ΕΛΕΓΧΟΣ ΟΡΘΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΜΝΗΜΩΝ DRAM ΜΕ ΧΡΗΣΗ ΤΩΝ NPSF ΚΑΙ NWSF ΜΟΝΤΕΛΩΝ ΣΦΑΛΜΑΤΩΝ: Ο Δ-ΤΥΠΟΣ ΓΕΙΤΟΝΙΑΣ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από τον

Γεώργιο Σφήκα

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΙΣ ΤΕΧΝΟΛΟΓΙΕΣ-ΕΦΑΡΜΟΓΕΣ

Ιανουάριος 2009

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΕΥΡΕΤΗΡΙΟ ΑΛΓΟΡΙΘΜΩΝ

ΠΕΡΙΛΗΨΗ

Γεώργιος Σφήκας του Ιωάννη και της Βασιλικής. MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιανουάριος 2009. Έλεγχος ορθής λειτουργίας μνημών DRAM με χρήση των NPSF και NWSF μοντέλων σφαλμάτων: ο Δ- τύπος γειτονιάς.
Επιβλέποντας: Γεώργιος Τσιατούχας.

Η πρόοδος της μικροηλεκτρονικής αυξάνει συνεχώς τις δυνατότητες των ψηφιακών ολοκληρωμένων κυκλωμάτων αυξάνοντας όμως παράλληλα και το βαθμό δυσκολίας του *Ελέγχου Ορθής Λειτουργίας (Ε.Ο.Λ.)* τους. Οι μνήμες δεν αποτελούν εξαίρεση σε αυτόν τον κανόνα. Η δυσκολία στον έλεγχο ορθής λειτουργίας στις σύγχρονες μνήμες οφείλεται σε μεγάλο βαθμό στο γεγονός ότι η συνεχής σμίκρυνση των διαστάσεων των στοιχειωδών αποθηκευτικών μονάδων, που καλούνται κύτταρα μνήμης (memory cells), έχει σαν αποτέλεσμα οι αλληλεπιδράσεις μεταξύ τους να επηρεάζουν όλο και περισσότερο τη λειτουργία της μνήμης. Ένα ρεαλιστικό μοντέλο που περιγράφει τις αλληλεπιδράσεις μεταξύ γειτονικών κυττάρων είναι το NPSF (Neighborhood Pattern Sensitive Fault) μοντέλο σφαλμάτων μνημών. Σε μία απλή πρώτη προσέγγιση ένας αλγόριθμος Ε.Ο.Λ. με βάση αυτό το μοντέλο ελέγχει τη συμπεριφορά κάθε κυττάρου μνήμης για όλες τις δυνατές τιμές και μεταβάσεις τιμών των ‘γειτονικών’ του κυττάρων. Η σωστή επιλογή των κυττάρων που θα θεωρήσουμε ως ‘γειτονικά’ σε αυτό το κύτταρο, έτσι ώστε να λάβουμε υπόψιν τα κύτταρα που μπορούν πραγματικά να το επηρεάσουν και μόνο αυτά, είναι προφανώς κρίσιμης σημασίας. Στην παρούσα εργασία προσαρμόζουμε έναν διαδεδομένο τύπο γειτονιάς, τη γειτονιά τύπου – 1, στην πραγματική τοπολογία της DRAM μνήμης και στη συνέχεια προτείνουμε έναν νέο τύπο γειτονιάς, τη γειτονιά τύπου – Δ με στόχο τη μείωση του κόστους Ε.Ο.Λ. Ακολούθως προτείνουμε ένα νέο μοντέλο σφαλμάτων, το NWSF (Neighborhood Word –line Sensitive Fault) και αναπτύσσουμε μεθόδους ανίχνευσης των σφαλμάτων που αυτό παράγει σε συνδυασμό με τα NPSF σφάλματα. Σε κάθε περίπτωση αναπτύσσουμε τους αντίστοιχους αλγορίθμους τόσο για μνήμες λειτουργίας bit (bit oriented) όσο και για μνήμες λειτουργίας λέξης. (word oriented)

EXTENDED ABSTRACT IN ENGLISH

Yiorgos Sfikas, Y.S. MSc, Computer Science Department, University of Ioannina, Greece. January 2009. DRAM Memory Testing using the NPSF and the NWSF fault models; the Δ – type neighborhood.

Thesis Supervisor: Yiorgos Tsiatouhas.

The rapid progress in microelectronics in the last decades has increased dramatically the capabilities of the digital integrated circuits. However, the difficulty in testing the produced integrated circuits has increased dramatically as well. The memories are not an exception to this rule, since the huge augmentation in their capacity has raised the need for more efficient and reliable testing methods. The difficulty in memory testing comes mainly as a consequence to the high density of their storage elements, called *memory cells*, which makes the interactions between them a factor that strongly affects the operation of the memory.

The most realistic and well established fault model that describes the susceptibility of a memory cell to its adjacent cells is the *Neighborhood Pattern Sensitive Fault (NPSF)* model. According to this, the content of a cell or the ability to apply a desired value at that cell, is affected by the values or transitions on the values of k neighbor cells in the memory array. In practice the cells (called *deleted neighborhood*) affecting the operation of a cell (called *base cell*) are those with physical proximity to that particular cell. The combination of the base cell and the deleted neighborhood is called *neighborhood* while the corresponding faults are called *Neighborhood Pattern Sensitive Faults* (*NPSFs*).

In order to produce an efficient testing method using this fault model, it is necessary to determine which cells are more likely to affect the base cell, in order to define a valid neighbourhood type. The most common neighborhood types that have been proposed are the Type – 1 and Type – 2 which use the criterion of the physical proximity to the base cell. Thus, the Type – 1 includes the 4 cells that are the closest to the base cell and therefore it is a 5 cell

neighborhood. The Type – 2 on the other hand includes the 8 closest to the base cell cells and therefore it is a 5 cell neighborhood.

However, none of these neighborhood types take into consideration the physical arrangement of the memory cells but only the logical one. Therefore, they perceive the memory cells as cells in a 2 - dimensional matrix when they determine the neighborhood of the base cell. As a result, they fail to include the cells that are closest to the base cell in the neighbourhood since the most common memory layout is much different than a 2-D matrix. In addition, the associated test application time cost is prohibitive for Type – 2 neighborhood and quite unattractive for Type – 1.

In this work we initially adapt the Type – 1 neighborhood on the physical layout of the memory, creating a new neighborhood type called the *Adapted Type – 1 Neighborhood*. Afterwards, we exclude one of the cells included in the Adapted Type – 1 Neighborhood, claiming that it cannot affect the base cell. We therefore create a new 4 cell neighbourhood called the *Delta – Type* (or *Δ – Type*) Neighborhood, which has a drastically decreased cost in test application time compared to the Type – 1, and we develop the corresponding algorithm. Finally, we introduce a new fault model called the *Neighborhood Word – Line Sensitive Fault – NWSF*. This model takes into consideration the strong coupling between the adjacent conductible lines that activate the memory cells, called *Word – Lines*. Following this step, we enhance the Δ – Type algorithm in order to be able to detect both NPSF and NWSF faults in various combinations. In all cases mentioned above we develop algorithms suitable for both bit – oriented and word oriented memories, maintaining the test application time significantly lower than the Type – 1 Neighborhood.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

1.1. Έλεγχος Σφαλμάτων σε Ψηφιακά Κυκλώματα και Μνήμες

Ο έλεγχος σφαλμάτων σε ψηφιακά κυκλώματα είναι ένας κρίσιμης σημασίας χώρος έρευνας, με μεγάλες δυνατότητες εξέλιξης και εφαρμογής νέων ιδεών. Η ολοένα αυξανόμενη πολυπλοκότητα των ψηφιακών ολοκληρωμένων κυκλωμάτων έχει σαν αποτέλεσμα ο έλεγχος ορθής λειτουργίας τους μετά την κατασκευή τους και πριν τη διοχέτευσή τους στην αγορά να γίνεται συνεχώς όλο και πιο δύσκολη, χρονοβόρα και συνεπώς υψηλού κόστους διαδικασία. Για το λόγο αυτό η συνεχής εξέλιξη των αλγορίθμων που εντοπίζουν σφάλματα σε ολοκληρωμένα κυκλώματα, ώστε αυτοί να παραμένουν αποτελεσματικοί και γρήγοροι είναι κρίσιμης σημασίας στις σύγχρονες CMOS νανοτεχνολογίες.

Ένας σημαντικός τομέας της έρευνας αυτής ασχολείται με τον έλεγχο σφαλμάτων σε μνήμες. Οι μνήμες, παρόλο που αποτελούν διατάξεις στις οποίες η δομή είναι επαναλαμβανόμενη και από αυτή την άποψη η πολυπλοκότητα που παρουσιάζουν είναι σχετικά μικρή, δεν ξεφεύγουν από τον κανόνα της ολοένα αυξανόμενης δυσκολίας στον έλεγχο ορθής λειτουργίας με την πρόοδο της τεχνολογίας, η οποία επιβάλλει όλο και μεγαλύτερη χωρητικότητα σε περιορισμένη επιφάνεια πυριτίου. Ο λόγος που υπάρχει αυτή η δυσκολία είναι ότι η συνεχής σμίκρυνση των διαστάσεων των κυττάρων μνήμης (memory cells) καθώς και των μεταξύ τους αποστάσεων κάνει όλο και σημαντικότερο το ρόλο που παίζουν οι μεταξύ τους αλληλεπιδράσεις. Έτσι, δεν αρκεί να ελέγξει κανείς το κάθε κύτταρο μνήμης ξεχωριστά για να αποφανθεί αν η μνήμη λειτουργεί σωστά. Αντίθετα, χρειάζεται να δει κανείς το κύτταρο σαν μέλος ενός συνόλου, όπου δέχεται την επίδραση των αποθηκευμένων τιμών και των μεταβάσεων των υπολοίπων κυττάρων και κατά αντίστοιχο τρόπο επηρεάζει

και αυτό το σύνολο με την τιμή που έχει αποθηκευμένη σε μια δεδομένη χρονική στιγμή και με τις μεταβάσεις του.

Ακολουθώντας με τις παραπάνω παρατηρήσεις το μοντέλο που χρησιμοποιούμε στην παρούσα εργασία, το Μοντέλο Σφαλμάτων Εξάρτησης από το Διάνυσμα Τιμών της Γειτονιάς (Neighborhood Pattern Sensitive Fault - NPSF), αν και πολύ παλιό, τα τελευταία χρόνια αποκτά όλο και μεγαλύτερη αξία γιατί εξετάζει ακριβώς την αλληλεπίδραση μεταξύ των κυττάρων της μνήμης. Εντούτοις, όπως φαίνεται εξακολουθεί να μην είναι ελκυστικό, εξ αιτίας του μεγάλου κόστους σε χρόνο εκτέλεσης που έχουν οι αντίστοιχοι αλγόριθμοι. Συνεπώς, οι σύγχρονες ανάγκες απαιτούν αλγορίθμους που διατηρούν ή και ενισχύουν την αποτελεσματικότητα του NPSF μοντέλου, μειώνοντας όμως το χρόνο εκτέλεσης του σχετικού αλγορίθμου ελέγχου.

1.2. Στόχοι

Στόχος της διατριβής είναι αφ' ενός η ανάπτυξη μεθόδων ελέγχου σφαλμάτων σύμφωνα με το NPSF μοντέλο, οι οποίες θα είναι προσαρμοσμένες στην αρχιτεκτονική των μνημών DRAM και αφ' ετέρου η εισαγωγή ενός νέου μοντέλου σφαλμάτων σε DRAM, που φέρει το όνομα Μοντέλο Σφαλμάτων Αλληλεπίδρασης Γειτονικών Word - Lines (Neighborhood Word - Line Sensitive Fault - NWSF), και ανάπτυξη των σχετικών μεθόδων ελέγχου ορθής λειτουργίας. Πιο αναλυτικά, οι επιμέρους στόχοι είναι οι ακόλουθοι:

- Να αναλυθεί η επικρατέστερη μέθοδος ελέγχου σφαλμάτων σύμφωνα με το NPSF μοντέλο σε μνήμες DRAM.
- Να προσαρμοστεί αυτή η μέθοδος λαμβάνοντας υπ’ όψιν τον φυσικό σχεδιασμό των DRAM μνημών ώστε να εξασφαλίζει υψηλή κάλυψη σφαλμάτων.
- Προεκτείνοντας την προηγούμενη προσέγγιση του φυσικού σχεδιασμού, να προταθεί και να αναλυθεί μία νέα μέθοδος η οποία θα καλύπτει τα σφάλματα του NPSF μοντέλου σε μικρότερο χρόνο.
- Να εισαχθεί ένα νέο μοντέλο σφαλμάτων, το NWSF, και να προταθεί μια νέα μέθοδος που θα καλύπτει αυτά τα σφάλματα σε συνδυασμό με εκείνα του NPSF μοντέλου.
- Στις δύο παραπάνω μεθόδους να αναλυθεί και να λυθεί το πρόβλημα της απόκρυψης σφαλμάτων όταν έχουμε μνήμη λειτουργίας λέξης.
- Περιγραφή των αντιστοίχων αλγορίθμων ελέγχου.

Στην παρούσα εργασία παρουσιάζεται αρχικά η προσαρμογή μίας υπάρχουσας μεθόδου ελέγχου που βασίζεται στο NPSF μοντέλο έτσι ώστε αυτός να είναι πιο αποτελεσματικός στις μνήμες DRAM εμφωλευμένης συστοιχίας (folded array) που είναι οι πιο διαδεδομένες στις σύγχρονες τεχνολογίες. Στη συνέχεια, εισάγεται το πρόβλημα της απόκρυψης σφαλμάτων κατά τον έλεγχο ορθής λειτουργίας DRAM μνημών λειτουργίας λέξης με τη χρήση του NPSF μοντέλου σφαλμάτων και προτείνεται μέθοδος επίλυσής του. Ακολούθως, με βάση το φυσικό σχεδιασμό του υπό εξέταση τύπου μνήμης προτείνουμε ένα νέο αλγόριθμο ελέγχου ορθής λειτουργίας για το NPSF μοντέλο που παρουσιάζει σημαντική μείωση στο χρόνο ελέγχου σε σχέση με τον προϋπάρχοντα αλγόριθμο στη διεθνή βιβλιογραφία. Τέλος, εισάγουμε έναν νέο τύπο σφάλματος (τον τύπο NWSF) που ενδέχεται να παρουσιαστεί σε μνήμες DRAM και τροποποιούμε τον αλγόριθμό μας έτσι ώστε να είναι σε θέση να εντοπίσει αυτά τα σφάλματα από κοινού με τα σφάλματα NPSF.

1.3. Δομή της Διατριβής

Η διατριβή αποτελείται από 6 κεφάλαια:

Το Κεφάλαιο 1 είναι εισαγωγικό στο θέμα της διατριβής.

Στο Κεφάλαιο 2 δίδεται αναλυτική περιγραφή της αρχιτεκτονικής, του φυσικού σχεδιασμού και του τρόπου λειτουργίας DRAM μνημών εμφωλευμένης συστοιχίας (folded array).

Στο Κεφάλαιο 3 γίνεται παρουσιάζονται τα μοντέλα σφαλμάτων μνημών και ειδικότερα του NPSF μοντέλου σφαλμάτων καθώς και των επικρατέστερων μεθόδων ελέγχου ορθής λειτουργίας σύμφωνα με αυτό το μοντέλο.

Στο Κεφάλαιο 4 προσαρμόζουμε την επικρατέστερη μέθοδο ελέγχου σφαλμάτων σύμφωνα με το NPSF μοντέλο στο φυσικό σχεδιασμό της μνήμης DRAM εμφωλευμένης συστοιχίας.

Στο Κεφάλαιο 5 προτείνουμε μία νέα μέθοδο με στόχο να μειώσουμε το χρόνο εκτέλεσης του αλγορίθμου ελέγχου, διατηρώντας την αποτελεσματικότητα της προηγούμενης μεθόδου. Επίσης εισάγουμε το νέο μοντέλο σφαλμάτων, το NWSF, και προσαρμόζουμε τη μέθοδό μας ώστε να καλύπτει και αυτόν τον τύπο σφαλμάτων.

Στο Κεφάλαιο 6 αναλύουμε τα συμπεράσματα και προτείνουμε πιθανές κατευθύνσεις μιας μελλοντικής ερευνητικής δραστηριότητας.

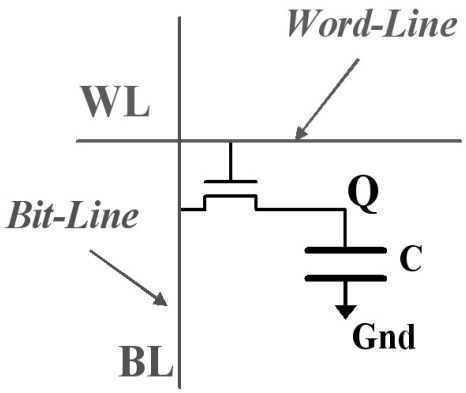
ΚΕΦΑΛΑΙΟ 2. ΔΟΜΗ ΚΑΙ ΛΕΙΤΟΥΡΓΙΑ ΤΩΝ ΜΝΗΜΩΝ DRAM

2.1. Δομή και Λειτουργία των DRAM Μνημών

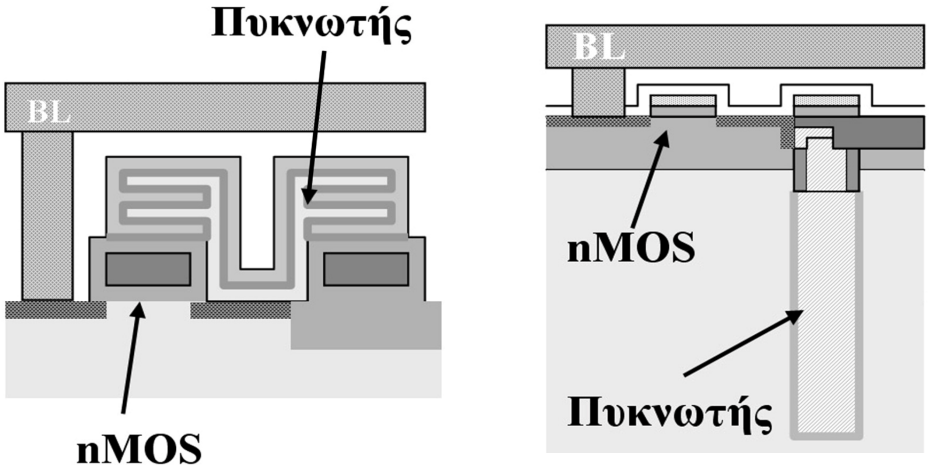
Η μνήμη σε γενικές γραμμές είναι μία διάταξη στοιχειωδών αποθηκευτικών μονάδων. Η διάταξη αυτή μπορεί να ειδωθεί με τη μορφή πίνακα. Η στοιχειώδης αποθηκευτική μονάδα μπορεί να αποθηκεύσει ένα bit και ονομάζεται *κύτταρο μνήμης (memory cell)*.

Το κύτταρο μνήμης μιας μνήμης DRAM φαίνεται στο Σχήμα 3.1. Αποτελείται από έναν πυκνωτή και ένα *τρανζίστορ διέλευσης* το οποίο απομονώνει ή συνδέει τον πυκνωτή με μία αγώγιμη γραμμή η οποία ονομάζεται *Bit – Line (BL)***.** Στον πυκνωτή κατακρατείται με τη μορφή ηλεκτρικού φορτίου η λογική τιμή που θέλουμε να αποθηκεύσουμε. Συνήθως η λογική τιμή 0 αντιστοιχεί σε τάση 0 Volt στον πυκνωτή ενώ η λογική τιμή 1 αντιστοιχεί στην τάση τροφοδοσίας V_{DD} της μνήμης. Το τρανζίστορ διέλευσης ενεργοποιείται όταν θέλουμε να κάνουμε ανάγνωση ή εγγραφή στο κύτταρο μνήμης. Η ενεργοποίηση γίνεται μέσω μίας αγώγιμης γραμμής, η οποία ονομάζεται *Word – Line (WL),* και η οποία συνδέεται στην πύλη του τρανζίστορ διέλευσης .

Ο πυκνωτής μπορεί να είναι κάτω από επίπεδο του τρανζίστορ βαθιά μέσα στο πυρίτιο, οπότε ονομάζεται *πυκνωτής πηγαδιού (trench capacitor)* ή επάνω από το επίπεδο του τρανζίστορ, οπότε ονομάζεται πυκνωτής στοίβας (stack capacitor). Στο Σχήμα 2.2 βλέπουμε κύτταρα μνήμης με τους δύο τύπους πυκνωτών. Παρόλο που δεν έχει γίνει κάποια υπόθεση σε αυτή την εργασία για τον τύπο του πυκνωτή, φαίνεται ότι οι πυκνωτές πηγαδιού προτιμούνται, κυρίως γιατί καταλαμβάνουν μικρότερη επιφάνεια πυριτίου [1].



Σχήμα 2.1 Το Κύτταρο Μνήμης

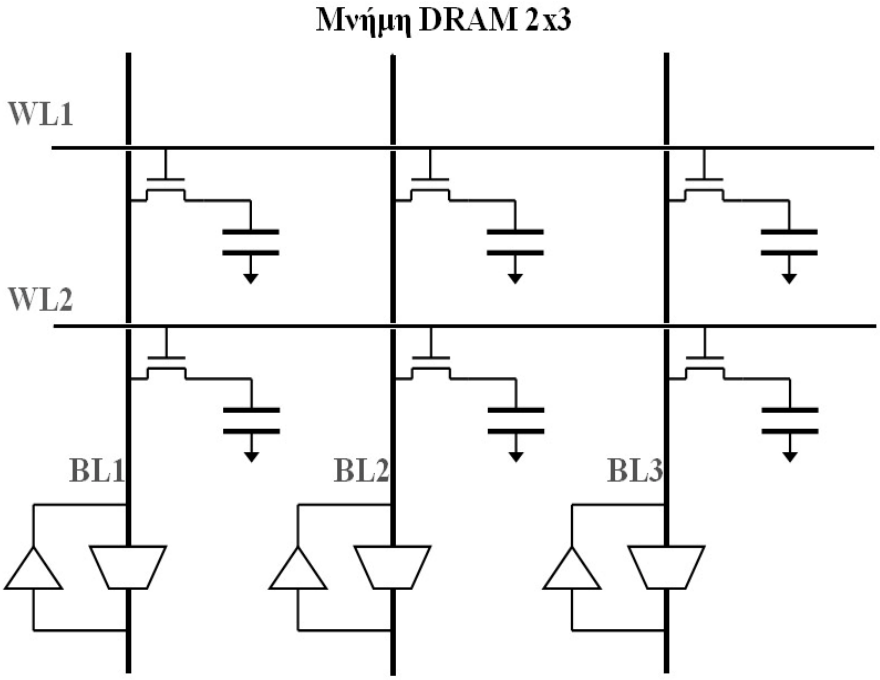


Σχήμα 2.2 Οι Δύο Τύποι Πυκνωτών

Τα κύτταρα μνήμης οργανώνονται σε *συστοιχίες* κυττάρων (Σχήμα 2.3). Για λόγους απλότητας θα θεωρήσουμε αρχικά ότι σε μία συστοιχία τα κύτταρα είναι διατεταγμένα σε μορφή πίνακα. Οι κυψελίδες που βρίσκονται στην ίδια γραμμή έχουν τις πύλες των τρανζίστορ διέλευσης συνδεδεμένες στην ίδια Word – Line ενώ οι κυψελίδες που βρίσκονται στην ίδια στήλη είναι συνδεδεμένες στην ίδια Bit – Line.

Ανάλογα με τον τύπο της μνήμης σε κάθε διαδικασία ανάγνωσης ή εγγραφής μπορεί να διαβάζεται ή να εγγράφεται ένα κύτταρο από μία συστοιχία *(λειτουργία bit - bit oriented memory)* ή περισσότερα από ένα *(λειτουργία λέξης - word oriented memory)*. Φυσικά οι σημερινές μνήμες χειρίζονται τα δεδομένα με τη μορφή λέξεων αλλά ενδέχεται η λέξη να

σχηματίζεται ενεργοποιώντας περισσότερες από μία συστοιχίες μνήμης και χρησιμοποιώντας ένα bit από κάθε μία. Στην παρούσα εργασία μελετάμε κάθε συστοιχία εντελώς ανεξάρτητα, αφού είναι αδύνατο να έχουμε αλληλεπίδραση μεταξύ κυττάρων που ανήκουν σε διαφορετικές συστοιχίες, οπότε μας ενδιαφέρει το πώς η συστοιχία χειρίζεται τα δεδομένα (με τη μορφή bit ή λέξης). Συνεπώς όταν αναφερόμαστε σε μνήμες λειτουργίας bit θα εννοούμε ότι η κάθε συστοιχία της μνήμης λειτουργεί με τον bit τρόπο.

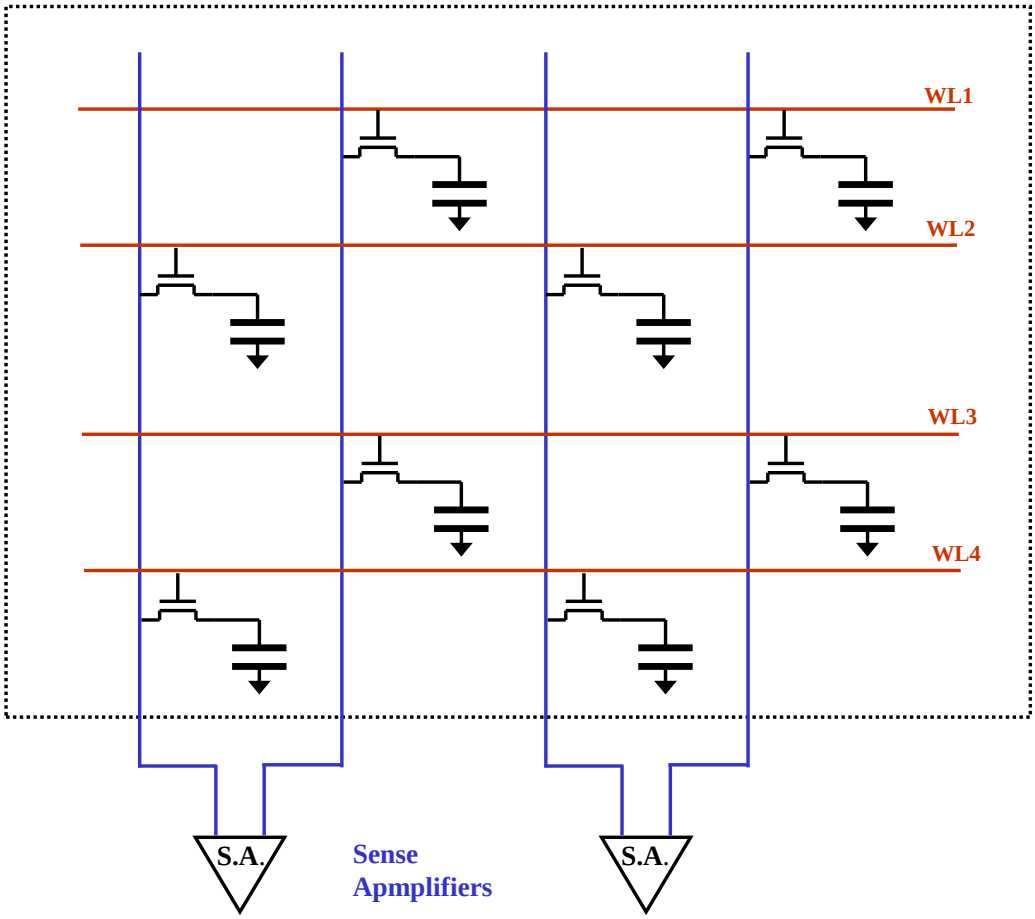


Σχήμα 2.3 Η Συστοιχία Μνήμης

Προκειμένου να είναι δυνατή η διαδικασία ανάγνωσης υπάρχει και μία ενισχυτική διάταξη, ο *Αισθητήρας Σήματος (Sense Amplifier).* Στη βασική του τοπολογία ο αισθητήρας σήματος αποτελείται από δύο αναστροφείς οι οποίοι είναι συνδεδεμένοι έτσι ώστε η είσοδος του ενός να συνδέεται με την έξοδο του άλλου. Στους δύο ακροδέκτες κάθε αισθητήρα σήματος συνδέονται δύο Bit – Lines. Έτσι δημιουργούνται ζεύγη από Bit – Lines, τις οποίες θα συμβολίζουμε στο εξής με BL και $\overline{BL}$ και κάθε ζεύγος συνδέεται σε έναν ενισχυτή σήματος.

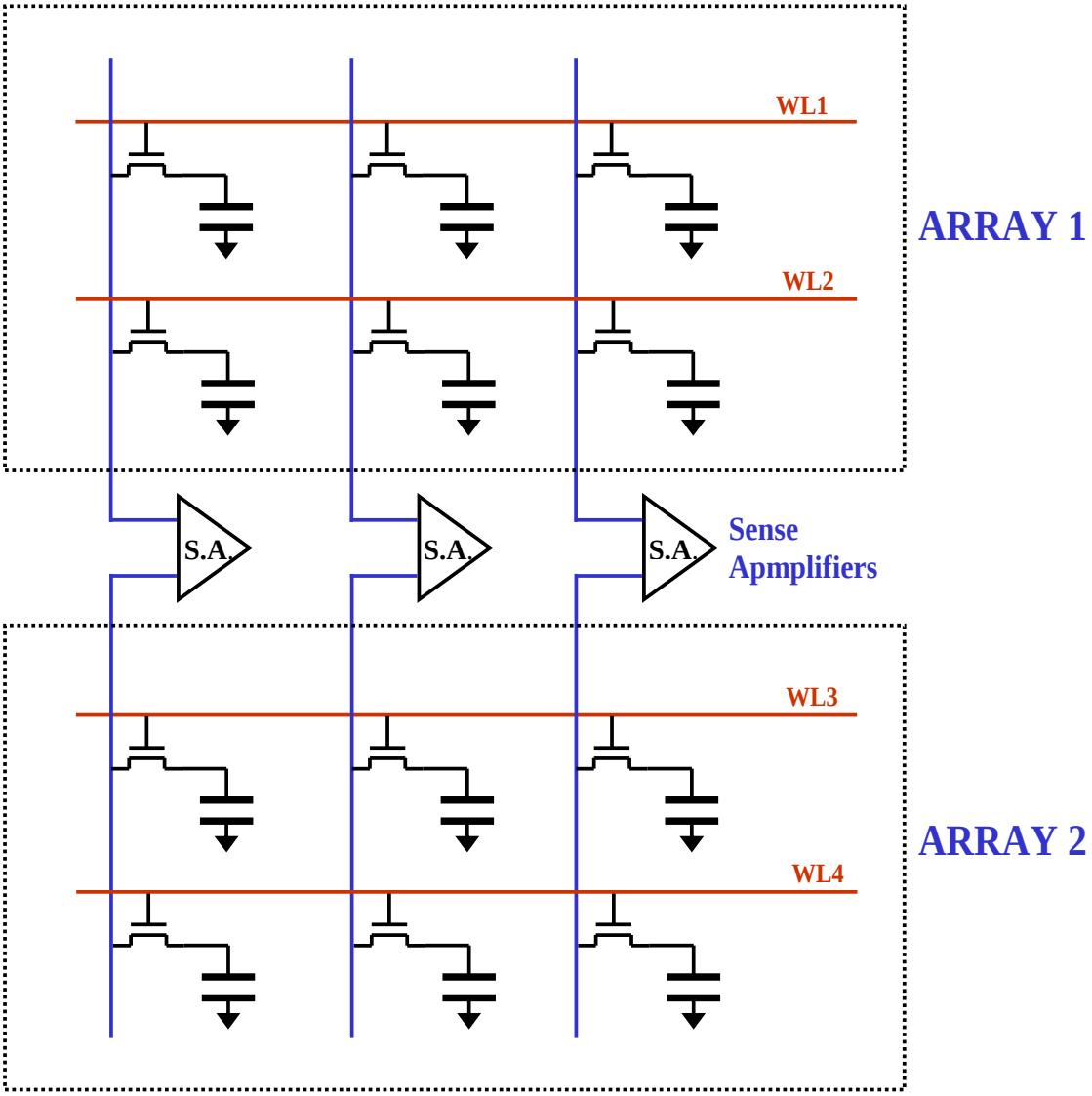
Σε κάθε τέτοιο ζεύγος Bit – Lines τα τρανζίστορ διέλευσης των κυττάρων μνήμης ενεργοποιούνται από διαφορετικές *ομάδες* Word – Lines, δηλαδή μία Word – Line μπορεί να ενεργοποιήσει ένα τρανζίστορ μίας μόνο εκ των BL και $\overline{BL}$. Οι δύο Bit – Lines κάθε ζεύγους μπορεί να βρίσκονται είτε σε διαφορετικές συστοιχίες μνήμης οπότε έχουμε

αρχιτεκτονική τύπου *ανοικτής συστοιχίας (open array)* είτε στην ίδια συστοιχία, οπότε έχουμε τύπο *εμφωλευμένης συστοιχίας (folded array)* [2]. Οι δύο τύποι φαίνονται στα Σχήματα 2.4 και 2.5

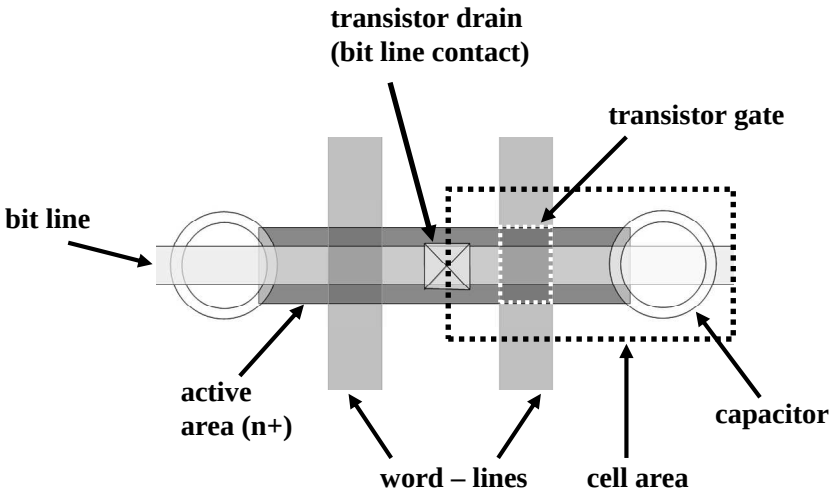


Σχήμα 2.4 Η Αρχιτεκτονική Εμφωλευμένης Συστοιχίας

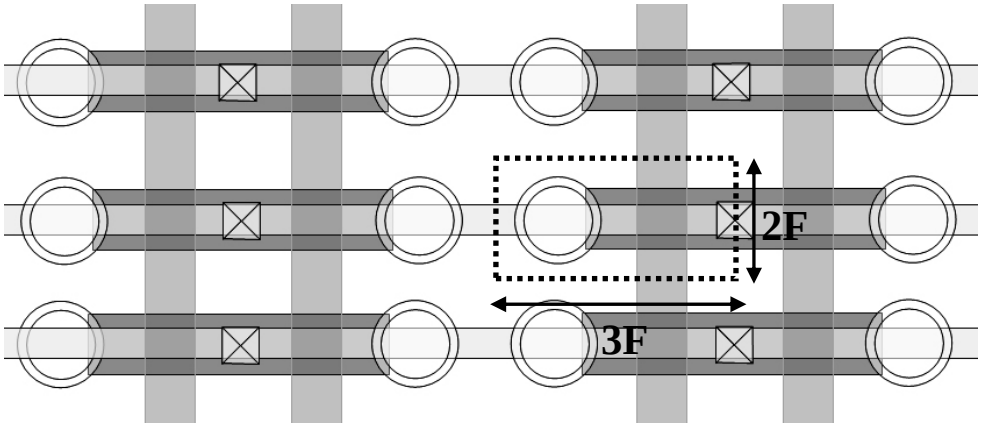
Τα κύτταρα τοποθετούνται σε ζεύγη όπου τα τρανζίστορ του κάθε ζεύγους έχουν κοινή υποδοχή και επαφή με τη Bit – Line. Η δομή ενός τέτοιου ζεύγους κυττάρων φαίνεται στο Σχήμα 2.6. Στα Σχήματα 2.7 και 2.8 βλέπουμε τη δομή μιας μνήμης ανοικτής συστοιχίας και μίας μνήμης εμφωλευμένης συστοιχίας αντίστοιχα. Στα σχήματα αυτά βλέπουμε και τον χώρο που καταλαμβάνει το κύτταρο μνήμης στις δύο αρχιτεκτονικές, συναρτήσει της *Ελάχιστης Λιθογραφικής Απόστασης F*. Το F είναι η ελάχιστη απόσταση μεταξύ δύο αγώγιμων γραμμών (η οποία είναι ίση και με το ελάχιστο πλάτος μίας αγώγιμης γραμμής) που επιτρέπει η τεχνολογία. Βλέπουμε ότι στην αρχιτεκτονική ανοικτής συστοιχίας η επιφάνεια που καταλαμβάνει το κύτταρο είναι ίση με $6F^2$ ενώ στην αρχιτεκτονική εμφωλευμένης συστοιχίας είναι ίση με $8F^2$.



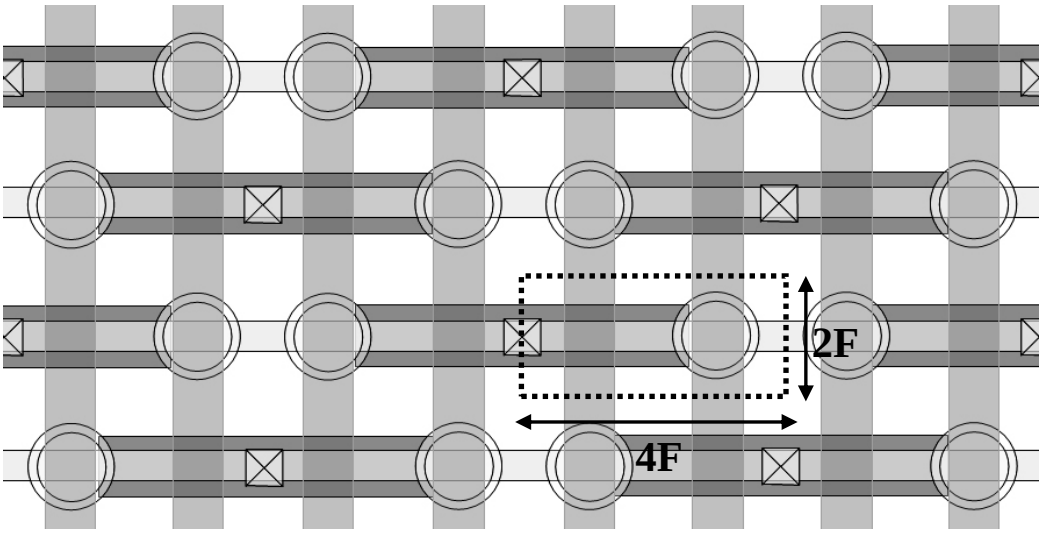
Σχήμα 2.5 Η Αρχιτεκτονική Ανοικτής Συστοιχίας



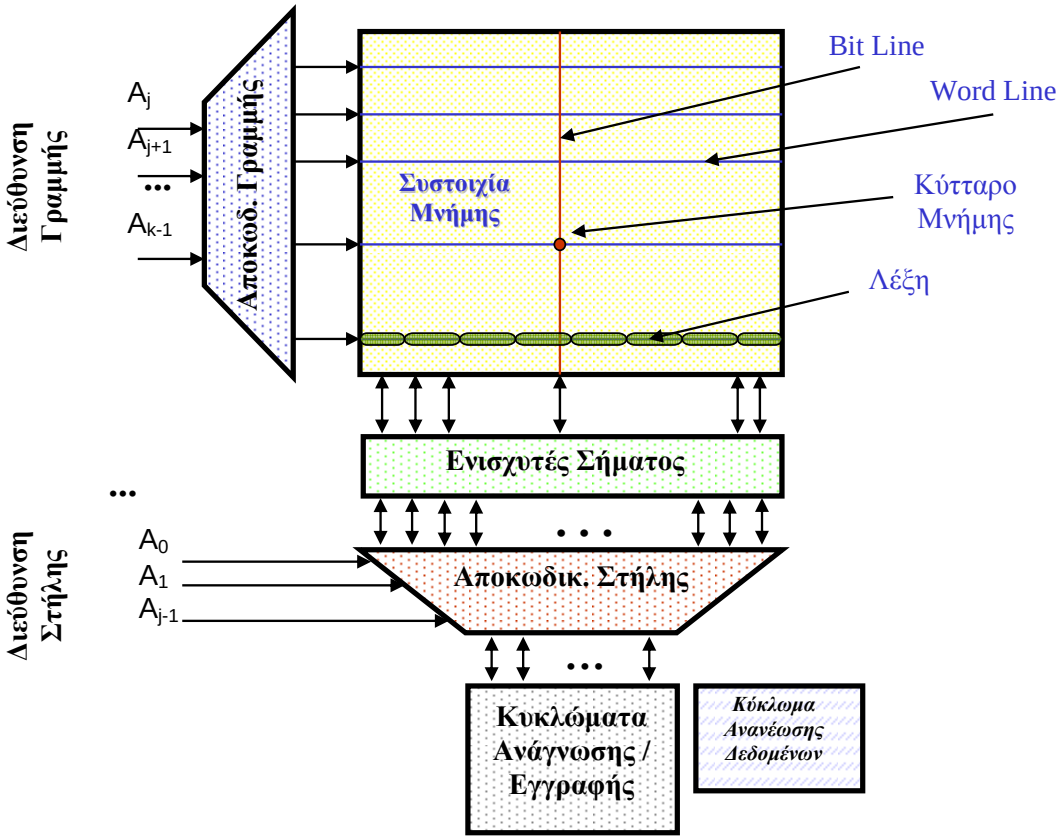
Σχήμα 2.6 Η Δομή των Κυττάρων Μνήμης



Σχήμα 2.7 Η Δομή της Μνήμης Ανοικτής Συστοιχίας



Σχήμα 2.8 Η Δομή της Μνήμης Εμφωλευμένης Συστοιχίας



Σχήμα 2.9 Η Συστοιχία Μνήμης με τα Βοηθητικά Κυκλώματα

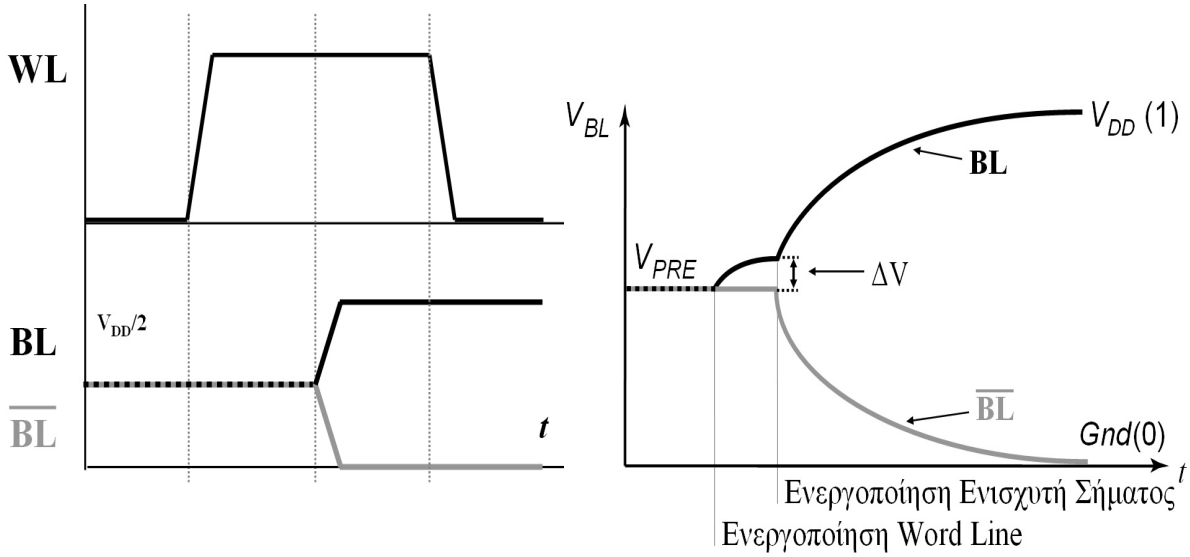
Κάθε διεύθυνση μνήμης αντιστοιχεί σε ένα κύτταρο μνήμης μέσα στην συστοιχία, αν έχουμε μνήμη λειτουργίας bit, ή σε μία ομάδα από κύτταρα, αν έχουμε μνήμη λειτουργίας λέξης. Η διεύθυνση χωρίζεται σε *διεύθυνση γραμμής* και *διεύθυνση στήλης*. Η πρόσβαση στο κατάλληλο κύτταρο (ή κύτταρα) επιτυγχάνεται με δύο διατάξεις, *τον αποκωδικοποιητή γραμμής (row decoder) και τον αποκωδικοποιητή στήλης (column decoder).* Ο αποκωδικοποιητής γραμμής με βάση τη διεύθυνση γραμμής ενεργοποιεί την σχετική Word – Line. Αντίστοιχα, ο αποκωδικοποιητής στήλης με βάση τη διεύθυνση στήλης επιλέγει την αντίστοιχη Bit – Line (ή τις αντίστοιχες Bit – Lines) προκειμένου είτε να διαβάσουμε είτε να γράψουμε δεδομένα (ανάλογα αν πρόκειται για πράξη ανάγνωσης ή εγγραφής), στο κύτταρο μνήμης (ή τα κύτταρα μνήμης) που ενεργοποιήθηκε(αν) από την Word – Line.

Η διαδικασία ανάγνωσης έχει γενικά ως εξής: πριν την ανάγνωση όλες οι Bit – Lines (BL και $\overline{BL}$) φορτίζουν σε τάση V_{DD} / 2. Στη συνέχεια, ο αποκωδικοποιητής γραμμής φορτίζει σε τάση > V_{DD} την Word – Line σύμφωνα με τη διεύθυνση μνήμης που θέλουμε να διαβάσουμε (δηλαδή τη γραμμή στην οποία βρίσκεται η κυψελίδα μνήμης που θέλουμε να διαβάσουμε). Μόλις γίνει αυτό, σε κάθε ζεύγος από Bit – Lines ενεργοποιείται ένα μόνο τρανζίστορ διέλευσης, το οποίο έστω ότι ανήκει στην BL του κάθε ζεύγους. Εξαιτίας της ενεργοποίησης του τρανζίστορ διέλευσης προκαλείται στην BL μία μικρή μεταβολή της τάσης προς το V_{DD} ή προς το 0, ανάλογα με το φορτίο του πυκνωτή της κυψελίδας μνήμης που βρίσκεται στην Word – Line που ενεργοποιήθηκε (προφανώς ενεργοποιείται μόνο μία Word – Line κάθε φορά).

Οι αισθητήρες σήματος σε πρώτη φάση ανιχνεύουν αυτές τις μεταβολές τάσης, χρησιμοποιώντας την τάση των $\overline{BL}$ ως τάση αναφοράς, και συνεπώς αντιλαμβάνονται τις λογικές τιμές που ήταν αποθηκευμένες σε αυτά τα κύτταρα. Σε δεύτερη φάση επιβάλλουν αυτές τις λογικές τιμές που ανίχνευσαν επάνω στις Bit – Lines. Η μετάβαση αυτή πραγματοποιείται με μεγαλύτερη αξιοπιστία όταν στα άκρα του ενισχυτή σήματος έχουν συνδεθεί ίδιες χωρητικότητες και για αυτό το λόγο χρησιμοποιείται για αναφορά η $\overline{BL}$ φορτισμένη σε τάση V_{DD} / 2 αντί να χρησιμοποιηθεί μία σταθερή πηγή τάσης V_{DD} / 2. Όταν τελειώσει η διαδικασία των μεταβάσεων, όλες οι BL θα έχουν την τιμή που είχε αποθηκευμένη η αντίστοιχη κυψελίδα που ενεργοποιήθηκε στη στήλη τους και οι αντίστοιχες $\overline{BL}$ το συμπλήρωμα αυτής. Με αυτό τον τρόπο στα κύτταρα που έγινε ανάγνωση

ξαναγράφεται η τιμή που αυτά αρχικά είχαν. Το βήμα αυτό είναι απαραίτητο καθώς η προηγούμενη διεργασία της ανάγνωσης παραποιεί τα αποθηκευμένα δεδομένα στο κύτταρο μνήμης. Για αυτό το λόγο λέμε ότι κατά τη διαδικασία της ανάγνωσης έχουμε στην ουσία ανάγνωση – επανεγγραφή.

Προφανώς ο ίδιος ενισχυτής σήματος χρησιμοποιείται και για την ανάγνωση των $\overline{\text{BL}}$. Σε αυτή την περίπτωση οι ρόλοι αντιστρέφονται και η BL χρησιμεύει για να παρέχει την τάση αναφοράς, ενώ οι τιμές των κυττάρων που διαβάσαμε είναι αυτές των $\overline{\text{BL}}$. Συνεπώς, ανάλογα με την ομάδα στην οποία ανήκει η Word – Line που ενεργοποιήθηκε, οι τιμές που μας ενδιαφέρουν μπορεί να είναι είτε αυτές των BL είτε αυτές των $\overline{\text{BL}}$. Για το λόγο αυτό χρησιμοποιείται και ένας πολυπλέκτης για να επιλέξουμε, ανάλογα με την Word – Line που ενεργοποιήθηκε, αν η έξοδός μας θα είναι η τιμή της BL ή της $\overline{\text{BL}}$.



Σχήμα 2.10 Μεταβάσεις Κατά την Διαδικασία Ανάγνωσης

Στο τέλος της διαδικασίας, η Word – Line επανέρχεται σε τάση 0. Οι μεταβάσεις αυτές φαίνονται παραστατικά στο Σχήμα 2.10

Στη διαδικασία της εγγραφής απλά τα κυκλώματα εγγραφής επιβάλλουν στις Bit – Lines τις

τιμές που θέλουμε να γραφούν, ώστε όταν ενεργοποιηθεί η Word – Line αυτές να επιβληθούν και στις αντίστοιχες κυψελίδες. Επειδή το μήκος της λέξης που γράφεται είναι συνήθως πολύ μικρότερο από τον συνολικό αριθμό των Bit – Lines που ενεργοποιεί η Word – Line, οι ενισχυτές σήματος παραμένουν ενεργοί έτσι ώστε οι Bit – Lines που δεν θα γραφούν να εκτελέσουν διαδικασία ανάγνωσης – επανεγγραφής ώστε να μην χάσουν τα δεδομένα τους. Έτσι, ενώ οι Bit – Lines που θέλουμε να γραφούν παίρνουν την τιμή που εμείς τους επιβάλλουμε με τα κυκλώματα εγγραφής, οι υπόλοιπες που ενεργοποιούνται μαζί τους παίρνουν την τιμή που έχει αποθηκευμένη το εκάστοτε αντίστοιχο κύτταρο που ενεργοποιείται, όπως ακριβώς γίνεται στη διαδικασία ανάγνωσης.

Επειδή τα τρανζίστορ διέλευσης παρουσιάζουν κάποιο μικρό ρεύμα διαρροής ακόμα και όταν είναι απενεργοποιημένα, το φορτίο των πυκνωτών των κυττάρων αλλάζει με το χρόνο κινούμενο προς μία κατάσταση ισορροπίας κοντά στην τάση V_{DD} / 2. Προκειμένου να μην έχουμε απώλεια δεδομένων η μνήμη εκτελεί κατά τακτά χρονικά διαστήματα μία διαδικασία ανανέωσης (refresh) των δεδομένων, η οποία δεν είναι τίποτε άλλο παρά ανάγνωση (και συνεπώς επανεγγραφή) των κυττάρων.

Η επικρατέστερη αρχιτεκτονική της συστοιχίας των κυττάρων μνήμης είναι αυτή όπου κάθε ζεύγος από Bit – Lines που συνδέονται στον ίδιο ενισχυτή σήματος βρίσκονται στην ίδια συστοιχία μνήμης, δηλαδή η αρχιτεκτονική εμφωλευμένης συστοιχίας, παρόλο που η συνολική επιφάνεια των κυττάρων μνήμης καταλαμβάνει 25% περισσότερο χώρο σε σχέση με την αρχιτεκτονική ανοικτής συστοιχίας [3] [4]. Αυτό συμβαίνει γιατί οι αρχιτεκτονικές εμφωλευμένης συστοιχίας έχουν καλύτερο λόγο σήματος προς θόρυβο με αποτέλεσμα μεγαλύτερες ταχύτητες και αυξημένη αξιοπιστία. Επίσης ο συνολικός επιπλέον χώρος που καταλαμβάνει η μνήμη εμφωλευμένης συστοιχίας είναι μικρότερος του 25% γιατί με αυτή την αρχιτεκτονική κάθε ενισχυτής σήματος είναι δυνατό να εξυπηρετεί τέσσερις Bit – Lines, αντί για δύο που εξυπηρετεί η ανοιχτή συστοιχία, και συνεπώς χρειαζόμαστε τους μισούς ενισχυτές σήματος [1].

ΚΕΦΑΛΑΙΟ 3. ΤΟ NPSF ΜΟΝΤΕΛΟ ΣΦΑΛΜΑΤΩΝ

3.1. Σφάλματα Ψηφιακών Κυκλωμάτων

Ο έλεγχος ορθής λειτουργίας ενός ηλεκτρονικού κυκλώματος είναι μία μείζονος σημασίας διαδικασία η οποία έχει σαν στόχο να εντοπίσει πιθανές κατασκευαστικές ατέλειες ή βλάβες πριν αυτό διοχετευθεί στην αγορά ή τοποθετηθεί πάνω σε κάποιο ηλεκτρονικό σύστημα. Αυτές οι κατασκευαστικές ατέλειες ή βλάβες παρουσιάζονται κατά τη διαδικασία κατασκευής του ολοκληρωμένου κυκλώματος και ονομάζονται *ελαττώματα (defects)*. Με τον όρο *σφάλματα (faults)* εννοούμε τις μοντελοποιήσεις της συμπεριφοράς του ολοκληρωμένου κυκλώματος υπό την παρουσία ελαττωμάτων.

Μερικά παραδείγματα γενικών μοντέλων σφαλμάτων είναι τα παρακάτω:

- *Σφάλμα μόνιμης τιμής (stuck at fault)*: ένας κόμβος του κυκλώματος παρουσιάζει μόνιμα την ίδια τιμή.
- *Σφάλμα μόνιμα αγώγιμου / μη αγώγιμου τρανζίστορ (Transistor Stuck On / Stuck Open)*: ένα τρανζίστορ βρίσκεται μόνιμα σε αγώγιμη / μη αγώγιμη κατάσταση
- Σφάλμα βραχυκύκλωσης (Bridging Fault): δύο κόμβοι έχουν βραχυκυκλωθεί
- *Σφάλμα ανοιχτοκυκλώματος (open circuit fault):* μία αγώγιμη γραμμή του κυκλώματος έχει κοπεί.
- *Σφάλμα καθυστέρησης διάδοσης σήματος (Delay Fault):* καθυστέρηση του σήματος σε μία ή περισσότερες διαδρομές του κυκλώματος

Η διαδικασία εντοπισμού ελαττωμάτων στην κατασκευή του κυκλώματος περιλαμβάνει

διάφορες μεθόδους, με συνηθέστερη την εφαρμογή ενός συνόλου συνδυασμών τιμών στην είσοδο του κυκλώματος (οι οποίοι συνδυασμοί τιμών ονομάζονται *διανύσματα*) και σύγκριση των αποκρίσεων του κυκλώματος με τις εκάστοτε αναμενόμενες. Μία λανθασμένη λογική απόκριση του κυκλώματος, η οποία ονομάζεται *λάθος (error)* αποδεικνύει την παρουσία ελαττώματος σε αυτό. Είναι δεδομένο ότι η εφαρμογή όλων των δυνατών διανυσμάτων αποτελεί συνήθως μια πρακτικά αδύνατη διαδικασία, λόγω του εξαιρετικά μεγάλου αριθμού των διανυσμάτων. Ένας πιο εφικτός στόχος είναι να επιλεγεί ένα κατάλληλο υποσύνολο των δυνατών διανυσμάτων το οποίο εφαρμοζόμενο στο υπό έλεγχο κύκλωμα να εγγυάται ότι αν υπάρχει σε αυτό κάποιο σφάλμα, σύμφωνα με ένα δεδομένο μοντέλο σφαλμάτων, τότε το κύκλωμα θα οδηγηθεί σε λανθασμένη απόκριση που θα μας επιτρέψει την ανίχνευση του σφάλματος.

3.2. Μοντέλα Σφαλμάτων Μνημών

Όπως είδαμε στο προηγούμενο κεφάλαιο οι μνήμες αποτελούνται από τα κύτταρα μνήμης και έναν αριθμό από υποκυκλώματα. Κάθε ένα από τα υποκυκλώματα αυτά, όπως οι ενισχυτές σήματος, οι αποκωδικοποιητές γραμμής – στήλης, διάφοροι καταχωρητές κτλ είναι επιφορτισμένο με μία συγκεκριμένη λειτουργία και συνήθως εμπλέκεται στις περισσότερες από τις λειτουργίες εγγραφής και ανάγνωσης που λαμβάνουν χώρα σε μία μνήμη. Για αυτό το λόγο είναι σχετικά εύκολο να εντοπίσουμε ένα πιθανό ελάττωμα σε ένα τέτοιο υποκύκλωμα, αφού σε ένα σύνολο λειτουργιών της μνήμης το ελάττωμα που φέρει θα έχει πολλές ‘ευκαιρίες’ να παράγει ένα λάθος.

Αντίθετα, κάθε κύτταρο μνήμης εμπλέκεται άμεσα μόνο στις διαδικασίες εγγραφής ή ανάγνωσης που αφορούν το ίδιο και επομένως η διαδικασία γίνεται πιο χρονοβόρα, αφού σε κάθε πράξη εγγραφής ή ανάγνωσης μόνο ένας μικρός αριθμός κυττάρων λαμβάνουν μέρος. Αυτό όμως που κάνει τα πράγματα ακόμα πιο δύσκολα είναι οι εξής παρατηρήσεις: **α)** Τα κύτταρα αλληλεπιδρούν μεταξύ τους, κυρίως λόγω της μικρής απόστασης στην οποία βρίσκονται. Η αλληλεπίδραση αυτή εξαρτάται από τις λογικές τιμές που φέρουν τα κύτταρα και από τις μεταβολές αυτών των τιμών (*λογικές μεταβάσεις*) **β)** Παρουσιάζονται στα κύτταρα κάποια ελαττώματα τα οποία είναι τόσο ασθενή που προκαλούν εμφάνιση λάθους μόνο υπό συγκεκριμένες συνθήκες. Επειδή σε αυτή την περίπτωση τον καθοριστικό ρόλο τον παίζει η αλληλεπίδραση μεταξύ των κυττάρων, (των γειτονικών όπως θα δούμε στη συνέχεια) η εμφάνιση ή όχι ενός λάθους στην τιμή που διαβάζουμε από ένα κύτταρο καθορίζεται από

τις τιμές και τις μεταβάσεις των υπολοίπων κυττάρων.

Υπάρχουν τέσσερις κατηγορίες μοντέλων σφαλμάτων για τις μνήμες:

- Σφάλματα στα οποία εμπλέκεται ένα μόνο κύτταρο. Τέτοια είναι τα *Σφάλματα Μόνιμης Τιμής (stuck at fault)* όπου ένα κύτταρο παρουσιάζει μόνιμα την ίδια τιμή και τα *Σφάλματα Μετάβασης (transition faults)*, όπου ένα κύτταρο αδυνατεί να εκτελέσει μετάβαση από μία λογική τιμή στην άλλη.
- Σφάλματα στα οποία εμπλέκονται δύο κύτταρα. Αυτά είναι τα *Σφάλματα Σύζευξης (coupling faults – CF)* [5]
- Σφάλματα στα οποία εμπλέκονται k κύτταρα. Αυτά χωρίζονται σε δύο υποκατηγορίες:
  α) αν τα k κύτταρα μπορεί να βρίσκονται οπουδήποτε στη μνήμη τότε έχουμε τα *k - σύζευξης σφάλματα (k - coupling faults)*, τα *σφάλματα γεφύρωσης (bridging faults)* και τα *σφάλματα σύζευξης κατάστασης (state coupling faults)* [5] [6].
  β) αν τα k κύτταρα είναι επιλεγμένα με βάση τη γειτνίασή τους πάνω στη συστοιχία μνήμης και σχηματίζουν μια *γειτονιά* κυττάρων τότε έχουμε το *Μοντέλο Σφαλμάτων με Εξάρτηση από το Διάνυσμα Τιμών Γειτονιάς (Neighborhood Pattern Sensitive Fault - NPSF)* [5] [7].

Μέχρι πρόσφατα το μοντέλο σφαλμάτων που κυριαρχούσε στον έλεγχο ορθής λειτουργίας των μνημών ήταν το μοντέλο σφαλμάτων σύζευξης. Αυτό οφειλόταν στο σχετικά χαμηλό κόστος εφαρμογής των αντίστοιχων αλγορίθμων ανίχνευσης. Η ανίχνευση αυτών των σφαλμάτων γίνεται με χρήση των καλούμενων *March αλγορίθμων* [5] [6] [8]. Ένας March αλγόριθμος είναι μία πεπερασμένη ακολουθία από στοιχεία March (March element). Κάθε στοιχείο March περιλαμβάνει την εφαρμογή μιας πεπερασμένης ακολουθίας πράξεων ανάγνωσης ή εγγραφής σε κάθε κύτταρο της μνήμης. Η εφαρμογή των πράξεων της ακολουθίας μπορεί να γίνει ξεκινώντας από το κύτταρο με τη μικρότερη διεύθυνση μνήμης και προχωρώντας προς τη μεγαλύτερη διεύθυνση ή και κατά την αντίθετη φορά. Σε κάθε περίπτωση εφαρμόζονται όλες οι πράξεις της ακολουθίας ενός στοιχείου στο τρέχον κύτταρο και κατόπιν προχωρούμε στο επόμενο. Κάθε στοιχείο εφαρμόζεται σε όλη τη μνήμη και στη συνέχεια προχωρούμε στο επόμενο στοιχείο [5] [6].

Προφανώς υπάρχουν άπειροι αλγόριθμοι που μπορούν να κατασκευαστούν με τον παραπάνω ορισμό. Στις αρχές της δεκαετίας του 90 οι πιο ακριβοί σε χρόνο εκτέλεσης March

αλγόριθμοι ήταν οι March A και March B με απαιτούμενο αριθμό πράξεων ανάγνωσης και εγγραφής 15*n και 17*n αντίστοιχα, όπου n είναι ο αριθμός των κυττάρων της μνήμης [5]. Με τις σημερινές μνήμες όμως απαιτούνται πιο πολύπλοκοι March αλγόριθμοι προκειμένου να έχουμε ικανοποιητική κάλυψη σφαλμάτων. Στην μέχρι σήμερα βιβλιογραφία έχουν προταθεί πολλοί MARCH αλγόριθμοι με κόστος από 5*n έως 33*n [5] [6]. Τα τελευταία χρόνια, και ως ανταπόκριση στις απαιτήσεις των σύγχρονων DRAM μνημών, έχουν προταθεί αλγόριθμοι με υψηλότερο κόστος, από 70*n ως 100*n [9] [10].

3.3. Το NPSF Μοντέλο Σφαλμάτων

Το γενικό μοντέλο που περιγράφει την ευαισθησία ενός κυττάρου μνήμης στις τιμές και τις μεταβάσεις των υπολοίπων κυττάρων είναι το Μοντέλο Σφαλμάτων Εξάρτησης από Διάνυσμα Τιμών *(Pattern Sensitive Fault – PSF)*. Μπορούμε να δούμε μία μνήμη που περιέχει n κύτταρα σαν ένα n – διάστατο διάνυσμα, όπου το πεδίο τιμών για κάθε διάσταση είναι το [0,1]. Σύμφωνα με αυτό το μοντέλο, για να γίνει έλεγχος της μνήμης πρέπει να γίνουν όλες οι δυνατές μεταβάσεις κάθε κυττάρου της και για όλες τις δυνατές τιμές των υπολοίπων κυττάρων και κάθε φορά να διαβάζουμε το περιεχόμενο της μνήμης. Το κόστος μίας τέτοιας διαδικασίας είναι της τάξης του 2^n και φυσικά είναι απαγορευτικό.

Ένα πιο ρεαλιστικό και ευρέως αποδεκτό μοντέλο είναι το μοντέλο *Σφαλμάτων Εξάρτησης από το Διάνυσμα Τιμών της Γειτονιάς (Neighborhood Pattern Sensitive Fault – NPSF)* [5-7] [11-15]. Σύμφωνα με αυτό το μοντέλο, η αποθηκευμένη τιμή ενός κυττάρου και η δυνατότητα να επιβάλλουμε σε αυτό μία τιμή επηρεάζεται από τις τιμές και τις μεταβάσεις των γειτονικών του κυττάρων σε μία συστοιχία μνήμης.

Σε αυτό το μοντέλο το κύτταρο που δέχεται την επιρροή λέγεται *κύτταρο βάσης (base cell)* ενώ τα γειτονικά του κύτταρα ονομάζονται *υπολειπόμενη γειτονιά (deleted neighbourhood)* . Στην πράξη τα κύτταρα της υπολειπόμενης γειτονιάς είναι τα κοντινότερα (και επομένως τα πιθανότερα να επηρεάσουν το κύτταρο βάσης). Ο συνδυασμός του κυττάρου βάσης με τα κύτταρα της υπολειπόμενης γειτονιάς ονομάζεται *γειτονιά (neighbourhood).*

Το μοντέλο NPSF περιλαμβάνει τις ακόλουθες τρεις υποκατηγορίες μοντέλων σφαλμάτων:

- *Ενεργό ή Δυναμικό (Active ή Dynamic) NPSF, το οποίο καλείται ANPSF*. Σύμφωνα με αυτό τα δεδομένα του κύτταρου βάσης αλλάζουν εξαιτίας μίας αλλαγής στην

υπολειπόμενη γειτονιά. Προκειμένου να εντοπιστούν τέτοια σφάλματα πρέπει κάθε κύτταρο βάσης να διαβαστεί σε τιμή 0 και σε τιμή 1 για κάθε δυνατή αλλαγή στην υπολειπόμενη γειτονιά (κάθε δυνατό συνδυασμό αρχικής και τελικής κατάστασης που διαφέρουν ως προς την τιμή ενός κυττάρου).

- *Παθητικό (Passive) NPSF, που καλείται PNPSF*. Σύμφωνα με αυτό τα δεδομένα του κυττάρου βάσης δεν είναι δυνατό να αλλάξουν εξαιτίας ενός συγκεκριμένου διανύσματος στην υπολειπόμενη γειτονιά. Για να εντοπιστεί ένα PNPSF σφάλμα πρέπει κάθε κύτταρο βάσης να γραφεί και να διαβαστεί σε τιμή 0 και σε τιμή 1 για κάθε δυνατό διάνυσμα στην υπολειπόμενη γειτονιά.
- *Στατικό (Static) NPSF, που καλείται SNPSF,* και σύμφωνα με το οποίο τα δεδομένα του κυττάρου βάσης ωθούνται σε μία συγκεκριμένη κατάσταση εξαιτίας ενός συγκεκριμένου διανύσματος στη υπολειπόμενη γειτονιά. Για να εντοπιστεί πρέπει το κύτταρο βάσης να διαβαστεί σε κατάσταση 0 και 1 για κάθε δυνατό διάνυσμα στην υπολειπόμενη γειτονιά.

Στο εξής θα χρησιμοποιούμε το σύμβολο ↑ για να δηλώσουμε ότι ένα κύτταρο εκτελεί μετάβαση από λογικό 0 σε 1 ενώ την αντίστροφη μετάβαση θα τη συμβολίσουμε με το σύμβολο ↓.

Προκειμένου να εντοπίσουμε τα Δυναμικά NPSF σφάλματα, χρειαζόμαστε για κάθε κύτταρο βάσης $(k-1)*2^k$ διανύσματα ελέγχου, όπου k είναι ο αριθμός των κυττάρων της γειτονιάς. Αυτό προκύπτει ως εξής: για κάθε μία από τις τιμές του κυττάρου βάσης (0 και 1) κάθε ένα από τα υπόλοιπα k – 1 κύτταρα πρέπει να κάνει και τις δύο δυνατές μεταβάσεις ↑, ↓ για κάθε δυνατό συνδυασμό διανυσμάτων των υπόλοιπων k-2 κυττάρων (2^{k-2} διανύσματα). Άρα συνολικά έχουμε $2*2*(k – 1)* 2^{k-2} = (k-1)*2^k$. Με άλλα λόγια, αυτό που πρέπει να γίνει είναι κάθε κύτταρο της γειτονιάς, εκτός του κυττάρου βάσης, να κάνει τις δύο δυνατές μεταβάσεις ↑ και ↓ για όλα τα δυνατά διανύσματα των υπολοίπων k – 1 κυττάρων. Εννοείται πως μετά από κάθε μετάβαση διαβάζουμε το κύτταρο βάσης.

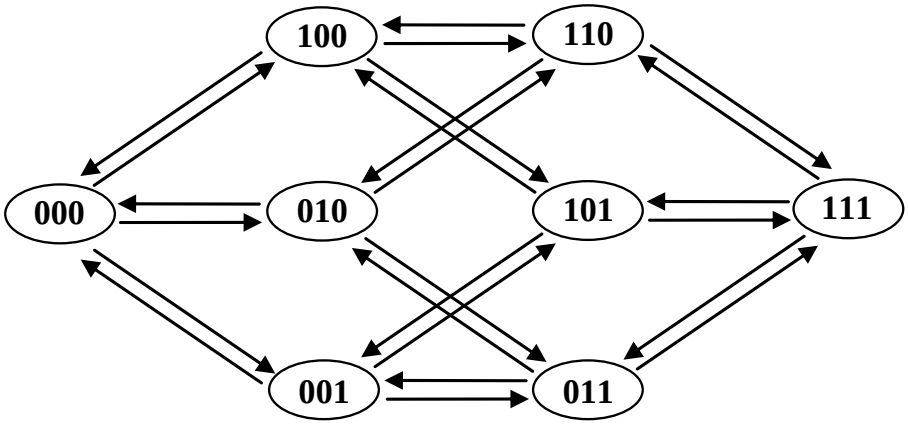
Παρόμοια, για να εντοπίσουμε τα Παθητικά NPSF σφάλματα, χρειαζόμαστε για κάθε κύτταρο βάσης 2^k διανύσματα ελέγχου, αριθμός που προκύπτει ως εξής: για κάθε δυνατό διάνυσμα των κυττάρων της υπολειπόμενης γειτονιάς (2^{k-1} διανύσματα) πρέπει το κύτταρο βάσης να κάνει τις δύο δυνατές μεταβάσεις ↑ και ↓.

Από τις παραπάνω περιγραφές καταλαβαίνουμε εύκολα ότι για να εντοπίσουμε και τα Δυναμικά και τα Παθητικά NPSF *(APNPSF)* σφάλματα χρειαζόμαστε $(k-1)*2^k + 2^k = k*2^k$ διανύσματα. Απαιτείται δηλαδή όλα τα κύτταρα της γειτονιάς (συμπεριλαμβανόμενου και του κυττάρου βάσης) να κάνουν τις δύο δυνατές μεταβάσεις ↑, ↓ για όλα τα δυνατά διανύσματα των υπολοίπων k – 1 κυττάρων. Εφόσον γίνει αυτό θα έχουμε καλύψει και τα Στατικά NPSF γιατί θα έχουμε διαβάσει το κύτταρο βάσης στις καταστάσεις 0 και 1 για κάθε δυνατό διάνυσμα στην υπολειπόμενη γειτονιά.

Να σημειώσουμε ότι όταν αναφέρουμε πως επιβάλλουμε $k*2^k$ διανύσματα ελέγχου αυτό δε σημαίνει ότι αυτά τα διανύσματα είναι όλα διαφορετικά μεταξύ τους, αφού άλλωστε με μήκος k υπάρχουν μόνο 2^k διαφορετικά διανύσματα. Αυτό που εννοούμε στην ουσία είναι ότι με αυτά τα 2^k διανύσματα κατασκευάζουμε μία ακολουθία διανυσμάτων μήκους $k*2^k$.

Για να επιβάλλουμε τα $k*2^k$ διανύσματα ελέγχου με τον ελάχιστο δυνατό αριθμό εγγραφών, η ακολουθία διανυσμάτων που επιλέγουμε είναι αυτή που προκύπτει διασχίζοντας τον αντίστοιχο *γράφο του Euler*. Ένας k-bit γράφος του Euler ορίζεται ως εξής [5]:

- Υπάρχει ένας κόμβος για κάθε (διαφορετικό) k-bit διάνυσμα
- Μεταξύ δύο κόμβων υπάρχουν δύο τόξα, ένα προς κάθε κατεύθυνση, αν και μόνο αν αυτοί διαφέρουν κατά ακριβώς ένα bit.



Σχήμα 3.1 Ο Γράφος του Euler για 3 - bit

Προφανώς ο γράφος αυτός αποτελείται από 2^k κόμβους και $k*2^k$ τόξα, αφού κάθε ένας από τους κόμβους συνδέεται με ακριβώς k άλλους κόμβους με δύο τόξα, ένα για κάθε κατεύθυνση. Στο Σχήμα 3.1 βλέπουμε έναν 3-bit γράφο του Euler.

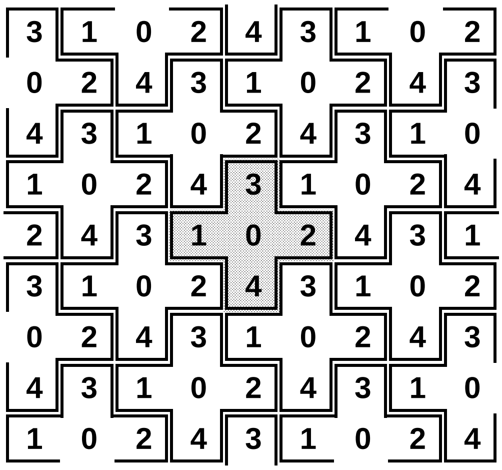
Ο έλεγχος για APNPSF σφάλματα επιτυγχάνεται διασχίζοντας όλα τα τόξα του γράφου (δηλ όλες τις ακμές) και προς τις δύο κατευθύνσεις. Η διάσχιση αυτή απαιτεί $k*2^k + 1$ εγγραφές και ισάριθμες αναγνώσεις ολόκληρης της μνήμης, μία εγγραφή και ανάγνωση για την επιβολή της αρχικής τιμής (δηλαδή να έρθουμε στην κατάσταση του αρχικού κόμβου του γράφου του Euler) και τις υπόλοιπες για να διασχίσουμε τα τόξα. Συνήθως στην αρχικοποίηση αυτή γράφουμε όλη τη μνήμη με τη λογική τιμή 0, οπότε ο αρχικός μας κόμβος είναι ο (0, 0, …,0).

Η επιλογή αυτής της ακολουθίας διανυσμάτων εξυπηρετεί και έναν άλλο στόχο: θέλουμε μεταξύ δύο αναγνώσεων του κυττάρου βάσης να γίνεται μόνο μία αλλαγή στη γειτονιά του, δηλαδή μόνο ένα κύτταρο να αλλάζει τα δεδομένα του. Βέβαια στην κανονική λειτουργία της μνήμης αυτό δε συμβαίνει πάντα και ενδέχεται η εμφάνιση ενός λάθους να είναι αποτέλεσμα δύο ή περισσότερων διαδοχικών αλλαγών στη γειτονιά του κυττάρου βάσης (ειδικά στις μνήμες DRAM όπως θα δούμε). Όμως τέτοιου είδους σφάλματα δεν εξετάζονται από αυτό το μοντέλο.

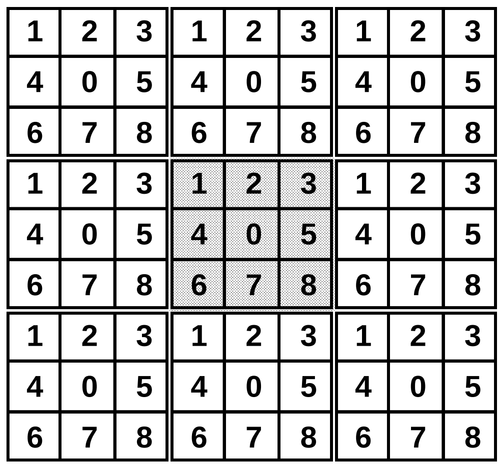
Διάφοροι τύποι γειτονιάς έχουν προταθεί κατά καιρούς στη βιβλιογραφία, μερικούς από τους οποίους θα αναφέρουμε. Έχει προταθεί το μοντέλο *Σφαλμάτων Διαταραχών Εξάρτησης από το Διάνυσμα Τιμών της Γειτονιάς (Disturb Neighborhood Pattern Sensitive Faults)* [16], το οποίο περιλαμβάνει k κύτταρα. Σύμφωνα με αυτό το μοντέλο το κύτταρο βάσης εκτελεί μία μετάβαση εξαιτίας μίας πράξης εγγραφής ή μίας πράξης ανάγνωσης ενός κυττάρου της γειτονιάς και ενός συγκεκριμένου διανύσματος που φέρουν τα υπόλοιπα k-2 κύτταρα. Ένα άλλο μοντέλο που έχει προταθεί είναι το μοντέλο *Σφαλμάτων Εξάρτησης από το Διάνυσμα Τιμών Γραμμής / Στήλης* (*Row / Column Pattern Sensitive Faults)* [17] σύμφωνα με το οποίο το κύτταρο βάσης είναι ευαίσθητο στα δεδομένα των κυττάρων που βρίσκονται στην ίδια Word – Line και την ίδια Bit – Line με αυτό. Επίσης έχει προταθεί ένας τύπος γειτονιάς που αποτελείται από τέσσερα κύτταρα, ο *Τύπος - T (T-type*) [18] [19], ο οποίος έχει σαν στόχο τον εντοπισμό τόσο των NPSF σφαλμάτων όσο και των σφαλμάτων που περιγράφονται από το μοντέλο *Σφαλμάτων Εξάρτησης Γειτονικών Bit - Line (Neighborhood Bit – Line Sensitive Faults - NBLSF)* το οποίο περιγράφει τα σφάλματα που δημιουργούνται από τη χωρητική σύζευξη μεταξύ των γειτονικών Bit - Lines.

Οι δύο επικρατέστερες γειτονιές είναι οι γειτονιές *Τύπου – 1* και *Τύπου – 2* [5-7]. Βλέποντας την μνήμη σαν ένα πίνακα, η *γειτονιά* Τύπου–1 όπως βλέπουμε στο Σχήμα 3.2 αποτελείται από το κύτταρο βάσης (κύτταρο 0) και από τα τέσσερα γειτονικά του κύτταρα επάνω, κάτω, δεξιά και αριστερά. Επομένως πρόκειται για μία γειτονιά με 5 κύτταρα και τα αντίστοιχα διανύσματα ελέγχου έχουν 5 bit, ενώ η αντίστοιχη ακολουθία διανυσμάτων αποτελείται από $5*2^5 = 160$ διανύσματα.

Αντίστοιχα, στην *γειτονιά Τύπου – 2* έχουν προστεθεί και τα τέσσερα διαγώνια κύτταρα όπως φαίνεται στο Σχήμα 3.3 οπότε η γειτονιά έχει 9 κύτταρα, τα διανύσματα ελέγχου έχουν 9 bit και η ακολουθία διανυσμάτων αποτελείται από $9*2^9 = 4608$ διανύσματα. Βλέπουμε ότι ο αριθμός των διανυσμάτων της ακολουθίας αυτής είναι πολύ μεγάλος σε σχέση με αυτόν της γειτονιάς Τύπου – 1.



Σχήμα 3.2 Η Γειτονιά Τύπου - 1



Σχήμα 3.3 Η Γειτονιά Τύπου - 2

Στα Σχήματα 3.2 και 3.3 βλέπουμε επίσης και τη *μέθοδο της παράθεσης (tiling method)* σε εφαρμογή. Αυτή η μέθοδος χρησιμοποιείται για να μειώσει τον αριθμό των εγγραφών που χρειάζεται να κάνουμε προκειμένου να επιβάλουμε ένα διάνυσμα. Βασίζεται στην εξής παρατήρηση: κάθε κύτταρο είναι ουσιαστικά μέλος σε k γειτονιές: σε μία όπου παίζει το ρόλο του κυττάρου βάσης και σε άλλες k-1 όπου παίζει το ρόλο του μέλος της υπολειπόμενης γειτονιάς, στις οποίες κύτταρο βάσης είναι κάθε ένα από τα υπόλοιπα k-1 κύτταρα.

Στην περίπτωση για παράδειγμα της γειτονιάς Τύπου – 1 κάθε κύτταρο συμμετέχει σε πέντε γειτονιές. Αυτές οι πέντε γειτονιές για το κύτταρο 0 φαίνονται στο Σχήμα 3.4. Όταν εκτελεί μετάβαση για παράδειγμα το κύτταρο 0, στην γειτονιά όπου παίζει το ρόλο του κυττάρου βάσης ελέγχουμε ένα PNPSF σφάλμα και στις υπόλοιπες τέσσερις ένα ANPSF σφάλμα.

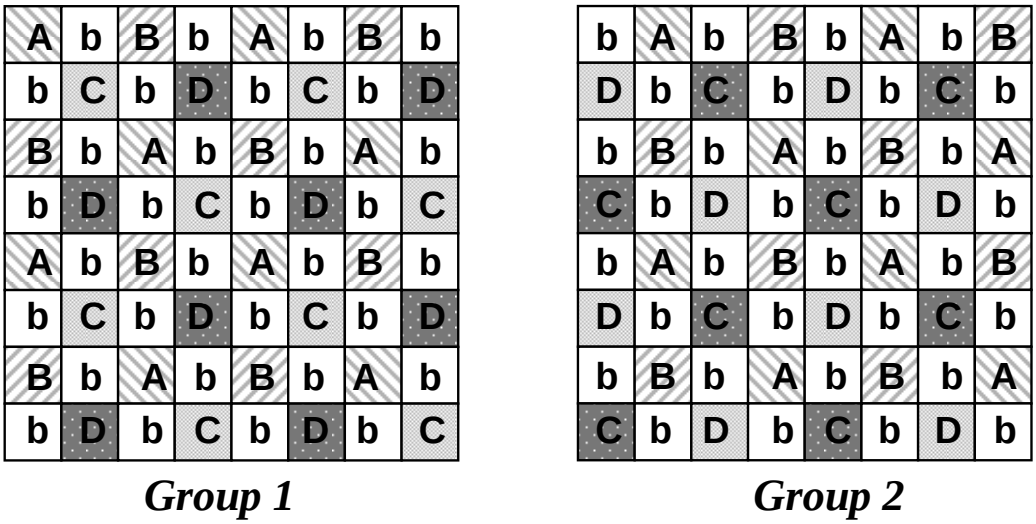
3	1	0	2	4	3	1	0	2	4
0	2	4	3	1	0	2	4	3	1
4	3	1	0	2	4	3	1	0	2
1	0	2	4	3	1	0	2	4	3
2	4	3	1	0	2	4	3	1	0
3	1	0	2	4	3	1	0	2	4
0	2	4	3	1	0	2	4	3	1
4	3	1	0	2	4	3	1	0	2
1	0	2	4	3	1	0	2	4	3
2	4	3	1	0	2	4	3	1	0

Σχήμα 3.4 Οι Πέντε Γειτονιές Τύπου - 1 Ενός Κυττάρου

Αντίστοιχες παρατηρήσεις μπορούν να γίνουν για τη γειτονιά Τύπου – 2. Με τη βοήθεια αυτής της μεθόδου το πλήθος των πράξεων εγγραφής σε μία μνήμη μειώνεται περίπου κατά έναν παράγοντα k, αφού με την εγγραφή ενός κυττάρου μεταβάλλουμε το διάνυσμα ελέγχου σε k γειτονιές. Να σημειώσουμε ότι αυτή η μέθοδος εφαρμόζεται μόνο στις περιπτώσεις όπου η γειτονία έχει κατάλληλο σχήμα έτσι ώστε να είναι δυνατό να γεμίσουμε όλη την επιφάνεια της μνήμης με μία ομάδα από γειτονιές οι οποίες δεν αλληλοεπικαλύπτονται, όπως έχουμε κάνει στα Σχήματα 3.2 και 3.3 για τις γειτονιές Τύπου – 1 και 2. Για τους τύπους γειτονιάς που είναι δυνατό αυτό, οποιοδήποτε κύτταρο αν διαλέξουμε σαν κύτταρο βάσης, όλα τα κύτταρα που ανήκουν στη γειτονιά του φέρουν διαφορετικό αριθμό τόσο με αυτό όσο και μεταξύ τους.

Για τη μείωση του αριθμού των εγγραφών εκτός από τη μέθοδο της παράθεσης υπάρχει και η μέθοδος των δύο ομάδων (two group method) [5]. Στη μέθοδο αυτή χωρίζουμε τη μνήμη σε δύο ομάδες, την Ομάδα 1 και την Ομάδα 2 με τον ίδιο ακριβώς τρόπο που μία σκακιέρα χωρίζεται σε άσπρα και μαύρα τετράγωνα. Στο Σχήμα 3.5 βλέπουμε τις δύο αυτές ομάδες. Σε κάθε μία το κύτταρο βάσης σημειώνεται με b και με A, B, C και D τα κύτταρα της

υπολειπόμενης γειτονιάς. Βλέπουμε ότι κάθε κελί - κύτταρο που είναι κύτταρο βάσης στη μία ομάδα είναι κύτταρο της υπολειπόμενης γειτονιάς στην άλλη και αντιστρόφως. Με αυτό τον τρόπο κάθε κύτταρο υπολειπόμενης γειτονιάς ανήκει σε τέσσερις γειτονιές, οπότε επηρεάζει τέσσερα κύτταρα βάσης. Κάθε ομάδα αποτελείται από n/2 κύτταρα βάσης και n/2 κύτταρα υπολειπόμενης γειτονιάς τα οποία χωρίζονται σε 4 υποομάδες (όπου n είναι ο συνολικός αριθμός των κυττάρων μνήμης). Κάθε υποομάδα αποτελείται από n/8 κύτταρα και απαρτίζεται είτε από όλα τα κύτταρα - A ή από όλα τα B ή από όλα τα C ή από όλα τα D. Ένα διάνυσμα ελέγχου μπορεί να εφαρμοστεί σε όλα τα n/2 κύτταρα μίας ομάδας γράφοντας όλα τα n/8 κύτταρα μίας υποομάδας και έτσι μειώνεται ο αριθμός των εγγραφών κατά έναν παράγοντα 4. Η μέθοδος αυτή είναι εξίσου αποτελεσματική ως προς τη μείωση του κόστους με τη μέθοδο της παράθεσης αλλά δεν είναι τόσο γενική ώστε να μπορεί να εφαρμοστεί παντού. Για παράδειγμα, είδαμε πως στην γειτονιά Τύπου - 2 η μέθοδος της παράθεσης εφαρμόζεται, ενώ αντίθετα η μέθοδος των δύο ομάδων δε μπορεί να εφαρμοστεί. Αυτό συμβαίνει γιατί στη γειτονία Τύπου - 2 έχουμε δύο ειδών κύτταρα υπολειπόμενης γειτονιάς: τα τέσσερα εσωτερικά και τα τέσσερα γωνιακά.



Σχήμα 3.5 Η Μέθοδος των Δύο Ομάδων στην Τύπου – 1 Γειτονιά

Ο αλγόριθμος που χρησιμοποιεί την γειτονιά Τύπου – 1 και τη μέθοδο της παράθεσης ονομάζεται στη βιβλιογραφία *TLAPNPSF1T*. Τα αρχικά του αλγορίθμου προκύπτουν από τη φράση: Test and Locate APNPSF 1 - Tiling, δηλαδή έλεγχος και ανίχνευση APNPSF σφαλμάτων με χρήση της γειτονιάς Τύπου - 1 και της μεθόδου της παράθεσης. Μία απλή έκφραση του αλγορίθμου που συναντάται στη βιβλιογραφία είναι αυτή που ακολουθεί.

```
write 0 in all cells;
read 0 from all cells;
for  j = 1 to k* 2^k do
{
apply_patern( j );
read_all_cells;
}
```

Αλγόριθμος 3.1: Ο Αλγόριθμος TLAPNPSF1T

Αρχικά γράφουμε ολόκληρη τη μνήμη με το πρώτο διάνυσμα που αντιστοιχεί στον πρώτο κόμβο του γράφου του Euler και στη συνέχεια τη διαβάζουμε. Συνήθως σαν πρώτο κόμβο του γράφου επιλέγουμε αυτόν που έχει όλα τα bit 0 και για αυτό βλέπουμε στον αλγόριθμο την εντολή 'write 0 in all cells'. Αυτή είναι η αρχικοποίηση της μνήμης. Ακολούθως διαβάζουμε όλα τα κύτταρα της μνήμης, με αναμενόμενη λογική τιμή 0 για κάθε ένα. Στη συνέχεια εφαρμόζουμε κάθε ένα διάνυσμα από τα $k*2^k$ (συνάρτηση apply_pattern(j)) και μετά από κάθε εφαρμογή διαβάζουμε τη μνήμη και συγκρίνουμε τις αναμενόμενες τιμές των κυττάρων με τις αναμενόμενες (συνάρτηση read_all_cells). Να σημειώσουμε ότι μόνο στην αρχικοποίηση της μνήμης γράφουμε όλα τα κύτταρα. Για αυτό το λόγο η συνάρτηση apply_patern(j) γράφει κάθε φορά μόνο τα κύτταρα που πρέπει να αλλάξουν τιμή ώστε να εφαρμοστεί το επόμενο διάνυσμα. Όπως αναφέρθηκε, ο αριθμός αυτών των κυττάρων είναι n/k.

Υποθέτουμε τώρα ότι το χρονικό κόστος των πράξεων εγγραφής και ανάγνωσης είναι το ίδιο, μία υπόθεση που συνηθίζεται σε παρόμοιες μελέτες [5], και ότι η μνήμη εκτελεί λειτουργία bit, δηλαδή με κάθε πράξη ανάγνωσης ή εγγραφής διαβάζουμε ή γράφουμε μόνο ένα bit. Με αυτές τις δύο υποθέσεις θα υπολογίσουμε το κόστος ενός αλγορίθμου ελέγχου που χρησιμοποιεί τη γειτονιά Τύπου - 1 και τη μέθοδο της παράθεσης. Αρχικά έχουμε να επιβάλλουμε στην μνήμη το πρώτο διάνυσμα, πράγμα του σημαίνει ότι θα γράψουμε όλη τη μνήμη και στη συνέχεια θα τη διαβάσουμε. Αυτή την πράξη αρχικοποίησης την κάνουμε ώστε να μεταβούμε στην κατάσταση που απαιτεί ο πρώτος κόμβος του γραφήματος του Euler, ώστε στη συνέχεια να μπορούμε να διασχίσουμε όλα τα τόξα αλλάζοντας ένα μόνο bit του διανύσματος κάθε φορά. Υποθέτουμε ότι η μνήμη έχει n κύτταρα.

Πρόκειται να γράψουμε και να διαβάσουμε $k*2^k+1$ φορές τη μνήμη, άρα εύκολα βλέπουμε ότι το κόστος των πράξεων ανάγνωσης είναι $n*(k*2^k+1)$. Για το κόστος των εγγραφών παρατηρούμε το εξής: στην αρχικοποίηση της μνήμης πρέπει να γράψουμε όλα τα κύτταρα (n εγγραφές) ενώ στη συνέχεια για την επιβολή των $k*2^k$ υπολοίπων διανυσμάτων χρειαζόμαστε $(n/k)*k*2^k = n*2^k$ εγγραφές. Συνολικά λοιπόν το κόστος των εγγραφών είναι $n*(2^k+1)$. Άρα για μνήμη με n κύτταρα και γειτονιά μεγέθους k, όπου μπορεί να εφαρμοστεί η μέθοδος της παράθεσης, το συνολικό κόστος ελέγχου για σφάλματα είναι ίσο με $n*(k*2^k+2^k+1)$. Για τη γειτονιά Τύπου – 1 αυτό το κόστος γίνεται 194*n και για τη γειτονιά Τύπου – 2 είναι 5122*n. Το κόστος της γειτονιάς Τύπου – 2 είναι φυσικά απαγορευτικά μεγάλο και για αυτό δεν θα ασχοληθούμε καθόλου στη συνέχεια με αυτή.

Ακολούθως θα παρουσιάσουμε μία πιο αναλυτική μορφή του αλγορίθμου η οποία θα είναι χρήσιμη στη συνέχεια της παρούσας εργασίας. Στο Σχήμα 3.2 βλέπουμε την αρίθμηση των κυττάρων όταν εφαρμόζουμε τη γειτονιά Τύπου – 1. Παρατηρούμε ότι σε κάθε Word – Line εμφανίζονται πέντε διαφορετικοί αριθμοί, όσα είναι και τα νούμερα των κυττάρων στην αρίθμηση αυτής της γειτονιάς. Επίσης, ξεκινώντας από την πρώτη (από πάνω) Word – Line και προχωρώντας προς τα κάτω παρατηρούμε ότι κάθε φορά η αρίθμηση των κυττάρων της επόμενης Word – Line προκύπτει αν ολισθήσουμε αυτή της προηγούμενης κατά δύο θέσεις αριστερά. Επομένως ένας τρόπος για να επιβάλουμε ένα διάνυσμα είναι να το γράψουμε σε έναν 5 bit καταχωρητή και διαδοχικά να το επιβάλουμε επαναλαμβανόμενα σε ολόκληρη τη Word – Line. Καθώς προχωρούμε στην επόμενη Word – Line θα κάνουμε το ίδιο αφού πρώτα ολισθήσουμε το διάνυσμα (κυκλικά) δύο θέσεις αριστερά για να ταιριάζει με την αρίθμηση των κυττάρων της. Αυτή η διαδικασία γίνεται και στην ανάγνωση – σύγκριση των δεδομένων της μνήμης με το διάνυσμα, έτσι ώστε να συγκρίνουμε το σωστό bit του διανύσματος με την τιμή κάθε κυττάρου της μνήμης.

Να σημειώσουμε ότι η αντιστοίχηση των ψηφίων του διανύσματος ελέγχου με τους αριθμούς των κυττάρων δεν οφείλει να είναι κάποια συγκεκριμένη, απλά πρέπει να είναι η ίδια σε όλη τη διάρκεια της εκτέλεσης του αλγορίθμου. Με αυτή την έννοια, ούτε η αρίθμηση των κυττάρων οφείλει να είναι αυτή που βλέπουμε στο Σχήμα 3.2. Αρκεί η πρώτη πεντάδα αριθμών στην πρώτη Word – Line να επαναλαμβάνεται με τον ίδιο τρόπο σε ολόκληρη τη Word – Line και σε κάθε περίπτωση η αρίθμηση των κυττάρων της επόμενης Word – Line να προκύπτει από αυτή της προηγούμενης με ολίσθηση δύο θέσεις αριστερά.

Στον αλγόριθμο που ακολουθεί υποθέτουμε ότι έχουμε μία συστοιχία μνήμης με p Word – Lines. Χρησιμοποιούμε δύο καταχωρητές των 5 bit που τους ονομάζουμε A και P και που στον αλγόριθμο εμφανίζονται σαν μονοδιάστατοι πίνακες bits. Ο καταχωρητής P έχει τη δυνατότητα να ολισθαίνει κυκλικά τα ψηφία του m θέσεις προς τα αριστερά (συνάρτηση *shift_left_pattern(P, m)* στον αλγόριθμο).

Η συνάρτηση *generate_pattern(j)* παράγει το επόμενο 5 – bit διάνυσμα που αντιστοιχεί στον επόμενο κόμβο του γράφου του Euler [20]. Το διάνυσμα αποθηκεύεται και στους δύο καταχωρητές A και P. Ο καταχωρητής A χρησιμεύει για να κρατάει την αρχική μορφή του διανύσματος, πριν τις διαδικασίες ολίσθησης, και έτσι να μπορούμε να επαναφέρουμε τον P στην αρχική του κατάσταση. Η συνάρτηση *apply_pattern(P,i)* εφαρμόζει το διάνυσμα που βρίσκεται στον καταχωρητή P στην i - στή Word – Line της μνήμης ενώ η συνάρτηση *compare_pattern(P, i)* συγκρίνει το διάνυσμα του P με τα δεδομένα της i - στής Word – Line και αν βρει διαφορά διακόπτει τη διαδικασία ελέγχου με αποτέλεσμα ‘αποτυχία’. Η διαφορά μεταξύ *apply_pattern(P,i)* και *apply_to_all_pattern(P,i)* είναι ότι η δεύτερη γράφει όλα τα κύτταρα μνήμης ενώ η πρώτη γράφει μόνο αυτά που χρειάζεται να κάνουν μετάβαση. Έτσι, η δεύτερη χρησιμοποιείται στην εφαρμογή του πρώτου διανύσματος στη μνήμη, όπου πρέπει να γραφεί όλη η μνήμη, ενώ η πρώτη εφαρμόζει όλα τα υπόλοιπα διανύσματα, γράφοντας μόνο τα κύτταρα που πρέπει να αλλάξουν τιμή. Η συνάρτηση *read_memory(P)* διαβάζει τη μνήμη και συγκρίνει τα δεδομένα με το διάνυσμα που βρίσκεται στον καταχωρητή P.

Στις μνήμες λειτουργίας bit η συνάρτηση apply_pattern(P, i) απλά γράφει τα κύτταρα μνήμης που πρέπει να πραγματοποιήσουν μετάβαση. Όμως σε μνήμες λειτουργίας λέξης θα πρέπει να γραφεί ολόκληρη η λέξη στην οποία ανήκει ένα κύτταρο που πρέπει να αλλάξει τιμή. Έτσι, το κύτταρο που πρέπει να αλλάξει θα γραφεί με τη νέα του τιμή ενώ τα υπόλοιπα κύτταρα της λέξης θα ξαναγραφούν με την ίδια τιμή.

Προηγουμένως υπολογίσαμε το κόστος του αλγορίθμου για μνήμες λειτουργίας bit. Θα υπολογίσουμε τώρα το κόστος εφαρμογής του αλγορίθμου σε μία μνήμη λειτουργίας λέξης, για μήκος λέξης $w = 2^q$ bits. Παρατηρούμε αρχικά ότι θα χρειαστούμε $n*(k*2^{k-q} + 2^{-q})$ πράξεις ανάγνωσης, αφού με κάθε πράξη ανάγνωσης πλέον διαβάζουμε w bits.

```
// Εφαρμογή και ανάγνωση του πρώτου διανύσματος

BIT MATRIX A[5], P[5];
A = P = generate_pattern(0);
for  i := 1 to p do
        {
        apply_to_all_pattern(P, i);
        shift_left_pattern(P, 2);
        }
P = A;
read_memory(P);

// Εφαρμογή και ανάγνωση των υπολοίπων διανυσματων

for  j = 1 to k* 2^k do
        {
        A = P = generate_pattern(j);

        for  i := 1 to p do
                {
                apply_pattern(P, i);
                shift_left_pattern(P, 2);
                }
        P = A;
        read_memory(P);
        }


function read_memory(P)
        {
        for  i := 1 to p do
                {
                compare_pattern(P, i);
                shift_left_pattern(P,2);
                }
        }
```

Αλγόριθμος 3.2: Ο Αλγόριθμος TLAPNPSF1T σε Αναλυτική Μορφή

Για να υπολογίσουμε τον αριθμό των εγγραφών, παρατηρούμε ότι οι αριθμοί επάνω σε κάθε Word – Line επαναλαμβάνονται με περίοδο 5 (αφού έχουμε 5 διαφορετικούς αριθμούς κυττάρων) και ότι για εφαρμογή ενός διανύσματος χρειάζεται να αλλάξουν τιμή n/k κύτταρα (με εξαίρεση το πρώτο διάνυσμα όπου χρειάζεται να αλλάξουν τιμή και τα n κύτταρα της μνήμης). Αν το μήκος της λέξης είναι 2 ή 4 με κάθε πράξη εγγραφής αλλάζει τιμή ένα μόνο

κύτταρο. Για μήκος λέξης 8 κάποιες φορές αλλάζουμε 1 κύτταρο και κάποιες 2, με μέσο όρο 8/5 =1.6 κύτταρα ανά εγγραφή. Άρα για μήκος λέξης 2^q , q>2 , αλλάζουμε την τιμή σε $2^q/k$ κύτταρα ανά εγγραφή. Σαν γενικό κανόνα μπορούμε να πούμε ότι για τις εγγραφές χρειαζόμαστε συνολικά $n*(2^k/max[1,2^q/k] + 2^{-q})$ πράξεις. Παρατηρούμε ότι οι δύο σχέσεις για τις πράξεις ανάγνωσης και εγγραφής ισχύουν και στην περίπτωση που q=0, δηλαδή καλύπτουν και τις μνήμες λειτουργίας bit.

Άρα το συνολικό κόστος του αλγορίθμου για τη γειτονιά Τύπου – 1 (k=5) είναι αυτό που δίδεται από την Εξ 2.1, μετρούμενο σε πράξεις ανάγνωσης / εγγραφής. Για μήκος λέξης w = 2 χρειαζόμαστε 113*n πράξεις, για w = 4 χρειαζόμαστε 72.5*n ενώ για w = 8 χρειαζόμαστε 40.3*n πράξεις.

$n*(5*2^{5-q} + 2^5/max[1,2^q/5] +2^{-q+1})$ Εξ 2.1

Ο αλγόριθμος που παρουσιάσαμε προηγουμένως για έλεγχο APNPSF σφαλμάτων με χρήση της γειτονιάς Τύπου – 1 υποθέτει ότι η φυσική διάταξη της μνήμης συμπίπτει με τη λογική διάταξη, που είναι ένας διδιάστατος πίνακας. Όπως θα δούμε στη συνέχεια, στην περίπτωση των DRAM εμφωλευμένης συστοιχίας που εξετάζουμε και που είναι η επικρατέστερη σήμερα, αυτή η υπόθεση απέχει πολύ από την πραγματικότητα.

3.4. Μνήμες DRAM και Σφάλματα

Το NPSF μοντέλο που περιγράψαμε είναι ένα γενικό μοντέλο σφαλμάτων για όλους τους τύπους μνημών. Στόχος του είναι να εντοπίσει ‘μικρά’ ελαττώματα σε μνήμες, τα οποία μπορεί να προκαλέσουν λανθασμένη απόκριση κάτω από πολύ συγκεκριμένες συνθήκες. Ελαττώματα τα οποία είναι ικανά να προκαλέσουν λανθασμένη απόκριση κάτω από σχεδόν οποιεσδήποτε συνθήκες είναι πολύ πιο εύκολο να εντοπιστούν. Για παράδειγμα, για να ανιχνεύσουμε αν σε ένα κύτταρο μνήμης υπάρχει ένα σφάλμα μόνιμης τιμής, δηλαδή την περίπτωση όπου το κύτταρο σε κάθε ανάγνωσή του παρουσιάζει πάντα την ίδια λογική τιμή (ανεξαρτήτως του διανύσματος που έχει εφαρμοστεί στα γειτονικά του), αρκεί να γράψουμε και να διαβάσουμε όλα τα κύτταρα με τις δύο δυνατές λογικές τιμές τους. Ένα Στατικό NPSF σφάλμα όμως, που αποτελεί μια πιο γενική περίπτωση του σφάλματος μόνιμης τιμής, απαιτεί όπως είδαμε να εφαρμόσουμε όλα τα δυνατά διανύσματα στα γειτονικά κύτταρα.

Οι μνήμες DRAM όμως έχουν και μία άλλη ιδιαιτερότητα: η αποθηκευτική τους μονάδα, ο πυκνωτής, είναι ένα παθητικό στοιχείο. Ως εκ τούτου, αν διαταραχθεί από την αρχική της κατάσταση δεν έχει τη δυνατότητα να επανέλθει από μόνη της σε αυτή. Αν για παράδειγμα ο πυκνωτής έχει αποθηκευμένη την λογική τιμή 1, οπότε βρίσκεται σε τάση V_{DD} , και εξαιτίας κάποιας επίδρασης στη γειτονιά του πέσει η τάση του σε 0.8 V_{DD} , δεν θα έχει τη δυνατότητα να επανέλθει σε τάση V_{DD} παρά μόνο στην επόμενη διαδικασία ανάγνωσης που θα υποστεί. Στο μεταξύ υπάρχει το περιθώριο να πέσει και άλλο η τάση του, είτε λόγω αλληλεπιδράσεων με τα γειτονικά κύτταρα μνήμης είτε καθαρά λόγω του ρεύματος διαρροής του τρανζίστορ του, και μπορεί να φτάσει τόσο κοντά στο V_{DD} / 2 που να υπάρχει πιθανότητα ο ενισχυτής σήματος είτε να αντιληφθεί λανθασμένη τιμή, είτε να αποκριθεί με τη σωστή τιμή αλλά με καθυστέρηση.

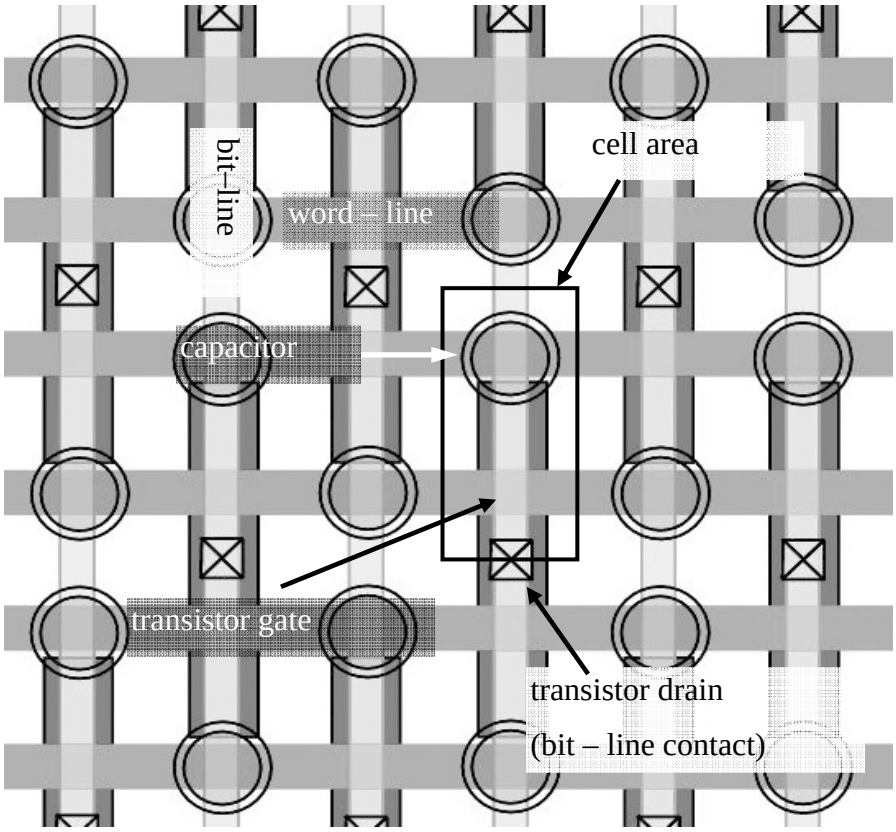
Να σημειώσουμε επίσης ότι όταν μιλάμε για σφάλματα NPSF σε μνήμες DRAM και λέμε ότι κάποιο κύτταρο χάνει τα δεδομένα του, γενικά δεν εννοούμε ότι αποκτά την αντίθετη λογική τιμή από την αναμενόμενη. Συνήθως αυτό που συμβαίνει είναι ότι η τάση του κυττάρου έρχεται τόσο κοντά στο V_{DD} / 2 που ο ενισχυτής σήματος δεν είναι σίγουρο ποια τιμή θα αντιληφθεί. Λέμε τότε ότι το κύτταρο βρίσκεται σε *κατάσταση αβεβαιότητας*. Σε αυτές τις περιπτώσεις φυσικά αρχίζει και γίνεται θέμα πιθανοτήτων προς ποια λογική τιμή θα κινηθεί ο ενισχυτής σήματος και είναι πολύ πιθανό λόγω τύχης σε ένα κύτταρο που βρίσκεται σε κατάσταση αβεβαιότητας να διαβάσουμε την αναμενόμενη τιμή. Με αυτό τον τρόπο μπορεί ακόμα και ένας άριστος αλγόριθμος ελέγχου να αποτύχει να εντοπίσει το σφάλμα σε μία DRAM.

ΚΕΦΑΛΑΙΟ 4. ΠΡΟΣΑΡΜΟΓΗ ΤΗΣ ΓΕΙΤΟΝΙΑΣ ΤΥΠΟΥ - 1 ΣΕ ΜΝΗΜΕΣ DRAM

4.1. Εφαρμογή Της Γειτονιάς Τύπου – 1 σε DRAM Εμφωλευμένης Συστοιχίας

Η δομή της DRAM τύπου εμφωλευμένης συστοιχίας φαίνεται στο Σχήμα 4.1, το οποίο προσεγγίζει τις πραγματικές αναλογίες των στοιχείων με εξαίρεση το πλάτος των Bit – Lines και Word – Lines τα οποία φαίνονται πιο μικρά για να είναι πιο εύκολα κατανοητό το σχήμα. Βλέπουμε ότι τα κύτταρα είναι διευθετημένα σε ζευγάρια με κοινή υποδοχή και επαφή με τη Bit – Line. Ο κάθε ενισχυτής σήματος εξυπηρετεί δύο διαδοχικές Bit – Lines, των οποίων τα τρανζίστορ ενεργοποιούνται από δύο διαφορετικές ομάδες Word – Lines, έτσι ώστε κάθε φορά μόνο η μία Bit – Line του ζεύγους να ενεργοποιείται.

Ας προσπαθήσουμε τώρα να αντιστοιχήσουμε την Τύπου – 1 γειτονιά όπως ορίστηκε στο προηγούμενο κεφάλαιο στην πραγματική δομή της DRAM τύπου εμφωλευμένης συστοιχίας. Παρατηρούμε καταρχήν ότι η συστοιχία μνήμης στην πραγματικότητα δεν είναι πίνακας, αφού σε έναν πίνακα για κάθε συνδυασμό γραμμής – στήλης έχουμε ένα κελί, ενώ εδώ κάθε Word – Line συνδυάζεται μόνο με τις μισές Bit – Lines. Βέβαια, ένα εξωτερικό κύκλωμα την βλέπει σαν πίνακα γιατί για κάθε ζεύγος BL και $\overline{\text{BL}}$ βλέπει την έξοδο ενός πολυπλέκτη που επιλέγει μία από τις δύο. Προφανώς όμως, σε κάθε *φαινόμενη Bit – Line* αντιστοιχούν ένα ζεύγος από *πραγματικές* ή *φυσικές Bit – Lines*. Έτσι, καθώς μετακινούμαστε από τη μία Word – Line στην επόμενη, ενώ παραμένουμε στην ίδια φαινόμενη Bit – Line, δεν βρισκόμαστε πάντα στην ίδια φυσική Bit – Line.

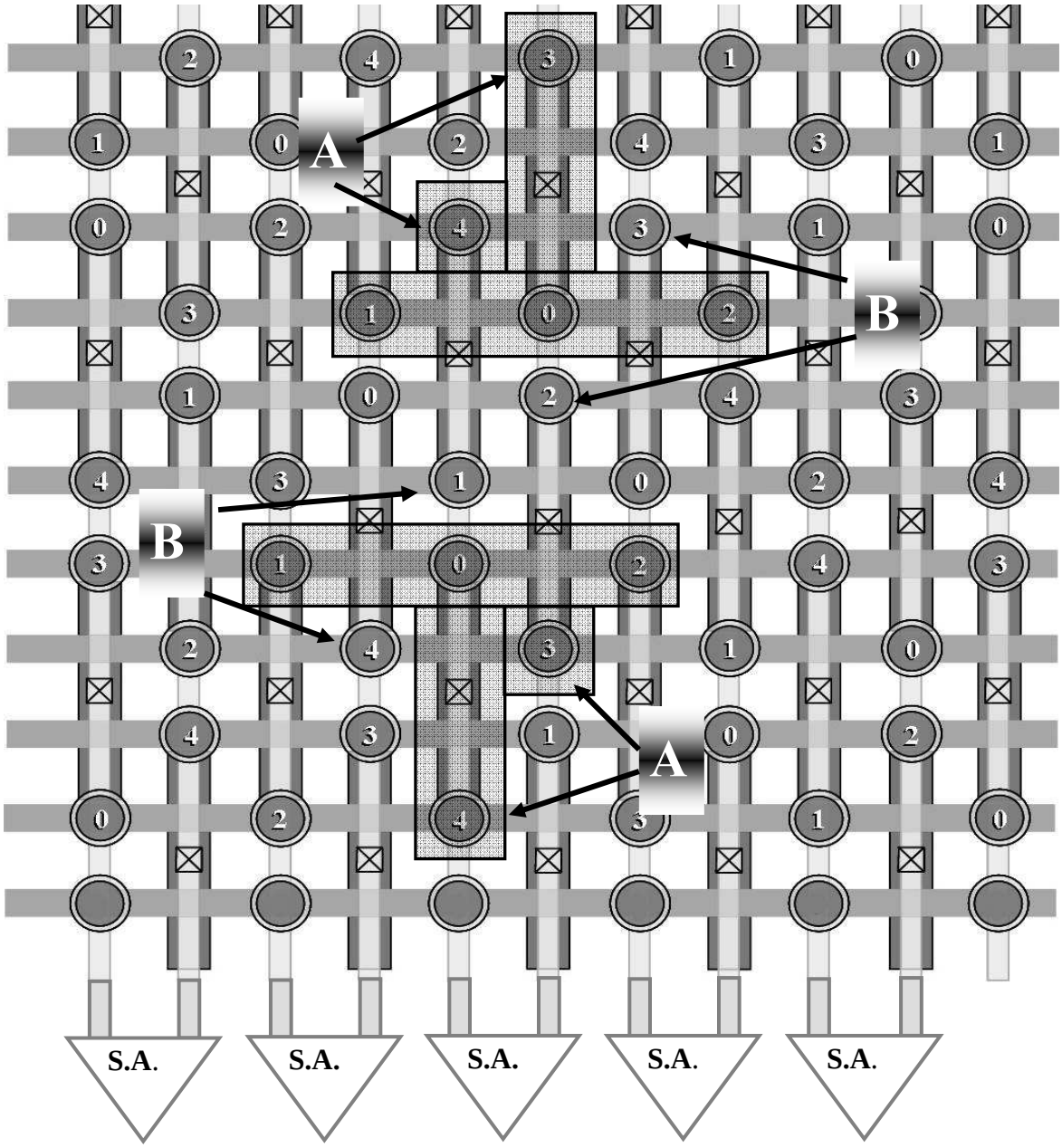


Σχήμα 4.1 Δομή Μνήμης Εμφωλευμένης Συστοιχίας

Στο Σχήμα 3.2 είδαμε την εφαρμογή της γειτονιάς Τύπου – 1 σε πίνακα, υποθέτοντας το 0 ως κύτταρο βάσης. Αν προσπαθήσουμε να μεταφέρουμε αυτή την εικόνα πάνω στην πραγματική μνήμη θα πάρουμε την εικόνα που βλέπουμε στο Σχήμα 4.2. Όπως βλέπουμε, μόνο δύο από τα κύτταρα της γειτονιάς είναι πραγματικά κοντά στο κύτταρο βάσης, αυτά που σημειώνονται ως A ενώ δύο άλλα που είναι επίσης κοντά, αυτά που σημειώνονται ως B, έχουν τεθεί εκτός γειτονιάς.

Από το ίδιο σχήμα μπορούμε επίσης να δούμε εύκολα ότι τα τέσσερα κοντινότερα στο κύτταρο βάσης είναι αυτά ακριβώς που σημειώνονται με τα γράμματα A και B, δηλαδή τα δύο που βρίσκονται εκατέρωθεν στην ίδια Bit – Line (και που μόνο το ένα εκ των δύο έχει Word – Line γειτονική σε αυτή του κυττάρου βάσης) και τα δύο άλλα που βρίσκονται σε εκατέρωθεν γειτονικές φυσικές Bit – Lines και ενεργοποιούνται από γειτονική Word – Line.

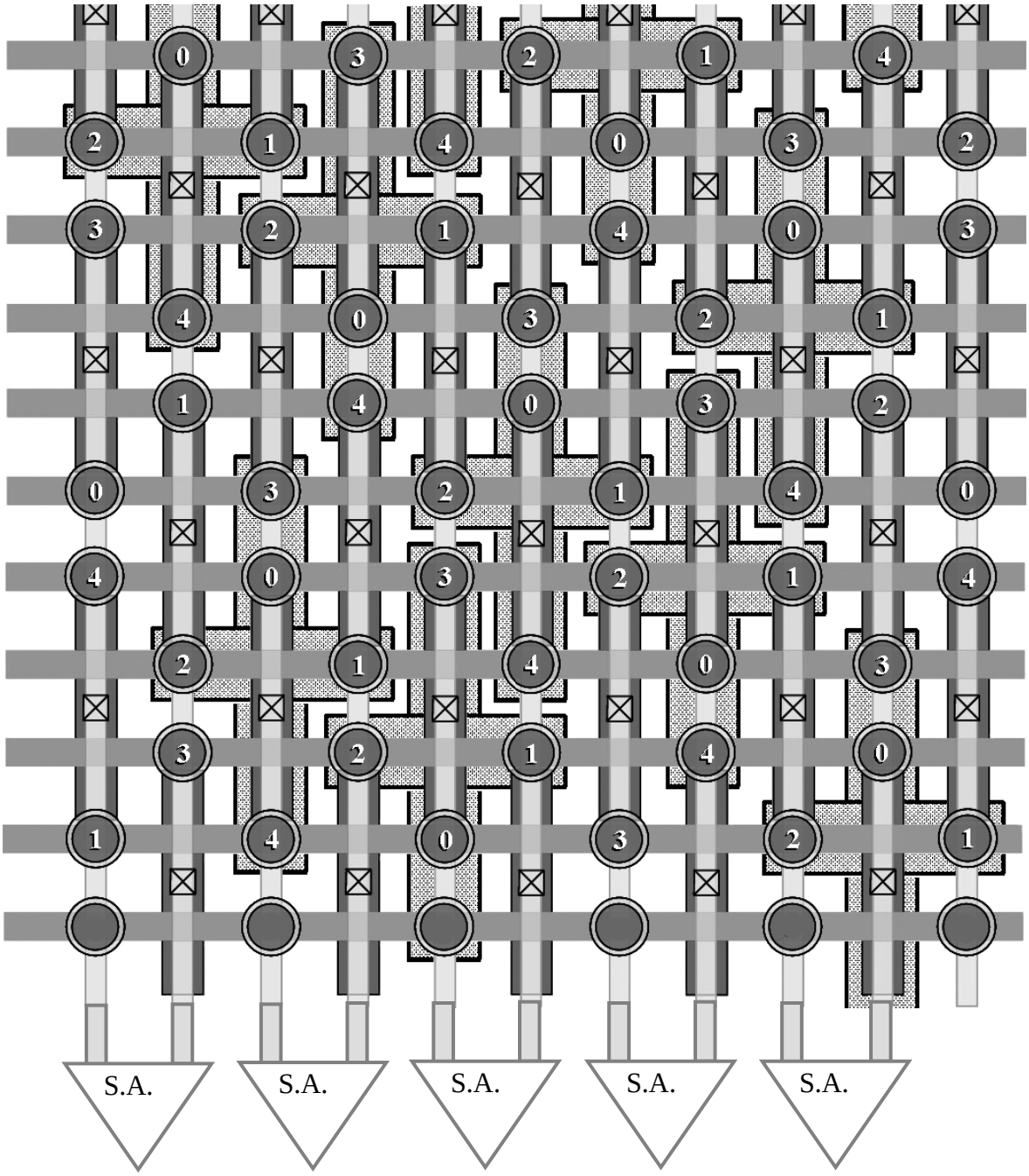
Ένα μόνο εκ των δύο τελευταίων ανήκει στην ίδια φαινόμενη Bit – Line, ενώ το άλλο σε γειτονική. Μία άλλη επίσης σημαντική παρατήρηση από το ίδιο σχήμα είναι ότι στην αρίθμηση που μας επιβάλλει η Τύπου – 1 γειτονιά κάποια από αυτά τα πέντε κύτταρα (το κύτταρο βάσης και τα τέσσερα γειτονικά) φέρουν τον ίδιο αριθμό και συνεπώς δεν είναι δυνατόν να εφαρμοστούν σε αυτά όλα τα διανύσματα ελέγχου, όπως απαιτεί το μοντέλο NPSF.



Σχήμα 4.2 Απεικόνιση της Γειτονιάς Τύπου – 1 στο Πραγματικό Διάγραμμα της Μνήμης

Για να προσαρμόσουμε την γειτονιά Τύπου – 1 στην πραγματική δομή της μνήμης θα προτείνουμε μια γειτονιά χρησιμοποιώντας τα 4 κοντινότερα στο κύτταρο βάσης κύτταρα μνήμης. Μία τέτοια γειτονιά φαίνεται στο Σχήμα 4.3, όπου φαίνεται ότι μπορεί να εφαρμοστεί η μέθοδος της παράθεσης. Αυτή τη νέα γειτονιά θα την ονομάσουμε Π*ροσαρμοσμένη Τύπου – 1* γειτονιά *(Προ.Τύπου - 1* γειτονιά *– Adapted Type -1* neighborhood*).*

Ακολουθώντας την αντίστροφη διαδικασία, θα μεταφέρουμε τώρα την αρίθμηση των κυττάρων της Προ. Τύπου – 1 γειτονιάς πάνω σε πίνακα, προκειμένου να δομήσουμε έναν αλγόριθμο ελέγχου ορθής λειτουργίας που θα τη χρησιμοποιεί. Η μεταφορά αυτή σε πίνακα φαίνεται στο Σχήμα 4.4. Για να κατανοήσουμε το σχήμα υπενθυμίζουμε ότι οι Bit – Lines μοιράζονται ανά δύο τον ίδιο αισθητήρα σήματος και σε κάθε ζεύγος οι δύο Bit – Lines που το αποτελούν ενεργοποιούνται από διαφορετική ομάδα Word – Lines, πράγμα που σημαίνει ότι έχουμε δύο ομάδες από Word – Lines. Τις δύο αυτές ομάδες θα ονομάσουμε *Λευκή* και *Γκρίζα* ομάδα, όνομα το οποίο αναφέρεται στο χρώμα φόντου που έχουν οι αντίστοιχες γραμμές στο σχήμα. Με άλλα λόγια, σε κάθε ζεύγος από φυσικές Bit – Lines, η μία Bit – Line ενεργοποιείται από τις λευκές Word – Lines και η άλλη από τις γκρίζες. Φυσικά δεν έχει απολύτως καμία σημασία ποια ομάδα ονομάζουμε Λευκή και ποια Γκρίζα, απλά χρειάζεται να γνωρίζουμε ποιες Word – Lines ανήκουν στην ίδια ομάδα και ποιες όχι. Σε ένα ζεύγος γειτονικών Word – Lines της ίδιας ομάδας (ίδιο χρώμα φόντου) θα αναφερόμαστε στην επάνω ως *ανώτερη* (λευκή ή γκρίζα) και στην κάτω ως *κατώτερη*.



Σχήμα 4.3 Η Προσαρμοσμένη Γειτονιά Τύπου - 1

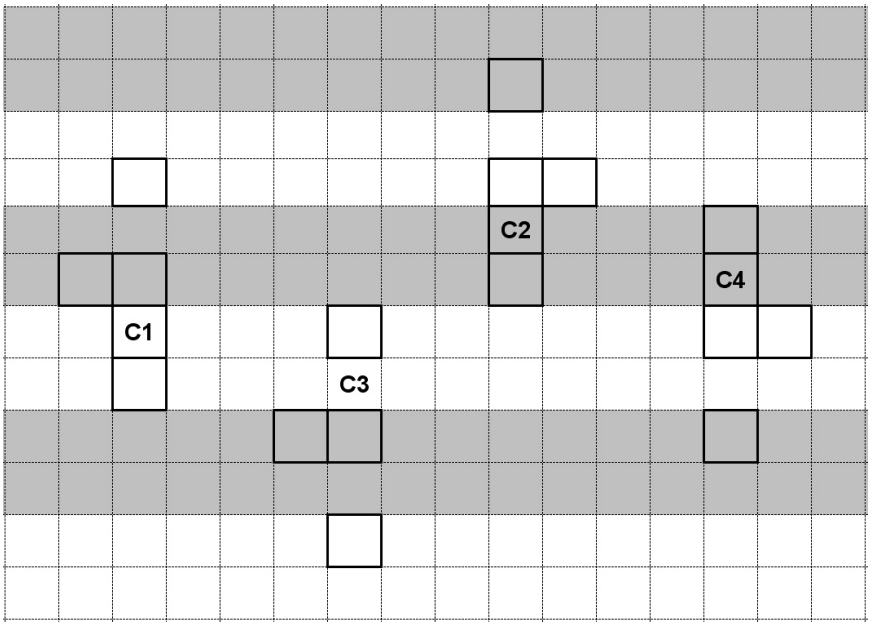
Στη συνέχεια θα μεταφέρουμε την αναπαράσταση της Προ. Τύπου – 1 γειτονιάς (δηλ. τον σταυρό) από το φυσικό επίπεδο στην αναπαράσταση σε πίνακα, η οποία φαίνεται στο Σχήμα 4.5. Σε αυτό το σχήμα τα κύτταρα βάσης σημειώνονται με C1, C2, C3, C4 δείχνοντας έτσι τις τέσσερις πιθανές θέσεις στις οποίες μπορεί να βρίσκεται ένα κύτταρο βάσης που μας

ενδιαφέρει, δηλαδή σε μία ανώτερη λευκή Word – Line (C1), σε μία ανώτερη γκρίζα Word – Line (C2), σε μία κατώτερη λευκή Word – Line (C3) ή σε μία κατώτερη γκρίζα Word – Line (C4). Η αρίθμηση των κυττάρων δεν εμφανίζεται επίτηδες σε αυτό το σχήμα γιατί δεν μας ενδιαφέρει, αφού η τοπολογία της γειτονιάς ενός κυττάρου στο επίπεδο του πίνακα δεν εξαρτάται από τον αριθμό του κυττάρου αλλά από το είδος της Word – Line στην οποία αυτό βρίσκεται (ανώτερη ή κατώτερη, άσπρη ή γκρίζα). Η χρησιμότητα αυτού του σχήματος θα φανεί σε επόμενη ενότητα όπου θα μιλήσουμε για απόκρυψη σφαλμάτων σε μνήμες λειτουργίας λέξης.

Με τη βοήθεια της αρίθμησης που παρουσιάζουμε στο Σχήμα 4.4 μπορούμε τώρα να προτείνουμε έναν αλγόριθμο ο οποίος θα εκτελεί τον έλεγχο ορθής λειτουργίας χρησιμοποιώντας την Προ. Τύπου – 1 γειτονιά. Η μόνη διαφοροποίηση με την κλασική Τύπου – 1 γειτονιά είναι ο τρόπος που κάνουμε ολίσθηση στο διάνυσμα (δηλ στον καταχωρητή P) όταν πηγαίνουμε στην επόμενη Word – Line. Παρατηρούμε πως όταν η επόμενη Word – Line αναφέρεται στην ίδια φυσική Bit – Line με την προηγούμενη, κάνουμε ολίσθηση κατά 1 bit δεξιά. Όταν μεταβαίνουμε σε Word - Line η οποία αναφέρεται σε άλλη φυσική Bit – Line, δηλαδή αν αλλάζουμε χρώμα Word – Line, τότε: α) αν πηγαίνουμε από λευκή σε γκρίζα κάνουμε ολίσθηση 2 bit δεξιά, ενώ β) αν πηγαίνουμε από γκρίζα σε λευκή ολισθαίνουμε 2 bit αριστερά.

2	1	4	0	3	2	1	4	0	3
0	3	2	1	4	0	3	2	1	4
4	0	3	2	1	4	0	3	2	1
3	2	1	4	0	3	2	1	4	0
0	3	2	1	4	0	3	2	1	4
1	4	0	3	2	1	4	0	3	2
2	1	4	0	3	2	1	4	0	3
4	0	3	2	1	4	0	3	2	1

Σχήμα 4.4 Η Μεταφορά της Αρίθμησης της Προσαρμοσμένης Τύπου – 1 Γειτονιάς σε Πίνακα



Σχήμα 4.5 Η Προσαρμοσμένη Τύπου – 1 Γειτονιά σε Πίνακα

Με αυτές τις παρατηρήσεις προτείνουμε τον αλγόριθμο 4.1 τον οποίο έχουμε ονομάσει *TLAPNPSFA1T* (Test and Locate APNPSF with Adapted type – 1 and Tiling method). Έχουμε υποθέσει, χωρίς βλάβη της γενικότητας, ότι οι δύο πρώτες Word – Lines δεν έχουν την ίδια ομάδα φυσικών Bit – Lines και επομένως αλλάζουμε ομάδα φυσικών Bit – Lines μόνο όταν πηγαίνουμε από περιττή σε άρτια Word – Line. Χρησιμοποιούμε και πάλι τους δύο καταχωρητές A και P με τη διαφορά ότι ο P έχει τη δυνατότητα να κάνει κυκλική ολίσθηση m θέσεις και προς τις δύο κατευθύνσεις (συναρτήσεις *shift_left_pattern(P, m)* και *shift_right_pattern(P, m)* στον αλγόριθμο). Χρησιμοποιούμε επίσης την μαθηματική συνάρτηση *mod* που μας δίνει το υπόλοιπο της διαίρεσης μεταξύ δύο ακεραίων, ενώ οι άλλες συναρτήσεις που χρησιμοποιούμε είναι όπως στον προηγούμενο αλγόριθμο. Υποθέτουμε πάλι ότι στη συστοιχία μνήμης έχουμε p Word – Lines.

Το κόστος του αλγορίθμου αυτού είναι σε κάθε περίπτωση ακριβώς ίδιο με το κόστος του αλγορίθμου της κλασικής Τύπου – 1 γειτονιάς, αφού το μόνο που αλλάζει είναι ο τρόπος που κάνουμε ολίσθηση στο διάνυσμα που εφαρμόζουμε στη μνήμη.

```
BIT MATRIX A[5], P[5];

// Εφαρμογή και ανάγνωση του πρώτου διανύσματος

A := P := generate_pattern(0);
for  i := 1 to p do
        {
        apply_to_all_pattern(P,i);

        if ( i mod 2 = 0 ) then shift_right_pattern(P,1);                    //even
        else if (  (i +1) /2 mod 2 ≠ 0 ) then shift_right_pattern(P, 2);     // 1,5,9 …
        else shift_left_pattern(P, 2);                                       // 3,7,11…
        }
P := A;
read_memory(P);

// Εφαρμογή και ανάγνωση των υπολοίπων διανυσματων

for  j = 1 to k* 2^k do
        {
        A := P := generate_pattern(j);

        for  i := 1 to p do
                {
                apply_pattern(P, i);

                if ( i mod 2 = 0 ) then shift_right_pattern(P,1);
                else if (  (i +1) /2 mod 2 ≠ 0 ) then shift_right_pattern(P, 2);
                else shift_left_pattern(P, 2);
                }
        P := A;
        read_memory(P);
        }
```

Η συνάρτηση **read_memory(P)** γίνεται ως εξής:

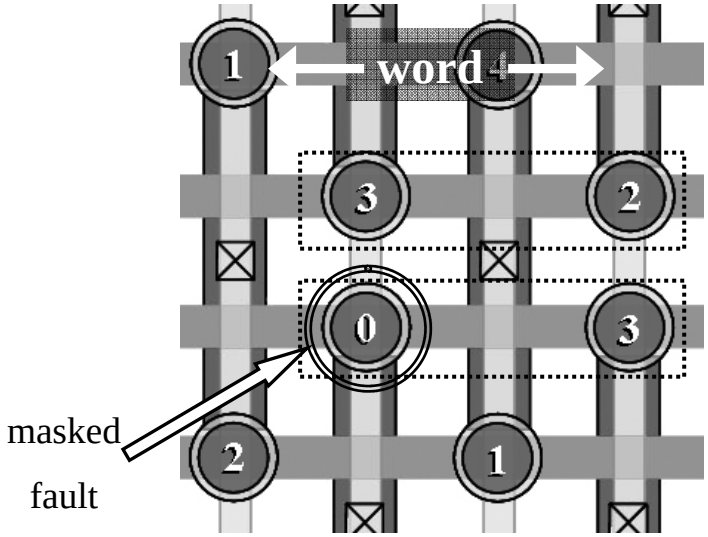
```
        function read_memory(P)
        {
        for  i := 1 to p do
                {
                compare_pattern(P,i);

                if ( i mod 2 = 0 ) then shift_right_pattern(P,1);
                else if (  (i +1) /2 mod 2 ≠ 0 ) then shift_right_pattern(P, 2);
                else shift_left_pattern(P, 2);
                }
        }
```

Αλγόριθμος 4.1: Ο Αλγόριθμος TLAPNPSFA1T

4.2. Απόκρυψη Σφαλμάτων σε DRAM Λειτουργίας Λέξης

Όπως έχουμε προαναφέρει, στις μνήμες λειτουργίας λέξης προκειμένου να αλλάξουμε την τιμή σε ένα κύτταρο μνήμης πρέπει να γράψουμε και τα υπόλοιπα κύτταρα που ανήκουν στην ίδια λέξη. Το γεγονός αυτό όπως θα δούμε μπορεί να οδηγήσει σε *απόκρυψη σφαλμάτων (fault masking)* που λαμβάνει χώρα κατά τη διάρκεια εφαρμογής των προαναφερθέντων αλγορίθμων ελέγχου σφαλμάτων, με αποτέλεσμα να μειώνεται η δυνατότητα κάλυψης σφαλμάτων και συνεπώς η αξιοπιστία των αλγορίθμων.



Σχήμα 4.6 Παράδειγμα Απόκρυψης Σφάλματος

Στο Σχήμα 4.6 βλέπουμε μία περίπτωση απόκρυψης σφάλματος σε μία μνήμη με μήκος λέξης 2. Οι τέσσερις εικονιζόμενες πραγματικές Bit – Lines (που αντιστοιχούν σε δύο φαινόμενες) ανήκουν στην ίδια λέξη. Υποθέτουμε ότι η εφαρμογή του τρέχοντος διανύσματος απαιτεί την αλλαγή τιμής των κυττάρων με τον αριθμό 3 και ότι εκτελούμε αυτή την αλλαγή ενεργοποιώντας τις Word – Lines από πάνω προς τα κάτω. Προκειμένου να αλλάξουμε την τιμή στο κάτω κύτταρο 3 απαιτείται να ξαναγράψουμε και το κύτταρο 0 που βρίσκεται δίπλα του, αφού τα κύτταρα αυτά ανήκουν στην ίδια λέξη και γράφονται ή διαβάζονται μαζί. Αν η μετάβαση που εφαρμόστηκε νωρίτερα στο επάνω κύτταρο 3 είχε προκαλέσει απώλεια δεδομένων στο κύτταρο 0, τότε αυτό το σφάλμα θα είναι αδύνατο να εντοπιστεί γιατί η εγγραφή του κάτω κυττάρου 3 απαιτεί την επανεγγραφή και του κυττάρου 0 με τα σωστά δεδομένα πριν αυτό διαβαστεί καθώς η διαδικασία ανάγνωσης πραγματοποιείται μετά από την εγγραφή όλων των κυττάρων.

Γενικά αυτός ο τύπος απόκρυψης σφάλματος ορίζεται ως εξής: μία εγγραφή που πρέπει να κάνουμε σε μία Word – Line προκειμένου να επιβάλουμε το τρέχον διάνυσμα διορθώνει ένα σφάλμα σε κύτταρο αυτής της Word – Line το οποίο είχε προκληθεί από μετάβαση ενός γειτονικού κυττάρου που ανήκει σε Word – Line που γράψαμε προηγουμένως.

Υπάρχουν δύο τρόποι για να λύσουμε αυτό το πρόβλημα. Ο ένας, με βάση το παράδειγμά μας, είναι να διαβάσουμε το κύτταρο 0 αμέσως αφού γράψουμε το επάνω κύτταρο 3. Αυτό φαίνεται εκ πρώτης όψεως μη συνεπές με τον ορισμό του APNPSF ελέγχου μνήμης, ο οποίος απαιτεί να εφαρμόσουμε πρώτα σε όλη τη μνήμη το διάνυσμα ελέγχου και μετά να κάνουμε ανάγνωση. Όμως, κανένα άλλο κύτταρο στην υπολειπόμενη γειτονιά του κυττάρου 0 δεν πρόκειται να αλλάξει στη συνέχεια της εφαρμογής του διανύσματος και επομένως, σύμφωνα με το μοντέλο μας, τίποτε άλλο δεν πρόκειται να το επηρεάσει μέχρι να ολοκληρωθεί η εγγραφή της μνήμης. Φυσικά σε αυτή την ‘ενδιάμεση’ πράξη ανάγνωσης πρέπει να λάβουμε υπ’ όψιν ότι το κάτω κύτταρο 3 δεν έχει αλλάξει ακόμα τιμή και άρα έχει την παλιά τιμή και όχι την νέα τιμή που του επιβάλλει το τρέχον διάνυσμα. Να σημειώσουμε ότι αυτή η ενδιάμεση ανάγνωση δεν μας απαλλάσσει από την ανάγκη να διαβάσουμε ολόκληρη τη μνήμη όταν ολοκληρωθεί η εγγραφή της, αφού πρέπει να διαπιστώσουμε αν η νέα τιμή γράφηκε με επιτυχία στο κάτω κύτταρο 3 και σε όλα τα κύτταρα 3 (έλεγχος PNPSF σφαλμάτων). Η ανάγνωση αυτή θα έχει σαν αποτέλεσμα και την περιττή εκ νέου ανάγνωση του κυττάρου 0, γεγονός αναπόφευκτο όμως αφού ανήκουν στην ίδια λέξη.

Ο άλλος τρόπος επίλυσης του προβλήματος είναι να εφαρμόσουμε τον αλγόριθμο δύο φορές γράφοντας τις Word – Lines με αντίθετη φορά (δηλ. στην πρώτη περίπτωση από πάνω προς τα κάτω και τη δεύτερη αντίθετα). Με αυτό τον τρόπο λύνουμε το πρόβλημα γιατί σίγουρα σε μία από τις δύο περιπτώσεις θα συμβεί πρώτα η εγγραφή που θα μπορούσε να προκαλέσει απόκρυψη σφάλματος και μετά η εγγραφή που δημιουργεί το σφάλμα αυτό. Όμως η δεύτερη μέθοδος διπλασιάζει το κόστος και αυτό την κάνει ασύμφορη σε σχέση με την πρώτη.

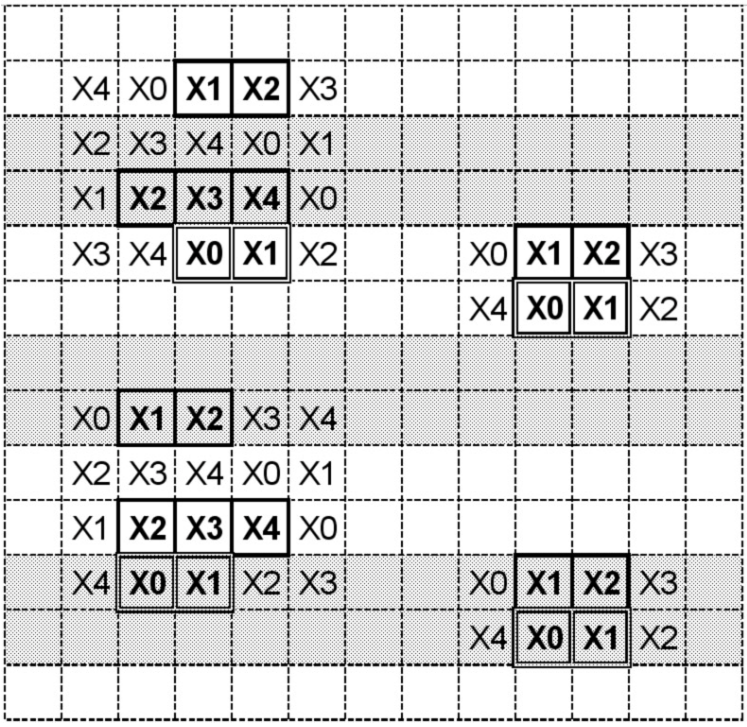
Θα υπολογίσουμε τώρα το επιπλέον κόστος του πρώτου τρόπου διόρθωσης του προβλήματος απόκρυψης σφάλματος στην εφαρμογή του αλγορίθμου της Προ. Τύπου – 1 γειτονιάς. Καταρχήν, στη διάρκεια της εφαρμογής του πρώτου διανύσματος δεν έχει νόημα να κάνουμε ενδιάμεσες αναγνώσεις γιατί η προηγούμενη κατάσταση της μνήμης είναι άγνωστη και έτσι

δεν μπορεί να υπάρξει σύγκριση δεδομένων. Στις υπόλοιπες περιπτώσεις, μία λέξη πρέπει να διαβαστεί πριν γραφεί, αν κάποιο από τα κύτταρά της βρίσκεται στη γειτονιά οποιουδήποτε κυττάρου το οποίο έχει ήδη εκτελέσει μετάβαση κατά την εφαρμογή του τρέχοντος διανύσματος ελέγχου.

Στόχος μας λοιπόν είναι να βρούμε για μία λέξη ποια είναι εκείνα τα κύτταρα που μπορούν να επηρεάσουν ένα ή περισσότερα κύτταρά της. Προκειμένου να το πετύχουμε αυτό θα χρησιμοποιήσουμε το Σχήμα 4.5 της προηγούμενης παραγράφου όπου είχαμε μεταφέρει την τοπολογία της Προ. Τύπου – 1 γειτονιάς (δηλ. τον σταυρό) στην αναπαράσταση σε πίνακα.. Σε αυτό το σχήμα βλέπουμε εύκολα ότι για κύτταρα βάσης που ανήκουν στις δύο ανώτερες (γκρίζες ή άσπρες) Word – Lines έχουμε τρία υποψήφια κύτταρα τα οποία ενδέχεται να έχουν εκτελέσει μετάβαση πριν γραφεί η λέξη στην οποία ανήκει το κύτταρο βάσης (υποθέτουμε ότι οι Word – Lines γράφονται με σειρά από πάνω προς τα κάτω). Αντίθετα, για κύτταρα βάσης που ανήκουν στις δύο κατώτερες Word – Lines έχουμε μόνο ένα.

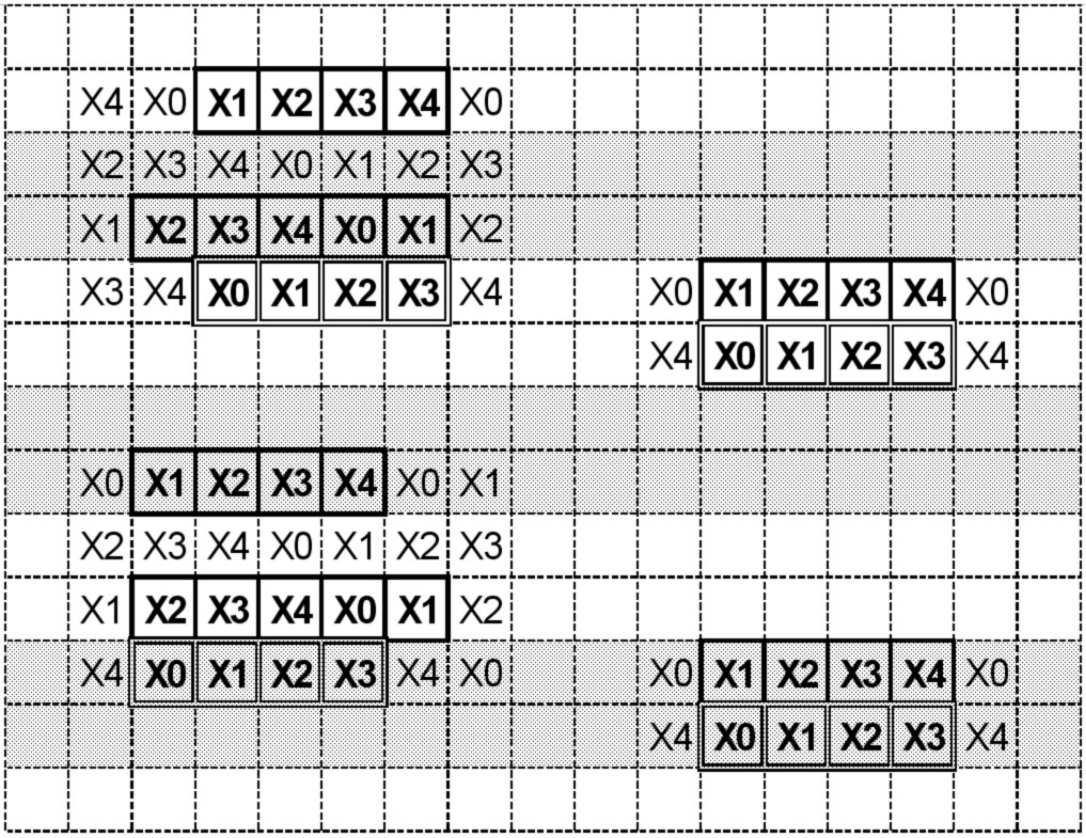
Με τη βοήθεια του προαναφερθέντος σχήματος θα βρούμε αρχικά ποια είναι τα κύτταρα (*κύτταρα – θύτες*) που μπορούν να επηρεάσουν ένα ή περισσότερα κύτταρα μίας λέξης μήκους 2 (*κύτταρα – θύματα*). Το αποτέλεσμα για κάθε περίπτωση φαίνεται στο Σχήμα 4.6 όπου η λέξη που μας ενδιαφέρει έχει περίγραμμα διπλής γραμμής ενώ τα κύτταρα που μπορούν να επηρεάσουν κάποιο κύτταρό της έχουν περίγραμμά με έντονη απλή γραμμή. Τα σύμβολα X0, X1 κτλ που βρίσκονται μέσα στα κελιά θα εξηγηθούν στη συνέχεια.

Θα προσδιορίσουμε τώρα τη συσχέτιση που έχουν τα κύτταρα - θύματα με τα κύτταρα - θύτες. Στο Σχήμα 4.4 και ακολουθώντας τη σειρά αρίθμησης που έχουν τα κύτταρα, βλέπουμε ότι μία διάταξη πέντε διαδοχικών κυττάρων σε μία Word – Line μπορεί να δημιουργηθεί με πέντε διαφορετικούς τρόπους, ανάλογα με τον αριθμό του πρώτου κυττάρου. Οι πέντε αυτές διαφορετικές διατάξεις που προκύπτουν είναι οι ακόλουθες: (3, 2, 1, 4, 0), (2, 1, 4, 0, 3), (1, 4, 0, 3, 2), (4, 0, 3, 2,1) και (0, 3, 2,1, 4). Θεωρούμε τώρα ότι η ομάδα αριθμών (X0, X1, X2, X3, X4) αντιστοιχεί σε κάποια από αυτές τις πέντε πιθανές διατάξεις. Υποθέτοντας ότι τα δύο κύτταρα της λέξης φέρουν αριθμούς X0 και X1 και με τη γνώση της διαδικασίας ολίσθησης ενός διανύσματος ελέγχου που επιτελείται κατά τον έλεγχο ορθής λειτουργίας, μπορούμε να βρούμε τους αριθμούς που αντιστοιχούν στα κύτταρα – θύτες. Αυτοί ακριβώς οι αριθμοί φαίνονται στο Σχήμα 4.7 με την έντονη απλή γραμμή.



Σχήμα 4.7 Κύτταρα Θύματα – Θύτες για Λέξη Μήκους 2 (4 περιπτώσεις)

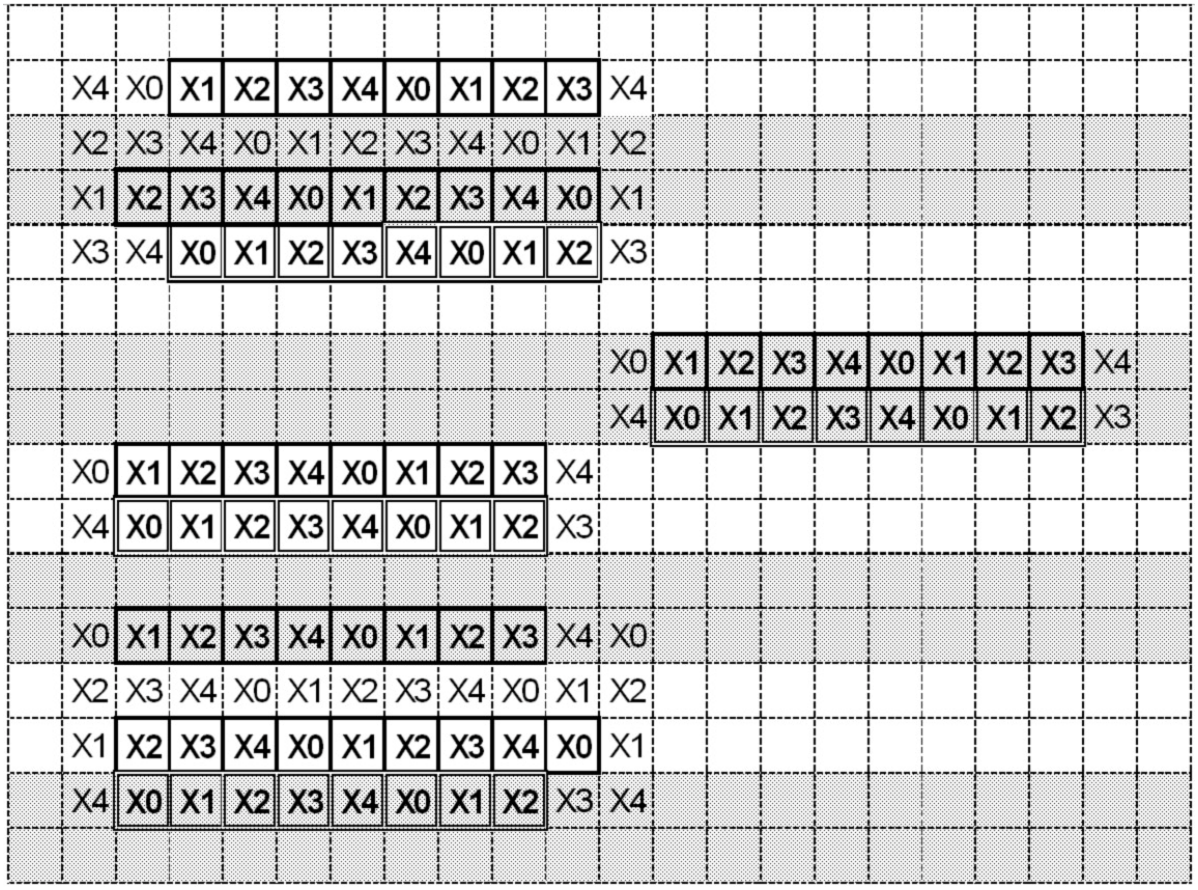
Με βάση την προηγούμενη ανάλυση και το Σχήμα 4.7 μπορούμε τώρα να προσδιορίσουμε ποιο είναι το ποσοστό των εγγραφόμενων λέξεων που θα πρέπει νωρίτερα να διαβαστούν. Θεωρούμε δεδομένο ότι η λέξη θα γραφεί και συνεπώς ότι ένα από τα X0, X1 εκτελεί μετάβαση. Παρατηρούμε πως στην περίπτωση όπου η λέξη ανήκει σε ανώτερη λευκή Word – Line, μόνο ο αριθμός X1 εμφανίζεται στα κύτταρα – θύτες. Άρα στο σύνολο των περιπτώσεων που θα γραφεί η λέξη, μόνο στις μισές (μετάβαση του X1) έχουμε απόκρυψη σφάλματος ενώ στις άλλες μισές (μετάβαση του X0) δεν έχουμε. Ακριβώς το ίδιο βλέπουμε πως ισχύει και στις υπόλοιπες περιπτώσεις, όταν δηλαδή η λέξη ανήκει σε κατώτερη λευκή ή σε ανώτερη / κατώτερη γκρίζα Word – Line. Άρα συνολικά όταν το μήκος λέξης είναι 2 οι μισές από τις λέξεις που θα γραφούν θα πρέπει να διαβαστούν προηγουμένως, γεγονός που οδηγεί σε αύξηση του κόστους ίση με το 50% του κόστους των εγγραφών. Σε κάθε περίπτωση έχουμε απόκρυψη σφάλματος όταν αλλάζει το δεξιότερο bit (X1) της λέξης που εξετάζουμε.



Σχήμα 4.8 Κύτταρα Θύματα – Θύτες για Λέξη Μήκους 4 (4 περιπτώσεις)

Με τους ίδιους συλλογισμούς μπορούμε να υπολογίσουμε την αύξηση του κόστους σε μνήμες με μήκος λέξης μεγαλύτερο από 2. Αν το μήκος λέξης είναι 4, κατασκευάζουμε τον πίνακα που βλέπουμε στο Σχήμα 4.8 όπου φαίνεται η λέξη και τα κύτταρα θύτες σε κάθε μία από τις τέσσερις δυνατές περιπτώσεις (ανάλογα με την Word - Line που βρίσκεται η λέξη). Θεωρούμε πάλι ως δεδομένο ότι η λέξη που μας ενδιαφέρει θα γραφεί και επομένως ότι κάποιο από τα κύτταρά της, (X0, X1, X2, X3) εκτελεί μετάβαση. Παρατηρούμε ότι στις ανώτερες λευκές ή γκρίζες Word – Lines εμφανίζονται όλοι οι αριθμοί X0, X1, X2 και X3 ανάμεσα στα κύτταρα θύτες, πράγμα που σημαίνει ότι σε αυτές τις περιπτώσεις πρέπει πάντα να διαβάσουμε μία λέξη πριν τη γράψουμε. Στις δε κατώτερες λευκές και γκρίζες εμφανίζονται οι τρεις από τους τέσσερις αριθμούς (δηλ όλοι εκτός από τον X0) ανάμεσα στα κύτταρα θύτες. Αυτό σημαίνει ότι το 75% των λέξεων των κατώτερων Word – Lines θα πρέπει να διαβαστεί πριν γραφεί, αφού μόνο όταν αλλάζει το αριστερότερο bit της λέξης δεν έχουμε απόκρυψη σφάλματος. Συνολικά ο αριθμός των επιπλέον αναγνώσεων είναι ίσος με τα 7/8 του αριθμού των εγγραφών.

Για μνήμες με μήκος λέξης από 8 και πάνω εύκολα επιβεβαιώνεται ότι κάθε λέξη που γράφουμε πρέπει νωρίτερα να διαβαστεί. Στο Σχήμα 4.9 παρουσιάζουμε την περίπτωση όπου το μήκος λέξης είναι ίσο με 8.



Σχήμα 4.9 Κύτταρα Θύματα – Θύτες για Λέξη Μήκους 8 (4 περιπτώσεις)

Θα τροποποιήσουμε τώρα τη σχέση που μας έδινε το κόστος του αλγορίθμου συναρτήσει του αριθμού των κυττάρων n και του μήκους της λέξης $w=2^q$ έτσι ώστε να λαμβάνει υπόψιν και το κόστος των επιπλέον αναγνώσεων που λύνουν το πρόβλημα της απόκρυψης σφαλμάτων. Η νέα σχέση δίδεται στην εξίσωση 4.1 που ακολουθεί:

$$n*(5*2^{5-q} + (1+mf)* [2^5/\max[1,2^q/5] + 2^{-q+1}]) \qquad \text{Εξ 4.1}$$

Η παράμετρος *mf* (*masking factor*) εξαρτάται από το μήκος της λέξης και παίρνει τις εξής τιμές: 0 για w=1 (q=0), 0.5 για w=2 (q=1), 7/8 για w=4 (q=2) και 1 για w≥8 (q≥3).

Με εφαρμογή της εξίσωσης Εξ. 4.2 προκύπτει ο επόμενος πίνακας που δίνει το χρονικό κόστος εφαρμογής του αλγορίθμου της Προ. Τύπου -1 γειτονιάς συναρτήσει του αριθμού των κυττάρων της μνήμης (n), για διάφορα μήκη λέξεων με και χωρίς διόρθωση του φαινομένου της απόκρυψης σφαλμάτων.

Πίνακας 4.1 Κόστος Εφαρμογής του Αλγορίθμου της Προ. Τύπου -1 Γειτονιάς

# bit Λέξης	**Κόστος**	
	Χωρίς Διόρθωση Απόκρυψης Σφαλμάτων	**Με Διόρθωση Απόκρυψης Σφαλμάτων**
1-BIT	194*n	194*n
2-BIT	113*n	129*n
4-BIT	72.5*n	100.5*n
8-BIT	40.3*n	60.3*n
16-BIT	20.1*n	30.1*n
32-BIT	10.1*n	15.1*n

Ακολούθως παρουσιάζουμε τον τροποποιημένο αλγόριθμο που διορθώνει την απόκρυψη σφαλμάτων, τον οποίο θα ονομάσουμε όπως νωρίτερα TLAPNPSFA1T – W.O. προσθέτοντας τα αρχικά W.O. που σημαίνουν Word Oriented υποδεικνύοντας έτσι ότι προορίζεται για χρήση σε μνήμες λειτουργίας λέξης. Το διάνυσμα *OLD_P[5]* κρατάει την αμέσως προηγούμενη μορφή του διανύσματος ελέγχου προκειμένου να γίνει ανάγνωση μιας λέξης πριν την εγγραφή της (δηλαδή πριν την εφαρμογή σε αυτή του νέου διανύσματος) όπου αυτό απαιτείται. Την ανάγνωση – σύγκριση την κάνει η συνάρτηση *compare_old_pattern(OLD_P, i)* η οποία λειτουργεί ως εξής:

- Αν η λέξη έχει μήκος 2, τότε διαβάζει τις λέξεις που πρόκειται να γραφούν, δηλαδή αυτές που περιέχουν bit που πρόκειται να αλλάξει, αν αυτό το bit είναι το δεξιότερο της λέξης.
- Αν η λέξη έχει μήκος 4, τότε διαβάζει τις λέξεις που πρόκειται να γραφούν αν αυτές βρίσκονται σε ανώτερη Word – Line ή αν βρίσκονται σε κατώτερη Word – Line και το bit που θα αλλάξει είναι οποιοδήποτε άλλο εκτός από το αριστερότερο της λέξης.
- Αν η λέξη έχει μήκος 8 ή μεγαλύτερο τότε διαβάζει όλες τις λέξεις (αφού όλες θα γραφούν).

```
BIT MATRIX A[5], P[5], OLD_P[5];

// Εφαρμογή και ανάγνωση του πρώτου διανύσματος

OLD_P :=A := P := generate_pattern(0);
for  i := 1 to p do
        {
        apply_to_all_pattern(P,i);

        if ( i mod 2 = 0 ) then shift_right_pattern(P,1);                    //even
        else if (  (i +1) /2 mod 2 ≠ 0 ) then shift_right_pattern(P, 2);     // 1,5,9 …
        else shift_left_pattern(P, 2);                                       // 3,7,11…
        }
P := A;
read_memory(P);

// Εφαρμογή και ανάγνωση των υπολοίπων διανυσματων

for  j = 1 to k* 2^k do
        {
        A := P := generate_pattern(j);

        for  i := 1 to p do
                {
                compare_old_pattern(OLD_P, i);
                apply_pattern(P, i);
                if ( i mod 2 = 0 ) then shift_right_pattern(P,1);
                else if (  (i +1) /2 mod 2 ≠ 0 ) then shift_right_pattern(P, 2);
                else shift_left_pattern(P, 2);
                }
        OLD_P := P := A;
        read_memory(P);
        }
```

Η συνάρτηση **read_memory(P)** παραμένει όπως στον αλγόριθμο 4.1

Αλγόριθμος 4.2: Ο Αλγόριθμος TLAPNPSFA1T– W.O με Διόρθωση Απόκρυψης Σφαλμάτων

ΚΕΦΑΛΑΙΟ 5. Η ΓΕΙΤΟΝΙΑ ΤΥΠΟΥ -Δ ΚΑΙ ΤΟ NWSF ΜΟΝΤΕΛΟ ΣΦΑΛΜΑΤΩΝ

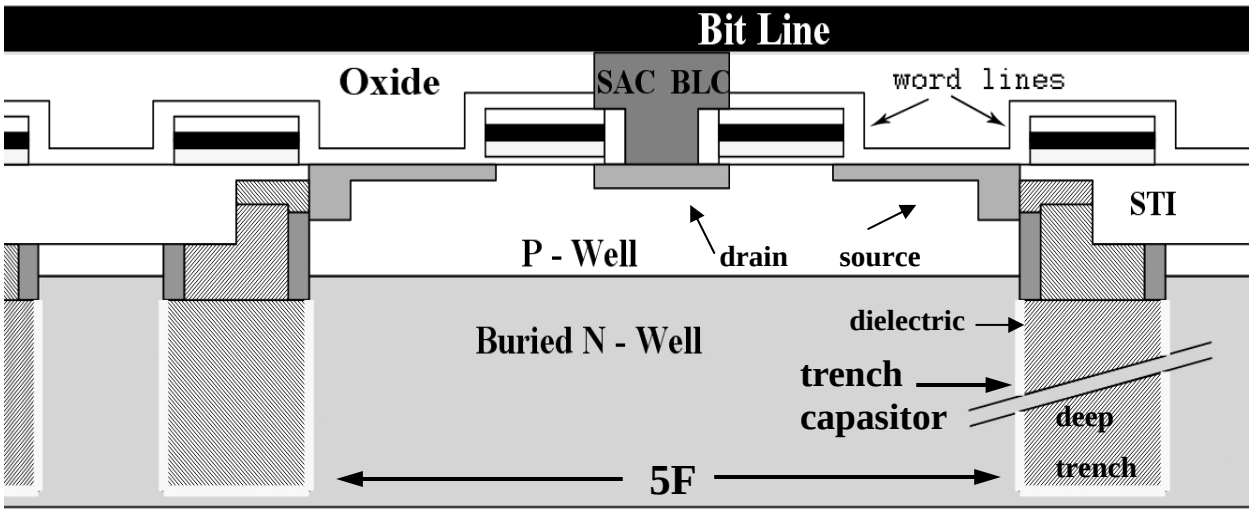
5.1. Η Γειτονιά Τύπου – Δ

Όπως έχουμε προαναφέρει το NPSF μοντελοποιεί την ευαισθησία που έχει ένα κύτταρο μνήμης στα δεδομένα και τις μεταβάσεις των γειτονικών κυττάρων. Αυτό σε φυσικό επίπεδο σημαίνει ότι τα κύτταρα είναι ευαίσθητα στο ηλεκτρικό φορτίο των γειτονικών κυττάρων καθώς και στις μεταβολές αυτού.

Το ηλεκτρικό φορτίο του κυττάρου αποθηκεύεται στον πυκνωτή του, ο οπλισμός του οποίου συνδέεται με την πηγή του τρανζίστορ διέλευσης. Από την άλλη, η υποδοχή του τρανζίστορ είναι συνδεδεμένη στη Bit-Line και συνεπώς φέρει το ίδιο φορτίο με αυτή. Συνεπώς, όταν το κύτταρο είναι σε κατάσταση ηρεμίας αλληλεπιδρά με το περιβάλλον του, είτε ως θύμα είτε ως θύτης, μέσω του πυκνωτή και της πηγής του τρανζίστορ του.

Όπως έχουμε δει, στην Προ. Τύπου – 1 γειτονιά έχει συμπεριληφθεί και το κύτταρο που έχει κοινή υποδοχή με το κύτταρο βάσης. Για να δούμε αν όντως δύο κύτταρα με κοινή υποδοχή αναμένεται να έχουν σημαντική αλληλεπίδραση θα μελετήσουμε το Σχήμα 5.1 όπου βλέπουμε μία κάθετη τομή μνήμης DRAM με πυκνωτή τύπου πηγαδιού. Κατ’ αρχήν βλέπουμε ότι η απόσταση των πυκνωτών των κυττάρων είναι πολύ μεγάλη και ίση με 5F, όπου F είναι η ελάχιστη λιθογραφική απόσταση. Η απόσταση μεταξύ των πηγών των τρανζίστορ τους είναι ελαφρώς μικρότερη αλλά παραμένει μεγάλη, ίση με περίπου 3F.

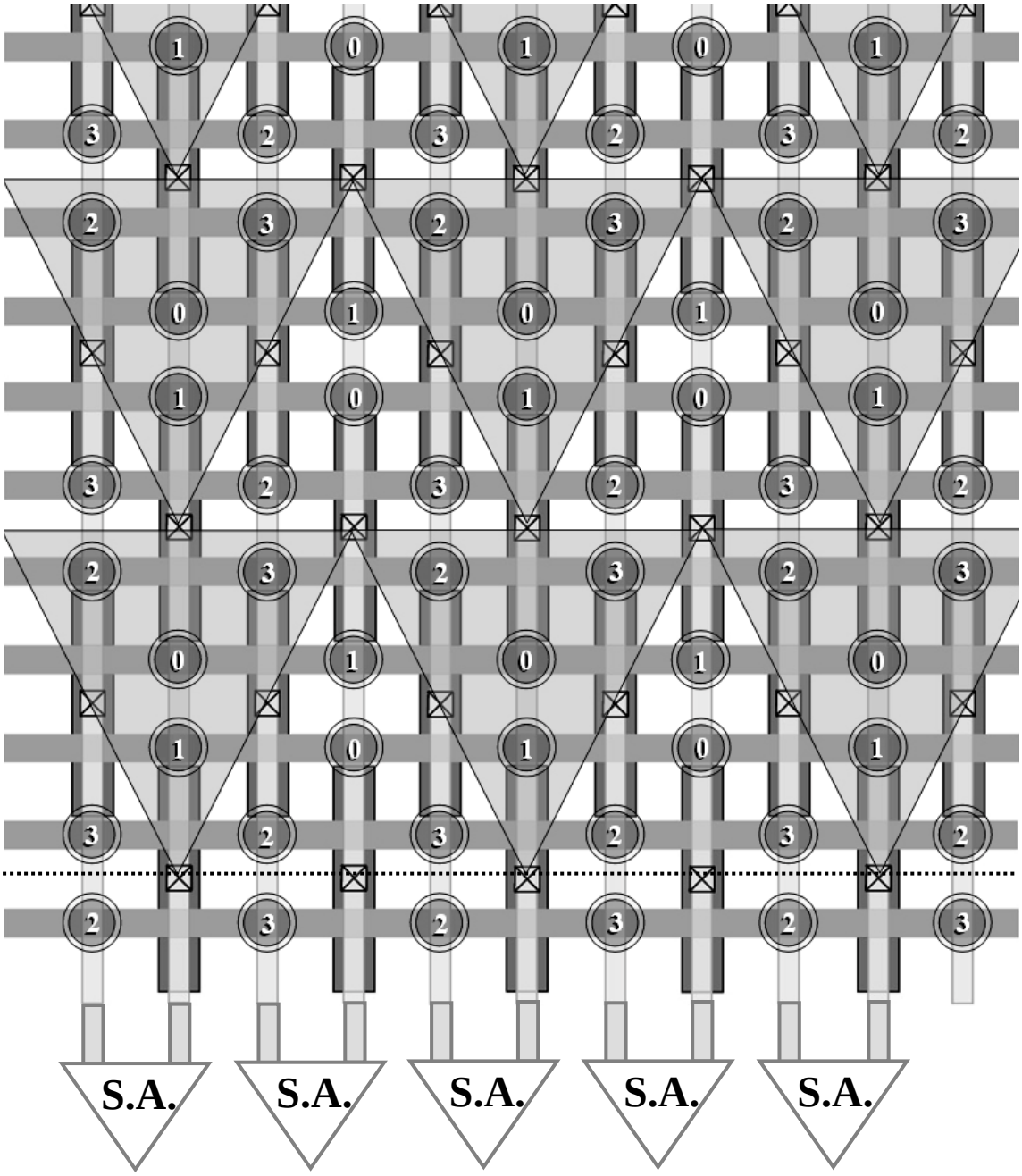
Επιπρόσθετα βλέπουμε ότι μεταξύ των πηγών των τρανζίστορ και στο ίδιο επίπεδο παρεμβάλλεται η κοινή υποδοχή. Οι πηγές των τρανζίστορ απέχουν ουσιαστικά τριπλάσια απόσταση μεταξύ τους από ότι απέχουν από την κοινή υποδοχή. Κατά συνέπεια, κάθε ένα από αυτά τα κύτταρα 'βλέπει' ως γειτονικό φορτίο σε αυτή τη διεύθυνση το φορτίο της κοινής υποδοχής, μέσω της πηγής του, και όχι το φορτίο του άλλου κυττάρου. Επιπρόσθετα, το γεγονός ότι η κοινή υποδοχή βρίσκεται ακριβώς ανάμεσα αποκλείει το ενδεχόμενο να υπάρχει (λόγω ελαττωματικής κατασκευής) μία αγώγιμη γραμμή μεταξύ των δύο πηγών χωρίς αυτή η γραμμή να συνδέεται και με την κοινή υποδοχή. Το φορτίο της κοινής υποδοχής αλλάζει με πολύ μεγάλη συχνότητα αφού αυτή είναι συνδεδεμένη με την Bit – Line, η οποία εκτελεί μετάβαση κάθε φορά που γίνεται ανάγνωση ή εγγραφή σε ένα κύτταρο που είναι συνδεδεμένο σε αυτή.



Σχήμα 5.1 Κάθετη Τομή Μνήμης με Πυκνωτή Τύπου Πηγαδιού

Από τα παραπάνω συμπεραίνουμε ότι αν υπάρχει σε αυτή τη διεύθυνση κάποια ευαισθησία του κυττάρου βάσης ως προς γειτονικά φορτία, αυτή θα είναι ως προς το πολύ κοντινό και με πολύ συχνές μεταβολές φορτίο της κοινής υποδοχής και όχι ως προς το μακρινό φορτίο της πηγής του άλλου κυττάρου [21]. Αυτή η παρατήρηση μας ωθεί να προτείνουμε ένα νέο τύπο γειτονιάς ο οποίος περιλαμβάνει τα κύτταρα της Προ.Τύπου – 1 γειτονιάς εξαιρώντας όμως το κύτταρο με κοινή υποδοχή ως προς το κύτταρο βάσης. Τη νέα αυτή γειτονιά, που φαίνεται στο Σχήμα 5.2, θα την ονομάσουμε *γειτονιά Τύπου Δέλτα (ή Τύπου – Δ γειτονιά)* εξαιτίας του τριγωνικού σχήματος που έχει. Στο ίδιο σχήμα βλέπουμε τη μέθοδο της παράθεσης να

εφαρμόζεται για αυτή τη γειτονιά, γεμίζοντας τη μνήμη με τριγωνικές περιοχές οι οποίες δεν αλληλοεπικαλύπτονται.



Σχήμα 5.2 Η Τύπου – Δ Γειτονιά

Παρατηρούμε ότι έχουμε δύο είδη τριγώνων: αυτά που η κορυφή τους είναι προσανατολισμένη προς τα επάνω, τα οποία στο εξής θα καλούμε *άνω προσανατολισμένα τρίγωνα* (*up – oriented*) ή απλά *άνω τρίγωνα* και αυτά που η κορυφή τους είναι προσανατολισμένη προς τα κάτω και τα οποία θα καλούμε *κάτω προσανατολισμένα τρίγωνα* (*down – oriented*) ή απλά *κάτω τρίγωνα*. Παρατηρούμε επίσης ότι με την αρίθμηση που έχουν τα κύτταρα, σε κάθε Word – Line εμφανίζονται κύτταρα που είτε φέρουν αριθμούς 0 και 1, είτε φέρουν αριθμούς 2 και 3. Αυτό σημαίνει ότι κάθε φορά που εφαρμόζουμε ένα διάνυσμα ελέγχου χρειάζεται να ενεργοποιήσουμε μόνο τις μισές Word – Lines, δηλαδή αυτές που περιέχουν κύτταρα που εκτελούν μετάβαση. Επίσης παρατηρούμε ότι σε κάθε ζεύγος κυττάρων με κοινή υποδοχή, οι αριθμοί που φέρουν τα κύτταρα είναι είτε 0 και 1, είτε 2 και 3. Στο εξής θα λέμε ότι τα κύτταρα που φέρουν αριθμούς 0 είναι *συζυγή* με τα κύτταρα που φέρουν αριθμό 1 και αντιστρόφως. Το ίδιο ισχύει και για τα κύτταρα με αριθμούς 2 και 3.

Ο αριθμός των κυττάρων στη γειτονιά Τύπου – Δ είναι 4 και συνεπώς η ακολουθία διανυσμάτων με βάση το γράφο του Euler που πρέπει να εφαρμοστεί για τον έλεγχο ορθής λειτουργίας αποτελείται από $k*2^k + 1 = 65$ διανύσματα ελέγχου. Μία τέτοια 4 – bit ακολουθία φαίνεται στο Σχήμα 5.3

Προκειμένου να γράψουμε έναν αλγόριθμο που θα πραγματοποιεί τον έλεγχο της μνήμης με βάση τη γειτονιά Τύπου – Δ, θα μεταφέρουμε την αρίθμηση των κυττάρων της που φαίνεται στο Σχήμα 5.2 πάνω σε πίνακα, όπως ακριβώς κάναμε και για την Προ. Τύπου – 1 γειτονιά. Η μεταφορά αυτή σε πίνακα φαίνεται στο Σχήμα 5.4, όπου και πάλι βλέπουμε τις δύο ομάδες Word – Lines (λευκές / γκρίζες). Παρατηρούμε σε αυτό το σχήμα ότι οι αριθμοί κυττάρων 0 και 1 εμφανίζονται μόνο στις γκρίζες Word – Lines ενώ οι αριθμοί 2 και 3 μόνο στις λευκές. Αυτό σημαίνει ότι οι συζυγείς αριθμοί κυττάρων ανήκουν στην ίδια ομάδα Word – Lines. Από τα προαναφερθέντα συμπεραίνουμε επίσης ότι, εκτός από την αρχικοποίηση της μνήμης, όταν εφαρμόζουμε ένα διάνυσμα ελέγχου πάνω στη μνήμη, τα κύτταρα που πρέπει να γράψουμε βρίσκονται όλα είτε πάνω στις γκρίζες Word – Lines, αν ο αριθμός κυττάρου που εκτελεί μετάβαση είναι 0 ή 1, είτε πάνω στις λευκές, αν ο αριθμός κυττάρου που εκτελεί μετάβαση είναι 2 ή 3.

$P_4P_3P_2P_1$	$P_4P_3P_2P_1$	$P_4P_3P_2P_1$	$P_4P_3P_2P_1$
0000	1001	0101	0011
0001	0001	0001	0001
0011	0000	1001	0101
0010	1000	1101	0111
0110	1010	1100	1111
0111	0010	1000	1101
0101	0011	0000	1001
0100	1011	0100	1011
1100	1111	0110	1010
1101	0111	0010	1000
1111	0110	1010	1100
1110	1110	1110	1110
1010	1100	1111	0110
1011	0100	1011	0100
1001	0101	0011	0000
1000	1101	0111	0010
			0000

Σχήμα 5.3 Διάσχιση του Γράφου του Euler για Διάνυσμα Μήκους 4

3	2	3	2	3	2	3	2
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
3	2	3	2	3	2	3	2
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3

Σχήμα 5.4 Μεταφορά της Αρίθμησης της Γειτονιάς Τύπου – Δ σε Πίνακα

Με αυτές τις παρατηρήσεις προτείνουμε τον Αλγόριθμο 5.1 που φαίνεται στην επόμενη σελίδα. Τον αλγόριθμο αυτό θα τον ονομάσουμε *TLAPNPSFΔT* ακολουθώντας τη λογική της ονομασίας των προαναφερθέντων αλγορίθμων. Τα αρχικά της ονομασίας σημαίνουν Test and Locate APNPSF using the Δ - neighborhood and the Tiling method. Στη συνέχεια θα εξηγήσουμε τη λειτουργία του.

Υποθέτουμε ότι τα δύο υψηλότερα σε αξία bit του διανύσματος ελέγχου αντιστοιχούν σε κύτταρα με αριθμούς 2 και 3 ενώ τα δύο χαμηλότερα σε αξία bit σε κύτταρα με αριθμούς 0 και 1. Όπως έχουμε προαναφέρει, εκτός από την αρχικοποίηση της μνήμης στις υπόλοιπες εγγραφές μόνο μία ομάδα Word – Lines συμμετέχει. Προκειμένου να βρούμε σε ποια ομάδα Word – Lines θα γίνουν οι εγγραφές, συγκρίνουμε το παλιό διάνυσμα ελέγχου με το νέο. Για το λόγο αυτό εκτός από τον καταχωρητή A που κρατάει το νέο διάνυσμα ελέγχου έχουμε και τον καταχωρητή *OLD_A* που κρατάει το παλιό διάνυσμα ελέγχου. Η συνάρτηση *diff (A, OLD_A)* συγκρίνει τα δύο διανύσματα και αν αυτά διαφέρουν στα δύο χαμηλότερα bit τους επιστρέφει την τιμή *LOW* ενώ αν διαφέρουν στα δύο υψηλότερα την τιμή *HIGH***.** Η τιμή LOW δείχνει ότι στην εγγραφή θα συμμετέχουν οι γκρίζες Word – Lines ενώ η τιμή HIGH δείχνει ότι θα συμμετέχουν οι λευκές.

Προκειμένου να επισκεφτούμε μόνο τις Word – Lines που θα συμμετέχουν στην εγγραφή δουλεύουμε ως εξής: α) αρχικοποιούμε κατάλληλα τη μεταβλητή init η οποία καθορίζει ποια θα είναι η πρώτη Word – Line που θα επισκεφθούμε (1 για λευκές Word – Lines, 2 για γκρίζες σύμφωνα με το Σχήμα 5.4), και β) όταν η επόμενη Word – Line ανήκει σε διαφορετική ομάδα από την τρέχουσα (που σημαίνει ότι αντιστοιχεί σε μονό αριθμό), προσπερνάμε δύο Word – Lines για να πάμε στην επόμενη της ίδιας ομάδας.

Αντίστοιχα, η συνάρτηση *low_bit (A)* επιστρέφει τα δύο χαμηλότερα bit του A (προκειμένου να γραφούν στον καταχωρητή P) ενώ η *high_bit (A)* τα δύο υψηλότερα. Αυτό γίνεται γιατί κατά την εφαρμογή ενός διανύσματος ελέγχου δεν χρειαζόμαστε και τα τέσσερα bit του καταχωρητή αλλά μόνο τα δύο που αντιστοιχούν στο κύτταρο που εκτελεί μετάβαση και στο συζυγές του, αφού οι Word – Lines που θα συμμετέχουν στην εγγραφή μόνο αυτούς τους αριθμούς κυττάρων θα περιέχουν.

```
    BIT MATRIX A[4], OLD_A[4], P[2];
// Εφαρμογή και ανάγνωση του πρώτου διανύσματος
A := OLD_A := generate_pattern(0);
P := high_bit(A);

for  i := 1 to p do
        {
        apply_to_all_pattern(P,i);
        if ( i mod 2 = 0 ) then shift_right_pattern(P,1);          //even
        else if (  (i +1) /2 mod 2 ≠ 0 ) then P := low_bit(A);     // 1,5,9 …
        else    P := high_bit(A);                                  // 3,7,11…
        }
read_memory(A);

// Εφαρμογή και ανάγνωση των υπολοίπων διανυσματων
for  j := 1 to k* 2^k do
        {
        A := generate_pattern(j);

        if ( diff(A, OLD_A) = LOW ) then
                {
                P := low_bit(A);
                init = 2;
                }
        else
                {
                P := high_bit(A);
                init := 1;
                 }
        OLD_A := A;

        for  i := init to p do
                {
                apply_pattern(P, i);
                if ( i mod 2 ≠ 0 ) then i = i + 2;
                shift_right_pattern(P,1);
                }
        read_memory(A);
        }

function read_memory(A)
        {
        BIT MATRIX P[2];
        P := high_bit(A);
        for  i := 1 to p do
                {
                compare_pattern(P,i);
                if ( i mod 2 = 0 ) then shift_right_pattern(P,1);          //even
                else if (  (i +1) /2 mod 2 ≠ 0 ) then P := low_bit(A);     // 1,5,9 …
                else    P := high_bit(A);                                  // 3,7,11…
                }
        }
```

Αλγόριθμος 5.1: Ο Αλγόριθμος TLAPNPSFΔT

Οι υπόλοιπες συναρτήσεις του αλγορίθμου είναι όπως τις έχουμε δει στον προηγούμενο αλγόριθμο της προσαρμοσμένης Τύπου – 1 γειτονιάς, με τη διαφορά ότι πλέον οι συναρτήσεις apply_to_all_pattern(P,i), apply_pattern(P,i) και compare_pattern(P,i) εφαρμόζουν ή συγκρίνουν διανύσματα των δύο bit και όχι των πέντε, αφού σε κάθε Word – Line εμφανίζονται δύο μόνο διαφορετικοί αριθμοί κυττάρων.

Το κόστος εφαρμογής του αλγορίθμου σε μία μνήμη λειτουργίας bit υπολογίζεται όπως στην περίπτωση της (προσαρμοσμένης ή όχι) Τύπου – 1 γειτονιάς. Σε μία μνήμη όπου η συστοιχία έχει n κύτταρα, ο συνολικός αριθμός των πράξεων ανάγνωσης είναι $n*(k*2^k + 1)$ και των πράξεων εγγραφής είναι $n*(2^k + 1)$. Το συνολικό κόστος του αλγορίθμου είναι $n*(k*2^k +2^k + 2)$ και για k=4 δίνει 82*n. Αυτό συνιστά μία δραματική μείωση σε σχέση με την Τύπου – 1 γειτονιά, της τάξης του 57%.

Στις μνήμες λειτουργίας λέξης με μήκος λέξης $w = 2^q$ bits, χρειάζονται $n*(k*2^{k-q} +2^{-q}) = n*(2^{6-q} +2^{-q})$ πράξεις ανάγνωσης. Για να υπολογίσουμε το κόστος εγγραφής παρατηρούμε ότι: α) με μήκος λέξης 2 (q=1) γίνεται μετάβαση σε ένα κύτταρο με κάθε πράξη εγγραφής, β) με q=2 σε δύο, γ) με q=3 σε τέσσερα κ.ο.κ. Άρα το κόστος εγγραφής θα είναι συνολικά $n*(2^{k-q+1} + 1/w) = n*(2^{5-q} + 2^{-q})$, $q\geq 1$. Συνεπώς, το συνολικό κόστος του αλγορίθμου θα δίνεται από την ακόλουθη εξίσωση:

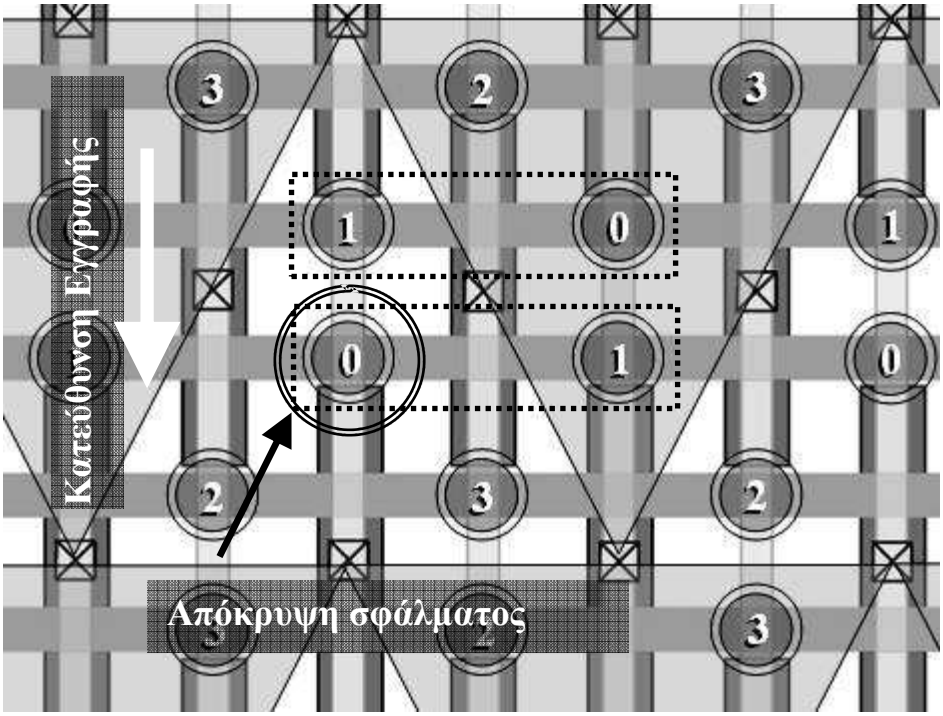
$$n*(2^{6-q} + 2^{5-q} + 2^{-q+1}) \qquad q\geq 1 \qquad \text{Εξ 5.1}$$

Για μήκος λέξης w = 2 χρειαζόμαστε 49*n πράξεις, για w = 4 χρειαζόμαστε 24.5*n, ενώ για w = 8 χρειαζόμαστε 12.3*n πράξεις.

5.2. Απόκρυψη Σφαλμάτων στην Γειτονιά Τύπου Δ

Στις μνήμες λειτουργίας λέξης παρουσιάζεται το φαινόμενο της απόκρυψης σφαλμάτων κατά την εφαρμογή του αλγορίθμου ελέγχου με βάση την Τύπου – Δ γειτονιά. Όπως είναι εύκολο να διαπιστώσουμε, κατά τη διάρκεια εφαρμογής ενός διανύσματος στις Word – Lines όπου πρόκειται να γίνει εγγραφή, κάθε λέξη με μήκος μεγαλύτερο ή ίσο του δύο περιέχει ίσο αριθμό κυττάρων που εκτελούν μετάβαση και κυττάρων που δεν εκτελούν μετάβαση, τοποθετημένα εναλλάξ. Τα δεύτερα κατά τη διάρκεια της εγγραφής θα επανεγγραφούν με τα δεδομένα που ήδη έχουν ή τουλάχιστον τα δεδομένα που αναμένεται να έχουν, εκτός και αν

κάποιο σφάλμα που ενεργοποιήθηκε νωρίτερα έχει προκαλέσει απώλεια των δεδομένων τους. Η επανεγγραφή μπορεί να αποκρύψει την ενδεχόμενη απώλεια δεδομένων που αυτά έχουν υποστεί από προηγούμενη μετάβαση κάποιου κυττάρου στη γειτονιά τους.



Σχήμα 5.5 Απόκρυψη Σφάλματος σε Γειτονιά Τύπου – Δ

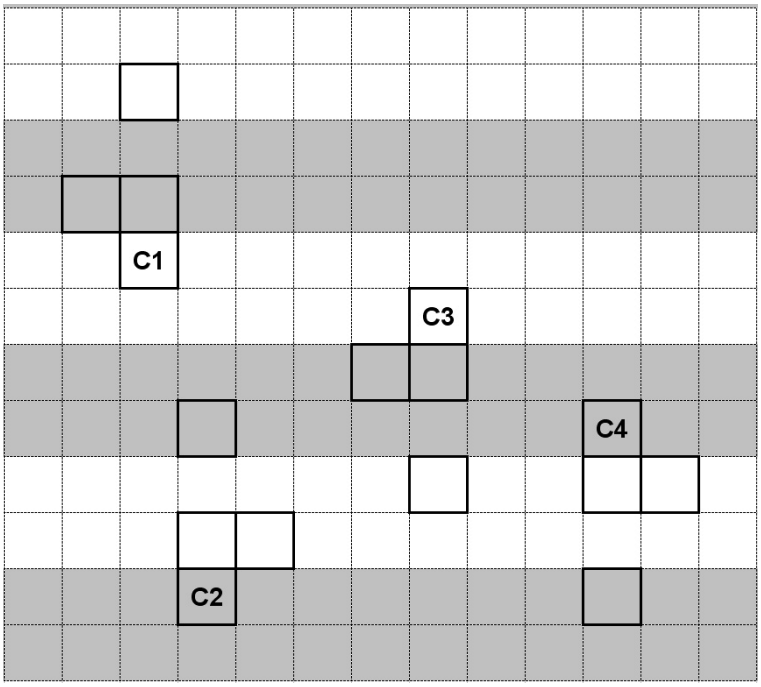
Θα χρησιμοποιήσουμε το Σχήμα 5.5 για να περιγράψουμε ένα παράδειγμα απόκρυψης σφάλματος. Με την υπόθεση ότι το κύτταρο βάσης είναι το κύτταρο που φέρει τον αριθμό 0, αυτό θα επανεγγραφεί με τα σωστά δεδομένα μόνο εφόσον τα κύτταρα με αριθμό 1 εκτελούν μετάβαση. Αν ένα από τα κύτταρα 2 ή 3 εκτελούν μετάβαση τότε στις Word – Lines που περιέχουν κύτταρα 0 και 1 δεν θα γίνει καμία εγγραφή. Άρα το μόνο σφάλμα που μπορεί να υποστεί απόκρυψη είναι μία απώλεια δεδομένων στο κύτταρο βάσης που προκαλείται από μετάβαση του κυττάρου 1, δηλαδή του κυττάρου που βρίσκεται στην κορυφή της τριγωνικής γειτονιάς του κυττάρου βάσης. Γενικά, όταν τα κύτταρα που φέρουν κάποιον αριθμό εκτελούν μετάβαση κατά την εφαρμογή ενός διανύσματος, μόνο στα συζυγή κύτταρα αυτών μπορεί να παρουσιαστεί απόκρυψη σφάλματος και αυτό συμβαίνει μόνο στις γειτονιές όπου η μετάβαση του κυττάρου στην κορυφή του τριγώνου προηγείται της επανεγγραφής του

συζυγούς της γειτονιάς του. Στο παράδειγμά μας, απόκρυψη σφάλματος μπορούμε να έχουμε μόνο όταν η μετάβαση του κυττάρου 1 στην κορυφή του τριγώνου προηγείται της επανεγγραφής του κυττάρου βάσης. Προφανώς δεν παίζει ρόλο αν το κύτταρο 0 του σχήματος ανήκει στην ίδια λέξη με το κύτταρο 1 που βρίσκεται στα δεξιά του (όπως φαίνεται στο Σχήμα 5.5) ή με αυτό στα αριστερά του γιατί και στις δύο περιπτώσεις θα επανεγγραφεί.

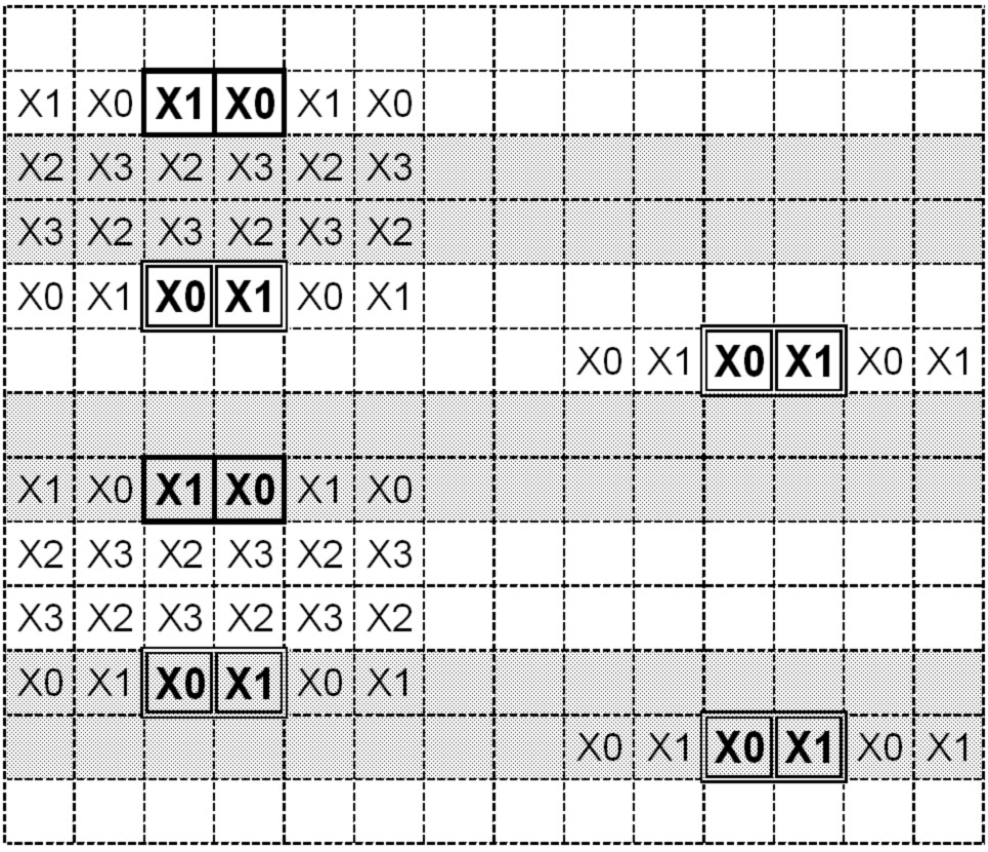
Προκειμένου να γίνουν πιο σαφή τα προηγούμενα, θα ακολουθήσουμε τη μεθοδολογία που χρησιμοποιήσαμε στην Προ. Τύπου – 1 γειτονιά. Σε πρώτη φάση θα δούμε τη μορφή που παίρνει η γειτονιά Τύπου – Δ πάνω σε πίνακα, όπως φαίνεται στο Σχήμα 5.6, ανάλογα με τη θέση του κυττάρου βάσης. Έστω ότι γράφουμε τις Word – Lines από πάνω προς τα κάτω. Για τα κύτταρα βάσης που ανήκουν στις ανώτερες (λευκές ή γκρίζες) Word – Lines παρατηρούμε ότι έχουμε τρία υποψήφια κύτταρα – θύτες τα οποία μπορούν να εκτελέσουν μετάβαση πριν γραφεί το κύτταρο βάσης. Όμως, δεδομένου ότι μόνο η ομάδα Word – Lines στην οποία ανήκει και το κύτταρο βάσης θα γραφεί (δηλαδή θα γραφούν μόνο οι Word – Lines με το ίδιο χρώμα φόντου με αυτή του κυττάρου βάσης), το μόνο πραγματικό κύτταρο – θύτης είναι αυτό που ανήκει και στην ίδια ομάδα Word – Lines με το κύτταρο βάσης. Αντίστοιχα, για τα κύτταρα βάσης που ανήκουν στις κατώτερες Word – Lines παρατηρούμε ότι δεν υπάρχει κανένα κύτταρο θύτης που να εκτελεί μετάβαση πριν το κύτταρο βάσης.

Με βάση τα παραπάνω σχεδιάζουμε το Σχήμα 5.7 που δείχνει τα κύτταρα θύματα – θύτες για λέξεις μήκους 2. Όπως κάναμε και στην Προ. Τύπου – 1 γειτονιά, θεωρούμε δεδομένο ότι η λέξη που είναι σημειωμένη με τη διπλή γραμμή θα γραφεί και επομένως ένα από τα X0, X1 θα εκτελέσει μετάβαση. Αν η λέξη ανήκει σε ανώτερη Word – Line βλέπουμε ότι τα κύτταρα θύτες φέρουν πάντα τους ίδιους αριθμούς που παρουσιάζονται και στα κύτταρα - θύματα, πράγμα που σημαίνει ότι θα έχουμε πάντα απόκρυψη σφάλματος.

Προφανώς, ακριβώς οι ίδιες καταστάσεις ισχύουν και για μήκος λέξης μεγαλύτερο του 2. Επομένως, μία ανώτερη Word – Line πρέπει πάντα να τη διαβάσουμε πριν τη γράψουμε προκειμένου να μην έχουμε απόκρυψη σφάλματος. Αντίθετα, οι λέξεις στις κατώτερες Word – Line δεν έχουν κανένα κύτταρο – θύτη και επομένως δεν μπορούν να παρουσιάσουν απόκρυψη σφάλματος.

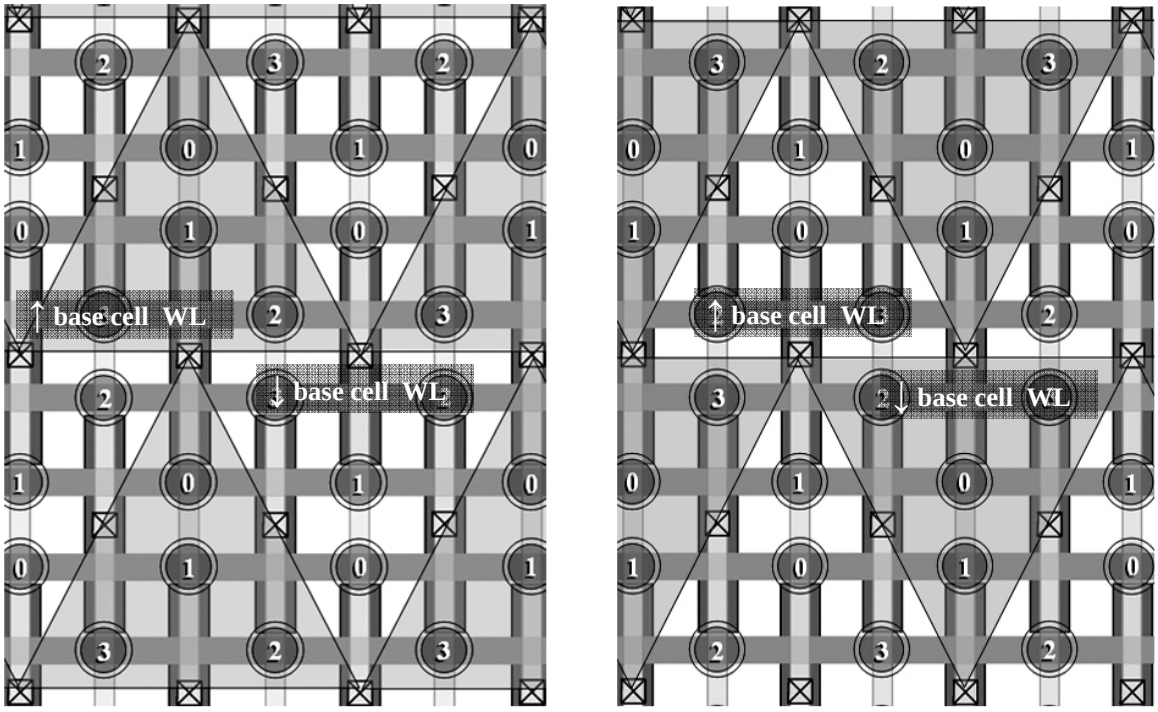


Σχήμα 5.6 Μεταφορά της Τοπολογίας της Γειτονιάς Τύπου – Δ σε Πίνακα



Σχήμα 5.7 Κύτταρα Θύματα – Θύτες για Λέξεις Μήκους 2

Για να συσχετίσουμε ακόμη περισσότερο την προηγούμενη ανάλυση με τη φυσική διάταξη της γειτονιάς, κάνουμε την εξής παρατήρηση: ένα κύτταρο βάσης που ανήκει σε άνω τρίγωνο ενεργοποιείται από ανώτερη Word – Line ενώ ένα κύτταρο βάσης που ανήκει σε κάτω τρίγωνο ενεργοποιείται από κατώτερη Word – Line. Στο Σχήμα 5.8 βλέπουμε ένα παράδειγμα για κύτταρα βάσης με αριθμούς 0 και 1 ενώ το ίδιο ακριβώς ισχύει και για κύτταρα με αριθμούς 2 και 3. Συνεπώς, απόκρυψη σφαλμάτων μπορούμε να έχουμε μόνο στα άνω τρίγωνα.



Σχήμα 5.8 Τα Κύτταρα Βάσης των Άνω Τριγώνων Ενεργοποιούνται Πάντα από Ανώτερες Word – Lines και Αυτά των Κάτω από Κατώτερες Word – Lines

Σύμφωνα με τα προηγούμενα, σε κάθε περίπτωση το κόστος των επιπλέον αναγνώσεων είναι ίσο με το μισό του κόστους των εγγραφών. Επομένως, το συνολικό κόστος ενός αλγορίθμου που εκτελεί έλεγχο για NPSF σφάλματα με χρήση γειτονιάς Τύπου – Δ είναι το ακόλουθο:

$$n*(2^{6-q} + 1.5 * 2^{5-q} + 2^{-q+1}) \qquad q>=1 \qquad \text{Εξ 5.2}$$

Για μήκος λέξης w = 2 χρειαζόμαστε $57*n$ πράξεις, για w = 4 χρειαζόμαστε $28.5*n$ ενώ για w = 8 χρειαζόμαστε $14.3*n$πράξεις.

Ακολούθως παρουσιάζεται ο προτεινόμενος Αλγόριθμος 5.2 που προσφέρει λύση στο πρόβλημα της απόκρυψης σφάλματος στην Τύπου – Δ γειτονιά. Να σημειώσουμε ότι στις ενδιάμεσες επιπλέον αναγνώσεις τα κύτταρα που πρόκειται να εκτελέσουν μετάβαση έχουν προφανώς ακόμα την παλιά τιμή τους και με αυτή πρέπει να γίνει η σύγκριση. Βέβαια στην πράξη η τιμή των συζυγών τους κυττάρων μας ενδιαφέρει για να λύσουμε το πρόβλημα της απόκρυψης σφαλμάτων αλλά αφού βρίσκονται στις ίδιες λέξεις αναγκαστικά θα διαβαστούν και θα συγκριθούν και αυτά μαζί. Χρησιμοποιούμε λοιπόν έναν επιπλέον καταχωρητή, τον OLD_P ο οποίος κρατάει τα κατάλληλα ψηφία του προηγούμενου διανύσματος ελέγχου προκειμένου να είναι δυνατή αυτή η σύγκριση. Το όνομα του νέου αλγορίθμου *TLAPNPSFΔT – W.O.* προκύπτει από το όνομα του αλγορίθμου 5.1 προσθέτοντας τα αρχικά W.O. που σημαίνουν Word – Oriented, υποδεικνύοντας έτσι ότι προορίζεται για χρήση σε μνήμες λειτουργίας λέξης

Ο Πίνακας 5.1 παρουσιάζει μία συνολική σύγκριση του κόστους ελέγχου με χρήση του Προ. Τύπου – 1 και του Τύπου – Δ , με και χωρίς αποφυγή απόκρυψης σφαλμάτων (Α.Σ). Βλέπουμε ότι η επίλυση του προβλήματος απόκρυψης σφαλμάτων προσθέτει ελάχιστο κόστος στην Τύπου – Δ γειτονιά ενώ στην Προ. Τύπου – 1 προσθέτει ένα κόστος της τάξης του 50% για μήκος λέξης πάνω από 4. Επίσης βλέπουμε ότι για μήκη λέξης από 8 και πάνω το κόστος εφαρμογής της Τύπου – Δ γειτονιάς χωρίς απόκρυψη σφαλμάτων είναι λιγότερο από το 24% της αντίστοιχης εφαρμογής της Τύπου – 1.

Πίνακας 5.1 Σύγκριση Κόστους Αλγορίθμων Προ. Τύπου – 1 Γειτονιάς και Γειτονιάς Τύπου – Δ

# bit λέξης	**Κόστος Εφαρμογής Αλγορίθμου**			
	Προ. Τύπου -1		**Τύπου - Δ**	
	Χωρίς διόρθωση Α.Σ.	Με διόρθωση Α.Σ.	Χωρίς διόρθωση Α.Σ.	Με διόρθωση Α.Σ.
1-BIT	194*n	194*n	82*n	82*n
2-BIT	113*n	129*n	49*n	57*n
4-BIT	72.5*n	100.5*n	24.5*n	28.5*n
8-BIT	40.3*n	60.3*n	12.3*n	14.3*n
16-BIT	20.1*n	30.1*n	6.1*n	7.1*n
32-BIT	10.1*n	15.1*n	3.1*n	3.6*n

```
BIT MATRIX A[4], OLD_A[4], P[2],OLD_P[2];

// Εφαρμογή και ανάγνωση του πρώτου διανύσματος

A := OLD_A := generate_pattern(0);
P := high_bit(A);

for  i := 1 to p do
        {
        apply_to_all_pattern(P,i);
        if ( i mod 2 = 0 ) then shift_right_pattern(P,1);          //even
        else if (  (i +1) /2 mod 2 ≠ 0 ) then P := low_bit(A);     // 1,5,9 …
        else    P := high_bit(A);                                  // 3,7,11…
        }
read_memory(A);

// Εφαρμογή και ανάγνωση των υπολοίπων διανυσματων

for  j := 1 to k* 2^k do
        {
        A := generate_pattern(j);

        if ( diff(A, OLD_A) = LOW ) then
                {
                P := low_bit(A);
                OLD_P := low_bit(OLD_A);
                init := 2;
                }
        else    {
                P := high_bit(A);
                OLD_P := high_bit(OLD_A);
                shift_right_pattern(OLD_P,1);
                init := 1;
                }

        OLD_A = A;

        for  i := init to p do
                {
                if ( i mod 2 = 0 ) then compare_pattern(OLD_P,i);
                apply_pattern(P, i);
                if ( i mod 2 ≠ 0 ) then i: = i + 2;
                shift_right_pattern(P,1);
                }
        read_memory(A);
        }
```

Η συνάρτηση **read_memory(P)** είναι όπως προηγουμένως.

Αλγόριθμος 5.2: Ο Αλγόριθμος TLAPNPSFΔT – W.O. με Διόρθωση Απόκρυψης Σφαλμάτων

5.3. Σφάλματα Οφειλόμενα στις Γειτονικές Word – Lines

Θα μελετήσουμε τώρα μία άλλη ενδιαφέρουσα αλληλεπίδραση που μπορεί να προκαλέσει σφάλματα σε μία μνήμη. Η αλληλεπίδραση αυτή είναι η χωρητική σύζευξη μεταξύ γειτονικών Word – Lines. Η χωρητική σύζευξη μεταξύ γειτονικών Word – Lines είναι σημαντική γιατί αυτές βρίσκονται πολύ κοντά μεταξύ τους, έχουν σχετικά μεγάλο μήκος και είναι παράλληλες. Εξαιτίας αυτής της παρασιτικής χωρητικής σύζευξης, όταν μία Word – Line ενεργοποιείται οι γειτονικές της εκτελούν και αυτές μία προσωρινή μετάβαση σε κάποια τιμή υψηλότερη των 0 Volt. Η προσωρινή μετάβαση των γειτονικών Word – Lines ενδέχεται να προκαλέσει μια αντίστοιχα προσωρινή αύξηση του ρεύματος διαρροής των συσχετιζόμενων με αυτή τρανζίστορ η οποία μπορεί να οδηγήσει σε απώλεια δεδομένων σε κάποια από τα σχετικά κύτταρα. Το μοντέλο που περιγράφει αυτό το μηχανισμό σφαλμάτων θα το ονομάσουμε Μοντέλο Σφαλμάτων Αλληλεπίδρασης Γειτονικών Word – Lines (Neighborhood Word-Line Sensitive Fault - NWSF).

Η σχέση που εκφράζει το ρεύμα διαρροής είναι η εξής:

$$I_{ds} = I_{ds\,0} e^{\frac{V_{gs} - V_t}{n v_T}} \left(1 - e^{-\frac{V_{ds}}{v_T}}\right) \qquad \text{Εξ 5.3}$$

όπου V_t είναι η τάση κατωφλίου, V_{gs} η τάση της πύλης του τρανζίστορ και V_{ds} η διαφορά δυναμικού πηγής – υποδοχής. Οι ποσότητες n, v_T και I_{ds0} εξαρτώνται από τα κατασκευαστικά χαρακτηριστικά του τρανζίστορ και τη θερμοκρασία. Στην παραπάνω σχέση παρατηρούμε ότι το ρεύμα διαρροής αυξάνει εκθετικά με την τάση της πύλης του τρανζίστορ. Επίσης παρατηρούμε ότι αυξάνει με τη διαφορά δυναμικού μεταξύ πηγής και υποδοχής, που στην περίπτωσή μας είναι η διαφορά δυναμικού μεταξύ του οπλισμού του πυκνωτή και της Bit – Line.

Σε μία πρώτη προσέγγιση, ένας αλγόριθμός που έχει στόχο να εντοπίσει τέτοια σφάλματα θα πρέπει να περιλαμβάνει όλους τους δυνατούς συνδυασμούς τιμών της Bit – Line και του κυττάρου βάσης όταν ενεργοποιείται το φαινόμενο, δηλαδή όταν ενεργοποιείται μία γειτονική Word – Line. Στη συνέχεια θα μελετήσουμε την ικανότητα των αλγορίθμων της Τύπου – Δ γειτονιάς να εντοπίσουν NWSF σφάλματα. Θα υποθέσουμε ότι σε κάθε κύτταρο

μόνο ένας εκ των δύο μηχανισμών NPSF και NWSF μπορεί να προκαλέσει απώλεια δεδομένων κάθε φορά (*Υπόθεση Απλού Σφάλματος - Single Fault Assumption*).

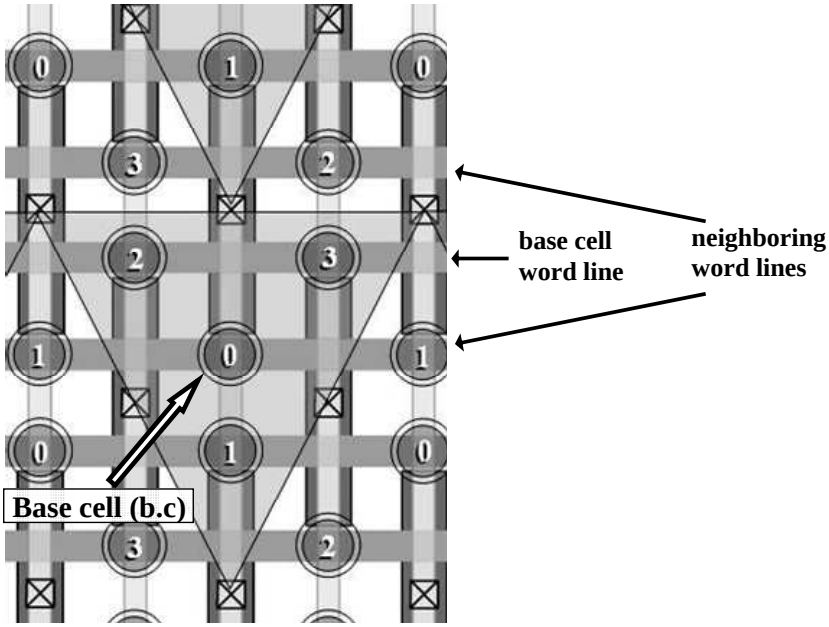
Στο Σχήμα 5.9 παρατηρούμε ότι για το κύτταρο 0 (κύτταρο βάσης) στην γειτονιά Τύπου – Δ του κάτω τριγώνου, οι δύο γειτονικές Word – Lines στη δική του Word – Line είναι οι εξής: από επάνω αυτή που ενεργοποιεί το κύτταρο 1 του προηγούμενου τριγώνου και από κάτω αυτή που ενεργοποιεί τα κύτταρα 2 και 3 της δικής του γειτονιάς. Προφανώς αν πάμε σε άνω τρίγωνο οι δύο αυτές γειτονικές Word – Lines αλλάζουν αμοιβαία θέσεις.

Στο σημείο αυτό θα εισάγουμε την εξής ορολογία: σε ένα τρίγωνο θα ονομάζουμε το κύτταρο που βρίσκεται στην κορυφή του τριγώνου *κορυφαίο κύτταρο (top cell)* και το κύτταρο που βρίσκεται στην κορυφή του αμέσως προηγούμενου τριγώνου επάνω στην ίδια Bit - Line θα το ονομάσουμε *συνορεύον κύτταρο (adjoining cell)*. Προφανώς τα δύο προαναφερθέντα κύτταρα φέρουν πάντα τον ίδιο αριθμό. Να σημειώσουμε εδώ ότι όσον αφορά τα τρίγωνα οι όροι ‘προηγούμενο’ και ‘επόμενο’ τρίγωνο δεν σχετίζονται με τον τρόπο που γίνεται η εγγραφή αλλά με τη διεύθυνση των τριγώνων. Ως ‘επόμενο’ τρίγωνο θεωρούμε αυτό που συνορεύει με την κορυφή του τρέχοντος τριγώνου.

Για τα κύτταρα 2 και 3, η Bit – Line του ενός εκ των δύο, έστω του κυττάρου 2, θα είναι συνδεδεμένη στον ίδιο ενισχυτή σήματος με το κύτταρο βάσης. Συνεπώς όταν αυτή η Word – Line ενεργοποιείται προκειμένου να γίνει μία ενέργεια ανάγνωσης ή εγγραφής πάνω στο κύτταρο 2, η Bit – Line του κυττάρου βάσης θα παίρνει πάντα την συμπληρωματική τιμή του κυττάρου 2 (αν πρόκειται για ανάγνωση) ή το συμπλήρωμα της τιμής που πρόκειται να γράψουμε στο κύτταρο 2 (αν πρόκειται για εγγραφή). Να σημειώσουμε ότι το ίδιο ακριβώς συμβαίνει και στην Προ. Τύπου – 1 γειτονιά.

Παρόμοια, όταν ενεργοποιείται η Word – Line του συνορεύοντος κυττάρου**,** η (κοινή) Bit – Line θα παίρνει την τιμή του συνορεύοντος κυττάρου αν πρόκειται για ανάγνωση, ή την τιμή που πρόκειται να γράψουμε στο συνορεύον τρίγωνο αν πρόκειται για εγγραφή. Επειδή τα δύο κύτταρα που καθορίζουν την τιμή της Bit – Line του κυττάρου βάσης έχουν διαφορετικούς αριθμούς από το κύτταρο βάσης, ο αλγόριθμος της Τύπου – Δ γειτονιάς εγγυάται ότι για κάθε μία από τις γειτονικές Word – Lines που μπορεί να επηρεάσουν εκείνη του κυττάρου βάσης, η Bit – Line και η αποθηκευμένη τιμή του κυττάρου βάσης θα παίρνουν όλους τους δυνατούς

συνδυασμούς τιμών. Συνεπώς, για μνήμες λειτουργίας bit ο αλγόριθμος TLAPNPSFΔT που έχουμε ήδη παρουσιάσει (αλγόριθμος 5.1) είναι κατάλληλος για τον εντοπισμό τόσο των NPSF όσο και των NWSF σφαλμάτων.

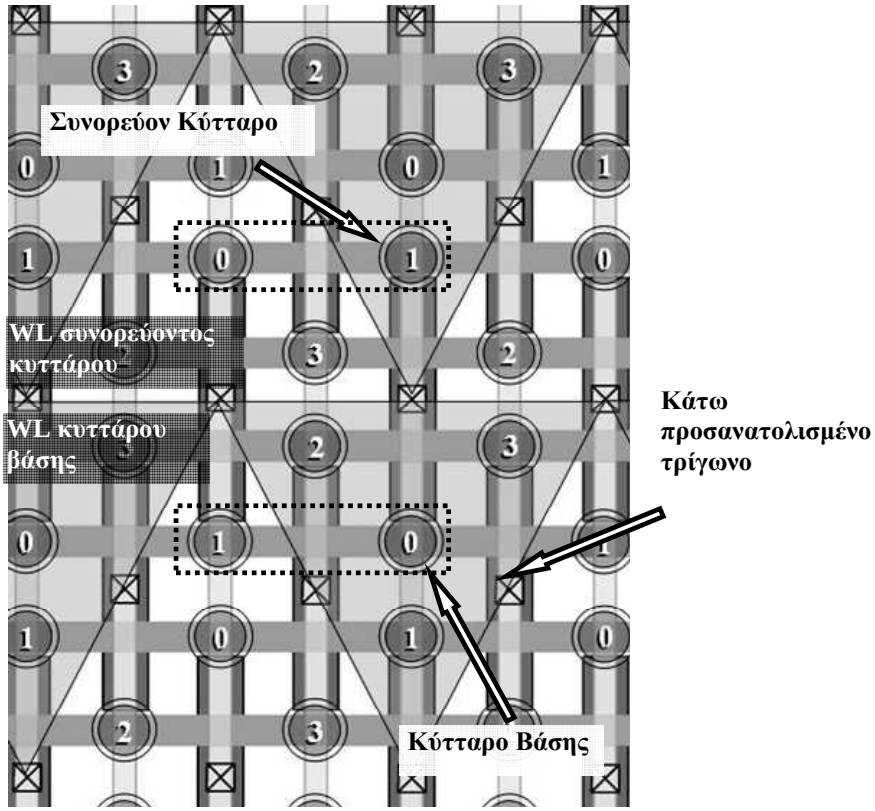


Σχήμα 5.9 Οι Γειτονικές Word – Lines ως προς αυτή του Κυττάρου Βάσης

Στις μνήμες λειτουργίας λέξης πρέπει να αποφύγουμε την απόκρυψη NWSF σφαλμάτων. Υπενθυμίζουμε ότι ο αλγόριθμος TLAPNPSFΔT – W.O. (αλγόριθμος 5.2), διάβαζε τις ανώτερες Word – Lines πριν τις γράψει για να αποφύγει την απόκρυψη NPSF σφαλμάτων. Όμως μπορούμε εύκολα να διαπιστώσουμε ότι με αυτόν τον αλγόριθμο μπορεί να έχουμε απόκρυψη NWSF σφαλμάτων στα κάτω προσανατολισμένα τρίγωνα. Πράγματι, όπως βλέπουμε στο Σχήμα 5.10 αν η μετάβαση του συνορεύοντος κυττάρου προκαλέσει απώλεια δεδομένων στο κύτταρο βάσης, η επανεγγραφή του κυττάρου βάσης με τα σωστά δεδομένα που θα ακολουθήσει θα αποκρύψει αυτό το σφάλμα.

Για να αποφύγουμε αυτού του τύπου την απόκρυψη σφαλμάτων εκτός από τις ανώτερες Word – Lines θα πρέπει να διαβάσουμε και τις κατώτερες πριν τις γράψουμε. Στην ουσία λοιπόν, για να εντοπίσουμε τα σφάλματα NWSF πρέπει να διαβάσουμε κάθε Word – Line πριν την γράψουμε. Ο τροποποιημένος αλγόριθμος (Αλγόριθμος 5.3) που προτείνεται, παρουσιάζεται ακολούθως. Το όνομα που επιλέξαμε για τον αλγόριθμο είναι TLAPNWPSSFΔT – W.O , τα αρχικά του οποίου σημαίνουν Test and Locate Active and

Passive Neighborhood Word – line and Pattern Sensitive Single Fault using the Δ – neighborhood and the Tiling method – Word Oriented.



Σχήμα 5.10 Απόκρυψη NWSF Σφάλματος σε Μνήμη Λειτουργίας Λέξης

Σε αυτό τον αλγόριθμο το συνολικό κόστος των επιπλέον αναγνώσεων είναι ίσο με το κόστος των εγγραφών, αφού διαβάζουμε κάθε Word – Line που γράφουμε. Επομένως το συνολικό κόστος ενός αλγορίθμου που εκτελεί έλεγχο για NPSF και NWSF σφάλματα σε μνήμες με μήκος λέξης $w = 2^q$ χρησιμοποιώντας την γειτονιά Τύπου – Δ και την υπόθεση Απλού Σφάλματος θα δίνεται από την ακόλουθη εξίσωση:

$$n*(2^{6-q} + 2*2^{5-q} + 2^{-q+1}) = n*(2^{7-q} + 2^{-q+1}) \quad q \geq 1 \qquad \text{Εξ } 5.4$$

Για μήκος λέξης w = 2 χρειαζόμαστε 65*n πράξεις, για w = 4 χρειαζόμαστε 32.5*n ενώ για w = 8 χρειαζόμαστε 16.3*n πράξεις.

```
BIT MATRIX A[4], OLD_A[4], P[2],OLD_P[2];
// Εφαρμογή και ανάγνωση του πρώτου διανύσματος
A := OLD_A := generate_pattern(0);
P := high_bit(A);

for  i := 1 to p do
        {
        apply_to_all_pattern(P,i);
        if ( i mod 2 = 0 ) then shift_right_pattern(P,1);          //even
        else if (  (i +1) /2 mod 2 ≠ 0 ) then P := low_bit(A);     // 1,5,9 …
        else    P := high_bit(A);                                  // 3,7,11…
        }
read_memory(A);

// Εφαρμογή και ανάγνωση των υπολοίπων διανυσματων

for  j := 1 to k* 2^k do
        {
        A := generate_pattern(j);

        if ( diff(A, OLD_A) = LOW ) then
                {
                P := low_bit(A);
                OLD_P := low_bit(OLD_A);
                init := 2;
                }
        else    {
                P := high_bit(A);
                OLD_P := high_bit(OLD_A);
                init := 1;
                }
        OLD_A := A;

        for  i := init to p do
                {
                compare_pattern(OLD_P,i);
                apply_pattern(P, i);
                if ( i mod 2 ≠ 0 ) then i = i + 2;
                shift_right_pattern(P,1);
                shift_right_pattern(OLD_P,1);
                }
        read_memory(A);
        }
```

Η συνάρτηση **read_memory(P)** είναι όπως προηγουμένως.

Αλγόριθμος 5.3 Ο Αλγόριθμος TLAPNWPSSFΔT – W.O. για Εντοπισμό των NPSF και NWSF Απλών Σφαλμάτων στις Μνήμες Λειτουργίας Λέξης

5.4. Σφάλματα NWSF και Διάνυσμα Δ - Γειτονιάς

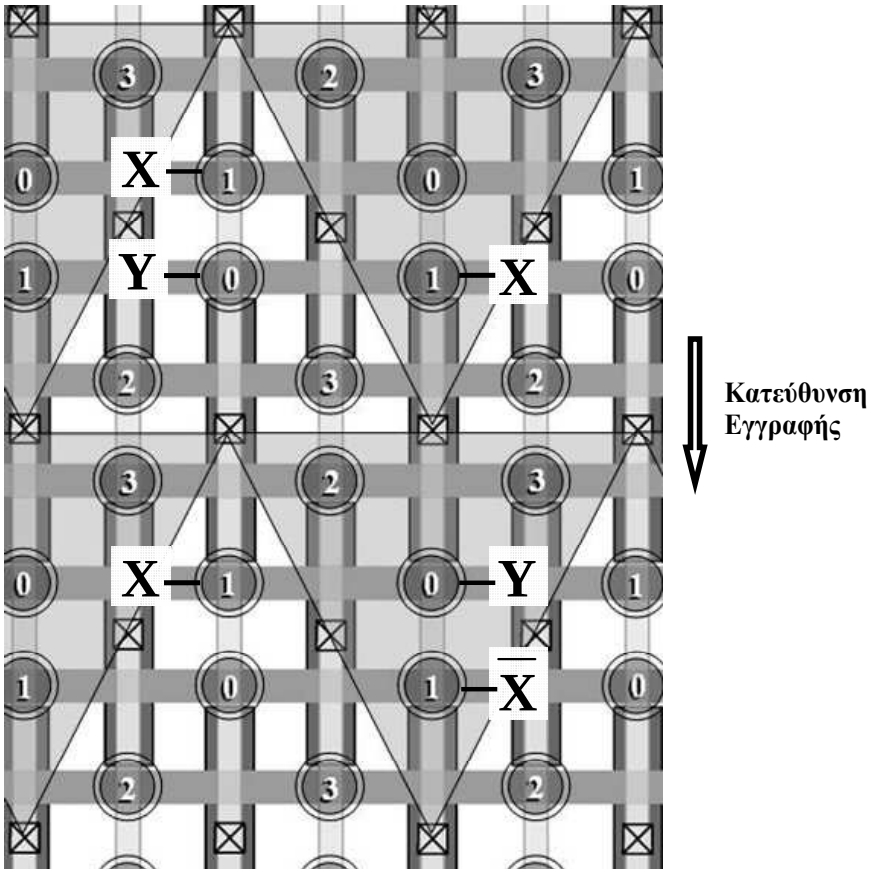
Στην προηγούμενη παράγραφο υποθέσαμε ότι οι δύο μηχανισμοί σφαλμάτων NPSF και NWSF είναι ανεξάρτητοι. Προκειμένου όμως να πετύχουμε καλύτερη κάλυψη σφαλμάτων ενδέχεται να θέλουμε να πραγματοποιήσουμε έλεγχο για NWSF σφάλματα για κάθε δυνατό συνδυασμό τιμών των κυττάρων της Τύπου – Δ γειτονιάς, με στόχο να εντοπίσουμε πιθανά NWSF σφάλματα τα οποία είναι τόσο ασθενή ώστε να εκδηλώνονται μόνο υπό την παρουσία κάποιου συγκεκριμένου διανύσματος γειτονιάς. Αυτό το νέο μοντέλο σφαλμάτων θα το ονομάσουμε Μοντέλο Σφαλμάτων Αλληλεπίδρασης Γειτονικών Word – Lines Υπό Συνθήκη (*Neighborhood Constrained Word - line Sensitive Fault – NCWSF*). Στη συνέχεια θα προτείνουμε αλγορίθμους που με χρήση της γειτονιάς τύπου –Δ και με την Υπόθεση Απλού Σφάλματος θα εντοπίζουν NPSF και NCWSF σφάλματα.

Έχουμε προαναφέρει ότι όταν η Word – Line του συνορεύοντος κυττάρου ενεργοποιείται, η τιμή της Bit – Line του κυττάρου βάσης (του οποίου η Word – Line είναι γειτονική αυτής του συνορεύοντος) καθορίζεται από την τιμή που είναι γραμμένη ή που πρόκειται να γραφεί στο συνορεύον κύτταρο. Εφόσον το συνορεύον κύτταρο έχει τον ίδιο αριθμό με το κορυφαίο κύτταρο της γειτονιάς του κυττάρου βάσης, εκ πρώτης όψεως φαίνεται ότι όταν ενεργοποιείται με αυτό τον τρόπο ο μηχανισμός NWSF η Bit – Line του κυττάρου βάσης θα έχει πάντα την ίδια τιμή με το κορυφαίο κύτταρο και ως εκ τούτου δεν είναι δυνατοί όλοι οι συνδυασμοί τιμών της Bit – Line με τις τιμές των κυττάρων της γειτονιάς. Στη συνέχεια θα δούμε πως δεν είναι ακριβώς έτσι τα πράγματα και πως προσθέτοντας κάποιες επιπλέον λειτουργίες στον αλγόριθμο μπορούμε να επιτύχουμε την επιθυμητή κάλυψη σφαλμάτων.

Για να διευκολυνθεί η περαιτέρω ανάλυση θα εισάγουμε τον εξής συμβολισμό: με ↕A|BT> συμβολίζουμε ένα κάτω προσανατολισμένο τρίγωνο στο οποίο γίνεται μετάβαση του συνορεύοντος κυττάρου στη λογική τιμή A ενώ το κύτταρο βάσης φέρει την τιμή B και το κορυφαίο κύτταρο την τιμή T (A, B, T $\in$ [0,1]). Παρόμοια, με <TB|A↕ αναφερόμαστε σε ένα άνω προσανατολισμένο τρίγωνο.

Αν εφαρμόσουμε έναν από τους αλγορίθμους της Τύπου –Δ γειτονιάς που προαναφέραμε γράφοντας τις Word – Lines από επάνω προς τα κάτω, τότε όπως βλέπουμε στο Σχήμα 5.11 σε ένα κάτω προσανατολισμένο τρίγωνο η μετάβαση του συνορεύοντος κυττάρου θα προηγηθεί της μετάβασης του κορυφαίου κυττάρου. Έτσι, όταν το συνορεύον κύτταρο θα

εκτελέσει την μετάβαση, το κορυφαίο κύτταρο θα έχει ακόμα την παλιά τιμή του, που θα είναι η συμπληρωματική της νέας τιμής του συνορεύοντος. Αυτή η συνθήκη συμβολίζεται με $\updownarrow X \mid Y\overline{X} >$.



Σχήμα 5.11 Οι Συνθήκες που Εμφανίζονται με Κατεύθυνση Εγγραφής από Επάνω προς τα Κάτω

Αντίστοιχα στα επάνω προσανατολισμένα τρίγωνα η ισχύουσα συνθήκη θα είναι $< XY \mid X \updownarrow$ γιατί εκεί η μετάβαση του κορυφαίου τριγώνου προηγείται της μετάβασης του συνορεύοντος τριγώνου. Προφανώς αν αντιστρέψουμε τη σειρά εγγραφής των Word – Lines θα αντιστραφούν και οι παραπάνω συνθήκες και θα εμφανιστούν οι καταστάσεις $\updownarrow X \mid YX >$ και $< \overline{X}Y \mid X \updownarrow$.

Οι τέσσερις προαναφερθείσες συνθήκες είναι όλες όσες μπορούν να προκύψουν. Έτσι αντιλαμβανόμαστε ότι αν εφαρμόσουμε δύο φορές τον αλγόριθμο εκτελώντας τις εγγραφές τη μία φορά από επάνω προς τα κάτω και την άλλη από κάτω προς τα επάνω θα επιτυγχάναμε

την επιθυμητή κάλυψη σφαλμάτων. Όμως αυτό θα διπλασίαζε το κόστος, πράγμα ανεπιθύμητο και, όπως θα δούμε στη συνέχεια, όχι αναγκαίο.

Όπως έχουμε προαναφέρει, με την πρώτη εφαρμογή του κατάλληλου αλγορίθμου (αλγόρ. 5.1 για λειτουργία bit και 5.3 για λειτουργία λέξης) καλύπτουμε όλα τα NPSF σφάλματα και τα μισά από τα NCWSF. Στόχος μας λοιπόν στη δεύτερη εφαρμογή όπου εκτελούμε τις εγγραφές με αντίθετη φορά είναι απλά να καλύψουμε τα υπόλοιπα NCWSF σφάλματα. Συνεπώς στη δεύτερη εφαρμογή δεν χρειάζεται να διαβάζουμε όλη τη μνήμη μετά από κάθε διάνυσμα ελέγχου που εγγράφεται αλλά μόνο τα κύτταρα που παίζουν το ρόλο του κυττάρου βάσης στις τριπλέτες $\updownarrow A \mid BT >$ και $< TB \mid A \updownarrow$ έτσι ώστε να καλύψουμε και τις καταστάσεις $\updownarrow X \mid YX >$ και $< \overline{X}Y \mid X \updownarrow$.

Εύκολα βλέπουμε ότι τα κύτταρα που πρέπει να διαβάσουμε είναι τα συζυγή αυτών που εκτελούν τη μετάβαση. Δηλαδή, αν τα κύτταρα με αριθμό 1 εκτελούν μετάβαση πρέπει να διαβάσουμε αυτά με αριθμό 0 και αντιστρόφως. Αντίστοιχα, αν τα κύτταρα με αριθμό 2 εκτελούν μετάβαση πρέπει να διαβάσουμε αυτά με αριθμό 3 και αντιστρόφως. Με άλλα λόγια, στη δεύτερη εφαρμογή θα διαβάσουμε τα κύτταρα που βρίσκονται στις Word – Lines που συμμετείχαν στη διαδικασία εγγραφής και που δεν εκτέλεσαν μετάβαση.

Η εφαρμογή αυτής της μεθόδου οδηγεί σε διαφορετικούς αλγορίθμους στις μνήμες λειτουργίας – bit σε σχέση με αυτές λειτουργίας λέξης προκειμένου να μην έχουμε απόκρυψη σφαλμάτων στις δεύτερες. Έτσι, στις μνήμες λειτουργίας – bit η ανάγνωση στη δεύτερη εφαρμογή μπορεί να γίνει στο τέλος της εγγραφής του εκάστοτε διανύσματος.

Στις μνήμες λειτουργίας λέξης υπενθυμίζουμε κατ’ αρχήν ότι το κύτταρο βάσης στα άνω τρίγωνα ενεργοποιείται πάντα από μία ανώτερη Word – Line ενώ στα κάτω τρίγωνα από μία κατώτερη Word – Line, ανεξαρτήτως του αριθμού που φέρει. Στη δεύτερη εφαρμογή του αλγορίθμου θέλουμε κατ’ αρχήν να διαβάσουμε τα κύτταρα βάσης μετά την μετάβαση του συνορεύοντος κυττάρου που ενεργοποιεί τον NWSF μηχανισμό. Αυτή η μετάβαση συμβαίνει πριν την επανεγγραφή του κυττάρου βάσης για τα άνω τρίγωνα, αφού η φορά εγγραφής είναι από κάτω προς τα πάνω και μετά την επανεγγραφή του για τα κάτω τρίγωνα. Συνεπώς στα κάτω τρίγωνα ισχύουν τα εξής: α) δεν έχουμε πρόβλημα απόκρυψης σφάλματος αφού η επανεγγραφή προηγείται της ενεργοποίησης του μηχανισμού NWSF, β) δεν έχει νόημα να

κάνουμε ανάγνωση του κυττάρου βάσης πριν την εγγραφή του γιατί δεν έχει εκτελέσει ακόμα μετάβαση το συνορεύων κύτταρο και συνεπώς δεν έχει ενεργοποιηθεί ακόμα ο NWSF μηχανισμός. Άρα η ανάγνωση πρέπει να γίνει αφού ολοκληρωθεί η εγγραφή της μνήμης. Αντίθετα, στα άνω τρίγωνα η μετάβαση του συνορεύοντος κυττάρου προηγείται της επανεγγραφής, άρα πρέπει να διαβάσουμε το κύτταρο βάσης πριν την επανεγγραφή του. Με βάση τα παραπάνω μπορούμε να καταλήξουμε στην εξής μέθοδο για τις μνήμες λειτουργίας λέξης: στη δεύτερη εφαρμογή και στις Word – Lines που πρέπει να γραφούν, τις μεν ανώτερες τις διαβάζουμε αμέσως πριν τις γράψουμε ενώ τις κατώτερες τις διαβάζουμε όταν τελειώσουν οι εγγραφές για το τρέχον διάνυσμα.

Οι δύο προτεινόμενοι Αλγόριθμοι 5.4 και 5.5 που προκύπτουν από την προηγούμενη περιγραφή παρουσιάζονται στις επόμενες σελίδες. Η συνάρτηση execute_first_pass() υποδεικνύει την εκτέλεση της πρώτης εφαρμογής, δηλαδή του πρώτου ‘περάσματος’. Αυτό το πρώτο πέρασμα είναι για μεν τον αλγόριθμο 5.4 η εκτέλεση του αλγορίθμου 5.1 (λειτουργία bit), για τον δε αλγόριθμο 5.5 είναι η εκτέλεση του 5.3 (λειτουργία λέξης). Και στους δύο αλγορίθμους έχουμε υποθέσει, χωρίς βλάβη της γενικότητας, ότι ο αριθμός των Word – Lines είναι δύναμη του 2, και συνεπώς η πρώτη και η τελευταία Word – Line ανήκουν στην μία ομάδα από Word - Lines (έστω γκρίζα) ενώ η δεύτερη και η προτελευταία στην άλλη (έστω λευκή). Το όνομα του αλγορίθμου 5.4 είναι TLAPNPCWSSFΔT – B.O. και του 5.5 TLAPNPCWSSFΔT - W.O., όπου τα αρχικά TLAPNPCWSSFΔT σημαίνουν Test and Locate Active and Passive Neighborhood Pattern and Constrained Word – line Sensitive Single Faults using the Δ – neighborhood and the Tiling method.

Για να υπολογίσουμε το κόστος του κάθε αλγορίθμου αρκεί να υπολογίσουμε το κόστος του δεύτερου περάσματος και να το προσθέσουμε σε αυτό του πρώτου που υπολογίσαμε νωρίτερα. Το κόστος του δεύτερου περάσματος του αλγορίθμου 5.4 που εφαρμόζεται στις μνήμες λειτουργίας bit είναι 16*n εγγραφές και 16*n αναγνώσεις. Άρα για αυτόν τον αλγόριθμο το συνολικό κόστος είναι $n*(82+16+16) = 114*n$. Αντίστοιχα, για μνήμες λειτουργίας λέξης (αλγόριθμος 5.5) έχουμε $n*2^{5-q}$ εγγραφές και $n*2^{5-q}$ αναγνώσεις ($q \geq 1$). Άρα το συνολικό κόστος του αλγορίθμου για μνήμες λειτουργίας λέξης είναι:

$$n*(2^{7-q} + 2^{6-q} + 2^{-q+1}) = n*(3* 2^{6-q} + 2^{-q+1}) \text{ , } q \geq 1 \qquad \text{Εξ. 5.5}$$

Για μήκος λέξης w = 2 χρειαζόμαστε $97*n$ πράξεις, για w = 4 χρειαζόμαστε $48.5*n$ ενώ για w = 8 χρειαζόμαστε $24.3*n$ πράξεις. Παρατηρούμε ότι εξακολουθούμε να έχουμε πολύ μεγάλη διαφορά στο κόστος σε σχέση με τους αντίστοιχους αλγορίθμους της Προ. Τύπου – 1 γειτονιάς, όπως το είδαμε στον Πίνακα 5.1.

Να σημειώσουμε ότι και στους δύο προαναφερθέντες αλγορίθμους στο δεύτερο πέρασμα δεν χρειάζεται να αρχικοποιήσουμε τη μνήμη (δηλαδή να γράψουμε όλα τα κύτταρα της μνήμης με το πρώτο διάνυσμα ελέγχου) αφού αυτή έχει ήδη το τελευταίο διάνυσμα του πρώτου περάσματος.

```
BIT MATRIX A[4], OLD_A[4], P[2];
// εκτέλεση πρώτου περάσματος
execute_first_pass();

// εκτέλεση δεύτερου περάσματος

for  j := 1 to k* 2^k do
        {
        A := generate_pattern(j);

        if ( diff(A, OLD_A) = LOW ) then
                {
                P := low_bit(A);
                init := p -1;
                }
        else
                {
                P := high_bit(A);
                init := p;
                }
        shift_right_pattern(P,1);
        OLD_A = A;

        for  i := init downto 1 do
                {
                apply_pattern(P, i);
                if ( i mod 2 = 0 ) then i := i - 2;
                shift_right_pattern(P,1);
                }
        second_pass_read (P,init);
        }
```

Η συνάρτηση **second_pass_read (P, init)** είναι η εξής:

```
function second_pass_read (P, init)
        {
        for  i := init downto 1 do
                {
                compare_pattern(P, i);
                if ( i mod 2 = 0 ) then i := i - 2;
                shift_right_pattern(P,1);
                }
        }
```

Αλγόριθμος 5.4 Ο Αλγόριθμος TLAPNPCWSSFΔΤ – Β.Ο για Εντοπισμό των NPSF και NCWSF Απλών Σφαλμάτων στις Μνήμες Λειτουργίας bit.

```
    BIT MATRIX A[4], OLD_A[4], P[2], OLD_P[2];
// εκτέλεση πρώτου περάσματος
execute_first_pass();

// εκτέλεση δεύτερου περάσματος

for  j = 1 to k* 2^k do
        {
        A = generate_pattern(j);

        if ( diff(A, OLD_A) = LOW ) then
                {
                P := low_bit(A);
                shift_right_pattern(P,1);
                OLD_P := low_bit(OLD_A);
                init := p -1 ;
                }
        else
                {
                P = high_bit(A);
                shift_right_pattern(P,1)
                OLD_P= high_bit(OLD_A);
                shift_right_pattern(OLD_P,1)
                init = p;
                }
        OLD_A = A;

        for  i := init downto 1 do
                {
                if ( i mod 2 = 0 ) then compare_pattern(OLD_P, i);        //ανώτερες
                apply_pattern(P, i);
                if ( i mod 2 ≠ 0 ) then i = i - 2;
                shift_right_pattern(P,1);
                }
        second_pass_read (P,init);
        }
```

Η συνάρτηση **second_pass_read (P, init)** είναι η εξής:

```
function second_pass_read (P, init)              // μόνο τις κατώτερες
        {
        if ( init mod 2 = 0 ) then init := init - 3;
        else shift_right_pattern(P,1);
        for  i := init downto 1 do
                {
                compare_pattern(P, i);
                i = i - 4;
                }
        }
```

Αλγόριθμος 5.5 Ο Αλγόριθμος TLAPNPCWSSFΔT – W.O για Εντοπισμό των NPSF και NCWSF Απλών Σφαλμάτων στις Μνήμες Λειτουργίας Λέξης.

5.5. Ταυτόχρονη Εμφάνιση NCWSF και NPSF Σφαλμάτων

Στην προηγούμενη παράγραφο παρουσιάσαμε αλγορίθμους για την κάλυψη NPSF και NCWSF σφαλμάτων με την υπόθεση ότι δεν είναι δυνατόν ένα κύτταρο να επηρεαστεί ταυτόχρονα και από τους δύο μηχανισμούς. Τώρα θα εξετάσουμε το ενδεχόμενο να υπάρξει ταυτόχρονη εμφάνιση NPSF και NCWSF σφάλματος για το ίδιο κύτταρο βάσης (*πολλαπλά σφάλματα – multiple fault assumption*).

Όπως γνωρίζουμε, ένα NWSF σφάλμα ενεργοποιείται από το συνορεύον κύτταρο, το οποίο φέρει τον ίδιο αριθμό με το κορυφαίο κύτταρο. Συνεπώς, εκείνη η κατάσταση που ενδεχομένως θα μπορούσε να συμβεί είναι να υπάρξει αλλαγή δεδομένων στο κύτταρο βάσης εξαιτίας μετάβασης του συνορεύοντος κυττάρου (ενεργοποίηση NWSF σφάλματος) και στη συνέχεια εκ νέου αλλαγή δεδομένων του κυττάρου βάσης εξαιτίας μετάβασης του κορυφαίου κυττάρου (ενεργοποίηση ANPSF σφάλματος) ή η αντίστροφη σειρά γεγονότων, ανάλογα με τον προσανατολισμό του τριγώνου που εξετάζουμε. Το αποτέλεσμα είναι ότι σε αυτές τις περιπτώσεις θα συμβεί απόκρυψη σφάλματος, καθώς η ενεργοποίηση του ενός μηχανισμού θα προκαλέσει σφάλμα στο κύτταρο βάσης και η ενεργοποίηση του δεύτερου μηχανισμού θα επαναφέρει στο κύτταρο τα σωστά δεδομένα.

Η περίπτωση απόκρυψης σφάλματος που μόλις περιγράψαμε, εύκολα μπορούμε να δείξουμε ότι δεν είναι δυνατόν να εμφανιστεί κατά την εφαρμογή του αλγόριθμου 5.5 που παρουσιάσαμε για έλεγχο μνημών λειτουργίας λέξης. Αυτό συμβαίνει γιατί οι επιπλέον αναγνώσεις που αυτός ο αλγόριθμος επιβάλλει προκειμένου να αποφευχθεί η απόκρυψη σφάλματος λόγω της αναπόφευκτης επανεγγραφής του κυττάρου βάσης με τα σωστά δεδομένα καλύπτουν και την περίπτωση απόκρυψης σφάλματος λόγω ταυτόχρονης ενεργοποίησης του NPSF και του NWSF μηχανισμού. Πράγματι, ο αλγόριθμος επιβάλλει την ανάγνωση του κυττάρου βάσης ανάμεσα στην μετάβαση του κορυφαίου κυττάρου και αυτή του συνορεύοντος κυττάρου (με όποια σειρά και αν γίνονται αυτές) και επομένως η επίδραση του κάθε μηχανισμού εξετάζεται ξεχωριστά. Άρα ο αλγόριθμος 5.5 εξ ορισμού καλύπτει τα NPSF και τα NCWSF σφάλματα στην περίπτωση που επηρεάσουν και τα δύο το ίδιο κύτταρο.

Δεν συμβαίνει όμως το ίδιο με τις μνήμες λειτουργίας – bit γιατί ο αλγόριθμός τους δεν περιλαμβάνει επιπλέον ενδιάμεσες αναγνώσεις. Για το λόγο αυτό θα ενισχύσουμε τον

αλγόριθμο με κατάλληλες ενδιάμεσες αναγνώσεις προκειμένου να αποφύγουμε την περίπτωση απόκρυψης σφάλματος που περιγράψαμε προηγουμένως.

Οι αναγνώσεις αυτές μπορούν να ενσωματωθούν στο πρώτο πέρασμα του αλγορίθμου. Εκείνο που προτείνουμε έχει ως ακολούθως: για κάθε Word – Line που συμμετέχει στην εγγραφή του διανύσματος, αφού κάνουμε τις εγγραφές θα διαβάσουμε τα κύτταρα που δεν συμμετείχαν στην εγγραφή, δηλαδή τα συζυγή αυτών που γράφηκαν. Αυτή η προσέγγιση βασίζεται στις εξής δύο παρατηρήσεις:

- Το κύτταρο που παίζει το ρόλο του κυττάρου βάσης στις τριπλέτες ↕A|BT> και <TB|A↕ είναι συζυγές του συνορεύοντος και του κορυφαίου κυττάρου. Συνεπώς η Word – Line του περιέχει κύτταρα με τον ίδιο αριθμό που φέρουν το κορυφαίο και το συνορεύον κύτταρο (τα οποία θα εκτελέσουν μετάβαση) και επομένως αυτή η Word – Line θα συμμετέχει στη διαδικασία εγγραφής.
- Γράφοντας τις Word – Lines με τη σειρά, η προσπέλαση της Word – Line του κυττάρου βάσης θα γίνει ανάμεσα στην προσπέλαση της Word – Line του κορυφαίου κυττάρου και αυτής του συνορεύοντος. Αυτό σημαίνει ότι η ανάγνωση του κυττάρου βάσης θα γίνει ανάμεσα στις μεταβάσεις του συνορεύοντος και του κορυφαίου κυττάρου. Συνεπώς, οι επιδράσεις αυτών των μεταβάσεων στα δεδομένα του κυττάρου βάσης θα εξεταστούν ξεχωριστά.

Στο δεύτερο πέρασμα δεν χρειάζονται επιπλέον αναγνώσεις, αφού δεν μας απασχολεί πλέον το ενδεχόμενο εμφάνισης διπλού σφάλματος. Αυτό συμβαίνει γιατί κατά τη διάρκεια του πρώτου περάσματος έχουμε καλύψει πλήρως τα NPSF σφάλματα (και τα μισά NCWSF) και επομένως αν μπορεί να εμφανιστεί κάποιο NPSF σφάλμα θα το έχουμε ήδη εντοπίσει και θα έχουμε κατατάξει τη μνήμη ως ελαττωματική, οπότε δεν υπάρχει λόγος να εκτελεστεί το δεύτερο πέρασμα. Με άλλα λόγια, αν πρόκειται να εκτελέσουμε δεύτερο πέρασμα αυτό σημαίνει ότι έχουμε αποκλείσει το ενδεχόμενο εμφάνισης NPSF σφάλματος και ως εκ τούτου και τον ενδεχόμενο πολλαπλού σφάλματος, οπότε θέτουμε μοναδικό στόχο να καλύψουμε τα υπόλοιπα μισά NCWSF.

Ο αλγόριθμος που εκτελεί έλεγχο NPSF και NCWSF σφαλμάτων με υπόθεση πολλαπλού σφάλματος σε μνήμες λειτουργίας bit παρουσιάζεται στην επόμενη σελίδα (Αλγόριθμος 5.6). Το όνομα του αλγορίθμου 5.6 είναι TLAPNPCWSMFΔT, όπου τα αρχικά σημαίνουν Test

and Locate Active and Passive Neighborhood Pattern and Constrained Word – line Sensitive Multiple Faults using the Δ – neighborhood and the Tiling method. Η συνάρτηση *compare_bit_pattern(P,i)* που εμφανίζεται εδώ συγκρίνει μόνο τα bit που δεν γράφηκαν προηγουμένως, δηλαδή τα κύτταρα που φέρουν συζυγή αριθμό σε σχέση με αυτά που γράφηκαν. Η συνάρτηση *execute_second_pass()* δηλώνει ότι σε αυτό το σημείο εκτελούμε το δεύτερο πέρασμα όπως αυτό περιγράφεται από τον αλγόριθμο 5.4.

Το κόστος των επιπλέον αναγνώσεων είναι όσο και το κόστος εγγραφής για κάθε διάνυσμα ελέγχου, αφού σε κάθε Word - Line τα μισά κύτταρα γράφονται και τα άλλα μισά διαβάζονται. Άρα το επιπλέον κόστος είναι 16*n σε σχέση με τον αλγόριθμο 5.4, οπότε το συνολικό κόστος του αλγορίθμου ανέρχεται σε 130*n, σημαντικά μικρότερο από το κόστος της προσαρμοσμένης Τύπου – 1 γειτονιάς. Στον Πίνακα 5.2 βλέπουμε μία συνολική σύγκριση των αλγορίθμων που έχουμε παρουσιάσει. Να σημειώσουμε ότι οι αλγόριθμοι της Προ. Τύπου – 1 γειτονιάς παρέχουν κάλυψη πολλαπλών NPSF και NCWSF σφαλμάτων, αφού εκεί τα κύτταρα που αντιστοιχούν στο συνορεύον και το κορυφαίο κύτταρο της Τύπου – Δ γειτονιάς φέρουν διαφορετικό αριθμό και συνεπώς, αφενός θα πάρουν μεταξύ τους όλες τις δυνατές τιμές, αφετέρου δεν θα εκτελούν μετάβαση κατά τη διάρκεια της εφαρμογής των ίδιων διανυσμάτων.

```
    BIT MATRIX A[4], OLD_A[4], P[2],OLD_P[2];
// Εφαρμογή και ανάγνωση του πρώτου διανύσματος
A := OLD_A := generate_pattern(0);
P := high_bit(A);

for  i := 1 to p do
        {
        apply_to_all_pattern(P,i);
        if ( i mod 2 = 0 ) then shift_right_pattern(P,1);          //even
        else if (  (i +1) /2 mod 2 ≠ 0 ) then P := low_bit(A);     // 1,5,9 …
        else    P := high_bit(A);                                  // 3,7,11…
        }
read_memory(A);

// Εφαρμογή και ανάγνωση των υπολοίπων διανυσματων
for  j := 1 to k* 2^k do
        {
        A := generate_pattern(j);

        if ( diff(A, OLD_A) = LOW ) then
                {
                P := low_bit(A);
                init := 2;
                }
        else
                {
                P := high_bit(A);
                init := 1;
                 }
        OLD_A := A;

        for  i := init to p do
                {
                apply_pattern(P, i);
                compare_bit_pattern(P,  i);
                if ( i mod 2 ≠ 0 ) then i = i + 2;
                shift_right_pattern(P,1);
                }
        read_memory(A);
        }

execute_second_pass();
```

Αλγόριθμος 5.6 Ο Αλγόριθμος TLAPNPCWSMFΔT για Εντοπισμό των NPSF και NCWSF Πολλαπλών Σφαλμάτων στις Μνήμες Λειτουργίας bit

Πίνακας 5.2 Σύγκριση Κόστους Αλγορίθμων Προ.Τύπου – 1 Γειτονιάς και της Τύπου - Δ Γειτονιάς σε Όλες τις Περιπτώσεις Σφαλμάτων.

#bit λέξης	**Προ.Τύπου - 1**		**Τύπου - Δ**											
	TLAPNPSFA1T	TLAPNPSFA1T – W.O.	TLAPNPSFΔT		TLAPNPSFΔT – W.O.		TLAPNWPSSFΔT – W.O		TLAPNCWPSSFΔT – B.O		TLAPNCWPSSFΔT– W.O		TLAPNCWPSMFΔT– B.O	
			κόστος	**κέρδος**	**κόστος**	**κέρδος**	**κόστος**	**κέρδος**	**κόστος**	**κέρδος**	**κόστος**	**κέρδος**	**κόστος**	**κέρδος**
1-BIT	194*n	194*n	82*n	57.7%	-	-	-	-	114*n	41.2%	-	-	130*n	33.0%
2-BIT	113*n	129*n	49*n	56.6%	57*n	55.8%	65*n	49.6%	-	-	97*n	24.8%	-	-
4-BIT	72.5*n	100.5*n	24.5*n	66.2%	28.5*n	71.6%	32.5*n	67.7%	-	-	48.5*n	51.7%	-	-
8-BIT	40.3*n	60.3*n	12.3*n	69.6%	14.3*n	76.3%	16.3*n	73.0%	-	-	24.3*n	59.8%	-	-
16-BIT	20.1*n	30.1*n	6.1*n	69.6%	7.1*n	76.3%	8.1*n	73.0%	-	-	12.1*n	59.8%	-	-
32-BIT	10.1*n	15.1*n	3.1*n	69.6%	3.6*n	76.3%	4.1*n	73.0%	-	-	6.1*n	59.8%	-	-
κάλυψη σφαλμάτων σε λειτουργία bit	NPSF + NCWSF πολλαπλά σφάλματα	Δεν εφαρμόζεται	NPSF + NWSF απλά σφάλματα		Δεν εφαρμόζεται		Δεν εφαρμόζεται		NPSF + NCWSF απλά σφάλματα		Δεν εφαρμόζεται		NPSF + NCWSF πολλαπλά σφάλματα	
κάλυψη σφαλμάτων σε λειτουργία λέξης	Μερική κάλυψη NPSF + NCWSF	NPSF + NCWSF πολλαπλά σφάλματα	Μερική κάλυψη NPSF + NWSF		NPSF + μερική κάλυψη NWSF		NPSF + NWSF απλά σφάλματα		Δεν εφαρμόζεται		NPSF + NCWSF πολλαπλά σφάλματα		Δεν εφαρμόζεται	

* Η στήλη ‘κέρδος’ των αλγορίθμων της γειτονιάς Τύπου – Δ δίνει το ποσοστό μείωσης του κόστους που παρέχει ο εκάστοτε αλγόριθμος συγκριτικά με τον αλγόριθμο TLAPNPSFA1T – W.O. της Προ. Τύπου – 1 γειτονιάς, εκτός της περίπτωσης του αλγορίθμου TLAPNPSFΔT όπου η σύγκριση γίνεται με τον TLAPNPSFA1T.

ΚΕΦΑΛΑΙΟ 6. ΣΥΜΠΕΡΑΣΜΑΤΑ - ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

Στην παρούσα διατριβή μελετήσαμε την εφαρμογή του NPSF μοντέλου πάνω σε μνήμες DRAM εμφωλευμένης συστοιχίας, προσαρμόζοντας την Τύπου – 1 γειτονιά πάνω στο φυσικό σχεδιασμό της μνήμης και λύνοντας το πρόβλημα απόκρυψης σφαλμάτων σε μνήμες λειτουργίας λέξης. Στη συνέχεια προτείναμε την γειτονιά Τύπου – Δ η οποία δίνει τη δυνατότητα να μειωθεί δραματικά ο χρόνος εκτέλεσης του αλγορίθμου για ανίχνευση NPSF σφαλμάτων. Εισάγαμε επίσης ένα νέο μοντέλο σφαλμάτων, το NWSF. Ακολούθως, μελετήσαμε διάφορους τρόπους με τους οποίους μπορούν να συνδυαστούν τα δύο μοντέλα σφαλμάτων και προτείναμε αντίστοιχες μεθόδους ανίχνευσης των συνδυασμένων σφαλμάτων, πάντα με χρήση της γειτονιάς Τύπου – Δ. Σε κάθε περίπτωση διατηρήσαμε σημαντική μείωση του χρόνου ελέγχου συγκριτικά με την προσαρμοσμένη Τύπου – 1 Γειτονιά, ιδίως στις μνήμες λειτουργίας λέξης.

Σε μια μελλοντική επέκταση της διατριβής, θα μπορούσε να πραγματοποιηθεί η σχεδίαση ενός ενσωματωμένου κυκλώματος αυτοελέγχου (Built-In Self Test – BIST) το οποίο θα υλοποιεί έναν ή περισσότερους από τους προτεινόμενους αλγορίθμους και θα προσφέρει τη δυνατότητα ελέγχου ορθής λειτουργίας DRAM μνημών. Μία δεύτερη πιθανή κατεύθυνση μελλοντικής εργασίας είναι η προσπάθεια περαιτέρω μείωσης του κόστους των αλγορίθμων που προτάθηκαν. Ένας τρόπος για να γίνει αυτό είναι να εντοπιστούν (αν υπάρχουν) τα διανύσματα που έχουν τη μέγιστη δυνατότητα να ενεργοποιήσουν ένα σφάλμα, αυτά δηλαδή που μεγιστοποιούν τα αποτελέσματα των φαινομένων διαρροής φορτίων και / ή των χωρητικών συζεύξεων μεταξύ των κυττάρων και εν συνεχεία να γίνει αποκλειστική χρήση μόνο αυτών των διανυσμάτων.

ΑΝΑΦΟΡΕΣ

[1] J.A. Mandelman, R.H. Bennard, G.B Bronner, J.K. DeBrosse, R. Divakaruni, Y. Li and C.J. Radens, “Challenges and Future Directions for the Scaling of Dynamic Random-Access Memory (DRAM),” IBM Journal of Research and Development, vol. 46, no. 2/3, pp. 187-212, 2002.

[2] B. Keeth and R.J. Baker, “DRAM Circuit Design: A Tutorial,” IEEE Press, Series on Microelectronic Systems, 2001.

[3] Y. Matsubara, et al, “Fully Compatible Integration of High Density Embedded DRAM with 65nm CMOS Technology (CMOS5),” IEEE Electron Devices Meeting, pp. 423-426, 2003.

[4] C-H. Chung and J-W. Chien, “Memories Having Charge Storage Node at Least Partially Located in a Trench in a Semiconductor Substrate and Electrically Coupled to a Source/Drain Region Formed in the Substrate,” US Patent 7,348,622 B2, 2008.

[5] J. van de Goor, “Testing Semiconductor Memories-Theory and Practice,” John Wiley & Sons Ltd., 1991.

[6] P. Mazumder and K. Chakraborty, “Testing and Testable Design of High-Density Random-Access Memories,” Kluwer Academic Publishers, 1996.

[7] J. P. Hayes, "Testing Memories for Single-Cell Pattern-Sensitive Faults," IEEE Transactions on Computers, vol. 29, no. 3, pp. 249-254, 1980.

[8] S. Hamdioui, Z. Al-Ars, A.D.J. van de Goor and M. Rodgers, “Dynamic Faults in Random Access Memories: Concept, Fault Models and Tests,” Journal of Electronic Testing: Theory and Applications, vol. 19, pp. 195-205, 2003.

[9] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto, “Automatic March Test Generation for Static and Dynamic Faults, in SRAMs,” Proc. European Test Symposium (ETS), 2005.

[10] G. Hautunyan, V.A. Vardanian and Y. Zorian, “Minimal March Tests for Dynamic Faults in Random Access Memories,” Proc. European Test Symposium (ETS), pp. 43-48, 2005.

[11] S. Banerjee, D.R. Chowdhury and B.B. Bhattacharya, “A Programmable Built-In Self-Test for Embedded DRAMs,” IEEE Int. Workshop on Memory Technology Design and Testing, pp. 58-63, 2005.

[12] S. Boutobza, M. Nicolaidis, K. M. Lamara and A. Costa, “A Transparent Based Programmable Memory BIST,” IEEE European Test Symposium, pp. 89-94, 2006.

[13] S. Banerjee, D.R. Chowdhury and B.B. Bhattacharya, “A Programmable Built-In Self-Test for Embedded Memory Cores,” IETE Technical Review, vol. 24, no. 4, pp. 287-311, 2007.

[14] J.Y. Kim, S.J. Hong and j. Kim, “Parallely Testable Design for Detection of Neighborhood Pattern Sensitive Faults in High Density DRAMs,” IEEE Int. Symposium on Circuits and Systems, pp. 5854-5857, 2005.

[15] Y-J. Huang and J-F. Li, “Testing Active Neighborhood Pattern-Sensitive Faults of Ternary Content Addressable Memories,” IEEE European Test Symposium, pp. 55-60, 2006.

[16] A.J. van de Goor and I.B.S. Tlili, “Disturb Neighborhood Pattern Sensitive Fault,” IEEE Int. VLSI Test Symposium, pp. 37-45, 1997.

[17] M. Franklin, K. Saluja and K. Kinoshita, “A Built In Self Test Algorithm for Row/Column Pattern Sensitive Faults in RAMs,” IEEE Journal of Solid-State Circuits, vol. 25, no. 2, pp. 514-524, 1990.

[18] D-C. Kang, S.M. Park and S-B Cho, “An Efficient Built-In Self Test Algorithm for Neighborhood Pattern and Bit-Line-Sensitive Faults in High Density Memories,” ETRI Journal, vol. 26, no. 6, pp. 520-534, 2004.

[19] D-C. Kang and S-B Cho, “An Efficient Built-In Self Test Algorithm for Neighborhood Pattern and Bit-Line-Sensitive Faults in High Density Memories,” IEEE KORUS, pp. 218-223, 2000.

[20] A. Chrisanthopoulos, Th. Haniotakis, Y.Tsiatouhas and A. Arapoyanni, “New Test Pattern Generation Units for NPSF Oriented Memory Built-In Self Test,” IEEE International Conference on Electronics, Circuits and Systems (ICECS01), pp. 749-752, 2001.

[21] H.D. Oberle and P. Muhmenthaler, “Test Patten Development and Evaluation for

DRAMs with Fault Simulator RAMSIM," IEEE Int. Test Conference, pp. 548-555, 1991.

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Ο Γεώργιος Σφήκας γεννήθηκε στα Ιωάννινα το 1973. Τον Ιούλιο του 1998 αποφοίτησε από το Τμήμα Φυσικής και τον Δεκέμβριο του 2006 από το Τμήμα Πληροφορικής του Πανεπιστημίου Ιωαννίνων. Στο ίδιο ακαδημαϊκό έτος ξεκίνησε μεταπτυχιακές σπουδές στο Τμήμα Πληροφορικής. Τα ερευνητικά του ενδιαφέροντα κατά κύριο λόγο εστιάζονται στη σχεδίαση και τον έλεγχο ορθής λειτουργίας ολοκληρωμένων κυκλωμάτων.

Contact e-mail: gsfikas@yahoo.com