Αρχιτεκτονική Υπολογιστών

Αναφορά 3ου Εργαστηρίου

Ομάδα Εργασίας:
Βαϊλάκης Αποστόλης - 2014030174
Φωτάκης Τζανής - 2014030054

Reorder Buffer Block

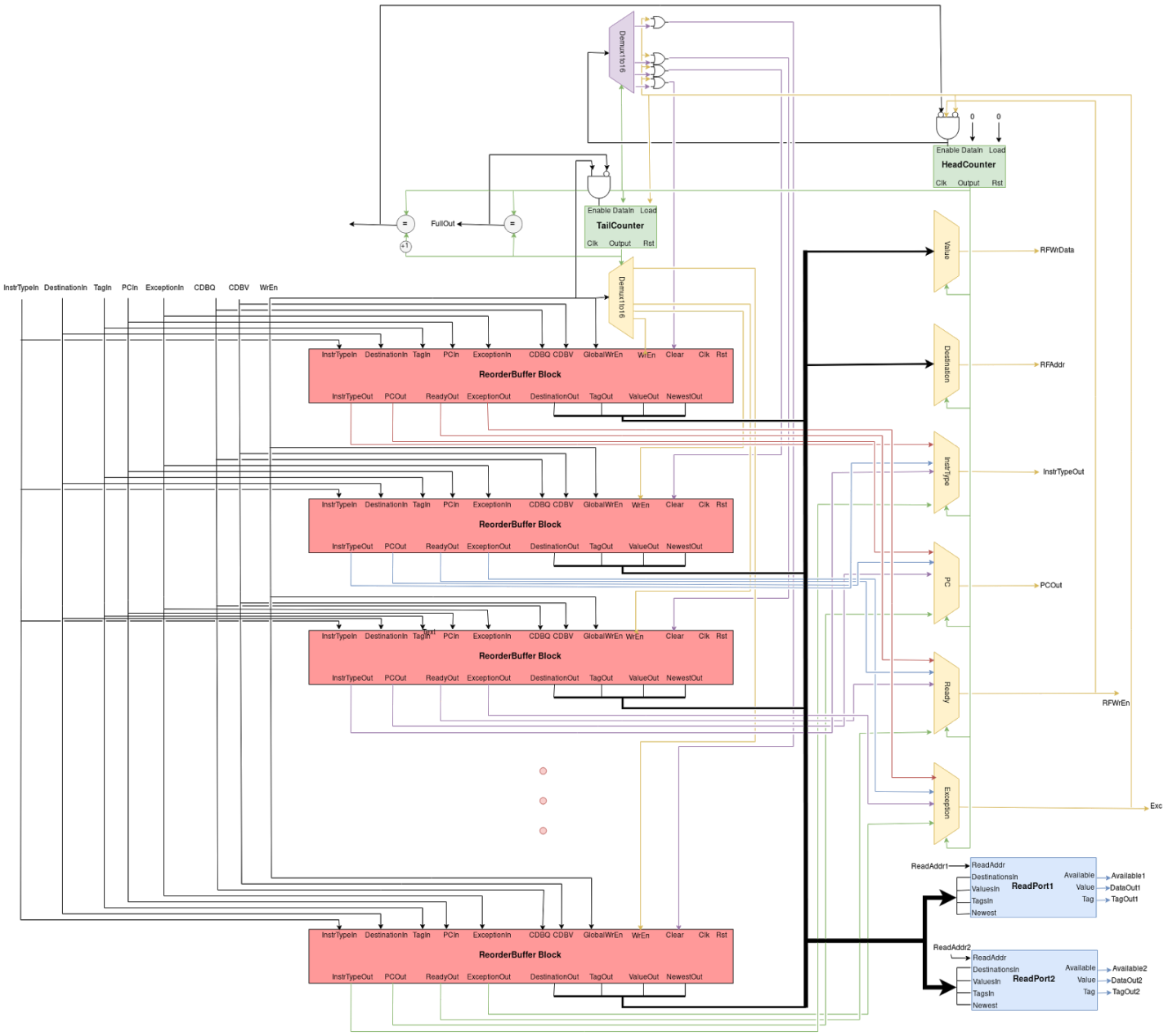
Για την δημιουργία του ReorderBuffer χρειάστηκαν ReorderBufferBlocks (RoBB). Αυτά έχουν την παρακάτω δομή :

InstrType	Destination	Tag	Data	PC	Exception	Ready	Newest
2 bit	5 bit	5 bit	32 bit	32 bit	1 bit	1 bit	1 bit

InstrType	Type of instruction
Destination	Destination in RF
Tag	Tag of RS
Data	Result of Instruction
PC	PC of Instruction
Exception	Instruction Returned Exception
Ready	Instruction is ready to be committed
Newest	Instruction is the newest with given Destination

Τα RoBB ελέγχουν το CDB για τα κατάλληλα δεδομένα, και όταν αυτά υπάρξουν, τα καταγράφουν στο πεδίο “Data” και θέτουν ‘1’ στο πεδίο “Ready” τους. Ακόμη τα RoBB χρησιμοποιούν το σήμα εισόδου “Clear” για να ξανά μηδενίσουν το πεδίο “Ready” τους, πράγμα που χρειάζεται όταν εκτελεστούν, ή όταν ο Reorder Buffer (RoB) ανιχνεύσει exception. Αξιοσημείωτο είναι επίσης το πεδίο “Newest”. Αυτό ορίζεται ως ‘1’ όταν γίνει κάποιο issue στο RoBB και μηδενίζεται όταν το RoBB εντοπίσει το ίδιο “destination” στην είσοδό του με αυτό που υπάρχει στην μνήμη του, χωρίς όμως να είναι ενεργοποιημένο το σήμα “WrEn” του. Αυτό σημαίνει ότι κάποια νεότερη εντολή με αυτό το “destination” εισέρχεται στο RoB (αφου στο RoB η είσοδος “Destination” είναι συνδεδεμένη με όλες τις αντίστοιχες εισόδους των RoBB). Έτσι όταν μια θύρα του RoB αναζητά δεδομένα που προορίζονται για ένα “destination” , θα πάρει πάντα τα νεότερα, αυτά δηλαδή με την σημαία “Newest”.

Reorder Buffer



Το Reorder Buffer που κατασκευάστηκε χρησιμοποιεί 16 RoBB σε κατανομή ουράς. Για να επιτευχθεί αυτή η κατανομή χρησιμοποιήθηκαν δύο μετρητές, οι οποίοι ορίζουν τους pointers “Head” και “Tail”. Λόγω της υλοποίησης ουράς, η πραγματική χωρητικότητα είναι 15 Blocks.

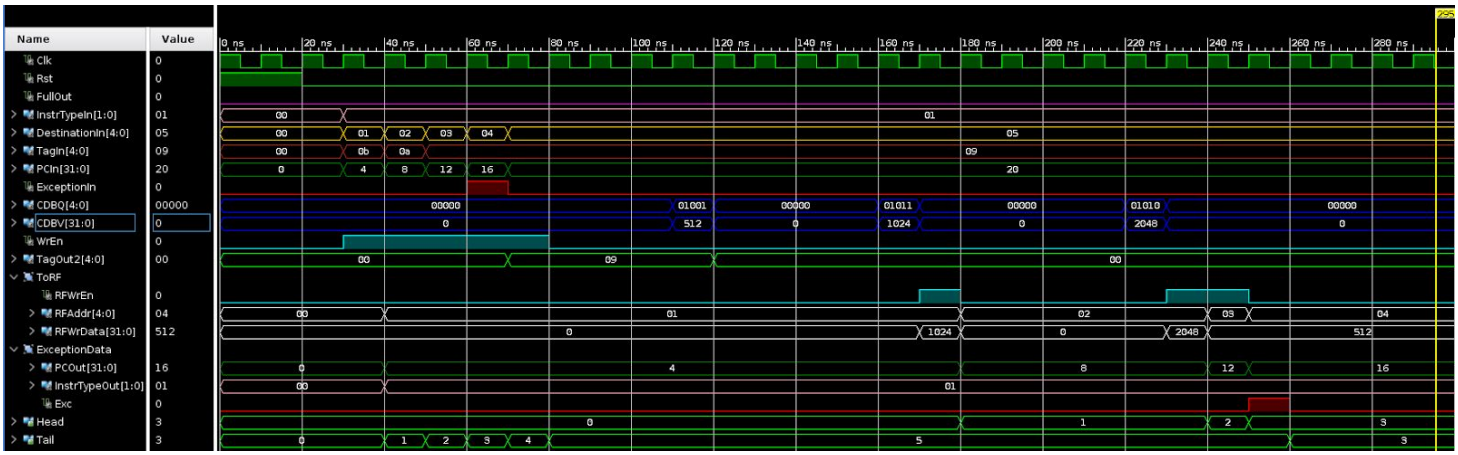
Για την σωστή λειτουργία του συστήματος χρειάζονται 2 Ports εξόδου από το RoB. Η κάθε μια από αυτές χρησιμοποιεί τα flags “Available1” και “Available2” για να σηματοδοτήσει ότι τα δεδομένα που της

ζητήθηκαν είναι διαθέσιμα. Κριτήριο για την αναζήτηση αυτών των δεδομένων είναι να υπάρχει κάποιο RoBB με κατάλληλο “tag”, και περίπτωση που υπάρχουν πάνω από ένα η θύρα εξάγει αυτό με το flag “Newest”.
Τέλος το RoB είναι σχεδιασμένο ώστε να σταματάει την λειτουργία του όταν το RoBB το οποίο πρόκειται να γίνει commit έχει υποστεί exception. Σε αυτην την περίπτωση το RoB αδειάζει την λίστα του χωρίς να κάνει commit τίποτα και εξάγει τα σήματα “Exc” ,”PCOut” και “InstrTypeOut”, τα οποία σηματοδοτούν την ύπαρξη exception, το PC της εντολής που το προκάλεσε, και τον τύπο της.

Testbench 1 - ROB

Στο παραπάνω τεστ ελέγχονται βασικές λειτουργίες της ROB. Στους χρόνους 30-60 ns το ROB γεμιζει 3 block της με εντολές οι οποίες έχουν Destination: ”00011” και Tag: “01001”. Σκοπός αυτών των εντολών είναι το γέμισμα του ROB, γι αυτόν τον λόγο και θα αναφέρονται ως “fillers” παρακάτω. Για τον έλεγχο των ports του ROB, οι ReadAddr 1 και 2 ορίζονται ως “00011” και “00100” αντίστοιχα. Μπορεί να παρατηρηθεί ότι μετά το γέμισμα του πρώτου RoBB (40 ns) το σήμα Available1 σηματοδοτεί ότι το port1 έχει τα κατάλληλα δεδομένα και το σήμα TagOut2 εξάγει το Tag της εντολής (“01001”). Μετά την εισαγωγή των πρώτων fillers, εισάγεται μια εντολή με Destination: “00100” και Tag: “01010”. Στον αμέσως επόμενο κύκλο το σήμα “Available2” σηματοδοτεί ότι το port2 έχει τα κατάλληλα δεδομένα και το σήμα TagOut2 εξάγει το Tag της εντολής (“01010”). Ταυτόχρονα στον ίδιο κύκλο το CDB δίνει τα κατάλληλα δεδομένα για αυτήν την εντολή, άρα και στον επόμενο κύκλο η port2 εξάγει τα κατάλληλα δεδομένα στο σήμα “DataOut2” και μηδενίζει το σήμα “TagOut2”. Στους χρόνους 70-140 ns εισήχθησαν ακόμη 7 εντολές “filler” και στον επόμενο κύκλο μια εντολή με Destination: “00100” και Tag: “01001” . Έτσι στον αμέσως επόμενο κύκλο η port2 αλλάζει την έξοδο της με TagOut2: “01001” το οποίο και σηματοδοτεί ότι η port2 εξάγει το νεότερο block με το κατάλληλο Destination Address. Μετά από αυτήν την εντολή το ROB γεμίζει με “fillers” μέχρι να ενεργοποιηθεί το σήμα “FullOut”.

Testbench 2 - ROB

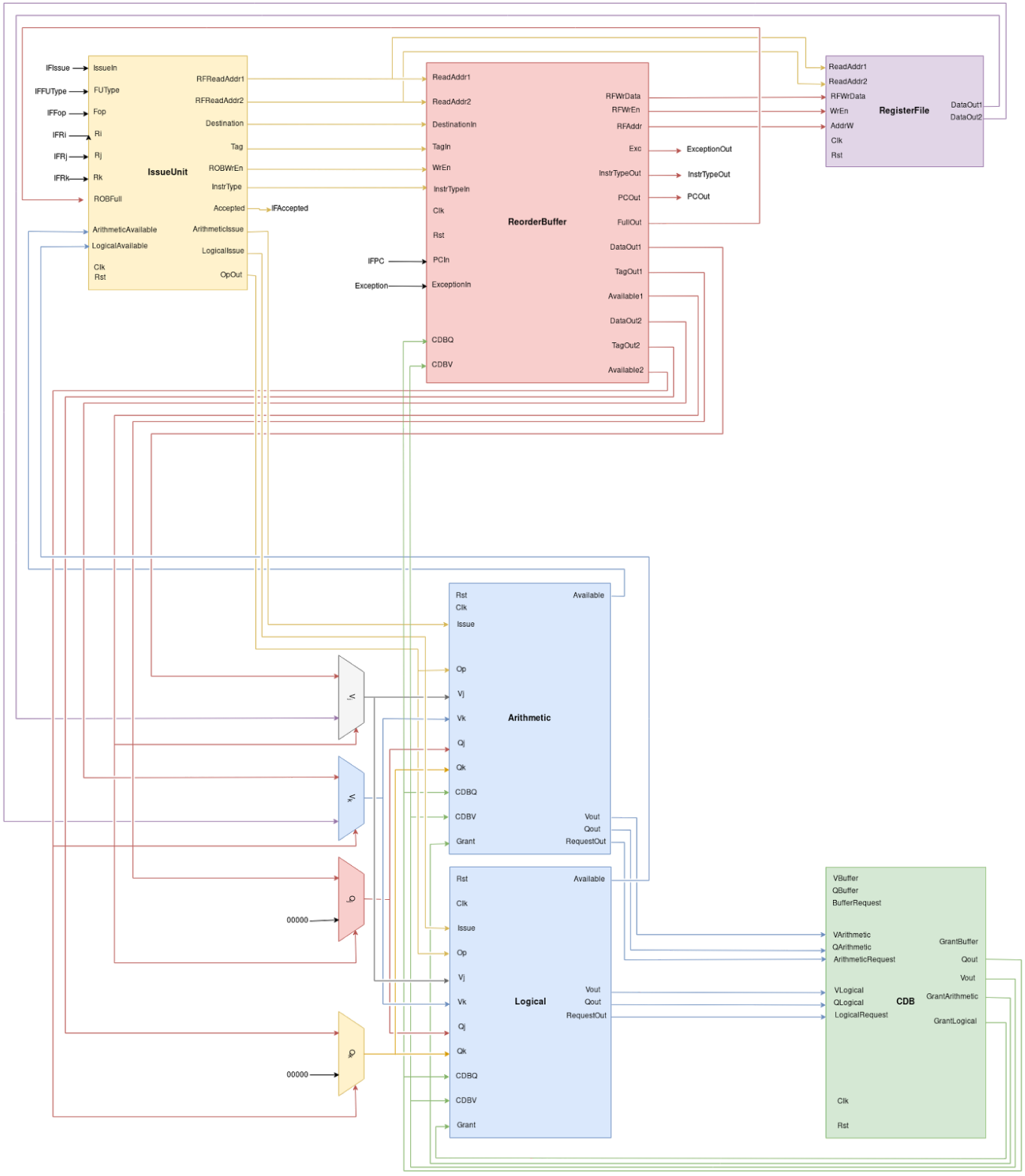


Αρχικά, στο Reorder Buffer εισάγονται εντολές ώστε να πάρει την παρακάτω μορφή (Στον παρακάτω πίνακα δεν εμφανίζονται τα πεδία data και newest) :

InstrType	Destination	Tag	PC	Exception	Ready	Block
01	1	01011	4	0	0	0
01	2	01010	8	0	0	1
01	3	01001	12	0	0	2
01	4	01001	16	1	0	3
01	5	01001	20	0	0	4

Έπειτα την χρονική στιγμή 110 ns η CDB εισάγει δεδομένα με tag “01001” ώστε να γίνουν ready τα blocks 2 - 4. Έπειτα την χρονική στιγμή 160 ns η CDB εισάγει δεδομένα με tag “01011” ώστε να γίνει ready το block 0. Στον αμέσως επόμενο κύκλο τα δεδομένα του block 0 εξάγονται προς την RF (βλ. σήματα RFWrEn, RFAddr, RFWrData). Την χρονική στιγμή 220 ns η CDB εισάγει δεδομένα με tag “01010” ώστε να γίνει ready το block 1. Στους αμέσως επόμενους δύο κύκλους το ROB εξάγει τα δεδομένα των block 1 και 2 διαδοχικά προς την RF άλλα σταματάει στο block 3 επειδή υπάρχει exception. Έτσι στον επόμενο κύκλο ενεργοποιείται το σήμα Exc και εξάγονται τα πεδία PC και InstrType της εντολής που προκάλεσε το exception. Τέλος στον επόμενο κύκλο αδειάζει η ουρά του ROB μεταφέροντας το Tail στο Head.

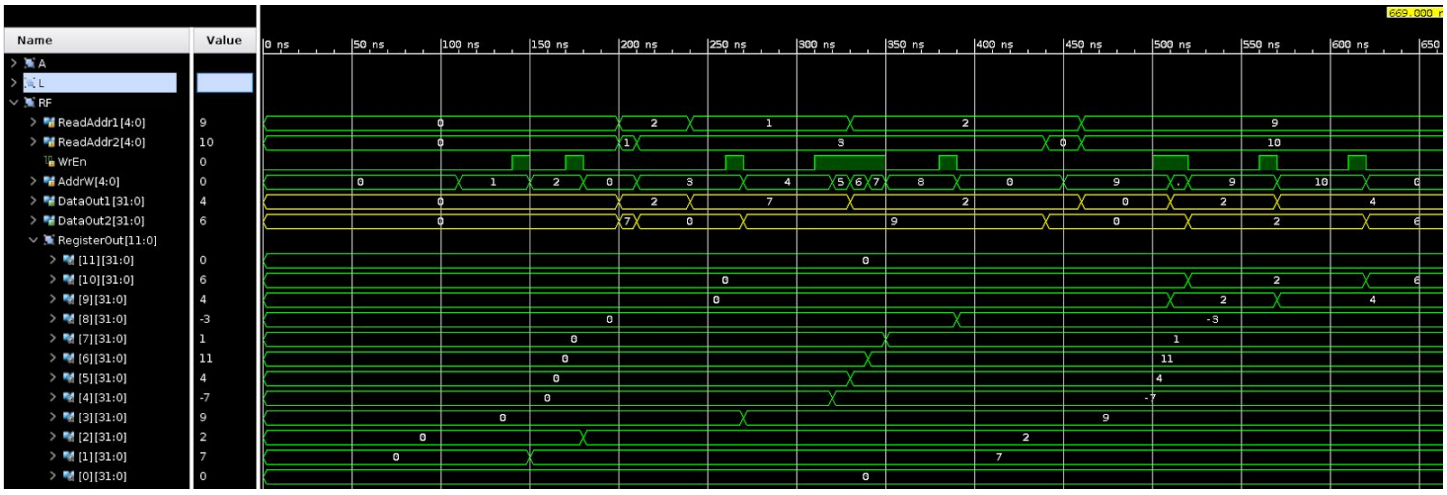
TOP MODULE


IFIssue
IFFUType
IFFop
IFRi
IFRj
IFRk
IssueIn
FUType
Fop
Ri
Rj
Rk
ROBFull
ArithmeticAvailable
LogicalAvailable
Clk
Rst
IssueUnit
RFReadAddr1
RFReadAddr2
Destination
Tag
ROBWrEn
InstrType
Accepted
ArithmeticIssue
LogicalIssue
OpOut
IFAccepted
ReadAddr1
ReadAddr2
DestinationIn
TagIn
WrEn
InstrTypeIn
Clk
Rst
PCIn
ExceptionIn
CDBQ
CDBV
ReorderBuffer
RFWrData
RFWrEn
RFAddr
Exc
InstrTypeOut
PCOut
FullOut
DataOut1
TagOut1
Available1
DataOut2
TagOut2
Available2
IFPC
Exception
ExceptionOut
InstrTypeOut
PCOut
ReadAddr1
ReadAddr2
RFWrData
WrEn
AddrW
Clk
Rst
RegisterFile
DataOut1
DataOut2
Rst
Clk
Issue
Op
Vj
Vk
Qj
Qk
CDBQ
CDBV
Grant
Available
Vout
Qout
RequestOut
Arithmetic
Logical
Vj
Vk
Qj
Qk
00000
00000
VBuffer
QBuffer
BufferRequest
VArithmetic
QArithmetic
ArithmeticRequest
VLogical
QLogical
LogicalRequest
CDB
GrantBuffer
Qout
Vout
GrantArithmetic
GrantLogical
Clk
Rst

Η διεπαφή του TOP Module παρέμεινε ίδια με του milestone 2, με εξαίρεση (pun intended) τα σήματα για exception handling (“Exc” ,”PCOut” και “InstrTypeOut”). Ακόμη χρησιμοποιείται και ένα σήμα “ExceptionIn” ώστε να εισαχθούν exceptions σε μία εντολή στο ReorderBuffer. Αυτό το σήμα βρίσκεται καθαρά για λόγους Simulation αφού τα Functional Units στα πλαίσια του project δεν παράγουν exceptions.
Ο RoB συνδέεται με το Issue Unit για την λήψη εντολών, με την CDB για εισαγωγή δεδομένων, και με την Register File για την εξαγωγή των τελικών αποτελεσμάτων. Ακόμη με την χρήση πολυπλεκτών, αξιοποιώντας τα σήματα “Available1” και “Available2”, αντίστοιχα επιλέγονται δεδομένα από την Register File, ή απο το RoB προς τα RS.

TestBench 1 - TOP

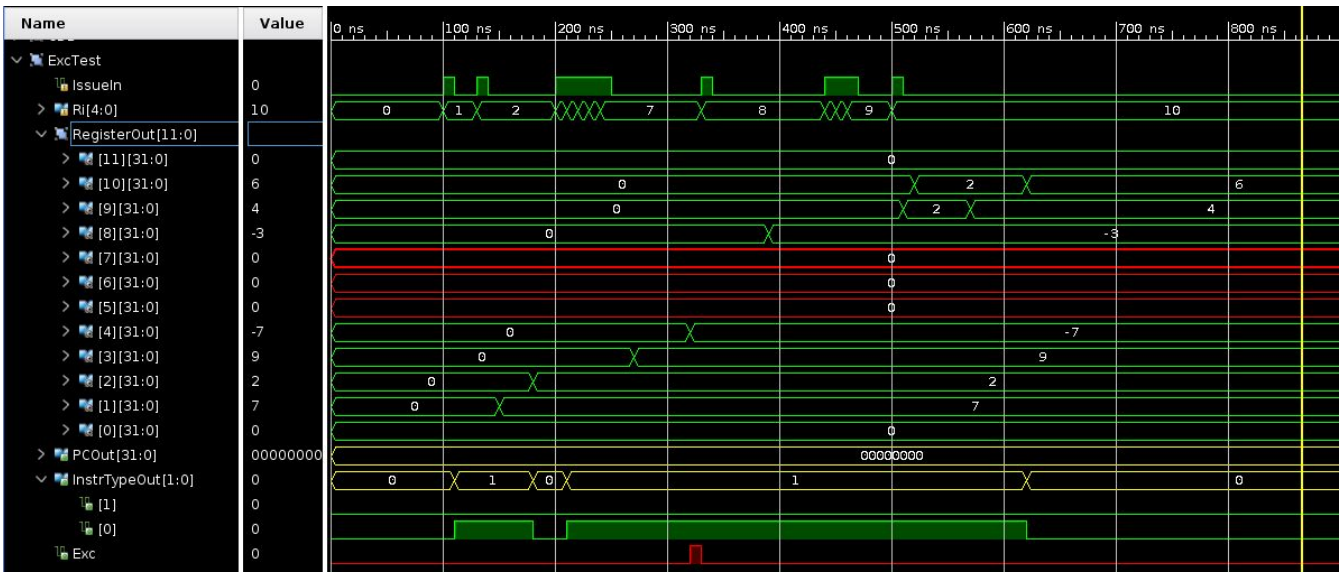
```
1.  LD R1, 7              // R1=7
2.  LD R2, 2              // R2=2
3.  ADD R3, R2, R1        // R3=7+2
4.  SUB R4, R2, R3        // R4=2-9=-7
5.  SHL R5, R2            // R5=2<<1=4
6.  OR R6, R2, R3         // R6=2 OR 9 = 11
7.  AND R7, R1, R3        // R7= 7 OR 9 = 1
8.  NOT R8, R2            // R8= NOT 2 = -3
9.  ADD R9, R2, R0        // R9= 2+0=2
10. ADD R10, R2, R0       // R10= 2+0=2
11. ADD R9, R9, R10       // R9= 2+2=4
12. ADD R10, R9, R10      // R10= 4+2=6
```



Ως πρώτο, εφαρμόστηκε το testbench του milestone2. Έτσι φαίνεται η διαφορά στην ύπαρξη και μη ενός ReorderBuffer, αφού πλέον η register file δέχεται τα δεδομένα της με την ίδια σειρά που οι εντολές έγιναν issue. Ακόμη έτσι ελέγχουμε και για τα ίδια Hazards που ελέγξαμε στο milestone 2.

TestBench 2 - TOP

```
1.  LD R1, 7              // R1=7
2.  LD R2, 2              // R2=2
3.  ADD R3, R2, R1        // R3=7+2
4.  SUB R4, R2, R3        // R4=2-9=-7
5.  SHL R5, R2            // R5=2<<1=4    ← With Exception !
6.  OR R6, R2, R3         // R6=2 OR 9 = 11
7.  AND R7, R1, R3        // R7= 7 OR 9 = 1
8.  NOT R8, R2            // R8= NOT 2 = -3
9.  ADD R9, R2, R0        // R9= 2+0=2
10. ADD R10, R2, R0       // R10= 2+0=2
11. ADD R9, R9, R10       // R9= 2+2=4
12. ADD R10, R9, R10      // R10= 4+2=6
```



Στο δεύτερο testbench χρησιμοποιήθηκε ο ίδιος κώδικας με το πρώτο, με μόνη διαφορά την εισαγωγή Exception στην εντολή 5. Έτσι το reorder buffer εξάγει σήμα “Exc” και ακυρώνει όλες τις εντολές οι οποίες είχαν γίνει issue πριν από αυτό (320 ns). Συνεπώς οι εντολές 6 και 7 δεν καταγράφονται στην RF, όμως οι επόμενες εντολές που γίνονται issue μετά το Exception εκτελούνται κανονικά (8 - 12). Έτσι χρησιμοποιώντας και τα σήματα “Exc”, ”PCOut” και “InstrTypeOut” μπορεί ένα Instruction Fetch Module με την σειρά του να κάνει issue τις κατάλληλες εντολές για Exception Handling.