

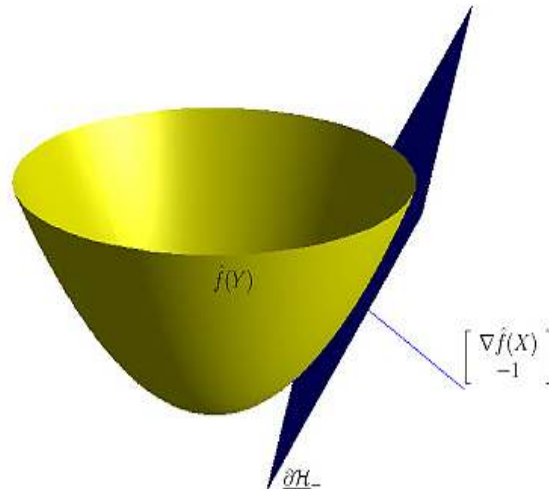
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ & ΔΙΚΤΥΩΝ

Διπλωματική Εργασία

Ερίνα – Γεωργία Αζιζάου

**Βελτιστοποίηση μεγέθους πυλών και αγωγών διασύνδεσης ολοκληρωμένων
κυκλωμάτων με μεθόδους κυρτού προγραμματισμού**



Εκπονήθηκε υπό την επίβλεψη των:
Ευμορφόπουλου Νέστορα
Σταμούλη Γεώργιου

Σεπτέμβριος 2010

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον βασικό επιβλέποντα καθηγητή της πτυχιακής αυτής εργασίας κ. Νέστορα Ευμορφόπουλο που μου έδωσε την ευκαιρία να πραγματοποιήσω αυτή την μελέτη. Η υποστήριξή του, η αμέριστη συμπαράστασή του, αλλά και οι διαρκείς και εύστοχες υποδείξεις του βοήθησαν στην έγκαιρη ολοκλήρωση αυτής της μελέτης.

Επιπρόσθετα, θα ήθελα να ευχαριστήσω τον δεύτερο επιβλέποντα καθηγητή κ. Γεώργιο Σταμούλη για τις συμβουλές, τη καθοδήγηση και την συμπαράστασή του κατά τη διάρκεια των σπουδών μου.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου που μου συμπαραστάθηκε σε όλη την διάρκεια της εκπόνησης αυτής της εργασίας.

Στην οικογένεια μου...

ΠΕΡΙΕΧΟΜΕΝΑ

Περίληψη	6
1 Εισαγωγή	7
2 Βασικές Έννοιες	9
2.1 Ορισμοί Monomial και Polynomial Συναρτήσεων	9
2.2 Ορισμός GGP προβλήματος	9
2.3 Συντελεστής Κλιμάκωσης	10
2.4 RC – μοντέλο καθυστέρησης πύλης	10
2.5 Μοντέλο καθυστέρησης αγωγών διασύνδεσης	11
3 Υλοποίηση Αλγορίθμου	15
3.1 Περιγραφή των συναρτήσεων που υλοποιήθηκαν	15
3.1.1 path_reader	15
3.1.2 p_initial_delay	15
3.1.3 parasitic_delay	15
3.1.4 total_delay	15
3.1.5 output file	16
3.2 Ροή του προγράμματος	16
3.3 Μορφή αρχείων εισόδου	17
3.4 Μορφή αρχείων εξόδου – Ερμηνεία Αποτελεσμάτων	18
3.4.1 Αρχείο GP προβλήματος	18
3.4.2 Λύση του κυρτού προγράμματος - Αποτελέσματα	19
3.5 Σταθερές του Προγράμματος – Τιμές	20
3.6 Εφαρμογή	21

3.7 Πειραματικά αποτελέσματα	27
3.7.1 Ολοκληρωμένο Κύκλωμα s1196_nominal_crit_path_a	27
3.7.2 Ολοκληρωμένο Κύκλωμα s7552_nominal_crit_path_a	29
3.7.3 Ολοκληρωμένο Κύκλωμα s880_nominal_crit_path_a	31
3.8 Solver	33
4 Συμπεράσματα	35
Παράρτημα Α	35
Βιβλιογραφία	49

ΠΕΡΙΛΗΨΗ

Η καθυστέρηση των πυλών, το οποίο εξαρτάται σε μεγάλο βαθμό από το μέγεθος τους, και η καθυστέρηση διασύνδεσης, λόγω των αγωγών διασύνδεσης ενός ολοκληρωμένου κυκλώματος είναι βασικοί παράγοντες προσδιορισμού της αποδοτικότητας ενός κυκλώματος. Η παρούσα εργασία ασχολείται με τη ταυτόχρονη βελτιστοποίηση του μεγέθους πυλών και αγωγών διασύνδεσης ολοκληρωμένων κυκλωμάτων σύμφωνα με το μοντέλο καθυστέρησης του Elmore. Το πρόβλημα διατυπώνεται ως ένα Γεωμετρικό Πρόγραμμα το οποίο στη συνέχεια μετατρέπεται σε ένα πρόβλημα κυρτού προγραμματισμού το οποίο λύνεται με χρήση μεθόδου σύγκλισης interior - point.

Ένα Γεωμετρικό Πρόγραμμα (Geometric Program – GP) είναι μαθηματικός τύπος προβλήματος βελτιστοποίησης το οποίο χαρακτηρίζεται από μια αντικειμενική συνάρτηση και από συναρτήσεις περιορισμού συγκεκριμένης μορφής. Συνεχώς αναπτύσσονται νέοι μέθοδοι επίλυσης προβλημάτων τέτοιας μορφής και είναι εξαιρετικά αποδοτικές ακόμα και για GP - προβλήματα μεγάλης κλίμακας, όπως η βελτιστοποίηση ολοκληρωμένων κυκλωμάτων. Έτσι, η βασική προσέγγιση στη μοντελοποίηση GP προβλημάτων είναι η έκφρασή τους σε μορφή γεωμετρικού προγράμματος.

Η παρούσα εργασία οργανώνεται ως εξής: Στη 1η ενότητα περιγράφεται η αναγκαιότητα ανάπτυξης και υλοποίησης μεθόδων βελτιστοποίησης του μεγέθους πυλών και αγωγών διασύνδεσης ολοκληρωμένων κυκλωμάτων καθώς γίνεται μια εισαγωγή στους ορισμούς των βασικών δομών που χρησιμοποιήθηκαν για την υλοποίηση του αλγορίθμου. Στη 2^η ενότητα ορίζονται και επεξηγούνται έννοιες και ορισμοί απαραίτητοι για την κατανόηση των μεθόδων που χρησιμοποιήθηκαν για την ανάπτυξη του αλγορίθμου και για την επίλυση του προβλήματος βελτιστοποίησης μεγέθους πυλών και αγωγών διασύνδεσης ολοκληρωμένων κυκλωμάτων. Στη 3η ενότητα περιγράφεται και αναλύεται η συνολική ροή του αλγορίθμου που υλοποιήθηκε και παρουσιάζονται τόσο μια πρακτική εφαρμογή του αλγορίθμου επίλυσης τέτοιων προβλημάτων όσο και παρουσιάζονται και αναλύονται τα πειραματικά αποτελέσματα από τις τιμές ορισμένων κυκλωμάτων. Στη 4η ενότητα παραθέτονται τα συμπεράσματα καθώς και προτείνονται μελλοντικές βελτιστοποιήσεις του αλγορίθμου που αναπτύχθηκε.

1. ΕΙΣΑΓΩΓΗ

Στη παρούσα εργασία παρουσιάζεται μια μέθοδος βελτιστοποίησης μεγέθους πυλών και αγωγών διασύνδεσης ολοκληρωμένων κυκλωμάτων η οποία βασίζεται στη διαμόρφωση του προβλήματος σε μορφή GP προγράμματος (ή GGP, δηλαδή γενικευμένου GP), η οποία μπορεί να μετατραπεί σε ένα πρόβλημα βελτιστοποίησης κυρτού προγραμματισμού (convex optimization problem) και έπειτα να λυθεί πολύ αποτελεσματικά.

Η βασική δομή ενός προβλήματος σε GP μορφή είναι η εξής:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 1, \quad i = 1, \dots, m, \\ & g_i(x) = 1, \quad i = 1, \dots, p, \end{array}$$

Όπου ως αντικειμενική συνάρτηση αναφέρεται η f_0 , οι μεταβλητές του προβλήματος είναι οι x_i και οι συναρτήσεις περιορισμού είναι οι $f_i(x) < 1$ και $g_i(x) = 1$.

Βασικό κίνητρο για μοντελοποίηση διαφόρων προβλημάτων βελτιστοποίησης σε μορφή GP αποτελεί η μεγάλη αποδοτικότητα που παρουσιάζουν οι αλγόριθμοι σύγκλισης interior-point που καλούνται να λύσουν προβλήματα αυτής της μορφής. Για παράδειγμα, ένα πρόβλημα GP με 1.000 μεταβλητές και 10.000 συναρτήσεις περιορισμού μπορεί να λυθεί σε λιγότερο από ένα λεπτό, σε ένα συμβατικό υπολογιστή. Οι αλγόριθμοι σύγκλισης interior-point ουσιαστικά δεν χρειάζονται παραμέτρους εισόδου, ούτε αρχική τιμή για τις μεταβλητές βελτιστοποίησης. Επίσης, βρίσκουν εγγυημένα την βέλτιστη λύση του προβλήματος η οποία είναι και ολικά βέλτιστη.

Το κυρίως βήμα για την επίλυση των GP προβλημάτων με αποδοτικό τρόπο είναι η μετατροπή τους σε μη γραμμικό αλλά κυρτό πρόβλημα βελτιστοποίησης. Ως κυρτό πρόβλημα βελτιστοποίησης ορίζεται ένα πρόβλημα όπου η αντικειμενική συνάρτηση και οι

συναρτήσεις περιορισμού είναι κυρτές. Για την επίλυση του προβλήματος που μελετάται στη παρούσα εργασία χρησιμοποιείται η μέθοδος Quasi - Newton II Φάσεων (Two Phase Quasi - Newton Method).

Η μετατροπή ενός προβλήματος GP σε πρόβλημα κυρτού προγραμματισμού βασίζεται σε μια λογαριθμική μετατροπή της αντικειμενικής συνάρτησης, των μεταβλητών του προβλήματος και των συναρτήσεων περιορισμού. Στη θέση των μεταβλητών x_i , χρησιμοποιείται ο λογάριθμος $y_i = \log x_i$, επομένως $x_i = e^{y_i}$. Αντί να ελαχιστοποιηθεί η αντικειμενική συνάρτηση f_0 , αρκεί να ελαχιστοποιηθεί η λογαριθμική της έκφραση $\log f_0$. Τέλος, όλες οι συναρτήσεις περιορισμών $f_i < 1$ ή $g_i = 1$ αντικαθίστανται αντίστοιχα με τις εκφράσεις $\log f_i < 0$ και $\log g_i = 0$. Έτσι, προκύπτει ένα πρόβλημα ελαχιστοποίησης κυρτού προγραμματισμού της μορφής:

$$\begin{aligned} & \text{minimize} && \log f_0(e^y) \\ & \text{subject to} && \log f_i(e^y) \leq 0, \quad i = 1, \dots, m, \\ & && \log g_i(e^y) = 0, \quad i = 1, \dots, p, \end{aligned}$$

Με μεταβλητές $y = (y_1, \dots, y_n)$. Αν και αυτή η μορφή του προβλήματος δεν φαίνεται να διαφέρει πολύ από την αρχική μορφή του GP, είναι σε κυρτή μορφή και λύνεται πολύ πιο αποδοτικά.

2. ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

2.1 Ορισμοί Monomial και Posynomial Συναρτήσεων

Έστω x_1, \dots, x_n πραγματικές θετικές μεταβλητές και $x = (x_1, \dots, x_n)$ το διάνυσμα με στοιχεία τα x_i . Μια συνάρτηση πραγματικής τιμής f με μεταβλητές x , της μορφής

$$f(x) = cx_1^{a_1} x_2^{a_2} \cdots x_n^{a_n},$$

όπου $c > 0$ και a_i ανήκει στο \mathbf{R} , ονομάζεται monomial συνάρτηση. Κάθε θετική σταθερά είναι monomial, όπως και κάθε μεταβλητή. Οι συναρτήσεις αυτής της μορφής διέπονται από τις πράξεις του πολλαπλασιασμού και της διαίρεσης.

Το άθροισμα μιας ή περισσότερων συναρτήσεων monomial αποτελούν μια συνάρτηση της μορφής

$$f(x) = \sum_{k=1}^K c_k x_1^{a_{1k}} x_2^{a_{2k}} \cdots x_n^{a_{nk}},$$

όπου $c > 0$, καλείται συνάρτηση posynomial (positive polynomial). Κάθε monomial συνάρτηση είναι και posynomial. Οι συναρτήσεις αυτής της μορφής διέπονται από τις πράξεις της πρόσθεσης, του πολλαπλασιασμού και της θετικής κλιμάκωσης.

2.2 Ορισμός GGP προγράμματος

Ένα Γενικευμένο Γεωμετρικό Πρόγραμμα (Generalized Geometric Program - GGP) είναι ένα πρόβλημα βελτιστοποίησης της μορφής:

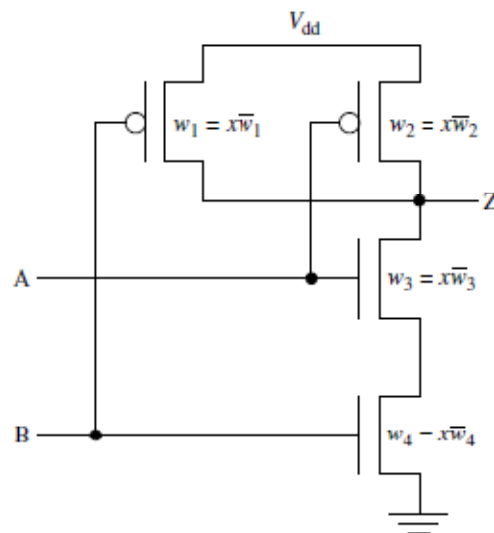
$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 1, \quad i = 1, \dots, m, \\ & && g_i(x) = 1, \quad i = 1, \dots, p, \end{aligned}$$

Όπου τα f_i είναι γενικευμένες posynomial συναρτήσεις, τα g_i είναι monomial συναρτήσεις και τα x_i είναι οι μεταβλητές βελτιστοποίησης.

Εφόσον, κάθε posynomial συνάρτηση είναι και γενικευμένη posynomial, κάθε GP πρόβλημα είναι και GGP.

2.3 Συντελεστής κλιμάκωσης

Σε κάθε πύλη i ενός ολοκληρωμένου κυκλώματος αντιστοιχίζεται ένας συντελεστής κλιμάκωσης x_i ο οποίος κλιμακώνει τα πλάτη των τρανζίστορ της πύλης. Στην εικόνα 1 φαίνεται μια πύλη NAND δύο εισόδων και του συντελεστή κλιμάκωσης x .



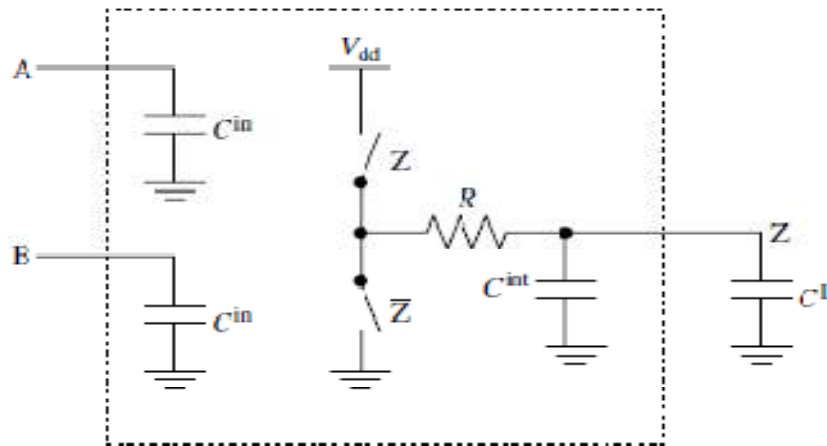
Εικόνα 1: Σχηματικό διάγραμμα πύλης NAND δυο εισόδων και του συντελεστή κλιμάκωσης

2.4 RC – μοντέλο καθυστέρησης πύλης

Κάθε πύλη χαρακτηρίζεται από τη καθυστέρηση, η οποία είναι ο χρόνος μετάβασης του σήματος εξόδου της πύλης σε νέα τιμή, αφού τα σήματα εισόδου της έχουν μεταβεί σε νέες τιμές. Το πιο απλό μοντέλο καθυστέρησης πύλης, το οποίο και χρησιμοποιείται στη πορεία, βασίζεται στο απλό RC (resistor - capacitor) κύκλωμα που φαίνεται στην εικόνα 1. Κάθε πύλη χαρακτηρίζεται από μια χωρητικότητα εισόδου C^{in} , εσωτερική χωρητικότητα C^{int} και οδηγεί χωρητικότητα C^L . Οι τιμές αυτών δίνονται από τους τύπους:

$$C_i^{in} = \bar{C}_i^{in} x_i, \quad C_i^{int} = \bar{C}_i^{int} x_i, \quad \text{και} \quad C_i^L = \sum_{j \in FO(i)} C_j^{in}, \quad \text{όπου } i$$

αντιπροσωπεύει την πύλη i . Η χωρητικότητα που οδηγεί η πύλη C^L είναι μια γραμμική συνάρτηση του συντελεστή κλιμάκωσης x , με θετικούς συντελεστές.



Εικόνα 2: RC μοντέλο μιας CMOS πύλης με δύο εισόδους A και B, και μια έξοδο Z

Επίσης, κάθε πύλη i οδηγεί αντίσταση R_i η οποία είναι αντιστρόφως ανάλογη του συντελεστή κλιμάκωσης x .

$$R_i = \bar{R}_i / x_i,$$

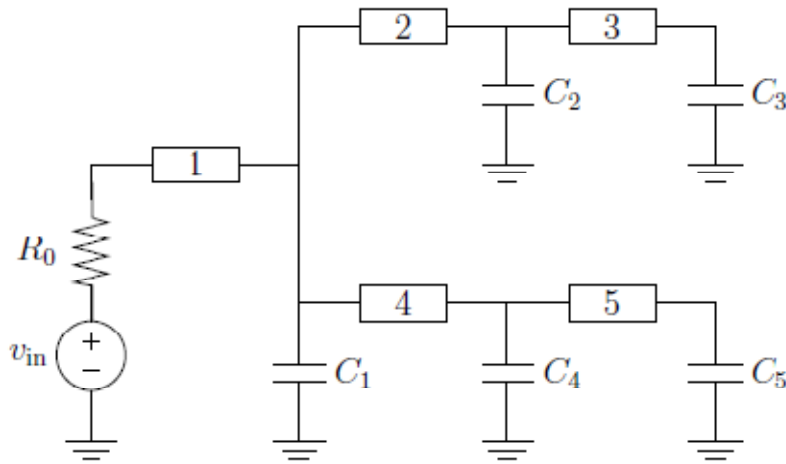
όπου \bar{R}_i είναι η αντίσταση που οδηγεί η πύλη αν $x = 1$.

2.5 Μοντέλο καθυστέρησης αγωγών διασύνδεσης

Έστω D_k η καθυστέρηση που προκαλείται από τους αγωγούς διασύνδεσης σε ένα ολοκληρωμένο κύκλωμα με n τμήματα αγωγών με πλάτη w_1, \dots, w_n . Το δίκτυο διασύνδεσης σχηματίζει ένα δένδρο, η ρίζα του οποίου οδηγείται από το σήμα εισόδου του κυκλώματος το οποίο μοντελοποιείται ως πηγή τάσης και μια σειρά από αντιστάσεις, όπως φαίνεται στην εικόνα 3. Χρησιμοποιείται το απλό π μοντέλο για κάθε αγωγό διασύνδεσης, όπως φαίνεται στην εικόνα 4. Η αντίσταση αγωγού και οι χωρητικότητες δίνονται από τον τύπο:

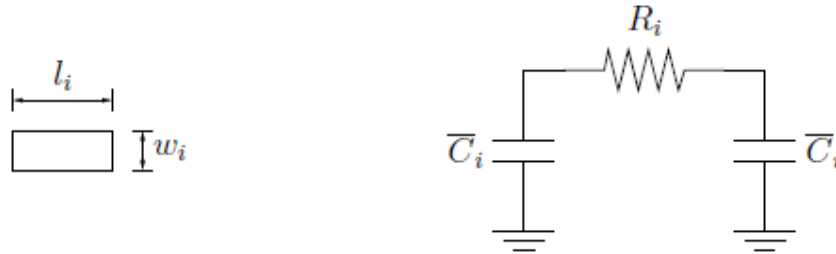
$$R_i = \alpha_i \frac{l_i}{w_i}, \quad \bar{C}_i = \beta_i l_i w_i + \gamma_i l_i,$$

όπου τα l_i και w_i είναι το μήκος και το πλάτος του αγωγού. (α, β, γ είναι θετικές σταθερές που εξαρτώνται από τα φυσικά χαρακτηριστικά του αγωγού διασύνδεσης).



Εικόνα 3: Δίκτυο διασύνδεσης μοναδικής εισόδου που οδηγεί ένα δένδρο 5 αγωγών διασύνδεσης και χωρητικότητες C_1, \dots, C_5

Οι αντιστάσεις και οι χωρητικότητες των αγωγών διασύνδεσης είναι posynomial συναρτήσεις των πλατών w_i τα οποία αποτελούν μεταβλητές βελτιστοποίησης για το πρόβλημα που μελετάται.

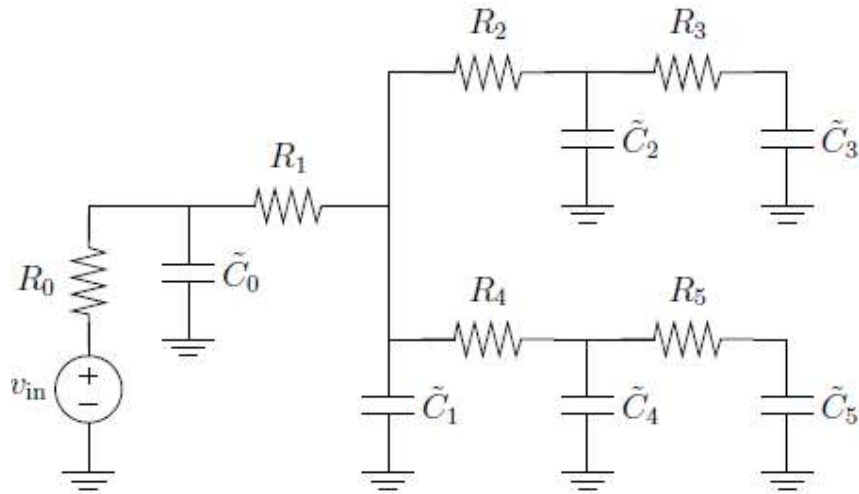


Εικόνα 4: Τμήμα αγωγού διασύνδεσης με μήκος l_i και πλάτος w_i και το αντίστοιχο π μοντέλο

Χρησιμοποιώντας το π μοντέλο σε κάθε τμήμα αγωγού διασύνδεσης, το δίκτυο διασύνδεσης μετατρέπεται σε ένα δένδρο RC (εικόνα 5) όπου

$$\begin{aligned} \tilde{C}_0 &= \bar{C}_1, \\ \tilde{C}_1 &= C_1 + \bar{C}_1 + \bar{C}_2 + \bar{C}_4, \\ \tilde{C}_2 &= C_2 + \bar{C}_2 + \bar{C}_3, \\ \tilde{C}_4 &= C_4 + \bar{C}_4 + \bar{C}_5, \\ \tilde{C}_5 &= C_5 + \bar{C}_5. \end{aligned}$$

Σε κάθε αντίσταση προκαλείται καθυστέρηση μεταβολής της τιμής όταν η πηγή τάσης αλλάζει τιμή. Για να μετρηθεί αυτή η καθυστέρηση, χρησιμοποιείται το μοντέλο καθυστέρησης Elmore.



Εικόνα 5: RC μοντέλο του δικτύου διασύνδεσης που παρουσιάζεται στην εικόνα 3

Για ένα δένδρο RC, το μοντέλο καθυστέρησης Elmore δίνεται από τον τύπο

$$D_k = \sum_{i=1}^n \tilde{C}_i \left(\sum R\text{'s upstream from capacitors } k \text{ and } i \right).$$

Η συνάρτηση καθυστέρησης D_k που προκύπτει είναι posynomial συνάρτηση των μεταβλητών πλατών αγωγών διασύνδεσης w_i .

Τέλος, εισάγονται περιορισμοί κάτω και άνω ορίου στις τιμές των πλατών w_i

$$w_i^{\min} \leq w_i \leq w_i^{\max},$$

3. ΥΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΟΥ- Solver

3.1 Περιγραφή των συναρτήσεων που υλοποιήθηκαν

3.1.1 path_reader

Η συνάρτηση path_reader αποτελεί το interface του αλγορίθμου. Δέχεται ως είσοδο ένα αρχείο δεδομένων τύπου txt, είναι υπεύθυνη για τη δημιουργία του τελικού αρχείου εξόδου, το οποίο είναι ένα αρχείο Matlab, για την αρχικοποίηση όλων των μεταβλητών του προβλήματος και για την εκτύπωση των μεγεθών των μεταβλητών βελτιστοποίησης x (μεγέθη πυλών) και w (μεγέθη αγωγών διασύνδεσης).

3.1.2 p_initial_delay

Η συνάρτηση p_initial_delay υπολογίζει για κάθε πύλη του κρίσιμου μονοπατιού ενός ολοκληρωμένου κυκλώματος τη καθυστέρηση σύμφωνα με το RC μοντέλο καθυστέρησης πυλών που περιγράφεται στην ενότητα 2.4 . Καλείται από την path_reader και αποθηκεύει τις τιμές που υπολογίζει σε πίνακα διαθέσιμο σε όλες τις συναρτήσεις που υλοποιούνται.

3.1.3 parasitic_delay

Η συνάρτηση parasitic_delay αποτελεί το ενδιάμεσο βήμα υπολογισμού των συνολικών καθυστερήσεων των πυλών και των αγωγών διασύνδεσης συναρτήσει των μεταβλητών βελτιστοποίησης.

3.1.4 total_delay

Η συνάρτηση total_delay σχηματίζει τις τελικές εκφράσεις των περιορισμών καθυστέρησης πυλών και αγωγών διασύνδεσης. Εγγράφει τα αποτελέσματα στο αρχείο εξόδου σε κατάλληλη μορφή έτσι ώστε να χρησιμοποιηθούν στη συνέχεια από τον solver. Επίσης, ορίζει τις τιμές των όλων των σταθερών των εκφράσεων καθυστέρησης.

3.1.5 output file

Το αρχείο εξόδου που δημιουργείται και συμπληρώνεται από τις παραπάνω συναρτήσεις είναι ένα εκτελέσιμο αρχείο Matlab, το οποίο καλεί τον solver σε κατάλληλη μορφή. Ορίζει κατάλληλα τις μεταβλητές βελτιστοποίησης, περιέχει τον πίνακα με τις συναρτήσεις περιορισμών, οι οποίες είναι posynomial συναρτήσεις και καλεί τη συνάρτηση επίλυσης του GP προβλήματος.

3.2 Ροή του προγράμματος

Η ροή του προγράμματος έχει ως εξής:

1. Καλείται αρχικά η path_reader η οποία ζητά από το χρήστη να εισαχθούν τα ονόματα των output file και του αρχείου εισόδου δεδομένων.
2. Η path_reader καλεί σε βρόχο την p_initial_delay ώστε να υπολογιστούν οι τιμές των χωρητικότητας των πυλών, όπως περιγράφεται παραπάνω.
3. Η path_reader καλεί σε βρόχο την parasitic_delay η οποία υπολογίζει τις ενδιάμεσες παρασιτικές καθυστερήσεις.
4. Η parasitic_delay καλεί σε κάθε της κλήση την total_delay η οποία σχηματίζει τις τελικές εκφράσεις των συναρτήσεων περιορισμών σε μορφή posynomial. Εκτυπώνει αυτές τις εκφράσεις στο αρχείο εξόδου που αναφέρθηκε παραπάνω.
5. Εκτελείται το αρχείο εξόδου Matlab που περιέχει ουσιαστικά ένα πρόβλημα GP. Ορίζονται οι μεταβλητές βελτιστοποίησης και καλείται η συνάρτηση βελτιστοποίησης gpsolve του ggplab Toolbox.
6. Η gpsolve δέχεται ως παραμέτρους την αντικειμενική συνάρτηση T (καθυστέρηση ολοκληρωμένου κυκλώματος), τον πίνακα περιορισμών, που αποτελείται από posynomial συναρτήσεις των μεταβλητών x (μέγεθος πυλών) και w (μέγεθος αγωγών διασύνδεσης) καθώς και τη συμβολοσειρά 'min' που υποδηλώνει ελαχιστοποίηση της αντικειμενικής συνάρτησης.

7. Το Toolbox που χρησιμοποιείται μετατρέπει το πρόβλημα σε μορφή κυρτού προγράμματος και καλεί τον gcnx solver, το οποίο λύνει το πρόβλημα με εφαρμογή της μεθόδου Quasi – Newton II Phase Method.

3.3 Μορφή αρχείων εισόδου που επεξεργάζεται ο αλγόριθμος

Τα αρχεία εισόδου περιέχουν τα κρίσιμα μονοπάτια διαφόρων ολοκληρωμένων κυκλωμάτων. Είναι αρχεία κειμένου txt και η μορφή των αρχείων εισόδου είναι η παρακάτω:

Όνομα Πύλης του Κρίσιμου Μονοπατιού	Τύπος Πύλης	Αριθμός Εισόδων Πύλης	Μήκος L1	Μήκος Loff	Όνομα fanout Πυλης	Τύπος fanout Πυλης	Αριθμός Εισόδων fanout Πύλης	Μήκος L2
-------------------------------------	-------------	-----------------------	----------	------------	--------------------	--------------------	------------------------------	----------

Σε κάθε γραμμή του αρχείου περιγράφεται μια πύλη του κρίσιμου μονοπατιού του ολοκληρωμένου κυκλώματος. Η πύλη μπορεί να οδηγεί ή όχι fanout πύλες (πύλες που δεν συμμετέχουν στο κρίσιμο μονοπάτι). Οι fanout πύλες αναφέρονται η μία μετά την άλλη σύμφωνα με τη μορφή που φαίνεται παρακάτω. Ως L1 αναφέρεται το μήκος του αγωγού διασύνδεσης της πύλης μέχρι τον κόμβο σύνδεσης των fanout πυλών που οδηγεί. Ως μήκος L2 αναφέρεται το υπόλοιπο μήκος του αγωγού διασύνδεσης μέχρι την επόμενη πύλη του κρίσιμου μονοπατιού.

Όνομα fanout Πύλης	Τύπος fanout Πύλης	Αριθμός Εισόδων fanout Πύλης
--------------------	--------------------	------------------------------

Σε περίπτωση που η πύλη του κρίσιμου μονοπατιού δεν οδηγεί fanout πύλες, τότε η γραμμή του αρχείου που αναφέρεται σε αυτή τη πύλη θα είναι της μορφής:

Όνομα Πύλης του Κρίσιμου Μονοπατιού	Τύπος Πύλης	Αριθμός Εισόδων Πύλης	Μήκος L1
-------------------------------------	-------------	-----------------------	----------

Ως μήκος L1 αναφέρεται το μήκος του αγωγού διασύνδεσης της πύλης με την επόμενη πύλη του κρίσιμου μονοπατιού.

Τέλος, αναφέρεται ότι οι πύλες των κρίσιμων μονοπατιών μπορεί να είναι τριών τύπων. Η κάθε πύλη μπορεί να είναι τύπου 1, δηλαδή NAND, τύπου 2, δηλαδή NOR ή τύπου 3, δηλαδή NOT. Ένα δείγμα τέτοιου αρχείου φαίνεται στην εικόνα 6.

```

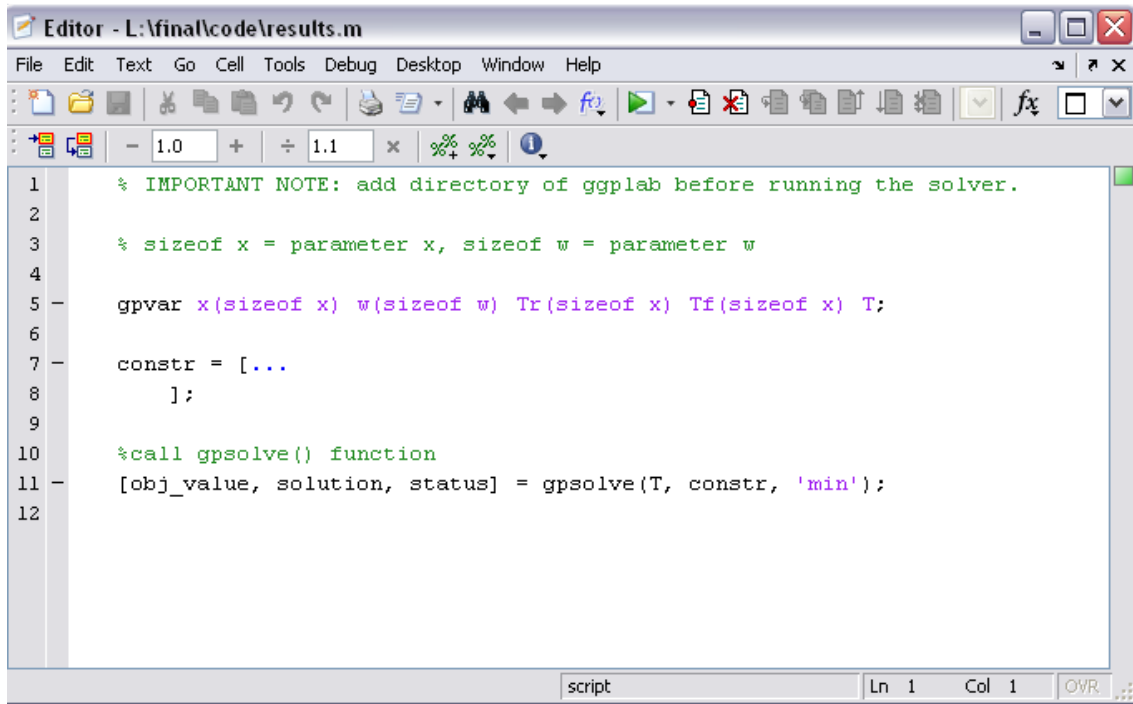
s27_nominal_crit_path_b - Notepad
File Edit Format View Help
19      3      1      82
22.1    1      2      58      39      29      2      2      36
22.2    3      1      74
25.1    2      2      74      39      24.1    2      2      75
25.2    3      1      85
27      1      2      76
30      2      2      62
20      3      1      35      89      16.1    3      1      29      2      2      98
|
  
```

Εικόνα 6: Μορφή αρχείων εισόδου

3.4 Μορφή αρχείων εξόδου – Ερμηνεία Αποτελεσμάτων

3.4.1 Αρχείο GP προβλήματος

Το αρχείο εξόδου που δημιουργείται περιέχει τη GP μορφή του προβλήματος βελτιστοποίησης μεγέθους πυλών και αγωγών διασύνδεσης ολοκληρωμένων κυκλωμάτων που προκύπτει με τη βοήθεια των συναρτήσεων που υλοποιήθηκαν και περιγράφονται παραπάνω. Η μορφή του, όπως φαίνεται στην εικόνα 7, περιγράφεται ως εξής:



```
Editor - L:\final\code\results.m
File Edit Text Go Cell Tools Debug Desktop Window Help
- 1.0 + ÷ 1.1 x %% %%
1 % IMPORTANT NOTE: add directory of ggpplab before running the solver.
2
3 % sizeof x = parameter x, sizeof w = parameter w
4
5 gpvar x(sizeof x) w(sizeof w) Tr(sizeof x) Tf(sizeof x) T;
6
7 constr = [...
8     ];
9
10 %call gpsolve() function
11 [obj_value, solution, status] = gpsolve(T, constr, 'min');
12
script Ln 1 Col 1 OVR
```

Εικόνα 7: Γενική μορφή Matlab αρχείων εξόδου

Ορίζονται οι μεταβλητές του προβλήματος βελτιστοποίησης καθώς και η αντικειμενική συνάρτηση. Ακολουθεί ο πίνακας των συναρτήσεων περιορισμού και η κλήση της συνάρτησης επίλυσης του προβλήματος.

3.4.2 Λύση του κυρτού προγράμματος - Αποτελέσματα

Η λύση του κυρτού προγράμματος προκύπτει από τον solver του πακέτου ggpplab Toolbox. Αρχικά, η αντικειμενική συνάρτηση μαζί με τις συναρτήσεις περιορισμού και της μεταβλητές βελτιστοποίησης μετατρέπονται σε μορφή κυρτού προγράμματος. Έπειτα λύνεται το κυρτό πρόγραμμα και εκτυπώνονται στο Παράθυρο Εντολών του Matlab τα εξής δεδομένα στη παρακάτω μορφή:

x: n – διάνυσμα. x είναι το βέλτιστο σημείο σύγκλισης του προβλήματος όταν το πρόβλημα είναι εφικτό ενώ περιέχει τις τιμές της τελευταίας επανάληψης της αρχικής 1ης Φάσης αν το πρόβλημα βρεθεί ανέφικτο.

status: είναι ένα αλφαριθμητικό με πιθανές τιμές 'Solved', 'Infeasible' και 'Failed'.

lamda: διάνυσμα $m + 2n$. Βέλτιστο διάνυσμα ευαισθησίας σε σχέση με τους περιορισμούς ανισοτήτων αν το πρόβλημα είναι εφικτό. Αν το πρόβλημα είναι ανέφικτο, το lamda αποτελεί πιστοποίηση της μη ύπαρξης εφικτής λύσης.

nu: $p -$ διάνυσμα. Βέλτιστο διάνυσμα ευαισθησίας σε σχέση με τους περιορισμούς ισοτήτων αν το πρόβλημα είναι εφικτό. Αν το πρόβλημα είναι ανέφικτο, το nu αποτελεί πιστοποίηση της μη ύπαρξης εφικτής λύσης.

Το τελικό αποτέλεσμα που εκτυπώνεται είναι η **τιμή της ελαχιστοποιημένης αντικειμενικής συνάρτησης**, η παράμετρος **Solution** που είναι ένας πίνακας κελιών που περιέχει τις τιμές των μεταβλητών βελτιστοποίησης και το **status** που περιγράψαμε παραπάνω.

3.5 Σταθερές του Προγράμματος – Τιμές

Οι σταθερές που χρησιμοποιούνται για την δημιουργία των εκφράσεων των συναρτήσεων περιορισμού της αντικειμενικής συνάρτησης αναφέρονται παρακάτω μαζί με τις μονάδες μέτρησης

Σταθερά	Σημασιολογία	Μονάδα Μετρησης
R = 2.5	Αντίσταση αγωγής τρανζίστορ μοναδιαίου πλάτους	kOhm*um
woff = 1	Πλάτος αγωγών διασύνδεσης fanout πυλών	um
Cff = 0.2	Πλευρική Χωρητικότητα Αγωγών διασύνδεσης	fF/um
Cpp = 0.1	Επιφανειακή Χωρητικότητα Αγωγών διασύνδεσης / μονάδα επιφάνειας	fF/um ²
Cout = 1000	Χωρητικότητα εξόδου κυκλώματος	fF
p_t = 1e-4	Αντίσταση αγωγού διασύνδεσης ανά μονάδα μήκους	kOhm
wmin = 1	Ελάχιστο πλάτος αγωγών διασύνδεσης	um

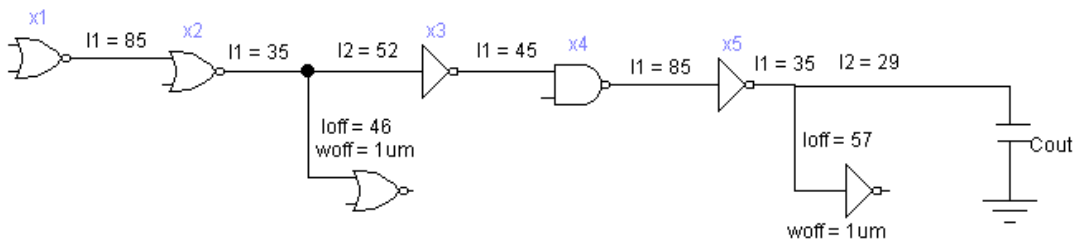
wmax = 20	Μέγιστο πλάτος αγωγών διασύνδεσης	um
xmax = 40	Μέγιστο πλάτος πυλών	um
C = 2	Εσωτερική / Εξωτερική χωρητικότητα τρανζίστορ μοναδιαίου πλάτους	fF/um ²

3.6. Πρακτική Εφαρμογή

Για μια πιο αναλυτική επισκόπηση του αλγορίθμου βελτιστοποίησης μεγέθους πυλών και αγωγών διασύνδεσης ολοκληρωμένων κυκλωμάτων ακολουθεί η εφαρμογή του πάνω σε απλουστευμένο ολοκληρωμένο κύκλωμα (κομμάτι από το κρίσιμο μονοπάτι του ολοκληρωμένου κυκλώματος *s27_nominal_crit_path_a* που φαίνεται στην εικόνα 9.). Το αρχείο των κρίσιμων μονοπατιών του συγκεκριμένου κυκλώματος φαίνεται παρακάτω:

Όνομα Πύλης του Κρίσιμου Μονοπατιού	Τύπος Πύλης	Αριθμός Εισόδων Πύλης	Μήκος L1	Μήκος Loff	Όνομα fanout Πύλης	Τύπος fanout Πύλης	Αριθμός Εισόδων fanout Πύλης	Μήκος L2
31	2	2	85					
24.1	2	2	35	46	32	2	2	52
24.2	3	1	45					
30	1	2	85					
20	3	1	35	57	16.1	3	1	29

Στην εικόνα 8 φαίνεται το κρίσιμο μονοπάτι του κυκλώματος που περιγράφεται παραπάνω.



Εικόνα 8: Κρίσιμο μονοπάτι του ολοκληρωμένου κυκλώματος του παραδείγματος

Σύμφωνα με τον αλγόριθμο, αρχικά υπολογίζονται οι χωρητικότητες κάθε πύλης ως εξής:

1η πύλη: NOR 2input $C_{in} = (2 * input + 1) * C = 5 * 2 = 10 \text{ fF/um}^2$

2η πύλη: NOR 2input $C_{in} = (2 * input + 1) * C = 5 * 2 = 10 \text{ fF/um}^2$

3η πύλη: NOT $C_{in} = 3 * C = 6 \text{ fF/um}^2$

4η πύλη: NAND 2input $C_{in} = (input + 2) * C = 4 * 2 = 8 \text{ fF/um}^2$

5η πύλη: NOT $C_{in} = 3 * C = 6 \text{ fF/um}^2$

Στη συνέχεια υπολογίζονται οι εκφράσεις καθυστέρησης των πυλών, dp Rise και dp Fall:

1η πύλη: NOR 2input $dpr = (3 * input + 2 * input(input-1)) * R * C = 10 * 2.5 * 2 = 50 \text{ ps}$

$$dpf = (3 * input + 4 * input(input-1)) * R * C = 14 * 2.5 * 2 = 70 \text{ ps}$$

2η πύλη: NOR 2input $dpr = (3 * input + 2 * input(input-1)) * R * C = 10 * 2.5 * 2 = 50 \text{ ps}$

$$dpf = (3 * input + 4 * input(input-1)) * R * C = 14 * 2.5 * 2 = 70 \text{ ps}$$

3η πύλη: NOT $dpr = dpf = 3 * R * C = 3 * 2.5 * 2 = 15 \text{ ps}$

4η πύλη: NAND 2input $dpr = (3 * input + 2 * input(input-1)) * R * C = 10 * 2.5 * 2 = 50 \text{ ps}$

$$dpf = (3 * input + input(input-1)) * R * C = 8 * 2.5 * 2 = 40 \text{ ps}$$

5η πύλη: NOT $dpr = dpf = 3 * R * C = 3 * 2.5 * 2 = 15 \text{ ps}$

Έπειτα, υπολογίζονται οι εκφράσεις συνολικής καθυστέρησης συμπεριλαμβανομένων και των καθυστερήσεων των αγωγών διασύνδεσης, σύμφωνα με τον τύπο:

Για κάθε Πύλη i:

Η πρώτη έκφραση είναι για dp rise to fall:

$$(1/Tr(i))*(dpf(i)+(R/x(i))*(Cin(i)*x(3)+Coff+Cp*11*w(i)+Cff*11+Cp*loff*woff+Cff*loff+Cp*12*w(i+1)+Cff*12)+\rho/\tau*(11*w(i))*(0.5*(Cp*11*w(i)+Cff*11)+Cp*loff*woff+Cff*loff+Coff+0.5*(Cp*12*w(i+1)+Cff*12)+\rho/\tau*(11+12)*w(i+1))*(0.5*(Cp*12*w(i+1)+Cff*12)+Cin(i+1)*x(i+1))<=1;$$

Η δεύτερη έκφραση είναι για dp fall to rise:

$$(1/Tf(i))*(dpr(i)+(R/x(i))*(Cin(i)*x(3)+Coff+Cp*11*w(i)+Cff*11+Cp*loff*woff+Cff*loff+Cp*12*w(i+1)+Cff*12)+\rho/\tau*(11*w(i))*(0.5*(Cp*11*w(i)+Cff*11)+Cp*loff*woff+Cff*loff+Coff+0.5*(Cp*12*w(i+1)+Cff*12)+\rho/\tau*(11+12)*w(i+1))*(0.5*(Cp*12*w(i+1)+Cff*12)+Cin(i+1)*x(i+1))<=1;$$

Οι περιορισμοί των μεγεθών πυλών και αγωγών διασύνδεσης πρέπει επίσης να φράσσονται, σύμφωνα με τις ανισότητες:

$$1/x(i)<=1;$$

$$x(i)/x_{max}<=1;$$

$$1/w(i)<=1;$$

$$w(i)/w_{max}<=1;$$

Τέλος προστίθενται και οι περιορισμοί καθυστέρησης:

$$Tr(\text{of last gate})/T<=1;$$

$$Tf(\text{of last gate})/T<=1;$$

File	Edit	Format	View	Help										
19	3	1	34											
22.1	1	2	64	35	29	2	2	46						
22.2	3	1	36											
22	2	2	52											
17.1	3	1	63											
17.2	3	1	76											
31	2	2	85											
24.1	2	2	35	46	32	2	2	52						
24.2	3	1	45											
27	1	2	85											
30	2	2	85											
20	3	1	35	46	16.1	3	1	29	2	2	75			

Εικόνα 9: Αρχείο κρίσιμου μονοπατιού κυκλώματος s27_nominal_crit_path_a

Για το παραπάνω παράδειγμα, αυτές οι ανισότητες περιορισμού είναι οι παρακάτω:

$$(1/\text{Tr}(1)) * (70.000 + (2.500000e+000/x(1)) * (10.000*x(2) + 0.100*85.000*w(1) + 0.200*85.000) + 0.000100000 * (85.000*w(1) * (0.5 * (0.100*85.000*w(1) + 0.200*85.000)))) \leq 1;$$

$$(1/\text{Tf}(1)) * (50.000 + (2.500000e+000/x(1)) * (10.000*x(2) + 0.100*85.000*w(1) + 0.200*85.000) + 0.000100000 * (85.000*w(1) * (0.5 * (0.100*85.000*w(1) + 0.200*85.000)))) \leq 1;$$

$$1/x(1) \leq 1;$$

$$x(1)/40.000 \leq 1;$$

$$1.000/w(1) \leq 1;$$

$$w(1)/20.000 \leq 1;$$

$$(1/\text{Tr}(2)) * (70.000 + (2.500000e+000/x(2)) * (6.000*x(3) + 10.000 + 0.100*35.000*w(2) + 0.200*35.000 + 0.100*46.000*1.000 + 0.200*46.000 + 0.100*52.000*w(3) + 0.200*52.000) + 0.000100000 * (35.000*w(2) * (0.5 * (0.100*35.000*w(2) + 0.200*35.000)))) \leq 1;$$

$$.200*35.000)+0.100*46.000*1.000+0.200*46.000+10.000+0.5*(0.100*52.000*w(3)+0.200*52.000))+0.000100000*87.000*w(3)*(0.5*(0.100*52.000*w(3)+0.200*52.000+6.000*x(3))))<=1;$$

$$(1/Tf(2))*(50.000+(2.500000e+000/x(2))*(6.000*x(3)+10.000+0.100*35.000*w(2)+0.200*35.000+0.100*46.000*1.000+0.200*46.000+0.100*52.000*w(3)+0.200*52.000))+0.000100000*(35.000*w(2))*(0.5*(0.100*35.000*w(2)+0.200*35.000)+0.100*46.000*1.000+0.200*46.000+10.000+0.5*(0.100*52.000*w(3)+0.200*52.000))+0.000100000*87.000*w(3)*(0.5*(0.100*52.000*w(3)+0.200*52.000+6.000*x(3))))<=1;$$

$$1/x(2)<=1;$$

$$x(2)/40.000<=1;$$

$$1.000/w(2)<=1;$$

$$w(2)/20.000<=1;$$

$$1.000/w(3)<=1;$$

$$w(3)/20.000<=1;$$

$$(1/Tr(3))*(15.000+(2.500000e+000/x(3))*(8.000*x(4)+0.100*45.000*w(4)+0.200*45.000))+0.000100000*(45.000*w(4)*(0.5*(0.100*45.000*w(4)+0.200*45.000))))<=1;$$

$$(1/Tf(3))*(15.000+(2.500000e+000/x(3))*(8.000*x(4)+0.100*45.000*w(4)+0.200*45.000))+0.000100000*(45.000*w(4)*(0.5*(0.100*45.000*w(4)+0.200*45.000))))<=1;$$

$$1/x(3)<=1;$$

$$x(3)/40.000<=1;$$

$$1.000/w(4)<=1;$$

$$w(4)/20.000<=1;$$

$$(1/Tr(4))*(40.000+(2.500000e+000/x(4))*(6.000*x(5)+0.100*85.000*w(5)+0.200*85.000))+0.000100000*(85.000*w(5)*(0.5*(0.100*85.000*w(5)+0.200*85.000))))<=1;$$

$$(1/Tf(4))*(50.000+(2.500000e+000/x(4))*(6.000*x(5)+0.100*85.000*w(5)+0.200*85.000))+0.000100000*(85.000*w(5)*(0.5*(0.100*85.000*w(5)+0.200*85.000))))<=1;$$

$$1/x(4) \leq 1;$$

$$x(4)/40.000 \leq 1;$$

$$1.000/w(5) \leq 1;$$

$$w(5)/20.000 \leq 1;$$

$$(1/Tr(5)) * (15.000 + (2.500000e+000/x(5)) * (1000.000 + 6.000 + 0.100 * 35.000 * w(6) + 0.200 * 35.000 + 0.100 * 57.000 * 1.000 + 0.200 * 57.000 + 0.100 * 29.000 * w(7) + 0.200 * 29.000) + 0.000100000 * 35.000 * w(6)) * ((0.5 * (0.100 * 35.000 * w(6) + 0.200 * 35.000) + 0.100 * 57.000 * 1.000 + 0.200 * 57.000 + 6.000 + 0.5 * (0.100 * 29.000 * w(7) + 0.200 * 29.000))) + 0.000100000 * 64.000 * w(7) * (0.5 * (0.100 * 29.000 * w(7) + 0.200 * 29.000) + 1000.000)) \leq 1;$$

$$(1/Tf(5)) * (15.000 + (2.500000e+000/x(5)) * (1000.000 + 6.000 + 0.100 * 35.000 * w(6) + 0.200 * 35.000 + 0.100 * 57.000 * 1.000 + 0.200 * 57.000 + 0.100 * 29.000 * w(7) + 0.200 * 29.000) + 0.000100000 * 35.000 * w(6)) * ((0.5 * (0.100 * 35.000 * w(6) + 0.200 * 35.000) + 0.100 * 57.000 * 1.000 + 0.200 * 57.000 + 6.000 + 0.5 * (0.100 * 29.000 * w(7) + 0.200 * 29.000))) + 0.000100000 * 64.000 * w(7) * (0.5 * (0.100 * 29.000 * w(7) + 0.200 * 29.000) + 1000.000)) \leq 1;$$

$$1/x(5) \leq 1;$$

$$x(5)/40.000 \leq 1;$$

$$1.000/w(6) \leq 1;$$

$$w(6)/20.000 \leq 1;$$

$$1.000/w(7) \leq 1;$$

$$w(7)/20.000 \leq 1;$$

$$Tr(5)/T \leq 1;$$

$$Tf(5)/T \leq 1;$$

Το πρόβλημα έχει λύση

$$\min T = 6.4359e+004 \text{ ps}$$

και τα βέλτιστα μεγέθη των πυλών και των αγωγών διασύνδεσης είναι:

Α. Α Πύλης Κρίσιμου μονοπατιού	Συντελεστής κλιμάκωσης x
1	6.3963
2	6.3568
3	6.4108
4	6.3724
5	40.0000

Α. Α Αγωγού διασύνδεσης Κρίσιμου μονοπατιού	Πλάτος Αγωγού Διασύνδεσης w (σε um)
1	4.4598
2	4.4668
3	4.4639
4	4.4581
5	4.4639
6	1.0000
7	1.0000

3.7 Πειραματικά αποτελέσματα

Για την εκτέλεση των πειραμάτων χρησιμοποιήθηκαν συγκεκριμένα αρχεία κρίσιμων μονοπατιών συγκεκριμένων γνωστών ολοκληρωμένων κυκλωμάτων. Τα αποτελέσματα φαίνονται στους παρακάτω πίνακες.

3.7.1 Ολοκληρωμένο Κύκλωμα *s1196_nominal_crit_path_a*

Τιμή αντικειμενικής Συνάρτησης => Συνολική καθυστέρηση κυκλώματος

$$T = 79.1596 \text{ ps (picosecond)}$$

Βέλτιστες Τιμές Μεγέθους Πυλών:

Α. Α Πύλης Κρίσιμου μονοπατιού	Συντελεστής Κλιμάκωσης x
1	6.4650
2	6.4068
3	6.4423
4	6.3968
5	6.4346
6	6.2748
7	6.4150
8	6.3331
9	6.3268
10	6.3845
11	6.3597
12	6.4210
13	6.2765
14	6.3987
15	6.3610
16	6.3582
17	6.3718
18	6.4118
19	6.3114
20	6.4068
21	6.4001
22	6.2852
23	6.4200
24	6.4184
25	40.0000

Βέλτιστες Τιμές Μεγέθους Αγωγών Διασύνδεσης:

Α.Α Πύλης Κρίσιμου Μονοπατιού	Πλάτος αγωγού διασύνδεσης w (σε μm)
1	4.4663
2	4.4646
3	4.4642
4	4.4642
5	4.4643
6	4.4590
7	4.4600
8	4.4572
9	4.4535

10	4.4677
11	4.4615
12	4.4568
13	4.4651
14	4.4668
15	4.4636
16	4.4581
17	4.4544
18	4.4661
19	4.4665
20	4.4581
21	4.4613
22	4.4597
23	4.4568
24	4.4570
25	4.4588
26	4.4531
27	4.4668
28	4.4558
29	4.4678
30	4.4503
31	4.4670
32	1.0000

3.7.2 Ολοκληρωμένο Κύκλωμα *s7552_nominal_crit_path_a*

Τιμή αντικειμενικής Συνάρτησης => Συνολική καθυστέρηση κυκλώματος

$$T = 113.1548 \text{ ps (picosecond)}$$

Βέλτιστες Τιμές Μεγέθους Πυλών:

A. A Πύλης Κρίσιμου μονοπατιού	Συντελεστής Κλιμάκωσης x
1	6.4459
2	6.3884
3	6.3887
4	6.4131
5	6.4012
6	6.4055
7	6.3778
8	6.4147
9	6.3081
10	6.4090

11	6.4005
12	6.3937
13	6.3857
14	6.3531
15	6.3699
16	6.3109
17	6.4521
18	6.2319
19	6.4362
20	6.3546
21	6.4195
22	6.3063
23	6.4137
24	6.3523
25	6.4177
26	6.2571
27	6.4291
28	6.3242
29	6.4309
30	6.2633
31	6.4071
32	6.3824
33	6.4010
34	40.0000

Βέλτιστες Τιμές Μεγέθους Αγωγών Διασύνδεσης:

Α.Α Πύλης Κρίσιμου Μονοπατιού	Πλάτος αγωγού διασύνδεσης w (σε μm)
1	4.4587
2	4.4613
3	4.4573
4	4.4583
5	4.4614
6	4.4584
7	4.4572
8	4.4494
9	4.4566
10	4.4593
11	4.4689
12	4.4638
13	4.4578
14	4.4597
15	4.4506
16	4.4687
17	4.4538

18	4.4570
19	4.4527
20	4.4703
21	4.4562
22	4.4652
23	4.4667
24	4.4680
25	4.4597
26	4.4713
27	4.4501
28	4.4605
29	4.4629
30	4.4538
31	4.4641
32	4.4501
33	4.4661
34	4.4633
35	4.4658
36	4.4649
37	4.4547
38	4.4619
39	4.4679
40	4.4628
41	4.4650
42	4.4611
43	4.4619
44	4.4610
45	4.4669
46	1.0000

3.7.3 Ολοκληρωμένο Κύκλωμα *s880_nominal_crit_path_a*

Τιμή αντικειμενικής Συνάρτησης => Συνολική καθυστέρηση κυκλώματος

$$T = 79.2021 \text{ ps (picosecond)}$$

Βέλτιστες Τιμές Μεγέθους Πυλών:

Α. Α Πύλης Κρίσιμου μονοπατιού	Συντελεστής Κλιμάκωσης x
1	6.3857
2	6.4178
3	6.3960
4	6.4044

5	6.2797
6	6.3648
7	6.3652
8	6.4063
9	6.2893
10	6.3672
11	6.3617
12	6.4207
13	6.2880
14	6.3235
15	6.3951
16	6.3722
17	6.3277
18	6.3338
19	6.3930
20	6.4088
21	6.2947
22	6.3585
23	6.4155
24	6.3327
25	6.3157
26	6.3941
27	6.3568
28	6.4090
29	40.0000

Βέλτιστες Τιμές Μεγέθους Αγωγών Διασύνδεσης:

Α.Α Πύλης Κρίσιμου Μονοπατιού	Πλάτος αγωγού διασύνδεσης w (σε μm)
1	4.4676
2	4.4636
3	4.4504
4	4.4569
5	4.4548
6	4.4655
7	4.4680
8	4.4682
9	4.4690
10	4.4681
11	4.4679
12	4.4674
13	4.4639

14	4.4605
15	4.4614
16	4.4599
17	4.4645
18	4.4666
19	4.4665
20	4.4710
21	4.4612
22	4.4648
23	4.4633
24	4.4685
25	4.4655
26	4.4670
27	4.4641
28	4.4649
29	4.4646
30	4.4638
31	4.4580
32	4.4514
33	4.4584
34	4.4699
35	4.4643
36	4.4573
37	4.4631
38	1.0000

3.8 Solver

Ο solver του πακέτου `ggplab` Toolbox που χρησιμοποιείται για την λύση του κυρτού προγράμματος βελτιστοποίησης μεγέθους πυλών και αγωγών διασύνδεσης ολοκληρωμένων κυκλωμάτων, εφαρμόζει τη μέθοδο interior point Quasi - Newton II Φάσεων. Τα βασικά βήματα μιας μεθόδου interior point Primal – Dual, περιγράφονται στο παρακάτω σχήμα:

Algorithm 11.2 *Primal-dual interior-point method.*

given x that satisfies $f_1(x) < 0, \dots, f_m(x) < 0, \lambda \succ 0, \mu > 1, \epsilon_{\text{feas}} > 0, \epsilon > 0$.

repeat

1. *Determine t . Set $t := \mu m / \hat{\eta}$.*
2. *Compute primal-dual search direction Δy_{pd} .*
3. *Line search and update.*

Determine step length $s > 0$ and set $y := y + s\Delta y_{\text{pd}}$.

until $\|r_{\text{pri}}\|_2 \leq \epsilon_{\text{feas}}, \|r_{\text{dual}}\|_2 \leq \epsilon_{\text{feas}},$ and $\hat{\eta} \leq \epsilon$.

Στη Φάση I, ο αλγόριθμος αποφασίζει αν υπάρχει εφικτή λύση για το πρόβλημα, μετατρέποντας κάθε συνάρτηση περιορισμού σε συνάρτηση ισότητας, προσθέτοντας νέες slack μεταβλητές, ως προς τις οποίες γίνεται η ελαχιστοποίηση της αντικειμενικής συνάρτησης και αν το πρόβλημα έχει λύση, τότε το αρχικό πρόβλημα είναι εφικτό και ο αλγόριθμος προχωράει στη Φάση II όπου και βρίσκονται οι βέλτιστες τιμές των μεταβλητών βελτιστοποίησης.

4. ΣΥΜΠΕΡΑΣΜΑΤΑ – ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Παρουσιάστηκε μια μέθοδος κυρτού προγραμματισμού επίλυσης του προβλήματος βελτιστοποίησης μεγέθους πυλών και αγωγών διασύνδεσης ολοκληρωμένων κυκλωμάτων.

Αρχικά, το πρόβλημα εκφράζεται σε μορφή GP ενώ έπειτα από εφαρμογή του αλγορίθμου που περιγράφεται, μετατρέπεται σε πρόβλημα κυρτού προγραμματισμού και ελαχιστοποιείται η αντικειμενική συνάρτηση καθυστέρησης με χρήση του πακέτου βελτιστοποίησης gglab Optimization Toolbox όπου εφαρμόζεται η μέθοδος Quasi - Newton II Φάσεων.

Επίσης, τα αρχεία εισόδου του αλγορίθμου που παρουσιάστηκε, επεκτάθηκαν με τυχαίες τιμές μηκών αγωγών διασύνδεσης (κλίμακας 10 – 100). Θα ήταν πολύ πιο ακριβή τα πειραματικά αποτελέσματα που παρουσιάστηκαν αν τα εργαλεία εξαγωγής των κρίσιμων μονοπατιών ολοκληρωμένων κυκλωμάτων εξήγαγαν και τις πραγματικές τιμές των μηκών των αγωγών διασύνδεσης.

Θα πρέπει να τονιστεί ότι χρησιμοποιήθηκαν το μοντέλο καθυστέρησης Elmore και ένα απλό μοντέλο καθυστέρησης των πυλών. Η ενσωμάτωση πιο ακριβών μοντέλων χρόνου θα οδηγούσε σε μια πιο ακριβή βελτιστοποίηση για οποιοδήποτε ολοκληρωμένο κύκλωμα.

Ιδέα για μελλοντική έρευνα προτείνεται η ενσωμάτωση περιορισμών ως προς την κατανάλωση δυναμικής και στατικής ισχύος η οποία καθιστά το πρόβλημα μη κυρτό. Προτείνεται η διαμόρφωση του προβλήματος σε ένα σχήμα ακολουθιακού κυρτού προγράμματος (Sequential Convex Programming).

ΠΑΡΑΡΤΗΜΑ Α

MATLAB CODE

A.1 Import Data

```
function [] = path_reader()
% this function is used to import and interpret data of critical
paths of
% integrated circuits

%constants
C = 2;      %fF/um^2
R = 2.5;    %kOhm*um

%global variables
global pinitD;      %matrix of parasitic delays of each gate
global counter;    %counter for pinitD matrix
global x;          %size of gates to optimize
global numofelementx; %number of gates
global w;          %size of wires to optimize
global numofelementw; %number of wire segments

%starting values
counter=1;
numofelementx=1;
numofelementw=1;

%open output file
fprintf('Note that output files with the same filename will be
overwritten.\n');
filename2 = input('Enter output filename: ', 's');
[fid2] = fopen(strcat(filename2, '.m'), 'w');
if (fid2 < 0)
    msg = 'error entering filename';
```

```
    disp(msg);
    return;
end

fprintf(fid2, '%s IMPORTANT NOTE: add directory of ggplab before
running the solver.\n\n');
fprintf(fid2, '%s sizeof x = parameter x, sizeof w = parameter
w\n\n');
fprintf(fid2, 'gpvar x(sizeof x) w(sizeof w) Tr(sizeof x) Tf(sizeof
x) T;\n\n');
fprintf(fid2, 'constr = [\n');

%open critical path file
filename = input('Open File: ', 's');
[fid] = fopen(filename, 'rt+');
if (fid < 0)
    msg = 'error opening file';
    disp(msg);
    return;
end

%read first line of file
curline = fgetl(fid);
numofgates = 1; %number of gates

%count critical path length
while(ischar(curline))
    numofgates=numofgates+1;
    curline = fgetl(fid);
end
numofgates=numofgates-1;

%memory preallocation
pinitD=zeros(numofgates);
x=ones(numofgates);

%calculate parasitic delay of gates
```

```
[fid] = fopen(filename, 'rt+');
curline = fgetl(fid);
while(ischar(curline))
    Readline = textscan(curline, '%f');
    LineMatrix = cell2mat(Readline);
    [m,~] = size(LineMatrix);

    %if line != EOF, call delay function
    if(m ~= 0)
        p_initial_delay(LineMatrix, C);
    end
    curline = fgetl(fid);
end

w=ones(numofelementw);

%initialize fid of file again
[fid] = fopen(filename, 'rt+');
curline = fgetl(fid);

%read file line by line
while(ischar(curline))
    Readline = textscan(curline, '%f');
    LineMatrix = cell2mat(Readline);
    [m,~] = size(LineMatrix);

    %if line != EOF, call delay function
    if(m ~= 0)
        parasitic_delay(LineMatrix, C, R, fid2);
    end
    curline = fgetl(fid);
end

fprintf(fid2, '\n];\n\n');
fprintf(fid2, '%call gpsolve() function\n');
fprintf(fid2, '[obj_value, solution, status] = gpsolve(T, constr,
'min')\n');
```

```
fprintf('size of x parameter is: %d\n', numofelementx);  
fprintf('size of w parameter is: %d\n', numofelementw);  
  
fclose(fid);  
fclose(fid2);  
  
return;  
end
```

A.2 Calculate parasitic Delays of each gate

```
function[] = p_initial_delay(LineMatrix, C)  
%p_initial_delay() calculates the parasitic delay of each gate  
  
global pinitD;  
global counter;  
  
type = LineMatrix(2);  
n = LineMatrix(3);  
  
%NAND  
if(type == 1)  
    c_in = (n + 2)*C;  
    pinitD(counter)=c_in;  
  
%NOR  
elseif(type == 2)  
    c_in = (2*n + 1)*C;  
    pinitD(counter)=c_in;  
  
else  
%NOT type == 3  
    c_in = 3*C;  
    pinitD(counter)=c_in;
```

```

end
counter=counter+1;
return;

    fprintf('delay function malfunctioned.. please check input and
try again!\n');
    return;
end

```

A.3 Calculate total parasitic delay expressions

```

function [] = parasitic_delay(LineMatrix, C, R, fid2)
% function parasitic_delay() calculates total parasitic delay for
each
% gate of critical path and calls total_delay to form final delay
expression

status=0;
[m,~] = size(LineMatrix);

if(m == 4) % there are no fanout gates
    type = LineMatrix(2);
    n = LineMatrix(3);
    l1 = LineMatrix(4);
    %NAND
    if(type == 1)
        dpr = 3*n*R*C+2*n*(n-1)*R*C;
        dpf = (3*n+n*(n-1))*R*C;
    %NOR
    elseif(type == 2)
        dpr = (3*n+2*n*(n-1))*R*C;
        dpf = (3*n+4*n*(n-1))*R*C;
    else
    %NOT type == 3
        dpr = 3*R*C;

```



```
        dpf = 3*R*C;
    end
    total_delay(fid2, status, dpr, dpf, l1, 0, 0, 0);
    return;
else
    type = LineMatrix(2);
    n = LineMatrix(3);
    l1 = LineMatrix(4);
    loff = LineMatrix(5);
    l2 = LineMatrix(m);
    status=1;
    %NAND
    if(type == 1)
        dpr = 3*n*R*C+2*n*(n-1)*R*C;
        dpf = (3*n+n*(n-1))*R*C;

    %NOR
    elseif(type == 2)
        dpr = (3*n+2*n*(n-1))*R*C;
        dpf = (3*n+4*n*(n-1))*R*C;
    else
    %NOT type == 3
        dpr = 3*R*C;
        dpf = 3*R*C;
    end
    i=6;
    coffpath = 0;
    while( i < m-1)
        offtype = LineMatrix(i+1);
        offn = LineMatrix(i+2);

    %NAND
    if(offtype == 1)
        c_in = (offn + 2)*C;

    %NOR
    elseif(offtype == 2)
```

```

        c_in = (2*offn + 1)*C;

    else
        %NOT type == 3
        c_in = 3*C;

    end

    coffpath = coffpath+c_in;
    i=i+3;
    end
    total_delay(fid2, status, dpr, dpf, l1, loff, coffpath, l2);
    return;
end

    fprintf('delay function malfunctioned.. please check input and
try again!\n');
    return;
end

```

A.4 Create output data

```

function[] = total_delay(fid2, status, dpr, dpf, l1, loff, Coff, l2)
% this function creates the final delay expressions of the given
critical path

%constants
R = 2.5;           %kOhm*um
woff = 1;         %um
Cff = 0.2;        %fF/um
Cpp = 0.1;        %fF/um^2
Cout = 1000;      %fF
p_t = 1e-4;       %kOhm
wmin = 1;         %minimum wire width
wmax = 20;        %maximum wire width
xmax = 40;        %maximum gate size

```

```

global pinitD;
global numofelementx;
global numofelementw;

if(status == 0)      %if gate does not lead any fanout gates
    if(numofelementx<size(pinitD)) %if gate is an internal one

        %dp fall to rise

fprintf(fid2, '(1/Tr(%d))*(.3f+(%d/x(%d))*(.3f*x(%d)+.3f*%
.3f*w(%d)+.3f*%
.3f)+.9f*(.3f*w(%d)*(0.5*(.3f*%
.3f*w(%d)+.3f*%
.3f)))<=1
;\n',...

numofelementx, dpf, R, numofelementx, pinitD(numofelementx+1), numofeleme
ntx+1, Cpp, l1, numofelementw, ...
    Cff, l1, p_t, l1, numofelementw, Cpp, l1, numofelementw, Cff, l1);

    %dp rise to fall

fprintf(fid2, '(1/Tf(%d))*(.3f+(%d/x(%d))*(.3f*x(%d)+.3f*%
.3f*w(%d)+.3f*%
.3f)+.9f*(.3f*w(%d)*(0.5*(.3f*%
.3f*w(%d)+.3f*%
.3f)))<=1
;\n',...

numofelementx, dpr, R, numofelementx, pinitD(numofelementx+1), numofeleme
ntx+1, Cpp, l1, numofelementw, ...
    Cff, l1, p_t, l1, numofelementw, Cpp, l1, numofelementw, Cff, l1);

fprintf(fid2, '1/x(%d)<=1;\n', numofelementx);
fprintf(fid2, 'x(%d)/.3f<=1;\n', numofelementx, xmax);
fprintf(fid2, '%.3f/w(%d)<=1;\n', wmin, numofelementw);
fprintf(fid2, 'w(%d)/.3f<=1;\n', numofelementw, wmax);

numofelementx=numofelementx+1;
numofelementw=numofelementw+1;

else %last gate of path
    %dp fall to rise

```

```

fprintf(fid2, '(1/Tr(%d))*(.3f+(%d/x(%d))*(.3f+%.3f*%.3f*w(%d)+%.3f
*%.3f)+%.9f*(%.3f*w(%d)*(0.5*(%.3f*%.3f*w(%d)+%.3f*%.3f)))<=1;\n',.
..

numofelementx,dpf,R,numofelementx,Cout,Cpp,l1,numofelementw,...
    Cff,l1,p_t,l1,numofelementw,Cpp,l1,numofelementw,Cff,l1);

    %dp rise to fall

fprintf(fid2, '(1/Tf(%d))*(.3f+(%d/x(%d))*(.3f+%.3f*%.3f*w(%d)+%.3f
*%.3f)+%.9f*(%.3f*w(%d)*(0.5*(%.3f*%.3f*w(%d)+%.3f*%.3f)))<=1;\n',.
..

numofelementx,dpr,R,numofelementx,Cout,Cpp,l1,numofelementw,...
    Cff,l1,p_t,l1,numofelementw,Cpp,l1,numofelementw,Cff,l1);

    fprintf(fid2, '1/x(%d)<=1;\n',numofelementx);
    fprintf(fid2, 'x(%d)/.3f<=1;\n',numofelementx, xmax);
    fprintf(fid2, '.3f/w(%d)<=1;\n',wmin,numofelementw);
    fprintf(fid2, 'w(%d)/.3f<=1;\n',numofelementw, wmax);
    fprintf(fid2, 'Tr(%d)/T<=1;\n', numofelementx);
    fprintf(fid2, 'Tf(%d)/T<=1;\n', numofelementx);

    end
else %status == 1 and gate leads fanout gates
    if(numofelementx<size(pinitD)) %internal gate
        %dp fall to rise
        %exp from size of gate

fprintf(fid2, '(1/Tr(%d))*(.3f+(%d/x(%d))*(.3f*x(%d)+%.3f+%.3f*%.3f
*w(%d)+%.3f*%.3f+%.3f*%.3f*%.3f+%.3f*%.3f+%.3f*%.3f*w(%d)+%.3f*%.3f)
',...

numofelementx,dpf,R,numofelementx,pinitD(numofelementx+1),numofeleme
ntx+1,Coff,Cpp,l1,numofelementw,...

```

```

Cff, l1, Cpp, loff, woff, Cff, loff, Cpp, l2, numofelementw+1, Cff, l2);

    %exp from size of wire

fprintf(fid2, '+%.9f*(%.3f*w(%d))*(0.5*(%.3f*%.3f*w(%d)+%.3f*%.3f)+%.
3f*%.3f*%.3f+%.3f*%.3f+%.3f+0.5*(%.3f*%.3f*w(%d)+%.3f*%.3f))', ...

p_t, l1, numofelementw, Cpp, l1, numofelementw, Cff, l1, Cpp, loff, woff, Cff, l
off, Coff, Cpp, l2, numofelementw+1, Cff, l2);

    %continues...

fprintf(fid2, '+%.9f*%.3f*w(%d)*(0.5*(%.3f*%.3f*w(%d)+%.3f*%.3f+%.3f*
x(%d)))<=1;\n', ...

p_t, l1+l2, numofelementw+1, Cpp, l2, numofelementw+1, Cff, l2, pinitD(numof
elementx+1), numofelementx+1);

    %dp rise to fall
    %exp from size of gate

fprintf(fid2, '(1/Tf(%d))*(%.3f+(%d/x(%d))*(%.3f*x(%d)+%.3f+%.3f*%.3f
*w(%d)+%.3f*%.3f+%.3f*%.3f*%.3f+%.3f*%.3f+%.3f*%.3f*w(%d)+%.3f*%.3f)
', ...

numofelementx, dpr, R, numofelementx, pinitD(numofelementx+1), numofeleme
ntx+1, Coff, Cpp, l1, numofelementw, ...

Cff, l1, Cpp, loff, woff, Cff, loff, Cpp, l2, numofelementw+1, Cff, l2);

    %exp from size of wire

fprintf(fid2, '+%.9f*(%.3f*w(%d))*(0.5*(%.3f*%.3f*w(%d)+%.3f*%.3f)+%.
3f*%.3f*%.3f+%.3f*%.3f+%.3f+0.5*(%.3f*%.3f*w(%d)+%.3f*%.3f))', ...

```

```
p_t,l1,numofelementw,Cpp,l1,numofelementw,Cff,l1,Cpp,loff,woff,Cff,l
off,Coff,Cpp,l2,numofelementw+1,Cff,l2);
```

```
    %continues...
```

```
fprintf(fid2, '+%.9f*%.3f*w(%d) * (0.5*(%.3f*%.3f*w(%d)+%.3f*%.3f+%.3f*
x(%d))) <=1;\n', ...
```

```
p_t,l1+l2,numofelementw+1,Cpp,l2,numofelementw+1,Cff,l2,pinitD(numof
elementx+1),numofelementx+1);
```

```
    fprintf(fid2, '1/x(%d) <=1;\n', numofelementx);
```

```
    fprintf(fid2, 'x(%d)/%.3f <=1;\n', numofelementx, xmax);
```

```
    fprintf(fid2, '%.3f/w(%d) <=1;\n', wmin, numofelementw);
```

```
    fprintf(fid2, 'w(%d)/%.3f <=1;\n', numofelementw, wmax);
```

```
    fprintf(fid2, '%.3f/w(%d) <=1;\n', wmin, numofelementw+1);
```

```
    fprintf(fid2, 'w(%d)/%.3f <=1;\n', numofelementw+1, wmax);
```

```
    numofelementx=numofelementx+1;
```

```
    numofelementw=numofelementw+2;
```

```
    else    %gate is last of path
```

```
    %dp fall to rise
```

```
fprintf(fid2, ' (1/Tr(%d)) * (%.3f+(%d/x(%d)) * (%.3f+%.3f+%.3f*%.3f*w(%d)
+%.3f*%.3f+%.3f*%.3f*%.3f*%.3f+%.3f*%.3f+%.3f*%.3f*w(%d)+%.3f*%.3f) +', ...
```

```
numofelementx, dpf, R, numofelementx, Cout, Coff, Cpp, l1, numofelementw, Cff
, l1, Cpp, loff, woff, Cff, loff, Cpp, l2, numofelementw+1, Cff, l2);
```

```
    %wire size exp
```

```
fprintf(fid2, '+%.9f*%.3f*w(%d) * ( (0.5*(%.3f*%.3f*w(%d)+%.3f*%.3f)+%.3f
*%.3f*%.3f+%.3f*%.3f+%.3f+0.5*(%.3f*%.3f*w(%d)+%.3f*%.3f) ) ) +', ...
```

```
p_t,l1,numofelementw,Cpp,l1,numofelementw,Cff,l1,Cpp,loff,woff,Cff,loff,Coff,Cpp,l2,numofelementw+1,Cff,l2);
```

```
%continues...
```

```
fprintf(fid2, '%.9f*%.3f*w(%d) * (0.5* (%.3f*%.3f*w(%d)+%.3f*%.3f)+%.3f) )<=1;\n', ...
```

```
p_t,l1+l2,numofelementw+1,Cpp,l2,numofelementw+1,Cff,l2,Cout);
```

```
%dp rise to fall
```

```
fprintf(fid2, ' (1/Tf(%d)) * (%.3f+(%d/x(%d)) * (%.3f+%.3f+%.3f*%.3f*w(%d) +%.3f*%.3f+%.3f*%.3f*%.3f+%.3f*%.3f+%.3f*%.3f*w(%d)+%.3f*%.3f) +', ...
```

```
numofelementx,dpr,R,numofelementx,Cout,Coff,Cpp,l1,numofelementw,Cff,l1,Cpp,loff,woff,Cff,loff,Cpp,l2,numofelementw+1,Cff,l2);
```

```
%wire size exp
```

```
fprintf(fid2, '%.9f*%.3f*w(%d) * ( (0.5* (%.3f*%.3f*w(%d)+%.3f*%.3f)+%.3f*%.3f*%.3f+%.3f*%.3f+%.3f+0.5* (%.3f*%.3f*w(%d)+%.3f*%.3f) ) +', ...
```

```
p_t,l1,numofelementw,Cpp,l1,numofelementw,Cff,l1,Cpp,loff,woff,Cff,loff,Coff,Cpp,l2,numofelementw+1,Cff,l2);
```

```
%continues...
```

```
fprintf(fid2, '%.9f+%.3f*w(%d) * (0.5* (%.3f*%.3f*w(%d)+%.3f*%.3f)+%.3f) )<=1;\n', ...
```

```
p_t,l1+l2,numofelementw+1,Cpp,l2,numofelementw+1,Cff,l2,Cout);
```

```
fprintf(fid2, '1/x(%d)<=1;\n', numofelementx);
```

```
fprintf(fid2, 'x(%d)/%.3f<=1;\n', numofelementx, xmax);
```

```
fprintf(fid2, '%.3f/w(%d)<=1;\n', wmin, numofelementw);
```

```
fprintf(fid2, 'w(%d)/%.3f<=1;\n', numofelementw, wmax);
```

```
fprintf(fid2, '%.3f/w(%d)<=1;\n', wmin, numofelementw+1);  
fprintf(fid2, 'w(%d)/%.3f<=1;\n', numofelementw+1, wmax);  
fprintf(fid2, 'Tr(%d)/T<=1;\n', numofelementx);  
fprintf(fid2, 'Tf(%d)/T<=1;\n', numofelementx);  
  
end  
end  
return;  
end
```

A.5 Final Solver (Γενική μορφή αρχείου)

```
% IMPORTANT NOTE: add directory of ggplab before running the solver.  
  
% sizeof x = parameter x, sizeof w = parameter w  
  
gpvar x(sizeof x) w(sizeof w) Tr(sizeof x) Tf(sizeof x) T;  
  
constr = ['inequalities and equalities constraints'];  
  
%call gpsolve() function  
[obj_value, solution, status] = gpsolve(T, constr, 'min')
```


ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] **Convex Optimization** - Stephen Boyd of Department of Electrical Engineering Stanford University and Lieven Vandenbergh of Electrical Engineering Department University of California – Cambridge University Press 2006

[2] **Digital Circuit Optimization via Geometric Programming** – Stephen Boyd, Seung-Jean Kim, Dinesh D.Patil, Mark A. Horowitz of Department of Electrical Engineering, Stanford University – Informs Operation Research 2005

[3] **A Tutorial on Geometric Programming** - Stephen P. Boyd and Seung-Jean Kim of Department of Electrical Engineering, Stanford University, Lieven Vandenbergh of Department of Electrical Engineering, University of California and Arash Hassibi of Clear Shape Technologies, Inc., Sunnyvale – 2005

[4] **A Sequential Quadratic Programming Approach to Concurrent Gate and Wire Sizing** – Noel Menezes Member IEEE, Ross Baldick, Member IEEE and Lawrence T Pileggi Senior Member IEEE – 1997 IEEE

[5] **Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation** – Chung-Ping Chen, Chris C. N. Chu and D. F. Wong, Member IEEE – 1999 IEEE

[6] **ggplab: v. 1.00 - A Matlab Toolbox for Geometric Programming** – Stephen Boyd, Seung-Jean Kim, Kwangmoo Koh and Almir Mutpcic of Department of Electrical Engineering Stanford University – 2006

[7] **gposy - A Matlab Solver for Geometric Programming** – Stephen Boyd, Seung-Jean Kim, Kwangmoo Koh and Almir Mutpcic of Department of Electrical Engineering Stanford University – 2006

[8] gpcvx - A Matlab Solver for Geometric Programs in Convex Form – Stephen Boyd, Seung-Jean Kim, Kwangmoo Koh and Almir Mutpcic of Department of Electrical Engineering Stanford University – 2006

[9] CVX: Matlab software for disciplined convex programming, version 1.0 beta - M. Grant, S. Boyd, and Y. Ye. - Available at www.stanford.edu/~boyd/cvx/ - April 2006.

[10] Crosstalk-driven interconnect optimization by simultaneous gate and wire sizing – I. Jang, Y. Chang and J. Jou, Transactions on Computer-Aided Design of Integrated Circuits and Systems– 2000 IEEE

[11] Convex Optimization – J. Webster, Wiley Encyclopedia of Electrical and Electronics Engineering – 1999 John Wiley & Sons, Inc.

[12] Exact Solution to The Transistor Sizing Problem – Sachin Sapatnekar of Department of Electrical Engineering and Computer Engineering at Iowa State University, Vasant B. Rao, Pravin M. Vaidya and Sung-Mo Kang – 1993 IEEE

[13] Integration – J. Cong – 1996 The VLSI Journal