ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	(Ελληνικά) Επιδράσεις της ακτινοβολίας νετρονίων σε πολυπύρηνο undervolted AMD x86 Επεξεργαστή - Μεθοδολογία και ανάλυση αποτελεσμάτων (Αγγλικά) Effects of Neutron Radiation in a multicore undervolted AMD x86 Processor - Methodology and analysis of the results
Ονοματεπώνυμο Φοιτητή	Αργυρίου Κωνσταντίνος
Πατρώνυμο	Χαράλαμπος
Αριθμός Μητρώου	Π/19017
Επιβλέπων	Ψαράκης Μιχάλης, Αναπληρωτής καθηγητής

Ημερομηνία Παράδοσης 07 2024

Περίληψη

Η ραγδαία αύξηση των επιδόσεων στους επεξεργαστές λόγω των μικροσκοπικών μεγεθών των τρανζίστορ, μεγέθη που φτάνουν μόλις μερικά νανόμετρα, και των τεχνολογιών που βελτιώνουν την απόδοση του επεξεργαστή σε υλικό επίπεδο, όπως η διοχέτευση και η ιεραρχία μνήμης, έχουν προκαλέσει μια επανάσταση στον χώρο, με ταχύτερους επεξεργαστές από ποτέ. Όμως, λόγω της πολύπλοκης αρχιτεκτονικής και όλο και αυξανόμενης ζήτησης για επιπλέον ταχύτητες, οι επεξεργαστές έφτασαν σε ένα σημείο που αρχίζουν και καταναλώνουν πολύ παραπάνω ενέργεια. Επιπλέον, λόγω των μικροσκοπικών αυτών μεγεθών που έχουν φτάσει τα τρανζίστορ, έχει έρθει στην επιφάνεια άλλο ένα σημαντικό πρόβλημα. Υψηλής ενέργειας σωματίδια προερχόμενα από κοσμικές πηγές ακτινοβολίας είναι σε θέση να αλλάζουν την κατάσταση ενός τρανζίστορ [1], αν δοθεί το απαραίτητο φορτίο, ανοίγοντας έτσι συζητήσεις περί ακεραιότητας των δεδομένων που βρίσκονται στις **Caches** και στην σωστή λειτουργία του επεξεργαστή.

Λέξεις Κλειδιά

Επεξεργαστής, ακτινοβολία, AMD, undervolt, πείραμα, σφάλματα, καταχωρητής, αρχιτεκτονική

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Πρόλογος

Η τεχνολογία στις μέρες μας έχει φτάσει σε επίπεδα που πλέον ένας μικρό ελεγκτής μπορεί να έχει μέχρι και δισεκατομμύρια τρανζίστορ [1]. Αυτό το γεγονός έχει κάνει τους επεξεργαστές και τους μικρο ελεγκτές πολύ ποιο γρήγορους και ποιο περίπλοκους από ποτέ, φέρνοντας έτσι στην επιφάνεια νέα προβλήματα προς αντιμετώπιση.

Σε αυτή την εργασία θα γίνει αναφορά σε δύο προβλήματα που έχουν έρθει στην επιφάνεια στον κλάδο. Το πρώτο αποτελεί το πρόβλημα της κατανάλωσης ενέργειας των νέων επεξεργαστών, λόγω της πολυπλοκότερης αρχιτεκτονικής, όπως η τύπου **CISC x86** αρχιτεκτονική που μελετάτε στην εργασία, και των επιπλέον λειτουργιών που παρέχονται, όπως η δυνατότητα να υπάρχουν όλο και περισσότεροι πυρήνες σε ένα μόνο **chip**. Το δεύτερο πρόβλημα που μελετάτε είναι αυτό της αξιοπιστίας ενός **x86** επεξεργαστή, όταν αυτός εκτίθεστε σε υψηλή ιοντίζουσα ακτινοβολίας προερχόμενη από μια ακτίνα νετρονίων. Αυτά τα θέματα και ο τρόπος με τον οποίο θα πρέπει ένα πείραμα ακτινοβολίας να βγει εις πέρας είναι τα θέματα αυτής της παρούσας εργασίας.

Κεφάλαιο 1

Εισαγωγή

Τα μικροσκοπικά μεγέθη που έχουν φτάσει τα τρανζίστορ έχουν επιτρέψει μια ανοδική πορεία στις επιδόσεις των υπολογιστικών συστημάτων, αφού πλέον σε έναν μικρό ελεγκτή είναι δυνατό να υπάρχουν δισεκατομμύρια τέτοια τρανζίστορ [1]. Τα πολυεπεξεργαστικά συστήματα είναι μια από τις δυνατότητες που επέτρεψαν την ραγδαία αυτή αύξηση στην ταχύτητα τους. Άλλη παράγοντες που συνείσφεραν στην σημερινή ταχύτητα των υπολογιστών είναι μέθοδοι που βελτιστοποιούν την λειτουργία ενός επεξεργαστή, όπως για παράδειγμα μέθοδοι προβλέψεις διακλάδωσης (**Branch Predictions**), η μέθοδος της διοχέτευσης (**Pipeline**) και η ιεραρχική δομή των μνημών.

Μαζί όμως με τα πλεονεκτήματα που επιφέρουν οι μηχανισμοί βελτιστοποίησης στο εσωτερικό του επεξεργαστή και το μέγεθος των τρανζίστορ, έρχονται και κάποια σοβαρά μειονεκτήματα. Δύο από τα βασικότερα μειονεκτήματα είναι αυτό της κατανάλωσης ενέργειας ενός επεξεργαστή και της αξιοπιστίας του. Το πρώτο, η κατανάλωση της ενέργειας, έχει κριθεί ως κρίσιμης σημασίας, καθώς οι επεξεργαστές καταναλώνουν όλο και περισσότερο ενέργεια, έτσι ώστε να μπορούν να βγάλουν εις πέρας τις απαιτήσεις για όλο και περισσότερη ταχύτητα. Το δεύτερο, η αξιοπιστία, έχει να κάνει με το πόσο ευάλωτος είναι ο επεξεργαστής σε διάφορες συνθήκες και περιβάλλοντα. Μάλιστα, το πρόβλημα της αξιοπιστίας πήρε σημαντικές διαστάσεις όταν άρχισε να υπάρχει η υποψία πως υψηλής ενέργειας σωματίδια προερχόμενα από κοσμικές πηγές, όπως εκρήξεις super nova αστέρων, είναι δυνατόν να εισέλθουν στην ατμόσφαιρά της Γης και να διαπεράσουν των επεξεργαστή [1]. Αυτό με την σειρά του δημιουργεί το πρόβλημα πως λόγω των μεγεθών των τρανζίστορ, είναι πιθανό ένα από αυτά τα σωματίδια να το φορτίσουν αρκετά, έτσι ώστε το τρανζίστορ να αλλάξει κατάσταση από **μηδέν** σε **ένα** ή το αντίστροφο [1], κάτι που μπορεί να είναι καταστροφικό αν αυτό το τρανζίστορ είναι μέλος ενός σημαντικού υπό συστήματος του επεξεργαστή ή αν είναι τρανζίστορ μιας κρυφής μνήμης (**Cache**) στο εσωτερικό του επεξεργαστή, φέρνοντας έτσι στην επιφάνεια πρόβλημα ακεραιότητας των δεδομένων. Στην βιβλιογραφία, ή απαραίτητη φόρτιση που χρειάζεται προκειμένου ένα τρανζίστορ να αλλάξει κατάσταση φέρνει το όνομα **Critical Charge** και στην περίπτωση που αλλάξει η κατάσταση του, τότε, αναλόγος με το αν προκλήθηκε κάποιο σφάλμα στο σύστημα, αυτό κατηγοριοποιείται είτε ως **SDC** (**Silent Data Corruption**) ή ως **DUE (Detectable Unreverable Error)**. Περισσότερες λεπτομέρειες για τους δύο τύπους σφαλμάτων μπορούν να βρεθούν στην ενότητα (4.1.2).

1.1 Κίνητρο

Το κίνητρο πίσω από αυτή την εργασία και άλλον πολλών ερευνών είναι να μελετηθεί το πώς μπορεί να επιτευχθεί καλύτερη εξοικονόμηση ενέργειας στους σημερινούς επεξεργαστές, χωρίς όμως να θυσιάζεται ή επίδοση του επεξεργαστή ή η αξιοπιστία που προσφέρετε από τους κατασκευαστές.

1.2 Συνεισφορές της εργασίας

Η εργασία αυτή παρουσιάζει ένα ισχυρό εργαλείο για την πραγματοποίηση πειραμάτων ακτινοβολίας, δεδομένα από ένα πραγματικό πείραμα στο οποίο επεξεργαστές εκτέθηκαν σε μια ακτίνα νετρονίων και μια μέθοδο με την οποία είναι δυνατή η αξικονόμηση της κατανάλωσης ενέργειας ενός **AMD** επεξεργαστή. Παρακάτω θα γίνει αναλυτική περιγραφή αυτών.

1.2.1 undervolting

Με την μέθοδο του **Undervolting**, όπως θα περιγραφεί αναλυτικά και στο κεφάλαιο (3.1), δύνεται η δυνατότητα μείωσης της λειτουργικής τάσης του επεξεργαστή, με σκοπό την μείωση της συνολικής κατανάλωσης του. Στην συγκεκριμένη εργασία γίνεται χρήση ενός **AMD** επεξεργαστή και έτσι ή μέθοδος που παρουσιάζεται στο αντίστοιχο κεφάλαιο μπορεί να μην είναι ακριβώς ίδια σε κάθε επεξεργαστή. Όμως γίνεται αναφορά σε μια γενική διαδικασία εύρεσης των απαραίτητων στοιχείων, όπως καταχωρητές, που συμβάλουν σημαντικά στο να επιτευχθεί η μείωση της τάσης σε έναν επεξεργαστή.

1.2.2 Πειράματα ακτινοβολίας

Τα πειράματα ακτινοβολίας είναι σημαντικά καθώς μπορούν να προσφέρουν πληροφορίες για την συμπεριφορά και την αξιοπιστία ενός επεξεργαστή στο βάθος του χρόνου. Με αυτά τα πειράματα είναι δυνατή η μελέτη και η βελτίωση των επεξεργαστών, ώστε να είναι ισχυρότερη απέναντι σε περιβάλλοντα που εκθέτουν των επεξεργαστή σε υψηλή ακτινοβολία.

1.2.3 Framework

Μια σημαντική παρουσίαση της εργασίας έχει να κάνει με ένα λογισμικό, το οποίο αποκαλείται Symphony. Το λογισμικό αυτό είναι σχεδιασμένο με τέτοιο τρόπο, έτσι ώστε να μπορεί να διαχειριστεί αποδοτικά και με τον ελάχιστο ανθρώπινο παράγοντα πειράματα ακτινοβολίας. Στο κεφάλαιο (4) θα γίνει αναλυτική περιγραφή της αρχιτεκτονικής του εν λόγο εργαλείου και στο πως διαχειρίζεται καταστάσεις που υπό άλλες συνθήκης θα έπρεπε να υπάρχει ο ανθρώπινος παράγοντας.

Κεφάλαιο 2

Υπόβαθρο

2.1 Machine Check Architecture

Στους επεξεργαστές της **x86** αρχιτεκτονικής υπάρχει μια δυνατότητα ή οποία επιτρέπει στο λογισμικό του υπολογιστή, συνήθως του πυρήνα του λειτουργικού συστήματος, να λαμβάνει και να διαχειρίζεται σφάλματα που μπορεί έχουν προκληθεί πάνω στο υλικό του. Αυτό έγινε δυνατό με την χρήση μιας αρχιτεκτονικής που ονομάστηκε **Machine Check Architecture**. Παρακάτω θα γίνει μια σύντομη εισαγωγή όσον αφορά αυτήν την αρχιτεκτονική.

2.1.1 Αρχιτεκτονική

Η αρχιτεκτονική αυτή αποτελείται από ένα υπό σύνολο από **MSR** καταχωρητές (3.1.2) που αποκαλούνται **MC** (**Machine Check**) καταχωρητές. Ο κάθε πυρήνας του επεξεργαστή περιέχει ένα αποκλειστικά δικό του σύνολο από αυτούς τους καταχωρητές και δύνεται η δυνατότητα από το λογισμικό να προσπελάσει και να διαβάσει τους αντίστοιχους καταχωρητές οποιουδήποτε από τους διαθέσιμους πυρήνες.

Οι πληροφορίες που περιέχονται μέσα στους **MC** καταχωρητές μπορούν να χρησιμοποιηθούν για μια πληθώρα διαδικασιών, αλλά ο κύριος σκοπός τους είναι να στέλνουν διαγνωστικές πληροφορίες στο λογισμικό, έτσι ώστε να παρθούν οι απαραίτητες διαδικασίες διόρθωσης, σε περίπτωση σφάλματος.

Στην περίπτωση που ο επεξεργαστής, **x86 αρχιτεκτονικής**, αντιληφθεί πως προέκυψε κάποιο σφάλμα σε κάποιο υπό σύστημα, τότε, πυροδοτείται μια εξαίρεση (**Exception**) ή οποία φέρνει τον αριθμό **int 18 (διακοπή 18, interrupt 18)** στην αρχιτεκτονική **x86**. Με το που πυροδοτηθεί μια τέτοια εξαίρεση ο πυρήνας του λειτουργικού συστήματος σταματάει όποια διαδικασία εκτελούσε και ύστερα προσπαθεί να διαχειριστεί την εξαίρεση με την χρήση μιας από τις ρουτίνες του. Η ρουτίνα του λειτουργικού συστήματος στην συνέχεια προσπαθεί να διαγνώσει το πρόβλημα, διαβάζοντας τους **MC** καταχωρητές και προσπαθεί να επιλύσει το πρόβλημα, αν μπορεί, με χρήση αξιδικευμένου λογισμικού. Μερικές από τις πληροφορίες που παρέχονται από την αρχιτεκτονική **MCA** μπορούν να φανούν παρακάτω.

- **ECC**, πληροφορία για εντοπισμό και διόρθωσης σφαλμάτων.
- **Διορθωμένα σφάλματα (Corrected Errors)**, σφάλματα που το σύστημα διόρθωσε.
- **Μη διορθωμένα σφάλματα (Uncorrected Errors)**, σφάλματα τα οποία δεν ήταν δυνατό να διορθωθούν.

2.1.2 MCA στους AMD

Στους **AMD** επεξεργαστές και ειδικότερα στους επεξεργαστές της γενιάς **17H** υπάρχουν συνολικά 96 **MCA Banks** από τα οποία τα 32 ανήκουν σε μια κατηγορία που ή **AMD** την ονομάζει **Legacy MCA** και άλλα 64 που κατηγοριοποιούνται στην ομάδα που ονομάζει **Machine Check Architecture eXtensions** ή **MCAX**. Τα 96 αυτά banks βρίσκονται σε **κάθε** έναν από τους πυρήνες του επεξεργαστή.

Στην **AMD** αρχιτεκτονική τα σφάλματα κατηγοριοποιούνται σε τρεις ομάδες σφαλμάτων, που αναφέρονται παρακάτω.

- **Uncorrected**, αποτελούν σφάλματα τα οποία δεν διορθώθηκαν από το λογισμικό ή το υλικό και τα οποία επηρεάζουν την σωστή λειτουργία του συστήματος.
- **Defered**, αποτελούν σφάλματα τα οποία δεν προκαλούν αρκετά σοβαρό πρόβλημα ώστε να σταματήσει η ροή της εκτέλεσης του επεξεργαστή. Αλλά ο πυρήνας του λειτουργικού συστήματος ενημερώνεται προκειμένου να πάρει τα απαραίτητα μέτρα μόλις κρίνει πως πρέπει.
- **Corrected**, αποτελούν σφάλματα τα οποία συνέβησαν σε κάποια μονάδα του συστήματος, αλλά έγινε ο κατάλληλος χειρισμός τους και το σύστημα κατάφερε να τα διορθώσει.

Οι παραπάνω ομάδες σφαλμάτων έχουν δείκτη σημαντικότητας με την σειρά που δόθηκε, από το σημαντικότερο προς το λιγότερο σημαντικό. Επιπλέον πληροφορίες που περιέχουν οι καταχωρητές **MC** στους **AMD** επεξεργαστές είναι επίσης οι παρακάτω.

- **TLB**, το σφάλμα προκλήθηκε στην μνήμη TLB
- **Memory**, το σφάλμα προκλήθηκε σε μια από τις μνήμες του συστήματος.
- **Bus**, Το σφάλμα προκλήθηκε στο Bus, για παράδειγμα όσο πληροφορίες έρχονταν από μια συσκευή εισόδου/εξόδου προς στον επεξεργαστή.

Οι παραπάνω πληροφορίες στην συνέχεια μπορούν να αναλυθούν ακόμα περισσότερο, φτάνοντας στο επίπεδο των εσωτερικών μονάδων του επεξεργαστή.

Κεφάλαιο 3

Μείωση της τάσης του επεξεργαστή

Σε αυτό το κεφάλαιο θα γίνει εις βάθος περιγραφή των διαδικασιών που θα πρέπει να γίνουν, με σκοπό την επίτευξη της μείωσης της τάσης του ενδιαφερόμενου επεξεργαστή, η αλλιώς του **undervolting**. Η διαδικασία είναι διαφορετική για κάθε αρχιτεκτονική και έτσι δεν είναι δυνατόν να δοθεί ένας γενικευμένος τρόπος για όλες τις αρχιτεκτονικές. Ο επεξεργαστής που χρησιμοποιήθηκε στην περίπτωση της εργασίας, είναι ο **AMD Ryzen 5 2400G Pro**, της **x86** αρχιτεκτονικής. Οι διαφοροποιήσεις που μπορεί να έχουν μεταξύ τους οι επεξεργαστές της αρχιτεκτονικής αυτής, ως προς το πως γίνεται ο έλεγχος της τάσης έχει να κάνει με το σε ποια διεύθυνση βρίσκεται ο **Model-Specific Register** (MSR) (ενότητα 3.1) καταχωρητής που ελέγχει την τάση του επεξεργαστή, σε ποια **bits** του καταχωρητή αυτού ορίζεται το πεδίο που καθορίζει την τάση του επεξεργαστή και το αν ο κατασκευαστής επιτρέπει η όχι την εγγραφή του εν λόγω καταχωρητή. Στην περίπτωση της εργασίας, θα γίνει αναφορά με βάση την **x86** αρχιτεκτονική που έχει υλοποιημένη ο κατασκευαστής **AMD**, ο οποίος επιτρέπει τόσο την ανάγνωση, όσο και την εγγραφή του εν λόγω καταχωρητή.

Ο αναγνώστης μετά την μελέτη του κεφαλαίου αυτού θα είναι σε θέση τόσο να καταλαβαίνει το πως ο επεξεργαστής παρέχει την δυνατότητα ελέγχου της τάσης, όσο και το να μπορεί να το κάνει και ίδιος, δεδομένου ότι έχει ένα μοντέλο επεξεργαστή, το οποίο να είναι συμβατό με τις διαδικασίες που θα αναφερθούν.

3.1 Αρχιτεκτονική

Σε αυτή την ενότητα θα γίνει μια γενική περιγραφή της αρχιτεκτονικής πίσω από τον έλεγχο της τάσης, θα γίνουν αναφορές δηλαδή στο τι είναι οι **MSR** καταχωρητές και το πως μέσω αυτών είναι δυνατόν να γίνει έλεγχος ενός άλλου πολύ σημαντικού στοιχείου του επεξεργαστή, που αποκαλείται **P-states**, που αυτό με την σειρά του ελέγχει την τάση του επεξεργαστή.

3.1.1 Τι είναι τα P-states

Οι κατασκευαστές των μοντέρνων επεξεργαστών της x86 αρχιτεκτονικής έχουν εμπλουτίσει το υλικό τους με διάφορες λειτουργικότητες, μια από τις οποίες είναι η τα **P-states (Power States)**. Τα **P-states** αποτελούν μια λειτουργικότητα με την οποία δίνεται η δυνατότητα διαχείρισης της κατανάλωσης ενέργειας ενός επεξεργαστή, κάτι που μπορεί να εκμεταλλευτεί το λειτουργικό σύστημα προκειμένου να κάνει εξοικονόμηση ενέργειας, εφόσον

κρίνει πως είναι απαραίτητο.

Τα **P-states** διαχειρίζονται την κατανάλωση ενέργειας επηρεάζοντας την τάση και την συχνότητα του επεξεργαστή. Για αυτόν τον σκοπό υπάρχει ένας καθορισμένος αριθμός από **P-states** που ορίζεται για έναν επεξεργαστή, ξεκινώντας την αρίθμηση από το 0 μέχρι έναν αριθμό *n*. Το P-state με αριθμό 0, είναι αυτό που καθορίζει την υψηλότερη κατανάλωση, ενώ αυτό με τον αριθμό n είναι αυτό που καθορίζει την λιγότερη κατανάλωση, δηλαδή, ποιο υψηλός αριθμός του P-state, συνεπάγεται σε λιγότερη κατανάλωση ενέργειας. Επομένως, αν ο σκοπός είναι η μείωση της τάσης του επεξεργαστή, αυτό που είναι απαραίτητο είναι να καθοριστεί διαφορετικό άνω φράγμα στην τάση του επεξεργαστή που ορίζεται στο **P-state 0**, με αυτόν τον τρόπο ο επεξεργαστής θα αναγκαστεί να καταναλώνει, **το πολύ**, μέχρι την ανάλογη κατανάλωση της τάσης που ορίστηκε σε αυτό το P-state. Στην συνέχεια, θα γίνει αναφορά για το πως γίνεται να αλλάξει η τιμή ενός οποιουδήποτε P-state με την χρήση των **MSR** καταχωρητών.

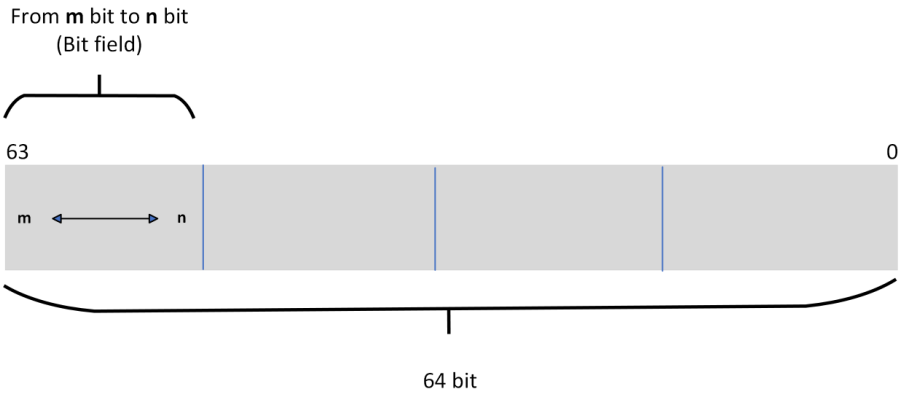
3.1.2 MSR καταχωρητές

Η **x86** αρχιτεκτονική παρέχει κάποιους καταχωρητές που αποκαλούνται ¨καταχωρητές ελέγχου" (**Control Registers**), οι καταχωρητές αυτοί χρησιμοποιούνται έτσι ώστε να αλλάζουν την συμπεριφορά του επεξεργαστή, η κάποιας συσκευής. Μερικές χρήσεις είναι για τον έλεγχο των interrupts του επεξεργαστή, η ακόμα και τον έλεγχο της λειτουργίας του συν επεξεργαστή (Coprocessor), αν αυτός υπάρχει.

Οι **MSR** καταχωρητές αποτελούν ένα υποσύνολο των καταχωρητών ελέγχου της x86 αρχιτεκτονικής, που συνήθως χρησιμοποιούνται για έλεγχο σφαλμάτων και για επιπρόσθετες λειτουργίες που συνήθως είναι αποκλειστικές για τον εν λόγω επεξεργαστή. Κάθε κατασκευαστής μπορεί να ορίσει δικές τους επιπρόσθετες λειτουργίες και μερικοί κατασκευαστές μπορεί να αφαιρούν η να προσθέτουν επιπλέον λειτουργίες σε κάθε νέα γενιά επεξεργαστών που παράγει. Για παράδειγμα, ο κατασκευαστής **Intel**, έχει ορίσει πως κάποιοι από τους MSR καταχωρητές θα πρέπει να είναι μόνιμοι και σταθεροί σε κάθε γενιά επεξεργαστών και τους οποίους τους αποκαλεί "MSR - της αρχιτεκτονικής" (Architectural MSR's), αλλά οι υπόλοιποι καταχωρητές από αυτούς, είναι αβέβαιο αν θα υπάρχουν σε επόμενη γενιά.

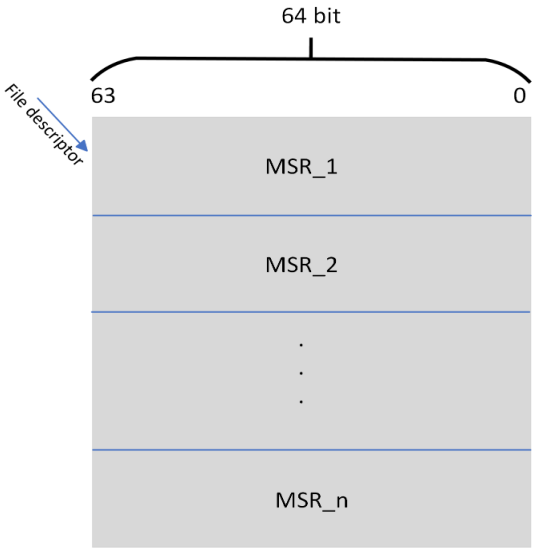
Ένας MSR καταχωρητής είναι ένας καταχωρητής ελέγχου των 32 bit (ή 64 bit) και χωρίζεται σε έναν καθορισμένο αριθμό πεδίων (Σχήμα 3.1). Τα πεδία καθορίζονται αυστηρά από τον κατασκευαστή ως ένα εύρος από bits, από το ***m-bit*** εώς το ***n-bit***, του καταχωρητή. Γράφοντας ένα συγκεκριμένο πεδίο με κάποια τιμή μπορεί να απενεργοποιεί, ενεργοποιεί ή να επηρεάζει κάποια λειτουργία του επεξεργαστή, όπως την τάση η την συχνότητα.

Μια από τις λειτουργίες που είναι δυνατόν να παρέχεται από αυτούς τους καταχωρητές ελέγχου είναι αυτή της διαχείρισης της κατανάλωσης ενέργειας του επεξεργαστή, δηλαδή είναι δυνατόν μέσω αυτών των καταχωρητών να τροποποιηθεί κάποιο P-state του επεξεργαστή, γράφοντας μια συγκεκριμένη τιμή σε ένα από τα πεδία του MSR καταχωρητή. Το P-state που επηρεάστηκε από την εγγραφή της τιμής αυτής στην συνέχεια καθορίζει την νέα, πλέον, κατάσταση του επεξεργαστή. Αναλόγως το πεδίο του MSR που θα τροποποιηθεί, μπορεί να επηρεαστεί είτε η τάση ή η συχνότητα του επεξεργαστή.



Σχήμα 3.1: *Παράδειγμα ενός MSR καταχωρητή - **AMD - Zen 2***

Τέλος, οι MSR καταχωρητές μπορούν να προσπελαστούν από το λειτουργικό σύστημα Linux μέσω ενός αρχείου που αποκαλείται msr και βρίσκεται στην τοποθεσία /dev/cpu/CPUNUM/msr, όπου CPUNUM αντιστοιχεί σε έναν αριθμό πυρήνα του επεξεργαστή. Εκεί το Linux έχει απεικονίσει όλους τους MSR καταχωρητές ενός πυρήνα ως αποστάσεις, σε bytes, από την αρχή του αρχείου (Σχήμα 3.2). Προκειμένου να τροποποιηθεί ένας από τους διαθέσιμους καταχωρητές, αρκεί να βρεθεί ο αριθμός των bytes, που αν προστεθεί στον δείκτη του αρχείου αυτού, ξεκινώντας από την αρχή, θα καταλήξει να βρίσκεται στην αρχή του καταχωρητή που ζητήθηκε. Αφού ο δείκτης του αρχείου msr δείξει στο πρώτο byte του καταχωρητή που ζητήθηκε, από εκεί και πέρα αρκεί να διαβαστούν (η να γραφούν) bytes ίσα με το μέγεθος του εν λόγω καταχωρητή. Στην περίπτωση που γίνει διάβασμα του καταχωρητή, το Linux επιστρέφει τα bytes του καταχωρητή. Αλλά, στην περίπτωση που γράφουν bytes σε αυτό το αρχείο, το Linux θα αλλάξει τα περιεχόμενα των καταχωρητών, που κατέχουν τα συγκεκριμένα bytes.



Σχήμα 3.2: *Δομή του /dev/cpu/CPUNUM/msr αρχείου*

3.2 Μεθοδολογία

Υπάρχουν δύο τρόποι με τους οποίους μπορεί να επιτευχθεί η μείωση της τάσης ενός επεξεργαστή, x86 αρχιτεκτονικής, από την προτεινόμενη τάση σε κάποια χαμηλότερη. Ο ένας τρόπος είναι να γίνει χειροκίνητα μέσω της βασικής συσκευής εισόδου εξόδου (BIOS), κάτι που οι μοντέρνες μητρικές που υπάρχουν στην αγορά, συνήθως, το επιτρέπουν. Ο δεύτερος τρόπος είναι να γίνει μέσω της διεπαφής που προσφέρει του λειτουργικού συστήματος προς τους καταχωρητές του επεξεργαστή. Συγκεκριμένα, τόσο το BIOS όσο και η διεπαφή του λειτουργικού συστήματος, αυτό που κάνουν είναι να τροποποιούν τους MSR (ενότητα 3.1) καταχωρητές, με τέτοιο τρόπο, ούτως ώστε να αλλάξουν την τάση του επεξεργαστή, ρυθμίζοντας τα p-states (ενότητα 3.1) του.

Λόγο της φύσης του πειράματος που διεξάχθηκε δεν θα ήταν εύχρηστο να γίνει χειροκίνητη τροποποίηση της τάσης του επεξεργαστή και έτσι, στην συνέχεια, θα γίνει αναφορά μόνο για την τροποποίηση της τάσης του επεξεργαστή μέσω της διεπαφής του λειτουργικού συστήματος Linux.

3.2.1 Εύρεση του MSR καταχωρητή που επηρεάζει την τάση

Όπως αναφέρθηκε και στην ενότητα (3.1), οι MSR καταχωρητές απεικονίζονται στο λειτουργικό σύστημα Linux μέσω κάποιον συγκεκριμένων διευθύνσεων, που οι διευθύνσεις αυτές αντιστοιχούν σε μια τοποθεσία μέσα στο περιεχόμενο ενός αρχείο που παρέχεται από το Linux, που βρίσκεται στην τοποθεσία `/dev/cpu/CPUNUM/msr`, όπου `CPUNUM` είναι ο αριθμός ενός πυρήνα του επεξεργαστή. Αν πρέπει να γίνει τροποποίηση των περιεχομένων ενός MSR καταχωρητή, τότε, θα πρέπει να βρεθεί η διεύθυνση, η απόσταση δηλαδή από την αρχή του αρχείου msr, που αντιστοιχεί στον συγκεκριμένο καταχωρητή ενδιαφέροντος.

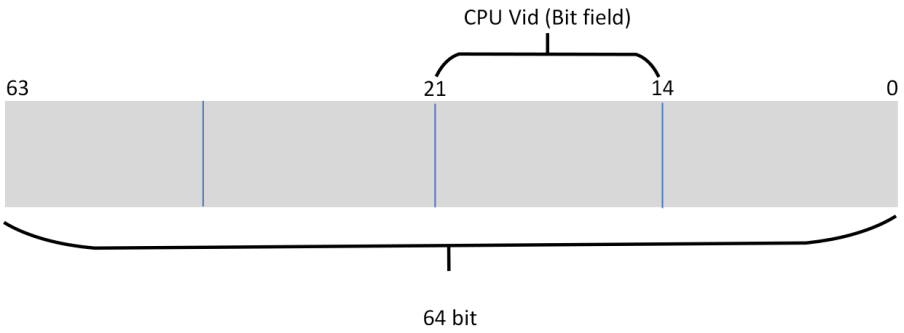
Στην συγκεκριμένη περίπτωση ο καταχωρητής ενδιαφέροντος είναι αυτός που μπορεί να αλλάξει την τάση του επεξεργαστή, στον πίνακα (3.1), φαίνεται η διεύθυνση βάσης του MSR καταχωρητή, που είναι υπεύθυνος, τόσο για την τροποποίηση της τάσης του επεξεργαστή, όσο και για την τροποποίηση της συχνότητας του.

MSR καταχωρητής ελέγχου των P-states	
Διεύθυνση	0xC001006[4...B]

Πίνακας 3.1: *Διεύθυνση του ενδιαφερόμενου MSR καταχωρητή - **AMD - Zen 2***

Όπως φαίνεται στον παραπάνω πίνακα 3.1, ο καταχωρητής προσδιορίζεται από μια 32-bit διεύθυνση, η οποία ορίζει πως μπορούν να υπάρχουν 8 p-states και ένας καταχωρητής για κάθε ένα από αυτά, όπου το πρώτο p-state βρίσκεται στην διεύθυνση `0xC0010064` και το τελευταίο στην διεύθυνση `0xC001006B`. Το περιεχόμενο των καταχωρητών αυτών αποτελείται από μια λέξη των 64-bit, σε αυτή την λέξη βρίσκονται διάφορα πεδία, που αντιστοιχούν σε κάποια καθορισμένα bytes αυτής της λέξεις. Ένα από αυτά τα πεδία ονομάζεται **CpuVID**, το πεδίο αυτό μπορεί να περιέχει πληροφορία η οποία αντιστοιχεί σε ένα byte, και ορίζεται ως το διάστημα, από το 21ο έως το 14ο bit των καταχωρητών (Σχήμα 3.3). Το **CpuVID** παίζει

καθοριστικό ρόλο στην διαδικασία της μείωσης της τάσης του επεξεργαστή, καθώς η αλλαγή τιμής σε αυτό το πεδίο, επηρεάζει την τάση που θα έχει ο επεξεργαστής.



Σχήμα 3.3: *Παράδειγμα του MSR καταχωρητή με υπόδειξη του* ***CPUVid*** *-* ***AMD - Zen 2***

3.2.2 Αλγόριθμος μείωσης της τάσης

Για να υλοποιηθεί αλγοριθμικά ένας τρόπος με τον οποίο θα είναι δυνατόν να μειωθεί η τάση του επεξεργαστή κατά βούληση, θα πρέπει να είναι δυνατόν να γίνουν τα εξής:

- Να γίνεται ανάγνωση/εγγραφή του περιεχομένου ενός MSR καταχωρητή
- Να προσδιοριστεί ο καταχωρητής που χρειάζεται μέσα από το αρχείο msr που παρέχει το λειτουργικό σύστημα Linux.
- Να προσδιοριστεί το εύρος των bits, από m-bit μέχρι n-bit, που αντιστοιχούν στο πεδίο CpuVid του καταχωρητή.

Ο αλγόριθμος (3.1) που φαίνεται παρακάτω, χρησιμοποιεί την διεύθυνση βάσης των καταχωρητών που αντιστοιχούν στα P-states του επεξεργαστή, όπως φαίνεται και στον πίνακα 3.1. Η διεύθυνση βάσης αντιστοιχεί στην διεύθυνση **0xC0010064** που πιο συγκεκριμένα, είναι η τοποθεσία του **P-state 0**. Όπως περιγράφηκε και στην ενότητα (3.1), τροποποιώντας αυτό το P-state επιτυγχάνεται η μείωση της τάσης, θέτοντας ένα άνω φράγμα στην κατανάλωση του επεξεργαστή.

μ 3.1: *Undervolt Algorithm*

msraddr ← 0*xC*0010064 ▷ MSR base address - P-state 0
corenum ← *N* ▷ Number of cores
while *corenum* ≠ 0 **do**
    writeCpuVidFieldOfMsr(msraddr, newCpuVid, *corenum*)
    corenum ← *corenum* − 1
end while

Προκειμένου να είναι δυνατή η μείωση της τάσης του επεξεργαστή, θα πρέπει να γίνει προσαρμογή των MSR καταχωρητών **όλων** των πυρήνων με τον ίδιο τρόπο.

3.2.3 Προετοιμασία του λειτουργικού συστήματος

Στην εργασία χρησιμοποιήθηκε το λειτουργικό σύστημα Linux, και έτσι τα επόμενα βήματα θα αποτελούν εντολές τερματικού Linux.

Για να είναι δυνατή η προσπέλαση των MSR καταχωρητών, απαιτείται να είναι ενεργοποιημένο ένα kernel modules, το οποίο χρησιμοποιείται προκειμένου το Linux να μπορεί να απεικονίσει τους καταχωρητές στο αρχείο msr. Το kernel module που απαιτείται ονομάζεται **msr** και είναι δυνατόν να ενεργοποιηθεί με την εντολή:

```
sudo modprobe msr
```

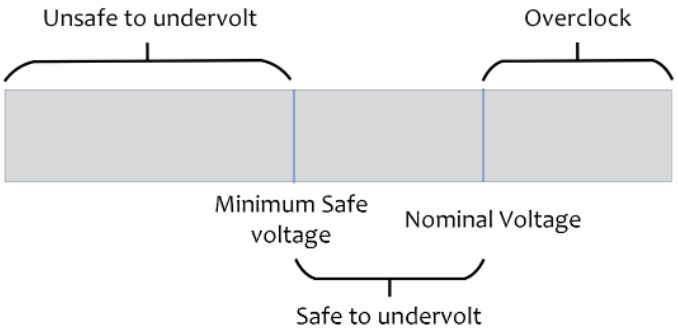
Μετά την εκτέλεση αυτής της εντολής, θα είναι δυνατόν από τον χρήστη του λειτουργικού συστήματος να προσπελάσει το αρχείο msr στην τοποθεσία `/dev/cpu/CPUNUM/msr`, όπου `CPUNUM` είναι ο αριθμός ενός πυρήνα.

Τέλος, το Linux για λόγους ασφάλειας, έχει απαγορεύσει την προσπέλαση των MSR καταχωρητών. Για να μπορεί να επιτραπεί στον χρήστη να προσπελάσει τους καταχωρητές αυτούς, θα πρέπει να εκτελεστεί η παρακάτω εντολή:

```
sudo bash -c echo on > /sys/module/msr/parameters/allow_writes
```

3.3 Εντοπισμός της χαμηλότερης λειτουργικής τάσης

Το κίνητρο με το οποίο γίνεται η διαδικασία της μείωσης της τάσης του επεξεργαστή είναι η εξοικονόμηση ενέργειας. Οι κατασκευαστές των επεξεργαστών προκειμένου να μην έχουν απρόοπτα προβλήματα κατά την κατασκευαστική διαδικασία ενός επεξεργαστή ορίζουν ένα άνω φράγμα στην τάση του επεξεργαστή, που όμως είναι πιο μεγάλο από αυτό που όντως χρειάζεται ο επεξεργαστής (Σχήμα 3.4). Αυτό γίνεται, επειδή κάθε όμοιος επεξεργαστής που δημιουργείται μπορεί να έχει ελαφρώς διαφορετικές απαιτήσεις κατανάλωσης και επομένως, θέτοντας ως άνω φράγμα της τάσης το όριο στο οποίο έδειξε ένας από τους όμοιους επεξεργαστές, δεν σημαίνει πως όλοι τους θα έχουν το ίδιο όριο και έτσι, οι κατασκευαστές των επεξεργαστών για να είναι σίγουροι, θέτουν ένα μεγαλύτερο όριο που θα ισχύει σίγουρα για όλους τους όμοιους επεξεργαστές.



Σχήμα 3.4: *Η τάση του επεξεργαστή*

Αφού σκοπός είναι η μέγιστη εξοικονόμηση ενέργειας, χωρίς να υπάρξει πτώση στις επιδόσεις, αυτό που χρειάζεται είναι να βρεθεί μια χαμηλή τάση στην οποία λειτουργεί σωστά

ο επεξεργαστής και αν μειώσουμε οσοδήποτε περισσότερο την τάση, ο επεξεργαστής να μην μπορεί να λειτουργήσει πλέον σωστά (Σχήμα 3.4). Για γίνει σωστά αυτή η διαδικασία, θα πρέπει να μειώνεται σταδιακά η τάση του επεξεργαστή και σε κάθε στάδιο να ελέγχεται με κάποιο τρόπο η σωστή λειτουργικότητα του εν λόγω επεξεργαστή. Ο καλύτερος τρόπος με τον οποίο μπορεί να γίνει ο έλεγχος αυτός είναι χρησιμοποιώντας κάποια απαιτητικά benchmarks , τα οποία θα φτάνουν τον επεξεργαστή στα όρια του. Ένας ενδεικτικός αλγόριθμος που θα έκανε αυτή την διαδικασία φαίνεται στον αλγόριθμο 3.2.

μ 3.2: *Find the minimum safe voltage*

currVoltage ← *nominal* ▷ Nominal Voltage
while cpu ≠ *break*(*currVoltage*) **do**
    oparate ← *testIftheCpuOparatesRight*()
    if *oparete* ≠ *True* **then**
        break
    end if
    currVoltage ← *nextStege*(*currVoltage*)
end while

Κεφάλαιο 4

Symphony

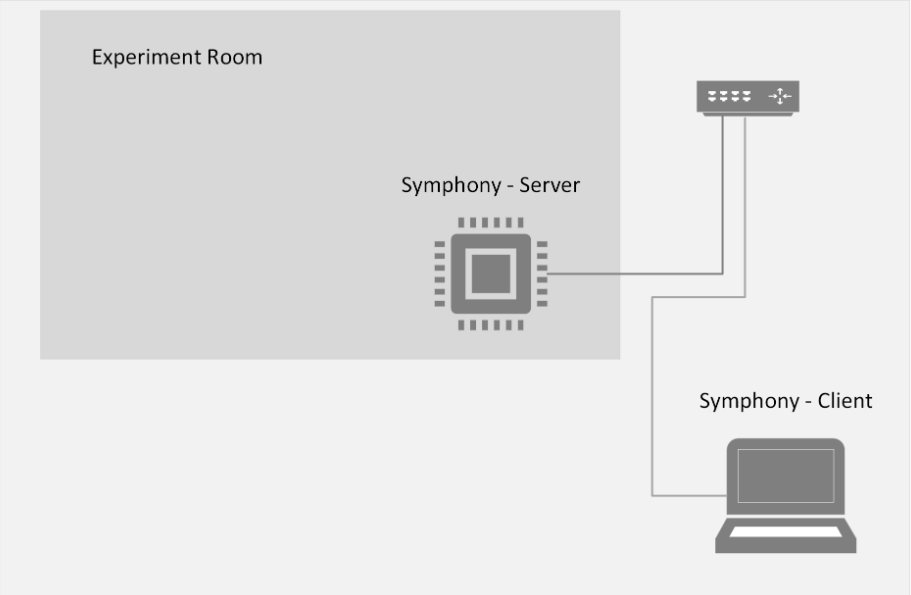
Σε ένα πείραμα στο οποίο ένα ή περισσότερα ηλεκτρονικά συστήματα εκτίθενται σε ακτινοβολία νετρονίων, χρειάζεται ένας τρόπος με τον οποίο να γίνεται τόσο η παρακολούθηση όσο και η διαχείριση των συστημάτων αυτών, σε περίπτωση που κάτι πάει στραβά κατά την ώρα της εκτέλεσης του πειράματος, π.χ στην περίπτωση που το σύστημα σταματήσει να λειτουργεί λόγο ενός **Detectable Unrecoverable Error (DUE)** σφάλματος ή στη περίπτωση που καεί κάποιο υποσύστημα. Το **Symphony** είναι ένα εργαλείο σχεδιασμένο τόσο για την παρακολούθηση όσο και για τον έλεγχο των ηλεκτρονικών συστημάτων που δέχονται υψηλή ιοντίζουσα ακτινοβολία κατά την διάρκεια του πειράματος. Σε αυτό το κεφάλαιο θα γίνει αναλυτική περιγραφή του πως επιτυγχάνεται ο έλεγχος και η διαχείριση των ηλεκτρονικών συστημάτων κατά την διάρκεια του πειράματος και θα γίνει αναφορά και στην αρχιτεκτονική του εν λόγω εργαλείου.

4.1 Αρχιτεκτονική

Σε αυτή την ενότητα θα γίνει αναλυτική περιγραφή της τοπολογίας, της γενικής λειτουργίας του Symphony κατά την διάρκεια του πειράματος και πως ανταπεξέρχεται σε συμβάντα που κάτι πάει στραβά.

4.1.1 Τοπολογία

Κατά την διάρκεια ενός πειράματος που σχετίζεται με την έκθεση ενός συστήματος σε υψηλή ακτινοβολία, χρειάζεται ένας τρόπος με τον οποίο να παρακολουθείτε το σύστημα, χωρίς όμως να χρειάζεται η ανθρώπινη παρέμβαση στον θάλαμο που γίνεται το πείραμα. Το Symphony είναι σχεδιασμένο με τέτοιον τρόπο, έτσι ώστε να επιτυγχάνεται η επικοινωνία του συστήματος, που βρίσκεται στον θάλαμο με υψηλή ακτινοβολία, με ένα άλλο σύστημα, το οποίο βρίσκεται εκτός του θαλάμου. Για να επιτευχθεί αυτή η επικοινωνία, γίνεται χρήση του μοντέλου πελάτη - εξυπηρέτη **(Client - server Model)**, θέτοντας ως πελάτη το σύστημα έξω από τον θάλαμο και ως εξυπηρέτη το σύστημα που βρίσκεται στον θάλαμο (Σχήμα 4.1).



Σχήμα 4.1: *Τοπολογία του Symphony*

Κατά την διάρκεια που το πείραμα βρίσκεται σε εξέλιξη, το Symphony στέλνει κάποιες εντολές προς εκτέλεση στον υπολογιστή που βρίσκεται μέσα στον θάλαμο του πειράματος και το σύστημα μέσα στον θάλαμο με την σειρά του στέλνει κάποιες πληροφορίες, που σχετίζονται με την εκτέλεση της εντολής, πίσω στο Symphony. Οι πληροφορίες που σχετίζονται με την εκτέλεση της εντολής είναι σημαντικές προκειμένου να φανούν τυχόν ανωμαλίες κατά την εκτέλεση των εντολών από το σύστημα εξυπηρέτη, όπως για παράδειγμα να γίνει κάποιο σφάλμα στην εντολή, που θα δικαιολογούνταν ότι προήλθε από ένα **SDC**.

4.1.2 Σφάλματα

Τα πειράματα για τα οποία έχει σχεδιαστεί το **Symphony**, μπορεί να εμφανίσουν δύο ειδών σφάλματα, τα οποία κατηγοριοποιούνται στα **DUE** και στα **SDCs**. Έτσι, θα πρέπει να υπάρχει κάποια μέθοδος, εντοπισμού και καταγραφής αυτών των σφαλμάτων, προκειμένου να είναι δυνατή η μετέπειτα ανάλυση της συμπεριφοράς του συστήματος **DUT**.

Τα δύο σφάλματα που μπορούν να προκληθούν, κατηγοριοποιούνται ως ξεχωριστά σφάλματα, αλλά όπως έχει περιγραφεί και σε προηγούμενη ενότητα, οφείλουν την εμφάνιση τους στο ίδιο φαινόμενο που παρατηρείται να εμφανίζεται στους κυρίως επεξεργαστές, αλλά και σε άλλα υποσυστήματα. Ο τρόπος όμως με τον οποίο τα σφάλματα αυτά επηρεάζουν την σωστή λειτουργία του συστήματος, είναι και αυτός που χρησιμοποιήθηκε για την κατηγοριοποίηση του κάθε ένα από τα δύο αυτά σφάλματα.

Σε περίπτωση που εμφανιστεί κάποιο σφάλμα στην κρυφή μνήμη του επεξεργαστή ή και σε κάποια εσωτερική μονάδα του, υπάρχουν δύο ενδεχόμενα. Το ένα είναι πως το υπό σύστημα εντοπισμού και διόρθωσης σφαλμάτων του επεξεργαστή θα εντοπίσει την λέξη που περιλαμβάνει το λάθος **bit** και έτσι, ο πυρήνας του λειτουργικού συστήματος θα ενημερωθεί κατάλληλα πως προκλήθηκε κάποιο σφάλμα σε κάποιο σημείο του επεξεργαστή. Το άλλο ενδεχόμενο είναι πως, δεν θα γίνει καμιά ενημέρωση, παρόλο που προκλήθηκε σφάλμα, λόγο της υψηλής ακτινοβολίας. Το τελευταίο ενδεχόμενο κατηγοριοποιείται ως **SDC**.

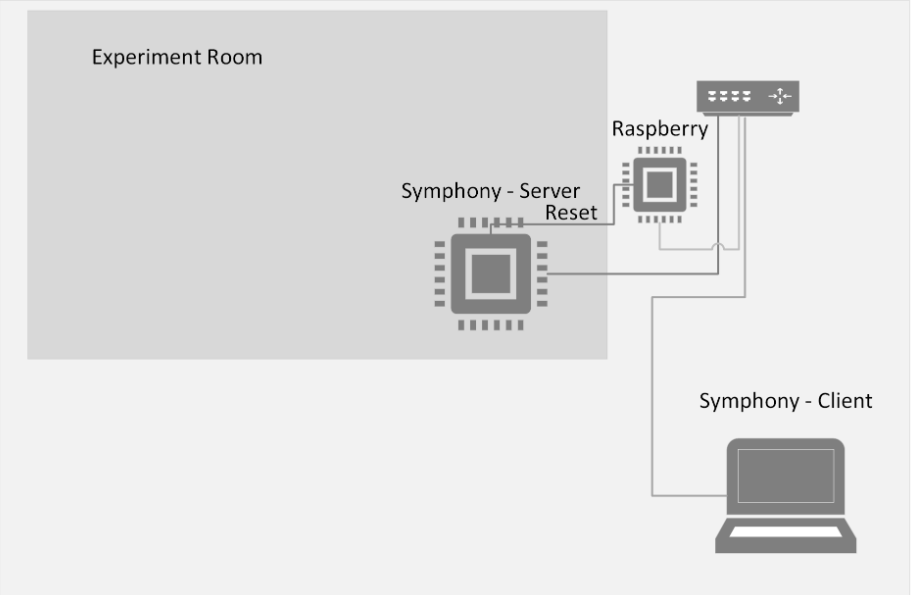
Παρόμοια με τα SDCs, τα DUE, προκαλούνται από σφάλματα στην κρυφή μνήμη του επεξεργαστή. Αλλά σε αντίθεση με τα SDCs, τα DUE προκαλούν τέτοια βλάβη στο λειτουργικό σύστημα, που ο μόνος τρόπος να επαναφερθεί το σύστημα στην σωστή του λειτουργιά είναι να γίνει επαναφορά ολόκληρου του συστήματος.

4.1.3 Διαχείριση DUE

Το σύστημα που βρίσκεται εντός του θαλάμου του πειράματος είναι πιθανό ανά διαστήματα να υπολειτουργεί ή ακόμα και να σταματήσει να λειτουργεί ολοκληρωτικά, κάτι που θα σήμαινε ότι προκλήθηκε από κάποιο **DUE**. Σε αυτές τις περιπτώσεις, υπό την προϋπόθεση ότι το σύστημα σταμάτησε να λειτουργεί λόγω προβλήματος του λογισμικού του, θα πρέπει το πείραμα να επαναφερθεί στην κατάσταση λειτουργίας, χωρίς την ανθρώπινη παρέμβαση.

Στο **Symphony** ο εντοπισμός της μη λειτουργικότητας του συστήματος εξυπηρέτη ισοδυναμεί με το ότι δεν στέλνονται πακέτα, μέσο του υποκείμενου δικτύου, από το σύστημα πελάτη στον εξυπηρέτη για κάποιο ορισμένο χρονικό διάστημα. Αφότου το Symphony αντιληφθεί ότι το σύστημα εξυπηρέτη έχει σταματήσει να λειτουργεί, θα πρέπει να πάρει τα κατάλληλα μέτρα, προκειμένου να το επαναφέρει το συντομότερο δυνατόν.

Σχετικά με την επαναφορά του συστήματος εξυπηρέτη, θα πρέπει να γίνει συνεργασία μεταξύ ενός τρίτου υπολογιστικού συστήματος και του Symphony. Το υπολογιστικό σύστημα αυτό είναι ένας μικρο ελεγκτής (**Raspberry PI**), ο οποίος έχει έλεγχο τόσο του φυσικού κουμπιού επαναφοράς, όσο και του κουμπιού ενεργοποίησης του συστήματος εξυπηρέτη (4.2). Έτσι, στην περίπτωση που πρέπει ο εξυπηρετητής να επανεκκινηθεί, το Symphony θα στείλει ένα αίτημα στον μικρό ελεγκτή και με την σειρά του αυτός θα πραγματοποιήσει μια υποχρεωτική επανεκκίνηση στο σύστημα εξυπηρέτη. Στο ενδεχόμενο που ο εξυπηρετητής δεν ανταποκριθεί στην αίτηση του Symphony, τότε, ο εξυπηρετητής ίσος έχει κάποιο καμένο υποσύστημα και σε αυτή την περίπτωση θα πρέπει να γίνει αναγκαστική παύση του πειράματος μέχρις ότου γίνει αλλαγή του υποσυστήματος που κάηκε.



Σχήμα 4.2: *Τοπολογία του Symphony με το Raspberry PI*

4.1.4 Προγράμματα αξιολόγησης

Όπως αναφέρθηκε και σε προηγούμενη ενότητα, το Symphony στέλνει στο σύστημα εξυπηρέτη κάποιες εντολές προς εκτέλεση. Οι εντολές αυτές μπορούν να είναι δύο ειδών. Το πρώτο είδος είναι εντολές που αλλάζουν την τάση του πελάτη (κεφάλαιο 3). Το δεύτερο είδος είναι εντολές αξιολόγησης, που χρησιμοποιούνται για να ελέγξουν την ανθεκτικότητα του συστήματος πελάτη σε θέματα που κυρίως αφορούν τον επεξεργαστή και την μνήμη τυχαίας προσπέλασης (RAM) του.

Τα προγράμματα αξιολόγησης που χρησιμοποιήθηκαν στο συγκεκριμένο πείραμά είναι μια ομάδα από προγράμματα τα οποία έχουν σχεδιαστεί και εξακολουθούν να συντηρούνται από την NASA με σκοπό την αξιολόγηση της απόδοσης υπέρ υπολογιστών (supercomputers). Κύριος σκοπός τους είναι να αξιολογήσουν την απόδοση ενός συστήματος σε ενέργειες που είναι εξαρτώμενες από την παράλληλη υπολογιστή δύναμη. Η NASA αποκαλεί την ομάδα των προγραμμάτων αυτών ως **NAS Parallel Benchmarks** ή **NPB**.

Τα προγράμματα από τα οποία αποτελείται αυτό το σύνολο προγραμμάτων αξιολόγησης αναφέρονται με τα ονόματα που φαίνονται στον πίνακα (4.1). Κάθε ένα από αυτά στοχεύουν στην αξιολόγηση μιας συγκεκριμένης μονάδας του υπό αξιολόγηση συστήματος. Ποιο συγκεκριμένα η αξιολόγηση γίνεται στον επεξεργαστή ή/και στην μνήμη τυχαίας προσπέλασης.

Προγράμματα αξιολόγησης (NPB)	
IS	EP
CG	MG
FT	BT
SP	LU

Πίνακας 4.1: *Προγράμματα αξιοολόγησης της NASA*

Για αυτά τα προγράμματα έχουν δημιουργηθεί δύο διαφορετικές εκδόσεις, οι οποίες διαχωρίζονται στην έκδοση που χρησιμοποιεί ή το μοντέλο σχεδίασης παράλληλων προγραμμάτων **MPI** ή το **OpenMP** υποβαθρο.

Στο πείραμα, χρησιμοποιήθηκαν τα επόμενα προγράμματα αξιολόγησης: **IS, CG, FT, LU** και **EP**. Το **Symphony** κατά την διάρκεια του πειράματος χαμηλώνει σταδιακά την τάση του επεξεργαστή του εξυπηρέτη και κάθε φορά εκτελεί για ένα ορισμένο χρονικό διάστημα ένα από τα 6 προγράμματα αξιολόγησης που αναφέρθηκαν παραπάνω. Έτσι, με αυτόν τον τρόπο γίνεται η αξιολόγηση του συστήματος που βρίσκεται στον θάλαμο του πειράματος με μια πληθώρα από δύσκολα υπολογιστικές πράξεις, που εκτελούνται για κάθε μια από της τάσεις στις οποίες το σύστημα αυτό είναι λειτουργικό. Σκοπός της εν λόγο διαδικασίας είναι να γίνει εντοπισμός των σφαλμάτων που προκαλεί η υψηλή ακτινοβολία στον επεξεργαστή του συστήματος στον θάλαμο. Η αναμενόμενη παρατήρηση είναι πως όσο το σύστημα στον θάλαμο του πειράματος έχει όλο και μικρότερη τάση, τότε θα παρατηρούνται όλο και ποιο συχνά σφάλματα στα αποτελέσματα που δίνουν τα προγράμματα αξιολόγησης.

4.1.5 Διαχείριση των SDCs

Στο πείραμα χρειάζεται να υπάρχει κάποιος τρόπος με τον οποίο να γίνεται ο εντοπισμός των **SDCs** που μπορεί να προκλήθηκαν κατά την διάρκεια της εκτέλεσης ενός προγράμματος αξιολόγησης. Εντοπίζοντας αυτά τα σφάλματα δημιουργείται μια εικόνα με την οποία μπορεί να φανεί η ροπή του επεξεργαστή ως προς το πόσα τέτοια σφάλματα γίνονται και το πως αυτό σχετίζεται με την μείωση της τάσης του. Έτσι, μπορεί να βγει το συμπέρασμα ότι αν σε χαμηλότερη τάση υπάρχουν περισσότερα τέτοια σφάλματα, τότε, ο επεξεργαστής είναι ποιο επιρρεπείς στην εμφάνιση τέτοιον σφαλμάτων, αφού σε αυτή την περίπτωση, το σύστημα θα δέχεται περισσότερα σφάλματα σε επίπεδο **bit** και έτσι κάποια από αυτά καταφέρνουν και ξεφεύγουν από τους μηχανισμούς εντοπισμού και διόρθωσης σφαλμάτων που έχει ενσωματωμένους στο εσωτερικό του ο επεξεργαστής.

Σε κάθε πρόγραμμα που περιγράφηκε στην υπό ενότητα (4.1.4), υπάρχει μια ένδειξη με την οποία η προγραμματιστές των προγραμμάτων αυτών δείχνουν στον χρήστη πως το τρέχων πρόγραμμα έκανε όλες τις πράξεις με επιτυχία και πως το τελικό αποτέλεσμα που υπολογίστηκε είναι το σωστό. Στην περίπτωση όμως που προκληθεί κάποιο **SDC** κατά την διάρκεια εκτέλεσης του εν λόγο προγράμματος αξιολόγησης, τότε, ενδέχεται να επηρεάσει το τελικό αποτέλεσμα και η ένδειξη να δείξει πως κάτι πήγε στραβά, αφού το τελικό αποτέλε-

σμα δεν συμφωνεί με την ορθή απάντηση. Έχοντας αυτή την πληροφορία, το Symphony, την χρησιμοποιεί έτσι ώστε να εντοπίζει τα **SDCs** που ίσος μπορεί να προκλήθηκαν κατά την εκτέλεση του εν λόγο προγράμματος αξιολόγησης. Χρησιμοποιώντας ύστερα αυτή την πληροφορία είναι δυνατών να παραχθεί το **SDCs rate**, με το οποίο μπορούν να βγουν τα σχετικά συμπεράσματα, για το πως ή τάση επηρεάζει την αξιοπιστία του συστήματος.

4.2 Αλγόριθμος

Σε αυτό το κεφάλαιο θα γίνει περιγραφή των θεωρητικών θεμάτων που αναφέρθηκαν στην προηγούμενη υπό ενότητα με ποιο τεχνικό τρόπο, από την οπτική γωνία του λογισμικού.

Το **Symphony** ως αλγόριθμος έχει γραφεί στην γλώσσα **Python**, έκδοση 3.X. Ο λόγος χρήσης αυτής της γλώσσας είναι πως αποτελεί μια γλώσσα η οποία είναι πολύ εμπλουτισμένη με πολύ ισχυρές βιβλιοθήκες, που διευκόλυναν την υλοποίηση αυτού του λογισμικού. Παρόλο που έχει γραφεί σε αυτή τη γλώσσα, σε αυτό το κεφάλαιο δεν θα δοθεί κάπου κώδικας **Python3**, αλλά συνοπτικά γραμμένος κώδικας σε ψευδό γλώσσα, όπου χρειάζεται.

Όπως αναφέρθηκε στην προηγούμενη υπό ενότητα, το **Symphony** χρησιμοποιεί το μοντέλο πελάτη - εξυπηρέτη, προκειμένου να εγκαθιδρύσει την σύνδεση μεταξύ του συστήματος που βρίσκεται μέσα στον θάλαμο του πειράματος, που αντιστοιχεί στον εξυπηρέτη, και του συστήματος ελέγχου, έξω από τον θάλαμο, που αντιστοιχεί στον πελάτη. Το σύστημα που βρίσκεται εξωτερικά, έλαβε την ονομασία Host, ενώ το σύστημα που βρίσκεται μέσα στον θάλαμο έλαβε την ονομασία **Device under test** ή **DUT**. Στις επόμενες υπό ενότητες θα γίνει αναλυτική περιγραφή στο πως γίνονται οι βασικές λειτουργίες που επιτυγχάνει το **Symphony**.

4.2.1 DUT

Το **DUT** αποτελεί το σύστημα που βρίσκεται εντός του θαλάμου του πειράματος. Ως λογισμικό έχει υλοποιημένη την μεριά του εξυπηρέτη της αρχιτεκτονικής πελάτη - εξυπηρέτη. Καθόλη την διάρκεια του πειράματος, το **DUT** αναμένει μια αίτηση για σύνδεση από το σύστημα ελέγχου **Host**. Όταν εγκαθιδρυθεί η σύνδεση μεταξύ του **Host** και του **DUT**, από εκεί και ύστερα ξεκινάνε να γίνονται διάφορα αιτήματα από τον **Host** προς το **DUT**, που θα αναλυθούν λεπτομερώς στις παρακάτω υπό ενότητες.

Λήψη αίτησης για εκτέλεση εντολής

Ο σκοπός του **DUT** είναι να λαμβάνει εντολές προς εκτέλεση από το σύστημα ελέγχου **Host**, να εκτελεί την εντολή και ύστερα να απαντάει στον **Host** με τις πληροφορίες που σχετίζονται με την εντολή αυτή. Ο αλγοριθμικός τρόπος με τον οποίο μπορεί να επιτευχθεί αυτή η λειτουργία μπορεί να φανεί στον ψευδοκώδικα (4.1).

μ 4.1: *Recieve and execute command from Host*

procedure RECEIVEANDEXECUTECOMMANDFROMHOST
waitForHostToConnect()
command ← *recieveCommandFromHost*()
startMonitoring()
commandOutput ← *executeCommand*(*command*)
monitorData ← *stopMonitoring*()
sendToHost(*MonitorData*)
sendToHost(*CommandOutput*)

Βάση του αλγορίθμου αυτού, το **DUT** λαβμάνει την εντολή προς εκτέλεση από το σύστημα **Host**. Το επόμενο βήμα είναι ή ενεργοποιήσει, **σε παράλληλη εκτέλεση με την εντολή**, μιας ρουτίνας παρακολούθησης, που αντιστοιχεί στην κλήση της ρουτίνας **startMonitoring()** στον ψευδό κώδικα. Η ρουτίνα αυτή χρησιμοποιείται προκειμένου να καταγραφούν κάποιες βασικές πληροφορίες (4.2.1) όσο εκτελείται μια εντολή. Τέλος, μετά το πέρας της εκτέλεσης της εντολής που ζητήθηκε, θα πρέπει να σταματήσει η ρουτίνα παρακολούθησης και να σταλθούν στο σύστημα **Host** οι πληροφορίες που καταγράφηκαν από την ρουτίνα παρακολούθησης, μαζί με την έξοδο που παράχθηκε από την εντολή που ζητήθηκε από τον **Host**.

Περιεχόμενα του μνήματος απάντησης

Όπως αναφέρθηκε και στην προηγούμενη υπό ενότητα, κατά την εκτέλεση μιας εντολής, η οποία στάλθηκε ως αίτημα για εκτέλεση από το σύστημα Host, θα πρέπει να εκτελείται παράλληλα μια ρουτίνα παρακολούθησης. Αυτή η ρουτίνα καταγράφει περιοδικά ένα πλήθος από τιμές, οι οποίες αντιστοιχούν στις παρακάτω πληροφορίες:

- Τα μηνύματα που παράχθηκαν από το σύστημα Linux και που βρίσκονται στο Dmesg.
- Την θερμοκρασία του επεξεργαστή.
- Την συχνότητα του επεξεργαστή.
- Την κατανάλωση του επεξεργαστή
- Την τάση του επεξεργαστή

Οι πληροφορίες αυτές χρησιμοποιούνται ιδιαίτερα στην διαδικασία ανάλυσης των τελικών αποτελεσμάτων, για να βγουν συμπεράσματα, που έχουν να κάνουν με την συμπεριφορά του επεξεργαστή κατά την διάρκεια του πειράματος. Ποιο συγκεκριμένα, οι σημαντικότερες πληροφορίες που χρησιμοποιούνται για το εν λόγο πείραμα, είναι η κατανάλωση και τα μηνύματα που παράγονται από τον πυρήνα του Linux. Η κατανάλωση χρησιμοποιείται έτσι ώστε να διακριθεί η συμπεριφορά του επεξεργαστή όσο σταδιακά μειώνεται η τάση του. Τα μηνύματα του πυρήνα του Linux, χρησιμοποιούνται, έτσι ώστε να γίνει η ταυτοποίηση των SDCs. Αυτό συμβαίνει, διότι, κατά την διάρκεια του πειράματος, ο μόνος τρόπος με

τον οποίο γίνεται δυνατός ο εντοπισμός των SDCs είναι να εντοπιστεί κάποιο υπολογιστικό σφάλμα κατά την εκτέλεση ενός προγράμματος αξιολόγησης (4.1.5). Όμως, αν στην περίπτωση αυτή, ο πυρήνας του Linux στείλει στο Dmesg μήνυμα από το σύστημα εντοπισμού και διόρθωσης σφαλμάτων του επεξεργαστή, τότε, μπορεί να μην διορθώθηκε, αλλά το υπό σύστημα του επεξεργαστή κατάφερε και εντόπισε το σφάλμα το οποίο προκλήθηκε και επομένως, συνεπάγεται πως τούτο το σφάλμα δεν μπορεί να κατηγοριοποιηθεί ως SDC. Σε αντίθετη περίπτωση, που συμβεί σφάλμα στο πρόγραμμα αξιολόγησης, αλλά δεν υπάρξει καμιά ενημέρωση από το υπό σύστημα του επεξεργαστή, μόνο σε αυτή την περίπτωση μπορεί να είναι σίγουρο ότι το σφάλμα αυτό αποτελεί SDC.

4.2.2 Host

Ο **Host** αποτελεί το σύστημα που βρίσκεται εκτός του θαλάμου του πειράματος. Ως λογισμικό έχει υλοποιημένη την μεριά του πελάτη της αρχιτεκτονικής πελάτη - εξυπηρέτη. Καθόλη την διάρκεια του πειράματος, ο **Host** κάνει διαρκώς προσπάθεια για να εγκαθιδρύσει μια σύνδεση προς το σύστημα DUT. Όταν η σύνδεση με το σύστημα μέσα στον θάλαμο του πειράματος εγκαθιδρυθεί με επιτυχία, από εκείνη την στιγμή και ύστερα, το σύστημα Host στέλνει εντολές, είτε για τον προσδιορισμό της τάσης του DUT, είτε για την εκτέλεση ενός προγράμματος αξιολόγησης. Μετά την εκτέλεση ενός προγράμματος αξιολόγησης, το σύστημα Host αναμένει την απάντηση από το σύστημα DUT. Σε αυτή την υπό ενότητα, θα γίνει αναλυτική περιγραφή όλων των λειτουργιών που εκτελούνται από την πλευρά του Host.

Επαναφορά λειτουργίας μετά από DUE

Μετά τον εντοπισμό ενός σφάλματος τύπου DUE, θα πρέπει να γίνουν οι κατάλληλες ενέργειες, έτσι ώστε το σύστημα που βρίσκεται εντός του θαλάμου να ξανά επαναφερθεί σε κατάσταση λειτουργίας. Αυτό επιτυγχάνεται, χρησιμοποιώντας έναν μικρο-ελεγκτή, Raspberry PI, ο οποίος έχει τον έλεγχο των φυσικών κουπιών επαναφοράς/τερματισμού λειτουργίας του DUT. Έτσι, μετά τον εντοπισμό ενός DUT, ο Host στέλνει ένα αίτημα μέσω του υποκείμενου δικτύου προς τον μικρό-ελεγκτή Raspberry Pi και αυτός με την σειρά του αναγκάζει το σύστημα DUT να κάνει υποχρεωτική επαναφορά συστήματος. Ο ψευδοκώδικας που κάνει αυτή την λειτουργία, μπορεί να φανεί παρακάτω (4.2).

μ 4.2: *Symphony detect a DUE error*

procedure DUEDETECTED
connectToRaspberryPi()
requestForRestart()
waitForDut()
continueExperiment()

Όπως φαίνεται και στον αλγόριθμο (4.2), αρχικά, ο Host εγκαθιδρύει μια σύνδεση προς τον μικρό-ελεγκτή, με την συνάρτηση **connectToRaspberryPi**. Ύστερα, στέλνει ένα αίτημα προς τον μικρό-ελεγκτή, έτσι ώστε να ζητήσει την υποχρεωτική επανεκκίνησή του συστήματος DUT, χρησιμοποιώντας την συνάρτηση **requestForRestart** και τέλος, περιμένει για το σύστημα DUT να ξανά μπει σε κατάσταση λειτουργίας. Ο χρόνος που αναμένει ο Host για την επαναφορά του συστήματος DUT είναι προ καθορισμένος, και αντιστοιχεί στον καταγεγραμμένο χρόνο που κάνει το σύστημα αυτό να επαναφερθεί. Όταν ο χρόνος αυτός περάσει, ο Host περιμένει πως το σύστημα DUT έχει πλέον επαναφερθεί σε κατάσταση λειτουργίας και προσπαθεί να συνεχίσει την εκτέλεση του πειράματος, από εκεί που το είχε αφήσει. Σε περίπτωση που το σύστημα DUT δεν καταφέρει να ξανά μπει σε κατάσταση λειτουργίας, τότε, ο Host θα ξανά προσπαθήσει να ξανά κάνει μια υποχρεωτική επαναφορά. Σε περίπτωση που το σύστημα όμως συνεχίζει και δεν ανταποκρίνεται στο αίτημα για υποχρεωτική επαναφορά, τότε, ενδέχεται να έχει προκληθεί μόνιμη βλάβη σε κάποιο υπό σύστημα του DUT, λόγο της υψηλής ακτινοβολίας και θα πρέπει να αντικατασταθεί.

Αίτηση για εκτέλεση εντολής

Ο Host κατά την εκτέλεση του πειράματος, στέλνει στο σύστημα DUT εντολές προς εκτέλεση, που μπορεί να είναι είτε για τον προσδιορισμό της τάσης του είτε για την εκτέλεση κάποιου από τα διαθέσιμα προγράμματα αξιολόγησης. Για να γίνει αυτή η διαδικασία, αρχικά, ο Host προσπαθεί να εγκαθιδρύσει μια νέα σύνδεση με το σύστημα DUT. Ύστερα στέλνει στο σύστημα εντός του θαλάμου την εντολή προς εκτέλεση και αναμένει μια απάντηση. Στην περίπτωση που η εντολή είναι για τον προσδιορισμό της τάσης, η απάντηση που αναμένετε είναι πως ή εντολή εκτελέστηκε σωστά, ενώ στην περίπτωση που η εντολή είναι ένα πρόγραμμα αξιολόγησης, τότε, ή απάντηση θα πρέπει να είναι αυτή που περιγράφηκε στην υπό ενότητα (4.2.1). Στην περίπτωση, που ο Host δεν καταφέρει για κάποιο λόγο να εγκαθιδρύσει μια νέα σύνδεση με το σύστημα DUT, τότε, αυτή ή προσπάθεια καταγράφεται σε έναν μετρητή, ο οποίος αν φτάσει στις τρεις προσπάθειες, τότε ο Host αντιλαμβάνεται πως το σύστημα εντός του θαλάμου ίσος να έχει δεχτεί κάποιο DUT και αυτή την στιγμή δεν βρίσκεται πλέον σε κατάσταση λειτουργίας. Έτσι, σε αυτή την περίπτωση γίνεται ή διαδικασία που περιγράφηκε στην υπό ενότητα (4.2.2), ώστε να επαναφερθεί το σύστημα αυτό στην σωστή λειτουργία ξανά. Η παραπάνω διαδικασία, μπορεί να φανεί αλγοριθμικά, στον παρακάτω ψευδοκώδικα.

μ 4.3: *Execute command on DUT*

```
procedure executeCommand(cmd, configureVoltage)
    connectToDut()
    output = requestDutExecution(cmd)
    if configureVoltage ≠ True then
        return output
    else
        success = commandSuccess(output)
        if success ≠ True then
            sendWarning()
            executeCommand(cmd, configureVoltage)
        end if
    end if
```

Όπως φαίνεται στον αλγόριθμο (4.3), αρχικά ο Host εγκαθιδρύει μια σύνδεση με το σύστημα DUT, με την συνάρτηση **connectToDut**. Ύστερα, ζητάει από το σύστημα DUT να εκτελέσει την ζητούμενη εντολή και αποθηκεύει τα αποτελέσματα που ήρθαν από το σύστημα DUT στην μεταβλητή **output**. Μετά την εκτέλεση της εντολής, ελέγχεται αν η εντολή αυτή ήταν για τον προσδιορισμό της τάσης ή ήταν ένα πρόγραμμα αξιολόγησης. Αν ήταν για τον προσδιορισμό της τάσης, τότε, επιστρέφεται η πληροφορίες που αναλύονται στην υπό ενότητα (4.2.1), αλλιώς, ελέγχεται αν η εντολή εκτελέστηκε με επιτυχία, χρησιμοποιώντας την συνάρτηση **commandSuccess**. Σε περίπτωση που η εντολή που καθορίζει την τάση του συστήματος DUT δεν εκτελεστεί με επιτυχία, τότε, στέλνεται ένα προειδοποιητικό μήνυμα και ύστερα γίνεται ξανά άλλη μια προσπάθεια για την εκτέλεση της.

Αποθήκευση αποτελεσμάτων

Ο σκοπός του Host είναι να καταφέρει να ελέγχει το σύστημα **DUT**κατά την διάρκεια του πειράματος και επιπλέον, να αποθηκεύει οποιαδήποτε πληροφορία έρχεται από αυτό. Οι πληροφορίες αυτές χρησιμοποιούνται μετά το πέρας του πειράματος, έτσι ώστε να γίνει η κατάλληλη ανάλυση των αποτελεσμάτων και να βγουν τα αντίστοιχα συμπεράσματα. Για την καταγραφή των πληροφοριών αυτών, ο Host δημιουργεί κάποια αρχεία, χρησιμοποιώντας την μορφοποίηση Json, για διευκόλυνση στην μετέπειτα ανάλυση. Στα αρχεία αυτά, δύνεται όνομα βάση την ημερομηνία, ώρα, τα λεπτά και δευτερόλεπτα που λήφθηκαν τα δεδομένα από το σύστημα DUT και βάση του ποιου προγράμματος αξιολόγησης εκτέλεσε και σε ποια τάση βρισκόταν όταν εκτελέστηκε αυτό.

Κεφάλαιο 5

Εργαλεία ανάλυσης

Μετά το πέρας του πειράματος, θα πρέπει να γίνουν οι κατάλληλες τεκμηριώσεις και να βγουν τα συμπεράσματα, βάση των δεδομένων που μάζεψε το σύστημα **Host** κατά την διάρκεια του πειράματος. Για τον σκοπό αυτόν, σχεδιάστηκαν μερικά εργαλεία ανάλυσης, προκειμένου τα δεδομένα να ταξινομηθούν και να επεξεργαστούν αναλόγος.

Πέραν των βασικών εργαλείων ανάλυσης, έγινε εκτεταμένη χρήση μιας βάσης δεδομένων που σχεδιάστηκε για την αποθήκευση των δεδομένων του πειράματος και είχε ως σκοπό την απλοποίηση της μετέπειτα διαδικασίας ανάλυσης και τεκμηρίωσης των δεδομένων.

Σε αυτό το κεφάλαιο θα γίνει αναλυτική περιγραφή το πως τα εργαλεία ανάλυσης διαχωρίζουν και ταξινομούν τα αποτελέσματα του πειράματος και το πως τοποθετούνται στην βάση δεδομένων. Επιπλέον θα αναφερθεί το πως γίνεται η αναζήτηση των πληροφοριών στην βάση δεδομένων, προκειμένου να βγουν τα συμπεράσματα για το πείραμα.

5.1 Ανάλυση των Cache upset

Τα **Cache upsets** αποτελούν ένα φαινόμενο το οποίο παρατηρείται στην κρυφή μνήμη **Cache** των επεξεργαστών. Είναι ένα φαινόμενο που κάνει κάποια μεμονωμένα bits που βρίσκονται στην Cache να αλλάζουν από **μηδέν** σε **ένα** ή από **ένα** σε **μηδέν**. Ένας από τους παράγοντες εμφάνισης αυτού το φαινομένου είναι η έκθεση του επεξεργαστή σε υψηλή ακτινοβολία ή ακόμα και ή περίπτωση που ένα υπό σωματίδιο, προερχόμενο από πηγές κοσμικής ακτινοβολίας, διαπεράσει τον επεξεργαστή, καταλήγοντας να προκαλέσει έτσι ένα Cache upset.

5.1.1 Εντοπισμός

Το σύστημα **DUT**, που βρίσκεται στο εσωτερικό του θαλάμου του πειράματος, όπως έχει περιγραφεί και σε προηγούμενο κεφάλαιο τρέχει με την χρήση του λειτουργικού συστήματος **Linux**. Το Linux μπορεί και αντιλαμβάνεται αν συμβεί κάποιο Cache upset, λαμβάνοντας πληροφορίες από το υπό σύστημα εντοπισμού και διαχείρισης σφαλμάτων που βρίσκεται εντός του επεξεργαστή. Ύστερα από τον εντοπισμό ενός Cache upset, στο λειτουργικό σύστημα θα φανεί ως ένα σφάλμα ισοτιμίας (**parity error**) και έτσι, το Linux θα γράψει στο αρχείο με όνομα Dmesg τις πληροφορίες σχετικά με το σφάλμα ισοτιμίας που προέκυψε. Μερικές από τις πληροφορίες που δίνει σε αυτό το μήνυμα είναι η μνήμη Cache στην οποία προέκυψε το εν λόγο σφάλμα, αν διορθώθηκε και αν προέκυψε κατά την διαδικασία ανάγνωσης ή

εγγραφής της.

Το υπό σύστημα εντοπισμού και διόρθωσης σφαλμάτων που βρίσκεται μέσα στον επεξεργαστή αποθηκεύει τις πληροφορίες περί του σφάλματος σε κάποιους καταχωρητές που φέρνουν το όνομα **MC** καταχωρητές (ενότητα 2.1). Η πληροφορίες που το λειτουργικό σύστημα διαβάζει από τους καταχωρητές **MC** είναι κωδικοποιημένες σε έναν δεκαεξαδικό κωδικό μεγέθους 64 bit, ο οποίος περιλαμβάνει όλες τις απαραίτητες πληροφορίες για το σφάλμα που προκλήθηκε. Ένα παράδειγμα ενός τέτοιου δεκαεξαδικού αριθμού μπορεί να φανεί στον πίνακα (5.1).

Μήνημα ενημέρωσης σφάλματος	
Κωδικός	0xdc20400000010166

Πίνακας 5.1: *MC κωδικός αναγνώρισης σφάλματος*

5.1.2 Ανάλυση

Για την ανάλυση των **Cache upsets** που προκλήθηκαν κατά την διάρκεια του πειράματος, δημιουργήθηκε ένα λογισμικό, το οποίο λαμβάνει στην είσοδο του το περιεχόμενο που περιεχόταν στο αρχείο Dmesg κατά την διάρκεια του πειράματος, ξεχωρίζει και συλλέγει όλους τους κωδικοποιημένους δεκαεξαδικούς αριθμούς που περιγράφουν τα σφάλματα που εντοπίστηκαν από την μονάδα εντοπισμού και διόρθωσης σφαλμάτων του επεξεργαστή. Ένα παράδειγμα σε ψευδοκώδικα μπορεί να φανεί στον ψευδοκώδικα (5.1)

μ 5.1: *Decode the contents of an MC register*

```
procedure DecodeMC(Dmesg)
    listOfErrorCodes = parseErrorCodes(Dmesg)
    for e ← 1 to listOfErrorCodes.Size do
        DecodeErrorCode(listOfErrorCodes[e])
    end for
```

5.2 Διαχωρισμός των αποτελεσμάτων

Ένα πρόβλημα προς επίλυση που προκύπτει από πειράματα αυτής της φύσης, είναι πως η ακτίνα νετρονίων που χρησιμοποιείται για το πέρας του πειράματος ενδέχεται να μην είναι πλήρως λειτουργική για κάποια χρονικά διαστήματα κατά την διάρκεια που το σύστημα DUT βρίσκεται εντός του θαλάμου. Για τον λόγο αυτόν, μετά από ένα τέτοιο πείραμα θα πρέπει να γίνει διαχωρισμός των δεδομένων που παράχθηκαν από το σύστημα Host έτσι ώστε να κρατηθούν μόνο τα δεδομένα που παράχθηκαν στους χρόνους που η ακτίνα ήταν επαρκώς λειτουργική για τις ανάγκες του πειράματος, με σκοπό την καλύτερη αξιολόγηση των τελικών αποτελεσμάτων.

Για τον σκοπό αυτόν δημιουργήθηκε ένα εργαλείο το οποίο λαμβάνει στην είσοδο του τα δεδομένα που παράχθηκαν από το σύστημα Host και ένα αρχείο που περιλαμβάνει την

ένταση της ακτίνας σε μια δεδομένη χρονική στιγμή. Με τις πληροφορίες αυτές ως είσοδο, το εργαλείο αυτό προσπαθεί να ξεχωρίσει τα δεδομένα που παράχθηκαν κατά την περίοδο που η ακτίνα είχε χαμηλότερη ένταση και να κρατήσει τα δεδομένα που προέκυψαν κατά την περίοδο με την ιδανική ένταση. Ένα παράδειγμα σε ψευδοκώδικα μπορεί να φανεί στον ψευδοκώδικα (5.2)

μ 5.2: *Filter results, based on beam activity*

```
procedure FilterResults(beamLogs)
    listOfResults = parseResults()
    for r ← 1 to listOfResults.Size do
        if beamActive(listOfResults[r], beamLogs) ≠ True then
            removeResult(listOfResults[r])
        end if
    end for
```

5.3 Βάση δεδομένων

Η επεξεργασία και η διαδικασία απονομής συμπερασμάτων βάση των αποτελεσμάτων που παράγονται από ένα πείραμα αποτελεί μια διαδικασία στην οποία εμπλέκονται διάφορα προγράμματα που αναζητούν μέσα στα δεδομένα του πειράματος για κάποια ζητούμενα χαρακτηριστικά, όπως για παράδειγμα τα σφάλματα που προκλήθηκαν στην μνήμη **Cache** σε μια δεδομένη τάση. Αυτού του είδους τα προγράμματα αποτελούν μια χρονοβόρα διαδικασία και ενδεχομένως είναι και εστία προβλημάτων, στην περίπτωση που μείνουν απαρατήρητα λογικά λάθη κατά την υλοποίηση τους.

Για την ελαχιστοποίηση των σφαλμάτων και την επιτάχυνση της διαδικασίας ανάλυσης των αποτελεσμάτων, έγινε χρήση μιας βάσης δεδομένων, στην οποία, μετά το πέρας του πειράματος αποθηκεύτηκαν όλα τα αποτελέσματα που παράχθηκαν και ύστερα έγιναν τα απαραίτητα ερωτήματα, για την ανάκληση των πληροφοριών που ζητήθηκαν, βάση κάποιων χαρακτηριστικών, όπως περιγράφηκε και προηγουμένος.

Η βάση που χρησιμοποιήθηκε είναι η **Postgresql**, που αποτελεί μια σχεσιακή βάση δεδομένων. Σε αυτή την ενότητα θα γίνει αναλυτική περιγραφή της βάσης δεδομένων που χρησιμοποιήθηκε για την αποθήκευση των αποτελεσμάτων.

5.3.1 Σχήμα της βάσης

Η βάση δεδομένων που χρησιμοποιήθηκε αποτελείται από πέντε βασικούς πίνακες, οι οποίοι έχουν ως στόχο την αποθήκευση των δεδομένων του πειράματος, με τέτοιο τρόπο, έτσι ώστε να μην υπάρχουν διπλότυπα δεδομένα, για την εξοικονόμηση τόσο του χώρου, όσο και για την προστασία της βάσης από θέματα που αφορούν την ακεραιότητα των δεδομένων. Οι πέντε πίνακες από τους οποίους αποτελείται η βάση δεδομένων μπορούν να φανούν παρακάτω.

- **Result**. Αποτελεί τον πίνακα που αποθηκεύονται όλες οι πληροφορίες για ένα δεδομένο αποτέλεσμα, όπως τα περιεχόμενα του **Dmesg**, ο αριθμός των νετρονίων, η θερμοκρασία κ.λ.π.
- **Batch**. Αποτελεί τον πίνακα που περιέχει κοινές πληροφορίες για ένα σύνολο από αποτελέσματα.
- **Benchmark**. Αποτελεί την λίστα από τα προγράμματα αξιολόγησης που χρησιμοποιήθηκαν στο σύστημα **DUT**.
- **Voltage**. Αποτελεί την λίστα από τις τάσης που δοκιμάστηκαν κατά πάνω στο σύστημα **DUT**.
- **Frequency**. Αποτελεί την λίστα από συχνώτητες που δοκιμάστηκαν πάνω στο σύστημα **DUT**

Επιπλέον των παραπάνω πινάκων, η βάση δεδομένων περιλαμβάνει επίσης τέσσερις επιπλέον πίνακες, οι οποίοι παίζουν το ρόλο του να συσχετίζουν τους παραπάνω, όπως για παράδειγμα να συσχετιστούν τα αποτελέσματα τα οποία έτρεχαν σε μια δεδομένη τάση, με την αντίστοιχη τάση που βρίσκεται στον πίνακα **Voltage**. Παρακάτω φαίνονται οι τέσσερις αυτοί πίνακες.

- **Bench_member**. Αποτελεί τον πίνακα που συσχετίζει τα δεδομένα ενός **Batch** με ένα συγκεκριμένο Benchmark (το Batch αποτελεί σύνολο από αποτελέσματα).
- **Batch_member**. Αποτελεί τον πίνακα που συσχετίζει τα δεδομένα ενός ή πολλών αποτελεσμάτων με ένα συγκεκριμένο **Batch**.
- **Voltage_member**. Αποτελεί τον πίνακα που συσχετίζει τα δεδομένα ενός **Batch** με μια συγκεκριμένη τάση.
- **Frequency_member**. Αποτελεί τον πίνακα που συσχετίζει τα δεδομένα ενός **Batch** με μια συγκεκριμένη συχνώτητα.

Τέλος, το σχήμα της βάσης περιέχει επιπλέον ένα πλήθος από προβολές, με σκοπό την απλοποίηση των ερωτημάτων προς την βάση, καθώς ο αριθμός τον πεδίων ανά πίνακα είναι μεγάλος και ενδεχομένως ερωτήματα να γίνονταν αρκετά περίπλοκα, αν δεν υπήρχαν αυτές οι προβολές.

5.3.2 Τελική ανάλυση των SDCs

Όπως περιγράφτηκε στο κεφάλαιο (4.1.5) κατά την διάρκεια του πειράματος είναι δυνατό ο εντοπισμός των πιθανών **SDCs** χρησιμοποιώντας μια ένδειξη που παράγεται από τα προγράμματα αξιολόγησης τα οποία εκτελούνται στο σύστημα **DUT**. Ο τρόπος όμως αυτός δεν εγγυάται πως το σφάλμα που προκλήθηκε είναι απαραίτητα SDC, καθώς τα σφάλματα αυτά χαρακτηρίζονται ως σφάλματα που προκλήθηκαν χωρίς τον εντοπισμό τους από το σύστημα εντοπισμού και διόρθωσης σφαλμάτων του επεξεργαστή. Προκειμένου να επιβεβαιώσουμε ποια από τα σφάλματα αυτά είναι πράγματι **SDCs** και ποια όχι, χρειάζεται επιπλέον

η πληροφορία για το αν την χρονική στιγμή που πυροδοτήθηκε αυτό το σφάλμα ανήλθε και ενημέρωση από τον πυρήνα του λειτουργικού συστήματος. Στην περίπτωση που τέτοια ενημέρωση είναι παρούσα, τότε, το σφάλμα δεν μπορεί να κατηγοριοποιηθεί ως **SDC**, αφού το σύστημα γνώριζε πως προέκυψε. Από την άλλη, αν την χρονική στιγμή που πυροδοτήθηκε το σφάλμα αυτό δεν υπήρξε κάποια ενημέρωση, τότε μπορεί με υψηλή βεβαιότητά να κατηγοριοποιηθεί ως ένα **SDC**.

Με την χρήση της βάσης δεδομένων που χρησιμοποιήθηκε, η παραπάνω διαδικασία περιορίζεται σε ένα μόνο ερώτημα προς την βάση δεδομένων, το οποίο μετράει όλες τις εγγραφές της βάσης στις οποίες προκλήθηκε κάποιο σφάλμα κατά την εκτέλεση του προγράμματος αξιολόγησης και ταυτόχρονα δεν υπήρξε κάποιο μήνυμα από τον πυρήνα που να ενημερώνει για τον εντοπισμό σφάλματος κατά την χρονική στιγμή που πυροδοτήθηκε και το εν λόγω σφάλμα.

Κεφάλαιο 6

Μελέτη αποτελεσμάτων

Στο κεφάλαιο αυτό θα γίνει αναλυτική παρουσίαση των αποτελεσμάτων που παράχθηκαν από τα πειράματα που διεξάχθηκαν. Τα αποτελέσματα που παράχθηκαν προέρχονται από δύο ξεχωριστά συστήματα που έτρεχαν ταυτόχρονα κατά την διάρκεια του πειράματος. Για λόγους σωστής σύγκρισης, στα δύο συστήματα είχαν τοποθετηθεί οι ίδιοι ακριβώς επεξεργαστές, οι οποίοι αποτελούν των **Ryzen 5 2400G Pro**. Στις παρακάτω ενότητες θα γίνει παρουσίαση των αποτελεσμάτων σε θέματα όπως η κατανάλωση ενέργειας των δύο επεξεργαστών κατά σε μικρότερες λειτουργικές τάσης και σε θέματα αξιοπιστίας και το πως αυτά μπορεί διαφοροποιούνται ακόμα και σε επεξεργαστές που θεωρούνται πανομοιότυποι.

6.1 Οι επεξεργαστές

Ο επεξεργαστής που χρησιμοποιήθηκε και στα δύο πειράματα που τοποθετήθηκαν στον θάλαμο της ακτίνας ήταν ο **Ryzen 5 2400G Pro**. Για την πραγματοποίηση του πειράματος, αρχικά έπρεπε να βρεθεί ή χαμηλότερη λειτουργική τάση των δύο επεξεργαστών. Όπως και επαληθεύτηκε, οι χαμιλότερες λειτουργικές τάσης των δύο επεξεργαστών μπορούν να φανούν στον παρακάτω πίνακα.

Επεξεργαστής - Τάση (V)	
CPU1	1.11875 V
CPU2	1.10000 V

Πίνακας 6.1: *Χαμηλότερες λειτουργικές τάσεις των δύο **Ryzen 5 2400G Pro** επεξεργαστών*

Όπως φαίνεται και στον πίνακα (6.1), παρόλο που και τα δύο συστήματα έχουν στο εσωτερικό τους το ίδιο μοντέλο επεξεργαστή, παρατηρείτε πως η χαμηλότερη λειτουργική τους τάση διαφέρει ελαφρός, **κατά 18.75mV**.

6.2 Κατανάλωση

Η ενέργεια που καταναλώνεται από έναν επεξεργαστή έχει τραβήξει το ενδιαφέρον, καθώς η ραγδαία εξέλιξη των επεξεργαστών τα τελευταία χρόνια, όπως ο αυξημένος αριθμός των πυρήνων και ή πολυπλοκότερη αρχιτεκτονική των νέων επεξεργαστών, συνεπάγεται σε

περισσότερες απαιτήσεις ενέργειας προκειμένου ο επεξεργαστής να μπορεί να λειτουργεί ομαλά. Το πρόβλημα που δημιουργείται από την περισσότερη κατανάλωση ενέργειας είναι ιδιαίτερα έντονο σε **Data Centers** όπου μπορεί να υπάρχουν εκατοντάδες έως και χιλιάδες επεξεργαστές, κάτι που οδηγεί σε σημαντικό κόστος, τόσο οικονομικά όσο και περιβαλλοντικά.

Μια μέθοδο που μπορεί να βοηθήσει στο πρόβλημα της κατανάλωσης που περιγράφηκε παραπάνω είναι αυτή του **Undervolting** που περιγράφηκε αναλυτικά στο κεφάλαιο (3.1). Με αυτή την μέθοδο είναι δυνατό να ελαχιστοποιηθεί η κατανάλωση του επεξεργαστή στο ελάχιστο, ενώ ταυτόχρονα λειτουργεί με την ίδια απόδοση που είχε και στην αρχική.

Στο πείραμα που πραγματοποιήθηκε καταγράφηκαν οι θερμοκρασίες των δύο επεξεργαστών σε μια διάρκεια παραπάνω από μιας μέρας, **σε τρεις διαφορετικές τάσης**, με σκοπό να δειχθεί πως ή μέθοδος του **Undervolting** μπορεί να βοηθήσει στο πρόβλημα της κατανάλωσης, ειδικά σε μεγάλη κλίμακα όπως τα **Data Centers**. Τα τελικά αποτελέσματα της κατανάλωσης των δύο επεξεργαστών μπορεί να φανούν στα δύο επόμενα διαγράμματα.

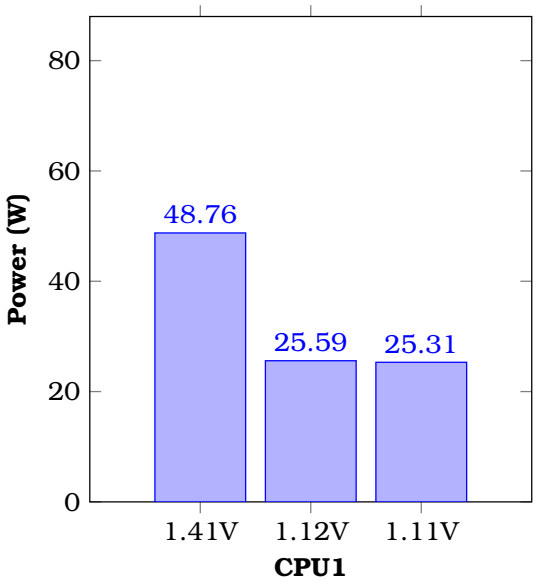


Figure 6.1: *Power Consumption of CPU1*

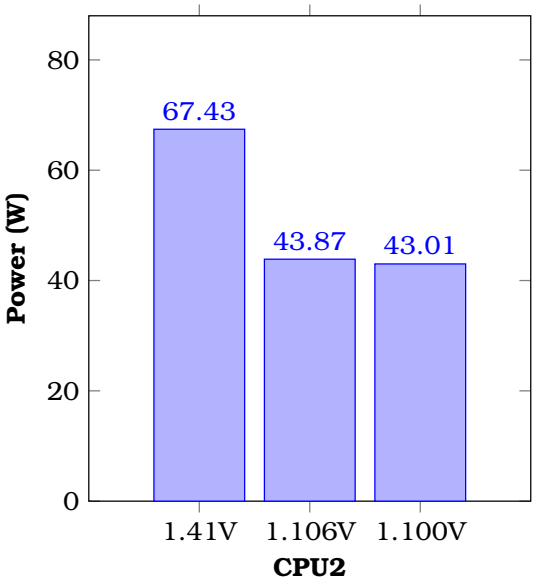


Figure 6.2: *Power Consumption of CPU2*

Στα διαγράμματα (6.1) και (6.2) φαίνονται τα αποτελέσματα της κατανάλωσης των δύο επεξεργαστών σε τρεις διαφορετικές τάσης αντίστοιχα. Όπως φαίνεται, στον οριζόντιο άξονα φαίνεται η μέση κατανάλωση, για όλα τα προγράμματα αξιολόγησης μαζί, σε **Watt**, ενώ στον κάθετο άξονα φαίνεται η λειτουργική τάση στην οποία λειτουργούσε ο κάθε επεξεργαστής. Το συμπέρασμα που βγαίνει αναλύοντας τα δύο αυτά διαγράμματα είναι πως και στις δύο περιπτώσεις όσο μειωνόταν η τάση των επεξεργαστών, μειωνόταν ανάλογα και η κατανάλωση τους κατά την διάρκεια λειτουργίας τους. Συνεπώς με την χρήση της μεθόδου **Undervoling** για την μείωση της λειτουργικής τάσης του επεξεργαστή έγινε εφικτό να μειωθεί η μέση κατανάλωση των επεξεργαστών κατά **51.9%** για την **CPU1** (6.1) και κατά **63.91%** για την **CPU2** (6.2).

6.3 Αξιοπιστία

Εκτός από την κατανάλωση ενός επεξεργαστή, άλλο ένα σημαντικό κριτήριο και κύριος λόγος που πραγματοποιήθηκε το εν λόγο πείραμα είναι το πόσο αξιόπιστος είναι ένας επεξεργαστής όταν αυτός εκτίθεστε σε υψηλή ιοντίζουσα ακτινοβολία.

Υψηλής ενέργειας σωματίδια προερχόμενα από κοσμικές πηγές ακτινοβολίας που εισβάλλουν στην ατμόσφαιρά της Γης είναι πιθανό να διαπεράσουν έναν επεξεργαστή [1]. Στην περίπτωση που στο εσωτερικό του επεξεργαστή υπάρξει κάποιο τρανζίστορ που έλαβε αρκετή φόρτιση, έτσι ώστε να αλλάξει κατάσταση από **0** σε **1** ή αντίστροφα, αυτό μπορεί να έχει καταστροφικές συνέπειες [1]. Στο πείραμα οι επεξεργαστές τοποθετήθηκαν σε μια ακτίνα νετρονίων, λαμβάνοντας υψηλά επίπεδα ακτινοβολίας τα οποία αντιστοιχούν σε 100 χρόνια έκθεσης του επεξεργαστή σε κοσμική ακτινοβολίας, αν ήταν τοποθετημένος στο επίπεδο της θάλασσας. Έτσι, με αυτόν τον τρόπο μπορεί να μελετηθεί ή συμπεριφορά του επεξεργαστή, ως προς την αξιοπιστία, σε μια διάρκεια 100 χρόνων.

Το κομμάτι που μελετάτε ως προς την αξιοπιστία των επεξεργαστών σε αυτό το πείραμα είναι το πόσα σφάλματα ανά λεπτό εμφανίζονταν στους δύο επεξεργαστές κατά την διάρκεια του πειράματος. Παρακάτω φαίνονται τα αντίστοιχα διαγράμματα που δείχνουν τα σφάλματα ανά λεπτό και στους δύο επεξεργαστές.

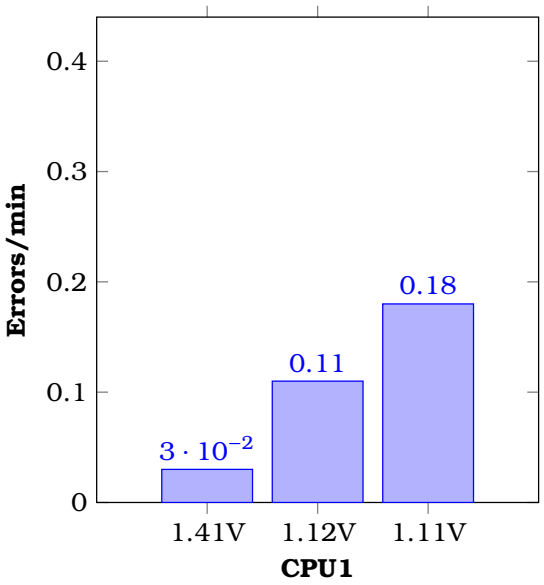


Figure 6.3: *Errors/min of CPU1*

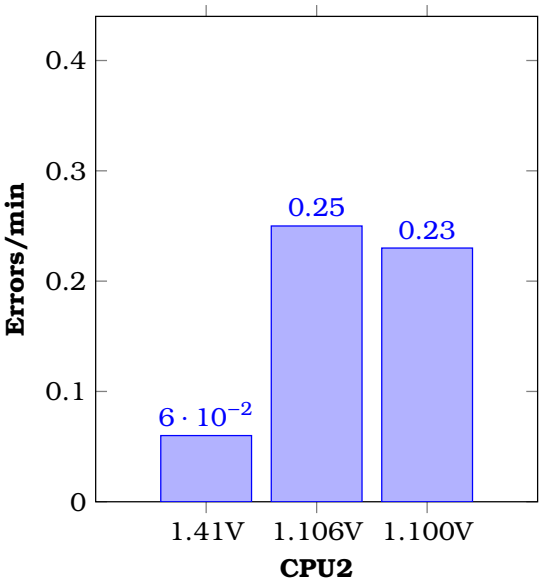


Figure 6.4: *Errors/min of CPU2*

Στα διαγράμματα (6.3) και (6.4) φαίνονται τα αποτελέσματα των σφαλμάτων ανά λεπτό των δύο επεξεργαστών σε τρεις διαφορετικές τάσης αντίστοιχα. Όπως φαίνεται, στον οριζόντιο άξονα φαίνονται τα σφάλματα ανά λεπτό, για όλα τα προγράμματα αξιολόγησης μαζί, σε Errors/min, ενώ στον κάθετο άξονα φαίνεται η λειτουργική τάση στην οποία λειτουργούσε ο κάθε επεξεργαστής. Συνεπώς, το συμπέρασμα που μπορεί να βγει αναλύοντας τα δύο αυτά διαγράμματα είναι πως οι επεξεργαστές λαμβάνουν πολλά περισσότερα σφάλματα όταν λειτουργούν σε χαμηλότερες τάσης από την τάση που έχει ορίσει ο κατασκευαστής του

επεξεργαστή.

6.4 Συμπεράσματα

Από τα δύο προηγούμενα κεφάλαια, (6.2) και (6.3), μπορεί να βγει το συμπέρασμα πως όταν σε έναν επεξεργαστή γίνει μείωση της λειτουργικής του τάσης, τότε υπάρχουν δύο σημαντικά αποτελέσματα. Το ένα είναι πως αν ένας επεξεργαστής φτάσει στην ελάχιστη λειτουργική του τάση, τότε μπορεί να επιτευχθεί μια τεράστια εξοικονόμηση ενέργειας, ειδικά σε περιπτώσεις που σε έναν χώρο υπάρχουν εκατοντάδες οι ακόμα και χιλιάδες επεξεργαστές, όπως σε ένα **Data Center**. Το δεύτερο αποτέλεσμα που συμπεραίνετε είναι πως προβλέπετε ότι ο επεξεργαστής θα είναι αρκετά ποιο ευάλωτος σε σφάλματα στην περίπτωση που λειτουργεί στην χαμηλότερη δυνατή λειτουργική του τάση. Αυτό θα μπορούσε να είναι πολύ επικίνδυνο σε περιπτώσεις των Data Centers, που στατιστικά λόγο του πλήθους των υπολογιστικών συστημάτων που υπάρχουν στον ίδιο χώρο, η πιθανότητα να προκληθεί ένα σφάλμα σε ένα από αυτά είναι είδη πολύ μεγάλη [1]. Συνεπώς, σε περίπτωση χρήσης της μεθόδου **Undervolting** για λόγους εξοικονόμησης ενέργειας, θα πρέπει να παρθούν οι σωστές αποφάσεις βάση και των δύο παραπάνω συμπερασμάτων.

Βιβλιογραφία

[1] Shubu Mukherjee. *architrecutre design for soft errors*. Morgan Kaufmann, 1η έκδοση, 2008.