# Σύστημα Εξομοίωσης Μόνιμων Σφαλμάτων σε FPGAs

**Γεώργιος Μπέτσης**

Μεταπτυχιακή Εργασία Εξειδίκευσης

— ♦ —

Ιωάννινα, Ιούλιος 2022


ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

# Σύστημα Εξομοίωσης Μόνιμων Σφαλμάτων σε FPGAs

Η Μεταπτυχιακή Διπλωματική Εργασία

υποβάλλεται στην ορισθείσα από τη συνέλευση
του Τμήματος Μηχανικών Η/Υ και Πληροφορικής
Εξεταστική Επιτροπή

από τον

Γεώργιο Μπέτση

ως μέρος των υποχρεώσεων για την απόκτηση του

ΔΙΠΛΩΜΑΤΟΣ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΣΤΗ ΜΗΧΑΝΙΚΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΜΕ ΕΙΔΙΚΕΥΣΗ
ΣΤΑ ΠΡΟΗΓΜΕΝΑ ΥΠΟΛΟΓΙΣΤΙΚΑ ΣΥΣΤΗΜΑΤΑ

Πανεπιστήμιο Ιωαννίνων
Πολυτεχνική Σχολή
Ιωάννινα 2022

Εξεταστική επιτροπή:

- **Τενέντες Βασίλειος,** Επίκουρος Καθηγητής Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Ιωαννίνων (Επιβλέπων)
- **Τσιατούχας Γεώργιος,** Καθηγητής Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Ιωαννίνων
- **Ευθυμίου Αριστείδης,** Επίκουρος Καθηγητής Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Ιωαννίνων

# ΠΕΡΙΕΧΟΜΕΝΑ

# ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

# Κατάλογος Πινάκων

# ΓΛΩΣΣΑΡΙ

| | |
|---|---|
| IP | Intellectual Property |
| SEM | Soft Error Mitigation |
| SEU | Single Event Upsets |
| MBU | Multiple Bit Upsets |
| TMR | Triple Modular Redundancy |
| ECC | Error Correction Codes |
| HFI | Hardware Fault Injection |
| SFI | Software Fault Injection |
| EFI | Emulation Fault Injection |
| FPGA | Field Programmable Gate Array |
| SRAM | Static Random Access Memory |
| DRAM | Dynamic Random Access Memory |
| RAM | Random Access Memory |
| BRAM | Block Random Access Memory |
| CRC | Cyclic Redundancy Check |
| DUT | Design Under Test |
| UART | Universal Asynchronous Receiver Transmitter |
| PVT | Process, Voltage, Temperature |
| IC | Integrated Circuit |
| ATPG | Automatic Test Pattern Generation |
| HDL | Hardware Description Language |
| PDF | Path Delay Faults |
| TF | Transient Faults |
| STF | Single Transient Faults |
| MTF | Multiple Transient Faults |
| DFT | Design for Testability |

# Περιληψη

Γεώργιος Μπέτσης, Δ.Μ.Σ. στη Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, Ιούλιος 2022
Σύστημα Εξομοίωσης Μόνιμων Σφαλμάτων σε FPGAs
Επιβλέπων: Βασίλειος Τενέντες Επίκουρος Καθηγητής

Η διάθεση αξιόπιστων ψηφιακών συστημάτων υλικού στην αγορά από την βιομηχανία αυτοματοποίησης του σχεδιασμού (Electronic Design Automation - EDA) και της κατασκευής ολοκληρωμένων συστημάτων υλικού απαιτεί 1) διαδικασίες ελέγχου της κατασκευαστικής τους ποιότητας και 2) διαδικασίες αξιολόγησης της αξιοπιστίας τους στο πεδίο της εφαρμογής τους. Μία από τις βασικές τεχνικές που εκτελείται τόσο κατά την σχεδίαση των διαδικασιών ελέγχου ολοκληρωμένων συστημάτων όσο και κατά την ανάλυση της αξιοπιστίας τους είναι η προσομοίωση σχεδιασμών υλικού παρουσίας σφαλμάτων (Faults simulation). Ωστόσο, οι παραδοσιακές τεχνικές προσομοίωσης σφαλμάτων είναι υπολογιστικά απαιτητικές διεργασίες αφού η πολυπλοκότητά τους επηρεάζεται από το μέγεθος του σχεδιασμού. Για την επιτάχυνση της σχεδίασης ολοκληρωμένων συστημάτων υλικού, η βιομηχανία του EDA αντικαθιστά διαδικασίες που απαιτούσαν χρονοβόρες προσομοιώσεις από συστήματα εξομοίωσης. Στην εργασία αυτή έχουν μελετηθεί τεχνικές εξομοίωσης τη διαδικασίας έγχυσης σφαλμάτων σε ψηφιακά συστήματα υλικού, μιας νέας κατηγορίας τεχνικών που στοχεύει στην επιτάχυνση των διεργασιών προσομοίωσης σφαλμάτων στα συστήματα υλικού, η οποία έχει υπολογιστική πολυπλοκότητα ανεξάρτητη του μεγέθους του σχεδιασμού. Έχει αναπτυχθεί ένα σύστημα αυτοματοποιημένης έγχυσης σφαλμάτων μόνιμης τιμής σε RTL σχεδιασμούς και η διαδικασία της εξομοίωσης σφαλμάτων υλοποιήθηκε σε Xilinx FPGA. Το σύστημα που αναπτύχθηκε αξιολογήθηκε με πειραματισμούς πάνω σε συνθετικούς σχεδιασμούς πολλαπλών σειρών λογικών αντιστροφέων και επιτυγχάνει ρυθμό εξομοίωσης σφαλμάτων, ο οποίος είναι ανεξάρτητος από το πλήθος των λογικών πυλών του σχεδιασμού. Ο ρυθμός εξομοίωσης σφαλμάτων έχει συγκριθεί με αυτόν που επιτυγχάνεται από τις κλασικές μεθόδους προσομοίωσης

σφαλμάτων που βασίζονται στο λογισμικό. Επιπρόσθετα, εξετάζοντας την δυνατότητα εφαρμογής του συστήματος στο πεδίο της εφαρμογής, το κόστος του συστήματος αξιολογήθηκε τόσο ως προς το επιπλέον υλικό, όσο και ως προς τον επιπλέον χρόνο απόκρισης με τα οποία επιβαρύνει έναν σχεδιασμό.

# EXTENDED ABSTRACT

George Betsis, M.Sc. in Data and Computer Systems Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, July 2022

Permanent Faults Emulation System in FPGAs

Advisor: Vasileios Tenentes Assistant Professor

In order to design and manufacture reliable digital hardware systems, the electronic design automation (EDA) and manufacturing of integrated systems industries require 1) manufacturing testing procedures for quality control; and 2) to be able to evaluate the reliability of devices in-the-field. A technique that is required for designing testing procedures and performing reliability assessment of electronic devices is the simulation of designs in the presence of faults (Faults simulation). However, traditional fault simulation techniques are computationally demanding processes since the design size impacts the complexity of simulations. For speeding up such computationally demanding processes, the EDA replaces simulation-based procedures with system emulation, whenever possible. In this Thesis, a system for fault emulation of digital circuits is developed which consists of an automated stuck-at fault injection process in RTL designs and a fault emulation process, which is implemented on a Xilinx FPGA platform. The developed system is evaluated through experiments on synthetic designs of cascaded inverters and performs with error simulation rate that is independent of the number of logic gates in the design. The error simulation rate is compared to that achieved by a classical software-based fault simulation technique. In addition, in order to evaluate the applicability of the developed system in-the-field, the cost of the injection mechanism on the design is evaluated in terms of additional logic and response time.

# ΚΕΦΑΛΑΙΟ 1

# ΕΙΣΑΓΩΓΗ



## 1.1 Προσομοίωση παρουσίας σφαλμάτων και εφαρμογές της

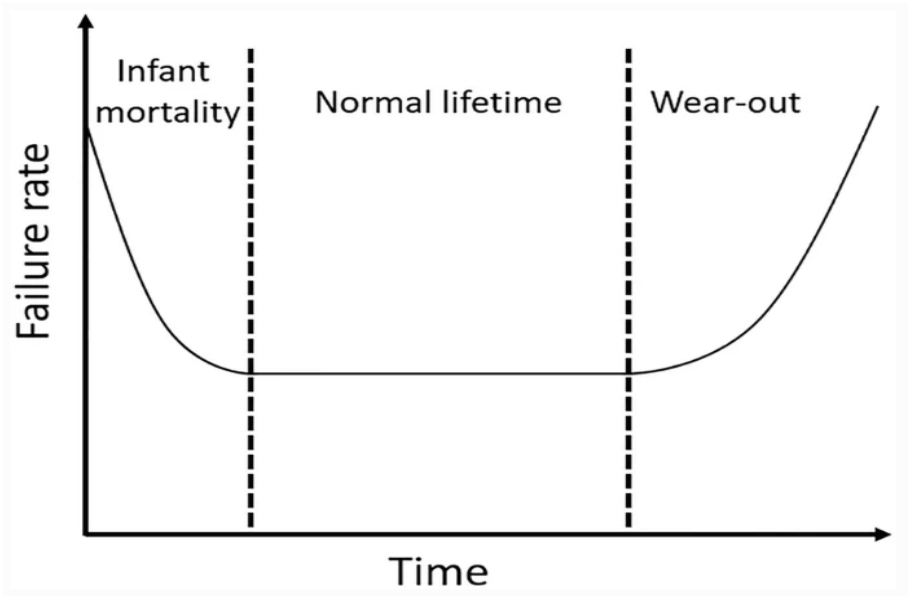
Η διάθεση αξιόπιστων ψηφιακών συστημάτων υλικού στην αγορά από την βιομηχανία σχεδιασμού και κατασκευής ολοκληρωμένων συστημάτων υλικού απαιτεί 1) διαδικασίες ελέγχου της κατασκευαστικής τους ποιότητας και 2) διαδικασίες αξιολόγησης της αξιοπιστίας τους στο πεδίο της εφαρμογής τους. Μία από τις βασικές τεχνικές που εκτελείται τόσο κατά την σχεδίαση των διαδικασιών ελέγχου ολοκληρωμένων συστημάτων όσο και κατά την ανάλυση της αξιοπιστίας τους είναι η προσομοίωση σχεδιασμών υλικού παρουσίας σφαλμάτων (Faults simulation). Η προσομοίωση σφαλμάτων αξιολογεί πως θα συμπεριφερθεί ένα ψηφιακό κύκλωμα παρουσία κατασκευαστικών ελαττωμάτων. Είναι ένα απαραίτητο εργαλείο για την αξιολόγηση της αξιοπιστίας των chip και για την επιβεβαίωση ότι αυτά θα λειτουργούν αξιόπιστα και σύμφωνα με τις προδιαγραφές τους. Λεπτομέρειες για την προσομοίωση σφαλμάτων υπάρχουν στο επόμενο κεφάλαιο. Ακολουθούν λεπτομέρειες των εφαρμογών της στον κατασκευαστικό έλεγχο και στην αξιολόγηση της αξιοπιστίας συσκευών στο πεδίο της εφαρμογής τους.

Ως αξιοπιστία στα ηλεκτρονικά κυκλώματα, ορίζουμε την πιθανότητα αυτό το κύκλωμα να λειτουργεί κανονικά και σωστά έτσι όπως προβλέπεται, κάτω από συγκεκριμένες συνθήκες, και για συγκεκριμένο χρόνο. Η αξιοπιστία εξαρτάται από πολλούς παράγοντες, όπως την ποιότητα των υλικών, την διαδικασία παραγωγής, των σχεδιασμό, τις δοκιμές, και το περιβάλλον κάτω από το οποίο λειτουργούν τα κυκλώματα. Όταν διενεργείται ανεπαρκής ανάλυση αξιοπιστίας σε ένα ηλεκτρονικό κύκλωμα, μπορεί να προκαλέσει απώλεια χρόνου, χρημάτων και ποιότητας ζωής.

Γι' αυτό λοιπόν, τα ηλεκτρονικά κυκλώματα πρέπει να καταλήγουν στην αγορά με αυξημένη αξιοπιστία, και απαλλαγμένα από σφάλματα.

Για τον έλεγχο της αξιοπιστίας των ηλεκτρονικών κυκλωμάτων, δεν υπάρχει μία καθολική και ενιαία μέθοδος, καθώς η αξιοπιστία μπορεί να ποικίλλει ανάλογα με διάφορους παράγοντες, όπως για παράδειγμα την ποιότητα των εξαρτημάτων που χρησιμοποιούνται, ο σχεδιασμός του κυκλώματος, και οι συνθήκες λειτουργίας. Ωστόσο, τα καλοσχεδιασμένα ηλεκτρονικά κυκλώματα, κατασκευασμένα με καλά υλικά, και με πολλές δοκιμές, μπορούν να είναι αρκετά αξιόπιστα. Η έγχυση σφάλματος (Fault Injection), είναι μία μέθοδος δοκιμής ηλεκτρονικών κυκλωμάτων με την οποία γίνεται εισαγωγή σφαλμάτων στο σύστημα, το οποίοι μπορεί να βοηθήσει στον εντοπισμό πιθανών προβλημάτων και να διασφαλίσει ότι το κύκλωμα είναι αρκετά ανθεκτικό ώστε να δουλέψει στις πραγματικές συνθήκες.

Η καμπύλη "bathtub" χρησιμοποιείται συχνά για την απεικόνιση της ζωής των ηλεκτρονικών κυκλωμάτων, και δείχνει ότι η αξιοπιστία του υλικού βελτιώνεται στην αρχή, στα στάδια σχεδιασμού, δοκιμών και υλοποίησης, έπειτα παραμένει σχετικά σταθερή, μέχρι το τέλος της ζωής των υλικών, όπου τα εξαρτήματα αρχίζουν να φθείρονται και να χαλάνε [2]. Το πρώτο από τα τρία κομμάτια της καμπύλης "bathtub", αφορά τον κατασκευαστικό έλεγχο.




Σχήμα 1.1 - Καμπύλη "bathtub" στην οποία απεικονίζεται η αξιοπιστία του υλικού [3]

Η διαδικασία κατασκευής των ημιαγωγών, μπορεί να προκαλέσει μόνιμα ελαττώματα σε ένα τσιπ κατά την διάρκεια ενός ή περισσότερων από τα βήματα

της διαδικασίας κατασκευής τους. Η ολοένα και μεγαλύτερη σμίκρυνση των συσκευών, καθιστά τα ηλεκτρονικά κυκλώματα όλο και πιο ευάλωτα σε αυτές τις κατασκευαστικές ανωμαλίες. Είναι σημαντικό οι δοκιμές κατά την διάρκεια κατασκευής, να είναι όσο το δυνατόν ενδελεχείς για να βρεθούν τα ελαττωματικά τσιπ πριν βγουν στην αγορά.

Επιπλέον, η κλιμάκωση της τεχνολογίας και η αυξανόμενη πολυπλοκότητα των ηλεκτρονικών κυκλωμάτων, προκαλούν ελαττώματα πιο δύσκολα στον εντοπισμό.

Η εισαγωγή νέων τεχνολογιών, ειδικά τεχνολογιών 14 nm ή μικρότερη, επέτρεψε στη βιομηχανία ημιαγωγών να συμβαδίσει με τις αυξημένες απαιτήσεις απόδοσης-χωρητικότητας από τους καταναλωτές. Παρόλα αυτά, κάθε νέα τεχνολογία έρχεται με νέα προβλήματα. Τα μικρότερα μεγέθη στα τσιπ, αυξάνουν την πιθανότητα να έχουν σφάλματα. Οι σύγχρονοι μικροεπεξεργαστές περιέχουν περισσότερα από 1000 pins. Εάν οποιοδήποτε τρανζίστορ μέσα σε ένα τσιπ γίνει ελαττωματικό, τότε ολόκληρο το τσιπ πρέπει να απορριφθεί. Ο εντοπισμός ενός μόνο ελαττωματικού τρανζίστορ από τα δισεκατομμύρια είναι δύσκολη διαδικασία. Η ιδέα να γίνει δοκιμή σε κάθε λειτουργικότητα του τσιπ με κάθε πιθανό συνδυασμό είναι τόσο χρονοβόρα που καθιστά κάτι τέτοιο απαγορευτικό.

Για να λυθεί αυτό το πρόβλημα, γίνεται χρήση μιας μεθοδολογίας που ονομάζεται DFT (συντομογραφία του Design for Testability). Και το χαρακτηριστικό που προσθέτει σε ένα τσιπ αυτή η μεθοδολογία, είναι η «δοκιμή» (Testability). Το DFT επιτυγχάνει δύο σημαντικούς στόχους στη διαδικασία κατασκευής ενός τσιπ.

α) Απόρριψη ελαττωματικών μονάδων (Ποιότητα προϊόντος)

Η διαδικασία της δοκιμής ελέγχει τα σφάλματα στη διαδικασία κατασκευής που δημιουργούν σφάλματα στα τσιπ που σχεδιάζονται. Εάν τα σφάλματα μπορούν να εντοπιστούν νωρίτερα, τότε η υποκείμενη διαδικασία που προκαλεί τα σφάλματα μπορεί να απορριφθεί σε αυτό το σημείο. Αυτό εξοικονομεί χρόνο και χρήμα καθώς τα ελαττωματικά τσιπ μπορούν να απορριφθούν ακόμη και πριν κατασκευαστούν.

β) Παρακολούθηση και βελτίωση της διαδικασίας παραγωγής

Η διαδικασία της δοκιμής εφαρμόζεται σε κάθε φάση σχεδιασμού του τσιπ, από το RTL έως την κατασκευή. Αυτό προσδιορίζει ακριβώς το στάδιο κατά το

οποίο το τεστ αποτυγχάνει. Αυτό απλοποιεί την ανάλυση αστοχίας προσδιορίζοντας την πιθανή θέση του ελαττώματος. Η σχολαστική παρακολούθηση βελτιώνει την ακρίβεια της διαδικασίας και μειώνει την πιθανότητα εμφάνισης σφάλματος.

Παρόλα αυτά το DFT δεν μπορεί να εγγυηθεί ότι το τσιπ δεν θα είναι ποτέ ελαττωματικό στο μέλλον. Μπορεί να προκύψουν σφάλματα ακόμα και όταν το τσιπ είναι στα χέρια του καταναλωτή. Ένα τσιπ μπορεί να έχει εσφαλμένη συμπεριφορά ανά πάσα στιγμή, εάν εκτεθεί σε πολύ υψηλή θερμοκρασία ή υγρό περιβάλλον ή λόγω γήρανσης.

## 1.2 Σκοπός και δομή εργασίας

Για την επιτάχυνση της σχεδίασης ολοκληρωμένων συστημάτων υλικού, η βιομηχανία αυτοματισμού της σχεδίασης ολοκληρωμένων συστημάτων (Electronic Design Automation – EDA) αντικαθιστά διαδικασίες που απαιτούν χρονοβόρες προσομοιώσεις από συστήματα εξομοίωσης. Για παράδειγμα στο Σχήμα 1.2 παρουσιάζεται ένα μηχάνημα που αποτελείται από συσκευές ανάπτυξης πρωτότυπων συστημάτων σε FPGAs, το οποίο διατίθεται από μία από τις μεγαλύτερες εταιρείες ανάπτυξης λογισμικού για EDA στους σχεδιαστές συστημάτων με σκοπό την επιτάχυνση των διαδικασιών σχεδίασης μέσω εξομοίωσης.


Σχήμα 1.2 - ZeBu EP1 - Μηχάνημα εξομοίωσης που αποτελείται από συσκευές ανάπτυξης πρωτότυπων συστημάτων σε FPGAs

Στην εργασία αυτή έχουν μελετηθεί τεχνικές εξομοίωσης τη διαδικασίας έγχυσης σφαλμάτων σε ψηφιακά συστήματα υλικού, μιας νέας κατηγορίας τεχνικών που στοχεύει στην επιτάχυνση των διεργασιών προσομοίωσης σφαλμάτων στα συστήματα υλικού, η οποία έχει υπολογιστική πολυπλοκότητα ανεξάρτητη του μεγέθους του σχεδιασμού.

Στο Κεφάλαιο 2, θα παρουσιαστούν λεπτομέρειες για τους τύπους σφαλμάτων και τις τεχνικές προσομοίωσης σφαλμάτων. Το βασικό πρόβλημα των παραδοσιακών τεχνικών προσομοίωσης σφαλμάτων, είναι η απαίτησή τους για μεγάλη υπολογιστική ισχύ. Στο ίδιο κεφάλαιο θα παρουσιαστούν τεχνικές εξομοίωσης έγχυσης σφαλμάτων σε ψηφιακά συστήματα υλικού, μιας νέας κατηγορίας τεχνικών που στοχεύει στην επιτάχυνση των διεργασιών προσομοίωσης σφαλμάτων στα συστήματα υλικού η οποία έχει υπολογιστική πολυπλοκότητα ανεξάρτητη του μεγέθους του σχεδιασμού.

Στο Κεφάλαιο 3, θα παρουσιαστεί ένα σύστημα που αναπτύχθηκε στα πλαίσια της εργασίας, το οποίο επιτυγχάνει αυτοματοποιημένη έγχυση σφαλμάτων μόνιμης τιμής σε RTL σχεδιασμούς και η διαδικασία της εξομοίωσης σφαλμάτων υλοποιήθηκε σε Xilinx FPGA.

Στο Κεφάλαιο 4, θα παρουσιαστούν πειραματικά αποτελέσματα από το σύστημα που αναπτύχθηκε. Οι πειραματισμοί για την αξιολόγηση του συστήματος έγινε σε συνθετικούς σχεδιασμούς πολλαπλών σειρών λογικών αντιστροφέων και επιτυγχάνει ρυθμό εξομοίωσης σφαλμάτων, ο οποίος είναι ανεξάρτητος από το πλήθος των λογικών πυλών του σχεδιασμού. Ο ρυθμός εξομοίωσης σφαλμάτων έχει συγκριθεί με αυτόν που επιτυγχάνεται από τις κλασικές μεθόδους προσομοίωσης σφαλμάτων που βασίζονται στο λογισμικό. Επιπρόσθετα, εξετάζοντας την δυνατότητα εφαρμογής του συστήματος στο πεδίο της εφαρμογής, το κόστος του συστήματος αξιολογήθηκε τόσο ως προς το επιπλέον υλικό, όσο και ως προς τον επιπλέον χρόνο απόκρισης με τα οποία επιβαρύνει έναν σχεδιασμό.

Στο Κεφάλαιο 5, παρουσιάζονται τα συμπεράσματα που προέκυψαν από την εξέλιξη της εργασίας.

# ΚΕΦΑΛΑΙΟ 2

# ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ



Στο κεφάλαιο αυτό παρουσιάζονται οι βασικές έννοιες που είναι απαραίτητες για την κατανόηση της παρούσας εργασίας. Επιπρόσθετα, παρουσιάζονται λεπτομέρειες για την προσομοίωση σφαλμάτων και τα συστήματα εξομοίωσης σφαλμάτων που έχουν εντοπιστεί.

## 2.1 Σφάλματα

Τα σφάλματα σε ένα ηλεκτρονικό κύκλωμα, μπορεί να ταξινομηθούν σύμφωνα με τα χρονικά του χαρακτηριστικά, το στάδιο του κύκλου ζωής του προϊόντος, της σοβαρότητάς του κτλ.

### i) Transient faults

Transient faults είναι τα σφάλματα που εμφανίζονται προσωρινά και μετά εξαφανίζονται. Μπορούν να προκληθούν από διάφορους παράγοντες, όπως ηλεκτρικό θόρυβο, ανωμαλίες στην τάση ισχύος και ηλεκτρομαγνητικές παρεμβολές [4] [12]. Τα τελευταία χρόνια, τα transient faults που προκαλούνται από ακτινοβολία έχουν λάβει μεγάλη προσοχή, καθώς αποτελούν μία από τις σημαντικότερες προκλήσεις για τη μελλοντική κλιμάκωση της τεχνολογίας στην αεροδιαστημικές εφαρμογές [13]. Στο παρελθόν, τα transient faults, αποτελούσαν

πρόβλημα μόνο στις μνήμες, με αποτέλεσμα να χρησιμοποιούνται ευρέως οι τεχνικές με τα Error Correcting Codes (ECC).

Επίσης, από τα 90nm και κάτω, οι πολύ μικρές διαστάσεις αλλά και τα χαμηλά επίπεδα τάσης λειτουργίας, οδηγούν σε συχνότερα σφάλματα που προκαλούνται από ακτινοβολία, με αποτέλεσμα τα ποσοστά σφάλματος να προσεγγίζουν αυτά των μνημών [14]. Επομένως, τα transient faults στα κυκλώματα γίνονται ένα σημαντικό πρόβλημα αξιοπιστίας για τα μελλοντικά κυκλώματα. Επιπλέον, δεδομένου ότι η απόσταση μεταξύ των επαφών μειώνεται με την κλιμάκωση, η ενέργεια των σωματιδίων ακτινοβολίας που απαιτείται για να προκαλέσει transient faults, μειώνεται. Αν και μέχρι στιγμής έχουν υπάρξει αρκετές σχεδιαστικές λύσεις που αντιμετωπίζουν το ζήτημα των transient faults, η πρόκληση πλέον (με δεδομένο τα ολοένα και μικρότερα μεγέθη των κυκλωμάτων) είναι η εύρεση λύσεων χαμηλού κόστους που παρουσιάζουν τις καλύτερες συμβιβαστικές λύσεις μεταξύ ισχύος, επιδόσεων και κόστους από τη μία πλευρά και αξιοπιστίας από την άλλη [15]. Η ανάλυση αξιοπιστίας αποδεικνύεται ότι είναι απαραίτητη στα πρώιμα στάδια σχεδιασμού για τη βελτίωση της διάρκειας ζωής του συστήματος.

Ως εκ τούτου, μια γρήγορη και ακριβής εκτίμηση των ποσοστών σφάλματος που προκύπτουν από single transient faults (STF) και multiple transient faults (MTFs) σε λογικά κυκλώματα είναι ζωτικής σημασίας για τον προσδιορισμό των χαρακτηριστικών που απαιτούνται για μελλοντικά αξιόπιστα κυκλώματα.

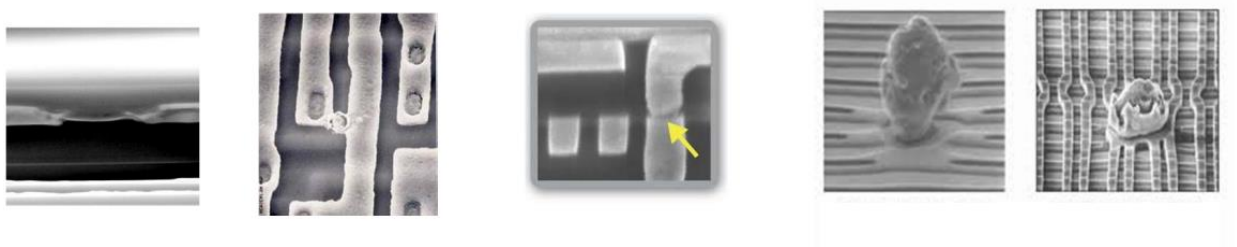
**SEU Faults**

Ένα Single Event Upset (SEU) σε ένα λογικό κύκλωμα, προκύπτει από ένα μόνο χτύπημα σωματιδίου πάνω σε ένα ευαίσθητο κύκλωμα, όπως για παράδειγμα σε έναν microcontroller, σε μία μνήμη ή σε έναν επεξεργαστή. Αναφέρεται συχνά ως Single Event Transient (SET). Τα SEU οφείλονται σε σωματίδια που συγκρούονται με άτομα στην ατμόσφαιρα, δημιουργώντας νετρόνια και πρωτόνια, τα οποία με τη σειρά τους μπορεί να αλληλεπιδράσουν με ηλεκτρονικά κυκλώματα.

Στο διάστημα, ιονίζοντα σωματίδια υψηλής ενέργειας υπάρχουν ως μέρος του φυσικού υποβάθρου, που αναφέρεται ως γαλαξιακή κοσμική ακτινοβολία. Τα σωματίδια αυτά, έχουν τόσο υψηλές ενέργειες, που καθιστούν την αυξημένη θωράκιση των διαστημοπλοίων άχρηστη όσον αφορά την εξάλειψη των SEU και των καταστροφικών αποτελεσμάτων τους.

### ii) Μόνιμα σφάλματα Stuck-at

Τα Stuck-at faults είναι τα σφάλματα που προκαλούνται από μόνιμα ελαττώματα στο πυρίτιο, τα οποία είτε υπάρχουν λόγω ατελειών κατά την παραγωγική διαδικασία, είτε προκαλούνται από το φαινόμενο της φθοράς κατά την διάρκεια της χρήσης. Η αύξηση του αριθμού των τρανζίστορ ανά σταθερή επιφάνεια, αυξάνει την πιθανότητα να έχουμε περισσότερα stuck-at σφάλματα σε ένα δεδομένο ηλεκτρονικό κύκλωμα. Επιπλέον, τα ηλεκτρονικά κυκλώματα λειτουργούν ως επί το πλείστων σε υψηλές συχνότητες χρονισμού και σε υψηλή τάση, επομένως αντιμετωπίζουν επιταχυνόμενη γήρανση λόγω θερμοκρασίας και τάσης [5].



Σχήμα 2.1 - Τυχαία ελαττώματα μπορεί να προκαλούνται από τυχαία σωματίδια πάνω στα κυκλώματα κατά τη διάρκεια της κατασκευής ή και κατά την διάρκεια της χρήσης

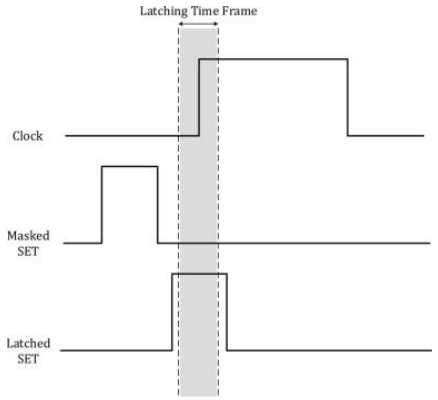
Τα μόνιμα σφάλματα είναι αποτέλεσμα μη αναστρέψιμων αλλαγών στη συσκευή και στο κύκλωμα, όπως για παράδειγμα το electromigration (ηλεκτρομετανάστευση), το οποίο συμβαίνει όταν η ένταση του ρεύματος είναι μεγάλη, και ταυτόχρονα η θερμοκρασία στον αγωγό πολύ υψηλή, και κατά συνέπεια άτομα από τον αγωγό διασπώνται και μετακινούνται. Αυτό το γεγονός προκαλεί είτε open-circuits, είτε short-circuits.

Κατά την διάρκεια της εξομοίωσης σφαλμάτων, τα permanent faults χρησιμοποιούνται περισσότερο για τον κατασκευαστικό έλεγχο των ολοκληρωμένων κυκλωμάτων, για να πιστοποιηθεί με αυτό τον τρόπο, ότι το τσιπ θα φτάσει στον τελικό χρήστη και δεν θα έχει σφάλματα. Τα stuck-at faults είναι ένα συγκεκριμένο μοντέλο σφάλματος που χρησιμοποιείται από προσομοιωτές σφαλμάτων και εργαλεία Automatic Test Pattern Generation (ATPG) για να μιμηθεί ένα κατασκευαστικό ελάττωμα σε ένα ολοκληρωμένο κύκλωμα. Υποτίθεται ότι τα μεμονωμένα σήματα και οι ακίδες είναι κολλημένα στα λογικά '1', '0' και 'X'. Για παράδειγμα, μια είσοδος συνδέεται με μια λογική κατάσταση 1 κατά τη δημιουργία

δοκιμής για να διασφαλιστεί ότι ένα κατασκευαστικό ελάττωμα με αυτόν τον τύπο συμπεριφοράς μπορεί να βρεθεί με ένα συγκεκριμένο πρότυπο δοκιμής. Ομοίως, η είσοδος θα μπορούσε να συνδεθεί με ένα λογικό 0 για να μοντελοποιήσει τη συμπεριφορά ενός ελαττωματικού κυκλώματος που δεν μπορεί να αλλάξει την ακίδα εξόδου του.

### iii) Timing errors

Σε αντίθεση με τους δύο προηγούμενους τύπους σφαλμάτων, τα ηλεκτρονικά κυκλώματα που έχουν timing errors, συνεχίζουν να παρέχουν σωστές λογικές εξόδους. Ωστόσο, έχουν μεγαλύτερες καθυστερήσεις μεταξύ των σημάτων εισόδου και εξόδου. Σφάλματα που α) προκαλούνται από τη αλλαγή των ηλεκτρικών χαρακτηριστικών των στοιχείων ενός κυκλώματος, β) που προκαλούνται από μεταβλητότητα PVT (process, voltage, temperature), γ) κατασκευαστικά ελαττώματα και δ) γήρανση, ευθύνονται για αυτόν τον τύπο σφαλμάτων. Με τη συνεχή μείωση στο μέγεθος των ηλεκτρονικών κυκλωμάτων, υπάρχει μια αυξανόμενη αβεβαιότητα για τη συμπεριφορά χρονισμού των σημερινών IC, που συχνά εκδηλώνονται ως σπάνια σφάλματα χρονισμού [6].




Σχήμα 2.2 - Παρουσία σφάλματος σε ένα κύκλωμα. Μη ανιχνεύσιμο λόγω του timing masking effect [11]

#### Transition delay Faults

Τα transition delay faults, είναι σφάλματα που προκαλούνται στη λειτουργία ενός κυκλώματος με βάση το χρονισμό του. Προκαλούνται από τους πεπερασμένους χρόνους ακμής (ανόδου και πτώσης) των σημάτων στις λογικές πύλες, καθώς και

από την καθυστέρηση διάδοσης των διασυνδέσεων μεταξύ των λογικών πυλών. Ο χρονισμός του κυκλώματος πρέπει να αξιολογηθεί προσεκτικά για να αποφευχθούν τέτοια σφάλματα στη λειτουργία του. Πρέπει να δημιουργηθούν δοκιμές ειδικά για να ληφθούν υπόψη αυτά τα σφάλματα. Αυτές οι δοκιμές μπορούν επίσης να χρησιμοποιηθούν για τον προσδιορισμό της σωστής συχνότητας ρολογιού στην οποία θα μπορούσε να λειτουργήσει το κύκλωμα και να εξακολουθεί να έχει τη σωστή λειτουργία του κυκλώματος.

Λόγω αυτού του πεπερασμένου χρόνου που χρειάζεται για να εμφανιστεί μια είσοδος μιας πύλης στην έξοδο, ενδέχεται να προκύψουν σφάλματα εάν δεν δοθεί ο απαιτούμενος χρόνος στα σήματα. Μία από τις προκλήσεις στη δοκιμή είναι η διάκριση μεταξύ ενός τέτοιου σφάλματος καθυστέρησης, όπου η έξοδος αποδίδει το σωστό αποτέλεσμα, και ενός πραγματικού σφάλματος στη λειτουργία του κυκλώματος.

Υπάρχουν δύο τύποι transition delay faults, slow-to-rise, όπου υπάρχει καθυστέρηση στην θετική ακμή, και slow-to-fall όπου υπάρχει καθυστέρηση στην αρνητική ακμή. Για ένα slow-to-rise σφάλμα, μπορεί να χρησιμοποιηθεί δοκιμή stuck-at-0 και stuck-at-1. Κατά την αρχικοποίηση τοποθετείται ένα 0 στη γραμμή (net) και μετά τον πρώτο κύκλο, τοποθετείται ένα 1 στο ίδιο σημείο. Με τον αντίστοιχο τρόπο, μπορεί γίνει δοκιμή για ένα slow-to-fall σφάλμα.

### Path delay Faults

Η ολοένα αυξανόμενες απαιτήσεις για την απόδοση στα ολοκληρωμένα κυκλώματα, δείχνουν την ανάγκη για την λειτουργία των κυκλωμάτων σε ολοένα και υψηλότερους χρονισμούς. Όσο όμως ανεβαίνει ο χρονισμός, τόσο αυξάνεται και ο κίνδυνος για Path delay Faults (PDF), τα οποία είναι σφάλματα που εμφανίζονται όταν η διαδρομή ενός ηλεκτρονικού σήματος καθυστερεί. Αυτό μπορεί να συμβεί όταν το σήμα διασχίζει ένα στοιχείο που δεν λειτουργεί σωστά, ή όταν η διαδρομή είναι φυσικά μεγάλη. Τα path delay faults, μπορεί να προκαλέσουν την άφιξη του σήματος πολύ αργά, ή πολύ νωρίς, γεγονός που μπορεί να προκαλέσει δυσλειτουργία του κυκλώματος.

Αυτό δημιουργεί την ανάγκη για πραγματοποίηση δοκιμών για τον έλεγχο της σωστής χρονικής συμπεριφοράς, γνωστή ως path delay. Τα Path delay Faults [11],

που χρησιμοποιούνται συνήθως σε δοκιμές καθυστέρησης, μοντελοποιούν σφάλματα ως καθυστερήσεις διάδοσης κατά μήκος διαφόρων paths των κυκλωμάτων.

### Ageing Faults

Τα σφάλματα λόγω γήρανσης των τσιπ (ageing faults), είναι ένα αυξανόμενο πρόβλημα, αλλά μέχρι στιγμής δεν έχει αντιμετωπιστεί στον σχεδιασμό και στην υλοποίηση των κυκλωμάτων. Αυτό θα αλλάξει σημαντικά, καθώς δημιουργούνται νέες απαιτήσεις αξιοπιστίας σε αγορές όπως η αυτοκινητοβιομηχανία, οι βιομηχανικές εφαρμογές ή οι διαστημικές εφαρμογές, όπου το κόστος αποτυχίας είναι πολύ υψηλό, οι οποίες απαιτούν πλήρη ανάλυση και αντιμετώπιση των παραγόντων που επηρεάζουν τη γήρανση. Οι συσκευές, καταπονούνται όλο και πιο πολύ. Λειτουργούν 24 ώρες την ημέρα, 7 ημέρες την εβδομάδα. Πρόκειται για συνεχή καταπόνηση.

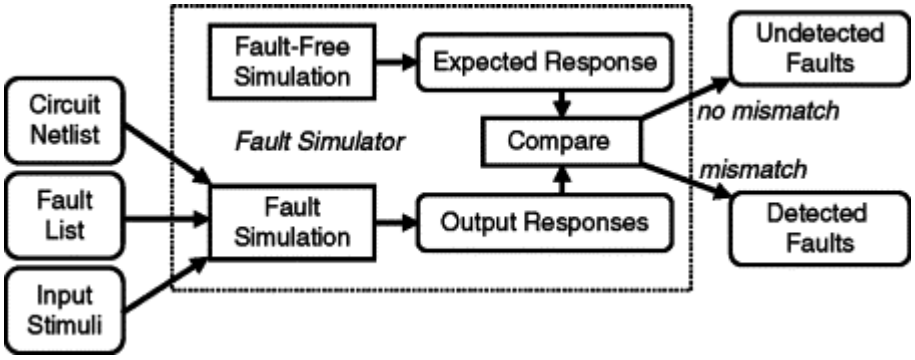
Οι συσκευές ημιαγωγών γερνούν με την πάροδο του χρόνου, αλλά αυτό που συχνά δεν είναι καλά κατανοητό είναι οι μηχανισμοί γήρανσης ή τα όρια που θα προκαλέσουν την αποτυχία ενός τσιπ. Επιπλέον, είναι βέβαιο ότι θα υπάρξουν απαιτήσεις στο μέλλον για μια ελάχιστη διάρκεια ζωής μιας συσκευής, η οποία θα εξαρτάται από την εφαρμογή. Δεδομένου ότι οι διαδικασίες γήρανσης είναι πολύπλοκες και συχνά δύσκολο να προβλεφθούν πλήρως, πολλά σχέδια τσιπ σήμερα είναι συχνά over-designed, ώστε να εξασφαλίζεται επαρκές περιθώριο για αξιόπιστη λειτουργία καθ' όλη τη διάρκεια ζωής.

Αυτό που το κάνει ακόμη πιο ανησυχητικό είναι ότι ορισμένοι τρόποι αστοχίας είναι τυχαίοι. Αν τα σφάλματα από την γήρανση μπορούσαν να γίνουν πιο ντετερμινιστικά, τότε θα γινόταν να αναπτυχθούν τσιπ που αντιδρούν και προσαρμόζονται στις επιδράσεις της γήρανσης ή ακόμη και να γίνει πρόβλεψη πότε μπορεί να συμβεί αστοχία του τσιπ.

## 2.2 Προσομοίωση Σφαλμάτων

Η προσομοίωση σφαλμάτων είναι μια τεχνική που χρησιμοποιείται για τον έλεγχο των ηλεκτρονικών κυκλωμάτων για ύπαρξη πιθανών σφαλμάτων που μπορεί να προκαλέσουν δυσλειτουργία του κυκλώματος. Η προσομοίωση σφάλματος επιτρέπει τη δοκιμή ενός κυκλώματος υπό διαφορετικές συνθήκες για να δούμε πώς ανταποκρίνεται το κύκλωμα σε ένα σφάλμα.

Για την προσομοίωση σφαλμάτων, υπάρχουν διάφορες τεχνικές, όπως για παράδειγμα σειριακών, παράλληλων, ταυτόχρονων και διαφορικών σφαλμάτων. Αυτές οι τεχνικές έχουν διάφορα χαρακτηριστικά, αλλά το πιο βασικό είναι ότι όσο μεγαλύτερη ακρίβεια έχουνε, τόσο περισσότερο χρόνο εκτέλεσης θέλουν, ο οποίος είναι κρίσιμος για τα ηλεκτρονικά κυκλώματα μεγάλης πολυπλοκότητας [7].




Σχήμα 2.3 - Τα αποτελέσματα από την προσομοίωση, συγκρίνονται με τα fault-free αποτελέσματα, για να διαπιστωθεί αν το CUT είναι προβληματικό ή όχι [8]

Η προσομοίωση σφάλματος χρησιμοποιείται για τη μέτρηση της αποτελεσματικότητας των μοτίβων δοκιμής στον εντοπισμό ελαττωμάτων που ενδεχομένως έχουν εισαχθεί κατά τη διαδικασία κατασκευής. Αυτό απαιτεί προσομοίωση της ελαττωματικής συμπεριφοράς του ηλεκτρονικού κυκλώματος για την ανίχνευση των μοντελοποιημένων σφαλμάτων ενδιαφέροντος. Επιπλέον, η προσομοίωση σφαλμάτων είναι αναπόσπαστο στοιχείο οποιουδήποτε προγράμματος ATPG.

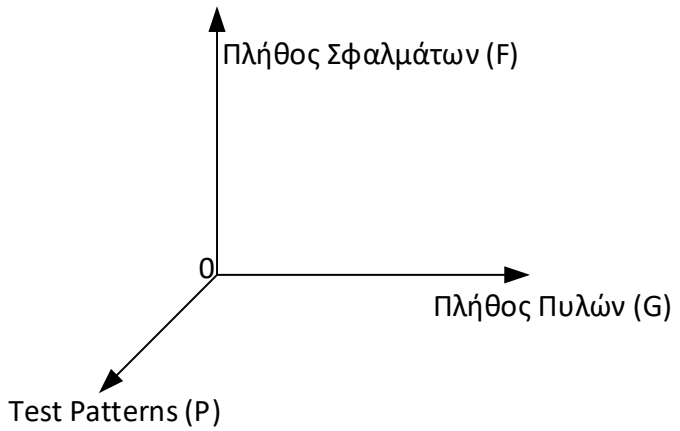
Η προσομοίωση σφαλμάτων, ασχολείται με τον έλεγχο της συμπεριφοράς των κατασκευασμένων κυκλωμάτων ως συνέπεια των αναπόφευκτων ατελειών της διαδικασίας κατασκευής. Τα κατασκευαστικά ελαττώματα (π.χ. βραχυκυκλώματα και ανοίγματα), εάν υπάρχουν, μπορεί να προκαλέσουν διαφορετική συμπεριφορά των κυκλωμάτων από την αναμενόμενη συμπεριφορά. Η προσομοίωση σφαλμάτων

γενικά προϋποθέτει ότι ο σχεδιασμός είναι λειτουργικά σωστός (δηλαδή, χωρίς σφάλματα σχεδιασμού) και στοχεύει στην καταγραφή κατασκευαστικών ελαττωμάτων.

## 2.3 Επιτάχυνση της διαδικασίας Προσομοίωσης Σφαλμάτων

Για την δοκιμή ενός κυκλώματος, χρησιμοποιούνται προγράμματα δομικών δοκιμών προκειμένου να μειωθεί ο χρόνος που απαιτείται για τη δοκιμή του κατασκευασμένου IC σε σύγκριση με μια εξαντλητική λειτουργική δοκιμή. Οι δομικές δοκιμές βασίζονται στην ανάπτυξη διανυσμάτων δοκιμής (test vectors) για την ανίχνευση συγκεκριμένων σφαλμάτων που θεωρούνται ότι υπάρχουν σε ένα κύκλωμα. Η δημιουργία των απαραίτητων διανυσμάτων δοκιμής πραγματοποιείται με τη χρήση τεχνικών και εργαλείων δημιουργίας προτύπων δοκιμής (test patterns) και προσομοίωσης σφαλμάτων.

Η προσομοίωση σφαλμάτων, είναι μία απαιτητική εργασία, λόγω της μεγάλης πολυπλοκότητας.




Σχήμα 2.4 – Χρονική πολυπλοκότητα της προσομοίωσης σφάλματος

Η χρονική πολυπλοκότητα της προσομοίωσης σφάλματος, είναι:

$$Complecity = P \times F \times G \sim O(G^3)$$

Κατά την προσομοίωση ενός σφάλματος τη φορά, ο χρόνος υπολογισμού είναι περίπου ανάλογος με το μέγεθος του κυκλώματος, τον αριθμό των test patterns και τον αριθμό των μοντελοποιημένων σφαλμάτων. Επειδή ο αριθμός των test patterns είναι κατά προσέγγιση ανάλογος με το μέγεθος του κυκλώματος, η συνολική χρονική πολυπλοκότητα της προσομοίωσης σφάλματος είναι $O(pg) \sim O(g^2)$, όπου $p$ το πλήθος των test patterns, και $g$ το πλήθος των λογικών πυλών, κάτι που καθίσταται ανέφικτο για μεγάλα κυκλώματα. Για τη βελτίωση της απόδοσης προσομοίωσης σφαλμάτων, έχουν αναπτυχθεί διάφορες τεχνικές προσομοίωσης σφαλμάτων.

Κρίνεται λοιπόν απαραίτητο, να βρεθούν νέες τεχνικές για την επιτάχυνση της ταχύτητας προσομοίωσης σφαλμάτων, υπάρχουν τέτοιες τεχνικές οι οποίες παρουσιάζονται εκτενώς στο κεφάλαιο 2.3. Στην παρούσα εργασία θα εστιάσουμε στις τεχνικές εξομοίωσης με την χρήση FPGAs [9].

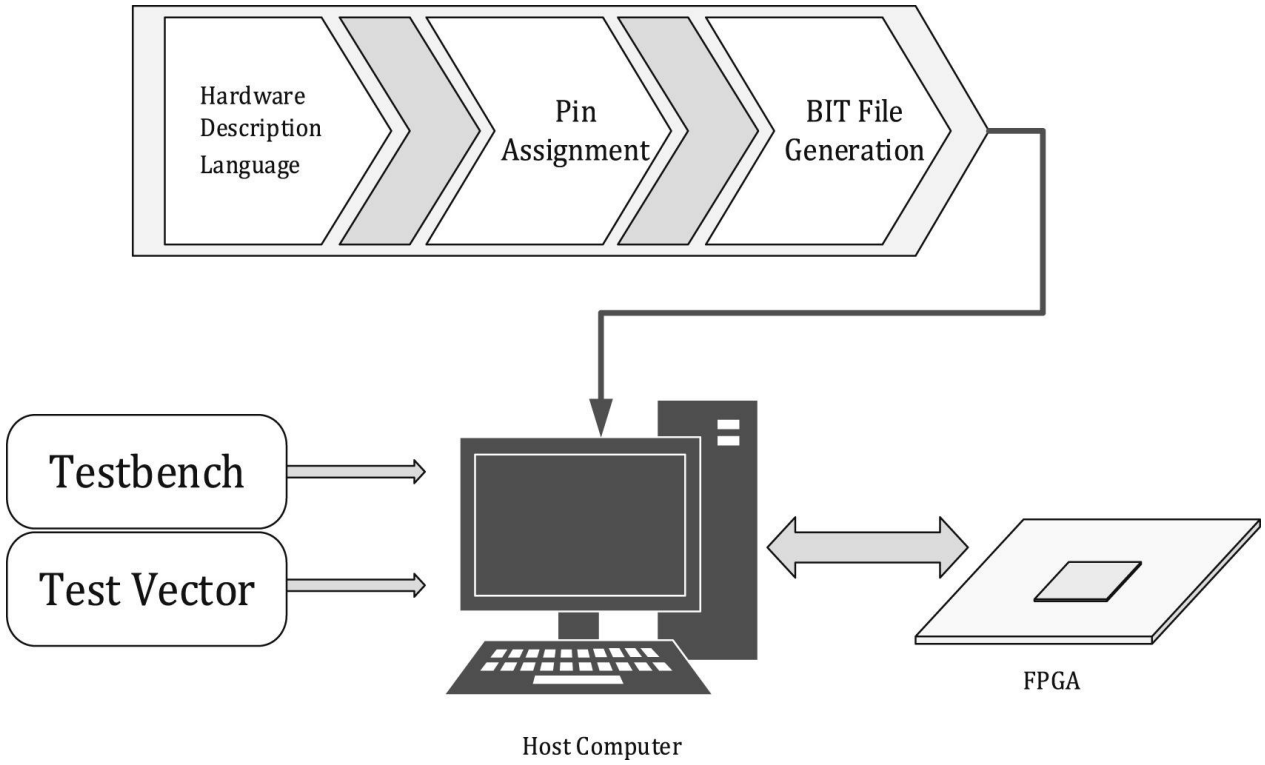
Στην εργασία αυτή θα επιταχύνουμε την προσομοίωση σφαλμάτων χρησιμοποιώντας εξομοιώσεις σφαλμάτων σε FPGAs, μια διαδικασία που θα αναφέρουμε ως εξομοίωση σφαλμάτων.

## 2.4 FPGAs

Οι συσκευές FPGA είναι ολοκληρωμένα κυκλώματα, με δυνατότητα προγραμματισμού μετά την κατασκευή. Ο προγραμματισμός των FPGA γίνεται χρησιμοποιώντας μια γλώσσα HDL. Τα FPGAs περιέχουν συστοιχίες από προγραμματιζόμενα λογικά blocks και μία σειρά διασυνδέσεων που επιτρέπουν στα block να συνδεθούν μεταξύ τους. Τα λογικά blocks μπορούν να διαμορφωθούν για να εκτελούν σύνθετες συνδυαστικές λειτουργίες ή να λειτουργούν ως απλές λογικές πύλες όπως AND και XOR. Στα περισσότερα FPGA, τα λογικά block περιλαμβάνουν επίσης στοιχεία μνήμης, τα οποία μπορεί να είναι απλά flip-flops ή πιο μεγάλα μπλοκ μνήμης [10].

Τα FPGA μπορούν να επαναπρογραμματιστούν για να υλοποιήσουν διαφορετικές λογικές λειτουργίες, επιτρέποντας ευέλικτους επαναδιαμορφώσιμους υπολογισμούς όπως ακριβώς εκτελούνται σε ένα λογισμικό σε έναν υπολογιστή.

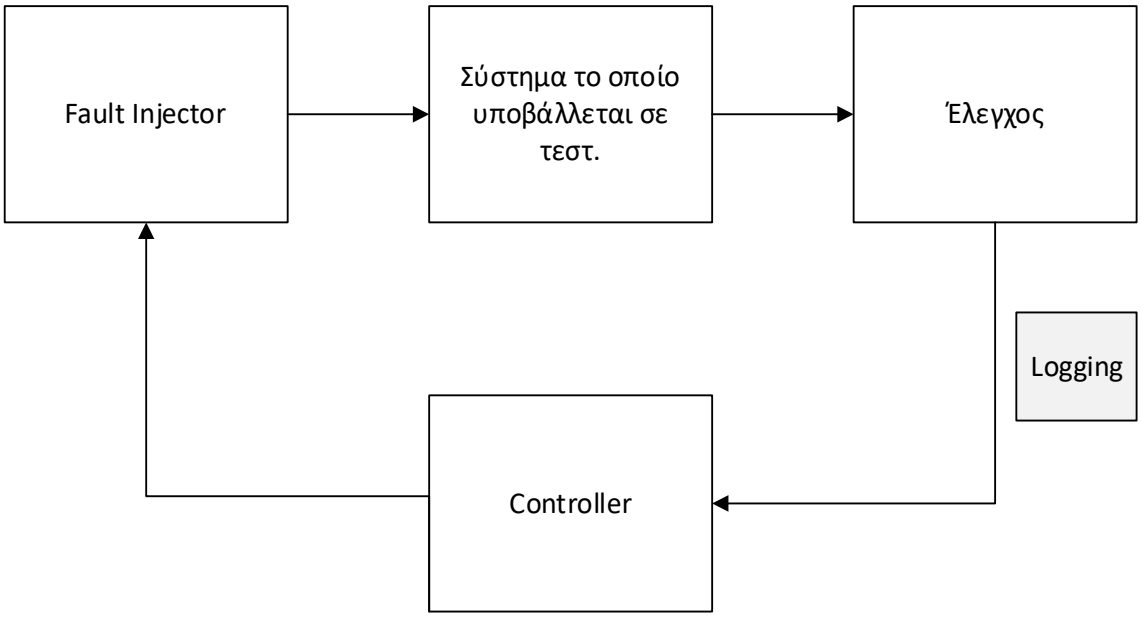
Στην παρούσα εργασία η τεχνική εξομοίωσης σφαλμάτων που θα αναπτυχθεί γίνεται με χρήση FPGA.




Σχήμα 2.5 - Προσομοίωση fault injection, με χρήση FPGA [11]

## 2.5 Τεχνικές προσομοίωσης παρουσίας σφαλμάτων (Fault Simulations)

Η έγχυση σφαλμάτων είναι μια ευρέως χρησιμοποιούμενη τεχνική για την αξιολόγηση της ανοχής στα σφάλματα κατά την διάρκεια κανονικής λειτουργίας των συστημάτων. Ουσιαστικά, οι μέθοδοι έγχυσης σφαλμάτων που παρουσιάζονται στη βιβλιογραφία εφαρμόζονται είτε με στοιχεία υλικού είτε με στοιχεία λογισμικού [16]. Μια κοινή αρχιτεκτονική για αυτού του είδους τα συστήματα παρουσιάζεται στο παρακάτω σχήμα:




Σχήμα 2.6 - Αρχιτεκτονική για Fault Simulations Techniques

όπου:

Ο controller είναι το στοιχείο που δημιουργεί τα ελεγχόμενα σφάλματα και υπολογίζει τα ληφθέντα αποτελέσματα.

Το Fault Injector επιβλέπει την εφαρμογή της έγχυσης σφάλματος που ορίζεται από τον ελεγκτή.

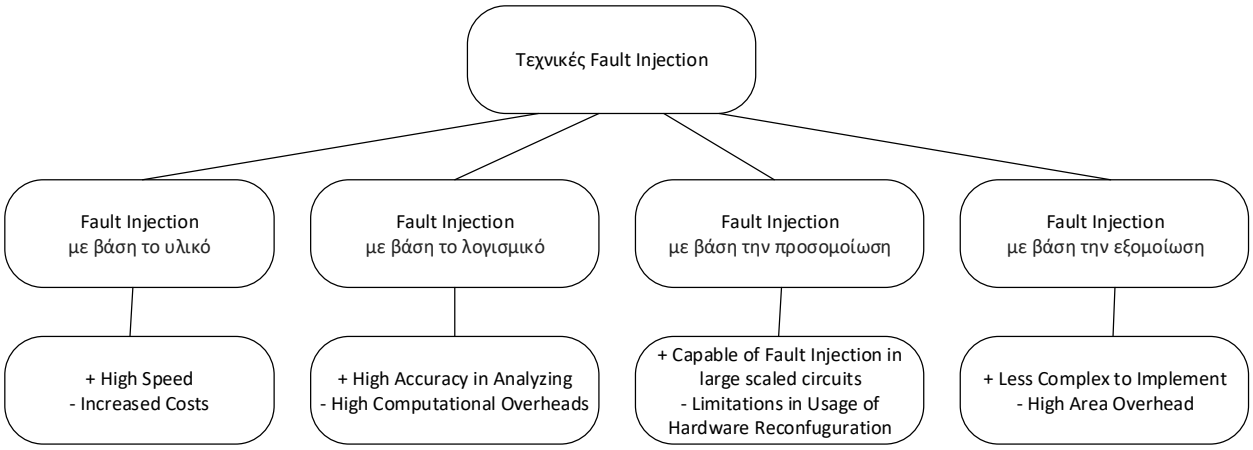
Το σύστημα που υποβάλλεται σε τεστ, είναι το κύκλωμα στόχος που πρέπει να μελετηθεί.

Ο Έλεγχος είναι η διεπαφή μετάδοσης μηνυμάτων για τον εντοπισμό όλων των αλληλεπιδράσεων του συστήματος.

Μια ευρέως αποδεκτή ταξινόμηση των διαφορετικών στρατηγικών έγχυσης σφαλμάτων, συνοψίζεται στις επόμενες υποενότητες, συμπεριλαμβανομένης μιας περιγραφής κάθε περίπτωσης και ορισμένων παραδειγμάτων.

Οι τεχνικές έγχυσης σφαλμάτων μπορούν να ταξινομηθούν ως εξής:

1. Με βάση την έγχυση σφαλμάτων υλικού (hardware fault injection).
2. Με βάση την έγχυση σφαλμάτων λογισμικού (software fault injection).
3. Με βάση την προσομοίωση σφάλματος έγχυσης (simulation fault injection).
4. Με βάση την εξομοίωση έγχυσης σφάλματος (emulation fault injection).




Σχήμα 2.7 - Κατηγορίες στις τεχνικές Fault Injection, και επιγραμματική αναφορά στα προτερήματα/μειονεκτήματα

### 2.5.1. Έγχυση σφάλματος με βάση το υλικό

Η έγχυση σφάλματος με βάση το υλικό συνίσταται στη δημιουργία φυσικών σφαλμάτων στα ολοκληρωμένα κυκλώματα. Οι δύο κύριες επιλογές είναι η έγχυση σφάλματος με επαφή και η έγχυση σφάλματος χωρίς επαφή. Στην πρώτη κατηγορία, υπάρχουν συστήματα όπως η έγχυση σφάλματος σε επίπεδο ακροδεκτών, η οποία βασίζεται στην ιδέα της διατάραξης των ολοκληρωμένων κυκλωμάτων με σφάλματα που εισάγονται στους ακροδέκτες που μιμούνται τόσο εξωτερικά όσο και εσωτερικά σφάλματα. Μερικά εργαλεία στη βιβλιογραφία είναι τα: RIFLE, FOCUS, MESSALINE και AFIT [17]. Από την άλλη πλευρά, η έγχυση σφάλματος χωρίς επαφή, βασίζεται στην ιδέα ότι η έγχυση δεν έχει άμεση φυσική επαφή με το υπό δοκιμή σύστημα. Σε αυτές τις περιπτώσεις, μια εξωτερική πηγή παράγει ένα φυσικό φαινόμενο όπως μια βαριά ακτινοβολία ιόντων που αλληλεπιδρά με το κύκλωμα και παράγει τα σφάλματα. Μερικά εργαλεία μπορούν να βρεθούν στη βιβλιογραφία είναι τα: FIST και MARS [18].

### 2.5.2. Έγχυση σφάλματος με βάση το λογισμικό

Το Software Fault Injection (SFI) εισάγει τεχνητά σφάλματα και καταστάσεις σφάλματος σε ένα σύστημα λογισμικού. Αυτά τα σφάλματα μπορούν να εισαχθούν κατά τη διάρκεια του χρόνου μεταγλώττισης ή του χρόνου εκτέλεσης.

Αυτά που βασίζονται σε σφάλματα κατά τη διάρκεια του χρόνου μεταγλώττισης εισάγουν τα σφάλματα στον πηγαίο κώδικα, του υπό δοκιμή προγράμματος.

Στην περίπτωση σφαλμάτων που εισάγονται κατά τη διάρκεια του χρόνου εκτέλεσης, είναι απαραίτητος ένας μηχανισμός trigger για την εισαγωγή των βλαβών. Αυτό το trigger (έναυσμα) δημιουργείται συνήθως μέσω:

Ενός timeout, όπου ένα χρονόμετρο λήγει ξεκινώντας το fault injection.

Ένα trap στο λογισμικό όπου ο έλεγχος του λογισμικού μεταφέρεται στη fault injection μονάδα.

Η εισαγωγή συγκεκριμένης εισόδου, αλλάζει τις οδηγίες του προγράμματος προκαλώντας την έγχυση σφάλματος σύμφωνα με την βούλησή μας.

Μεταξύ άλλων, ορισμένα σχετικά εργαλεία είναι: FERRARI, Orchestra, FTAPE, FIAT και XCEPTION [19].

### 2.5.3. Έγχυση σφάλματος με βάση την προσομοίωση

Η έγχυση σφάλματος βάσει προσομοίωσης είναι ένας μηχανισμός όπου το υπό δοκιμή σύστημα, προσομοιώνεται μέσω μιας γλώσσας όπως η VHDL ή η Verilog και τα σφάλματα εγχέονται μέσω λογισμικού. Οι κύριες επιλογές που μπορούν να βρεθούν σε αυτού του είδους τα συστήματα είναι:

Αυτά που τροποποιούν την σχεδίαση του υπό δοκιμή συστήματος, προσθέτοντας μια μονάδα σαμποτέρ, η οποία είναι υπεύθυνη για τη διαδικασία έγχυσης σφαλμάτων.

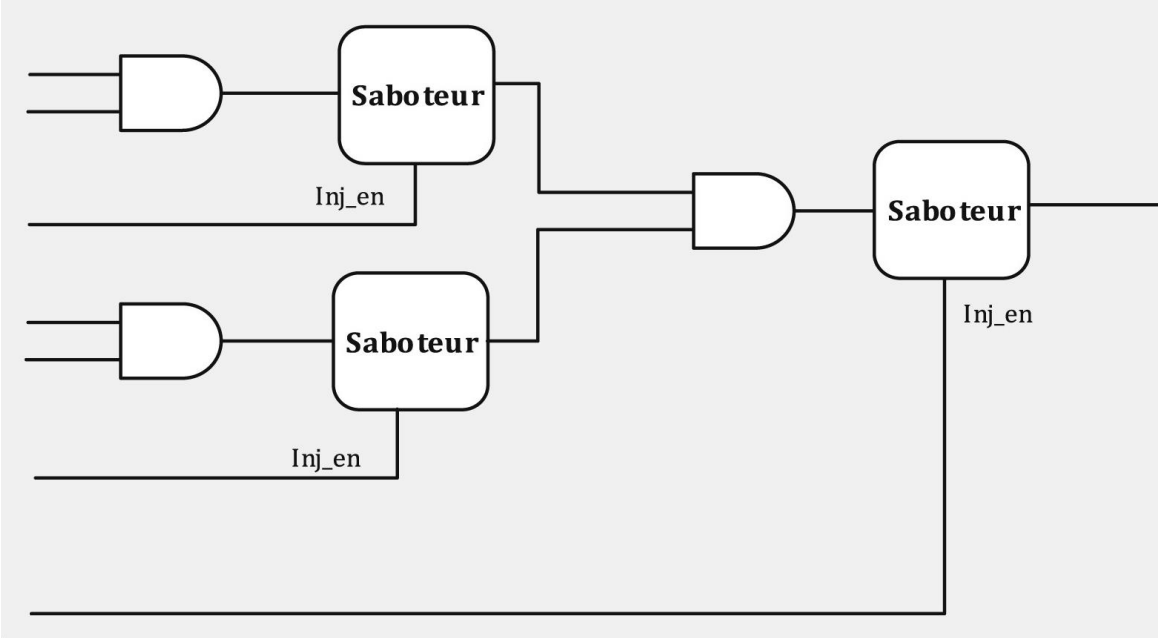
Αυτά που χρησιμοποιούν τις ενσωματωμένες εντολές ενός προσομοιωτή όπως το "force", για να εισάγουν σφάλματα στην προσομοίωση του σχεδίου, όχι στην περιγραφή υλικού του ίδιου του σχεδίου.

Για παράδειγμα, ορισμένα εργαλεία που βασίζονται σε αυτές τις τεχνικές είναι τα: SST, MEFISTO και VERIFY [20].

### 2.5.4. Έγχυση σφάλματος με βάση την εξομοίωση

Η έγχυση σφάλματος που βασίζεται σε εξομοίωση συνίσταται σε έναν μηχανισμό που υλοποιεί το υπό δοκιμή σύστημα σε ένα FPGA. Για αυτές τις πλατφόρμες, η πλακέτα ανάπτυξης συνδέεται με έναν κεντρικό υπολογιστή που χρησιμοποιείται για τον καθορισμό της μεθόδου εισαγωγής των σφαλμάτων, για τον έλεγχο των πειραμάτων της εισαγωγής σφαλμάτων και για την εμφάνιση των αποτελεσμάτων.

Μερικά παραδείγματα συστημάτων έγχυσης σφαλμάτων που βασίζονται σε εξομοίωση για είναι τα FT-UNSHADES [21], FLIPPER [22], SPFFI [23] και XRTC [24].




Σχήμα 2.8 - Προσθήκη μονάδων "σαμποτέρ" στις εισόδους/εξόδους των πυλών ενός κυκλώματος [11]

Πίνακας 2.1 - Προτερήματα / Μειονεκτήματα των διάφορων τεχνικών για fault injection

| Techniques | Advantages | Disadvantages |
|---|---|---|
| Hardware-based | • Υπάρχει πρόσβαση σε περιοχές απρόσιτες από τις άλλες μεθόδους<br>• Υψηλή ανάλυση στον χρόνο, για τα εναύσματα και για την παρακολούθηση των αποτελεσμάτων<br>• Κατάλληλο για low-level μοντέλα σφαλμάτων<br>• Τα πειράματα είναι γρήγορα | • Ενέχει κίνδυνο για πιθανή πραγματική ζημιά στο hardware.<br>• Στα πολύπλοκα κυκλώματα, είναι δύσκολο να γίνει έγχυση σφαλμάτων λόγω της μεγάλης πυκνότητας των στοιχείων του κυκλώματος<br>• Χαμηλή φορητότητα<br>• Περιορισμένος αριθμός των injection points και των injection faults. |
| Software-based | • Μπορεί να γίνει χειρισμός από εφαρμογές και από λειτουργικά συστήματα<br>• Τα πειράματα μπορούν να εκτελεστούν σε σχεδόν πραγματικό χρόνο<br>• Δεν χρειάζεται ειδικό hardware. | • Περιορισμένος αριθμός των injection points και των injection faults.<br>• Αδυναμία εισαγωγής faults σε σημεία που δεν είναι προσβάσιμα από το software<br>• Περιορισμένη παρακολούθηση των αποτελεσμάτων<br>• Πολύ δύσκολη η μοντελοποίηση permanent faults. |
| Simulation-based | • Πλήρης έλεγχος των fault models<br>• Πλήρης έλεγχος των μηχανισμών έγχυσης<br>• Χαμηλό κόστος καθώς δεν χρειάζεται ειδικό hardware | • Μεγάλη προσπάθεια για την ανάπτυξη του συστήματος<br>• Χρονοβόρα διαδικασία<br>• Το μοντέλο μπορεί να μην περιέχει faults τα οποία πιθανόν να υπάρχουν στο πραγματικό hardware. |
| Emulation-based | • Οι χρόνοι έγχυσης είναι στιγμιαίοι<br>• Ο χρόνος των πειραμάτων μπορεί να μειωθεί, καθώς μπορεί να γίνει στοχευμένη έγχυση σφαλμάτων. | • Το Verilog αρχείο πρέπει να είναι βελτιστοποιημένο έτσι ώστε να είναι εύκολη η προσθήκη των απαραίτητων μηχανισμών<br>• Το κόστος ενός γενικού hardware (ενός FPGA board) |

## 2.6 Fault simulation Techniques on FPGAs (fault emulators on FPGAs)

Οι συσκευές FPGA επιτρέπουν υψηλή ενοποίηση και επεκτασιμότητα στη λειτουργικότητα των μονάδων επικοινωνίας. Για παράδειγμα, νέοι αλγόριθμοι μπορούν να ενσωματωθούν με αυτές τις συσκευές χωρίς την απαίτηση αλλαγής των τσιπ επάνω στο board.

Αυτή η εργασία επικεντρώνεται στην δημιουργία μιας πλατφόρμας έγχυσης σφαλμάτων που βασίζεται σε έγχυση σφαλμάτων υλικού στις εισόδους και στις εξόδους των λογικών πυλών (FPGAs) που βασίζονται ελεγκτή intellectual property (IP) του Xilinx soft error mitigation (SEM).

Υπάρχουσες FPGA Fault-Injections τεχνικές, είναι οι ακόλουθες:

### i) FT-UNSHADES

Το σύστημα έγχυσης σφαλμάτων FT-UNSHADES [21], είναι μια πλατφόρμα βασισμένη στο FPGA. Βασίζεται σε μια σουίτα εργαλείων για τον Υπολογιστή και σε μια αποκλειστική πλατφόρμα υλικού που βασίζεται σε ένα Xilinx FPGA της οικογένειας Virtex-II. Στο σύστημα FT-UNSHADES, τα σφάλματα (SEU) εγχέονται ως bit-flips σε έναν ή περισσότερους καταχωρητές στο design. Πακέτα με bit διαμόρφωσης που περιέχουν την πραγματική τιμή ενός καταχωρητή, φορτώνονται από την πλατφόρμα υλικού, υποβάλλονται σε επεξεργασία και μεταφορτώνονται με νέες τιμές. Το λογισμικό FT-UNSHADES τροφοδοτείται με το αρχείο εκχώρησης bit που λαμβάνεται από το design του Xilinx. Σε αυτό το αρχείο αναφέρεται η φυσική διεύθυνση του καταχωρητή εντός του FPGA και συσχετίζεται με το λογικό του όνομα, που προκύπτει από τη διαδικασία σύνθεσης υψηλού επιπέδου στο Vivado.

### ii) FLIPPER και FLIPPER2

Τα συστήματα FLIPPER και FLIPPER2 [22], εισάγουν σφάλματα bit-flip στην μνήμη του FPGA όπου είναι αποθηκευμένοι οι καταχωρητές, μέσω μερικής επαναδιαμόρφωσης. Το σύστημα αποτελείται από μια πλατφόρμα υλικού (Xilinx XQR2V6000) και ένα λογισμικό που τρέχει σε υπολογιστή. Η συσκευή DUT είναι μια συσκευή FPGA της Xilinx, ενώ τα test vectors και οι τιμές αναφοράς για τη λειτουργική δοκιμή, εισάγονται από την εφαρμογή λογισμικού του Υπολογιστή, από έναν εξωτερικό προσομοιωτή HDL.

### 3) SPFFI

Το σύστημα SPFFI [23], αποτελείτε μόνο από το FGPA στο οποίο τρέχει το DUT, και το κατάλληλο λογισμικό στον Υπολογιστή. Το λογισμικό αυτό, είναι υπεύθυνο για την έγχυση των σφαλμάτων και την συλλογή των αποτελεσμάτων. Τα σφάλματα εισάγονται βασισμένα σε παραμέτρους που δίνει ο χρήστης. Υπάρχει επίσης ένας bitstream parser, ο οποίος αναλύει το bitstream πριν προγραμματιστεί

στο FPGA, και εξάγει σχετικές πληροφορίες για τον σχεδιασμό και την αρχιτεκτονική του DUT.

#### 4) XRTC

Το σύστημα XRTC [24] αναπτύχθηκε για την αξιολόγηση επαναδιαμορφώσιμων συσκευών σε περιβάλλοντα ακτινοβολίας, λόγω του αυξανόμενου ενδιαφέροντος για τη χρήση FPGAs για αεροδιαστημικές εφαρμογές. Το σύστημα αποτελείτε από ένα XRTC board, ένα FPGA στο οποίο τρέχει το DUT, και μία non-volatile μνήμη, η οποία περιέχει το bitstream από το DUT, αλλά και τα mask files του DUT

# ΚΕΦΑΛΑΙΟ 3

# ΣΥΣΤΗΜΑ ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΗΣ ΕΓΧΥΣΗΣ ΣΦΑΛΜΑΤΩΝ ΜΟΝΙΜΗΣ ΤΙΜΗΣ ΣΕ RTL ΣΧΕΔΙΑΣΜΟ ΚΑΤΑ ΤΗΝ ΕΞΟΜΟΙΩΣΗ ΤΟΥ ΣΕ XILINX FPGA.



Στο κεφάλαιο αυτό παρουσιάζεται το προτεινόμενο σύστημα αυτοματοποιημένης έγχυσης σφαλμάτων μόνιμης τιμής σε RTL σχεδιασμούς το οποίο βασίζεται στο FLIPPER με την προσθήκη ελεγκτή ειδικού σκοπού, που σχεδιάστηκε, ο οποίος επιτρέπει την εκτέλεση της διαδικασίας της εξομοίωσης σφαλμάτων μέσω λογισμικού. Αρχικά, παρουσιάζεται η βασική ιδέα του προτεινόμενου συστήματος. Έπειτα, παρουσιάζονται τεχνικές λεπτομέρειες για την υλοποίηση του υλικού (hardware) του προτεινόμενου συστήματος και ακολουθούν λεπτομέρειες για το λογισμικό που οδηγεί το υλικό του συστήματος.

## 3.1 Η πλατφόρμα FPGA Xilinx Zynq-7000 SoC ZC702

Η πλατφόρμα FPGA της Xilinx Zynq-7000 SoC ZC702, περιλαμβάνει όλα τα βασικά στοιχεία υλικού, τα εργαλεία σχεδίασης και τα IP blocks, που την καθιστούν μία πλήρη embedded πλατφόρμα επεξεργασίας.

Τα πιο βασικά χαρακτηριστικά της, είναι τα εξής:

- Είναι βελτιστοποιημένη για γρήγορο prototyping σε embedded applications.
- Μνήμη 1GB DDR3
- Δυνατότητα σύνδεσης με χρήση USB OTG, UART, IIC, CAN Bus

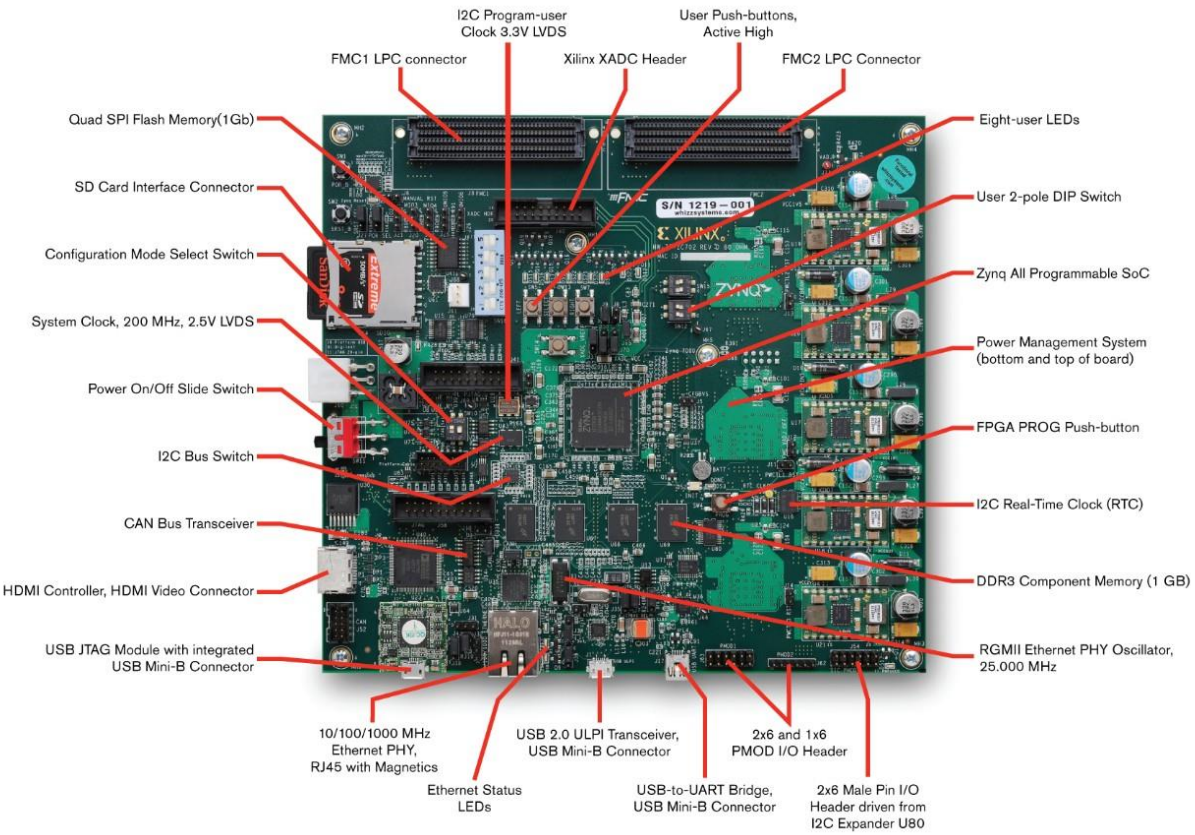
- Υποστήριξη embedded processing με χρήση του Dual ARM Cortex-A9 core processor
- Υποστήριξη δικτυακών εφαρμογών με 10/100/1000 Mbps Ethernet (GMII, RGMII and SGMII)
- Υλοποίηση Video display εφαρμογών, με το HDMI out
- Επέκταση των θυρών I/O με χρήση του FPGA Mezzanine Card (FMC) interface

Με αυτή την πλατφόρμα είναι πολύ εύκολο να σχεδιαστούν embedded applications, με την χρήστη σχεδιαστικού εργαλείου που παρέχει η Xilinx, το Vivado Design suite.

Σε ότι αφορά το υλικό, η υλοποίηση έχει γίνει με την χρήση ενός Ηλεκτρονικού Υπολογιστή (PC) και την ανωτέρω πλατφόρμα (Zynq-7000), τα οποία είναι συνδεδεμένα με μία σειριακή σύνδεση μέσω του USB-UART Interface, και με χρήση του USB-JTAG για την επικοινωνία με τον microcontroller.

Σε ότι αφορά το λογισμικό, χρησιμοποιούνται τα εξής προγράμματα:

- Vivado 2022.1, για την σύνταξη της verilog, την δημιουργία των αντίστοιχων netlist, την δημιουργία IP από την fault_injection_ready netlist και για την δημιουργία του bitstream για τον προγραμματισμό του FPGA.
- Xilinx SDK 2022.1, για την ανάπτυξη εφαρμογής στην γλώσσα C, μέσα από την οποία περνάμε τις αρχικές εισόδους στα υπό δοκιμή κυκλώματα, και για τον έλεγχο των drivers του παραγόμενου IP block από το Vivado.
- Python, για την μετατροπή του αρχικού netlist, στο fault_injection_ready netlist.




Σχήμα 3.1 - Η πλατφόρμα FPGA Xilinx Zynq-7000 SoC ZC702

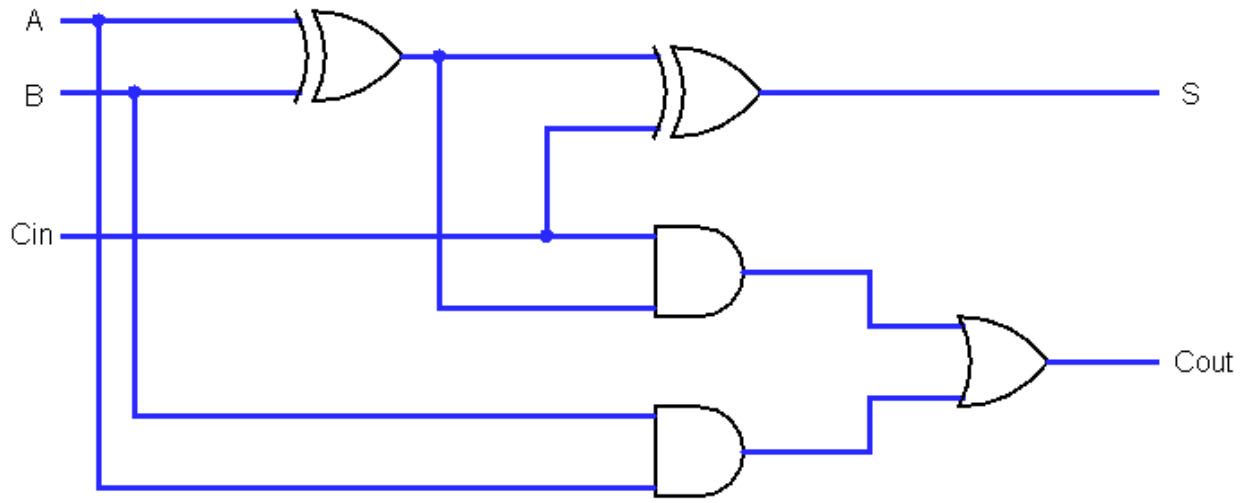
Σε αυτό το σημείο, παρουσιάζεται ένας οδηγός βήμα προς βήμα που περιγράφει το εργαλείο έγχυσης σφαλμάτων με βάση την εξομοίωση. Η ρύθμιση βασίζεται στην τεχνολογία της Xilinx. Οι κύριες ενότητες που απαρτίζουν το σύστημα είναι οι ακόλουθες:

- Ο ελεγκτής LogiCORE IP Soft Error Mitigation (SEM).
- Μια πλακέτα FPGA Zynq-7000S (με ARM επεξεργαστή) της Xilinx.
- Ένα design under test (DUT) που υλοποιήθηκε με verilog για το FPGA, προκειμένου να μετρηθεί η αξιοπιστία του σε περίπτωση σφαλμάτων.
- Μια μονάδα (UART) για την υλοποίηση της επικοινωνίας μεταξύ FPGA και Υπολογιστή.

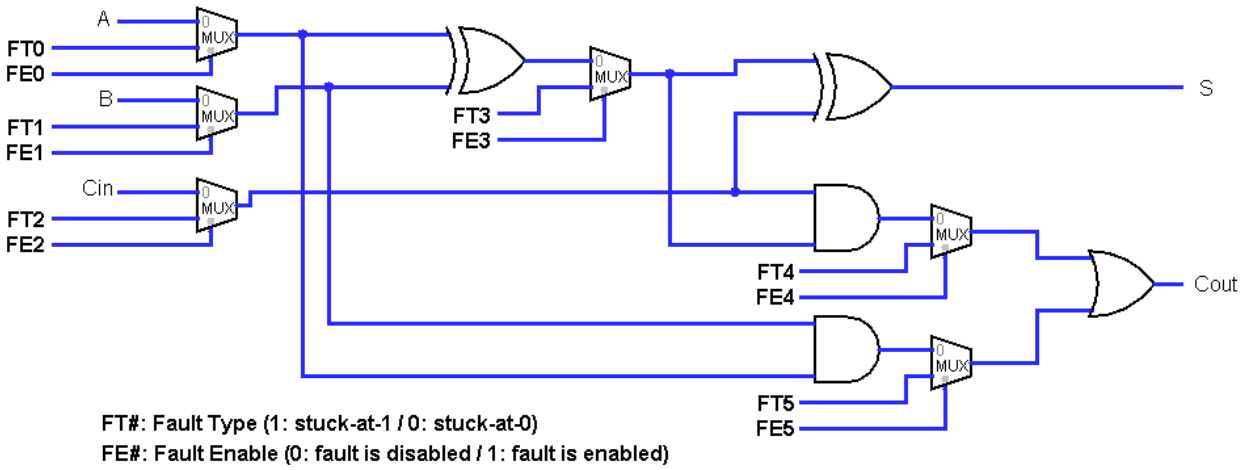
Μια επισκόπηση ολόκληρου του συστήματος εξηγείται στη συνέχεια.

## 3.2 Design under test (DUT) με Verilog

Το σύστημα που θα υποβληθεί σε test, θα πρέπει να περιγραφεί με γλώσσα Verilog, και θα δωθεί σαν input στο Vivad. Στην συνέχεια, θα πρέπει να προστεθούν σε κάθε net (wire) του συστήματος, ένας 2:1 Multiplexer, με τον οποίο θα γίνεται η επιλογή για το αν θα εισαχθεί σφάλμα ή όχι, και τι σφάλμα θα εισαχθεί (stuck-at-0, stuck-at-1).




Σχήμα 3.2 - Ένα κύκλωμα παράδειγμα ενός Full Adder πριν την προσθήκη των 2:1 Multiplexors



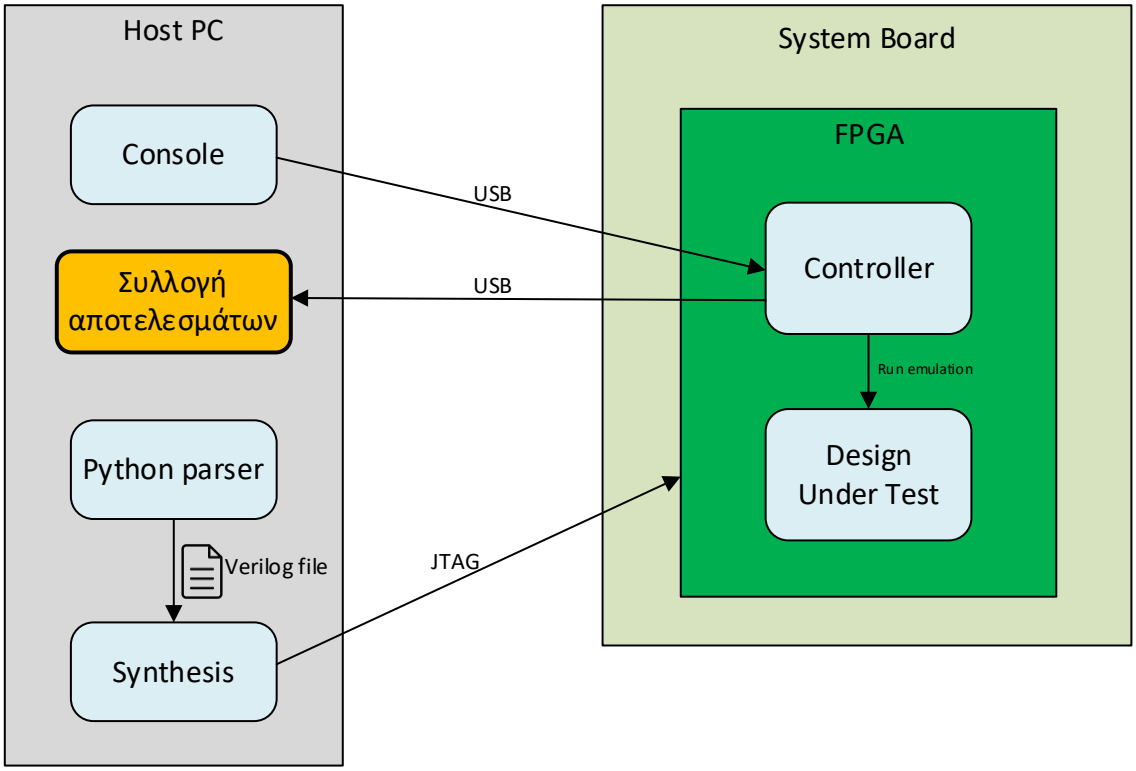

Σχήμα 3.3 - Το κύκλωμα Full Adder, με την προσθήκη των απαραίτητων 2:1 Multiplexors σε κάθε net

Για την αυτόματη και γρήγορη προσθήκη όλων των 2:1 Multiplexors (έναν σε κάθε net), γίνεται πρώτα η εξαγωγή του verilog αρχείου του DUT, σε ένα netlist

αρχείο (παράρτημα Γ2), έτσι ώστε να είναι εμφανή όλα τα wires του κυκλώματος. Έπειτα γίνεται εισαγωγή των multiplexors με χρήση του παρακάτω κώδικα python (παράρτημα Ε), ο οποίος ανιχνεύει όλα τα wires, και πραγματοποιεί τις παρακάτω προσθήκες:

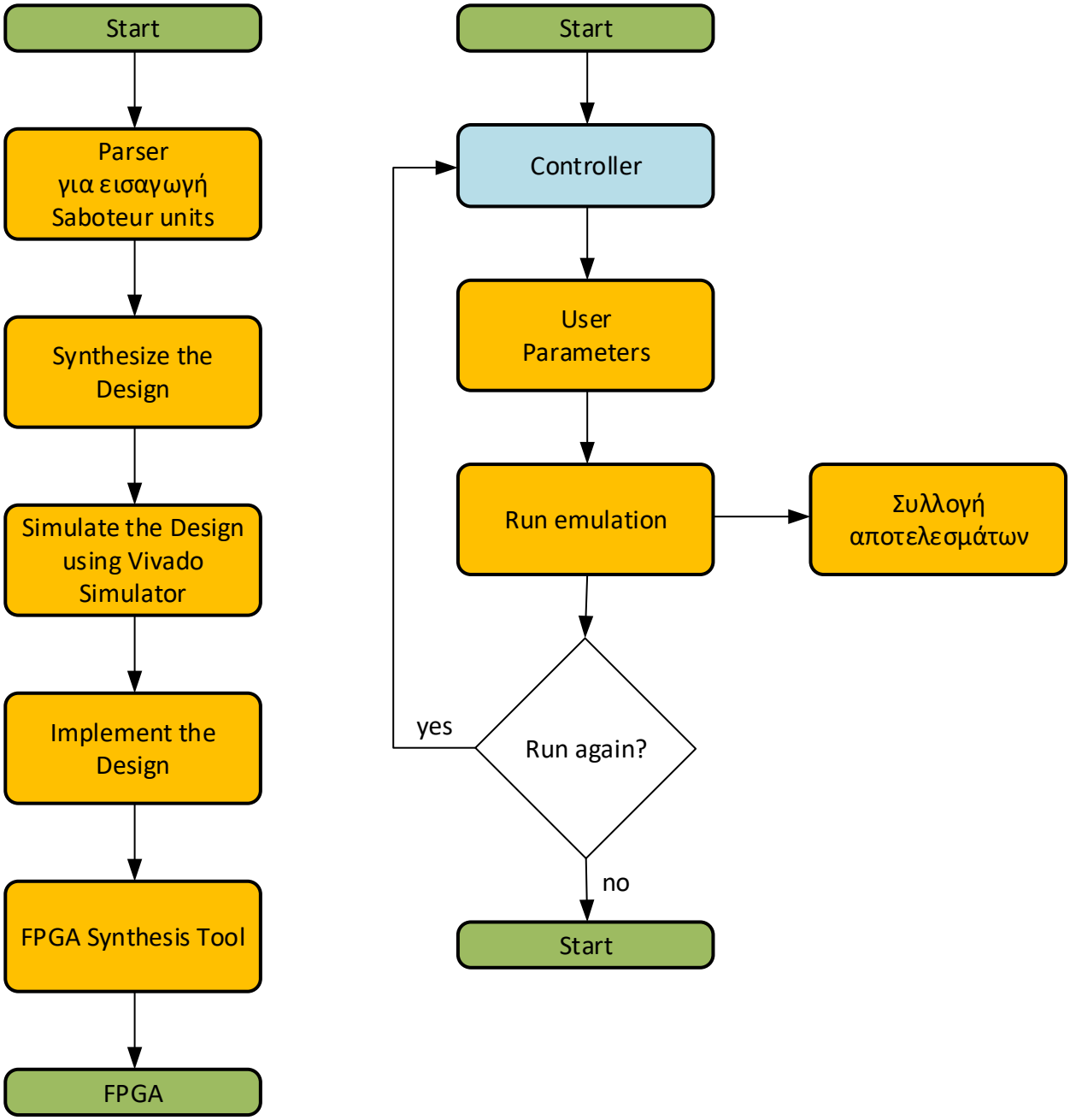
- προσθήκη αντίστοιχων πορτών στο port list του module.
- προσθήκη των αντίστοιχων εγγραφών input για κάθε είσοδο του κάθε multiplexer.
- προσθήκη του ίδιου του multiplexer υπό την μορφή ternary operator της verilog (π.χ. assign A_WITHFAULT= (FE0) ? FT0 : A ;)

Η αρχιτεκτονική όλου του συστήματος φαίνεται στο παρακάτω σχήμα.




Σχήμα 3.4 - Αρχιτεκτονική του συστήματος

Το data flow diagram του συστήματος, φαίνεται στο παρακάτω σχήμα.




Σχήμα 3.5 – Data flow diagram του συστήματος

Το πως φαίνεται το αρχείο verilog πριν και μετά από την προσθήκη των Multiplexors από την python, φαίνεται παρακάτω:


Σχήμα 3.6 - Το αρχείο netlist πριν και μετά από την προσθήκη των Multiplexors

## 3.3 Η διαδικασία της εξομοίωσης σφαλμάτων στο προτεινόμενο σύστημα

Στη συνέχεια, θα περιγραφούν τα διάφορα βήματα που απαιτούνται για την εφαρμογή της εξομοίωσης έγχυσης σφαλμάτων με βάση την σχεδίαση που ορίστηκε στο προηγούμενο κεφάλαιο.

Αρχικά, δημιουργείται ένα νέο έργο RTL για το Vivado, το οποίο περιλαμβάνει τα αρχεία HDL που αναφέρονται τόσο στη σχεδίαση για τα πειράματα, όσο και στην μονάδα UART. Η συμπερίληψη στο project του IP SEM θα παρουσιαστεί λεπτομερώς αργότερα.

Επίσης, σε αυτό το σημείο πρέπει να επιλεγεί το μοντέλο του board. Για τον συγκεκριμένο σχεδιασμό, το board είναι ένα Zynq-7000 SoC FPGA. Μόλις εφαρμοστεί το DUT, και πριν συνεχιστεί η υπόλοιπη διαδικασία, μια καλή πρακτική είναι η προσομοίωση και ο έλεγχος του σχεδιασμού, χωρίς την έγχυση σφαλμάτων.

Μετά την επαλήθευση του αρχικού κυκλώματος χωρίς σφάλματα, ο IP SEM θα πρέπει να δημιουργηθεί για να ενεργοποιηθεί και να ελεγχθεί η διαδικασία έγχυσης σφάλματος. Ο IP SEM βρίσκεται στον κατάλογο IP του Vivado, ωστόσο το IP block με τα παραγόμενο από την Python verilog αρχείο, δεν υπάρχει στον κατάλογο IP του Vivado, και θα πρέπει να δημιουργηθεί και να εισαχθεί. Η μέγιστη συχνότητα ρολογιού του ανταποκρίνεται στη συχνότητα του board (για το Zynq-7000) είναι 766 MHz. Κάτω από αυτή τη συχνότητα, ο IP SEM λειτουργεί σωστά.

Στη συνέχεια, δημιουργούμε ένα νέο project Vivado, για να κάνουμε εξαγωγή των source αρχείων του SEM IP για την περαιτέρω ενσωμάτωσή τους στο κύριο project, προκειμένου να ολοκληρωθεί ο σχεδιασμός για το πείραμά μας.

Στη συνέχεια, τα source αρχεία πρέπει να αντιγραφούν στο αρχικό project. Αλλά πριν την εισαγωγή του κυρίως verilog αρχείου, θα πρέπει να προστεθεί σε κάθε net, ένας 2:1 Multiplexer με τον οποίο θα γίνεται επιλογή για το πότε και τι σφάλμα θα εισάγεται.

Σε αυτό το σημείο, όλα είναι έτοιμα για την εφαρμογή του σχεδιασμού στο FPGA. Για το σκοπό αυτό, θα πρέπει να δημιουργηθεί το bitstream, για να προγραμματιστεί στη συνέχεια το FPGA.

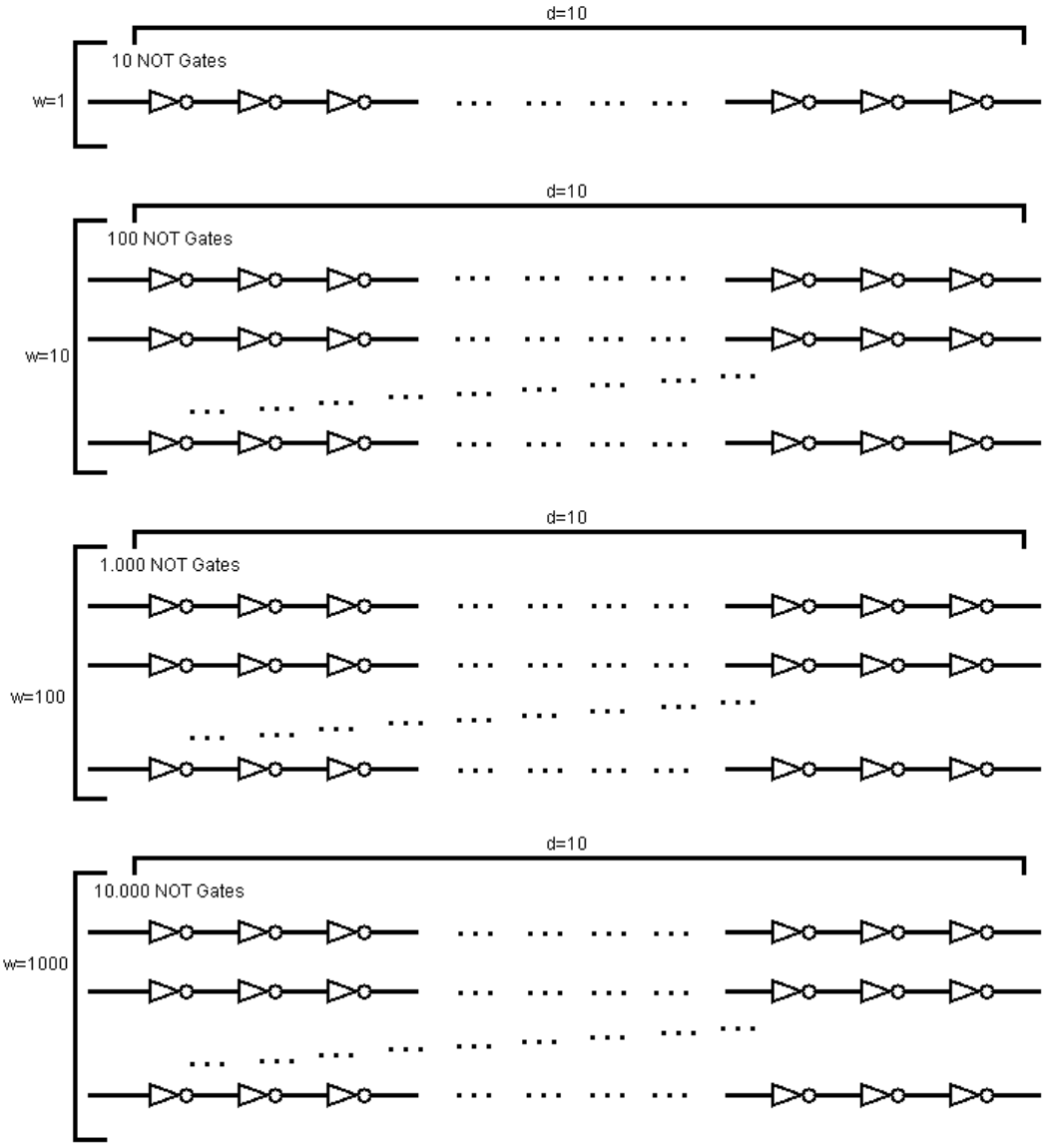
Μόλις το board φορτωθεί με το bitstream που προέκυψε, στο board ανάβουν δύο πράσινα LED. Το πρώτο είναι το LED "FPGA ready" και το δεύτερο έχει σχέση με την κατάσταση του IP SEM που δηλώνει ότι το IP SEM βρίσκεται σε λειτουργία παρατήρησης. Επομένως, υποδεικνύει ότι το SEM IP είναι έτοιμο να λειτουργήσει.

# ΚΕΦΑΛΑΙΟ 4

## ΠΕΙΡΑΜΑΤΑ

Στο κεφάλαιο αυτό θα δείξουμε πως παράγουμε αυτόματα συνθετικούς σχεδιασμούς που αποτελούνται από πολλαπλές σειρές λογικών αντιστροφέων τους οποίους θα χρησιμοποιήσουμε για την αξιολόγηση του προτεινόμενου συστήματος εξομοίωσης σφαλμάτων που βασίζεται σε FPGA. Έπειτα, με τη χρήση αυτών των συνθετικών σχεδιασμών θα αξιολογήσουμε την απόδοση του συστήματός που αναπτύχθηκε ως προς το ρυθμό εξομοίωσης σφαλμάτων (faults emulation rate = faults emulations number per second) σε σύγκριση με τον ρυθμό προσομοίωσης σφαλμάτων μιας τεχνικής που βασίζεται μόνο σε λογισμικό.

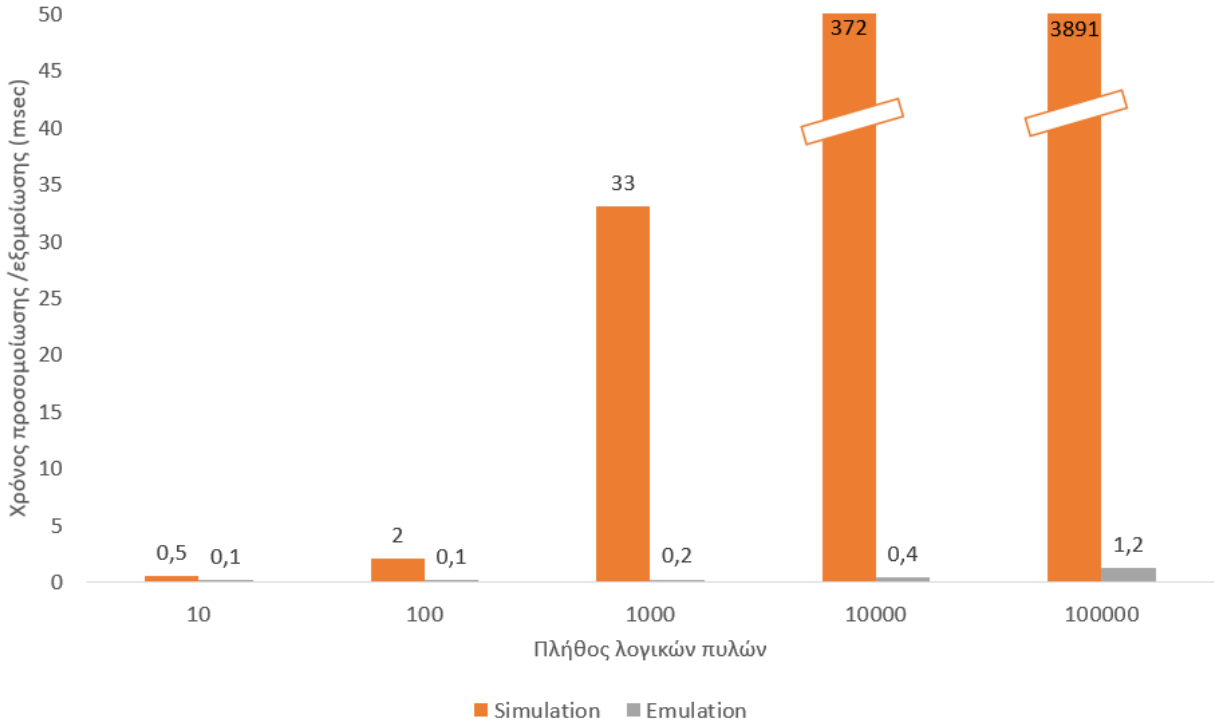
Τα πειράματα έγιναν με συστοιχίες αντιστροφέων (cascaded inverters), με διάφορα πλήθη πυλών. Έγινε με 10 αντιστροφείς, 100 αντιστροφείς, 1.000 αντιστροφείς και 10.000 αντιστροφείς. Σε κάθε μία από τις παραπάνω περιπτώσεις, έγινε προσομοίωση/εξομοίωση για πολλαπλά faults.




Σχήμα 4.1 - Συστοιχίες αντιστροφέων, με τους οποίους έγινε εκτέλεση των προσομοιώσεων/εξομοιώσεων

Για κάθε μία από τις παραπάνω συστοιχίες, έγινε έλεγχος τόσο με προσομοίωση (με χρήση λογισμικού), όσο και με την προτεινόμενη λύση, δηλαδή με την εξομοίωση με χρήση FPGA. Το d (depth) αναφέρεται στο πόσες λογικές πύλες έχει η κάθε σειρά από λογικές πύλες, ενώ το w (width) αναφέρεται στο πόσες σειρές έχει όλη η συστοιχία.

Στον πίνακα 2, και στο διάγραμμα στο Σχήμα 4.2, φαίνονται οι αντίστοιχοι χρόνοι για d=10, και για w={1,10,100,1000,10000}.




Σχήμα 4.2 - Διάγραμμα στο οποίο φαίνεται ο χρόνος προσομοίωσης/εξομοίωσης ανά πλήθος λογικών πυλών με συστοιχία inverters

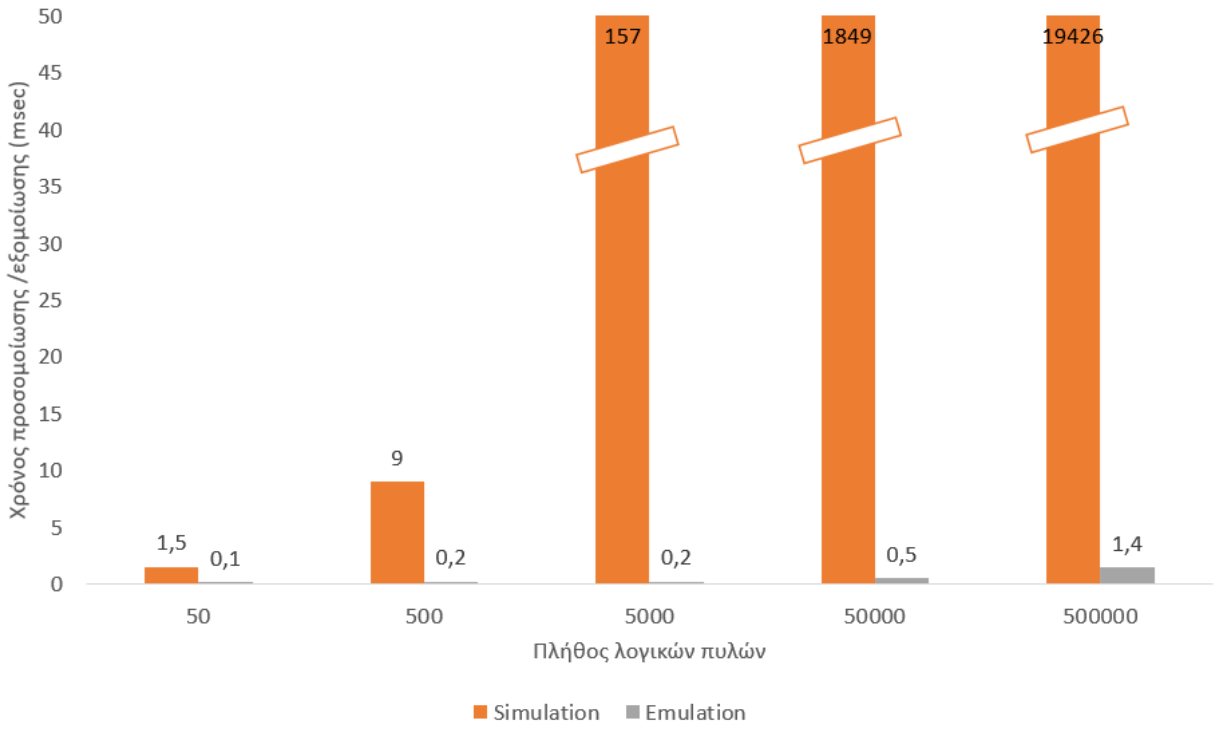
Πίνακας 4.1 - Χρόνος προσομοίωσης/εξομοίωσης ανά πλήθος λογικών πυλών με συστοιχία inverters

| width (w) | depth (d) | Gates | Simulation (msec) | Emulation (msec) |
|---|---|---|---|---|
| 1 | 10 | 10 | 0,5 | 0,1 |
| 10 | 10 | 100 | 2 | 0,1 |
| 100 | 10 | 1000 | 33 | 0,2 |
| 1000 | 10 | 10000 | 372 | 0,4 |
| 10000 | 10 | 100000 | 3891 | 1,2 |

Από το παραπάνω διάγραμμα, φαίνεται ότι ο χρόνος στην περίπτωση της προσομοίωσης επηρεάζεται από το πλήθος των πυλών, ενώ στην περίπτωση της εξομοίωσης, ο χρόνος εκτέλεσης δεν επηρεάζεται. Αξίζει να σημειωθεί ότι η μικρή αύξηση στον χρόνο εκτέλεσης της εξομοίωσης, οφείλεται στο γεγονός ότι ο Controller χρησιμοποιεί καταχωρητές 32bit για την είσοδο στοιχείων από τον χρήστη για τις τιμές των εισόδων, και των τύπων σφαλμάτων σε κάθε net, οπότε χρειάζεται περισσότερους καταχωρητές ανάλογα με το πλήθος των πυλών και των nets που υπάρχουν στο υπό δοκιμή κύκλωμα. Σε καμία περίπτωση όμως αυτή η

αύξηση του χρόνου δεν είναι εκθετική (όπως στην περίπτωση της προσομοίωσης), αλλά είναι γραμμική.

Στον πίνακα 3, και στο διάγραμμα στο Σχήμα 4.3, φαίνονται οι αντίστοιχοι χρόνοι για d=50, και για w={1,10,100,1000,10000}.

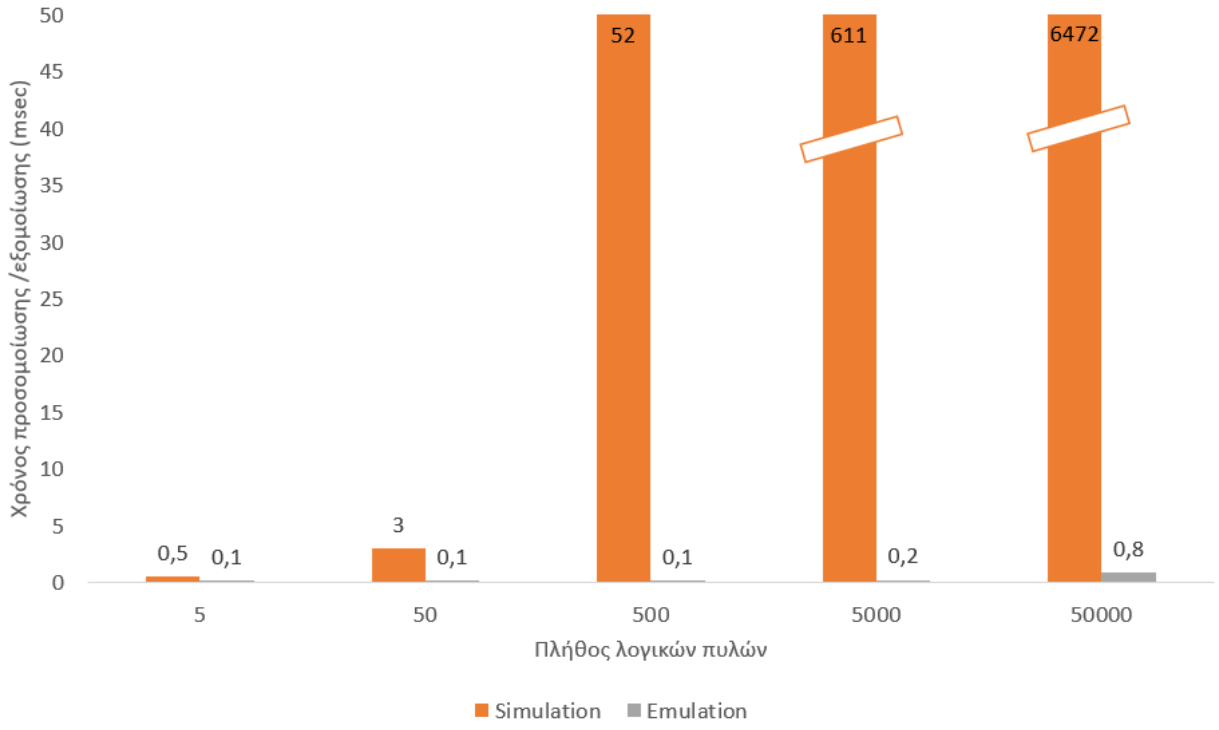



Σχήμα 4.3 - Διάγραμμα στο οποίο φαίνεται ο χρόνος προσομοίωσης/εξομοίωσης ανά πλήθος λογικών πυλών με συστοιχία inverters

Πίνακας 4.2 - Χρόνος προσομοίωσης/εξομοίωσης ανά πλήθος λογικών πυλών με συστοιχία inverters

| width (w) | depth (d) | Gates | Simulation (msec) | Emulation (msec) |
|---|---|---|---|---|
| 1 | 50 | 50 | 1,5 | 0,1 |
| 10 | 50 | 500 | 9 | 0,2 |
| 100 | 50 | 5000 | 157 | 0,2 |
| 1000 | 50 | 50000 | 1849 | 0,5 |
| 10000 | 50 | 500000 | 19426 | 1,4 |

Στον πίνακα 4, και στο διάγραμμα στο Σχήμα 4.4, φαίνονται οι αντίστοιχοι χρόνοι για την προσομοίωση/εξομοίωση ενός κυκλώματος με συστοιχία από full-adders, για 1, 10, 100 και 1000 full-adders.



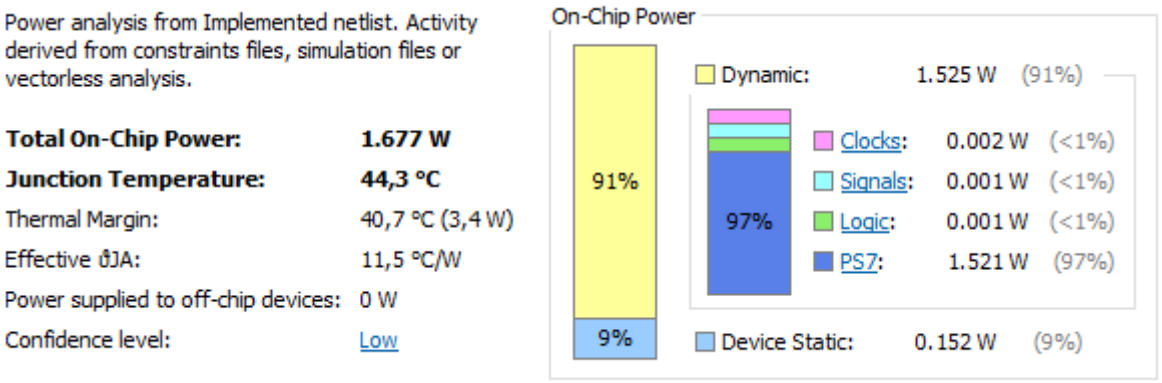

Σχήμα 4.4 - Διάγραμμα στο οποίο φαίνεται ο χρόνος προσομοίωσης/εξομοίωσης ανά πλήθος λογικών πυλών σε κύκλωμα με full-adders

Πίνακας 4.3 - Χρόνος προσομοίωσης/εξομοίωσης ανά πλήθος λογικών πυλών σε κύκλωμα με full-adders

| Number of Full-Adders | Number of logic Gates | Simulation (msec) | Emulation (msec) |
|---|---|---|---|
| 1 | 5 | 0,5 | 0,1 |
| 10 | 50 | 3 | 0,1 |
| 100 | 500 | 52 | 0,1 |
| 1000 | 5000 | 611 | 0,2 |
| 10000 | 50000 | 6472 | 0,8 |

Το εργαλείο της Xilinx το Vivado, παρέχει διάφορες αναφορές για κάθε design που γίνεται implement στο FPGA, μέσα από τις οποίες μπορούμε να εξάγουμε διάφορα συμπεράσματα για την ενέργεια και για την ταχύτητα του κάθε design.

Όσον αφορά την ενέργεια, σε όλους τους σχεδιασμούς η κατανάλωση είναι κάτω από 0,001Watt. Το μεγαλύτερο ποσοστό της ενέργειας, καταναλώνεται από τον πυρήνα Processing System 7 του FPGA, ο οποίος έχει προστεθεί στον σχεδιασμό καθώς είναι το σύστημα επεξεργασίας του Zynq-7000 SoC και είναι υπεύθυνο για την λογική που έχει σχεδιαστεί και υπάρχει στο board.




Σχήμα 4.5 - Η κατανάλωση ενέργειας γίνεται από το Processing System 7, ενώ η λογική έχει κατανάλωση ενέργειας αμελητέα μπροστά στο υπόλοιπο σύστημα.

Το signal rate σε όλους τους σχεδιασμούς, είναι περίπου ίδιο και δεν μεταβάλεται. Συγκεκριμένα είναι περίπου 1300 Mtr/s (millions of transitions per second). Το Signal rate καθορίζει το πλήθος σε εκατομμύρια των αλλαγών (high-to-low και low-to-high) ανά δευτερόλεπτο (Mtr/s) για το ζητούμενο κάθε φορά κύκλωμα. Η γενική εξίσωση για τον υπολογισμό του signal rate είναι:

Signal Rate (Mtr/s) = Clock Frequency (Mhz) * Effective Toggle Rate (%)

Το κόστος του σχεδιασμού σε λογικές πύλες, φαίνεται στον παρακάτω πίνακα:

Πίνακας 4.4 - Ποσοστό αύξησης του κόστους σε σχέση με τις λογικές πύλες

| | Πλήθος πυλών χωρίς fault-injection | Πλήθος πυλών με fault-injection | Ποσοστό αύξησης |
|---|---|---|---|
| Inverters | 1000 | 5000 | 400% |
| Full Adders | 5000 | 29000 | 480% |

# ΚΕΦΑΛΑΙΟ 5

## ΣΥΜΠΕΡΑΣΜΑΤΑ

Σε αυτή την εργασία, παρουσιάστηκε ένα σύστημα εξομοίωσης έγχυσης σφαλμάτων με χρήση FPGA υψηλής ταχύτητας μαζί με μια μεθοδολογία που μπορεί να ελέγξει την ευαισθησία ενός σχεδιασμού στα SEU. Το σύστημα αυτό είναι οικονομικά προσιτό και βασίζεται στην τεχνολογία Xilinx μαζί με τα απαιτούμενα εργαλεία Vivado και Vivado SDK. Το σύστημα αυτό μπορεί να χρησιμοποιηθεί για την αξιολόγηση της συμπεριφοράς των ηλεκτρονικών εξαρτημάτων που απαιτούνται να λειτουργούν κάτω από δύσκολες συνθήκες. Ο στόχος αυτής της εργασίας, είναι να βοηθήσει τους προγραμματιστές υλικού να δοκιμάσουν την αξιοπιστία ενός σχεδίου με απλό και ολοκληρωμένο τρόπο. Η συγκεκριμένη μέθοδος έγχυσης σφαλμάτων, μπορεί να χρησιμοποιηθεί στα πιο πρόσφατα FPGAs και μπορεί να προσαρμοστεί και για μελλοντικές κάρτες FGPA.

Επίσης, στην εργασία παρουσιάζονται τα αποτελέσματα πειραμάτων έγχυσης σφαλμάτων σε πολλαπλούς σχεδιασμούς και έγινε η σύγκριση σε ταχύτητα με την προσομοίωση της έγχυσης σφαλμάτων με χρήση λογισμικού.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] Nikhil Vichare, Ping Zhao, Diganta Das, Michael Pecht. 2006. Electronic Hardware Reliability, Digital Avionics Handbook, Third Edition.

[2] Hyejeong Hong, Jaeil Lim, Hyunyul Lim, and Sungho Kang. 2015. Lifetime Reliability Enhancement of Microprocessors: Mitigating the Impact of Negative Bias Temperature Instability.

[3] Wei Ye, Mohamed Baker Alawieh, Che-Lun Hsu, Yibo Lin & David Z. Pan. 2020. Dealing with Aging and Yield in Scaled Technologies

[4] Natasa Miskov-Zivanov, Diana Marculescu, 2010. Multiple Transient Faults in Combinational and Sequential Circuits: A Systematic Approach.

[5] Chang-Chih Chen and L. Milor. "Microprocessor Aging Analysis and Reliability Modeling Due to Back-End Wearout Mechanisms". In: Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 23.10 (2015), pp. 2065–2076. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2014.2357756.

[6] J. Mathew, R.A. Shafik, and D.K. Pradhan. Energy-Efficient Fault-Tolerant Systems. Springer New York, 2013. ISBN: 9781461441922

[7] H. M. Quinn, D. A. Black, W. H. Robinson and S. P. Buchner, Fault Simulation and Emulation Tools to Augment Radiation-Hardness Assurance Testing.

[8] Charles E. Stroud, Laung-Terng Wang, Yao-Wen Chang, 2009. Electronic Design Automation.

[9] Jiahua Fan, Zhifeng Zhang. Speeding up Fault Simulation using Parallel Fault Simulation. 2011.

[10] G.L. Smith, "Model for Delay Faults Based Upon Paths," Proceedings of the International Test Conference , Nov. 1985, pp. 342-349.

[11] Mohammad Eslami, Behnam Ghavami, Mohsen Raji, Ali Mahani. 2020. A survey on fault injection methods of digital integrated circuits.

[12] G. P. Saggese, N. J. Wang, Z. T. Kalbarczyk, S. J. Patel and R. K. Iyer, "An Experimental Study of Soft Errors in Microprocessors," in IEEE Micro, Vol. 25, No. 6, pp. 30-39, November 2005.

[13] S. Borkar, "Thousand Core Chips – A Technology Perspective," in Proc. of Design Automation Conference (DAC), pp. 746-749, June 2007.

[14] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi,, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic," in Proc. of International Conference on Dependable Systems and Networks, pp. 389-398, 2002.

[15] A. KleinOsowski, E. H. Cannon, P. Oldiges and L. Wissel, "Circuit design and modeling for soft errors," in IBM Journal of Research and Development, Vol. 52, No. 3, pp. 255-263, May 2008.

[16] M. Melgara. 1988. Fault Simulation Techniques — Theory and Practical Examples

[17] Madeira, Henrique & Zenha-Rela, Mário & Moreira, Francisco & Silva, João. 1994. RIFLE: A general purpose pin-level fault injector.

[18] Hsueh, Mei-Chen & Tsai, Timothy & Iyer, Ravishankar. 1997. Fault injection techniques and tools.

[19] Carreira, João & Madeira, Henrique & Silva, João. 1998. Xception: A Technique for the Experimental Evaluation of Dependability in Modern Computers.

[20] Sieh, V.; Tschache, O.; Balbach, F. 1997. VERIFY: Evaluation of reliability using VHDL-models with embedded fault descriptions

[21] Guzman-Miranda H., Tombs J.N., Aguirre M.A. 2008. FT-UNSHADES-uP: A platform for the analysis and optimal hardening of embedded systems in radiation environments

[22] Alderighi M., Casini F., D'Angelo S., Mancini M., Codinachs D.M., Pastore S., Poivey C., Sechi G.R., Sorrenti G., Weigand R. 2010. Experimental Validation of Fault Injection Analyses by the FLIPPER Tool

[23] Cieslewski G., George A. 2021. SPFFI: Simple, Portable FPGA Fault Injector FPGA Fault Injector

[24] Harward N.A. [2021] Measuring Soft Error Sensitivity of FPGA Soft Processor Designs Using Fault Injection

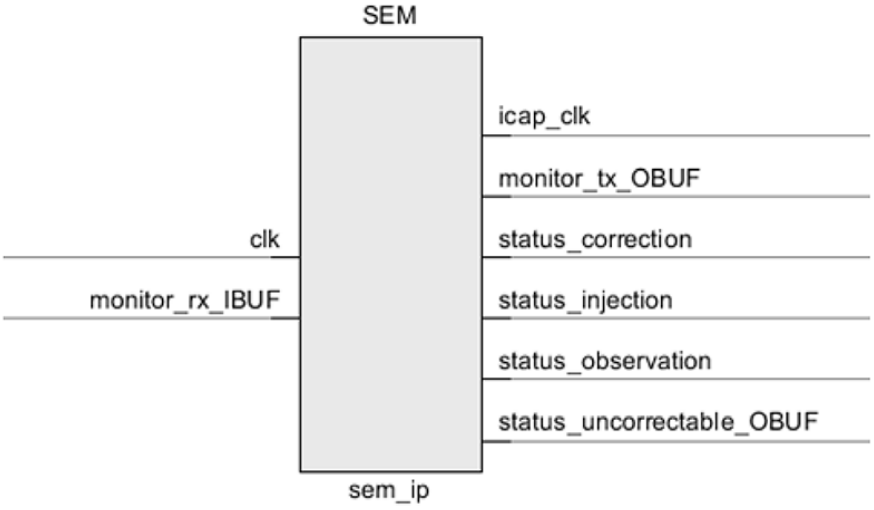
# ΠΑΡΑΡΤΗΜΑ Α

## Soft Error Mitigation (SEM) IP Controller

Ο SEM IP πυρήνας, είναι η μηχανή έγχυσης σφαλμάτων για τη διαδικασία εξομοίωσης. Αυτός ο πυρήνας IP που υποστηρίζεται από το Xilinx, είναι μια λύση για τον εντοπισμό και τη διόρθωση σφαλμάτων των FPGA Xilinx.

Αυτή η μονάδα μπορεί να διαβάσει τη μνήμη διαμόρφωσης για να αναζητήσει σφάλματα. Σε περίπτωση εντοπισμού οποιουδήποτε αλλαγμένου bit, αναστρέφεται από το SEM IP για να το διορθώσει. Αυτή η ικανότητα ανάγνωσης και εγγραφής στη μνήμη διαμόρφωσης χρησιμοποιείται για τη διαχείριση της έγχυσης σφάλματος με μη επεμβατικό τρόπο. Ο χρήστης που χρησιμοποιεί SEM IP μπορεί να εισάγει αναστροφές bit στη μνήμη διαμόρφωσης, δοκιμάζοντας την αξιοπιστία της λύσης που εφαρμόστηκε για ένα FPGA μετά την έγχυση των σφαλμάτων. Το IP SEM χρησιμοποιείται επειδή είναι το μόνο εργαλείο στα FPGA Xilinx που επιτρέπει τον έλεγχο της κατάστασης της μνήμης διαμόρφωσης των συσκευών για την αναζήτηση πιθανών σφαλμάτων.

Προκειμένου να προστεθεί ο πυρήνας SEM IP σε ένα project στο Vivado, πρέπει να δημιουργηθεί από τον κατάλογο IP του Vivado.




Σχήμα Α.1 - Soft Error Mitigation (SEM) IP Controller

Το σήμα εισόδου ρολογιού (clk) πρέπει να συνδεθεί στο global clock του σχεδιασμού.

Το monitor_rx_IBUF είναι ένα UART που σειριοποιεί τις πληροφορίες κατάστασης που δημιουργούνται από το IP SEM για σειριακή μετάδοση μεταξύ του υπολογιστή και του FPGA μέσω των γραμμών TX/RX.

Το RX λαμβάνει ως είσοδο τις εντολές προς το SEM IP, τις οποίες καλείται να εκτελέσει, όπως για παράδειγμα εγχύσεις σφαλμάτων.

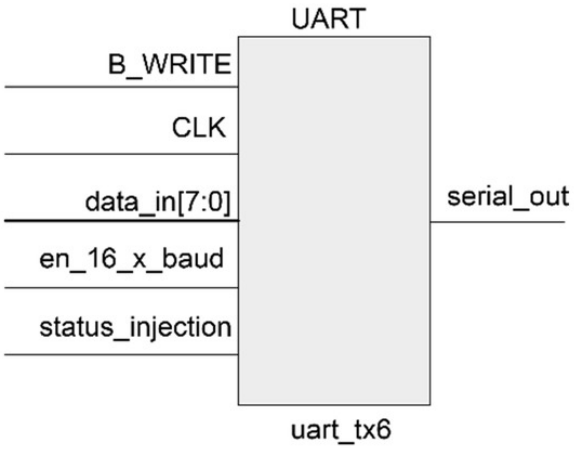
Το TX λαμβάνει αναφορές από το SEM IP, όπως για παράδειγμα οι νέες καταστάσεις που επιτυγχάνονται (Initialization, injection, error, idle κλπ).

# ΠΑΡΑΡΤΗΜΑ Β

# UART

Η μονάδα UART περιλαμβάνεται για τη διαχείριση των σειριακών επικοινωνιών μεταξύ του Υπολογιστή και του FPGA μέσω της γραμμής serial_out όπως φαίνεται στο Σχήμα Β.1.

Το UART επιτρέπει μια διαδικασία παρακολούθησης παρόμοια με αυτή που υποστηρίζεται από το IP SEM, αλλά σε αυτήν την περίπτωση, χρησιμοποιείται για αποστολή στον Υπολογιστή, μετά από κάθε fault injection.




Σχήμα Β.1 - Το pinout της μονάδας UART του FPGA

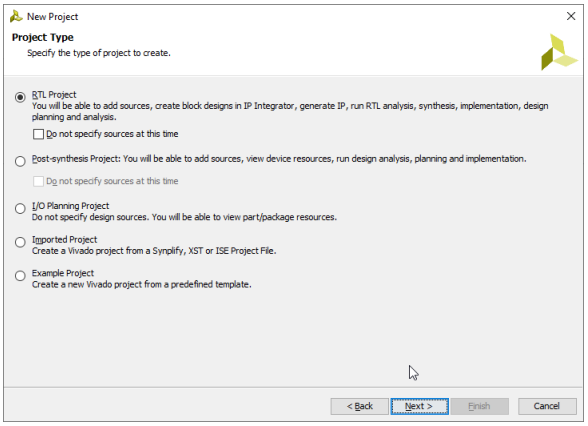
# ΠΑΡΑΡΤΗΜΑ Γ

## ΔΗΜΙΟΥΡΓΙΑ ΤΟΥ PROJECT ΣΤΟ VIVADO

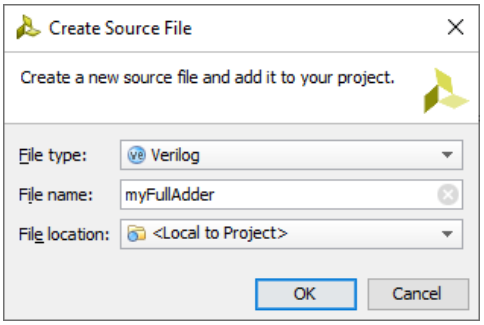


## Γ.1 Καθορισμός του σωστού board στο Vivado

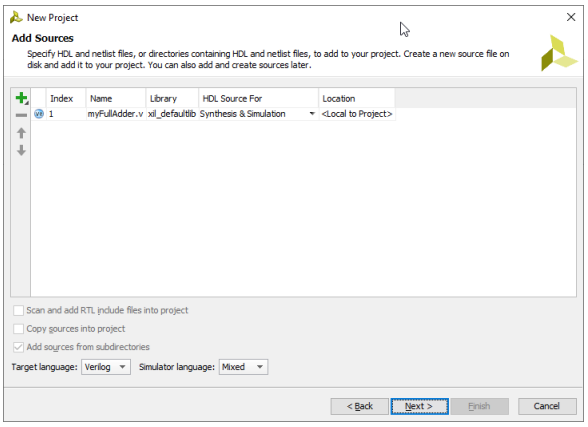
Στα επόμενα σχήματα, φαίνεται η διαδικασία για την δημιουργία Project στο Vivado, ο καθορισμός των αρχείων verilog που θα εξομοιωθούν, καθώς επίσης και πως γίνεται η επιλογή του board στο οποίο θα εξομοιωθεί το design.



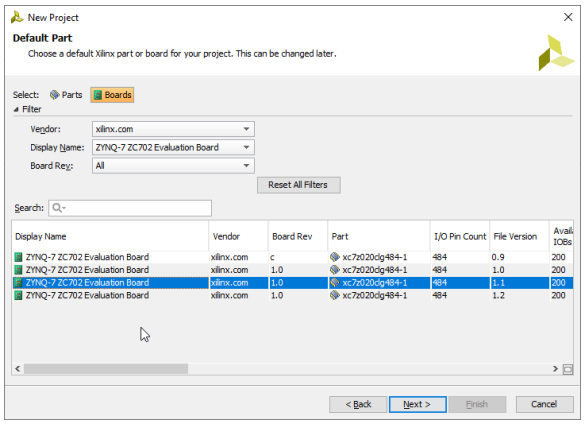

Σχήμα Γ.1 - Δημιουργία ενός νέου RTL Project



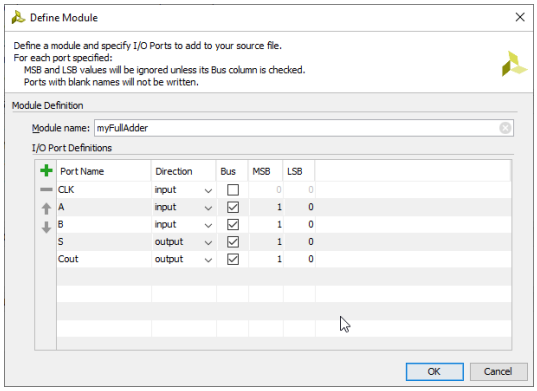

Σχήμα Γ.2 - Προσθήκη στο Project του Verilog αρχείου πριν την έγχυση σφαλμάτων



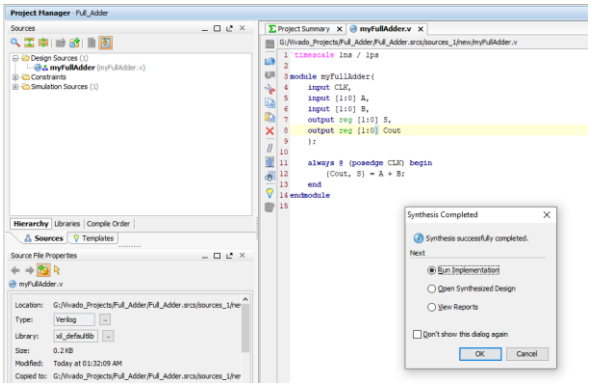

Σχήμα Γ.3 - Η λίστα με τα verilog αρχεία που θα υπάρχουν στο Project (μόνο ένα στην συγκεκριμένη περίπτωση)




Σχήμα Γ.4 - Σε αυτό το σημείο θα πρέπει να επιλεγεί σωστά το board με το οποίο θα γίνει η εξομοίωση



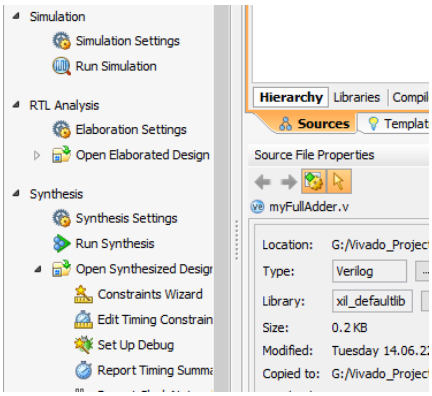

Σχήμα Γ.5 - Προσθήκη των εισόδων/εξόδων έτσι όπως ορίζονται στην verilog



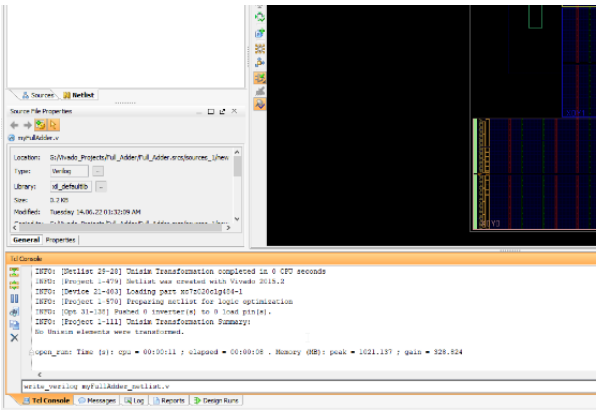

Σχήμα Γ.6 - Σε αυτό το σημείο όλα τα αρχεία έχουν εισαχθεί στο Project και μπορούμε να τρέξουμε το implementation για την επιβεβαίωση της ορθής λειτουργίας

## Γ.2 Εξαγωγή verilog αρχείων σε netlist

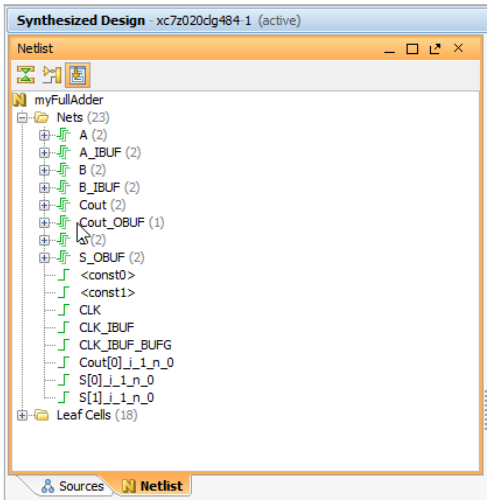
Στην συνέχεια, στα επόμενα σχήματα, φαίνεται η διαδικασία για την εξαγωγή του verilog αρχείου σε Netlist, έτσι ώστε αργότερα να γίνει parse από το python αρχείο που θα προσθέσει τους 2:1 Multiplexors.




Σχήμα Γ.7 - Εμφάνιση όλων των source αρχείων στο Project



Σχήμα Γ.8 - Εξαγωγή του netlist αρχείου με χρήση της εντολής "write_verilog"




Σχήμα Γ.9 - Το netlist αρχείο, έτσι όπως φαίνεται στο Synthesized Design

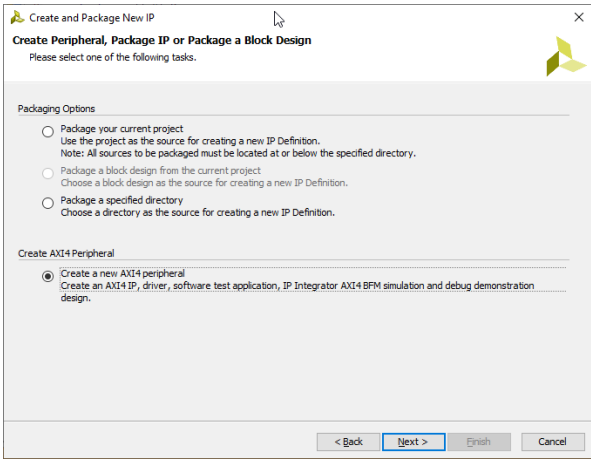
```
16 module myFullAdder
17    (CLK,
18     A,
19     A_WITHFAULT,
20     FE0,
21     FT0,
22     B,
23     S,
24     Cout);
25   input CLK;
26   input [1:0]A;
27   input [1:0]A_WITHFAULT;
28   input [1:0]FE0;
29   input [1:0]FT0;
30   input [1:0]B;
31   output [1:0]S;
32   output [1:0]Cout;
33
34   wire \<const0> ;
35   wire \<const1> ;
36   wire [1:0]A;
37   assign A_WITHFAULT = (FE0) ? FT0 : A;
38   wire [1:0]A_WITHFAULT;
39   wire [1:0]A_IBUF;
40   wire [1:0]B;
41   wire [1:0]B_IBUF;
```

Σχήμα Γ.10 - Το netlist αρχείο, έτσι όπως φαίνεται στο source files

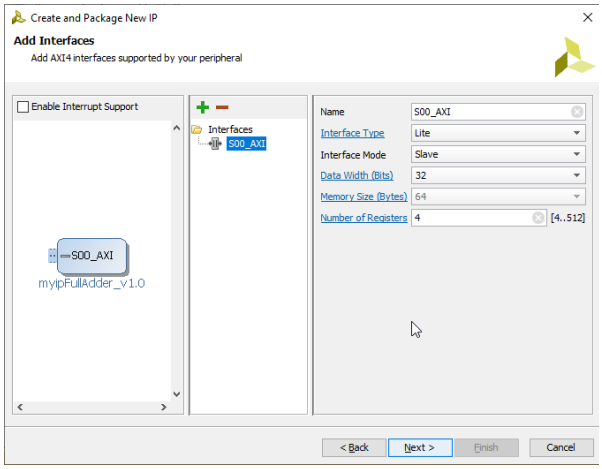
Στο Netlist που γίνεται εξαγωγή από το Vivado, θα πρέπει να μπουν τόσα Inputs όσα και τα wires, όπως επίσης θα πρέπει και να προστεθούν τα 2:1 Multiplexers.

## Γ.3 Δημιουργία IP block στο Vivado

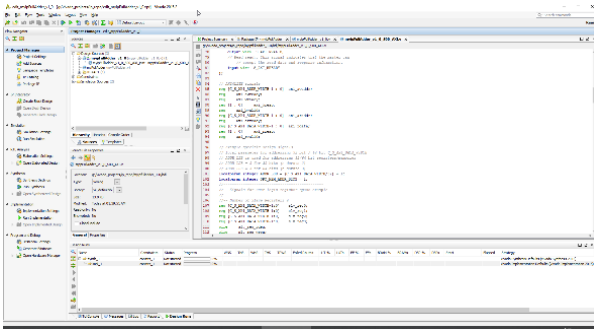
Στην συνέχεια, φαίνεται η διαδικασία για την δημιουργία του IP block, το οποίο θα πρέπει να προστεθεί στην σύνθεση του Vivado.



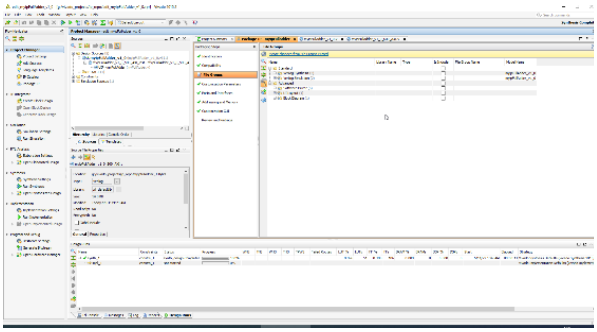

Σχήμα Γ.11 - Έναρξη διαδικασίας δημιουργίας ενός AXI4 peripheral




Σχήμα Γ.12 - Ορισμός της ποσότητας και του μεγέθους των καταχωρητών για το νέο IP block (στην συγκεκριμένη περίπτωση, 4 καταχωρητές των 32 bits)



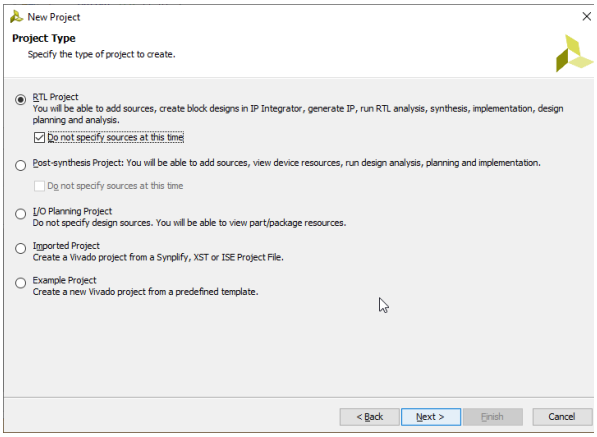
Σχήμα Γ.13 - Το παραγόμενο verilog αρχείο από το Vivado, για το νέο IP block



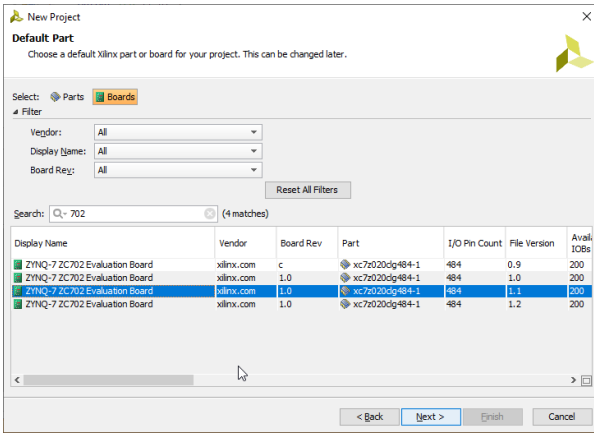
Σχήμα Γ.14 - Ορισμός της βασικής λειτουργίας του IP block. Να χρησιμοποιεί δηλαδή την λειτουργία του αρχικού μας verilog αρχείου

## Γ.4 Δημιουργία τελικού Project για εξομοίωση στο FPGA

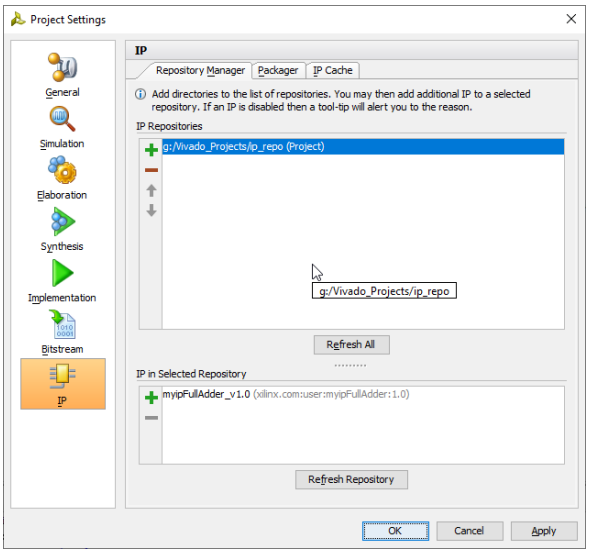
Στα επόμενα σχήματα, φαίνεται η δημιουργία του τελικού Project, το οποίο και θα τρέξει στο FPGA.



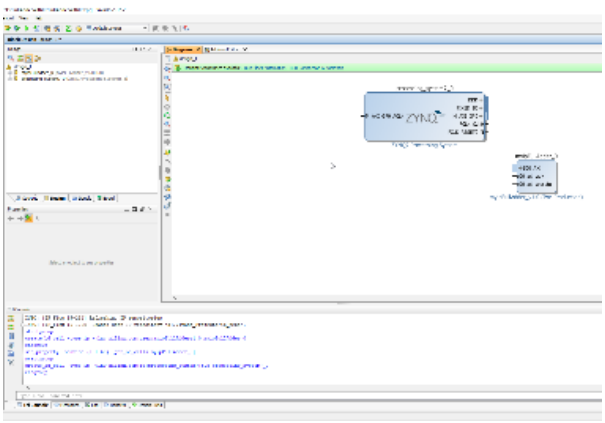

Σχήμα Γ.15 - Δημιουργία ενός νέου RTL Project



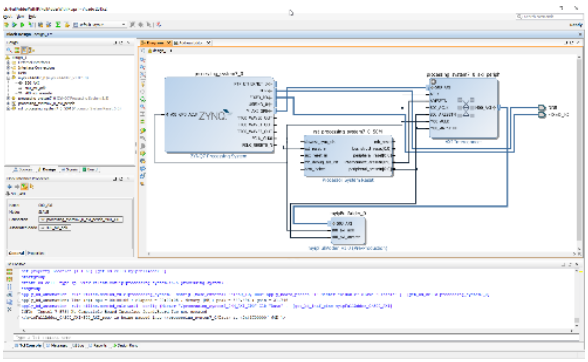

Σχήμα Γ.16 - Σε αυτό το σημείο θα πρέπει να επιλέξουμε σε ποιο board θα τρέξει το Project



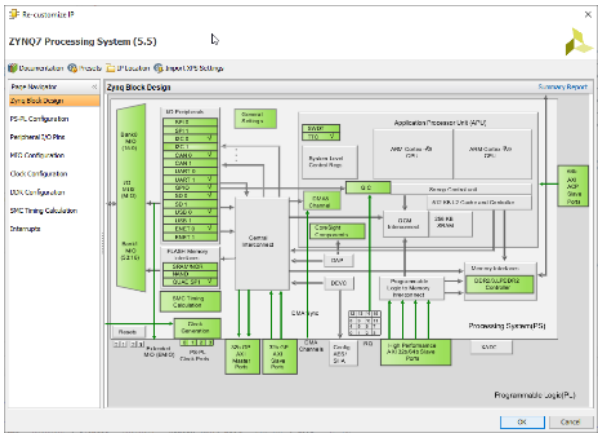

Σχήμα Γ.17 - Θα πρέπει να επιλέξουμε να εισάγομε από το Repository Manager, το IP block που δημιουργήσαμε στο προηγούμενο βήμα



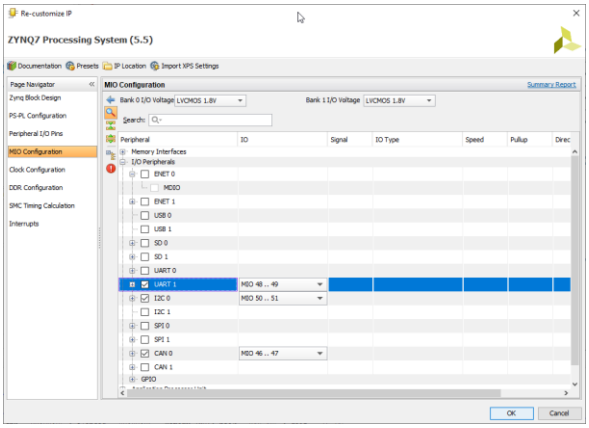
Σχήμα Γ.18 - Το IP block όπως φαίνεται στο design view



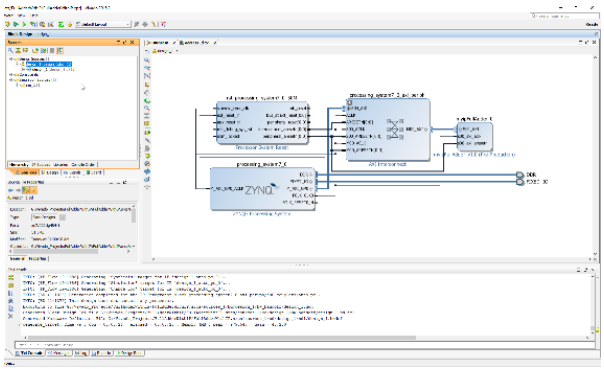
Σχήμα Γ.19 - Το Design αμέσως μετά την εκτέλεση του connection automation που προτείνει το Vivado



Σχήμα Γ.20 - Σε αυτό το σημείο θα πρέπει να αφαιρέσουμε τα modules που δεν χρειαζόμαστε (π.χ. Ethernet, USB κλπ) έτσι ώστε να απλοποιηθεί το Design



Σχήμα Γ.21 - Ομοίως θα πρέπει να αφαιρέσουμε και τα αντίστοιχα περιφερειακά στοιχεία

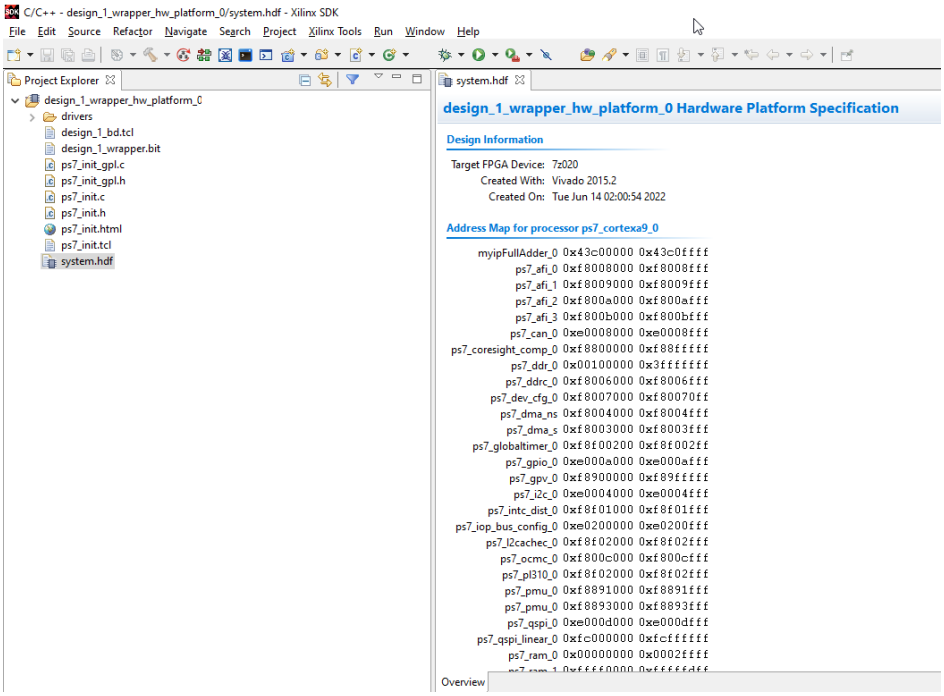


Σχήμα Γ.22 - Το Design αμέσως μετά την εκτέλεση του connection automation που προτείνει το Vivado

# ΠΑΡΑΡΤΗΜΑ Δ

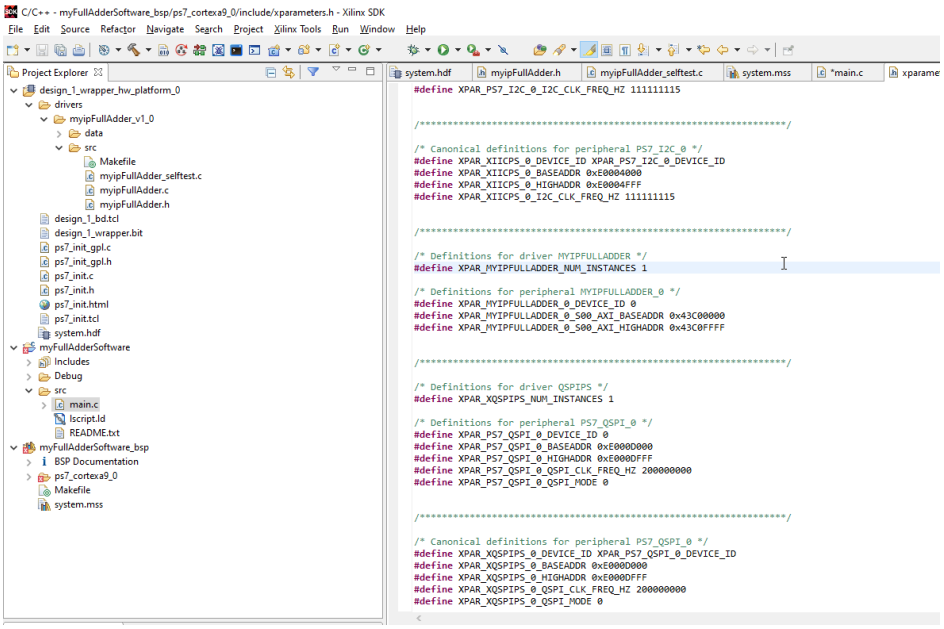
## ΚΑΘΟΡΙΣΜΟΣ ΔΙΕΥΘΥΝΣΕΩΝ ΜΝΗΜΗΣ ΓΙΑ ΤΟΥΣ ΚΑΤΑΧΩΡΗΤΕΣ ΤΟΥ DUT

Πριν γραφτεί ο κώδικας για τον controller, θα πρέπει να γίνει έλεγχος στο αρχείο system.hdf, έτσι ώστε να δούμε σε ποιο σημείο της μνήμης είναι οι καταχωρητές του κάθε IP block που θα υπάρχει στο τελικό design, οπότε και του IP block που θα κάνει την εξομοίωση. Στο συγκεκριμένο project, όπως φαίνεται και στην παρακάτω φωτογραφία, οι καταχωρητές βρίσκονται στην περιοχή μνήμης 0x43c00000 έως 0x430ffff.




Σχήμα Δ.1 - Σε αυτό το σημείο, θα πρέπει να σημειώσουμε την περιοχή μνήμης στην οποία έχουν αντιστοιχηθεί οι καταχωρητές του δικού μας IP block, για να γίνει εγγραφή/ανάγνωση αργότερα από τον controller

Στην συνέχεια, έτσι όπως φαίνεται και στο παρακάτω σχήμα, θα πρέπει να συμπληρώσουμε σωστά την περιοχή μνήμης στην οποία βρίσκονται οι καταχωρητές. Αυτό θα πρέπει να γίνει στο αρχείο xparameters.h.




Σχήμα Δ.2 - Θα πρέπει στο αρχείο xparameters.h να οριστεί η περιοχή μνήμης από το προηγούμενο βήμα

# ΠΑΡΑΡΤΗΜΑ Ε

## PYTHON ΚΩΔΙΚΑΣ ΓΙΑ ΤΟ PARSING ΕΝΟΣ NETLIST ΚΑΙ ΤΗΝ ΕΙΣΑΓΩΓΗ 2:1 MULTIPLEXORS

```
import re
import sys

netlist_file = open(sys.argv[1], 'r')
lines = netlist_file.readlines()

muxes_with_faults = []
input_outputs = []
fe_ft = []

faults_counter = 0

for line in lines:
      split_line = line.strip().split()
      if split_line and split_line[0] == 'wire':
            if split_line[1].startswith('['):
                  wire = re.split('\]|\[|;', split_line[1])
                  input_outputs.append(f'input
[{wire[1]}]{wire[2]}_WITHFAULT;')
                  muxes_with_faults.append(f'assign {wire[2]}_WITHFAULT
= (FE{faults_counter}) ? FT{faults_counter} : {wire[2]}')
                  fe_ft.append(f'FE{faults_counter},
FT{faults_counter},')
                  faults_counter = faults_counter + 1
```

# ΠΑΡΑΡΤΗΜΑ ΣΤ

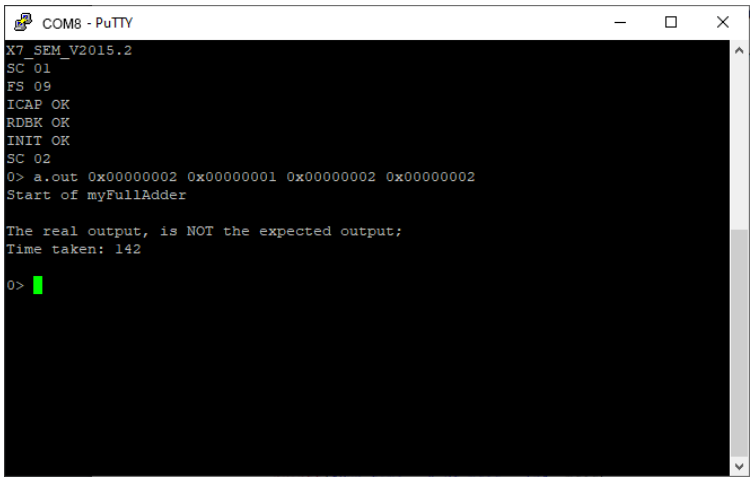
## Δημιουργια του Controller στο C++ Xilinx SDK

Τέλος, θα πρέπει να γραφτεί κώδικας C, στο C++ Xilinx SDK, το οποίο θα είναι ο controller, που θα ξεκινάει την εξομοίωση, ανάλογα με το Input κάθε φορά του χρήστη.

Τέλος, θα πρέπει να γράψουμε τον controller, ο οποίος θα παίρνει τέσσερις παραμέτρους, τέσσερις 32bit αριθμούς. Ο πρώτος 32bit αριθμός, θα είναι οι τιμές εισόδου στο κύκλωμα, ο δεύτερος 32bit αριθμός, θα είναι ο αναμενόμενες τιμές εξόδου από το κύκλωμα. Ο τρίτος 32bit αριθμός, θα είναι οι τιμές για το "Fault Enable" του κάθε 2:1 Multiplexor, το οποίο θα δηλώνει αν θα είναι ενεργοποιημένο το Fault ή όχι. Ο τέταρτος 32bit αριθμός, θα είναι οι τιμές για το "Fault Type" του κάθε 2:1 Multiplexor, το οποίο θα δηλώνει τον τύπο του σφάλματος (Stuck-at-1/stuck-at-0).

```
#include "xparameters.h"
#include "xil_io.h"
#include "xbasic_types.h"
#include "xtime_l.h"

int char2int(char input)
{
  if(input >= '0' && input <= '9')
    return input - '0';
  if(input >= 'A' && input <= 'F')
    return input - 'A' + 10;
  if(input >= 'a' && input <= 'f')
    return input - 'a' + 10;
  return 0;
}

// This function assumes src to be a zero terminated sanitized string
with
// an even number of [0-9a-f] characters, and target to be sufficiently
large
void hex2bin(const char* src, char* target)
{
  while(*src && src[1])
  {
    *(target++) = char2int(*src)*16 + char2int(src[1]);
    src += 2;
  }
}

int main(int argc, char *argv[]) {

      if (argc != 5) {
            xil_printf("\nNo Input/Output given. You have to give
exactly 4 parameters as a HEX values.\n");
            xil_printf("The first parameter is the input values
(32bit).\n");
            xil_printf("The second parameter is the expected output
values (32bit).\n");
            xil_printf("The third parameter is the Fault Enable values
(32bit).\n");
            xil_printf("The first parameter is the Fault Type values
(32bit).\n");
            return 1;
      }


      u32 adder_in;
      hex2bin(argv[1], adder_in);
      u32 adder_out;
      hex2bin(argv[2], adder_out);
      u32 faults_enable;
      hex2bin(argv[3], faults_enable);
      u32 faults_type;
      hex2bin(argv[4], faults_type);
```

```
	XTime start;
	XTime_GetTime(&start);

	xil_printf("Start of myFullAdder\n\n");

	Xil_Out32(XPAR_MYIPFULLADDER_0_S00_AXI_BASEADDR, adder_in);
	Xil_Out32(XPAR_MYIPFULLADDER_0_S00_AXI_BASEADDR+4,
faults_enable);
	Xil_Out32(XPAR_MYIPFULLADDER_0_S00_AXI_BASEADDR+8, faults_type);

	u32 real_adder_out;
	real_adder_out =
Xil_In32(XPAR_MYIPFULLADDER_0_S00_AXI_BASEADDR+12);

	XTime stop;
	XTime_GetTime(&stop);
	double duration;
	duration = 1.0 * (stop - start) / COUNTS_PER_SECOND;

	if (real_adder_out == adder_out) {
		xil_printf("The real output, is the expected output;\n");
	} else {
		xil_printf("The real output, is NOT the expected
output;\n");
	}
	xil_printf("Time taken: %d\n", duration);

	return 0;
}
```




Σχήμα ΣΤ.1 - Ο SEM IP Controller αρχικοποιημένος. Έχει γίνει εκτέλεση του προγράμματος C.

# Σύντομο Βιογραφικό

Ο Γεώργιος Μπέτσης γεννήθηκε στα Ιωάννινα, και είναι πτυχιούχος του Τμήματος Πληροφορικής του Πανεπιστημίου Δυτικής Αττικής. Από το 2002 έως σήμερα, εργάζεται ως Διαχειριστής Δικτύου και Προγραμματιστής στο Πληροφοριακό Σύστημα της Ελληνικής Αστυνομίας. Στα ερευνητικά του ενδιαφέροντα συμπεριλαμβάνονται τα FPGAs, οι Microcontrollers και όλες οι τεχνολογίες του Internet of Things, καθώς και οι τεχνολογίες των δικτύων δεδομένων.