4/11/2018

1ή Εργαστηριακή Άσκηση

ΔΥΝΑΜΙΚΗ ΟΜΟΧΕΙΡΙΑ ΣΕ ΠΥΡΗΝΑ ΕΠΕΞΕΡΓΑΣΤΗ - ΑΛΓΟΡΙΘΜΟΣ TOMASULO

Ομάδα LAB41538988

ΧΡΗΣΤΟΣ ΖΗΣΚΑΣ ΑΜ 2014030191
ΜΑΝΩΛΗΣ ΠΕΤΡΑΚΟΣ ΑΜ 2014030009

Σκοπός εργαστηριακής άσκησης

Σκοπός της εργαστηριακής άσκησης είναι η τμηματική υλοποίηση του back-end ενός επεξεργαστή βασιζόμενου στην εκτέλεση του αλγορίθμου Tomasulo. Η υλοποίηση της σχεδίασης απαιτεί την κατασκευή 5 βασικών δομικών στοιχείων:

- Reservation Station
- Register File
- CDB Bus
- Functional Units
- Issue stage

Αυτά σε συνδυασμό με τα αντίστοιχα Control των μονάδων και την διαχείριση των σημάτων αναμονής & ετοιμότητας μεταξύ τους, ενοποιούν τη συνολική σχεδίαση. Η σχεδίαση αυτή εξυπηρετεί ώστε οι εντολές να επεξεργάζονται από το σύστημα μέσω της υπηρεσίας διοχετευούσης (pipe-line).

Σημείωση: Η δρομολόγηση εντολών από την μονάδα IF αποτελεί εξωτερική μονάδα του συστήματος και θεωρείται αδιάφορη από το σύστημα πέρα από την επικοινωνία για αποδοχή ή απόρριψη της εντολής.

Διεπαφές

Διεπαφή Issue stage (εκτός IF stage)

Σήμα	Πλάτος	Είδος	Περιγραφή
RS_tags	15 bit	Είσοδος	Tags που στέλνουν τα RS (5 bit ανά RS)
available	3 bit	Είσοδος	Δείχνει πια RS έχουν διαθέσιμο χώρο (1 bit ανά RS)
issue_RS	3 bit	Έξοδος	Υποδεικνύει στον κατάλληλο RS να αποθηκεύσει την εντολή (1 bit ανά RS)

Fop	2 bit	Έξοδος	Κωδικός Πράξης
Ri	5 bit	Έξοδος	Αριθμός καταχωρητή προορισμού
Rj	5 bit	Έξοδος	Αριθμός καταχωρητή πηγής #1
Rk	5 bit	Έξοδος	Αριθμός καταχωρητή πηγής #2
tagRF	5 bit	Έξοδος	Το tag του RS block που δέχτηκε την εντολή
Instr_Valid	1 bit	Έξοδος	Ενημερώνει το RF ότι η εντολή είναι έγκυρη

Διεπαφή Common Data Bus

Σήμα	Πλάτος	Είδος	Περιγραφή
request	3 bit	Είσοδος	Σήματα αίτησης χρήσης του CDB από τα FU
grant	3 bit	Έξοδος	Ενημερώνει τα FU αν θα πάρουν τον έλεγχο τον επόμενο κύκλο
Q1	5 bit	Είσοδος	Tag από το FU1
Q2	5 bit	Είσοδος	Tag από το FU2
Q3	5 bit	Είσοδος	Tag από το FU3
Value1	32 bit	Είσοδος	Αποτέλεσμα του FU1
Value2	32 bit	Είσοδος	Αποτέλεσμα του FU2
Value3	32 bit	Είσοδος	Αποτέλεσμα του FU3
CDBout	38 bit	Έξοδος	Συνδυασμός του σωστού tag και αποτελέσματος, καθώς και του CDB_valid(1 αν τα αποτελέσματα είναι έγκυρα, αλλιώς 0)

Διεπαφή Register File

Σήμα	Πλάτος	Είδος	Περιγραφή
Ri	5 bit	Είσοδος	Αριθμός καταχωρητή προορισμού
Rj	5 bit	Είσοδος	Αριθμός καταχωρητή πηγής #1
Rk	5 bit	Είσοδος	Αριθμός καταχωρητή πηγής #2
Instr_valid	1 bit	Είσοδος	Υποδεικνύει αν η εντολή είναι έγκυρη
Vj	32 bit	Έξοδος	Δεδομένα καταχωρητή πηγής #1
Vk	32 bit	Έξοδος	Δεδομένα καταχωρητή πηγής #2
Qj	5 bit	Έξοδος	Tag καταχωρητή πηγής #1
Qk	5 bit	Έξοδος	Tag καταχωρητή πηγής #2
CDB	38 bit	Είσοδος	Δεδομένα που κοινοποιεί ο CDB

Διεπαφή RS

Σήμα	Πλάτος	Είδος	Περιγραφή

RF_Vj	32 bit	Είσοδος	Δεδομένα καταχωρητή πηγής #1
RF_Vk	32 bit	Είσοδος	Δεδομένα καταχωρητή πηγής #2
RF_Qj	5 bit	Είσοδος	Tag καταχωρητή πηγής #1
RF_Qk	5 bit	Είσοδος	Tag καταχωρητή πηγής #2
Fop	2 bit	Είσοδος	Κωδικός Πράξης
issue_ready	1 bit	Είσοδος	Εγκυρότητα εντολής. Υποδεικνύει ότι πρέπει να αποθηκευτεί
available	1 bit	Έξοδος	Ενημερώνει ότι έχει ελεύθερο χώρο
tagRF	5 bit	Έξοδος	Δείχνει σε πιο block αποθηκεύεται η εντολή
CDB	38 bit	Είσοδος	Δεδομένα που κοινοποιεί ο CDB
ready_FU	1 bit	Είσοδος	Υποδεικνύει ότι το FU είναι διαθέσιμο
Vj_out	32 bit	Έξοδος	Δεδομένα πρώτου τελεστέου
Vk_out	32 bit	Έξοδος	Δεδομένα δεύτερου τελεστέου
Fop_out	2 bit	Έξοδος	Κωδικός Πράξης
tagFU	5 bit	Έξοδος	Δείχνει από πιο block έρχεται η εντολή

Σημείωση: Όλα τα RS έχουν την ίδια διεπαφή.

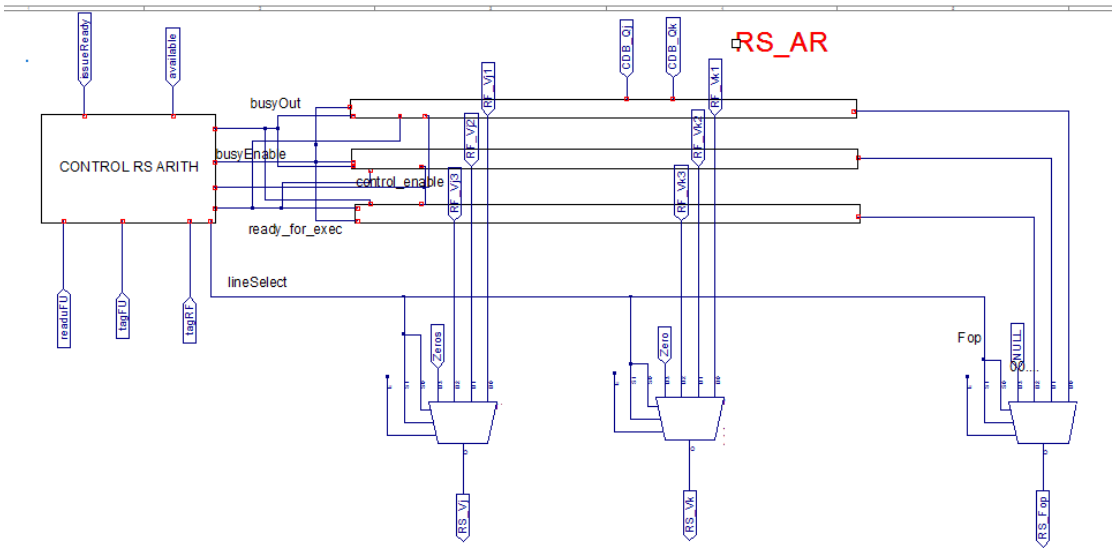
Διεπαφή FU

Σήμα	**Πλάτος**	**Είδος**	**Περιγραφή**
tag	5 bit	Είσοδος	Δείχνει από πιο block έρχεται η εντολή
op	2 bit	Είσοδος	Κωδικός Πράξης
value_in1	32 bit	Είσοδος	Δεδομένα πρώτου τελεστέου
value_in2	32 bit	Είσοδος	Δεδομένα δεύτερου τελεστέου
ready	1 bit	Έξοδος	Υποδεικνύει ότι το FU είναι διαθέσιμο
value_out	32 bit	Έξοδος	Αποτέλεσμα του FU
Q	5 bit	Έξοδος	Δείχνει από πιο block έρχεται η εντολή
request	1 bit	Έξοδος	Σήμα αίτησης χρήσης του CDB από το FU
grant	1 bit	Είσοδος	Ενημερώνει το FU αν θα πάρει τον έλεγχο του CDB τον επόμενο κύκλο

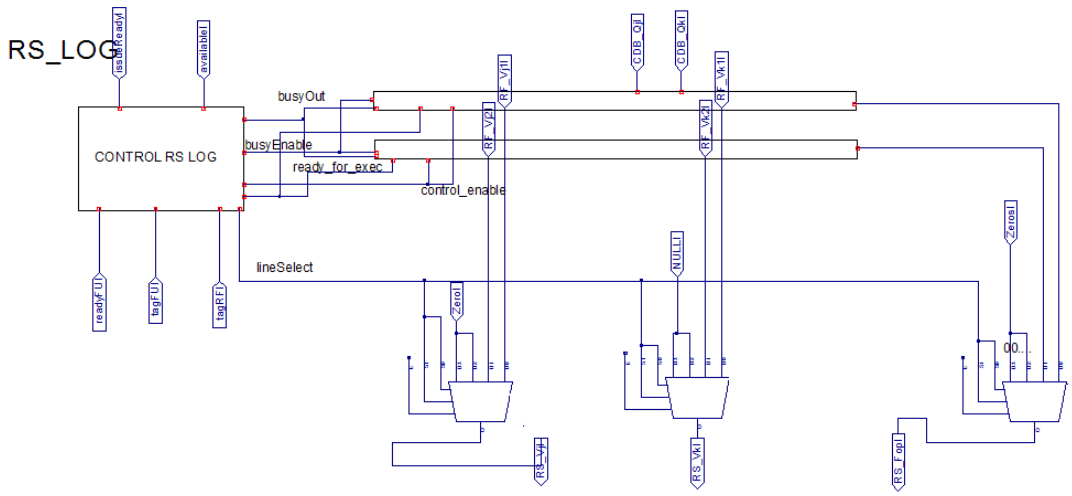
Σημείωση: Και τα δύο FU έχουν την ίδια διεπαφή αλλά τα δεδομένα τους έρχονται από διαφορετικά RS.

Σχηματικά διαγράμματα

ΒΑΘΜΙΔΑ Reservation Station-arithmetic

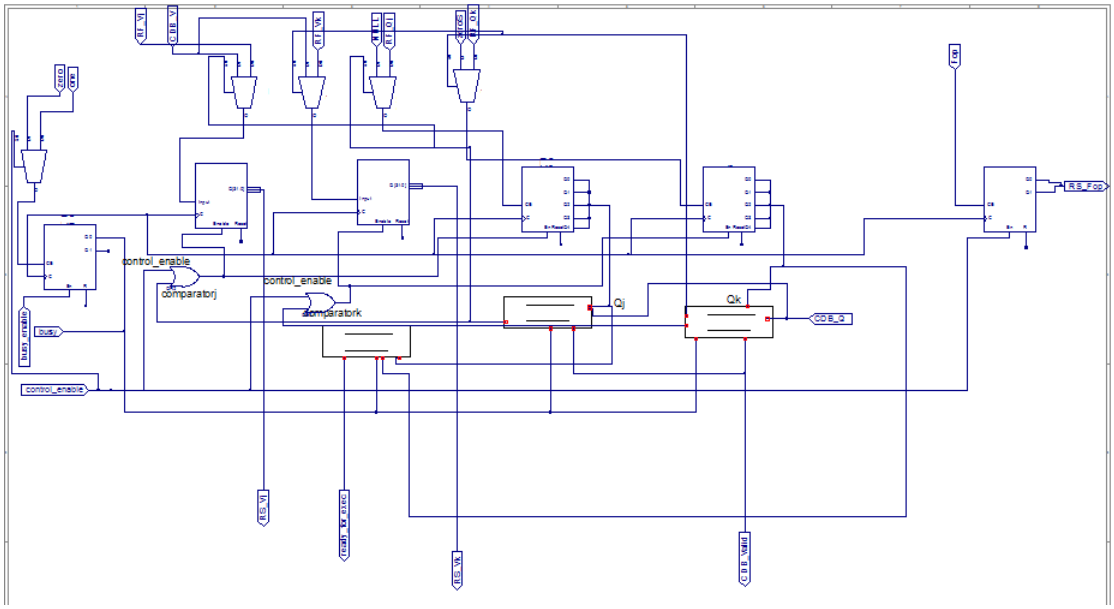


ΒΑΘΜΙΔΑ Reservation Station-logical

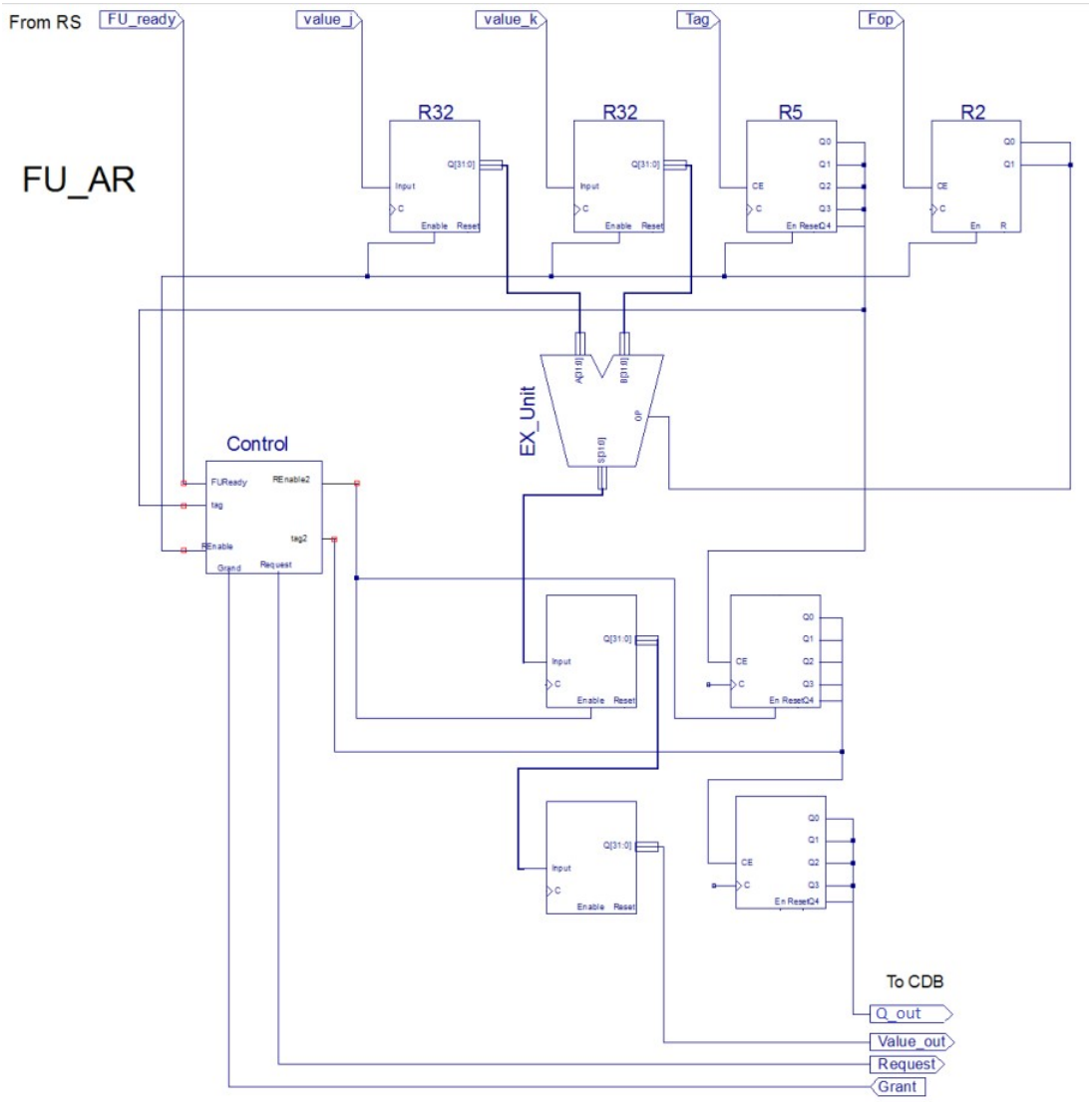


Σημείωση: Οι γραμμές καταχωρητών έχουν υλοποιηθεί με τον παρακάτω τρόπο(RS Block).

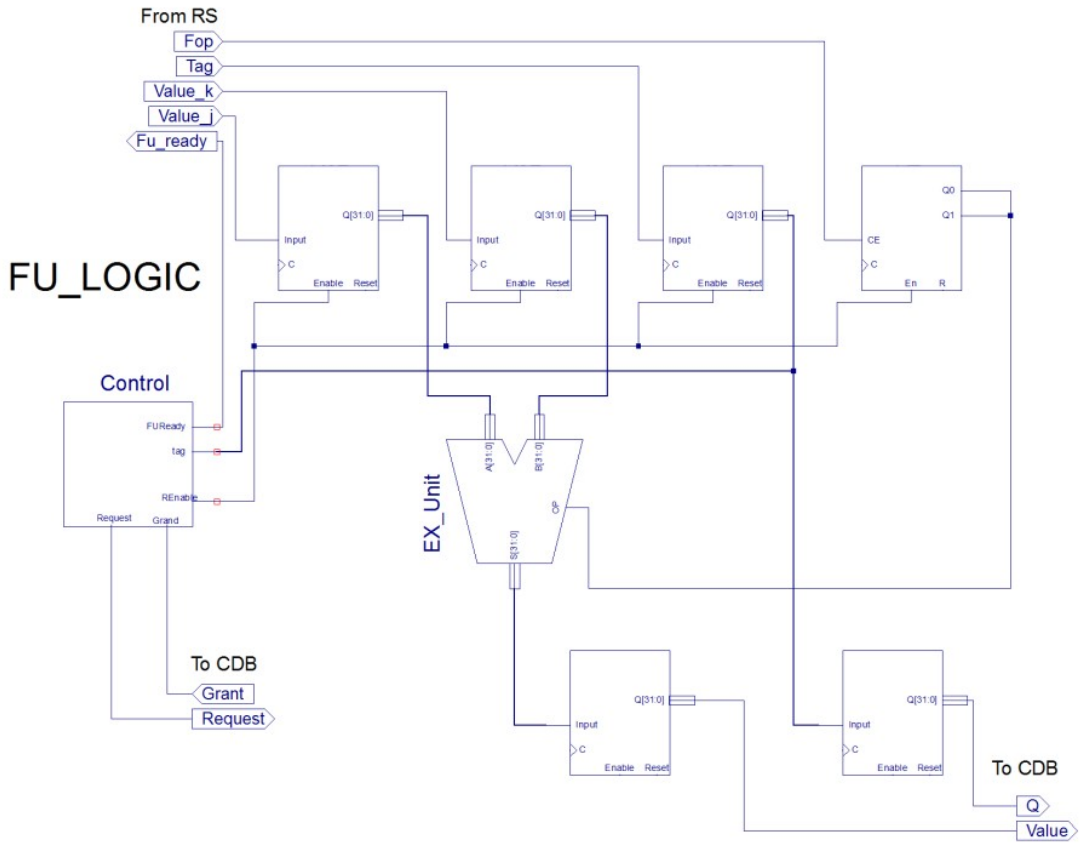
Σχεδιάγραμμα RS Block



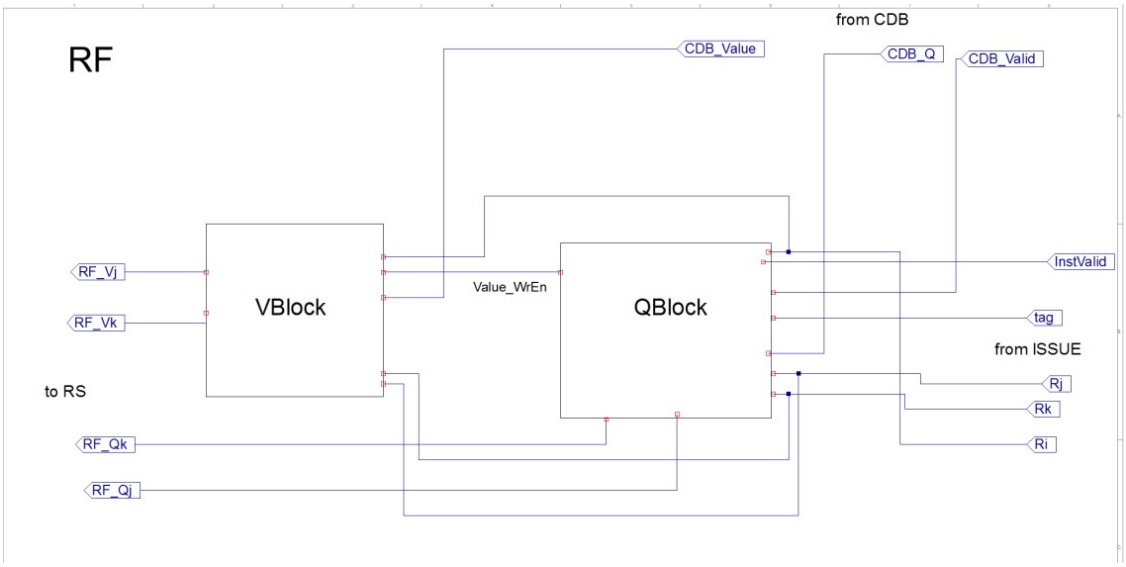
ΒΑΘΜΙΔΑ Functional Unit-arithmetic



ΒΑΘΜΙΔΑ Functional Unit-logical

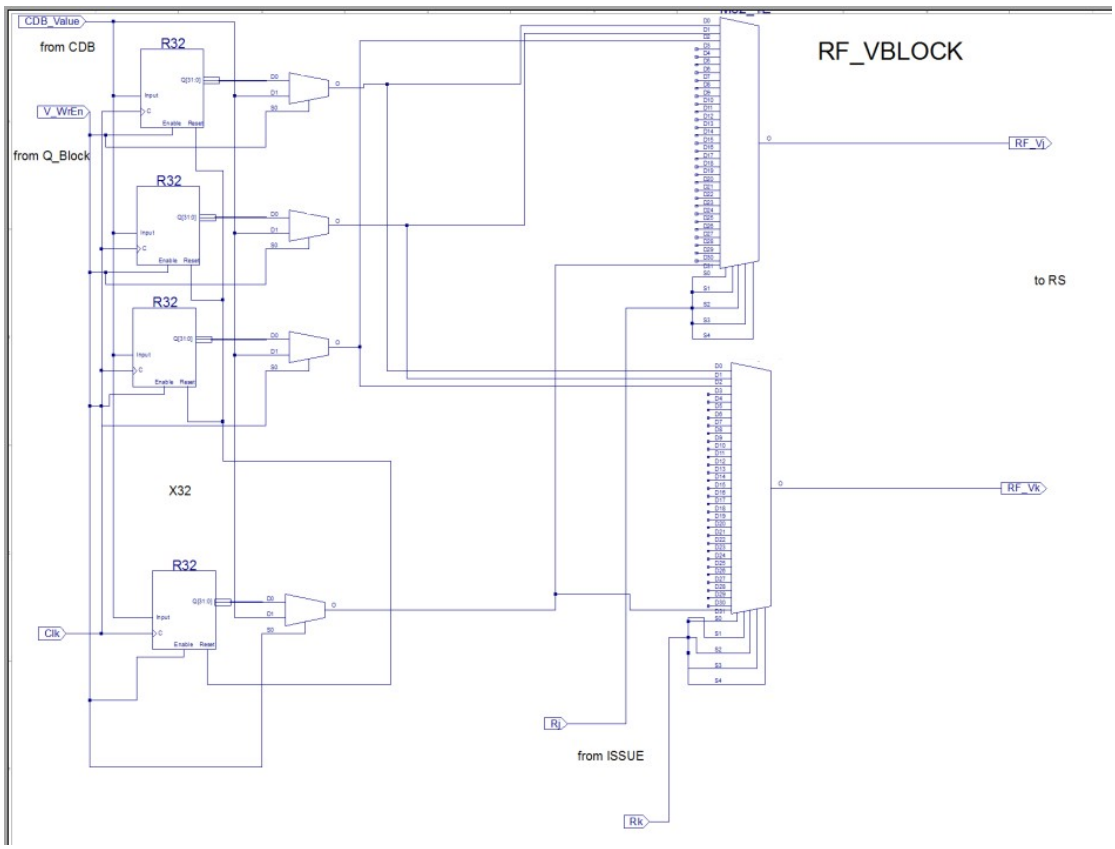


ΒΑΘΜΙΔΑ Register File

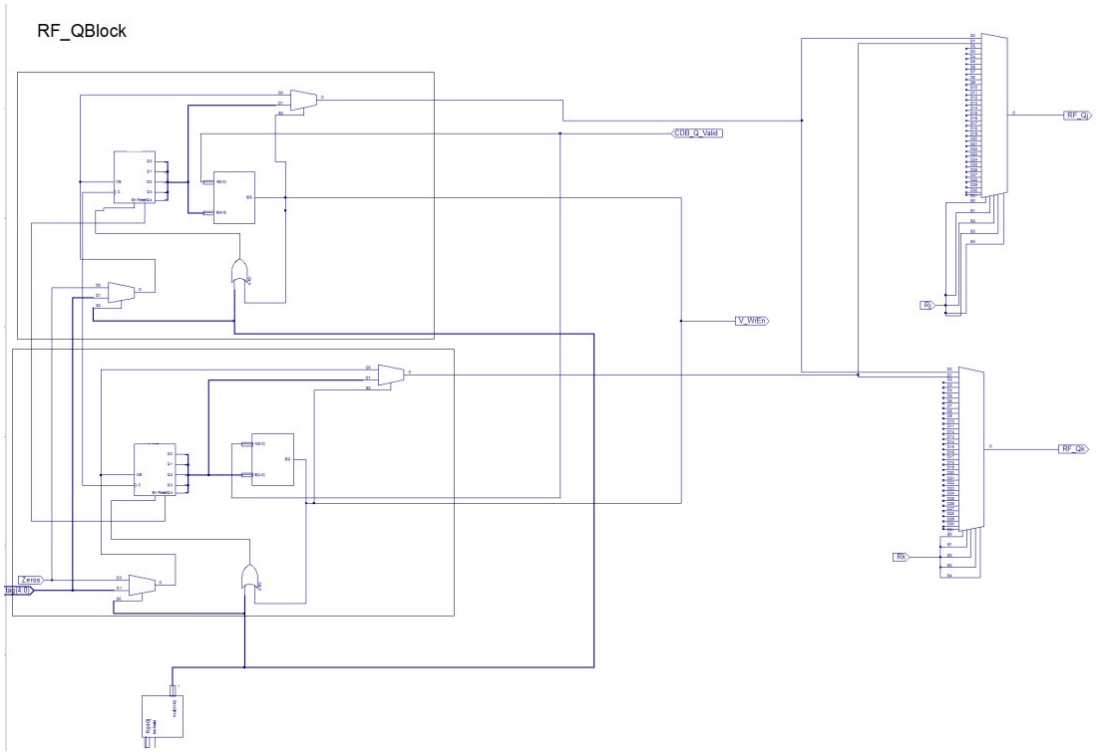


Σημείωση: Υπάρχουν 32 καταχωρητές V και Q σε αντιστοιχία. Στα παρακάτω διαγράμματα φαίνεται η δομή τους. Δεν έχουν σχεδιαστεί και οι 32 καταχωρητές για λόγους απλότητασ.

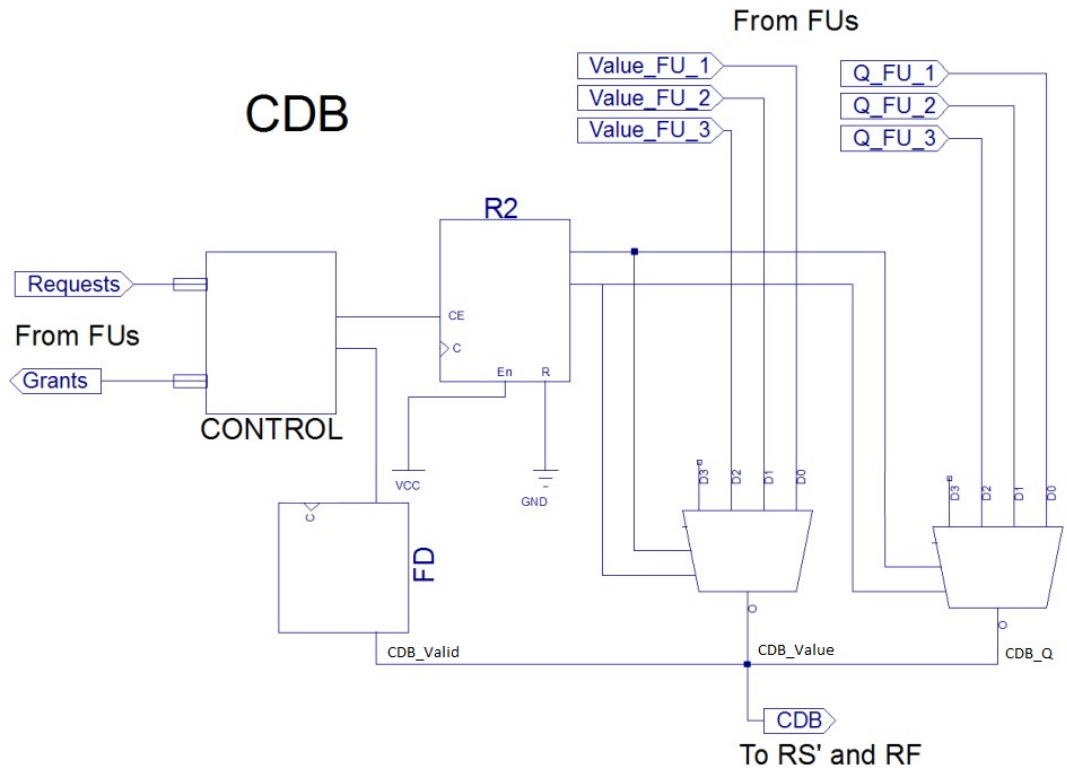
Σχεδιάγραμμα V_block



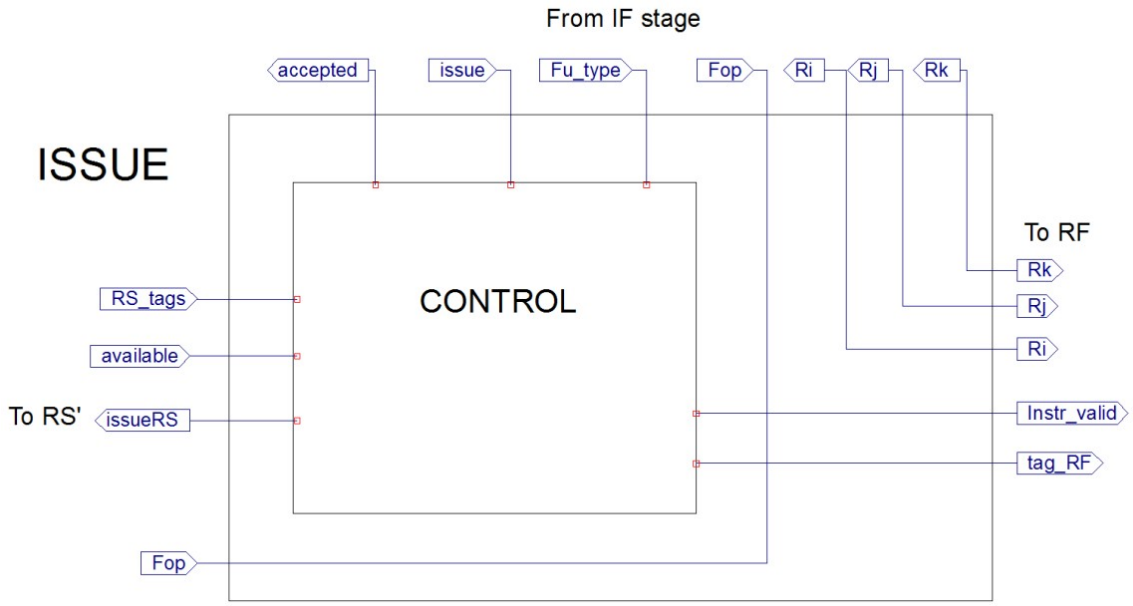
Σχεδιάγραμμα Q_block



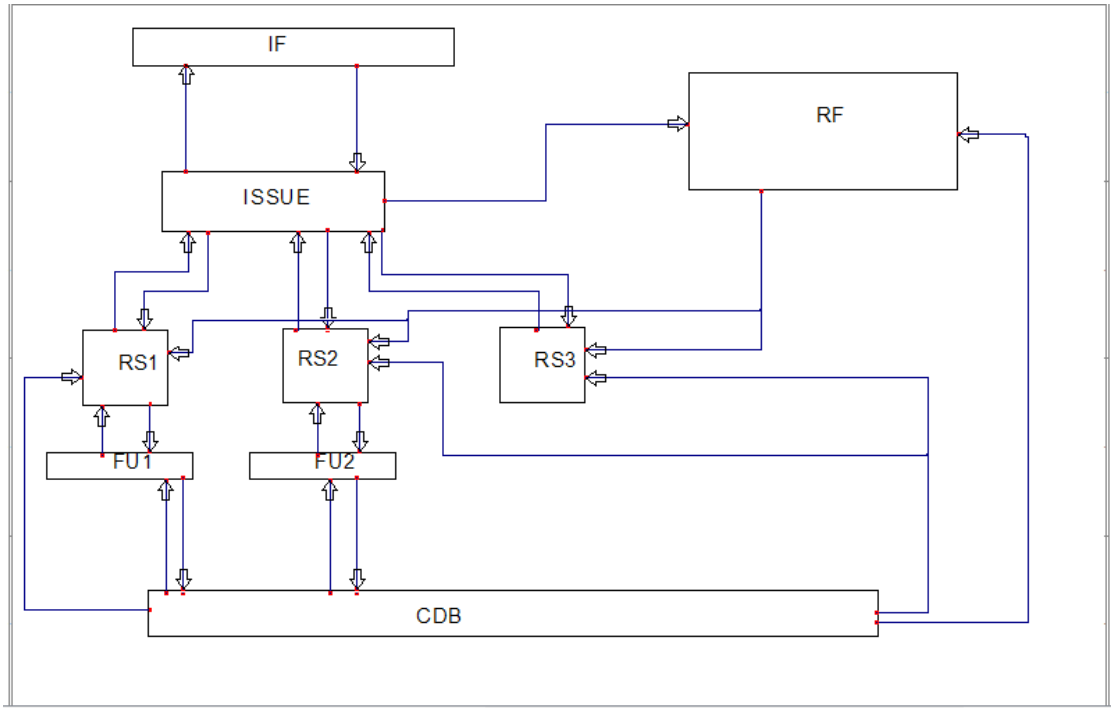
ΒΑΘΜΙΔΑ Common Data Bus-CDB



ΒΑΘΜΙΔΑ Issue



Ολοκληρωμένο διάγραμμα



Διάγραμμα χρονισμού

Διάγραμμα χρονισμού μίας AND. Θεωρούμε ότι δεν υπάρχουν συγκρούσεις και τα δεδομένα πηγής είναι σωστά.

- 1ος κύκλος

Έλεγχος δομικών κινδύνων από το Issue.

Διάβασμα καταχωρητών πηγής από τον RF.

Renaming καταχωρητή προορισμού στον RF.

Αποθήκευση εντολής στο RS.

- 2ος κύκλος

Αποστολή της εντολής από το RS στο FU.

- 3ος κύκλος

Εκτέλεση πράξης στο FU.

Το FU ζητάει από τον έλεγχο του CDB άδεια πρόσβασης για τον επόμενο κύκλο. Το CDB απαντάει άμεσα.

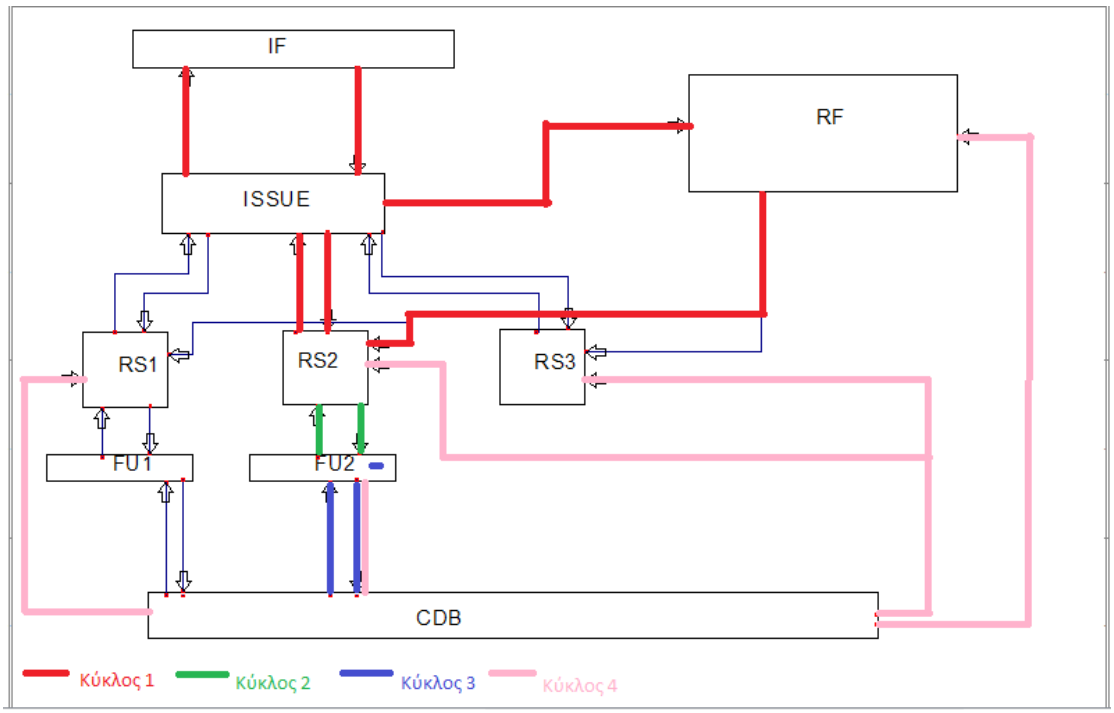
- 4ος κύκλος

Αποστολή αποτελέσματος από το FU στο CDB.

Κοινοποίηση αποτελέσματος από το CDB στο FU και τα RS.

Αποθήκευση αποτελέσματος στο FU και τα RS που το περιμένουν.

Το παρακάτω διάγραμμα απεικονίζει σχηματικά την προαναφερθείσα περιγραφή.



Περιγραφή

Η ολοκλήρωση του data-path της συνολικής σχεδίασης καθιστά αναγκαία την κατασκευή των επιμέρους units που προσδιορίζουν παραλληλία στην διευθυνσιοδότηση των εντολών . Οι controllers των διαφόρων μονάδων λειτουργούν ασύγχρονα με το ρολόι. Οι καθυστερήσεις, όπου χρειάζονται, υλοποιούνται με καταχωρητές.

Reservation Station: Τα RS εξυπηρετούν τρεις λειτουργίες. Η πρώτη είναι η προσωρινή αποθήκευση της εντολής κατά την έκδοση της. Η δεύτερη είναι, σε περίπτωση που τα δεδομένα που ήρθαν από τον RF είναι ξεπερασμένα, να τα αντικαταστήσει με τα κατάλληλα από το CDB. Η τελευταία είναι η προώθηση της εντολής στο FU.

Η αποθήκευση της εντολής γίνεται κατά την έκδοση της και απαιτεί η RS να έχει ελεύθερο χώρο (available = ‘1’). Τότε η μονάδα θα δεχτεί αίτηση από το Issue να αποθηκεύσει την εντολή (issue_ready = ‘1’). Απευθείας, θα επιλέξει το πρώτο αδέσμευτο RS_block, θα στείλει το tag του και θα αποθηκεύσει τα δεδομένα από το RF σε αυτό. Επίσης, το block γίνεται “busy” μέχρι την αποχώρηση της εντολής.

Εάν τα Q που ήρθαν από τον RF δεν είναι “00000”, σημαίνει ότι τα δεδομένα V είναι άκυρα και πρέπει να αντικατασταθούν με τα αποτελέσματα κάποιας εντολής που δεν έχει ολοκληρωθεί. Τότε, σε κάθε κύκλο, η μονάδα παρακολουθεί το CDB_Q μέχρι να είναι ίδιο με τα Q της εντολής. Όταν αυτό συμβεί, η εντολή την οποία περίμενε ολοκληρώθηκε και τα αποτελέσματά της αποθηκεύονται στα κατάλληλα V. Επίσης, μηδενίζονται τα Q και η εντολή μπορεί να συνεχίσει. Τα Q είναι ανεξάρτητα μεταξύ τους και μπορούν να εξυπηρετηθούν σε διαφορετικό κύκλο.

Αν και τα δύο Q έχουν τιμή “00000” σημαίνει ότι η εντολή είναι έτοιμη για αποστολή στο FU. Αν το FU είναι διαθέσιμο, τότε το Control δρομολογεί την εντολή που έχει την προτεραιότητα εκτέλεσης. Η προτεραιότητα καθορίζεται με round-robin τεχνική, ώστε να μην μείνει πολύ ώρα μια εντολή στάσιμη και δημιουργηθεί συμφόρηση. Αυτόματα, το πεδίο busy του block μηδενίζεται και είναι έτοιμο να φιλοξενήσει μια καινούργια εντολή. Η αλλαγή του busy γίνεται forward, ώστε να μπορεί να έρθει η καινούργια εντολή κατευθείαν και η μονάδα να είναι πλήρως pipe-lined.

Register File:Το αρχείο καταχωρητών εξυπηρετεί τρεις λειτουργίες. Στέλνει τα δεδομένα στα RS, κάνει το rename του Tomasulo και αποθηκεύει τα αποτελέσματα που έρχονται από το CDB. Οι πρώτες δύο χρειάζονται για την έκδοση εντολών και η τρίτη για την ολοκλήρωση τους. Αποτελείται από δύο block καταχωρητών και την λογική που τους συμπληρώνει (V_block, Q_block), δύο πόρτες εξόδου προς τα RS και μία πόρτα ανάγνωσης από το CDB.

Η αποστολή δεδομένων προς τα RS υλοποιείται χρησιμοποιώντας δύο πολυπλέκτες 32 προς 1 σε κάθε έξοδο. Οι τιμές διαλέγονται βάση των αριθμών καταχωρητών πηγής(Rj, Rk) που στέλνει το Issue stage. Μαζί με τα δεδομένα V στέλνονται και τα αντίστοιχα Q. Η διαδικασία γίνεται κατευθείαν και δεν απαιτεί ρολόι.

Κατά την εισαγωγή μιας εντολής στο σύστημα, εάν την έχει εγκρίνει το Issue stage (Instr_valid = ‘1’), το tag που προέρχεται από το RS θα αποθηκευτεί στον καταχωρητή Q που υποδεικνύει το σήμα Ri. Με αυτόν τον τρόπο, θα γνωρίζει σε πιο καταχωρητή πρέπει να αποθηκευτεί το αποτέλεσμα όταν αυτό δημιουργηθεί.

Για τη αποθήκευση των αποτελεσμάτων χρησιμοποιείται ένας συγκριτής για κάθε καταχωρητή Q. Αν τα περιεχόμενα του CDB είναι έγκυρα (CDB_valid = ‘1’) και η τιμή κάποιου καταχωρητή Q είναι ίδιο με το CDB_Q, αποθηκεύεται το πεδίο CDB_V στον αντίστοιχο καταχωρητή V. Επίσης, η τιμή του καταχωρητή Q γίνεται “00000”.

Κάθε κύκλο μπορεί να γίνει μία έκδοση εντολής (αποστολή δεδομένων και rename) και μία ανάγνωση του CDB ταυτόχρονα. Υπάρχουν δύο περιπτώσεις σύγκρουσης τους. Η πρώτη

είναι στον ίδιο κύκλο να γίνεται εγγραφή αποτελέσματος σε ένα καταχωρητή (match με το CDB_Q) και ανάγνωση του ιδίου καταχωρητή. Τότε τα νέα δεδομένα (CDB_V) γίνονται forward και το Q διαβάζεται μηδέν. Η δεύτερη είναι στον ίδιο κύκλο να γίνεται εγγραφή σε ένα καταχωρητή (match με το CDB_Q) και rename σε αυτόν. Σε αυτήν τη περίπτωση, δίνεται προτεραιότητα στην νέα εντολή και γίνεται το rename.

CDB Bus: Ο σκοπός του CDB είναι να κοινοποιεί τα αποτελέσματα των FUs στα RS και τον RF. Ο δίαυλος δεδομένων δέχεται σήματα αίτησης(request) από τα FUs και άμεσα εξάγει σήματα ανταπόκρισης (grant) προς αυτά. Σε περίπτωση συγκρούσεων η διευθέτηση πρόσβασης πραγματοποιείται με round -robin τεχνική, ώστε να αποφεύγεται η συνεχόμενη εξυπηρέτηση του request από κάποιο συγκεκριμένο FU. Το FU που θα δεχθεί το σήμα grant παίρνει τον έλεγχο του CDB στον επόμενο κύκλο. Ο controller επιτυγχάνει αυτό, δημιουργώντας 2 ακόμα σήματα. Το πρώτο (sel) αφορά ποια τιμή θα δρομολογηθεί από τους πολυπλέκτες που κατευθύνουν τις τιμές των εκάστοτε FUs (Q,Value). Το δεύτερο σήμα αποτελεί σήμα εγκυρότητας των αποτελεσμάτων στον δίαυλο (CDB-Valid). Ως έξοδος του συστήματος είναι ο συνδυασμός των παραπάνω σημάτων(CDB-valid, Q, Value). Κατά τη διάρκεια τη εξαγωγής του αποτελέσματος, γίνεται η επόμενη διευθέτηση πρόσβασης και το σύστημα είναι pipe-lined.

Issue: Το Issue είναι ένα ασύγχρονο κύκλωμα, όπου εκδίδει τις εντολές στο υπόλοιπο σύστημα και ελέγχει δομικούς κινδύνους πάνω σε αυτές. Δέχεται από το επίπεδο IF μια αποκωδικοποιημένη εντολή και μια αίτηση Issue. Εφόσον η έτοιση είναι έγκυρη, ελέγχει αν υπάρχουν οι δομικοί πόροι για να ολοκληρωθεί η εντολή, κοιτώντας το είδος της εντολής(FU_type) και εάν το κατάλληλο RS έχει ελεύθερο χώρο(available). Σε περίπτωση που ο έλεγχος είναι επιτυχής στέλνει ένα σήμα(IssueRS) στο RS να δεχτεί την εντολή μαζί με τον τύπο της εντολής(Fop) και ανακατευθύνει στον RF τους καταχωρητές πηγής(Rj, Rk) και προορισμού(Ri) καθώς και το tag του Reservation Block που θα αποθηκευτεί η εντολή. Επίσης για να βεβαιώσει τον RF ότι η εντολή ειναι έγκυρη στέλνει ένα σήμα Instr_valid. Σε περίπτωση αποτυχίας, τα σήματα IssueRS και Instr_valid γίνονται '0' και όλες οι υπόλοιπες δομές αγνοούν την εντολή. Όλη διαδικασία γίνεται ταυτόχρονα, με αποτέλεσμα να μπορεί να γίνει μία έκδοση εντολής ανά κύκλο, όπως σε έναν μη δυναμικά pipelined επεξεργαστή.

Functional Units: Ο επεξεργαστής έχει δύο μονάδες εκτέλεσης εντολών. Η πρώτη υλοποιεί τις λογικές πράξεις (OR/NOT/AND) με καθυστέρηση 2 κύκλων και η δεύτερη τις αριθμητικές(+/-) με καθυστέρηση 3 κύκλων. Οι καθυστερήσεις επιτυγχάνονται με την χρήση επιπέδων καταχωρητών. Όσο έχουν ελεύθερους πόρους(Fu_ready), δέχονται από τα RS τον τύπο της πράξης(Fop), τους τελεστές(V1, V2) και το tag του Reservation Block που είχε αποθηκευτεί η εντολή. Αναγνωρίζουν αν ήρθε έγκυρη εντολή κοιτώντας ότι το πεδίο tag είναι διάφορο του "00000".

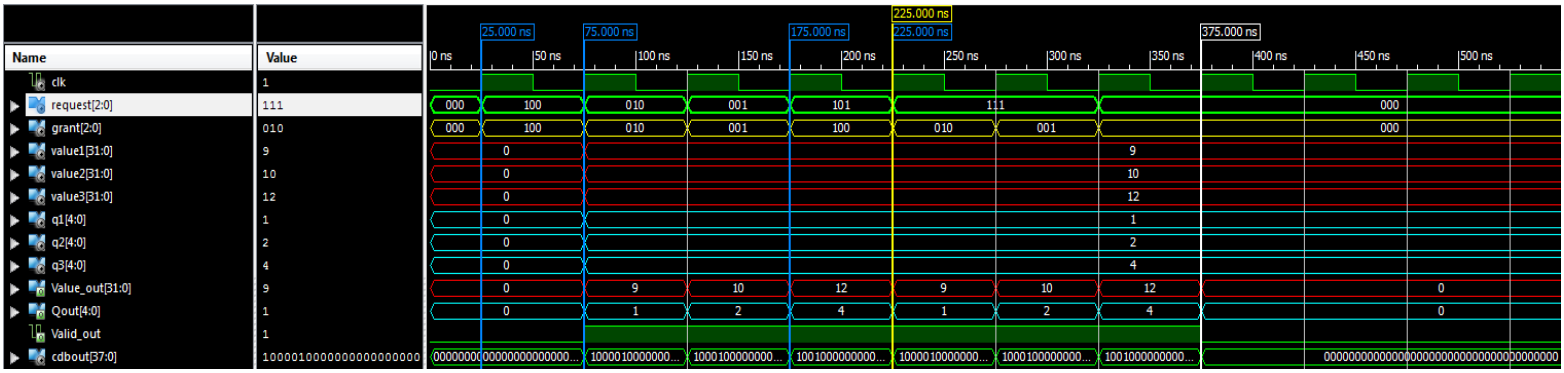
Εφόσον η εντολή είναι κατάλληλη για επεξεργασία, ένα κύκλο πριν ολοκληρωθεί ζητάνε άδεια πρόσβασης στο CDB. Αν απαντήσει αυτός θετικά, στον επόμενο κύκλο θα σταλθεί το αποτέλεσμα. Όταν δεν δοθεί άδεια, σταματάει η λειτουργία του επιπέδου καθώς και όλων των επομένων που έχουν έγκυρη εντολή. Αν όλα τα επίπεδα είναι κατειλημμένα και δεν

υπάρχουν διαθέσιμοι πόροι, σβήνει το σήμα Fu_ready ώστε να μην πανογραφούν από άλλες εντολές και χαθούν. Αυτή η λειτουργία συνεχίζεται επαναληπτικά μέχρι να δοθεί άδεια και να μπορούν αποσταλούν δεδομένα.

Σε κάθε κύκλο μπορούν να στέλνουν μία εντολή, να ζητούν άδεια για μία δεύτερη και να δέχονται μία τρίτη, δηλαδή είναι πλήρως pipelined. Στο αριθμητικό FU επειδή υπάρχει ένα ακόμα επίπεδο καταχωρητών, μπορεί να υλοποιείται και μια τέταρτη.

Κυματομορφές-Προσομοίωση

CDB:



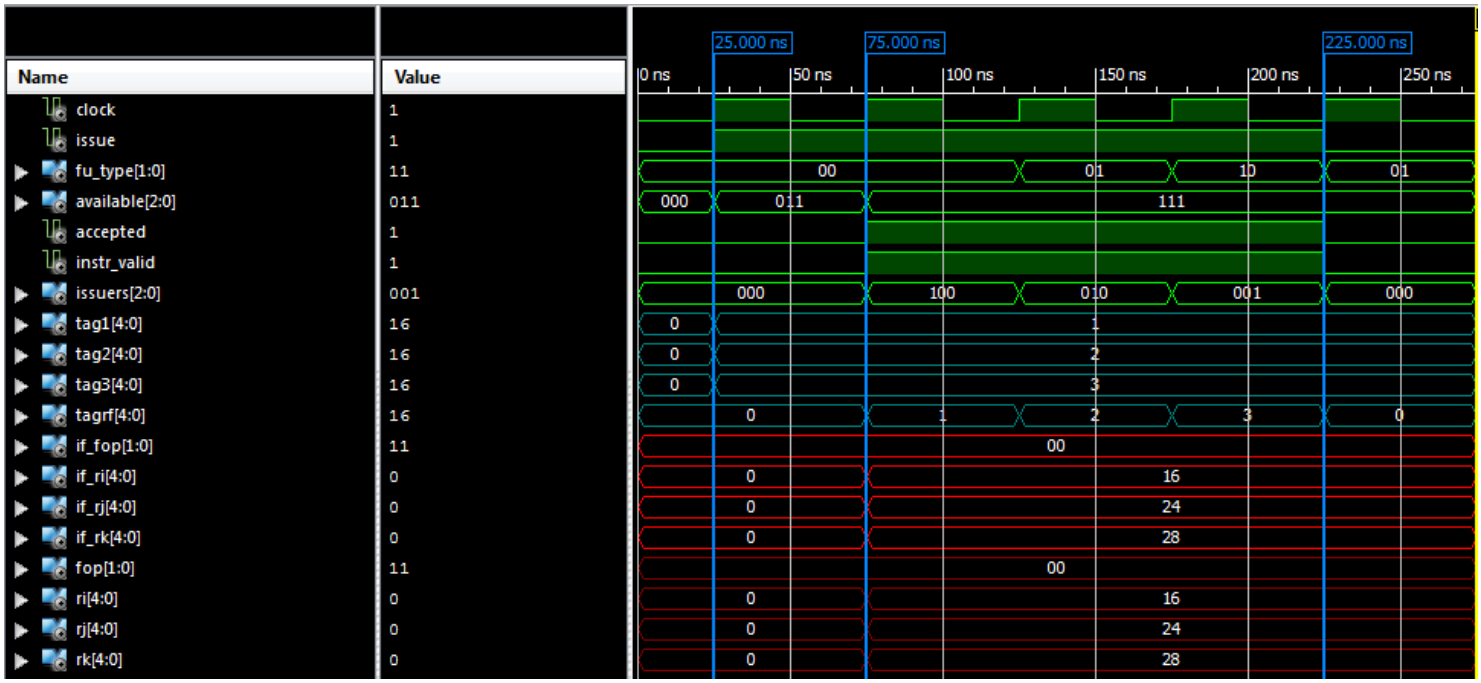
Τα πεδία Value_out, Qout και Valid_out συνδέονται σε ένα (cdbout) κατά την διάρκεια λειτουργίας, αλλά για το test παρουσιάζονται ξεχωριστά για να διαβάζεται πιο εύκολα.

25 ns: Με request = ‘100’ ζητάει άδεια μόνο το πρώτο FU και του δίνεται απευθείας grant (100). Στον επόμενο κύκλο (75 ns) περνούν τα δεδομένα του (value_out = value1, Qout = q1), το valid bit γίνεται 1 και ζητάει το δεύτερο FU άδεια(request = ‘010’) όπου την παίρνει κατευθείαν. Φαίνεται πως στον ίδιο κύκλο εξυπηρετείται ένας FU και γίνεται η επόμενη διευθέτηση άδειας, δηλαδή ότι η μονάδα είναι πλήρως pipelined.

175 – 325 ns: Γίνονται συγκρούσεις καθώς πάνω από ένα bit του request είναι 1. Φαίνεται ο round-robin αλγόριθμος καθώς το bit του grant που είναι 1 γυρνάει. Επίσης τα δεδομένα εξόδου αλλάζουν σε κάθε κύκλο. Γίνονται ίσα με αυτά εκείνου που κέρδισε τον έλεγχο στον προηγούμενο.

375 ns: Το πεδίο valid_out γίνεται ‘0’ γιατί στον προηγούμενο κύκλο δεν ζήτησε κανένας άδεια (request = “000”). Δηλαδή τα δεδομένα είναι άκυρα.

Issue:



25 ns: Το issue είναι ‘1’ αλλά η μονάδα που χρειάζεται η εντολή δεν είναι διαθέσιμη, καθώς fu_type = “00” και το αντίστοιχο available(2) = ‘0’. Τότε η εντολή δεν θα γίνει δεκτή (accepted = ‘0’), ο RF ενημερώνεται να την αγνοήσει (instr_valid = ‘0’) και κανένα RS δεν θα την αποθηκεύσει (issueRS = “000”).

75 – 225 ns: Έρχονται εντολές και όλες οι μονάδες είναι διαθέσιμες (available = “000”). Οι εντολές γίνονται δεκτές καθώς τα σήματα accepted και instr_valid γίνονται ‘1’ και τα αντίστοιχα RS ενημερώνονται (fu_type = “00” και issueRS = “100”, fu_type = “01” και issueRS = “010”, fu_type = “10” και issueRS = “001”). Επίσης το tagrf παίρνει την τιμή του αντίστοιχου tag. Τα υπόλοιπα δεδομένα περνάνε αυτούσια (fop, ri, rj, rk).

225 ns: Το issue γίνεται ‘0’ και εντολή δεν είναι έγκυρη. Τότε όλα τα σήματα ελέγχου (accepted, instr_valid, issueRS) γίνονται ‘0’ και δεν εκδίδεται εντολή.

Register File:

Τα πεδία cdb_value, cdb_q και cdb_valid συνδέονται σε ένα (cdb) κατά την διάρκεια λειτουργίας, αλλά για το test παρουσιάζονται ξεχωριστά για να διαβάζεται πιο εύκολα. Επίσης παρουσιάζονται μόνο οι καταχωρητές 0, 1 και 16 γιατί αυτοί χρησιμοποιούνται κατά την διάρκεια του test.

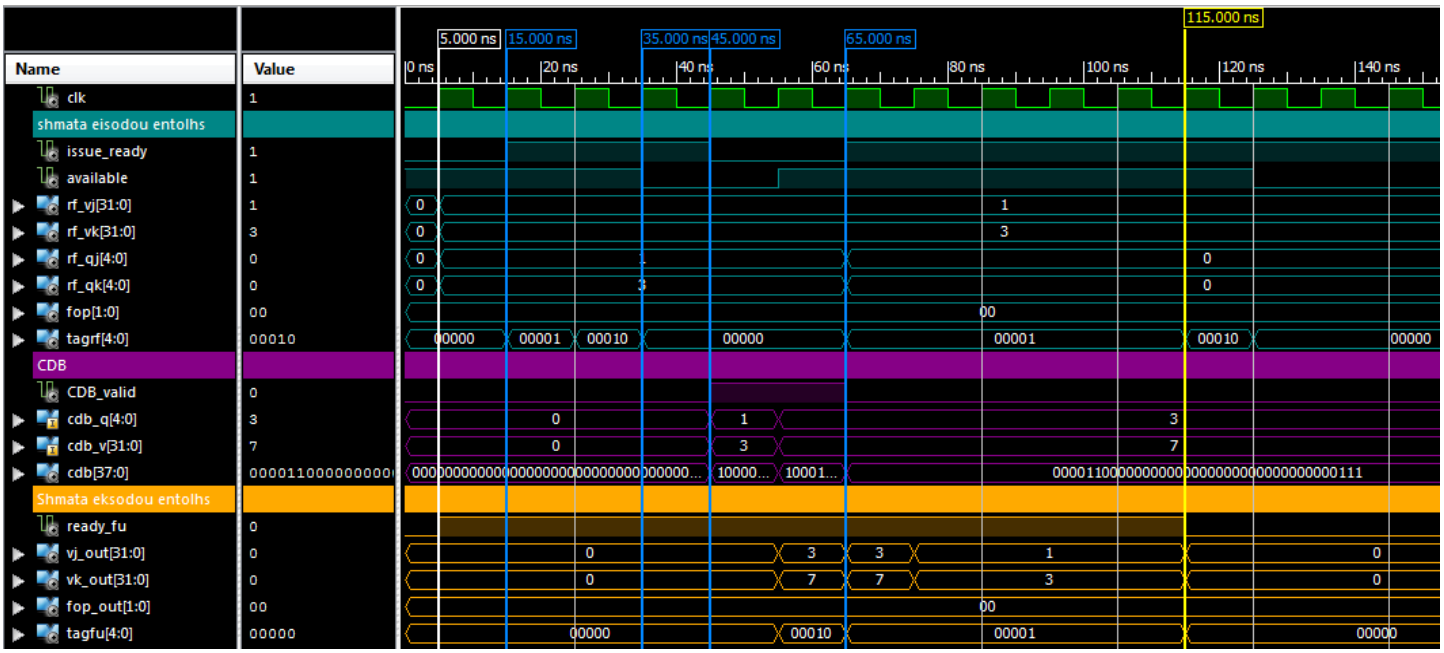
5 ns: Γίνεται έκδοση εντολής (instr_valid = ‘1’) στον καταχωρητή ri = 0 με tag = 3. Στον επόμενο κύκλο φαίνεται ότι το rename πραγματοποιήθηκε καθώς ο Q_reg0 έχει την τιμή 3.

15 ns: Έρχεται έγκυρο αποτέλεσμα από το CDB με tag = 3. Πράγματι στον επόμενο κύκλο ο καταχωρητής V_reg0 έχει το αποτέλεσμα και ο καταχωρητής Q_reg0 μηδενίζεται. Όμως στα 15 ns διαβάζεται ο καταχωρητής 0 (rj = 0) πριν προλάβει να γραφτεί το αποτέλεσμα. Τότε αυτό γίνονται forward και vj = 3 καθώς και το qj = 0.

25 ns: Η εντολή δεν είναι έγκυρη (instr_valid = ‘0’) και δεν γίνεται renaming αφού ο καταχωρητής Q_reg1 δεν αλλάζει τιμή.

45 ns: Ο καταχωρητής 16 έχει tag = 3. Έρχεται αποτέλεσμα από τον CDB με αυτό tag, αλλά ταυτόχρονα και μια καινούργια εντολή για τον καταχωρητή 16 με tag = 15. Τότε προτεραιότητα δίνεται στο renaming και στον επόμενο κύκλο ο καταχωρητής Q_reg16 δεν μηδενίζεται και έχει το νέο tag (15). Επίσης, διαβάζεται ο καταχωρητής στην πύλη rj και το νέο tag γίνεται forward, αφού φαίνεται στο πεδίο qj πριν γραφτεί στον καταχωρητή. Το αποτέλεσμα του CDB γράφεται στον καταχωρητή V_reg16, αλλά είναι αδιάφορο καθώς το αντίστοιχο tag δεν είναι 0 και τα RS θα το αγνοήσουν.

Reservation Station:



Οι κυματομορφές αφορούν το λογικό reservation station. Η μόνη διαφορά που έχει με το αριθμητικό είναι πόσες εντολές μπορούν να αποθηκευτούν και δεν υπάρχει κάποια ουσιαστική αλλαγή στις κυματομορφές. Απλά κάνει ένα κύκλο παραπάνω να γεμίσει και να αδειάσει. Επίσης τα πεδία cdb_value, cdb_q και cdb_valid συνδέονται σε ένα (cdb) κατά την διάρκεια λειτουργίας, αλλά για το test παρουσιάζονται ξεχωριστά για να διαβάζεται πιο εύκολα.

5 ns: Με issue_ready = ‘0’ η εντολή δεν είναι έγκυρη και δεν αποθηκεύεται. Αυτό φαίνεται γιατί το tagrf που επιστρέφει είναι “00000”, όπου είναι η άκυρη τιμή.

15 - 35 ns: Το issue_ready γίνεται ‘1’ και οι εντολές αποθηκεύονται στα block που υποδεικνύει το σήμα tagrf. Επειδή τα δεδομένα έρχονται με μη μηδενικά Q, οι εντολές δεν είναι έτοιμες για εκτέλεση και δεν στέλνονται στο FU (tagfu = “00000”) και ας είναι διαθέσιμος(ready_fu = ‘1’).

35 ns: Όλα τα διαθέσιμα μπλοκ γέμισαν και το σήμα available γίνεται ‘0’ με αποτέλεσμα να μην μπαίνει άλλη εντολή.

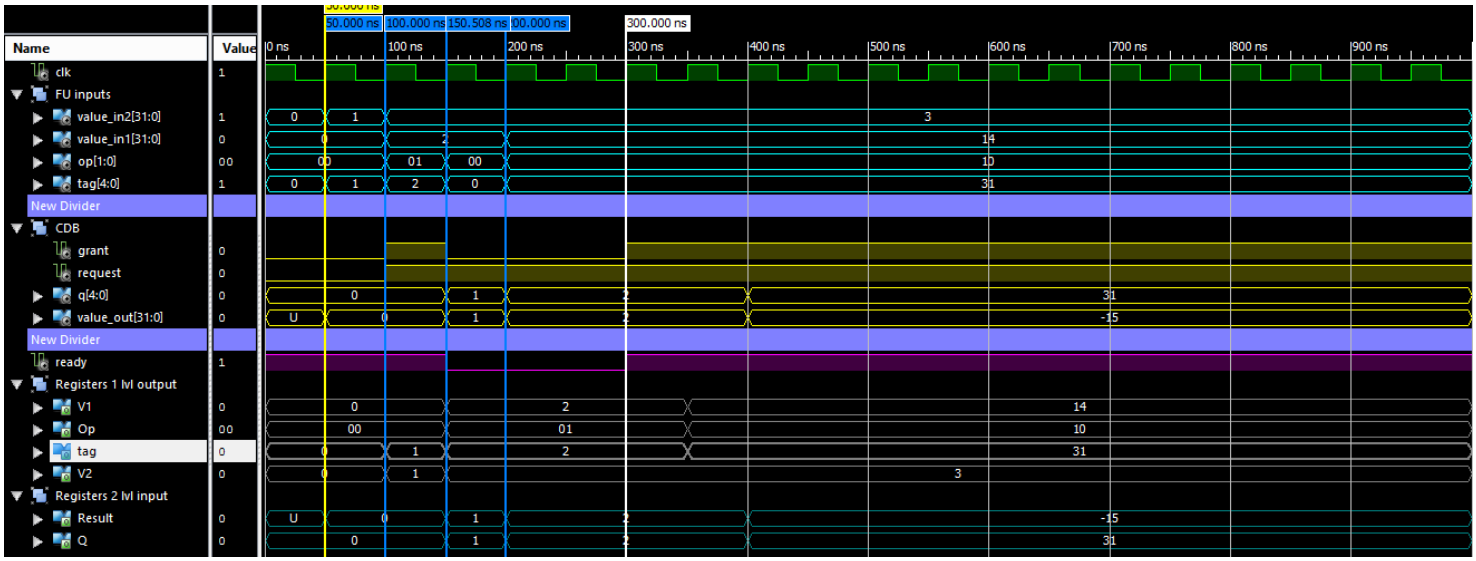
45 - 65 ns: Έρχονται τα αποτελέσματα που περίμεναν οι εντολές μέσω του CDB. Με το που έρχεται το τελευταίο αποτέλεσμα κατευθείαν αποστέλνεται στο FU μια εντολή (tagfu = “00010”). Δηλαδή με το που τελειώσει μια εντολή, αμέσως μπορεί να ξεκινήσει μια άλλη που εξαρτιόταν από αυτή. Έτσι επιτυγχάνεται pipe-line σε περίπτωση εξαρτήσεων και ο επεξεργαστής είναι τουλάχιστον όσο καλός όσο αυτός της οργάνωσης. Επίσης φαίνεται πως τα δεδομένα που στέλνονται για εκτέλεση είναι αυτά που ήρθαν από τον CDB.

65 ns: Για τους επόμενους 5 κύκλους έρχονται εντολές έτοιμες για εκτέλεση (rf_qj = ‘0’, rf_qk = ‘0’). Παρατηρώντας ότι το tagrf είναι ίσο με το tagfu, φαίνεται πως σε ένα block

μπορεί να μπαίνει μία εντολή ενώ ταυτόχρονα βγαίνει μία άλλη. Έτσι τα blocks είναι πλήρως pipe-lined και περνάει μια εντολή ανά κύκλο.

115 ns: Το FU σταματάει να είναι διαθέσιμο (ready_fu = ‘0’), με αποτέλεσμα να μην στέλνεται κάποια εντολή (tagfu = “00000”) και να γεμίζουν όλα τα block εντολές. Με το που γίνει αυτό, το σήμα available γίνεται ‘0’ και η μονάδα δεν δέχεται άλλες εντολές (tagrf = “00000”) .

Function Unit:



0 ns:Το FU βρίσκεται σε αδράνεια και δέχεται σκουπίδια .

50 ns: Ο καταχωρητής στο πρώτο επίπεδο έχει αποθηκεύσει το απαγορευμένο tag οπότε το fu είναι έτοιμο να δεχθεί νέα δεδομένα και δεν αποστέλνεται request προς το cdb. Παράλληλα, δεδομένα εισέρχονται στο fu.

100 ns: Η εισαγωγή της εντολής στα 50 ns έχει αποθηκευτεί στους καταχωρητές πρώτου επιπέδου και το control ελέγχει το tag ώστε να προέρχεται από κάποιο αποδεκτό reservation station. Τότε, στέλνει request στο cdb και αυτό απαντάει με το σήμα grant. Εφόσον το grant είναι ‘1’, αναμένεται στον επόμενο κύκλο να σταλούν τα δεδομένα .Παράλληλα νέα εντολή εισέρχεται στο σύστημα του functional unit με tag = ‘2’.

τιμή A = 0, τιμή B = 1, Op = OR, Result = 1, tag = 1.

150 ns: Το αποτέλεσμα εξάγεται στο cdb έπειτα από καθυστέρηση 2 κύκλων. Το αποτέλεσμα είναι το αναμενόμενο. Επιπλέον, στέλνεται request για την εντολή με tag = ‘2’ αφού είναι και αυτή αποδεκτή, άρα αναμένεται να εξαχθεί στον επόμενο κύκλο. Όμως ο cdb δεν είναι έτοιμος (grant = ‘0’). Τέλος, έρχεται εντολή που δεν είναι αποδεκτή και κόβεται το enable των καταχωρητών πρώτου επιπέδου, ώστε να μην χαθούν τα δεδομένα που υπάρχουν μέσα στο κύκλωμα. Όταν σταματήσει μια εντολή λόγω του CDB σταματάνε και όλες οι έγκυρες επόμενες της.

200 ns: Τιμή A = 3, τιμή B = 2, Op = AND, Result = 2, tag = 2. Επαληθεύεται το αποτέλεσμα της εξόδου. Καθώς το CDB είναι αδύνατον να δεχθεί το αποτέλεσμα της FU η εντολή με tag = 2 δεν ξεφεύγει από το επίπεδο καταχωρητών 2. Οπότε, παραμένει εκεί και στέλνει επανειλημμένα request μέχρι να δώσει grant ο CDB. Κάθε νέα εντολή που έρχεται, δεν εισάγεται καθώς το πρώτο επίπεδο καθυστέρησης είναι κατειλημμένο με έγκυρη εντολή. Επίσης, ενημερώνει το RS να μην στείλει έγκυρες και χαθούν (ready = '0').

300 ns: Το CDB στέλνει grant. Στον επόμενο κύκλο, η εντολή που περίμενε μπορεί να προχωρήσει και νέα δεδομένα μπορούν να εισέλθουν στο πρώτο επίπεδο καταχωρητών.

400 ns : Έπειτα από καθυστέρηση δυο κύκλων εξέρχονται τα δεδομένα . Το cdb είναι ακατάπαυστα έτοιμο οπότε η εισαγωγή και η εξαγωγή δεδομένων δεν διακόπτεται. Εμφανίζεται το tag = 31 με την πράξη not που είχε εισαχθεί στα 300 ns. Το εξαγόμενο αποτέλεσμα είναι ορθό.

Συμπεράσματα

Η εργαστηριακή άσκηση καταλήγει σε σημαντικά αποτελέσματα, καθώς καλύπτεται η τμηματική μελέτη της σχεδίασης του επεξεργαστή διοχέτευσης σύμφωνα με τον αλγόριθμο Tomasulo. Ο συνδυασμός όλων των κομματιών που έχουν αναφερθεί στο κύκλωμα προάγει την ολοκληρωμένη δυναμική του συστήματος. Μπορούν να παρατηρηθούν όλες οι υπηρεσίες που προσφέρουν οι βαθμίδες σε κάθε εξυπηρέτηση εντολής, η ανταπόκριση του συστήματος κατά τη διάρκεια συμφόρησης των μονάδων (πχ. Reservation Station, FUs), καθώς επίσης και περιπτώσεις που είναι άξιες ενδιαφέροντος προς το σύστημα (corner-cases).

Σημείωση: Επισυνάπτουμε σε ξεχωριστούς φακέλους τις κυμματομορφές και τα σχεδιαγράμματα, ώστε να μπορούν να διαβαστούν σε περίπτωση που δεν φαίνονται καλά στο pdf.