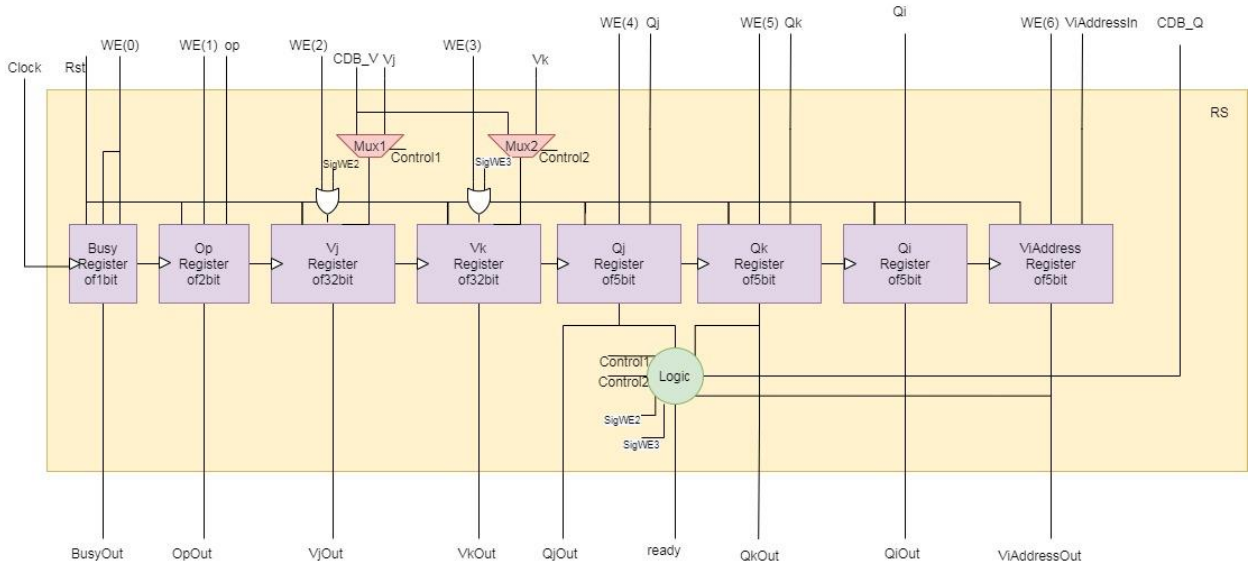
ΑΝΑΦΟΡΑ tomasulo 1ο μέρος

Κουρκουλος Αγγελος ΑΜ:2017030111

Στόχος της εργασίας ήταν η υλοποίηση του αλγορίθμους του tomasulo την εκτέλεσε μερικών εντολών σε αυτόν καθώς και η επιβεβαίωση της ορθής λειτουργίας του . Για την κατασκευή του λοιπόν δημιουργήθηκαν τα επιμέρους συστήματα , Reservation Station ,Register File , Common Data Bus, Issue , τα οποία συνδυάζοντας τα κατάλληλα κατασκευάζεται το τελικό κύκλωμα. Ο αναλυτικός τρόπος κατασκευής του κάθε υποσυστήματος και η επαλήθευση της ορθής λειτουργίας τους καταγράφεται παρακάτω:

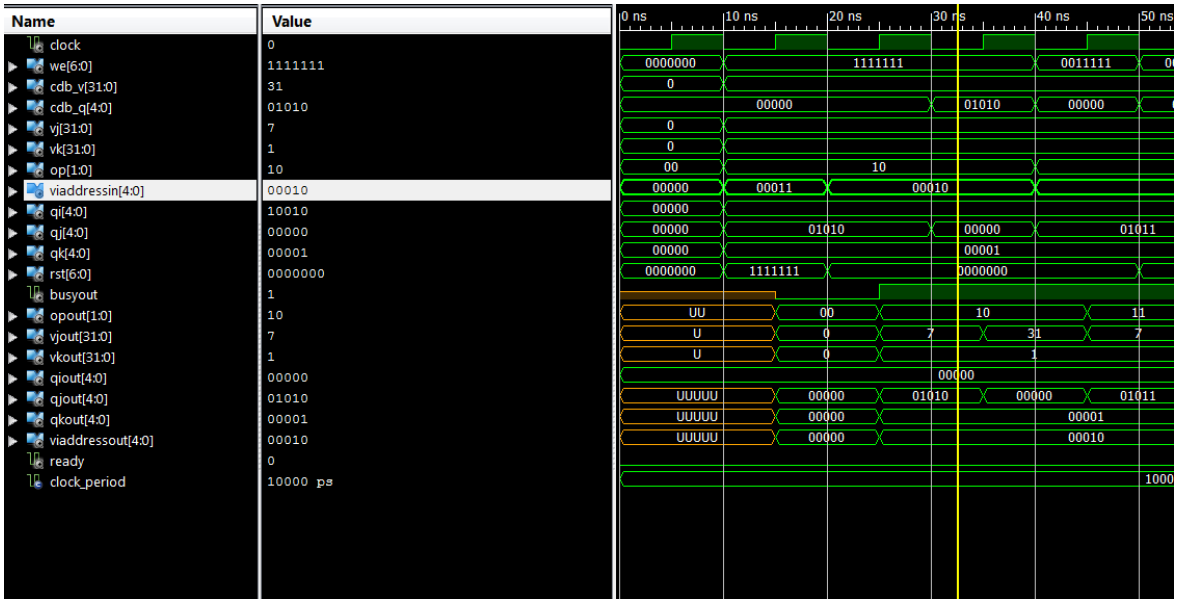
ReservationStation (RS):

Σκοπός του reservation station είναι η προσωρινή αποθήκευση χρήσιμων δεδομένων. Για την κατασκευή αυτής της μονάδας χρησιμοποιήθηκαν 8 καταχωρειτες οι οποίοι είναι υπεύθυνοι για την αποθήκευση των απαραίτητων πεδίων που χρειαζόμαστε, 2 πολυπλεκτες οι οποίοι επιλέγουν ποια τιμή θα γραφεί στα παιδιά value του RS, καθώς και λογική η οποία δημιουργεί τα απαραίτητα σήματα ελέγχου για τους πολυπλεκτες και τους καταχωρειτες.



Στην προσομοίωση αυτής της μονάδας αρχικά έγινε rst=1 για κάθε καταχωρητή ώστε να αρχικοποιηθουν όλα στο 0 και στη συνέχεια μπήκαν τιμές στις εισόδους

του κάθε καταχωρητή για να επαληθευτεί ότι η αποθήκευση των δεδομένων γίνεται σωστά. Έπειτα ελέγχτηκε η περίπτωση όπου το σήμα cdb_q είναι ίδιο με την τιμή του καταχωρητή της τιμής Qj Και επαληθεύτηκε ότι η επιλογή του πολυπλεκτη για το Vj γίνεται σωστά στην τιμή cdb_v . Στη συνέχεια μετά από ελέγχω όλων των άλλων σημάτων όπως WE κτλ επιβεβαιώθηκε η ορθότητα της λειτουργίας του
Reservation Station

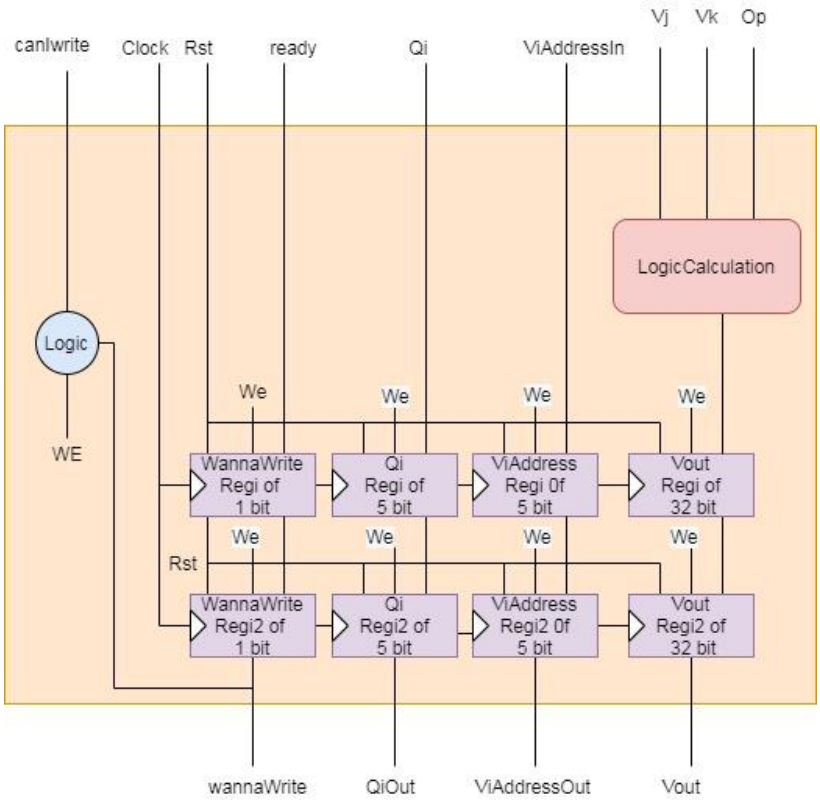


Functional Unit (FU):

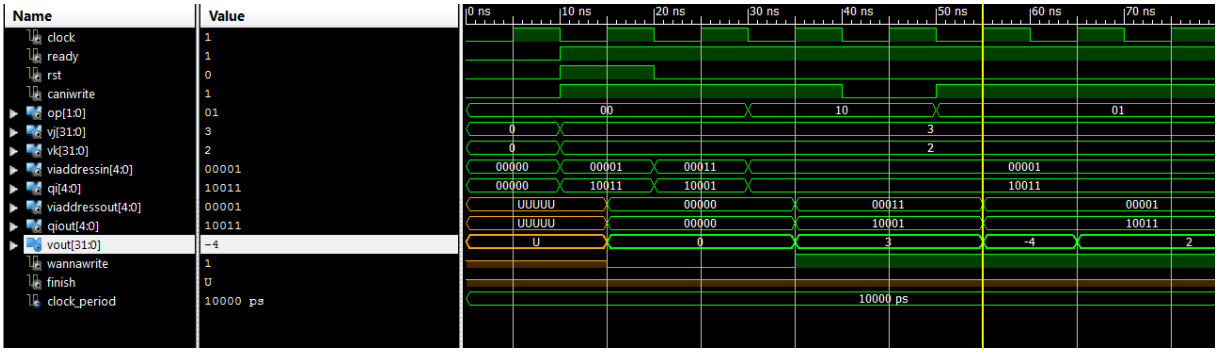
Υπαρχουν 2 ειδών functional unit το LogicFU και το ArithmeticFU. Σκοπός του καθε Functional Unit είναι η εκτέλεση μιας λογικής η αριθμητικής πράξης και η δημοσίευση του αποτελέσματος της μετά από 2 και 3 κύκλους καθυστέρησης αντίστοιχα για κάθε FU. Τα FU ένα κύκλο πριν την δημοσίευση των αποτελέσματων ζητάνε άδεια από κάποια άλλη μονάδα (CDB) και σε περίπτωση που η απάντηση είναι θετική δημοσιεύουν τα αποτελέσματα αλλιώς περιμένουν ένα κύκλο παραπάνω . Αυτό πετυχαίνετε έχοντας ένα κομμάτι που κάνει την αριθμητική πράξεις καθώς και κάποιους καταχωρείτε σε σειρά ώστε να εξάγουν τα αποτελέσματα και τα απαραίτητα στοιχεία μετά από συγκεκριμένο αριθμό κύκλων καθώς υπάρχει και λογική η οποία ελέγχει αν η απάντηση για τη

δημοσίευση των αποτελεσμάτων είναι θετική και δίνει τα κατάλληλα σήματα ελέγχου στο σύστημα.

LogicFU:



Για αυτή την προσομοίωση αρχικά δόθηκε ως είσοδος rst=1 για να μηδενιστεί κάθε καταχωρητης και στη συνέχεια έγιναν 3 πράξεις για τις τιμές Vj= 0..011 και Vk= 0..010 . Η πρώτη ήταν and η δεύτερη Not Και η τρίτη or. Τη στιγμή που δημοσιευόταν όμως η τιμή της and δόθηκε αρνητική απάντηση για την δημοσίευση του αποτελέσματος .



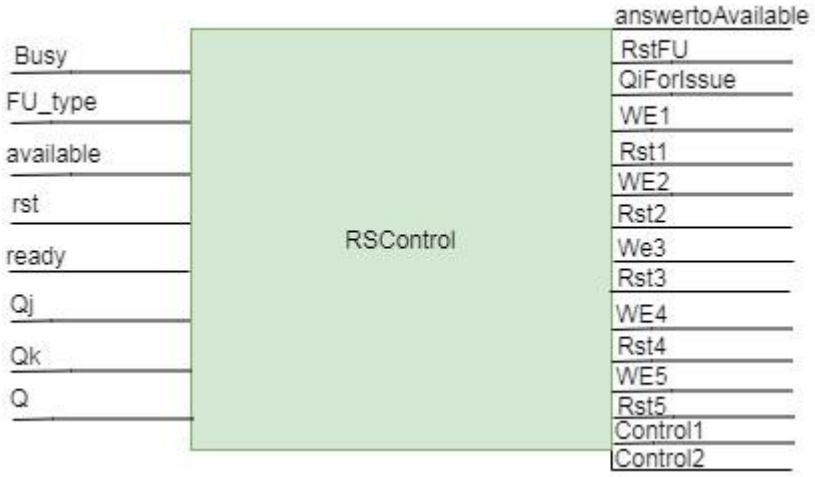
Αυτό που παρατηρήθηκε στην προσομοίωση είναι ότι τα αποτελέσματα διεξήχθησαν σωστά με ένα κύκλο καθυστέρηση καθώς πάγωσα το αποτέλεσμα της and για ένα κύκλο αφού δεν είχε έγκριση για δημοσίευση. Τα αποτελέσματα ήταν αυτά που περιμέναμε το οποίο αποδεικνύει την ορθότητα της FU. (Τα 32bita νούμερα για τις πράξεις απεικονίζονται σε δεκαδικό αναπαράσταση)

ArithmFU:

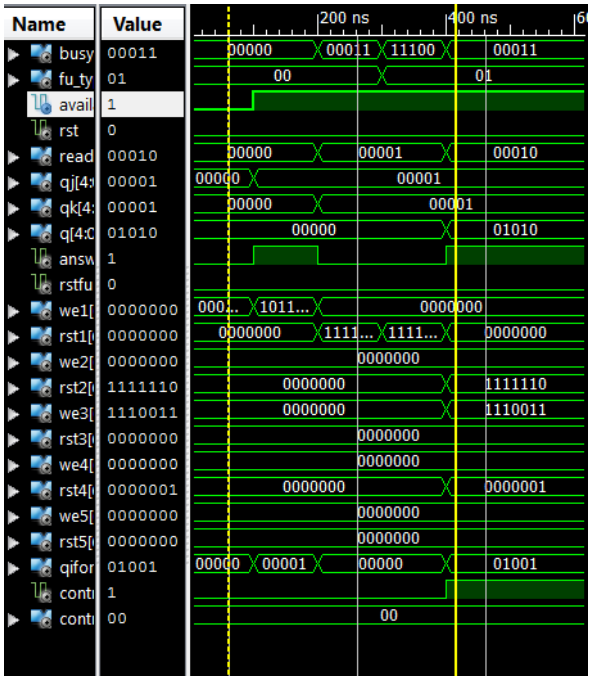
Το ArithmeticFU είχε ακριβώς την είσαι τιμή με το Logic απλά είχε ένα επίπεδο καταχωρητων παραπάνω το οποίο καθυστερούσε ένα ακόμα κύκλο την έξοδο του κυκλώματος. Η προσομοίωση ήταν τελείως αντίστοιχη με την παραπάνω και όπως και τα αποτελέσματα τα οποία περιμέναμε να δούμε.

RSControl:

Το RScontrol είναι η μονάδα που είναι υπεύθυνοι για δημιουργία όλων των σημάτων ελέγχου του RS και FU και απάντηση προς τις υπόλοιπες μονάδες για το αν υπάρχει χώρος δηλαδή ελεύθερο Reservation Station ώστε να μπορέσει να γίνει issue Νέα εντολή. Ποιό συγκεκριμένα αν υπάρχει διαθέσιμο RS σύμφωνα με τον τύπο πράξης που του ζητείται να γίνει ,ενεργοποιεί τα κατάλληλα WE των καταχωρητων και όταν το RS είναι σε κατάσταση ready το αφήνει να δώσει τα περιεχόμενα του στο FU Και κάνει rst όλους τους καταχωρείτε του Εκτός του busy . Όταν η FU τελειώσει την πράξη το RSControl είναι υπεύθυνοι να μηδενιστεί και το busy του συγκεκριμένου RS ωστε να μπορέσει να το χρησιμοποιήσει για την επόμενη εντολή . Το RSControl είναι υλοποιημένο με ακολουθιακή λογική και οι έξοδοι του εξαρτώνται από την κατάσταση των RS και τις εισόδους από το RF Και Issue.



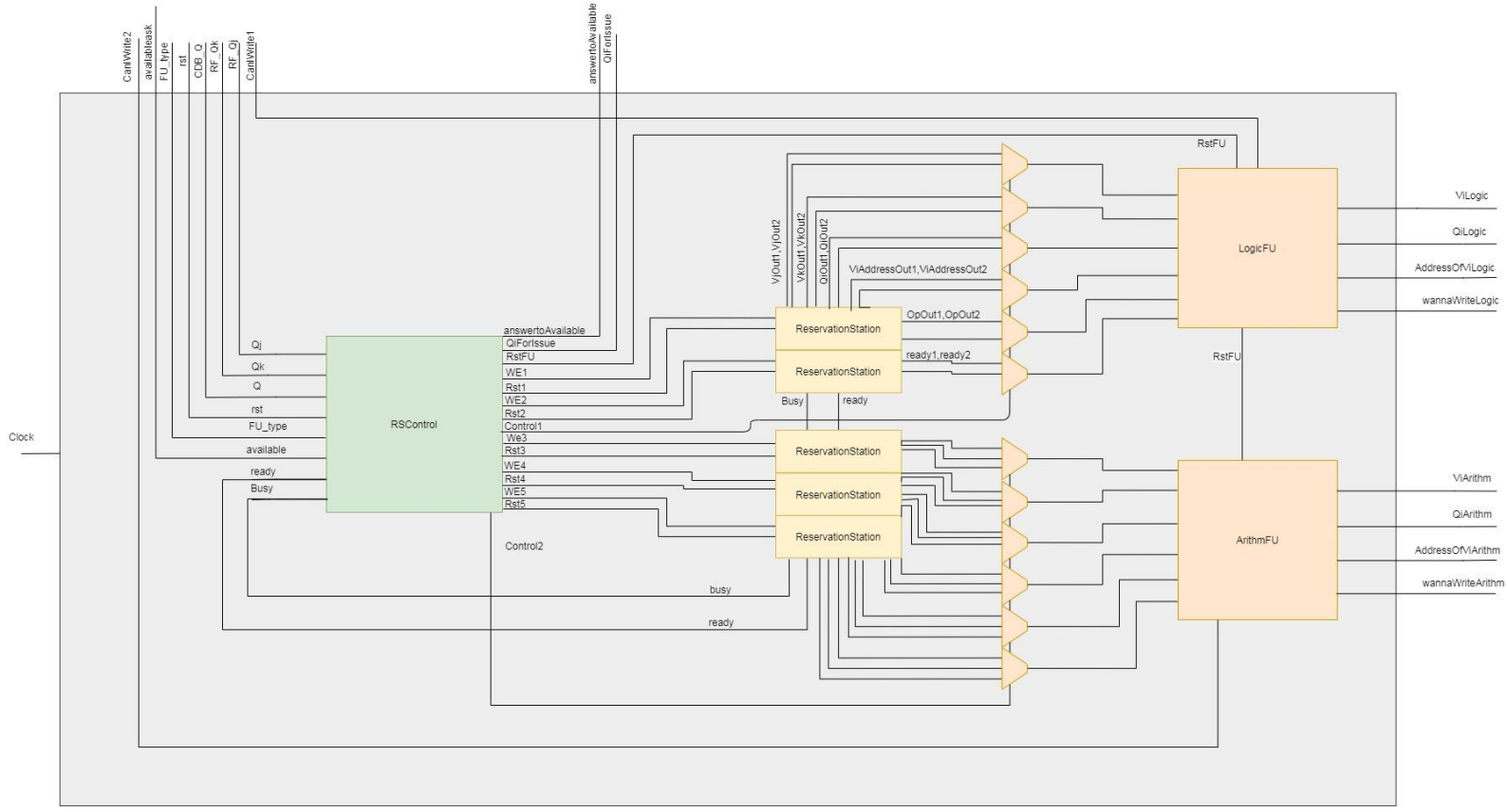
Για την προσομοίωση αρχικά ζητήθηκε να γίνει μια λογική πράξη και δόθηκαν είσοδοι έτσι ώστε να ελέγξουμε αν ενεργοποιούνται σωστά τα WE του κατάλληλου RS .Στη συνέχεια ζητήθηκε να γίνει πάλι λογική πράξη αλλά αυτή τη φορά όλα τα σήματα busy ήταν ενεργοποιημένα οπότε δεν υπήρχε χώρος για την αποδοχή της εντολής καθώς κάποιο σήμα ready κάποιου RS ήταν ενεργοποιημένα και έτσι επαληθεύτηκε η σωστή λειτουργία της απάντησης available και rst που δίνει το RScontrol. Αντίστοιχα έγινε και για τις αριθμητικές πράξεις. Τέλος περάστηκε μια τιμή ως είσοδος στο πεδίο Q που είναι η έξοδος του FU για την πράξη που έχει πραγματοποιηθεί και ελέγχτηκε αν γίνετε σωστά το rst του busy του σωστού RS ώστε να αποδεσμευτεί.



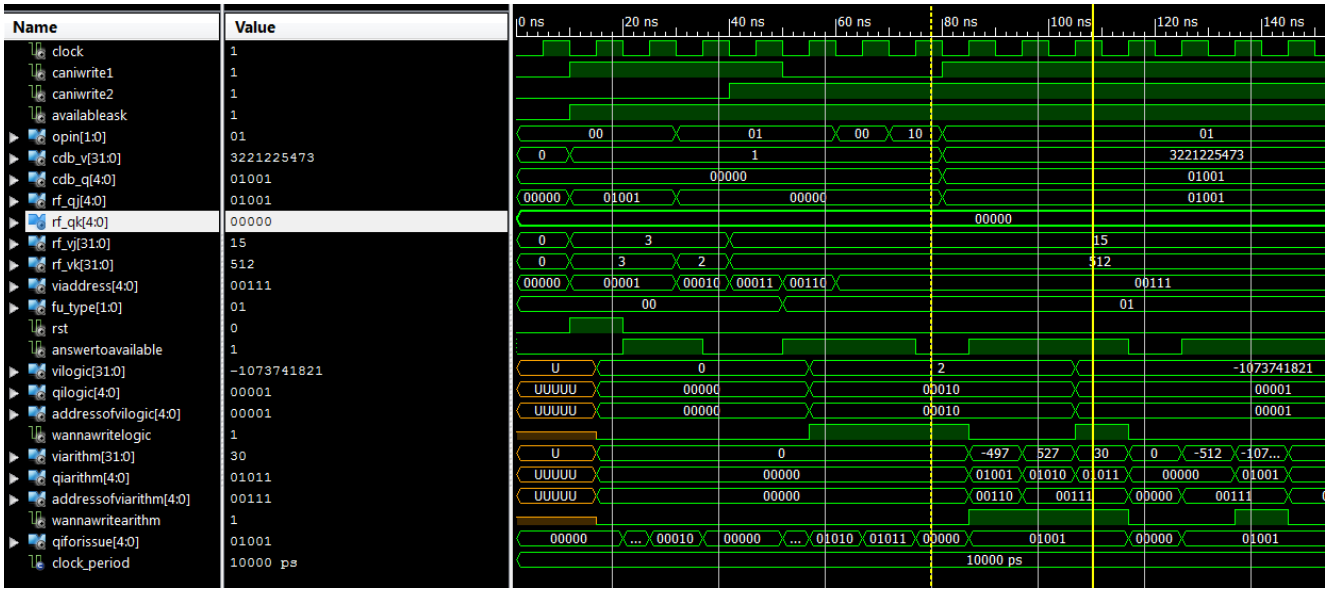
Η λειτουργία του RSControl σύμφωνα με την προσομοίωση είναι αυτη που περιμέναμε.

CompleteRS:

Για τη μονάδα CompleteRS χρησιμοποιήσαμε 5 στιγμιότυπα από τα Reservation Station,2 για την αποθήκευση λογικών και 3 για αριθμητικών πράξεων, μερικοι πολυπλεκτες για την επιλογή των σωστών σημάτων από τα RS προς τα FU , ενα Logic Functional Unit για τον υπολογισμό λογικών πράξεων, ένα Arithmetic Functiona Unit για τον υπολογισμο αριθμητικών πράξεων και ένα RSControl για την παραγωγή σωστών σημάτων ελέγχου των παραπάνω.



Για την προσομοίωση του CompleteRS αρχικά ενεργοποιήθηκε το σήμα rst =1 για την Αρχικοποίηση των καταχωρειτων σε 0 . Έπειτα δοθήκαν είσοδοι για 3 λογικές πράξεις από τις οποίες η 1 περνάει στο καταχωρειτή Qj μια τιμή με αποτέλεσμα να μην μπορεί να εκτελεστεί άμεσα ενώ περιμένουμε η 3 να μην γίνει δεκτή λόγο έλλειψης χώρου (RS) . Στη συνέχεια μπήκαν είσοδοι για την εκτέλεση 4ων αριθμητικών πράξεων οι οποίες περιμένουμε να εκτελεστούν κανονικά Εκτός την τελευταία η οποία εκτός του ότι δεν γίνεται δεκτή λόγο χώρο έχει την τιμή που περιμένει η πρώτη λογική πράξη στις εισόδους από το CDB οπότε ξεκινάει η εκτέλεση της . Μετά από 2 κύκλους προβλέπετε να έχουν τελειώσει όλες οι πράξεις και να έχουν δημοσιευθεί τα σωστά αποτελέσματα

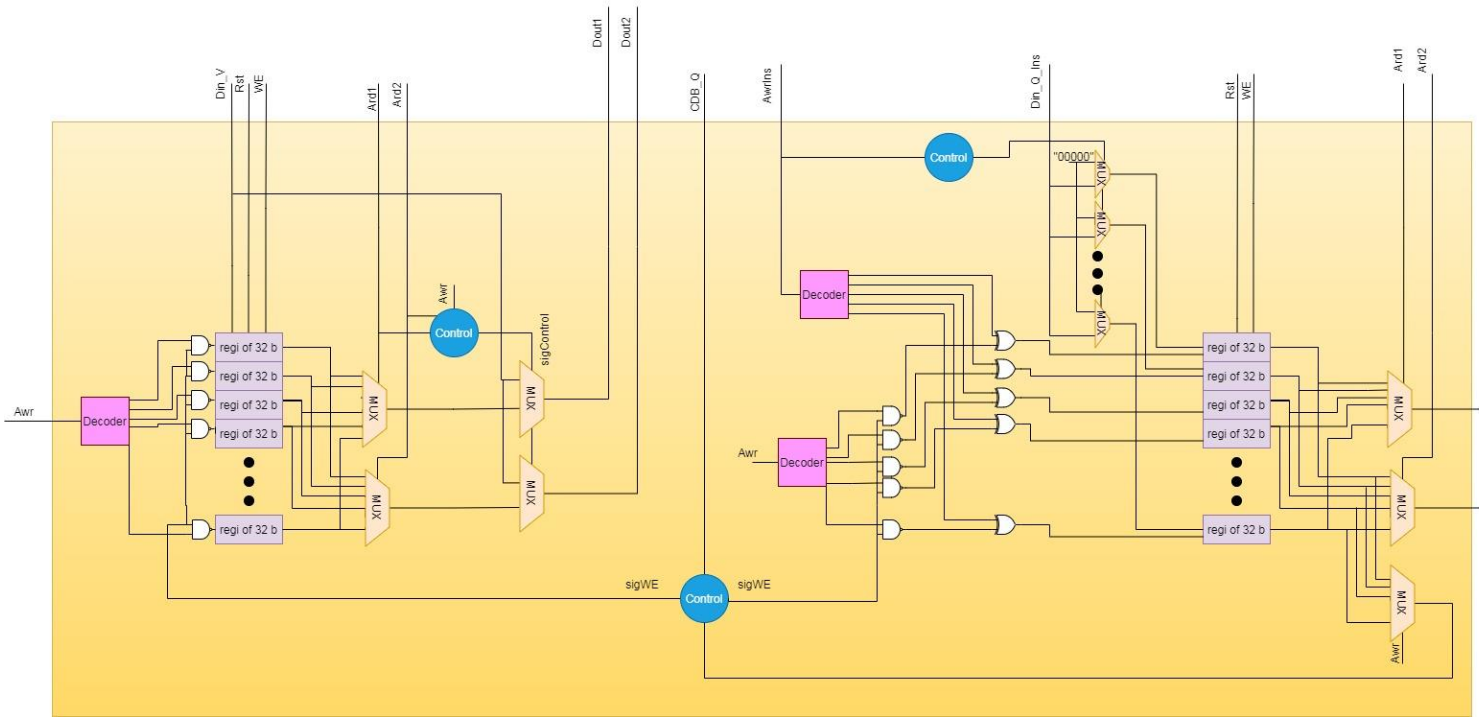


Αυτό που βλέπουμε στην προσομοίωση είναι αυτό που προβλεπόταν δηλαδή έγινε rst στο 2ο κύκλο ρολογιού όπως φαίνεται στο διάγραμμα. Δόθηκαν ως είσοδοι 3 λογικές πράξεις από τις οποίες η 2 τελείωσε στον 6 κύκλο η 3 απορρίφθηκε ενω η πρώτη τελείωσε στον 11 κύκλο δηλαδή 2 κύκλους μετά από τη στιγμή που ήρθε η είσοδος που περίμενε . Επίσης δόθηκαν οι είσοδοι για 4 αριθμητικές πράξεις στον 6 7 8 9 κυκλο οι οποίες τελείωσαν 3 κύκλους μερα την εισαγωγή τους άκρως την τελευταία που απορρίφθηκε. (Τα 32bita νούμερα για τις πράξεις απεικονίζονται σε δεκαδικό αναπαράσταση)

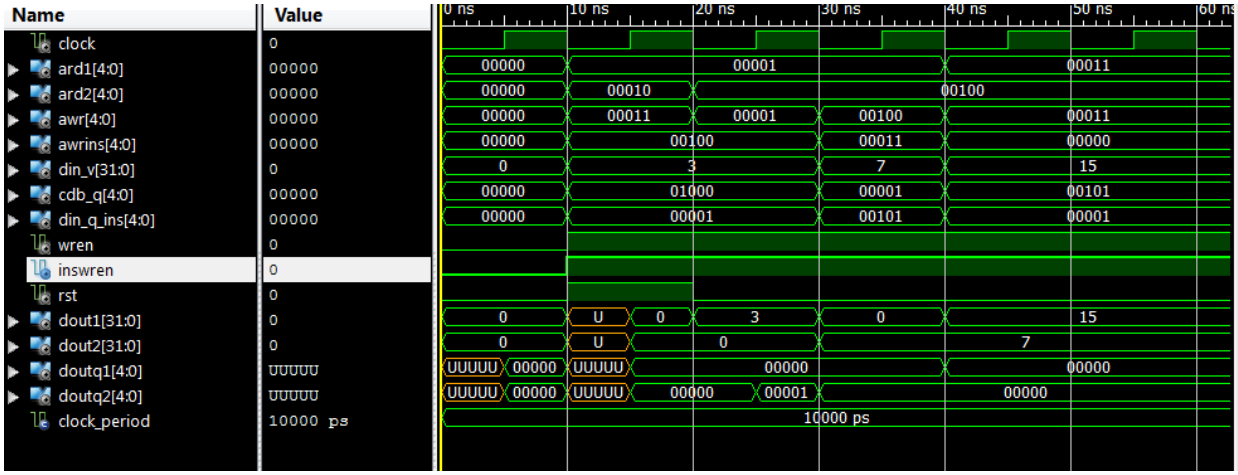
Register File (RF):

Για την υλοποίηση της μονάδας Register File χρειάστηκαν 64 καταχωρείτε των 32 bit , οι 32 για την αποθήκευση των Value Και οι υπόλοιποι για την αποθήκευση των Q δηλαδή το tag του reservation station που εκτελεί την πράξη που θα αποθηκευτεί στο αντίστοιχη παιδιά V . Ο καταχωριτής 0 από κάθε πεδίο μένει πάντα στην τιμή 0 . Ακόμα χρειάστηκαν ένας Decoder για την ενεργοποίηση του WE του κατάλληλου καταχωρητή από τα Value και 2 ακόμα Decoders Για την ενεργοποίηση των κατάλληλων WE ώστε να γινει εγγραφή από νέα εντολή αλλά και μηδενισμός αν έχει τελειώσει κάποια παλιότερη εντολή στους κατάλληλος registers Q . Επίσης χρησιμοποιήθηκαν 4 πολιπλεκτς για την επιλογή των εξόδων από τους καταχωρείτες V, από τους οποίους οι 2 είναι υπεύθυνοι για την τεχνική fall-through . Χρειάστηκαν 32 πολυπλεκτες ακόμα για την επιλογή του Din_Q_ins η του 00000 για την

εγγραφή από νέα εντολή η το μηδενισμό από παλιότερη που τελείωσε στους καταχωρειτες Q , 2 πολυπλεκτες για τις εξόδους των Q και ένας για την επιλογή του καταχωρητή της διευθύνσεις που πάει να γράφει για την σύγκριση και απόφαση του αν πρέπει να γράφει το αποτέλεσμα η όχι. Τέλος δημιουργήθηκαν κάποια μικρά control για τη διεξαγωγή των σημάτων ελέγχου των πολυπλεκτων και WE καθώς και μερικές πύλες για τον συνδυασμό των σημάτων από τους Decoders και των control.

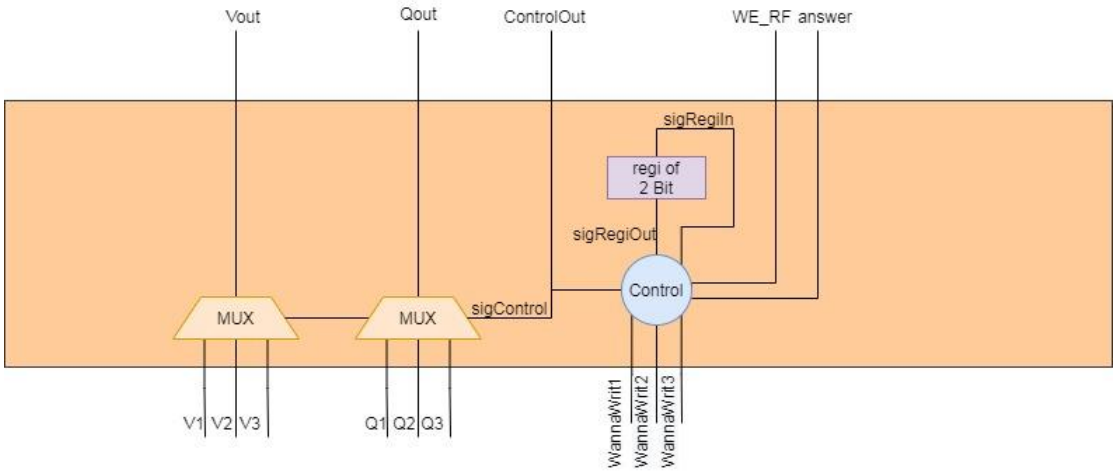


Στην προσομοίωση μ αρχικά στον κύκλο 2 το rst γίνεται 1 και μηδενίζει όλους τους καταχωρείτε. Έπειτα στον 3 κύκλο διαβάζουμε τους καταχωρείτε 1 και 4 προσπαθώντας να γράψουμε στον καταχωρητή 1 και παρατηρούμε ότι το αποτέλεσμα περνάει στην έξοδο λόγο της τεχνικής fall through αλλά δεν γράφεται πραγματικά στον καταχωρητή γιατί το tag Που έχει αποθηκευμένο στον αντίστοιχη καταχωρητή του Q είναι 00000 . Στον ίδιο κύκλο γράφουμε στον καταχωρητή 4 των Q την τιμή 1 και παρατηρούμε ότι το έχουμε στην έξοδο μετά την ακμή του ρολογιού όπως περιμέναμε. Στη συνέχεια στον επόμενο κύκλο συνεχίζουμε να διαβάζουμε τους ίδιους καταχωρειτες και προσπαθούμε να γράψουμε στον καταχωρητή 4 των τιμή 7 έχοντας στην είσοδο του cdb_q την τιμή 1 που είχε καταχωρηθεί στον προηγούμενο κύκλο στον αντίστοιχο καταχωρητή q και παρατηρούμε ότι το αποτέλεσμα βγαίνει κανονικά στην έξοδο και σε αυτόν το κύκλο λόγο του fall through και στον επόμενο οπότε το αποτέλεσμα γράφεται κανονικά όπως θέλαμε. Επίσης γράφουμε στον ίδιο κύκλο τον καταχωρητή 3 των Q την τιμή 00101 και συνεχίζεται η εγγραφή και ανάγνωση των καταχωρητή κατά αυτό τον τρόπο και στους υπόλοιπους κύκλους.

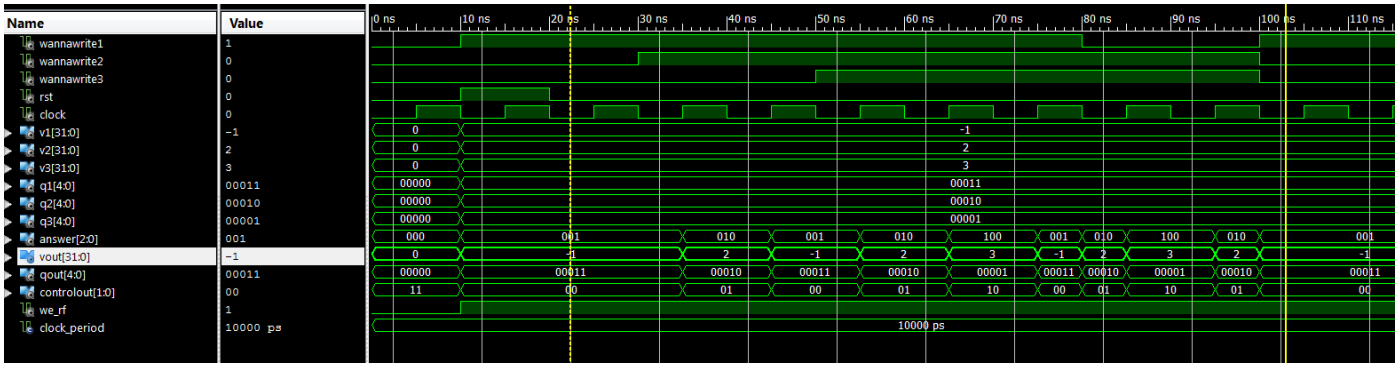


Common Data Bus(CDB):

Για το Common Data Bus χρησιμοποιήσαμε 2 πολυπλεκτες για την επιλογή V και Q από τις λειτουργικές μονάδες, ένα καταχωρητή 2 bit Για να θυμόμαστε σε τι κατάσταση είμαστε και σε ποιον πρέπει να δώσουμε άδεια για να γράψει στον επόμενο κύκλο και μια λογική control που συνεργάζεται με τον καταχωρητή για να δώσει τα κατάλληλα σήματα ελέγχου στους πολυπλεκτες ,να διαχειριστεί τα αιτήματα και να απαντήσει στις λειτουργικές μονάδες . Ο βασικός τρόπος που το πετυχαίνει αυτό το control είναι βάζοντας διαφορετική κατάσταση στο register ανάλογα με το πόσες και ποιες συγκρούσεις υπήρξαν στις αιτήσεις που του ήρθαν για γράψιμο και αποδέχεται μια μια κυκλικά τις αιτήσεις αλλάζοντας κατάσταση στον register μέχρι να τελειώσουν και έπειτα ξανακοιταει ποιες αιτήσεις του ξαναήρθαν

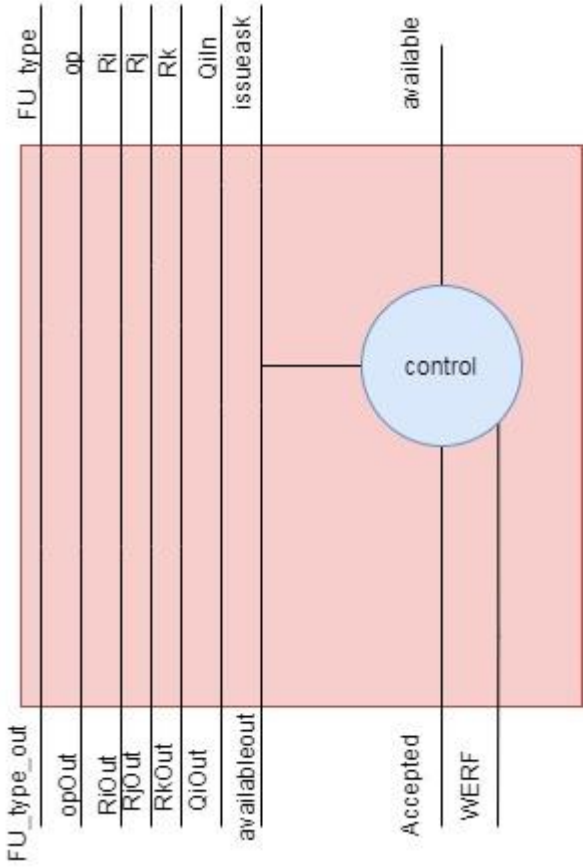


Για την προσομοίωση κάναμε ένα reset Στον κύκλο 2 για να Αρχικοποίηθει ο register Και έπειτα υπήρξε 1 μονο αίτηση για γράψιμο στον CDB οπου το κοντρόλ της έδωσε άδεια άμεσος σε και γράφτηκε η τιμή -1 στο σωστό σήμα στο bus . Στη συνέχεια στον κύκλο 4 υπήρξε σύγκρουση μεταξύ πρώτης και δεύτερης μονάδας οπότε το CDB δέχτηκε πρώτα την πρώτη αίτηση και αριστερά την δεύτερη στον επόμενο κύκλο . Μετά υπήρξε τριπλή σύγκρουση οπότε οι αιτήσεις έγιναν δεκτές με τη σειρά 1-2-3 στους επόμενους κόκκους και τέλος υπήρξε μια διπλή σύγκρουση στις 2 τελευταίες μονάδες ο που έγιναν δεκτές με τη σειρά 2-3 και επαληθεύτηκε η λειτουργία της μονάδας .



Issue:

Η μονάδα issue πρακτικά είναι αυτή που απαντάει στον έξω κόσμο αν η επόμενη εντολή μπορεί να γίνει δεκτή σύμφωνα με τις διαθέσιμες θέσεις που υπάρχουν στο RS , δέχεται ως είσοδο μια αποκωδικοποιημένη εντολή και την περνάει στις κατάλληλες μονάδες τα στοιχεία που χρειάζονται για την εκτέλεση της.

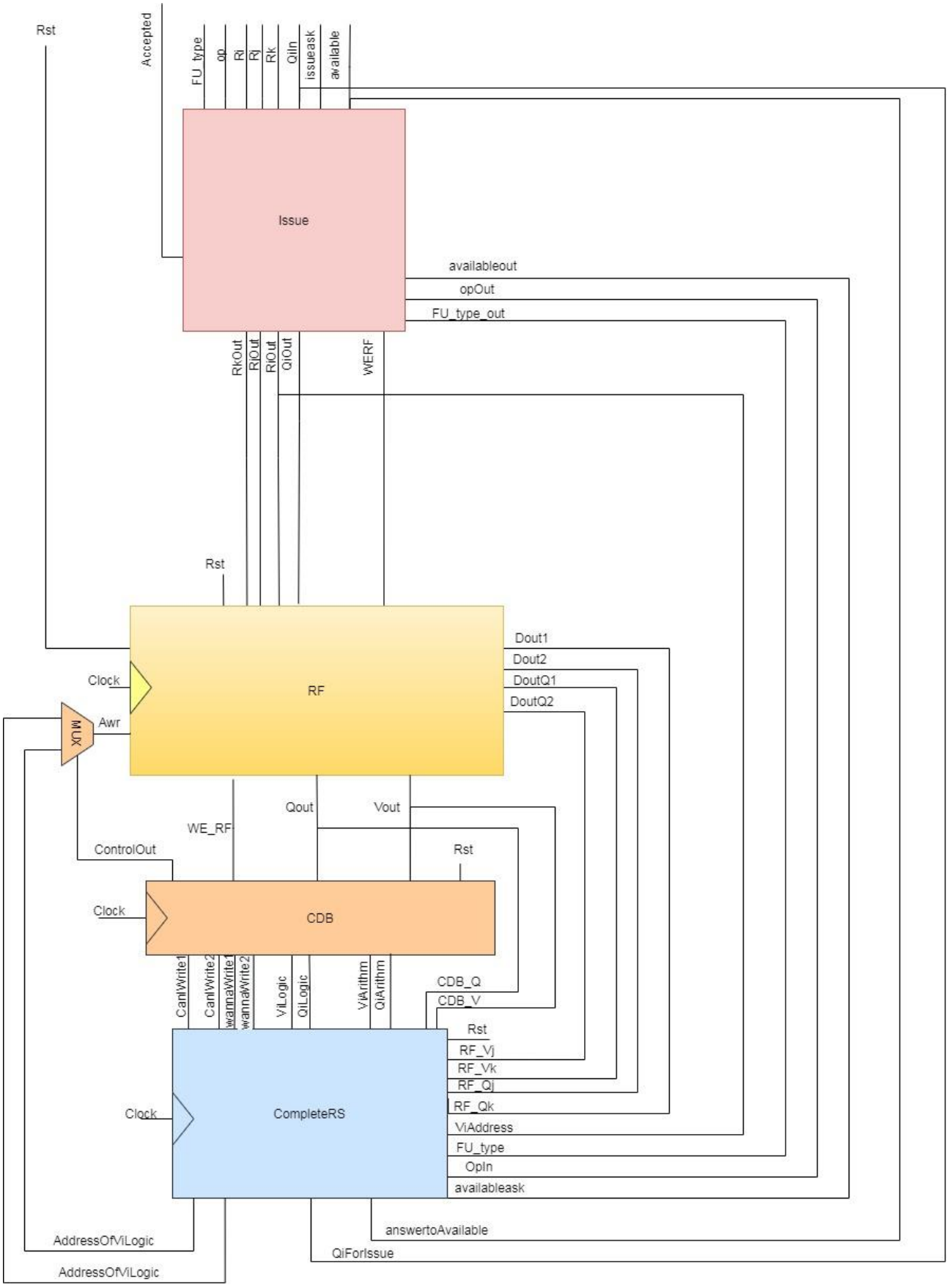


Αυτό που ελέγχουμε στην προσομοίωση είναι το αν λειτουργεί σωστά το control του Issue Δηλαδή αν τα σήματα accepted και WERF βγαίνουν σωστά στην έξοδο σύμφωνα με την απάντηση που δέχεται στην είσοδο available για το αν υπάρχουν ελεύθερα RS από το RScontrol

Αρχικά είχαμε το σήμα available που δείχνει αν υπάρχει ελεύθερο RS =0 και βλέπουμε ότι η απάντηση του Issue accepted είναι 0 . Μόλις το σήμα available γίνει ένα παρατηρούμε ότι και η απάντηση accepted Καθώς και το WERF γίνονται ένα όπως θέλαμε . Τέλος βλέπουμε ότι όλα τα υπόλοιπα σήματα περνάνε σωστά στην έξοδο οπότε η μονάδα μας δουλεύει σωστά.

Tomasulo:

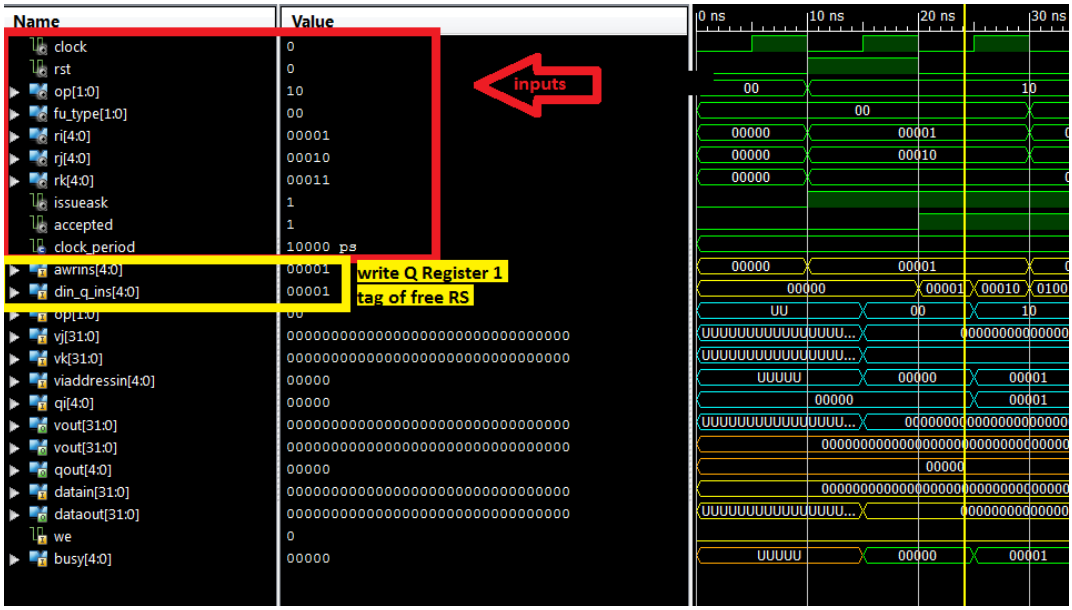
Για το συνολικό κυκλομα του tomasulo συνδυάσουμε 1 στιγμιότυπο της μονάδας Issue ,1 της μονάδας RF, 1 του CDB ,ένα του CompleteRS και ένα πολυπλεκτη . Συνδέουμε τις εξοδους του Issue στις κατάλληλες πόρτες του RF για την ανάγνωση καταχωρητων και στις πόρτες του CompleteRS για να πάρει τις απαραίτητες πληροφορίες που χρειάζεται για την εκτέλεση της πράξης αλλά και για την απάντηση της ύπαρξης χώρου. Ακόμα συνδέουμε τις εξόδους του RF με το CompleteRS για να μπορεί να παίρνει τις τιμές για τις πράξεις. Επίσης συνδέουμε το Complete RS με το CDB ώστε να γίνεται αίτηση από της λειτουργικές μονάδες και να μπορούν να μεταδίδουν πληροφορίες όταν τελειώσουν την πράξη που εκτελούν και με τη χρήση ενός πολυπλεκτη δίνεται η τιμή της διεύθυνσης για γράψιμο στο RF από το ComoleteRS.Τελος το CDB συνδέεται με το RF ώστε να μεταφερεται και να γράφεται η πληροφορία από το Complete RS στο RF και να ολοκληρώνεται η εντολή


Rst
Accepted
FU_type
op
Ri
Rj
Rk
Qiin
issueask
available
Issue
availableout
opOut
FU_type_out
RkOut
RjOut
RiOut
QiOut
WERF
Rst
Dout1
Dout2
DoutQ1
DoutQ2
Clock
RF
MUX
Awr
Qout
Vout
WE_RF
ControlOut
Rst
Clock
CDB
CanIWrite1
CanIWrite2
wannaWrite1
wannaWrite2
ViLogic
QiLogic
ViArithm
QiArithm
CDB_Q
CDB_V
Rst
RF_Vj
RF_Vk
RF_Qj
RF_Qk
ViAddress
FU_type
OpIn
availableask
Clock
CompleteRS
answertoAvailable
QiForIssue
AddressOfViLogic
AddressOfViLogic

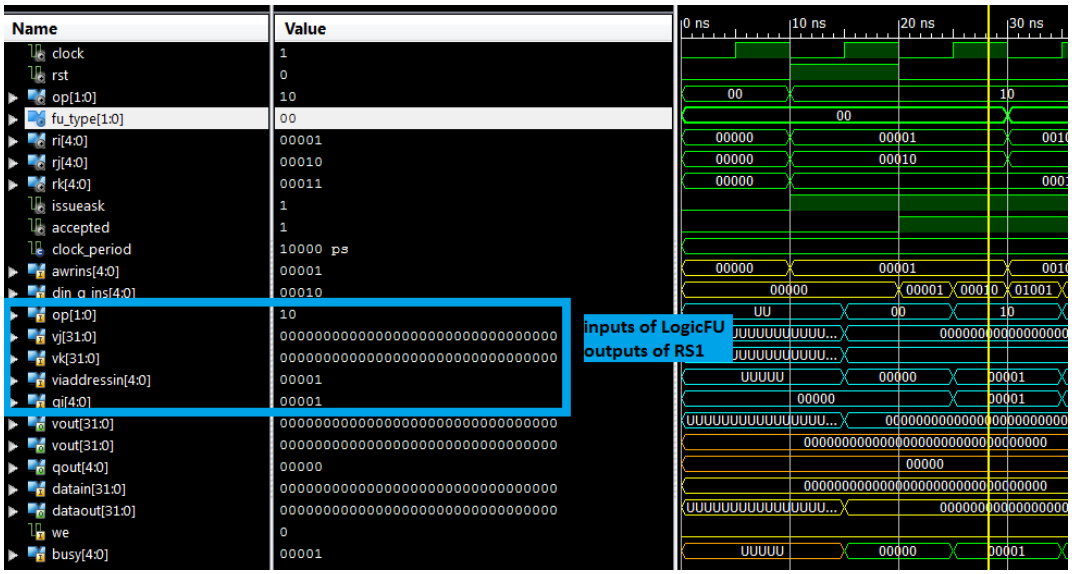
Στην προσομοίωση του συνολικού κυκλώματος μετά το μηδενισμό όλων των καταχωρητή με τη χρήση του reset εκτελείτε η λογική πράξη Not

```
Rst <='0';
        op <="10";
        FU_type <="00";
        Ri <="00001";
        Rj <="00010";
        Rk <="00011";
        Issueask<='1';
```

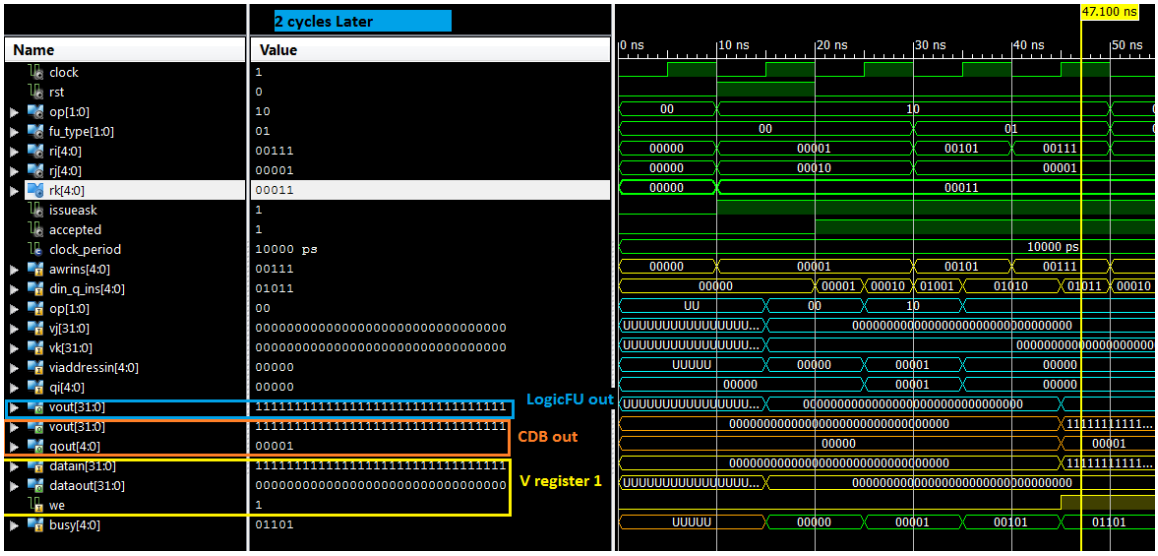
Αρχικά βλέπουμε ότι εισέρχονται στον RF τα σήματα για την δέσμευση του καταχωρείτη στης θέσεις 1 από το RS με tag 00001

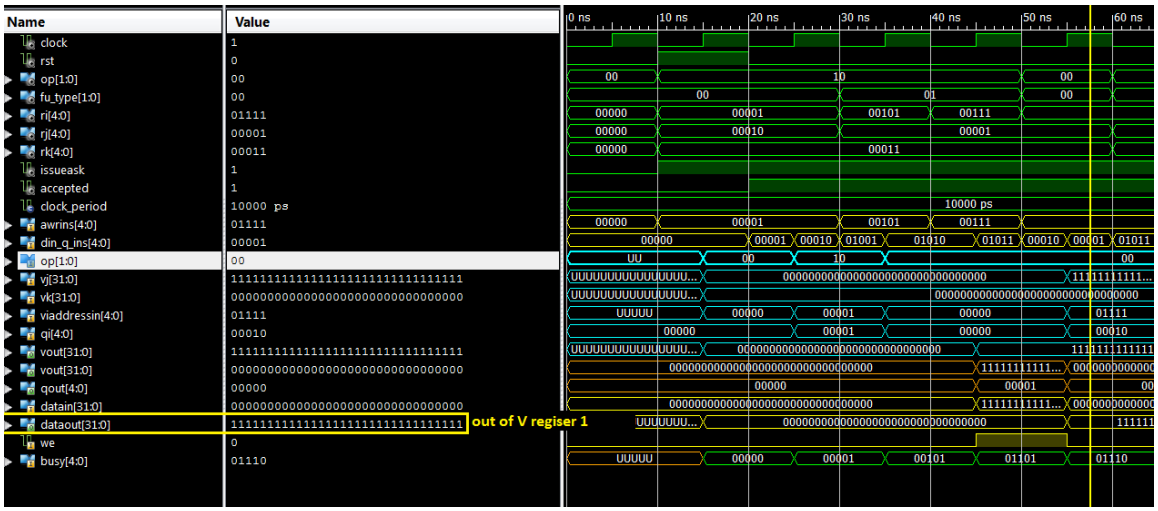


Στον πρώτο παλμό του ρολογιού μετά την έκδοση της εντολής βλέπουμε τις εξόδους του 1 RS και την είσοδο τους στο LogicFU



2 κύκλους αργότερα βλέπουμε το αποτέλεσμα της πράξης NOT και την μεταφορά του μέσω του CDB Στον RF καθώς και την εγγραφή του στο register 1 των V που είναι και η διεύθυνση που επιλέξαμε να αποθηκεύσουμε την εντολή





Με την εγγραφή λοιπόν του αποτελέσματος της πρώτης πράξης τελειώνω επιτυχώς η πρώτη εντολή

Η δεύτερη εντολή που εκτελέστηκε ήταν ίδια με την πρώτη απλά είχε καταχωρητή προορισμού τον καταχωρητή 2 και η πράξη έγινε τελείως αντίστοιχα με την πρώτη απλά με ένα κοινό διάφορα.

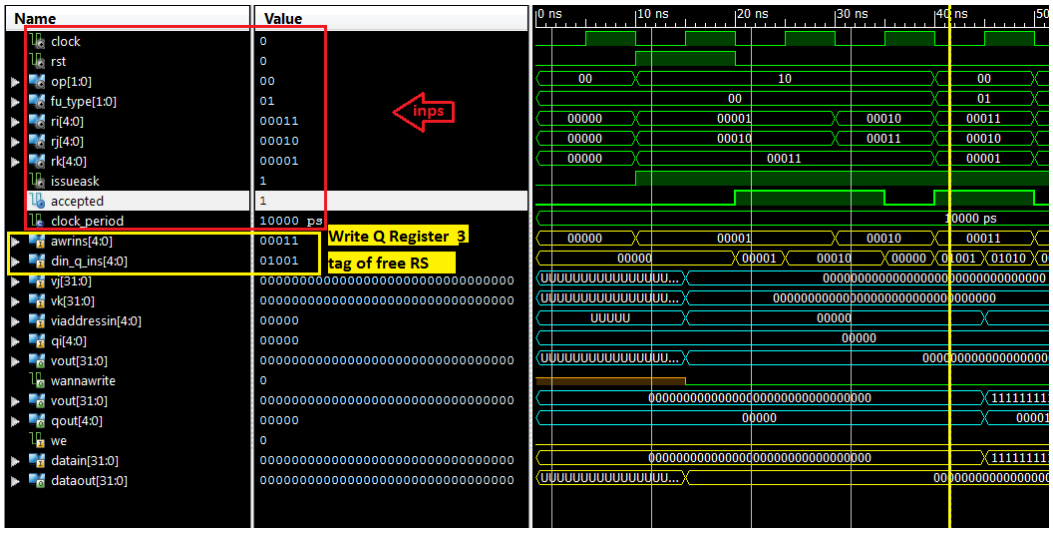
```
        wait for Clock_period*1;

Rst <='0';
        op <="10";
        FU_type <="00";
        Ri <="00010";
        Rj <="00011";
        Rk <="00011";
        Issueask<='1';
        wait for Clock_period*1;
```

Η τρίτη εντολή ήταν μια αριθμητική εντολή που έκανε πρόσθεση των 2 προηγούμενων αποτελεσμάτων και αποθηκεύει το αποτέλεσμα στον καταχωρητή 3

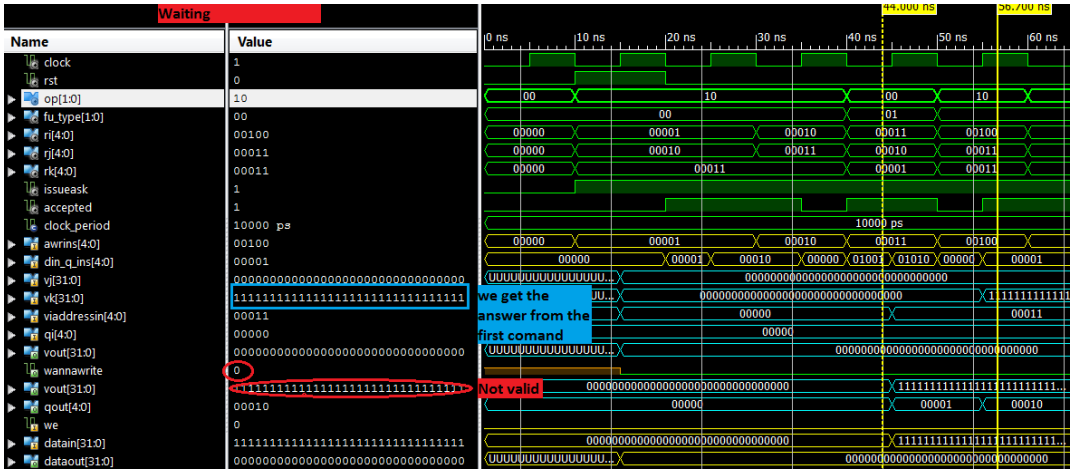
```
Rst <='0';
        op <="00";
        FU_type <="01";
        Ri <="00011";
        Rj <="00010";
        Rk <="00001";
        Issueask<='1';
        wait for Clock_period*1
```

Αρχικά γράφεται η τιμή 01001 στο καταχωρητη 3 του παιδιού Q

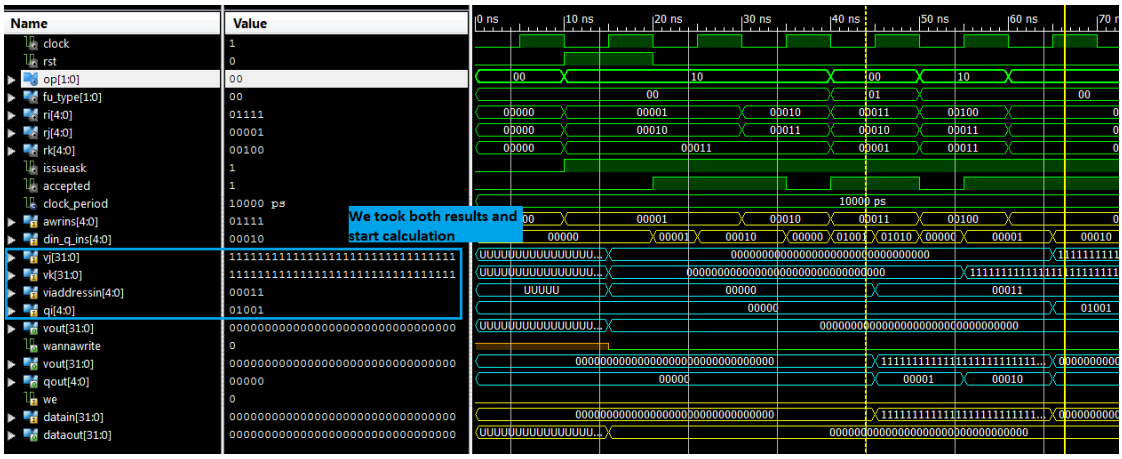


Βλέπουμε ότι δεν περνάει καμία τιμή στο παιδιά Vj και Vk του RS διότι δεν έχουν βγει ακόμα τα αποτελέσματα από τις προηγούμενες εντολές.Το αποτέλεσμα της FU δεν είναι σωστό γι'αυτό και δεν ζήτησε πρόσβαση στο CDB εφόσον έχουμε wannawrite=0

Στον επόμενο κύκλο βγαίνει το πρώτο αποτέλεσμα της πράξης που περιμένουμε

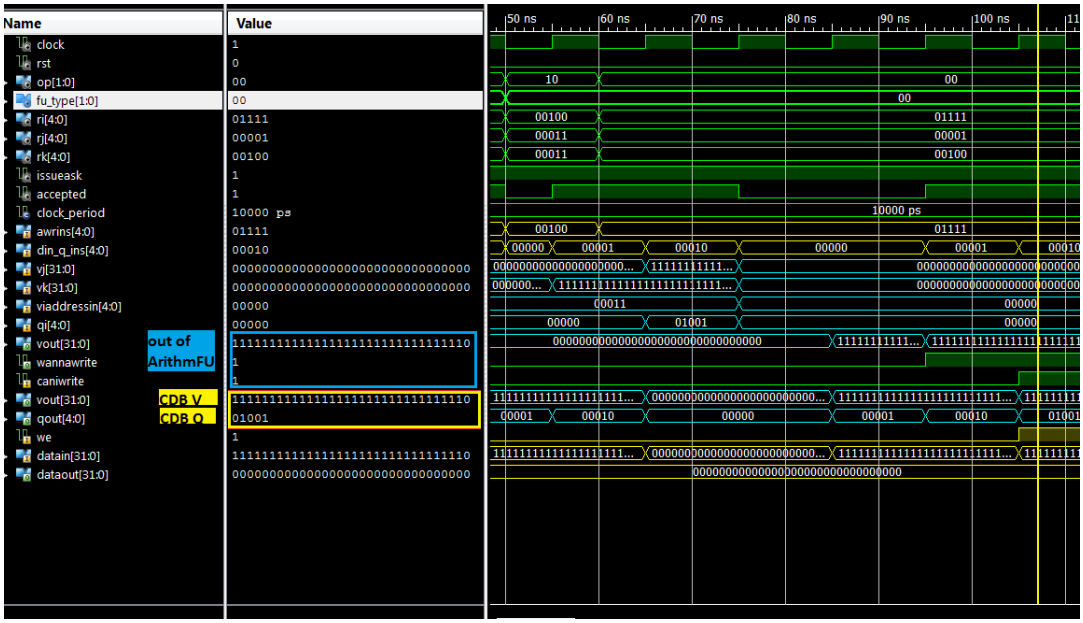


Στον επόμενο κύκλο βγαίνει το επόμενο αποτέλεσμα της πράξης που περιμένουμε

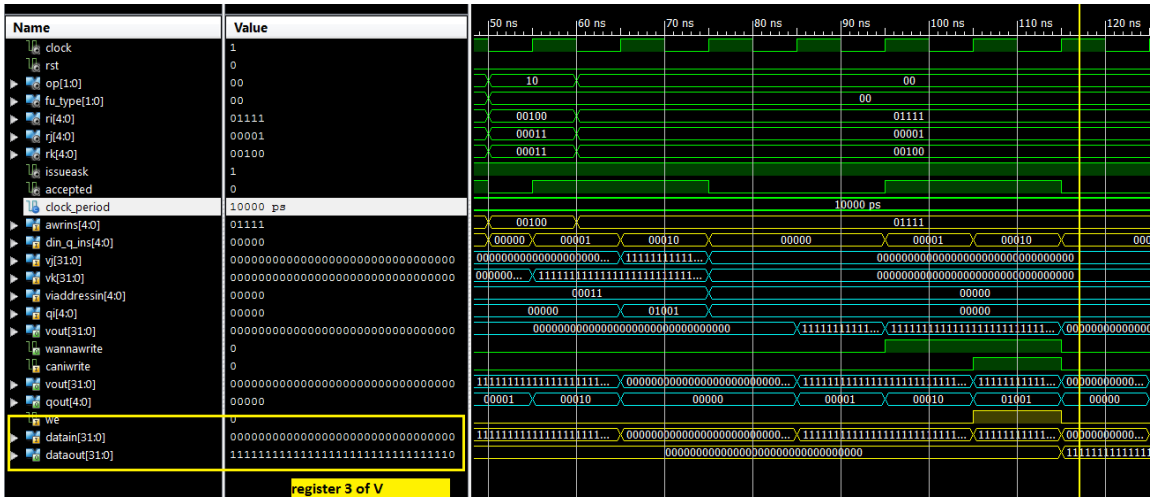


Μετά από 3 κύκλους έχουμε το αποτέλεσμα της πράξης αλλά δεν μπορούμε να το μεταδώσουμε στο CDB διότι υπάρχει σύγκρουση με άλλη εντολή και η απάντηση canIwrire=0 πότε απλά περιμένουμε

Έρχεται η σειρά αυτής της εντολής να γράφει στο CDB και στη συνέχεια στο RF



Γράφεται η τιμή στον καταχωρητή Q 3 στις και τελειώνει έτσι και αυτή η εντολή



Οι 3 πρωτες εντολές βλέπουμε ότι εκτελούνται σωστά. Το simulation περιέχει και άλλες εντολές οι οποίες εκτελούνται σωστά αλλά δεν εξηγούνται αναλυτικά όπως οι παραπάνω . Τα παραπάνω simulation βλέπουν συγκεκριμένα σήματα επιλεγμένα έτσι ώστε να μπορεί να γίνει σαφή επεξήγηση της κάθε εντολής. Ακολουθεί το ολοκληρωμένο simulation που έχει όλα τα σήματα και την πλήρη προσομοίωση .