ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

΄΄ Υλοποίηση ενός προσομοιωτή ψηφιακών κυκλωμάτων για την μελέτη της κατανάλωσης ισχύος΄΄

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μπουντάς Δημήτριος

Χανιά
Οκτώβριος 2004

Σε αυτή την σελίδα θα ήθελα να ευχαριστήσω τον κ. Σταμούλη για την υποστήριξη και καθοδήγηση του στην προσπάθεια μου για την υλοποίηση αυτής της διπλωματικής.

Επίσης ευχαριστώ τους κ Πνευματικάτο και κ. Δόλλα για την ανάγνωση του κειμένου και την όλη τους συμμετοχή στην διαδικασία παρουσίασης της διπλωματικής.

Ευχαριστώ τους παλιούς μου συναδέλφους κ. Δ. Καραμπατζάκη για το χρήσιμο υλικό που μου έδωσε και την υποστήριξη του καθώς και κ. Δ. Συρίβελη και κ. Σ. Κοψιδά για την βοηθεια και καθοδήγηση τους με τα μηχανήματα Sun.

Περιεχόμενα

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΑΠΟΤΕΛΕΣΜΑΤΑ ΠΡΟΣΟΜΟΙΩΣΕΩΝ

Στατική Ισχύς

Δυναμική Ισχύς

Single Event Upsets

Ορολογία στοιχείων αλγορίθμων

NSUBC_IN(n):Το σύνολο των υποκυκλωμάτων των οποίων ο κόμβος n είναι είσοδος
NSUBC_OUT(n): Το υποκύκλωμα του οποίου ο κόμβος n είναι έξοδος
MSUBC(m):Το υποκύκλωμα του οποίου το τρανζίστορ m είναι μέρος
TYPE(m):Ο τύπος του τρανζίστορ m (p-type , n-type)
GATE(m):Ο κόμβος gate του τρανζίστορ m
SOURCE(m):Ο κόμβος source του τρανζίστορ m
DRAIN(m): Ο κόμβος drain του τρανζίστορ m
VALUE(n):Η τιμή του κόμβου n (0 , 1 , X= απροσδιόριστη , A = βραχυκύκλωμα)
MPASS(n):Μεταβλητή που ‘μαρκάρει’ το πέρασμα από τον κόμβο n
TPASS(t): Μεταβλητή που ‘μαρκάρει’ το πέρασμα από το τρανζίστορ t
OUTPUT(s): Η έξοδος του κυκλώματος s
n:κόμβος
t:τρανζίστορ

Κεφάλαιο 1ο

ΕΙΣΑΓΩΓΗ

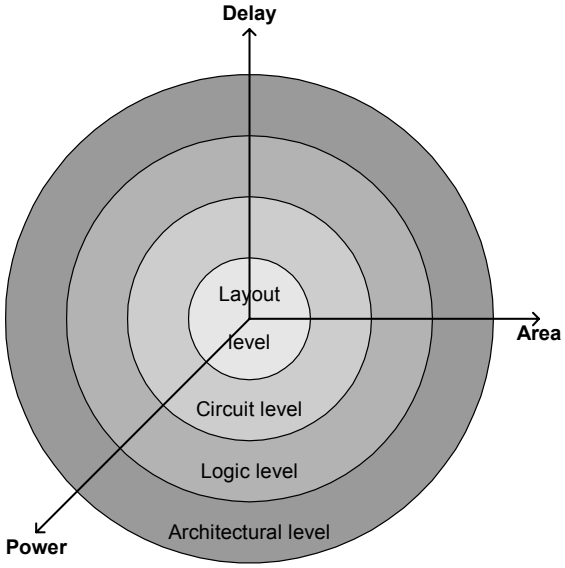
Οι περισσότερες ερευνητικές και βιομηχανικές προσπάθειες στον τομέα των ψηφιακών ηλεκτρονικών είχαν συγκεντρωθεί στο να αυξηθεί η ταχύτητα και η πολυπλοκότητα των ολοκληρωμένων κυκλωμάτων. Η προσπάθεια αυτή οδήγησε σε μια πανίσχυρη αλλά δαπανηρή ενεργειακά σχεδιαστική τεχνολογία η οποία άνοιξε τον δρόμο για την κατασκευή και ανάπτυξη των προσωπικών υπολογιστών, των υπολογιστών με δυνατότητες απεικόνισης σύνθετων γραφικών και φυσικά υπολογιστικών συστημάτων πολυμέσων όπως η αναγνώριση φωνής σε πραγματικό χρόνο και το video σε πραγματικό χρόνο. Η προσοχή επικεντρώνονταν στην ταχύτητα και το μέγεθος του ολοκληρωμένου κυκλώματος, και η κατανάλωση ισχύος παραμελούνταν.

Η κατάσταση αυτή όμως τα τελευταία χρόνια αλλάζει. Η κατανάλωση ισχύος ενός ολοκληρωμένου κυκλώματος αγγίζει τα όρια των δυνατοτήτων που προσφέρουν οι τεχνολογίες. Το αποτέλεσμα είναι να μειώνεται η αξιοπιστία της συσκευής , και να περιορίζεται η ταχύτητα λειτουργίας και πιθανών και οι εφαρμογές του ολοκληρωμένου. Η αντιμετώπιση των προβλημάτων κατανάλωσης ισχύος γρήγορα και αποτελεσματικά γίνεται ένα από τα πιο απαιτητικά θέματα στον σχεδιασμό ψηφιακών ηλεκτρονικών συστημάτων. Η αυξανόμενη ζήτηση φορητών συσκευών στις τεχνολογικές περιοχές των τηλεπικοινωνιακών , υπολογιστικών και εμπορικών ηλεκτρονικών εκτείνει το πρόβλημα παροχής ισχύος και απαγωγής θερμότητας λόγω των αυστηρών και χαμηλών ορίων που τίθενται σε αυτές. Οι βελτιώσεις στις τεχνολογίες των μπαταριών εξισορροπούνται συνήθως από την πολυπλοκότητα και τις υψηλές απαιτήσεις σε απόδοση των σύγχρονων εφαρμογών.

Η περιοχή του υπολογισμού και μείωσης της κατανάλωσης ισχύος αποδεικνύεται σε ένα δυναμικό πεδίο τόσο για έρευνα όσο και για πρακτικές εφαρμογές με σαφή δυνατότητες ραγδαίας περαιτέρω ανάπτυξης.

1.1 Η ισχύς ως παράμετρος της σχεδίασης ολοκληρωμένων κυκλωμάτων

Είναι γνωστό ότι η σχεδίαση ενός ψηφιακού ολοκληρωμένου κυκλώματος λαμβάνει χώρα σε διάφορα ιεραρχικά *επίπεδα αφαίρεσης* (*abstraction levels*, τα κυριότερα από τα οποία είναι το επίπεδο *αρχιτεκτονικής* ή *καταχωρητή* (*architectural* or *register-transfer*), το επίπεδο *λογικής* ή *πύλης* (*logical* or *gate*), το επίπεδο *κυκλώματος* ή *τρανζίστορ* (*circuit* or *transistor*) και το επίπεδο *φυσικού σχεδίου* (*layout*). Για ένα κύκλωμα δεδομένης λειτουργικής συμπεριφοράς, η διαδικασία σχεδίασης στα 4 προηγούμενα επίπεδα έχει ως αντικείμενο τη βέλτιστη λύση ως προς τις τρεις κύριες παραμέτρους της καθυστέρησης, της επιφάνειας που καταλαμβάνει το κύκλωμα και της κατανάλωσης ισχύος, κάτω από ορισμένους περιορισμούς για κάθε παράμετρο οι οποίοι δίνονται υπό τη μορφή προδιαγραφών. Η πρώτη από τις πιο πάνω παραμέτρους (καθυστέρηση) έχει σχέση με την απόδοση του κυκλώματος, καθώς μέσω αυτής καθορίζεται η συχνότητα ρολογιού και επομένως η ταχύτητα λειτουργίας, ενώ οι άλλες δύο (επιφάνεια και ισχύς) αντιστοιχούν στην απαίτηση για εκμετάλλευση όσο το δυνατόν λιγότερων φυσικών πόρων. Υπό το πρίσμα αυτό, το πρόβλημα της σχεδίασης ψηφιακών κυκλωμάτων VLSI μπορεί να θεωρηθεί κατά μία (μακρινή) έννοια ως ένα πρόβλημα *βελτιστοποίησης υπό περιορισμούς* (*constrained optimization*) σε ένα χώρο τριών διαστάσεων, όπως φαίνεται στο ακόλουθο σχήμα 1.1.



Σχήμα .1.1.
Τρισδιάστατος χώρος σχεδίασης και επίπεδα αφαίρεσης της διαδικασίας σχεδίασης ψηφιακών κυκλωμάτων CMOS VLSI.

Προκειμένου να μπορεί να πραγματοποιηθεί η βελτιστοποίηση της σχεδίασης στον παραπάνω τρισδιάστατο χώρο και να ελεγχθεί η συμμόρφωση με τις προδιαγραφές που έχουν τεθεί, απαιτούνται ακριβείς εκτιμήσεις των τριών παραμέτρων που αναφέρθηκαν σε καθένα από τα επίπεδα αφαίρεσης, οι οποίες να είναι διαθέσιμες κατά τη φάση της σχεδίασης και πριν το κύκλωμα φτάσει στο κατασκευαστικό στάδιο. Για τις παραμέτρους της καθυστέρησης και της επιφάνειας μπορούν να εξαχθούν πολύ ακριβείς εκτιμήσεις, και συγκεκριμένα για την πρώτη από το άθροισμα των καθυστερήσεων διάδοσης των μεμονωμένων πυλών που ανήκουν στο *κρίσιμο μονοπάτι* (*critical path*) του κυκλώματος το οποίο καταλαμβάνει το μέγιστο αριθμό λογικών επιπέδων, ενώ για τη δεύτερη από την εκάστοτε *χωροθέτηση* (*floorplanning*) του κυκλώματος και την επιφάνεια κάθε μεμονωμένου στοιχείου. Αντίθετα η παράμετρος της ισχύος είναι εξαιρετικά δύσκολη στην εκτίμησή της καθώς εξαρτάται από τα συγκεκριμένα διανύσματα εισόδου του κυκλώματος που προξενούν κάποια αλλαγή στη λογική του κατάσταση. Επιπλέον, ένα άλλο πρόβλημα είναι ότι ενώ οι παράμετροι της καθυστέρησης και της επιφάνειας είναι μονοσήμαντα ορισμένες, η κατανάλωση ισχύος έχει τουλάχιστον δύο διαφορετικές εκφάνσεις που ενδιαφέρουν κατά τη σχεδίαση ολοκληρωμένων κυκλωμάτων και οι οποίες είναι η μέση και η μέγιστη ισχύς. Στο παρελθόν η παράμετρος της ισχύος κατείχε μικρότερη σημασία σε σχέση με τις άλλες δύο παραμέτρους, κυρίως λόγω του ανταγωνισμού που επικρατούσε για την επίτευξη ολοένα και υψηλότερων ταχυτήτων με ταυτόχρονη αύξηση της πυκνότητας ολοκλήρωσης, αλλά και σε ένα βαθμό λόγω της προαναφερθείσας δυσκολίας που παρουσίαζε ο χαρακτηρισμός της. Τα τελευταία, όμως, χρόνια όμως το σκηνικό έχει αλλάξει δραματικά και η παράμετρος της ισχύος θεωρείται πλέον ως ίσης τουλάχιστον σημασίας για τη σχεδίαση ολοκληρωμένων κυκλωμάτων, γεγονός το οποίο καθιστά την ανάγκη ανάλυσης και εκτίμησής της εντονότερη από ποτέ.

1.2 Η κατανάλωση ισχύος στα κυκλώματα CMOS VLSI

Βασικές έννοιες

Για την αποτελεσματική αντιμετώπιση του προβλήματος ανάλυσης και εκτίμησης της ισχύος θα πρέπει πρώτα από όλα να εξετάσουμε και να κατανοήσουμε την προέλευση της κατανάλωσης ισχύος στα ψηφιακά κυκλώματα CMOS VLSI. Υποθέτοντας αρχικά ότι η πηγή τροφοδοσίας του κυκλώματος είναι ιδανική πηγή τάσης, δηλαδή μπορεί να δώσει ανά πάσα στιγμή όλο το ρεύμα που χρειάζεται το κύκλωμα για τη λειτουργία του διατηρώντας σταθερή την τάση τροφοδοσίας V_{DD}, η *στιγμιαία* (*instantaneous*) ισχύς του κυκλώματος για κάθε χρονική στιγμή t θα δίνεται από το γινόμενο της τάσης V_{DD} με το συνολικό ρεύμα $I(t)$ που εισέρχεται στους ακροδέκτες τροφοδοσίας:

$$P(t) = V_{DD} \cdot I(t) \qquad (1.1)$$

Με αυτό τον τρόπο προκύπτει ότι η κατανάλωση ισχύος του κυκλώματος θα είναι ανάλογη του ρεύματος εισόδου, πράγμα που σημαίνει ότι οι δύο αυτές ποσότητες θα μπορούν στο εξής να χρησιμοποιούνται εναλλακτικά η μια της άλλης. Το κύριο μέρος της ανάλυσης και εκτίμησης ισχύος διεξάγεται συνήθως στα δύο ενδιάμεσα

αφαιρετικά επίπεδα της λογικής και του κυκλώματος, καθώς στο ανώτερο επίπεδο της αρχιτεκτονικής η ανάλυση δεν είναι δυνατό να έχει την απαιτούμενη ακρίβεια και χρησιμοποιείται μόνο για μια αρχική αποτίμηση της κατάστασης, ενώ στο κατώτερο φυσικό επίπεδο οι περισσότερες σχεδιαστικές παράμετροι έχουν ήδη παγιωθεί και η εκτίμηση θα πρέπει να υπάρχει από τα προηγούμενα επίπεδα πριν η σχεδίαση καταλήξει εδώ. Επικεντρώνοντας, λοιπόν, την προσοχή μας στα δύο ενδιάμεσα αυτά επίπεδα, και καθώς εκεί η φυσική διάταξη των αγωγών τροφοδοσίας μέσα στο ολοκληρωμένο κύκλωμα δεν έχει ακόμα σχηματιστεί, μπορούμε να θεωρήσουμε ότι όλες οι πύλες του κυκλώματος τροφοδοτούνται από την ίδια σταθερή τάση η οποία είναι ίση με V_{DD} (στην πραγματικότητα υπάρχει μια πτώση τάσης πάνω στους αγωγούς τροφοδοσίας που σχηματίζονται στο φυσικό επίπεδο). Καθώς επιπλέον οι πύλες αυτές συνδέονται με παράλληλο τρόπο πάνω στους αγωγούς τροφοδοσίας και γείωσης, το συνολικό ρεύμα του κυκλώματος θα λαμβάνεται από την επαλληλία των ρευμάτων κάθε μεμονωμένης πύλης ξεχωριστά και επομένως η στιγμιαία ισχύς του (εάν το κύκλωμα αποτελείται από q πύλες) θα είναι:

$$(1.2) \quad P(t) = V_{DD} \sum_{i=1}^{q} I_i(t)$$

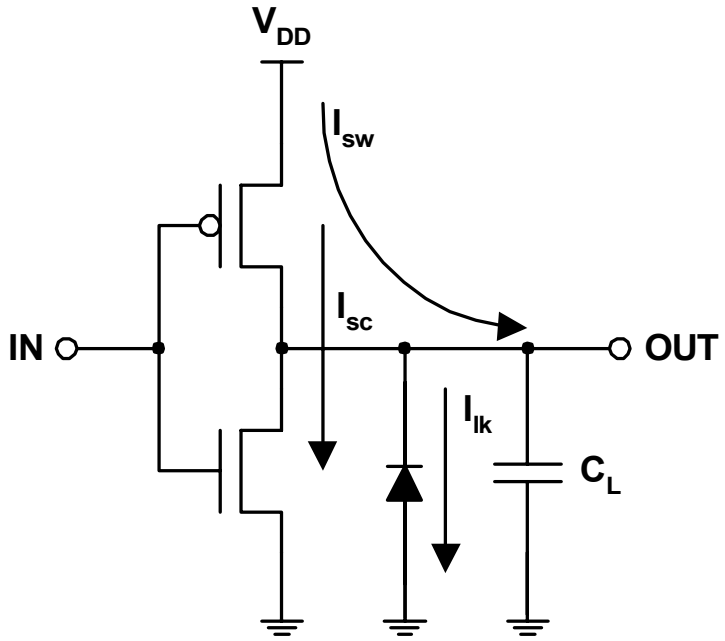
Εκτός από τη στιγμιαία κατανάλωση ισχύος, ένας άλλος σημαντικός τύπος ισχύος είναι η *μέση* (*average*) ισχύς που καταναλώνεται σε ένα χρονικό διάστημα T, η οποία θα λαμβάνεται από το χρονικό μέσο της στιγμιαίας ισχύος για το διάστημα αυτό:

$$(1.3) \quad P_T = \frac{1}{T}\int_0^T P(t)dt = \frac{V_{DD}}{T}\int_0^T I(t)dt$$

Εύκολα μπορεί κανείς να διαπιστώσει ότι και η μέση ισχύς του κυκλώματος δύναται να γραφεί ως η επαλληλία των επιμέρους τιμών της για κάθε μεμονωμένη πύλη ξεχωριστά, ως εξής:

$$(1.4) \quad P_T = V_{DD} \sum_{i=1}^{q} \frac{1}{T}\int_0^T I_i(t)dt$$

Η κατανάλωση ισχύος σε ένα κύκλωμα CMOS γενικά μπορεί να αναλυθεί σε άθροισμα τριών συνιστωσών οι οποίες είναι η ισχύς *μεταγωγής* (*switching*) P_{sw}, η ισχύς *βραχυκυκλώματος* (*short-circuit*) P_{sc} και η ισχύς *διαρροής* (*leakage*) P_{lk}. Οι δύο πρώτες είναι οι δυναμικές συνιστώσες της συνολικής ισχύος καθώς, όπως θα δούμε στη συνέχεια, εμφανίζονται μόνο κατά τη *μετάβαση* (*transition*) μεταξύ δύο λογικών καταστάσεων, ενώ η τρίτη είναι η στατική συνιστώσα η οποία αντιπροσωπεύει μια μόνιμη πηγή κατανάλωσης. Οι συνιστώσες αυτές παριστάνονται γραφικά στο σχήμα 1.2 για μια γενική πύλη CMOS και εξετάζονται αναλυτικότερα στις ενότητες που ακολουθούν.



Σχήμα 1.2
Συνιστώσες κατανάλωσης ισχύος σε μια πύλη CMOS.

1.3 Οι ορισμοί της μέσης και της μέγιστης ισχύος στα κυκλώματα CMOS VLSI

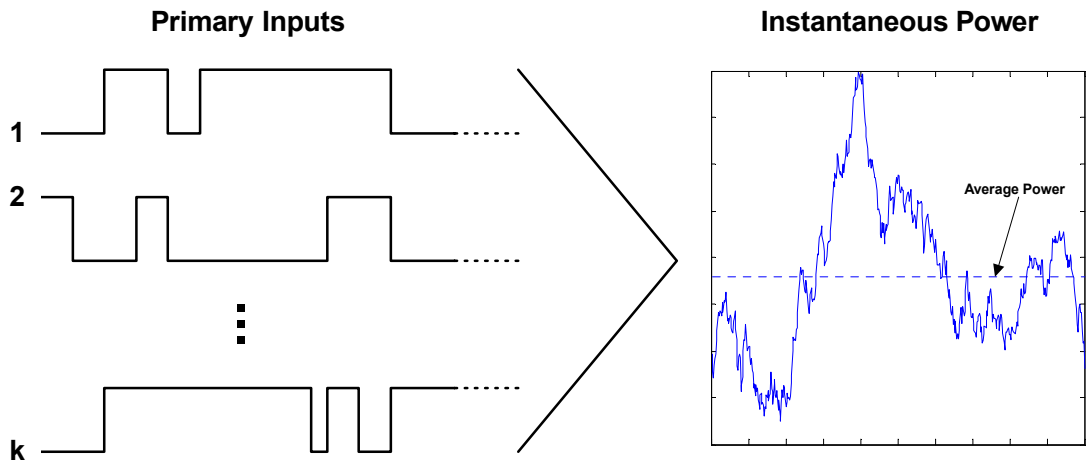
Στην αρχή της προηγούμενης ενότητας κάναμε λόγο για τις δύο κυριότερες μορφές ισχύος που υπάρχουν γενικά, δηλαδή τη στιγμιαία και τη μέση ισχύ, οι οποίες ορίστηκαν στις σχέσεις (1.1) και (1.3) ως συναρτήσεις των χρονικών μεταβλητών *t* (χρονική στιγμή) και *T* (χρονικό διάστημα) αντίστοιχα. Στη συνέχεια επίσης κατέστη εμφανές ότι ειδικά για την περίπτωση των κυκλωμάτων CMOS, λόγω της ύπαρξης των δυναμικών συνιστωσών μεταγωγής (κατά κύριο λόγο) και βραχυκυκλώματος οι οποίες εμφανίζονται αποκλειστικά κατά την αλλαγή λογικών καταστάσεων, ο προσδιορισμός των παραπάνω μορφών ισχύος από τις αντίστοιχες σχέσεις τους έχει νόημα μόνο εάν είναι γνωστά τα συγκεκριμένα διανύσματα εισόδου του κυκλώματος που εφαρμόζονται (πάντα σε συγχρονισμό με το ρολόι) τη στιγμή ή το διάστημα όπου αυτές εξετάζονται. Με βάση, όμως, τις προαναφερθείσες σχέσεις μπορούν για κάθε κύκλωμα να οριστούν δύο μοναδικές ανεξάρτητες του χρόνου τιμές ισχύος, ήτοι η μέγιστη ισχύς P_{mx} και η μέση (ανεξαρτήτως διαστήματος) ισχύς P_{av}, οι οποίες

καλύπτουν το σύνολο των διανυσμάτων εισόδου και έχουν ξεχωριστή σημασία γενικότερα για τη σχεδίαση ολοκληρωμένων κυκλωμάτων.

Διαισθητικά, ως μέση ισχύς ενός κυκλώματος CMOS νοείται η μέση ισχύς της σχέσης (1.3) για ένα εκτεταμένο χρονικό διάστημα T το οποίο περιλαμβάνει μεγάλο πλήθος κύκλων ρολογιού και εξαντλεί όλες τις δυνατές μεταβολές μεταξύ των διανυσμάτων εισόδου, όπως απεικονίζεται στο σχήμα 1.3. Σε αυστηρούς μαθηματικούς όρους, η μέση ισχύς του κυκλώματος ορίζεται ως το ακόλουθο όριο της σχέσης (1.3) για $T \to +\infty$:

(1.5) $$P_{av} = \lim_{T \to +\infty} P_T = \lim_{T \to +\infty} \frac{1}{T} \int_0^T P(t)dt$$

το οποίο σύμφωνα με ορισμένα αποτελέσματα υπάρχει πάντοτε και συγκλίνει σε μια συγκεκριμένη τιμή P_{av} για κάθε κύκλωμα.



Σχήμα 1.3.
Γραφική απεικόνιση του ορισμού της μέσης ισχύος στα κυκλώματα CMOS VLSI.

Από την άλλη, η μέγιστη ισχύς ενός κυκλώματος CMOS (η οποία είναι συνήθως αρκετά μεγαλύτερη της αντίστοιχης μέσης) θα είναι όπως αναμένεται η μέγιστη ή αλλιώς η *χειρότερη δυνατή* (*worst case*) τιμή της στιγμιαίας ισχύος (1.1) που καταναλώνεται από το κύκλωμα για κάθε χρονική στιγμή t:

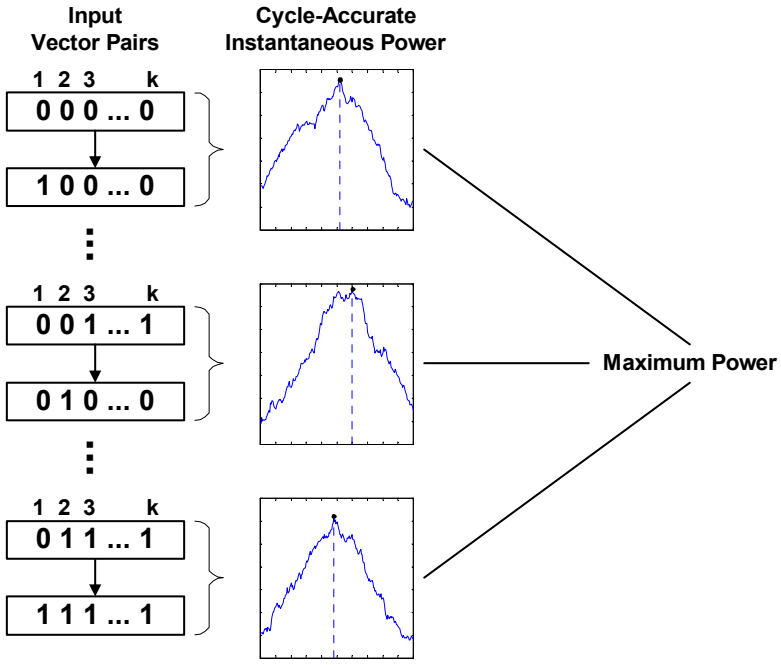
(1.6) $$P_{mx} = \max_{t \in \Re} P(t)$$

Με βάση την ανάλυση της ισχύος των κυκλωμάτων CMOS σε δύο δυναμικές και μια στατική συνιστώσα που έγινε στην προηγούμενη ενότητα, και λαμβάνοντας επίσης υπόψη ότι η τελευταία είναι αρκετά μικρότερη από τις άλλες δύο, προκύπτει ότι οι δυσμενέστερες συνθήκες ως προς την κατανάλωση ισχύος θα υφίστανται στο διάστημα που επακολουθεί μιας λογικής μετάβασης και το οποίο εκτείνεται για μια

περίοδο τ του ρολογιού μέχρι την επόμενη μετάβαση, πράγμα που σημαίνει ότι η εξέταση της στιγμιαίας ισχύος δύναται να περιοριστεί εκεί. Γενικότερα όταν η στιγμιαία ισχύς εξετάζεται στο διάστημα μιας περιόδου ή ενός κύκλου του ρολογιού, τότε θα αναφέρεται ως (ακριβής) στιγμιαία ισχύς *κύκλου* (*cycle-accurate*). Έτσι τελικά η μέγιστη ισχύς του κυκλώματος θα είναι ίση με τη μέγιστη στιγμιαία ισχύ που μπορεί να καταναλωθεί μέσα σε έναν κύκλο ρολογιού, οπότε θα ορίζεται αυστηρά ως η μέγιστη τιμή των κορυφών (μεγίστων) της στιγμιαίας ισχύος κύκλου ανάμεσα σε όλα τα δυνατά ζεύγη $(\underline{v}_1, \underline{v}_2)$ των διανυσμάτων εισόδου που σηματοδοτούν μια λογική μετάβαση στην απαρχή ενός νέου κύκλου:

$$P_{mx} = \max_{(\underline{v}_1, \underline{v}_2)\in\Omega} \left(\max_{t\in[0,\tau]} P_{(\underline{v}_1, \underline{v}_2)}(t) \right) \quad (1.7)$$

όπου $\underline{v}_1$ και $\underline{v}_2$ είναι το "προηγούμενο" και το "επόμενο" διάνυσμα της μετάβασης αντίστοιχα, ενώ Ω είναι το πλήρες σύνολο των ζευγών $(\underline{v}_1, \underline{v}_2)$. Ο ορισμός αυτός της μέγιστης ισχύος απεικονίζεται γραφικά στο ακόλουθο σχήμα 1.4.



Σχήμα 1.4
Γραφική απεικόνιση του ορισμού της μέγιστης ισχύος στα κυκλώματα CMOS VLSI.

Κεφάλαιο 2
ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Το πρόγραμμα μας είναι ένας προσομοιωτής ψηφιακών κυκλωμάτων VLSI σε switch level και gate level. Ο προσομοιωτης δέχεται σαν είσοδο net-list αρχεία τα οποία περιγράφουν τα κυκλώματα και περιλαμβάνει στρατηγικές προσομοίωσης οι οποίες εξάγουν αποτελέσματα για κατανάλωση ισχύος καθώς και για την συμπεριφορά ου κυκλώματος σε single event upset.

2.1 FORMAT εισόδου προγράμματος

Η είσοδος του προγράμματος είναι ένα net-list σε SPICE format και έχει την ακόλουθη δομή:

- Δήλωση σημάτων πηγής και γείωσης.
- Δήλωση υπό κυκλωμάτων, η οποία περιλαμβάνει
  1. Δήλωση του ονόματος και του αριθμού εισόδων του υπό κυκλώματος
  2. περιγραφή δομής και συνδεσμολογίας σε επίπεδο τρανζίστορ καθώς και σε επίπεδο πυλών.
  3. χρήση προηγούμενος ορισμένως υπό κυκλωμάτων.
- Δήλωση των εισόδων του κυκλώματος
- Περιγραφή δομής και συνδεσμολογίας του κυκλώματος σε επίπεδο πυλών, οι πύλες που χρησιμοποιούνται είναι τα υποκυκλώματα που ορίστηκαν στο δεύτερο βήμα.

Η γραμματική της εισόδου του προγράμματος δεν είναι ίδια με αυτή του spice αλλά είναι μόνο ένα επιλεγμένο κομμάτι αυτής. Παρακάτω περιγράφεται αυτή η γραμματική.

****

vdd [όνομα τροφοδοσίας] [όνομα γείωσης] [διαφορά τάσης]

[v[αριθμός εισόδου] [όνομα εισόδου]]+

[.subckt [όνομα υπό κυκλώματος][αριθμός εισόδων][όνομα εξόδου] [όνομα εισόδου]+

[m[αριθμός τρανζίστορ] [όνομα κόμβου στο drain] [όνομα κόμβου στο gate] [όνομα

κόμβου στο source] [τύπος τρανζίστορ [modp | modn] [μέγεθος τρανζίστορ(w = μέγεθος)]]*

[x[αριθμός πύλης] [όνομα κόμβου εξόδου] [όνομα κόμβου εισόδου]+ [όνομα τροφοδοσίας] [τύπος πύλης]]*

.ends]+

[x[αριθμός πύλης] [όνομα κόμβου εξόδου] [όνομα κόμβου εισόδου]+ [όνομα τροφοδοσίας] [τύπος πύλης]]*

.end

****

Ένα παράδειγμα ενός τέτοιου αρχείου δίνεται παρακάτω:

C17.ckt

```
*c17 iscas example (to test conversion program only)
*-----------------------------------------------
*
*
*  total number of lines in the netlist ..............    17
*  simplistically reduced equivalent fault set size =     22
*        lines from primary input  es .......     5
*        lines from primary output es .......     2
*        lines from interior e outputs ......     4
*        lines from **     3 ** fanout stems ...     6
*
*        avg_fanin  =  2.00,     max_fanin  =  2
*        avg_fanout =  2.00,     max_fanout =  2
*
*
*
```

```
*
*
v1 1 0 pulse(0 5 15n)
v2 2 0 pulse(0 5 15n)   *Δήλωση των εισόδων του κυκλώματος
v3 3 0 pulse(0 5 15n)
v6 6 0 pulse(0 5 15n)
v7 7 0 pulse(0 5 15n)


.print tran v(22)
.print tran v(23)
vdd  9999 0 dc 5v

.model modn nmos vto=1.25 kp=5.75e-5

.model modp pmos vto=-0.95 kp=3.26e-5

*Δήλωση των sub circuits
*παρακάτω δηλώνονται οι πύλες not1, nand2, and2, nor2,or2


*Η πύλη not1 δηλώνεται με την χρήση 2 τρανζιστορς
.subckt not1 2 1 9999
m1 2 1 9999 9999 modp l=2u w=8u as=8p ad=8p ps=12u pd=12u
m2 2 1 0 0 modn l=2u w=8u as=16p ad=16p ps=16u pd=16u
.ends not1

*Οι πύλες nand2 , nor2 ορίζονται σε επίπεδο τρανζίστορ.
*Οι πύλες and2 και or2 ορίζονται με την χρήση των ήδη
*δηλωμένων nand2 , nor2 και με την χρήση του αντιστροφέα
*not.Η δήλωση είναι σε gate-level
.subckt nand2 3 1 2 9999
m1 3 1  9999 9999 modp l=2u w=8u as=8p ad=8p ps=12u pd=12u
m3 3 2  9999 9999 modp l=2u w=8u as=8p ad=8p ps=12u pd=12u
m2 3 1   4  0 modn l=2u w=8u as=16p ad=16p ps=16u pd=16u
m4 4 2   0  0 modn l=2u w=8u as=16p ad=16p ps=16u pd=16u
.ends nand2

.subckt and2 3 1 2 9999
x1 4 1 2 9999 nand2
x2 3 4 9999 not1
.ends and2

.subckt nor2 3 1 2 9999
m1 4 1 9999 9999 modp l=2u w=8u as=8p ad=8p ps=12u pd=12u
m3 3 2 4 9999 modp l=2u w=8u as=8p ad=8p ps=12u pd=12u
m2 3 1 0 0 modn l=2u w=8u as=16p ad=16p ps=16u pd=16u
m4 3 2 0 0 modn l=2u w=8u as=16p ad=16p ps=16u pd=16u
.ends nor2

.subckt or2 3 1 2 9999
x1 4 1 2 9999 nor2
```

```
x2 3 4 9999 not1
.ends or2


*Εδώ δηλώνεται το κυρίως κύκλωμα σε gate level.


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

Επειδή το πρόγραμμα δεν χρησιμοποιεί βιβλιοθήκες πρέπει να ορίζονται όλες οι πύλες που πρόκειται να χρησιμοποιηθούν. Ακόμα και οι πιο απλές όπως not ή nand δεν υπάρχουν προκαθορισμένες. Ωστόσο μπορεί να γίνει εύκολα μια επέκταση και να χρησιμοποιούμε βιβλιοθήκες όπου θα βρίσκονται οι ορισμοί των πυλών. Έτσι με ένα πέρασμα από ένα τέτοιο αρχείο θα μπορούν να ορίζονται οι πύλες και στην συνέχεια στο βασικό μας αρχείο να υπάρχει η περιγραφή του κυκλώματος. Θα μπορούσαμε να κατασκευάσουμε εμείς τέτοιες βιβλιοθήκες με τις βασικές πύλες (not ,nor,nand…) ενώ ο χρήστης θα έχει την δυνατότητα να κατασκευάσει και αυτός δικές του βιβλιοθήκες και να τις χρησιμοποιεί αργότερα στα κυκλώματα του.
Όταν ξεκινάμε μια προσομοίωση το πρόγραμμα διαβάζει το αρχείο μια φορά και σχηματίζει το κύκλωμα. Είναι σημαντικό να γεγονός ότι το πρόγραμμα δουλεύει με ένα μόνο πέρασμα του αρχείου κάτι που συνεπάγεται ταχύτερα αποτελέσματα. Για να γίνει αυτό το πρόγραμμα ακολουθεί μια σειρά ενεργειών. Αυτές οι εργασίες περιγράφονται στην επόμενη παράγραφο.

2.2 Διαδικασία κατασκευής γράφου

Ο γράφος κατασκευάζεται στην μνήμη κατά την ανάγνωση του αρχείου που περιγράφει το κύκλωμα. Ακολουθώντας και την δομή του αρχείου ο γράφος κατασκευάζεται ακολουθώντας τα παρακάτω βήματα:

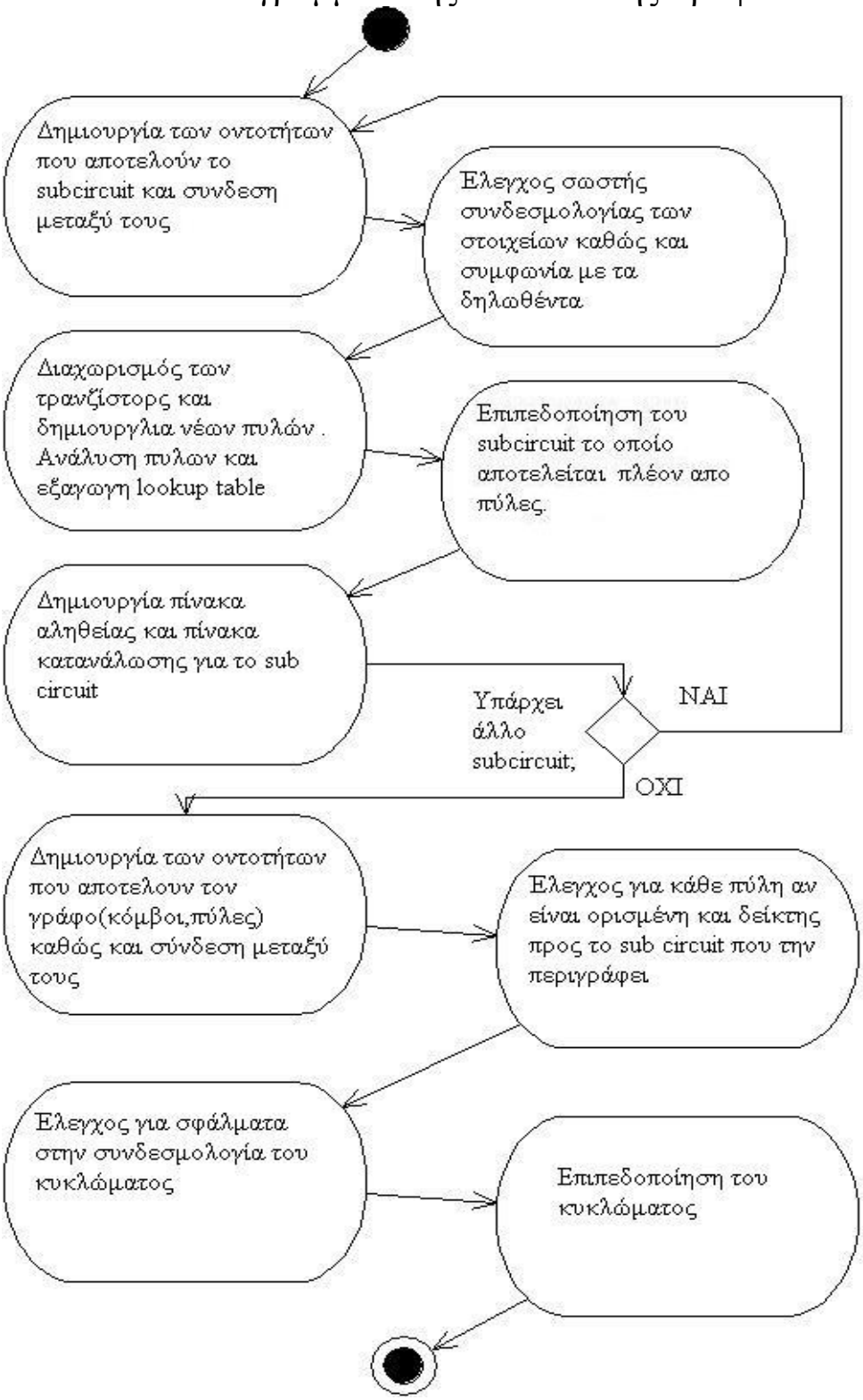
- Δημιουργία των sub circuits
  1. Δημιουργία των οντοτήτων που αποτελούν το γράφο (τρανζίστορς, κόμβων και πυλών) καθώς και συνδεσμολογία μεταξύ των στοιχείων. Επιτρέπεται η χρήση μόνο προηγούμενος δηλωμένων πυλών.

2. Έλεγχος σωστής συνδεσμολογίας. Ελέγχεται αν ο τρόπος που είναι συνδεδεμένα τα τρανζιστορς καθώς και οι πύλες τηρεί κάποιους βασικούς κανόνες ορθότητας. Επίσης ελέγχονται λεπτομέρειες όπως συμφωνία αριθμού εισόδων με αυτόν που δηλώθηκε στην επικεφαλίδα, ύπαρξη μετέωρων κόμβων κτλ.
3. Διαχωρισμός των τρανζιστορς και δημιουργία των πυλών που αυτά σχηματίζουν. Ανάλυση των πυλών και εξαγωγή πίνακα αληθείας για κάθε ένα από αυτούς.
4. Επιπεδοποίηση του υπό κυκλώματος το οποίο πλέον αποτελείται μόνο από πύλες.
5. Δημιουργία πίνακα αληθείας και πίνακα κατανάλωσης στατικής ισχύς για το υποκύκλωμα.

- Δημιουργία γράφου κυκλώματος
    1. Δημιουργία των οντοτήτων που αποτελούν τον γράφο (πύλες και κόμβοι) καθώς και σύνδεση μεταξύ τους.
    2. Έλεγχος για κάθε πύλη αν είναι ορισμένη και δημιουργία δείκτη προς το sub circuit που την περιγράφει.
    3. Έλεγχος για σφάλματα στην συνδεσμολογία του κυκλώματος.
    4. Επιπεδοποίηση του κυκλώματος.

Μετά από αυτά τα βήματα έχουμε σχηματίσει στην μνήμη μια πλήρη αναπαράσταση του κυκλώματος πάνω στην οποία μπορούμε να πραγματοποιήσουμε τις προσομοιώσεις. Έχοντας διαχωρίσει τους κόμβους σε επίπεδα όπως περιγράφεται στο κεφάλαιο 2.7 μπορούμε εύκολα να κάνουμε την προσομοίωση ενώ για κάθε πύλη έχουμε έτοιμες όλες τις πληροφορίες που θα μας χρειαστούν χωρίς να χρειάζεται να τις αναλύουμε ξανά και ξανά.

Διάγραμμα Ροής Κατασκευής Γράφου



2.3 Μορφή του γράφου

Είπαμε πιο πριν ότι το πρόγραμμα σχηματίζει ένα γράφο στην μνήμη για να προσομοιώσει το κύκλωμα. Αυτός ο γράφος σχηματίζεται κατά την ανάγνωση του αρχείου. Ο γράφος αποτελείται από τέσσερις βασικές οντότητες. Αυτές είναι η πύλη(gate), ο κόμβος(signal), το τρανζίστορ και το υποκύκλωμα (sub circuit). Για κάθε ένα από αυτά υπάρχει μια δομή η οποία περιγράφει τα χαρακτηριστικά του, τις ιδιότητες του, ενώ υπάρχουν και δείκτες από μια δομή σε άλλη. Παρακάτω γίνεται μια πιο λεπτομερή περιγραφή των δομών του γράφου καθώς και των χαρακτηριστικών τους.

2.3.1 Τα μέρη του γράφου (βασικές δομές)

Πύλη: Το βασικότερο ίσως στοιχείο του γράφου. Θα μπορούσε να θεωρηθεί και ως υποκύκλωμα καθώς τα sub circuits που ορίζονται στην αρχή του αρχείου αναπαρίστανται στο κύκλωμα με δομές πύλης οι οποίες έχουν δείκτες προς το sub-circuit που τις περιγράφει.. Υπάρχουν επίσης σύνδεσμοι που ενώνουν την πύλη με όλα τα σήματα που έχει είτε ως εισόδους είτε ως έξοδο.

Πύλη
Ονομασία πύλης
Δείκτης προς το αντίστοιχο sub circuit που την περιγράφει
Αριθμός εισόδων
Δείκτης προς λίστα τρανζιστορς (σε περίπτωση που δηλώθηκε σε transistor level)
Πίνακας με δείκτες προς τους κόμβους εισόδου
Δείκτης προς τον κόμβο εξόδου

Signal: Το σήμα μας αποτελεί έναν κόμβο. Ουσιαστικά μια απεικόνιση του ‘καλωδίου’ που ενώνει τις πύλες μεταξύ τους. Ένα απλό στοιχείο με πολύ σημαντικές όμως παραμέτρους για μας. Η τιμή του, η χωρητικότητα του, οι πύλες που οδηγεί, είναι μερικές μόνο από τις ιδιότητες που πρέπει να κρατάμε για κάθε σήμα μας. Αν και δεν έχει ιδιότητες σαν αυτές της πύλης έχει συνδέσμους σε όλες τις πύλες με τις οποίες έρχεται σε επαφή δίνοντας μας έτσι την ιδιότητα να περνάμε μέσο αυτού από πύλη σε πύλη όταν διατρέχουμε το κύκλωμα.

Κόμβος(Signal)
Ονομασία κόμβου
Επίπεδο στο κύκλωμα στο οποίο ανήκει ο κόμβος
Κατάσταση κόμβου(λογικό 1,0 ή απροσδιόριστο)
Προσωρινή κατάσταση κόμβου (χρησιμεύει στην προσομοίωση για SEU)
Χωρητικότητα
Έλεγχος διάσχισης (Pass check) Χρησιμεύει στον αλγόριθμο επιπεδοποίησης
Πίνακας με δείκτες προς όλες τις πύλες που οδηγεί
Δείκτης προς την πύλη της οποίας αποτελεί έξοδο
Δείκτες προς τρανζιστορς (με διαχωρισμό αν πρόκειται για drain, source ή gate)
Καταχωρητής ενέργειας , χρησιμεύει στην μελέτη στατικής ισχύς
Μετρητής αλλαγών κατάστασης, χρησιμεύει στην μελέτη δυναμικής ενέργειας

Τρανζίστορ: Αν και στην περιγραφή του βασικού μας κυκλώματος δεν υπάρχουν τρανζίστορ χρειάζεται κάπως να το αναπαραστήσουμε γιατί και η αναπαράσταση σε επίπεδο τρανζίστορ των sub circuits γίνεται με έναν αντίστοιχο γράφο. Οι πληροφορίες που κρατούνται από την αναπαράσταση του τρανζίστορ εκτός από τα signals με τα οποία έρχεται σε επαφή αφορούν ιδιότητες του τρανζίστορ όπως μέγεθος και τύπος (τύπου p ή n).

Τρανζίστορ
Ονομασία τρανζίστορ
Δείκτες προς τους κόμβους που ενώνεται σε drain , source και gate
Τύπος τρανζίστορ (τύπου n ή p)
Έλεγχος διάσχισης. (παρόμοιος με το pass_check του signal)

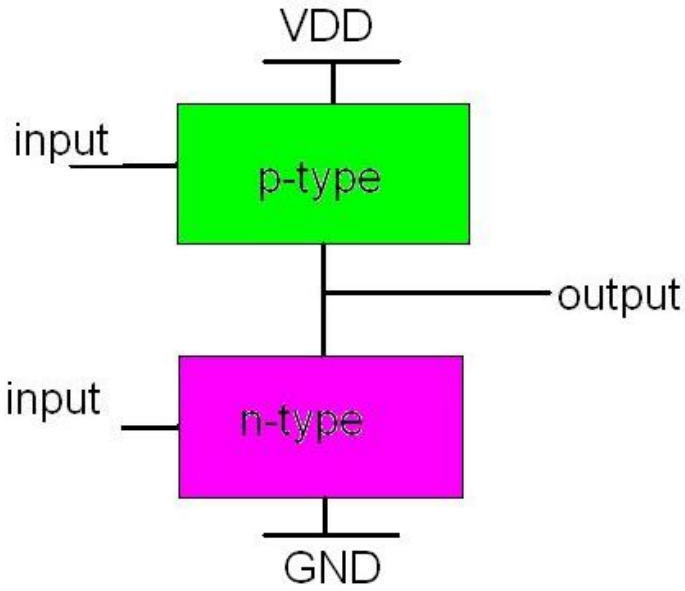
Sub circuit: Επειδή το πρόγραμμα δεν περιέχει καθιερωμένες βιβλιοθήκες, όλες οι πύλες που χρησιμοποιούνται ορίζονται πριν την περιγραφή του κυκλώματος. Επίσης το format εισόδου του προγράμματος επιτρέπει και τον ορισμό υπό κυκλωμάτων τα οποία μπορούν να χρησιμοποιηθούν αργότερα στο κύκλωμα μας ως πύλες . Όταν γίνεται ο ορισμός ενός νέου τέτοιου υποκυκλώματος μια νέα οντότητα σχηματίζεται η οποία έχει την δομή ενός sub circuit. Σε αυτό δημιουργείται ένα μικρό κύκλωμα χρησιμοποιώντας είτε τρανζιστορς είτε πύλες ή ακόμα και ένα μίγμα αυτών. Σε αυτό το κύκλωμα το πρόγραμμα αφού εκτελέσει μια σειρά εργασιών θα μπορέσει να εντοπίσει τις πύλες που το αποτελούν και να εξάγει τον πίνακα αληθείας. Στην συνέχεια επιτρέπει στον χρήστη να χρησιμοποιεί το συγκεκριμένο υποκύκλωμα όσες φορές θέλει στο κυρίως κύκλωμα του. Εκεί θα αναπαρασταθεί σαν μια πύλη η οποία θα έχει έναν δείκτη προς την συγκεκριμένη δομή του sub circuit. Σαν sub circuit θα θεωρηθεί από την πιο απλή πύλη που θα περιγραφή (π.χ. not) έως και ένα πραγματικά περίπλοκο υποκύκλωμα.

Sub circuit
Λίστα με τους κόμβους του υποκυκλώματος
Λίστα με τις πύλες του υποκυκλώματος
Λίστα με τα τρανζιστορς του υποκυκλώματος
Λίστα με τις εισόδους του υποκυκλώματος
Δείκτης προς τον κόμβο έξοδο (πρέπει να έχει μόνο έναν)
Όνομα υποκυκλώματος
Δισδιάστατος πίνακας όπου θα μπουν τα vector εισόδων για την δημιουργία lookup table
Πίνακας με τις εξόδους του υποκυκλώματος για κάθε vector εισόδων του παραπάνω πίνακα
Πίνακας με τις αντίστοιχες τιμές κατανάλωσης στατικής ισχύος για κάθε vector εισόδων

2.4 Δημιουργία των πυλών

Όπως αναφέραμε και πριν στο πρώτο βήμα της δημιουργίας του κυκλώματος πρέπει να οριστούν οι πύλες που θα χρησιμοποιηθούν. Κάθε πύλη ορίζεται ουσιαστικά σαν

ένα υποκύκλωμα με εισόδους και μια έξοδο. Μέσα σε αυτό μπορούν να χρησιμοποιηθούν τρανζίστορ CMOS αλλά και πύλες που έχουν οριστεί πιο πάνω. Κάθε τρανζίστορ έχει πληροφορίες όπως τα σήματα course drain και gate, τον τύπο του n-type ή p-type καθώς και το μέγεθος του. Το format του αρχείου επιτρέπει και συνδεσμολογία πυλών με τρανζίστορ. Ωστόσο η συνδεσμολογία των τρανζίστορ μπορεί να γίνει μόνο με τον τρόπο που φαίνεται στο σχήμα 2.1. Εάν μετά την σύνδεση των τρανζιστορς στον γράφο το πρόγραμμα διαπιστώσει ότι δεν τηρείται αυτός ο κανόνας τότε διακόπτεται η προσομοίωση. Θεωρεί ότι είτε πρόκειται για λάθος στην συνδεσιμότητα είτε έχουν ακολουθηθεί κανόνες που το πρόγραμμα δεν μπορεί να προσομοιώσει (πχ C-switch). Το πρόγραμμα δεν αναγνωρίζει τις λογικές πράξεις που επιτελεί μια πύλη. Από την συνδεσμολογία εξάγει τον πίνακα αληθείας και από εκεί μπορεί να παίρνει όποτε χρειάζεται την έξοδο του υποκυκλώματος. Σαν πύλη μπορούν να οριστούν και μεγαλύτερα κυκλώματα τα οποία θα είναι υποκυκλώματα στο βασικό μας κύκλωμα. Το πρόγραμμα θα δημιουργήσει και για αυτά τον πίνακα αληθείας ενώ θα κάνει και όλους τους υπολογισμούς για να υπολογίσει στατική ενέργεια και χωρητικότητα.



Σχήμα 2.1
Εσωτερική απεικόνιση πύλης. Συνδεσμολογία τρανζιστορς

2.4.1 Συνδεσιμότητα των τρανζιστορς

Όπως φαίνεται στο παράδειγμα του αρχείου εισόδου στα sub circuits, όπου γίνεται χρήση τρανζιστορς η περιγραφή γίνεται με τον ορισμό των τρανζίστορ ένα-ένα αναφέροντας ταυτόχρονα και τους κόμβους με τους οποίους έρχονται σε επαφή. Επειδή όμως ο προσομοιωτής δουλεύει με μια μονή ανάγνωση του αρχείου η συνδεσμολογία πρέπει να γίνεται αμέσως με την δημιουργία των οντοτήτων των τρανζίστορ. Έτσι με το τέλος της ανάγνωσης και του τελευταίου τρανζίστορ ο γράφος θα πρέπει να είναι έτοιμος με όλα τα τρανζιστορς ενωμένα. Αυτό επιτυγχάνεται με την χρήση της δομής του γράφου που αναφέραμε. Όταν ένα τρανζίστορ ορίζεται η δήλωση του έχει το εξής format:

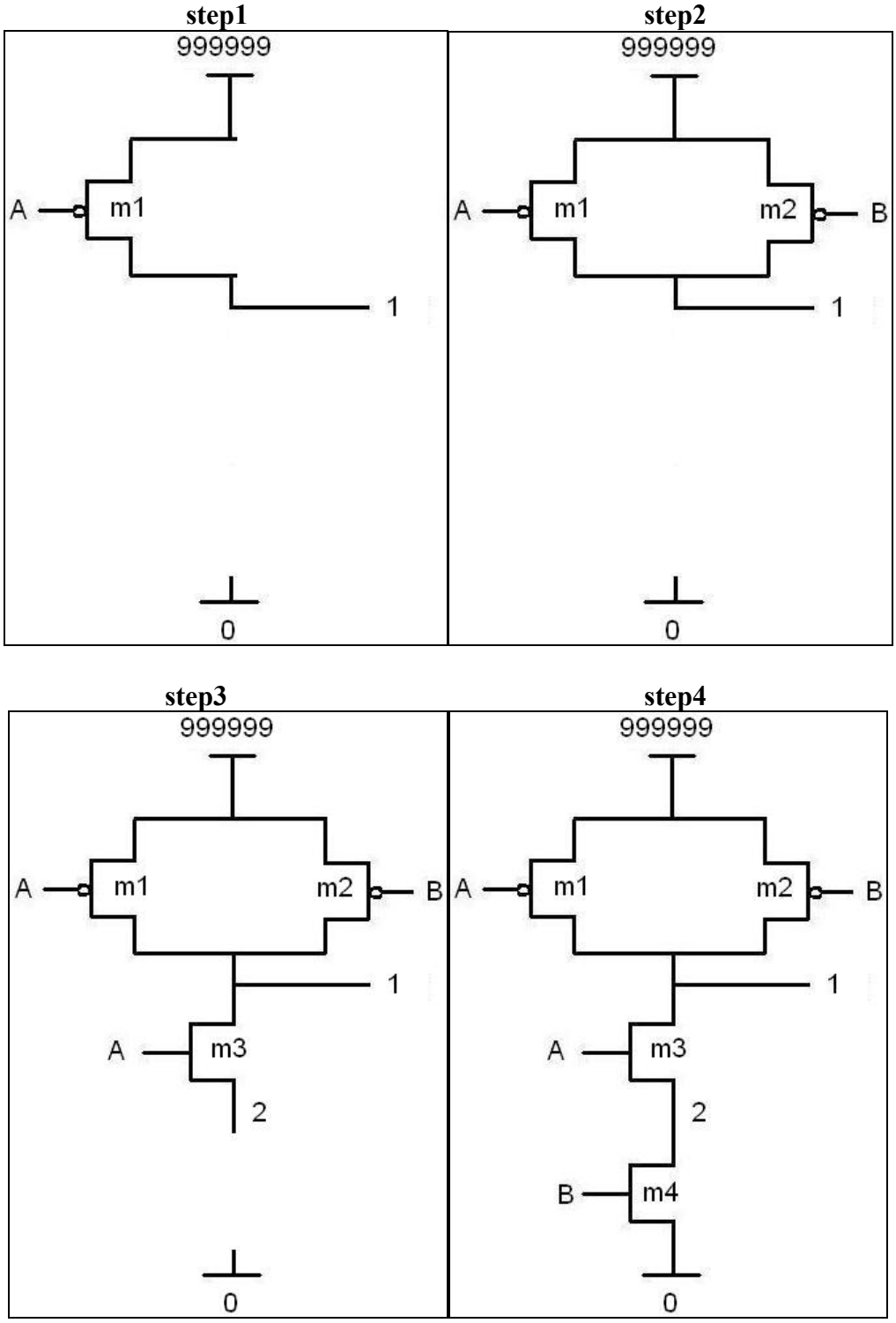
```
m3 1 2 3 999999 modp l=2u w=8u as=8p ad=8p ps=12u pd=12u
```

Από αυτή την δήλωση έχουμε ότι **m3** είναι το όνομα του συγκεκριμένου τρανζίστορ, **1** είναι το drain , **2** είναι το gate και **3** είναι το source. Τα υπόλοιπα είναι χαρακτηριστικά του τρανζίστορ όπως τύπος, μέγεθος κτλ. Από έναν τέτοιο ορισμό έχουμε ότι εκτός από ένα νέο τρανζίστορ με όνομα m3 έχουμε και τρεις κόμβους , τους 1,2,3 οι οποίοι είτε υπάρχουν, είτε πρέπει να δημιουργηθούν και είναι συνδεδεμένοι με το τρανζίστορ με τον τρόπο που προαναφέραμε. Με αυτό τον τρόπο κατά την δήλωση ενός sub circuit σε transistor level , αν η συνδεσμολογία δεν είναι λανθασμένη τότε με την δημιουργία του τελευταίου τρανζίστορ ο γράφος που θα δημιουργηθεί θα είναι πλήρης. Παρακάτω ακολουθεί ένα παράδειγμα ορισμού της πύλης NAND2 σε transistor level με 2 τρόπους. Και στις 2 περιπτώσεις η συνδεσμολογία γίνεται σωστά ανεξαρτήτως της σειράς ορισμού των τρανζιστορς.

Περίπτωση 1η

```
m1 999999 A 1 999999 modp l=2u w=8u as=8p ad=8p ps=12u pd=12u
m2 999999 B 1 999999 modp l=2u w=8u as=8p ad=8p ps=12u pd=12u
m3 1 A 2 0 modn as=16p ad=16p ps=16u pd=16u
m4 2 B 0 0 modn as=16p ad=16p ps=16u pd=16u
```

Ακολουθώντας αυτή την σειρά ο προσομοιωτής θα φτιάξει τον γράφο με τα τρανζιστορς με τον παρακάτω τρόπο:



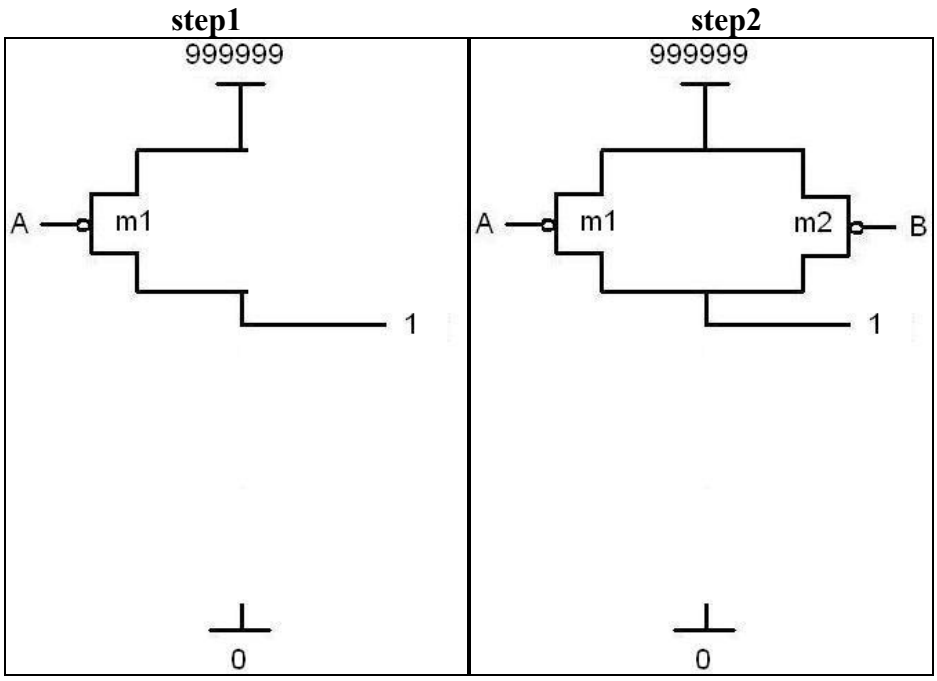
Σχήμα 2.2.1
Συνδεσιμότητα τρανζιστορς περίπτωση 1η

Ωστόσο η δήλωση των τρανζιστορς θα μπορούσε να γίνει και με άλλη σειρά. Έτσι για παράδειγμα έχουμε την δήλωση με την παρακάτω σειρά:

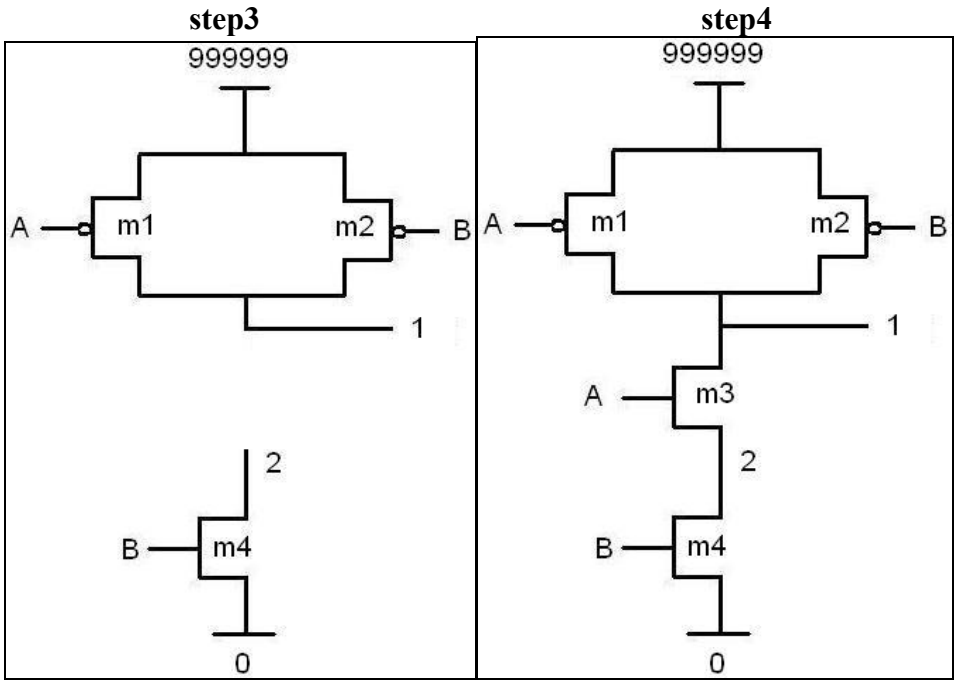
Περίπτωση 2η

```
m1 999999 A 1 999999 modp l=2u w=8u as=8p ad=8p ps=12u pd=12u
m2 999999 B 1 999999 modp l=2u w=8u as=8p ad=8p ps=12u pd=12u
m4 2 B 0 0 modn as=16p ad=16p ps=16u pd=16u
m3 1 A 2 0 modn as=16p ad=16p ps=16u pd=16u
```

Σε αυτήν την περίπτωση τα δυο n-type τρανζιστορς δηλώνονται με ανάποδη σειρά. Στο σχήμα 2.2.2 φαίνεται ο τρόπος δημιουργίας του γράφου σε αυτήν την περίπτωση.



Στο βήμα 3 το τρανζίστορ m4 δημιουργείται μαζί με τον κόμβο 2 και ας μένει αποκομμένος από τον υπόλοιπο γράφο. Στο βήμα 4 το τρανζίστορ m3 θα δημιουργηθεί και θα ενωθεί με τους κόμβους 1 και 2 οι οποίοι είναι ήδη ορισμένοι, ενώνοντας έτσι τα σπασμένα κομμάτια του γράφου.



Σχήμα 2.2.2
Συνδεσιμότητα τρανζιστορς περίπτωση 2η

Το τελευταίο τρανζίστορ m3 ενώνεται απευθείας με τους κόμβους 1 και 2 οι οποίοι έχουν ήδη οριστεί από πριν.

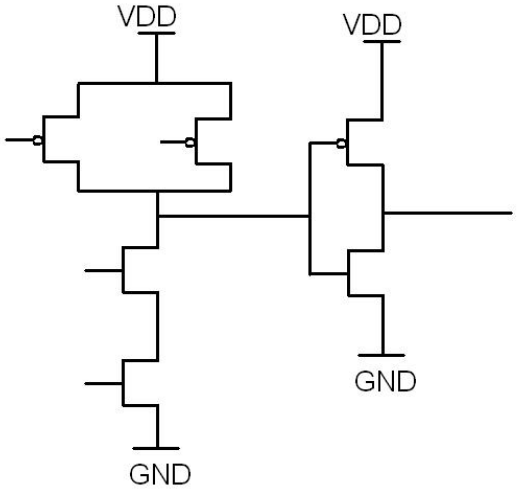
2.4.2 Διαχωρισμός των πυλών σε επίπεδο τρανζίστορ

Κατά την δημιουργία ενός υποκυκλώματος (sub circuit) το πρόγραμμα πρέπει να εξάγει πίνακα αληθείας. Αυτό είναι εύκολο όταν αυτά αποτελούνται από μια πύλη (π.χ. not, nand, nor), όταν όμως γίνονται περισσότερο πολύπλοκα τότε χρειάζεται να γίνουν κάποιες παραπάνω λειτουργίες για να έχουμε το επιθυμητό αποτέλεσμα. Πρέπει να αναλύσουμε το υποκύκλωμα στις πύλες που το αποτελούν και στην συνέχεια να προχωρήσουμε. Όταν η περιγραφή κάνει αυτό τον διαχωρισμό τότε τα πράγματα είναι εύκολα. Ένα απλό παράδειγμα είναι η πύλη **and** η οποία αποτελείται από μια **nand** και μια **not** σε σειρά τις οποίες τις έχει ορίσει πιο πάνω στο αρχείο. Ακόμα και πιο περίπλοκα κυκλώματα μπορεί να τα περιγράψει με την χρήση έτοιμων πυλών.

Υπάρχει το ενδεχόμενο η περιγραφή του υπό κυκλώματος να γίνεται σε επίπεδο τρανζίστορ με περισσότερα του ενός λογικά επίπεδα. Σε αυτή την περίπτωση πρέπει να κάνουμε την ανάλυση και να βρούμε τις πύλες που το αποτελούν. Πύλες που πρέπει να είναι της μορφής του σχήματος 2.1 και που στην συνέχεια θα αναλύσουμε κάθε μια χωριστά για να βρούμε πίνακα αληθείας αλλά και στατική ενέργεια.

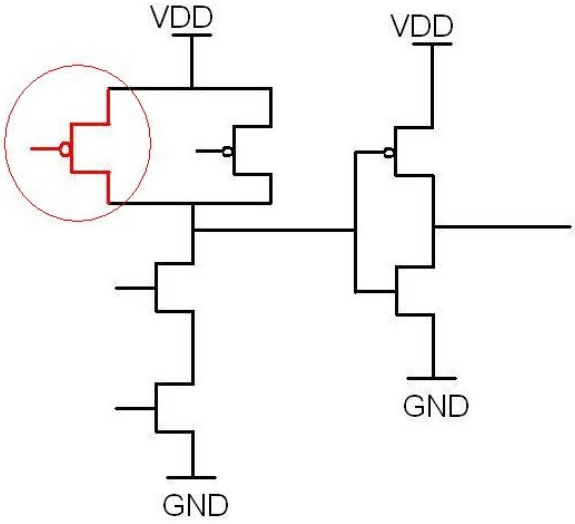
Το πρώτο βήμα σε αυτή την διαδικασία είναι να βρούμε τις πύλες από τα τρανζίστορ. Εφόσον οι πύλες είναι τις μορφής που δείχνει το σχήμα 2.1 τότε αυτό που έχουμε να κάνουμε είναι να βρούμε όλα τα τρανζίστορ που είναι ενωμένα με αυτόν τον τρόπο

και τα οποία αποτελούν μια πύλη. Αυτό γίνεται ξεκινώντας από ένα τρανζίστορ και προχωρώντας πάνω και κάτω (παίρνοντας αυτά που ενώνονται σε course και drain και όχι στο gate) μαρκάρουμε όλα τα τρανζίστορ που βρίσκουμε. Το ίδιο κάνουμε και για τα μαρκαρισμένα τρανζίστορ μέχρι να εξαντληθούν όλα όσα μπορούμε να μαρκάρουμε. Αυτά τα τρανζίστορ αποτελούν μια πύλη.
Παρακάτω φαίνεται και σχηματικά πως θα εξελιχθεί αυτή η διαδικασία σε μια πολύ γνωστή περίπτωση τέτοιου υποκυκλώματος.



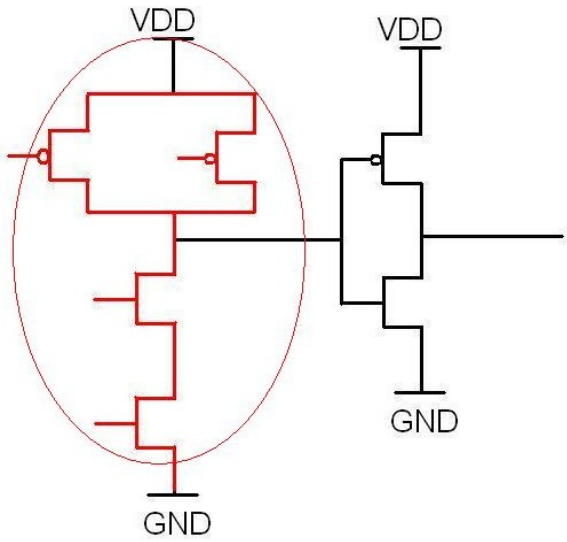
Σχήμα 2.3.1
Διαδικασία διαχωρισμού πυλών

Ας υποθέσουμε ότι στην παραπάνω πύλη, την γνωστή μας **AND** ξεκινάμε από ένα τρανζίστορ, έστω το πάνω αριστερά όπως φαίνεται στο σχήμα 2.3.2.



Σχήμα 2.3.2

Ακολουθώντας αυτήν την διαδικασία φτάνουμε στο παρακάτω αποτέλεσμα με όλα τα μαρκαρισμένα τρανζίστορ να αποτελούν μια πύλη.



Σχήμα 2.3.3

Ο αλγόριθμος που επιτυγχάνει τον διαχωρισμό των πυλών παρατίθεται παρακάτω.

Αλγόριθμος 1 (Διαχωρισμός πυλών)

```
1:  make_gate_by_tranz(transistor a, transistor list b){
2:    if(transistor a do not exists in b   AND TPASS(a)=false){
3:              insert a into b
4:              TPASS(a) = true
5:              for(each transistor t is connected to DRAIN(a))
6:                      make_gate_by_tranz(t,b)
7:              for(each transistor t is connected to SOURSE(a))

8:                      make_gate_by_tranz(t,b)
9:              TPASS(a) = false
10:     }
```

Καλώντας τον παραπάνω αλγόριθμο με αρχικά την λίστα b άδεια και σαν τρανζίστορ a αρχικά ας πούμε το σημαδεμένο τρανζίστορ του σχήματος 2.3.2 έχουμε τελικά όλα τα τρανζίστορ που αποτελούν την πύλη στη b. Στην συνέχεια της διαδικασίας το πρόγραμμα θα εξετάσει την νέα πύλη που δημιουργήθηκε και θα εξάγει τον πίνακα αληθείας και τον πίνακα στατικής ενέργειας.

Αυτή η διαδικασία θα ακολουθηθεί και για τα υπόλοιπα τρανζίστορ. Στο τέλος αφού έχουμε εξαντλήσει όλα τα τρανζίστορ θα έχουμε μόνο πύλες που να

περιγράφουν το υποκύκλωμα μας. Σκοπός μας είναι να φτιάξουμε τον πίνακα αληθείας και τον πίνακα με την στατική ενέργεια για το sub circuit. Ο τρόπος που βγαίνει ο πίνακας αληθείας ενός κυκλώματος ,έστω και υποκυκλώματος, διαφέρει από εκείνον μιας πύλης. Έχουν το κοινό στοιχείο ότι θα βάλουμε σαν είσοδο όλα τα πιθανά vector, ωστόσο η ανάλυση του κυκλώματος είναι πιο περίπλοκη. Στο υποκύκλωμα χρειάζεται να κάνουμε προσομοίωση όμοια με αυτήν του κυρίως κυκλώματος οπότε η ανάλυση της μεθόδου θα γίνει εκεί . Ο πίνακας με την στατική ενέργεια σχηματίζεται με το να αθροίζουμε σε κάθε πέρασμα την ενέργεια κάθε πύλης του υποκυκλώματος, τιμή που έχουμε έτοιμη στον αντίστοιχο πίνακα κάθε πύλης. Αυτή η μέθοδος είναι παρόμοια με την τεχνική προσομοίωσης για τον υπολογισμό της στατικής ισχύος.

2.5 Δημιουργία πίνακα αληθείας

Το κομμάτι όπου εξετάζουμε πως δημιουργείται ο πίνακας αληθείας χωρίζεται σε δυο μέρη. Στο μέρος όπου εξετάζουμε μια πύλη σε επίπεδο τρανζίστορ και σε αυτό όπου εξετάζουμε ένα υποκύκλωμα αποτελούμενο από έναν αριθμό πυλών. Το πρώτο κομμάτι είναι αυτό που θα μας απασχολήσει τώρα ενώ το δεύτερο όπως αναφέραμε θα μελετηθεί σαν υποπερίπτωση της προσομοίωσης που γίνεται στο βασικό μας κύκλωμα.

2.5.1 Πίνακας αληθείας σε επίπεδο τρανζίστορ

Έχοντας διαχωρίσει τα τρανζίστορ που αποτελούν την προς εξέταση πύλη επόμενο βήμα είναι να βρούμε την έξοδο. Το πρόγραμμα θεωρεί ότι η έξοδος πρέπει να είναι το μοναδικό σήμα που ακουμπάει τόσο σε τρανζίστορ τύπου p όσο και σε τρανζίστορ τύπου n. Αυτό γιατί θεωρεί ότι η πύλη ακολουθεί την δομή του σχήματος 2.1. Επίσης πρέπει να βρεθούν τα σήματα εισόδου της πύλης. Είναι φανερό ότι σε κάθε τρανζίστορ αντιστοιχεί και ένα σήμα το οποίο είναι είσοδος και το οποίο καταλήγει στην υποδοχή gate. Επομένως εφόσον έχουμε όλα τα τρανζίστορ και η συνδεσμολογία έχει γίνει μπορούμε εύκολα να βρούμε πόσα και ποια είναι τα signals εισόδου. Τα signals αυτά θα μπουν σε μια λίστα προκειμένου να πάρουν όλους τους πιθανούς συνδυασμούς τιμών και αντίστοιχα για κάθε τιμή να βρεθεί η τιμή του signal εξόδου, δημιουργώντας έτσι βήμα-βήμα τον πίνακα αληθείας.
Αρχικά για να βγάλουμε την έξοδο πρέπει να έχουμε ένα vector με τις τιμές των σημάτων εισόδου. Αν έχουμε N εισόδους τότε ο αριθμός των διαφορετικών vectors που πρέπει να εξετάσουμε είναι 2^N . Έτσι όταν ξεκινά η διαδικασία της εξαγωγής του πίνακα αληθείας δημιουργούνται αρχικά 3 πίνακες. Ένας $N \text{ X } 2^N$ και δύο $1\text{X}\,2^N$.Ο πρώτος θα περιέχει όλα τα vectors εισόδου που είναι ουσιαστικά 2^N πίνακες-στήλες N στοιχείων , ενώ ο δεύτερος και ο τρίτος είναι δυο πίνακες-στήλη N στοιχείων με τις αντίστοιχες τιμές που θα πάρει η έξοδος και το ποσό στατικής ενέργειας που καταναλώνεται με αυτό το input. Η διαδικασία θα είναι να περάσουν και οι 2^N vectors από την είσοδο και να συμπληρωθούν οι άλλοι δυο πίνακες. Τα

σήματα εισόδου είναι ταξινομημένα και το κάθε ένα αντιστοιχίζεται σε μια στήλη. Μια συνάρτηση επαναλαμβάνεται 2^N φορές και περνάει τις τιμές από τον πίνακα στις εισόδους. Στην συνέχεια ο τρόπος που εξάγουμε την έξοδο είναι κοιτώντας μεταξύ εξόδου-τροφοδοσίας και εξόδου-γείωσης. Για κάθε περίπτωση βρίσκουμε όλα τα μονοπάτια μεταξύ τάσης(VDD) και εξόδου καθώς και μεταξύ γείωσης και εξόδου. Σε κάθε μονοπάτι υπολογίζουμε τον αριθμό των τρανζίστορ που είναι σε αποκοπή. Από αυτές τις τιμές υπολογίζουμε τόσο την τιμή της εξόδου όσο και το ποσό στατικής ενέργειας.
Η τιμή εξόδου υπολογίζεται βρίσκοντας το μονοπάτι με τα λιγότερα τρανζίστορ σε αποκοπή. Από το αποτέλεσμα υπολογίζουμε μέσο του πίνακα 2.1 την τιμή εξόδου.

Πίνακας 2.1

Τρανζίστορ σε αποκοπή μεταξύ εξόδου-τροφοδοσίας	Τρανζίστορ σε αποκοπή μεταξύ εξόδου-γείωσης	Τιμή εξόδου
>0	**0**	(λογικό)**0**
0	**>0**	(λογικό)**1**
>0	**>0**	**Απροσδιόριστο**
0	**0**	**Βραχυκύκλωμα**

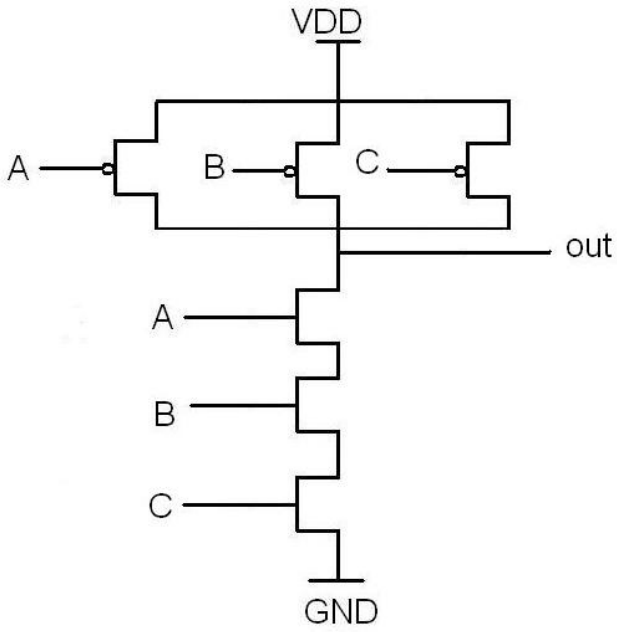
Στις περιπτώσεις που έχουμε απροσδιόριστη τιμή (η έξοδος είναι αποκομμένη από τροφοδοσία και γείωση) ή βραχυκύκλωμα το πρόγραμμα ενημερώνει τον χρήστη. Στις υπόλοιπες περιπτώσεις από τις δυο τιμές που μετράμε έχουμε μια μηδενική και μια μεγαλύτερη του μηδενός. Η μηδενική τιμή ουσιαστικά σημαίνει ότι δεν υπάρχει αποκοπή μεταξύ εξόδου και τάσης, ή γείωσης, οπότε η έξοδος φορτίζεται ανάλογα.

Όσων αφορά τον υπολογισμό της στατικής ενέργειας αυτή γίνεται με την χρήση του πίνακα 2.2. Αυτός ο πίνακας είναι για συγκεκριμένη τεχνολογία (0,18μ). Θα μπορούσαμε να δημιουργήσουμε και για άλλη τεχνολογία εάν τα μοντέλα που μας δίνονται περιέγραφαν αξιόπιστα την λειτουργία στην αποκοπή. Για κάθε μονοπάτι υπολογίζουμε τον αριθμό των τρανζίστορς σε αποκοπή και από τον πίνακα προσθέτουμε το αντίστοιχο ποσό στην συνολική κατανάλωση στατικής ενέργειας της πύλης. Έτσι στο τέλος η τιμή που υπολογίσαμε για το κομμάτι που είναι σε αποκοπή είναι και η τιμή που θέλουμε. Πρέπει να αναφέρουμε πως οι τιμές στον πίνακα είναι το ρεύμα διαρροής και είναι σε μΑ.

Πίνακας 2.2

Αριθμός τρανζίστορ σε αποκοπή	Ρεύμα διαρροής σε τρανζίστορ p-type (μA)	Ρεύμα διαρροής σε τρανζίστορ n-type (μA)
1	100	200
2	10	20
3	4	8
4	2	4
5+	1	2

Παρακάτω ακολουθεί ένα απλό παράδειγμα του πως λειτουργεί το σύστημα πάνω σε μια απλή πύλη.



Σχήμα 2.4
Η πύλη NAND3 σε επίπεδο τρανζίστορ

Πίνακας 2.3

A	B	C	Ελάχιστα τρανζίστορ σε αποκοπή μεταξύ εξόδου-τροφοδοσίας	Ελάχιστα τρανζίστορ σε αποκοπή μεταξύ εξόδου-γείωσης	Τύπος τρανζίστορ σε αποκοπή	Τιμή εξόδου Out	Ποσό στατικής ενέργειας
0	0	0	0	3	n	1	8
0	0	1	0	2	n	1	20
0	1	0	0	2	n	1	20
0	1	1	0	1	n	1	200
1	0	0	0	2	n	1	20
1	0	1	0	1	n	1	200
1	1	0	0	1	n	1	200
1	1	1	1	0	p	0	300

Ο αλγόριθμος με τον οποίον το πρόγραμμα τρέχει το κύκλωμα με τα τρανζιστορς παρατίθεται πιο κάτω. Πρόκειται για έναν αναδρομικό αλγόριθμο ο οποίος τρέχει πάνω σε έναν γράφο. Ο γράφος είναι η αναπαράσταση του sub circuit στην μνήμη όπως περιγράψαμε πιο πάνω. Στην συγκεκριμένη περίπτωση χρησιμοποιούμε το κομμάτι που περιέχει τα τρανζιστορς. Αν το sub circuit περιγράφεται μόνο με την χρήση πυλών δεν υπάρχει λόγος να γίνει αυτή η εργασία. Αυτός ο αλγόριθμος ξεκινά αρχικά από την τροφοδοσία και διασχίζοντας μέσα από τα τρανζιστορς ψάχνει να βρει την έξοδο. Σαν έξοδο δεχόμαστε τον πρώτο κόμβο που θα βρούμε ο οποίος έρχεται σε επαφή τόσο με τρανζίστορ τύπου n όσο και με τρανζίστορ τύπου p.Αν η συνδεσμολογία είναι σωστή τότε αυτή πρέπει να είναι η έξοδος της πύλης.. Όταν ένα τρανζίστορ από το οποίο περνάει είναι σε αποκοπή αυξάνει έναν μετρητή. Κάνοντας μια διάσχιση του γράφου τύπου **Depth First Search** (DFS) βρίσκει όλα τα μονοπάτια από τροφοδοσία προς έξοδο και για κάθε ένα τον αριθμό των τρανζιστορς σε αποκοπή. Σημαδεύοντας τα signals από όπου περνάει αποφεύγονται οι άσκοπες λούπες. Κάθε φορά που φτάνει στην έξοδο θεωρεί ότι έφτασε στο μέγιστο βάθος του γράφου και γυρίζει πίσω. Επίσης αυτό είναι το σημείο κοιτάει τον αριθμό από τρανζίστορ σε αποκοπή που βρήκε. Από αυτές τις τιμές θέλουμε αρχικά την ελάχιστη. Επίσης για κάθε τιμή που βρίσκει σε κάθε διαδρομή προσθέτει και το αντίστοιχο ποσό ενέργειας. Όταν τελειώσει επαναλαμβάνει την ίδια διαδικασία αυτή την φορά από την γείωση προς την έξοδο.

Αλγόριθμος 2(Υπολογισμός Αριθμού Τρανζίστορς σε αποκοπή)

```
1:    calculate_output(signal*check,int NoOfTransist, int
      *MinNoOfTranist ,signal*output,long int*static_energy,
      char type){
2:
3:    int next_value
4:
5:          MPASS(check)=pass
6:          for(each transistor t connected to check){
7:    /*Αν το transistor t είναι σε αποκοπή*/
8:          if((TYPE(t)='p' AND GATE(p)='1')OR (TYPE(t)='n'
9:          AND GATE(p)='0'))
10:                         next_value= NoOfTransist+1
11:   /*Διαφορετικά*/   else
12:                         next_value= NoOfTransist
13:
14:/*Πάμε στον κόμβο source του τρανζίστορ αν δεν είναι
15:μαρκαρισμένος*/
16:         if(MPASS(SOURCE(t))=not pass){
17:         /*Κοιτάμε αν φτάσαμε στην έξοδο*/
18:               if(SOURCE(t)=output){
19:                     if(TYPE(t)='p'){
20:                      switch(next_value){
21:                            case 0: break;
22:                               case 1:* static_energy +=100;
23:                                     break;
24:                                case 2:* static_energy +=10;
25:                                   break;
26:                                case 3:* static_energy +=4;
27:                                   break;
28:                                case 4:* static_energy +=2;
29:                                      break;
30:                                default:* static_energy +=1;
31:                                      break;
```

```
32:                                 }//τέλος switch
33:                                 }//telos if
34:                             if(TYPE(t)='n'){
35:                                     switch(next_value){
36:                                 case 0: break;
37:                                 case 1:* static_energy +=200;
38:                                         break;
39:                                 case 2:* static_energy +=20;
40:                                    break;
41:                                 case 3:* static_energy +=8;
42:                                    break;
43:                                 case 4:* static_energy +=4;
44:                                     break;
45:                                 default:* static_energy +=2;
46:                                      break;
47:                                 }//τέλος switch
48:                                 }//telos if(TYPE(t)='n')
49:
50              }       /*telos if(SOURCE(t)=output)*/
51:             calculate_output(SOURCE(check),next_value,
            MinNoOfTranist,output, static_energy ,type);
52:         }/*telos if(MPASS(SOURCE(t))=not pass)*/
```

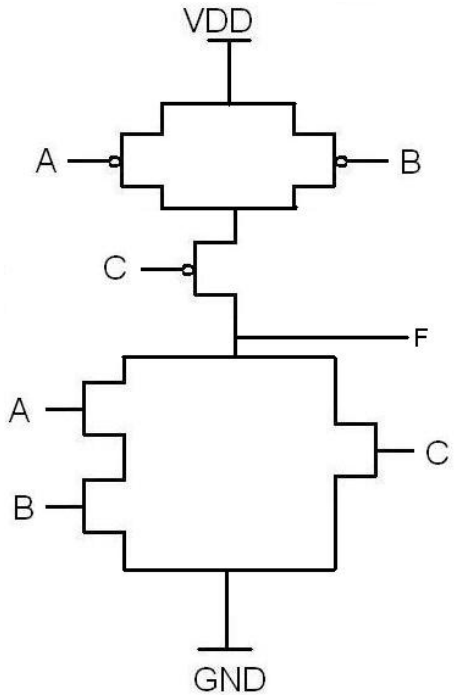
/*Επαναλαμβάνεται ο κώδικας από την γραμμή 16 έως την 52 αλλά αντί για τον κόμβο source του τρανζίστορ κοιτάμε τον κόμβο drain.*/

}

Ο παραπάνω αλγόριθμος μπορεί να δουλέψει και για πιο περίπλοκες πύλες. Ας θεωρήσουμε για παράδειγμα ότι έχουμε την πύλη που υλοποιεί την λογική πράξη

$$F = \overline{AB + C}$$

Μια τέτοια πύλη μπορεί να σχεδιαστεί με τον παρακάτω τρόπο:

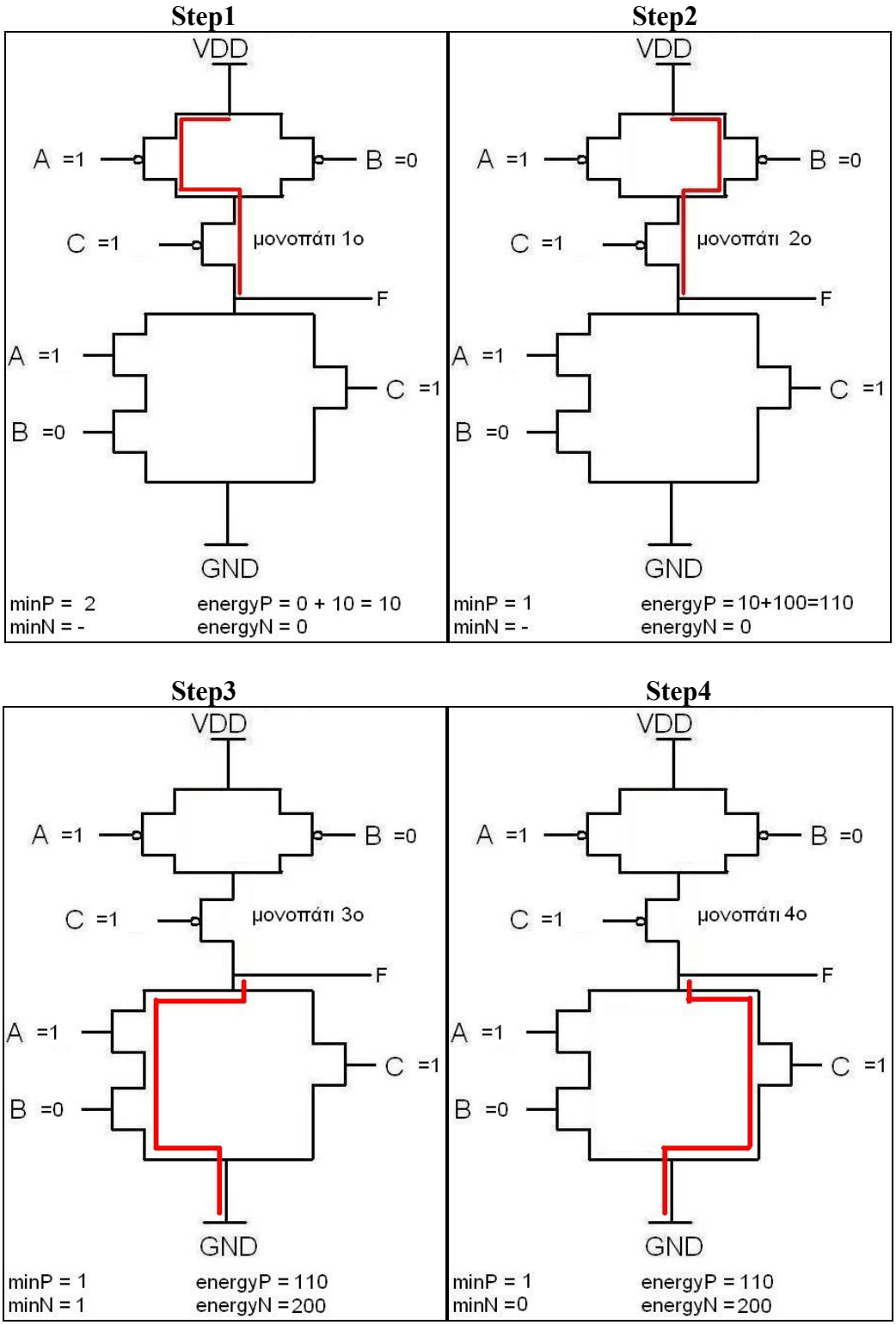


Σχήμα 2.5

Η πύλη της εξίσωσης $F = \overline{AB + C}$ σε επίπεδο τρανζίστορ

Έχουμε 3 εισόδους, άρα N = 3. Αρχικά πρέπει να σχηματίσουμε τον πίνακα με όλους τους δυνατούς συνδυασμούς τιμών που μπορούν να πάρουν οι είσοδοι. Συνολικά θα έχουμε 8 δυνατούς συνδυασμούς. Επειδή είναι λίγο δύσκολο να δείξουμε πως θα δουλέψει ο αλγόριθμος και για τις 8 περιπτώσεις θα εξετάσουμε μια από αυτές. Έστω ότι έχουμε A = 1, B=0 , C=1. Παρακάτω φαίνονται σχηματικά οι ενέργειες που κάνει ο αλγόριθμος για κάθε διαδρομή.

Η διάσχιση του γράφου χωρίζεται σε 2 κομμάτια. Το πρώτο ακολουθεί τις διαδρομές από την πηγή προς την έξοδο και στο δεύτερο από την γείωση προς την έξοδο. Για κάθε μια περίπτωση κρατάει σε δύο μεταβλητές τον ελάχιστο αριθμό τρανζιστορς σε αποκοπή και το ποσό ενέργειας που καταναλώνει. Στο τέλος θα έχουμε 2 ποσά ενέργειας. Ωστόσο σε μια από τις δυο περιπτώσεις ο ελάχιστος αριθμός τρανζιστορς σε αποκοπή θα πρέπει να είναι 0. Το αντίστοιχο ποσό ενέργειας σε αυτήν την περίπτωση αγνοείται και δεχόμαστε το άλλο.



Σχήμα 2.6
Διαδικασία εξαγωγής πίνακα αληθείας σε επίπεδο τρανζίστορ

Βλέπουμε με το τελείωμα της διαδικασίας έχουμε ότι για την διαδρομή από VDD σε έξοδο έχουμε ελάχιστο αριθμό τρανζιστορς σε αποκοπή 1 και ποσό στατικής ενέργειας ίσο με 110. Από την άλλη για την διαδρομή από GND στην έξοδο έχουμε ελάχιστο αριθμό τρανζιστορς σε αποκοπή ίσο με 0 και πόσο ενέργειας 200. Το γεγονός ότι υπάρχει μονοπάτι μεταξύ GND και εξόδου με κανένα τρανζίστορ σε αποκοπή σημαίνει ότι όλα άγουν όποτε η έξοδος F για τις συγκεκριμένες τιμές εισόδων θα παίρνει τιμή λογικό 0. Σύμφωνα επίσης με τα όσα είπαμε πριν ,το ποσό στατικής ενέργειας για αυτήν την περίπτωση θα είναι 110. Ακολουθώντας την ίδια διαδικασία για όλα τα vector εισόδων σχηματίζεται ο πίνακας αληθείας καθώς και ο πίνακας στατικής ενέργειας.

2.5.2 Πίνακας αληθείας σε επίπεδο πυλών

Εκτός από τις περιπτώσεις όπου ένα υποκύκλωμα αποτελεί την περιγραφή μιας πύλης σε επίπεδο τρανζίστορ, υπάρχει και η περίπτωση το υποκύκλωμα να αποτελείται από περισσότερες της μιας πύλες. Πύλες οι οποίες είτε είναι πρωτύτερα δηλωμένες, είτε περιγράφονται σε επίπεδο τρανζίστορ. Στην δεύτερη περίπτωση θα γίνει διαχωρισμός των πυλών (κεφ. 2.4.2) και θα βγουν ο πίνακας αληθείας και πίνακας ενέργειας για κάθε πύλη (κεφ.2.5.1).

Σε αυτό το σημείο θα έχουμε για το υποκύκλωμα μας περιγραφή σε επίπεδο πυλών. Στην ουσία θα έχουμε να κάνουμε με ένα μικρό κύκλωμα το οποίο θα μπορούσε κάλλιστα να είναι και το βασικό κύκλωμα ενός τέτοιου αρχείου καθώς αποτελείται μόνο από πύλες. Σκοπός μας είναι να εξάγουμε για αυτό το υποκύκλωμα πίνακα αληθείας και πίνακα στατικής ενέργειας. Έτσι θα έχουμε το πλεονέκτημα όταν κάνουμε την ανάλυση του κυρίως κυκλώματος ότι θα μπορούμε να θεωρήσουμε αυτό το υποκύκλωμα σαν μια πύλη με τον πίνακα αληθείας και τον πίνακα στατικής ενέργειας. Αυτό συνεπάγεται ότι στον γράφο του κυκλώματος ολόκληρο το υποκύκλωμα θα αποτελεί μια πύλη η οποία με έναν δείκτη θα έχει πρόσβαση στο sub circuit όπου έχουν γίνει όλες οι απαραίτητες μετρήσεις και έχουμε έτοιμα όλα τα αποτελέσματα.

Για να εξάγουμε τα πίνακα αληθείας και πίνακα ενέργειας δημιουργούμε πάλι τους πίνακες που δημιουργήσαμε στο κεφ. 2.5.1. Αυτή την φορά όμως πρέπει να τρέξουμε το κύκλωμα για να βγάλουμε αποτελέσματα. Αυτό απαιτεί κάποιες διαδικασίες όπως επιπεδοποιηση. Η περιγραφή του τρόπου που γίνεται η επιπεδοποιηση (levelization) γίνεται παρακάτω καθώς αποτελεί μια από τις βασικές λειτουργίες του προγράμματος πάνω στο κυρίως κύκλωμα πριν την προσομοίωση. Ο τρόπος που βγάζει τις τιμές εξόδου και συμπληρώνει τον πίνακα ενέργειας του υποκυκλώματος είναι ίδιος με τον τρόπο που υπολογίζει την στατική ενέργεια που καταναλώνει το κυρίως κύκλωμα και που αναλύεται λεπτομερώς στην συνέχεια. Κυριότερη διαφορά μπορούμε να αναφέρουμε ότι είναι το γεγονός ότι η προσομοίωση στο κύκλωμα γίνεται για έναν συγκεκριμένο αριθμό με τυχαία vectors εισόδων ενώ για τα υποκυκλώματα γίνεται για όλα τα πιθανά vectors εισόδων. Αυτό βέβαια γίνεται γιατί μια πύλη έχει αριθμό εισόδων γύρω στις 2-3 ενώ σπάνια

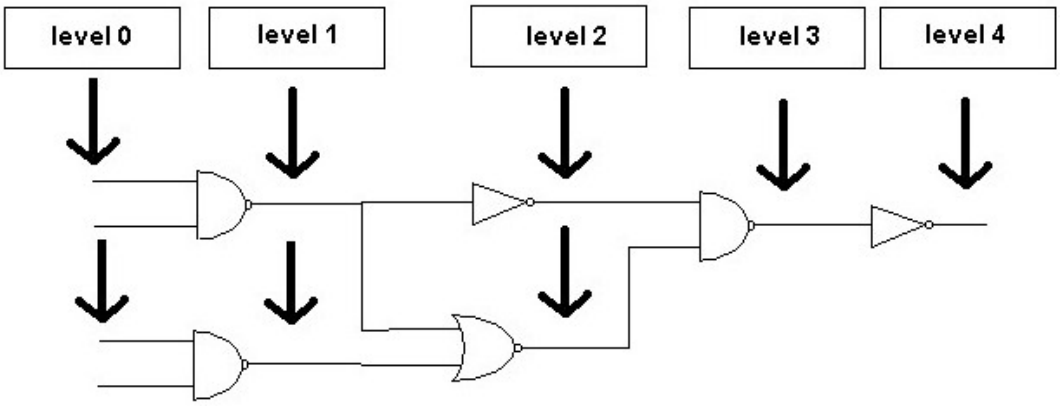
ξεπερνούν τις 9. Αντίθετα σε ένα κύκλωμα με μεγάλο αριθμό πυλών ο αριθμός των εισόδων μπορεί να είναι μερικές δεκάδες. Εφόσον λοιπόν ο αριθμός των δυνατών vectors εισόδων είναι $2^{Αριθμος Εισοδων}$ είναι πρακτικά αδύνατο να πετύχουμε τέτοιον αριθμό προσομοιώσεων στο κύκλωμα για μεγάλο αριθμό εισόδων.

2.6 Ύπαρξη flip flop D-type (DFF) στο κύκλωμα

Αναφέραμε στην αρχή ότι το πρόγραμμα δεν έχει βιβλιοθήκες και πως ό,τι χρησιμοποιείται, ορίζεται πρώτα. Αυτός ο κανόνας δεν ισχύει όμως σε μια περίπτωση. Στα ακολουθιακά κυκλώματα όπου υπάρχει παρουσία DFF υπήρξε αναγκαστικά μια εξαίρεση. Ο λόγος είναι η ιδιαιτερότητα του DFF και το γεγονός ότι πλέον η δομή του δεν ακολουθεί τη μορφή του σχήματος 2.1. Έτσι όταν υπάρχει DFF αντιμετωπίζεται ως ειδική περίπτωση. Αυτό που γίνεται είναι ότι θεωρούμε την είσοδο του DFF ως ένα σήμα που δεν οδηγεί καμία πύλη ενώ την έξοδο του την θεωρούμε ως είσοδο του κυκλώματος. Αυτή η τεχνική έχει νόημα αν σκεφτούμε πως δουλεύει ένα σύγχρονο κύκλωμα με DFF. Όταν έχουμε πυροδότηση ρολογιού τα σήματα 'ξεκινάνε' την διαδρομή τους στο κύκλωμα από τις εισόδους του κυκλώματος. Αντίστοιχα όμως το ίδιο γίνεται και με την έξοδο του DFF όπου το σήμα δεν εξαρτάται πλέον από το τι γίνεται πίσω του. Για να λειτουργήσει όμως σωστά αυτή η τεχνική πρέπει σε κάθε κύκλο η τιμή της εισόδου, που είναι ουσιαστικά έξοδος του DFF, να έχει την τιμή που είχε η είσοδος του στον προηγούμενο κύκλο. Το DFF το αναπαριστάμε σαν ένα σύνδεσμο μεταξύ του κόμβου εξόδου του κυκλώματος (που στην ουσία είναι είσοδος του DFF) και του κόμβου εισόδου του κυκλώματος (που στην ουσία είναι η έξοδος του DFF). Ο σύνδεσμος εξασφαλίζει ότι σε κάθε κύκλο η έξοδος του DFF θα έχει την τιμή που είχε η είσοδος του στον αμέσως προηγούμενο κύκλο.

2.7 Επιπεδοποίηση (levelization)

Η προσομοίωση του κυκλώματος απαιτεί να έχει προηγηθεί ο διαχωρισμός του σε επίπεδα. Κάθε επίπεδο αντιστοιχεί σε ένα σύνολο από πύλες και εκφράζει τον μέγιστο αριθμό κόμβων μεταξύ των εισόδων και των συγκεκριμένων πυλών. Αυτός ο διαχωρισμός είναι απαραίτητος γιατί πάνω του βασίζεται όλη η διαδικασία της προσομοίωσης. Με τον διαχωρισμό σε επίπεδα το κύκλωμα μας αποκτά τη μορφή που φαίνεται στο σχήμα 2.7.1



Σχήμα 2.7.1
Επιπεδοποίηση στο κύκλωμα

Για να επιτευχθεί η επιπεδοποίηση του κυκλώματος χρησιμοποιείται ένας αναδρομικός αλγόριθμος ο οποίος τρέχει τον γράφο-κύκλωμα ακολουθώντας, για κάθε είσοδο, όλα τα πιθανά μονοπάτια. Κάθε φορά που περνά από έναν κόμβο αυξάνει και ένας μετρητής ο οποίος αντιπροσωπεύει και την απόσταση του συγκεκριμένου κόμβο από την είσοδο. Ο συγκεκριμένος αλγόριθμος περιγράφεται παρακάτω.

αλγόριθμος 3 (Επιπεδοποίηση)

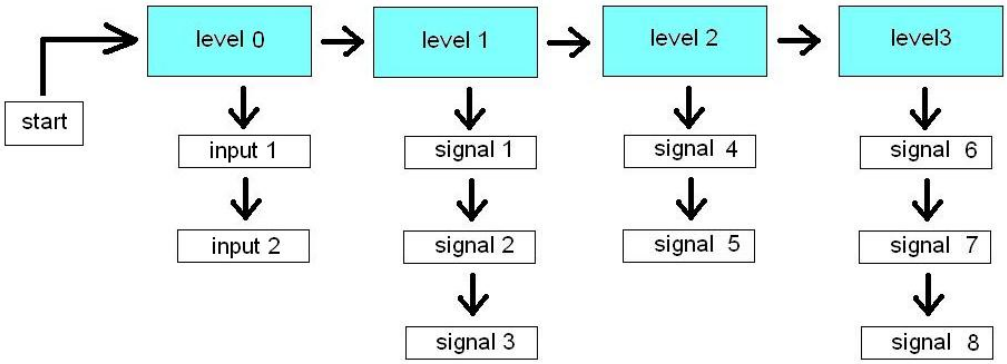
```
1:      for(each n ∈ input)
2:          levelization(n,0)

        αλγόριθμος 3.1
1:          levelization(n,counter)
2:          {
3:              if(MPASS(n)=not pass)
4:              if(counter > n → level OR n → level = 0)
5:              {
6:                  MPASS(n)=pass
7:                  n → level = counter
8:
9:          /*Για κάθε output πύλης οδηγούμενης από το n*/
10:                 for(each ni ∈ OUTPUT(NSUBC_IN(n))
11:                     levelization(ni,counter+1)
12:
```

13: MPASS(n)=not pass
14: }
15: }

Ο παραπάνω αλγόριθμος αναλύει τα κύκλωμα μας ακολουθώντας όλα τα πιθανά μονοπάτια. Η συνθήκη στη γραμμή 4 ανακόπτει μια διαδρομή όταν η μεταβλητή level του κόμβου, που εκφράζει το επίπεδο στο οποίο ανήκει ο κόμβος, δεν είναι μικρότερη από την μεταβλητή counter του αλγορίθμου. Αυτό γιατί ο counter μετράει των αριθμό τον κόμβων που έχει διανύσει από μια είσοδο έως και εκείνο το σημείο. Το level όμως στο οποίο ανήκει ένας κόμβος εκφράζει τον μέγιστο αριθμό κόμβων ανάμεσα σε είσοδο και σε αυτόν. Επομένως η μεταβλητή level μπορεί μόνο να αυξάνει.

Κατά την επιπεδοποιηση ενός κυκλώματος ομαδοποιούμε τους κόμβους σε επίπεδα. Για να κρατήσουμε τις επιπλέον πληροφορίες στον γράφο πρέπει να χρησιμοποιήσουμε κάποια επιπλέον στοιχεία. Έτσι τα επίπεδα αναπαριστάνονται από μια λίστα της οποίας κάθε στοιχείο είναι μια νέα λίστα με όλους τους κόμβους που ανήκουν στο συγκεκριμένο level. Η μορφή που έχει αυτή η λίστα διευκολύνει την διάσχιση ανά επίπεδα που κάνουμε. Η λίστα με τα levels μάλιστα είναι διπλά συνδεδεμένη επιτρέποντας και διασχίσεις προς τα πίσω του κυκλώματος, ιδιότητα που χρησιμεύει στην προσομοίωση για εύρεση δυναμικής ισχύς.



Σχήμα 2.7.2
Αντιστοίχιση επιπεδοποίησης σε δομές δεδομένων

2.7.1 Επιπεδοποίηση ακολουθιακών κυκλωμάτων

Το πρόγραμμα υποστηρίζει τόσο συνδυαστικά όσο και ακολουθιακά κυκλώματα. Ωστόσο όταν έχουμε να κάνουμε με ακολουθιακά κυκλώματα έχουμε και κάποιες επιπλέον διαδικασίες προκειμένου να γίνει σωστά το levelization. Τα ακολουθιακά κυκλώματα μπορούν να περιγράφονται με δυο τρόπους. Ο ένας είναι με

την χρήση DFF. Σε αυτή την περίπτωση η έξοδος του DFF θεωρείται είσοδος του κυκλώματος , οπότε μπαίνει στο level 0 ενώ η είσοδος του DFF θεωρείται σαν έξοδος. Αυτή η περίπτωση περιγράφεται στο κεφ.2.6 και είναι μια διαδικασία που γίνεται πριν την το levelization κατά την δημιουργία του γράφου.

2.7.2 Ανίχνευση Feedback

Ενδιαφέρον ωστόσο παρουσιάζει η δεύτερη περίπτωση. Αυτήν όπου δεν γίνεται χρήση DFF αλλά υπάρχει feedback μεταξύ των πυλών. Σε αυτήν την περίπτωση πρέπει ο αλγόριθμος 3 να μπορεί να εντοπίσει το feedback γιατί αλλιώς θα περιέλθει σε έναν ατέρμονα βρόχο. Ο τρόπος που το πετυχαίνει αυτό είναι με την χρήση της μεταβλητής **MPASS(n)**. Η MPASS(n) κάθε κόμβου μαρκάρεται με το που θα περάσει από εκεί ο αλγόριθμος. Εάν σε κάποιο σημείο υπάρχει feedback ο αλγόριθμος θα επιχειρήσει να περάσει πάλι από κόμβο που έχει ήδη περάσει και μαρκάρει. Η γραμμή 3 του αλγορίθμου 3 εξασφαλίζει ότι η διάσχιση του κυκλώματος σε αυτή την περίπτωση θα διακοπεί.

2.8 Benchmark circuits

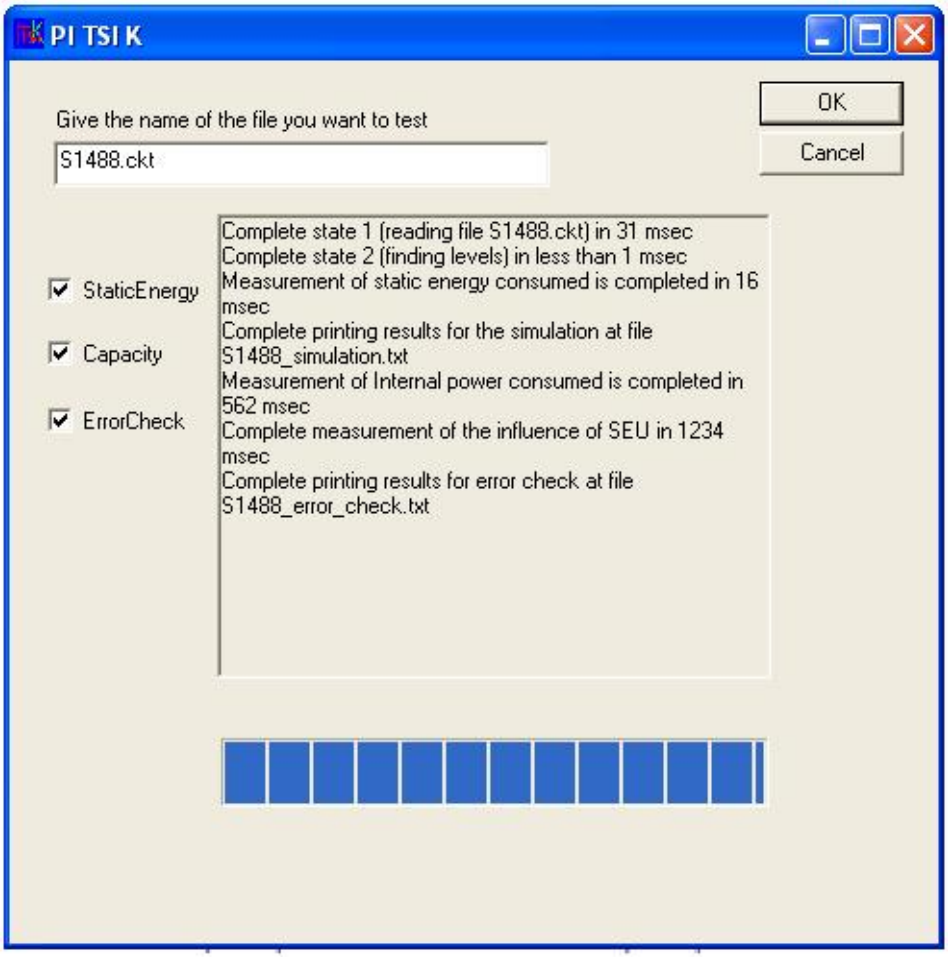
Για να ελέγξουμε την σωστή λειτουργία του προσομοιωτή χρησιμοποιήσαμε τα κυκλώματα αναφοράς (benchmark circuits) ISCAS-85 και ISCAS-89. Στον παρακάτω πίνακα παρουσιάζονται ορισμένα στοιχεία για αυτά τα κυκλώματα.

Ονομασία κυκλώματος	Αριθμός Εισόδων	Αριθμός Εξόδων	D-type flipflops	inverters	ANDs	NANDs	ORs	NORs
C17	5	2	0	0	0	6	0	0
C432	36	7	0	40	4	76	0	19
C499	41	32	0	40	56	0	2	0
C880	60	26	0	63	117	86	29	61
C1908	33	25	0	277	63	377	0	0
C2670	233	140	0	321	333	254	77	12
C6288	32	32	0	32	256	0	0	2128
C5315	178	123	0	581	718	454	214	27
C7552	207	108	0	876	776	1028	244	54
S27	4	1	3	2	1	1	2	4
S208_1	10	1	8	38	21	15	14	16
S208	10	1	8	38	21	15	14	16
S298	3	6	14	44	31	9	16	19
S386	7	7	6	41	83	0	35	0
S382	3	6	21	59	11	30	24	34
S344	9	11	15	59	44	18	9	30
S349	9	11	15	57	44	19	10	31
S400	3	6	21	58	11	36	25	34
S444	3	6	21	62	13	58	14	34

S526	3	6	21	52	56	22	28	35
S526N	3	6	21	54	55	22	28	35
S420	19	2	16	74	28	46	9	39
S510	19	7	6	32	34	61	29	55
S420_1	18	1	16	78	49	29	28	34
S832	28	19	5	25	78	54	64	66
S820	18	19	5	33	76	54	60	66
S641	35	24	19	272	90	4	13	0
S713	35	23	19	254	94	28	17	0
S953	16	23	29	84	49	114	36	112
S838_1	34	1	32	158	105	57	56	70
S838	34	1	32	158	105	57	56	70
S1238	14	14	18	80	134	125	112	57
S1196	14	14	18	141	118	119	101	50
S1494	8	19	6	89	354	0	204	0
S1488	8	19	6	103	350	0	200	0
S1423	17	5	74	167	197	64	137	92
S5378	35	49	179	1775	0	0	239	765
S9234	19	22	228	3570	955	528	431	113
S9234_1	36	39	211	3570	955	528	431	113
S13207	31	121	669	5378	1114	849	512	98
S13207_1	32	152	638	5378	1114	849	512	98
S15850	14	87	597	6324	1619	968	710	151
S15850_1	77	150	534	6324	1619	968	710	151
S35932	35	320	1728	3861	4032	7020	1152	0
S38584	12	278	1452	7805	5516	2126	2621	1185
S38584_1	38	304	1426	7805	5516	2126	2621	1185
S38417	28	106	1636	13470	4154	2050	226	2279

2.9 Απαιτήσεις σε μνήμη και χρόνο

Το πρόγραμμα ελέγχθηκε για την αποτελεσματικότητα του και την ορθότητα των αποτελεσμάτων που βγάζει πάνω στα Benchmark Circuits του προηγούμενου πίνακα. Για την ευκολία στην χρήση του προγράμματος δημιουργήσαμε το γραφικό περιβάλλον του σχήματος 2.8



Σχήμα 2.8
Το interface του προγράμματος

Οι απαιτήσεις σε μνήμη είναι περίπου 2.7 MB για τα γραφικά και από εκεί και πέρα αυξάνει ανάλογα με τις απαιτήσεις της προσομοίωσης. Για να τρέξει η προσομοίωση χρειάζεται να γίνουν κάποιες προεργασίες όπως τις αναφέραμε στις προηγούμενες παραγράφους. Εργασίες όπως η δημιουργία και ανάλυση των sub circuits, δημιουργία των lookup tables έχουν απαιτήσεις τόσο σε χρόνο όσο και σε μνήμη. Παρακάτω παρουσιάζεται ένας πίνακας με τις απαιτήσεις σε μνήμη και τον χρόνο που θέλει το πρόγραμμα για να τρέξει. Ο χρόνος αυτός αντιστοιχεί στο χρόνο που θέλει για να διαβάσει το αρχείο, να κατασκευάσει και να αναλύσει όλα τα υποκυκλώματα και να σχηματίσει στην μνήμη τον γράφο του κυκλώματος. Επίσης παρουσιάζεται και ο χρόνος που απαιτεί το levelization. Οι χρόνοι είναι σε msec.

Κύκλωμα	Μνήμη Kbytes	Χρόνος δημιουργίας γράφου(msec)	Χρόνος levelization (msec)
C17	3.240	31	<1
C432	3.256	16	<1
C499	3.228	31	<1
C880	3.428	31	<1
C1908	3.796	62	<1
C2670	4.192	140	<1
C5315	5.924	922	<1
C6288	5.968	1015	<1
C7552	6.044	2781	<1
S27	2.992	15	<1
S208	3.040	15	<1
S208_1	3.036	16	<1
S298	3.080	16	<1
S344	3.120	15	<1
S349	3.112	16	<1
S382	3.120	16	<1
S386	3.120	16	<1
S400	3.112	15	<1
S420	3.144	16	<1
S420_1	3.148	16	<1
S444	3.104	15	<1
S520	3.144	15	<1
S526	3.144	16	<1
S526N	3.160	15	<1
S641	3.260	15	<1
S713	3.276	15	<1
S820	3.228	31	<1
S832	3.240	47	<1
S838	3.304	15	<1
S838_1	3.280	62	<1
S953	3.236	62	<1
S1196	3.364	62	<1
S1238	3.348	47	<1
S1423	3.448	62	<1
S1488	3.468	140	<1
S1494	3.450	47	<1
S5378	5.280	1406	<1
S9234	7.512	7140	16
S9234_1	7.528	6828	63
S13207	9.752	15906	31

S15850	10.996	22843	32
S15850_1	11.032	21937	94
35932	17.460	80672	468
S38417	21.776	133719	281
S38584	19.820	104203	297
S38584_1	19.828	100953	750

Στα μικρά κυκλώματα ο χρόνος δημιουργίας του γράφου είναι εξαιρετικά μικρός και εξαρτάται κυρίως από την ταχύτητα μεταφοράς των δεδομένων από τον δίσκο. Βλέπουμε ότι και ο χρόνος που απαιτείται για την δημιουργία των υπό κυκλωμάτων είναι επίσης μικρός καθώς συμπεριλαμβάνεται στον χρόνο που μετράμε. Όσο μεγαλώνει το κύκλωμα ο χρόνος αυξάνει χωρίς όμως να φτάνει σε πολύ μεγάλα επίπεδα. Οι απαιτήσεις σε μνήμη όπως παρουσιάζονται είναι κάτι λιγότερο από 3 MB για το μικρότερο κύκλωμα με τα 2,7 να χρειάζονται για το παράθυρο του σχήματος 2.8. Έχει σημασία να υπενθυμίσουμε ότι σε αυτή την μνήμη συμπεριλαμβάνονται και οι γράφοι των sub circuits , και τα lookup tables όλων των πυλών που ορίστηκαν καθώς και μνήμη για το levelization. Όσο μεγαλώνει το κύκλωμα αυξάνει και η μνήμη. Βλέπουμε ότι στα μεγάλα κυκλώματα οι απαιτήσεις φτάνουν σε επίπεδα 20 MB τιμή που οφείλεται στον μεγάλο αριθμό πυλών και κόμβων. Αυτό το πλήθος όμως συνεπάγεται και επιπλέον μνήμη για συνδέσμους μεταξύ πυλών και κόμβων , για το levelization , για την αποθήκευση των αποτελεσμάτων όταν, όπως στην περίπτωση SEU , μετράμε τιμή για κάθε κόμβο.

Το σύστημα το οποίο χρησιμοποιήσαμε είναι ένας Pentium 4 3.05G με 512 μνήμη στα 400Mh και το λειτουργικό είναι Windows XP.

Τεχνικές Προσομοίωσης

Το πρόγραμμα πραγματοποιεί προσομοίωση και εξαγωγή δεδομένων στις εξής κατηγορίες:

Κατανάλωση στατικής ισχύος.
Κατανάλωση δυναμικής ισχύος.
Συμπεριφορά κυκλώματος σε single event upset(SEU)
Η προσομοίωση που γίνεται είναι level-based και event-driven
και γίνεται σε λογικό επίπεδο(switch level simulation).

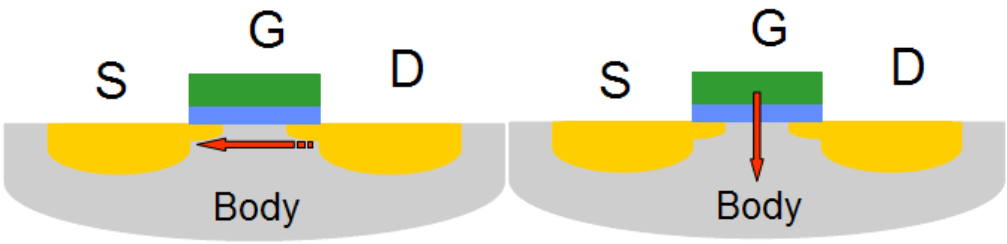
Στα επόμενα κεφάλαια αναλύεται λεπτομερώς ο τρόπος λειτουργίας της κάθε μεθόδου.

Κεφάλαιο 3^{ο}

ΣΤΑΤΙΚΗ ΙΣΧΥΣ

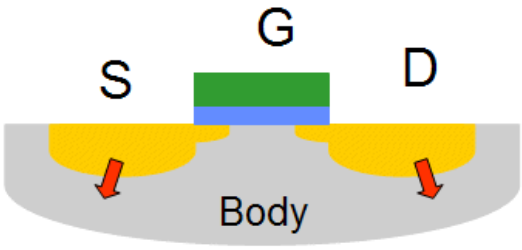
3.1 Ορισμός Στατικής Ισχύος

Στατική ισχύς είναι η ισχύς που καταναλώνει μια πύλη όταν δεν αλλάζει την λογική τιμή της εξόδου της. Στατική ισχύς καταναλώνεται για πολλούς λόγους. Το μεγαλύτερο ποσοστό κατανάλωσης στατικής ισχύος στις τεχνολογίες που αναλύθηκαν, όπως φαίνεται και στο σχήμα 3.1 προκύπτει από την αγωγή υπό κατωφλίου μεταξύ πηγής και υποδοχής (source-to-drain sub threshold leakage) και το ρεύμα διαρροής πύλης καναλιού που οφείλεται σε φαινόμενα διόδευσης από το οξείδιο(tunneling effects).



Σχήμα 3.1
Αγωγή υπό κατωφλίου τρανζίστορ και αγωγή μεταξύ πύλης και καναλιού.

Κατανάλωση στατικής ισχύος, βλέπε σχήμα 3.2 παρουσιάζεται επειδή άγει η ανάστροφα πολωμένη δίοδος μεταξύ των στρωμάτων διαχύσεως και του υποστρώματος. Για το λόγο αυτό, η στατική ισχύς συχνά αναφέρεται ως ισχύς διαρροής (leakage power).



Σχήμα 3.2
Αγωγή της ανάστροφα πολωμένης διόδου που σχηματίζεται από την περιοχή διαχύσεως και το υπόβαθρο.

3.2 Υπολογισμός Ισχύος Διαρροής (Leakage Power Calculation).

Όταν ζητείται από έναν Power Compiler να δώσει μια ανάλυση κατανάλωσης ισχύος για ένα κύκλωμα, υπολογίζει την συνολική ισχύ διαρροής προσθέτοντας την ισχύ διαρροής κάθε τεχνολογικού κελιού που έχει χρησιμοποιηθεί στο συγκεκριμένο κύκλωμα, όπως φαίνεται και στους παρακάτω μαθηματικούς τύπους:

$$P_{LeakageTotal} = \sum_{\forall cells(i)} P_{CellLeakagei}$$

όπου:

$P_{LeakageTotal}$: Συνολική κατανάλωση ισχύος διαρροής για το κύκλωμα.

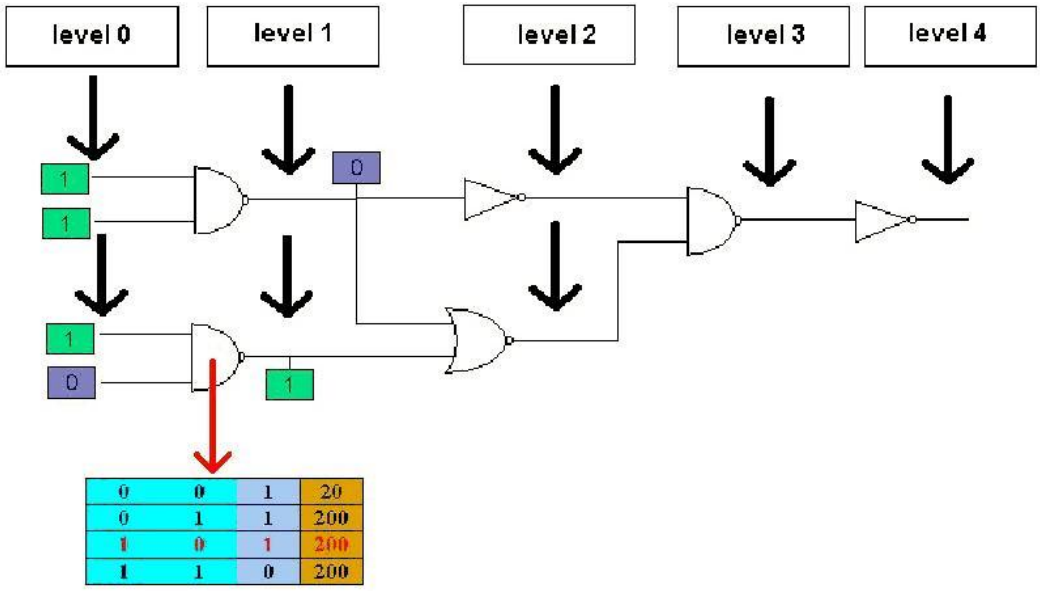
$P_{CellLeakage\_i}$: Κατανάλωση ισχύος διαρροής για το κάθε κελί i.

Οι μηχανικοί που αναπτύσσουν κελιά για βιβλιοθήκες (library cells) επισυνάπτουν μέσα στην περιγραφή του μοντέλου τους και την κατά προσέγγιση συνολική τιμή της ισχύος διαρροής που καταναλώνει το κάθε κελί της βιβλιοθήκης.
Η ισχύς διαρροής πολλές φορές εξαρτάται και από την λογική κατάσταση που βρίσκεται το κελί (state-dependent leakage power). Η τιμή αυτή μπορεί να μοντελοποιηθεί κατά την δημιουργία της βιβλιοθήκης προσαρτώντας την νέα πληροφορία με ένα when statement στην περιγραφή του LEAKAGE_POWER σε κάθε κελί.
Σύμφωνα με μετρήσεις για κυκλώματα που είναι ενεργά τον περισσότερο χρόνο της λειτουργίας τους, η ισχύς διαρροής είναι μικρότερη του 1% της συνολικής κατανάλωσης. Ωστόσο, για κυκλώματα που συνήθως είναι ανενεργά, η μοντελοποίηση της ισχύος διαρροής είναι σημαντική. Τελευταία όμως η κατανάλωση στατικής ισχύος αποκτά ιδιαίτερη σημασία στον σχεδιασμό κυκλωμάτων. Αυτό γιατί καθώς προχωρά η τεχνολογία αυξάνει σημαντικά ο αριθμός των φορητών συσκευών και γενικά συστημάτων τα οποία έχουν έναν ιδιαίτερο συνδυασμό. Μένουν σε κατάσταση αναμονής για πάρα πολύ ώρα ενώ την ίδια στιγμή η τροφοδοσία τους βασίζεται σε μπαταρίες. Γίνεται λοιπόν φανερό πως το πρόβλημα που προκύπτει σε αυτού του είδους τις συσκευές είναι η όσο το δυνατόν μείωση της κατανάλωσης ισχύος, η οποία όμως ισχύ είναι σε μεγάλο βαθμό ισχύς διαρροής.

3.3 Υπολογισμός Στατικής ισχύς μέσω προσομοίωσης.

Ο υπολογισμός της κατανάλωσης ισχύος γίνεται με την χρήση του πίνακα κατανάλωσης κάθε πύλης. Δίνοντας αρχικές, τυχαίες τιμές στα σήματα εισόδου και κάνοντας μια διάσχιση του κυκλώματος αθροίζουμε τις καταναλώσεις σε ένα κοινό άθροισμα. Θεωρούμε ότι οι είσοδοι του κυκλώματος μας βρίσκονται στο πρώτο level, το level 0. Με μια γεννήτρια τυχαίων τιμών αρχικοποιούμε τις εισόδους σε κατάσταση 0 ή 1 . Ξεκινώντας από το level 1 προσδιορίζουμε την κατάσταση όλων

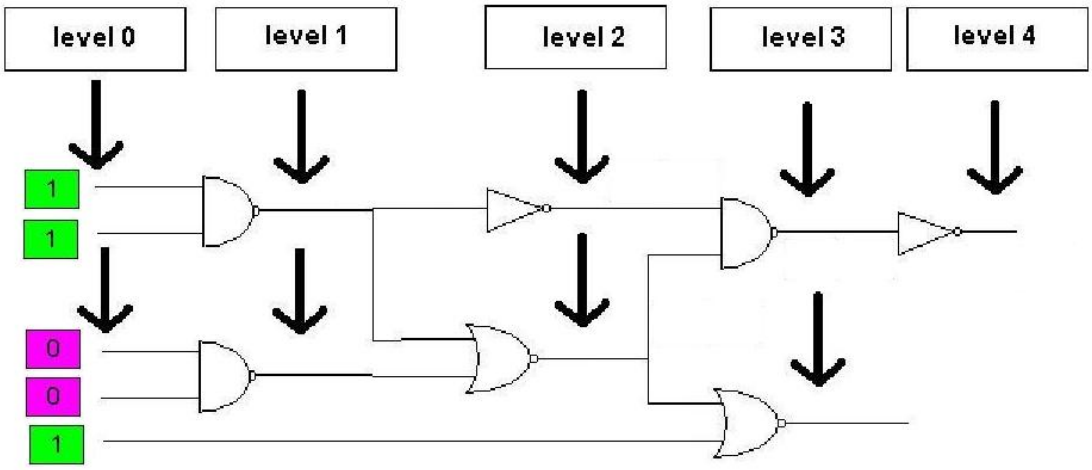
των κόμβων που ανήκουν σε αυτό. Στην συνέχεια επαναλαμβάνουμε για το level 2 και για κάθε level μέχρι το τέλος. Ο προσδιορισμός της κατάστασης ενός κόμβου γίνεται κοιτώντας την πύλη η οποία οδηγεί αυτόν τον κόμβο. Κάθε πύλη φέρει από πριν έναν πίνακα αληθείας καθώς και τον πίνακα με τις καταναλώσεις στατικής ενέργειας. Κοιτώντας τους κόμβους που αποτελούν τις εισόδους της συγκεκριμένης πύλης οδηγούμαστε από τον πίνακα αληθείας στην κατάσταση που πρέπει να δώσουμε στον υπό εξέταση κόμβο. Από τον ίδιο πίνακα παίρνουμε και το ποσό στατικής ισχύος που καταναλώνεται σε αυτό τον κόμβο. Έτσι έχουμε γρήγορα τα αποτελέσματα που θέλουμε.



Σχήμα 3.3
Χρήση του lookup table και πίνακα ενέργειας στον γράφο

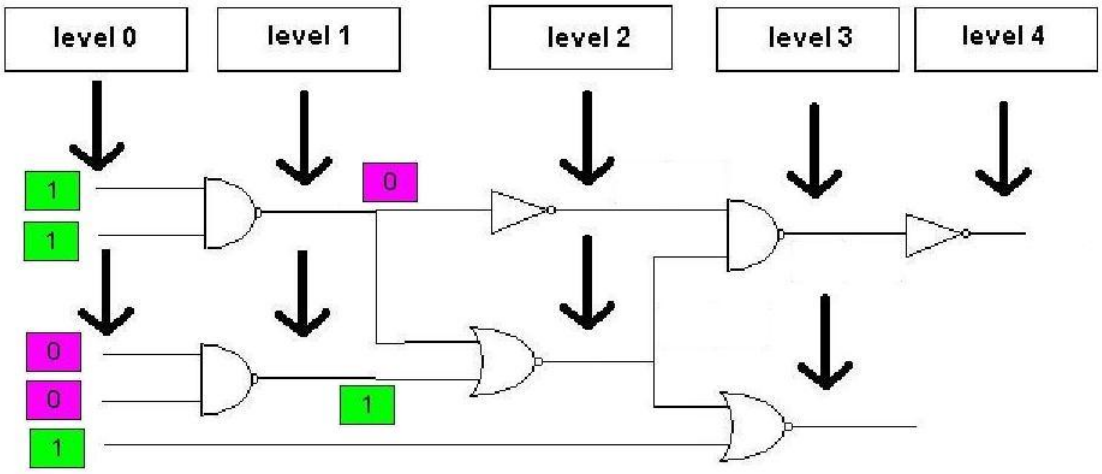
Κατά την διάσχιση του κυκλώματος εξάγουμε σε κάθε κόμβο την κατάσταση του καθώς και το ποσό στατικής ενέργειας που καταναλώνεται σε αυτόν. Για να έχουμε μια εικόνα για την συνολική κατανάλωση στατικής ισχύος αυτό που κάνουμε είναι να αθροίζουμε όλες τις καταναλώσεις σε ένα κοινό σύνολο. Μόλις τελειώσουμε και με το τελευταίο level έχουμε μια καλή προσέγγιση του ποσού στατικής ενέργειας που καταναλώνει το κύκλωμα για αυτό το συγκεκριμένο vector εισόδου. Για να έχουμε μια καλή εκτίμηση του ποσού στατικής ενέργειας που καταναλώνει και του πως είναι κατανεμημένη επαναλαμβάνουμε την προσομοίωση δίνοντας νέες τιμές εισόδου. Οι τιμές εισόδου επιλέγονται τυχαία προσπαθώντας έτσι να καλύψουμε όσο το δυνατόν περισσότερες καταστάσεις από αυτές που μπορεί να επέλθει το κύκλωμα.

Μπορούμε δείξουμε σχηματικά ένα παράδειγμα για το πως δουλεύει η level based προσομοίωση. Έχοντας κάνει το levelization έχουμε το κύκλωμα στην μνήμη αρχικά όπως στο σχήμα 3.4.1



Σχήμα 3.4.1
Βήματα προσομοίωσης υπολογισμού στατικής ενέργειας

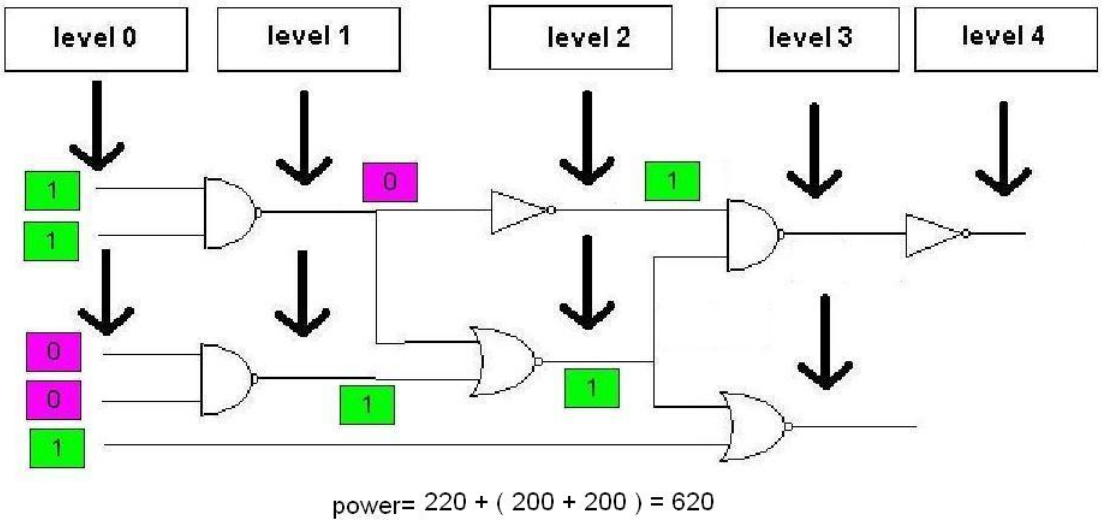
Οι κόμβοι που βρίσκονται στο level 0 αποτελούν τις εισόδους του κυκλώματος. Εμείς τώρα θέλουμε να βρούμε την κατάσταση που θα πρέπει να πάρουν και οι υπόλοιποι κόμβοι για να σταθεροποιηθεί το κύκλωμα. Ξεκινάμε από το πρώτο level. Για κάθε κόμβο που ανήκει σε αυτό το level κοιτάμε την πύλη που το οδηγεί. Σε κάθε μια από αυτές τις πύλες γνωρίζουμε ότι όλοι οι κόμβοι που την οδηγούν έχουν κατασταλάξει σε μια κατάσταση. Αυτός ο κανόνας ισχύει για κάθε level και ο λόγος είναι ότι εφόσον κοιτάμε τα levels από το πρώτο με την σειρά όλοι οι κόμβοι που προηγούνται του ,υπό εξέταση κόμβου, έχουν σταθεροποιηθεί σε μια κατάσταση. Έτσι λοιπόν κοιτάμε πρώτα το level 1 και φτιάχνουμε τους κόμβους που ανήκουν σε αυτό. Για κάθε κόμβο προσθέτουμε και το πόσο ενέργειας που αντιστοιχεί στον πίνακα της πύλης που τον οδηγεί.



Σχήμα 3.4.2

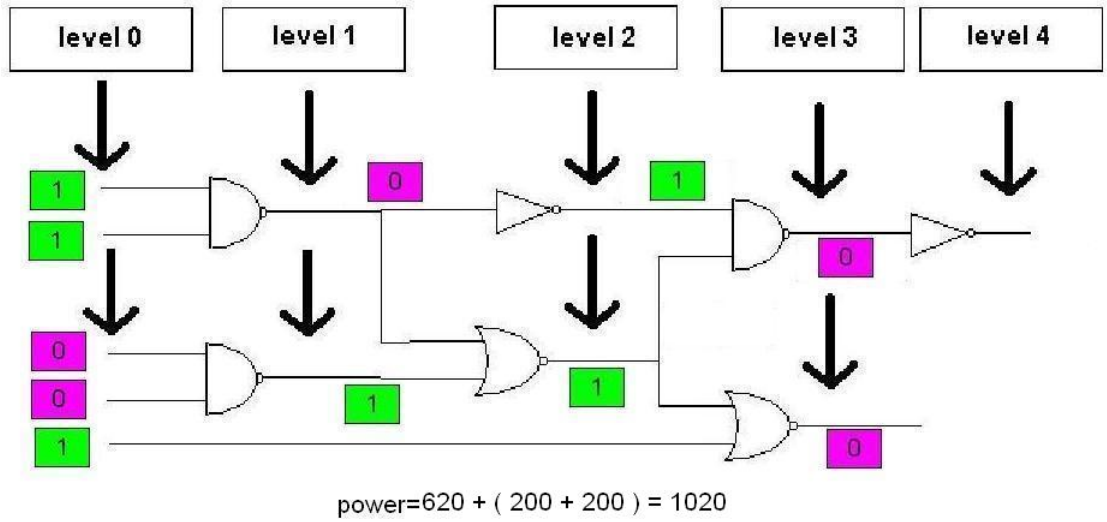
Η μέχρι τώρα ενέργεια που έχουμε μετρήσει είναι 200 από την πρώτη NAND καθώς τόση δίνει ο πίνακας για είσοδο 0 και 0 καθώς και 20 από την δεύτερη NAND. Στην συνέχεια για τα levels 2,3,4 με αυτή την σειρά η προσομοίωση θα πάει ως εξής:

Για το level 2



Σχήμα 3.4.3

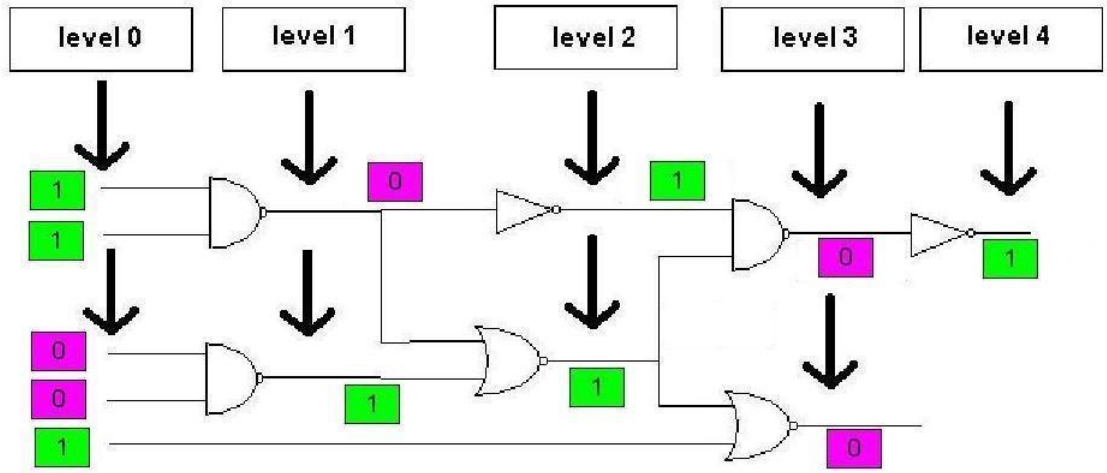
Για το level 3



power=620 + (200 + 200) = 1020

Σχήμα 3.4.4

και τέλος για το level 4 που είναι και το τελευταίο



power= 1020 + 200 = 1220

Σχήμα 3.4.5

Τελειώνοντας με το τελευταίο level όλοι οι κόμβοι έχουν κατασταλάξει σε μια κατάσταση με κύκλωμα να βρίσκεται πλέον σε κατάσταση ηρεμίας ενώ υπολογίσαμε και το πόσο ενέργειας μέσω της τιμής του ρεύματος διαρροής.

3.4 Απαιτήσεις σε χρόνο

Το πρόγραμμα δοκιμάστηκε για την αποτελεσματικότητα του σε αυτήν την προσομοίωση καθώς και για την ορθότητα των αποτελεσμάτων του στα benchmark circuits που παρουσιάσαμε στο προηγούμενο κεφάλαιο. Στον παρακάτω πίνακα παρουσιάζονται οι χρόνοι που χρειάστηκε το σύστημα μας καθώς και το πλήθος των προσομοιώσεων σε κάθε κύκλωμα. Για να κυμαίνονται οι χρόνοι σε ίδια επίπεδα όσο

μεγαλώνει το κύκλωμα μειώνουμε τον αριθμό των προσομοιώσεων. Έτσι έχουμε το παρακάτω πίνακα αποτελεσμάτων. Οι χρόνοι είναι σε msec.

Κύκλωμα	Χρόνος Στατικής	No of tests
C17	281	10000
C432	422	10000
C499	438	10000
C880	672	10000
C1908	3625	10000
C2670	6500	10000
C5315	1266	1000
C6288	1266	1000
C7552	203	100
S27	250	10000
S208	328	10000
S208_1	343	10000
S298	375	10000
S344	375	10000
S349	391	10000
S382	406	10000
S386	406	10000
S400	407	10000
S420	438	10000
S420_1	454	10000
S444	422	10000
S520	453	10000
S526	469	10000
S526N	469	10000
S641	563	10000
S713	578	10000
S820	641	10000
S832	656	10000
S838	828	10000
S838_1	766	10000
S953	703	10000
S1196	1063	10000
S1238	1125	10000
S1423	1735	10000
S1488	1985	10000
S1494	1813	10000
S5378	1422	1000
S9234	265	100
S9234_1	265	100

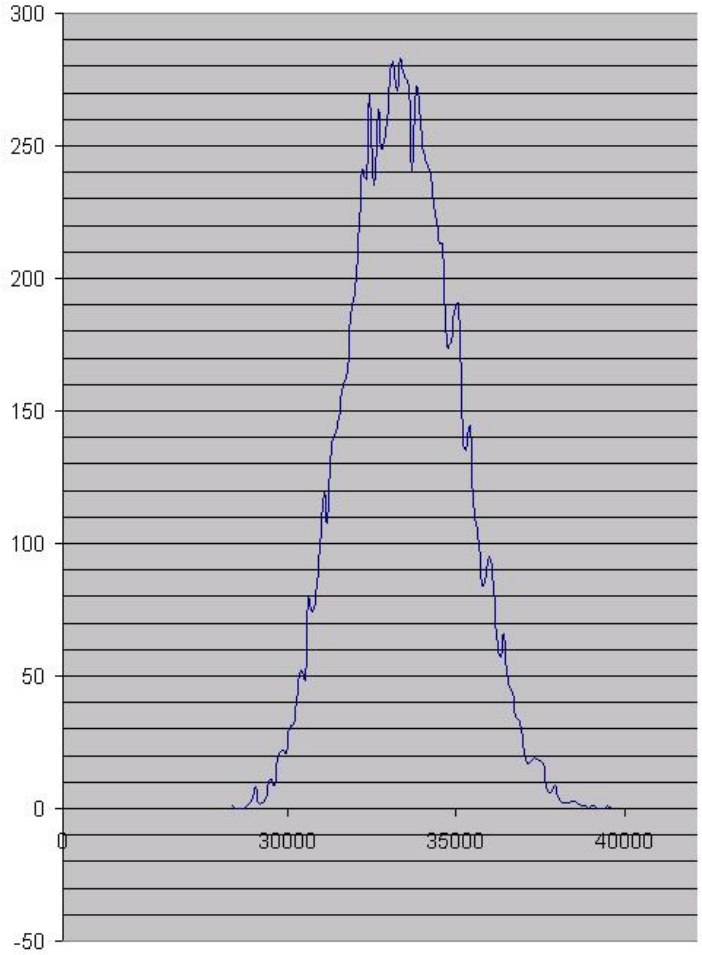
S13207	391	100
S15850	468	100
S15850_1	484	100
35932	907	100
S38417	1156	100
S38584	1078	100
S38584_1	1125	100

Η μορφή των αποτελεσμάτων αυτού του είδους της προσομοίωσης καθώς και τα αποτελέσματα που βγήκαν κατά την προσομοίωση ορισμένων κυκλωμάτων αναφοράς (benchmark circuits) παρουσιάζονται στην συνέχεια..

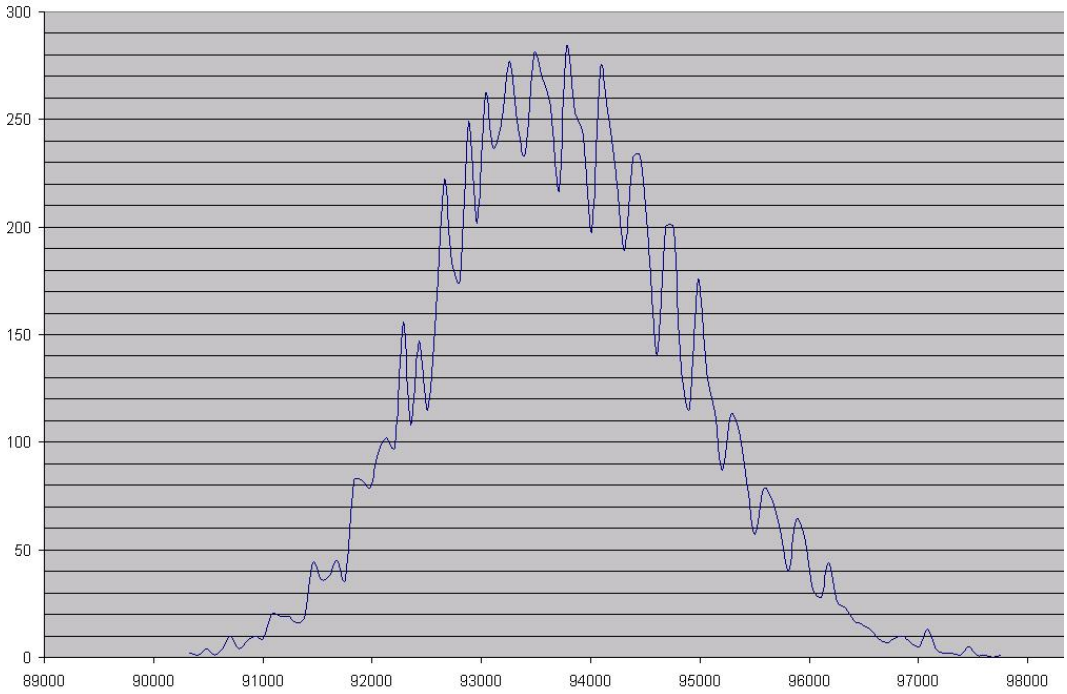
3.5 Αποτελέσματα

Παρακάτω παρουσιάζονται τα αποτελέσματα των προσομοιώσεων. Στον άξονα x έχουμε τα ποσά ενέργειας (ρεύμα διαρροής) σε μΑ και στον y η συχνότητα εμφάνισης.

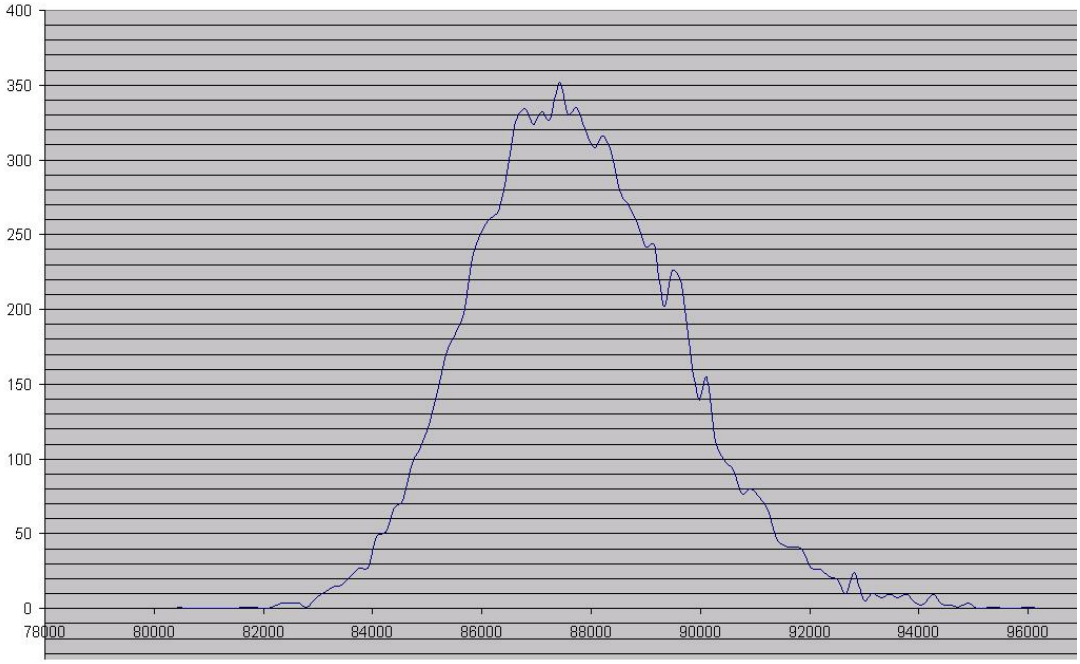
C432



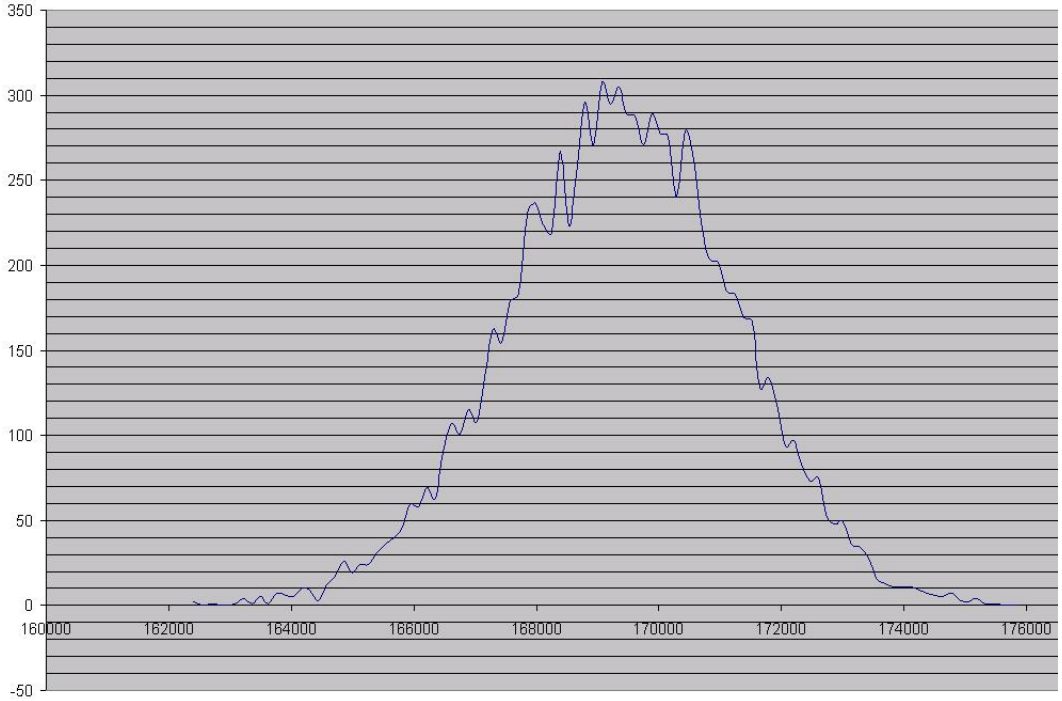
c499


300
250
200
150
100
50
0
89000
90000
91000
92000
93000
94000
95000
96000
97000
98000

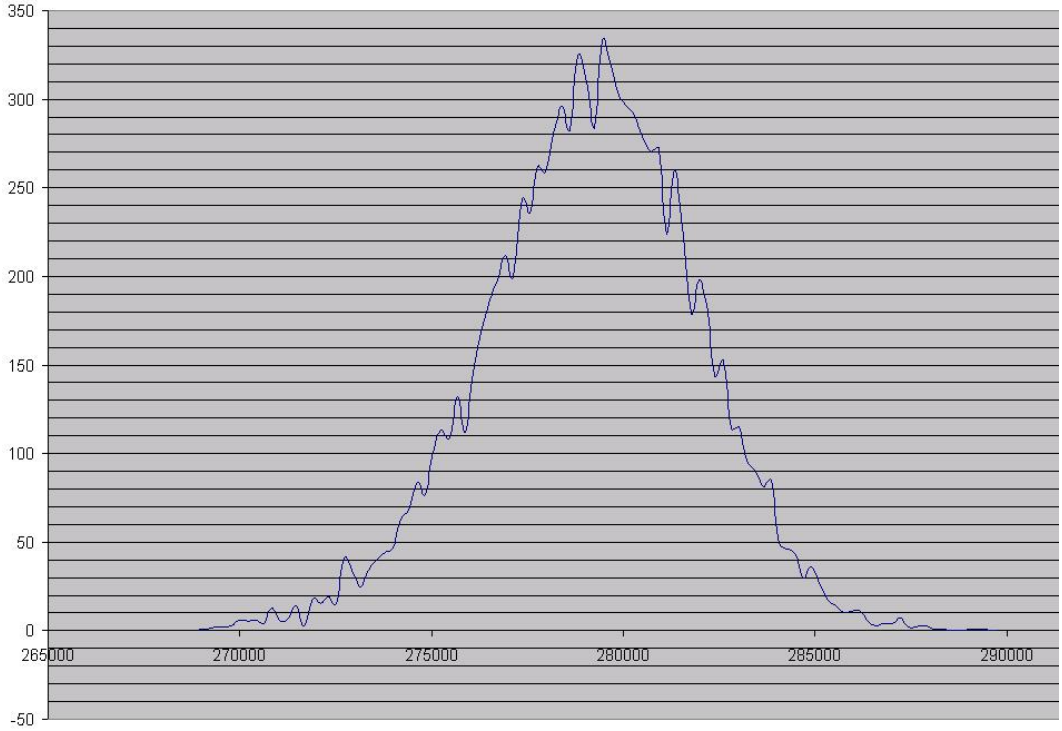
c880


400
350
300
250
200
150
100
50
0
78000
80000
82000
84000
86000
88000
90000
92000
94000
96000

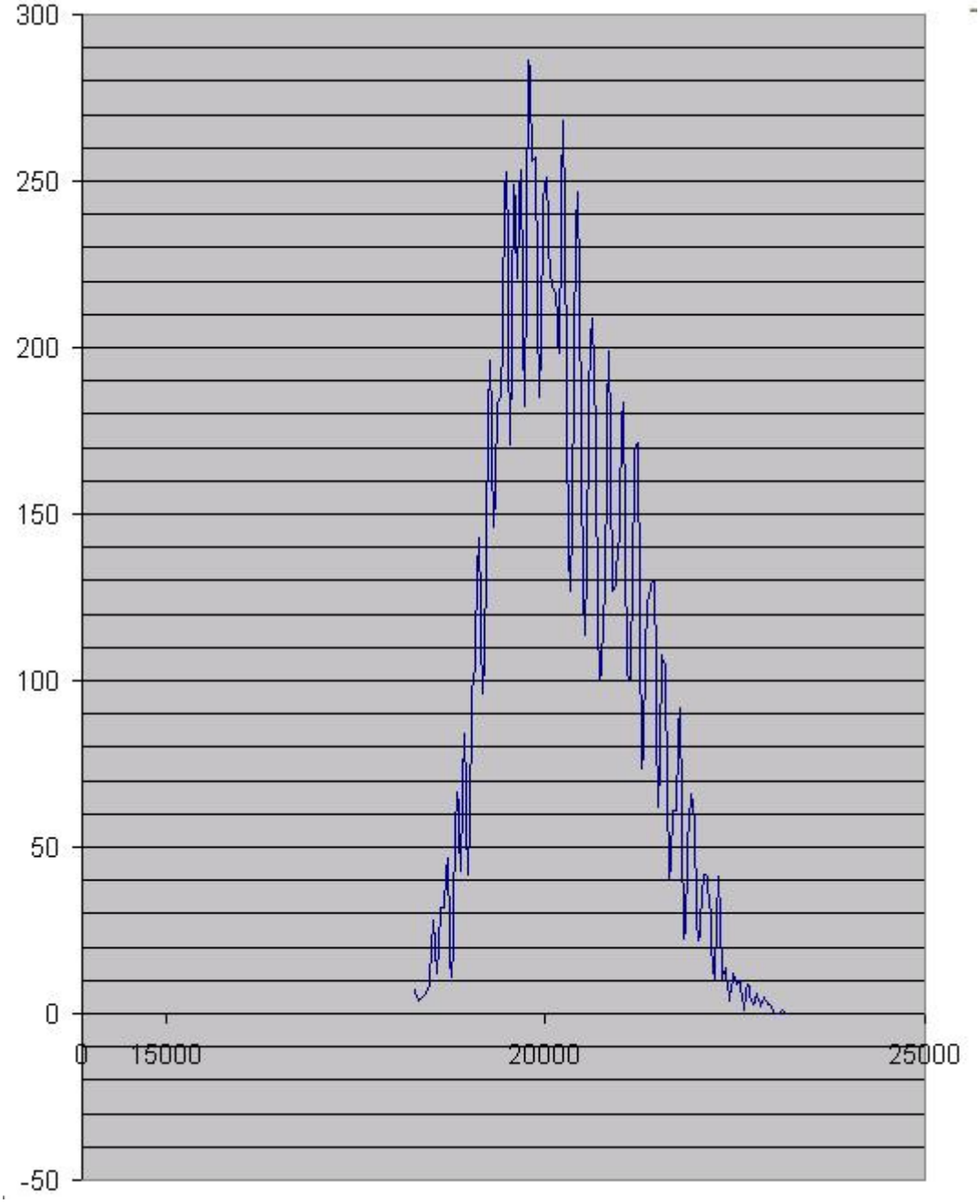
c1908


350
300
250
200
150
100
50
0
-50
160000
162000
164000
166000
168000
170000
172000
174000
176000

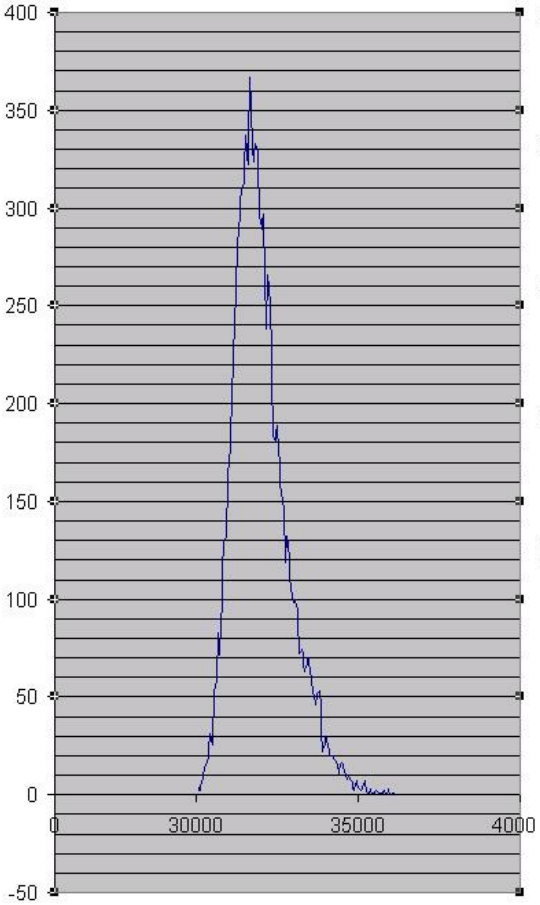
c2670


350
300
250
200
150
100
50
0
-50
265000
270000
275000
280000
285000
290000

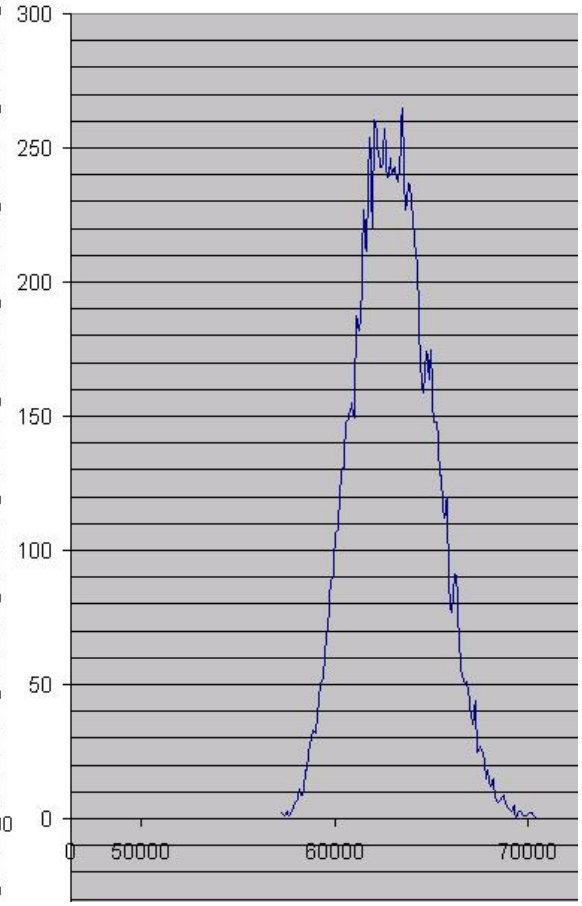
s208



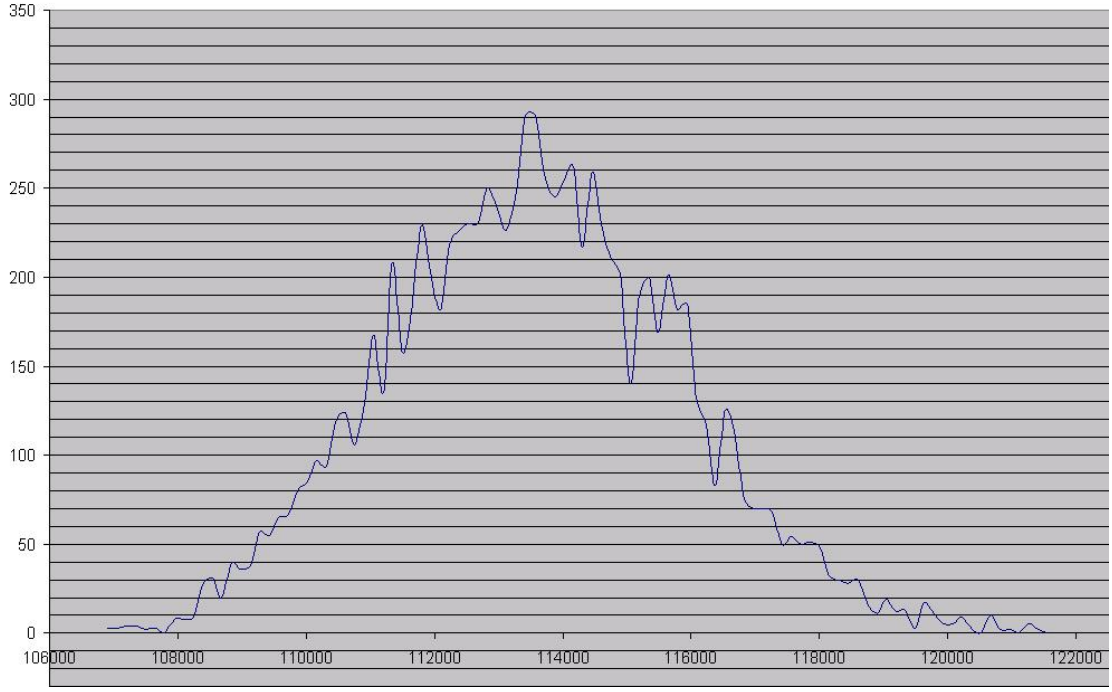
s420


400
350
300
250
200
150
100
50
0
-50
0
30000
35000
4000

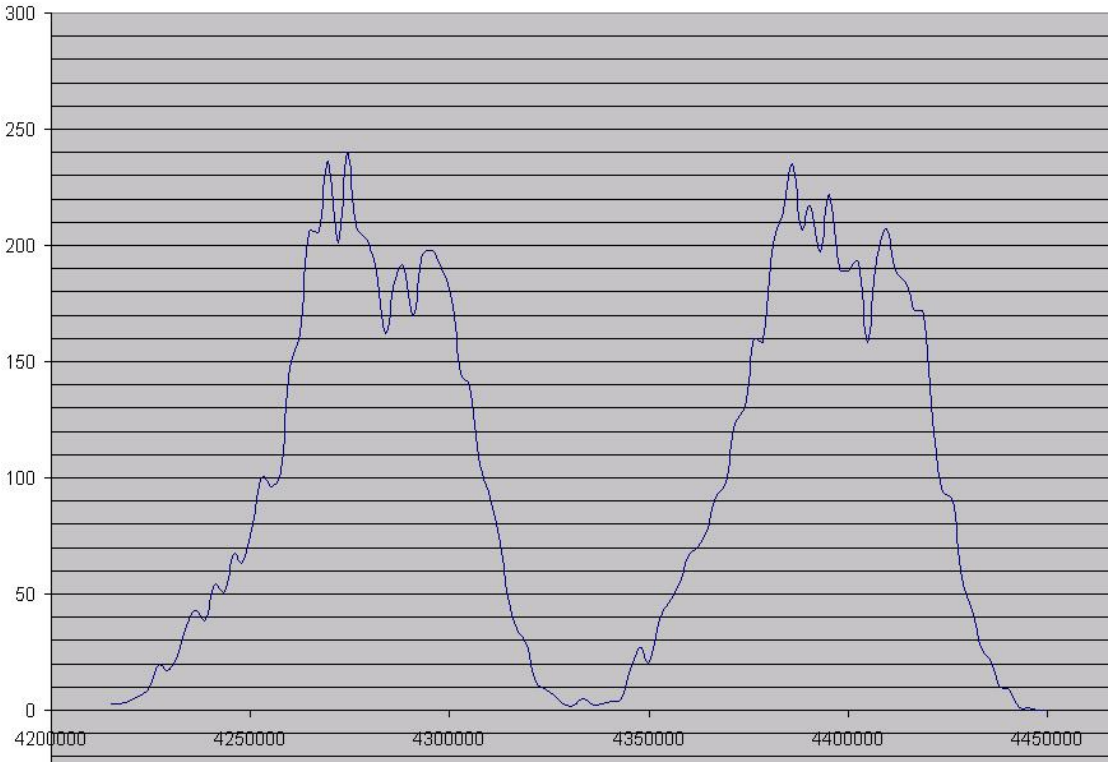
s820


300
250
200
150
100
50
0
0
50000
60000
70000

s1238



s38584



Κεφάλαιο 4ο

ΔΥΝΑΜΙΚΗ ΙΣΧΥΣ

4.1 Ορισμός Δυναμικής Ισχύος.

Δυναμική ισχύς είναι η ισχύς που καταναλώνεται όταν μια πύλη είναι ενεργή. Ένα κύκλωμα είναι ενεργό κάθε φορά που οι τάσεις των δικτύων του εναλλάσσονται σύμφωνα πάντα με τις εισόδους που εφαρμόζονται στο κύκλωμα. Επειδή λοιπόν, η τιμή της τάσης σε ένα δίκτυο εισόδου μπορεί να αλλάξει χωρίς αυτό να σημαίνει ότι θα έχουμε και μια εναλλαγή λογικής τιμής στο δίκτυο εξόδου, δυναμική κατανάλωση ισχύος μπορεί να παρατηρηθεί και σε περιπτώσεις που ένα δίκτυο εξόδου δεν αλλάξει λογική τιμή.

Η δυναμική κατανάλωση ισχύος σε ένα κύκλωμα συνθέτεται από δύο συντελεστές, οι οποίοι παρουσιάζονται παρακάτω:

- Ισχύς λόγω μεταγωγής λογικής τιμής (Switching power).
- Εσωτερική ισχύς (Internal power).

4.1.1 Ορισμός ισχύος λόγω μεταγωγής λογικής τιμής.

Η κατανάλωση ισχύος λόγω μεταγωγής λογικής τιμής ενός οδηγούμενου κελιού προκύπτει από την φόρτιση και εκφόρτιση της χωρητικότητας φορτιού στην έξοδο του κελιού. Η συνολική χωρητικότητα φορτίου στην έξοδο του οδηγούμενου κελιού είναι το άθροισμα της χωρητικότητας του δικτύου και της πύλης όπου οδηγείται η έξοδος.

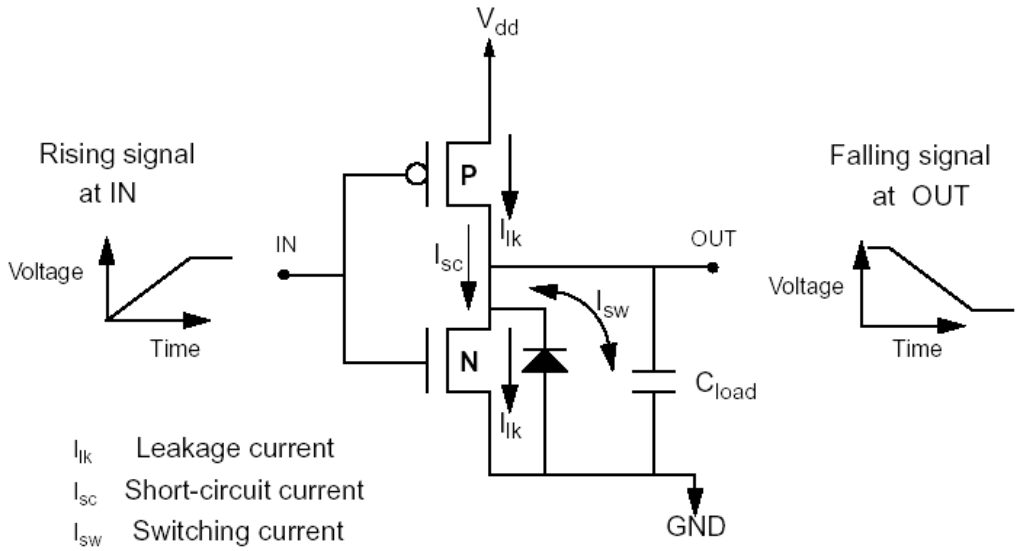
Επειδή τέτοιες φορτίσεις και εκφορτίσεις είναι αποτέλεσμα των λογικών μεταγωγών των τιμών της εξόδου του κελιού, η κατανάλωση ισχύος λόγω μεταγωγής λογικής τιμής αυξάνεται όσο και η συχνότητα των μεταγωγών λογικών τιμών αυξάνεται. Καταλήγουμε λοιπόν, ότι η ισχύς λόγω μεταγωγής λογικής τιμής είναι μια συνάρτηση δύο παραγόντων, της συνολικής χωρητικότητας φορτίου στην έξοδο του κελιού και της συχνότητας των εναλλαγών λογικών τιμών. Η κατανάλωση ισχύος λόγω εναλλαγής λογικής τιμής αποτελεί το μεγαλύτερο ποσοστό της κατανάλωσης ισχύος ενός ενεργού CMOS κυκλώματος.

4.1.2 Ορισμός εσωτερικής ισχύος.

Η κατανάλωση εσωτερικής ισχύος λαμβάνει χώρα μέσα στα όρια του κελιού. Κατά την διάρκεια της εναλλαγής, ένα κύκλωμα καταναλώνει εσωτερική ισχύ λόγω της φόρτισης και εκφόρτισης των οποιοδήποτε εσωτερικών χωρητικοτήτων που διαθέτει το κελί. Κατανάλωση εσωτερικής ισχύος έχουμε λόγω της στιγμιαίας

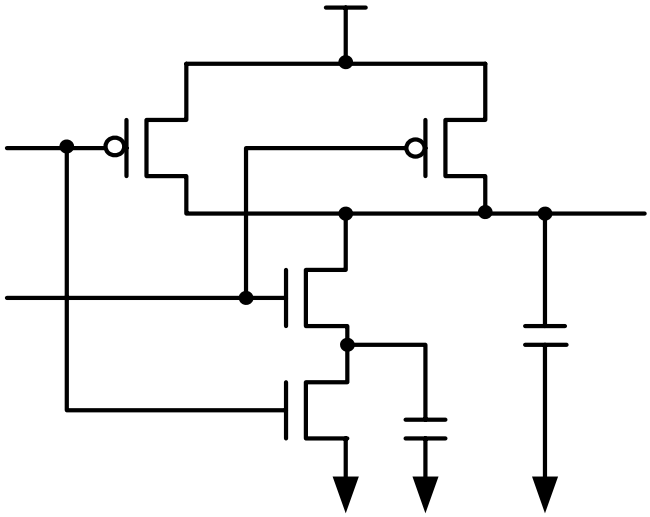
εμφάνισης βραχυκυκλώματος μεταξύ του P και N τρανζίστορ της πύλης, η οποία αναφέρεται και ως ισχύς βραχυκυκλώσεως (short-circuit power).

Για να αντιληφθούμε για ποιο λόγο έχουμε αυτή την κατανάλωση, δώστε προσοχή στην πύλη που παρουσιάζεται στο σχήμα 4.1 παρακάτω. Ένα ανοδικό σήμα εφαρμόζεται στην είσοδο IN. Αφού το σήμα εναλλάσσεται από χαμηλή τιμή σε υψηλή, το τρανζίστορ τύπου N ανοίγει και το τρανζίστορ τύπου P κλείνει. Ωστόσο, για λίγο χρονικό διάστημα όσο έχουμε την εναλλαγή του σήματος, τόσο το P και το N τύπου τρανζίστορ μπορούν να είναι ανοιχτά ταυτόχρονα. Στο αναφερθέν αυτό χρονικό διάστημα το ρεύμα I_{SC} ρέει από την τάση V_{dd} στην γείωση GND, προκαλώντας κατανάλωση ισχύος βραχυκυκλώματος P_{SC}.



Σχήμα 4.1
Μια απλή πύλη όπου φαίνεται σε ποια σημεία έχουμε στατική και δυναμική κατανάλωση ισχύος.

Ένα ακόμα χαρακτηριστικό παράδειγμα κατανάλωσης εσωτερικής ισχύος είναι όταν έχουμε μεταγωγή ενός εσωτερικού κόμβου χωρίς να έχουμε μεταγωγή στην έξοδο. Για παράδειγμα θα δούμε την CMOS πύλη NAND 2-εισόδων που φαίνεται στο παρακάτω σχήμα:



Σχήμα 4.2
. Μια CMOS πύλη NAND 2-εισόδων.

Αν στην είσοδο της πύλης εφαρμόσουμε τις εισόδους που φαίνονται στον παρακάτω πίνακα, η έξοδος δεν αλλάζει κατάσταση όμως ο κόμβος X αλλάζει λογική τιμή με αποτέλεσμα να καταναλώνει ισχύ.

Πίνακας 4.1
Μια CMOS πύλη NAND 2-εισόδων.

A	1	0	0	0	1
B	0	0	1	0	0
OUT	1	1	1	1	1
X	1	1	0	0	1

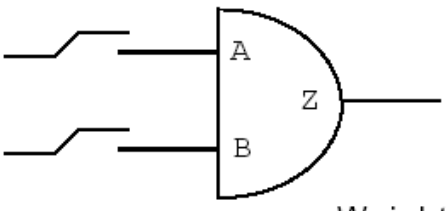
Για κυκλώματα με γρήγορους χρόνους μεταγωγής, η κατανάλωση ισχύος βραχυκυκλώματος μπορεί να είναι μικρή. Ωστόσο, για κυκλώματα με αργούς χρόνους μεταγωγής, η κατανάλωση ισχύος βραχυκυκλώματος μπορεί να προκαλεί περίπου το 30% επί της συνολικής κατανάλωσης της πύλης. Η κατανάλωση ισχύος βραχυκυκλώματος επηρεάζεται από το μέγεθος του τρανζίστορ και την χωρητικότητα φορτίου στην έξοδο της πύλης.

Στα απλά κελιά που παρέχονται από μια βιβλιοθήκη, η εσωτερική ισχύς οφείλεται συνήθως στην ισχύ βραχυκυκλώματος. Για το λόγο αυτό οι δύο αυτοί όροι θεωρούνται συνώνυμοι.

4.1.3 Υπολογισμός Εσωτερικής Ισχύος (Internal Power Calculation).

Για τον υπολογισμό της εσωτερικής κατανάλωσης ισχύος χρησιμοποιείται πληροφορία από την τεχνολογική βιβλιοθήκη. Έτσι, σε κάθε βιβλιοθήκη υπάρχει η αντίστοιχη ομάδα παραμέτρων για INTERNAL_POWER. Οι μηχανικοί που αναπτύσσουν βιβλιοθήκες κατασκευάζουν τον πίνακα κατανάλωσης εσωτερικής ισχύος πάνω στον οποίο μοντελοποιούν την κατανάλωση της εν λόγω ισχύος για κάθε ακροδέκτη (pin) του κελιού.

Η συνολική κατανάλωση της εσωτερικής ισχύος ενός κελιού είναι το άθροισμα της κατανάλωση όλων των ακροδεκτών εισόδου και εξόδου του κελιού σύμφωνα πάντα με την μοντελοποίηση του στην βιβλιοθήκη. Στο σχήμα 4.3 φαίνεται ποίους τύπους χρησιμοποιεί το λογισμικό της SYNOPSYS για power analysis ώστε να υπολογίσει την κατανάλωση εσωτερικής ισχύος για ένα απλό συνδυαστικό κελί, U1.



$$P_{Int} = E_Z \times TR_Z$$

$$E_Z = f\left[C_{Load}, WeightAvg_{(Trans)}\right]$$

$$WeightAvg_{(Trans)} = \frac{\sum_{i=A,B} TR_i \times Trans_i}{\sum_{i=A,B} TR_i}$$

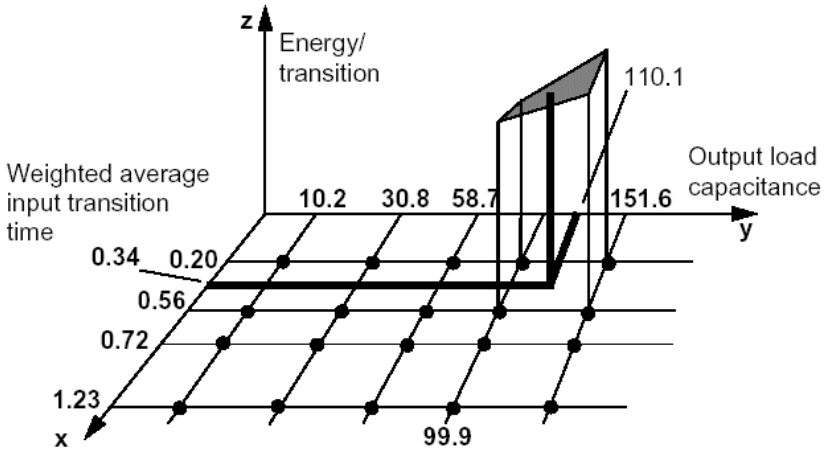
Σχήμα 4.3
Μοντέλο υπολογισμού κατανάλωσης εσωτερικής ισχύος για ένα απλό συνδυαστικό κελί, U1.

όπου:

P_{Int}	: Συνολική εσωτερική ισχύς του κελιού.
E_Z	: Εσωτερική ενέργεια της εξόδου Z συναρτήσει των μεταγωγών της λογικής τιμής της εισόδου και του φορτίου εξόδου.
TR_Z	: Μέσος αριθμός μεταγωγών ακροδέκτη εξόδου Z.
TR_i	: Μέσος αριθμός μεταγωγών ακροδέκτη εισόδου i, εναλλαγές/sec.

$Trans_i$: Χρόνος μεταγωγής εισόδου i.
$WeightAvg_{(Trans)}$: Χρόνος μεταγωγής με βάρη της εξόδου Z.

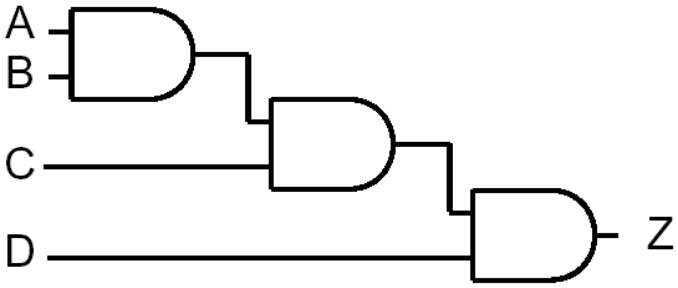
Με βάση πληροφορίες όπως τον μέσο αριθμό μεταγωγών και τον χρόνο μεταγωγής της εισόδου, ο Power Compiler παράγει ένα μέσο χρόνο μεταγωγής με βάρη (weighted average transition time), ο οποίος χρησιμοποιείται ως δείκτης στο πίνακα τιμών για την κατανάλωση εσωτερικής ισχύος στον ακροδέκτη εξόδου. Η χωρητικότητα φορτίου της εξόδου (output load capacitance) χρησιμοποιείται από τον Power Compiler ως πρόσθετος δείκτης. Οι δύο αυτοί δείκτες δίνουν την δυνατότητα στον Power Compiler να διαβάσει τον δισδιάστατο πίνακα τιμών (two-dimensional lookup table) της εξόδου, όπως και φαίνεται στο σχήμα 4.4 :



Σχήμα 4.4
Δισδιάστατος πίνακας τιμών (two-dimensional lookup table) της εξόδου.

Τα κελιά συνήθως καταναλώνουν διαφορετική εσωτερική ισχύ, η οποία εξαρτάται από το ποιος ακροδέκτη εισόδου αλλάζει κατάσταση ή από την κατάσταση της λογικής τιμής που βρίσκεται το κελί. Έχουμε, λοιπόν, κατανάλωση εσωτερικής ισχύος εξαρτώμενη από την κατάσταση (state dependent internal power) και εξαρτώμενη από την διαδρομή (path dependent internal power).

Για να σας παρουσιάσουμε ένα παράδειγμα κατανάλωσης εσωτερικής ενέργειας εξαρτώμενης από τη διαδρομή (path dependent internal power), ας παρατηρήσουμε το σχήμα 4.5 όπου έχουμε ένα απλό κελί μιας βιβλιοθήκης, το οποίο έχει τρία επίπεδα λογικής και ένα πλήθος από ακροδέκτες εισόδου.



Σχήμα 4.5
Απλό κελί με τρία επίπεδα λογικής και τέσσερις εισόδους.

Οι είσοδοι A και D μπορούν κάθε μια ξεχωριστά να προκαλέσουν αλλαγή στην λογική τιμή της εξόδου. Ωστόσο, η είσοδος D επηρεάζει μόνο ένα επίπεδο λογικής, ενώ η είσοδος A επηρεάζει και τις τρεις. Όπως, λοιπόν, είναι φυσιολογικό μια εναλλαγή στην έξοδο Z να καταναλώνει περισσότερη εσωτερική ισχύ όταν είναι αποτέλεσμα μιας αλλαγής της τιμής της εισόδου A σε σύγκρισή με την κατανάλωση όταν προκαλείται από μια εναλλαγή της εισόδου D. Έτσι, μπορούμε να καθορίζουμε πολλαπλούς πίνακες τιμών (lookup tables) για τις εξόδους, εξαρτώμενους από τις εναλλαγές στις εισόδους.

Η επιλογή του κατάλληλου πίνακα με πληροφορίες που έχουν να κάνουν με εξαρτήσεις μονοπατιού για κάθε έξοδο, από τον Power Compiler, γίνεται με τον έλεγχο της μεταβλητής RELATED_PIN που βρίσκεται στην βιβλιοθήκη.

Ένα χαρακτηριστικό παράδειγμα κελιού με κατανάλωση ισχύος με εξάρτηση κατάστασης (state dependent internal power) είναι το κελί μνήμης (RAM cell). Ένα κελί μνήμης καταναλώνει διαφορετικά ποσά εσωτερικής ενέργειας ανάλογα σε τι κατάσταση λειτουργίας (mode) είναι, εγγραφής ή ανάγνωσης. Μπορούμε να ορίσουμε διαφορετικούς πίνακες τιμών για εσωτερική ισχύ, εξαρτώμενους από την λογική κατάσταση ή την κατάσταση λειτουργίας του κελιού.

Όταν ένα σήμα αλλάζει λογική κατάσταση, η εσωτερική ενέργεια που καταναλώνεται όταν το σήμα είναι ανοδικό (από 0 σε 1) είναι διαφορετική από αυτή που καταναλώνεται όταν είναι καθοδικό (από 1 σε 0). Ο Power Compiler υποστηρίζει την δυνατότητα να μπορεί κάποιος να ορίζει ξεχωριστά τις δύο αυτές τιμές ισχύος. Επίσης, υποστηρίζει και μοντέλα βιβλιοθηκών οι οποίες υποστηρίζουν το μέσο όρο αυτών των δύο τιμών.

Στην περίπτωση που ένα κελί έχει ακροδέκτες εισόδου που οι λογικές τους τιμές είναι ίσες ή αντίθετες, ο Power Compiler μπορεί να χρησιμοποιήσει ένα τρισδιάστατο πίνακα τιμών (three-dimensional lookup table). Ο πίνακας αυτός δημιουργείται με των χρόνο εναλλαγής της εισόδου και τις χωρητικότητες εξόδου των δύο ακροδεκτών εξόδου που έχουν ίδιες ή αντίθετες λογικές τιμές. Ένας τέτοιος πίνακας θα μπορούσε

να χρησιμοποιηθεί για την περιγραφή ενός flip-flop, το οποίο έχει Q και Q-bar εξόδους με αντίθετες τιμές.

Η ομάδα παραμέτρων INTERNAL_POWER της βιβλιοθήκης υποστηρίζει μόνο-, δισ- ή τρισ - διάστατους πίνακες τιμών. Ο πίνακα 4.2 παρουσιάζει τους τύπους των πινάκων τιμών, που εφαρμόζονται και τι τιμές καταχωρούνται σε αυτούς.

Πίνακας 4.2
Πίνακες Τιμών (Lookup Tables)

Πίνακας τιμών	**Ορίζεται στην**	**Με δείκτες**
Μονοδιάστατος	Είσοδος Έξοδος	Μεταγωγή εισόδου Χωρητικότητα φορτίου εξόδου
Δισδιάστατος	Έξοδος	Μεταγωγή εισόδου και χωρητικότητα φορτίου εξόδου
Τρισδιάστατος	Έξοδος	Μεταγωγή εισόδου και χωρητικότητα φορτίου εξόδου των δύο ακροδεκτών εξόδου που έχουν ίδιες ή αντίθετες λογικές τιμές

4.1.4 Υπολογισμός Ισχύος Μεταγωγής (Switching Power Calculation).

Ο Power Compiler κατά την ανάλυση ισχύος που πραγματοποιεί υπολογίζει και την ισχύ Μεταγωγής (P_C) σύμφωνα με τον παρακάτω τύπο:

$$P_C = \frac{V_{dd}^2}{2} \sum_{\forall nets(i)} (C_{Load_i} \times TR_i)$$

όπου:

P_C : Ισχύς Μεταγωγής του κυκλώματος
C_{Loadi} : Χωρητικότητα φορτίου του δικτύου i
TR_i : Μέσος αριθμός μεταγωγών του δικτύου i, μεταγωγές/sec
V_{dd} : Τάση τροφοδοσίας

Η παράμετρός C_{Loadi} αντιπροσωπεύει την συνολική χωρητικότητα του δικτύου i, δηλαδή είναι το άθροισμα της παρασιτικής χωρητικότητας (parasitic capacitance), της χωρητικότητας πύλης (gate capacitance) και υποδοχής (drain capacitance) όλων των ακροδεκτών που είναι συνδεδεμένοι με το δίκτυο i.

Το λογισμικό του Power Compiler για τους υπολογισμούς αυτούς χρησιμοποιεί πληροφορία από το τύπο του μοντέλου καλωδίωσης (wire load model) για το δίκτυο και από την πληροφορία που προσφέρει η τεχνολογική βιβλιοθήκη για της πύλες που είναι συνδεδεμένες με το δίκτυο. Επίσης, αν διαθέτουμε και τον Physical Compiler μπορούμε στο κύκλωμα που θα έχουμε σε transistor-level να κρατήσουμε την χωρητική πληροφορία του(back-annotate capacitance) και να την χρησιμοποιήσουμε για πιο ακριβής υπολογισμούς.

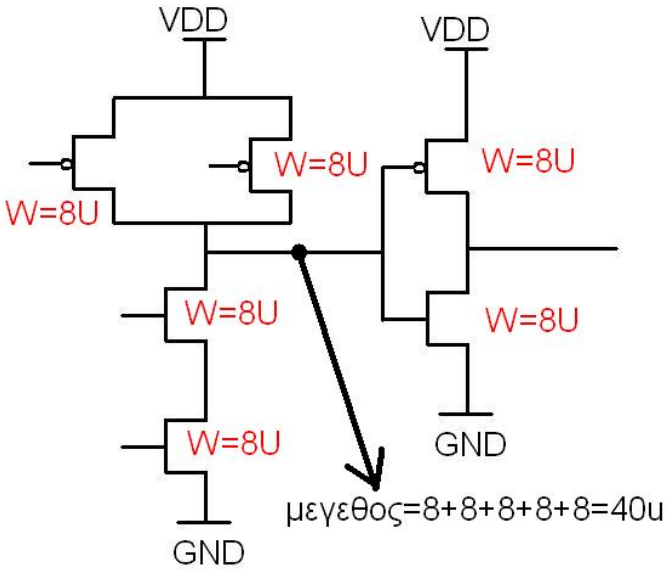
4.1.5 Υπολογισμός Δυναμικής Ισχύος (Dynamic Power Calculation).

Επειδή η δυναμική ισχύς(Dynamic Power) είναι η ισχύς που καταναλώνεται όταν το κύκλωμα είναι ενεργό, το άθροισμα της ισχύος μεταγωγής και της εσωτερικής ισχύος μας δίνει το συνολικό ποσό της δυναμικής ισχύος που καταναλώνεται, άρα:

Δυναμική Ισχύς = Ισχύς Μεταγωγής + Εσωτερική Ισχύς

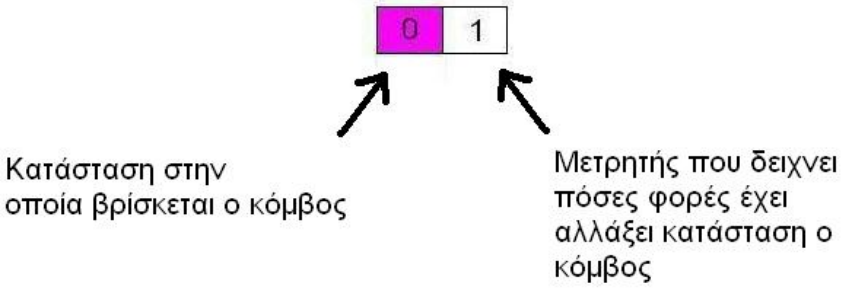
4,2 Υπολογισμός κατανάλωσης δυναμικής ισχύος μέσω προσομοίωσης

Δεύτερη χρήση του προγράμματος είναι η εξαγωγή αποτελεσμάτων για την κατανάλωση δυναμικής ισχύος και συγκεκριμένα ισχύος μεταγωγης. Τα αποτελέσματα της προσομοίωσης βασίζονται στον τύπο $\frac{1}{2}CV^2$ Για να γίνει αυτό χρειάζεται εκ των προτέρων να έχουμε κρατήσει επιπλέον πληροφορία για την χωρητικότητα του κάθε κόμβου. Αυτό γίνεται κατά την δημιουργία του κυκλώματος καθώς στην περιγραφή σε transistor level μας δίνεται και το μέγεθος των τρανζίστορς. Από τα εφαπτόμενα σε κάθε κόμβο τρανζίστορς μπορούμε να εξάγουμε την χωρητικότητα του κάθε κόμβου όπως φαίνεται και στο σχήμα 4.6.



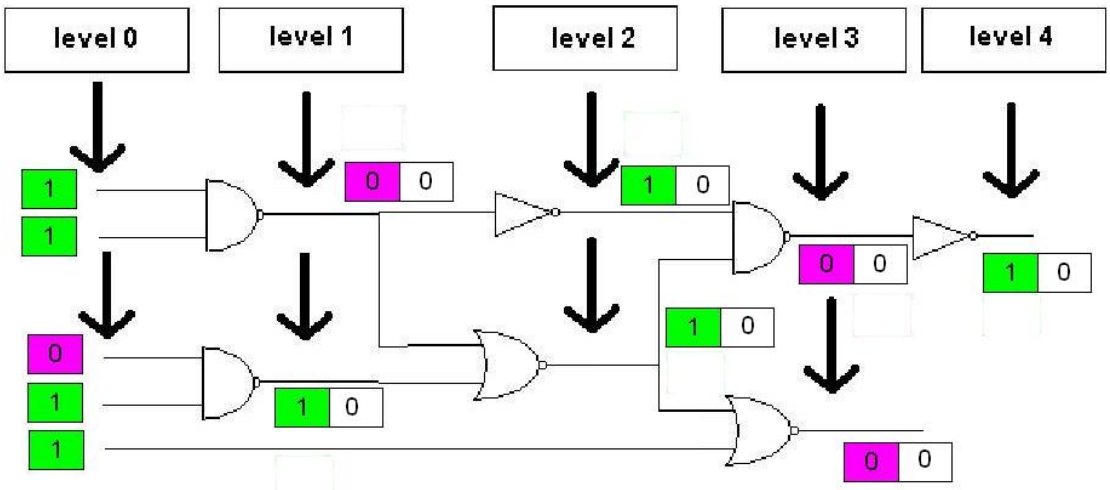
Σχήμα 4.6
Εύρεση χωρητικότητας κόμβου

Επιπλέον χρειαζόμαστε για κάθε κόμβο να βρούμε πόσες φορές θα αλλάξει ενδιάμεσες καταστάσεις όταν ολόκληρο το κύκλωμα περνά από μια κατάσταση σε μια άλλη. Η μέθοδος που χρησιμοποιούμε σε αυτή την προσομοίωση είναι η ανάποδη διάσχιση του κυκλώματος. Έχοντας το κύκλωμα μας αρχικά σε μια κατάσταση, αλλάζουμε τις τιμές των εισόδων και ξεκινώντας από το τελευταίο level διασχίζουμε το κύκλωμα προς τα πίσω. Αυτό που κάνουμε δηλαδή είναι να προσδιορίζουμε την κατάσταση των κόμβων του τελευταίου level, μετά τα προηγούμενου, μετά του προηγούμενου και αυτό συνεχίζεται έως ότου φτάσουμε στο level 1. Σε κάθε κόμβο έχουμε έναν buffer ο οποίος αυξάνει κατά 1 κάθε φορά που ο κόμβος αλλάζει κατάσταση και μετράει τον συνολικό αριθμό που ο κόμβος άλλαξε κατάσταση. Η ποσότητα που μας ενδιαφέρει, η ισχύ μεταγωγής που καταναλώνεται, βγαίνει για κάθε κόμβο από το γινόμενο του πόσες φορές άλλαξε κατάσταση ο κόμβος επί το μέγεθος του.
Σχηματικά η προσομοίωση ακολουθεί τα στάδια που φαίνονται στο σχήμα 4.7. Σε κάθε κόμβο φαίνονται η λογική τιμή του (0 ή 1) καθώς και ο μετρητής που δείχνει πόσες φορές άλλαξε κατάσταση ο κόμβος. Οι μετρητές είναι αρχικοποιημένοι όλοι στο μηδέν στην αρχή της προσομοίωσης όπως επίσης και το κύκλωμα είναι αρχικοποιημένο σε μια κατάσταση (σχήμα 4.7.1). Τους κόμβους που αποτελούν τις εισόδους του κυκλώματος δεν τους εξετάζουμε καθώς θεωρούμε ότι η κατάσταση στην οποία βρίσκονται επηρεάζεται από εξωτερικούς παράγοντες και όχι από το δικό μας κύκλωμα.

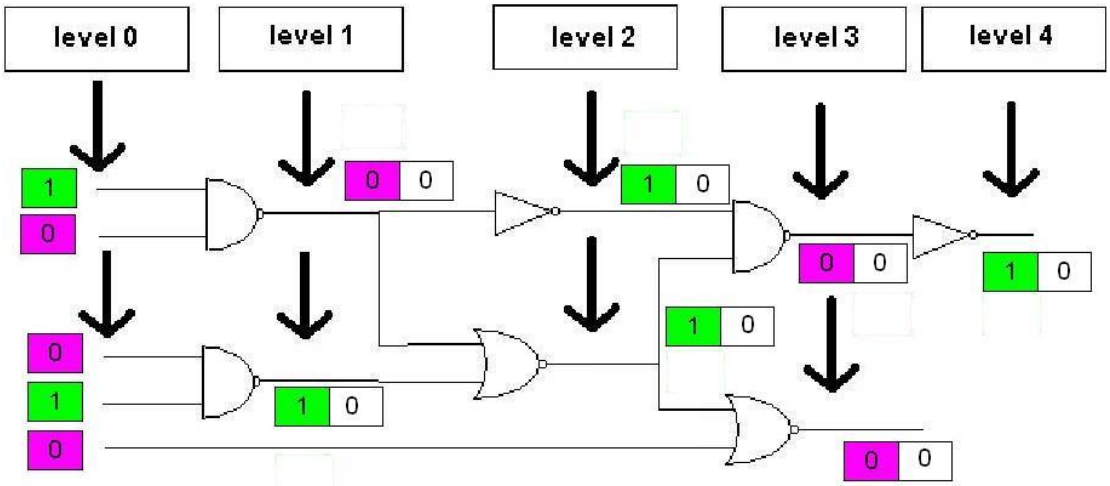


Σχήμα 4.7
Βήματα προσομοίωσης υπολογισμού δυναμικής ενέργειας

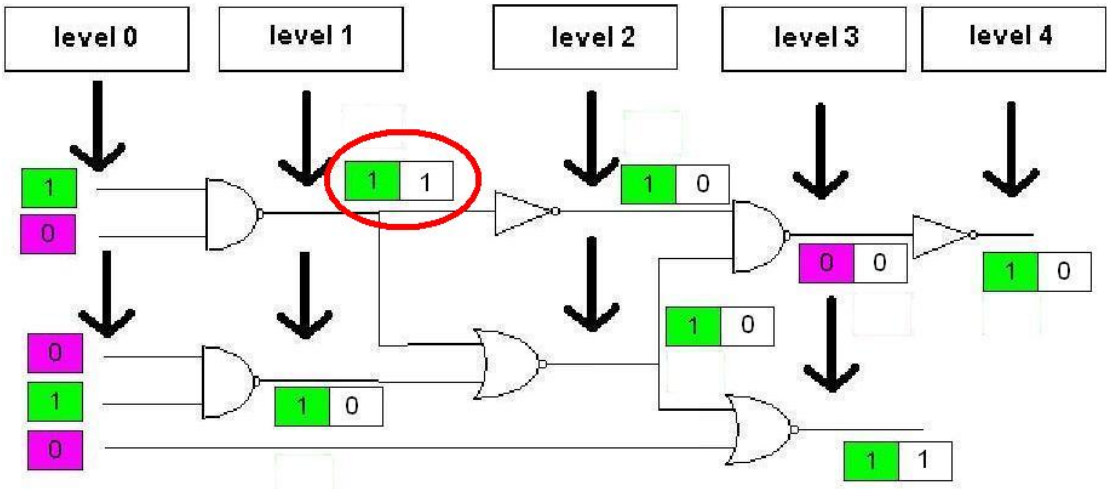
Η προσομοίωση ακολουθεί τα βήματα όπως φαίνονται στα παρακάτω σχήματα. Αρχικά έχουμε το κύκλωμα σε μια σταθερή κατάσταση (σχήμα 4.7.1). Το αμέσως επόμενο βήμα είναι να αλλάξουμε τις εισόδους. Με μια συνάρτηση δίνουμε σε κάθε μια είσοδο μια νέα τιμή τυχαία και ανεξάρτητη από την προηγούμενη (σχήμα 4.7.2).



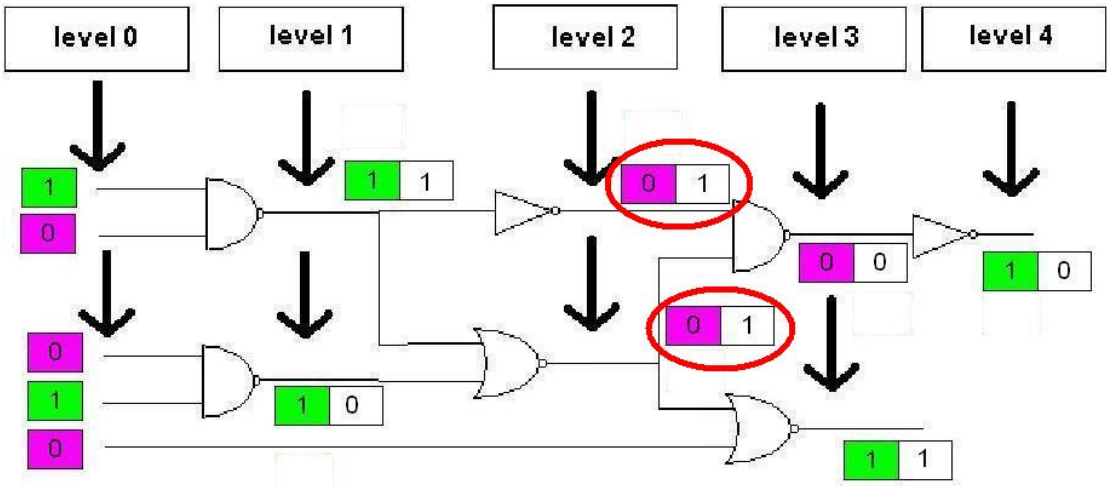
Σχήμα 4.7.1
Το κύκλωμα σε σταθερή κατάσταση



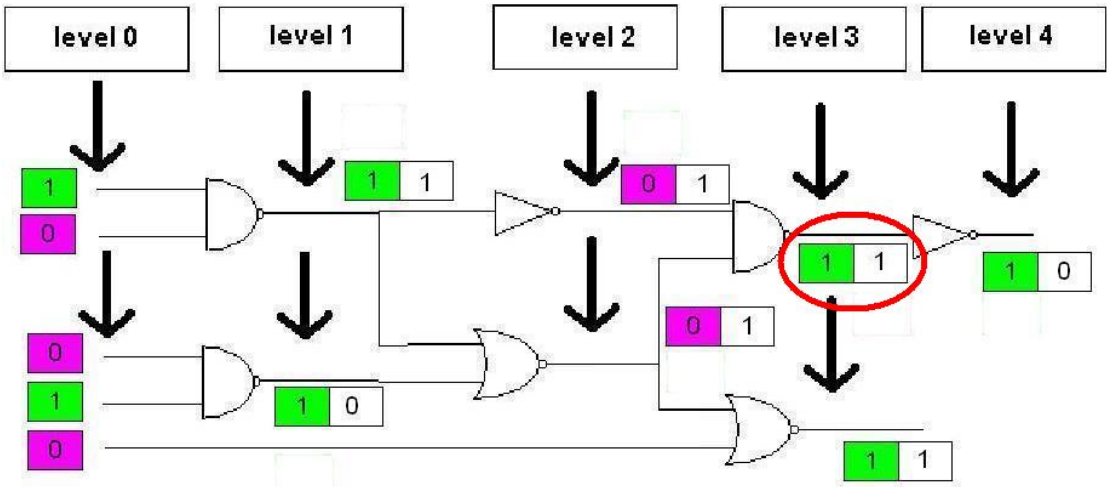
Σχήμα 4.7.2
Νέο vector εισόδων



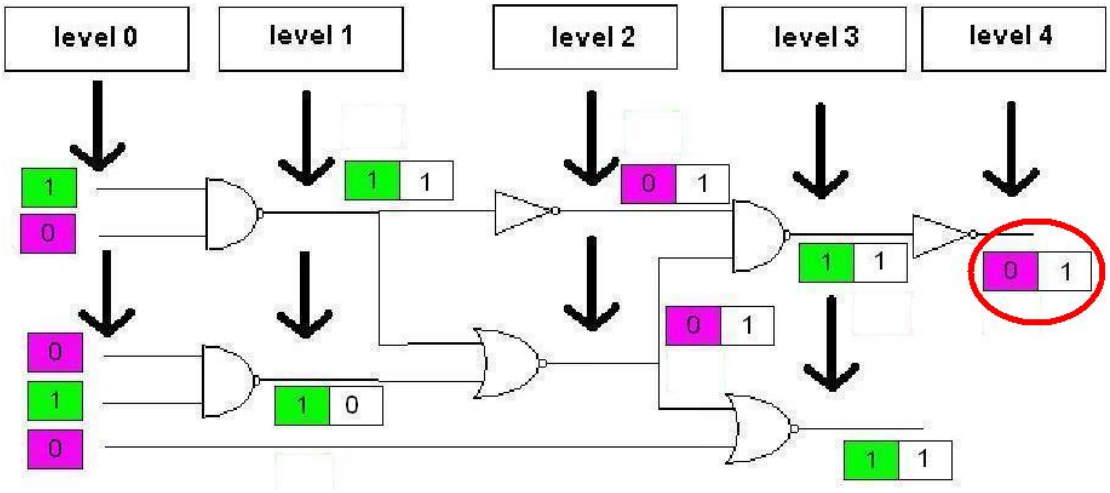
Σχήμα 4.7.3
Μετά την πρώτη προς τα πίσω διάσχιση του γράφου



Σχήμα 4.7.4
Μετά την δεύτερη προς τα πίσω διάσχιση του γράφου



Σχήμα 4.7.5
Μετά την τρίτη προς τα πίσω διάσχιση του γράφου



Σχήμα 4.7.6
Μετά την τέταρτη προς τα πίσω διάσχιση του γράφου

Ο αριθμός των προς τα πίσω διασχίσεων πρέπει να είναι ίδιος με τον αριθμό των levels του κυκλώματος. Αφού κάνουμε αυτόν τον αριθμό οπισθοδρομικών διασχίσεων το κύκλωμα θα έχει πλέον σταθεροποιηθεί στην νέα κατάσταση που επιβάλει το νέο vector εισόδων. Για μια νέα προσομοίωση μηδενίζουμε τους μετρητές των κόμβων, αλλάζουμε ξανά τις εισόδους όπως στο σχήμα 4.7.2 και ακολουθούμε τα ίδια βήματα. Στο τέλος κάθε προσομοίωσης για να υπολογίσουμε το πόσο δυναμικής ενέργειας υπολογίζουμε για κάθε κόμβο το γινόμενο χωρητικότητα με τον αριθμό αλλαγών καταστάσεων και αθροίζουμε όλα τα ποσά από όλους τους κόμβους. Είναι φανερό ότι μεγαλύτεροι σε μέγεθος κόμβοι καταναλώνουν περισσότερη ισχύ κάθε φορά που χρειάζεται να αλλάξουν κατάσταση. Αυτή η προσομοίωση γίνεται γιατί όταν επέλθει μια αλλαγή στις εισόδους ενός κυκλώματος δεν περνάει αμέσως σε όλους τους κόμβους. Αυτό που γίνεται είναι ότι ο κόμβος μπορεί να αλλάξει και περισσότερες από μια φορές έως ότου σταθεροποιηθεί στην τελική του κατάσταση, η οποία παρεμπίπτοντος μπορεί να είναι ίδια με την αρχική αλλά στο ενδιάμεσο ο κόμβος να άλλαξε τιμή. Αυτές οι ενδιάμεσες αλλαγές όμως σημαίνουν κατανάλωση ισχύος, πράγμα που το πρόγραμμα προσπαθεί να εντοπίσει και να μετρήσει.

Όπως είναι φανερό για να γίνει μια ολοκληρωμένη μέτρηση από την στιγμή που οι είσοδοι αλλάζουν έως ότου όλοι οι κόμβοι σταθεροποιηθούν σε μια κατάσταση χρειάζεται να γίνουν αρκετές διασχίσεις του κυκλώματος. Συγκεκριμένα ο αριθμός των διασχίσεων που πρέπει να γίνουν είναι ίσος με τον αριθμό των levels που υπάρχουν στο κύκλωμα. Αυτή η ιδιαιτερότητα όμως σημαίνει ότι ο χρόνος για να ολοκληρωθεί μια πλήρης προσομοίωση, όπου οι είσοδοι θα αλλάξουν χιλιάδες φορές, είναι πολλαπλάσιος του χρόνου που θα χρειαζόταν αν είχαμε μόνο μια διάσχιση όπως

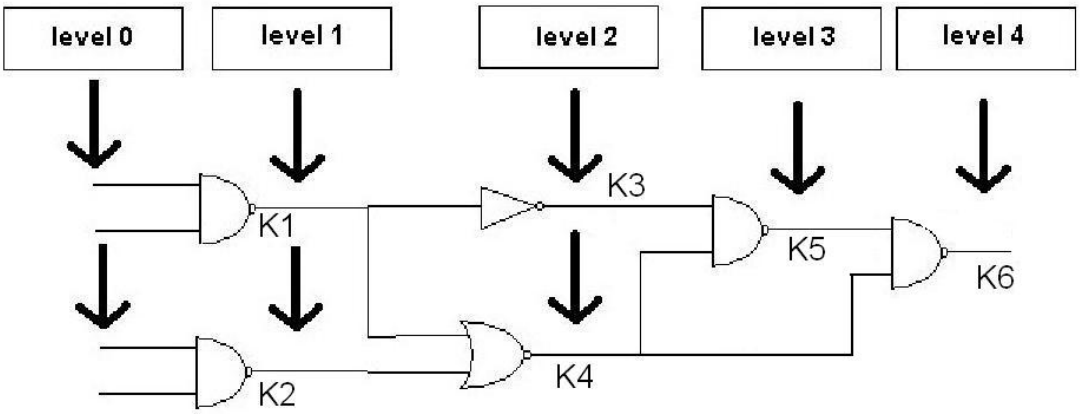
στον υπολογισμό στατικής ισχύος. Για αυτόν τον λόγο προσπαθήσαμε να βρούμε μεθόδους που θα μείωναν κάπως τις απαιτήσεις σε χρόνο. Οι μέθοδοι που χρησιμοποιήθηκαν παρατίθενται παρακάτω.

4.2.1 Μέθοδοι βελτιστοποίησης χρόνου προσομοίωσης υπολογισμού δυναμικής ισχύος.

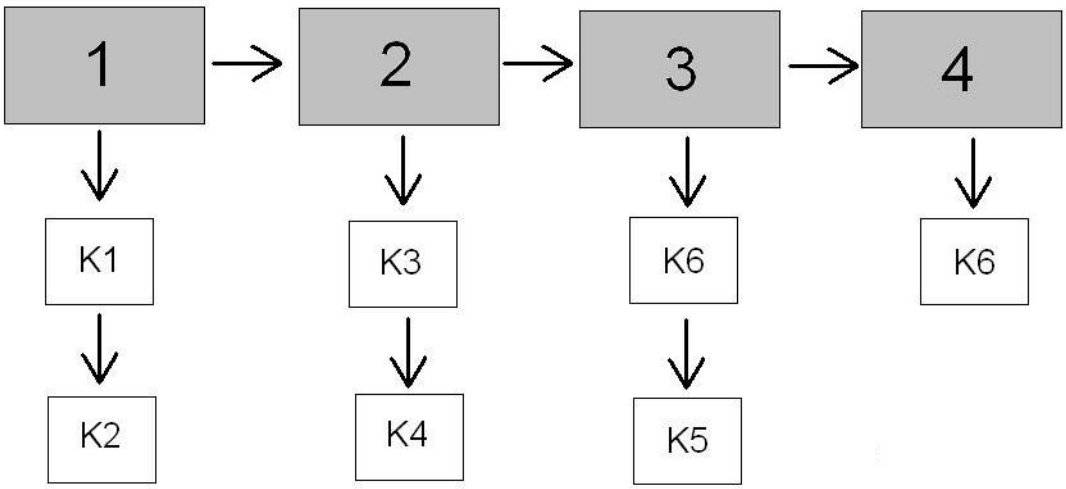
Η προσπάθεια εύρεσης τρόπων βελτίωσης του χρόνου εκτέλεσης στηρίχτηκε στην παρατήρηση κάποιων ιδιαιτεροτήτων που υπάρχουν στην 'ανάποδη' διάσχιση του κυκλώματος. Συγκεκριμένα για να έχει ένας κόμβος πιθανότητα να αλλάξει σε μια διάσχιση θα πρέπει ένας από τους κόμβους που αποτελούν είσοδο της πύλης που τον οδηγεί να είχε αντίστοιχα πιθανότητα στην αμέσως προηγούμενη διάσχιση. Έτσι στην πρώτη διάσχιση από την αλλαγή των εισόδων πιθανότητα να αλλάξουν έχουν μόνο οι κόμβοι που οδηγούνται από μια τουλάχιστον είσοδο του κυκλώματος. Στο αμέσως επόμενο πέρασμα υποψήφιοι για αλλαγή είναι οι κόμβοι που οδηγούνται από έναν τουλάχιστον από του προηγουμένως υποψήφιους κόμβους. Ένας κόμβος είναι πολύ πιθανόν να είναι αρκετές φορές υποψήφιος για αλλαγή. Με αυτόν τον τρόπο μπορούμε να απομονώσουμε εκείνους τους κόμβους οι οποίοι γνωρίζουμε εκ των προτέρων ότι είναι αδύνατον να αλλάξουν κατάσταση οποιαδήποτε και αν είναι η κατάσταση του κυκλώματος. Μια συνάρτηση του προγράμματος κάνει αυτόν τον διαχωρισμό. Ο τρόπος που αυτή η συνάρτηση το επιτυγχάνει εξηγείται παρακάτω.

Ξεκινώντας έχουμε σαν δεδομένο ότι ο αριθμός των διασχίσεων είναι ίσος με τον αριθμό των levels. Έστω ότι με no_of_levels συμβολίζουμε αυτόν τον αριθμό. Δημιουργούμε λοιπόν μια λίστα με no_of_levels στοιχεία, όπου κάθε στοιχείο αυτής της λίστας είναι μια νέα λίστα που αποθηκεύει κόμβους. Έτσι οι κόμβοι που θα αποθηκεύονται στο πρώτο στοιχείο της λίστας είναι οι κόμβοι που πρέπει να ελεγχθούν στην πρώτη διάσχιση, αυτοί στο δεύτερο στοιχείο στην δεύτερη διάσχιση κ.ο.κ.. Στην συνέχεια μια αναδρομική συνάρτηση θα τρέξει για κάθε μια είσοδο του κυκλώματος και θα διασχίσει προς τα εμπρός το κύκλωμα. Για κάθε κόμβο που περνάει θα αυξάνει και έναν αρχικά μηδενισμένο μετρητή. Έτσι σε κάθε κόμβο θα έχει μετρήσει τους κόμβους που τον χωρίζουν από την συγκεκριμένη είσοδο. Αυτός ο κόμβος θα μπαίνει στην λίστα, στο στοιχείο με αριθμό ίσο με αυτόν που έχει ο μετρητής. Είναι φανερό ότι η μεγαλύτερη τιμή που μπορεί να πάρει ο μετρητής είναι no_of_levels. Εάν ο κόμβος βρίσκεται ήδη μέσα σε εκείνη την λίστα δεν ξαναμπαίνει. Επειδή όμως υπάρχουν πολλές λίστες, ένας κόμβος μπορεί να μπει σε περισσότερες από μια λίστες. Επαναλαμβάνουμε αυτή την διαδικασία για κάθε είσοδο. Στο τέλος έχουμε μέσα σε κάθε λίστα, που αντιστοιχεί σε μια διάσχιση, τους κόμβου που έχουν πιθανότητα να αλλάξουν σε αυτή την διάσχιση και όχι εκείνους που είναι σίγουρο ότι θα παραμείνουν σταθεροί. Πρέπει επίσης να επισημάνουμε ότι όταν οι κόμβοι μπαίνουν σε μια από τις λίστες, μπαίνουν με σειρά το level στο οποίο ανήκουν κατά φθίνουσα σειρά. Όταν θελήσουμε να τρέξουμε την προσομοίωση θα τρέξουμε πρώτα τους κόμβους της πρώτης λίστας, μετά εκείνους της δεύτερης έως ότου φτάσουμε στο νούμερο no_of_levels.

Παρακάτω φαίνεται σε ένα παράδειγμα το πως λειτουργεί αυτή η μέθοδος σε ένα απλό κύκλωμα. Ας θεωρήσουμε ότι έχουμε το κύκλωμα του σχήματος 4.8. Κάθε κόμβος έχει ονομαστεί με KX με X από 1 έως 6. Η λίστα όπως την περιγράψαμε πιο πάνω μετά από την μελέτη και των τεσσάρων εισόδων θα έχει την μορφή του σχήματος 4.9.



Σχήμα 4.8
Ένα κύκλωμα και οι κόμβοι του



Σχήμα 4.9
Μέθοδος βελτιστοποίησης χρόνου προσομοίωσης υπολογισμού δυναμικής ενέργειας

Βλέπουμε ότι για 4 συνολικά διασχίσεις έχουμε να εξετάσουμε μονάχα 7 φορές κάποιον κόμβο αν άλλαξε κατάσταση. Με την κλασική μέθοδο θα είχαμε και τους 6 κόμβους επί 4 διασχίσεις 24 φορές έλεγχο σε κάποιο κόμβο. Αν

αναλογιστούμε ότι αυτή η διαδικασία επαναλαμβάνεται για χιλιάδες διαφορετικά vectors εισόδου διαπιστώνουμε ότι το κέρδος σε χρόνο είναι πολύ σημαντικό. Βέβαια απαιτείται και κάποιος χρόνος για να φτιαχτούν αυτές οι λίστες το κέρδος όμως από την προσομοίωση είναι πολύ περισσότερο ειδικά αν ο αριθμός των προσομοιώσεων είναι ιδιαίτερα μεγάλος. Δυστυχώς αυξάνονται και οι απαιτήσεις σε μνήμη αλλά αν στόχος μας είναι το κέρδος σε χρόνο τότε αυτή η λύση έχει σπουδαία αποτελέσματα.

Παρακάτω βρίσκεται ο αλγόριθμος που δημιουργεί τις λίστες σαν αυτές του σχήματος 4.9.

αλγόριθμος 4 (Βελτιστοποίηση)

```
1:      for(each n ∈ input)
2:          InsertList(n,0)
```

αλγόριθμος 4.1

```
1:      InsertList (n,counter)
2:      {
3:          if(MPASS(n)=not pass)
5:          {
6:              MPASS(n)=pass
7:
8:          InsertNodeToList (n ,counter)
9:
10:     /*Για κάθε output πύλης οδηγούμενης από το n*/
11:             for(each ni ∈ OUTPUT(NSUBC_IN(n))
12:                 InsertList (ni,counter+1)
13:
14:             MPASS(n)=not pass
15:         }
16:     }
```

αλγόριθμος 4.2

```
/*εισάγει τον κόμβο n στο στοιχείο #counter της αρχικής λίστας*/
1:      InsertNodeToList (n ,counter)
2:      {
3:          if(n not exists in List Element # counter)
4:              insert n into List Element # counter
5:              sort by n->level
6:      }
```

4.3 Απαιτήσεις σε χρόνο

Παρακάτω παρουσιάζεται ένας πίνακας με τους χρόνους που χρειάζεται το σύστημα για μια προσομοίωση με την μέθοδο που φαίνεται στο σχήμα 4.7. Για κάθε

κύκλωμα γίνεται και έναν συγκεκριμένο αριθμό επαναλήψεων ανάλογα με το μέγεθος του κυκλώματος .Οι χρόνοι είναι σε msec.

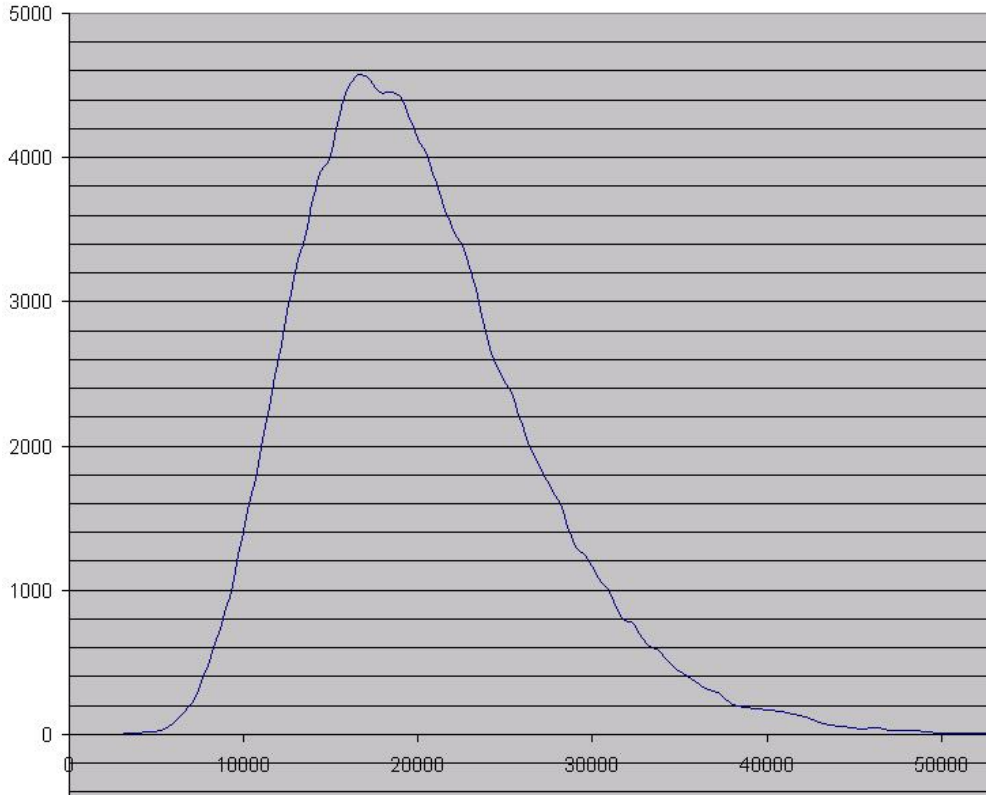
Κύκλωμα	Χρόνος προσομοίωσης σε msec	No of tests
C17	47	10000
C432	6953	10000
C499	9485	10000
C880	28922	10000
C1908	147078	10000
C2670	247547	10000
C5315	84671	1000
C6288	180922	1000
C7552	10656	100
S27	156	10000
S208	3047	10000
S208_1	3047	10000
S298	3297	10000
S344	8234	10000
S349	8360	10000
S382	3593	10000
S386	6985	10000
S400	3828	10000
S420	12406	10000
S420_1	7719	10000
S444	4672	10000
S520	6844	10000
S526	5859	10000
S526N	5828	10000
S641	68937	10000
S713	73094	10000
S820	10516	10000
S832	10812	10000
S838	22656	10000
S838_1	22312	10000
S953	16000	10000
S1196	44000	10000
S1238	41875	10000
S1423	144234	10000
S1488	56984	10000
S1494	57235	10000
S5378	39640	1000

S9234	19688	100
S9234_1	20016	100
S13207	28578	100
S15850	47609	100
S15850_1	49485	100
35932	32984	100
S38417	69391	100
S38584	81031	100
S38584_1	81406	100

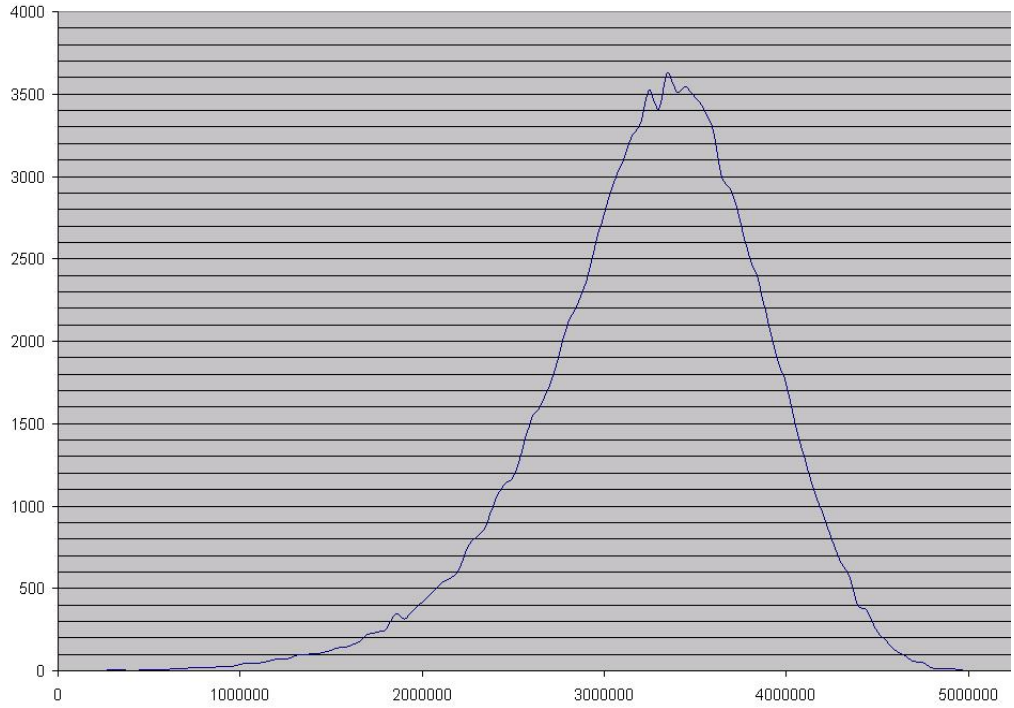
4.4 Αποτελέσματα

Τα αποτελέσματα που βγάζει το πρόγραμμα σε αυτή την προσομοίωση είναι ένα αρχείο με την τιμή που βρίσκει για κάθε προσομοίωση. Έτσι αν κάναμε για παράδειγμα σε ένα κύκλωμα 100.000 προσομοιώσεις το πρόγραμμα θα μας έβγαζε ένα αρχείο με 100.000 τιμές δυναμικής ενέργεια που καταναλώθηκε για 100.000 τυχαία τιμές εισόδου. Μερικά από τα αποτελέσματα που έβγαλε το πρόγραμμα παρουσιάζονται παρακάτω. Στον άξονα x παρουσιάζονται τα ποσά ενέργειας και στον άξονα y οι συχνότητα εμφάνισης.

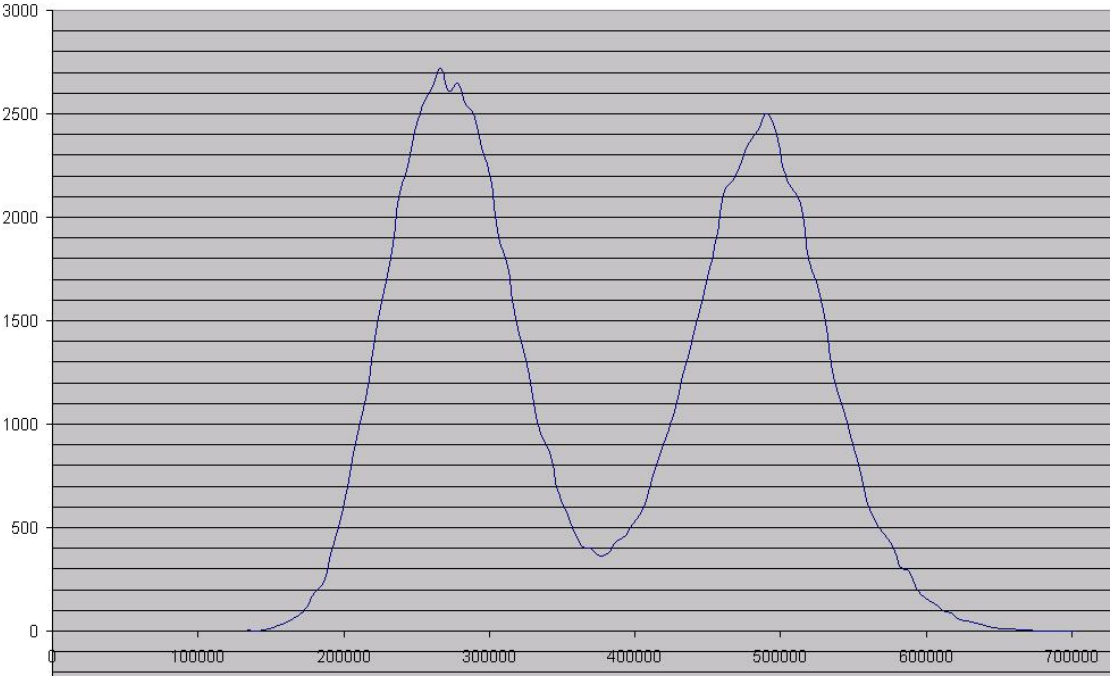
c880



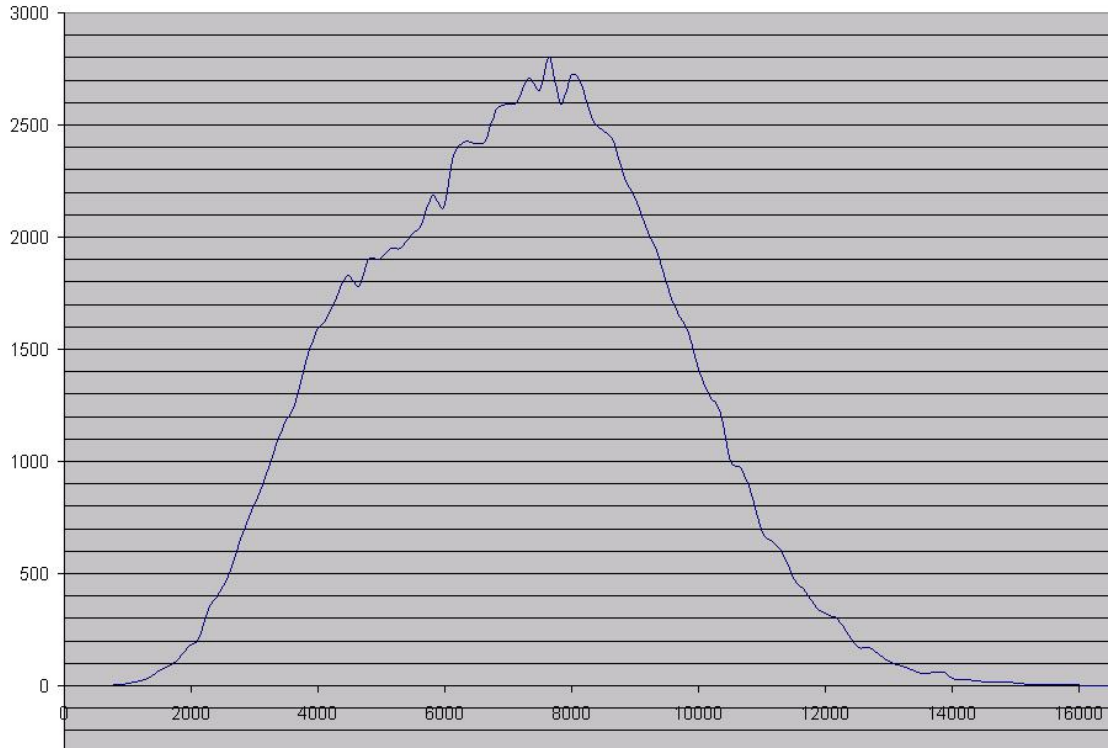
c6288



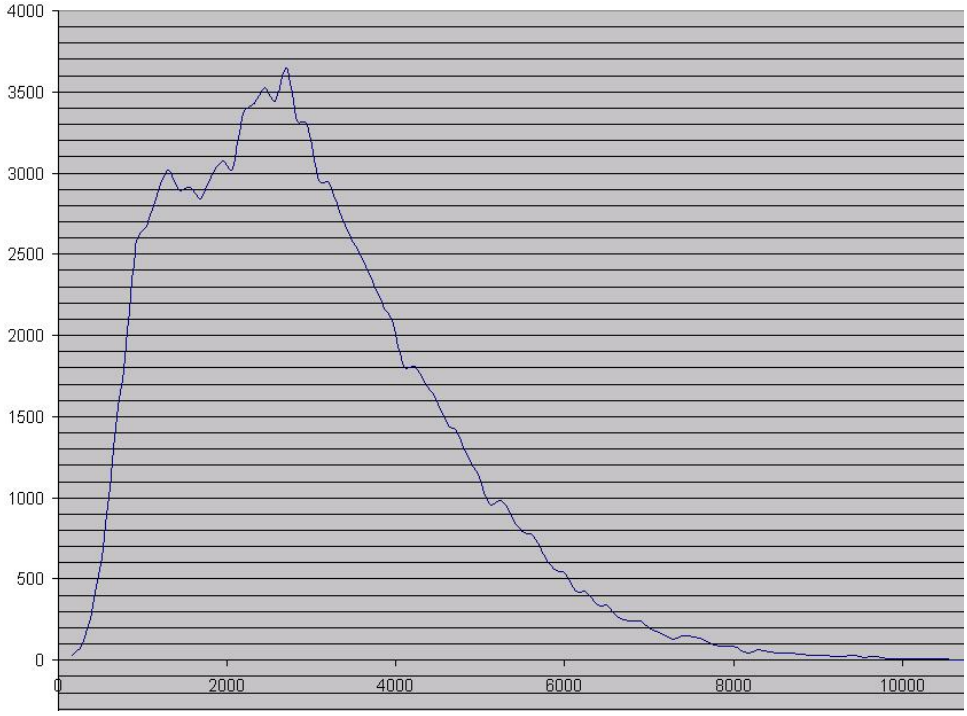
c7552



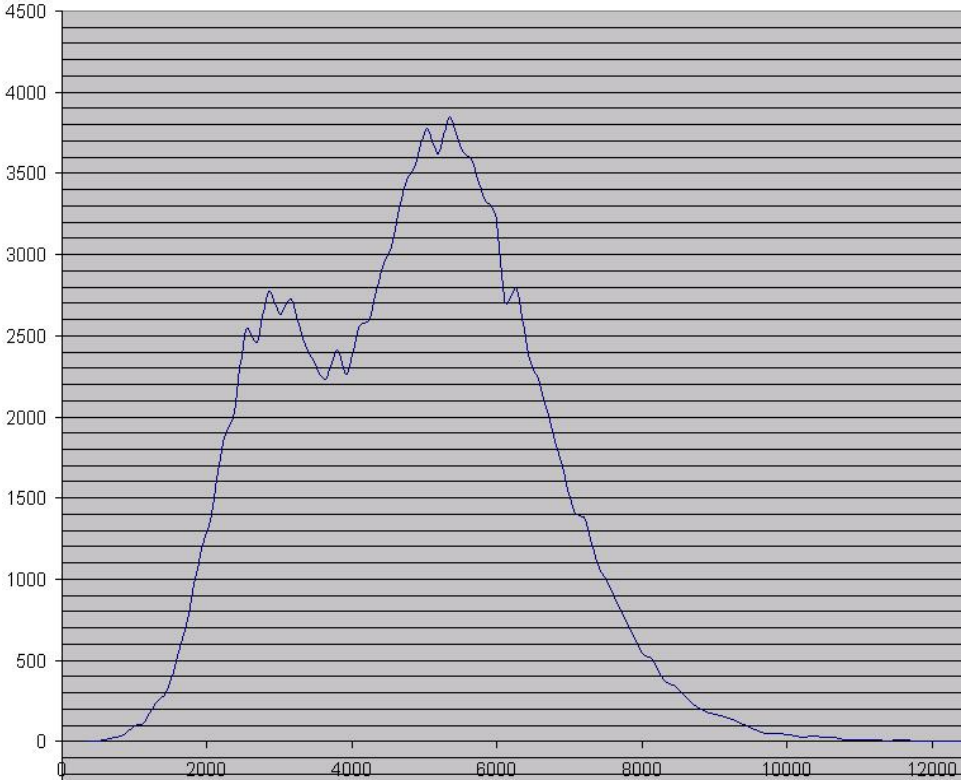
s400


3000
2500
2000
1500
1000
500
0
0
2000
4000
6000
8000
10000
12000
14000
16000

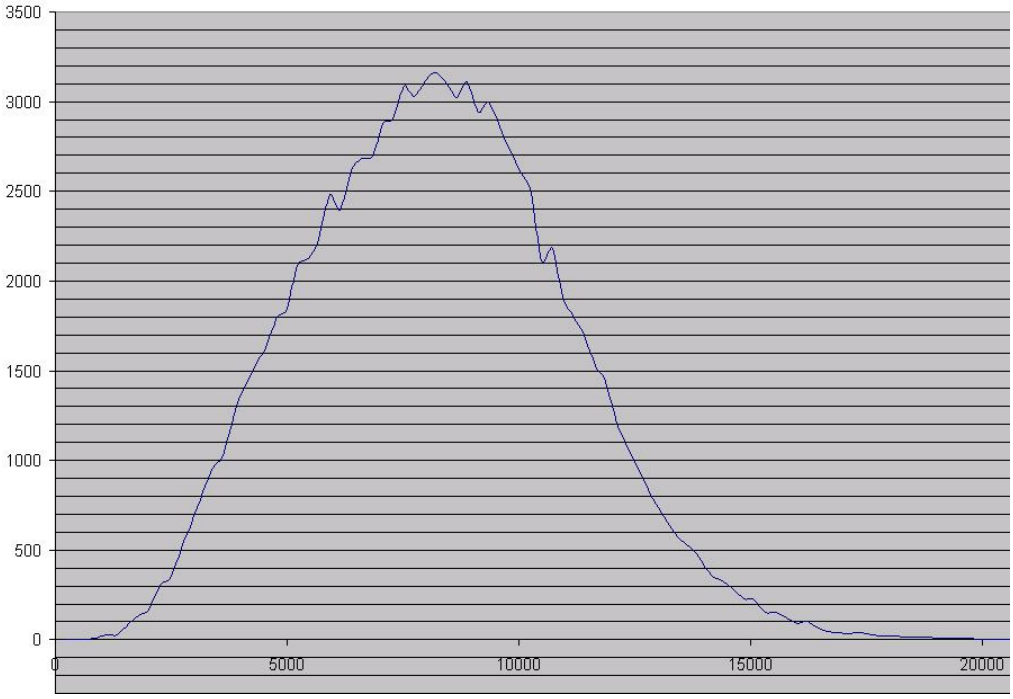
s420


4000
3500
3000
2500
2000
1500
1000
500
0
0
2000
4000
6000
8000
10000

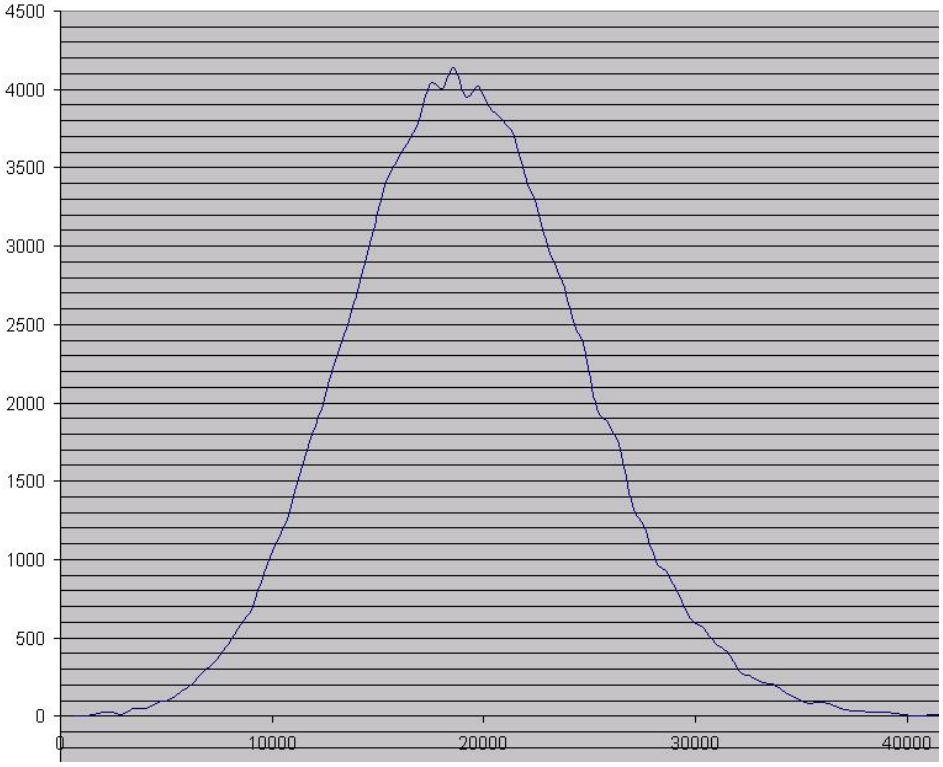
s420_1


4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
2000
4000
6000
8000
10000
12000

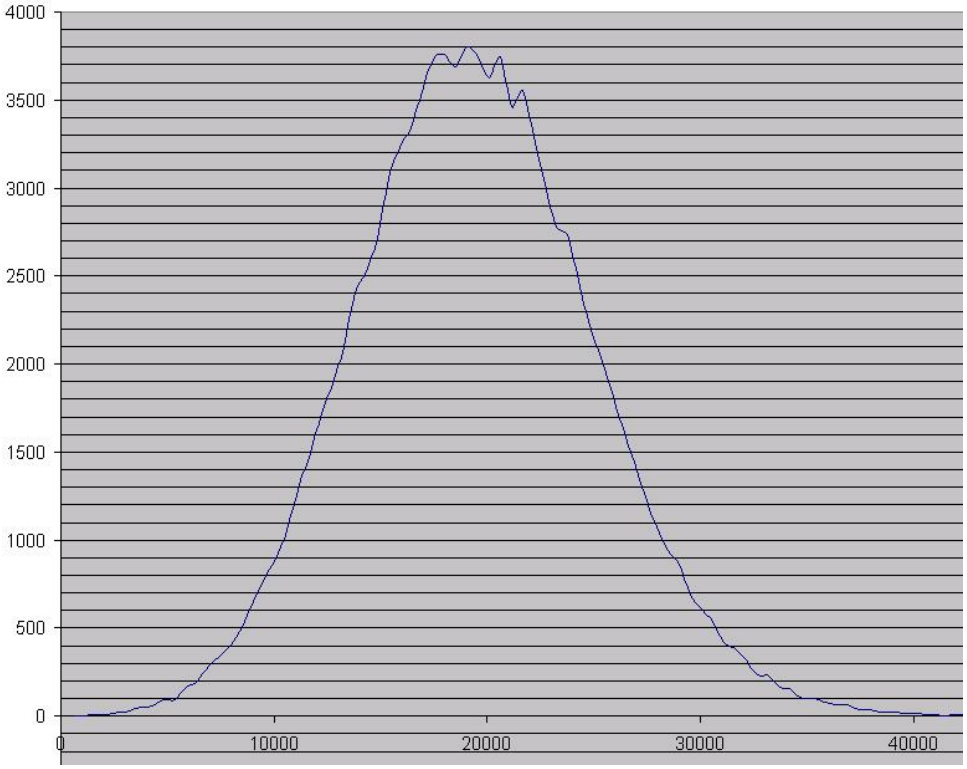
s444


3500
3000
2500
2000
1500
1000
500
0
0
5000
10000
15000
20000

s1196


4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
10000
20000
30000
40000

s1238


4000
3500
3000
2500
2000
1500
1000
500
0
0
10000
20000
30000
40000

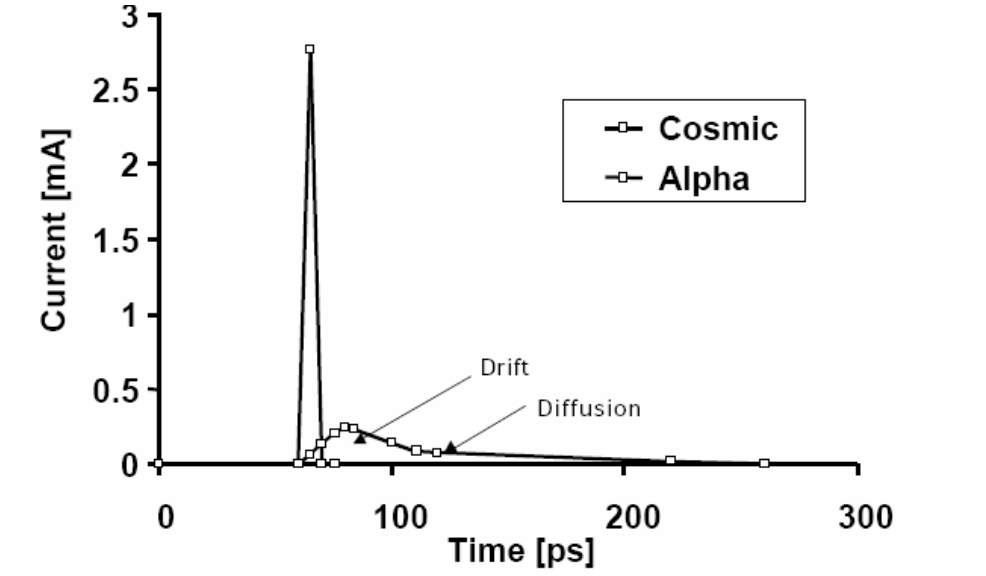
Κεφάλαιο 5ο
SINGLE EVENT UPSETS

5.1 Ορισμός SINGLE EVENT UPSET (SEU)

Με αυτό τον όρο αναφερόμαστε στο φαινόμενο εκείνο όπου σε ένα κύκλωμα έχουμε ξαφνική ανεπιθύμητη αλλαγή της κατάστασης ενός κόμβου που προκλήθηκε από διάφορους παράγοντες. Τέτοιοι παράγοντες μπορεί να είναι η επίδραση κάποιας εξωτερικής ακτινοβολίας , η αλληλεπίδραση με άλλους κόμβους ή και ακόμα η φθορά του χρόνου στο κύκλωμα. Από αυτούς τους τρεις παράγοντες ο πρώτος παρουσιάζει το μεγαλύτερο ποσοστό επίδρασης. Αυτό γιατί η αλληλεπίδραση των κόμβων μεταξύ τους αντιμετωπίζεται από την σχεδίαση του κυκλώματος ενώ η φθορά του χρόνου είναι παράγοντας αναμενόμενος, αποδεκτός και μη αναστρέψιμος.

5.1.1 Η ρόλος της εξωτερικής ακτινοβολίας στην πρόκληση SEU

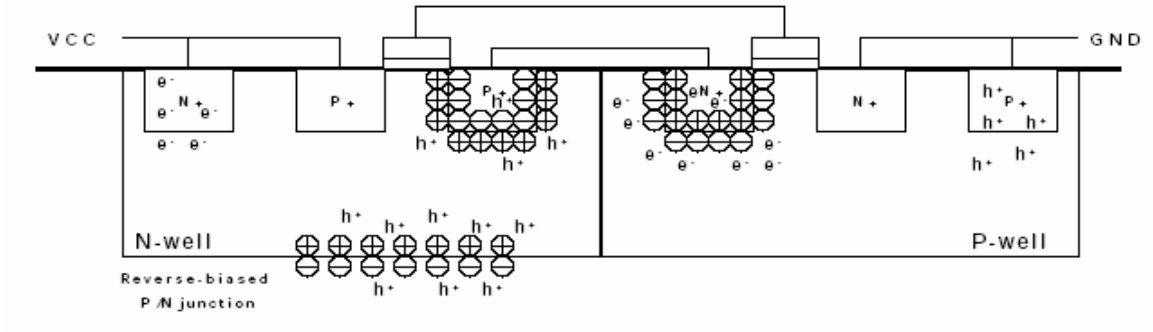
Η κυριότερες μορφές ακτινοβολίας που επηρεάζουν την λειτουργία ενός κυκλώματος είναι η ακτινοβολία άλφα και η κοσμική ακτινοβολία. Οι κυματομορφές αυτών των δυο φαίνονται στο σχήμα 5.1



Σχήμα 5.1
Κυματομορφές ακτινοβολίας άλφα και κοσμικής ακτινοβολίας

Στο σχήμα φαίνεται ότι η κοσμική ακτινοβολία είναι πιο έντονη αλλά έχει μικρή διάρκεια σε σχέση με την ακτινοβολία άλφα.

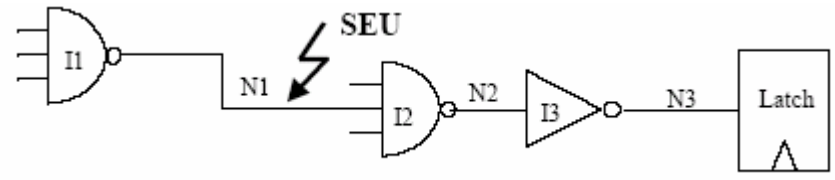
Τι γίνεται κατά την διάρκεια επίδρασης της ακτινοβολίας στο κύκλωμα; Όπως φαίνεται και στο σχήμα 5.2 ηλεκτρόνια θα συγκεντρωθούν στο N+ drain με αποτέλεσμα να 'τραβάνε' τον κόμβο σε λογικό 0 αν βρίσκεται σε λογικό 1. Την ίδια στιγμή οπές θα συγκεντρωθούν στο P+ drain με αποτέλεσμα να 'τραβάνε' τον κόμβο σε λογικό 1 αν βρίσκεται σε λογικό 0



Σχήμα 5.2
Η επίδραση της ακτινοβολίας στον κόμβο

Αυτού του είδους τα φαινόμενα τα λέμε και soft errors και εμφανίζονται σε κυκλώματα όπως επεξεργαστές , μνήμες και κάθε είδους chip. Έχει παρατηρηθεί ότι εμφανίζονται πιο έντονα και πιο συχνά σε μεγάλα υψόμετρα από ότι σε ύψος κοντά στο επίπεδο της θάλασσας. Μελέτες έδειξαν ότι η συχνότητα εμφάνισης τους στο Denver που βρίσκεται σε υψόμετρο 1,5 km 4 με 5 φορές μεγαλύτερη από ότι στην Νέα Υόρκη που βρίσκεται σε υψόμετρο κάτω των 100 ποδών.
Ένα SEU μπορεί να εξελιχθεί σε αλλοίωση των δεδομένων αν ,όπως φαίνεται στο σχήμα 5.3 , ισχύουν τα παρακάτω

- Η αλλαγή στην τάση είναι αρκετά μεγάλη για να θεωρηθεί σαν αλλαγή κατάστασης.
- Δεν εξαφανίζεται κατά την διάρκεια που η αλλαγή διαδίδεται στα l2 και l3.
- Έχει λογικό μονοπάτι προς έναν latch και δεν εξαλείφεται από τον συνδυασμό των εισόδων του l2



Σχήμα 5.3
Ένα SEU σε ένα κύκλωμα

5.2 Υπολογισμός συμπεριφοράς του κυκλώματος σε single event upsets (SEU)

Τρίτη χρήση του προγράμματος είναι η μελέτη της συμπεριφοράς του κυκλώματος σε περιπτώσεις που έχουμε single event upset δηλαδή όταν ένας κόμβος για κάποιο λόγο πάει σε λάθος κατάσταση από αυτή που θα έπρεπε. Τα κυκλώματα που εξετάζουμε είναι ακολουθιακά και για την εξαγωγή συμπερασμάτων σχετικά με το αν επηρεάζεται το κύκλωμα ή όχι εξετάζουμε τις εισόδους των DFF's.

Για μια πλήρη προσομοίωση χρειάζονται να γίνουν κάποια βήματα. Αρχικά δίνουμε τυχαίες εισόδους στο κύκλωμα και αφού το διασχίσουμε όλο, αρχικοποιούμε όλους τους κόμβους στις τιμές που πρέπει να έχουν για το συγκεκριμένο vector εισόδων. Στην συνέχεια παίρνουμε έναν-έναν τους κόμβους ,τους αλλάζουμε κατάσταση και μελετάμε πως αυτή η αλλαγή επηρεάζει το κύκλωμα. Από τον κόμβο που αλλάξαμε αρχίζουμε να διαδίδουμε αυτήν την αλλαγή προς τα εμπρός έως ότου φτάσουμε στο τέλος του κυκλώματος. Στην συνέχεια εξετάζουμε τις εισόδους των DFF's προκειμένου να διαπιστώσουμε αν έστω μια από αυτές έχει πάρει τιμή διαφορετική από αυτήν που είχε αρχικά. Εάν αυτό συμβαίνει θεωρούμε πως αυτή η αλλαγή της τιμής του κόμβου επηρέασε την λειτουργία του κυκλώματος. Μετά το τέλος της μελέτης ενός κόμβου αρχικοποιούμε ξανά το κύκλωμα και περνάμε στον επόμενο κόμβο ακολουθώντας την ίδια διαδικασία. Αφού τελειώσουμε με όλους τους κόμβους δίνουμε νέες τιμές εισόδου στο κύκλωμα, το αρχικοποιούμε σε μια νέα κατάσταση και επαναλαμβάνουμε ξανά όλα τα μέρη για όλους ξανά τους κόμβους.

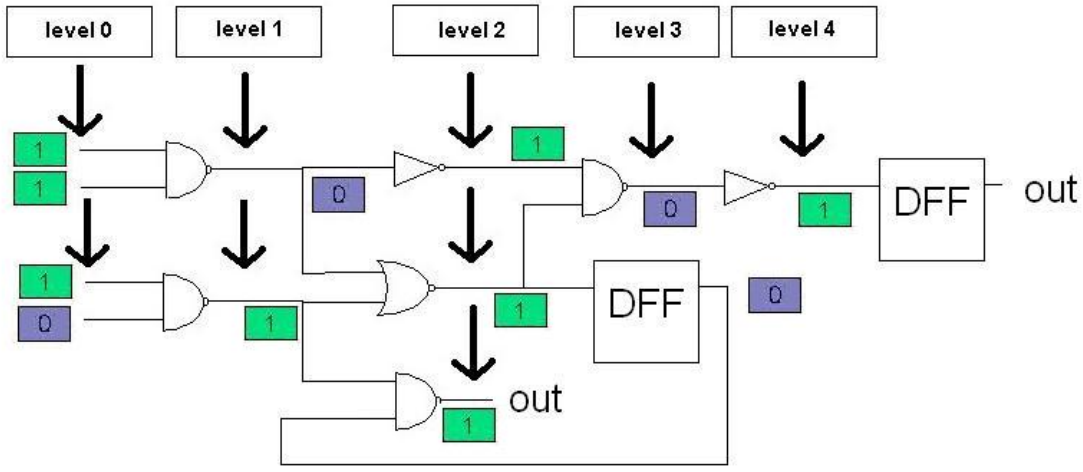
Η διαδικασία που περιγράψαμε παραπάνω αν και φαίνεται απλή, έχει πολλά βήματα που πρέπει να γίνουν και πολλές λεπτομέρειες που μπορεί να επηρεάσουν όχι μόνο την ορθότητα των αποτελεσμάτων αλλά και να αυξήσουν δραματικά το χρόνο που χρειάζεται η προσομοίωση. Αναφερόμαστε στο χρόνο προσομοίωσης γιατί το συγκεκριμένο είδος προσομοίωσης ήταν και το πιο χρονοβόρο από τα τρία συνολικά που κάνει το πρόγραμμα και η βελτίωση του χρόνου προσομοίωσης ήταν πρόβλημα. Για αυτό το λόγο έγιναν κάποιες τροποποιήσεις στο τρόπο που δουλεύει ο προσομοιωτής.

Για να μπορέσει να δουλέψει ο προσομοιωτής χρειάστηκε να προσθέσουμε κάποια πράγματα. Έτσι σε κάθε κόμβο μπορούμε να αποθηκεύσουμε περισσότερες από μια καταστάσεις. Η μια είναι σταθερή κατάσταση και η άλλη η κατάσταση του κόμβου κατά την διάδοση του upset. Έτσι μπορούμε να κάνουμε προσομοιώσεις έχοντας στο κύκλωμα μας μόνιμα αποθηκευμένη την κανονική του κατάσταση και με τον βοηθητικό καταχωρητή κάθε κόμβου να διαδίδουμε το event χωρίς να χάνουμε την πληροφορία της αρχικής κατάστασης του κυκλώματος.

Αναφέραμε πριν ότι κριτήριο για το αν επηρεάζει το SEU την λειτουργία του κυκλώματος, είναι το αν αλλάζει κατάσταση κόμβος που αποτελεί είσοδο DFF. Επομένως το υπό εξέταση κύκλωμα μας έχει τουλάχιστον ένα DFF.Κάθε φορά που έχουμε SEU σε έναν κόμβο αρχίζουμε να διαδίδουμε αυτό το event από τον κόμβο που ξεκίνησε έως ότου φτάσουμε και στο τελευταίο DFF του κυκλώματος. Αυτό το πράγμα μπορεί να γίνει με πολλούς τρόπους. Αυτός που χρησιμοποιήσαμε αρχικά ήταν ,αφού αλλάζουμε κατάσταση στον υπό εξέταση κόμβο, να αρχίσουμε να προσδιορίζουμε την κατάσταση όλων των κόμβων από το επόμενο level και μετά. Οι

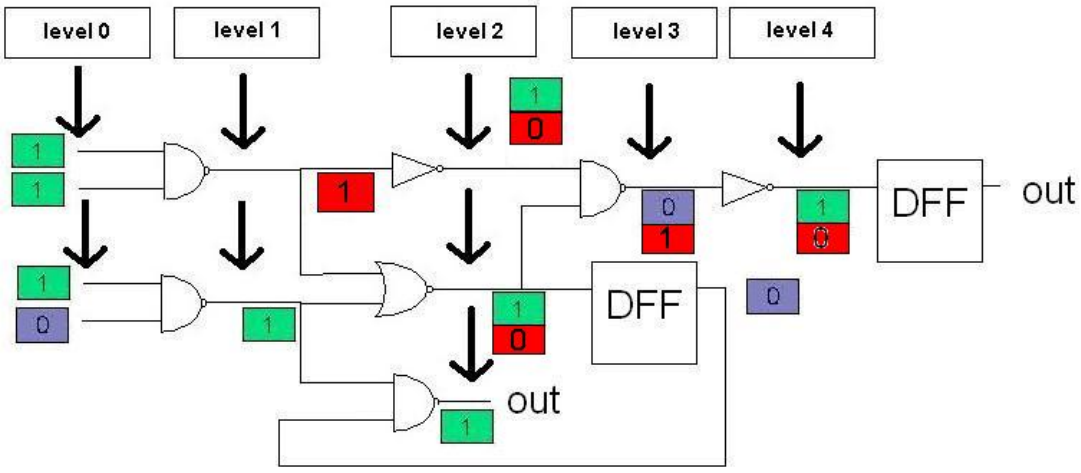
νέες καταστάσεις κάθε κόμβου αποθηκεύονται στην βοηθητική μεταβλητή κάθε κόμβου. Η διαδικασία αυτή συνεχίζεται έως ότου φτάσουμε στο τελευταίο level που έχει είσοδο DFF. Είναι φανερό ότι από εκεί και πέρα δεν έχει νόημα η συνέχιση της διάδοσης του event. Στην συνέχεια εξετάζουμε όλα τα DFF's και συγκεκριμένα τις εισόδους τους. Εάν βρούμε ότι η τιμή της κανονικής κατάστασης σε έστω και ένα κόμβο εισόδου είναι διαφορετική από αυτήν που βρίσκεται στον βοηθητικό καταχωρητή του, τότε θεωρούμε ότι το event αυτό επηρέασε την λειτουργία του κυκλώματος και μετράμε αυτό το λάθος στον αντίστοιχο μετρητή στον κόμβο.

Στο παρακάτω σχήμα φαίνεται ένα τέτοιο παράδειγμα μιας προσομοίωσης σε ένα απλό κύκλωμα. Ας υποθέσουμε ότι έχουμε το κύκλωμα του σχήματος 5.4 αρχικοποιημένο στην κατάσταση που φαίνεται στο σχήμα.



Σχήμα 5.4
Ένα ακολουθιακό κύκλωμα σε κατάσταση ηρεμίας

Κατά την διάρκεια της προσομοίωσης σε κάποιο κόμβο θα θεωρήσουμε ότι έχουμε ένα λανθασμένο event. Έστω ότι βρίσκεται στο level 1. Η διάδοση αυτού του event όπως γίνεται από το πρόγραμμα φαίνεται στο σχήμα 5.5. Κάτω από τις κανονικές τιμές των κόμβων βρίσκονται οι τιμές που παίρνουν μετά το event.



Σχήμα 5.5
Ένα ακολουθιακό κύκλωμα μετά την επίδραση ενός SEU

Τα αποτελέσματα που βγάζει το πρόγραμμα είναι αναλυτικά για κάθε κόμβο το ποσοστό επιρροής που έχει ένας αριθμός από SEUs στο κύκλωμα. Η μορφή των αποτελεσμάτων είναι όπως φαίνεται στο παρακάτω παράδειγμα.

S27

Signal name	Error frequency	
1	9073/10000	
2	4147/10000	
3	6605/10000	
4	1571/10000	
6	2253/10000	
7	10000/10000	input to DFF
8	2276/10000	
9	10000/10000	input to DFF
10	3129/10000	
11	10000/10000	input to DFF
12	9073/10000	
5	0/10000	Do not have any effect on the circuit
13	4546/10000	
14	3117/10000	
15	6234/10000	
16	2332/10000	
17	5473/10000	

Σε αυτό το κύκλωμα υπάρχουν 17 κόμβοι. Το πρόγραμμα δείχνει αναλυτικά τα αποτελέσματα για κάθε έναν από αυτούς. Μάλιστα εντοπίζει και εκείνους που δεν επηρεάζουν το κύκλωμα με τον τρόπο που περιγράφεται στην επόμενη παράγραφο καθώς επίσης και εκείνους που αποτελούν είσοδο σε κάποιο DFF.

5.3 Μέθοδοι βελτιστοποίησης χρόνου προσομοίωσης συμπεριφοράς κυκλώματος σε single event upset

Όπως αναφέραμε και πιο πάνω αυτή η τεχνική προσομοίωσης είναι η πιο χρονοβόρα από όλες που υλοποιεί ο προσομοιωτής. Για αυτόν τον λόγο προσπαθήσαμε να βρούμε τεχνικές που να βελτιώνουν όσο το δυνατόν την απόδοση του προγράμματος. Η πιο σημαντική είναι η ανίχνευση των κόμβων που δεν επηρεάζουν κανένα DFF του κυκλώματος. Σε αυτούς τους κόμβους δεν υπάρχει ούτε ένα μονοπάτι που να τους ενώνει με κάποιο από τα DFF's. Έτσι δεν υπάρχει λόγος να κάνουμε καμία μελέτη για αυτούς αλλά θεωρούμε απευθείας ότι το ποσοστό που επηρεάζουν σε SEU είναι μηδέν. Για να εντοπίσουμε αυτούς τους κόμβους χρησιμοποιούμε μια συνάρτηση η οποία τρέχει για κάθε DFF. Σε κάθε είσοδο DFF ξεκινάει και διαδίδεται προς τα πίσω μαρκάροντας όλους τους κόμβους από όπου περνάει. Ακολουθώντας την ίδια διαδικασία για κάθε είσοδο DFF θα έχουμε μαρκάρει στο τέλος όλους τους κόμβους οι οποίοι μπορούν να επηρεάσουν κάποια είσοδο. Οι κόμβοι που ζητάμε είναι όσοι δεν έχουν σημειωθεί από αυτόν τον αλγόριθμο. Έτσι όταν στην συνέχεια τρέξουμε την προσομοίωση στο κύκλωμα κάθε κόμβο που θα βλέπουμε χωρίς αυτό το σημάδι θα τον αγνοούμε και θα περνάμε στον επόμενο.

Μια δεύτερη τεχνική είναι ο εντοπισμός αυτών των κόμβων οι οποίοι είναι άμεσα συνδεδεμένοι με κάποια είσοδο DFF με τέτοιο τρόπο ώστε όποτε συμβαίνει σε αυτούς κάποιο SEU η είσοδος του DFF να αλλάζει πάντα. Αυτό γίνεται βασικά στους κόμβους που είναι από μόνοι τους είσοδοι σε κάποιο DFF ή οδηγούν κάποιον αντιστροφέα ο οποίος στην συνέχεια οδηγεί μια είσοδο DFF. Αυτοί οι κόμβοι θα επηρεάζουν το κύκλωμα σε ποσοστό 100% οπότε δεν υπάρχει λόγος να κάνουμε την προσομοίωση. Τέτοιοι κόμβοι είναι εύκολο να βρεθούν. Απλά μαρκάρουμε αντίστοιχα τις εισόδους των DFF's και ψάχνουμε για περιπτώσεις που κάποιος αντιστροφέας τις οδηγεί. Υπάρχουν περιπτώσεις όπου κόμβοι έχουν επιρροή 100% στην συμπεριφορά του κυκλώματος σε SEU χωρίς όμως αυτό να είναι φανερό από την αρχή. Η παραπάνω μέθοδος εντοπίζει μόνο τις περιπτώσεις που κάτι τέτοιο είναι φανερό.

5.4 Απαιτήσεις σε χρόνο

Η συγκεκριμένη προσομοίωση είναι η πλέον απαιτητική σε χρόνο από τις τρεις. Και αυτό γιατί για μια προσομοίωση χρειάζεται να αρχικοποιήσουμε το κύκλωμα και

στην συνέχεια για κάθε κόμβο να κάνουμε για διάσχιση του κυκλώματος για να βρούμε αν επηρεάζει ή όχι. Για να βγάλουμε όμως όσο το δυνατόν πιο κοντά στην πραγματικότητα αποτελέσματα χρειάζεται να κάνουμε έναν αρκετά μεγάλο αριθμό προσομοιώσεων σε ένα κύκλωμα της τάξης των δεκάδων χιλιάδων. Αυτό συνεπάγεται όμως τεράστια απαίτηση σε χρόνο. Ο παρακάτω πίνακας παρουσιάζει τις απαιτήσεις σε χρόνο του συστήματος μας για να κάνει ορισμένες προσομοιώσεις στα benchmark circuits.Οι χρόνοι είναι σε msec.

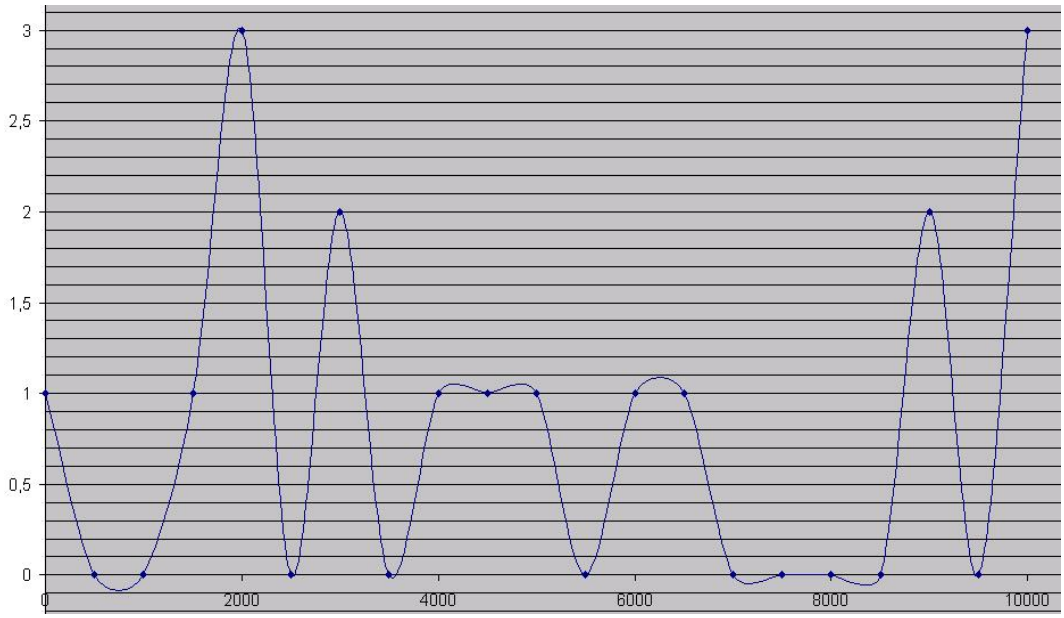
Κύκλωμα	Χρόνος SEU msec	No of tests
S27	375	10000
S208	3235	10000
S208_1	3219	10000
S298	7766	10000
S344	12312	10000
S349	12500	10000
S382	11688	10000
S386	9343	10000
S400	12563	10000
S420	7922	10000
S420_1	12297	10000
S444	15938	10000
S510	23219	10000
S526	22500	10000
S526N	22390	10000
S641	50406	10000
S713	6250	1000
S820	4422	1000
S832	4719	1000
S838	6547	1000
S838_1	5734	1000
S953	8828	1000
S1196	12093	1000
S1238	13656	1000
S1423	44750	1000
S1488	12062	1000
S1494	13406	1000
S5378	17127	10
S9234	75547	10
S9234_1	15578	2
S13207	23859	2

S15850	44266	2
S15850_1	43609	2
S35932	150313	2
S38417	272047	2
S38584	210703	2
S38584_1	212313	2

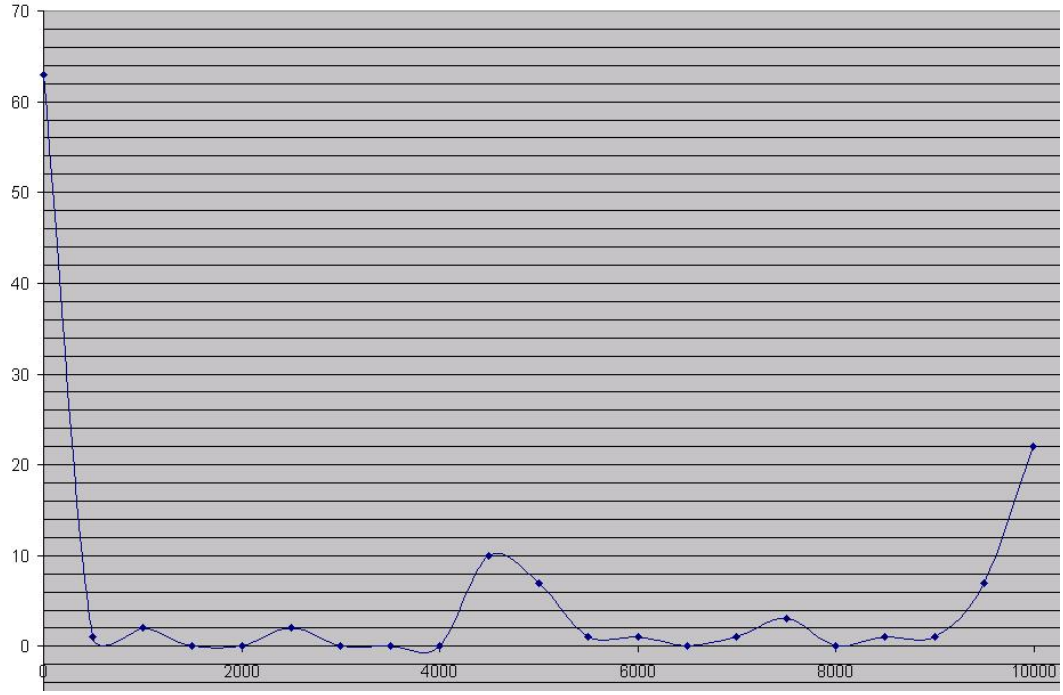
5.5 Αποτελέσματα

Από τα αποτελέσματα βγάλαμε και κάποιους πίνακες που δείχνουν πως κυμαίνονται τα ποσοστά επιρροής των κόμβων.

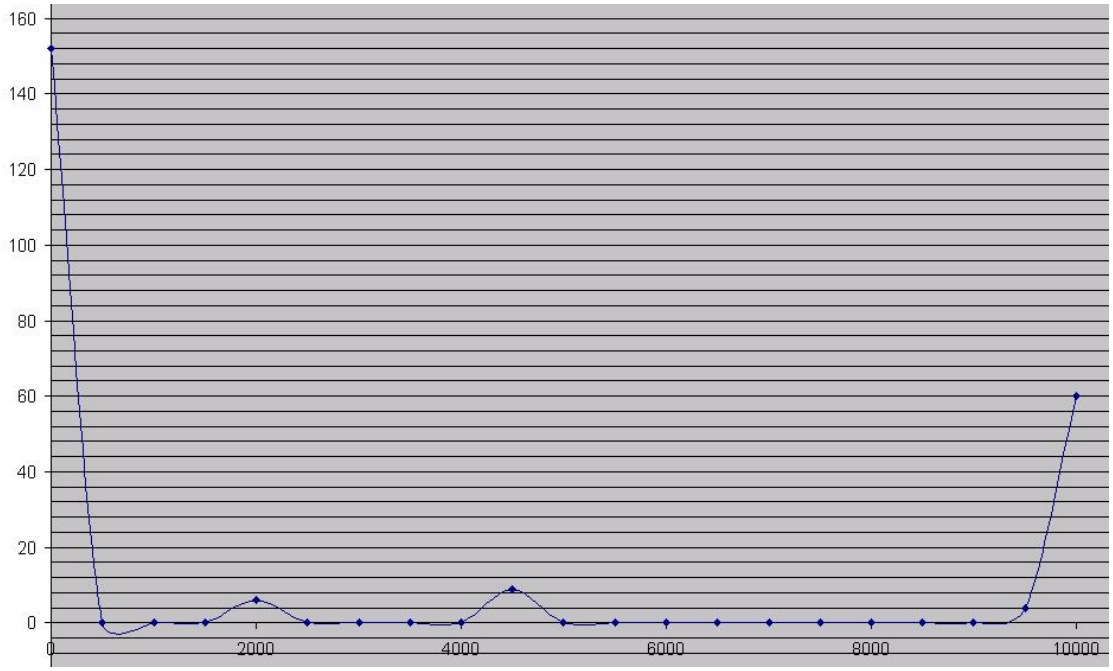
s27



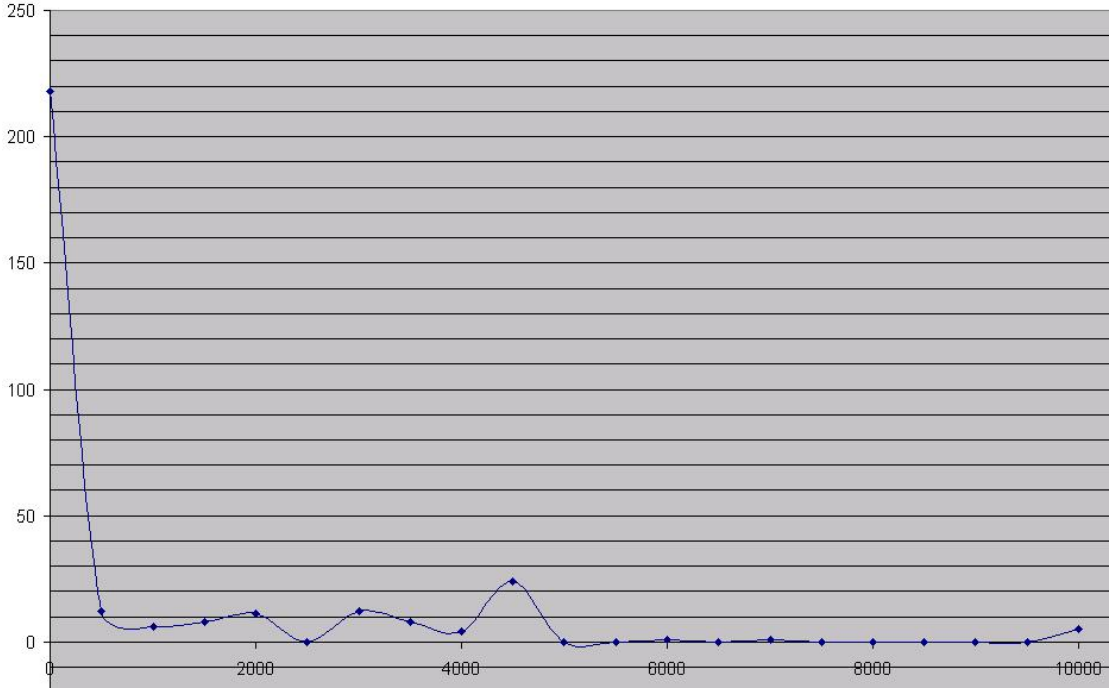
s208


70
60
50
40
30
20
10
0
0
2000
4000
6000
8000
10000

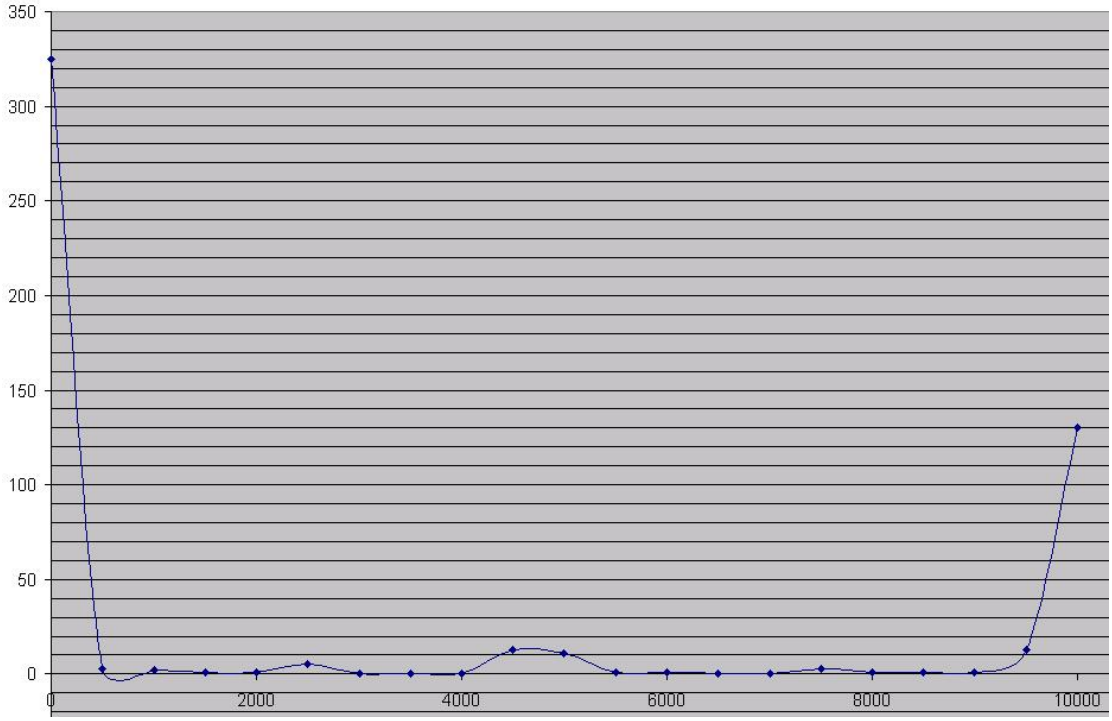
s420


160
140
120
100
80
60
40
20
0
0
2000
4000
6000
8000
10000

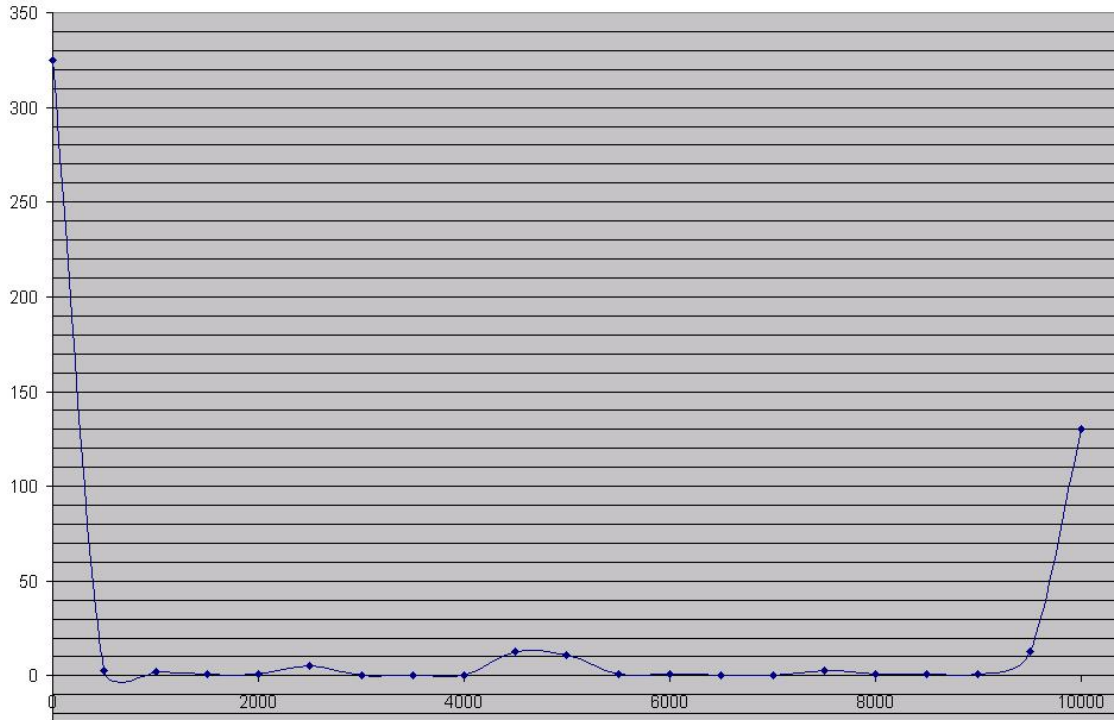
s832


250
200
150
100
50
0
0
2000
4000
6000
8000
10000

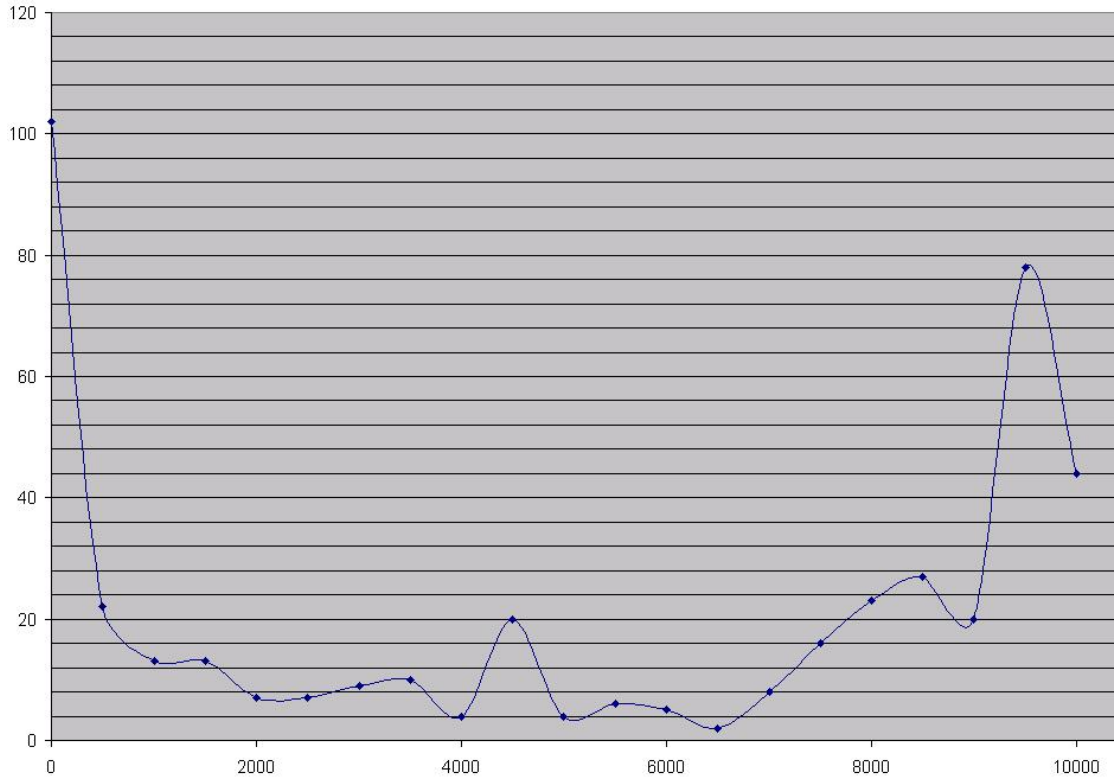
s838_1


350
300
250
200
150
100
50
0
0
2000
4000
6000
8000
10000

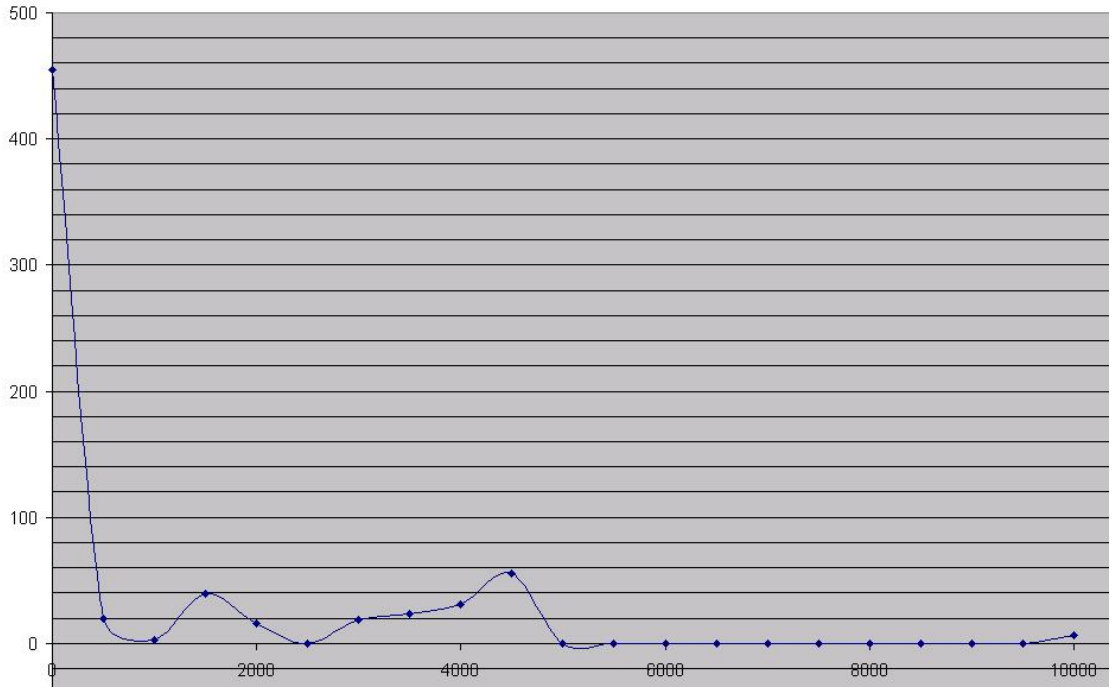
s838


350
300
250
200
150
100
50
0
0
2000
4000
6000
8000
10000

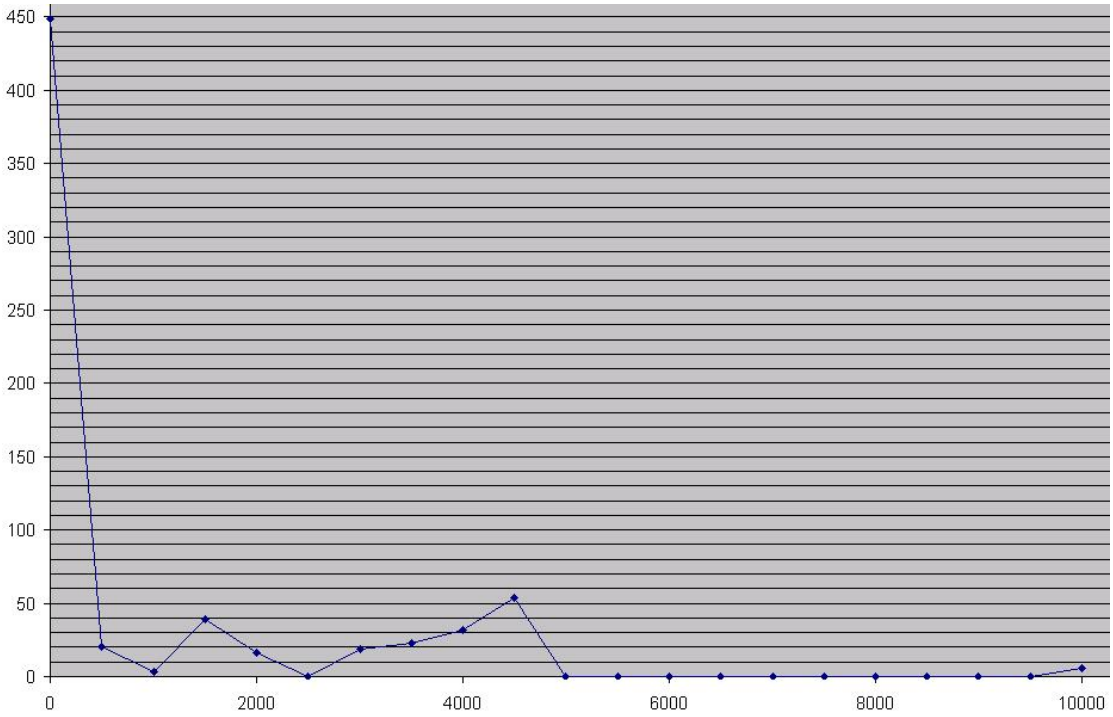
s953


120
100
80
60
40
20
0
0
2000
4000
6000
8000
10000

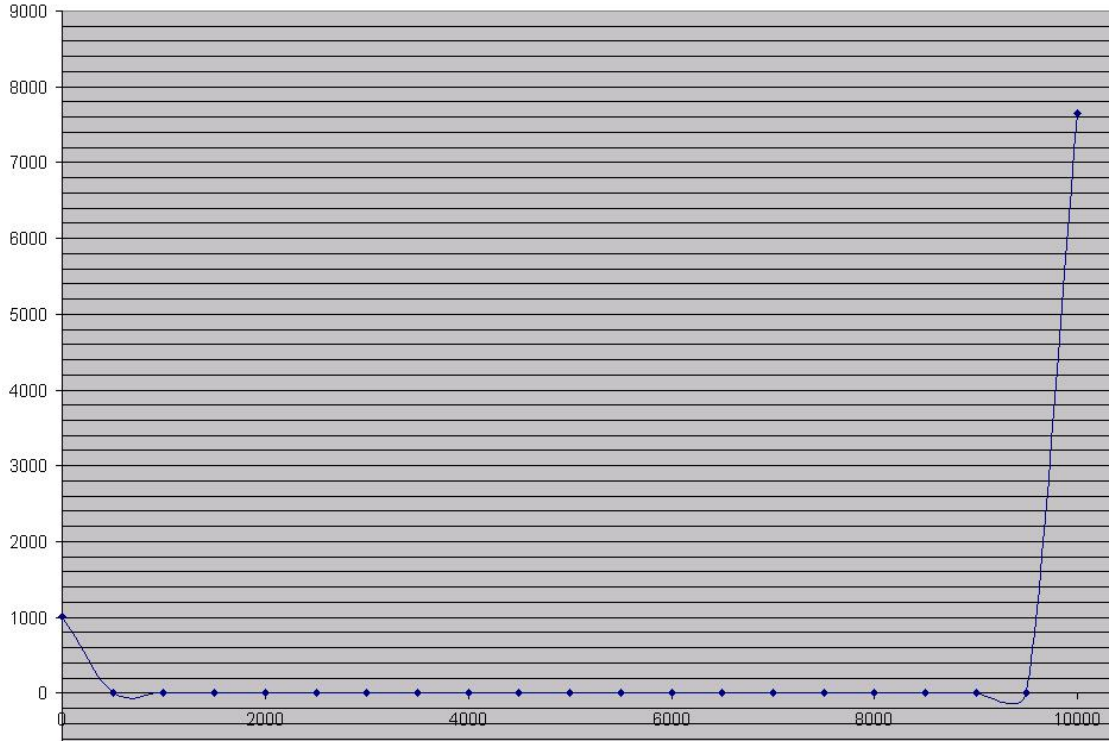
s1488


500
400
300
200
100
0
0
2000
4000
6000
8000
10000

s1494


450
400
350
300
250
200
150
100
50
0
0
2000
4000
6000
8000
10000

s13207_1


9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
2000
4000
6000
8000
10000

Κεφάλαιο 6ο
ΣΥΜΠΕΡΑΣΜΑΤΑ-ΕΡΕΥΝΗΤΙΚΕΣ ΚΑΤΕΥΘΥΝΣΕΙΣ

Το πρόγραμμα που φτιάξαμε αποτελεί ένα πολύ χρήσιμο εργαλείο για την μελέτη και ανάλυση ψηφιακών κυκλωμάτων. Η βάση του που είναι η δημιουργία ενός γράφου στην μνήμη για την προσομοίωση αποτελεί το σημαντικότερο του στοιχείο καθώς πάνω σε αυτό μπορούμε να χτίσουμε πάρα πολλές εφαρμογές και να επεκτείνουμε σε μεγάλο βαθμό τις δυνατότητες του. Χαρακτηριστικά αναφέρουμε ότι υπάρχει η δυνατότητα επέκτασης σε timing simulation με την κατασκευή τεχνικών time-wheel ενώ υπάρχει και η δυνατότητα electrical simulation καθώς η ύπαρξη του γράφου μας δίνει αυτήν την δυνατότητα.

Πέραν των όποιων επεκτάσεων υπάρχουν και πολλές δυνατότητες βελτιώσεις των ήδη υπαρχόντων τεχνικών προσομοίωσης που αναλύσαμε στα προηγούμενα κεφάλαια. Οι απαιτήσεις σε μνήμη , το memory location, οι χρόνοι πραγματοποίησης των προσομοιώσεων είναι κάποια από τα μέρη εκείνα του προγράμματος που μπορούν να δεχθούν βελτιώσεις έτσι ώστε μελλοντικά να έχουμε καλύτερα αποτελέσματα. Ωστόσο σε κάποιους τομείς ,όπως η μελέτη του SEU, τα πράγματα στον τομέα του simulation είναι σε αρχικό ακόμα στάδιο οπότε υπάρχουν ακόμα αρκετά περιθώρια μελέτης και βελτίωσης.

6.1 Κατευθύνσεις έρευνας

Η σημαντικότερη έρευνα που γίνεται πάνω σε αυτόν τον τομέα αποσκοπεί στην αντιμετώπιση προβλημάτων που δημιουργούνται κατά τις προσομοιώσεις καθώς και η ελαχιστοποίηση των απαιτήσεων σε υπολογιστική ισχύ και σε μνήμη. Συνοπτικά μπορούμε να εντοπίσουμε τα εξής σημεία που επικεντρώνεται η έρευνα.

- **Ελαχιστοποίηση του μεγέθους των δεδομένων της μνήμης.**

  Σε ένα κύκλωμα με εκατοντάδες χιλιάδες ή και εκατομμύρια τρανζίστορς πρέπει να βρούμε τρόπο να κρατήσουμε την μνήμη σε λογικά επίπεδα. Ο τρόπος που επιδιώκουμε να το πετύχουμε είναι με την ελαχιστοποίηση του μεγέθους των οντοτήτων που αναπαριστάνουν τα στοιχεία του κυκλώματος στον γράφο. Η εξοικονόμηση μέχρι και του τελευταίου bit είναι σημαντική καθώς για κάθε μια από αυτές τις δομές (τρανζιστορς, πύλες ,κόμβοι) θα έχουμε εκατομμύρια οντότητες στην μνήμη.

- **Καταμερισμός του κυκλώματος.**

  Είναι φανερό ότι σε ένα τεράστιο κύκλωμα είναι πολλές φορές δύσκολο να γίνει μια πλήρης προσομοίωση του. Για αυτό τον λόγο είναι θεμιτό να επιδιώκουμε τον καταμερισμό του κυκλώματος σε μικρότερα υποκυκλώματα με σκοπό την επιμέρους ανάλυση τους. Με αυτό τον τρόπο μπορεί όχι μόνο η προσομοίωση να γίνεται ευκολότερη αλλά και στην σχεδίαση ενός κυκλώματος ο εντοπισμός ατελειών να γίνεται πιο γρήγορα και αποτελεσματικότερα. Ο καταμερισμός όμως παίζει σημαντικό ρόλο και στην ταχύτητα με την οποία γίνεται η προσομοίωση.

- **Memory locality**
  Επειδή μια προσομοίωση βασίζεται κυρίως σε διασχίσεις γράφων είναι συχνό το φαινόμενο να μην εκμεταλλευόμαστε επαρκώς την cache αλλά θα χρειάζεται να έχουμε συνεχώς ανανεώσεις των δεδομένων από την μνήμη. Έτσι μπορεί οι ταχύτητες των CPUs να έχουν ξεφύγει αλλά ο συνολικός χρόνος των προσομοιώσεων θα μένει σε υψηλά επίπεδα. Για αυτόν τον λόγο γίνεται επιτακτική η ανάγκη διάθρωσης των δεδομένων στην μνήμη έτσι ώστε να μειώσουμε τον αριθμό προσβάσεων σε αυτην και να βελτιώσουμε τους χρόνους προσομοίωσης.

6.2 Μελλοντικές δυνατότητες επέκτασης του προσομοιωτή

Όπως αναφέραμε στην αρχή του κεφαλαίου το μεγάλο πλεονέκτημα του προγράμματος μας είναι η δυνατότητα να δημιουργεί έναν γράφο αναπαράσταση του κυκλώματος στην μνήμη. Σε αυτό τον γράφο έχει την δυνατότητα να πραγματοποιεί προσομοιώσεις οι οποίες να αναπαριστούν σε μεγάλο βαθμό και με αρκετά μεγάλη ακρίβεια την πραγματική κατάσταση που μπορεί να επέλθει το κύκλωμα μας. Έτσι χάρη σε αυτή την ιδιότητα μπορούμε να χτίσουμε ένα πολύ μεγάλο αριθμό εφαρμογών και διαφορετικών στρατηγικών προσομοιώσεων. Μπορεί μέχρι τώρα οι προσομοιώσεις να γίνονται σε switch level αλλά υπάρχει η δυνατότητα επέκτασης σε timing simulation. Βέβαια ο λόγος που προτιμήσαμε αυτή την τεχνική είναι η ταχύτητα προσομοίωσης αλλά και η ακρίβεια των αποτελεσμάτων πάνω στην μελέτη κατανάλωσης ισχύος. Ωστόσο πάνω στον γράφο που έχουμε κατασκευάσει μπορούν να εφαρμοστούν στρατηγικές που υλοποιούν time wheel για να έχουμε και εκτιμήσεις χρονικής καθυστέρησης. Επίσης μια τεχνική μπορεί να είναι η εύρεση του critical path. Μια περαιτέρω βελτίωση του προγράμματος θα ήταν και η υλοποίηση τεχνικών για electrical simulation αν και τότε τα αποτελέσματα που θα παίρναμε θα είχαν περισσότερο χαρακτήρα μελέτης χρόνου και λιγότερο ενέργειας.

Βιβλιογραφία

[1] S. Kang and Y. Leblebici, **CMOS Digital Integrated Circuits**, 2nd ed., McGraw-Hill, 1999.

[2] T. Fjeldly and M. Shur, "**Threshold voltage modeling and the subthreshold regime of operation of short-channel MOSFETs**", IEEE Trans. Electron Devices, vol. 40, pp. 137-145, 1993.

[3] W. Maly and M. Patyra, "**Design of ICs applying built-in current testing**", J. Electronics Testing Theory and Applications, vol. 3, pp. 397-406, 1992.

[4] A. Keshavarzi, K. Roy, and C. Hawkins, "**Intrinsic I_{DDQ} : origins, prediction, and applications in deep sub-μm low power CMOS ICs**", IEEE Int. Test Conf., 1997.

[5] P. Maxwell and J. Rearick, "**Estimation of defect-free I_{DDQ} in submicron circuits using switch level simulator**", IEEE Int. Test Conf., 1998.

[6] J. Galambos, **The Asymptotic Theory of Extreme Order Statistics**, 2nd ed., Krieger, 1987.

[7] Z. Chen, M. Johnson, L. Wei, and K. Roy, "**Estimation of standby leakage power in CMOS circuits considering accurate modeling of transistor stacks**", ACM/IEEE Int. Symp. Low Power Electronics and Design, 1998.

[8] M. Johnson, D. Somasekhar, and K. Roy, "**Models and algorithms for bounds on leakage in CMOS circuits**", IEEE Trans. Computer-Aided Design, vol. 18, pp. 714-725, 1999.

[9] A. Ferre and J. Figueras, "**Leakage power bounds in CMOS digital technologies**", IEEE Trans. Computer-Aided Design, vol. 21, pp. 731-738, 2002.

[10] N. Evmorfopoulos, G. Stamoulis, and J. Avaritsiotis, "**A Monte Carlo approach for maximum power estimation based on extreme value theory**", IEEE Trans. Computer-Aided Design, vol. 21, pp. 415-432, 2002.

[11] S. Resnick, **Extreme Values, Regular Variation and Point Processes, Springer**, 1987.

[12] I. Ibragimov and Y. Linnik, **Independent and Stationary Sequences of Random Variables**, Wolters-Noordhoff, 1971.

[13] E. Castillo, **Extreme Value Theory in Engineering**, Academic Press, 1988.

[14] Q. Wu, Q. Qiu, and M. Pedram, "**Estimation of peak power dissipation in VLSI circuits using the limiting distributions of extreme order statistics**", IEEE Trans. Computer-Aided Design, vol. 20, pp. 942-956, 2001.

[15] C. Rao, **Linear Statistical Inference and its Applications**, 2nd ed., Wiley, 1973.

[16] S. Resnick, "**Tail equivalence and its applications**", J. Applied Probability, vol. 8, pp. 136-156, 1971.

[17] M. Abramowitz and I. Stegun, Handbook of Mathematical Functions, Dover, 1964.

[18] R. Gu and M. Elsmary, "**Power dissipation analysis and optimization of deep submicron CMOS digital circuits**", IEEE J. Solid State Circuits, vol. 31, pp. 707-713, 1996.
[19] A. Hill, C. Teng, and S. Kang, "**Simulation-based maximum power estimation**", IEEE Int. Symp. Circuits and Systems, 1996.
[20] C. Ding, Q. Wu, C. Hsieh, and M. Pedram, "**Statistical estimation of the cumulative distribution function for power dissipation in VLSI circuits**", ACM/IEEE Design Automation Conf., 1997.
[21] C. Wang and K. Roy, "**Maximum power dissipation for CMOS circuits using deterministic and statistical approaches**", IEEE Trans. VLSI Systems, vol. 6, pp. 134-140, 1998.
[22] R. Fletcher, Practical **Methods of Optimization**, 2nd ed., Wiley, 1987.
[23] **VHDL for Designers**, Stefan Sjoholm and Lennart Lindh.
[24] **Fast timing simulation of MOS VLSI circuits** by David Vincent Overhauser
[25] **System effects of single event upsets** by A. M. Finn
[26] **Tutorial: Soft errors included by Alpha particles**
[27] **A Systematic Approach to Processor SER Estimation and Solutions b**y Hang T. Nguyen

E-books:

[1]. **Model*SIM* SE** Tutorial.
[2]. **Model*SIM* SE** Manual.
[3]. **Design Compiler™,** Reference Manual: Constraints and Timing, Version 2001.08, August 2001.
[4]. **Design Compiler™,** Command-Line Interface Guide, Version 2001.08, August 2001.
[5]. **Design Compiler™,** Tutorial, Version 2001.08, August 2001.
[6]. **Design Compiler™,** User Guide, Version 2001.08, August 2001.
[7]. **Power Compiler™,** Quick Reference, Version 2001.08, August 2001.
[8]. **Power Compiler™,** User Guide, Version 2001.08, August 2001.
[9]. **Power Compiler™,** Reference Manual, Version 2001.08, August 2001
[10]. **DesignWare™,** User Guide, Version 2001.00, August 2001.
[11]. **UMC *e*Si-Route/9™ High Density Standard Cell Library Datasheet**, Part Number: UMCL13U210T3, Revision 2.5, May, 2002.
[12]. **UMC eSi-Route/11™ Standard Cell Library Datasheet**. Part Number: UMCL25U250T3, Revision 1.2, February, 2002.
[13]. **UMC eSi-Route/11™ High Performance 0.18µ Standard Cell Library.** Part Number: UMCL18U250, Rev. 2.1, January, 2001.
[14]. **O'REILLY – Perl IN A NUTSHELL**, Ellen Siever, Stephen Spainhour & Nathan Patwardhan,First Edition, December 1998.

[15]. **O'REILLY – Learning Perl**, By Randal Schwartz, Tom Christiansen & Larry Wall; Second Edition, July 1997.
[16]. **O'REILLY – Perl Cookbook**, By Tom Christiansen & Nathan Torkington; First Edition, August 1998.