Technical University of Crete

Αρχιτεκτονική Υπολογιστών

HPY 415

Αναφορά Εργαστηρίου 2 : Σύνδεση επιμέρους οντοτήτων και ολοκλήρωση βασικών δοκιμών λειτουργίας

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. Διαφορές/Αλλαγές από το εργαστήριο 1

Προκειμένου να μπορούν να συγχρονιστούν όσο το δυνατόν καλύτερα - τα μέχρι τώρα ανεξάρτητα units - χρειάστηκε να προβούμε σε μερικές αλλαγές, κυρίως όχι στο κομμάτι του σχεδιασμού αλλά της υλοποίησης. Ώστε τα εσωτερικά σήματα ελέγχου των μονάδων να ενεργοποιούνται τις κατάλληλες χρονικές στιγμές. Τέτοιες αλλαγές χρειάστηκε να γίνουν στην RF και περιγράφονται παρακάτω. Επίσης, λόγω της φύσης των σημάτων που υπάρχουν στο ISSUE (κάποια σήματα είναι απλά βραχυκυκλώματα) και του γεγονότος ότι πλέον υλοποιήσαμε το Top Module, αλλάξαμε την σχεδίαση του συγκεκριμένου unit.
Τελος, προκειμένου να μπορούμε να ελέγχουμε την λειτουργία του συστήματος όσο το δυνατόν πιο ρεαλιστικά (καθώς μετά από Reset δεν πρέπει να υπάρχει στην RF κάποια valid τιμή και δεν υπάρχει ακόμα η υλοποίηση της memory) προσθέσαμε την δυνατότητα να γίνονται εντολές I-type. Συγκεκριμένα πλέον υποστηρίζονται οι εντολές addi, subi, andi και ori. Με την βοήθεια του Register 0 της RF (ο οποίος έχει συνέχεια την τιμή 0) μπορούμε πλέον να αρχικοποιήσουμε την RF.

1.1 ISSUE

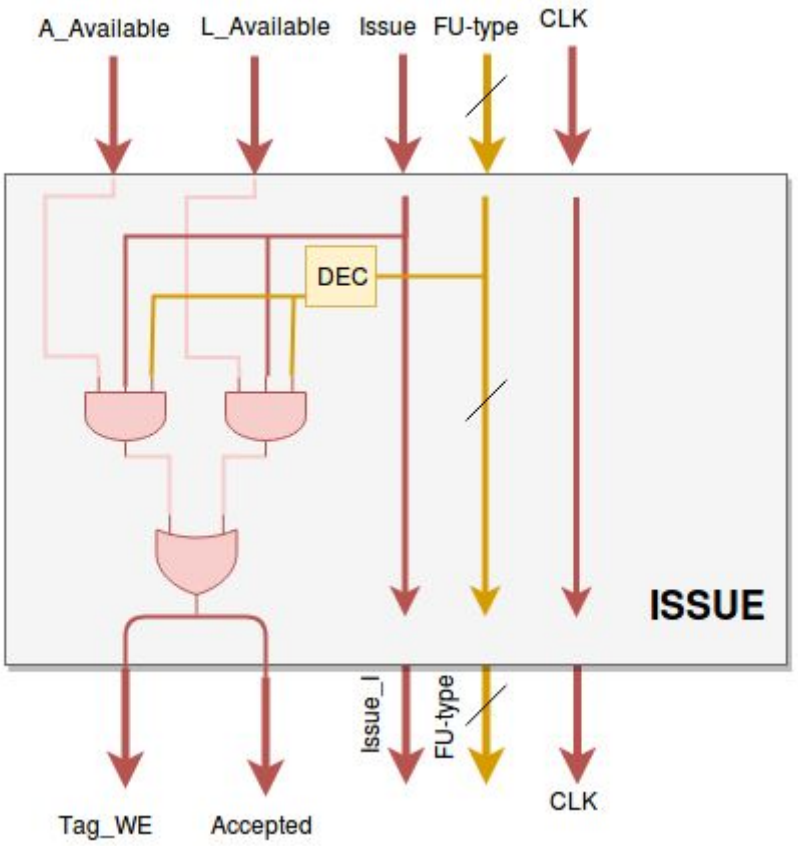
Πλέον το ISSUE unit μας αποτελείται μόνο από τα παρακάτω σήματα.

Σήμα	Μέγεθος (bits)	Είσοδος / Έξοδος	Περιγραφή
ISSUE	1	In	Έκδοση εντολής
FU-type	2	In	Τύπος εντολής
A_Available	1	In	Διαθέσιμο Αριθμητικό RS
L_Available	1	In	Διαθέσιμο Λογικό RS
Accepted	1	out	Αποδοχή εντολής
Tag_WE	1	out	Ενεργοποίηση εγγραφής στον Ri

Πίνακας 1.1.1 *- Διεπαφή ISSUE.*

Λειτουργικότητα:

Δεν έχει αλλάξει η λειτουργικότητα, η μόνη διαφορά είναι πλέον ότι τα σήματα που ήταν βραχυκυκλώματα από αυτήν την στιγμή και έπειτα πηγαίνουν από τον “έξω κόσμο” αμέσως στα units που τα χρειάζονται (παρουσιάζεται στο Top Module σχεδιάγραμμα παρακάτω). Αυτό που πλέον κάνει το ISSUE, και μόνο αυτό σε αντίθεση με πριν, είναι ο έλεγχος αποδοχής ή όχι μιας εντολής που θέλει να μπει στο σύστημα ανάλογα με την κατάσταση του.



Σχηματικό Διάγραμμα 1.1.1 *- ISSUE*

1.2 RF

Όπως αναφέραμε παραπάνω για τον RF δεν έχει πραγματοποιηθεί κάποια διαφορά στην σχεδίαση του ή στους “εξωτερικούς” χρονισμούς διεπαφής με το υπόλοιπο σύστημα. Αντίθετα αυτό που έχει αλλαχθεί είναι η εσωτερική υλοποίηση του ώστε να μπορούν οι καταχωρητές να έχουν έτοιμες τις τιμές εισόδου τους καθώς και το σήμα ελέγχου enable πριν το rising edge (στο οποίο γράφουμε τις τιμές) προκειμένου να μην καθυστερείται η εκτέλεση ακόμα ένα κύκλο με την ήδη υπάρχουμε υλοποίηση (η μόνη διαφορά που μπορεί να παρατηρηθεί στα simulation είναι να έχουμε μισό κύκλο πριν τα σήματα που αναφέρονται παραπάνω).

Old RF implementation (Για την εγγραφη στους Registers)

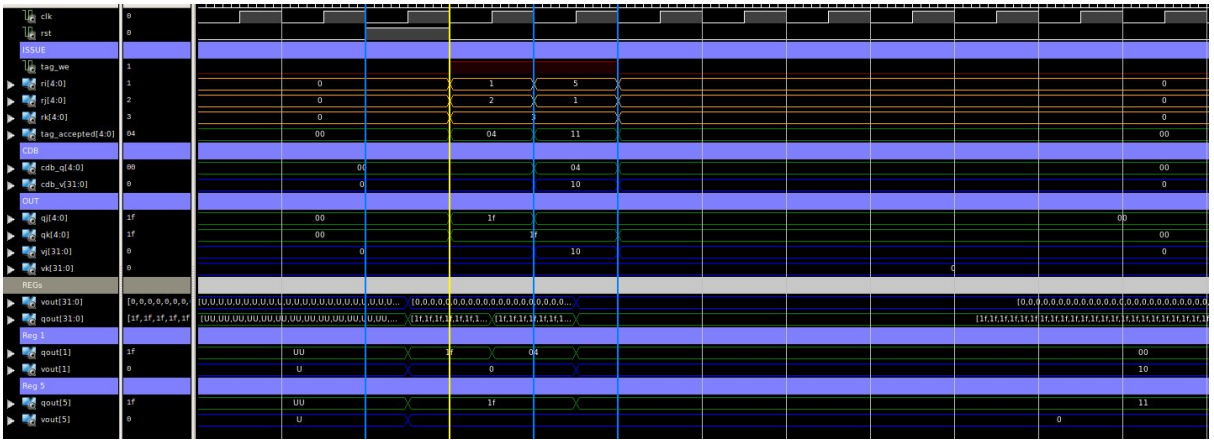
```
if(falling_edge(CLK)) then
    for n in 31 DOWNTO 0 LOOP
        if (Tag_WE='1' AND n=to_integer(UNSIGNED(Ri))) then
            QIN(to_integer(UNSIGNED(Ri)))<=Tag_Accepted;
            WEN(to_integer(UNSIGNED(Ri)))<='1';
        elsif (CDB_Q/="00000" AND QOUT(n)=CDB_Q) then
            VIN(n)<=CDB_V;
            QIN(n)<="00000";
            WEN(n)<='1';
        else
           WEN(n)<='0';
        end if;
    end loop;
else
    for n in 31 DOWNTO 0 LOOP
        WEN(n)<='0';
   end loop;
end if;
```

New RF implementation (Για την εγγραφη στους Registers)

```
Sel <= '0' WHEN Tag_WE='1' AND ID = Ri ELSE '1';

En <= '1' WHEN Tag_WE='1' AND ID = Ri ELSE
         '1' WHEN CDB_Q/= "00000" AND CDB_Q = Q_O ELSE
         '0';

V_IN : Mux_2x32bits
Port map( In0  => "00000000000000000000000000000000",
              In1  => CDB_V,
              Sel  => Sel,
              Outt => V_I);

Q_IN : Mux_2x5bits
Port map( In0  => Tag_Accepted,
              In1  => "00000",
              Sel  => Sel,
              Outt => Q_I);

Reg : Reg_V_Q_N
Port map( CLK  => CLK,
              RST  => RST,
              EN   => En,
              VIN  => V_I,
              QIN  => Q_I,
              VOUT => V,
             QOUT => Q_O);
Q <= Q_O;
```

Λειτουργικότητα:

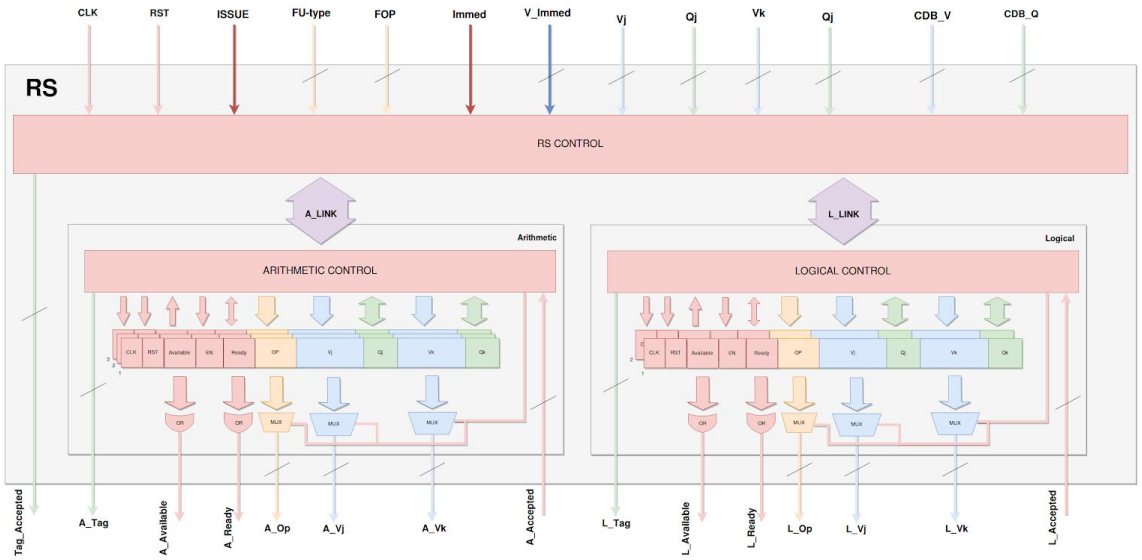
Αυτό που ουσιαστικά κάναμε είναι να δημιουργήσουμε ένα αυτόνομο Register ο οποίος είναι σε θέση χωρίς την άμεση ανάγκη για ρολόι πλέον (υπάρχει με έμμεσο τρόπο επειδή στα signal που γίνονται οι έλεγχοι αλλάζουν κάθε κύκλο) να μπορεί να επιλέξει τις τιμές εισόδου του καταχωρητή καθώς και το αν πρέπει να γράψει τα δεδομένα αυτά (γνωρίζοντας μαζί και το ID του) είτε αυτά έρχονται από το ISSUE είτε από το CDB. Έπειτα, δημιουργήσαμε τα 32 στιγμιότυπα δίνοντας στο καθένα διαφορετικό ID.



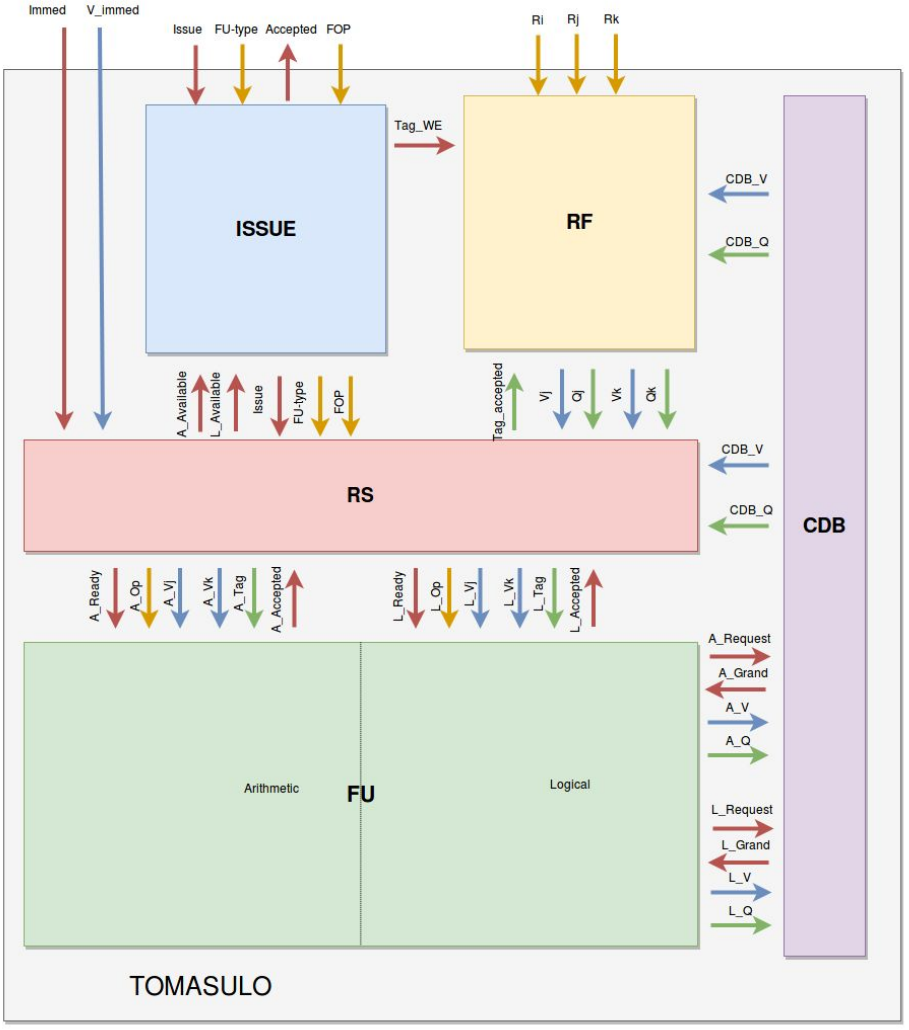
Εικόνα 1.2.1 Simulation - RF

1.3 RS

Από την στιγμή που ενσωματώσαμε την δυνατότητα να μπορούν να γίνουν εντολές I-type χρειάζεται να αναφέρουμε τα εξής. Αρχικά θεωρούμε ότι στο V_immed, το οποίο δέχεται το σύστημα, έχει γίνει ήδη είτε το απαραίτητο sign extension είτε το zero extension και είναι ένα έτοιμο προς πράξη (32bits) signal. Θεωρούμε τις ίδιες κωδικοποιήσεις που είχαμε κάνει για τις πράξεις με την μόνη διαφορά την ύπαρξη του σήματος Immed το οποίο όταν είναι ενεργό μας ενημερώνει για την διαφορετική λειτουργικότητα που πρέπει να έχουμε. Αυτό δεν σημαίνει ότι οι εντολές I-type δεν παύουν να διαχειρίζονται σαν αριθμητικές ή λογικές, αντίθετα διαχειρίζονται ακριβώς με τον ίδιο τρόπο από το σύστημα (π.χ. για το αν θα τις αποδεχτούμε με βάση την διαθεσιμότητα) και είναι απλά προέκταση αυτών, άρα το μόνο που χρειάστηκε να κάνουμε είναι να προσθέσουμε δύο πολυπλέκτες με σήμα ελέγχου το Immed. Όταν είναι ανενεργό στα RS θα πηγαίνει η τιμή του Rk από το RF, για το Vk και Qk ενώ όταν είναι ενεργό θα προωθούμε το V_immed με Q=”00000”. Απο την στιγμή που είναι επιπρόσθετη λειτουργία και δεν έχει αλλάξει κάτι άλλο δεν έχουν προστεθεί στην αναφορά κυματομορφές, παρόλα αυτά υπάρχουν τα testbench και simulation αρχεία μέσα στο project folder στο οποίο μπορεί να επαληθευτεί η λειτουργικότητα του.



Σχηματικό Διάγραμμα 1.3.1 *- RS*



Σχηματικό διάγραμμα ολόκληρου του συστήματος

2 Έλεγχος Συστήματος

Ο έλεγχος του συστήματός μας πραγματοποιείται μέσω ενός προγράμματος που χωρίζεται σε δύο τμήματα. Στο πρώτο τμήμα χρησιμοποιούμε επιπλέον εντολές που σχεδιάσαμε στο σύστημά μας, addi, ori, subi, με σκοπό να περάσουμε τιμές στους καταχωρητές και να έχουν valid δεδομένα. Όλοι οι καταχωρητές αρχικά έχουν την τιμή μηδέν χωρίς να είναι valid.

CC	ID	Instr	Assembly
1	ia	RST	
2	ib	ori	$10,$0,10
3	ic	addi	$1,$0,1
4	id	subi	$11,$0,-5
5	ie	NOP x5	

Πίνακας 2.1 *Αρχικοποίηση Καταχωρητών με επιπλέον εντολές που προσθέσαμε.*

Αρχικά, κάνουμε reset και στη συνέχεια πραγματοποιούμε την εντολή ori μεταξύ του καταχωρητή $10 και της τιμή 10. Εφόσον όλοι οι καταχωρητές έχουν την τιμή 0 το αποτέλεσμα που θα αποθηκευτεί στον συγκεκριμένο καταχωρητή είναι η τιμή 10.

Ομοίως στον καταχωρητή $1 μέσω της addi περνάμε την τιμή 1 και στον καταχωρητή $11 μέσω της subi αποθηκεύουμε την τιμή 5.

Στη συνέχεια, αφήνουμε 5 κύκλους να περάσουν ώστε να ολοκληρωθούν οι παραπάνω εντολές και να είμαστε έτοιμοι να προχωρήσουμε στον έλεγχο του Tomasulo.

Στο πρόγραμμα με το οποίο ελέγχουμε τις δυνατότητες της υλοποίησής μας, εξετάζουμε τις περιπτώσεις όπου:

- **Έχουμε γεμίσει όλους τους Reservation Stations** και εντολές που ζητάνε ISSUE για αυτό το είδος εντολής απορρίπτονται.
- Εντολές περιμένουν να υπολογιστεί το αποτέλεσμα που χρειάζονται από μία άλλη λειτουργική μονάδα (περίπτωση **read after write - RAW**).
- Εντολές που λήφθηκαν αργότερα (**in order issue**) εκτελούνται νωρίτερα από τον επεξεργαστή, όταν είναι έτοιμες προς εκτέλεση σε αντίθεση με άλλες που περιμένουν αποτελέσματα από άλλες μονάδες (**out of order execution και out of order completion**).
- Ο CDB εφαρμόζει **Round Robin** λογική για την επιλογή ποιας εντολής θα λάβει από την FU όταν είναι περισσότερες από μία που ζητάνε Grant.
- Ο RS εφαρμόζει **Round Robin** λογική για την επιλογή της εντολής που θα εισάγει στην FU όταν δύο ή περισσότερες από αυτές είναι ready στον ίδιο κύκλο.

- Εντολές που γράφουν στον ίδιο καταχωρητή με δεδομένο ότι δεν έχει προλάβει να εκτελεστή πλήρως οποιαδήποτε πλην της τελευταίας (περίπτωση **write after write - WAW**).
- Εντολές που γίνονται ISSUE και χρειάζεται να διαβάζουν το αποτέλεσμα του CDB να μπορούν στον ίδιο κύκλο να παίρνουν τα αποτελέσματα του (λειτουργία **fall through** για την RF).

CC	ID	Instr	Assembly
10	i1	add	$2,$1,$1
11	i2	or	$3,$2,$1
12	i3	sub	$4,$1,$1
13	i4	not	$5,$1,$1
14	i5	shift	$6,$3,$1
15	i6	and	$7,$4,$1
16	i7	and	$7,$4,$1
17	i8	and	$7,$4,$1
18	i9	sub	$8,$2,$1
19	i10	and	$9,$3,$5
20	i11	shift	$9,$2,$1

Πίνακας 2.2 *Κύριο πρόγραμμα ελέγχου λειτουργιών Tomasulo*

CC	Arithmetic						Logical				CDB	RF	
	RS			FU			RS		FU				
	A1	A2	A3	A1	A2	A3	L1	L2	L1	L2			
10	i1												
11	i1			i1			i2						
12	i1	i3			i1		i2						
13	i1	i3		i3		i1	i2	i4			i1		
14	i1	i3	i5		i3		i2	i4	i4		i1=R2	i1=R2	R2 = 2
15		i3	i5			i3	i2	i4	i2	i4	i4/i3		
16		i3	i5			i3	i2	i4		i2	i2/i3	i4=R5	R5 = -2
17		i3	i5				i2	i8			i2	i3=R4	R4 = 0
18	i9		i5				i2	i8	i8			i2=R3	R3 = 3
19	i9		i5	i5			i10	i8		i8	i8		
20	i9	i11	i5	i9	i5		i10	i8	i10			i8=R7	R7 = 0
21	i9	i11	i5	i11	i9	i5	i10			i10	i5/i10		
22	i9	i11	i5		i11	i9	i10				i9/i10	i5=R6	R6 = 6
23	i9	i11			i11	i9	i10				i9	i10=R9	
24	i9	i11				i11					i11	i9=R8	R8 = 1
25		i11										i11=R9	R9 = 4

Πίνακας 2.3 *Αναμενόμενη ροή προγράμματος σύμφωνα με τη σχεδίασή μας.*

Στον Πίνακα 2.3 μπορούμε να δούμε πώς ακριβώς εκτελούνται οι εντολές, από ποια σημεία του επεξεργαστή περνούν σε κάθε κύκλο, καθώς και τις τιμές που αποθηκεύονται στους καταχωρητές.

Οι εντολές στην RS με ανοιχτό πράσινο είναι οι εντολές που είναι έτοιμες αλλά δεν έχουν ολοκληρωθεί ενώ όταν μια εντολή έχει έντονο πράσινο χρώμα σημαίνει ότι στο συγκεκριμένο κύκλο η εντολή ολοκληρώθηκε.

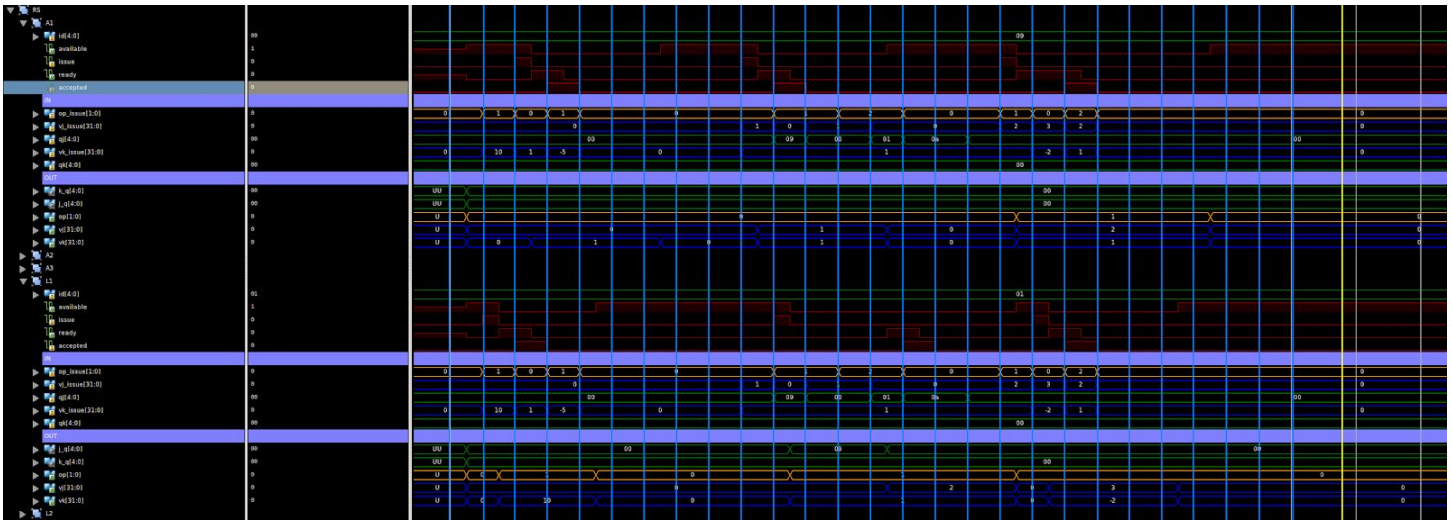
Εντολές με κόκκινο χρώμα υποδηλώνουν ότι περιμένουν την ολοκλήρωση προηγούμενης εντολής για να εκτελεστούν ενώ το έντονο κόκκινο στον κύκλο 23 στην εντολή i10 σημαίνει ότι η τιμή δεν γράφεται στον καταχωρητή (λόγω του WAW).

Θα θέλαμε να τονίσουμε ότι το τι συμβαίνει ανά κύκλο ρολογιού όπως φαίνεται στον παραπάνω πίνακα, συμφωνεί απόλυτα με τα αποτελέσματα του simulation.

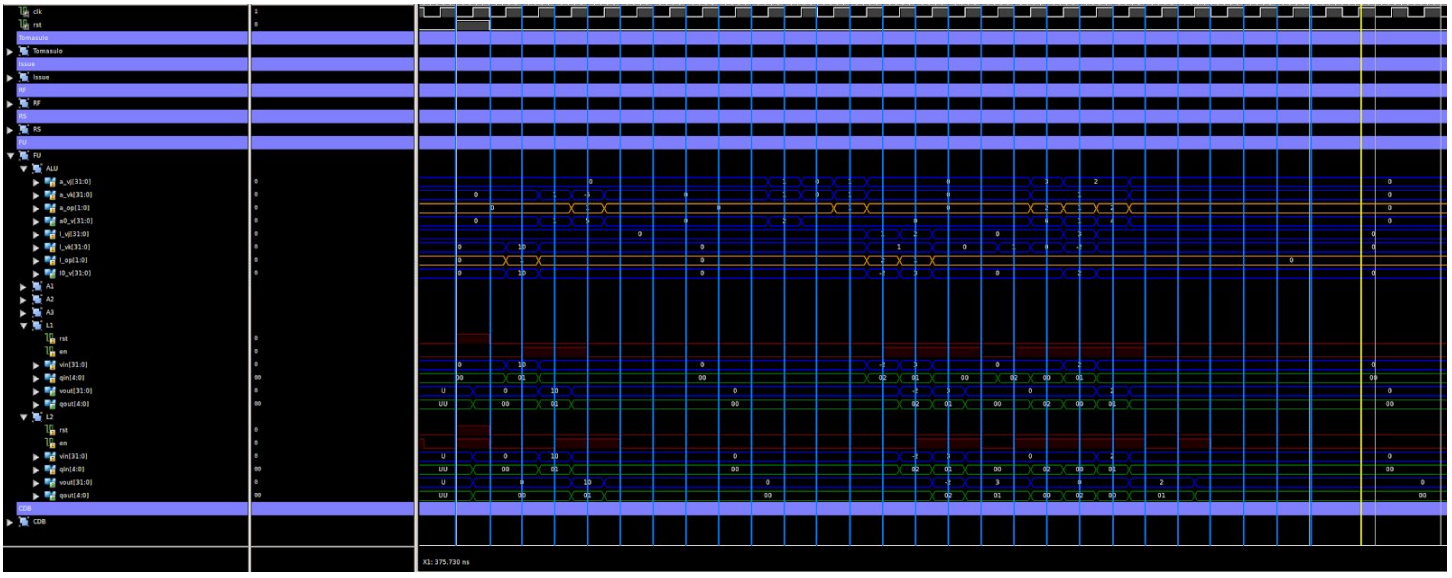
Τι αξίζει να προσέξει κανείς

- **Read After Write**: Στον κύκλο 11, η εντολή i2 εισάγεται στον RS αλλά δεν είναι έτοιμη ακόμα αφού περιμένει την ολοκλήρωση της εντολής i1. Αντίστοιχο φαινόμενο παρατηρείται και μεταξύ των εντολών i5 και i2 όπου η πρώτη περιμένει τη δεύτερη.
- **Full Reservation Stations**: Στους κύκλος 15 και 16 εισάγουμε μια λογική εντολή σύμφωνα με τον πίνακα 2.2. Παρατηρούμε ότι η συγκεκριμένη εντολή δεν γίνεται δεκτή αφού τα Reservation Stations για τις logical εντολές είναι κατειλημμένα από τις i2 και i4.
- **CDB Round Robin**: Στον κύκλο 15 έχουμε δύο εντολές, μια αριθμητική και μια λογική όπου βρίσκονται στην είσοδο του CDB και εκείνος καλείται να δώσει προτεραιότητα σε μια απο τις δύο. Επειδή η τελευταία εντολή που πέρασε από τον CDB ήταν i1 όπου είναι αριθμητική, σύμφωνα με τη λογική Round Robin δίνεται προτεραιότητα στην i4 που είναι λογική εντολή. Συνεχίζοντας, στον επόμενο κύκλο θα δοθεί προτεραιότητα στην αριθμητική εντολή, i3 έναντι i2. Αντίστοιχη περίπτωση έχουμε και στους κύκλους 21 και 22 όπου δίνεται προτεραιότητα σε αριθμητική και σε λογική αντιστοίχως.
- **Out of Order**: Στον κύκλο 11 έχουμε λάβει στο Reservation Station την εντολή i2 που όμως δεν αλλάζει σε κατάσταση ready πριν τον κύκλο 14. Στον κύκλο 13, λαμβάνουμε την εντολή i4 η οποία είναι σε κατάσταση ready και ξεκινά την εκτέλεσή της αμέσως. Σαν αποτέλεσμα έχουμε ότι η εντολή i4 εκτελείται πριν την i2 παρόλο που η δεύτερη εισήχθη νωρίτερα στο Reservation Station.
- **RS Round Robin**: Στον κύκλο 18 έχουμε δύο εντολές, i9 και i5 όπου έγιναν ready ταυτόχρονα. Εφόσον στους προηγούμενους κύκλους το Reservation Station A2 ήταν έτοιμο και στο A3 έχω εντολή έτοιμη, σύμφωνα με τη λογική Round Robin, θα εισάγουμε την εντολή i5 στην FU, όπως πολύ σωστά φαίνεται στον πίνακα 2.3 και στον επόμενο κύκλο θα περάσουμε την εντολή i9.
- **Fall Through**: Η εντολή i5 περιμένει την εντολή i2 να ολοκληρωθεί ώστε να διαβάσει το περιεχόμενο του καταχωρητή $3. Παρατηρούμε ότι στον κύκλο που η εντολή i2 ολοκληρώνεται και γίνεται εγγραφή στον καταχωρητή, η εντολή i5 μπορεί να διαβάσει το περιεχόμενο του καταχωρητή και επομένως αλλάζει στον ίδιο κύκλο σε κατάσταση ready, με επιτυχία.
- **Write After Write**: Στον κύκλο 23 βλέπουμε ότι το αποτέλεσμα της εντολής i10 δεν γράφεται στον καταχωρητή $9. Αυτό συμβαίνει γιατί η εντολή i11 που εκτελέστηκε πιο πρόσφατα, χρησιμοποιεί και εκείνη τον καταχωρητή $9 για να γράψει. Αυτό έχει σαν αποτέλεσμα, το tag στον καταχωρητή να αλλάξει πριν στείλει ο CDB το αντίστοιχο σήμα της εντολής i10 σε αυτόν. Έτσι τα αποτελέσματα της πρώτης εντολής χάνονται και αποθηκεύουμε μόνο αυτά της πιο πρόσφατης.

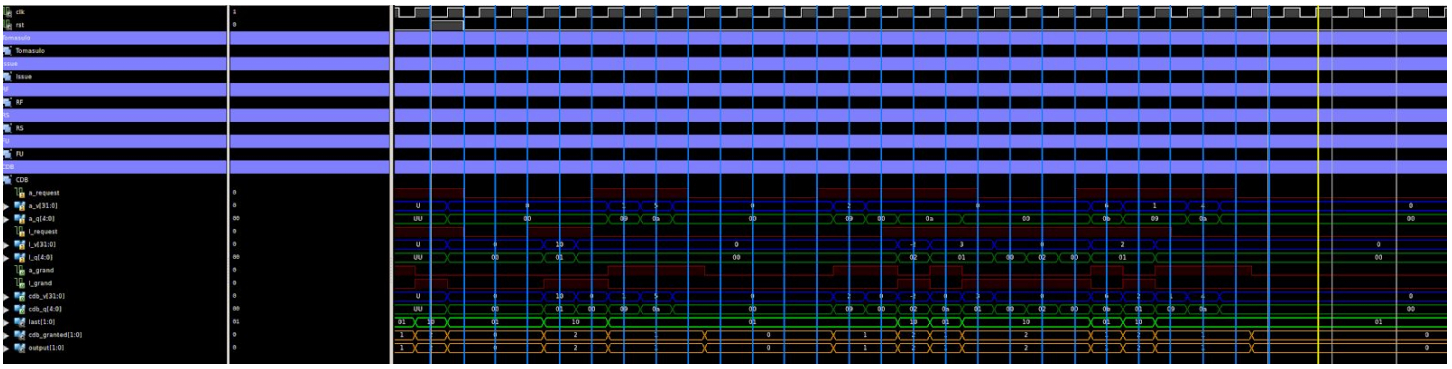
Παρακάτω, μπορούμε να επιβεβαιώσουμε ότι κάθε μονάδα λειτουργεί σωστά, σύμφωνα με τον πίνακα 2.3 παρατηρώντας τις παρακάτω εικόνες που αποτελούν το Simulation των προγραμμάτων στους πίνακες 2.1 και 2.2. Δεν γίνεται επιπλέον ανάλυση των Εικόνων 2.1 - 5 αφού η περιγραφή αυτών συμφωνεί με όσα έχουμε πει παραπάνω.



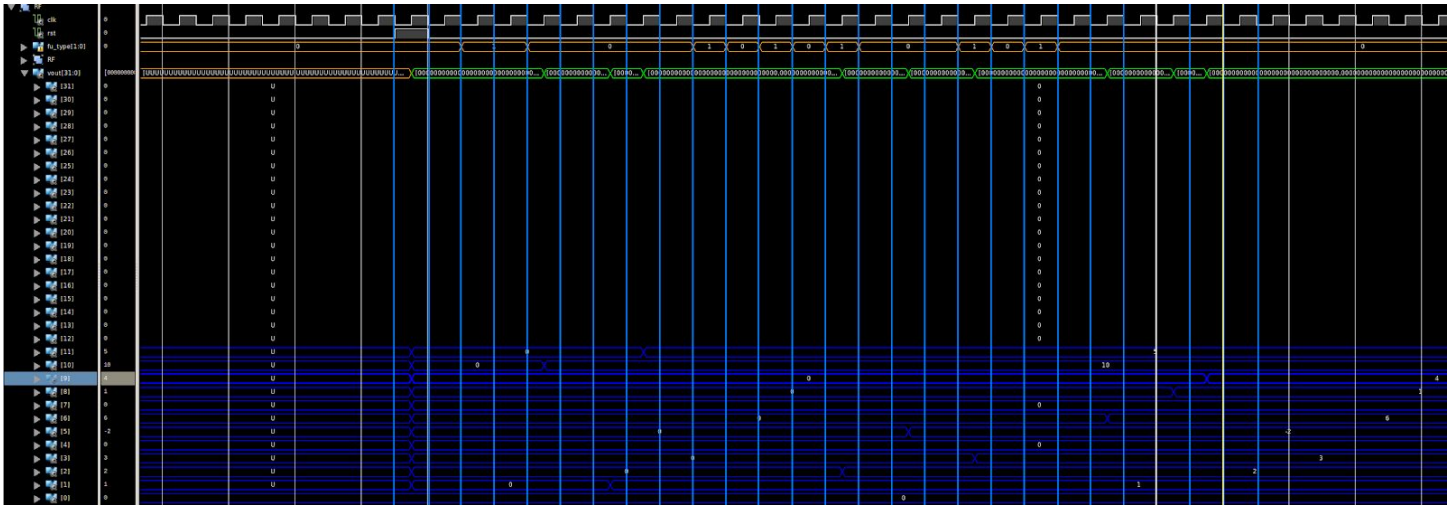
Εικόνα 2.1 *Simulation - Reservation Station Arithmetic 1 and Logical 1*



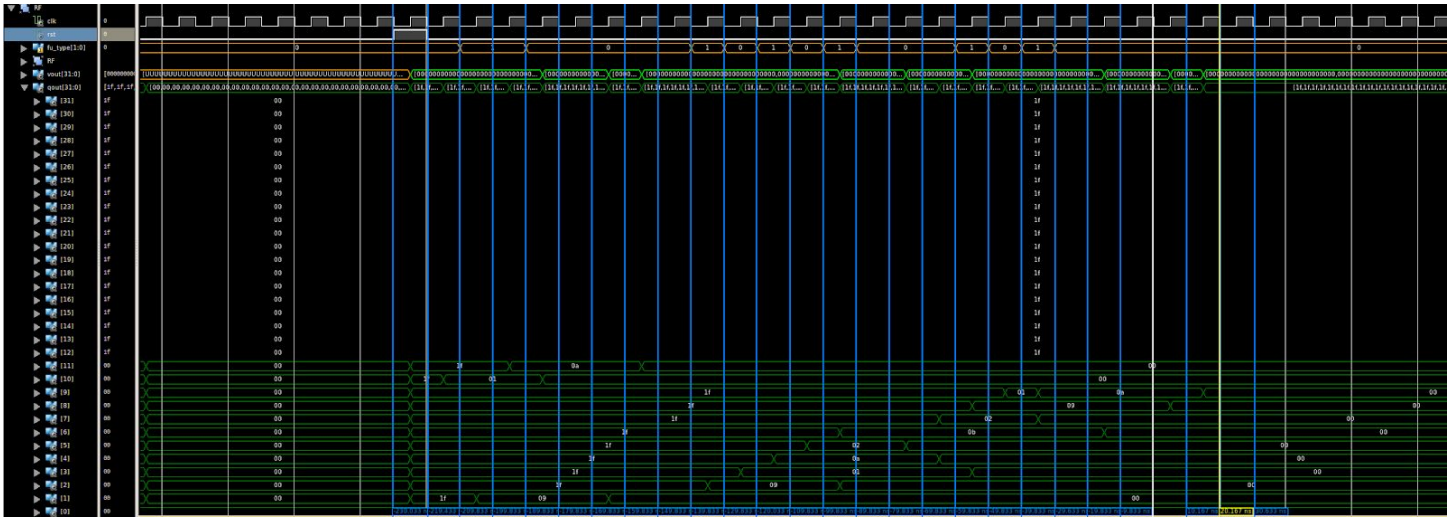
Εικόνα 2.2 *Simulation - Functional Unit Logical 1 & 2*



Εικόνα 2.3 *Simulation - Common Data Bus*



Εικόνα 2.4 *Simulation - Register File Values*



Εικόνα 2.5 *Simulation - Register File Q field*