Οργάνωση Υπολογιστών

Αναφορά Δεύτερου Project

Ονοματεπώνυμα: Αμπλιανίτης Κωνσταντίνος

Αριθμός μητρώου: 2017030014

Προεργασία

Σκοπός του δεύτερου project ήταν η δημιουργία δύο επεξεργαστών διαφορετικού τύπου. Έναν πολλάπλών κύκλων και έναν pipelined cpu. Για την δημιουργία και των δύο επεξεργαστών σημαντική ήταν η σωστή λειτουργία του επεξεργαστή μονού κύκλου που δημιουργήσαμε στο πρώτο project, καθώς θα χρησιμοποιούσαμε πολλά από τα modules που υλοποιήθηκαν εκεί. Για την υλοποίηση του Project χρησιμοποιήθηκε το εργαλείο Xillinx ISE (version 14.7). Επίσης χρησιμοποιήθηκαν τοα εργαλεία Xillinx PlanAhead καθώς και το εργαλείο σχεδίασης DIA.

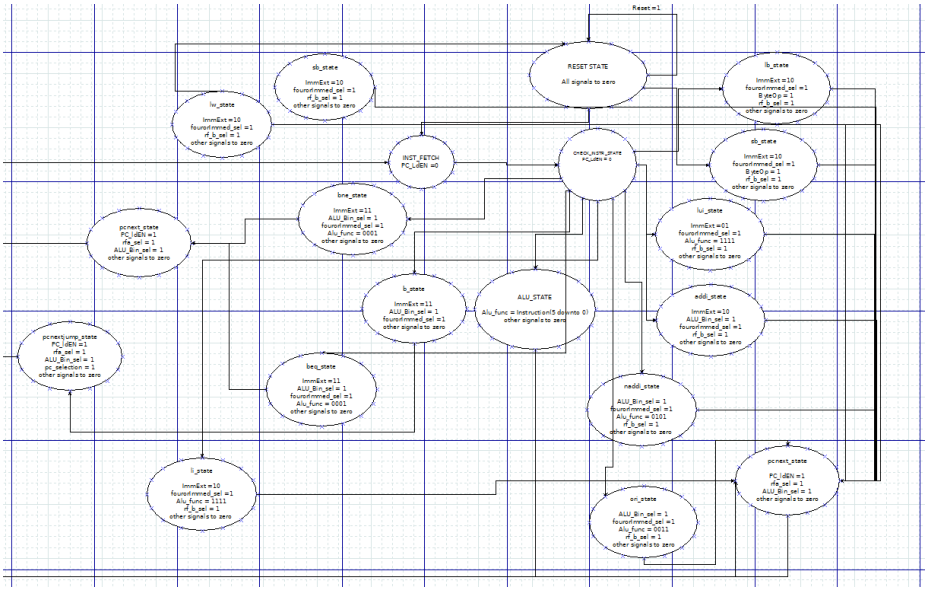
Σχεδίαση Επεξεργαστή πολλαπλών κύκλων(Multicycled Processor)

Για να υλοποιήσουμε τον επεξεργαστή πολλαπλών κύκλων προχωρήσαμε στις εξής ενέργειες: Αρχικά κρατήσαμε μόνο τον καταχωρήτη Pc από το if-stage με σκοπό να γλιτώσουμε τις καθυστερήσεις που θα έδιναν οι adder στο συγκεκριμένο κομμάτι. Έπειτα αφού περάσουμε την διεύθυνση της text μνήμης που μας δίνεται από τον pc βάζουμε έναν καταχωρητή. Χρησιμοποιούμε το ίδιο decstage που χρησιμοποιήσαμε στον επεξεργαστή μονού κύκλου με την διαφορά ότι πλέον περνάμε όλες τις εξόδους τους από καταχωρητές. Προχωρώντας στο κομμάτι ex-stage, θα προσθέσουμε δύο επιπλέον πολυπλέκτες στις εισόδους των καταχωρήτων RF_A και Immediate. Ο πολυπλέκτης που τοποθετείται στην είσοδο RF_A χρησιμοποιείται με σκοπό να μπορούμε να προχωρήσουμε τον καταχωρήτη Pc (ο πολυπλέκτης παίρνει ως είσοδους τον RF_A που προέρχεται από το Dec-Stage λαο την έξοδο του καταχωρήτη Pc). Ο δεύτερος πολυπλέκτης που τοποθετείται στην είσοδο του Immediate μπαίνει με σκοπό να γίνεται επίλογή μεταξύ Immediate και 4(ώστε να προχωράει κατά μία θέση το Pc). Επιπλέον έχει προστεθεί ένας adder και ένας ακόμα πολυπλέκτης των οποίων η χρήση είναι αποκλειστικά για το PC. Πιο αναλυτικα, στις εισόδους του πολυπλέκτη τοποθετείται το αποτέλεσμα της ALU και το (αποτέλεσμα της ALU με την προσθήκη του Immediate(b εντολές). Το αποτέλεσμα του πολυπλέκτη καταλήγει στην είσοδο του PC. Τέλος για το ex-stage περνάμε την το αποτέλεσμα της Alu από έναν καταχωρητή και προχωράμε στο mem-stage. Στο mem-stage κρατάμε το ίδιο module με αυτό που είχαμε δημιουργήσει τον επεξεργαστή μονού κύκλου, με την διαφορά ότι στο τέλος όλα τα σήματα εισόδου- εξόδου περνάνε από καταχωρητές πριν γίνει σύνδεση με το υπόλοιπο Datapath ή την μνήμη.

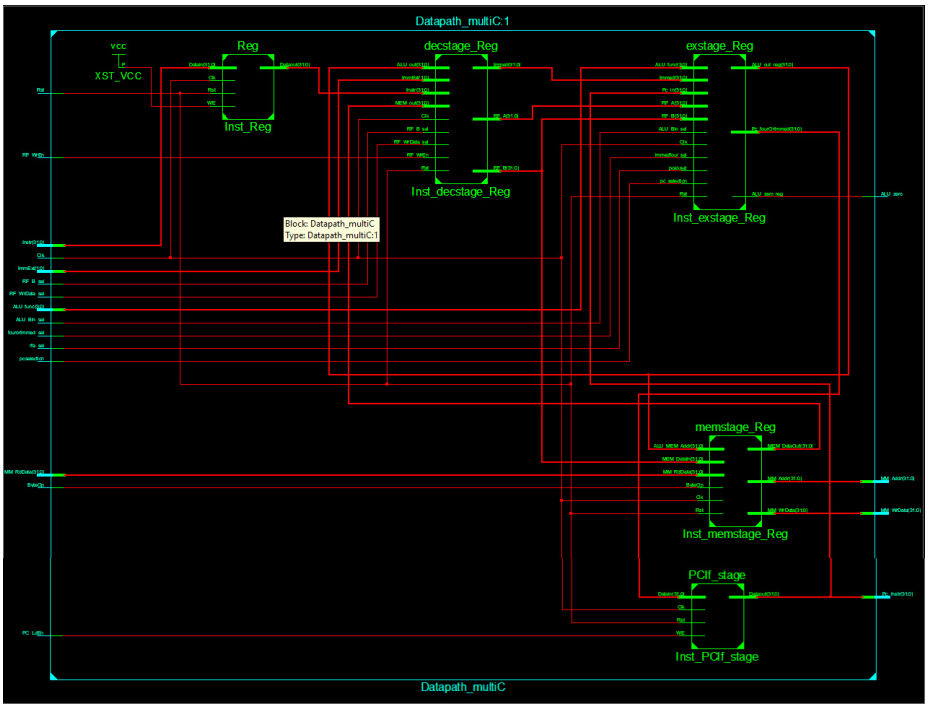
Για το χερισμό ελέγχου του επεξεργαστή πολλαπλών κύκλων υλοποιήσαμε μία Finite State Machine(FSM). Κάθε εντολή του CHARIS αποτελεί μία κατάσταση της μονάδας ελέγχου.

Επιπλέον υπάρχει η κατάσταση reset και οι καταστάσεις pnext(για να προχωρήσει το pc κατά 4 αλλά να μην γίνει εγγραφή στο αρχείο καταχωρητών η την μνήμη), pcnext(για να προχωρήσει το pc κατά 4 και να γίνει εγγραφή στο αρχείο καταχωρητών) και τέλος η pcjumpstate(ώστε να προχωρήσει το pc κατά Immediate).

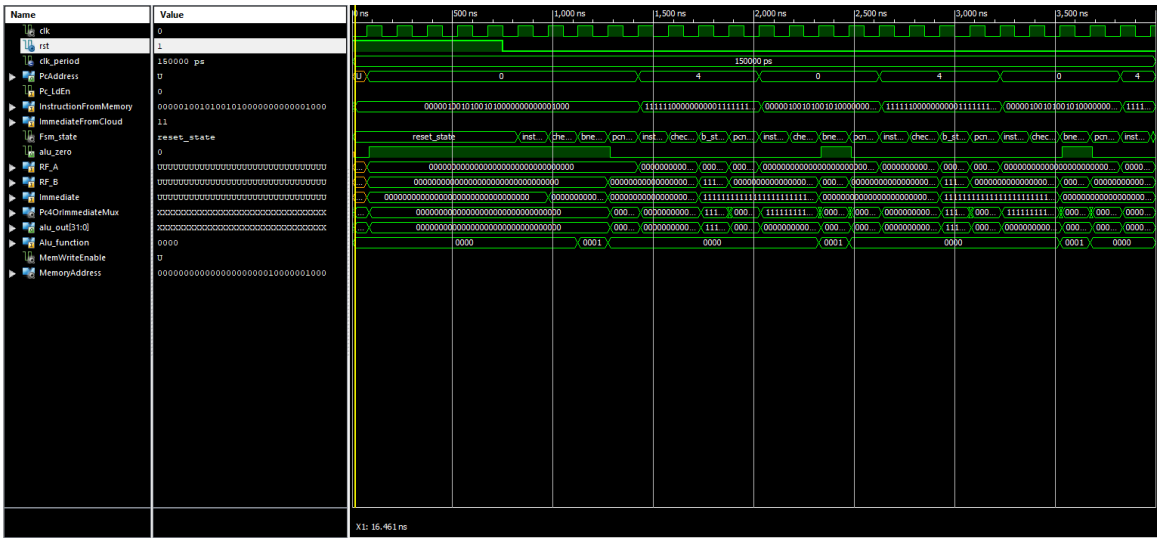
Η Fsm του Control



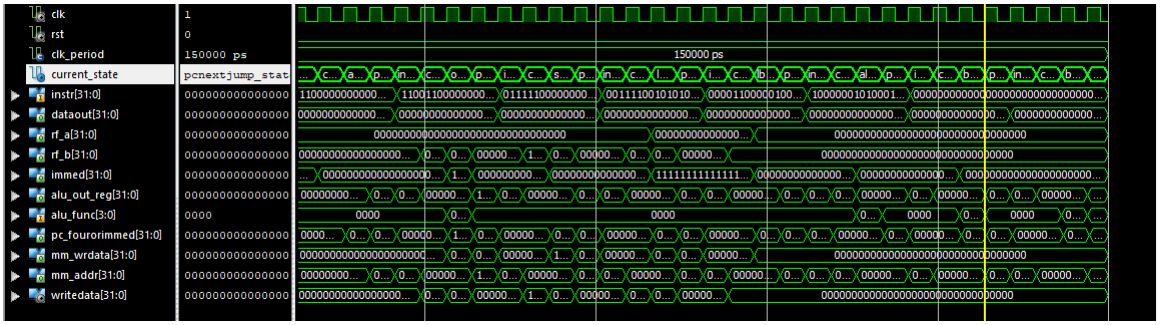
Datapath του Multicycled Επεξεργαστή



Κυμματομορφές για το πρόγραμμα αναφοράς 2



Παρατηρούμε ότι ο επεξαργαστής κινείται όπως πρέπει ανάμεσα στα states. Επίσης βλέπουμε ότι η εντολή που εισάγεται από την μνήμη αλλάζει μεταξύ δύο συγκεκριμένων (των πρώτων δύο εντολών που έχει το πρόγραμμα) και ότι το PC παίρνει τις τιμές 0,4. Άρα η τρίτη εντολή δεν εκτελείται ποτέ(όπως και θα έπρεπε), και συνεπώς παρατηρούμε σωστή λειτουργία του επεξεργαστή.

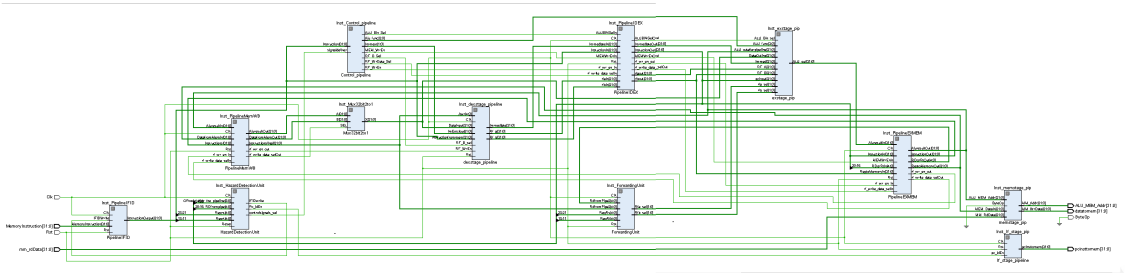


Παρατηρούμε ότι για το πρόγραμμα αναφοράς 1 ότι και εδώ γίνεται η εναλλαγή ανάμεσα στα states. Το pc μας προσθέτει κατά immediate όταν οι εντολές τύπου branch καλούνται και η φόρτωση στην μνήμη γίνεται κανονικά, όπως και στο αρχείο καταχωρήτών. Για καλύτερη ανάγνωση ανοίξαμε την καρτέλα Memory στο isim και είδαμε ότι οι τιμές των σημάτων που θέλουμε περνιούνται τόσο στην μνήμη όσο και στο αρχείο καταχρωτών, όταν είναι απαραίτητο.

Pipelined Επεξεργαστής

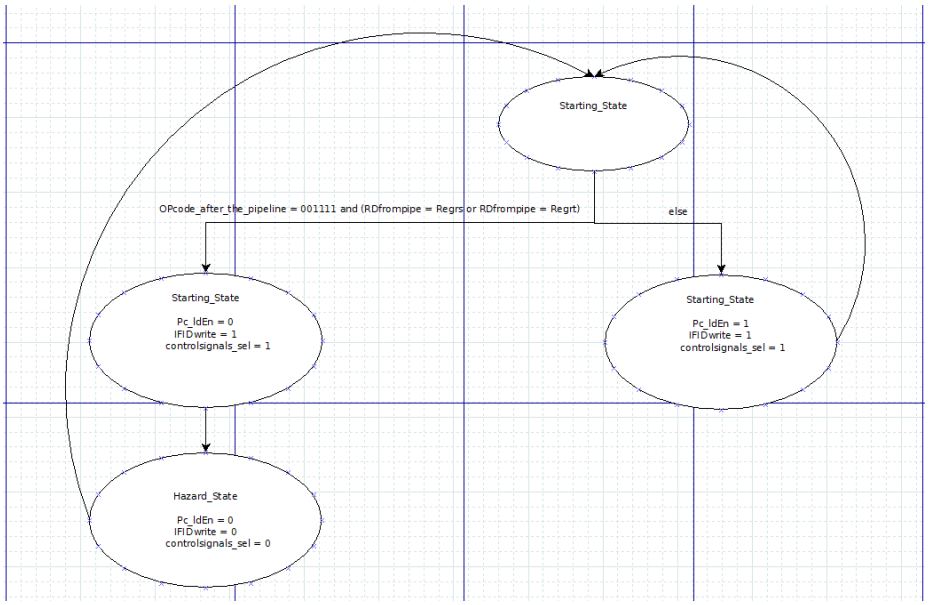
Για την υλοποίηση του pipelined επεξεργαστή κάναμε αρκετές αλλαγές στον επεξεργαστή μονού κύκλου. Αρχικά το instruction-fetch stage αντικαταστάθηκε από τον καταχωρητή PC και τον αντίστοιχο adder που προχωράει τον καταχωρητή κατά 4 ανά κύκλο ρολογιού. Η έξοδος του stage συνδέεται με την είσοδο της μνήμης η οποία, όπως και στον επεξεργαστή μονού κύκλου θα μας δώσει την αντίστοιχη εντολή από την μνήμη. Έπειτα, τοποθετούμε το pipeline IFID ο οποίος περνάει το σήμα που έρχεται από την μνήμη με σκοπό να καθυστερήσει την είσοδο της στο DECSTAGE κατά ένα κύκλο(για να τελειώσει ο πρώτος κύκλος). Στην συνέχεια αλλάξαμε ελάχιστα το DECSTAGE του επεξεργαστή μονού κύκλου (προσθέσαμε μία είσοδο η οποία δεχόταν τον rd για να γράφει σωστά στο αρχείο καταχωρητών). Επιπλέον, στο DECSTAGE προστέθηκε το Control και το Hazard Unit Detection. Το control μας υλοποιήθηκε με nested-if ώστε να πετύχουμε τον πολυπλέκτη που θα δίνει τα κατάλληλα σήματα η 0 σε όλα αυτά (ανάλογα με το Hazard Unit Detection). Το Hazard Unit Detection υλοποιήθηκε με μία FSM η οποία μεταπηδά σε κατάσταση που "κλείνει" την ολίσθιση του PC και το pipeline IFID ανάλογα με τις συνθήκες που δόθηκαν από τις διαλέξεις του μαθήματος. Προχωρώντας την διαδικασία, περνάμε από τον δεύτερο pipeline (IDEx) τους καταχωρήτες εξόδου του DECSTAGE, το Immediate καθώς και τα σήματα του κοντρολ που θα χρειαστούν για τα επόμενα στάδια του επεξεργαστή(σήματα μνήμης, σήματα που εξυπηρετούν στο EXSTAGE, καθώς και σήματα που είναι υπεύθυνα για την εγγραφή στο αρχείο καταχωρητών) προσθέτοντας έτσι άλλον ένα κύκλο καθυστέρησης. Τα σήματα που περνούν από τον παραπάνω καταχωρητή χρησιμοποιούνται για το EXSTAGE. Για να υλοποιήσουμε το EXSTAGE προσθέσαμε δύο πολυπλέκτες (3 εισόδων) στις εισόδους του EXSTAGE του επεξεργαστή μονού κύκλου. Οι κάθε ένας από τους δύο πολυπλέκτες περιελάμβανε τις εξόδους του αντίστοιχου καταχωρητή από το DECSTAGE, την έξοδο της alu που πραγματοποιήθηκε από την προηγούμενη εντολή και το αποτέλεσμα της εισαγωγής στο αρχείο καταχωρητών. Η επιλογή της εκάστοτε εισόδου γινεται από το forward unit που επίσης υλοποιήθηκε σύμφωνα με τις διαλέξεις του μαθήματός. Έπειτα, το αποτέλεσμα του EXSTAGE περνά πάλι από ένα pipeline(EXMEM) και από εκεί συνδέεται στο MEMSTAGE το οποίο δεν έχει υποστεί αλλαγές συγκριτικά με το αντίστοιχο του module μονού κύκλου. Σε περίπτωση που έχουμε είσοδο πληροφορίας από την μνήμη περνάμε το αποτέλεσμα αυτό από τον τελευταίο pipeline (MEMWB) από τον οποίο θα περάσουμε και το αποτέλεσμα της alu που έχει προκύψει ως έξοδο από το pipeline (EXMEM). Τέλος αφού έχουμε τόσο τα σήματα από την μνήμη όσο και το αποτέλεσμα της alu, μέσω ενός πολυπλέκτη αποφασίζουμε ποια θα είναι η πληροφορία εισαγωγής στο αρχείο καταχωρητών (σε αυτό το στάδιο ανοίγουμε και το rf_wren, το οποίο είναι υπεύθυνο για να επιτρέψει την εγγραφή).

Datapath του Pipelined Επεξεργαστή

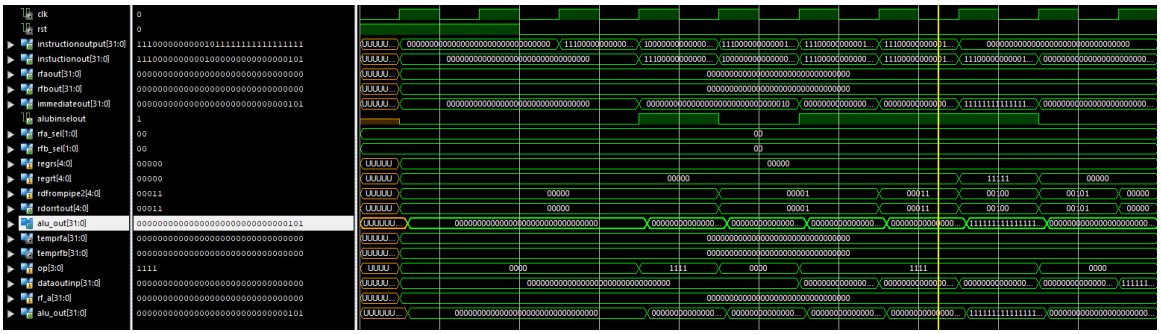


Σχήμα 1: Λόγω του ότι είναι αρκετά μεγάλο παρακαλεισθε να κάνετε zoom στην εικόνα

FSM του Hazard Unit Detection



Οι Κυματομορφές του Pipelined Επεξεργαστή



Παρατηρούμε σωστή μετακίνηση του PC και κατά συνέπεια σωστό διάβασμα των εντολών από το text memory της μνήμης. Παρατηρούμε ότι κάθε κύκλο ρολογιού έχουμε διαφορετική εντολή προς επεξεργασία σε κάθε stage όπως και θα έπρεπε. Επιπλέον, βλέπουμε σωστή φόρτωση των τιμών από το forward unit για τις τιμές li. Βλέπουμε σωστή φόρτωση των τιμών που θέλουμε στο αρχείο καταχωρητών, στον σωστό κύκλο ρολογιού. Παρατηρούμε ότι δεν έχουμε αλλάζουν τα rf_a_sel, rf_b_sel στην πρώτη εντολή καθώς δεν χρησιμοποιούμε τον καταχωρητή προορισμού από την προηγούμενη εντολή η την την εντολή πριν από αυτή. Συνολικά βλέπουμε ότι κάθε βαθμίδα παίρνει τιμές με διαφορά χρόνου ενός κύκλου από την άλλη αλλά σε κάθε κύκλο, κάθε βαθμίδα τρέχει κομμάτι διαφορετικής εντολής που είναι και η ουσία το pipelined cpu.