Αρχιτεκτονική Υπολογιστών

Μέρος 3ο :Tomasulo + Reorder Buffer

Κωνσταντίνος Κασφίκης – 2013030108
Νίκος Κανάκης – 2013030175

Εισαγωγή

Ζητούμενο της παρούσας εργαστηριακής άσκησης είναι η προσθήκη στον pipelined Tomasulo επεξεργαστή ενός ReorderBuffer, αλλάζοντας παράλληλα το υπάρχον Register File. Το Register File πλέον καταχωρεί μόνο δεδομένα (χωρίς tag) που λαμβάνει από τον πρώτο καταχωρητή του ReorderBuffer (δεν περιμένει κοινοποιήσεις από το CDB). Σκοπός του Reorder Buffer στην παρούσα υλοποίηση είναι η in order εκτέλεση των εντολών.

Reorder Buffer

Στην υλοποίηση μας ο ReorderBuffer αποτελείται από 7 καταχωρητές των 43 bit

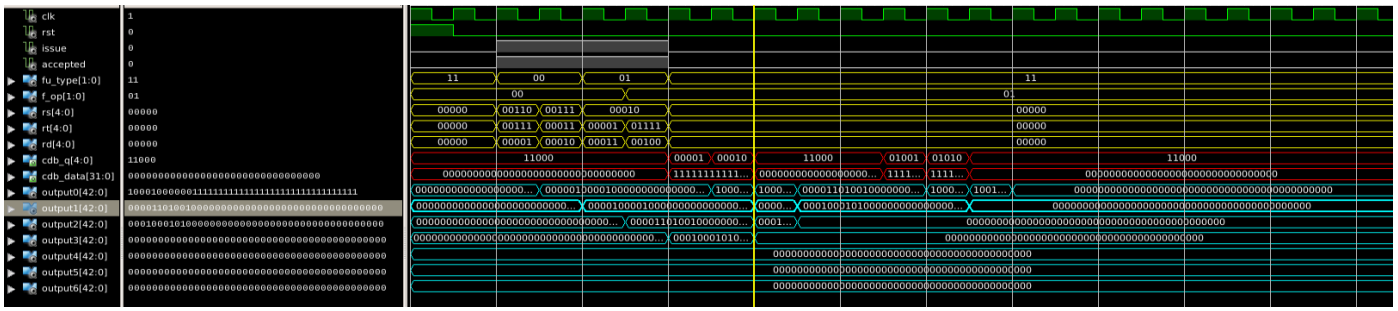
- **43ο bit** : command ready
- **42ο – 37ο bit** : Rd – καταχωρητής προορισμού εντολής
- **36ο – 32ο bit** : RS tag – FU type και name reservation station που πιθανόν φιλοξενεί την εντολή (σε περίπτωση που Rs,Rt δεν είναι έτοιμα)
- **31ο – 0ο bit** : δεδομένα για εγγραφή στον Rd.

Με την έκδοση κάποιας εντολής , ο Reorder Buffer γεμίζει τα πεδία Rd και tag του πρώτου διαθέσιμου καταχωρητή του από τις αντίστοιχες τιμές που λαμβάνει από τον Issuer, ενώ τα υπόλοιπα πεδία αρχικοποιούνται σε μηδενικές τιμές. Κάθε καταχωρητής του reorder buffer που είναι γεμάτος περιμένει κοινοποίηση από το CDB με tag ίδιο με το δικό του προκειμένου να ενημερώσει τα δεδομένα του, μηδενίζοντας παράλληλα το tag του και θέτοντας ενεργό τον πεδίο ready. Μόλις τα δεδομένα του πρώτου καταχωρητή του reorder buffer είναι έτοιμα (το 43ο bit είναι άσσος), ο reorder buffer μπορεί να προβεί σε commit της εντολής στο Register File, περνώντας στο πεδίο AWR του Register File το Rd του και στο πεδίο DataIn τα δεδομένα του. Το σήμα commit αποτελεί έξοδο του reorder buffer και συνδέεται με το write enable του Register File. Μόλις ο reorder buffer κάνει commit τα δεδομένα του πρώτου του καταχωρητή ολισθαίνει τα δεδομένα όλων τα γεμάτων καταχωρητών κατά μία θέση , αντικαθιστώντας κατά αυτό τον τρόπο τα δεδομένα του πρώτου καταχωρητή με τα δεδομένα του δεύτερου κοκ. Προκειμένου να αποφύγουμε περιπτώσεις μη ορθής ενημέρωσης καταχωρητών, λόγω ταυτόχρονης κοινοποίησης και ολίσθησης, οι ίδιοι οι καταχωρητές παρακολουθούν το CDB και ενημερώνουν κατάλληλα τα δεδομένα τους βάσει του tag που έχουν σε περίπτωση που το write enable τους είναι ανενεργό, αλλιώς ενημερώνουν τα δεδομένα τους βάσει του tag της εισόδου.

Με την προσθήκη του reorder buffer, παρότι δεν αλλάζει η λειτουργικότητα των module της προηγούμενης εργαστηριακής άσκησης, αλλάζει αρκετά η λογική λειτουργίας του επεξεργαστή μας καθώς υπάρχουν σημαντικές αλλαγές στο top level. Όπως προαναφέρθηκε το Register File ενημερώνεται αποκλειστικά πλέον από τον reorder Buffer, ενώ δεν περιέχει πεδία q στους καταχωρητές του και συνεπώς δεν υπάρχουν πλέον οι έξοδοι Qout1 και Qout2 (αντίστοιχες με read1 και read2 αλλά για το tag). Τα tag πλέον της εκάστοτε εντολής βρίσκονται στον Reorder Buffer. Καθώς υπάρχει η ανάγκη τα reservation station να λαμβάνουν τα δεδομένα Vj,Vk της εκάστοτε εντολής μόνο εφόσον είναι έτοιμα από προηγούμενες εντολές, τόσο οι είσοδοι ARD1,ARD2 όσο και οι έξοδοι Read1,Read2 του Register File συνδέονται πλέον ως είσοδοι στον reorder Buffer. Ο Reorder Buffer ελέγχει αν τα ARD1 ή ARD2 ταυτίζονται με το Rd κάποιου από τους καταχωρητές του και σε αυτήν την περίπτωση περνάει στην έξοδο Qout1, Qout2 αντίστοιχα τα tag των καταχωρητών με μηδενικά δεδομένα, αλλιώς στις εξόδους του εμφανίζονται τα read1, read2 με μηδενικά tag. Έναν κύκλο με τα την έκδοση της εντολής ,λοιπόν, το εκάστοτε Reservation Station θα λάβει τα δεδομένα του από τον Reorder Buffer.

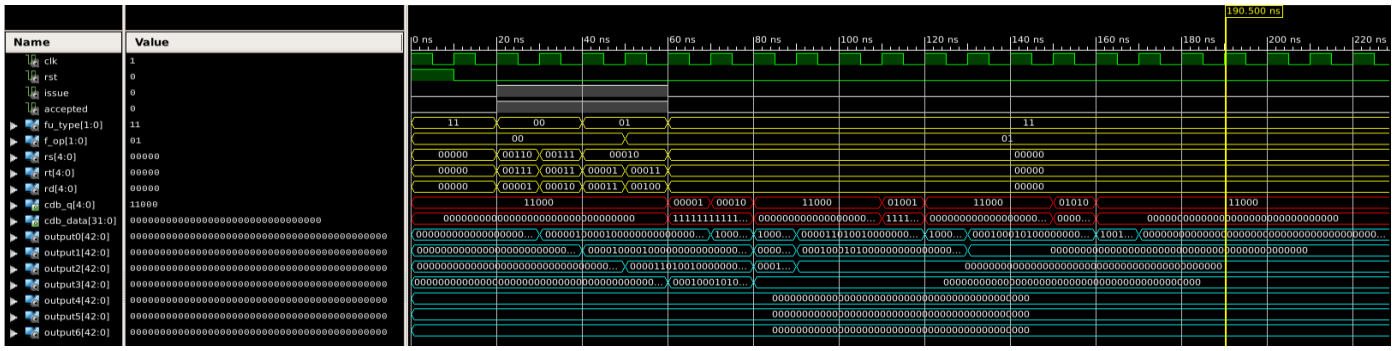
Simulations

Στο παρακάτω simulation εκτελούμε 4 εντολές. Στην αρχή εκτελούμε 2 εντολές NOT και στην συνέχεια 1 εντολή ADD και 1 εντολή SUB. Η πρώτη εντολή ADD προκειμένου να ολοκληρωθεί πρέπει να είναι έτοιμα τα αποτελέσματα των 2 προηγούμενων εντολών NOT. Όπως φαίνεται και στο simulation , όντως η 3η εντολή κοινοποιείται με καθυστέρηση καθώς η εκτέλεση της ξεκινά με την κοινοποίηση των 2 προηγούμενων NOT (τόσο τα tag όσο και τα δεδομένα των κοινοποιήσεων φαίνονται με κόκκινο χρώμα). Αντίστοιχα η 4η εντολή χρησιμοποιεί τα αποτελέσματα της 2ης NOT και κοινοποιεί αποτελέσματα ακριβώς μετά την ADD. Κατά την έκδοση των εντολών παρατηρούμε ότι γεμίζουν οι 4 πρώτοι καταχωρητές του ReorderBuffer (με γαλάζιο χρώμα). Μόλις κάποια εντολή κοινοποιηθεί, ο καταχωρητής που «φιλοξενεί» την συγκεκριμένη εντολή ενημερώνει την τιμή του και ενεργοποιεί το πρώτο bit του, δεικνύοντας έτσι ότι η εντολή έχει εκτελεστεί. Εφόσον η εντολή του πρώτου καταχωρητή είναι έτοιμη τότε μπορεί να γίνει commit στο Register File. Κατόπιν του commit οι υπόλοιπες εντολές ολισθαίνονται στους καταχωρητές κατά μία θέση μπροστά . Η ίδια διαδικασία επαναλαμβάνεται και για τις υπόλοιπες εντολές, μέχρις ότου ο ReorderBuffer να είναι άδειος.

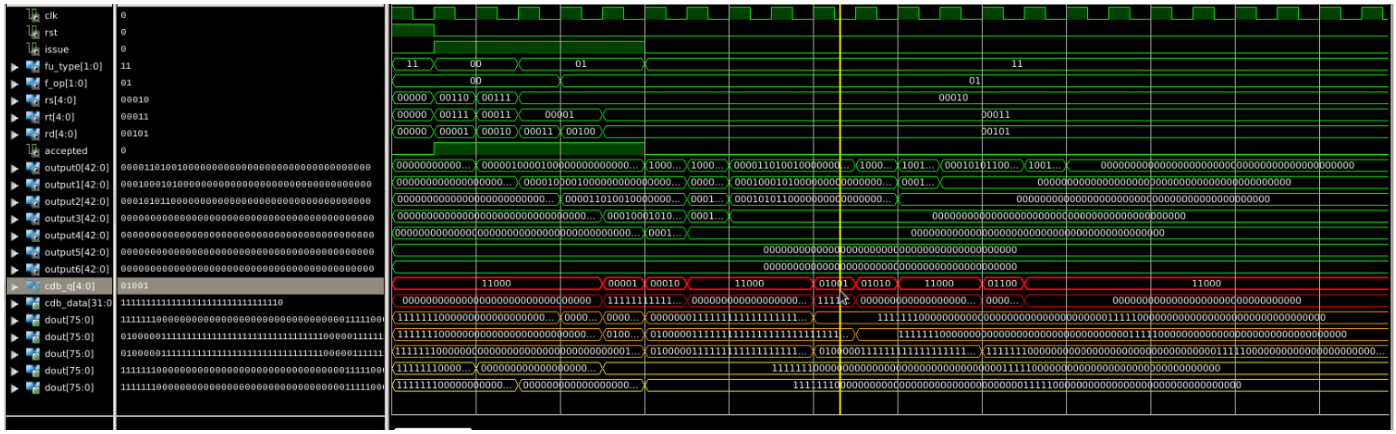


Αντίστοιχα με το παραπάνω simulation, εκτελούμε στο παρακάτω τις ίδιες εντολές, με την διαφορά ότι η τελευταία εντολή SUB περιμένει τα αποτελέσματα της προηγούμενης ADD.

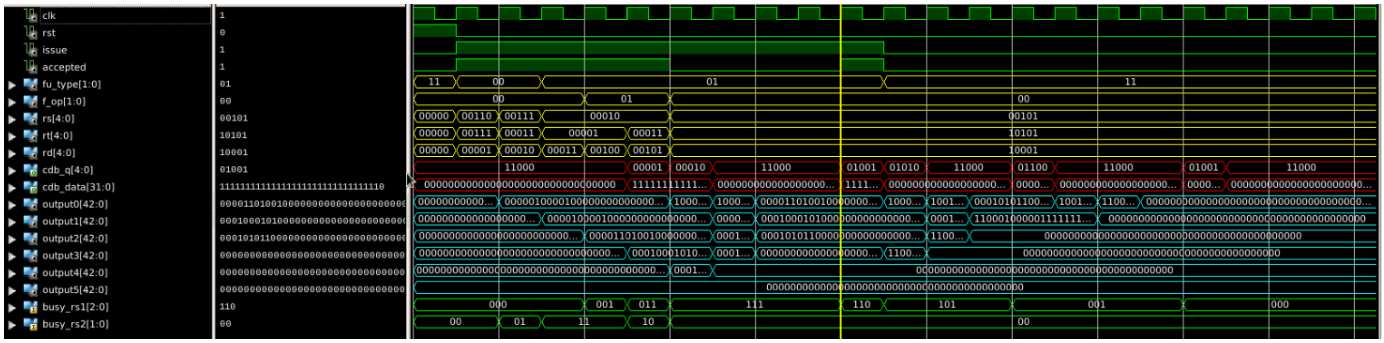
Όντως, τα αποτελέσματα τις 4ης εντολής κοινοποιούνται 3 κύκλους μετά την 3η εντολή, ενώ τα αποτελέσματα αυτής γίνονται commit στο Register File από τον Reorder Buffer έναν κύκλο μετά.



Εν συνεχεία , σχεδιάσαμε ένα simulation όπου γεμίζουν όλα τα reservation station, έχοντας ξανά 2 εντολές NOT για αρχή, έπειτα μια εντολή ADD και κατόπιν 2 εντολές SUB. Η ADD και η πρώτη SUB χρησιμοποιούν τα αποτελέσματα των NOT, οπότε εκτελούνται η μία διαδοχικά της άλλης μετά από 3 κύκλους την κοινοποίηση των NOT. Η τελευταία εντολή εξαρτάται από την προηγούμενη της , για αυτόν τον λόγο και αργεί 3 κύκλους να κοινοποιηθεί από την προηγούμενη της.



Έπειτα δοκιμάζουμε την περίπτωση όπου αφού γεμίσουν όλα τα reservation station εκδίδεται καινούργια εντολή. Παρατηρούμε ότι μόλις εκδοθούν οι πρώτες 5 εντολές το σήμα accepted απενεργοποιείται και η εντολή γίνεται δεκτή μόλις αδειάσει ένα κατάλληλο reservation station, ενώ όσο το accepted είναι ανενεργό τόσο τα reservation station όσο και ο Reorder Buffer δεν φορτώνουν καινούργιες εντολές.



Τελικά , με το παρακάτω simulation θέλουμε να δείξουμε ότι παρότι κοινοποιήσεις αποτελεσμάτων εντολών μπορούν να προκύψουν out of order, τα commit από τον Reorder Buffer στην Register File γίνονται in order. Συγκεκριμένα παρατηρούμε στο παρακάτω simulation ότι γίνεται πρώτα η κοινοποίηση εντολής που καταχωρήθηκε στο 2ο reservation station (01010) και κατόπιν αυτή του 1ου (01001), καθώς η εντολή με tag 01001 προκειμένου να εκτελεστεί πρέπει να περιμένει τα αποτελέσματα των 2 προηγούμενων NOT. Παρότι κοινοποιείται πρώτη η εντολή με tag 01010 ο reorder Buffer πρώτα κάνει commit την εντολή με tag 01001, διατηρώντας κατά αυτόν το τρόπο την σειρά κατά την οποία εκδόθηκαν οι εντολές. Με πορτοκαλί χρώμα παρουσιάζονται οι τιμές των register προορισμού της κάθε εντολής . Για λόγους απλότητας χρησιμοποιούμε ως καταχωρητές προορισμού τους καταχωρητές 1 – 5, ανάλογα με την σειρά έκδοσης της κάθε εντολής . Παρατηρούμε ότι για κάθε εντολή τα αποτελέσματα της εμφανίζονται στην έξοδο του εκάστοτε καταχωρητή του Register File μετά από έναν κύκλο από την κοινοποίηση της, εκτός από την 4η εντολή που καταχωρείται στο Register File 2 κύκλους μετά την κοινοποίηση του tag 01001. Όσον αφορά την 3η αφαίρεση εξαρτάται και αυτήν από την προηγούμενη ADD για αυτό και η κοινοποίηση της αργεί 3 κύκλους, γράφοντας τελικά τα αποτελέσματα στην Register File στον καταχωρητή 5 έναν κύκλο μετά.