ΑΤΕΙ ΚΑΛΑΜΑΤΑΣ
ΠΑΡΑΡΤΗΜΑ ΣΠΑΡΤΗΣ
ΤΜΗΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ :

«ΤΕΧΝΙΚΕΣ ΕΛΕΓΧΟΥ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΨΗΦΙΑΚΩΝ ΚΥΚΛΩΜΑΤΩΝ (*Digital Testing*) ΚΑΙ ΣΧΕΔΙΑΣΗ ΓΙΑ ΕΛΕΓΞΙΜΟΤΗΤΑ (*Design For Testability – DFT*)»

Νίκας Έρλιντ **Α.Μ:2005048**

Χριστοδουλόπουλος Γεώργιος **Α.Μ:2005045**

Επιβλέπων: ΓΙΑΝΝΗΣ ΛΙΑΠΕΡΔΟΣ

ΣΠΑΡΤΗ, 2011

Ευχαριστίες

Στο σημείο αυτό θα θέλαμε να εκφράσουμε τις ειλικρινείς μας ευχαριστίες σε όλους αυτούς τους ανθρώπους που συνέβαλλαν στο να φέρουμε εις πέρας την παρούσα Πτυχιακή Εργασία.

Πρώτα απ' όλα, θέλουμε να ευχαριστήσουμε τον επιβλέποντα της εργασίας αυτής, κ. **Λιαπέρδο** για την άψογη συνεργασία, τις πολύτιμες συμβουλές και την καθοδήγησή του σε όλη την διάρκεια της εκπόνησης αυτής της πτυχιακής εργασίας.

Επίσης θα θέλαμε να ευχαριστήσουμε όλους τους φίλους και συναδέλφους μας τόσο για τα όμορφα φοιτητικά χρόνια που περάσαμε μαζί όσο και για την ηθική υποστήριξη και κατανόησή τους, ιδιαίτερα κατά τη διάρκεια των τελευταίων μηνών της προσπάθειάς μας.

Θα θέλαμε επίσης να απευθύνουμε τις ευχαριστίες μας στους γονείς μας, οι οποίοι στήριξαν τις σπουδές μας με διάφορους τρόπους, φροντίζοντας για την καλύτερη δυνατή μόρφωσή μας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Οι προκλήσεις της εποχής μας σε προσωπικό αλλά και κοινωνικό επίπεδο είναι πολλές και εντείνονται από τη ραγδαία εξέλιξη των νέων τεχνολογιών. Γίνεται κατανοητό από την εξέλιξη αυτή, ότι είναι επιτακτική η ανάγκη προσαρμογής του ανθρώπου τόσο στις νέες αυτές καταστάσεις αλλά και στο νέο συνεχώς μεταβαλλόμενο κοινωνικό περιβάλλον.

Μετά τις υπολογιστικές μηχανές του Pascal και του Leibnitz, μπορεί να εντοπίσει κανείς τα ίχνη του πρώτου πλήρους ψηφιακού υπολογιστή στα σχέδια του CharlesBabbage για την Αναλυτική Μηχανή του(1832). Χρειάστηκε όμως η εξέλιξη των αυτόματων τηλεφωνικών κέντρων και των συστημάτων κρυπτογραφήσεως στη δεκαετία του1930, για να γίνει συνειδητή μία νέα οντότητα, το ψηφιακό σύστημα, και να αρχίσει η έρευνα, θεωρητική και πρακτική, για τη σχεδίαση και κατασκευή τέτοιων συστημάτων. Οι ανάγκες του Δευτέρου Παγκοσμίου Πολέμου επιτάχυναν την εξέλιξη της τεχνολογίας τους με την κατασκευή δύο μεγάλων για την εποχή εκείνη ηλεκτρονικών υπολογιστών, του Colossus στην Αγγλία(1943)και του ENIAC στις Ηνωμένες Πολιτείες (1946).Η ανακάλυψη του τρανζίστορ (1948) και του Ολοκληρωμένου Κυκλώματος (1958) έδωσαν την υλική βάση για τη σημερινή εξάπλωση αυτών των συστημάτων. Σήμερα η παρουσία, αφανής τις περισσότερες φορές, των ψηφιακών συστημάτων στη ζωή μας έχει καθιερώσει τον όρο "ψηφιακός" να συμβολίζει, πολλές φορές καταχρηστικά, την τελευταία λέξη της τεχνολογίας.

Η κοινωνία της πληροφορικής στηρίζεται κατά μεγάλο βαθμό στη συλλογή, αποθήκευση, επεξεργασία και ανταλλαγή πληροφοριών. Οι πληροφορίες αυτές είτε περιγράφουν το είδος και την υφή γεγονότων, καταστάσεων ή αντικειμένων είτε χαρακτηρίζουν ποσοτικά μεγέθη, όπως η διακύμανση της θερμοκρασίας ή το ύψος του τιμάριθμου. Από τα μεγέθη αυτά, είτε οφείλονται σε γεγονότα είτε σε καταστάσεις είτε σε αντικείμενα, μερικά είναι αναλογικά και τα άλλα είναι ψηφιακά.

Οι ψηφιακές τεχνικές χρησιμοποιούνται σήμερα εκτεταμένα και είναι σχεδόν αδύνατο να διανοηθούμε σύγχρονες ηλεκτρονικές συσκευές ξέχωρα από αυτές. Η βελτίωση των ψηφιακών μεθόδων έχει δώσει συσκευές μικρού μεγέθους, μικρού λειτουργικού κόστους, μεγάλης αξιοπιστίας και ικανοτήτων.

Η τεχνολογία των ολοκληρωμένων κυκλωμάτων , μέσα σε λίγα χρόνια, έχει παράγει νέες σειρές εξαρτημάτων, που κάνουν την εποχή του τρανζίστορ, τρεις δεκαετίες πριν, να φαντάζει αρκετά απόμακρη και σε πολλούς ακόμη και "ρομαντική".

Τα βασικά δομικά λογικά στοιχεία στη σχεδίαση συνδυαστικών συστημάτων είναι οι λογικές πύλες. Η σχεδίαση συνίσταται κυρίως στη διαδικασία της σύνδεσης αυτών των λογικών στοιχείων, που κυκλοφορούν στο εμπόριο, προκειμένου να επιτευχθεί ένα κύκλωμα, το οποίο θα εξυπηρετεί ένα στόχο.

Οι ψηφιακές τεχνικές αποτελούν ένα χρήσιμο εργαλείο όχι μόνο για τον σπουδαστή πεδίων μελέτης σχετικών με την ηλεκτρονική (από αυτοματισμούς έως τηλεπικοινωνίες), αλλά και για τους ερασιτέχνες εκείνους, που η αγάπη τους για τις ψηφιακές εφαρμογές τους καθιστά βαθύς γνώστες επιστημονικών θεμάτων.

Ψηφιακά είναι τα μεγέθη, και τα αντίστοιχα συστήματα, που οι καταστάσεις τους παίρνουν διακριτές τιμές. Παραδείγματα ψηφιακών συστημάτων είναι οι φωτεινοί σηματοδότες της τροχαίας, που μεταδίδουν τις πληροφορίες τους χρησιμοποιώντας τρία χρώματα διακριτά μεταξύ τους, οι ηλεκτρικοί διακόπτες, που οι καταστάσεις τους εκφράζονται με πληροφορίες της μορφής ανοικτός - κλειστός, κ.λπ.

Γενικώς, ψηφιακό ονομάζεται ένα μέγεθος του οποίου οι καταστάσεις και κατ' επέκταση οι πληροφορίες, που τις καταγράφουν, εκφράζονται ή μπορούν να παρασταθούν με τα στοιχεία ενός αριθμήσιμου συνόλου, όπως π.χ. είναι οι ακέραιοι αριθμοί.

Τα ψηφιακά μεγέθη προέρχονται από συστήματα που είτε έχουν από τη φύση τους ψηφιακή συμπεριφορά, όπως π.χ. το παιχνίδι της ρουλέτας, είτε σκοπίμως κατασκευάζονται, όπως π.χ. οι ψηφιακοί υπολογιστές, για να βοηθήσουν στην επεξεργασία δεδομένων, δηλ. πληροφοριών που παράγονται από άλλα συστήματα. Θα μπορούσαμε να θεωρήσουμε ότι ένα ψηφιακό σύστημα, που χρησιμοποιείται για την επεξεργασία δεδομένων τυπικά εμπίπτει στον ανωτέρω ορισμό. Είναι ένα σύστημα που δέχεται ψηφιακά δεδομένα και εξ αυτών παράγει νέα ψηφιακά δεδομένα σύμφωνα με κανόνες που υλοποιούν τις βασικές πράξεις της λογικής, καθώς και κάθε πολύπλοκη σύνθεση αυτών.

Ένα ψηφιακό σύστημα επεξεργασίας δεδομένων, γενικά, μπορεί να παρασταθεί σαν ένα μηχάνημα (κλειστό κιβώτιο), που έχει μία σειρά από εισόδους, από όπου εισέρχονται τα δεδομένα X, τα οποία μετασχηματίζονται σύμφωνα με μία "λογική" σχέση F που έχει υλοποιηθεί στο εσωτερικό του μηχανήματος, και τα παραγόμενα νέα δεδομένα Y=F(X) γίνονται προσιτά μέσω των εξόδων του.

Τη λειτουργία ενός ψηφιακού συστήματος μπορούμε να την περιγράψουμε κατά πολλούς τρόπους, ανάλογα με το βαθμό λεπτομέρειας που μας ενδιαφέρει, που ονομάζονται ιεραρχικά επίπεδα περιγραφής. Σκοπός του κάθε επιπέδου περιγραφής είναι να παρέχει τόσες λεπτομέρειες, όσες απαιτούνται, για να γίνει κατανοητή η λειτουργία στο ζητούμενο επίπεδο, αποφεύγοντας να επιβαρύνει την περιγραφή με πλεονάζοντα στοιχεία. Τα κυριότερα επίπεδα είναι:

α) Το επίπεδο της συμπεριφοράς. Περιγράφει τις συναρτήσεις και γενικά τις σχέσεις που υπάρχουν μεταξύ των σημάτων εισόδου και εξόδου.
β) Το επίπεδο μεταφοράς. Περιγράφει ομαδικά τα σήματα που πρέπει να απομνημονευθούν (*καταχωρητές*) και τους διαδοχικούς μετασχηματισμούς τους (συναρτήσεις), καθώς μετακινούνται από τη μία κατάσταση απομνημονεύσεως στην επόμενη.
γ) Το λογικό επίπεδο. Περιγράφει το σύστημα σαν ένα αφηρημένο (ιδεατό) μηχάνημα, για το οποίο μας ενδιαφέρει να γνωρίζουμε μόνο το λογικό τρόπο με τον οποίο εργάζεται, δηλ. τους αναλυτικούς κανόνες με τους οποίους συνδυάζει τις πληροφορίες εισόδου, παράγει ενδιάμεσες πληροφορίες κ.λπ., για να δημιουργήσει τις πληροφορίες εξόδου, αδιαφορώντας για τη φυσική μορφή που έχουν στο εσωτερικό του οι πληροφορίες και ο μηχανισμός επεξεργασίας,
δ) Το κυκλωματικό επίπεδο, που αναφέρει το είδος των λογικών στοιχείων (λογικών πυλών, μνημονικών στοιχείων) που χρησιμοποιούνται από το σύστημα και τις μεταξύ τους διασυνδέσεις (κύκλωμα),
ε) Το ηλεκτρικό επίπεδο (για τα ηλεκτρονικά συστήματα), που βρίσκεται ιεραρχικά χαμηλότερα από το κυκλωματικό και περιέχει πρόσθετες πληροφορίες για τα ηλεκτρικά κυκλώματα που υλοποιούν τις πύλες, και
στ) Το επίπεδο υλοποιήσεως, όπου λαμβάνονται υπ' όψιν εκτός της λογικής και οι φυσικοί περιορισμοί, τους οποίους συνεπάγεται η φυσική (υλική) μορφή του συστήματος, π.χ. υλικό κατασκευής των τρανζίστορ, γεωμετρικό σχήμα τους, διάταξη τους στο χώρο, εξαερισμός κ.λπ.

Η "λογική" ενός ψηφιακού συστήματος μπορεί να υλοποιηθεί με διαφόρους τρόπους, π.χ. να κατασκευασθεί από ηλεκτρονικές πύλες, όπως είναι συνήθως τα σημερινά συστήματα, οπότε έχουμε ένα ηλεκτρονικό ψηφιακό σύστημα. Η λογική μπορεί ακόμη να πραγματοποιηθεί με οδοντωτούς τροχούς και άξονες, όπως ήταν τα παλαιότερα συστήματα ή και μερικά σημερινά σε ειδικευμένες εφαρμογές, και τότε το σύστημα ονομάζεται μηχανικό. Το σύστημα θα μπορούσε να είναι υδραυλικό, που η λογική του υλοποιείται με βαλβίδες και σωλήνες μέσα στους οποίους κυκλοφορεί κάποιο υγρό με πίεση. Θα μπορούσε ο φορέας της πληροφορίας να είναι μία ή περισσότερες φωτεινές δέσμες, που η ένταση ή η φάση τους διαμορφώνεται με τη βοήθεια οπτικών φακών και διαφραγμάτων και ονομάζεται οπτικό σύστημα..

Κεφάλαιο 1

1.1 Εισαγωγή

Τα ψηφιακά κυκλώματα κατασκευάζονται κυρίως με χρήση ολοκληρωμένων κυκλωμάτων (που λέγονται για συντομία ICs – *IntegratedCircuits*). Κάθε IC είναι ένας μικρός κρύσταλλος ημιαγωγού πυριτίου (*Si*), καλούμενος CHIP. Το CHIP περιλαμβάνει ηλεκτρικά στοιχεία όπως τρανζίστορς, διόδους αντιστάσεις και πυκνωτές. Τα στοιχεία αυτά είναι συνδεδεμένα μέσα στο CHIP ώστε να σχηματίζουν ένα ηλεκτρονικό κύκλωμα. Το CHIP τοποθετείται πάνω σε μεταλλικό ή πλαστικό στέλεχος και οι συνδέσεις συγκολλούνται σε εξωτερικά "ποδαράκια", έτσι σχηματίζεται το IC. Τα ολοκληρωμένα κυκλώματα διαφέρουν από τα συμβατικά κυκλώματα διακριτών στοιχείων στο ότι τα στοιχεία τους δεν μπορούν να διαχωριστούν ή να αποσυνδεθούν από το κύκλωμα του εσωτερικού του ολοκληρωμένου πακέτου. Η σύνδεση του ολοκληρωμένου με το υπόλοιπο εξωτερικό κύκλωμα γίνεται μόνο με τους εξωτερικούς του ακροδέκτες.

Πλεονεκτήματα των IC:

- Πολύ μικρό μέγεθος
- Χαμηλή τιμή κόστους μαζικής παραγωγής.
- Μικρή κατανάλωση ισχύος.
- Υψηλή αξιοπιστία λειτουργίας.
- Υψηλή ταχύτητα λειτουργίας.
- Μείωση εξωτερικών καλωδιακών συνδέσεων.

Αυτά τα κυκλώματα, πλην των πλεονεκτημάτων μεγέθους και βάρους, έχουν το βασικό πλεονέκτημα, ότι κατασκευάζονται με μεθόδους μαζικής παραγωγής, έτσι ώστε το κόστος τους να είναι πολύ μικρότερο από το ανάλογο κόστος διακριτών στοιχείων. Άλλο πλεονέκτημα είναι ότι, τα περισσότερα καταναλώνουν πολύ μικρή ισχύ και επιπλέον δεν υπάρχει ανάγκη για κολλήσεις και καλωδιώσεις (όλα τα στοιχεία και οι συνδέσεις κατασκευάζονται ταυτόχρονα στον κρύσταλλο).Σήμερα η περιπλοκότητα των ολοκληρωμένων κυκλωμάτων ολοένα και αυξάνεται, ενώ οι τιμές κόστους συνεχίζουν να πέφτουν.

1.2 Βασικές λογικές πράξεις – λογικές πύλες

Οι λογικές πύλες είναι στοιχειώδεις ψηφιακές διατάξεις, που αποτελούν τους δομικούς λίθους γενικότερων λογικών κυκλωμάτων. Οι λογικές πύλες δέχονται στην είσοδο ή στις εισόδους τους ένα ή περισσότερα δεδομένα και παρέχουν στην έξοδό τους το αποτέλεσμα, δηλαδή αποτελούν την υλοποίηση σε ηλεκτρονικό κύκλωμα των διαφόρων λογικών συναρτήσεων. Οι πύλες αποτελούνται από βασικά στοιχεία της ηλεκτρονικής όπως διόδους και τρανζίστορ. Έχουν τουλάχιστον μία ή δύο εισόδους και μία μόνο έξοδο. Τα κυκλώματα λογικής αποτελούνται από πύλες συνδεδεμένες η μία με την άλλη ώστε να πραγματοποιούν διάφορες πράξεις (π.χ. αθροίσματα αριθμών).

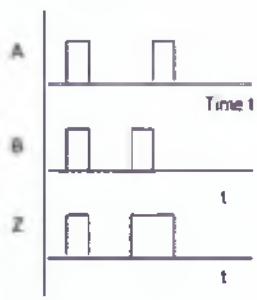
-Οι βασικές πύλες είναι:

- Πύλη NOT (*ΟΧΙ*)
- Πύλη AND (*ΚΑΙ*)
- Πύλη OR (*Η*)
- Πύλη XOR (*Αποκλειστικό Η*)
- Συνδυασμοί των παραπάνω

1.2.1 Πύλη NOT (*ΟΧΙ*)

Η πύλη **NOT** εκτελεί την πράξη της αντιστροφής. Το λογικό της σύμβολο φαίνεται παρακάτω σχήμα. Έχει μόνο μία είσοδο και μία έξοδο. Η λειτουργία της πύλης **NOT** περιγράφεται από τον παρακάτω πίνακα αλήθειας. Η πύλη αντιστρέφει την τιμή της εισόδου (εάν η είσοδος είναι λογικό **1** η έξοδος γίνεται λογικό 0, εάν η είσοδος είναι λογικό 0 η έξοδος γίνεται λογικό 1).

A	X
0	1
1	0

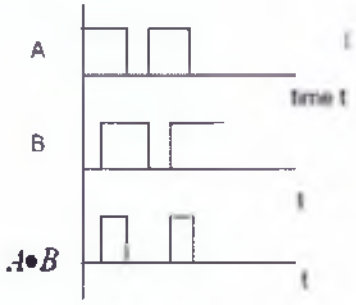


$$Z = \overline{A}$$

1.2.2 Πύλη AND (*ΚΑΙ*)

Η πύλη **AND** εκτελεί την λογική πράξη **ΚΑΙ**. Το σύμβολό της φαίνεται παρακάτω. Έχει δύο ή περισσότερες εισόδους και μία έξοδο. Η λειτουργία της πύλης **AND** περιγράφεται από τον πίνακα αλήθειας, που έχει δύο στήλες, όσες και οι είσοδοι, και μία τρίτη στήλη για την έξοδο **X**. Από τον πίνακα αλήθειας συμπεραίνουμε ότι για να είναι η έξοδος λογικό 1 θ α πρέπει η είσοδος **A ΚΑΙ** η είσοδος **B** να είναι λογικό 1. Αν μία από τις εισόδους είναι λογικό 1 ή και οι δύο είναι λογικό 0 τότε και η έξοδος είναι 0.

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1



$$Z = A \cdot B$$

1.2.3 Πύλη OR (*Η*)

Η πύλη **OR** εκτελεί την λογική πράξη Η. Το σύμβολο της πύλης φαίνεται παρακάτω. Έχει δύο ή περισσότερες εισόδους και μία έξοδο. Η λειτουργία της πύλης **OR** περιγράφεται στον παρακάτω πίνακα αλήθειας όπου συμπεραίνουμε ότι για να είναι η έξοδος μιας πύλης **OR** ίση με λογικό 1 ή η μια είσοδος ή η άλλη ή και οι δύο πρέπει να είναι λογικό 1. Εάν και οι δύο είσοδοι είναι μηδέν τότε και η έξοδος θα είναι λογικό 0.

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

$$Z = A + B$$

1.2.4 Πύλη XOR (*Αποκλειστικό Η*)

Η πύλη **XOR** εκτελεί την λογική πράξη "αποκλειστικό**Η**". Το σύμβολό της φαίνεται παρακάτω. Έχει δύο ή περισσότερες εισόδους και μία έξοδο. Από τον πίνακα αλήθειας μιας πύλης **XOR** παρατηρούμε ότι όταν μία από τις δύο εισόδους είναι λογικό 1, τότε η έξοδος είναι και αυτή λογικό 1.

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

$$Z = A \oplus B$$

1.2.5 Πύλες NAND, NOR, XNOR

Η πύλη **NOT** μπορεί να συνδυαστεί με τις άλλες πύλες που περιγράψαμε παραπάνω και να προκύψουν τρεις ακόμη πύλες. Έτσι με το συνδυασμό των **AND** και **NOT** προκύπτει η **NAND**, με το συνδυασμό **OR** και **NOT** προκύπτει η **NOR** και τέλος με το συνδυασμό **XOR** και **NOT** προκύπτει η **XNOR**.

NAND

A	B	X
0	0	1
0	1	1
1		1
1	1	0

NOR

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

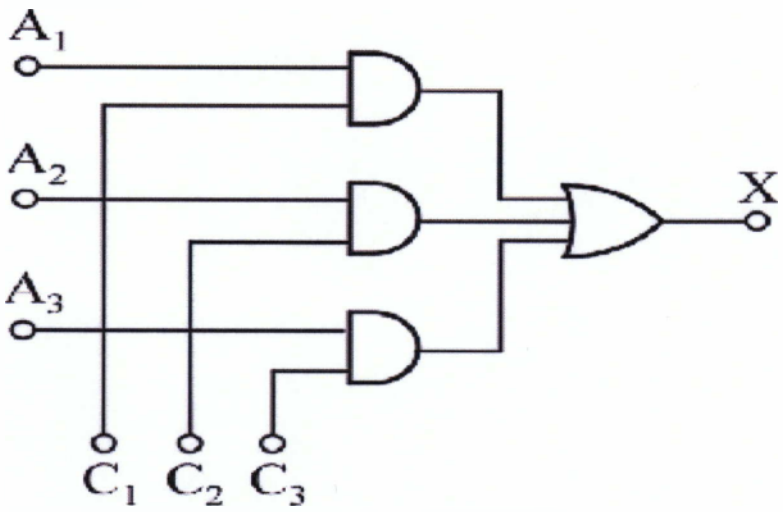
XNOR

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

$$Z = \overline{A \cdot B} \quad Z = \overline{A + B} \quad Z = \overline{A \oplus B}$$

1.3 Έλεγχος και πολύπλεξη λογικών σημάτων

Κάθε πύλη μπορεί να δράσει ως **ψηφιακός διακόπτης** (ή **μεταγωγός**) (*digitalswitch*). Εάν στη μία είσοδο μιας πύλης ANDεφαρμοσθεί ένα λογικό σήμα Α (π.χ. μια αλληλουχία καταστάσεων 0 και 1), τότε εάν η λογική κατάσταση που εφαρμόζεται στηδεύτερη είσοδο, η οποία δρα ως **είσοδος ελέγχου** (*controlinput*),είναι 1, στην έξοδο της πύλης θα εμφανίζεται το σήμα Α (X =A * 1 = A). Αντίθετα, εάν εφαρμοσθεί λογική κατάσταση 0 η έξοδος θα "κλειδωθεί" σε κατάσταση 0 (X = A * 0 = 0). Ανάλογα, εάν ως ψηφιακός διακόπτης χρησιμοποιηθεί πύλη OR, τότε το λογικό σήμα Α θα εμφανίζεται στην έξοδο X, όταν στην είσοδο ελέγχου εφαρμόζεται λογική κατάσταση 0 (X = A + 0 =A). Αντίθετα, εάν εφαρμοσθεί 1 η έξοδός της θα "**κλειδωθεί**" σε κατάσταση 1 (X = A + 1 = 1).

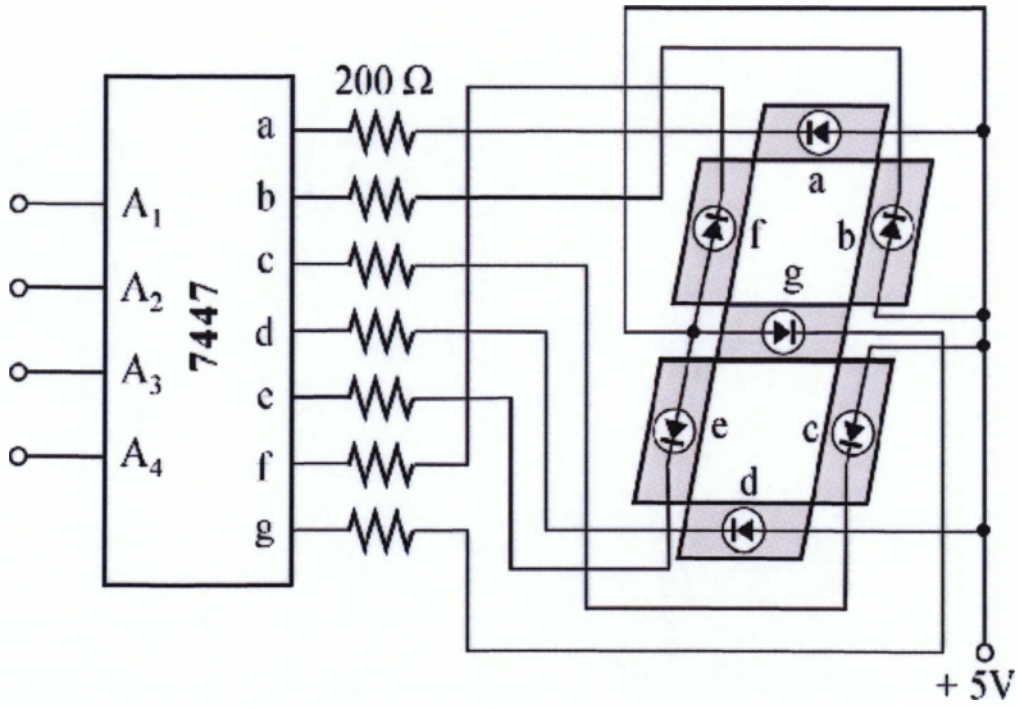


Σχήμα 1.3:Πολυπλέκτης ψηφιακού σήματος (*τριών εισόδων*).

Στο παραπάνω σχήμα απεικονίζεται κύκλωμα **ψηφιακού πολυπλέκτη** (*digitalmultiplexer*). Από τη συνδεσμολογία του κυκλώματος των πυλών είναι προφανές ότι το λογικό σήμα εξόδου του κυκλώματος αποδίδεται από την εξίσωση X = A1 * C1 + A2 * C2 + A3 * C3.
Η τελευταία εξίσωση διαβάζεται ως εξής: "Το X είναι *ΑΛΗΘΕΙΑ* ("1") όταν το A1 *ΚΑΙ* το C1 είναι *ΑΛΗΘΕΙΑ*, *Ή* το A2 *ΚΑΙ* το C2 είναι *ΑΛΗΘΕΙΑ*, *Ή* το A3 *ΚΑΙ* το C3 είναι *ΑΛΗΘΕΙΑ*".
Έτσι, με σήματα ελέγχου C1 = 0, C2 = 1 και C3 = 0, το σήμα εξόδου θα είναι εκείνο της εισόδου A2, επειδή: X = A1.0 + A2.1 + A3.0 = 0 + A2 + 0 = A2.

1.4 Αποκωδικοποιητές

Οι **αποκωδικοποιητές** (*decoders)* είναι συνδυασμοί λογικών πυλών που διαθέτουν δύο ή περισσότερες (M) εισόδους και μία ή περισσότερες (N) εξόδους. Με αποκωδικοποιητές είναι δυνατή η αλλαγή της μορφής των δυαδικών λέξεων ή αριθμητικών παραστάσεων. Έτσι σε κάθε κατάσταση εισόδου AM AM−1 ...A2 A1 αντιστοιχεί μία κατάσταση εξόδου XM XM−1... X2 X1. Η αλλαγή αυτή συνήθως εξυπηρετεί ανάγκες κωδικοποίησης και αποκωδικοποίησης ψηφιακών σημάτων. Τυπικό παράδειγμα αποκωδικοποιητή είναι ο αποκωδικοποιητής **BCD προς αριθμό 7τμημάτων** (BCD to 7-segment numberdecoder) ή αποκωδικοποιητής "4-προς-7". Ο αποκωδικοποιητής αυτός δέχεται στις τέσσερις εισόδους τους αριθμούς 0,1,..9 κωδικοποιημένους στο δυαδικό σύστημα (BCD: BinaryCodedDecimal), δηλ. τις παραστάσεις 00002 = 0, 00012 = 1, ..., 10012 = 9. Οι επτά έξοδοί του αποκτούν λογικές καταστάσεις, οι οποίες αντιστοιχούν στα τμήματα που πρέπει να φωτισθούν (κατάσταση "0") ή να παραμείνουν σκοτεινά (κατάσταση "1"), ώστε να εμφανισθεί ο αριθμός εισόδου (στον δεκαδικό του συμβολισμό). Το κύκλωμα αυτό διατίθεται ως ολοκληρωμένο κύκλωμα, διαθέτει επιπλέον εισόδους ελέγχου και αποτελείται συνολικά από 44 λογικές πύλες και αντιστροφείς.



Σχήμα 1.4: Αριθμός επτά φωτιζόμενων τμημάτων και σύνδεση με αποκωδικοποιητή "4-προς-7".

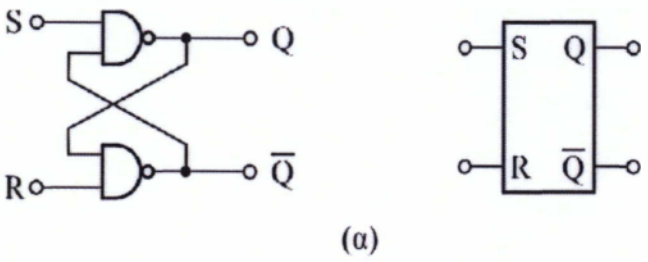
Πίνακας αλήθειας αποκωδικοποιητή "4 προς 7"

Δεκαδικός αριθμός	Είσοδοι A_3 A_2 A_1 A_0	Έξοδοι a	b	c	d	e	f	g	Οπτική παρουσίαση
0	0 0 0 0	0	0	0	0	0	0	1	0
1	0 0 0 1	1	0	0	1	1	1	1	1
2	0 0 1 0	0	0	1	0	0	1	0	2
3	0 0 1 1	0	0	0	0	1	1	0	3
4	0 1 0 0	1	0	0	1	1	0	0	4
5	0 1 0 1	0	1	0	0	1	0	0	5
6	0 1 1 0	1	1	0	0	0	0	0	6
7	0 1 1 1	0	0	0	1	1	1	1	7
8	1 0 0 0	0	0	0	0	0	0	0	8
9	1 0 0 1	0	0	0	1	1	0	0	9

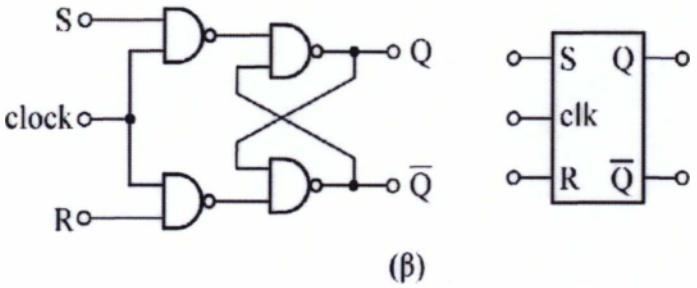
1.5 Φλιπ-φλοπ

Τα **φλιπ-φλοπ**(*flip-flop*, **FF**) είναι ψηφιακά κυκλώματα με δύο σταθερές καταστάσεις εξόδου, ικανά να μεταπίπτουν από τη μια στην άλλη, όταν δεχθούν ένα κατάλληλο σήμα ελέγχου και να παραμένουν σταθερά στη νέα κατάσταση μετά την απομάκρυνση του σήματος σε αντίθεση με τους μονοσταθερούςπολυδονητές. Έτσι, τα φλιπ-φλοπ είναι τυπικά **δισταθερά**(bistable) κυκλώματα. Όλοι οι τύποι φλιπ-φλοπ έχουν δύο συμπληρωματικές εξόδους Q και Q, όπως ακριβώς και οι μονοσταθεροίπολυδονητές. Στο εξής ως έξοδος θα εννοείται μόνο η Q, εκτός εάν αναφέρεται διαφορετικά. Οι διάφοροι τύποι φλιπ-φλοπ χαρακτηρίζονται από επιπλέον εισόδους. Τυπικές είσοδοι ελέγχου και η δράση τους είναι οι ακόλουθες:Είσοδος *CLOCK (ή T)*. Εάν συντρέχουν οι απαραίτητες προϋποθέσεις, λογική μετάπτωση στην είσοδο αυτή αντιστρέφει τη λογική κατάσταση της εξόδου. Το εάν υπάρχουν και ποιες είναι αυτές οι προϋποθέσεις, εξαρτάται από τον συγκεκριμένο τύπο του φλιπ-φλοπ.Είσοδος *SET (ή PRESET)*. Λογική μετάπτωση στην είσοδο αυτή **θέτει πάντοτε** την έξοδο στην κατάσταση 1 (Q = 1).Είσοδος *CLEAR (ή RESET)*. Λογική μετάπτωση στην είσοδο αυτή **επαναφέρει πάντοτε** την έξοδο στην κατάσταση 0 (Q = 0).

Το ποια μετάπτωση (0 → 1 ή 1 → 0) είναι η δραστική για την κάθε είσοδο, εξαρτάται από τον τύπο του φλιπ-φλοπ. Οι κυριότεροι τύποι φλιπ-φλοπ και η εσωτερική δομή των απλούστερων από αυτά περιγράφονται συνοπτικά στη συνέχεια.

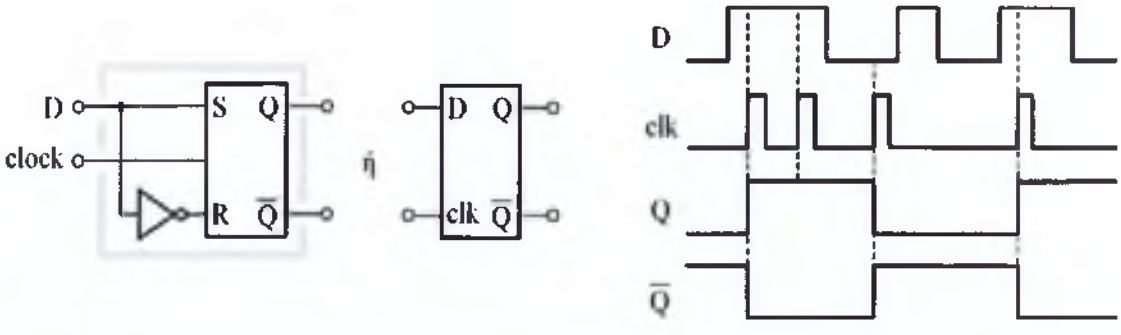


Είσοδοι S	R	Έξοδοι Q	$\overline{Q}$
0	0	απρόβλεπτο	
0	1	0	1
1	0	1	0
1	1	ως είχαν	



Είσοδοι S	R	clock	Έξοδοι Q	$\overline{Q}$
0	0	0	ως είχαν	
0	1	0	"	
1	0	0	"	
1	1	0	"	
0	0	1	"	
0	1	1	0	1
1	0	1	1	0
1	1	1	απρόβλεπτο	

Σχήμα **1.5 (α) & 1.5 (β):** Κύκλωμα, συμβολισμός και πίνακας αλήθειας: **(α)** φλιπ-φλοπR-S, **(β)** φλιπ-φλοπRS με είσοδο clock (*clk*).



Σχήμα 1.5 (γ): Συμβολισμός απλού φλιπ-φλοπ D και τυπική αλληλουχία σημάτων εισόδου-εξόδου.

1.5.1 Φλιπ-φλοπ R-S (*Reset-Set*).

Είναι ο θεμελιώδης τύπος φλιπ-φλοπ και αποτελείται από δύο πύλες **NAND**. Το κύκλωμα, ο συμβολισμός και ο πίνακας αλήθειας του δείχνονται στο Σχήμα 4.4.6α. Για S = 1 και R = 0 το φλιπ-φλοπ εξαναγκάζεται στην κατάσταση Q = 1. Για S = 0 και R = 1 εξαναγκάζεται στην κατάσταση Q = 0. Για S = 1 και R = 1 η Q παραμένει στην υπάρχουσα κατάσταση. Για S = 0 και R = 0 και οι δύο έξοδοι θα οδηγηθούν στην *"ανώμαλη"* κατάσταση Q = 1 και Q= 1, όταν όμως γίνουν πάλι S = 1 και R = 1 είναι απρόβλεπτο ποια από τις δύο σταθερές καταστάσεις θα επικρατήσει.

1.5.2 Φλιπ-φλοπ R-S μεείσοδο clock (*Clocked R-S flip-flop*).

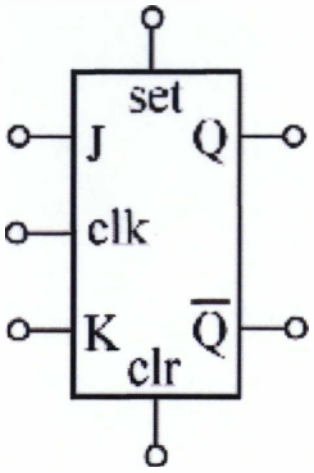
Μετάπτωση από τη μια κατάσταση στην άλλη (που καθορίζεται από τα σήματα στις εισόδους R, S) θα πραγματοποιηθεί μόνο όταν εφαρμοσθεί 1 στην είσοδο clock (*clk*).

1.5.3 Απλό φλιπ-φλοπ D.

Προέρχεται από το προηγούμενο φλιπ-φλοπ με σύνδεση των εισόδων R και S με έναν αντιστροφέα. Για clock = 0 οι έξοδοι παραμένουν ως έχουν. Για clock = 1 θα είναι πάντοτε Q= D. Αυτό σημαίνει ότι μετάπτωση 0 → 1 στην είσοδο clock (clk) ισοδυναμεί με την εντολή *"Γράψε στην Q ό,τι βλέπεις στην D"*. Επομένως το φλιπ-φλοπ D μπορεί να χρησιμοποιηθεί ως στοιχείο αποθήκευσης (ή μνήμης) της κατάστασης που εφαρμόζεται στην είσοδο D (data). Ο συμβολισμός και τυπικό παράδειγμα αλληλουχίας των σημάτων εισόδου και εξόδου ενός φλιπ-φλοπ D δείχνεται στο πιο πάνω σχήμα.

1.5.4 Φλιπ-φλοπ J-K.

Είναι το φλιπ-φλοπ με τις περισσότερες εφαρμογές. Στο παρακάτω σχήμα δείχνεται ο συμβολισμός του και ο πίνακας αλήθειας. Οι ιδιότητές του συνοψίζονται ως εξής: με J = 1και K = 1 η κατάσταση στην Q θα αλλάξει με κάθε μετάπτωση 1 → 0 στην είσοδο clock. Όταν είναι J = 1 και K = 0 με την επόμενη μετάπτωση 1 → 0 στην είσοδο clock γίνεται Q = 1 και αποτρέπεται πλέον η μετάπτωση 1 → 0 στην Q. Όταν είναι J = 0 και K = 1 με την επόμενη μετάπτωση 1 → 0 στην είσοδο clock γίνεται Q = 0 και αποτρέπεται πλέον η μετάπτωση 0 → 1 στην Q. Με J = 0 και K = 0 το φλιπ-φλοπ παραμένει στην τρέχουσα κατάσταση, ανεξάρτητα από τις όποιες μεταπτώσεις συμβούν στην είσοδο clock. Μεταπτώσεις 1 → 0 στις εισόδους set και clear (clr) εξαναγκάζουν κανονικά την Q σε 1 και 0, αντιστοίχως.

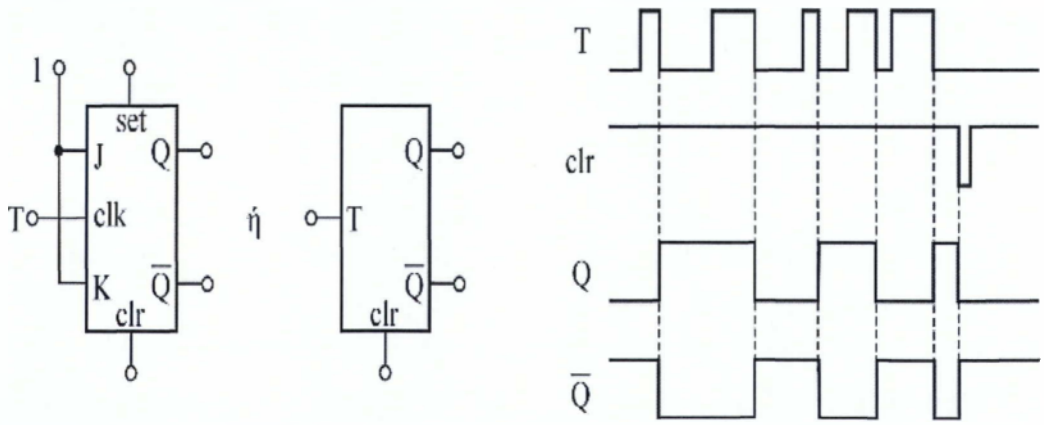


Είσοδοι		Έξοδοι	
J	K	Q	$\overline{Q}$
0	0	ως είχαν	
0	1	0	1
1	0	1	0
1	1	αλλαγή	

Συμβολισμός και πίνακας αλήθειας ενός φλιπ-φλοπ**J-K**.

1.5.5 Φλιπ-φλοπ Τ.

Η βασική λειτουργία των φλιπ-φλοπ Τ συνοψίζεται στα εξής: Κάθε μετάπτωση 1 → 0 στην είσοδο Τ (ή clock) αλλάζει την κατάσταση εξόδου του φλιπ-φλοπ. Μια μετάπτωση 1 → 0 στην είσοδο clear (clr) επαναφέρει τις εξόδους στην αρχική τους κατάσταση (Q = 0, Q = 1). Ο συμβολισμός και τυπικό παράδειγμα αλληλουχίας των σημάτων εισόδου και εξόδου ενός φλιπ-φλοπ Τ δείχνεται στο πιο κάτω σχήμα που ακολουθεί. Ένα φλιπ-φλοπ J-K με τις εισόδους J και K σε κατάσταση 1 αντιστοιχεί σε ένα φλιπ-φλοπ Τ.

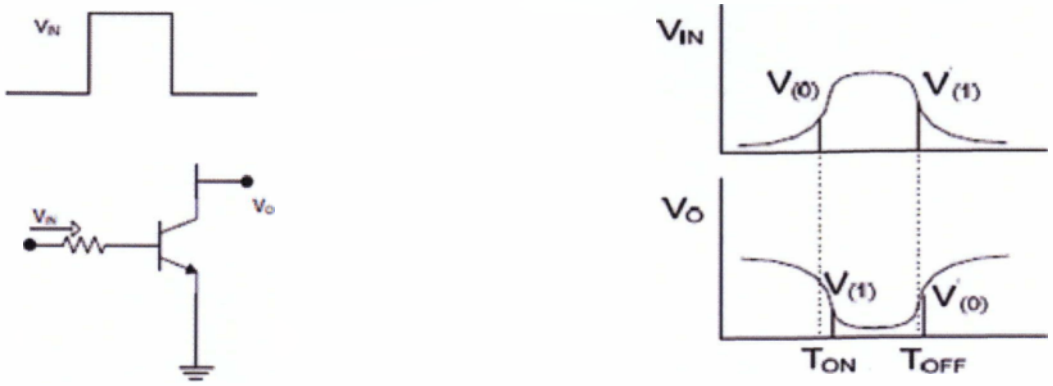


Συμβολισμός φλιπ-φλοπ**Τ** και τυπική αλληλουχία σημάτων εισόδου-εξόδου.

1.6 Χαρακτηριστικά

Τα χαρακτηριστικά των ψηφιακών λογικών οικογενειών συνήθως συγκρίνονται με ανάλυση του κυκλώματος της βασικής πύλης κάθε οικογένειας. Το πιο σημαντικά χαρακτηριστικά που χρησιμοποιούνται για τη σύγκριση αυτή, είναι τα ακόλουθα:

1)Ταχύτητα.: Χρόνος καθυστερήσεως της διαδόσεως –propagationdelayTpd.



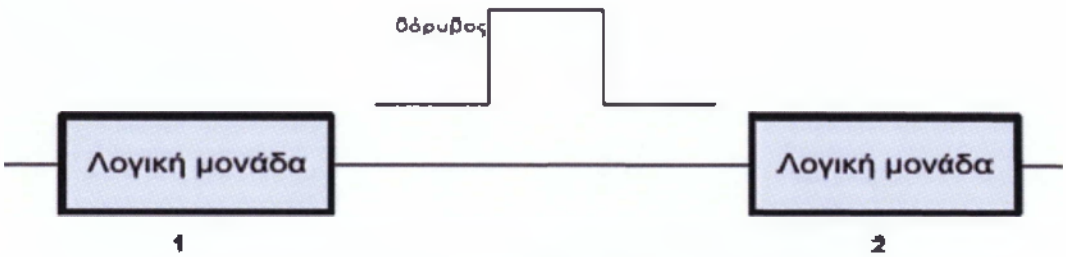
Σχήμα 1.6 (α).Σχήμα 1.6 (β).

Αν υποθέσουμε ότι στη βάση του τρανζίστορ του σχ: 1.6(β) επιβάλλεται ο παλμός του σχ: 1.6(α). Τότε υπάρχει κάποιος χρόνος αντιδράσεως της πύλης αντιστροφής. Στο σχήμα 1.6(γ) και στην κυματομορφή του παλμού εισόδου **VIN** η στάθμη τάσης V(0) παριστάνει την τάση του κατωφλίου από τη χαμηλή στάθμη στην υψηλή. Για την μετάβαση από τη ψηλή στάθμη στη χαμηλή αυτή η τάση κατωφλίου παριστάνεται με τη V'(1). Αντίστοιχα στην έξοδο προκαλείται μια μεταβολή από τη ψηλή στάθμη στη χαμηλή. Οι τάσεις κατωφλίου αντίστοιχα όπως φαίνονται και στο σχ: 1.6(γ) είναι V(1) και V'(0). Ορίζεται ως TON η διαφορά: TON=V(1)-V(0), δηλαδή ο χρόνος που χρειάσθηκε η πύλη να αλλάξει κατάσταση, όταν αλλάζει η είσοδος. Αντίστοιχα για το τέλος του παλμού εισόδου και την τελική αλλαγή στην έξοδο έχουμε TOFF=V'(0)-V'(1).

2) Πλήθος εισόδων (*fan-in*) – πλήθος εξόδων (*fan-out*). Ο ορισμός των παραπάνω φαίνεται στο ακόλουθο σχήμα 1.6(γ).

Σχήμα 1.6 (γ).

3) Αναισθησία θορύβου (*noiseimmunity*).



Σχήμα 1.6 (δ).

Είναι η ικανότητα των πυλών στο να παρέχουν κάποιο περιθώριο θορύβου, ώστε να μην αλλάζουν κατάσταση, (π.χ. η Λογική Μονάδα 2), από τυχόν θόρυβο στη γραμμή μεταφοράς, επηρεάζεται και επηρεάζει τα επόμενα λογικά κυκλώματα. Βλέπε σχ: 1.6(δ).

4) Απαιτήσεις για τροφοδοσία: Τα πιο απλά απαιτούν μόνο μια τάση τροφοδοσίας. Άλλες οικογένειες πυλών απαιτούν περισσότερες τάσεις τροφοδοσίας θετικές ή και αρνητικές.
5) Καταναλισκόμενη ισχύς (powerdissipation).
6) Καταλληλότητα για κατασκευή ολοκληρωμένου κυκλώματος.
7) Θερμοκρασιακή περιοχή λειτουργίας.
8) Πλήθος διαθεσίμων λογικών πράξεων (δυνατότητα για ενσύρματη λογική).
9) Κόστος.

Τέλος ένας παράγοντας για την εκλογή μιας συγκεκριμένης κατηγορίας πυλών είναι η προσωπική εκτίμηση και πείρα του σχεδιαστή μηχανικού.

1.7 Οικογένειες Λογικών Κυκλωμάτων

Από την εποχή που πρωτοεισήχθη η πρώτη λογική οικογένεια ολοκληρωμένων κυκλωμάτων, με διπολικά κορεσμένα τρανζίστορ, υπήρξαν πολλές πρόοδοι στην κατασκευαστική τεχνολογία, με αποτέλεσμα την παραγωγή νέων γενεών ή οικογενειών διπολικών εξαρτημάτων. Σήμερα στα ψηφιακά συστήματα υπάρχουν οι εξής οικογένειες:

η **RTL**

η **DTL**

η **TTL**

η **ECL** ή **CML**

η **IIL**

η **MOSFET** ή **MOSTEL** και η **CMOSL**.

Οι κυριότερες οικογένειες είναι η **TTL** και η **CMOS**.

-Παρακάτω δίνονται αναλυτικά οι οικογένειες λογικών κυκλωμάτων, όπως:

α) Η οικογένεια λογικής αντιστάσεων-τρανζίστορ, *RTL*:Είναι μία από τις πρώτες τεχνικές για τη υλοποίηση λογικών πυλών, των βασικών στοιχείων για τη δόμηση των ψηφιακών συνδυαστικών κυκλωμάτων. Η οικογένεια RTL χρησιμοποιεί αντιστάσεις και τρανζίστορ. Σε κάθε είσοδο αντιστοιχεί μία αντίσταση και ένα τρανζίστορ, όπου όλα δε τα τρανζίστορ είναι συνδεδεμένα παράλληλα.

β) Η οικογένεια λογικής διόδων-τρανζίστορ, *DTL*:Η οικογένεια DTL χρησιμοποιεί διόδους σε κάθε δε είσοδο του λογικού κυκλώματος αντιστοιχεί και μία δίοδος. Στην έξοδο υπάρχει κύκλωμα τρανζίστορ, που λειτουργεί ως ενισχυτής. Η οικογένεια αυτή έχει σαν χαρακτηριστικό της τη χαμηλή κατανάλωση ισχύος και σαν μειονέκτημά της τη χαμηλή ταχύτητα και προβλήματα θορύβου.

γ) Η οικογένεια λογικής τρανζίστορ-τρανζίστορ, *TTL*:Είναι βελτιωμένη έκδοση της οικογένειας DTL, όπου αντί των διόδων εισόδου χρησιμοποιούνται τα διπολικά τρανζίστορ πολλαπλών εκπομπών, ακολουθούμενα από άλλα διπολικά τρανζίστορ, που λειτουργούν μεταξύ αποκοπής και κορεσμού.

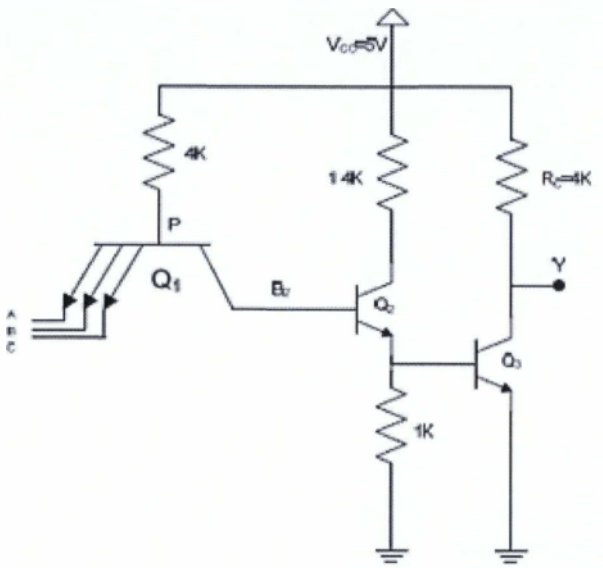
δ) Η οικογένεια λογικής ζεύξης εκπομπού, *ECL* ή *CML*:Λέγεται και ¨λογική μη κορεσμού¨, διότι ο συμβατικός τύπος της χρησιμοποιεί διπολικά τρανζίστορ που λειτουργούν μεταξύ αποκοπής και ενεργού περιοχής. Είναι σχεδόν η ταχύτερη υπάρχουσα σήμερα οικογένεια, με συχνότητες λειτουργίας που προσεγγίζουν το ένα *Gigahertz*.

ε) Η οικογένεια λογικής ολοκληρωμένης έκχυσης, *IIL*:Χρησιμοποιείται κυρίως στα μέσης κλίμακας ολοκλήρωσης εξαρτήματα. Προσφέρει αρκετά ικανοποιητική ταχύτητα και πάρα πολύ μικρή κατανάλωση ισχύος. Χρησιμοποιεί τρανζίστορ πολλαπλών συλλεκτών.

ζ) Η οικογένεια λογικής τρανζίστορ επίδρασης πεδίου ημιαγωγού μεταλλικού οξειδίου, *MOSFETL* ή*MOSTL*:Χαρακτηριστικά της είναι η χαμηλή κατανάλωση ισχύος, η μεγάλη αντίσταση κατά του θορύβου και το πολύ μικρό μέγεθος κρυστάλλου που καταλαμβάνουν τα κυκλώματά της. Επίσης η σημερινή τεχνολογία της δίνει μεγάλες δυνατότητες ταχύτητας.

1.7.1 TTL (*Transistor – TransistorLogic*)

Είναι η πιο δημοφιλής οικογένεια ολοκληρωμένων. Λειτουργούν με μία τάση τροφοδοσίας (+5V). Το βασικό τους στοιχείο είναι πύλες **NAND**.



Σχήμα 1.7.1.

Το τρανζίστορ Q1 είναι ένα τρανζίστορ πολλαπλών εκπομπών, πράγμα που υλοποιείται με την τεχνολογία κατασκευής των ολοκληρωμένων κυκλωμάτων. Οι είσοδοι A, B και C υποτίθεται ότι τροφοδοτούνται από εξόδους όμοιων πυλών. Το ίδιο και η έξοδος Y τροφοδοτεί μια από τις εισόδους (A, B ...) μιας επόμενης όμοιας πύλης.

α) Αν μία τουλάχιστον από τις εισόδους A, B, ή C είναι χαμηλά (0, 2V=τάση κόρου εξόδου προηγούμενης βαθμίδας), τότε η αντίστοιχη δίοδος του εκπομπού του Q1 άγει και η τάση στη βάση του Q1 θα είναι Vp=(0,2+0,7)V=0,9V. (Το 0,7V είναι η πτώση τάσης στα άκρα της αγούσης διόδου).Η τάση όμως αυτή δεν είναι ικανή για να φέρει σε αγωγή τα τρανζίστορ Q2 και Q3, διότι απαιτείται τάση τουλάχιστον 0,7+0,7+0,7=2,1V (0,7V=Vγ διόδου B-C του Q1, ενώ οι υπόλοιπες δύο τάσεις των 0,7V είναι οι τάσεις για την έναρξη της λειτουργίας των Q2 και Q3). Άρα τα Q2 και Q3 βρίσκονται στην αποκοπή και Y=5V (*ψηλά*).

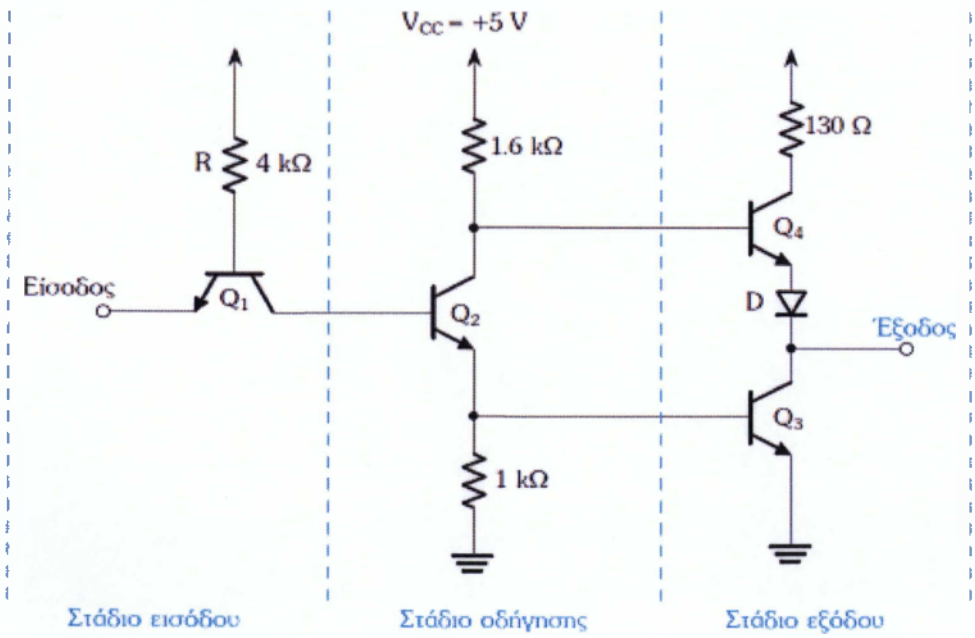
β) Αν όλες οι είσοδοι είναι ψηλά (5V), παύουν να λειτουργούν οι δίοδοι εκπομπού-βάσης του Q1, αλλά η βάση βρίσκεται σε δυναμικό περίπου 5V. Τότε λειτουργεί η δίοδος βάσης-συλλέκτη και ικανό ρέει προς τη βάση του Q2, το οποίο ωθείται στον κόρο. Το ίδιο συμβαίνει και για το Q3, η δε έξοδος αποκτά την τάση κόρου 0,2V-> Y=0,2V. Έτσι επαληθεύεται πίνακας αλήθειας της πύλης **NAND**.

Υπάρχουν αρκετές παραλλαγές των **TTL**. Οι βασικότερες από αυτές και τα κύρια χαρακτηριστικά τους φαίνονται στον ακόλουθο πίνακα:

Όνομα Παραλλαγής	Συντομογραφία	Propagation delay (nSec)	Power Dissipation (mW)/πύλη	Speed power Product (pJ)	Παράδειγμα συμβολισμού πύλης
STANDARD	TTL	10	10	100	7400
LOW-POWER	LTTL	33	1	33	74L00
HIGH-SPEED	HTTL	6	22	132	74H00
SCHOTTKY	STTL	3	19	57	74S00
LOW-POWER SCHOTTKY	LSTTL	11.5	2	19	74LS00

1.7.2 Το Πλήρες Κύκλωμα Πύλης TTL

Στο παρακάτω σχήμα απεικονίζεται το πλήρες κύκλωμα πύλης TTL. Αποτελείται από τρία στάδια: το τρανζίστορ εισόδου Q1, το στάδιο οδηγού Q2, που ρόλο έχει να παράγει τα δύο συμπληρωματικά σήματα τάσης τα οποία απαιτούνται για να οδηγήσουν το κύκλωμα τοτέμ, που είναι το τρίτο στάδιο (εξόδου) της πύλης. Το κύκλωμα τοτέμ στην πύλη TTL έχει δύο πρόσθετα συστατικά στοιχεία: την αντίσταση 130-Ω στο κύκλωμα συλλέκτη του Q4 και τη δίοδο D στο κύκλωμα εκπομπού του Q4. Το οδηγητικό στάδιο Q2 παράγει δύο συμπληρωματικά σήματα τα οποία είναι γνωστά και ως **διαχωριστής φάσης** (*phasesplitter*).



-Το πλήρες κύκλωμα πύλης TTL, όπου εικονίζεται μόνο ένας ακροδέκτης εισόδου.

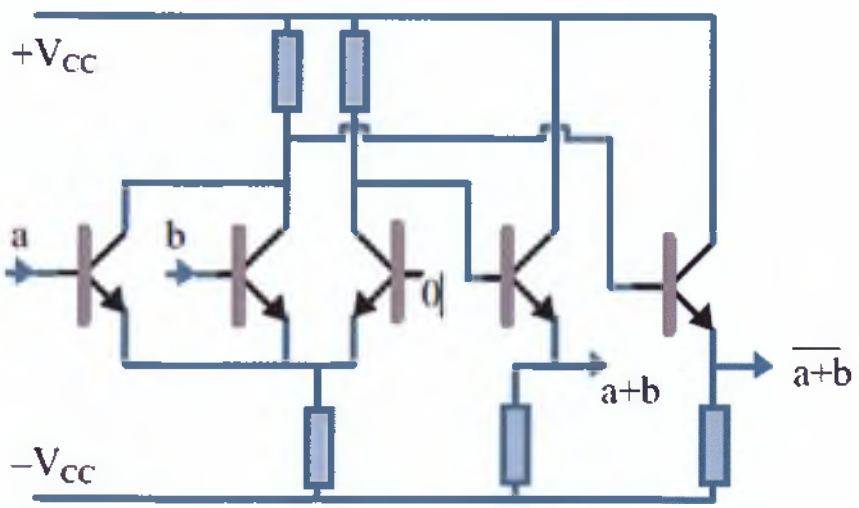
1.8 Οικογένειες ECL

Σήμερα υπάρχουν δύο δημοφιλείς μορφές ECL- συγκεκριμένα η ECL 10K και η ECL 100K. Η σειρά ECL 100K παρουσιάζει καθυστερήσεις πύλης τάξης μεγέθους 0.75 ns και καταναλώνει περίπου 40 mW/πύλη για ένα γινόμενο καθυστέρησης-ισχύος 30 pJ.Παρόλο που η κατανάλωση ισχύος είναι σχετικά μεγάλη, η σειρά 100K παρέχει τη μικρότερη δυνατή καθυστέρηση.

Η σειρά 10K είναι κάπως πιο αργή. Παρουσιάζει καθυστέρηση διάδοσης πύλης 2 ns και κατανάλωση ισχύος 25 mW για ένα γινόμενο καθυστέρησης-ισχύος 50 pJ.Παρόλο που η τιμή του DP είναι μεγαλύτερη από εκείνη της σειράς 100K, η σειρά 10K είναι ευκολότερη στη χρήση.

1.8.1 Σειρά ECL

Σειρά ECL (*Emitter Coupled Logic*). Χρησιμοποιεί διπολικά τρανσίστορ. Είναι τα ταχύτερα λογικά κυκλώματα που υπάρχουν. Επιτυγχάνουν χρόνους διαδόσεως της τάξεως των 2 ns, διότι χάρη στην ηλεκτρονική τους διάταξη τα τρανσίστορ δεν οδηγούνται στον κόρο. Το κύκλωμα μιας τυπικής πύλης φαίνεται στο παρακάτω σχήμα. Τα κυκλώματα των υπερυπολογιστών (π.χ. σειρά CRAY) είναι τεχνολογίας ECL.



ECL

-Ενιαίο κύκλωμα πυλών OR και NOR τεχνολογίας ECL

1.8.2 EMMITER - σε συνδυασμό LOGIC – ECL

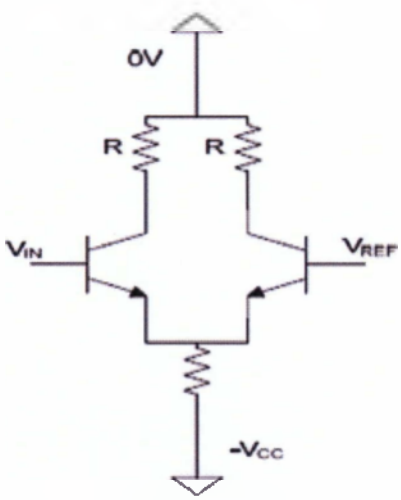
Οι οικογένεια ECL χρησιμοποιείται για συστήματα που απαιτούν λειτουργία σε υψηλές ταχύτητες. Τα *Ttransistors* στις πύλες ECL λειτουργούν σε κατάσταση όχι-κόρου (*Nonsaturation State*). Η συνθήκη αυτή επιτρέπει την ύπαρξη *Propagation Delay* από 1 ως 2 NS. Το βασικό
κύκλωμα των ECL είναι η πύλη NOR, πολλά όμως ολοκληρωμένα ECL παρέχουν έξοδο και για την πύλη OR.

Το σύμβολο της οικογένειας πυλών ECL είναι το ακόλουθο:

Το βασικό τους ηλεκτρονικό κύκλωμα είναι ένας διαφορικός ενισχυτής του τύπου που φαίνεται στο Σχ. 1.8.2. HVIN, συγκρίνεται με την τάση αναφοράς VREF και τα τρανζίστορ λειτουργούν πάντα στην ενεργό περιοχή (αύξηση ταχύτητας).

Σχήμα 1.8.2.

Απαιτούμενες τάσεις τροφοδοσίας
V_{CC}=-5.2V
V_{REF}=-1.5V
$V_{(0)}$=-1.7V
$V_{(1)}$=-0.9V



-Κύρια χαρακτηριστικά της E.C.L.:

1) Πολύ μεγάλη ταχύτητα Tpd 1/8 Ns
2) Λίγος παραγόμενος θόρυβος στο εσωτερικό της
3) Παίρνουμε την έξοδο καθώς και το συμπλήρωμά της (Y.Y)
4) Οι έξοδοι μπορούν να ενωθούν για πραγματοποίηση ενσύρματης λογικής.

5) Καλή αναισθησία θορύβου (*μεγάλη*), λόγω μικρών συνθετών αντιστάσεων.
6) Μεγάλη κατανάλωση ισχύος ~ 80 mW/πύλη.
7) Πολλές και διάφορες στάθμες τάσεων τροφοδοσίας, πράγμα που τις κάνει δύσχρηστες και δύσκολες στην προσαρμογή τους με άλλες κατηγορίες ηλεκτρονικών πυλών.

-Γενικά η χρήση τους είναι ανεπιθύμητη εκεί που δεν απαιτείται υψηλή ταχύτητα. Βρίσκουν ευρεία εφαρμογή στους ηλεκτρονικούς υπολογιστές.

1.9 Fan-Out

Όταν το σήμα εισόδου σε μια πύλη ECL αντιστοιχεί στο λογικό μηδέν, το ρεύμα εισόδου είναι ίσο με το ρεύμα που περνάει στην αντίσταση pull-down των 50 kΩ.
Έτσι

$$I_{IL} = \frac{-1.77+5.2}{50} \simeq 69\mu A$$

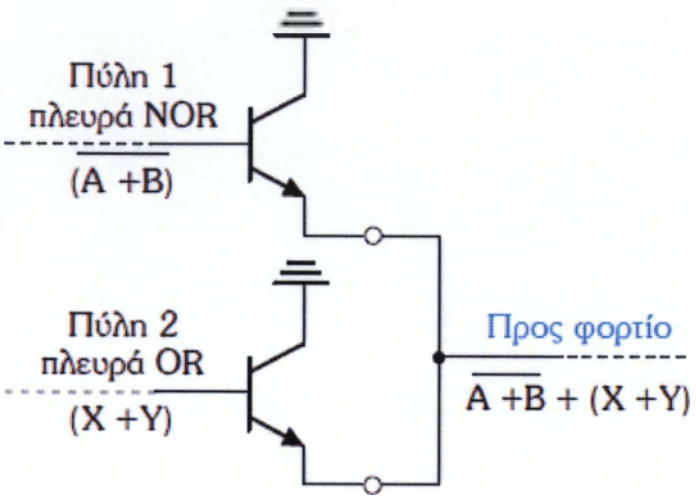
Όταν η είσοδος αντιστοιχεί στη λογική μονάδα, το ρεύμα εισόδου είναι μεγαλύτερο εξαιτίας του ρεύματος βάσης του τρανζίστορ εισόδου. Έτσι, υποθέτοντας το β του τρανζίστορ ίσο με 100, παίρνουμε

$$I_{IH} = \frac{-0.88+5.2}{50} + \frac{4}{101} \simeq 126\mu A$$

Και τα δύο αυτά ρεύματα έχουν μικρές τιμές, οι οποίες σε συνδυασμό με το γεγονός ότι η αντίσταση εξόδου της πύλης ECL είναι πολύ μικρή, εξασφαλίζουν ελάχιστη μείωση του επιπέδου των λογικών σημάτων εξαιτίας των ρευμάτων εισόδου των πυλών fan-out.

1.10 Η Δυνατότητα Καλωδιωμένου OR (*Wired-OR*)

Το στάδιο εξόδου που ακολούθου εκπομπού της οικογένειας των πυλών ECL επιτρέπει την πραγματοποίηση ενός πρόσθετου λογικού επιπέδου με πολύ μικρό κόστος με την απλή παράλληλη διασύνδεση των εξόδων πολλών πυλών μαζί. Αυτό εικονίζεται στο παρακάτω σχήμα όπου οι έξοδοι δύο πυλών έχουν καλωδιωθεί μαζί. Σημειώνεται ότι οι δίοδοι βάσης-εκπομπού των ακολούθων της εξόδου παρέχουν τη λειτουργία μιας σαφούς λογικής συνάρτησης OR. Αυτή η σύνδεση καλωδιωμένου OR μπορεί να χρησιμοποιηθεί για να μας δώσει πύλες που έχουν υψηλό fan-in καθώς και να αυξήσει την ευελιξία χρήσης της ECL στη λογική σχεδίαση.



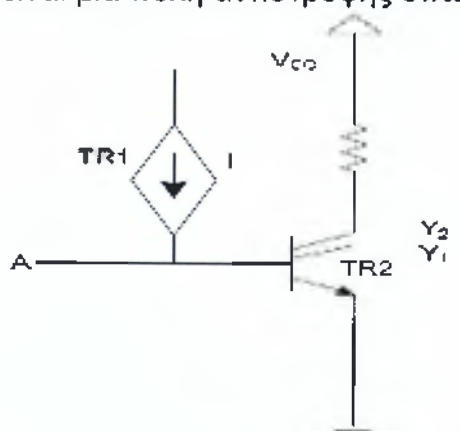
Σχήμα 1.10: Η δυνατότητα καλωδιωμένου OR των ECL.

Παρατήρηση

Η λογική οικογένεια ECL έχει εφαρμοσθεί με επιτυχία στο σχεδιασμό ψηφιακών τηλεπικοινωνιακών συστημάτων υψηλής ταχύτητας και στο σχεδιασμό υπολογιστικών συστημάτων. Εφαρμόζεται επίσης στο σχεδιασμό κυκλωμάτων LSI και VLSI.

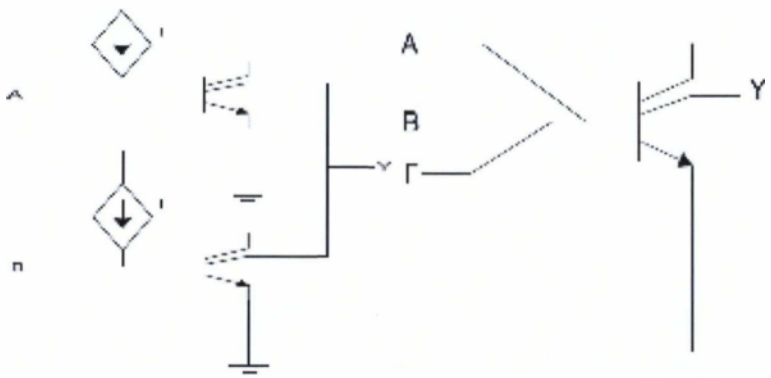
1.11 Κυκλώματα I2L ή IIL (*IntergratedInjectionLogic ή CurrentInjectionLogic*)

Το ολοκληρωμένο κύκλωμα των πυλών αυτών, περιλαμβάνει μία μονάδα παροχής σταθερού ρεύματος I, (*τρανζίστορ TR1*), και το τρανζίστορ TR2 που είναι μια πύλη αντιστροφής όπως φαίνεται στο πιο κάτω σχήμα.



Σχήμα 1.11.(α): Οικογένεια πύλης **I2L**. Πραγματοποίηση λογικού NOT.

Λειτουργία: Για χαμηλή στάθμη εισόδου, το ρεύμα I ακολουθεί τον αριστερό κλάδο προς την είσοδο Α, το τρανζίστορ TR2 αποκόπτεται (OFF) και οι έξοδοι είναι σε υψηλή στάθμη. Εάν η στάθμη εισόδου είναι ψηλά, τότε το ρεύμα I θέτει σε λειτουργία το τρανζίστορ TR2, που πηγαίνει στον κόρο, ενώ οι έξοδοί του γίνονται Y1=Y2=0,2V (*χαμηλά*).Έτσι το κύκλωμα λειτουργεί σαν πύλη αντιστροφής NOT. Με τη λογική απευθείας σύνδεσης επιτυγχάνεται και η παραγωγή και των άλλων λογικών συναρτήσεων. Στα σχήματα 1.11.(β) και 1.11.(γ) που ακολουθούν, φαίνονται τα κυκλώματα I2L που πραγματοποιούν τις λογικές πράξεις NOR και NAND αντίστοιχα.



Σχήμα 1.11.(β): $Y=\overline{A+B}$ **Σχήμα 1.11.(γ):** $Y=\overline{AB\Gamma}$

1.12 METAL – OXIDE SEMICONDUCTOR – MOS και MOS

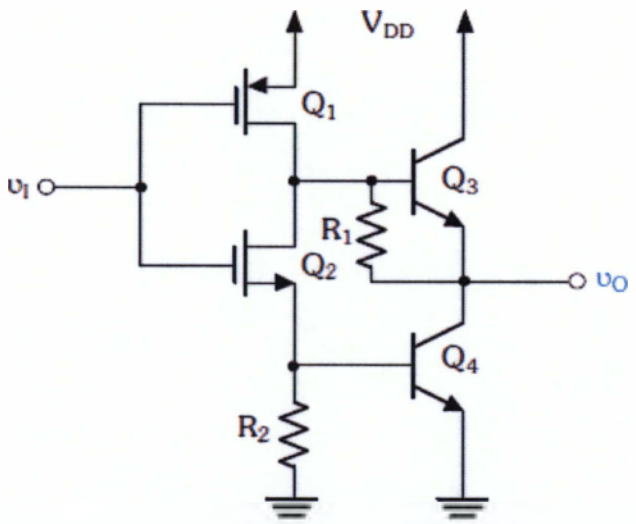
Το MOS είναι ένα μονοπολικό TRANSISTOR στο οποίο λαμβάνει χώρα, ροή ενός μόνο είδους ηλεκτρικού φορτίου που μπορεί να είναι είτε ηλεκτρόνια (N-CHANNEL) είτε οπές (PCHANNEL). Ένα P-CHANNEL MOS, απαιτεί αρνητική τάση για τη λειτουργία του και αναφέρεται ως PMOS, ενώ ένα N-CHANNEL MOS, απαιτεί θετική τάση για τη λειτουργία του και αποκαλείται NMOS. Τα CMOS (COMPLEMENTARY MOS) χρησιμοποιούν ένα PMOS και ένα NMOS και απαιτούν θετική τάση για τη λειτουργία τους. Τα τρία βασικά πλεονεκτήματα των MOS έναντι του διπολικού TRASISTOR είναι τα ακόλουθα:

- Υψηλή πυκνότητα ολοκλήρωσης, που επιτρέπει την ύπαρξη, σε δεδομένο CHIP, περισσοτέρων κυκλωμάτων.
- Απλούστερες και οικονομικότερες τεχνικές κατασκευής.
- Μικρή κατανάλωση ισχύος.

1.13 Ψηφιακά Κυκλώματα BiCMOS

Η τεχνολογία BiCMOS συνδυάζει διπολικά και CMOS κυκλώματα στο ίδιο τσίπ ολοκληρωμένου κυκλώματος. Τα κυκλώματα αυτά διατηρούν τη χαμηλή ισχύ, την υψηλή σύνθετη αντίσταση εισόδου και τα μεγάλα περιθώρια θορύβου των CMOS, αλλά και την ικανότητα οδήγησης μεγάλων ρευμάτων καθώς και τη μεγάλη ταχύτητα λειτουργίας των διπολικών τρανζίστορ. Το αποτέλεσμα είναι μία τεχνολογία κυκλωμάτων που είναι ικανή να υλοποιήσει πολύ συχνά, γρήγορα και χαμηλής ισχύος ψηφιακά ολοκληρωμένα κυκλώματα. Επιπλέον, επειδή η τεχνολογία BiCMOS είναι κατάλληλη για την υλοποίηση αναλογικών κυκλωμάτων μεγάλων απαιτήσεων, καθίσταται εφικτή η υλοποίηση αναλογικών και ψηφιακών λειτουργιών πάνω στο ίδιο τσίπ, κάνοντας πια το "ένα σύστημα πάνω σ΄ένατσίπ" εφικτό στόχο.

Ο βασικός λογικός αντιστροφέας BiCMOS εικονίζεται στο παρακάτω σχήμα:



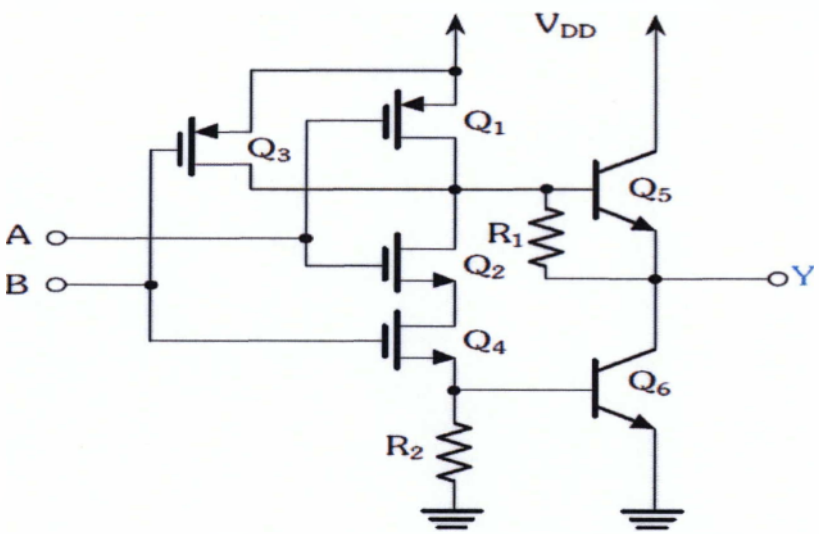
Ο βασικός λογικός αντιστροφέας BiCMOS, αποτελείται από έναν αντιστροφέαCMOS, όπου σχηματίζεται από τα τρανζίστορ Q1 και Q2 και ένα διπολικό στάδιο εξόδου, που σχηματίζεται από τα τρανζίστορ Q3 και Q4 μαζί με τις αντιστάσεις R1 και R2. Για να δούμε πως λειτουργεί ο λογικός αντιστροφέαςBiCMOS θεωρούμε πρώτα την περίπτωση όπου η τάση εισόδου u1 είναι στο χαμηλό επίπεδο στα 0V. Το τρανζίστορ Q2 θα είναι αποκομμένο και δε θα περνάει ρεύμα στη βάση του Q4. Η αντίσταση R2 μηδενίζει την τάση στη βάση του Q4 οπότε και το Q4 θα είναι αποκομμένο. Το τρανζίστορ PMOSQ1 θα άγει, δημιουργώντας ένα αγώγιμο κανάλι χαμηλής σύνθεσης αντίστασης μεταξύ της βάσης του Q3 και της V_{DD}. Αν οι ακροδέκτες εξόδου της πύλης είναι ανοικτοκυκλωμένος, δε θα περνάει ρεύμα από το Q3 ή την R1 και συνεπώς δε θα περνάει ρεύμα από το Q1. Άρα η πτώση τάσης στο Q1 θα είναι περίπου μηδέν και η αντίσταση R1 θα κάνει την τάση στη έξοδο της πύλης ίση με V_{DD}. Έτσι η αντίσταση R1 θα λειτουργεί ως αντίσταση pull-up με αποτέλεσμα V_{OH}- V_{DD}.

Βλέπουμε λοιπόν πως ο αντιστροφέαςBiCMOS παρουσιάζει τη μεγάλη διακύμανση τάσης εξόδου των CMOS ,την ικανότητα οδήγησης μεγάλων ρευμάτων και τις αντίστοιχα μικρές καθυστερήσεις μετάδοσης που είναι χαρακτηριστικές των BJT. Το κύκλωμα BiCMOS παρουσιάζει μεγαλύτερη ταχύτητα λειτουργίας από αυτήν που μπορεί να πετύχει κανείς με CMOS και χαμηλότερη κατανάλωση ισχύος από αυτήν που μπορούμε να πετύχουμε στα διπολικά κυκλώματα.

Με **fan-out** μηδενικό η καθυστέρηση μιας πύλης **BiCMOS** είναι τυπικά 1 ns, κάπως μεγαλύτερη από εκείνη της **CMOS**. Το πλεονέκτημα στην ταχύτητα των **BiCMOS** γίνεται φανερό όταν η πύλη χρειάζεται να οδηγήσει άλλες. Για παράδειγμα, για fan-out 10 η χρονική καθυστέρηση μιας πύλης **BiCMOS** είναι ακόμα κοντά στο 1 ns ενώ εκείνη της πύλης **CMOS** είναι διπλάσια.

1.13.1 Λογική Λειτουργία του Κυκλώματος BiCMOS

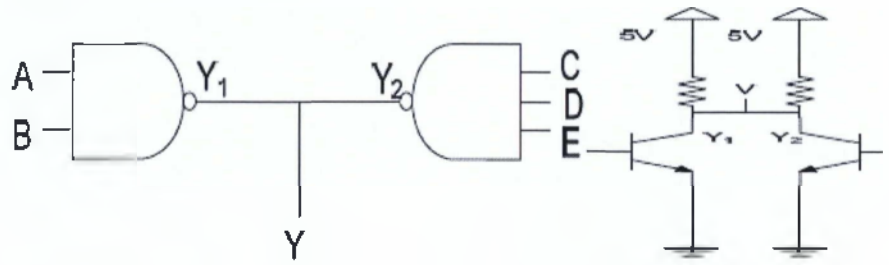
Η λογική λειτουργία του κυκλώματος αντιστροφέαBiCMOSπραγματοποιείται από τον αντιστραφέαCMOS που σχηματίζεται από τα Q1 και Q2.



-Πύλη NANDBiCMOS.Η έξοδος Y είναι χαμηλά μόνον όταν τα A και B είναι ταυτόχρονα ψηλά: $\overline{Y} = AB$, που περιγράφεται και ως $Y = \overline{AB}$.

-Η τεχνολογία BiCMOS εφαρμόζεται σήμερα σε μια μεγάλη κλίμακα προϊόντων που περιλαμβάνει μικροεπεξεργαστές, στατικές RAM και gatearrays.

1.14 Λογική Απευθείας Σύνδεσης (*Wiredlogic*)



Σχήμα 1.14.(α). **Σχήμα 1.14.(β).**

-Συνδέοντας μαζί τις εξόδους δύο (ή και περισσότερων) πυλών NAND, θα έχουμε την εξής λειτουργία. Αν μία από τις εξόδους των πυλών είναι χαμηλά, η Y επίσης πάει χαμηλά. Έτσι πραγματοποιείται μία πράξη "AND" μεταξύ των αρχικών εξόδων. Η τελική έξοδος Y, σε σχέση με τις εισόδους θα είναι:

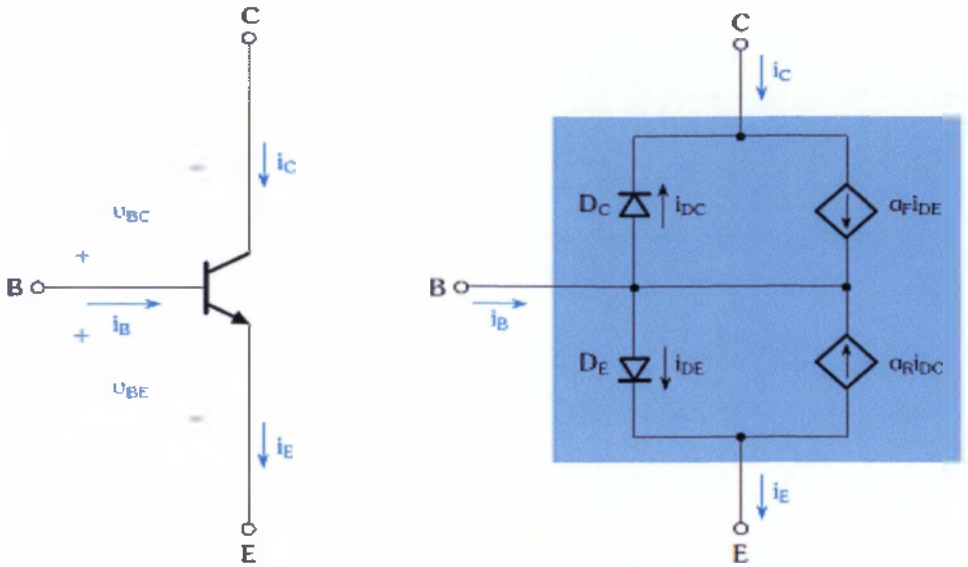
$$Y = Y_1 \bullet Y_2 = \overline{(A \bullet B)}\,\overline{(C \bullet D \bullet E)} = \overline{AB + CDE}$$

Έτσι πραγματοποιείται η Λογική Απευθείας Σύνδεσης και η λογική πράξη λέγεται AND-OR-INVERT (AOI).

1.15 Το διπολικό τρανζίστορ ως ψηφιακό κυκλωματικό στοιχείο-Λογική Κόρου-Λογική Μη Κόρου

Η πιο συνηθισμένη χρήση του διπολικού τρανζίστορ (BJT) στα ψηφιακά κυκλώματα είναι η χρησιμοποίηση των δύο ακραίων τρόπων λειτουργίας του: της αποκοπής και του κόρου. Τα κυκλώματα που προκύπτουν είναι γνωστά ως κυκλώματα **Λογικής Κόρου**. Ως πλεονέκτημα αυτού του τρόπου εφαρμογής, μπορούμε να αναφέρουμε τα σχετικά μεγάλα και ξεκάθαρα ¨πηδήματα¨ λογικού επιπέδου και τη σχετική χαμηλή κατανάλωση ισχύος. Το μεγαλύτερο μειονέκτημα είναι η σχετικά αργή απόκριση εξαιτίας των μεγάλων χρόνων αποκοπής των κορεσμένων τρανζίστορ.Για να πάρουμε γρηγορότερη λογική, το κύκλωμα πρέπει να σχεδιαστεί με τέτοιο τρόπο ώστε το BJT να μη φθάσει στον κόρο.

Το Διπολικό Τρανζίστορ Ενώσεως (*BJT*) αποτελείται από δύο επαφές pn, την επαφή εκπομπού-βάσης και την επαφή συλλέκτη-βάσης. Μπορούμε λοιπόν να εκφράσουμε τα ρεύματα στους ακροδέκτες του BJT ως υπέρθεση των ρευμάτων που οφείλονται στις δύο επαφές pn, όπως φαίνεται στο παρακάτω σχήμα:



-Τρανζίστορ npn και το αντίστοιχο μοντέλο **Ebers-Moll (EM)**

Στο πιο πάνω σχήμα, εικονίζεται ένα τρανζίστορ npn μαζί με το EM μοντέλο Του. Το μοντέλο αποτελείται από δύο διόδους και δύο ελεγχόμενες πηγές. Οι δίοδοι είναι: η DE, η δίοδος εκπομπού-βάσης και η DC, η δίοδος συλλέκτη-βάσης. Τα ρεύματα των διόδων I_{DE} και I_{DC} δίνονται από την εξίσωση της διόδου:

$$i_{DE} = I_{SE}\left(e^{v_{BE}/V_T}-1\right)$$

1.16 Μοντέλο EM (*Ebers-Moll*)

Το μοντέλο EM είναι ένα μοντέλο χαμηλών συχνοτήτων (στατικό). Βασίζεται στο γεγονός ότι το διπολικό τρανζίστορ επαφές (*BJT-BipolarJunctionTransistor*) αποτελείται από δύο επαφές pn, την επαφή πομπού-βάσης και την επαφή συλλέκτη-βάσης.

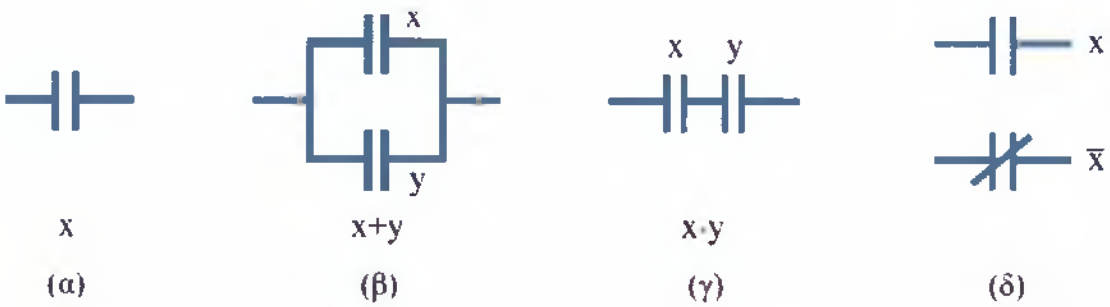
1.17 Κυκλώματα με Διακόπτες

Κυκλώματα με διακόπτες χρησιμοποιούνται σήμερα είτε σε απλές εφαρμογές είτε στα πολύ σύνθετα κυκλώματα **VLSI**, π.χ. μικροϋπολογιστές. Είναι ενδιαφέρον να μάθουμε να τα χρησιμοποιούμε, κυρίως για απλές περιπτώσεις, π.χ. για μικρές εφαρμογές στο σπίτι.

Κυκλώματα με διακόπτες είναι τα ψηφιακά κυκλώματα Πολύ Μεγάλης Κλίμακας Ολοκλήρωσης (**VLSI**), που κατασκευάζονται σήμερα και χρησιμοποιούν σαν διακόπτες τρανσίστορ τύπου **MOS**. Τα τρανζίστορ **MOS** αποτελούν ιδανικούς διακόπτες για τα **VLSI** λόγω του πολύ μικρού μεγέθους τους (διαστάσεις μερικά μm), της ευκολίας κατασκευής τους και της αρκετά μεγάλης ταχύτητας λειτουργίας τους.

1.17.1 Βασικά Κυκλώματα Διακοπτών και Συμβολισμοί

(α) διακόπτης (β) παράλληλη (γ) σε σειρά σύνδεση (δ) αντιστροφή



(α) (β) (γ) (δ)

-Ορίζεται σαν συνάρτηση μεταφοράς FAB (x1,…,xn) μεταξύ δύο σημείων A και B ενός κυκλώματος μια δίτιμη συνάρτηση που παίρνει την τιμή "1", όταν υπάρχει δίοδος μεταξύ των A και B, και την τιμή "0", όταν υπάρχει διακοπή. Συνοπτικά, η σύνθεση δύο συναρτήσεων μεταφοράς F1 και F2 δίνεται από τον παρακάτω πίνακα:

F_1 F_2	Παράλληλη OR F_1+F_2	Σειρά AND $F_1.F_2$	Αντιστρ. NOT F_1'
0 0	0	0	1
0 1	1	0	1
1 0	1	0	0
1 1	1	1	0

-Πίνακας Αληθείας Συναρτήσεων Μεταφοράς

1.18 Πολυπλοκότητα Πυλών

Συνηθίζεται να κατατάσσονται τα IC στις τέσσερις ακόλουθες κατηγορίες, ανάλογα με την πυκνότητα πυλών που υπάρχουν σε ένα CHIP:

1. **SSI** (*SMLL SCALE INTEGRATION*): λιγότερεςαπό 10 πύλεςστο CHIP

2. **MSI** (*MEDIUM SCALE INTEGRATION*): από 10 έως 100 πύλες.

3. **LSI** (*LARGE SCALE INTEGRATION*): πάνωαπό 100 πύλες.

4. **VLSI** (*VERY LARGE SCALE INTEGRATION*): πάνωαπό 1000 πύλες.

1.19 Κλίμακες Ολοκλήρωσης

Τα ολοκληρωμένα κυκλώματα κατατάσσονται σε κατηγορίες ανάλογα με την πολυπλοκότητα τους. Υπάρχουν κυκλώματα:

α. μικρής κλίμακας ολοκλήρωσης (*SmallScaleIntegration* ή **SSI**), που περιέχουν μέχρι 10 πύλες ανά chip,

β. μέσης κλίμακας ολοκλήρωσης (*MediumScaleIntegration* ή **MSI**), που περιέχουν μεταξύ 10 και 100 πύλες ανά chip και συνήθως υλοποιούν πολύπλοκες λογικές συναρτήσεις, π.χ. κωδικοποιητές, αριθμητικές μονάδες,

γ. μεγάλης κλίμακας ολοκλήρωσης (*LargeScaleIntegration* ή **LSI**) από 100 έως 5000 πύλες και

δ. πολύ μεγάλης κλίμακας ολοκλήρωσης (*VeryLargeScaleIntegration* ή **VLSI**) με περισσότερες από 5000 πύλες και υλοποιούν συνήθως πλήρη ψηφιακά συστήματα, π.χ. υπολογιστές ή κυκλώματα μνημών.

Ανάλογα με τον τρόπο κατασκευής τους τα ολοκληρωμένα κυκλώματα. ταξινομούνται σε οικογένειες. Κάθε οικογένεια έχει τα δικά της, "κοινά", χαρακτηριστικά, όπως είδος τρανσίστορ, τάση τροφοδοσίας, ταχύτητα λειτουργίας κ.λπ. Κύριο χαρακτηριστικό μιας οικογένειας είναι η ηλεκτρική συμβατότης των κυκλωμάτων της, ώστε οποιοδήποτε κύκλωμα της οικογένειας να μπορεί να συνδεθεί με οποιοδήποτε άλλο της οικογένειας χωρίς δυσκολία. Αυτό επιτυγχάνεται με την τυποποίηση (α) της τάσεως τροφοδοσίας και (β) των ηλεκτρικών χαρακτηριστικών των σημάτων που εκπροσωπούν τις λογικές καταστάσεις.

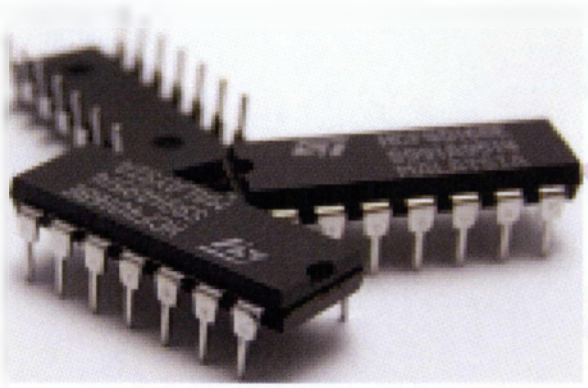
1.20 Συσκευασία Ολοκληρωμένων

Οι δύο βασικές συσκευασίες των IC, η FLAT και η DIP (*DUAL IN LINEPACKAGE*) φαίνονται στα σχήματα 1 και 2 αντίστοιχα:

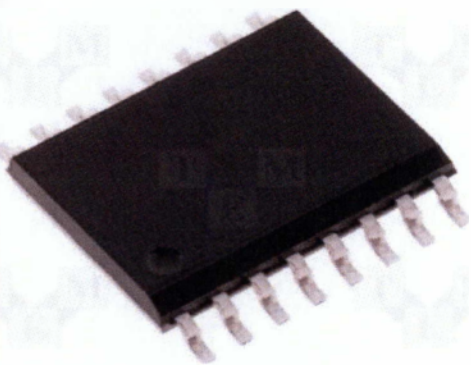
FLAT PACKAGE
Σχήμα 1.20.(α)

DUAL-IN-LINE PACKAGE
Σχήμα 1.20.(β)

- Μικρότερα σε διαστάσεις ολοκληρωμένα κυκλώματα με τους ακροδέκτες στην άνω επιφάνεια (περίπου σαν τα "flatpackage" αλλά πιο μικρά) είναι τα chip με την τεχνολογία SMD (*SurfaceMountingDevices*).



Dual-In-Line package



Flat package

Κεφάλαιο 2°

2.1 ΕΙΣΑΓΩΓΗ

Το testing ενός συστήματος είναι ένα πείραμα κατά το οποίο το σύστημα εξετάζεται και τα αποτελέσματά του αναλύονται για να επιβεβαιωθεί ότι συμπεριφέρεται σωστά. Αν διαπιστωθεί μη αποδεκτή συμπεριφορά τότε ένας δεύτερος στόχος του testing είναι να διαγνώσει ή να εντοπίσει την αιτία της μη αποδεκτής συμπεριφοράς. Η διάγνωση απαιτεί γνώση της εσωτερικής δομής του εξεταζόμενου συστήματος. Γενικά οι είσοδος και η απόκριση του συστήματος στο testing αντιστοιχούν στον τύπο της πληροφορίας που επεξεργάζεται το σύστημα. Επομένως το testing καλύπτει ένα ευρύ φάσμα από δραστηριότητες και περιβάλλοντα όπως ένα ή περισσότερα υποσυστήματα που αλληλοελέγχονται, ένας επεξεργαστής που ελέγχει τον εαυτό του, εξοπλισμός αυτομάτου ελέγχου (ATE – *AutomaticTestEquipement*) που δημιουργούν διανύσματα ελέγχου και παρακολουθούν τις αποκρίσεις. Η πολυπλοκότητα ενός κυκλώματος σχετίζεται με το επίπεδο αφαίρεσης που χρειάζεται για να περιγραφεί η λειτουργία του με έναν κατανοητό τρόπο.

Διακρίνουμε το εξής επίπεδα:

- Το **λογικό επίπεδο** (*logiclevel*), όπου η πληροφορία αναπαριστάνεται με διακριτές λογικές τιμές 0 και 1. Ένα πιο ακριβές μοντέλο αναπαράστασης είναι με περισσότερες από δύο λογικές τιμές. Ένας ακόμη διαχωρισμός είναι σε συνδυαστικά και ακολουθιακά κυκλώματα. Στην πρώτη κατηγορία οι τιμές εξόδου του κυκλώματος εξαρτώνται μόνο από τις τρέχουσες λογικές τιμές των εισόδων του. Στην δεύτερη κατηγορία το κύκλωμα μπορεί να «θυμάται» παλαιότερες καταστάσεις και έτσι επεξεργάζεται ακολουθίες από λογικές τιμές.

- Το **επίπεδο καταχωρητών** (*registerlevel*), όπου θεωρούμε ότι το κύκλωμα αποτελεί πλέον σύστημα αφού η λειτουργία του όσο αφορά την επεξεργασία των λογικών τιμών δεν έχει νόημα ή είναι αδύνατη. Έτσι θεωρούμε ότι το σύστημα αποτελείται από ένα τμήμα δεδομένων (datapart) το οποίο αλληλεπιδρά με ένα τμήμα ελέγχου (controlpart). Αν και η λειτουργία του συστήματος καθορίζεται ακόμη από λογικές τιμές, η επεξεργασία της πληροφορίας γίνεται με λέξεις (words). Με τον όρο λέξη εννοούμε ένα σύνολο από λογικές τιμές που λέγεται διάνυσμα (vector). Επειδή οι λέξεις αποθηκεύονται σε καταχωρητές το επίπεδο αυτό ονομάστηκε επίπεδο καταχωρητών.

- Το **επίπεδο εντολών** (*instructionsetlevel*), όπου η πληροφορία εξακολουθεί να είναι οργανωμένη σε λέξεις που λέγονται εντολές (instructions). Το σύστημα που περιγράφεται από ένα σύνολο εντολών λέγεται επεξεργαστής συνόλου εντολών (instructionsetprocessor).

- Το **επίπεδο επεξεργαστή** (*processorlevel*), όπου θεωρούμε ότι το σύστημα επεξεργάζεται ακολουθίες από εντολές που ονομάζονται προγράμματα (programs) και ενεργούν πάνω σε τμήματα δεδομένων που ονομάζονται δομές δεδομένων (datastructures).

- Το **επίπεδο συστήματος** (*systemlevel*), όπου θεωρούμε ότι το σύστημα αποτελείται από ένα σύνολο ανεξάρτητων υποσυστημάτων ή μονάδων, τα οποία επικοινωνούν με τμήματα λέξεων που ονομάζονται μηνύματα (*messages*).

2.2 Λάθη και Αστοχίες

Με τον όρο λάθη, (*errors*) αναφερόμαστε σε μια μη σωστή λειτουργία του συστήματος που εξετάζουμε (UUT – *UnitUnderTest*). Οι λόγοι στους οποίους οφείλονται τα λάθη είναι:

- **Λάθη σχεδίασης** (*designerrors*): αυτά μπορεί να προέρχονται από ανεπαρκή χαρακτηριστικά, παραβίαση των κανόνων σχεδίασης (*designrules*), μη σωστή αντιστοίχηση μεταξύ των διαφόρων επιπέδων σχεδίασης.

- **Λάθη κατασκευής** (*fabricationerrors*): αυτά οφείλονται σε λάθος εξαρτήματα, λάθος καλωδίωση (wiring), βραχυκυκλώματα λόγω μη σωστών κολλήσεων.

- **Λάθη από κατασκευαστικά ελαττώματα** (*fabricationdefects*): αυτά οφείλονται στον ανθρώπινο παράγοντα.

- **Λάθη από φυσικές αποτυχίες** (*physicalfailures*): οφείλονται κυρίως στην κόπωση των υλικών και σε περιβαλλοντικούς παράγοντες.

Τα λάθη κατασκευής (*fabricationerrors*), τα λάθη από κατασκευαστικά ελαττώματα (*fabricationdefects*) και τα λάθη από φυσικές αποτυχίες (*physicalfailures*) ονομάζονται και **φυσικά σφάλματα** (*physicalfaults*). Ανάλογα με την σταθερότητά τους στον χρόνο τα φυσικά σφάλματα διακρίνονται σε **μόνιμα** (*permanent*) τα οποία από την στιγμή που εμφανίζονται παραμένουν συνεχώς, σε **διακοπτόμενα** (*intermittent*) τα οποία εμφανίζονται κατά διαστήματα, και σε **παροδικά** (*transient*) τα οποία παρουσιάζονται μια φορά και οφείλονται σε μια προσωρινή μεταβολή σε περιβαλλοντικούς παράγοντες. Γενικά ένα σφάλμα ανιχνεύεται από ένα λάθος που προκαλείται από αυτό.

2.3 Μοντέλα Σφαλμάτων

Τα μοντέλα σφαλμάτων τα οποία χρησιμοποιούνται συνήθως κατά τον έλεγχο ενός κυκλώματος φαίνονται στο παρακάτω σχήμα:

Μοντέλο Σφαλμάτων	Περιγραφή
Απλά σφάλματα μόνιμης τιμής (Single stuck-at faults)	Μία μόνο γραμμή του κυκλώματος βρίσκεται μόνιμα κολλημένη στο λογικό 0 ή στο λογικό 1
Πολλαπλά σφάλματα μόνιμης τιμής (Multiple stuck-at faults)	Δύο ή περισσότερες γραμμές του κυκλώματος έχουν μόνιμες τιμές (όχι απαραίτητα τις ίδιες)
Σφάλματα γεφύρωσης (Bridging faults)	Δύο ή περισσότερες γραμμές ανεξάρτητες μεταξύ τους, είναι μόνιμα συνδεδεμένες
Σφάλματα ανοιχτών γραμμών σε τρανζίστορ (Stuck-open faults)	Ελαττώματα σε κάποια από τα τρανζίστορ μιας πύλης CMOS, τα οποία την κάνουν να συμπεριφέρεται σαν στοιχείο μνήμης
Σφάλματα μόνιμα αγόντων τρανζίστορ (Stuck-on faults)	Ένα τρανζίστορ άγει πάντα
Σφάλματα καθυστέρησης (Delay faults)	Σφάλματα που προκαλούνται από καθυστερήσεις σε ένα ή περισσότερα μονοπάτια του κυκλώματος

Τα πιο ευρέως χρησιμοποιούμενα μοντέλα σφαλμάτων

2.3.1 Σφάλματααπόβραχυκύκλωμα (*Bridging or Short-circuit faults*)

Τα σφάλματα αυτά προκύπτουν από βραχυκύκλωμα μεταξύ δύο (ή περισσοτέρων) γραμμών οι οποίες υπό κανονικές συνθήκες δε συνδέονται μεταξύ τους.

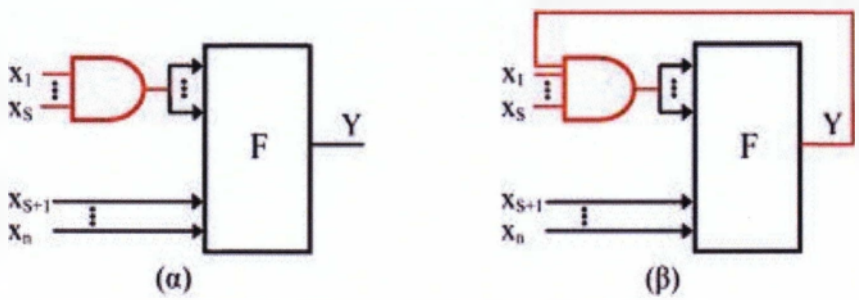
Ένα βραχυκύκλωμα μεταξύ οποιωνδήποτε s γραμμών από τις n ενός κυκλώματος θα προκαλέσει (n/s) απλά σφάλματα από βραχυκύκλωμα και ο αριθμός των πολλαπλών σφαλμάτων από βραχυκύκλωμα θα είναι πολύ μεγαλύτερες.

Τα σφάλματα από βραχυκύκλωμα μπορούν να ταξινομηθούν σε:

- Βραχυκύκλωμα εισόδου (*input bridging*)
- Βραχυκύκλωμα ανάδρασης (*feedback bridging*)
- Βραχυκύκλωμα χωρίς ανάδραση (*non-feedbackbridging*)

Η παρουσία σε κύκλωμα ενός σφάλματος από βραχυκύκλωμα ανάδρασης μπορεί να προκαλέσει τη ταλάντωση του κυκλώματος ή να το μετατρέψει σε ακολουθιακό.

Τα σχήματα που ακολουθούν δείχνουν τα λογικά μοντέλα περιγραφής για το βραχυκύκλωμα εισόδου και το βραχυκύκλωμα ανάδρασης αντίστοιχα:



(α). Λογικό μοντέλο για βραχυκύκλωμα s εισόδων ($X_1, X_2,...,X_S$)
(β). Λογικό μοντέλο για βραχυκύκλωμα ανάδρασης ($YX_1X_2X_S$)

2.3.2 Σφάλματα καθυστερήσεων (*Delayfaults*)

Τα σφάλματα αυτά δε μεταβάλουν την λογική συνάρτηση που υλοποιεί το κύκλωμα αλλά επηρεάζουν τη χρονική συμπεριφορά του κυκλώματος, με αποτέλεσμα αυτή να αποκλίνει από τις σχεδιαστικές προδιαγραφές. Μια βλάβη για παράδειγμα μπορεί να προκαλέσει καθυστέρηση στη μετάπτωση ενός σήματος από τη λογική στάθμη "0" στη λογική στάθμη "1". Το αποτέλεσμα της ύπαρξης αυτής της βλάβης μπορεί να μοντελοποιηθεί από ένα σφάλμα καθυστέρησης.

Δύο τύποι σφαλμάτων καθυστέρησης έχουν προταθεί:

- Τα σφάλματα καθυστέρησης πύλης (*gatedelayfaults*) που μοντελοποιούν βλάβες που μεταβάλουν τη χρονική καθυστέρηση διάδοσης του σήματος από την πύλη (gatepropagationdelay). Το μειονέκτημα αυτού του μοντέλου είναι ότι δεν μπορεί να περιγράψει διαδιδόμενες βλάβες.

- Σφάλματα καθυστέρησης μονοπατιού (*pathdelayfault*) που μοντελοποιούν βλάβες που έχουν ως αποτέλεσμα η χρονική καθυστέρηση των σημάτων εισόδου κατά μήκος ενός μονοπατιού κυκλώματος να υπερβαίνει τις προδιαγραφές.

2.4 Αξιοπιστία και Ανάγκη για Έλεγχο

Με την συνεχόμενη κλιμάκωση της CMOS τεχνολογίας (*Complementary Metal Oxide Semiconductor*), οι διαστάσεις των ψηφιακών κυκλωμάτων ολοένα και μειώνονται, ενώ η πολυπλοκότητα και η ταχύτητά τους αυξάνεται. Αποτέλεσμα αυτών των αλλαγών είναι η εμφάνιση νέων τύπων ελαττωμάτων στα ψηφιακά κυκλώματα. Αυτός ο λόγος οδήγησε στο να γίνει πλέον βασική επιδίωξη η αξιοπιστία των ηλεκτρονικών συσκευών. Σημαντική προϋπόθεση για την ύπαρξη αξιόπιστων ηλεκτρονικών συστημάτων, είναι η ύπαρξη τεχνικών που να επιβεβαιώνουν την ύπαρξη ή μη ελαττωμάτων. Επειδή τα συστήματα αποτελούνται από υλικό (*hardware*) και λογισμικό (*software*), θα μπορούσε ένα ελάττωμα να προκύψει από οποιαδήποτε από τις δύο κατηγορίες. Από την πλευρά του υλικού, η πλειονότητα σήμερα των ηλεκτρονικών συστημάτων είναι ψηφιακά, κυκλώματα δηλαδή που διαχειρίζονται ψηφιακή (*δυαδική*) πληροφορία.

Για να επιβεβαιωθεί ότι ένα κύκλωμα δουλεύει σύμφωνα με τις προδιαγραφές, πρέπει να ελεγχθεί. Ένας τέτοιος έλεγχος ανιχνεύει αστοχίες εξαιτίας κατασκευαστικών λαθών, αλλά και εξαιτίας της γήρανσης του συστήματος, των περιβαντολογικών αλλαγών και των διακυμάνσεων στην τάση τροφοδοσίας.

Τα διανύσματα εισόδου ή διανύσματα ελέγχου καθορίζουν τις τιμές στις οποίες πρέπει να τεθούν οι είσοδοι του κυκλώματος, ώστε η απόκρισή του να υποδεικνύει την ύπαρξη ή όχι ελαττωμάτων. Τα διανύσματα αυτά μπορεί να εξαχθούν είτε χειρονακτικά, είτε συνηθέστερα με τη βοήθεια προγραμμάτων αυτόματης παραγωγής διανυσμάτων δοκιμής (*AutomaticTestPatternGeneration - ATPG*). Το σύνολο των διανυσμάτων που τελικά προκύπτει λέγεται σύνολο διανυσμάτων δοκιμής (*testset*) το οποίο ανιχνεύει όλα τα σφάλματα του κυκλώματος

Τα διανύσματα ελέγχου εφαρμόζονται στο σύστημα με τη βοήθεια μιας συσκευής αυτομάτου ελέγχου (*AutomaticTestEquipment - ATE*). Η ίδια συσκευή είναι αυτή που συγκρίνει και τις δύο αποκρίσεις του κυκλώματος. Εξαιτίας όμως της πολυπλοκότητας της διαδικασίας ελέγχου, τα τελευταία χρόνια έχει εφαρμοστεί μια διαδικασία σχεδιασμού που στοχεύει στη δημιουργία ψηφιακών κυκλωμάτων που ελέγχονται εύκολα. Αυτός ο σχεδιασμός, γνωστός ως σχεδίαση για αυξημένη δοκιμαστικότητα (*DesignForTestability – DFT*) στοχεύει στην ενσωμάτωση δομών ελέγχου στο αρχικό κύκλωμα, με στόχο την καλύτερη ελεγξιμότητα (*controllability*) και παρατηρησιμότητα (*observability*) των εσωτερικών του κόμβων

2.4.1 Αναγκαιότητα του ελέγχου ενός κυκλώματος

Υπάρχουν αρκετοί λόγοι που υπαγορεύουν τον έλεγχο ενός ψηφιακού κυκλώματος. Όταν το κύκλωμα βρίσκεται στο στάδιο της σχεδίασης, ο έλεγχος με τη μορφή της διαδικασίας επαλήθευσης της σχεδίασης (*designverification*) προσπαθεί να διασφαλίσει ότι το κύκλωμα θα πληροί τις προδιαγραφές λειτουργίας και χρονισμού. Στο στάδιο της κατασκευής (υλοποίησης) ο έλεγχος αποσκοπεί στην ανεύρεση ελαττωμάτων που τυχόν προκύπτουν από την κατασκευαστική διαδικασία και που παρεμποδίζουν την αξιόπιστη λειτουργία και απόδοση του κυκλώματος. Όταν ένα κύκλωμα λειτουργεί, ο έλεγχος στοχεύει στη διαπίστωση της σωστής και απρόσκοπτης λειτουργίας του. Η μεγάλη τεχνολογική ανάπτυξη συνετέλεσε στην κατασκευή ολοκληρωμένων κυκλωμάτων που:

- Χαρακτηρίζονται από υψηλό βαθμό ολοκλήρωσης (*VLSI*) δίνοντας στον σχεδιαστή τη δυνατότητα κατασκευής κυκλωμάτων μικρότερων σε διαστάσεις αλλά αυξημένης πολυπλοκότητας που έχουν ταχύτερη απόκριση και λειτουργούν σε υψηλότερες συχνότητες.

- Έχουν χαμηλή τάση τροφοδοσίας καθιστώντας τα κυκλώματα πιο ευαίσθητα στο θόρυβο.

- Η σχεδίασή τους ακολουθεί τη φιλοσοφία του "ολοκληρωμένου συστήματος σε ένα chip" (*SOC*) δηλαδή την ενσωμάτωση στο ίδιο chip πολλών διαφορετικών ανεξάρτητων κυκλωμάτων για την υλοποίηση ενός συνθετότερου συστήματος.

Το μέγεθος και η πολυπλοκότητα των μοντέρνων VLSI κυκλωμάτων αυξάνει και την πολυπλοκότητα του τρόπου ελέγχου τους.

2.4.2 Έλεγχος Ψηφιακών Συστημάτων και επίπεδα περιγραφής τους

Έλεγχος ενός συστήματος είναι μια πειραματική διαδικασία κατά την οποία το σύστημα εξασκείται (*διεγείρεται*) με την εφαρμογή ενός συγκεκριμένου συνόλου τιμών εισόδων και η απόκρισή του αναλύεται για τη διακρίβωση της ορθής συμπεριφοράς του.

Ως ψηφιακό σύστημα ορίζεται ένα πολύπλοκο ψηφιακό κύκλωμα. Η φαινόμενη πολυπλοκότητα ενός κυκλώματος σχετίζεται με το αφαιρετικό επίπεδο περιγραφής της λειτουργίας του. Το επίπεδο περιγραφής ενός κυκλώματος χαρακτηρίζεται από το είδος της πληροφορίας που επεξεργάζεται. Αν και το ψηφιακό κύκλωμα μπορεί να θεωρηθεί ότι επεξεργάζεται αναλογικές ποσότητες όπως τάσεις και ρεύματα, το χαμηλότερο επίπεδο περιγραφής είναι το λογικό επίπεδο.

Ένα κύκλωμα θεωρείται ως σύστημα όταν η περιγραφή της λειτουργίας του, όσον αφορά τις λογικές τιμές που επεξεργάζεται, καθίσταται δυσνόητη. Στο επίπεδο περιγραφής των καταχωρητών (RTL επίπεδο) η πληροφορία που επεξεργάζεται το κύκλωμα αποτελείται από λέξεις που αποθηκεύονται σε καταχωρητές. Στο επόμενο επίπεδο περιγραφής το επίπεδο του συνόλου εντολών η πληροφορία ελέγχου οργανώνεται σε λέξεις που ονομάζονται εντολές.

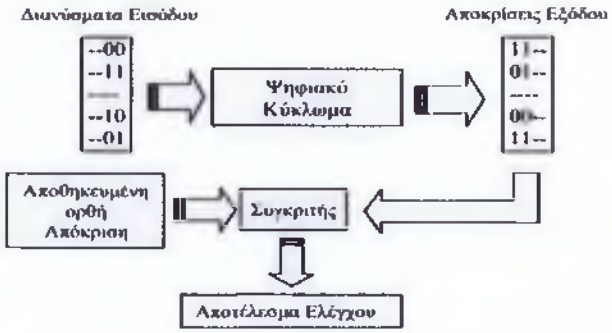
Η διέγερση και η απόκριση που ορίζουν ένα πείραμα ελέγχου καθορίζονται από το είδος της πληροφορίας που επεξεργάζεται το σύστημα υπό έλεγχο. Έτσι η διαδικασία ελέγχου μπορεί να καλύπτει εντελώς διαφορετικές δράσεις και περιβάλλοντα όπως για παράδειγμα:

- Ένα ή περισσότερα υποσυστήματα που αλληλοελέγχονται μέσω της αποστολής της λήψης μηνυμάτων.
- Ένας επεξεργαστής που αυτοελέγχεται εκτελώντας ένα διαγνωστικό πρόγραμμα.
- Συσκευές αυτόματου ελέγχου (*ATE*) που ελέγχουν ένα κύκλωμα εφαρμόζοντας σ'αυτό και παρατηρώντας ακολουθίες δυαδικών τιμών.

Τμήμα ελέγχου	Τμήμα Δεδομένων	Επίπεδο Περιγραφής
Λογικές τιμές (ή ακολουθίες λογικών τιμών)		Λογικό
Λογικές τιμές	Λέξεις	Καταχωρητών
Εντολές	Λέξεις	Σύνολο Εντολών
Προγράμματα	Δομές Δεδομένων	Επεξεργαστή
Μηνύματα		Σύστημα

Σχήμα 2.4.2 (α) : Επίπεδα περιγραφής στην επεξεργασία της πληροφορίας από ένα ψηφιακό σύστημα.

Στη βιομηχανία για τον έλεγχο των ψηφιακών κυκλωμάτων χρησιμοποιούνται ειδικές συσκευές αυτομάτου ελέγχου (*AutomaticTestEquipment*) που εφαρμόζουν μια σειρά από διανύσματα ελέγχου (*testvectors*) στους ακροδέκτες εισόδου του ολοκληρωμένου κυκλώματος και συγκρίνουν την απόκριση των ακροδεκτών εξόδου του με την αναμενόμενη απόκριση. Διαφορά στην απόκριση υποδηλώνει την ύπαρξη προβλήματος στο κύκλωμα που ακολουθεί:



Σχήμα 2.4.2 (β): Έλεγχος Ψηφιακού Κυκλώματος

2.5 Έλεγχος Ορθής Λειτουργίας

Από τη στιγμή που ένας σχεδιασμός έχει υλοποιηθεί σε πυρίτιο, μπορεί να επαληθευθεί εφαρμόζοντας τα κατάλληλα διανύσματα ελέγχου και ελέγχοντας τις αποκρίσεις του κυκλώματος. Αυτός ο έλεγχος ορθής λειτουργίας έχει σαν στόχο να επαληθεύσει την ορθότητα της κατασκευαστικής διαδικασίας. Από την άλλη πλευρά, η διαδικασία ελέγχου ορθής σχεδίασης του κυκλώματος κατά τις διάφορες φάσεις της σχεδίασης, ονομάζεται επαλήθευση ή επιβεβαίωση σχεδίασης (*designverification*) και πραγματοποιείται κυρίως με εξομοιωτή.

Όπως έχει ήδη αναφερθεί, σκοπός του ελέγχου είναι να μπορεί να αποδείξει ότι το κύκλωμα έχει κατασκευαστεί (ή συνεχίζει να λειτουργεί) χωρίς λάθη (*errorfree*). Πρέπει λοιπόν πρώτα να οριστούν τα προς ανίχνευση λάθη. Για να μπορέσουμε να διαχειριστούμε το πρόβλημα του ελέγχου, είναι προτιμότερο να αναπαραστήσουμε το αποτέλεσμα των ελαττωμάτων σε επίπεδο λογικών τιμών στους κόμβους που διασυνδέουν τα διάφορα συστατικά μέρη του κυκλώματος ή με άλλα λόγια, να «μοντελοποιήσουμε» τα ελαττώματα στο επίπεδο των λογικών πυλών. Αν η αντιστοίχηση που θα πραγματοποιηθεί είναι ένα προς ένα, θα προκύψει ένας μεγάλος αριθμός από μοντέλα σφαλμάτων (*faultmodel*). Για το λόγο αυτό, ένα σφάλμα ενός μοντέλου δεν είναι απαραίτητο να αναπαριστά ένα ελάττωμα επακριβώς. Θα πρέπει απλά η ανίχνευσή του να συνεπάγεται την ανίχνευση του ελαττώματος αυτού καθώς και άλλων πιθανών ελαττωμάτων στο υπό έλεγχο κύκλωμα. Με άλλα λόγια σε ένα σφάλμα ενός μοντέλου αντιστοιχούν περισσότερα από ένα ελαττώματα.

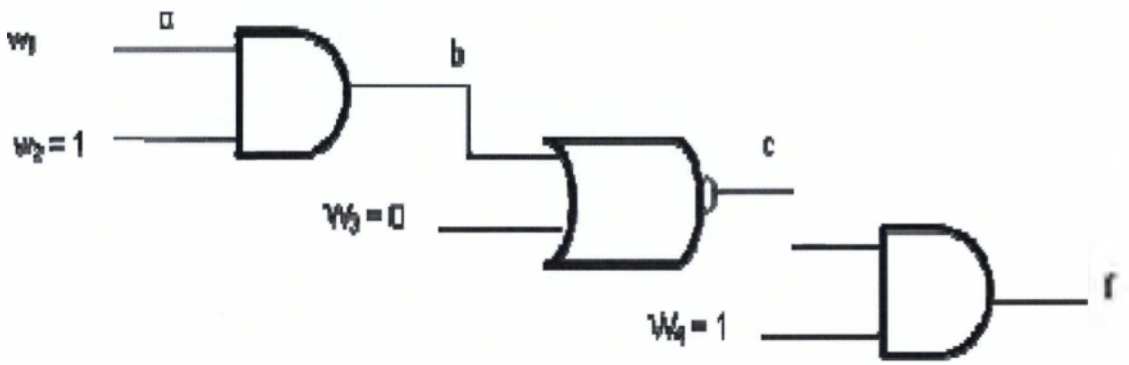
2.6 Πολυπλοκότητα ενός Συνόλου Δοκιμών

Η διαφορά ανάμεσα στον έλεγχο συνδυαστικών και ακολουθιακών κυκλωμάτων είναι μεγάλη, επειδή η συμπεριφορά ενός κυκλώματος υπό έλεγχο επηρεάζεται από τις εφαρμοζόμενες δοκιμές στις εξωτερικές του εισόδους αλλά και από τις καταστάσεις στις οποίες ευρίσκεται το κύκλωμα.

2.6.1 Ευαισθητοποίηση Διαδρομής

Μια εναλλακτική λύση είναι να θεωρήσουμε έναν αριθμό καλωδίων που σχηματίζουν μία διαδρομή ως μία οντότητα που μπορεί να ελεγχθεί για την ύπαρξη σφαλμάτων με τη βοήθεια μίας δοκιμής.

Ένα παράδειγμα αποτελεί ο έλεγχος ευαισθητοποιημένων διαδρομών όπως αυτό φαίνεται στο παρακάτω σχήμα:



Μία ευαισθητοποιημένη διαδρομή

2.7 Βλάβες Ψηφιακών Κυκλωμάτων

Οποιοδήποτε κύκλωμα μπορεί να πάθει βλάβη. Είναι σημαντικό, κυρίως σε πολύπλοκα κυκλώματα, π.χ. κυκλώματα υπολογιστών, να μπορούμε να ανιχνεύουμε έγκαιρα μία βλάβη όπως για παράδειγμα:

• Τις κύριες αιτίες των βλαβών.

• Τη σημασία του διαγνωστικού ελέγχου σ' ένα κύκλωμα.

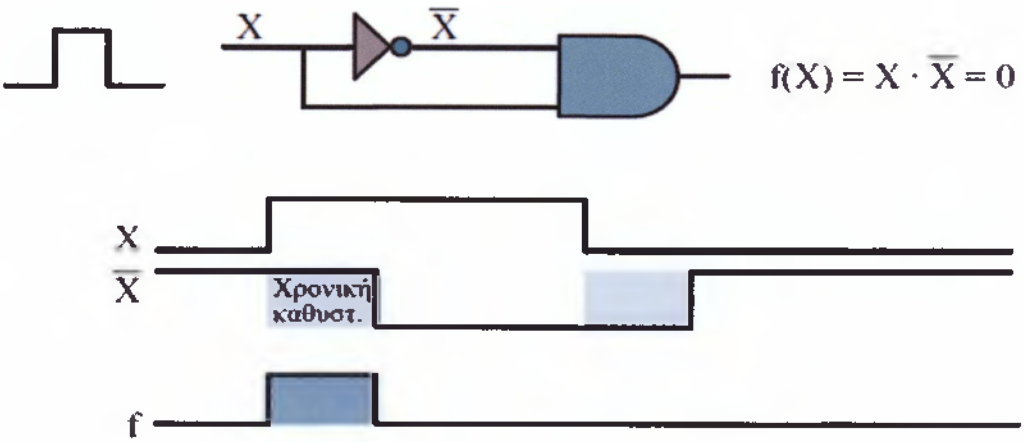
• Τις βασικές αρχές διάγνωσης s–a–0 και s–a–1 βλαβών..

Βλάβη ή ανώμαλη λειτουργία προκαλείται, όταν ένα ή περισσότερα στοιχεία του κυκλώματος αδυνατούν να εκτελέσουν την προβλεπόμενη αποστολή τους. Αποτέλεσμα είναι η μη κανονική συμπεριφορά του κυκλώματος, η δε βλάβη μπορεί να είναι είτε διαλείπουσα (*intermittent ή softfault*) είτε μόνιμη (*permanent ή hardfault*).

Οι κύριες αιτίες βλαβών είναι οι εξής:

α)Κακή σχεδίαση. Παραδείγματα:
α. Δεν ελήφθη υπόψη το FanOut κάποιας πύλης με αποτέλεσμα να προκληθεί αλλοίωση του παραγόμενου σήματος.

β. Δεν ελήφθη υπόψη ο χρόνος διαδόσεως του σήματος. Ας θεωρήσουμε το κύκλωμα του παρακάτω σχήματος. Η λογική συνάρτησή του είναι f(x)=0. Λόγω όμως του διαφορετικού χρόνου αφίξεως των σημάτων στην πύλη AND δημιουργείται στην έξοδό της παρασιτικός παλμός, του οποίου οι συνέπειες ενδέχεται να είναι βλαβερές (*hazard*).



β) Βλάβη στοιχείου. Η βλάβη ενός στοιχείου ή πύλης μπορεί να προέλθει είτε από τυχαία αιτία είτε από γήρανση του στοιχείου.

Η καλή σχεδίαση ενός κυκλώματος και γενικότερα ενός συστήματος αποβλέπει (α) στην ορθή "λογική" συμπεριφορά του κυκλώματος και (β) στην αποφυγή και προστασία, κατά το δυνατόν, από ενδεχόμενες βλάβες.

Έχουν αναπτυχθεί πολλές στρατηγικές σχεδίασης με σκοπό την αύξηση της αξιοπιστίας (*reliability*) ενός κυκλώματος, δηλαδή της προστασίας του από βλάβες και συνεπώς της αύξησης του χρόνου καλής λειτουργίας του. Η απλούστερη μέθοδος είναι η χρησιμοποίηση τριών ή περισσότερων όμοιων κυκλωμάτων σε παράλληλη λειτουργία και η εφαρμογή της αρχής της πλειοψηφίας. Όταν συμβεί βλάβη σ' ένα κύκλωμα, το σύστημα συνεχίζει να λειτουργεί σωστά, θεωρώντας σαν ορθή τη λογική κατάσταση στην οποία συμφωνεί η πλειοψηφία των κυκλωμάτων.

Η τεχνολογία των ολοκληρωμένων κυκλωμάτων προσφέρει μεγάλη αξιοπιστία και χάρη σ'αυτά έγινε δυνατή η πρακτική κατασκευή κυκλωμάτων πολύ μεγάλης πολυπλοκότητας. Παρά αυτό το πλεονέκτημα υπάρχουν δυσκολίες,

γιατί οποιαδήποτε βλάβη στο εσωτερικό τους είναι αδύνατο να επισκευασθεί, η δε διάγνωσή της σ' αυτά είναι δύσκολη για το λόγο ότι δεν είναι προσιτά προς εξέταση τα επιμέρους στοιχεία τους.

Στα μεγάλα ψηφιακά συστήματα και ολοκληρωμένα κυκλώματα το πλήθος των διαγνωστικών ελέγχων, που απαιτούν οι κλασικές μέθοδοι, αυξάνεται εκθετικά με το μέγεθος του κυκλώματος, ώστε η διαδικασία να γίνεται εξαιρετικά χρονοβόρα και αντιοικονομική. Για το λόγο αυτό έχει αναπτυχθεί σημαντικός κλάδος των ψηφιακών συστημάτων, που αποσκοπεί (α) στην εύρεση ταχύτερων διαγνωστικών μεθόδων και (β) στην εξαρχής πρόνοια στη σχεδίαση του κυκλώματος με σκοπό τη διευκόλυνση του διαγνωστικού ελέγχου στο έτοιμο κύκλωμα (*DesignforTestability*).

2.8 Τεχνικές Ελέγχου

Για την μελέτη των συστημάτων χρησιμοποιούμε «μοντέλα» (*models*) τα οποία δημιουργούν μια αναπαράσταση του συστήματος στον ηλεκτρονικό υπολογιστή με δομές δεδομένων και προγράμματα. Με την χρήση του μοντέλου μπορούμε να μελετήσουμε τα σήματα του συστήματος με την πάροδο του χρόνου.

Ένας σημαντικός παράγοντας του testing είναι η αξιολόγηση (*evaluation*) με την οποία αναφερόμαστε στην αποτελεσματικότητα ή την ποιότητα του ελέγχου. Με τον όρο ποιότητα αναφερόμαστε στον λόγο του αριθμού των σφαλμάτων που ανιχνεύονται προς τα συνολικά λάθη και αποδίδεται με τον όρο **κάλυψη σφαλμάτων (***faultcoverage***)**.

Οι τεχνικές ελέγχου μπορούν να ταξινομηθούν σύμφωνα με πολλά κριτήρια. Μερικά από αυτά είναι:

- **Έλεγχος με διαγνωστικά προγράμματα (***diagnosticprograms***),** όπου το σύστημα λειτουργεί σε κατάσταση αυτοδιάγνωσης και δημιουργεί εσωτερικά τα σήματα διέγερσης.

- **Έλεγχος κατά την λειτουργία (***on-linetesting***),** όπου τα σήματα διέγερσης παρέχονται από patterns που λαμβάνονται κατά την λειτουργία του συστήματος. Αυτό που ενδιαφέρει περισσότερο είναι οι ιδιότητες της απόκρισης του συστήματος και όχι τόσο η απόκριση.

- **Guided-probetesting,** είναι μια τεχνική που χρησιμοποιείται σε επίπεδο ελέγχου πλακέτας (*boardleveltesting*). Αν παρουσιαστεί ένα λάθος κατά τον αρχικό έλεγχο των ακροδεκτών τότε η συσκευή ελέγχου (*tester*) ειδοποιεί τον χειριστή να τοποθετήσει ένα probe σε μια γραμμή του συστήματος ώστε να μπορέσει να παρακολουθήσει την διάδοση των λαθών κατά μήκος μιας διαδρομής του συστήματος.

2.9 Διάγνωση s-α-0 και s-α-1 Βλαβών

Έστω μια πύλη F με δύο εισόδους A και B και έξοδο Z. Υπό κανονική λειτουργία έχουμε: Z = F(A,B). Εάν, όμως, συμβεί κάποια βλάβη στο κύκλωμα της πύλης, τότε η πύλη θα παρουσιάσει ένα από τα εξής συμπτώματα:

α. Η έξοδος Z παραμένει πάντοτε 0 (ή πάντοτε 1) ανεξάρτητα από τις τιμές εισόδου. Στην περίπτωση αυτή λέμε ότι η πύλη κρατείται στο 0 (ή στο 1) ή σύμφωνα με τη διεθνή ορολογία ότι είναι stack–at–zero (ή stack–at–one). Για συντομία οι βλάβες αυτές συμβολίζονται: s–a–0 και s–a–1.

β. Η πύλη λειτουργεί μερικώς και συμπεριφέρεται σαν μια είσοδός της, π.χ. η A, να κρατείται στο 0 (ή στο 1). Τότε λέμε ότι η A κρατείται στο 0 (ή A κρατείται στο 1) ή ότι A stack–at–0 (ή A stack–at–1). Π.χ. όταν μια πύλη OR φαίνεται να "αδιαφορεί" για την είσοδο της A, δηλ. εμφανίζει μονίμως Z = B, τότε θεωρείται ότι η βλάβη της είναι "A s–a–0", όταν δε μια πύλη AND συμπεριφέρεται ώστε η έξοδός της να είναι Z = B ανεξάρτητα από την είσοδο A, τότε η βλάβη της είναι "A s–a–1".

Οι βλάβες του τύπου s–a–0 και s–a–1 δεν είναι οι μοναδικές που μπορούν να συμβούν, είναι όμως οι συνηθέστερες και γι' αυτό παρουσιάζουν το μεγαλύτερο ενδιαφέρον. Η απλούστερη μέθοδος για τη διαπίστωση τέτοιας βλάβης είναι ο έλεγχος του πίνακα αληθείας του κυκλώματος.Εάν όμως το κύκλωμα έχει n εισόδους, τότε – ως γνωστόν– πρέπει να ελεγχθούν 2n συνδυασμοί τιμών (εκθετική αύξηση δαπάνης) με αποτέλεσμα ο έλεγχος να γίνεται ασύμφορος.

Ας εξετάσουμε την περίπτωση μιας πύλης OR δύο εισόδων, A και B, και εξόδου Z. Στον Πίνακα 4.7 παρουσιάζονται ο πίνακας αληθείας της κανονικής πύλης OR και οι πίνακες που προκύπτουν στις περιπτώσεις που η είσοδος A είναι A s–a–0 και A s–a–1. Με [0] και [1] δηλώνονται οι αντίστοιχες stack–at–τιμές.

Παράδειγμα:

	πύλη OR			Z sa0	Z sa1	A s-a-0			A s-a-1		
α/α	A	B	Z	Z	Z	A	B	Z	A	B	Z
0	0	0	0	0	[1]	0	0	0	[1]	0	[1]
1	0	1	1	[0]	1	0	1	1	[1]	1	1
2	1	0	1	[0]	1	[0]	0	[0]	1	0	1
3	1	1	1	[0]	1	[0]	1	1	1	1	1

Πίνακας Αληθείας της OR για βλάβες s-α-0 και s-α-1

Από το παραπάνω παράδειγμα φαίνεται ότι για τον έλεγχο ενός ψηφιακού κυκλώματος συνήθως αρκεί να δοκιμαστεί ένα υποσύνολο μόνο των δυνατών συνδυασμών τιμών των εισόδων, συντομεύοντας έτσι σημαντικά τον έλεγχο. Το υποσύνολο, το οποίο περιλαμβάνει το μικρότερο δυνατό αριθμό συνδυασμών για τον πλήρη έλεγχο ενός κυκλώματος, ονομάζεται αναγκαίο σύνολο δοκιμών (*EssentialTestSet*). Η εύρεση, όμως, του αναγκαίου συνόλου δοκιμών για ένα μεγάλο κύκλωμα είναι από μόνη της μια εξαιρετικά πολύπλοκη διαδικασία

2.9.1 Διάγνωση και Επισκευή

Για την διάγνωση των σφαλμάτων υπάρχουν δύο προσεγγίσεις. Η πρώτη προσέγγιση είναι μια ανάλυση αιτίου και αποτελέσματος (*cause-effectanalysis*) κατά την οποία απαριθμούνται όλα τα πιθανά σφάλματα που υπάρχουν σε ένα μοντέλο και οι αντίστοιχες αποκρίσεις τους. Έτσι δημιουργείται μια βάση δεδομένων που λέγεται λεξικό σφαλμάτων (*faultdictionary*). Η δεύτερη προσέγγιση ακολουθεί αντίστροφη διαδικασία. Λαμβάνει μια πραγματική απόκριση του συστήματος και προσπαθεί να προσδιορίσει απευθείας μόνο τα σφάλματα που μπορούν να παράγουν αυτή την απόκριση.

2.9.2 Παραγωγή Ελέγχου (*TestGeneration*)

Παραγωγή Ελέγχου ((*TestGeneration*(*TG*)) είναι η διαδικασία για τον καθορισμό μιας κατάλληλης εισόδου για τον έλεγχο ενός κυκλώματος. Το TG μπορεί να είναι είτε προσανατολισμένη προς τα σφάλματα είτε προς την λειτουργία. Στην πρώτη περίπτωση προσπαθούμε να δημιουργήσουμε συγκεκριμένα tests για να διαγνώσουμε συγκεκριμένα σφάλματα, ενώ στην δεύτερη περίπτωση προσπαθούμε να δημιουργήσουμε κατάλληλα tests τα οποία αν το σύστημα τα περάσει τότε λέμε ότι πραγματοποιεί την συγκεκριμένη λειτουργία.

2.9.3 Κόστος

Το κόστος του ελέγχου ενός συστήματος είναι ένας σημαντικός παράγοντας στο κόστος παραγωγής και συντήρησης του συστήματος. Για τον λόγο αυτό έχουν αναπτυχθεί τεχνικές σχεδίασης για **testability**, οι οποίες αρκετές φορές μπορούν να υπαγορεύσουν την δομή της σχεδίασης.

2.10 Βασικές Έννοιες

Ένα ψηφιακό σύστημα μπορούμε να το θεωρήσουμε σαν ένα «μαύρο κουτί» το οποίο επεξεργάζεται τις πληροφορίες που δέχεται στις εισόδους του και παράγει τις εξόδους του. Η **συμπεριφορά** του συστήματος καθορίζεται από την αντιστοίχιση εισόδων – εξόδων (*I/Omapping*) που κάνει το μαύρο κουτί. Ο μετασχηματισμός των εισόδων γίνεται σε συνάρτηση με τον χρόνο. Σε αρκετές περιπτώσεις είναι πιο βολικό, στην μελέτη του συστήματος, να διαχωρίσουμε τον τομέα των τιμών από τον τομέα του χρόνου. Έτσι έχουμε την **λογική συνάρτηση** που είναι ο μετασχηματισμός των τιμών των εισόδων για να πάρουμε τις εξόδους, χωρίς χρονικές σχέσεις. Το **λειτουργικό μοντέλο** (*functionalmodel*) είναι μια αναπαράσταση της λογικής συνάρτησης του συστήματος. Προσθέτοντας στο λειτουργικό μοντέλο την αναπαράσταση του χρόνου, έχουμε το **μοντέλο συμπεριφοράς (behavioralmodel)** του συστήματος. Το **μοντέλο δομής** (*structuralmodel*) του συστήματος περιγράφει το κουτί σαν μια συλλογή διασυνδεδεμένων μικρότερων κουτιών που λέγονται στοιχεία ή συνθετικά. Ο τρόπος περιγραφής είναι ιεραρχικός και τα κουτιά του τελευταίου επιπέδου λέγονται πρωταρχικά στοιχεία (*primitiveelements*).

Η λειτουργία ενός στοιχείου φαίνεται από τον τύπο του, ο οποίος δηλώνεται γραφικά με ειδικά σχήματα. Ακόμη μπορούμε να διακρίνουμε το **εξωτερικό μοντέλο** (*externalmodel*) του συστήματος, που είναι το μοντέλο που βλέπει ο χρήστης και το **εσωτερικό μοντέλο** (*internalmodel*) το οποίο αποτελείται από τις δομές δεδομένων και τα προγράμματα που παριστάνουν το σύστημα σε έναν υπολογιστή.

2.11 Μοντελοποίηση σε Λογικό Επίπεδο

Τα κυκλώματα μοντελοποιούνται σε επίπεδο λογικής (*logiclevel*) ενώ τα συστήματα μοντελοποιούνται σε επίπεδο καταχωρητών (*registerlevel*).
Ο απλούστερος τρόπος για να περιγράψουμε ένα συνδυαστικό κύκλωμα είναι με τον **πίνακα αληθείας** (*truthtable*) του. Για να έχουμε μια πιο συμπαγή αναπαράσταση, στην περίπτωση που ο πίνακας είναι μεγάλος και περίπλοκος, μπορούμε να χρησιμοποιήσουμε την κυβική αναπαράσταση (*cubicalnotation*). Μια ακολουθιακή μηχανή πεπερασμένων καταστάσεων μπορεί να παρασταθεί με έναν **πίνακα καταστάσεων** (*statetable*). Στον πίνακα αυτόν κάθε γραμμή αντιστοιχεί σε μια εσωτερική κατάσταση της μηχανής, ενώ κάθε στήλη αντιστοιχεί σε κάθε πιθανή είσοδο. Το περιεχόμενο του πίνακα στην διασταύρωση μιας γραμμής και μιας στήλης δείχνει την επόμενη κατάσταση της μηχανής.

		x	
		0	1
q	1	2,1	3,0
	2	2,1	4,0
	3	1,0	4,0
	4	3,1	4,0

Για την αναπαράσταση μιας ακολουθιακής συνάρτησης μ' αυτόν τον τρόπο κάνουμε την υπόθεση ότι υπάρχει συγχρονισμός με την χρήση μιας επιπλέον γραμμής εισόδου, της γραμμής ρολογιού.

Ακολουθιακά κυκλώματα μπορούν να σχεδιασθούν και χωρίς ρολόι, οπότε έχουμε τα ασύγχρονα κυκλώματα. Στην περίπτωση αυτή, η συμπεριφορά του συστήματος καθορίζεται από τον **πίνακα ροής** (*flowtable*). Στον πίνακα ροής μια statetransition μπορεί να περιλαμβάνει πολλές αλλαγές κατάστασης, που προκαλούνται από μια μόνο αλλαγή στην είσοδο, μέχρι να φτάσουμε σε μια σταθερή κατάσταση. Ένας άλλος τρόπος αναπαράστασης μιας ακολουθιακής συνάρτησης είναι τα **binarydecisiondiagrams**. Αυτά είναι μια γραφική αναπαράσταση της λειτουργίας του κυκλώματος. Μια απλή διάσχιση του γραφήματος καθορίζει την τιμή της εξόδου εξετάζοντας με την σειρά τις τιμές των εισόδων.

Τέλος μια διαφορετική προσέγγιση στην μοντελοποίηση της λειτουργίας ενός κυκλώματος είναι απείθειας με πρόγραμμα. Αυτός ο τρόπος κωδικοποίησης χρησιμοποιείται για τα πρωταρχικά στοιχεία που συνθέτουν του μοντέλο δομής.

2.11.1 Λειτουργική Μοντελοποίηση σε Λογικό Επίπεδο

Η περιγραφή ενός μοντέλου για κάποιο σύστημα γίνεται με την χρήση μιας γλώσσας RTL σε επίπεδο καταχωρητή (*register*) ή συνόλου εντολών (*instructionset*). Οι λέξεις αποθηκεύονται σε καταχωρητές και οι μνήμες είναι οργανωμένες σαν πίνακες από καταχωρητές (arraysofregisters). Έτσι για να ορίσουμε έναν καταχωρητή 8-bits με όνομα IR γράφουμε: registerIR [0 → 7], ενώ η περιγραφή μιας μνήμης 256 θέσεων των 8-bits γίνεται με την δήλωση: **memoryABC** [0 → 255; 0 → 15].

Οι διαδρομές των δεδομένων καθορίζονται έμμεσα με την περιγραφή της επεξεργασίας που γίνεται και της μεταφοράς των λέξεων μεταξύ των καταχωρητών. Τα μοντέλα που ορίζονται με την βοήθεια μιας γλώσσας RTL χαρακτηρίζονται σαν λειτουργικά μοντέλα, αφού το μεγαλύτερο βάρος δίνεται στην λειτουργική περιγραφή. Από την άλλη πλευρά παρέχονται μόνο περιληπτικές πληροφορίες για την δομή του μοντέλου.

2.12 Γλώσσες RTL

Οι γλώσσες **RTL** χωρίζονται σε διαδικασιακές (*procedural*) και μη-διαδικασιακές (*nonprocedural*). Οι πρώτες είναι παρόμοιες με τις συμβατικές γλώσσες προγραμματισμού όπου οι προτάσεις εκτελούνται σειριακά και το αποτέλεσμα μιας πρότασης είναι άμεσα διαθέσιμο για την επόμενη. Αντίθετα οι προτάσεις των μη-διαδικασιακώνRTL γλωσσών εκτελούνται παράλληλα. Ένα μοντέλο RTL μπορεί εσωτερικά να παριστάνεται είτε με δομές δεδομένων είτε με μεταγλωττισμένο κώδικα. Στην πρώτη περίπτωση το μοντέλο μπορεί να μεταφρασθεί για διαφορετικές εφαρμογές από κατάλληλα προγράμματα. Στην δεύτερη περίπτωση το μοντέλο χρησιμοποιείτε μόνο για προσομοίωση.

2.13 Μοντέλα Δομής (*StructuralModels*)

Ένα τυπικό μοντέλο δομής ενός συστήματος σε μια γλώσσα σύνδεσης (*connectivitylanguage*) καθορίζει τις γραμμές εισόδου/εξόδου (*I/Olines*), τα τμήματά του (*components*) και τα σήματα εισόδου/εξόδου για κάθε τμήμα (*I/Osignals*). Μια γραμμή αναφέρεται επίσης και σαν δίκτυο (*net*). Κάθε μοντέλο αποτελείται από πρωτεύοντα τμήματα (systemprimitivecomponents) και από τμήματα που έχουν ορισθεί προηγουμένως και βρίσκονται σε βιβλιοθήκες (*libraryofcomponents*). Συνήθως τα μοντέλα δομής κτίζονται με μακροεντολές (*macroapproach*). Η δυνατότητα των μακροεντολών είναι μια τεχνική που βασίζεται σε τεχνικές επεξεργασίας κειμένου (*text-processing*technique), κατά τις οποίες σε ένα σώμα κειμένου αποδίδεται ένα όνομα. Μια αναφορά στο όνομα της μακροεντολής έχει σαν αποτέλεσμα την αντικατάστασή του από το σώμα της μακροεντολής.

2.13.1 Δομικές Ιδιότητες (*StructuralProperties*)

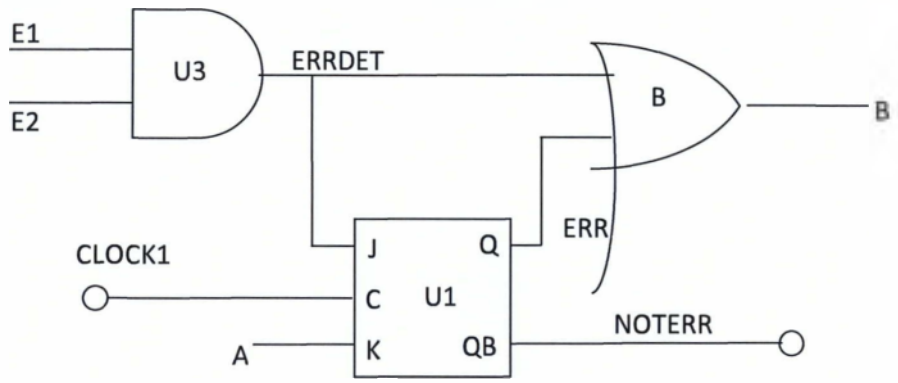
Το μοντέλο της δομής ενός συστήματος μπορεί να παρασταθεί με την βοήθεια ενός γράφου. Στην περίπτωση που ένα δίκτυο ενώνει μόνο δύο τμήματα τότε μιλάμε για κατευθυνόμενο γράφο (directedgraph). Γενικότερα ένα δίκτυο μπορεί να μεταδίδει ένα σήμα από μια πηγή σε περισσότερους από έναν προορισμούς. Σε ένα τέτοιο σήμα λέμε ότι υπάρχει fanout. Ενώ ένα κύκλωμα που δεν έχει καθόλου fanout λέγεται **fanout-free**. Με τον όρο επανασυγκλίνωνfanout (reconvergentfanout) αναφερόμαστε σε διαφορετικές διαδρομές από το ίδιο σήμα που καταλήγουν στο ίδιο τμήμα. Η παρουσία επανασυγκλίνοντοςfanout δημιουργεί πολλά προβλήματα στην δημιουργία δοκιμών ελέγχου και στην διάγνωση.

2.13.2 Εσωτερική Αναπαράσταση

Η εσωτερική αναπαράσταση του μοντέλου δομής γίνεται χρησιμοποιώντας τυπικές δομές δεδομένων. Οι δομές αυτές αποτελούνται από δύο set παράλληλων πινάκων:

1. του πίνακα στοιχείων (*elementtable*) και του πίνακα σημάτων (*signaltable*)
2. του πίνακα Fanin (*Fanintable*) και του πίνακα Fanout (*Fanout table*)

- Κάθε στοιχείο αναγνωρίζεται από έναν δείκτη στον πίνακα των στοιχείων και κάθε σήμα από έναν δείκτη στον πίνακα των σημάτων:



ELEMENT TABLE

	NAME	TYPE	Nout	OUT	NFI	FIN
1	U1	JKFF	2	1	3	1
2	B	OR	1	8	2	4
3	U3	AND	1	3	2	6
4	CLOCK1	PI	1	4	0	-
5	NOTERR	PO	0	-	1	8

SIGNAL TABLE

	NAME	SOURCE	NFO	FOUT
1	ERR	1	2	1
2	NOTERR	1	1	3
3	ERRDET	3	2	4
4	CLOCK1	4	1	6
5	E1	...	1	7
6	E2	...	1	8
7	A	...	1	9
8	B	2	...	...

	FANIN TABLE INPUTS
1	3
2	4
3	7
4	1
5	3
6	5
7	6
8	2

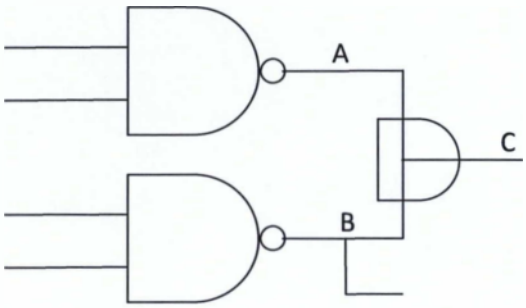
	FANOUT TABLE FANOUTS
1	2
2	…
3	5
4	1
5	2
6	1
7	3
8	3
9	1

Η παραπάνω δομή μπορεί να επεκταθεί για διαφορετικές εφαρμογές προσθέτοντας στήλες όπως καθυστέρηση (*delay*), τιμές των σημάτων (*signalvalues*)κλπ. Μια εναλλακτική προσέγγιση μπορεί να γίνει με την χρήση υποσυστημάτων (subsystemapproach). Η προσέγγιση αυτή επιτρέπει μια ιεραρχική εσωτερική αναπαράσταση του συστήματος με δύο επίπεδα ιεραρχίας. Στο επίπεδο κορυφής υπάρχουν τα υποσυστήματα και στο επόμενο επίπεδο υπάρχουν οι διασυνδέσεις μεταξύ των υποσυστημάτων. Με την προσέγγιση αυτή γίνεται σημαντική εξοικονόμηση μνήμης, αλλά από την άλλη πλευρά έχουμε μεγαλύτερη πολυπλοκότητα και ένα μικρό overhead λόγω της συνεχούς εναλλαγής μεταξύ των δύο επιπέδων.

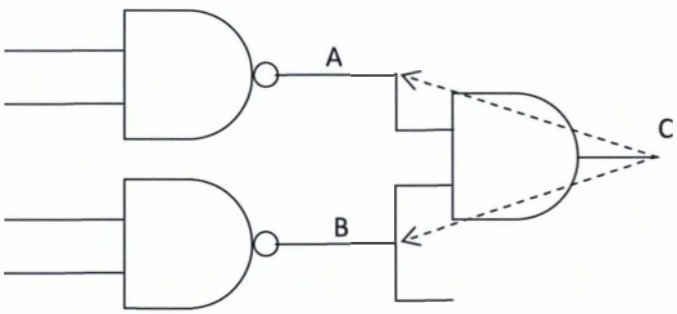
2.14 Μοντελοποίηση Εξόδου με Απευθείας Σύνδεση (*WiredLogic*)

Σε αρκετές τεχνολογίες είμαστε αναγκασμένοι να συνδέσουμε απευθείας πολλές εξόδους. Έτσι το συνδεδεμένο δίκτυο δημιουργεί μια νέα λογική συνάρτηση. Ο μηχανισμός αυτός ονομάζεται λογική της γραμμής (*wiredlogic*) για να δηλώσει ότι η λογική συνάρτηση προέρχεται από την γραμμή και όχι από κάποιο στοιχείο. Ένας απλός τρόπος για να μοντελοποιήσουμε τον μηχανισμό είναι να εισάγουμε μια εικονική πύλη (*dummygate*) η οποία θα πραγματοποιεί την λειτουργία της γραμμής. Αν και οι τιμές εξόδου θα έχουν τις σωστές τιμές η αμφίδρομη λειτουργία που εννοείται από την εικονική πύλη μπορεί να οδηγήσει σε λάθος τιμές για τις εισόδους.

Μια πιο σωστή τεχνική μοντελοποίησης είναι να εμφανίσουμε τις εισόδους της πύλης σαν fanouts της εξόδου.



(α)



(β)

2.15 Τύποι Προσομοίωσης

Οι προσομοιωτές διακρίνονται σε κατηγορίες ανάλογα τον τύπο του εσωτερικού μοντέλου που επεξεργάζονται. Έτσι αν εκτελούν ένα μοντέλο μεταγλωττισμένου κώδικα λέγονται **compiledsimulators**. Αν εκτελούν ένα μοντέλο που βασίζεται σε δομές δεδομένων λέγονται table-drivensimulators. Στην περίπτωση που ο προσομοιωτής προσομοιώνει μόνο το ενεργό τμήμα του κυκλώματος τότε κάνουμε λόγο για activity-directed ή event-drivensimulation. Με τον όρο activity εννοούμε τον λόγο των ενεργών σημάτων προς τα συνολικά σήματα που υπάρχουν σε ένα κύκλωμα, ενώ με τον όρο γεγονός (*event*) εννοούμε μια αλλαγή στην τιμή ενός σήματος.

2.15.1 Λογική Προσομοίωση (*LogicSimulation*)

Η λογική προσομοίωση είναι ένας τρόπος για να ελέγξουμε την σχεδίαση του συστήματος και να επιβεβαιώσουμε ότι ανταποκρίνεται στα προκαθορισμένα χαρακτηριστικά του, τόσο σε επίπεδο λειτουργίας όσο και σε επίπεδο χρονισμού. Ο έλεγχος γίνεται με σύγκριση των τιμών που προκύπτουν από την προσομοίωση με τις αναμενόμενες τιμές που προέρχονται από τα χαρακτηριστικά.

Με την προσομοίωση ελέγχουμε αν η λειτουργία του συστήματος:

- είναι σωστή και ανεξάρτητη από την αρχική κατάσταση (*power-onstate*),
- δεν είναι ευαίσθητη σε πιθανές αλλαγές στις καθυστερήσεις,
- είναι απαλλαγμένη από κρίσιμες διαδρομές (*criticalraces*), oscillations, μη αποδεκτές καταστάσεις εισόδου, και καταστάσεις «κρεμάσματος» (*hang-upstates*).

Παραδοσιακά οι σχεδιαστές χρησιμοποιούσαν την πρωτοτυποποίηση για να ελέγξουν μια νέα σχεδίαση. Το πρωτότυπο έχει το πλεονέκτημα ότι λειτουργεί την ταχύτητα του συστήματος αλλά είναι πολύ ακριβό και χρονοβόρο. Η προσομοίωση αντικατέστησε τα πρωτότυπα αν και έχει πιο αργή εκτέλεση, γιατί έχει πολλά πλεονεκτήματα, όπως:

- μπορούμε να ελέγξουμε τις καταστάσεις λάθους
- την δυνατότητα να αλλάξουμε τους χρόνους καθυστέρησης για να ελέγξουμε τους χρόνους
- να ελέγξουμε τον χρόνο σε ασύγχρονα κυκλώματα κλπ

2.15.2 Προβλήματα στην Προσομοίωση με βάση την επαλήθευση του σχεδιασμού

Κατά τον έλεγχο ενός συστήματος με προσομοίωση υπάρχουν τρία αλληλοεπηρεαζόμενα προβλήματα.

- Ο τρόπος δημιουργίας των τιμών εισόδου (*testgeneration*)
- Η αποτίμηση της πληρότητας των tests ελέγχου (*testevaluation*)
- Η ορθότητα των αποτελεσμάτων

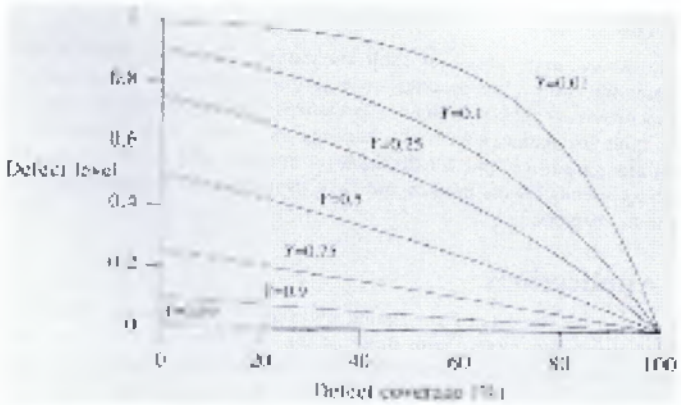
Για την δημιουργία tests ελέγχου χρησιμοποιούμε testcases που ελέγχουν συγκεκριμένες συμπεριφορές του μοντέλου. Υπάρχει διαφορά στην δημιουργία **tests**για έλεγχο της σχεδίασης και εντοπισμό των αντίστοιχων λαθών και στην δημιουργία tests για εντοπισμό φυσικών σφαλμάτων.

2.15.3 Προσομοίωση Σφαλμάτων (*FaultSimulation*)

Με το όρο προσομοίωση σφαλμάτων (*faultsimulation*) εννοούμε την προσομοίωση ενός κυκλώματος υπό την παρουσία σφαλμάτων. Εξετάζοντας τα αποτελέσματα που προκύπτουν από την προσομοίωση χωρίς σφάλματα με τα αποτελέσματα υπό την παρουσία σφαλμάτων μπορούμε να καθορίσουμε τα σφάλματα που ανιχνεύει ένα testT.

Η αξιολόγηση ενός test γίνεται από την κάλυψη σφάλματος (*faultcoverage)* που είναι ο λόγος του αριθμού των σφαλμάτων που ανιχνεύει το testT προς τον συνολικό αριθμό των λαθών που προσομοιώνονται. Η κάλυψη σφάλματος (*faultcoverage*) είναι μόνο το κατώτερο όριο της κάλυψης του ελαττώματος (*defectcoverage*), το οποίο είναι η πιθανότητα το testT να ανιχνεύσει ένα φυσικό σφάλμα στο κύκλωμα.

Η ποιότητα ενός test επηρεάζει σε μεγάλο βαθμό την ποιότητα του προϊόντος που παράγεται. Αν Y είναι η απόδοση παραγωγής (*manufacturingyield*) που είναι η πιθανότητα ένα κύκλωμα να μην έχει σφάλματα, DL είναι το επίπεδο ελαττώματος (*defectlevel*) που είναι η πιθανότητα να δώσουμε στην κατανάλωση ένα ελαττωματικό προϊόν, και d η κάλυψη ελαττώματος (*defectcoverage*) του test που χρησιμοποιήθηκε για τον έλεγχο των λαθών παραγωγής τότε ισχύει: $DL = 1 - Y^{1-d}$



Επίπεδο ελαττώματος σαν συνάρτηση της απόδοσης και της κάλυψης ελαττωμάτων.

Η προσομοίωση παίζει σημαντικό ρόλο στην δημιουργία των κατάλληλων test ελέγχου. Πολλά συστήματα χρησιμοποιούν προσομοιωτές για να αξιολογήσουν ένα testT και στην συνέχεια αλλάζουν το test μέχρι να επιτευχθεί η επιθυμητή κάλυψη ελαττωμάτων (*faultcoverage*).

2.15.4 Βασικές Τεχνικές Προσομοίωσης

Οι βασικές τεχνικές προσομοίωσης που χρησιμοποιούνται είναι:

- Η παράλληλη προσομοίωση (*parallelfaultsimulation*): το «καλό» κύκλωμα και ένας αριθμός W από κυκλώματα με σφάλματα προσομοιώνονται ταυτόχρονα
- Η συμπερασματική προσομοίωση (*deductivefaultsimulation*): προσομοιώνεται το «καλό» κύκλωμα και από τα αποτελέσματα βγάζει συμπεράσματα για την συμπεριφορά όλων των κυκλωμάτων με σφάλματα. Η έκφραση «όλων των κυκλωμάτων» είναι μια θεωρητική δυνατότητα που εξαρτάται από το μέγεθος της διαθέσιμης μνήμης.

- Η προσομοίωση συμφωνίας (*concurrentfaultsimulation*): βασίζεται στην παρατήρηση ότι στις περισσότερες περιπτώσεις, οι περισσότερες τιμές στα κυκλώματα με λάθη συμφωνούν με τις τιμές στο «καλό» κύκλωμα. Η προσομοίωση συμφωνίας προσομοιώνει το «καλό» κύκλωμα και για κάθε κύκλωμα με σφάλματα προσομοιώνει εκείνα τα στοιχεία του, που διαφέρουν από τις αντίστοιχες τιμές στο «καλό» κύκλωμα.

2.16 Η άγνωστη λογική τιμή

Η απόκριση ενός ακολουθιακού κυκλώματος εξαρτάται από την αρχική του κατάσταση. Στην περίπτωση όμως της εκκίνησης, η αρχική κατάσταση της μνήμης είναι συνήθως απροσδιόριστη. Αυτό γίνεται γιατί πριν ξεκινήσει η κανονική λειτουργία, εφαρμόζεται μια ακολουθία αρχικοποίησης που φέρνει το κύκλωμα στην κατάσταση reset. Για να επεξεργαστούμε την αρχική άγνωστη κατάσταση, ο αλγόριθμος προσομοίωσης χρησιμοποιεί μια λογική τιμή που δηλώνεται με το u, και δείχνει μια άγνωστη λογική τιμή. Η τιμή u συμμετέχει στις πράξεις μαζί με τις τιμές 0 και 1, έτσι επεκτείνονται οι γνωστές πράξεις της άλγεβρας Boole στην λογική των τριών τιμών 0, 1 και u. Προβλήματα προκύπτουν κατά την προσομοίωση αν κάποιες γραμμές έχουν τιμή u.

Γενικά αν K γραμμές ενός στοιχείου έχουν τιμή u τότε υπάρχουν 2^K δυνατοί συνδυασμοί τους οποίους μπορεί να εκτελέσει το στοιχείο. Αυτό έχει σαν αποτέλεσμα αύξηση του κόστους καθώς για ακριβή μελέτη πρέπει να εξετάσουμε όλες τις δυνατές περιπτώσεις.

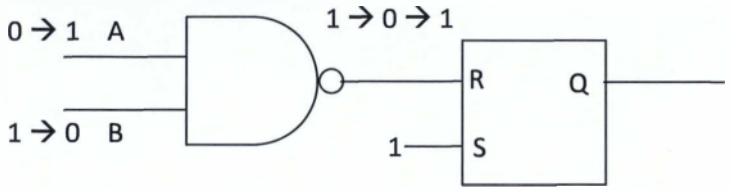
2.17 Μοντέλα Καθυστέρησης (*DelayModels*)

Η μοντελοποίηση της καθυστέρησης είναι ένα στοιχείο κλειδί μεταξύ της ακρίβειας και της πολυπλοκότητας του αλγόριθμου της προσομοίωσης. Κάθε πύλη εισάγει μια καθυστέρηση στα σήματα που διαδίδονται μέσα από αυτήν. Το βασικό μοντέλο καθυστέρησης είναι αυτό της **καθυστέρησης μεταφοράς** (*transportdelay*), το οποίο καθορίζει το διάστημα d το οποίο χωρίζει την αλλαγή στην τιμή της εξόδου από τις αλλαγές στις εισόδους που την προκάλεσαν. Ένα άλλο μοντέλο καθυστέρησης είναι αυτό της **αδράνειας της εισόδου** (*inputinertialdelay*), το οποίο καθορίζει την ελάχιστη χρονική διάρκεια d_I που απαιτείται για μια αλλαγή στην είσοδο ώστε η έξοδος να αλλάξει κατάσταση. Αν η χρονική διάρκεια είναι μικρότερη από τον χρόνο d_I τότε έχουμε ένα spike το οποίο φιλτράρεται από την πύλη.

Σε κυκλώματα υψηλής ταχύτητας η καθυστέρηση από τις γραμμές μεταφοράς είναι εξίσου σημαντική με την καθυστέρηση από τα στοιχεία του κυκλώματος. Η καθυστέρηση αυτή εξαρτάται από το μήκος των γραμμών και γίνεται γνωστή μόνο μετά από την ολοκλήρωση του routing.

Πρόβλημα δημιουργείται αν μια γραμμή σήματος έχει περισσότερα από ένα fanout καθένα από τα οποία εισάγει την δικιά του καθυστέρηση. Σε τέτοιες περιπτώσεις η μοντελοποίηση γίνεται με εισαγωγή κατάλληλων στοιχείων καθυστέρησης (*delayelements*).

-Ανίχνευση κινδύνων



Έστω ότι συμβαίνουν οι αλλαγές στις εισόδους που φαίνονται στο παραπάνω σχήμα. Αν οι αλλαγές αυτές είναι τέτοιες που για ένα μικρό χρονικό διάστημα A=B=1 τότε η έξοδος μπορεί να έχει έναν παλμό 1 → 0 → 1 ο οποίος μπορεί να κάνει reset το latch. Αυτή η πιθανή εμφάνιση ενός μεταβατικού παλμού σε μια γραμμή σήματος ονομάζεται στατικό κίνδυνο,
(*statichazard*). Αντίθετα η εμφάνιση ενός μεταβατικού παλμού κατά την διάρκεια μιας αλλαγής ενός σήματος 0 → 1 ή 1 → 0, ονομάζεται ανίχνευση κινδύνων (*dynamichazard).*

2.18 Μοντέλα Λογικών Σφαλμάτων (*LogicalFaultModels*)

Τα λογικά σφάλματα (**logicalfaults**) αντιπροσωπεύουν το αποτέλεσμα των φυσικών σφαλμάτων (**physicalfaults**) στην συμπεριφορά του μοντέλου του συστήματος. Στην μοντελοποίηση των διαφόρων τμημάτων κάνουμε έναν διαχωρισμό μεταξύ της λογικής συνάρτησης και του χρονισμού (*timing*). Έτσι διακρίνουμε σφάλματα που επηρεάζουν την λογική συνάρτηση και σφάλματα καθυστέρησης που επηρεάζουν την ταχύτητα λειτουργίας του κυκλώματος.

Μοντελοποιώντας φυσικά σφάλματα σαν λογικά σφάλματα κερδίσουμε ότι μειώνεται η πολυπλοκότητα του προβλήματος, ότι έχουμε ανεξαρτησία από την τεχνολογία και ότι μπορούμε να εφαρμόσουμε τα λογικά σφάλματα και σε φυσικά τα οποία δεν έχουμε καταλάβει τον τρόπο με τον οποίο επηρεάζουν την λειτουργία του κυκλώματος.

Ένα μοντέλο για λογικά σφάλματα μπορεί να είναι είτε σαφώς ορισμένο είτε υπονοούμενο. Στην πρώτη περίπτωση έχουμε το σαφές μοντέλο σφάλματος(**explicitfaultmodel**) ενώ στην δεύτερη το σιωπηρό μοντέλο σφάλματος(**implicitfaultmodel**). Στο σαφές μοντέλο σφάλματος (**explicitfault**) κάθε ξεχωριστό σφάλμα είναι αναγνωρίσιμο και έτσι μπορεί να γίνει απαρίθμηση των σφαλμάτων που θα αναλυθούν. Στο σιωπηρό μοντέλο σφάλματος (**implicitfaultmodel**) γίνεται μια περιληπτική περιγραφή των σφαλμάτων που θα αναγνωρίζονται και καθορίζονται οι χαρακτηριστικές τους ιδιότητες.

2.18.1 Κατηγορίες Σφαλμάτων

Διακρίνουμε τις εξής κατηγορίες σφαλμάτων:

- **Διαρθρωτικά Σφάλματα**: είναι σφάλματα που σχετίζονται με το δομικό μοντέλο του συστήματος, με αποτέλεσμα να επηρεάζουν τις συνδέσεις μεταξύ των διαφόρων τμημάτων. Στο χρησιμοποιούμενο μοντέλο υποθέτουμε ότι τα διάφορα components δεν έχουν λάθη και οι μεταξύ τους συνδέσεις μπορεί είτε να έχουν διακοπεί (*opens*) είτε να έχουν βραχυκυκλώσει (*shorts*). Το αντίστοιχο λογικό λάθος είναι ένα σήμα να εμφανίζεται «κολλημένο» σε μια σταθερή τιμή (**stuckatvalue, s-a-v**).

- **Λειτουργικά Σφάλματα**: είναι σφάλματα που σχετίζονται με το λειτουργικό μοντέλο του συστήματος, έτσι πχ μπορεί να φέρουν αλλαγές στον πίνακα αλήθειας μιας πύλης ή ενός τμήματος.

- **Περιοδικά & Παροδικά Σφάλματα**: είναι σφάλματα που δεν εμφανίζονται συνεχώς και χρειάζονται στατιστικά δεδομένα που καθορίζουν την πιθανότητα να εμφανισθούν. Τα σφάλματα αυτά συνήθως τα ανιχνεύουμε με on-linetesting.

Βασική παραδοχή είναι ότι έχουμε ένα μόνο σφάλμα (**single-faultassumption**) στο σύστημα. Η παραδοχή αυτή βασίζεται στην υπόθεση ότι η συχνότητα ανίχνευσης σφαλμάτων είναι τέτοια ώστε η πιθανότητα να συμβούν δύο σφάλματα μεταξύ δύο διαδοχικών ανιχνεύσεων να είναι μικρή.

2.18.2 Ανίχνευση Σφαλμάτων

Έστω ότι Z(x) είναι η λογική συνάρτηση ενός συνδυαστικού κυκλώματος N, με x να είναι ένα διάνυσμα εισόδου. Η παρουσία σφάλματος μετασχηματίζει το κύκλωμα σε ένα νέο κύκλωμα N_f με λογική συνάρτηση $Z_f(x)$. Το κύκλωμα δοκιμάζεται από ένα σύνολο teststt$_1$, t_2, t_3, …, t_m. Ένα διάνυσμα testst λέμε ότι ανιχνεύει ένα λάθος f αν και μόνο αν ισχύει $Z_f(t) \neq Z(t)$.

2.19 Ευαισθητοποίηση

Έστω ότι σε ένα κύκλωμα υπάρχει ένα σφάλμα s-a-v, τότε ορίζουμε δύο τιμές μια χωρίς σφάλμα v και μια με σφάλμα v_f με την μορφή v/v_f. Ένα σφάλμα ανιχνεύεται αν οι τιμές εξόδου στις δύο περιπτώσεις είναι διαφορετικές.

- Ένα testt το οποίο ανιχνεύει το σφάλμα f, **ενεργοποιεί** (*activates*) το f.
- Ένα testt**διαδίδει το σφάλμα** (*propagatestheerror*) σε μια έξοδο, όταν κάνει όλες τις γραμμές κατά μήκος τουλάχιστον ενός μονοπατιού μεταξύ του σημείου του σφάλματος και της εξόδου να έχουν διαφορετικές τιμές v και v_f.

- Μια γραμμή της οποίας η τιμή για ένα testt αλλάζει με την παρουσία σφάλματος f λέγετε **ευαισθητοποιημένη** (*sensitized*) στο σφάλμα f από το testt.
- Ένα μονοπάτι από ευαισθητοποιημένες γραμμές λέγεται **ευαισθητοποιημένο μονοπάτι** (*sensitizedpath*).
- Μια πύλη της οποίας η έξοδος είναι ευαισθητοποιημένη στο σφάλμα f έχει τουλάχιστον μία είσοδό της ευαισθητοποιημένη στο σφάλμα. Έστω G μια πύλη με *inversioni* και *controllingvaluec*, της οποίας η έξοδος είναι ευαισθητοποιημένη στο σφάλμα f, τότε:
- Όλες οι είσοδοι της G που είναι ευαισθητοποιημένες στο f έχουν την ίδια τιμή, έστω α
- Όλες οι είσοδοι της G που δεν είναι ευαισθητοποιημένες στο f έχουν την τιμή $\bar{c}$
- Η έξοδος της G έχει τιμή $a \oplus i$

Η τιμή $\bar{c}$ λέγεται **enablingvalue** γιατί επιτρέπει την διάδοση του λάθους.

2.19.1 Ανιχνευσιμότητα

Ένα σφάλμα f λέγεται **ανιχνεύσιμο** (*detectable*) αν υπάρχει ένα testt το οποίο ανιχνεύει το f, αλλιώς το f λέγεται **μη ανιχνεύσιμο** (*undetectable*). Για ένα μη ανιχνεύσιμο σφάλμα ισχύει $Z_f(x) = Z(x)$ και κανένα test δεν μπορεί ταυτόχρονα να ενεργοποιήσει το f και να δημιουργήσει ένα ευαισθητοποιημένο μονοπάτι προς μια έξοδο.

2.19.2 Πλεονασμός

Ένα συνδυαστικό κύκλωμα που έχει ένα μη ανιχνεύσιμο σφάλμα λέγεται περιττός (**redundant**), αφού μπορεί πάντοτε να απλοποιηθεί με αφαίρεση τουλάχιστον μιας πύλης ή μιας εισόδου μιας πύλης. Αντίθετα ένα συνδυαστικό κύκλωμα στο οποίο όλα τα σφάλματα είναι ανιχνεύσιμα λέγεται απέριττο (*irredundant*).

2.19.3 Ενιαίο Μοντέλο Σφαλμάτων (*SSFs*)

Είναι το πρώτο και πιο διαδεδομένο μοντέλο. Λέγεται επίσης και κλασικό μοντέλο ή standard. Αν και η αξιοπιστία του δεν είναι καθολική εντούτοις είναι πολύ χρήσιμο γιατί μπορεί να αναπαραστήσει πολλά διαφορετικά φυσικά σφάλματα, είναι ανεξάρτητο από την τεχνολογία, tests που ανιχνεύουν SSFs μπορούν να ανιχνεύσουν πολλά μη κλασικά σφάλματα, μπορεί να χρησιμοποιηθεί για να μοντελοποιήσουμε άλλους τύπους σφαλμάτων.

2.20 Δοκιμές για Ενιαία Κολλημένα Ελαττώματα

Η δημιουργία κατάλληλων test (*testgeneration**TG***) είναι σύνθετο πρόβλημα με πολλές παραμέτρους που αλληλεπιδρούν. Οι πιο σημαντικές είναι το κόστος της δημιουργίας του test, η ποιότητα του test που δημιουργήθηκε και το κόστος εφαρμογής του συγκεκριμένου test.

-RandomTG (*RTG*) είναι μια απλή μέθοδος η οποία περιλαμβάνει δημιουργία τυχαίων διανυσμάτων ελέγχου. Για να έχουμε όμως υψηλή ποιότητα χρειαζόμαστε έναν μεγάλο αριθμό από διανύσματα ελέγχου.
-Ντετερμινιστικό (*DeterministicTG***)** είναι μια μέθοδος που δημιουργεί διανύσματα ελέγχου από την επεξεργασία του κυκλώματος. Σε σύγκριση με την RTG είναι πιο ακριβή αλλά παράγει συντομότερα και υψηλότερης ποιότητας tests. Η παραγωγή των διανυσμάτων ελέγχου μπορεί να είναι είτε χειροκίνητη (*manual*) είτε αυτόματη (*automaticATG*). Επίσης μπορεί να στοχεύει στην δημιουργία διανυσμάτων ελέγχου για συγκεκριμένα σφάλματα (*fault-oriented*) είτε να είναι ανεξάρτητη από συγκεκριμένα σφάλματα (*fault-independent*).

2.20.1 Προσανατολισμένα Ντετερμινιστικά Ελαττώματα (*Fault-Oriented ATG)*

Θεωρούμε την περίπτωση που έχουμε ένα κύκλωμα χωρίς fanout. Τα δύο βασικά βήματα στην δημιουργία ενός test για ένα σφάλμα s-a-v είναι πρώτα η ενεργοποίηση του λάθους και μετά η διάδοση του λάθους σε μια πρωτεύουσα έξοδο (primaryoutputPO). Ενεργοποίηση του λάθους σημαίνει ότι θα πρέπει να δώσουμε στις εισόδους τέτοιες τιμές ώστε να προκαλέσουν στην γραμμή l τιμή $\bar{v}$. Αυτό είναι ένα πρόβλημα υπολογισμού των γραμμών (*line-justificationproblem*), ώστε να βρούμεκατάλληλες εισόδους για να έχουμε σαν αποτέλεσμα μια συγκεκριμένη γραμμή στο κύκλωμα να πάρει την επιθυμητή τιμή.

Για να παρακολουθούμε την διάδοση του λάθους είναι απαραίτητο να έχουμε τιμές σύνθετης λογικής (*compositelogicvalues*) της μορφής **v/v$_f$** για το κύκλωμα χωρίς σφάλματα και με σφάλματα αντίστοιχα. Οι σύνθετες τιμές για τις περιπτώσεις λάθους είναι 1/0 και 0/1 και δηλώνονται με τα σύμβολα D και $\bar{D}$.

v/v$_f$	
0/0	0
1/1	1
1/0	D
0/1	$\bar{D}$

AND	0	1	D	$\bar{D}$	x
0	0	0	0	0	0
1	0	1	D	$\bar{D}$	x
D	0	D	D	0	x
$\bar{D}$	0	$\bar{D}$	0	$\bar{D}$	x
x	0	x	x	x	x

OR	0	1	D	$\bar{D}$	x
0	0	1	D	$\bar{D}$	0
1	1	1	1	1	1
D	D	1	D	1	X
$\bar{D}$	$\bar{D}$	1	1	$\bar{D}$	x
x	x	1	x	x	x

Πίνακες τιμών σύνθετης λογικής

	c	i
AND	0	0
OR	1	0
NAND	0	1
NOR	1	1

Έλεγχος τιμών c και αναστροφή i

Η δομή ενός αλγόριθμου για την δημιουργία ενός test για μια γραμμή l τύπου s-a-v περιλαμβάνει την αρχικοποίηση όλων των τιμών σε x και στην συνέχεια τα δύο βασικά βήματα που είναι οι υπορουτίνεςστοίχησης και διάδοσης.

begin
  set all values to x
  Justify(l, v)
  if v = 0 **then***Propagate*(l, D)
  else*Propagate*(l, $\overline{D}$)
end

Ο υπολογισμός μιας γραμμής είναι μια αναδρομική διαδικασία κατά την οποία η τιμή εξόδου μιας πύλης υπολογίζεται από τις τιμές των εισόδων της κλπ, μέχρι να φτάσουμε στις πρωτεύουσες εισόδους (*primaryinputsPI*) του κυκλώματος.Για την διάδοση του λάθους στην πρωτεύουσα έξοδο PO του κυκλώματος πρέπει να ευαισθητοποιήσουμε ένα μοναδικό μονοπάτι από το l στην PO. Κάθε πύλη σ' αυτό το μονοπάτι έχει ακριβώς μια είσοδο ευαισθητοποιημένη στο σφάλμα. Όλες οι άλλες είσοδοι της πύλης G πρέπει να έχουν την μη ελέγξιμη τιμή για την συγκεκριμένη πύλη.

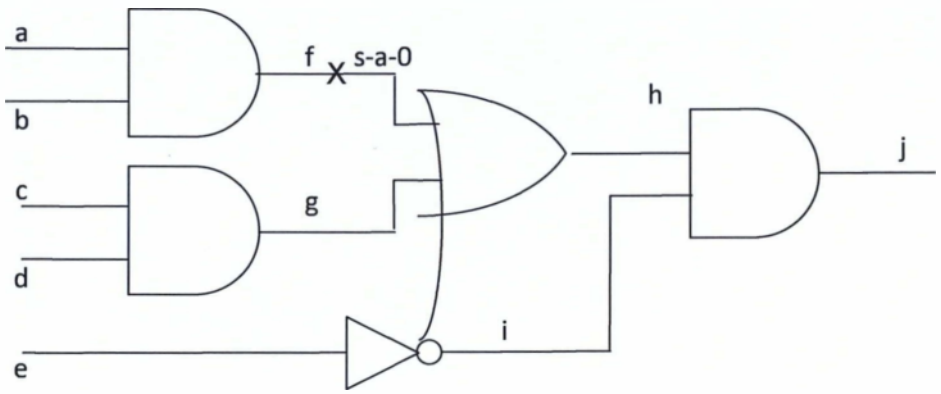
```
Justify (l, val)
begin
  set l to val
  if l is a PI thenreturn
  /* l is a gate (output) */
  c = controlling value of l
  i = inversion of l
  inval = val ⊕ i
  if inval = c̄ then
```

```
Propagate (l, err)
/* err is D or D̄ */
begin
  set l to err
  if l is a PO thenreturn
  k = the fanout of l
  c = controlling value of k
```

Υπορουτίνες στοίχισης και διάδοσης σε κυκλώματα χωρίς έξοδο.

Παράδειγμα 1:

Έστω το κύκλωμα του παρακάτω σχήματος και f ένα σφάλμα s-a-0



Τα βήματα που ακολουθούμε είναι τα εξής:

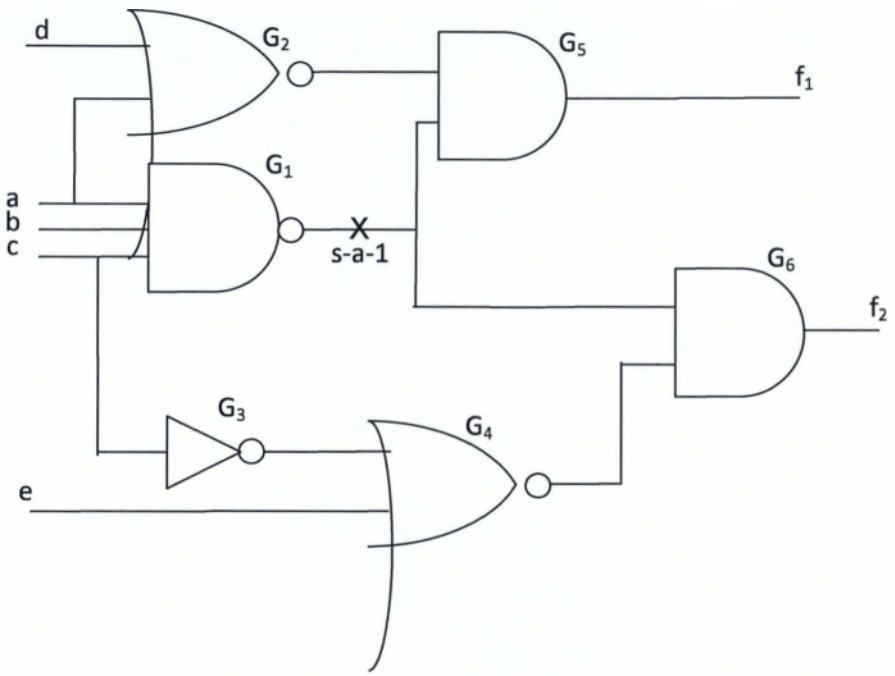
- Στοίχιση (**Justify** (f, 1)). Αυτό σημαίνει ότι a = 1 και b = 1.
- Διάδοση (**Propagate** (f, D)). Αυτό σημαίνει ότι πρέπει το D να διαδοθεί δια μέσου της OR. Από τους πίνακες της σύνθετης λογικής βρίσκουμε ότι πρέπει g = 0. Άρα πρέπει να υπολογίσουμε Justify (g, 0) το οποίο μας οδηγεί σε c = 0 και d = x.

Τελικά πρέπει να έχουμε Propagate (h, D). Αυτό σημαίνει ότι το D οδηγείται στην POj. Για να γίνει αυτό από τους πίνακες της σύνθετης λογικής βρίσκουμε ότι πρέπει να είναι i = 1 και συνεπώς e = 0.
Άρα το ζητούμενο test είναι το 110x0 για τις εισόδους a-b-c-d-e αντίστοιχα.

Στην περίπτωση κυκλώματος με fanout σημαίνει ότι έχουμε πολλούς δρόμους να διαδώσουμε το λάθος σε μια PO. Όταν στο κύκλωμα υπάρχει fanout ή επανασυγκλίνωνfanout τότε οι υπολογισμοί των γραμμών δεν είναι πλέον ανεξάρτητοι μεταξύ τους.

Παράδειγμα 2:

Έστω το κύκλωμα του παρακάτω σχήματος και G_1 ένα σφάλμα s-a-1



-Τα βήματα που ακολουθούμε είναι τα εξής:

- ✓ Justify (G_1, 0). Αυτό σημαίνει ότι a = 1 και b = 1 και c = 1.Λόγω του υπάρχοντος εξόδου έχουμε την δυνατότητα να διαδώσουμε το σφάλμα είτε μέσω της πύλης G_5 είτε μέσω της πύλης G_6. έστω ότι επιλέγουμε την πρώτη περίπτωση.

- ✓ Propagate (G_5, D). Από τους πίνακες της σύνθετης λογικής βρίσκουμε ότι πρέπει να είναι G_2 = 1, δηλαδή a = 0 και d = 0. Επομένως καταλήγουμε σε σύγκρουση (contradiction) αφού πρέπει a = 0 και a = 1.

- ✓ Επομένως η προηγούμενη επιλογή ήταν λάθος και άρα πρέπει να επιλέξουμε την διάδοση μέσω της G_6.

- ✓ Propagate (G_6, D). Από τους πίνακες της σύνθετης λογικής βρίσκουμε ότι πρέπει να είναι G_4 = 1, δηλαδή G_3 = 0 και e = 0. Αυτό μας οδηγεί σε Justify (G_3, 0) δηλαδή c=1.

Άρα το ζητούμενο test είναι το 111x0 για τις εισόδους a-b-c-d-e αντίστοιχα.

2.20.2 Συμπέρασμα

Παρατηρούμε ότι η αναζήτηση μιας λύσης περιλαμβάνει λήψη αποφάσεων, η οποία μπορεί να μας οδηγήσει σε συγκρούσεις. Για τον λόγο αυτό πρέπει να έχουμε μια στρατηγική backtracking ώστε να μπορέσουμε να εξερευνήσουμε όλες τις πιθανές λύσεις. Για να γίνει αυτό θα πρέπει να υπάρχει η δυνατότητα ανάκαμψης από μη σωστές λύσεις, με αποκατάσταση των τιμών πριν την λάθος κατάσταση. Συνήθως γίνονται αναθέσεις τιμών, οι οποίες προκύπτουν από αποφάσεις που οδηγούν σε μοναδικές τιμές. Η διαδικασία αυτή είναι γνωστή ως επίπτωση (*implication)*. Στους περισσότερους αλγόριθμους που υποστηρίζουν backtracking πρέπει να καταγράφονται οι τιμές ώστε να έχουμε την δυνατότητα της διαγραφής σε περίπτωση σύγκρουσης.

2.21 Σύνορα D *(The D-Frontier)*

Αποτελείται από όλες τις πύλες που η τρέχουσα τιμή εξόδου είναι x αλλά έχουν σε μια ή περισσότερες εισόδους σήμα D ή $\overline{D}$. Η διάδοση του σφάλματος γίνεται επιλέγοντας μια από τις πύλες του D-frontier και θέτοντας τιμές στις εισόδους της πύλης ώστε το σφάλμα να περάσει στην έξοδό της.

2.22 Σύνορα J *(The J-Frontier)*

Είναι ένα σύνολο από όλες τις πύλες των οποίων είναι γνωστή η έξοδος αλλά δεν υποδηλώνεται από τις εισόδους τους. Δηλαδή πρόκειται για άλυτα μέχρι στιγμής προβλήματα υπολογισμού γραμμών.

2.23 ΑλγόριθμοςD*(TheD-Algorithm)*

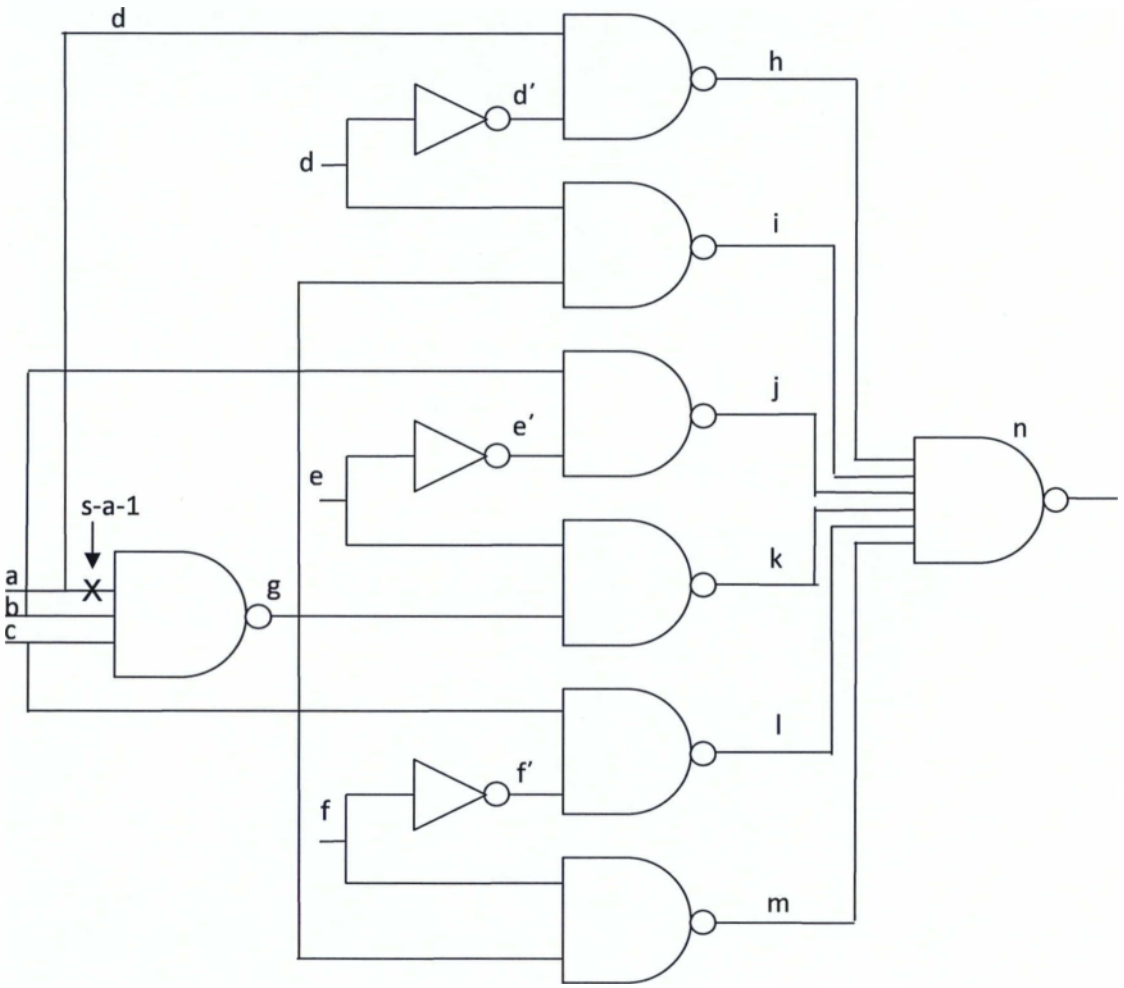
Είναι ένας αλγόριθμος ευαισθητοποίησης μονοπατιού. Στον αλγόριθμο αυτό γίνεται η παραδοχή ότι η διάδοση του λάθους έχει προτεραιότητα σε σχέση με τον υπολογισμό. Όλες οι αναθέσεις γίνονται από την υπορουτίνα Imply_and_check. Ένα από τα χαρακτηριστικά του D-algorithm είναι η δυνατότητά του να διαδίδει το σφάλμα σε επανασυγκλίνοντα εξόδους. Αυτό είναι γνωστό ως πολλαπλά ευαισθητοποιημένα μονοπάτια (multiple-pathsensitization).

- **D-Algorithm**

```
D-alg()
begin
  if Imply_and_check( ) = FAILURE then return FAILURE
  if (error not at PO) then
    begin
      if D-frontier = ∅ then return FAILURE
      repeat
        begin
          select an untried gate (G) from D-frontier
          c = controlling value of G
          assign c̄ to every input of G with value x
          if D-alg( ) = SUCCESS then return SUCCESS
        end
      until all gates from D-frontier have been tried
      return FAILURE
    end
  /* error propagate at PO */
  if J-frontier = ∅ then return SUCCESS
```

-Εφαρμογή αλγορίθμου D

Παράδειγμα 1
Έστω το κύκλωμα και το σφάλμα του παρακάτω σχήματος.



Η εφαρμογή του **D-algorithm** έχει τα εξής βήματα:

Αρχικά πρέπει να ενεργοποιηθεί τα σφάλμα. Αυτό οδηγεί σε a = 0 και h = 1.
Το D-frontier έχει μια και μοναδική διαδρομή {g}. Άρα θα πρέπει το σφάλμα να διαδοθεί μέσω της g που σημαίνει g = D. Από τον πίνακα με τις controllingvalues για την πύλη NAND είναι 0 και σύμφωνα με τον αλγόριθμο πρέπει να θέσουμε σε κάθε είσοδο της g την τιμή $\bar{c}$. Επομένως b = 1 και c = 1.

Το D-frontier είναι {i, k, m}.
Έστω ότι επιλέγουμε την διάδοση μέσω της πύλης i. Από τους πίνακες σύνθετης λογικής όταν η είσοδος της NAND είναι D έχουμε: i = $\overline{D}$ και d = 1 και d΄ = 0.

- ToD-frontiereίναι {k, m, n}.
Για διάδοση μέσω της πύλης n ώστε το σφάλμα να βγει στην έξοδο έχουμε:
n = D και j = 1, k = 1, l = 1, m = 1.
j = 1 σημαίνει e΄ = 0 και e = 1.
e = 1 σημαίνει k = $\overline{D}$ **σύγκρουση-αποτυχία**επομένως κάνουμεοπισθοδρόμηση (*backtracking*)

- ToD-frontiereίναι {k, m, n}.
Για διάδοση μέσω της πύλης k έχουμε:
k = $\overline{D}$ και e = 1 και e΄ = 0 και j = 1.

- ToD-frontiereίναι {m, n}.
Για διάδοση μέσω της πύλης n ώστε το σφάλμα να βγει στην έξοδο έχουμε:
n = D και l = 1 και m = 1.
l = 1 σημαίνει f΄ = 0 και f = 1.
f = 1 σημαίνει m = $\overline{D}$ **σύγκρουση-αποτυχία**επομένως κάνουμεοπισθοδρόμηση (*backtracking*)

- ToD-frontiereίναι {m, n}.
Για διάδοση μέσω της πύλης m έχουμε:
m = $\overline{D}$ και f = 1 και f΄ = 0 και l = 1.
l = 1 σημαίνει f΄ = 0 και f = 1.

- ToD-frontiereίναι {n}.
Το σφάλμα διαδίδεται μέσω της πύλης n και έχουμε n = D **επιτυχία**.

- Άρα το ζητούμενο test είναι 011111 για τις εισόδους a-b-c-d-e-f αντίστοιχα.

2.24 Λειτουργικός Έλεγχος (*FunctionalTesting*)

Ο λειτουργικός έλεγχος (**functionaltesting**) στηρίζεται στο λειτουργικό μοντέλο (**functionalmodel**) του κυκλώματος και σε ένα μεγάλο ποσοστό είναι ανεξάρτητο από την υλοποίησή του. Μπορεί να χρησιμοποιηθεί όχι μόνο για τον έλεγχο φυσικών σφαλμάτων αλλά και για έλεγχο σχεδιαστικών λαθών κατά την υλοποίηση. Στο σημείο αυτό πρέπει να σημειώσουμε ότι τα tests που προκύπτουν από το μοντέλο δομής δεν μπορούν να εξασφαλίσουν ότι η επιθυμητή λειτουργία έχει υλοποιηθεί σωστά.

Στόχος του λειτουργικού ελέγχου (*functionaltesting*) είναι ελέγξει την σωστή λειτουργία ενός κυκλώματος σε σχέση με τα λειτουργικά του χαρακτηριστικά. Αυτό μπορεί να γίνει με τους εξής τρόπους/προσεγγίσεις:

- Υποθέτουμε ότι υπάρχουν συγκεκριμένων μοντέλων λειτουργικών σφαλμάτων και προσπαθούμε να δημιουργήσουμε tests για την ανίχνευση των λαθών αυτών.
- Δεν ενδιαφερόμαστε για συγκεκριμένους τύπους λαθών και προσπαθούμε να δημιουργήσουμε tests βασιζόμενοι στην συμπεριφορά του κυκλώματος χωρίς σφάλματα. Έτσι πχ ένα functionaltest για ένα flip-flop περιλαμβάνει τον έλεγχο αν μπορεί να γίνει set και reset, και τον έλεγχο αν μπορεί το flip-flop να κρατήσει την κατάστασή του.
- Υποθέτουμε ότι οποιοδήποτε σφάλμα είναι πιθανό να συμβεί και προσπαθούμε να δημιουργήσουμε tests. Η προσέγγιση αυτή λέγεται exhaustive. Λόγω του μήκους τέτοιων tests, η εφαρμογή τους γίνεται μόνο σε μικρά κυκλώματα.

Γενικότερα ο λειτουργικός έλεγχος (*functionaltesting*) προσπαθεί να προσεγγίσει την δημιουργία ελέγχου (*test*) σε υψηλότερο επίπεδο, χωρίς όμως να έχει φτάσει σε ωριμότητα και επιτυχία των διαρθρωτικών δοκιμών (*structuraltesting*). Οι δυσκολίες είναι αρκετές τόσο από την χρήση των δυαδικών διαγραμμάτων απόφασης (*binarydecisiondiagrams*), των οποίων η χρησιμότητα δεν έχει αποδειχθεί πλήρως, όσο και από την χρήση των ψευδοεξαντλητικών δοκιμών. Για τα τελευταία εφαρμόζονται τεχνικές στεγανοποίησης (*partitioning*) ώστε το εξεταζόμενο κύκλωμα να είναι μικρότερο για να έχουμε μικρότερους χρόνους και μικρότερο αριθμό εξεταζόμενων περιπτώσεων. Οι τεχνικές στεγανοποίησης (*partitioning*) όμως απαιτούν γνώση της εσωτερικής δομής του κυκλώματος, η οποία δεν είναι πάντοτε γνωστή και επιπλέον δεν υπάρχουν καλοί αλγόριθμοι για στεγανοποίηση (*partitioning*).

2.25 Αξιοπιστία Κυκλωμάτων

Η αξιοπιστία είναι μια παράμετρος που σχετίζεται με την καλή λειτουργία ενός κυκλώματος και η οποία συνήθως παραμερίζεται όπως π.χ:

- Τον ορισμό της αξιοπιστίας ενός συστήματος.
-
- Τις παραμέτρους: ρυθμός βλαβών και μέσος χρόνος μεταξύ βλαβών.
- Να υπολογίζουμε την αξιοπιστία ενός συνθέτου συστήματος.

Η αξιοπιστία ενός κυκλώματος ή συστήματος σχετίζεται με την εύρυθμη λειτουργία του για κάποιο χρονικό διάστημα. Ακριβέστερα, σαν αξιοπιστία (Reliability) ενός στοιχείου ορίζεται η πιθανότητα καλής λειτουργίας του για ένα καθορισμένο χρονικό διάστημα.

-Ομοίως, σαν αξιοπιστία ενός συστήματος ορίζεται η πιθανότητα καλής λειτουργίας όλου του συστήματος για ορισμένο χρονικό διάστημα.

Ένα στοιχείο, π.χ. μια πύλη, μπορεί να πάθει βλάβη ανά πάσα στιγμή. Ο ρυθμός των βλαβών ενός στοιχείου, παριστάνεται με το λ, είναι χαρακτηριστικός του στοιχείου και δηλώνει το πλήθος των στοιχείων αυτού του είδους που παθαίνουν βλάβη ανά μονάδα χρόνου. Συνήθως, ο συντελεστής λ παραμένει σταθερός στη διάρκεια της "χρησίμου ζωής" του στοιχείου. Στην περίπτωση αυτή η αξιοπιστία του στοιχείου για ένα χρονικό διάστημα t, όπου ο χρόνος t εμπίπτει στο διάστημα της "χρησίμου ζωής" του, δίνεται από τη σχέση:

$$R(t) = e^{-\lambda t}$$

Εναλλακτικά, η αξιοπιστία του στοιχείου μπορεί να δοθεί από τη σχέση:

$$R(t) = e^{-tm}$$

όπου το m = 1/λ και ονομάζεται **μέσος χρόνος μεταξύ βλαβών** (*MeanTimeBetweenFailure* - **MTBF**).

Η αξιοπιστία ενός συστήματος που αποτελείται από n στοιχεία, με σταθερό λi i=1...n, και το οποίο για να εργάζεται ικανοποιητικά πρέπει να εργάζονται σωστά και τα n στοιχεία του δίνεται, στην απλή περίπτωση που θεωρούμε τα στοιχεία ανεξάρτητα μεταξύ τους, από το γινόμενο των επιμέρους αξιοπιστιών των στοιχείων:

$$R_{συστ} = R_1 . R_2 . \ldots R_n$$

και ο ρυθμός βλαβών $\lambda_{συστ}$ του συστήματος προκύπτει:

$$\lambda_{συστ} = \lambda_1 + \lambda_2 + \ldots + \lambda_n$$

Η αξιοπιστία ενός στοιχείου δεν επηρεάζεται από την παρουσία άλλων στοιχείων. Σε **VLSI** κυκλώματα, όπως π.χ. οι μνήμες των υπολογιστών, η αξιοπιστία ενός κυττάρου μνήμης επηρεάζεται από τα γειτονικά του κύτταρα και όχι μόνο. Σε τέτοιες περιπτώσεις η Rσυστ υπολογίζεται λαμβάνοντας υπόψη τη σύνθετη πιθανότητα (compoundprobability) λειτουργίας των στοιχείων.Για τη δημιουργία αξιόπιστων συστημάτων από μη αξιόπιστα συστατικάέχουν αναπτυχθεί ειδικές μέθοδοι σχεδίασης ψηφιακών συστημάτων. Οιμέθοδοι αυτοί, που είναι ανάλογοι με τη σχεδίαση κωδίκων πληροφορίας ανεκτικών στο θόρυβο. Βασίζονται στην προσθήκη πλεοναζόντων στοιχείων στο κύκλωμα. Τα κατασκευαζόμενα συστήματα ονομάζονται ανεκτικά στις βλάβες (faulttolerant) και έχουν αξιοπιστία πολύ μεγαλύτερη απ' ότι τα αντίστοιχα απλά συστήματα, παρ' όλο που η ποιότητα των βασικών τους συστατικών, δηλ. το λ του κάθε στοιχείου, παραμένει η ίδια.

Κεφάλαιο 3°

3.1 Εισαγωγή

Στα σημερινά κυκλώματα η πολυπλοκότητα των tests είναι πολύ μεγάλη και η διαδικασία χρονοβόρα. Έτσι διάφορες μελέτες έδειξαν ότι μπορεί να δημιουργηθεί μια σχέση κόστους, η οποία να σχετίζεται με την διαδικασία του testing. Υπάρχουν πολλές παράμετροι που επηρεάζουν το κόστος, όπως: το κόστος της δημιουργίας των διανυσμάτων ελέγχου, το κόστος της προσομοίωσης των σφαλμάτων, το κόστος των εξαρτημάτων ελέγχου, και το κόστος της ίδιας της διαδικασίας ελέγχου δηλαδή του χρόνου που χρειάζεται για τον εντοπισμό των σφαλμάτων. Πολλές φορές το κόστος αυτό είναι μεγαλύτερο από κόστος σχεδίασης. Για να κρατηθεί το κόστος σε λογικά επίπεδα εφαρμόζονται τεχνικές σχεδίασης που επιτρέπουν τον εύκολο έλεγχο του κυκλώματος (**designfortestability**) και μειώνουν τα κόστος από τις παραπάνω παραμέτρους. Για παράδειγμα τεχνικές scandesign μπορούν να μειώσουν το κόστος της δημιουργίας tests ελέγχου αλλά να αυξήσουν τον αριθμό των pins (*καρφιτσών*) εισόδου/εξόδου την επιφάνεια του κυκλώματος και τον χρόνο δοκιμών.

Ο όρος ελεγξιμότητα (**testability)** είναι ένα χαρακτηριστικό σχεδίασης που επηρεάζει το κόστος του ελέγχου (*testing*). Συνήθως επιτρέπει να καθορίσουμε την κατάσταση της συσκευής (**κανονική, εκτός λειτουργίας, υποβαθμισμένη**) και να έχουμε σε σύντομο χρόνο την απομόνωση των σφαλμάτων.

Δύο σημαντικοί παράγοντες που επηρεάζουν την ελεγξιμότητα (**testability**) είναι η δυνατότητα ελέγχου (**Controllability**) και η παρατηρησιμότητα (**Observability**).

- Η δυνατότητα ελέγχου (**Controllability**), είναι η δυνατότητα να καθορίζουμε συγκεκριμένες τιμές σήματος σε κάθε κόμβο του κυκλώματος θέτοντας τιμές στις εισόδους του.

- Η παρατηρησιμότητα (**Observability),** είναι η δυνατότητα να μπορούμε να καθορίσουμε την τιμή σε οποιοδήποτε κόμβο του κυκλώματος, ελέγχοντας τις τιμές των εισόδων και παρατηρώντας τις τιμές των εξόδων.

3.2 Ελεγξιμότητα

Ελεγξιμότητα είναι ένα χαρακτηριστικό που επηρεάζει το σχεδιασμό των διαφόρων δαπανών που συνδέονται με τις δοκιμές. Συνήθως επιτρέπει (1) την κατάσταση (κανονική, ακατάλληλη για χρήση, υποβαθμισμένη) μιας συσκευής που θα καθοριστεί και η απομόνωση των βλαβών εντός της συσκευής που πρέπει να εκτελεστούν γρήγορα, για να μειώσει τόσο τη διάρκεια της δοκιμής και του κόστους, και (2) το κόστος -την αποτελεσματική ανάπτυξη των αναλύσεων για τον προσδιορισμό αυτής της κατάστασης. Το σχέδιο για τις τεχνικές δυνατότητες δοκιμής είναι οι προσπάθειες σχεδιασμού που υιοθετούνται για να εξασφαλιστεί ότι μια συσκευή είναι ελέγξιμη.

Κυκλώματα συνήθως δύσκολο να ελεγχθούν είναι οι αποκωδικοποιητές, κυκλώματα με ανάδραση, ταλαντωτές, και γεννήτριες ρολογιού. Ένα άλλο παράδειγμα είναι ένα 16-bit μετρητής του οποίου οι παράλληλες είσοδοι είναι γειωμένες. Στην περίπτωση αυτή απαιτούνται 12^{15} παλμοί ρολογιού ώστε το πιο σημαντικό (significant) bit να λάβει την τιμή 1.

Ένα κύκλωμα έχει συχνά κακή παρατηρησιμότητα, εάν απαιτεί ένα μοναδικό πρότυπο εισόδου ή μια μακρά σύνθετη ακολουθία των προτύπων εισόδου για τη διάδοση της κατάστασης ενός ή περισσότερων κόμβων στις εξόδους του κυκλώματος. Τα λιγότερα ευαίσθητα κυκλώματα περιλαμβάνουν ακολουθιακά κυκλώματα, τα κυκλώματα με ανάδραση, ενσωματωμένες μνήμες **RAMs**, **ROMs**, ή **PLAs**, και τα κυκλώματα με περιττούς κόμβους.

Ο αντίκτυπος της πρόσβασης στις δοκιμές οδηγεί στις ακόλουθες γενικές παρατηρήσεις:

- Η ακολουθιακή λογική είναι πολύ πιο δύσκολο να δοκιμαστεί από τη συνδυαστική λογική.
- Η λογική ελέγχου είναι πιο δύσκολο να δοκιμαστεί για τον έλεγχο των δεδομένων - λογική πορεία.
- Η τυχαία λογική είναι πιο δύσκολο να δοκιμαστεί από τη δομημένη.

3.3 Σχεδίαση για Ελεγξιμότητα (*DFT*)

Οι περισσότερες τεχνικές **DFT** εφαρμόζουν ανασύνθεση της υπάρχουσας σχεδίασης ή προσθήκη επιπλέον υλικού (*hardware*) στην σχεδίαση. Οι περισσότερες προσεγγίσεις απαιτούν αλλαγές στο κύκλωμα και επηρεάζουν την επιφάνεια του κυκλώματος, τον αριθμό των I/Opins και την καθυστέρηση του κυκλώματος. Γενικά πρέπει να βρεθεί η χρυσή τομή μεταξύ της ποσότητας DFT που πρέπει να χρησιμοποιήσουμε και του κέρδους που θα έχουμε εφαρμόζοντας αυτή την τεχνική ώστε να έχουμε μείωση του κόστους, αύξηση της ποιότητας των tests και μείωση του επιπέδου ελαττώματος (*defectlevel*). Επίσης επηρεάζονται το μήκος των tests, το απαιτούμενο ποσό μνήμης στο μηχάνημα ελέγχου και ο χρόνος εφαρμογής των tests. Για παράδειγμα αύξηση στην επιφάνεια του κυκλώματος και της πολυπλοκότητας της λογικής του, έχει σαν αποτέλεσμα την αύξηση της κατανάλωσης και την μείωση της απόδοσης. Όμως μείωση του απόδοσης σημαίνει αύξηση στον αριθμό των παραγόμενων ελαττωματικών chips. Αν ένα ελαττωματικό chip τοποθετηθεί σε μια πλακέτα τότε και αυτή θα είναι ελαττωματική. Το κόστος για να ανίχνευση μιας ελαττωματικής πλακέτας είναι 10 με 20 φορές μεγαλύτερο από το κόστος σε επίπεδο chip. Αν μια ελαττωματική πλακέτα χρησιμοποιηθεί σε ένα σύστημα τότε το κόστος για τον έλεγχο και τον εντοπισμό της σε επίπεδο συστήματος είναι 10 με 20 φορές μεγαλύτερο από το κόστος σε επίπεδο πλακέτας.

3.3.1 Κανόνες για DFT

- **Δημιουργούμε σημεία δοκιμών (testpoints)για να αυξήσουμε την δυνατότητα για τη δυνατότητα ελέγχου (controllability) και τη παρατηρησιμότητα (*observability*).** Υπάρχουν δύο τύποι τέτοιων σημείων. Τα σημεία ελέγχου (*controlpointsCP*) και τα σημεία παρατήρησης (*observationpointsOP*). Τα πρώτα είναι πρωτεύοντες είσοδοι ενώ τα δεύτερα είναι πρωτεύοντες έξοδοι στο κύκλωμα.

- **Σχεδιάζουμε το κύκλωμα ώστε να μπορεί εύκολα να γίνει αρχικοποίηση (*initialization*).**Η αρχικοποίηση είναι μια διαδικασία ώστε ένα ακολουθιακό κύκλωμα να έρθει σε μια γνωστή κατάσταση, σε πεπερασμένο γνωστό χρόνο. πχ κατά την εκκίνηση του κυκλώματος.

- **Απενεργοποιούμε εσωτερικά one-shots κατά την διάρκεια των tests.** Οι μονοσταθείςπολυδονητές (*monostablemultivibrators ή one-shots*) δημιουργούν παλμούς εσωτερικά στο κύκλωμα και κάνουν δύσκολο τον εξωτερικό έλεγχο, αφού τα μηχανήματα ελέγχου χάνουν τον συγχρονισμό τους με το κύκλωμα.

- **Απενεργοποιούμε εσωτερικούς ταλαντωτές** (*oscillators*) **και ρολόγια** (*clocks*)Για τους ίδιους λόγους που αναφέρθηκαν στην προηγούμενη περίπτωση.

- **Χωρίζουμε μεγάλους μετρητές** (*counters*) **και τους καταχωρητές ολίσθησης** (*shiftregisters*) **σε μικρότερες μονάδες.** Οι μετρητές (**counters**) και σε μικρότερο βαθμό οι καταχωρητές ολίσθησης (**shiftregisters**) είναι μονάδες δύσκολες στον έλεγχο γιατί οι ακολουθίες ελέγχου απαιτούν πολλούς κύκλους ρολογιού. Έτσι για να αυξηθεί η δυνατότητα ελέγχου πρέπει να χωρισθούν σε μικρότερες συσκευές ώστε η σειριακή είσοδος και το ρολόι τους να ελέγχονται εύκολα.

- **Χωρίζουμε μεγάλα κυκλώματα σε μικρότερα υποκυκλώματα ώστε να μειωθεί το κόστος δημιουργίας tests.**

- **Αποφεύγουμε την χρήση πλεονάζουσας λογικής (redundantlogic)** Αν και μερικές φορές προσθέτουμε πλεονάζουσα λογική (**redundancy**) σε ένα κύκλωμα, αυτό πρέπει να γενικότερα να αποφεύγεται διότι ένα σφάλμα πλεονάζουσας λογικής μπορεί να απαξιώσει tests για άλλα σφάλματα, και δημιουργείται πρόβλημα στον υπολογισμό της κάλυψης σφάλματος (**faultcoverage**) με αποτέλεσμα να αυξάνεται ο χρόνος για την δημιουργία των tests.

3.4 Δομημένη Προσέγγιση

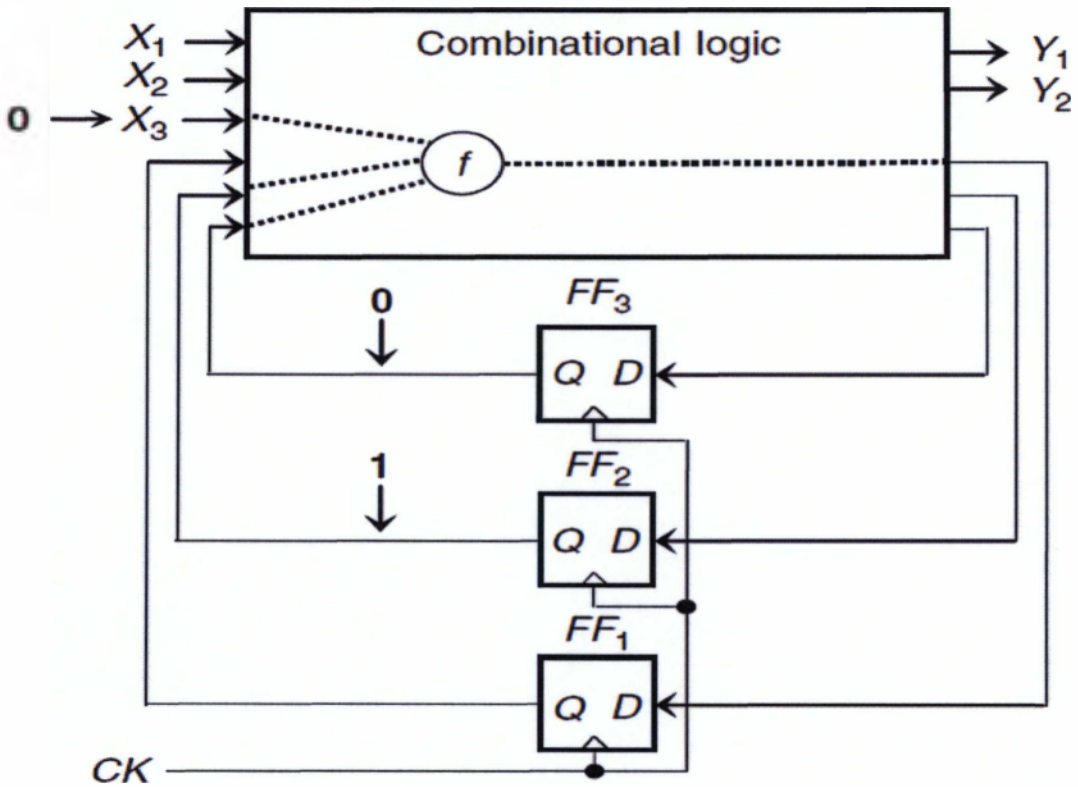
Η δομημένη προσέγγιση **DFT** προσπαθεί να βελτιώσει τη συνολική ελεγξιμότητα ενός κυκλώματος με μια εξέταση με γνώμονα τη μεθοδολογία σχεδιασμού [*Williams 1983*], [*McCluskey 1986*]. Αυτή η προσέγγιση είναι μεθοδική και συστηματική, με πολύ πιο προβλέψιμα αποτελέσματα.

Το παρακάτω ακολουθιακό κύκλωμα, που ακολουθεί, περιέχει συνδυαστική λογική και τρεις D flip-flops. Ας υποθέσουμε ότι ένα κολλημένο, στο σφάλμα, στην συνδυαστική λογική απαιτεί την εισαγωγή πρωτογενούς X3, flip-flop FF2, και flip-flop FF3 να εισαχθεί στο 0, 1 και 0, αντίστοιχα, για να συλλάβει το αποτέλεσμα της βλάβη σε FF1.

Επειδή οι τιμές που είναι αποθηκευμένες σε FF2 και FF3 δεν είναι άμεσα ελεγχόμενες από την αρχική είσοδο, μια μακρά ακολουθία των εργασιών μπορεί να χρειαστούν για τα FF2 και FF3 με τις απαιτούμενες τιμές.

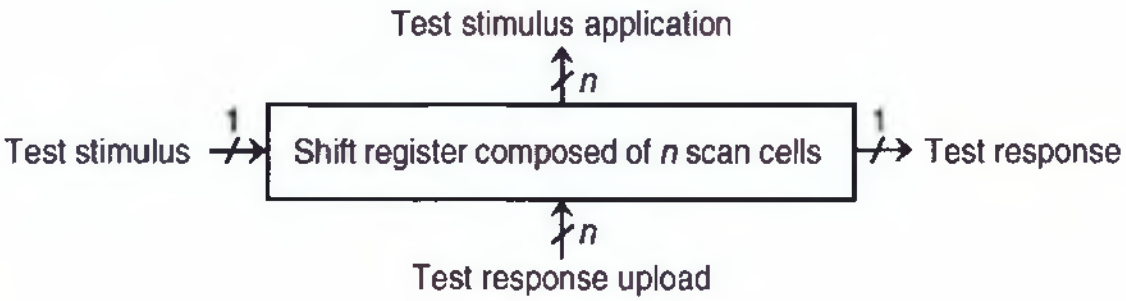
Επιπλέον, προκειμένου να παρατηρήσουν το αποτέλεσμα, σφάλμα στα ήδη υπάρχοντα flip-flop FF1, μπορεί να χρειαστεί ένας μακρύς έλεγχος για την αξιοπιστία του FF1 στο πρωτεύον σημείο εξόδου. Από αυτό το παράδειγμα, μπορεί να διαπιστωθεί ότι η κύρια δυσκολία κατά την εξέταση ενός διαδοχικού κυκλώματος πηγάζει από το γεγονός ότι είναι δύσκολο να ελεγχθεί και να τηρεί την εσωτερική κατάσταση του κυκλώματος.

Σχήμα 3.4:



Ο σχεδιασμός σάρωσης, ο σχεδιασμός του οποίου απεικονίζεται στο σχήμα που ακολουθεί, επιχειρεί να διευκολύνει την επιλογή αυτή δυσκολία με την παροχή εξωτερικής πρόσβασης σε επιλεγμένα στοιχεία αποθήκευσης σε ένα σχέδιο. Αυτό επιτυγχάνεται μετατρέποντας αρχικά επιλεγμένα στοιχεία αποθήκευσης στο σχεδιασμό σάρωσης κυττάρων και στη συνέχεια ραφής τους μαζί για να σχηματίσουν ένα ή περισσότερα μητρώα στροφή, που ονομάζεται σάρωση αλυσίδων.

Η σάρωση του σχεδιασμού φαίνεται στο παρακάτω σχήμα, είναι τα στοιχεία αποθήκευσης η μετατόπισης λειτουργίας. Κάθε ερέθισμα δοκιμών και απόκριση μπορεί τώρα να μετατοπιστεί από και προς τα κύτταρα σάρωσης n σε n κύκλους ρολογιού, αντίστοιχα...



...χωρίς να χρειάζεται να καταφύγουν σε εφαρμογή ενός εκθετικού αριθμού κύκλων ρολογιού για να αναγκάσει όλα τα στοιχεία αποθήκευσης στην επιθυμητή εσωτερική κατάσταση. Επειδή ο σχεδιασμός της σάρωσης παρέχει πρόσβαση σε εσωτερικά στοιχεία αποθήκευσης, και η παραγωγή δοκιμής της πολυπλοκότητας μειώνεται.

3.5 Αρχιτεκτονικές Σάρωσης

Υπάρχουν τρεις δημοφιλείς αρχιτεκτονικές σάρωσης:

- Πλήρους σάρωσης, όπου όλα τα στοιχεία αποθήκευσης μπορούν να μετατραπούν σε σάρωση κυττάρων και των συνδυαστικών **ATPG**, που χρησιμοποιούνται για την παραγωγή δοκιμής

- Μερικής-σάρωσης, όπου ένα υποσύνολο των στοιχείων αποθήκευσης μετατρέπεται σε σάρωση κυττάρων και διαδοχικών ATPG όπου συνήθως χρησιμοποιείται για την παραγωγή δοκιμής

- Τυχαίας προσπέλασης, όπου χρησιμοποιείται τυχαίος μηχανισμός αντιμετώπισης, αντί των σειριακών αλυσίδων σάρωσης, για να παρέχει άμεση πρόσβαση στην ανάγνωση ή εγγραφή σε οποιοδήποτε κελί.

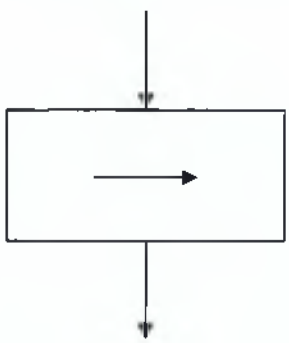
3.5.1 Σάρωση Σχεδιαστικών Κυψελών

Ένα κελί σάρωσης έχει δύο διαφορετικές τιμές εισόδου που μπορούν να επιλεγούν. Η πρώτη είσοδος, εισαγωγή δεδομένων, οδηγείται από την συνδυαστική λογική του κυκλώματος, ενώ η δεύτερη, η σάρωση, οδηγείται από την έξοδο ενός άλλου κελιού σάρωσης, προκειμένου να σχηματίσουν ένα ή περισσότερους καταχωρητές ολίσθησης, που ονομάζεται και σάρωση αλυσίδων.Αυτές οι αλυσίδες σάρωσης γίνονται εξωτερικά προσβάσιμα με τη σύνδεση της πρώτης σάρωσης εισόδου του πρώτου κελιού , και τη σάρωση εισροών και των εκροών εισόδου/εξόδου του τελευταίου κελιού στο σημείο εξόδου. Αυτό, διότι υπάρχουν δύο πηγές εισόδου σε ένα κελί σάρωσης, ένας μηχανισμός επιλογής όπου πρέπει να καταστεί δυνατή η σάρωση ώστε να λειτουργεί σε δύο διαφορετικούς τρόπους: κανονική/σύλληψη, λειτουργίας και τον τρόπο αλλαγής.

3.5.2 Καταχωρητές Σάρωσης

Με την χρήση σημείων ελέγχου αυξάνεται η δυνατότητα ελέγχου (***controllability***) και η παρατηρησιμότητα (***observability***) ενός κυκλώματος. Αυτό όμως έχει επιπτώσεις στον αριθμό των εισόδων/εξόδων, I/Opins. Ένας άλλος τρόπος για να αυξήσουμε την δυνατότητα αυτή είναι με την χρήση ενός ειδικού καταχωρητή που λέγεται καταχωρητής σάρωσης (*scanregister*(**SR**)). Ο καταχωρητής αυτός έχει δυνατότητα τόσο παράλληλης όσο και σειριακής φόρτωσης και τα κελιά του χρησιμοποιούνται σαν σημεία παρατήρησης ή έλεγχου ή και τα δύο. Η χρήση καταχωρητή σάρωσης αποτελεί συμβιβασμό μεταξύ του χρόνου ελέγχου, του **overhead** από την αύξηση της επιφάνειας του κυκλώματος και του αριθμού των ακροδεκτών I/O.

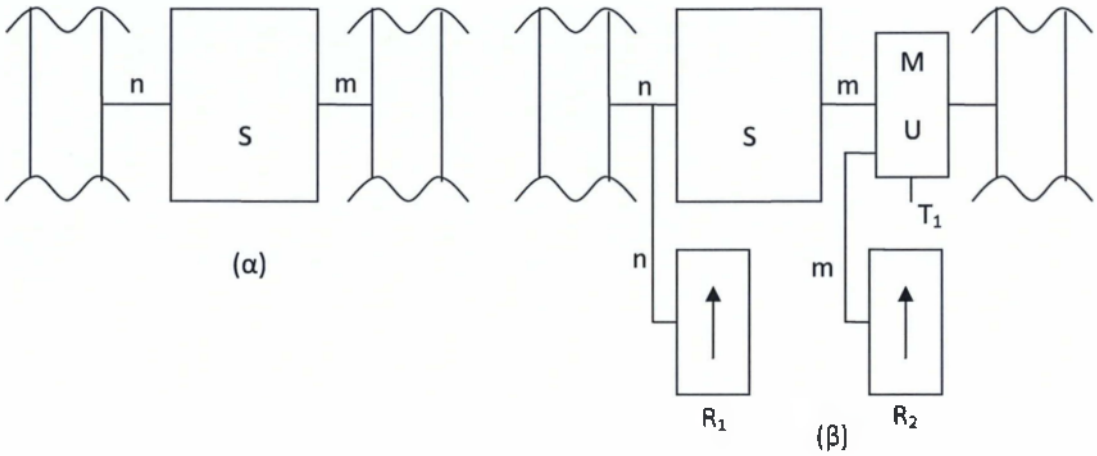
Το σύμβολο καταχωρητή σάρωσης (scanregister) φαίνεται στο παρακάτω σχήμα:



Σχήμα 3.5.2 (α): Ανίχνευση Ορίου

Είναι χρήσιμο σε πολύπλοκα chips ή πολύπλοκες πλακέτες να έχουμε την δυνατότητα να απομονώσουμε ένα τμήμα του κυκλώματος από τα υπόλοιπα.

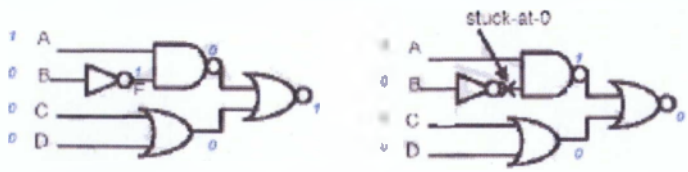
Σχήμα 3.5.2 (β):



Αυτό μπορεί να γίνει με την χρήση του boundaryscan η υλοποίηση του οποίου φαίνεται στο παραπάνω σχήμα. Το αρχικό κύκλωμα (α) έχει n εισόδους και m εξόδους. Στο τροποποιημένο κύκλωμα (β) χρησιμοποιήθηκαν δύο scanregistersR_1 και R_2. Ο R_1 μπορεί να χρησιμοποιηθεί για να παρατηρήσουμε τις τιμές των εισόδων του κυκλώματος, ενώ ο R_2 για να οδηγήσουμε τις εξόδους του κυκλώματος.

3.6 Ικανότητα Ανίχνευσης Σφαλμάτων

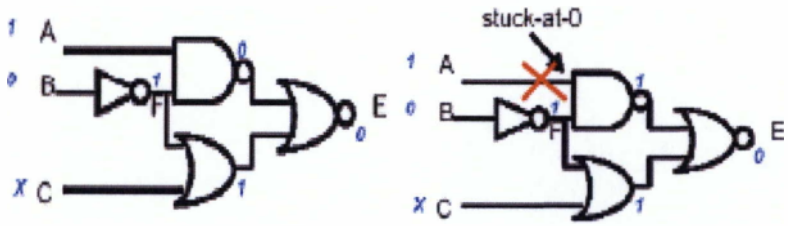
- Ελεγξιμότητα (***Controllability***): η ικανότητα να τεθεί ένας κόμβος σε μια δεδομένη κατάσταση (0 ή 1) με κατάλληλες τιμές στις εισόδους του ΟΚ.
- Παρατηρησιμότητα (***Observability***): η ικανότητα να διαπιστωθεί η τιμή ενός κόμβου από τις αποκρίσεις στις εξόδους του ΟΚ.



Ελεύθερο από σφάλματα κύκλωμαΎπαρξη σφάλματος μόνιμης τιμής

Η διαδικασία της ανίχνευσης σφαλμάτων μπορεί να ιδωθεί ως η ενέργεια εκείνη με την οποία θα ενεργοποιήσουμε (*sensitise*) ένα σφάλμα από το εξωτερικό περιβάλλον (*κύριες είσοδοι*), ώστε αυτό να οδηγήσει στην εμφάνιση μιας λανθασμένης τιμής σε ένα κόμβο του κυκλώματος, και εν συνέχεια να διαδώσουμε (*propagate*) την λανθασμένη τιμή σε κάποια κύρια έξοδο για να την παρατηρήσουμε από το εξωτερικό περιβάλλον.

Η ύπαρξη σημείων επανασύγκλισης σημάτων (*reconvergentfan-outpoints*) δυσκολεύει σημαντικά την ανίχνευση σφαλμάτων. Το γεγονός αυτό οφείλεται στο ότι δεν μπορούμε να θέσουμε σε δύο κόμβους τις επιθυμητές τιμές ανεξάρτητα μεταξύ τους. Στο παράδειγμα που ακολουθεί δεν υπάρχει τιμή για τον κόμβο Β που να επιτρέπει ταυτόχρονα την ενεργοποίηση του σφάλματος και την διάδοσή του στην έξοδο. Το σημείο επανασύγκλισης είναι ο κόμβος Ε όπως φαίνονται στα παρακάτω σχήματα:



Ελεύθερο από σφάλματα κύκλωμα

Ύπαρξη σφάλματος μόνιμης τιμής

3.7 Ειδικές Σχεδιαστικές Τεχνικές

Ο σχεδιασμός για την αύξηση της ικανότητας ελέγχου ορθής λειτουργίας (***designfortestability – DFT***) αποτελεί την αδιαμφισβήτητη τεχνική επίτευξης του ελέγχου της ορθής λειτουργίας στα σύγχρονα ΟΚ.

- Χρήση τεχνικών σειριακής σάρωσης (***scantechnique***). Επιτρέπουν την πρόσβαση στις εσωτερικές καταστάσεις του ΟΚ.
- Τεχνικές ενσωματωμένου αυτοελέγχου (***built-inselftest-BIST***). Ενσωμάτωση στο ΟΚ τεχνικών και κυκλωμάτων που θα επιτρέψουν τον αυτοέλεγχό του. Τα κυκλώματα αυτά παρέχουν διανύσματα εισόδου στο κυρίως κύκλωμα και/ή παρατηρούν τις εξόδους για την ανίχνευση λαθών. Ενδέχεται να παρέχουν κάλυψη σφαλμάτων και κατά τη λειτουργία του κυκλώματος (***on-linetesting***).
- Τεχνικές περιφερειακής σάρωσης (***boundaryscan***). Πρόκειται για τεχνικές σε επίπεδο ΟΚ που επιτρέπουν με τη χρήση ενός μικρού αριθμού από εξειδικευμένες ακροδέκτες σημάτων να πραγματοποιηθεί γρήγορα και πρακτικά ο έλεγχος της ορθής λειτουργίας (π.χ. πρωτόκολλα IEEE 1149.1 και IEEE P 1500).

3.7.1 Ειδικό σχέδιο για τις τεχνικές σχεδίασης ελεγξιμότητας

Πολλές από τις τεχνικές που αναπτύχθηκαν για πλακέτες τυπωμένων κυκλωμάτων, όπου μερικές από αυτές ισχύουν για το σχεδιασμό των ολοκληρωμένων κυκλωμάτων (**IC**). Θεωρούνται ειδικοί είναι (και όχι αλγοριθμική), επειδή δεν εξετάζουν μια συνολική μεθοδολογία σχεδιασμού που εξασφαλίζει ευκολία της παραγωγής δοκιμής, και μπορούν να χρησιμοποιηθούν ως επιλογή των σχεδιαστών ανάλογα με την περίπτωση. Ο στόχος τους είναι να αυξηθεί η ελεγξιμότητα, η παρατηρησιμότητα, και / ή την προβλεψιμότητα.

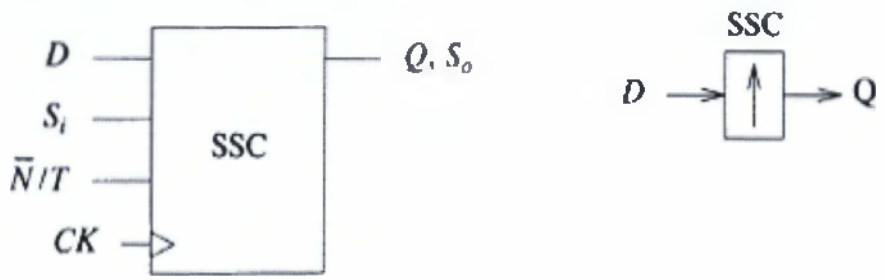
Οι τεχνικές σχεδίασης για ελεγξιμότητα (**DFT**) εξετάζει τις ακόλουθες έννοιες:

- σημεία ελέγχου – δοκιμής,
- αρχικοποίηση – έναρξη,
- μονοσταθής multivibrators (*ένα - βολές*),
- ταλαντωτές και ρολόγια,
- κατάλογος μετρητών/μετατόπισης,
- στεγανοποίηση – χωρισμός των μεγάλων κυκλωμάτων,
- λογικός πλεονασμός,
- σφαιρικό σπάσιμο ανατροφοδότησης πορειών/μονοπατιών.

3.8 Ελεγξιμότητα και παρατηρησιμότητα, με τη βοήθεια καταχωρητών ολίσθησης

Με τη χρησιμοποίηση των σημείων ελέγχου, μία σάρωση ενισχύει εύκολα την παρατηρησιμότητα και την ελεγξιμότητα ενός κυκλώματος. Αλλά η ενίσχυση μπορεί να είναι δαπανηρή από την άποψη των I/O ακροδεκτών. Ένας άλλος τρόπος για να ενισχυθεί η παρατηρησιμότητα και η ελεγξιμότητα είναι με τη χρησιμοποίηση ενός καταχωρητή σάρωσης (*Scan Register*). Η χρήση του καταχωρητή σάρωσης για να αντικαταστήσει τα I/O pins (καρφίτσες) εξετάζει τη σχέση μεταξύ του χρόνου δοκιμής, της περιοχής εξόδου, και I/O pins. Το σχήμα που ακολουθεί, δείχνει μια γενική μορφή μιας σάρωσης αποθήκευσης (**SSC**) και του αντίστοιχου καταχωρητή σάρωσης. Εδώ όταν $\bar{N}$/T=0 (κανονική λειτουργία), τα δεδομένα φορτώνονται στο κύτταρο αποθήκευσης σάρωσης από τη γραμμή εισαγωγής δεδομένων (D), όταν οι $\bar{N}$/T=1 (δοκιμαστική λειτουργία) τα στοιχεία φορτώνονται από το Si. Μια σάρωση καταλόγων R μετατοπίζονται, όταν $\bar{N}$/T=1 φορτώνει τα δεδομένα, παράλληλα, όταν $\bar{N}$/T=0.

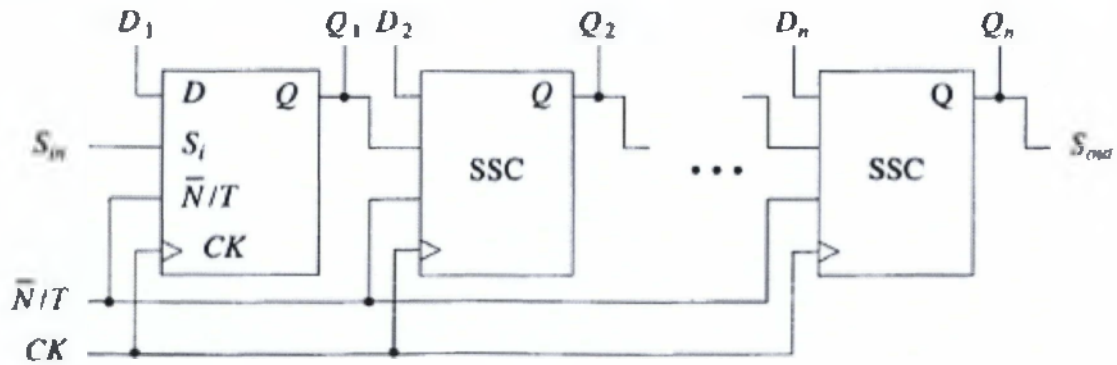
Τα δεδομένα φορτώνονται στο R από την γραμμή Sin όταν $\bar{N}/T$=1 αναφέρεται ως μια σάρωση-εισόδου, σε λειτουργία, η ανάγνωση των δεδομένων από το R, από τη γραμμή Sout αναφέρεται ως μια σάρωση – εξόδου, σε λειτουργία.



(α) (β)

Σχήμα 3.8 (α): Ένα κύτταρο σάρωσης αποθήκευσης (SSC)

Σχήμα 3.8 (β): Σύμβολο για ένα SSC



(γ)

(δ)

Σχήμα 3.8 (γ): Μια αλυσίδα (**SR**) σάρωσης ή καταχωρητών ολίσθησης

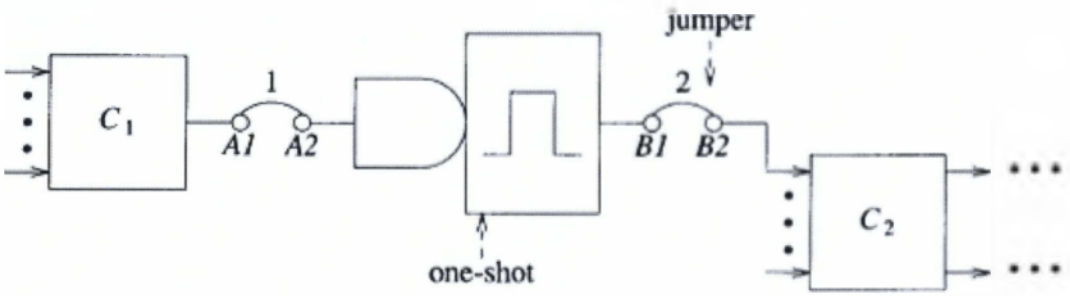
Σχήμα 3.8 (δ): Σύμβολο για έναν καταχωρητή σάρωσης

3.9 Ταλαντωτές και Ρολόγια

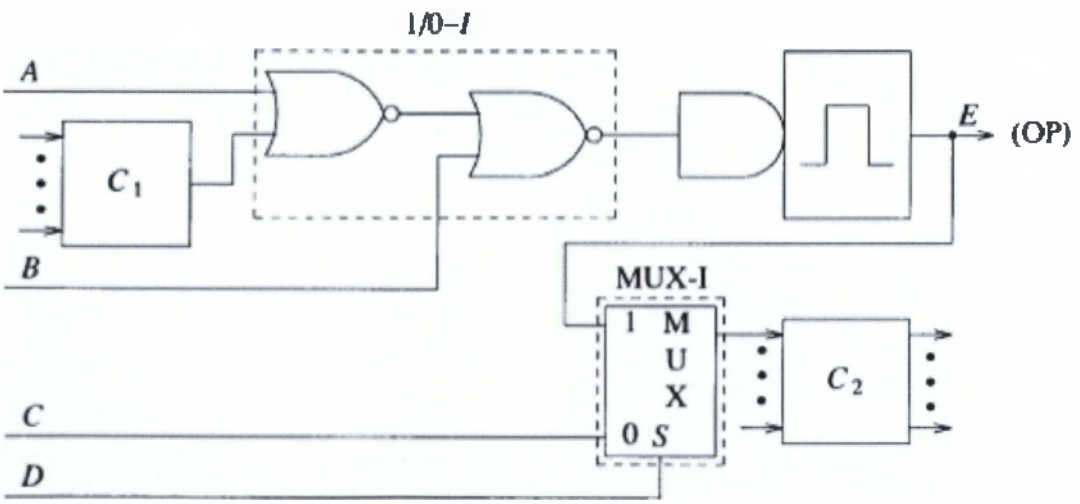
Κανόνας: Θέστε εκτός λειτουργίας τους εσωτερικούς ταλαντωτές και τα ρολόγια κατά τη διάρκεια της δοκιμής

Η χρήση των ελεύθερων τρεχόντων ταλαντωτών ή των ρολογιών οδηγεί σε προβλήματα συγχρονισμού παρόμοια με αυτά που προκαλούνται από one-shot. Οι τεχνικές για να αποκτήσουν τη δυνατότητα ελέγχου και παρατηρησιμότητας είναι επίσης παρόμοιες. Το πιο κάτω σχήμα δείχνει ένα τέτοιο κύκλωμα διαμόρφωσης **DFT**.

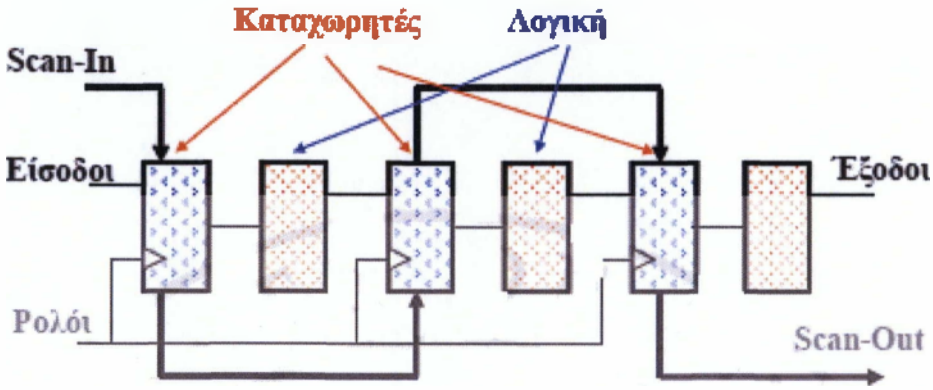
Σχήμα 3.9(α): Να θέσει εκτός λειτουργίας ένα one-shoot χρησιμοποιώντας jumpers.



Σχήμα 3.9(β): λογικός έλεγχος (*logicalcontrol*) και απενεργοποίηση της λειτουργίας του one-shot.



3.10 Σειριακή Σάρωση



Έχουμε ενοποίηση των καταχωρητών του κυκλώματος σε έναν ενιαίο ολισθητή καταχωρητή γνωστό ως καταχωρητή σάρωσης (***scanregister***). Κατά αυτόν τον τρόπο η εσωτερική κατάσταση του κυκλώματος μπορεί να καθορίζεται εισάγοντας σειριακά δεδομένα στον καταχωρητή σάρωσης. Επιπλέον οποιαδήποτε εσωτερική κατάσταση μπορεί να παρατηρηθεί από το εξωτερικό περιβάλλον εξάγοντας σειριακά τα δεδομένα του καταχωρητή σάρωσης.

3.11 Ο Αυτοέλεγχος

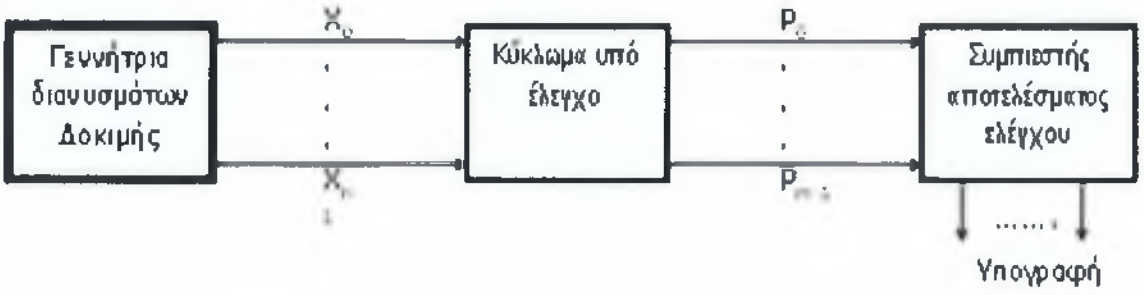
Διάφορες μορφές αυτοελέγχου χρησιμοποιούνται στο σύγχρονο ψηφιακό σχεδιασμό ολοκληρωμένου συστήματος. Στις πιο κάτω ενότητες που ακολουθούν εξετάζονται μερικές από αυτές. Ο έλεγχος που πραγματοποιείται συνήθως για τον εντοπισμό μόνιμων βλαβών και μόνο, καλείται έλεγχος εκτός κανονικής λειτουργίας (*off-linetesting*), αφού το υπό έλεγχο κύκλωμα δεν είναι δυνατόν να πραγματοποιεί την κανονική του λειτουργία την ώρα που ελέγχεται. Από την άλλη, αν θέλουμε να είμαστε σίγουροι ότι ένα κύκλωμα λειτουργεί συνεχώς αξιόπιστα, τότε θα πρέπει αυτό να ελέγχεται κατά την διάρκεια της κανονικής του λειτουργίας. Ο έλεγχος αυτός ονομάζεται έλεγχος κατά την κανονική λειτουργία (*on-linetesting*), έχει τη δυνατότητα να ανιχνεύει και προσωρινές βλάβες του συστήματος, και για την εφαρμογή του χρησιμοποιούνται εντελώς διαφορετικές τεχνικές από αυτές που χρησιμοποιούνται για τον έλεγχο εκτός κανονικής λειτουργίας

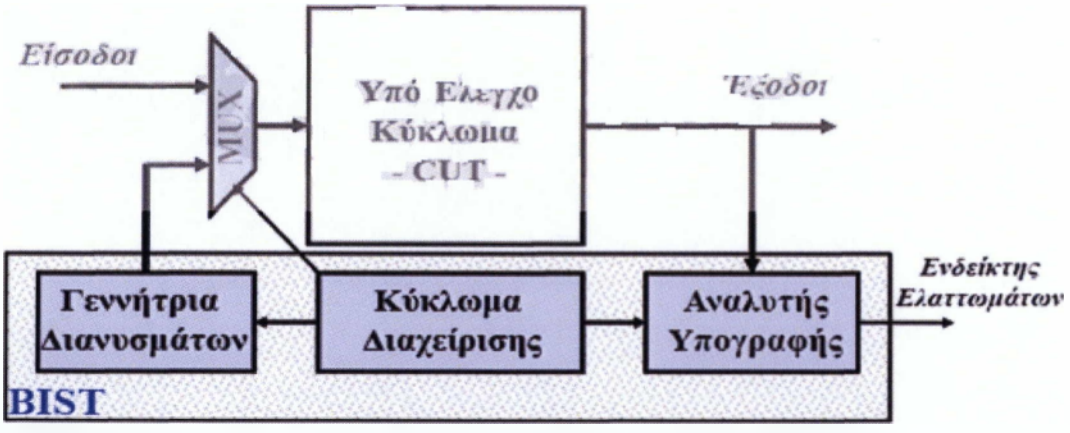
3.11.1 Ενσωματωμένος Αυτοέλεγχος (*BISIT*)

Ο ενσωματωμένος αυτοέλεγχος είναι η ενσωμάτωση της δυνατότητας ελέγχου μέσα στο ίδιο κύκλωμα, έτσι ώστε να μην απαιτείται εξωτερικός εξοπλισμός.

Το παρακάτω σχήμα μας δείχνει μία πιθανή μορφή ελέγχου **BIS**T (*Built-InSelf-Test*), όπου μία γεννήτρια διανυσμάτων δοκιμής παράγει τα διανύσματα δοκιμών που πρέπει να εφαρμοστούν στο κύκλωμα υπό έλεγχο.Οι τεχνικές BIST προσπαθούν να συνδυάσουν στην ίδια επιφάνεια πυριτίου τόσο το υπό
έλεγχο κύκλωμα όσο και το κύκλωμα ελέγχου, δίνοντας έτσι τη δυνατότητα στο συνολικό κύκλωμα να «αυτοελέγχεται».

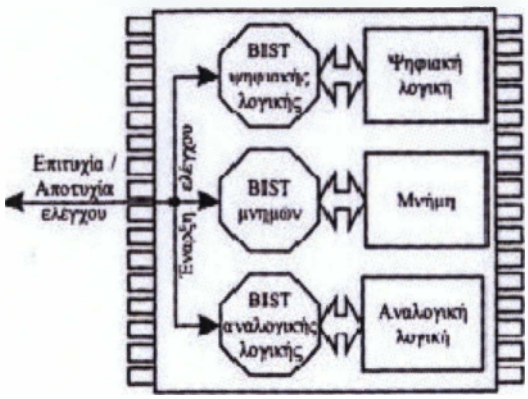
Σχήμα 3.11.1(α): Ενσωματωμένος αυτοέλεγχος





Στον ενσωματωμένο αυτοέλεγχο τα διανύσματα ελέγχου δημιουργούνται εσωτερικά στο ΟΚ (***Ολοκληρωμένο Κύκλωμα***) και εφαρμόζονται κάτω από τον έλεγχο ενός κυκλώματος διαχείρισης (***BISTcontroller***). Οι αποκρίσεις του κυκλώματος αναλύονται από τον αναλυτή υπογραφής και το τελικό αποτέλεσμα μετά το πέρας του ελέγχου της ορθής λειτουργίας συγκρίνεται με το αναμενόμενο έτσι ώστε το κύκλωμα διαχείρισης να επισημάνει με το κατάλληλο σήμα στο εξωτερικό περιβάλλον την πιθανή ύπαρξη σφαλμάτων στο ΟΚ.

Στο παρακάτω σχήμα **3.11.1(β),** παρουσιάζεται η εφαρμογή ελέγχου με χρήση BIST δομών. Όπως βλέπουμε, οι απαιτήσεις από τον εξωτερικό ελεγκτή είναι οι ελάχιστες δυνατές, αφού το μόνο που πρέπει να γίνει από τη μεριά του, είναι να ενεργοποιηθεί ο αυτοέλεγχος στην αρχή και μετά το πέρας αυτού, να ελεγχθεί το αποτέλεσμά του (αν δηλαδή είναι επιτυχής ή μη).

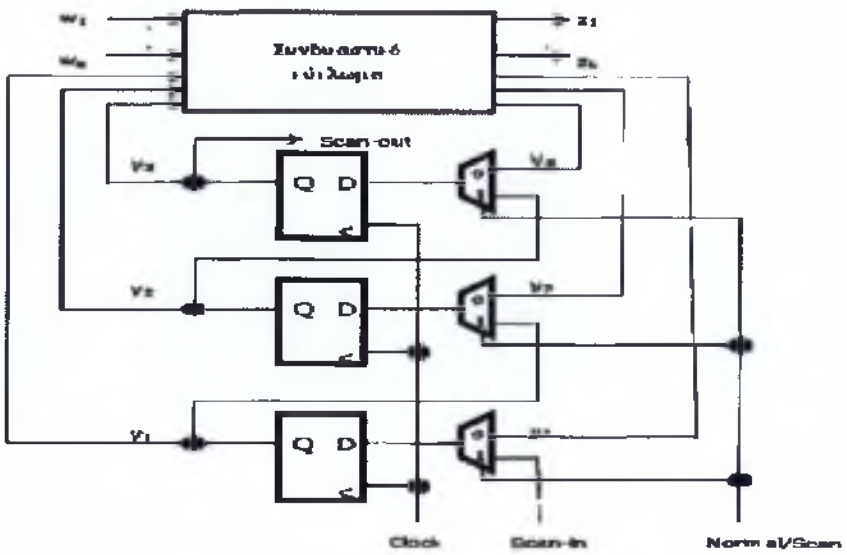


Σήμα 3.11.1(β): Εφαρμογή του ελέγχου με χρήση BIST δομών.

3.11.2 Έλεγχος Ακολουθιακών Κυκλωμάτων

Ο έλεγχος ακολουθιακών κυκλωμάτων είναι πολύ δύσκολος αφού η ύπαρξη στοιχείων μνήμης επιτρέπει σε ένα ακολουθιακό κύκλωμα να βρίσκεται σε διάφορες καταστάσεις και η απόκριση του κυκλώματος σε εξωτερικά εφαρμοζόμενες εισόδους εξαρτάται από την κατάσταση του κυκλώματος. Στο παρακάτω σχήμα φαίνεται η τεχνική σάρωσης διαδρομής και δίνεται το κύκλωμα που επιτρέπει τη διεξαγωγή ελέγχου με την τεχνική σάρωσης διαδρομής.

Σχήμα 3.11.2: Έλεγχος ακολουθιακών κυκλωμάτων



Εξελίξεις

Η αλματώδης εξέλιξη των ψηφιακών συστημάτων, το χαμηλό κόστος τους και η εντυπωσιακή πολυπλοκότητά τους δεν θα ήταν δυνατά χωρίς τους ανθρώπους και τα μέσα που εξασφαλίζουν ταχύτερους και αποτελεσματικότερους τρόπους μελέτης, σχεδίασης, ανάπτυξης και κατασκευής τους. Αν και το αντικείμενο αυτού του βιβλίου είναι η Ψηφιακή Σχεδίαση, είναι αδύνατο να παρουσιασθούν όλοι αυτοί οι τομείς ανάπτυξης ενός σύγχρονου ψηφιακού συστήματος σ' ένα μόνο βιβλίο, πολύ λιγότερο δε στην παρούσα ενότητα.

Σκιαγραφούνται απλώς εδώ:

- Η επιλογή του ηλεκτρικού ρεύματος σαν φορέα.
- Η θεωρητική βάση για τη μελέτη των ψηφιακών συστημάτων (*C.Shannon*)
- Η προσφορά της μικροηλεκτρονικής

Η μεταβιομηχανική εποχή, και ιδιαίτερα η σύγχρονη μορφή της, η εποχή της πληροφορικής, δημιούργησαν την ανάγκη να βρεθούν αποτελεσματικότερα μέσα για τη μετάδοση και επεξεργασία των πληροφοριών. Οι "φρυκτωρίες" της Ομηρικής εποχής, ένα από τα αρχαιότερα ψηφιακά συστήματα τηλεπικοινωνίας, το οποίο χρησιμοποιούσε συνδυασμούς πυρσών, προ πολλού έπαψαν να είναι αποτελεσματικά. Οι "calculators" του 19ου αιώνα, δηλ. οι άνθρωποι που είχαν ως επάγγελμα την εκτέλεση αριθμητικών πράξεων, είναι και αυτοί ανεπαρκείς.

Τα ψηφιακά συστήματα, βοηθούμενα και από τις εξελίξεις στην τεχνολογία της μικροηλεκτρονικής, έχουν εξαπλωθεί σε κάθε είδους εφαρμογές. Ταυτόχρονα όμως οι ανάγκες της βιομηχανίας για γρήγορη παραγωγή ολοένα και περισσότερο πολύπλοκων συστημάτων απαιτούν την συνεχή βελτίωση των μεθόδων σχεδίασης και κατασκευής τους. Ο σχεδιασμός αυτών των συστημάτων απαιτεί ειδικές γνώσεις και είναι δυνατόν να γίνει μόνο με την βοήθεια υπολογιστών

Η Ευρώπη αντιπροσωπεύει τη στιγμή αυτή το ένα τρίτο περίπου της παγκόσμιας αγοράς ψηφιακών τεχνολογιών, αλλά παράγει μόλις το 23%.

- Ο συντονισμός μεταξύ των έργων ψηφιακής έρευνας και καινοτομίας στα 27 κράτη μέλη είναι ανεπαρκής.
- Εξακολουθούν να υπάρχουν φραγμοί μεταξύ των εθνικών αγορών για καινοτόμα ψηφιακά προϊόντα και υπηρεσίες. Για να δημιουργηθούν νέες επιχειρήσεις και να μπορέσουν να κατακτήσουν την παγκόσμια αγορά, πρέπει η Ένωση να λειτουργεί ως ενιαία αγορά.
- Οι δημόσιες αρχές συχνά καθυστερούν πολύ να υιοθετήσουν καινοτόμα ψηφιακά προϊόντα και υπηρεσίες προκειμένου να εκσυγχρονίσουν τα συστήματα υγείας, μεταφορών ή εκπαίδευσης. Αν αλλάξει η στάση αυτή, θα δημιουργηθούν νέες αγορές και θα προσελκυστούν νέοι επενδυτές.

-ΤΙ ΑΚΡΙΒΩΣ ΠΡΟΚΕΙΤΑΙ ΝΑ ΑΛΛΑΞΕΙ;

• Η στρατηγική προτείνει περισσότερες επενδύσεις σε έρευνα και καινοτομία στον τομέα των ψηφιακών τεχνολογιών, αποτελεσματικότερο συντονισμό των επενδύσεων σε όλη την ΕΕ, και άνοιγμα νέων ευρωπαϊκών αγορών για τις καινοτόμες ψηφιακές υπηρεσίες.
• Θα βοηθήσει την Ευρώπη να παραμείνει στην πρωτοπορία όσον αφορά τη διαμόρφωση και την αξιοποίηση των μελλοντικών εξελίξεων στις ψηφιακές τεχνολογίες, σύμφωνα με την στρατηγική της ΕΕ για την καινοτομία. Επίσης, θα οδηγήσει σε πρωτοβουλίες ενίσχυσης της ερευνητικής συνεργασίας εντός της Ευρώπης.
• Θα δημιουργήσει κίνητρα ώστε η Ευρώπη να επενδύσει περισσότερο στην τεχνολογία του 21ου αιώνα και θα τονώσει την ανάπτυξη επιχειρήσεων που βασίζονται στις πλέον πρόσφατες καινοτομίες της ΤΠ.
• Παράλληλα, θα βοηθήσει τους ευρωπαίους προμηθευτές να κατακτήσουν μεγαλύτεροτμήμα της παγκόσμιας αγοράς ψηφιακών τεχνολογιών, και θα δώσει ώθηση σε σημαντικές ιδιωτικές και δημόσιες επενδύσεις, ειδικότερα από μικρές επιχειρήσεις.

-ΠΟΙΟΙ ΘΑ ΟΦΕΛΗΘΟΥΝ ΚΑΙ ΠΩΣ;

• Με τη στρατηγική αυτή θα δημιουργηθεί μεγαλύτεροςσυσχετισμός μεταξύ των αναγκών και των επιθυμιών της κοινωνίας και των επιχειρήσεων, αφενός, και των τεχνολογικών δυνατοτήτων και στόχων των παραγωγών.
• Θα αυξηθούν οι επενδύσεις στους τομείς ψηφιακής τεχνολογίας.
• Θα ενισχυθεί η συνεργασία μεταξύ της βιομηχανίας, των εκπαιδευτικών ιδρυμάτων και των δημόσιων αρχών.
• Η ευρεία χρήση των ψηφιακών τεχνολογιών θα συμβάλει στη δημοσιονομική και οικονομικήανάκαμψη, δεδομένου ότι θα δημιουργηθούν νέες δημόσιες και ιδιωτικές αγορές, θα μικρύνουν οι κύκλοι καινοτομίας και θα δοθούν ταχύτερα απαντήσεις στις κυριότερες κοινωνικοοικονομικές προκλήσεις.

-ΓΙΑΤΙ ΠΡΕΠΕΙ ΝΑ ΑΝΑΛΑΒΕΙ ΔΡΑΣΗ Η ΕΕ;

• Η ψηφιακή έρευνα και καινοτομία έχουν εξ ορισμού διεθνή διάσταση και οι εξελίξεις υπαγορεύουν όλο και μεγαλύτερη διακρατική συνεργασία.
• Κοινά και εταιρικά σχέδια είναι θεμελιώδουςσημασίας για την ανταλλαγή στρατηγικών και πολιτικών, τη συνδυασμένη αξιοποίηση χρηματοδοτικών πόρων και ανθρώπινου δυναμικού, και την εφαρμογή των ψηφιακών τεχνολογιών και λύσεων σε όλη την Ευρώπη αλλά και πέρα από αυτή.

Βιβλιογραφία

[1]. **M. Morris Mano and Charles R. Kime**, *Logic and Computer DesignFundamentals (Third Rdition),***Pearson Prentice Hall, New Jersey, 2004**

[2]. **DonaldP. LeachandAlbertPaulMalvino**, *Ψηφιακά ΗλεκτρονικάΘεωρία και Εφαρμογές (5η Έκδοση),***εκδόσεις ΤΖΙΟΛΑ Ε, 1996**

[3]. **Schaum'soutlineseries**, *Θεωρία και προβλήματα στην εισαγωγή τωνψηφιακών (2η Έκδοση),***εκδόσεις Α. ΤΖΙΟΛΑ Ε, 1998**

[4]. **Bruer, M.A, and A.D. Friedman (1976)**, *Diagnostic and Reliable Designof Digital Systems, Computer Science Press,***New York**

[5]. **Abadir, M., et al. (1989),***Logic design verification via test generation, IEETrans. Comput.-Aided Des. Integrated Circuits Syst., Vol. CAD-7, No.1*

[6]. **Eldret, R. D. (1959),***Tests routins based on symbolic logic systems, J. ACM, Vol. 6, No 1*

[7]. **Fujiwara, H. (1986)**, *Logic Testing and Design for Testability, MIT Press, Cambridge, MA.*

[8]. **McCluskey, E. J. (1984),***Verification testing: a pseudo-exhaustive test technique, IEE Trans. Comput., Vol.C-33, No 6*

[9]. **Bardell, P. H. (1987),Built in Test for VLSI**: *Pseudo-random Techniques, Wiley, New York*

[10]. **Rajski, J., and J. Tyzer (1998),***Built-in Self-Test for embedded systems, Prentice Hall*

[11]. **Agrawal, V., et al. (1994),***Built-in Self-Test for digital integrated circuits, AT&T Tech. J., Vol. 73*

[12]. **Laung-Terng Wang, Cheng-Wen Wu, Cheng-Wen Wu (EE Ph.D),***VLSI test principles and architectures : design for testability*

[13]. **R. L. Tokheim, Ψηφιακά Ηλεκτρονικά**, *Εκδόσεις **Α. ΤΖΙΟΛΑ**, Θεσσαλονίκη 1991, Έκδοση 5η*

[14]. **NelsonVictorP., NagleH. Troy, IrwinJ. David, CarrollBill**, *Ανάλυση και σχεδίαση κυκλωμάτων ψηφιακής λογικής, Εκδόσεις Επίκεντρο Α.Ε., Έκδοση 1η 2007*

[15]. **M. Morris Mano, Digital Design (3rd Edition)**, *Prentice Hall 2001*

[16]. **John F. Wakerly, Digital Design:***Principles and Practices (3rd Edition), Prentice Hall 2000*

Ιστοσελίδες

- http://www.wikipedia.org
- http://www.tech-faq.com
- http://www.technologyevaluation.com/
- http://www.noc.teithe.gr/-kath/.../digit_gr.htm
- http://www.anamorfosi.teiser.gr/d11_kat3_exam6.html
- http://www.uoi.gr/gr/schools/sciences/cs.php
- http://www.pyr.gr/.../ilektol-mixanikon-texnologias-ypolog.htm
- http://www.ece.uc.edu/-wjone/DFTnew.pdf
- http://www.cmosvli.com
- http://www.databasecentral.info/ECE%20271/DFT_notes.pdf
- http://www.cs.colostate.edu/-cs530/digital_testing.pdf
- http://www.codebetter.com/.../design-and-testability/