κάθε κύκλο, ελέγχεται αν ικανοποιήθηκαν τα κριτήρια τερματισμού, και αν αυτό συνέβη σταματάει και η επανάληψη του κύκλου εργασιών.

3.1.4.1. Υπολογισμός των Βοηθητικών Ποσοτήτων

Σε αυτό το βήμα, γίνεται ο υπολογισμός για κάθε πύλη των βοηθητικών ποσοτήτων της μεθόδου του logical effort και τα αποτελέσματά τους αποθηκεύονται στους αντίστοιχους κόμβους και πύλες της δομής του κυκλώματος.

Πιο συγκεκριμένα, για κάθε πύλη τίθενται οι τιμές του logical effort, της παρασιτικής καθυστέρησης, των τιμών της χωρητικότητας εξόδου και εισόδου, και μέσω αυτών και του electrical effort της πύλης. Έτσι, έχοντας γνώση αυτών των τιμών για κάθε πύλη, είμαστε σε θέση να γνωρίζουμε την καθυστέρησή της αλλά και την συνολική καθυστέρηση του κάθε μονοπατιού.

3.1.4.2. Εύρεση των *N* πιο Κρίσιμων Μονοπατιών

Ένα κύκλωμα, θεωρείται τόσο αργό όσο αργό είναι το πιο αργό μονοπάτι του. Έχει επικρατήσει, το πιο αργό μονοπάτι ενός κυκλώματος να το λέμε κρίσιμο μονοπάτι.

Ένας αλγόριθμος εύρεσης των *N* κρισιμότερων μονοπατιών, προϋποθέτει τη γνώση της καθυστέρησης της κάθε πύλης την οποία γνωρίζουμε από το προηγούμενο βήμα. Ο αλγόριθμος που θα εφαρμόσουμε στο κύκλωμα είναι ο Αλγόριθμος 7.1 από τη Βιβλιογραφία [10].

Η δύναμη του αλγορίθμου αυτού έγκειται στο ότι μπορεί με δεδομένο το πιο αργό μονοπάτι του κυκλώματος χρησιμοποιώντας την έννοια του *branch slack* (ελάττωση διακλάδωσης) να βρει πολύ φθηνά και γρήγορα ποιο είναι το αμέσως λιγότερο αργό μονοπάτι. Επαναλαμβάνοντας αυτή τη διαδικασία *N*-1 φορές, βρίσκοντας σε κάθε

βήμα το αμέσως λιγότερο αργό μονοπάτι, έχουμε τελικά τα *N* κρισιμότερα. Ο πίνακας που ακολουθεί δείχνει την υπεροχή αυτού του αλγορίθμού σε σχέση με τους αλγορίθμους Best First και Depth First. Στον Πίνακα 3.1 φαίνονται οι χρόνοι εκτέλεσης σε δευτερόλεπτα CPU για τα τρία μεγαλύτερα κυκλώματα καθώς και το ποσοστό των χωρίς αναγκαιότητα διαπεράσεων ακμών.

Κύκλωμα	*Αλγόριθμος*	*N = 50*	*N = 500*	*N = 5000*
C5315	Best First	0.10 (3%)	1.19 (3%)	40.24 (3%)
	Depth First	0.18 (58%)	0.91 (61%)	16.84 (64%)
	Αλγόριθμος 7.1	0.06 (0%)	0.52 (0%)	5.15 (0%)
C6288	Best First	0.22 (4%)	2.46 (5%)	61.30 (6%)
	Depth First	0.64 (70%)	8.06 (78%)	94.83 (83%)
	Αλγόριθμος 7.1	0.17 (0%)	1.49 (0%)	14.52 (0%)
C7552	Best First	0.08 (4%)	1.00 (3%)	32.77 (3%)
	Depth First	0.13 (70%)	1.15 (56%)	11.56 (59%)
	Αλγόριθμος 7.1	0.03 (0%)	0.44 (0%)	4.44 (0%)

Πίνακας 3.1:

Χρόνος σε δευτερόλεπτα CPU των αλγορίθμων Best First, Depth First και 7.1. Σε παρένθεση αναγράφεται το ποσοστό των χωρίς αναγκαιότητα διαπεράσεων ακμών κατά την εκτέλεση του κάθε αλγορίθμου. Αξίζει να παρατηρήσουμε πως το ποσοστό αυτό για τον αλγόριθμο 7.1 είναι 0%.

3.1.4.3. Εφαρμογή Logical Effort στα Μονοπάτια

Σε αυτό το στάδιο έχουμε στη διάθεσή μας διατεταγμένα ως προς την καθυστέρηση τα *N* πιο κρίσιμα μονοπάτια του κυκλώματος. Σε αυτό το σημείο εφαρμόζουμε σε κάθε ένα από αυτά τη μέθοδο του logical effort. Η εφαρμογή της μεθόδου γίνεται από το λιγότερο αργό προς το πιο αργό μονοπάτι, διότι με τον τρόπο αυτό έχουμε καλύτερα αποτελέσματα.

Διαισθητικά, ο λόγος για τον οποίο συμβαίνει αυτό είναι ότι βελτιστοποιώντας ένα λιγότερο αργό μονοπάτι είναι πιθανό μεν να βλάψουμε ένα κρισιμότερό του, όμως προσωρινά, καθώς η βελτιστοποίηση εκείνου δεν έχει γίνει ακόμη. Έτσι, ακόμα και αν χειροτερέψει, όταν θα βελτιστοποιηθεί στη συνέχεια, θα γίνει ξανά βέλτιστο, αναιρώντας πιθανώς μεν μέρος της βελτιστοποίησης του λιγότερο αργού (το οποίο έχει περιθώρια να γίνει πιο αργό καθώς δεν είναι το πιο αργό μονοπάτι του κυκλώματος) όμως συνολικά κάνοντας το κύκλωμα πιο γρήγορο από πριν. Αν βελτιστοποιούσαμε από το πιο αργό προς το λιγότερο αργό, θα ήταν αρκετά πιθανό οι βελτιστοποιήσεις των λιγότερο αργών μονοπατιών που θα γίνονταν μετά τις

βελτιστοποιήσεις των κρίσιμων, να χειροτέρευαν ξανά τα πιο κρίσιμα μονοπάτια, πράγμα που θα σταματούσε εκεί τον αλγόριθμο διότι το κρισιμότερο μονοπάτι δεν θα είχε μεγάλα περιθώρια να αυξηθεί η καθυστέρησή του.

Η εφαρμογή της μεθόδου του logical effort γίνεται σταδιακά, όπως αναφέρθηκε στο Κεφάλαιο 2: Με δεδομένο το μονοπάτι, υπολογίζουμε τις τιμές των:

1. Branching Effort *B*
2. Logical Effort *G*
3. Electrical Effort *H*.

Ας σημειωθεί πως οι τιμές αυτές αναφέρονται στο μονοπάτι. Το γινόμενό τους θα είναι το path effort *F* από το οποίο προκύπτει το ιδανικό stage effort μέσω ύψωσης στη δύναμη (1/*K*) όπου *K* ο αριθμός των πυλών του μονοπατιού. Στη συνέχεια περνάμε στο στάδιο της διαδοχικής αλλαγής των μεγεθών των πυλών του μονοπατιού, αλλά και των πυλών που ανήκουν στις άμεσες διακλαδώσεις του.

Εδώ εφαρμόζουμε διαδοχικά για κάθε χωρητικότητα εισόδου, από την τελευταία ως την πρώτη, την σχέση:

$$C_{in_i} = \frac{g_i C_{out_i}}{\hat{f}}$$

Εκτός από την χωρητικότητα εισόδου της πύλης του μονοπατιού στο τρέχον επίπεδο, πρέπει να αλλάξουν μεγέθη και οι χωρητικότητες εισόδου των πυλών που βρίσκονται στην ίδια διακλάδωση με την πύλη που ανήκει στο μονοπάτι και η οποία μόλις άλλαξε μέγεθος. Εάν *Cold* είναι η χωρητικότητα εισόδου της πύλης του μονοπατιού πριν την αλλαγή και *Cnew* η χωρητικότητά της μετά την αλλαγή, η χωρητικότητά των πυλών της διακλάδωσης πρέπει να πολλαπλασιαστεί με τον παράγοντα (*Cold/Cnew*). Όταν αυτή η διαδικασία επαναληφθεί για κάθε επίπεδο λογικής του μονοπατιού, η καθυστέρησή του πρέπει να έχει ελαχιστοποιηθεί, ενώ η χωρητικότητα εισόδου της πύλης του πρώτου επιπέδου θα έχει μείνει αμετάβλητη.

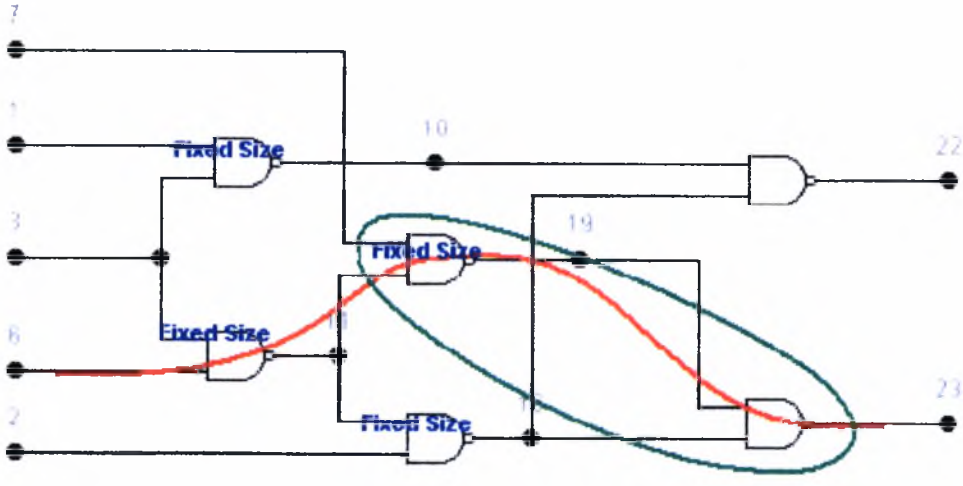
Πρέπει να παρατηρηθεί πως η εφαρμογή της μεθόδου του logical effort δεν μπορεί να γίνει πάντοτε αυτούσια με κλειστά τα μάτια λόγω των προδιαγραφών των κυκλωμάτων. Κάθε κύκλωμα πρέπει να έχει σταθερή χωρητικότητα εισόδου σε κάθε κόμβο εισόδου του. Αυτός ο περιορισμός σε ορισμένες περιπτώσεις δεν μας επιτρέπει την εφαρμογή του αλγορίθμου σε ολόκληρο το μονοπάτι, αλλά μόνο σε ελεύθερα από πλευράς περιορισμών αλλαγής των μεγεθών των πυλών τους, υπομονοπάτια. Ο

περιορισμός αυτός μας δημιουργεί πρόβλημα όταν κατά τη διάρκεια της βελτιστοποίησης ενός μονοπατιού, προσπαθούμε να διατηρήσουμε το branching effort σταθερό σε κάθε κόμβο ώστε όπως έχουμε πει να διατηρηθεί τελικά η χωρητικότητα εισόδου στο πρώτο επίπεδο πυλών σταθερή. Αυτό συμβαίνει όταν κατά την απόπειρα αλλαγής των μεγεθών των πυλών διακλάδωσης κατά τον ίδιο παράγοντα αλλαγής με αυτόν της πύλης που ανήκει στο μονοπάτι (ώστε να διατηρηθεί σταθερό το branching effort), διαπιστώνουμε πως μια ή περισσότερες από τις πύλες αυτές δεν επιτρέπεται να αλλάξει μέγεθος (άρα ούτε να πολλαπλασιαστεί με τον παράγοντα) καθώς μία από τις εισόδους της τυχαίνει να είναι και κύρια είσοδος του κυκλώματος.

Τα παραπάνω προβλήματα καθώς και μια άλλη σημαντική πιθανότητα που πρέπει να ληφθεί υπ' όψιν θα αναλυθούν εκτενέστερα στις ενότητες που ακολουθούν.

3.1.4.3.1. Σπάσιμο Μονοπατιών

Όταν σε ένα μονοπάτι εκτός από την πρώτη κατά σειρά πύλη, υπάρχουν και άλλες οι οποίες πρέπει ή εμείς προτιμούμε να μείνουν αμετάβλητες ως προς το μέγεθός τους, τότε εφαρμόζουμε τον αλγόριθμο του logical effort τμηματικά, δηλαδή τον εφαρμόσουμε σε υπομονοπάτια του εν λόγω κρίσιμου μονοπατιού αφήνοντας τις επιπλέον πύλες αμετάβλητες.



Σχήμα 3.2

3.1.4.3.1.1. Πρώτη Εκδοχή

Όπως φαίνεται στο Σχήμα 3.2, το κρίσιμο μονοπάτι είναι αυτό που αποτελείται από τους κόμβους 23, 19, 11 και 6. Οι πύλες που αντιστοιχούν στους κόμβους 11 και 19 είναι δηλωμένες ως σταθερού μεγέθους διότι τουλάχιστον μία από τις εισόδους τους είναι είσοδος του κυκλώματος και άρα πρέπει να διατηρήσουν τη χωρητικότητα εισόδου τους για να υπάρχει συμμόρφωση με τις προδιαγραφές. Αυτό σημαίνει πως δεν μπορούμε να εφαρμόσουμε τη μέθοδο βελτιστοποίησης σε ολόκληρο το μονοπάτι αλλά πρέπει να αποκλειστεί επιπλέον η αλλαγή της πύλης 19 (ως πύλη 19 εννοούμε την πύλη που έχει ως έξοδο τον κόμβο 19). Έτσι προσπαθούμε να βελτιστοποιήσουμε το μονοπάτι που αποτελείται από τις πύλες 23 και 19 με την 19 αμετάβλητη καθώς όπως έχουμε πει, η 19 ως πρώτη στο μονοπάτι θα παραμείνει σταθερή λόγω της φύσης της μεθόδου του logical effort.

Όταν σε ένα μεγαλύτερο από αυτό του παραδείγματος μονοπάτι οι πύλες που είναι δηλωμένες ως αμετάβλητες είναι διάσπαρτες κατά μήκος αυτού, τότε εφαρμόζουμε τη μέθοδο του logical effort σε περισσότερα του ενός υπομονοπάτια βελτιστοποιώντας εκείνα και άρα το συνολικό μονοπάτι, χωρίς να μεταβληθούν τα μεγέθη των πυλών που θέλουμε να μείνουν σταθερά.

3.1.4.3.1.2. Δεύτερη Εκδοχή

Εκτός από το υποχρεωτικό κριτήριο των αμετάβλητων πυλών για το σπάσιμο των μονοπατιών, υπάρχει και ένα επιπλέον προαιρετικό, με στόχο την μεγαλύτερη βελτίωση ορισμένων κυκλωμάτων.

Η ιδέα για αυτή την παραλλαγή του αλγορίθμου είναι πως όταν βελτιστοποιείται το κρίσιμο μονοπάτι, τα μεγέθη των τρανζίστορ των πυλών του, καθώς και των πυλών που ανήκουν στις διακλαδώσεις των κόμβων του αποκτούν βέλτιστες τιμές. Έτσι, σε επόμενο κύκλο του αλγορίθμου με νέο μονοπάτι προς βελτιστοποίηση ίσως θέλουμε να βελτιώσουμε το νέο κρίσιμο μονοπάτι, χωρίς να χειροτερέψουν τα μονοπάτια που έχουν ήδη βελτιστοποιηθεί. Έτσι, αυτή η παραλλαγή, επιτρέπει σε μια πύλη, να αλλάξει μόνο μια φορά, όταν δηλαδή θα πάρει τη βέλτιστη τιμή της για ένα από τα πιο κρίσιμα μονοπάτια, ενώ από εκείνο το σημείο μέχρι το πέρας της διαδικασίας βελτιστοποίησης του κυκλώματος, θεωρείται πύλη με σταθερό μέγεθος. Εφόσον η πύλη αυτή ανήκει και σε ένα από τα μεταγενέστερα κρίσιμα μονοπάτια σε κάποιο

κύκλο του αλγορίθμου, το μονοπάτι εκείνο θα σπάσει σε υπομονοπάτια θεωρώντας αυτή και κάθε αλλαγμένη πύλη, ως πύλη με σταθερό μέγεθος.

Η εκδοχή αυτή έχει πρόσθετο πλεονέκτημα τη μειωμένη διάρκεια εκτέλεσης του αλγορίθμου καθώς το τρέχον κρίσιμο μονοπάτι έχει την τάση όσο ο αλγόριθμος προχωράει, να σπάει σε μικρότερου μήκους υπομονοπάτια και να εφαρμόζεται η μέθοδος του logical effort σε μικρότερο συνολικό αριθμό πυλών του μονοπατιού.

3.1.4.3.2. Επαναληπτικός Αλγόριθμος

Υπάρχουν μερικά κυκλώματα, που η ιδιόμορφη τοπολογία τους δημιουργεί προβλήματα που δεν λύνονται με σπάσιμο των μονοπατιών. Τα προβλήματα αυτά είναι τρία:

1. Σταθερές Πύλες Διακλάδωσης
2. Κύκλος Πύλης Διακλάδωσης
3. Χωρητικότητες Διασύνδεσης

Οι τρεις αυτές περιπτώσεις, δεν επιτρέπουν στη μέθοδο του logical effort να διατηρήσει σταθερή τη χωρητικότητα εισόδου του μονοπατιού χωρίς κάποια επέμβαση. Ο επαναληπτικός αλγόριθμος είναι ένας αλγόριθμος σύγκλισης που προσπαθεί να λύσει αυτό το πρόβλημα και κάνει τα εξής:

1. Πριν την έναρξη της βελτιστοποίησης ενός μονοπατιού αποθηκεύει τις τιμές του branching effort που έχει κάθε κόμβος.
2. Κατά τη διάρκεια της αλλαγής των μεγεθών της κάθε πύλης του μονοπατιού και των αντίστοιχων πυλών διακλάδωσης, ελέγχει αν το branching effort στον εκάστοτε κόμβο έμεινε το ίδιο σε σχέση με την τιμή που είχε πριν την έναρξη της τρέχουσας επανάληψης.
    a. Αν έμεινε ίδιο, συνεχίζει κανονικά την βελτιστοποίηση του μονοπατιού στον κόμβο που έχει σειρά.
    b. Αν άλλαξε, σταματάει την βελτιστοποίηση στον κόμβο που βρίσκεται καθώς δεν πρέπει να συνεχίσει για να μην αλλοιώσει την τιμή της σταθερής χωρητικότητας εισόδου του μονοπατιού. Στη συνέχεια εφαρμόζει τη μέθοδο του logical effort στο μονοπάτι από την αρχή.

Επαναλαμβάνοντας την εφαρμογή της μεθόδου του logical effort ξανά και ξανά, το branching effort αρχίζει να συγκλίνει σε μια συγκεκριμένη τιμή, επομένως η διαφορά

της τιμής του branching effort του βήματος *n* σε σχέση με εκείνη του βήματος *n-1* πέφτει κάτω από ένα όριο ανοχής που έχουμε ορίσει, οπότε και η ανάθεση βέλτιστων μεγεθών συνεχίζεται κανονικά σε επόμενες (πιο κοντινές στις εισόδους του κυκλώματος) πύλες του μονοπατιού.

Αν σε κάποια επανάληψη όλα τα branching efforts στο μονοπάτι παραμείνουν ίδια με αυτά της προηγούμενης επανάληψης (η απόλυτη διαφορά τους είναι μικρότερη από το όριο ανοχής), ο αλγόριθμος τερματίζει. Μετά το πέρας του αλγορίθμου, η χωρητικότητα στην πύλη εισόδου θα είναι ίδια με την αρχική ικανοποιώντας τις προδιαγραφές του κυκλώματος. Τώρα θα δούμε αναλυτικότερα τις τρεις περιπτώσεις στις οποίες απαιτείται η εφαρμογή του επαναληπτικού αλγορίθμου.

3.1.4.3.2.1. Σταθερές Πύλες Διακλάδωσης

Το πρώτο πρόβλημα αφορά ξανά στους αρχικούς περιορισμούς του κυκλώματος. Η μέθοδος του logical effort εκτός από τις πύλες που ανήκουν στο υπό βελτιστοποίηση μονοπάτι, μεταβάλλει τα μεγέθη και των πυλών που ανήκουν στις άμεσες διακλαδώσεις του μονοπατιού. Είναι όμως πιθανό οι πύλες των διακλαδώσεων να είναι πύλες σταθερού μεγέθους λόγω των προδιαγραφών.

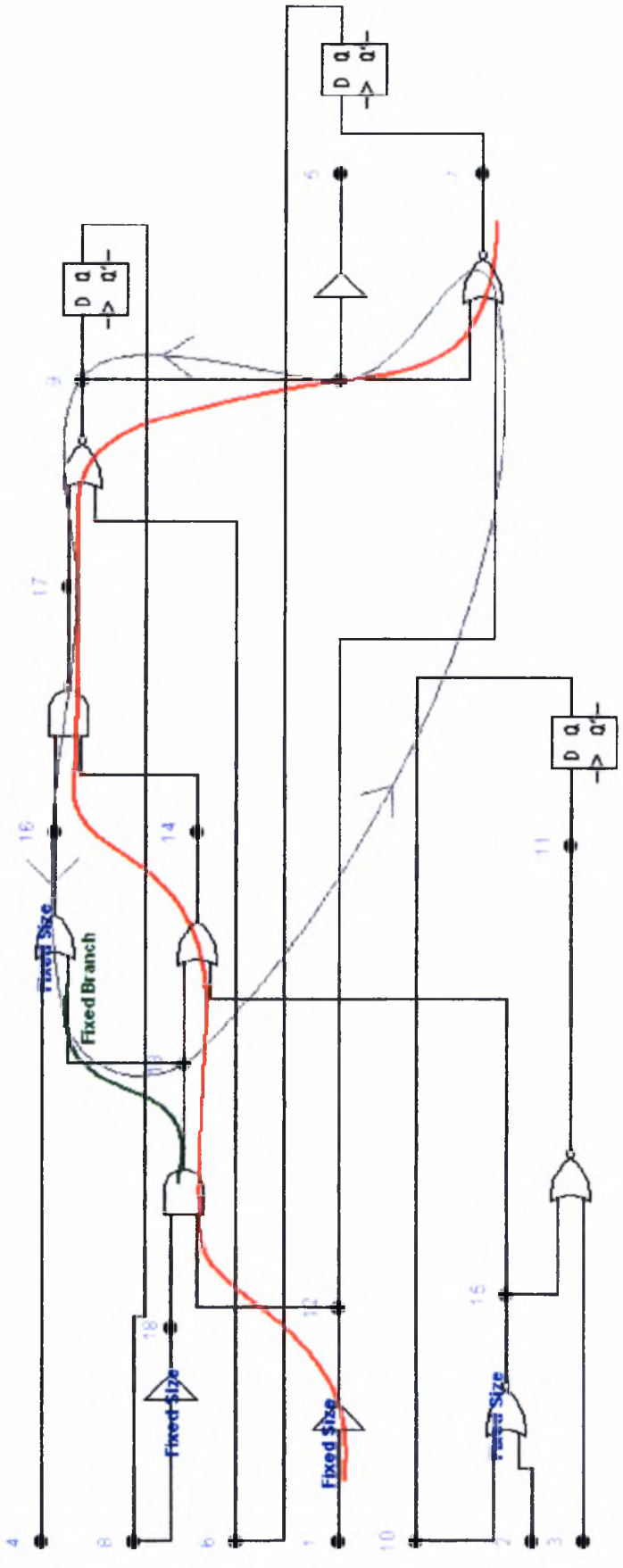
Σε αυτή την περίπτωση, κατά τη διάρκεια της εφαρμογής της μεθόδου, αλλάζουμε κανονικά όλες τις πύλες της διακλάδωσης, εκτός αυτών που πρέπει να μείνουν σταθερές, το οποίο όμως προκαλεί αλλαγή στο branching effort του κόμβου με αποτέλεσμα την ενεργοποίηση του επαναληπτικού αλγορίθμου.

Στο Σχήμα 3.3 βλέπουμε ένα ακολουθιακό κύκλωμα, στο οποίο η πύλη 16 έχει οριστεί ως σταθερού μεγέθους, όμως παράλληλα ανήκει στις διακλαδώσεις του κόμβου 13 του κρίσιμου μονοπατιού. Αυτό έχει σαν αποτέλεσμα, όταν η μέθοδος του logical effort φτάσει στον κόμβο αυτό να μην αλλάξει το μέγεθος της πύλης 16 με συνέπεια την αλλαγή του branching effort στον κόμβο. Τότε, η μέθοδος του logical effort σταματά εκεί χωρίς να αλλάξει το μέγεθος της πύλης 13 και επαναλαμβάνει τους υπολογισμούς του stage effort για το μονοπάτι ώστε να αρχίσει από την αρχή νέα βελτιστοποίηση, μέχρις ότου να συγκλίνει η τιμή του branching effort του κόμβου 13 σε μια σταθερή τιμή. Η ροή του επαναληπτικού αλγορίθμου αναπαρίσταται μέσω της χρώματος *γκρι* κυκλικής διαδρομής. Όταν έχουμε σύγκλιση, θα μπορούμε πια να αλλάξουμε με ασφάλεια το μέγεθος της πύλης 13 όπως

προβλέπει η μέθοδος του logical effort όταν έχουμε σταθερά branching efforts κατά μήκος του μονοπατιού καθ' όλη τη διαδικασία εφαρμογής της.

Όπως είπαμε και στην ανάλογη περίπτωση της ενότητας όπου αναλύεται το σπάσιμο μονοπατιών, μπορούν να θεωρηθούν σταθερές, πύλες διακλάδωσης που έχουν αλλαχθεί σε προηγούμενο κύκλο βελτιστοποίησης για τυχόν καλύτερα αποτελέσματα.

Σύμφωνα με αυτά που έχουμε αναλύσει μέχρι τώρα, είμαστε σε θέση μέσω του επαναληπτικού αλγορίθμου, να κρατάμε σταθερά τα μεγέθη πυλών εισόδου που έχουν χαρακτηριστεί ως αμετάβλητες σε ένα μονοπάτι, είτε συναντήσουμε αμετάβλητες πύλες στον κορμό, είτε τις συναντήσουμε στις άμεσες διακλαδώσεις του μονοπατιού.



Σχήμα 3.3

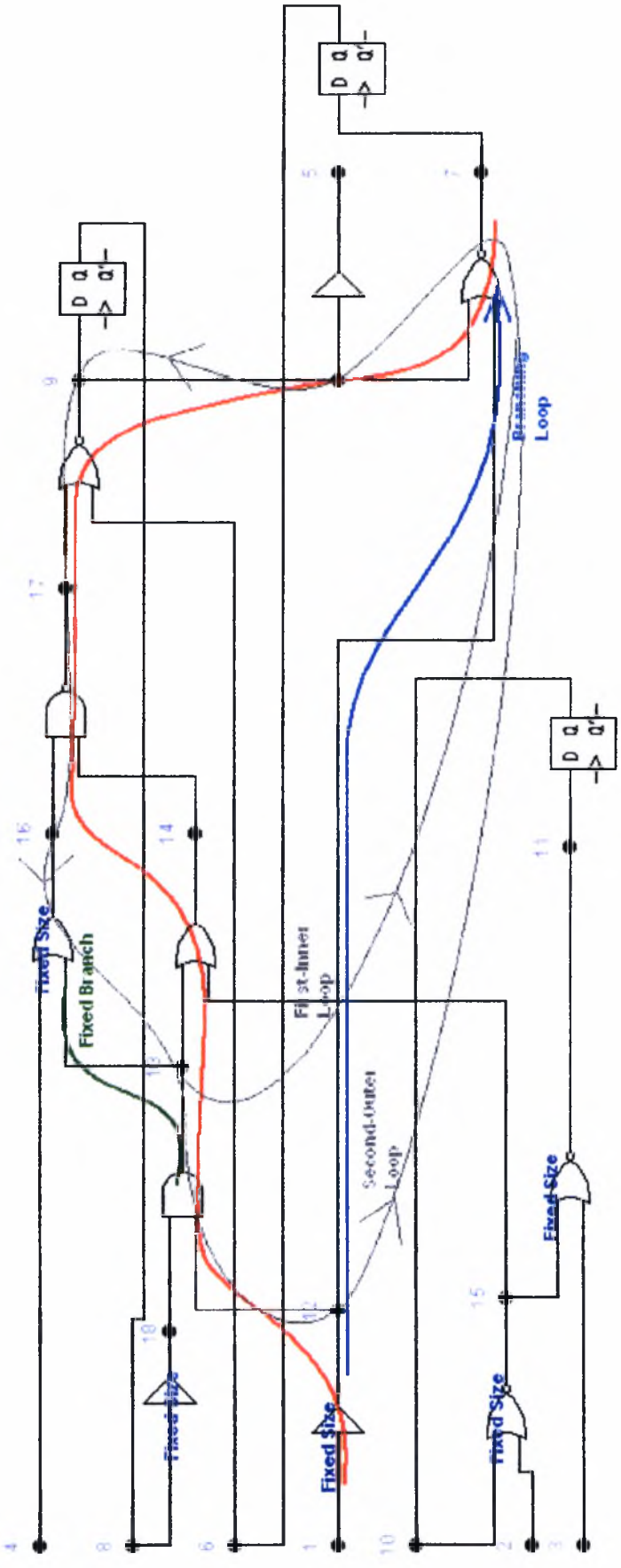
3.1.4.3.2.2. Κύκλος Πύλης Διακλάδωσης

Το δεύτερο πρόβλημα εμφανίζεται όταν μια από τις πύλες που επηρεάζει η μέθοδος του logical effort, δηλαδή μία από τις πύλες του κορμού του μονοπατιού ή μια από αυτές των άμεσων διακλαδώσεών του, αλλάζει τουλάχιστον δύο φορές κατά τη διάρκεια μιας εφαρμογής της μεθόδου. Αυτό μπορεί να συμβεί σε δύο περιπτώσεις:

1. Όταν η πύλη διακλάδωσης ενός μπροστινού κόμβου στο μονοπάτι είναι και διακλάδωση ενός πίσω κόμβου του μονοπατιού.
2. Όταν μια μπροστινή πύλη του μονοπατιού, είναι και διακλάδωση ενός πίσω κόμβου του μονοπατιού.

Η διπλή αυτή αλλαγή αυτή προκαλεί και αλλαγή στο branching effort του πίσω κόμβου, με αποτέλεσμα την ενεργοποίηση του επαναληπτικού αλγορίθμου. Σε ένα μονοπάτι μπορούν να εμφανιστούν πολλαπλοί κόμβοι που να εμφανίσουν διαφοροποιημένο branching effort σε μία απόπειρα εφαρμογής της μεθόδου του logical effort. Αυτό δυστυχώς συνήθως συνεπάγεται πως ο αλγόριθμος θα εφαρμοστεί πολύ περισσότερες φορές μέχρι να συγκλίνει διότι κάθε επιπλέον κόμβος με διαφοροποιημένο branching effort σημαίνει επανάληψη μεγαλύτερης τάξης, δηλαδή εμφωλευμένες επαναλήψεις. Στην πράξη η πολυπλοκότητα του αλγορίθμου είναι αρκετά μικρότερη από L^C (όπου L ο μέσος αριθμός επαναλήψεων για μια απλή σύγκλιση και C ο αριθμός των κόμβων επανάληψης στο ίδιο μονοπάτι), καθώς η κάθε απλή επανάληψη, μετά την πρώτη της σύγκλιση αλλοιώνεται (μεγαλώνει το σφάλμα της μεταβλητής σύγκλισής της) σε μικρό βαθμό από μια εξωτερική επανάληψη, με αποτέλεσμα να μην χρειάζεται να αρχίσει από την αρχή ολόκληρη η διαδικασία σύγκλισης για τον εσωτερικό – πρώτο που συναντούμε στη διαδικασία σύγκλισης κόμβο.

Στο Σχήμα 3.4, φαίνεται το προηγούμενο παράδειγμα, με τη διαφορά ότι η διαδικασία έχει προχωρήσει έναν κόμβο προς τα πίσω όπου συναντάται η περίπτωση της κοινής πύλης διακλάδωσης στην πύλη 7. Η πύλη 7 ανήκει στο μονοπάτι, αλλά είναι και διακλάδωση μια πίσω πύλης (της 12) του μονοπατιού. Αυτό το σενάριο έχει σαν αποτέλεσμα την διπλή αλλαγή του μεγέθους της πύλης 7, μία φορά στην αρχή, και μία κατά την αλλαγή των μεγεθών των διακλαδώσεων της πύλης 12.


Fixed Size
Fixed Branch
Fixed Size
Fixed Size
Fixed Size
Fixed Size
Loop
Branching

Σχήμα 3.4

Στο παράδειγμα αυτό, όπως βλέπουμε δημιουργείται εμφωλευμένη, διπλή επανάληψη κάθε σκέλος της οποίας αναπαρίσταται με μια χρώματος *γκρι* κυκλική διαδρομή. Πρώτα συγκλίνει το branching effort της πύλης 13, και στη συνέχεια αλλάζει το branching effort της πύλης 12. Έτσι, για κάθε αλλαγή του branching effort της πύλης 12, απαιτείται ξανά μια σύγκλιση της πύλης 13 μέχρις ότου να συγκλίνει και η 12. Η πολυπλοκότητα όπως είπαμε φαίνεται τετραγωνική, όμως στην πράξη, όσο συγκλίνει η πύλη 12 σε μια σταθερή τιμή, τόσο μειώνονται τα βήματα σύγκλισης της πύλης 13 καθώς μειώνεται η απόλυτη θετική επίδραση που έχει η κάθε επανάληψη στην πύλη 12, και κατά συνέπεια η αρνητική επίδραση που έχει η παραπάνω στην πύλη 13.

3.1.4.3.2.3. Χωρητικότητες Διασύνδεσης

Το πρόβλημα των χωρητικοτήτων διασύνδεσης είναι ακριβώς ανάλογο με αυτό της πρώτης περίπτωσης ενεργοποίησης του επαναληπτικού αλγορίθμου, δηλαδή με το πρόβλημα των σταθερών πυλών διακλάδωσης. Η χωρητικότητα διασύνδεσης σε κάθε κόμβο, μπορεί και αυτή να θεωρηθεί ως μια σταθερού μεγέθους πύλη που ανήκει στις διακλαδώσεις του κόμβου. Έτσι, προστίθεται πολύ απλά σαν χωρητικότητα, στην χωρητικότητα εξόδου της κάθε πύλης. Έτσι έχουμε άλλη μια ενεργοποίηση του επαναληπτικού αλγορίθμου.

Από πλευράς χρόνου εκτέλεσης, κυκλώματα με χωρητικότητες διασύνδεσης σε κάθε κόμβο, τείνουν να έχουν αυξημένη πολυπλοκότητα καθώς πια κάθε κόμβος του κυκλώματος και άρα του οποιουδήποτε μονοπατιού, θα συνιστά και ένα σημείο που απαιτείται σύγκλιση του branching effort. Αυτό συνεπάγεται πολλαπλές εμφωλευμένες επαναλήψεις, με το ελαφρυντικό της αρκετά μικρότερης από L^C πολυπλοκότητας όπως είπαμε και στην προηγούμενη ενότητα.

3.1.4.4. Έλεγχος Κριτηρίων Τερματισμού

Μετά τον κάθε κύκλο βελτίωσης πάνω στο κύκλωμα, υπάρχει ένα κριτήριο τερματισμού. Αυτό είναι πολύ απλό, και ελέγχει εάν το κύκλωμα βελτιώθηκε μετά τον τελευταίο κύκλο ή όχι. Εάν έχει βελτιωθεί, περνάμε σε επόμενο κύκλο βελτιστοποίησης. Εάν όχι, το κύκλωμα που προέκυψε μετά το πέρας του τελευταίου κύκλου, είναι χειρότερο από πριν. Στο σημείο αυτό, το καλύτερο κύκλωμα που έχουμε συναντήσει είναι εκείνο που είχαμε πριν την εφαρμογή του τελευταίου

κύκλου βελτιστοποίησης. Αυτό μας δείχνει πως είτε πρέπει να κρατάμε πάντα ένα αντίγραφο του κυκλώματος του προηγούμενου κύκλου, ή να υποστηρίζεται κάποια μέθοδος αναίρεσης των αλλαγών που επήλθαν από τον κύκλο βελτίωσης που μόλις τελείωσε. Επειδή για μεγάλα κυκλώματα, η αντιγραφή αυτή κοστίζει πολύ, προτιμήσαμε τη δεύτερη λύση, δηλαδή τη δυνατότητα αναίρεσης η οποία θα αναλυθεί στην επόμενη ενότητα.

Τελικά, όσο βελτιώνεται το κύκλωμα συνεχίζουμε να εφαρμόζουμε σε αυτό κύκλους βελτίωσης μέχρι κάποιος από αυτούς να αποτύχει, οπότε παραδίδουμε το τελικό κύκλωμα (το καλύτερο που συναντήσαμε κατά τη διάρκεια της διαδικασίας) στην έξοδο του προγράμματος.

3.1.5. Αναίρεση Τελευταίων Αλλαγών

Το πρόγραμμά μας, πρέπει να έχει τη δυνατότητα αναίρεσης των αλλαγών του τελευταίου κύκλου βελτιστοποίησης. Σε κάθε κύκλο, γίνονται τροποποιήσεις σε *N* μονοπάτια, όπου *N* ο αριθμός των χειρότερων μονοπατιών που έχουν προκύψει από τον σχετικό αλγόριθμο ο οποίος αριθμός δίνεται ως είσοδος της εφαρμογής. Έτσι, πριν την τροποποίηση του κάθε μονοπατιού, αποθηκεύουμε τα μεγέθη των πυλών που πρόκειται να αλλαχθούν κατά την εφαρμογή της μεθόδου του logical effort και του επαναληπτικού αλγορίθμου.

Με την κάθε εφαρμογή του αλγόριθμου του logical effort, οι πύλες που τροποποιούνται, εκτός από εκείνες που ανήκουν στο μονοπάτι, είναι και αυτές που ανήκουν στις άμεσες διακλαδώσεις του μονοπατιού. Έτσι πρέπει με κάποιον τρόπο να αποθηκεύουμε όλες τις υποψήφιες να αλλάξουν πύλες σε μια δομή αναίρεσης. Η δομή αυτή είναι μια λίστα με κατάλληλη σειρά αποθήκευσης, ώστε να είμαστε σε θέση να αντιστοιχίσουμε τις πύλες του μονοπατιού και αυτές των άμεσων διακλαδώσεών του, σε μια από τις πύλες της λίστας αναίρεσης.

Πρέπει να γίνει κατανοητός, ο λόγος για τον οποίο γίνεται αντιγραφή των υποψήφιων πυλών ενός μονοπατιού, αμέσως πριν βελτιστοποιηθεί, και όχι πριν την έναρξη του συνολικού αλγορίθμου βελτιστοποίησης για οποιοδήποτε μονοπάτι. Ο λόγος είναι καθαρά σχεδιαστικός. Θα μπορούσαμε να κρατάμε αντίγραφα για όλα τα *N* μονοπάτια, και στη συνέχεια μετά την αποτυχημένη βελτιστοποίηση να τους θέτουμε τις παλιές τους τιμές (με πιθανότητα μια πύλη να τεθεί στην πιο παλιά τιμή της περισσότερες της μιας φορές εφόσον ανήκει σε δύο ή περισσότερα υπό

βελτιστοποίηση μονοπάτια του κύκλου). Ο λόγος που δεν προτιμήσαμε κάτι τέτοιο, είναι η δυνατότητα αναίρεσης των αλλαγών που προκάλεσε η βελτιστοποίηση των τελευταίων μόνο μονοπατιών και όχι και των *N*. Αυτή η μέθοδος προτιμάται καθώς είναι πιθανό το κύκλωμα να βρέθηκε σε μια ενδιάμεση καλύτερη κατάσταση, την οποία όμως να χάλασε η βελτιστοποίηση ενός επόμενου από τα *N* μονοπάτια. Στην παρούσα εφαρμογή, κάτι τέτοιο είναι γενικά απίθανο να συμβεί, καθώς τα μονοπάτια βελτιστοποιούνται από το λιγότερο αργό προς το πιο αργό (και δεν είναι τόσο σύνηθες ένα λιγότερο αργό μονοπάτι να μειώνει την καθυστέρηση του πιο κρίσιμου), όμως προτιμήσαμε αυτή τη σχεδίαση για να είμαστε σε θέση να δοκιμάσουμε παραλλαγές του αλγορίθμου, πολλές από τις οποίες για μερικά κυκλώματα έχουν καλύτερα αποτελέσματα.

Αποθηκεύοντας τα αντίγραφα των υποψήφιων πυλών κάθε μονοπατιού αμέσως πριν τη βελτιστοποίηση του, αποθηκεύουμε τις τιμές που είχαν οι πύλες του μετά τις βελτιστοποιήσεις των προηγούμενων μονοπατιών (τα οποία είναι πολύ πιθανό να είχαν πύλες κοινές, που είτε ανήκουν σε αυτό, είτε στις άμεσες διακλαδώσεις του και να τις είχαν αλλάξει), και άρα κάνοντας αναίρεση με αντίστροφη σειρά, μπορούμε να φέρουμε το κύκλωμα σε οποιαδήποτε από τις *N*+1 καταστάσεις που πέρασε αμέσως πριν, και κατά τη διάρκεια της βελτιστοποίησης των *N* μονοπατιών.

3.1.6. Κβάντιση των Μεγεθών των Τρανζίστορ

Η κβάντιση των μεγεθών των τρανζίστορ είναι ένα βήμα που έχει να κάνει με την τεχνολογία κατασκευής. Τα μεγέθη των τρανζίστορ μιας πύλης δεν είναι δυνατόν να λάβουν οποιαδήποτε πραγματική τιμή. Για το λόγο αυτό εφαρμόζουμε μια προσέγγιση. Έχουμε κάνει την παραδοχή πως η τεχνολογία κατασκευής χαρακτηρίζεται από ένα ελάχιστο βήμα, πολλαπλάσια του οποίου λαμβάνουν όλες οι πύλες. Έτσι, μετά το πέρας του αλγορίθμου, κάνουμε διαπέραση του κυκλώματος και θέτουμε σε όλες τις πύλες, την πλησιέστερη κβαντισμένη τιμή που υποστηρίζει η τεχνολογία.

Το βήμα που έχουμε χρησιμοποιήσει για τα παραδείγματα που θα αναπτύξουμε είναι ενδεικτικά 0.000001 μέτρα χωρίς να βασίζεται σε δεδομένα πραγματικής τεχνολογίας.

3.1.7. Εκτύπωση

Καθ' όλη τη διάρκεια εκτέλεσης της, η εφαρμογή εκτυπώνει κάθε βήμα της διαδικασίας, ενώ στο τέλος εκτυπώνει τη χρήσιμη έξοδο του προγράμματος, δηλαδή τα νέα μεγέθη των πυλών που τροποποιήθηκαν κατά τη διάρκεια του αλγορίθμου. Η εκτύπωση γίνεται σε μορφή κώδικα HTML ώστε να είναι αισθητικά αρεστή στο χρήστη, ενώ γίνεται εκτεταμένη χρήση πινάκων ώστε να είναι ταυτόχρονα και ευανάγνωστη.

3.1.7.1. Πλήρης Διαδικασία Εκτέλεσης

Όπως θα φανεί και στα παραδείγματα εκτέλεσης, κατά τη διάρκεια βελτιστοποίησης, το πρόγραμμα εκτυπώνει παράλληλα και όλα τα βήματα της διαδικασίας. Η εκτύπωση αυτή παρουσιάζεται ως εξής: Για κάθε κύκλο βελτιστοποίησης δημιουργείται ένας κύριος πίνακας. Δύο κύριοι πίνακες διαδοχικών κύκλων χωρίζονται μεταξύ τους μέσω κενών γραμμών. Κάθε κύριος πίνακας αποτελείται από πολλούς υποπίνακες. Όπως έχουμε αναφέρει παραπάνω, ένας κύκλος βελτιστοποίησης βρίσκει πρώτα τα *N* χειρότερα μονοπάτια και στη συνέχεια τα βελτιστοποιεί ένα-ένα, ενώ όταν κρίνεται απαραίτητο, εφαρμόζει τον επαναληπτικό αλγόριθμο σε κάποια από αυτά. Ο κύριος πίνακας λοιπόν ενός κύκλου, χωρίζεται σε *N* υποπίνακες, έναν για κάθε ένα από τα κρισιμότερα μονοπάτια. Κάθε ένας από αυτούς τους υποπίνακες, περιγράφει μέσω παράθεσης των τιμών των μεταβλητών που συνδέονται με τη μέθοδο του logical effort τη διαδικασία βελτιστοποίησης, ενώ όπου έχουμε σπάσιμο ενός μονοπατιού σε υπομονοπάτια, γίνεται αντίστοιχη τμηματοποίηση των κόμβων στους πίνακες, στα υπομονοπάτια αυτά. Οι επαναλήψεις που λαμβάνουν χώρα κατά τη διάρκεια του επαναληπτικού αλγορίθμου εκτυπώνονται σε ενδιάμεσες θέσεις στους υποπίνακες ως γραμμές, παραθέτοντας τις τιμές των branching efforts στους εκάστοτε κόμβους πριν και μετά την έναρξη της εφαρμογής της μεθόδου του logical effort στο τρέχον μονοπάτι.

3.1.7.2. Τροποποιημένες Πύλες και Νέα Μεγέθη

Μετά το πέρας του αλγορίθμου, πρέπει να εμφανίσουμε την χρήσιμη έξοδό του. Η έξοδος αυτή, θα είναι μια λίστα από τις πύλες που άλλαξαν κατά τη διάρκειά του, σε σχέση με το αρχικό κύκλωμα εισόδου, καθώς και τα νέα μεγέθη αυτών.

Κατά τη διάρκεια εκτέλεσης της εφαρμογής, κάθε πύλη αρχίζει με ένα δείκτη αλλαγής, ο οποίος έχει τιμή 0. Στη συνέχεια, με κάθε αλλαγή μεγέθους μιας πύλης, ο δείκτης αυτός αυξάνει κατά ένα, ούτως ώστε μια θετική τιμή του να σημαίνει πως η πύλη αυτή έχει αλλάξει τιμή. Ο λόγος που είναι απαραίτητος ο αριθμός αλλαγών είναι η φάση της αναίρεσης, καθώς δεν θα γνωρίζαμε ποιες πύλες πρέπει να θεωρηθεί πως δεν είχαν αλλάξει πριν τον τελευταίο κύκλο και ποιες είχαν αλλάξει ήδη νωρίτερα. Έτσι στη φάση της αναίρεσης, απλά μειώνουμε την τιμή του δείκτη με κάθε αναίρεση της τελευταίας τροποποίησης του μεγέθους μιας πύλης.

Τέλος, εκτός από το όνομα και τα μεγέθη των πυλών, εμφανίζουμε και μια τιμή (αληθής ή ψευδής) που χρησιμεύει στην επαλήθευση της ορθότητας του αλγορίθμου μας. Η τιμή αυτή μας γνωστοποιεί εάν η πύλη που εμφανίζεται ήταν μια από τις πύλες που απαγορεύεται να αλλάξουν μέγεθος ή όχι. Η εμφάνισή τους είναι περιττή όσον αφορά στην αντικειμενική πρακτική εφαρμογή του λογισμικού όμως μας επιβεβαιώνουν πως καμία από αυτές τις πύλες δεν άλλαξε τιμή σε σχέση με την αρχική της, η οποία για τα κυκλώματα που εφαρμόσαμε τον αλγόριθμο έχει τιμή 0.000016 μέτρα. Στον Πίνακα 3.2 βλέπουμε ενδεικτικά τα αποτελέσματα της εφαρμογής για το κύκλωμα S27IC.CKT.

Updated Gate Code	Fixed Size	n-size	p-size
12	Yes	0.000008	0.000008
13	Yes	0.000008	0.000008
14	No	0.000004	0.000004
16	Yes	0.000008	0.000008
17	No	0.000005	0.000005
9	Yes	0.000008	0.000008
8	No	0.000015	0.000015
5	No	0.000015	0.000015
7	No	0.000015	0.000015

Πίνακας 3.2:

Οι κωδικοί των πυλών που τροποποιήθηκαν καθώς και τα νέα μεγέθη των τρανζίστορ τύπου-n και τύπου-p. Στη μέση βλέπουμε αν η πύλη έπρεπε να διατηρήσει το αρχικό της μέγεθος, δηλαδή τα 0.000008m.

3.2. Χειρισμός Συνδυαστικών και Ακολουθιακών Κυκλωμάτων

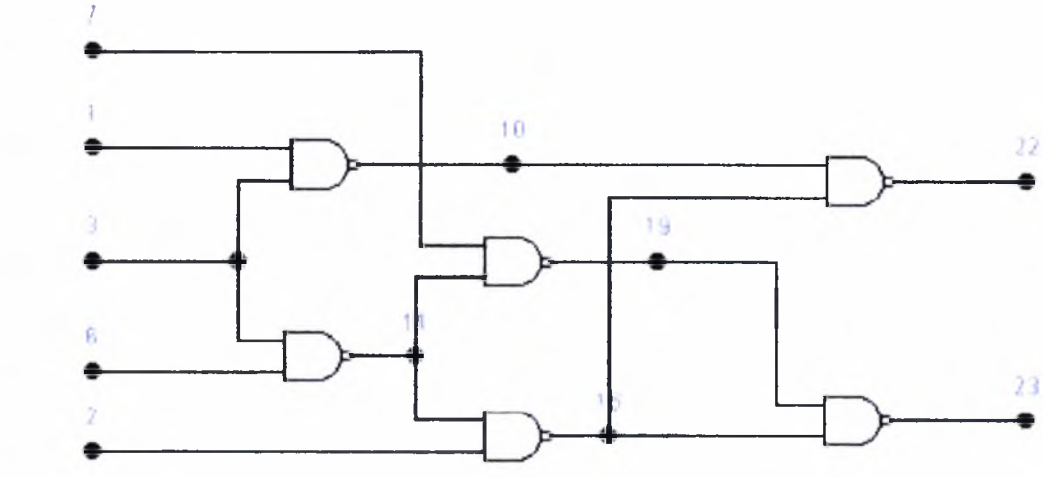
Παρακάτω θα παρατεθούν συνοπτικά τρία παραδείγματα εκτέλεσης της εφαρμογής, με είσοδο τρία διαφορετικά κυκλώματα:

1. Ένα συνδυαστικό
2. Ένα ακολουθιακό χωρίς χωρητικότητες διασύνδεσης
3. Το ίδιο ακολουθιακό με χωρητικότητες διασύνδεσης

Για το πρώτο θα δούμε ένα-ένα τα βήματα του αλγορίθμου σε μορφή πινάκων, αντίγραφα αυτών που κατασκευάζει σε κάθε εκτέλεσή της η εφαρμογή. Για τα επόμενα δύο παραδείγματα, θα δούμε μόνο επιλεκτικά τμήματα της εκτέλεσης καθώς η έξοδος και των δύο κυκλωμάτων είναι απαγορευτικά εκτεταμένη σε μέγεθος.

3.2.1. Παράδειγμα Συνδυαστικού Κυκλώματος

Το Σχήμα 3.5 βλέπουμε ένα κύκλωμα που το είδαμε εν συντομία νωρίτερα, το συνδυαστικό κύκλωμα C17.CKT, το οποίο θα προσπαθήσουμε να βελτιώσουμε μέσω της εφαρμογής μας.



Σχήμα 3.5

Παρακάτω, στους Πίνακες 3.3, 3.4 και 3.5, διακρίνονται όλα τα βήματα της εκτέλεσης του βελτιωτικού αλγορίθμου για το συγκεκριμένο κύκλωμα. Στην αρχή κάθε νέου κύκλου φαίνονται η προηγούμενη και η τρέχουσα καθυστέρησή του.

Cycle Number : 1	Number of Critical Paths selected : 1	Current Circuit Delay : 36.333333	Previous Circuit Delay : 1.#INF00

Analyzing the 1st slowest path

Optimizing the 1st slowest path

Starting new subpath cycle

Gate Code	Gate Type	Logical Effort (gl)	Electrical Effort (hl)	Parasitic Delay (pl)	Branching Effort (bl)	Gate Delay (dl=gl*hl+pl)	n-size	p-size
22	nand2	1.333333	18.750000	2.000000	-	27.000000	0.000008	0.000008
16	nand2	1.333333	2.000000	2.000000	2.000000	4.666667	0.000008	0.000008
11	input	-	-	-	-	-	-	-
Path Effort (F=G*H*B)	F-Optimal (Fopt=F^(1/2))	Logical Effort (G)	Electrical Effort (H)	Parasitic Delay (P)	Branching Effort (B)	Path Delay (D)	Optimal Path Delay (Dopt)	Best Number of Stages
66.666667	8.164966	1.777778	18.750000	4.000000	2.000000	31.666667	20.329932	3

Starting new subpath cycle (double checking)

Gate Code	Gate Type	Logical Effort (gl)	Electrical Effort (hl)	Parasitic Delay (pl)	Branching Effort (bl)	Gate Delay (dl=gl*hl+pl)	n-size	p-size
22	nand2	1.333333	6.123724	2.000000	-	10.164966	0.000024	0.000024
16	nand2	1.333333	6.123724	2.000000	2.000000	10.164966	0.000008	0.000008
11	input	-	-	-	-	-	-	-
Path Effort (F=G*H*B)	F-Optimal (Fopt=F^(1/2))	Logical Effort (G)	Electrical Effort (H)	Parasitic Delay (P)	Branching Effort (B)	Path Delay (D)	Optimal Path Delay (Dopt)	Best Number of Stages
66.666667	8.164966	1.777778	18.750000	4.000000	2.000000	20.329932	20.329932	3

Gate Code	Gate Type	n-size before Optimization	p-size before Optimization	n-size after Optimization	p-size after Optimization	Branching Effort (bl) after Optimization	Gate Cin before Optimization	Gate Cout during Optimization	Gate Cin after Optimization
22	nand2	0.000008	0.000008	0.000024	0.000024	-	0.000016	0.000300	0.000049
16	nand2	0.000008	0.000008	0.000008	0.000008	2.000000	0.000016	0.000098	0.000016
11	nand2	0.000008	0.000008	0.000008	0.000008	2.000000	0.000016	0.000032	0.000016
6	input	-	-	-	-	-	-	-	-

Path Delay before Optimization : 36.333333 | Path Delay after Optimization : 24.996598

Πίνακας 3.3

Cycle Number : 2	Number of Critical Paths selected : 1	Current Circuit Delay : 24.996598	Previous Circuit Delay : 36.333333

Analyzing the 1st slowest path

Optimizing the 1st slowest path

Starting new subpath cycle

Gate Code	Gate Type	Logical Effort (gi)	Electrical Effort (hi)	Parasitic Delay (pi)	Branching Effort (bi)	Gate Delay (di=gi*hi+pi)	n-size	p-size
22	nand2	1.333333	6.123724	2.000000	-	10.164966	0.000024	0.000024
16	nand2	1.333333	6.123724	2.000000	2.000000	10.164966	0.000008	0.000008
11	input	-	-	-	-	-	-	-

Path Effort (F=G*H*B)	F-Optimal (Fopt=F^(1/2))	Logical Effort (G)	Electrical Effort (H)	Parasitic Delay (P)	Branching Effort (B)	Path Delay (D)	Optimal Path Delay (Dopt)	Best Number of Stages
66.666667	8.164966	1.777778	18.750000	4.000000	2.000000	20.329932	20.329932	3

Starting new subpath cycle (double checking)

Gate Code	Gate Type	Logical Effort (gi)	Electrical Effort (hi)	Parasitic Delay (pi)	Branching Effort (bi)	Gate Delay (di=gi*hi+pi)	n-size	p-size
22	nand2	1.333333	6.123724	2.000000	-	10.164966	0.000024	0.000024
16	nand2	1.333333	6.123724	2.000000	2.000000	10.164966	0.000008	0.000008
11	input	-	-	-	-	-	-	-

Path Effort (F=G*H*B)	F-Optimal (Fopt=F^(1/2))	Logical Effort (G)	Electrical Effort (H)	Parasitic Delay (P)	Branching Effort (B)	Path Delay (D)	Optimal Path Delay (Dopt)	Best Number of Stages
66.666667	8.164966	1.777778	18.750000	4.000000	2.000000	20.329932	20.329932	3

Gate Code	Gate Type	n-size before Optimization	p-size before Optimization	n-size after Optimization	p-size after Optimization	Branching Effort (bi) after Optimization	Gate Cin before Optimization	Gate Cout during Optimization	Gate Cin after Optimization
22	nand2	0.000024	0.000024	0.000024	0.000024	-	0.000049	0.000300	0.000049
16	nand2	0.000008	0.000008	0.000008	0.000008	2.000000	0.000016	0.000098	0.000016
11	nand2	0.000008	0.000008	0.000008	0.000008	2.000000	0.000016	0.000032	0.000016
6	input	-	-	-	-	-	-	-	-

Path Delay before Optimization : 24.996598

Path Delay after Optimization : 24.996598

Πίνακας 3.4

Final Cycle : 3	Current Circuit Delay : 24.996598	Previous Circuit Delay : 24.996598

Previous optimization caused greater or equal overall circuit delay. Rolling back...

Circuit Initial Delay : 36.333333	Circuit Final Delay : 24.996598	Overall Delay Cutback : 31.202023%
Total Optimization Time : 0.015000 sec	Total Time Inside Loop : 0.000000 sec	Inside Loop Time/Total Time ratio : 0.000000%

The optimization will now exit...

Updated Gate Code	Fixed Size	n-size	p-size
11	Yes	0.000008	0.000008
19	Yes	0.000008	0.000008
16	Yes	0.000008	0.000008
22	No	0.000024	0.000024
23	No	0.000024	0.000024

Πίνακας 3.5

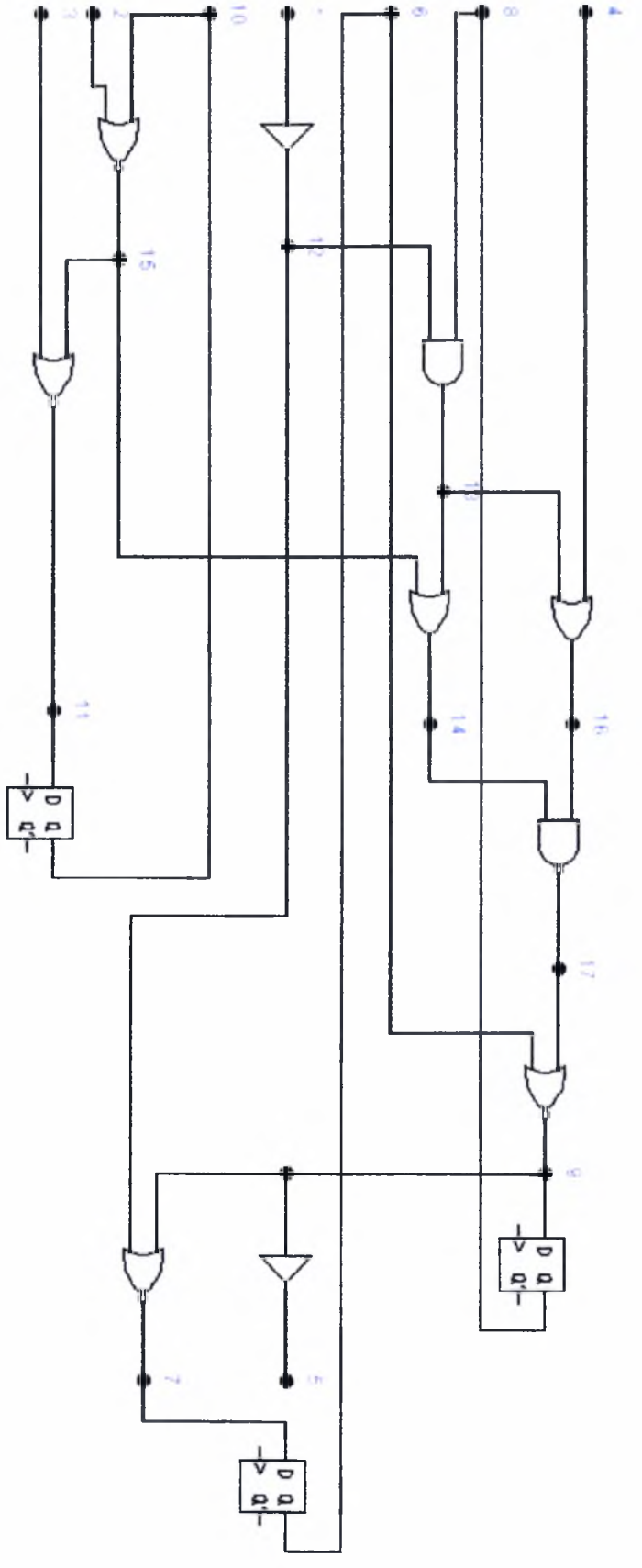
Όπως βλέπουμε, στο κύκλωμα αυτό δεν χρειάστηκε να εφαρμοστεί ο επαναληπτικός αλγόριθμος, όμως αξίζει να παρατηρήσουμε πως ο κόμβος 6 δεν συμπεριλαμβάνεται στο κρίσιμο μονοπάτι υπό βελτιστοποίηση. Ο λόγος είναι πως στο μονοπάτι δεν συμπεριλαμβάνεται η πύλη 11 καθώς είναι πύλη με σταθερό μέγεθος. Όπως βλέπουμε στον Πίνακα 3.5, η πύλη 11 φαίνεται να έχει συμπεριληφθεί τυπικά στις πύλες που άλλαξαν καθώς ανήκει σε κρίσιμο μονοπάτι, όμως ουσιαστικά βλέπουμε πως έχει διατηρήσει την αρχική τιμή της όπως και οι πύλες 16 και 19.

Στο τέλος της βελτιστοποίησης βλέπουμε την σχέση μεταξύ της αρχικής καθυστέρησης του κυκλώματος πριν την έναρξη του αλγορίθμου, και της τελικής μειωμένης καθυστέρησης. Βλέπουμε επίσης και το ποσοστό βελτίωσης της καθυστέρησης που είναι 31.202023%. Τέλος, όπως παρουσιάστηκε και νωρίτερα, βλέπουμε τις πύλες που άλλαξαν μέγεθος, καθώς και το τελικό τους μέγεθος.

3.2.2. Παράδειγμα Ακολουθιακού Κυκλώματος

Σε αυτό το κομμάτι θα δούμε αποσπάσματα της βελτιστοποίησης ενός ακολουθιακού κυκλώματος, του κυκλώματος S27.CKT που φαίνεται στο Σχήμα 3.6. Θα προσπαθήσουμε να βελτιστοποιήσουμε το συγκεκριμένο κύκλωμα δύο φορές, τη μία θεωρώντας πως δεν υπάρχουν χωρητικότητες διασύνδεσης, και τη δεύτερη

συμπεριλαμβάνοντάς τις χωρητικότητες στην περιγραφή του κυκλώματος, μέσω του αρχείου S27IC.CKT το οποίο είναι ίδιο με το S27.CKT με τη διαφορά πως ορίζονται επιπλέον χωρητικότητες σε κάθε κόμβο του κυκλώματος και όχι μονάχα στους κόμβους εξόδου του.



Σχήμα 3.6

3.2.2.1. Παράδειγμα Χωρίς Χωρητικότητες Διασύνδεσης

Σε αυτό το παράδειγμα καλείται ο επαναληπτικός αλγόριθμος λόγω της σταθερής διακλάδωσης της πύλης 16, στον κόμβο 13 του μονοπατιού. Στον Πίνακα 3.7 βλέπουμε τα υπομονοπάτια στα οποία έχει σπάσει το κρίσιμο μονοπάτι στην αρχική κατάσταση καθώς και την αρχική καθυστέρηση του κυκλώματος. Από αυτά τα υπομονοπάτια, μόνο το δεύτερο ενεργοποιεί τον επαναληπτικό αλγόριθμο. Στον Πίνακα 3.6 επισυνάπτουμε τις τιμές που λαμβάνει το branching effort στον κόμβο 13 κατά τη διαδικασία της σύγκλισης ενώ πιο κάτω, στον Πίνακα 3.8 βλέπουμε την τελική μορφή του κρίσιμου μονοπατιού μετά τη διαδικασία της σύγκλισης και της βελτιστοποίησης, καθώς και τη συνολική βελτίωση του κυκλώματος. Η καθυστέρηση του κυκλώματος μετά το τέλος του αλγορίθμου, μειώθηκε όπως φαίνεται κατά 6.877238%.

Gate Node: 13, **bi** before = **2.000000**, **bi** after = **2.473613**, **bi** diff = **0.473613**
Gate Node: 13, **bi** before = **2.473613**, **bi** after = **2.697922**, **bi** diff = **0.224310**
Gate Node: 13, **bi** before = **2.697922**, **bi** after = **2.799076**, **bi** diff = **0.101154**
Gate Node: 13, **bi** before = **2.799076**, **bi** after = **2.843769**, **bi** diff = **0.044693**
Gate Node: 13, **bi** before = **2.843769**, **bi** after = **2.863343**, **bi** diff = **0.019574**
Gate Node: 13, **bi** before = **2.863343**, **bi** after = **2.871884**, **bi** diff = **0.008541**
Gate Node: 13, **bi** before = **2.871884**, **bi** after = **2.875604**, **bi** diff = **0.003720**
Gate Node: 13, **bi** before = **2.875604**, **bi** after = **2.877224**, **bi** diff = **0.001620**
Gate Node: 13, **bi** before = **2.877224**, **bi** after = **2.877929**, **bi** diff = **0.000705**
Gate Node: 13, **bi** before = **2.877929**, **bi** after = **2.878235**, **bi** diff = **0.000307**
Gate Node: 13, **bi** before = **2.878235**, **bi** after = **2.878369**, **bi** diff = **0.000133**
Gate Node: 13, **bi** before = **2.878369**, **bi** after = **2.878427**, **bi** diff = **0.000058**
Gate Node: 13, **bi** before = **2.878427**, **bi** after = **2.878452**, **bi** diff = **0.000025**
Gate Node: 13, **bi** before = **2.878452**, **bi** after = **2.878463**, **bi** diff = **0.000011**
Gate Node: 13, **bi** before = **2.878463**, **bi** after = **2.878468**, **bi** diff = **0.000005**
Gate Node: 13, **bi** before = **2.878468**, **bi** after = **2.878470**, **bi** diff = **0.000002**
Gate Node: 13, **bi** before = **2.878470**, **bi** after = **2.878471**, **bi** diff = **0.000001**
Gate Node: 13, **bi** before = **2.878471**, **bi** after = **2.878471**, **bi** diff = **0.000000**

Πίνακας 3.6:

Σύγκλιση των τιμών του branching effort κατά τη διάρκεια του επαναληπτικού αλγορίθμου στην πύλη 13. **bi before**: η τιμή του branching effort πριν τη βελτιστοποίηση του μονοπατιού, **bi after**: η τιμή του branching effort μετά τη βελτιστοποίηση και **diff**: η απόλυτη διαφορά των δύο τιμών μέχρι αυτή να μηδενιστεί.

Όπως βλέπουμε από τη ροή της απόλυτης διαφοράς των branching effort δύο διαδοχικών επαναλήψεων, ο ρυθμός σύγκλισης είναι γεωμετρικός, με πολλαπλασιαστικό παράγοντα γύρω στο $0.47 < 1/2$.

Ο αλγόριθμος δεν κατάφερε να μειώσει την καθυστέρηση του κρίσιμου μονοπατιού τόσο ώστε να γίνει κάποιο άλλο μονοπάτι του κυκλώματος το νέο κρίσιμο μονοπάτι, επομένως η διαδικασία σταματάει στο πρώτο μονοπάτι. Το ίδιο συμβαίνει και στο τελευταίο παράδειγμα.

Cycle Number : 1	Number of Critical Paths selected : 1	Current Circuit Delay : 43.416667	Previous Circuit Delay : 1.#INF00

Analyzing the **1st** slowest path

Optimizing the **1st** slowest path

Starting new subpath cycle

Gate Code	Gate Type	Logical Effort (gi)	Electrical Effort (hi)	Parasitic Delay (pi)	Branching Effort (bi)	Gate Delay (di=gi*hi+pi)	n-size	p-size
5	NOT1	1.000000	18.750000	1.000000	-	19.750000	0.000008	0.000008
9	NOR2	1.666667	3.000000	2.000000	3.000000	7.000000	0.000008	0.000008
17	input	-	-	-	-	-	-	-
Path Effort (F=G*H*B)	**F-Optimal (Fopt=F^(1/2))**	**Logical Effort (G)**	**Electrical Effort (H)**	**Parasitic Delay (P)**	**Branching Effort (B)**	**Path Delay (D)**	**Optimal Path Delay (Dopt)**	**Best Number of Stages**
93.750000	9.682458	1.666667	18.750000	3.000000	3.000000	26.750000	22.364917	4
Gate Code	**Gate Type**	**Logical Effort (gi)**	**Electrical Effort (hi)**	**Parasitic Delay (pi)**	**Branching Effort (bi)**	**Gate Delay (di=gi*hi+pi)**	**n-size**	**p-size**
17	NAND2	1.333333	1.000000	2.000000	-	3.333333	0.000008	0.000008
14	OR2	1.666667	1.000000	3.000000	1.000000	4.666667	0.000008	0.000008
13	AND2	1.333333	2.000000	3.000000	2.000000	5.666667	0.000008	0.000008
12	input	-	-	-	-	-	-	-
Path Effort (F=G*H*B)	**F-Optimal (Fopt=F^(1/3))**	**Logical Effort (G)**	**Electrical Effort (H)**	**Parasitic Delay (P)**	**Branching Effort (B)**	**Path Delay (D)**	**Optimal Path Delay (Dopt)**	**Best Number of Stages**
5.925926	1.809612	2.962963	1.000000	8.000000	2.000000	13.666667	13.428835	1

Gate Node: 13, **bi** before = **2.000000**, **bi** after = **2.473613**, **bi** diff = **0.473613**

Πίνακας 3.7

Cycle Number : 2	Number of Critical Paths selected : 1	Current Circuit Delay : 40.430799	Previous Circuit Delay : 43.416667

Analyzing the 1st slowest path

Optimizing the 1st slowest path

Starting new subpath cycle

Starting new subpath cycle (double checking)

Gate Code	Gate Type	n-size before Optimization	p-size before Optimization	n-size after Optimization	p-size after Optimization	Branching Effort (bi) after Optimization	Gate Cin before Optimization	Gate Cout during Optimization	Gate Cin after Optimization
5	NOT1	0.000008	0.000008	0.000015	0.000015	-	0.000016	0.000300	0.000031
9	NOR2	0.000008	0.000008	0.000008	0.000008	3.000000	0.000016	0.000093	0.000016
17	NAND2	0.000008	0.000008	0.000005	0.000005	1.000000	0.000016	0.000016	0.000010
14	OR2	0.000008	0.000008	0.000004	0.000004	1.000000	0.000016	0.000010	0.000009
13	AND2	0.000008	0.000008	0.000008	0.000008	2.878472	0.000016	0.000025	0.000016
12	NOT1	0.000008	0.000008	0.000008	0.000008	2.936492	0.000016	0.000047	0.000016
1	Input	-	-	-	-	-	-	-	-
Path Delay before Optimization : 43.416667					Path Delay after Optimization : 40.430799				

Final Cycle : 3	Current Circuit Delay : 40.430799	Previous Circuit Delay : 40.430799
Previous optimization caused greater or equal overall circuit delay. Rolling back...		
Circuit Initial Delay : 43.416667	Circuit Final Delay : 40.430799	Overall Delay Cutback : 6.877238%
Total Optimization Time : 0.031000 sec	Total Time Inside Loop : 0.031000 sec	Inside Loop Time/Total Time ratio : 100.000000%
The optimization will now exit...		

Πίνακας 3.8

3.2.2.2. Παράδειγμα Με Χωρητικότητες Διασύνδεσης

Στο παράδειγμα αυτό, ο επαναληπτικός αλγόριθμος καλείται σε όλους τους κόμβους του κρίσιμου μονοπατιού λόγω των χωρητικοτήτων διασύνδεσης που τον ενεργοποιούν. Οι Πίνακες 3.9, 3.10 και 3.11 είναι ανάλογοι με αυτούς του προηγούμενου παραδείγματος, μόνο που σε αυτή την περίπτωση αξίζει να παρατηρήσουμε τις εμφωλευμένες επαναλήψεις. Ο Πίνακας 3.9 που δείχνει τις διαδοχικές τιμές των branching efforts και τις απόλυτες διαφορές τους έχει μια διαφορά σε σχέση με τον Πίνακα 3.6 του προηγούμενου παραδείγματος.

Τα branching efforts, λόγω των εμφωλευμένων επαναλήψεων δεν αναφέρονται καθ' όλη τη διάρκεια του αλγορίθμου στον ίδιο κόμβο, αλλά οι κόμβοι αναφοράς εναλλάσσονται σύμφωνα με τη ροή του αλγορίθμου από εσωτερικούς σε εξωτερικούς βρόχους και αντιστρόφως.

Όπως φαίνεται από τον Πίνακα 3.9, υπάρχει εναλλαγή μεταξύ των κόμβων 13 και 14. Η επανάληψη αρχίζει από τον κόμβο 14, μειώνοντας την απόλυτη διαφορά μεταξύ των branching efforts κάτω από το κατώφλι ελέγχου σε ένα μόνο βήμα, συνεχίζοντας τη διαδικασία στον κόμβο 13. Ο κόμβος 13 χρειάζεται αρκετά βήματα σύγκλισης διότι η απόλυτη διαφορά των branching efforts εκεί είναι αρκετά μεγάλη μετά την πρώτη επανάληψη (αυτό συνέβη διότι οι τιμές των διακλαδώσεών του 13 και συγκεκριμένα της πύλης 14, απείχαν πολύ από τις βέλτιστες της τρέχουσας επανάληψης) και δεν μπορεί να μικρύνει ικανοποιητικά σε ένα μόνο βήμα. Έτσι, όταν αλλάζει η τιμή του branching effort στον κόμβο 13, η αλλαγή αυτή είναι σχετικά απότομη και προκαλεί στην επόμενη επανάληψη την αύξηση στη διαφορά του branching effort στον κόμβο 14. Η διαδικασία αυτή επαναλαμβάνεται, μέχρις ότου η αλλαγή στον κόμβο 13 να είναι τόσο μικρή ώστε να μην επηρεάζει τον κόμβο 14. Έτσι αφού επικεντρωθούμε πια στον κόμβο 13, γίνεται σύγκλιση και ο επαναληπτικός αλγόριθμος σταματά.

Ας σημειωθεί πως στο σημείο όπου σταματά η εναλλαγή μεταξύ των κόμβων 13 και 14, δεν ισχύει ότι το branching effort στον κόμβο 14 έμεινε το ίδιο ή ότι η τιμή του άλλαξε πολύ λίγο. Αυτό που συμβαίνει στην πραγματικότητα είναι ότι πια σε κάθε επανάληψη, η αλλαγή του branching effort σε σχέση με αυτή της προηγούμενης επανάληψης είναι μικρή, όμως αθροιστικά οι συνεχείς επαναλήψεις στον κόμβο 13 έχουν σίγουρα μεταβάλλει αισθητά (σε σχέση με το κατώφλι ενεργοποίησης) την τιμή του branching effort και στον κόμβο 14. Δεν είναι όμως αυτό το κριτήριο επανεκκίνησης της μεθόδου.

Gate Node: 14, **bi** before = **1.000062**, **bi** after = **1.000085**, **bi** diff = **0.000022**
Gate Node: 13, **bi** before = **2.000062**, **bi** after = **2.473663**, **bi** diff = **0.473601**
Gate Node: 14, **bi** before = **1.000085**, **bi** after = **1.000091**, **bi** diff = **0.000006**
Gate Node: 13, **bi** before = **2.473663**, **bi** after = **2.697964**, **bi** diff = **0.224301**
Gate Node: 14, **bi** before = **1.000091**, **bi** after = **1.000094**, **bi** diff = **0.000003**
Gate Node: 13, **bi** before = **2.697964**, **bi** after = **2.799114**, **bi** diff = **0.101149**
Gate Node: 14, **bi** before = **1.000094**, **bi** after = **1.000095**, **bi** diff = **0.000001**
Gate Node: 13, **bi** before = **2.799114**, **bi** after = **2.843804**, **bi** diff = **0.044690**
Gate Node: 14, **bi** before = **1.000095**, **bi** after = **1.000095**, **bi** diff = **0.000001**
Gate Node: 13, **bi** before = **2.843804**, **bi** after = **2.863377**, **bi** diff = **0.019573**
Gate Node: 14, **bi** before = **1.000095**, **bi** after = **1.000096**, **bi** diff = **0.000000**
Gate Node: 13, **bi** before = **2.863377**, **bi** after = **2.871917**, **bi** diff = **0.008540**
Gate Node: 14, **bi** before = **1.000096**, **bi** after = **1.000096**, **bi** diff = **0.000000**
Gate Node: 13, **bi** before = **2.871917**, **bi** after = **2.875637**, **bi** diff = **0.003720**
Gate Node: 14, **bi** before = **1.000096**, **bi** after = **1.000096**, **bi** diff = **0.000000**
Gate Node: 13, **bi** before = **2.875637**, **bi** after = **2.877257**, **bi** diff = **0.001619**
Gate Node: 14, **bi** before = **1.000096**, **bi** after = **1.000096**, **bi** diff = **0.000000**
Gate Node: 13, **bi** before = **2.877257**, **bi** after = **2.877961**, **bi** diff = **0.000705**
Gate Node: 13, **bi** before = **2.877961**, **bi** after = **2.878268**, **bi** diff = **0.000307**
Gate Node: 13, **bi** before = **2.878268**, **bi** after = **2.878401**, **bi** diff = **0.000133**
Gate Node: 13, **bi** before = **2.878401**, **bi** after = **2.878459**, **bi** diff = **0.000058**
Gate Node: 13, **bi** before = **2.878459**, **bi** after = **2.878485**, **bi** diff = **0.000025**
Gate Node: 13, **bi** before = **2.878485**, **bi** after = **2.878495**, **bi** diff = **0.000011**
Gate Node: 13, **bi** before = **2.878495**, **bi** after = **2.878500**, **bi** diff = **0.000005**
Gate Node: 13, **bi** before = **2.878500**, **bi** after = **2.878502**, **bi** diff = **0.000002**
Gate Node: 13, **bi** before = **2.878502**, **bi** after = **2.878503**, **bi** diff = **0.000001**
Gate Node: 13, **bi** before = **2.878503**, **bi** after = **2.878504**, **bi** diff = **0.000000**
Gate Node: 13, **bi** before = **2.878504**, **bi** after = **2.878504**, **bi** diff = **0.000000**
Gate Node: 13, **bi** before = **2.878504**, **bi** after = **2.878504**, **bi** diff = **0.000000**
Gate Node: 13, **bi** before = **2.878504**, **bi** after = **2.878504**, **bi** diff = **0.000000**
Gate Node: 13, **bi** before = **2.878504**, **bi** after = **2.878504**, **bi** diff = **0.000000**

Πίνακας 3.9:

Σύγκλιση των τιμών του branching effort κατά τη διάρκεια του επαναληπτικού αλγορίθμου στις πύλες 13 και 14. **bi before**: η τιμή του branching effort πριν τη βελτιστοποίηση του μονοπατιού, **bi after**: η τιμή του branching effort μετά τη βελτιστοποίηση και **diff**: η απόλυτη διαφορά των δύο τιμών μέχρι αυτή να μηδενιστεί.

Η καθυστέρηση του κυκλώματος μετά το τέλος του αλγορίθμου, μειώθηκε κατά 6.877019%, που είναι λίγο μικρότερο ποσοστό σε σχέση με το παράδειγμα χωρίς χωρητικότητες διασύνδεσης ακριβώς λόγω των χωρητικοτήτων οι οποίες τείνουν να αυξάνουν την καθυστέρηση σε κάθε πύλη. Παρ' όλ' αυτά η διαφορά είναι πολύ μικρή λόγω των μικρών τιμών χωρητικοτήτων διασύνδεσης σε κάθε κόμβο σε σχέση με τις χωρητικότητες εισόδου των πυλών του κυκλώματος.

Στον Πίνακα 3.10 που ακολουθεί, βλέπουμε και μια επανάληψη που συνέβη στο υπομονοπάτι των κόμβων 5, 9 και 17, η οποία όμως έκανε σύγκλιση σε ένα μόνο βήμα χωρίς να επηρεαστεί ξανά ο κόμβος 9.

Cycle Number : 1	Number of Critical Paths selected : 1	Current Circuit Delay : 43.417104	Previous Circuit Delay : 1.#INF00

Analyzing the 1st slowest path

Optimizing the 1st slowest path

Starting new subpath cycle

Gate Code	Gate Type	Logical Effort (gl)	Electrical Effort (hl)	Parasitic Delay (pl)	Branching Effort (bl)	Gate Delay (dl=gl*hl+pl)	n-size	p-size
5	NOT1	1.000000	18.750000	1.000000	-	19.750000	0.000008	0.000008
9	NOR2	1.666667	3.000062	2.000000	3.000062	7.000104	0.000008	0.000008
17	Input	-	-	-	-	-	-	-
Path Effort (F=G*H*B)	F-Optimal (Fopt=F^(1/2))	Logical Effort (G)	Electrical Effort (H)	Parasitic Delay (P)	Branching Effort (B)	Path Delay (D)	Optimal Path Delay (Dopt)	Best Number of Stages
93.751953	9.682659	1.666667	18.750000	3.000000	3.000062	26.750104	22.365118	4

Gate Node: 9, bi before = 3.000062, bi after = 3.000032, bi diff = 0.000030

Gate Code	Gate Type	Logical Effort (gl)	Electrical Effort (hl)	Parasitic Delay (pl)	Branching Effort (bl)	Gate Delay (dl=gl*hl+pl)	n-size	p-size
17	NAND2	1.333333	1.000062	2.000000	-	3.333417	0.000008	0.000008
14	OR2	1.666667	1.000062	3.000000	1.000062	4.666771	0.000008	0.000008
13	AND2	1.333333	2.000062	3.000000	2.000062	5.666750	0.000008	0.000008
12	Input	-	-	-	-	-	-	-
Path Effort (F=G*H*B)	F-Optimal (Fopt=F^(1/3))	Logical Effort (G)	Electrical Effort (H)	Parasitic Delay (P)	Branching Effort (B)	Path Delay (D)	Optimal Path Delay (Dopt)	Best Number of Stages
5.926852	1.809706	2.962963	1.000062	8.000000	2.000188	13.666938	13.429116	1

Gate Node: 14, bi before = 1.000062, bi after = 1.000085, bi diff = 0.000022

Πίνακας 3.10

Gate Code	Gate Type	n-size before Optimization	p-size before Optimization	n-size after Optimization	p-size after Optimization	Branching Effort (bi) after Optimization	Gate Cin before Optimization	Gate Cout during Optimization	Gate Cin after Optimization
5	NOT1	0.000008	0.000008	0.000015	0.000015	-	0.000016	0.000300	0.000031
9	NOR2	0.000008	0.000008	0.000008	0.000008	3.000032	0.000016	0.000093	0.000016
17	NAND2	0.000008	0.000008	0.000005	0.000005	1.000063	0.000016	0.000016	0.000010
14	OR2	0.000008	0.000008	0.000004	0.000004	1.000096	0.000016	0.000010	0.000009
13	AND2	0.000008	0.000008	0.000008	0.000008	2.878504	0.000016	0.000025	0.000016
12	NOT1	0.000008	0.000008	0.000008	0.000008	2.936544	0.000016	0.000047	0.000016
1	Input	-	-	-	-	-	-	-	-
Path Delay before Optimization : 43.417104						Path Delay after Optimization : 40.431302			

Final Cycle : 3	Current Circuit Delay : 40.431302
Previous Circuit Delay : 40.431302	
Previous optimization caused greater or equal overall circuit delay. Rolling back...	
Circuit Initial Delay : 43.417104	Circuit Final Delay : 40.431302
Overall Delay Cutback : 6.877019%	
Total Optimization Time : 0.031000 sec	Total Time Inside Loop : 0.031000 sec
Inside Loop Time/Total Time ratio : 100.000000%	
The optimization will now exit...	

Πίνακας 3.11

3.3. Διαφορετικές Παράμετροι Εκτέλεσης

Η εφαρμογή κατά την εκτέλεσή της δέχεται το πολύ τρεις παραμέτρους. Η πρώτη είναι το όνομα του αρχείου που αναπαριστά το κύκλωμα προς επεξεργασία. Η δεύτερη είναι ο αριθμός των χειρότερων μονοπατιών *N*, που βελτιστοποιούνται σε κάθε κύκλο του αλγορίθμου. Η τρίτη παράμετρος, είναι η εκδοχή του αλγορίθμου ως προς το ποιες πύλες θεωρούνται σταθερού μεγέθους και λαμβάνει τις τιμές 0 και 1 για την κανονική εκδοχή, και για την εκδοχή της «αλλαγμένης ήδη μια φορά πύλης» αντίστοιχα. Εάν το πρόγραμμα κληθεί με 2 παραμέτρους, εφαρμόζεται η κανονική εκδοχή του αλγορίθμου, ενώ αν κληθεί με μια, εφαρμόζεται πάλι η κανονική εκδοχή, ενώ το *N* θεωρείται ίσο με 1, δηλαδή γίνεται βελτιστοποίηση μόνο του πιο κρίσιμου μονοπατιού.

Στην επόμενη παράγραφο θα παραθέσουμε στατιστικά στοιχεία για τα αποτελέσματα της εκτέλεσης της εφαρμογής πάνω σε μια πλειάδα κυκλωμάτων όπου για κάθε διαφορετικό κύκλωμα η εφαρμογή εκτελείται με 6 διαφορετικά ζεύγη παραμέτρων εισόδου.

3.4. Στατιστικά Στοιχεία Πειραματικών Αποτελεσμάτων

Τα κυκλώματα στα οποία εφαρμόσαμε τον αλγόριθμο της εφαρμογής για τις διάφορες παραμέτρους είναι ορισμένα από τα κυκλώματα στα οποία αναφέρεται η Βιβλιογραφία [10], και είναι τα εξής: *C17.CKT, C432.CKT, C499.CKT, C880.CKT, C1908.CKT, C3540.CKT, C5315.CKT, C7552.CKT, S27.ckt, S27IC.ckt, S208.ckt, S208_1.ckt, S298.ckt, S344.ckt, S349.ckt, S382.ckt, S386.ckt, S400.ckt, S420.ckt, S420_1.ckt, S444.ckt, S510.ckt, S526.ckt, S526N.ckt, S641.ckt, S713.ckt, S820.ckt, S832.ckt, S838.ckt, S838_1.ckt, S953.ckt, S1196.ckt, S1238.ckt, S1423.ckt, S5378.ckt, S9234.ckt, S9234_1.ckt.*

3.4.1. Ποσοστά Βελτίωσης

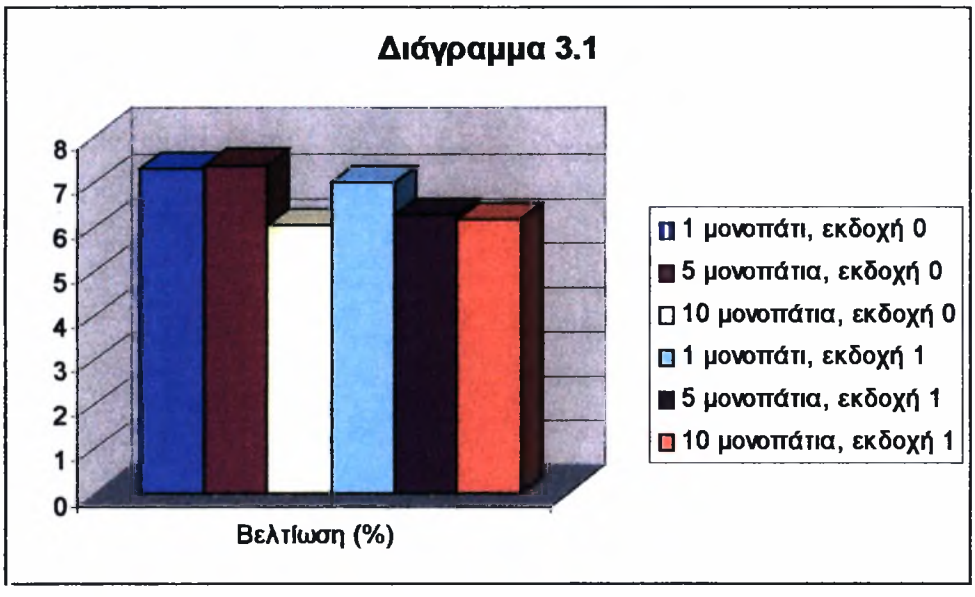
Οι διαφορετικές παράμετροι για τις οποίες εκτελέσαμε την εφαρμογή σε κάθε κύκλωμα είναι οι εξής:

1. 1 Μονοπάτι – Κανονικός Αλγόριθμος
2. 5 Μονοπάτια – Κανονικός Αλγόριθμος

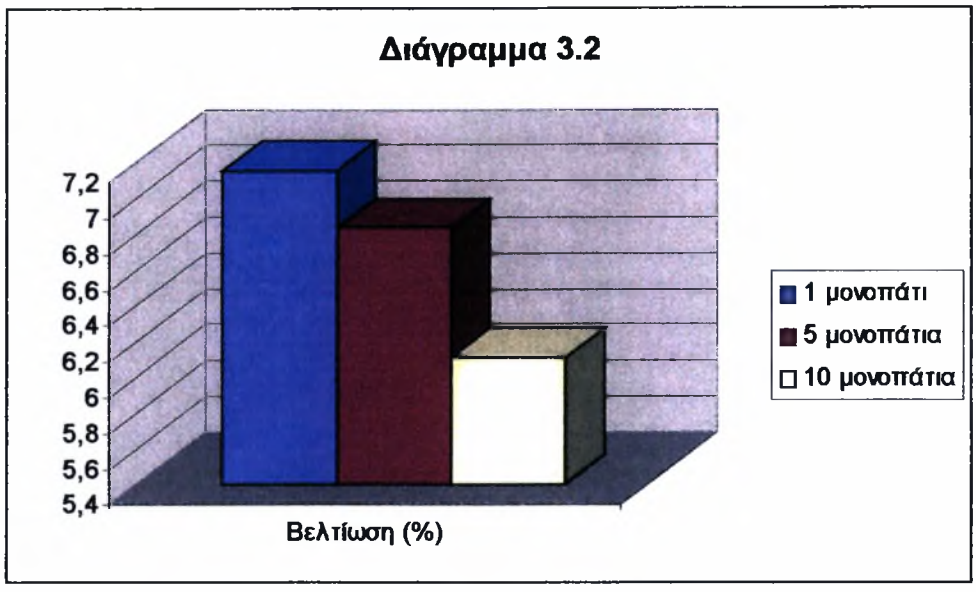
3. 10 Μονοπάτια – Κανονικός Αλγόριθμος
4. 1 Μονοπάτι – Δεύτερη Εκδοχή Αλγορίθμου
5. 5 Μονοπάτια – Δεύτερη Εκδοχή Αλγορίθμου
6. 10 Μονοπάτια – Δεύτερη Εκδοχή Αλγορίθμου

Το κριτήριο αξιολόγησης του κάθε αλγορίθμου ήταν το τελικό ποσοστό βελτίωσης του κυκλώματος. Στα Διαγράμματα 3.1, έως 3.3 έχουμε κατηγοριοποιήσει τις εκτελέσεις:

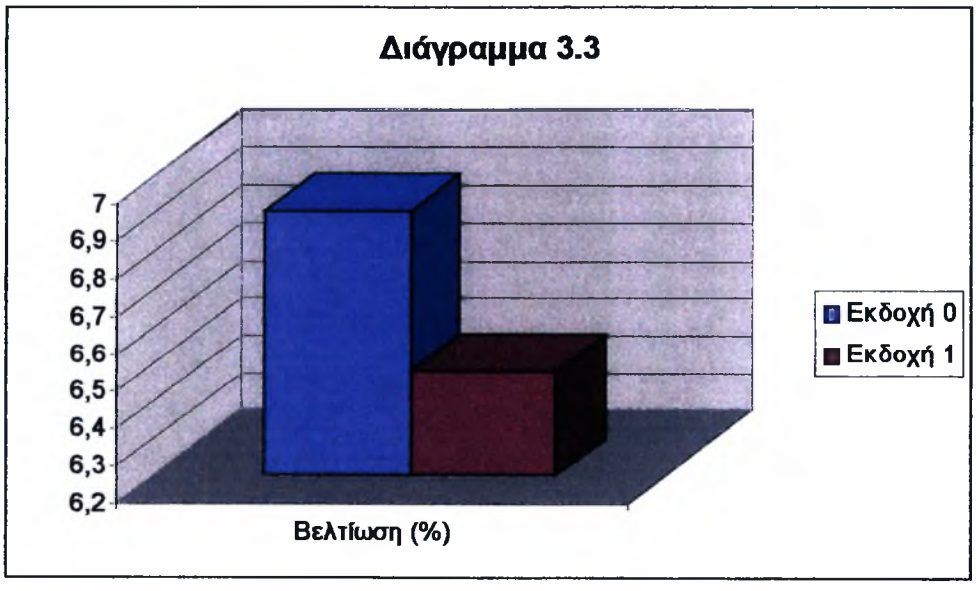
1. Ανά ζεύγος παραμέτρων (Διάγραμμα 3.1)
2. Ανά αριθμό μονοπατιών που βελτιστοποιούνται σε κάθε κύκλο (Διάγραμμα 3.2)
3. Ανά εκδοχή του αλγορίθμου (Διάγραμμα 3.3)



Μέσα ποσοστά βελτίωσης του χρόνου καθυστέρησης του παραπάνω συνόλου κυκλωμάτων μετά από εφαρμογή του αλγορίθμου για τις διαφορετικές παραμέτρους εκτέλεσης όπως φαίνονται στο διάγραμμα.



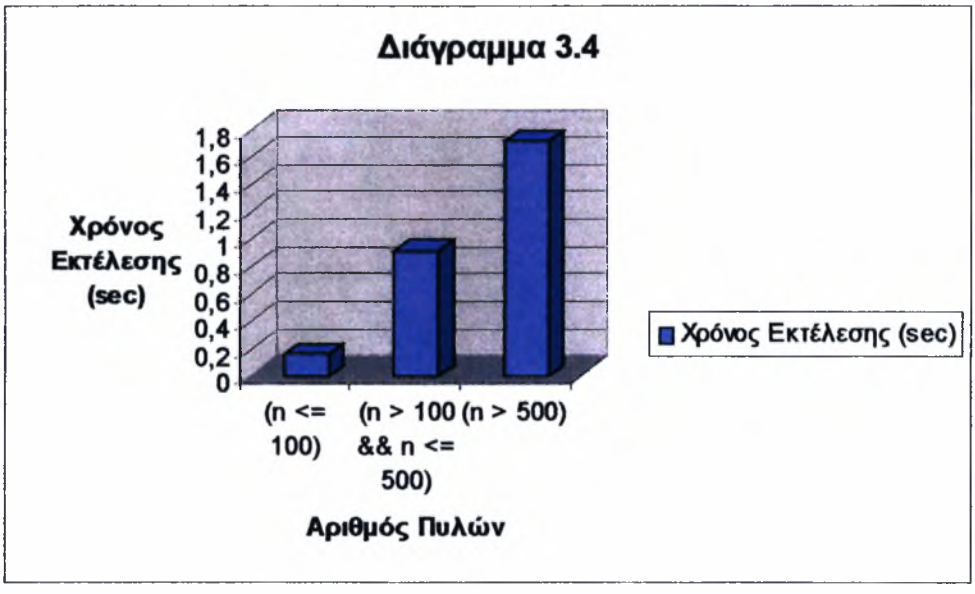
Μέσα ποσοστά βελτίωσης του χρόνου καθυστέρησης του παραπάνω συνόλου κυκλωμάτων μετά από εφαρμογή του αλγορίθμου. Η κατηγοριοποίηση έγινε ως προς τον αριθμό κρίσιμων μονοπατιών που χρησιμοποιήθηκαν.



Μέσα ποσοστά βελτίωσης του χρόνου καθυστέρησης του παραπάνω συνόλου κυκλωμάτων μετά από εφαρμογή του αλγορίθμου. Η κατηγοριοποίηση έγινε ως προς την εκδοχή του αλγορίθμου που επιλέχθηκε.

3.4.2. Χρόνοι Εκτέλεσης

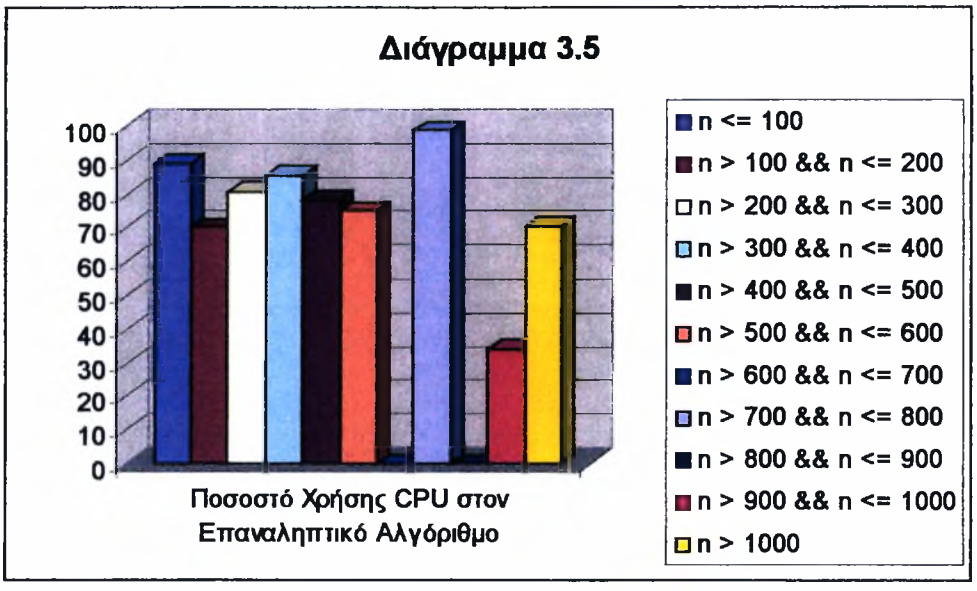
Για κάθε κύκλωμα που περνά στην εφαρμογή, μεσολαβεί ένας χρόνος εκτέλεσης μέχρι το πέρας της διαδικασίας βελτιστοποίησης και της εξαγωγής του κυκλώματος με τα τροποποιημένα μεγέθη πυλών. Ο χρόνος αυτός δεν εξαρτάται άμεσα από το μέγεθος του κυκλώματος όπως θα δούμε, όμως σίγουρα υπάρχει μια εξάρτηση του χρόνου εκτέλεσης από το μέγεθος αυτό. Στο Διάγραμμα 3.4 μπορούμε να δούμε σε γενικές γραμμές πώς αυξάνει κατά μέσο όρο ο χρόνος εκτέλεσης του προγράμματος σε σχέση με τον αριθμό πυλών του κυκλώματος εισόδου.



Μέσος χρόνος εκτέλεσης της εφαρμογής (σε sec), για κυκλώματα εισόδου n πυλών. Οι χρόνοι εκτέλεσης έχουν κατηγοριοποιηθεί ως προς τρία διαστήματα στα οποία μπορεί να ανήκει το n.

Μια άλλη παράμετρος που πρέπει να λάβουμε υπ' όψιν μας, είναι το ποσοστό κατά το οποίο τα δευτερόλεπτα CPU αφιερώνονται στις διάφορες φάσεις του προγράμματος, και συγκεκριμένα στη φάση του επαναληπτικού αλγορίθμου σε σχέση με το συνολικό χρόνο εκτέλεσης της εφαρμογής. Μέσω πειραμάτων, παρατηρήσαμε όπως φαίνεται και στο Διάγραμμα 3.5 πως καθώς μεγαλώνει ο αριθμός των πυλών των κυκλωμάτων, το ποσοστό αυτό δεν αυξάνει όπως ίσως θα ανάμενε κανείς στην περίπτωση μιας πολυπλοκότητας L^C (όπου L ο μέσος αριθμός επαναλήψεων για μια απλή σύγκλιση και C ο αριθμός των κόμβων επανάληψης στο ίδιο μονοπάτι). Το συμπέρασμα που μπορούμε να βγάλουμε λοιπόν είναι πως η πολυπλοκότητα του

επαναληπτικού αλγορίθμου είναι αρκετά μικρότερη από μια τόσο απαγορευτική πολυπλοκότητα.



Ποσοστό χρήσης CPU στον Επαναληπτικό Αλγόριθμο για κυκλώματα εισόδου n πυλών. Τα ποσοστά χρήσης έχουν κατηγοριοποιηθεί ως προς 11 διαστήματα στα οποία μπορεί να ανήκει το n.

Κεφάλαιο 4 - Συμπεράσματα

Σε αυτό το κεφάλαιο θα κάνουμε *αξιολόγηση* της εφαρμογής με βάση τα στατιστικά στοιχεία που παραθέσαμε, θα αναφέρουμε τα *προβλήματα* – αδυναμίες της, κάποιους πιθανούς τρόπους αντιμετώπισής τους, καθώς και τους ανοιχτούς δρόμους *μελλοντικής εξέλιξής* της.

4.1. Αξιολόγηση

Η εφαρμογή βελτιστοποίησης μπορεί να βελτιώσει αρκετά μεγάλο αριθμό κυκλωμάτων. Το ποσοστό βελτίωσης ποικίλει ανάλογα με το μέγεθος, την τοπολογία του κυκλώματος και τα αρχικά μεγέθη των πυλών σε αυτό. Το μέγεθος του κυκλώματος παίζει καθοριστικό ρόλο στο ποσοστό βελτίωσης και για πολύ μεγάλα κυκλώματα θα δούμε και παρακάτω πως σπανίως μπορούμε να βελτιώσουμε το συνολικό κύκλωμα ενώ θα προτείνουμε πιθανές λύσεις.

Από τα στατιστικά στοιχεία του προηγούμενου κεφαλαίου, βλέπουμε πως τα καλύτερα αποτελέσματα προκύπτουν από το συνδυασμό των 5 μονοπατιών ανά κύκλο, και της εκδοχής 0 του αλγορίθμου, δηλαδή της προεπιλεγμένης. Επίσης, για τους αριθμούς μονοπατιών 1 και 5, ο πρώτος αλγόριθμος έχει μεγαλύτερα ποσοστά βελτίωσης, ενώ για 10 μονοπάτια καλύτερη είναι η δεύτερη εκδοχή του. Τέλος, παρατηρούμε πως συνολικά κατά μέσο όρο, το 1 μονοπάτι είναι καλύτερο από τα 5 και αυτό με τη σειρά του από τα 10. Αυτό φυσικά δεν σημαίνει πως δεν υπάρχει κανένα κύκλωμα στο οποίο τα 10 μονοπάτια να είχαν καλύτερα αποτελέσματα από τα 5 και ούτω καθ' εξής. Με τον ίδιο τρόπο παρατηρούμε πως η πρώτη εκδοχή, δηλαδή η 0, έχει κατά μέσο όρο ελαφρώς καλύτερα ποσοστά βελτίωσης από την 1.

4.2. Προβλήματα

Ένα πρόβλημα που συναντήσαμε, είναι ο μεγάλος χρόνος εκτέλεσης για μεγάλα κυκλώματα, όταν αυτά ενεργοποιούν τον επαναληπτικό αλγόριθμο. Αυτό οφείλεται στην πολυπλοκότητα του αλγορίθμου όπως την αναλύσαμε παραπάνω. Ταυτόχρονα, τα κυκλώματα που αποτελούνται από σχετικά μικρό αριθμό πυλών έχουν κατά μέσο όρο αισθητά μεγαλύτερα ποσοστά βελτίωσης από εκείνα με πολύ μεγάλο αριθμό πυλών και μεγάλου μήκους μονοπάτια.

Στα μεγαλύτερα από τα κυκλώματα που εφαρμόσαμε τον αλγόριθμο είχαμε πολλές φορές μηδενική βελτιστοποίηση. Η αιτία της μικρής βελτιστοποίησης των μεγάλων κυκλωμάτων, είναι σε μεγάλο βαθμό το ότι στα μεγάλα κυκλώματα, η βελτιστοποίηση του κρίσιμου μονοπατιού, επηρεάζει αισθητά έναν πολύ μεγάλο αριθμό ξεχωριστών μονοπατιών του κυκλώματος, κάποια από τα οποία την ώρα που το κρίσιμο μονοπάτι βελτιστοποιείται, η καθυστέρησή τους αυξάνει δραματικά.

Μια λύση για αυτό το πρόβλημα, θα ήταν να σπάμε τα μεγάλα κυκλώματα σε μικρότερα υποκυκώματα και να εφαρμόζουμε τον αλγόριθμο σε εκείνα, ενώ μετά το πέρας της διαδικασίας να τα ενώνουμε στο αρχικό κύκλωμα. Η εκδοχή αυτή αφήνει ανοιχτά πολλά ζητήματα τα οποία πρέπει να μελετηθούν εκτενέστερα, κάποια από τα οποία είναι η επιλογή του μεγέθους των υποκυκλωμάτων, οι χωρητικότητες εισόδου και εξόδου των κομματιών αυτών κ.α.

Μια επιπλέον αδυναμία της παρούσας εφαρμογής, είναι η περιορισμένη ευελιξία της όσον αφορά στις χωρητικότητες διασύνδεσης. Η περιγραφή των κυκλωμάτων μέσω του SPICE στην μορφή που παρουσιάστηκαν υποθέτει πως κάθε πύλη στο κύκλωμα ακολουθείται από έναν κόμβο, και κάθε κόμβος ακολουθείται από πύλη. Η μοντελοποίηση αυτή προκαλεί περιορισμούς, καθώς ένα κύκλωμα στο οποίο έχει ολοκληρωθεί το στάδιο της τοποθέτησης, έχει σε πολλά σημεία κόμβους διακλάδωσης, οι οποίοι ακολουθούνται από νέους κόμβους διακλάδωσης κι ούτω καθ' εξής. Αυτό το πεδίο εξέλιξης μαζί με άλλα αντίστοιχα πεδία, θα συζητηθούν στην επόμενη παράγραφο.

4.3. Μελλοντικές Προοπτικές

Σε επόμενο στάδιο, η εφαρμογή θα υποστηρίζει την ανάγνωση κυκλωμάτων σε μορφή αρχείων DEF (Digital Exchange Format), στα οποία περιγράφεται πλήρως το στάδιο της τοποθέτησης ώστε το μοντέλο των χωρητικοτήτων διασύνδεσης να είναι πιο ευέλικτο.

Η αλλαγή των κριτηρίων τερματισμού της εφαρμογής θα ήταν επίσης μια μελλοντική προοπτική, καθώς αυτή τη στιγμή, με την πρώτη αποτυχία βελτίωσης σταματά η βελτιστοποίηση. Μια πιθανή βελτίωση του αλγορίθμου θα ήταν μετά την αποτυχία και την αναίρεση του τελευταίου βήματος, να προσπαθεί να εντοπίσει τα σημεία στα οποία αλλοιώνονται άλλα μονοπάτια λόγω της τελευταίας βελτιστοποίησης και να σπάσει το κρίσιμο μονοπάτι σε περισσότερα πιο μικρά

υπομονοπάτια στα οποία δεν θα συμπεριλαμβάνονται τα κομμάτια που είναι κρίσιμα για το συνολικό κύκλωμα. Το πρόβλημα της αύξησης της καθυστέρησης παράπλευρων μονοπατιών συναντάται έντονα από τον πρώτο κύκλο βελτιστοποίησης των μεγάλων κυκλωμάτων οπότε και δεν βελτιώνονται.

Μια άλλη εναλλακτική λύση που αφορά στα μεγάλα κυκλώματα θα ήταν το «σπάσιμό» τους σε πολλά μικρότερα υποκυκώματα όπως περιγράψαμε στην προηγούμενη ενότητα.

5. Βιβλιογραφία

1. http://en.wikipedia.org/wiki/Electronic_circuit
2. http://en.wikipedia.org/wiki/Very-large-scale_integration
3. http://en.wikipedia.org/wiki/Integrated_circuit_design
4. http://en.wikipedia.org/wiki/Electronic_design_automation
5. ***Logical Effort, Designing Fast CMOS Circuits*** των *Ivan Sutherland, Bob Sproull* και *David Harris*.
6. Η παρουσίαση: ***Logical Effort: Επιλογή του Μεγέθους των Πυλών*** του καθηγητή κ. *Γεώργιου Σταμούλη*.
7. http://en.wikipedia.org/wiki/Logical_effort
8. http://www.electronics-lab.com/downloads/schematic/013/ (Ιστοσελίδα που αφορά στο πρόγραμμα SPICE)
9. http://en.wikipedia.org/wiki/Combinational_logic
10. ***Incremental Circuit Simulation and Timing Analysis Techniques*** του *YUN-CHENG JU*.
11. ***Σχεδίαση Ολοκληρωμένων Κυκλωμάτων CMOS VLSI*** των *N. H. Weste* και *K. Eshraghian* σε μετάφραση – επιμέλεια των *Κ. ΠΕΚΜΕΣΤΖΗ, Δ. ΣΟΥΝΤΡΗ* και *Κ. ΓΚΟΥΤΗ*.
12. ***Σχεδίαση Ψηφιακών Συστημάτων με τη Γλώσσα VHDL*** των *Stephen Brown* και *Zvonko Vranesic* σε μετάφραση – επιμέλεια των *Νικόλαου Ι. Μάργαρη, Παύλου Χρ. Κούρου, Χρήστου Β. Τζίκα* και *Ιωάννη Πεταλά*.