ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΜΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΔΙΚΤΥΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΜΕΛΕΤΗ ΓΙΑ ΤΗ ΣΧΕΔΙΑΣΗ ΓΙΑ ΈΛΕΓΧΟ ΣΕ ΡΟΕΣ ΣΧΕΔΙΑΣΗΣ VLSI ΚΥΚΛΩΜΑΤΩΝ ΑΝΟΙΧΤΟΥ ΚΩΔΙΚΑ

Μπασούνας Διονύσιος

Επιβλέπων: Φώτιος Βαρτζιώτης

Επίκουρος Καθηγητής, ΔΕΠ

Άρτα, Ιούνιος, 2026

STUDY ON DESIGN FOR TEST IN OPEN-SOURCE VLSI DESIGN FLOWS

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, 24/6/2026

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής

   Φώτιος Βαρτζιώτης,

   Επίκουρος Καθηγητής
2. Μέλος επιτροπής

   Γρηγόριος Δουμένης,

   Αναπληρωτής Καθηγητής
3. Μέλος επιτροπής

   Ανδρέας Τσορμπατζόγλου,

   Αναπληρωτής Καθηγητής

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα μεταπτυχιακή εργασία είναι εκ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Μπασούνας, Διονύσιος

Υπογραφή

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή μου, κ. Φώτιο Βαρτζιώτη, για την καθοδήγηση, τις χρήσιμες παρατηρήσεις και την υποστήριξή του κατά τη διάρκεια εκπόνησης της παρούσας μεταπτυχιακής εργασίας. Η συμβολή του υπήρξε ιδιαίτερα σημαντική για την ολοκλήρωση και τη βελτίωση της εργασίας. Τέλος, θα ήθελα να ευχαριστήσω θερμά την οικογένειά μου για την αμέριστη στήριξη, την κατανόηση και την ενθάρρυνση που μου προσέφερε σε όλη τη διάρκεια των σπουδών μου.

ΠΕΡΙΛΗΨΗ

Η Σχεδίαση για Έλεγχο (Design for Testability – DfT) αποτελεί βασική προϋπόθεση για την αξιόπιστη παραγωγή σύγχρονων VLSI κυκλωμάτων. Στο πλαίσιο αυτό, η παρούσα εργασία μελετά την ενσωμάτωση και την επαλήθευση δομών scan chains σε πλήρως ανοικτή ροή σχεδίασης, βασισμένη σε εργαλεία όπως τα mflowgen και OpenROAD, με σημείο αναφοράς το benchmark κύκλωμα s5378 και τη βιβλιοθήκη standard cells Nangate45.

Αφετηρία της μελέτης αποτελεί ένα αρχικό scan-enhanced netlist που παράγεται από εργαλείο ανοικτού κώδικα για DfT, στο οποίο έχουν ήδη εισαχθεί scan flip-flops σε εσωτερικό υποσύστημα του κυκλώματος. Πάνω σε αυτό αναπτύσσεται μια μεθοδολογία αναδιοργάνωσης των scan δομών, η οποία εντοπίζει τα flip-flops, αξιοποιεί πληροφορία ομαδοποίησης των αλυσίδων και τα ανασυνδέει σε τέσσερις σαφώς ορισμένες scan chains με ανεξάρτητες εισόδους και εξόδους, παράγοντας ένα τροποποιημένο netlist κατάλληλο για περαιτέρω επαλήθευση.

Για τη λειτουργική επαλήθευση των αλυσίδων αναπτύσσονται ειδικά προγράμματα δοκιμής σε Verilog, τα οποία υλοποιούν ακολουθίες επαναφοράς, εκκαθάρισης και δοκιμές τύπου walking-1. Μέσω αυτών επιβεβαιώνεται ότι ένα μοναδικό λογικό ‘1’ διαδίδεται σωστά μέσα από κάθε αλυσίδα και εμφανίζεται ακριβώς μία φορά στην αντίστοιχη έξοδο, σύμφωνα με το μήκος της αλυσίδας. Τα αποτελέσματα των προσομοιώσεων παρέχουν τεκμηριωμένη ένδειξη της ορθής συνδεσμολογίας και της λειτουργικής ακεραιότητας των scan chains.

Παράλληλα, αξιοποιούνται τα αρχεία φυσικής υλοποίησης που προκύπτουν από τη ροή place and route για την εξαγωγή των συντεταγμένων των scan flip-flops και τη δημιουργία γραφικών αναπαραστάσεων των αλυσίδων στο layout. Με τον τρόπο αυτό επιβεβαιώνεται η συνέπεια μεταξύ λογικής περιγραφής και φυσικής υλοποίησης, γεφυρώνοντας το επίπεδο netlist με το επίπεδο διάταξης.

Συνολικά, η εργασία προτείνει μια επαναλήψιμη μικρο-ροή DfT σε ανοικτό περιβάλλον σχεδίασης, η οποία καλύπτει την αναδιοργάνωση, τη λειτουργική επαλήθευση και τη χωρική τεκμηρίωση των scan chains. Η προσέγγιση αυτή μπορεί να αποτελέσει βάση για μελλοντικές επεκτάσεις που σχετίζονται με ATPG, fault coverage και φυσική βελτιστοποίηση δομών ελέγχου.

Λέξεις-κλειδιά: Σχεδίαση για Έλεγχο (DfT), scan chains, ροές VLSI ανοικτού κώδικα, Fault, OpenROAD.

ABSTRACT

Design for Testability (DfT) is a fundamental prerequisite for the reliable production of modern VLSI circuits. In this context, the present thesis investigates the integration and verification of scan-chain structures within a fully open-source design flow, based on tools such as mflowgen and OpenROAD, using the s5378 benchmark circuit and the Nangate45 standard-cell library as a reference platform.

The study starts from an initial scan-enhanced netlist generated by an open-source DfT tool, in which scan flip-flops have already been inserted into an internal subsystem of the circuit. Building on this starting point, a methodology for scan-chain reorganization is developed, which identifies the relevant flip-flops, exploits chain-grouping information, and reconnects them into four clearly defined scan chains with independent inputs and outputs, producing a modified netlist suitable for further verification.

For the functional verification of the scan chains, dedicated Verilog testbenches are developed, implementing reset, flush, and walking-1 test sequences. Through these tests, it is confirmed that a single logic '1' propagates correctly through each chain and appears exactly once at the corresponding output, according to the chain length. The simulation results provide documented evidence of the correct interconnection and functional integrity of the scan chains.

In parallel, physical implementation files generated by the place-and-route flow are used to extract the coordinates of the scan flip-flops and to produce graphical representations of the chains at the layout level. In this way, consistency between the logical description and the physical implementation is verified, effectively bridging the netlist level and the layout level.

Overall, the thesis proposes a reproducible DfT micro-flow in an open-source design environment, covering scan-chain reorganization, functional verification, and spatial documentation. This approach can serve as a foundation for future extensions related to ATPG, fault coverage, and the physical optimization of test structures.

Keywords: Design for Testability (DfT), scan chains, open-source VLSI design flows, Fault, OpenROAD.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ/ΕΙΚΟΝΩΝ

ΠΙΝΑΚΑΣ ΣΥΝΤΟΜΟΓΡΑΦΙΩΝ

ATPG ..... Automatic Test Pattern Generation
BIST ..... Built-In Self-Test
CSV ..... Comma-Separated Values
CTS ..... Clock Tree Synthesis
DEF ..... Design Exchange Format
DfT ..... Design for Testability
DFF ..... D-type Flip-Flop
DRC ..... Design Rule Check
EDA ..... Electronic Design Automation
GDS / GDSII ..... Graphic Data System II
HDL ..... Hardware Description Language
JTAG ..... Joint Test Action Group
LEF ..... Library Exchange Format
RTL ..... Register-Transfer Level
VCD ..... Value Change Dump
VLSI ..... Very Large Scale Integration
WSL ..... Windows Subsystem for Linux
DAG ..... Directed Acyclic Graph

ΑΠΟΔΟΣΗ ΟΡΩΝ / ΓΛΩΣΣΑΡΙΟ

Automatic Test Pattern Generation (ATPG)
Αυτόματη δημιουργία διανυσμάτων ελέγχου για την ανίχνευση σφαλμάτων σε ψηφιακά κυκλώματα.

Boundary scan
Τεχνική ελέγχου διασυνδέσεων, συνήθως σύμφωνα με το πρότυπο JTAG, μέσω σειριακά συνδεδεμένων κελιών στις εισόδους/εξόδους του κυκλώματος.

Clock Tree Synthesis (CTS)
Στάδιο της φυσικής σχεδίασης στο οποίο κατασκευάζεται το δίκτυο διανομής του ρολογιού.

Design for Testability (DfT)
Σχεδιαστική προσέγγιση κατά την οποία ενσωματώνονται στο κύκλωμα δομές που διευκολύνουν τον έλεγχο και τη διάγνωση σφαλμάτων.

Fault coverage
Ποσοστό των θεωρητικά ανιχνεύσιμων σφαλμάτων που καλύπτονται από ένα σύνολο διανυσμάτων ελέγχου.

Flip-flop
Ακολουθιακό στοιχείο αποθήκευσης ενός bit πληροφορίας.

Full scan
Αρχιτεκτονική ελέγχου στην οποία σχεδόν όλα τα flip-flops του κυκλώματος μετατρέπονται σε scan flip-flops.

Gate-level netlist
Περιγραφή κυκλώματος σε επίπεδο λογικών πυλών και διασυνδέσεων, μετά τη σύνθεση.

Layout
Η φυσική διάταξη των στοιχείων και των διασυνδέσεων ενός ολοκληρωμένου κυκλώματος

πάνω στο chip.

Place and route
Η διαδικασία τοποθέτησης των standard cells και δρομολόγησης των συνδέσεων στο φυσικό σχέδιο.

Scan chain
Σειριακή αλυσίδα από scan flip-flops που επιτρέπει τη φόρτωση, μετατόπιση και παρατήρηση εσωτερικών καταστάσεων του κυκλώματος.

Scan flip-flop
Flip-flop με πρόσθετη λογική επιλογής ώστε να λειτουργεί είτε σε κανονική λειτουργία είτε ως στοιχείο scan αλυσίδας.

Shift mode
Κατάσταση λειτουργίας στην οποία οι scan αλυσίδες λειτουργούν ως shift registers για τη μεταφορά test δεδομένων.

Standard cell
Προτυποποιημένο λογικό κελί βιβλιοθήκης που χρησιμοποιείται στη σύνθεση και φυσική υλοποίηση ψηφιακών κυκλωμάτων.

Stuck-at fault
Μοντέλο σφάλματος κατά το οποίο ένας κόμβος θεωρείται μόνιμα καθηλωμένος στη λογική τιμή 0 ή 1.

Πρόγραμμα δοκιμής
Περιβάλλον προσομοίωσης που εφαρμόζει ερεθίσματα στο κύκλωμα και ελέγχει τις αποκρίσεις του.

Walking-1 test
Δοκιμή κατά την οποία ένα μοναδικό λογικό “1” μετατοπίζεται μέσα από μια scan αλυσίδα για έλεγχο της σωστής σειριακής συνδεσμολογίας.

ΕΙΣΑΓΩΓΗ

Η συνεχής αύξηση της πολυπλοκότητας των ψηφιακών ολοκληρωμένων κυκλωμάτων έχει καταστήσει τον έλεγχο αναπόσπαστο και ιδιαίτερα κρίσιμο μέρος της σχεδιαστικής διαδικασίας. Σε σύγχρονα VLSI συστήματα, η μεγάλη πυκνότητα ολοκλήρωσης, ο αυξημένος αριθμός λογικών στοιχείων και οι υψηλές απαιτήσεις αξιοπιστίας καθιστούν αναγκαία την ενσωμάτωση μηχανισμών που διευκολύνουν την ανίχνευση και τη διάγνωση σφαλμάτων ήδη από τα πρώτα στάδια της σχεδίασης. Στο πλαίσιο αυτό αναπτύχθηκε η φιλοσοφία της Σχεδίασης για Έλεγχο (Design for Testability – DfT), η οποία εισάγει κατάλληλες δομές στο κύκλωμα με σκοπό τη βελτίωση της ελεγξιμότητας και της παρατηρησιμότητας των εσωτερικών του καταστάσεων.

Κεντρική θέση στις τεχνικές DfT κατέχουν οι scan chains, δηλαδή σειριακές αλυσίδες από flip-flops που επιτρέπουν την πρόσβαση σε εσωτερικές καταστάσεις του κυκλώματος μέσω ειδικών εισόδων και εξόδων ελέγχου. Σε συνδυασμό με τεχνικές αυτόματης παραγωγής διανυσμάτων ελέγχου, οι scan δομές αποτελούν βασικό εργαλείο για την επίτευξη υψηλής κάλυψης σφαλμάτων. Ωστόσο, η αξιοποίησή τους πραγματοποιείται συχνά μέσα από εμπορικά περιβάλλοντα σχεδίασης, γεγονός που περιορίζει την αναπαραγωγιμότητα και τη δυνατότητα ακαδημαϊκής διερεύνησης πλήρων ροών ελέγχου.

Τα τελευταία χρόνια, η εξέλιξη εργαλείων ανοικτού κώδικα για λογική και φυσική σχεδίαση έχει δημιουργήσει νέες δυνατότητες για την ανάπτυξη πλήρων ροών VLSI, από την περιγραφή του κυκλώματος έως τη φυσική του υλοποίηση. Μέσα σε αυτό το περιβάλλον καθίσταται πλέον εφικτή όχι μόνο η υλοποίηση ψηφιακών σχεδίων με ανοικτά εργαλεία, αλλά και η διερεύνηση του τρόπου με τον οποίο μπορούν να ενσωματωθούν, να αναδιοργανωθούν και να επαληθευθούν δομές Σχεδίασης για Έλεγχο.

Η παρούσα εργασία εξετάζει την ενσωμάτωση και την επαλήθευση scan δομών σε ανοικτή ροή σχεδίασης VLSI. Η μελέτη επικεντρώνεται στην αναδιοργάνωση των scan chains ενός ακολουθιακού κυκλώματος, στη λειτουργική επαλήθευση της ορθής σειριακής τους συμπεριφοράς και στη σύνδεση της λογικής περιγραφής με το φυσικό επίπεδο υλοποίησης. Βασική συνεισφορά της εργασίας είναι ότι, μέσα σε μια ανοικτή ροή σχεδίασης, προσφέρεται η δυνατότητα μετατροπής μιας ενιαίας scan chain σε πολλαπλές ανεξάρτητες scan chains. Η προσέγγιση αυτή μειώνει τον αριθμό κύκλων που απαιτούνται για τη μετατόπιση ενός πλήρους

διανύσματος ελέγχου και, συνεπώς, μπορεί να μειώσει τον συνολικό χρόνο εφαρμογής των δοκιμών.

Για τον σκοπό αυτό αξιοποιούνται εργαλεία ανοικτού κώδικα για τη δημιουργία scan-enhanced netlist, για τη φυσική σχεδίαση και για την προσομοίωση, ενώ αναπτύσσονται και βοηθητικοί μηχανισμοί αυτοματοποιημένης επεξεργασίας και ελέγχου. Συγκεκριμένα, η εργασία οργανώνει τα εσωτερικά flip-flops του υπό μελέτη κυκλώματος σε πολλαπλές scan chains με ανεξάρτητες εισόδους και εξόδους, και στη συνέχεια επαληθεύει τη σωστή λειτουργία τους μέσω ειδικών προγραμμάτων δοκιμής.

Ιδιαίτερη έμφαση δίνεται στη γεφύρωση μεταξύ λογικού και φυσικού επιπέδου. Πέρα από τη δομική και λειτουργική επαλήθευση των scan chains, εξετάζεται και η αντιστοίχισή τους με τη φυσική τοποθέτηση των στοιχείων στο layout, ώστε να τεκμηριώνεται η συνέπεια ανάμεσα στη netlist περιγραφή και στη φυσική υλοποίηση. Με τον τρόπο αυτό, η εργασία δεν περιορίζεται στη θεωρητική προσέγγιση των scan δομών, αλλά προτείνει μια επαναλήψιμη μεθοδολογία ανάλυσης και επαλήθευσης σε περιβάλλον ανοικτού κώδικα.

Η εργασία οργανώνεται ως εξής: αρχικά παρουσιάζεται το θεωρητικό υπόβαθρο της Σχεδίασης για Έλεγχο και των scan-based τεχνικών. Στη συνέχεια περιγράφονται τα εργαλεία και η ροή σχεδίασης που χρησιμοποιήθηκαν, και κατόπιν αναλύεται η διαδικασία αναδιοργάνωσης και επαλήθευσης των scan chains. Ακολουθεί η ανάλυση σε επίπεδο φυσικής διάταξης και η οπτικοποίηση των αλυσίδων στο layout, ενώ στο τέλος συνοψίζονται τα βασικά συμπεράσματα και προτείνονται κατευθύνσεις για μελλοντική έρευνα.

1. Εισαγωγή στη Σχεδίαση για Έλεγχο και Στόχοι της Εργασίας

1.1 Κίνητρο και πλαίσιο της εργασίας

Η συνεχής αύξηση της πολυπλοκότητας των ολοκληρωμένων κυκλωμάτων VLSI καθιστά τον έλεγχο (testing) ένα από τα πιο κρίσιμα στάδια της σχεδίασης. Τα σύγχρονα κυκλώματα περιλαμβάνουν εκατοντάδες χιλιάδες ή και εκατομμύρια λογικές πύλες, γεγονός που καθιστά πρακτικά αδύνατο τον εξαντλητικό έλεγχο χωρίς την ύπαρξη ειδικών δομών Σχεδίασης για Έλεγχο (Design for Testability – DfT)**.** Σε αυτό το πλαίσιο, τεχνικές όπως οι scan chains, τα boundary scan και τα σχήματα Built-In Self-Test (BIST) αποτελούν βασικά εργαλεία για την επίτευξη υψηλού fault coverage με λογικό κόστος σε χρόνο και πόρους. [1], [18]

Τα τελευταία χρόνια, παρατηρείται αυξανόμενο ενδιαφέρον για την αξιοποίηση ροών σχεδίασης ανοικτού κώδικα (open-source VLSI design flows), τόσο στην ακαδημαϊκή κοινότητα όσο και στη βιομηχανία. Ροές όπως το mflowgen σε συνδυασμό με εργαλεία φυσικού σχεδιασμού (place & route) και βιβλιοθήκες standard cells ανοικτού κώδικα, δημιουργούν ένα οικοσύστημα στο οποίο είναι δυνατή η αναπαραγώγιμη και οικονομικά προσιτή υλοποίηση ψηφιακών σχεδιάσεων. Ωστόσο, η ενσωμάτωση δομών DfT σε τέτοιες ροές δεν είναι ακόμη πλήρως ώριμη ούτε τόσο τυποποιημένη όσο σε εμπορικά EDA εργαλεία. [3]–[6]

Σε αυτό το πλαίσιο εντάσσεται και η παρούσα μεταπτυχιακή εργασία, η οποία επικεντρώνεται στη μελέτη και πρακτική ενσωμάτωση δομών DfT σε μια ανοικτή ροή σχεδίασης για ένα αντιπροσωπευτικό κύκλωμα ελέγχου, το benchmark s5378**.** [2] Η εργασία αξιοποιεί εργαλεία όπως το Fault [8], [9] για εισαγωγή scan chains και παραγωγή netlists με δομές DfT, καθώς και ροή φυσικού σχεδιασμού μέσω του mflowgen και της πλατφόρμας Nangate45. [3]–[6], [11], [12]

1.2 Στόχοι και συνεισφορά της εργασίας

Κεντρικός στόχος της παρούσας εργασίας είναι η συστηματική μελέτη και υλοποίηση δομών Σχεδίασης για Έλεγχο σε ανοικτή ροή σχεδίασης VLSI, με έμφαση στις scan chains. Πιο συγκεκριμένα, η εργασία επικεντρώνεται στην ανάλυση και αναδιοργάνωση scan δομών που

έχουν εισαχθεί σε ακολουθιακό ψηφιακό κύκλωμα, στη δημιουργία πολλαπλών ανεξάρτητων scan chains και στην επαλήθευση της ορθής λειτουργίας τους τόσο σε επίπεδο netlist όσο και σε επίπεδο φυσικής υλοποίησης.

Η συνεισφορά της εργασίας είναι κατά κύριο λόγο πρακτική και συνδέεται άμεσα με τα βήματα που υλοποιήθηκαν. Αρχικά, αναπτύχθηκε μηχανισμός μετασχηματισμού και ανασύνδεσης του netlist, ο οποίος επιτρέπει την αυτόματη αναδιοργάνωση των scan chains με βάση εξωτερική πληροφορία περιγραφής των αλυσίδων που προκύπτει από τη ροή σχεδίασης. Μέσω της διαδικασίας αυτής δημιουργήθηκε ένα τροποποιημένο netlist, στο οποίο τα εσωτερικά flip-flops οργανώνονται σε πολλαπλές σαφώς ορισμένες scan chains με ανεξάρτητες εισόδους και εξόδους.

Στη συνέχεια, αναπτύχθηκαν ειδικά προγράμματα δοκιμής για τη λειτουργική επαλήθευση των αλυσίδων, με σκοπό να επιβεβαιωθεί ότι τα δεδομένα που εισάγονται στις scan εισόδους διαδίδονται σωστά κατά μήκος κάθε αλυσίδας και εμφανίζονται στις αντίστοιχες εξόδους. Για τον σκοπό αυτό χρησιμοποιήθηκαν διανύσματα τύπου walking-1 και διαδικασίες εκκαθάρισης, ώστε να ελεγχθεί τόσο η ορθή σειριακή σύνδεση των flip-flops όσο και η απουσία ανεπιθύμητων ασυνεπειών στη δομή των αλυσίδων.

Τέλος, παράχθηκαν γραφικές αναπαραστάσεις των scan chains σε επίπεδο διάταξης, οι οποίες αποτυπώνουν τη χωρική κατανομή των flip-flops κάθε αλυσίδας και διευκολύνουν τη σύνδεση της λογικής περιγραφής με τη φυσική υλοποίηση. Με τον τρόπο αυτό, η εργασία δεν περιορίζεται μόνο στη λειτουργική επαλήθευση των scan δομών, αλλά συμβάλλει και στη συστηματική τεκμηρίωση της συνέπειας μεταξύ λογικού και φυσικού επιπέδου σε περιβάλλον ανοικτού κώδικα.

1.3 Δομή της μεταπτυχιακής εργασίας

Η υπόλοιπη εργασία οργανώνεται σε έξι κεφάλαια, ακολουθώντας μια πορεία από το γενικό θεωρητικό υπόβαθρο προς τα ειδικά πρακτικά αποτελέσματα. Στο Κεφάλαιο 2 παρουσιάζεται το θεωρητικό πλαίσιο της Σχεδίασης για Έλεγχο. Γίνεται εισαγωγή στις βασικές έννοιες DfT, στις scan chains, στο boundary scan και στα σχήματα BIST, καθώς και στις βασικές αρχές των μεθόδων ATPG και στους δείκτες αξιολόγησης, όπως το fault coverage και η επιβάρυνση

επιφάνειας και ισχύος. Στόχος του κεφαλαίου είναι να δώσει στον αναγνώστη το απαραίτητο υπόβαθρο για να κατανοήσει τις αποφάσεις σχεδίασης που ακολουθούν.

Στο Κεφάλαιο 3 περιγράφεται αναλυτικά η ανοικτή ροή σχεδίασης που χρησιμοποιείται στην εργασία. Παρουσιάζονται τα βασικά εργαλεία της ροής, όπως το mflowgen και η πλατφόρμα φυσικού σχεδιασμού (OpenROAD / Nangate45), καθώς και το εργαλείο Fault για την εισαγωγή scan chains και την παραγωγή netlists με δομές DfT. Επιπλέον, περιγράφονται τα κύρια χαρακτηριστικά του benchmark κυκλώματος s5378, ώστε να καταστεί σαφές γιατί επιλέχθηκε ως μελέτη περίπτωσης.

Το Κεφάλαιο 4 επικεντρώνεται στα βήματα ενσωμάτωσης των scan chains στη ροή. Αρχικά παρουσιάζεται το αρχικό netlist και οι περιορισμοί της αυτόματης εισαγωγής scan από το Fault. Στη συνέχεια αναλύεται το εσωτερικό block *_uuf_* και περιγράφεται η διαδικασία ανακατασκευής τεσσάρων ανεξάρτητων scan chains με τη βοήθεια του script rewrite_scan_chains_from_uuf.py και των αρχείων uuf_chain*.txt. Η αφήγηση οδηγεί μέχρι τη δημιουργία του τροποποιημένου netlist s5378.chained.multi.v και τη σύνδεσή του με την υπόλοιπη ροή.

Στο Κεφάλαιο 5 παρουσιάζεται η πειραματική αξιολόγηση της προτεινόμενης προσέγγισης. Περιγράφονται τα προγράμματα δοκιμής που αναπτύχθηκαν, τα αποτελέσματα των προσομοιώσεων από τα walking-1 tests και τις διαδικασίες flush, καθώς και τα αρχεία VCD και τα logs που τεκμηριώνουν την ορθή λειτουργία των τεσσάρων scan chains. Όπου είναι εφικτό, συζητούνται επίσης στοιχεία που σχετίζονται με τη φυσική υλοποίηση, όπως η επίδραση των δομών DfT στην επιφάνεια και στο timing.

Τέλος, στο Κεφάλαιο 6 συνοψίζονται τα βασικά συμπεράσματα της εργασίας. Συζητούνται τα πλεονεκτήματα και οι περιορισμοί της ενσωμάτωσης DfT σε ανοικτές ροές σχεδίασης, και προτείνονται κατευθύνσεις για μελλοντική έρευνα, όπως η εφαρμογή της ίδιας μεθοδολογίας σε άλλα benchmark κυκλώματα, η πλήρης διασύνδεση με ροές ATPG ή η επέκταση σε πιο σύνθετες δομές DfT και πιο προηγμένες τεχνολογίες υλοποίησης.

2. Θεωρητικό υπόβαθρο Σχεδίασης για Έλεγχο (DfT)

2.1 Βασικές έννοιες Σχεδίασης για Έλεγχο (DfT)

Η Σχεδίαση για Έλεγχο (Design for Testability – DfT) αναφέρεται στο σύνολο των τεχνικών που εφαρμόζονται κατά τη φάση της σχεδίασης ενός ολοκληρωμένου κυκλώματος, με σκοπό να διευκολύνουν τον έλεγχο μετά την υλοποίησή του στο πυρίτιο. Καθώς η πολυπλοκότητα των ψηφιακών κυκλωμάτων αυξάνεται και οι τεχνολογίες ολοκλήρωσης γίνονται ολοένα και πυκνότερες, η πιθανότητα εμφάνισης κατασκευαστικών σφαλμάτων και λειτουργικών αστοχιών παραμένει σημαντική. Χωρίς ειδικές δομές DfT, ο έλεγχος ενός κυκλώματος σε επίπεδο εισόδων και εξόδων είναι είτε ιδιαίτερα δύσκολος είτε πρακτικά αδύνατος, καθώς μεγάλο μέρος της εσωτερικής κατάστασης, όπως τα εσωτερικά flip-flops, οι κόμβοι και τα ενδιάμεσα σήματα, δεν είναι εύκολα προσβάσιμο ή ελέγξιμο από τις κύριες θύρες. [1], [18]

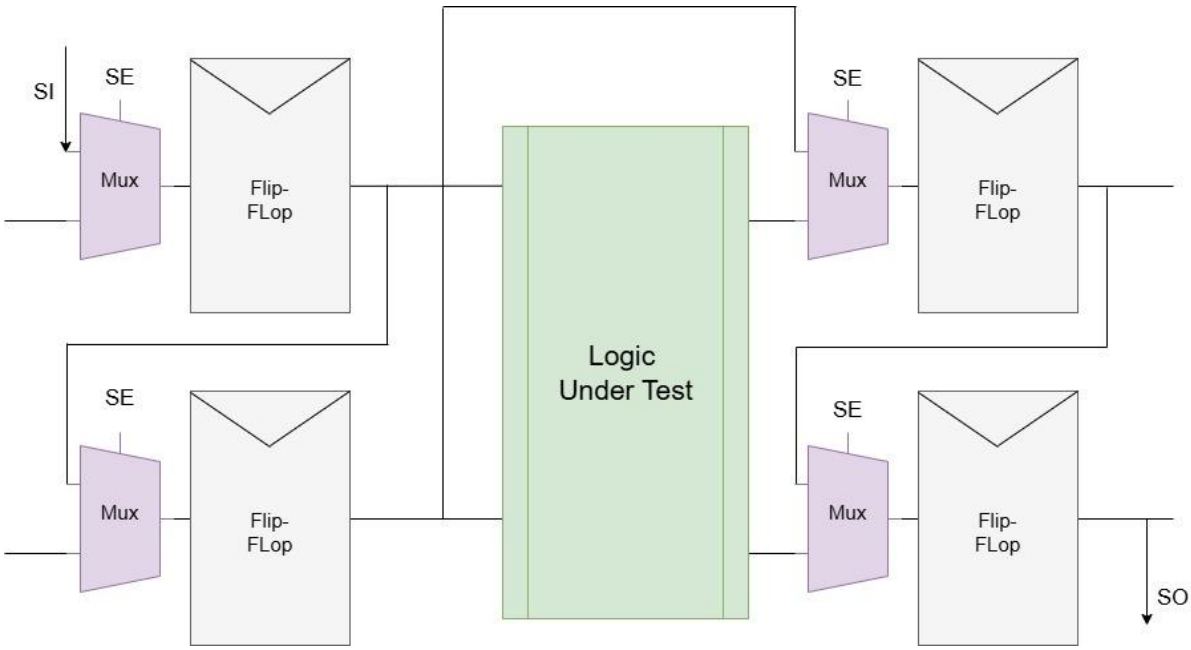
Βασική ιδέα της DfT είναι η εισαγωγή κατάλληλων «σημείων παρατήρησης» (observability) και «σημείων διέγερσης» (controllability) μέσα στο κύκλωμα. Με άλλα λόγια, ο σχεδιαστής επιδιώκει να διασφαλίσει ότι για κάθε σημαντικό εσωτερικό κόμβο μπορεί, αφενός, να επιβάλει μία επιθυμητή λογική τιμή μέσω κάποιου μονοπατιού εισόδου και, αφετέρου, να παρατηρήσει την επίδρασή της σε κάποια έξοδο. Στην πράξη, αυτό επιτυγχάνεται κυρίως με τη χρήση scan chains, boundary scan και δομών αυτοελέγχου (BIST), οι οποίες εμπλουτίζουν το κύκλωμα με πρόσθετη λογική, χωρίς ιδανικά να επηρεάζουν τη λειτουργία του σε κανονικό (functional) mode.

Για να αξιολογηθεί η αποτελεσματικότητα μιας τεχνικής DfT, χρησιμοποιούνται αφηρημένα μοντέλα σφαλμάτων, δηλαδή θεωρητικές προσεγγίσεις που περιγράφουν με απλοποιημένο τρόπο πιθανά ελαττώματα ή αστοχίες του κυκλώματος. Ένα από τα πιο διαδεδομένα είναι το μοντέλο stuck-at, σύμφωνα με το οποίο ένας κόμβος θεωρείται ότι έχει «κολλήσει» μόνιμα στη λογική τιμή 0 ή 1, ανεξάρτητα από τη σωστή λειτουργία του κυκλώματος. Άλλα συνήθη μοντέλα είναι τα transition ή delay faults, τα οποία σχετίζονται με καθυστερήσεις στη μετάβαση των σημάτων. Με βάση αυτά τα μοντέλα, ο έλεγχος ενός κυκλώματος μετατρέπεται σε πρόβλημα δημιουργίας διανυσμάτων εισόδου (test vectors), τα οποία μπορούν να ενεργοποιήσουν συγκεκριμένα σφάλματα και να καταστήσουν τις επιπτώσεις τους ορατές στις εξόδους. Το ποσοστό των θεωρητικά ανιχνεύσιμων σφαλμάτων που καλύπτονται από ένα σύνολο test vectors εκφράζεται συνήθως ως fault coverage και αποτελεί έναν από τους βασικούς δείκτες ποιότητας μιας λύσης DfT. [1], [15], [18]

Στο πλαίσιο της παρούσας εργασίας, η κύρια δομή DfT που εξετάζεται είναι οι scan chains στο εσωτερικό του υπό μελέτη ολοκληρωμένου κυκλώματος, όπως αυτές εισάγονται και αναδιαμορφώνονται με χρήση εργαλείων ανοικτού κώδικα.

2.2 Scan-based αρχιτεκτονική ελέγχου

Η πιο διαδεδομένη τεχνική Σχεδίασης για Έλεγχο (DfT) στα σύγχρονα ψηφιακά κυκλώματα είναι η χρήση scan chains. Σε ένα scan-based κύκλωμα, τα περισσότερα ή όλα τα flip-flops του σχεδίου τροποποιούνται έτσι ώστε να μπορούν να λειτουργούν με δύο διαφορετικούς τρόπους: ως συμβατικά flip-flops κατά την κανονική λειτουργία του κυκλώματος (functional mode) και ως τμήματα μιας σειριακής αλυσίδας μετατόπισης κατά τη λειτουργία ελέγχου (scan mode). Η εναλλαγή ανάμεσα στις δύο αυτές καταστάσεις ελέγχεται συνήθως από ένα ειδικό σήμα ελέγχου, όπως το scan enable ή το shift. [1], [18]



Εικόνα 2.1 Βασική αρχιτεκτονική scan

Η βασική δομική μονάδα μιας τέτοιας αρχιτεκτονικής είναι το scan flip-flop. Στην πράξη, ένα scan flip-flop προκύπτει από ένα συμβατικό D-type flip-flop στο οποίο προστίθεται λογική επιλογής, συνήθως ένας πολυπλέκτης στην είσοδο. Ο πολυπλέκτης αυτός επιλέγει είτε το κανονικό σήμα δεδομένων που προέρχεται από τη λειτουργική λογική του κυκλώματος είτε το σειριακό σήμα που προέρχεται από τη scan αλυσίδα. Όταν το κύκλωμα βρίσκεται σε κανονική λειτουργία, τα flip-flops ενημερώνονται από τη λογική του σχεδίου όπως σε ένα συνηθισμένο

ακολουθιακό κύκλωμα. Όταν όμως ενεργοποιείται η λειτουργία scan, τα ίδια στοιχεία συνδέονται σειριακά μεταξύ τους και λειτουργούν ως καταχωρητής ολίσθησης, επιτρέποντας τη φόρτωση και την παρατήρηση εσωτερικών καταστάσεων που διαφορετικά δεν θα ήταν άμεσα προσβάσιμες.

Σε μια πλήρη scan αρχιτεκτονική (full scan), σχεδόν όλα τα flip-flops του κυκλώματος μετατρέπονται σε scan flip-flops και εντάσσονται σε μία ή περισσότερες αλυσίδες. Αυτό σημαίνει ότι οι εσωτερικές καταστάσεις του ακολουθιακού κυκλώματος μπορούν να φορτωθούν ή να διαβαστούν σειριακά από τις scan εισόδους και εξόδους. Με τον τρόπο αυτό, το αρχικά ακολουθιακό κύκλωμα μπορεί, για τους σκοπούς του ελέγχου, να αντιμετωπιστεί σε μεγάλο βαθμό ως συνδυαστικό ανάμεσα σε δύο διαδοχικές φάσεις φόρτωσης και εκφόρτωσης των scan chains. Η ιδιότητα αυτή απλοποιεί σημαντικά τη διαδικασία αυτόματης παραγωγής διανυσμάτων ελέγχου (Automatic Test Pattern Generation – ATPG), δηλαδή τη διαδικασία εύρεσης κατάλληλων συνδυασμών εισόδων και εσωτερικών καταστάσεων που μπορούν να ενεργοποιήσουν πιθανά σφάλματα και να καταστήσουν τις επιπτώσεις τους ορατές στις εξόδους ή στις scan αλυσίδες. [1], [18]

Στην πράξη, το αποτέλεσμα της εισαγωγής scan δομών αποτυπώνεται σε ένα τροποποιημένο netlist, το οποίο συχνά αναφέρεται ως chained netlist ή scan-enhanced netlist. Πρόκειται για netlist στο οποίο τα αρχικά flip-flops έχουν αντικατασταθεί ή εμπλουτιστεί με scan flip-flops και έχουν προστεθεί οι απαραίτητες συνδέσεις ώστε να σχηματίζονται σειριακές αλυσίδες ελέγχου. Με άλλα λόγια, το chained netlist είναι η εκδοχή του κυκλώματος μετά την εισαγωγή της scan λογικής, έτοιμη να χρησιμοποιηθεί τόσο για προσομοίωση και επαλήθευση όσο και για περαιτέρω βήματα ελέγχου, όπως ATPG και fault simulation.

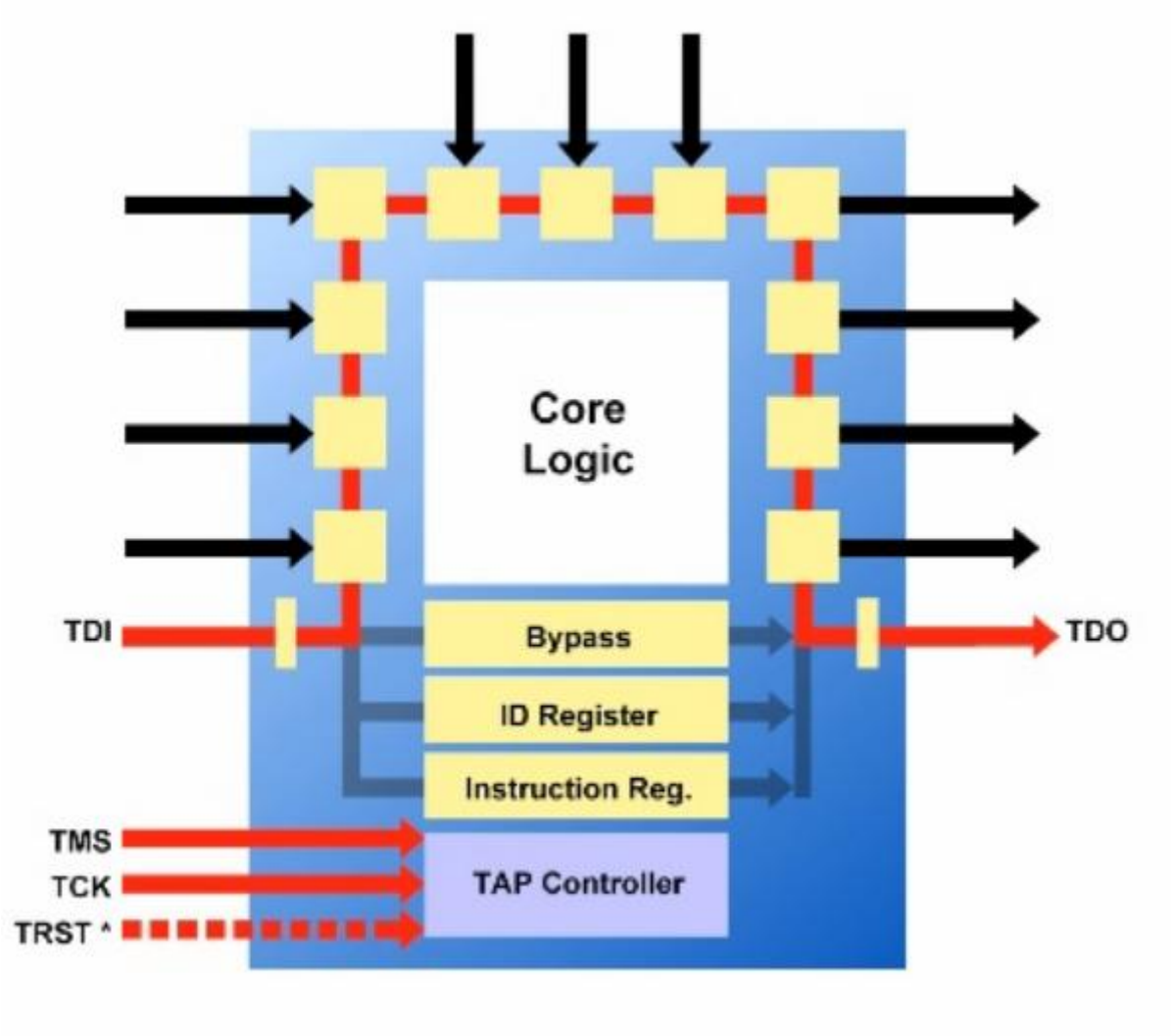
Υπάρχουν επίσης προσεγγίσεις μερικού scan (partial scan), όπου μόνο ένα υποσύνολο των flip-flops εντάσσεται στις scan αλυσίδες, ώστε να επιτευχθεί καλύτερη ισορροπία ανάμεσα στη testability, την επιφάνεια, την κατανάλωση ισχύος και την επίπτωση στο timing. Η επιλογή μεταξύ full scan και partial scan εξαρτάται από τους στόχους της σχεδίασης και τους περιορισμούς της εκάστοτε ροής υλοποίησης.

Στο πλαίσιο της παρούσας εργασίας, η scan-based αρχιτεκτονική αποτελεί τον βασικό μηχανισμό ελέγχου που μελετάται. Η ανάλυση επικεντρώνεται στην αναδιοργάνωση πολλαπλών scan chains, στη δομική και λειτουργική επαλήθευσή τους και στη σύνδεσή τους με το φυσικό επίπεδο υλοποίησης. Η ύπαρξη πολλαπλών αλυσίδων, αντί μίας ενιαίας και πολύ

μεγάλης αλυσίδας, παρουσιάζει πρακτικά πλεονεκτήματα, καθώς μειώνει τον αριθμό κύκλων που απαιτούνται για τη μετατόπιση ενός πλήρους διανύσματος ελέγχου και μπορεί να διευκολύνει καλύτερη χωρική κατανομή των flip-flops στο layout.

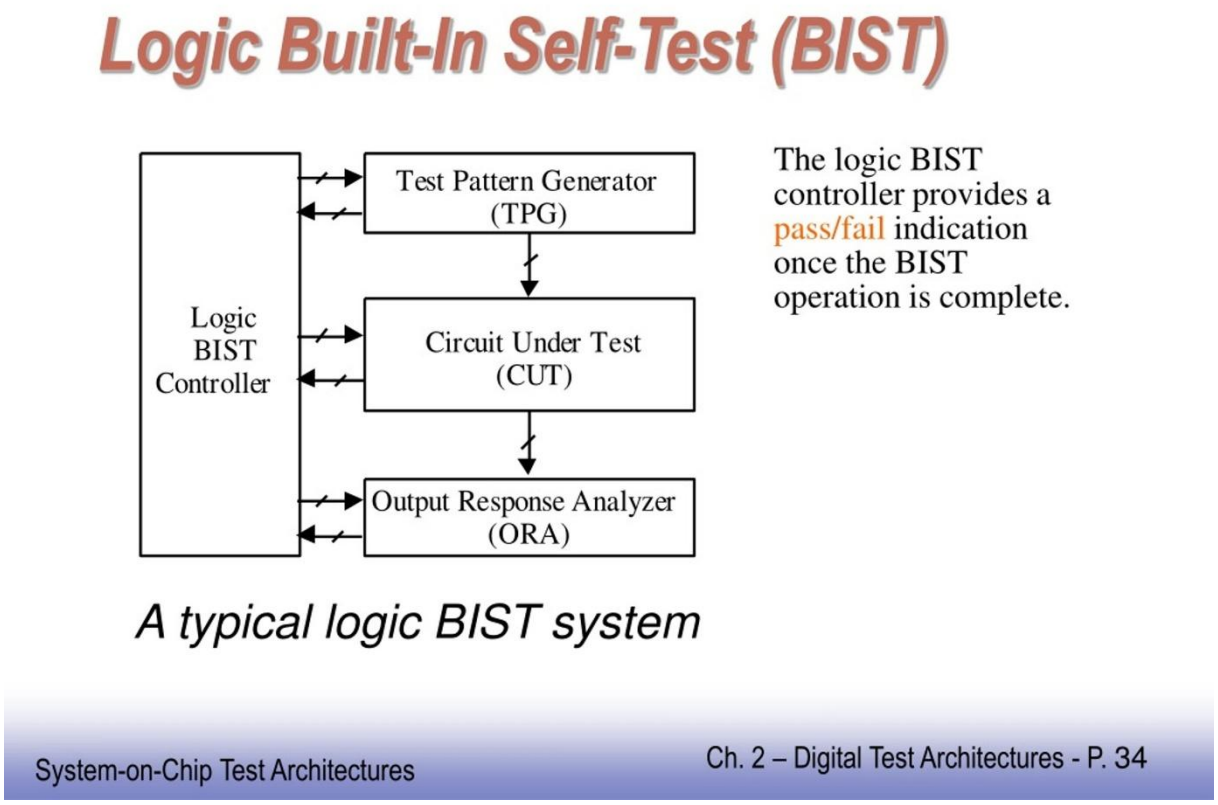
2.3 Boundary scan και BIST

Πέρα από τις εσωτερικές scan chains, στις σύγχρονες σχεδιάσεις σημαντικό ρόλο διαδραματίζουν και οι τεχνικές boundary scan και Built-In Self-Test (BIST). Το boundary scan, όπως τυποποιήθηκε στο πρότυπο JTAG (IEEE 1149.1), εισάγει μια τυποποιημένη αρχιτεκτονική πρόσβασης ελέγχου στο ολοκληρωμένο κύκλωμα μέσω του Test Access Port (TAP) και ενός boundary-scan register. Στην αρχιτεκτονική αυτή, ειδικά boundary scan cells τοποθετούνται γύρω από τον πυρήνα του κυκλώματος, συνήθως στις εισόδους και εξόδους του, ώστε να καθίσταται δυνατή η δειγματοληψία, η φόρτωση και η παρατήρηση σημάτων χωρίς να απαιτείται φυσική πρόσβαση σε κάθε ακροδέκτη. Με τον τρόπο αυτό διευκολύνεται ο έλεγχος διασυνδέσεων μεταξύ ολοκληρωμένων κυκλωμάτων επάνω σε πλακέτα, καθώς και ο εντοπισμός σφαλμάτων όπως ανοιχτές συνδέσεις και βραχυκυκλώματα. [16]



Εικόνα 2.2. Βασική αρχιτεκτονική boundary scan με TAP controller και boundary scan cells γύρω από την core logic.

Το BIST ακολουθεί διαφορετική φιλοσοφία, καθώς μεταφέρει μέρος της διαδικασίας ελέγχου στο εσωτερικό του ίδιου του κυκλώματος. Αντί η παραγωγή διανυσμάτων ελέγχου και η συλλογή αποκρίσεων να βασίζονται αποκλειστικά σε εξωτερικό εξοπλισμό, ενσωματώνονται στο chip κατάλληλες δομές που επιτρέπουν αυτοέλεγχο. Σε μια τυπική αρχιτεκτονική logic BIST, μία μονάδα παραγωγής ψευδοτυχαίων διανυσμάτων, συχνά υλοποιημένη με LFSR, τροφοδοτεί το κύκλωμα υπό έλεγχο, ενώ οι αποκρίσεις του συμπυκνώνονται από έναν αναλυτή υπογραφής, συχνά τύπου MISR. Το τελικό αποτέλεσμα συγκρίνεται με μία αναμενόμενη υπογραφή, ώστε να εξαχθεί ένδειξη επιτυχίας ή αποτυχίας του ελέγχου. Το πλεονέκτημα της προσέγγισης αυτής είναι ότι μειώνει την εξάρτηση από εξωτερικά ATE και τον όγκο των απαιτούμενων test δεδομένων, αν και συνεπάγεται πρόσθετο hardware overhead και αυξημένη σχεδιαστική πολυπλοκότητα. [1], [18]



Εικόνα 2.3 Ενδεικτική αρχιτεκτονική logic BIST με ελεγκτή BIST, γεννήτρια διανυσμάτων ελέγχου (TPG), κύκλωμα υπό έλεγχο (CUT) και αναλυτή αποκρίσεων (ORA).

Σε σχέση με τις εσωτερικές scan chains, οι τεχνικές boundary scan και BIST καλύπτουν διαφορετικές αλλά συμπληρωματικές ανάγκες. Το boundary scan εστιάζει κυρίως στον έλεγχο διασυνδέσεων και στην τυποποιημένη πρόσβαση ελέγχου σε επίπεδο chip και πλακέτας, ενώ το BIST εστιάζει στην ενσωμάτωση μηχανισμών αυτοελέγχου μέσα στο ίδιο το κύκλωμα. Για τον λόγο αυτό, αποτελούν βασικά στοιχεία του ευρύτερου πλαισίου της Σχεδίασης για Έλεγχο, ακόμη και όταν η κύρια έμφαση μιας μελέτης δίνεται στις εσωτερικές scan δομές.

Στο πλαίσιο της παρούσας εργασίας, η κύρια έμφαση δεν δίνεται στην υλοποίηση boundary scan ή πλήρων BIST δομών, αλλά στις εσωτερικές scan chains και στη διασύνδεσή τους με την ανοικτή ροή σχεδίασης. Παρ’ όλα αυτά, η κατανόηση των τεχνικών αυτών είναι χρήσιμη, επειδή εντάσσει τη μελέτη των scan chains μέσα στο ευρύτερο πλαίσιο των σύγχρονων μεθοδολογιών DfT.

2.4 Αυτόματη δημιουργία διανυσμάτων ελέγχου (ATPG) και μετρικές ελέγξιμότητας

Η ύπαρξη scan chains, από μόνη της, δεν αρκεί για να διασφαλίσει υψηλή ποιότητα ελέγχου. Απαιτείται επίσης η παραγωγή κατάλληλων διανυσμάτων τεστ (test vectors), τα οποία θα διεγείρουν συγκεκριμένα μοντέλα σφαλμάτων και θα κάνουν τις επιπτώσεις τους ορατές στις εξόδους. Η διαδικασία αυτόματης παραγωγής διανυσμάτων ονομάζεται Automatic Test Pattern Generation (ATPG) και αποτελεί βασικό κομμάτι του flow δοκιμών. [1], [18]

Σε ένα full-scan κύκλωμα, το ATPG αντιμετωπίζει το σχέδιο ως μια ακολουθία συνδυαστικών δικτύων μεταξύ διαδοχικών φορτώσεων και εκφορτώσεων των scan chains. Τα flip-flops λειτουργούν ως παρατηρήσιμα/ελεγχόμενα σημεία, γεγονός που απλοποιεί την αναζήτηση διανυσμάτων για μοντέλα σφαλμάτων όπως τα stuck-at. Ο αλγόριθμος ATPG προσπαθεί, για κάθε πιθανό σφάλμα, να βρει μια εκχώρηση στις εισόδους και στις scan αλυσίδες ώστε η διαφορά μεταξύ "σωστού" και "ελαττωματικού" κυκλώματος να διαδοθεί σε κάποια έξοδο ή σε κάποιο flip-flop που μπορούμε να διαβάσουμε μέσω scan.

Στο μοντέλο stuck-at fault, κάθε κόμβος ή γραμμή του κυκλώματος εξετάζεται σαν να έχει καθηλωθεί μόνιμα στη λογική τιμή 0 ή στη λογική τιμή 1. Για να θεωρηθεί ένα τέτοιο σφάλμα ανιχνεύσιμο, το ATPG πρέπει πρώτα να το ενεργοποιήσει, δηλαδή να δημιουργήσει στο σωστό κύκλωμα την αντίθετη λογική τιμή από αυτήν του σφάλματος, και στη συνέχεια να διαδώσει τη διαφορά μέχρι ένα παρατηρήσιμο σημείο, όπως κύρια έξοδο ή έξοδο scan chain. Επομένως, η ύπαρξη scan chains αυξάνει πρακτικά την παρατηρησιμότητα των εσωτερικών καταστάσεων και διευκολύνει την ανίχνευση stuck-at faults που χωρίς scan θα έμεναν κρυμμένα μέσα στην ακολουθιακή λογική.

Η ποιότητα της διαδικασίας ελέγχου μετριέται με διάφορους δείκτες. Ο σημαντικότερος είναι συνήθως το fault coverage, δηλαδή το ποσοστό των θεωρητικά ανιχνεύσιμων σφαλμάτων που καλύπτονται από το σύνολο των διανυσμάτων. Άλλες σημαντικές παράμετροι είναι ο αριθμός διανυσμάτων (test length), ο χρόνος που απαιτείται για την εκτέλεσή τους, καθώς και η επιβάρυνση σε επιφάνεια, κατανάλωση ισχύος και καθυστερήσεις που προκαλούν οι δομές DfT στο κύκλωμα. [1], [15], [18]

Στην παρούσα εργασία, η έμφαση δίνεται περισσότερο στην ορθή δομή και λειτουργία των scan chains στο netlist και λιγότερο στη λεπτομερή ποσοτική ανάλυση του fault coverage. Παρ'

όλα αυτά, τα προγράμματα δοκιμής που αναπτύχθηκαν (μέσω walking-1 και flush tests) υλοποιούν στην πράξη ειδικές περιπτώσεις διανυσμάτων ελέγχου, με στόχο να αποδειχθεί ότι κάθε αλυσίδα μεταφέρει σωστά την πληροφορία από την είσοδο στην έξοδο και ότι δεν υπάρχουν σφάλματα στην ανακατασκευή των αλυσίδων. Αυτή η διαδικασία μπορεί να θεωρηθεί ως "δομική" επαλήθευση της DfT αρχιτεκτονικής, η οποία συμπληρώνει και προετοιμάζει το έδαφος για πιο πλήρη ATPG-based αξιολόγηση.

Συμπερασματικά, τα stuck-at faults αποτελούν το βασικό και πιο απλό μοντέλο σφαλμάτων που χρησιμοποιείται για την αρχική αξιολόγηση μιας ροής DfT. Παρότι δεν περιγράφουν όλες τις πιθανές αστοχίες ενός πραγματικού ολοκληρωμένου κυκλώματος, όπως σφάλματα καθυστέρησης ή μεταβατικά φαινόμενα, παρέχουν ένα πρακτικό μέτρο ελεγξιμότητας και παρατηρησιμότητας. Για τον λόγο αυτό, η βελτίωση της δυνατότητας ανίχνευσής τους μέσω scan chains και ATPG αποτελεί σημαντικό πρώτο βήμα πριν από πιο σύνθετες αναλύσεις fault coverage, timing-aware testing και φυσικής βελτιστοποίησης των δομών ελέγχου. [1], [15], [18]

3. Εργαλεία ανοιχτού κώδικα και ροές σχεδίασης που χρησιμοποιήθηκαν

3.1 Το κύκλωμα s5378 ως μελέτη περίπτωσης

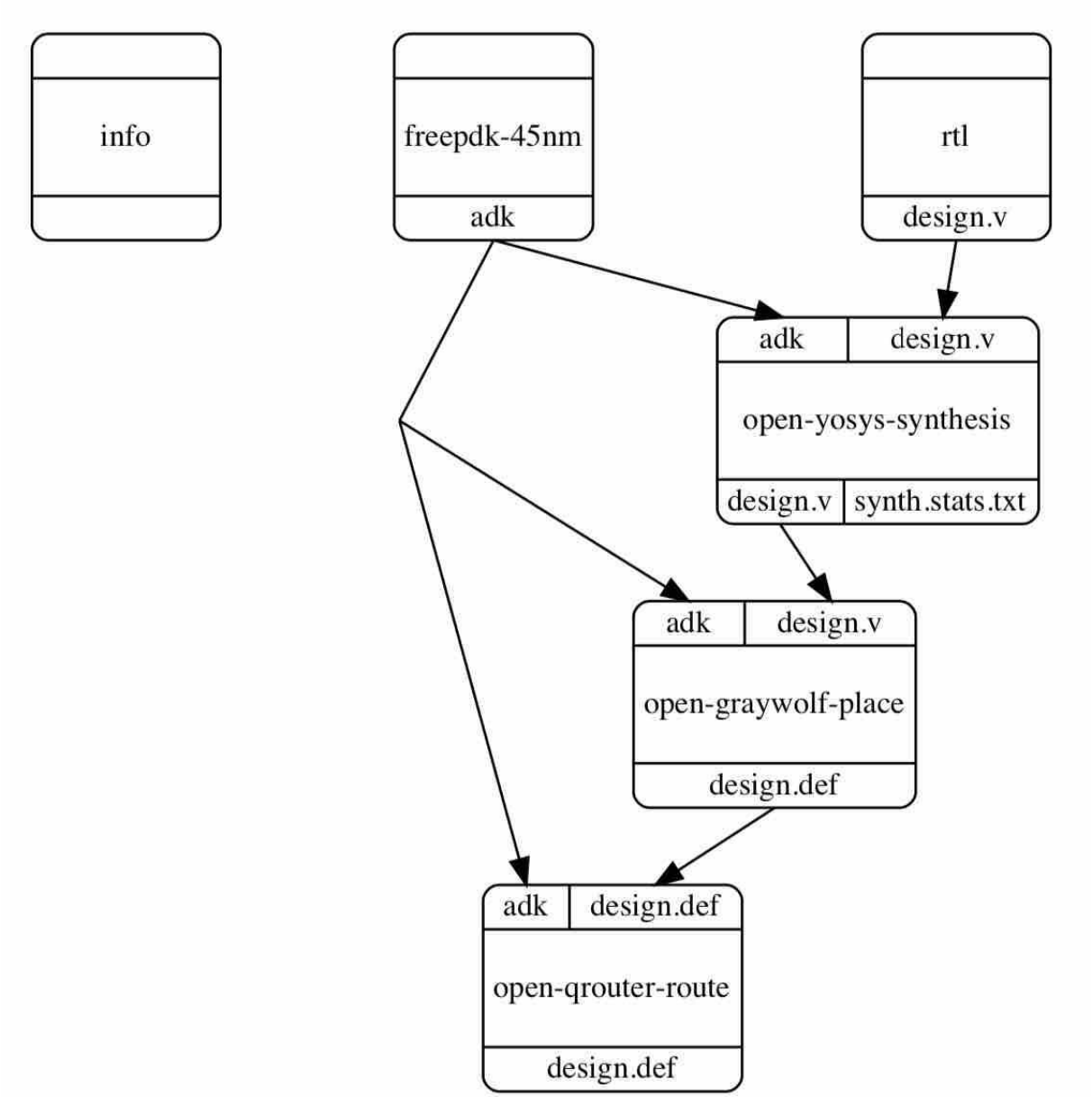
Ως βασικό αντικείμενο μελέτης στην παρούσα εργασία χρησιμοποιείται το κύκλωμα s5378, ένα τυπικό ακολουθιακό benchmark κύκλωμα από την οικογένεια ISCAS'89, το οποίο συναντάται συχνά στη βιβλιογραφία για αξιολόγηση τεχνικών DfT και ATPG. Το κύκλωμα περιλαμβάνει σημαντικό αριθμό flip–flops και εσωτερική λογική, με αποτέλεσμα να αποτελεί ρεαλιστικό παράδειγμα για μελέτη scan-based ελέγχου. [2]

Στο περιβάλλον της εργασίας, το s5378 δεν αντιμετωπίζεται μόνο ως αφηρημένο RTL, αλλά ως ένα πλήρες netlist πυλών, το οποίο περνά μέσα από μια ανοικτή ροή φυσικού σχεδιασμού. Στο netlist που παρέχεται από το εργαλείο Fault εμφανίζεται ένα εσωτερικό block με όνομα __uuf__, το οποίο συγκεντρώνει τα περισσότερα από τα flip–flops του κυκλώματος. Η αναδιοργάνωση των scan chains εστιάζει κυρίως σε αυτό το block, καθώς εκεί συγκεντρώνονται οι δομές που επηρεάζουν περισσότερο την ελέγξιμη και παρατηρήσιμη κατάσταση του κυκλώματος.

Η επιλογή του s5378 ως μελέτη περίπτωσης είναι ιδιαίτερα χρήσιμη για δύο λόγους. Από τη μία πλευρά, είναι αρκετά μικρό ώστε να μπορεί να αναλυθεί λεπτομερώς, τόσο σε επίπεδο netlist όσο και σε επίπεδο layout. Από την άλλη πλευρά, είναι αρκετά σύνθετο ώστε να αναδείξει πρακτικά ζητήματα όπως ο τρόπος με τον οποίο τα εργαλεία σύνθεσης, scan insertion και φυσικού σχεδιασμού τροποποιούν τη δομή των flip–flops και των scan chains.

3.2 Το mflowgen και ροή OpenROAD για φυσικό σχεδιασμό

Για τη ροή φυσικού σχεδιασμού χρησιμοποιείται το mflowgen, ένα σύστημα περιγραφής και εκτέλεσης flows ανοιχτού κώδικα, το οποίο επιτρέπει τη σύνθεση πολύπλοκων αλυσίδων εργαλείων (toolchain) με αναπαραγώγιμο τρόπο. Στο πλαίσιο της παρούσας εργασίας αξιοποιείται μια παραμετροποιημένη ροή, η οποία λαμβάνει ως είσοδο το netlist του υπό μελέτη κυκλώματος και παράγει σταδιακά αρχεία DEF και GDS κατάλληλα για την τελική φυσική υλοποίηση. Η επιλογή του mflowgen δεν περιορίζεται σε πρακτικούς λόγους αυτοματοποίησης, αλλά συνδέεται άμεσα με τη δυνατότητα πλήρους τεκμηρίωσης και επανάληψης της διαδικασίας σχεδίασης. [5], [6]



Εικόνα 3.1 Παράδειγμα γράφου ροής (DAG) στο mflowgen

Η εικόνα παρουσιάζει έναν κατευθυνόμενο άκυκλο γράφο (DAG) της ροής φυσικού σχεδιασμού, στον οποίο αποτυπώνονται τα βασικά στάδια της σύνθεσης, της τοποθέτησης και της δρομολόγησης, καθώς και οι εξαρτήσεις μεταξύ τους. Από το διάγραμμα φαίνεται ότι η περιγραφή του κυκλώματος και το τεχνολογικό υπόβαθρο της ροής τροφοδοτούν διαδοχικά τα στάδια υλοποίησης, μέχρι την παραγωγή του τελικού φυσικού σχεδίου.

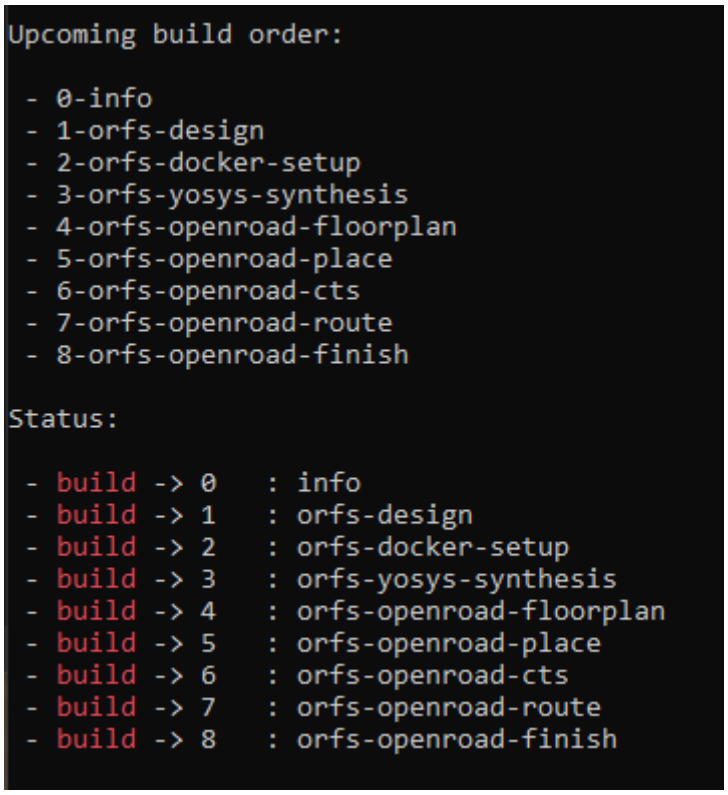
Το freepdk-45nm αποτελεί την τεχνολογική βάση της ροής σχεδίασης και χρησιμοποιείται με τη μορφή adk (ASIC Design Kit) σε όλα τα βασικά στάδια υλοποίησης. Μέσω αυτού παρέχονται στα εργαλεία της ροής τα απαραίτητα τεχνολογικά δεδομένα, όπως οι διαθέσιμες

standard-cell βιβλιοθήκες, οι περιγραφές φυσικής σχεδίασης και οι κανόνες που απαιτούνται για σύνθεση, τοποθέτηση και δρομολόγηση. Όπως φαίνεται και στο διάγραμμα ροής, το ίδιο adk τροφοδοτεί τόσο το στάδιο σύνθεσης όσο και τα επόμενα στάδια φυσικού σχεδιασμού, εξασφαλίζοντας ότι όλα τα εργαλεία λειτουργούν με κοινό τεχνολογικό υπόβαθρο. Η ενιαία αυτή τεχνολογική αναφορά είναι κρίσιμη για τη συνέπεια της ροής, καθώς επιτρέπει τη μετάβαση από τη λογική περιγραφή του κυκλώματος στην τελική φυσική υλοποίηση με συμβατό και αναπαραγώγιμο τρόπο.

Η εκτέλεση της ροής πραγματοποιείται σε διαδοχικά στάδια, τα οποία αντιστοιχούν σε συγκεκριμένες φάσεις της φυσικής υλοποίησης. Αρχικά δημιουργείται καθαρός κατάλογος build, ώστε κάθε εκτέλεση να ξεκινά από γνωστή και ελεγχόμενη κατάσταση, αποφεύγοντας την επίδραση παλαιότερων ενδιάμεσων αρχείων. Στη συνέχεια, η εντολή δημιουργίας της ροής (mflowgen run) παράγει τη δομή των σταδίων που απαιτούνται για την υλοποίηση του σχεδίου, σύμφωνα με την παραμετροποίηση της αντίστοιχης σχεδιαστικής ροής. Η εκτέλεση προχωρά μέσω διαδοχικών κλήσεων των σταδίων της ροής, από την αρχικοποίηση και τη σύνθεση μέχρι το τελικό στάδιο ολοκλήρωσης (finish).

Πρακτικά, για να διασφαλιστεί η αναπαραγωγιμότητα, κάθε νέα εκτέλεση της ροής ξεκινά από καθαρό κατάλογο build, ώστε να αποφεύγεται η επίδραση παλαιότερων ενδιάμεσων αποτελεσμάτων. Η ροή παράγεται μέσω του mflowgen και εκτελείται ως ακολουθία αριθμημένων σταδίων, από την αρχικοποίηση έως το τελικό στάδιο “finish”. Η ολοκλήρωση του τελικού σταδίου οδηγεί στην παραγωγή των τελικών αρχείων φυσικού σχεδιασμού, όπως αρχεία DEF και GDS, τα οποία αποτελούν το σημείο αναφοράς για τη μεταγενέστερη ανάλυση του layout.

Ο πυρήνας της ροής φυσικού σχεδιασμού βασίζεται στο OpenROAD, το οποίο αναλαμβάνει λειτουργίες όπως floorplanning, placement, clock tree synthesis (CTS), routing και τελική επεξεργασία. Σε κάθε στάδιο παράγονται ενδιάμεσα αρχεία που καταγράφουν την κατάσταση του σχεδίου, ενώ στο τελικό στάδιο δημιουργούνται τα αρχεία φυσικής υλοποίησης του κυκλώματος. Τα αρχεία αυτά αποτυπώνουν την τελική μορφή του σχεδίου και χρησιμοποιούνται ως βάση για περαιτέρω ανάλυση και επαλήθευση σε επίπεδο διάταξης. [3], [4]



Εικόνα 3.2 Build order της ροής mflowgen/OpenROAD για το s5378

Η Εικόνα 3.2 παρουσιάζει το build order της ροής mflowgen/OpenROAD, δηλαδή τη σειρά εκτέλεσης των σταδίων που οδηγούν από την αρχική περιγραφή του κυκλώματος μέχρι την τελική φυσική υλοποίηση. Η ροή εκτελείται ως ακολουθία βημάτων, όπως σύνθεση, floorplanning, placement, clock tree synthesis, routing και finish. Το build order είναι σημαντικό επειδή δείχνει τις εξαρτήσεις μεταξύ των σταδίων και τεκμηριώνει ότι η παραγωγή των τελικών αρχείων DEF και GDS γίνεται με επαναλήψιμο και οργανωμένο τρόπο.

Το αρχείο DEF περιλαμβάνει την αφαιρετική περιγραφή της τελικής τοποθέτησης των standard cells, τις συντεταγμένες των instances και βασικές πληροφορίες δρομολόγησης, ενώ το αρχείο GDS αποτελεί τη γεωμετρική αναπαράσταση της διάταξης σε επίπεδο πολυγώνων. Τα αρχεία αυτά αποτυπώνουν τη φυσική υλοποίηση του κυκλώματος πάνω σε standard cells της βιβλιοθήκης Nangate45. Παρότι η κύρια συνεισφορά της εργασίας δεν είναι η βελτιστοποίηση του φυσικού σχεδιασμού, η δυνατότητα πρόσβασης στα αρχεία DEF και GDS επιτρέπει τη συσχέτιση των λογικών δομών με συγκεκριμένα φυσικά στοιχεία του layout, γεγονός ιδιαίτερα σημαντικό σε μελέτες Σχεδίασης για Έλεγχο. [11], [12], [13], [14]

Ιδιαίτερη σημασία έχει η αναπαραγώγιμη φύση της ροής. Κάθε στάδιο εκτελείται με σαφώς ορισμένες εξαρτήσεις και παραμέτρους, και τα παραγόμενα αρχεία μπορούν να αναπαραχθούν

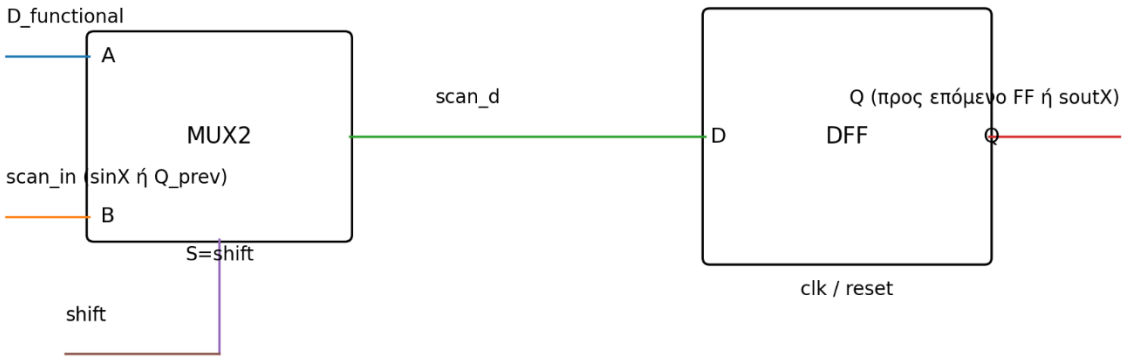
από οποιονδήποτε διαθέτει το ίδιο περιβάλλον εργαλείων. Με τον τρόπο αυτό, το layout που χρησιμοποιείται στην εργασία δεν αποτελεί αφηρημένη ή θεωρητική αναπαράσταση, αλλά αποτέλεσμα πλήρους και τεκμηριωμένης ροής φυσικού σχεδιασμού. Η ιδιότητα αυτή ενισχύει τη μεθοδολογική εγκυρότητα της εργασίας και επιτρέπει τη μελλοντική επέκταση ή επαλήθευση των αποτελεσμάτων.

Επιπλέον, στο πλαίσιο της ροής δημιουργούνται βοηθητικά αρχεία περιγραφής, τα οποία αποτυπώνουν τη σειρά εμφάνισης επιλεγμένων flip-flops στις scan chains μετά τα στάδια τοποθέτησης και δρομολόγησης. Τα αρχεία αυτά δεν αποτελούν απλώς ενδιάμεση πληροφορία της ροής, αλλά αξιοποιούνται ενεργά στα επόμενα στάδια της εργασίας. Συγκεκριμένα, λειτουργούν ως «γέφυρα» μεταξύ του φυσικού σχεδιασμού (OpenROAD) και των μετασχηματισμών σε επίπεδο netlist που υλοποιούνται με χρήση του εργαλείου Fault και προσαρμοσμένων scripts.

Με αυτόν τον τρόπο, η ροή mflowgen/OpenROAD εντάσσεται οργανικά στη συνολική μεθοδολογία της εργασίας. Δεν χρησιμοποιείται μόνο για την παραγωγή layout, αλλά παρέχει κρίσιμη πληροφορία που επηρεάζει την οργάνωση και επαλήθευση των scan chains. Η σύνδεση μεταξύ φυσικού και λογικού επιπέδου αποτελεί βασικό στοιχείο της προτεινόμενης προσέγγισης DfT και αναδεικνύει τη δυνατότητα υλοποίησης και μελέτης δομών ελέγχου σε πλήρως ανοικτό και αναπαραγώγιμο περιβάλλον σχεδίασης.

3.3 Το εργαλείο Fault για εισαγωγή scan chains και προσομοίωση

Για την ενσωμάτωση δομών Σχεδίασης για Έλεγχο (DfT) αξιοποιείται το εργαλείο ανοικτού κώδικα Fault, το οποίο υποστηρίζει λειτουργίες λογικής σύνθεσης, εισαγωγής scan δομών, προεπεξεργασίας για ATPG και βασικής προσομοίωσης. Στο πλαίσιο της παρούσας εργασίας, το Fault δεν χρησιμοποιείται αποσπασματικά, αλλά εντάσσεται σε μια συνεκτική και επαναλήψιμη ροή, η οποία ξεκινά από την περιγραφή του κυκλώματος σε επίπεδο RTL και οδηγεί σε netlist με ενσωματωμένες scan chains, κατάλληλο για περαιτέρω επεξεργασία και επαλήθευση. [8], [9]



Εικόνα 3.3 Αρχή scan εισαγωγής με MUX2 πριν από DFF (επιλογή functional/scan με σήμα shift)

Σε πρώτο στάδιο, το κύκλωμα σε μορφή Verilog υποβάλλεται σε λογική σύνθεση, ώστε να παραχθεί gate-level netlist συμβατό με τη βιβλιοθήκη standard cells Nangate45. Ακολουθεί το στάδιο cut, το οποίο αποτελεί κρίσιμο βήμα προετοιμασίας για τη διαδικασία αυτόματης παραγωγής διανυσμάτων ελέγχου (ATPG). Κατά το στάδιο αυτό, το αρχικά ακολουθιακό κύκλωμα μετασχηματίζεται σε μορφή περισσότερο κατάλληλη για έλεγχο, καθώς τα flip-flops απομονώνονται λειτουργικά από τη λογική ροή και αντιμετωπίζονται ως σημεία ελέγχου και παρατήρησης. Με τον τρόπο αυτό, η εσωτερική κατάσταση του κυκλώματος καθίσταται πιο άμεσα προσβάσιμη, ενώ η αναζήτηση κατάλληλων test vectors απλοποιείται ουσιωδώς, δεδομένου ότι το πρόβλημα μπορεί να προσεγγιστεί, σε σημαντικό βαθμό, ως πρόβλημα συνδυαστικού ελέγχου και όχι ως πλήρως ακολουθιακής ανάλυσης. Στο πλαίσιο αυτό, το εργαλείο δύναται να παράγει διανύσματα ελέγχου που ενεργοποιούν συγκεκριμένα μοντέλα σφαλμάτων, όπως τα stuck-at, και να καθιστούν τις επιπτώσεις τους παρατηρήσιμες σε κατάλληλα σημεία του κυκλώματος. [8], [9]

Το επόμενο καθοριστικό στάδιο είναι η εισαγωγή scan chains στο netlist. Η λειτουργία chain του Fault μετατρέπει τα συμβατικά flip-flops του κυκλώματος σε scan flip-flops και παράγει ένα scan-enhanced netlist, στο οποίο τα στοιχεία αυτά συνδέονται σειριακά, σχηματίζοντας αλυσίδες ελέγχου. Το παραγόμενο netlist αποτελεί τη βάση για τα επόμενα στάδια αναδιοργάνωσης και επαλήθευσης των scan δομών. [8], [9]

Η εκτέλεση των παραπάνω σταδίων πραγματοποιείται σε περιβάλλον Docker, με χρήση συγκεκριμένης έκδοσης του εργαλείου Fault. Η επιλογή αυτή διασφαλίζει σταθερότητα εκδόσεων και απομόνωση του περιβάλλοντος εκτέλεσης, ενισχύοντας την αναπαραγωγιμότητα των αποτελεσμάτων. Όπως και στην περίπτωση της ροής mflowgen/OpenROAD, η διαδικασία είναι πλήρως τεκμηριωμένη και δύναται να επαναληφθεί υπό ισοδύναμες συνθήκες.

Το scan-enhanced netlist που παράγεται από το Fault περιλαμβάνει τις απαραίτητες scan δομές, χωρίς όμως η οργάνωσή τους να είναι κατ' ανάγκη πλήρως προσαρμοσμένη στις απαιτήσεις της παρούσας μελέτης. Ειδικότερα, οι αλυσίδες δεν είναι πάντοτε ρητά εκτεθειμένες μέσω πολλαπλών ανεξάρτητων εισόδων και εξόδων, ούτε οργανώνονται υποχρεωτικά με βάση τη φυσική πληροφορία που προκύπτει από το layout. Για τον λόγο αυτό, το netlist αυτό αντιμετωπίζεται ως ενδιάμεσο στάδιο και όχι ως τελικό αποτέλεσμα.

Στο επόμενο βήμα της μεθοδολογίας, το παραγόμενο netlist υφίσταται μετασχηματισμό μέσω προσαρμοσμένων scripts, τα οποία αναδιοργανώνουν τα flip-flops σε διακριτές scan chains. Η πληροφορία που αξιοποιείται για την αναδιάταξη αυτή προέρχεται από τη ροή φυσικού σχεδιασμού, γεγονός που αναδεικνύει τη στενή σύνδεση μεταξύ των εργαλείων DfT και της φυσικής υλοποίησης στο πλαίσιο της εργασίας.

Τέλος, το αναδομημένο netlist υποβάλλεται σε προσομοίωση με χρήση iverilog, επίσης μέσα από το ίδιο περιβάλλον Docker. Μέσω ειδικά σχεδιασμένων προγράμματα δοκιμής επαληθεύεται ότι οι scan chains λειτουργούν ορθά σε κατάσταση shift, ότι τα σήματα εισόδου μεταδίδονται σειριακά κατά μήκος των flip-flops και ότι οι αντίστοιχες έξοδοι αντανακλούν ορθά την εσωτερική κατάσταση κάθε αλυσίδας. Με τον τρόπο αυτό ολοκληρώνεται ένας πλήρης κύκλος DfT, από την εισαγωγή των scan δομών έως τη λειτουργική επαλήθευσή τους σε επίπεδο netlist.

Η συνδυασμένη χρήση των εργαλείων Fault και mflowgen/OpenROAD καθιστά εφικτή τη μελέτη δομών Σχεδίασης για Έλεγχο σε ένα πλήρως ανοικτό περιβάλλον σχεδίασης, στο οποίο το λογικό και το φυσικό επίπεδο συνδέονται άμεσα. Το Fault παρέχει το αρχικό scan-enhanced netlist, ενώ η περαιτέρω επεξεργασία και επαλήθευση ενσωματώνει πληροφορία από το layout, διαμορφώνοντας μια συνεκτική και επαναλήψιμη ροή DfT.

3.4 Περιβάλλον ανάπτυξης και βοηθητικά scripts

Η υλοποίηση της προτεινόμενης ροής Σχεδίασης για Έλεγχο πραγματοποιήθηκε σε περιβάλλον Linux μέσω Windows Subsystem for Linux (WSL), το οποίο παρέχει συμβατότητα με τα εργαλεία ανοικτού κώδικα που χρησιμοποιούνται στην εργασία. Η επιλογή αυτού του περιβάλλοντος επιτρέπει την ομοιογενή εκτέλεση τόσο της ροής φυσικού σχεδιασμού (mflowgen/OpenROAD) όσο και της ροής εισαγωγής και επαλήθευσης scan δομών (Fault), χωρίς ασυμβατότητες μεταξύ λειτουργικών συστημάτων.

Κεντρικό στοιχείο της υποδομής αποτελεί η χρήση Docker containers για την εκτέλεση του εργαλείου Fault και των σχετικών προσομοιώσεων. Με τον τρόπο αυτό διασφαλίζεται ότι οι εκδόσεις των εργαλείων, των βιβλιοθηκών standard cells και των προσομοιωτών παραμένουν σταθερές καθ' όλη τη διάρκεια της εργασίας. Η απομόνωση του περιβάλλοντος εκτέλεσης ενισχύει τη δυνατότητα αναπαραγωγής των αποτελεσμάτων και μειώνει τον κίνδυνο σφαλμάτων που οφείλονται σε διαφοροποιήσεις εκδόσεων ή εξαρτήσεων.

Πέρα από τα βασικά εργαλεία της ροής, καθοριστικό ρόλο διαδραματίζουν τα βοηθητικά scripts που αναπτύχθηκαν στο πλαίσιο της εργασίας. Το σημαντικότερο από αυτά είναι το script rewrite_scan_chains_from_uuf.py, το οποίο υλοποιεί τον μετασχηματισμό του αρχικού scan-enhanced netlist σε netlist με τέσσερις σαφώς ορισμένες scan chains. Το script αυτό αναλύει το gate-level netlist, εντοπίζει τα flip–flops του block **uuf**, τα συσχετίζει με τις αλυσίδες που περιγράφονται στα αρχεία uuf_chain1.txt έως uuf_chain4.txt και εισάγει τους απαραίτητους πολυπλέκτες και τα νέα σήματα scan. Το αποτέλεσμα είναι ένα τροποποιημένο netlist στο οποίο οι αλυσίδες είναι ρητά ορισμένες με εισόδους sin0–sin3 και εξόδους sout0–sout3.

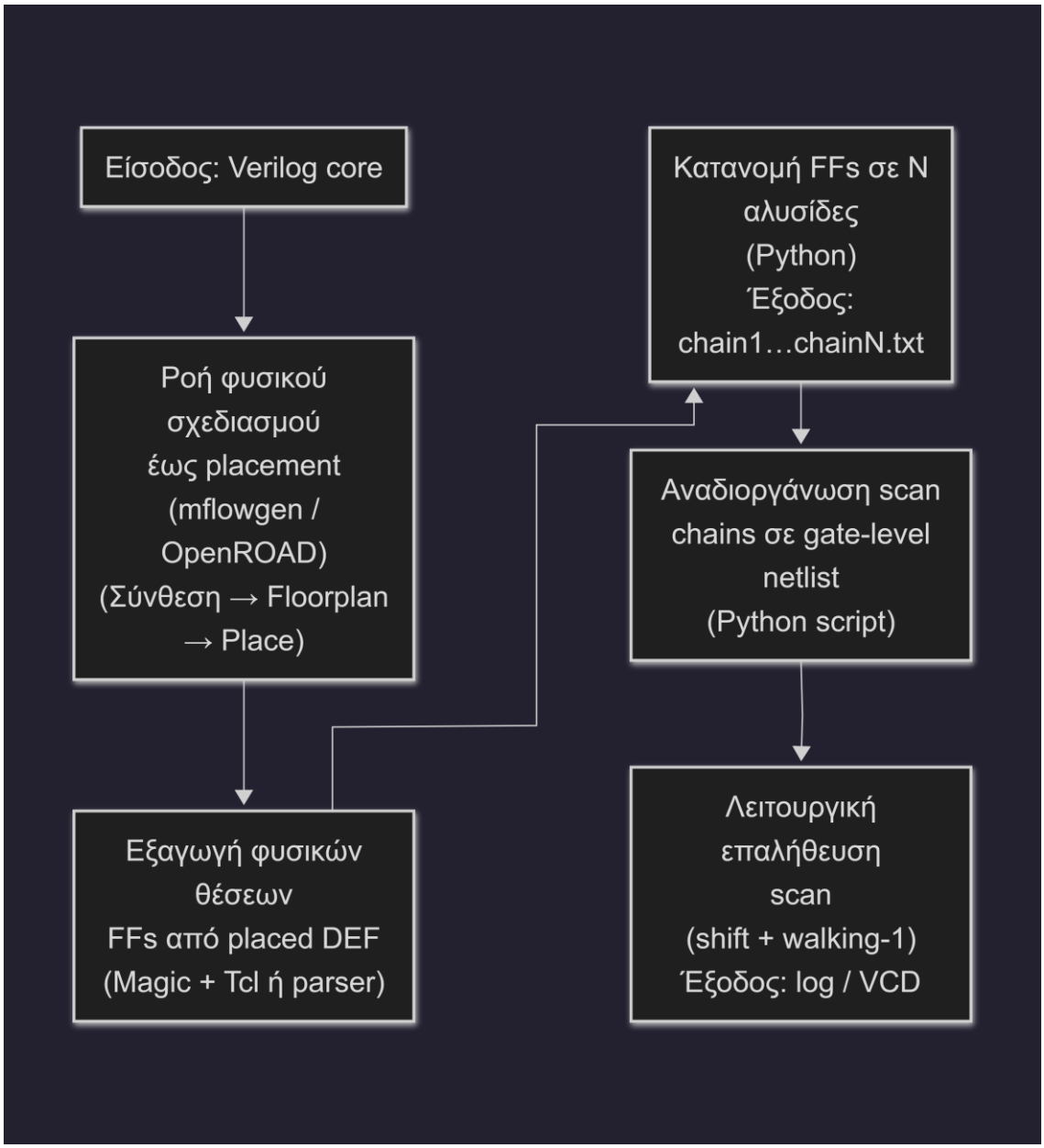
Παράλληλα, χρησιμοποιείται βοηθητικό script για τον διαχωρισμό των flip-flops σε πολλαπλές αλυσίδες με βάση τη φυσική τους θέση στο layout. Το script αξιοποιεί αρχείο CSV που εξάγεται από το DEF και περιέχει τα ονόματα των instances, τον τύπο των cells και τις συντεταγμένες τοποθέτησής τους. Σε πρώτο στάδιο πραγματοποιείται φιλτράρισμα, ώστε να διατηρηθούν μόνο τα flip-flops που είναι κατάλληλα για ένταξη στις scan chains, ενώ απομακρύνονται βοηθητικά στοιχεία που δεν συμμετέχουν στη δομή ελέγχου. Στη συνέχεια, τα επιλεγμένα flip-flops ταξινομούνται ως προς τη συντεταγμένη x, ώστε να ληφθεί υπόψη η χωρική τους κατανομή κατά μήκος του ολοκληρωμένου κυκλώματος. Η ταξινομημένη λίστα διαιρείται έπειτα σε επιμέρους ομάδες περίπου ίσου μεγέθους, οι οποίες αντιστοιχίζονται στις

τελικές scan chains. Η διαδικασία αυτή επιτρέπει την ενσωμάτωση της φυσικής πληροφορίας τοποθέτησης στη λογική οργάνωση των αλυσίδων, ενισχύοντας τη συνέπεια μεταξύ netlist και layout.

Για την επαλήθευση της λειτουργίας των scan chains αναπτύχθηκαν ειδικά προγράμματα δοκιμής σε Verilog, τα οποία προσομοιώνονται με χρήση iverilog μέσα από το ίδιο περιβάλλον Docker. Τα προγράμματα δοκιμής αυτά υλοποιούν διαδικασίες reset, flush και δοκιμές τύπου “walking–1”, επιτρέποντας τη συστηματική επιβεβαίωση της σειριακής συνδεσμολογίας των αλυσίδων. Τα αποτελέσματα των προσομοιώσεων καταγράφονται σε αρχεία log και VCD, παρέχοντας τεκμηριωμένη απόδειξη της ορθής λειτουργίας. [10]

Συνολικά, το περιβάλλον ανάπτυξης και τα βοηθητικά scripts δεν αποτελούν απλώς τεχνικές λεπτομέρειες υλοποίησης, αλλά οργανικό μέρος της μεθοδολογίας της εργασίας. Η συνδυασμένη χρήση mflowgen, OpenROAD, Fault, Docker και προσαρμοσμένων Python scripts διαμορφώνει μία ολοκληρωμένη και επαναλήψιμη ροή DfT, η οποία συνδέει το λογικό, το φυσικό και το επίπεδο επαλήθευσης σε ένα ενιαίο πλαίσιο. Με αυτόν τον τρόπο, η εργασία δεν περιορίζεται σε θεωρητική ανάλυση των scan chains, αλλά παρουσιάζει μια πλήρη και τεκμηριωμένη υλοποίηση τους σε περιβάλλον ανοικτού κώδικα.

Η ολοκληρωμένη ροή που παρουσιάστηκε στο παρόν κεφάλαιο συνθέτει όλα τα επιμέρους εργαλεία και στάδια που απαιτούνται για την υλοποίηση και ανάλυση δομών Σχεδίασης για Έλεγχο σε περιβάλλον ανοικτού κώδικα. Από τη φυσική υλοποίηση μέσω mflowgen και OpenROAD, μέχρι την εισαγωγή scan δομών με το Fault και την αξιοποίηση προσαρμοσμένων scripts για αναδιοργάνωση και επαλήθευση των αλυσίδων, διαμορφώνεται ένα ενιαίο και επαναλήψιμο πλαίσιο DfT. Στο επόμενο κεφάλαιο, η έμφαση μετατοπίζεται από την περιγραφή της ροής στην εις βάθος ανάλυση της αναδιοργάνωσης των scan chains, με στόχο τη συστηματική τεκμηρίωση της δομικής και λειτουργικής ορθότητάς τους σε επίπεδο netlist και προσομοίωσης.



Εικόνα 3.4 Συνολική ροή από Verilog s5378 έως layout, ανασύνδεση scan chains και επαλήθευσης

4. Αναδιοργάνωση των scan chains και επαλήθευση λειτουργίας

Στο προηγούμενο κεφάλαιο παρουσιάστηκε η συνολική ροή υλοποίησης του κυκλώματος s5378 σε περιβάλλον ανοικτού κώδικα, από το στάδιο της φυσικής υλοποίησης έως την εισαγωγή δομών Σχεδίασης για Έλεγχο μέσω του εργαλείου Fault. Η ροή αυτή παρείχε ένα scan-enhanced netlist καθώς και φυσική πληροφορία διάταξης των flip–flops στο layout. Ωστόσο, η αρχική μορφή των scan chains δεν ανταποκρίνεται πλήρως στους στόχους της παρούσας μελέτης, οι οποίοι περιλαμβάνουν την ύπαρξη πολλαπλών ανεξάρτητων αλυσίδων, ρητά ορισμένων στο module interface και συνεπών με τη φυσική τοποθέτηση των στοιχείων.

Το παρόν κεφάλαιο επικεντρώνεται στη συστηματική αναδιοργάνωση των scan chains και στην τεκμηριωμένη επαλήθευση της ορθής λειτουργίας τους. Η ανάλυση πραγματοποιείται τόσο σε δομικό επίπεδο (netlist transformation) όσο και σε λειτουργικό επίπεδο (προσομοίωση με ειδικά προγράμματα δοκιμής), διαμορφώνοντας ένα συνεκτικό πλαίσιο ελέγχου της υλοποίησης DfT.

4.1 Στόχος και συνοπτική προσέγγιση

Η αφετηρία της διαδικασίας αναδιοργάνωσης είναι το netlist s5378.chained.v που παράγεται από το εργαλείο Fault. Το netlist αυτό περιλαμβάνει scan flip–flops και βασική σειριακή δομή ελέγχου, ωστόσο η συνδεσμολογία των αλυσίδων δεν είναι διαμορφωμένη σύμφωνα με τις απαιτήσεις της παρούσας μελέτης. Συγκεκριμένα, οι scan δομές εμφανίζονται ως ενιαία αλυσίδα και δεν είναι ρητά εκτεθειμένες σε πολλαπλές ανεξάρτητες εισόδους και εξόδους στο επίπεδο του module. [8], [9]

Η ανάγκη για αναδιοργάνωση των scan chains δεν προκύπτει μόνο από σχεδιαστική επιλογή, αλλά από τη μεθοδολογική απαίτηση ευθυγράμμισης του λογικού επιπέδου με το φυσικό επίπεδο υλοποίησης. Σε κλασικές ροές DfT, οι scan chains ορίζονται αποκλειστικά σε επίπεδο netlist και αντιμετωπίζονται ανεξάρτητα από τη φυσική τοποθέτηση των flip–flops. Στην παρούσα εργασία, αντιθέτως, επιχειρείται η ενσωμάτωση πληροφορίας από το layout στη λογική οργάνωση των αλυσίδων, διαμορφώνοντας μια πιο συνεκτική και ρεαλιστική προσέγγιση σχεδίασης για έλεγχο.

Αλγόριθμος 4.1: Διαχωρισμός flip-flops σε N scan chains με βάση τη συντεταγμένη x και φίλτρο TAP cells.

```
Είσοδος: CSV με πεδία inst_name, cell_type, x_um, y_um και πλήθος αλυσίδων N
Έξοδος: αρχεία uuf_chain1.txt ... uuf_chainN.txt

φόρτωσε τις εγγραφές του CSV
DFFs <- κενή λίστα
Για κάθε εγγραφή του CSV:
   αν το cell_type δεν ξεκινά από "DFF": συνέχισε στην επόμενη εγγραφή
   αν το inst_name ή το cell_type περιέχει "TAP": αγνόησε την εγγραφή
   διάβασε τις συντεταγμένες x_um και y_um
   πρόσθεσε το flip-flop στη λίστα DFFs

ταξινόμησε τη λίστα DFFs κατά αύξουσα συντεταγμένη x_um
per_chain <- ceil(πλήθος(DFFs) / N)
Για c από 0 μέχρι N-1:
   start <- c * per_chain
   end <- min((c + 1) * per_chain, πλήθος(DFFs))
   chunk <- DFFs[start : end]
   γράψε τα inst_name του chunk στο αρχείο της αντίστοιχης αλυσίδας
```

Παράλληλα, από τη ροή φυσικού σχεδιασμού αντλείται πληροφορία για τα τοποθετημένα instances του layout, συμπεριλαμβανομένου του τύπου του κάθε κελιού και των φυσικών συντεταγμένων του. Η διαδικασία ξεκινά με την επιλογή μόνο των κελιών που αντιστοιχούν σε flip-flops, καθώς αυτά είναι τα στοιχεία που μπορούν να συμμετάσχουν στις scan chains. Στη συνέχεια εφαρμόζεται φίλτρο εξαίρεσης για TAP-related cells, ώστε να απομακρυνθούν στοιχεία τεχνολογικής υποστήριξης τα οποία δεν αποτελούν μέρος της λειτουργικής scan δομής. Αφού ολοκληρωθεί το φιλτράρισμα, τα flip-flops ταξινομούνται ως προς τη συντεταγμένη x, με στόχο η ομαδοποίησή τους να αντανακλά τη χωρική τους κατανομή στο layout. Η τελική λίστα διαιρείται σε N ομάδες σχεδόν ίσου πλήθους, οι οποίες ορίζουν τις επιμέρους scan chains. Η προσέγγιση αυτή επιτρέπει την ενσωμάτωση φυσικής πληροφορίας στη λογική οργάνωση των αλυσίδων και ενισχύει τη συνέπεια μεταξύ layout και netlist.

Στόχος του παρόντος κεφαλαίου είναι η μετατροπή της αρχικής scan δομής σε τέσσερις διακριτές και σαφώς ορισμένες scan chains, καθεμία με ανεξάρτητη είσοδο και έξοδο, καθώς και η τεκμηρίωση της ορθής λειτουργίας τους μέσω προσομοίωσης. Η προσέγγιση που ακολουθείται συνδυάζει αυτοματοποιημένο μετασχηματισμό του netlist με δομική και λειτουργική επαλήθευση, διασφαλίζοντας ότι η τελική μορφή των αλυσίδων είναι συνεπής, πλήρης και ελεγχόμενη.

Στις επόμενες ενότητες παρουσιάζεται αναλυτικά ο μηχανισμός μετασχηματισμού του netlist και τα κριτήρια με τα οποία αξιολογείται η επιτυχία της αναδιοργάνωσης.

4.2 Περιγραφή του script rewrite_scan_chains.py

Σε πρώτο στάδιο, το script φορτώνει τα αρχεία περιγραφής των scan chains και κατασκευάζει εσωτερικές αντιστοιχίσεις που περιγράφουν τη δομή τους. Όπως φαίνεται στον Αλγόριθμο 4.2, για κάθε flip-flop που εμφανίζεται στις λίστες των αλυσίδων και εντοπίζεται στο netlist, καταγράφεται αφενός η αλυσίδα στην οποία ανήκει και αφετέρου το αμέσως προηγούμενο flip-flop της ίδιας αλυσίδας. Παράλληλα, προσδιορίζεται και το τελευταίο flip-flop κάθε αλυσίδας, το οποίο θα χρησιμοποιηθεί αργότερα για τον ορισμό της αντίστοιχης scan εξόδου. Με τον τρόπο αυτό δημιουργείται μια ενδιάμεση δομή αντιστοίχισης ανάμεσα στις λίστες των αλυσίδων και στα πραγματικά instances του netlist, η οποία αποτελεί τη βάση για το επόμενο στάδιο της επανασύνδεσης.

Αλγόριθμος 4.2: Δημιουργία αντιστοιχίσεων των flip-flops στις scan chains.

```
Είσοδος: chains = λίστες ονομάτων FF ανά αλυσίδα, dffs = FFs που βρέθηκαν στο
netlist
Έξοδος: chain_of, prev_of, last_of

chain_of <- κενό λεξικό        // σε ποια αλυσίδα ανήκει κάθε FF
prev_of  <- κενό λεξικό        // προηγούμενο FF στην ίδια αλυσίδα
last_of  <- κενό λεξικό        // τελευταίο FF κάθε αλυσίδας

Για κάθε αλυσίδα με δείκτη cidx:
   prev <- KENO
   Για κάθε όνομα FF στη λίστα της αλυσίδας:
      inst <- καθαρισμένο όνομα FF
      αν το inst δεν υπάρχει στο netlist: συνέχισε με το επόμενο όνομα
      chain_of[inst] <- cidx
      prev_of[inst]  <- prev
      prev <- inst
   αν prev δεν είναι KENO: last_of[cidx] <- prev

επέστρεψε chain_of, prev_of και last_of
```

Στη συνέχεια, το script διαβάζει το αρχικό netlist s5378.chained.v γραμμή προς γραμμή και εντοπίζει όλα τα instances τύπου DFFS_X1 που ανήκουν στο block __uuf__. Σε κάθε εκτέλεση αναφέρεται στο τερματικό ο αριθμός των flip-flops που βρέθηκαν· χαρακτηριστικά, στην

εργασία αυτή εμφανίζεται μήνυμα του τύπου «Βρέθηκαν συνολικά 162 uuf DFF instances στο netlist». Το πλήθος αυτό πρέπει να συμφωνεί με το συνολικό πλήθος ονομάτων που φορτώθηκαν από τα αρχεία περιγραφής των αλυσίδων, κάτι που επιβεβαιώνεται επίσης κατά την εκτέλεση.

Αφού επιβεβαιωθεί η αντιστοίχιση, το script περνά στο στάδιο της επανασύνδεσης. Για κάθε flip–flop μιας αλυσίδας, εξάγεται το αρχικό σήμα που οδηγεί την είσοδό του D και το όνομα του σήματος εξόδου Q. Αντί η είσοδος D να συνδέεται απευθείας στο αρχικό σήμα, δημιουργείται ένα νέο ενδιάμεσο σήμα τύπου scan_d___uuf____XXXX_ και εισάγεται μπροστά από το flip–flop ένας πολυπλέκτης MUX2_X1 με όνομα scanmux___uuf__.... Ο πολυπλέκτης αυτός έχει είσοδο A το αρχικό "λειτουργικό" σήμα, είσοδο B το "scan" σήμα της αλυσίδας (είτε sinX είτε την έξοδο Q του προηγούμενου flip–flop της ίδιας αλυσίδας) και επιλογή S το global σήμα shift. Η έξοδος Z του πολυπλέκτη οδηγεί το νέο σήμα scan_d___uuf____XXXX_, το οποίο πλέον συνδέεται στην είσοδο D του flip–flop.

Αλγόριθμος 4.3: Αυτόματη εισαγωγή MUX2 πριν από κάθε DFF και σύνδεση του scan_in ανά chain.

```
Είσοδος: αρχικό netlist, dffs, chain_of, prev_of, last_of
Έξοδος: τροποποιημένο netlist με sin0..sin3 και sout0..sout3

Για κάθε DFF block του __uuf__ μέσα στο netlist:
   inst <- όνομα του τρέχοντος DFF
   αν το inst δεν ανήκει σε κάποια scan chain:
      κράτησε το block αμετάβλητο και συνέχισε με το επόμενο DFF

   chain_idx <- chain_of[inst]
   prev_inst <- prev_of[inst]
   αν prev_inst είναι KENO:
      scan_in <- sin[chain_idx]
   αλλιώς:
      scan_in <- Q(prev_inst)

   functional_D <- αρχική είσοδος D του DFF
   δημιούργησε νέο wire scan_d για το συγκεκριμένο DFF
   εισήγαγε MUX2 με A=functional_D, B=scan_in, S=shift, Z=scan_d
   ξαναγράψε το DFF ώστε η θύρα .D να οδηγείται από scan_d

ενημέρωσε το module header με sin0..sin3 και sout0..sout3
Για κάθε αλυσίδα c:
   σύνδεσε sout[c] <- Q(last_of[c])
```

Όπως φαίνεται στον Αλγόριθμο 4.3, το στάδιο της επανασύνδεσης υλοποιείται μέσω συνάρτησης που ξαναγράφει επιλεκτικά τα τμήματα του netlist τα οποία αντιστοιχούν στα flip-flops των scan chains. Για κάθε flip-flop που έχει αντιστοιχιστεί σε κάποια αλυσίδα, το script προσδιορίζει πρώτα σε ποια αλυσίδα ανήκει και ποιο είναι το προηγούμενο στοιχείο της ίδιας ακολουθίας. Με βάση αυτή την πληροφορία ορίζεται το κατάλληλο scan input: για το πρώτο flip-flop της αλυσίδας χρησιμοποιείται η εξωτερική είσοδος sinX, ενώ για κάθε επόμενο χρησιμοποιείται η έξοδος Q του προηγούμενου flip-flop. Στη συνέχεια δημιουργείται ένα νέο ενδιάμεσο σήμα scan και εισάγεται πριν από το DFF ένας πολυπλέκτης MUX2_X1, ο οποίος επιλέγει μεταξύ του αρχικού λειτουργικού σήματος και του scan σήματος με έλεγχο από το shift. Με τον τρόπο αυτό, το script δεν αντικαθιστά απλώς συνδέσεις, αλλά αναδομεί συστηματικά κάθε flip-flop ώστε να εντάσσεται στη νέα scan chain χωρίς να χάνεται η κανονική λειτουργία του κυκλώματος.

Για το πρώτο flip–flop κάθε αλυσίδας, η είσοδος B του πολυπλέκτη συνδέεται στη θύρα sinX που αντιστοιχεί στην αλυσίδα X. Με αυτόν τον τρόπο, όταν το shift τίθεται σε λογικό ‘1’, τα bits που εφαρμόζονται στην είσοδο sinX “σπρώχνονται” σειριακά από το πρώτο flip–flop προς τα υπόλοιπα της αλυσίδας. Για όλα τα επόμενα flip–flops της ίδιας αλυσίδας, η είσοδος B συνδέεται στο Q του προηγούμενου flip–flop. Έτσι, σχηματίζεται μια καθαρή σειριακή αλυσίδα μετατόπισης, χωρίς να αλλοιώνεται η λειτουργική συνδεσμολογία όταν το shift είναι ‘0’.

Παράλληλα, το script ενημερώνει το module header του s5378 ώστε να συμπεριλάβει τις νέες scan θύρες. Εισάγονται δηλώσεις input sin0; … input sin3; και output sout0; … output sout3;, καθώς και οι αντίστοιχες δηλώσεις για τα ενδιάμεσα σήματα. Στο τέλος του module, προστίθενται αναθέσεις τύπου assign sout0 = __uuf__.II851;, assign sout1 = __uuf__.n2037gat; κ.ο.κ., οι οποίες συνδέουν κάθε έξοδο soutX με το Q του τελευταίου flip–flop της αντίστοιχης αλυσίδας, όπως αυτό προκύπτει από την ανάλυση των netlists.

Η υλοποίηση του script έγινε με προσοχή ώστε να διατηρηθεί η συντακτική ορθότητα του Verilog netlist. Σε παλαιότερες ενδιάμεσες εκδόσεις χρησιμοποιήθηκαν πιο “επιθετικές” αντικαταστάσεις κειμένου, οι οποίες οδηγούσαν σε σφάλματα κατά τη σύνταξη (π.χ. διπλές παρενθέσεις μετά από instances ή ελλιπείς δηλώσεις wire). Η τρέχουσα εκδοχή αναλύει ιεραρχικά το netlist, ξαναγράφει μόνο τα συγκεκριμένα τμήματα που αφορούν τα DFFS_X1 του __uuf__ και προσθέτει τις νέες δηλώσεις με ελεγχόμενο τρόπο, όπως επιβεβαιώνεται από τη δυνατότητα επιτυχούς σύνθεσης με iverilog χωρίς συντακτικά σφάλματα.

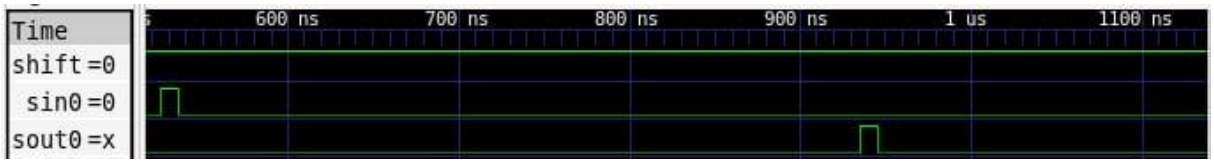
4.3 Προγράμματα δοκιμής και επαλήθευση με walking–1

Για να επαληθευτεί ότι οι τέσσερις scan chains στο s5378.chained.multi.v είναι πράγματι σωστά συνδεδεμένες και λειτουργικές, αναπτύχθηκαν προγράμματα δοκιμής σε Verilog που ασκούν τις εισόδους sin0–sin3 και παρατηρούν τις εξόδους sout0–sout3 υπό ελεγχόμενες συνθήκες. Η επαλήθευση γίνεται σε δύο επίπεδα: βασική λειτουργική δοκιμή και συστηματική δοκιμή τύπου walking–1.

Η βασική λειτουργική δοκιμή υλοποιείται στο πρόγραμμα δοκιμής scan_tb_s5378_simple.v. Σε αυτή την περίπτωση το κύκλωμα s5378 συνδέεται με το netlist βιβλιοθήκης Tech/nangate45/cells.v και προσομοιώνεται με το iverilog μέσα από το Docker περιβάλλον του Fault. Το πρόγραμμα δοκιμής εφαρμόζει μια απλή ακολουθία στο σήμα shift και στις εισόδους sin0–sin3, ενώ παράλληλα καταγράφει τα σήματα του κυκλώματος σε αρχείο VCD (scan_tb_s5378_simple.vcd). Η ανάλυση του VCD επιβεβαιώνει ότι, όταν το shift είναι ενεργό, οι τιμές των sinX διαδίδονται προς τις αντίστοιχες soutX με τον αναμενόμενο τρόπο. [10]

Για πιο αυστηρή επαλήθευση σχεδιάστηκε ειδικό πρόγραμμα δοκιμής scan_tb_s5378_chainwalk.v, το οποίο υλοποιεί για κάθε αλυσίδα μια διαδικασία “flush + walking–1”. Η διαδικασία για κάθε chain X περιλαμβάνει τρεις φάσεις. Στην πρώτη φάση, ενεργοποιείται το reset του κυκλώματος ώστε να καθαριστεί η εσωτερική κατάσταση. Στη δεύτερη φάση (“flush”), το shift τίθεται σε ‘1’ και στην είσοδο sinX εφαρμόζονται διαδοχικά λογικά ‘0’ για αριθμό κύκλων μεγαλύτερο από το μήκος της αλυσίδας. Στο τέλος αυτής της φάσης, η soutX θα πρέπει να έχει σταθεροποιηθεί σε ‘0’, κάτι που επιβεβαιώνεται από τις καταγραφές στο log.

Στην τρίτη φάση εφαρμόζεται το ίδιο το walking–1 test. Διατηρώντας το shift στο ‘1’, στην είσοδο sinX εισάγεται μία μόνο παλμό ‘1’ (ένα λογικό ‘1’ ακολουθούμενο από ακολουθία ‘0’). Καθώς η αλυσίδα αποτελείται από N flip–flops, το μοναδικό αυτό ‘1’ θα πρέπει, θεωρητικά, να εμφανιστεί στην έξοδο soutX ακριβώς μία φορά, μετά από N κύκλους, και σε καμία άλλη χρονική στιγμή. Θεωρητικά, μια σωστά συνδεδεμένη σειριακή αλυσίδα μήκους N μπορεί να μοντελοποιηθεί ως γραμμικό shift register, στο οποίο η είσοδος u(t) εμφανίζεται στην έξοδο μετά από ακριβώς N χρονικά βήματα. Η ιδιότητα αυτή χρησιμοποιείται ως κριτήριο επαλήθευσης, καθώς οποιαδήποτε απόκλιση (π.χ. εμφάνιση περισσότερων του ενός λογικών ‘1’ ή λανθασμένος αριθμός κύκλων καθυστέρησης) υποδηλώνει σφάλμα συνδεσμολογίας. [1]



Εικόνα 4.1 Αποτέλεσμα δοκιμής walking-1 για τη scan chain 0: ο παλμός που εφαρμόζεται στην είσοδο sin0 εμφανίζεται στην έξοδο sout0 μετά από την αναμενόμενη καθυστέρηση της αλυσίδας.

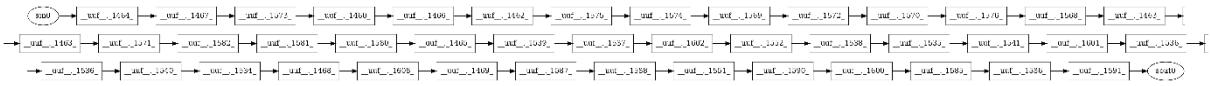
Το πρόγραμμα δοκιμής μετρά τον αριθμό των κύκλων στους οποίους το soutX είναι ‘1’ και καταγράφει τον συνολικό αριθμό “ones_seen” για κάθε chain.

Η προσομοίωση εκτελείται ξανά με iverilog μέσα από το Docker image του Fault, και η έξοδος του προγράμματος δοκιμής αποθηκεύεται σε αρχείο κειμένου scan_chainwalk_log.txt. Σε αυτό, για κάθε αλυσίδα εμφανίζονται μηνύματα της μορφής:

CHAIN0 summary: ones_seen = 1 (expected 1) CHAIN0 WALKING-1: PASS

Αντίστοιχα αποτελέσματα παρατηρούνται και για τις υπόλοιπες αλυσίδες, με τα μήκη 41, 41, 40 και 40 κύκλων αντίστοιχα. Με άλλα λόγια, για κάθε αλυσίδα το walking–1 test επιβεβαιώνει ότι υπάρχει ακριβώς μία και μοναδική στιγμή όπου η έξοδος soutX γίνεται ‘1’ μετά τη φάση flush, γεγονός που αποτελεί ισχυρή ένδειξη ότι:

α) όλα τα flip–flops της αλυσίδας είναι συνδεδεμένα σειριακά μεταξύ τους,
β) δεν υπάρχουν “βραχυκυκλώματα” ή παράλληλες συνδέσεις που θα οδηγούσαν σε πολλαπλά ‘1’,
γ) η είσοδος sinX και η έξοδος soutX αντιστοιχούν πράγματι στην αρχή και στο τέλος της ίδιας αλυσίδας.

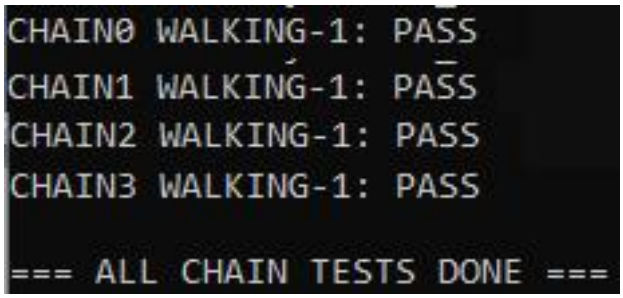


Εικόνα 4.2 Οπτικοποίηση της scan chain 0 στο layout

Έτσι, το scan_tb_s5378_chainwalk.v λειτουργεί ως αυτοτελής απόδειξη ορθής λειτουργίας των scan chains στο επίπεδο του netlist, βασισμένη αποκλειστικά σε προσομοίωση και σε μετρήσιμα κριτήρια.

Ως πρακτικό κριτήριο επιτυχίας της επαλήθευσης χρησιμοποιείται η σύνοψη που καταγράφεται στο αρχείο log της προσομοίωσης. Για κάθε αλυσίδα αναμένεται να εμφανίζεται ένδειξη

επιτυχίας (PASS) σε συνδυασμό με ακριβώς μία ανίχνευση λογικού ‘1’ στην έξοδο (ones_seen = 1). Η ταυτόχρονη επιτυχία και των τεσσάρων αλυσίδων αποτελεί συνοπτική αλλά σαφή τεκμηρίωση ότι οι scan είσοδοι και έξοδοι αντιστοιχούν στα σωστά άκρα των αλυσίδων και ότι δεν υπάρχουν ανεπιθύμητες διασυνδέσεις μεταξύ τους.



Εικόνα 4.3 Επιτυχής επαλήθευση walking–1 της scan chain

4.4 Αξιολόγηση testability με ATPG / stuck-at fault coverage (Fault)

Αφού επιβεβαιώθηκε η συνέπεια λογικού–φυσικού επιπέδου μέσω DEF/CSV, συμπληρώνεται η αξιολόγηση με ένα βασικό ποσοτικό μέτρο testability (fault coverage) πριν και μετά την εισαγωγή scan δομών.

Παρότι ο κύριος στόχος του παρόντος κεφαλαίου είναι η σύνδεση των scan chains με το φυσικό layout, είναι χρήσιμο να συμπληρωθεί η αξιολόγηση με ένα βασικό ποσοτικό κριτήριο testability. Για τον σκοπό αυτό αξιοποιήθηκε το εργαλείο Fault, το οποίο υποστηρίζει παραγωγή test vectors (ATPG), fault simulation και compaction στο μοντέλο stuck-at. Η αξιολόγηση εκτελέστηκε σε δύο εκδόσεις του κυκλώματος: (α) στο baseline netlist πριν την εισαγωγή scan δομών και (β) στο netlist μετά την εισαγωγή scan chains/scan logic. Και στις δύο περιπτώσεις χρησιμοποιήθηκαν οι ίδιες παράμετροι παραγωγής διανυσμάτων (v=1000, r=50, target coverage m=95%, ceiling=1000), ώστε τα αποτελέσματα να είναι συγκρίσιμα. [8], [9]

Τα αποτελέσματα συνοψίζονται στον Πίνακα 1 Παρατηρείται ότι μετά την εισαγωγή scan δομών το stuck-at fault coverage αυξάνεται σημαντικά, από 34.73% σε 69.32%. Παράλληλα, αυξάνεται και ο αριθμός των fault sites (από 3543 σε 4782), γεγονός αναμενόμενο λόγω της πρόσθετης λογικής και των επιπλέον ports/διασυνδέσεων που εισάγονται από την DfT τροποποίηση. Παρά το αυξημένο “fault universe”, η κάλυψη βελτιώνεται αισθητά, στοιχείο

που υποστηρίζει ότι η εισαγωγή scan δομών δεν είναι μόνο λειτουργικά ορθή (Κεφ. 4) και χωρικά συνεπής (Κεφ. 5.3), αλλά και ωφέλιμη ως προς τη βασική δυνατότητα ελέγχου.

Για κάθε έκδοση του netlist χρησιμοποιήθηκαν οι ίδιες παράμετροι παραγωγής διανυσμάτων (v=1000, r=50, m=95%, ceiling=1000). Η εξαγωγή των μετρικών προκύπτει από το συνοπτικό report του Fault (fault sites, τελικό coverage) και το στάδιο compaction (raw vs compacted test vectors)

Περίπτωση	**Fault sites (gates/ports)**	**Coverage (%)**	**Raw TVs**	**Compacted TVs**
Πριν (baseline)	3543 (949 / 86)	**34.730453**	1000	**20**
Μετά (scan)	4782 (1239 / 421)	**69.322464**	1000	**19**

Πίνακας 1. Σύγκριση stuck-at fault coverage πριν/μετά scan insertion

Οπως φαίνεται στον Πίνακα, η κάλυψη αυξήθηκε κατά 34.59 ποσοστιαίες μονάδες (από 34.73% σε 69.32%), παρότι ο αριθμός των fault sites αυξήθηκε (3543 → 4782) λόγω πρόσθετης scan λογικής και επιπλέον ports. Το αποτέλεσμα υποστηρίζει ότι η scan τροποποίηση δεν είναι μόνο λειτουργικά ορθή (Κεφ. 4) και χωρικά συνεπής (Κεφ. 5.3), αλλά οδηγεί και σε ουσιαστική βελτίωση της βασικής testability του κυκλώματος. Σημειώνεται ότι και στις δύο περιπτώσεις το εργαλείο δεν έφτασε τον στόχο 95% εντός του ceiling των 1000 vectors (“hit ceiling”), κάτι που αποδίδεται στον περιορισμένο προϋπολογισμό διανυσμάτων και στη φύση της μεθόδου παραγωγής. Η επέκταση με πιο συστηματικές στρατηγικές ATPG και μεγαλύτερο budget αποτελεί φυσική προοπτική μελλοντικής εργασίας.

Ιδιαίτερη προσοχή χρειάζεται στην ερμηνεία της μεταβολής του δεύτερου αριθμού μέσα στις παρενθέσεις, δηλαδή από 86 σε 421. Η τιμή αυτή δεν εκφράζει μείωση της ποιότητας του ελέγχου, αλλά αύξηση των port-related fault sites που αναγνωρίζει το εργαλείο μετά την εισαγωγή της scan λογικής. Με τις πρόσθετες scan θύρες και τις νέες διασυνδέσεις, το κύκλωμα αποκτά περισσότερα σημεία πρόσβασης και παρατήρησης. Έτσι, το Fault απαριθμεί περισσότερα σημεία στα οποία μπορεί να οριστεί stuck-at fault. Παρά την αύξηση αυτού του συνολικού χώρου σφαλμάτων, το ποσοστό κάλυψης αυξάνεται σημαντικά, γεγονός που δείχνει ότι η βελτιωμένη controllability και observability που προσφέρουν οι scan chains ενισχύει ουσιαστικά την testability του κυκλώματος.

4.5 Συνοπτικά συμπεράσματα κεφαλαίου

Στο κεφάλαιο αυτό παρουσιάστηκε η πλήρης διαδρομή από το αρχικό scan–enhanced netlist του Fault μέχρι ένα "καθαρό" netlist με τέσσερις σαφώς ορισμένες scan chains, καθώς και η μεθοδολογία επαλήθευσης της λειτουργίας τους. Το script rewrite_scan_chains_from_uuf.py αποδεικνύεται κομβικό, καθώς επιτρέπει τον ελεγχόμενο επανακαθορισμό των αλυσίδων με βάση τις πληροφορίες της ροής φυσικού σχεδιασμού (αρχεία uuf_chain*.txt), χωρίς να θυσιάζεται η συντακτική και λειτουργική εγκυρότητα του netlist.

Τα προγράμματα δοκιμής scan_tb_s5378_simple.v και scan_tb_s5378_chainwalk.v, σε συνδυασμό με την προσομοίωση μέσω iverilog στο περιβάλλον του Fault, παρέχουν σαφείς ενδείξεις ότι οι τέσσερις αλυσίδες είναι σωστά συνδεδεμένες: κάθε αλυσίδα έχει συγκεκριμένο μήκος, οι είσοδοι sin0–sin3 τροφοδοτούν όντως τα πρώτα flip–flops, και οι έξοδοι sout0–sout3 αντιστοιχούν στα τελευταία στοιχεία της σειράς. Το walking–1 test, καταγεγραμμένο στο αρχείο scan_chainwalk_log.txt, αποτελεί ένα απλό αλλά ιδιαίτερα εκφραστικό κριτήριο, το οποίο μπορεί να παρουσιαστεί αυτούσιο στον επιβλέποντα ως "πειραματική απόδειξη" της ορθής λειτουργίας των scan chains.

Στα επόμενα κεφάλαια, τα αποτελέσματα αυτά μπορούν να συσχετιστούν αφενός με την ανάλυση του layout (μέσω Magic και των αντίστοιχων αρχείων GDS/DEF) και αφετέρου με μελλοντική διασύνδεση σε εργαλεία ATPG, ώστε να εξεταστούν περισσότερο μετρικοί δείκτες, όπως το fault coverage και η επιβάρυνση σε επιφάνεια και ισχύ λόγω των δομών DfT.

Η συμβολή του παρόντος κεφαλαίου δεν περιορίζεται στην υλοποίηση ενός μετασχηματισμού netlist, αλλά στην ανάπτυξη μιας ελεγχόμενης και επαληθεύσιμης διαδικασίας αναδιοργάνωσης scan chains σε περιβάλλον ανοικτού κώδικα. Η διαδικασία αυτή συνδυάζει πληροφορία φυσικής τοποθέτησης, αυτοματοποιημένη επεξεργασία κειμένου Verilog και συστηματική λειτουργική επαλήθευση, συγκροτώντας μια μικρο–ροή DfT που μπορεί να επαναχρησιμοποιηθεί και σε άλλα σχέδια.

"Τα πλήρη waveforms για όλες τις αλυσίδες παρατίθενται στο Παράρτημα Α."

5. Ανάλυση σε επίπεδο διάταξης και οπτικοποίηση των scan chains

5.1 Στόχος και σύνδεση με τα προηγούμενα

Στα προηγούμενα κεφάλαια έγινε η ενσωμάτωση των δομών scan στο επίπεδο netlist και η επαλήθευση της λειτουργίας τους μέσω προσομοίωσης. Το επόμενο φυσικό βήμα είναι να εξεταστεί πώς αυτές οι δομές ενσωματώνονται στο φυσικό layout που παράγεται από την ανοικτή ροή mflowgen / OpenROAD, και ειδικότερα αν οι scan chains που ορίζονται λογικά στο netlist αντιστοιχούν με συνέπεια σε φυσικά instances στο ολοκληρωμένο κύκλωμα.

Στο κεφάλαιο αυτό εξετάζονται τα αποτελέσματα του σταδίου place & route για το κύκλωμα s5378 και περιγράφεται μια μεθοδολογία οπτικοποίησης των scan chains σε επίπεδο διάταξης, με βάση τα αρχεία DEF που παράγει η ροή OpenROAD. Αντί να βασιστούμε αποκλειστικά σε διαδραστικά εργαλεία layout, υλοποιείται μια ημι-αυτόματη προσέγγιση: εξαγωγή των συντεταγμένων των flip–flops από το DEF, συσχέτισή τους με τις αλυσίδες uuf_chain1.txt–uuf_chain4.txt, και παραγωγή γραφικών παραστάσεων (PDF) που αποτυπώνουν τη γεωμετρία κάθε scan chain.

Ο στόχος είναι διπλός. Αφενός, να υπάρχει μια "γέφυρα" μεταξύ του λογικού ανασυνδεδεμένου netlist s5378.chained.multi.v και του φυσικού layout, και αφετέρου να παρέχεται ένα εργαλείο οπτικής επιβεβαίωσης ότι οι αλυσίδες είναι καλά ορισμένες και αποτελούνται από το σωστό πλήθος flip–flops στις αναμενόμενες θέσεις του chip.

Η προσέγγιση αυτή μετατοπίζει την ανάλυση από την καθαρά λειτουργική ορθότητα σε μια πιο ολοκληρωμένη θεώρηση της DfT υλοποίησης, όπου η συνέπεια μεταξύ λογικού και φυσικού επιπέδου εξετάζεται ρητά. Σε πολλές περιπτώσεις, οι scan chains αντιμετωπίζονται αποκλειστικά ως λογικές δομές, χωρίς να αξιολογείται η χωρική τους διάσταση. Στην παρούσα εργασία επιχειρείται η συστηματική σύνδεση των δύο επιπέδων, αναδεικνύοντας τη σημασία της φυσικής πληροφορίας στη διαμόρφωση και τεκμηρίωση των δομών ελέγχου.

5.2 Αρχεία DEF/GDS από τη ροή OpenROAD

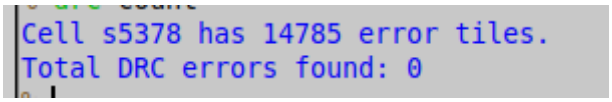
Η ροή mflowgen για το benchmark s5378 παράγει, στο τελικό στάδιο "openroad-finish", τα αρχεία φυσικού σχεδιασμού του κυκλώματος για την τεχνολογία Nangate45. Στον αντίστοιχο

κατάλογο αποτελεσμάτων εντοπίζονται τόσο αρχεία DEF όσο και αρχεία GDS, με ενδεικτικές διαδρομές:

8-orfs-openroad-finish/flow/results/nangate45/s5378/base/6_final.def
και
8-orfs-openroad-finish/flow/results/nangate45/s5378/base/6_1_merged.gds.

Το αρχείο GDS αποτελεί την τελική γεωμετρική αναπαράσταση του layout, σε επίπεδο πολυγώνων και μεταλλικών στρωμάτων. Για τον λόγο αυτό είναι κατάλληλο για προβολή του τελικού φυσικού σχεδίου και για ελέγχους DRC σε εργαλεία layout, όπως το Magic. Αντίθετα, το αρχείο DEF περιγράφει τη φυσική τοποθέτηση του σχεδίου σε πιο αφαιρετικό επίπεδο. Περιλαμβάνει τις τοποθετήσεις των standard cells, τα ονόματα των instances, τον τύπο κάθε κελιού και τις συντεταγμένες τους στο floorplan. Για τους σκοπούς της παρούσας εργασίας, το DEF είναι πιο χρήσιμο από το GDS, επειδή επιτρέπει την άμεση συσχέτιση των λογικών στοιχείων του netlist με τις φυσικές τους θέσεις στο layout. [13], [14]

Στο πλαίσιο της εργασίας έγινε επίσης προβολή του layout στο Magic και εκτελέστηκε έλεγχος DRC. Κατά τον έλεγχο αυτό εμφανίστηκε ένδειξη 14785 error tiles, όπως φαίνεται στην Εικόνα 5.1. Στο Magic, τα error tiles είναι περιοχές του layout που επισημαίνονται από τον ελεγκτή κανόνων σχεδίασης ως πιθανές παραβιάσεις των τεχνολογικών κανόνων. Ο αριθμός αυτός δεν σημαίνει απαραίτητα 14785 ανεξάρτητα σχεδιαστικά σφάλματα, καθώς μία ασυμβατότητα ή μία μη πλήρως προσαρμοσμένη τεχνολογική περιγραφή μπορεί να δημιουργήσει πολλές επιμέρους επισημάνσεις στο layout.



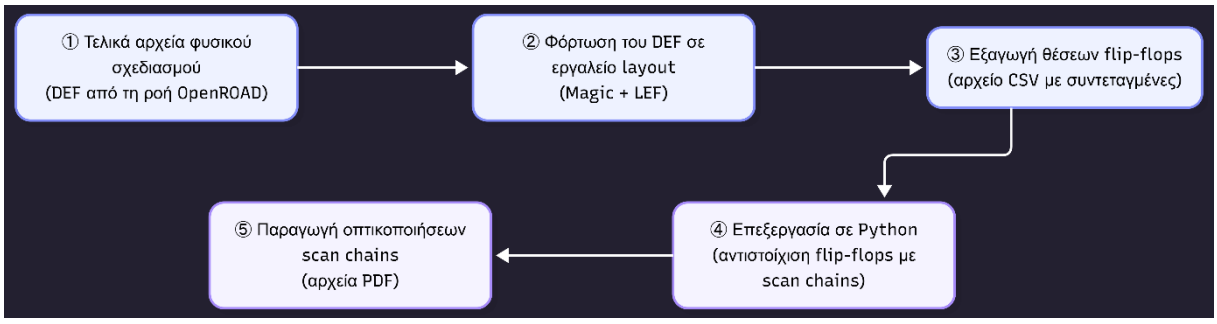
Εικόνα 5.1 Ένδειξη DRC error tiles στο Magic κατά την προβολή του layout.

Η ένδειξη αυτή αντιμετωπίζεται ως περιορισμός της διαδικασίας προβολής και ελέγχου στο Magic και όχι ως βασικό αντικείμενο αξιολόγησης της παρούσας μελέτης. Ο στόχος του κεφαλαίου δεν είναι η πλήρης φυσική επαλήθευση του GDS σε επίπεδο κατασκευαστικών κανόνων, αλλά η αξιοποίηση της φυσικής πληροφορίας για την ανάλυση των scan chains. Συγκεκριμένα, η εργασία επικεντρώνεται στον εντοπισμό των flip-flops που συμμετέχουν στις τέσσερις scan chains και στη συσχέτισή τους με τις φυσικές τους συντεταγμένες στο layout. [7]

Για τον λόγο αυτό, ο βασικός φορέας πληροφορίας στην παρούσα ενότητα είναι το αρχείο 6_final.def. Το DEF χρησιμοποιείται σε συνδυασμό με τη βιβλιοθήκη LEF της Nangate45 και φορτώνεται στο Magic, ώστε να εξαχθούν οι θέσεις των flip-flops. Μέσω κατάλληλου script εξαγωγής δημιουργείται το αρχείο s5378_dffs_positions.csv, το οποίο περιλαμβάνει για κάθε σχετικό instance το όνομα, τον τύπο του κελιού, τις συντεταγμένες σε DBU και μm, καθώς και τον προσανατολισμό του.

Το CSV αυτό λειτουργεί ως ενδιάμεση αναπαράσταση ανάμεσα στο φυσικό layout και στα Python scripts που χρησιμοποιούνται στη συνέχεια. Με βάση τις συντεταγμένες των flip-flops, μπορούν να δημιουργηθούν γραφικές απεικονίσεις των scan chains και να ελεγχθεί αν τα flip-flops που έχουν συνδεθεί λογικά στο αναδομημένο netlist αντιστοιχούν σε πραγματικά τοποθετημένα στοιχεία του τελικού σχεδίου. Έτσι, η ανάλυση δεν περιορίζεται στο επίπεδο Verilog/netlist, αλλά επεκτείνεται και στο φυσικό επίπεδο υλοποίησης.

Η διαδικασία αυτή ολοκληρώνει τη σύνδεση ανάμεσα στη λογική περιγραφή και στη φυσική διάταξη του κυκλώματος. Από τη μία πλευρά, το netlist και τα αρχεία uuf_chain*.txt καθορίζουν τη σειρά των flip-flops στις scan chains. Από την άλλη πλευρά, το DEF και το παραγόμενο CSV παρέχουν τις φυσικές θέσεις αυτών των flip-flops στο layout. Με τον τρόπο αυτό δημιουργείται μία συνεπής αλυσίδα τεκμηρίωσης, η οποία επιβεβαιώνει ότι οι scan chains που επαληθεύθηκαν λειτουργικά μπορούν να συσχετιστούν και με πραγματικά στοιχεία της φυσικής υλοποίησης.

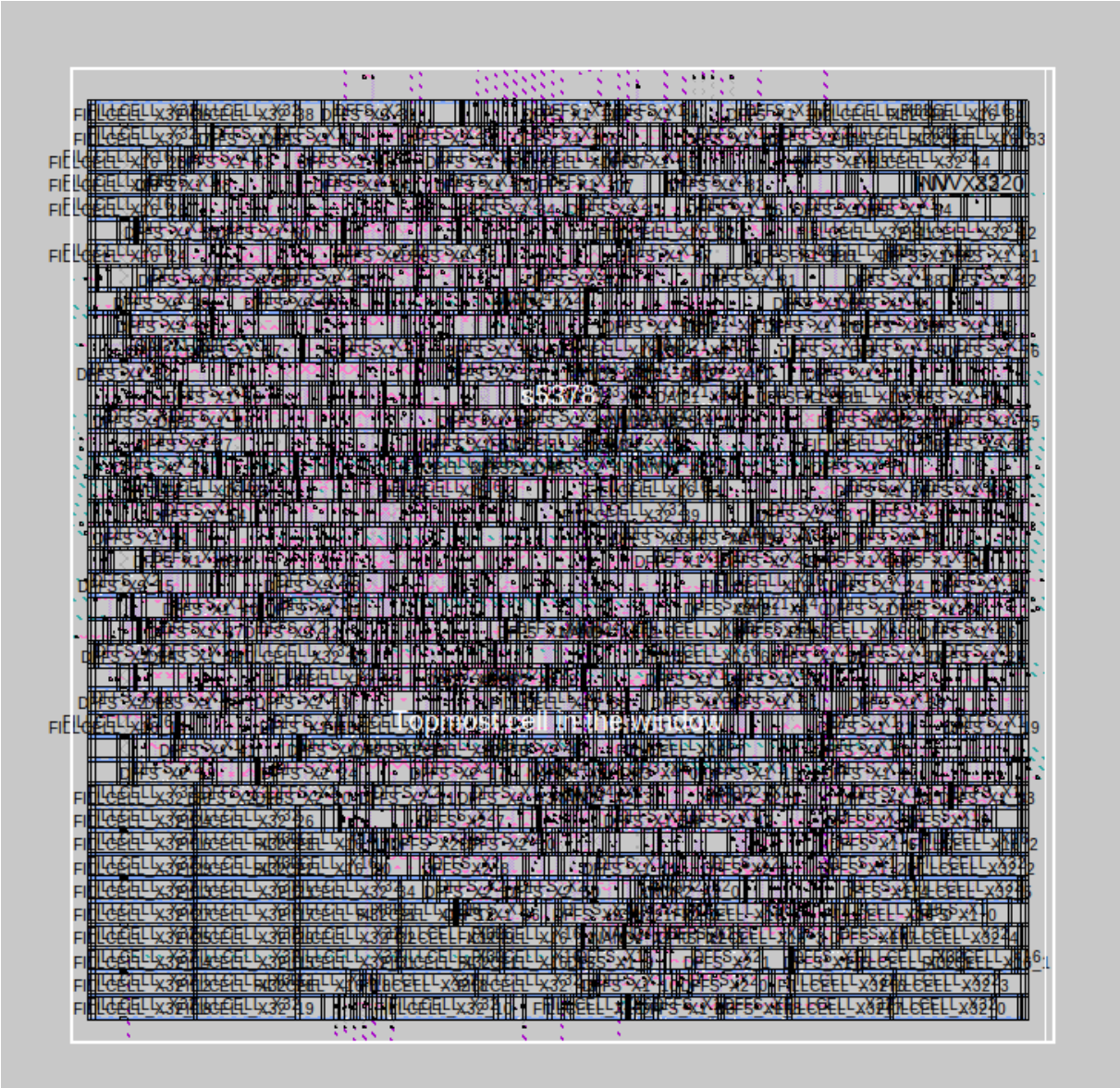


Εικόνα 5.2 Ροή εξαγωγής θέσεων flip-flops από DEF και παραγωγής οπτικοποιήσεων ανά scan chain.

5.3 Εξαγωγή και οπτικοποίηση των θέσεων των scan chains

Για να συνδεθεί η πληροφορία «ποιο flip–flop ανήκει σε ποια scan chain» με τη φυσική διάταξη του κυκλώματος, ακολουθήθηκε μία ημι-αυτόματη διαδικασία που αξιοποιεί το τελικό αρχείο DEF της ροής OpenROAD, εργαλεία layout και προσαρμοσμένα scripts επεξεργασίας. Η

διαδικασία αυτή δεν αποσκοπεί μόνο στην παραγωγή διαγραμμάτων οπτικοποίησης, αλλά κυρίως στη δημιουργία μίας ελεγχόμενης «αλυσίδας τεκμηρίωσης» που διασφαλίζει ότι τα flip–flops που συνδέθηκαν λογικά στις scan chains αντιστοιχούν σε πραγματικά, τοποθετημένα instances στο τελικό σχέδιο.



Εικόνα 5.3 Απεικόνιση του layout του s5378 στο Magic

Στην προβολή του layout εμφανίζονται επίσης πολλά βοηθητικά φυσικά κελιά που συχνά αναφέρονται ως fill shells ή filler cells. Τα κελιά αυτά δεν υλοποιούν λογική λειτουργία του κυκλώματος και δεν συμμετέχουν στις scan chains. Εισάγονται στο στάδιο της φυσικής υλοποίησης για να καλύψουν κενές περιοχές ανάμεσα στα standard cells, να διατηρήσουν τη συνέχεια των power rails και των wells και να ικανοποιηθούν οι τεχνολογικοί κανόνες του layout. Για τον λόγο

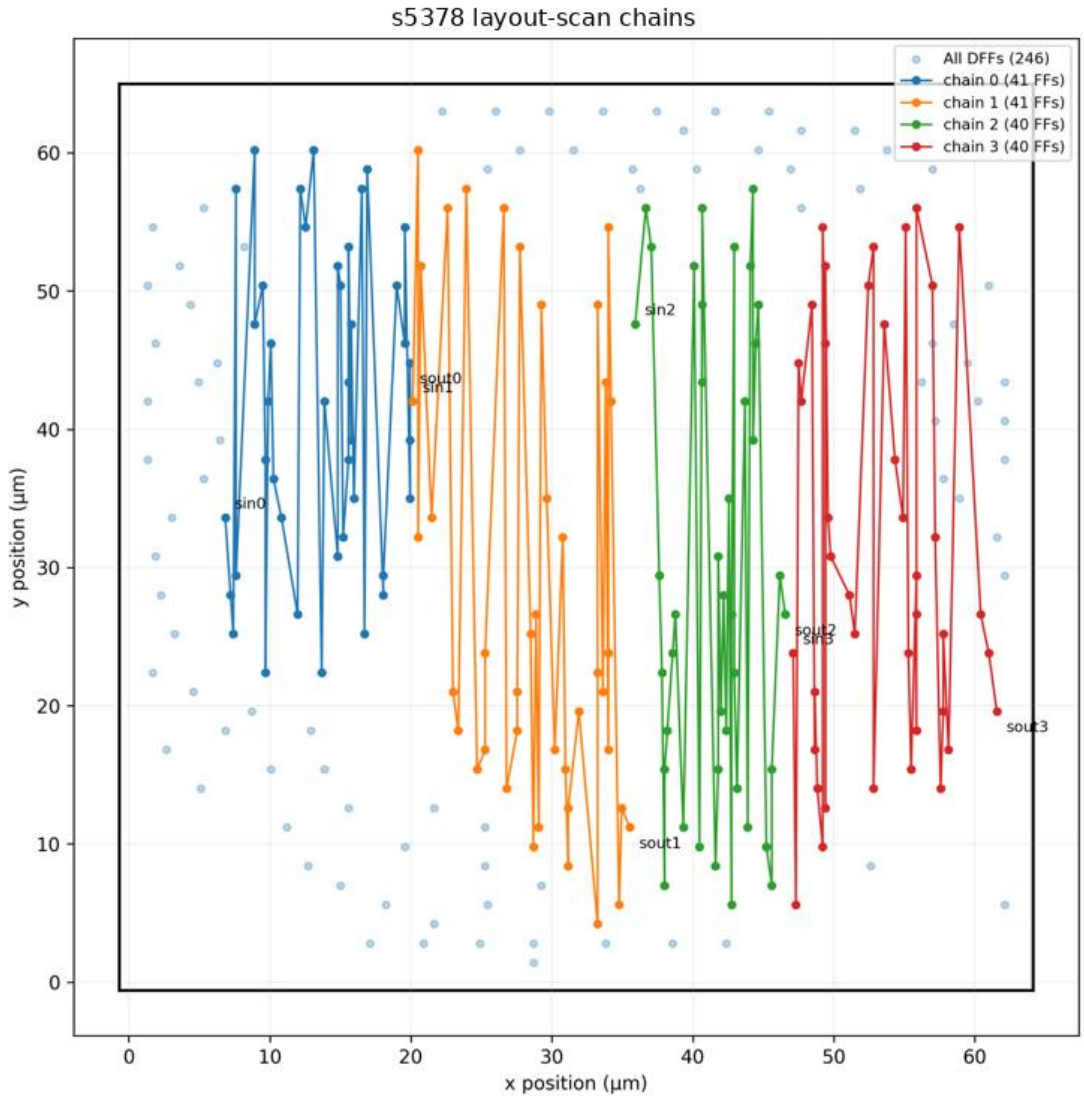
αυτό, κατά την εξαγωγή των θέσεων και τη δημιουργία των scan chain διαγραμμάτων λαμβάνονται υπόψη μόνο τα πραγματικά flip-flops και όχι τα filler ή άλλα physical-only cells.

Η αφετηρία της ανάλυσης είναι το τελικό DEF αρχείο (6_final.def), το οποίο προκύπτει μετά την ολοκλήρωση των σταδίων placement, clock tree synthesis και routing στο OpenROAD. Το DEF αποτυπώνει την τελική τοποθέτηση των standard cells μέσα στο floorplan, μέσω δηλώσεων components που περιλαμβάνουν το όνομα κάθε instance, τον τύπο του cell και τις συντεταγμένες τοποθέτησης. Καθώς το DEF αποτελεί αρχείο κειμένου, είναι κατάλληλο για αυτοματοποιημένη εξαγωγή συντεταγμένων και μεταγενέστερη επεξεργασία σε Python, χωρίς να απαιτείται αποκλειστικά διαδραστική χρήση layout viewer. [3], [4]

Ωστόσο, για πρακτικούς λόγους και για να διατηρηθεί ενιαίο σχήμα εξαγωγής, πραγματοποιείται και ένα ενδιάμεσο στάδιο φόρτωσης του DEF σε εργαλείο layout (Magic), μαζί με τα αντίστοιχα αρχεία βιβλιοθήκης (LEF) της Nangate45. Στο περιβάλλον του Magic εκτελείται script εξαγωγής που παράγει αρχείο CSV με τις θέσεις των flip–flops. Η χρήση CSV ως ενδιάμεσης αναπαράστασης είναι σημαντική, διότι μετατρέπει μια πληροφορία που βρίσκεται «εγκλωβισμένη» στο layout σε δομημένα δεδομένα κατάλληλα για ανάλυση, φίλτρα και ελέγχους συνέπειας.

Το παραγόμενο CSV περιλαμβάνει, για κάθε flip–flop που εντοπίζεται, πεδία όπως το όνομα instance, ο τύπος του cell, οι συντεταγμένες σε μονάδες του σχεδιαστικού πλέγματος (database units) και οι αντίστοιχες τιμές σε μικρόμετρα, καθώς και την πληροφορία προσανατολισμού (orientation). Η ύπαρξη των συντεταγμένων σε μικρόμετρα διευκολύνει την απευθείας οπτικοποίηση σε 2D επίπεδο και επιτρέπει την ποιοτική σύγκριση της χωρικής κατανομής των στοιχείων. Παράλληλα, η πληροφορία προσανατολισμού είναι χρήσιμη ως ένδειξη ότι το instance αντιστοιχεί σε πραγματικό placement standard cell και όχι σε τεχνητή ή ελλιπή εγγραφή.

Με βάση τα δεδομένα του CSV, ακολουθεί το στάδιο συσχέτισης των flip–flops με τις scan chains. Η συσχέτιση αυτή πραγματοποιείται μέσω των αρχείων uuf_chain1.txt έως uuf_chain4.txt, τα οποία περιέχουν τα ονόματα των flip–flops του block **uuf** ανά αλυσίδα. Τα αρχεία αυτά λειτουργούν ως «λογική περιγραφή» του membership κάθε chain, ενώ το CSV/DEF παρέχει τη «φυσική περιγραφή» (θέση) των ίδιων instances. Η σύνδεση των δύο επιπέδων επιτυγχάνεται με αντιστοίχιση ονομάτων (name matching) μεταξύ των λιστών των chains και των instances που εμφανίζονται στο CSV.



Εικόνα 5.4 Οπτικοποίηση των τεσσάρων scan chains του s5378 στο layout, με βάση τις θέσεις των flip-flops από το CSV και τις σωστές ενώσεις του παραρτήματος.

Στην Εικόνα 5.4 οι κουκκίδες που δεν ενώνονται με τις γραμμές των scan chains αντιστοιχούν σε boundary σημεία/περιμετρικά σημεία αναφοράς του layout και δεν πρέπει να ερμηνεύονται ως σπασμένα ή ασύνδετα flip-flops των αλυσίδων. Οι πραγματικές συνδέσεις των scan chains αποτυπώνονται από τις γραμμές που ενώνουν διαδοχικά τα flip-flops κάθε chain. Για να είναι πιο ευανάγνωστη η κατανομή, ο άξονας x έχει χρησιμοποιηθεί και ως άξονας διαχωρισμού των τεσσάρων αλυσίδων: η περιοχή 0–20 αντιστοιχεί στην πρώτη chain, η περιοχή 20–40 στη δεύτερη, η περιοχή 40–60 στην τρίτη και η περιοχή 60–80 στην τέταρτη. Με αυτόν τον τρόπο φαίνεται καθαρά τόσο η φυσική θέση των στοιχείων όσο και η αντιστοίχισή τους στις τέσσερις ανεξάρτητες scan chains.

Για τη διασφάλιση της ορθότητας της αντιστοίχισης, εφαρμόζονται έλεγχοι συνέπειας πριν την παραγωγή των τελικών διαγραμμάτων. Πρώτον, ελέγχεται ότι ο συνολικός αριθμός flip–flops ανά chain συμφωνεί με τις αναμενόμενες τιμές που έχουν προκύψει από τη λογική αναδιοργάνωση στο Κεφάλαιο 4. Δεύτερον, ελέγχεται ότι κάθε flip–flop που εμφανίζεται σε chain list υπάρχει και στο CSV/DEF ως τοποθετημένο instance, ώστε να αποκλειστούν περιπτώσεις ασυνέπειας λόγω βελτιστοποιήσεων ή αφαίρεσης στοιχείων στη ροή. Τρίτον, επιβεβαιώνεται ότι δεν υπάρχουν διπλές εγγραφές (duplicate instances) σε περισσότερες από μία αλυσίδες, καθώς κάτι τέτοιο θα υποδήλωνε σφάλμα στην παραγωγή των chain lists ή στη διαδικασία αντιστοίχισης.

Αφού ολοκληρωθούν οι παραπάνω έλεγχοι, υλοποιείται η οπτικοποίηση. Αναπτύχθηκε βοηθητικό script σε Python (ενδεικτικά make_chain_graphs.py), το οποίο ανακτά τις συντεταγμένες των flip–flops κάθε chain και δημιουργεί 2D διαγράμματα διασποράς, όπου κάθε flip–flop απεικονίζεται ως σημείο στο επίπεδο (x, y). Για κάθε scan chain παράγεται ξεχωριστό αρχείο PDF (chain0.pdf–chain3.pdf), ώστε να είναι εύκολη η μεμονωμένη μελέτη της κατανομής της κάθε αλυσίδας. Η επιλογή παραγωγής PDF επιτρέπει την ενσωμάτωση των αποτελεσμάτων στο κείμενο της διπλωματικής ή στο παράρτημα, χωρίς εξάρτηση από λογισμικό προβολής waveforms ή layout viewers.

Οι παραγόμενες αναπαραστάσεις λειτουργούν ως «χάρτες» των scan chains σε επίπεδο διάταξης. Από αυτά μπορεί να εξαχθεί ποιοτική εικόνα για το αν μία αλυσίδα είναι χωρικά συμπαγής, αν παρουσιάζει μεγάλη διασπορά ή αν φαίνεται να «τέμνει» πολλές περιοχές του floorplan. Παρότι η παρούσα εργασία δεν επιδιώκει πλήρη φυσική βελτιστοποίηση των scan chains, η οπτικοποίηση επιτρέπει να εντοπιστούν χαρακτηριστικά που θα επηρέαζαν μελλοντικά το wirelength ή τη συμφόρηση στη δρομολόγηση.

Στο πλαίσιο της παρούσας εργασίας, τα διαγράμματα chain0.pdf–chain3.pdf τεκμηριώνουν ότι οι τέσσερις αλυσίδες περιλαμβάνουν αντίστοιχα 41, 41, 40 και 40 flip–flops. Το άθροισμα των στοιχείων αυτών ισούται με το συνολικό πλήθος των 162 flip–flops του block uuf, όπως αυτό καταγράφηκε κατά την ανάλυση του netlist στο Κεφάλαιο 4. Η συμφωνία αυτή αποτελεί ένδειξη ότι η μεταφορά πληροφορίας μεταξύ netlist, αρχείων uuf_chain*.txt και φυσικής αναπαράστασης (DEF/CSV) πραγματοποιήθηκε χωρίς απώλειες, ασυνέπειες ή διπλοεγγραφές. Επιπλέον, το γεγονός ότι τα instances που εμφανίζονται στις οπτικοποιήσεις αντιστοιχούν σε στοιχεία του uuf επιβεβαιώνει ότι η διαδικασία εξαγωγής και αντιστοίχισης διατηρεί τη συνέπεια μεταξύ λογικού και φυσικού επιπέδου.

Με αυτόν τον τρόπο, η ανάλυση σε επίπεδο διάταξης παραμένει πλήρως συμβατή με τη λογική αναδιοργάνωση των scan chains που παρουσιάστηκε στο Κεφάλαιο 4 και συμπληρώνει τη λειτουργική επαλήθευση (walking–1) με μία σαφή χωρική τεκμηρίωση, η οποία είναι ιδιαίτερα χρήσιμη όταν ο στόχος είναι η μελέτη DfT σε πλήρως τοποθετημένο και δρομολογημένο σχέδιο.

5.4 Συμπεράσματα από την ανάλυση σε layout και περιορισμοί

Η οπτικοποίηση των scan chains σε επίπεδο διάταξης συμπληρώνει την καθαρά λογική επαλήθευση που παρουσιάστηκε στο προηγούμενο κεφάλαιο και εισάγει μία δεύτερη διάσταση τεκμηρίωσης, βασισμένη στη φυσική υλοποίηση. Στο Κεφάλαιο 4, η διαδικασία “flush + walking–1” επιβεβαιώνει λειτουργικά ότι οι αλυσίδες είναι σωστά συνδεδεμένες και σειριακά συνεκτικές. Στο παρόν κεφάλαιο, η ανάλυση μέσω DEF/CSV και η παραγωγή των διαγραμμάτων chain0.pdf–chain3.pdf επιβεβαιώνει ότι τα ίδια flip–flops υφίστανται ως πραγματικά τοποθετημένα instances στο τελικό layout και ότι η αντιστοίχιση των chains προς φυσικές θέσεις είναι συνεπής.

Η σύζευξη λογικού και φυσικού επιπέδου είναι ιδιαίτερα κρίσιμη σε δομές DfT, επειδή η αποτελεσματικότητα των scan chains σε πραγματικές ροές δεν εξαρτάται μόνο από τη συνδεσμολογία, αλλά και από τη χωρική κατανομή των flip–flops που τις αποτελούν. Για παράδειγμα, μια αλυσίδα που είναι λειτουργικά σωστή αλλά χωρικά πολύ διασκορπισμένη ενδέχεται να οδηγήσει σε αυξημένο wirelength των scan διασυνδέσεων, επιπλέον φόρτο στη δρομολόγηση και μεγαλύτερη πιθανότητα timing impact ή συμφόρησης. Αντιθέτως, μια πιο συμπαγής χωρικά αλυσίδα μπορεί να είναι ευνοϊκότερη ως προς wirelength και congestion, ιδιαίτερα σε μεγαλύτερα σχέδια.

Αν και η παρούσα εργασία δεν στοχεύει σε πλήρη φυσική βελτιστοποίηση των scan chains, η οπτικοποίηση που παράγεται αποτελεί φυσικό σημείο εκκίνησης για ποσοτική αξιολόγηση. Μία άμεση επέκταση της μεθοδολογίας θα ήταν ο υπολογισμός εκτιμώμενων μετρικών wirelength για κάθε αλυσίδα, π.χ. με βάση Manhattan αποστάσεις μεταξύ διαδοχικών flip–flops στην ακολουθία της chain. Παράλληλα, θα μπορούσε να υπολογιστεί η χωρική «συμπαγότητα» κάθε αλυσίδας, όπως το εμβαδό του bounding box που περικλείει τα flip–flops της, ή δείκτες διασποράς που ποσοτικοποιούν αν μία chain συγκεντρώνεται σε υποπεριοχή του floorplan ή κατανέμεται σε όλο το chip.

Πέρα από την αξιολόγηση, τα ίδια δεδομένα θα μπορούσαν να χρησιμοποιηθούν και για βελτιστοποίηση. Σε μελλοντική εργασία, η γεωμετρική πληροφορία από το DEF/CSV μπορεί να οδηγήσει σε αλγοριθμική αναδιάταξη της σειράς των flip–flops μέσα σε κάθε chain, με κριτήριο τη μείωση του συνολικού wirelength. Τέτοιες προσεγγίσεις θα μπορούσαν να βασιστούν σε κλασικές ευρετικές (nearest-neighbor ordering), σε clustering κατά περιοχές, ή ακόμη και σε παραλλαγές προβλημάτων τύπου traveling salesman (TSP) για την εύρεση πιο «σύντομης» ακολουθίας επισκέψεων των flip–flops. Η ενσωμάτωση τέτοιων τεχνικών θα μετέτρεπε τη διαδικασία από «οπτικοποίηση» σε πλήρη «DFT-aware physical optimization», κάτι που συνδέεται άμεσα με ζητήματα που αντιμετωπίζονται συνήθως σε εμπορικές ροές.

Ένας πρακτικός περιορισμός που αναδείχθηκε είναι ότι τα κλασικά layout viewers, όπως το Magic, δεν είναι πάντοτε άμεσα αξιοποιήσιμα για σύγχρονες βιβλιοθήκες standard cells χωρίς κατάλληλα προσαρμοσμένο τεχνολογικό αρχείο. Σε τέτοιες περιπτώσεις, η οπτική ανάλυση σε επίπεδο στρωμάτων μπορεί να είναι δύσχρηστη ή να μην αποδίδει καθαρά το routing. Για τον λόγο αυτό, η επιλογή να στηριχθεί το κύριο μέρος της ανάλυσης στο DEF (και σε ενδιάμεσο CSV) είναι πιο ευέλικτη, αναπαραγώγιμη και σύμφωνη με τη φιλοσοφία των ανοιχτών εργαλείων.

Πρέπει επίσης να σημειωθεί ότι η ανάλυση που βασίζεται σε θέσεις standard cells αποτυπώνει τη χωρική κατανομή των flip–flops, αλλά δεν αποτυπώνει άμεσα τη λεπτομερή γεωμετρία των scan διασυνδέσεων όπως τελικά δρομολογούνται. Με άλλα λόγια, το γεγονός ότι δύο flip–flops βρίσκονται κοντά γεωμετρικά δεν εγγυάται απαραίτητα ότι η πραγματική διασύνδεση θα είναι αντίστοιχα «σύντομη», επειδή η τελική δρομολόγηση επηρεάζεται από congestion, blockages, clock routing και γενικότερους περιορισμούς. Επομένως, η οπτικοποίηση αποτελεί ισχυρό εργαλείο συνέπειας και ποιοτικής αξιολόγησης, αλλά όχι πλήρη υποκατάσταση εργαλείων λεπτομερούς route analysis.

Συνοψίζοντας, το παρόν κεφάλαιο επιβεβαιώνει ότι οι τέσσερις scan chains που αναδιοργανώθηκαν στο επίπεδο netlist αντανακλώνται συνεπώς και στο επίπεδο διάταξης. Υπάρχει συμφωνία ανάμεσα στις πληροφορίες των αρχείων uuf_chain*.txt, στο αναδομημένο netlist s5378.chained.multi.v και στο τελικό DEF της OpenROAD, ενώ η οπτικοποίηση μέσω των chain0.pdf–chain3.pdf παρέχει ένα πρακτικό και ισχυρό εργαλείο ελέγχου συνέπειας μεταξύ λογικού και φυσικού επιπέδου.

Η ανάλυση σε επίπεδο διάταξης, σε συνδυασμό με τη λειτουργική επαλήθευση του Κεφαλαίου 4, καταδεικνύει ότι η προτεινόμενη μικρο–ροή DfT δεν παραμένει σε αφηρημένο επίπεδο netlist, αλλά εφαρμόζεται και τεκμηριώνεται σε πλήρως τοποθετημένο και δρομολογημένο σχέδιο. Στο επόμενο κεφάλαιο συνοψίζονται τα συμπεράσματα της εργασίας και συζητούνται περιορισμοί και προτάσεις επέκτασης, όπως η ενσωμάτωση πληρέστερων ATPG μετρικών (fault coverage) και η διερεύνηση κριτηρίων φυσικής βελτιστοποίησης των scan chains.

6. Συμπεράσματα και μελλοντική εργασία

6.1 Σύνοψη στόχων και υλοποιηθέντος έργου

Στόχος της παρούσας μεταπτυχιακής εργασίας ήταν η μελέτη της Σχεδίασης για Έλεγχο (Design for Testability – DfT) στο πλαίσιο ανοιχτών ροών σχεδίασης VLSI, με έμφαση στην ενσωμάτωση και επαλήθευση δομών scan σε ένα συγκεκριμένο benchmark κύκλωμα. Η εργασία επικεντρώθηκε στη ροή mflowgen / OpenROAD και στη χρήση εργαλείων ανοιχτού κώδικα, τόσο για το στάδιο της λογικής σύνθεσης όσο και για το στάδιο της φυσικής υλοποίησης. [3]–[6]

Πρακτικά, η υλοποίηση βασίστηκε στο κύκλωμα s5378 και στο netlist που παρήχθη από το εργαλείο Fault σε μορφή s5378.chained.v, όπου υπήρχαν ήδη ενσωματωμένα scan flip–flops σε μπλοκ τύπου __uuf__. Πάνω σε αυτό το netlist αναπτύχθηκε ένας μηχανισμός αναδιάταξης των scan chains, με στόχο την κατασκευή τεσσάρων σαφώς ορισμένων αλυσίδων, καθώς και μία πλήρης ροή προσομοίωσης και επαλήθευσης της λειτουργίας τους. Στη συνέχεια, τα αποτελέσματα συνδέθηκαν με το φυσικό layout που παρήγαγε η ροή OpenROAD, ώστε να υπάρξει συνέπεια μεταξύ λογικού και φυσικού επιπέδου. [8], [9]

Η διαδικασία αυτή οδήγησε σε μια συγκεκριμένη, επαναλήψιμη “μικρο–ροή DfT” πάνω από το υπάρχον flow: από το αρχικό netlist με ενσωματωμένα scan στοιχεία, στην αναδόμηση των αλυσίδων (rewire), στην προσομοίωση ελέγχου (chain testbenches) και τέλος στην οπτικοποίηση των scan chains σε επίπεδο διάταξης.

6.2 Συμπεράσματα για την προτεινόμενη ροή DfT

Ένα από τα βασικά αποτελέσματα της εργασίας είναι ότι, ξεκινώντας από ένα netlist τύπου s5378.chained.v, είναι δυνατή η αξιόπιστη επανασύνδεση των scan flip–flops σε πολλαπλές αλυσίδες, σύμφωνα με εξωτερικά αρχεία περιγραφής (uuf_chain1.txt–uuf_chain4.txt). Το script rewrite_scan_chains_from_uuf.py υλοποιεί αυτή τη διαδικασία κατά τρόπο πλήρως αυτοματοποιημένο: εντοπίζει τα instances του block __uuf__, τα αντιστοιχίζει στις αλυσίδες, εισάγει τα κατάλληλα scan multiplexers και παράγει το αναδομημένο netlist s5378.chained.multi.v, στο οποίο το module s5378 αποκτά σαφώς ορισμένες θύρες sin0..sin3 και sout0..sout3. [8], [9]

Η ορθότητα της σύνδεσης των αλυσίδων επαληθεύτηκε αρχικά σε λογικό επίπεδο. Το πρόγραμμα δοκιμής scan_tb_s5378_simple.v επιτρέπει μια βασική λειτουργική δοκιμή των scan chains. Πιο αναλυτικά, το πρόγραμμα δοκιμής scan_tb_s5378_chainwalk.v εφαρμόζει μια διαδικασία "flush + walking–1", στην οποία αρχικά όλες οι αλυσίδες μηδενίζονται με διαδοχικά shift–cycles και στη συνέχεια εισάγεται ένα μοναδικό "1" στην είσοδο κάθε chain. Από τα log files της προσομοίωσης (scan_chainwalk_log.txt) φαίνεται ότι για κάθε μία από τις τέσσερις αλυσίδες, το "1" διαδίδεται μέσα από όλα τα flip–flops και εμφανίζεται ακριβώς μία φορά στην αντίστοιχη έξοδο soutX μετά από N κύκλους, όπου N είναι το μήκος της αλυσίδας (41, 41, 40 και 40 θέσεις αντίστοιχα). Αυτό αποτελεί μια ισχυρή ένδειξη ότι:

- οι αλυσίδες είναι σειριακά συνεκτικές,
- δεν υπάρχουν "σπασίματα" ή fan–out μέσα στην αλυσίδα,
- και δεν έχουν χαθεί flip–flops κατά τη διαδικασία ανασύνδεσης.

Στη συνέχεια, τα λογικά αποτελέσματα συνδέθηκαν με το φυσικό επίπεδο. Με βάση το DEF αρχείο 6_final.def που παράγει η ροή OpenROAD και τα αρχεία uuf_chain*.txt, υλοποιήθηκε ένα βοηθητικό script (π.χ. make_chain_graphs.py) που εξάγει τις συντεταγμένες των flip–flops και παράγει τα διαγράμματα chain0.pdf–chain3.pdf. Τα διαγράμματα αυτά δείχνουν την ακριβή γεωμετρία κάθε αλυσίδας πάνω στο chip, προσφέροντας μια οπτική επιβεβαίωση ότι:

- τα flip–flops κάθε chain είναι όντως παρόντα στο layout,
- ο αριθμός τους συμφωνεί με τον λογικό ορισμό των αλυσίδων,

και οι αλυσίδες εκτείνονται σε λογικές περιοχές του floorplan, σύμφωνα με την τοποθέτηση των standard cells που πραγματοποίησε το OpenROAD.

Σε συνδυασμό, τα παραπάνω βήματα δείχνουν ότι είναι εφικτό, χρησιμοποιώντας αποκλειστικά εργαλεία ανοιχτού κώδικα, να σχεδιαστεί και να επαληθευθεί μια πλήρης υλοποίηση scan chains τόσο σε netlist όσο και σε layout επίπεδο, για ένα μη τετριμμένο κύκλωμα όπως το s5378.

6.3 Περιορισμοί της υφιστάμενης υλοποίησης

Παρά τα θετικά αποτελέσματα, η παρούσα εργασία έχει και σαφείς περιορισμούς, οι οποίοι ορίζουν και τα περιθώρια για πιθανές επεκτάσεις.
Πρώτον, η μελέτη περιορίστηκε σε ένα μόνο benchmark κύκλωμα (s5378) και σε μία

συγκεκριμένη τεχνολογία standard–cell (Nangate45). Αυτό επιτρέπει λεπτομερή ανάλυση της ροής, αλλά δεν αρκεί για γενίκευση συμπερασμάτων όσον αφορά την κλιμάκωση σε μεγαλύτερα σχέδια ή σε διαφορετικές βιβλιοθήκες.

Δεύτερον, η εστίαση ήταν κυρίως στη δομή των scan chains και στη λειτουργική/φυσική συνέπειά τους. Σε επίπεδο ποσοτικών μετρικών, πραγματοποιήθηκε **βασική** αξιολόγηση testability μέσω stuck-at fault coverage πριν και μετά την εισαγωγή scan δομών (Ενότητα 5.4), όμως δεν πραγματοποιήθηκε συστηματική και ολοκληρωμένη αποτίμηση του κόστους της DfT τροποποίησης σε όρους επιφάνειας (area) ή κατανάλωσης ισχύος (power) από τα reports της ροής OpenROAD. Με άλλα λόγια, η εργασία απαντά αξιόπιστα στο ερώτημα «είναι σωστά συνδεδεμένες, λειτουργικές και συνεπείς με το layout οι αλυσίδες;», αλλά δεν απαντά πλήρως στο «ποιο είναι το area/power overhead που εισάγουν οι scan δομές;».

Τρίτον, αν και έγινε προσπάθεια να χρησιμοποιηθούν εργαλεία layout όπως το Magic, αποδείχθηκε ότι χωρίς κατάλληλα τεχνολογικά αρχεία προσαρμοσμένα στην Nangate45, η απεικόνιση δεν είναι ιδιαίτερα πρακτική. Η επιλογή να βασιστούμε στο DEF και σε scripts για την οπτικοποίηση των chains είναι λειτουργική και αναπαραγώγιμη, αλλά δεν αντικαθιστά πλήρως μια ολοκληρωμένη "DFT-aware" προβολή layout (π.χ. με λεπτομερή αποτύπωση routing των scan διασυνδέσεων).

Τέλος, παρότι πραγματοποιήθηκε ένα ενδεικτικό πείραμα ATPG/fault simulation με το Fault (stuck-at coverage και compaction των test vectors, Ενότητα 5.4), η εργασία δεν προχώρησε σε πλήρη αξιολόγηση βιομηχανικού επιπέδου. Συγκεκριμένα, δεν εξετάστηκαν transition faults, deterministic ATPG στρατηγικές, εναλλακτικά budgets διανυσμάτων/χρόνου, ούτε συγκριτικές μετρικές όπως pattern count έναντι κάλυψης ή επίπτωση της scan λογικής σε timing/λειτουργικούς περιορισμούς. Επομένως, το κομμάτι της testability τεκμηριώνεται σε βασικό επίπεδο, αλλά παραμένει ανοικτό πεδίο για πιο συστηματική επέκταση.

6.4 Προτάσεις για μελλοντική έρευνα και επεκτάσεις

Με βάση τα παραπάνω, προκύπτουν αρκετές φυσικές κατευθύνσεις για μελλοντική εργασία. Μια πρώτη κατεύθυνση είναι η επέκταση της ροής DfT σε περισσότερα benchmarks και μεγαλύτερα κυκλώματα, καθώς και σε άλλες τεχνολογικές βιβλιοθήκες. Η εφαρμογή της ίδιας μεθοδολογίας σε πολλαπλά designs θα επιτρέψει την εκτίμηση της κλίμακωσης της λύσης και θα αναδείξει τυχόν ζητήματα απόδοσης ή αυτοματοποίησης που δεν εμφανίζονται στο s5378.

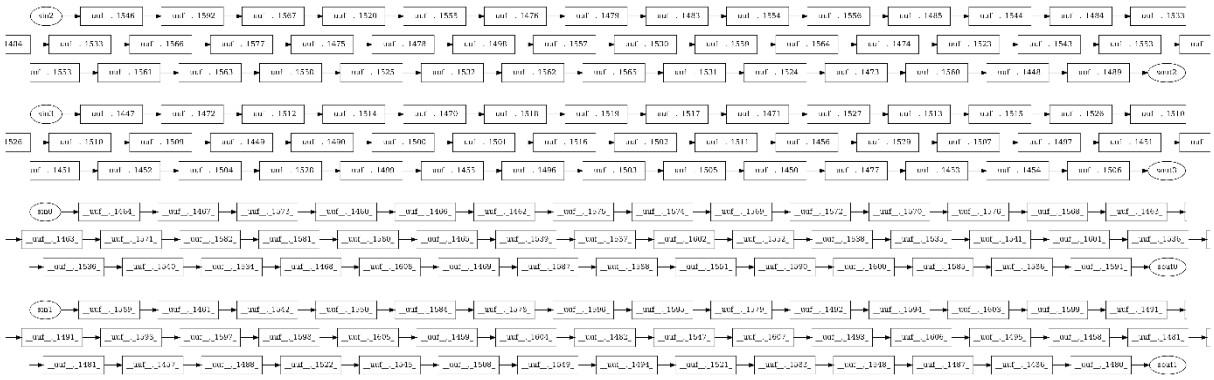
Μια δεύτερη κατεύθυνση είναι η στενότερη ενσωμάτωση εργαλείων ATPG στην ίδια ροή. Ιδανικά, από το αναδομημένο netlist με τις τέσσερις scan chains θα μπορούσαν να παράγονται αυτόματα διανύσματα stuck-at και transition, να εφαρμόζονται στην προσομοίωση και να υπολογίζεται το fault coverage. Με αυτό τον τρόπο, η ροή δεν θα περιοριζόταν στην "δομική" ορθότητα των chains, αλλά θα παρείχε μια πλήρη εικόνα της δοκιμότητας του κυκλώματος.

Μια ακόμη προέκταση αφορά το physical optimization των scan chains. Στην τρέχουσα εργασία, οι αλυσίδες ορίζονται βάσει των αρχείων uuf_chain*.txt χωρίς να λαμβάνεται υπόψη η γεωμετρία του layout. Μελλοντικά, θα μπορούσε να εξεταστεί η αναδιάταξη των chains με κριτήρια όπως το συνολικό μήκος配 των διασυνδέσεων ή η ελαχιστοποίηση συμφόρησης στη δρομολόγηση, αξιοποιώντας είτε τις πληροφορίες του DEF είτε APIs εργαλείων όπως το OpenROAD.

Τέλος, αξίζει να διερευνηθεί η δυνατότητα ενοποίησης όλων των παραπάνω βημάτων σε μία πιο "συμπαγή" ροή, π.χ. μέσω ενός ενιαίου Python wrapper ή ενός mflowgen node, έτσι ώστε, με μία παραμετροποιημένη εκτέλεση, να προκύπτουν: το netlist με scan, οι αλυσίδες, τα προγράμματα δοκιμής, τα αρχεία προσομοίωσης και οι οπτικοποιήσεις των chains. Μια τέτοια ροή θα μπορούσε να λειτουργήσει ως σημείο εκκίνησης για άλλα ερευνητικά έργα που επιθυμούν να αξιοποιήσουν ανοιχτά εργαλεία DfT σε ακαδημαϊκό ή βιομηχανικό περιβάλλον.

Συνολικά, η παρούσα εργασία δείχνει ότι, ακόμη και με τους περιορισμούς των ανοιχτών ροών και βιβλιοθηκών, είναι εφικτή μια συνεκτική υλοποίηση DfT για αλυσίδα scan chains, από το λογικό netlist μέχρι το τελικό layout. Τα βήματα που υλοποιήθηκαν για το s5378 μπορούν να αποτελέσουν τη βάση για πιο πλήρεις, "DFT-aware" ροές VLSI σχεδίασης στο μέλλον, όπου η σχεδίαση και ο έλεγχος θα αντιμετωπίζονται ως δύο όψεις της ίδιας διαδικασίας.

ΠΑΡΑΡΤΗΜΑ



```python
def split_into_chains(dffs, num_chains):
    dffs_sorted = sorted(dffs, key=lambda t: t[2])

    n = len(dffs_sorted)
    per_chain = math.ceil(n / num_chains)

    for c in range(num_chains):
        start = c * per_chain
        end = min((c + 1) * per_chain, n)
        chunk = dffs_sorted[start:end]

        out_name = f"dffs_chain{c+1}_no_tap.txt"
        with open(out_name, "w") as out_f:
            for inst_name, cell_type, x, y in chunk:
                out_f.write(inst_name + "\n")

        print(f"[INFO] Εγραψα {len(chunk):3d} FFs στο {out_name}")
```

Διαχωρισμός flip-flops σε N scan chains με βάση τη συντεταγμένη x και φίλτρο TAP-cells.

```python
def build_chain_maps(
    chains: List[List[str]],
    dffs: Dict[str, dict],
) -> Tuple[Dict[str, int], Dict[str, str], Dict[int, str]]:
    chain_of: Dict[str, int] = {}
    prev_of: Dict[str, str] = {}
    last_of: Dict[int, str] = {}

    for cidx, chain in enumerate(chains):
        prev = None
        for raw_name in chain:
            inst = raw_name.strip()
            if inst not in dffs:
                continue

            chain_of[inst] = cidx
            prev_of[inst] = prev
            prev = inst

        if prev is not None:
            last_of[cidx] = prev
            print(f"[INFO] Αλυσίδα {cidx}: last FF = {prev}")
        else:
            print(f"[WARN] Αλυσίδα {cidx} δεν έχει κανένα FF μέσα στο netlist.")

    print(f"[INFO] Θα ξαναδέσω {len(chain_of)} FFs σε αλυσίδες (όσα βρέθηκαν στο netlist).")
    return chain_of, prev_of, last_of
```

Δημιουργία αντιστοιχίσεων των flip-flops στις scan chains

```python
def rewrite_text_with_mux(
    text: str,
    dffs: Dict[str, dict],
    chain_of: Dict[str, int],
    prev_of: Dict[str, str],
    n_chains: int,
) -> str:
    dff_pat = re.compile(
        r'(?P<indent>\s*)'
        r'(?P<cell>DFF\S*_X1)\s+\\(?P<inst>__uuf__\.[^ \t(]+)\s*\(\s*\n'
        r'(?P<body>.*?^\s*\);\s*\n)',
        re.MULTILINE | re.DOTALL,
    )

    def safe_name(inst: str) -> str:
        s = inst.replace(".", "_")
        s = s.replace("[", "_").replace("]", "_")
        return s

    def repl(m: re.Match) -> str:
        inst = m.group("inst")
        cell = m.group("cell")
        indent = m.group("indent")
        body = m.group("body")

        if inst not in dffs or inst not in chain_of:
            return m.group(0)

        info = dffs[inst]
        chain_idx = chain_of[inst]
        prev_inst = prev_of.get(inst)

        if prev_inst is None:
            scan_in = f"sin{chain_idx}"
        else:
            prev_q = fmt_net(dffs[prev_inst]["Q"])
            scan_in = prev_q

        d_fun = fmt_net(info["D"])
        safe = safe_name(inst)

        body_new = re.sub(
            r'(\.D\s*\(\s*([^()]+?)\s*(\))',
            rf'\1scan_d___uuf____{safe}\3',
            body,
            count=1,
        )

        mux_lines: List[str] = []
        mux_lines.append(f"{indent}wire scan_d___uuf____{safe};\n")
        mux_lines.append(f"{indent}MUX2_X1 scanmux___uuf____{safe} (\n")
        mux_lines.append(f"{indent}  .A({d_fun}),\n")
        mux_lines.append(f"{indent}  .B({scan_in}),\n")
        mux_lines.append(f"{indent}  .S(shift),\n")
        mux_lines.append(f"{indent}  .Z(scan_d___uuf____{safe})\n")
        mux_lines.append(f"{indent});\n")

        dff_header = f"{indent}{cell} \\{inst}  (\n"
        new_block = "".join(mux_lines) + dff_header + body_new

        return new_block

    new_text = dff_pat.sub(repl, text)
    return new_text
```

Αυτόματη εισαγωγή MUX2 πριν από κάθε DFF και σύνδεση scan_in ανά chain

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*. New York, NY, USA: Computer Science Press, 1990.

[2] F. Brglez, D. Bryan, and K. Kozminski, "Notes on the ISCAS'89 benchmark circuits," Microelectronics Center of North Carolina (MCNC), Tech. Rep., 1989.

[3] A. B. Kahng and T. Spyrou, "The OpenROAD Project: Unleashing hardware innovation," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, 2021.

[4] The OpenROAD Project, "OpenROAD-flow-scripts (ORFS): RTL-to-GDSII flow scripts." [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts

[5] A. Carsello *et al.*, "Enabling reusable physical design flows with modular flow generation (mflowgen)," 2021. [Online]. Available: https://ctorng.com/pdfs/carsello-mflowgen-arxiv2021.pdf

[6] mflowgen, "mflowgen documentation." [Online]. Available: https://mflowgen.readthedocs.io/

[7] Magic VLSI, "Magic VLSI layout tool (Open Circuit Design)." [Online]. Available: https://opencircuitdesign.com/magic/

[8] A. Gaber *et al.*, "Fault, an open source DFT toolchain," in *Workshop on Open-Source EDA Technology (WOSET)*, 2019. [Online]. Available: https://woset-workshop.github.io/PDFs/2019/a13.pdf

[9] AUCOHL, "Fault: Open-source design-for-test (DFT) toolchain." [Online]. Available: https://github.com/AUCOHL/Fault

[10] Icarus Verilog, "Icarus Verilog documentation." [Online]. Available: https://steveicarus.github.io/iverilog/

[11] NCSU EDA, "FreePDK45 process design kit." [Online]. Available: https://eda.ncsu.edu/freepdk/freepdk45/

[12] OSCC-IP, "Nangate Open Cell Library." [Online]. Available: https://github.com/oscc-ip/nangate

[13] Si2 (Silicon Integration Initiative), "LEF/DEF downloads (standards)." [Online]. Available: https://si2.org/lef-def-downloads/

[14] *LEF/DEF 5.8 Language Reference*. 2017. [Online]. Available: https://coriolis.lip6.fr/doc/lefdef/lefdefref/lefdefref.pdf

[15] E. B. Eichelberger and T. W. Williams, "A logic design structure for LSI testability," in *Proc. Design Automation Conf. (DAC)*, 1977.

[16] IEEE, "IEEE Standard for Test Access Port and Boundary-Scan Architecture," IEEE Std 1149.1-2013, 2013.

[17] YosysHQ, "Yosys Open SYnthesis Suite documentation." [Online]. Available: https://yosyshq.readthedocs.io/projects/yosys/en/latest/

[18] L.-T. Wang, C.-W. Wu, and X. Wen, VLSI Test Principles and Architectures: Design for Testability. Burlington, MA, USA: Morgan Kaufmann, 2006.