ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ & ΔΙΚΤΥΩΝ

«Αυτόματη Αλλαγή Τρανζίστορ σε Φυσικό Επίπεδο»

Διπλωματική Εργασία

Ιακώβου Αριστείδης
Χαλκιάς Παναγιώτης

Βόλος
Σεπτέμβριος,2011

Ευχαριστίες

Ολοκληρώνοντας τις προπτυχιακές μας σπουδές θα θέλαμε να ευχαριστήσουμε τα άτομα που στάθηκαν δίπλα μας και μας στήριξαν κατά την διάρκεια της φοίτησης μας και κατά την εκπόνηση της παρούσας εργασίας.

Αρχικά, θα θέλαμε να ευχαριστήσουμε τον καθηγητή του τμήματος Μηχανικών Υπολογιστών Τηλεπικοινωνιών και Δικτύων κ. Γεώργιο Σταμούλη για την ευκαιρία που μας έδωσε να πραγματοποιήσουμε αυτήν την μελέτη, καθώς και τους επιβλέποντες καθηγητές μας κ. Νέστορα Ευμορφόπουλο και κ. Γεώργιο Δημητρίου.

Επιπρόσθετα θα θέλαμε να ευχαριστήσουμε τις αγαπημένες μας οικογένειες για την αμέριστη συμπαράσταση τους καθ' όλη την ακαδημαϊκή μας πορεία, που στάθηκαν δίπλα μας σε κάθε δυσκολία που αντιμετωπίσαμε και αποτέλεσαν το κατ' εξοχήν βασικό στήριγμά μας.

Επιπλέον, δεν θα μπορούσε να παραλείψουμε από τις ευχαριστίες μας τους συμφοιτητές και συνεργάτες μας στην παρούσα διπλωματική εργασία Κωνσταντίνα και Μαρία για την άψογη συνεργασία που είχαμε.

Τέλος ένα μεγάλο ευχαριστώ σε όλους τους φίλους και τα κοντινά μας πρόσωπα που με την κατανόηση και εμψύχωσή τους στάθηκαν αρωγοί σε αυτή την πολυετή προσπάθειά μας.

Ιακώβου Αριστείδης
Χαλκιάς Παναγιώτης

ΣΥΝΟΨΗ

Οι αυξανόμενες απαιτήσεις για υλοποίηση ολοένα και περισσότερων λειτουργιών από ένα κύκλωμα, έχει οδηγήσει σε υψηλό βαθμό ολοκλήρωσης. Κάτι τέτοιο, οδηγεί σε αναζήτηση νέων μεθόδων κατασκευής και υλοποίησης ολοκληρωμένων κυκλωμάτων. Η ανάγκη αυτή προέκυψε από την ελάττωση των διαστάσεων της τεχνολογίας που απαιτείται σε κάθε φάση εξέλιξης και βασικό μέλημα των κατασκευαστών ήταν η ταχύτητα απόκρισης τους. Τα τελευταία χρόνια, όμως, έχει παρατηρηθεί μια αυξανόμενη απαίτηση για σχεδιάσεις χαμηλής κατανάλωσης ισχύος. Έτσι, αφού ο σχεδιαστής γράψει τον κατάλληλο κώδικα, και λαμβάνοντας υπόψη του όλες τις τεχνικές προδιαγραφές που του έχουν δοθεί, καλείται να υπολογίσει την ισχύ που καταναλώνεται, προκειμένου να επιβεβαιώσει ότι η εκάστοτε σχεδίαση ανταποκρίνεται στις δεδομένες απαιτήσεις. Αυτό γίνεται έτσι ώστε να μην έχουμε προβλήματα ορθότητας κατά την τελική κατασκευή του ολοκληρωμένου κυκλώματος.

Στην παρούσα εργασία, υλοποιούμε ένα πρόγραμμα για διαχείριση ενέργειας από τυποποιημένα κύτταρα. Στόχος του εργαλείου που υλοποιήσαμε είναι η ελαχιστοποίηση κατανάλωσης ισχύος μέσω μείωσης του πλάτους των τρανζίστορ. Ικανοποιώντας όλους τους απαιτούμενους σχεδιαστικούς κανόνες (DRC), και λαμβάνοντας αρχεία εισόδου σε GDSII μορφή από βιβλιοθήκες τυποποιημένων κυττάρων της εταιρείας NandgateOpenCell Library, το μεταεργαλείο που αναπτύχθηκε στην παρούσα εργασία, είναι ικανό να παράγει μια φυσική αναπαράσταση ενός κυκλώματος με μειωμένα πλάτη τρανζίστορ. Με την τεχνική αυτή επιτυγχάνεται χαμηλή καταναλισκόμενη ισχύς και αύξηση αντοχής των συσκευών με πολύ μικρές επιπτώσεις στην απόκριση ταχύτητας.

ABSTRACT

The growing demands for the achievement of more and more processes of one circuit

have led to a high level of integration. Such a thing increases the need for research of new integrated circuits' construction and materialization methods. This need emerged from the decrease of technology's dimensions, which is demanded on every phase of evolution and t he basic concern of the manufacturers was the speed response. The last years though, a growing demand for low power consumption designs, has been observed. So when the designer produces the appropriate code, considering all the technical standards, he has to calculate the consumed power in order to confirm that the design meets the given needs. That is, for avoiding any accuracy issues during the construction of the integrated circuit.

In this project we implement a program for power management by standard cells. Our goal is to achieve low power consumption by decreasing the transistors' width. By having satisfied the required designing rules (DRC) and having received input files in GDSII form from standard cell libraries of NandgateOpenCell Library company, the tool that we developed in this project is capable of yielding a physical layout of a circuit with decreased transistor widths. With this technique, a low consumed power and an increased resistance of the devices is achieved, with just a small effect on the speed response.

Περιεχόμενα

ΕΙΣΑΓΩΓΗ

Οι ραγδαίες αλλαγές που συντελέστηκαν στον τομέα των επικοινωνιών και των υπολογιστών, κατά την διάρκεια της παγκόσμιας ιστορίας, έγιναν στα πλαίσια της αποκαλούμενης «ψηφιακής επανάστασης». Ήταν αυτή, η οποία καθόρισε σε πολύ μεγάλο βαθμό της μετέπειτα τεχνολογικές συνθήκες και κατόρθωσε να θέσει νέες παραμέτρους στις σύγχρονες κοινωνίες. Οι περισσότερες ερευνητικές και βιομηχανικές προσπάθειες βασίστηκαν στην αύξηση της ταχύτητας και της πολυπλοκότητας των ψηφιακών κυκλωμάτων καθώς και στην παραγωγή τους με χαμηλό κόστος. Με το πέρας των δεκαετιών, ανοίχτηκαν οι δρόμοι για μια νέα εποχή που την χαρακτηρίζουν κατά βάση ασύρματα δίκτυα που επιτυγχάνουν τεράστιες ταχύτητες, προσωπικοί υπολογιστές με δυνατότητες απεικόνισης σύνθετων γραφικών και υπολογιστικών συστημάτων, και κινητά τρίτης γενιάς που μπορούν να αναπαράγουν video και διαθέτουν πλοηγό ιστοσελίδων. Η ψηφιακή τεχνολογία έχει ενταχθεί ποικιλοτρόπως στην καθημερινότητά μας κι όλα δείχνουν πως κινείται στην κατεύθυνση πολλών επεξεργαστών πάνω σε ένα μόνο chip.

Στον πυρήνα της ψηφιακής τεχνολογίας λαμβάνουν χώρα οι διατάξεις των ολοκληρωμένων κυκλωμάτων. Δημιουργείται λοιπόν η ανάγκη τα ολοκληρωμένα κυκλώματα να λαμβάνουν ολοένα και λιγότερο χώρο ενώ οι απαιτήσεις τους για ταχύτητα να αυξάνονται. Από τα δεδομένα αυτά ανακύπτει το ζήτημα της κατανάλωσης ισχύος, η οποία πρέπει να είναι χαμηλή για να ανταποκρίνεται στις αυστηρές απαιτήσεις φορητών συσκευών και να μεγαλώνουν την διάρκεια ζωής των μπαταριών τους.

Ο υπολογισμός κι η μείωση της καταναλισκόμενης ισχύος σε ένα κύκλωμα αποτελεί ένα θέμα καίριας σημασίας κι ένα πεδίο με περαιτέρω και ραγδαίες δυνατότητες ανάπτυξης. Για το λόγο αυτό, πολλοί σχεδιαστές αλλά και εταιρείες σχεδίασης ολοκληρωμένων κυκλωμάτων προχώρησαν σε δημιουργία εργαλείων που στόχο έχουν την ελαχιστοποίηση της καταναλισκόμενης ισχύος. Επιπλέον, για τον σκοπό αυτό, προχώρησαν στην βελτιστοποίηση άλλων παραμέτρων, όπως την αλλαγή μεγεθών στα στοιχεία του κυκλώματος. Με βάση αυτές τις απαιτήσεις, έγινε η προσπάθεια υλοποίησης αλγορίθμου ενός τέτοιου εργαλείου που περιγράφεται στην παρούσα εργασία.

1.1.1 Αντικείμενο πτυχιακής εργασίας

Βασικός στόχος της παρούσας διπλωματικής εργασίας είναι η δημιουργία ενός εργαλείου CAD για power management από τυποποιημένα κύτταρα (standard cells). Συγκεκριμένα καλούμαστε να υλοποιήσουμε έναν parser για την ελαχιστοποίηση του πλάτους των τρανζίστορ με σκοπό την βελτίωση της κατανάλωσης ισχύος. Τα αρχεία εισόδου που χρησιμοποιήσαμε είναι σε μορφή GDSII και η φυσική αναπαράσταση της εξόδου (layout) των ολοκληρωμένων κυκλωμάτων σε GDSII stream format έγινε μέσω του λογισμικού (layout viewer) "OwlVision GDSII Viewer"

Η υλοποίηση του αλγορίθμου πραγματοποιήθηκε σε γλώσσα προγραμματισμού C. Η ανάπτυξη του κώδικα και η αποσφαλμάτωση του έγινε σε περιβάλλον UNIX προκειμένου να βελτιστοποιηθεί η κατανάλωση μνήμης και η δυνατότητα μεταφερσιμότητας του κώδικα.

1.2 Οργάνωση του τόμου

Στο πρώτο κεφάλαιο παρατίθεται μία σύντομη περιγραφή του αντικειμένου και της δομής που θα αναλύσουμε στην παρούσα διπλωματική εργασία.

Στο δεύτερο κεφάλαιο αναλύονται και περιγράφονται κάποιες βασικές έννοιες και ορισμοί που θα συναντήσουμε και θα μας απασχολήσουν. Συγκεκριμένα, δίνονται οι ορισμοί των ολοκληρωμένων κυκλωμάτων (IC), γίνεται αναφορά στην έννοια των Standard Cells, στην βασισμένη σε αυτά μέθοδο σχεδιασμού καθώς και στους ορισμούς της κατανάλωσης ισχύος (δυναμική- στατική).

Στο τρίτο κεφάλαιο περιγράφεται η μορφή και δομή των αρχείων GDSII που χρησιμοποιήθηκαν ως αρχεία εισόδου κατά την υλοποίηση του αλγορίθμου.

Στο τέταρτο κεφάλαιο αναλύουμε την μέθοδο μείωσης πλάτους των τρανζίστορ που χρησιμοποιήσαμε για να επιτύχουμε ελαχιστοποίηση κατανάλωση ισχύος

Στο πέμπτο κεφάλαιο αναλύονται τα βήματα του προγράμματος που υλοποιήσαμε. Γίνεται αναφορά στα εργαλεία που χρησιμοποιήθηκαν και επεξηγούνται όλες οι βιβλιοθήκες και οι συναρτήσεις που δημιουργήσαμε.

Στο έκτο κεφάλαιο παρατίθενται παραδείγματα χρήσης και πειραματικές μετρήσεις του προγράμματος που υλοποιήθηκε.

Στο έβδομο κεφάλαιο συνοψίζονται τα συμπεράσματα της εργασίας και οι μελλοντικές επεκτάσεις του εργαλείου.

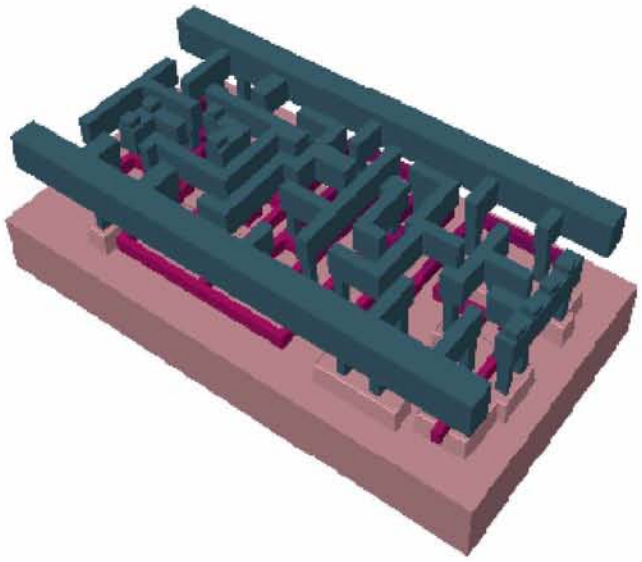
Τέλος παρουσιάζονται οι πηγές από όπου αντλήσαμε τι πληροφορίες για την υλοποίηση και ανάπτυξη του αντικειμένου της παρούσας διπλωματικής.

2 ΠΕΡΙΓΡΑΦΗ ΑΡΧΕΙΩΝ GDSII

2.1 Εισαγωγικά

Για πολλά χρόνια στον σχεδιασμό ολοκληρωμένων κυκλωμάτων το δημοφιλέστερο σχήμα για ανταλλαγή ήταν το GDS II (Graphic Data System) stream format αναπτυγμένο από την Calma Company. Υπήρξε το μοναδικό στο είδος του και πολλοί προμηθευτές το χρησιμοποιούσαν στα συστήματα τους.

Στην παρούσα υλοποίηση του αλγορίθμου χρησιμοποιούμε, όπως ειπώθηκε παραπάνω, GDSII αρχεία ως αρχεία εισόδου. Πρόκειται για αρχεία σε δυαδική μορφή που αντιπροσωπεύουν τις επίπεδες γεωμετρικές μορφές, τις ετικέτες κειμένων, και άλλες πληροφορίες για το Layout με ιεραρχική δομή. Τα στοιχεία είναι τοποθετημένα σε layers. Είναι ένα δυαδικό σχήμα που είναι ανεξάρτητο πλατφορμών καθώς χρησιμοποιεί εσωτερικά καθορισμένα σχήματα για τους τύπους δεδομένων του. Διαβάζοντας αρχεία GDSII, οι εσωτερικοί τύποι δεδομένων GDSII (όπως reals, integers κ.λπ.) πρέπει να μετατραπούν σε platform/CAE package. Το σχήμα GDSII είναι ένας διαδοχικός κατάλογος αρχείων, όπου κάθε αρχείο περιέχει έναν header (επικεφαλίδα) που περιέχει τις πληροφορίες που βρίσκονται στο αρχείο. Η σειρά του αρχείου πρέπει να είναι σύμφωνα με το GDSII BNF και λόγω αυτής της αυστηρής οργάνωσης είναι σχετικά εύκολο γίνει parse. Ο μέγιστος αριθμός vertixes επίσημα, είναι μόνο μέχρι 200 ζευγάρια X,Y παρόλα αυτά πολλά packages μπορούν να διαβάσουν μέχρι 64k/2=32k διότι αυτό είναι και το μέγιστο μήκος αρχείων που μπορεί να υπάρξει (2 bytes).



Εικόνα 2.1:Απόδοση ενός μικρού GDSII standard cell με τρία μεταλλικά στρώματα

Δεδομένης της δυαδικής μορφής του, τα GDSII αρχεία είναι δύσκολο να διαβαστούν και ο αναγνώστης έχει την δυνατότητα να δει τα περιεχόμενα του εκφρασμένα σε μορφή ASCII. Τέλος, έχει αναπτυχθεί μια μορφή ASCII (KEY format), η οποία είναι κάτι περισσότερο από μια αναπαράσταση κειμένου. Είναι δυνατή η μετατροπή GDSIIformat σε KEYformat και ανάποδα και μπορεί να περιέχει κύκλους, τόξα, πολύγωνα με τμήματα τόξων. Ένα παράδειγμα αναπαράστασης GDSII αρχείου σε KEY format φαίνεται παρακάτω:

```
# KEY file for GDS-II
# File = example.key
#
=============================================================================

HEADER 5; # version
BGNLIB;
LASTMOD {98-8-25  15:53:12}; # last modification time
LASTACC {98-8-25  15:53:12}; # last access time
LIBNAME TEMPEGS.DB;
UNITS;
USERUNITS 0.01; PHYSUNITS 1e-08;

BGNSTR; # Begin of structure
CREATION {98-8-25  15:53:12}; # creation time
```

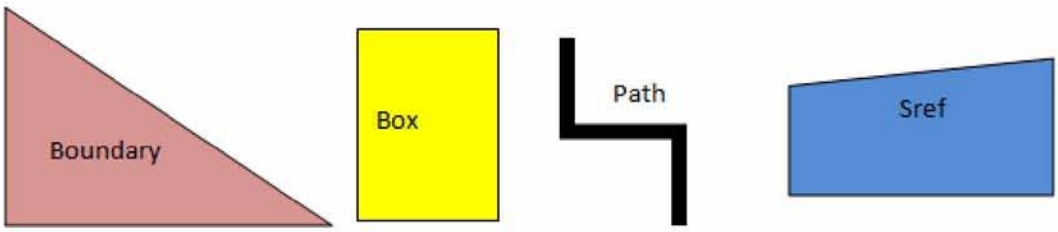
```
LASTMOD  {98-8-25  15:53:12}; # last modification time
STRNAME AAP;

BOUNDARY; LAYER 1;  DATATYPE 0;
   XY 5;
   X     -920000.000; Y      452000.000;   X      656500.000; Y
765500.000;
   X      175000.000; Y     -174000.000;   X     -756000.000; Y     -
198000.000;
   X     -920000.000; Y      452000.000;
ENDEL;

ENDSTR AAP;

BGNSTR; # Begin of structure
CREATION {98-8-25  15:53:12}; # creation time
LASTMOD  {98-8-25  15:53:12}; # last modification time
STRNAME LAYOUT;

BOUNDARY; LAYER 0;  DATATYPE 0;
   XY 5;
   X    -2032000.000; Y     1410000.000;   X     1427000.000; Y
1666000.000;
   X      502000.000; Y    -1580500.000;   X      502000.000; Y    -
1523500.000;
   X    -2032000.000; Y     1410000.000;
ENDEL;

BOX; LAYER 2;  BOXTYPE 0;
   XY 5;
   X     1526500.000; Y    -1034500.000;   X     2623500.000; Y    -
1034500.000;
   X     2623500.000; Y     1105500.000;   X     1526500.000; Y
1105500.000;
   X     1526500.000; Y    -1034500.000;
ENDEL;

SREF;
SNAME AAP;
STRANS 0,0,0;
   XY 1;

   X    -1112500.000; Y    -1267000.000;
ENDEL;

PATH; LAYER 3;  DATATYPE 0;WIDTH 100000;
   XY 4;
   X      891912.000; Y     2322024.000;   X      966537.000; Y
1854278.000;
   X     2599515.000; Y     2311647.000;   X     2626485.000; Y
2005353.000;
ENDEL;
```

```
TEXT; LAYER 3;
TEXTTYPE 0; PRESENTATION 0,2,0; PATHTYPE 1; STRANS 0,0,0; MAG 1875;
   XY 1;

   X     -2256500.000; Y      1539500.000;
STRING "Boundary";
ENDEL;

TEXT; LAYER 3;
TEXTTYPE 0; PRESENTATION 0,2,0; PATHTYPE 1; STRANS 0,0,0; MAG 1875;
   XY 1;

   X      -151500.000; Y      1924500.000;
STRING "Path";
ENDEL;

TEXT; LAYER 3;
TEXTTYPE 0; PRESENTATION 0,2,0; PATHTYPE 1; STRANS 0,0,0; MAG 1875;
   XY 1;

   X     -1740000.000; Y      -511500.000;
STRING "Sref";
ENDEL;

TEXT; LAYER 3;
TEXTTYPE 0; PRESENTATION 0,2,0; PATHTYPE 1; STRANS 0,0,0; MAG 1875;
   XY 1;

   X      1579000.000; Y      1301500.000;
STRING "Box";
ENDEL;

ENDSTR

LAYOUT;
ENDLIB;
```

Μια περιγραφή ενός GDS II κυκλώματος είναι μια συλλογή από κελιά (cells) που μπορεί να περιέχουν γεωμετρικές ή άλλες αναφορές κελιών. Αυτά τα κελιά, όπου στην GDSII γλώσσα καλούνται structures (δομές) έχουν αλφαριθμητικά ονόματα με μέγεθος μέχρι 32 χαρακτήρων . Μια βιβλιοθήκη από structures περιλαμβάνεται σε ένα αρχείο που αποτελείται από έναν library header , μια ακολουθία από structures, και μια library tail. Κάθε structure στην ακολουθία αποτελείται από έναν structure header , μια ακολουθία στοιχείων, και ένα structure tail.

Υπάρχουν επτά είδη στοιχείων:

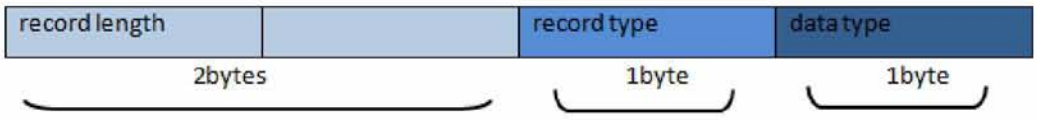
- ***Boundary*** ,το οποίο καθορίζει ένα γεμισμένο πολύγωνο,
- ***Path***, το οποίο καθορίζει ένα καλώδιο,
- ***Structure reference***, η οποία επικαλείται ένα subcell,
- ***Array reference***, η οποία επικαλείται έναν πίνακα από subcells,
- ***Text***, το οποίο περιλαμβάνει την τεκμηρίωση (documentation)
- ***Node***, ο οποίος καθορίζει ένα μονοπάτι ηλεκτρονικού κυκλώματος,
- ***Box***, το οποίο τοποθετεί την ορθογώνια γεωμετρία



Εικόνα 2.2:Αναπαράσταση ακολουθίας στοιχείων σε ένα structure

2.2 Record format

Για να γίνει κατανοητή η ακριβής μορφή ενός GDSII αρχείου, θα πρέπει πρώτα να περιγράψουμε την γενική μορφή του. Κάθε αρχείο GDSII αποτελείται από μία επικεφαλίδα (header) μεγέθους 4 Byte η οποία προσδιορίζει το μέγεθος της εγγραφής και της συνάρτησης. Τα πρώτα 2 Byte σχηματίζουν έναν ακέραιο αριθμό των 16 bit που δηλώνει το μήκος της εγγραφής σε bytes και το οποίο με την σειρά του περιλαμβάνει έναν header (επικεφαλίδα) που πρέπει να είναι πάντα ένα ζυγός αριθμός. Το τέλος μιας εγγραφής μπορεί να περιέχει ένα μηδενικό (null) byte όταν το περιεχόμενο της εγγραφής είναι περιττός αριθμός bytes. Το τρίτο byte της επικεφαλίδας περιέχει τον τύπο της εγγραφής και το τέταρτο byte περιέχει τον τύπο των δεδομένων της.



Το μήκος εγγραφής χρησιμοποιείται για να βρεθεί ο αριθμός των στοιχείων για συγκεκριμένα datatypes

Data Type	Value
No Data	0
Bit Array	1
2Byte Signed Int	2
4Byte Signed Int	3
4Byte Real	4(not used)
8Byte Real	5
ASCII string	6

Επειδή ο τύπος δεδομένων είναι σταθερός για κάθε τύπο εγγραφής, αυτό το πεδίο των 2-byte ορίζει και τα πιθανά αρχεία , όπως φαίνεται στα παρακάτω σχήματα.

File Header Records:	**Bytes 3 and 4**	**Parameter Type**
HEADER	0002	2-byte integer
BGNLIB	0102	12 2-byte integers
LIBNAME	0206	ASCII string
REFLIBS	1F06	2 45-character ASCII strings
FONTS	2006	4 44-character ASCII strings
ATTRTABLE	2306	44-character ASCII string
GENERATIONS	2202	2-byte integer
FORMAT	3602	2-byte integer
MASK	3706	ASCII string
ENDMASKS	3800	No data
UNITS	0305	2 8-byte floats
File Tail Records:	**Bytes 3 and 4**	**Parameter Type**
ENDLIB	0400	No data
Structure Header Records:	**Bytes 3 and 4**	**Parameter Type**
BGNSTR	0502	12 2-byte integers
STRNAME	0606	Up to 32-characters ASCII string
Structure Tail Records:	**Bytes 3 and 4**	**Parameter Type**
ENDSTR	0700	No data

Εικόνα 3.2: GDSII header records types

Element Header Records:	Bytes 3 and 4	Parameter Type
BOUNDARY	0800	No data
PATH	0900	No data
SREF	0A00	No data
AREF	0B00	No data
TEXT	0C00	No data
NODE	1500	No data
BOX	2D00	No data

Element Contents Records:	Bytes 3 and 4	Parameter Type
ELFLAGS	2601	2-byte integer
PLEX	2F03	4-byte integer
LAYER	0D02	2-byte integers
DATATYPE	0E02	2-byte integer
XY	1003	Up to 200 4-byte integer pairs
PATHTYPE	2102	2-byte integer
WIDTH	0F03	4-byte integer
SNAME	1206	Up to 32-character ASCII string
STRANS	1A01	2-byte integer
MAG	1B05	8-byte float
ANGLE	1C05	8-byte float
COLROW	1302	2 2-byte integers
TEXTTYPE	1602	2-byte integer
PRESENTATION	1701	2-byte integer

Εικόνα 2.3: GDSII elements record type

2.3 Library Head and Tail

Η επικεφαλίδα ενός GDSII αρχείου ξεκινά με την εγγραφή HEADER οι παράμετροι της οποίας περιλαμβάνουν πληροφορίες για τον αριθμό έκδοσης του αρχείου. Παραδείγματος χάριν, η σειρά των Bytes 0, 6, 0, 2, 0, 1 στην αρχή του αρχείου αποτελούν την επικεφαλίδα μιας εγγραφής ενός αρχείου αριθμού έκδοσης 1. Έπειτα ακολουθεί μια εγγραφή BGNLIB που περιέχει την ημερομηνία της τελευταίας τροποποίησης και την ημερομηνία της τελευταίας πρόσβασης στο αρχείο. Η τρίτη εγγραφή ενός αρχείου είναι η LIBNAME, το οποίο προσδιορίζει το όνομα αυτού του αρχείου της βιβλιοθήκης. Για παράδειγμα, τα bytes 0, 8, 2, 6, "C", "H", "I", "P" ορίζουν μια βιβλιοθήκη που ονομάζεται “chip”. Μετά από αυτήν την εγγραφή μπορεί να υπάρχουν κάποιες επικεφαλίδες προαιρετικών εγγραφών όπως: REFLIB που περιέχει τα ονόματα των βιβλιοθηκών αναφοράς, FONTS όπου αναφέρονται έως και τέσσερις γραμματοσειρές και FORMAT που υποδεικνύει το είδος του αρχείου.

Η τελευταία εγγραφή είναι η UNITS, η οποία δεν είναι προαιρετική και οι παράμετροι τις οποίας περιέχουν τον αριθμό μονάδων ανά μονάδα δεδομένων και τον αριθμό των μέτρων ανά μονάδα βάσης δεδομένων.

Μετά τις εγγραφές του HEADER file ακολουθούν οι εγγραφές των structures. Αφού έχει οριστεί και το τελευταίο structure το αρχείο τερματίζει με μια εγγραφή ENDLIB.

2.4 Structure Head and tail

Κάθε structure περιέχει δύο header records και έναν tail record. Η πρώτη από τους header records είναι η εγγραφή BGNSTR η οποία περιέχει την ημερομηνία δημιουργίας και την ημερομηνία τελευταίας τροποποίησης και η δεύτερη είναι η εγγραφή STRNAME που περιέχει το όνομα του structure. Η τελευταία εγγραφή είναι η ENDSTR. Μετά θα πρέπει να ακλουθεί άλλο ένα BGNSTR ή το τέλος της βιβλιοθήκης, ENDLIB.

2.5 Boundary Element

Το στοιχείο αυτό ορίζει ένα πολύγωνο και ξεκινά με την εγγραφή BOUNDARY (ενδεχομένως να περιέχει και τις προαιρετικές ELFLAGS και PLEX) και εν συνεχεία απαιτούνται οι εγγραφές **LAYER, DATATYPE,** και **XY**.

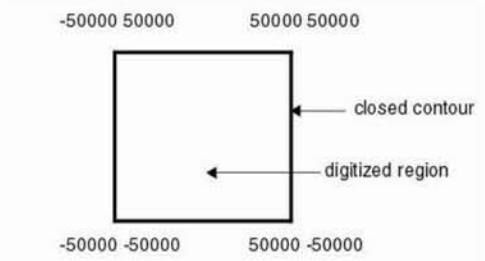
Η εγγραφή **LAYER** είναι απαραίτητη για να προσδιορίσει ποιο layer (αριθμημένο από 0 έως 63) χρησιμοποιείται από το boundary. Οι έννοιες των layers δεν έχουν αυστηρούς ορισμούς και για τον λόγο αυτόν θα πρέπει να προσδιορίζονται για κάθε σχεδιαστικό περιβάλλον και βιβλιοθήκη.

Η εγγραφή **DATATYPE** περιέχει πληροφορίες ήσσονος σημασίας γι' αυτό και η τιμή της πρέπει να είναι μηδενική.

Η εγγραφή XY περιέχει από 4 έως 200 ζεύγη συντεταγμένων τα οποία καθορίζουν το περίγραμμα του πολυγώνου άρα το πρώτο ζεύγος συντεταγμένων πρέπει να ταυτίζεται με το τελευταίο. Ο αριθμός των σημείων σε αυτό το αρχείο καθορίζεται από το μήκος της εγγραφής.

Ένα παράδειγμα φαίνεται παρακάτω:

```
BOUNDARY 5 0        <-- Boundary on layer 5, datatype 0
-50000  -50000      <-- First X,Y vertex
 50000  -50000      <-- Second X,Y vertex
 50000   50000      <-- Third X,Y vertex
-50000   50000      <-- Fourth X,Y vertex
-50000  -50000      <-- Back to First X,Y vertex
ENDEL
```



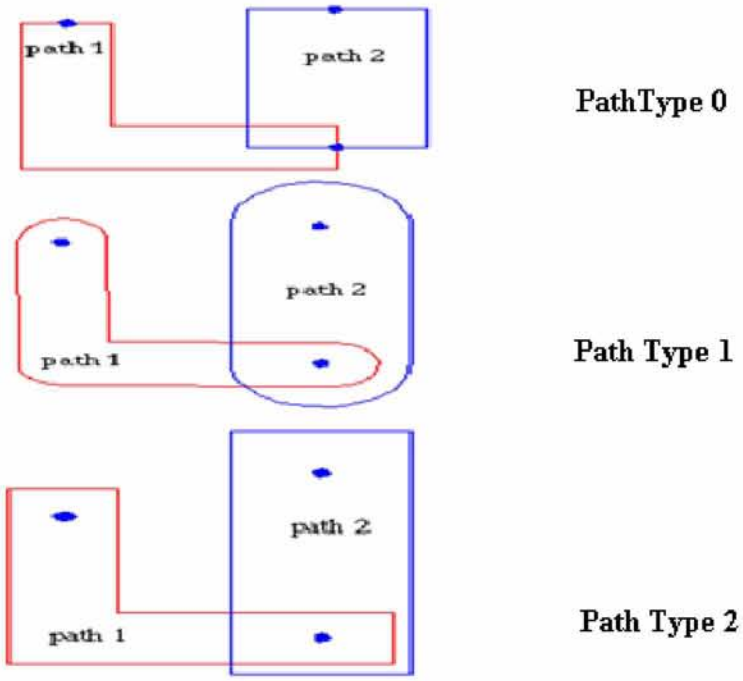
2.6 Path Element

Ένα path είναι μία τεθλασμένη γραμμή με μη μηδενικό πλάτος που συνήθως χρησιμοποιείται για να τοποθετηθούν τα καλώδια. Το στοιχείο αυτό αρχικοποιείται με μία εγγραφή PATH και ακολουθείται από τις προαιρετικές εγγραφές ELFLAGS και PLEX. Έπεται η εγγραφή LAYER που είναι απαραίτητη για τον προσδιορισμό του υλικού του path, η εγγραφή DATATYPE και η XY που προσδιορίζει τις συντεταγμένες του (από 2 έως 200).

Πριν από την XY μπορεί να υπάρχουν δύο προαιρετικές εγγραφές που ονομάζονται PATHTYPE και WIDTH. Η PATHTYPE περιγράφει το είδος των άκρων του path, σύμφωνα με την τιμή της παραμέτρου του. Η τιμή αυτή είναι:

- Μηδέν, εάν έχουν τετραγωνικές άκρες και τερματίζουν στις κορυφές του μονοπατιού
- Ένα, εάν έχουν στρογγυλεμένες άκρες
- Δύο, εάν έχουν στρογγυλεμένες άκρες που επικαλύπτουν τις κορυφές τους κατά το ήμισυ του πλάτους του

Παρακάτω παρουσιάζονται σχηματικά:



Εικόνα 2.3: Το είδος των άκρων του PATHTYPE, σύμφωνα με την τιμή της παραμέτρου του.

2.7 Structure Reference Element

Η ιεραρχία επιτυγχάνεται επιτρέποντας αναφορές σε structures (στιγμιότυπα) να εμφανίζονται σε άλλα structures. Η εγγραφή SREF υποδηλώνει μια αναφορά σε structure και ακολουθείται από τη προαιρετική ELFLAGS και PLEX. Η εγγραφή SNAME κατονομάζει στη συνέχεια την επιθυμητή δομή και μια εγγραφή XY περιέχει μία ξεχωριστή συντεταγμένη όπου θα τοποθετηθεί το στιγμιότυπο. Είναι επίσης επιτρεπτό να γίνεται αναφορά σε δομές που δεν έχουν ακόμη καθορισθεί με STRNAME.

2.8 Array of Structures Element

Για λόγους ευκολίας, ένας πίνακας από structure στιγμιότυπα μπορεί να καθοριστεί με την εγγραφή AREF. Μετά την προαιρετική ELFLAGS και PLEX εγγραφή ακολουθεί η SNAME για τον προσδιορισμό του πίνακα. Στη συνέχεια, η προαιρετική μετατροπή των εγγραφών STRANS, MAG, και ANGLE μπορεί να δώσει τον προσανατολισμό των στιγμιότυπων.

Ακολουθεί η εγγραφή COLROW η οποία καθορίζει τον αριθμό των στηλών και τον αριθμό των γραμμών του πίνακα. Η τελική εγγραφή είναι η XY με τρία σημεία: μία συντεταγμένη για την γωνία, μία συντεταγμένη για το τελευταίο στιγμιότυπο στην κατεύθυνση στήλης, και μία συντεταγμένη για το τελευταίο στιγμιότυπο προς στην κατεύθυνση σειράς. Από τις πληροφορίες αυτές, μπορεί να καθοριστεί το ποσό της επικάλυψης ή του διαχωρισμού των στιγμιότυπων.

2.9 Text element

Με την εγγραφή TEXT μπορούμε να συμπεριλάβουμε μηνύματα μέσα σε ένα κύκλωμα. Μετά τις προαιρετικές ELFLAGS, PLEX και την υποχρεωτική LAYER ακολουθεί η TEXTTYPE με μηδενική τιμή. Εν συνεχεία ακολουθεί η προαιρετική εγγραφή PRESENTATION η οποία καθορίζει την γραμματοσειρά σε Bits όπως φαίνεται παρακάτω:

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Word1	6(hex) of bytes in record															
Word2	17(hex)								01(hex)							
Word3	unused										Font number		Vertical presentation		Horizontal presentation	

Τέλος, πέραν των προαιρετικών εγγραφών PATHTYPE, WIDTH, STRANS, MAG, και ANGLE που μπορεί να εμφανιστούν μετά το TEXT απαιτούνται άλλες δύο εγγραφές : η XY με μία συντεταγμένη για να τοποθετήσει το κείμενο και η STRING που το ορίζει πλήρως.

2.10 Node element

Τα ηλεκτρονικά δίκτυα μπορούν να καθοριστούν από μία εγγραφή NODE. Μετά τις προαιρετικές ELFLAGS, PLEX και την υποχρεωτική LAYER ακολουθεί η NODETYPE με μηδενική τιμή. Στην συνέχεια ακολουθεί η XY με ένα έως πενήντα σημεία που προσδιορίζουν τις συντεταγμένες για το ηλεκτρονικό δίκτυο. Οι πληροφορίες σε αυτό το στοιχείο δεν είναι σε γραφικό περιβάλλον και δεν επηρεάζει το κατασκευασμένο κύκλωμα.

2.11 Box element

Το τελευταίο στοιχείο ενός αρχείου GDS II είναι το box. Μετά τις προαιρετικές ELFLAGS, PLEX και την υποχρεωτική LAYER ακολουθεί η BOXTYPE με μηδενική τιμή και η XY. Η XY πρέπει να περιέχει πέντε σημεία που περιγράφουν ένα κλειστό, τετράπλευρο κουτί. Σε αντίθεση με το όριο, αυτό δεν αποτελεί ένα γεμάτο σχήμα.

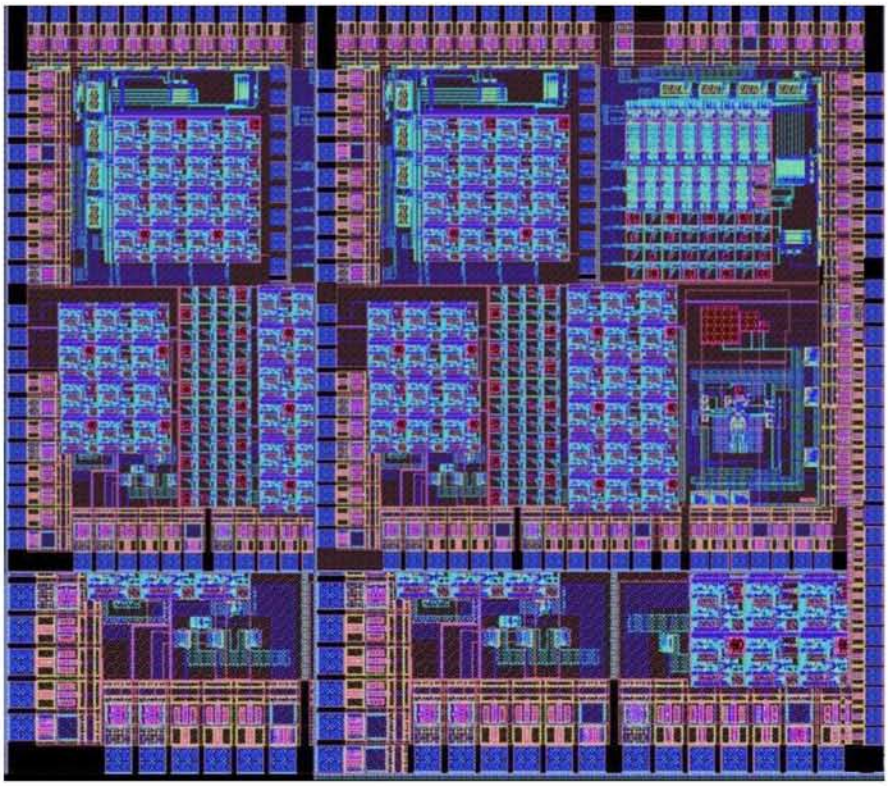
3 ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

3.1 Ολοκληρωμένα κυκλώματα (Integrated Circuit)

Ολοκληρωμένο κύκλωμα (*integrated circuit*) ή απλά *ολοκληρωμένο* ονομάζεται ένα κύκλωμα συνδεδεμένων λογικών πυλών, δημιουργημένο πάνω σε ένα φύλλο. Η συντριπτική πλειοψηφία των ολοκληρωμένων κυκλωμάτων δημιουργούνται πάνω σε φύλλα ημιαγωγών, κατά κύριο λόγο πυριτίου. Οι λογικές πύλες με την παρούσα τεχνολογία υλοποιούνται με παθητικά στοιχεία, οπότε τα ολοκληρωμένα κυκλώματα είναι παθητικά. Το ολοκληρωμένο κύκλωμα δεν είναι τίποτα άλλο παρά ένα πολύ προηγμένο ηλεκτρικό κύκλωμα. Ένα ηλεκτρικό κύκλωμα αποτελείται από διάφορα ηλεκτρικά στοιχεία όπως τρανζίστορ, αντιστάσεις και διόδους, τα οποία ενώνονται μεταξύ τους με διάφορους τρόπους. Αυτά τα στοιχεία έχουν διαφορετική συμπεριφορά.

- Το *τρανζίστορ* δουλεύει σαν ένας διακόπτης. Μπορεί να ενεργοποιήσει (ή να απενεργοποιήσει) την ροή του ηλεκτρικού ρεύματος ή να αυξομειώσει την ένταση του ρεύματος.
- Η *αντίσταση* περιορίζει την ροή του ηλεκτρισμού και δίνει την ικανότητα ελέγχου της ποσότητας του ρεύματος που είναι επιτρεπτό να περάσει.
- Ο *πυκνωτής* συγκεντρώνει ρεύμα και το απελευθερώνει με μια γρήγορη έκρηξη.
- Η *δίοδος* σταματά τον ηλεκτρισμό κάτω από κάποιες συνθήκες και επιτρέπει τη διέλευση ρεύματος όταν οι συνθήκες αυτές αλλάξουν.

Τα ολοκληρωμένα κυκλώματα διακρίνονται σε τέσσερις κατηγορίες SSI (μικρή κλίμακας ολοκλήρωσης), MSI (μεσαίας κλίμακας ολοκλήρωσης), LSI (μεγάλης κλίμακας ολοκλήρωσης) και VLSI (πολύ μεγάλης κλίμακας ολοκλήρωσης), ανάλογα με τον αριθμό των λογικών πυλών από τα οποία αποτελούνται. Παρακάτω φαίνεται ένα παράδειγμα ολοκληρωμένου κυκλώματος.



Εικόνα 3.1: Ένα ολοκληρωμένο κύκλωμα (Integrated circuit)

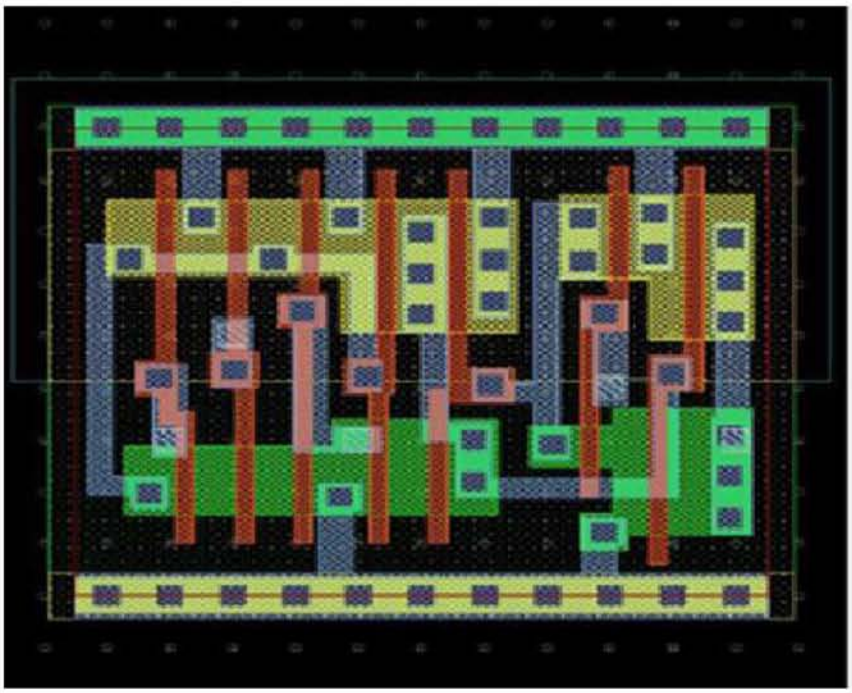
3.2 Κελιά (Cells)

Με τον όρο κελί (cell) στην παρούσα διπλωματική αναφέρεται κάθε πύλη ή ομάδα πυλών που χρησιμοποιήθηκε έχοντας γνώση, σε κάθε επίπεδο σχεδίασης, των βασικών χαρακτηριστικών και μεγεθών της (μήκος, πλάτος)

3.3 Standard Cell

Στη σχεδίαση ημιαγωγών, η μέθοδος standard cell αποτελεί μια μέθοδο σχεδιασμού ASIC κυκλωμάτων κυρίως με ψηφιακά-λογικά χαρακτηριστικά. Αποτελεί ένα παράδειγμα αφαιρετικού σχεδιασμού (design abstraction), όπου ένα χαμηλού επιπέδου layout ενσωματώνεται σε μια αφαιρετική λογική αναπαράσταση, όπως η πύλη NAND. Η μέθοδος βασιζόμενη στα cells (cell

based methodology) παρέχει τη δυνατότητα σε ένα σχεδιαστή να εστιάζει σε υψηλού επιπέδου πτυχή της ψηφιακής σχεδίασης (logical function) ενώ παράλληλα κάποιος άλλος μπορεί να εστιάζει στην υλοποίηση (physical). Παράλληλα με την πρόοδο στην κατασκευή ημιαγωγών, η βασισμένη στα standard cells μέθοδος ήταν υπεύθυνη για την κλιμάκωση των κυκλωμάτων ASIC από συγκριτικά απλά ολοκληρωμένα κυκλώματα, τα οποία εκτελούν μια συγκεκριμένη λειτουργία (αποτελούμενα από αρκετές χιλιάδες πύλες), σε σύνθετες συσκευές αποτελούμενες από πολλά εκατομμύρια πύλες (SoC-System on chip).



Εικόνα 3.2: Παράδειγμα Standard Cell Layout

3.3.1 Περιγραφή Standard Cell Design

Στο Standard Cell Design παρέχεται στον σχεδιαστή μια βιβλιοθήκη και προχωράει στον σχεδιασμό με την διασύνδεση των στοιχείων που περιέχονται σ' αυτή την βιβλιοθήκη, με όλα τα κομμάτια-σχέδια, που δημιουργούνται, να βασίζονται στα έτοιμα σχεδιαστικά κομμάτια αυτής. Η βιβλιοθήκη αυτή παρέχεται από τον κατασκευαστή του ολοκληρωμένου κυκλώματος και μπορεί να περιέχει από πολύ λίγα έως πάρα πολλά και πολύ σύνθετα σχεδιαστικά κομμάτια. Μια standard cell based βιβλιοθήκη θα μπορούσε να περιέχει μόνο τη λογική πύλη NAND δύο εισόδων μιας και κάθε λογική συνάρτηση μπορεί να εκφραστεί συναρτήσει αυτής της πύλης. Όμως οι βιβλιοθήκες με τις οποίες εφοδιάζονται οι σχεδιαστές σήμερα, χάριν διευκόλυνσής τους, περιέχουν όλες τις λογικές πύλες (σε διάφορες εκδόσεις ταχύτητας, εμβαδού και οδηγητικής

ικανότητας), στοιχεία μνήμης, μικρά έως μεσαία συνδυαστικά κυκλώματα μικρά έως μεσαία ακολουθιακά κυκλώματα (καταχωρητές, ολισθητές, μετρητές κλπ).

Οι τυποποιημένες βιβλιοθήκες cell απαιτούνται από σχεδόν όλα τα εργαλεία CAD για το σχεδιασμό των chip. Πρότυπες βιβλιοθήκες cell περιέχουν πρωτόγονα κύτταρα που απαιτούνται για την ψηφιακή σχεδίαση. Ωστόσο, μπορούν επίσης να συμπεριληφθούν και πιο σύνθετα cells
Ο κύριος σκοπός των CAD εργαλείων είναι η εφαρμογή της λεγόμενης RTL-to-GDS ροή. Το τελικό αποτέλεσμα από τη διαδικασία του σχεδιασμού είναι η πλήρης διάταξη ενός τσιπ, ως επί το πλείστον στην μορφή GDSII. Για να παραχθεί ένας λειτουργικά σωστός σχεδιασμός που πληροί όλες τις προδιαγραφές και τους περιορισμούς, απαιτείται ένας συνδυασμός διαφορετικών εργαλείων στο σχεδιασμό των ροών. Τα εργαλεία αυτά απαιτούν συγκεκριμένες πληροφορίες σε διάφορες μορφές για κάθε ένα από τα cells των βιβλιοθηκών που τους παρέχονται για τον σχεδιασμό

3.3.2 Σύνθεση Standard Cell

Ένα standard cell απαρτίζεται από ένα σύνολο τρανζίστορ και διασυνδεδεμένων δομών τα οποία αναπαριστούν μία Boolean συνάρτηση ή μια συνάρτηση αποθήκευσης. Τα απλά cell είναι αναπαραστάσεις στοιχειωδών Boolean συναρτήσεων ενώ χρησιμοποιούνται και πιο πολύπλοκα όπως ένας πλήρης αθροιστής. Μια λογική συνάρτηση cell ονομάζεται λογική αναπαράσταση (logical view) και η λειτουργική συμπεριφορά τους περιγράφεται από έναν πίνακα αληθείας ή από μία εξίσωση ή από έναν πίνακα μετάβασης.

Το σχέδιο ενός standard cell αναπτύσσεται σε επίπεδο transistor και αποτελεί τη μορφή netlist του transistor που είναι μια περιγραφή transistors, που λαμβάνουν χώρα στο σχέδιο, των συνδέσεων μεταξύ τους και με το εξωτερικό περιβάλλον. Για την προσομοίωση της ηλεκτρονική συμπεριφορά της μορφής netlist του αρχικού σχεδίου, χρησιμοποιούνται CAD προγράμματα από σχεδιαστές δηλώνοντας διαφορές παραμέτρους εισαγωγής και υπολογίζοντας την απόκριση του κυκλώματος. Επιπλέον απαιτείται και η φυσική αναπαράσταση του standard cell η οποία ονομάζεται layout view και από κατασκευαστικής άποψης αποτελεί την πιο σημαντική αφού μοιάζει με ένα ακριβές «αποτύπωμα» του, οργανωμένο σε επίπεδα μετάλλων. Για κάθε μορφή netlist, μπορεί να υπάρξουν πολλά διαφορετικά layouts, τα οποία συμβαδίζουν με τις παραμέτρους απόδοσης της netlist. Στόχος κάθε σχεδιαστή είναι η ελαχιστοποίηση του κόστους κατασκευής του layout, λαμβάνοντας υπόψη τις

διάφορες απαιτήσεις, σχετικές με την ταχύτητα και την απόδοση ισχύος του cell, μια αρκετά επίπονη διαδικασία.

Τέλος, τα στοιχεία ενός standard cell έχουν συνήθως όλα ένα σταθερό ύψος (height). Για τον λόγο αυτό, δίνεται η δυνατότητα να μπορούν να τοποθετηθούν το ένα δίπλα στο άλλο, ώστε να επιτυγχάνεται η μεταξύ τους διασύνδεση στα πλαίσια ενός πιο πολύπλοκου κυκλώματος. Συνεπώς, η έκταση του standard cell στο chip αποτελείται από ένα μεγάλο αριθμό cells τοποθετημένα στη σειρά με τη τροφοδοσία και τη γείωση συνδεδεμένες στο πάνω και στο κάτω μέρος του συνολικού χώρου. Η επεξεργασία στα επιμέρους στοιχεία γίνεται από ειδικά εργαλεία και εξαρτάται από την λογική του κυκλώματος που θα υλοποιηθεί.

3.4 Κατανάλωση Ισχύος

3.4.1 Κατανάλωση Ισχύος του CMOS

Όπως και σε κάθε είδος κυκλώματος, έτσι και στα κυκλώματα CMOS παρατηρείται κατανάλωση ισχύος. Συγκεκριμένα για τα CMOS η κατανάλωση ισχύος μπορεί να χωριστεί σε δύο κατηγορίες, σύμφωνα με τον τρόπο που γίνεται η κατανάλωση, σε στατική και δυναμική. Συγκεκριμένα η στατική οφείλεται στα ρεύματα διαρροής των κυκλωμάτων, ή σε άλλα ρεύματα που ρέουν συνεχώς από την τροφοδοσία, ενώ αντίθετα η δυναμική κατανάλωση οφείλεται στα ρεύματα μεταγωγής και φόρτισης-εκφόρτισης των χωρητικών φορτίων. Παρακάτω θα αναλύσουμε αναλυτικά τους όρους αυτούς.

3.4.2 Στατική Ισχύς

Ως στατική ισχύ χαρακτηρίζουμε την ισχύ που καταναλώνει μια πύλη όταν αυτή είναι αδρανής ή στατική. Στα CMOS «ιδανικά» καταναλώνεται μηδενική στατική ισχύ αφού στην κατάσταση ισορροπίας τους δεν υπάρχει μονοπάτι που να συνδέει την πηγή με την γείωση. Στην πραγματικότητα όμως, πάντοτε υπάρχουν έστω και πολύ μικρές τιμές ρευμάτων διαρροής. Το μεγαλύτερο ποσοστό στατικής ισχύος οφείλεται σε ένα δυναμικό που αναπτύσσεται ανάμεσα στην πηγή και την γείωση, το οποίο προκαλείται από ελαττωμένα δυναμικά κατωφλίου τα οποία εμποδίζουν την εκάστοτε πύλη από το να κλείσει εντελώς.

Τα χαρακτηριστικά αυτά αποδίδονται στη λειτουργία των ημιαγωγών και την ιδιότητα των περιοχών διαχύσεων να σχηματίζουν ανάστροφα πολωμένες

διόδους με το υπόστρωμα και άρα να μην άγουν σε ανάστροφη πόλωση. Γνωρίζουμε ότι στις διόδους με ανάστροφη πόλωση διαπερνούν μικρά ρεύματα διαρροής σύμφωνα με την εξίσωση :

$$i_o = i_s (e^{qV/kT} - 1)$$

όπου

i_s : ανάστροφο ρεύμα κόρου

V : τάση διόδου

q : φορτίο ηλεκτρονίου (1.602×10^{-19} C)

k: σταθερά του Boltzman (1.38×10^{-23} J/K)

T : θερμοκρασία

Η στατική κατανάλωση ισχύος είναι το γινόμενο μεταξύ ρεύματος διαρροής του στοιχείου και τάσης τροφοδοσίας. Αν το κύκλωμα αποτελείται από n στοιχεία, η ολική κατανάλωση ισχύος του κυκλώματος δίνεται από τον τύπο:

$$\mathbf{P_s} = \sum_{1}^{n} i_o \times V$$

3.4.3 Δυναμική Ισχύς

Δυναμική ονομάζεται η ισχύς που καταναλώνεται όταν μία πύλη είναι ενεργή. Ένα κύκλωμα είναι ενεργό όταν οι τάσεις του δικτύου εναλλάσσονται λόγω κάποιου εξωτερικού ερεθίσματος που εφαρμόζουμε στην έξοδο. Επειδή η τάση στην είσοδο μπορεί να αλλάξει, χωρίς αυτό να συνεπάγεται κάποια λογική μεταβολή στην έξοδο, δυναμική ισχύς καταναλώνεται κι όταν η έξοδος δεν αλλάζει την λογική της κατάσταση.

Η δυναμική ισχύς που καταναλώνεται σε ένα κύκλωμα δίνεται από τον τύπο:

$$P_d = C_L V_{DD}^2 f_p$$

Όπου:

C_L: χωρητικό φορτίο εξόδου

V_{DD} : τάση τροφοδοσίας

f_p : συχνότητα παλμού εισόδου

Εάν θέλουμε να αναφερθούμε σε μεγαλύτερο πλήθος κύκλων ρολογιού θα πρέπει να συμπεριλάβουμε και τον παράγοντα μεταβάσεων (**switching activity**) '**a**' ο οποίος υπολογίζει τον αριθμό μεταβάσεων ανά κύκλο ρολογιού. Κι ο τύπος που προκύπτει είναι ο εξής:

$$P_d = a\, C_L V_{DD}^2 f_p$$

Πιο αναλυτικά η δυναμική κατανάλωση ισχύος εξαρτάται από δυο συντελεστές, την *ισχύ λόγω μεταγωγικής λογικής τιμής (switching power)* και την *εσωτερική ισχύ (internal power)* , οι οποίοι αναλύονται παρακάτω:

- Ισχύς λόγω μεταγωγικής λογικής τιμής (switching power):
  Η switching power ενός κελιού της σχεδίασης είναι η συνολική ισχύς που καταναλώνεται από την φόρτιση και εκφόρτιση της χωρητικότητας στην έξοδο του κελιού. Η συνολική χωρητικότητα φορτίου στην έξοδο υπολογίζεται από το άθροισμα της χωρητικότητας του δικτύου και της πύλης στην έξοδο. Επειδή η φόρτιση και η εκφόρτιση είναι αποτέλεσμα λογικών μεταβάσεων συνεπώς η ισχύς αυξάνεται όσο αυξάνεται κι ο αριθμός μεταβάσεων που έχουμε στο κύκλωμα μας.

- Εσωτερική ισχύς (internal power):

  Αποτελεί την ισχύ που καταναλώνεται στα όρια ενός κελιού η οποία προκαλείται από την φόρτιση και εκφόρτιση των πυκνωτών στο εσωτερικό του κελιού. . Η εσωτερική ισχύς περιλαμβάνει και ένα ποσοστό ισχύος το οποίο ονομάζεται short-circuit power και προκαλείται από την στιγμιαία ένωση μεταξύ του P και του N τρανζίστορ μιας πύλης.

  Επειδή λοιπόν η δυναμική ισχύ καταναλώνεται όταν το κύκλωμα είναι ενεργό το συνολικό ποσό που καταναλώνεται υπολογίζεται ως εξής:

Δυναμική Ισχύς= Ισχύς Μεταγωγής + Εσωτερική Ισχύς

4 ΕΛΑΧΙΣΤΟΠΟΙΗΣΗ ΚΑΤΑΝΑΛΩΣΗΣ ΙΣΧΥΟΣ

4.1 Μείωση πλάτους των τρανζίστορ για ελαχιστοποίηση κατανάλωσης ισχύος

Για τον τύπο της ισχύος έχουμε:

P=VI

Άρα έχουμε δύο ελεύθερους παράγοντες για να επέμβουμε.

Εφόσον η ισχύς εξαρτάται από την τάση τροφοδοσίας και το ρεύμα διαρροής, αυτοί είναι και οι παράγοντες οι οποίοι θα πρέπει να μειώσουμε για να έχουμε ελαχιστοποίηση της ενέργειας. Επιλέγουμε να αλλάζουμε το ρεύμα και όχι την τάση. Η τάση τροφοδοσίας έχει τα χαρακτηριστικά παραμέτρου της σχεδίασης και οι σύγχρονες σχεδιάσεις χαμηλής κατανάλωσης ισχύος έχουν τάση τροφοδοσίας μεταξύ 1.5 και 3 V. Άρα δεν μπορούμε να επέμβουμε στην τάση.

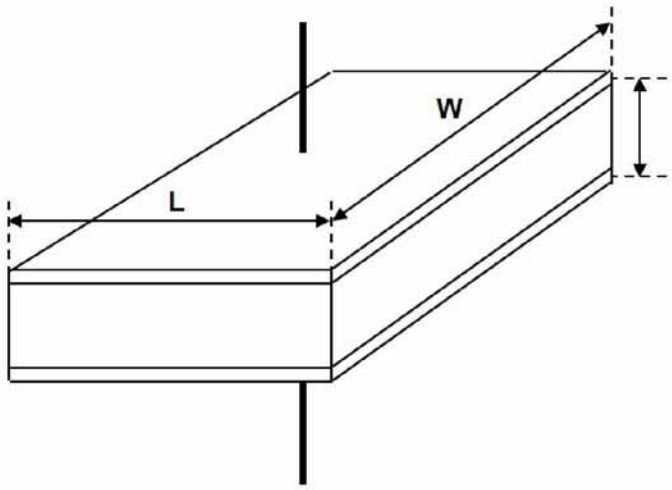
Η ελαχιστοποίηση λοιπόν της κατανάλωσης ισχύος άπτεται στην μείωση του ρεύματος διαρροής. Πρώτος τρόπος επίτευξης αυτού είναι η χρησιμοποίηση συμπληρωματικών πυλών, καθώς επίσης, γνωρίζουμε ότι η διαρροή ρεύματος σε επαφές p-n είναι ανάλογες με την επιφάνεια.

Οι τύποι ρευμάτων για τα n-mos και p-mos αντίστοιχα είναι:

$$\textbf{Περιοχές λειτουργίας}\begin{cases}\textbf{Αποκοπής} & I_{ds}=0,\quad V_{gs}\le V_t\\ \textbf{Γραμμική} & I_{ds}=\beta\left[(V_{gs}-V_t)V_{ds}-\frac{V_{ds}^2}{2}\right],\quad 0<V_{ds}<V_{gs}-V_t\\ \textbf{Κόρου} & I_{ds}=\beta\frac{(V_{gs}-V_t)^2}{2},\quad 0<V_{gs}-V_t<V_{ds}\end{cases}$$

$$\beta=\frac{\mu\varepsilon}{t_{ox}}\left(\frac{W}{L}\right)$$

Με β συντελεστή κέρδους, μ κινητικότητα φορέων, ε επιδεκτικότητα μονωτή πύλης, t_{ox} πάχος μονωτή πύλης, W/L λόγος διαστάσεων. Τα τρανζίστορ εργάζονται στην κατάσταση κόρου.



Εικόνα 4.1: Ένα πυκνωτής παράλληλων πλακών

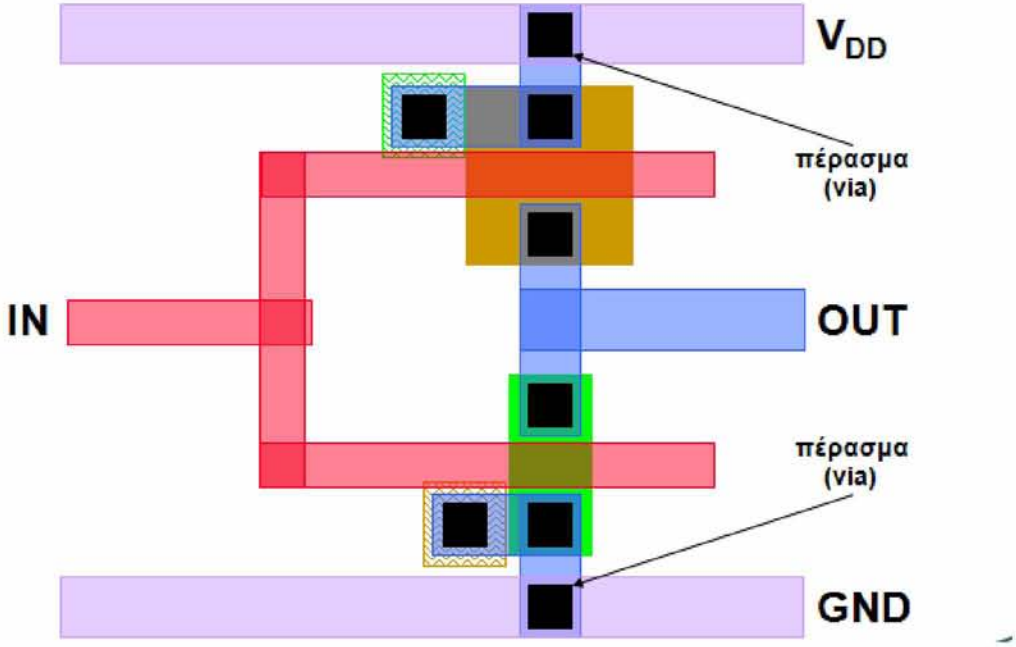
- *Παράγοντας L*

Εφόσον τις τάσεις δεν μπορούμε όπως είπαμε να τις επηρεάσουμε θα στραφούμε στους παράγοντες W και L από τα οποία εξαρτάται, όπως είδαμε παραπάνω, το ρεύμα διαρροής. Τα μήκη L κατασκευάζονται συνήθως στην ελάχιστη δυνατή διάσταση που τους επιτρέπεται από την υπάρχουσα τεχνολογία, και είναι συγκεκριμένη στα standard cell εφόσον έχουν κανονικότητα στην γεωμετρία και το φυσικό τους ύψος είναι σταθερό. Τα μεγέθη L είναι σταθερά σε κάθε τεχνολογία CMOS. Το μέγεθος θα έπρεπε να διπλασιαστεί για να υποδιπλασιαστεί η κατανάλωση. Οπότε οποιαδήποτε αλλαγή στο μέγεθός τους θα προκαλούσε και θέματα placement. Επίσης αν αλλάζουμε το L επηρεάζουμε την συχνότητα προς το χειρότερο. Άρα γι' αυτό το λόγο δεν επιλέγουμε να αλλάζουμε το L.

- *Παράγοντας W*

Ο μοναδικός ελεύθερος παράγοντας που απομένει για να επηρεάσουμε είναι ο παράγοντας W. Εφόσον το πλάτος W είναι ανάλογο με το ρεύμα διαρροής, θα πρέπει να μειώσουμε το μέγεθος του για να προκαλέσουμε κατ' επέκταση ελαχιστοποίηση της ισχύς. Η μείωση του πλάτους των τρανζίστορ όμως πρέπει να γίνει με προσοχή και συγκεκριμένο τρόπο γιατί υπάρχουν κάποια όρια και σχεδιαστικοί κανόνες που πρέπει να ακολουθήσουμε έτσι ώστε το κύκλωμα μας

να δουλεύει σωστά. Παρακάτω φαίνεται το φυσικό σχέδιο ενός αντιστροφέα CMOS.



Εικόνα 4.2: Layout αντιστροφέα CMOS

Το πλάτος των τρανζίστορ πρέπει να μειωθεί από μέσα προς τα έξω κι όχι ανάποδα, γιατί στην περίπτωση αυτή υπάρχει κίνδυνος οι τάσεις τροφοδοσίας και τα μέταλλα, όπως φαίνεται και στο σχήμα, να βγουν εκτός επιφάνειας. Κάτι τέτοιο θα προκαλούσε προβλήματα ορθής λειτουργίας του κυκλώματος.

Στον πηγάδι τύπου n μειώνουμε το πάνω μέρος από πάνω προς τα κάτω ενώ το πηγάδι τύπου p το κάτω μέρος από κάτω προς τα πάνω.

Επίσης αν αλλάξουμε το πλάτος πρέπει να βγάζουμε τα vias. Πρέπει όμως να έχουμε υπόψη μας ότι πρέπει να μείνει τουλάχιστον ένα vias για κάθε είσοδο , έξοδο και τάση τροφοδοσίας. Ένα ακόμη σημείο που πρέπει αν προσέξουμε είναι η μείωση των αντίστοιχων πηγαδιών που επηρεάζονται από την αλλαγή του πλάτους καθώς και των μετάλλων.

Αλλάζοντας λοιπόν το τρανζίστορ με φορά από «μέσα προς τα έξω», θα πρέπει στην επόμενη φάση να καταφέρουμε να το προσαρμόσουμε σύμφωνα με προκαθορισμένους γεωμετρικούς κανόνες σχεδίασης που υπαγορεύει το DRC (Design Rule Check) πρόγραμμα, προκειμένου το κύκλωμα που κατασκευάζεται να λειτουργεί σωστά. Ένα παραδείγματος χάριν, ένα vias, σύμφωνα με το DRC, έπρεπε να απέχει 10nm από κάθε πηγάδι, μετά την επεξεργασία του πλάτους η προδιαγραφή αυτή δεν επιτρέπεται να αλλάξει.

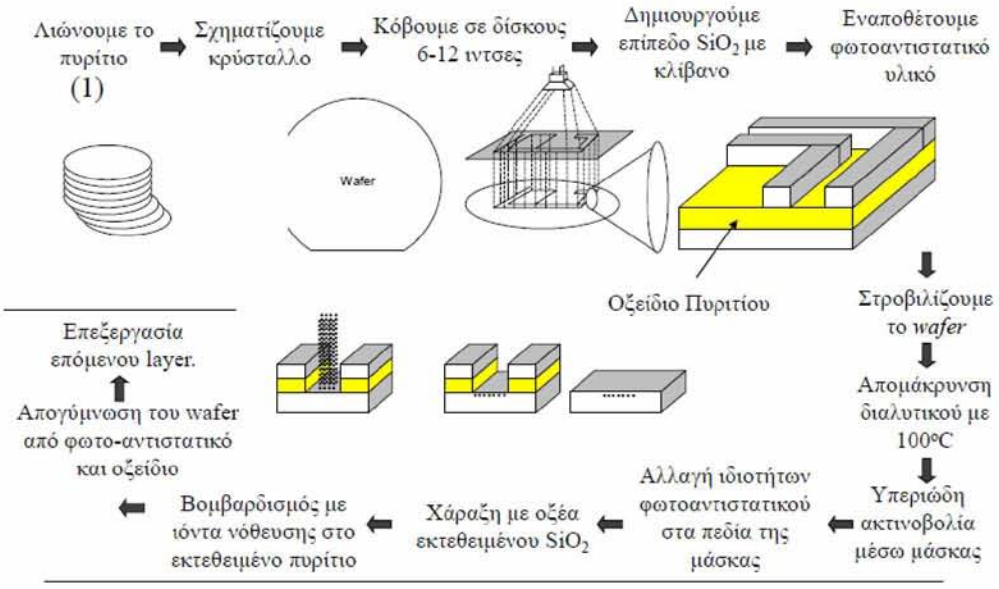
4.2 Διαδικασία κατασκευής CMOS

Για την ορθή λειτουργία του κυκλώματος εκτός από τους γεωμετρικούς κανόνες που πρέπει να ακολουθήσουμε, πρέπει η αλληλοσυσχέτιση των μασκών μέσω των βημάτων επεξεργασίας της να παράγει σωστά κυκλώματα.

Συγκεκριμένα τα τρανζίστορ κατασκευάζονται από πυρίτιο (ένας όγκος πυριτίου σε μορφή καρότου) το οποία τεμαχίζεται σε πολύ λεπτές φέτες τα wafers. Πάνω στην επιφάνεια του πυριτίου τοποθετούμε πολλαπλά επίπεδα αγώγιμων και μονωτικών υλικών. Είναι μια διαδικασία η οποία γίνεται σε βήματα που βασίζονται σε μια σειρά χημικών διεργασιών:

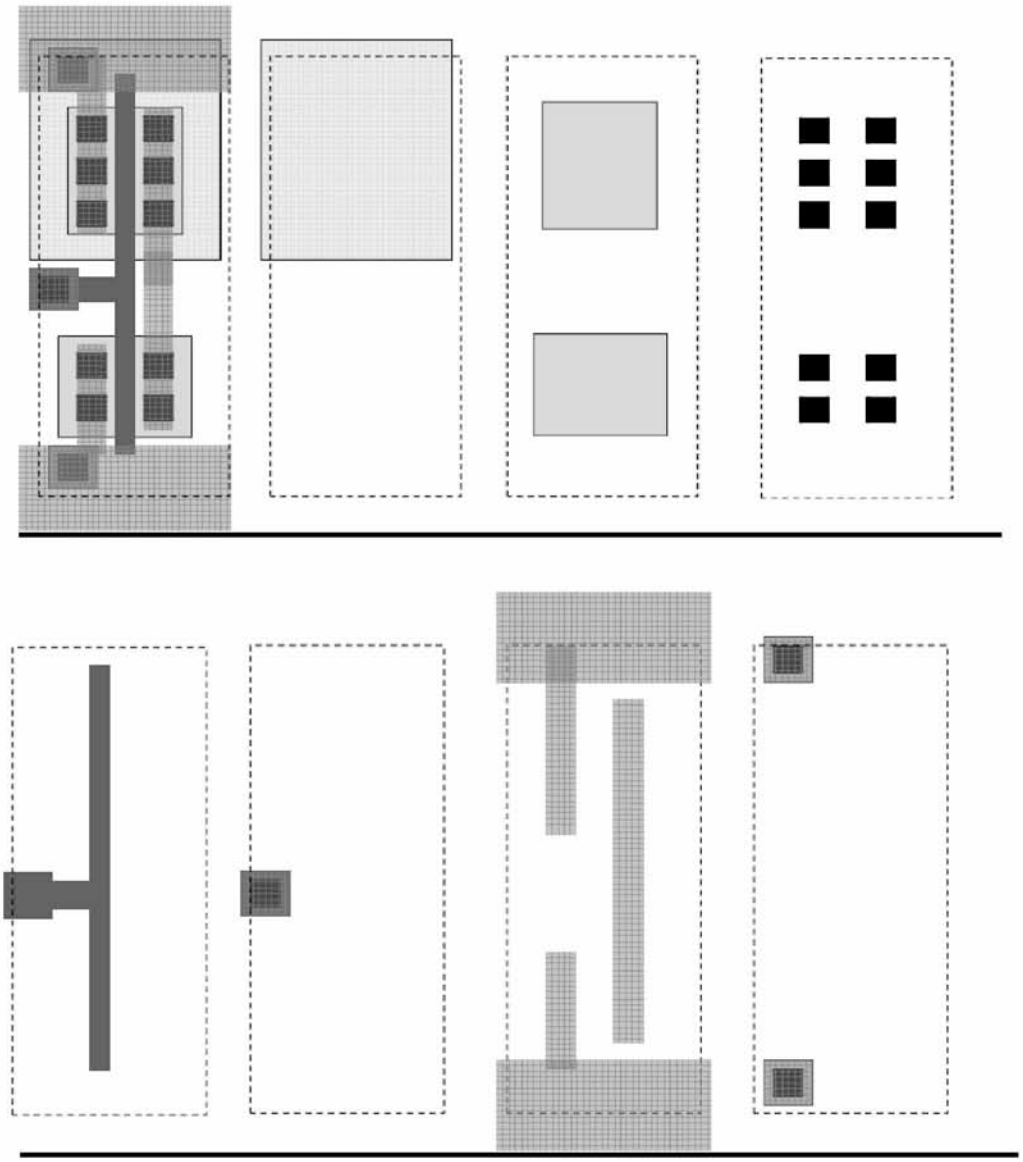
- Οξείδωση πυριτίου
- Διάχυση προσμίξεων
- Απόθεση και χάραξη αλουμινίου

Αναλυτικά η διαδικασία κατασκευής περιγράφεται στο παρακάτω σχήμα:



Εικόνα 4.3: Διαδικασία κατασκευής CMOS

Η διαδικασία κατασκευής είναι άμεσα συνδεδεμένη με την κατασκευή ειδικών μασκών. Οι μάσκες πρέπει να τηρούν κι αυτές τους κανόνες κατασκευής (DRC) κι όλες οι απαιτούμενες λεπτομέρειες κατασκευής είναι καταγεγραμμένες στις βιβλιοθήκες των κυττάρων. Παρακάτω, φαίνονται στις εικόνες τα βήματα σχεδιασμού μασκών για την κατασκευή.



Εικόνα 4.4: Βήματα για το διαχωρισμό του σχεδιασμού σε μάσκες

Συνεπώς η οποιαδήποτε αλλαγή που θα αφορά των πλάτος των τρανζίστορ θα πρέπει λοιπόν να είναι σύμφωνη με το DRC του κατασκευαστή. Εφόσον δεν επηρεάσουμε την κατασκευή των μασκών και λειτουργήσουμε βάσει των σχεδιαστικών κανόνων, η κατασκευή του Layout και η λειτουργία του κυκλώματος θα είναι σωστές.

Γενικά επιλέγουμε να επηρεάζουμε το μέγεθος των τρανζίστορ από μέσα προς τα έξω. Μόλις τελειώσουμε με αυτή την αλλαγή επιλέγουμε τα vias τα οποία πρέπει αν φύγουν και τα αντίστοιχα πηγάδια και μέταλλα που πρέπει να επηρεαστούν. Αν υπάρχει πρόβλημα στο DRC μετά το τέλος αυτής της διαδικασίας σταματάμε.

Στο παρόν κεφάλαιο θα γίνει αναφορά στα εργαλεία που χρησιμοποιήσαμε και θα αναλυθούν τα βήματα του προγράμματος που υλοποιήθηκε.

5 ΥΛΟΠΟΙΗΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

5.1 Εργαλεία:

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής είναι η γλώσσα προγραμματισμού C. Η ανάπτυξη του κώδικα και η αποσφαλμάτωση του έγινε σε περιβάλλον UNIX ενώ έγινε χρήση του μεταγλωττιστή gcc και των κατάλληλων παραμέτρων του.

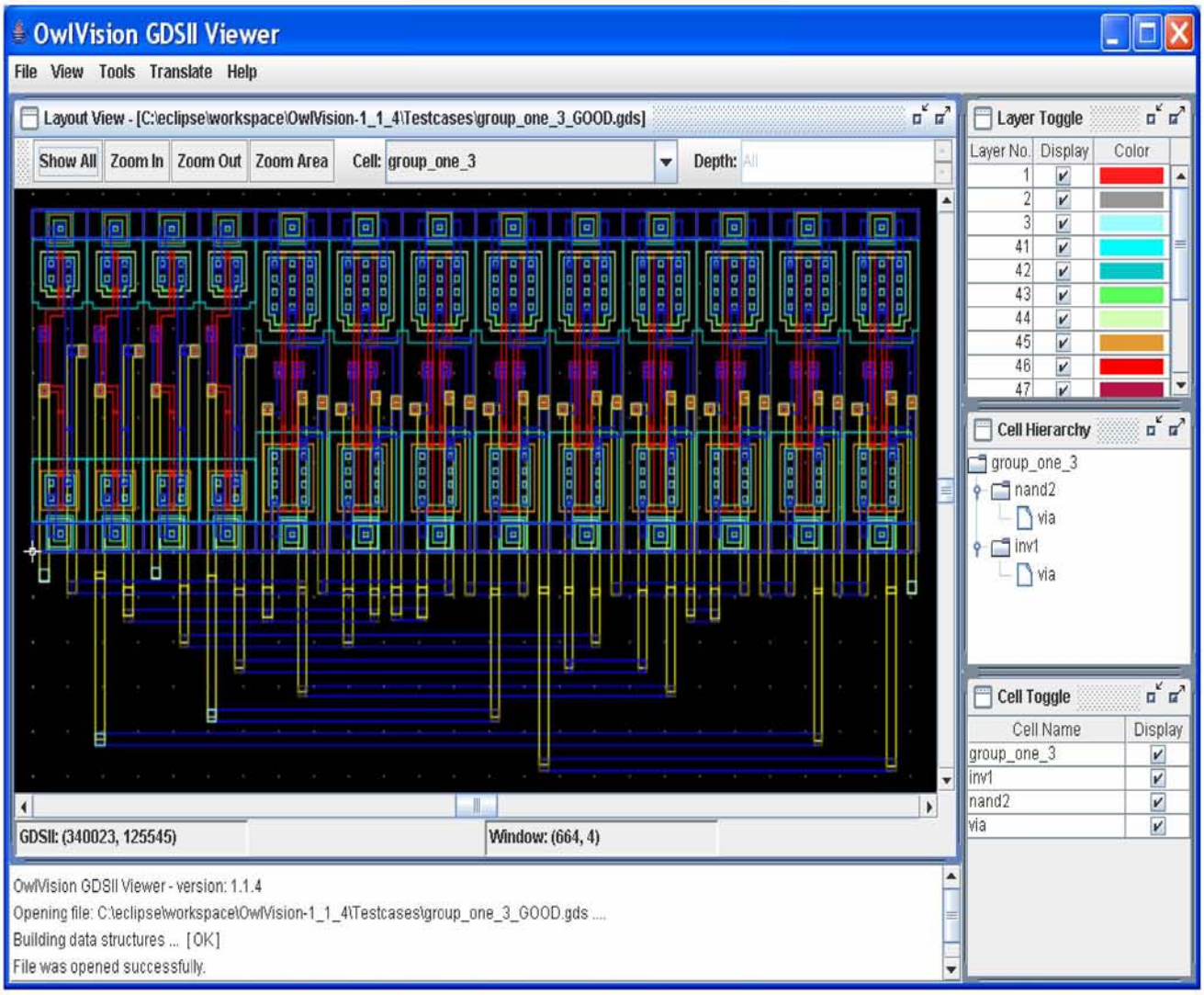
Οι βιβλιοθήκες που χρησιμοποιήσαμε είναι οι εξής:

- #include <stdio.h>
- #include <stdlib.h>
- #include <string.h>
- #include "lib/tran/transferfile.h"
- #include "lib/deco/decodefile.h"

Χρησιμοποιήθηκαν επίσης standard cell libraries και συγκεκριμένα βιβλιοθήκες της NandgateOpenCell Libraries.

Owl Vision GDSII Viewer"

Επιπροσθέτως, χρησιμοποιήθηκε και το λογισμικό "Owl Vision GDSII Viewer" το οποίο αποτελεί ένα πρόγραμμα φυσικής αναπαράστασης εξόδου (Layout Viewer) των ολοκληρωμένων κυκλωμάτων σε GDSII stream format. Το πρόγραμμα αυτό μας βοήθησε να κάνουμε εκτίμηση και σύγκριση των πυλών που αναπαρίσταντο πρίν και μέτα την επεξεργασία τους από τον κώδικα που υλοποιήσαμε.



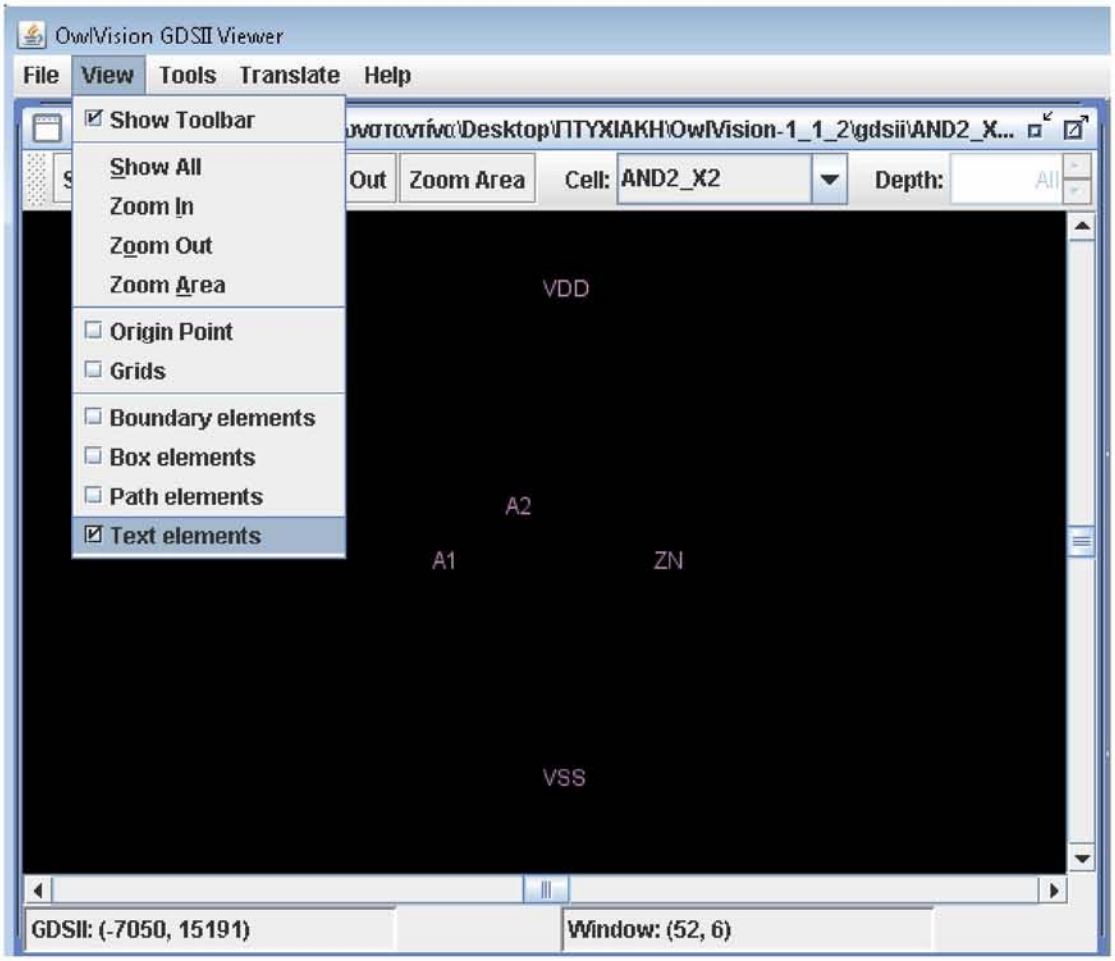
Εικόνα 5.1: Περιβάλλον λογισμικού "OwlVision GDSII Viewer"

Αναλυτικά, όπως φαίνεται και στην εικόνα κάθε layer έχει όνομα ενός αριθμού και χαρακτηρίζεται από ένα χρώμα όπου αντιστοίχηση του λαμβάνει χώρα στο παράθυρο "Layer Toggle". Παραδείγματος χάριν με το κόκκινο χρώμα αναπαρίστανται τα Layer 1, με το γρι χρώμα τα Layer 2, με το γαλάζιο τα Layer 3 κοκ.

Στο κάτω μέρος της αναπαράστασης φαίνονται οι συντεταγμένες κάθε σημείου όσο αναφορά το GDSII αλλά και το παράθυρο του layer viewer.

Το πρόγραμμα, επιπλέον έχει την δυνατότητα να κάνει μετάφραση ενός GDSII αρχείου σε ASCII και ανάποδα.

Τέλος, έχει την δυνατότητα να αναπαραστήσει συγκεκριμένα στοιχεία από την ακολουθία στοιχείων της δομής. Στην εικόνα παρακάτω, φαίνεται ένα παράδειγμα αναπαράστασης του "Text elements" του AND2_X2 cell.



Εικόνα 5.2: Αναπαράσταση του **Text Element** σε GDSII Layout Viewer

5.2 Ανάλυση Μεθοδολογίας

Στην ενότητα αυτή θα περιγράψουμε αναλυτικά τα βήματα του προγράμματος που υλοποιήθηκε, θα αναλύσουμε τις τεχνικές που χρησιμοποιήσαμε, τους λόγους εφαρμογής τους καθώς και τα προβλήματα που αντιμετωπίσαμε.

Το μεταεργαλείο που κληθήκαμε να δημιουργήσουμε, βασικό του στόχο έχει την ελαχιστοποίηση της κατανάλωσης ενέργειας σε standard cell. Οι μέθοδοι για την ελαχιστοποίηση της ενέργειας μπορεί να ποικίλουν και να πραγματοποιούνται με διάφορες τεχνικές όπως μέσω της μείωσης της τάσης τροφοδοσίας, της χωρητικότητας μεταγωγής και της συχνότητας λειτουργίας του ρολογιού. Εμείς αυτό το επιτυγχάνουμε μέσω της μείωσης του πλάτους των τρανζίστορ. Συγκεκριμένα ο σχεδιαστής είναι αυτός που θα επιλέξει πόσο θέλει να μικρύνει το μέγεθος, εφόσον αυτό είναι εφικτό, καθώς σε συγκεκριμένες σχεδιάσεις υπάρχουν κάποια όρια ως προς το πόσο μειώνουν την κατανάλωση ισχύος.

Η διεκπεραίωση της εργασίας χωρίστηκε σε πέντε βασικά μέρη τα οποία είναι:

1. Αποκωδικοποίηση των αρχείων GDSII
2. Διάβασμα του αρχείου και των δεδομένων, διαχωρισμός των layer και εισαγωγή των μεγεθών τους σε λίστες
3. Επεξεργασία του πλάτους των τρανζίστορ και τεχνικές προσαρμογής τους στο κύκλωμα
4. Αλλαγή των vias με αφαίρεση τους, αν κρίνεται απαραίτητο, και αλλαγή των υπολοίπων layers που εμπλέκονται
5. Κωδικοποίηση του νέου, επεξεργασμένου αρχείου με τα αλλαγμένα πλάτη σε αρχείο GDSII
6. Εκτίμηση, μετρήσεις και έλεγχος της βελτιστοποίησης του κυκλώματος όσο αναφορά την ελαχιστοποίηση της κατανάλωσης ισχύος

5.2.1 Αποκωδικοποίηση των αρχείων GDSII

Πρώτο μέλημα μας είναι να καταφέρουμε να διαβάσουμε τα GDSII αρχεία. Όπως αναλύσαμε και στο κεφάλαιο 3, πρόκειται για αρχεία σε δυαδική μορφή που αντιπροσωπεύουν τις επίπεδες γεωμετρικές μορφές, τις ετικέτες κειμένων, και άλλες πληροφορίες για το Layout με ιεραρχική δομή. Λόγω της δυαδικής μορφής του, τα GDSII αρχεία είναι δύσκολο να διαβαστούν και ο αναγνώστης έχει την δυνατότητα να δει τα περιεχόμενα του εκφρασμένα σε μορφή ASCII (μέσω KEY Format).

Παρακάτω θα παραθέσουμε ένα κομμάτι σε μορφή ASCII (Key format) του GDSII αρχείου "AND2_X2" μιας και σύμφωνα με αυτό έγινε και η υλοποίηση του κώδικα. Επειδή το μέγεθος του είναι αρκετά μεγάλο, θα δείξουμε ένα μικρό μέρος αυτού με την περιγραφή των δύο πρώτων layer (layer 3 και layer 2).

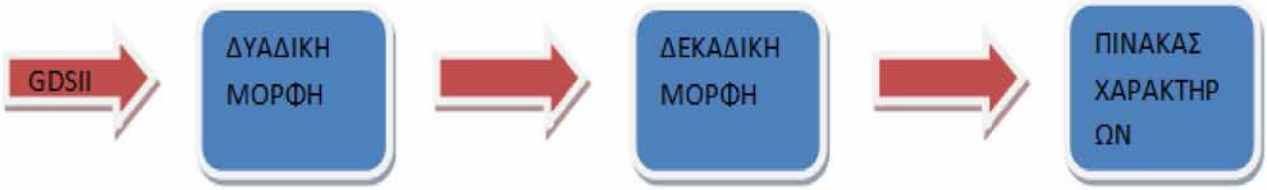
Αρχείο GDS της AND2_X2:

```
HEADER 3;  # version
BGNLIB;
  LASTMOD  {109-7-17  17:21:30};  # last modification time
  LASTACC  {109-7-17  17:21:30};  # last access time
LIBNAME NangateOpenCellLibrary;
UNITS;
  USERUNITS 1.0E-4;
  PHYSUNITS 1.0E-10;

BGNSTR;  # Begin of structure
  CREATION {109-7-17  17:21:30};  # creation time
```

```
  LASTMOD  {109-7-17  17:21:30};  # last modification time
STRNAME AND2_X1;

BOUNDARY;
LAYER 3;
DATATYPE 0;
XY 5;
  X: -1150;            Y: 5900;
  X: 8750;             Y: 5900;
  X: 8750;             Y: 15150;
  X: -1150;            Y: 15150;
  X: -1150;            Y: 5900;
ENDEL;

BOUNDARY;
\LAYER 2;
DATATYPE 0;
XY 5;
  X: -1150;            Y: -1150;
  X: 8750;             Y: -1150;
  X: 8750;             Y: 5900;
  X: -1150;            Y: 5900;
  X: -1150;            Y: -1150;
ENDEL;
```

Σκοπός μας λοιπόν είναι να καταφέρουμε να ανασύρουμε αυτές τις πληροφορίες από το αρχείο μας, να διαβάζονται και να επεξεργάζονται αυτόματα, καθώς η παραπάνω μορφή του ενδεικτικού μέρους είναι σε μορφή text.

Η τεχνική που ακολουθήσαμε ήταν να μετατρέψουμε το GDSII αρχείο από δυαδική μορφή σε δεκαδική και εν συνεχεία να το αποθηκεύσουμε σε έναν πίνακα χαρακτήρων. Οι υπόλοιπες συναρτήσεις που αφορούν τον διαχωρισμό των layers, την σμίκρυνση του πλάτους και της επεξεργασίας των δεδομένων έγιναν σύμφωνα με αυτόν τον πίνακα.



Βιβλιοθήκες "decodefile.h" και "transferfile.h"

Για την μεταφορά των αρχείων στην μνήμη δημιουργήσαμε την βιβλιοθήκη "transferfile"

#include "lib/tran/transferfile.h"

Και περιλαμβάνει τις παρακάτω συναρτήσεις:

int file_legth(char *);
int *transfer_file(char *);
int test_buffer(int *,int);

Για την διαδικασία αποκωδικοποίησης συγκεντρώσαμε όλες τις συναρτήσεις σε μια βιβλιοθήκη την οποία ονομάσαμε "decodefile"

#include "lib/deco/decodefile.h"

Και η οποία περιέχει τις εξής συναρτήσεις τις οποίες θα εξηγήσουμε παρακάτω:

int negativeNo(char *);
int bin2dec(char *,char *);
void dec2bin(long, char *);
char *decode_file(int *,int);
int test_data2use(char *,int);
int header_finder(char *,int,int);

- *Μεταφορά αρχείου στη μνήμη*

Ξεκινώντας την διαδικασία αποκωδικοποίησης, πρωταρχικό μέλημά μας είναι να μεταφέρουμε κάθε αρχείο GDSII που λαμβάνουμε σαν είσοδο στη μνήμη και να το επεξεργαζόμαστε από εκεί. Συγκεκριμένα, δίνοντας το όνομα του αρχείου, επιστρέφεται το μέγεθός του και δεσμεύεται η κατάλληλη μνήμη. Κατόπιν, τα περιεχόμενα του αρχείου αποθηκεύονται σε κελιά των 32 bit σε έναν καθολικό πίνακα που ονομάσαμε **"buffer"** και στη συνέχεια μεταφέρεται στην μνήμη. Έπειτα, γίνεται έλεγχος της ορθής μεταφοράς εκτυπώνοντάς τον σε ένα αρχείο text μορφή που έχουμε ονομάσει **"buffer_note".** Οι συναρτήσεις που υλοποιήσαμε για την διαδικασία μεταφοράς στην μνήμη περιγράφονται παρακάτω:

- **int file_legth(char *namegiven):**
  Η συνάρτηση αυτή λαμβάνει σαν όρισμα το όνομα ενός αρχείου και επιστρέφει το μέγεθος του.

- **int *transfer_file(char *namegiven):**
  Εδώ, δίνουμε σαν όρισμα το όνομα ενός αρχείου, και όλο τα δεδομένα του αποθηκεύονται σε έναν πίνακα **"buffer"**, σε κελιά μεγέθους 32 bit, ο οποίος με την σειρά του μεταφέρεται στην μνήμη.

- **int test_buffer(int *input_matix_buffer,int input_buffer_length):**
  Η συνάρτηση αυτή υλοποιήθηκε για τον έλεγχο της ορθής μεταφοράς του καθολικού πίνακα **"buffer"** στην μνήμη. Σκοπός της δημιουργίας της είναι η εκτύπωση του πίνακα σε ένα αρχείο μορφής κειμένου το οποίο ονομάσαμε **"buffer_note".**

  Η εκτύπωση των περιεχομένων του "buffer" στο αρχείο "buffer_note" έγινε σε δεκαεξαδική αναπαράσταση χάριν ευκολίας (λόγω του χαρακτήρα του GDSII format) και ένα μέρος (20 πρώτων θέσεων) του αρχείου φαίνεται στην παρακάτω εικόνα.

ΑΡΧΕΙΟ "buffer_note"

1. 02000600
2. 1c000300
3. 6d000201
4. 11000700
5. 15001100
6. 6d001e00
7. 11000700
8. 15001100
9. 1a001e00
10. 614e0602
11. 7461676e
12. 65704f65
13. 6c65436e
14. 62694c6c
15. 79726172
16. 05031400
17. 8bdb683d
18. b40c71ac
19. 7ff36d38
20. ecf65e67

Εικόνα 5.3: Αναπαράσταση περιεχομένων του αρχείου "buffer_note". Με κόκκινο φαίνονται οι θέσεις του πίνακα "buffer" ενώ με μαύρο τα περιεχόμενα του

- *Αποκωδικοποίηση αρχείου από την μνήμη*

Στην φάση αυτή καλούμαστε να αποκωδικοποιήσουμε τα στοιχεία του αρχείου **"buffer_note".** Όπως προείπαμε, αυτά είναι αποθηκευμένα στην καθολική μεταβλητή "**buffer**". Η τεχνική που ακολουθήσαμε είναι να διαχωρίσουμε τους binary αριθμούς των 32 bit σε τέσσερα κελιά των οχτώ bit και να τα αποθηκεύσουμε ένα-ένα σε byte σε έναν πίνακα χαρακτήρων σε μορφή integer, τον οποίο ονομάσαμε **"data2use".** Κατόπιν, δεσμεύουμε μνήμη για να μεταφέρουμε τον «επεξεργασμένο» **"buffer"** στον **"data2use"** και ελέγχουμε την ορθότητα της μεταφοράς εκτυπώνοντας τον σε ένα αρχείο κειμένου **"char_note".**
Οι συναρτήσεις που υλοποιήσαμε είναι οι εξής:

- **int negativeNo(char *binary):**
Πρόκειται για μια βοηθητική συνάρτηση και την υλοποιήσαμε για να έχουμε την δυνατότητα χειρισμού των αρνητικών αριθμών σε δυαδική αναπαράσταση.

- **void dec2bin(long decimal, char *CharByte):**
Η συνάρτηση αυτή λαμβάνει σαν ορίσματα έναν δεκαδικό αριθμό και έναν πίνακα χαρακτήρων. Μετατρέπει τον δεκαδικό αριθμό σε δυαδικό και στην συνέχεια κάθε ψηφίο 0 και 1 σε χαρακτήρες "0" και "1". Στην συνέχεια μετρά τον αριθμό των bits κάθε αριθμού και συμπληρώνει με μηδενικά μέχρις ότου να συμπληρωθεί το πλήθος των 32 ψηφίων. Στην συνέχεια καλείται η συνάρτηση "**bin2dec"** η οποία περιγράφεται αμέσως μετά.

- **int bin2dec(char *bin,char *CharByte):**
Η συνάρτηση αυτή λαμβάνει σαν ορίσματα έναν πίνακα από 32 ψηφία ενός δυαδικού αριθμού (που υπολογίστηκε από την "dec2bin") και έναν πίνακα χαρακτήρων. Στην συνέχεια χωρίζει τον αριθμό αυτό σε 4 μέρη των οχτώ bit και τα αποθηκεύει ανά οχτώ σε ένα byte. Κάθε byte αποθηκεύεται σε μία θέση πίνακα χαρακτήρων μεγέθους τεσσάρων θέσεων, τον οποίο ονομάσαμε **"CharByte".** Κάθε πίνακας **"CharByte"** αναπαριστά έναν αριθμό 32 bit χωρισμένο σε τέσσερα κελιά, με κάθε κελί να περιέχει έναν χαρακτήρα σε μορφή Integer, μεγέθους ενός byte (8 bit).

- **Char*decode_file(int*input_matix_buffer,int input_buffer_length)**
Στην συνάρτηση αυτή – καλώντας στο σώμα της τις παραπάνω συναρτήσεις- γίνεται η αποκωδικοποίηση του πίνακα **"buffer"** και δημιουργείται ο τελικός πίνακας "**data2use".** Ο πίνακας αυτός περιλαμβάνει όλα τα στοιχεία του **"buffer_note"** σε χαρακτήρες και συγκεκριμένα με κάθε θέση του να περιέχει ένα χαρακτήρα, μεγέθους ενός byte, σε μορφή integer (τον οποίο έχει λάβει από τον πίνακα "CharByte"). Κατόπιν ο **"data2use"** μεταφέρεται στην μνήμη για μετέπειτα επεξεργασία.

- **int test_data2use(char*input_matrix_data2use,int input_buffer_length):**
  Τέλος, η συνάρτηση αυτή υλοποιήθηκε για τον έλεγχο της ορθής μεταφοράς του καθολικού πίνακα **"data2use"** στην μνήμη. Σκοπός της δημιουργίας της είναι η εκτύπωση του πίνακα σε ένα αρχείο μορφής κειμένου το οποίο ονομάσαμε **"char_note".**

Το αποτέλεσμα στο οποίο καταλήξαμε ήταν το αρχείου εισόδου σε μορφή GDSII να μετατραπεί σε μια πιο ευανάγνωστη μορφή όπως φαίνεται στην παρακάτω εικόνα, το οποίο πλέον μας επιτρέπει να αναγνωρίσουμε τους header κάθε structure και συνεπώς να διαβάσουμε και να επεξεργαστούμε τα δεδομένα του . Παρακάτω φαίνεται ένα μέρος (20 πρώτων θέσεων) των περιεχομένων του τελικού αρχείου **"char_note".**

ΑΡΧΕΙΟ "char_note"

0. 0
1. 6
2. 0
3. 2
4. 0
5. 3
6. 0
7. 28
8. 1
9. 2
10. 0
11. 109
12. 0
13. 7
14. 0
15. 17
16. 0
17. 17
18. 0
19. 21
20. 0

Εικόνα 5.4: Αναπαράσταση περιεχομένων του αρχείου "char_note". Με κόκκινο φαίνονται οι θέσεις του πίνακα "data2use" ενώ με μαύρο τα περιεχόμενα του

5.2.2Εντοπισμός θέσης και διαχωρισμός των layers

Το επόμενο βήμα της υλοποίησης είναι η δημιουργία συναρτήσεων που θα μας δίνουν την δυνατότητα να διαβάσουμε το αρχείο μας και να εντοπίζουμε την θέση κάθε εγγραφής και το πλήθος τους μέσα στο αρχείο, εισάγοντας απλά και μόνο τον header της.

Στην συνέχεια, ο στόχος μας είναι διαχωρίζουμε τα layers που χρησιμοποιούνται από κάθε boundary και να τα εισάγουμε σε μία κύρια λίστα. Κάθε κόμβος (layer) της λίστας δείχνει με την σειρά του σε μια άλλη λίστα που περιέχει τις τιμές των δεδομένων (συντεταγμένες) που αντιστοιχούν σε κάθε layer.

- *Εντοπισμός θέσης*

Στο κομμάτι αυτό καλούμαστε να εντοπίσουμε τη θέση και το πλήθος των εγγραφών που περιγράφονται σε κάθε στοιχείο BOUNDARY (LAYER, DATATYPE, XY) μέσα στο αρχείο. Στο κεφάλαιο 3 στην εικόνα 3 κάναμε αναφορά στους header όλων των εγγραφών καθώς και στην αναπαράσταση των περιεχομένων ενός GDSII αρχείο σε μορφή ASCII (KEY format).

Πιο αναλυτικά, υλοποιήθηκαν οι συναρτήσεις:

- ***int layer_counter(char *input_matrix_data2use,int input_header1,int input_header2):***
  Η συνάρτηση αυτή, επιστρέφει το πλήθος των Layers που περιγράφονται από κάθε BOUNDARY στοιχείο μέσα στο αρχείο (συγκεκριμένα στον πίνακα **"data2use"**). Συγκεκριμένα μετρά τις φορές που θα συναντήσει τον header του στοιχείου BOUNDARY μέσα στο αρχείο, έως ότου τελειώσει η περιγραφή των Layers.

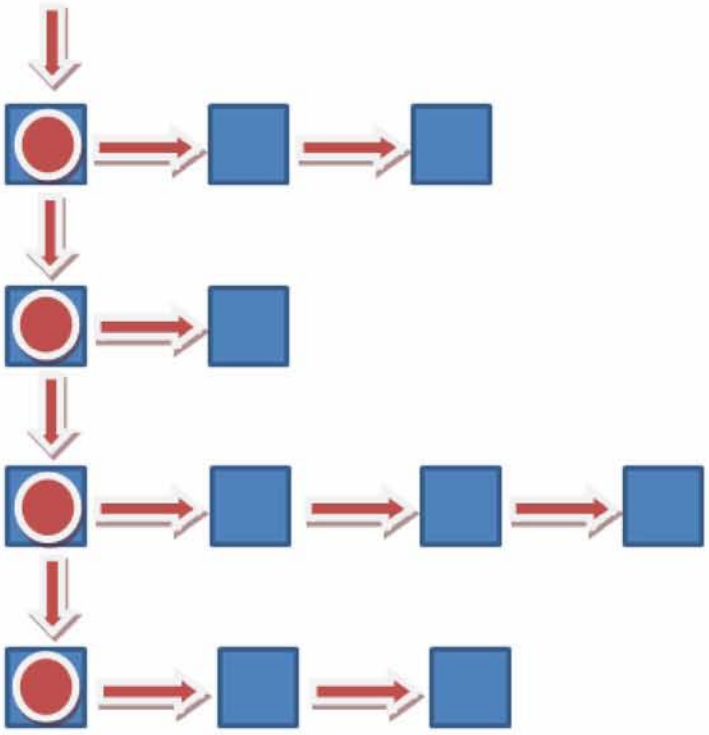
- ***int *find(char *input_matrix_data2use,int input_header1,int input_header2,int cnt_layer):***
  Η συνάρτηση αυτή υλοποιήθηκε για να επιστρέφει έναν πίνακα με όλες τις θέσεις των εγγραφών που υπάρχουν στο αρχείο, σύμφωνα με το πλήθος αυτών (που επιστρέφεται από την **"layer_counter"**) αλλά και τους header τους που εισάγουμε σαν όρισμα .

Γνωρίζοντας, το πλήθος και τις θέσεις όλων των εγγραφών του BOUNDARY element μπορεί να γίνει πλέον ο διαχωρισμός και η εισαγωγή τους σε λίστες.

- *Διαχωρισμός και εισαγωγή σε λίστες*

  Δημιουργούμε μία κύρια λίστα όπου κάθε κόμβος της περιέχει τις εγγραφές κάθε BOUNDARY και δείχνει σε μια επόμενη λίστα όπου περιέχει τις συντεταγμένες που καθορίζουν το πολύγωνο που σχετίζεται με κάθε Layer. Κάθε κόμβος περιέχει ένα ζεύγος συντεταγμένων X,Y.

  Η δομή των λιστών φαίνεται παρακάτω:



  Κάθε κουτί με κύκλο είναι δείκτης και περιέχει έναν σειριακό αριθμό, τον τύπο κάθε LAYER, την εγγραφή DATATYPE και την εγγραφή XY και των αριθμό ζεύγους των συντεταγμένων της. Κάθε άλλο κουτί περιέχει ένα ζεύγος συντεταγμένων που σχετίζεται με το περίγραμμα του πολυγώνου που περιγράφεται στο εκάστοτε BOUNDARY element.

  Οι συναρτήσεις που υλοποιήθηκαν για τον διαχωρισμό των LAYERS και την εισαγωγή των δεδομένων τους σε λίστες είναι οι εξής:

- ***int *x_values(char *input_matrix_data2use,int xy_val,int t,int *position_finder):***
  Η συνάρτηση αυτή, υλοποιήθηκε βοηθητικά, και αποθηκεύει σε ένα πίνακα όλες τις τιμές των συντεταγμένων X σε δεκαδικό, αφού πρώτα τις έχουμε μετατρέψει από χαρακτήρες.

- ***int *y_values(char *input_matrix_data2use,int xy_val,int t,int *position_finder):***
  Η συνάρτηση αυτή, υλοποιήθηκε βοηθητικά, και αποθηκεύει σε ένα πίνακα όλες τις τιμές των συντεταγμένων Y σε δεκαδικό, αφού τις έχουμε μετατρέψει από χαρακτήρες.

- ***int list_organisation(char *data2use)***
  Στην συνάρτηση αυτή γίνεται όλη η οργάνωση των λιστών. Μέσω των γνωστών δομών της αρχικοποίησης και εισαγωγής σε λίστα δημιουργούμε την κύρια λίστα **"layer_pointer_list"** η οποία περιέχει έναν σειριακό αριθμό, τον τύπο του LAYER, του DATATYPE και την εγγραφή XY με τον αριθμό των ζεύγους συντεταγμένων που περιέχει. Στην συνέχεια υλοποιείται η λίστα **"layer_data_list"** που δείχνει κάθε κόμβος της κύριας που περιέχει όλες τις συντεταγμένες του πολυγώνου που περιγράφονται σε κάθε LAYER. Η δομή αυτή μεταφέρεται στην μνήμη για μετέπειτα επεξεργασία.

- ***int list_organisation_print()***
  Εδώ γίνεται ο έλεγχος της ορθής μεταφοράς στην μνήμη μέσω εκτύπωσης των λιστών σε ένα αρχείου κειμένου το οποίο ονομάσαμε **"lista.txt".**

Η μορφή των λιστών που λαμβάνουμε από το αρχείο "lista.txt" και περιέχει όλες τις απαιτούμενες τιμές και τύπους εγγραφών ένας μέρος της (2 πρώτα LAYER) φαίνεται παρακάτω:

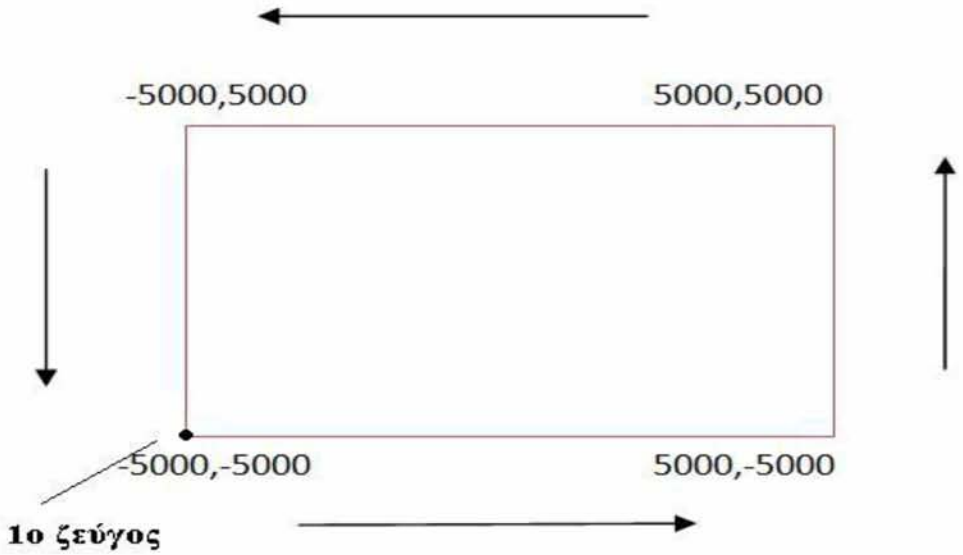
ΑΡΧΕΙΟ "lista.txt"

```
Serial: 0
Layer: 3
Datatype: 0
XY: 5
     dataserial:0 X:400 Y:1650
     dataserial:1 X:3000 Y:1650
     dataserial:2 X:3000 Y:2550
     dataserial:3 X:400 Y:2550
     dataserial:4 X:400 Y:1650

Serial: 1
Layer: 2
Datatype: 0
XY: 5
     dataserial:0 X:400 Y:11450
     dataserial:1 X:3000 Y:11450
     dataserial:2 X:3000 Y:12800
     dataserial:3 X:400 Y:12800
     dataserial:4 X:400 Y:11450
```

Εικόνα 5.5: Αναπαράσταση περιεχομένων του αρχείου "lista.txt". Με κόκκινο φαίνονται τα στοιχεία των κόμβων της "layer_pointer_list" ενώ με μαύρο τα στοιχεία της "data_layer_pointer_list"

5.2.3Ελαχιστοποίηση πλάτους των τρανζίστορ

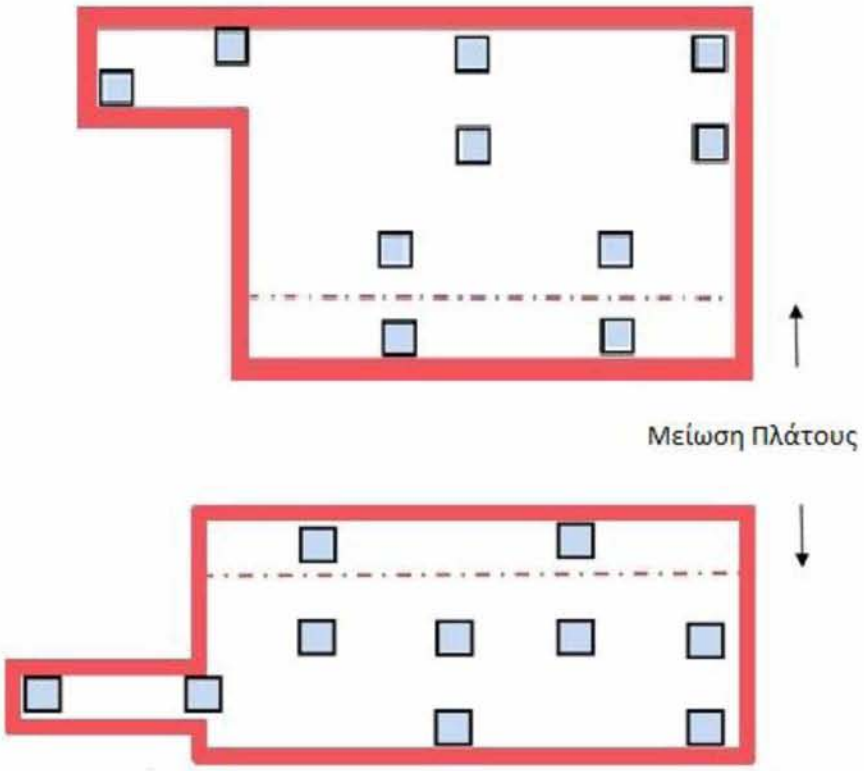
Στο τρίτο κεφάλαιο αναφέραμε ότι η εγγραφή XY περιέχει το πλήθος ζευγών συντεταγμένων που περιγράφουν ένα πολύγωνο. Επιπλέον, το τελευταίο ζεύγος συντεταγμένων πρέπει να συμπίπτει με το πρώτο ζεύγος για να εγγυηθούμε τα κλειστά όρια του πολυγώνου. Συγκεκριμένα, η περιγραφή του πολυγώνου ξεκινάει από τα κάτω αριστερά και κινείται αριστερόστροφα όπως φαίνεται στο παρακάτω σχήμα:



Εικόνα 5.7: Αριστερόστροφη φορά κατά την περιγραφή του πολυγώνου, ξεκινώντας από το κάτω αριστερά ζεύγος

Στην ενότητα αυτή θα αναλύσουμε την τεχνική που ακολουθήσαμε για την ελαχιστοποίηση του πλάτους των τρανζίστορ. Ο σχεδιαστής είναι αυτός που θα καθορίσει το μέγεθος της ελαχιστοποίησης, αναλόγως την περίπτωση και την πύλη που χρησιμοποιεί.

Στο κεφάλαιο τέσσερα εξηγήσαμε ότι το πλάτος των τρανζίστορ πρέπει να μειώνεται με φορά από μέσα προς τα έξω για να αποφύγουμε τον κίνδυνο οι τάσεις τροφοδοσίας και τα μέταλλα να βγουν εκτός επιφάνειας. Στο πηγάδι τύπου n μειώνουμε το πάνω μέρος από πάνω προς τα κάτω ενώ το πηγάδι τύπου p το κάτω μέρος από κάτω προς τα πάνω όπως φαίνεται στο παρακάτω σχήμα.

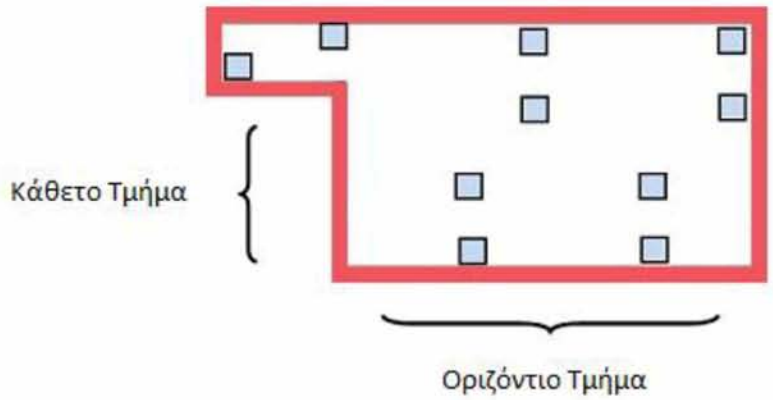


Εικόνα 5.8: Μείωση από κάτω προς τα πάνω στο τύπου n και από κάτω προς τα πάνω στο τύπου p.

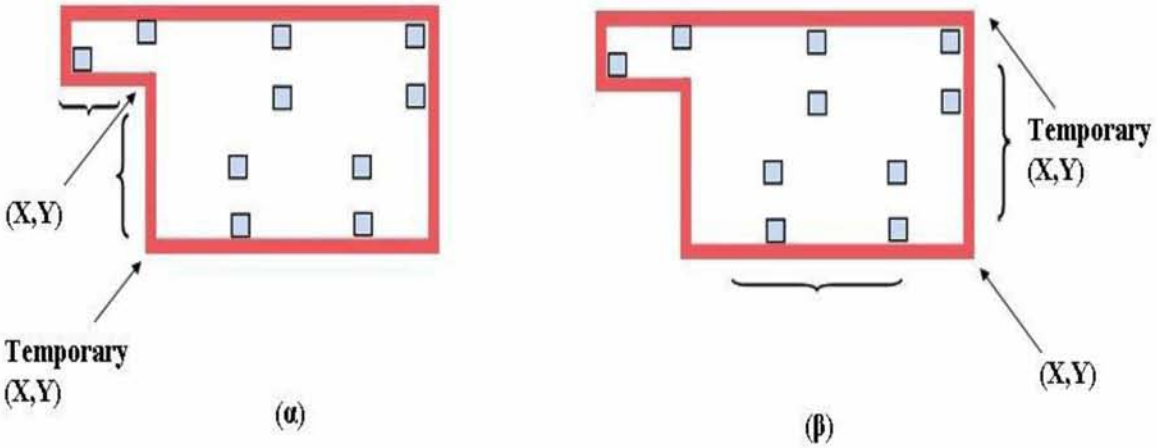
Σε κάθε layer που βρισκόμαστε, εξετάζουμε τις συντεταγμένες που περιγράφουν το πολύγωνο με τον εξής τρόπο:

Ένας δείκτης διατηρεί το πρώτο ζεύγος συντεταγμένων κι ένα άλλος temporary δείκτης διατηρεί το αμέσως επόμενο ζεύγος. Κάθε φορά αφού ορίσουμε το block στο οποίο θα γίνει ελαχιστοποίηση πρέπει να γίνεται έλεγχος των Vias που λαμβάνουν χώρα και κατά πόσο είναι δυνατή η αφαίρεση τους από το block. Τα κριτήρια και οι συνθήκες για την διαδικασία αυτή περιγράφονται αναλυτικά στην αμέσως επόμενη ενότητα.

- Για τρανζίστορ **τύπου p**
  Κάθε φορά πρέπει να ελέγχουμε τα ευθύγραμμα τμήματα που σχηματίζουν το πολύγωνο. Όταν από ένα οριζόντιο τμήμα ακολουθεί κάθετο τμήμα με φορά προς τα κάτω, δεν μπορούμε να προβούμε σε μείωση πλάτους **(εικόνα 5.10(α)).** Η μείωση μπορεί να γίνει με αντικατάσταση των συντεταγμένων (στο μέγεθος που έχει ορίσει ο σχεδιαστής) σε ένα ή περισσότερα από τα κάτω οριζόντια τμήματα του πολυγώνου τα οποία ακολουθούνται από ένα κάθετο τμήμα με φορά προς τα πάνω **(εικόνα 5.10(β)).**



Εικόνα 5.9: Τμήματα ενός πολυγώνου. Αναπαράσταση τρανζίστορ με κόκκινο, vias με γαλάζιο



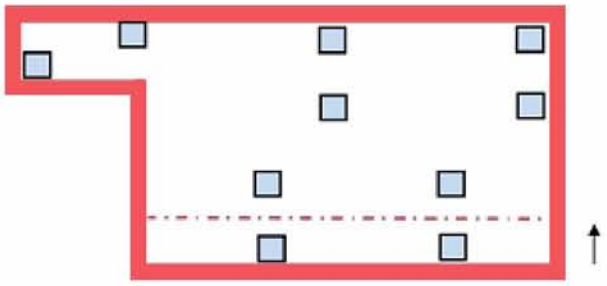
Εικόνα 5.10: **(α)** Στο οριζόντιο τμήμα της εικόνας δεν μπορούμε να παρέμβουμε εφόσον ακολουθεί κάθετο τμήμα με φορά προς τα κάτω

(β) Στο οριζόντιο τμήμα της εικόνας μπορούμε να παρέμβουμε εφόσον ακολουθεί κάθετο τμήμα με φορά προς τα πάνω

Εάν η τιμή Y του δείκτη ταυτίζεται με την τιμή Y του δείκτη που έπεται και η τιμή του X αντίστοιχα μειώνεται ή αυξάνεται σημαίνει ότι είμαστε σε οριζόντιο τμήμα του πολυγώνου. Επειδή όμως στα τρανζίστορ τύπου p η μείωση γίνεται από κάτω προς τα πάνω, μας αφορά το οριζόντιο τμήμα του πολυγώνου όπου η τετμημένη αυξάνεται κατά μήκος της ευθείας από τα αριστερά προς τα δεξιά.

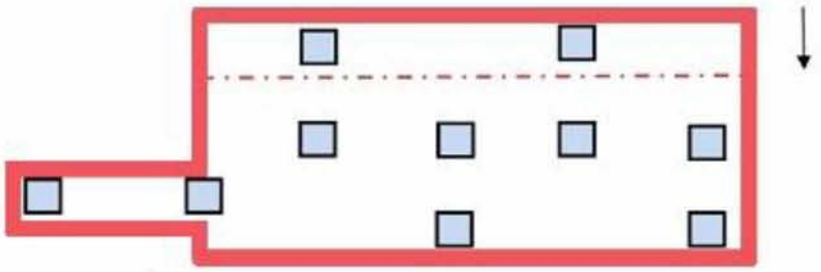
Για να ορίσουμε το κατώτερο οριζόντιο τμήμα του πολυγώνου πρέπει να προσδιορίσουμε τα κάθετα τμήματα που προηγούνται και έπονται αυτού, ως εξής:

Εάν η τιμή X του δείκτη ταυτίζεται με την τιμή X του δείκτη που έπεται ενώ η τιμή Y αντίστοιχα μειώνεται σημαίνει ότι βρισκόμαστε σε κάθετο τμήμα του πολυγώνου με φορά προς τα κάτω. Εάν η τιμή του Y αυξάνεται βρισκόμαστε σε κάθετο τμήμα πολυγώνου με φορά προς τα πάνω. Στο οριζόντιο τμήμα που παρεμβάλλεται σε δύο τέτοια ευθύγραμμα τμήματα (με το X να αυξάνεται κατά μήκος της ευθείας) μας επιτρέπεται να επέμβουμε. Στην συνέχεια, αφού γίνει ο έλεγχος και η πιθανή αφαίρεση των vias, αντικαθιστούμε τις παλιές τιμές Y που ορίζουν την ευθεία με τις καινούργιες τιμές -που περιλαμβάνουν το μέγεθος για μείωση που επιθυμεί ο σχεδιαστής- και το πλάτος του τρανζίστορ μειώνεται.



Εικόνα 5.10: Μείωση τρανζίστορ του κάθετου τμήματός του από κάτω προς τα πάνω

- Για τρανζίστορ **τύπου n**
  Ο προσδιορισμός των οριζόντιων και κάθετων τμημάτων, εδώ, γίνεται ομοίως με πριν με την διαφορά ότι τα τρανζίστορ τύπου n θα πρέπει να τα μειώσουμε με φορά από πάνω προς τα κάτω. Συνεπώς, μας ενδιαφέρει να προσδιορίσουμε τα ανώτερα οριζόντια τμήματα του πολυγώνου (με φθίνουσα τετμημένη κατά μήκος της ευθείας από τα δεξιά προς τα αριστερά).
  Ένα τέτοιο ευθύγραμμο τμήμα θα πρέπει να παρεμβάλλεται ανάμεσα από ένα κάθετο τμήμα με φορά προς τα πάνω (X σταθερή, Y αύξουσα) και από ένα κάθετο τμήμα με φορά προς τα κάτω (X σταθερή, Y φθίνουσα). Στην συνέχεια προχωράμε σε μείωση του πλάτους, εφόσον μας επιτρέπεται από τον έλεγχο που κάνουμε για τα vias, με την διαδικασία που περιγράψαμε παραπάνω, αντικαθιστώντας τις παλιές τιμές Y που ορίζουν την οριζόντια ευθεία με τις καινούργιες τιμές που περιλαμβάνουν το μέγεθος για μείωση που επιθυμεί ο χρήστης.



Εικόνα 5.11:Μείωση τρανζίστορ του κάθετου τμήματός του από κάτω προς τα πάνω

Η συνάρτηση που υλοποιήθηκε για την μείωση του πλάτους των τρανζίστορ είναι η εξής:

- ***int reduce_transistor_width(int number_to_reduce):***

  Η συνάρτηση αυτή παίρνει σαν όρισμα το μέγεθος της μείωσης του πλάτους που καθορίζεται από τον χρήστη. Χρησιμοποιείται ένας δείκτης που κρατά το πρώτο ζεύγος συντεταγμένων (X,Y) και ακόμα ένας προσωρινός δείκτης που κρατά το αμέσως επόμενο ζεύγος. Οι τιμές των X,Y λαμβάνονται από την δομή "data_layer_pointer_list". Στην συνέχεια, γίνεται έλεγχος των τιμών που διατηρούν οι δύο δείκτες, προσδιορίζονται τα κάθετα και οριζόντια τμήματα του τρανζίστορ και προχωρά στη μείωση του πλάτους εφόσον αυτό είναι επιτρεπτό -όπως αναλύθηκε παραπάνω.

Στην συνέχεια, το τελευταίο βήμα αφορά την αφαίρεση των vias στον χώρο που ορίσαμε για ελαχιστοποίηση του πλάτους. Τα κριτήρια και η τεχνική που ακολουθήσαμε περιγράφεται στην αμέσως επόμενη ενότητα.

5.2.4 Αφαίρεση των Vias

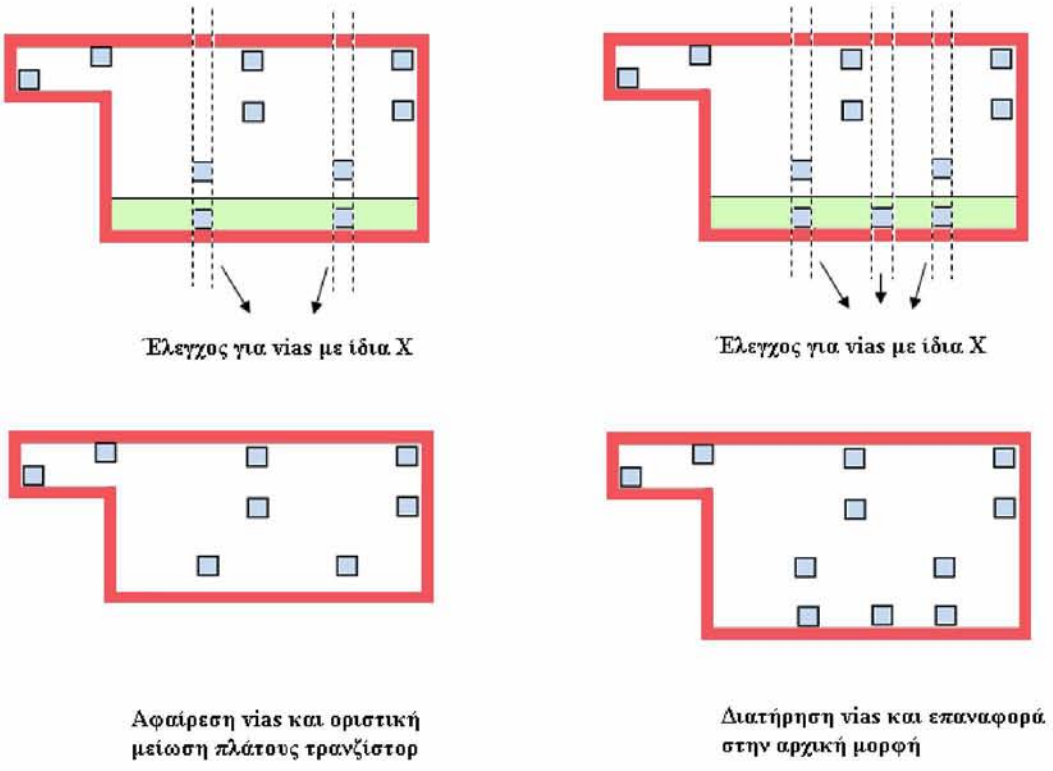
Έχοντας ορίσει πλέον το Block που μας επιτρέπεται να γίνει ελαχιστοποίηση μέσα από την ***"reduce_transistor_width"*** υλοποιούμε μια συνάρτηση υπεύθυνη για τον έλεγχο ύπαρξης vias στην περιοχή, την αφαίρεση τους -αν κρίνεται απαραίτητο- και αλλαγή των υπολοίπων layers που εμπλέκονται.

Η διαδικασία που ακολουθήσαμε έχει ως εξής:

Σε πρώτη φάση διατηρούμε τις αρχικές συντεταγμένες που περιγράφουν το πολύγωνο για πιθανή επαναφορά του σε περίπτωση που δεν μας επιτρέπεται η μείωση του πλάτους μετά τον έλεγχο των vias. Ο έλεγχος γίνεται στο Block που αφαιρέθηκε απο το τρανζίστορ το οποίο προέκυψε από την συνάρτηση ***"reduce_transistor_width"***. Εάν εντοπιστούν vias μέσα στο block, για να επιβεβαιωθεί η ορθότητα της μείωσης του πλάτους του τρανζίστορ πρέπει να ισχύει το εξής:

Για κάθε vias που λαμβάνει χώρα στο αποκομμένο τμήμα (block) θα πρέπει να υπάρχει τουλάχιστον ένα Vias με τις ίδιες τετμημένες (X,X') το οποίο να περιλαμβάνεται μέσα στα όρια του πολυγώνου -που έχουν οριστεί μετά την μείωση του πλάτους του. Εφόσον αυτό ισχύει, τα vias αφαιρούνται και οριστικοποιούνται οι νέες διαστάσεις του μειωμένου τρανζίστορ. Σε αντίθετη περίπτωση, εάν εντοπιστεί έστω και ένα vias στο αποκομμένο τμήμα που δεν πληροί την παραπάνω προδιαγραφή δεν πραγματοποιείται μείωση του πλάτους και το τρανζίστορ επιστρέφει στην αρχική του μορφή **(Εικόνα 5.12).**

Ο λόγος που πρέπει να ισχύουν οι παραπάνω περιορισμοί, όπως επεξηγήθηκε και στο κεφάλαιο τέσσερα, είναι επειδή πρέπει να υπάρχει ένα τουλάχιστον vias για κάθε τάση τροφοδοσίας, προκειμένου να επιτευχθεί ορθή λειτουργία του κυκλώματος.



Εικόνα 5.12: Έλεγχος για αφαίρεση vias και για οριστική μείωση του πλάτους των τρανζίστορ.

Η συνάρτηση που υλοποιήθηκε για τον έλεγχο και την αφαίρεση των vias είναι η εξής:

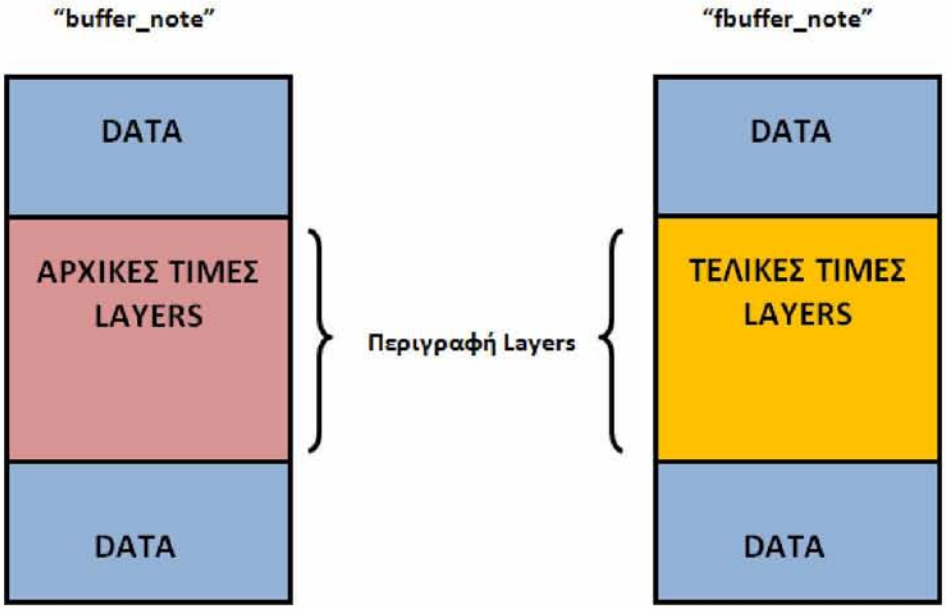
- ***int remove_vias_transistor()***

Η συνάρτηση αυτή διατηρεί τις συντεταγμένες (X,Y) από τα αρχικά όρια του πολυγώνου προς επεξεργασία και εν συνεχεία κάνει έλεγχο ύπαρξης Vias στο στα όρια του αποκομμένου τμήματος. Γνωρίζοντας τις συντεταγμένες του αποκομμένου τμήματος και των vias ελέγχεται κατά πόσο τα vias βρίσκονται μέσα στα όρια της αποκομμένης περιοχής . Για καθένα από αυτά, εξετάζεται εάν υπάρχει vias με τις ίδιες τιμές X μέσα στα όρια του μειωμένου τρανζίστορ και προβαίνει σε αφαίρεση των vias- εάν επιτρέπεται- και οριστική μείωση ή επαναφορά των διαστάσεων του πολυγώνου. Οι κόμβοι των layers που περιγράφουν τα vias που αφαιρέθηκαν αποκόπτονται από την δομή "layer_pointer_list" ενώ οι τελικές τιμές των συντεταγμένων του εκάστοτε πολυγώνου αντικαθίστανται στην δομή " layer_data_list".

5.2.5 Κωδικοποίηση σε αρχείο GDSII

Το τελευταίο βήμα της μεθοδολογίας είναι η δημιουργία ενός νέου αρχείου GDSII που θα περιέχει τις νέες τιμές συντεταγμένων και τα αλλαγμένα layers.
Η διαδικασία έχει ως εξής:

Δημιουργούμε ένα νέο αρχείο το οποίο περιέχει όλα τα δεδομένα του αρχικού πίνακας **"buffer"** μέχρι πριν την περιγραφή των layers (μέχρι τον πρώτο header της εγγραφής LAYER) και τα δεδομένα που υπάρχουν μετά την περιγραφή των layers. Στην συνέχεια αντιγράφουμε στο νέο αρχείο και ανάμεσα σε αυτά τα δεδομένα τις νέες τιμές των layers που έχουμε διατηρήσει στις δομές λιστών. Τα νέα δεδομένα μεταφέρονται στην μνήμη και ο έλεγχος ορθής μεταφοράς γίνεται από το αρχείο μορφής "text" το οποίο ονομάσαμε **"fbuffer_note".** Το νέο αρχείο **"fbuffer_note"** είναι της μορφής GDSII, εφόσον δεν προβήκαμε ενδιάμεσα σε κάποια μορφοποίηση του πίνακα **"buffer",** και έχουμε πλέον την δυνατότητα να το χρησιμοποιήσουμε σαν αρχείο εισόδου σε οποιοδήποτε πρόγραμμα φυσικής αναπαράστασης εξόδου (Layout Viewer) ολοκληρωμένων κυκλωμάτων σε GDSII stream format.



Εικόνα 5.13: Αναπαράσταση δημιουργίας τελικού αρχείου gdsii με τις νέες τιμές των layers

Η συνάρτηση που υλοποιήσαμε για την πραγματοποίηση των παραπάνω είναι η εξής:

- ***int encode_file ():***
  Είναι υπεύθυνη για την αντιγραφή των δεδομένων που λαμβάνουν χώρα πριν και μετά την περιγραφή των layers από τον πίνακα **"buffer"**, την αντικατάσταση των νέων αλλαγμένων layers, την μεταφορά τους στην μνήμη και τον έλεγχο αυτής μέσω του αρχείου **"fbuffer_note".**

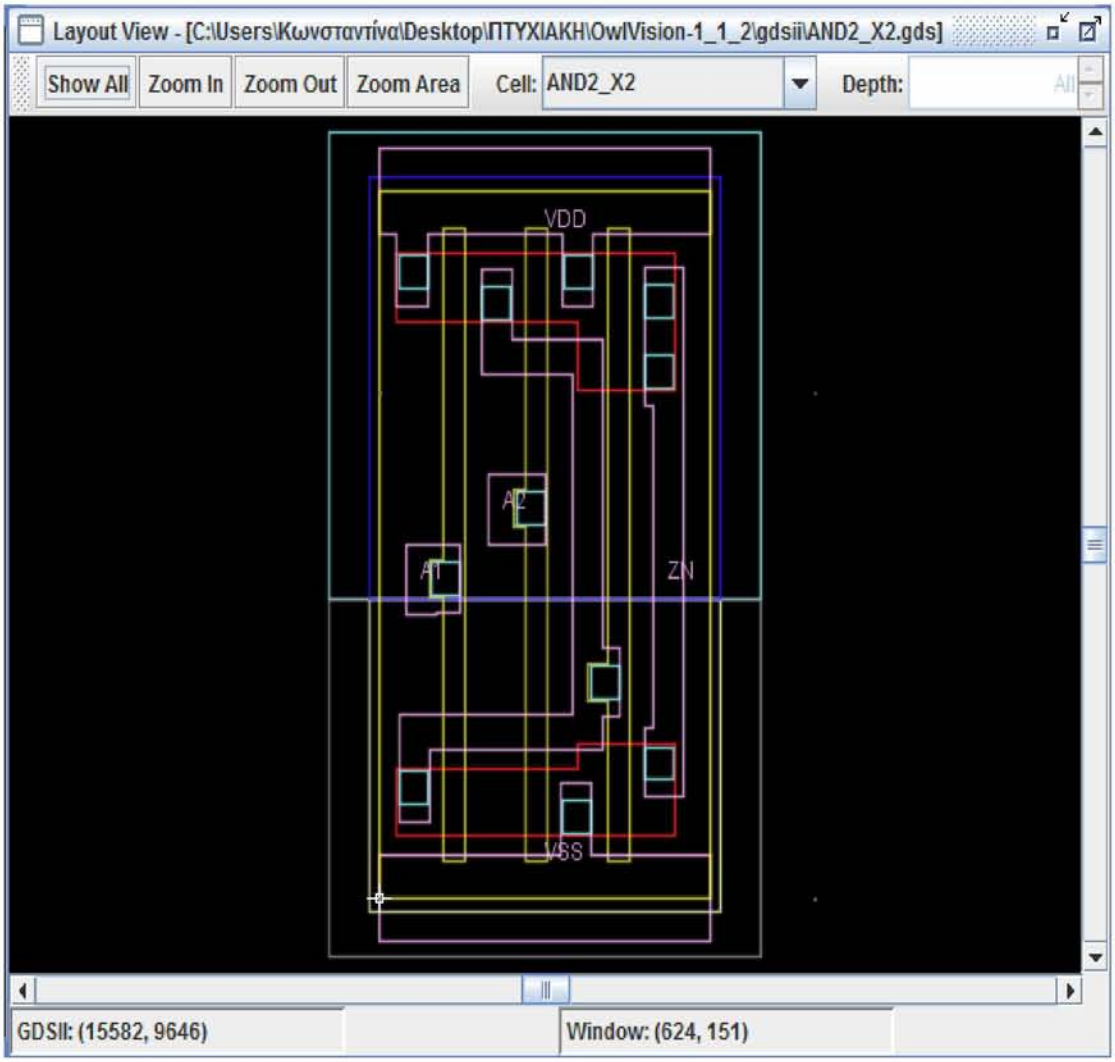
5.3 Συμπεράσματα Ανάλυσης Μεθοδολογίας

Κλείνοντας την ανάλυση του προγράμματος, καταφέραμε να κάνουμε parse τα GDSII αρχεία που λάβαμε σαν είσοδο, να αλλάξουμε τα πλάτη των τρανζίστορ και να αφαιρέσουμε τα vias, μετά από κατάλληλους ελέγχους και εφόσον αυτό ήταν επιτρεπτό. Τέλος μεταφέραμε τις αλλαγές σε ένα νέο GDSII αρχείο το οποίο μπορούμε πλέον να το χρησιμοποιήσουμε σαν νέο αρχείο εισόδου σε προγράμματα αναπαράστασης εξόδου για την πραγματοποίηση των σχετικών μετρήσεων.

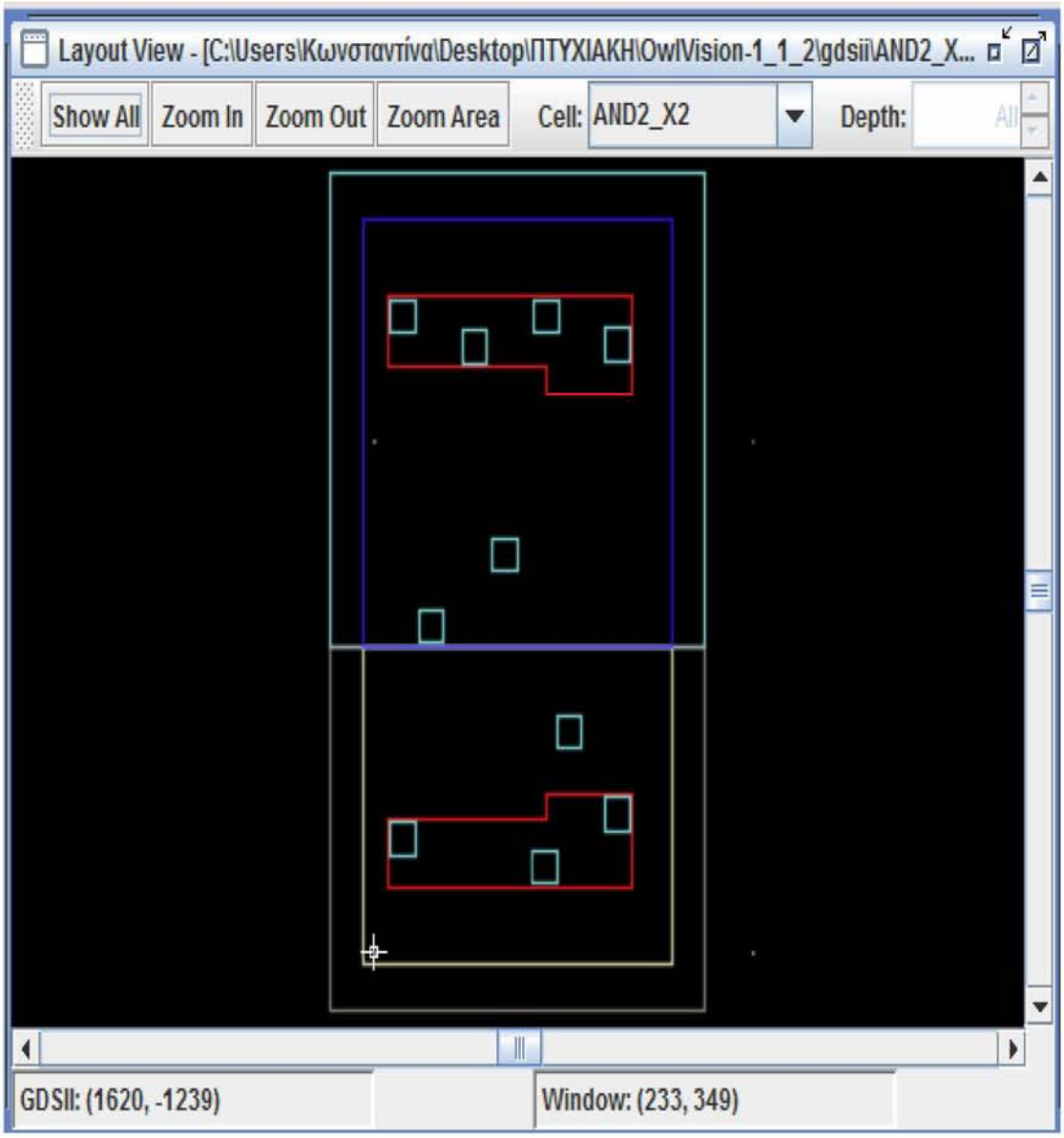
6 ΠΕΙΡΑΜΑΤΙΚΕΣ ΜΕΤΡΗΣΕΙΣ

Παρακάτω παρατίθενται κάποια παραδείγματα του εργαλείου που αναπτύξαμε. Θα παρουσιάσουμε την εφαρμογή του προγράμματος σε κάποιες πύλες που περιγράφονται μέσω GDSII αρχείων της βιβλιοθήκης NandGateOpenCell . Το εργαλείο για την φυσική αναπαράσταση εξόδου είναι το Owl Vision Gdsii Viewer. Συγκεκριμένα, δείχνουμε το layout των πυλών πριν και μετά την μείωση πλάτους των τρανζίστορ. Επίσης γίνεται αναφορά και στις «χειρότερες» περιπτώσεις πυλών όπου το εργαλείο μας δεν επιτρέπεται να επέμβει στις διαστάσεις του κυκλώματος λόγω περιορισμού σχεδιαστικών κανόνων ή μη ορθότητας λειτουργίας του κυκλώματος .

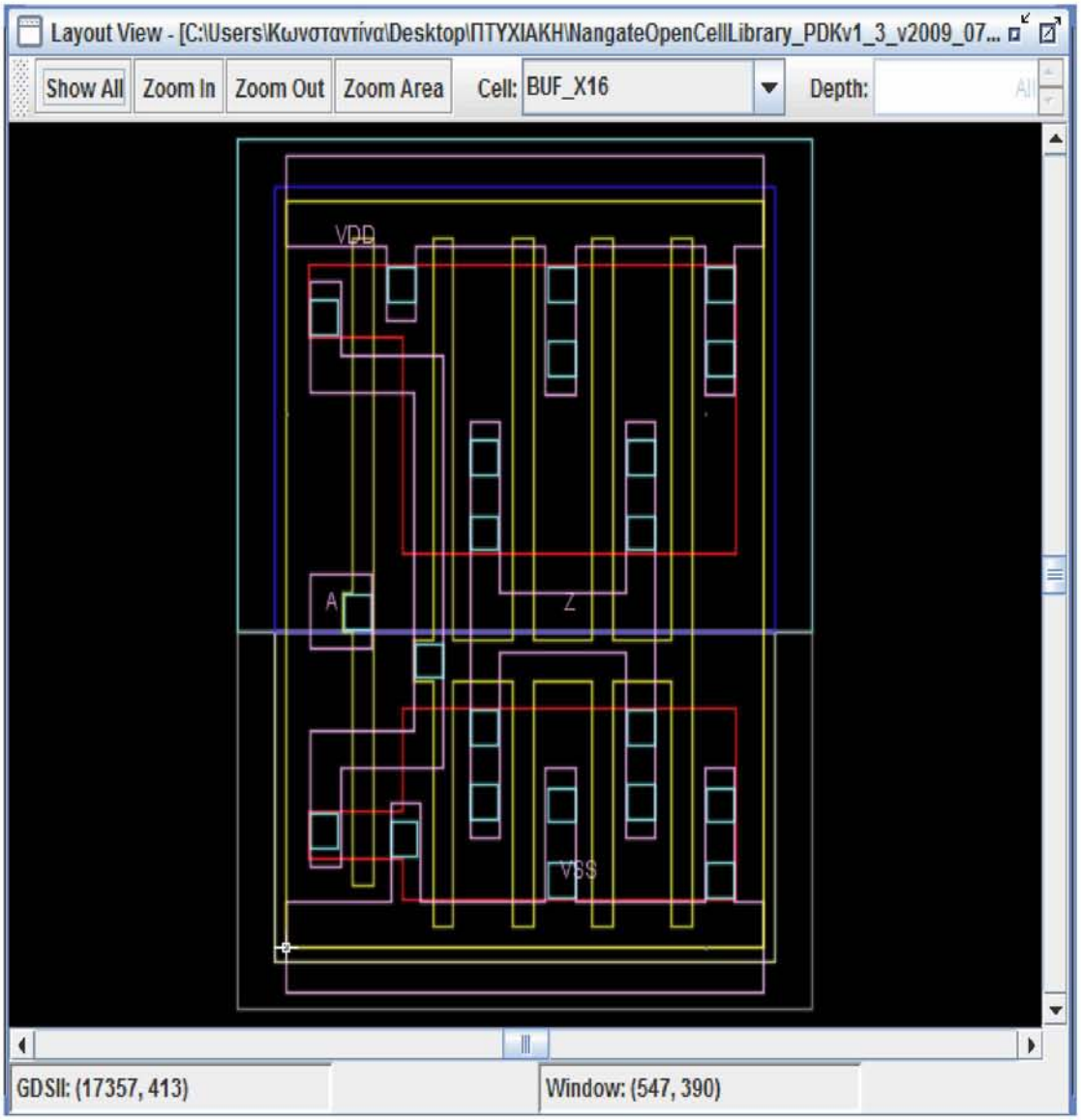
- GDSII αρχείο “AND2_X2”:



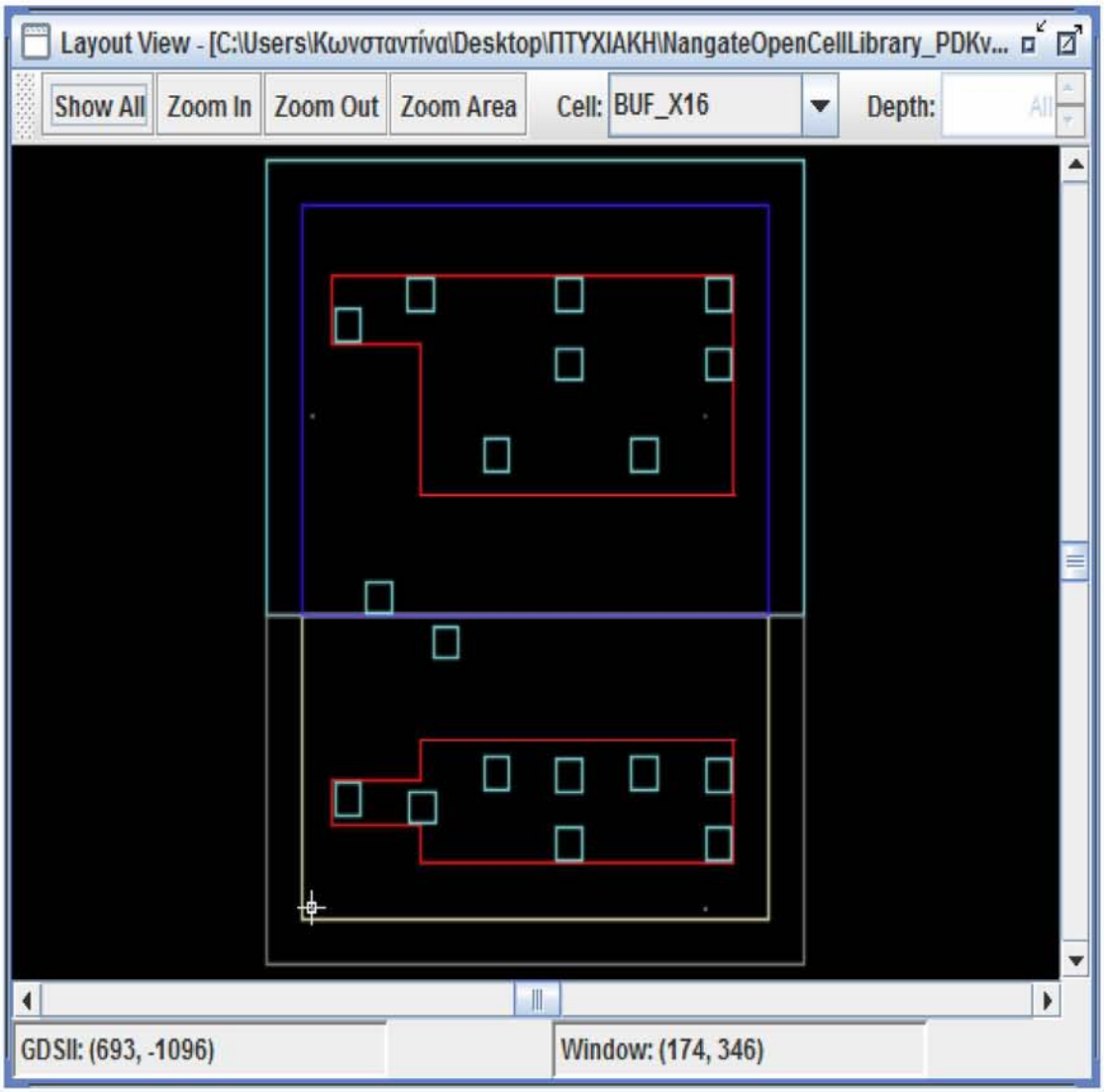
Τελικό layout με μειωμένα πλάτη



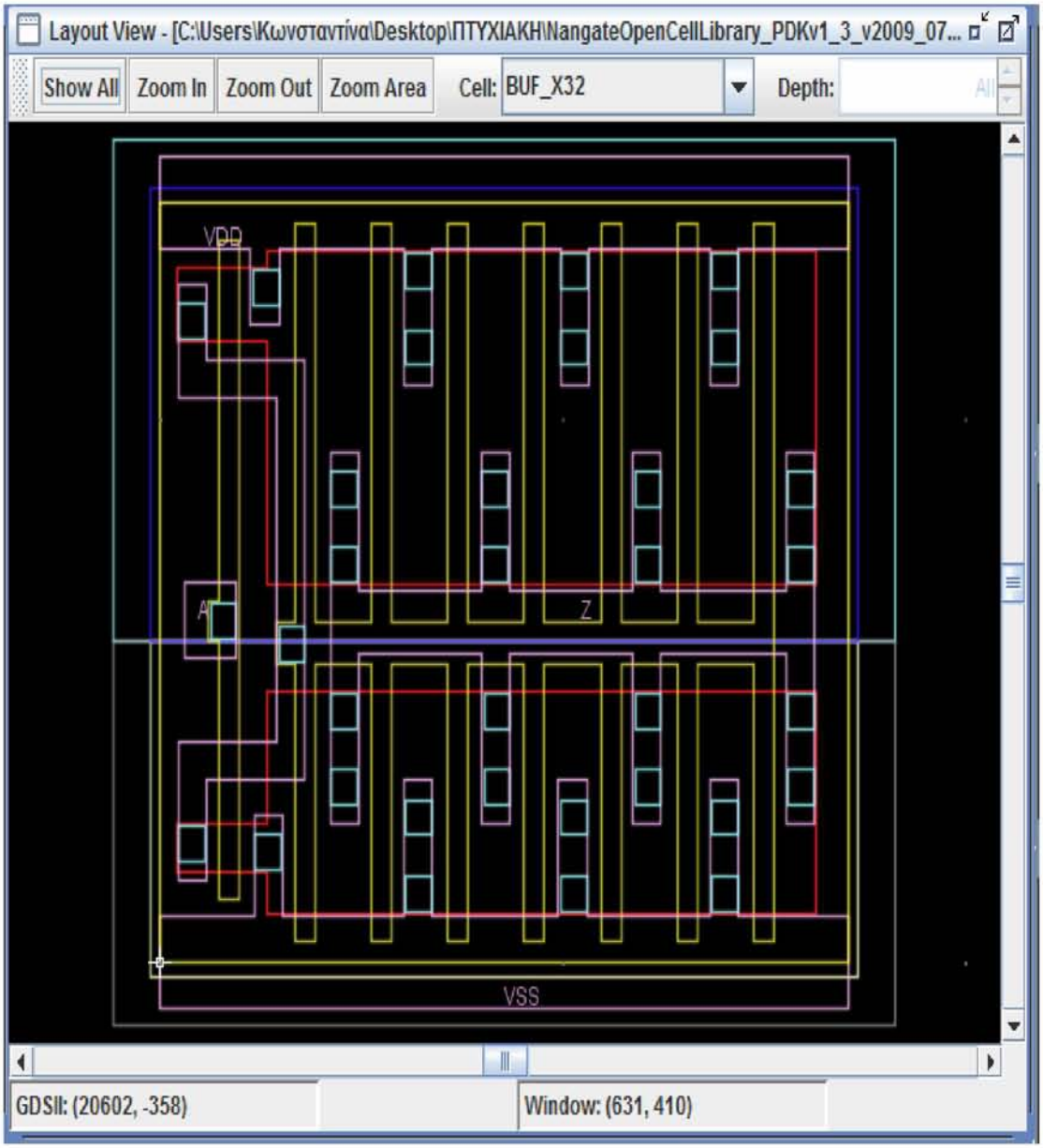
- GDSII αρχείο “BUF_X16”:



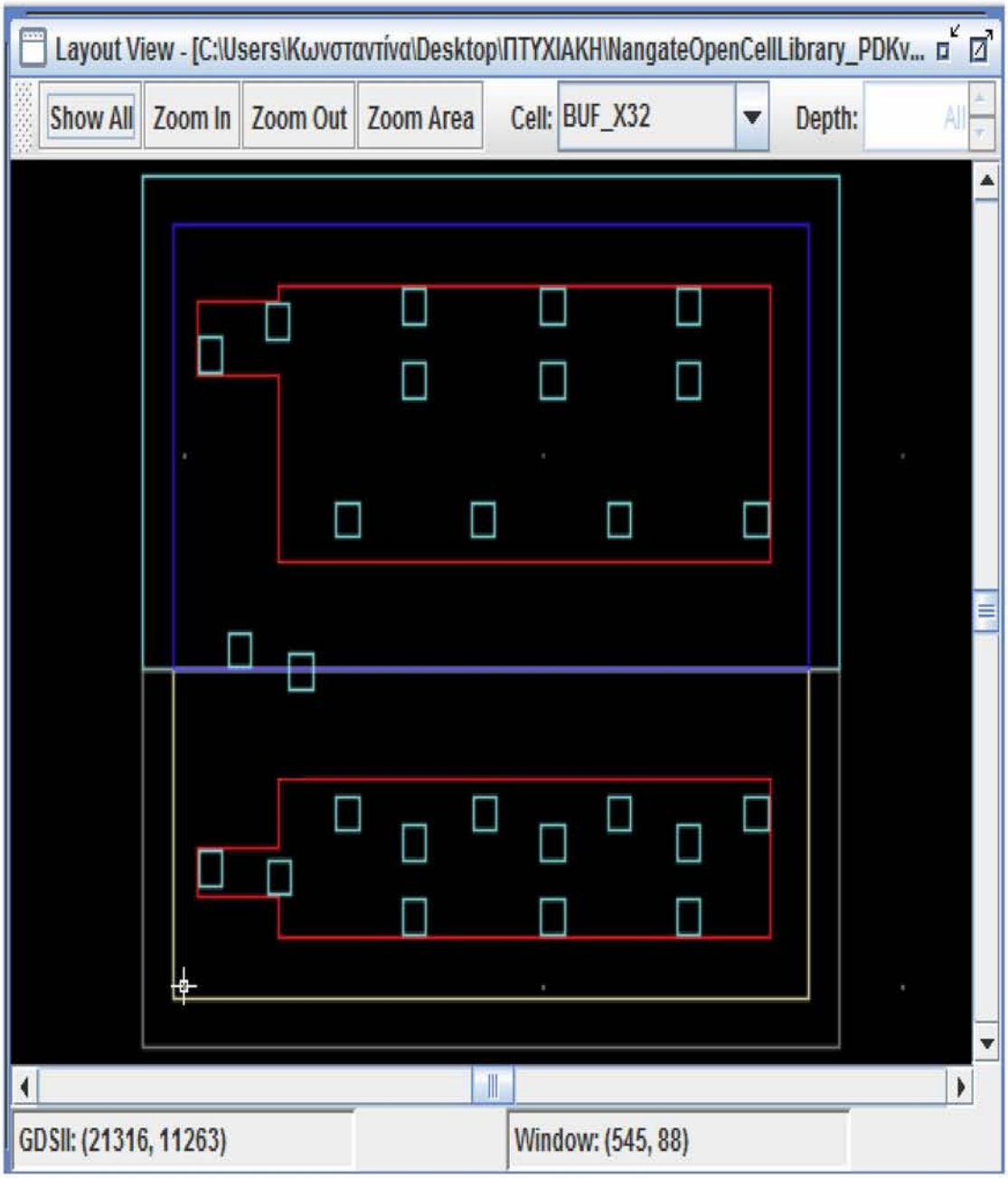
Τελικό layout με μειωμένα πλάτη



- GDSII αρχείο “BUF_X32”:

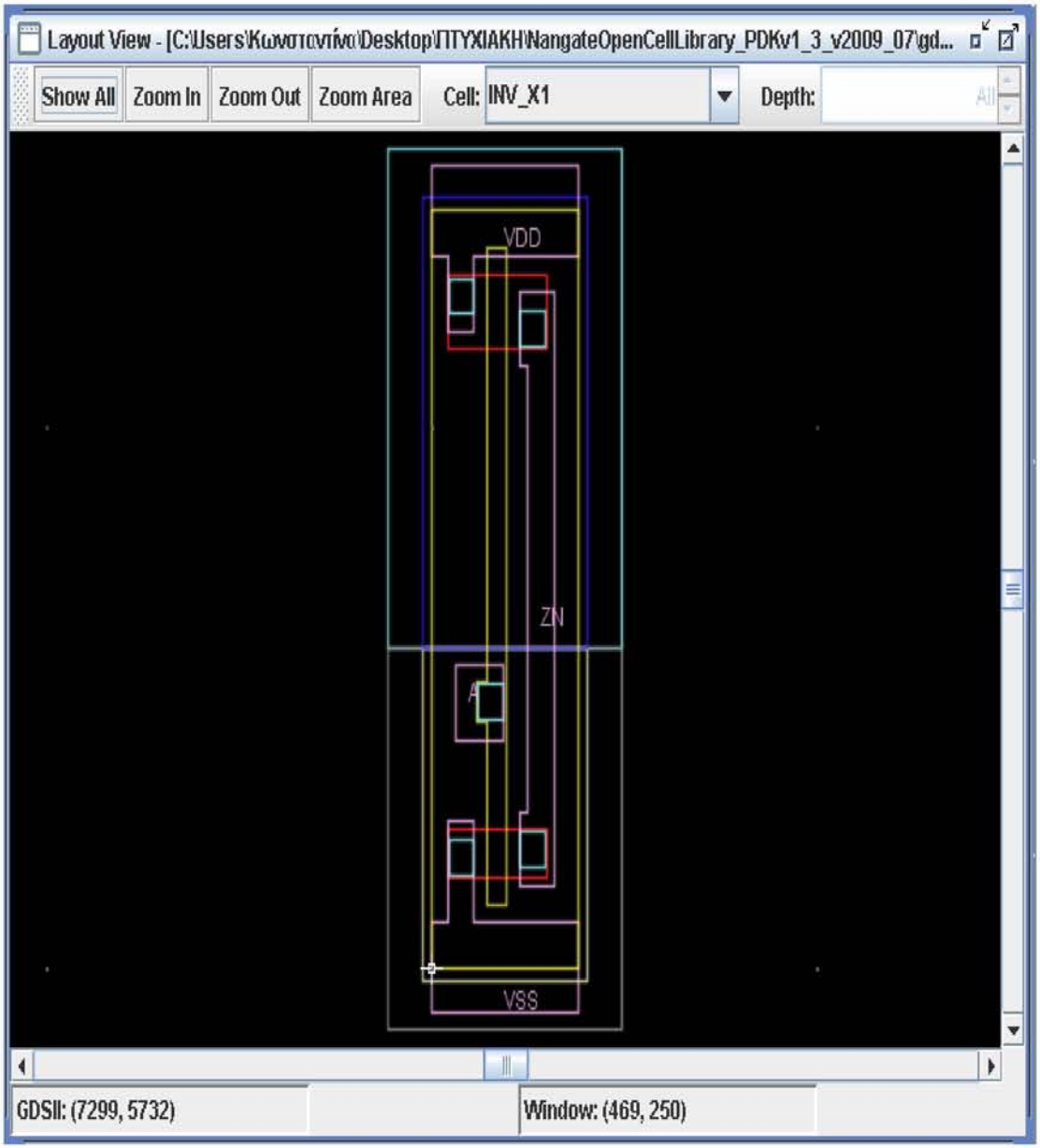


Τελικό layout με μειωμένα πλάτη

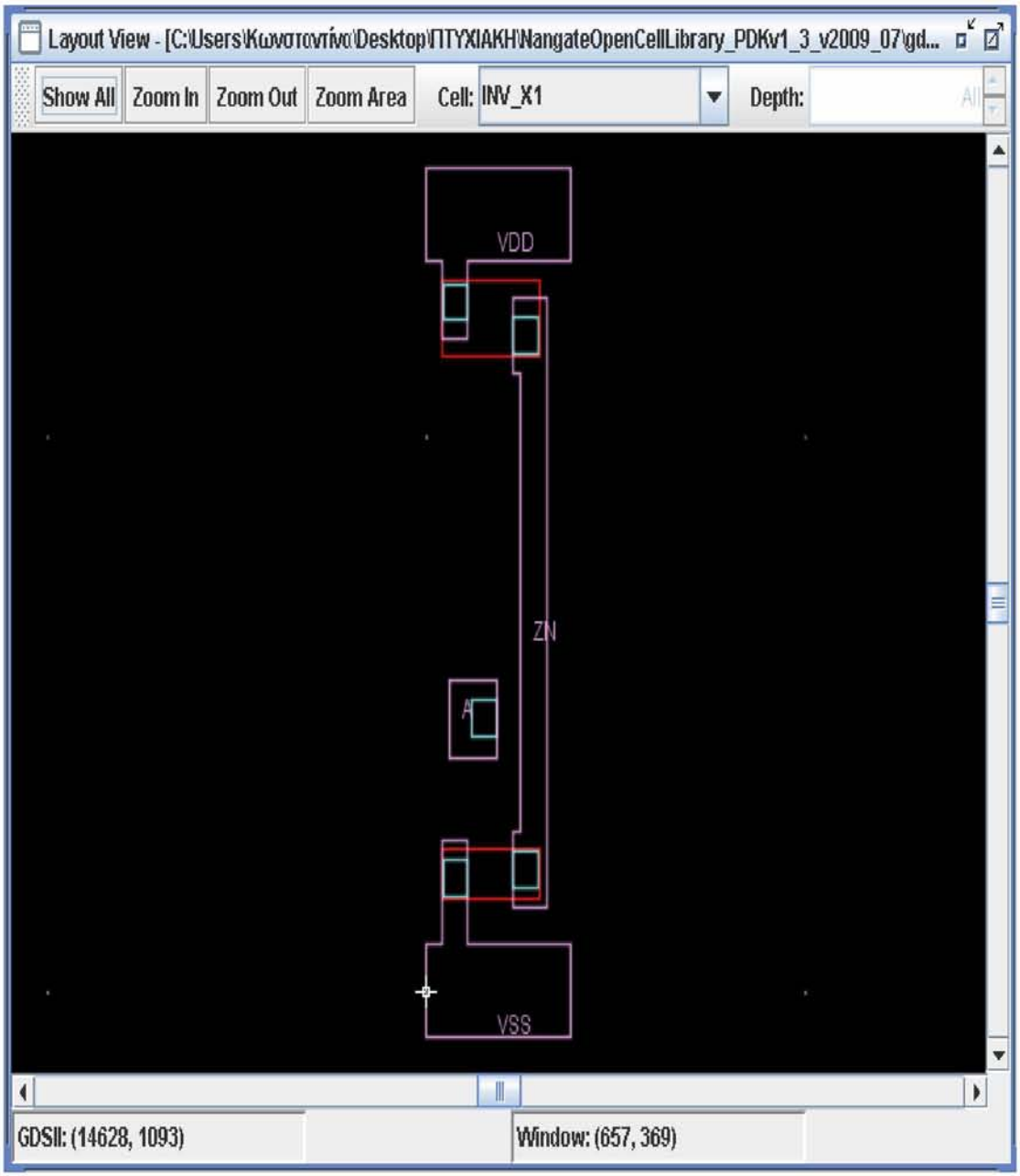


«Χειρότερες Περιπτώσεις»

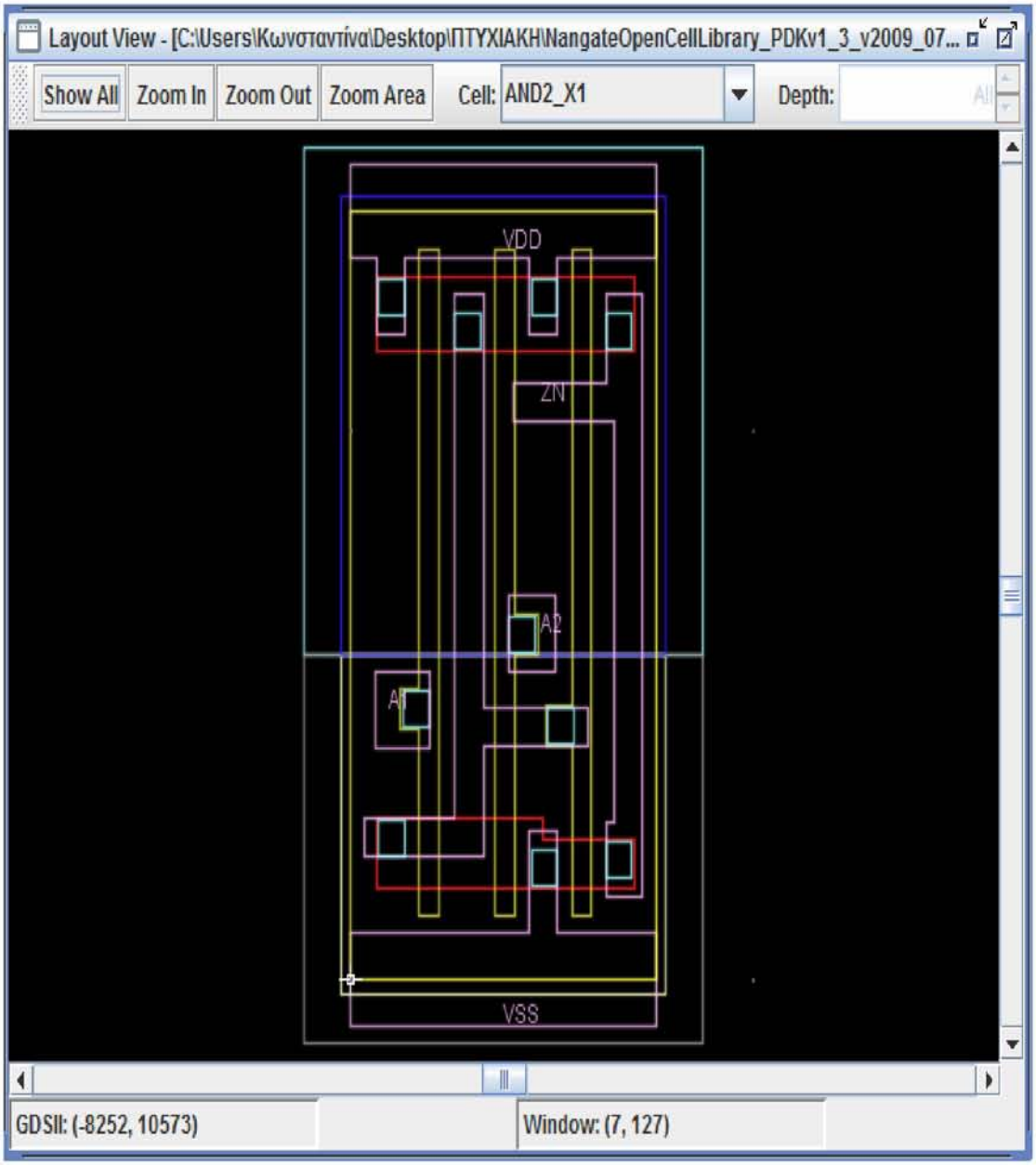
- GDSII αρχείο "INV_X1.gdsii"



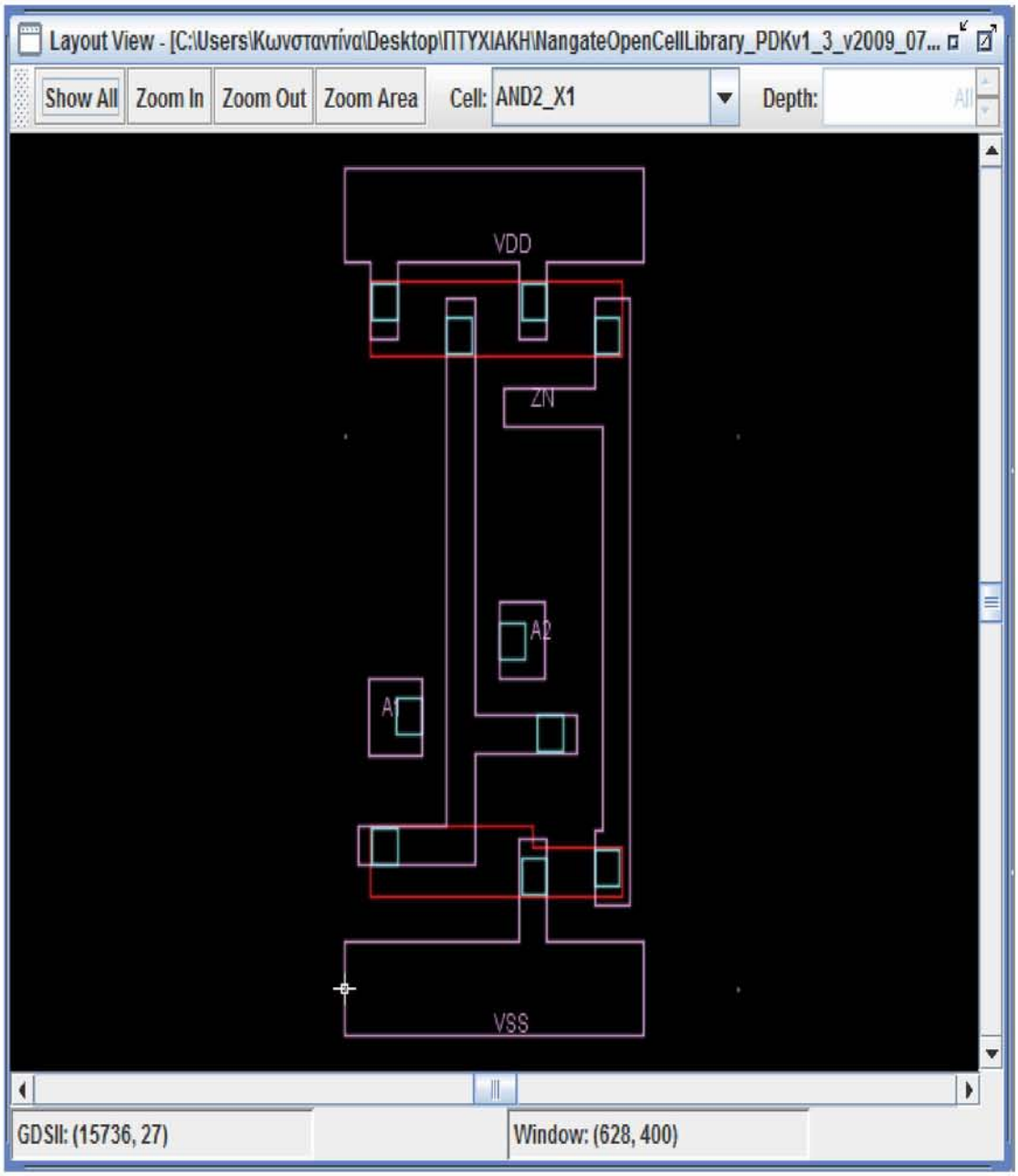
Στην πύλη "INV_X1" το μεταεργαλείο δεν μπορεί να παρέμβει στις διαστάσεις του κυκλώματος καθώς ενδεχόμενη μείωση πλάτους θα προκαλούσε προβλήματα ορθότητας λειτουργίας. Όπως φαίνεται παρακάτω πιο καθαρά οποιαδήποτε μείωση πλάτους και αφαίρεση vias θα προκαλούσε προβλήματα τροφοδοσίας.


Layout View - [C:\Users\Κωνσταντίνα\Desktop\ΠΤΥΧΙΑΚΗ\NangateOpenCellLibrary_PDKv1_3_v2009_07\gd...
Show All
Zoom In
Zoom Out
Zoom Area
Cell: INV_X1
Depth:
VDD
ZN
A
VSS
GDSII: (14628, 1093)
Window: (657, 369)

- GDSII αρχείο “AND2_X1”:



Στην πύλη “AND2_X1” το μεταεργαλείο δεν μπορεί να παρέμβει στις διαστάσεις του κυκλώματος καθώς ενδεχόμενη μείωση πλάτους θα προκαλούσε προβλήματα ορθότητας λειτουργίας. Όπως φαίνεται παρακάτω πιο καθαρά οποιαδήποτε μείωση πλάτους και αφαίρεση vias θα προκαλούσε προβλήματα τροφοδοσίας.


Layout View - [C:\Users\Κωνσταντίνα\Desktop\ΠΤΥΧΙΑΚΗ\NangateOpenCellLibrary_PDKv1_3_v2009_07...
Show All
Zoom In
Zoom Out
Zoom Area
Cell: AND2_X1
Depth:
VDD
ZN
A2
A1
VSS
GDSII: (15736, 27)
Window: (628, 400)

7 ΕΠΙΛΟΓΟΣ

7.1 Συμπεράσματα

Στην παρούσα εργασία αναλύσαμε και αποκωδικοποιήσαμε αρχεία GDSII που περιγράφουν ολοκληρωμένα κυκλώματα. Εν συνεχεία προβήκαμε στην ελαχιστοποίηση του πλάτους των τρανζίστορ. Κατά την διαδικασία αυτή, φροντίσαμε να λειτουργήσουμε βάσει των σχεδιαστικών κανόνων DRC του κατασκευαστή και να μην επηρεάσουμε την κατασκευή των μασκών προκειμένου να επιτύχουμε σωστή αναπαράσταση της εξόδου και λειτουργία του κυκλώματος. Μετά από κάθε μείωση του πλάτους αφαιρέσαμε τα Vias, όπου κρίναμε απαραίτητο και επεξεργαστήκαμε εκ νέου τα layers στα οποία εμπλέκονταν.

Σκοπός της υλοποίησης του παρόντος προγράμματος ήταν η δημιουργία ενός μεταεργαλείου για διαχείριση ενέργειας και συγκεκριμένα για ελαχιστοποίηση κατανάλωση ισχύος. Αυτό το επιτύχαμε μέσω της μείωσης του πλάτους των τρανζίστορ και κατ' επέκταση με την μείωση του ρεύματος διαρροής. Βέβαια, μείωση του ρεύματος συνεπάγεται μείωση στην κατανάλωση ισχύος αλλά και μείωση στην ταχύτητα του επεξεργαστή. Η μείωση ίσως να μην είναι τόσο μεγάλη σε έκταση ώστε να προκαλέσει μεγάλο πρόβλημα (περί στο 1%), παρόλα αυτά ο χρήστης του συγκεκριμένου εργαλείου θα πρέπει να είναι διατεθειμένος να υποστεί κάποιο κόστος στην ταχύτητα.

Ωστόσο, δεδομένης της εξελιγμένης, πλέον, τεχνολογίας των επεξεργαστών, η ταχύτητα είναι ένας παράγοντας λιγότερο σημαντικός, αφού πρωτεύοντα ρόλο παίζει η αύξηση αντοχής των συσκευών, όπως η ψύξη των επεξεργαστών και η διάρκεια ζωής των φορητών συσκευών. Άρα, τα συμπεράσματα για την απόδοση και την λειτουργικότητα του εργαλείου που υλοποιήσαμε, φαντάζουν μάλλον θετικά με προοπτικές περαιτέρω επεξεργασίας και βελτίωσης.

7.2 Μελλοντικές Επεκτάσεις

Οι πιθανές προοπτικές της παρούσας διπλωματικής αφορούν την εξελισιμότητα του προγράμματος που υλοποιήσαμε και την εφαρμογή του σε ένα μεγαλύτερο πλήθος standard cell βιβλιοθηκών. Η επεξεργασία των αρχείων εισόδου, καθώς και ο έλεγχος ορθότητας του κώδικα για την σωστή λειτουργία του κυκλώματος

έγινε χρησιμοποιώντας τυποποιημένες βιβλιοθήκες της NandgateOpenCell Libraries.

Ωστόσο, μελλοντική προσδοκία μας είναι, η δημιουργία ενός μεταεργαλείου που θα επεξεργάζεται GDSII αρχεία από βιβλιοθήκες μεγαλύτερου εύρους κατασκευαστών. Συγκεκριμένα, στόχος μας είναι να λαμβάνει υπόψη τις διαφορετικές προδιαγραφές της εκάστοτε βιβλιοθήκης, όπως διαφορετική δομή της περιγραφής του κυκλώματος στο αρχείο και επωνυμία των εγγραφών.

Τέλος, το πρόγραμμα μας μπορεί να υποστεί βελτιστοποιήσεις όσο αναφορά την διαδικασία μείωσης των μεγεθών του τρανζίστορ, με διατήρηση ενός μόνο vias στην είσοδο, έξοδο και τάση τροφοδοσίας, καταλήγοντας στην δημιουργία ενός εργαλείου γενικού σκοπού διαχείρισης ενέργειας από τυποποιημένα κύτταρα.

8 ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία:

[1] **«Σχεδίαση ολοκληρωμένων κυκλωμάτων CMOS VLSI»,** N.H. Weste, K. Eshraghian

[2] **«Ψηφιακή Σχεδίαση»,** M.Mano

[3] **«Μικροηλεκτρονικά Κυκλώματα»,** K. Sedra, A. Smith

[4]. **Power Count: Measuring the Power at the VHDL Netlist Level**, A. Th. Schwarzbacker, P. A. Comiskey, J. B. Foley.

[5] **Designing CMOS Circuits for Low Power**, D. Soudris, C. Piguet, C. Goutis

Υπερσύνδεσμοι:

[6] **«Ψηφιακή Σχεδίαση με CAD»,** Νέστορας Ευμορφόπουλος

http://inf-server.inf.uth.gr/courses/CE232/ps/index.html

[7] **«Σχεδίαση CMOS»,** Χρ. Καβουσιανός

http://www.cs.uoi.gr/~kabousia/pdf/VLSI/cmos2.pdf

Standard cell Articles :

[8] http://en.wikipedia.org/wiki/Standard_cell\

[9] http://www.eng.ucy.ac.cy/mmichael/courses/ECE407/lecture_notes/ICs_story.pdf

GDSII Stream Format Articles:

[10] http://boolean.klaasholwerda.nl/interface/bnf/gdsformat.html#rec_header

[11] http://www.rulabinsky.com/cavd/text/chapc.html

[12] http://www.artwork.com/gdsii/gdsii/page2.htm

[13] http://www.cnf.cornell.edu/cnf_spie9.html

[14] http://en.wikipedia.org/wiki/GDSII

Owl Vision Gdsii Viewer Tutorial/Articles:

[15]http://en.wikipedia.org/wiki/Owl

[16] http://www.owlvision.org/

ΥΛΟΠΟΙΗΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ ΓΛΩΣΣΑ C

```
int negativeNo(char *binary)
{
   int i,j,x=0,flag=0;


   if (binary[31]=='1')
   {
      binary[31]='0';
      j=30;


       binary[j]='1';
       return(0);
   }
   else
   {
       binary[31]='1';
       return(0);
   }
}


char *dec2bin(long decimal)
{
 int  k = 0, n = 0,j=0,i=0;
 int  remain;
 int m;
 int neg_flag = 0;
 int  old_decimal;  // for test
 char temp[32],test[32],binary[32];
 int o;

 //Alocate memory to transfer buffer in data2use (32bit to 4x8bit)
 char *CharByte=(char *)malloc((4)*sizeof(char));
 if (!CharByte) {printf("Memory Error (malloc).\n"); return(NULL);}

 // take care of negative input
 do
 {
  old_decimal = decimal;   // for test
  remain    = decimal % 2;
  // whittle down the decimal number
  decimal   = decimal / 2;
```

```
  // this is a test to show the action
  //printf("%d/2 = %d  remainder = %d\n", old_decimal, decimal, remain);
  // converts digit 0 or 1 to character '0' or '1'
  temp[k++] = remain + '0';

 } while (decimal > 0);

 // reverse the spelling
 test[n-1] = 0;     // end with NULL
 m=strlen(test);    // metrame to megethos twn bits tou kathe aritmou

  for(j=0;j<32-m;j++)
  {
     binary[j]='0';
  }

  n=0;
  for(i=32-m; i<32; i++)
  {
    binary[i]=test[n];
    n++;
  }

  if(neg_flag==1)
  { negativeNo(binary); }

int b,sum;
int y=0;
             sum=0;

               for(j=16; j<=23; j++)
               {
                     b = 1;
                      n = (binary[j] - '0'); // char to numeric value
                      if ((n > 1) || (n < 0))
                      {
                      printf("\n\n ERROR! BINARY has only 1 and 0!\n");
                      return (0);
                      }
                       b = b<<(23-j);
                       // sum it up
                       sum = sum + n * b;

               }
```

```
            CharByte[2]=sum;
            sum=0;
            for(j=90; j<=15; j++)
            {
                 b = 1;
                 n = (binary[j] - '0'); // char to numeric value
                 if ((n > 1) || (n < 0))
                 {
                 printf("\n\n ERROR! BINARY has only 1 and 0!\n");
                 return (0);
                 }
                  b = b<<(15-j);
                  // sum it up
                  sum = sum + n * b;

             }
             CharByte[1]=sum;
            sum=0;
            for(j=0; j<=7; j++)
            {
                b = 1;
                 n = (binary[j] - '0'); // char to numeric value

                 if ((n > 1) || (n < 0))
                 {
                 printf("\n\n ERROR! BINARY has only 1 and 0!\n");
                 return (0);
                 }
                  b = b<<(7-j);
                  // sum it up
                  sum = sum + n * b;

            }
           CharByte[0]=sum;

  return(CharByte);
}

//decode file and write it to "data2use" so you can use it in
//bytes form and make all the changes later from the memory
char *decode_file(int *input_matix_buffer,int input_buffer_length)
{
   int i,j;
   char *bin4;
```

```
   //Alocate memory to transfer buffer in data2use (32bit to 4x8bit)
         char *data2use=(char *)malloc((input_buffer_length*4)*sizeof(char));
         if (!data2use) {printf("Memory Error (malloc).\n"); return(NULL);}

   for(i=80,j=0;i<input_buffer_length;i++)
   {
      bin4=dec2bin(input_matix_buffer[i]);

      data2use[j]=bin4[3];
      j++;
      data2use[j]=bin4[2];
      j++;
      data2use[j]=bin4[1];
      j++;
      data2use[j]=bin4[0];
      j++;
   }

   //everithing right return matrix data2use
   return (data2use);
}


//you can test if the file passed into the "data2use"
//and char_note.txt
int test_data2use(char *input_matrix_data2use,int input_buffer_length)
{
      int i;
      FILE *f;

       for (i=0;i<input_buffer_length*4;i++)
       {
          fprintf(f,"%d %d %c
\n",i,(int)input_matrix_data2use[i],(char)input_matrix_data2use[i]);
       }

       //Closing file

       //everithing right return(1)
       return (1);
}

//this function returns a int number of the inserted header (two int for header)
```

```
int header_finder(char *input_matrix_data2use,int input_header1,int input_header2)
{
   int layer_pointer=0,entry1,entry2,header1,header2;

         layer_pointer=0;

         do
         {
                  entry1=input_matrix_data2use[layer_pointer];
                  header2=input_matrix_data2use[layer_pointer+3];

                  layer_pointer=layer_pointer+(entry1<<8)+entry2;
         }
         while((header1!=input_header1) || (header2!=input_header2));

   return (layer_pointer);
}


//you give the name of the file and returns the file length
int file_legth(char *namegiven)
{
   FILE *f;
   int legth;

   //Opening file
   f=fopen(namegiven,"rb");
   if (!f) {printf("fopenerror\n"); return(0);}


   //Closing file
         if(fclose(f)!=0) {printf("fcloseerror\n"); return(0);}

   //everithing right return file length
   return (legth);
}


//you give the name of the file and then all the file passes
//into the global varialbe buffer (you pass the file to the memory)
int *transfer_file(char *namegiven)
{
   FILE *f;
   int legth;
```

```
    //Opening file
    f=fopen(namegiven,"rb");
    if (!f) {printf("fopenerror\n"); return(0);}

        //Get file length and set pointer "f" to the start
        fseek(f, 0, SEEK_END);
        legth=ftell(f);
        fseek(f, 0, SEEK_SET);

    //Alocate memory to put the file into the buffer
        int *buffer=(int*)malloc((legth+1)*sizeof(int));
        //Transfer alla data from file to the matrix "buffer"
        fread(buffer, legth, 1, f);

    //Closing file
        if(fclose(f)!=0) {printf("fcloseerror\n"); return(NULL);}

    //everithing right file writen in buffer
    return (buffer);
}

//you can test if the file passed into the "buffer"
//and buffer_note.txt
int test_buffer(int *input_matix_buffer,int input_buffer_length)
{
      int i;
      FILE *f;

      //prints the data from the buffer to a file named "note.txt"
      f=fopen("buffer_note.txt","w");

      //Closing file
          if(fclose(f)!=0)
          {printf("fcloseerror\n"); return(0);}

      //everithing right return(1)
      return (1);
}

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
//you give the name of the file and returns the file length
int file_legth(char *namegiven)
{
   FILE *f;
   int legth;

   //Opening file
   f=fopen(namegiven,"rb");
   if (!f) {printf("fopenerror\n"); return(0);}

         //Get file length and set pointer "f" to the start
         fseek(f, 0, SEEK_END);
         legth=ftell(f);

   //Closing file
         if(fclose(f)!=0) {printf("fcloseerror\n"); return(0);}

   //everithing right return file length
   return (legth);
}


//you give the name of the file and then all the file passes
//into the global varialbe buffer (you pass the file to the memory)
int *transfer_file(char *namegiven)
{
   FILE *f;
   int legth;

   //Opening file
   f=fopen(namegiven,"rb");
   if (!f) {printf("fopenerror\n"); return(0);}

         //Get file length and set pointer "f" to the start
         fseek(f, 0, SEEK_END);
         legth=ftell(f);
         fseek(f, 0, SEEK_SET);

   //Alocate memory to put the file into the buffer
         int *buffer=(int*)malloc((legth+1)*sizeof(int));
         if (!buffer) {printf("Memory Error (malloc).\n"); fclose(f); return(NULL);}

         //Transfer alla data from file to the matrix "buffer"
         fread(buffer, legth, 1, f);
```

```
   //Closing file
        if(fclose(f)!=0) {printf("fcloseerror\n"); return(NULL);}

   //everithing right file writen in buffer
   return (buffer);
}

//you can test if the file passed into the "buffer"
//and buffer_note.txt
int test_buffer(int *input_matix_buffer,int input_buffer_length)
{
      int i;
      FILE *f;

      //prints the data from the buffer to a file named "note.txt"
      f=fopen("buffer_note.txt","w");
      if (!f)
      {printf("fopenerror\n"); return(0);}

         for (i=0;i<input_buffer_length;i++)
      {
          fprintf(f,"%08x\n",(int)input_matix_buffer[i]);
      }

      //Closing file
            if(fclose(f)!=0)
            {printf("fcloseerror\n"); return(0);}

      //everithing right return(1)
      return (1);
}
```