Οργάνωση Υπολογιστών

Αναφορά Πρώτου Project

Ονοματεπώνυμα: Αμπλιανίτης Κωνσταντίνος, Κυρίτσης Βασίλης

Αριθμός μητρώου: 2017030014, 2017030037

Προεργασία

Σκοπός του εργαστηρίου ήταν η δημιουργία ένος επεξεργαστή μόνου κύκλου σύμφωνα με την σταδιακή περιγραφή που μας δώθηκε στην εκφώνηση του εργαστηρίου. Σημαντική για το εργαστήριο ήταν η κατανόηση των λειτουργιών των επιμέρους compartments καθώς και τη μελέτη των εντολών (CHARIS). Για την υλοποίηση του πρότζεκτ χρησιμοποιήθηκε το Xillinx ISE (version 14.3) καθώς και το πρόγραμμα Dia για την δημιουργία των block diagramms.

Σχεδίαση μονάδας αριθμιτικών-λογικών πράξεων

Η μονάδα αριθμιτικών-λογικών πράξεων (ALU) η οποία σχεδιάσαμε, αποτελείται από δύο εισόδους A, B (θεωρούνται ότι είναι αριθμοί 32bit σε συμπλήρωμα ως προς 2), μία είσοδο τεσσάρων bit (Op) η οποία θα καθορίζει την πράξη που θα γίνεται και δίνει μία έξοδο 32bit ((Outp)) η οποία θα μας εμφανίζει το αποτέλεσμα. Επίσης θα υπάρχουν τα σήματα Ovf(σήμα υπερχείλισης), Ovf(υπόδειξη κρατουμένου σε περίπτωση υπερχείλισης) και Zeroout(σήμα υπόδειξης ότι το αποτέλεσμά μας δίνει μηδέν).

Για την υλοποίηση των πράξεων χρησιμοποιήσαμε την βιβλιοθήκη ieee numeric std ώστε να μην καταφύγουμε σε υλοποίηση περεταίρω κυκλωμάτων(π.χ Full Adders).Μέσω ενός if statement, ανάλογα με το σήμα Op διαλέγουμε την κατάλληλη πράξη που πρέπει να γίνει. Για να επιτύχουμε να παίρνουμε κάθε φορά σωστό αποτέλεσμα για όσον αναφορά την υπερχείλιση απλά κάνουμε την πράξη με 33 bits όπου το MSB και των δύο εισόδων θα είναι μηδεν (στο τέλος το κρατάμε το 33^o ως Ovf και συνεπώς και ως Cout).

Σχεδίαση Καταχωρητή

Ο καταχωρητής που έχουμε σχεδιάσει αποτελείται από τα σήματα εισόδουReset, Clk,WE, Datain και έχει ως μοναδική έξοδο το Dataout. Ο καταχωρητής "γράφει" μόνο όταν το WE είναι ενεργοποιημένο και το Reset απενεργοποιημένο. Αρχικοποιείται στο μηδέν (μέσω της ενεργοποίησης του Reset). Όταν το WE δεν είναι ενεργοποιημένο (όπως και το Reset), ο καταχωρητής κρατάει την παλιά του τιμή.

Σχεδίαση Αρχείου Καταχωρητών Register-File(RF)

Για να δημιουργήσουμε το συγκεκριμένο κομμάτι, χρειαστήκαμε 32 καταχωρητές. Οι είσοδοι που έχει το αρχείο καταχωρητών είναι δύο Adr (καθορίζουν τα αποτελέσματα των δύο εξόδων Dout1, Dout2 μέσω των τελευταίων πολυπλέκτων), το Awr(καθορίζει την πιθανή διεύθυνση καταχωρητή πουενδέχεται να γίνει εγγραφή), το WE (καθορίζει αν θα γίνει εγγραφή στον καταχωρητή της διεύθυνσης που καθορίζεται από τον Awr) και το Din καθορίζει την τιμή εγγραφής. Για να μπορέσουμε στους δύο τελευταίους πολυπλέκτες να καταφέρουμε να έχουμε 32 εισόδους καταχωρητών(δηλαδή 32 εισόδους 32 bit), δημιουργήσαμε ένα package(arraypackage) με σκοπό να φτιάξουμε έναν πίνακα 32x32 ο οποίος θα προσομοίωνε το RF.

Υλοποίηση Κύριας μνήμης

Χρησιμοποιώντας τον κώδικα που μας δώθηκε από την εκφώνηση, δημιουργήσαμε ένα module μνήμης 2048 θέσεων. Η μνήμη φορτώνει από ένα αρχείο .data όπου περιέχει τις εντολές του εκάστοτε προγράμματος. Έχει δύο εισόδους 11 bit που είναι υπεύθυνες για να δίνουν την διεύθυνση, είτε στο text stage είτε στο data stage.To datawe το οποίο μας δείχνει αν θα γράψουμε στο data stage η όχι, την είσοδο δεδομένων στην μνήμη DataDin και τέλος υπάρχει η έξοδος δεδομένων από την μνήμη που είναι υπεύθυνη για την μεταφορά δεδομένων από την μνήμη στο αρχείο καταχωρητών.

Υλοποίηση IF-STAGE

Η βαθμίδα του if-stage αποτελείται από έναν καταχωρητή που η δουλεία του είναι να στέλνει την θέση της εντολής έτσι ώστε να γίνεται ανάκλιση από την μνήμη (text stage). Ως είσοδος του καταχωρητή είναι είτε η προηγούμενη τιμή του συν 4 είτε η προηγούμενη τιμή του συν 4 συν το Immediate(χρησιμοποιείται για εντολές συνθηκών η για εντολή b- jump). Η τιμή εισόδου καθορίζεται από το σήμα Pc-sel.

Υλοποίηση DEC-STAGE

Η βαθμίδα του dec-stage περιέχει το αρχείο καταχωρητών. Παίρνει ως είσοδο του Instruction που έρχεται από την μνήμη (προέρχεται από το ifstage $\rightarrow text - stage \rightarrow instruction$) και τοποθετεί διαφορετικά τμήματα του μέσα στην υλοποίηση του αρχείου καταχωρητών, δίνοντας κατευθυντήριες οδηγίες για το ποιούς καταχωρητές θα διαβάσει (η θα γράψει, ανάλογα με το write enable). Επιπλέον, περιέχεται το module του συννέφου που δέχεται ένα κομμάτι του instruction(τα 16 τελευταία bit) και τα επεξεργάζεται ώστε να δημιουργηθεί το immediate (Να κάνει τα 16 bit 32 με μία από τις παρακάτω επιλογές: zerofill, shift left and zero fill,

sign extend, shift left 2 bit and sign extend). Ακόμα, μέσω ενός πολυπλέκτη επιλέγει αν θα γράψει στο αρχείο καταχωρητών το αποτέλεσμα που προκύπτει από το στάδιο της ALU η θα φορτώσει κάποια τιμή από τη μνήμη. Ώς έξοδο μας δίνει τις τιμές των δύο καταχωτητών που αναγράφονται στο instruction(ο δεύτερος καταχωρητής επιλέγεται από το rfb sel που καθορίζει ποια bit του instruction θα δωθούν ως θέση στο RF) με σκοπό να μπορέσουμε να κάνουμε την πράξη της επιλογής μας στο παρακάτω στάδιο που είναι το στάδιο της ALU (EX-STAGE).

Υλοποίηση EX-STAGE

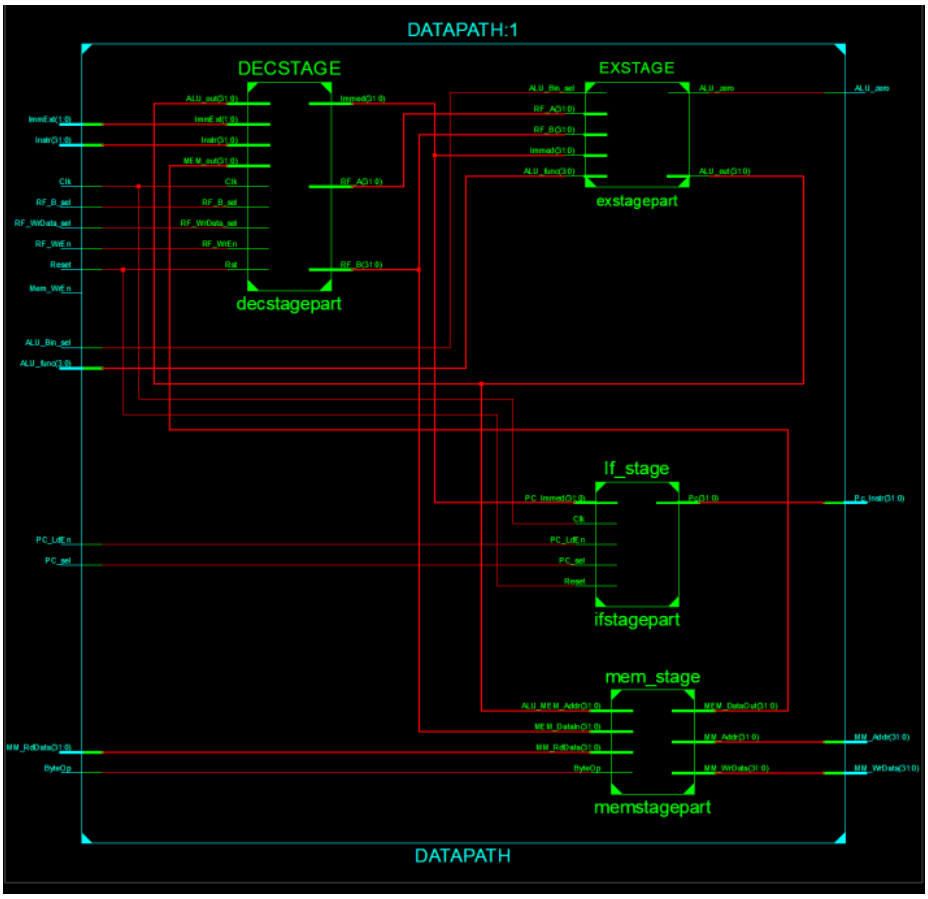
Η βαθμίδα του EX-stage ή ALU-stage αποτελείται κυρίως από την ALU. Σαν πρώτη είσοδο της παίρνουμε τον καταχωρήτη που αντιστοιχεί στο rs του instruction(Έρχεται από το dec-stage ως RF-A). Ως δεύτερη είσοδος, επιλέγεται μέσω ενός πολυπλέκτη αν θα εισαχθεί ο καταχωρητής RF-B(DEC-STAGE), ή το Immediate που έρχεται από το σύννεφο. Τέλος μέσω της ALU παράγεται ένα αποτέλεσμα που συνδέεται στον πολυπλέκτη επιλογής αποθήκευσης στο RF. Επίσης, χρησιμοποιείται για να καθορίσει την διεύθυνση πιθανής αποθήκευσης των δεδομένων ενός καταχωρητή στην μνήμη.

Υλοποίηση MEM-STAGE

Η βαθμίδα του MEM-stage είναι υπεύθυνση για την δρομολόγιση των δεδομένων από την RAM στο αρχείο καταχωρητών και τούμπαλιν. Μέσω μιας πράξης λογικού and με ένα σήμα συγκεκριμένης τιμής(σήμα που έχει τα τελευταία 8 bit ενεργοποιημένα και τα υπόλοιπα στο 0), κάθε φορά που το ByteOp είναι ενεργοποιημένο αποθηκεύει στην μνήμη (η φορτώνει από αυτήν στο αρχείο καταχωρητών - καθορίζεται από το mem-write-en) μόνο τα τελευταία 8 bit που έρχονται ως είσοδος για εγγραφή στον καταχωρητή (η ως έξοδος από αυτόν για αποθήκευση). Αλλιώς αποθηκεύει στην μνήμη (ή φορτώνει από αυτήν στο αρχείο καταχωρητών) ολόκληρη την πληροφορία.

Υλοποίηση DATAPATH

Υλοποιείται μέσω ενός portmap που συνδέει όλα τα stages σε ένα. Ο τρόπος σύνδεσης φαίνεται στο παρακάτω block diagramm.

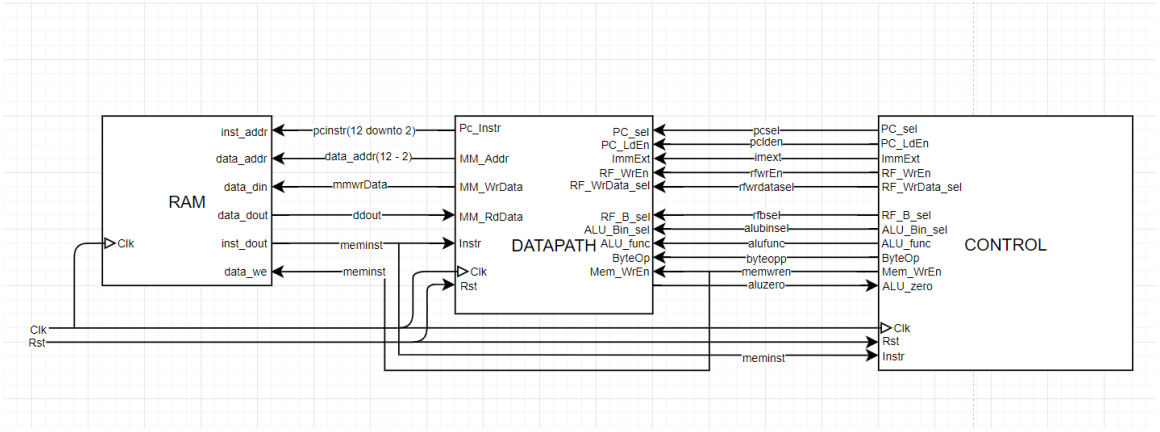


Υλοποίηση CONTROLSTAGE

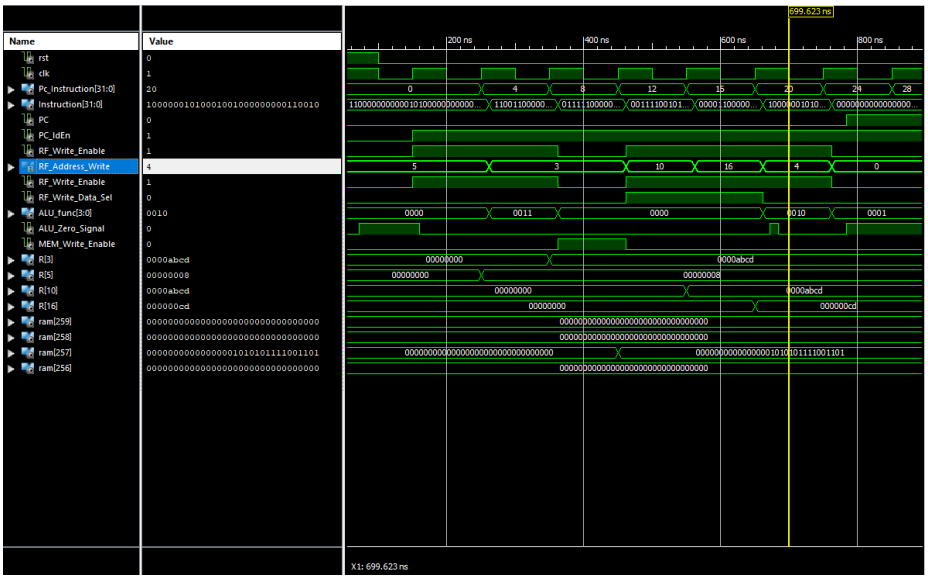
Το control αποτελείται από μία fsm δύο καταστάσεων (καταστάσεις reset,chec) όπου ανάλογα με το instruction που μας δίνεται, παράγει τα αντίστοιχα σήματα ελέγχου ώστε το datapath να μπορέσει να κάνει την σωστή λειτουργία που του ανατέθηκε.

Υλοποίηση PROC-SC

Είναι το τελικό στάδιο και είναι ένα portmap που συνδέει datapath, control και μνήμη όπως φαίνεται στο παρακάτω σχήμα.



Τα αποτελέσματα από τα προγράμματα αναφορών παρουσιάζονται παρακάτω:
Πρόγραμμα αναφοράς 1:



Πρόγραμμα αναφοράς 2:

Περιγραφή αποτελεσμάτων:
Γενικά παρατηρούμε ότι όλες οι εντολές γίνονται με μία καθυστέρηση όπως αναμέναμε, (λόγω της χρήσης της εντολής after με σκοπό την προσομοίωση καθυστέρησης των κυκλωμάτων) αλλά κάθε καινούρια εντολή εκτελείται κανονικά πριν το τέλος του αντίστοιχου κύκλου. Παρατηρούμε εγγραφή στη μνήμη όταν το αντίστοιχο Write enable είναι ενεργοποιημένο. Επιπλέον παρατηρούμε την σωστή λειτουργία του ByteOp αφού βλέπουμε ότι φορτώνει μόνο τα τελευταία 8 bit της λέξης όταν το καλούμε(τα πρώτα 24 έχουν φορτώσει στο 0). Τέλος, αφού η μνήμη μας είναι οργανωμένη σε λέξεις, παρατηρούμε σωστή διεύθυνση εγγραφής αφού βλέπουμε ότι κάνοντας store στην πρώτη διεύθυνση του data stage η διεύθυνση μνήμης που 'γράφει' είναι σωστά η 256 ($1024/4 = 256$).
Για το πρόγραμμα 2 επιβεβαιώνουμε την ατέρμονη επαναληπτική διαδικασία κοιτώντας το αποτέλεσμα του καταχωρητή Pc. Παρατηρούμε ότι αφού ελέγξει ότι η συνθήκη δεν ισχύει (τιμή το PC $= 0$ άρα πρώτη εντολή) προχωράει στην ακριβώς επόμενη (θέση 4) και στην συνέχεια, αφού η επόμενη εντολή ξαναστέλνει τον PC στο 0, σημαίνει ότι επαναλαμβάνει την διαδικασία. Συνεπώς καταλήγουμε σε μία ατέρμονη επαναληπτική διαδικασία (όπως έπρεπε) και η τρίτη εντολή του προγράμματος δεν εκτελείται ποτέ.