ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΣΥΣΧΕΤΙΣΤΙΚΩΝ ΜΝΗΜΩΝ (CAM) ΜΕ NAND ΓΡΑΜΜΗ ΤΑΥΤΟΠΟΙΗΣΗΣ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Μηχανικών Η/Υ και Πληροφορικής
Εξεταστική Επιτροπή

από τον

Γεώργιο Παπαθεοδώρου

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΙΣ ΤΕΧΝΟΛΟΓΙΕΣ-ΕΦΑΡΜΟΓΕΣ

Ιανουάριος 2014

ΑΦΙΕΡΩΣΗ

Στη μνήμη του αδερφού μου Αδαμαντίου.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κ. Ευθυμίου Αριστείδη, Επίκουρο Καθηγητή του Τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Ιωαννίνων για την εξαιρετική συνεργασία που είχαμε και για την συνεχή και υποδειγματική επιστημονική καθοδήγησή του κατά την εκπόνηση της εργασίας αυτής.

Επίσης θα ήθελα να ευχαριστήσω τους κυρίους Καβουσιανό Χρυσοβαλάντη, Επίκουρο Καθηγητή, Τσιατούχα Γεώργιο, Αναπληρωτή Καθηγητή και Παρσόπουλο Κωνσταντίνο, Επίκουρο Καθηγητή του Τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Ιωαννίνων για τις συμβουλές και την πολύτιμη βοήθεια που μου προσέφεραν.

Τέλος, θα ήθελα να ευχαριστήσω τη σύζυγό μου Γιώτα, τους γονείς μου Θωμά και Αρετή, των οποίων έχω την αδιάκοπη και ουσιώδη στήριξη καθ' όλη τη διάρκεια των σπουδών μου καθώς επίσης και το νέο μέλος της οικογένειας, την κόρη μου Αδαμαντία.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΛΗΨΗ

Γεώργιος Παπαθεοδώρου του Θωμά και της Αρετής.
MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιανουάριος, 2014.
Μοντελοποίηση Συσχετιστικών Μνημών (CAM) με NAND Γραμμή Ταυτοποίησης.
Επιβλέποντας: Αριστείδης Ευθυμίου.

Στην παρούσα εργασία αναπτύσσεται ένα παραμετροποιημένο μοντέλο συσχετιστικών μνημών, CAM, με οργάνωση γραμμής ταυτοποίησης (Match Line) τύπου NAND καθώς και του λογισμικού που αποτελεί την υλοποίηση του μοντέλου. Συγκεκριμένα, η ανάπτυξη του λογισμικού αποσκοπεί στον υπολογισμό του χρόνου προσπέλασης (access time), του κύκλου (cycle time), της επιφάνειας (area) και της δυναμικής καταναλισκόμενης ενέργειας (dynamic energy per access).

Οι συσχετιστικές μνήμες χρησιμοποιούνται σε εφαρμογές όπου απαιτείται γρήγορη αναζήτηση όπως σε κρυφές μνήμες (cache memory) και πίνακες δρομολόγησης μεταγωγών. Οι κρυφές μνήμες (Cache Memory) είναι απαραίτητες στα σύγχρονα υπολογιστικά συστήματα γιατί καλύπτουν το χάσμα απόδοσης μεταξύ του επεξεργαστή και της κύριας μνήμης.

Το μοντέλο που παρουσιάζεται, βασίζεται σε γενίκευση της NAND CAM που είχε προταθεί από τους Vikas Chaudhary και Lawrence Clark στο άρθρο “Low-Power High-Performance NAND Match Line Content Addressable Memories” [8], για οποιοδήποτε μέγεθος ετικέτας χρησιμοποιώντας ένα νέο αλγόριθμο διαχωρισμού της ετικέτας σε μικρότερα τμήματα. Το λογισμικό που υλοποιήθηκε, βασίστηκε σε προσεγγίσεις που έγιναν στο μοντέλο CACTI [4], το οποίο έχει καθιερωθεί στον χώρο της αρχιτεκτονικής υπολογιστών, λόγω της ταχύτητας που προσφέρει κατά την αποτίμηση διαφόρων αρχιτεκτονικών μνημών.

Αν και βασισμένο στο CACTI, το προτεινόμενο λογισμικό διαφοροποιείται σημαντικά, διότι είναι προσανατολισμένο στο να υπολογίζει τα προαναφερθέντα μεγέθη για NAND CAM, επιλογή η οποία έως τώρα δεν προσφέρεται από το CACTI, που μοντελοποιεί μόνο NOR CAMs. Η οργάνωση NOR είναι απλούστερη και θεωρείται ταχύτερη από τη NAND, αλλά προκαλεί σημαντικά μεγαλύτερη κατανάλωσης δυναμικής ενέργειας. Συνεπώς το προτεινόμενο μοντέλο καλύπτει ένα μεγάλο κενό του CACTI δίνοντας σε αρχιτέκτονες υπολογιστών τη δυνατότητα να διερευνήσουν τη χρήση NAND CAM σε ένα υπολογιστικό σύστημα.

Με το εργαλείο λογισμικού που υλοποιήθηκε, διεξήχθησαν υπολογισμοί του εμβαδού, της κατανάλωσης δυναμικής ενέργειας, του χρόνου και του κύκλου προσπέλασης, για μία σειρά από περιπτώσεις χρήσης NAND CAM, διαφόρων μεγεθών. Τα αποτελέσματα που παρουσιάστηκαν, έδειξαν πως όλα τα μεγέθη (εμβαδό, δυναμική ενέργεια, χρόνος και κύκλος προσπέλασης) αυξάνονται αναλογικά με το πλήθος των bits που απαρτίζουν τη tag line. Επιπλέον, έγινε σύγκριση της NAND CAM με NOR CAM, η μοντελοποίηση της οποίας έγινε με το CACTI. Η σύγκριση των υπολογισμένων μεγεθών για τις δύο διαφορετικές οργανώσεις, έδειξε πως η NOR CAM υπερέχει ελάχιστα στο κομμάτι της απαιτούμενης επιφάνειας καθώς και στο πεδίο του χρόνου, ενώ η NAND CAM εμφανίζει αξιοσημείωτα μειωμένη κατανάλωση δυναμικής ενέργειας σε σχέση με αυτή της NOR.

Τα αποτελέσματα που προέκυψαν από μετρήσεις με το εργαλείο που υλοποιήθηκε, βρίσκονται σε πλήρη αντιστοιχία με αποτελέσματα που προέκυψαν από λογισμικό προσομοίωσης τύπου SPICE καθώς επίσης και με αποτελέσματα από αντίστοιχες επιστημονικές εργασίες. Για τον λόγο αυτό, το προτεινόμενο λογισμικό μπορεί να αποτελέσει χρήσιμο εργαλείο στα χέρια των σχεδιαστών που ασχολούνται με την αύξηση της απόδοσης των κρυφών μνημών και κατ επέκταση των σύγχρονων υπολογιστικών συστημάτων.

EXTENDED ABSTRACT IN ENGLISH

Georgios, Th. Papatheodorou MSc, Computer Science Department, University of Ioannina, Greece. January, 2014.
Modeling a Content Addressable Memory (CAM) With NAND Match Line.
Thesis Supervisor: Aristides Efthymiou .

Content Addressable Memories (CAM) are essential components used in applications where fast associative search is required, such as, cache memories of computing systems and routing tables of network switches.

This dissertation presents a model of a parameterized CAM with NAND Match Line and the corresponding software tool. The software tool calculates the access time, the cycle time, the area and the dynamic energy per access of a CAM with NAND Match Line.

The model is based on a generalized version of the design proposed by Vikas Chaudhary and Lawrence Clark in the paper “Low-Power High-Performance NAND Match Line Content Addressable Memories” [8]. In contrast to the above work, the proposed model works for any tag size and any number of words using a novel algorithm for dividing the tag line into smaller parts.

The software tool is based on the CACTI model [4], which has been well-established in the field of computer architecture, as it provides great speed and convenience. Never the less, the developed software tool is significantly different as it has the ability of calculating the access time, the cycle time, the area and the dynamic energy per access of a CAM with NAND Match Line, an option which CACTI does not provide, as it only models NOR CAMs. Although the NOR CAM organization is

simpler and is considered to be faster, its dynamic power consumption is higher. Therefore the proposed model fills a void in high-level memory modeling and constitutes a useful tool for computer architects.

Using the developed software tool, a number of measurements were made for various sizes of NAND CAM. In each case the access time, cycle time, area and dynamic energy per access have been calculated. The results of the case study show that every aforesaid attribute is increasing linearly with tag line size. Moreover, a comparison of NAND CAM to NOR CAM has taken place, using CACTI to model the latter. The results of the comparison have shown that NOR CAM barely outclasses NAND CAM in cycle time and area, but in regard to dynamic search energy, NAND CAM shows better efficiency.

The results of the measurements conducted with the software tool are in line with results obtained from SPICE-level simulation and also with known results from relevant papers. Therefore, the software developed for the purposes of this dissertation is accurate enough to be a very useful tool in the hands of those who conduct research regarding cache memories and other computational systems.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

1.1. Εισαγωγή

Οι σύγχρονες τάσεις της Επιστήμης Σχεδιασμού των Υπολογιστικών Συστημάτων έχουν υιοθετήσει την χρήση των Κρυφών Μνημών (Cache), αποβλέποντας στην απόκρυψη της Καθυστέρησης της Κύριας Μνήμης των Συστημάτων (Memory Latency) και την γεφύρωση του χάσματος της απόδοσης του Επεξεργαστή και της Κύριας Μνήμης (Processor – Memory Performance Gap). Έτσι οι κρυφές μνήμες έχουν αποκτήσει αδιαμφισβήτητα πρωτεύοντα ρόλο στην Ιεραρχία Μνήμης των Ηλεκτρονικών Υπολογιστών.

Πολλές φορές οι κρυφές μνήμες πρέπει να έχουν μεγάλη συσχετιστικότητα (associativity) και ένας τρόπος σχεδιασμού τέτοιων κρυφών μνημών είναι με τη χρήση συσχετιστικής μνήμης (content addressable memory – CAM) για αποθήκευση και αναζήτηση ετικετών (tags). Οι μνήμες CAM βρίσκουν επίσης εφαρμογή και σε άλλα πεδία, όπως η αποθήκευση του πίνακα δρομολόγησης σε μεταγωγούς (switches) δικτύων δεδομένων.

Κεντρικό θέμα της εργασίας αυτής αποτελεί η ανάπτυξη ενός παραμετροποιημένου μοντέλου CAM Μνημών με οργάνωση NAND Match Line και του αντίστοιχου λογισμικού. Συγκεκριμένα, η ανάπτυξη του προγράμματος αποσκοπεί στον υπολογισμό του χρόνου προσπέλασης (access time), του κύκλου (cycle time), της

επιφάνειας (area) καθώς και της καταναλισκόμενης δυναμικής ενέργειας (energy per access).

Το λογισμικό που προτείνεται βασίστηκε σε προσεγγίσεις που έγιναν στο μοντέλο CACTI [4] για υπολογισμό των παραπάνω μέτρων επίδοσης σε CAM μνήμες με NOR Match Line – μία διαφορετική τεχνική σχεδίασης μνημών CAM, που έχει μεγάλη κατανάλωση ενέργειας. Το εργαλείο CACTI, χρησιμοποιείται για την αποτίμηση βελτιστοποιήσεων που επιχειρούνται σε μνήμες (κρυφές μνήμες διαφόρων οργανώσεων, SRAM, CAM, DRAM), προσφέροντας στους αρχιτέκτονες υπολογιστών, την απαιτούμενη πληροφορία σε πολύ μικρό χρόνο και με τον ελάχιστο κόπο, καθώς για την εξαγωγή των αποτελεσμάτων, απαιτούνται υψηλού επιπέδου πληροφορίες εισόδου.

Η ταχύτητα και η ευκολία με την οποία γίνεται η αποτίμηση των διαφόρων αρχιτεκτονικών, είναι κυρίως αυτή που έχει καθιερώσει το CACTI στον χώρο, διότι εναλλακτικά η διαδικασία αξιολόγησης μίας αρχιτεκτονικής μνήμης, θα πρέπει να γίνει με τη χρήση κάποιου εργαλείου τύπου SPICE (Simulation Program with Integrated Circuit Emphasis), πράγμα που την καθιστά επίπονη και κυρίως χρονοβόρα.

Αν και βασισμένο στις μεθόδους υπολογισμού CAM μνημών με NOR Match Line του CACTI, το προτεινόμενο λογισμικό διαφοροποιείται σημαντικά, διότι είναι προσανατολισμένο στο να υπολογίζει χρόνο πρόσβασης, κατανάλωση ενέργειας και επιφάνεια για NAND CAM, επιλογή η οποία έως τώρα δεν προσφέρεται από το CACTI. Επιπλέον, η εργασία γενικεύει τη μέθοδο σχεδίασης NAND CAM που είχε παρουσιαστεί στο [8] για οποιοδήποτε μέγεθος ετικέτας χρησιμοποιώντας ένα νέο αλγόριθμο διαχωρισμού της ετικέτας σε μικρότερα τμήματα.

Δεδομένου ότι τα αποτελέσματα που προέκυψαν από μετρήσεις με το εργαλείο που υλοποιήθηκε, βρίσκονται σε πλήρη αντιστοιχία με αποτελέσματα που προέκυψαν από λογισμικό προσομοίωσης τύπου SPICE καθώς επίσης και με αποτελέσματα από αντίστοιχες επιστημονικές εργασίες, το προτεινόμενο λογισμικό μπορεί να αποτελέσει χρήσιμο εργαλείο στα χέρια των σχεδιαστών που ασχολούνται με την

αύξηση της απόδοσης των cache μνημών και κατ επέκταση των σύγχρονων υπολογιστικών συστημάτων.

1.2. Δομή της Διατριβής

Η εργασία αυτή είναι χωρισμένη σε πέντε κεφάλαια. Αρχικά, στο κεφάλαιο 2 περιγράφεται η ιεραρχία μνήμης και οι λόγοι ύπαρξής της. Στη συνέχεια, αναλύεται η λειτουργία κα η οργάνωση της κρυφής μνήμης, καθώς επίσης και της συσχετιστικής μνήμης. Τέλος, παρουσιάζονται διαφορετικές αρχιτεκτονικές της συσχετιστικής μνήμης, όπως αυτή με NOR γραμμή ταυτοποίησης και με NAND γραμμή ταυτοποίησης και γίνεται αναφορά σε εργαλεία λογισμικού που χρησιμοποιήθηκαν.

Στο κεφάλαιο 3, παρουσιάζεται και αναλύεται το μοντέλο της συσχετιστικής μνήμης που χρησιμοποιείται στην εργασία. Στη συνέχεια αναλύονται οι μέθοδοι που ακολουθούνται για τους υπολογισμούς στο πεδίο του χρόνου και παρουσιάζεται η μεθοδολογία ανάλυσης του ισοδυνάμου δικτυώματος αντίστασης - πυκνωτή. Ακολουθεί η περιγραφή ενός αλγορίθμου διαχωρισμού των κυττάρων μίας συσχετιστικής μνήμης με NAND γραμμή ταυτοποίησης σε ομάδες και παρουσιάζεται το θεωρητικό του υπόβαθρο. Τέλος αναλύονται οι μέθοδοι υπολογισμού της απαιτούμενης επιφάνειας και της καταναλισκόμενης δυναμικής ενέργειας.

Στο κεφάλαιο 4, παρουσιάζεται και αξιολογείται λογισμικό που κατασκευάστηκε με σκοπό τον υπολογισμό του χρόνου πρόσβασης και κύκλου καθώς επίσης της δυναμικής ενέργειας και της επιφάνειας για μία συσχετιστική μνήμη με NAND γραμμή ταυτοποίησης οποιουδήποτε μεγέθους. Στη συνέχεια παρουσιάζονται τα αποτελέσματα από τα πειράματα που εκτελέστηκαν και αξιολογείται η απόδοση της NAND CAM για διάφορες περιπτώσεις χρήσης όπως επίσης και συγκρινόμενη με NOR CAM.

Τέλος, στο κεφάλαιο 5, αναφέρονται συνοπτικά τα συμπεράσματα που προέκυψαν από τις σειρές πειραμάτων και προτείνονται ιδέες για μελλοντική έρευνα.

ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ ΚΑΙ ΕΡΓΑΛΕΙΑ

2.1. Μνήμη Cache

Κρυφή Μνήμη (Cache) ονομάζεται μια σχετικά μικρής χωρητικότητας μνήμη, που χρησιμοποιείται για την αποθήκευση πληροφορίας, που αναμένεται ότι θα χρησιμοποιηθεί άμεσα ή με μεγάλη συχνότητα στο μέλλον. Κρυφή μνήμη χρησιμοποιείται συνήθως μεταξύ της Κεντρικής Μονάδας Επεξεργασίας και της κύριας μνήμης [1].

2.1.1. Ιστορική Αναδρομή

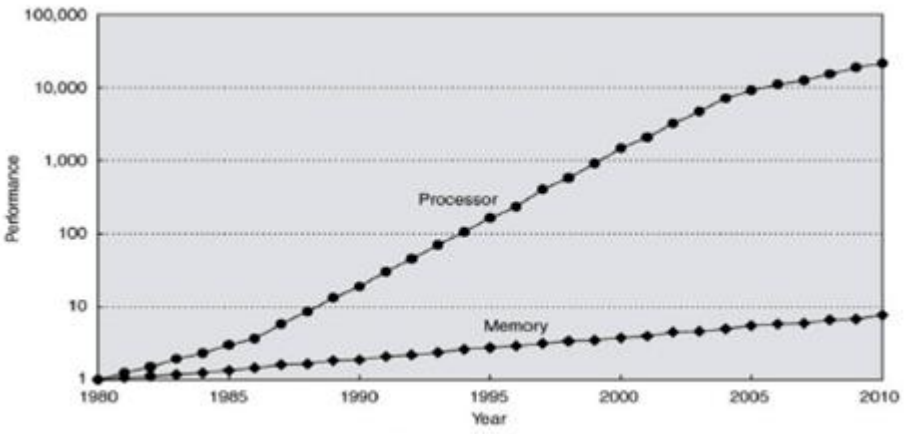
Η έννοια “Cache”, συναντάται για πρώτη φορά στην δεκαετία του 1960. Η Cache χρησιμοποιούταν, λόγω της μεγαλύτερης ταχύτητας της, ως αποθηκευτικός χώρος για την “προνοητική μεταφορά” (prefetching) εντολών ή δεδομένων, ακριβώς πριν αυτά απαιτηθούν από τον επεξεργαστή. Στην εποχή των προσωπικών υπολογιστών (PC), η έννοια Cache, κάνει την επανεμφάνισή της όταν ο επεξεργαστής της Intel 80386 έχει συχνότητα 20MHz και η DRAM μνήμη παρουσιάζει καθυστέρηση (latency) της τάξης των 120ns. Από τον επεξεργαστή Pentium Pro της Intel και ως σήμερα, η cache

είναι ενσωματωμένη στο chip του επεξεργαστή και χρονισμένη σε παρόμοια συχνότητα με αυτόν.

Οι μνήμες ακολουθούσαν τον νόμο του Moore για 20 έτη με ένα τετραπλάσιας χωρητικότητας ολοκληρωμένο κύκλωμα κάθε 3 έτη [2]. Όμως, λόγω της μειωμένης ζήτησης για DRAM's, από το 1998 η χωρητικότητα διπλασιάζεται κάθε δύο έτη, ενώ από το 2006 αυτή η τάση δείχνει να αυξάνει.

Τα παραπάνω σε συνδυασμό με τη ραγδαία αύξηση της συχνότητας λειτουργίας (ταχύτητας – CPU Clock Speed) των επεξεργαστών, είχε ως αποτέλεσμα οι επιδόσεις των επεξεργαστών να αυξάνονται με ρυθμό 60% ετησίως, (Σχήμα 2.1) ενώ οι επιδόσεις των Μνημών με μόλις 7%. Αυτή η διαφορά είναι γνωστή ως "Processor – Memory Performance Gap", ή όπως ονομάστηκε από τους William Wulf και Sally McKee το 1994 στην εργασία Hitting the Memory Wall, το "Επικείμενο Τοίχος Μνήμης" (Memory Wall) [2].

Δεδομένου του χάσματος απόδοσης επεξεργαστή - μνήμης, οι μνήμες επέβαλαν την χαμηλή ταχύτητά τους σε ολόκληρο το υπολογιστικό σύστημα (Memory Bottleneck), οδηγώντας τους επεξεργαστές σε κατάσταση αναμονής (wait state) έως ότου τα νέα δεδομένα να οδηγηθούν στους καταχωρητές τους και έτσι να συνεχιστούν οι υπολογισμοί. Αποτέλεσμα του φαινόμενου αυτού ήταν να μην γίνεται πλήρης εκμετάλλευση της ταχύτητας των επεξεργαστών. Το πρόβλημα επιλύθηκε με την προσθήκη των κρυφών μνημών στην ιεραρχία της μνήμης.



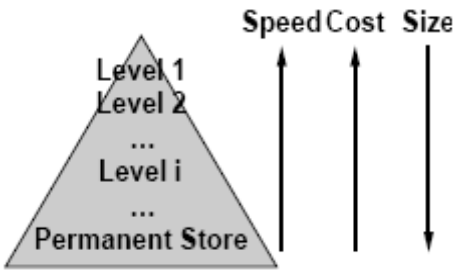
Σχήμα 2.1 CPU – Memory Gap [2]

Με εκκίνηση την απόδοση του 1980, στο Σχήμα 2.1 εμφανίζεται η διαφορά στην απόδοση μεταξύ της μνήμης και του επεξεργαστή μέχρι το 2010. Πρέπει να τονιστεί ότι ο κατακόρυφος άξονας είναι σε λογαριθμική κλίμακα ώστε να αποτυπώσει τη διαφορά στην απόδοση (gap) μεταξύ της μνήμης και του επεξεργαστή. Η μνήμη έχει μέγεθος 64 KB το 1980 με βελτίωση στη καθυστέρηση 1.07 ανά έτος. Ο επεξεργαστής παρουσιάζει βελτίωση στη καθυστέρηση 1.25 ανά έτος μέχρι το 1986 και βελτίωση στη καθυστέρηση 1.52 ανά έτος μέχρι το 2004 καθώς επίσης βελτίωση στη καθυστέρηση 1.20 ανά έτος από εκεί και μετά. [2]

2.2. Ιεραρχία Μνήμης

Όπως είναι γνωστό όλοι οι επεξεργαστές είναι κατασκευασμένοι με τέτοιο τρόπο ώστε να βασίζουν τη λειτουργία τους σε μία μνήμη τυχαίας προσπέλασης. Αν υποθέσουμε ότι δεν υπάρχει ιεραρχία στη μνήμη, δηλαδή η μνήμη τυχαίας προσπέλασης είναι η συνολική μνήμη του συστήματος, τότε η μνήμη αυτή πρέπει να είναι γρήγορη ώστε να ταιριάζει με την ταχύτητα λειτουργίας του επεξεργαστή, να παρουσιάζει τεράστια χωρητικότητα ώστε να εξυπηρετεί τις σύγχρονες ανάγκες αποθήκευσης και με πολύ χαμηλό κόστος κατασκευής [3].

Η ιεραρχία της μνήμης σχεδιάστηκε και εφαρμόστηκε για να παρέχει ταυτόχρονα τις αμοιβαίως αποκλειόμενες υπηρεσίες που προαναφέρθηκαν, βασιζόμενη στην τοπικότητα των αναφορών που αναλύεται παρακάτω. Έτσι με την κατάλληλη επιλογή των επιπέδων της ιεραρχίας, μπορεί να σχεδιαστεί ένα σύστημα που έχει την ταχύτητα του ταχύτερου στοιχείου, κόστος ανά bit ίσο με αυτό του φθηνότερου στοιχείου και κατανάλωση ενέργειας ίση με αυτή του λιγότερο ενεργοβόρου στοιχείου.



Σχήμα 2.2 Ιεραρχία Μνήμης [3]

Στο Σχήμα 2.2 εμφανίζεται η πυραμίδα της ιεραρχίας μνήμης. Τα υψηλότερα επίπεδα στην ιεραρχία (υψηλότερα επίπεδα στην πυραμίδα) παρουσιάζουν μικρότερη δυνατότητα αποθήκευσης, καλύτερη απόδοση, ενώ έχουν μεγαλύτερο κόστος ανά bit σε σχέση με τα χαμηλότερα επίπεδα. Μία σύγχρονη ιεραρχία μνήμης αποτελείται από τα επόμενα στοιχεία, που το καθένα παίζει τον δικό του ειδικό ρόλο στο σύστημα [3].

Cache Memory: Η κατασκευή της κρυφής μνήμης βασίζεται σε κυψέλες τύπου SRAM. Η κρυφή μνήμη παρέχει πρόσβαση στα δεδομένα και τις εντολές του προγράμματος με πολύ μικρή καθυστέρηση και πολύ μεγάλο εύρος ζώνης. Επιπλέον, για κάθε προσπέλαση, η κρυφή μνήμη εμφανίζει σχετικά χαμηλή κατανάλωση ενέργειας, συγκρινόμενη με άλλες τεχνολογίες μνήμης [3].

Random Access Memory: Η κατασκευή της RAM βασίζεται συνήθως σε κυψέλες τύπου DRAM. Η RAM παρέχει δυνατότητα αποθήκευσης, η οποία είναι σχετικά μεγάλη, γρήγορή και φθηνή. Είναι μεγάλη και φθηνή συγκρινόμενη με την κρυφή μνήμη , ενώ είναι γρήγορη συγκρινόμενη με την μονάδα δίσκου [3].

Μονάδα Σκληρού Δίσκου: Η μονάδα σκληρού δίσκου (Hard Disk Drive) βασίζει την λειτουργία της σε Μαγνητικούς Δίσκους, στους οποίους αποθηκεύονται και ανακτώνται τα δεδομένα με τη βοήθεια ενός ηλεκτρομηχανικού συστήματος (κεφαλές), καθώς επίσης και σε περιφερειακή κυκλωμάτωση. Η HDD παρέχει μόνιμη αποθήκευση με πάρα πολύ χαμηλό κόστος ανά bit, ενώ εμφανίζει πολύ μεγάλο χρόνο πρόσβασης (τάξης millisecond) σε σχέση τις cache και την κύρια μνήμη [3].

2.3. Αρχή Λειτουργίας Cache

Η λειτουργία της κρυφής μνήμης, βασίζεται σε μία κοινή ιδιότητα των προγραμμάτων, που καλείται τοπικότητα των αναφορών (locality of references). Σύμφωνα με την τοπικότητα των αναφορών, [3] η πληροφορία (εντολές και δεδομένα) που χρησιμοποιήθηκε πρόσφατα είναι πιθανόν να ξαναχρησιμοποιηθεί στο άμεσο μέλλον και η πληροφορία που βρίσκεται κοντά στην πληροφορία που χρησιμοποιείται τώρα είναι πιθανόν να χρησιμοποιηθεί στο άμεσο μέλλον. Η ιδιότητα αυτή ονομάστηκε με τον όρο «τοπικότητα», διότι έχει παρατηρηθεί ότι οι αναφορές

των προγραμμάτων στη μνήμη τείνουν να γίνονται τοπικά όσον αφορά στον χρόνο και στον χώρο.

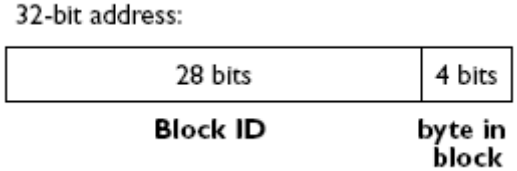
Έτσι αν ένα πρόγραμμα αναφερθεί σε μία διεύθυνση μνήμης, τότε είναι πιθανό πολύ σύντομα να αναφερθεί ξανά στην ίδια διεύθυνση (Χρονική Τοπικότητα / Temporal Locality), ενώ αν ένα πρόγραμμα αναφερθεί σε μία διεύθυνση μνήμης, τότε είναι πιθανό πολύ σύντομα να αναφερθεί και σε πλησίον αυτής διευθύνσεις μνήμης (Χωρική Τοπικότητα / Spatial Locality) [3].

2.4. Λειτουργία και Οργάνωση Cache

2.4.1. Block ID

Η κρυφή μνήμη αποθηκεύει τμήματα δεδομένων, που προέρχονται από κάποια μνήμη χαμηλότερου επίπεδου στην ιεραρχία (πχ: κύρια μνήμη), στη δομική της μονάδα που ονομάζεται Cache Block ή Cache Line [3]. Έτσι όταν ο επεξεργαστής ζητήσει να διαβάσει μια λέξη, τότε το Block της κύριας μνήμης που την περιέχει μεταφέρεται ολόκληρο σε μια Cache Line.

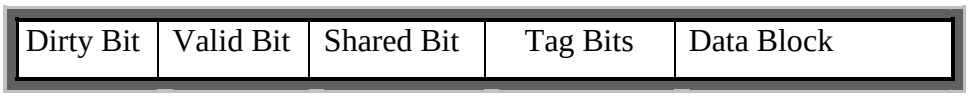
Αποτέλεσμα αυτής της βασισμένης σε cache lines αποθήκευσης στην κρυφή μνήμη, είναι ο νοητός τεμαχισμός της μνήμης από την οποία προέρχονται τα δεδομένα σε τμήματα μεγέθους ίσου με αυτό της cache line. Το φαινόμενο αυτό, μας επιτρέπει την αναγνώριση ενός συγκεκριμένου cache line χρησιμοποιώντας ολόκληρο ή τμήμα του block id. Το block id αποτελεί τμήμα της διεύθυνσης των δεδομένων στη κύρια μνήμη.



Σχήμα 2.3 32 bit διεύθυνσης

Στο Σχήμα 2.3, απεικονίζεται μία διεύθυνση των 32 bit διαχωρισμένη σε δύο τμήματα: το block id (28 bits) και το byte in block (4 bits). Ο αριθμός των bits στο “byte in block” φανερώνει τη διεύθυνση της λέξης στο block της κύριας μνήμης. [3]

Στο Σχήμα 2.4 φαίνεται η δομή μίας cache line, η οποία αποτελείται από τα Control Bits (Dirty, Valid, Shared), τα Tag Bits, και το Data Block.



Σχήμα 2.4 Δομή cache line

2.4.2. Data Block, Control Bits

Επιγραμματικά περιγράφονται τα Data block και Control bits.

Data block ονομάζεται το block των δεδομένων ή των εντολών που αντιγράφονται από την κύρια στην κρυφή μνήμη. Τα Control bits περιλαμβάνουν τα:

Dirty bit: Όταν χρησιμοποιείται Write-Back τεχνική, τα νέα δεδομένα που αποθηκεύονται στην κρυφή μνήμη, θα αποθηκευτούν και στην κύρια μνήμη μόνο όταν πρόκειται να αντικατασταθεί η συγκεκριμένη γραμμή από κάποια άλλη. Έτσι η κύρια μνήμη δεν είναι πλήρως ενημερωμένη σε κάθε χρονική στιγμή. Αν το Dirty Bit έχει τιμή "1" τότε τα περιεχόμενα της γραμμής έχουν μεταβληθεί και επομένως η γραμμή θα πρέπει να αντιγραφεί στην κύρια μνήμη, ενώ αν έχει τιμή "0", τότε η γραμμή απλώς αντικαθίσταται.

Valid bit: Ονομάζεται το bit που χρησιμοποιείται για να δείξει ότι η γραμμή έχει ή όχι έγκυρα δεδομένα.

Shared Bit: Ονομάζεται αυτό που χρησιμοποιείται για να δείξει αν υπάρχουν αντίγραφα αυτής της Γραμμής και σε άλλες κρυφές μνήμες σε ένα πολυεπεξεργαστικό σύστημα με κοινόχρηστη μνήμη.

2.4.3. Cache Tag (Tag bits)

Εκτός των data block και control bits, η cache line αποτελείται και από την cache tag. Δεδομένου ότι η κρυφή μνήμη διαθέτει πολύ μικρότερο μέγεθος από την μνήμη που βρίσκεται ακριβώς πριν από αυτή στην ιεραρχία (πχ: κύρια μνήμη), δηλαδή τα cache lines είναι πολύ λιγότερα από τα blocks της μνήμης, υπάρχει πολύ μεγάλη πιθανότητα δεδομένα που θα ζητήσει ο επεξεργαστής από την κρυφή μνήμη, να μην υπάρχουν σε αυτή [3]. Η κατάδειξη της ύπαρξης ή μη των δεδομένων στην κρυφή μνήμη, γίνεται μέσω των Tag Bits, ή όπως αλλιώς συνηθίζεται να ονομάζονται Cache Tag. Το Cache Tag αποτελείται από ολόκληρο ή μέρος του block id, κάτι το οποίο εξαρτάται από τον τρόπο αντιστοίχισης του block της κύριας μνήμης στις cache lines (Memory Mapping).

2.4.4. Memory Mapping

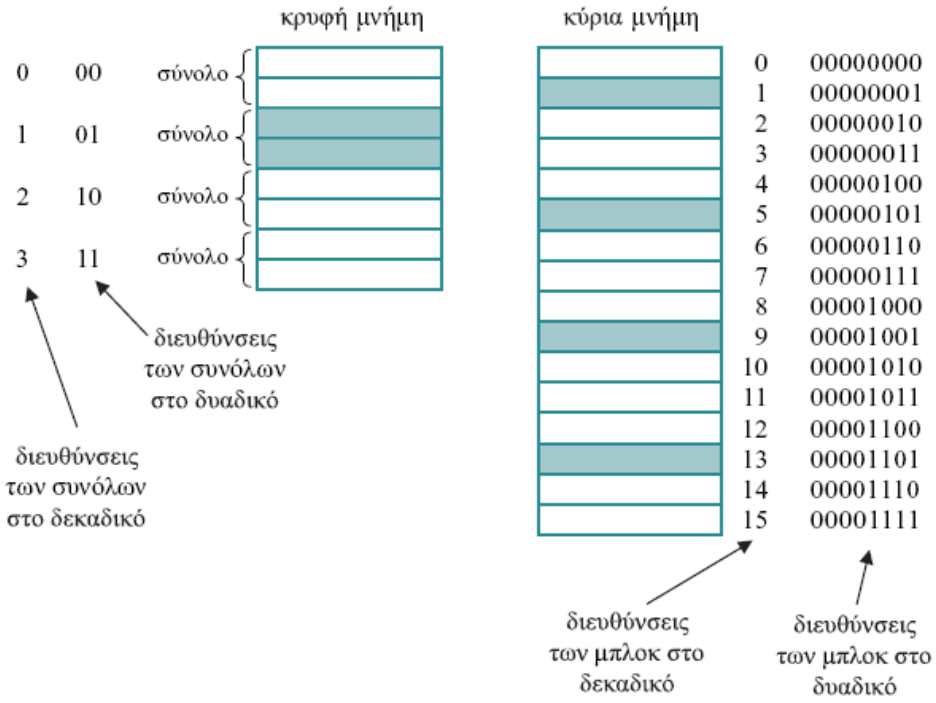
Το Memory Mapping μέσω των Mapping Functions (set associative, direct mapped, fully associative), καθορίζει τον τρόπο με τον οποίο αντιστοιχίζεται το Block της κύριας μνήμης που περιέχει την λέξη που ζήτησε ο επεξεργαστής σε μια συγκεκριμένη Cache Line.

Η κρυφή μνήμη με οργάνωση **τ** – τρόπων συνόλου συσχέτισης (set associative) αποτελείται από ομάδες των τ cache lines, που καλούνται σύνολα (sets). Κάθε block της κυρίας μνήμης μπορεί να τοποθετηθεί σε οποιαδήποτε cache line ενός συγκεκριμένου set στο οποίο αντιστοιχεί μία διεύθυνση (έστω C). Αν A η διεύθυνση του block της κύριας μνήμης και J το πλήθος των sets της κρυφής μνήμης, τότε η διεύθυνση του set δίδεται από το υπόλοιπο της διαίρεσης του A δια του J.

$$C = A \bmod J$$

Στο Σχήμα 2.5 δίνεται μία κρυφή μνήμη τεσσάρων sets, όπου κάθε ένα από αυτά αποτελείται από δύο cache lines, και μία κύρια μνήμη των δεκαέξι blocks. Είναι εμφανές πως στο set με διεύθυνση ένα (01_b) μπορούν να βρίσκονται το πολύ δύο από τα block της κυρίας μνήμης, που το υπόλοιπο της διαίρεσης της διεύθυνσής τους δια

του τέσσερα ισούται με ένα (01_b), δηλαδή δύο από τα μπλοκ με διευθύνσεις 1, 5, 9 και 13 [1]. Γενικεύοντας, σε ένα set της κρυφής μνήμης με μία συγκεκριμένη διεύθυνση μπορούν να βρίσκονται κάθε χρονική στιγμή, μόνο δύο από τα block της κύριας μνήμης των οποίων το υπόλοιπο της διαίρεσης της διεύθυνσής των δια του πλήθους των set ισούται με τη διεύθυνση του set [1].



Σχήμα 2.5 Κρυφή Μνήμη με Οργάνωση T – Τρόπων Συνόλου Συσχέτισης [1]

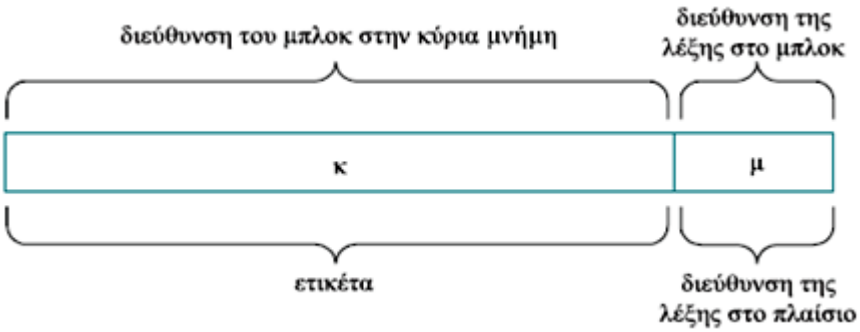
Να σημειωθεί ότι ακραίες τιμές του "**τ**" ορίζουν διαφορετικά είδη mapping. Έτσι αν το τ = 1 η κρυφή μνήμη με οργάνωση τ–τρόπων συνόλου συσχέτισης, έχει ένα μία cache line ανά set, επομένως είναι άμεσης οργάνωσης (direct mapped). Στην περίπτωση της κρυφής μνήμης άμεσης οργάνωσης κάθε block της κύριας μνήμης μπορεί να τοποθετηθεί σε μία συγκεκριμένη cache line.

Αντίστοιχα αν ο αριθμός των cache lines ανά set (**τ**) γίνει ίσος με τον συνολικό αριθμό των cache lines (s), δηλαδή τ = s, τότε η κρυφή μνήμη έχει οργάνωση πλήρους συσχέτισης [1]. Στην περίπτωση κρυφής μνήμης με οργάνωση πλήρους συσχέτισης (fully associative), κάθε block της κύριας μνήμης μπορεί να τοποθετηθεί σε οποιαδήποτε cache line της κρυφής μνήμης.

Σε μία κρυφή μνήμη πλήρους συσχέτισης, η υλοποίηση καθώς επίσης και ο τρόπος διευθυνσιοδότησης διαφέρουν σημαντικά σε σχέση με την υλοποίηση και τον τρόπο διευθυνσιοδότησης της κρυφής μνήμης με οργάνωση τ – τρόπων συνόλου συσχέτισης και με άμεσης οργάνωσης.

Η κρυφή μνήμη πλήρους συσχέτισης, αποτελείται από μία συσχετιστική μνήμη (CAM) για το τμήμα ετικετών, καθώς και από μία SRAM μνήμη για το data τμήμα. Σε κάθε θέση της συσχετιστικής μνήμης αποθηκεύονται ένα δυαδικό ψηφίο εγκυρότητας και τα δυαδικά ψηφία της ετικέτας. Τα δυαδικά ψηφία της πληροφορίας του block (Data Block), αποθηκεύονται στο SRAM τμήμα.

Έστω πως το μέγεθος της κύριας μνήμης είναι $2^{\nu+\mu}$ λέξεις (2^{ν} blocks, με 2^{μ} λέξεις / block). Έστω ακόμη πως το μέγεθος της κρυφής μνήμης είναι $2^{\kappa+\mu}$ λέξεις (2^{κ} cache lines, και 2^{μ} λέξεις / cache line). Το μέγεθος της κρυφής μνήμης είναι μικρότερο από το μέγεθος της κύριας μνήμης, δηλαδή $\kappa < \nu$. Στην περίπτωση αυτή κάθε διεύθυνση που παράγει ο επεξεργαστής αποτελείται από δύο πεδία όπως φαίνεται στο Σχήμα 2.6 [1].



Σχήμα 2.6 Ερμηνεία της Διεύθυνσης Μνήμης (fully associative) [1]

Όταν ο επεξεργαστής παράγει μία διεύθυνση, τα δυαδικά ψηφία του πεδίου ετικέτα οδηγούνται στη συσχετιστική μνήμη και συγκρίνονται ταυτόχρονα (παράλληλα, όπως αναλύεται παρακάτω) με όλες τις ετικέτες που είναι αποθηκευμένες στη συσχετιστική μνήμη. Εάν το πεδίο “ετικέτα” της διεύθυνσης που παρήγαγε ο επεξεργαστής είναι

ίδιο με κάποια από τις αποθηκευμένες ετικέτες και το αντίστοιχο δυαδικό ψηφίο εγκυρότητας έχει τη λογική τιμή ένα, τότε η έξοδος "επιτυχία" (hit) παίρνει τη λογική τιμή ένα.

Εάν η γραμμή εξόδου επιτυχία/αποτυχία έχει την τιμή μηδέν (miss), τότε τα δεδομένα στην έξοδο της κρυφής μνήμης αγνοούνται και το block της κύριας μνήμης, που περιέχει τη ζητούμενη πληροφορία, προσκομίζεται από την κύρια μνήμη στην κρυφή μνήμη.

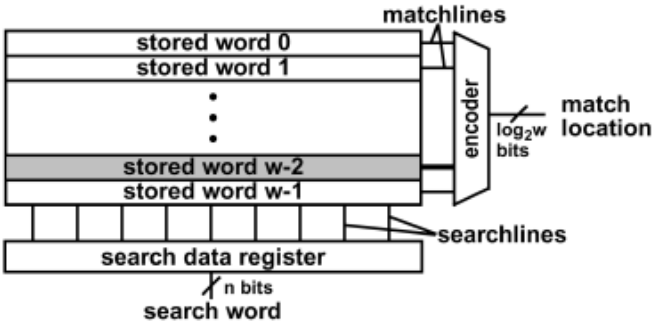
2.5. Content Addressable Memory – CAM

Η Content Addressable Memory (CAM), όπως και οι κοινές μνήμες SRAM, παρέχει την δυνατότητα για εγγραφή, ανάγνωση και αναζήτηση δεδομένων. Η μέθοδος με την οποία γίνεται το τελευταίο, είναι αυτή που κάνει τη βασική διαφορά. Η CAM έχει τη δυνατότητα να κάνει σύγκριση ζητούμενων με αποθηκευμένα bit κατά τη διάρκεια μίας **παράλληλης αναζήτησης**. Συγκεκριμένα η CAM συγκρίνει τα προς αναζήτηση δεδομένα, που υπάρχουν στην είσοδό της, με ένα πίνακα με αποθηκευμένα δεδομένα επιστρέφοντας, στη γενική περίπτωση εφαρμογής, τη διεύθυνση αυτών που ταιριάζουν ή, σε εξειδικευμένες εφαρμογές, ένα σήμα ευστοχίας (hit) ανά γραμμή. Είτε υλοποιηθεί με SRAM cell, είτε με DRAM cell, η CAM έχει ρυθμαπόδοση ενός κύκλου, καθιστώντας τη ταχύτερη από κάθε σύστημα αναζήτησης είτε βασισμένο σε υλικό, είτε σε λογισμικό. Η χρήση της CAM συναντάται σε πολλές σύγχρονες εφαρμογές που απαιτούν υψηλή ταχύτητα αναζήτησης. Τέτοιες εφαρμογές συμπεριλαμβάνουν μετασχηματισμούς Hough, κωδικοποίηση Huffman, Lempel–Ziv συμπίεση και κωδικοποίηση εικόνας. Η βασική εμπορική εφαρμογή της CAM είναι στους δρομολογητές δικτύου (network routers), καθώς χρησιμοποιείται για την ταξινόμηση και την προώθηση IP (internet protocol) πακέτων [5]. Επιπλέον η ταχύτατη αναζήτηση που παρέχει η CAM αποδεικνύεται ιδιαιτέρως χρήσιμη στην υποστήριξη της ποιότητας των υπηρεσιών (Quality of Service - QoS) που απαιτείται στις εφαρμογές πραγματικού χρόνου (real time) όπως η μετάδοση πολυμεσικών δεδομένων [6].

2.5.1. Αρχή Λειτουργίας CAM

Όπως φαίνεται στο παρακάτω απλοποιημένο διάγραμμα μίας CAM (Σχήμα 2.7), η είσοδος στο σύστημα είναι η λέξη προς αναζήτηση (search word), η οποία μεταδίδεται μέσω των γραμμών αναζήτησης (search-lines) στον πίνακα (array) με τα αποθηκευμένα δεδομένα. Σε κάθε αποθηκευμένη λέξη υπάρχει μία γραμμή ταυτοποίησης (match-line), η οποία καταδεικνύει εάν η search word και η αποθηκευμένη λέξη ταυτίζονται (περίπτωση match) ή είναι διαφορετικές (περίπτωση mismatch, ή miss). Οι matchlines οδηγούν έναν κωδικοποιητή (encoder), ο οποίος δημιουργεί ένα σήμα που δείχνει την θέση της matchline στην οποία υπήρξε ταυτοποίηση.

Σε εφαρμογές CAM όπου μπορεί να προκύψουν ταυτοποιήσεις σε περισσότερες από μία match-lines, χρησιμοποιείται επιπλέον ένας κωδικοποιητής προτεραιότητας , ο οποίος επιλέγει για την έξοδό του την θέση με την μεγαλύτερη προτεραιότητα [5]. Όταν η CAM χρησιμοποιείται σε εφαρμογή κρυφής μνήμης, για να αποθηκεύσει τα δεδομένα ετικέτας, ο κωδικοποιητής (encoder στο Σχήμα 2.7) δεν χρειάζεται: οι matchlines οδηγούν απευθείας τις αντίστοιχες γραμμές λέξης (word-lines) της SRAM που περιέχει τα δεδομένα των γραμμών κρυφής μνήμης.

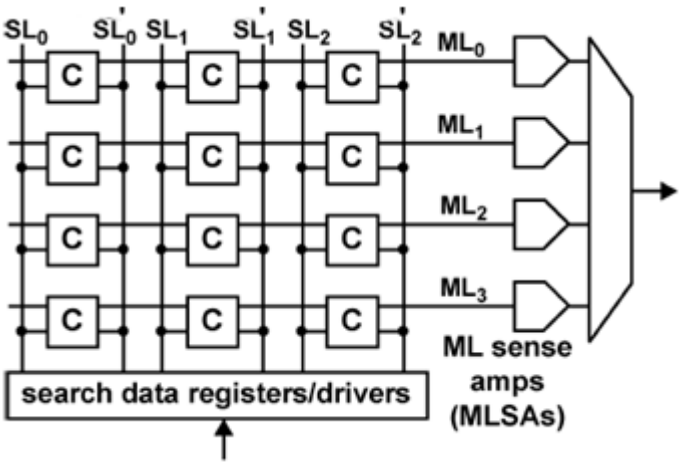


Σχήμα 2.7 Απλοποιημένο διάγραμμα CAM που περιέχει w λέξεις [5].

Στο Σχήμα 2.8 φαίνεται , σε μεγαλύτερη λεπτομέρεια, το απλοποιημένο διάγραμμα της CAM που παρουσιάστηκε παραπάνω. Η εικονιζόμενη CAM αποτελείται από 4 λέξεις, με κάθε λέξη να αποτελείται από 3 bits. Τα 3 bits της λέξης είναι οργανωμένα οριζόντια και αντιστοιχούν σε 3 CAM cells. Για κάθε αποθηκευμένη λέξη υπάρχει

μία matchline τύπου NOR ή NAND (ML0, ML1, ML2, κτλ) , η οποία οδηγεί έναν matchline sense amplifier (MLSA), ενώ για κάθε bit της searchword υπάρχει ένα ζευγάρι διαφορικών searchlines (SL0, SL0', SL1, SL1', κτλ).

Η λειτουργία αναζήτησης στην CAM ξεκινά με τη φόρτωση των δεδομένων αναζήτησης στους καταχωρητές αναζήτησης (search registers ή data drivers) και την ταυτόχρονη προφόρτιση των matchlines στο λογικό ένα, για τον πιο διαδεδομένο τύπο που είναι η NOR (η NAND περίπτωση εξετάζεται λεπτομερώς στη συνέχεια), θέτοντας τες προσωρινά σε κατάσταση match. Στη συνέχεια οι search registers μεταδίδουν τα δεδομένα αναζήτησης στις searchlines και το κάθε CAM cell συγκρίνει το bit που είναι αποθηκευμένο σε αυτό, με την τιμή της αντίστοιχης searchline. Να σημειωθεί ότι ο search driver θέτει την τιμή του κάθε bit που είναι αποθηκευμένο στη κατάλληλη searchline (SL) και τη συμπληρωματική τιμή αυτού του bit στην συμπληρωματική serchline (SL') του ζεύγους. Οι NOR matchlines των οποίων όλα τα bits κάνουν match παραμένουν σε κατάσταση λογικού ένα, ενώ αντίθετα αυτές που έχουν έστω και ένα bit να κάνει miss αποφορτίζονται εμφανίζοντας λογικό μηδέν. Ο MLSA ανιχνεύει την κατάσταση της matchline και ο κωδικοποιητής κωδικοποιεί σε διεύθυνση την θέση matchline που έκανε match.

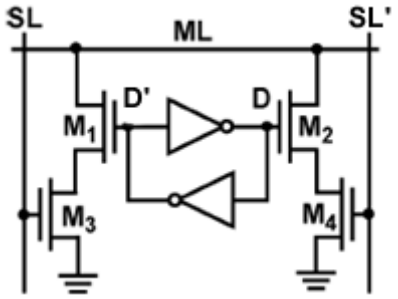


Σχήμα 2.8 Απλοποιημένο διάγραμμα CAM που περιέχει w λέξεις με 3 cells / λέξη.

Δεδομένου ότι συνήθως η υλοποίηση του CAM cell βασίζεται σε SRAM, εφεξής η αναφορά στο cell να υπονοεί υλοποίηση με SRAM όπου η αποθήκευση ενός bit γίνεται με διασταυρωμένους αντιστροφείς (cross coupled inverters).

2.5.2. NOR Match Line CAM Cell

Στο Σχήμα 2.9 παρουσιάζεται ένα τύπου NOR Match Line CAM Cell, όπου η αποθήκευση του bit υλοποιείται με SRAM, στο οποίο cross coupled inverters αποθηκεύουν το bit εμφανίζοντάς το στον κόμβο D και το συμπληρωματικό του στον D’ [5].



Σχήμα 2.9 10-T NOR CAM Cell [5]

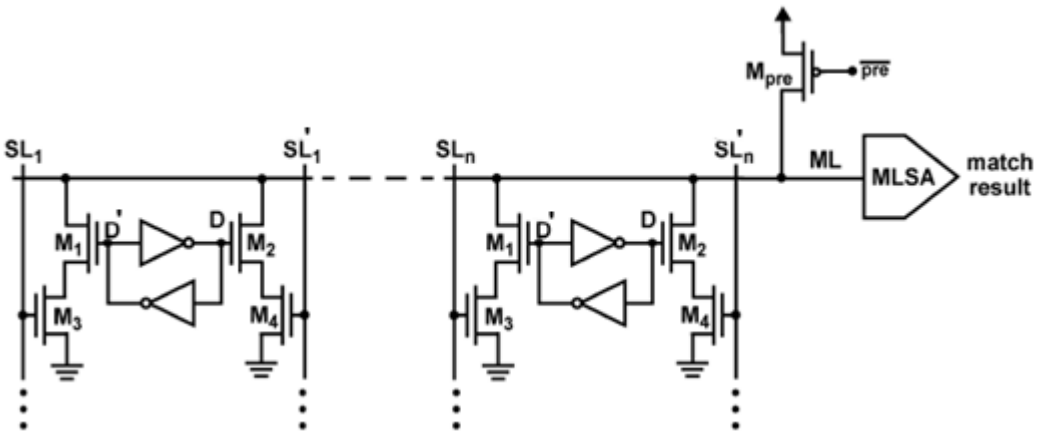
(Οι γραμμές που χρησιμοποιούνται για την εγγραφή και την ανάγνωση – bitlines - καθώς και τα transistor που τις «συνδέουν» με το cell παραλείπονται για λόγους ευκρίνειας)

Η σύγκριση μεταξύ των συμπληρωματικών bit στους κόμβους D και D’ και των συμπληρωματικών bit στις συμπληρωματικές searchlines SL και SL’ επιτυγχάνεται με τη χρήση των transistors M1, M2, M3 και M4. Τα transistor ανά ζεύγη (M1,M3) και (M2,M4) δημιουργούν δύο μονοπάτια αποφόρτισης (pulldown paths), συνδέοντας το καθένα την ML με τη γείωση. Το κάθε μονοπάτι αποφόρτισης υλοποιεί μία XOR λογική (XOR: Δίνει στην έξοδό λογικό ένα, μόνο όταν οι είσοδοι έχουν μεταξύ τους διαφορετικές λογικές τιμές) με εισόδους τις SL, D και SL’, D’ αντίστοιχα.

Έτσι όταν το προς αναζήτηση bit έχει την τιμή 1 (SL=1, SL’=0) και στους cross coupled inverters είναι αποθηκευμένο λογικό ένα (D=1, D’=0), γίνεται match και κανένα από τα δύο μονοπάτια δεν ενεργοποιείται (M1:off, M3:on, M2:on, M4:off). Αντίστοιχα αν (SL=0, SL’=1) και στους cross coupled inverters είναι αποθηκευμένο λογικό ένα (D=0, D’=1), γίνεται match και όπως και πριν κανένα από τα δύο μονοπάτια δεν ενεργοποιείται (M1:on, M3:off, M2:off, M4:on).

Αντίθετα αν (SL=1, SL'=0) και στους cross coupled inverters είναι αποθηκευμένο λογικό μηδέν (D=0, D'=1), γίνεται miss και το μονοπάτι αποφόρτισης που δημιουργούν τα M1, M3 (M1:on, M3:on) εξισώνει το δυναμικό της ML με αυτό της γείωσης. Το ίδιο συμβαίνει και στην περίπτωση όπου SL=0, SL'=1) και στους cross coupled inverters είναι αποθηκευμένο λογικό ένα (D=1, D'=0).

Η NOR λογική του cell (NOR: Δίνει στην έξοδό λογικό ένα, μόνο όταν όλες οι είσοδοι βρίσκονται σε λογικό μηδέν) γίνεται εμφανής όταν πολλαπλά cells συνδέονται παράλληλα, σχηματίζοντας μία λέξη της CAM (Σχήμα 2.10). Η παράλληλη διάταξη διαμορφώνεται από τα πολλαπλά μονοπάτια αποφόρτισης που είναι συνδεδεμένα σε μία κοινή matchline (ML). Η matchline της λέξης βρίσκεται σε κατάσταση match μόνο όταν κάθε ένα cell στη λέξη έχει κάνει match, δηλαδή όταν κάθε μονοπάτι αποφόρτισης είναι αποκομμένο [5]. Στο Σχήμα 2.10 εμφανίζεται η δομή μίας NOR matchline αποτελούμενη από n cells. Το transistor Mpre προφορτίζει τη matchline και ο MLSA αξιολογεί την κατάσταση της matchline, δημιουργώντας το κατάλληλο σήμα ταυτοποίησης.



Σχήμα 2.10 NOR matchline δομή [5]

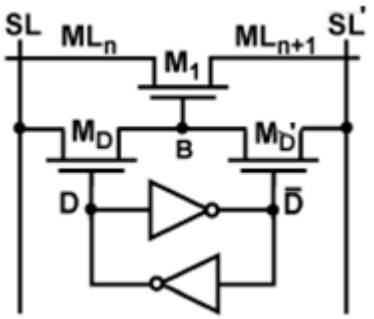
Ένας τυπικός NOR κύκλος αναζήτησης (search cycle) ολοκληρώνεται σε τρεις φάσεις, την προφόρτιση των searchline, την προφόρτιση των matchline και την φάση υπολογισμού (evaluation) της matchline. Αρχικά οι seachlines προφορτίζονται σε δυναμικό ίσο με αυτό της γείωσης (λογικό μηδέν), με αποτέλεσμα την

απενεργοποίηση των μονοπατιών αποφόρτισης, αποσυνδέοντας με αυτόν τον τρόπο την matchline από τη γείωση. Στη συνέχεια η matchline προφορτίζεται σε υψηλό δυναμικό (λογικό ένα) μέσω του transistor Mpre. Τέλος οι searchlines οδηγούνται από τον search driver με τα ζητούμενα bits, εγείροντας τη διαδικασία αξιολόγησης της matchline από τον MLSA. Στην περίπτωση ενός match η τάση της ML παραμένει σε υψηλό δυναμικό και τα μονοπάτια αποφόρτισης είναι ανενεργά. Στην περίπτωση ενός miss υπάρχει τουλάχιστον ένα μονοπάτι αποφόρτισης ενεργό, μέσω του οποίου η matchline αποφορτίζεται. Ο προαιρετικός matchline sense amplifier (MLSA) αξιολογεί την τάση της ML και δημιουργεί ένα match σήμα στην έξοδό του [5].

Το κύριο χαρακτηριστικό της NOR matchline είναι η υψηλή ταχύτητα λειτουργίας. Η πιο αργή περίπτωση εμφανίζεται όταν υπάρχει miss σε ένα μόνο bit της λέξης, αναγκάζοντας την εκφόρτιση ολόκληρης της matchline μέσα από τα δύο σε σειρά συνδεδεμένα transistor του ενεργού μονοπατιού αποφόρτισης.

2.5.3. NAND Match Line CAM Cell

Στο Σχήμα 2.11 παρουσιάζεται ένα τύπου NAND Match Line CAM Cell, όπου η αποθήκευση του bit υλοποιείται με SRAM, στο οποίο cross coupled inverters αποθηκεύουν το bit εμφανίζοντάς το στον κόμβο D και το συμπληρωματικό του στον D’ [5]. Οι γραμμές που χρησιμοποιούνται για την εγγραφή και την ανάγνωση – bitlines - καθώς και τα transistor που τις «συνδέουν» με το cell παραλείπονται για λόγους ευκρίνειας.

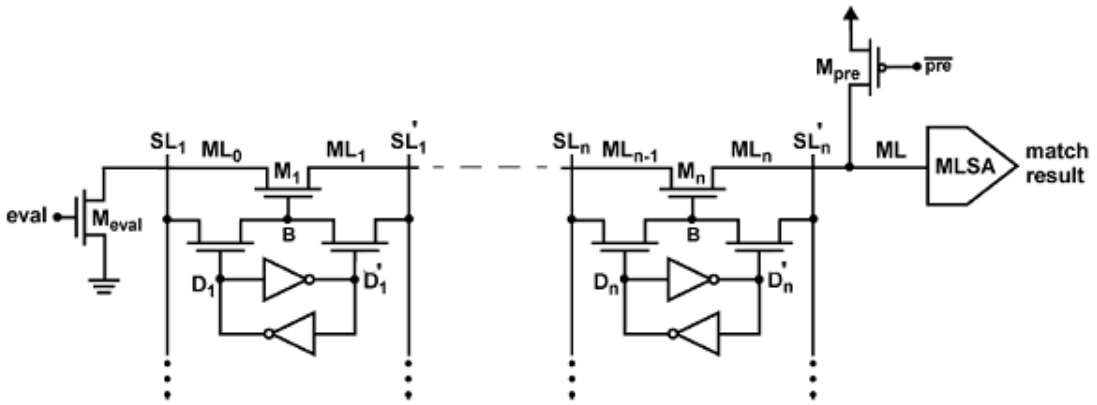


Σχήμα 2.11 9-T NAND CAM Cell [5]

Η σύγκριση μεταξύ των συμπληρωματικών bit στους κόμβους D και D’ και αυτών στις συμπληρωματικές searchlines SL και SL’ επιτυγχάνεται με τη χρήση του transistor M1 και των pass transistors MD και MD’.

Όταν το προς αναζήτηση bit έχει την τιμή 1 (SL=1, SL’=0) και στους cross coupled inverters είναι αποθηκευμένο λογικό ένα (D=1, D’=0), γίνεται match. Σε αυτή την περίπτωση το MD άγει και περνάει το λογικό ένα από την SL στον κόμβο B. Το λογικό ένα στον κόμβο B θέτει το transistor M1 σε αγώγιμη κατάσταση (on) (το ίδιο συμβαίνει όταν SL=0, SL’=1 και D=0, D’=1). Στις περιπτώσεις όπου η τιμή της SL είναι διαφορετική από αυτή του D ή η τιμή της SL’ είναι διαφορετική από αυτή του D’, γίνεται miss, με αποτέλεσμα η τιμή του κόμβου B να είναι σε λογικό μηδέν και το transistor M1 να είναι κλειστό. Συνεπώς ο κόμβος B υλοποιεί μία XNOR λογική με εισόδους τις SL, D: δίνει στην έξοδό λογικό ένα, μόνο όταν οι όλες οι είσοδοι έχουν μεταξύ τους όμοιες λογικές τιμές.

Η NAND λογική (δίνει στην έξοδό λογικό μηδέν, μόνο όταν όλες οι είσοδοι βρίσκονται σε λογικό ένα) της CAM γίνεται εμφανής όταν πολλαπλά cells συνδέονται σε σειρά σχηματίζοντας μία CAM λέξη (Σχήμα 2.12: Δομή μίας NAND matchline αποτελούμενη από n cells. Το transistor Mpre προφορτίζει τη matchline και ο MLSA αξιολογεί την κατάσταση της matchline, δημιουργώντας το match). Η σειριακή σύνδεση, αναφέρεται σε transistors όπως το M1 συνδεδεμένα το ένα μετά το άλλο, διαμορφώνοντας μία αλυσίδα NMOS. Η αλυσίδα αυτή των transistors αποτελεί το μονοπάτι αποφόρτισης (pulldown path), το οποίο ενεργοποιείται μόνο όταν τα transistor που το συνιστούν είναι σε αγώγιμη κατάσταση δηλαδή όλα τα cell της λέξης είναι σε κατάσταση match [5].



Σχήμα 2.12 NAND matchline δομή [5]

Ένας τυπικός NAND κύκλος αναζήτησης (search cycle) ξεκινάει την πρώτη του φάση με το transistor προφόρτισης Mpre να προφορτίζει τον κόμβο ML της Match Line θέτοντας την τάση του ίση με αυτή της τροφοδοσίας Vdd, ενώ οι SL βρίσκονται σε μηδενικό δυναμικό και το Meval δεν άγει. Στη δεύτερη φάση το Mpre παύει να είναι αγώγιμο, οδηγούνται οι Search Lines με τα προς αναζήτηση δεδομένα και το transistor αξιολόγησης Meval γίνεται αγώγιμο. Στην περίπτωση ενός match όλα τα NMOS transistor M1 έως και το Mn άγουν, δημιουργώντας ένα μονοπάτι από τον κόμβο ML προς τη γείωση, εκφορτίζοντας τον κόμβο ML, ενώ στην περίπτωση ενός miss τουλάχιστον ένα από τα NMOS transistor M1 έως το Mn είναι σε κατάσταση off, διατηρώντας την τάση του κόμβου ML στην τιμή Vdd. Τέλος ο MLSA εντοπίζει

το υψηλό (miss) ή χαμηλό (match) δυναμικό της matchline, δημιουργώντας κατάλληλο σήμα στην έξοδό του [5].

Ένα πλεονέκτημα της NAND matchline σε σχέση με τη NOR είναι ότι στη NAND, η διάδοση του σήματος σταματάει στην περίπτωση ενός miss, με αποτέλεσμα να μη υπάρχει κατανάλωση ενέργειας περά από το τελευταίο σε αγώγιμη κατάσταση transistor της NMOS αλυσίδας. Τυπικά, μόνο μία matchline ταιριάζει, πράγμα που σημαίνει πως σε κάθε μία από τις υπόλοιπες matchlines του array, μόνο ένας μικρός αριθμός από transistors άγει. Για τον λόγο αυτό η κατανάλωση ενέργειας παραμένει σε χαμηλά επίπεδα. [5].

2.5.4. Mixed NOR (parallel) – NAND (serial) Match Line Cell

Η αντιπαράθεση μεταξύ των υποστηρικτών της παράλληλης (NOR) και αυτών που υποστηρίζουν τη σειριακή (NAND) διασύνδεση των cells, έχει οδηγήσει σε δομές που εμπεριέχουν και τις δύο οργανώσεις. Στο [7] προτείνεται μία προσαρμοζόμενη serial-parallel CAM (SPCAM), η οποία εκμεταλλεύεται την μη συχνή μεταβολή των περισσότερο σημαντικών ψηφίων (MSB – most significant bit) των ετικετών. Οι σχεδιαστικές αρχές που ακολουθούνται είναι επιγραμματικά οι εξής:

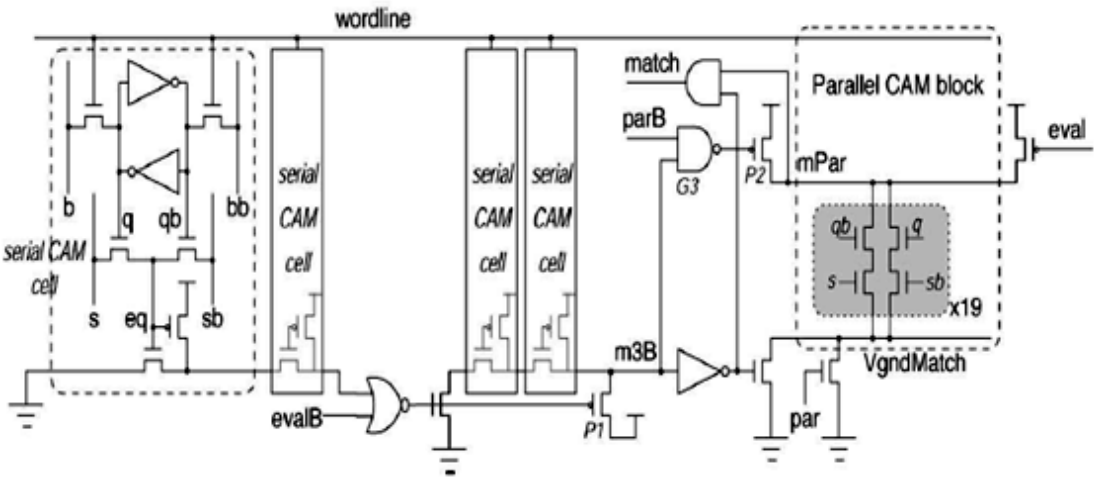
- Ελαχιστοποίηση των μεταβάσεων των matchlines
- Χρήση ξεχωριστών γραμμών για τις search και τις bit lines
- Η μη εξώθηση των searchlines σε λογικό μηδέν ή λογικό ένα κατά τη διάρκεια της φόρτισης της matchline.
- Η ελαχιστοποίηση της χρήσης σημάτων χρονισμού.

Η SPCAM μπορεί να λειτουργήσει είτε με παράλληλο, είτε με σειριακό τρόπο (Σχήμα 2.13). Στην παράλληλη λειτουργία χρησιμοποιούνται τυπικά 10-transistor cells όπου η σύνδεση με τη γείωση των δύο pulldown NMOS αλυσίδων (που κάνουν το τεστ ισοδυναμίας) να έχει αντικατασταθεί με VgndMatch. Το σήμα αυτό κινείται κατά μήκος της λέξης, διασυνδέοντας όλα τα παράλληλα cells.

Τα τέσσερα bits των σειριακών cells διαχωρίζονται σε δύο ομάδες των δύο, με σκοπό τον περιορισμό του μέγιστου αριθμού των transistor σε σειρά σε τρία. Στα match σήματα των δύο ομάδων, εφαρμόζεται λογικό AND για την εξαγωγή του τελικού αποτελέσματος.

Κατά τη σειριακή λειτουργία, τα τέσσερα λιγότερο σημαντικά ψηφία (LSB – least significant bit) της κάθε γραμμής ελέγχονται με σειριακό τρόπο και αν όλα κάνουν match, τότε τα υπόλοιπα bits ελέγχονται με παράλληλο τρόπο "τραβώντας" το VgndMatch σε δυναμικό της γείωσης.

Κατά την παράλληλη λειτουργία, και τα δύο τμήματα ελέγχονται παράλληλα. Τα τέσσερα LSBs ελέγχονται σειριακά, λειτουργία που επικαλύπτεται από τον παράλληλο έλεγχο των υπολοίπων. Με τον τρόπο αυτό η εξαγωγή του τελικού αποτελέσματος (match-miss) γίνεται σε μικρότερο χρόνο. Δεδομένου ότι το λογισμικό που αναπτύχτηκε στα πλαίσια της παρούσας εργασίας μοντελοποιεί NAND CAM ενώ υπάρχουν ήδη μοντέλα για NOR CAM (παρέχονται στο CACTI), η παρούσα δουλειά θα μπορούσε στο μέλλον να επεκταθεί έτσι ώστε να μοντελοποιεί και SP CAM.



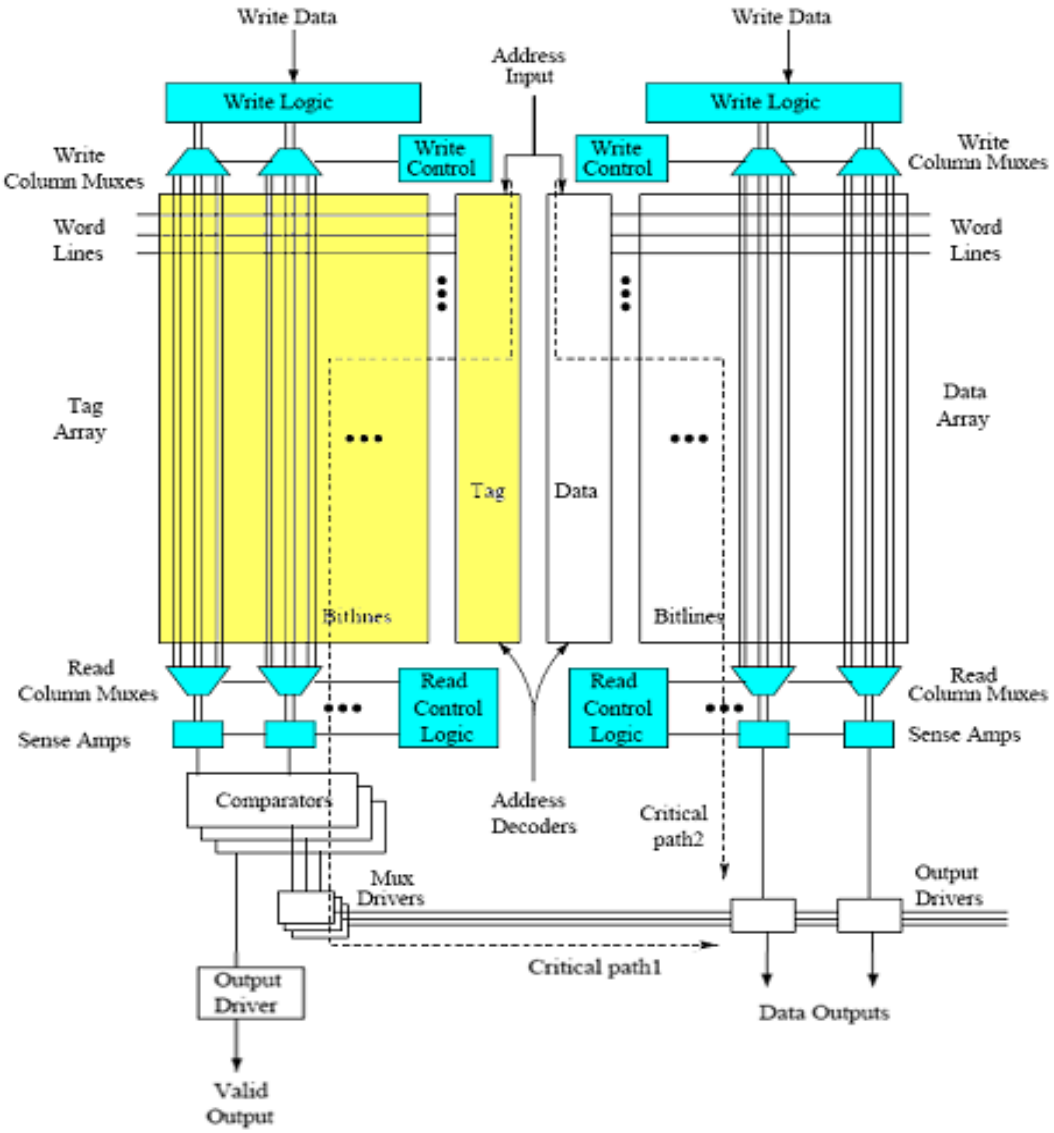
Σχήμα 2.13 SPCAM [7]

2.6. Εργαλεία CACTI και SPECTRE

CACTI

Το εργαλείο CACTI [4] χρησιμοποιείται για την αποτίμηση των βελτιστοποιήσεων στις αρχιτεκτονικές μνήμης. Ως αποτέλεσμα το CACTI παρέχει τους χρόνους πρόσβασης (access time), την επιφάνεια (area), και την κατανάλωση ισχύος (energy consumption), για όλα τα δομικά κομμάτια μιας κρυφής μνήμης όπως είναι οι αποκωδικοποιητές (decoders), τα bitlines, τα wordlines, οι συγκριτές (comparators), οι ενισχυτές στάθμης (sense amplifiers), οι οδηγοί εξόδου (output driver) κτλ.

Η βασική δομή της κρυφής μνήμης φαίνεται στο ακόλουθο σχήμα.



Σχήμα 2.14 Δομή μιας κρυφής μνήμης [9]

Το CACTI παίρνει τις ακόλουθες παραμέτρους ως είσοδο και παρέχει την βελτιστοποιημένη οργάνωση μιας κρυφής μνήμης μέσω μιας διαδικασίας βελτιστοποίησης. Οι παράμετροι εισόδου που δέχεται το CACTI είναι οι ακόλουθες:

C: Μέγεθος Κρυφής Μνήμης σε bytes
B: Μέγεθος Block σε bytes
A: Συσχετικότητα (Associativity)
TECH: Τεχνολογία
Nsubbanks: Αριθμός ξεχωριστών τμημάτων της κρυφής μνήμης

Η λειτουργία βελτιστοποίησης του CACTI έγκειται στην εύρεση της βέλτιστης λύσης ανάμεσα σε διάφορες οργανωτικές παραμέτρους, όπως ο αριθμός των οριζόντιων και κάθετων χωρισμάτων του data και του tag array.

Για ένα δοσμένο συνδυασμό παραμέτρων κρυφής μνήμης (μέγεθος κρυφής μνήμης, μέγεθος block, και συσχετικότητα), ο χρόνος πρόσβασης της κρυφής μνήμης, η κατανάλωση ισχύος, και η επιφάνεια μεταβάλλονται καθώς μεταβάλλονται οι ανωτέρω οργανωτικές παράμετροι. Για τον προσδιορισμό της βέλτιστης οργάνωσης κρυφής μνήμης, το CACTI χρησιμοποιώντας διάφορα αναλυτικά μοντέλα, εξαντλητικά υπολογίζει επιφάνεια, ενέργεια και χρόνο πρόσβασης για κάθε πιθανό συνδυασμό των παραμέτρων αυτών και επιλέγει εκείνο το συνδυασμό ο οποίος αντιστοιχεί στα βέλτιστα αποτελέσματα.

Το εργαλείο λογισμικού για την μοντελοποίηση CAM Μνημών με NAND Match Line που παρουσιάζεται στο επόμενο κεφάλαιο βασίζεται στο CACTI, αφού έχουν χρησιμοποιηθεί αρκετές μέθοδοι από αυτό. Όμως, το προτεινόμενο λογισμικό διαφοροποιείται σημαντικά, διότι είναι προσανατολισμένο στο να υπολογίζει χρόνο πρόσβασης, κατανάλωση ενέργειας και επιφάνεια για NAND CAM, επιλογή η οποία δεν προσφέρεται από το CACTI.

SPECTRE

Η πιστοποίηση της ορθότητας των αποτελεσμάτων, που δίδει το εργαλείο λογισμικού που παρουσιάζεται, ήγειρε την απαίτηση για σύγκρισή τους με αποτελέσματα που θα παραχθούν από κάποιο έγκυρο προσομοιωτή ηλεκτρονικών κυκλωμάτων. Για τον λόγο αυτό έγινε χρήση του SPECTRE, το οποίο είναι μια παραλλαγή του SPICE (Simulation Program with Integrated Circuit Emphasis), που ανέπτυξε η εταιρία Cadence. Το SPECTRE, εκτός των άλλων λειτουργιών, επιτρέπει την απεικόνιση των αποτελεσμάτων σε μορφή γραφημάτων.

Η προσομοίωση με SPECTRE/SPICE θεωρείται ότι παρέχει τη μέγιστη δυνατή ακρίβεια αποτελεσμάτων και τα αποτελέσματα που παρέχει σχεδόν ταυτίζονται με αυτά που μετριούνται σε κατασκευασμένα κυκλώματα. Το SPECTRE, επειδή υπολογίζει αναλυτικά όλα τα στοιχεία του κυκλώματος, είναι εξαιρετικά αργό ειδικά για μεγάλα κυκλώματα όπως αυτά των μνημών. Επιπλέον, τέτοιου είδους εφαρμογές απαιτούν ακριβές άδειες χρήσης. Για τους λόγους αυτούς, εφαρμογές όπως το CACTI είναι εξαιρετικά χρήσιμες στους αρχιτέκτονες υπολογιστών και δεν μπορούν να υποκατασταθούν από προσομοιωτές κυκλωμάτων όπως το SPICE.

ΚΕΦΑΛΑΙΟ 3. ΜΟΝΤΕΛΟΠΟΙΗΣΗ NAND CAM

Η παρούσα εργασία στηρίζεται σε μία σύγχρονη NAND CAM που προτάθηκε από τους Vikas Chaudhary και Lawrence Clark στο άρθρο “Low-Power High-Performance NAND Match Line Content Addressable Memories” [8]. Η NAND CAM που προτάθηκε στο [8], έχει σχεδιαστεί σε επίπεδο transistor και συνεπώς έχει συγκεκριμένο μέγεθος.

Αντίθετα, στην παρούσα εργασία μοντελοποιείται NAND CAM με την παραπάνω οργάνωση αλλά για οποιοδήποτε μέγεθος είτε του αριθμού των στηλών είτε των σειρών του Tag Array, χρησιμοποιώντας μια μεθοδολογία παρόμοια με αυτή του

εργαλείου CACTI [9]. Στο κεφάλαιο αυτό περιγράφεται αναλυτικά η οργάνωση και στη συνέχεια ο τρόπος με τον οποίο μοντελοποιείται μια NAND CAM. Τέλος παρουσιάζεται η υλοποίηση ενός εργαλείου που υπολογίζει το εμβαδό, τη κατανάλωση δυναμικής ενέργειας, το χρόνο και το κύκλο προσπέλασης μιας NAND CAM οποιουδήποτε μεγέθους.

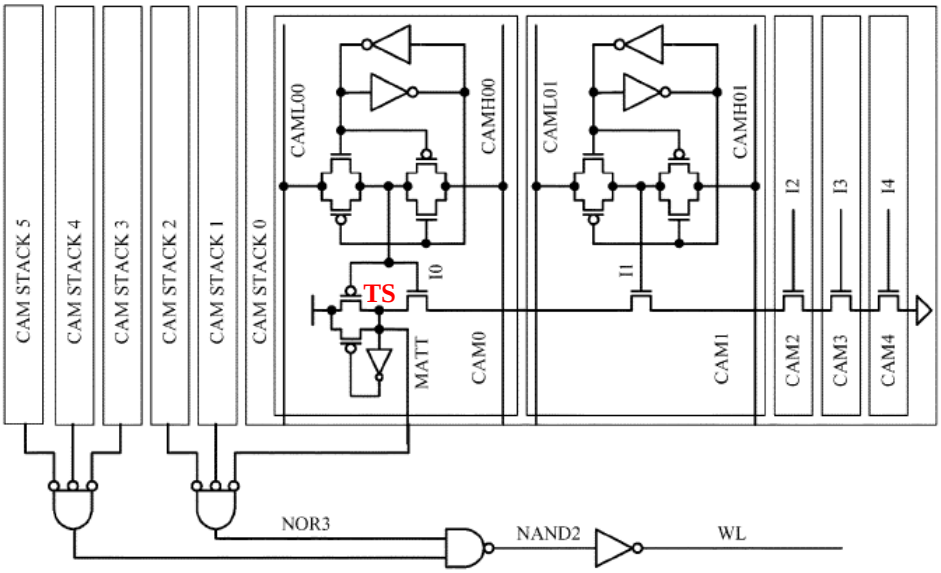
3.1. Βασικό Μοντέλο NAND CAM

Σύμφωνα με το [8] η καθυστέρηση της NAND match line αυξάνεται σχεδόν γραμμικά με το πλήθος των cells της tag line που βρίσκονται συνδεδεμένα στην ίδια match line. Για τον λόγο αυτό, γίνεται επιτακτική η ανάγκη διαχωρισμού των cells της tag line σε τμήματα (stacks) των οποίων το μέγεθος (stack depth) θα είναι τέτοιο που θα συμβάλλει στην βελτιστοποίηση των χαρακτηριστικών της καθυστέρησης που εμφανίζει η κρυφή μνήμη, της επιφάνειας που απαιτείται καθώς και της καταναλισκόμενης ενέργειας. Οι έξοδοι των επιμέρους τμημάτων συνενώνονται από μία δενδρική δομή από λογικές πύλες, παράγοντας με τον τρόπο αυτό το σήμα ταυτοποίησης ML της κάθε tag line.

Στο Σχήμα 3.1 παρουσιάζεται η υλοποίηση μίας CAM με tag line μεγέθους 26 bit, όπως προτείνεται στο [8]. Είναι εμφανής η δενδρική δομή των λογικών πυλών που συνενώνουν τα stacks. Το μέγεθος των δύο stack είναι 5 bits και των υπολοίπων είναι 4 bits, με αποτέλεσμα την ύπαρξη έξι stacks. Αξίζει να σημειωθεί ότι το cell στην κορυφή του stack (κοντά στην έξοδο) είναι διαφορετικό από τα υπόλοιπα: περιέχει επιπλέον κυκλωμάτωση για την «προφόρτιση» της εξόδου (TS/MATT) και κύκλωμα συγκράτησης (keeper), των οποίων η λειτουργία αναλύεται παρακάτω.

Ο συνδυασμός της πρώτης και της δεύτερης τριάδας γίνεται με την χρήση δύο NOR3 πυλών, οι έξοδοι των οποίων δίδονται ως είσοδος σε μία NAND2. Ο αντιστροφέας που ακολουθεί επαναφέρει το ανεστραμμένο σήμα που παράγεται στη σωστή λογική στάθμη και επιπλέον παίζει τον ρόλο του ενισχυτή σήματος. (Το πλήθος των αντιστροφέων που μπορεί να υπάρχουν εξαρτάται από το φορτίο που οδηγείται και πάντα είναι σε μονό αριθμό). Οι πύλες αυτές αποτελούν το δέντρο συνένωσης που

συνδυάζει τα επιμέρους σήματα match και δίνει ως έξοδο το συνολικό σήμα match της γραμμής.



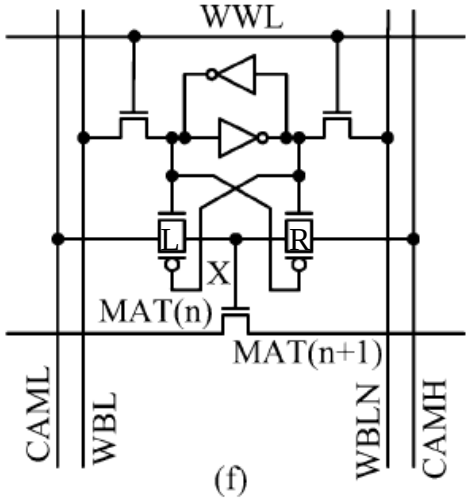
Σχήμα 3.1 CAM Tag με ιεραρχική NAND match line εύρους 26 bits

(Οι bitlines, wordlines καθώς και η περιφερειακή κυκλωμάτωση παραλείπονται για λόγους ευκρίνειας) [8]

Το κύκλωμα ολοκληρώνει ένα πλήρη κύκλο αναζήτησης σε δύο φάσεις. Κατά την πρώτη φάση (precharge), όλες οι searchlines οδηγούνται σε μηδενικό δυναμικό με αποτέλεσμα τα Match Line transistors των cells (I0, I1,..) να είναι σε κατάσταση αποκοπής και το top of the stack (σημείο TS, Σχήμα 3.1) να φορτίζεται σε υψηλό δυναμικό. Κατά συνέπεια το σήμα εξόδου (WL) να είναι μηδέν. Στη δεύτερη φάση οι συμπληρωματικές Search Lines οδηγούνται από τους Search Line drivers με τα προς αναζήτηση bits. Ακολουθεί σύγκριση στα cells με το αποθηκευμένο σε αυτά bit, προκαλώντας στη Match Line του stack κατάσταση miss ή match με την δεύτερη περίπτωση να αποφορτίζει ολόκληρη τη Match Line του stack. Στην περίπτωση όπου όλα τα stacks αποφορτιστούν, δηλαδή τα δεδομένα που είναι αποθηκευμένα στα cells ολόκληρης της tag line ταιριάζουν με τα προς αναζήτηση δεδομένα, παράγεται μέσω του δέντρου συνένωσης το τελικό σήμα WL με τιμή 1.

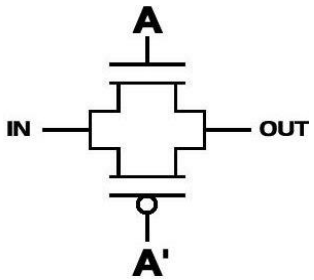
3.1.1. NAND CAM Cell

Τα NAND CAM cells, πέρα από τους δύο cross coupled inverters και τα δύο NMOS transistors που χρησιμοποιούνται για την εγγραφή και ανάγνωση των δεδομένων μέσω των bitlines στους inverters, χρησιμοποιούν δύο συμπληρωματικές passgates για τη σύγκριση της αποθηκευμένης τιμής με τις SL.



Σχήμα 3.2 11 T SRAM Cell [8]

Οι passgates του cell αποτελούνται από δύο transistor η κάθε μία, ένα NMOS και ένα PMOS συνδεδεμένα παράλληλα όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 3.3 Σχηματική αναπαράσταση μίας pass gate (ή transmission gate)

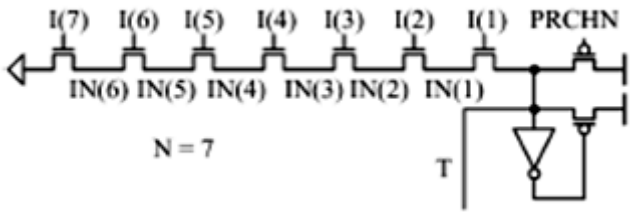
Τα transistor των passgates είναι ταυτόχρονα on, εμφανίζοντας στην έξοδο την τάση της εισόδου, ή off εμφανίζοντας στην έξοδο υψηλή εμπέδηση (high Z). Αυτό επιτυγχάνεται με την κατάλληλη διασύνδεση με τους cross coupled inverters. Όταν

στο cell είναι αποθηκευμένο λογικό ένα, τότε αυτό εμφανίζεται στο σημείο A της δεξιάς (R) passgate και στο σημείο A’ της αριστερής (L) passgate, ενώ εμφανίζεται λογικό μηδέν στα σημεία A’ και A των R και L αντίστοιχα. Στην περίπτωση αυτή τα transistor της passgate R βρίσκονται και τα δύο σε αγώγιμη κατάσταση, ενώ αυτά της L σε αποκοπή.

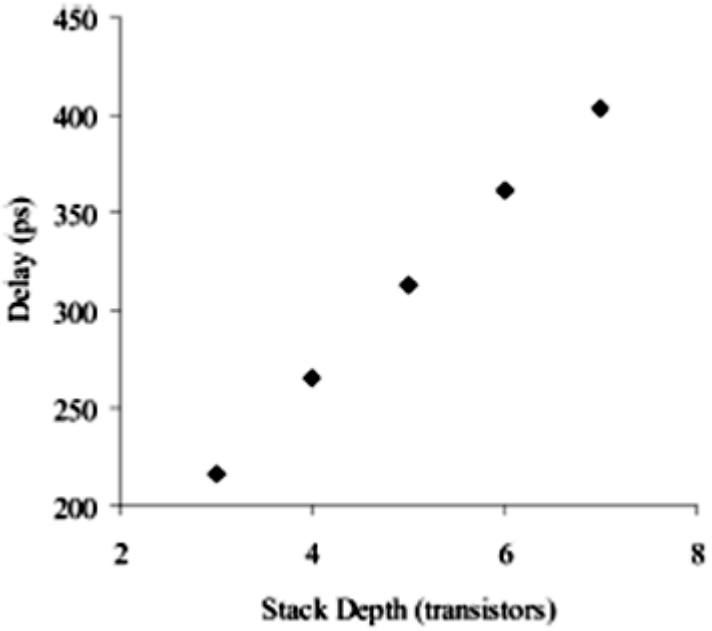
Δεδομένου ότι στα PMOS transistor “περνάει” καλύτερα το λογικό ένα, ενώ αντίθετα στα NMOS “περνάει” καλύτερα το λογικό μηδέν, ο συνδυασμός τους σε passgate, εξασφαλίζει πως όποια και να είναι η λογική στάθμη στην είσοδό της θα εμφανιστεί επίσης σωστά στην έξοδό της και κατ’ επέκταση και στον κόμβο X. Έτσι το ML transistor οδηγείται με την κατάλληλη τάση (full rail) στην πύλη του με αποτέλεσμα να μην επηρεάζεται αρνητικά η καθυστέρηση της ML αλυσίδας.

3.2. Περιορισμοί μεγέθους Stack (stack depth)

Ένα γενικό NAND stack κύκλωμα (Σχήμα 3.4), προσομοιώθηκε με Cadence SPECTRE για τον καθορισμό της ταχύτητας σε τεχνολογία 0.25 μm [8]. Η καθυστέρηση μετρήθηκε με όλες τις εισόδους I(x) να αλλάζουν ταυτόχρονα όπως θα συνέβαινε στη CAM. Από τα αποτελέσματα που παρουσιάζονται στο Σχήμα 3.5 είναι εμφανές πως η αύξηση της καθυστέρησης είναι σχεδόν γραμμικά ανάλογη με την αύξηση του μεγέθους του stack, περιορίζοντας έτσι το πρακτικά χρήσιμο βάθος του stack σε μικρά μεγέθη.



Σχήμα 3.4 NAND stack κύκλωμα με stack depth 7 [8]



Σχήμα 3.5 Η καθυστέρηση του stack σε σχέση με το πλήθος των transistors που το συνιστούν. [8]

Εκτός από τη μεγάλη καθυστέρηση, ένα μεγάλο stack αντιμετωπίζει προβλήματα λόγω του φαινομένου διαμοιρασμού φορτίου (charge sharing). Το φαινόμενο αυτό εμφανίζεται σε δυναμικά κυκλώματα όταν μετά τη φάση προφόρτισης ενός κόμβου, Τ στο Σχήμα 3.4, άγει ένα ή περισσότερα transistor (π.χ. I1 στο σχήμα) που συνδέονται στο δυναμικό κόμβο (συνήθως η έξοδος) και σε άλλους κόμβους που έχουν χαμηλότερο δυναμικό (π.χ. IN(1) στο σχήμα), αλλά δεν είναι συνδεδεμένοι απευθείας στη γείωση. Σε μια τέτοια περίπτωση, φορτίο από τον προφορτισμένο κόμβο Τ, διαφεύγει, μέσω των transistor, στους άλλους κόμβους μέχρι όλοι οι συνδεδεμένοι κόμβοι να φτάσουν στο ίδιο δυναμικό. Το αποτέλεσμα είναι ότι ο προφορτισμένος κόμβος καταλήγει να έχει χαμηλότερο δυναμικό από ότι ήταν αναμενόμενο. Αν η μείωση είναι μεγάλη μπορεί η επόμενη βαθμίδα να ερμηνεύσει λάθος τη λογική τιμή.

Αναφερόμενοι στο κύκλωμα του Σχήματος 3.4, η χειρότερη περίπτωση charge sharing εμφανίζεται στο ακόλουθο σενάριο που διαρκεί δύο διαδοχικούς κύκλους ρολογιού. Στον πρώτο κύκλο ρολογιού αποφορτίζονται όλοι οι ενδιάμεσοι κόμβοι IN(1) – IN(6) εκτός του πρώτου Τ (κατάσταση miss), επειδή τα cells 2 έως 7 κάνουν

match, ενώ στον ακόλουθο κύκλο κάνουν match τα cells 1 έως 6. Στην περίπτωση αυτή, μετά τη φάση προφόρτισης στην αρχή του δεύτερου κύκλου, το φορτίο του κόμβου T, μοιράζεται στους κόμβους IN(1) - IN(6) αφού τα transistors I(1) – I(6) βρίσκονται σε αγώγιμη κατάσταση και κατά τον προηγούμενο κύκλο το δυναμικό των κόμβων IN(1) – IN(6) είχε πέσει στο 0. Η τάση στον κόμβο T, παρόλο της ύπαρξης κυκλώματος διατήρησης (το δεύτερο NMOS transistor μαζί με τον αντιστροφέα), μπορεί να γίνει μικρότερη από το λογικό κατώφλι (threshold) της επόμενης πύλης, διαδίδοντας σε αυτή θόρυβο ως έγκυρο λογικό επίπεδο.

Στο σενάριο που περιγράφηκε παραπάνω ένα stack μεγέθους 8 cells σε τεχνολογία 0.25 μm, αποτυγχάνει λόγω εντονότατου charge sharing αφού παρατηρείται η τάση στην έξοδο να πέφτει κάτω από τα Vdd/2 σε όλα τις process corners ενώ ένα stack των 7 cells αποτυγχάνει στις περισσότερες από αυτές [8]. Κατά συνέπεια, το μέγιστο μέγεθος ενός stack είναι 6, αφού stacks αυτού του μεγέθους λειτουργούν σωστά σε όλες τις συνθήκες λειτουργίας.

Ο όρος process corners αναφέρεται στις ακραίες τιμές, που μπορεί να πάρουν διάφορες παράμετροι είτε της τεχνολογίας είτε του περιβάλλοντος του Ο.Κ. όπως τάση κατωφλίου, θερμοκρασία, τάση λειτουργίας, κτλ. Όταν ένα κύκλωμα αποτυγχάνει να λειτουργήσει σε όλες τις corners τότε ο σχεδιασμός του θεωρείται απαράδεκτος αναφορικά με την ευελιξία λειτουργίας.

3.3. Μοντελοποίηση της NAND CAM

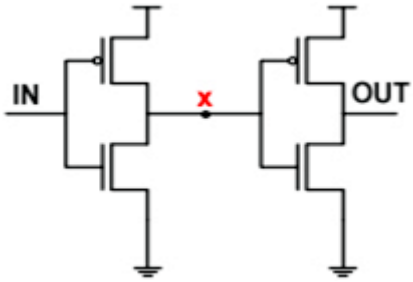
Η μεθοδολογία που χρησιμοποιήθηκε για την ανάλυση είναι παρόμοια με αυτή που χρησιμοποιείται στο CACTI. Συγκεκριμένα, για τον υπολογισμό καθυστέρησης, γίνεται RC ανάλυση για κάθε στοιχείο του κυκλώματος, μοντελοποιώντας τα transistor που άγουν ως αντιστάσεις, λαμβάνοντας όμως υπ όψη και τις παρασιτικές χωρητικότητες της πύλης (gate), της πηγής (source) και της απορροής (drain).

Επιπλέον, οι αγωγοί μοντελοποιούνται και αυτοί ως αντιστάσεις και παρασιτικές χωρητικότητες. Δεδομένου ότι η μοντελοποίηση αναφέρεται σε NAND CAM, χρησιμοποιήθηκαν από το CACTI οι μέθοδοι που υπήρχαν για NOR CAM για τις λειτουργίες της ανάγνωσης και εγγραφής και αναπτύχτηκαν νέες για τη διαδικασία που διαφοροποιείται, δηλαδή την αναζήτηση.

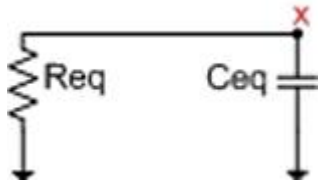
Η μέγιστη καθυστέρηση εμφανίζεται στο κρίσιμο μονοπάτι του κυκλώματος, το οποίο ξεκινάει από τον οδηγό της SL μέχρι και την πιο απομακρυσμένη γραμμή του array, η καθυστέρηση που εμφανίζει η ML του μεγαλύτερου stack και τέλος η καθυστέρηση του δέντρου συνένωσης των stacks. Το ακριβές κρίσιμο μονοπάτι εκτός του τμήματος της SL, εξαρτάται από τον τρόπο διαχωρισμού της tag line σε stacks, κάτι που εξετάζεται, παρακάτω, στην παράγραφο 3.6.1.

3.3.1. Υπολογισμός Καθυστέρησης με RC Ανάλυση

Σύμφωνα με το [9], κάθε συστατικό στοιχείο ενός κυκλώματος, μπορεί να αποσυντεθεί σε ένα ή περισσότερα RC κυκλώματα πρώτης ή δεύτερης τάξης. RC κύκλωμα πρώτης τάξης ονομάζουμε το δικτύωμα που αποτελείται από έναν μόνο πυκνωτή και μία αντίσταση διασυνδεμένα σε σειρά, ενώ δεύτερης τάξης ονομάζουμε το δικτύωμα που αποτελείται από τουλάχιστον δύο πυκνωτές διασυνδεμένους παράλληλα με μία ή περισσότερες αντιστάσεις να παρεμβάλλονται.



Σχήμα 3.6 Σχηματικό κυκλώματος δύο αντιστροφέων



Σχήμα 3.7 Ισοδύναμο RC δικτύωμα πρώτης τάξης

Στο Σχήμα 3.6 εμφανίζεται το σχηματικό κύκλωμα δύο αντιστροφέων. Η καθυστέρηση του αντιστροφέα που οδηγεί τον κόμβο x, μπορεί να προσδιοριστεί με τη χρήση του ισοδύναμου κυκλώματος του Σχήματος 3.7.

Στο κύκλωμα του Σχήματος 3.7, το μονοπάτι αποφόρτισης του πρώτου αντιστροφέα (στην περίπτωση όπου η τάση εισόδου αυξάνει – rising input) έχει αντικατασταθεί από μία αντίσταση και οι χωρητικότητες της εκροής του πρώτου αντιστροφέα και πύλης του δεύτερου αντιστροφέα έχουν αντικατασταθεί από έναν πυκνωτή. Στις περιπτώσεις όπου οι δύο πύλες διασυνδέονται με ένα μεγάλου μήκους αγωγό, τότε η παρασιτική χωρητικότητα και η αντίστασή του συνυπολογίζονται στην ισοδύναμη χωρητικότητα Ceq και αντίσταση Req αντίστοιχα.

Η καθυστέρηση του ισοδύναμου κυκλώματος του Σχήματος 3.7 υπολογίζεται χρησιμοποιώντας τις εξισώσεις που παρουσιάζονται στο [11]. Έτσι με το δυναμικό της εισόδου να αυξάνεται, η καθυστέρηση είναι:

$$delay = t_f \sqrt{\left[\log\left(\frac{V_{th}}{V_{dd}}\right)\right]^2 + 2t_{rise} b\left(1 - \frac{V_{th}}{V_{dd}}\right)} / t_f$$

(Σχέση 3.3.1)

Και με το δυναμικό της εισόδου να μειώνεται, η καθυστέρηση είναι:

$$delay = t_f \sqrt{\left[\log\left(1 - \frac{V_{th}}{V_{dd}}\right)\right]^2 + \left(\frac{2t_{fall} \cdot b \cdot V_{th}}{t_f \cdot V_{dd}}\right)}$$

(Σχέση 3.3.2)

Όπου :

- tf, η σταθερά χρόνου τ (με tf = Req * Ceq)
- V_{th}, το δυναμικό κατωφλίου του αντιστροφέα
- V_{dd}, το δυναμικό της τροφοδοσίας
- t_{rise}, ο χρόνος που απαιτείται για την μεταβολή της εισόδου (ανερχόμενη ράμπα)
- t_{fall}, ο χρόνος που απαιτείται για την μεταβολή της εισόδου (κατερχόμενη ράμπα)
- b, το κλάσμα του δυναμικού της εισόδου , κατά το οποίο η έξοδος αλλάζει (για ανερχόμενη είσοδο b = 0.5 και για κατερχόμενη είσοδο b = 0.4)

Ως καθυστέρηση πύλης ορίζεται ο χρόνος από τη στιγμή που η είσοδος βρίσκεται σε δυναμικό ίσο με το δυναμικό κατωφλίου (threshold voltage) της πύλης Vth1, μέχρι τη στιγμή που το δυναμικό της εξόδου της πύλης γίνεται ίσο με το δυναμικό κατωφλίου της επόμενης πύλης Vth2. [9]. Θεωρώντας πως με τον όρο πύλη, μπορεί να περιγραφεί κάθε καλά ορισμένο τμήμα ενός κυκλώματος που αποτελείται από

αγωγούς και transistors, ενός δηλαδή RC δικτυώματος, οι παρακάτω σχέσεις μπορούν να βρουν εφαρμογή και στον υπολογισμό της καθυστέρησης της Match Line καθώς και της Search Line.

Έτσι για είσοδο που το δυναμικό της αυξάνεται (rising input) η καθυστέρηση είναι [9]:

$$delay = t_f \sqrt{\left[\log\left(\frac{V_{th1}}{V_{dd}}\right)\right]^2 + 2t_{rise} b\left(1-\frac{V_{th1}}{V_{dd}}\right)/t_f} + t_f\left[\log\left(\frac{V_{th1}}{V_{dd}}\right) - \log\left(\frac{V_{th2}}{V_{dd}}\right)\right]$$

(Σχέση 3.3.3)

Ενώ για είσοδο που το δυναμικό της μειώνεται (falling input) η καθυστέρηση είναι [9]:

$$delay = t_f \sqrt{\left[\log\left(1-\frac{V_{th1}}{V_{dd}}\right)\right]^2 + \left(\frac{2t_{fall}\cdot b\cdot V_{th1}}{t_f\cdot V_{dd}}\right)} + t_f\left[\log\left(1-\frac{V_{th1}}{V_{dd}}\right) - \log\left(1-\frac{V_{th2}}{V_{dd}}\right)\right]$$

(Σχέση 3.3.4)

3.3.2. Υπολογισμός δυναμικής ενέργειας και επιφάνειας

Για τον υπολογισμό της δυναμικής ενέργειας που καταναλώνει η CAM, αρκεί να υπολογιστεί η χωρητικότητα των κόμβων που αλλάζουν λογική τιμή, ενώ για τον υπολογισμό της επιφάνειας που καταλαμβάνει η CAM υπολογίστηκαν οι διαστάσεις των CAM cells, των πυλών του δέντρου συνένωσης και των περιφερειακών κυκλωμάτων. Οι υπολογισμοί αυτοί περιγράφονται αναλυτικά στις παραγράφους 3.9 και 3.10.

Στη συνέχεια, ακολουθούν οι RC αναλύσεις για την NAND Match Line, Search Line καθώς και για τη δενδρική δομή συνένωσης των stacks της Match Line, τα οποία αποτελούν τα δομικά στοιχεία του κρίσιμου μονοπατιού.

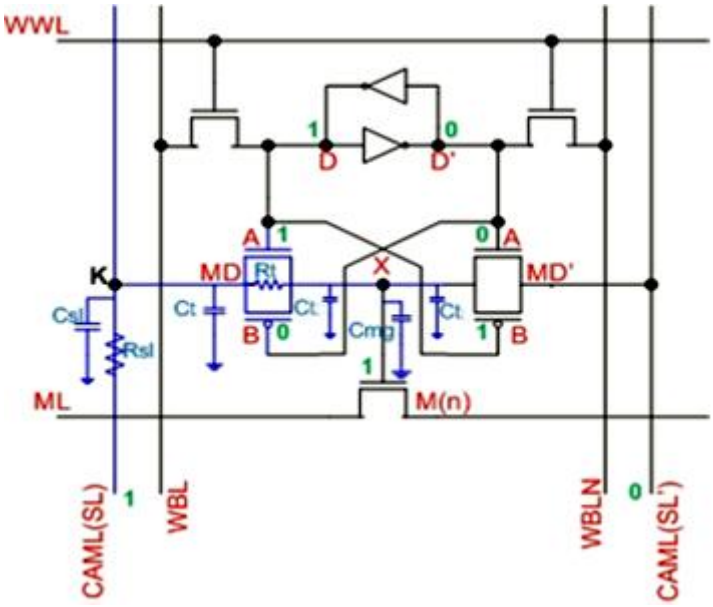
3.4. RC Ανάλυση της Search Line

Η χειρότερη περίπτωση καθυστέρησης κατά τη διαδικασία αναζήτησης, εμφανίζεται όταν όλα τα cells της ίδιας στήλης του memory array έχουν την ίδια τιμή με αυτή που αναζητείται. Αυτό συμβαίνει διότι το σήμα που ξεκινάει από τον Searchline Driver πρέπει να διαδοθεί όχι μόνο σε ολόκληρη τη searchline αλλά και εσωτερικά στο κάθε cell που κάνει match αφού η passgate που συνδέεται στη SL είναι ανοιχτή.

Στο Σχήμα 3.8 εμφανίζεται με μπλε χρώμα το μονοπάτι που ενεργοποιείται στο cell (με πράσινο χρώμα οι λογικές τιμές), όταν κατά τη διαδικασία αναζήτησης με SL=1 και D=1 έχουμε περίπτωση match (αντίστοιχα και όταν SL'=1 και D'=1).

Η καθυστέρηση που εμφανίζεται σε κάθε κελί της ίδιας στήλης του memory array που έχει την ίδια τιμή με αυτή που αναζητείται, συντίθεται από τις επιμέρους καθυστερήσεις των στοιχείων που βρίσκονται στο μονοπάτι που ενεργοποιείται. Τα στοιχεία αυτά είναι:

- Searchline: Η καθυστέρηση λόγω χωρητικότητας και αντίστασης της searchline με μήκος από τον κόμβο Κ ενός cell μέχρι τον κόμβο Κ του επόμενου.
- Passgate: Η καθυστέρηση λόγω χωρητικότητας των drain και source, καθώς και λόγω της αντίστασης της passgate (MD A, B).
- Passgate: Η καθυστέρηση λόγω χωρητικότητας της source passgate MD' A, B του cell.
- Match line transistor's gate: Η καθυστέρηση λόγω χωρητικότητας της πύλης του NMOS transistor (M(n)) που όταν ενεργοποιείται συνδέει τα δύο τμήματα της matchline.



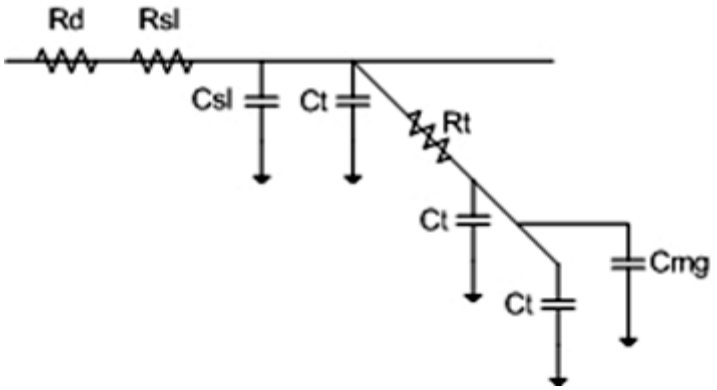
Σχήμα 3.8 RC Ανάλυση της Search Line

Επιπλέον υπάρχει και η αντίσταση του αντιστροφέα – οδηγού (data driver) της search line Rd (δεν εμφανίζεται στο Σχήμα 3.8). Σύμφωνα με το τρόπο λειτουργίας της NAND CAM, μία από τις δύο συμπληρωματικές searchlines θα οδηγηθεί σε λογικό 1, αφού κατά την πρώτη φάση του κύκλου αναζήτησης και οι δύο γραμμές είχαν οδηγηθεί σε λογικό 0. Επομένως η αντίσταση Rd είναι ίση με την αντίσταση του PMOS transistor που βρίσκεται σε αγώγιμη κατάσταση. Η Rd δεν εμφανίζεται στο παραπάνω σχήμα για λόγους ευκρίνειας.

Στο Σχήμα 3.9 παρουσιάζεται ένα ισοδύναμο RC μοντέλο του μονοπατιού που παρουσιάζεται με μπλε χρώμα στο Σχήμα 3.8, καθώς και την αντίσταση Rd που προαναφέρθηκε.
Το μοντέλο αποτελείται από:

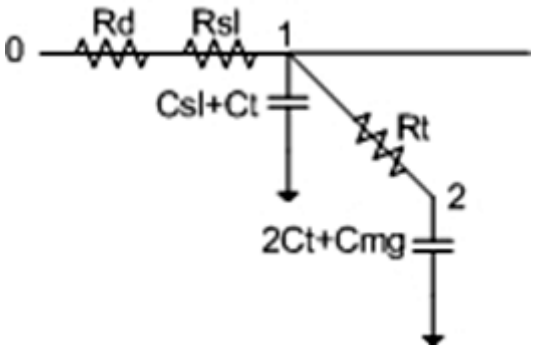
- Την αντίσταση (Rsl) και την χωρητικότητα (Csl) του τμήματος της searchline με μήκος από τον κόμβο K ενός cell μέχρι τον κόμβο K του επόμενου.
- Τη χωρητικότητα της drain (Ct), την αντίσταση αγώγιμης κατάστασης (Rt) και την χωρητικότητα της source (Ct) της passgate MD A, B. (Θεωρούμε για λόγους απλότητας ότι χωρητικότητα της drain είναι ίση με τη χωρητικότητα της source).

- Τη χωρητικότητα της source (Ct) της passgate MD’ A, B.
- Τη χωρητικότητα της gate (Cmg) της πύλης του transistor της matchline που ενεργοποιείται.
- Την αντίσταση Rd του data driver της search line.



Σχήμα 3.9 Ισοδύναμο Search Line RC μοντέλο (cell)

Στο Σχήμα 3.10 φαίνεται το μοντέλο σε απλοποιημένη μορφή. Οι χωρητικότητες Csl (searchline) και Ct (drain της passgate) προστίθενται αφού βρίσκονται σε παράλληλη σύνδεση, κάτι που συμβαίνει και με τις χωρητικότητες Ct (source της passgate MD), Cmg (gate του matchline transistor) και Ct (source της passgate MD’).



Σχήμα 3.10 Ισοδύναμο Search Line απλοποιημένο RC μοντέλο (cell)

Στο Σχήμα 3.11 φαίνεται μία searchline που διατρέχει μία στήλη του array με n cells. Όπως προαναφέρθηκε, η χειρότερη περίπτωση καθυστέρησης, εμφανίζεται όταν όλα τα cells της ίδιας στήλης έχουν την τιμή του αναζητείται. Η καθυστέρηση των search

lines είναι ο χρόνος που απαιτείται για τη φόρτιση του πυκνωτή του κόμβου **N**, δηλαδή του κόμβου που βρίσκεται πιο μακριά από τον οδηγό της searchline.

Σύμφωνα με τον κανόνα του W. C. Elmore [14] η καθυστέρηση διάδοσης του σήματος από τον κόμβο μηδέν (0) στον ι-οστό κόμβο, κατά προσέγγιση είναι:

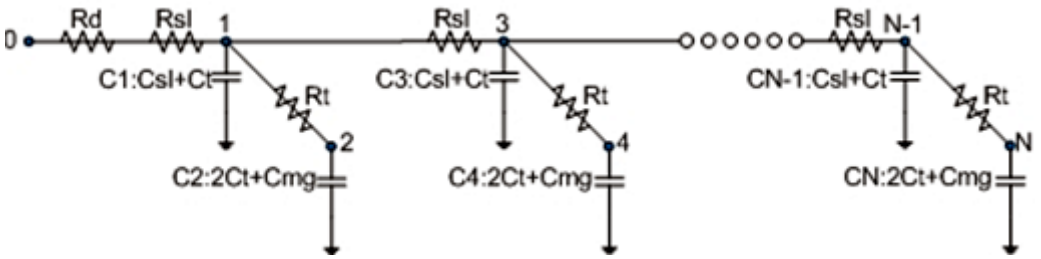
$$t_{pi} \approx \ln(2)\sum_{k=1}^{N} C_k R_{ik}$$

(Σχέση 3.4.1)

Όπου Rik είναι η αντίσταση του κοινού μονοπατιού για τους κόμβους i, k (πχ η αντίσταση που είναι κοινή για το μονοπάτι από τον κόμβο μηδέν στον κόμβο i και τον κόμβο μηδέν στον κόμβο k).

$$R_{ik} = \sum R_j \in [\, path(0 \to i) \cap (path(0 \to k))]$$

(Σχέση 3.4.2)



Σχήμα 3.11 Searchline RC μοντέλο

Θα πρέπει αρχικά να υπολογιστεί η αντίσταση των μονοπατιών από τον κόμβο μηδέν (0) προς όλους τους κόμβους (δηλαδή για όλες τις τιμές του K), που είναι κοινή με την αντίσταση του μονοπατιού από τον κόμβο μηδέν προς τον κόμβο N.

Με τη σχέση (3.4.2) και για τις τιμές i=N και K από 1 έως N έχουμε:

i=N, K=1: $R_{N1} = \sum R_j \in [path(0 \to N) \cap (path(0 \to 1))] = Rd + Rsl$

i=N, K=2: $R_{N2} = \sum R_j \in [path(0 \to N) \cap (path(0 \to 2))] = Rd + Rsl$

i=N, K=3: $R_{N3} = \sum R_j \in [path(0 \to N) \cap (path(0 \to 3))] = Rd + Rsl + Rsl$

i=N, K=4: $R_{N4} = \sum R_j \in [path(0 \to N) \cap (path(0 \to 4))] = Rd + Rsl + Rsl$

…….

i=N, K=N-1: $R_{N,N-1} = \sum R_j \in [path(0 \to N) \cap (path(0 \to N-1))] = Rd + \frac{\mathrm{N}}{2} Rsl$

i=N, K=N: $R_{N,N} = \sum R_j \in [path(0 \to N) \cap (path(0 \to N))] = Rd + \frac{\mathrm{N}}{2} Rsl + Rt$

Να σημειωθεί ότι ο υπολογισμός του πλήθους των Rsl για κάθε τιμή του K δίδεται από την κλαδική συνάρτηση:

$$f(K) = \begin{cases} \frac{K+1}{2} Rsl, για\_K\_περιττό \\ \frac{K}{2} Rsl, για\_K\_άρτιο \end{cases}$$

Κατά συνέπεια από τη σχέση (3.4.1) έχουμε:

$$t_{pN} \approx \ln(2) \sum_{k=1}^{N} C_k R_{\mathrm{N},k} \Leftrightarrow$$
$$t_{pN} \approx \ln(2)[C_1 R_{N,1} + C_2 R_{N,2} + C_3 R_{N,3} + C_4 R_{N,4} + ... + C_{N-1} R_{N,N-1} + C_N R_{N,N}] \Leftrightarrow$$

$$t_{pN} \approx \ln(2)[(C_{sl} + C_t) R_{N,1} + (2C_t + C_{mg}) R_{N,2} + (C_{sl} + C_t) R_{N,3} + (2C_t + C_{mg}) R_{N,4} + ...$$
$$+ (C_{sl} + C_t) R_{N,N-1} + (2C_t + C_{mg}) R_{N,N}] \Leftrightarrow$$

$$t_{pN} \approx \ln(2)[(C_{sl} + C_t)(R_d + R_{sl}) + (2C_t + C_{mg})(R_d + R_{sl}) + (C_{sl} + C_t)(R_d + R_{sl} + R_{sl})$$
$$+ (2C_t + C_{mg})(R_d + R_{sl} + R_{sl}) + ... + (C_{sl} + C_t)(R_d + \frac{\mathrm{N}}{2} R_{sl}) + (2C_t + C_{mg})(R_d + \frac{\mathrm{N}}{2} R_{sl} + R_t)] \Leftrightarrow$$

$$t_{pN} \approx \ln(2)[(C_{sl}+C_t)(R_d+R_{sl})+(2C_t+C_{mg})(R_d+R_{sl})+(C_{sl}+C_t)(R_d+R_{sl}+R_{sl})$$
$$+(2C_t+C_{mg})(R_d+R_{sl}+R_{sl})+.....+(C_{sl}+C_t)(R_d+\frac{\mathrm{N}}{2}R_{sl})$$
$$+(2C_t+C_{mg})R_d+(2C_t+C_{mg})\frac{\mathrm{N}}{2}R_{sl}+(2C_t+C_{mg})R_t] \Leftrightarrow$$
$$t_{pN} \approx \ln(2)[R_{sl}(3C_t+C_{sl}+C_{mg})+R_d(3C_t+C_{sl}+C_{mg})$$
$$+2R_{sl}(3C_t+C_{sl}+C_{mg})+R_d(3C_t+C_{sl}+C_{mg})+.....$$
$$+\frac{\mathrm{N}}{2}R_{sl}(3C_t+C_{sl}+C_{mg})+R_d(3C_t+C_{sl}+C_{mg})+R_t(2C_t+C_{mg})]$$

Δεδομένου ότι κάθε ένα από τα n cell περιέχει δύο από τους N κόμβους, δηλαδή το πλήθος των cells είναι $n=\frac{N}{2}$, με αντικατάσταση στην παραπάνω σχέση η καθυστέρηση γίνεται:

$$t_{pN} \approx \ln(2)[(R_{sl}+2R_{sl}+...+nR_{sl})(3C_t+C_{sl}+C_{mg})+nR_d(3C_t+C_{sl}+C_{mg})+R_t(2C_t+C_{mg})] \Leftrightarrow$$

$$t_{pN} \approx \ln(2)[(nR_d+\frac{n(n+1)}{2}R_{sl})(3C_t+C_{sl}+C_{mg})+R_t(2C_t+C_{mg})]$$

Και η σταθερά χρόνου τ είναι :

$$\tau=(nR_d+n\frac{(n+1)}{2}R_{sl})(3C_t+C_{sl}+C_{mg})+R_t(2C_t+C_{mg})$$

(Σχέση 3.4.3)

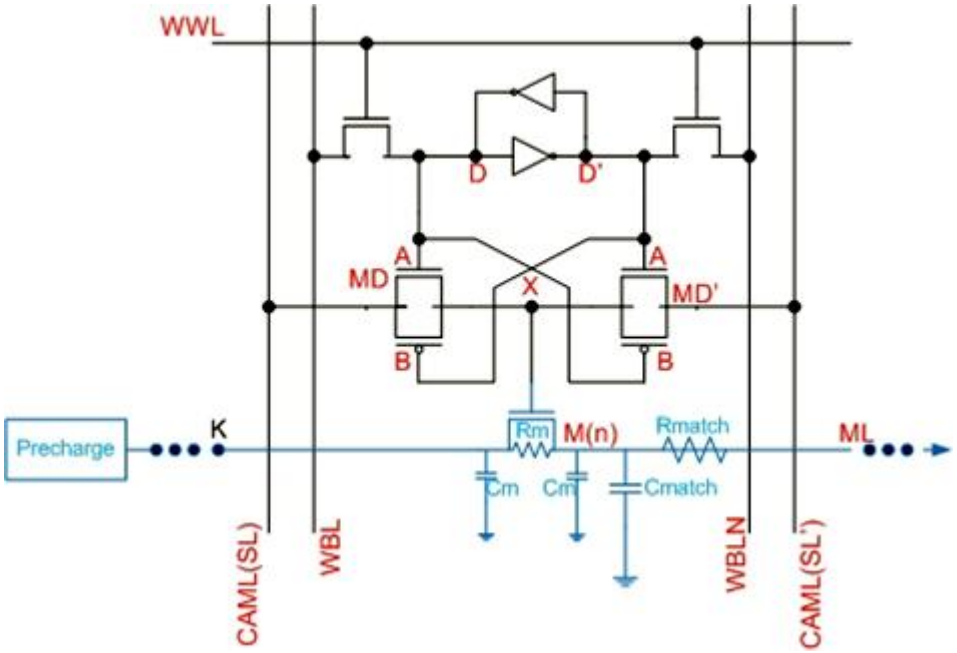
3.5. RC Ανάλυση της Match Line

Όπως προαναφέρθηκε, όταν όλα τα cells που βρίσκονται "πάνω" στην ίδια matchline περιέχουν τα bits που αναζητούνται κατά τη διαδικασία αναζήτησης, τότε ορίζεται περίπτωση match. Πριν τη διαδικασία αξιολόγησης, δηλαδή τη διαδικασία όπου ελέγχεται αν υπάρχει match ή miss, όλες οι NAND match lines σε ένα array προφορτίζονται, με το κύκλωμα προφόρτισης, σε λογικό ένα. Στη συνέχεια κατά τη διαδικασία αξιολόγησης, η matchline στην οποία εμφανίζεται περίπτωση match αποφορτίζεται και οδηγείται σε λογικό μηδέν.

Στο Σχήμα 3.12 εμφανίζεται με μπλε χρώμα το μονοπάτι που ενεργοποιείται στο cell, όταν κατά τη διαδικασία αναζήτησης με SL=1 και D=1 έχουμε περίπτωση match (αντίστοιχα και όταν SL'=1 και D'=1).

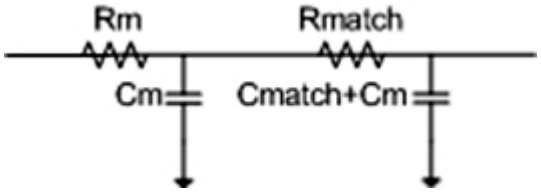
Η καθυστέρηση που εμφανίζεται σε κάθε κελί που βρίσκεται στην ίδια match line σε περίπτωση match, συντίθεται από τις επιμέρους καθυστερήσεις των στοιχείων που βρίσκονται στο μονοπάτι που ενεργοποιείται. Τα στοιχεία αυτά είναι:

- Matchline: Η καθυστέρηση λόγω χωρητικότητας (C match) και αντίστασης (R match) της matchline με μήκος από τον κόμβο K ενός cell μέχρι τον κόμβο K του επόμενου.
- Match transistor: Η καθυστέρηση λόγω χωρητικότητας των drain (Cm) και source (Cm), καθώς και λόγω της αντίστασης (Rm) του transistor της matchline.



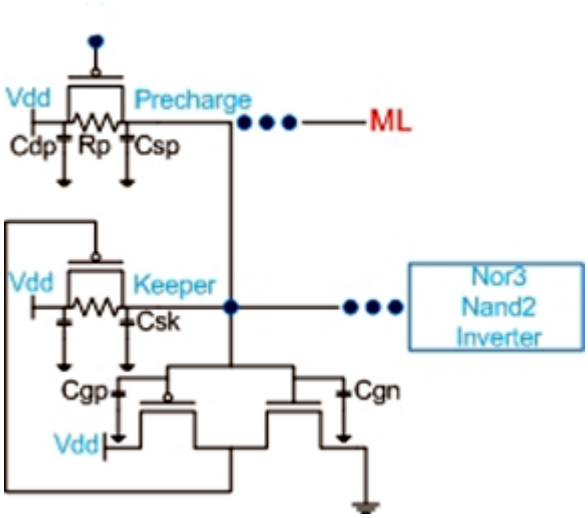
Σχήμα 3.12 RC ανάλυση της Match Line (cell)

Στο Σχήμα 3.13 παρουσιάζεται ένα ισοδύναμο RC μοντέλο του μονοπατιού που παρουσιάζεται με μπλε χρώμα στο Σχήμα 3.12.



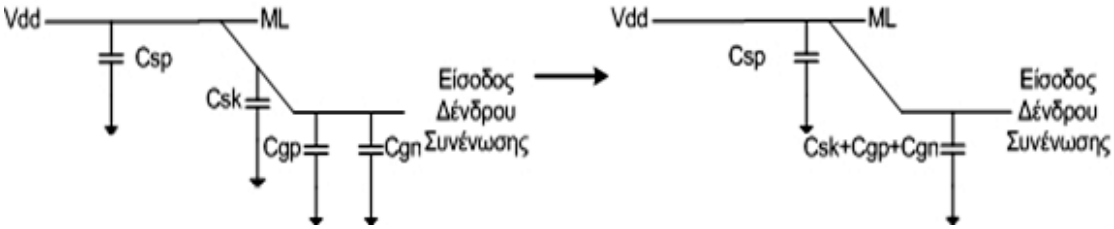
Σχήμα 3.13 Ισοδύναμο Match Line RC μοντέλο

Στο τελευταίο cell ενός stack (κορυφή) που δίνει την έξοδο, η matchline συνδέεται στο PMOS transistor προφόρτισης (Precharge), σε ένα ασθενές transistor συγκράτησης (Keeper) και στον αντιστροφέα του κυκλώματος συγκράτησης, όπως φαίνεται στο Σχήμα 3.14. Το κύκλωμα προφόρτισης επιβαρύνει τη matchline με τις χωρητικότητες των source (Csp) και (Csk) των PMOS transistor προφόρτισης και συγκράτησης αντίστοιχα, καθώς και με τις χωρητικότητες των πυλών (Cgn), (Cgp) των NMOS και PMOS transistor που συνιστούν τον αντιστροφέα.



Σχήμα 3.14 Κύκλωμα Precharger & συγκράτησης

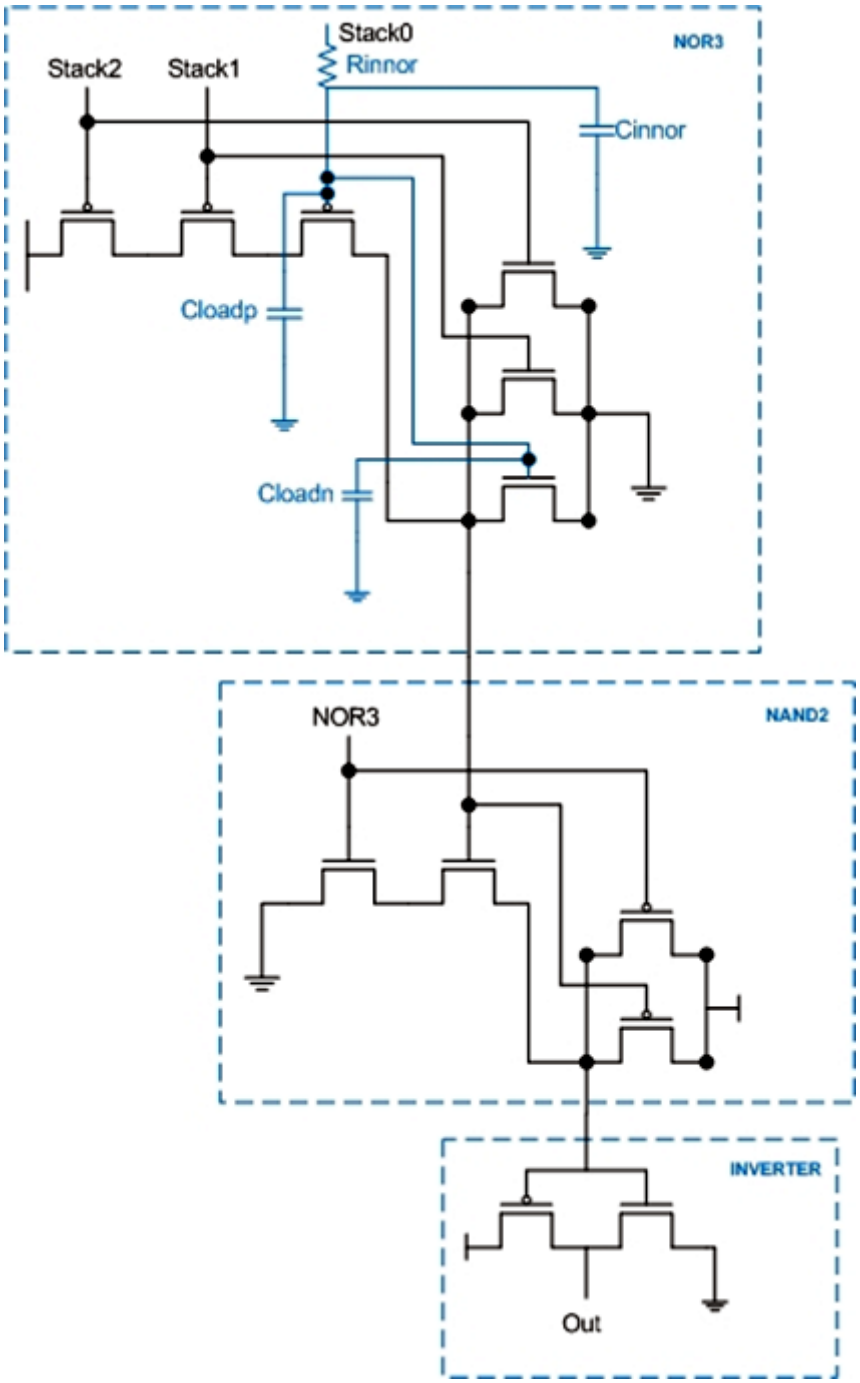
Πρέπει να σημειωθεί ότι ενώ το PMOS προφόρτισης (Precharge) είναι ανενεργό κατά τη διάρκεια λειτουργίας του matchline evaluation, το PMOS συγκράτησης (Keeper) είναι ενεργό και δυσκολεύει την εκφόρτισή της για ένα μικρό χρονικό διάστημα, μέχρι η τάση της εξόδου του stack να πέσει κάτω από το κατώφλι του αντιστροφέα συγκράτησης. Το φαινόμενο αυτό δεν μοντελοποιείται στην παρούσα εργασία, αλλά τα τελικά αποτελέσματα δείχνουν ότι το σφάλμα στην ακρίβεια της συνολικής καθυστέρησης είναι πολύ μικρό.



Σχήμα 3.15 Απλοποιημένο μοντέλο του κυκλώματος Precharge-συγκράτησης

Η έξοδος κάθε stack συνδέεται στο δέντρο συνένωσης που εξάγει το τελικό αποτέλεσμα για ολόκληρη τη γραμμή. Έτσι στο παραπάνω RC μοντέλο προστίθενται και η αντίσταση του αγωγού που συνδέει την έξοδο του stack με την είσοδο της πρώτης πύλης του δέντρου, συνήθως μία NOR 2 ή 3 εισόδων. Στο Σχήμα 3.16

εμφανίζονται οι πύλες ενός τέτοιου δένδρου, όπως υλοποιούνται με CMOS transistors και με μπλε χρώμα το μονοπάτι που ενεργοποιείται όταν υπάρχει match σε ένα stack.

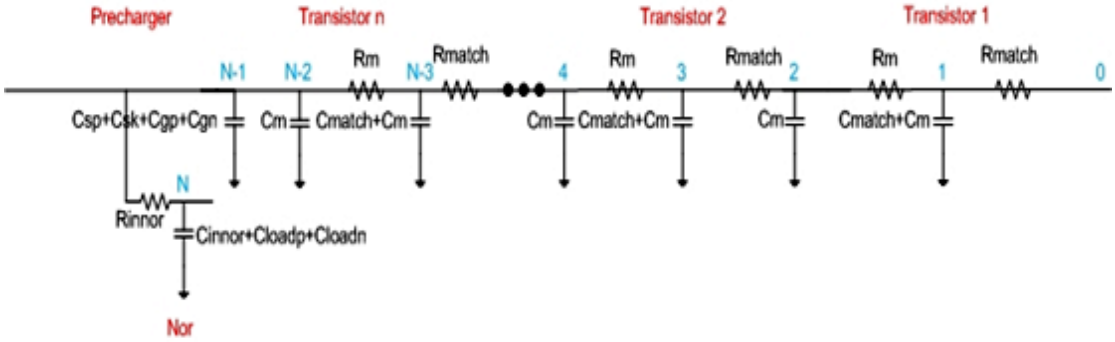


Σχήμα 3.16 Το μονοπάτι που ενεργοποιείται όταν υπάρχει match σε ένα stack

Συνολική Καθυστέρηση της NAND Match Line

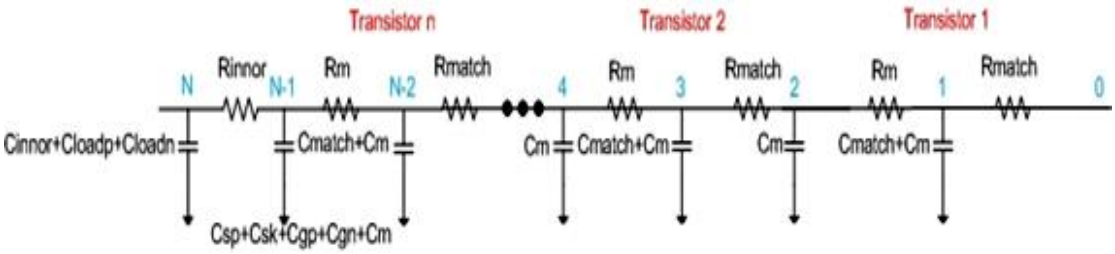
Η περίπτωση όπου όλα τα cell των stack κάνουν match, είναι αυτή που εμφανίζει τη μεγαλύτερη καθυστέρηση. Αυτό συμβαίνει διότι το μηδενικό δυναμικό πρέπει να διαδοθεί σε ολόκληρη τη matchline (μέσω του match transistor του κάθε cell που κάνει match) αποφορτίζοντας τη match line και τελικά οδηγώντας την σε δυναμικό ίσο με της γείωσης. Επιπλέον όπως προαναφέρθηκε, η ύπαρξη μηχανισμού συγκράτησης (Keeper) του υψηλού δυναμικού της matchline, καθώς και οι πύλες που ομαδοποιούν τα stacks, επιφέρουν επιπλέον καθυστέρηση στην αποφόρτιση της matchline, αφού ο μεν πρώτος συνεχίζει να την τροφοδοτεί με ρεύμα και οι δεύτερες εισάγουν αντίσταση και παρασιτική χωρητικότητα.

Στο επόμενο σχήμα (3.17) παρουσιάζεται το RC δικτύωμα μίας match line με βάθος stack n transistors.



Σχήμα 3.17 Το RC δικτύωμα μίας match line με stack depth n transistors

Στο παρακάτω σχήμα, εμφανίζεται το κύκλωμα απλοποιημένο.



Σχήμα 3.18 Το απλοποιημένο 3.17 σχήμα

Σύμφωνα με τον κανόνα του W. C. Elmore [14] η καθυστέρηση διάδοσης του σήματος από τον κόμβο μηδέν (0) στον ι-οστό κόμβο, κατά προσέγγιση είναι:

$$t_{pi} \approx \ln(2)\sum_{k=1}^{N} C_k R_{ik}$$

(Σχέση 3.5.1)

Όπου Rik είναι η αντίσταση του κοινού μονοπατιού για τους κόμβους i, k (πχ η αντίσταση που είναι κοινή για το μονοπάτι από τον κόμβο μηδέν στον κόμβο i και τον κόμβο μηδέν στον κόμβο k).

$$R_{ik} = \sum R_j \in [path(0 \to i) \cap (path(0 \to k))]$$

(Σχέση 3.5.2)

Θα πρέπει αρχικά να υπολογιστεί η αντίσταση των μονοπατιών από τον κόμβο μηδέν (0) προς όλους τους κόμβους (δηλαδή για όλες τις τιμές του K), που είναι κοινή με την αντίσταση του μονοπατιού από τον κόμβο μηδέν προς τον κόμβο N.

Με τη σχέση (3.5.2) και για τις τιμές i=N και K από 1 έως N έχουμε:

i=N, K=1: $R_{N1} = \sum R_j \in [path(0 \to N) \cap (path(0 \to 1))] = Rmatch$

i=N, K=2: $R_{N2} = \sum R_j \in [path(0 \to N) \cap (path(0 \to 2))] = Rmatch + Rm$

i=N, K=3: $R_{N3} = \sum R_j \in [path(0 \to N) \cap (path(0 \to 3))] = 2Rmatch + Rm$

i=N, K=4: $R_{N4} = \sum R_j \in [path(0 \to N) \cap (path(0 \to 4))] = 2Rmatch + 2Rm$

i=N, K=5: $R_{N5} = \sum R_j \in [path(0 \to N) \cap (path(0 \to 5))] = 3Rmatch + 2Rm$

i=N, K=6: $R_{N6} = \sum R_j \in [path(0 \to N) \cap (path(0 \to 6))] = 3Rmatch + 3Rm$

…….

i=N, K=N-2:

$$R_{N,N-2} = \sum R_j \in [path(0 \to N) \cap (path(0 \to N-2))] = (\frac{N-1}{2})Rmatch + (\frac{N-3}{2})Rm$$

i=N, K=N-1:

$$R_{N,N-1} = \sum R_j \in [path(0 \to N) \cap (path(0 \to N-1))] = (\frac{N-1}{2})Rmatch + (\frac{N-1}{2})Rm$$

i=N, K=N:

$$R_{N,N} = \sum R_j \in [path(0 \to N) \cap (path(0 \to N))] = (\frac{N-1}{2})Rmatch + (\frac{N-1}{2})Rm + Rinnor$$

Να σημειωθεί ότι ανεξαρτήτως του πλήθους n των cells (άρτιο ή περιττό), το πλήθος των κόμβων είναι πάντα περιττό. Αυτό οφείλεται στο γεγονός ότι το σύνολο των κόμβων των cells είναι πάντα άρτιο, αφού σε κάθε cell αναλογούν δύο κόμβοι, ενώ στο σύνολο αυτό προστίθεται ένας επιπλέον κόμβος (ο N-οστός). Έτσι το ο συνολικός αριθμός των κόμβων ισούται με N=2n+1.

Το πλήθος των R για κάθε τιμή του K δίδεται από την κλαδική συνάρτηση:

$$f(K) = \begin{cases} \frac{K+1}{2}Rmatch + \frac{K-1}{2}Rm, \forall K : περιττό \ \& \ K \neq N \\ \frac{K}{2}Rmatch + \frac{K}{2}Rm, \forall K : άρτιο \ \& \ K \neq N \\ \frac{K}{2}Rmatch + \frac{K}{2}Rm + Rinnor, για\_K = N \end{cases}$$

Κατά συνέπεια από τη σχέση (3.5.1) έχουμε:

$$t_{pN} \approx \ln(2)\sum_{k=1}^{N} C_k R_{N,k} \Leftrightarrow$$

$$t_{pN} \approx \ln(2)[(C_{match} + C_m)R_{N,1} + (C_m)R_{N,2} + (C_{match} + C_m)R_{N,3} + (C_m)R_{N,4} + ...$$
$$+(C_{match} + C_m)R_{N,N-2} + (C_{sp} + C_{sk} + C_{gp} + C_{gn} + C_m)R_{N,N-1} + (C_{loadp} + C_{loadn} + C_{innor})R_{N,N}] \Leftrightarrow$$

$$t_{pN} \approx \ln(2)[(C_{match} + C_m)R_{match} + (C_m)(R_{match} + R_m) + (C_{match} + C_m)(2R_{match} + R_m) +$$
$$(C_m)(2R_{match} + 2R_m) + ... + (C_{match} + C_m)(\frac{N-1}{2}R_{match} + \frac{N-3}{2}R_m) +$$
$$(C_{sp} + C_{sk} + C_{gp} + C_{gn} + C_m)(\frac{N-1}{2}R_{match} + \frac{N-1}{2}R_m) +$$
$$(C_{innor} + C_{loadp} + C_{loadn})(\frac{N-1}{2}R_{match} + \frac{N-1}{2}R_m + R_{innor})] \Leftrightarrow$$

$$t_{pN} \approx \ln(2)[(C_{match}+C_m)R_{match}+(C_m)(R_{match}+R_m)+(C_{match}+C_m)(2R_{match}+R_m)+$$
$$(C_m)(2R_{match}+2R_m)+.....+(C_{match}+C_m)(\frac{N-1}{2}R_{match}+\frac{N-1}{2}R_m-R_m)+$$
$$(C_{sp}+C_{sk}+C_{gp}+C_{gn}+C_m)(\frac{N-1}{2}R_{match}+\frac{N-1}{2}R_m)+$$
$$(C_{innor}+C_{loadp}+C_{loadn})(\frac{N-1}{2}R_{match}+\frac{N-1}{2}R_m+R_{innor})]$$

Όπως προαναφέρθηκε κάθε ένα από τα n cell περιέχει δύο από τους N-1 κόμβους, δηλαδή το πλήθος των cells είναι $n=\frac{N-1}{2}$.

Με αντικατάσταση στην παραπάνω σχέση η καθυστέρηση γίνεται:

$$t_{pN} \approx \ln(2)[(C_{match}+C_m)R_{match}+(C_m)(R_{match}+R_m)+(C_{match}+C_m)(2R_{match}+R_m)+$$
$$(C_m)(2R_{match}+2R_m)+.....+(C_{match}+C_m)(nR_{match}+nR_m-R_m)+$$
$$(C_{sp}+C_{sk}+C_{gp}+C_{gn}+C_m)(nR_{match}+nR_m)+$$
$$(C_{innor}+C_{loadp}+C_{loadn})(nR_{match}+nR_m+R_{innor})] \Leftrightarrow$$

$$t_{pN} \approx \ln(2)[R_{match}(C_{match}+2C_m)+R_m(C_{match}+2C_m)+2R_{match}(C_{match}+2C_m)+$$
$$2R_m(C_{match}+2C_m)+.....+nR_{match}(C_{match}+2C_m)+nR_m(C_{match}+2C_m)+$$
$$(nR_{match}+nR_m)(C_{sp}+C_{sk}+C_{gp}+C_{gn}+C_{innor}+C_{loadp}+C_{loadn})+$$
$$R_{innor}(C_{innor}+C_{loadp}+C_{loadn})-2R_m(C_{match}+C_m)] \Leftrightarrow$$

$$t_{pN} \approx \ln(2)[\frac{n(n+1)}{2}R_{match}(C_{match}+2C_m)+\frac{n(n+1)}{2}R_m(C_{match}+2C_m)+$$
$$(nR_{match}+nR_m)(C_{sp}+C_{sk}+C_{gp}+C_{gn}+C_{innor}+C_{loadp}+C_{loadn})+$$
$$R_{innor}(C_{innor}+C_{loadp}+C_{loadn})-2R_m(C_{match}+C_m)] \Leftrightarrow$$

Και η σταθερά χρόνου είναι :

$$\tau = n(R_{match}+R_m)[\frac{(n+1)}{2}(C_{match}+2C_m)+(C_{sp}+C_{sk}+C_{gp}+C_{gn}+C_{innor}+C_{loadp}+C_{loadn})]+$$
$$R_{innor}(C_{innor}+C_{loadp}+C_{loadn})-2R_m(C_{match}+C_m)$$

(Σχέση 3.5.3)

Καθυστέρηση του δέντρου συνένωσης

Το δέντρο συνένωσης υλοποιεί τη λογική μιας πύλης NOR πολλών εισόδων. Επειδή είναι πρακτικά αδύνατη η υλοποίηση μια πύλης NOR με περισσότερες από 3 εισόδους, χρησιμοποιείται ένα δέντρο από πύλες με μικρότερο αριθμό εισόδων: στο πρώτο επίπεδο χρησιμοποιούνται πύλες NOR 2 ή 3 εισόδων (ή και αντιστροφείς) στις οποίες συνδέονται οι έξοδοι των stacks. Στο δεύτερο επίπεδο πύλες NAND 2 ή 3 εισόδων συνενώνουν τις εξόδους των NOR του πρώτου επιπέδου και στο τρίτο επίπεδο ένας αντιστροφέας που δίνει την τελική έξοδο. Αν το δέντρο έχει πολλές εισόδους και δεν αρκεί ένα δέντρο με 3 επίπεδα, η παραπάνω οργάνωση συνεχίζεται: NOR στο 3ο επίπεδο, NAND στο 4ο κλπ.

Για τη μοντελοποίηση της καθυστέρησης του δέντρου συνένωσης αρκεί να υπολογιστεί η καθυστέρηση των πυλών που βρίσκονται στο κρίσιμο μονοπάτι. Η καθυστέρηση κάθε πύλης υπολογίζεται από το ισοδύναμο RC κύκλωμα, όπως έχει καταδειχθεί παραπάνω, μόνο που το ισοδύναμο RC κύκλωμα είναι απλούστερο αφού δεν υπάρχουν πολλαπλοί ενδιάμεσοι κόμβοι.

3.6. Διαχωρισμός των Cells της Tag Line σε Stacks

Η NAND CAM που προτάθηκε στο [8], έχει σχεδιαστεί σε επίπεδο transistor και συνεπώς έχει συγκεκριμένο μέγεθος. Η παρούσα εργασία έχει ως στόχο τη μοντελοποίηση NAND CAM χρησιμοποιώντας την οργάνωση του [8] αλλά για οποιοδήποτε μέγεθος CAM. Συνεπώς χρειάζεται μια κατάλληλη μέθοδος διαχωρισμού των cells της tag line σε stacks που να είναι βέλτιστη σύμφωνα με κριτήρια που θέτει ο χρήστης.

Λόγω του φαινομένου charge sharing το stack depth, επιλέχθηκε να μην είναι μεγαλύτερο από έξι cells. Αντίθετα το κάτω όριο του μεγέθους των stacks επιλέχθηκε να είναι τρία cells. Η επιλογή του κάτω ορίου των τριών cells, έγινε με σκοπό την αποφυγή της αυξημένης κατανάλωσης ενέργειας που παρουσιάζουν πολλά μικρού μεγέθους stacks των ενός ή δύο cells, καθώς και της μεγάλης επιφάνειας που

απαιτούν, λόγω των κυκλωμάτων precharger και διατήρησης που κάθε ένα από αυτά διαθέτει.

Ο αλγόριθμος διαχωρισμού των cells της tag line σε stacks επιλέγει να δημιουργεί stacks των οποίων το stack depth διαφέρει το πολύ κατά ένα cell, μέσα πάντα στο όριο των τριών έως έξι cells. Ο περιορισμός αυτός, βασίζεται στην πειραματική παρατήρηση ότι η επιπρόσθετη καθυστέρηση έστω και ενός stack που είναι μεγαλύτερο από τα υπόλοιπα stacks κατά δύο cells, είναι τόσο μεγάλη που δεν εξισορροπείται ακόμη και αν το μεγάλο stack συνδεθεί σε είσοδο του δέντρου συνένωσης που επιτρέπει την ταχύτερη δυνατή διέλευση.

Συγκεκριμένα, με χρήση του εργαλείου SPECTRE, έγινε προσομοίωση της καθυστέρησης ενός stack το οποίο ήταν μεγαλύτερο κατά δύο cells από τα υπόλοιπα και που συνδεόταν απευθείας στην τελευταία πύλη NAND του δένδρου συνένωσης, ώστε να έχει την ελάχιστη δυνατή καθυστέρηση μέσα από το δένδρο. Ακόμη και σε αυτή την περίπτωση, η καθυστέρηση αυτού του stack είναι πολύ μεγαλύτερη από αυτή των υπολοίπων stacks. Το συμπέρασμα αυτού του πειράματος είναι ότι το δέντρο συνένωσης και τα stacks πρέπει να είναι όσο ισορροπημένα γίνεται και κατά συνέπεια όλα τα stacks να είναι ίσου μεγέθους. Επειδή αυτό δεν είναι δυνατό για όλες τις πιθανές τιμές tag size, ο τελικός περιορισμός είναι τα stacks να έχουν μέγεθος που διαφέρει το πολύ κατά ένα cell. Ο αλγόριθμος συνδέει τα μεγαλύτερα stacks στις ταχύτερες εισόδους του δέντρου συνένωσης, ώστε οι διαφορές της συνολικής καθυστέρησης από κάθε stack να είναι οι ελάχιστες δυνατές.

Έχουν σχεδιαστεί πολλαπλές παραλλαγές δένδρων συνένωσης για αριθμό εισόδων από 2 μέχρι 15 stacks. Για κάθε αριθμό εισόδων, τα καλύτερα δέντρα συνένωσης προσομοιώθηκαν με το εργαλείο SPECTRE και επιλέχθηκε το ταχύτερο και πιο ισορροπημένο. Έτσι όταν ο αλγόριθμος διαχωρισμού καταλήγει σε μία πιθανή λύση, ο αριθμός των stacks της λύσης προσδιορίζει και το δέντρο συνένωσης που θα χρησιμοποιηθεί. Με 15 stacks καλύπτονται όλες οι πρακτικά χρήσιμες περιπτώσεις: 45 cells tag line για stacks των 3 cells μέχρι 90 cells tag line για stacks των 6 cells. Η μέθοδος σχεδίασης των δέντρων γενικεύεται για οποιοδήποτε μέγεθος tag line.

3.6.1. Θεωρητικό υπόβαθρο του αλγορίθμου διαχωρισμού

Πριν τη παρουσίαση του αλγόριθμου διαχωρισμού της tag line σε stacks, πρέπει να αποδειχθεί ότι για κάθε μέγεθος tag υπάρχει η δυνατότητα διαχωρισμού της tag line σε stacks με μεγέθη που διαφέρουν το πολύ κατά ένα cell. Συνεπώς αποδεικνύεται ότι η προτεινόμενη εφαρμογή μπορεί να έχει γενική εφαρμογή.

Δεδομένου ότι ο παραπάνω αλγόριθμος για τον διαχωρισμό των cells σε stacks, όπως προαναφέρθηκε, υλοποιεί την γραμμική εξίσωση αx+(1+α)y = C : α, C θετικοί ακέραιοι, μπορεί να δώσει ακέραια θετική λύση για stack μεγέθους α και tag line μεγέθους C cells.

Αυτό συμβαίνει διότι:

$$ax + (1+a)y = C \Leftrightarrow ax + (1+a)y = C(a+1-a) \Leftrightarrow ax + (1+a)y = (1+a)C - aC \Leftrightarrow$$
$$ax + aC = (1+a)C - (1+a)y \Leftrightarrow a(x+C) = (1+a)(C-y) \Leftrightarrow$$

$$\frac{a(x+C)}{1+a} = C - y \quad \text{, με το } \alpha+1>0 \text{ διότι } \alpha>0$$

(Σχέση 3.6.1)

Αν το y είναι ακέραιος, τότε το δεξί μέλος της (3.6.1) είναι ακέραιος. Επομένως πρέπει να είναι και το αριστερό μέλος ακέραιος. Όμως το α δεν διαιρείται ακριβώς με το 1+α (παραβλέπουμε την περίπτωση α=0 διότι τότε η εξίσωση είναι y=C). Συνεπώς, πρέπει το (x+C) να διαιρείται ακριβώς με το (1+α), δηλαδή πρέπει: $x + C = \lambda(1+a) \forall \lambda \in \mathbb{Z}$ ή ισοδύναμα:

$$x = \lambda(1+a) - C$$

(Σχέση 3.6.2)

Το x στην σχέση 3.6.2 είναι ακέραιος διότι λ, (1+α) και C είναι ακέραιοι.

Αντίστροφα αν επιλεγεί το $x = \lambda(1+a) - C, \forall \lambda \in \mathbb{Z}$, το αντίστοιχο y που δίδεται από την εξίσωση:

$$y = C - \frac{a(x+C)}{1+a}$$

(Σχέση 3.6.3)

θα είναι επίσης ακέραιος και λύση της εξίσωσης.

Όμως υπάρχει μία τουλάχιστον θετική (x,y) λύση:

Για να είναι το x θετικό, θα πρέπει το λ(1+α)-C να είναι θετικό δηλαδή:

$$\lambda(1+a) - C > 0 \Leftrightarrow$$

$$\lambda > \frac{C}{1+a}, \text{ με το } \alpha+1>0 \text{ διότι } \alpha>0$$

(Σχέση 3.6.4)

Με x θετικό άρα και με $\lambda > \frac{C}{1+a}$, το y είναι θετικό όταν το α είναι:

$$C - \frac{a(x+C)}{1+a} > 0 \Leftrightarrow$$

$$C > \frac{a(x+C)}{1+a} \Leftrightarrow \frac{C}{x+C} > \frac{a}{1+a} \Leftrightarrow \frac{x+C}{C} < \frac{1+a}{a} \Leftrightarrow$$

$$\frac{x+C}{C} < \frac{1}{a} + \frac{a}{a} \Leftrightarrow \frac{x+C}{C} - 1 < \frac{1}{a} \Leftrightarrow \frac{x+C}{C} - \frac{C}{C} < \frac{1}{a} \Leftrightarrow$$

$$\frac{x+C-C}{C} < \frac{1}{a} \Leftrightarrow \frac{x}{C} < \frac{1}{a} \Leftrightarrow \frac{C}{x} > \frac{a}{1} \Leftrightarrow$$

$$a < \frac{C}{x}, \text{ με το } x>0$$

(Σχέση 3.6.5)

3.6.2. Περιγραφή λειτουργίας του αλγορίθμου διαχωρισμού

Ο προτεινόμενος αλγόριθμος, δέχεται ως είσοδο το πλήθος των cells, C καθώς και το ελάχιστο και μέγιστο μέγεθος stack. Ξεκινώντας από το μικρότερο μέγεθος stack, ο αλγόριθμος προσπαθεί να χωρίσει το C σε τμήματα κατάλληλου μεγέθους, δηλαδή τμήματα που διαφέρουν το πολύ κατά 1 cell. Κάθε δυνατή λύση καταγράφεται γιατί δεν υπάρχει κάποιο κριτήριο επιλογής στην παρούσα φάση. Η καλύτερη λύση εξαρτάται από τις επιλογές του χρήστη (ελαχιστοποίηση ενέργειας, επιφάνειας, χρόνου προσπέλασης) και από τα υπόλοιπα κυκλώματα και συνεπώς, εξετάζεται αργότερα.

Ο αλγόριθμος στηρίζεται στην εξής λειτουργία: Για κάθε μέγεθος stack α υπολογίζεται το ακέραιο πηλίκο k της διαίρεσης του πλήθους C των cells της γραμμής με το α και το ακέραιο υπόλοιπο της διαίρεσης r. Αν το r είναι μηδέν, δηλαδή το σύνολο των C cells είναι πολλαπλάσιο του επιλεγμένου stack depth (α), επιτυγχάνεται ο διαχωρισμός σε k stacks του ιδίου μεγέθους και η λύση καταγράφεται. Αν υπάρχει ακέραιο υπόλοιπο r>0 και r<k, τότε το r μπορεί να διαμοιραστεί σε άλλα stacks αυξάνοντάς τα κατά 1 cell, ώστε τελικά να υπάρχουν r stacks μεγέθους α+1 και k-r stacks μεγέθους α.

Υπάρχουν αρκετές ειδικές περιπτώσεις που χρειάζονται διαφορετική αντιμετώπιση όπως για παράδειγμα την περίπτωση δημιουργίας x αριθμού stack με μέγεθος α και y αριθμού stack με μέγεθος (α + 1), όπου όμως το y=α και το x =α+1, δηλαδή όπου η γραμμική ισότητα που υλοποιεί ο αλγόριθμος είναι της μορφής α(α+1) + (α+1)α = C.

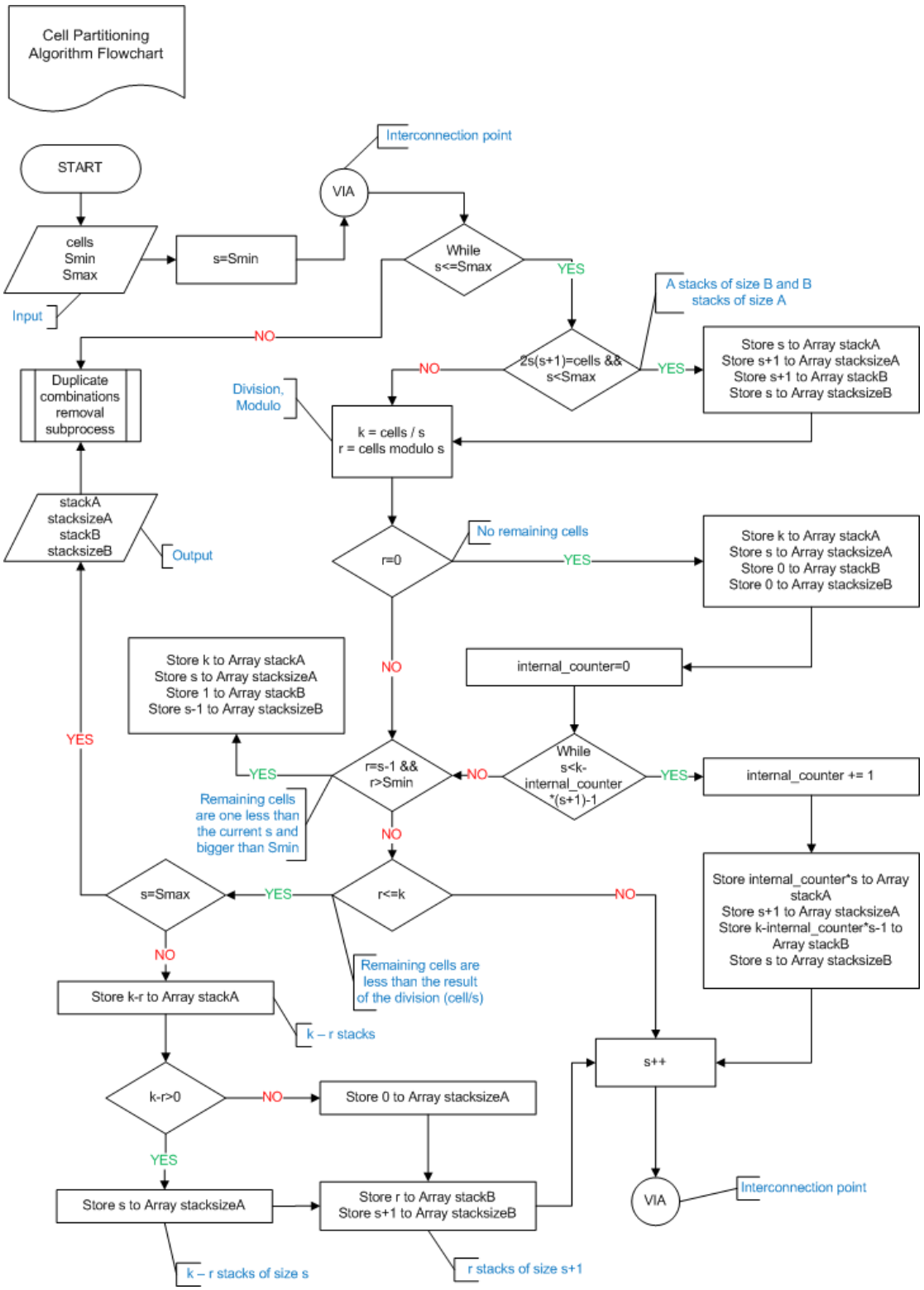
Μετά την εύρεση όλων των δυνατών διαχωρισμών εκτελείται υπορουτίνα του αλγορίθμου που εξετάζει και απομακρύνει διπλότυπες λύσεις.

Τέλος, για κάθε δυνατή λύση διαχωρισμού, γίνεται ταξινόμηση των stacks ανάλογα με το μέγεθός τους και ανάθεση (σύνδεση) του καθενός από αυτά σε μία είσοδο ενός κατάλληλου δένδρου συνένωσης, όπου όπως προαναφέρθηκε, στις ταχύτερες εισόδους του δέντρου συνένωσης συνδέονται τα μεγαλύτερα stacks.

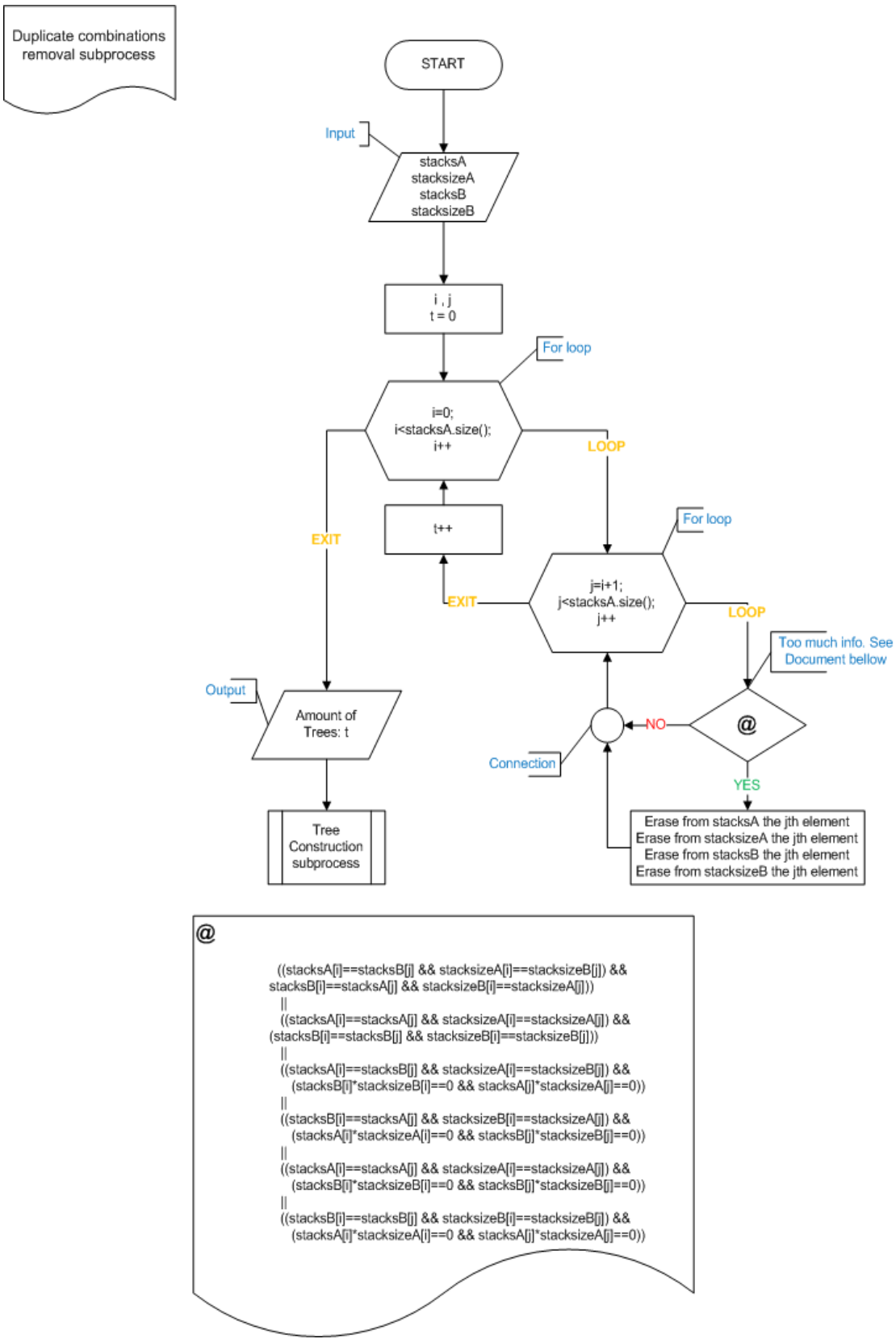
Μετά την ανάθεση, καταγράφεται το stack με τη μεγαλύτερη καθυστέρηση, δηλαδή το μεγαλύτερο stack που συνδέεται στην πιο “αργή” είσοδο του δένδρου συνένωσης.

Έτσι, για διαχωρισμό 23 bits σε 2 stacks των 4 bits και 5 stacks των 3 bits (4, 4, 3, 3, 3, 3, 3), στην ταχύτερη είσοδο του δένδρου συνένωσης συνδέεται ένα stack με 4 bits, στην αμέσως πιο αργή το δεύτερο stack με 4 bits και σε κάθε μία από τις υπόλοιπες πιο αργές εισόδους, από ένα stack με 3 bits. Το stack με τη μεγαλύτερη καθυστέρηση είναι μεγέθους 4 bits και συνδέθηκε στην δεύτερη ταχύτερη είσοδο του δένδρου.

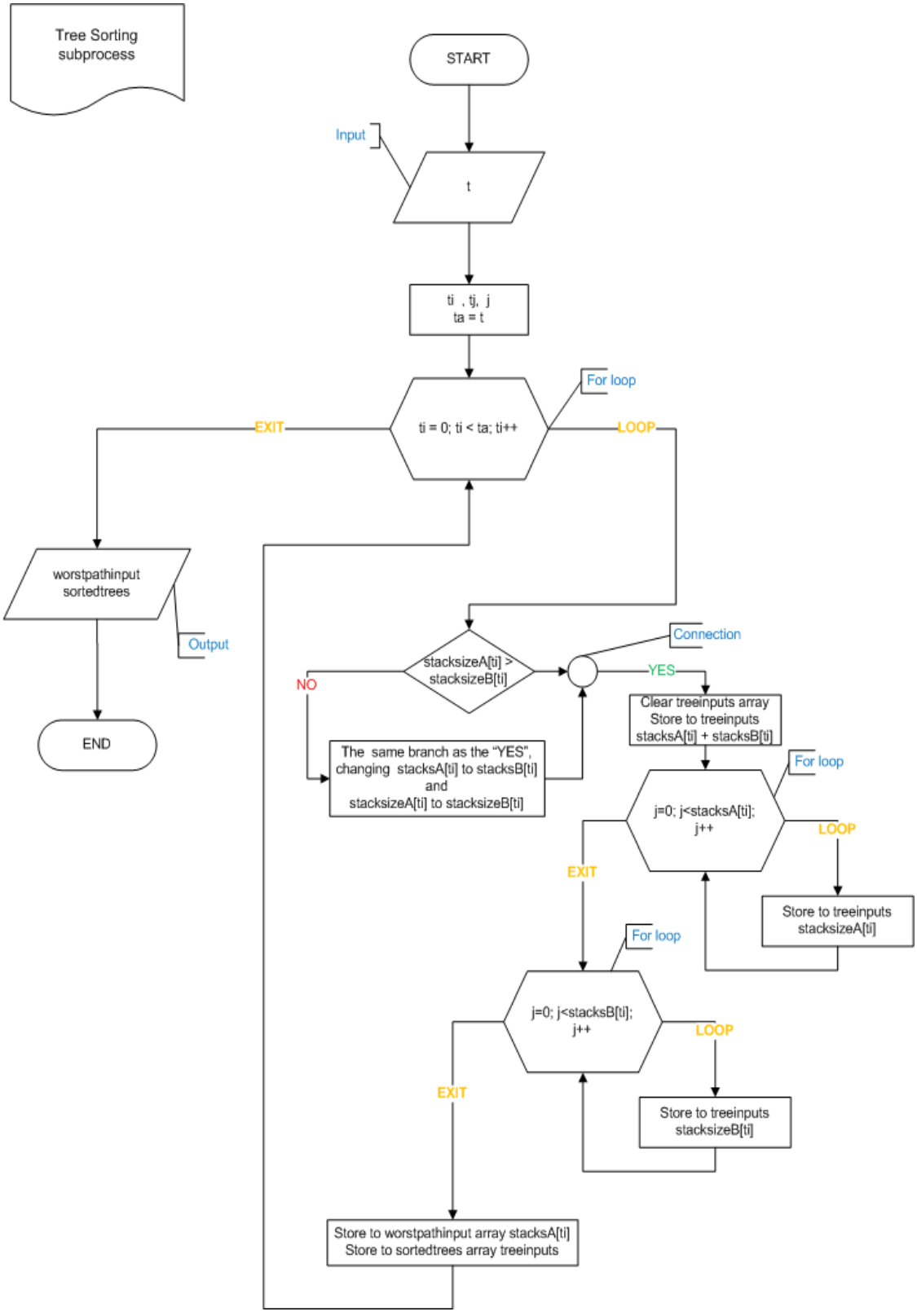
Όλες οι λεπτομέρειες περιγράφονται στα διαγράμματα ροής που ακολουθούν, όπου το πλήθος των cell της γραμμής C να εμφανίζεται ως “cell”, το μέγεθος του stack α ως “s” και το ελάχιστο και μέγιστο μέγεθος που μπορεί να πάρει το stack ως “Smin” και “Smax” αντίστοιχα.



Σχήμα 3.19 Διάγραμμα αλγορίθμου διαχωρισμού των cell σε Stacks



Σχήμα 3.20 Διάγραμμα υπορουτίνας εύρεσης και αφαίρεσης διπλοτύπων Stacks



Σχήμα 3.21 Διάγραμμα υπορουτίνας ταξινόμησης stacks ανάλογα με την καθυστέρηση που εμφανίζουν και υπολογισμού του stack με τη χειρότερη καθυστέρηση για κάθε tag line.

3.7. Δένδρα Συνένωσης

Όπως αναφέρθηκε προηγουμένως, υπάρχει ένα προεπιλεγμένο, βέλτιστο δέντρο συνένωσης για κάθε αριθμό stacks. Υπάρχουν όμως κάποιες λεπτομέρειες σχετικές με το δέντρο συνένωσης που εξαρτώνται από τα ακριβή μεγέθη των stacks:

- Οι αποστάσεις μεταξύ των πυλών που απαρτίζουν το δέντρο.
- Το κρίσιμο μονοπάτι του δέντρου που εξαρτάται από την είσοδο στην οποία συνδέεται το μεγαλύτερο stack.
- Τα μεγέθη των transistor των πυλών του δέντρου που εξαρτώνται από τη συνολική χωρητικότητα που οδηγεί κάθε πύλη και η έξοδος του δέντρου.

Για κάθε λύση που παράγει ο αλγόριθμος διαχωρισμού, παράγεται και βελτιστοποιείται το κατάλληλο δέντρο συνένωσης. Υπολογίζονται οι παρασιτικές χωρητικότητες των αγωγών που συνδέουν τις πύλες του δέντρου στο κρίσιμο μονοπάτι. Με βάση αυτές τις πληροφορίες υπολογίζονται τα βέλτιστα μεγέθη των transistor χρησιμοποιώντας τον αλγόριθμο unified logical effort [10], που είναι μια παραλλαγή του γνωστού logical effort που παίρνει υπόψη του και τις παρασιτικές αντιστάσεις και χωρητικότητες των αγωγών.

3.8. Υπολογισμοί καθυστέρησης

Το προτεινόμενο εργαλείο λογισμικού υπολογίζει την καθυστέρηση και το κύκλο προσπέλασης της CAM για αναζητήσεις. Η καθυστέρηση προσπέλασης είναι ο χρόνος που απαιτείται από τη στιγμή που δίνεται μία λέξη προς αναζήτηση στη CAM μέχρι να εξαχθεί το τελικό σήμα επιτυχίας ή αποτυχίας της αναζήτησης. Ο κύκλος προσπέλασης είναι ο χρόνος που απαιτείται μεταξύ διαδοχικών προσπελάσεων και είναι μεγαλύτερος από τη καθυστέρηση προσπέλασης γιατί χρειάζεται να ολοκληρωθεί η φάση προφόρτισης των searchlines και matchlines πριν να μπορέσει να γίνει νέα αναζήτηση. Η χειρότερη περίπτωση καθυστέρησης συμβαίνει όταν μια γραμμή ταιριάζει με τα δεδομένα αναζήτησης γιατί τότε θα πρέπει να εκφορτιστούν οι ML των stacks και το δέντρο συνένωσης να αλλάξει την έξοδό του σε λογικό ένα.

Στη NAND CAM η καθυστέρηση προσπέλασης περιλαμβάνει τη καθυστέρηση των αντιστροφέων που αποτελούν τους οδηγούς των searchlines, τη καθυστέρηση της

matchline του μεγαλύτερου stack και τη καθυστέρηση του δέντρου συνένωσης. Από τις παραπάνω συνιστώσες, η πρώτη είναι σταθερή και ανεξάρτητη του διαχωρισμού σε stacks. Οι άλλες δύο υπολογίζονται για κάθε πιθανό διαχωρισμό που εξετάζεται και κρατούνται μέχρι να επιλεγεί η βέλτιστη οργάνωση (που εξαρτάται και από τις άλλες παραμέτρους: εμβαδού και ενέργειας).

Έτσι αρχικά υπολογίζεται η καθυστέρηση των αντιστροφέων που αποτελούν τους SL drivers, εκτός από τον τελευταίο και η καθυστέρηση του τελευταίου αντιστροφέα που οδηγεί τη SL σε υψηλό δυναμικό. Για τους υπολογισμούς αυτούς χρησιμοποιούνται οι σταθερές χρόνου που προκύπτουν από τα ισοδύναμα RC κυκλώματα και οι σχέσεις (3.3.3) και (3.3.4) της παραγράφου 3.3.1 για ανερχόμενα και κατερχόμενα σήματα αντίστοιχα.

Η σταθερά χρόνου για την SL όπως υπολογίστηκε κατά την RC ανάλυση (Σχέση 3.4.3) :

$$\tau = (nR_d + n\frac{(n+1)}{2}R_{sl})(3C_t + C_{sl} + C_{mg}) + R_t(2C_t + C_{mg})$$

Στη συνέχεια, για κάθε διαχωρισμό, ξεκινάει ο υπολογισμός της καθυστέρησης για την ML του stack που βρίσκεται στο κρίσιμο μονοπάτι. Έτσι υπολογίζεται η καθυστέρηση της ML για να αποφορτιστεί (Σχέση 3.3.4) .

Η σταθερά χρόνου για την ML όπως υπολογίστηκε κατά την RC ανάλυση στο (Σχέση 3.5.3):

$$\tau = n(R_{match} + R_m)[\frac{(n+1)}{2}(C_{match} + 2C_m) + (C_{sp} + C_{sk} + C_{gp} + C_{gn} + C_{innor} + C_{loadp} + C_{loadn})] +$$
$$R_{innor}(C_{innor} + C_{loadp} + C_{loadn}) - 2R_m(C_{match} + C_m)$$

Τέλος, για κάθε διαχωρισμό, υπολογίζεται η καθυστέρηση του δέντρου συνένωσης από την είσοδό όπου συνδέεται το stack με το μεγαλύτερο μέγεθος. Αν υπάρχουν περισσότερα από ένα μεγάλα stacks, χρησιμοποιείται η πιο αργή είσοδος από αυτές στις οποίες συνδέονται τα μεγάλα stacks. Επειδή τα stacks θα είναι σε λογικό μηδέν και η τελική έξοδος σε λογικό ένα, κάθε επίπεδο του δέντρου θα έχει εναλλάξ τιμές

λογικού ένα και μηδέν: στο πρώτο επίπεδο που αποτελείται από πύλες NOR, οι τιμές θα γίνουν 1, στο δεύτερο επίπεδο που αποτελείται από πύλες NAND, οι τιμές θα γίνουν 0... Η καθυστέρηση του δέντρου υπολογίζεται προσθέτοντας τις καθυστερήσεις των επιμέρους πυλών που βρίσκονται στο μονοπάτι από την κατάλληλη είσοδο ως την έξοδο του δέντρου.

Ο υπολογισμός του Cycle Time γίνεται με την πρόσθεση στον Access Time, όπως αυτός περιγράφτηκε νωρίτερα, του χρόνου επαναφοράς δηλαδή του χρόνου που απαιτείται για να φτάσουν οι SL σε μηδενικό δυναμικό, οι ML των stacks σε υψηλό δυναμικό και η έξοδος του δένδρου συνένωσης των stacks σε μηδενικό δυναμικό.

Όπως παραπάνω η καθυστέρηση για τις SL είναι σταθερή ενώ αυτές των ML και του δέντρου εξαρτώνται από το διαχωρισμό της ML σε stacks. Ο χρόνος επαναφοράς (precharge time) είναι ελαφρώς μικρότερος από τη καθυστέρηση προσπέλασης (access time). Αυτό συμβαίνει γιατί στα stacks η σειρά των NMOS transistor (τα transistor της ML) δεν άγει και ο κόμβος εξόδου φορτίζεται από ένα PMOS (precharger), ενώ στις πύλες του δέντρου συνένωσης άγουν τα παράλληλα transistors. Για παράδειγμα στις πύλες NOR του πρώτου επιπέδου η έξοδός τους αλλάζει σε 0 οπότε οδηγείται από τα (2 ή 3 αντίστοιχα για NOR2 ή NOR3) παράλληλα NMOS.

Πρέπει να σημειωθεί ότι για τον υπολογισμό της καθυστέρησης των υπολοίπων λειτουργιών (ανάγνωση, εγγραφή) της NAND CAM, χρησιμοποιούνται οι υπάρχουσες μέθοδοι του CACTI καθώς οι λειτουργίες αυτές είναι όμοιες με αυτές της CAM με NOR Match Line που μοντελοποιεί το CACTI.

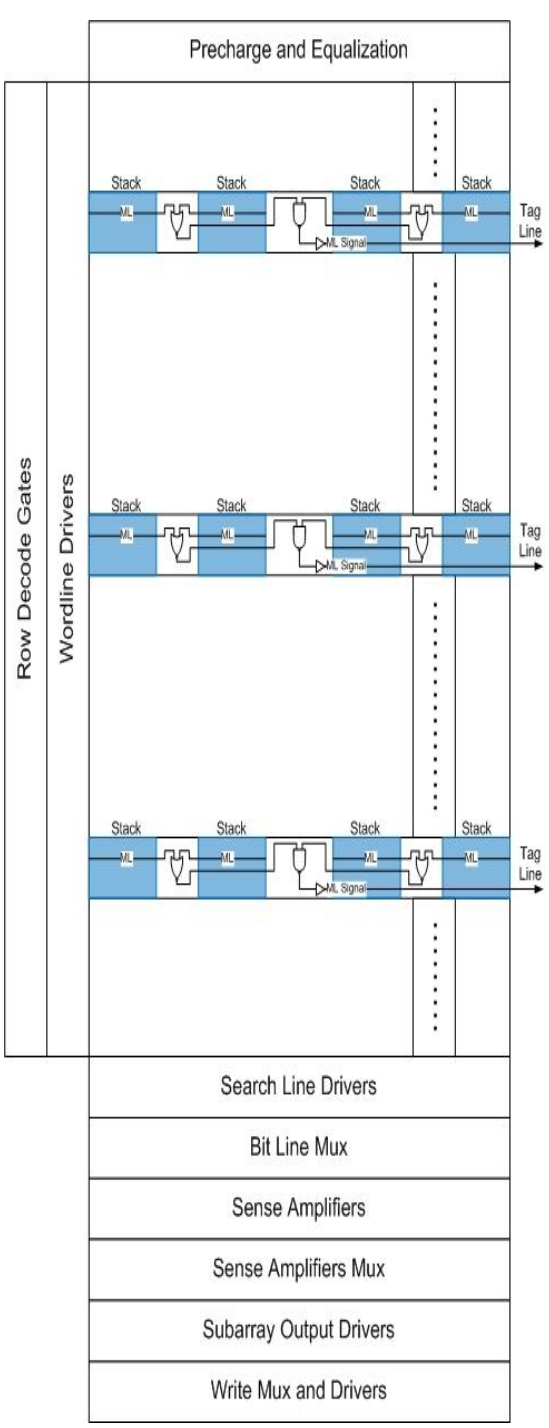
3.9. Υπολογισμός Επιφάνειας

Ο υπολογισμός της επιφάνειας μίας NAND CAM, γίνεται σταδιακά, καθώς συντίθεται από τις επιφάνειες που καταλαμβάνουν τα επιμέρους στοιχεία του.
Ο υπολογισμός των μεγεθών των επιμέρους στοιχείων του tag array (τμήματος ετικετών), στηρίζεται σε μεγέθη βασικών δομικών μονάδων, όπως transistors και αγωγών (wires) καθώς και των ελαχίστων απαιτουμένων μεταξύ τους αποστάσεων. Τα μεγέθη των βασικών δομικών μονάδων είναι κτήμα της επιστημονικής κοινότητας και παρέχονται από τα road maps των διαφόρων τεχνολογιών. Το μέγεθος κάθε βασικής δομικής μονάδας, πολλαπλασιάζεται με κατάλληλο συντελεστή (π.χ. το πλάτος του transistor). Το νέο μέγεθος που προκύπτει είναι τέτοιο που επιτρέπει στη μονάδα να ανταποκρίνεται σε συγκεκριμένα ηλεκτρικά χαρακτηριστικά και απαιτήσεις που υπαγορεύονται από τη θέση που αυτή κατέχει στο κύκλωμα.

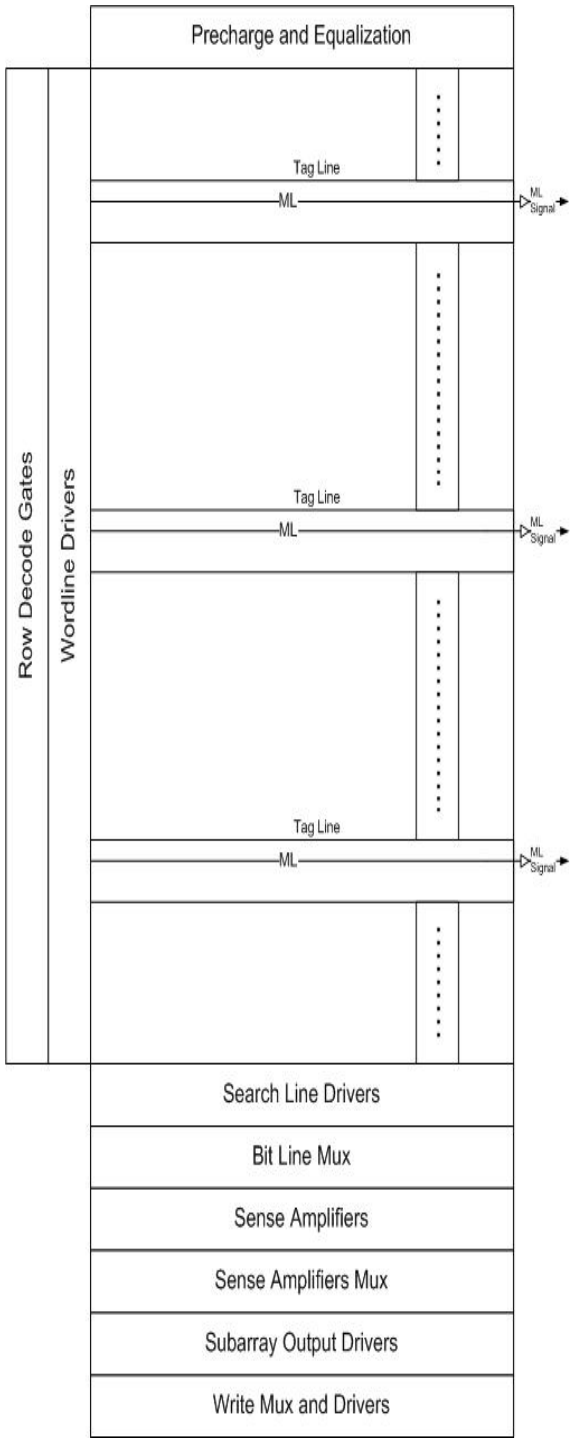
Το CACTI περιλαμβάνει υπορουτίνες υπολογισμού της επιφάνειας βασικών δομικών μονάδων και αποκωδικοποιητών, πολυπλεκτών κ.α. Η παρούσα εργασία εξέτασε τις διαφορές των οργανώσεων των NOR και NAND CAM και πρόσθεσε ή προσάρμοσε υπορουτίνες του CACTI ώστε να μπορεί να υπολογίσει με ακρίβεια την επιφάνεια μιας NAND CAM οποιουδήποτε μεγέθους.

Στα Σχήματα 3.22 και 3.23 εμφανίζονται, σε υψηλό επίπεδο αφαίρεσης, η δομή ενός array αποτελούμενο από NAND CAM cells και ενός array αποτελούμενο από NOR CAM cells αντίστοιχα μαζί με την περιφερειακή τους κυκλωμάτωση. Στο Σχήμα 3.22 είναι εμφανής ο διαχωρισμός της tag line σε stacks και ο τρόπος με τον οποίο το δένδρο συνένωσης κατανέμεται ανάμεσα στα stacks. Αντίθετα στο Σχήμα 3.23 δεν υπάρχουν stacks αλλά ούτε και δένδρο συνένωσης.

Επομένως οι αλλαγές που έγιναν στο CACTI αφορούν τις νέες διαστάσεις του CAM cell, το επιπλέον πλάτος του τελευταίου cell κάθε stack και το πλάτος των πυλών του δέντρου συνένωσης που μπαίνουν ανάμεσα σε stacks.



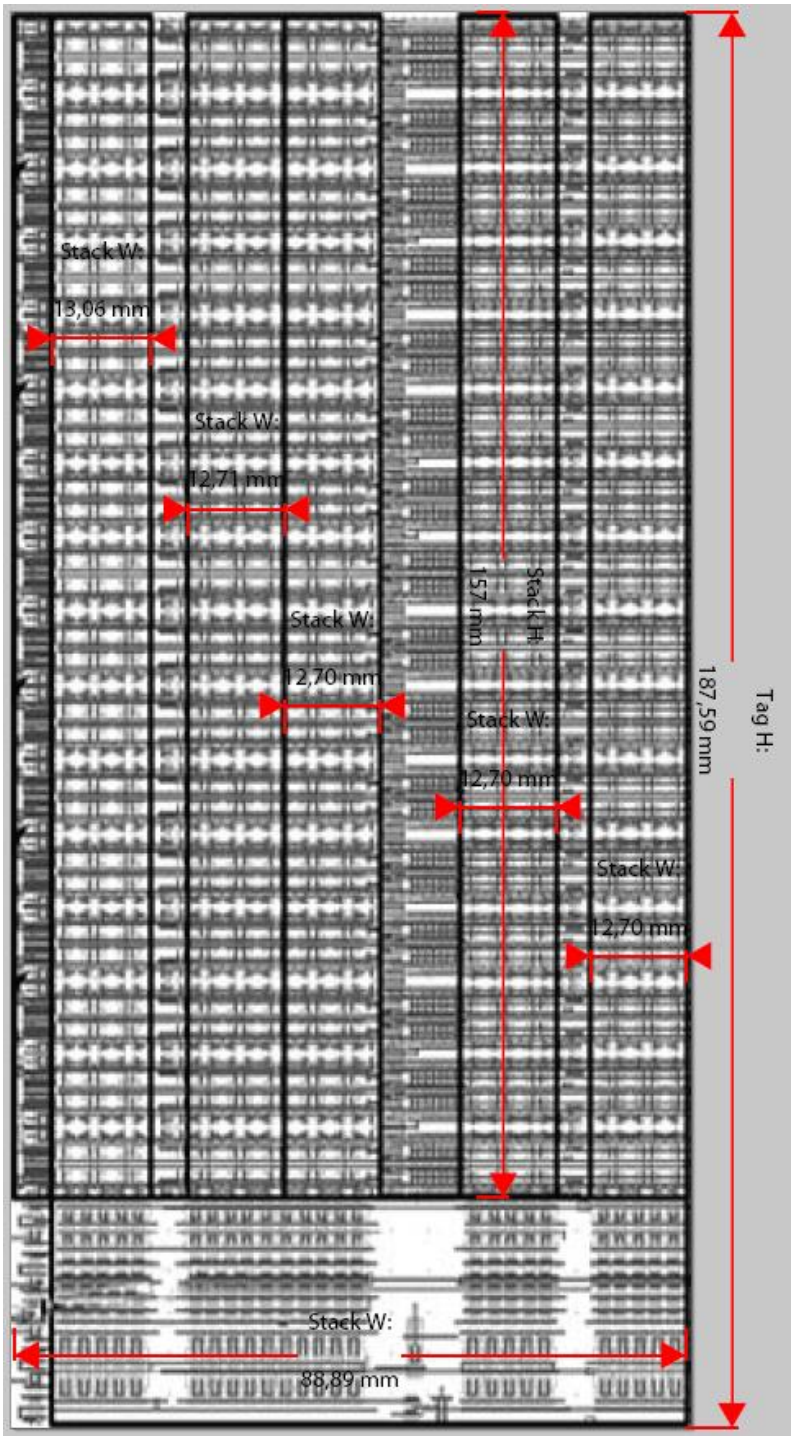
Σχήμα 3.22 NAND CAM Array



Σχήμα 3.23 NOR CAM Array

Επειδή το άρθρο [8], στο οποίο βασίστηκε αυτή η εργασία, δεν αναφέρει τις ακριβείς διαστάσεις του CAM cell, παρά μόνο την επιφάνειά του ($11.38\mu m^2$ σε τεχνολογία

130μm), χρειάστηκε να εκτιμηθούν κάποιες απαραίτητες διαστάσεις από τη μικροφωτογραφία μιας NAND CAM 32x25 που δίνεται στο άρθρο αυτό. Στο Σχήμα 3.24, εμφανίζεται σε κλίμακα η μικροφωτογραφία [8], με τα επιμέρους τμήματα και τις διαστάσεις τους να σημειώνονται με κόκκινο χρώμα.



Σχήμα 3.24 Μικροφωτογραφία [8]

Δεδομένου ότι υπάρχουν περισσότερα από ένα stack και η μέτρηση του πλάτους για κάθε ένα από αυτά διαφέρει, υπολογίζεται ο μέσος όρος: 12.774 μm. Όμως όπως αναφέρεται στο [8], κάθε stack αποτελείται από 5 CAM cells, με το πλάτος του top of the stack cell να είναι κατά 35% μεγαλύτερο από αυτό των υπολοίπων stack. Κατά συνέπεια το πλάτος του κάθε cell είναι 12.774 / 5.35 = 2.38766 μm, με το top of the stack να έχει πλάτος 3.223341 μm.

Η κάθε στήλη των CAM cells περιέχει 32 cells, με το συνολικό ύψος 157μm, συνεπώς το κάθε CAM cell έχει ύψος: 4.90625 μm. Το πλάτος της NAND3 του δένδρου συνένωσης, μαζί με τον Driver του Match Line σήματος για την οδήγηση της word line του SRAM τμήματος της cache είναι 10.58 μm.

Το συνολικό πλάτος του δένδρου συνένωσης, υπολογίστηκε αφαιρώντας από το συνολικό πλάτος του array το πλάτος των cells του stack και των περιφερειακών κυκλωμάτων. Από το συνολικό πλάτος του δένδρου, εκτιμήθηκε το πλάτος της κάθε πύλης του δένδρου συνένωσης να είναι ίσο με αυτό ενός CAM cell.

Οι παραπάνω διαστάσεις που ισχύουν για τεχνολογία 130μm, τέθηκαν στις κατάλληλες μεταβλητές του CACTI ώστε να μπορούν να μεταβάλλονται (scaling) ανάλογα με τη τεχνολογία που επιλέγει ο χρήστης.

Τέλος, η συνολική επιφάνεια του tag array είναι η επιφάνεια της γραμμής του array, πολλαπλασιαζόμενη με το σύνολο των γραμμών που απαρτίζουν το array καθώς επίσης και η επιφάνεια που απαιτείται από την περιφερειακή κυκλωμάτωση, της οποίας ο υπολογισμός γίνεται και αυτός με χρήση βασικών δομικών μονάδων (ο όρος περιφερειακή κυκλωμάτωση αναφέρεται στα Word line Drivers, Row Decoders, Prechargers, Equalizers, Search line Drivers, Bit line Mux, Sense Amplifiers, Sense Amplifiers Mux, Subarray Output Drivers, Write Mux, Write Drivers).

3.10. Υπολογισμός Δυναμικής Ενέργειας

Κατά τη διάρκεια μίας αναζήτησης η καταναλισκόμενη δυναμική ενέργεια είναι ίση με την ενέργεια που καταναλώνεται από τη λειτουργία των search lines και από την ενέργεια που καταναλώνεται για τη λειτουργία των matchlines. Θεωρούμε ότι η περιφερειακή κυκλωμάτωση, εκτός των οδηγών των searchlines, δεν καταναλώνει ενέργεια.

Για τον υπολογισμό της ενέργειας χρειάζεται να υπολογιστεί η χωρητικότητα που μεταβάλλεται, αφού η δυναμική ενέργεια δίνεται από τον τύπο $E = CV^2$ και η τάση παραμένει σταθερή. Τυπικά θα έπρεπε να υπολογιστεί μόνο η χωρητικότητα των κόμβων που μεταβάλλονται σε υψηλό δυναμικό σε έναν πλήρη κύκλο αναζήτησης. Όμως, επειδή στη CAM υπάρχει η φάση προετοιμασίας (precharge), όλοι κόμβοι που εκφορτίζονται κατά τη φάση αναζήτησης (π.χ. matchlines των stacks), φορτίζονται κατά τη φάση προετοιμασίας. Συνεπώς υπολογίζεται η χωρητικότητα όλων των κόμβων που αλλάζουν λογική τιμή στη φάση αναζήτησης ανεξάρτητα από τη κατεύθυνση της αλλαγής.

3.10.1. Υπολογισμός δυναμικής ενέργειας των search lines

Η ενέργεια που καταναλώνεται για την λειτουργία μίας SL είναι ίση με τη συνολική χωρητικότητα της SL (Cs) επί το τετράγωνο του δυναμικού της τροφοδοσίας.

$$Esl = Cs * Vdd^2$$

(Σχέση 3.10.1)

Η συνολική χωρητικότητα της κάθε SL είναι ίση με το άθροισμα των χωρητικοτήτων των επιμέρους στοιχείων που τη συνιστούν, όπως η χωρητικότητα εξόδου (intrinsic) του SL Driver (Cd), η παρασιτική χωρητικότητα του αγωγού (Cw), η χωρητικότητα των transmission (ή pass) gates κάθε cell (Ct) και η εσωτερική χωρητικότητα (Cx) των cell που αποθηκεύουν την ίδια τιμή με την τιμή αναζήτησης.

Η χωρητικότητα αυτή έχει υπολογιστεί για το σκοπό του υπολογισμού της καθυστέρησής στη παράγραφο 3.4, όπου στη χειρότερη περίπτωση όλα τα cells της στήλης αποθηκεύουν τη τιμή αναζήτησης. Για τον υπολογισμό της χωρητικότητας

της SL για κατανάλωση ενέργειας, η παραπάνω περίπτωση είναι εξαιρετικά σπάνια και η ενέργεια που υπολογίζεται θα ήταν μεγαλύτερη της τυπικής.

Λόγω έλλειψης ακριβέστερων πληροφοριών και επειδή το προτεινόμενο εργαλείο είναι γενικής χρήσης, η πιθανότητα ενός cell να κάνει match ορίστηκε να είναι ίση με 0.5: από τους 4 δυνατούς συνδυασμούς των 2 δυαδικών τιμών, της αποθηκευμένης τιμής και της τιμής αναζήτησης, στους μισούς συνδυασμούς οι τιμές είναι ίσες μεταξύ τους (00, 11). Επομένως, αν Cx είναι η εσωτερική χωρητικότητα ενός cell (2Cts + Cmg), συνολικά η χωρητικότητα μιας SL είναι:

$$Cs = Cw + Cd + Ct * \text{Tag rows} + Cx * \text{Tag rows} * 0.5$$

(Σχέση 3.10.2)

Παρόλο που υπάρχουν δύο SL ανά στήλη, μόνο η μία μεταβάλλεται γιατί είναι συμπληρωματικές και στη φάση προετοιμασίας και οι δύο εκφορτίζονται στο λογικό 0. Συνεπώς η συνολική καταναλισκόμενη ενέργεια όλων των SL του array είναι ίση με την ενέργεια που καταναλώνεται από μία SL επί το πλήθος των στηλών του array, δηλαδή το πλήθος των cells του tag line.

$$Esl\_total = Esl * \text{Tag Line size}$$

(Σχέση 3.10.3)

3.10.2. Υπολογισμός δυναμικής ενέργειας της match line

Η δυναμική ενέργεια που καταναλώνεται από τις matchlines υπολογίζεται με πιο σύνθετο τρόπο καθώς εξαρτάται από τα δεδομένα που αποθηκεύονται και αναζητούνται. Η μέθοδος που ακολουθείται στοχεύει στον υπολογισμό της μέσης ενέργειας αντί να υπολογίζονται οι μέγιστη και η ελάχιστη τιμή. Ακολουθείται η υπόθεση ότι σε κάθε κύκλο η κρυφή μνήμη ευστοχεί (hit rate 100%), δηλαδή μόνο μία γραμμή ταιριάζει ολόκληρη στα δεδομένα αναζήτησης ενώ όλες οι υπόλοιπες γραμμές αστοχούν (miss).

Η ενέργεια που καταναλώνεται από ένα stack που ταιριάζει, ισούται με την συνολική χωρητικότητά εξόδου του (Cml) επί το τετράγωνο της τροφοδοσίας. Από τη

παράγραφο 3.5 όπου υπολογίζονται όλες οι χωρητικότητες ενός stack:

$$Cml = Csp+Csk+Cgp+Cgn+Cinnor+Cloadn+Cloadp \quad \text{(Σχέση 3.10.4)}$$

Όμως, ένα stack καταναλώνει ενέργεια ακόμα και αν δεν αποφορτίζει την έξοδό του. Το φαινόμενο αυτό οφείλεται στην μερική αποφόρτιση (partial discharge) του stack που προκαλείται λόγω charge sharing [8]. Φορτίο που, λόγω διαμοιρασμού φορτίου (charge sharing), έχει περάσει στους ενδιάμεσους κόμβους, εκφορτίζεται χωρίς όλα τα cells του stack να κάνουν match. Το μέγιστο δυναμικό των ενδιάμεσων κόμβων ισούται με Vdd – Vt αφού φορτίζονται μέσω ενός NMOS transistor. Στο [8] υπολογίζεται πως κατά μέσο όρο η τιμή του δυναμικού είναι (Vdd – Vt) / 2 .

Σύμφωνα με το [8], η ισχύς (Pd) λόγω του φαινομένου μερικής αποφόρτισης για κάθε stack είναι:

$$Pd = (2Cm + Cmatch)\left\{\frac{\sum_{i=1}^{n-1} 2^i}{2^n}\right\}\left(\frac{Vdd - Vth}{2}\right)Vdd \bullet f \quad \text{(Σχέση 3.10.5)}$$

Όπου n το stack depth και f η συχνότητα λειτουργίας.

Κατά συνέπεια, η ενέργεια λόγω φαινομένου μερικής αποφόρτισης για κάθε stack είναι:

$$Epd = (2Cm + Cmatch)\left\{\frac{\sum_{i=1}^{n-1} 2^i}{2^n}\right\}\left(\frac{Vdd - Vth}{2}\right)Vdd \quad \text{(Σχέση 3.10.6)}$$

Για τη μοναδική γραμμή που ταιριάζει, η κατανάλωση ενέργειας αποτελείται από το άθροισμα των καταναλώσεων όλων των stacks και τη κατανάλωση του δέντρου συνένωσης. Επειδή όλοι οι κόμβοι του δέντρου συνένωσης αλλάζουν κατάσταση μόνο όταν όλα τα stacks ταιριάζουν, η ενέργεια που καταναλώνει είναι ίση με το άθροισμα των χωρητικοτήτων όλων των κόμβων του δέντρου επί το τετράγωνο της τροφοδοσίας. Η χωρητικότητα κάθε κόμβου περιλαμβάνει την εσωτερική χωρητικότητα της οδηγήτριας πύλης (intrinsic), την χωρητικότητα του αγωγού από την έξοδο της οδηγήτριας μέχρι την είσοδο της οδηγούμενης πύλης και τέλος την χωρητικότητα της εισόδου της οδηγούμενης πύλης.

Στις γραμμές που δεν ταιριάζουν απόλυτα, κάποια επιμέρους stacks τους ταιριάζουν, καταναλώνοντας ενέργεια. Για να υπολογιστεί η ενέργεια που καταναλώνουν αυτά τα stacks και τα αντίστοιχα τμήματα του δέντρου συνένωσης, υπολογίζεται η ισχύουσα χωρητικότητα (effective capacitance) ως το γινόμενο της χωρητικότητας ενός κόμβου με τη πιθανότητα ο κόμβος να αλλάξει κατάσταση. Αν η πιθανότητα ένα cell να ταιριάζει είναι ½, η πιθανότητα ενός stack των n cell να ταιριάζει είναι $1/2^n$, η ισχύουσα χωρητικότητα είναι Cml * $1/2^n$, και η ενέργεια Cml * $1/2^n$ * Vdd^2. Στην καταναλισκόμενη ενέργεια κάθε stack πρέπει να προστεθεί και η ενέργεια λόγω μερικής αποφόρτισης όπως υπολογίστηκε παραπάνω.

Παρόμοια με τα stacks υπολογίζονται και οι ισχύουσες χωρητικότητες των κόμβων του δέντρου συνένωσης ως το γινόμενο της χωρητικότητας του κόμβου με τη πιθανότητα να αλλάξει λογική τιμή. Στη φάση αναζήτησης, για να αλλάξει τιμή οποιαδήποτε πύλη του δέντρου, πρέπει να αλλάξουν τιμή όλες οι είσοδοί του. Για παράδειγμα, στο πρώτο επίπεδο του δέντρου οι πύλες NOR έχουν ως είσοδο τις εξόδους των stacks. Για να αλλάξει μια NOR σε λογικό 1 θα πρέπει όλα τα stack της εισόδου της να αλλάξουν τιμή (δηλαδή να ταιριάζουν). Συνεπώς, η πιθανότητα ενός κόμβου να αλλάξει τιμή είναι ίση με το γινόμενο των αντίστοιχων πιθανοτήτων όλων των εισόδων της πύλης που οδηγεί τον κόμβο.

3.11. Υλοποίηση του Μοντέλου NAND CAM σε Λογισμικό

Τα παραπάνω αναλυτικά μοντέλα υπολογισμού χρόνου προσπέλασης, κύκλου, επιφάνειας και δυναμικής ενέργειας υλοποιήθηκαν σε λογισμικό ώστε να εξεταστεί η ορθότητα και η ακρίβεια των προτεινόμενων μοντέλων. Ο αρχικός στόχος ήταν τα προτεινόμενα μοντέλα να ενταχθούν στο CACTI ώστε να μπορεί ο χρήστης να επιλέξει μεταξύ των υλοποιήσεων NAND και NOR για μνήμες CAM, είτε ως ανεξάρτητες μονάδες είτε ως τμήματα μιας κρυφής μνήμης. Δυστυχώς αυτό δεν κατέστη δυνατό διότι η δομή του CACTI δεν ακολουθεί τις αρχές του προγραμματισμού προσανατολισμένου σε αντικείμενα (OOP - object oriented programming). Έτσι η μεταβολή του υλοποιημένου κώδικα με σκοπό την προσθήκη και άλλων μοντέλων μνήμης στην ήδη υπάρχουσα υλοποίηση, καθίσταται εξαιρετικά δύσκολη. Για τον παραπάνω λόγο, το λογισμικό που υλοποιήθηκε υπολογίζει τα μέτρα επίδοσης μόνο για μνήμες CAM με οργάνωση NAND, χρησιμοποιώντας αρκετές υπορουτίνες από το CACTI.

Η υλοποίηση έγινε με τη γλώσσα προγραμματισμού C++, στο περιβάλλον Code Blocks (έκδοση 12.11) και με τον μεταγλωττιστή GNU gcc 4. Χρησιμοποιήθηκε ως βάση η έκδοση 6.5 του CACTI όπως περιλαμβάνεται στο McPAT 0.8 [12]. Στο κεφάλαιο 4 το προτεινόμενο λογισμικό αξιολογείται για την ακρίβεια των αποτελεσμάτων που δίνει, συγκρίνοντάς το με γνωστά από άλλες εργασίες αποτελέσματα καθώς και με αποτελέσματα που παρήχθησαν από προσομοιώσεις με το εργαλείο SPECTRE.

Για την εύρεση της βέλτιστης οργάνωσης μίας μνήμης, το CACTI δέχεται ως είσοδο ένα αριθμό παραμέτρων. Οι κύριες παράμετροι εισόδου που δέχεται είναι το μέγεθος της κρυφής μνήμης, το μέγεθος του block, η συσχετικότητα, η τεχνολογία και ο αριθμός ξεχωριστών τμημάτων της κρυφής μνήμης. Εκτός από τις προαναφερθείσες υπάρχει και ένας ακόμη μεγάλος αριθμός οργανωτικών παραμέτρων (των οποίων όμως παραλείπεται η αναφορά διότι είναι δευτερεύουσας σημασίας για τους σκοπούς της παρούσης εργασίας), πράγμα που καθιστά την χειροκίνητη εισαγωγή τους στο CACTI, μία επίπονη και χρονοβόρα διαδικασία. Για τον λόγο αυτό η εισαγωγή των παραμέτρων στο CACTI γίνεται με την χρήση ενός αρχείου ρυθμίσεων (configuration file).

Για την εισαγωγή των οργανωτικών παραμέτρων της προς προσομοίωση μνήμης, όμοια φιλοσοφία ακολουθήθηκε και στο προτεινόμενο λογισμικό. Έτσι, όπως ακριβώς και στο CACTI, στο προτεινόμενο εργαλείο λογισμικού, χρησιμοποιείται ένα configuration file στο οποίο καταγράφονται τα χαρακτηριστικά της προς ανάλυσης NAND CAM. Στη συνέχεια παρουσιάζεται ένα configuration file που χρησιμοποιείται στο προτεινόμενο λογισμικό.

```
# Type of memory
-cache type "nand_cam"

# Cache size
-size (bytes) 192

# Technology
-technology (u) 0.090

# Tag Size
-tag size (b) 24

#TAG STACK SIZE (bits)
-min stack size 3
-max stack size 6

# DESIGN OBJECTIVE
-design objective (weight delay, dynamic power, leakage power, cycle time, area)
0:0:0:100:0

# Ports
-read-write port 1
-search port 1

# Tag array cell type(itrs-hp,itrs-lstp,itrs-lop)
-Tag array cell type - "itrs-hp"

# Tag array peripheral type(itrs-hp,itrs-lstp,itrs-lop)
-Tag array peripheral type - "itrs-hp"

# Temperature (in K, 300-400 in steps of 10)
-operating temperature (K) 380
```

Ο χρήστης του προτεινόμενου λογισμικού, παραμετροποιώντας κατάλληλα το configuration file, έχει τη δυνατότητα να ορίσει το μέγεθος της CAM (ονομάζεται

παράμετρος Cache size, για συμβατότητα με το CACTI) σε bytes και το μέγεθος της εφαρμοζόμενης τεχνολογίας (παράμετρος Technology), το πλήθος των cells στην tag line (παράμετρος Tag Size), το ελάχιστο και το μέγιστο μέγεθος stack (παράμετρος TAG STACK SIZE) καθώς και το χαρακτηριστικό στο οποίο θα πραγματοποιηθεί βελτιστοποίηση (παράμετρος DESIGN OBJECTIVE). Τέλος, ο χρήστης έχει τη δυνατότητα να μεταβάλλει τον αριθμό των θυρών ανάγνωσης, εγγραφής και αναζήτησης, χαρακτηριστικά της τεχνολογίας (χαμηλής κατανάλωσης, υψηλής ταχύτητας, ...) σύμφωνα με το «χάρτη τεχνολογίας» ITRS (Road Map) καθώς και την θερμοκρασία λειτουργίας.

Η παράμετρος design objective δίδει τη δυνατότητα βελτιστοποίησης του χρόνου προσπέλασης, της δυναμικής ενέργειας, του χρόνου κύκλου και τέλος της απαιτούμενης επιφάνειας (η βελτιστοποίηση τις καταναλισκόμενης ενέργειας από διαρροές δεν εξετάστηκε στην παρούσα εργασία). Το ποσοστό βελτιστοποίησης για μία από τις προαναφερθείσες παραμέτρους, ορίζεται σε σχέση με αυτό των υπολοίπων. Έτσι για παράδειγμα η παραμετροποίηση «0:0:0:100:0» δηλώνει ότι απαιτείται βελτιστοποίηση μόνο του cycle time, η παραμετροποίηση «0:0:0:75:25» δηλώνει ότι απαιτείται βελτιστοποίηση του cycle time κατά 75% και της area κατά 25%, ενώ αν η παραμετροποίηση που ζητηθεί έχει τη μορφή «100:100:0:100:100» δηλώνει ότι απαιτείται βελτιστοποίηση των access time, dynamic energy,cycle time και area, κάτι που όμως δεν είναι δυνατό.

Το προτεινόμενο εργαλείο με την εισαγωγή των οργανωτικών παραμέτρων και την τιμή που αυτές κάθε φορά έχουν, εμφανίζει στην οθόνη τα υπολογισμένα μεγέθη για το Tag Array, των Height, Width, Area, Access Time, Cycle Time και Search Energy per Access

ΚΕΦΑΛΑΙΟ 4. ΑΞΙΟΛΟΓΗΣΗ ΤΟΥ ΜΟΝΤΕΛΟΥ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

Επειδή το άρθρο που πρότεινε την οργάνωση NAND CAM [8], την οποία μοντελοποιεί η εργασία αυτή, δεν παρέχει πλήρη αποτελέσματα μετρήσεων, για την αξιολόγηση του προτεινόμενου μοντέλου έγιναν μια σειρά από πειράματα. Για την αξιολόγηση της ακρίβειας των αποτελεσμάτων που δίνει το μοντέλο στο πεδίο του χρόνου (χρόνος προσπέλασης), προσομοιώθηκε το κρίσιμο μονοπάτι σε SPECTRE για μία συγκεκριμένου-μεγέθους CAM σε τεχνολογία 90nm. Για την αξιολόγηση της ακρίβειας των αποτελεσμάτων στο πεδίο της επιφάνειας, το λογισμικό μετατράπηκε ώστε να επιλέγει το δέντρο συνένωσης του [8] με το ίδιο φορτίο εξόδου, σε τεχνολογία 130nm. Επίσης παρουσιάζονται συγκρίσεις με NOR CAM μοντέλα από το CACTI και δίνονται αποτελέσματα των μέτρων αξιολόγησης για διάφορα μεγέθη μνημών NAND CAM.

Δεν κατέστη δυνατό να συγκριθούν τα αποτελέσματα κατανάλωσης ενέργειας. Το [8] δεν αναφέρει τελικές τιμές κατανάλωσης, αλλά εστιάζει στις διαφορές κατανάλωσης των επιμέρους τμημάτων (π.χ. ενέργεια λόγω διαμοιρασμού φορτίου στις match-lines) σε σχέση με αδημοσίευτη δουλειά του ενός συγγραφέα στην Intel. Η μέτρηση της μέσης κατανάλωσης σε προσομοιωτή κυκλωμάτων για τη NAND CAM είναι εξαιρετικά δύσκολη. Οι προσομοιωτές κυκλωμάτων χρειάζονται συγκεκριμένες τιμές δεδομένων και έτσι μπορούν να υπολογίσουν μέγιστες και ελάχιστες τιμές

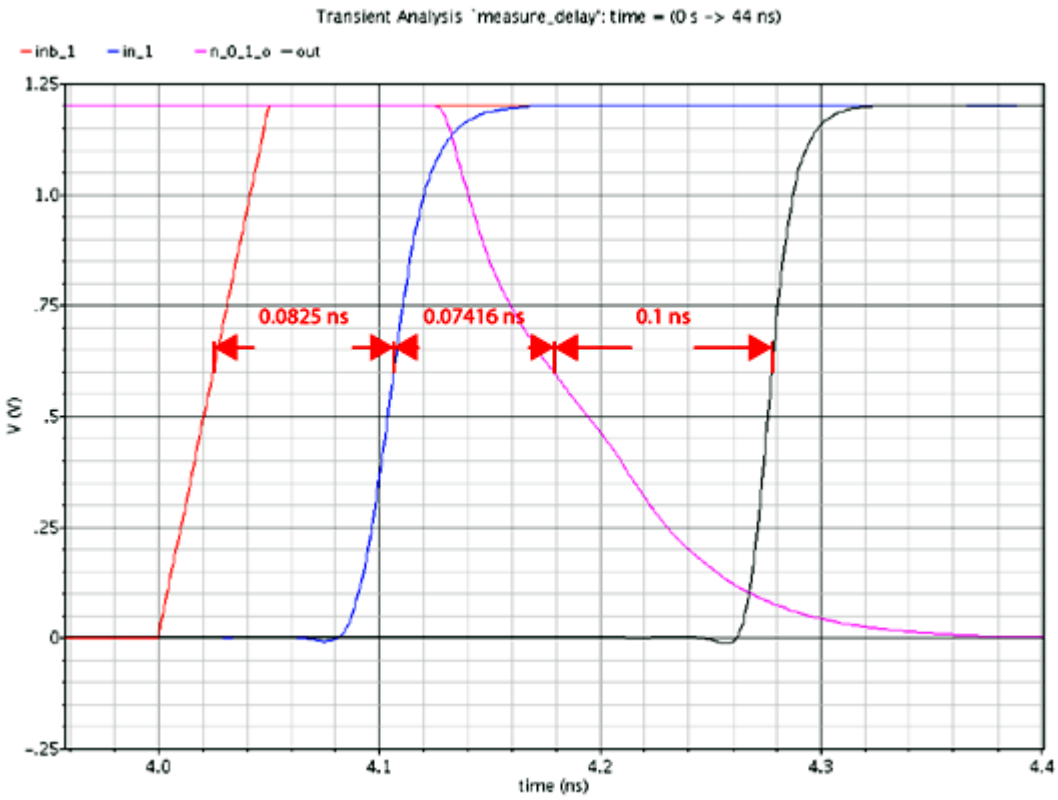
κατανάλωσης. Αντίθετα, όπως παρουσιάστηκε στη παράγραφο 3.10, η παρούσα εργασία υπολογίζει τη μέση τιμή κατανάλωσης διότι είναι αυτή που χρειάζονται οι αρχιτέκτονες υπολογιστών.

4.1. Σύγκριση με αποτελέσματα προσομοίωσης στο SPECTRE

Για την πιστοποίηση της ορθής λειτουργίας του προτεινόμενου λογισμικού στο πεδίο του χρόνου, έγινε σύγκριση των αποτελεσμάτων που αυτό έδωσε, με τα αποτελέσματα προσομοίωσης που εξήχθησαν από το εργαλείο SPECTRE. Η μνήμη CAM που προσομοιώθηκε και στα δύο εργαλεία είναι μεγέθους 100 Bytes με μέγεθος tag line 25 cells διαχωρισμένη σε 5 stacks Το μέγεθος της τεχνολογίας και στις δύο περιπτώσεις είναι 90nm και έχει προσομοιωθεί φορτίο 256 SRAM cell στην έξοδο της κάθε matchline.

Στο Σχήμα 4.1 αποτυπώνονται τα αποτελέσματα που παρήχθησαν από το εργαλείο SPECTRE. Συγκεκριμένα, εμφανίζεται η μεταβολή της τάσης της match line ενός stack που κάνει match (ροζ καμπύλη), της εισόδου του searchline driver (κόκκινη καμπύλη), της searchline (μπλε καμπύλη) και της τελικής εξόδου (μαύρη καμπύλη).

Οι μετρήσεις της καθυστέρησης έγιναν στα 0.6 Volts (στο μισό της μέγιστης τάσης) αφού πρόκειται για αυξανόμενη τάση εισόδου (rising input), με το b (το κλάσμα του δυναμικού της εισόδου, κατά το οποίο η έξοδος αλλάζει) να είναι ίσο με 0.5, όπως προαναφέρθηκε στην παράγραφο 3.3.1.



Σχήμα 4.1 Αποτελέσματα από SPECTRE

Σύμφωνα με τις μετρήσεις που έγιναν στο γράφημα του Σχήματος 4.1, από τη στιγμή που οι είσοδοι των searchline drivers είναι στα 0.6V, μέχρι τη στιγμή όπου οι searchlines (δηλαδή οι έξοδοι των SL drivers) θα φορτιστούν και αυτές στα 0.6 V είναι ίση με 0.0825ns.

Αντίστοιχα, εμφανίζεται καθυστέρηση 0.07416ns, από το σημείο όπου οι SL βρίσκονται στα 0.6V έως τη στιγμή που η matchline ενός stack που κάνει match, αποφορτιστεί στα 0.6V (η ML ήταν ήδη προφορτισμένη από το στάδιο προετοιμασίας). Τέλος, η καθυστέρηση από τη στιγμή όπου η matchline του stack είναι στα 0.6V μέχρι η έξοδος της ιεραρχικής ML (με το φορτίο των 256 SRAM cells) να φτάσει και αυτή το ίδιο δυναμικό, ισούται με 0,1ns. Η συνολική καθυστέρηση της διαδικασίας αναζήτησης ισούται με το άθροισμα των επιμέρους καθυστερήσεων: 0.0825 + 0.0741 + 0.1 = 0.2566ns

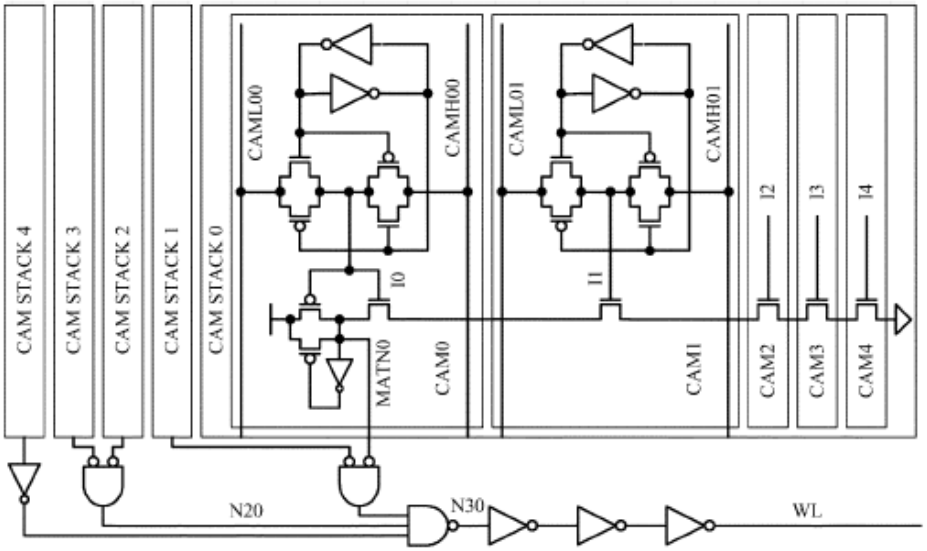
Η καθυστέρηση που υπολογίστηκε για την ίδια CAM από το προτεινόμενο λογισμικό, ισούται με 0.2651ns με τη διαφορά μεταξύ αυτής και της καθυστέρησης που υπολογίστηκε με SPECTRE να είναι ίση με 0.0085ns (3.31%).

Αντίστοιχη με την ποσοστιαία διαφορά των αποτελεσμάτων των υπολογισμών που έγιναν με το προτεινόμενο λογισμικό και προσομοιώσεων που έγιναν με το SPECTRE, παρατηρήθηκε και κατά τη διαδικασία πιστοποίησης της ορθής λειτουργίας του CACTI. Στο [9] περιγράφεται η διαδικασία πιστοποίησης, κατά την οποία συγκρίνονται αποτελέσματα υπολογισμών του CACTI, με αποτελέσματα από προσομοιώσεις που έγιναν με SPICE. Έτσι για διάφορες οργανώσεις και μεγέθη κρυφής μνήμης, η σύγκριση των αποτελεσμάτων που υπολογίστηκαν από το CACTI και των προσομοιώσεων που έγιναν με SPICE, έδωσε μέση ποσοστιαία διαφορά γύρω στο 6%.

Αν και η παραπάνω αξιολόγηση θα μπορούσε να θεωρηθεί αδόκιμη, καθώς τα ποσοστά που προαναφέρθηκαν δεν εξήχθησαν από την ίδια ακριβώς διαδικασία σύγκρισης, παρόλα αυτά καταδεικνύει ότι το προτεινόμενο λογισμικό παρέχει αρκετά μεγάλη ακρίβεια στους υπολογισμούς του, σε επίπεδα αυτής του CACTI. Αφού το CACTI θεωρείται ότι παρέχει ικανοποιητική ακρίβεια για αρχιτέκτονες υπολογιστών, τότε και το προτεινόμενο λογισμικό, που είναι αντίστοιχης ακρίβειας, θα είναι ένα χρήσιμο εργαλείο.

4.2. Σύγκριση με NAND CAM [8]

Στο [8], εξετάζεται μία NAND CAM με tag line μεγέθους 25 bits και stack depth 5 bits. Τα 5 stacks, συνενώνονται, με το δένδρο συνένωσης, σε μία ιεραρχική match line. Όπως φαίνεται στο Σχήμα 4.2, οι έξοδοι των τεσσάρων πρώτων stacks συνενώνονται ανά δύο σε μία NOR2 πύλη, ενώ η έξοδος του τελευταίου σε έναν αντιστροφέα. Η έξοδος του αντιστροφέα και των δύο NOR2 συνδυάζονται με μία NAND3.



Σχήμα 4.2 NAND CAM των 25 bits, με stack depth 5 bit

Για τεχνολογία στα 130 nm, σε μία τυπική process corner και τάση τροφοδοσίας Vdd = 1.2 V, το κύκλωμα λειτουργεί σε συχνότητα οριακά μεγαλύτερη του 1 GHz [8]. Για την δυναμική ενέργεια που καταναλώνεται σε κάθε αναζήτηση, δεν υπάρχουν τελικά δεδομένα στο [8] παρά μόνο συγκρίσεις συγκεκριμένων τμημάτων της CAM με προηγούμενη αδημοσίευτη εργασία του ενός συγγραφέα στην Intel.

Ιδανικά, το προτεινόμενο μοντέλο θα πρέπει να παράγει παρόμοια αποτελέσματα με τα αποτελέσματα που υπάρχουν στο [8] για τις ίδιες οργανωτικές παραμέτρους μίας NAND CAM. Δεδομένου ότι στο [8] το NAND CAM τμήμα ετικετών οδηγεί και SRAM τμήμα μεγέθους 256 cells, στο προτεινόμενο λογισμικό έγινε προσαρμογή στους οδηγούς των τελικών matchlines για να ληφθεί υπόψη και το φορτίο.

Στον παρακάτω πίνακα αποτυπώνονται τα μεγέθη (ύψος, πλάτος, επιφάνεια, χρόνος προσπέλασης και χρόνος κύκλου) μίας NAND CAM μεγέθους 100 Bytes σε τεχνολογία 130nm, όπως αυτά αναφέρονται στο [8] (με πράσινο χρώμα) και όπως υπολογίστηκαν από το προτεινόμενο λογισμικό (με μπλε χρώμα). Για την δυναμική ενέργεια που καταναλώνεται σε κάθε αναζήτηση, δεν υπάρχουν δεδομένα στο [8].

Πίνακας 4.1 Σύγκριση NAND CAM. Υπολογισμοί με πράσινο χρώμα από CACTI και με μπλε από το προτεινόμενο λογισμικό

NAND CAM	**Tag Line** (bits)	**Tag Rows**	**Height** (μm)	**Width** (μm)	**Area** ($μm^2$)	**Access Time** (ns)	**Cycle Time** (ns)	**Search Energy / Access** (nJ)
[8]	25	32	188.0000	90.0000	16920	0.500000	1.000000	-
Soft ware	25	32	196.1610	90.1141	17676.9	0.487466	0.994732	0.135007
Percentage Difference from [8]			4.3409%	0.1267%	4.4734%	-2.5068%	-0.5268%	-

(Το πλήθος των γραμμών της cache - Tag Rows - υπολογίζεται από τον μέγεθος της κρυφής μνήμης πολλαπλασιασμένο με 8 bits και διαιρεμένο με το μέγεθος της tag line)

Είναι εύκολα παρατηρήσιμο, ότι τα μεγέθη που υπολογίστηκαν με το προτεινόμενο λογισμικό βρίσκονται σε πλήρη αντιστοιχία με τα μεγέθη του [8], με τη μεγαλύτερη διαφορά (4.4734%) να εμφανίζεται στον υπολογισμό της επιφάνειας, κάτι που λόγω των χειροκίνητων μετρήσεων της μικροφωτογραφίας (Παράγραφος 3.9), εκτιμάται πως βρίσκεται σε αποδεκτά όρια. Επιπλέον για τον υπολογισμό των διαστάσεων των περιφερειακών κυκλωμάτων χρησιμοποιήθηκαν οι μέθοδοι του CACTI αντί για μετρήσεις από το [8]. Αυτή η επιλογή έγινε ώστε να μπορεί μελλοντικά να ενταχθεί το προτεινόμενο λογισμικό ως τμήμα του CACTI.

Η διαφορά των υπολογισμένων από το προτεινόμενο λογισμικό μεγεθών χρόνου (Access Time και Cycle Time), σε σχέση με αυτά που παρουσιάζονται στο [8], κρίνεται ως αμελητέα, δεδομένου ότι δεν ξεπερνά το 2.5%. Επιπλέον, η διαφορά αυτή ενδέχεται να είναι και μικρότερη του 2,5% αφού στο [8] δεν αναφέρονται με ακρίβεια τα μεγέθη αλλά ότι η συχνότητα λειτουργίας ξεπερνάει οριακά το 1 GHz, πληροφορία από την οποία εξήχθησαν τα Cycle Time = 1ns και το Access Time = 0.5ns .

4.3. Σύγκριση με αποτελέσματα μοντέλου NOR CAM από το CACTI

Με σκοπό την αξιολόγηση των επιδόσεων μίας CAM με ιεραρχική NAND Match Line, κρίθηκε σκόπιμο να γίνει σύγκριση των αποτελεσμάτων που υπολογίστηκαν από το προτεινόμενο λογισμικό, με αποτελέσματα που υπολογίστηκαν από το CACTI για μία NOR CAM. Η επιλογή της NOR CAM για την σύγκριση έγινε δεδομένου ότι η μόνη διαφορά της με την NAND CAM εντοπίζεται στην οργάνωση της Match Line.

Στον παρακάτω πίνακα αποτυπώνονται τα μεγέθη (ύψος, πλάτος, επιφάνεια, χρόνος προσπέλασης και δυναμική ενέργεια ανά αναζήτηση) μίας NAND CAM (με μπλε χρώμα) και μίας NOR CAM (με κόκκινο χρώμα) σε τεχνολογίες 90nm και 130nm. Και στις δύο τεχνολογίες, το μέγεθος των cache είναι 100 Bytes, με μέγεθος tag line στα 25 bits, ορίζοντας 32 tag lines στο array.

Πίνακας 4.2 Σύγκριση NAND CAM (μπλε χρώμα) με NOR CAM (κόκκινο χρώμα) σε τεχνολογίες 90 και 130 nm

ML Type	**Tech** (nm)	**Height** (μm)	**Width** (μm)	**Area** (μm^2)	**Access Time** (ns)	**Cycle Time** (ns)	**Search Energy / Access** (nJ)
NAND	90	137.6060	62.7390	8633	0.27948	0.55759	0.0061311
NOR	90	129.7600	60.4150	7839	0.27489	0.55442	0.0075920
Percentage Difference		6.0465%	3.8467%	10.1288%	1.6697%	0.5717%	-19.2426%
NAND	130	196.1610	90.1140	17677	0.51328	1.03037	0.0135007
NOR	130	185.4310	87.2130	16172	0.46081	0.96046	0.0157441
Percentage Difference		5.7865%	3.3263%	9.3062%	11.3864%	7.2788%	-14.2491%

Παρατηρώντας τον Πίνακα 4.2, είναι εύκολο να διαπιστώσει κανείς πως και στις δύο τεχνολογίες η επιφάνεια που απαιτείται για τη NAND CAM είναι μεγαλύτερη κατά περίπου 10% από αυτή που απαιτείται για την NOR CAM. Το γεγονός αυτό οφείλεται στο κύκλωμα του δένδρου αναζήτησης που μόνο η NAND CAM διαθέτει.

Επιπλέον, είναι φανερή η διαφορά στο πεδίο του χρόνου και για τις δύο τεχνολογίες. Η NOR CAM υπερέχει της NAND ελαφρώς σε Access και Cycle Time, κάτι που επιβεβαιώνει την επικρατούσα άποψη ότι η NOR CAM είναι ταχύτερη.

Τέλος στο πεδίο της δυναμικής ενέργειας η NAND CAM υπερέχει κατά μέσο όρο περίπου 16%. Η υπεροχή της NAND έναντι της NOR CAM οφείλεται στο γεγονός ότι σε περίπτωση match σε NAND CAM αποφορτίζεται μόνο μία match line (αυτή της γραμμής που περιέχει τα δεδομένα που ταιριάζουν), σε αντίθεση με την NOR όπου αποφορτίζονται όλες εκτός αυτής που ταιριάζει. Παρατηρείται επίσης, πως καθώς η τεχνολογία βελτιώνεται, η διαφορά στην καταναλισκόμενη ενέργεια αυξάνεται υπέρ της NAND CAM.

4.4. Case Studies NAND CAM με το προτεινόμενο λογισμικό

Στη συνέχεια περιγράφονται περιπτώσεις χρήσης για διάφορα μεγέθη NAND CAM με σταθερό ύψος (64 γραμμές – Tag Rows) και για διαφορετικά μεγέθη Tag line σε τεχνολογία 130 nm. Συγκεκριμένα στους τρείς πίνακες που ακολουθούν εμφανίζονται τα αποτελέσματα για Cycle Time, Dynamic Energy και Area με κατάλληλη τιμή στο configuration file (Παράμετρος Design Objective) για βελτιστοποιημένο Cycle Time στον Πίνακα 4.3, για βελτιστοποιημένη Energy στον Πίνακα 4.4 και τέλος στον Πίνακα 4.5, για βελτιστοποιημένη την Area.

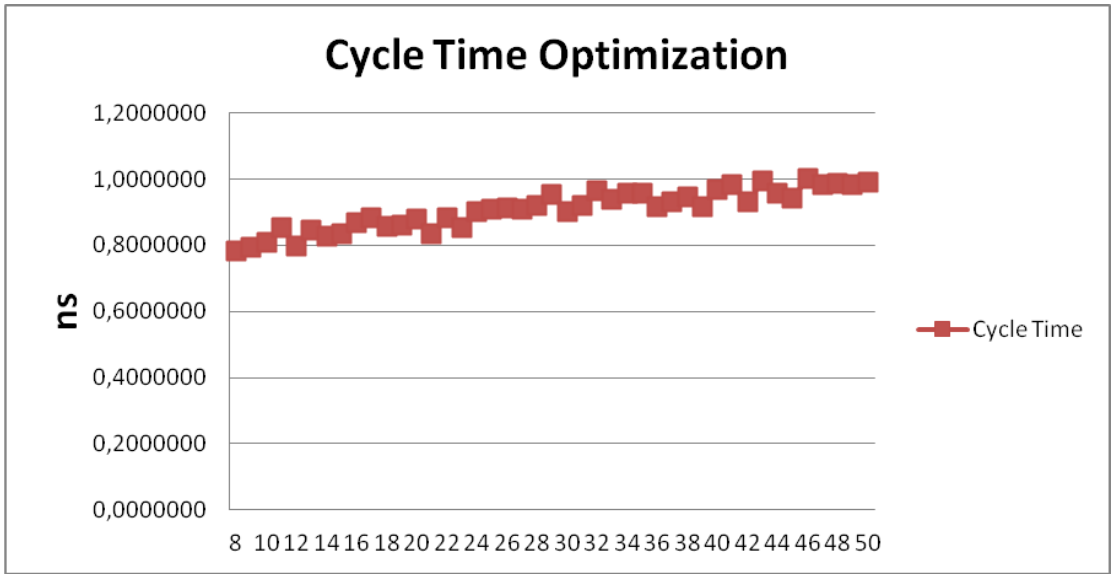
Πίνακας 4.3 Αποτελέσματα από το προτεινόμενο λογισμικό με βελτιστοποιημένο Cycle Time

Capacity(Bytes)	Tag size(bits)	Access time(ns)	Cycle time(ns)	Search energy / access(nJ)	Total area(mm^2)
64	8	0,3658030	0,7836840	0,0093534	0,0114923
72	9	0,3689230	0,7928490	0,0105475	0,0145700
80	10	0,3781980	0,8080410	0,0113357	0,0154089
88	11	0,4045560	0,8531130	0,0122849	0,0162547
96	12	0,3553350	0,7968300	0,0133349	0,0171073
104	13	0,4131230	0,8467790	0,0142800	0,0179671
112	14	0,3854950	0,8274050	0,0150728	0,0188339
120	15	0,3803660	0,8353610	0,0164191	0,0204862
128	16	0,4148540	0,8698620	0,0172438	0,0216680

136	17	0,4183630	0,8829120	0,0181424	0,0225624
144	18	0,3966270	0,8582670	0,0193022	0,0234641
152	19	0,4075550	0,8605580	0,0199542	0,0243732
160	20	0,4095520	0,8790590	0,0208377	0,0252896
168	21	0,4010410	0,8339190	0,0219849	0,0278160
176	22	0,4170120	0,8847750	0,0230884	0,0287550
184	23	0,4033110	0,8529920	0,0237474	0,0297014
192	24	0,4033110	0,9002610	0,0246366	0,0306551
200	25	0,4359030	0,9093640	0,0257010	0,0340673
208	26	0,4381020	0,9109080	0,0265809	0,0350475
216	27	0,4253880	0,9102190	0,0282390	0,0368598
224	28	0,4359030	0,9188610	0,0287871	0,0378587
232	29	0,4671680	0,9552630	0,0298417	0,0388649
240	30	0,4209770	0,9032450	0,0311536	0,0407150
248	31	0,4293740	0,9204640	0,0317215	0,0417399
256	32	0,4716310	0,9651930	0,0328507	0,0430929
264	33	0,4342080	0,9382270	0,0339906	0,0458293
272	34	0,4527710	0,9574080	0,0344967	0,0468860
280	35	0,4728330	0,9579350	0,0358007	0,0479500
288	36	0,4209770	0,9171620	0,0366808	0,0473035
296	37	0,4549140	0,9313530	0,0373937	0,0483748
304	38	0,4285870	0,9463300	0,0380284	0,0494535
312	39	0,4159650	0,9182860	0,0393315	0,0531511
320	40	0,4589360	0,9695870	0,0403162	0,0542562
328	41	0,4782150	0,9822540	0,0416667	0,0553687
336	42	0,4159650	0,9304500	0,0423906	0,0573707
344	43	0,4857310	0,9956610	0,0437415	0,0585020
352	44	0,4646850	0,9568640	0,0442675	0,0596407
360	45	0,4341480	0,9441080	0,0457245	0,0616803
368	46	0,4878940	1,0003600	0,0467788	0,0628378
376	47	0,4669520	0,9837720	0,0472766	0,0640027
384	48	0,4711550	0,9869740	0,0486997	0,0655207
392	49	0,4981290	0,9825890	0,0478712	0,0621609
400	50	0,4622760	0,9893590	0,0485677	0,0633309

Σε κάθε γραμμή του Πίνακα 4.3 αποτυπώνονται τα αποτελέσματα που προκύπτουν από την εκτέλεση του προτεινόμενου λογισμικού (συνολικά 43 εκτελέσεις) για διαφορετική χωρητικότητα της CAM. Με γκρι χρώμα σημειώνεται το βελτιστοποιημένο μέγεθος (Cycle Time). Είναι εύκολα παρατηρήσιμο το γεγονός ότι η τιμή του Cycle time για κάθε μέγεθος Tag line, είναι περίπου διπλάσια από αυτή

του Access time για το αντίστοιχο μέγεθος Tag line. Στο γράφημα του Σχήματος 4.3, εμφανίζεται το Cycle Time με τιμές όπως προκύπτουν από τον Πίνακα 4.3.



Σχήμα 4.3 Οι τιμές του Cycle Time για Tag Line size από 8 μέχρι και 50 bits

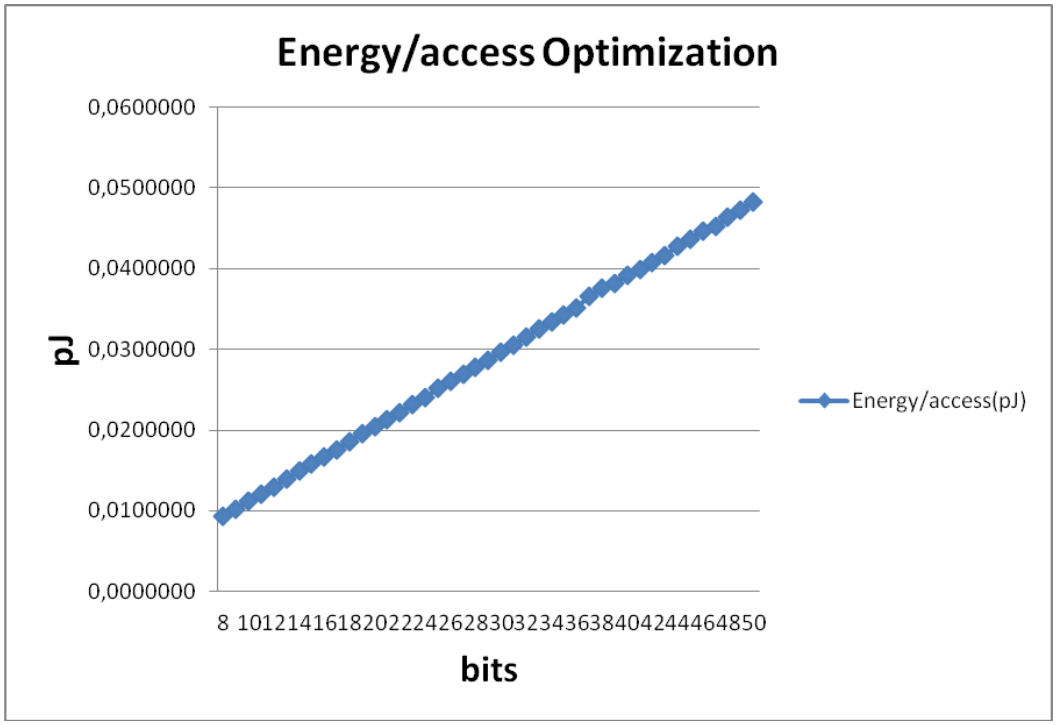
Οι μικρές διακυμάνσεις που εμφανίζονται είναι απόρροια της επιλογής του μεγέθους του ελαχίστου μεγέθους stack, καθώς και του δένδρου συνένωσης των stacks που επιλέγεται κάθε φορά, από το προτεινόμενο λογισμικό. Συγκεκριμένα οι προς τα κάτω διακυμάνσεις οφείλονται στο γεγονός ότι το tag size είναι πολλαπλάσιο του ελάχιστου stack size όπως αυτό έχει οριστεί στο configuration file, ενώ οι προς τα πάνω διακυμάνσεις (γειτονικές των προηγουμένων) οφείλονται σε tag size που δεν είναι πολλαπλάσιο του ελαχίστου μεγέθους stack με αποτέλεσμα να δημιουργούνται και stacks μεγαλύτερου μεγέθους που είναι αυτά που επιφέρουν την επιπλέον καθυστέρηση, γιατί όπως έχει προαναφερθεί η καθυστέρηση είναι ανάλογη του μεγέθους του stack.

Πίνακας 4.4 Αποτελέσματα από το προτεινόμενο λογισμικό με βελτιστοποιημένη Dynamic Search Energy

Capacity(Bytes)	Tag size(bits)	Access time(ns)	Cycle time(ns)	Search energy / access(nJ)	Total area(mm^2)
64	8	0,3658030	0,7836840	0,0093534	0,0114923
72	9	0,5199590	1,0640600	0,0102521	0,0128382
80	10	0,5223090	1,0682600	0,0111395	0,0136670
88	11	0,5627550	1,1167300	0,0120313	0,0145029
96	12	0,6810940	1,3527700	0,0129388	0,0153459
104	13	0,5263240	1,0791500	0,0139812	0,0187756
112	14	0,5651580	1,1422100	0,0149167	0,0196468
120	15	0,6837680	1,3593300	0,0158126	0,0205251
128	16	0,6895600	1,3761500	0,0166691	0,0217070
136	17	0,7298720	1,4347000	0,0175958	0,0226017
144	18	0,8559840	1,6651000	0,0185001	0,0235036
152	19	0,7067820	1,4005000	0,0195560	0,0246906
160	20	0,6713340	1,3798000	0,0203931	0,0256085
168	21	0,8669650	1,6802600	0,0213517	0,0265339
176	22	0,7073840	1,4167500	0,0221833	0,0274665
184	23	0,8814920	1,7098400	0,0231324	0,0284066
192	24	0,9432530	1,8935400	0,0239993	0,0293541
200	25	0,7082610	1,4308800	0,0251322	0,0314120
208	26	0,7496870	1,4725000	0,0260114	0,0323796
216	27	1,0921500	2,1613600	0,0269565	0,0333546
224	28	0,8630800	1,6974300	0,0278338	0,0343370
232	29	0,8690050	1,7099800	0,0287021	0,0353268
240	30	1,1137900	2,2042300	0,0296231	0,0363241
248	31	0,8487160	1,6796000	0,0305796	0,0376230
256	32	0,8284960	1,6595600	0,0314763	0,0389592
264	33	0,9953190	1,9593000	0,0324789	0,0399818
272	34	0,8814430	1,7327700	0,0333522	0,0410119
280	35	1,0026700	1,9753000	0,0342566	0,0420495
288	36	1,2547500	2,4633700	0,0351728	0,0430945
296	37	0,5910650	1,1999200	0,0366163	0,0502298
304	38	0,6357390	1,2568300	0,0376391	0,0513168
312	39	1,1985800	2,4258600	0,0381429	0,0512361
320	40	0,8678430	1,7254000	0,0391293	0,0523328
328	41	0,7459240	1,4648700	0,0398774	0,0534369
336	42	1,1110400	2,1937100	0,0408194	0,0545486
344	43	0,8658920	1,7257300	0,0416741	0,0556677
352	44	1,1532200	2,2783300	0,0427202	0,0567942
360	45	1,2736600	2,5169500	0,0436276	0,0579283

368	46	0,7453540	1,4766600	0,0445943	0,0602809
376	47	0,9849490	1,9672100	0,0452623	0,0602188
384	48	1,3990600	2,7452100	0,0463294	0,0617220
392	49	0,7562420	1,4989300	0,0472914	0,0641137
400	50	0,8054480	1,5792100	0,0482893	0,0652918

Όπως και στον προηγούμενο πίνακα, σε κάθε γραμμή του Πίνακα 4.4 αποτυπώνονται τα αποτελέσματα που προκύπτουν από τις 43 εκτελέσεις του προτεινόμενου λογισμικού για διαφορετική χωρητικότητα της CAM. Επίσης, με γκρι χρώμα σημειώνεται το βελτιστοποιημένο μέγεθος (Dynamic Search Energy). Στο Σχήμα 4.4 που ακολουθεί, εμφανίζονται οι τιμές της καταναλισκόμενης ενέργειας, όπως αποτυπώθηκαν στον Πίνακα 4.4.



Σχήμα 4.4 Οι τιμές του Energy για Tag Line size από 8 μέχρι και 50 bits

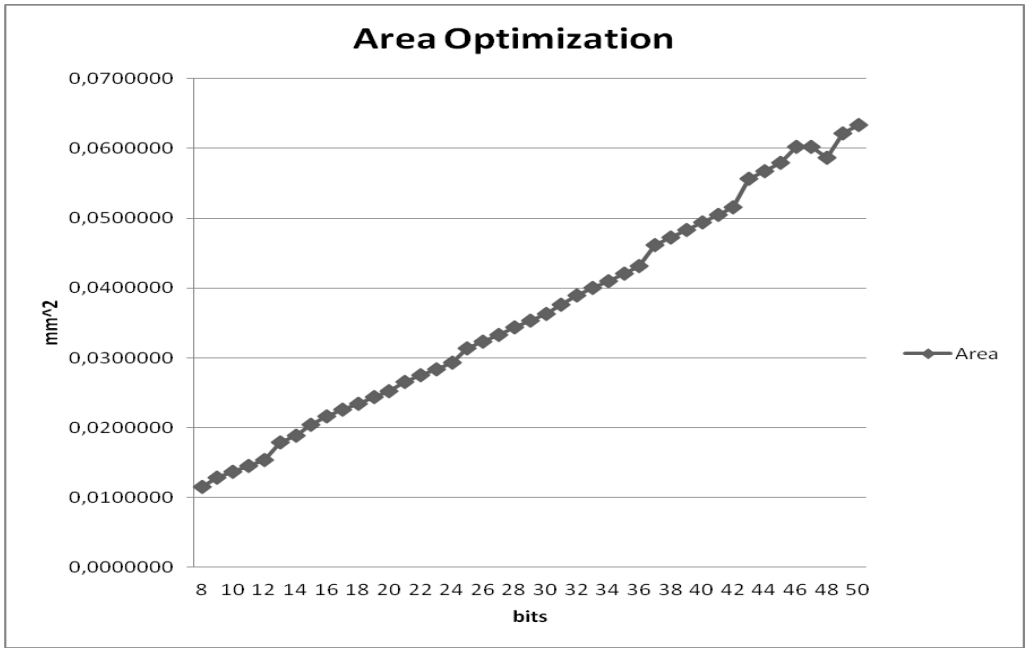
Στο γράφημα του Σχήματος 4.4, παρατηρούμε τις τιμές που υπολογίστηκαν για την καταναλισκόμενη δυναμική ενέργεια. Οι υπολογισμένες τιμές εμφανίζονται να «κινούνται» σχεδόν επάνω στην ίδια ευθεία, φανερώνοντας μία γραμμική σχέση της καταναλισκόμενης ενέργειας με το μέγεθος της Tag line.

Στον Πίνακα 4.5, παρουσιάζονται οι τιμές που υπολογίστηκαν από τις 43 εκτελέσεις του προτεινόμενου λογισμικού με ρύθμιση για βελτιστοποίηση της επιφάνειας (Area) που καταλαμβάνει η CAM. Επίσης, με γκρι χρώμα σημειώνεται το βελτιστοποιημένο μέγεθος (Area). Στο Σχήμα 4.5 που ακολουθεί, εμφανίζονται οι τιμές της καταναλισκόμενης ενέργειας, όπως αποτυπώθηκαν στον Πίνακα 4.5.

Πίνακας 4.5 Αποτελέσματα από το προτεινόμενο λογισμικό με βελτιστοποιημένη Area

Capacity(Bytes)	Tag size(bits)	Access time(ns)	Cycle time(ns)	Search energy / access(nJ)	Total area(mm^2)
64	8	0,3658030	0,7836840	0,0093534	0,0114923
72	9	0,5199590	1,0640600	0,0102521	0,0128382
80	10	0,5223090	1,0682600	0,0111395	0,0136670
88	11	0,5627550	1,1167300	0,0120313	0,0145029
96	12	0,6810940	1,3527700	0,0129388	0,0153459
104	13	0,4131230	0,8467790	0,0142800	0,0179671
112	14	0,3854950	0,8274050	0,0150728	0,0188339
120	15	0,3803660	0,8353610	0,0164191	0,0204862
128	16	0,4148540	0,8698620	0,0172438	0,0216680
136	17	0,4183630	0,8829120	0,0181424	0,0225624
144	18	0,3966270	0,8582670	0,0193022	0,0234641
152	19	0,4075550	0,8605580	0,0199542	0,0243732
160	20	0,4095520	0,8790590	0,0208377	0,0252896
168	21	0,8669650	1,6802600	0,0213517	0,0265339
176	22	0,7073840	1,4167500	0,0221833	0,0274665
184	23	0,8814920	1,7098400	0,0231324	0,0284066
192	24	0,9432530	1,8935400	0,0239993	0,0293541
200	25	0,7082610	1,4308800	0,0251322	0,0314120
208	26	0,7496870	1,4725000	0,0260114	0,0323796
216	27	1,0921500	2,1613600	0,0269565	0,0333546
224	28	0,8630800	1,6974300	0,0278338	0,0343370
232	29	0,8690050	1,7099800	0,0287021	0,0353268
240	30	1,1137900	2,2042300	0,0296231	0,0363241
248	31	0,8487160	1,6796000	0,0305796	0,0376230
256	32	0,8284960	1,6595600	0,0314763	0,0389592
264	33	0,9953190	1,9593000	0,0324789	0,0399818
272	34	0,8814430	1,7327700	0,0333522	0,0410119
280	35	1,0026700	1,9753000	0,0342566	0,0420495
288	36	1,2547500	2,4633700	0,0351728	0,0430945

296	37	0,7162630	1,4146200	0,0360810	0,0461746
304	38	0,7196110	1,4705500	0,0369750	0,0472436
312	39	1,3536700	2,6980200	0,0379749	0,0483200
320	40	1,0266100	2,0213400	0,0389095	0,0494039
328	41	0,9088850	1,7617800	0,0397823	0,0504953
336	42	1,2287800	2,3941800	0,0406377	0,0515942
344	43	0,8658920	1,7257300	0,0416741	0,0556677
352	44	1,1532200	2,2783300	0,0427202	0,0567942
360	45	1,2736600	2,5169500	0,0436276	0,0579283
368	46	0,7453540	1,4766600	0,0445943	0,0602809
376	47	0,9849490	1,9672100	0,0452623	0,0602188
384	48	1,5626900	3,0221200	0,0461279	0,0586920
392	49	0,4981290	0,9825890	0,0478712	0,0621609
400	50	0,4622760	0,9893590	0,0485677	0,0633309



Σχήμα 4.5 Οι τιμές της επιφάνειας που καταλαμβάνει το Tag array για διάφορες τιμές μεγέθους Tag line

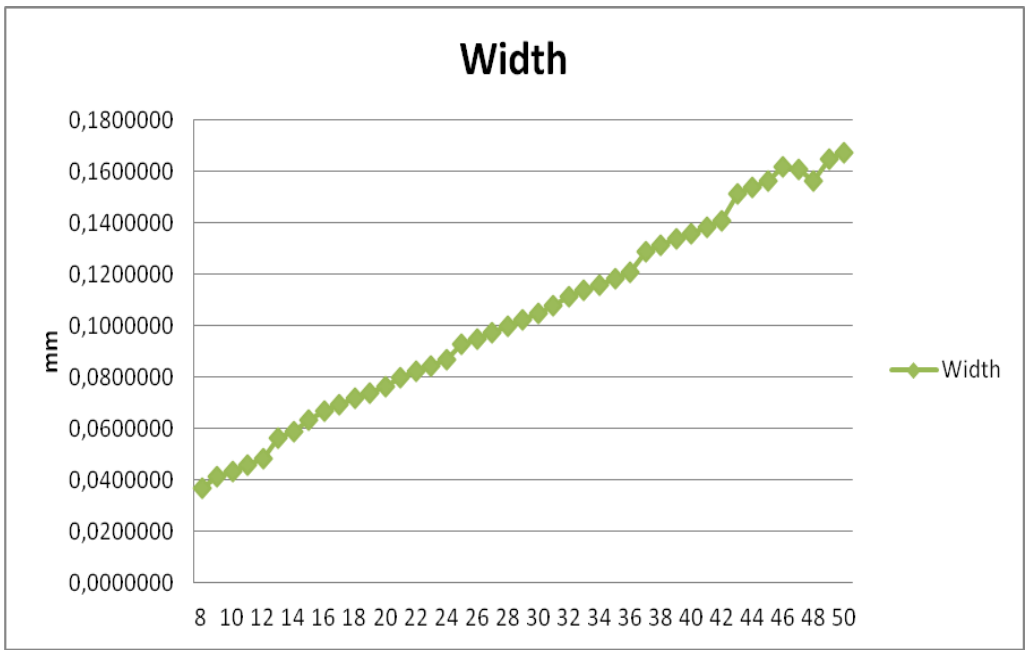
Στο γράφημα του Σχήματος 4.5 αποτυπώνονται τις τιμές που υπολογίστηκαν για την επιφάνεια που καταλαμβάνει η CAM. Οι υπολογισμένες τιμές εμφανίζουν μία γραμμική σχέση με το μέγεθος της tag line με σχεδόν διπλάσια επιφάνεια (αύξηση κατά περίπου 90%) για κάθε διπλασιασμό του πλήθους των cells στο tag line. Το γεγονός αυτό οφείλεται στην αύξηση σχεδόν στο διπλάσιο του πλάτους της tag line

κάθε φορά που διπλασιάζεται το πλήθος των cells. Το φαινόμενο αυτό παρουσιάζεται στον παρακάτω πίνακα (Πίνακας 4.6), καθώς και στα δύο γραφήματα που ακολουθούν (Σχήμα 4.5 & 4.6). Η αύξηση του ύψους της CAM κινείται γύρω στο 4% ανά διπλάσιο tag line και οφείλεται στις γραμμές δεδομένων αναζήτησης που, ακολουθώντας τη σύμβαση του CACTI, από κάθετες μέσα στο array, γίνονται οριζόντιες, αυξάνοντας το ύψος της CAM. Η αύξηση αυτή είναι πολύ μικρή σε αντίθεση με αυτή του πλάτους που είναι περίπου στο 80% για κάθε διπλασιασμό του tag size.

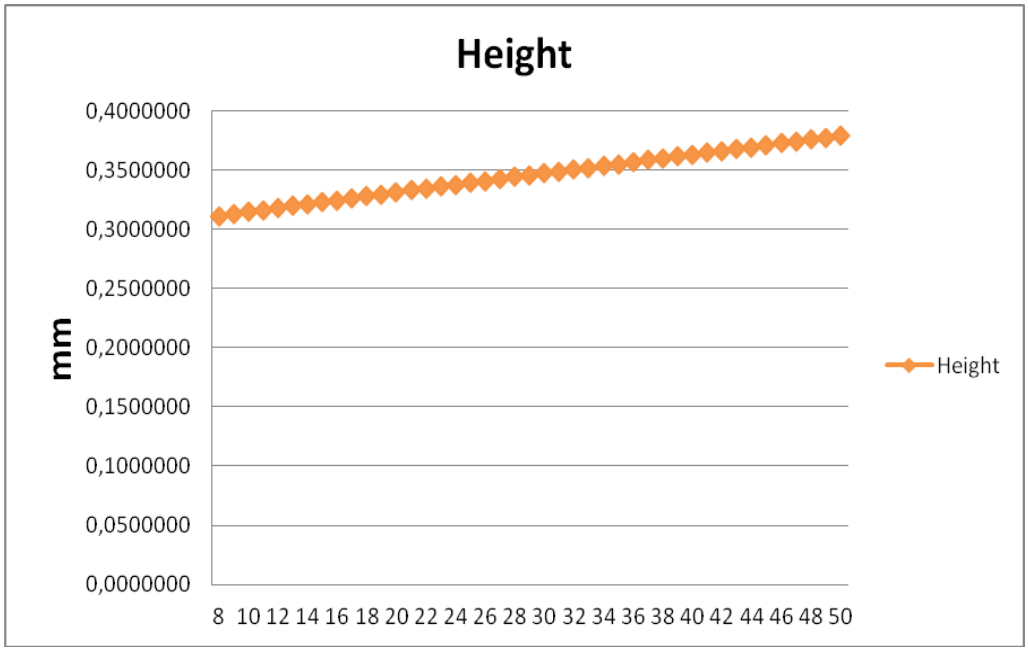
Πίνακας 4.6 Ύψος, πλάτος και επιφάνεια του Tag array

Capacity(Bytes)	Tag size(bits)	Height(mm)	Width(mm)	Total area(mm^2)
64	8	0,3109380	0,0369601	0,0114923
72	9	0,3128090	0,0410416	0,0128382
80	10	0,3146290	0,0434385	0,0136670
88	11	0,3164100	0,0458358	0,0145029
96	12	0,3181590	0,0482333	0,0153459
104	13	0,3198840	0,0561675	0,0179671
112	14	0,3215880	0,0585655	0,0188339
120	15	0,3232750	0,0633708	0,0204862
128	16	0,3246340	0,0667458	0,0216680
136	17	0,3263100	0,0691442	0,0225624
144	18	0,3279740	0,0715427	0,0234641
152	19	0,3296280	0,0739414	0,0243732
160	20	0,3312750	0,0763402	0,0252896
168	21	0,3329130	0,0797020	0,0265339
176	22	0,3345460	0,0821010	0,0274665
184	23	0,3361730	0,0845002	0,0284066
192	24	0,3377940	0,0868994	0,0293541
200	25	0,3394120	0,0925484	0,0314120
208	26	0,3410250	0,0949478	0,0323796
216	27	0,3426340	0,0973473	0,0333546
224	28	0,3442410	0,0997469	0,0343370
232	29	0,3458440	0,1021470	0,0353268
240	30	0,3474450	0,1045460	0,0363241
248	31	0,3490430	0,1077890	0,0376230
256	32	0,3504640	0,1111640	0,0389592

264	33	0,3520630	0,1135640	0,0399818
272	34	0,3536590	0,1159650	0,0410119
280	35	0,3552540	0,1183650	0,0420495
288	36	0,3568470	0,1207650	0,0430945
296	37	0,3584380	0,1288220	0,0461746
304	38	0,3600280	0,1312220	0,0472436
312	39	0,3616160	0,1336230	0,0483200
320	40	0,3632030	0,1360230	0,0494039
328	41	0,3647880	0,1384240	0,0504953
336	42	0,3663730	0,1408240	0,0515942
344	43	0,3679570	0,1512890	0,0556677
352	44	0,3695390	0,1536890	0,0567942
360	45	0,3711210	0,1560900	0,0579283
368	46	0,3727010	0,1617410	0,0602809
376	47	0,3742810	0,1608920	0,0602188
384	48	0,3757390	0,1562040	0,0586920
392	49	0,3773190	0,1647440	0,0621609
400	50	0,3788990	0,1671450	0,0633309



Σχήμα 4.6 Οι τιμές του πλάτους που καταλαμβάνει το Tag array για διάφορες τιμές μεγέθους Tag line



Σχήμα 4.7 Οι τιμές του ύψους που καταλαμβάνει το Tag array για διάφορες τιμές μεγέθους Tag line

Το προτεινόμενο λογισμικό, ανάλογα με το κριτήριο βελτιστοποίησης που έχει τεθεί, επιλέγει να διαχωρίσει τα cells της tag line με διαφορετικό τρόπο, τέτοιον που κάθε φορά να εξυπηρετεί την επιθυμητή βελτιστοποίηση. Από τα λεπτομερή αποτελέσματα των αρχείων καταγραφής (δεν παρουσιάζονται εδώ) παρατηρείται ότι ο βέλτιστος τρόπος διαχωρισμού ακολουθεί κάποια πρότυπα.

Όταν το ζητούμενο είναι η ελαχιστοποίηση του χρόνου κύκλου, το προτεινόμενο λογισμικό δημιουργεί όσο το δυνατόν περισσότερα stacks με το μικρότερο δυνατό μέγεθος. Αυτό συμβαίνει γιατί το μέγεθος του stack είναι αυτό που κυρίως καθορίζει το χρόνο προσπέλασης (και κύκλου) παρά το μέγεθος του δέντρου συνένωσης. Έτσι για βελτιστοποίηση χρόνου, το λογισμικό προτιμά να δημιουργήσει μικρά stacks σε βάρος του μεγέθους του δέντρου συνένωσης.

Αντίθετα, η ελαχιστοποίηση της καταναλισκόμενης δυναμικής ενέργειας ή της επιφάνειας, οδηγεί το προτεινόμενο λογισμικό να δημιουργήσει όσο το δυνατόν λιγότερα stacks με το μεγαλύτερο δυνατό μέγεθος. Ο λόγος εδώ είναι ότι τα μεγαλύτερα stacks έχουν μικρότερη πιθανότητα να ταιριάσουν και να αποφορτιστούν,

άρα και να καταναλώσουν ενέργεια. Επίσης όταν η γραμμή tag χωρίζεται σε μεγάλα stacks, το δέντρο συνένωσης έχει λιγότερες εισόδους, συνεπώς αποτελείται από λιγότερες πύλες.

ΚΕΦΑΛΑΙΟ 5. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΠΡΟΕΚΤΑΣΕΙΣ

5.1. Συμπεράσματα

Στην παρούσα εργασία, μελετήθηκε το τμήμα ετικετών μίας κρυφής μνήμης, αποτελούμενο από μνήμη CAM με μία ιεραρχική γραμμή ταυτοποίησης τύπου NAND.

Αρχικά, έγινε εκτεταμένη αναφορά στην αρχή λειτουργίας και την οργάνωση των κρυφών μνημών (Cache), καθώς επίσης και των συσχετιστικών μνημών CAM. Για τις δεύτερες, αναλύθηκαν οι τρείς κυρίαρχες τάσεις (NOR, NAND και Mixed). Επιπλέον, έγινε αναφορά στα εργαλεία CACTI και SPECTRE που από το μεν πρώτο χρησιμοποιήθηκαν διάφορες μέθοδοι υπολογισμού, ενώ στο δεύτερο έγιναν προσομοιώσεις για σύγκριση αποτελεσμάτων.

Στη συνέχεια, μοντελοποιήθηκε μία NAND CAM με οργάνωση αντίστοιχη με αυτή που παρουσιάζεται στο άρθρο “Low-Power High-Performance NAND Match Line Content Addressable Memories” [8], αλλά για οποιοδήποτε μέγεθος είτε του αριθμού των στηλών είτε των σειρών του τμήματος ετικετών (Tag Array). Η μοντελοποίηση έγινε χρησιμοποιώντας μεθοδολογία παρόμοια με αυτή του εργαλείου CACTI [9], δηλαδή έγινε ανάλυση του κάθε στοιχείου από το οποίο αποτελείται ένα cell σε επίπεδο RC δικτυώματος και στη συνέχεια δόμηση μίας μνήμης από cells και υπολογισμοί των μεγεθών χρόνου, ενέργειας και επιφάνειας.

Στα πλαίσια της μοντελοποίησης της NAND CAM, υλοποιήθηκε αλγόριθμος για τον βέλτιστο διαχωρισμό των cells της κάθε γραμμής (tag line) του τμήματος ετικετών και αποδείχθηκε η γενική ισχύς του με αλγεβρικό τρόπο.

Για την αξιολόγηση του μοντέλου κρίθηκε αναγκαία η υλοποίηση ενός εργαλείου λογισμικού που υπολογίζει το εμβαδό, τη κατανάλωση δυναμικής ενέργειας, το χρόνο και το κύκλο προσπέλασης μιας NAND CAM οποιουδήποτε μεγέθους. Το εργαλείο λογισμικού, πιστοποιήθηκε για την ορθότητα των υπολογισμών που κάνει, συγκρίνοντας αποτελέσματα που εξήχθησαν από αυτό με αποτελέσματα προσομοιώσεων με το SPECTRE καθώς και με γνωστά αποτελέσματα από αντίστοιχες εργασίες. Η ποσοστιαία διαφορά των αποτελεσμάτων των υπολογισμών που έγιναν με το προτεινόμενο λογισμικό και των αποτελεσμάτων των προσομοιώσεων με το SPECTRE, εμφανίζει πολύ μικρή διαφορά (περίπου 3%) με αυτή που προέκυψε από τη διαδικασία πιστοποίησης (συγκρίνοντας με SPICE) της ορθότητας των υπολογισμών που γίνονται από το CACTI.

Με το εργαλείο λογισμικού που υλοποιήθηκε, διεξήχθησαν υπολογισμοί του εμβαδού, της κατανάλωσης δυναμικής ενέργειας, του χρόνου και του κύκλου προσπέλασης, για μία σειρά από περιπτώσεις χρήσης NAND CAM, διαφόρων μεγεθών. Τα αποτελέσματα που παρουσιάστηκαν, έδειξαν πως όλα τα μεγέθη (εμβαδό, δυναμική ενέργεια, χρόνος και κύκλος προσπέλασης) αυξάνονται αναλογικά με το πλήθος των bits που απαρτίζουν τη tag line.

Στο πεδίο του χρόνου, οι μικρές διακυμάνσεις που εμφανίζονται είναι απόρροια της επιλογής του μεγέθους του ελαχίστου μεγέθους stack, καθώς και του δένδρου συνένωσης των stacks που επιλέγεται κάθε φορά. Συγκεκριμένα, όπως προαναφέρθηκε, οι προς τα κάτω διακυμάνσεις οφείλονται στο γεγονός ότι το tag size είναι πολλαπλάσιο του ελάχιστου stack size, ενώ οι (γειτονικές των προηγουμένων) προς τα πάνω διακυμάνσεις οφείλονται σε tag size που δεν είναι πολλαπλάσιο του ελαχίστου μεγέθους stack. Αποτέλεσμα του τελευταίου είναι η δημιουργία stacks μεγαλύτερου μεγέθους τα οποία επιφέρουν επιπλέον καθυστέρηση.

Αναφορικά με την καταναλισκόμενη δυναμική ενέργεια, η αύξηση είναι και εδώ γραμμική και ανάλογη του μεγέθους της tag line, με αυτή να κυμαίνεται περίπου στο 80% για κάθε διπλασιασμό του πλήθους των cells.

Τέλος, η επιφάνεια που απαιτείται εμφανίζει μία γραμμική σχέση με το μέγεθος της tag line με σχεδόν διπλάσια επιφάνεια (αύξηση κατά περίπου 90%) για κάθε διπλασιασμό του πλήθους των cells. Το γεγονός αυτό οφείλεται στην αύξηση σχεδόν στο διπλάσιο του πλάτους της tag line κάθε φορά που διπλασιάζεται το πλήθος των cells. Συγκεκριμένα, η αύξηση του ύψους του tag array είναι περίπου 4% και του πλάτους περίπου 80% για κάθε διπλασιασμό του μεγέθους της tag line.

Τα αποτελέσματα που υπολογίστηκαν για όλες τις περιπτώσεις χρήσης κρίνονται ως αναμενόμενα, με τους υπολογισμούς στο πεδίο του χρόνου να εμφανίζουν ιδιαίτερο ενδιαφέρον. Συγκεκριμένα οι προς τα κάτω διακυμάνσεις που όπως προαναφέρθηκε οφείλονται στο γεγονός ότι τα stacks που δημιουργούνται είναι πολλαπλάσια του ελάχιστου stack size, καταδεικνύουν πως η επιλογή του κατάλληλου πλήθους των cells της tag line και κατ’ επέκταση του δένδρου συνένωσης, μπορούν να αποτελέσει παράγοντα στη βελτιστοποίηση των χρόνων μίας NAND CAM.

Η υλοποίηση του εργαλείου λογισμικού για την αξιολόγηση της NAND CAM, έδωσε τη δυνατότητα σύγκρισής της με NOR CAM, η μοντελοποίηση της οποίας μπορεί να γίνει με το CACTI. Έτσι για ίδια χωρητικότητα, ίδιο μέγεθος tag line και για δύο διαφορετικές τεχνολογίες, η σύγκριση των υπολογισμένων μεγεθών, έδειξε ότι η NOR CAM υπερέχει στο κομμάτι της απαιτούμενης επιφάνειας κατά περίπου 10%. Επίσης, υπερέχει και στο πεδίο του χρόνου εμφανίζοντας όμως μία τάση η υπεροχή αυτή να μειώνεται δραστικά καθώς βελτιώνεται η τεχνολογία (μείωση της διαφοράς χρόνου προσπέλασης και κύκλου κατά περίπου 10% από τα 130 στα 90nm). Τέλος, η NAND CAM εμφανίζει αξιοσημείωτα μειωμένη κατανάλωση δυναμικής ενέργειας σε σχέση με αυτή της NOR (14% στα 130 και 19% στα 90 nm), η οποία βελτιώνεται καθώς βελτιώνεται η τεχνολογία.

Σύμφωνα με τα παραπάνω, είναι εμφανές ότι η χρήση των ιεραρχικών NAND CAM είναι εφικτή, σε εφαρμογές που το ζητούμενο δεν είναι μόνο η χαμηλή κατανάλωση

ενέργειας αλλά και η υψηλή ταχύτητα λειτουργίας. Το τελευταίο προκύπτει αφού, σε σύγχρονες τεχνολογίες (π.χ. 90nm), η NAND CAM δεν υπερτερεί μόνο στην καταναλισκόμενη δυναμική ενέργεια, αλλά εμφανίζεται να έχει το ίδιο περίπου χρόνο κύκλου με τη NOR CAM, η οποία υπερτερεί μόνο κατά 0.5%.

5.2. Μελλοντικές Κατευθύνσεις και Προεκτάσεις

Δεδομένου ότι το λογισμικό που δημιουργήθηκε στα πλαίσια της παρούσης εργασίας υπολογίζει τους χρόνους προσπέλασης και κύκλου, την καταναλισκόμενη δυναμική ενέργεια και την απαιτούμενη επιφάνεια για CAM μνήμες με ιεραρχική NAND match line, είναι δυνατόν να πραγματοποιηθεί ένας αριθμός βελτιώσεων σε αυτό, καθώς επίσης και προεκτάσεων για μοντελοποίηση και άλλων αρχιτεκτονικών. Για τις μελλοντικές αυτές κατευθύνσεις κρίνεται, ωστόσο, απαραίτητο να πραγματοποιηθεί επιπλέον έρευνα. Συνοπτικά, τα θέματα που προτείνονται είναι τα ακόλουθα:

- Επέκταση της λειτουργικότητας του λογισμικού στην κατεύθυνση υπολογισμού της κατανάλωσης της στατικής ενέργειας, δηλαδή αυτής που προέρχεται από διαρροές ρευμάτων.
- Επέκταση της λειτουργικότητας ώστε να είναι δυνατή η διεξαγωγή υπολογισμών και για άλλες NAND CAM αρχιτεκτονικές, όπως αυτή που περιγράφεται στο [13].
- Επέκταση της λειτουργικότητας ώστε να είναι δυνατή η διεξαγωγή υπολογισμών και για αρχιτεκτονικές υβριδικών μνημών CAM, όπως η "Mixed NOR–NAND Match Line CAM" [7] που περιγράφηκε στην παράγραφο 2.5.4.
- Προσάρτηση των μεθόδων υπολογισμού, που αναπτύχθηκαν κατά τη διαδικασία υλοποίησης του προτεινόμενου λογισμικού, στο εργαλείο CACTI. Η επέκταση του CACTI με την προσθήκη αυτή, θα έδινε τη δυνατότητα σε όσους ασχολούνται με τη μελέτη μνημών και μικροεπεξεργαστών, να διεξάγουν μετρήσεις σε μία αρχιτεκτονική που ως τώρα δεν καλυπτόταν από το CACTI. Επιπλέον, οι σύγκριση της NAND CAM με ευρέως διαδεδομένες αρχιτεκτονικές όπως η NOR CAM, στο ενοποιημένο περιβάλλον του CACTI, θα μπορούσε να φανεί πολύ χρήσιμη λόγω της άμεσης εξαγωγής συμπερασμάτων.

ΑΝΑΦΟΡΕΣ

[1] Δ. Νικολός, Ψηφιακά Συστήματα – Αρχιτεκτονική Υπολογιστών Ι, Τόμος Β'. Πάτρα: Ελληνικό Ανοικτό Πανεπιστήμιο, 2001.

[2] John L. Hennessy, David A. Patterson, Computer Architecture, 4th ed.: A Quantitative Approach, San Francisco CA: Morgan Kaufman, 2007.

[3] Bruce Jacob, Spencer W. Ng, David T. Wang, Memory Systems Cache, DRAM, Disk, Burlington, MA: Morgan Kaufman, 2008.

[4] S.Wilton and N. P. Jouppi, "An Enhanced Access and Cycle TimeModel for On-Chip Caches," DECWRL, Tech. Rep. Technical report number 93/5, 1994.

[5] Kostas Pagiamtzis and Ali Sheikholeslami "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey"

[6] Pedro Echeverr´ıa, Jos´e L. Ayala and Marisa L´opez-Vallejo "Practical Implementation of a Low-Power Content-Addressable Memory"

[7] A. Efthymiou and J. D. Garside, "A CAM with mixed serial-parallel comparison for use in low energy caches," IEEE Trans. VLSI Syst., vol. 12, no. 3, pp. 325–329, Mar. 2004.

[8] V. Chaudhary and Lawrence. T. Clark, "Low-Power High-Performance NAND Match Line Content Addressable Memories," IEEE Trans. VLSI Syst., vol. 14, no. 8, pp. 895–905, Aug. 2006.

[9] S. Wilton and N. Jouppi, "CACTI: An enhanced cache access and cycle time model," IEEE Journal of Solid-State Circuits,vol. 31, no. 5, pp. 677–688, May 1996.

[10] Arkadiy Morgenshtein, Eby G. Friedman, Ran Ginosar and Avinoam Kolodny. "Unified Logical Effort A Method for Delay Evaluation and Minimization in Logic Paths with RC Interconnect," IEEE Trans. VLSI Syst., vol. 18, no. 5, pp. 689–996, May. 2010.

[11] M. A. Horowitz, "Timing models for MOS circuits," PhD Thesis, Tech. Rep. SEL803-003, Integrated Circuits Laboratory, Stanford Univ., 1983.

[12] http://www.hpl.hp.com/research/mcpat/

[13] H.Y. Li, C. C. Chen, J. S. Wang, C. Yeh “An AND-type match-line scheme for high-performance energy-efficient content addressable memories,” IEEE Journal of Solid-State Circuits, vol. 41 no 5, pp. 1108-1119, May. 2006.

[14] W. C. Elmore “The transient response of damped linear networks,” Journal of Applied Physics, vol. 19, pp.55–63, 1948.

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Ο Παπαθεοδώρου Γιώργος έλαβε δίπλωμα επαγγελματικής κατάρτισης Τεχνικού Η/Υ και Ηλεκτρονικών Μηχανών Γραφείου το 2000 από το 1ο ΙΕΚ Ιωαννίνων και το 2010 έλαβε πτυχίο Πληροφορικής από το τμήμα Πληροφορικής του Ελληνικού Ανοιχτού Πανεπιστημίου. Από το 2011 είναι μεταπτυχιακός φοιτητής του τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Ιωαννίνων.

Από το 1998 μέχρι το 2005 εργάστηκε σε δύο εταιρίες του χώρου, παρέχοντας υπηρεσίες ως τεχνικός Η/Υ, ηλεκτρονικών μηχανών γραφείου και δικτύων, ενώ το 2005 μέχρι το 2008 διεύθυνε εταιρία τις οποίας υπήρξε συνιδρυτής, στην οποία παρείχε έργο ως μηχανικός λογισμικού, web developer και εκπαιδευτής πληροφορικής. Το 2009 ίδρυσε εταιρία με διακριτικό τίτλο “InfoGear” που δραστηριοποιείται στον χώρο του διαδικτύου και των εφαρμογών λογισμικού.

Τον Ιανουάριο του 2014, ολοκλήρωσε το μεταπτυχιακό πρόγραμμα σπουδών, αποκτώντας Μεταπτυχιακό Δίπλωμα με ειδίκευση στις «Τεχνολογίες – Εφαρμογές».