Αρχιτεκτονική Υπολογιστών

Αναφορά 2ου Εργαστηρίου

Ομάδα Εργασίας:
Βαϊλάκης Αποστόλης - 2014030174
Φωτάκης Τζανής - 2014030054

Υλοποίηση

Ολοκληρώνοντας τα επιμέρους κομμάτια για την υλοποίηση του αλγόριθμου Tomasulo με βάση τις προδιαγραφές του 1ου εργαστηρίου, δημιουργήθηκαν και αλλάχθηκαν κάποια από τα χρησιμοποιούμενα modules. Πιο συγκεκριμένα, αρχικά δημιουργήθηκε το “Top Module” το οποίο περιλαμβάνει όλα τα modules του προηγούμενου εργαστηρίου μαζί με την κατάλληλη διασύνδεσή τους. Λόγω του γεγονότος ότι για να γίνουν οι απαραίτητες μαθηματικές πράξεις με τις functional units (Arithmetic & Logical) απαιτείται να φορτωθεί τουλάχιστον μία αρχική τιμή διάφορη του μηδενός σε έναν Register στην Register File του συστήματος, χρησιμοποιήθηκε η ήδη υπάρχουσα υποδομή για την ενσωμάτωση του Load Buffer. Η λειτουργία αυτή επιτυγχάνεται δίχως την υλοποίηση του τελευταίου, αλλά με την απλή προσομοίωσή του στο simulation του συνολικού συστήματος θέτοντας καταλλήλως τα αντίστοιχα σήματα.

Αλλαγές

Απαραίτητες ήταν οι αλλαγές κάποιων modules για την εύρυθμη λειτουργία του συστήματος. Πιο συγκεκριμένα, το σύστημα θεωρεί ότι το ουδέτερο tag είναι το “00000”. Αυτό οδηγεί στο ότι μετά το γράψιμο του tag σε κάποιο register της Register File, η τελευταία περιμένει από το CDB ένα tag ίσο με το tag που βρίσκεται σε κάποιον από τους registers της ώστε να γράψει τα δεδομένα του CDB στον register αυτό. Αμέσως μετά την εγγραφή των δεδομένων το tag πρέπει να μηδενιστεί ώστε να μην πάρει λανθασμένα κάποια άλλη τιμή από το CDB. Ο μηδενισμός αυτός γίνεται μέσω ενός σήματος reset προς τον καταχωρητή του συγκεκριμένου tag. Η αλλαγή που έγινε είναι ότι κατά την υλοποίηση του προηγούμενου εργαστηρίου το σήμα reset ερχόταν μαζί με τον ισότητα του Register Tag και του CDB Tag πράγμα που οδηγούσε σε άμεση καταστροφή του σήματος αυτού αφού ο Register του Tag μηδενίζονταν και έτσι δεν ίσχυε η ισότητα. Η λύση που δόθηκε είναι η καθυστέρηση του σήματος reset κατά έναν κύκλο, αμέσως μετά δηλαδή την εγγραφή των data στην Register File.

Ακόμα, ήταν απαραίτητη η αλλαγή της λογικής με την οποία γράφονται τα δεδομένα στο Reservation Station. Στην προηγούμενη υλοποίηση δεν συμπεριλαμβάνονταν η περίπτωση στην οποία την στιγμή που γράφεται το Reservation Station, του οποίου τα δεδομένα δεν είναι ακόμα γραμμένα στην Register File, δηλαδή η Register File επιστρέφει στο Reservation Station τα tags των Register που αναζητά, τα δεδομένα αυτά δίνονται από το CDB ταυτοχρόνως. Έτσι, προστέθηκε ένας επιπλέον έλεγχος, ο οποίος δίνει προτεραιότητα στα δεδομένα του CDB όταν το δοσμένο από την Register File Tag συνάδει με αυτό που μεταφέρει αυτή την στιγμή το CDB.

Επιπλέον, στα Reservation Station χρησιμοποιείται ένα σήμα ονόματι Busy το οποίο ενδεικνύει ότι το Reservation Station είναι γεμάτο και δεν μπορεί να δεχτεί καινούρια εντολή. Κατά την αρχική υλοποίηση του, το σήμα αυτό μηδενίζονταν μόλις εισάγονταν στην functional unit η περιεχόμενη του συγκεκριμένου Reservation Station εντολή και εν συνεχεία εισάγονταν στο ίδιο μία νέα εντολή. Το παραπάνω οδηγούσε σε λανθασμένη εγγραφή των tag των καταχωρητών αφού τουλάχιστον δύο μη ολοκληρωμένες εντολές χρησιμοποιούν ίδιο tag στον καταχωτητή των αποτελεσμάτων τους. Το πρόβλημα αυτό αποφεύχθηκε απλά με την αλλαγή στην λογική του μηδενισμού του σήματος Busy, δηλαδή στον μηδενισμό του μόνο όταν στο CDB βρεθεί το tag του Resevation Station αυτού.

TESTBENCH

Για το έλεγχο του συστήματος χρησιμοποιήθηκαν οι θύρες που προορίζονταν για τον Load Buffer ώστε να καταχωρηθούν κάποιες αρχικές τιμές στους καταχωρητές 1 και 2. Αφότου έγινε αυτό, το testbench γεμίζει όλα τα functional Units (πρώτα το αριθμητικό και έπειτα το λογικό) με εντολές οι οποίες και εμφανίζουν read after write hazards. Στο σχήμα (1) φαίνονται και οι πρώτοι 10 καταχωρητές, μαζί με τα Tag τους σε σχέση με την πάροδο του χρόνου εκτέλεσης των εντολών.

Επίσης χρησιμοποιώντας διαδοχικές εντολές μεταξύ δύο καταχωρητών (Code Snippet Γραμμές 9-12) εμφανίζονται write after read και write after write hazards. Μέσω του σχήματος (1) φαίνεται πως οι καταχωρητές 9 και 10 λαμβάνουν απευθείας τις τελικές τιμές τους, χωρίς να λαμβάνουν τις αρχικές ($r9 = 2,$r10 = 2), πράγμα που προβλέπεται από τον αλγόριθμο Tomasulo.

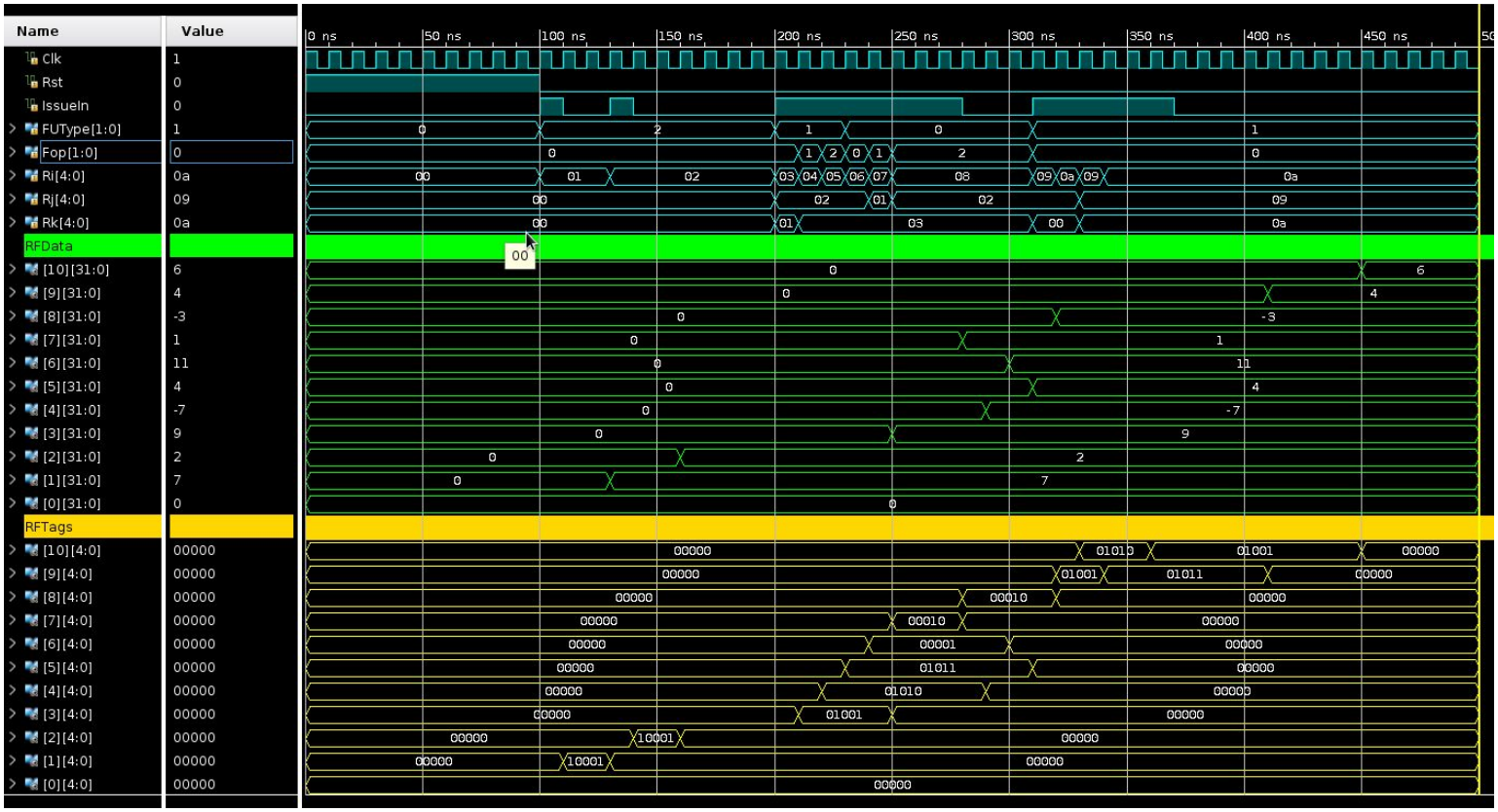
Το σχήμα (2) αναφέρεται στην Arithmetic Unit κατά την διάρκεια των πράξεων. Σε αυτό το σχήμα φαίνονται και οι αλλαγές οι οποίες προαναφέρθηκαν και παραπάνω, οι οποίες και αναγκάζουν κάθε RS να παραμένει Busy μέχρι τα δεδομένα της πράξης του να εμφανιστούν στο CDB. Σημαντικό ακόμη είναι και το σήμα “Available” το οποίο και ενημερώνει το “Issue Unit” για το εάν, και ποιο RS είναι διαθέσιμο. Μέσω αυτού του σήματος το “Issue Unit” καταγράφει το κατάλληλο tag σε κάθε register της ”RF”. Στο σχήμα (2), στην χρονική στιγμή 340ns το σήμα αυτό παίρνει την τιμή ‘0’, σηματοδοτώντας και την έλλειψη ελεύθερων Reservation Station.

Τέλος στο σχήμα (3) απεικονίζεται η λειτουργία της CDB. Ιδιαίτερη σημασία έχει το χρονικό διάστημα 260ns - 320ns στο οποίο και τα σήματα ArithmeticRequest και LogicalRequest συνυπάρχουν και ανταγωνίζονται για το CDB. Εκεί το CDB μέσω της φιλοσοφίας round robin εναλλάσει τα σήματα ArithmeticGrand και Logical Grand, όπως και τις αντίστοιχες εξόδους CDB.Q και CDB.V, εξαλείφοντας τις πιθανότητες για starvation.

Code Snippet:

```
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
```

Σχήμα 1



- 100ns - 130ns -> LD R1, 7
- 130ns - 160ns -> LD R2, 2
- 200ns - 250ns -> ADD R3, R2, R1
- 210ns - 290ns -> SUB R4, R2, R3
- 220ns - 310ns -> SHL R5, R2
- 230ns - 300ns -> OR R6, R2, R3
- 240ns - 270ns -> AND R7, R1, R3
- 270ns - 320ns -> NOT R8, R2 (30ns Delay Beause No RS is available)

- 310ns - ?!? -> ADD R9, R2, R0 (Tag is overridden so data go immediately to next RS)
- 320ns - ?!? -> ADD R10, R2, R0 (Similarly to above, thus we can't see it in the RF)
- 330ns - 410ns -> ADD R9, R9, R10
- 340ns - 450ns -> ADD R10, R9, R10

Σχήμα 2

Σχήμα 3