Αρχιτεκτονική Υπολογιστών

HPY 415

Αναφορά Εργαστηρίου 1β : Σχεδίαση και Υλοποίηση διεπαφών/οντοτήτων των λειτουργικών μονάδων για τον αλγόριθμο Tomasulo

Ομάδα Εργασίας
Μπελλώνιας Παναγιώτης - 2014030058
Σπυριδάκης Χρήστος - 2014030022

Όλες οι εικόνες που είναι ενσωματωμένες στην αναφορά υπάρχουν στον φάκελο /doc/sim και /doc/schematic

Στα σχεδιαγράμματα που ακολουθούν ισχύει ο παρακάτω χρωματικός κώδικας για τα σήματα:

Κόκκινο - Ελέγχου
Μπλε - Δεδομένα
Πορτοκαλί - Βοηθητικά / Πληροφορίας
Πράσινο - Tag

Στα testbench κάθε κάθετο marker υποδηλώνει και ένα νέο κύκλο ρολογιού

1. ISSUE

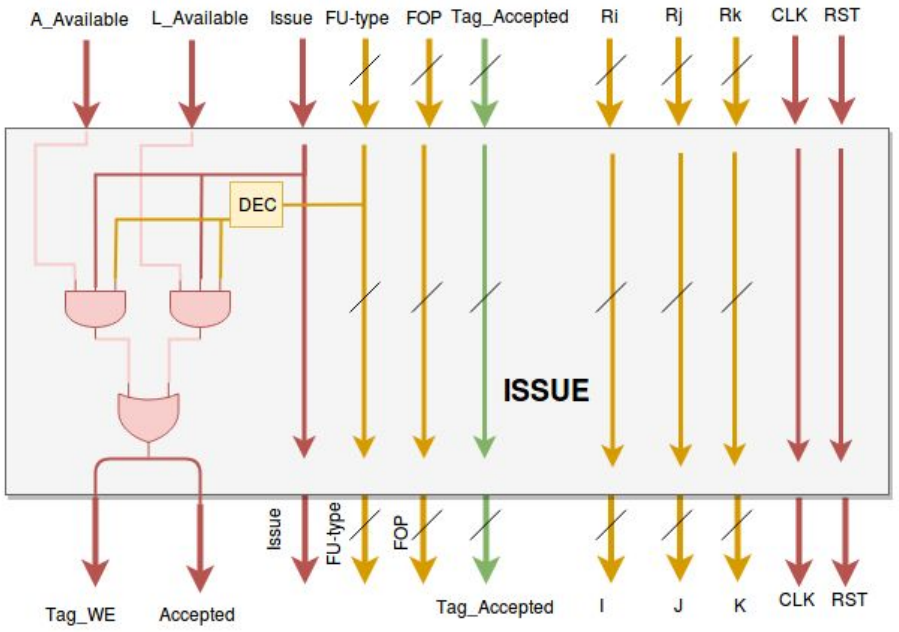
ΜΟΝΟ ΤΑ ΕΠΙΠΛΕΟΝ ΣΗΜΑΤΑ

Σήμα	Μέγεθος (bits)	Είσοδος / Έξοδος	Περιγραφή
A_Available	1	In	Διαθέσιμο Αριθμητικό RS
L_Available	1	In	Διαθέσιμο Λογικό RS
Tag_Accepted	5	out	Το RS που δεχτηκε την εντολή
Tag_WE	1	out	Ενεργοποίηση εγγραφής στον Ri

Πίνακας 1.1 - Διεπαφή ISSUE.

Λειτουργικότητα:

Σχετικά με το ISSUE ακολουθήθηκε η διεπαφή που υπάρχει στην εκφώνηση της άσκησης εμπλουτισμένη ώστε να κάνει κάποιους επιπρόσθετους ελέγχους. Αυτό σημαίνει ότι ο παραπάνω πίνακας περιέχει ΜΟΝΟ τα επιπρόσθετα αυτά σήματα. Ο λόγος που προστέθηκαν είναι για να γνωρίζουμε άμεσα αν θα γίνει έκδοση της εντολής γνωρίζοντας τα σήματα X_Available καθώς και το είδος της (το γνωρίζουμε ήδη λόγω του FU-type). Το άλλο σημαντικό σήμα που προστέθηκε είναι το Tag_Accepted που περιέχει το Tag που έκανε τελικά αποδοχή της εντολής.



Σχηματικό Διάγραμμα 1.1 - ISSUE

Implementation:

Για την υλοποίηση του Issue Unit το μόνο που χρειάστηκε να κάνουμε είναι να δημιουργήσουμε ένα νέο module στο οποίο η διασύνδεση των I/O ήταν σύμφωνα με το παραπάνω σχηματικό. Το πεδίο dec αυτό που κάνει είναι απλά να δίνει ένα ενεργό σήμα στην αντίστοιχη μονάδα ανάλογα με την κωδικοποίηση.

Testing:

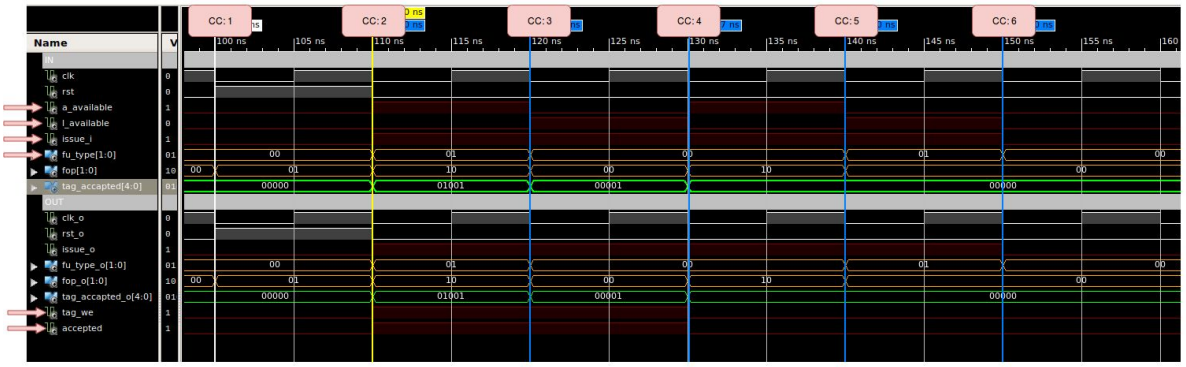
Για να επαληθεύσουμε ότι λειτουργούσε σύμφωνα με τις προδιαγραφές δημιουργήσαμε ένα testbench στο οποίο δώσαμε τις παρακάτω εισόδους και περιμέναμε να πάρουμε τις παρακάτω εξόδους.

Inputs

CC	Pseudocode
1	Reset
2	Arithmetic Instruction ISSUE + Arithmetic RS Available
3	Logical Instruction ISSUE + Logical RS Available
4	Logical Instruction ISSUE + Arithmetic RS Available
5	Arithmetic Instruction ISSUE + Logical RS Available

Expected Outputs

CC	Results
1	Reset
2	Accept Instruction
3	Accept Instruction
4	Instruction Aborted
5	Instruction Aborted



Simulation *1.1 - ISSUE*

Κατά την εκτέλεση του simulation, αυτό που παρατηρούμε είναι ότι το σύστημα λειτουργεί σύμφωνα με τις προδιαγραφές καθώς δέχετε όταν έχει διαθέσιμα slot στον RS (ανάλογα με το είδος της εντολής, π.χ. στους κύκλους 2 και 3 αποδέχεται τις εντολές) ενώ τις απορρίπτει όταν δεν έχει διαθέσιμα slot (π.χ. στους κύκλους 4 και 5). Τα υπόλοιπα είναι απλά βραχυκύκλωμα εισόδων/εξόδων ώστε να μεταφέρει στα σωστά units την πληροφορία.

2. Reservation Stations (RS)

Σήμα	Μέγεθος (bits)	Είσοδος / Έξοδος	Περιγραφή
ISSUE	1	In	Εκδοση Εντολής
FU-type	2	In	Τύπο εντολής
FOP	2	In	Πράξη πραγματοποίησης
Vj	32	In	Δεδομένα Καταχωρητή Rj
Qj	5	In	Tag Καταχωρητή Rj
Vk	32	In	Δεδομένα Καταχωρητή Rk
Qk	5	In	Tag Καταχωρητή Rk
CDB_V	32	In	Δεδομένα CDB
CDB_Q	5	In	Tag CDB
Tag_accepted	5	Out	Tag του RS που δέχτηκε την εντολή για ISSUE
A_Tag	5	Out	Tag Αριθμητικού RS που είναι έτοιμο για εκτέλεση
A_Available	1	Out	Διαθέσιμο Αριθμητικό Rs
A_Ready	1	Out	‘Ετοιμο για εκτελεση Αριθμητικό RS
A_Op	2	Out	Αριθμητική πράξη
A_Vj	32	Out	Δεδομένα Καταχωρητή Rj
A_Vk	32	Out	Δεδομένα Καταχωρητή Rk
A_Accepted	1	In	Αποδοχή εκτέλεσης Αριθμητικής πράξης
L_Tag	5	Out	Tag Λογικού RS που είναι έτοιμο για εκτέλεση
L_Available	1	Out	Διαθέσιμο Λογικό Rs

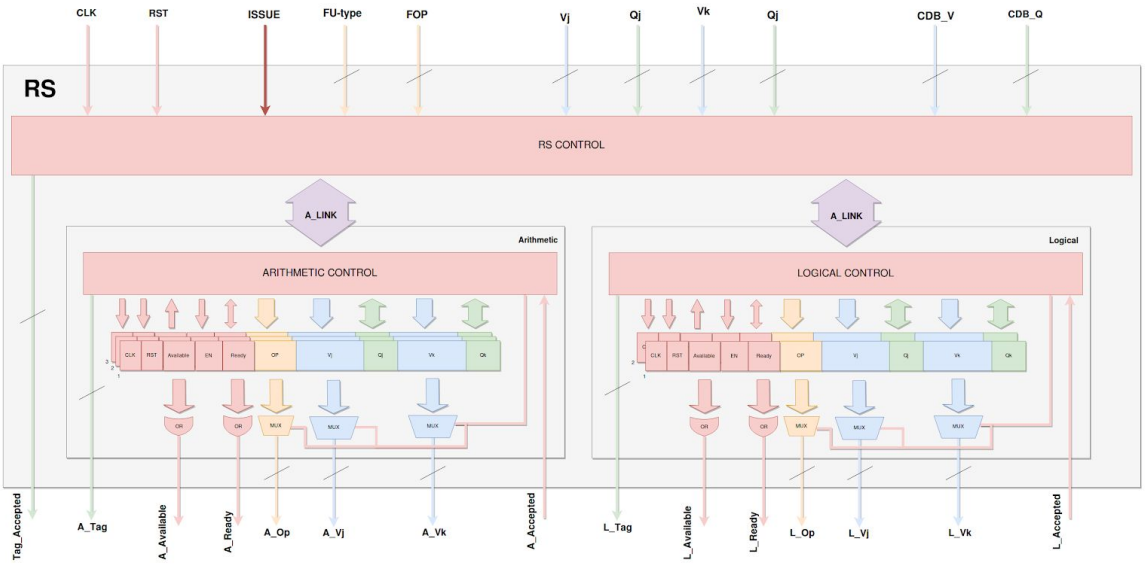
L_Ready	1	Out	‘Ετοιμο για εκτέλεση Λογικό RS
L_Op	2	Out	Λογική πράξη
L_Vj	32	Out	Δεδομένα Καταχωρητή Rj
L_Vk	32	Out	Δεδομένα Καταχωρητή Rk
L_Accepted	1	In	Αποδοχή εκτέλεσης Λογικής πράξης
CLK	1	In	Ρολόι
RST	1	In	Επαναφορά

Πίνακας 2.1 - Διεπαφή Reservation Stations.

Λειτουργικότητα:

Χρησιμοποιούμε τα σήματα A_Available και L_Available ώστε να γνωρίζουμε σε κάθε χρονική στιγμή αν μπορεί να εκδοθεί η επόμενη εντολή ανάλογα με το είδος της και τα διαθέσιμα RS που εχουμε. Στην περίπτωση έκδοσης εντολής με την βοήθεια των σημάτων εισόδου ISSUE, FU-type, FOP γνωρίζουμε ότι πραγματοποιείται έκδοση εντολής καθώς και το αν χρειάζεται να δρομολογηθούν τα σήματα Vj, Vk, Qj, Qk σε αριθμητικό ή σε λογικό RS. Χρησιμοποιούμε στην συνέχεια το σήμα CDB_Q ώστε να επιβεβαιώσουμε ότι έχουμε δεδομένα προς εγγραφή από το CDB και σε περίπτωση που υπάρχουν καταγράφουμε το CDB_V στο εκάστοτε RS ενημερώνοντας ταυτόχρονα και το πεδίο tag του.

Όταν είναι έτοιμο προς εκτέλεση ένα RS τότε ενεργοποιείται το σήμα X_Ready (όπου X αναφερόμαστε είτε σε A -Arithmetic- είτε σε L -Logical- καθώς ακολουθείται η ίδια διαδικασία) το οποίο είναι είσοδος προς την FU για να ενημερώσει ότι πρόκειται να στείλει δεδομένα. Μαζί περνάμε τα σήματα X_Op, X_Vj, X_Vk και X_Tag ώστε να έχουμε όλες τις πληροφορίες για το ποια πράξη με ποιες εισόδους πρόκειται να πραγματοποιήσει ο FU και εκ μέρους ποιου RS. Τέλος λαμβάνουμε από το FU ως feedback το σήμα X_Accepted ώστε να γνωρίζουμε αν δέχτηκε να πραγματοποιήσει την πράξη.



Σχηματικό Διάγραμμα 2.1 - Reservation Stations

Implementation:

Για να υλοποιήσουμε την παραπάνω λειτουργική μονάδα δημιουργήσαμε αρχικά το module “Reg_V_Q” (το ίδιο χρησιμοποιούμε επίσης στην RF όσο και στο FU) στο οποίο μπορούμε να αποθηκεύσουμε ταυτόχρονα Data καθώς και το Tag το οποίο τα συνοδεύει. Στην συνέχεια, δημιουργήσαμε ένα μεμονωμένο Reservation station slot το ονομα του οποίου είναι “Reg_RS”. Αυτό έχει όλους τους απαραίτητους καταχωρητές και λογική που χρειάζεται για να γίνεται issue μιας εντολής σε αυτό, να ενημερώνει τις τιμές του από το CDB και να ενεργεί αυτόνομα έχοντας από το περιβάλλον σαν εισόδους σήματα ελέγχου. Με αποτέλεσμα να είναι μία ολοκληρωμένη μονάδα, πλέον το πρόβλημα μας ήταν να επιλέγουμε σε ποιο από τα 5 (που θα είχαμε διαθέσιμα) να γίνεται Issue ανάλογα με την κατάσταση τους, όπως επίσης ποιο θα στέλνουμε κάθε χρονική στιγμή στην FU ή ακόμα και ποιού τελικά την κατάσταση θα ενημερώνουμε σε περίπτωση που το αποδεχτεί η FU. Δημιουργήσαμε, λοιπόν, τα modules “RS”, “A_RS” και “L_RS” τα οποία περιλαμβάνουν τα control units αυτών των Slots. Ακολουθήσαμε την ιεραρχία που είχαμε σχεδιάσει στο 1α Milestone σύμφωνα με το παραπάνω διάγραμμα.

Testing:

Αρχικά δοκιμάσαμε την λειτουργία του μεμονωμένου slot ώστε να είμαστε σίγουροι ότι ενημερώνει τις τιμές των καταχωρητών του μόνο τις στιγμές όπου χρειάζεται. Ενώ στην συνέχεια δοκιμάσαμε όλο το RS unit ότι ελέγχει ορθά κάθε ένα από αυτά τα slots.

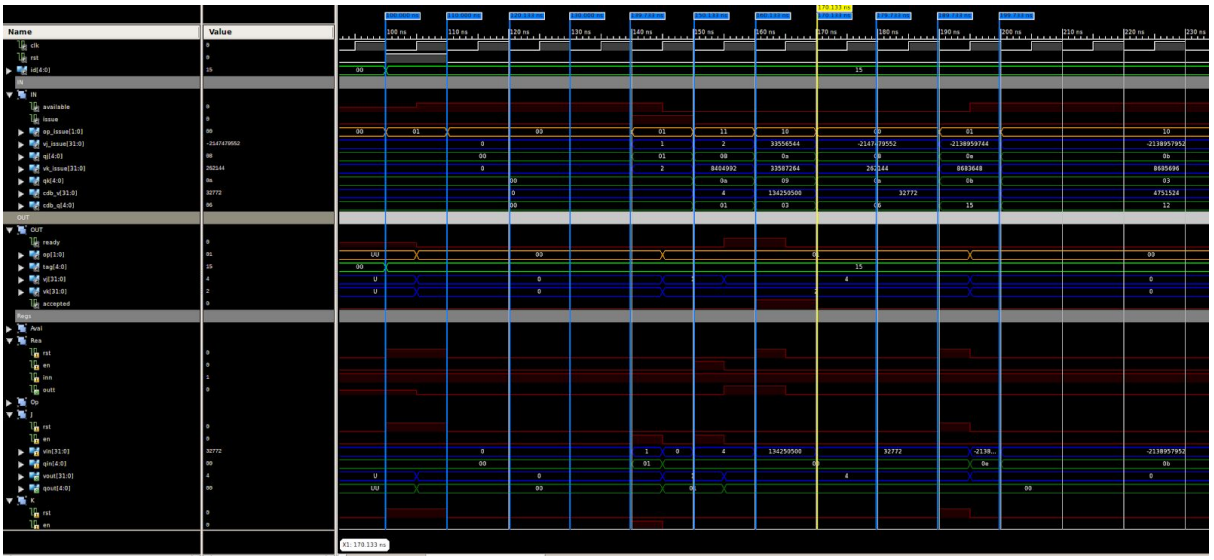
Reg_RS

Inputs

CC	Pseudocode
1	Reset
2	Nop for 3 CCs
5	Issue instruction with J NON valid and K valid
6	CDB_Q = J_Q (πλεόν είναι Ready)
7	RS accepted from FU
8	Nop for 2 CCs
10	Operation completed (RS_ID=CDB_Q)

Expected Outputs

CC	Results
1	Reset
5	Accept Instruction (WEN για τα J, K, Op και το Available=0)
6	Update J Value (WEN για το J και πλέον Ready η εντολή)
7	Αφού το FU αποδέχτηκε την εντολή το Ready=0 (για να μην ζητάει δικαίωμα αποστολής προς το FU) αλλά συνεχίζει το slot να είναι κατειλημμένο μέχρι την διεκπεραίωση της
10	Η εντολή ολοκληρώθηκε το slot αποδεσμεύεται και είναι πάλι πλέον Available



Simulation *2.1 - Reg_RS*

Όπως φαίνεται στο *Simulation 2.1* μπορούμε να διαπιστώσουμε ότι το module γράφει στους “καταχωρητές” Op, J και K τον 5ο CC (φαίνεται το En των group J και K αλλά το ίδιο ισχύει και για το Op), γράφει μόνο στον J στον 6 CC και γίνεται Ready το slot (φαίνεται το En του group J όπως επισης και του Ready). Έπειτα, στον 7 κύκλο που γίνεται accept το Ready δέχεται reset οπότε πλέον δεν είναι ενεργό και τέλος στον 10 κυκλο βλέπουμε πλέον ότι είναι πάλι διαθέσιμο το RS. Καθ’ όλη την διάρκεια, στις κυματομορφές υπάρχουν και οι τιμές των registers για να φαίνεται ότι οι ενέργειες που χρειάζεται να γίνουν έχουν όντως διεκπεραιωθεί.

Αφού δημιουργήσαμε και δοκιμάσαμε ένα μεμονωμένο slot, γνωρίζοντας ότι λειτουργεί αυτόνομα, συνεχίσαμε με τη δοκιμή ολόκληρης της λειτουργίας του RS. Το σημαντικό κομμάτι πλέον είναι σε ποιο από τα 5 slots ανάλογα με το είδος της εντολής γίνεται ISSUE (με βάση και την διαθεσιμότητα του), και σε ποιο από αυτά στέλνονται το σήμα -ελέγχου-αποδοχής από την FU. Συνεπώς, αυτά είναι τα μόνα σήματα τα οποία θα μας απασχολήσουν (τουλάχιστον στο κομμάτι της αναφοράς, καθώς κατά το πειραματικό μας μέρος επαληθεύσαμε ότι κάθε Slot έχει τις πληροφορίες που σχεδιαστικά θα έπρεπε να είχε). Παρακάτω παρατίθενται τα αποτελέσματα των δοκιμών.

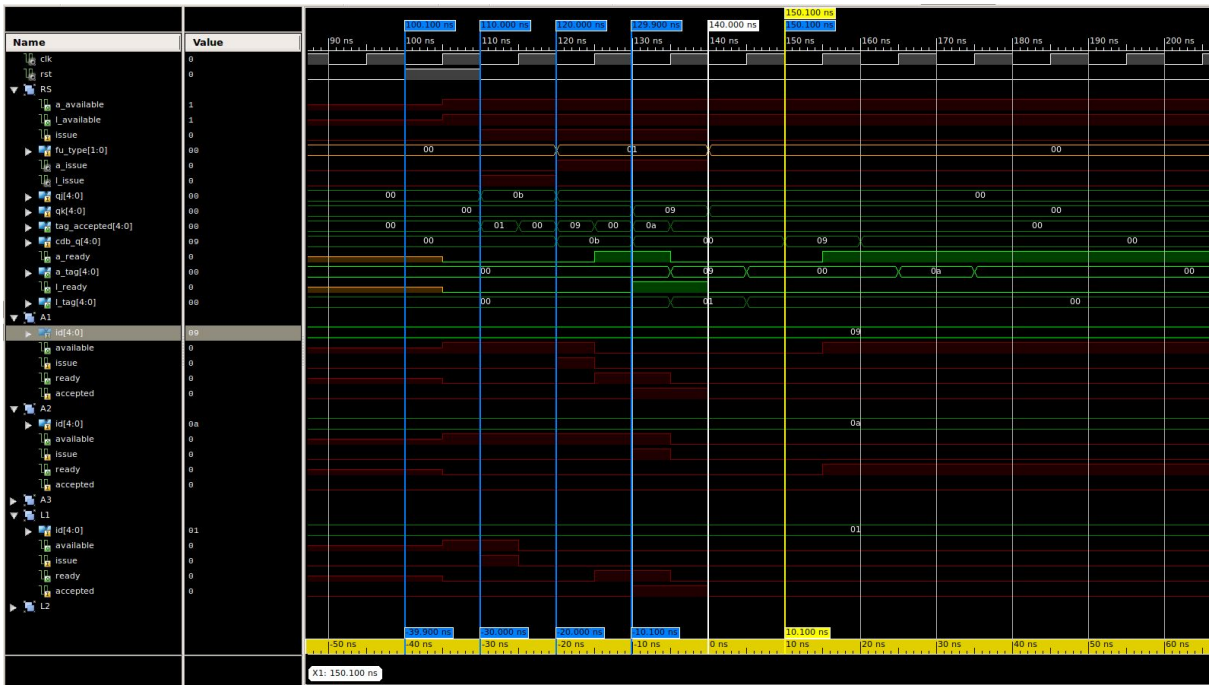
RS

Inputs

CC	Pseudocode
1	Reset
2	Issue Logical NON Ready Instruction with J_Q=01011
3	Issue Arithmetic Ready Instruction + CDB_Q=01011
4	Issue Arithmetic NON Ready Instruction with K_Q=01001 + A1_RS accepted + L1_RS accepted
5	Nop
6	A1 completed (CDB_Q=01001)
7	Nop

Expected Outputs

CC	Results
1	Reset
2	Accept Logical Instruction (WEN για τα J, K, Op και το Available=0 για το L1_RS)
3	Accept Arithmetic Instruction (WEN για τα J, K, Op και το Available=0 για το A1_RS) επίσης στον ίδιο κύκλο το A1_Ready=1 αλλά επειδή το CDB έφερε δεδομένα και για το L1 τότε και το L1_Ready=1
4	Accept Arithmetic Instruction (WEN για τα J, K, Op και το Available=0 για το A2_RS) επίσης το A1_Ready=0 αφού το αποδέχτηκε το FU όπως επίσης και το L1_Ready=0 αφού και αυτό το αποδέχτηκε το αντίστοιχο FU του
6	Απελευθέρωση του A1_RS καθώς ολοκληρώθηκε ο υπολογισμός ενώ πλέον το A2_RS είναι Ready αφού περίμενε δεδομένα από το A1_RS που μόλις ολοκληρώθηκε



Simulation *2.2 - RS*

Στο παραπάνω simulation μπορούμε να δούμε ότι τα σήματα ελέγχου για το RS λειτουργούν όπως ήταν αναμενόμενο. Ένα κομμάτι το οποίο δεν αναφέρεται στα expected outputs είναι το tag που αποδέχτηκε την εντολή (όταν ήταν δυνατόν αυτό) η οποία πρόκειται να μπει στο σύστημα καθώς και το tag το οποίο πάει στην FU όταν είναι 1 ή περισσότερα Ready. Σχετικά με το Tag αποδοχής θα είναι το πρώτο διαθέσιμο του κάθε slot και ανάλογα με την φύση της εντολής επιλέγουμε το κατάλληλο. Ενώ για το ποιο Arithmetic/Logical θα προωθηθεί στην FU είναι το πρώτο Ready που έχει προτεραιότητα σε Round-robin λογική.

3. Functional Units (FU)

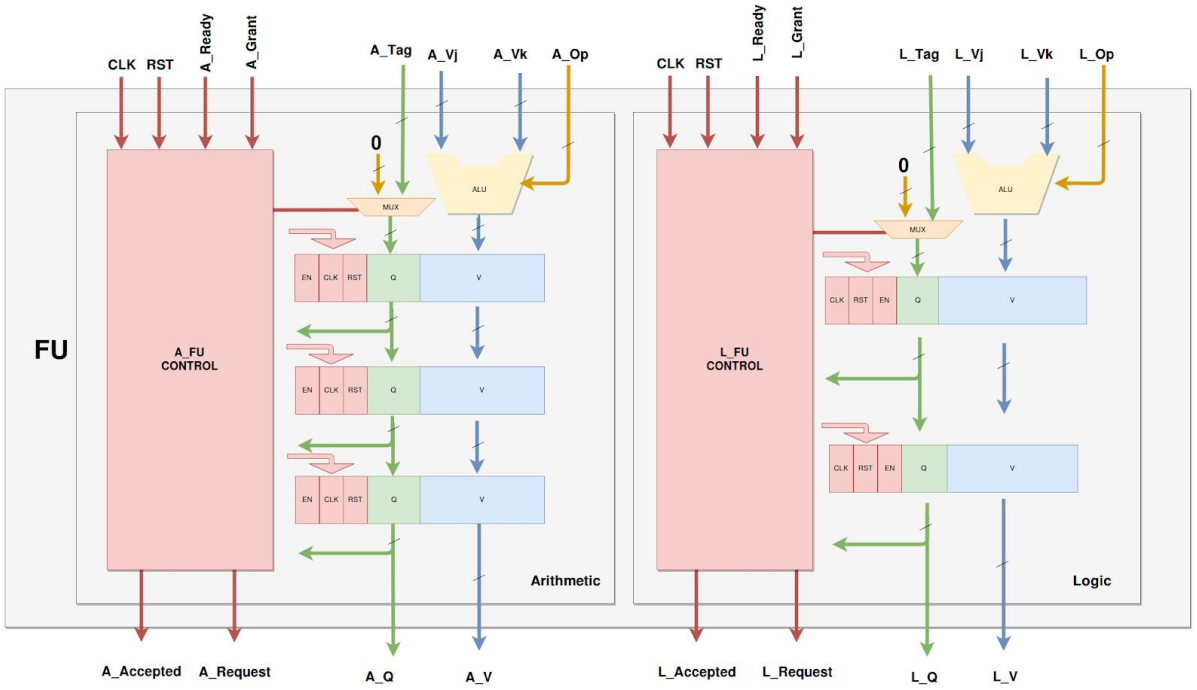
Σήμα	Μέγεθος (bits)	Είσοδος / Έξοδος	Περιγραφή
A_Ready	1	In	Υπαρχει Αριθμητική πράξη για εκτέλεση
A_Grant	1	In	Δικαίωμα εγγραφής της αριθμητικής FU στον CDB
A_Tag	5	In	Το Αριθμητικό RS που ζητά να πραγματοποιηθεί πράξη
A_Vj	32	In	Δεδομένα καταχωρητή Rj
A_Vk	32	In	Δεδομένα καταχωρητή Rk
A_Op	2	In	Αριθμητική πράξη πραγματοποίησης
L_Ready	1	In	Υπαρχει Λογικη πράξη για εκτέλεση
L_Grant	1	In	Δικαίωμα εγγραφής της Λογικης FU στον CDB
L_Tag	5	In	Το Λογικη RS που ζητά να πραγματοποιηθεί πράξη
L_Vj	32	In	Δεδομένα καταχωρητή Rj
L_Vk	32	In	Δεδομένα καταχωρητή Rk
L_Op	2	In	Λογικη πράξη πραγματοποίησης
A_Accepted	1	Out	Αποδοχή της Αριθμητικής πράξης
A_Request	1	Out	Αίτημα κοινοποίησης αποτελέσματος στο CDB
A_Q	5	Out	Το αριθμητικό RS που ζήτησε την πράξη

A_V	32	Out	Αποτέλεσμα Αριθμητικής πράξης
L_Accepted	1	Out	Αποδοχή της Λογικής πράξης
L_Request	1	Out	Αίτημα κοινοποίησης αποτελέσματος στο CDB
L_Q	5	Out	Το Λογικό RS που ζήτησε την πράξη
L_V	32	Out	Αποτέλεσμα Λογικής πράξης
CLK	1	In	Ρολόι
RST	1	In	Επαναφορά

Πίνακας 3.1 - Διεπαφή Funcional Units.

Λειτουργικότητα:

Όμοια με το RS θα χρησιμοποιηθεί και εδώ η παραδοχή ότι όπου X αναφερόμαστε είτε σε A -Arithmetic- είτε σε L -Logical- καθώς ακολουθείται η ίδια διαδικασία. Οταν μας έρθει σήμα X_Ready σημαίνει ότι υπάρχει έτοιμο RS για πραγματοποίηση πράξης. Αν μπορούμε να την λάβουμε στέλνουμε προς το RS το σήμα X_Accepted για να το ενημερώσουμε. Σε περίπτωση όπου είμαστε διαθέσιμοι να εκτελέσουμε την πράξη χρησιμοποιούμε τα πεδία X_Tag, X_Vj, X_Vk, X_Op και έπειτα πραγματοποιούμε την καθυστέρηση της πράξης χρησιμοποιώντας καταχωρητές. Μόλις φτάσουμε ένα κύκλο πριν την πραγματοποίησης στέλνουμε αίτημα κοινοποίησης στον CDB μέσω του σήματος X_Request, σε περίπτωση όπου στον επόμενο κύκλο το X_Grant είναι ενεργό, σημαίνει ότι έχουμε το δικαίωμα να χρησιμοποιήσουμε εμείς τον CDB οπότε τότε κοινοποιούμε τα αποτελέσματα της πράξης τα οποία είναι το X_V και X_Q.



Σχηματικό Διάγραμμα 3.1 - Functional Units

Implementation:

Για την δημιουργία του FU αξιοποιήσαμε τους “register” Reg_V_Q που αναφέραμε παραπάνω. Επίσης, δημιουργήσαμε μία κοινή ALU όπου παίρνει ανεξάρτητα εισόδους και παράγει ταυτόχρονα αποτελέσματα τόσο για Arithmetic όσο και για Logical. Επιπλέον, στην είσοδο του Q του πρώτου επιπέδου και στο Arithmetic και στο Logical προσθέσαμε έναν πολυπλέκτη ο οποίος είτε θα δίνει μηδενική τιμή είτε αν έχουμε είσοδο από το RS θα δίνει το Tag της εισόδου, αυτό χρειάστηκε για να κάνουμε την λειτουργία του Pipeline μεταξύ των καταχωρητών. Τέλος, το μόνο που έπρεπε ήταν να υλοποιήσουμε το control αυτών των καταχωρητών ώστε να καταγράφουν στα πεδία τους μόνο όταν χρειάζεται.

Testing:

Για να δοκιμάσουμε την λειτουργικότητα της FU εισαγάγαμε ενέργειες για την καθεμία ξεχωριστά όπως επίσης και για την ταυτόχρονη, μεταξύ των 2 μονάδων, λειτουργία τους.

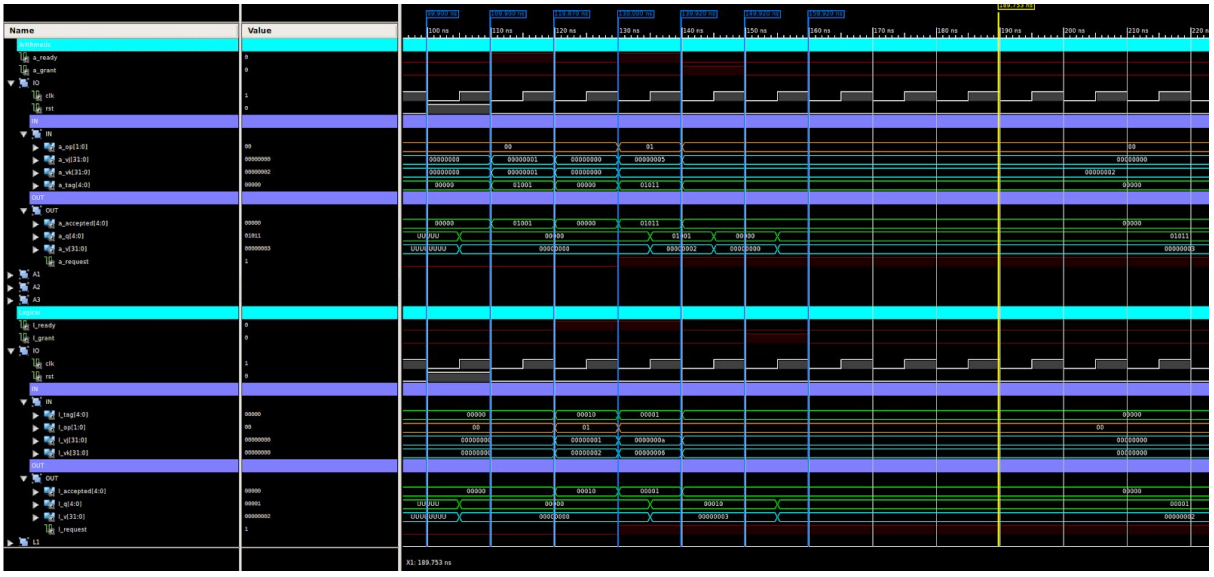
FU

Inputs

CC	Pseudocode
1	Reset
2	A1_RS request (1 + 1)
3	L2_RS request (1 OR 2)
4	A3_RS request (5 - 2) + L1_RS request (10 AND 6)
5	Grand A
6	Grand L

Expected Outputs

CC	Results
1	Reset
2	Accept operation
3	Accept operation
4	Accept operation + Request A1 + Request L2
5	Broadcast A1_RS result + Request L1
6	Broadcast L2_RS result + Request A3
7...	A1_RS and L2_RS Wait for Grand



Simulation *3.1 - FU*

Από την παραπάνω κυματομορφή αυτό που μπορούμε να δούμε είναι ότι τα αποτελέσματα τα οποία μας εμφανίζει είναι σύμφωνα με αυτά τα οποία περιμέναμε να δούμε από τον παραπάνω πίνακα (τα σήματα που μας ενδιαφέρουν κυρίως είναι τα A_Request/L_Request, A_Ready/L_Ready, A_Grant/L_Grant καθώς και οι τιμές των εξόδων των μονάδων), κάτι που πρέπει να σημειωθεί είναι ότι μόλις γίνει αποδεκτή μία πράξη ειδοποιήται το RS μέσω του A_Accepted/L_Accepted ότι το RS με το ίδιο ID έγινε αποδεκτό.

Σημείωση: Στον παραπάνω πίνακα αναγράφονται πότε ξεκινάει να συμβαίνει ένα γεγονός. Παραδείγματος χάρη στον 5 CC εκτός από αυτά έχουμε και το Request του Request L2 για να χρησιμοποιήσει το CDB που δεν έχει γίνει ακόμα και πραγματοποιείται τελικά στον επόμενο κύκλο αλλά δεν το γράφουμε για λόγους απλοποίησης.

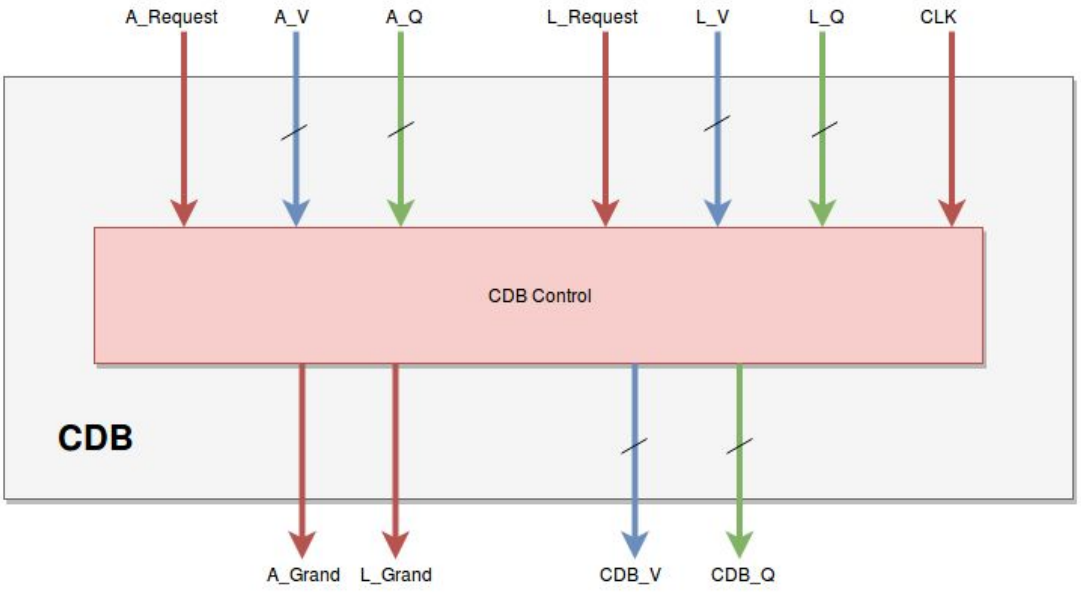
4. Common Data Bus (CDB)

Σήμα	Μέγεθος (bits)	Είσοδος / Έξοδος	Περιγραφή
A_Request	1	In	Έλεγχος Αίτησης για Αριθμητική Μονάδα
A_V	32	In	Τιμή Αριθμητικής Μονάδας
A_Q	5	In	Q Αριθμητικής Μονάδας
L_Request	1	In	Έλεγχος αίτησης για Λογική Μονάδα
L_V	32	In	Τιμή Λογικής Μονάδας
L_Q	5	In	Q Λογικής Μονάδας
A_Grand	1	Out	Αριθμητική Μονάδα στο CDB
L_Grand	1	Out	Λογική Μονάδα στο CDB
CDB_V	32	Out	Έξοδος CDB_V
CDB_Q	5	Out	Έξοδος CDB_Q
CLK	1	In	Ρολόι
RST	1	In	Επαναφορά

Πίνακας 4.1 - Διεπαφή Common Data Bus.

Λειτουργικότητα:

Το Common Data Bus είναι ο δίαυλος επικοινωνίας του συστήματος, τον οποίο χρησιμοποιούν τα Reservation Stations ώστε να ανανεώνουν και να περνούν τις τιμές τους ώστε να εγγράφονται στη διεπαφή Register File. Αποτελείται από μία είσοδο αίτησης για κάθε Functional Unit, Αριθμητικό και Λογικό αντίστοιχα, καθώς και είσοδοι για τη λήψη των Tags και των δεδομένων τους. Όταν τα παραπάνω Units έχουν διαθέσιμα αποτελέσματα, μπορούν να ενημερώσουν τις υπόλοιπες λειτουργικές μονάδες μέσω του CDB. Αν η αίτηση για ενημέρωση είναι θετική τότε τα αντίστοιχα σήματα εξόδου Granted παίρνουν την τιμή 1 ώστε να γνωρίζει το υπόλοιπο σύστημα ποιος θα κοινοποιήσει στον επόμενο κύκλο του ρολογιού. Το Common Data Bus ακολουθεί λογική round-robin με σκοπό να αποφασίσει ποια μονάδα θα κοινοποιήσει. Οι έξοδοι CDB_V/Q οδηγούνται στη Register File και στα διάφορα Reservation Stations.



Σχηματικό Διάγραμμα 4.1 - Common Data Bus

Implementation:

Το μόνο που χρειάστηκε να υλοποιηθεί ήταν η λογική Round-robin (προς το παρόν μόνο για το Arithmetic και Logical) με την προσοχή ότι στον ένα κύκλό γίνεται το Request και στον επόμενο θα δίνεται το Grant

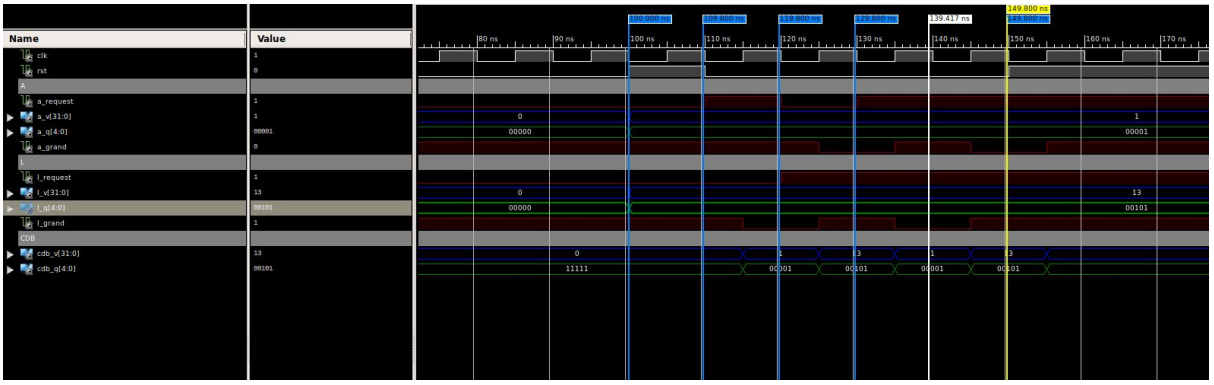
Testing:

Inputs

CC	Pseudocode
1	Reset
2	Arithmetic Request
3	Logical Request
4	Arithmetic Request + Logical Request
5	Arithmetic Request + Logical Request
6	Reset

Expected Outputs

CC	Results
1	Reset
3	Grant Arithmetic
4	Grant Logical
5	Grant Arithmetic
6	Reset



Simulation *4.1 - CDB*

Και σε αυτό το simulation το Unit λειτουργεί όπως περιμέναμε και περιγράφεται αναλυτικά στον παραπάνω πίνακα.

5. Register File (RF)

Σήμα	Μέγεθος (bits)	Είσοδος / Έξοδος	Περιγραφή
Ri	5	In	Καταχωρητής Εγγραφής
Rj	5	In	Καταχωρητής Ανάγνωσης
Rk	5	In	Καταχωρητής Ανάγνωσης
Tag_WE	1	In	Write - Enable για τον Ri καταχωρητή
Tag_Accepted	5	In	Το Tag εγγραφής στον Ri
CDB_V	32	In	Είσοδος CDB_V
CDB_Q	5	In	Είσοδος CDB_Q
Qj	5	Out	Tag Καταχωρητή Rj
Vj	32	Out	Δεδομένα Καταχωρητή Rj
Qk	5	Out	Tag Καταχωρητή Rk
Vk	32	Out	Δεδομένα Καταχωρητή Rk
CLK	1	In	Ρολόι
RST	1	In	Επαναφορά

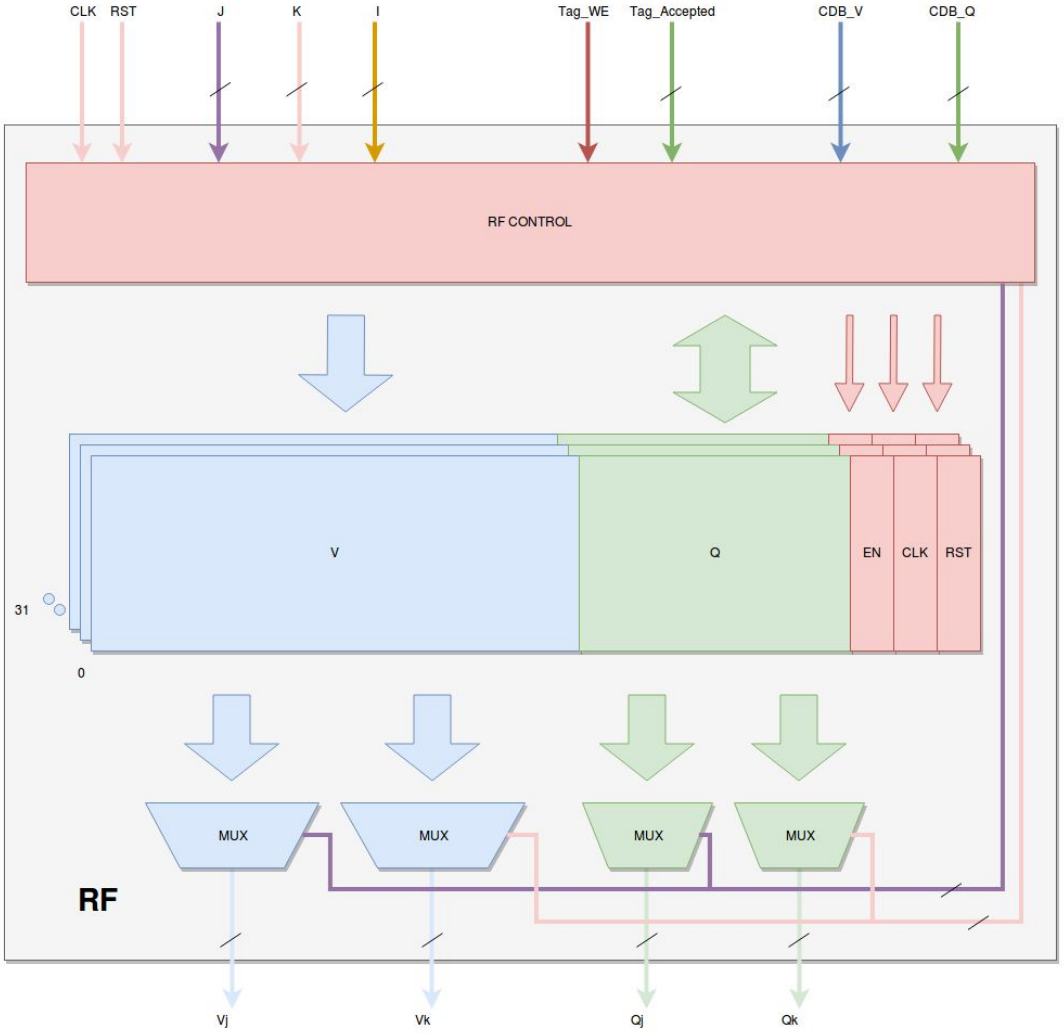
Πίνακας 5.1 - Διεπαφή Register File.

Λειτουργικότητα:

Η Register File είναι μία τυπική διεπαφή με το χαρακτηριστικό ότι εγγραφή σε αυτή μπορεί να γίνει μόνο μέσω του Common Data Bus (CDB), στην περίπτωση που το Tag του ταιριάζει με κάποιο από αυτά των καταχωρητών της. Αποτελείται από 64 καταχωρητές, 32 των 32 bits για τα δεδομένα και 32 των 5 bits για τα Tags. Η RF δέχεται σαν είσοδο τρεις καταχωρητές από το Issue και μέσω του σήματος Tag_WE γνωρίζει οποιαδήποτε χρονική στιγμή αν κάποια εντολή εκδίδεται. Στην περίπτωση που το παραπάνω σήμα είναι ενεργό, πραγματοποιείται εγγραφή στο Tag του καταχωρητή Ri, το Tag_Accepted που λαμβάνει από τη λειτουργική μονάδα που δέχεται την εντολή.

Επιπλέον, η RF δέχεται σαν είσοδο, της εξόδους CDB.V/Q του Common Data Bus, μέσω της πύλης εγγραφής της RF. Όταν υπάρξει έτοιμη τιμή του Q, τότε πραγματοποιείται έλεγχος

των Tags των καταχωρητών της και αν βρεθεί η συγκεκριμένη τιμή, το σήμα CDB.V εγγράφεται στον αντίστοιχο καταχωρητή. Κάθε φορά που πραγματοποιείται η παραπάνω διαδικασία το Tag του καταχωρητή παίρνει την τιμή 00000 ώστε να φαίνεται πως περιέχει έγκυρα δεδομένα. Στην αντίθετη περίπτωση, το CDB.V αγνοείται.



Σχηματικό Διάγραμμα 5.1 - Register File

Implementation:

Για να υλοποιήσουμε την λογική του RF χρησιμοποιήσαμε και σε αυτό το Unit τον “Reg_V_Q” που δημιουργήσαμε παραπάνω. Δημιουργήσαμε 32 τέτοιους καταχωρητές γνωρίζοντας ότι θα έπρεπε να παίρνουν πρακτικά τιμές δεδομένων μόνο από τον CDB ενώ Tag μόνο κατά το Issue. Ελέγχουμε, λοιπόν, σε κάθε κύκλο ρολογιού για κάθε register αν έχουμε ISSUE και η σειριακή τιμή του register είναι ίδια με το Ri ώστε να αποθηκεύουμε στον καταχωρητή το δεδομένο Tag ενώ σε αντίθετη περίπτωση βλέπουμε αν για τον ίδιο register το tag του είναι ίδιο με τα tag που μας παρέχει ο CDB. Ταυτόχρονα, βλέπουμε αν χρειάζεται να κάνουμε forward δεδομένα που μας κοινοποιεί ο CDB στην περίπτωση που κάποιος οι κάποιοι από τους register που διαβάζουμε κατά το issue παίρνει τιμή στην ίδιο κύκλο.

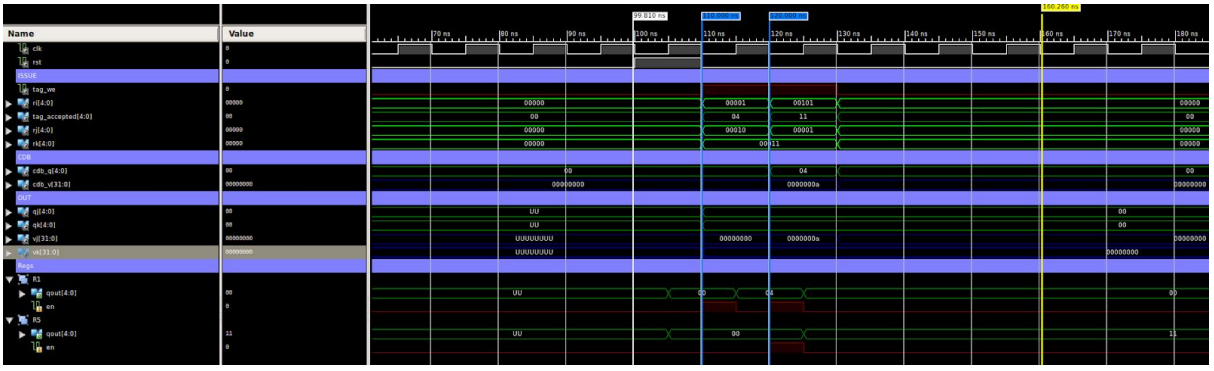
Testing:

Inputs

CC	Pseudocode
1	Reset
2	Issue with Ri=1, Rj=2, Rk=3, Tag=4
3	Issue with Ri=5, Rj=1, Rk=3, Tag=11 + CDB_Q=4, CDB_V=10

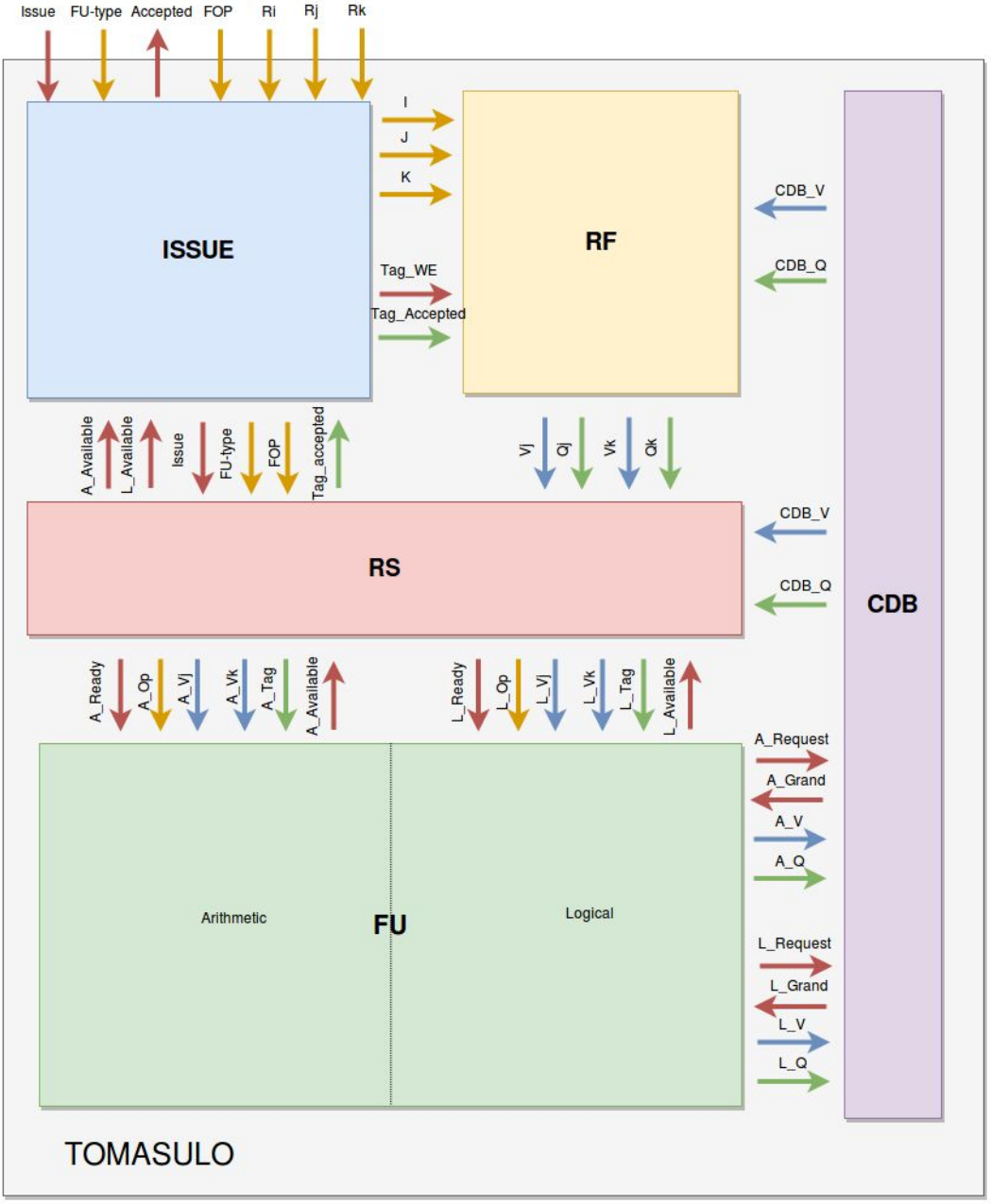
Expected Outputs

CC	Results
1	Reset
2	Read Rj and Rk + Write R1_Q=4
3	Write R5_Q=17 + Αφού το CDB_Q=Rj_Q τότε θα πρέπει να διαβάσουμε για το Rj την ανανεωμένη τιμή άρα το 10 ενώ για το Rk πρέπει να πάρουμε την τιμή που έχει ήδη + να γραφτεί στον R1 η τιμή του CDB



Simulation *5.1 - CDB*

Στην παραπάνω κυματομορφή μπορούμε πάλι να επαληθεύσουμε την λειτουργία της RF μονάδας. Συγκεκριμένα εμφανίζονται τα EN signals των R1 και R5 όπως επίσης και τα Tags που είναι αποθηκευμένα στο καθένα από αυτά οπότε μπορούμε να δούμε πότε αλλάζουν τιμές και αν η αλλαγή των τιμών είναι σύμφωνα με το πότε το περιμέναμε. Τέλος, μπορούμε να δούμε τις εξόδους της RF να είναι αυτές που περιγράφονται παραπάνω στον πίνακα με τα expected αποτελέσματα.



Σχηματικό διάγραμμα ολόκληρου του συστήματος