ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Τεχνοοικονομική Διοίκηση και Ασφάλεια Ψηφιακών Συστημάτων»

Διπλωματική Εργασία

Τεχνολογία Μνήμης Flash

Υλοποίηση Εργαστηριακής Λύσης, Ανάγνωση Μνήμης Τεχνολογίας NAND Flash - Ανάλυση Εικόνας Μνήμης

Τσεκούρας Βασίλειος

Απρίλιος 2014

Επιβλέπων Καθηγητής

Χρήστος Ξενάκης, Επίκουρος Καθηγητής
Πανεπιστήμιο Πειραιώς

Εξεταστική Επιτροπή

Σωκράτης Κάτσικας, Καθηγητής

Πανεπιστήμιο Πειραιώς

Κωνσταντίνος Λαμπρινουδάκης, Αναπληρωτής Καθηγητής

Πανεπιστήμιο Πειραιώς

Χρήστος Ξενάκης, Επίκουρος Καθηγητής

Πανεπιστήμιο Πειραιώς

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Πίνακας Περιεχομένων

Περιεχόμενα

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Περίληψη

Οι σύγχρονες ηλεκτρονικές συσκευές (smartphones, routers, digital cameras, usb stick…) αποτελούν αναπόσπαστο κομμάτι της ζωής μας και μια πληθώρα προσωπικών πληροφοριών όπως εικόνες, κωδικοί πρόσβασης, έγγραφα και πολλά άλλα αρχεία είναι αποθηκευμένα σε αυτές. Σε επίπεδο υλικού συσκευής, εκείνο το δομικό στοιχείο -ηλεκτρονικό εξάρτημα- που είναι υπεύθυνο για την αποθήκευση των δεδομένων, είναι το ολοκληρωμένο μνήμης τεχνολογίας Flash. Το μεγαλύτερο ποσοστό των ηλεκτρονικών συσκευών, χρησιμοποιούν ολοκληρωμένα μνήμης NAND Flash. Τα χαρακτηριστικά αυτού του είδους μνήμης είναι ιδανικά για χρήση σε ενθυλακωμένα συστήματα και χρησιμοποιούνται από τις εταιρίες για την εγκατάσταση του Λειτουργικού Συστήματος της ηλεκτρονικής συσκευή και την αποθήκευση προσωπικών δεδομένων του χρήστη κατά την λειτουργία της.

Στην παρούσα διπλωματική εργασία παρουσιάζονται τα γενικά χαρακτηριστικά των μνημών τεχνολογίας Flash και αναλύονται σε βάθος, τα χαρακτηριστικά και η μέθοδος λειτουργίας των μνημών NAND Flash. Πραγματοποιείται επίσης, η υλοποίηση εργαστηριακής λύσης για την ανάγνωση εικόνας μνήμης από ολοκληρωμένα NAND Flash, αξιοποιώντας κατάλληλο λογισμικό ανοιχτού κώδικα -ftdinandreader- και υλικού από την εταιρία FTDI. Στη συνέχεια αφού υλοποιήσαμε την κατασκευή μας, προχωρήσαμε στην ανάγνωση εικόνας μνήμης από ένα usb stick, αποκολλώντας το ολοκληρωμένο μνήμης από αυτό. Το δυαδικό αρχείο που προέκυψε κατά την ανάγνωση, αναλύθηκε κι έγινε αξιολόγηση και προσπάθεια ανάκτησης του ανώτερου συστήματος αρχείων που είχε χρησιμοποιηθεί για την αποθήκευση των δεδομένων στην ηλεκτρονική συσκευή.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Λέξεις Κλειδιά: Τεχνολογία Flash, Τεχνολογία NAND Flash, Ανάγνωση Εικόνας Μνήμης, ftdinandreader, FTDI FT2232H mini module, Ανάλυση Εικόνας Μνήμης, USB stick.

Κεφάλαιο 1ο

Εισαγωγή

1.1 Περιγραφή του προβλήματος

Ως ερευνητές είμαστε συνηθισμένοι σε προκλήσεις. Προκλήσεις που αφορούν είτε δυσλειτουργικούς σκληρούς δίσκους, είτε κρυπτογραφημένα δεδομένα, καινούργιες εφαρμογές και τεχνολογικά μέσα. Πολλές φορές καταφέρνουμε να ανταπεξέλθουμε σε αυτές τις προκλήσεις, υπάρχουν όμως και στιγμές που ακόμη και οι πιο έμπειροι δύσκολα τα καταφέρνουν. Μια από αυτές τις περιπτώσεις είναι η εξαγωγή όλων των δεδομένων και η ανάκτηση πληροφοριών από την στατική μνήμη ηλεκτρονικής συσκευής. Συνήθως για να ολοκληρωθεί με επιτυχία η εξαγωγή των δεδομένων και η ανάκτηση πιθανού διαγραμμένου υλικού, απαιτείται πλήρης πρόσβαση σε κάθε bit πληροφορίας που περιέχεται στον αποθηκευτικό χώρο της συσκευής. Στην περίπτωση παραδοσιακών σκληρών δίσκων, και αναφέρομαι στον όρο παραδοσιακών για να τους ξεχωρίσω από τους σύγχρονους solid state σκληρούς δίσκους, η παραπάνω διαδικασία αναφέρεται ως bit-stream image και περιλαμβάνει την αντιγραφή όλων των τομέων (sectors) του δίσκου. Στο εμπόριο μπορεί να ανακαλύψει κανείς πληθώρα εργαλείων, software και hardware, που βοηθούν στην ανάκτηση των δεδομένων σχετικά εύκολα. Δυστυχώς όμως, οι επιλογές πρόσβασης σε χαμηλού επιπέδου δεδομένα, συσκευών με ενσωματωμένη μνήμη flash, όπως είναι τα σύγχρονα κινητά τηλέφωνα ή ένα router, είναι περιορισμένες και αντιστρόφως ανάλογες με την δημοσιότητα των συγκεκριμένων συσκευών. Για παράδειγμα, πολλοί από εμάς είναι κάτοχοι ενός έξυπνου κινητού, με εφάμιλλες δυνατότητες και λειτουργίες με αυτές ενός υπολογιστή, το οποίο βρίσκεται κάθε λεπτό στην τσέπη μας. Ας καθίσουμε να αναλογιστούμε για μερικά δευτερόλεπτα το σύνολο των πληροφοριών που μπορεί να περιέχει.

Παρότι πολλοί προμηθευτές διαφημίζουν μια πληθώρα εργαλείων τα οποία βοηθούν τον ερευνητή να εξάγει και να αναλύσει δεδομένα από κινητές και ηλεκτρονικές συσκευές, οι δυνατότητες εξαγωγής, συνδέονται άμεσα με το μοντέλο και την φίρμα της εταιρίας. Λόγω της μονοπωλιακής φύσης των συγκεκριμένων συσκευών, οι προμηθευτές εργαλείων πρέπει να ερευνήσουν και να αναπτύξουν λύσεις για κάθε συσκευή ή ομάδα συσκευών ξεχωριστά. Με σχεδόν χιλιάδες μοντέλα συσκευών στην αγορά σήμερα, είναι αναμενόμενο η

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

υποστήριξη να είναι περιορισμένη για συγκεκριμένα προϊόντα, με προτίμηση τα πιο δημοφιλή. Ένα από τα πρώτα βήματα των ερευνητών είναι, να ανατρέξουν στα διαθέσιμα εργαλεία που έχουν στην κατοχή τους, ελπίζοντας ότι ορισμένα από αυτά θα είναι κατάλληλα ώστε να βοηθήσουν στην άντληση των δεδομένων. Σε ορισμένες περιπτώσεις, είναι δυνατόν να επιτευχθεί η εξαγωγή μιας πλήρης εικόνας από την στατική μνήμη της συσκευής, χρησιμοποιώντας κάποιο χρήσιμο εμπορικό εργαλείο ή εξελιγμένες τεχνικές. Πολύ συχνά όμως οι δυνατότητες εξαγωγής είναι περιορισμένες σε λογικού-επιπέδου δεδομένα, όπως μηνύματα κειμένου, φωτογραφίες, μητρώο κλήσεων κλπ. Η περιορισμένη εξαγωγή δεδομένων, δεν επιτρέπει την ανάκτηση όλων των πληροφοριών από την συσκευή, την ανάκτηση διαγραμμένων δεδομένων ή κρίσιμων δεδομένων του συστήματος και των εφαρμογών που ενδεχομένως να είναι εγκατεστημένες, επιτρέποντας να χαθούν σημαντικά δεδομένα και στοιχεία στην έρευνα.

Το ερώτημα λοιπόν που τίθεται είναι, πώς θα μπορέσουμε να αποκτήσουμε μια πλήρης εικόνα της στατικής μνήμης που περιέχεται ενσωματωμένη σε μια ηλεκτρονική συσκευή, όταν η εξαγωγή των δεδομένων, δεν υποστηρίζεται από εμπορικά εργαλεία; Ή τι θα μπορούσε να συμβεί στην περίπτωση που η συσκευή είναι κατεστραμμένη; Αυτές τις περιπτώσεις καλούμαστε να ερευνήσουμε και να μελετήσουμε, χρησιμοποιώντας τεχνικές εξαγωγής δεδομένων μέσα από το ίδιο το ολοκληρωμένο της συσκευής, αφαιρώντας το από αυτή.

1.2 Δομή της διπλωματικής

Η διπλωματική εργασία αποτελείται από έξι κεφάλαια. Στις επόμενες παραγράφους περιγράφεται συνοπτικά το περιεχόμενο του κάθε κεφαλαίου.

Το κεφάλαιο 1 περιέχει την περιγραφή του προβλήματος και τους στόχους, καθώς επίσης τη δομή και τη συνεισφορά της διπλωματικής.

Στο κεφάλαιο 2 αναλύονται οι κύριες κατηγορίες μνήμης (NAND, NOR) τεχνολογίας Flash και περιγράφονται οι διαφορές τους.

Στο κεφάλαιο 3 περιγράφονται τα ιδιαίτερα χαρακτηριστικά της μνήμης NAND Flash και ο τρόπος λειτουργίας της.

Εν συνεχεία, στο κεφάλαιο 4, παρουσιάζεται η υλοποίηση εργαστηριακής λύσης η οποία αποσκοπεί στην ανάγνωση της δυαδικής εικόνας μνήμης NAND Flash, από φορητό μέσο αποθήκευσης (usb stick).

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Στο κεφάλαιο 5, πραγματοποιείται στατική ανάλυση της εικόνας μνήμης που ανακτήθηκε με σκοπό την αναδόμηση του ανώτερου συστήματος αρχείων (file system) που έχει χρησιμοποιηθεί για την εγγραφή των δεδομένων στην μνήμη.

Η ολοκλήρωση της διπλωματικής γίνεται με το κεφάλαιο 6, όπου παρατίθενται τα γενικά συμπεράσματα από την όλη ερευνητική προσπάθεια.

1.3 Συνεισφορά της διπλωματικής

Η ερευνητική προσπάθεια που διατελέστηκε στο πλαίσιο αυτής της διπλωματικής εργασίας συμβάλλει στην γνωριμία με τις μνήμες τεχνολογίας Flash, στην κατανόηση του τρόπου λειτουργίας μιας ειδικής κατηγορίας μνήμης Flash, της μνήμης τεχνολογίας NAND Flash και έχει ως απώτερο σκοπό την υλοποίηση κατασκευής που θα μας επιτρέψει να εξάγουμε μια πλήρη εικόνα μνήμης, από ολοκληρωμένο τεχνολογίας NAND Flash.

Παρουσιάζουμε το απαραίτητο υλικό (hardware) που χρειάζεται για την κατασκευή μας, καθώς επίσης και μια έκδοση προγράμματος (software) ανοιχτού κώδικα, κατάλληλο για την ανάγνωση εικόνας μνήμης από ολοκληρωμένο NAND Flash στα 3,3 Volt με δίαυλο επικοινωνίας 8 bit.

Ως παράδειγμα άσκησης χρησιμοποιούμε ένα ολοκληρωμένο μνήμης τεχνολογίας NAND Flash που βρίσκεται προσαρτημένο σε αποθηκευτικό μέσο usb stick, το οποίο αποκολλήσαμε κάνοντας χρήση ειδικού εξοπλισμού.

Επιπλέον γίνεται έρευνα ώστε να παραχθούν τα κατάλληλα εργαλεία με απώτερο σκοπό την αναδόμηση των λογικών διευθύνσεων μνήμης από τις φυσικές διευθύνσεις του ολοκληρωμένου, ώστε να προκύψει μια πλήρη εικόνα του ανώτερου συστήματος αρχείων - στην περίπτωσή μας FAT32 - που χρησιμοποιείται από το λειτουργικό σύστημα για την εγγραφή των δεδομένων.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Κεφάλαιο 2ο

Τεχνολογία Flash Memory

2.1 Μνήμη Flash

Η μνήμη Flash είναι είδος μνήμης, το οποίο δεν απαιτεί την παροχή ηλεκτρικού ρεύματος προκειμένου να διατηρήσει τα δεδομένα που είναι αποθηκευμένα σε αυτήν. Γνωστή και με τον όρο ως non-volatile memory. Εφαρμόζοντας ηλεκτρικό ρεύμα μπορεί να διαγραφεί και να επαναπρογραμματιστεί αρκετές φορές και χωρίζεται σε δυο κατηγορίες, NOR Flash και NAND Flash μνήμη, οι ποίες πήραν το όνομά τους από την μέθοδο που είναι δομημένες κατασκευαστικά. Σε αντίθεση με την μνήμη NAND, η μνήμη NOR μπορεί να διαβαστεί ανά byte συνεχόμενα και για αυτό το λόγω συχνά χρησιμοποιείται όταν ο πρωταρχικός σκοπός της μνήμης είναι η αποθήκευση και η εκτέλεση του κώδικα εκκίνησης (firmware) μιας ηλεκτρονικής συσκευής. Ενώ μέρος της μνήμης μπορεί επίσης να χρησιμοποιηθεί και για την αποθήκευση δεδομένων του χρήστη. Οι περισσότερες όμως φορητές συσκευές, όπως τα USB stick, οι σύγχρονες ψηφιακές κάμερες, τα smartphones χρησιμοποιούν μνήμες NAND που τους επιτρέπουν την άμεση προσάρτηση αποθηκευτικού χώρου για την αποθήκευση δεδομένων. Για να γίνει καλύτερος διαχωρισμός της μνήμης flash σε σύγκριση με τις υπόλοιπες μνήμες παραθέτουμε τον παρακάτω πίνακα 2.1.

είδος μνήμης	υποκατηγορία	παράδειγμα
Volatile διατηρεί τα δεδομένα όσο υπάρχει τροφοδοσία. Όταν η συσκευή απενεργοποιηθεί τα δεδομένα χάνονται.	**Static memory** διατηρεί τα δεδομένα επ' αόριστον με σχεδόν μηδενική ή πολύ μικρή κατανάλωση ρεύματος.	SRAM, CPU cache
	Dynamic memory διατηρεί τα δεδομένα για μικρό χρονικό διάστημα και για όσο υπάρχει τροφοδοσία. Απαιτείται ανανέωση των δεδομένων και κατανάλωση ενέργειας.	SDRAM

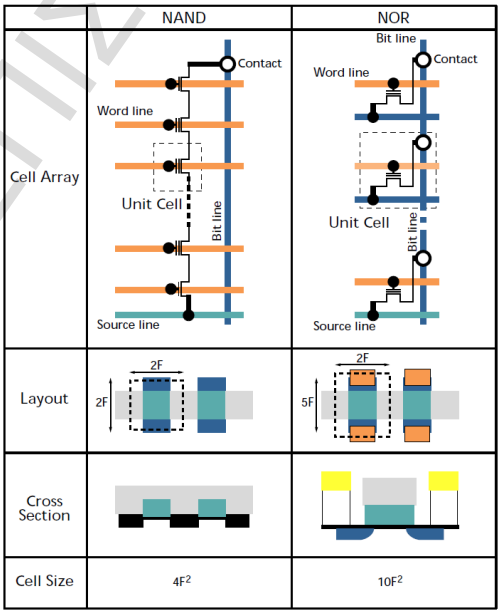
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Non-volatile διατηρεί τα δεδομένα χωρίς να απαιτείται τροφοδοσία ρεύματος.	**Programmable memory** δεδομένα μπορούν να γραφτούν στην συσκευή πολλές φορές.	NAND Flash NOR Flash
	One-time programmable memory δεδομένα μπορούν να εγγραφούν στην συσκευή κατά την διαδικασία κατασκευής της.	ROM

Πίνακας 2.1: Κατηγορίες Μνήμης

2.2 Σύγκριση NAND Flash με NOR Flash

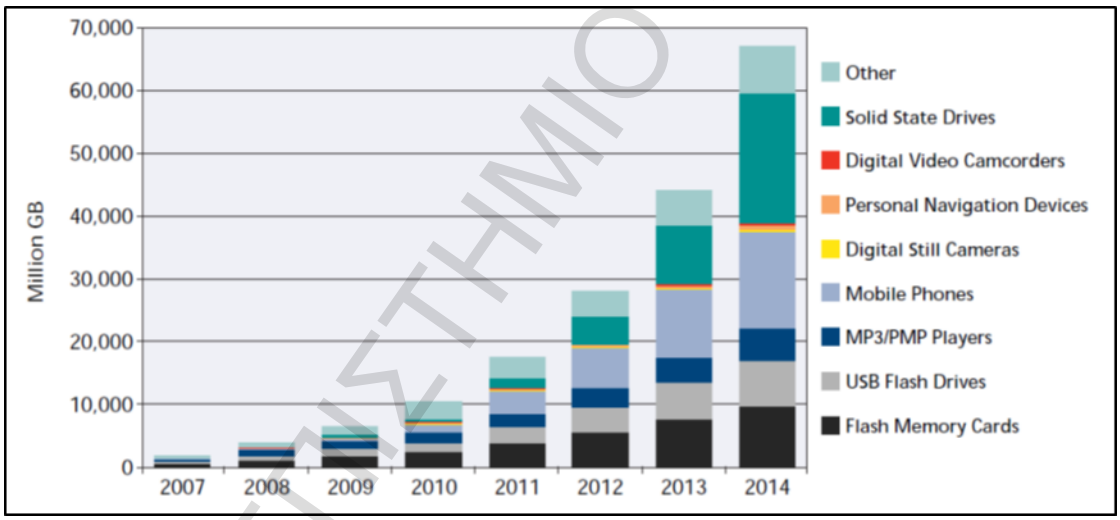
Όπως προαναφέραμε η μεγαλύτερη διαφορά ανάμεσα στην NAND Flash και την NOR Flash είναι η μέθοδος με την οποία τα κελιά μνήμης είναι δομημένα. Και τα δύο είδη μνήμης (NAND,NOR) χρησιμοποιούν για την διατήρηση των δεδομένων, κελιά μνήμης μονού τρανζίστορ. Στις μνήμες NOR τα κελιά είναι συνδεδεμένα παράλληλα με τον κύριο κόμβο της γραμμής bit ενώ στις μνήμες NAND τα κελιά είναι συνδεδεμένα στο κύριο κόμβο σε σειρά. Τα κελιά μνήμης NAND έχουν το πλεονέκτημα ότι μπορούν να στοιβαχτούν πολύ κοντύτερα μεταξύ τους, κερδίζοντας μέχρι και 60% του χώρου, σε αντίθεση με τα κελιά μνήμης NOR. Στην παρακάτω εικόνα 2.1 φαίνεται ξεκάθαρα η εσωτερική δομή τους.



Εικόνα 2.1: NAND, NOR δομή

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Οι μνήμες NAND Flash είναι παρόμοιες στην λειτουργία τους με έναν σκληρό δίσκο. Είναι δομημένες κατά τμήματα (sectors) και κατά σελίδες (pages) και συνεπώς ιδανικές για την συνεχή αποθήκευση δεδομένων, όπως φωτογραφίες, βίντεο, ήχος ή δεδομένα ηλεκτρονικού υπολογιστή. Επίσης, παρόλο που στις μνήμες NAND Flash η μέθοδος τυχαίας προσπέλασης μπορεί να υλοποιηθεί σε επίπεδο συστήματος με την επονομαζόμενη μέθοδο “shadowing”, η διαδικασία απαιτεί την ύπαρξη μνήμης RAM στο σύστημα. Όπως οι σκληροί δίσκοι, έτσι και οι μνήμες NAND Flash μπορεί να παρουσιάσουν ελαττωματικά κελιά (bad blocks) και επομένως απαιτείται αλγόριθμος διόρθωσης σφάλματος (error correction code, ECC) προκειμένου να διατηρηθεί η ακεραιότητα των δεδομένων. Το γεγονός ότι οι μνήμες NAND Flash καταλαμβάνουν 60% λιγότερο χώρο σε σύγκριση με τις μνήμες NOR Flash - παρέχοντας υψηλότερη πυκνότητα - τις καθιστά ιδανικές για την χρήση τους στις σημερινές ηλεκτρονικές συσκευές, όπου το μέγεθος της συσκευής παίζει σημαντικό ρόλο στην προώθηση ενός προϊόντος. Στο παρακάτω γράφημα (εικόνα 2.2) παρουσιάζεται η διείσδυση των μνημών NAND Flash σε προϊόντα της αγοράς ανά έτος.



Εικόνα 2.2: NAND Flash Market

2.3 Πλεονεκτήματα - Μειονεκτήματα κάθε τύπου μνήμης

Υπάρχουν συγκεκριμένα πλεονεκτήματα και μειονεκτήματα στην χρήση μνημών NAND Flash ή NOR Flash σε ενθυλακωμένα συστήματα (embedded systems). Οι μνήμες NAND Flash είναι περισσότερο κατάλληλες για εφαρμογές αρχείων και διαδοχικών-δεδομένων, ενώ οι μνήμες NOR Flash είναι περισσότερο κατάλληλες για εφαρμογές που απαιτούν τυχαία προσπέλαση δεδομένων. Τα πλεονεκτήματα των μνημών NAND Flash έναντι των μνημών NOR Flash είναι οι υψηλότερες ταχύτητες λειτουργίας προγραμματισμού και διαγραφής. Τα πλεονεκτήματα των μνημών NOR Flash έναντι των μνημών NAND Flash είναι η δυνατότητα τυχαίας προσπέλασης και η δυνατότητα εγγραφής ανά byte.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Το χαρακτηριστικό της τυχαίας προσπέλασης που παρουσιάζουν οι μνήμες NOR Flash, δίνει την δυνατότητα υλοποίησης της λειτουργίας execute-in-place (XiP), η οποία συχνά απαιτείται στα ενθυλακωμένα συστήματα. Παρόλα αυτά ένας μεγάλος αριθμός επεξεργαστών περιλαμβάνουν απευθείας διεπαφή για την χρήση NAND Flash μνήμης, προσφέροντας την δυνατότητα εκκίνησης (boot) της συσκευής μέσω μνήμης NAND Flash, χωρίς την χρήση NOR Flash. Οι συγκεκριμένοι επεξεργαστές αποτελούν μια πρακτική λύση όταν το κόστος, η χωροταξία και ο διαθέσιμος αποθηκευτικός χώρος της συσκευής, αποτελούν σημαντικοί παράγοντες. Έτσι, η δυνατότητα της λειτουργίας XiP παύει να αποτελεί σκέψη, στην σχεδίαση ενθυλακωμένων συστημάτων με μνήμες NAND Flash. Στον παρακάτω πίνακα 2.2 φαίνονται συνοπτικά τα πλεονεκτήματα και τα μειονεκτήματα του κάθε τύπου μνήμης.

	NAND	**NOR**
Πλεονεκτήματα	Fast programs	Random access
	Fast erases	Byte programs
Μειονεκτήματα	Slow random access	Slow programs
	Byte programs difficult	Slow erases
Εφαρμογές	File applications, any large sequential data	Replacement of EEPROM
	Voice, data, video recorder	Execute directly from non-volatile memory

Πίνακας 2.2: Πλεονεκτήματα-Μειονεκτήματα NAND, NOR Flash

2.4 Χρόνοι Τυχαίας Προσπέλασης

Οι χρόνοι τυχαίας προσπέλασης για τις μνήμες NOR Flash έχουν καθοριστεί στα 0,075μs, ενώ για τις μνήμες NAND Flash για την προσπέλαση του πρώτου byte απαιτείται αισθητά μεγαλύτερος χρόνος, της τάξης των 25μs. Παρόλα αυτά, μόλις επιτευχθεί η ανάγνωση του πρώτου byte, τα υπόλοιπα μπορούν να διαβαστούν από την μνήμη NAND με ρυθμό 0,025μs ανά byte. Αυτό μας οδηγεί σε ένα εύρος της τάξης των 26MB/s για 8-bit I/O's και 41MB/s για 16-bit I/O's. Στον παρακάτω πίνακα 2.3 περιγράφονται συνοπτικά οι χρόνοι προσπέλασης για μνήμη 2Gb με μέγεθος block των 128KB.

χαρακτηριστικά	NAND Flash	NOR Flash
Random Access READ speed	25μs(first byte) 0,025μs per remaining bytes	0,075μs

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

READ speed	26MB/s (x8) or 41MB/s (x16)	31MB/s (x8) or 62MB/s (x16)
Random WRITE speed	220μs/2112bytes	128μs/32bytes
WRITE speed	7,5MB/s	0,250MB/s
Erase Block Size	128KB	128KB
Erase time per block	500μs	1sec

Πίνακας 2.3: Χρόνοι Προσπέλασης NAND, NOR Flash

2.5 Σχεδιαστικά οφέλη

Το πραγματικό όφελος από την χρήση μνήμης NAND Flash σε μια ηλεκτρονική συσκευή, είναι οι υψηλότεροι χρόνοι προγραμματισμού και διαγραφής των δεδομένων που επιτυγχάνονται καθώς επίσης και η δυνατότητας συνεχής εγγραφής δεδομένων με ρυθμό 7MB/s. Η ταχύτητα διαγραφής ενός block είναι στα 500μs για τις μνήμες NAND Flash, ενώ ο αντίστοιχος χρόνος για μνήμες NOR Flash αυξάνεται στο 1sec. Φαίνεται ξεκάθαρα λοιπόν ότι οι μνήμες NAND Flash προσφέρουν ασυναγώνιστα πλεονεκτήματα παρόλο το μειονέκτημά τους ότι προσφέρουν χαμηλούς χρόνους τυχαίας προσπέλασης, το οποίο όπως αναφέραμε μπορεί να παρακαμφθεί κάνοντας χρήση της μεθόδου προσπέλασης “shadowing”.

2.6 Δομικές Διαφορές

Ξεκινώντας με τον αριθμό των επαφών του κάθε ολοκληρωμένου, παρατηρούμε ότι οι μνήμες NAND Flash διαφέρουν σημαντικά και πλεονεκτούν έναντι των μνημών NOR Flash. Οι μνήμες NOR Flash απαιτούν σχεδόν 44 I/O επαφές για 16-bit συσκευή, ενώ οι μνήμες NAND Flash απαιτούν μόνο 23 επαφές για μια ίδιων χαρακτηριστικών συσκευή. Η πολυπλεξία των εντολών, των διευθύνσεων και των γραμμών δεδομένων επιτρέπει την μείωση των επαφών στις μνήμες NAND Flash σχεδόν κατά 45%. Ένα επιπλέον πλεονέκτημα της μεθόδους πολυπλεξίας είναι ότι οι υψηλότερης χωρητικότητας μνήμες NAND Flash μπορούν να υποστηριχθούν στο ίδιο υλικό πακέτο χωρίς να χρειάζεται επανασχεδιασμός του τυπωμένου κυκλώματος (printed circuit board – PCB). Η κοινή βάση αρχιτεκτονικής TSOP-1 χρησιμοποιείται εδώ και πολλά χρόνια, επιτρέποντας στους κατασκευαστές την υλοποίηση κατασκευής, με υψηλότερη χωρητικότητα, στην ίδια πλακέτα. Ένα ολοκληρωμένο μνήμης των 2Gb μπορεί να αντικατασταθεί με ένα των 8Gb χωρίς αλλαγές στην πλακέτα, την αρχιτεκτονική της και στα υλικά. Στο παρακάτω πίνακα 2.4, φαίνονται ενδεικτικά οι επαφές που χρησιμοποιούνται σε κάθε τύπο μνήμης.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

NAND Flash: 23 pins (x16bit)		NOR Flash: 44 pins	
I/O device-type interface composed of:		Random-access interface composed of:	
CE#	Chip enable	CE#	Chip enable
WE#	Write enable	WE#	Write enable
RE#	Read enable	OE#	Output enable
CLE	Command latch enable	D[15:0]	Data bus
ALE	Address latch enable	A[23:0]	Address bus
I/O[7:0]	Data bus [15:0] for 16bit	WP#	Write protect
WP#	Write protect		
R/B#	Read/Busy		

Πίνακας 2.4: Ακροδέκτες NAND, NOR Flash

Στην συνέχεια θα εξηγήσουμε τα βασικά χαρακτηριστικά της τεχνολογίας μνήμης Flash σε φυσικό επίπεδο και θα αναλύσουμε τα χαρακτηριστικά της μνήμης τεχνολογίας NAND Flash και τον τρόπο λειτουργίας της.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Κεφάλαιο 3ο

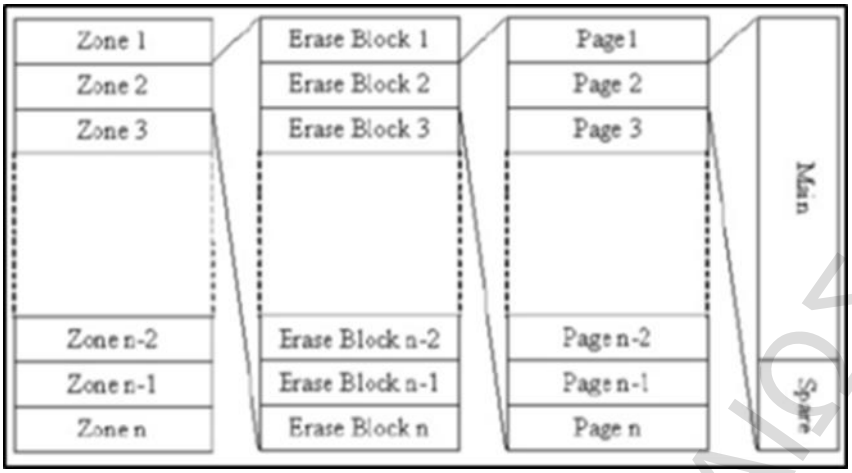
Τεχνολογία NAND Flash Memory

3.1 Μνήμη NAND Flash

Ο φυσικός μηχανισμός για την αποθήκευση δεδομένων στις μνήμες flash, όπως προαναφέραμε στην εισαγωγή, βασίζεται στην αποθήκευση ηλεκτρικής ενέργειας-φορτίου, στην πύλη ενός τρανζίστορ, το οποίο λειτουργεί σαν διακόπτης. Το ηλεκτρικό φορτίο μπορεί να αποθηκευτεί για μεγάλες χρονικές περιόδους, χωρίς την χρήση εξωτερικής τροφοδοσίας, και θα διαρρεύσει με το πέρασμα του χρόνου από φυσικά αίτια. Η διατήρηση των δεδομένων που προδιαγράφεται στις συγκεκριμένες μνήμες flash κυμαίνεται από 10 έως 100 έτη.

Οι μνήμες flash μπορούν να γράφονται ανά byte, όπως οι μνήμες EEPROM, αλλά πρέπει να διαγράφονται ανά μπλοκ (block) κάθε φορά, προκειμένου να ξαναγραφτούν. Η διαδικασία διαγραφής έχει ως αποτέλεσμα την δημιουργία ενός μπλοκ με άσσους 1(φυσική κατάσταση HIGH για τα τρανζίστορ). Στις μνήμες NAND Flash, τα διαγραμμένα μπλοκ χωρίζονται σε σελίδες (pages) των 32 ή πολλαπλάσιων τεμαχίων, ανά μπλοκ. Οι σελίδες είναι συνήθως πολλαπλάσια των 512 bytes σε μέγεθος ώστε να εξομοιώνουν τα 512 byte που συναντούνται στα τμήματα (sectors) των αρχείων συστήματος που συναντάμε στα μαγνητικά μέσα αποθήκευσης. Επιπρόσθετα, κάθε σελίδα περιλαμβάνει κι έναν επιπλέον αριθμό byte, τα επονομαζόμενα “meta-data”, τα οποία χρησιμοποιούνται για την αποθήκευση βοηθητικών δεδομένων (spare area ή out of band area). Επίσης ορισμένες μνήμες flash χρησιμοποιούν την ιδέα των ζωνών (zones). Μια ζώνη είναι μια ομάδα από μπλοκ, συνήθως από 16 μέχρι 1024. Σε αντίθεση με τα μπλοκ και τις σελίδες, μια ζώνη είναι ένα λογικό σενάριο και δεν υπάρχει φυσική απεικόνιση στην δομή της μνήμης. Στην εικόνα 3.1 φαίνεται ένα παράδειγμα λογικής ανατομίας της μνήμης.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ



Εικόνα 3.1: Λογική Ανατομία Μνήμης NAND Flash

Φυσικά Χαρακτηριστικά.

Κάθε σελίδα περιλαμβάνει μια περιοχή από bytes που συχνά αναφέρεται ως πλεονάζουσα ή επιπλέον περιοχή (redundant or spare area). Στο πίνακα 3.1 παρουσιάζονται παραδείγματα πλεονάζουσας περιοχής (spare area) για διαφορετικού μεγέθους σελίδες (pages).

Page size	Spare area size	Total page size	Pages/Block	Block size
256	8	264	32	8448
512	16	528	32	16896
2048	64	2112	64	135168

Πίνακας 3.1: Size of NAND Flash

Η πλεονάζουσα περιοχή μπορεί να περιέχει πληροφορίες της κατάστασης του μπλοκ ή της σελίδας. Για παράδειγμα, όταν ένα μπλοκ χαρακτηριστεί ελαττωματικό, θα σημειωθεί στην πλεονάζουσα περιοχή. Επίσης η πλεονάζουσα περιοχή μπορεί να περιέχει δεδομένα ελέγχου σφάλματος (Error Correction Code – ECC) για την διάγνωση σφαλμάτων σε δεδομένα μιας σελίδας. Με τον αλγόριθμο ECC, σφάλματα του ενός bit μπορούν να διορθωθούν και εκ των υστέρων το μπλοκ να μαρκαριστεί ως ελαττωματικό. Τέλος, στον πλεονάζουσα χώρο μπορούν να αναγράφονται πληροφορίες απαραίτητες για την φυσική σε λογική αντιστοίχηση των διευθύνσεων μνήμης.

Η διαγραφή ενός μπλοκ προκαλεί την επιδείνωση της αντοχής του. Τα μπλοκ μπορούν να διαγραφτούν από 10^4 έως 10^6 φορές, πριν τα bit στο μπλοκ γίνουν μη εγγράψιμα και παραμείνουν μόνιμα σε κατάσταση μηδενικών 0 (φυσική κατάσταση LOW για τα τρανζίστορ) και το μπλοκ χαρακτηριστεί ελαττωματικό. Οι μνήμες NAND Flash συνήθως

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

περιλαμβάνουν ελαττωματικά μπλοκ ακόμη κι όταν παραδίδονται ολοκαίνουργιες από το εργοστάσιο. Για το λόγο αυτό, στα εγχειρίδια χρήσης των μνημών NAND Flash, αναφέρεται ο ελάχιστος αριθμός καλής κατάστασης μπλοκ της μνήμης, που εγγυάται ο κατασκευαστής. Τυπικά, το ποσοστό αυτό κυμαίνεται στην τάξη του 98% των μπλοκ που βρίσκονται σε καλή κατάσταση και έτοιμα προς χρήση. Τα ελαττωματικά μπλοκ αναφέρονται στην πλεονάζουσα περιοχή.

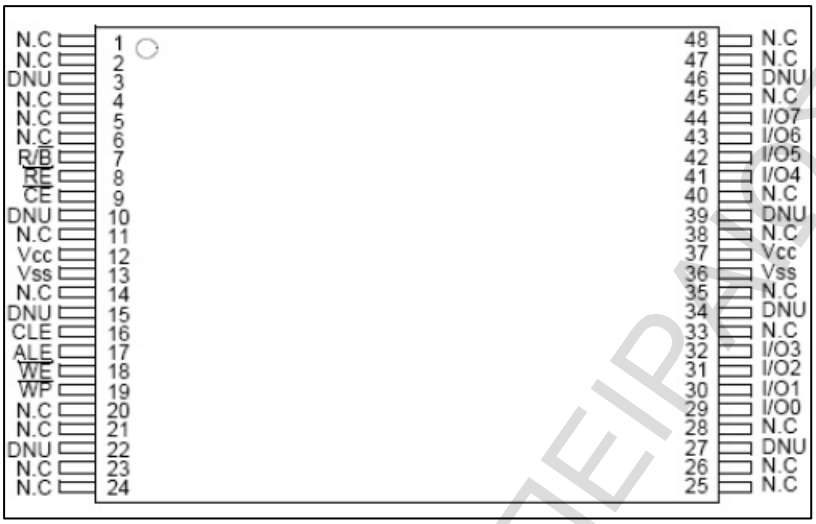
Προκειμένου να ομαλοποιηθεί η εγγραφή των μπλοκ όσο καλύτερα γίνεται κατά μήκος όλων των φυσικών μπλοκ που υπάρχουν στην μνήμη, οι κατασκευαστές ανέπτυξαν μια μέθοδο που την αποκαλούν “wear leveling”. Η ιδέα ήταν ότι εξαπλώνοντας την φθορά που προκαλείται από την συνεχή διαγραφή των μπλοκ κατά μήκος όλης της χωρητικότητας της μνήμης flash, θα μπορέσουν να αυξήσουν την συνολική διάρκεια της μνήμης, και το πέτυχαν. Ο αλγόριθμος “wear leveling” αποτελεί ευαίσθητο και αξιόπιστο αγαθό για τους κατασκευαστές και οποιεσδήποτε ερωτήσεις και πληροφορίες προς τους κατασκευαστές για την δομή του, παραμένουν αναπάντητες.

Παρόλα αυτά, για την αναδόμηση των δεδομένων στις μνήμες flash, δεν είναι απαραίτητο να γνωρίζουμε πως ο αλγόριθμος δημιούργησε το φυσικό τμήμα της μνήμης, το οποίο αντιγράφεται από το ολοκληρωμένο. Το μόνο που απαιτείται είναι να γνωρίζουμε πώς να επαναδημιουργήσουμε την σωστή σειρά των φυσικών μπλοκ, ώστε να δημιουργηθεί ένα αντίγραφο της λογικής δομής της μνήμης, προσπελάσιμο από το ανώτερο σύστημα διαχείρισης αρχείων. Με άλλα λόγια, ο αλγόριθμος “wear leveling” μπορεί να θεωρηθεί ως μια δυναμική διεργασία συνεχής ανασύνταξης των σελίδων ή/και των μπλοκ της μνήμης, ώστε να επιτευχθεί μεγαλύτερη διάρκεια ζωής του ολοκληρωμένου.

Η ηλεκτρική διεπαφή των μνημών NAND Flash διαφέρει από αυτές των συνηθισμένων RAM. Οι μνήμες NAND περιλαμβάνουν ένα δίαυλο πολυπλεξίας των γραμμών διευθύνσεων και δεδομένων, ο οποίος περιγράφεται γενικά ως γραμμές εισόδου/εξόδου (Input/Output – I/O’s). Οι συγκεκριμένες γραμμές έχουν συνήθως εύρος 8 ή 16 bit. Στην εικόνα 3.2 φαίνετε ένα παράδειγμα των ηλεκτρικών επαφών μιας μνήμης NAND Flash με την περιγραφή των επαφών για καθεμία από αυτές. Η προσπέλαση των δεδομένων στις μνήμες NAND Flash επιτυγχάνεται στέλνοντας πρώτα στις γραμμές I/O, την διεύθυνση μνήμης στις οποίες έχουν καταχωρηθεί τα δεδομένα προς ανάκτηση. Όταν η τιμή της διεύθυνσης μνήμης είναι μεγαλύτερη από το εύρος των τιμών που μπορούν να καλύψουν οι συνδυασμοί των 8 ή 16 bit γραμμών του ολοκληρωμένου, η διεύθυνση μνήμης προωθείτε σειριακά στο ολοκληρωμένο μέσα σε 3 ή 5 κύκλους μηχανής διευθύνσεων. Όταν τελειώσει η

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

καταχώρηση της διεύθυνσης μνήμης στο ολοκληρωμένο, τα δεδομένα μπορούν να εξαχθούν από τις ίδιες γραμμές εισόδου/εξόδου (I/O). Μια τυπική διαδικασία πρόσβασης στα δεδομένα της μνήμης παρουσιάζεται στον πίνακα 3.2.



Εικόνα 3.2: Ηλεκτρικές Επαφές Μνήμης NAND Flash

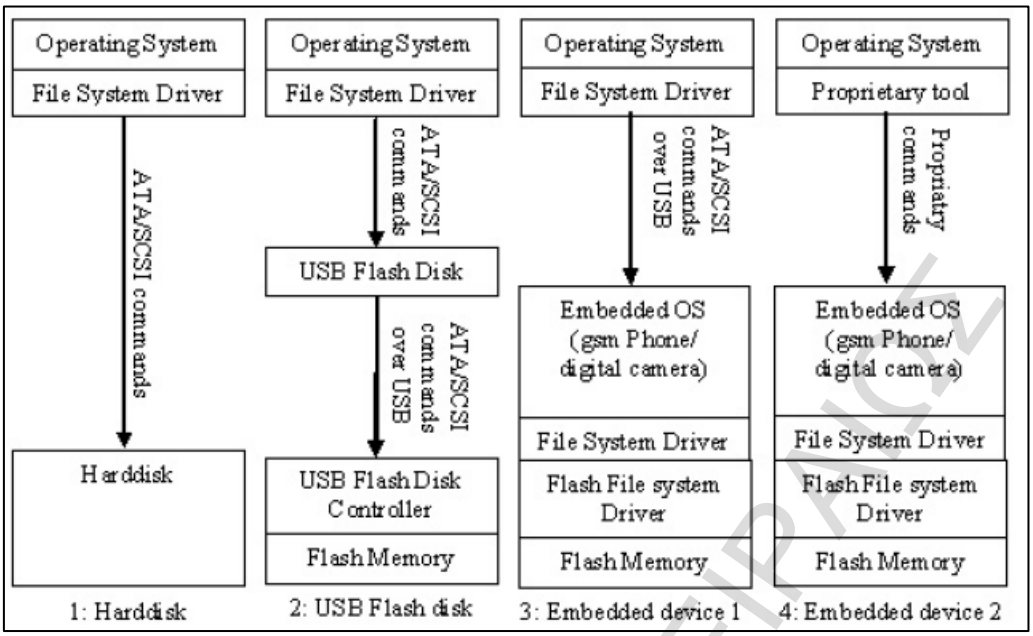
Cycle	I/O_0	I/O_1	I/O_2	I/O_3	I/O_4	I/O_5	I/O_6	I/O_7	
1	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	Column Address
2	A_8	A_9	A_{10}	A_{11}	X	X	X	X	Column Address
3	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	Row Address
4	A_{20}	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	A_{26}	A_{27}	Row Address

Πίνακας 3.2: Κύκλοι Διευθυνσιοδότησης

Λογικά Χαρακτηριστικά.

Υπάρχουν πάρα πολλοί μέθοδοι, με τους οποίους οι μνήμες flash μπορούν να χρησιμοποιηθούν για την αποθήκευση αρχείων, σε ενθυλακωμένα συστήματα (embedded systems). Τρείς συνηθισμένες μέθοδοι παρουσιάζονται και εξηγούνται στην συνέχεια. Ένα απλό διάγραμμα που περιλαμβάνει τα στοιχεία που συμμετέχουν στην επικοινωνία του Λειτουργικού Συστήματος (OS) με την μνήμη flash παρουσιάζεται στην εικόνα 3.3. Ως στοιχείο αναφοράς παρουσιάζεται και ένα διάγραμμα με σκληρό δίσκο (HDD). Στην περίπτωση του σκληρού δίσκου το Λειτουργικό Σύστημα έχει πρόσβαση στα δεδομένα του σκληρού, μέσω του οδηγού του συστήματος αρχείων (file system driver – FSD). Οι οδηγοί του συστήματος αρχείων προωθούν εντολές στο σκληρό δίσκο, για παράδειγμα την ATA εντολή ανάγνωση τμήματος (Read Sector) για να διαβαστούν τα δεδομένα από μια Λογική Διεύθυνση ενός Μπλοκ (Logical Block Address - LBA).

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ



Εικόνα 3.3: NAND Flash Connection Types

Στην περίπτωση ενός USB Flash stick η συσκευή παρουσιάζει τον εαυτό της ως μέσο αποθήκευσης. Μετά την προσάρτησή της, το Λειτουργικό Σύστημα μπορεί να έχει πρόσβαση στην συσκευή. Σε πλατφόρμες που βασίζονται σε Windows-Intel για παράδειγμα, ένας νέος οδηγός συσκευής δημιουργείτε με την τοποθέτηση του USB stick στην USB θύρα επικοινωνίας του μηχανήματος. Η πρόσβαση στα δεδομένα επιτυγχάνεται με την προώθηση των εντολών του οδηγού FSD διαμέσου του καναλιού επικοινωνίας με την USB θύρα προς την συσκευή. Ο ελεγκτής (controller) της συσκευής USB stick που βρίσκεται ενσωματωμένος στην πλακέτα, αναλαμβάνει την διαχείριση των σημάτων και επιτυγχάνεται η πρόσβαση στα δεδομένα. Για την διαχείριση των ειδικών ιδιοτήτων της μνήμης Flash ο ελεγκτής γενικά αποθηκεύει επιπλέον πληροφορίες (meta data) με τα δεδομένα. Για παράδειγμα ο λογικός πίνακας διευθυνσιοδότησης στις εντολές ATA, δεν είναι ο ίδιος με τις φυσικές διευθύνσεις στο ολοκληρωμένο της μνήμης, όπου αποθηκεύονται τα δεδομένα. Οι απαραίτητες πληροφορίες για την αντιστοίχηση του λογικού πίνακα διευθύνσεων (LBA) με αυτές των φυσικών διευθύνσεων της μνήμης, είναι επίσης αποθηκευμένες σε αυτή.

Στα ενθυλακωμένα συστήματα, όπως είναι ένα κινητό τηλέφωνο ή μια ψηφιακή φωτογραφική μηχανή, μπορεί να χρησιμοποιηθεί παρόμοιος μηχανισμός με αυτόν που παρουσιάζεται στην εικόνα 3.3 περίπτωση 3. Όταν η συσκευή συνδεθεί στο σύστημα, το Λειτουργικό Σύστημα - φιλοξενίας μπορεί να έχει πρόσβαση στο σύστημα διαχείρισης αρχείων της μνήμης, χρησιμοποιώντας γενικές εντολές πρόσβασης δίσκου από τις

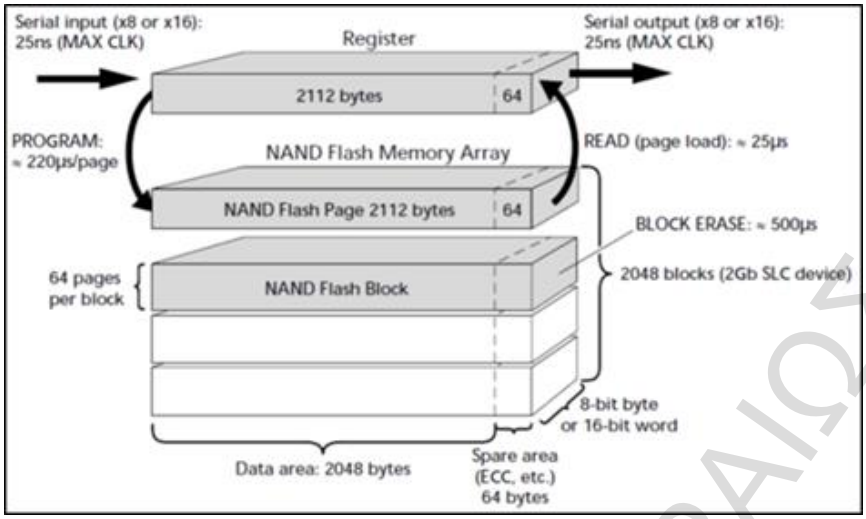
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

εγκατεστημένες βιβλιοθήκες. Το Λειτουργικό Σύστημα της συσκευής λαμβάνει τις εντολές προσπέλασης δίσκου από το Λειτουργικό Σύστημα που το φιλοξενεί, τις εκτελεί και επιστρέφει τα ζητούμενα δεδομένα. Σε αυτή την περίπτωση το Λειτουργικό Σύστημα – φιλοξενίας δεν αντιλαμβάνεται καμία διαφορά, αν η συσκευή είναι ένα σκληρός δίσκος ή μια διαφορετική μονάδα αποθήκευσης προσαρτημένη σε αυτό. Τέλος, στην εικόνα 3.3 περίπτωση 4 παρουσιάζεται η προσπέλαση των δεδομένων στο ενθυλακωμένο σύστημα διαμέσου μιας κατάλληλης εφαρμογής εγκατεστημένης στο Λειτουργικό Σύστημα – φιλοξενίας. Η εφαρμογή αναλαμβάνει την επικοινωνία με το Λειτουργικό Σύστημα της συσκευής και κατάλληλες εντολές στέλνονται προς αυτό για την προσπέλαση των δεδομένων. Παράδειγμα θα μπορούσε να αποτελέσει η εφαρμογή ActiveSync της Microsoft η οποία κάνει χρήση του Remote Application Programming Interface (RAPI) για την πρόσβαση στα δεδομένα μνήμης του ενθυλακωμένου συστήματος.

3.2 Αρχιτεκτονική και Βασικές Λειτουργίες SLC

Στη συνέχεια θα μελετήσουμε την αρχιτεκτονική της μνήμης NAND Flash μελετώντας ως παράδειγμα μία NAND Flash χωρητικότητας 2GB. Η συγκεκριμένη μνήμη είναι οργανωμένη σε 2048 block και κάθε block έχει 64 pages. Το μέγεθος κάθε σελίδας ορίζεται στην χωρητικότητα των 2112 bytes εκ των οποίων τα 2048 bytes αποτελούν τα κύρια δεδομένα (main data) και τα υπόλοιπα 64 byte βοηθητικά δεδομένα (meta data). Τα βοηθητικά δεδομένα χρησιμοποιούνται για την αποθήκευση του αλγόριθμου ECC, wear leveling και άλλων λειτουργιών του λογισμικού του ελεγκτή (controller) της μνήμης. Οι μνήμες NAND Flash παρέχονται με δίαυλο επικοινωνίας των 8 ή 16 bit. Στις 16bit συσκευές η διευθυνσιοδότηση και η εκτέλεση των εντολών γίνεται στις πρώτες 8 γραμμές του διαύλου επικοινωνίας [7:0] ενώ οι υπόλοιπες 8 γραμμές χρησιμοποιούνται για την εισαγωγή/εξαγωγή των δεδομένων. Στην εικόνα 3.4 φαίνεται η φυσική δομή της μνήμης των 2GB.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ



Εικόνα 3.4: Φυσική Δομή Μνήμης NAND Flash 2GB

Για την διαγραφή ενός block απαιτείται χρόνος 500μs. Μόλις τα δεδομένα φορτωθούν στο καταχωρητή ο προγραμματισμός μιας σελίδας απαιτεί χρόνο 220μs. Η διαδικασία ανάγνωσης της σελίδας (PAGE READ) απαιτεί χρόνο 25μs, κατά τον οποίο η σελίδα θα προσπελαθεί από τον πίνακα μνήμης και θα φορτωθεί σε έναν 16.896 bit (2112 bytes) καταχωρητή. Στην συνέχεια τα δεδομένα του καταχωρητή είναι διαθέσιμα στον χρήστη για εξαγωγή.

Επιπρόσθετα με τον γενικό δίαυλο επικοινωνίας (I/O bus), η διεπαφή της μνήμης NAND Flash αποτελείται και από 6 ακροδέκτες ελέγχου. Στον πίνακα 3.3 περιγράφεται η λειτουργία καθενός ξεχωριστά. Το σύμβολο (#) υποδηλώνει ότι το σήμα σε ηρεμία είναι σε κατάσταση LOW.

Symbol	Signal	Description
ALE	Address Latch Enable	Όταν το σήμα ALE είναι σε κατάσταση HIGH, οι διευθύνσεις μνήμης προωθούνται στο καταχωρητή μνήμης με την άνοδο του σήματος WE#.
CE#	Chip enable	Όταν το σήμα CE **δεν** είναι σε κατάσταση HIGH η μνήμη NAND Flash παραμένει σε κατάσταση ηρεμίας (standby mode).
CLE	Command Latch Enable	Όταν το σήμα CLE είναι σε κατάσταση HIGH, οι εντολές μνήμης προωθούνται στον καταχωρητή εντολών με την άνοδο του σήματος WE#.
R/B#	Read/Busy	Όταν η μνήμη NAND Flash είναι απασχολημένη με μια διαδικασία READ, PROGRAM ή ERASE το σήμα είναι σε

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

		κατάσταση LOW. (απαιτείται pull-up αντίσταση).
RE#	Read Enable	Ενεργοποιεί τους buffers εξόδου δεδομένων.
WE#	Write Enable	Είναι υπεύθυνο για το συγχρονισμό (clocking) των σημάτων δεδομένων, εντολών και διευθύνσεων κατά την διαδικασία λειτουργίας.

Πίνακας 3.3: Περιγραφή Ακροδεκτών Μνήμης NAND Flash

Τα δεδομένα ολισθαίνουν προς ή από τους καταχωρητές της μνήμης NAND Flash ανά 8 ή 16 bit κάθε φορά. Κατά την διαδικασία προγραμματισμού (PROGRAM), τα δεδομένα προς προγραμματισμό ολισθαίνουν στον καταχωρητή δεδομένων με την άνοδο του σήματος στον ακροδέκτη WE# (write enable). Ειδικές εντολές χρησιμοποιούνται για την τυχαία προσπέλαση των δεδομένων ή την μετακίνηση των δεδομένων μέσα στους καταχωρητές ώστε να επιτευχθεί η τυχαία προσπέλαση. Η διαδικασία εξαγωγής των δεδομένων από τον καταχωρητή πραγματοποιείται με παρόμοια διαδικασία, ενεργοποιώντας τον ακροδέκτη RE# (read enable), ο οποίος είναι υπεύθυνος για την έξοδο των δεδομένων και την μετάβαση των δεδομένων της επόμενης διεύθυνσης μνήμης. Τα σήματα στους ακροδέκτες WE# και RE# μπορούν να εκτελούνται με ρυθμό 25μs ανά μεταφορά.

Όταν οι ακροδέκτες RE# ή CE# δεν είναι σε κατάσταση LOW, οι μνήμες εξόδου (buffers) είναι στην επονομαζόμενη τριπλή-κατάσταση (tri-stated). Ο συγκεκριμένος συνδυασμός των ακροδεκτών ενεργοποιεί τις μνήμες εξόδου, επιτρέποντας στην μνήμη NAND Flash να μοιράζεται τα δεδομένα στο δίαυλο με άλλους τύπους μνήμης όπως NOR Flash, SRAM, DRAM. Η συγκεκριμένη λειτουργία πολλές φορές περιγράφεται και ως “chip enable don’t care”.

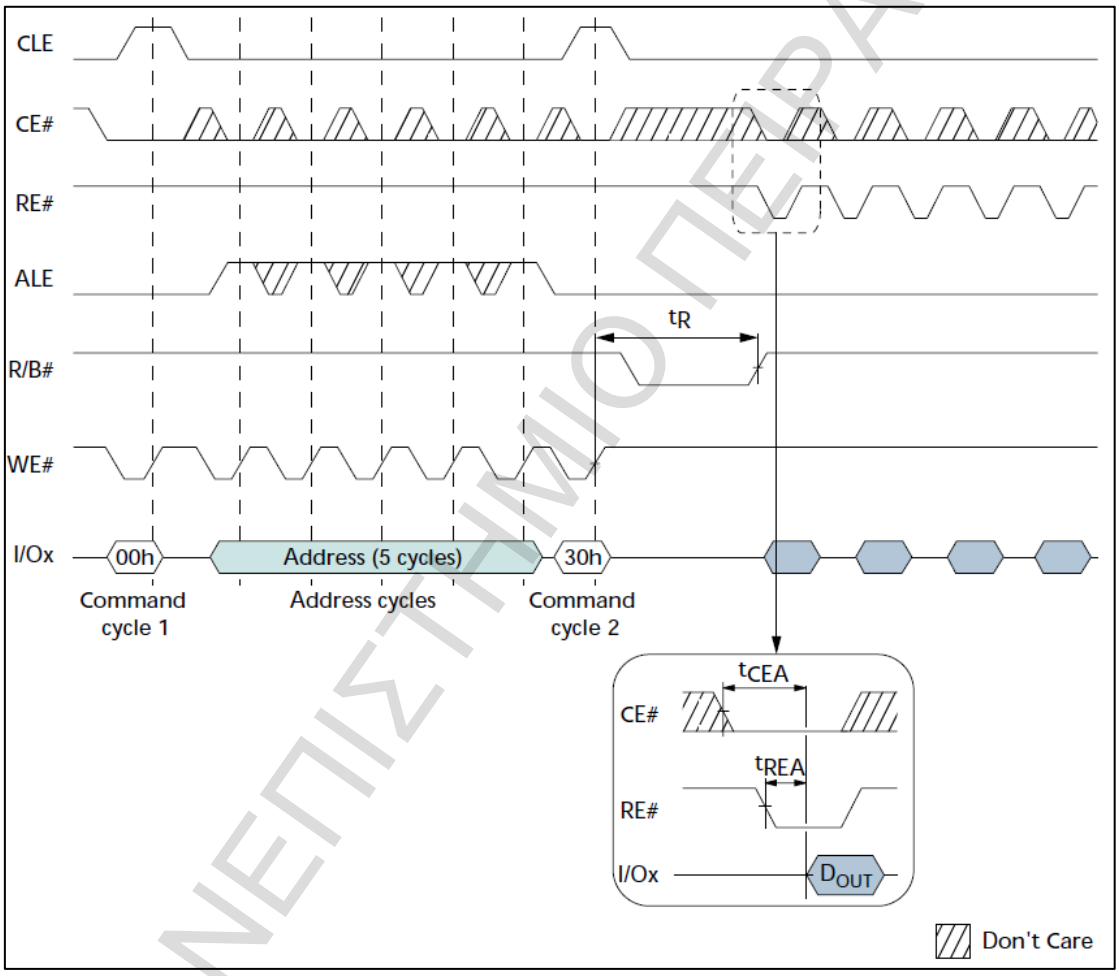
Όλες οι λειτουργίες της μνήμης NAND Flash ξεκινάνε υλοποιώντας ένα κύκλο εντολών. Η διαδικασία πραγματοποιείται τοποθετώντας την δεκαεξαδική τιμή της εντολής στο δίαυλο δεδομένων (I/O [0:7]), οδηγώντας τον ακροδέκτη CE# σε κατάσταση LOW και τον ακροδέκτη CLE σε κατάσταση HIGH, και τέλος εναλλάσσοντας την κατάσταση του σήματος στον ακροδέκτη WE# όσες φορές απαιτείται. Τόσο οι εντολές, όσο και οι διευθύνσεις αλλά και τα δεδομένα πρέπει να τοποθετούνται στο δίαυλο κατά την άνοδο του σήματος στον ακροδέκτη WE#. Οι περισσότερες εντολές απαιτούν ένα αριθμό από κύκλους διευθύνσεων ακολουθούμενες από ένα δεύτερο κύκλο εντολών. Με εξαίρεση την εντολή επανεκκίνησης (RESET) και ανάγνωση κατάστασης (READ STATUS), νέες εντολές δεν πρέπει να εκτελούνται όταν η μνήμη είναι απασχολημένη. Στον πινάκα 3.4 φαίνεται η μέθοδος διευθυνσιοδότησης στο δίαυλο, ενώ στην εικόνα 3.5 φαίνεται ένα παράδειγμα λειτουργίας.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Cycle	I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	I/O0
First	CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0
Second	LOW	LOW	LOW	LOW	CA11	CA10	CA9	CA8
Third	BA7	BA6	PA5	PA4	PA3	PA2	PA1	PA0
Fourth	BA15	BA14	BA13	BA12	BA11	BA10	BA09	BA08
Fifth	LOW	LOW	LOW	LOW	LOW	LOW	LOW	BA16

Πίνακας 3.4: Μέθοδος Διευθυνσιοδότησης στο Δίαυλο

CA= Column Address, BA= Block Address, PA= Page Address



Εικόνα 3.5: Εκτέλεση Λειτουργίας

3.3 Εντολές (Commands)

Για την εκτέλεση των εντολών στην μνήμη NAND Flash οι ακροδέκτες CE# και ALE πρέπει να είναι σε κατάσταση LOW, ο ακροδέκτης CLE σε κατάσταση HIGH, ενώ οι κύκλοι εγγραφής συγχρονίζονται με το σήμα στον ακροδέκτη WE#. Για την διευθυνσηοδότηση της μνήμης NAND Flash οι ακροδέκτες CE# και CLE πρέπει να είναι σε κατάσταση LOW, ο ακροδέκτης ALE σε κατάσταση HIGH, ενώ οι κύκλοι εγγραφής συγχρονίζονται με το σήμα στον

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ακροδέκτη WE#. Όταν η μνήμη είναι απασχολημένη, μόνο δυο εντολές μπορούν να εκτελεστούν: RESET και READ STATUS. Στον πίνακα 3.6 φαίνονται οι εντολές για κάθε λειτουργία της μνήμης NAND Flash καθώς και οι κύκλοι που απαιτούνται για την εκτέλεσή τους σύμφωνα με τις προδιαγραφές που έχει καθορίσει ο οργανισμός ONFI (Open Nand Flash Interface). Εν συνεχεία περιγράφεται ο τρόπος λειτουργίας ορισμένων εξ αυτών.

Command	Command Cycle 1	Number of Address Cycles	Data Cycles Required	Command Cycle 2	Valid During Busy
READ PAGE	00h	5	No	30h	No
READ PAGE CACHE SEQUENTIAL	31h	-		-	No
READ PAGE CACHE SEQUENTIAL LAST	3Fh	-	No	-	No
READ for INTERNAL DATA MOVE	00h	5	No	35h	No
RANDOM DATA READ	05h	2	No	E0h	No
READ ID	90h	1	No	-	Yes
READ STATUS	70h	-	No	-	No
PROGRAM PAGE	80h	5	Yes	10h	No
PROGRAM PAGE CACHE	80h	5	Yes	15h	No
PROGRAM for INTERNAL DATA MOVE	85h	5	Optional	10h	No
RANDOM DATA INPUT	85h	2	Yes	-	No
ERASE BLOCK	60h	3	No	D0h	No
RESET	FFh	-	No	-	Yes

Πίνακας 3.6: Εντολές NAND Flash

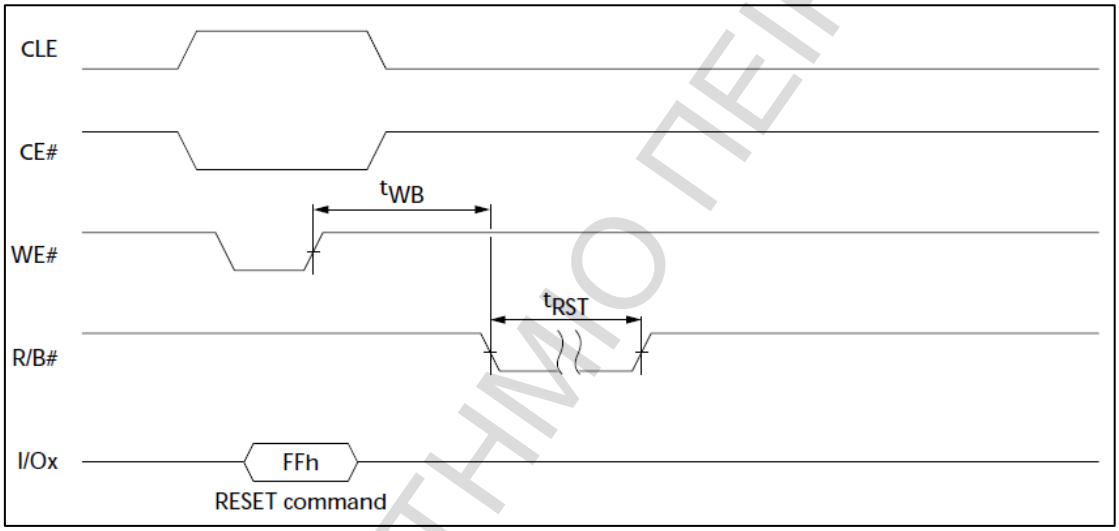
Στη συνέχεια θα περιγράψουμε ορισμένες εντολές και τον τρόπο λειτουργίας τους.

RESET Operation

Η πιο απλή εντολή στις μνήμες NAND Flash, είναι η εντολή ΕΠΑΝΕΚΙΝΗΣΗΣ (RESET-FFh). Κατά την εκτέλεση της εντολής RESET, δεν χρειάζεται να ακολουθήσει κάποια διεύθυνση μνήμης ή επιπλέον κύκλοι εγγραφής. Απλά, ενεργοποιούμε το σήμα CLE και γράφουμε στο

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

δίαυλο δεδομένων την τιμή FFh. Η εντολή RESET πρέπει να εκτελείται απευθείας μετά την τροφοδοσία της συσκευής και πριν την εκτέλεση οποιασδήποτε άλλης εντολής. Παρόλα αυτά, η εντολή RESET είναι μία από τις δυο εντολές που μπορούν να διαβιβαστούν ενώ η μνήμη NAND Flash είναι απασχολημένη. Σε αυτή την περίπτωση η εκτέλεση της εντολής RESET ακυρώνει την προηγούμενη λειτουργία της μνήμης. Αν η προηγούμενη λειτουργία ήταν η εκτέλεση μιας εντολής ΔΙΑΓΡΑΦΗΣ (ERASE) ή ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ (PROGRAM) η εντολή RESET ακυρώνει την προηγούμενη εντολή μόνιμα και η επιθυμητή λειτουργία δεν ολοκληρώνεται. Επειδή οι εντολές RESET και PROGRAM απαιτούν μεγάλους χρόνους για την εκτέλεση τους, πολλές φορές είναι επιθυμητή η ακύρωσή τους και η εκτέλεση τους μια άλλη χρονική στιγμή.

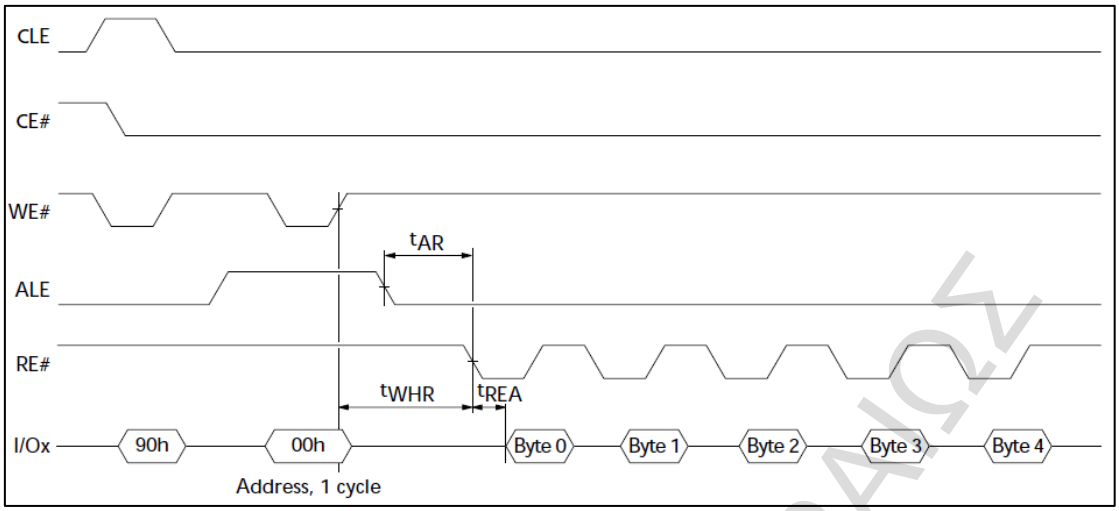


Εικόνα 3.6: Λειτουργία Επανεκκίνησης (RESET)

READ ID Operation

Η εντολή ΑΝΑΓΝΩΣΗ ΤΑΥΤΟΤΗΤΑΣ (READ ID – 90h) απαιτεί έναν κενό κύκλο διεύθυνσης (00h) αλλά δεν χρειάζεται δεύτερο κύκλο εντολής. Μετά την διαβίβαση της εντολής και της διεύθυνσης, τα δεδομένα ΤΑΥΤΟΤΗΤΑΣ μπορούν να εξαχθούν διατηρώντας τα σήματα CLE και ALE σε κατάσταση LOW και εναλλάσσοντας το σήμα RE# για κάθε byte ΤΑΥΤΟΤΗΤΑΣ. Στην εικόνα 3.7 φαίνεται η λειτουργία READ ID για την ανάγνωση ID μήκους 5 byte και στην εικόνα 3.8 φαίνεται ως παράδειγμα, ο πίνακας απάντησης (ID RESPONSE) από μια μνήμη NAND Flash της εταιρίας Micron.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ



Εικόνα 3.7: Λειτουργία Ανάγνωσης Ταυτότητας (READ ID)

READ ID Response

	Option	I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	I/O0	Value[1]
Byte 0 – Manufacturer ID										
Manufacturer	Micron	0	0	1	0	1	1	0	0	2Ch
Byte 1 – Device ID										
MT29F2G08AAD	2Gb, x8, 3V	1	1	0	1	1	0	1	0	DAh
MT29F2G16AAD	2Gb, x8, 3V	1	1	0	0	1	0	1	0	CAh
MT29F2G08ABD	2Gb, x8, 1.8V	1	0	1	0	1	0	1	0	AAh
MT29F2G16ABD	2Gb, x16, 1.8V	1	0	1	1	1	0	1	0	BAh
Byte 2										
Number of die per CE#	1							0	0	00b
Cell type	SLC					0	0			00b
Number of simultaneously programmed pages	1			0	0					01b
Interleaved operations between multiple die	Not supported		0							0b
Cache programming	Supported	1								1b
Byte value	MT29F2Gxxxxx	1	0	0	0	0	0	0	0	80h
Byte 3										
Page size	2KB							0	1	01b
Spare area size (bytes)	64B						1			1b
Block size (w/o spare)	128KB			0	1					01b
Organization	x8		0							0b
	x16		1							1b
Serial access (MIN)	25ns	1				0				1xxxb
Serial access (MIN)	35ns	0				0				0xxx0b
Byte value	MT29F2G08AAD	1	0	0	1	0	1	0	1	95h
	MT29F2G16AAD	1	1	0	1	0	1	0	1	D5h
Byte value	MT29F2G08ABD	0	0	0	1	0	1	0	1	15h
	MT29F2G16ABD	0	1	0	1	0	1	0	1	55h
Byte 4										
Reserved								0	0	00b
Planes per CE#	1					0	0			00b
Plane size	2Gb		1	0	1					101b
Reserved		0								0b
Byte value	MT29F2Gxx	0	1	0	1	0	0	0	0	50h

Notes: 1. b = binary; h = hexadecimal

Εικόνα 3.8: Απάντηση Λειτουργίας Ανάγνωσης Ταυτότητας (READ ID Response)

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

READ STATUS Operation

Η εντολή ΑΝΑΓΝΩΣΗ ΚΑΤΑΣΤΑΣΗΣ (READ STATUS – 70h), είναι η δεύτερη εντολή που μπορεί να διαβιβαστεί ενώ η μνήμη NAND Flash είναι απασχολημένη. Η συγκεκριμένη εντολή δεν απαιτεί για την εκτέλεση της ένα κύκλο διεύθυνσης ή δεύτερο κύκλο εντολής. Η κατάσταση της μνήμης NAND Flash μπορεί να παρακολουθείται ενεργοποιώντας το σήμα RE# ακολουθούμενο από την εντολή READ STATUS. Όταν η εντολή READ STATUS χρησιμοποιείται για να εξακριβωθεί πότε η μνήμη είναι έτοιμη για την εκτέλεση επόμενης εντολής, η κατάσταση της μνήμης μπορεί να αναγνωστεί επανεργοποιώντας το σήμα RE#. Διαφορετικά, το σήμα RE# μπορεί να διατηρείται σε κατάσταση LOW, αναμένοντας το κατάλληλο bit κατάστασης πριν την μετάβαση στην επόμενη λειτουργία. Η εντολή READ STATUS αναφέρει επίσης και την κατάσταση του σήματος εγγραφής-προστασίας (write-protect) αλλά και του σήματος επιτυχής-αποτυχής (pass-fail) κατάστασης των εντολών PROGRAM και ERASE. Η αναφορά του σήματος επιτυχής κατάστασης (pass status) είναι υποχρεωτική ώστε να επιβεβαιωθεί η ακεραιότητα των δεδομένων.

READ STATUS Response

SR Bit	PROGRAM PAGE	PROGRAM PAGE CACHE MODE	PAGE READ	PAGE READ CACHE MODE	BLOCK ERASE	Definition
0	Pass/fail	Pass/fail (N)	–	–	Pass/fail	0 = Successful PROGRAM/ERASE 1 = Error in PROGRAM/ERASE
1	–	Pass/fail (N - 1)	–	–	–	0 = Successful PROGRAM/ERASE 1 = Error in PROGRAM/ERASE
2	–	–	–	–	–	0
3	–	–	–	–	–	0
4	–	–	–	–	–	0
5	Ready/busy	Ready/busy[1]	Ready/busy	Ready/busy[1]	Ready/busy	0 = Busy 1 = Ready
6	Ready/busy	Ready/busy cache[2]	Ready/busy	Ready/busy cache[2]	Ready/busy	0 = Busy 1 = Ready
7	Write protect	Write protect	Write protect	Write protect	Write protect	0 = Protected 1 = Not protected
[15:8]	–	–	–	–	–	0

Notes:
1. Status register bit 5 is 0 during the actual programming operation. If cache mode is used, this bit will be 1 when all internal operations are complete.
2. Status register bit 6 is 1 when the cache is ready to accept new data. R/B# follows bit 6.

Εικόνα 3.9: Απάντηση Λειτουργίας Ανάγνωσης Κατάστασης (READ STATUS Response)

READ Operation

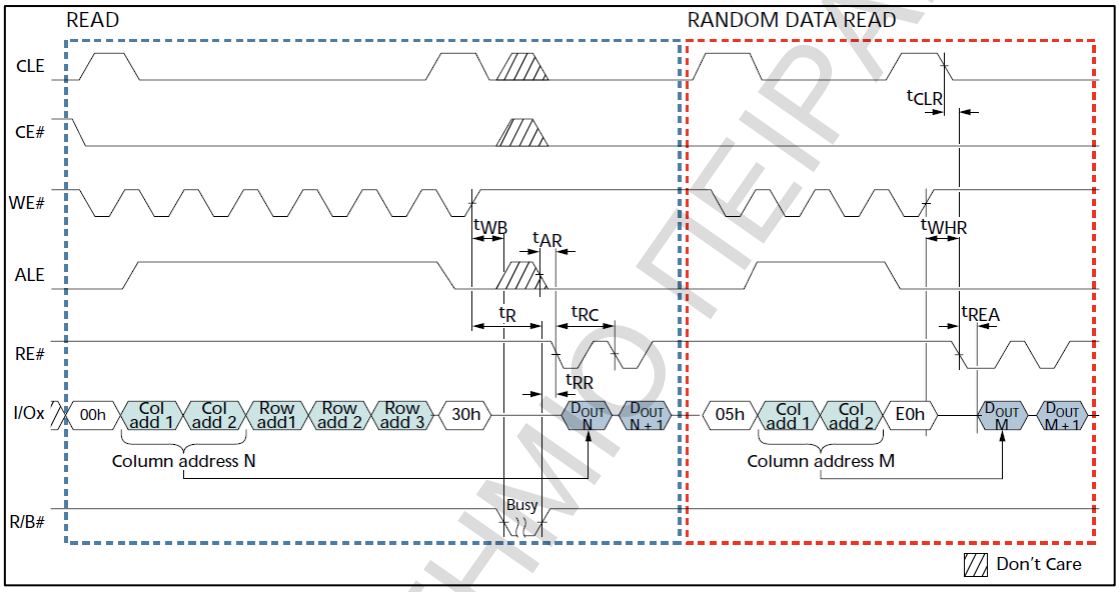
Η λειτουργία ΑΝΑΓΝΩΣΗΣ (READ), ξεκινάει διαβιβάζοντας ως εντολή την δεκαεξαδική τιμή 00h ακολουθούμενη από πέντε κύκλους διευθύνσεων και εν συνεχεία διαβιβάζοντας την δεκαεξαδική τιμή 30h για την επιβεβαίωση της ακολουθίας της εντολής. Μετά το πέρας του χρόνου που απαιτείται για την εντολή RAED και αφού έχει παρέλθει ο χρόνος μεταφοράς (tR) των 25μs, τα δεδομένα φορτώνονται στο καταχωρητή και είναι έτοιμα να εξαχθούν.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Ενεργοποιώντας το σήμα RE# και εναλλάσσοντας την κατάστασή του τα δεδομένα ολισθαίνουν ανά byte στο δίαυλο I/O. Η προσπάθεια ανάγνωσης δεδομένων πέρα από το μέγεθος μιας σελίδας, οδηγεί σε μη έγκυρα δεδομένα.

RANDOM DATA READ Operation

Ο χρήστης μπορεί να αποκτήσει απευθείας πρόσβαση σε τυχαία δεδομένα διαβιβάζοντας την δεκαεξαδική τιμή 05h, δυο κύκλους διευθύνσεων και εν συνεχεία την δεκαεξαδική τιμή E0h για την επιβεβαίωση του τέλους της ακολουθίας.

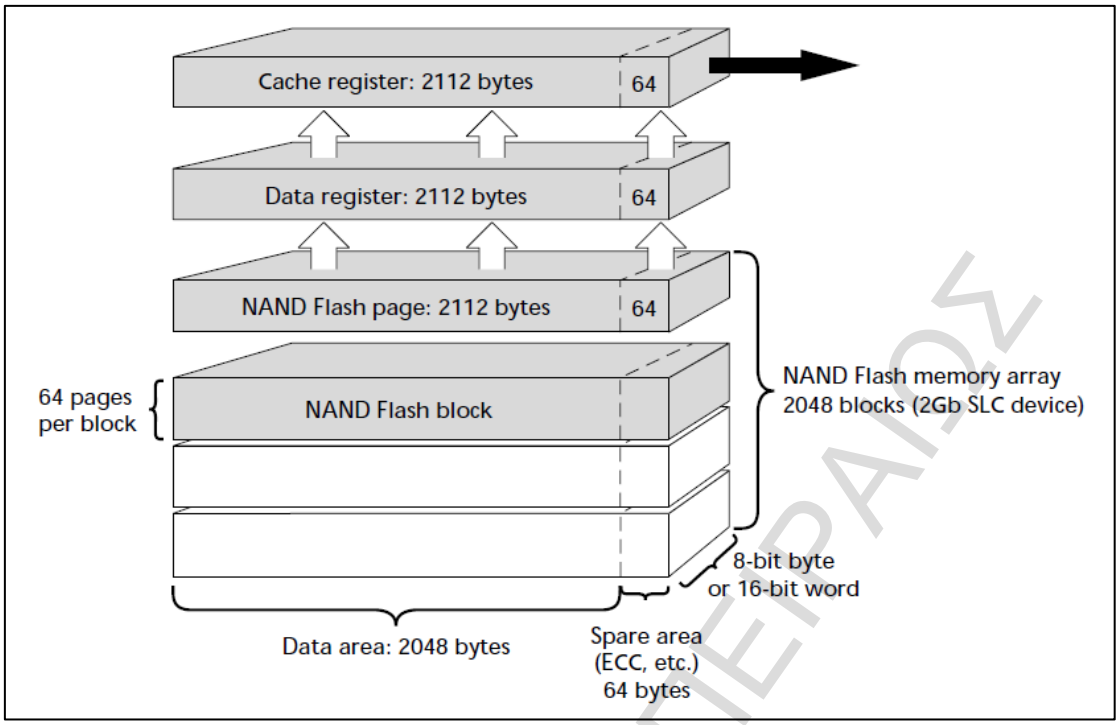


Εικόνα 3.10: Λειτουργία Ανάγνωσης Δεδομένων

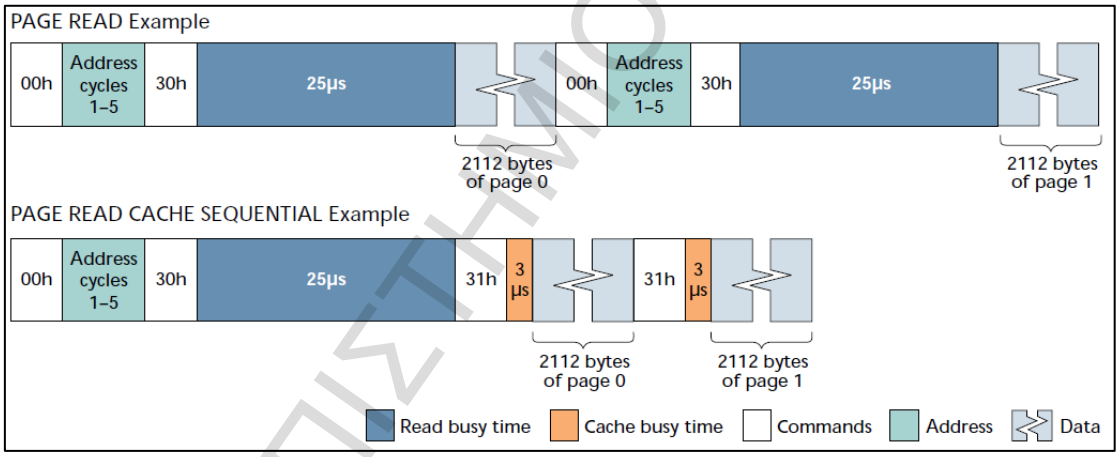
READ PAGE CACHE SEQUENTIAL Operation

Μέχρι στιγμής έχει αναφερθεί μόνο ένας καταχωρητής στις μνήμες NAND Flash. Στην πραγματικότητα υπάρχουν δυο καταχωρητές, ένας καταχωρητής δεδομένων (data register) και ένα καταχωρητής κρυφής μνήμης (cache register). Οι ιδιότητες των δυο καταχωρητών παίζουν σημαντικό ρόλο στους διάφορους τρόπους λειτουργίας της μνήμης NAND Flash. Η εντολή READ PAGE CACHE SEQUENTIAL επιτρέπει στον χρήστη την διασύνδεση με την επόμενη διεύθυνση μνήμης, την στιγμή που εξάγονται τα δεδομένα της προηγούμενης. Με την συγκεκριμένη τεχνική είναι δυνατόν να εξαλείψουμε τους χρόνους μεταφοράς (tR) που προκύπτουν κάθε φορά, κατά την εκτέλεση της εντολής READ. Με αυτή τη διαδικασία παρατηρείται βελτίωση της απόδοσης της μνήμης NAND Flash έως και 33% κατά την ανάγνωση πολλαπλών σελίδων για συσκευές με δίαυλο επικοινωνίας 8 bit και έως 40% για συσκευές με δίαυλο επικοινωνίας 16 bit.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ



Εικόνα 3.11: Καταχωρητές Μνήμης NAND Flash



Εικόνα 3.12: Χρόνοι Λειτουργίας Ανάγνωσης

3.4 Multi-Level Cell (MLC)

Οι μνήμες τύπου MLC (Multi-Level Cell), είναι ειδική κατηγορία Nand Flash μνημών οι οποίες χρησιμοποιούν κελιά-τρανζίστορ που επιτρέπουν την αποθήκευση 2 bit πληροφορίας ανά κελί, σε αντίθεση με τις παραδοσιακές μνήμες NAND Flash, οι οποίες επιτρέπουν την αποθήκευση πληροφορίας 1 bit ανά κελί. Η τεχνολογία MLC προφανώς προσφέρει πλεονεκτήματα δημιουργίας μνημών με μεγαλύτερη χωρητικότητα. Παρόλα αυτά, οι μνήμες NAND Flash τύπου MLC δεν επιτυγχάνουν την ταχύτητα και την αξιοπιστία των μνημών που βασίζονται στην τεχνολογία Single-Level Cell (SLC). Για το λόγο αυτό, μνήμες τύπου SLC χρησιμοποιούνται κατά πλειοψηφία σε εφαρμογές που απαιτούνται

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

υψηλές επιδόσεις και διάρκεια ζωής, ενώ μνήμες τύπου MLC χρησιμοποιούνται σε χαμηλού κόστους και ευρείας κατανάλωσης εφαρμογές. Στην εικόνα 3.13 παρουσιάζονται ορισμένες ειδοποιός διαφορές.

Symbol		MLC NAND 3.3V (x8)			SLC NAND 3.3V (x16/x8)			Units
		Min	Typ	Max	Min	Typ	Max	
^tPROG	Time to transfer contents of data register to the NAND Flash array	–	900	2200	–	220	600	µs
NOP	Number of partial-page programs supported per page before an ERASE is required	–	–	1	–	–	4	Cycles
^tR	Time to transfer contents of one page in the NAND Flash array to the data register	–	–	50	–	–	25	µs
Endurance with ECC and invalid block marking		5K	–	–	100K	–	–	PROGRAM/ ERASE cycles
MIN ECC required		12	–	–	1	–	–	Correctable bits per 512 bytes
N_{VB}	16Gb MLC, 4Gb SLC	1998	–	2048	2008	–	2048	Blocks

Εικόνα 3.13: Διαφορές MLC – SLC NAND Flash

3.5 Error Correction Code (ECC)

Όπως έχει προαναφερθεί, για την ορθή λειτουργία της μνήμης NAND Flash απαιτείται η υλοποίηση μεθόδου του αλγόριθμου ECC ώστε να εξασφαλισθεί η ακεραιότητα (integrity) των δεδομένων κατά την αποθήκευση. Ο αλγόριθμος ECC έχει χρησιμοποιηθεί εδώ και πολλά χρόνια τόσο σε μνήμες RAM αλλά και σε άλλα αποθηκευτικά μέσα και μπορεί να χρησιμοποιηθεί σε συσκευές που είναι ευάλωτες σε σφάλματα δεδομένων. Στην περίπτωση του παραδείγματος μας όπου η μνήμη διαθέτει 64 byte βοηθητικού χώρου ανά σελίδα, δηλαδή 16 bytes ανά 512 byte τμήματος, ο χώρος αυτός μπορεί να χρησιμοποιηθεί και για την αποθήκευση δεδομένων του αλγόριθμου ECC κι άλλον επιπρόσθετων πληροφοριών, όπως ο αλγόριθμος wear leveling ή η αντιστοίχηση των φυσικών διευθύνσεων μνήμης με τις λογικές διευθύνσεις. Ο αλγόριθμος ECC μπορεί να υλοποιηθεί σε επίπεδο υλικού ή λογισμικού, με υψηλότερη απόδοση λειτουργίας στην πρώτη επιλογή.

Κατά την διαδικασία προγραμματισμού της μνήμης, η μονάδα ECC υπολογίζει τις τιμές ECC που απαιτούνται, ανάλογα με τα δεδομένα που θα αποθηκευτούν σε κάθε τομέα. Στην συνέχεια οι τιμές αυτές εγγράφονται στην κατάλληλη περιοχή του βοηθητικού χώρου. Όταν απαιτείται ανάγνωση των δεδομένων, οι τιμές του αλγόριθμου ECC διαβάζονται και πραγματοποιείται επαλήθευση των δεδομένων που διαβάστηκαν χρησιμοποιώντας την αντίστροφη διαδικασία, επιτρέποντας την διόρθωση των δεδομένων έως ένα βαθμό. Ο αριθμός των σφαλμάτων που μπορούν να διορθωθούν, εξαρτάται από τον βαθμό διόρθωσης που χρησιμοποιήθηκε στον αλγόριθμο. Αναμενόμενο λοιπόν, η χρήση του αλγόριθμου, να αποτελεί μια ξεκάθαρη και σίγουρη λύση σε επίπεδο συστήματος.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Ο κώδικας του Hamming αποτελεί την ποιο εύκολη υλοποίηση του αλγόριθμου σε επίπεδο υλικού, αλλά μπορεί να διορθώσει σφάλματα σε επίπεδο ενός bit. Ο κώδικας Reed-Solomon παρέχει καλύτερες δυνατότητες διόρθωσης σφάλματος και χρησιμοποιείται από πολλούς ελεγκτές (controllers) στην σημερινή παγκόσμια αγορά. Τέλος, ο κώδικας BCH γίνεται όλο και πιο δημοφιλής, λόγο της καλύτερης απόδοσης του, έναντι του κώδικα Reed-Solomon. Ο πίνακας 3.7 δείχνει τον αριθμό των bit που χρειάζονται σε κάθε κώδικα, για διαφορετικά επίπεδα διόρθωσης σφάλματος. Τα σκιαγμένα κελία δείχνουν το επίπεδο υλοποίησης στην μνήμη Nand Flash του παραδείγματος.

Error Correction Level	Bits Required in the NAND Flash Spare Area		
	Hamming	Reed-Solomon	BCH
1	13	18	13
2	N/A	36	26
3	N/A	54	39
4	N/A	72	52
5	N/A	90	65
6	N/A	108	78
7	N/A	126	91
8	N/A	144	104
9	N/A	162	117
10	N/A	180	130

Πίνακας 3.7: Επίπεδο Υλοποίησης Αλγόριθμων ECC

3.6 Controller Software

Ειδικό λογισμικό απαιτείται για την διαχείριση και τον έλεγχο των block στην μνήμη NAND Flash. Επίσης, το λογισμικό διαχειρίζεται τον αλγόριθμο wear leveling και την μέθοδο αντιστοίχησης των φυσικών διευθύνσεων της μνήμης με τις λογικές διευθύνσεις, ενώ επίσης, μπορεί να παρέχει και υλοποίησης του αλγόριθμου ECC.

Είναι σημαντικό μετά από την εκτέλεση μιας διαδικασίας προγραμματισμού (PROGRAM) ή διαγραφής (ERASE), να επαληθευτεί η επιτυχής ολοκλήρωσή της. Αν η διαδικασία δεν ολοκληρωθεί με επιτυχία, το συγκεκριμένο block πρέπει να χαρακτηριστεί ελαττωματικό και να μην ξαναχρησιμοποιηθεί στο μέλλον. Τα δεδομένα πρέπει να μετακινηθούν από το ελαττωματικό block, σε ένα νέο σε καλή κατάσταση.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Η προδιαγραφή για μνήμη NAND Flash τύπου SLC χωρητικότητας 2GB, ορίζει ότι μπορεί να περιλαμβάνει μέχρι 40 ελαττωματικά block. Το συγκεκριμένο νούμερο υποδηλώνει χρόνο ζωής μνήμης, με περίπου 100.000 κύκλους προγραμματισμού/διαγραφής. Είναι συχνό φαινόμενο, οι μνήμες NAND Flash να παραδίδονται εξ αρχής από το εργοστάσιο, με έναν αριθμό ελαττωματικών block. Το λογισμικό της μνήμης διαχειρίζεται τα block που έχουν χαρακτηριστεί ελαττωματικά και δεν τα χρησιμοποιεί και επιπλέον σαρώνει όλα τα block για να ελέγξει για επιπλέον ελαττωματικά. Για τον χαρακτηρισμό ενός block ως ελαττωματικού το πρώτο byte του βοηθητικού χώρου, δηλαδή το byte στην στήλη 2048, γράφεται με την τιμή 0 (x00h). Το λογισμικό δημιουργεί έναν πίνακα με τα ελαττωματικά block και τον χρησιμοποιεί ως σημείο αναφοράς για μελλοντική χρήση.

Μεγάλη προσοχή απαιτείται ώστε να μην διαγραφούν τα στοιχεία με τα 'σημαδεμένα' - ελαττωματικά block. Κατά την εργοστασιακή παραγωγή του ολοκληρωμένου, η μνήμη NAND Flash ελέγχεται σε συγκεκριμένες θερμοκρασίες και τάσεις λειτουργίας. Ορισμένα block τα οποία έχουν χαρακτηριστεί ελαττωματικά, μπορεί να είναι προσωρινά λειτουργικά αλλά πιθανόν να αποτύχουν στο μέλλον και για το λόγο αυτό δεν πρέπει να χρησιμοποιούνται. Αν η πληροφορία των ελαττωματικών block διαγραφεί, δεν υπάρχει δυνατότητα ανάκτησής της.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Κεφάλαιο 4ο

Υλοποίηση εργαστηριακής λύσης

4.1 Προετοιμασία

Για την υλοποίηση της εργαστηριακής λύσης *ανάγνωση εικόνας δεδομένων από ολοκληρωμένο μνήμης NAND Flash*, θα βασιστούμε σε συγκεκριμένο υλικό της εταιρίας FTDI και συγκεκριμένα στο mini module που έχει αναπτύξει η εταιρία με το ολοκληρωμένο FT2232H, για την επικοινωνία με την μνήμη. Επιπλέον θα γίνει περιγραφή της βάσης που χρησιμοποιήθηκε για την προσάρτηση του ολοκληρωμένου της μνήμης καθώς και των απαραίτητων φυσικών συνδέσεων που επιτεύχθηκαν με την σχεδίαση πλακέτας (PCB). Επίσης, θα χρησιμοποιήσουμε λογισμικό ανοιχτού κώδικα ftdinandreader, κατά την εκτέλεση του οποίου θα αναγνωριστεί το μοντέλο της μνήμης, τα ιδιαίτερα χαρακτηριστικά της και ο κατασκευαστής της, και θα εξάγει σε ένα αρχείο μια πλήρη δυαδική εικόνα της μνήμης. Στο συγκεκριμένο παράδειγμα χρησιμοποιούμε ένα USB stick της **Kingston** το οποίο χρησιμοποιεί για την αποθήκευση των δεδομένων, NAND Flash μνήμη χωρητικότητας 2GB της **Toshiba**, με κωδικό προϊόντος **TH58NVG4D4CTG00**.

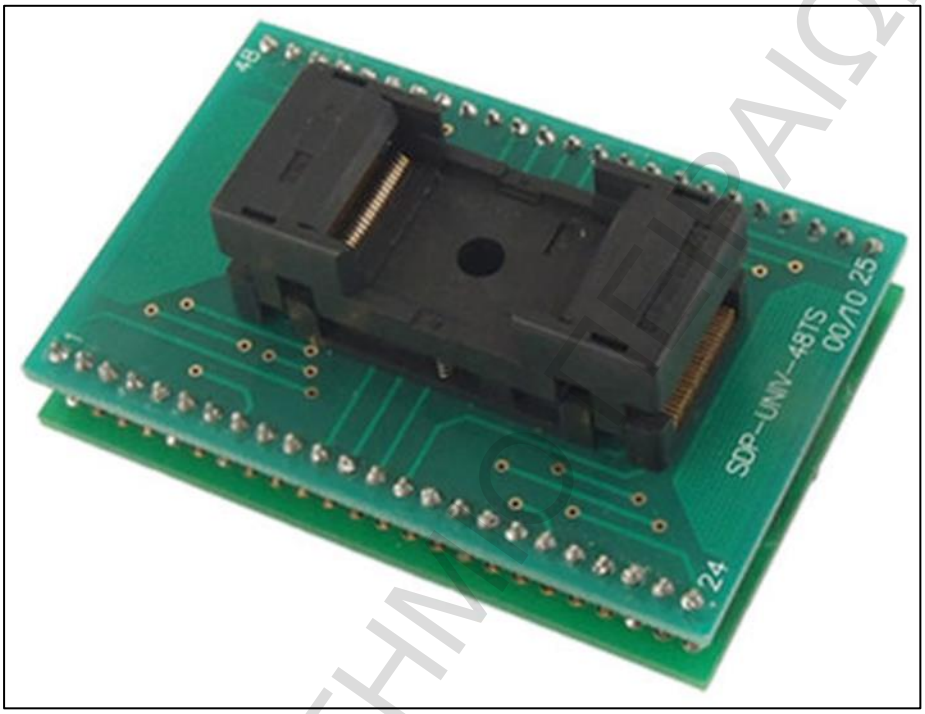
4.2 Υλικό (Hardware)

Για την επιλογή των υλικών θεσπίστηκαν κάποια κριτήρια τα οποία θα μας επέτρεπαν την ανάγνωση της εικόνας μνήμης μέσω μιας εύκολης και γρήγορης διαδικασίας.

- USB θύρα επικοινωνίας. Οι παράλληλες θύρες έχουν σχεδόν εκλείψει από την αγορά ενώ όλοι οι σύγχρονοι υπολογιστές είναι εφοδιασμένοι με θύρα USB, η οποία όπως φαίνεται θα υποστηρίζεται και στο προσεχές μέλλον.
- Ταχύτητα. Θέλαμε όσο το δυνατόν υψηλότερη ταχύτητα μεταφοράς δεδομένων, αν σκεφτούμε ότι οι σημερινές χωρητικότητες στις μνήμες NAND Flash είναι της τάξης των 2GB και άνω.
- Εύκολη διασυνδεσιμότητα. Η αποκόλληση και η επικόλληση ενός ολοκληρωμένου από την πλακέτα, εγκυμονεί κινδύνους καταστροφής της μνήμης. Για το σκοπό αυτό θέλαμε μια βάση που θα μας επέτρεπε την εύκολη προσάρτηση της μνήμης πάνω σε αυτή.
- Κόστος. Στόχος ήταν η δημιουργία μιας κατασκευής που θα μας επέτρεπε την ανάγνωση των δεδομένων για πειραματικούς σκοπούς, με κόστος κάτω των 100€.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Ο στόχος της εύκολης διασυνδεσιμότητας, ήταν ο πιο εύκολος να εκπληρωθεί. Ψάχνοντας στο eBay μπορέσαμε γρήγορα να εντοπίσουμε μια TSOP48 ZIF βάση, κατάλληλη για να μπορέσουμε να προσαρμόσουμε το ολοκληρωμένο της μνήμης πάνω σε αυτή. Το κόστος της περίπου 30€ και μια πληθώρα πωλητών να την προσφέρουν προς αγορά. Επιπλέον, η συγκεκριμένη βάση υποστήριζε και ολοκληρωμένα με μικρότερο αριθμό ακροδεκτών για διαφορετική χρήση.

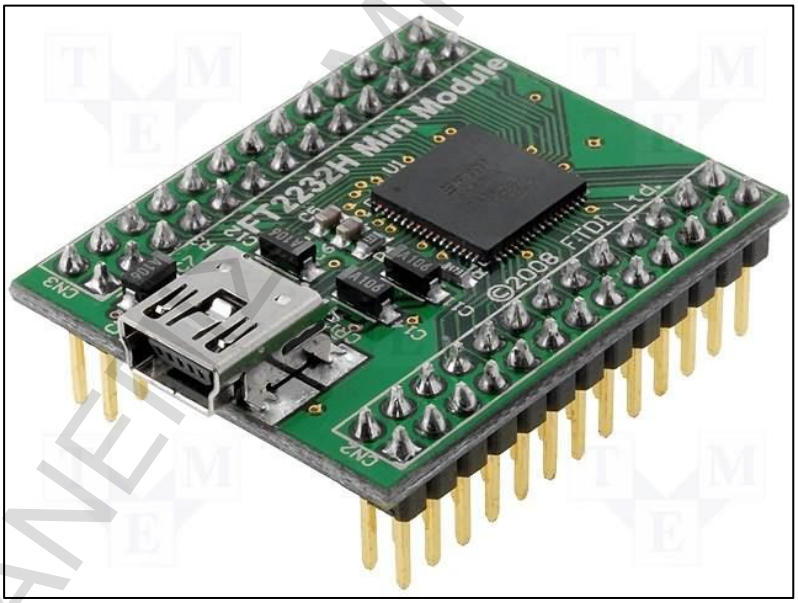


Εικόνα 4.1: Βάση Μνήμης TSOP-48 ZIF

Η επίτευξη του στόχου της USB συνδεσιμότητας και της γρήγορης ταχύτητας, - δηλαδή κάποιου μικροελεγκτή που θα υποστήριζε USB 2.0 επικοινωνία, διατηρώντας ταυτόχρονα το κόστος χαμηλά - μας οδήγησε στην επιλογή του ολοκληρωμένου FT2232H από την εταιρία FTDI. Το συγκεκριμένο ολοκληρωμένο περιγράφεται ως dual-port USB2-to-serial converter και εξομοιώνει την επικοινωνία μέσω μιας θύρας USB, για την παραγωγή σημάτων FIFO, JTAG, I2C. Επίσης, υποστηρίζει έναν καινούργιο τρόπο λειτουργίας, τον επονομαζόμενο “host emulation mode”. Ο συγκεκριμένος τρόπος λειτουργίας παρουσιάζει ιδιαίτερο ενδιαφέρον διότι, επιτρέπει την λειτουργία των ακροδεκτών του ολοκληρωμένου ως δίαυλοι επικοινωνίας, για την πολυπλεξία σημάτων δεδομένων και διευθυνσιοδότησης, με την δυνατότητα ανάγνωσης και εγγραφής. Δημιουργώντας τις κατάλληλες διασυνδέσεις, θα μπορούσαμε να δημιουργήσουμε μια διεπαφή για την επικοινωνία με ολοκληρωμένα μνήμης flash και με ταχύτητες της τάξης των 480Mbit/s που υποστηρίζει η θύρα USB2.0.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

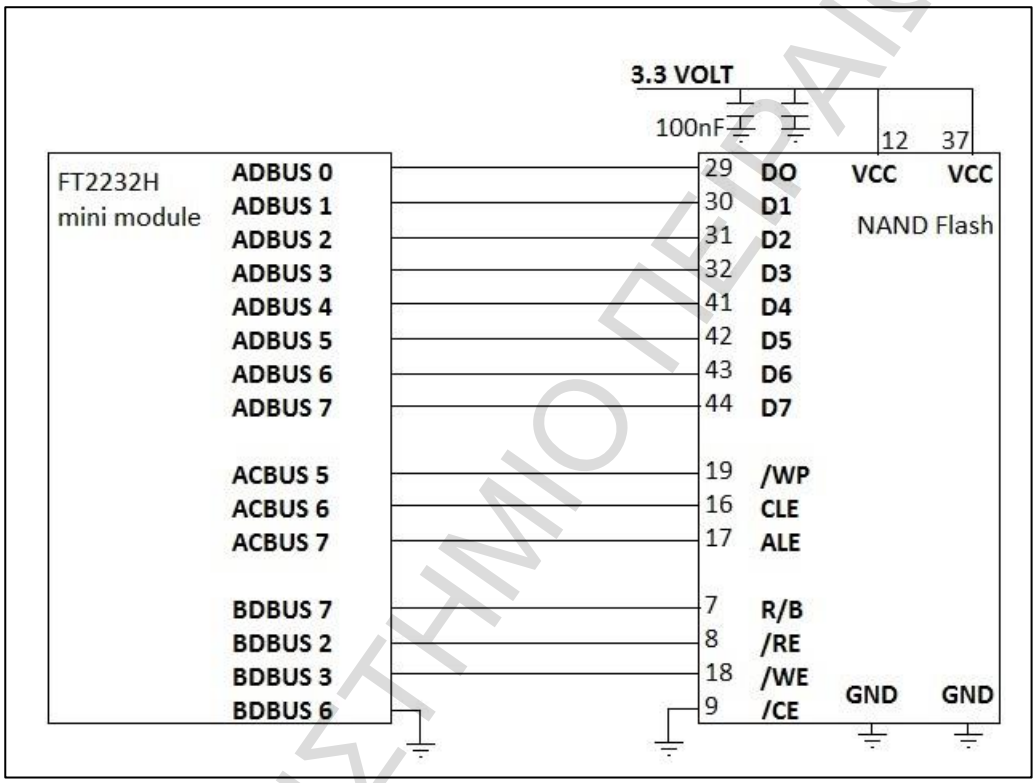
Για την λειτουργία του ολοκληρωμένου FT2232H απαιτούνται ορισμένα επιπρόσθετα εξαρτήματα και επιπλέον πωλείται σε βάση αρχιτεκτονικής TQFP, η οποία δεν θα μας επέτρεπε την χρησιμοποίηση του σε ένα breadboard. Παρόλα αυτά η εταιρία FTDI παράγει για τον σκοπό αυτό ένα mini module με το ολοκληρωμένο FT2232H και τα απαραίτητα εξαρτήματα για την λειτουργία του συνδεδεμένα. Επίσης, παρέχονται δυο σειρές θηλυκών ακροδεκτών για την εύκολη διασύνδεση του. Το συγκεκριμένο module μπορεί να προμηθευτεί από την ίδια την εταιρία αλλά κι από άλλους μεταπωλητές. Κάνοντας μια έρευνα στο eBay, μπορέσαμε να αποκτήσουμε ένα, με κόστος που κυμαινόταν περίπου στα 50€. Στο datasheet του ολοκληρωμένου και στην σελίδα 53 παρουσιάζονται οι απαραίτητες συνδέσεις. Το ολοκληρωμένο υποστηρίζει τάσεις λειτουργίας στα 5, 3.3 και 1.8 volt. Η δύο πρώτες μπορούν να επιτευχθούν και από την τροφοδοσία που παρέχεται μέσω της θύρας USB. Απευθείας τα 5volt ή με ενσωματωμένο διαιρέτη τάσης πάνω στο mini module, χρησιμοποιώντας κατάλληλες γέφυρες, για την παραγωγή 3.3volt, με την προϋπόθεση ότι το ρεύμα που καταναλώνεται δεν ξεπερνάει τα 500mA. Διαφορετικά, μπορεί να χρησιμοποιηθεί εξωτερικό τροφοδοτικό τάσης, για την λειτουργία της μνήμης NAND Flash σύμφωνα με την προδιαγραφή της.



Εικόνα 4.2: FT2232H mini module

Οι συνδέσεις που έπρεπε να σχεδιαστούν ανάμεσα στο mini module FT2232H και στην βάση ZIF ήταν σχετικά απλές. Οι γραμμές δεδομένων (data-lines) του mini module συνδέθηκαν απευθείας με τις γραμμές δεδομένων της NAND ZIF βάσης. Οι ακροδέκτες ανάγνωσης κι εγγραφής, συνδέθηκαν επίσης απευθείας με τις αντίστοιχες της βάσης που αντιστοιχούσαν στην μνήμη NAND Flash. Οι γραμμές που αντιστοιχούσαν στους ακροδέκτες

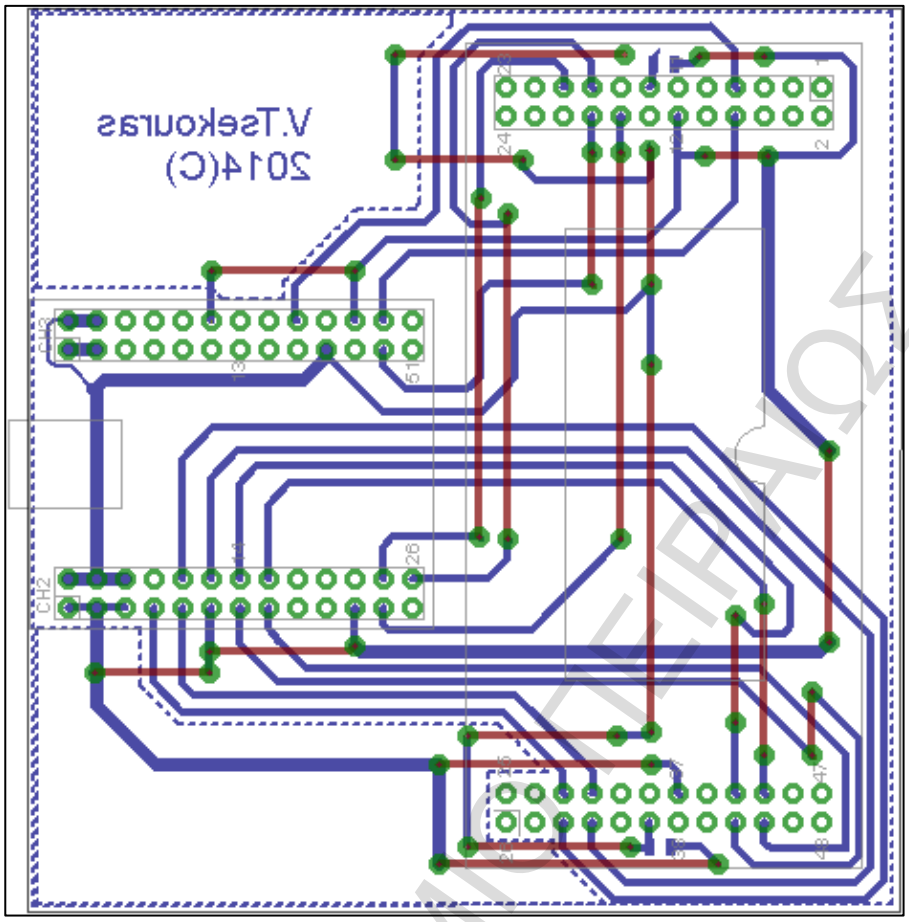
ALE, CLE και WP συνδέθηκαν σε γραμμές του mini module που αντιστοιχούσαν σε κατάσταση HIGH. Με αυτό τον τρόπο, για να γράψουμε ή να διαβάσουμε μια συγκεκριμένη διεύθυνση, δεν είχαμε παρά να θέσουμε τις τιμές στις συγκεκριμένες γραμμές. Τέλος, η γραμμή R/B, στην οποία η μνήμη NAND Flash υποδηλώνει την κατάστασή της, συνδέθηκε σε μια ελεύθερη διαθέσιμη I/O γραμμή του mini module. Φυσικά, δεν παρέλειψαν να γίνουν οι απαραίτητες συνδέσει για την τάση τροφοδοσίας της μνήμης καθώς και των απαραίτητων γειώσεων.



Εικόνα 4.3: Σχεδιάγραμμα Συνδέσεων FT2232H με Μνήμη NAND Flash

Τέλος, απαραίτητη ήταν και η σχεδίαση μια πλακέτας PCB για την επίτευξη των φυσικών συνδέσεων των εξαρτημάτων μεταξύ τους. Για τον σκοπό αυτό χρησιμοποιήθηκε το πρόγραμμα EAGLE PCB, στο οποίο σχεδιάστηκε η αρχιτεκτονική διάταξη των εξαρτημάτων, βάση των προδιαγραφών που καθορίζονταν από τα datasheet, ώστε να σχεδιαστούν οι απαραίτητοι διάδρομοι σε τυπωμένο κύκλωμα. Στην εικόνα 4.4 φαίνεται το τυπωμένο κύκλωμα.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ



Εικόνα 4.4: Πλακέτα Ηλεκτρικών Συνδέσεων

4.3 Λογισμικό (Software)

Επιπρόσθετα από το υλικό, για την ανάγνωση της μνήμης NAND Flash χρειαζόταν και κατάλληλο λογισμικό το οποίο θα μας επέτρεπε τον έλεγχο των γραμμών του mini module FT2232H για την επικοινωνία του με την NAND Flash. Για το σκοπό αυτό χρησιμοποιήσαμε το πρόγραμμα ftdinandreader - ανοιχτού κώδικα - το οποίο είναι γραμμένο από τον χρήστη SpritesMods σε γλώσσα προγραμματισμού C++ και βασίζεται επίσης στις ανοιχτού κώδικα βιβλιοθήκες libFTDI για περιβάλλον Linux. Το συγκεκριμένο πρόγραμμα θέτει το mini module σε host-bus-emulation τρόπο λειτουργίας, προσπαθεί αυτόματα να αναγνωρίσει την μνήμη NAND Flash και να συλλέξει πληροφορίες για το ολοκληρωμένο. Στην συνέχεια χρησιμοποιεί αυτές τις πληροφορίες για την ρύθμιση των παραμέτρων και του αλγόριθμου που χρειάζεται για την ανάγνωση και εξαγωγή των δεδομένων από την μνήμη. Για την παραμετροποίηση των ολοκληρωμένων μνήμης βασίζεται σε μια καταγραφή που έχει πραγματοποιηθεί από τον χρήστη Thomas Gleixner, μέλος ομάδας για την υποστήριξη του πυρήνα Linux και υπεύθυνης για την ανάπτυξη των βιβλιοθηκών libusb. Με αυτή την

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

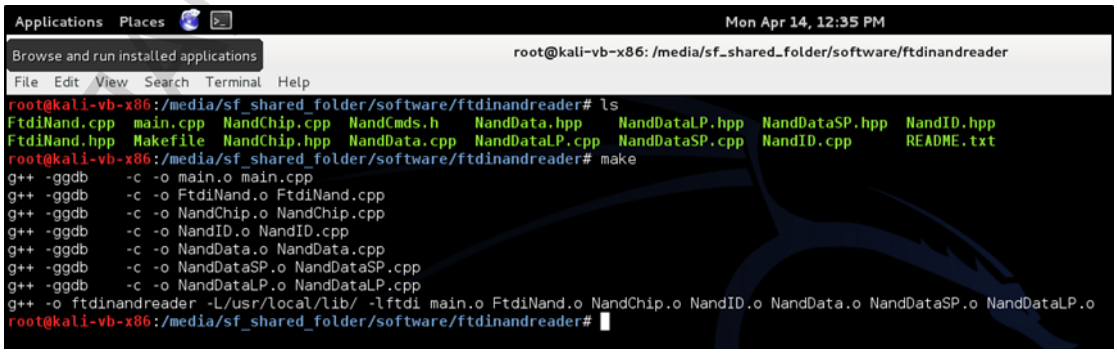
μέθοδο, παρέχεται η δυνατότητα ανάγνωσης τόσο των κύριων δεδομένων της μνήμης (data area) όσο και των βοηθητικών (spare area) που υπάρχουν ανά σελίδα.

Θεωρητικά το πρόγραμμα υποστηρίζει ταχύτητες ανάγνωσης της τάξης των 250Kbytes/sec, οι οποίες είναι αρκετά χαμηλότερες από τον μέγιστο ρυθμό ανάγνωσης που υποστηρίζει το mini module. Παρόλα αυτά, λόγω των χρόνων καθυστέρησης (latency) που παρουσιάζεται στο ολοκληρωμένο της μνήμης και στο πρωτόκολλο επικοινωνίας του USB κάθε φορά που πραγματοποιείται εναλλαγή στην ανάγνωση/εγγραφή των δεδομένων, οι ταχύτητες που επιτεύχθηκαν είναι αρκετά μικρότερες. Η ανάγνωση των δεδομένων ανά σελίδα προκαλεί αυξημένους χρόνους καθυστέρησης. Παρόλα αυτά το πρόγραμμα θα μπορούσε να γίνει ταχύτερο στέλνοντας εντολή ανάγνωσης για πολλαπλές σελίδες κάθε φορά.

Επίσης, ενδιαφέρον θέμα θα αποτελούσε και η δυνατότητα εγγραφής δεδομένων στην μνήμη NAND Flash. Για την εγγραφή όμως μιας σελίδας μνήμης απαιτείται και η εγγραφή των δεδομένων στην περιοχή “spare area”. Δηλαδή, των τιμών του αλγόριθμου ECC, του αλγόριθμου wear-leveling και οποιοδήποτε άλλων πληροφοριών απαιτούνται. Επειδή όμως η μέθοδος αποθήκευσης των meta-data δεν έχει προτυποποιηθεί, διαφέρει από συσκευή σε συσκευή, και δεν γίνεται να γενικευθεί με ένα πρόγραμμα.

Το πρόγραμμα ftdinandreader παρέχεται κάτω από την άδεια GPLv3. Μέσω του διαδικτυακού ιστότοπου GitHub παρέχεται ο πηγαίος κώδικας που είναι γραμμένος σε C++. Για την εκτέλεση του προγράμματος σε λειτουργικό σύστημα Linux, απαιτείται η μεταγλώττισή του. Για την διαδικασία παρέχεται κατάλληλο αρχείο makefile. Απαραίτητη είναι και η ύπαρξη στη διανομή Linux, του μεταγλωττιστή g++. Για την δημιουργία του εκτελέσιμου αρχείου, από ένα τερματικό εκτελούμε από τον φάκελο αρχείων:

```
root@kalivmx86:~# make
```



Εικόνα 4.5: Μεταγλώττιση Προγράμματος ftdinandreader

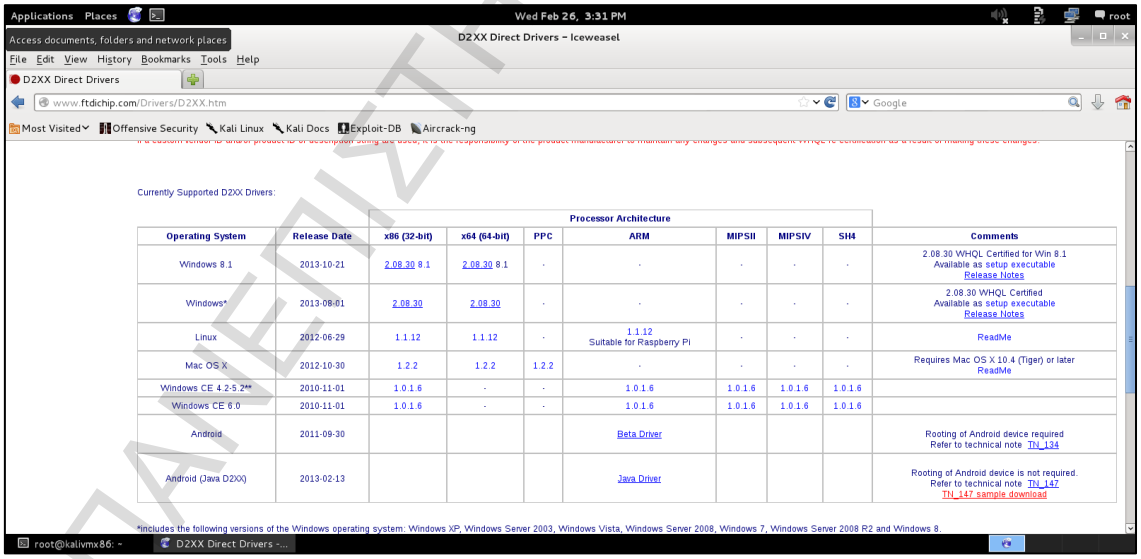
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

4.4 Επιβεβαίωση Λειτουργίας

Έχοντας υλοποιήσει την κατασκευή μας, είχαμε στα χέρια μας ένα NAND-reader που μας επέτρεπε την ανάγνωση δεδομένων από μνήμες NAND Flash 8bit στα 3.3Volt, με κόστος κατασκευής περίπου 100€. Για την επαλήθευση της λειτουργίας χρησιμοποιήσαμε ηλεκτρονικό υπολογιστή με Host λειτουργικό σύστημα Windows 7 64bit πάνω στο οποίο εγκαταστήσαμε την ανοιχτού κώδικα πλατφόρμα Virtual Box 4.3.8 της Oracle και προχωρήσαμε στην εγκατάσταση της διανομής Kali Linux 1.0.6-i386. Επίσης, προχωρήσαμε στην ενημέρωση του λειτουργικού συστήματος Kali Linux εκτελώντας της εντολές:

```
root@kalivmx86:~# apt-get update
root@kalivmx86:~# apt-get install
root@kalivmx86:~# apt-get upgrade
root@kalivmx86:~# apt-get dist-upgrade
```

Πριν την εκτέλεση του προγράμματος ftdinandreader απαραίτητη ήταν και η εγκατάσταση των βιβλιοθηκών libftdi καθώς και της τελευταίας έκδοσης των οδηγών libftd2xx 1.1.12 για την αναγνώριση του mini module FT2232H από το λειτουργικό σύστημα Kali Linux. Η μεταφόρτωση των οδηγών έγινε από το site της εταιρίας FTDI.



Εικόνα 4.6: Ιστότοπος Μεταφόρτωσης Οδηγών FTDI

Από την κονσόλα του Kali Linux εκτελέσαμε:

- Για την εγκατάσταση των βιβλιοθηκών:

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

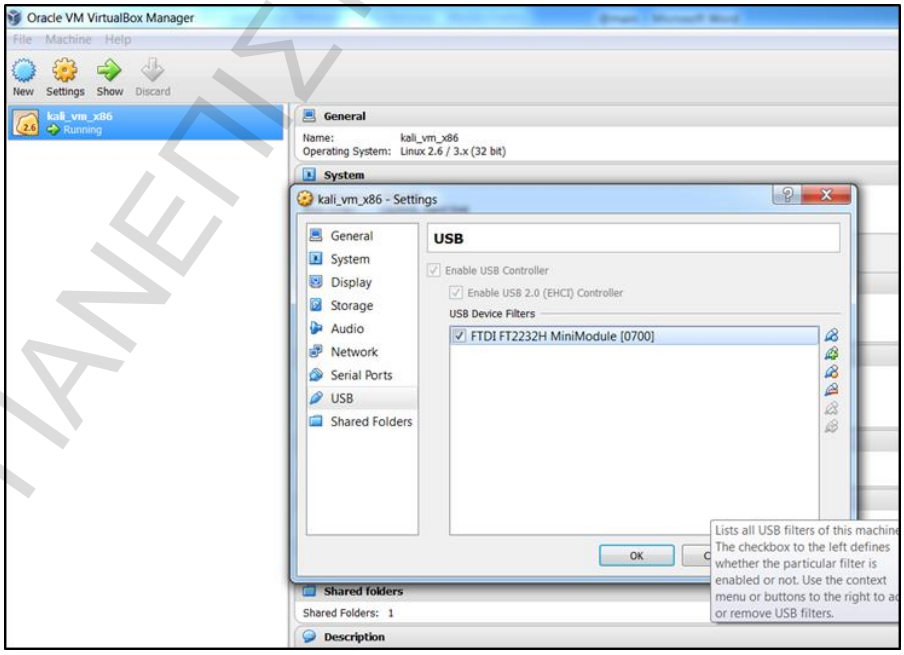
```
root@kalivmx86:~# apt-cache search ftdi
root@kalivmx86:~# apt-get install libftdi-dev
```

- Για την εγκατάσταση των οδηγών:

```
root@kalivmx86:~# gunzip libftd2xx1.1.12.tar.gz
root@kalivmx86:~# tar –xvf libftd2xx1.1.12.tar
root@kalivmx86:~# cd Desktop/release/build/i386
root@kalivmx86:~# cp lib* /usr/local/lib
root@kalivmx86:~#  cd /usr/local/lib
root@kalivmx86:~# ln –s libftd2xx.so.1.1.12 libftd2xx.so
root@kalivmx86:~# chmod 0755 libftd2xx.so.1.1.12
```

Περισσότερες πληροφορίες για την εγκατάσταση των οδηγών του mini module FT2232H μπορούν να βρεθούν και στο αρχείο “AN_220 FTDI Drivers Installation Guide for Linux.pdf” της FTDI.

Επίσης, πριν την εκίνηση του συστήματος Kali Linux χρειάστηκε να δηλώσουμε στην θύρα USB του VirtualBox, την ύπαρξη της συσκευής mini module FT2232H, για την αναγνώριση του από το λειτουργικό σύστημα.



Εικόνα 4.7: Ενεργοποίηση θύρας USB στην Εικονική Μηχανή

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Για την εκτέλεση του προγράμματος ftdinandreader πληκτρολογούμε από το τερματικό τα εξής:

```
root@kalivmx86:~# ./ftdinandreader –t both –r Toshiba_both
```

όπου -r το όνομα του αρχείου που θα αποθηκευτεί η εικόνα της μνήμης.

Παρατήρηση: το πρόγραμμα δέχεται την παράμετρο –t με τις εξής επιλογές:

- -t both

  "ανάγνωση x bytes ανά σελίδα, όπου x το άθροισμα των data area και oob area"
- -t main

  "ανάγνωση x bytes ανά σελίδα, όπου x το μέγεθος των data area"
- -t oob

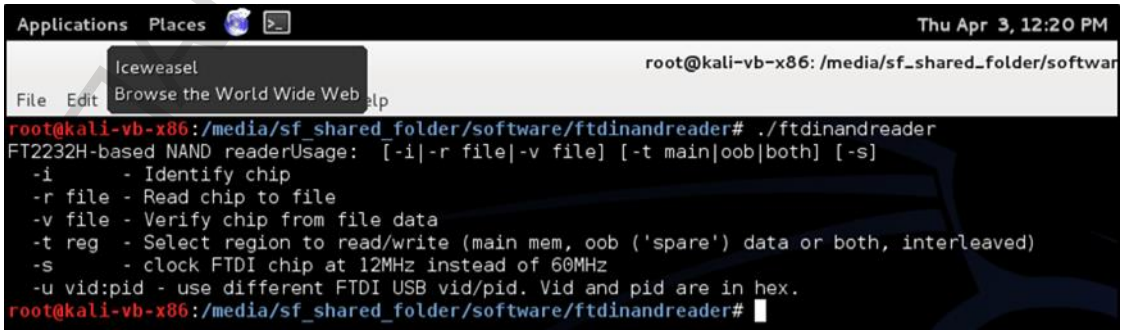
  "ανάγνωση x bytes ανά σελίδα, όπου x το μέγεθος των oob area"

Ορισμένες μνήμες NAND Flash δεν αποθηκεύουν τα meta-data στο τέλος της σελίδας, αλλά ανά 512bytes χρησιμοποιούν το ¼ αυτών. Στην περίπτωση που μελετήσαμε υπήρχαν 16 bytes meta-data ανά 512 bytes main-data. Θα μπορούσαμε λοιπόν να πούμε ότι τα δεδομένα ανά σελίδα διαμορφώνονται ως εξής:

Page = 512 + 16 + 512 + 16 + 512 + 16 + 512 + 16 = 2112 bytes

Page = main + oob + main + oob + main + oob + main + oob

Η ανάγνωση των δεδομένων με τις παραμέτρους <-t main> και <-t oob> εγκυμονεί κινδύνους και συνίσταται η πλήρης ανάγνωση των δεδομένων <-t both>, της σελίδας για να εξακριβωθεί η διάταξη. Στην επόμενη εικόνα παρουσιάζεται μια πλήρη περιγραφή της επιλογής των παραμέτρων.



Εικόνα 4.8: Παράμετροι Προγράμματος ftdinandreader

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Με το πέρας της εργασίας είχαμε στην διάθεσή μας ένα αρχείο, το οποίο ονομάσαμε “Toshiba_both”, μεγέθους 2GB (2.147.483.648 bytes) προς ανάλυση. Η ανάγνωση των δεδομένων από την μνήμη NAND Flash είναι μόνο η μισή εργασία που απαιτείται. Μετά την απόκτηση της εικόνας μνήμης, απαιτούνται ειδικά εργαλεία για την μελέτη και την αναδιάταξη των δεδομένων. Όπως έχουμε αναφέρει τα δεδομένα αποθηκεύονται στη μνήμη από τον controller και το λογισμικό που εκτελείται σε αυτόν, εκτελώντας διεργασίες όπως διαχείριση εσφαλμένων block, wear-leveling κι άλλες. Αποτέλεσμα, τα δεδομένα που προέρχονται από το Host μηχάνημα και αποθηκεύονται στο USB stick, να μην είναι γραμμικά αποθηκευμένα ανά σελίδα αλλά με μια δυναμική άγνωστη μέθοδο.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Κεφάλαιο 5ο

Ανάλυση Εικόνας Μνήμης

5.1 Στατική Ανάλυση

Η φάση της στατικής ανάλυσης δεν υλοποιείται με αυτόματο τρόπο διότι, αποτελεί χειρωνακτική διαδικασία η οποία απαιτεί την χρήση διάφορων εργαλείων σε αλληλεπίδραση με τον αναλυτή. Στην περίπτωση ενός άγνωστου flash συστήματος αρχείου (flash file system), τεχνικές reverse engineering πρέπει να αναπτυχθούν για την ανάλυση του. Μια πανομοιότυπη USB stick συσκευή θα μπορούσε να χρησιμοποιηθεί και να ερευνηθεί, ώστε η όλη διαδικασία να γίνει πιο εύκολη.

Στην συνέχεια αναφέρουμε και περιγράφουμε ορισμένα εργαλεία που χρησιμοποιήσαμε σε περιβάλλον Linux και Windows. Κάποια που αναπτύξαμε με γλώσσα προγραμματισμού Python 2.7.6 καθώς και την μέθοδο που ακολουθήσαμε στην προσπάθεια που κάναμε για την ανάλυση της εικόνας μνήμης.

5.2 Flash File System

Έχοντας αποκτήσει την εικόνα μνήμης από την NAND Flash, πρώτο βήμα ήταν η διερεύνηση για τον αν το σύστημα αρχείων εμπίπτει σε κάποιο από τα γνωστά και ευρέως διαδεδομένα συστήματα αρχείων που χρησιμοποιούνται για την εγγραφή δεδομένων σε ολοκληρωμένα NAND Flash. Για το σκοπό αυτό εξετάσαμε την εικόνα μνήμης στο περιβάλλον Kali Linux που είχαμε εγκαταστήσει χρησιμοποιώντας το γραφικό περιβάλλον (autopsy) του εργαλείου The Sleuth Kit. Το συγκεκριμένο εργαλείο αναγνωρίζει μια πληθώρα συστημάτων αρχείων συμπεριλαμβανομένων αυτών που χρησιμοποιούνται σε NAND Flash ολοκληρωμένα, όπως τα γνωστά, JFFS2 (Journaling Flash File System, Version 2) και το YAFFS (Yet Another Flash File System). Στον πίνακα 5.1 φαίνονται ορισμένα από τα συστήματα αρχείων που χρησιμοποιούν οι εταιρίες μέχρι στιγμής.

Product Name	Company/Sponsor
FlPack Angel & Jet	TOKYO ELECTRON DEVICE
FlashFX	DataLight
JFFS2	Red Hat
NAND File System	Kyoto Software Research

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

smxFFS	Micro Digital
TargetFFS-NAND	Blunk Microsystems
TrueFFS	Wind River Systems
YAFFS	Toby Churchill

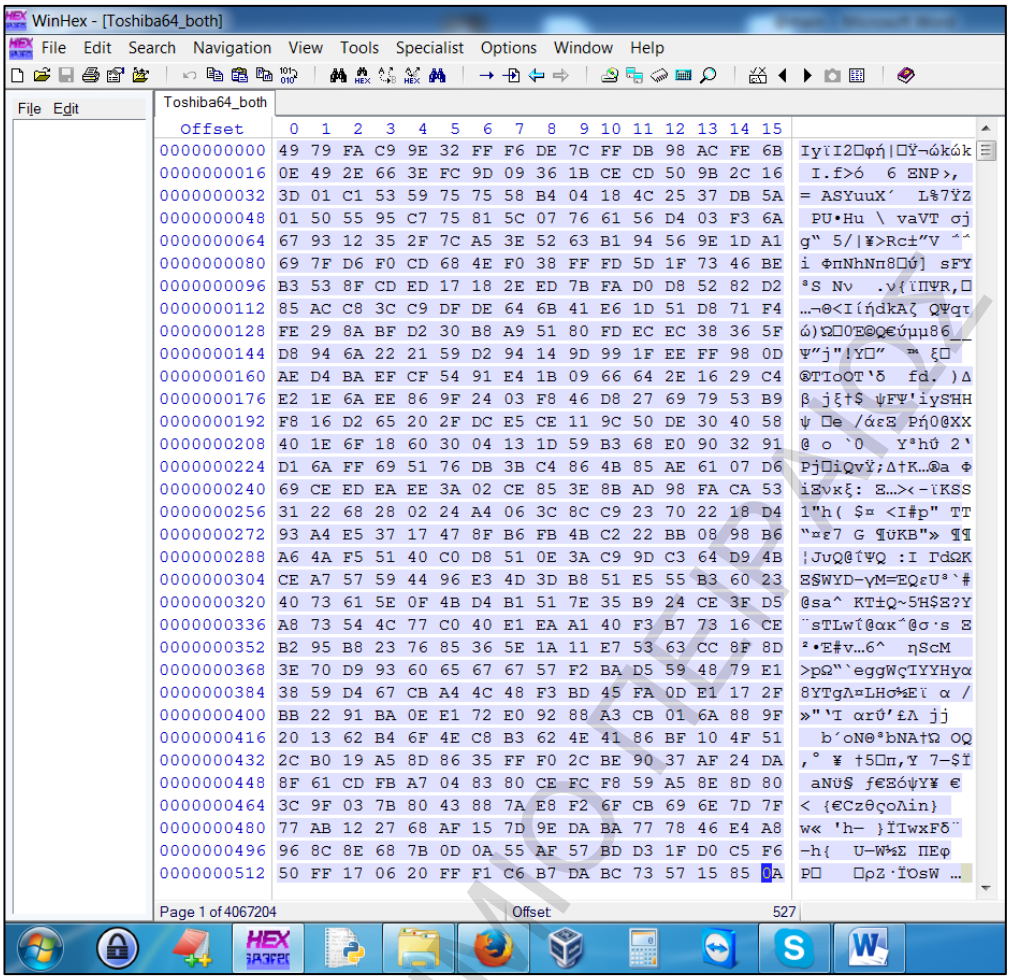
Πίνακας 5.1: Flash File Systems

Δυστυχώς η έρευνα απέδειξε ότι κανένα από τα παραπάνω δεν εμπίπτει στο σύστημα αρχείων που έχει χρησιμοποιηθεί από τον controller του USB stick για την εγγραφή των δεδομένων στην μνήμη NAND Flash. Για την εγγραφή/ανάγνωση των δεδομένων στη μνήμη ο controller της συσκευής χρησιμοποιεί δική του μεθοδολογία. Πρέπει λοιπόν, να αναλυθούν τα δυαδικά (binary) δεδομένα της εικόνας μνήμης και να προσπαθήσουμε να τα ανασυντάξουμε με τρόπο που θα μας οδηγήσουν στο ανώτερο σύστημα αρχείων που χρησιμοποιείται από το λειτουργικό σύστημα, και στην περίπτωση των USB stick είναι το σύστημα αρχείων FAT.

5.3 Εργαλεία σε Python

Για να μπορέσουμε να εξετάσουμε και να μελετήσουμε τα δυαδικά δεδομένα που ήταν αποθηκευμένα στην εικόνα μνήμης χρησιμοποιήσαμε το πρόγραμμα WinHex. Εξετάζοντας τα πρώτα 2112 bytes της εικόνας μνήμης παρατηρήσαμε ότι τα δεδομένα ήταν γραμμένα ανά σελίδα με συγκεκριμένη διάταξη. Κάθε σελίδα περιείχε 512 bytes (sector) από κύρια δεδομένα (main data) και ακολουθούσαν 16 bytes βοηθητικά δεδομένα (meta data). Σε όλα τα βοηθητικά δεδομένα, τα δυο πρώτα byte είχαν την χαρακτηριστική δεκαεξαδική τιμή 50FF και την χρησιμοποιήσαμε για την αναγνώρισή τους. Η συγκεκριμένη διάταξη επαναλαμβανόταν τέσσερις φορές για κάθε σελίδα, μέχρι την συμπλήρωση των 2112 bytes. Στην εικόνα 5.1 φαίνονται τα πρώτα 528 bytes μιας σελίδας όπως παρουσιάστηκαν στο πρόγραμμα WinHex.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ



Εικόνα 5.1: 528bytes σελίδας μνήμης

5.3.1 splitimage.py

Εξετάζοντας τη δομή των υπόλοιπων σελίδων παρατηρήσαμε ότι όλη η εικόνα μνήμης ήταν δομημένη με την παραπάνω μέθοδο. Χρειαζόμασταν λοιπόν ένα εργαλείο, σκοπός του οποίου ήταν η αφαίρεση των βοηθητικών δεδομένων από την εικόνα μνήμης. Το εργαλείο που δημιουργήσαμε το ονομάσαμε splitimage.py και αφαιρεί τα βοηθητικά δεδομένα (meta data) από κάθε σελίδα της εικόνας μνήμης.

Με το συγκεκριμένο εργαλείο μπορούσαμε να διαχωρίσουμε τα δεδομένα οποιασδήποτε εικόνας μνήμης σε δυο αρχεία. Το ένα αρχείο να περιέχει μόνο τα κύρια δεδομένα και το άλλο αρχείο μόνο τα βοηθητικά.

```
import os
f = file('c:/python27/tmp/newimage_full','rb')
newoobfile = file('c:/python27/tmp/newimage_full_oob','ab'
newmainfile = file('c:/python27/tmp/newimage_full_main','ab')
```

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

```
f.seek(0,os.SEEK_END)
size = f.tell()
print size
pagenumbers = size/2112
print pagenumbers
i = 0
main_bytelocation = 0
oob_bytelocation = 512
while i <= pagenumbers:
    f.seek(main_bytelocation)
    mainstring = f.read(512)
    main_bytelocation = main_bytelocation + 528
    newmainfile.write(mainstring)
    f.seek(oob_bytelocation)
    oobstring = f.read(16)
    oob_bytelocation = oob_bytelocation + 528
    newoobfile.write(oobstring)
f.close()
newmainfile.close()
newoobfile.close()
exit()
```

5.3.2 blocksorted.py

Στο επόμενο βήμα που υλοποιήσαμε, εξετάζοντας προσεκτικά τα δεδομένα που προέκυψαν από το αρχείο με τα βοηθητικά δεδομένα (meta data) προέκυψαν ορισμένες διαπιστώσεις.

Σε κάθε κομμάτι των βοηθητικών δεδομένων που αντιστοιχούσαν σε ένα τμήμα σελίδας του ίδιου block, υπήρχε γραμμένη η ίδια τιμή στην ίδια θέση του $3^{ου}$, $4^{ου}$ και $5^{ου}$ byte των βοηθητικών δεδομένων. Εξετάζοντας κι άλλα βοηθητικά δεδομένα στα επόμενα block της εικόνας μνήμης παρατηρήσαμε ότι περισσότερα από ένα block περιείχαν την συγκεκριμένη τιμή.

Προκειμένου να εξετάσουμε ακριβώς τι συμβαίνει αναγκαστήκαμε να δημιουργήσουμε ένα καινούργιο εργαλείο σκοπός του οποίου ήταν να τεμαχίσει την πλήρης εικόνα μνήμης σε

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

block δίνοντας σαν όνομα αρχείου για κάθε block την τιμή των τριών byte που περιεχόταν στα βοηθητικά δεδομένα του τμήματος των σελίδων, ενωμένη με την αύξουσα τιμή του κάθε block στην δεκαεξαδική μορφή της. Ταυτόχρονα δημιουργήσαμε κι ένα καινούργιο δυαδικό αρχείο το οποίο ονομάσαμε table_block (εικόνα 5.2) και το οποίο περιείχε όλες τις εγγραφές με τα ονόματα αρχείων που προέκυπταν και στην ουσία ήταν αρχειοθετημένο με την αύξουσα σειρά που προέκυπτε από τα block.

Στον πίνακα 5.2 φαίνεται ένα παράδειγμα 16 bytes βοηθητικών δεδομένων που αντιστοιχούν σε ένα τμήμα των 512 byte μιας σελίδας. Η έγχρωμη τιμή επαναλαμβάνεται σε όλα τα βοηθητικά δεδομένα που υπάρχουν μέσα στο ίδιο block, καθώς επίσης και σε άλλα block της εικόνας μνήμης.

Meta data																
byte location	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
data	50	FF	17	06	20	FF	F1	C6	B7	DA	BC	73	57	15	85	0A

Πίνακας 5.2: Βοηθητικά Δεδομένα Τμήματος Σελίδας

```
from struct import *
import os
f = file('Toshiba64_both','rb')
tb = file('c:/python27/tmp/table_block','w+b')
f.seek(0,os.SEEK_END)
size = f.tell()
blockNumbers = size/270336
print blockNumbers
blockOffset = 270336
counterOffset = 514

""" split image into blocks with fliename according to zoneID and blockID """
for i in range(0,blockNumbers):

   """ create ID's """
   f.seek(i*blockOffset + counterOffset, 0)
   zoneID = f.read(3)
   blockID = pack('>h',i)
```

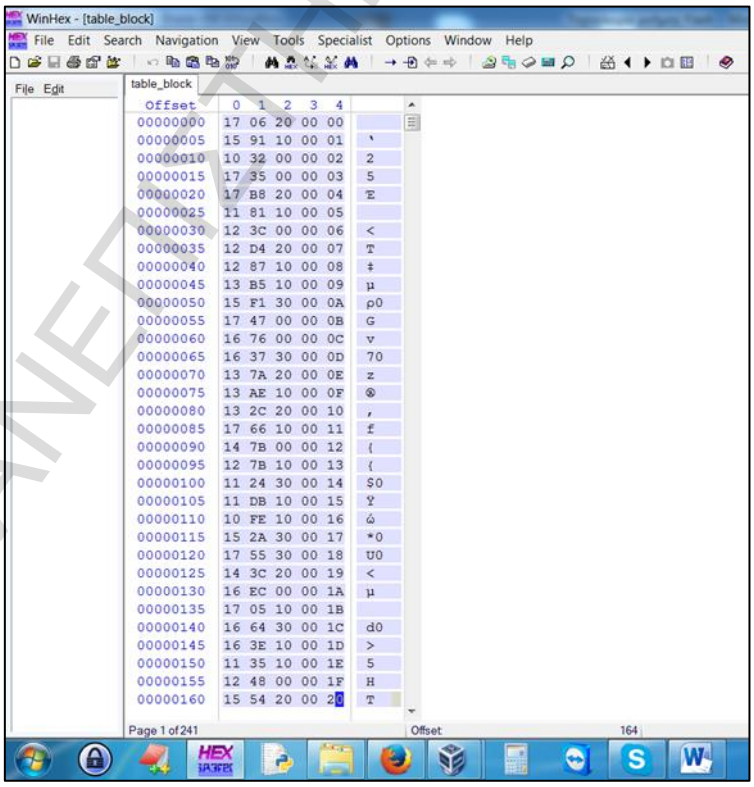
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

```
    ID = zoneID + blockID


    """ save blocks """
    f.seek(i*blockOffset, 0)
    block = f.read(blockOffset)
    blockname                                                              =
file('c:/Python27/tmp/%s%s' %(str(zoneID.encode('hex')),blockID.encode('hex')), 'wb')
    blockname.write(block)
    blockname.close()

    """ create a table with all ID's(5 bytes) sorted per block """
    tb.seek(i*5)
    tb.write(ID)

f.close()
tb.close()
exit()
```

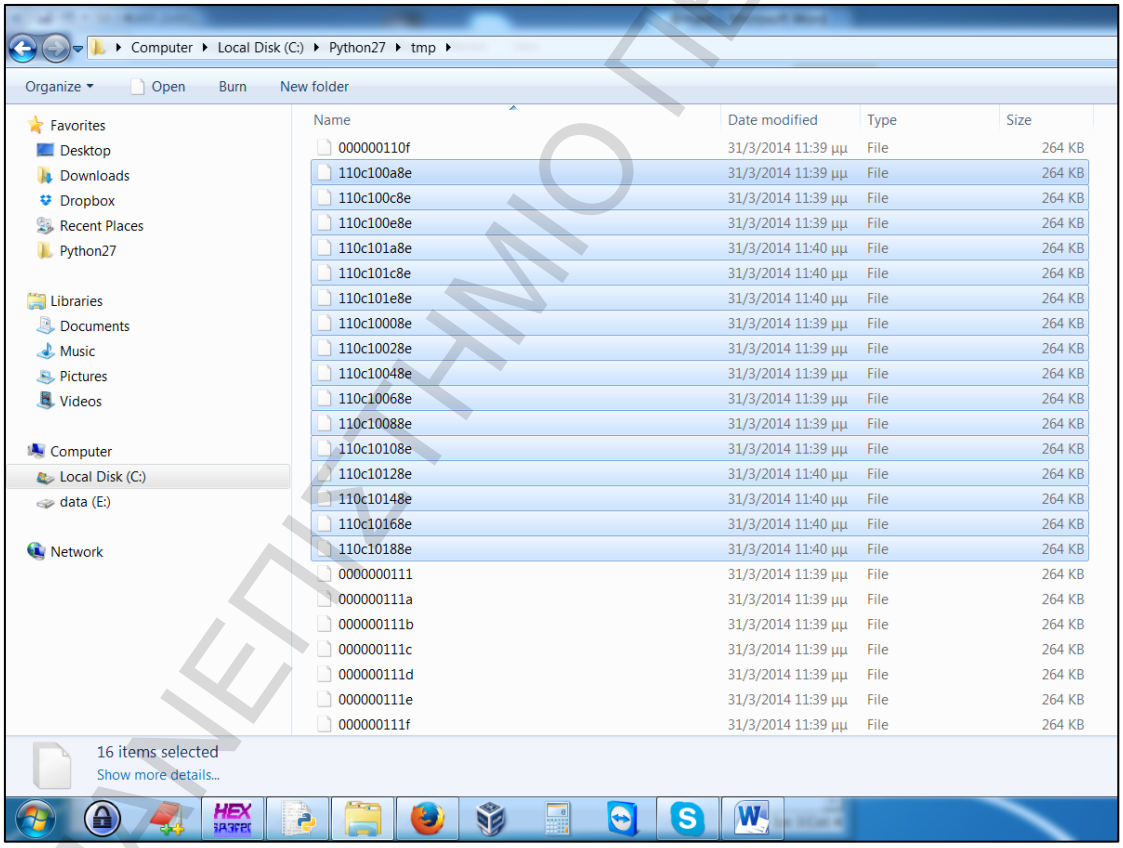


Εικόνα 5.2: Δεδομένα Αρχείου table_block

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

5.3.3 zoneblocksort.py

Με την διάσπαση της εικόνας μνήμης σε block και παρατηρώντας τα αρχεία που προέκυψαν, μας οδήγησαν στο συμπέρασμα ότι, ανά 16 block τα αρχεία είχαν την ίδια τιμή στο 3^{o},4^{o} και 5^{o} byte των βοηθητικών δεδομένων. Οπότε κάθε block αποτελούσε σύνολο μιας μεγαλύτερης ομάδας, από 16 block, την οποία την ονομάσαμε ζώνη (zone). Επίσης η ομάδα αρχείων με χαρακτηριστική τιμή 000000xxxx σαν όνομα αρχείου block, αποτελούσε εκείνα τα block τα οποία δεν είχαν γραφτεί μέχρι στιγμής και ήταν περισσότερα από 16, ενώ η ομάδα αρχείων με την χαρακτηριστική τιμή FFFFFFxxxx, αποτελούσε εκείνα τα block που είχαν σημαδευτεί ως ελαττωματικά (bad blocks) από την NAND Flash. Στην εικόνα 5.3 φαίνεται μια ομάδα των 16 block με το ίδιο χαρακτηριστικό ζώνης, ενώ τα δυο τελευταία byte αποτελούν τον αύξον αριθμό του block.



Εικόνα 5.3: Αρχεία block Μνήμης

Για να μπορέσουμε να ανασυντάξουμε τα block πρώτα με την αύξουσα σειρά της ζώνης στην οποία ανήκαν και εκ των υστέρων με την αύξουσα σειρά των block στην μνήμη NAND Flash, μεταχειριστήκαμε το όνομα του αρχείου που είχαμε δώσει σε κάθε block σαν ενιαίο δεκαδικό αριθμό, από τον πίνακα-αρχείο table_block που είχε προκύψει από το πρόγραμμα blocksorted.py. Ύστερα, δημιουργήσαμε ένα καινούργιο πίνακα-αρχείο ο

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

οποίος περιείχε όλες τις τιμές αρχειοθετημένες. Το καινούργιο πρόγραμμα το ονομάσαμε zoneblocksort.py

```
import os
tb = file('c:/python27/tmp/table_block','rb')
tzb = file('c:/python27/tmp/table_zone_block', 'w+b')
tb.seek(0,os.SEEK_END)
size = tb.tell()
numberofIDstb = size/5

""" create a table with all ID's(5 bytes) sorted per zone/block """
tb.seek(0)
first_ID = tb.read(5)
tb.seek(5,0)
second_ID = tb.read(5)

if int(first_ID.encode('hex'),16) < int(second_ID.encode('hex'),16):
   tzb.seek(0)
   tzb.write(first_ID + second_ID)
else:
   tzb.write(second_ID + first_ID)

for x in range(2,numberofIDstb):
   print x
   """ read each ID @tb """
   tb.seek((x*5),0)
   ID = tb.read(5)

   """ number of ID's @ tzb """
   tzb.seek(0,os.SEEK_END)
   size = tzb.tell()
   numberofIDstzb = size/5
   #print 'number of IDs @tzb', numberofIDstzb
```

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

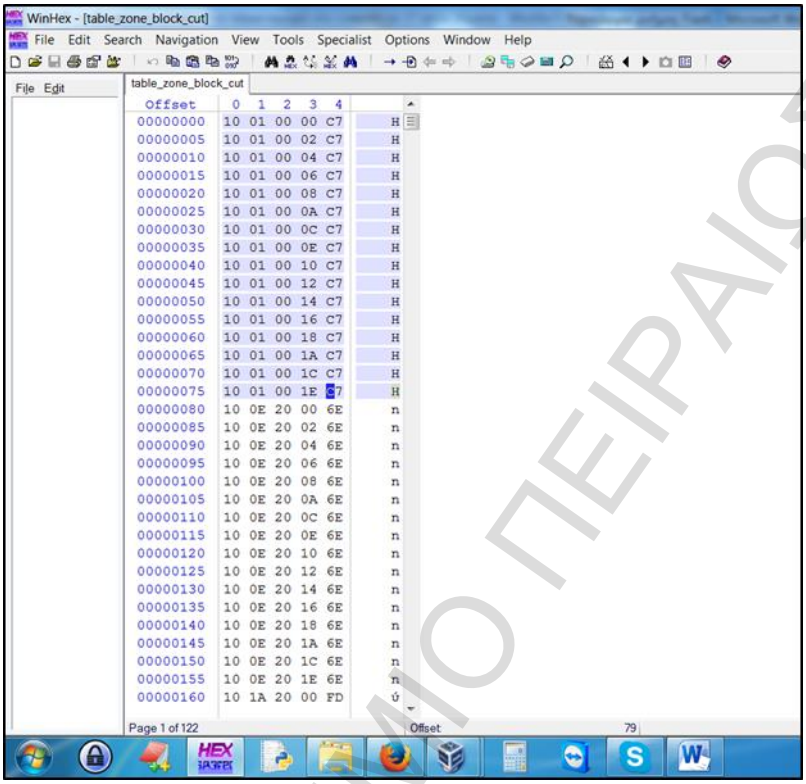
```
    """ read ID's until now @tzb """
    tzb.seek(0)
    old = tzb.read()

    """ loop throught @tzb pairs """
    for y in range(0,numberofIDstzb - 1):
        tzb.seek(0)
        tzb.seek(y*5)
        tzb_first_ID = tzb.read(5)
        tzb_next_ID = tzb.read(5)
        #print 'ID = ', int(ID.encode('hex'),16)
        #print 'tzb_first_ID = ', int(tzb_first_ID.encode('hex'),16)
        #print 'tzb_next_ID = ', int(tzb_next_ID.encode('hex'),16)
        if int(ID.encode('hex'),16) < int(tzb_first_ID.encode('hex'),16) and y == 0:
            tzb.seek(0)
            tzb.write(ID + old)
        elif (int(ID.encode('hex'),16) > int(tzb_first_ID.encode('hex'),16)) and
(int(ID.encode('hex'),16) < int(tzb_next_ID.encode('hex'),16)):
            tzb.seek(0)
            tzb.write(old[:(y*5)] + ID + old[(y*5):])
        elif int(ID.encode('hex'),16) > int(tzb_next_ID.encode('hex'),16) and (y ==
numberofIDstzb - 2) :
            tzb.seek(0)
            tzb.write(old + ID)
        #print '--------------------------------------'
tb.close()
tzb.close()
exit()
```

Με την εκτέλεση του προγράμματος το καινούργιο αρχείο που δημιουργήθηκε (table_zone_block), περιείχε καταγραμμένα όλα τα block και αρχειοθετημένα κατά αύξουσα σειρά ζώνης και block. Επειδή εκείνα τα block που ήταν κενά δεν μας ενδιέφεραν, καθώς επίσης και εκείνα τα block που είχαν χαρακτηριστεί ελαττωματικά, για το λόγο αυτό επεξεργαστήκαμε το αρχείο table_zone_block με το πρόγραμμα WinHex αφαιρώντας από

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

την αρχή την ομάδα με ονόματα αρχείου 000000xxxx και από το τέλος την ομάδα με ονόματα αρχείου FFFFFFxxxx. Το καινούργιο πίνακα-αρχείο τον ονομάσαμε table_zone_block_cut.



Εικόνα 5.4: Δεδομένα Αρχείου table_zone_block_cut

5.3.4 createimage.py

Το επόμενο βήμα ήταν να δοκιμάσουμε να ξαναδημιουργήσουμε την εικόνα μνήμης, ώστε να μπορέσουμε να την εξετάσουμε με το γνωστό εργαλείο ανάλυσης δίσκων, TestDisk. Για το σκοπό αυτό δημιουργήσαμε ένα καινούργιο εργαλείο το οποίο διάβαζε από το αρχείο-πίνακα table_zone_block_cut που είχαμε δημιουργήσει, τα ονόματα των αρχείων που είχαν τα block και δημιουργούσε την νέα εικόνα μνήμης. Το καινούργιο πρόγραμμα για την δημιουργία της νέα εικόνας μνήμης το ονομάσαμε createimage.py.

```
import os
tzb = file('c:/python27/tmp/table_zone_block_cut', 'rb')
newimage = file('c:/python27/tmp/newimage_cut', 'ab')
tzb.seek(0,os.SEEK_END)
size = tzb.tell()
numIDs = size/5
```

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

```
print numIDs


for i in range(0,numIDs):
   """read file ID from tzb """
   tzb.seek(i*5, 0)
   fileID = tzb.read(5)
   print fileID.encode('hex')


   """read block """
   block = file('c:/python27/tmp/%s' %str(fileID.encode('hex')), 'rb')
   block.seek(0)
   blockdata = block.read()
   block.close()


   """ append blockdata to newimage """
   newimage.write(blockdata)


tzb.close()
newimage.close()
exit()
```

5.3.5 TestDisk

Η καινούργια εικόνα μνήμης που δημιουργήθηκε μετά την εκτέλεση του προγράμματος είχε μέγεθος περίπου 1GB και περιείχε μόνο εκείνα τα block που ήταν αρχειοθετημένα με αύξουσα σειρά ζώνης και αύξουσα σειρά φυσικής διεύθυνσης block στη μνήμη NAND Flash, χωρίς το κενό χώρο (κενά block) καθώς επίσης και χωρίς τα ελαττωματικά block (bad block). Στην συνέχεια εκτελέσαμε το αρχικό πρόγραμμα που είχαμε δημιουργήσει (splitimage.py) στην νέα εικόνα μνήμης για να αφαιρέσουμε τα βοηθητικά δεδομένα (meta data). Το νέο αρχείο εικόνας μνήμης με τα κύρια δεδομένα (main data) ονομάστηκε newimage_cut_main και προχωρήσαμε στην ανάλυση του με την εφαρμογή TestDisk σε περιβάλλον Linux.

Από το τερματικό της εικονικής μηχανής Kali Linux εκτελέσαμε:

```
root@kalivmx86:~# testdisk newimage_cut_main
```

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

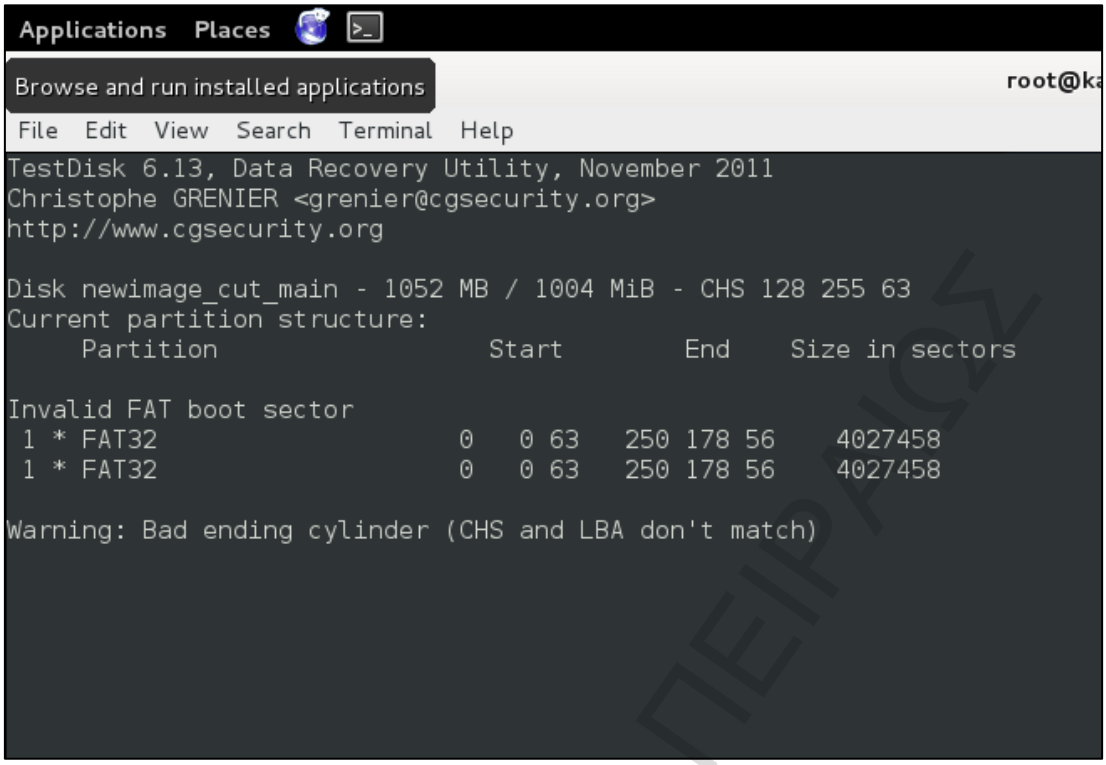
Ξεκινώντας την εφαρμογή και αναλύοντας την εικόνα μνήμης παρατηρούμε ότι το πρόγραμμα ανακαλύπτει ένα δίσκο με σύστημα αρχείων FAT32.Μια βαθύτερη ανάλυση της εικόνας μνήμης από το πρόγραμμα μας ενημερώνει για την ύπαρξη τριών partition tables τα οποία μέχρις στιγμής δεν είναι ανακτήσιμα.

Δυστυχώς, περεταίρω ανάλυση της εικόνας μνήμης απαιτείται ώστε να μπορέσει το πρόγραμμα TestDisk να ανακτήσει τα δεδομένα από την μνήμη NAND Flash.

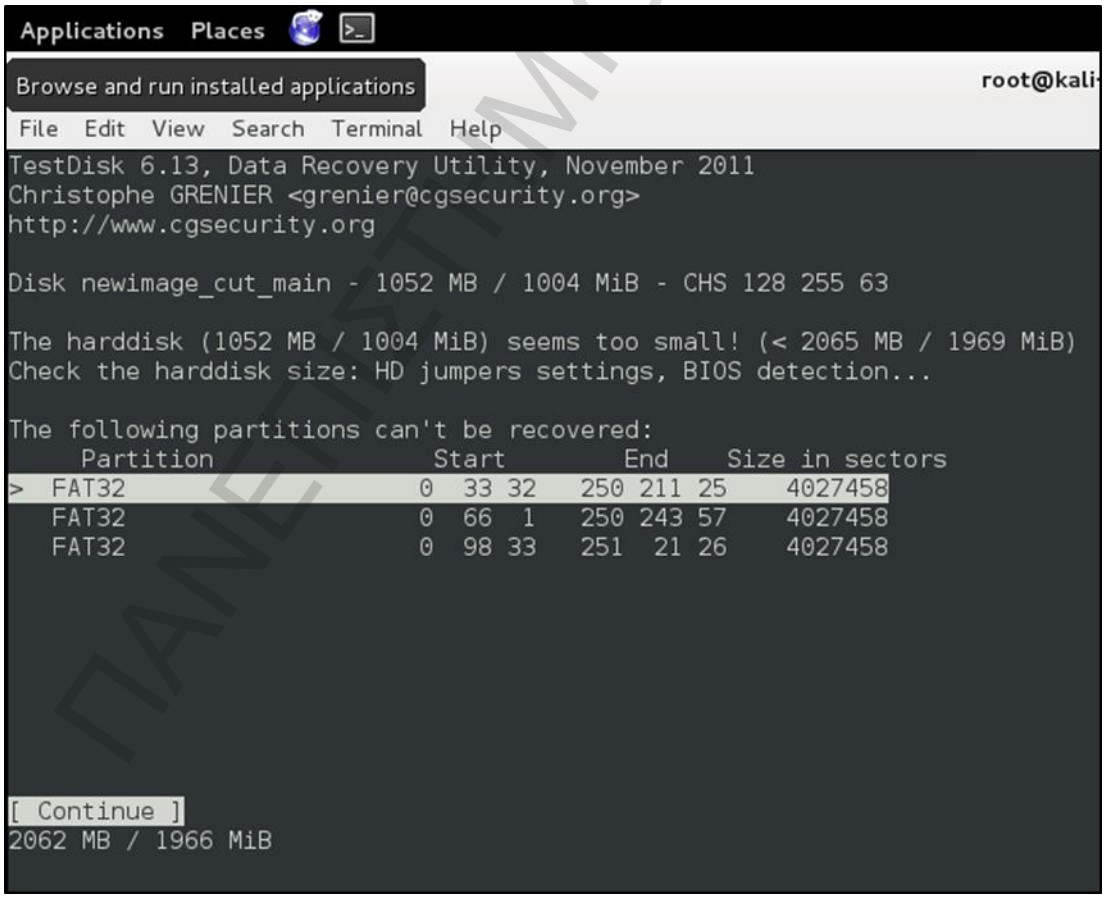
Τα block της εικόνας μνήμης που είχαμε δημιουργήσει περιέχουν και πληροφορίες από παλαιότερα block με δεδομένα που πλέων δεν είναι έγκυρα. Κατάλληλος μηχανισμός ανίχνευσης και περαιτέρω έρευνα απαιτείται ώστε να ανασυσταθεί η εικόνα μνήμης σωστά και να αποκτήσουμε τα δεδομένα από το USB stick.

Ο λόγος, διότι κάθε block μνήμης μπορεί να γραφτεί/τροποποιηθεί μέχρι τέσσερις φορές. Μετά, για να μπορέσει να ξαναχρησιμοποιηθεί πρέπει να προηγηθεί η ολική διαγραφή του block. Για την βελτιστοποίηση της απόδοσης της μνήμης NAND Flash, όταν ο διαθέσιμος κενός χώρος από block που υπάρχουν σε μια ζώνη περιοριστεί κάτω από ένα συγκεκριμένο ποσοστό, τα block που έχουν φτάσει στο μέγιστο αριθμό εγγραφής τους και δεν χρησιμοποιούνται πλέων, διαγράφονται όλα μαζί, ώστε να είναι ξανά διαθέσιμα μέσα στην ζώνη. Η συγκεκριμένη διαδικασία λειτουργίας θα αποτελούσε ίσως κρίσιμο όπλο για έναν ερευνητή που προσπαθεί να ανακτήσει πληροφορίες για δεδομένα που ο χρήστης θεωρεί ότι έχει διαγράψει ή για την ανάκτηση παλαιότερων εκδόσεων των αρχείων που υπήρχαν, μιας κι εξακολουθούν να υφίστανται μέσα στην εικόνα μνήμης. Δυσκολεύουν όμως το έργο του ερευνητή που προσπαθεί να ανακτήσει την τελευταία λογική κατάσταση της εικόνας μνήμης. Περεταίρω έρευνα απαιτείται για την αναγνώριση των έγκυρων block μέσα στην εικόνα μνήμης και την ανασυγκρότηση ενός υγιούς συστήματος αρχείων που θα ανταποκρίνεται στο σύστημα αρχείων που έχει χρησιμοποιηθεί από το ανώτερο λειτουργικό σύστημα και θα είναι ανακτήσιμο.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ



Εικόνα 5.5: TestDisk Ανάλυση Εικόνας Μνήμης



Εικόνα 5.6: TestDisk Ανάλυση Partition Tables Εικόνας Μνήμης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Κεφάλαιο 6ο

Συμπεράσματα

6.1 Ειδικά εργαλεία και ικανότητες

Η αποκόλληση ενός ολοκληρωμένου από την πλακέτα απαιτεί ειδικά εργαλεία αλλά και γνώσεις και εμπειρία ηλεκτρονικού. Για την εκτέλεση της εργασίας απαιτούνται ηλεκτρικά εργαλεία και μονάδες προγραμματισμού όπως, σταθμός βάσης αποκόλλησης ολοκληρωμένου (ζεστού αέρα ή υπέρυθρης ακτινοβολίας), προθερμαντήρες, εργαλεία καθαρισμού των επαφών του ολοκληρωμένου, αλλά και προγραμματιστικές μονάδες για την επικοινωνία με το τσιπ, καθώς επίσης και ειδικές βάσεις για την προσαρμογή του σε αυτές. Επιπλέον είναι επιθυμητός μεγεθυντικός φακός, για την καλύτερη οπτική παρατήρηση του ολοκληρωμένου και της πλακέτας στην οποία είναι τοποθετημένο.

Όπως γνωρίζεται οι προγραμματιστικές μονάδες χρειάζονται για την ανάγνωση των δεδομένων από το ολοκληρωμένο και υπάρχει ένας ευρύς αριθμός προμηθευτών που παρέχουν λύσεις για την ανάγνωση μνημών. Παρόλα αυτά καμία μονάδα προγραμματισμού δεν μπορεί να υποστηρίξει το σύνολο από τα χιλιάδες ολοκληρωμένα που διατίθενται στην αγορά, ανάμεσα σε ένα μεγάλο πλήθος κατασκευαστών. Για το λόγο αυτό, είναι κατανοητό ότι απαιτείται πρόσβαση στον ερευνητή σε ένα πλήθος μονάδων προγραμματισμού, οι οποίες πολλές φορές είναι αρκετά ακριβές.

Ένα άλλο εξάρτημα το οποίο απαιτείται μετά την αποκόλληση του ολοκληρωμένου, είναι η κατάλληλη βάση για την προσαρμογή του ολοκληρωμένου με την μονάδα προγραμματισμού. Οι συγκεκριμένες βάσεις χρησιμοποιούνται για την διευκόλυνση των ηλεκτρικών συνδέσεων της μονάδας προγραμματισμού με το ολοκληρωμένο. Ορισμένοι προγραμματιστές χρησιμοποιούν κοινής χρήσης βάση η οποία είναι συμβατή με πολλά ολοκληρωμένα, ενώ κάποιοι άλλοι απαιτούν εξειδικευμένη βάση ανάλογα με τον τύπο.

Τέλος, αξίζει να αναφέρουμε ότι οι ικανότητες που χρειάζονται για την αφαίρεση, τον καθαρισμό και την ανασύσταση των ακροδεκτών του ολοκληρωμένου, είναι εξειδικευμένες και απαιτούν πολλές ώρες εξάσκησης και εμπειρία στο συναφές αντικείμενο προκειμένου κάποιος να αποκτήσει τον τίτλο του επαγγελματία τεχνικού. Πολλές φορές απαιτείται η συνεργασία μιας ομάδας ατόμων, ώστε να επιτευχθούν τα επιθυμητά αποτελέσματα.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

6.2 Μελλοντικές εργασίες

Η επιλογή της ανάγνωσης δεδομένων από NAND Flash η οποία είναι προσαρτημένη σε USB stick αποτελεί μια από τις δυσκολότερες διαδικασίες λόγω της ανάλυσης που απαιτείται για την εικόνα μνήμης. Στις ηλεκτρονικές συσκευές (router, switch κ.α.) όπου χρησιμοποιούνται NAND Flash ολοκληρωμένα για την αποθήκευση αρχείων αλλά και του λειτουργικού συστήματος της ενθυλακωμένης συσκευής, η ανάλυση της εικόνας μνήμης είναι κατά πολύ πιο εύκολη. Το σύστημα αρχείων που χρησιμοποιείται για την εγγραφή των δεδομένων στην μνήμη NAND Flash μπορεί να προσδιοριστεί εύκολα με υπάρχοντα εργαλεία και το μόνο που απαιτείται από τον ερευνητή είναι η αφαίρεση των βοηθητικών δεδομένων και η προσάρτηση (mount) της εικόνας μνήμης σε ένα λειτουργικό σύστημα Linux.

Επίσης μεγάλο ενδιαφέρων θα αποτελούσε η γραφή κώδικα με σκοπό την δημιουργία προγράμματος το οποίο θα μας επιτρέπει την ανάγνωση δεδομένων από τις μνήμες NAND Flash 16 bit, βάσης TSOP 56, η οποία περιέχει ξεχωριστό δίαυλο για την διευθυνσιοδότηση της μνήμης. Οι συγκεκριμένες μνήμες NAND Flash είναι πιο γρήγορες στην εκτέλεση των λειτουργιών τους και αποτελούν αντικείμενο κατανάλωσης σε πολλές σύγχρονες ηλεκτρονικές συσκευές αλλά και στους νέους σκληρούς δίσκους SSD.

Η εγγραφή κώδικα για την ανάγνωση εικόνας μνήμης από NAND Flash 16 bit αποτελεί ισχυρό εργαλείο το οποίο μας επιτρέπει να εξάγουμε την υπάρχουσα κατάσταση του λειτουργικού συστήματος σε ενθυλακωμένες συσκευές, συμπεριλαμβανομένου και των πληροφοριών που είναι αποθηκευμένες σε αυτό. Με την συγκεκριμένη μέθοδο, οποιοσδήποτε μπορεί να αποκτήσει πρόσβαση στο λογισμικό ενός smartphone, ενός router, μιας ψηφιακής κάμερας και οποιασδήποτε άλλης ηλεκτρονικής συσκευής που περιέχει μνήμη NAND Flash, αφού όμως πρώτα έχει εξασφαλίσει φυσική πρόσβαση στο ολοκληρωμένο.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Βιβλιογραφικές Αναφορές

[1] FTDI, FT2232H mini module
http://www.ftdichip.com/Products/Modules/DevelopmentModules.htm

[2] FTDI, FT2232H Mini Module USB Hi-Speed FT2232H Evaluation Module Datasheet
http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_FT2232H_Mini_Module.pdf

[3] FTDI, Download Virtual COM Port Drivers
http://www.ftdichip.com/Drivers/VCP.htm

[4] FTDI, Download D2XX Drivers
http://www.ftdichip.com/Drivers/D2XX.htm

[5] FTDI, Drivers Installation Guide for Linux
http://www.ftdichip.com/Support/Documents/AppNotes/AN_220_FTDI_Drivers_Installation_Guide_for_Linux%20.pdf

[6] FTDI, Software Application Development D2XX Programmer's Guide
http://www.ftdichip.com/Support/Documents/ProgramGuides/D2XX_Programmer%27s_Guide%28FT_000071%29.pdf

[7] FTDI, MPSSE Example Projects
http://www.ftdichip.com/Support/SoftwareExamples/MPSSE.htm

[8] libFTDI - FTDI USB driver with bitbang mode
http://www.intra2net.com/en/developer/libftdi/index.php

[9] libFTDI - API documentation
http://www.intra2net.com/en/developer/libftdi/documentation/

[10] SpritesMods - FT2232H NAND flash reader
http://spritesmods.com/?art=ftdinand&page=1

[11] B.Kerler - NANDReader_FTDI
https://github.com/bkerler/NANDReader_FTDI

[12] Introduction to FTDI bitbang mode
http://hackaday.com/2009/09/22/introduction-to-ftdi-bitbang-mode/

[13] Reading bare NAND flash chips with a microcontroller
http://hackaday.com/2012/09/20/reading-bare-nand-flash-chips-with-a-microcontroller/

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

[14] Reverse engineering NAND Flash for fun and profit
http://h30499.www3.hp.com/t5/HP-Security-Research-Blog/Reverse-engineering-NAND-Flash-for-fun-and-profit/ba-p/6418140

[15] Memory Technology Devices - MTD
http://www.linux-mtd.infradead.org/archive/tech/nand.html

[16] Linux Cross Reference
Linux/drivers/mtd/nand/nand_ids.c

[17] MTD - NAND Extend MLC support
http://www.serverphorums.com/read.php?12,142485

[18] MTD NAND Driver Programming Interface
http://www.linux-mtd.infradead.org/tech/mtdnand/book1.html

[19] NAND Flash Support Table
http://www.linux-mtd.infradead.org/nand-data/nanddata.html

[20] ELNEC - Device: TH58NVG4D4CTG00 [TSOP48]
http://www.elnec.com/device/Toshiba/TH58NVG4D4CTG00+%5BTSOP48%5D/

[21] ebay - TSOP 48 socket
http://www.ebay.com/sch/i.html?_trksid=m570.l3201&_nkw=tsop+48+socket&_sacat=0

[22] Flash memory basics and its interface to a processor
http://www.eeherald.com/section/design-guide/esmod16.html

[23] Read Embedded Flash Chips
http://www.uchobby.com/index.php/2007/05/05/read-embedded-flash-chips/

[24] Flash Extractor
http://www.flash-extractor.com/manual/

[25] flashrom
http://www.flashrom.org/Flashrom

[26] Wikipedia - Flash memory
http://en.wikipedia.org/wiki/NAND_flash#NAND_memories

[27] NAND Flash memory in embedded systems by Michal Jedrak, Evatronix S.A.
http://www.design-reuse.com/articles/24503/nand-flash-memory-embedded-systems.html

[28] Texas Instruments Wiki – File system in NOR or NAND
http://processors.wiki.ti.com/index.php/Filesystem_in_NOR_or_NAND

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

[29] COEN 152 Computer Forensics Master Boot Record Example http://www.cse.scu.edu/~tschwarz/COEN252_09/Lectures/MBR-Example.html

[30] COEN 152 Computer Forensics FAT File Systems http://www.cse.scu.edu/~tschwarz/COEN252_09/Lectures/FAT.html

[31] Eureka Technology, " NAND Flash FAQ, apn5_b7"

[32] Toshiba America Electronic Components Inc., "NAND Flash Applications Design Guide, April 2003"

[33] Nina Mitiukhina, "Overview of the NAND Flash High-Speed Interfacing and Controller Architecture"

[34] Hyojun Kim, Youjip Won, "MNFS: Mobile Multimedia File System for NAND Flash based Storage Device"

[35] Amber Huffman (Senior Staff Architect, Intel), Michael Abraham (Sr. Applications Engineer, Micron), "A Standard Interface for NAND Flash"

[36] Rino Micheloni, Luca Crippa, Alessia Marelli, "Inside NAND Flash Memories"

[37] ScienceDirect, Jeong-Uk Kang, Jin-Soo Kim, Chanik Park, Hyoungjun Par, Joonwon Lee "A multi-channel architecture for high-performance NAND flash-based storage system"

[38] MICRON, "2Gb, 4Gb, 8Gb: x8, x16 NAND Flash Memory"

[39] MICRON, "Technical Note NAND Flash 101: An Introduction to NAND Flash and How to Design It In to Your Next Product"

[40] MICRON, "Technical Note Enabling a Flash Memory Device into the Linux MTD"

[41] MICRON, "Technical Note Wear Leveling in Micron® NAND Flash Memory"

[42] MICRON, "Technical Note Enabling Software BCH Error Correction Code (ECC) on a Linux Platform"

[43] ELNEC, "NAND Flash Memories and Programming NAND Flash Memories Using Elnec Device Programmers, January 2014"

[44] Australian Digital Forensics Conference, Security Research Institute Conferences: Krishnun Sansurooah, Edith Cowan University "A forensics overview and analysis of USB flash memory devices, 2009"

[45] SMALL SCALE DIGITAL DEVICE FORENSICS JOURNAL, VOL. 2, NO. 1, JUNE 2008 ISSN# 1941-6164, James Luck & Mark Stokes, "An Integrated Approach to Recovering Deleted Files from NAND Flash Data"

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

[46] Marcel Breeuwsma, Martien de Jongh, Coert Klaver, Ronald van der Knijff and Mark Roeloffs, “Forensic Data Recovery from Flash Memory”

[47] CHIP-OFF FORENSICS, “Extracting a full bit-stream image from devices containing embedded flash memory, by Jim Swauger”

[48] Dr Sergei Skorobogatov, “Fault attacks on secure chips: from glitch to flash, University of CAMBRIDGE, Computer Laboratory”

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ