ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ

Εργασία 1:Επεξεργαστής ενός κύκλου

Μιχαήλ Μαρκετάκης

2017030165

1η ΦΑΣΗ ΤΗΣ ΕΡΓΑΣΙΑΣ

Α) Μονάδα αριθμητικών και λογικών πράξεων (Arithmetic Logic Unit ή ALU)

Το πρώτο μέρος της εργασίας ήταν η δημιουργία της ALU,ενός συνδυαστικού κυκλώματος που δεν έχει ρολόι.

Η ALU παίρνει σαν είσοδο έναν 4-bit αριθμό**(Op)** όπου αντιστοιχεί στον κωδικό της πράξης,καθώς επίσης και τους δύο 32-bit αριθμούς**(A,B)** με τους οποίους θα γίνει η αντίστοιχη πράξη.

Το αποτέλεσμα της πράξης αποθηκεύεται στην εξοδο **ALUout** (32-bit).

Στις πράξεις της ALU αυτό που είχε μεγάλο ενδιαφέρον είναι οι περιπτώσεις όπου είχαμε **OverFlow(ovf)** ,**Carry Out(cout)** ή συνδιασμό αυτών.

Πιο συγκεκρίμενα,Το σήμα του OverFlow άναβε(=1) στις εξής περιπτώσεις:

1)Στην πρόσθεση

Όταν αθροίζαμε δύο θετικούς αριθμούς και το αποτέλεσμα ήταν αρνητικό (32-bit =>1).

Όταν αθροίζαμε δύο αρνητικούς αριθμούς και το αποτέλεσμα ήταν θετικός (32-bit=>0).

2) Στην αφαίρεση

Όταν από έναν θετικό αριθμό αφαιρούσαμε έναν αρνητικό και το αποτέλεσμα της πράξης ήταν αρνητικός(32-bit =>1).

Όταν από έναν αρνητικό αφαιρούσαμε έναν θετικό και το αποτέλεσμα της πράξης ήταν θετικός(32-bit=>0).

Όσο αφορά το σημα για το κρατούμενο μας (cout) για τον χειρισμό του χρησιμοποιήσαμε ένα εσωτερικό σημα 33-bit οπου το MSB θα ήταν ακριβώς αυτό το κρατόυμενο στις περιπτώσεις όπου υπήρχε.

Όπως και στην περίπτωση του Overflow,έτσι και τώρα την περίπτωση του κρατουμένου, την συναντήσαμε μόνο στις πράξεις της πρόσθεσης και της αφαίρεσης.Πιο συνοπτικά,σε πράξη πρόσθεσης ή αφαίρεσης όπου το αποτέλεσμα της πράξης ήταν μεγαλύτερο των 32-bits τότε θα άναβε(=1) το σήμα του κρατουμένου.

Σε κάθε άλλη περίπτωση πράξης,Το σήμα OverFlow(ovl) και Carry Out(cout) ήταν ίσα με μηδέν.

‘Αλλο ένα σήμα όπου έκανε την <<διαφορά>> στο κομμάτι της ALU ήταν το σήμα **zero**,το οποίο άναβε(=1) Μόνο στην περίπτωση όπου το αποτέλεσμα της πράξης ήταν ίσο με Μηδέν.

Τέλος μια σημαντική παρατήρηση θα ήταν ότι η έξοδος της ALU προκύπτει κάθε φορά με μια καθυστέρηση 10 ns μετά από την είσοδο με την χρήση του after.

Β) Αρχείο καταχωρητών (Register File ή RF)

Β1. Δημιουργία καταχωρητή

Αρχικά ξεκινήσαμε την υλοποίηση φτιάχνοντας έναν απλό register 32-bit,ο οποίος είχε μία είσοδο **Datain(32-bit)** μία έξοδο **Dataout(32-bit)** καθώς επίσης τα σήματα **Clock{CLK(1-bit)}**,**Reset{RST(1-bit)}**,**Write Enable{WE(1-bit)}**.

Ο register μας λειτουργούσε ως εξής, Κάθε φορά όπου είχαμε RST=0 και WE=1,τότε η είσοδος Datain έβγαινε στην έξοδο 10 ns μετα από τον θετικό παλμό του ρολογιού(υλοποιήθηκε με την χρήση του after της VHDL όπως και στην ALU)

Στην περίπτωση όπου είχαμε RST=1 τοτε η έξοδος ηταν υποχρωτικά ίση με Μηδέν.

Σε κάθε άλλη περίπτωση (πχ RST=0 και WR=0) η έξοδος θα ήταν ίση με Μηδέν.

Β2. Δημιουγία αρχείου καταχωρητών RF

Στην συνέχεια του Πρότζεκτ,υλοποιήσαμε το αρχείο καταχωρητών(Register File), το οποίο αποτελούνταν από 3 τμήματα:

1) 32 καταχωρητές 32-bit.

2) Έναν αποκωδικοποιητή

3) Δύο πολυπλέκτες 32 προς 1

Αρχικά η είσοδος εγγραφής **Awr(5_bit)** έστελνε στον αποκωδικοποιητή σήμα το οποίο θα ενεργοποιούσε λειτουργία καταχώρησης για έναν μόνο Register την κάθε φόρα.Τα δεδομένα προς Εγγραφή έφταναν στον αντίστοιχο Register από την είσοδο **Din(32-bit)**.

Για την υλοποίηση των 31 καταχωρητών έγινε χρήση του for-generate της VHDL καθως επισης ο καταχωρητής μηδέν αρχικοποιήθηκε να εχει πάντα μηδενική έξοδο.

Τέλος οι 32 έξοδοι(32 bit η κάθε μία) των καταχωρητών πήγαιναν σε δύο πολυπλέκτες 32 προς 1 ο κάθε ενας από αυτους.

Για την δημιουργία πολυπλέκτη με 32(32-μπιτες) εισοδους ο κάθε ένας Χρησιμοποιήθηκε ένα πακέτο όπου έγινε η δημιουργία ενός **MadeType** array 32 θέσεων,με 32 bit η κάθε θέση.

Τα σήματα **Ard1(5-bit)** και **Ard2(5-bit)** ήταν τα σήματα επιλογής για τον κάθε πολυπλέκτη αντιστοιχα, όπου με βάση την τιμή τους(0-31 στο δεκαδικό) ο κάθε πολυπλέκτης έβγαζε ως έξοδο την τιμή του αντίστοιχου Register(0-31).

Οι έξοδοι των πολυπλεκτών από το RF ήταν οι **Dout1(32-bit)** και **Dout2(32-bit)** για τον πολυπλεκτη 1 και 2 αντίστοιχα,και όπως υπώθηκε παραπάνω είναι οι τίμες όπου διάβασε ο κάθε πολυπλέκτης από κάποιον register σύμφωνα με την τιμή του αντίστοιχου σήματος Ard

2η ΦΑΣΗ ΤΗΣ ΕΡΓΑΣΙΑΣ

Βαθμίδα ανάκλησης εντολών (IF)

Η βαθμια IFSTAGE ,υλοποιήθηκε με την χρήση του Register που φτιάξαμε παραπάνω ο οποίος αντιπροσωπεύει τον καταχωρητή PC(32 bit).Επιπλέον υλοποιήσαμε έναν 2 αθροιστες,και έναν πολυπλέκτη 2 προς 1.

Ο πρώτος αθροιστής αθροίζει κάθε φορα την έξοδο του καταχωρητή PC (32-bit) με το 4(PC+4).

Ο δεύτερος αθροιστης αθροιζει την έξοδο του πρώτου αθροιστή μαζι με μια άλλη είσοδο της βαθμίδας IF,το σήμα **PC_Immed** {(PC+4) + SignExt(Immed)*4}

Τέλος οι έξοδοι από τους δύο αθροιστές φτάνουν σε έναν πολυπλέκτη 2 προς 1,όπου αναλόγως την τιμή του σήματος επιλογής(**PC_Sel**) 0 ή 1 περνάει στην έξοδο η είσοδος από τον πρώτο αθροιστή ή από τον δεύτερο αντίστοιχα.

Η έξοδος του πολυπλέκτη θα καταλήξει στον καταχωρητή PC και πάλι ο οποιος με καταλληλες τιμες στο **PC_LdEn**(=1) και **Reset**=0 θα δωσει την έξοδο του στην μνήμη,η οποία αποτελεί την διεύθυνση από την οποία η μνήμη θα διαβάσει την αντίστοιχη εντολή.

Για ολες τις εντολες ο PC αυξανεται +4 οποτε η μνήμη πηγαίνει από την 1η στην 2η στην 3η θέση κλπ..

Για τις Εντολές (b beq bne) ο PC αυξάνεται +4 + SignExt(Immed)*4 οποτε η μνημη πηγαίνει στην αντίστοιχη διεύθυνση.Υπενθυμίζουμε ότι η είσοδος της μνήμης από τα 32 bit της εξόδου του καταχωρητή PC θα παρει ως διεύθυνση τα 12 downto 2 bits(διευθυνση της μνήμης),και θα διαβασει από το text segment (θέσεις 0-1024) την αντίστοιχη εντολή (32-bit) όπου θα την βγάλει ως έξοδο(**Instruction**).

Βαθμίδα αποκωδικοποίησης εντολών (DECODE)

Στην συνέχεια της 2η φασης του Πρότζεκτ,έγινε η δημιουργία του DECSTAGE,

Το οποίο αποτελούνταν από:

Το Αρχείο καταχωρητών (Register File),2 πολυπλεκτες καθως επισης και το συννεφακι (**cloud**)

Συγκεκριμενα το DECSTAGE αποκωδικοποιούσε την εντολη όπου εβγαινε από την μνήμη του IFSTAGE(**Instruction**).

Έτσι λοιπόν ο πρώτος καταχωρητής που θα διαβασει(εννοείται η τιμή του) ο πρώτος πολυπλέκτης του RF αντιστοιχεί στο Instr(25-21) και ήταν ο rs,όπου αντιστοιχούσε στο Ard1 του RF,και η εξοδος του DECSTAGE είναι το σημα **RF_A(32-bit)**

Ο δεύτερος αντιστοιχεί στο Instr(15-11) και είναι ο rt όπου αντιστοιχούσε στο Ard2 του RF.

Το Awr του RF αντιστοιχίζεται με το Instr(20-16) όπου είναι ο Rdestination,δηλαδή η διεύθυνση εγγραφής.

Τα σηματα Instr(15-11) και Instr(20-16) πήγαιναν σε έναν πολυπλέκτη όπου αναλόγως την τιμη του σήματος επιλογης **RF_B_sel**(0 ή 1) πέρναγε το αντίστοιχο σήμα.Για τις τύπου R εντολές πέρναγε το Instr(15-11) δηλαδη ο rt ενώ για τις τύπου i εντολες πέρναγε το Instr(20-16) δηλαδη ο Rd,αρα αναλόγως με το τι θα περάσει στην έξοδο του πολυπλέκτη έχουμε στην έξοδο του DECSTAGE το σήμα **RF_B(32-bit)**

Τέλος το Instr(15-0) πήγαινε ως είσοδο στο λεγόμενο συννεφάκι(**Cloud**) στο οποιο ορίσαμε τις 4 εντολες όπου παίρνουν χώρο με τα 2 bit ελέγχου **CloudOp(00,01,10,11)** για κάθε μία από τις περιπτώσεις αντίστοιχα:

SignedExtend σε 32 bit--> 00

Zerofilling σε 32 bit -->01

Zerofilling σε 32 bit (shift left 16)-->10

SignedExtend σε 32 (shift left 2)-->11

Η έξοδος από το συννεφάκι είναι το σήμα **Immed**(32-bit).(Αντιστοιχίζεται με το σήμα Pc Immed του IFSTAGE)

Τέλος στο DECSTAGE υπάρχει και ένας δεύτερος πολυπλέκτης με δύο σήματα εισόδου **ALU_out**,**MEM_out**(32 bit το κάθε ένα) ,ο οποίος αναλόγως την τιμη του σήματος επιλογής **RF_WrData_sel** καθοριζει αν η εξοδος του πολυπλέκτη η οποία συνδέεται με το Din του RF(δηλαδή τα δεδομένα προς εγγραφή) θα προέρχονται από την ALU ή από την μνήμη(εντολές lw,lb).

Βαθμίδα εκτέλεσης εντολών (EX)

Οι έξοδοι από το DECSTAGE (RF_A,RF_B,Immed)(32-bit) η κάθε μια από αυτές πηγαίνουν στην βαθμίδα εκτέλεσης εντολών EXSTAGE.

Πιο Συγκεκριμένα το EXSTAGE αποτελείται από την ALU καθώς επίσης και από έναν πολυπλέκτη.

Ο πολυπλέκτης παίρνει ως είσοδο το σημα **RF_B** και το σήμα I**mmed** και αναλόγως την τιμη του **ALU_Bin_Sel**(0 ή 1) περνάει το RF_B ή το Immed αντίστοιχα(R type ή i type εντολή αντίστοιχα)

Τελικά το σήμα που θα περάσει αποτελεί μαζι με το σημα RF_A τις δύο (32-bit) εισόδους της ALU,όπου θα γίνει η αντίστοιχη πράξη σύμφωνα με το **ALU_func**

Η εξοδος της ALU θα είναι το σημα ALU_Out(32-bit)

Βαθμίδα πρόσβασης μνήμης (MEM)

Η βαθμίδα MEM μας εξυπηρετεί για τις εντολες store και load προς και από την μνημη

Ο έλεγχος για τις εντολες sw,sb,lw,lb γινεται με βαση τα σήματα ByteOp, Mem_WrEn και MM_WrEn (1 –bit),

Η εξοδος της ALU από το EXSTAGE γίνεται είσοδος της βαθμίδας MEM,η οποία συνδέεται με την Μνήμη,άρα το αποτέλεσμα της ALU θΑ είναι η διεύθυνση στην οποία θα κάνουμε (sw,sb,lw,lb),δηλαδή το σήμα **ALU_MEM_Addr**.Φυσικά η μνήμη όπως ειπώθηκε και παραπάνω παίρνει σαν είσοδο τα 12 downto 2 bits του σηματος και σε αυτό γίνεται πρόσθεση του 1024 ώστε οι εντολες STORE,LOAD να γίνονται στις θεσεις 1024-2028.

Επίσης η βαθμίδα MEM έχει ως είσοδο το σημα RF_B από την βαθμίδα DECSTAGE,όπου επειδή έχουμε εντολή τύπου i θα ισχύει ότι RF_B_Sel=1,

Αρα η έξοδος RF_B θα είναι η τιμή του Rdestination όπου θα αποτελεί το σήμα προς εγγραφή στην μνήμη(sw,sb),**MEM DataIn-->MM_WrData**

Για τα δεδόμενα που διαβάζει η Μνήμη MM_RdData(lw,lb) από την καταλληλη διευθυνση που εχει λαβει αυτά βγαίνουν έξω από την μνήμη από την έξοδο MEM_DataOut<------ MM_RdData.

Η έξοδος αυτή πηγαίνει στην Βαθμίδα DEC όπου αποτελεί την είσοδο του πολυπλέκτη (**MEM_out)** για τα δεδομένα προς εγγραφή.

DATAPATH

Γίνεται σύνδεση όλων των βαθμίδων,IF,DEC,EX,MEM.

Στο τελικο module DATAPATH ως είσοδο δίνουμε όλα τα σήματα επιλογής των πολυπλεκτών που έχουμε στις παραπάνω βαθμίδες,καθώς και για τις περιπτώσεις του Cloud ,αλλα και τα σήματα επιλογής ByteOp,Mem_WrEn για τις εντολές store και load προς και από την μνήμη.Ως έξοδο έχουμε τα σήματα OverFlow,Cout,Zero από την ALU(EXSTAGE) καθως επίσης και το Instruction(από το IFSTAGE) δηλαδή την εντολή που διαβάστηκε από το text segment της μνήμης,η οποία θα χρειαστεί για το CONTROL.Τέλος ορίζουμε ένα

Σήμα BBFLAG(1 –bit) το οποίο ενεργοποιείται όταν οι έξοδοι του DEC (RF_A ΚΑΙ RF_B) είναι ίσες. (για τις εντολες beq,bne)

Στο Test_bench του datapath υλοποιούμε το προγραμμα αναφοράς 1.

CONTROL

Στο control γίνεται αυτοματοποίση των εντολών,χωρίς να χρειάζεται πλέον να δίνουμε τιμές σε όλα τα σήματα επιλογής.

Αυτό συμβαίνει διακρίνοντας περιπτώσεις με βάση το OPCODE κάθε εντολής .

Όλα τα σήματα πλέον λοιπόν που δίναμε στο datapath είναι αρχικοποιημένα με βάση το opcode κάθε εντολής,και αποτελούν εξόδους του control.

Οι μόνες είσοδοι του control είναι το instruction,ώστε να αποκωδικοποιήσουμε το opcode(opcode=Instruction(5 downto 0)) και το func για την πραξη που θα γίνει,καθώς επίσης και το σήμα BBFlag που ορίσαμε στο datapath(beq,bne)

PROC_SC

Πραγματοποιείται σύνδεση των βαθμίδων Control και Datapath

Αποτελεί την τελική βαθμίδα του Επεξεργαστή μονού κύκλου.

Εκτελείται το πρόγραμμα αναφοράς 2.