Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

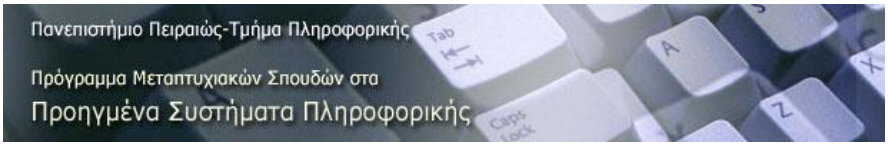
Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Με Υποτροφία από το Ιδρυμα Κρατικών Υποτροφιών (Ι.Κ.Υ.)

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Τεχνικές εισαγωγής, ανίχνευσης και διόρθωσης προσωρινών σφαλμάτων σε προγραμματιζόμενες συσκευές λογικής Xilinx Virtex-5**
Ονοματεπώνυμο Φοιτητή	**ΤΣΩΝΗ ΠΑΝΑΓΙΩΤΑ του ΕΥΣΤΑΘΙΟΥ**
Αριθμός Μητρώου	**ΜΠΣΠ/08058**
Κατεύθυνση	**Ενσωματωμένα Υπολογιστικά Συστήματα**
Επιβλέπων	**Ψαράκης Μιχάλης, Επίκουρος καθηγητής**



Ημερομηνία Παράδοσης: **Μάρτιος 2011**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Μιχαήλ Ψαράκης, Επίκουρος καθηγητής	Δημήτριος Γκιζόπουλος, Αναπληρωτής Καθηγητής	Δημήτριος Βέργαδος, Λέκτορας

Abstract

Single Event Upsets (SEUs) are transient faults that appear in semiconductor devices. SEUs occur when charged particles hit the silicon transferring enough energy in order to provoke a fault in the system. The main consequences of the transient effect are ψηφίο flips in the memory elements. SEUs have been constantly magnified the last years, due to the continuous technology evolution that has led to highly complex architectures, which integrate a large amount of embedded memories, followed by an amazing downscaling of transistor feauture sizes. So protecting integrated circuits against upsets has become imperative need. For that reason, researchers have proposed fault tolerant techniques that maintain the reliable operation of ICs despite the existence of upsets. Our aim was to create an application, in which we could implement SEUs and observe the reaction of the device. The application consists of two different circuits. The first one combines an SEU Controller macro from XILINX and a soft core called PicoBlaze and it is responsible for injecting, detecting and correcting SEUs in the device. The second circuit is the main application and combines a second PicoBlaze with an LCD Controller. The circuit is responsible for shifting a message to the LCD display of the device and it is the main area that the SEUs are taking place. The application implemented in an FPGA device from XILINX (VIRTEX-5 ML505) and we injected SEUs at specific locations of the configuration map and then observe the reaction and the critical areas of the design.

Περίληψη

Ως απλή προσωρινή διαταραχή (Single Event Upset – SEU) ορίζεται η αλλαγή κατάστασης, σε κόμβο ενός ηλεκτρονικού κυκλώματος, που προκαλείται από την σύγκρουση του τελευταίου, με φορτισμένα σωματίδια της ατμόσφαιρας. Η σύγκρουση εκλύει τόση ενέργεια, ώστε να προξενήσει διαταραχή στο σύστημα που την επιδέχεται. Τα τελευταία χρόνια, οι επιπτώσεις των SEUs στα κυκλώματα γίνονται ολοένα και σημαντικότερες, λόγω των τεχνολογικών εξελίξεων στην διαδικασία κατασκευής των ολοκληρωμένων κυκλωμάτων. Οι ερευνητές λοιπόν αναζητούν μεθόδους ανοχής σφαλμάτων, ώστε τα νέα συστήματα να μπορούν να αποδώσουν την ορθή λειτουργία τους ακόμα και υπό την επήρεια SEUs. Στην παρούσα μεταπτυχιακή διατριβή δημιουργήθηκε εφαρμογή, στην οποία είναι εφικτή η εισαγωγή SEUs και η παρατήρηση της αντίδρασης του κυκλώματος. Το σύστημα αποτελείται από δύο μέρη. Το πρώτο φέρει τον SEU Controller της XILINX ο οποίος προγραμματίζεται μέσω του PicoBlaze, ενός ενσωματωμένου μικροεπεξεργαστή, με στόχο την εισαγωγή, ανίχνευση και διόρθωση προσωρινών σφαλμάτων στην συσκευή. Το δεύτερο μέρος φέρει έναν ακόμα PicoBlaze ο οποίος ελέγχει την LCD οθόνη της συσκευής. Ουσιαστικά, φροντίζει για την κύλιση ενός μηνύματος στην οθόνη και αποτελεί το κύριο μέρος του κυκλώματος που επιδέχεται τα σφάλματα από τον SEU Controller. Η εφαρμογή υλοποιήθηκε σε FPGA συσκευή της XILINX (VIRTEX-5 ML505), στην οποία εισήχθη πληθώρα SEUs σε καθορισμένες περιοχές της μνήμης διαμόρφωσης, με στόχο να παρατηρηθεί η αντίδραση του κυκλώματος και να εντοπιστούν τα κρίσιμα σημεία της σχεδίασης.

Πίνακας Περιεχομένων

Λίστα Εικόνων και Πινάκων

1. Εισαγωγή

Τα τυχαία και βραχύβια σφάλματα στις συσκευές ημιαγωγών, αποτελούν αντικείμενο έρευνας, από την εποχή των πρώτων αστοχιών, που παρατηρήθηκαν σε διαστημικές εφαρμογές. Ο στόχος της έρευνας αφορούσε την βελτίωση της αντίδρασης των κυκλωμάτων, σε παρόμοιες διαταραχές, ώστε να μην επιτρέπονται τυχόν αποκλίσεις από την ομαλή τους λειτουργία.

Ως κύρια αιτία, των εν λόγω διαταραχών, ορίστηκε η κοσμική ακτινοβολία και τα σωματίδια που περιέχει. Η σύγκρουση μεταξύ των ημιαγωγών και των εξωγενών αυτών σωματιδίων, συνεπάγεται την διαταραχή της ύλης και άρα την πρόκληση εσφαλμένης συμπεριφοράς. Καθώς οι σημερινές συσκευές τείνουν να συρρικνώνονται, αυτές οι συγκρούσεις έχουν δυσμενέστερα αποτελέσματα στην ομαλή λειτουργία των κυκλωμάτων. Μιας και η εξάλειψη του φαινομένου είναι αδύνατη, αφού η επαφή με το περιβάλλον είναι αναπόφευκτη, αναπτύσσονται μέθοδοι άμβλυνσης των επιπτώσεων στα κυκλώματα (Mitigation Techniques) [1].

Η σύγκρουση αυτή, συσκευών και σωματιδίων, πήρε το όνομα απλό προσωρινό φαινόμενο (Single Event Effect - SEE) και συνεπάγεται μόνιμες ή βραχύβιες βλάβες, ανάλογα με το ποσοστό ενέργειας που εκλύεται [2]. Είθισται, οι διαταραχές να είναι προσωρινές και κατατάσσονται ως SEUs και σε αυτή την περίπτωση αφορούν την αντιστροφή μεμονωμένων ψηφία στα στοιχεία της μνήμης του κυκλώματος.

Οι τεχνικές άμβλυνσης που υιοθετούνται έχουν ως στόχο την ομαλή λειτουργία των ολοκληρωμένων κυκλωμάτων (ICs), ανεξάρτητα από την παρουσία ενδεχόμενων SEUs. Αν και έχουν προταθεί πλήθος τέτοιων μεθόδων, το θέμα απασχολεί ακόμα έντονα τους ερευνητές. Η βασικότερη παρατήρηση στον τομέα της επιλογής μεθόδου, είναι ότι κάθε συσκευή φέρει ιδιαίτερα σχεδιαστικά χαρακτηριστικά και αυτά ορίζουν την ικανοποιητικότερη μέθοδο προστασίας της. Αρκεί να αναλογιστεί κανείς την εξέλιξη των ICs την τελευταία δεκαετία. Με στόχο την καλύτερη απόδοση, την περισσότερη μνήμη και το μικρότερο κόστος, τα κυκλώματα έχουν αποκτήσει περιπλοκότερες αρχιτεκτονικές δομές και σαφώς μικρότερο μέγεθος. Άρα και ο τρόπος χειρισμού των SEUs σε σχέση με τα ολοκληρωμένα κυκλώματα, διαφοροποιείται ανάλογα με την εξέλιξη της δομής τους.

Παραδείγματα των ανωτέρω είναι οι δομές των ολοκληρωμένων κυκλωμάτων ειδικής εφαρμογής (Application Specific Integrated Circuits - ASIC), οι νέοι μικροεπεξεργαστές με τα εκατομμύρια των τρανζίστορ, τα πρότυπα ολοκληρωμένα κυκλώματα (Field Programmable Gate Array - FPGA) και τα συστήματα-σε-τσιπ (SOC), που φέρουν ενσωματωμένους μικροεπεξεργαστές, μνήμες και αναλογικά μπλοκ, σε μία δομή. Αυτές οι αρχιτεκτονικές άλλαξαν δραματικά τον σχεδιασμό νέων εφαρμογών και οδήγησαν στην ικανότητα επεξεργασίας μεγάλης ποσότητας πληροφορίας από ένα και μόνο τσιπ. Ειδικότερα για τα FPGAs, η χρήση τους πλέον σε εμπορική κλίμακα έχει αυξηθεί σημαντικά, μιας και εξασφαλίζουν υψηλό βαθμό απόδοσης και ευελιξίας στον τελικό σχεδιασμό[3]. Οι ενσωματωμένοι μικροεπεξεργαστές (soft cores) που υποστηρίζουν, καλύπτουν ανάγκες σε συστήματα ψηφιακής επεξεργασίας σήματος (Digital Signal Processing - DSP), ενώ ταυτόχρονα καταλαμβάνουν μικρό χώρο, απαιτούν λιγότερη τροφοδοτική ισχύ και είναι οικονομικότερα [4,5].

Τα πιο δημοφιλή FPGAs είναι τα βασιζόμενα σε SRAM (SRAM-based), που παρέχουν εξαιρετική ευελιξία στον χρήστη μιας και πλέον μπορεί να επανδιαμορφώση την συσκευή του, στον

χώρο λειτουργίας της. Στην παρούσα μεταπτυχιακή διατριβή θα αναφερθούμε στις μεθόδους άμβλυνσης των SEUs σε συσκευές ημιαγωγών, με ιδιαίτερη μνεία στις τεχνικές που υιοθετούνται για τα SRAM-based FPGAs. Ακολούθως, θα αναπτυχθούν οι στρατηγικές αντιμετώπισης που προτείνει η XILINX στους σχεδιαστές εφαρμογών που χρησιμοποιούν τα FPGAs της εταιρείας.

Το ουσιαστικότερο μέρος της διατριβής καταλαμβάνει το πειραματικό μέρος, στο οποίο δημιουργήθηκε εφαρμογή σε συσκευή VIRTEX-5 ML505, της XILINX. Το κύκλωμα χωρίζεται σε δύο ανεξάρτητα μέρη. Το ένα αποτελείται από έναν ενσωματωμένο μικροεπεξεργαστή, τον PicoBlaze, ο οποίος υλοποιεί όλα τα σήματα επικοινωνίας για την LCD οθόνη της συσκευής και επιτυγχάνει την δημιουργία μηνύματος υπό κύλιση. Επίσης, συνδέει τα τέσσερα από τα πέντε κομβία πίεσης της συσκευής και τέσσερις από τους οκτώ διακόπτες, με αντίστοιχα LEDs. Εν γένει, στόχος ήταν να χρησιμοποιηθούν τα περιφερειακά εκείνα της συσκευής, που αν διαταραχθεί η λειτουργία τους, να γίνει ορατό το αποτέλεσμα από τον χρήστη. Το δεύτερο μέρος του κυκλώματος αποτελείται από τον SEU Controller της XILINX ο οποίος ελέγχεται από έναν ακόμα PicoBlaze και προγραμματίζεται ώστε να εισάγει σφάλματα στο προαναφερόμενο κύκλωμα. Ο SEU Controller έχει την δυνατότητα να εισάγει σφάλματα σε προκαθορισμένες περιοχές, να εκτελεί ανίχνευση SEUs σε όλη την συσκευή και να διορθώνει όποια διαταραχή εντοπίσει. Αφενός, χρησιμοποιήθηκε για να προκαλεί στοχευόμενες διαταραχές στο κύκλωμα, ώστε να διαπιστωθεί το μέγεθος της ευαισθησίας τόσο της συσκευής απέναντι στα τυχαία σφάλματα, αλλά και του ίδιου του κυκλώματος της εφαρμογής. Αφετέρου, στόχος ήταν η ενασχόληση με τον ίδιο τον SEU Controller, για να καθοριστεί αν πρόκειται για ένα εύρωστο εργαλείο, που μπορεί να λειτουργήσει αξιόπιστα και κυρίως υπό ποιες συνθήκες. Ένα τέτοιο εργαλείο, που θα δοκιμάζει τις εφαρμογές για πιθανές λειτουργικές αστοχίες, προτού καταλήξουν στον χρήστη, είναι εξαιρετικά χρήσιμο, αν και εφόσον λειτουργεί ικανοποιητικά.

Στο κομμάτι των πειραμάτων, εντοπίστηκε αρχικά ο χώρος που καταλαμβάνει τόσο το κύκλωμα του PicoBlaze και τα σήματα της LCD στην συσκευή, όσο και το κύκλωμα του PicoBlaze και του SEU Controller, στο οποίο δεν θα έπρεπε να εισαχθούν σφάλματα. Για να είναι αξιόπιστες οι δοκιμές, το κομμάτι που ελέγχει την εισαγωγή σφαλμάτων και την διόρθωση αυτών πρέπει να παραμείνει ακέραιο. Εν συνεχεία, η περιοχή ενδιαφέροντος χωρίστηκε στα κελιά διαμόρφωσης και μπλοκ μνήμης και SEUs εισήχθησαν και στα δύο μέρη. Τα σφάλματα εισήχθησαν τόσο από τον χρήστη, σε κάθε πλαίσιο (frame) ξεχωριστά, όσο και μέσω προγράμματος που εκτελούσε ο PicoBlaze και αξιολογήθηκαν όλες οι καταγεγραμμένες αντιδράσεις.

Στα κεφάλαια που ακολουθούν θα αναπτυχθούν τα είδη των προσωρινών σφαλμάτων που μας απασχόλησαν, οι τεχνικές που χρησιμοποιούνται ευρέως για άμβλυνση αυτών των φαινομένων στις συσκευές που εντοπίζονται και ιδιαίτερα οι τρόποι αντιμετώπισης των SEUs, που προτείνει η ίδια η XILINX για τις συσκευές της. Τέλος, αναπτύσσεται το πειραματικό μέρος και παρατίθενται όλα εκείνα τα στοιχεία που προέκυψαν κατά τις δοκιμές, καθώς και τα συμπεράσματά μας.

2. Προσωρινά Σφάλματα

2.1. Ιστορική Αναδρομή

Τα προσωρινά σφάλματα αποτελούν αντικείμενο μελέτης, για τους μηχανικούς διαφόρων τομέων της επιστήμης, όπως της αεροναυπηγικής και της ηλεκτρονικής, για πάνω από μισό αιώνα. Την περίοδο 1954-1957 είχαν σημειωθεί οι πρώτες αστοχίες σε ψηφιακά κυκλώματα, που σχετίζονταν με δοκιμές σε πυρηνικές βόμβες. Τότε, είχαν θεωρηθεί ως ανωμαλίες στο σύστημα παρατήρησης των δοκιμών, μιας και ήταν τυχαίες και δεν μπορούσαν να αιτιολογηθούν, λόγω της ορθής λειτουργίας του υλικού [6]. Θεωρητικά, η πρώτη μελέτη στο ζήτημα του επηρεασμού των ηλεκτρονικών κυκλωμάτων από τις κοσμικές ακτίνες είναι των Wallmark and Marcus [7]. Οι συγγραφείς είχαν προβλέψει ότι οι κοσμικές ακτίνες θα διαδραμάτιζαν αρνητικό ρόλο στην ομαλή λειτουργία των μικροκυκλωμάτων, μέσω ισχυρού ιονισμού, όταν αυτά τα κυκλώματα θα αποκτούσαν ιδιαίτερα μικρό μέγεθος [8]. Κατά την δεκαετία του 1970 και έως τα μέσα της δεκαετίας του 1980, ο αντίκτυπός της ακτινοβολίας άρχισε να απασχολεί ολοένα και περισσότερους ερευνητές, οι οποίοι προσπάθησαν να τον ερμηνεύσουν σύμφωνα με τους κανόνες της φυσικής. Επιπρόσθετα, από το 1950 άρχισαν να αναπτύσσονται θεωρίες, για υπολογιστικά συστήματα με ανοχή στο σφάλμα και ικανότητα αυτό-ίασης (self-healing), λόγω των αυξημένων απαιτήσεων σε αξιοπιστία που υπόκεινται κάποια συστήματα, όπως αυτά που χρησιμοποιούνταν σε διαστημικές αποστολές [9].

Οι May και Woods της Intel Corporation [10] καθόρισαν ότι αυτά τα σφάλματα προέρχονταν από τα ελάχιστα ραδιενεργά τμήματα των κυκλωμάτων, όπως το ουράνιο και το θόριο, τα οποία κατέρρεαν μέσω ιονισμού. Στην ερευνά τους, γίνεται η πρώτη αναφορά σε λάθη που προέρχονται από την ακτινοβολία και ονομάζονται προσωρινά σφάλματα (soft errors). Ο όρος χρησιμοποιήθηκε για να γίνει ο διαχωρισμός μεταξύ αυτών των λαθών και εκείνων που οφείλονταν σε μόνιμες βλάβες του υλικού. Οι Guenzer και Wolicki [11] ανέφεραν ότι τέτοιου τύπου λάθη, δεν οφείλονταν αποκλειστικά στα μέρη που αποτελούνταν από ουράνιο ή θόριο, αλλά ότι λάμβαναν χώρα πυρηνικές αντιδράσεις, που παρήγαγαν νετρόνια και πρωτόνια υψηλής ενεργειακής άξιας, προξενώντας διαταραχές στα κυκλώματα. Λόγω του τίτλου της μελέτης τους «Single Event Upset of Dynamic RAMs by Neutrons and Protons», ο όρος «SEU» χρησιμοποιείται έκτοτε, για να περιγράψει αυτή την μορφή αυτών των σφαλμάτων. Το 1979, οι Ziegler και Lanford από την IBM [12], προέβλεψαν ότι οι κοσμικές ακτίνες θα επηρέαζαν τα ηλεκτρονικά κυκλώματα που θα βρίσκονταν ακόμα και στο επίπεδο της θάλασσας, κατά συνέπεια το πρόβλημα δεν αφορά μόνο τις διαστημικές αποστολές.

Παραδείγματα Εφαρμογών	Ύψος (Feet)	Ροή Νετρονίων (σχετική)	FPGAs /Σύστημα	Αριθμός Διαταραχών/ 1M-FPGA πύλες/μέρα (,13μ)	MTBF (ώρες)		FIT (εκατομμύρια)	
					0,13μ	0,09μ	0,13μ	0,09μ
Επίγεια Δίκτυα	5000	1	512	4,19E-4	112	58	8,92	17,24

Επικοινωνίας								
Πολιτικά Συστήματα αεροπλοΐας	30.000	~40	4	1,85E-2	324	162	3,09	6,17
Στρατιωτικά Συστήματα αεροπλοΐας	60.000	>160	16	8,33E-2	18	9	55,56	111,11

Πίνακας 2.1:Ρυθμός αστοχίας σε SRAM-based FPGAs λόγω φαινομένων νετρονίων (ACTEL) [13]

Προσφάτως, ο ρυθμός προσωρινών σφαλμάτων (Soft Error Rate - SER),που μετρήθηκε σε SRAM-based FPGAs, δείχνει έναν σημαντικό και διαρκώς αυξανόμενο κίνδυνο λειτουργικών αστοχιών των κυκλωμάτων, λόγω των λαθών που προκύπτουν, στις περιπτώσεις υψηλής πυκνότητας των δεδομένων διαμόρφωσης. Στον πίνακα 2.1 φαίνεται ο ρυθμός αστοχίας, για διαφορετικές εφαρμογές, στις οποίες δεν χρησιμοποιείται κανενός είδους προστασία. Ο αριθμός των λαθών που σημειώνονται στο 1 εκατομμύριο πύλες/μέρα, αυξάνεται μεταξύ των περιπτώσεων 1-3, λόγω της ανάλογης σχέσης μεταξύ υψόμετρου και πυκνότητας νετρονίων. Ο πίνακας περιλαμβάνει τον ρυθμό αστοχίας για διεργασίες στα 90nm. Αναμένεται ότι τα προσωρινά σφάλματα που σχετίζονται με την εκπομπή νετρονίων, διπλασιάζονται καθώς η τεχνολογία προχώρα από τα 0,13μ στα 0,09μ.

Εν κατακλείδι, τα λάθη λόγω ακτινοβολίας είναι πλέον μία από τις σημαντικότερες αιτίες αποτυχίας των σύγχρονων ηλεκτρονικών κυκλωμάτων. Ο ρυθμός αστοχίας των εμπορικών ολοκληρωμένων κυκλωμάτων παίρνει τιμές μεταξύ 100-1000 FIT (Failure in Time: ο αριθμός των αστοχιών ανά 10^9 ώρες λειτουργίας). Συγκρινόμενος με τον ρυθμό αστοχίας του υλικού που είναι μεταξύ 1-100 FIT, ο ρυθμός αστοχίας σε μία ενσωματωμένη SRAM μπορεί εύκολα να αγγίξει τα 1000FIT/Mbit. Γι' αυτό ακολουθείται μία προσέγγιση τεσσάρων φάσεων, ώστε οι διαταραχές να αντιμετωπιστούν με επιτυχία [14]:

1. Μέθοδοι για να προστατευτούν τα ολοκληρωμένα κυκλώματα από τα προσωρινά σφάλματα (*πρόληψη* – prevention).
2. Μέθοδοι ανίχνευσης των προσωρινών σφαλμάτων (*δοκιμές* – testing).
3. Μέθοδοι αξιολόγησης του αντίκτυπου των προσωρινών σφαλμάτων (*εκτίμηση* – assessment).
4. Μέθοδοι ανάκαμψης από τα λάθη (*αποκατάσταση* – recovery).

2.2. Κατηγορίες Προσωρινών Σφαλμάτων

Στην βιομηχανία των ηλεκτρονικών κυκλωμάτων, οι αστοχίες διαχωρίζονται σε προσωρινές (soft fail) και μόνιμες (hard ή permanent fail) που είναι επισκευάσιμες [12]. Μετά την τέλεση και διαπίστωση ενός προσωρινού σφάλματος, δεν συνεπάγεται ότι το κύκλωμα που το υπέστη είναι λιγότερο αξιόπιστο από πριν, μιας και πρόκειται για ένα εντελώς τυχαίο γεγονός. Αυτές οι αστοχίες μπορούν να προκληθούν λόγω γειτνίασης με πηγές παραγωγής θορύβου, όπως είναι τα τροφοδοτικά, ο φωτισμός, τα φαινόμενα ηλεκτροστατικής εκφόρτισης (ESD) ή λόγω της θερμικής ακτινοβολίας του πλανήτη, η οποία αναφέρεται

στα ακτινοβολούντα άστρα και στα ατμοσφαιρικά αέρια. Τα προσωρινά σφάλματα είναι μη καταστροφικά και χωρίζονται σε δύο βασικές κατηγορίες [15]:

1. Στα *βραχύβια* (transient) λάθη που προκαλούνται εξαιτίας των περιβαλλοντικών συνθηκών, όπως είναι η θερμοκρασία, η υγρασία, η πίεση, η τάση, η ισχύς, τα τροφοδοτικά, οι δονήσεις, οι ηλεκτρομαγνητικές παρεμβολές, η γείωση και οι κοσμικές ακτίνες.
2. Στα διακοπτόμενα (intermittent) λάθη που οφείλονται σε στοιχεία του συστήματος, όπως είναι οι προβληματικές συνδέσεις, ο χρόνος ζωής των εξαρτημάτων, ο χρονισμός, ο θόρυβος λόγω της τροφοδοσίας, οι αλλαγές στην ωμική, επαγωγική ή χωρητική αντίσταση και ο θόρυβος του κυκλώματος.

Η εξέλιξη στο σχεδιασμό και την κατασκευή των κυκλωμάτων, μπορεί σχεδόν να αποκλείσει την δημιουργία SEUs εξαιτίας μη-περιβαλλοντικών παραγόντων. Παρόλα αυτά, τα λάθη που προξενούνται από τις κοσμικές ακτίνες, παραμένουν ο κύριος παράγοντας αστοχίας των ηλεκτρονικών συστημάτων.

2.3. Ημιαγωγοί και μηχανισμοί ακτινοβόλησης

Τρεις είναι οι βασικές πηγές ακτινοβόλησης, οι οποίες ευθύνονται για την δημιουργία προσωρινών σφαλμάτων στις συσκευές ημιαγωγών [16]:

1. Τα μόρια alpha [*] που εκπέμπονται όταν ο πυρήνας ενός ασταθούς ισότοπου εξασθενεί σε χαμηλότερη ενεργειακή στάθμη. Στη φύση, οι κυριότερες πηγές των μορίων alpha είναι οι ραδιενεργοί ρύποι όπως τα ισότοπα που βρίσκονται στα flip-chip, ο χρυσός που βρίσκεται στην σύνδεση των καλωδιώσεων και στα καπάκια, το αλουμίνιο στα κεραμικά περιβλήματα, τα κράματα μολύβδου και οι μεταλλικές διασυνδέσεις.
2. Τα υψηλής κινητικής ενέργεια νετρόνια (> 1 MeV) που παράγονται λόγω κοσμικής ακτινοβολίας, μπορούν να οδηγήσουν σε προσωρινό σφάλμα τις συσκευές ημιαγωγών, μέσω δευτερογενή ιονισμού που προκύπτει από την αντίδραση του νετρονίου με το νουκλεϊνικό πυρίτιο. Οι κοσμικές ακτίνες που αρχικά προξενούνται από επαφή του γαλαξία με την ατμόσφαιρα της γης, μπορούν να παράγουν πληθώρα περίπλοκων μορίων. Λιγότερο από το 1% των αρχικών σωματιδίων φτάνουν στην γη και τα κυρίαρχα μόρια περιλαμβάνουν μυόνια, νετρόνια, πρωτόνια και πιόνια. Επειδή τα μυόνια και πιόνια έχουν μικρό χρόνο ζωής και τα πρωτόνια και τα ηλεκτρόνια ασθενούν λόγω της αλληλεπίδρασης με την ατμόσφαιρα (δυνάμεις Coulomb), απομένουν τα νετρόνια ως την πιο πιθανή πηγή κοσμικής ακτινοβολίας, η οποία προκαλεί SEUs σε μικρού μεγέθους ημιαγωγούς. Η πυκνότητα της ροής των νετρονίων αυξάνεται ανάλογα, σε σχέση με υψόμετρο από το επίπεδο της θάλασσας [17].
3. Η τρίτη σημαντικότερη πηγή ιονιζόμενων μορίων σε ηλεκτρονικές συσκευές, είναι η δευτερογενής ακτινοβολία που παράγεται από την αλληλεπίδραση μεταξύ των νετρονίων των

[*] Τα Alpha particle αποτελούνται από 2 πρωτόνια και 2 νετρόνια και η μορφή του μορίου που σχηματίζουν είναι πανομοιότυπη με εκείνη του helium nucleus, το οποίο παράγεται κατά την διεργασία alpha decay.

κοσμικών ακτινών και του ισότοπου του βορίου (boron-10, ^{10}B), το οποίο είναι εξαιρετικά διαδεδομένο και χρησιμοποιείται για την δημιουργία μονωτικών στρωμάτων στα ολοκληρωμένα κυκλώματα.

2.4. FPGAs και SEUs

Οι συσκευές FPGA λειτουργικά, βασίζονται κυρίως σε SRAM μνήμες (Static Random Access Memory), η αξιοπιστία των οποίων περιορίζεται, καθώς παρουσιάζουν ευαισθησία σε SEUs ακόμα και σε χαμηλό υψόμετρο [18].Η πιθανότητα να συμβεί ένα SEU εξαρτάται από το περιβάλλον λειτουργίας της συσκευής καθώς και την εφαρμογή που καλείται να εκτελέσει και ποικίλλει ανάλογα με την ηλιακή φωτεινότητα, το γεωγραφικό πλάτος και το ύψος [6].

Τα SEUs επηρεάζουν όλα τα μέρη μιας SRAM, όπως είναι η μνήμη διαμόρφωσης, η κατανεμημένη RAM, οι καταχωρητές και η λογική ελέγχου και επαναδιαμόρφωσης. Αναλόγως, με την περιοχή που θα υποστεί διαταραχή, το παραγόμενο SEU μπορεί να οδηγήσει σε λειτουργική αστοχία ή και απώλεια ελέγχου η οποία ορίζεται ως απλό φαινόμενο λειτουργικής διακοπής (Single Event functional interrupt - SEFI)[19].Το SEU μπορεί να επηρεάσει πολλαπλά κελιά μνήμης, ειδικότερα όταν αυτά γειτονεύουν. Υπάρχουν τρεις κύριες μέθοδοι ώστε να αξιολογηθούν οι παράμετροι εξάρτησης υλικού και SEUs: η προσομοίωση (software simulation), η εξομοίωση (hardware emulation) και η ακτινοβόληση (irradiation) μέσω ειδικών δοκιμών [20,21].

Η επάρκεια του εκάστοτε προσομοιωτή εξαρτάται από την αρτιότητα κατασκευής του (συνήθως προέρχεται από τον κατασκευαστή του FPGA). Λειτουργικά οι προσομοιωτές περιορίζονται σε ενασχόληση, στο επίπεδο των λογικών στοιχείων με χρήση πίνακα (Look-Up Tables - LUTs) σε λογική μεταφοράς καταχωρητή (Register Transfer Logic - RTL), χωρίς να εμβαθύνουν στην δρομολόγηση μέσα στο ίδιο το FPGA [21]. Όσο καλύτερο είναι το μοντέλο προσομοίωσης που χρησιμοποιείται, τόσο περισσότερο χρόνο δαπανά για να δώσει αποτελέσματα.

Η ακτινοβόληση παρέχει μία ακριβή αξιολόγηση της λειτουργικότητας του κυκλώματος, υπό την επήρεια των SEUs. Δεν απαιτείται η πλήρης εικόνα της δομής του FPGA, ούτε και της λογικής ελέγχου του. Η προετοιμασία για τις δοκιμές απαιτεί πληροφορίες για το τυπωμένο κύκλωμα (printed circuit board - PCB) του FPGA και με τα δεδομένα εξόδου. Για την δοκιμή καθαυτή, χρειάζεται ειδικός θάλαμος στον οποίο θα διενεργούνται τα τεστ που θα συνοδεύεται από μια εγκατάσταση ακτινοβόλησης [20]. Τα αποτελέσματα αυτών των δοκιμών είναι αξιόπιστα και ενδεικτικά για τον τρόπο λειτουργίας της συσκευής σε πραγματικές συνθήκες, όταν το περιβάλλον δεν είναι ιδανικό.

Μεταξύ της προσομοίωσης και της ακτινοβόλησης, λαμβάνει χώρα η εξομοίωση, κατά την οποία όμως επιβάλλεται η πλήρης γνώση της δομής του FPGA. Τα αποτελέσματα προκύπτουν από την λεπτομερή χαρτογράφηση και ανάλυση της συσκευής και συνήθως είναι εγγύτερα στα αποτελέσματα που προκύπτουν από τις δοκιμές ακτινοβόλησης. Σε όλα τα προαναφερόμενα ο χρόνος διαδραματίζει σημαντικό ρόλο και σε μεγάλες εφαρμογές είναι εξαιρετικά περιορισμένος.

2.5. Δομή του FPGA

Τα FPGA ανάλογα με την περιοχή που είναι τοποθετημένα και την λειτουργία τους χωρίζονται στα:

1. LUT: διατηρούν την λογική συνάρτηση στην SRAM μνήμη.
2. Διασύνδεση κελιών (Cell interconnection): Αποτελεί την διαμόρφωση των κελιών λογικής. Αυτά τα ψηφία ευθύνονται για την σωστή επιλογή εισόδων στα LUTs, την ανάδραση στα λογικά κελιά και την ορθή έξοδο.
3. Δίαυλος–σε-κελί/κελί-σε-δίαυλο (BUS to Cell/Cell to BUS): Πρόκειται για μία αμφίδρομη διασύνδεση, η οποία συνδέει τα λογικά κελιά με έναν από τους διαύλους.
4. Διασταύρωση διαύλων (BUS crossing): είναι η κεντρική κάθετη σύνδεση στην διασταύρωση των διαύλων, η οποία επιτρέπει την σύνδεση μεταξύ αυτών των γραμμών.
5. Επαναλήπτης διαύλου (BUS repeater): πρόκειται για μεταγωγέα τεσσάρων θυρών, που επιτρέπει την οδήγηση κάθε καλωδίου εισόδου.
6. Απαγορευμένα (Forbidden): αποτελείται από συγκεκριμένα ψηφία, των ψηφίων ακολουθίας διαμόρφωσης (bitstream), στα οποία δεν έχει αντιστοιχηθεί συγκεκριμένο SRAM κελί.

Η ανωτέρω λίστα δεν είναι πλήρης. Περιλαμβάνονται επίσης μνήμες RAM, σημεία επανεκκίνησης (reset), ρολόγια (clocks) και σημεία εισόδου/εξόδου (I/O pad).

2.6. Σφάλματα στα FPGA

Κατά την διαμόρφωση ενός FPGA, ώστε να εκτελέσει συγκεκριμένη λειτουργία, εισάγονται στην συσκευή ψηφία ελέγχου (control bits) υπό την μορφή ακολουθίας Τα SRAM κελιά εντός του FPGA μπορούν να αντιμετωπιστούν σαν ένας μακρύς καταχωρητής, στον οποίο τα ψηφία διαμόρφωσης κατηγοριοποιούνται σε χρηστικά, αχρησιμοποίητα και άγνωστα. Βάσει αυτών, ομαδοποιούνται και τα πιθανά σφάλματα που υπόκειται η συσκευή (Εικόνα 2.1). Ανάλογα με το κύκλωμα που έχει ενσωματωθεί στο FPGA ορίζεται η κρισιμότητα κάθε περιοχής και άρα το ποσοστό διακύβευσης της τελικής λειτουργίας, αν υποστεί SEU.

Ο συσχετισμός των κατηγοριών των ψηφίων και των σφαλμάτων έχει ως εξής:

1. Τα *αχρησιμοποίητα* ψηφία (unused bits) δεν αφορούν την περιοχή σχεδιασμού και δεν αναμένεται να προξενήσουν στατικές ή δυναμικές αλλαγές στο κύκλωμα. Παρόλα αυτά, ένα σφάλμα σε αυτό το γκρουπ μπορεί να οδηγήσει σε επιπλέον κατανάλωση ρεύματος. Μία περαιτέρω κατηγοριοποίηση αυτής της ομάδας δεν είναι εφικτή, μιας και τα αποτελέσματα της διαταραχής, δεν είναι ορατά.
2. Τα *χρηστικά* ψηφία (used bits) δημιουργούνται κυρίως από το ίδιο το κύκλωμα. Κάθε ψηφίο αυτής της ομάδας επηρεάζει ένα ενεργό μέρος του σχεδιασμού. Ουσιαστικά, καταλαμβάνουν θέσεις εξαιρετικά κρίσιμες και οποιαδήποτε αλλαγή στην κατάστασή τους, μπορεί να αλλάξει την ορθή λειτουργία του κυκλώματος. Περαιτέρω κατηγοριοποίηση τους έχει ως εξής:

a. *Ανοιχτά* (open): αναπαριστούν την διακοπή μιας καλωδίωσης (ανοιχτοκύκλωμα), κάτι που μπορεί να λάβει χώρα στο FPGA από πολλαπλές αιτίες. Οι πιο χαρακτηριστικές περιπτώσεις είναι η διακοπή στις διασταυρώσεις διαύλων, σε πολυπλέκτες και πύλες μεταφοράς.

b. *Εναλλασσόμενα* (Alternate): διαφοροποιούν τον τελικό σχεδιασμό χωρίς καμία επίδραση στον δίαυλο. Για παράδειγμα, η αλλαγή στην είσοδο ενός πολυπλέκτη 2:1, συνεπάγεται την διαφοροποίηση του πίνακα αληθείας του LUT.

c. *Συγκρουόμενα* (Conflict): πρόκειται για μία ειδική κατηγορία η οποία καθορίζεται από την σύνδεση δύο ή περισσοτέρων καλωδιώσεων. Η σύγκρουση που προκύπτει, οδηγεί σε βραχυκύκλωμα μεταξύ της τροφοδοσίας και της γείωσης. Το αποτέλεσμα είναι δύσκολο να προβλεφθεί, παρεκτός και είναι γνωστό το λεπτομερές σχέδιο του FPGA. Η σύγκρουση μπορεί να λάβει χώρα σε διασταυρώσεις διαύλων, σε πολυπλέκτες όταν περισσότερες των μία εισόδων επιλεχθεί κ.τ.λ. Η σύγκρουση χωρίζεται σε δύο υποκατηγορίες:

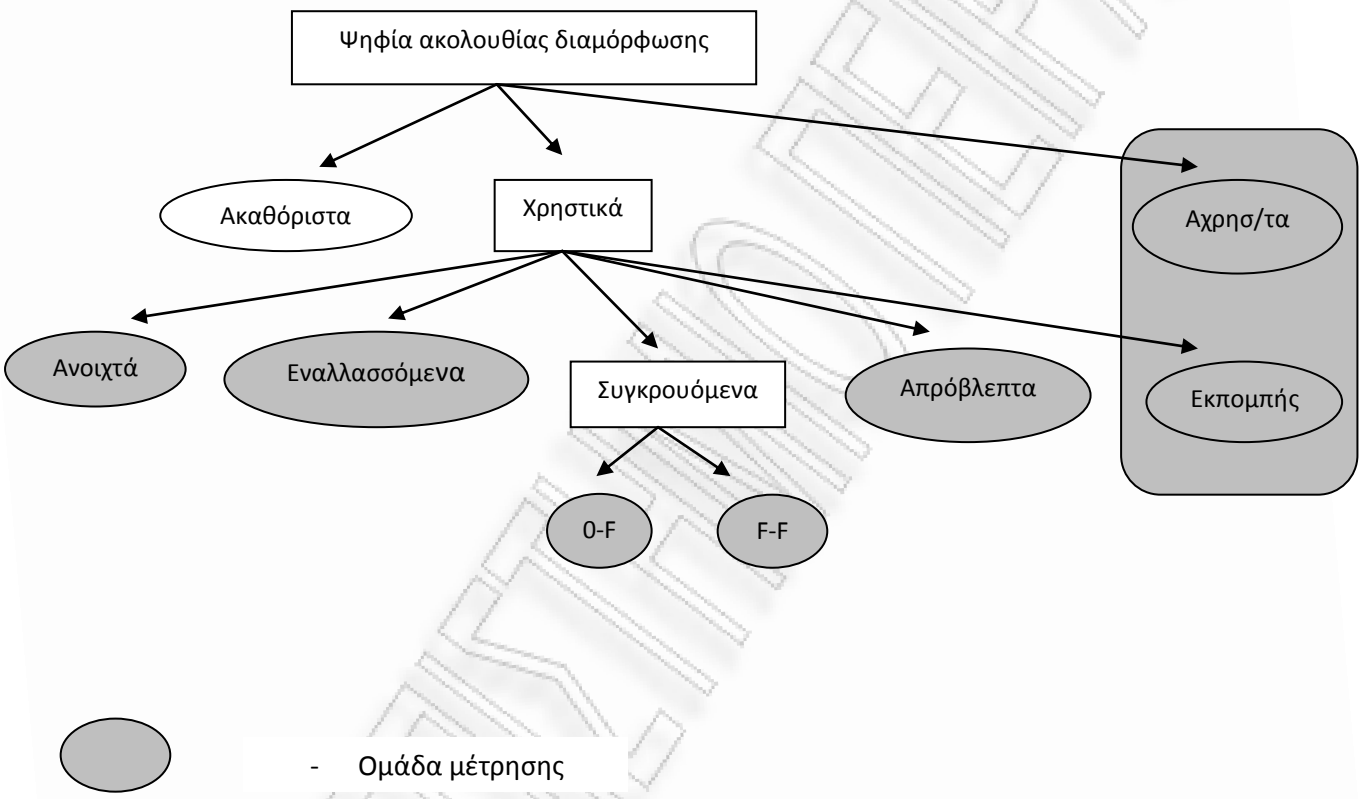
   i. "*F-F*" (Function conflicts with Function): σύγκρουση μεταξύ συναρτήσεων.

   ii. "*0-F*" (constant logical 0 conflicts with Function): σύγκρουση συνάρτησης με σημείο σταθερού μηδέν.

d. *Απρόβλεπτα* (Unpredictable): πρόκειται για μία ειδική περίπτωση των ανοιχτών σφαλμάτων, όπου η αλλαγή της κατάστασης ενός ψηφίου συνεπάγεται αλλαγή του επιλογέα (σε πύλη μεταφοράς ή πολυπλέκτη) από μία μόνιμη τιμή (0 – 1), σε ασύνδετη κατάσταση (Z).

e. *Εκπομπής* (Antenna): συμβαίνει όταν ένα αχρησιμοποίητο καλώδιο συνδεθεί στο μονοπάτι των δεδομένων. Τέτοιου είδους σφάλματα στατιστικά δεν επηρεάζουν το τελικό κύκλωμα, αλλά συνεπάγονται καθυστερήσεις λόγω της επιπλέον χωρητικότητας φορτίου που προστίθεται.

3. Τα *ακαθόριστα* (unknown) σφάλματα προξενούνται από άγνωστες καταστάσεις που επέρχεται η καλωδίωση ή από ελλιπής πληροφορίες που έχουμε για τον σχεδιασμό και δεν μπορούν να αξιολογηθούν ως προς την σπουδαιότητά τους.



Εικόνα 2.1: Ομάδες σφαλμάτων

3. Τεχνικές Άμβλυνσης Προσωρινών Σφαλμάτων

Οι αιτίες πρόκλησης των SEUs είναι τέτοιες, που είναι αδύνατο να εξαλειφθούν από το περιβάλλον λειτουργίας των συσκευών. Για να αντιμετωπιστούν λοιπόν, υιοθετούνται μέθοδοι από τους σχεδιαστές των εφαρμογών, ώστε τα κυκλώματα να αντιδρούν σε πιθανές διαταραχές και να μην επιτρέπουν την απόκλιση από την ορθή τους λειτουργία. Η πρώτη προσπάθεια εξομάλυνσης, που εφαρμόστηκε και χρησιμοποιήθηκε κατά κόρον, ήταν η χρήση θωράκισης των κυκλωμάτων, με την οποία περιορίστηκε η εισροή των μορίων της κοσμικής ακτινοβολίας στις συσκευές, χωρίς όμως να παρακωλυθεί ολοκληρωτικά. Η λύση επαρκούσε τα πρώτα χρόνια, αλλά κατέστη ασθενής, καθώς τα ολοκληρωμένα κυκλώματα περιορίζονταν σε μέγεθος και αύξαναν σε πυκνότητα εξαρτημάτων.

Όσον αφορά τα FPGAs, οι τεχνικές άμβλυνσης που προτάθηκαν, ακολούθησαν δύο διαφορετικές πορείες. Η πρώτη προσέγγιση αφορούσε τον εκ νέου σχεδιασμό της μήτρας του FPGA, η οποία θα αποτελείται εξ' ολοκλήρου, από στοιχεία με ανοσία στα σφάλματα. Αυτά τα νέα στοιχεία, θα μπορούν να αντικαταστήσουν την μέχρι τώρα χρησιμοποιούμενη δομή, υπό την ίδια ή διαφορετική αρχιτεκτονική. Η λύση αυτή όμως είναι ακριβή και δεν βοηθά στον υπολογισμό των ωρών ανάπτυξης και των απαιτούμενων μηχανικών, στοιχεία ουσιώδη για την έναρξη δημιουργίας μιας οποιασδήποτε εμπορικής εφαρμογής. Στον αντίποδα, προτάθηκαν μέθοδοι με στόχο την προστασία σε υψηλό επίπεδο, με την χρήση κάποιου είδους πλεονασμού (χρονικού ή υλικού), σε πολύ συγκεκριμένες περιοχές της σχεδίασης και όχι σε όλη την συσκευή. Ο τρόπος χρήσης του πλεονασμού και οι περιοχές εφαρμογής του, καθορίζονται αυστηρά από την εφαρμογή σε συνδυασμό με την συσκευή που υλοποιείται και δεν υπάρχει μία λύση για όλα. Υπάρχει η δυνατότητα επιλογής της καταλληλότερης μεθόδου για το κάθε κύκλωμα και η δυνατότητα ελιγμού ανάμεσα σε διαφορετικές τεχνικές, με στόχο το βέλτιστο αποτέλεσμα. Με αυτό τον τρόπο μπορούν να χρησιμοποιηθούν τα εμπορικά FPGAs και να τους εισαχθούν οι τεχνικές άμβλυνσης προτού διαμορφωθούν, δηλαδή στον αρχικό σχεδιασμό. Το κόστος της μεθόδου είναι σαφώς μικρότερο, από το προηγούμενο, αλλά η ευθύνη της ικανοποιητικής ανοχής στα σφάλματα βαραίνει τον σχεδιαστή.

Οι τεχνικές άμβλυνσης που έχουν προταθεί τα τελευταία χρόνια, ώστε να αποφευχθούν τα σφάλματα στα ψηφιακά κυκλώματα, συμπεριλαμβανομένου και της προγραμματιζόμενης λογικής, κατηγοριοποιούνται ως εξής:

1. Τεχνικές κατασκευής, όπως είναι:
    a. Επιταξιακή κατασκευή CMOS (epitaxial CMOS).
    b. Πυρίτιο σε μονωτή (silicon-on-insulator, SOI).
2. Τεχνικές βασισμένες στο σχεδιασμό, όπως:
    a. Τεχνικές ανίχνευσης:
        i. Πλεονασμός υλικού (hardware redundancy).
        ii. Πλεονασμός χρόνου (Time redundancy).
        iii. Κώδικας ανίχνευσης λαθών (Error Detection Code - EDC).

iv. Τεχνικές αυτοελέγχου (self-checking).

b. Τεχνικές άμβλυνσης

i. Τριπλός αρθρωτός πλεονασμός (Triple Modular Redundancy - TMR).

ii. Πολλαπλός πλεονασμός με επιλογή.

iii. Κώδικας ανίχνευσης και διόρθωσης σφαλμάτων (Error detection and correction coding - EDAC).

iv. Εύρωστο επίπεδο κελιών μνήμης.

3. Τεχνικές ανάκαμψης (μόνο στις περιπτώσεις της προγραμματιζόμενης λογικής), όπως:

a. Επαναδιαμόρφωσηreconfiguration)..

b. Μερική διαμόρφωση (partial reconfiguration).

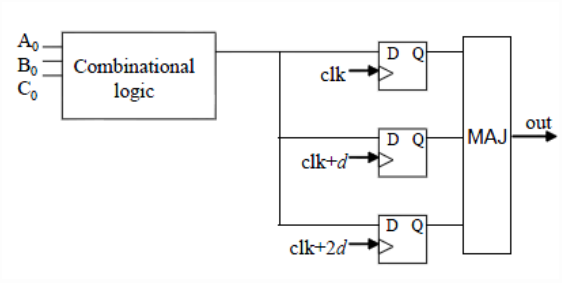
c. Επαναδρομολόγηση του κυκλώματος .

Οι τεχνικές που επικεντρώνονται στην κατασκευαστική διεργασία, μπορούν να περιορίσουν τα τυχαία σφάλματα, αλλά δεν είναι δυνατόν να εξαλείψουν διαταραχές προερχόμενες από SEUs. Περισσότερο αποδεκτές είναι οι τεχνικές σχεδίασης ή οι τεχνικές αρχιτεκτονικής, γιατί εφαρμόζονται σε διαφορετικά επίπεδα λογικής του κυκλώματος, χωρίς κατασκευαστική επέμβαση. Είθισται βέβαια, να εισάγουν κάποιου είδους υλικό ή να απαιτούν περισσότερο χρόνο εκτέλεσης, αλλά είναι εξαιρετικά αξιόπιστες. Η εύρεση της κατάλληλης τεχνικής αποτελεί πρόκληση μιας και πρέπει να ικανοποιεί την ταχίστη απόκριση κατά την εμφάνιση του σφάλματος, να είναι οικονομική, αποδοτική και αξιόπιστη. Οι τεχνικές άμβλυνσης για τα SRAM-based FPGAs πρέπει να εξασφαλίζουν χαμηλό κόστος σε σχέση με:

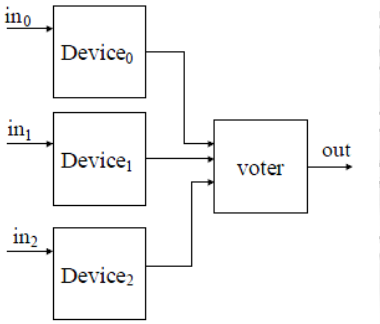
- τον χρόνο προς την αγορά
- την ανάπτυξη
- την απαιτούμενη υψηλή απόδοση
- την κατασκευή
- την ενεργειακή κατανάλωση
- την απαιτούμενη υψηλή αξιοπιστία

3.1 Πλεονασμός χρόνου και υλικού

Στις περιπτώσεις πλεονασμού, είτε χρόνου είτε υλικού, η βασική ιδέα είναι η χρήση επιπλέον κυκλώματος, το οποίο λαμβάνει την έξοδο μιας εφαρμογής ή ενός κρίσιμου σημείου αυτής, σε διαφορετικές χρονικές στιγμές ή μέσω διαφορετικών κυκλωμάτων και με την βοήθεια επιλογέα, αποδίδει την ορθή τιμή στην έξοδο(Εικόνα 3.1 και 3.2). Οι τεχνικές αυτές αυξάνουν το τελικό κύκλωμα και εισάγουν χρονική καθυστέρηση, αλλά επιτυγχάνουν την διόρθωση οποιουδήποτε SEU εμφανιστεί στο αρχικό κύκλωμα. Αν όμως το σφάλμα διαταράξει το κύκλωμα του επιλογέα, τότε οι μέθοδοι καθίστανται αναποτελεσματικές. Η ικανότητά τους, επίσης περιορίζεται στην διόρθωση **ενός** σφάλματος την φορά και



Εικόνα 3.1: Χρονικός Πλεονασμός



Εικόνα 3.2: TMR για ολόκληρη την συσκευή

άρα στην περίπτωση της παρουσίας πολλαπλών σφαλμάτων, οδηγούν σε συσσώρευση. Για την επίλυση της συσσώρευσης προτάθηκαν μέθοδοι που εισάγουν την χρήση τριών επιλογέων με ανάδραση [21,22].

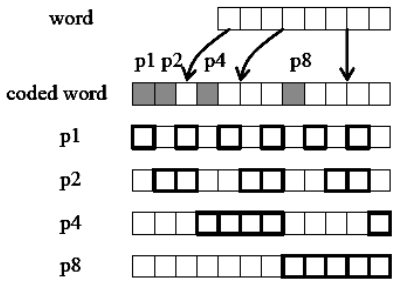
Μία πιο αποτελεσματική μέθοδος απομάκρυνσης των SEUs από την συνδυαστική λογική, βασίζεται στον διπλασιασμό και στην διατήρηση της λέξης κώδικα (code word state preserving – CWSP). Στην προκειμένη περίπτωση δεν απαιτούνται επιλογείς ή συγκριτές. Στην τεχνική πρωταγωνιστικό ρόλο διαδραματίζει το CWSP στάδιο, το οποίο αντικαθιστά τις τελευταίες πύλες του κυκλώματος με μία συγκεκριμένη δομή πυλών, που επιτρέπει την διέλευση μόνο της ορθής τιμής της λογικής ακόμα και με την παρουσία ενός SEU. Παρόλα αυτά, στα συστήματα που απαιτούνται λύσεις χειρισμού πολλαπλών σφαλμάτων είναι περιπλοκότερα και απαιτείται προσεκτική μελέτη.

3.2 Κώδικας Ανίχνευσης

Η τεχνική EDAC [16] χρησιμοποιείται για την επίλυση προβλημάτων SEU στα ολοκληρωμένα κυκλώματα, κυρίως στις περιοχές μνήμης. Διατίθενται ποικίλοι κώδικες για την προστασία των συστημάτων είτε από απλές, είτε από πολλαπλές διαταραχές. Ένα παράδειγμα του EDAC είναι ο κώδικας hamming [22], ο οποίος ανιχνεύει όλα τα **απλά και διπλά** λάθη στα ψηφία και διορθώνει τα **απλά**. Ο κώδικας δημιουργείται μέσω ενός συνδυαστικού μπλοκ που είναι υπεύθυνο για την κωδικοποίηση των δεδομένων, δηλαδή την προσθήκη ψηφίων στην λέξη για να εξασφαλίσει έλεγχο ισοτιμίας (παραπάνω μανταλωτές ή flip – flops) και ένα ακόμα συνδυαστικό μπλοκ το οποίο είναι υπεύθυνο για την αποκωδικοποίηση των δεδομένων.

Ο κώδικας hamming μπορεί να προστατέψει δομές καταχωρητών και μνημών, αρκεί κάθε καταχωρητής να έχει συνδεδεμένες τις εισόδους του στο μπλοκ κωδικοποίησης και τις εξόδους του στο μπλοκ αποκωδικοποίησης. Σημαντική παρατήρηση είναι ότι μόνο ένας καταχωρητής μπορεί να χρησιμοποιηθεί κατά την διάρκεια ενός κύκλου ρολογιού. Το κυριότερο πλεονέκτημα της μεθόδου είναι ότι απαιτείται μόνο ένας κωδικοποιητής και αποκωδικοποιητής, στους οποίους καταλήγουν οι καταχωρητές κατόπιν πολυπλεξίας. Επιπλέον αύξηση υλικού συνεπάγονται οι επιπρόσθετες μονάδες μνήμης για την αποθήκευση των ψηφίων ελέγχου. Προσεγγιστικά για κάθε n ψηφίο λέξη που κωδικοποιείται απαιτούνται $\log_2 n$ περισσότερα κελιά αποθήκευσης. Παρόλα αυτά, τα δύο βασικά μπλοκ μετέχουν στο κρίσιμο μονοπάτι του κυκλώματος και η καθυστέρηση που εισάγουν αυξάνεται, όσο αυξάνονται τα ψηφία των κωδικοποιημένων λέξεων.

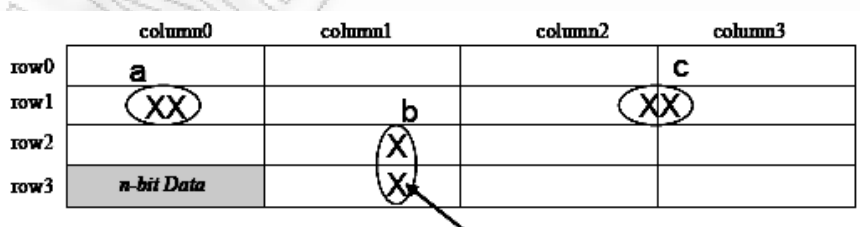
Η αδυναμία επίλυσης διπλών σφαλμάτων, του κώδικα τον καθιστά σχεδόν ανεφάρμοστο, σε



Εικόνα 3.3: Hamming κώδικας για αρχική λέξη 8 ψηφία και κωδικοποιημένη 12 ψηφία.

εφαρμογές με υψηλή πυκνότητα κελιών μνήμης και μικρό μέγεθος κατασκευής [23]. Σε αναζήτηση λοιπόν άλλου τύπου κώδικα που θα υπερπηδούσε αυτό το εμπόδιο, προέκυψε ο Reed-Solomon (RS) [22]. Ο κώδικας αναλαμβάνει να διορθώσει πολλαπλά σφάλματα σε ένα σύστημα και εφαρμόζεται σε πληθώρα εφαρμογών, όπως σε συσκευές αποθήκευσης, ασύρματες και φορητές επικοινωνίες, υψηλής ταχύτητας modems κ.α. Ομοίως, ακολουθεί την λογική κωδικοποίησης και αποκωδικοποίησης με την χρήση ισοτιμίας

Στην περίπτωση της εφαρμογής του στις μνήμες, μία λέξη δεδομένων χωρίζεται σε σύμβολα και κάθε τέτοια λέξη αποτελεί και μια διαφορετική RS κωδικοποιημένη λέξη. Για παράδειγμα, σε μία μνήμη n γραμμών, όπου η λέξη δεδομένων καλύπτει ολόκληρη γραμμή, η τελευταία χωρίζεται σε m σύμβολα, σύμφωνα με το μέγεθος συμβόλου και το μέγεθος της μνήμης. Πολλαπλές διαταραχές μπορούν να συμβούν στο σύστημα, αν και οι πιο πιθανές είναι διπλές αναστροφές σε ψηφία, είτε στο ίδιο σύμβολο (τύπου a), σε κάθετα γειτνιάζοντα σύμβολα (τύπου b), είτε οριζόντια γειτνιάζοντα σύμβολα (τύπου c- Εικόνα 3.4).Η πιο απλή περίπτωση είναι οι διαταραχές τύπου a, εφόσον ανήκουν στο ίδιο σύμβολο. Στο τύπο b, τα λάθη διορθώνονται μεμονωμένα, μιας και κάθε γραμμή αποτελεί και ένα ξεχωριστό σύμβολο. Αλλά τα σφάλματα τύπου c, δεν διορθώνονται από τον κώδικα, μιας και πρόκειται για διαταραχές δύο διαφορετικών συμβόλων, της ίδιας όμως λέξης.

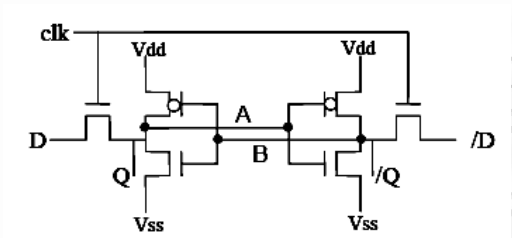


Εικόνα 3.4: Τύποι διπλών διαταραχών σε RS κώδικα

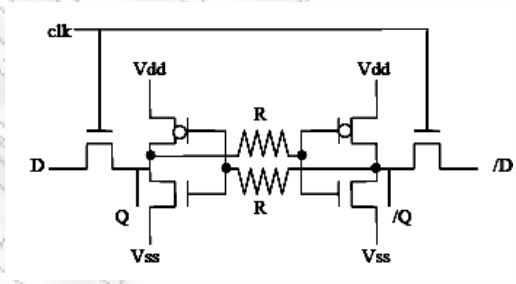
3.3. Εύρωστα κελιά μνήμης

Μία ακόμα τεχνική άμβλυνσης των SEU αποτελεί αυτή της διαμόρφωσης των κελιών μνήμης, με επιπλέον εξαρτήματα, όπως είναι οι αντιστάσεις και τα τρανζίστορ, ώστε να μπορούν να ανακύπτουν από πιθανές διαταραχές, που συμβαίνουν στους απαγωγούς των τρανζίστορ, όταν τα τελευταία είναι ανενεργά και αποδίδουν εσφαλμένα, έξοδο. Ένα τυπικό κελί μνήμης αποτελείται από 6 τρανζίστορ (Εικόνα 3.5). Όταν αποθηκεύει μία τιμή, δύο τρανζίστορ είναι ενεργά (κατάσταση 'on') και δύο ανενεργά (κατάσταση 'off').

Όταν διαταραχθεί ένα από τα τρανζίστορ, η ενέργεια που εκλύεται, μπορεί να αντιστρέψει την κατάστασή του και άρα να αλλάξει την αποθηκευμένη τιμή στο κελί. Αν όμως τοποθετηθεί μία αντίσταση μεταξύ της εξόδου του ενός αναστροφέα και της εισόδου του άλλου, το σήμα καθυστερείται για τόσο χρόνο, όσος απαιτείται για να μην αντιστραφεί το αποθηκευμένο ψηφίο (Εικόνα 3.6), [24]. Τα βασικότερα μειονεκτήματα της μεθόδου είναι η θερμική ευαισθησία και η ανάγκη δημιουργίας επιπλέον μάσκας κατασκευής για τις αντιστάσεις.



Εικόνα 3.5: Τυπικό κελί μνήμης



Εικόνα 3.6: Εύρωστα κελιά μνήμης μέσω αντίστασης

Τα κελιά μνήμης μπορούν να προστατευθούν επίσης μέσω ανάδρασης ή οποία θα εξασφαλίζει την άμεση διόρθωση κατόπιν σύγκρουσης με ιόν. Τροχοπέδη αποτελεί η πρόσθεση τρανζίστορ για να εξασφαλιστεί η ανάδραση και άρα η ύπαρξη νέων ευαίσθητων σημείων στην περιοχή. Τα πιο χαρακτηριστικά παραδείγματα εφαρμογής της μεθόδου είναι τα IBM εύρωστα κελιά [25], τα κελιά HIT [26,27,28] και τα κελιά Canaris [11,29] .Η μέθοδος δεν επηρεάζεται από τις αλλαγές της θερμοκρασίας και την τροφοδοσία και παρέχει ικανοποιητική ανοσία στα SEUs. Το βασικότερο μειονέκτημά της είναι η μεγάλη εξάπλωση σε επίπεδο πυριτίου, λόγω των επιπλέον τρανζίστορ και του μεγέθους τους. Επίσης, ως λύση άμβλυνσης προτείνεται και η αποθήκευση του ψηφίο, σε δύο διαφορετικά μέρη μέσα στο κελί, ώστε το εσφαλμένο τμήμα να μπορεί να ανακτηθεί. Χαρακτηριστικά παραδείγματα είναι τα DICE [11]και τα NASA κελιά [30,31]. Η μέθοδος διατηρεί τα ανωτέρω πλεονεκτήματα, ανοσίας στην θερμοκρασία και τη τάση καθώς και υψηλή απόδοση ανάγνωσης και εγγραφής.

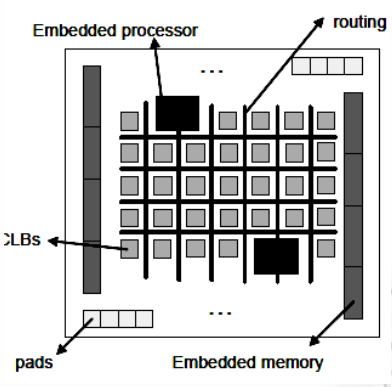
Ακολουθεί συγκριτικός πίνακας των τεχνικών άμβλυνσης των SEUs [32].

SEU Τεχνικές Άμβλυνσης	Εύρωστα κελιά μνήμης	Hamming Code	TMR
Επέκταση της κατασκευαστικής περιοχής	Εισάγει διπλασιασμό της περιοχής του κυκλώματος. Εξαρτάται άμεσα από το μέγεθος των τρανζίστορ	Εξαρτάται από τον αριθμό των ψηφία που θα προστατεύει. Εισάγει επιπλέον συνδυαστική και ακολουθιακή λογική.	Τριπλασιάζει την περιοχή εφαρμογής, εξαιτίας του επιλογέα.
Απόδοση	Η απόδοση δεν επηρεάζεται εάν τα επιπλέον τρανζίστορ και οι αντιστάσεις ενεργούν μόνο σε λειτουργία αναμονής.	Ο κωδικοποιητής και αποκωδικοποιητής επηρεάζουν την απόδοση.	Η απόδοση δεν επηρεάζεται σημαντικά. Η μόνη πηγή καθυστέρησης είναι ο επιλογέας.
Διόρθωση Σφαλμάτων	Απομακρύνει το λάθος μέσω καθυστέρησης στον βρόγχο της μνήμης (πλεονασμός/ανάκτηση)	Τυπικά, διορθώνει ένα σφάλμα ανά λέξη, αλλά για να εξασφαλίσει ανανέωση, απαιτεί ένα επιπλέον μονοπάτι.	Δεν διορθώνει τα σφάλματα, τα οποία συσσωρεύονται αν δεν υπάρχει επιπλέον λογική ανανέωσης.
Πολλαπλές Διαταραχές	Ικανά για διόρθωση σφαλμάτων τρίτου επιπέδου μιας και κάθε κελί αυτό-προστατεύεται.	Αναποτελεσματικός για πολλαπλά λάθη, στην ίδια κωδικοποιημένη λέξη. Αποδοτικός, για πολλαπλά σφάλματα σε διαφορετικά μονοπάτια του κυκλώματος.	Ανταπεξέρχεται σε πολλαπλές διαταραχές που βρίσκονται σε διαφορετικά μέρη του κυκλώματος , αλλά όχι ταυτόχρονα, σε ένα TMR σήμα.
Τεχνολογία	Μπορεί να χρησιμοποιήσει μεγαλύτερη περιοχή λόγω της ασυμμετρίας των τρανζίστορ και των μεγάλων αντιστάσεων.	Πλήρως συμβατό με την τεχνολογία CMOS.	

Πίνακας 3.1: Σύγκριση μεθόδων άμβλυνσης

3.4. Τεχνικές Άμβλυνσης Αρχιτεκτονικής για FPGAs βασιζόμενα σε SRAM

Σύμφωνα με τα προαναφερόμενα, οι προγραμματιζόμενες συσκευές αποτελούνται από μία πληθώρα εξαρτημάτων, όπως περίπλοκα λογικά μπλοκ με LUTs, πολυπλέκτες και flip-flops, ενσωματωμένες



Εικόνα 3.7: Αρχιτεκτονική FPGA

μνήμες και PLLs, εκ των οποίων άλλα παρουσιάζουν μεγαλύτερη και άλλα μικρότερη ευαισθησία στα τυχαία σφάλματα (Εικόνα 3.8). Επιπρόσθετα, τα τωρινά FPGAs, με στόχο την βελτίωση της απόδοσης και της ικανοποιητικής επεξεργασίας δεδομένων, τοποθετούν στο τσιπ τόσο ενσωματωμένους (soft) όσο και διακριτούς (hard) μικροεπεξεργαστές. Έτσι, στα νέα FPGAs, το πρόβλημα της άμβλυνσης των SEUs μπορεί να αναλυθεί υπό δύο διαφορετικές σκοπιές: από την μεριά του μικροεπεξεργαστή και από την μεριά της προγραμματιζόμενης λογικής. Κάθε δομή χωρίζεται σε μέρη, ανάλογα με την λειτουργία και τα χαρακτηριστικά που παρουσιάζουν και κάθε μέρος εξετάζεται ως προς την ευαισθησία του και την δυνατότητα προστασίας του.

Το πρόβλημα της προστασίας των μικροεπεξεργαστών από τα SEUs είναι κατανοητό και οι διαθέσιμες μέθοδοι άμβλυνσης μπορούν να εφαρμοστούν, ώστε να επιτευχτεί η αξιόπιστη λειτουργία της κατασκευής. Ο συνδυασμός hamming κώδικα, για τους καταχωρητές και την εσωτερική μνήμη [33,34] και η ανανέωση κάποιων μερών του κυκλώματος, κυρίως στον κομμάτι της μνήμης, ώστε να αποφευχθεί η συσσώρευση σφαλμάτων, απεδείχθησαν ικανά για να δημιουργήσουν ένα εύρωστο σύστημα, με ανοχή σε απλές και πολλαπλές διαταραχές [35,36].

Παρόλα αυτά, στην περίπτωση των FPGAs βασιζόμενων σε SRAM, το πρόβλημα της αναζήτησης τεχνικής, ικανοποιητικώς αποδοτικής, υπό το πρίσμα της βελτίωσης της δομής, της τελικής απόδοσης και της τροφοδοσίας, παρουσιάζει δυσκολίες, λόγω της υψηλής περιπλοκότητας της αρχιτεκτονικής τους. Όταν μια διαταραχή λαμβάνει χώρα στην συνδυαστική λογική ενός FPGA, συνεπάγεται μία βραχύβια αντίδραση, που όμως ακολουθείται από ένα μόνιμο αποτέλεσμα, μιας και ουσιαστικά πρόκειται για έναν μακρύ καταχωρητή ολίσθησης. Η διαταραχή μπορεί να επηρεάσει τόσο την συνδυαστική λογική, όσο και την δρομολόγηση εντός της συσκευής. Οι επιπτώσεις ενός τέτοιου φαινομένου, δεν μπορούν να αντιμετωπιστούν από τις προαναφερόμενες τεχνικές όπως είναι η EDAC, ο κώδικας hamming ή η TMR, μιας και ένα σφάλμα στον κωδικοποιητή ή αποκωδικοποιητή ή στον επιλογέα, θα σήμαινε και την ολοκληρωτική αποτυχία της μεθόδου προστασίας.

Στο επίπεδο της αρχιτεκτονικής, οι προαναφερόμενες τεχνικές αφήνουν τις εξής εκκρεμότητες:

- 19Πώς θα αντιμετωπιστούν τα SEUs στα λογικά μπλοκ διαμόρφωσης (Configurable Logic Block – CLB), ώστε να αποφευχθεί η αποθήκευση των σφαλμάτων στα flip-flops.
- Πώς θα αντιμετωπιστούν οι πολλαπλές διαταραχές σε επίπεδο LUTs, δρομολόγησης και ειδικότερα ενσωματωμένης μνήμης.

Αρχικά, πρέπει να καθοριστούν οι ευαίσθητες περιοχές στην προγραμματιζόμενη μήτρα και τα χαρακτηριστικά τους, ώστε να προταθούν οι βελτιώσεις στις τεχνικές άμβλυνσης. Ο πίνακας 3.2 δείχνει τα σετ των κελιών διαμόρφωσης σε ένα CLB, για τα FPGAs της οικογένειας VIRTEX της XILINX. Αναλύοντας το ποσοστό για κάθε τύπο κελιού SRAM σε σχέση με το μέγεθος όλων των στοιχείων της μνήμης στα CLBs, τα LUTs αντιπροσωπεύουν το 7,4%, τα flip-flops το 0,46%, τα ψηφία διαμόρφωσης μέσα στο CLB το 6,36% και η γενική δρομολόγηση το 82,9%.

Βασιζόμενοι σε αυτά τα ποσοστά, οι επιπτώσεις μιας διαταραχής στο πεδίο της δρομολόγησης, δείχνει να είναι το βασικότερο πρόβλημα, μιας και αφενός καταλαμβάνει σημαντικό ποσοστό επί του συνόλου των στοιχείων και αφετέρου συνεπάγεται μόνιμη βλάβη, όπως ένα βραχυκύκλωμα ή ένα ανοιχτοκύκλωμα, στις τελικές συνδέσεις του λογικού σχεδιασμού. Λύσεις λοιπόν, που αυξάνουν την επιφάνεια αυτής της λογικής διαμόρφωσης, δεν ενδείκνυνται και επιπλέον είναι ακριβές

Κατηγοριοποίηση	**Αριθμός ψηφίων διαμόρφωσης (Ποσοστό)**
Πολυπλεξία εξόδου (Output mux)	56 (6,48%)
CLB μέρη	125 (14,47%)
Απομονωτές τριών καταστάσεων (3-state buffers)	14 (1,62%)
Πολυπλεξία εισόδου (Input mux)	231 (26,74%)
Απλή μήτρα (Single matrix)	304 (35,18%)
Δεκαεξαδική μήτρα (Hex matrix)	112 (12,96%)
Μη διαθέσιμα (Not available)	22 (2,55%)

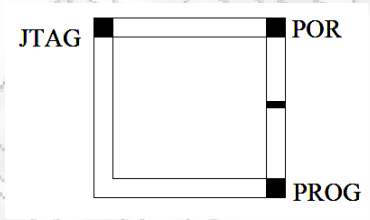
Πίνακας 3.2: Ο αριθμός των ψηφίων διαμόρφωσης σε ένα CLB[32]

Εκτός από τα προγραμματιζόμενα λογικά κελιά, υπάρχουν και άλλα στοιχεία μνήμης στο FPGA που μπορούν να διαταραχθούν από τα τυχαία σφάλματα:

- Οι θύρες επιλογής πρόσβασης μικροεπεξεργαστή (Selectable Microprocessor Access Port - SelectMap).

- Η σύνδεση του καταχωρητή ολίσθησης των SRAM κελιών με την αλυσίδα σάρωσης (Joint Test Action Group – JTAG- IEEE Std. 1149.1x)
- Οι θύρες δοκιμής πρόσβασης (TestAccess Port - TAP).
- Επιπρόσθετοι μανταλωτές από ποικίλα ενσωματωμένα μη-προγραμματιζόμενα χαρακτηριστικά, όπως είναι η περιοχή φόρτωσης της ακολουθίας των ψηφίων διαμόρφωσης και ο έλεγχος της επαναφοράς της συσκευής κατά την εκκίνηση (power-on-reset - POR).

Τα τυχαία σφάλματα, στα προαναφερόμενα, ονομάζονται SEFI και αναφέρονται σε φαινόμενα διαταραχής του κυκλώματος που διαμορφώνει την συσκευή και του JTAG. Επίσης, στο POR κύκλωμα υπάρχουν flip-flops ή μανταλωτές (< 40) που αν επηρεαστούν προκαλούν μικρές διακοπές, οι οποίες δεν μπορούν να παραβλεφθούν μιας και μπορεί να συνεπάγονται ενδεχόμενη επαναδιαμόρφωση. Στην Εικόνα 3.8 φαίνεται η τοποθέτηση αυτών των flip-flops στο FPGA matrix. Προτεινόμενες λύσεις για την προστασία του POR είναι η χρήση TMR για όλο το μπλοκ ή η αντικατάσταση των κελιών με εύρωστα κελιά ή η χρήση πλεονάζουσας λογικής, ώστε να απενεργοποιείτε το POR μετά τον προγραμματισμό της συσκευής, μέσω ενός εσωτερικού ακροδέκτη (pin). Η λύση των εύρωστων κελιών θεωρείται καταλληλότερη για την αντικατάσταση των SRAM κελιών στο κομμάτι της δρομολόγησης, της γενικής διαμόρφωσης και των LUTs, μιας και παρουσιάζουν μικρή κατασκευαστική εξάπλωση, σε σχέση με τον λογικό πλεονασμό και την EDAC. Η τελική περιοχή θα είναι περίπου διπλασιασμένη, κάτι που θεωρείται ικανοποιητικό, υπό το πρίσμα της υψηλής αξιοπιστίας



Εικόνα 3.8: Ειδικά στοιχεία στην FPGA μήτρα

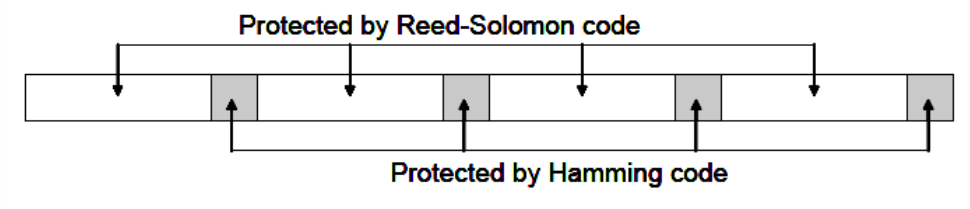
3.4.1.Μέθοδος EDAC

Ειδική μνεία γίνεται για την προστασία της ενσωματωμένης μνήμης στα FPGAS, η οποία παρουσιάζει μεγάλη ευαισθησία. Προτεινόμενη μέθοδος είναι η EDAC που θεωρείται κατάλληλη, αν εφαρμόσει κώδικα που θα προσφέρει ανοσία από πολλαπλές διαταραχές ψηφίων (multiple bit upsets – MBU) για δύο κύριους λόγους:

- Οι ιδιότητες των πολύ μικρών τεχνολογιών κατασκευής SRAM (Very Deep Sub Micron - VDSM,) είναι ευαίσθητες σε MBU.
- Η διαδικασία scrubbing δεν επαναδιαμορφώνει την εσωτερική μνήμη, με αποτέλεσμα να αυξάνεται η πιθανότητα συσσώρευσης σφαλμάτων.

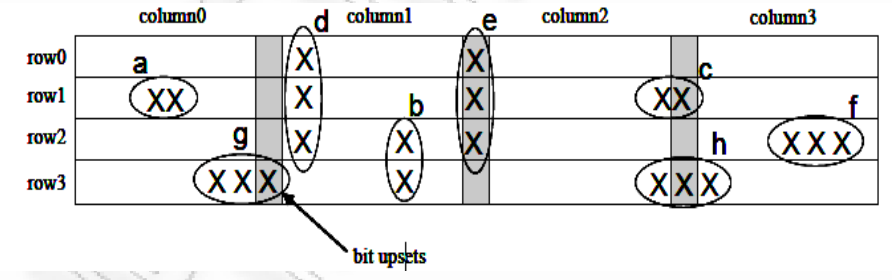
Κατά συνέπεια, γεννήθηκε η ανάγκη για έναν νέο κώδικα που θα διόρθωνε τα πιθανά διπλά σφάλματα. Η αρχική επιλογή ήταν ο κώδικας RS, μιας και διορθώνει έως 2 λάθη σε διαφορετικές λέξεις. Πρόβλημα όμως αποτελεί ο διπλασιασμός της περιοχής του κυκλώματος και η υψηλή καθυστέρηση που εισήγαγε [22], κάτι που τον καθιστά ακατάλληλο για κυκλώματα μνήμης. Επιπρόσθετα, στις τεχνολογίες VDSM η ύπαρξη δύο λαθών στην ίδια λέξη είναι εξαιρετικά πιθανή και άρα ο κώδικας είναι ανεφάρμοστος.

Σ' αυτήν λοιπόν την κατεύθυνση αναπτύχθηκε μια τεχνική, που διορθώνει διπλά λάθη σε όλες



Εικόνα 3.9: Σχηματική απεικόνιση μιας γραμμής μνήμης προστατευμένης από τους κώδικες RS και hamming

τις VDSM μνήμες. Η μέθοδος συνδυάζει τους κώδικες hamming και RS με δυνατότητα διόρθωσης απλών συμβόλων, ώστε να επιτυγχάνεται η 100% επίλυση του προβλήματος των διπλών σφαλμάτων, με την χρήση του χαμηλού κόστους RS. Ουσιαστικά, ο κώδικας hamming προστατεύει τα ψηφία μεταξύ των RS συμβόλων. Ο αριθμός των ψηφίων που προστατεύει ο τελευταίος, είναι ο ίδιος με τον αριθμό των συμβόλων που προστατεύει ο RS, ώστε να μην αυξάνεται σημαντικά το κύκλωμα. Στην εικόνα 3.9 φαίνεται η εισαγωγή του hamming σε ήδη κωδικοποιημένη λέξη υπό RS [37].Η μέθοδος αποδείχτηκε



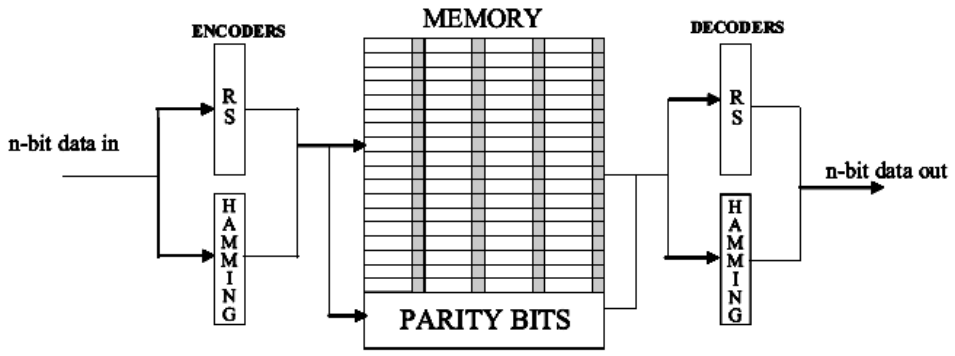
Εικόνα 3.10: γραμμές μνήμης προστατευόμενες από τους κώδικες RS και hamming

εξαιρετικά αποτελεσματική, υπό την παρουσία απλών, διπλών και πολλαπλών με μόνη εξαίρεση (τύπος h – Εικόνα 3.10.) στην περίπτωση που τρία ή περισσότερα σφάλματα εμφανίζονται σε δύο διαφορετικά RS σύμβολα. Ο RS κωδικοποιητής μπορεί να ενσωματωθεί για οποιοδήποτε μέγεθος μνήμης.

Η περίπτωση εφαρμόστηκε σε VHDL και πρωτοτυποποιήθηκε σε Virtex®E FPGA, χρησιμοποιώντας μπλοκ σύγχρονης μνήμης RAM (BlockRAM) και CLBs ώστε να αξιολογηθεί κατά περιοχή, απόδοση και ανοχή σφαλμάτων. Τα αποτελέσματα φαίνονται στον πίνακα 3.3.

	16-ψηφίο Hamming		112-ψηφίο RS	
	Κωδικοποιητής	Αποκωδικοποιητής	Κωδικοποιητής	Αποκωδικοποιητής
# 4-LUTs	22	99	215	538
# Επιπλέον flip-flops	5 x # σειρών		14 x # σειρών	
Καθυστέρηση (ns)	9,3	21,7	14,5	47,6

Πίνακας 3.3: Μέγεθος και καθυστέρηση για τους κώδικες RS και hamming κατά την προστασία μνήμης [32]



Εικόνα 3.11: Αρχιτεκτονική μνήμης υπό τους κώδικες RS και hamming

Σύμφωνα με τα αποτελέσματα, φαίνεται ότι η ανοχή της μνήμης στα σφάλματα οδηγεί σε αύξηση του κυκλώματος, λόγω της περιοχής που χρησιμοποιείται από του κωδικοποιητές και αποκωδικοποιητές. Η αύξηση αυτή είναι της τάξης των δύο μπλοκ σύγχρονης RAM (BlockRam-BRAM), ένα για την αποθήκευση των RS συμβόλων και ένα για hamming ψηφία. Η ποινή στην απόδοση αγγίζει το 50%, αν και μπορεί να μειωθεί αν ο κωδικοποιητής και αποκωδικοποιητής ενσωματωθούν με την χρήση τυχαίας λογικής και όχι CLBs, όπως έγινε στο πρωτότυπο.

Συνοψίζοντας, η τεχνική συνδυασμού των κωδικών RS και hamming, ώστε να προστατευτεί η μνήμη από τα SEUs αποτελεί μια ελκυστική λύση που μπορεί να εφαρμοστεί στα εύρωστα κελιά των SRAM-based FPGA. Η μνήμη προστατεύεται από διπλές και πολλαπλές διαταραχές, χωρίς να αυξάνεται υπέρμετρα η περιοχή του κυκλώματος και χωρίς να υποσκάπτει την ομαλή του λειτουργία και διαμόρφωση των κελιών ενσωματωμένης μνήμης.

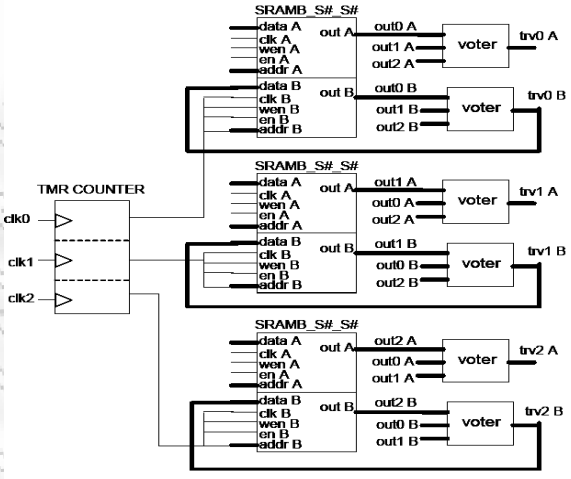
3.4.2. Μέθοδος TMR

Οι μέχρι τώρα εξεταζόμενες τεχνικές που προτάθηκαν, για την αντιμετώπιση των σφαλμάτων στα FPGAs, αν και εξασφαλίζουν υψηλή αξιοπιστία, εισάγουν και αύξηση του τελικού κόστους. Η αλλαγή

ενός κυκλώματος συνεπάγεται την περαιτέρω μελέτη της ανάπτυξης, εκτενέστερες δοκιμές και σαφώς μεγαλύτερες κατασκευές. Οικονομικότερη λύση είναι μία τεχνική άμβλυνσης υψηλού επιπέδου, που εύκολα ενσωματώνεται στο κύκλωμα τόσο από τον χρήστη, όσο και από τους εταιρικούς σχεδιαστές, στα εμπορικά FPGAs ή σε εξαρτήματα που κατασκευάζονται με τεχνολογία, η οποία μπορεί να αποφύγει την μανδάλωση. Η τεχνική συντέθηκε σε VIRTEX αρχιτεκτονική και βασίζεται κυρίως σε μέθοδο TMR συνδυασμένη με scrubbing [38,10]. Η TMR τεχνική άμβλυνσης στην προκειμένη περίπτωση χρησιμοποιεί τρία πανομοιότυπα λογικά κυκλώματα (μπλοκ 0- 2), εντός του FPGA, τα οποία πραγματοποιούν την ίδια λειτουργία ακολουθιακά (το ένα μετά το άλλο) και με τις εξόδους τους να συγκρίνονται εντός ενός μεγάλου επιλογέα [38, 39].

Η σωστή ενσωμάτωση του TMR κυκλώματος σε VIRTEX αρχιτεκτονική εξαρτάται από τον τύπο των δεδομένων που θα υποστούν επεξεργασία. Σκοπός της χρήσης του TMR είναι η απομάκρυνση όλων των **απλών** σημείων αποτυχίας του κυκλώματος. Αυτό εξασφαλίζεται αρχικά με τις εισόδους του FPGA. Αν μία απλή (μονή) είσοδος συνδεόταν και με τα τρία πλεονάζοντα μονοπάτια που τοποθετήθηκαν εντός FPGA, τότε μία αστοχία της θα μεταδιδόταν σε όλο το κύκλωμα και άρα το λάθος δεν θα απομακρυνόταν. Άρα, κάθε πλεονάζον μονοπάτι, το οποίο χρησιμοποιεί τις εισόδους του FPGA, πρέπει να διαθέτει ένα δικό του σετ εισόδων, ώστε σε ενδεχόμενη αστοχία να επηρεαστεί ένα εκ των τριών μονοπατιών. Επίσης, μιας και η TMR τριπλασιάζει κάθε λογικό μονοπάτι, πρέπει να υπάρχει σχεδιασμός, ώστε τα τρία αυτά μονοπάτια να ξαναγίνουν ένα, όπως θα έπρεπε να είναι αρχικά. Αυτό επιτυγχάνεται με επιλογείς που τοποθετούνται εντός του λογικού μπλοκ εξόδου.

Η αρχιτεκτονική VIRTEX παρέχει έναν αριθμό ειδικών χαρακτηριστικών, όπως είναι οι BRAMs, οι βρόγχοι σταθερής καθυστέρησης (Delay Locked Loops – DLLs) κ.α, τα οποία απαιτούν συγκεκριμένες μεθόδους για να ενσωματώσουν αποτελεσματικά πλεονάζουσα λογική. Μία αξιόπιστη μέθοδος εφαρμογής της τεχνικής TMR σε BRAMs είναι η συνεχής ανανέωση του περιεχόμενου του BRAM (εικόνα 3.12). Μιας και πρόκειται για μνήμη διπλής εισόδου, η μία θύρα θα μπορούσε να αναλάβει την ανίχνευση λαθών και η άλλη την διόρθωση, κάτι που όμως συνεπάγεται, ότι η μνήμη θα χρησιμοποιηθεί ουσιαστικά ως απλής εισόδου. Για την ανανέωση των περιεχομένων της μνήμης,



Εικόνα 3.12: BRAM TMR με ανάδραση

ενδείκνυται η χρήση μετρητή, ο οποίος κυκλικά θα παράγει τις διευθύνσεις των μνημών, αυξάνοντας την διεύθυνση κάθε n κύκλους. Τα δεδομένα για κάθε διεύθυνση θα υπόκεινται σύγκριση σε επιλογέα, υπό ορισμένη συχνότητα και η τιμή που θα εξάγεται θα είναι και αυτή που θα αποθηκεύεται στα κελιά.

Τέλος, σε έναν τυπικό FPGA σχεδιασμό, πρέπει να χειριστούν αποτελεσματικά και όλα εκείνα τα σήματα που αναφέρονται σε σταθερές τιμές, όπως είναι η τροφοδοσία και η γείωση (VCC, GND), αλλά δεν μπορούν να θεωρηθούν αποκομμένα από το υπόλοιπο κύκλωμα. Όταν τα εργαλεία τοποθέτησης του σχεδίου (Place and Route), συναντούν αυτά τα σήματα, τα ενσωματώνουν στο τελικό κύκλωμα με τέτοιον τρόπο, ώστε να μεγιστοποιείται η χρήση των πόρων της συσκευής. Αυτό επιτυγχάνεται με την δημιουργία κυκλωμάτων συγκράτησης (keeper), στις εισόδους των CLBs και I/O μπλοκ. Τα κυκλώματα συγκράτησης συνυπάρχουν σε σειρά με τα κανάλια δρομολόγησης και τις εισόδους των λογικών μπλοκ. Έτσι, όταν ένα μονοπάτι δρομολόγησης μεταφέρει ένα σήμα, το κύκλωμα συγκράτησης επιτρέπει την διέλευση, ενώ όταν δεν λαμβάνει χώρα δρομολόγηση, διατηρεί την προηγούμενη του τιμή, η οποία καθορίστηκε κατά την έναρξη λειτουργίας του FPGA, μετά τον προγραμματισμό. Όταν ένα λογικό στοιχείο (π.χ. flip-flop), εντός ενός λογικού μπλοκ (CLB ή IOB) απαιτεί την χρήση μιας τέτοιας σταθεράς, η τελευταία προσκομίζεται από το κύκλωμα συγκράτησης, μέσω ενός αχρησιμοποίητου ακροδέκτη του λογικού μπλοκ. Η πολικότητά του μπορεί να επιλεχθεί, μέσω προγραμματιζόμενης αντίστροφης, εντός του μπλοκ. Ένα SEU μπορεί να αλλάξει την κατάσταση του κυκλώματος συγκράτησης, είτε μέσω σύγκρουσης με τα προαναφερόμενα ιόντα, είτε έμμεσα, λόγω στιγμιαίας σύνδεσης με ένα ενεργό μονοπάτι δρομολόγησης. Σε οποιαδήποτε περίπτωση, συνεπάγεται λειτουργική διαταραχή, η οποία μπορεί να εντοπιστεί από την ανάδραση ή να διορθωθεί μέσω *μερικής επαναδιαμόρφωσης*, μέθοδο που προτείνει η XILINX για τις συσκευές της και αποτελεί και την κύρια μέθοδο ενασχόλησης στην παρούσα μεταπτυχιακή διατριβή.

3.5. Μερική Επαναδιαμόρφωση

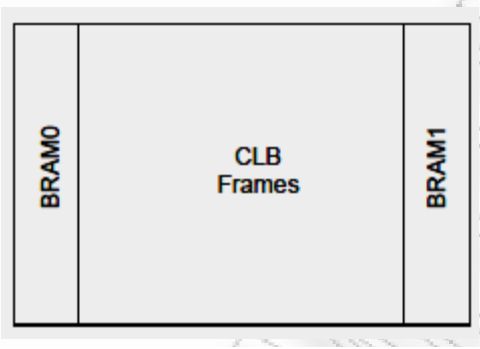
Η χρήση της τεχνικής TMR δεν επαρκεί ως προς την εξασφάλιση της αξιοπιστίας για μεγάλο διάστημα λειτουργία, μιας και διαταραχές μπορεί να συσσωρευτούν στην μήτρα του FPGA, καταλύοντας την μέθοδο. Είναι αναγκαίο να απαλείφονται όλα τα συσσωρευμένα σφάλματα με τέτοια συχνότητα, που θα εγγυάται την ορθή εφαρμογή της TMR τεχνικής. Η XILINX προτείνει την μέθοδο της μερικής επαναδιαμόρφωσης, η οποία εξασφαλίζει λειτουργία εγγραφής στην συσκευή, μετά την διαμόρφωση της, συνδυασμένη με την λειτουργία σάρωσης (readback) που εξασφαλίζει διαρκή ανάγνωση της συσκευής προκειμένου να εντοπιστούν τυχόν διαταραχές[40]. Σημαντικό, όμως είναι να γνωρίζουμε αυτές οι λειτουργίες σε ποια σημεία του κυκλώματος εφαρμόζονται και αν συνεπάγονται αντίκτυπο στην λειτουργία της συσκευής.

3.5.1.Αρχιτεκτονική της μνήμης διαμόρφωσης

Η μνήμη διαμόρφωσης του FPGA χωρίζεται σε τρία διαφορετικά τμήματα: στα CLB τμήματα και τα πλαίσια BRAM0 και BRAM1.Τα BRAM τμήματα διατηρούν το περιεχόμενο των RAM κελιών και διευθυνσιοδοτούνται διαφορετικά από το CLB απαιτώντας έτσι, ξεχωριστές εντολές εγγραφής και ανάγνωσης. Γενικά, οι λειτουργίες πρόσβασης στα τμήματα BRAM αποφεύγονται μετά την διαμόρφωση της συσκευής μια και μπορεί να οδηγήσουν σε διακοπή της λειτουργίας της. Τα πλαίσια CLB περιέχουν

όλα τα δεδομένα διαμόρφωσης, για τα προγραμματιζόμενα στοιχεία της συσκευής και άρα εξασφαλίζεται η πρόσβαση τους, με την ίδια εντολή ανάγνωσης ή εγγραφής. Οι εντολές εκτελούνται δίχως να διακοπεί η λειτουργία του κυκλώματος, εκτός και αν τα LUTs χρησιμοποιούνται ως μέρος της RAM. Τέλος, οι καταχώρητες flip-flops είναι αποκομμένοι από τους μανταλωτές διαμόρφωσης και άρα δεν προσπελαύνονται κατά την διαμόρφωση, με αποτέλεσμα να μην τους επηρεάζουν οι λειτουργίες σάρωσης και μερικής διαμόρφωσης.

Τα τμήματα της μνήμης διαμόρφωσης διαιρούνται περαιτέρω σε στήλες πλαισίων δεδομένων



Εικόνα 3.13: Τμήματα Πλαισίων σε συσκευή VIRTEX

(data frames). Ένα πλαίσιο δεδομένων είναι το μικρότερο κομμάτι των δεδομένων διαμόρφωσης, που μπορεί να αναγνωστεί ή να εγγραφεί. Η CLB περιοχή περιέχει τέσσερις κατηγορίες στηλών πλαισίων. Ενδεικτικά αναφέρουμε, μία κεντρική με οκτώ πλαίσια, τις στήλες CLB με 48 πλαίσια ανά στήλη, δύο στήλες BRAM διασύνδεσης με 27 πλαίσια ανά στήλη και δύο στήλες IOB με 54 πλαίσια ανά στήλη. Ο αριθμός των στηλών και το μέγεθος των πλαισίων ποικίλλει ανάλογα με την συσκευή, αλλά το τελευταίο παραμένει σταθερό σε όλο τον διαχωρισμό. Κάθε μεμονωμένο πλαίσιο αντιμετωπίζεται ως ένα ξεχωριστό μπλοκ δεδομένων.

3.5.2. Λειτουργίες μερικής ανάγνωσης και εγγραφής

Για να επιτευχθεί εγγραφή, σε μια σειρά από πλαίσια δεδομένων, αρχικά χρησιμοποιείται ένας καταχωρητής διευθυνσιοδότησης των πλαισίων (Frame Address Register - FAR), ο οποίος λαμβάνει την τιμή της διεύθυνσης του πρώτου πλαισίου. Κατόπιν, καθορίζεται ο αριθμός των λέξεων δεδομένων (32-bits) που θα γραφούν και δίνονται τα δεδομένα. Άρα, ο αριθμός των λέξεων που θα γραφούν ισούται με τον αριθμό των πλαισίων που θα γραφούν επί τον αριθμό των λέξεων/πλαίσιο. Αν η εγγραφή καταλάβει πολλαπλά πλαίσια, το πρώτο θα γραφεί στην αρχική διεύθυνση και κατόπιν ο FAR θα αυξάνει κατά μία διεύθυνση πλαισίου έως το τέλος των δεδομένων.

Για κάθε λειτουργία εγγραφής, στον αριθμό των λέξεων που περιέχει ένα πλαίσιο, προστίθεται και μία λέξη που δεν ανήκει στα δεδομένα και η οποία βοηθά στην ολοκλήρωση της διαδικασίας. Τα προς εγγραφή δεδομένα τοποθετούνται στον καταχωρητή εισόδου δεδομένων πλαισίου (Frame Data Register Input – FDRI), διαιρούνται σε ομάδες των 32 ψηφίων και κατόπιν φορτώνονται στον καταχωρητή πλαισίου, ο οποίος έχει χωρητικότητα ίση με το μέγεθος ενός πλαισίου δεδομένων. Όταν ο

καταχωρητής πλαισίου γεμίσει, τα περιεχόμενα του τοποθετούνται παράλληλα στους μανταλωτές της μνήμης διαμόρφωσης. Η τελευταία, όμως λέξη παραμένει εντός του καταχωρητή, μιας και δεν υπάρχουν άλλα ψηφία για να την ολισθήσουν. Γι' αυτό το λόγο φορτώνεται και μία προεπιλεγμένη λέξη (dummy word), ώστε να ολισθήσουν στους μανταλωτές και τα τελευταία δεδομένα. Η λέξη αυτή παραμένει στον καταχωρητή και άρα φορτώνεται ως πρώτη λέξη στο επόμενο πλαίσιο που θα εγγραφεί.

Η διεύθυνση ενός πλαισίου εκφράζεται ως ο συνδυασμός μιας μέγιστης και μιας ελάχιστης τιμής. Η μέγιστη τιμή αντιστοιχεί στον αριθμό της στήλης και η ελάχιστη στον αριθμό του πλαισίου μέσα στην στήλη. Η τιμή που γράφεται στον FAR περιέχει ένα πεδίο τύπου μπλοκ, την μέγιστη και την ελάχιστη διεύθυνση. Το πεδίο μπλοκ διατηρεί πάντα την τιμή '00' ώστε να υποδηλώνει τα τμήματα πλαισίων CLB. Η μέγιστη διεύθυνση τοποθετείται μεταξύ των ψηφίων 17-24 και η ελάχιστη μεταξύ των 9-16 και τα υπόλοιπα ψηφία είναι '0'. Άρα για να τελεστεί εγγραφή ή ανάγνωση στο πρώτο πλαίσιο, της πρώτης στήλης, ο FAR θα πάρει τιμή ίση με '00000000h'.

3.5.3. Μέθοδοι διόρθωσης SEU

Η μια μέθοδος που προτείνεται είναι η χρήση της σάρωσης (ανάγνωση μετά την διαμόρφωση), ώστε να ανιχνευθούν οι πιθανές διαταραχές που έχουν συμβεί στην μνήμη διαμόρφωσης. Όταν μία διαταραχή ανιχνευτεί, μόνο τα δεδομένα του πλαισίου που περιέχουν το εσφαλμένο ψηφίο θα διορθωθούν. Έτσι, η μνήμη διαμόρφωσης θα βρίσκεται σε λειτουργία εγγραφής για πολύ λίγο χρονικό διάστημα, ενώ γενικά θα υπόκειται αναγνώσεις. Με αυτό τον τρόπο μειώνεται σημαντικά η πιθανότητα, ένα SEU να διαδοθεί και στην υπόλοιπη μνήμη διαμόρφωσης και να έχει δυσμενή αποτελέσματα στην λειτουργία της συσκευής. Παρόλα αυτά η μέθοδος απαιτεί την χρήση επιπλέον κυκλώματος, το οποίο θα υλοποιεί πληθώρα αλγορίθμων, που θα εκτελούν τις αναγνώσεις και τις συγκρίσεις, για κάθε πλαίσιο δεδομένων και θα απαιτεί μνήμη, ώστε να αποθηκεύει τις σταθερές και τις μεταβλητές του.

Απλούστερη θεωρείται η μέθοδος αφαίρεσης ρύπων-σφαλμάτων (scrubbing) με την οποία παραλείπονται οι λειτουργίες ανάγνωσης και ανίχνευσης πιθανών SEUs και απλώς επαναδιαμορφώνεται όλη η CLB περιοχή, σε καθορισμένο χρονικό διάστημα. Η μέθοδος scrubbing απαιτεί λιγότερο επιπλέον κύκλωμα αλλά η μνήμη διαμόρφωσης θα βρίσκεται σε λειτουργία εγγραφής για περισσότερο χρονικό διάστημα. Παρόλα αυτά ένας πλήρης κύκλος scrub είναι σχετικά μικρός καθώς η διασύνδεση SelectMAP μπορεί να λειτουργήσει με ρυθμό 400 Mbits/s.

3.5.4. Ανίχνευση SEUs

Η πιο κοινή μέθοδος για την επαλήθευση των αποθηκευμένων δεδομένων διαμόρφωσης, είναι αυτή της σύγκρισης ανά ψηφίο, μεταξύ ενός αρχείου μάσκας (.msk), το οποίο περιέχει τις πληροφορίες των ψηφίων της ακολουθίας διαμόρφωσης και ενός αρχείου σάρωσης (.rbb), που περιέχει τα δεδομένα που αναγνώστηκαν από την συσκευή. Όμως η τεχνική τριπλασιάζει το μέγεθος της μνήμης που απαιτείται και κατά συνέπεια δεν ενδείκνυται για εφαρμογές με περιορισμό όγκου.

Γι' αυτό το λόγο αναπτύχθηκε μία τεχνική που καταγράφει μία τιμή 16 ψηφίων (CRC) για κάθε πλαίσιο δεδομένων. Η τιμή αυτή παράγεται εκ νέου για κάθε πλαίσιο, κάθε φορά που εκτελείται σάρωση της συσκευής και συγκρίνεται με την CRC τιμή που είχε προκύψει κατά την έναρξη της διαμόρφωσης.

Μιας και ένα πλαίσιο δεδομένων είναι το μικρότερο μέρος της μνήμης διαμόρφωσης, που μπορεί να αναγνωστεί ή να εγγραφεί, δεν είναι αναγκαίο να γνωρίζουμε ποιο ακριβώς ψηφίο του πλαισίου έχει διαταραχθεί, αλλά πιο πλαίσιο και εν συνεχεία αν εντοπιστεί εσφαλμένο, να αντικατασταθεί.

3.5.5. Διόρθωση SEU

Για να παραχθούν οι CRC τιμές που προαναφέρθηκαν προτείνονται δύο μέθοδοι από την XILINX. Αν πρόκειται για συσκευή που δεν θα αλλάξει λειτουργία, σε όλο το χρόνο ζωή της, τότε οι σταθερές τιμές CRC παράγονται με την βοήθεια λογισμικού και αποθηκεύονται στην ROM της συσκευής. Στην περίπτωση συσκευών, που θα αναβαθμιστούν κάποια στιγμή, οι τιμές CRC πρέπει να μπορούν να παραχθούν από το σύστημα που φιλοξενεί την συσκευή και να αποθηκευτούν στην RAM.

Στην περίπτωση που ανιχνευτεί ένα σφάλμα στις τιμές CRC, ο αριθμός του πλαισίου που εντοπίστηκε καταγράφεται στην μνήμη, ώστε να χρησιμοποιηθεί αφότου ολοκληρωθεί η διαδικασία σάρωσης της συσκευής. Το σύστημα πρέπει να μπορεί να αποθηκεύει πολλαπλές διευθύνσεις πλαισίων, μιας και διαταραχή μπορεί να εμφανιστεί σε περισσότερα πλαίσια του ενός. Η διαδικασία που ακολουθείται για την διόρθωση του εσφαλμένου πλαισίου είναι :

I. Ματαίωση (abort): εντολή τριών κύκλων ρολογιού, με την οποία επανεκκινείται το SelectMAP και η λογική διαμόρφωσης, ώστε οι διασυνδέσεις να επανασυγρονιστούν. Με τον τρόπο αυτό δεν απαιτείται ο υπολογισμός των κύκλων ρολογιού, που χρειάζονται μεταξύ της ολοκλήρωσης της σάρωσης και της εγγραφής και απαλείφονται οποιαδήποτε SEUs βρίσκονται στην μνήμη διαμόρφωσης αυτόματα.

II. Συγχρονισμός(Synchronize): πριν μια καινούργια διαδικασία χρησιμοποιήσει τις διασυνδέσεις του SelectMAP, το τελευταίο πρέπει να επανασυγχρονιστεί, μέσω φόρτωσης της λέξης συγχρονισμού.

III. Έναρξη αρχείου κυματομορφής διαμόρφωσης (Wave Configuration File -WCFG): επιτρέπει την εγγραφή στην περιοχή της μνήμης διαμόρφωσης, με την φόρτωση του αρχείου μέσω CMD.

IV. Φόρτωση FAR με την διεύθυνση του πλαισίου.

V. Καθορισμός του μήκους του πλαισίου σε λέξεις των 32 ψηφίων στον FDRI.

VI. Φόρτωση των δεδομένων στο FPGA μαζί με την πλεονάζουσα λέξη.

VII. Επανεκκίνηση του CRC, με μηδενισμό του καταχωρητή του.

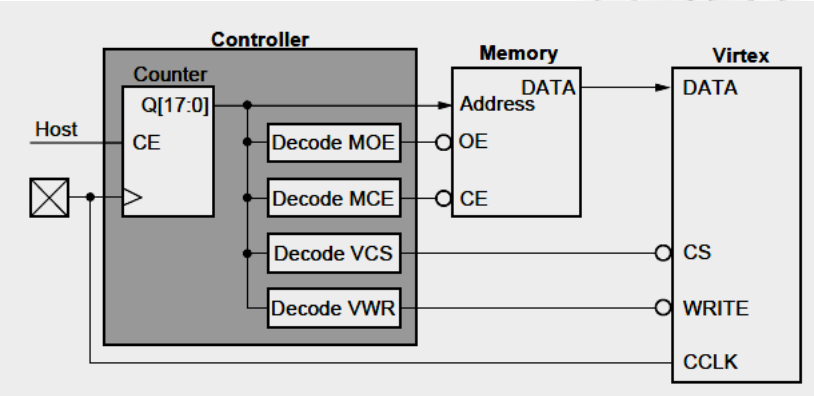
VIII. Ματαίωση.

3.5.6. SEU Scrubbing

Η μέθοδος scrubbing προσεγγίζει την επιδιόρθωση σφαλμάτων με απλούστερο τρόπο και δεν απαιτεί τις λειτουργίες της σάρωσης και της επαλήθευσης των δεδομένων διαμόρφωσης. Η διαδικασία επαναδιαμορφώνει όλο το CLB τμήμα, σε ένα προεπιλεγμένο χρονικό σημείο[5]. Αρχικά φορτώνει την

ακολουθία ψηφίων διαμόρφωσης στην συσκευή, αλλά σταματά μετά το πέρας της πρώτης εγγραφής στον καταχωρητή FDRI. Η πρώτη εγγραφή στον FDRI, περιλαμβάνει όλα τα δεδομένα διαμόρφωσης που αφορούν τα πλαίσια CLB στην περιοχή της μνήμης. Η υπόλοιπη ακολουθία ψηφίων διαμόρφωσης περιέχει τις πληροφορίες για τα τμήματα BRAM και τον έλεγχο CRCκαι την διαδικασία έναρξης της συσκευής, τα οποία και δεν εξετάζονται κατά την μερική επαναδιαμόρφωση.

Πριν και μετά από έναν κύκλο scrub εκτελείται εντολή ματαίωσης, αν και η εντολή που



Εικόνα 3.14: Κύκλωμα ελέγχου Scrubbing

προηγείται μπορεί να παραλειφθεί, αν έχει συμπεριληφθεί στο τέλος της αρχικής διαμόρφωσης. Το μοναδικό επιπλέον κύκλωμα που απαιτεί η μέθοδος είναι ένας μετρητής, ο οποίος θα παράγει τις διευθύνσεις μνήμης και θα συμμετέχει σε μια λογική αποκωδικοποίησης, η οποία θα συνδέει τα σήματα ελέγχου της μνήμης και τα σήματα διασύνδεσης του SelectMAP, με σταθερές τιμές που θα παράγει ο ίδιος μετρητής.

Στο παράδειγμα της εικόνας 3.14 χρησιμοποιείται μια παράλληλη μνήμη των 8 ψηφίων, τα οποία συνδέονται σε ένα VIRTEX FPGA στους ακροδέκτες των δεδομένων και τις διασυνδέσεις του VIRTEX SelectMAP. Αν οι θύρες των δεδομένων της μνήμης ήταν διαφορετικής διαμόρφωσης, τότε οι πληροφορίες θα έπρεπε να χωριστούν σε λέξεις των 8 ψηφίων, μέσα στο τσιπ ελέγχου. Στην προκειμένη περίπτωση ο μετρητής ελέγχει την διαδικασία scrubbing. Η έξοδος του λιγότερου σημαντικού ψηφίου (Least Significant Bit – LSB) χρησιμοποιείται για να διευθυνσιοδοτήσει την μνήμη. Παρόλα αυτά, αν το ρολόι του συστήματος έχει υψηλότερη συχνότητα λειτουργίας από αυτή που απαιτούν οι διασυνδέσεις διαμόρφωσης στο κύκλωμα (μέγιστη 50 MHz), τότε οι γραμμές των διευθύνσεων του μετρητή θα ολισθήσουν σε υψηλότερες θέσεις, αφήνοντας τα τελευταία ψηφία για να παράγει ένα διαιρέτη της συχνότητας του ρολογιού.

Ο μετρητής αποκωδικοποιεί τέσσερα σήματα: το σήμα επίτρεψης της εξόδου της μνήμης (Memory Output Chip Enable – MOE), το σήμα επίτρεψης του τσιπ της μνήμης (Memory Chip Enamble – MCE), το σήμα επίτρεψης του τσιπ VIRTEX (VIRTEX Chip Enable –VCS) και τέλος το σήμα εγγραφής VIRTEX (VIRTEX Write – VWR). Η πολυπλοκότητα της αποκωδικοποίησης εξαρτάται από τον αριθμό των τσιπ μνήμης και FPGAs που έχουν ενσωματωθεί στην τελική εφαρμογή. Αν υπήρχαν πολλαπλά τσιπ μνήμης το κύκλωμα θα απαιτούσε κι ένα αποκωδικοποιητή MCE, Στην περίπτωση όμως

ενός τσιπ, το σήμα MCE μπορεί να συνδεθεί στον αποκωδικοποιητή MOE. Ο τελευταίος πρέπει να απενεργοποιήσει την έξοδο της μνήμης κατά την εντολή ματαίωσης. Όμως τα σήματα VCS και VWRδεν μπορούν να συνδυαστούν, ακόμα και αν υπάρχει μόνο μία συσκευή FPGA, γιατί η εντολή της ματαίωσης απαιτεί ξεχωριστό έλεγχο αυτών των σημάτων. Στον πίνακα 3.4 φαίνονται οι αλλαγές κατάστασης για μια πλήρη λειτουργία scrub, στην οποία συμπεριλαμβάνονται και οι εντολές ματαίωσης και οι κύκλοι ρολογιού που απαιτούνται.

Καταστάσεις Μετάβασης					**Κύκλοι Ρολογιού**		
Τύπος Εντολής	**MOE**	**MCE**	**VCS**	**VWR**	**XQVR300**	**XQVR600**	**XQVR1000**
Φόρτωση (Load)	L	L	L	L	207.972	435.312	745.596
Ματαίωση (Abort)	H	H	L	H	4		
Απενεργοποίηση (Disable)	H	H	H	H	1		

Πίνακας 3.4: Καταστάσεις μετάβασης σε διαδικασία scrubbing

Ο κύκλος Scrubbing εξαρτάται από την συχνότητα του ρολογιού διαμόρφωσης και το μέγεθος του της ακολουθίας ψηφίων διαμόρφωσης. Για την XQVR300 συσκευή απαιτούνται 207,972 κύκλοι ρολογιού, ώστε να ολοκληρωθεί ένα πλήρες scrubbing (Scrub κύκλος = # κύκλοι ρολογιού x περίοδος ρολογιού). Ο ρυθμός scrubbing υποδηλώνει πόσο συχνά συμβαίνει ένας κύκλος scrub και καθορίζεται από το πόσο συχνά αναμένεται η εμφάνιση SEU στην συγκεκριμένη εφαρμογή. Ο ρυθμός scrubbing πρέπει να οριστεί ώστε κάθε SEU στην μνήμη διαμόρφωσης να επιδιορθωθεί, πριν εμφανιστεί το επόμενο. Στην πραγματικότητα ο ρυθμός scrubbing ελαχιστοποιείται, ώστε να είναι ίσος με τον κύκλο Scrub. Με αυτό τον τρόπο η λογική διαμόρφωσης ανανεώνεται συνεχώς. Το ενσωματωμένο κύκλωμα μπορεί επίσης να επηρεάσει τους προαναφερόμενους ρυθμούς, επιβάλλοντας κατά μέσο όρο scrub, τουλάχιστον 10 φορές μεταξύ των διαταραχών (και του ρυθμού τους).

4. Τεχνικές ανίχνευσης και διόρθωσης προσωρινών σφαλμάτων σε Συσκευές XILINX VIRTEX – 5

Ακολούθως, θα αναπτυχθούν οι προτεινόμενες τεχνικές ανίχνευσης και διόρθωσης σφαλμάτων σε συσκευές VIRTEX – 5 της XILINX, που αναφέρονται στην εργασία «SEU Strategies for Virtex-5 Devices» των Ken Chapman και Les Jones [41]. Μιας και στην πραγματικότητα τα SEUs είναι απρόβλεπτα και τυχαία, η XILINX προτείνει την χρήση ενός SEU controller macro με την βοήθεια του οποίου είναι δυνατή η εισαγωγή διαταραχών σε ένα επιλεγμένο κύκλωμα και η διόρθωση τους. Στόχος είναι η αξιολόγηση των νέων εφαρμογών ως προς την ανοσία τους στα προσωρινά σφάλματα, ο εντοπισμός των κρίσιμων μερών τους, που ίσως να απαιτούν χρήση κυκλωμάτων άμβλυνσης που χρησιμοποιούνται και η εκτίμηση της αξιόπιστης λειτουργίας των τελευταίων. Στην παρούσα μεταπτυχιακή διατριβή εξετάζουμε συσκευές XILINX VIRTEX-5 ML505 XC5VLX50T και ακολούθως θα ασχοληθούμε εκτενώς με την αξιολόγησή της.

4.1. Εκτίμηση Αστοχίας Και Προδιαγραφές

Η οποιαδήποτε ανάλυση αξιοπιστίας ενός κυκλώματος, πρέπει να έχει ως σημείο εκκίνησης την παράθεση των κρίσιμων εκείνων στοιχείων, μονοπατιών, περιοχών του σχεδιασμού και της κατασκευής και να υπάρχει μία στάθμιση σε σχέση με την πιθανότητα αστοχίας του συστήματος. Μετρητικά, όλο αυτό εκφράζεται ως ο ρυθμός αστοχιών στο χρόνο (FIT – failures in time) ή ως ο χρόνος μεταξύ αστοχιών (MTBF – mean time between failures).

Βάσει πειραμάτων σε συσκευές VIRTEX-5 (Device Reliability Report), προέκυψε ότι τυπικά ο ρυθμός για τα *κελιά διαμόρφωσης* αγγίζει το 131 FIT/Mb (προσέγγιση κατά 90%), με εύρος ανοχής από -20% έως +26%[41]. Ειδικά για την συσκευή XC5VLX50T σε ML505 αναπτυξιακή πλακέτα, που διαθέτει 11,37 Mb κελιά διαμόρφωσης, παρουσιάζει επιδεκτικότητα σε σφάλματα με ρυθμό 1.489 FIT ή 77 χρόνια MTBF (114.155/1.489 FIT = 77 χρόνια MTBF).

Όταν η σύνθεση ενός κυκλώματος γίνεται περιπλοκότερη, η πιθανότητα να προσβληθεί από ένα SEU αυξάνεται αναλογικά. Για την XC5VLX50T στα 1.000 προϊόντα ο ρυθμός αστοχίας είναι 1.489.000, κάτι που ουσιαστικά υποδηλώνει ένα SEU κάθε 28 χρόνια. Το συμπέρασμα αυτό, δεν πρέπει να παρερμηνευτεί ως την πιθανότητα να εμφανιστεί διαταραχή σε μία μεμονωμένη συσκευή. Επιπρόσθετα, η πιθανότητα εμφάνισης ενός δεύτερου SEU καθορίζεται από τον FIT της συγκεκριμένης συσκευής και όχι από τον συνολική εικόνα γι' αυτά τα συστήματα. Πρόκειται για μια σημαντική παρατήρηση, ειδικά όταν πρέπει να ληφθούν αποφάσεις σε σχέση με την στρατηγική που θα ακολουθηθεί για την πρόληψη της αστοχίας στο κύκλωμα.

Σύμφωνα με πειράματα που έχουν διεξαχθεί, έχει αποδειχτεί ότι ένα SEU διαταράσσει ένα μεμονωμένο ψηφίο, ενώ πολλαπλές διαταραχές συμβαίνουν σπάνια έως καθόλου. Επίσης, υψηλή είναι η πιθανότητα αυτό το ψηφίο να έχει ελάχιστή έως καθόλου σημασία για το τελικό κύκλωμα, μιας και λιγότερο από το 20% (τυπικά 10%) των κελιών διαμόρφωσης έχουν ουσιαστικό ρόλο στην ενσωμάτωση

του σχεδίου. Συνήθως, ένα σημαντικό μέρος των πόρων την συσκευής παραμένει αχρησιμοποίητο και άρα δεν μετέχει στο τελικό ποσοστό αστοχίας.

Στην συσκευή που εξετάζουμε, κάθε πλαίσιο διαμόρφωσης αποτελείται από 1.312 ψηφία, στα οποία εμπεριέχονται 12 ψηφία κώδικα διόρθωσης (error correction code-ECC) [42]. Οποιαδήποτε αλλαγή στα ECC ψηφία, λόγω SEU, δεν έχει κανένα αντίκτυπο στο κύκλωμα. Κάθε πλαίσιο περιέχει επίσης 16 αχρησιμοποίητα ψηφία (ψηφία 656-671) και άρα υπάρχουν συνολικά 28 ψηφία που δεν επηρεάζουν τον τελικό σχεδιασμό. Συμπερασματικά, 2,13% από τα ψηφία διαμόρφωσης μπορούν να αποκλειστούν από τους υπολογισμούς του ρυθμού των διαταραχών, σε οποιαδήποτε περίπτωση.

Επίσης, υπάρχει ένας μεγάλος αριθμός από αχρησιμοποίητα κελιά διαμόρφωσης σε περιοχές της συσκευής, όπου ο πόρος θεωρείται χρηστικός. Για παράδειγμα, η προγραμματιζόμενη διασύνδεση έχει πολλές δυνατότητες, αλλά ελάχιστες από αυτές χρησιμοποιούνται στον εκάστοτε σχεδιασμό. Κατά συνέπεια, ένα SEU που θα προκαλούσε την σύνδεση ενός αχρησιμοποίητου τμήματος με ένα χρηστικό, είναι σχεδόν απίθανο να προκαλέσει κάποια αξιομνημόνευτη αλλαγή.

Άρα, μια πρώτη συντηρητική προσέγγιση σε σχέση με την διαμόρφωση της συσκευής, συνεπάγεται ότι ο FIT μπορεί υποπενταπλασσιαστεί, για να προκύψει ένας εγγύτερος FIT στην πραγματικότητα. Για την XC5VLX50T κάτι τέτοιο σημαίνει ότι ο MTBF πλησιάζει τα 245-520 χρόνια (95% προσέγγιση) και αυτός ο αριθμός βελτιώνεται όσο το κύκλωμα καταλαμβάνει λιγότερο ποσοστό του συνόλου της συσκευής.

4.2. Σύγκριση SEUs Διαμόρφωσης και Δεδομένων

Αν ένα SEU συμβεί σε ένα ψηφίο, η κατάστασή του αντιστρέφεται. Αυτό το ψηφίο μπορεί να σχετίζεται με την διαμόρφωση της συσκευής ή μπορεί να αλλάξει ένα λειτουργικό δεδομένο, όπου δεδομένο σημαίνει οποιαδήποτε μεταβλητή ενός κυκλώματος, συμπεριλαμβανομένου των περιεχομένων των RAMs και flip-flops.Στις περισσότερες περιπτώσεις, μία τέτοια αλλαγή δεδομένου μπορεί να αγνοηθεί. Για παράδειγμα, μπορεί να διαταραχθεί το ψηφίο ενός ASCII χαρακτήρα ή το πίξελ μιας εικόνας αλλά εφόσον το δεδομένο είναι παροδικό, θα αντικατασταθεί σύντομα από νέο. Παρόλα αυτά, όταν τα δεδομένα είναι εντολές ή καταστάσεις που απαιτούνται για την λειτουργία του κυκλώματος, η συνέπεια ενός SEU μπορεί να είναι καταλυτική. Για παράδειγμα, μία μηχανή καταστάσεων, μπορεί να υποπέσει σε άγνωστη κατάσταση ή μια διεύθυνση IP να προκύψει λανθασμένη. Γι' αυτές τις περιπτώσεις απαιτείται η εκτίμηση του κινδύνου για το κύκλωμα και η λήψη μέτρων προστασίας.

Πέραν, του υπολογισμού λοιπόν του FIT ή του MTBF για τα κελιά διαμόρφωσης απαιτείται αντίστοιχος υπολογισμός και για τα περιεχόμενα των *μπλοκ μνήμης* της συσκευής. Για την XC5VLX50T, η οποία περιέχει εξήντα BRAMs των 36 Kb και συνολική χωρητικότητα 2,11Mb, ο τυπικός ρυθμός FIT/Mb είναι 1.339, δηλαδή MTBF 85 χρόνια[43]. Αν και υπάρχουν πέντε φορές περισσότερα κελιά διαμόρφωσης απ' ότι BRAMs ψηφία μνήμης, οι δύο ρυθμοί FIT/Mb είναι κοντά. Και αυτό γιατί ενώ τα κελιά διαμόρφωσης είναι πιο εύρωστα και με μεγαλύτερη ανοσία, τα μπλοκ μνήμης αλλάζουν δυναμικά και γρήγορα λόγω λειτουργικών απαιτήσεων και άρα είναι πιο επιρρεπή στα SEUs.

Επίσης, τα κυκλώματα είθισται να χρησιμοποιούν μεγάλο μέρος των BRAMs αυξάνοντας την πιθανότητα να παρουσιαστεί ένα SEU δεδομένων. Συγκριτικά, ο αριθμός των κελιών διαμόρφωσης που

χρησιμοποιούντα, τυπικά είναι χαμηλός και άρα ο FIT δεδομένων θα κυριαρχήσει στο τελικό κύκλωμα. Άρα, όταν ένα BRAM χρησιμοποιείται εκτενώς σε μια εφαρμογή, η σπουδαιότητα των δεδομένων που περιλαμβάνει πρέπει να εξεταστεί , πριν την διαμόρφωση.

Τα δεδομένα που εμπεριέχονται σε flip-flops είναι το λιγότερο πιθανά να υποστούν SEUs. Εκτενής δοκιμές έχουν δείξει ότι ο FIT για τα flip-flops είναι μεταξύ 1-2 /Mb. Λόγω αυτής της μικρής τιμής ρυθμού και του λίγου αριθμού flip-flops στην συσκευή, τα flip-flops συνήθως παραλείπονται από τους υπολογισμούς αστοχίας. Στην XC5VLX50T υπάρχουν περίπου 0,03 Mb flip-flops και ο μέγιστος FIT είναι 0,06 ή MTBF στα 2 x 10^6 χρόνια, ακόμα και αν χρησιμοποιούνται όλα τα flip-flops της συσκευής.

Παρόλα αυτά, αν στο κύκλωμα υπάρχουν δεδομένα υψηλής αξίας για την ομαλότητα της τελικής λειτουργίας, ακόμα και αν κρατούνται σε flip-flops, ο σχεδιαστής θα πρέπει να εισάγει λογική προστασίας, με την οποία θα μπορεί να ανιχνεύει, διορθώνει, αγνοεί ή ανακτά το δεδομένο του, όταν η εφαρμογή θα βρίσκεται υπό την επήρεια SEU. Μόνο το ίδιο το κύκλωμα έχει την δυνατότητα να αντιληφθεί πότε ένα δεδομένο είναι λανθασμένο και γι' αυτό είναι θεμιτό στα BRAMs να υπάρχει η επιλογή ECC, αν και η απόλυτη προστασία των δεδομένων ίσως να απαιτεί την ενσωμάτωση μεθόδων άμβλυνσης, όπως είναι η TMR.

4.3. Στρατηγικές Χειρισμού των SEUs Διαμόρφωσης

Αν και στις πιο απλές εφαρμογές, γενικά το θέμα των SEUs μπορεί να θεωρηθεί αμελητέο και να παραβλεφθεί, υπάρχουν κυκλώματα που δεν επιτρέπουν την παραμικρή απόκλιση από την ομαλή λειτουργία τους. Γι' αυτό και απαιτούνται τρόποι με τους οποίους θα αποφευχθούν τέτοιου είδους διαταραχές, κυρίως στο κομμάτι της διαμόρφωσης. Οι στρατηγικές που θα αναλυθούν και προτείνονται από την XILINX προς του σχεδιαστές συστημάτων, είναι η προγραμματιζόμενη συντήρηση, η επείγουσα συντήρηση, η επιδιόρθωση και τέλος ο συνδυασμός τους. Επειδή, τα SEUs δεν εμφανίζονται συχνά, οι σχεδιαστές πρέπει να μελετήσουν προσεκτικά το κύκλωμά τους και τις προτεινόμενες στρατηγικές αντιμετώπισης, ιδιαιτέρως ως προς τα μειονεκτήματα και πλεονεκτήματα που εισάγουν στο τελικό κύκλωμα και αν η επιλογή τους είναι αναγκαία.

4.3.1. Στρατηγική Προγραμματιζόμενης Συντήρησης

Ας θωρήσουμε ένα πιθανό αποτέλεσμα ενός SEU, το οποίο αντιστρέφει το ψηφίο σε ένα κελί διαμόρφωσης. Ο αντίκτυπος στο κύκλωμα μπορεί να είναι άμεσος ή να μην παρατηρηθούν ανωμαλίες για συγκεκριμένο χρονικό διάστημα. Για παράδειγμα, μία διαταραχή στην διανομή του κεντρικού ρολογιού, μπορεί να έχει απευθείας ένα παρατηρούμενο αποτέλεσμα, αλλά μία διαταραχή σε ένα κύκλωμα που θα μετρούσε ώρες, μπορεί να μην εντοπιστεί για πολλές ώρες μετά. Αυτό σημαίνει, ότι και αν ο χρήστης δεν είναι σε θέση να παρατηρήσει πιθανές διαταραχές, το σύστημα πρέπει να προστατεύει Χρήσιμη είναι η εκτίμηση του χρόνου που μία διαταραχή θα επηρεάσει τελικά το κύκλωμα και του αποτελέσματος που θα επιφέρει. Τελικά, πρέπει να αποφασιστεί αν μετά την διαταραχή το κύκλωμα θα συνεχίσει την κανονική του λειτουργία ή θα αποτύχει με ασφάλεια, ώστε να καθοριστούν τα απαραίτητα μέτρα προστασίας.

Αν και υπάρχουν εφαρμογές που αναμένεται να λειτουργούν αδιάκοπα για μεγάλα χρονικά διαστήματα, πολύ λίγα είναι αυτά που δεν διακόπτεται η λειτουργίας τους για όλο τον χρόνο ζωής τους. Στην πραγματικότητα, οι περισσότερες εφαρμογές διανύουν ανενεργές περιόδους, κυρίως λόγω εργασιών συντήρησης. Κατά την στρατηγική προγραμματιζόμενης συντήρησης (Scheduled Maintenance), αυτές οι περίοδοι χρησιμοποιούνται στο έπακρο για την επαναδιαμόρφωση της συσκευής. Ο καλύτερος τρόπος εφαρμογής της στρατηγικής είναι η επαναδιαμόρφωση να εκτελείται σε κάθε χρονική στιγμή, που το κύκλωμα δεν διαδραματίζει ενεργό ρόλο στο σύστημα. Η επαναδιαμόρφωση εκτελείται χωρίς να απαιτείται ο εντοπισμός κάποιας διαταραχής.

Τίθεται όμως, το ερώτημα τι συμβαίνει στο χρονικό διάστημα μεταξύ ενός SEU και της επόμενης επαναδιαμόρφωσης, κατά την οποία σίγουρα θα διορθωθεί. Εδώ είναι που τα μέτρα προστασίας πρέπει να αποφασίσουν αν θα συνεχιστεί η λειτουργία ή το σύστημα θα αποτύχει. Κατά συνέπεια, απαιτείται κάποιου είδους πλεονασμός στα κυκλώματα αυτά (TMR τεχνική), ο οποίος θα συγκεντρώνει την αστοχία σε ένα μέρος του κυκλώματος, επιτρέποντας στα άλλα δύο να λειτουργούν ομαλά. Με τον τρόπο αυτό θα αντιμετωπίζονται επιτυχώς τα απλά σφάλματα στο σύστημα. Υπάρχει όμως η πιθανότητα να εμφανιστεί και δεύτερη διαταραχή, πριν την επόμενη επαναδιαμόρφωση. Η συχνότητα ενός τέτοιου φαινομένου υπολογίζεται βάσει του MTBF και ως γενικός κανόνας ισχύει ότι όσο πιο συχνές είναι οι επαναδιαμορφώσεις τόσο μικρότερη είναι η πιθανότητα εμφάνισης διπλών διαταραχών.

Πάντως, ακόμα κι αν συμβεί ένα δεύτερο SEU, θα πρέπει να επηρεάσει ένα ψηφίο, σε συγκεκριμένο μέρος του κυκλώματος ώστε να προξενήσει ένα δυσμενές αποτέλεσμα. Δεδομένου ότι λιγότερο από 10% των ψηφίων διαμόρφωσης χρησιμοποιούνται απευθείας από το κύκλωμα, προκύπτει μια πιθανότητα μικρότερη του 1%, ότι και τα δύο SEUs θα επηρεάσουν κρίσιμα σημεία του κυκλώματος. Επίσης, αν ο σχεδιασμός έχει προβλέψει την ενσωμάτωση τεχνικών άμβλυνσης, όπως η TMR, ένα δεύτερο λάθος σε περιοχή που ήδη έχει διαταραχθεί δεν προξενεί περαιτέρω αναστάτωση. Συμπερασματικά, η προγραμματιζόμενη στρατηγική συντήρησης διορθώνει οποιοδήποτε σφάλμα που έχει συμβεί και αν το κύκλωμα διαθέτει και τεχνικές άμβλυνσης, η μέθοδος θεωρείται ικανοποιητική.

4.3.2. Στρατηγική Επείγουσας Συντήρησης

Ο στόχος της επείγουσας στρατηγικής (Emergency Maintenance) είναι να επαναδιαμορφώσει το FPGA, όταν μια διαταραχή λάβει χώρα και να μην αναμένει για την επόμενη προγραμματιζόμενη συντήρηση. Η στρατηγική επιβάλλει την ύπαρξη ανάλυσης, σε σχέση με το πόσο άμεση πρέπει να είναι η επαναδιαμόρφωση και αν και πόσο μπορεί να καθυστερήσει, με γνώμονα βέβαια την πιο γρήγορη αντίδραση στο φαινόμενο.

Προφανώς, ο πιο σημαντικός παράγοντας για την εφαρμογή της συγκεκριμένης στρατηγικής είναι να διαπιστωθεί αν ένα SEU έχει συμβεί σε κάποιο κελί διαμόρφωσης[42,43]. Οι VIRTEX-5 συσκευές διαθέτουν ένα ενσωματωμένο σύστημα μέσω του μηχανισμού της μερικής επαναδιαμόρφωσης, σαρώνουν διαρκώς τα κελιά διαμόρφωσης της συσκευής και παράγουν μία CRC τιμή των 32 ψηφίων. Με αυτή την τιμή των 32 ψηφίων, το κύκλωμα έχει δυνατότητα να σαρώσει έως 2^{32} κελιά και μπορεί να διαπιστώσει αν κάποιο κελί έχει υποστεί αλλαγή.

Η αντίδραση σε ενδεχόμενη διαταραχή εξαρτάται από την εφαρμογή και πρέπει να συνυπολογιστούν δύο παράγοντες. Κατ' αρχήν πρέπει να υπενθυμίσουμε ότι το σήμα του λάθους υποδηλώνει σφάλμα σε κελί διαμόρφωσης και όχι στα δεδομένα πληροφοριών της εφαρμογής. Αν λοιπόν υπάρχουν κρίσιμα δεδομένα, τότε απαιτούνται επιπλέον μέτρα ασφάλειας για την προστασία τους. Δεύτερον, η ανίχνευση ενός λάθους από το κύκλωμα σάρωσης CRC απαιτεί χρόνο. Για την ML505 XC5VLX50T συσκευή απαιτούνται 355.190 κύκλοι ρολογιού για κάθε πλήρη σάρωση, κάτι που σημαίνει ότι για συχνότητα λειτουργίας του ρολογιού στα 60 MHz χρειάζονται 5,92 ms.

Ο χρόνος που απαιτείται για να αναφερθεί η ύπαρξη διαταραχής εξαρτάται από την σχετική θέση μεταξύ της σάρωσης και του σημείου που εμφανίστηκε το SEU. Αν το SEU εμφανίστηκε μετά την έναρξη της σάρωσης θα αναφερθεί νωρίτερα από ένα SEU, που εμφανίστηκε μετά το πέρας της σάρωσης και θα πρέπει να περιμένει την επόμενη για να εντοπιστεί. Κατά συνέπεια ο χρόνος εντοπισμού μιας διαταραχής μπορεί να είναι μέχρι δύο φορές το χρόνο σάρωσης, με μέσο όρο την διάρκεια μίας σάρωσης. Αν ο μέγιστος χρόνος σάρωσης είναι ανεκτός από την εφαρμογή, τότε η στρατηγική δεν απαιτεί οποιοδήποτε άλλο κύκλωμα και ικανοποιεί τις απαιτήσεις με διαδοχικές σαρώσεις και επαναδιαμόρφωση όταν απαιτείται. Αν όμως ο χρόνος ανίχνευσης θεωρείται σημαντικός (για παράδειγμα 20ms είναι ισοδύναμα με 3.000.000 κύκλους ρολογιού, σε εφαρμογή με συχνότητα λειτουργίας τα 150MHz), τότε πρέπει να συμπεριληφθούν κυκλώματα άμβλυνσης.

Ανάλογα με την προστασία που θα εισαχθεί στην εφαρμογή και τις λειτουργικές της απαιτήσεις, μπορεί να εκτιμηθεί αν η επαναδιαμόρφωση θα καθυστερήσει, έως ότου το κύκλωμα φτάσει σε ένα επιθυμητό σημείο λειτουργίας. Δεδομένου ότι λιγότερο από 20% των κελιών διαμόρφωσης έχουν άμεση σχέση με τις αντιδράσεις του κυκλώματος, ακόμα και αν η σάρωση CRC εντοπίσει σφάλμα, στατιστικά ένα στα πέντε σφάλματα θα προκαλέσει διαταραχή. Η επαναδιαμόρφωση λοιπόν κάθε φορά που θα εντοπιστεί σφάλμα, δεν ενδείκνυται για την σπουδαιότητα των σφαλμάτων και η ενσωμάτωση τεχνικών άμβλυνσης και η προγραμματιζόμενη στρατηγική είναι πιο ικανοποιητικές.

Βασιζόμενοι στις τιμές του XC5VLX50T που υπολογίστηκαν προηγουμένως, ακόμα και η ελάχιστη τιμή του MTBF στα 61 χρόνια, υποδηλώνει ότι η επείγουσα συντήρηση θα είναι ένα πολύ σπάνιο γεγονός και αν ληφθεί υπόψη η μέγιστη τιμή, τότε απλώς μπορεί να παραλειφθεί. Εν κατακλείδι, η εφαρμογή της επείγουσας στρατηγικής έχει ουσία μόνο αν ο χρόνος σάρωσης είναι ανεκτός σε σχέση με τα ενδεχόμενα αποτελέσματα μιας διαταραχής. Τότε, όλα τα επιπλέον κυκλώματα άμβλυνσης θεωρούνται περιττά, απλοποιώντας έτσι τον σχεδιασμό, δημιουργώντας μικρότερες συσκευές με περιορισμένη πυκνότητα, οι οποίες είναι εκ προοιμίου λιγότερο επιρρεπής στις διαταραχές.

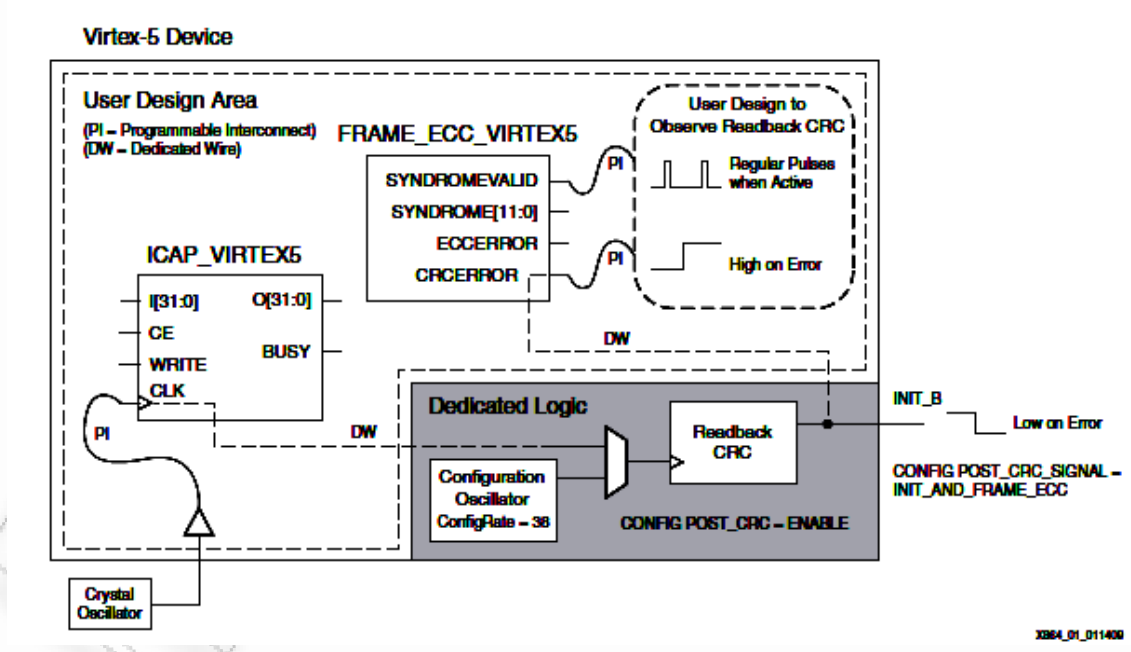
4.3.3. Το Κύκλωμα σάρωσης CRC

Στην εικόνα 4.1 φαίνεται το κύκλωμα που εκτελεί την λειτουργία σάρωσης CRC (Readback CRC), το οποίο ενεργοποιείται από το σήμα POST_CRC. Για υψηλότερο βαθμό αξιοπιστίας της ανίχνευσης, εισάγεται εσωτερικός ταλαντωτής διαμόρφωσης που συνήθως σχετίζεται με το CCLK καθώς και τον ακροδέκτη INIT_B που παρατηρείται από την κατάσταση (status) διαμόρφωσης. Το INIT_B οδηγείται σε λογικό χαμηλό (low) κάθε φορά που επιχειρείται προγραμματισμός με εσφαλμένη εικόνα διαμόρφωσης ή κατά την προετοιμασία για προγραμματισμό, οπότε και είναι σημαντική η παρακολούθηση του.

Συνήθως, χρησιμοποιείται ένας εξωτερικός ελεγκτής με αρκετή λογική, ώστε να ερμηνεύει για πιο γεγονός το INIT_B είναι χαμηλό, βάσει της κατάστασης διαμόρφωσης της συσκευής.

Το προαιρετικό κύκλωμα FRAME_ECC_VIRTEX5 παρέχει εσωτερική πρόσβαση στην ίδια τιμή που υποδηλώνει εσφαλμένη κατάσταση CRC και παρουσιάζεται εξωτερικά μέσω του ακροδέκτη INIT_B και μπορεί να χρησιμοποιηθεί για να παρατηρηθούν τα σφάλματα διαμόρφωσης και να πραγματοποιηθούν οι κατάλληλες ενέργειες. Είναι πιθανό το INIT_B να απενεργοποιηθεί μέσω του περιορισμού POST_CRC_SIGNAL = FRAME_ECC_ONLY, ώστε να χρησιμοποιείται μόνο το εσωτερικό σήμα για την παρατήρηση των σφαλμάτων. Ωστόσο πρέπει να συνυπολογιστεί το γεγονός, ότι η εσωτερική διασύνδεση και η δρομολόγηση του εσωτερικού CRCERROR, είναι ομοίως με το υπόλοιπο κύκλωμα επιρρεπής σε SEUs. Κατά συνέπεια, το INIT_B αποτελεί το σημείο αναφοράς, της υψηλότερης αξιοπιστίας. Για συστήματα με υψηλές απαιτήσεις, μπορούν να χρησιμοποιηθούν και τα δύο σήματα, σε διαφορετικά κυκλώματα, προς ανίχνευση σφαλμάτων.

Αν στο κύκλωμα είναι αναγκαία η ύπαρξη ενός ακόμα ρολογιού (κυρίως για ανιχνεύσεις χρόνου), παρέχεται μέσω σύνδεσης με την είσοδο CLK του προαιρετικού κυκλώματος ICAP_VIRTEX5. Με αυτή την σύνδεση, η αξιοπιστία της λογικής σάρωσης CRC εξαρτάται από την ακεραιότητα του έτερου ρολογιού και την σύνδεσή του. Το ρολόι πρέπει να παρέχεται όσο πιο άμεσα γίνεται (Εικόνα 4.1), μιας και οποιαδήποτε επιπρόσθετη λογική μπορεί να υποστεί SEU. Αν ο εξωτερικό ρολόι διακοπεί ή αποτύχει να φτάσει στο κύκλωμα CRC, η σάρωση της συσκευής τερματίζεται και άρα δεν υπάρχει δυνατότητα αναφοράς σφαλμάτων.



Εικόνα 4.1: Readback CRC σε VIRTEX-5 συσκευή

Επίσης, το FRAME_ECC_VIRTEX5 παρέχει πρόσβαση στο σήμα SYNDROMEVALID, το οποίο σχετίζεται με το σήμα των 12ψηφίων SYNDROME και το σήμα εξόδου ECCERROR. Στην προκειμένη περίπτωση παραμένει αχρησιμοποίητο και εν δυνάμει μπορεί να επιβεβαιώνει αν το κύκλωμα CRC σαρώνει και άρα το ρολόι λειτουργεί.

4.3.4. Χρόνος Σάρωσης CRC

Ο αριθμός των κύκλων ρολογιού που απαιτούνται για να τελεστεί μία σάρωση συγκεκριμένης συσκευής είναι καθορισμένος, αλλά η συχνότητα χρονισμού του κυκλώματος και άρα ο χρόνος ολοκλήρωσης κάθε σάρωσης καθορίζεται από τις ρυθμίσεις του χρήστη και τις δυνατότητες της συσκευής.

Η πιο αξιόπιστη ανίχνευση επιτυγχάνεται όταν το κύκλωμα CRC οδηγείται από τον εσωτερικό ταλαντωτή, ο οποίος τυπικά συσχετίζεται με το κύριο ρολόι της συσκευής (CCLK). Και αυτό γιατί δεν εμπλέκεται κανένας εξωτερικός ταλαντωτής και άρα περιορίζεται ο κίνδυνος που προέρχεται από τα ίδια τα εξαρτήματα και τις διασυνδέσεις τους, καθώς και ο κίνδυνος εμφάνισης SEUs στα σημεία δρομολόγησης του σήματος του εξωτερικού ταλαντωτή στην εφαρμογή. Παρόλα αυτά, ο εσωτερικός ταλαντωτής ποικίλλει κατασκευαστικά, ως προς την τάση τροφοδοσίας και την θερμοκρασιακή ευαισθησία του (PVT - Process-Voltage-Temperature) και μπορεί να παρεκκλίνει έως και ±50% από την ονομαστική τιμή λειτουργίας του, η οποία καθορίζεται από την επιλογή ConfigRate στο εργαλείο BitGen. Οι ονομαστικές τιμές ConfigRate που διατίθενται είναι 2, 6, 9, 13, 17, 20, 24, 27, 31, 35, 38, 42, 46, 49, 53, 56 και 60 (MHz). Η επιλογή των μεγαλύτερων τιμών συνεπάγεται συντομότερη σάρωση και συνήθως είναι προτιμότερο να επιλέγεται άλλη τιμή από την προκαθορισμένη στα 2 MHz. Πρέπει όμως να τεθεί υπό μελέτη η επιλογή συχνότητας, ώστε να εξασφαλιστεί η συμβατότητα με την τρέχουσα μέθοδο διαμόρφωσης και να υπολογιστεί ότι η μέγιστη συχνότητα λειτουργίας του readback CRC, δεν ξεπερνιέται αν ο ταλαντωτής βρεθεί στο +50% της ονομαστικής του τιμής. Γι’ αυτό το λόγο, η μέγιστη τιμή του ConfigRate που επιλέγεται είναι συνήθως το 38, που αντιστοιχεί σε συχνότητα 19-57 MHz.

Εναλλακτικά, μπορεί να χρησιμοποιηθεί ένα ρολόι χρήστη στο κύκλωμα CRC, που θα επιτρέπει στο χρόνο σάρωσης να είναι προβλέψιμος και τυπικά συντομότερος. Παρόλα αυτά, η αξιοπιστία της ανίχνευσhw υποβαθμίζεται ελαφρώς, μιας και υπάρχει εξάρτηση από το εξωτερικό ρολόι, τις συνδέσεις του ρολογιού με το κύκλωμα και τον κίνδυνο των SEUs στην δρομολόγηση του σήματος. Είναι λοιπόν απαραίτητο να εξασφαλιστούν, η αξιοπιστία της πηγής του ρολογιού και ότι όλες οι συνδέσεις διατηρούνται όσο το δυνατόν απλές και σύντομες.

4.3.5. Στρατηγική Επιδιόρθωσης

Η στρατηγική επιδιόρθωσης (Repairs Strategy) ικανοποιεί εφαρμογές που πρέπει να συνεχίσουν την λειτουργίας τους και μετά την ύπαρξη σφάλματος και μπορούν να διορθώσουν γρήγορα και τοπικά τα απλά σφάλματα. Λόγω της υψηλής πιθανότητας (>80%) ένα SEU, να μην έχει κανένα δυσμενές αποτέλεσμα στην εφαρμογή, αυτή η μέθοδος αποφεύγει την διακοπή για συντήρηση της συσκευή, που περιλαμβάνει πλήρη επαναδιαμόρφωσή της. Παρόλα αυτά, όταν ένα SEU προκύψει, η διαδικασία της επισκευής είναι πολύ σημαντική και με περιορισμούς που πρέπει να ληφθούν υπόψη.

Όπως και με την στρατηγική της επείγουσας συντήρησης, η δυνατότητα readback CRC των VIRTEX-5 συσκευών είναι καθοριστική για την ανίχνευση του σφάλματος στα κελιά διαμόρφωσης, γρήγορα και αξιόπιστα. Επίσης, στην περίπτωση των λογικών λαθών μιας και μπορεί να οδηγήσουν σε αλυσιδωτές αντιδράσεις, εισάγεται επιπλέον λογική στο κύκλωμα. Μία λύση που προτείνεται είναι η επανεκκίνηση όλων των κρίσιμων μερών του κυκλώματος, τα οποία σχετίζονται με τον εντοπισμό του σφάλματος. Αν και με αυτό τον τρόπο διακόπτεται η λειτουργία του κυκλώματος, ο χρόνος που απαιτείται είναι σαφώς μικρότερος από το χρόνο που θα διαρκούσε μία πλήρης επαναδιαμόρφωση, ειδικότερα σε εφαρμογές υψηλής πυκνότητας.

Αρχικά, η ακριβής θέση που εμφανίστηκε διαταραχή, καθορίζεται με την χρήση του ECC (Error Correcting Code) και του syndrome κυκλώματος υπολογισμού. Σε κάθε πλαίσιο διαμόρφωσης (1.312 ψηφία = 41 λέξεις των 32 ψηφία) υπάρχουν 12 ψηφία ECC. Καθώς το πλαίσιο διαβάζεται σε 41 κύκλους ρολογιού, η ενσωματωμένη λογική ECC υπολογίζει την τιμή ECC, για το τρέχον περιεχόμενο και την συγκρίνει με την αρχική τιμή κατά την πρώτη διαμόρφωση. Έτσι, μπορεί να εντοπιστεί με την χρήση του FRAME_ECC_VIRTEX5, η ακριβής τοποθεσία του SEU μέσα στο πλαίσιο.

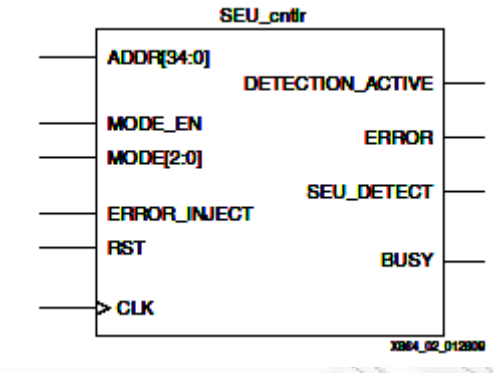
Στην συγκεκριμένη στρατηγική, η διόρθωση είναι άμεση. Το εσφαλμένο πλαίσιο αντιγράφεται σε έναν απομονωτή, το SEU απομονώνεται μέσω ερμηνείας της ECC τιμής και εν συνεχεία διορθώνεται. Η διόρθωση απλώς αντιστρέφει το ψηφίο που είχε αρχικά αντιστρέψει το SEU. Τέλος, το σωστό πλαίσιο αντιγράφεται στην αρχική του θέση. Στην πράξη αυτά τα βήματα είναι αρκετά περίπλοκα, γι' αυτό και εισάγεται η χρήση ενός SEU-Controller macro, που εκτελεί τις αναγκαίες ενέργειες. Η χρήση όμως ενός τέτοιου ελεγκτή δίνει στο εφαρμογή προτερήματα αλλά και περιορισμούς. Αν το κύκλωμα περιέχει επαρκή πλεονασμό, τότε η στρατηγική της προγραμματιζόμενης συντήρησης αρκεί και η πρόσθεση του ελεγκτή ίσως να οδηγήσει σε περιττή πολυπλοκότητα τον τελικό σχεδιασμό.

Μόλις ένα σφάλμα διορθωθεί, το κύκλωμα readback CRC επανεκκινείται και επιβεβαιώνει την ορθότητα της διαμόρφωσης. Παρόλα αυτά, αν και το ECC επιτρέπει την διόρθωση απλών σφαλμάτων, δεν είναι δυνατό να ανιχνεύσει διπλά σφάλματα σε ένα πλαίσιο. Αν κάτι τέτοιο συμβεί, το σφάλμα θεωρείται επίμονο και τότε πρέπει να ληφθούν άλλα μέτρα αντιμετώπισης. Ο μόνος τρόπος για να αντιμετωπιστούν, με την δομή ως έχει, είναι τα σφάλματα να είναι κατανεμημένα σε διαφορετικά πλαίσιο, γεγονός εξαιρετικά απίθανο.

Εν κατακλείδι, τα συστήματα που απαιτούν υψηλό βαθμό αξιοπιστίας, μπορούν να χρησιμοποιήσουν σε συνδυασμό και της τρεις στρατηγικές με τέτοιο τρόπο ώστε η μία να συμπληρώνει αλλά και να καλύπτει την άλλη. Ο πλεονασμός υλικού που θα χρησιμοποιηθεί πρέπει να έχει τέτοιο όγκο ώστε να καλύπτει το κύκλωμα τόσο όσο απαιτείται μέχρι την επισκευή του. Ανεξάρτητα δε από όλα τα μέτρα αντιμετώπισης, η πλήρης επαναδιαμόρφωση του κυκλώματος όταν ο χρόνος το επιτρέπει, είναι αυτή που θα εξασφαλίσει την βεβαιότητα της ορθής λειτουργίας.

4.4. SEU Controller Macro

Ο SEU controller macro είναι ένα εξαιρετικά χρήσιμο κύκλωμα το οποίο εξυπηρετεί δύο σκοπούς: ο πρώτος είναι να επιδιορθώνει σφάλματα διαμόρφωσης, αποτελώντας την ουσία της στρατηγικής επιδιόρθωσης και ο δεύτερος είναι η εισαγωγή ελεγχόμενων σφαλμάτων στα κυκλώματα. Επίσης,



Εικόνα 4.2: SEU Controller Macro

παρέχει τα μέσα για την αξιολόγηση και δοκιμή του κυκλώματος readback CRC και τις ικανότητες διόρθωσης σφαλμάτων, κάτι που είναι αδύνατο να ελεγχθεί με πραγματικά SEUs.

Οι είσοδοι και έξοδοι του ελεγκτή φαίνονται στην εικόνα 4.2 Εντός του ελεγκτή τα κυκλώματα ICAP_VIRTEX5 και FRAME_ECC_VIRTEX χρησιμοποιούνται όπως προαναφέρθηκε για χρονισμό και παρατήρηση στο readback CRC. Επίσης, περιλαμβάνεται και ένας ελεγκτής που συνδέεται στις άλλες θύρες των ανωτέρω κυκλωμάτων, ώστε να εκτελούνται οι αναγκαίες λειτουργίες για τον εντοπισμό και την διόρθωση των SEUs, μέσω της ECC δυνατότητας. Για λόγους δοκιμών, η σύνδεση του ICAP χρησιμοποιείται για την διευκόλυνση της ελεγχόμενης εισαγωγής σφαλμάτων διαμόρφωσης.

Σημαντική παρατήρηση είναι ότι επειδή ο ελεγκτής και το readback CRC, συνδέονται με την διαμόρφωση της συσκευής, η χρήση των εναλλακτικών θυρών διαμόρφωσης, πρέπει να αποφεύγετε. Για παράδειγμα, η χρήση του JTAG για πρόσβαση στην διαμόρφωση θα διέκοπτε την λειτουργία και θα προξενούσε αλλαγές που θα μεταφράζονταν ως σφάλματα. Ομοίως, η επιλογή PERSIST πρέπει να οριστεί NO αλλιώς το ICAP θα απενεργοποιηθεί. Στον πίνακα που ακολουθεί γίνεται περιγραφή των θυρών του ελεγκτή.

Πίνακας 4.1: Σήματα εισόδου και εξόδου του SEU Controller

Σήμα	I/O	Περιγραφή
MODE [2:0]	I	• Τιμή των 3 ψηφίων που καθορίζει τον τρόπο λειτουργίας του ελεγκτή. Η τιμή διαβάζεται από τον ελεγκτή κατά την άνοδο του CLK όταν το MODE_EN = High. • Κατά την εκκίνηση το MODE = 000
MODE_EN	I	• Επιτρέπει την ανάγνωση του MODE[2:0]. Ο ελεγκτής θα υιοθετήσει το νέο MODE μετά το πέρας 40 κύκλων

		ρολογιού, από την στιγμή που θα ενεργοποιηθεί το MODE_EN.Αυτός ο χρόνος πρέπει να τηρηθεί προτού εφαρμοστεί ERROR_INJECT παλμός. • Αν ο ελεγκτής είναι απασχολημένος (BUSY = 1) τότε το MODE θα εφαρμοστεί μετά το πέρας της συγκεκριμένης λειτουργίας.
RST	I	• Όταν RST = High ο ελεγκτής επανεκκινείται κατά την άνοδο του CLK, με μηδενισμό του MODE (000) και του δείκτη εισαγωγής λάθους. • Δεν πραγματοποιείται RST όταν ο ελεγκτής είναι busy. • Λόγω της υψηλότερης προτεραιότητας της θύρας JTAG, αν εφαρμοστεί σήμα ο ελεγκτής επανεκκίνειται και ας μην υπάρχει σήμα RST.
CLK	I	• Ρολόι εσόδου που χρησιμοποιείται από τον ελεγκτή και το readback CRC. Πρόκειται για ένα αξιόπιστο ρολόι που λειτουργεί γύρω στα 60MHz και διανέμεται όσο πιο άμεσα γίνεται (χωρίς επιπρόσθετη λογική). • Η παρακολούθηση του σήματος εξόδου DETECTION_ACTIVE μπορεί να επιβεβαιώσει ότι το ρολόι εργάζεται ομαλά.
SEU_DETECT	O	Όταν το SEU_DETECT = High συμπεραίνεται ότι ο ελεγκτής εντόπισε ένα σφάλμα στην συσκευή. Αυτό το σήμα συνδέεται απευθείας με την έξοδο CRCERROR του FRAME_ECC_VIRTEX5 (εντός του ελεγκτή) και εμφανίζεται και στο pin INIT_B, εκτός και είναι απενεργοποιημένο (δεν ενδείκνυται). Το σήμα θα επανέλθει σε σε λογικό χαμηλό (low) όταν το σφάλμα διορθωθεί.
DETECTION_ACTIVE	O	• Το σήμα συνδέεται άμεσα με την έξοδο SYNDROMEVALID του FRAME_ECC_VIRTEX5 και χρησιμοποιείται για να επιβεβαιώνει ότι η διαδικασία readback CRC είναι ενεργή. Σε κανονική λειτουργία το σήμα θα είναι σε λογικό υψηλό (High) κάθε 41 κύκλους του CLK με εξαίρεση ένα διάστημα των 49 κύκλων, που λαμβάνει χώρα στο τέλος κάθε πλήρους σάρωσης. • Το σήμα θα εμφανίζεται ακανόνιστο όταν ο ελεγκτής θα εισάγει ή θα διορθώνει σφάλματα στην συσκευή.

		• Αν οι παλμοί είναι ανενεργοί, συνεπάγεται κάποια δυσλειτουργία στο CLK. Απαιτείται λοιπόν ένα watchdog σύστημα παρακολούθησης γι' αυτό το σήμα, το οποίο θα λειτουργεί με διαφορετικό ρολόι.
ERROR_INJECT	I	• Το σήμα πρέπει να είναι σε λογικό υψηλό μόνο όταν το mode λειτουργίας είναι το 4,5,6 ή 7 και υποδηλώνει μία ενέργεια σχετιζόμενη με αυτό. Κάθε κύκλος του σήματος απαιτεί χειραψία με την έξοδο BUSY, με την ακόλουθη σειρά: ❖ Επιβεβαίωση ότι το BUSY = Low ή αναμονή μέχρι να γίνει ❖ ERROR_INJECT= High ❖ Αναμονή για BUSY = High ❖ ERROR_INJECT = Low ❖ Πέραν, του απαιτούμενου συγχρονισμού των ανωτέρω βημάτων με το CLK, δεν υπάρχουν άλλοι χρονικοί περιορισμοί. • Η εφαρμογή ERROR_INJECT στα mode λειτουργίας 0,1,2 ή 3 οδηγεί σε μη αναμενόμενη συμπεριφορά. • Πρέπει να επιτραπεί το πέρας 40 κύκλων CLK, μετά την επιλογή του mode λειτουργίας, ώστε να εφαρμοστεί το ο παλμός ERROR_INJECT.
BUSY	O	• Τα σήμα είναι σε λογικό υψηλό όταν ο ελεγκτής εκτελεί μια εργασία, συνήθως την εισαγωγή ή διόρθωση σφαλμάτων. Ο κύριος λόγος ύπαρξής του είναι η συνεργασία με το προαναφερόμενο σήμα, ώστε να ορίζει πότε ολοκληρώθηκε η εργασία και να μπορεί το κύκλωμα να προχωρήσει σε επόμενη εντολή. • Υπό κανονικές συνθήκες το σήμα είναι σε λογικό χαμηλό (low).
ADDR[34:0]	I	• Πρόκειται για είσοδο των 35 ψηφία, η οποία χρησιμοποιείται μόνο από το mode λειτουργίας 4 και καθορίζει την διεύθυνση στην οποία θα εισαχθεί σφάλμα. Η τιμή της διεύθυνσης πρέπει να παραμένει σταθερή στην είσοδο πριν αλλά και κατά την εφαρμογή του ERROR_INJECT. Ο χρόνος που απαιτείται εξαρτάται από την ίδια την τιμή της διεύθυνσης (πόσο μεγάλη

		είναι). • Η μεγαλύτερη τιμή διεύθυνσης είναι η 0x7FFFFFFF hex και χρειάζεται 4 λεπτά για να ολοκληρωθεί με CLK = 50 MHz. • Η μεγαλύτερη τιμή που θα χρησιμοποιηθεί είναι η 0x00433C500 hex, που αντιστοιχεί στην μεγαλύτερη συσκευή XC5VLX330T και χρειάζεται περί τα 0.5 sec για CLK = 50 MHz. • Ο πίνακας 9.4 δείχνει τις μέγιστες διευθύνσεις για κάθε συσκευή. • Αν η ADDR υπερβεί το μέγεθος της συσκευής, οι περιττές τιμές θα αναδιπλωθούν και θα ξεκινήσει η καταμέτρηση από την αρχή.
ERROR	O	• Ο ελεγκτής θα θέσει σε λογικό υψηλό το συγκεκριμένο σήμα, μόνο όταν μία ενέργεια του θεωρηθεί ανεπιτυχής. Η σημασία του σήματος εξαρτάται από το mode λειτουργίας του ελεγκτή, την τρέχουσα στιγμή. • Στο mode = 1, όπου ο ελεγκτής περιμένει να ανιχνευτεί ένα λάθος και κατόπιν να το διορθώσει, αν το ERROR = High συνεπάγεται ότι το σφάλμα δεν διορθώθηκε. Το γεγονός θεωρείται απίθανο και η μόνη περίπτωση που μπορεί να συμβεί είναι κατά την στοχευόμενη εισαγωγή πολλαπλών λαθών στην συσκευή. • Στα modes 4 και 7, όπου γίνεται εισαγωγή σφαλμάτων σε καθορισμένες θέσεις, σημαίνει ότι το σφάλμα δεν εισήχθη. Το ERROR θα επιστρέψει σε τιμή σε λογικό χαμηλό (low), μόνο όταν το mode λειτουργίας αλλάξει σε 0 ή 1. Υπάρχουν δύο πιθανοί λόγοι, για να αποτύχει η εισαγωγή ενός σφάλματος: i. δεν είναι δυνατή η αντιστροφή όλων των ψηφίων στην μνήμη διαμόρφωσης. Υπάρχουν 16 αχρησιμοποίητα ψηφία σε κάθε πλαισίου. ii. υπάρχει περίπτωση να εισαχθεί λάθος στο ίδιο σημείο δύο φορές. Η αντιστροφή του ήδη αντιστρεφόμενου ψηφίο συνεπάγεται τελικώς την ορθότητά του.

4.4.1. Macro Modes

Πίνακας 4.2: SEU Controller Modes Λειτουργίας

Mode	MODE[2:0]	Περιγραφή
0	000	Ανίχνευση
1	001	Ανίχνευση και διόρθωση
2	010	Ανενεργό
3	011	Ανενεργό
4	100	Εισαγωγή λάθους στην ADDR
5	101	Αύξηση του index κατά 1
6	110	Μηδενισμός του index
7	111	Εισαγωγή λάθους στο index

* **Mode 0 – Ανίχνευση**

Το mode 0 είναι το προεπιλεγμένο mode λειτουργίας που έχει η συσκευή μετά από τη διαμόρφωση της ή μετά από σήμα RST. Κατά την συγκεκριμένη λειτουργία, το readback CRC σαρώνει διαρκώς την συσκευή, για τον εντοπισμό σφαλμάτων. Ο ελεγκτής έχει διασφαλίσει εξ' αρχής ότι το ψηφίο RBCRC_EN, εντός του καταχωρητή επιλογής διαμόρφωσης (Configuration Options - COR1) έχει τεθεί '1' και άρα η σάρωση έχει ενεργοποιηθεί, ακόμα και αν ο περιορισμός POST_CRC = ENABLE έχει παραληφθεί ή απενεργοποιηθεί κατά την παραγωγή της ακολουθίας των ψηφίων διαμόρφωσης. Αν ανιχνευτεί ένα SEU, το σήμα SEU_DETECT οδηγείται σε λογικό υψηλό και το INIT_B σε λογικό χαμηλό. Καμία άλλαηενέργεια δεν θα επιτελεστεί.

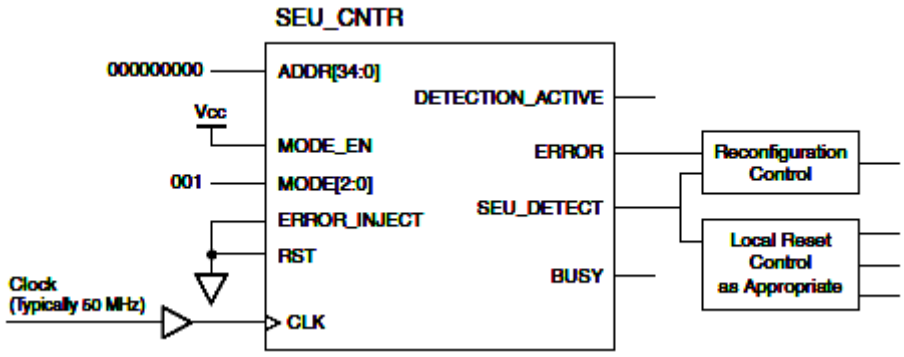
Αν στην εκάστοτε εφαρμογή είναι αναγκαία μόνο η ανίχνευση σφαλμάτων, τότε η χρήση του SEU Controller θεωρείται περιττή. Ο περιορισμός POST_CRC = ENABLE είναι το μόνο που απαιτείται να οριστεί στο readback CRC κύκλωμα, ενώ τα προαιρετικά κυκλώματα ICAP_VIRTEX5 και FRAME_ECC_VIRTEX5, μπορούν να εισαχθούν αν θεωρηθεί αναγκαίο. Παρόλα αυτά, ο ελεγκτής είναι εξαιρετικά χρήσιμο εργαλείο για την δοκιμή των εφαρμογών και την διερεύνηση της αντίδρασής τους στα SEU και άρα μπορεί να εισαχθεί σε πρώτο στάδιο και να απομακρυνθεί μετά την ολοκλήρωση των δοκιμών.

* **Mode 1 – Ανίχνευση και Αυτόματη Διόρθωση**

Αυτή η λειτουργία είναι η κυριότερη για τον SEU Controller και αποτελεί τον λόγο ύπαρξης του. Ισχύουν οι προηγούμενες ρυθμίσεις και ο τρόπος λειτουργίας του mode 0. Κατά την εύρεση ενός σφάλματος, ο ελεγκτής διακόπτει την λειτουργία readback CRC και αναγιγνώσκει την μνήμη διαμόρφωσης, ώστε να εντοπίσει την περιοχή του σφάλματος μέσω του ECC. Το εσφαλμένο ψηφίο αναστρέφεται και αντιγράφεται πίσω στην θέση του. Κατόπιν, το readback CRC σαρώνει την συσκευή και επιβεβαιώνει την διόρθωση του σφάλματος.

Είναι προτεινόμενη η χρήση του σήματος SEU_DETECT για να εισαχθούν τα κατάλληλα σημεία επανεκκίνησης σε μέρη του κυκλώματος που θεωρούνται κρίσιμα. Με αυτό τον τρόπο θα εμποδιστεί η διάδοση των συμπτωμάτων του σφάλματος, στο χρονικό διάστημα μεταξύ του εντοπισμού του και της διόρθωσής του. Επίσης, προτείνεται η καταγραφή των SEUs, ώστε να συνεπάγονται την επαναδιαμόρφωση της συσκευής το συντομότερο δυνατό.

Στην εικόνα 4.3. φαίνονται οι ελάχιστες συνδέσεις που απαιτούνται, ώστε να χρησιμοποιηθεί ο ελεγκτής για ανίχνευση και αυτόματη διόρθωση, μέσα στο κύκλωμα. Το σήμα MODE_EN είναι μόνιμα σε λογικό υψηλό για να εξασφαλιστεί ότι μετά την διαμόρφωση το mode λειτουργίας θα είναι το 1. Επίσης, διασφαλίζει την επιστροφή σε αυτό το mode, αν η λειτουργία του ελεγκτή διακοπεί απροσδόκητα (π.χ. JTAG διακοπή) και υπάρξει υποχρεωτική επανεκκίνηση του. Οι είσοδοι ADDR και ERROR_INJECT, βρίσκονται μόνιμα σε κατάσταση λογικού χαμηλού μιας και δεν χρησιμοποιούνται σε αυτή την λειτουργία. Το SEU_DETECT μπορεί να χρησιμοποιηθεί για τον έλεγχο κυκλωμάτων, τα οποία ίσως επηρεάστηκαν από την παρουσία ενός SEU. Το σήμα ERROR είναι κυρίως υπεύθυνο για την επαναδιαμόρφωση της συσκευής.



Εικόνα 4.3: Ελάχιστες προτεινόμενες συνδέσεις του SEU-Controller σε mode λειτουργίας 1

* **Mode 1 και Mode 2 – Reserved**

Τα συγκεκριμένα mode λειτουργίας είναι κρατημένα, για μελλοντική χρήση και δεν πρέπει να χρησιμοποιούνται. Αν παρόλα αυτά επιλεχθούν, η SEU ανίχνευση θα εξακολουθήσει όπως στο mode 0, αλλά οποιαδήποτε αλλαγή στις εισόδους ADDR ή ERROR_INJECT θα συνεπαγόταν απρόβλεπτη συμπεριφορά του ελεγκτή.

* **Modes Εισαγωγής Σφαλμάτων (4,5,6 και 7)**

Λόγω της σπανιότητας των SEUs, η πλήρης αξιολόγηση και εκτίμηση της συμπεριφοράς μιας συσκευής, υπό την επήρεια τους, είναι πρακτικώς αδύνατη, αν αναμένεται η φυσική αντίδρασή της [41]. Με την χρήση των συγκεκριμένων modes επιτυγχάνεται η προσομοίωση τέτοιων σφαλμάτων, με την βοήθεια του ICAP_VIRTEX5. Όταν είναι επιθυμητό, η μνήμη διαμόρφωσης διαβάζεται, ένα ψηφίο του επιλεγμένου πλαισίου αντιστρέφεται και κατόπιν το πλαίσιο αντιγράφεται στην αρχική του θέση. Έτσι δίνεται η δυνατότητα το SEU να προσομοιωθεί κατά απαίτηση, με προβλεψιμότητα και επαναληπτικότητα.

Χάριν ευκολίας και προς εξυπηρέτηση διαφορετικών πειραματικών διαδικασιών, ο ελεγκτής διαθέτει δύο διαφορετικούς τρόπους εισαγωγής λαθών σε συγκεκριμένες διευθύνσεις, αν και επιτυγχάνουν το ίδιο αποτέλεσμα. Η χρήση και των δύο μεθόδων είναι εφικτή για το ίδιο πείραμα, μιας και οι παράμετροι που χρησιμοποιούν είναι ανεξάρτητες. Κατά τις δοκιμές τα εισαγόμενα σφάλματα μπορούν να διορθωθούν, αν ο ελεγκτής τεθεί σε mode 1 και τα πειράματα θα συνεχιστούν, ενώ μετά την εισαγωγή παραπάνω του ενός σφάλματα επιβάλλεται η επαναδιαμόρφωση της συσκευής. Γενικότερα, προτείνεται η συχνή επαναδιαμόρφωση κατά την διάρκεια των δοκιμών. Συγκεκριμένα για κάθε mode ισχύουν τα εξής:

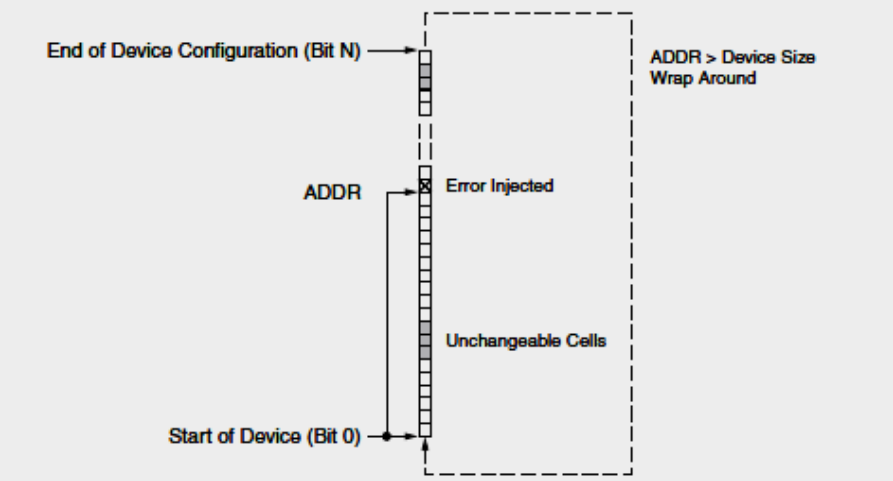
* **Mode 4 – Εισαγωγή σφάλματος στην διεύθυνση ADDR**

Η λειτουργία επιτρέπει την εισαγωγή σφάλματος σε οποιαδήποτε θέση της μνήμης διαμόρφωσης, η οποία ορίζεται από την διεύθυνση των 35 ψηφίων ADDR. Η διαδικασία μπορεί να διαρκέσει έως και 0,5 sec για την μεγαλύτερη XC5VLX330T συσκευή και ο χρόνος πρέπει να ληφθεί υπόψη για την παρακολούθηση των σφαλμάτων, ώστε να διεξάγονται έγκυρα συμπεράσματα.

Για να καθοριστεί η σωστή διεύθυνση, που θα εισαχθεί το σφάλμα, θεωρείται ότι η μνήμη είναι οργανωμένη σε N περιοχές του ενός ψηφίο, όπου N είναι ο συνολικός αριθμός των κελιών διαμόρφωσης της συσκευής. Αυτά ουσιαστικά είναι τα ίδια ψηφία που σαρώνει το κύκλωμα readback CRC και παρόμοια εξαιρούνται τα μπλοκ της μνήμης (μπλοκ RAM). Το εύρος της ADDR ξεκινά από το μηδέν έως το συνολικό αριθμό των ψηφίων διαμόρφωσης της συσκευής. Η χρήση μεγαλύτερων διευθύνσεων από τις αναμενόμενες είναι επιτρεπτή, μιας και όταν ο δείκτης φτάσει στην μεγαλύτερη διεύθυνση, μηδενίζεται και ξεκινά το μέτρημα από το μηδέν, με τελική διεύθυνση την [ADDR (τελική) = ADDR (εισαγωγής) – μέγεθος συσκευής]. Δεν είναι λάθος λοιπόν να εισαχθούν μεγαλύτερες διευθύνσεις αλλά το αποτέλεσμα είναι χρονοβόρο και περιττό.

Για την εισαγωγή του λάθους επιλέγεται το mode= 100 και κατόπιν εισάγεται παλμός στο MODE_EN, με το readback CRC να συνεχίζει την σάρωση. Ακολούθως, εισάγεται η επιθυμητή

διεύθυνση στην ADDR και διασφαλίζεται η παραμονή της εκεί για όσο χρόνο απαιτείται από τα υπο-κυκλώματα. Για να ξεκινήσει η εισαγωγή του σφάλματος, το σήμα ERROR_INJECT οδηγείται σε λογικό



Εικόνα 4.4: Mode 4

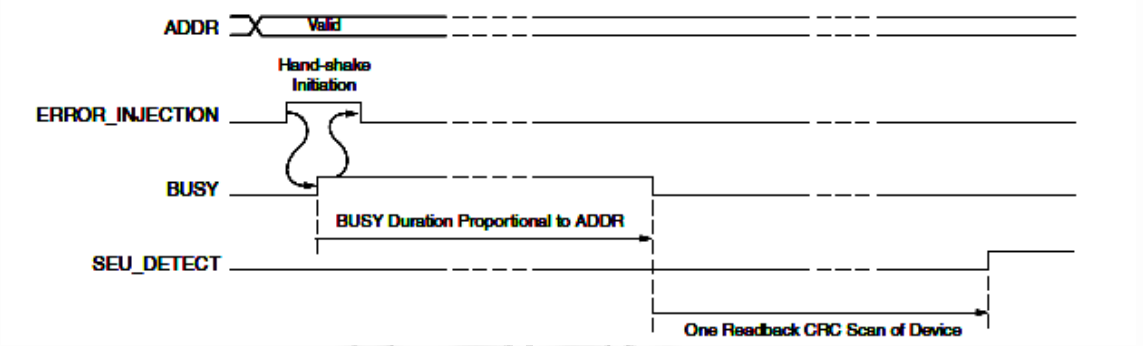
υψηλό και παραμένει σε αυτή την κατάσταση έως ότου το σήμα BUSY οδηγηθεί σε λογικό υψηλό, κάτι που συνεπάγεται την έναρξη της διαδικασίας. Κατόπιν το ERROR_INJECT μπορεί να επανέλθει σε λογικό χαμηλό και το BUSY θα παραμείνει σε λογικό υψηλό μέχρι το σφάλμα να εισαχθεί. Το readback CRC επίσης απενεργοποιείται. Τέλος, κάποιοι παλμοί θα είναι παρόντες στην έξοδο DETECTION_ACTIVE, καθώς ο ελεγκτής θα προσπελαύνει την μνήμη.

Ο χρόνος για την ολοκλήρωση της διαδικασίας εξαρτάται από το μέγεθος της ADDR. Παρόλα αυτά υπάρχει περίπτωση κάποιες από τις ADDR να μην επιτρέπεται να προσπελαστούν και να διαφοροποιηθούν. Σ' αυτές τις περιπτώσεις το σήμα ERROR θα οδηγηθεί σε λογικό υψηλό, όταν το BUSY γίνει Low. Η αποτυχία εισαγωγής σφάλματος αποτελεί επίσης ένδειξη ότι όλα τα SEUs δεν επηρεάζουν την λογική διαμόρφωσης της συσκευής.

Δύο είναι οι βασικοί λόγοι που κάποια ψηφία, σε συγκεκριμένες διευθύνσεις, παραμένουν αμετάβλητα[42]. Πρωταρχικά κάθε πλαίσιο περιέχει 16 αχρησιμοποίητα ψηφία (από το 656 έως το 671), τα οποία δεν μπορούν να γραφούν. Επίσης, υπάρχουν κάποια πλαίσια, κυρίως κοντά στα όρια της συσκευής, που έχουν περισσότερα αχρησιμοποίητα ψηφία από τα προαναφερόμενα. Τρίτον, κατά την χρήση LUT6 από την σχεδίαση, για την διανομή της μνήμης (RAM ή SLR συναρτήσεις) και ο έλεγχος γραφής που συνδέεται με τα LUT6, περνά από την διασύνδεση της διαμόρφωσης στο κύκλωμα. Έτσι παρόλο που τα ψηφία των LUT6 έχουν μία θέση στη μνήμη διαμόρφωσης, είναι μόνο για ανάγνωση από την διασύνδεση. Ο αριθμός και η θέση των συγκεκριμένων ψηφίων εξαρτάται από το κύκλωμα και δεν είναι ορισμένος από την αρχή. Ενώ ένα SEU θα επηρέαζε αυτά τα ψηφία, το αποτέλεσμα θα γινόταν εμφανές στις μεταβλητές δεδομένων του κυκλώματος και όχι στην λογική διαμόρφωσης. Το readback

CRC επίσης αγνοεί τα ψηφία που συνδέονται με αυτού του είδους την μνήμη, οπότε και αν το λάθος εισαγόταν ηθελημένα στο κύκλωμα, δεν θα μπορούσε να ανιχνευτεί.

Ακολουθεί πιο αναλυτική προσέγγιση του τρόπου λειτουργίας του ελεγκτή όταν βρίσκεται σε mode 4,ώστε να εξηγηθεί η σχετικά μεγάλη διάρκεια της διαδικασίας όταν η ADDR είναι εκτενής. Ο ελεγκτής ξεκινά διαβάζοντας το πρώτο πλαίσιο της μνήμης διαμόρφωσης και ορίζει έναν προσωρινό δείκτη στο σημείο μηδέν, το οποίο αντιστοιχεί στο πρώτο ψηφίο του πρώτου πλαισίου. Εν συνεχεία αυξάνει επαναλαμβανόμενα την τιμή του δείκτη, έως ότου εξισωθεί με την τιμή της ADDR, στην οποία θα αντιστραφεί το ψηφίο και ακολούθως ολόκληρο το πλαίσιο που ανήκει, θα γραφτεί πίσω στην αρχική του θέση. Κάθε φορά που η τιμή του δείκτη φτάνει στο τέλος ενός πλαισίου, ανακτάται το επόμενο από την μνήμη, επιτρέποντας έτσι την συνεχόμενη ανάγνωση όλης της συσκευής. Παρόλα αυτά, αυτή η συνεχής ανάγνωση συχνά συναντά πλαίσια που αντιστοιχούν σε περιεχόμενα μνήμης (μπλοκ RAM) ή σε άλλα μη διαμορφούμενα πλαίσια. Όταν τέτοιου είδους πλαίσια αναγνωριστούν, αγνοούνται από τον ελεγκτή με τον ίδιο τρόπο που τα προσπερνά και το κύκλωμα readback CRC. Ο προσωρινός δείκτης



Εικόνα 4.5: Κυματομορφή επιτυχούς εισαγωγής λάθους με mode 4

λοιπόν αυξάνει μόνο όταν συναντά πλαίσια διαμόρφωσης, ώστε τα λάθη να εισάγονται σε περιοχές που καθορίζουν την λογική της συσκευής.

* **Mode 5 – Αύξηση του Index κατά 1**

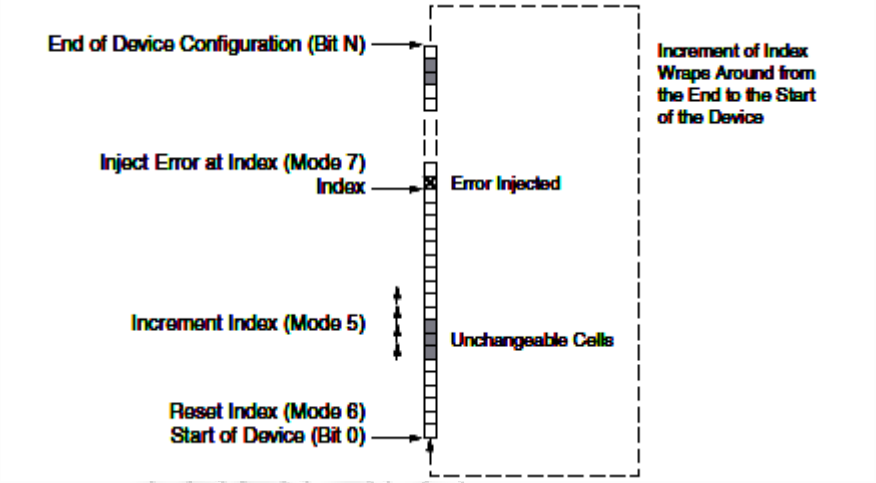
* **Mode 6 – Μηδενισμός του Index**

* **Mode 7 – Εισαγωγή σφάλματος στην διεύθυνση Index**

Τα τρία αυτά mode λειτουργούν συνδυαστικά για να παρέχουν έναν δεύτερο τρόπο εισαγωγής λάθους στη συσκευή. Η προετοιμασία για μια τέτοια δοκιμή είναι χρονοβόρα και ίσως να απαιτεί και προσεκτικό εντοπισμό των περιοχών εισαγωγής σφάλματος, αλλά έχει το πλεονέκτημα ότι η διαδικασία είναι πλήρως προβλεπόμενη σε σχέση με το mode 4 που εξαρτάται από την τιμή της ADDR.

Όπως φαίνεται στην εικόνα 4.6 στο mode 7 εισάγεται σφάλμα στην μνήμη διαμόρφωσης, σε θέση που ορίζει η τιμή ενός εσωτερικού δείκτη (index). Θεωρήστε ότι η μνήμη χωρίζεται σε N περιοχές του ενός ψηφίο, όπου N είναι ο συνολικός αριθμός των κελιών διαμόρφωσης της συσκευής. Αυτός ουσιαστικά είναι ο ίδιος αριθμός ψηφία που παρατηρούνται από το readback CRC, με εξαίρεση πάντα των περιεχομένων των μπλοκ RAM. Ο index αρχικά δείχνει στο πρώτο ψηφίο του πρώτου πλαισίου της μνήμης διαμόρφωσης και επανέρχεται σε αυτή την θέση μέσω του mode 6. Ο δείκτης μπορεί να δείξει σε οποιαδήποτε θέση της μνήμης μέσω του mode 5, το οποίο αυξάνει την τιμή του κατά ένα, όσες φορές είναι επιθυμητό. Αν αυξάνοντας την τιμή του, φτάσουμε στον αριθμό N, η επόμενη αύξηση αυτόματα θα επιστρέψει το δείκτη στο αρχικό σημείο (μηδέν). Κατά συνέπεια, αυτή η διαδικασία απαιτεί τον συνδυασμό των modes 5 και 6 για να καθοριστεί η τιμή του index και το mode 7 για την εισαγωγή του σφάλματος.

Σημαντική παρατήρηση είναι ότι η αύξηση του index κατά n φορές, μετά το τελευταίο reset του και η

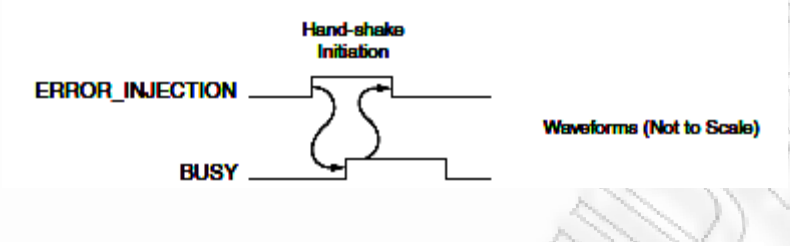


Εικόνα 4.6: Χρήση των modes 5,6 και 7 για την εισαγωγή σφάλματος

εισαγωγή σφάλματος με το mode 7, θα είναι στην ίδια διεύθυνση με αυτή, αν το σφάλμα εισαγόταν με mode 4 σε ADDR = n.

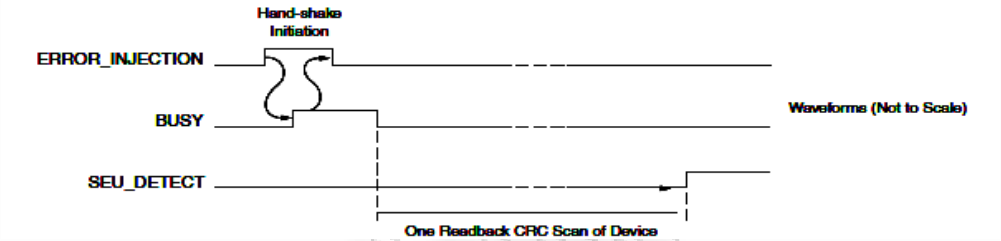
Πιο αναλυτικά, για να επανέλθει η τιμή του index στο μηδέν, ορίζεται το MODE = 110 και εφαρμόζεται παλμός στο σήμα MODE_EN. Μόλις το mode εφαρμοστεί το σήμα ERROR_INJECT οδηγείται σε λογικό υψηλό για να πραγματοποιηθεί επανεκκίνηση. Για την αύξηση της τιμής του index το MODE = 101 και τα ανωτέρω σήματα λαμβάνουν τις ίδιε τιμές. Σε κάθε αύξηση του index, ο ελεγκτής εξασφαλίζει ότι επιτρέπεται η πρόσβαση μόνο σε πλαίσια διαμόρφωσης διαφορετικά αγνοούνται.

Στα modes 5 και 6 το σήμα ERROR_INJECT χρησιμοποιείται για τον έλεγχο της διαδικασίας και όχι για να εισάγει σφάλμα στην συσκευή. Η έξοδος παραμένει σε λογικό υψηλό έως ότου το BUSY οδηγηθεί ομοίως σε λογικό υψηλό, επιβεβαιώνοντας έτσι ότι η διαδικασία ξεκίνησε και τότε το ERROR_INJECT μπορεί να επιστρέψει σε λογικό χαμηλό (Εικόνα 4.7). Το BUSY θα παραμείνει σε



Εικόνα 4.7: Κυματομορφές στα modes 5 και 6

λογικό υψηλό μέχρι να επιτευχθεί ο στόχος του mode, το readback CRC θα είναι ανενεργό μόνο για κείνη την περίοδο και κάποιο παλμοί θα εμφανιστούν στην έξοδο DETECTION_ACTIVE που υποδηλώνουν ότι ο ελεγκτής προσπελαύνει την μνήμη διαμόρφωσης.



Εικόνα 4.8: Εισαγωγή σφάλματος σε index διεύθυνση

Για να εισαχθεί ένα σφάλμα στην διεύθυνση index επιλέγεται MODE= 111 και τα σήματα MODE_EN και ERROR_INJECT οδηγούνται όπως και στα προηγούμενα modes. Στην εικόνα 4.8 φαίνεται η αλληλουχία των σημάτων του ελεγκτή. Ομοίως και με το mode 4, δεν επιτρέπεται η εισαγωγή σφαλμάτων σε όλες τις πιθανές τιμές που μπορεί να λάβει ο index.

4.5 Μέγεθος Ελεγκτή και Ανάλυση Αξιοπιστίας

Ο SEU controller macro απασχολεί περίπου 95 λογικά τμήματα (slices) και 2 μπλοκ μνήμης των 18 Kb. Ο ελεγκτής περιλαμβάνει επίσης τα κυκλώματα ICAP_VIRTEX5 και FRAME_ECC_VIRTEX5 που είναι απαραίτητα για την ανίχνευση, διόρθωση και εισαγωγή σφαλμάτων στην συσκευή. Μιας και ο ελεγκτής αποτελεί μέρος του FPGA, υπόκεινται σε σφάλματα. Γι' αυτό το λόγο πρέπει να αξιολογηθεί, σαν ένα οποιοδήποτε μέρος της τελικής εφαρμογής.

Ξεκινώντας με τα κελιά διαμόρφωσης, γνωρίζουμε ότι έχουν ρυθμό αστοχίας 131FIT/Mb[41]. Χρειαζόμαστε όμως έναν τρόπο ώστε να μετατραπούν τα 95 λογικά τμήματα, οι συνδέσεις τους και οι 2 BRAMs, σε έναν αριθμό ψηφίων διαμόρφωσης. Για να επιτευχθεί αυτή η διαδικασία, χρησιμοποιείται ο πίνακας 4.3 ώστε να υπολογιστούν κατά προσέγγιση. Έτσι, ο αριθμός των ψηφία διαμόρφωσης είναι (95 × 1,181) + (2 × 585) = 113,365 ψηφία ή 0.108 Mb. Άρα, ο FIT του ελεγκτή είναι 16.33 ή 6.992 χρόνια σε MTBF.

Χαρακτηριστικά συσκευής	Αριθμός ψηφίων διαμόρφωσης
1 logic slice	1.181
1 μπλοκ RAM (36 Kb)	1.170
1 μπλοκ RAM (18 Kb)	585
1 I/O μπλοκ	2.657
1 DSP48E slice	4.592

Πίνακας 4.3: Ψηφία διαμόρφωσης του SEU Controller

Ένας εναλλακτικός τρόπος υπολογισμού του FIT του ελεγκτή είναι να συγκριθούν τα ποσοστά της μνήμης που καταλαμβάνει, σε σχέση με το συνολικό μέγεθος της συσκευής. Ο SEU controller macro που είναι τοποθετημένος σε μία XC5VLX50T συσκευή, καταλαμβάνει 1,32% των τμημάτων και 1,67% των BRAMs. Η XC5VLX50T διαθέτει 8.663 πλαίσια των 1.312 το καθένα, δηλαδή 11.365.856 ψηφία διαμόρφωσης, από τα οποία 170.488 απασχολούνται από τον ελεγκτή και άρα συνεπάγονται FIT στα 25,55 ή MTBF στα 4.650 χρόνια. Αυτός ο υπολογισμός είναι 50% πιο απαισιόδοξος από τον προηγούμενο αλλά πιο ακριβής.

Ανεξαρτήτως του τρόπου υπολογισμού του FIT, ισχύει ότι λιγότερο από το 20% των κελιών διαμόρφωσης απασχολούνται από μία εφαρμογή και άρα ο FIT του ελεγκτή μπορεί κι αυτός να υποπενταπλασιαστεί, οδηγώντας σε μία τιμή που αγγίζει τα 3,27 FIT ή MTBF στα 34.960 χρόνια.

Ακολούθως, πρέπει να συνυπολογιστεί και η δεκτικότητα των BRAMs στα SEUs. Ο υπολογισμός είναι άμεσος μιας και ο ελεγκτής διαθέτει δύο BRAMs των 18 Kb, δηλαδή των 36.864 ψηφίων. Η ονομαστική τιμή FIT για τις BRAMs είναι 635, κατά συνέπεια για τον ελεγκτή η τιμή FIT είναι 11 ή σε MTBF στα 10.377 χρόνια. Ο αριθμός που σχετίζεται με τα flip-flops αγνοείται για τους λόγους που αναφέρθηκαν σε προηγούμενο κεφάλαιο.

Τελικά, συνδυάζοντας τους ρυθμούς FIT της διαμόρφωσης (3,27) και των δεδομένων (11) προκύπτει ότι η συνολική πιθανότητα του SEU controller macro να υποπέσει σε σφάλμα είναι περίπου 14.3 ή σε MTBF στα 7.983 χρόνια. Αυτό που απασχολεί βεβαίως είναι τι θα συμβεί αν τελικά ένα τέτοιο σφάλμα συμβεί. Κατ' αρχήν εφόσον η ανίχνευση εξαρτάται αποκλειστικά από το readback CRC, σίγουρα το σφάλμα θα εντοπιστεί αν πρόκειται για σφάλμα διαμόρφωσης. Αν όμως επηρεάσει το ρολόι που τροφοδοτεί το ICAP_VIRTEX5, τότε το readback CRC θα απενεργοποιηθεί. Για να γνωρίζει ο χρήστης πότε ένα τέτοιο γεγονός συνέβη, απαιτείται η παρακολούθηση της εξόδου DETECTION_ACTIVE του ελεγκτή, η οποία συνδέεται απευθείας με την έξοδο SYNDROMEVALID του FRAME_ECC_VIRTEX5 και επιβεβαιώνει την λειτουργία του readback CRC. Αν το readback CRC δεν επηρεαστεί και ανιχνεύσει ένα σφάλμα, τότε η έξοδος CRCERROR του FRAME_ECC_VIRTEX5 οδηγείται σε λογικό υψηλό. Αν το SEU για κάποιο λόγο αποτρέψει τον ελεγκτή να διαβάσει το σφάλμα, τότε δεν θα αντιδράσει καθόλου. Παρόλα αυτά το INIT_B της συσκευής θα οδηγηθεί σε λογικό χαμηλό (low), παρέχοντας έτσι μια εξωγενή παράμετρο παρακολούθησης σφαλμάτων ανεξάρτητη του ελεγκτή. Η εισαγωγή περαιτέρω

κυκλωμάτων παρακολούθησης για τον ίδιο τον ελεγκτή, θεωρείται υπερβολική κι αυτό γιατί ο SEU Controller βρίσκεται εκεί για παρακολούθηση και ένα σφάλμα στην λογική του, δεν επηρεάζει την λειτουργία της εφαρμογής καθαυτής.

5. Πειραματικό Μέρος

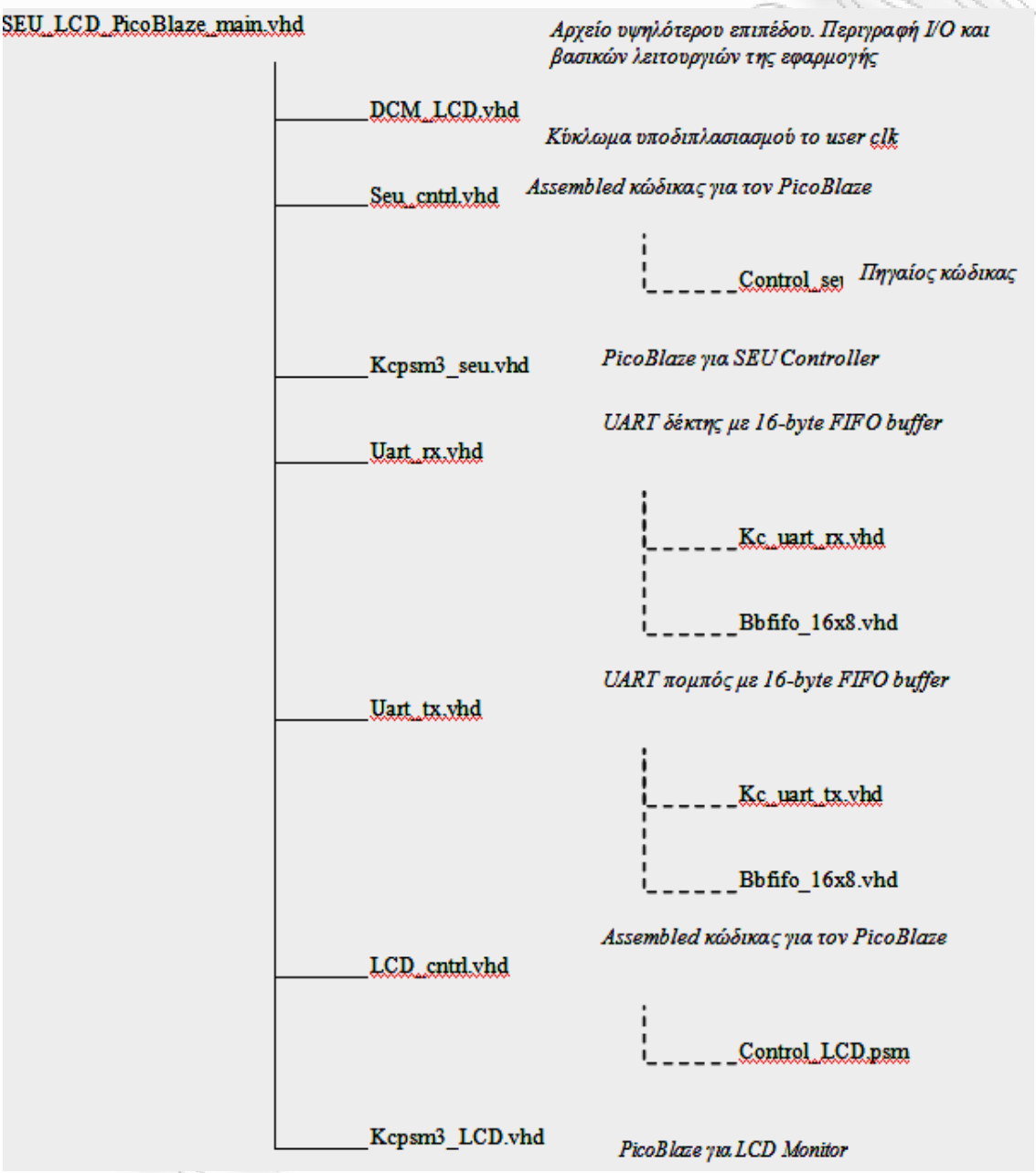
Στόχος της εργασίας είναι η χρήση του SEU Controller macro σε ένα κύκλωμα, ώστε να παρατηρηθεί ο τρόπος λειτουργίας του και η ικανότητα εισαγωγής και διόρθωσης σφαλμάτων. Θα αποδειχθεί ότι η ανεύρεση των κρίσιμων σημείων σε ένα κύκλωμα δεν είναι απλή υπόθεση και ότι όντως τα SEUs είθισται να έχουν μικρό αντίκτυπο στην τελική λειτουργία της εφαρμογής.

5.1 Η εφαρμογή

Η εφαρμογή που δημιουργήθηκε εκμεταλλεύτηκε τις δυνατότητες οπτικής απεικόνισης της συσκευής, ώστε να είναι ευκολότερο από τον παρατηρητή να εντοπίζει τυχόν ανωμαλίες στην λειτουργία κατά την εισαγωγή λαθών. Το project χωρίζεται σε δύο μέρη:

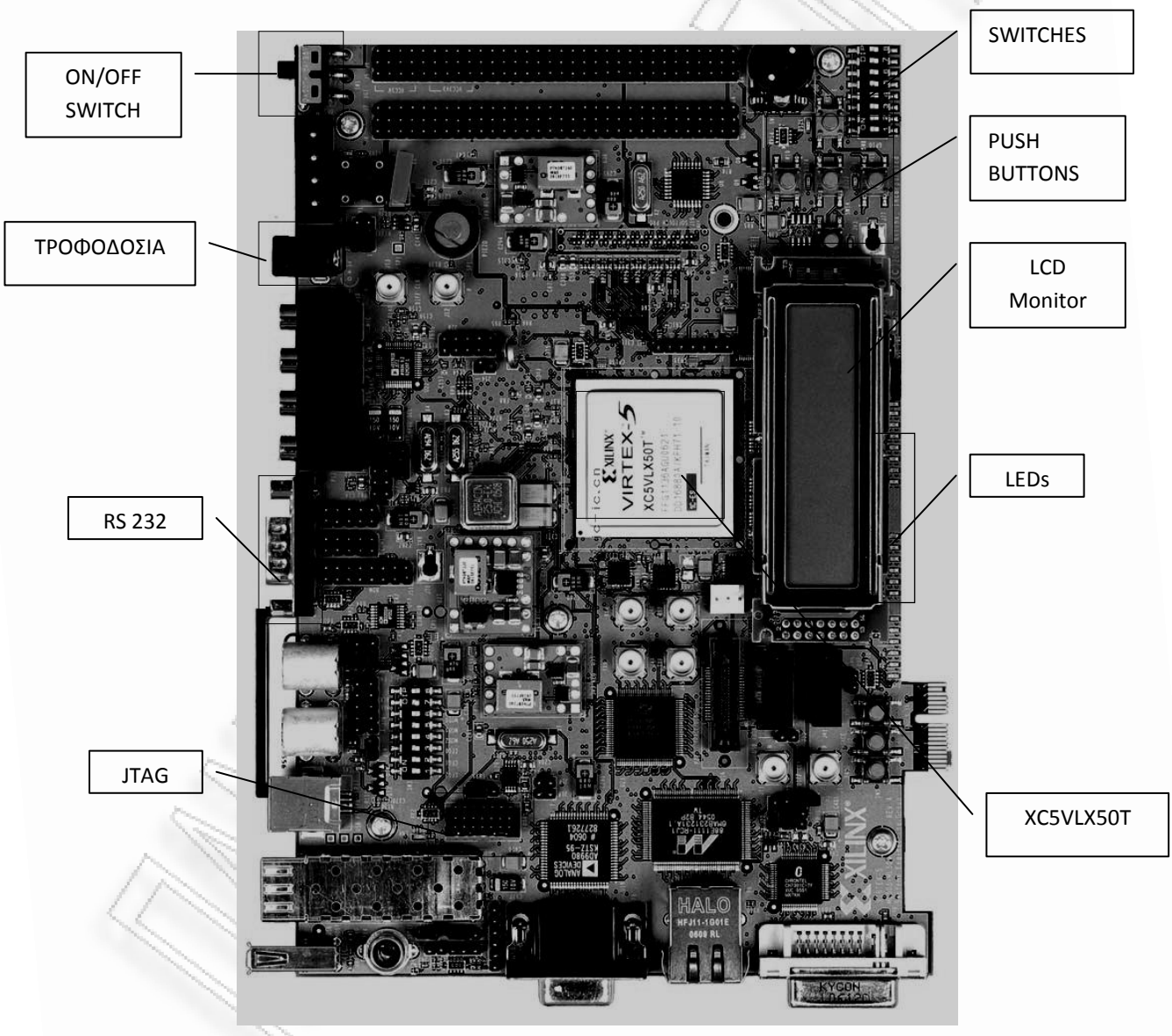
1. Στο SEU Controller Monitor το οποίο ελέγχεται από τον soft μικροεπεξεργαστή Picoblaze και αναλαμβάνει την ανίχνευση, εισαγωγή και διόρθωση σφαλμάτων στο κύκλωμα.
2. Στο LCD το οποίο είναι υπεύθυνο για την κύλιση ενός μηνύματος 2 γραμμών στην Lcd οθόνη της πλακέτας και ελέγχεται λειτουργικά από έναν έτερο soft μικροεπεξεργαστή Picoblaze. Επίσης, ευθύνεται για την διασύνδεση τεσσάρων βασικών διακοπτών (switch) και τεσσάρων κομβίων (push button) της πλακέτας με αντίστοιχο αριθμό LED εξόδου, ώστε όταν τα προαναφερόμενα βρίσκονται σε κατάσταση θέσης να ενεργοποιούνται τα LEDs.

5.1.1. Δομή και Αρχεία του Προγράμματος

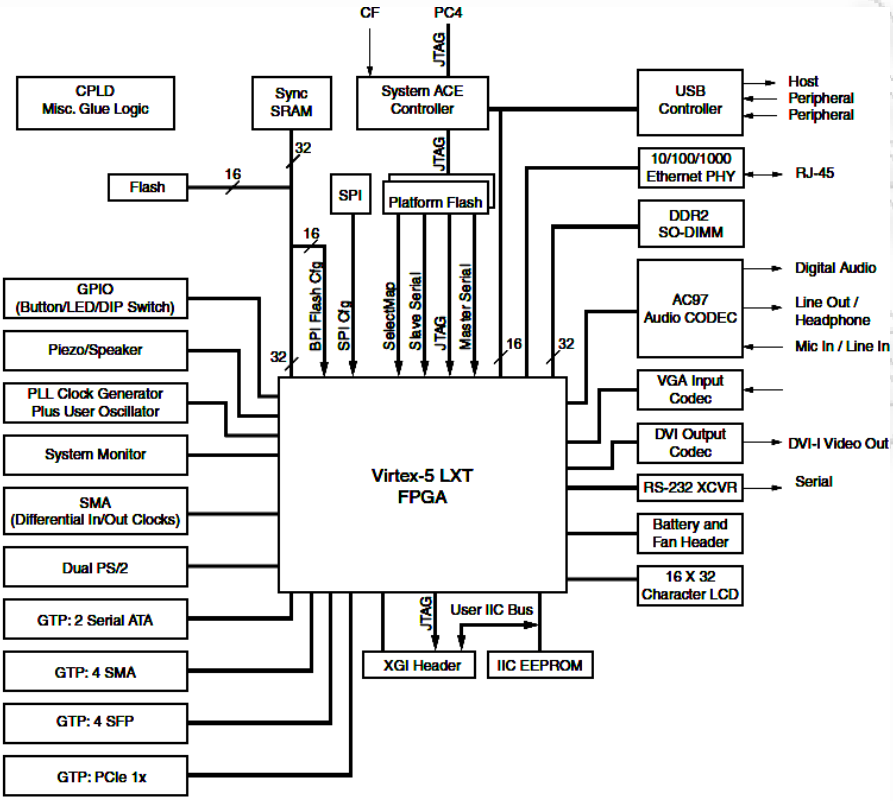


5.1.2. Η συσκευή

Η εφαρμογή υλοποιήθηκε σε FPGA συσκευή από την XILINX. Ανήκει στην οικογένεια VIRTEX-5 και πρόκειται για την ML505 με κύριο εξάρτημα το XC5VLX50T. Η ανάπτυξη του προγράμματος έγινε σε VHDL με την βοήθεια του XILINX ISE Project Navigator 10.1, στον οποίο δημιουργήθηκε το project και οι διασυνδέσεις με τα περιφερειακά της πλακέτας και παρήχθει η ακολουθία ψηφίων διαμόρφωσης με το οποίο προγραμματίστηκε η συσκευή, μέσω JTAG.



Εικόνα 5.1: Συσκευή VIRTEX-5 ML505



Εικόνα 5.2: Δομικό διάγραμμα ML505

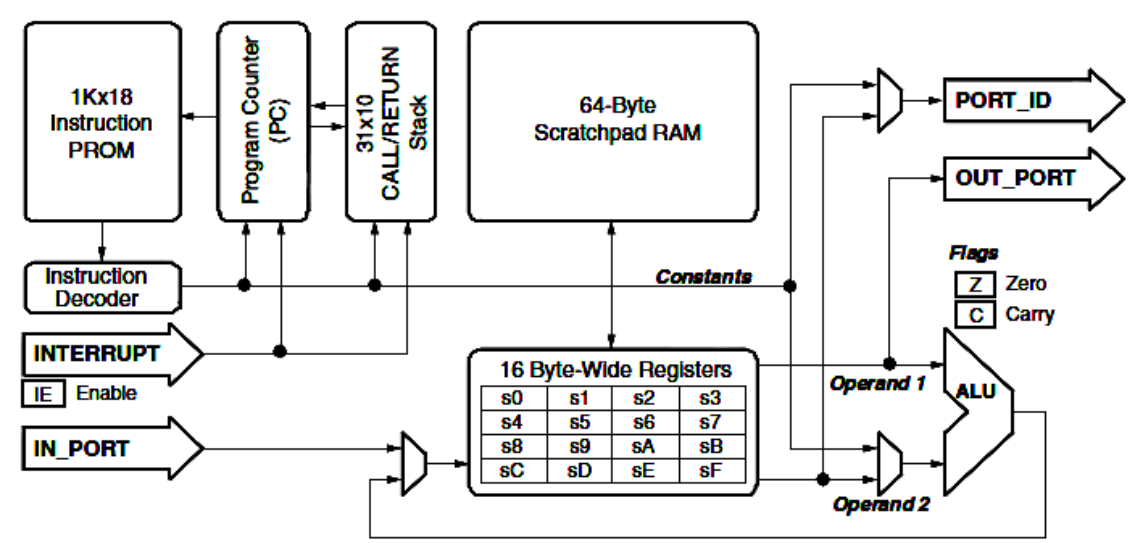
5.2. Τα Βασικά Μέρη του Προγράμματος

Το πρόγραμμα ποικίλλει σε σχεδιαστικά αρχεία καθένα από τα οποία είναι υπεύθυνο για μία ξεχωριστή λειτουργία. Ακολούθως θα αναλυθούν τα βασικότερα σημεία της σχεδίασης, ενώ λεπτομερής αναφορά παρατίθεται στα Παραρτήματα Α και Β.

5.2.1 PICOBLAZE

Ο Picoblaze αποτελεί έναν 8 ψηφίο ενσωματωμένο RISC μικροεπεξεργαστή που σχεδιάστηκε από τον Ken Chapman και βοηθά στον έλεγχο και την επεξεργασία δεδομένων, με απλό και οικονομικό τρόπο [44]. Απασχολεί περί τα 96 FPGA slices και έχει δυνατότητα εκτέλεσης από 44 έως 10^8 εντολών το δευτερόλεπτο, ανάλογα με την συσκευή που υλοποιείται και την ταχύτητα που υιοθετεί.

Τα βασικά δομικά στοιχεία του είναι (Εικόνα 5.3):



Εικόνα 5.3: Δομικό διάγραμμα Picoblaze

1. 16 καταχωρητές γενικού σκοπού, με εύρος 16 byte (S_0 – S_F). Στον Picoblaze δεν υπάρχει συσσωρευτής και δεν τηρείται καμία προτεραιότητα.
2. Η μνήμη προγράμματος έχει μέγιστη χωρητικότητα τις 1.024 εντολές (μέσω ενός BRAM), των 18 ψηφίων.
3. Η ALU με δυνατότητα εκτέλεσης προσθαφαιρέσεων, AND, OR, XOR, αριθμητικών συγκρίσεων, ολίσθησης και περιστροφής. Υποστηρίζονται οι σημαίες μηδενισμού (zero), κρατουμένου (carry) και ενεργοποίησης διακοπής (interrupt enable). Οι πράξεις εκτελούνται μεταξύ οποιουδήποτε καταχωρητή Sx και αποθηκεύονται ομοίως, ανάλογα με τον ορισμό της εντολής. Επίσης, υπάρχει δυνατότητα εκτέλεσης πράξεων με απευθείας ορίσματα των 8 ψηφίων.
4. Η scratchpad RAM των 64-bytes γενικού σκοπού, με προσπέλαση μέσω των εντολών store (εγγραφή Sx στην μνήμη) και fetch (ανάγνωση θέσης της μνήμης και αποθήκευση του περιεχομένου σε ένα Sx) . Η διευθυνσιοδότηση είναι άμεση ή έμμεση.
5. 256 θύρες εισόδου και εξόδου με διευθυνσιοδότηση μέσω του port_id.
6. Ο μετρητής προγράμματος (PC) ο οποίος δείχνει στην επόμενη προς εκτέλεση εντολή και έχει εύρος 10 ψηφία (000-3FF).
7. Οι 3 εντολές αλλαγής της ροής του προγράμματος. Πρόκειται για τις JUMP, η οποία κατευθύνει τον PC σε συγκεκριμένη απόλυτη διεύθυνση εντολής του προγράμματος, τις CALL και RETURN που υποστηρίζουν υπορουτίνες και την RETURNI η οποία εξασφαλίζει επιστροφή στο σημείο που διακόπηκε ένα πρόγραμμα (interrupt service routine).

8. Η δυνατότητα διακοπής του προγράμματος μέσω μίας προαιρετικής εισόδου διακοπών (INTERRUPT). Η αντίδραση στο ασύγχρονο αυτό γεγονός επιτυγχάνεται σε 5 κύκλους.
9. Η είσοδος RESET ώστε να επαναφέρεται ο μικροεπεξεργαστής στην αρχική του κατάσταση, στην οποία τυπικά ισχύει ότι ο PC = 0, οι σημαίες μηδενίζονται, οι διακοπές απενεργοποιούνται ενώ οι καταχωρητές και η RAM παραμένουν αναλλοίωτοι

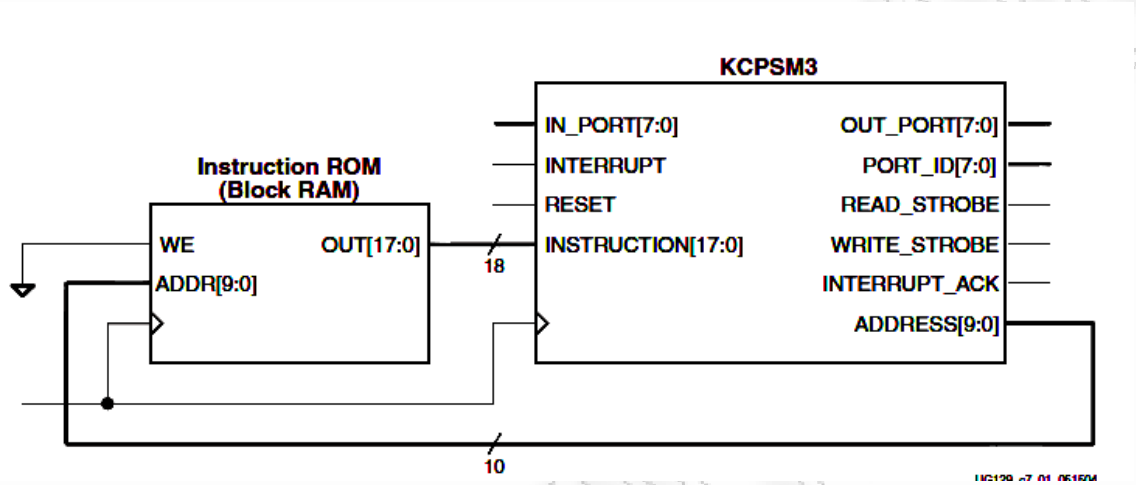
Τα σήματα διασύνδεσης με τον επεξεργαστή αποτελούνται από 4 εισόδους και 5 εξόδους (Εικόνα 5.4). Οι είσοδοι είναι:

1. IN_PORT [7:0]: θύρα θετικής ακμής στην οποία εισάγονται δεδομένα υπό την οδηγία εντολής εισόδου.
2. INTERRUPT: αν τεθεί η σημαία interrupt_enable τότε εισάγεται παλμός σε λογικό υψηλό για 2 κύκλους, διαφορετικά αγνοείται ως είσοδος.
3. RESET: Επαναφέρει τον μικροεπεξεργαστή στην αρχική του κατάσταση αν είναι σε λογικό υψηλό για τουλάχιστον ένα παλμό ρολογιού.
4. CLK: ο μικροεπεξεργαστής συγχρονίζεται στο θετικό μέτωπο του ρολογιού, ενώ το εύρος του παλμού και η μέγιστη συχνότητά του καθορίζονται από το FPGA.

Οι έξοδοι είναι:

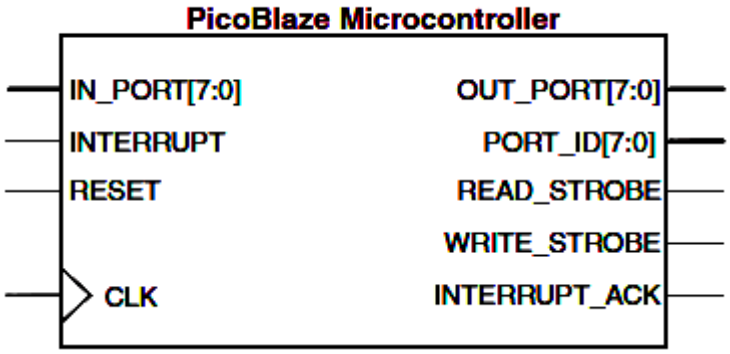
1. OUT_PORT [7:0]: στην θύρα τοποθετούνται τα δεδομένα που εξάγονται από τον μικροεπεξεργαστή υπό την οδηγία εντολής εξόδου, για τουλάχιστον δύο κύκλους ρολογιού.
2. PORT_ID [7:0]: η θύρα παρέχει την διεύθυνση που απαιτείται για να πραγματοποιηθούν οι εντολές εισόδου-εξόδου.
3. READ_STROBE: αν η θύρα οδηγηθεί σε λογικό υψηλό στον δεύτερο κύκλο ρολογιού μιας εντολής ανάγνωσης (εισόδου), επιβεβαιώνεται η επιτυχία της διαδικασίας.
4. WRITE_STROBE: : αν η θύρα οδηγηθεί σε λογικό υψηλό στον δεύτερο κύκλο ρολογιού μιας εντολής εγγραφής (εξόδου), επιβεβαιώνεται η επιτυχία της διαδικασίας.
5. INTERRUPT_ACK: αν η θύρα οδηγηθεί σε σε λογικό υψηλό συνεπάγεται ότι επετεύχθη διακοπή.

Ο Picoblaze εκτελεί κώδικα από τους πόρους της μνήμης που βρίσκονται ενσωματωμένοι στο FPGA και ουσιαστικά η σύστασή του, στην αρχιτεκτονική μια εφαρμογής περιέχει δύο βασικά μπλοκ



Εικόνα 5.4: Ενσωμάτωση 1K x 18 μπλοκ RAM ως μνήμη εντολών

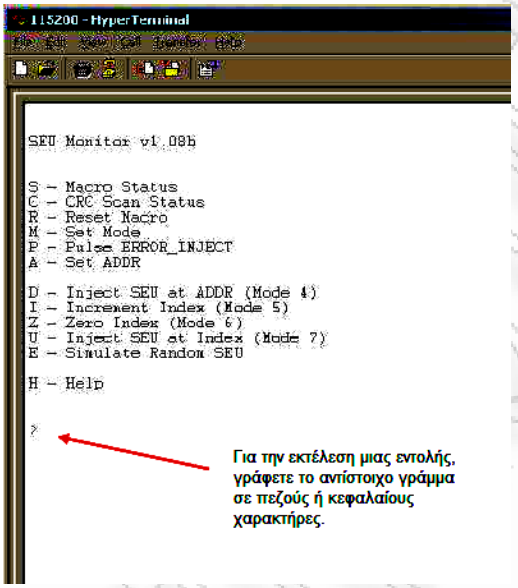
(Εικόνα 5.4). Το ένα είναι το κύκλωμα του μικροεπεξεργαστή (KCPSM3) που περιέχει όλα τα προαναφερόμενα και το άλλο είναι ένα μπλοκ RAM που παρέχεται στον μικροεπεξεργαστή ως αποθήκη εντολών (κώδικα). Το μπλοκ RAM χρησιμοποιείται ως 1Kx18 ROM και για να αποτελέσει ουσιαστικά μία ROM-σε-τσιπ, ο ακροδέκτης επίτρεψης εγγραφής (write enable) γειώνεται, ώστε αποφευχθούν οποιεσδήποτε εντολές εγγραφής. Έτσι ο κώδικας της εφαρμογής συντάσσεται (assemble και compile) ως μέρος του FPGA σχεδίου και οι εντολές τοποθετούνται στο μπλοκ RAM κατά την διαμόρφωση.



Εικόνα 5.5: Σήματα διασύνδεσης Picoblaze

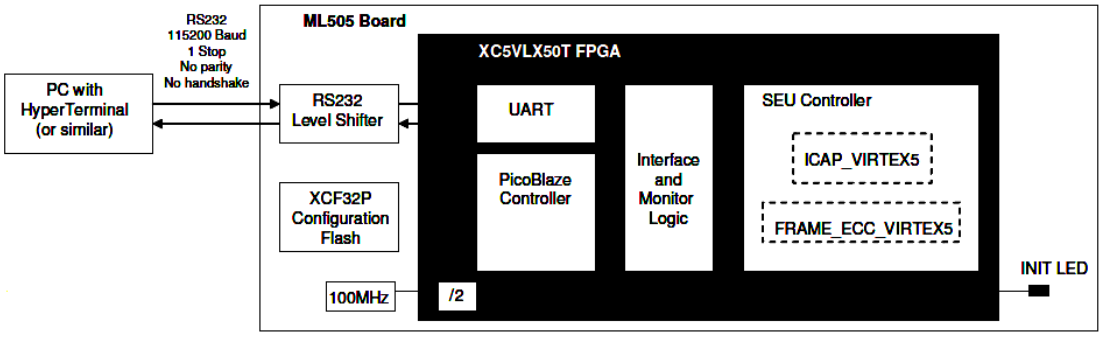
5.2.2.VIRTEX-5 SEU Monitor

Το SEU Controller macro συνδέεται στην ML505 και επικοινωνεί με τον χρήστη, μέσω καλωδίου σειριακής διασύνδεσης RS232 (UART) που συνδέει πλακέτα και υπολογιστή, στον οποίο τρέχει εφαρμογή HyperTerminal (Εικόνα 5.6). Στο παράθυρο του HyperTerminal εμφανίζεται το μενού εντολών του SEU Controller και μέσω αυτού ο χρήστης εισάγει εντολές και διαβάζει την κατάσταση της



Εικόνα 5.6: Διασύνδεση HyperTerminal

συσκευής. Οι εντολές από τον υπολογιστή κατευθύνονται αρχικά στον Picoblaze, που μεταφράζονται και κατόπιν εφαρμόζονται στον ελεγκτή, που εκτελούνται. Τα αποτελέσματα ομοίως προβάλλονται στην οθόνη του χρήστη ακολουθώντας την αντίστροφη πορεία.



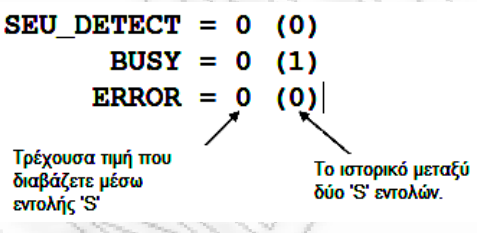
Εικόνα 5.7: Δομικό διάγραμμα διασύνδεση του SEU Controller με τον Picoblaze και τον χρήστη

Η κατάλληλη επιλογή των εντολών επιτρέπει στον χρήστη την εναλλαγή μεταξύ modes λειτουργίας, την εισαγωγή σφαλμάτων, την διόρθωσή τους και τέλος την ανίχνευσή τους. Κατά την έναρξη της επικοινωνίας στην οθόνη προβάλλεται η εικόνα 5.6.

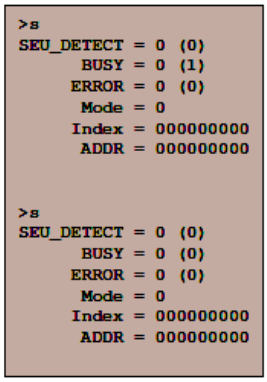
Οι εντολές που έχει δυνατότητα να εκτελέσει ο χρήστης είναι η εξής:

- **'S' – Macro Status**:

Η εντολή θεωρείται από τις πιο χρηστικές του ελεγκτή μιας και χρησιμοποιείται για την παρατήρηση των τριών εξόδων του SEU Controller, επιβεβαιώνει το mode λειτουργίας και δίνει τις τιμές των δεικτών εισαγωγής λάθους (ADDR και Index). Όταν ο Picoblaze λάβει μία εντολή status, διαβάζει άμεσα τις τιμές των ακροδεκτών SEU_DETECT, BUSY και ERROR και τις αποδίδει στο HyperTerminal. Αν και οι πληροφορίες αυτών των τιμών εξόδου είναι εξαιρετικά χρήσιμες, αναμένεται εμφάνιση αιχμών στους ακροδέκτες, τις οποίες δεν θα απεικονίσει η εντολή status ακόμα και αν τρέχει διαρκώς. Γι' αυτό στο κύκλωμα εισάγονται επιπλέον κυκλώματα που μανταλώνουν την κατάσταση των εξόδων, δηλαδή οποιοδήποτε σε λογικό υψηλό συνέβη από την τελευταία S εντολή. Έτσι, κατά την εκτέλεση της εντολής διαβάζεται η τιμή αυτών των μανταλωτών και κατόπιν μηδενίζονται. Για παράδειγμα, αν ένα SEU συμβεί και ο ελεγκτής βρισκόταν σε λειτουργία αυτόματης διόρθωσης (mode = 1), η ανίχνευση και διόρθωση θα επιτυγχανόταν μέσα σε λίγα milliseconds. Κατά την διάρκεια αυτού του φαινομένου οι έξοδοι SEU_DECTECT και BUSY θα οδηγούνταν σε λογικό υψηλό και οι μανταλωτές θα συγκρατούσαν αυτή την τιμή. Έτσι, κατά την επόμενη Status εντολή θα φαινόταν ότι είχε συμβεί μια ανίχνευση και μία διόρθωση (τιμές στις παρενθέσεις – Εικόνα 5.8), αν και η τρέχουσα τιμή των pins είναι σε λογικό χαμηλό (low).



Εικόνα 5.8: Παράδειγμα εντολής Status



Εικόνα 5.9: Η εντολή Status στο HyperTerminal

- **'C' – CRC Scan Active:**

Η έξοδος DETECTION_ACTIVE του ελεγκτή πρέπει να παράγει κανονικούς παλμούς όταν η VIRTEX-5 συσκευή πραγματοποιεί σάρωση μέσω Readback CRC. Όταν δοθεί η εντολή 'C', ο Picoblaze θα απαντήσει με ένα μήνυμα που θα επιβεβαιώνει ότι η σάρωση είναι ενεργή ή ανενεργή

ανάλογα με τον αριθμό των παλμών που θα καταμετρήσει. Εν συνεχεία θα αναφερθούν πληροφορίες για το ρυθμό σάρωσης και τον αριθμό των frames που σαρώνονται.

Ουσιαστικά, ο Picoblaze, θα μετρήσει του παλμούς στην έξοδο DETECTION_ACTIVE για περίπου ένα δευτερόλεπτο (η καθυστέρηση παρατηρείται στην μικρή αργοπορία κατά την επιστροφή του αποτελέσματος). Σύμφωνα με την εικόνα 10.11 η XC5VLX50T σαρώνεται διαρκώς, μιας και χρονίζεται στα 50MHz και διαθέτει 8663 πλαίσια.. Γνωρίζοντας ότι απαιτούνται 41 κύκλοι ρολογιού για την σάρωση κάθε frame, η πλήρης σάρωση της συσκευής απαιτεί 355.183 κύκλους, που στα 50MHz είναι 7,1ms χρόνος σάρωσης και άρα 140,8 πλήρης κύκλους σάρωσης το δευτερόλεπτο. Η εντολή 'C' παράγει ακέραιους αριθμούς και γι' αυτό η τιμή που παρέχει είναι είτε 140 είτε 141 σαρώσεις το δευτερόλεπτο.

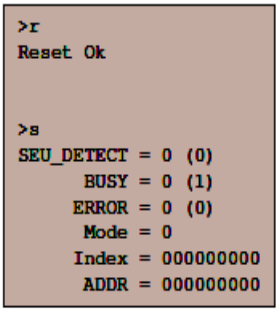
Το Readback CRC πρέπει να είναι πάντοτε ενεργό, εκτός αν ο ελεγκτής διακόψει την σάρωση για να έχει πρόσβαση στην μνήμη διαμόρφωσης και να εισάγει ή να διορθώσει σφάλμα. Είναι σχετικά απίθανο να προκύψει άλλο αποτέλεσμα στην εντολή 'C'. Στην παρούσα εφαρμογή, το κύκλωμα που χρησιμοποιεί ο Picoblaze για να εξετάσει το σήμα DECTECTION_ACTIVE και ο SEU Controller, συγχρονίζονται με το ίδιο ρολόι, κάτι που τυπικά δεν θεωρείται ορθό. Αν η εφαρμογή που βρίσκεται υπό δοκιμή έχει υψηλές απαιτήσεις αξιοπιστίας (όχι στην παρούσα εφαρμογή), τότε χρειάζεται ανεξάρτητη παρατήρηση.

```
>c
CRC Scan Active (141 CRC Scans per second)
Device Frame Count = 8663

>c
CRC Scan Active (140 CRC Scans per second)
Device Frame Count = 8663

>c
CRC Scan Active (141 CRC Scans per second)
Device Frame Count = 8663
```

Εικόνα 5.10 Η εντολή 'C' στο HyperTerminal

- **'R' – Reset Macro:**

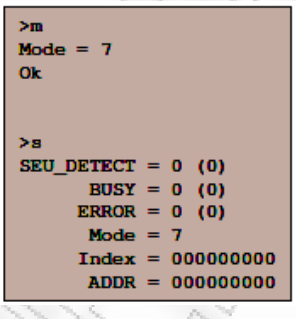
Η εντολή προφανώς επανεκκινεί τον SEU controller, μέσω παλμού θετικού μετώπου που οδηγεί ο Picoblaze στην είσοδο RST. Ο παλμός RST παραμένει σε λογικό υψηλό για 80ns (4 κύκλους ρολογιού των 50 MHz), που είναι αρκετά μεγαλύτερος χρόνος, από τον ένα παλμό που απαιτείται. Παρόλα αυτά, αν ο ελεγκτής βρίσκεται σε κατάσταση BUSY = High, τότε θα αγνοήσει τον παλμό RST, ώστε να μην διαταραχθεί η διαμόρφωση της συσκευής και οδηγηθεί σε άγνωστη κατάσταση. Γι' αυτό το λόγο ο Picoblaze πρώτα εξετάζει την τιμή του BUSY και αν διαπιστώσει ότι είναι σε λογικό υψηλό, τότε επιστρέφει απάντηση 'Reset Failed'. Έτσι, αν στην υπό δοκιμή εφαρμογή υπάρξει κάποιο reset σήμα, τότε και αυτό με τη σειρά του, πρέπει να εξετάζει το σήμα BUSY πριν εφαρμοστεί. Τέλος, πρέπει να σημειωθεί ότι όταν η συσκευή προσπελαύνετε από το JTAG, τότε το reset γίνεται αυτόματα και διακόπτει τον ελεγκτή σε οποιαδήποτε τιμή και αν είναι το BUSY.



Εικόνα5.11: Η εντολή 'R' στο HyperTerminal

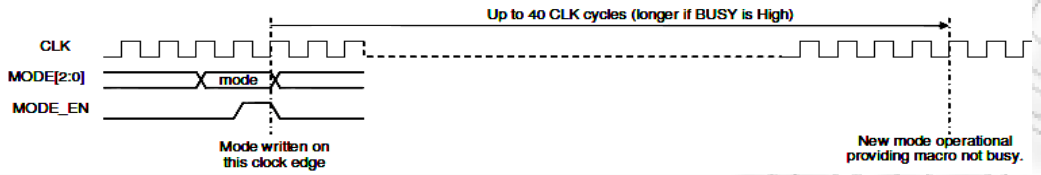
- **'M' – Set Mode**

Η εντολή χρησιμοποιείται για να ορίσει το mode λειτουργίας του SEU controller, με τιμές μεταξύ 0-7. Όταν η εντολή αναγνωσθεί από τον PicoBlaze, τότε επιστρέφει ερώτηση για το πιο mode επιθυμεί ο χρήστης. Μόνο με την εντολή 'S' επιβεβαιώνεται σε πιο mode εργάζεται ο ελεγκτής και το προεπιλεγμένο mode είναι το 0. Επιπρόσθετα, αν και τα modes 2 και 3 είναι κρατημένα, μπορούν να εφαρμοστούν με τον ελεγκτή να παραμένει ενεργός και να συνεχίζει την σάρωση, αλλά ο PicoBlaze θα εμποδίσει την οποιαδήποτε προσπάθεια για εισαγωγή σφαλμάτων.



Εικόνα 5.12: Η εντολή 'M' στο HyperTerminal

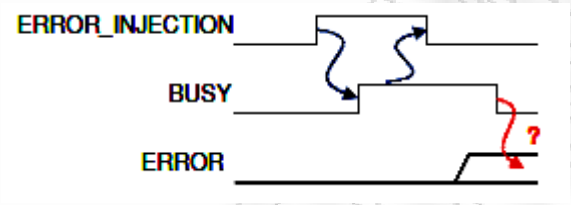
Ο PicoBlaze χρησιμοποιεί μια καθορισμένη έξοδο για να γράψει απευθείας στον καταχωρητή mode του ελεγκτή (το σήμα MODE_EN προκύπτει από αποκωδικοποίηση των σημάτων 'port_id' και 'write_strobe'). Ο ελεγκτής χρειάζεται έως και 40 κύκλους για να αλλάξει mode και η διεργασία καθυστερεί περισσότερο αν η αλλαγή του mode ξεκινήσει όταν το BUSY = High, όπου θα πρέπει να αναμένει να ολοκληρώσει ο ελεγκτής την τρέχουσα εργασία. Μετά την αλλαγή, μπορεί η εισαγωγή λάθους να εισαχθεί γρηγορότερα από το επιτρεπτό και γι' αυτό το κύκλωμα φροντίζει να εξασφαλίζει την απαραίτητο χρονικό διάστημα μεταξύ των δύο εντολών.



Εικόνα 5.13: Χρονισμός για την αλλαγή του mode

- **'P' – Pulse ERROR_INJECT**

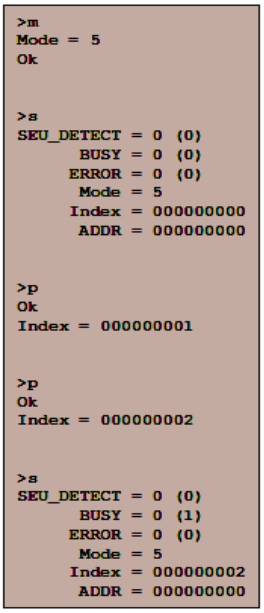
Η εντολή χρησιμοποιείται για την εφαρμογή ενός απλού παλμού στην είσοδο ERROR_INJECT του SEU controller, ώστε να τελεστούν οι λειτουργίες των modes 4,5,6 και 7. Αν παρόλα αυτά, ένας παλμός εφαρμοστεί είσοδο ERROR_INJECT όταν ο ελεγκτής βρίσκεται στα modes λειτουργίας 0,1,2 και 3, τότε ο PicoBlaze αποτρέπει την εκτέλεση της εντολές και απαντά με το μήνυμα Failed. Ο παλμός εφαρμόζεται όταν το σήμα BUSY = Low και ακολουθεί χειραψία μεταξύ των δύο σημάτων, όπως φαίνεται στην εικόνα 10.15.Για την ολοκλήρωση της εντολής και την επιστροφή μηνύματος επιβεβαίωσης στον χρήστη ο PicoBlaze εξετάζει το σήμα ERROR.



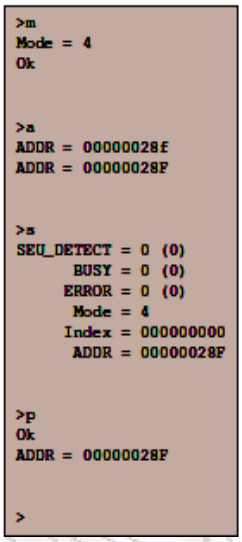
Εικόνα 5.14: Χειραψία των σημάτων ERROR_INJECT και BUSY

- **'A' – Set ADDR**

Η εντολή 'A' χρησιμοποιείται για τον καθορισμό της τιμής των 35 ψηφίων του ADDR, κάτι που καθορίζεται στον ελεγκτή σε mode λειτουργίας 4. Πριν την 'A' λοιπόν, ο ελεγκτής εισάγεται σε mode = 4 και κατόπιν εκτελείται η 'A', όπου ο PicoBlaze επιστρέφει ερώτηση για την επιθυμητή τιμή της ADDR. Ο χρήστης πρέπει να εισάγει 9 δεκαεξαδικά ψηφία μεταξύ 000000000 και 7FFFFFFFF (πεζά ή κεφαλαία) και ακολούθως να εκτελέσει εντολή 'P' για την εισαγωγή σφάλματος . Αν η εισαγωγή αποτύχει συνεπάγεται ότι η τιμή της ADDR αντιστοιχεί σε ψηφίο που δεν υπόκεινται σε. Τέλος, όσο μεγαλύτερη είναι η τιμή της ADDR τόσο περισσότερος χρόνος απαιτείται για την εκτέλεση της εντολής.



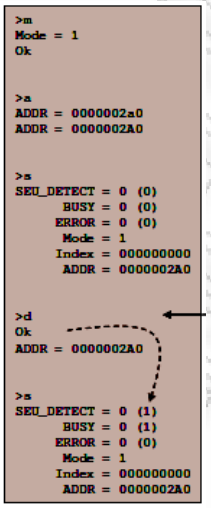
Εικόνα 5.15: Η εντολή 'P' στο HyperTerminal



Εικόνα 5.16: Η εντολή 'A' στο HyperTerminal

- **'D' – Inject SEU at ADDR**

Η εντολή 'D' απλοποιεί τις διαδικασίες που πρέπει να ακολουθηθούν για την εισαγωγή σφάλματος σε συγκεκριμένη ADDR. Ουσιαστικά, είναι υπεύθυνη για την αλλαγή του mode λειτουργίας



Εικόνα 5.17: Η εντολή 'D' στο HyperTerminal

σε 4, την εφαρμογή παλμού στο ERROR_INJECT, ώστε να εισαχθεί σφάλμα σε ADDR που έχει καθοριστεί προηγουμένως και τελικώς την επιστροφή του ελεγκτή στο mode λειτουργίας που ήταν πριν την εφαρμογή της εντολής.

- **'I' & 'Z' – Increment & Zero Index**

Οι εντολές 'I' και 'Z' αυτοματοποιούν την λειτουργία των modes 5 και 6 , αυξάνοντας ή

```
>s
SEU_DETECT = 0 (0)
      BUSY = 0 (0)
     ERROR = 0 (0)
      Mode = 1
     Index = 000000006
      ADDR = 00000028F


>z
Ok
Index = 000000000


>i
Ok
Index = 000000001


>i
Ok
Index = 000000002


>s
SEU_DETECT = 0 (0)
      BUSY = 0 (1)
     ERROR = 0 (0)
      Mode = 1
     Index = 000000002
      ADDR = 00000028F
```

Εικόνα 5.18: Οι εντολές 'I' και 'Z' στο HyperTerminal

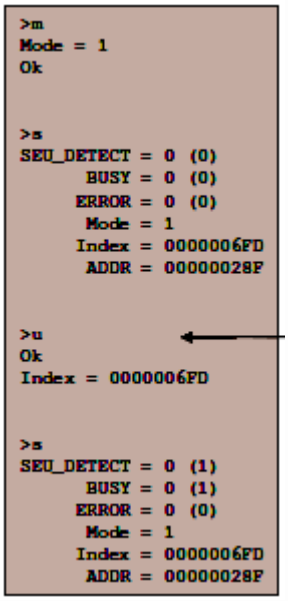
μηδενίζοντας την τιμή του index, αντίστοιχα. Ο τρόπος εκτέλεσης των εντολών είναι ο ίδιος με τον αυτόν της εντολής 'D'.

- **'U' – Inject SEU at Index**

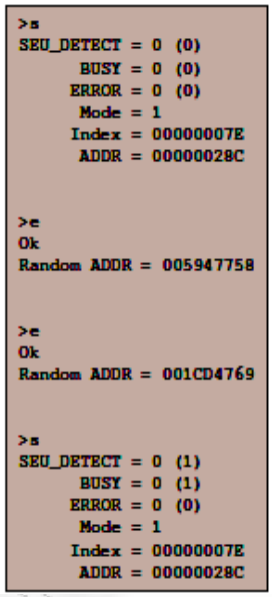
Η εντολή 'U' αυτοματοποιεί τις διαδικασίες που πρέπει να ακολουθηθούν για την εισαγωγή σφάλματος σε συγκεκριμένο index. Ουσιαστικά, είναι υπεύθυνη για την αλλαγή του mode λειτουργίας σε 7, την εφαρμογή παλμού στο ERROR_INJECT, ώστε να εισαχθεί σφάλμα στο index που έχει καθοριστεί προηγουμένως και τελικώς την επιστροφή του ελεγκτή στο mode λειτουργίας που ήταν πριν την εφαρμογή της εντολής. Η εντολή αποτυγχάνει αν η τιμή του index δεν επιτρέπει αντιστοιχεί σε μη επιτρεπτή θέση.

- **'E' – Simulate Random SEU**

Η εντολή 'E' εισάγει ένα σφάλμα σε μια τυχαία θέση μέσα στην συσκευή και αποτελεί τον καλύτερο τρόπο εξομοίωσης των SEUs σε μία εφαρμογή. Ο PicoBlaze διαθέτει έναν LFSR (Linear



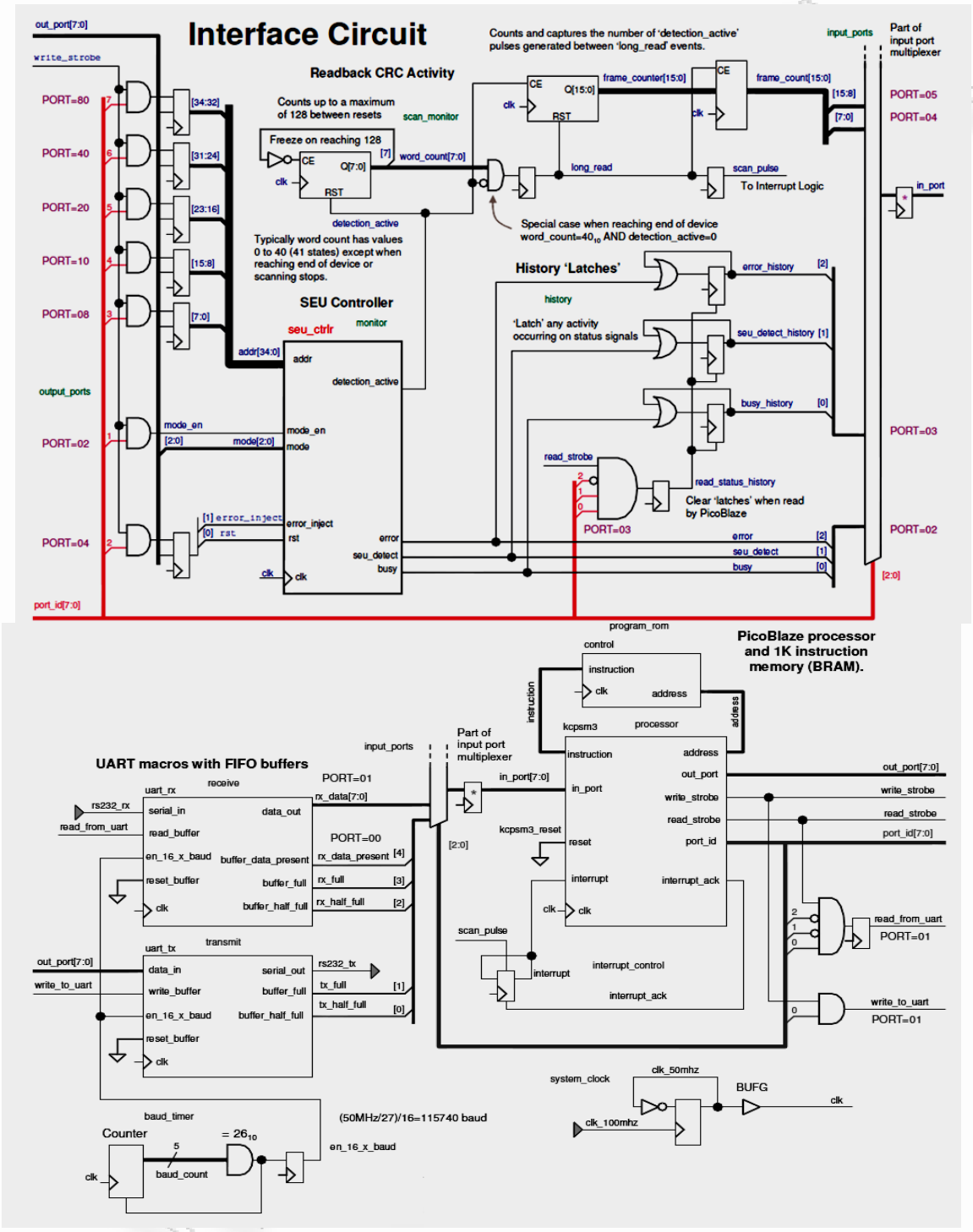
Εικόνα 5.19: Η εντολή 'U' στο HyperTerminal



Εικόνα 5.20: Η εντολή 'E' στο HyperTerminal

Feedback Shift Register) μετρητή των 27 ψηφίων, που παράγει ψευδοτυχαίους αριθμούς, ικανοποιητικά μεγάλους ώστε να καλύπτουν το εύρος της ADDR, ακόμα και για τις μεγαλύτερες VIRTEX-5 συσκευές. Ο LFSR αυξάνει κάθε 880ns, αναμένοντας μια εντολή ‘E’, κατά την οποία η τιμή της ADDR θα εξισωθεί με την τρέχουσα τιμή του μετρητή, ο ελεγκτής θα τεθεί σε mode 4, θα εισαχθεί σφάλμα μέσω του ERROR_INJECT και τέλος ο ελεγκτής θα επιστρέψει στο mode λειτουργίας και η ADDR στην τιμή, που είχαν πριν την εντολή.

Ακολουθεί η διασύνδεση του PicoBlaze με τον SEU Controller



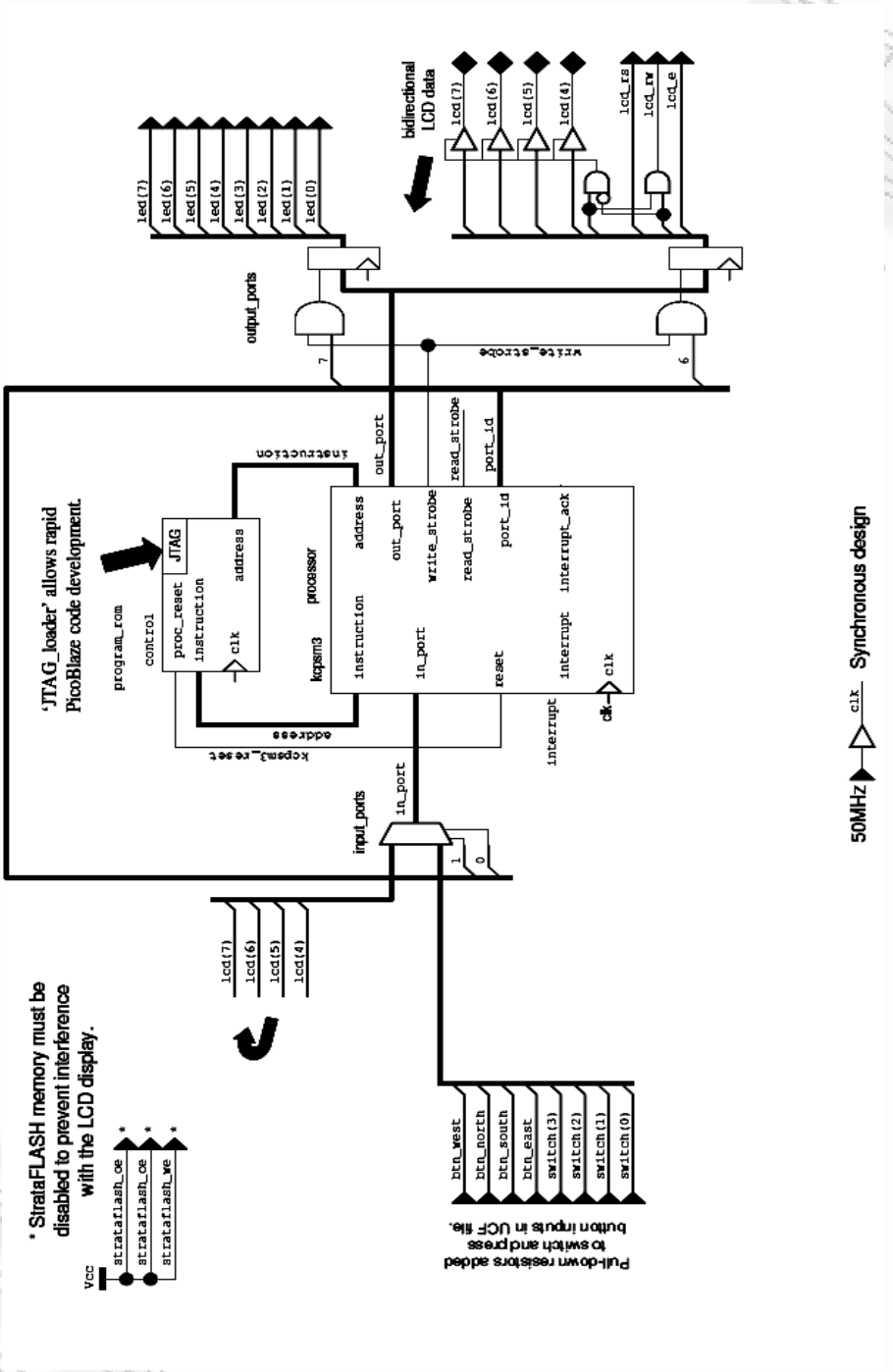
Εικόνα5.21: Διασύνδεση του PicoBlaze με τον SEU Controller

5.2.3. LCD Display

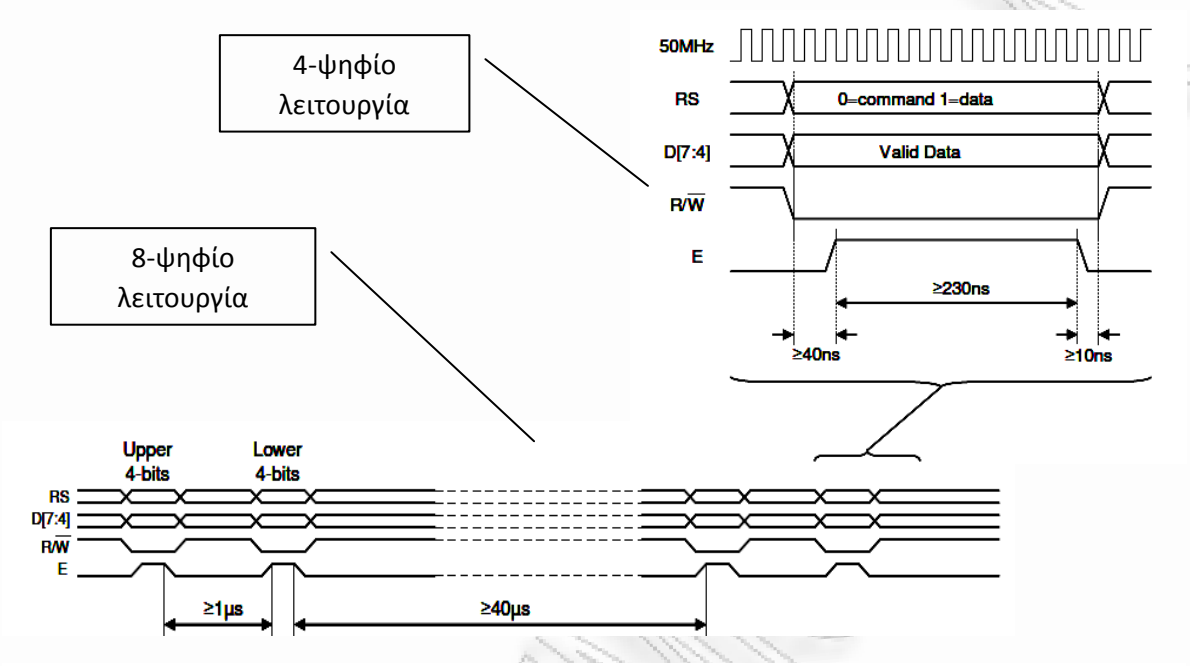
Η εφαρμογή χρησιμοποιεί την LCD οθόνη της συσκευής, για να προβάλλει ένα απλό μήνυμα 2 γραμμών, υπό κύλιση. Η οθόνη για να προβάλλει μηνύματα χρησιμοποιεί χαρακτήρες ASCII καθώς και επιτρέπει την δημιουργία χαρακτήρων από τον χρήστη. Το μήνυμα και η λειτουργία κύλισης ουσιαστικά υλοποιούνται μέσω του PicoBlaze, ο οποίος αναλαμβάνει να παρέχει τα δεδομένα σωστά και συγχρονισμένα. Επίσης, αναλαμβάνει και την διασύνδεση των push buttons και switches με τα LEDs της πλακέτας. Δομικό διάγραμμα της διασύνδεσης φαίνεται στην Εικόνα 5.22.

Ο ρυθμός λειτουργίας της οθόνης είναι σχετικά μικρός και στην παρούσα εφαρμογή η κύλιση πραγματοποιείται κάθε 0,5 sec, χρόνος που θεωρείται οριακός, για να υπάρχει ικανοποιητική ευκρίνεια. Ο χαμηλός ρυθμός απόδοσης σχετίζεται με τα σήματα της οθόνης που χρησιμοποιούνται για την επικοινωνία και παρόλο που η συχνότητα λειτουργίας είναι τα 50 MHz, η αντίδραση της οθόνης παρουσιάζεται εξαιρετικά αργή. Γι’ αυτό εισάγεται η χρήση του PicoBlaze, ώστε να ενσωματώσει τις απαιτούμενες λειτουργίες και να χειριστεί έτσι αποδοτικά τις καθυστερήσεις, καθώς και να αναλάβει τον έλεγχο των περιεχομένων προβολής στην οθόνη.

Ο χρονισμός μίας απλή εγγραφής των 4 ψηφίων φαίνεται στην εικόνα 5.23. Στο διάγραμμα απεικονίζεται ο ελάχιστος επιτρεπόμενος αριθμός επανάληψης, για τις λειτουργίες setup, hold και enable pulse, στα 50 MHz (20ns περίοδος).Τα δεδομένα D[7:4], το σήμα επιλογής καταχωρητή RS (Register Select) και το σήμα ελέγχου της εγγραφής RW (write control) πρέπει να διαμορφώνονται τουλάχιστον 40ns, προτού το σήμα επίτρεψης ‘E’ οδηγηθεί σε λογικό υψηλό. Επιπρόσθετα, το E πρέπει να διατηρείται σε λογικό υψηλό για τουλάχιστον 230ns, δηλαδή για 12 κύκλους ρολογιού.



Εικόνα 5.22: Διασύνδεση PicoBlaze με LCD Monitor και οδήγηση των LEDs



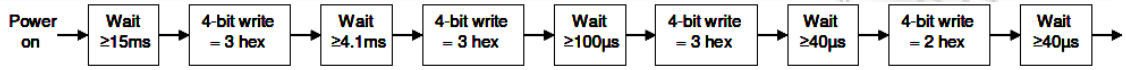
Εικόνα 5.23: Χρονισμός LCD Display

Αφού καθοριστούν οι αρχικές τιμές και εξασφαλιστεί η επικοινωνία με την οθόνη, αναγνωρίζουμε ότι όλα τα δεδομένα που οδηγούνται είναι είτε ASCII χαρακτήρες 8-ψηφίων, είτε bytes πληροφορίας, είτε διευθύνσεις των 8-ψηφίο. Άρα κάθε μεταφορά είναι συνολικά 8-ψηφία και πρέπει να διαιρεθεί σε δύο των 4-ψηφίων, μιας και η απλή έγγραφη στην οθόνη έχει εύρος 4-ψηφία. Η χρονική απόσταση των δύο τμημάτων του τελικού δεδομένου είναι τουλάχιστον 1μs και μεταξύ δύο διαδοχικών δεδομένων 40μs. Αυτές οι χρονικές αποστάσεις συναθροίζονται με τον χρόνο εκκαθάρισης (clear) της οθόνης που είναι στα 1,64ms.

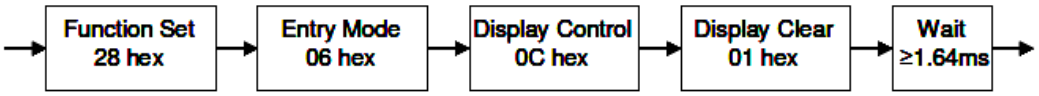
Η χρήση του PicoBlaze στον σχεδιασμό είναι για να ενσωματωθεί η επικοινωνία με την LCD οθόνη, 100% στο λογισμικό. Το γεγονός ότι ο μικροεπεξεργαστής είναι από φύσης ένα ακολουθιακό κύκλωμα συνεπάγεται ότι οι απαιτούμενες χρονικές καθυστερήσεις μπορούν να παραχθούν απλώς εκτελώντας τον κατάλληλο αριθμό εντολών. Ο PicoBlaze απλοποιεί ακόμα περισσότερο το πρόβλημα του χρονισμού μιας και όλες του οι εντολές εκτελούνται σε 2 κύκλους ρολογιού, υπό οποιεσδήποτε συνθήκες. Έτσι, δημιουργούμε ένα πρόγραμμα βρόγχου 25 εντολών, με το οποίο εισάγουμε καθυστέρηση της τάξεως του 1μs και το οποίο χρησιμοποιούμε ως βάση για να παράγουμε ποικίλες χρονικές καθυστερήσεις.

Η αρχικοποίηση της οθόνης απαιτείται για να μπορεί να λειτουργήσει για πρώτη φορά και να εξασφαλίσει επικοινωνία με την εφαρμογή. Η διαδικασία εκτελείται άπαξ και καθ' όλη την διάρκεια ο μικροεπεξεργαστής είναι διαθέσιμος, ώστε να εκτελεί και άλλες λειτουργίες, συμπεριλαμβανομένου και του ελέγχου της οθόνης.

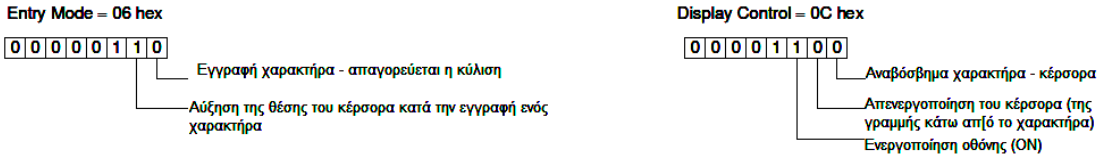
Στο πρώτο μέρος της αρχικοποίησης πρέπει να καθοριστεί ότι χρησιμοποιείται διασύνδεση των 4 –ψηφίων (Εικόνα 5.24). Η καθυστέρηση που εισάγει η διαδικασία σε ορισμένα σημεία είναι της τάξης



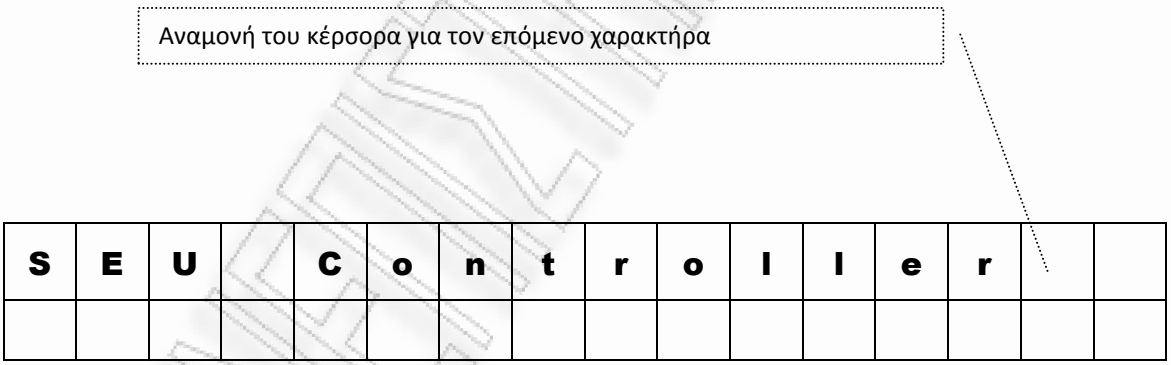
Εικόνα 5.24: Αρχικοποίηση των διασυνδέσεων της LCD οθόνης



Εικόνα 5.25: Επικοινωνία με την LCD οθόνη



Εικόνα 5.26: Λειτουργία εισαγωγής και ελέγχου της οθόνης



Εικόνα 5.27: Διάταξη θέσεων οθόνης

των ms. Όταν η διασύνδεση επιτευχθεί, διαμορφώνεται η επικοινωνία σε εγγραφές των 8 ψηφίων και επειδή πρόκειται για εντολές εγγραφής το RS πρέπει να οδηγηθεί σε Low (Εικόνα 5.25). Ακολούθως, καθορίζεται ο τρόπος απεικόνισης της οθόνης. Έτσι, χωρίζεται σε 2 γραμμές, στις οποίες προβάλλονται χαρακτήρες με μέγεθος 5x7 pixels και η επικοινωνία παραμένει στα 4 καλώδια (Εικόνα 5.26).

Η πιο απλή εισαγωγή στην οθόνη είναι αυτή των χαρακτήρων ASCII και επειδή θεωρούνται δεδομένα το RS οδηγείται σε λογικό υψηλό. Στην εφαρμογή εισάγουμε στην πρώτη γραμμή το κείμενο

«**SEU Controller**» και στην δεύτερη το κείμενο «**VIRTEX – 5 ML505**». Οι χαρακτήρες θα γραφούν σειριακά, αλλά όταν φτάσουν στο τέλος της γραμμής, δεν θα μεταβούν αυτόματα στην δεύτερη γραμμή, γιατί οι διευθύνσεις μνήμης των θέσεων της οθόνης δεν είναι διαδοχικές, από γραμμή σε γραμμή. Για να τοποθετηθεί ο κέρσορας στην δεύτερη γραμμή, πρέπει να γραφεί η διεύθυνση 8 ψηφίων του πρώτου χαρακτήρα της και επειδή η εγγραφή εδώ είναι εντολή το RS θα οδηγηθεί σε λογικό χαμηλό.

5.3 Δοκιμές

Όπως προαναφέρθηκε η εφαρμογή αναπτύχθηκε στο ISE Project Navigator 10.1. Στον Πίνακα 5.1φαίνονται οι τα σήματα εισόδου και εξόδου της εφαρμογής. Ακολουθεί ο πίνακας 5.2στο οποίο φαίνεται το ποσοστό του FPGA που καταλαμβάνει η εφαρμογή, ενώ στην Εικόνα 5.28 φαίνονται οι είσοδοι και έξοδοι του κυκλώματος. Έτσι, ως είσοδοι ορίζονται πέραν του ρολογιού, τα τέσσερα από τα πέντε push buttons της πλακέτας και τέσσερα από τα 8 switches και τέλος η γραμμή receive στην σειριακή επικοινωνία με τον υπολογιστή μέσω του RS232. Ως έξοδοι λαμβάνονται τα 8 LEDs της πλακέτας, τα σήματα επικοινωνίας με την LCD οθόνη (RW, RS, E), τα δεδομένα που καταλήγουν στην οθόνη (D[7:0]) και τέλος η γραμμή transmit του RS232.

Θύρες	**Εισόδου/Εξόδου**	**Περιγραφή**
Switch[3:0]	IN	Τα σήματα των τεσσάρων διακοπτών της πλακέτας
Btn_east	IN	Τα σήματα των τεσσάρων push buttons της πλακέτας
Btn_north	IN	
Btn_south	IN	
Btn_west	IN	
Rs232_rx	IN	Σήμα σειριακής επικοινωνίας από τον υπολογιστή στην πλακέτα (receive)
CLK	IN	Ρολόι χρονισμού του κυκλώματος (50 MHz)
Led[7:0]	OUT	Τα σήματα ενεργοποίησης των 8 LEDs της πλακέτας
LCD_e	OUT	Σήμα επίτρεψης του τσιπ της οθόνης (LCD Enable)
LCD_rs	OUT	Σήμα επιλογής καταχωρητή

		οθόνης (LCD Register Select)
LCD_rw	OUT	Σήμα εγγραφής στην οθόνη (LCD Write)
LCD_d[7:4]	OUT	Δεδομένα που αποστέλλονται στην οθόνη
RS232_tx	OUT	Σήμα σειριακής επικοινωνίας από την πλακέτα προς τον υπολογιστή (transmit)

Πίνακας 5.1: Θύρες εισόδου και εξόδου της εφαρμογής

Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	464	28800	1%
Number used as Flip Flops	464		
Number of Slice LUTs	601	28800	2%
Number used as logic	480	28800	1%
Number using O6 output only	378		
Number using O5 output only	20		
Number using O5 and O6	82	7680	1%
Number used as Memory	114		
Number used as Dual Port RAM	24		
Number using O5 and O6	24		
Number used as Single Port RAM	54		
Number using O6 output only	54		

Number used as Shift Register	36		
Number using O6 output only	36		
Number used as exclusive route-thru	7		
Number of route-thrus	30	57600	1%
Number using O6 output only	27		
Number using O5 output only	3		
Slice Logic Distribution			
Number of occupied Slices	269	7200	3%
Number of LUT Flip Flop pairs used	720		
Number with an unused Flip Flop	256	720	35%
Number with an unused LUT	119	720	16%
Number of fully used LUT-FF pairs	345	720	47%
Number of unique control sets	71		
IO Utilization			
Number of bonded IOBs	26	480	5%
Specific Feature Utilization			
Number of BlockRAM/FIFO	3	60	5%
Number using BlockRAM only	3		
Total primitives used			
Number of 36k BlockRAM used	1		
Number of 18k BlockRAM used	3		
Total Memory used (KB)	**90**	**2160**	**4%**
Number of BUFG/BUFGCTRLs	2	32	6%

Number used as BUFGs	2		
Number of DCM_ADVs	1	12	8%
Number of ICAPs	1	2	50%

Πίνακας 5.2: Ποσοστά χρήσης της συσκευής

Με την βοήθεια του λογισμικού διαμόρφωσης εντοπίστηκαν οι διευθύνσεις των πλαισίων συσχετιζόμενες με τα τμήματα της συσκευής που απασχολούνται από την εφαρμογή (Παράρτημα Β) και ακολούθησαν δοκιμές, ώστε να διαπιστωθεί τόσο η συμπεριφορά του SEU Controller Monitor και ο βαθμός αξιόπιστης συνεργασίας με τον PicoBlaze, όσο και η συμπεριφορά της εφαρμογής καθαυτής.

Επειδή ο PicoBlaze έχει περιορισμένη μνήμη προγράμματος (1.024 εντολές) ήταν αδύνατο να υλοποιήσει μεγάλα προγράμματα δοκιμών, δηλαδή να εισάγει και να διορθώσει σφάλματα σε πολλές διευθύνσεις. Και αυτό γιατί ήδη η μνήμη εντολών του μικροεπεξεργαστή, που σχετίζεται με τον ελεγκτή, περιείχε ένα μεγάλο κώδικα για την επικοινωνία του χρήστη με τον ελεγκτή και την τέλεση των εντολών. Για να επιτευχθούν μεγάλα προγράμματα δοκιμών, θα ήταν αναγκαία η χρήση και ενός τρίτου Picoblaze επεξεργαστή, κάτι που στο παρόν θεωρήθηκε περιττό.

5.3.1. Δοκιμή Α – Εισαγωγή σφαλμάτων στα τμήματα διαμόρφωσης (configuration slices)

Το πείραμα είχε ως στόχο την εισαγωγή λάθους σε συγκεκριμένη διεύθυνση του κυκλώματος, την αναμονή μέσω διαδοχικών εντολών status για τυχόν εμφανής επιπτώσεις στην λειτουργία του κυκλώματος, την διόρθωση του λάθους μέσω mode = 1 και την εισαγωγή λάθους σε νέα διεύθυνση.

- **Δοκιμή Α1**

Με την βοήθεια του PicoBlaze, εισήχθη κώδικας στον SEU Controller με την ακόλουθη μορφή:

```
S x 2          ; δύο εντολές Status
A = 000108e9f  ; ορισμός διεύθυνσης (ADDR)
D              ; εισαγωγή σφάλματος στην διεύθυνση
S x 20         ; 20 εντολές Status για να υπάρξει αρκετός χρόνος για τον οπτικό εντοπισμό
               ; σφάλματος στην τελική λειτουργία της εφαρμογής
M=1            ;αλλαγή σε mode αυτόματης διόρθωσης
S x 10         ; 10 εντολές Status για να δοθεί χρόνος ώστε να ολοκληρωθεί ή διόρθωση
M = 0          ; επιστροφή σε λειτουργία ανίχνευσης κ.ο.κ.
```

Στο περιβάλλον του HyperTerminal καταγράφηκαν τα εξής:

```
>S
SEU_DETECT = 0 (0)
```

```
          BUSY = 1 (1)          ←----------ο ελεγκτής είναι απασχολημένος
         ERROR = 0 (0)
           Mode = 0
           Index = 000000000
          ADDR = 000000000

>S
SEU_DETECT = 0 (0)
          BUSY = 1 (1)
         ERROR = 0 (0)
           Mode = 0
           Index = 000000000
          ADDR = 000000000

>A
ADDR = 000108e9f
ADDR = 000108E9F

>D
Ok
ADDR = 000108E9F

>S
SEU_DETECT = 0 (0)          ←----------δεν εισήχθη λάθος
          BUSY = 0 (1)      ←----------κατά την εισαγωγή λαθών το busy = 1
         ERROR = 0 (0)
           Mode = 0
           Index = 000000000
          ADDR = 000108E9F

>S
SEU_DETECT = 0 (0)
          BUSY = 0 (0)
         ERROR = 0 (0)
           Mode = 0
           Index = 000000000
          ADDR = 000108E9F
```

Συμπέρασμα: Κατά την έναρξη του προγράμματος, ο ελεγκτής εμφανίζεται απασχολημένος σε κάποια λειτουργία, που είναι απροσδιόριστη από τον χρήστη. Η λειτουργία δεν τερματίζεται με αποτέλεσμα η απόπειρα εισαγωγής λάθους να αποβεί άκαρπη.

- **Δοκιμή Α2**

Για την αποφυγή των ανωτέρω, δίνεται περισσότερος χρόνος στον SEU Controller να «εγκλιματιστεί», συσχετιζόμενος λειτουργικά με τον Picoblaze. Έτσι αυξάνουμε τις εντολές Status στην αρχή του κώδικα της δοκιμής, από δύο σε δέκα.

Στο περιβάλλον του HyperTerminal καταγράφηκαν τα εξής:

```
>S
SEU_DETECT = 0 (0)
        BUSY = 1 (1)
       ERROR = 0 (0)
         Mode = 0
         Index = 000000000
        ADDR = 000000000                (+9 status ομοίως)

>A
ADDR = 000108e9f
ADDR = 000108E9F


>S
SEU_DETECT = 0 (0)
        BUSY = 1 (1)
       ERROR = 0 (0)
         Mode = 0
         Index = 000000000
        ADDR = 000108E9F                (+ 4 status ομοίως)

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (1)            ←---------- τι αλλάζει;
       ERROR = 0 (0)
         Mode = 0
         Index = 000000000
        ADDR = 000108E9F

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)            ←---------- ομαλοποίηση λειτουργίας
       ERROR = 0 (0)
         Mode = 0
         Index = 000000000
        ADDR = 000108E9F        (+ 1 status ομοίως)
>D
Ok
ADDR = 000108E9F

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (1)
       ERROR = 0 (0)
         Mode = 0
         Index = 000000000
        ADDR = 000108E9F
```

```
>S
SEU_DETECT = 1 (1)          ←---------- σωστή ανίχνευση λάθους
      BUSY = 0 (0)
     ERROR = 0 (0)
      Mode = 0
      Index = 000000000
     ADDR = 000108E9F              (+ 5 status ομοίως)

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)          ←----------σωστή διόρθωση λάθους
      BUSY = 0 (1)
     ERROR = 0 (0)
      Mode = 1
      Index = 000000000
     ADDR = 000108E9F

>S
SEU_DETECT = 0 (0)
      BUSY = 0 (0)
     ERROR = 0 (0)
      Mode = 1
      Index = 000000000
     ADDR = 000108E9F
```

Συμπέρασμα: ο ελεγκτής κατά την έναρξη της λειτουργίας του παρουσίαζε πάλι κατάσταση BUSY = High, αλλά μετά την εισαγωγή τιμής στην ADDR, το BUSY οδηγήθηκε σε λογικό χαμηλό, μετά το πέρας μία status εντολής (η αιτία παραμένει άγνωστη). Κατόπιν, η εισαγωγή σφάλματος τελέστηκε επιτυχώς, αν και στην τελική λειτουργία του κυκλώματος δεν υπήρξαν εμφανής αλλοιώσεις. Θεωρούμε ότι ο χρόνος που ο PicoBlaze εισάγει τις εντολές στον SEU Controller, ίσως είναι σύντομος και ο ελεγκτής εμφανίζει δυσχέρεια αντίδρασης.

- **Δοκιμή Α3**

Σύμφωνα με το συμπέρασμα της δοκιμής Α2 και με στόχο την εξέταση της εφαρμογής καθαυτής, τελέστηκαν διαδοχικές χειρωνακτικές εφαρμογές λαθών, σε όλες τις διευθύνσεις των πλαισίων διαμόρφωσης (configuration frames) που σχετίζονται με την λειτουργία της LCD και των LEDs. Ο χρόνος αναμονής μετά την εισαγωγή κάθε σφάλματος και την διόρθωση του, ήταν ικανοποιητικός για να εμφανιστούν τυχόν αλλοιώσεις στην εφαρμογή.

Στο περιβάλλον του HyperTerminal καταγράφηκαν τα εξής:

A

ADDR = 00010081F

ADDR = 00010081F

>D

Ok

ADDR = 00010081F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 00010081F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 00010081F

>M

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 00010081F

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 00010081F

>M

Mode = 0

Ok

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 00010081F

>A

ADDR = 000100B9E

ADDR = 000100B9E

>D

Ok

ADDR = 000100B9E

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100B9E

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100B9E

>M

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100B9E

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100B9E

>M

Mode = 0

Ok

>2

?

>A

ADDR = 00100C1E

ADDR = 000100C1E

ADDR = 000100C1E

>D

Ok

ADDR = 000100C1E

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100C1E

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100C1E

>M

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100C1E

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100C1E

```
>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C1E

>A
ADDR = 000100C1F
ADDR = 000100C1F

>D
Ok
ADDR = 000100C1F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C1F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C1F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100C1F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100C1F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C1F

>A
ADDR = 000100C9E
ADDR = 000100C9E

>D
Ok
ADDR = 000100C9E
```

```
>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C9E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C9E

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100C9E

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100C9E

>M
Mode = ,
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C9E
```

```
>A
ADDR = 000100C9F
ADDR = 000100C9F

>D
Ok
ADDR = 000100C9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C9F

>M
```

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100C9F

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100C9F

>M

Mode = 0

Ok

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100C9F

>A

ADDR = 000100CA0

ADDR = 000100CA0

>D

Ok

ADDR = 000100CA0

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100CA0

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100CA0

>M

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100CA0

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100CA0

```
>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000100CA0

>A
ADDR = 000100E1F
ADDR = 000100E1F

>D
Ok
ADDR = 000100E1F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000100E1F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000100E1F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000100E1F

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000100E1F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000100E1F

>A
ADDR = 000100E9F
ADDR = 000100E9F

>
?
```

```
>D
Ok
ADDR = 000100E9F

>S
SEU_DETECT = 1 (1)
      BUSY = 0 (1)
     ERROR = 0 (0)
      Mode = 0
      Index = 000000000
     ADDR = 000100E9F

>S
SEU_DETECT = 1 (1)
      BUSY = 0 (0)
     ERROR = 0 (0)
      Mode = 0
      Index = 000000000
     ADDR = 000100E9F

>M
Mode = 1
Ok
```

```
>S
SEU_DETECT = 0 (1)
      BUSY = 0 (1)
     ERROR = 0 (0)
      Mode = 1
      Index = 000000000
     ADDR = 000100E9F

>S
SEU_DETECT = 0 (0)
      BUSY = 0 (0)
     ERROR = 0 (0)
      Mode = 1
      Index = 000000000
     ADDR = 000100E9F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
      BUSY = 0 (0)
     ERROR = 0 (0)
      Mode = 0
      Index = 000000000
```

```
     ADDR = 000100E9F

>A
ADDR = 000100881
ADDR = 000100881

>F
?

>A
ADDR = 00010881F
ADDR = 00010881F

>D
Ok
ADDR = 00010881F

>S
SEU_DETECT = 1 (1)
      BUSY = 0 (1)
     ERROR = 0 (0)
      Mode = 0
      Index = 000000000
     ADDR = 00010881F
```

```
>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 00010881F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 00010881F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
```

```
Index = 000000000
ADDR = 00010881F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 00010881F

>A
ADDR = 000108B1E
ADDR = 000108B1E

>D
Ok
ADDR = 000108B1E

>S
```

```
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B1E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B1E

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108B1E
```

```
>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108B1E

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B1E

>A
ADDR = 0010B

ADDR = 00108B9E
ADDR = 000108B9E
ADDR = 000108B9E

>D
Ok
ADDR = 000108B9E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B9E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B9E

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108B9E

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108B9E

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
```

BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B9E

>A
ADDR = 000108B9F
ADDR = 000108B9F

>D
Ok
ADDR = 000108B9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B9F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108B9F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108B9F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B9F

>A
ADDR = 000108B9F
ADDR = 000108B9F

>D
Ok
ADDR = 000108B9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0

```
        Index = 000000000
        ADDR = 000108B9F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
        ADDR = 000108B9F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
        Mode = 1
        Index = 000000000
        ADDR = 000108B9F

>S
SEU_DETECT = 0 (0)
```

```
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 1
        Index = 000000000
        ADDR = 000108B9F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
        ADDR = 000108B9F

>A
ADDR = 000108C1F
ADDR = 000108C1F

>D
Ok
ADDR = 000108C1F
```

```
>S
SEU_DETECT = 1 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
        ADDR = 000108C1F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
        ADDR = 000108C1F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
```

Mode = 1
Index = 000000000
ADDR = 000108C1F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108C1F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C1F

>A
ADDR = 000108C9E
ADDR = 000108C9E

>D
Ok
ADDR = 000108C9E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C9E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C9E

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108C9E

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108C9E

>M
Mode = 0
Ok

>A
ADDR = 000108C9F
ADDR = 000108C9F

>D

Ok

ADDR = 000108C9F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108C9F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108C9F

>M

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000108C9F

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000108C9F

>M

Mode = 0

Ok

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108C9F

>A

ADDR = 000108D1F

ADDR = 000108D1F

>D

Ok

ADDR = 000108D1F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108D1F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

```
        ADDR = 000108D1F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
      ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000108D1F

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
      ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000108D1F

>M
Mode = 0
Ok
```

```
>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
      ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000108D1F

>A
ADDR = 000108E1E
ADDR = 000108E1E

>D
Ok
ADDR = 000108E1E

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (1)
      ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000108E1E
```

```
>S
SEU_DETECT = 1 (1)
        BUSY = 0 (0)
      ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000108E1E

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
      ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000108E1E

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
      ERROR = 0 (0)
        Mode = 1
```

Index = 000000000
ADDR = 000108E1E

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E1E

>A
ADDR = 00010831F
ADDR = 00010831F

>D
Ok
ADDR = 00010831F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 00010831F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 00010831F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 00010831F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 00010831F

>A
ADDR = 000108E9E
ADDR = 000108E9E

>D
Ok
ADDR = 000108E9E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)

ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E9E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E9E

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108E9E

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108E9E

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E9E

>A
ADDR = 000108E9F
ADDR = 000108E9F

>D

Ok
ADDR = 000108E9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E9F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)

BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108E9F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E9F

>A
ADDR = 000108F1F
ADDR = 000108F1F

>D
Ok
ADDR = 000108F1F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108F1F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108F1F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)

Mode = 1
Index = 000000000
ADDR = 000108F1F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108F1F

>A
ADDR =

SEU Monitor v1.08b

S - Macro Status
C - CRC Scan Status
R - Reset Macro
M - Set Mode

P - Pulse ERROR_INJECT

A - Set ADDR

D - Inject SEU at ADDR (Mode 4)

I - Increment Index (Mode 5)

Z - Zero Index (Mode 6)

U - Inject SEU at Index (Mode 7)

E - Simulate Random SEU

H - Help

>S

SEU_DETECT = 0 (0)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

>A

ADDR = 000108E9F

ADDR = 000108E9F

>D

Ok

ADDR = 000108E9F

>M

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000108E9F

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000108E9F

>M

Mode = 0

Ok

>A

ADDR = 000108E9E

ADDR = 000108E9E

>D

Ok

ADDR = 000108E9E

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108E9E

```
>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E9E

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108E9E

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
```

```
Index = 000000000
ADDR = 000108E9E

>M
Mode = 0
Ok

>A
ADDR = 000100E9F
ADDR = 000100E9F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100E9F

>D
Ok
ADDR = 000100E9F

>S
```

```
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100E9F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100E9F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100E9F
```

```
>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100E9F

>A
ADDR = 000108B1F
ADDR = 000108B1F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B1F
```

```
>D
Ok
ADDR = 000108B1F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B1F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B1F

>M
Mode = 1
Ok
```

```
>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108B1F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108B1F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
```

ADDR = 000108B1F

>A

ADDR = 000108B1F

ADDR = 000108B1F

>D

Ok

ADDR = 000108B1F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108B1F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108B1F

>M

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000108B1F

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000108B1F

>M

Mode = 0

Ok

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108B1F

>A

ADDR = 000101E1E

ADDR = 000101E1E

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000101E1E

>D

Ok

ADDR = 000101E1E

```
>S
SEU_DETECT = 1 (1)
        BUSY = 0 (1)
      ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000101E1E

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
      ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000101E1E

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
      ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000101E1E

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
      ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000101E1E

>A
ADDR = 000108E1F
ADDR = 000108E1F

>D
Ok
ADDR = 000108E1F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (1)
      ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000108E1F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (0)
      ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000108E1F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
      ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000108E1F
```

```
>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108E1F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E1F

>a
ADDR = 00100A00

ADDR = 000100CA0
ADDR = 000100CA0
```

```
>D
Ok
ADDR = 000100CA0

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100CA0

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100CA0

>M
Mode = 1
Ok
```

```
>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100CA0

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100CA0

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
```

Index = 000000000

ADDR = 000100CA0

>A

ADDR = 0001003

ADDR = 000100E9F

ADDR = 000100E9F

>D

Ok

ADDR = 000100E9F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100E9F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100E9F

>M

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100E9F

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100E9F

>M

Mode = 0

Ok

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100E9F

>A

ADDR = 000100B9F

ADDR = 000100B9F

>D

Ok

ADDR = 000100B9F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

```
        ADDR = 000100B9F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
      ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000100B9F

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
      ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000100B9F

>M
Mode = 0
Ok
```

```
>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
      ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000100B9F

>A
ADDR = 000108B9F
ADDR = 000108B9F

>D
Ok
ADDR = 000108B9F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (1)
      ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000108B9F
```

```
>S
SEU_DETECT = 1 (1)
        BUSY = 0 (0)
      ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000108B9F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
      ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000108B9F

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
      ERROR = 0 (0)
        Mode = 1
```

Index = 000000000
ADDR = 000108B9F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B9F

>A
ADDR = 000108C9F
ADDR = 000108C9F

>D
Ok
ADDR = 000108C9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C9F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108C9F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108C9F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C9F

>A
ADDR = 000101

ADDR = 000108D1F

```
ADDR = 000108D1F

>D
Ok
ADDR = 000108D1F

>A
ADDR = S
ADDR = 000108D1F
ADDR = 000108D1F

>
?

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
        Mode = 0
       Index = 000000000
        ADDR = 000108D1F

>S
SEU_DETECT = 1 (1)
```

```
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
       Index = 000000000
        ADDR = 000108D1F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
        Mode = 1
       Index = 000000000
        ADDR = 000108D1F

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 1
       Index = 000000000
        ADDR = 000108D1F
```

```
>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
       Index = 000000000
        ADDR = 000108D1F

>A
ADDR = 000108C1F
ADDR = 000108C1F

>D
Ok
ADDR = 000108C1F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
```

Mode = 0
Index = 000000000
ADDR = 000108C1F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C1F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108C1F

>S

SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108C1F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C1F

>A
ADDR = 000108C9E
ADDR = 000108C9E

>D
Ok

ADDR = 000108C9E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C9E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C9E

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)

ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108C9E

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108C9E

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C9E

>A
ADDR = 000108C9F
ADDR = 000108C9F

>D
Ok
ADDR = 000108C9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C9F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108C9F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108C9F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)

BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C9F

>A
ADDR = 000100C9E
ADDR = 000100C9E

>D
Ok
ADDR = 000100C9E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C9E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C9E

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100C9E

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100C9E

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C9E

>A
ADDR = 000108D1F
ADDR = 000108D1F

>D
Ok
ADDR = 000108D1F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0

```
        Index = 000000000
        ADDR = 000108D1F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
        ADDR = 000108D1F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
        Mode = 1
        Index = 000000000
        ADDR = 000108D1F

>S
SEU_DETECT = 0 (0)
```

```
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 1
        Index = 000000000
        ADDR = 000108D1F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
        ADDR = 000108D1F

>A
ADDR = 000100C1F
ADDR = 000100C1F

>D
Ok
ADDR = 000100C1F
```

```
>S
SEU_DETECT = 1 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
        ADDR = 000100C1F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
        ADDR = 000100C1F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
```

Mode = 1

Index = 000000000

ADDR = 000100C1F

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100C1F

>M

Mode = 0

Ok

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100C1F

>A

ADDR = 000108D1F

ADDR = 000108D1F

>D

Ok

ADDR = 000108D1F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108D1F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108D1F

>M

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000108D1F

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000108D1F

>M

Mode = 0

Ok

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108D1F

>A
ADDR = 000108B1E
ADDR = 000108B1E

>D
Ok
ADDR = 000108B1E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B1E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)

Mode = 0
Index = 000000000
ADDR = 000108B1E

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108B1E

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108B1E

>M

Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108B1E

>A
ADDR = 000100C1E
ADDR = 000100C1E

>D
Ok
ADDR = 000100C1E

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000

ADDR = 000100C1E

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100C1E

>M

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100C1E

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000100C1E

>M

Mode = 0

Ok

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100C1E

>A

ADDR = 000100C1F

ADDR = 000100C1F

>D

Ok

ADDR = 000100C1F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100C1F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100C1F

>M

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000
ADDR = 000100C1F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100C1F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100C1F

>A
ADDR = 000108E9F

ADDR = 000108E9F

>D
Ok
ADDR = 000108E9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E9F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108E9F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000108E9F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108E9F

>A

ADDR = 000108F1F

ADDR = 000108F1F

>D

Ok

ADDR = 000108F1F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108F1F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108F1F

>M

Mode = S

Mode = S

Mode = M

Mode = 0

Ok

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108F1F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108F1F

>M

Mode = 1

Ok

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000108F1F

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000108F1F

>M

Mode = 0

Ok

```
>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000108F1F

>A
ADDR = 000108B9F
ADDR = 000108B9F

>D
Ok
ADDR = 000108B9F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000108B9F

>S
```

```
SEU_DETECT = 1 (1)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000108B9F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000108B9F

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000108B9F
```

```
>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000108B9F

>A
ADDR = 000108E1F
ADDR = 000108E1F

>D
Ok
ADDR = 000108E1F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (1)
```

```
        ERROR = 0 (0)
          Mode = 0
         Index = 000000000
         ADDR = 000108E1F

>S
SEU_DETECT = 1 (1)
          BUSY = 0 (0)
        ERROR = 0 (0)
          Mode = 0
         Index = 000000000
         ADDR = 000108E1F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
          BUSY = 0 (1)
        ERROR = 0 (0)
          Mode = 1
         Index = 000000000
         ADDR = 000108E1F
```

```
>S
SEU_DETECT = 0 (0)
          BUSY = 0 (0)
        ERROR = 0 (0)
          Mode = 1
         Index = 000000000
         ADDR = 000108E1F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
          BUSY = 0 (0)
        ERROR = 0 (0)
          Mode = 0
         Index = 000000000
         ADDR = 000108E1F

>A
ADDR = 000108C9F
ADDR = 000108C9F

>M
```

```
Mode = 1
Ok

>S
SEU_DETECT = 0 (0)
          BUSY = 0 (0)
        ERROR = 0 (0)
          Mode = 1
         Index = 000000000
         ADDR = 000108C9F

>S
SEU_DETECT = 0 (0)
          BUSY = 0 (0)
        ERROR = 0 (0)
          Mode = 1
         Index = 000000000
         ADDR = 000108C9F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
```

BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108C9F

>A
ADDR = 000100E1F
ADDR = 000100E1F

>D
Ok
ADDR = 000100E1F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100E1F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100E1F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100E1F

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 1
Index = 000000000
ADDR = 000100E1F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000100E1F

>A
ADDR = 000100E9F
ADDR = 000100E9F

>D
Ok
ADDR = 000100E9F

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0

```
        Index = 000000000
        ADDR = 000100E9F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000100E9F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000100E9F

>S
SEU_DETECT = 0 (0)
```

```
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 1
        Index = 000000000
       ADDR = 000100E9F

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000100E9F

>A
ADDR = 000100C9F
ADDR = 000100C9F

>D
Ok
ADDR = 000100C9F
```

```
>S
SEU_DETECT = 1 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000100C9F

>S
SEU_DETECT = 1 (1)
        BUSY = 0 (0)
       ERROR = 0 (0)
        Mode = 0
        Index = 000000000
       ADDR = 000100C9F

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
        BUSY = 0 (1)
       ERROR = 0 (0)
```

Mode = 1		Index = 000000000
Index = 000000000		ADDR = 000100C9F
ADDR = 000100C9F	>M	
	Mode = 0	
	Ok	>S
>S		SEU_DETECT = 0 (0)
SEU_DETECT = 0 (0)		BUSY = 0 (0)
BUSY = 0 (0)	>S	ERROR = 0 (0)
ERROR = 0 (0)	SEU_DETECT = 0 (0)	Mode = 0
Mode = 1	BUSY = 0 (0)	Index = 000000000
Index = 000000000	ERROR = 0 (0)	ADDR = 000100C9
ADDR = 000100C9F	Mode = 0	

Συμπέρασμα: Οι δοκιμές και η εισαγωγή σφαλμάτων στα πλαίσια που απασχολεί η εφαρμογή, δεν οδήγησαν σε ορατές αλλοιώσεις το κύκλωμα ή ακόμα και σε πιθανή κατάρρευση της λειτουργίας, γεγονός που επιδιωκόταν. Άρα, επιβεβαιώνεται η πεποίθηση, ότι η ύπαρξη SEUs σε ένα κύκλωμα δεν συνεπάγεται απαραιτήτως την διαταραχή της λειτουργίας του και τις περισσότερες φορές πρόκειται για γεγονός που εξελίσσεται υπό την πλήρη άγνοια του χρήστη, ακόμα και για ολόκληρο το χρόνο ζωής του κυκλώματος.

5.3.2.Δοκιμή Β: Εισαγωγή σφαλμάτων στα BRams του FPGA

Η δομή του προγράμματος που εφαρμόστηκε είναι ακριβώς η ίδια με αυτή που ακολουθήθηκε προηγουμένως στα πειράματα Α, με αλλαγή στην περιοχή στοχοθέτησης. Επικεντρωνόμαστε στην χειρωνακτική εισαγωγή λαθών σε περιοχές μνήμης που απασχολεί η εφαρμογή.

Στο περιβάλλον του HyperTerminal καταγράφηκαν τα εξής:

	A - Set ADDR	>A ADDR = 100
	D - Inject SEU at ADDR (Mode 4)	ADDR = 0001005A0
	I - Increment Index (Mode 5)	ADDR = 0001005A0
SEU Monitor v1.08b	Z - Zero Index (Mode 6)	
	U - Inject SEU at Index (Mode 7)	>D
S - Macro Status	E - Simulate Random SEU	Ok
C - CRC Scan Status		ADDR = 0001005A0
R - Reset Macro	H - Help	
M - Set Mode		
P - Pulse ERROR_INJECT		>S

```
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 0001005A0


>A
ADDR = 1

ADDR = 0001005A1
ADDR = 0001005A1


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 0001005A1


>M
Mode = 1
Ok


>S
SEU_DETECT = 0 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 1
   Index = 000000000
    ADDR = 0001005A1


>S
SEU_DETECT = 0 (0)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 1
   Index = 000000000
    ADDR = 0001005A1


>M
Mode = 0
Ok
```

```
>A
ADDR = 0001005A2
ADDR = 0001005A2


>S
SEU_DETECT = 0 (0)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 0001005A2


>D
Ok
ADDR = 0001005A2


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 0001005A2


>A
ADDR = 0001005A2
ADDR = 0001005A2


>A
ADDR = 0001005A3
ADDR = 0001005A3


>D
Ok
ADDR = 0001005A3


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
```

```
   Index = 000000000
    ADDR = 0001005A3


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 0001005A3


>M
Mode = 1
Ok


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 1 (1)
    Mode = 1
   Index = 000000000
    ADDR = 0001005A3


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (0)
   ERROR = 1 (1)
    Mode = 1
   Index = 000000000
    ADDR = 0001005A3


>

SEU Monitor v1.08b


S - Macro Status
C - CRC Scan Status
R - Reset Macro
M - Set Mode
P - Pulse ERROR_INJECT
A - Set ADDR

D - Inject SEU at ADDR (Mode
4)
```

```
I - Increment Index (Mode 5)
Z - Zero Index (Mode 6)
U - Inject SEU at Index (Mode 7)
E - Simulate Random SEU

H - Help


>A
ADDR = 0001006A0
ADDR = 0001006A0


>D
Ok
ADDR = 0001006A0


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 0001006A0


>A
ADDR = 0001006A1
ADDR = 0001006A1


>D
Ok
ADDR = 0001006A1


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 0001006A1


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
```

```
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 0001006A1


>M
Mode = 1
Ok


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 1 (1)
    Mode = 1
   Index = 000000000
    ADDR = 0001006A1


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (0)
   ERROR = 1 (1)
    Mode = 1
   Index = 000000000
    ADDR = 0001006A1


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (0)
   ERROR = 1 (1)
    Mode = 1
   Index = 000000000
    ADDR = 0001006A1


>M
Mode = 0
Ok


>S
SEU_DETECT = 1 (1)
    BUSY = 0 (0)
   ERROR = 1 (1)
    Mode = 0
   Index = 000000000
    ADDR = 0001006A1
```

```
>

SEU Monitor v1.08b


S - Macro Status
C - CRC Scan Status
R - Reset Macro
M - Set Mode
P - Pulse ERROR_INJECT
A - Set ADDR

D - Inject SEU at ADDR (Mode 4)
I - Increment Index (Mode 5)
Z - Zero Index (Mode 6)
U - Inject SEU at Index (Mode 7)
E - Simulate Random SEU

H - Help


>S
SEU_DETECT = 0 (0)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000000000


>S
SEU_DETECT = 0 (0)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000000000


>A
ADDR = 000108BA0
ADDR = 000108BA0


>D
Ok
```

```
ADDR = 000108BA0


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108BA0


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108BA0


>M
Mode = 1
Ok


>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000108BA0


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000108BA0


>M
Mode = 0
Ok


>S
```

```
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108BA0


>A
ADDR = 000108BA1
ADDR = 000108BA1


>D
Ok
ADDR = 000108BA1


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108BA1


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108BA1


>M
Mode = 1
Ok


>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000108BA1
```

```
>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000108BA1


>M
Mode = 0
Ok


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108BA1


>A
ADDR = 000108BA2
ADDR = 000108BA2


>D
Ok
ADDR = 000108BA2


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108BA2


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108BA2
```

```
>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000108BA2

>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000108BA2

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108BA2

>A
ADDR = 000108BA3
ADDR = 000108BA3

>D
Ok
ADDR = 000108BA3

>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108BA3

>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108BA3

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000108BA3

>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000108BA3

>M
Mode = 0
Ok

>A
ADDR = 000108F20
ADDR = 000108F20

>D
Ok
ADDR = 000108F20

>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108F20

>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108F20

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000108F20

>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000108F20

>M
Mode = 0
Ok
```

```
>A
ADDR = 000108F21
ADDR = 000108F21

>D
Ok
ADDR = 000108F21

>S
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000108F21

>S
SEU_DETECT = 1 (1)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000108F21

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 1
   Index = 000000000
    ADDR = 000108F21

>S
SEU_DETECT = 0 (0)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 1
   Index = 000000000
    ADDR = 000108F21
```

```
>M
Mode = 0
Ok

>A
ADDR = 000108F22
ADDR = 000108F22

>S
SEU_DETECT = 0 (0)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000108F22

>S
SEU_DETECT = 0 (0)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000108F22

>D
Ok
ADDR = 000108F22

>S
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000108F22

>S
SEU_DETECT = 1 (1)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000108F22
```

```
>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 1
   Index = 000000000
    ADDR = 000108F22

>S
SEU_DETECT = 0 (0)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 1
   Index = 000000000
    ADDR = 000108F22

>M
Mode = 0
Ok

>A
ADDR = 00108F23

ADDR = 000108F23
ADDR = 000108F23

>D
Ok
ADDR = 000108F23

>S
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000108F23
```

```
>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000108F23


>M
Mode = 1
Ok


>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000108F23


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000108F23


>M
Mode = 0
Ok


>A
ADDR = 000300080
ADDR = 000300080


>D
Ok
ADDR = 000300080


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000300080


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000300080


>M
Mode = 1
Ok


>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000300080


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000300080


>M
Mode = 0
Ok


>A
ADDR = 000300085
ADDR = 000300085


>D
Ok
ADDR = 000300085


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000300085


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000300085


>M
Mode = 1
Ok


>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000300085


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000300085


>M
Mode = 0
Ok
```

```
>A
ADDR = 00030008A
ADDR = 00030008A


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 00030008A


>D
Ok
ADDR = 00030008A


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 00030008A


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 00030008A


>M
Mode = 1
Ok


>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 00030008A
```

```
>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 00030008A


>M
Mode = 0
Ok


>A
ADDR = 00030008F
ADDR = 00030008F


>D
Ok
ADDR = 00030008F


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 00030008F


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 00030008F


>M
Mode = 1
Ok


>S
SEU_DETECT = 0 (1)
```

```
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 00030008F


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 00030008F


>M
Mode = 0
Ok


>A
ADDR = 000300090
ADDR = 000300090


>D
Ok
ADDR = 000300090


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000300090


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000300090


>M
```

```
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000300090

>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000300090

>M
Mode = 0
Ok

>A
ADDR = 000300095
ADDR = 000300095

>D
Ok
ADDR = 000300095

>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000300095

>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 000300095

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000300095

>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 000300095

>M
Mode = 0
Ok

>A
ADDR = 00030009A
ADDR = 00030009A

>D
Ok
ADDR = 00030009A

>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 00030009A

>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 00030009A

>M
Mode = 1
Ok

>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 00030009A

>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 00030009A

>M
Mode = 0
Ok

>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 00030009A

>A
ADDR = 00030009F
```

```
ADDR = 00030009F


>D
Ok
ADDR = 00030009F


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 00030009F


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 00030009F


>M
Mode = 1
Ok


>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 00030009F


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 00030009F


>M
```

```
Mode = 0
Ok


>A
ADDR = 0003000A0
ADDR = 0003000A0


>D
Ok
ADDR = 0003000A0


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 0003000A0


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 0003000A0


>M
Mode = 1
Ok


>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 0003000A0


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 0003000A0


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
```

```
     Mode = 1
    Index = 000000000
     ADDR = 0003000A0


>M
Mode = 0
Ok


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 0003000A0


>A
ADDR = 0003000A5
ADDR = 0003000A5


>D
Ok
ADDR = 0003000A5


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 0003000A5


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 0003000A5


>M
Mode = 1
Ok
```

```
>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 0003000A5


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 0003000A5


>M
Mode = 0
Ok


>A
ADDR = 0003000AA
ADDR = 0003000AA


>D
Ok
ADDR = 0003000AA


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 0003000AA


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 0003000AA


>M
Mode = 1
Ok


>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 0003000AA


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 0003000AA


>M
Mode = 0
Ok


>A
ADDR = 0003000AF
ADDR = 0003000AF


>D
Ok
ADDR = 0003000AF


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 0003000AF


>S
SEU_DETECT = 1 (1)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 0
    Index = 000000000
     ADDR = 0003000AF


>M
Mode = 1
Ok


>S
SEU_DETECT = 0 (1)
     BUSY = 0 (1)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 0003000AF


>S
SEU_DETECT = 0 (0)
     BUSY = 0 (0)
    ERROR = 0 (0)
     Mode = 1
    Index = 000000000
     ADDR = 0003000AF


>M
Mode = 0
Ok


>A
ADDR =
```

Συμπέρασμα: παρόμοια με την δομική A3 το κύκλωμα δεν παρουσίασε εμφανής διαταραχές.

5.3.3.Δοκιμή Γ: Εισαγωγή διπλών σφαλμάτων σε γειτονικά τμήματα διαμόρφωσης ή BRAM

Η δοκιμή διεξήχθη χειρωνακτικά και επελέχθησαν τυχαία γειτονικά τμήματα, ώστε να υποβάλλουμε το κύκλωμα σε καταστάσεις που δεν μπορεί να χειριστεί ο SEU Controller. Στόχος η παρατήρηση κάποιας διαταραχής. Επειδή ο SEU δεν έχει την δυνατότητα διόρθωσης διπλών σφαλμάτων, μετά από κάθε εισαγωγή σφάλματος η συσκευή επαναδιαμορφωνόταν (κάθε νέα διαμόρφωση ξεκινά στο HyperTerminal με την SEU Monitor v1.08b)

Το περιβάλλον του HyperTerminal για γειτονικά configuration slices περιείχε τα εξής:

Monitor v1.08b

>s

SEU_DETECT = 0 (0)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000000000

>s

SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000000000

>a

ADDR = 000108e9f

ADDR = 000108E9F

>d

Ok

ADDR = 000108E9F

>s

SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E9F

>s

SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E9F

>a

ADDR = 000108e9e

ADDR = 000108E9E

>d

Ok

ADDR = 000108E9E

>s

SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000108E9E

>s

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108E9E

>m

Mode = 1

Ok

>s

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 1 (1)

Mode = 1

Index = 000000000

ADDR = 000108E9E

>s

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 1 (1)

Mode = 1

Index = 000000000

ADDR = 000108E9E

>m

Mode = 0

Ok

>

SEU Monitor v1.08b

>s

SEU_DETECT = 0 (0)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

>s

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

>a

ADDR = 000108b1f

ADDR = 000108B1F

>d

Ok

ADDR = 000108B1F

>s

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108B1F

>s

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

```
    ADDR = 000108B1F

>a
ADDR = 000108b1e
ADDR = 000108B1E

>d
Ok
ADDR = 000108B1E

>s
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000108B1E

>s
SEU_DETECT = 1 (1)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000108B1E
```

```
>m
Mode = 1
Ok

>s
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 1 (1)
    Mode = 1
   Index = 000000000
    ADDR = 000108B1E

>s
SEU_DETECT = 1 (1)
    BUSY = 0 (0)
   ERROR = 1 (1)
    Mode = 1
   Index = 000000000
    ADDR = 000108B1E

>m
Mode = 0
Ok
```

```
>
SEU Monitor v1.08b

>s
SEU_DETECT = 0 (0)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000000000

>s
SEU_DETECT = 0 (0)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000000000

>a
ADDR = 000100b9e
ADDR = 000100B9E

>d
```

Ok

ADDR = 000100B9E

>d

Failed

ADDR = 000100B9E

>d

Ok

ADDR = 000100B9E

>d

Failed

ADDR = 000100B9E

>d

Ok

ADDR = 000100B9E

>a

ADDR = 000

SEU Monitor v1.08b

>a

ADDR = 000108d1f

ADDR = 000108D1F

>d

Ok

ADDR = 000108D1F

>s

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108D1F

>s

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108D1F

ADDR = 000108c1f

ADDR = 000108C1F

>d

Ok

ADDR = 000108C1F

>s

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108C1F

>m

Mode = 1

Ok

>s

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 1 (1)

Mode = 1

Index = 000000000

ADDR = 000108C1F

>s

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 1 (1)

Mode = 1

Index = 000000000

ADDR = 000108C1F

>

SEU Monitor v1.08b

>s

SEU_DETECT = 0 (0)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

>s

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

>a

ADDR = 000100c1e

ADDR = 000100C1E

>d

Ok

ADDR = 000100C1E

>a

ADDR = 000100ci

ADDR = 000100c1f

ADDR = 000100C1F

>d

Ok

ADDR = 000100C1F

>s

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100C1F

>s

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000100C1F

>m

Mode = 1

Ok

>s

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 1 (1)

Mode = 1

Index = 000000000

ADDR = 000100C1F

>s

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 1 (1)

Mode = 1

Index = 000000000

ADDR = 000100C1F

>

SEU Monitor v1.08b

>a

ADDR = 000108e9f

ADDR = 000108E9F

>d

Ok

ADDR = 000108E9F

>a

ADDR = 000108f1f

ADDR = 000108F1F

>d

Ok

ADDR = 000108F1F

>a

ADDR = 000108b9f

ADDR = 000108B9F

>d

Ok

ADDR = 000108B9F

>a

ADDR = 000100e1f

ADDR = 000100E1F

>d

Ok

ADDR = 000100E1F

>

Το περιβάλλον του HyperTerminal για γειτονικά μπλοκ RAM περιείχε τα εξής:

S	>S	>S
SEU_DETECT = 0 (0)	SEU_DETECT = 1 (1)	SEU_DETECT = 1 (1)
BUSY = 0 (1)	BUSY = 0 (1)	BUSY = 0 (1)
ERROR = 0 (0)	ERROR = 0 (0)	ERROR = 0 (0)
Mode = 0	Mode = 0	Mode = 0
Index = 000000000	Index = 000000000	Index = 000000000
ADDR = 000000000	ADDR = 000108BA0	ADDR = 000108BA1
>S	>S	>S
SEU_DETECT = 0 (0)	SEU_DETECT = 1 (1)	SEU_DETECT = 1 (1)
BUSY = 0 (0)	BUSY = 0 (0)	BUSY = 0 (0)
ERROR = 0 (0)	ERROR = 0 (0)	ERROR = 0 (0)
Mode = 0	Mode = 0	Mode = 0
Index = 000000000	Index = 000000000	Index = 000000000
ADDR = 000000000	ADDR = 000108BA0	ADDR = 000108BA1
>A	>A	>M
ADDR = 000108BA0	ADDR = 000108BA1	Mode = 1
ADDR = 000108BA0	ADDR = 000108BA1	Ok
>D	>D	>S
Ok	Ok	SEU_DETECT = 1 (1)
ADDR = 000108BA0	ADDR = 000108BA1	BUSY = 0 (1)
		ERROR = 1 (1)
		Mode = 1

Index = 000000000

ADDR = 000108BA1

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 1 (1)

Mode = 1

Index = 000000000

ADDR = 000108BA1

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 1 (1)

Mode = 1

Index = 000000000

ADDR = 000108BA1

>

SEU Monitor v1.08b

S - Macro Status

C - CRC Scan Status

R - Reset Macro

M - Set Mode

P - Pulse ERROR_INJECT

A - Set ADDR

D - Inject SEU at ADDR (Mode 4)

I - Increment Index (Mode 5)

Z - Zero Index (Mode 6)

U - Inject SEU at Index (Mode 7)

E - Simulate Random SEU

H - Help

>S

SEU_DETECT = 0 (0)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

>A

ADDR = 000108F21

ADDR = 000108F21

>D

Ok

ADDR = 000108F21

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108F21

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108F21

>A

ADDR = 000108F22

ADDR = 000108F22

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108F22

>D

Ok

ADDR = 000108F22

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108F22

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000108F22

>M

Mode = 1

Ok

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 1 (1)

Mode = 1

Index = 000000000

ADDR = 000108F22

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 1 (1)

Mode = 1

Index = 000000000

ADDR = 000108F22

>

SEU Monitor v1.08b

S - Macro Status

C - CRC Scan Status

R - Reset Macro

M - Set Mode

P - Pulse ERROR_INJECT

A - Set ADDR

D - Inject SEU at ADDR (Mode 4)

I - Increment Index (Mode 5)

Z - Zero Index (Mode 6)

U - Inject SEU at Index (Mode 7)

E - Simulate Random SEU

H - Help

>S

SEU_DETECT = 0 (0)

BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000000000

>S
SEU_DETECT = 0 (0)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000000000

>A
ADDR = 000300080
ADDR = 000300080

>D
Ok
ADDR = 000300080

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)

ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000300080

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000300080

>A
ADDR = 000300081
ADDR = 000300081

>D
Ok
ADDR = 000300081

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)

Mode = 0
Index = 000000000
ADDR = 000300081

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000300081

>A
ADDR = 000300082
ADDR = 000300082

>D
Ok
ADDR = 000300082

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0

Index = 000000000
ADDR = 000300082

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000300082

>A
ADDR = 000300083
ADDR = 000300083

>D
Failed
ADDR = 000300083

>D
Ok
ADDR = 000300083

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (1)
Mode = 0
Index = 000000000
ADDR = 000300083

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000300083

>A
ADDR = 000300084
ADDR = 000300084

>D
Ok
ADDR = 000300084

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000300084

>S
SEU_DETECT = 1 (1)
BUSY = 0 (0)
ERROR = 0 (0)
Mode = 0
Index = 000000000
ADDR = 000300084

>M
Mode = 1
Ok

>S
SEU_DETECT = 1 (1)
BUSY = 0 (1)
ERROR = 1 (1)
Mode = 1
Index = 000000000
ADDR = 000300084

>W

?

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 1 (1)

Mode = 1

Index = 000000000

ADDR = 000300084

>

SEU Monitor v1.08b

S - Macro Status

C - CRC Scan Status

R - Reset Macro

M - Set Mode

P - Pulse ERROR_INJECT

A - Set ADDR

D - Inject SEU at ADDR (Mode 4)

I - Increment Index (Mode 5)

Z - Zero Index (Mode 6)

U - Inject SEU at Index (Mode 7)

E - Simulate Random SEU

H - Help

>S

SEU_DETECT = 0 (0)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

>S

SEU_DETECT = 0 (0)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

>A

ADDR = 0003000FA

ADDR = 0003000FA

>D

Ok

ADDR = 0003000FA

>S

SEU_DETECT = 1 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 0003000FA

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 0003000FA

>A

ADDR = 003000FB

ADDR = 0003000FB

ADDR = 0003000FB

```
>D
Ok
ADDR = 0003000FB

>S
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 0003000FB

>S
SEU_DETECT = 1 (1)
    BUSY = 0 (0)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 0003000FB

>M
Mode = 1
Ok
```

```
>S
SEU_DETECT = 1 (1)
    BUSY = 0 (1)
   ERROR = 1 (1)
    Mode = 1
   Index = 000000000
    ADDR = 0003000FB

>S
SEU_DETECT = 1 (1)
    BUSY = 0 (0)
   ERROR = 1 (1)
    Mode = 1
   Index = 000000000
    ADDR = 0003000FB

>
SEU Monitor v1.08b

S - Macro Status
C - CRC Scan Status
R - Reset Macro
M - Set Mode
P - Pulse ERROR_INJECT
```

```
A - Set ADDR

D - Inject SEU at ADDR (Mode 4)
I - Increment Index (Mode 5)
Z - Zero Index (Mode 6)
U - Inject SEU at Index (Mode 7)
E - Simulate Random SEU

H - Help

>S
SEU_DETECT = 0 (0)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000000000

>A
ADDR = 0003000DA
ADDR = 0003000DA

>D
Ok
ADDR = 0003000DA
```

	ADDR = 0003000DB	Mode = 0
	ADDR = 0003000DB	Index = 000000000
>A		ADDR = 0003000DB
ADDR = 00030000D		
ADDR = 00030000D	>D	
	Ok	>S
	ADDR = 0003000DB	SEU_DETECT = 1 (1)
>F		BUSY = 0 (0)
?		ERROR = 0 (0)
	>S	Mode = 0
>A	SEU_DETECT = 1 (1)	Index = 000000000
ADDR = 0000	BUSY = 0 (1)	ADDR = 0003000DB
	ERROR = 0 (0)	

Συμπέρασμα: Καμία διαταραχή στο κύκλωμα.

5.3.4. Δοκιμή Δ: Εισαγωγή τυχαίων σφαλμάτων

Στόχος των δοκιμών Δ είναι η χρήση της εντολής τυχαίων σφαλμάτων

➢ Δοκιμή Δ1: Εισαγωγή τυχαίων σφαλμάτων και διόρθωση

Ο ελεγκτής τέθηκε σε mode = 1 και στο κύκλωμα εισήχθησαν τυχαία λάθη μέσω της εντολής Ε. Σε γενικές γραμμές τα λάθη εισήχθησαν κανονικά και διορθώθηκαν κατά τα προβλεπόμενα με ελάχιστες εξαιρέσεις. Το πρόγραμμα εκτελέστηκε μέσω του PicoBlaze για αρκετή ώρα και πάλι δεν εντοπίστηκαν διαταραχές ορατές από τον χρήστη.

Στο περιβάλλον του HyperTerminal καταγράφηκαν τα εξής:

>M	Ok	BUSY = 0 (1)
Mode = 1	Random ADDR = 000000002	ERROR = 0 (0)
Ok		Mode = 1
		Index = 000000000
	>S	ADDR = 000000000
>E	SEU_DETECT = 0 (0)	

>E

Ok

Random ADDR = 000000009

>S

SEU_DETECT = 0 (0)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000000000

>E

Ok

Random ADDR = 000000025

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (1) <---- υποδηλώνει ότι υπήρξε κάποια αποτυχία στην διόρθωση και έγινε και δεύτερη προσπάθεια

Mode = 1

Index = 000000000

ADDR = 000000000

>E

Ok

Random ADDR = 000000094

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000000000

>E

Ok

Random ADDR = 000000251

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000000000

>E

Ok

Random ADDR = 000000947

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000000000

>E

Ok

Random ADDR = 00000251F

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 1

Index = 000000000

ADDR = 000000000

>E	Mode = 1	
Ok	Index = 000000000	>S
Random ADDR = 00000947D	ADDR = 000000000	SEU_DETECT = 0 (1)
		BUSY = 0 (1)
		ERROR = 0 (0)
>S	>E	Mode = 1
SEU_DETECT = 0 (1)	Ok	Index = 000000000
BUSY = 0 (1)	Random ADDR = 0000251F6	ADDR = 000000000
ERROR = 0 (0)		

Η καταγραφή συνεχίζεται για περίπου ακόμα 50 σελίδες οι οποίες δεν παρατίθενται χάριν συντομίας.

- **Δοκιμή Δ2: Εισαγωγή τυχαίων σφαλμάτων χωρίς διόρθωση**

Το πρόγραμμα θα εισήγαγε διαδοχικά τυχαία λάθη (E) με την βοήθεια του PicoBlaze. Προφανώς οι χρόνοι εκτέλεσης της εντολής, δεν επέτρεψαν την επανάληψη. Τα τρία αρχικά >E του hyperterminal αρχείο καταγραφής (log file) υποδηλώνουν τρεις ξεχωριστές διαμορφώσεις (configuration) του κυκλώματος. Ακολούθως, το πρόγραμμα υπέπεσε σε λειτουργικό σφάλμα και επαναλάμβανε την εντολή status *χωρίς* να του έχει δοθεί. Στην επόμενη διαμόρφωση πάλι δεν μπόρεσε να ανταποκριθεί σωστά.

>E

>E

>E

Ok

Random ADDR = 000000001

>S

SEU_DETECT = 0 (0)

BUSY = 1 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

>S

SEU_DETECT = 0 (0)

BUSY = 1 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

>S

SEU_DETECT = 0 (0)

BUSY = 1 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 000000000

Συμπέρασμα: Ο χρόνος μετάβασης του δείκτη ADDR στην ψευδοτυχαία τιμή ήταν μεγαλύτερος από την ταχύτητα που ο PicoBlaze απέδιδε τις εντολές με αποτέλεσμα την διαταραχή του ελεγκτή.

➢ Δοκιμή Δ3: Εισαγωγή τυχαίου σφάλματος με χρονική καθυστέρηση

Για να προλάβει ο ελεγκτής να εκτελέσει την εντολή E, ειδικά αν η τιμή της ADDR που θα του δοθεί είναι μεγάλη, χρειάζεται κάποιο χρονικό διάστημα. Έτσι, συντάχθηκε κώδικας για τον PicoBlaze ώστε μετά την εντολή E να ακολουθεί μία εντολή Status. Στα δύο πρώτα configurations η εφαρμογή δεν ανταποκρίθηκε. Την τρίτη φορά το πρόγραμμα εκτελέστηκε κατά τα αναμενόμενα.

```
>E

>E

>E
Ok
Random ADDR = 000000001

>S
SEU_DETECT = 0 (0)
    BUSY = 1 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000000000

>E
Ok
Random ADDR = 000000004
```

```
>S
SEU_DETECT = 0 (0)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000000000

>E
Ok
Random ADDR = 000000012

>S
SEU_DETECT = 0 (0)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000000000

>E
Ok
```

```
Random ADDR = 00000004A

>S
SEU_DETECT = 0 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000000000

>E
Ok
Random ADDR = 000000128

>S
SEU_DETECT = 0 (1)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000000000
```

>E	>E	>E
Ok	Ok	Ok
Random ADDR = 0000004A3	Random ADDR = 00000128F	Random ADDR = 000004A3E
>S	>S	>S
SEU_DETECT = 0 (1)	SEU_DETECT = 0 (1)	SEU_DETECT = 0 (1)
BUSY = 0 (1)	BUSY = 0 (1)	BUSY = 0 (1)
ERROR = 0 (0)	ERROR = 0 (0)	ERROR = 0 (0)
Mode = 0	Mode = 0	Mode = 0
Index = 000000000	Index = 000000000	
ADDR = 000000000	ADDR = 000000000	*Ακολουθούν πλήθος παρόμοιων σελίδων, οι οποίες δεν παρατίθενται, χάριν συντομίας.*

5.3.5.Τυχαία δοκιμή

Στην προσπάθεια μας να δημιουργήσουμε ορθά και εκτελέσιμα προγράμματα εντολών, με την βοήθεια του PicoBlaze, προέκυψε μια αλληλουχία τυχαία τοποθετημένων εντολών, οι οποίες όμως είχαν ως αποτέλεσμα την απόκρυψη της δεύτερης γραμμής της LCD οθόνης. Όσες φόρες επαναλάβαμε τον τυχαίο κώδικα με την βοήθεια του PicoBlaze το αποτέλεσμα παρέμενε το ίδιο. Στην χειρωνακτική εφαρμογή των ίδιων εντολών, με την ίδια αλληλουχία, δεν παρατηρήθηκε αλλοίωση στην απεικόνιση της οθόνης. Συμπεραίνουμε λοιπόν, ότι οι χρόνοι εκτέλεσης είναι αυτοί που προκάλεσαν αυτή την διαταραχή. Λόγω, της μέχρι τώρα αδυναμίας να προκαλέσουμε εμφανής αλλοιώσεις στη εφαρμογή, παραθέτουμε τον τυχαίο κώδικα.

```
>C

CRC Scan Active (184 CRC Scans per second)

Device Frame Count = 8663


>S

SEU_DETECT = 0 (0)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000000
    ADDR = 000000000


>P

Failed

Mode?


>M

Mode = 2

Ok
```

```
>M

Mode = 0

Ok


>E

Ok

Random ADDR = 000000025


>A

ADDR = 00000028f

ADDR = 00000028F


>D

Ok

ADDR = 00000028F


>I

Ok
```

```
Index = 000000001


>I

Ok

Index = 000000002


>S

SEU_DETECT = 0 (0)
    BUSY = 0 (1)
   ERROR = 0 (0)
    Mode = 0
   Index = 000000002
    ADDR = 00000028F


>Z

Ok

Index = 000000000


>D
```

Ok

ADDR = 00000028F

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000000

ADDR = 00000028F

>I

Ok

Index = 000000001

>U

Ok

Index = 000000001

>S

SEU_DETECT = 0 (1)

BUSY = 0 (1)

ERROR = 0 (0)

Mode = 0

Index = 000000001

ADDR = 00000028F

>S

SEU_DETECT = 1 (1)

BUSY = 0 (0)

ERROR = 0 (0)

Mode = 0

Index = 000000001

ADDR = 00000028F

>C

CRC Scan Active (141 CRC Scans per second)

Device Frame Count = 8663

>C

CRC Scan Active (141 CRC Scans per second)

Device Frame Count = 8663

>C

CRC Scan Active (140 CRC Scans per second)

Device Frame Count = 8663

>C

CRC Scan Active (141 CRC Scans per second)

Device Frame Count = 8663

6. Συμπεράσματα

Τα SEUs αποτελούν φαινόμενο κατά το οποίο διαταράσσεται η λειτουργία των συσκευών ημιαγωγών. Λαμβάνοντας υπόψη την κατασκευαστική δομή των ηλεκτρονικών κυκλωμάτων και την σχεδόν ολοκληρωτική τους εξάρτηση από τα ολοκληρωμένα κυκλώματα, συμπεραίνουμε ότι τα SEUs αποτελούν πρόβλημα για την αξιοπιστία των τελικών εφαρμογών.

Η αποφυγή του φαινομένου είναι πρακτικώς αδύνατη και γι' αυτό οι ερευνητές αναζητούν μεθόδους με τις οποίες θα εξασφαλίζεται η ομαλή λειτουργίας των συστημάτων ακόμα και όταν βρίσκονται υπό την επήρεια SEUs. Οι μέθοδοι άμβλυνσης των επιπτώσεων των διαταραχών, ποικίλλουν και εφαρμόζονται κατά περίπτωση συσκευής και σχεδιαστικής εφαρμογής. Ειδική αναφορά γίνεται για τα SRAM-based FPGAs λόγω της αυξημένης ζήτησής τους στην αγορά και λόγω της υιοθέτησης τους από το πειραματικό μέρος της διατριβής. Αναλύθηκαν, όλες εκείνες οι μέθοδοι που μπορούν να προστατέψουν επιτυχώς τα FPGAs, τόσο στο κομμάτι της συνδυαστικής λογικής όσο και στο ακολουθιακό.

Στο δεύτερο μέρος η ανάλυση στοχοθετήθηκε στα FPGAs της XILINX και σε πειράματα που έχει διενεργήσει η εταιρεία, με τα οποία αποδεικνύει την σπανιότητα του φαινομένου αλλά και τις αμελητέες επιπτώσεις που έχει στις τελικές εφαρμογές, λόγω της κατασκευαστικής δομής των συσκευών. Ακολούθως, αναπτύχθηκαν οι στρατηγικές αντιμετώπισης που προτείνει η εταιρεία οι οποίες διαφοροποιούνται ανάλογα με την κρισιμότητα του κυκλώματος και τον χρόνο αντίδρασης. Εκτός λοιπόν από τη υιοθέτηση τεχνικών άμβλυνσης κατά περίπτωση, η XILINX έχει ενσωματώσει κύκλωμα σάρωσης στα VIRTEX – 5 FPGAs με το οποίο παρακολουθείται η διαμόρφωση της συσκευής και ενημερώνει τον χρήστη αν προκύψει αλλαγή στην μνήμη. Μετά τον εντοπισμό του σφάλματος προτείνει στα κρίσιμα κυκλώματα την άμεση επαναδιαμόρφωση της περιοχής, στην οποία εντοπίστηκε το σφάλμα και σε τακτά χρονικά διαστήματα (ανεξαρτήτου ύπαρξης SEU) την επαναδιαμόρφωση ολόκληρης της συσκευής. Τέλος, αν η εφαρμογή διαθέτει κυκλώματα άμβλυνσης, τότε απλώς μπορεί να αναμένει την χρονική στιγμή που δεν θα είναι απαραίτητη στο τελικό σύστημα για να επαναδιαμορφωθεί.

Επιπρόσθετα, η XILINX έχει αναπτύξει ένα εργαλείο δοκιμών, τον SEU Controller macro,με το οποίο εισάγει ανιχνεύει και διορθώνει σφάλματα (αντιστροφές ψηφίων σε πλαίσια της μνήμης) στην εφαρμογή του χρήση. Ο στόχος του ελεγκτή είναι να εξετάσει μια εφαρμογή, ως προς τα κρίσιμα σημεία της, στα οποία κατόπιν θα χρειαστεί να ληφθούν επιπλέον μέτρα προστασίας, ώστε να μην θέσει σε κίνδυνο το σύστημα στο οποίο θα μετέχει.

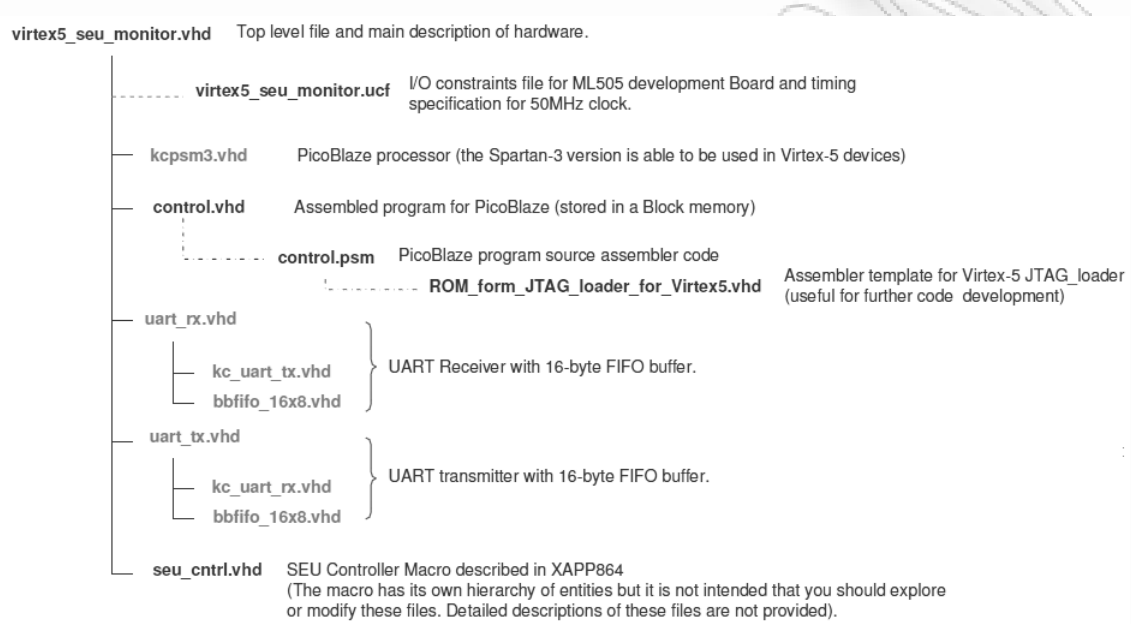
Σε αυτή την κατεύθυνση και για να διαπιστωθεί η αξιοπιστία του ελεγκτή δημιουργήθηκε εφαρμογή με τον ενσωματωμένο μικροεπεξεργαστή PicoBlaze, τον SEU Controller και κύκλωμα οδήγησης της LCD οθόνης, του FPGA VIRTEX-5 ML505 το οποίο υπέστη σειρά δοκιμών. Ουσιαστικά, ο SEU Controller ελεγχόταν από τον PicoBlaze και το κύκλωμα οδήγησης της LCD οθόνης από έναν έτερο PicoBlaze. Πολλαπλά σφάλματα εισήχθησαν στην εφαρμογή και ο ελεγκτής απεδείχθη ικανός τόσο στην ανίχνευση όσο και στην διόρθωση αυτών, όταν η εισαγωγή εκτελούνταν από τον χρήστη. Όταν προγραμματιζόταν ο PicoBlaze, ώστε να στέλνει εντολές εισαγωγής και διόρθωσης λαθών στον SEU Controller, τότε **σε ορισμένες μόνο περιπτώσεις**, η απόκριση του τελευταίου διαταρασσόταν λόγω της αυξημένης ταχύτητας εισαγωγής των εντολών. Ο ελεγκτής απαιτεί χρόνο για να διατρέξει ολόκληρη την συσκευή και να εντοπίσει το πλαίσιο που θα εισαχθεί το επιθυμητό σφάλμα. Ο μικροεπεξεργαστής αποδείχθηκε ταχύτερος και παρακωλούσε την ομαλή λειτουργία του ελεγκτή. Τέλος, παρόλη την προσπάθεια κατάρρευσης του κυκλώματος, ακόμα και με την εισαγωγή πολλαπλών σφαλμάτων σε διαφορετικά πλαίσια διαμόρφωσης της εφαρμογής, ο στόχος δεν επετεύχθη και το κύκλωμα παρέμενε σε κανονική λειτουργία καθ' όλη την διάρκεια των δοκιμών. Σύμφωνα, με τα εργαλεία που μας παρέχει η XILINX, αν και θεωρητικά είχαμε τις σωστές διευθύνσεις πλαισίων που απασχολούσε η εφαρμογή μας, δεν υπήρχε δυνατότητα να απομονώσουμε το ψηφίο που αφορούσε τον σχεδιασμό μας (από τα 1.312 ψηφία) και να το αντιστρέψουμε. Εν γένει διαπιστώθηκε ότι υπάρχει έλλειψη τεκμηρίωσης από την

XILINX όσον αφορά τις διευθύνσεις των πλαισίων και για αυτό δεν είμαστε βέβαιοι για την επιτυχία της εισαγωγής των σφαλμάτων μας.

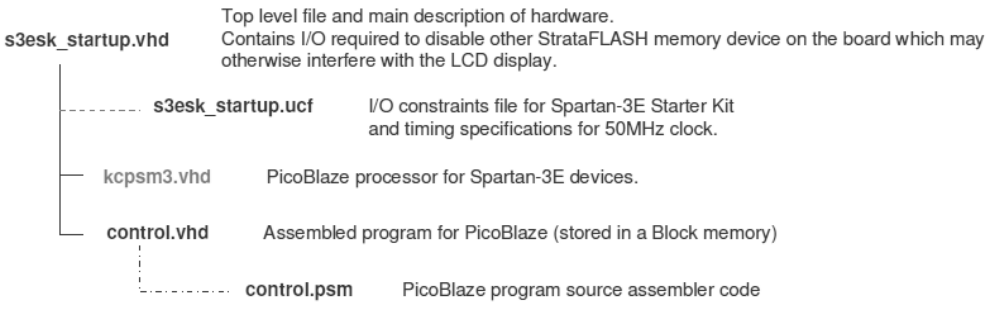
Εν κατακλείδι, η ενασχόληση με τα φαινόμενα SEUs και η λήψη μέτρων προστασίας που συνεπάγονται αύξηση του τελικού κυκλώματος και άρα περαιτέρω κόστος και ανάγκη τροφοδοσίας, θεωρούνται αναγκαίες μόνο σε εφαρμογές με απαιτήσεις υψηλού βαθμού αξιοπιστίας. Διαφορετικά, η απλή επαναδιαμόρφωση των συσκευών σε τακτά χρονικά διαστήματα θεωρείται ικανή για να εξασφαλίσει την ομαλή λειτουργία των κυκλωμάτων σε όλο το χρόνο ζωής της συσκευής.

ΠΑΡΑΡΤΗΜΑ Α: Αρχεία Σχεδιασμού.

Η εφαρμογή χωρίζεται σε δύο βασικά μέρη. Το πρώτο μέρος αποτελείται από τον PicoBlaze και τον SEU Controller, ενώ το δεύτερο από ένα άλλο PicoBlaze και τα αρχεία ελέγχου της οθόνης και παραγωγής μηνυμάτων. Στην εφαρμογή λοιπόν συνδυάστηκαν δύο ξεχωριστά projects της XILINX το **VIRTEX – 5 SEU Monitor**[45] και το **Initial Design for Spartan-3E Starter Kit (LCD Display Control)**[46].



Εικόνα Α1: Σχεδιαστικά αρχεία του VIRTEX-5 SEU monitor



Εικόνα Α2: Σχεδιαστικά αρχεία του LCD Monitor

Στο αρχείο υψηλότερου επιπέδου της παρούσας εφαρμογής, συνδυάστηκαν τα αρχεία υψηλότερου επιπέδου των ανωτέρω. Λόγω της χρήσης δύο διαφορετικών μικροεπεξεργαστών PicoBlaze το module εισάγεται δύο φορές, ως εξάρτημα με διαφορετική ονομασία των θυρών του (*http://www.xilinx.com/products/ipcenter/picoblaze-S3-V2-Pro.htm)*. Έτσι, έχουμε :

- Το εξάρτημα

```
component kcpsm3

    Port (      address : out std_logic_vector(9 downto 0);
            instruction : in std_logic_vector(17 downto 0);
                port_id : out std_logic_vector(7 downto 0);
```

```
        write_strobe : out std_logic;
            out_port : out std_logic_vector(7 downto 0);
         read_strobe : out std_logic;
             in_port : in std_logic_vector(7 downto 0);
           interrupt : in std_logic;
       interrupt_ack : out std_logic;
               reset : in std_logic;
                 clk : in std_logic);
  end component;
```

- Για τον SEU Controller

```
    seu_processor: kcpsm3
  port map(address => seu_address,
           instruction => seu_instruction,
               port_id => seu_port_id,
          write_strobe => seu_write_strobe,
              out_port => seu_out_port,
           read_strobe => seu_read_strobe,
               in_port => seu_in_port,
             interrupt => seu_interrupt,
         interrupt_ack => seu_interrupt_ack,
                 reset => seu_kcpsm3_reset,
                   clk => clk_div);

 seu_program_rom: control
  port map(      address => seu_address,
             instruction => seu_instruction,
                     clk => clk_div);

 seu_kcpsm3_reset <= '0';
```

- Για το LCD Monitor

```
lcd_processor: kcpsm3
  port map(address => lcd_address,
           instruction => lcd_instruction,
               port_id => lcd_port_id,
          write_strobe => lcd_write_strobe,
              out_port => lcd_out_port,
           read_strobe => lcd_read_strobe,
               in_port => lcd_in_port,
             interrupt => lcd_interrupt,
         interrupt_ack => lcd_interrupt_ack,
                 reset => lcd_kcpsm3_reset,
                   clk => clk_div);

 lcd_program_rom: lcd_ctrl
  port map(address => lcd_address,
           instruction => lcd_instruction,
                   clk => clk_div);
```

Η εφαρμογή για το LCD Monitor έχει ως στόχο συσκευή Spartan – 3 και χρονίζεται στα 50 MHz. Στην συσκευή ML505 το user clk είναι στα 100 MHz και γι' αυτό το λόγο χρησιμοποιείται κύκλωμα DCM για διαίρεση της συχνότητας (clk_div). Το αρχείο έχει ως εξής:

```
entity DCM_LCD is
port (  CLKIN_IN            : in    std_logic;
        RST_IN              : in    std_logic;
        CLKFX_OUT           : out   std_logic;
        CLKIN_IBUFG_OUT : out   std_logic;
        CLK0_OUT            : out   std_logic;
        LOCKED_OUT          : out   std_logic);
end DCM_LCD;

architecture Behavioral of DCM_LCD is
  signal CLKFB_IN       : std_logic;
  signal CLKFX_BUF      : std_logic;
  signal CLKIN_IBUFG    : std_logic;
  signal CLK0_BUF       : std_logic;
  signal GND_ΨΗΦΊΟ          : std_logic;

begin

  GND_ΨΗΦΊΟ             <= '0';
  CLKIN_IBUFG_OUT <= CLKIN_IBUFG;
  CLK0_OUT            <= CLKFB_IN;
  CLKFX_BUFG_INST : BUFG
    port map (I=>CLKFX_BUF,
              O=>CLKFX_OUT);

  CLKIN_IBUFG_INST : IBUFG
    port map (I=>CLKIN_IN,
              O=>CLKIN_IBUFG);

  CLK0_BUFG_INST : BUFG
    port map (I=>CLK0_BUF,
              O=>CLKFB_IN);

  DCM_INST : DCM
  generic map
       ( CLK_FEEDBACK            => "1X",
         CLKDV_DIVIDE            => 2.0,
         CLKFX_DIVIDE            => 4,
         CLKFX_MULTIPLY          => 2,
         CLKIN_DIVIDE_BY_2       => FALSE,
         CLKIN_PERIOD            => 20.000,
         CLKOUT_PHASE_SHIFT      => "NONE",
         DESKEW_ADJUST           => "SYSTEM_SYNCHRONOUS",
         DFS_FREQUENCY_MODE      => "LOW",
         DLL_FREQUENCY_MODE      => "LOW",
         DUTY_CYCLE_CORRECTION => TRUE,
         FACTORY_JF              => x"8080",
         PHASE_SHIFT             => 0,
         STARTUP_WAIT            => FALSE)
    port map
```

```
                (CLKFB    =>CLKFB_IN,
                 CLKIN    =>CLKIN_IBUFG,
                 DSSEN    =>GND_ΨΗΦΊΟ,
                 PSCLK    =>GND_ΨΗΦΊΟ,
                 PSEN     =>GND_ΨΗΦΊΟ,
                 PSINCDEC=>GND_ΨΗΦΊΟ,
                 RST      =>RST_IN,
                 CLKDV    =>open,
                 CLKFX    =>CLKFX_BUF,
                 CLKFX180 =>open,
                 CLK0     =>CLK0_BUF,
                 CLK2X    =>open,
                 CLK2X180 =>open,
                 CLK90    =>open,
                 CLK180   =>open,
                 CLK270   =>open,
                 LOCKED   =>LOCKED_OUT,
                 PSDONE   =>open,
                 STATUS   =>open);
    end Behavioral;
```

Στα αρχεία του SEU Controller Monitor δεν προέκυψαν άλλες αλλαγές, εκτός από την πρόσθεση εντολών στην ROM του PicoBlaze, κατά την εκτέλεση των πειραμάτων στα οποία θα αναφερθούμε ακολούθως. Στα αρχεία του LCD Monitor αφαιρέθηκαν τα κομμάτια του κώδικα που αναφέρονται στο κουμπί πίεσης και περιστροφής (rotary signals) και άλλαξε το μήνυμα απεικόνισης στην οθόνη.

ΠΑΡΑΡΤΗΜΑ Β: Κώδικας δοκιμών

Ο κώδικα που γράφτηκε για να οδηγήσει ο PicoBlaze σειρά εντολών στον SEU Controller, τοποθετήθηκε στο control.psm αρχείο του project Virtex5_seu_monitor. Η δημιουργία αρχείων για τον μικροεπεξεργαστή μπορεί να τελεστεί με την χρήση δύο διαφορετικών λογισμικών. Το ένα είναι το KCPSM3.EXE, που παρέχεται μαζί με τα αρχεία του PicoBlaze και πρόκειται για ένα assembler που τρέχει σε περιβάλλον DOS. Το άλλο είναι το Mediatronix pBlazIDE που είναι γραφικό περιβάλλον ανάπτυξης και περιλαμβάνει τόσο τον assembler όσο simulator με πλήρες σετ εντολών.

Στον παρόν χρησιμοποιήθηκε ο KCPSM3.EXE, που συνοδεύεται από τα αρχεία ROM_form.vhd, ROM_form.v και ROM_form.coe. Όλα τα αρχεία τοποθετούνται στον ίδιο φάκελο μαζί με το αρχείο εντολών που περιέχει τον κώδικα που θα υλοποιήσει ο μικροεπεξεργαστής. Το αρχείο γράφεται σε απλή μορφή κειμένου και σώζεται ως filename.psm. Για να μεταφραστεί από τον assembler, ανοίγουμε ένα DOS παράθυρο και οδηγούμε στο φάκελο του assembler. Γράφουμε:

```
kcpsm3 <filename>[.psm]
```

Αν το αρχείο είναι σωστά συνταγμένο, τότε μεταφράζεται σε ένα αρχείο με κατάληξη .vhd και είναι έτοιμο για χρήση στο βασικό project. Από το control.psm αφαιρέθηκαν τα κομμάτια του κώδικα που σχετίζονταν με την εντολή HELP για να εξοικονομηθούν εντολές, ώστε να συνταχθούν τα προγράμματα (περιορισμός στις 1.024 εντολές). Μετά τον LFSR το πρόγραμμα εξετάζει αν κάποιο γράμμα εντολής έχει δοθεί στο HyperTerminal. Συνδέουμε τους καταχωρητές σε μόνιμες τιμές ώστε να θεωρεί ότι έχει λάβει εντολή.

```
;INPUT s0, UART_status_port          ;test Rx_FIFO buffer

LOAD s0,01
```

```
        ;TEST s0, rx_data_present

        TEST s0,01

        JUMP Z, random_wait_loop
```

Για να εκτελεστούν x εντολές δημιουργούμε έναν δείκτη (instr_number) με αρχική τιμή την ‘0’, ο οποίος αυξάνει μόλις εκτελεστεί μία εντολή και έτσι οδηγεί στην εκτέλεση της επόμενης εντολής, μέχρι την x, όπου και το πρόγραμμα μπαίνει σε κατάσταση αναμονής.

```
                FETCH s9,instr_number
                COMPARE s9,00

                JUMP Z,instr_s

                COMPARE s9,01

                JUMP Z,instr_a

                COMPARE s9,02

                JUMP Z,instr_d

                COMPARE s9,03

                JUMP Z,instr_s

                COMPARE s9,04
                JUMP Z,restart

                ;

restart:        LOAD s9,00

                STORE s9,instr_number
```

Όλες οι εντολές που μπορεί να εκτελέσει ο SEU Controller καταγράφονται στη συνέχεια στον κώδικα και ο μετρητής προγράμματος μέσω της εντολή JUMP, instr_x καθοδηγεί ποια θα εκτελεστεί. Επειδή, στην πραγματικότητα όταν γράφουμε στο περιβάλλον του HyperTerminal ο χαρακτήρας μεταφράζεται σε μια ASCII κωδικοποιημένη τιμή, οι εντολές στέλνουν ένα αριθμό στον SEU. Οι εντολές αντιστοιχούν στους δεκαεξαδικούς αριθμούς (βάσει ASCII) :

Εντολή	ASCII Κωδικοποίηση
‘S’	53
‘C’	43
‘R’	52
‘M’	4D
‘P’	50
‘A’	41

'D'	44
'I'	49
'Z'	5A
'U'	55
'E'	45

Οι εντολές συντάσσονται ως εξής:

```
instr_s:
        LOAD s1,53                  ;status command (S)
        CALL send_to_UART           ;echo received character
        CALL upper_case             ;convert to upper case
        ADD s9,01
        STORE s9,instr_number
        JUMP alphabetical
        ;
instr_a:
        LOAD s1,41                  ;Set ADDR command (A)
        CALL send_to_UART           ;echo received character
        CALL upper_case             ;convert to upper case
        ADD s9,01
        STORE s9,instr_number
        JUMP alphabetical
        ;
```

Αν η εντολή απαιτεί και δεύτερο όρισμα, όπως για παράδειγμα μια εντολή mode θέλει έναν αριθμό και μία ADDR 9 ψηφίο διεύθυνσης τότε η υπορουτίνα read_from_UART, ακολουθώντας την ίδια λογική με τα προαναφερόμενα γίνεται ως εξής:

```
read_from_UART:
        LOAD s0,01
        ;INPUT s0, UART_status_port     ;test Rx_FIFO buffer        TEST s0,01
                                        ;                           ;TEST s0,
rx_data_present
    JUMP Z, read_from_UART
```

```
        FETCH s9, instr_plus (δείκτης)
;---------------------- address ----------------------------
        COMPARE s9,00
        JUMP Z, read_character0
        COMPARE s9,01
        JUMP Z, read_character0
        COMPARE s9,02
        JUMP Z, read_character0
        COMPARE s9,03
        JUMP Z, read_character1
        COMPARE s9,04
        JUMP Z, read_character0
        COMPARE s9,05
        JUMP Z, read_character0
        COMPARE s9,06
        JUMP Z, read_character5
        COMPARE s9,07
        JUMP Z, read_charactera
        COMPARE s9,08
        JUMP Z, read_character2
;--------------mode=1 --------------------------
        COMPARE s9,09
        JUMP Z, read_character1

read_character0:
        LOAD s1,30                              ;decimal number '0'
        CALL send_to_UART                       ;echo received character
        ADD s9,01
        STORE s9,instr_plus
    RETURN
```

```
        ;
read_character1:
        LOAD s1,31                                  ;decimal number '1'
        CALL send_to_UART           ;echo received character
    ADD s9,01
        STORE s9,instr_plus
        RETURN
        ;
```

Κ.ο.κ

Το αρχείο δεν έχει υποστεί άλλες αλλαγές.

Σημείωση επί του λογισμικού: Οι διευθύνσεις των frames που χρησιμοποιήθηκαν στο πειραματικό μέρος δόθηκαν από τον ISE Project Navigator. Κατά την διαδικασία Generate Programming File ορίζουμε στις ιδιότητες Readback την δημιουργία Logic Allocation File. Το λογισμικό παράγει ένα αρχείο .dll το οποίο ανοίγει με απλό σημειωματάριο και περιέχει όλες τις πληροφορίες για τα απασχολούμενα slices, από πιο κομμάτι της εφαρμογής και σε πιο frame address αντιστοιχούν.

Βιβλιογραφία

1. SIA SEMICONDUCTOR INDUSTRY ASSOCIATION. The National Technology Roadmap for Semiconductors. USA, 1994.

2. BARTH, J. Applying Computer Simulation Tools to Radiation Effects Problems. In: IEEE NUCLEAR SPACE RADIATION EFFECTS CONFERENCE, NSREC, 1997. Proceedings… IEEE Computer Society, 1997. p. 1-83.

3. DAC, WHEN or Will FPGAs kill ASICs? Panel presented at ACM Design Automation Conference,2001

4. ALTERA INC. Data Book. USA, 2001. Available at: <www.altera.com>. Visited on November, 2001.

5. XILINX INC. Virtex® Series Configuration Architecture User Guide: Application Notes 151. USA, 2000c.

6. J. F. ZIEGLER, "IBM Experience in Soft Fails in Computer Electronics (1978-1994)," *IBM Jour. Res. and Dev.*, vol. 40, no. 1, pp. 3–18, 1996.

7. J. T. WALLMARK and S. M. MARCUS, "Minimum Size and Maximum Packing Density of Non-Redundant Semiconductor Devices," *Proc. IRE*, vol. 50, pp. 286–298, Mar. 1962.

8. G. C. MESSENGER and M. ASH, *Single Event Phenomena*. Chapman & Hall, 1997.

9. J. VON NEUMANN, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components (1959)," in A. H. Taub, editor, *John von Neumann: Collected Works, Volume V: Design of Computers, Theory of Automata and Numerical Analysis*, Oxford University Press, 1963, pp. 329–378.

10. T. C. MAY and M. H. WOODS, "A New Physical Mechanism for Soft Errors in Dynamic Memories," in *Proc. 16th Annua Reliability Physics Symp.*, 1978, pp. 33–40.

11. CANARIS, J.; WHITAKER, S. Circuit techniques for the radiation environment of space. In: CUSTOM INTEGRATED CIRCUITS CONFERENCE, 1995. Proceedings... IEEE Computer Society, 1995. p. 77-80.

12. J. F. ZIEGLER and W. A. LANFORD, "Effect of Cosmic Rays on Computer Memories," *Science*, vol. 206, no. 4420, pp. 776–788, Nov. 1979.

13. FAN WANG and VISHWANI D. AGRAWAL, Single Event Upset: An Embedded Tutorial, 21st International Conference on VLSI Design, 2008 IEEE

14. K. ROY, S. KUNDU, R. GALIVANCHE, V. NARAYANAN, R. RAINA, and P. N. SANDA, "Is the Concern for Soft-Error Overblown?," in *Proc. International Test Conf. (Panel Discussion)*, 2005.

15. A. J. VAN DE GOOR, *Testing Semiconductor Memories: Theory and Practice*. Wiley, 1991.

16. R. Baumann, “Soft Errors in Advanced Computer Systems,”IEEE Design & Test of Computers, vol. 22, no. 3, pp. 258–266,2005..

17. C. L. CLAEYS and E. SIMOEN, *Radiation Effects in Advanced Semiconductor Materials and Devices*. Springer, 2002.

18. M. BELLATO, P. BERNARDI, D. BORTOLATO, A. CANDELORI, M. CESCHIA, A. PACCAGNELLA, M. REBAUDENGO, M. REORDA, M. VIOLANTE, and P. ZAM-BOLIN.Evaluating the effects of SEUs affecting the configuration memory of an SRAM-based FPGA. Proceedings of Design, Automation and Test in Europe Conference and Exhibition, 1:584{589 Vol.1, Feb. 2004.

19. K. A. LaBel. SECCA - single event effect criticality analysis, NASA/GSFC, 1996.http://radhome.gsfc.nasa.gov/radhome/papers/seecai.htm

20. GODDARD Space Flight Center Radiation Effects Facility.

http://radhome.gsfc.nasa.gov/radhome/ref/GSFC_REF.html

21. L. KAFKA and O. NOVAK. FPGA-based fault simulator. Proceeding of Design and Diagnostics of Electronic Circuits and systems, 2006 IEEE, pages 272{276, 2006.

22. KATZ, R. et al. An SEU-Hard flip-Flop for Antifuse FPGAs. In: INTERNATIONAL CONFERENCE ON MILITARY AND AEROSPACE APPLICATIONS OF PROGRAMMABLE LOGIC DEVICES, MAPLD, 2001. Proceedings..., 2001.

23. REED, R. A. et al. Heavy ion and proton-induced single event multiple upset. IEEE Transactions on Nuclear Science, New York, v.44, n.6, p. 2224-2229, Dec. 1997.

24. WEAVER, H.; et al. An SEU Tolerant Memory Cell Derived from Fundamental Studies of SEU Mechanisms in SRAM. IEEE Transactions on Nuclear Science, New York, v.34, n.6, Dec. 1987.

25. ROCKETT, L. R. An SEU-hardened CMOS data latch design. IEEE Transactions on Nuclear Science, New York, v.35, n.6, p. 1682-1687, Dec. 1988.

26. BESSOT, D.; VELAZCO, R. Design of SEU-hardened CMOS memory cells: the HIT Cell. In: EUROPEAN CONFERENCE ON RADIATION AND ITS EFFECTS ON COMPONENTS AND SYSTEMS, RADECS, 2., 1993. Proceedings… IEEE Computer Society, 1993. p. 563-570.

27. CALIN, T.; NICOLAIDIS, M.; VELAZCO, R. Upset hardened memory design for submicron CMOS technology. IEEE Transactions on Nuclear Science, New York, v.43, n.6, p. 2874 -2878, Dec. 1996.

28. VELAZCO, R. et al. Two CMOS memory cells suitable for the design of SEU-tolerant VLSI circuits. IEEE Transactions on Nuclear Science, New York, v.41, n.6, p. 2229-2234, Dec. 1994.

29. WISEMAN, D. et al. Design and Testing of SEU / SEL Immune Memory and Logic Circuits in a Commercial CMOS Process. In: IEEE NUCLEAR SPACE RADIATION EFFECTS CONFERENCE, NSREC, 1993. Proceedings… IEEE Computer Society, 1993.

30. LIU, M. N.; WHITAKER, S. Low power SEU immune CMOS memory circuits. IEEE Transactions on Nuclear Science, New York, v.39, n.6, p. 1679-1684, Dec. 1992.

31. WHITAKER, S.; CANARIS, J.; LIU, K. SEU hardened memory cells for a CCSDS Reed-Solomon encoder. IEEE Transactions on Nuclear Science, New York, v.38, n.6, p. 1471-1477, Dec. 1991.

32. FERNANDALIMA KASTENSMIDT, LUIGI CARRO,RICARDO REIS, FAULT-TOLERANCE TECHNIQUES FOR SRAM-BASED FPGAS

33. COTA, E. et al. Implementing a self-testing 8051 microprocessor. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 1999. Proceedings... Los Alamitos: IEEE Computer Society, 1999. p. 202-205.

34. LIMA, F.; REZGUI, S.; COTA, E.; CARRO, L.; LUBASZEWSKI, M.; VELAZCO, R.; REIS, R. Designing and Testing a Radiation Hardened 8051-like Micro-controller. In: INTERNATIONAL CONFERENCE ON MILITARY AND AEROSPACE APPLICATIONS OF PROGRAMMABLE LOGIC DEVICES, MAPLD, 2000. Proceedings..., 2000b.

35. LIMA, F.; CARRO, L.; VELAZCO, R.; REIS, R. Injecting Multiple Upsets in a SEU tolerant 8051 Micro-controller. In: LATIN AMERICA TEST WORKSHOP, LATW, 2002. Proceedings... Amissville: IEEE Computer Society, 2002a.

36. LIMA, F.; REZGUI, S.; CARRO, L.; VELAZCO, R.; REIS, R. On the use of VHDL simulation and emulation to derive error rates. In: EUROPEAN CONFERENCE ON RADIATION AND ITS EFFECTS ON COMPONENTS AND SYSTEMS, RADECS, 2001. Proceedings... IEEE Computer Society, 2001a. p. 253-260.

37. NEUBERGER, G.; LIMA, F.; CARRO, L.; REIS, R. A Multiple Bit Upset Tolerant SRAM Memory. Transactions on Design Automation of Electronic Systems, TODAES, New York, v.8, n.4, Oct. 2003.

38. CARMICHAEL, C. Triple Module Redundancy Design Techniques for Virtex® Series FPGA: Application Notes 197. San Jose, USA: Xilinx, 2000

39. LIMA, F.; CARMICHAEL, C.; FABULA, J.; PADOVANI, R.; REIS, R. A fault injection analysis of Virtex® FPGA TMR design methodology. In: EUROPEAN CONFERENCE ON RADIATION AND ITS EFFECTS ON COMPONENTS AND SYSTEMS, RADECS, 2001. Proceedings... IEEE Computer Society, 2001b. p. 275-282.

40. C.CARMICHAEL, M.CAFFREY, A. SALAZAR; Los Alamos National Laboratories, Correcting Single-Event Upsets Through Virtex Partial Configuration, XILINX 2000

41. K. CHAPMAN, SEU Strategies for Virtex-5 Devices, XILINX 2010

42. XILINX INC. Virtex-5 FPGA Configuration User Guide: Application Notes UG191 (v3.8) August 14, 2009

43. XILINX INC. Spartan-3 Generation Configuration User Guide Extended Spartan-3A, Spartan-3E, and Spartan-3 FPGA Families: Application Notes UG332 (v1.6) October 26, 2009

44. XILINX INC. ML505/ML506 Evaluation Platform User Guide: Application Notes UG347 (v2.2) April 18, 2007

45. XILINX INC. SEU Strategies for Virtex-5 Devices: Application Notes XAPP864 (v1.0.1) March 5, 2009

46. XILINX INC. PicoBlaze - Initial Design for Spartan-3E Starter Kit (LCD Display Control) Ken Chapman Xilinx Ltd 16th February 2006