Αυτόματη Ροή για Αποσυγχρονισμό Ψηφιακών Κυκλωμάτων

Ανδρίκος Νικόλαος
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Επιτηρητής καθηγητής: Σωτηρίου Χρήστος
Επόπτης καθηγητής: Κατεβαίνης Μανόλης

19 Ιουλίου 2004

Στην οικογένειά μου

Περίληψη

Τα ασύγχρονα κυκλώματα έχουν αρκετά και σημαντικά πλεονεκτήματα σε σχέση με τα αντίστοιχα σύγχρονα τους σύμφωνα με την βιβλιογραφία. Παραθέτονται όμως και μερικά ζητήματα που εμποδίζουν την εκτεταμένη υιοθέτηση της τεχνικής. Το κυριότερο πρόβλημα είναι η έλλειψη βιομηχανικής ποιότητας εργαλείων για ασύγχρονο σχεδιασμό. Ο αποσυγχρονισμός είναι μία μέθοδος για την εύκολη δημιουργία ασύγχρονων κυκλωμάτων από σύγχρονη περιγραφή που επιτρέπει την αυτοματοποιημένη υλοποίηση ασύγχρονων κυκλωμάτων με τα υπάρχοντα βιομηχανικά εργαλεία. Η εργασία αυτή επικεντρώνεται στην δημιουργία μίας αυτόματης ροής για την υλοποίηση αποσυγχρονισμένων κυκλωμάτων η οποία λαμβάνει την περιγραφή ενός σύγχρονου κυκλώματος, υλοποιεί την αποσυγχρονισμένη έκδοσή του και παράγει την τελική διάταξη. Η ροή αυτή υλοποιήθηκε, ενοποιήθηκε με εμπορικά εργαλεία σύγχρονης σχεδίασης και δοκιμάστηκε με επιτυχία σε πραγματικά κυκλώματα.

Ευχαριστίες

Καταρχήν να εκφράσω την ευγνωμοσύνη μου στην οικογένειά μου για την στήριξη που μου παρείχε όλα αυτά τα χρόνια. Χωρίς την δική τους βοήθεια και συμπαράσταση δεν θα μπορούσα να είχα φτάσει εδώ.

Έπειτα να ευχαριστήσω τη Φανή που μου ομορφαίνει τη ζωή και μου δίνει κουράγιο όταν το χρειάζομαι.

Θα ήθελα να αναγνωρίσω τη συμβολή του Χρήστου για την ολοκλήρωση αυτής της εργασίας και να τον ευχαριστήσω για την καθοδήγησή, τα εποικοδομητικά του σχόλια και τις ώρες που αφιέρωσε.

Τέλος, θα ήθελα να ευχαριστήσω τον Βαγγέλη που ήταν δίπλα μου, μεταφορικά και κυριολεκτικά, καθ΄ όλη τη διάρκεια αυτής της εργασίας υπενθυμίζοντας μου ότι δεν είμαι μόνος.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

Ένας διαχωρισμός που μπορεί να γίνει στα ψηφιακά κυκλώματα είναι μεταξύ ασύγχρονων και σύγχρονων. Το κύριο χαρακτηριστικό των σύγχρονων κυκλωμάτων είναι η ύπαρξη δικτύων ρολογιού. Τα ρολόγια αποτελούν τα σήματα ενεργοποίησης των στοιχείων μνήμης του κυκλώματος. Παράγονται συνήθως εκτός του ολοκληρωμένου με κρυσταλλικούς ταλαντωτές και έπειτα διανέμονται εσωτερικά σε ολόκληρο το σύστημα. Η χρήση ρολογιού προσφέρει κοινή χρονική αναφορά απλουστεύοντας την σχεδίαση

Τα ασύγχρονα κυκλώματα είναι αρκετά διαφορετικά. Δεν υπάρχουν ένα ή περισσότερα καθολικά ρολόγια, αλλά πολλά επιμέρους τοπικά σήματα ενεργοποίησης που οδηγούν ένα η περισσότερα στοιχεία μνήμης. Για την παραγωγή αυτών των σημάτων δεν χρησιμοποιούνται εξωτερικοί κρυσταλλικοί ταλαντωτές αλλά ασύγχρονα κυκλώματα που ονομάζονται ελεγκτές μανταλωτών (latch controllers). Οι ελεγκτές συνδέονται μεταξύ τους και συγχρονίζονται με κάποιο πρωτόκολλο χειραψίας(handshaking) ώστε να εξασφαλίζουν την σωστή μετάδοση των δεδομένων. Παρακάτω παραθέτουμε τα πλεονεκτήματα και τα μειονεκτήματα της ασύγχρονης σχεδίασης.

1.1 Πλεονεκτήματα της ασύγχρονης σχεδίασης

- Χαμηλότερη κατανάλωση ενέργειας

  Η ορθή σχεδίαση των ασύγχρονων κυκλωμάτων μπορεί να οδηγήσει σε κυκλώματα με χαμηλότερη κατανάλωση ενέργειας. Αυτό οφείλεται κυρίως στο ότι δεν υπάρχει ένα καθολικό σήμα ρολογιού που ανεβοκατεβαίνει καθ΄ όλη την διάρκεια της λειτουργίας του κυκλώματος. Σε ένα σύγχρονο κύκλωμα το ρολόι συμβατικά οδηγεί και τμήματα που δεν χρησιμοποιούνται στη δεδομένη στιγμή καταναλώνοντας περιττή ενέργεια. Έτσι, οι τεχνικές για ασύγχρονη σχεδίαση που δεν αυξάνουν σημαντικά το μέγεθος του κυκλώματος παρουσιάζουν χαμηλότερη κατανάλωση ενέργειας.

  Σημαντική κατανάλωση ενέργειας παρατηρείται επίσης και στο δέντρο ενίσχυσης του σήματος του ρολογιού(clock tree). Το δέντρο αυτό καλύπτει ολόκληρο το ολοκληρωμένο και γιαυτό αποτελείται από μεγάλο αριθμό πυλών. Όλες αυτές οι πύλες αναβοσβήνουν με την συχνότητα του ρολογιού αυξάνοντας σημαντικά την συνολική κατανάλωση ενέργειας του κυκλώματος. Τα ασύγχρονα κυκλώματα δεν περιέχουν καθολικό ρολόι. Τα σήματα ενεργοποίησης που χρησιμοποιούν μπορεί να χρειάζονται ενίσχυση, αλλά είναι περιορισμένα τοπικά κι έτσι το δέντρο ενίσχυσης τους είναι περιορισμένο.

- Ευκολότερη διανομή των ρολογιών

  Ένα από τα μεγαλύτερα προβλήματα που υπάρχουν στα σημερινά σύγχρονα κυκλώματα είναι της απόκλισης άφιξης του σήματος ρολογιού (clock skew) σε όλα τα σημεία του κυκλώματος. Η μέγιστη διαφορά ανάμεσα στους χρόνους άφιξης του ρολογιού σε δύο καταχωρητές προστίθεται στην καθυστέρηση της λογικής για να υπολογίσουμε την ελάχιστη περίοδο λειτουργίας του κυκλώματος. Καθώς οι ταχύτητες λειτουργίας των κυκλωμάτων αυξάνονται με την πάροδο του χρόνου, το πρόβλημα αυτό γίνεται εντονότερο καθώς η χρονική απόκλιση του ρολογιού αποτελεί ολοένα και μεγαλύτερο ποσοστό της περιόδου του.

Κάποιες ασύγχρονες προσεγγίσεις δεν περιέχουν καθόλου σήματα γενικού συγχρονισμού ενώ κάποιες άλλες, όπως ο αποσυγχρονισμός για παράδειγμα, περιέχουν σήματα ενεργοποίησης τα οποία πρέπει να διανεμηθούν στο κύκλωμα αντίστοιχα με την διανομή του ρολογιού. Ακόμα και σε αυτές τις περιπτώσεις όμως το πρόβλημα είναι ευκολότερο επειδή αυτά τα σήματα προορίζονται για μία τοπικά περιορισμένη περιοχή του κυκλώματος. Έτσι, η απόκλιση για το κάθε τέτοιο σήμα είναι μικρότερη καθώς δεν χρειάζεται να διανεμηθεί σε ολόκληρο το ολοκληρωμένο, παρά μόνο σε περιορισμένη περιοχή.

- Καλύτερη ιεραρχικότητα και επαναχρησιμοποίηση

  Τα ασύγχρονα υποκυκλώματα είναι ολοκληρωμένες μονάδες με τοπικές χρονικές απαιτήσεις. Έτσι, μεγαλύτερα κυκλώματα συντίθενται εύκολα από μικρότερα υποκυκλώματα καθώς οι τοπικοί περιορισμοί δεν ανάγονται σε καθολικούς. Τα σύγχρονα κυκλώματα απεναντίας περιέχουν μόνο τις απαιτήσεις δεδομένων και μοιράζονται το ρολόι μεταξύ τους. Γι αυτό το λόγο ο σχεδιαστής χρειάζεται να φροντίζει για τις χρονικές απαιτήσεις του καθενός, όπως είναι η συχνότητα του ρολογιού. Έτσι τα ήδη υλοποιημένα ασύγχρονα υποκυκλώματα προσαρμόζονται ευκολότερα στο κύκλωμα σε σχέση με τα αντίστοιχα σύγχρονά τους.

- Μεγαλύτερη ταχύτητα λειτουργίας

  Στα ασύγχρονα κυκλώματα μπορούν να εφαρμοστούν τεχνικές που παρέχουν γνώση της ολοκλήρωσης του κάθε υπολογισμού. Με αυτή την γνώση μπορεί να βελτιωθεί σημαντικά η απόδοση των ασύγχρονων κυκλωμάτων καθώς η ταχύτητα τους θα καθορίζεται από τα δεδομένα και τις πραγματικές καθυστερήσεις κι όχι από την καθυστέρηση της κρίσιμης οδού (critical path). Έτσι χρειάζεται να περιμένουμε μόνο ακριβώς όσο πραγματικά χρειάζεται. Τα σύγχρονα κυκλώματα απ΄ εναντίας πρέπει να περιμένουν πάντα την χειρότερη καθυστέρηση ακόμα κι αν αυτή συμβαίνει πολύ σπάνια.

- Αντιμετώπιση μεταβλητότητας (variability)

  Μεταβλητότητα είναι το φαινόμενο κατά το οποίο όμοια στοιχεία ενός κυκλώματος παρουσιάζουν διαφορετικά στατικά ή δυναμικά χαρακτηριστικά λόγω κατασκευής ή των συνθηκών λειτουργίας. Μεταβλητότητα παρουσιάζεται όχι μόνο μεταξύ δειγμάτων του ίδιου ολοκληρωμένου, αλλά και μέσα στο ίδιο ολοκληρωμένο. Ένα τέτοιο παράδειγμα είναι μία πύλη να έχει καθυστέρηση πολύ μεγαλύτερη από την αναμενόμενη. Αυτό είναι δυνατόν να οφείλεται για παράδειγμα στην πολύ υψηλή θερμοκρασία του κυκλώματος σε εκείνο το σημείο. Το πρόβλημα αυτό μπορεί να οδηγήσει σε ανεπιθύμητα αποτελέσματα όπως στην παραβίαση της περιόδου του ρολογιού, αν το πρόβλημα είναι στο κρίσιμο μονοπάτι. Σε αυτή την περίπτωση το κύκλωμα δεν δουλεύει σωστά καθώς τα δεδομένα κατά την άφιξη του ρολογιού δεν είναι έγκυρα

  Το πρόβλημα μπορεί να αντιμετωπιστεί αποτελεσματικά από ασύγχρονες προσεγγίσεις που παρέχουν γνώση για την ακριβή στιγμή ολοκλήρωσης του κάθε υπολογισμού. Με αυτές τις προσεγγίσεις το κύκλωμα "περιμένει" τον κάθε υπολογισμό ακριβώς όσο διαρκεί και τα δεδομένα διαβάζονται μόνο όταν είναι πραγματικά έγκυρα. Έτσι το κύκλωμα δουλεύει ακόμα κι αν οι καθυστερήσεις είναι διαφορετικές από τις αναμενόμενες.

- Χαμηλή ηλεκτρομαγνητική εκπομπή

  Στα σύγχρονα κυκλώματα έχουμε ένα καθολικό ρολόι που ενεργοποιεί όλο το κύκλωμα. Κατά την ακμή του ρολογιού παρουσιάζεται μεγάλος ηλεκτρομαγνητικός θόρυβος στο ολοκληρωμένο επειδή είναι η στιγμή που αλλάζουν όλα τα σήματα. Λόγω της απουσίας του καθολικού ρολογιού στα ασύγχρονα κυκλώματα μειώνεται σημαντικά αυτός ο θόρυβος.

1.2 Προβλήματα στην υιοθέτηση του ασύγχρονου σχεδιασμού

Παρ΄ όλα τα σημαντικά πλεονεκτήματα που περιγράφηκαν στην προηγούμενη παράγραφο η ασύγχρονη σχεδίαση δεν είναι ο κανόνας στα σημερινά κυκλώματα. Αυτό οφείλεται σε έναν

αριθμό ζητημάτων που καθιστούν δύσκολη την υιοθέτηση της από την βιομηχανία κυρίως και εξηγούνται παρακάτω.

- Έλλειψη εργαλείων ηλεκτρονικού αυτοματισμού

  Το μεγαλύτερο μέρος των σύγχρονων ηλεκτρονικών κυκλωμάτων υλοποιούνται με χρήση εμπορικών εργαλείων αυτόματης ηλεκτρονικής σχεδίασης (EDA/CAD). Τα εργαλεία αυτά είναι φτιαγμένα και βελτιστοποιημένα για σύγχρονα κυκλώματα. Αυτό σημαίνει ότι σε πολλές περιπτώσεις δεν μπορούν να χρησιμοποιηθούν με τον καλύτερο τρόπο για τα ασύγχρονα κυκλώματα. Έτσι, γενικά η έρευνα στα ασύγχρονα οδηγεί στην ανάπτυξη νέων εργαλείων που να είναι ιδανικά για την ασύγχρονη σχεδίαση κυκλωμάτων. Κάποια ώριμα εργαλεία υπάρχουν αλλά είναι κυρίως ακαδημαϊκά. Η βιομηχανία όμως χρειάζεται εργαλεία της ποιότητας των ήδη υπαρχόντων εμπορικών εργαλείων. Έτσι τα εργαλεία για την ασύγχρονη σχεδίαση θα πρέπει να είναι βιομηχανικής ποιότητας, εξίσου ώριμα και η σωστή λειτουργία τους να αποδεικνύεται από πραγματικές υλοποιήσεις.

- Δυσκολότερος σχεδιασμός

  Η πλειοψηφία των σχεδιαστών υλικού (hardware) στις μέρες μας σχεδιάζει κυκλώματα με το συμβατικό σύγχρονο τρόπο. Η βασική υπόθεση είναι ότι υπάρχει ένα γενικό σήμα ρολογιού το οποίο φτάνει ταυτόχρονα σε όλους τους καταχωρητές του κυκλώματος. Η επόμενη κατάσταση είναι συνάρτηση της παρούσας κατάστασης και των εισόδων. Αυτή η προσέγγιση είναι πολύ βολική επειδή επιτρέπει στον σχεδιαστή να διαχωρίσει την λειτουργικότητα από τον χρονισμό. Στα ασύγχρονα κυκλώματα όμως δεν υπάρχει το καθολικό ρολόι. Έτσι ο σχεδιαστής απαιτείται να λαμβάνει υπ΄ όψιν του και θέματα χρονισμού. Τα προβλήματα λόγω χρονισμού, όπως σπινθήρες (hazards), κυνηγητά (races) και μεταστάθεια (metastability), καθιστούν την ασύγχρονη προσέγγιση αρκετά δυσκολότερη.

- Επιβάρυνση άμεσου συγχρονισμού

  Στα σύγχρονα κυκλώματα ο συγχρονισμός γίνεται έμμεσα από το ρολόι για όλο το κύκλωμα. Στα ασύγχρονα κυκλώματα ο συγχρονισμός είναι άμεσος και επιτελείται από τοπικούς ελεγκτές. Η εισαγωγή τους όμως στο κύκλωμα επιπλέον της ήδη υπάρχουσας λογικής μπορεί να επιβαρύνει το κύκλωμα σε ταχύτητα, μέγεθος και κατανάλωση ενέργειας.

- Χρήση πυλών ειδικού τύπου

  Σε κάποιες μεθόδους σχεδιασμού ασύγχρονων κυκλωμάτων υπάρχει η απαίτηση για ύπαρξη πυλών ειδικού τύπου. Η απουσία τους από τις βιβλιοθήκες των τεχνολογιών οδηγεί στην σύνθεσή τους από τις ήδη υπάρχουσες πύλες της τεχνολογίας αν αυτό είναι δυνατόν. Αυτό επιβαρύνει το μέγεθος και την απόδοση του παραγόμενου κυκλώματος. Στην περίπτωση που οι απαιτούμενες πύλες δεν μπορούν να προκύψουν από την υπάρχουσα τεχνολογία τότε η ασύγχρονη μέθοδος δεν μπορεί να χρησιμοποιήσει τα εργαλεία CAD καθιστώντας την υλοποίηση πολυπλοκότερη.

- Δυσκολία καθορισμού απόδοσης

  Η απόδοση στα σύγχρονα κυκλώματα υπολογίζεται χρησιμοποιώντας την περίοδο του ρολογιού και τους κύκλους που δαπανήθηκαν για κάποιο υπολογισμό. Από την άλλη, απόδοση των ασύγχρονων κυκλωμάτων επηρεάζεται ανάλογα με τα προς επεξεργασία δεδομένα. Αυτό σημαίνει ότι η δυναμική συμπεριφορά που παρουσιάζουν δυσχεραίνει πολύ τον υπολογισμό την πρόβλεψη της απόδοσης.

1.3 Στόχος της εργασίας

Η παρούσα εργασία εστιάζει στην έλλειψη εργαλείων βιομηχανικής ποιότητας για ασύγχρονα κυκλώματα. Υλοποιεί μία ροή με βάση βιομηχανικής ποιότητας εργαλεία που βασίζεται στη μέθοδο του αποσυγχρονισμού, ο οποίος παρέχει την δυνατότητα υλοποίησης ασύγχρονων κυκλωμάτων ξεκινώντας από σύγχρονη περιγραφή.

Η εργασία οργανώνεται ως εξής. Στο κεφάλαιο 2 περιγράφεται η μέθοδος του αποσυγχρονισμού και η αυτόματη ροή που υλοποιήθηκε. Στο κεφάλαιο 3 δίνονται οι λεπτομέρειες της υλοποίησης. Στο κεφάλαιο 4 παρουσιάζονται τα πειραματικά αποτελέσματα από τη χρήση της ροής. Στο κεφάλαιο 5 παρουσιάζονται τα συμπεράσματα και η μελλοντική εργασία στο αντικείμενο.

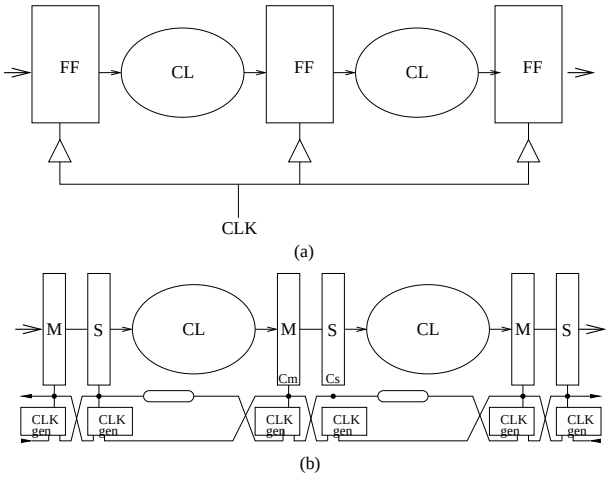
Κεφάλαιο 2

Αποσυγχρονισμός

Σε αυτό το κεφάλαιο περιγράφεται η έννοια του αποσυγχρονισμού. Επίσης γίνεται και μία γενική περιγραφή της αυτόματης ροής που υλοποιήθηκε και χρησιμοποιεί αυτή την μέθοδο για την αυτόματη υλοποίηση ασύγχρονων κυκλωμάτων ξεκινώντας από σύγχρονη περιγραφή.

2.1 Αποσυγχρονισμός

Με τον όρο αποσυγχρονισμός δηλώνεται η παραγωγή ασύγχρονων κυκλωμάτων ξεκινώντας από σύγχρονη περιγραφή [2,3]. Σε αυτή την μέθοδο το γενικό ρολόι του σύγχρονου κυκλώματος αντικαθίσταται αυτόματα με ένα δίκτυο τοπικών ελεγκτών (Σχήμα 2.1). Η μέθοδος είναι απόλυτα αυτοματοποιήσιμη και η παρούσα εργασία επικεντρώνεται στην υλοποίηση και την σύνδεσή της με υπάρχοντα εμπορικά εργαλεία. Υποστηρίζεται από μια αυστηρά θεμελιωμένη θεωρία που εγγυάται ότι η συμπεριφορά του παραγόμενου κυκλώματος θα είναι ισοδύναμη με αυτή της αρχικής περιγραφής (ισοδυναμία ροής - flow equivalence) [1]. Το κυριότερο πλεονέκτημα της μεθόδου είναι ότι ο σχεδιαστής δεν χρειάζεται καμία γνώση ασύγχρονων κυκλωμάτων. Από το αρχικό κύκλωμα το μόνο που αλλάζει είναι το καθολικό σήμα του ρολογιού που μετατρέπεται σε δίκτυο ελεγκτών. Έτσι, το datapath παραμένει ως έχει και είναι το ίδιο και στην αποσυγχρονισμένη έκδοση.



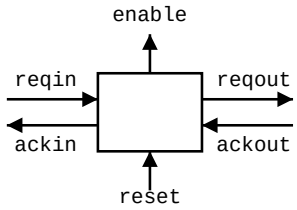
Σχήμα 2.1: Αποσυγχρονισμός ενός κυκλώματος.

Πολύ σημαντικό πλεονέκτημα είναι ότι η μέθοδος είναι πλήρως αυτοματοποιήσιμη και η αρχική είσοδος της ροής είναι σύγχρονη περιγραφή. Αυτό μας επιτρέπει να χρησιμοποιήσουμε τα ήδη υπάρχοντα εμπορικά εργαλεία και να μην απαιτείται από τον σχεδιαστή καμία επιπλέον ιδιαίτερη γνώση για ασύγχρονη σχεδίαση. Ακόμα, η λογική συγχρονισμού, δηλαδή η λογική των ελεγκτών, είναι παράλληλη με την υπολογιστική λογική του κυκλώματος κι έτσι δεν επιβαρύνει την ταχύτητα του κυκλώματος. Επίσης οι ελεγκτές είναι αρκετά μικροί,

άρα η επιβάρυνση στο μέγεθος και την κατανάλωση ενέργειας του κυκλώματος είναι πολύ περιορισμένη και σίγουρα όχι μεγαλύτερη από τις αντίστοιχες επιβαρύνσεις του ρολογιού στο σύγχρονο σχεδιασμό. Τέλος, για την υλοποίηση των ελεγκτών δεν απαιτούνται ειδικές πύλες, άρα αυτή η ροή μπορεί να χρησιμοποιηθεί για κάθε τεχνολογία που χρησιμοποιείται ήδη για την σύγχρονη σχεδίαση.

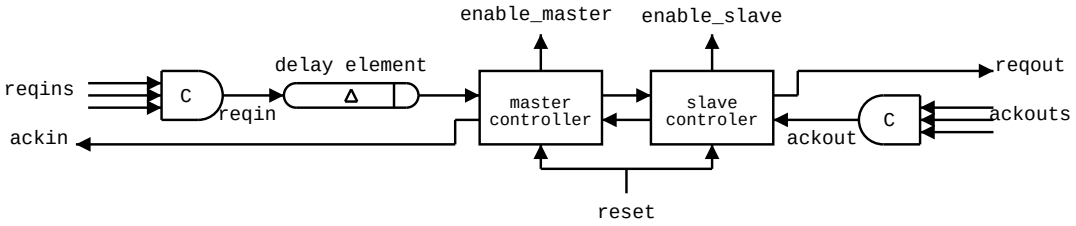
Για την χρήση της συνδυαστικής λογικής χρησιμοποιούνται στοιχεία καθυστέρησης (delay elements) που υλοποιούν ανάλογη καθυστέρηση με αυτή που παρουσιάζει η λογική. Η χρήση στοιχείων καθυστέρησης έχει το πλεονέκτημα ότι ακολουθεί τις διαβαθμίσεις στη συνθήκη λειτουργίας του ολοκληρωμένου. Αυτές είναι οι περιπτώσεις που χαρακτηρίζονται από ομοιογενείς παραμετρικές διαφοροποιήσεις. Για παράδειγμα όταν αυξάνεται η θερμοκρασία λειτουργίας, αυξάνεται και η καθυστέρηση των στοιχείων καθυστέρησης οπότε συνεχίζουν να μοντελοποιούν σωστά την καθυστέρηση της λογικής. Αυτό βέβαια με την προϋπόθεση ότι το στοιχείο καθυστέρησης βρίσκεται κοντά στη λογική μέσα στο ολοκληρωμένο ώστε να δέχονται την ίδια επίδραση. Ιδιαίτερη προσοχή πρέπει να δοθεί στην χρήση περιορισμένου αριθμού τέτοιων στοιχείων καθώς σε περίπτωση εκτεταμένης χρήσης είναι δυνατόν να επιβαρυνθεί σημαντικά το μέγεθος και η κατανάλωση της ενέργειας του κυκλώματος.

Οι ελεγκτές είναι κυκλώματα συγχρονισμού που φροντίζουν για τον σωστό χρονισμό στην μετάδοση των δεδομένων. Επικοινωνούν με άλλους ελεγκτές με κάποιο πρωτόκολλο χειραψίας ώστε να παράγουν σωστά τα σήματα ενεργοποίησης. Η γενική εικόνα ενός ελεγκτή φαίνεται στο Σχήμα 2.2. Το σήμα reqin δηλώνει ότι ο προηγούμενος ελεγκτής-παραγωγός έχει έτοιμα δεδομένα, ενώ το σήμα ackin ότι ο ελεγκτής έχει επεξεργαστεί τα δεδομένα που του ήρθαν. Από την άλλη μεριά, το σήμα reqout δείχνει ότι ο ελεγκτής έχει έτοιμα δεδομένα και το σήμα ackout ότι ο επόμενος ελεγκτής-καταναλωτής έχει επεξεργαστεί τα δεδομένα που του δόθηκαν Το σήμα enable είναι το σήμα ενεργοποίησης που οδηγεί τους μανταλωτές, ενώ το σήμα reset χρησιμοποιείται για την αρχικοποίηση του ελεγκτή κατά την εκκίνηση του κυκλώματος.



Σχήμα 2.2: Γενική μορφή ενός ελεγκτή.

Τα κυκλώματα ελεγκτών που χρησιμοποιούνται σε αυτή την μέθοδο αποτελούνται από πέντε μέρη όπως φαίνεται στο Σχήμα 2.3. Τα δύο κεντρικά μέρη (master/slave controller) είναι ελεγκτές όπως στο Σχήμα 2.2. Αυτοί αποτελούν τον πυρήνα του κυκλώματος και μπορεί να είναι οποιουδήποτε τύπου, αρκεί να ακολουθούν σωστά το πρωτόκολλο χειραψίας. Συνδέονται μεταξύ τους ώστε να συγχρονίζονται και να παράγουν σωστά τα σήματα ενεργοποίησης για τους μανταλωτές. Υπάρχουν επίσης δύο στοιχεία συγχρονισμού (reqins/ackouts synchronizers). Τα στοιχεία συγχρονισμού χρησιμοποιούνται ώστε να συγχρονίζουν πολλαπλά σήματα αιτήσεως. Στα κυκλώματα των ελεγκτών χρησιμοποιούνται για το συγχρονισμό των σημάτων αιτήσεως (reqin) και επιβεβαιώσεως (ackout) προς τον ελεγκτή. Τα σήματα αυτά παράγονται από τους ελεγκτές των ομάδων που παράγουν ή καταναλώνουν δεδομένα από την ομάδα του ελεγκτή αντίστοιχα. Τέλος, χρησιμοποιείται κι ένα στοιχείο καθυστέρησης (delay element) ώστε να μοντελοποιείται ο χρόνος που απαιτείται για την ολοκλήρωση του υπολογισμού από την συνδυαστική λογική της ομάδας του ελεγκτή. Όπως φαίνεται στο σχήμα το στοιχείο καθυστέρησης καθυστερεί την άφιξη του συγχρονισμένου σήματος αιτήσεως (reqin) στον πυρήνα του κυκλώματος ώστε να υπάρχουν έγκυρα δεδομένα την στιγμή που θα ενεργοποιηθεί ο μανταλωτής αφέντης.



Σχήμα 2.3: Σχηματική αναπαράσταση των ελεγκτών που χρησιμοποιούνται στον αποσυγχρονισμό.

2.2 Αλγόριθμος αποσυγχρονισμού

Για τον αποσυγχρονισμό των κυκλωμάτων ακολουθείται ένας αλγόριθμος ο οποίος δέχεται σαν είσοδο μία περιγραφή ενός σύγχρονου κυκλώματος μετά από σύνθεση, δηλαδή ένα κατάλογο πυλών (gate netlist) και καταλήγει με μία αποσυγχρονισμένη έκδοση με ισοδύναμη συμπεριφορά. Τα βήματα που ακολουθεί ο αλγόριθμος είναι τα εξής:

1. Αντικατάσταση καταχωρητών από μανταλωτές

   Κάθε καταχωρητής αντικαθίσταται με ένα ζευγάρι μανταλωτών αφέντη - σκλάβου (master - slave). Αυτό είναι απαραίτητο ώστε τα δεδομένα να μην επαναγράφονται πριν χρησιμοποιηθούν από τους καταναλωτές τους. Έτσι, την στιγμή που εκχωρείται η καινούργια τιμή στο μανταλωτή αφέντη, τα προηγούμενα δεδομένα είναι ήδη αποθηκευμένα στο μανταλωτή σκλάβο για χρήση από τον καταναλωτή τους, όταν αυτός θα είναι έτοιμος να δεχτεί νέα δεδομένα.

2. Ομαδοποίηση των στοιχείων του κυκλώματος

   Εφαρμόζεται μία μέθοδος ομαδοποίησης και παράγονται οι ομάδες της συνδυαστικής λογικής. Στην κάθε ομάδα εισάγονται κατά ζεύγη οι μανταλωτές που οδηγούνται από την λογική της ομάδας. Έπειτα, για κάθε ομάδα δημιουργείται ένας ελεγκτής. Τα στοιχεία καθυστέρησης των ελεγκτών αντιστοιχούν στην καθυστέρηση της λογικής της ομάδας τους. Η ομαδοποίηση είναι απαραίτητη ώστε να ελαχιστοποιηθεί το πλήθος των ελεγκτών που θα χρησιμοποιηθούν. Η χρήση πολλών ελεγκτών οδηγεί σε επιβάρυνση της ταχύτητας, της κατανάλωσης και του μεγέθους του κυκλώματος, λόγω της επιπλέον λογικής χειραψίας (συγχρονισμού).

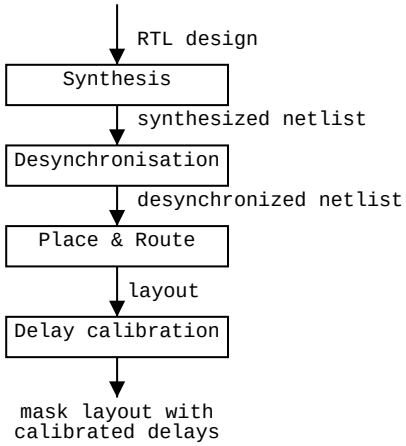
   Η μέθοδος με την οποία μπορεί να γίνει η ομαδοποίηση ποικίλει. Μπορεί να γίνεται χειροκίνητα ή να χρησιμοποιείται κάποιος αλγόριθμος ώστε να γίνεται αυτόματα. Οποιαδήποτε μέθοδος κι αν ακολουθηθεί θα πρέπει η ομαδοποίηση να γίνει με αποτελεσματικό τρόπο ώστε να βρεθεί την "χρυσή τομή" στο πλήθος των ομάδων. Πάρα πολλές ομάδες συνεπάγονται μεγάλη επιβάρυνση στην ταχύτητα, κατανάλωση και το μέγεθος του κυκλώματος από το μεγάλο πλήθος των ελεγκτών. Πολύ λίγες ομάδες οδηγούν σε περιορισμένη παραλληλία.

3. Δημιουργία δικτύου ελέγχου

   Τα σήματα ενεργοποίησης του κάθε ελεγκτή χρησιμοποιούνται για την ενεργοποίηση των μανταλωτών της ομάδας του. Έπειτα, οι ελεγκτές συνδέονται μεταξύ τους σύμφωνα με τις εξαρτήσεις των δεδομένων ανάμεσα στις ομάδες. Αυτές οι συνδέσεις επιτυγχάνουν το συγχρονισμό τους με τη χρήση του πρωτοκόλλου χειραψίας ώστε να εξασφαλίζεται η σωστή χρονικά μετάδοση των δεδομένων. Έτσι, το καθολικό σήμα του ρολογιού αντικαθίσταται από το δίκτυο των τοπικών ελεγκτών.

2.3 Αυτόματη ροή για αποσυγχρονισμό

Η παρούσα εργασία εστιάζεται στην δημιουργία μιας αυτόματης ροής για αποσυγχρονισμό κυκλωμάτων. Βασίζεται εξολοκλήρου σε εμπορικά εργαλεία για σύνθεση και για τοποθέτηση και διασύνδεση - ΤΚΔ (Place and Route - PNR). Η ροή ξεκινά από σύγχρονο netlist



Σχήμα 2.4: Τα στάδια της αυτόματης ροής αποσυγχρονισμού.

σε Verilog και τελειώνει σε διάταξη (layout). Η ακολουθία των σταδίων της ροής φαίνεται σχηματικά στο Σχήμα 2.4. Τα βήματα είναι τα εξής:

1. Σύνθεση του αρχικού σύγχρονου σχεδίου (design)

   Το αρχικό σύγχρονο σχέδιο δίνεται στο εργαλείο σύνθεσης Synopsys Design Compiler (Synopsys DC) ώστε να παραχθεί netlist σε επίπεδο πυλών.

2. Αποσυγχρονισμός

   Το σύγχρονο netlist που έχει παραχθεί από το προηγούμενο βήμα δίνεται στο εργαλείο αποσυγχρονισμού. Το εργαλείο αυτό αποτελεί το κυριότερο μέρος της υλοποίησης της ροής. Υλοποιεί τα βήματα του αποσυγχρονισμού που αναφέρθηκαν στην προηγούμενη ενότητα και τελικά παράγει το αποσυγχρονισμένο netlist που περιέχει τους ελεγκτές και είναι έτοιμο για ΤΚΔ.

   Ιδιαίτερη έμφαση έχει δοθεί στη μέθοδο της ομαδοποίησης. Παρέχονται δυο διαφορετικοί αλγόριθμοι για την αυτόματη ομαδοποίηση της λογικής που μπορούν να συνδυαστούν και μεταξύ τους. Ο ένας βασίζεται στην ιεραρχία του κυκλώματος, ενώ ο δεύτερος στις συνδέσεις μεταξύ των πυλών. Έτσι, το εργαλείο παρέχει την επιλογή του αλγορίθμου για βέλτιστα αποτελέσματα ανάλογα με τις ιδιαιτερότητες του κάθε κυκλώματος.

3. Τοποθέτηση και διασύνδεση

   Το αποσυγχρονισμένο netlist από το προηγούμενο βήμα εισάγεται στο εργαλείο ΤΚΔ SOC-Encounter. Επίσης, εισάγονται και οι περιγραφές των σημάτων ενεργοποίησης των ελεγκτών ως εικονικά ρολόγια, έτσι ώστε να φτιαχτούν τα κατάλληλα δέντρα ενίσχυσης (clock trees). Το εργαλείο παράγει την διάταξη από την οποία γνωρίζουμε τις πραγματικές καθυστερήσεις του κυκλώματος.

4. Διόρθωση των στοιχείων καθυστέρησης (delay calibration)

   Στο στάδιο του αποσυγχρονισμού υποθέτουμε ότι τα σήματα ενεργοποίησης από τους ελεγκτές είναι ιδανικά καλώδια (ideal nets). Στην πραγματικότητα όμως τα δέντρα ενίσχυσης καθυστερούν την άφιξη των σημάτων ενεργοποίησης στους μανταλωτές. Οι τιμές καθυστέρησης για το κάθε δέντρο ενίσχυσης του κυκλώματος δίνονται από το εργαλείο ΤΚΔ.

   Αυτές οι καθυστερήσεις συνεπάγονται ότι τα στοιχεία καθυστέρησης χρειάζονται διόρθωση ώστε να υλοποιούν την πραγματική απαιτούμενη καθυστέρηση. Σε διαφορετική περίπτωση τα δεδομένα είναι δυνατόν να είναι διαθέσιμα από την λογική είτε μετά, είτε πολύ πριν το σήμα ενεργοποίησης. Στην πρώτη περίπτωση δεν λειτουργεί σωστά το κύκλωμα ενώ στη δεύτερη μειώνεται άσκοπα η απόδοσή του. Έτσι για κάθε ομάδα υπολογίζεται η καινούργια τιμή του στοιχείου καθυστέρησης και διορθώνεται στην τελική διάταξη.

Σε αυτό το κεφάλαιο δόθηκε η περιγραφή του αποσυγχρονισμού ως μίας μεθόδου για την αυτόματη υλοποίηση ασύγχρονων κυκλωμάτων από σύγχρονη περιγραφή. Επίσης αναλύθηκαν τα στάδια του αλγορίθμου που χρησιμοποιείται για την υλοποίηση των αποσυγχρονισμένων κυκλωμάτων. Τέλος, δόθηκε η γενική περιγραφή της αυτόματης ροής που χρησιμοποιεί την μέθοδο του αποσυγχρονισμού η οποία ξεκινάει από σύγχρονη περιγραφή ενός κυκλώματος και καταλήγει σε αποσυγχρονισμένη διάταξή του. Στο επόμενο κεφάλαιο περιγράφεται λεπτομερώς η υλοποίηση του κάθε σταδίου αυτής της ροής.

Κεφάλαιο 3

Υλοποίηση της ροής

Στο προηγούμενο κεφάλαιο περιγράφηκε η μέθοδος του αποσυγχρονισμού και τα στάδια της ροής που υλοποιεί τη μέθοδο. Σε αυτό το κεφάλαιο θα δοθούν όλες οι τεχνικές λεπτομέρειες για την υλοποίηση της ροής.

Για την υλοποίηση της ροής χρησιμοποιήθηκαν scripts σε Bourne shell, σε awk και προγράμματα γραμμένα σε C. Η ροή έχει ως είσοδο την σύγχρονη περιγραφή ενός κυκλώματος και ως έξοδο την τελική περιγραφή της διάταξης του αποσυγχρονισμένου κυκλώματος μετά από ΤΚΔ. Η ροή περιλαμβάνει τα στάδια που περιγράφονται παρακάτω.

1. Επεξεργασία της βιβλιοθήκης της τεχνολογίας

   Τα προγράμματα της ροής χρειάζονται να ξέρουν τις ιδιότητες της τεχνολογίας που χρησιμοποιείται. Έτσι χρησιμοποιούμε το πρόγραμμα libgates το οποίο διαβάζει το αρχείο-βιβλιοθήκη της τεχνολογίας και παράγει ένα αρχείο περιγραφής των πυλών που ονομάζεται gatefile. Το αρχείο αυτό περιέχει πληροφορίες για τον κάθε τύπο πύλης και ένα δείγμα του φαίνεται στο Παράρτημα Α′. Οι δύο πρώτες γραμμές περιγράφουν τον τρόπο σύνδεσης των μανταλωτών που αντικαθιστούν τους καταχωρητές. Έπειτα ακολουθούν οι περιγραφές για κάθε τύπο πύλης. Περιγράφονται οι θύρες (ports) της πύλης, δηλώνοντας τον τύπο (είσοδοι, έξοδοι, ρολόγια) και το όνομα της καθεμιάς. Σε περίπτωση αντικαταστάσιμων καταχωρητών δίνονται και οι τύποι του ζευγαριού των μανταλωτών που χρησιμοποιούνται για την αντικατάσταση. Το πρόγραμμα αυτό παράγει επίσης ένα αρχείο με τους καταχωρητές που δεν μπορούμε να αντικαταστήσουμε, έτσι ώστε να αποτρέψουμε την χρήση τους στο Synopsys DC (set_dont_use). Η εκτέλεση αυτού του σταδίου απαιτείται μόνο κάθε φορά που αλλάζουμε τεχνολογία.

2. Σύνθεση του κυκλώματος με περιορισμούς

   Η ροή δέχεται μια σύγχρονη περιγραφή κυκλώματος. Αυτή η περιγραφή εισάγεται στο εργαλείο σύνθεσης Synopsys DC ώστε να παραχθεί το netlist σε επίπεδο πυλών. Επίσης, δίνονται και οι περιορισμοί του προηγούμενου βήματος για την αποτροπή χρήσης αναντικατάστατων καταχωρητών. Η έξοδος από το εργαλείο σύνθεσης δίνεται σαν είσοδος στο εργαλείο αποσυγχρονισμού.

3. Αποσυγχρονισμός του κυκλώματος

   Τον αποσυγχρονισμό του κυκλώματος αναλαμβάνει το εργαλείο αποσυγχρονισμού. Αυτό αποτελεί το μεγαλύτερο προγραμματιστικά κομμάτι της ροής και χρειάστηκαν περίπου 3000 γραμμές κώδικα C για την υλοποίησή του. Δέχεται ως είσοδο σύγχρονο netlist και παράγει ως έξοδο το αποσυγχρονισμένο netlist σε γλώσσα Verilog. Για την σωστή λειτουργία του εργαλείου, η είσοδός του πρέπει να ικανοποιεί τις παρακάτω προϋποθέσεις.

   - Το netlist πρέπει να είναι σε γλώσσα Verilog
   - Η κάθε μονάδα (module) θα πρέπει να υλοποιείται πριν χρησιμοποιηθεί, άρα ως κορυφαία (top-level) θεωρείται η τελευταία μονάδα του αρχείου

- Το σήμα του ρολογιού θα πρέπει να υλοποιείται ως ιδανικό καλώδιο, δηλαδή να μην μεσολαβεί καμία πύλη ανάμεσα στην είσοδο του στο κύκλωμα και στους καταχωρητές που οδηγεί

Τα εσωτερικά βήματα που ακολουθεί το εργαλείο αποσυγχρονισμού είναι τα παρακάτω:

(α΄) Ανάγνωση του αρχείου περιγραφής των πυλών
Το πρόγραμμα διαβάζει το αρχείο gatefile και δημιουργεί τις δομές για τις πληροφορίες των πυλών καθώς και τους κανόνες αντικαταστάσεις των καταχωρητών.

(β΄) Λεξικογραφική ανάλυση του αρχείου εισόδου
Η λεξικογραφική ανάλυση του αρχείου εισόδου γίνεται με τη βοήθεια του προγράμματος λεκτικής ανάλυσης flex. Στο αρχείο lex.l περιγράφεται η μορφή των συμβόλων (tokens) της γλώσσας Verilog για χρήση με τον συντακτικό αναλυτή.

(γ΄) Συντακτική ανάλυση του αρχείου εισόδου
Η συντακτική ανάλυση του αρχείου εισόδου γίνεται με τη βοήθεια του προγράμματος συντακτικής ανάλυσης yacc. Στο αρχείο yacc.y περιγράφεται η σύνταξη της γλώσσας Verilog και καλούνται οι απαραίτητες συναρτήσεις για την επεξεργασία του κυκλώματος.

(δ΄) Δημιουργία του γράφου του κυκλώματος
Μετά την ολοκλήρωση της συντακτικής ανάλυσης έχουν δημιουργηθεί όλες οι δομές δεδομένων που θα χρησιμοποιηθούν αργότερα. Σε αυτές τις δομές περιλαμβάνεται και ο γράφος που αναπαριστά το κύκλωμα. Σε αυτό το γράφο κάθε κόμβος αναπαριστά μία πύλη και κάθε ακμή μία σύνδεση ανάμεσα σε δύο πύλες. Ο γράφος φτιάχνεται ιεραρχικά για κάθε μονάδα του κυκλώματος όπως καθορίζεται στο αρχείο περιγραφής του κυκλώματος.

(ε΄) Αντικατάσταση των καταχωρητών με ζευγάρια μανταλωτών
Κάθε καταχωρητής μετατρέπεται σε ένα ζευγάρι μανταλωτών αφέντη - σκλάβου. Ο τύπος που θα χρησιμοποιηθεί για τον κάθε μανταλωτή καθώς και η αντικατάσταση των συνδέσεων γίνεται σύμφωνα με τις πληροφορίες του αρχείου περιγραφής των πυλών.

(στ΄) Ομαδοποίηση της λογικής του κυκλώματος
Αυτό είναι το πιο σημαντικό στάδιο του αλγορίθμου, καθώς όπως αναλύσαμε στο προηγούμενο κεφάλαιο, παίζει σημαντικό ρόλο στην ποιότητα (απόδοση, κατανάλωση ενέργειας, μέγεθος) του τελικού κυκλώματος. Για την ομαδοποίηση υλοποιήθηκαν οι εξής αλγόριθμοι:

- Ομαδοποίηση σύμφωνα με την ιεραρχία
  Σύμφωνα με αυτό τον τρόπο κάθε μονάδα του κυκλώματος γίνεται και μία ξεχωριστή ομάδα που περιέχει όλες τις πύλες της. Σε περίπτωση που υπάρχουν υπομονάδες (submodules) τότε αυτά τοποθετούνται στην ομάδα της μονάδας μόνο στην περίπτωση που αποτελούνται μόνο από συνδυαστική λογική. Σε διαφορετική περίπτωση αποτελούν ξεχωριστή ομάδα.
  Όπως φαίνεται αυτός ο τρόπος βασίζεται στην ιεραρχία του κυκλώματος από τον σχεδιαστή. Δηλαδή ο σχεδιαστής δηλώνει άμεσα τον τρόπο ομαδοποίησης και κατ΄ επέκταση τα σημεία που εισάγονται οι ελεγκτές μανταλωτών τοποθετώντας τα στοιχεία της κάθε ομάδας σε ξεχωριστή μονάδα κατά την αρχική περιγραφή του κυκλώματος του.
- Ομαδοποίηση σύμφωνα με τις συνδέσεις
  Αρχικά ο αλγόριθμος ισοπεδώνει την ιεραρχία του κυκλώματος και έπειτα αφαιρεί τους μανταλωτές από τον γράφο του κυκλώματος. Κάθε συνδεδεμένος υπογράφος αποτελεί και μια ομάδα λογικής. Στη συνέχεια, κάθε ζευγάρι μανταλωτών εισάγεται στην ομάδα της λογικής από την οποία οδηγείται. Στο Σχήμα 3.1 φαίνεται ο αλγόριθμος ομαδοποίησης. Το όρισμα G είναι ο γράφος του κυκλώματος.
  Αυτός ο αλγόριθμος δεν βασίζεται στο σχεδιαστή κι έτσι μειώνουμε τις απαιτήσεις από τους χρήστες της ροής. Επίσης με αυτό τον αλγόριθμο

```
CONNECTIONGROUPING(G)
 1  flatten_graph(G)
 2  G_comb ← null
 3  for each node in G
 4  do
 5     group(node) ←-1
 6     if type(node) ≠ register
 7       then
 8            G_comb ← G_comb ∪ node
 9            for each target in targets(node)
10            do
11               if type(target) ≠ register
12                 then
13                      G_comb ← G_comb ∪ (node, target)
14
15  count ← 1
16  for each cc in connected_components(G_comb)
17  do
18     for each node in cc
19     do
20        group(node) ← count
21     count ← count + 1
22
23  continue ← 1
24  while continue = 1
25  do
26     continue ← 0
27     for each node in G
28     do
29        if type(node) = registerandgroup(node) =-1
30          then
31               source ← source(node)
32               if source = null
33                 then
34                      group(node) ← 0
35                 else
36                      if type(node) ≠ register
37                        then
38                             group(node) ← group(source)
39                        else
40                             if group(source) ≠-1
41                               then
42                                    group(node) ← group(source) + count
43                               else
44                                    continue ← 1
```

Σχήμα 3.1: Αλγόριθμος ομαδοποίησης σύμφωνα με τις συνδέσεις.

μπορούν να φανούν τα ανεξάρτητα κομμάτια της λογική του κυκλώματος. Έτσι το τελικό κύκλωμα είναι δυνατόν να έχει μεγαλύτερο βαθμό παραλληλίας από όσο θα είχε υποθέσει αρχικά ο σχεδιαστής.

- Συνδυασμός
  Υπάρχει η δυνατότητα συνδυασμού των δύο αλγορίθμων. Ο συνδυαστικός αυτός αλγόριθμος εκτελεί ομαδοποίηση σύμφωνα με τις συνδέσεις σε κάθε ομάδα που προκύπτει από την ομαδοποίηση σύμφωνα με την ιεραρχία. Το κύριο πλεονέκτημα αυτού του αλγορίθμου είναι ότι συνδυάζει τα πλεονεκ-

τήματα της ομαδοποίησης σύμφωνα με τις συνδέσεις χωρίς να επηρεάζεται από συνδέσεις μεταξύ ομάδων λογικής. Αυτό είναι πολύ σημαντικό καθώς δεν επηρεάζεται από λογική ενίσχυσης. Για παράδειγμα το κύκλωμα ενίσχυσης του καθολικού reset σήματος μπορεί να οδηγήσει τον αλγόριθμο ομαδοποίησης σύμφωνα με τις συνδέσεις στο να εισάγει όλη την λογική του κυκλώματος σε μία ομάδα.

(ζ΄) Υπολογισμός της καθυστέρησης της λογικής

Αφού έχουμε βρει τις ομάδες υπολογίζουμε την καθυστέρηση της συνδυαστικής λογικής της καθεμίας. Για να γίνει αυτό, καλείται ένα εξωτερικό πρόγραμμα μέσα από το εργαλείου αποσυγχρονισμού και το οποίο ονομάζεται compute_critical. Σε αυτό το πρόγραμμα δίνεται ένα αρχείο Verilog, που παράγεται αυτόματα από το εργαλείο αποσυγχρονισμού και περιέχει μόνο την συνδυαστική λογική της εν λόγω ομάδας. Το compute_critical συνδέεται με το εργαλείο SOC-encounter το οποίο αναλύει χρονικά το κύκλωμα εκτελώντας στατική χρονική ανάλυση (static timing analysis), έπειτα επεξεργάζεται το αρχείο αναφοράς που παράγεται και επιστρέφει το μέγεθος του απαιτούμενου στοιχείου καθυστέρησης για την υλοποίηση της καθυστέρηση της λογικής.

(η΄) Δημιουργία του γράφου εξαρτήσεων

Στη συνέχεια φτιάχνουμε τον γράφο που δείχνει τις εξαρτήσεις των δεδομένων ανάμεσα στις ομάδες. Οι κόμβοι του γράφου αναπαριστούν τις ομάδες του κυκλώματος και οι ακμές του τις εξαρτήσεις δεδομένων ανάμεσα στις ομάδες. Αυτός ο γράφος είναι απαραίτητος για την κατασκευή των ελεγκτών καθώς και για τον τρόπο σύνδεσης μεταξύ τους.

(θ΄) Δημιουργία των ελεγκτών

Έπειτα δημιουργούνται οι ελεγκτές, ένας για κάθε ομάδα του κυκλώματος. Για κάθε ελεγκτή απαιτούνται τρεις παράμετροι. Το μέγεθος του στοιχείου καθυστέρησης και το πλήθος των σημάτων αιτήσεως (reqin) και επιβεβαίωσης (ackout) προς αυτόν. Το μέγεθος του στοιχείου καθυστέρησης δίνεται από την καθυστέρηση της λογικής που υπολογίστηκε πριν. Το πλήθος των σημάτων αιτήσεως ισούται με το πλήθος των ομάδων λογικής που παράγουν δεδομένα για την ομάδα του ελεγκτή, ενώ των σημάτων επιβεβαίωσης με το πλήθος των αντίστοιχων ομάδων που καταναλώνουν δεδομένα από τον ελεγκτή. Οι δυο αυτοί αριθμοί εξάγονται άμεσα από το γράφο εξαρτήσεων. Σε αυτές τις ομάδες συμπεριλαμβάνεται και η επικοινωνία με τον κόσμο εκτός κυκλώματος. Γνωρίζοντας αυτές τις τιμές το εργαλείο αποσυγχρονισμού καλεί ένα εξωτερικό πρόγραμμα με ορίσματα τις τρεις αυτές τιμές.

Το υπεύθυνο για την υλοποίηση των ελεγκτών πρόγραμμα ονομάζεται create_controller. Για την υλοποίηση των ελεγκτών καλεί τα προγράμματα cgate_create και matched_delay_create για την κατασκευή των στοιχείων συγχρονισμού και καθυστέρησης αντίστοιχα. Ο πυρήνας των ελεγκτών είναι κοινός για όλους τους ελεγκτές. Για την κατασκευή των απαιτουμένων στοιχείων τα προγράμματα παράγουν τις απαιτούμενες περιγραφές σε Verilog κι έπειτα συνδέονται με το εργαλείο σύνθεσης Synopsys DC και έτσι δημιουργείται το netlist για την κάθε περιγραφή.

Στο τέλος του προγράμματος έχουν παραχθεί τα αρχεία controllers.v, cgates_mapped.v και delems_mapped.v που περιέχουν την υλοποίηση των απαιτούμενων από τους ελεγκτές μονάδων. Όλα αυτά τα αρχεία χρησιμοποιούνται αργότερα στα στάδιο της ΤΚΔ. Τελικά το create_controller μάς επιστρέφει τον τύπο του ελεγκτή που πρέπει να χρησιμοποιηθεί για την συγκεκριμένη ομάδα.

(ι΄) Σύνδεση ελέγχου των στοιχείων μνήμης

Μετά την δημιουργία των ελεγκτών σειρά έχει η σύνδεσή τους στο κύκλωμα. Αυτή περιλαμβάνει τις συνδέσεις με τους μανταλωτές που οδηγεί ο κάθε ελεγκτής καθώς και την επικοινωνία μεταξύ τους. Η σύνδεση με τους μανταλωτές γίνεται μέσω των δυο σημάτων ενεργοποίησης, από ένα για τους μανταλωτές αφέντες και σκλάβους της ομάδας αντίστοιχα. Η σύνδεση με τους ελεγκτές των

άλλων ομάδων γίνεται ως εξής: Για κάθε ακμή στο γράφο εξαρτήσεων το σήμα αιτήσεως (reqout) της ομάδας πηγής συνδέεται στα σήματα αιτήσεως (reqin) της ομάδας προορισμού. Επίσης το σήμα επιβεβαίωσης της ομάδας προορισμού (ackin) εισάγεται στα σήματα επιβεβαίωσης (ackout) της ομάδας πηγής. Έτσι δημιουργείται το δίκτυο ελέγχου που αντικαθιστά το ρολόι.

(ια΄) Εξαγωγή των τελικών αρχείων

Σε αυτό το σημείο το εργαλείο αποσυγχρονισμού έχει ολοκληρώσει την δημιουργία του κυκλώματος. Έτσι εξάγει την περιγραφή του αποσυγχρονισμένο netlist σε ένα Verilog αρχείο. Εξάγεται επίσης και το αρχείο περιγραφής των σημάτων ενεργοποίησης των ελεγκτών ως ρολόγια ώστε να φτιαχτούν τα κατάλληλα δέντρα ενίσχυσης για αυτά. Αυτά τα αρχεία χρησιμοποιούνται στο εργαλείο ΤΚΔ ώστε να δημιουργηθεί η τελική διάταξη του αποσυγχρονισμένου κυκλώματος. Τέλος, το εργαλείο εξάγει και τους βοηθητικούς γράφους του κυκλώματος για επαλήθευση των αποτελεσμάτων του αλγορίθμου. Τα αρχεία που περιέχουν τους γράφους είναι σε μορφή επεξεργάσιμη από το πακέτο αναπαράστασης γράφων graphviz.

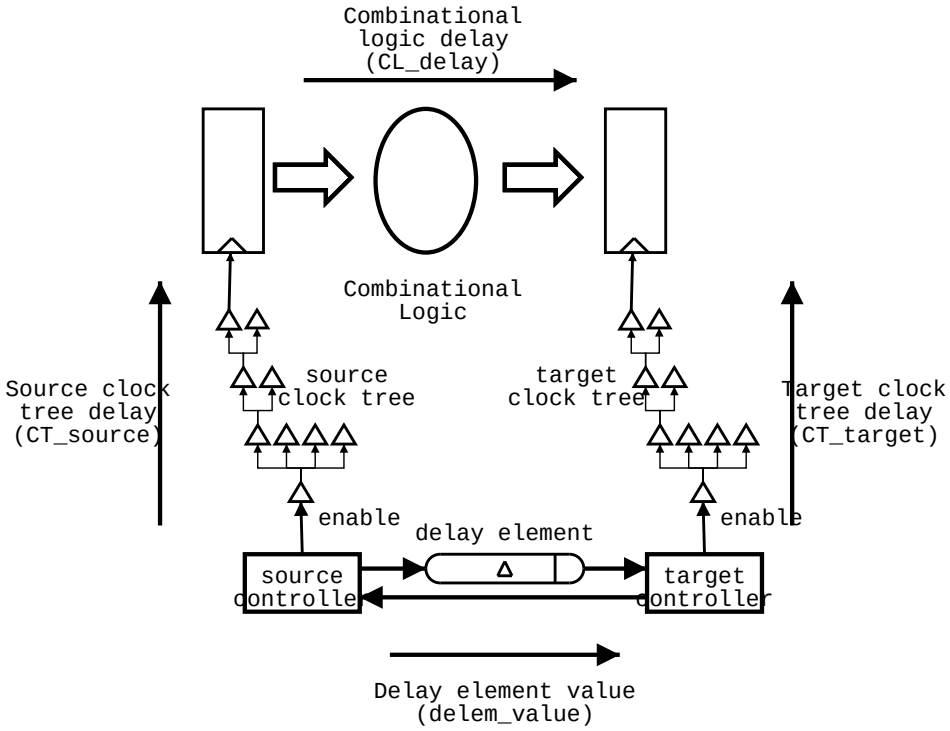
4. Τοποθέτηση και διασύνδεση

Το εργαλείο που χρησιμοποιείται για ΤΚΔ είναι το SOC-Encounter. Σε αυτό δίνονται σαν είσοδος τα αρχεία που παρήγαγε το εργαλείο αποσυγχρονισμού. Αυτά περιέχουν την υλοποίηση του αποσυγχρονισμένου κυκλώματος και των ελεγκτών καθώς επίσης και την περιγραφή των σημάτων ελέγχου των στοιχείων μνήμης. Το SOC-Encounter έτσι μπορεί να παράγει την τελική διάταξη του κυκλώματος.

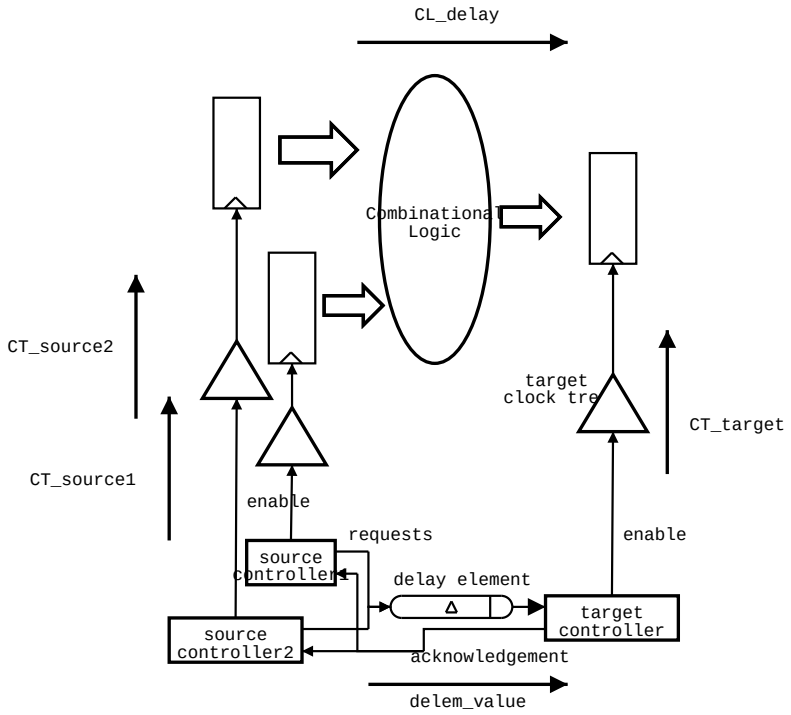
5. Διόρθωση των στοιχείων καθυστέρησης

Αφού έχει παραχθεί η τελική διάταξη, έχουμε τις πραγματικές καθυστερήσεις του κάθε δέντρου ενίσχυσης για τα σήματα ενεργοποίησης. Έτσι υπολογίζουμε τις τιμές διόρθωσης για το κάθε στοιχείο καθυστέρησης. Για κάθε δέντρο ρολογιού αφέντη κρατάμε την ελάχιστη τιμή που θα μας δώσει το εργαλείο, ενώ για τα αντίστοιχα δέντρα σκλάβου την μέγιστη. Η εξίσωση για τον υπολογισμό των στοιχείων καθυστέρησης είναι $CT\_source + CL\_delay < delem\_value + CT\_target$ (Σχήμα 3.2) [1]. Η τιμή της διόρθωσης δίνεται από τον τύπο $new\_delem\_value = delem\_value + max(CT\_source) - CT\_target$ (Σχήμα 3.3). Οι μανταλωτές πηγής είναι τύπου σκλάβου και οι μανταλωτές προορισμού τύπου αφέντη. Μετά την εύρεση των τιμών των απαιτούμενων διορθώσεων, παράγεται ένα αρχείο διόρθωσης της διάταξης. Αυτό το αρχείο μπορεί να χρησιμοποιήσει τη διαδικασία Engineering Change Order (ECO) του εργαλείου για την διόρθωση της διάταξης και περιέχει την εισαγωγή ή αφαίρεση πυλών από αυτήν.

[1] Η καθυστέρηση G to Q θεωρείται αμελητέα



Σχήμα 3.2: Ο τρόπος υπολογισμού των στοιχείων καθυστερήσεως.



Σχήμα 3.3: Ο τρόπος διόρθωσης των στοιχείων καθυστέρησης στην τελική διάταξη.

Κεφάλαιο 4

Πειραματικά Παραδείγματα

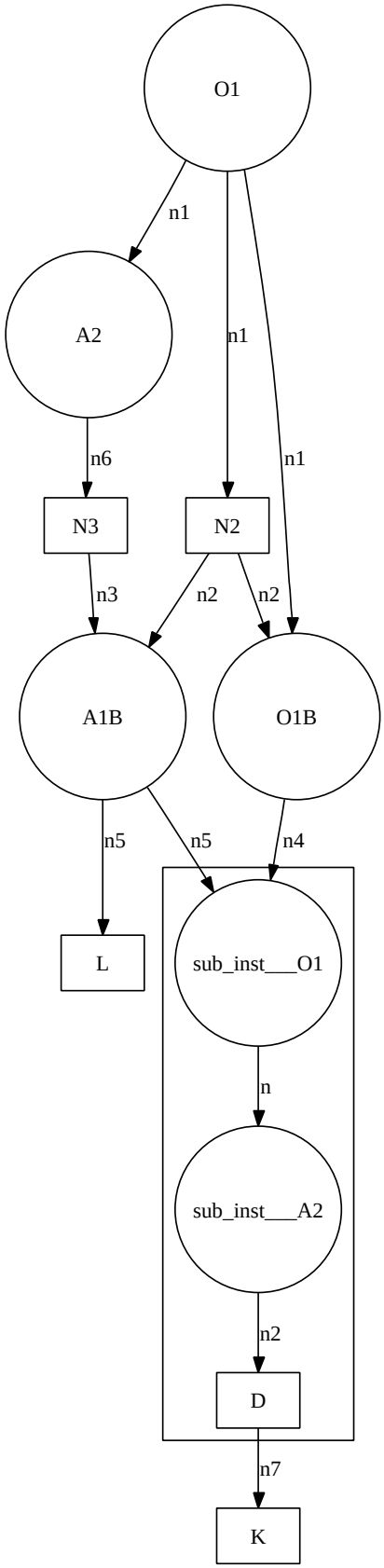
Στο προηγούμενο κεφάλαιο δόθηκαν οι λεπτομέρειες υλοποίησης της ροής. Αναλύθηκαν τα στάδια που ακολουθούνται καθώς και ο τρόπος υλοποίησης του καθενός. Σε αυτό το κεφάλαιο θα παρουσιαστούν τα πειραματικά αποτελέσματα από τη χρήση της ροής. Τα παραδείγματα που χρησιμοποιήθηκαν ήταν αρχικά μικρά ώστε να φαίνονται και να διορθώνονται τα άμεσα παρατηρήσιμα σφάλματα.

4.1 Αρχικός έλεγχος ορθότητας

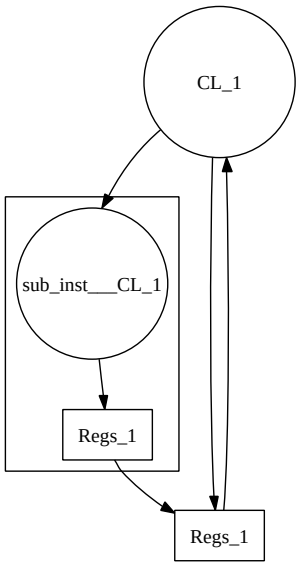
Αρχικά ο έλεγχος ορθότητας (verification) επικεντρώθηκε στο εργαλείο συγχρονισμού καθώς είναι το μεγαλύτερο προγραμματιστικά μέρος της ροής με αποτελέσματα την ύπαρξη αρκετών σφαλμάτων. Ο έλεγχος αυτός έγινε με την χρήση παραδειγμάτων κυκλωμάτων περιορισμένης κλίμακας. Στο Σχήμα 4.1 φαίνεται η περιγραφή ενός τέτοιου παραδείγματος και στο Σχήμα 4.2 η γραφική του αναπαράσταση. Τα αποτελέσματα των διαφορετικών αλγορίθμων ομαδοποίησης φαίνονται στα Σχήματα 4.3, 4.4 και 4.5. Σε κάθε σχήμα αριστερά είναι η γραφική αναπαράσταση και δεξιά τα περιεχόμενα κάθε ομάδας. Για καλύτερη εποπτεία τα στοιχεία συνδυαστικής λογικής φαίνονται ξεχωριστά από τα στοιχεία μνήμης που οδηγούν. Ένα παράδειγμα αποσυγχρονισμένου netlist μπορεί να βρεθεί στο Παράρτημα Β′. Το συγκεκριμένο νετλιστ είναι το αποτέλεσμα του αλγορίθμου ομαδοποίησης σύμφωνα με την ιεραρχία και περιέχει επίσης και την υλοποίηση για όλες τις μονάδες που χρειάζονται οι ελεγκτές.

```
module sub_test (clk, a, b, c, d);

input clk, a, b, c;
output d;

wire n, n2;

OR2D1 O1 ( .Z(n), .A1(a), .A2(b) );
AND2D1 A2 ( .Z(n2), .A1(c), .A2(n) );

DFFPQ1 D ( .Q(d), .CK(clk), .D(n2) );
endmodule

module test( clk, a, b, c, k, l );

input a, b, c, clk;
output k, l;

wire n1, n2, n3, n4, n5, n6, n7;

OR2D1 O1 ( .Z(n1), .A1(a), .A2(b) );
AND2D1 A2 ( .Z(n6), .A1(c), .A2(n1) );

DFFPQ1 N2 ( .Q(n2), .CK(clk), .D(n1) );
DFFPQ1 N3 ( .Q(n3), .CK(clk), .D(n6) );

OR2D1 O1B ( .Z(n4), .A1(n2), .A2(n1) );
AND2D1 A1B ( .Z(n5), .A1(n2), .A2(n3) );

sub_test sub_inst( .clk(clk), .a(n4), .b(n5), .c(c), .d(n7) );

DFFPQ1 K ( .Q(k), .CK(clk), .D(n7) );
DFFPQ1 L ( .Q(l), .CK(clk), .D(n5) );

endmodule
```

Σχήμα 4.1: Το αρχικό σχέδιο του πρώτου παραδείγματος.



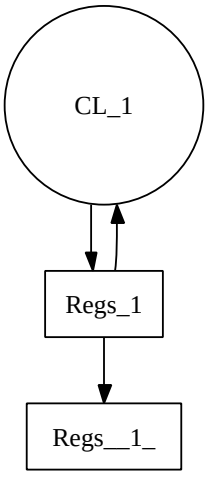
Σχήμα 4.2: Γραφική αναπαράσταση του πρώτου παραδείγματος.



(α′) Γραφική αναπαράσταση

```
Group: CL_1
    A1B
    A2
    O1
    O1B
Group: Regs_1
    K
    L
    N2
    N3
Group: sub_inst___CL_1
    sub_inst___A2
    sub_inst___O1
Group: sub_inst___Regs_1
    sub_inst___D
```

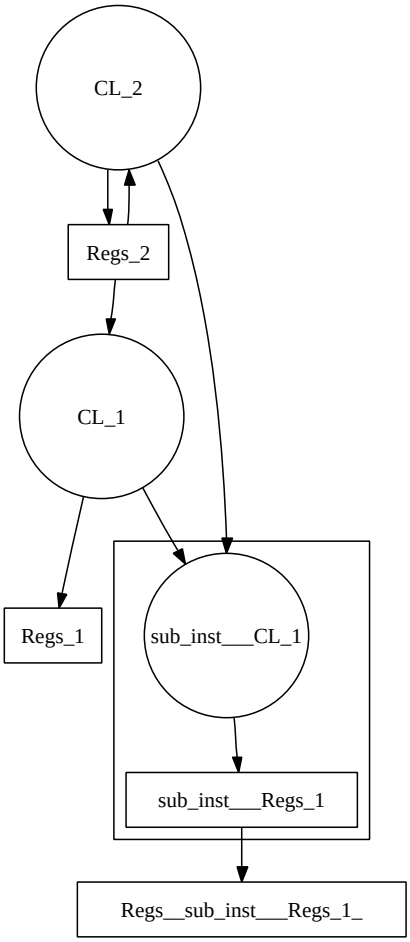
Σχήμα 4.3: Ομαδοποίηση σύμφωνα με την ιεραρχία



(α′) Γραφική αναπαράσταση

```
Group: CL_1
    A1B
    A2
    O1
    O1B
    sub_inst___A2
    sub_inst___O1
Group: Regs_1
    L
    N3
    N2
    sub_inst___D
Group: Regs__1_
    K
```

Σχήμα 4.4: Ομαδοποίηση σύμφωνα με τις συνδέσεις



(α΄) Γραφική αναπαράσταση

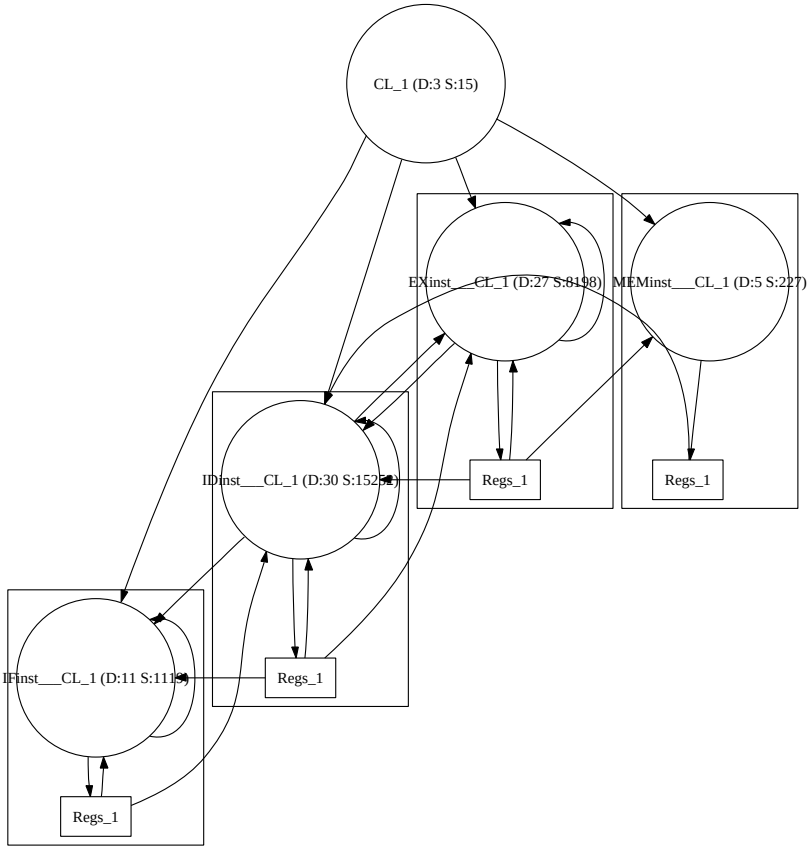
```
Group: CL_1
    A1B
Group: CL_2
    A2
    O1
    O1B
Group: Regs_1
    L
Group: Regs_2
    N3
    N2
Group: Regs__sub_inst___Regs_1_
    K
Group: sub_inst___CL_1
    sub_inst___A2
    sub_inst___O1
Group: sub_inst___Regs_1
    sub_inst___D
```

Σχήμα 4.5: Ομαδοποίηση σύμφωνα με τον συνδυαστικό αλγόριθμο

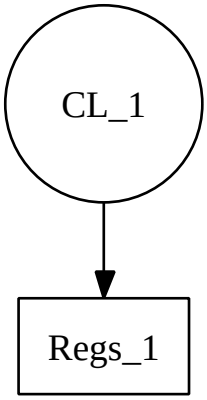
4.2 Δοκιμή με πραγματικό παράδειγμα (**DLX**)

Στην συνέχεια δοκιμάσαμε την ροή με μία σύγχρονη υλοποίηση του DLX. Ο DLX είναι ένας επεξεργαστής που χρησιμοποιείται κυρίως για διδακτικούς σκοπούς σε μαθήματα αρχιτεκτονικής. Αποτελείται από ένα pipeline πέντε σταδίων: Instruction Fetch (IF), Instruction Decode (ID), Execution (EX), Memory (MEM), WriteBack (WB).

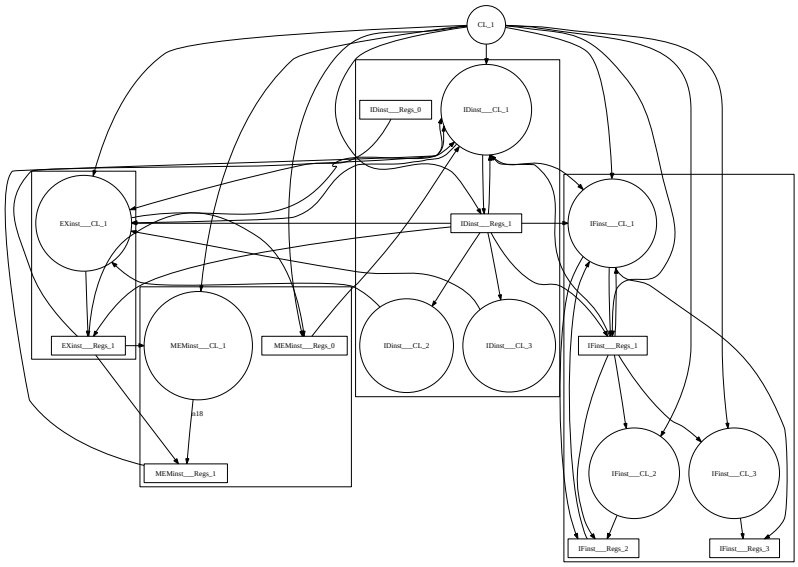
Αφού περάσαμε το σχέδιο από σύνθεση στο εργαλείο Synopsys DC πήραμε το netlist και το περάσαμε από το εργαλείο αποσυγχρονισμού. Οι γράφοι που δείχνουν την ομαδοποίηση για όλους τους αλγορίθμους ομαδοποίησης φαίνονται στα Σχήματα 4.6, 4.8 και 4.7.



Σχήμα 4.6: Ομαδοποίηση σύμφωνα με την ιεραρχία



Σχήμα 4.7: Ομαδοποίηση σύμφωνα με τις συνδέσεις



Σχήμα 4.8: Ομαδοποίηση σύμφωνα με τον συνδυαστικό αλγόριθμο

4.2.1 Σχολιασμός των αποτελεσμάτων

Ο κάθε αλγόριθμος ομαδοποίησης παρήγαγε τα αναμενόμενα αποτελέσματα. Αυτό δείχνει ότι η υλοποίηση έγινε σωστά. Παρακάτω σχολιάζεται η καταλληλότητα του κάθε αλγορίθμου για το συγκεκριμένο κύκλωμα.

- Ομαδοποίηση σύμφωνα με την ιεραρχία

  Όπως ήταν αναμενόμενο ο αλγόριθμος που ομαδοποιεί ανάλογα με την ιεραρχία του χρήστη έχει τα αναμενόμενα από τον χρήστη αποτελέσματα. Έχει ομαδοποιήσει κάθε στάδιο του pipeline ξεχωριστά. Αυτό οφείλεται στο ότι το σχέδιο περιέχει το κάθε στάδιο του επεξεργαστή σε μία μονάδα.

- Ομαδοποίηση σύμφωνα με τις συνδέσεις

  Παρατηρούμε ότι ο απλός αλγόριθμος σύμφωνα με τις συνδέσεις εισάγει ολόκληρη την λογική του κυκλώματος σε μία ομάδα. Αυτό οφείλεται στην ύπαρξη κυκλώματος ενίσχυσης για το σήμα reset, που αποτελεί την ομάδα CL_1 στις άλλες δύο αναπαραστάσεις. Επειδή αυτό το κομμάτι λογικής συνδέεται και με όλα τα υπόλοιπα, ο αλγόριθμος θεωρεί ολόκληρη την λογική του κυκλώματος, ως ένα "σύννεφο". Το αποτέλεσμα αυτού του αλγορίθμου δεν είναι καθόλου ικανοποιητικό για το συγκεκριμένο κύκλωμα.

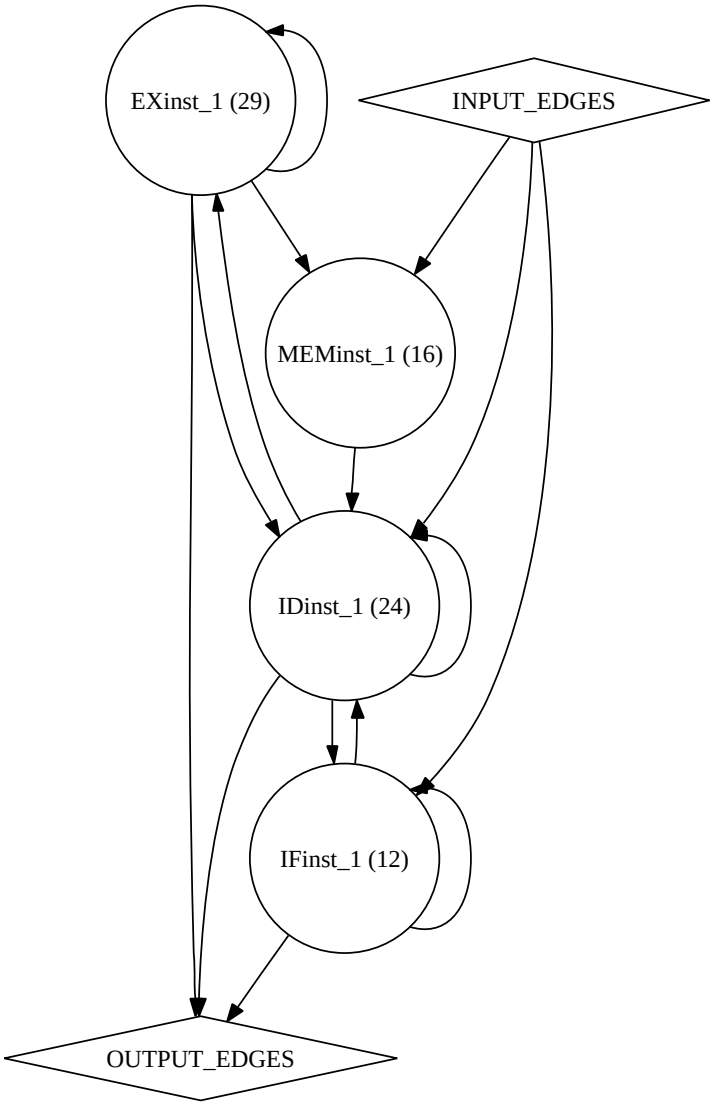
- Ομαδοποίηση με τον συνδυαστικό αλγόριθμο

  Ο συνδυαστικός αλγόριθμος βλέπουμε ότι παράγει παρόμοια αποτελέσματα με τον αλγόριθμο σύμφωνα με την ιεραρχία. Τα αποτελέσματα για τα στάδια MEM και EX είναι ίδια με τον πρώτο αλγόριθμο, ενώ οι επιπλέον ομάδες λογικής ID περιέχουν από μία πύλη ενίσχυσης σήματος (buffer). Για το στάδιο του IF η λογική χωρίζεται σε τρεις ομάδες επειδή η καθεμιά δέχεται διαφορετική έξοδο από την λογική ενίσχυσης του reset. Το συμπέρασμα στο οποίο καταλήγουμε είναι ότι ο συνδυαστικός αλγόριθμος παράγει αρκετά ικανοποιητικά αποτελέσματα που μπορούν να ακολουθήσουν αποτελεσματικά την ιεραρχία του χρήστη. Το συγκεκριμένο κύκλωμα δεν αναδεικνύει την αποτελεσματικότητα του αλγορίθμου στην αξιοποίηση της παραλληλίας του κυκλώματος.

4.2.2 Γράφος εξαρτήσεων

Ο γράφος που δείχνει τις εξαρτήσεις δεδομένων φαίνεται στο Σχήμα 4.9. Οι κόμβοι INPUT_EDGES και OUTPUT_EDGES δείχνουν εξαρτήσεις δεδομένων από το εξωτερικό του

κυκλώματος. Ο γράφος αυτός προκύπτει από το αποτέλεσμα του αλγορίθμου ομαδοποίησης σύμφωνα με την ιεραρχία. Οι αριθμοί μέσα στις παρενθέσεις δείχνουν το μέγεθος του στοιχείου καθυστέρησης της κάθε ομάδας.



Σχήμα 4.9: Οι εξαρτήσεις μεταξύ των ομάδων του DLX.

Παρατηρούμε ότι το στάδιο ID διαβάζει δεδομένα από όλες τις άλλες ομάδες. Από το IF την εντολή προς εκτέλεση, από το EX τον καταχωρητή προορισμού της προηγούμενης εντολής, ενώ από το MEM τις πληροφορίες για την εγγραφή του καταχωρητή προορισμού δύο εντολές νωρίτερα. Το γεγονός αυτό μειώνει την παραλληλία στο σύστημά καθώς το ID περιμένει κάθε άλλη ομάδα πριν συνεχίσει. Το γεγονός αυτό δείχνει ότι πιο "επιθετικές" μέθοδοι θα μπορούσαν να δώσουν καλύτερα αποτελέσματα. Για παράδειγμα, στην ασύγχρονη σχεδίαση του DLX για το Async 2004 είχε χρησιμοποιηθεί ένας επιπλέον καταχωρητής στο στάδιο του ID που κρατούσε τον καταχωρητή προορισμού της προηγούμενης εντολής. Έτσι δεν υπήρχε πια η ακμή εξάρτησης του σταδίου ID από το στάδιο EX και το κύκλωμα αποκτά περισσότερη παραλληλία.

4.2.3 Διόρθωση των στοιχείων καθυστέρησης

Το εργαλείο αποσυγχρονισμού παρήγαγε το αποσυγχρονισμένο κύκλωμα, τα κυκλώματα για τους ελεγκτές και το αρχείο περιγραφής των σημάτων ενεργοποίησης. Όλα αυτά τα χρησιμοποιήσαμε στο εργαλείο SOC-Encounter ώστε να γίνει το ΤΚΔ. Αφού παρήχθη

η διάταξη του κυκλώματος χρησιμοποιήσαμε το εργαλείο διόρθωσης των στοιχείων καθυστέρησης για να δούμε τις αλλαγές που πρέπει να κάνουμε στα στοιχεία καθυστέρησης του κυκλώματος.

Οι καθυστερήσεις των δέντρων ενίσχυσης για τα σήματα ενεργοποίησης του κάθε ελεγκτή φαίνονται στο Σχήμα 4.10. Οι τιμές των δέντρων για τους μανταλωτές αφέντες είναι οι ελάχιστες εκτιμήσεις του εργαλείου ενώ αυτές των δέντρων για τους μανταλωτές σκλάβους οι μέγιστες. Στο Σχήμα 4.11 φαίνονται οι αλλαγές που πρέπει να γίνουν στα στοιχεία καθυστέρησης του κάθε ελεγκτή. Οι τιμές που φαίνονται είναι σε picoseconds (ps). Από τις τιμές των διορθώσεων παράγεται ένα αρχείο με τις ρυθμίσεις για την επιλογή ECO που παρέχεται από το SOC-Encounter. Αυτό το αρχείο δίνεται μαζί με την πρώτη διάταξη στο εργαλείο και παράγεται η τελική διάταξη με διορθωμένα στοιχεία καθυστέρησης.

```
Ctrl__EXinst___Regs_1_master  177
Ctrl__EXinst___Regs_1_slave   187.9
Ctrl__IDinst___Regs_1_master  509.9
Ctrl__IDinst___Regs_1_slave   606.1
Ctrl__IFinst___Regs_1_master  241.1
Ctrl__IFinst___Regs_1_slave   293.2
Ctrl__MEMinst___Regs_1_master 192.2
Ctrl__MEMinst___Regs_1_slave  211.5
```

Σχήμα 4.10: Οι τιμές της καθυστέρησης των δέντρων ενίσχυσης του ρολογιού για κάθε ελεγκτή του DLX.

```
Ctrl__EXinst___Regs_1    delay_calibrate: 606.1 - 177   =  429.1
Ctrl__IDinst___Regs_1    delay_calibrate: 293.2 - 509.9 = -216.7
Ctrl__IFinst___Regs_1    delay_calibrate: 606.1 - 241.1 =  365
Ctrl__MEMinst___Regs_1   delay_calibrate: 187.9 - 192.2 = -4.3
```

Σχήμα 4.11: Οι απαιτούμενες διορθώσεις στα στοιχεία καθυστέρησης για κάθε ελεγκτή.

Κεφάλαιο 5

Συμπεράσματα - Μελλοντική Εργασία

Τα παραδείγματα επιβεβαίωσαν την ορθή λειτουργία της ροής. Τα αποτελέσματα ήταν τα αναμενόμενα και δείχνουν ότι η υλοποίηση έγινε σωστά. Η ευχρηστία και η ενοποίησή της με δημοφιλή εμπορικά εργαλεία συνηγορούν υπέρ της χρήσης της για την εύκολη και αυτόματη δημιουργία ασύγχρονων κυκλωμάτων ξεκινώντας από σύγχρονη περιγραφή, χωρίς να απαιτείται καμία γνώση για ασύγχρονα κυκλώματα.

Η μελλοντική εργασία περιλαμβάνει κυρίως βελτιώσεις στο εργαλείο αποσυγχρονισμού. Κατ΄ αρχήν πρέπει να γίνει πιο αλληλεπιδραστικό με τον χρήστη και να υποστηρίζονται επιλογές στην γραμμή εντολών ώστε να δίνεται περισσότερη πληροφορία από τον χρήστη στο εργαλείο. Αυτές οι πληροφορίες μπορεί να είναι τυπικής φύσεως όπως είναι η αντικατάσταση των τυπικών τιμών κάποιων μεταβλητών, όπως το όνομα του αρχείου εξόδου του τελικού κυκλώματος ή μπορεί να είναι πιο ουσιαστικής φύσεως, όπως για παράδειγμα η δήλωση συνδέσεων που θα πρέπει να αγνοηθούν από τους αλγορίθμους αυτόματης ομαδοποίησης. Περαιτέρω εργασία επίσης πρέπει να γίνει στην καλύτερη διαχείριση μνήμης καθώς το πρόγραμμα είναι αρκετά μνημοβόρο (για την επεξεργασία του DLX απαιτούνται περίπου 200 MB μνήμης).

Οι βελτιώσεις στην μέθοδο έγκεινται κυρίως στην αντικατάσταση των στοιχείων καθυστέρησης με αναγνώριση ολοκλήρωσης ώστε να επωφεληθούμε από την δυναμική συμπεριφορά του κυκλώματος όσον αφορά την παραγωγή των δεδομένων. Αυτό μας παρέχει την γνώση της ακριβής στιγμής της ολοκλήρωσης των υπολογισμών και οδηγεί σε κυκλώματα που όχι μόνο είναι ταχύτερα αλλά και ανεπηρέαστα από φαινόμενα μεταβλητότητας. Τέλος, μπορούν να χρησιμοποιηθούν τεχνικές για την εξάλειψη ακμών από τον γράφο εξαρτήσεων του κυκλώματος ώστε να επιτύχουμε μεγαλύτερη παραλληλία. Αυτές μπορούν να βασίζονται σε διάφορους παράγοντες όπως ο χρονισμός και οι εξαρτήσεις μεταξύ των ομάδων [4] .

Βιβλιογραφία

[1] I. Blunno, J. Cortadella, A. Kondratyev, L. Lavagno, K. Lwin, and C. Sotiriou. Handshake protocols for de-synchronization. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 149–158. IEEE Computer Society Press, Apr. 2004.

[2] J. Cortadella, A. Kondratyev, L. Lavagno, K. Lwin, and C. Sotiriou. From synchronous to asynchronous: an automatic approach. In *Proc. Design, Automation and Test in Europe (DATE)*, pages 1368–1369, Feb. 2004.

[3] J. Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou. A concurrent model for de-synchronization. In *Proc. International Workshop on Logic Synthesis*, pages 294–301, 2003.

[4] A. Davare, K. Lwin, A. Kondratyev, and A. Sangiovanni-Vincentelli. The best of both worlds: the efficient asynchronous implementation of synchronous specifications. In *Proc. ACM/IEEE Design Automation Conference*, pages 588–591, 2004.

Παράρτημα Α΄

Παράδειγμα αρχείου περιγραφής πυλών

```
MASTER D(D) Q(BETWEEN_LATCHES) QB(NULL) G(ENABLE1) GB(ENABLE1) SB(SB) RB(RB)
SLAVE  D(BETWEEN_LATCHES) Q(Q) QB(QB)   G(ENABLE2) GB(ENABLE2) SB(SB) RB(RB)

AND2D1
        IN      A1      A2
        OUT     Z

DFFPQ1  FF
        IN      D
        CLK     CK
        OUT     Q
        MASTER  LATNB1
        SLAVE   LATNB1

DFFRSNB1  FF
        IN      D       RB      SB
        CLK     CKB
        OUT     Q       QB
        MASTER  LATRSPB1
        SLAVE   LATRSPB1

LATNB1  LATCH
        IN      D       G
        OUT     Q       QB

LATRSPB1  LATCH
        IN      D       RB      SB      GB
        OUT     Q       QB

NOR2D1
        IN      A1      A2
        OUT     Z
```

Παράρτημα Β΄

Παράδειγμα **netlist** αποσυγχρονισμένου κυκλώματος

```
module C_gate2 ( in1, in2, out );
input  in1, in2;
output out;
    wire n5, n6, n7, n8, n9, n10, n11;
    INVDL U3 ( .A(n8), .Z(n5) );
    BUFDL U4 ( .A(in2), .Z(n6) );
    INVD2 U5 ( .A(in1), .Z(n8) );
    AOI21D1 U6 ( .A1(n8), .A2(n9), .B(n10), .Z(n7) );
    AOI21D1 U7 ( .A1(n8), .A2(n9), .B(n11), .Z(out) );
    OAI22D1 U8 ( .A1(n6), .A2(n5), .B1(n6), .B2(n7), .Z(n11) );
    OAI22D1 U9 ( .A1(in2), .A2(in1), .B1(in2), .B2(n7), .Z(n10) );
    INVD1 U10 ( .A(n7), .Z(n9) );
endmodule


module matched_delay2 ( in, out );
input  in;
output out;
    wire net1, n1;
    AND2D1 and1 ( .A1(in), .A2(in), .Z(net1) );
    AND2D1 and2 ( .A1(n1), .A2(net1), .Z(out) );
    BUFDL U1 ( .A(in), .Z(n1) );
endmodule


module pulse_delay ( in, out );
input  in;
output out;
    wire net1, net2, net3, net4, net5, net6, net7, net8, net9, net10, net11,
        net12, net13, net14, net15;
    INVD1 inv1 ( .A(in), .Z(net1) );
    INVD1 inv2 ( .A(net1), .Z(net2) );
    INVD1 inv3 ( .A(net2), .Z(net3) );
    INVD1 inv4 ( .A(net3), .Z(net4) );
    INVD1 inv5 ( .A(net4), .Z(net5) );
    INVD1 inv6 ( .A(net5), .Z(net6) );
    INVD1 inv7 ( .A(net6), .Z(net7) );
    INVD1 inv8 ( .A(net7), .Z(net8) );
    INVD1 inv9 ( .A(net8), .Z(net9) );
    INVD1 inv10 ( .A(net9), .Z(net10) );
    INVD1 inv11 ( .A(net10), .Z(net11) );
    INVD1 inv12 ( .A(net11), .Z(net12) );
    INVD1 inv13 ( .A(net12), .Z(net13) );
    INVD1 inv14 ( .A(net13), .Z(net14) );
    INVD1 inv15 ( .A(net14), .Z(net15) );
    INVD1 inv16 ( .A(net15), .Z(out) );
endmodule


module lc_semi_dec_master ( rst, ri, ai, ro, ao, l );
```

```
input  rst, ri, ao;
output ai, ro, l;
    wire n1, n4, n9, nrst, nro, roainp, n7, net129, n8, net136, net137, net142,
        net144, req;
    pulse_delay pd( .in(ri), .out(req));
    OAI211D1 U1 ( .A1(n7), .A2(n9), .B(n8), .C(net129), .Z(ai) );
    INVD1 U2 ( .A(n1), .Z(n7) );
    INVD1 U3 ( .A(net142), .Z(net137) );
    INVD1 U4 ( .A(roainp), .Z(n4) );
    INVD1 U5 ( .A(n9), .Z(nro) );
    INVD1 U6 ( .A(nro), .Z(ro) );
    INVD1 U7 ( .A(rst), .Z(nrst) );
    INVD1 U8 ( .A(req), .Z(n8) );
    NOR2D1 U9 ( .A1(n9), .A2(n7), .Z(net144) );
    NAN2D1 syn82 ( .A1(n1), .A2(net137), .Z(net129) );
    NOR3M1D2 syn16 ( .A1(n8), .A2(net136), .A3(net144), .Z(l) );
    INVD2 U10 ( .A(net129), .Z(net136) );
    BUFD1 U11 ( .A(ao), .Z(net142) );
    AND2D2 norro ( .A1(n4), .A2(ai), .Z(n9) );
    AND2D4 rnand ( .A1(ai), .A2(nrst), .Z(n1) );
    AND2D1 andro ( .A1(nro), .A2(ao), .Z(roainp) );
endmodule


module lc_semi_dec_slave ( rst, ri, ai, ro, ao, l );
input  rst, ri, ao;
output ai, ro, l;
    wire n5, n6, ainrst, nro, roainp, net138, net140, net141, net142, net153,
        n8, n9,req;
    pulse_delay pd( .in(ri), .out(req));
    INVBD2 U1 ( .A(net141), .Z(l) );
    NAN2D2 U2 ( .A1(net138), .A2(net142), .Z(net141) );
    NAN2D2 U3 ( .A1(nro), .A2(n8), .Z(net142) );
    INVD1 U4 ( .A(ainrst), .Z(n8) );
    OAI21M20D2 U5 ( .A1(n8), .A2(n9), .B(net138), .Z(ai) );
    INVD2 U6 ( .A(ro), .Z(nro) );
    INVBD2 syn113 ( .A(net140), .Z(net138) );
    AO21M20D1 net171 ( .A1(ainrst), .A2(net153), .B(req), .Z(net140) );
    BUFD1 U7 ( .A(ao), .Z(net153) );
    INVD1 U8 ( .A(ro), .Z(n9) );
    AND2D4 __tmp51 ( .A1(l), .A2(n5), .Z(ainrst) );
    AND2D2 andro ( .A1(nro), .A2(ao), .Z(roainp) );
    INVBD2 U9 ( .A(roainp), .Z(n6) );
    INVBD2 U10 ( .A(rst), .Z(n5) );
    AND3D2 norro ( .A1(n5), .A2(n6), .A3(ai), .Z(ro) );
endmodule

module controller_d2_r1_a1(reset, en1, en2, ri1, ai, ro, ao1);

input ri1, ao1;
output reset, en1, en2, ai, ro;

wire rx, ax, ri_synchr_delayed, ri_synchr, ao_synchr;

assign ri_synchr = ri;
assign ao_synchr = ao;

matched_delay2 delay (.in(ri_synchr), .out(ri_synchr_delayed));

lc_semi_dec_master master( .rst(reset), .ri(ri_synchr_delayed),
        .ai(ai), .ro(rx), .ao(ax), .l(en1) );
lc_semi_dec_slave  slave ( .rst(reset), .ri(rx), .ai(ax), .ro(ro),
        .ao(ao_synchr), .l(en2) );

endmodule


module controller_d2_r2_a1(reset, en1, en2, ri1, ri2, ai, ro, ao1);

input ri1, ri2, ao1;
output reset, en1, en2, ai, ro;

wire rx, ax, ri_synchr_delayed, ri_synchr, ao_synchr;
```

```
C_gate2 cgate_reqins ( ri1, ri2, ri_synchr );
assign ao_synchr = ao;

matched_delay2 delay (.in(ri_synchr), .out(ri_synchr_delayed));

lc_semi_dec_master master( .rst(reset), .ri(ri_synchr_delayed),
        .ai(ai), .ro(rx), .ao(ax), .l(en1) );
lc_semi_dec_slave  slave ( .rst(reset), .ri(rx), .ai(ax), .ro(ro),
        .ao(ao_synchr), .l(en2) );

endmodule

module sub_test_desync (  a, b, c, d, Ctrl__Regs_1__en1, Ctrl__Regs_1__en2 );

input  a, b, c, Ctrl__Regs_1__en1, Ctrl__Regs_1__en2;
output  d;

wire D__m2s, n, n2;


        AND2D1 A2 ( .A1(c), .A2(n), .Z(n2) );
        LATNB1 D__master ( .D(n2), .G(Ctrl__Regs_1__en1), .Q(D__m2s), .QB() );
        LATNB1 D__slave ( .D(D__m2s), .G(Ctrl__Regs_1__en2), .Q(d), .QB() );
        OR2D1 O1 ( .A1(a), .A2(b), .Z(n) );

endmodule


module test_desync (  a, b, c, k, l, Ctrl__Regs_1__en1, Ctrl__Regs_1__en2,
        Ctrl__sub_inst___Regs_1__en1, Ctrl__sub_inst___Regs_1__en2 );

input  a, b, c, Ctrl__Regs_1__en1, Ctrl__Regs_1__en2,
        Ctrl__sub_inst___Regs_1__en1, Ctrl__sub_inst___Regs_1__en2;
output  k, l;

wire K__m2s, L__m2s, N2__m2s, N3__m2s, n1, n2, n3, n4, n5, n6, n7;


        AND2D1 A1B ( .A1(n2), .A2(n3), .Z(n5) );
        AND2D1 A2 ( .A1(c), .A2(n1), .Z(n6) );
        LATNB1 K__master ( .D(n7), .G(Ctrl__Regs_1__en1), .Q(K__m2s), .QB() );
        LATNB1 K__slave ( .D(K__m2s), .G(Ctrl__Regs_1__en2), .Q(k), .QB() );
        LATNB1 L__master ( .D(n5), .G(Ctrl__Regs_1__en1), .Q(L__m2s), .QB() );
        LATNB1 L__slave ( .D(L__m2s), .G(Ctrl__Regs_1__en2), .Q(l), .QB() );
        LATNB1 N2__master ( .D(n1), .G(Ctrl__Regs_1__en1), .Q(N2__m2s), .QB() );
        LATNB1 N2__slave ( .D(N2__m2s), .G(Ctrl__Regs_1__en2), .Q(n2), .QB() );
        LATNB1 N3__master ( .D(n6), .G(Ctrl__Regs_1__en1), .Q(N3__m2s), .QB() );
        LATNB1 N3__slave ( .D(N3__m2s), .G(Ctrl__Regs_1__en2), .Q(n3), .QB() );
        OR2D1 O1 ( .A1(a), .A2(b), .Z(n1) );
        OR2D1 O1B ( .A1(n2), .A2(n1), .Z(n4) );
        sub_test_desync sub_inst ( .a(n4), .b(n5), .c(c), .d(n7),
                .Ctrl__Regs_1__en1(Ctrl__sub_inst___Regs_1__en1),
                .Ctrl__Regs_1__en2(Ctrl__sub_inst___Regs_1__en2) );

endmodule


module test_desync_with_ctrls (  a, b, c, k, l, Ctrl__reset, Ctrl__Regs_1__ri,
        Ctrl__Regs_1__ai, Ctrl__sub_inst___Regs_1__ri, Ctrl__sub_inst___Regs_1__ai,
        Ctrl__Regs_1__ro, Ctrl__Regs_1__ao );

input  a, b, c, Ctrl__reset, Ctrl__Regs_1__ri, Ctrl__sub_inst___Regs_1__ri,
        Ctrl__Regs_1__ao;
output  k, l, Ctrl__Regs_1__ai, Ctrl__sub_inst___Regs_1__ai, Ctrl__Regs_1__ro;

wire Ctrl__Regs_1__en1, Ctrl__Regs_1__en2, Ctrl__sub_inst___Regs_1__en1,
        Ctrl__sub_inst___Regs_1__en2, Ctrl__sub_inst___Regs_1__ro;


        controller_d2_r2_a1 Ctrl__Regs_1 ( .reset(Ctrl__reset),
                .en1(Ctrl__Regs_1__en1), .en2(Ctrl__Regs_1__en2),
                .ri1(Ctrl__Regs_1__ri), .ri2(Ctrl__sub_inst___Regs_1__ro),
```

```
        .ai(Ctrl__Regs_1__ai), .ro(Ctrl__Regs_1__ro),
        .ao1(Ctrl__Regs_1__ao) );
    controller_d2_r1_a1 Ctrl__sub_inst___Regs_1 ( .reset(Ctrl__reset),
        .en1(Ctrl__sub_inst___Regs_1__en1), .en2(Ctrl__sub_inst___Regs_1__en2),
        .ri1(Ctrl__sub_inst___Regs_1__ri), .ai(Ctrl__sub_inst___Regs_1__ai),
        .ro(Ctrl__sub_inst___Regs_1__ro), .ao1(Ctrl__Regs_1__ai) );
    test_desync test ( .a(a), .b(b), .c(c), .k(k), .l(l),
        .Ctrl__Regs_1__en1(Ctrl__Regs_1__en1),
        .Ctrl__Regs_1__en2(Ctrl__Regs_1__en2),
        .Ctrl__sub_inst___Regs_1__en1(Ctrl__sub_inst___Regs_1__en1),
        .Ctrl__sub_inst___Regs_1__en2(Ctrl__sub_inst___Regs_1__en2) );

endmodule
```