## ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ

**1Η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ**

**ΚΩΝΣΤΑΝΤΙΝΟΣ ΝΙΚΟΛΟΣ**

**ΑΜ:2019030096**

**Σκοπός:**

Σκοπός της συγκεκριμένης άσκησης είναι η δημιουργία ενός πλήρως λειτουργικού single circle επεξεργαστή με τη δυνατότητα υλοποίησης ορισμένων εντολών τύπου R-type , I-type.

Πιο συγκεκριμένα στις 3 φάσεις είδαμε:

* 1η Φάση

Υλοποιήθηκε η ALU, μια μονάδα υπεύθυνη για την λειτουργία βασικών πράξεων και ολισθήσεων, και ένα αρχείο αποτελούμενο από 32 καταχωρητές για εγγραφή

και ανάγνωση τους.

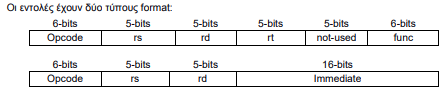
* 2η Φάση

Δημιουργήθηκαν 4 διαφορετικά στάδια απαραίτητα για τη λειτουργία του επεξεργαστή IFSTAGE,DECSTAGE,EXSTAGE,MEMSTAGE.

* 3η Φάση

Κατασκευάστηκε το Datapath ενα σύνολο αποτελούμενο από τα 4 προηγούμενα στάδια και το control τα οποία συνδέθηκαν δημιουργώντας τον επεξεργαστή.

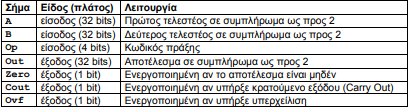
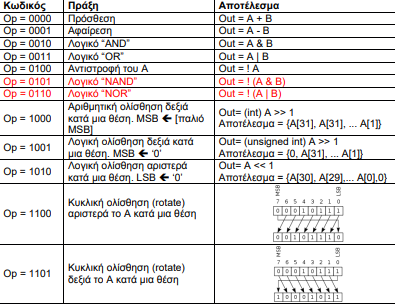
**Περιγραφή:**

Για την υλοποίηση των εντολών που ζητούνται βασιζόμαστε στην σταδιακή δημιουργία σημάτων που τις αποτελούν όπως τα opcode, rs ,rd καταλήγοντας στα δύο format εντολών που μας δίνονται. 

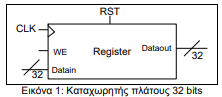
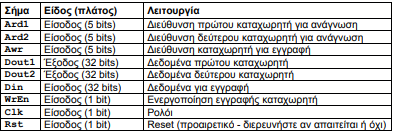
**Αναλυτικότερα στην 1ή Φάση:**

Για την υλοποίηση της ALU χρειάστηκαν δύο καταχωρητές και ένα σήμα opcode για την επιλογή της μεταξύ τους πράξης.Για την δημιουργία του σήματος Cout(cary out) επιλέξαμε την αποθήκευση του αποτελέσματος σε vector 32 bit χρησιμοποιώντας το τελευταίο ως Cout. Ενώ το σήμα Ovf ενεργοποιείται οτάν προκύπτει θετικό αποτέλεσμα ενώ προσθέτουμε δύο αρνητικούς ή όταν προσθέτουμε δύο θετικούς και προκύπτει αρνητικό αποτέλεσμα.Όσο για το Zero ενεργοποείται όποτε το αποτέλεσμα μας ισούται με 0 .

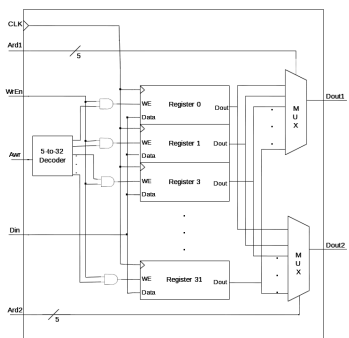
Οι πράξεις που υλοποιήθηκαν είναι οι εξής:

.

Ακόμη δημιουργείται ένα αρχείο καταχωρητών αποτελούμενο από 2 MUX των 32 bits, ένα decoder και τους καταχωρητές στους οποίους επιτρέπεται η εγγραφή εφόσον το σήμα WΕ είναι ενεργό

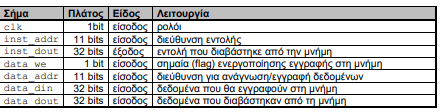
 . 

Χρησιμοποιούμε τις εισόδους Ard1,Ard2 για την επιλογή των καταχωρητών που θα εμφανιστούν στην έξοδο του αρχείου μέσω την πολυπλεκτών. Εισάγοντας την είσοδο Awr στον αποκωδικοποιητή γνωρίζουμε τη διεύθυνση καταχωρητή προς εγγραφή.

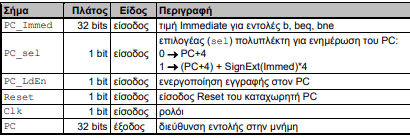
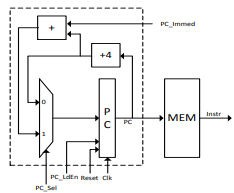


**Αναλυτικότερα στην 2ή Φάση:**

1. Σε αυτή τη φάση υλοποιήθηκε το module της μνήμης RAM



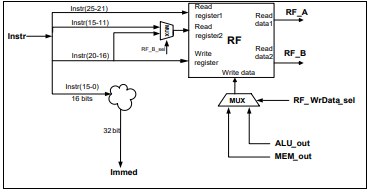
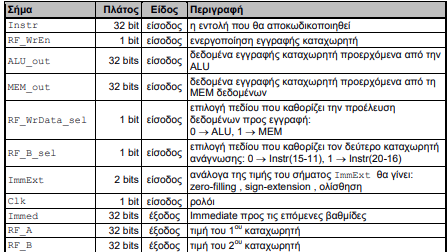
Η μνήμη είναι χωρισμένη σε δύο τομείς τον Text segment (0 ..1024) και τον Data segment (1024..2048).Για την αρχικοποίηση της δόθηκαν τα αρχεία rom που βρίσκονται στον φάκελο Waveforms.

1. Δημιουργία του IFSTAGE. Δεχόμαστε ως είσοδο το PCImmediate για την πιθανή διακλάδωση του επεξεργαστή σε εντολές τύπου branch, μέσω ενός πολυπλέκτη 2x1 επιλέγεται εάν θα μετακινήσουμε τον καταχωρητή PC (Program Counter) στην αμέσως επόμενη εντολή ή αν θα προσθέσουμε το immediate για την διακλάδωση του.  

**Γ) DEC-STAGE**

Αποτελείται από το αρχείο καταχωρητών 2 MUX των 2 bits και το Cloud Entity με σκοπό να

δεχόμαστε ως είσοδο μια εντολή μεγέθους 32 bit η οποία θα χωρίζεται στα εξής σηματα:

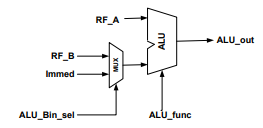
 

Πιο συγκεκριμένα όπως φαίνεται και στην παραπάνω εικόνα χρησιμοποιούμε τα bits 25-21 για τον 1ο καταχωρητή και επιλέγουμε μέσω ενός multiplexer αν θα χρησιμοποιήσουμε τα bits 15-11 ή 20-16 για την είσοδο του 2ου καταχωρητή ανάλογα τον τύπο εντολής που δεχόμαστε. Ακόμη το DECSTAGE είναι υπεύθυνο για την διαμόρφωση του σήματος immediate που θα εισαχθεί αργότερα στο IFSTAGE και EXSTAGE. Για την επίτευξη του συγκεκριμένου σκοπού το Cloud\_Entity δέχεται ως είσοδο 2bits και επιλέγει την πράξη με την οποία θα διαμορφωθεί το immediate. Τέλος, χρησιμοποιούμε έναν ακόμη πολυπλέκτη ο οποίος δέχεται το σήμα RF\_WrData\_Sel για να καθορίσει την είσοδο προς εγγραφή στο αρχείο καταχωρητών.

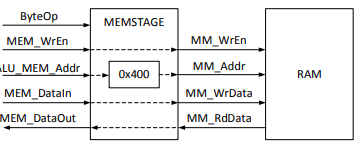
Σύμφωνα με το 2bit σήμα ImmExt έχουμε 4 περιπτώσεις κωδικοποίησης του Immediate.

|  |  |
| --- | --- |
| ImmExt | Κωδικοποίηση |
| 00 | Zero Fill |
| 01 | ZeroFill & Sll16 |
| 10 | Sign Extend |
| 11 | SignExt & Sll2 |

**Δ) EXSTAGE**

****Στο συγκεκριμένο στάδιο χρησιμοποιείται η αριθμομηχανή ALU της οποίας η είσοδος ελέγχεται από έναν πολεπλέκτη 2x1, σύμφωνα με το σήμα ALU\_Bin\_sel το οποίο εξαρτάται από τον τύπο εντολής που θα υλοποιήσουμε (επιλέγουμε Immediate για εντολές τύπου addi,ori).Ακόμη η δίνεται ως είσοδος το ALU\_func κάνοντας γνωσστή την πράξη μεταξύ των δύο καταχωρητών που ζητείται.

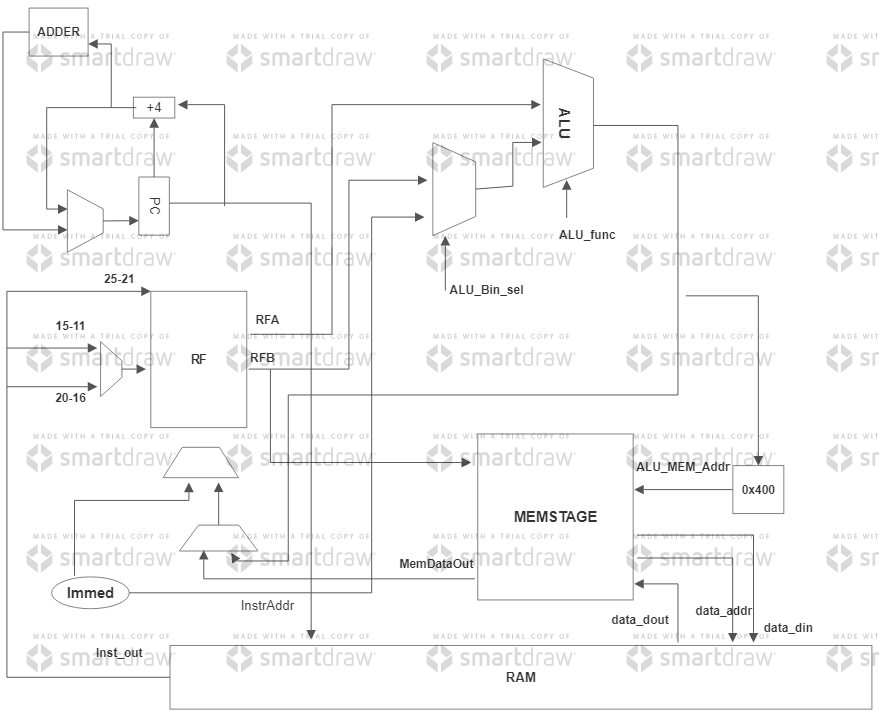
**Ε) MEMSTAGE**

****

Τέλος υλοποιήσαμε το MEMSTAGE που είναι υπεύθυνο για την επικοινωνία του επεξεργαστή με την μνήμη υλοποιώντας τις λειτουργίες ανάγνωσης και εγγραφής από και προς τη μνήμη. Πιο συγκεκριμένα δέχεται ως είσοδο το ByteOp με το οποίο επιλέγεται αν θα διαβάσουμε/γράψουμε μία ολόκληρη λέξη των 32bit ή ένα μόνο byte αποτελούμενο από 8 bits. Αξίζει να αναφερθεί ότι για να μεταφέρουμε κατάλληλα την διεύθυνση εισερχόμενη από την ALU στην μνήμη προσθέτουμε ένα offset 1024.

**Αναλυτικότερα στην 3ή Φάση:**

Στην 3η φάση υλοποιήθηκε αρχικά το datapath στο οποίο συνδέσαμε τα προαναφερθέντα 4 stages σύμφωνα με το παρακάτω block diagram:



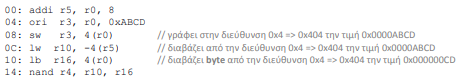
Έπειτα κατασκευάστηκε το CONTROL για την δημιουργία των σημάτων που καθορίζουν την λειτουργία του datapath όπως το ImmExt ή το ALU\_func βασισμένα στην είσοδο Opcode.

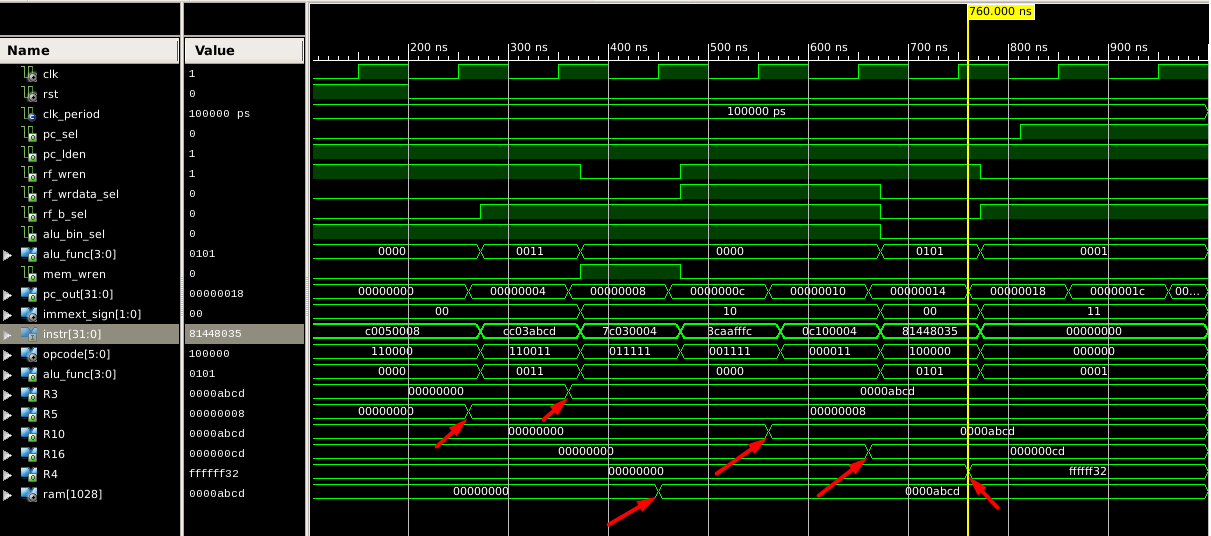
Τέλος δημιουργήθηκε το PROC\_SH το οποίο δέχεται μόνο CLK , RST ως σήματα και ενώνει τα 2 components datapath και control. **Για το testebnch του PROC\_SH χρησιμοποιούμε τα αρχεία rom1, rom2 που βρίσκονται στο φάκελο waveforms μαζί με μία ανάλυση των κυματομορφών στο αρχείο pdf.**

**ΠΡΟΓΡΑΜΜΑΤΑ ΑΝΑΦΟΡΑΣ**

**1ο Πρόγραμμα**

Ακλουθούμε μία σειρά εντολών τσεκάροντας την εγγραφή και την ανάγνωση από την μνήμη.

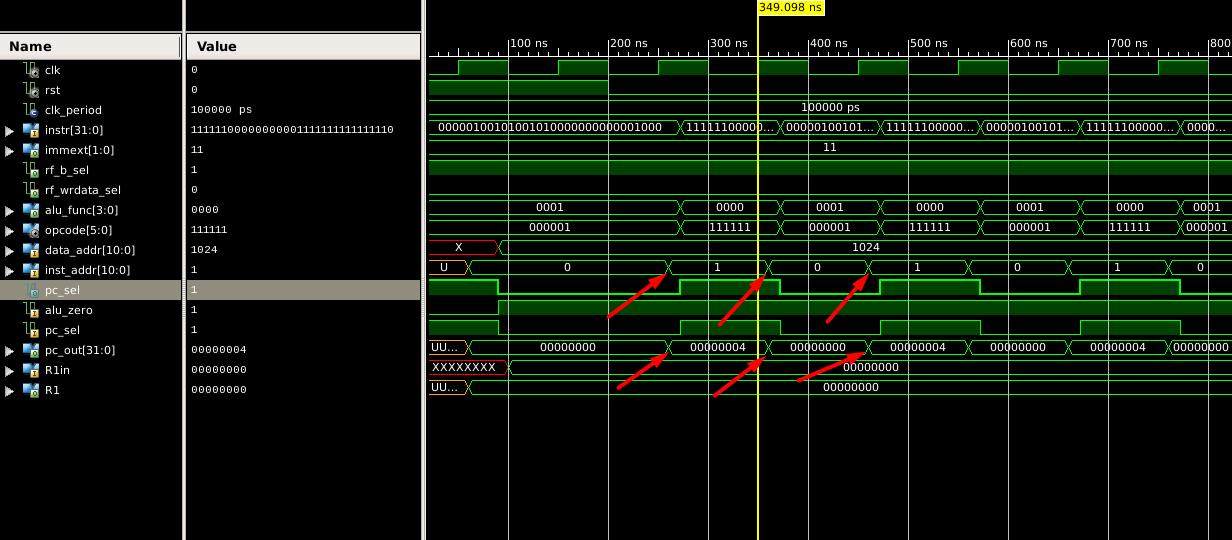




Τη χρονική στιγμή 200ns απελευθερώνουμε το RST και παρατηρούμε την αλλαγή της τιμής του καταχωρητή R5 σε 8. Έπειτα τοποθετούμε την τιμή 0xABCD τον καταχωρητή R3 με την εντολή ori μια χρονική περίοδο αργότερα. Με την εντολή sw γράφουμε στην διεύθυνση 1028 της μνήμης (R0=0 +4) την τιμή του καταχωρητή R3 στα 450ns. Φορτώνουμε με την εντολή lw στον καταχωρητή R10 από τη διεύθυνση 1028 της μνήμης (R5=8 -4) στα 560ns . Ακόμη φορτώνουμε με lb στον καταχωρητή R16 τα 8bits της τιμής στην διεύθυνση μνήμης 1028 ¨κόβοντας¨ την από 0xABCD σε CD στα 660ns. Τέλος εκτελούμε την πράξη R10 NAND R16 με αποτέλεσμα ο καταχωρητής R4 να πάρει τιμή ffffff32 στα 760ns.

**2ο Πρόγραμμα**

****



Απελευθερώνουμε το RST στα 200 ns. Αρχικά ελέγχουμε αν ο καταχωρητής R5 έχει την τιμή 8 κάτι που δεν ισχύει επομένως δεν γίνεται κάποιο branch και προχωράμε στην επόμενη εντολή, το inst\_addr γίνεται 1 στα 260ns.Στην επόμενη εντολή κάνουμε branch μία θέση πίσω (-2 +1) επιστρέφοντας στο inst\_addr=0 στα 36ns. Ακολουθεί μία ατέρμονη λούπα επιστρέφοντας ξανά και ξανά στις θέσεις 0 ,1 μεταξύ των 2 πρώτων εντολών ενώ δεν φτάνουμε ποτέ στην 3η εντολή άρα ο καταχωρητής R1 δεν παίρνει ποτέ την τιμή 1 και παραμένει στο 0.

**Συμπεράσματα:**

Μετά το πέρας της συγκεκριμένης εργαστηριακής άσκησης κατανοήσαμε πλήρως την λειτουργία του προγράμματος Xilinx, της top level αρχιτεκτονικής ακόμη και της αποτελμάτωση μικρότερων τμημάτων κώδικα vhdl υλοποιώντας τον επεξεργαστή συνδυάζοντας γνώσεις που αποκτήθηκαν στα μαθήματα προχωρημένη λογική σχεδίαση και ψηφιακούς υπολογιστές.