ΕΡΓΑΣΙΑ – ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ HW 1

ΕΙΡΗΝΗ ΣΜΥΡΝΑΚΗ ΑΕΜ: 10839

ΤΗΜΜΥ ΑΠΘ

ΑΣΚΗΣΗ 1

Όρισα τις εισόδους (op1,op2,alu_op) και εξόδους (zero,result) του module alu βάση της εκφώνησης. Στη συνέχεια χρησιμοποίησα παραμέτρους για να δημιουργήσω σταθερές για κάθε μια από τις εννιά εντολές λειτουργίας της ALU. Έπειτα πρόσθεσα ένα always μπλοκ και μέσα σε αυτό έβαλα μια case εντολή για να αναγνωρίζει το alu_op και να εκτελεί κάθε φορά την αντίστοιχη λειτουργία. Σε κάθε περίπτωση του case υλοποίησα την αντίστοιχη λειτουργία ώστε το result να παίρνει το αναμενόμενο αποτέλεσμα, όπως περιγράφεται στην εκφώνηση. Και τέλος αν το result είναι 0 όρισα το zero να γίνεται 1 διαφορετικά να γίνεται 0.

ΑΣΚΗΣΗ 2

Για το calc_enc:

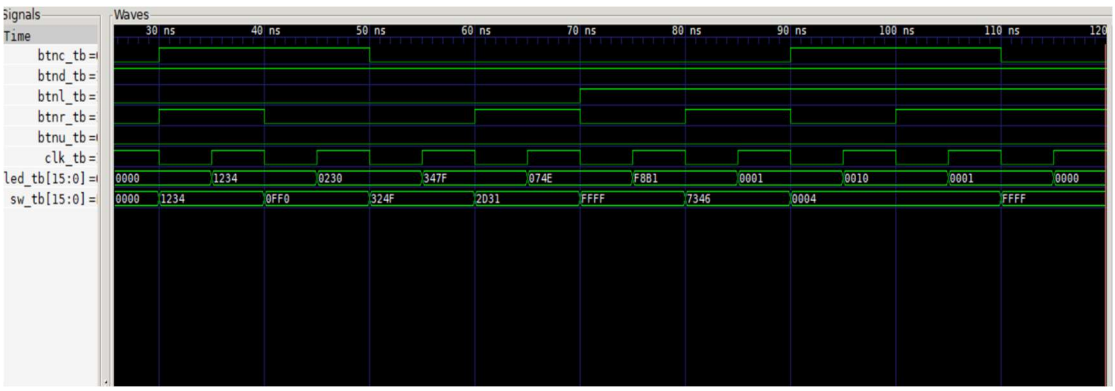
Όρισα τις εισόδους (btnr,btnl,btnc) και την έξοδο(alu_op) του module calc_enc βάση τις ανάγκες του και μετά υλοποίησα μέσω των λογικών πυλών τους υπολογισμούς για κάθε alu_op όπως ήταν στα σχήματα 2-5.

Για το calc:

Όρισα τις εισόδους (clk,btnc,btnl,btnu,btnr,btnd,sw) και την έξοδο (led) του module calc βάση της εκφώνησης. Έπειτα όρισα τα accumulator,alu_op1,alu_op2,alu_result και alu_op που χρειαζόμουν για τη συνέχεια. Κάλεσα το module calc_enc με ορίσματα για είσοδο (btnr,btnl,btnc) και για έξοδο (alu_op). Μέσω assign έθεσα τιμές στα alu_op1 και alu_op2. Κάλεσα το module alu με ορίσματα για είσοδο (alu_op1,alu_op2,alu_op) και για έξοδο (alu_result). Μέσω ενός always block φρόντισα σε κάθε άνοδο της ακμής του ρολογιού αν το btnu είναι πατημένο ο accumulator να γίνεται 0 αλλιώς αν το btnd είναι πατημένο ο accumulator να παίρνει ως τιμή το 16 χαμηλότερα bits του alu_result. Τέλος στο led έβαλα την τρέχουσα τιμή του accumulator.

Για το calc_tb:

Έβαλα το timescale, όρισα το module και τις εισόδους και εξόδους που χρειαζόμουν για το module calc. Κάλεσα το module calc με εισόδους (clk_tb,btnc_tb,btnl_tb,btnu_tb,btnr_tb,btnd_tb,sw_tb) και έξοδο (led_tb). Μέσω ενός initial block έκανα την τιμή του clk_tb 0 και φρόντισα η τιμή του να αντιστρέφεται κάθε 5ps. Στη συνέχεια μέσω πάλι ενός initial block αρχικοποίησα τις τιμές των υπολοίπων και πρόσθεσα σενάρια δοκιμών ώστε να ελεγχθεί η ορθή λειτουργία της ALU και της αριθμομηχανής.



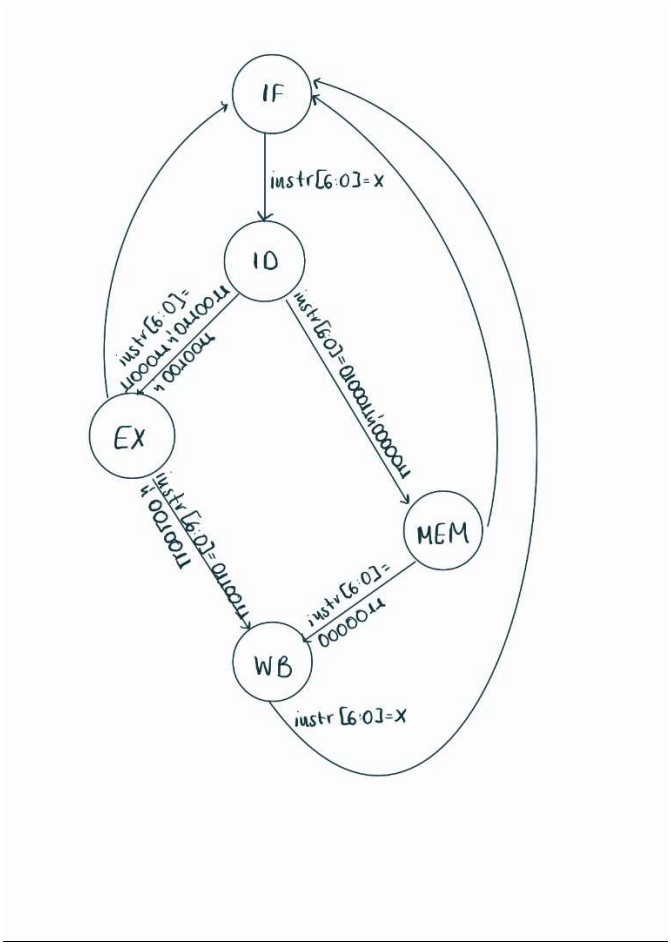
ΑΣΚΗΣΗ 3

Όρισα τις εισόδους (clk,readReg1,readReg2,writeReg,writeData,write)και εξόδους(readData1,readData2) του module regfile βάση της εκφώνησης. Στη συνέχεια δημιούργησα το αρχείο καταχωρήτων 32x32 bit και μέσω ενός initial block και ενός for loop αρχικοποίησα τους καταχωρητές με μηδενικά .Ύστερα μέσω ενός always block φρόντισα σε κάθε άνοδο της ακμής του ρολογιού να διαβάζονται τα δεδομένα από τους καταχωρητές που επιλέγονται από τις διευθύνσεις readReg1 και readReg 2 και εάν το σήμα εγγραφής write είναι ενεργοποιημένο και η διεύθυνση εγγραφής writeReg δεν ταυτίζεται με τις διευθύνσεις ανάγνωσής readReg1 και readReg2 τότε τα δεδομένα να εγγράφονται στον καταχωρήτη που καθορίζεται από τη διεύθυνση εγγραφής.

ΑΣΚΗΣΗ 4

Όρισα τις εισόδους (clk,rst,instr,PCSrc,ALUSrc,RegWrite,MemToReg,ALUCtrl,loadPC,dReadData) και εξόδους (PC,Zero,dAddress,dWriteData,WriteBackData) και την παράμετρο (INITIAL_PC) του module datapath βάση της εκφώνησης. Έπειτα όρισα τα ALU_op2,WriteData,branch_offset,immIext,immBext,immSext,readData1,readData2,ALU_result,readReg1,readReg2,writeReg που χρειαζόμουν για την συνέχεια. Κάλεσα το module regfile με εισόδους (clk,readReg1,readReg2,writeReg,WriteData,RegWrite) και εξόδους (readData1,readData2) και το module alu με εισόδους (readData1,ALU_op2,ALUCtrl) και εξόδους (Zero,ALU_result). Υστέρα δημιούργησα τον Program Counter ,μέσω ενός always block φρόντισα σε κάθε άνοδο της ακμής του ρολογιού ή κάθοδο της ακμής του rst αν το rst είναι 0 το PC να παίρνει την τιμή του INITIAL_PC αλλιώς εάν το loadPC είναι 1 αν το PCSrc είναι 1 το PC να παίρνει την τιμη PC + branch_offset αλλιώς να παίρνει την τιμή PC + 4. Μέσω ενός always block φρόντισα σε κάθε αλλαγή τιμής του instr να ενημερώνεται η τιμή των readReg1,readReg2 και writeReg που εξαρτώνται από αυτό όπως φαίνεται από το σχήμα 7. Έπειτα δημιούργησα τις τιμές του Immediate Generation όπως περιέγραφε η εκφώνηση και με την βοήθεια της σελίδας 148.Μέσω πάλι ενός always block φρόντισα αν το ALUSrc είναι 1 το ALU_op2 να παίρνει την τιμή του immIext αλλιώς να παίρνει την τιμή του readData2 και παράλληλα δημιούργησα το branch_offset που είναι το PC + immBext. Δημιούργησα το Write Back, μέσω ενός always block φρόντισα αν το MemToReg είναι 1 το WriteData να παίρνει την τιμή του dReadData αλλιώς να παίρνει την τιμή του ALU_result , το WriteBackData να παίρνει την τιμή του dWriteData. Τέλος βάση του σχήματος 7 φρόντισα μέσω ενός always block το dAddress να παίρνει την τιμή του ALU_result και το dWriteData την τιμή του readData2.

ΑΣΚΗΣΗ 5



Για το top_proc:

Όρισα τις εισόδους (clk,rst,instr,dReadData)και τις εξόδους(PC, dAddress, dWriteData, MemRead,MemWrite,WriteBackData) του module top_proc βάση της εκφώνησης. Όρισα τις παραμέτρους INITIAL_PC,IF,ID,EX,MEM,WB και τα current_state, next_state, ALUCtlr, ALUSrc, loadPC, PCSrc, MemToReg, RegWrite,Zero που χρειαζόμουν για την συνέχεια. Κάλεσα το module datapath με εισόδους (clk, rst, instr, PCSrc, ALUSrc, RegWrite, MemToReg, ALUCtrl, loadPC, dReadData), εξόδους (PC,Zero,dAddress,dWriteBack,WriteBackData) και παράμετρο (INITIAL_PC). Δημιούργησα το State Memory μέσω ενός always block φρόντισα σε κάθε άνοδο της ακμής του ρολογιού ή κάθοδο της ακμής του rst αν το rst είναι 0 το current_state γίνεται IF αλλιώς το current_state γίνεται next_state. Στη συνέχεια δημιούργησα το Next State Logic μέσω ενός always block και μιας case εντολής για να αναγνωρίζει την current_state.

- Αν το current_state είναι IF το next_state γίνεται ID
- Αν το current_state είναι ID έβαλα μια case εντολή που να αναγνωρίζει το instr[6:0], αν το instr[6:0] είναι 0000011 ή 0100011 το next_state γίνεται MEM αλλιώς αν το instr[6:0] είναι 0110011 ή 0010011 ή 1100011 το next_state γίνεται EX

- Αν το current_state είναι EX έβαλα μια case εντολή που να αναγνωρίζει το instr[6:0], αν το instr[6:0] είναι 0110011 ή 0010011 το next_state γίνεται WB αλλιώς αν το instr[6:0] είναι 1100011 το next_state γίνεται IF.
- Αν το current_state είναι MEM έβαλα μια case εντολή που να αναγνωρίζει το instr[6:0] αν το instr[6:0] είναι 0000011 το next_state γίνεται WB αλλιώς αν το instr[6:0] είναι 0100011 το next_state γίνεται IF.
- Αν το current_state είναι WB το next_state γίνεται IF.

Υστέρα δημιούργησα το Output Logic μέσω ενός always block αρχικοποίησα όλα τα ALUCtrl,ALUSrc,MemRead,MemWrite,MemToReg,RegWrite,loadPC,PCSrc σε 0 και πρόσθεσα μια case εντολή για να αναγνωρίζει το current_state.

- Αν το current_state είναι IF έκανα τα loadPC και PCSrc να γίνουν 0.
- Αν το current_state είναι ID έβαλα μια case εντολή που να αναγνωρίζει το instr[6:0]
    - αν το instr[6:0] είναι 0000011 ή 0100011 έβαλα στο ALUCtrl την τιμή 0010 και στο ALUSrc την τιμή 1
    - αν το instr[6:0] είναι 0110011 έκανα την τιμή του ALUSrc 0 και έβαλα μια case εντολή που να αναγνωρίζει τα instr[31:25] instr[14:12] για να καθορίζει ποια πράξη της ALU θα εκτελεστεί
        - αν το {instr[31:25],instr[14:12]} είναι 0000000111 έβαλα στο ALUCtrl την τιμή 0000 για να εκτελεστεί η πράξη AND
        - αν το {instr[31:25],instr[14:12]} είναι 0000000110 έβαλα στο ALUCtrl την τιμή 0001 για να εκτελεστεί η πράξη OR
        - αν το {instr[31:25],instr[14:12]} είναι 0000000001 έβαλα στο ALUCtrl την τιμή 0011 για να εκτελεστεί η πράξη SLL
        - αν το {instr[31:25],instr[14:12]} είναι 0000000101 έβαλα στο ALUCtrl την τιμή 0100 για να εκτελεστεί η πράξη SRL
        - αν το {instr[31:25],instr[14:12]} είναι 0100000101 έβαλα στο ALUCtrl την τιμή 0101 για να εκτελεστεί η πράξη SRA
        - αν το {instr[31:25],instr[14:12]} είναι 0000000010 έβαλα στο ALUCtrl την τιμή 0111 για να εκτελεστεί η πράξη SLT
        - αν το {instr[31:25],instr[14:12]} είναι 0000000100 έβαλα στο ALUCtrl την τιμή 0100 για να εκτελεστεί η πράξη XOR
    - αν το instr[6:0] είναι 0010011 έκανα το ALUSrc 1 και έβαλα μια case εντολή που να αναγνωρίζει το instr[14:12]
        - αν το instr[14:12] είναι 000 έβαλα στο ALUCtrl την τιμή 0010 για να εκτελεστεί η πράξη ADDI
        - αν το instr[14:12] είναι 111 έβαλα στο ALUCtrl την τιμή 0000 για να εκτελεστεί η πράξη ANDI
        - αν το instr[14:12] είναι 110 έβαλα στο ALUCtrl την τιμή 0001 για να εκτελεστεί η πράξη ORI
        - αν το instr[14:12] είναι 001 έβαλα στο ALUCtrl την τιμή 0011 για να εκτελεστεί η πράξη SLLI
        - αν το instr[14:12] είναι 101 έβαλα στο ALUCtrl την τιμή 0100 για να εκτελεστεί η πράξη SRLI
        - αν το instr[14:12] είναι 010 έβαλα στο ALUCtrl την τιμή 0111 για να εκτελεστεί η πράξη SLTI

        - αν το instr[14:12] είναι 100 έβαλα στο ALUCtrl την τιμή 1000 για να εκτελεστεί η πράξη XORI
    - αν το instr[6:0] είναι 1100011 έβαλα στο ALUSrc την τιμή 0 και στο ALUCtrl την τιμή 0110 για να εκτελεστεί η πράξη SUB
- αν το current_state είναι EX έβαλα μια case εντολή για να αναγνωρίζει το instr[6:0] αν το instr[6:0] είναι 0110011 ή 0010011 έβαλα στο MemToReg την τιμή 0 και στο RegWrite την τιμή 1 αλλιώς αν το instr[6:0] είναι 1100011 έβαλα στο loadPC την τιμή 1 και αν το Zero είναι 1 έκανα το PCSrc να γίνεται 0 αλλιώς να γίνεται 1
- αν το current_state είναι MEM έβαλα μια case εντολή που να αναγνωρίζει το instr[6:0] αν το instr[6:0] είναι 0000011 έκανα το MemRead να παίρνει την τιμή 1 αλλιώς αν το instr[6:0] είναι 0100011 έβαλα στο MemWrite την τιμή 1 και στο PCSrc την τιμή 0
- αν το current_state είναι WB έκανα την τιμή του RegWrite 1 και έβαλα μια case εντολή που να αναγνωρίζει το instr[6:0]
    - αν το instr[6:0] είναι 0110011 ή 0010011 έκανα το loadPC να παίρνει την τιμή 1 και το PCSrc να παίρνει την 0
    - αν το instr[6:0] είναι 0000011 έκανα το MemToReg να παίρνει την τιμή 1 ,το loadPC να παίρνει την τιμή 1 και το PCSrc να παίρνει την 0
    - αν το instr[6:0] είναι 1100011έκανα το loadPC να παίρνει την τιμή 1 και αν το Zero είναι 1 το PCSrc να παίρνει την 1

<u>Για το top_proc_tb:</u>

Έβαλα το timescale, όρισα το module και τις εισόδους και εξόδους που χρειαζόμουν για το module. Κάλεσα το module top_proc με εισόδους (clk,rst,instr,dReadData) και εξόδους (PC,dAddress,dWriteData,MemRead,MemWrite,WriteBackData),το module DATA_MEMORY με είσοδους(clk,MemWrite,dAddress[8:0],dWriteData) και έξοδο(dReadData) και το module INSTRUCTION_MEMORY με εισόδους (clk,PC[8:0])και έξοδο(instr). Μέσω ενός initial block έκανα την τιμή του clk_tb 0 και φρόντισα η τιμή του να αντιστρέφεται κάθε 5ps. Στη συνέχεια μέσω πάλι ενός initial block έκανα την τιμή του rst 0 και φρόντισα να γίνει 1 μετά από 15ps και να τερματίσει το πρόγραμμα μετά από 1000ps.