14/4/2022

Εργασία #1: Σχεδίαση επεξεργαστή ενός κύκλου

Μανουδάκη Χρυσηίς 2019030201

Σκοπός της εργασίας

Σκοπός της συγκεκριμένης εργασίας, είναι η δημιουργία ενός επεξεργαστή, ο οποίος θα λειτουργεί με έναν κύκλο ρολογιού (single cycle clock) και θα υλοποιείται από τμήματα ενός non-pipelined επεξεργαστή. Το project αυτό, αποτελείται από τρείς φάσεις με την τελευταία φάση να υλοποιεί το top Level module που συγκεντρώνει τα περιεχόμενα κάθε φάσης. Η έκδοση του εργαλείου Xilinx που χρησιμοποιείται είναι η Xilinx ISE 14.7.

Φάση 1η

Στην 1η φάση υλοποιήθηκαν τα ολοκληρωμένα κυκλώματα:

- Μονάδα αριθμητικών και λογικών πράξεων (ALU)
- Αρχείο καταχωρητών (Register File ή RF)

ALU

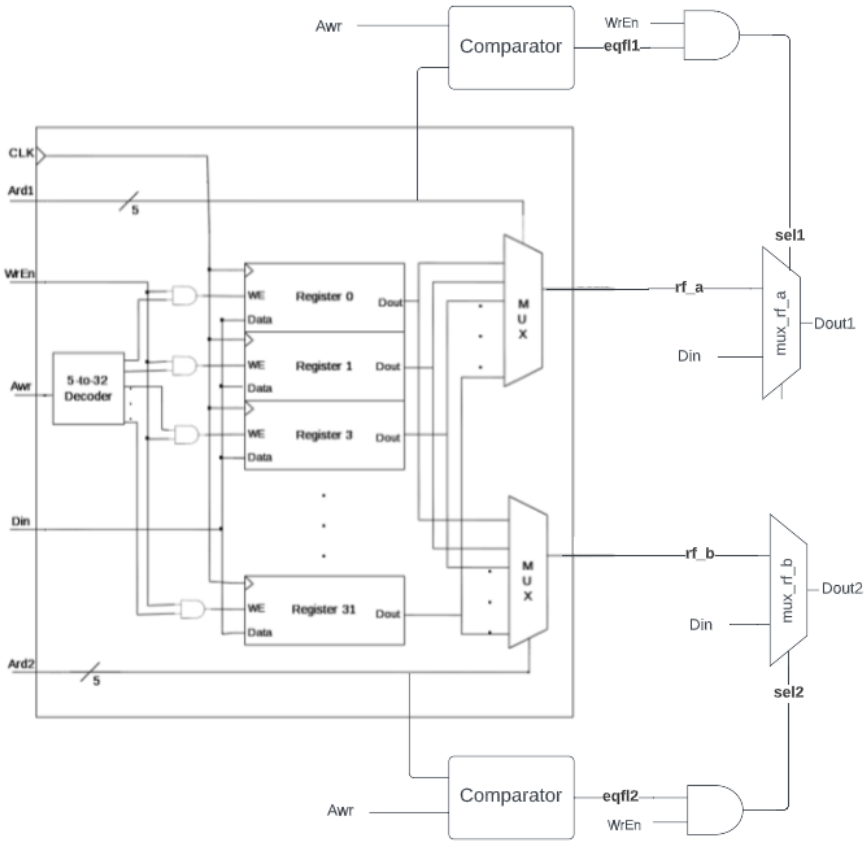
Για την ALU χρησιμοποιήθηκαν δύο 32-bits σήματα εισόδου A και B που ήταν σε συμπλήρωμα ως προς 2 και ένα σήμα εισόδου Op 4-bits το οποίο ορίζει τον κωδικό της πράξης που θα εκτελέσει η ALU. Τα σήματα εξόδου, αποτελούνται από το 32-bits σήμα Out το οποίο είναι το αποτέλεσμα της ALU σε συμπλήρωμα ως προς δύο και τα τρία σήματα 1-bit που εντοπίζουν την υπερχείλιση του κυκλώματος, το κρατούμενο εξόδου και το μηδενικό αποτέλεσμα. Τα σήματα αυτά είναι το Ovf, Cout και Zero αντίστοιχα. Να σημειωθεί ότι τα προαναφερθέντα σήματα είναι active high.

Για την υλοποίηση της ALU χρησιμοποιήθηκε η τεχνική case με το σήμα enum_op το οποίο παίρνει όλες τις πιθανές περιπτώσεις του κωδικού Op . Οι πράξεις στην ALU ορίστηκαν σύμφωνα με τον πίνακα που παρατίθεται στην σελίδα 2 της εκφώνησης και υλοποιήθηκαν με εντολές από τις έτοιμες βιβλιοθήκες : use IEEE.STD_LOGIC_1164.ALL, use ieee.numeric_std.all, use ieee.std_logic_unsigned.all, use ieee.std_logic_signed.all.

Register File (RF)

Για το Αρχείο Καταχωρητών (Register File) χρησιμοποιήθηκαν 32 καταχωρητές των 32-bits εισόδου-εξόδου (Din, Dout), που αποτελούνταν από ένα ρολόι (CLK), ένας σήμα Reset που μηδενίζει τους καταχωρητές και ένα σήμα εισόδου 1-bit (WE) για να πραγματοποιείται η εγγραφή σε αυτούς. Ακόμα, στο RF χρησιμοποιήθηκε ένας αποκρυπτογράφος (Decoder) που μετέτρεπε ένα σήμα εισόδου των 5-bits σε 32-bits ανάλογα με την 5-Bits διεύθυνση καταχωρητή για εγγραφή (Awr) και δίνει ως έξοδο ένα 32-bits σήμα (dcout) που περιέχει την πληροφορία για το ποιος καταχωρητής θα γραφτεί. Στη συνέχεια, κάθε ένα από τα 32 bit της εξόδου dcout, εισέρχεται αντίστοιχα σε μία από τις 32 πύλες AND, 2 εισόδων, μαζί με το σήμα ενεργοποίησης εγγραφής (WrEn) και κάθε έξοδος από τις πύλες δίνεται ως σήμα εγγραφής στον αντίστοιχο καταχωρητή. Επίσης, χρησιμοποιήθηκαν 2 πολυπλέκτες των 32-1 καθώς και 2 πολυπλέκτες 2-1. Οι πολυπλέκτες 32-1, συνδέθηκαν με έναν πίνακα 32X32 (mux_in) που περιέχει τις εξόδους από τους 32 καταχωρητές. Οι πολυπλέκτες 2-1, δέχονται στη θέση μηδέν την έξοδο από τον αντίστοιχο πολυπλέκτη 32-1 και στη θέση 1 την είσοδο Din. Το σήμα που διαλέγει την έξοδο (sel) προκύπτει από ένα κύκλωμα που θα εξηγηθεί παρακάτω και δεν ζητείται από την εκφώνηση. Προκειμένου λοιπόν να παραχτούν τα σωστά σήματα από το αρχείο καταχωρητών, υλοποιήθηκαν 2 όμοια κυκλώματα που αποτελούνται από έναν συγκριτή (Comparator), ο και μία πύλη AND. Η λειτουργία αυτής της προσθήκης, είναι η σύγκριση κάθε διεύθυνσης καταχωρητή για ανάγνωση (Ard1, Ard2) με την διεύθυνση καταχωρητή για εγγραφή

(Awr) ώστε να δοθεί ως έξοδος, Dout1 ή Dout2 αντίστοιχα, η είσοδος Din όταν είναι ίδια και η έξοδος του πολυπλέκτη στην περίπτωση που είναι διαφορετικά. Η πύλη AND, χρησιμοποιείται μεταξύ του σήματος WrEn και του αποτελέσματος κάθε Comparator ώστε να παραχθεί σήμα μόνο αν επιτρέπεται η εγγραφή.



Νέα εκδοχή του Register File

Για την υλοποίηση του Αρχείου Καταχωρητών (Register File) στο περιβάλλον του Xilinx χρησιμοποιήθηκε η τεχνική του for-generate για τη δημιουργία των 32 καταχωρητών και η διαδικασία του Port Mapping για την κατάλληλη ένωση των κυκλωμάτων που παράχθηκαν.

Φάση 2η

Στην 2η φάση υλοποιήθηκαν τα ολοκληρωμένα κυκλώματα:

- Ανάκλησης εντολών (Instruction Fetch ή IF)
- Αποκωδικοποίησης εντολών (Instruction Decoding ή DEC)
- Εκτέλεσης εντολών (Execution ή EX)
- Πρόσβασης μνήμης (Memory ή MEM)

Σαν γενική ιδέα της 2ης φάσης ασχοληθήκαμε με τη δημιουργία των κυκλωμάτων του Datapath χωρίς να γίνει κάποια ένωση μεταξύ τους.

RAM

Αρχικά υλοποιήθηκε η μνήμη RAM με τον έτοιμο κώδικα που μας δόθηκε με μόνη διαφορά το ROM αρχείο που φτιάχτηκε σύμφωνα με τα προγράμματα αναφοράς στην εκφώνηση.

Instruction Fetch Stage(IF)

Για την υλοποίηση της βαθμίδα ανάκλησης εντολών (IFSTAGE) χρησιμοποιήθηκε ένα 32-bits σήμα εισόδου (PC_Immed), το οποίο είναι λογικά ολισθημένο αριστερά (Sll) κατά 2 και επίσης είναι σήμα εισόδου σε έναν αθροιστή. Επιπλέον ,περιέχεται ένας 2-1 πολυπλέκτης με συνθήκη ένα σήμα εισόδου 1-bit (PC_sel),ο οποίος δέχεται ως εισόδους :το 32-bits σήμα που προέρχεται από τον αθροιστή και το 32-bits σήμα που προέρχεται από το PC αυξημένο κατά "0100"(4). Στη συνέχεια, χρησιμοποιήθηκε ένας καταχωρητής 32 bit ο οποίος είχε ως είσοδο την έξοδο 32-bits του πολυπλέκτη , ένα σήμα 1-bit (PC_LdEn) που ενεργοποιεί την εγγραφή στο PC, ένα σήμα 1-bit (Reset) που μηδενίζει την έξοδο του PC για ένα κύκλο, όταν ενεργοποιείται και ένα ρολόι (Clk). Συμπερασματικά, η Ανάκλησης εντολών (Instruction Fetch ή IF) ανάλογα με το σήμα εισόδου 1-bit (PC_sel) που παίρνιε, παράγει μια έξοδο 32-bits (PC_Out) η οποία ή είναι αυξημένη κατά "0100"(4) ή αυξημένη τόσο όσο η είσοδος που δόθηκε (PC_Immed) λογικά ολισθημένη αριστερά (Sll) κατά "10"(2) συν "0100"(4) από την προηγούμενη έξοδο.

Decode Stage (DEC)

Για την υλοποίηση της βαθμίδα αποκωδικοποίησης εντολών (DECSTAGE) χρησιμοποιήθηκε ένα 32-bits σήμα εισόδου (Instr) ,το οποίο χωρίστηκε στα εξής bits: Instr(15-0) ,Instr(15-11), Instr(20-16), Instr(25-21) και Instr(31-26). Το σήμα Instr(15-11) είναι το Rt και το σήμα Instr(20-16) το Rd, όπου θα εισαχθούν ως επιλογές σε ένα πολυπλέκτη 2-1 με έξοδο να εισάγεται κατόπιν στον δεύτερο καταχωρητή για ανάγνωση του Register File. Το σήμα Instr(25-21) είναι το Rs και εισάγεται στον πρώτο καταχωρητή για ανάγνωση του Register File. Στις περιπτώσεις που έχουμε εντολή με immediate το σήμα Instr(15-0), που τον αναπαριστά, εισάγεται σε ένα 'συννεφάκι' το οποίο σύμφωνα με το σήμα ImmExt 2bit που εισάγεται, μετατρέπει τον 15bits immediate σε 32bits με τη χρήση Sign-Extend, Sign-Extend και αριστερή ολίσθηση κατά 2, Zero-Fill και αριστερή ολίσθηση κατά 16 και Zero-Fill για τις τιμές '00', '01', '10', '11' του ImmExt αντίστοιχα. Για να παραχθεί το σήμα αυτό, δημιουργήθηκε ένας encoder ο οποίος ανάλογα με τον κωδικό εντολής που λαμβάνει από το σήμα Instr(31-26), αποφασίζει με ποιον τρόπο θα επεκταθεί ο κάθε immediate και δίνει την αντίστοιχη τιμή στο σήμα ImmExt. Να σημειωθεί ότι πως θα επεκταθεί ο immediate Της κάθες εντολής είναι προκαθορισμένο και μας δίνεται στον πίνακα της σελίδας 9 στην εκφώνηση. Για να αποφασιστεί τι δεδομένα θα γραφτούν στους καταχωρητές του RegisterFile, χρησιμοποιείται ένας πολυπλέκτης 2-1 με εισόδους την έξοδο από τη μνήμη (MEM_out) και την έξοδο από τον αθροιστή (ALU_out) του EXSTAGE, καθώς και ένα σήμα (RF_WrData_sel) που επιλέγει την τιμή και δίνεται από το Control.

Execution Stage (EX)

Η βαθμίδα εκτέλεσης εντολών (EXSTAGE) παίρνει κάποια σήματα εισόδου και άλλα ελέγχου και αναλόγως παράγει το τελικό αποτέλεσμα μέσα από αριθμητικές ή λογικές πράξεις. Για την υλοποίησή της χρησιμοποιήθηκαν 3 32-bits σήματα εισόδου (RF_A,RF_B, Immed).Τα 2 από αυτά (RF_B,Immed) λειτουργούν ως σήματα εισόδου σε έναν πολυπλέκτης 2-1 με συνθήκη ένα σήμα εισόδου 1-bit (ALU_Bin_sel) και έξοδο ένα προσωρινό 32-bits σήμα (ALU_in). Σε αυτή την βαθμίδα έγινε χρήση της ALU που υλοποιήθηκε στην 1η φάση. Επομένως έχει 2 εισόδους των 32-bits(RF_A, Mux_out),μια 4-bits είσοδο (ALU_func) που δρα ως συνθήκη όπως το σήμα Op στην ALU και παράγει μια έξοδο 32-bits (ALU_out) και μια έξοδο 1-bit (ALU_zero),η οποία λειτουργεί ως zero flag στην περίπτωση που οι αριθμητικές και λογικές πράξεις βγάζουν αποτέλεσμα μηδέν.

Memory Stage (MEM)

Για την υλοποίηση της βαθμίδας πρόσβασης μνήμης (MEMSTAGE), χρησιμοποιήθηκε ένα σήμα ελέγχου για επιλογή μεταξύ sw/lw (0) και lb/sb (1) 1-bit (ByteOp), ένα σήμα 1-bit για την ενεργοποίησης εγγραφής στη μνήμη (Mem_WrEn) και για την επιλογή μεταξύ Store και Load, δυο σήματα εισόδου 32-bits(ALU_MEM_Addr, MEM_DataIn) και ένα σήμα εισόδου 32-bits(MM_RdData),το οποίο παίρνει διευθύνσεις μνήμης από το αρχείο Rom και τις περνάει ως είσοδο στη βαθμίδα πρόσβασης μνήμης (MEM). Δυο είναι μόνο οι υλοποιήσεις σε αυτή την βαθμίδα : η μια είναι ότι στην είσοδο (ALU_MEM_Addr) προσθέτουμε το 0x400(1024) το λεγόμενο offset και το θέτουμε σε ένα προσωρινό σήμα memAdd από το οποίο ύστερα βάζουμε τα 12-2 bits στην 11Bits έξοδο MM_Addr για να διευθυνσιοδοτήσουμε τη μνήμη και η άλλη είναι ότι στην περίπτωση που το σήμα ελέγχου (ByteOp) είναι 1, τότε η 32bits έξοδος (MEM_DataOut) παίρνει μόνο τα 8 πρώτα bits, δηλαδή από το 0-7 και στα υπόλοιπα τα κάνει Zero Filling. Οι άλλες δυο έξοδοι των 32- bits(MM_WrEn,MM_WrData) μένουν ανέπαφες. Γενικά, όλα τα σήματα εξόδου προς την μνήμη (RAM) θα δώσουν τις απαραίτητες διευθύνσεις ώστε ο επεξεργαστής να ξέρει αν θα φορτώσει (Load) ή θα αποθηκεύσει (Store) στη μνήμη (RAM). Για την υλοποίηση της βαθμίδα πρόσβασης μνήμης (MEMSTAGE) στο περιβάλλον του Xilinx χρησιμοποιήθηκε η τεχνική του case-when και if-else και η διαδικασία του Port Mapping για την κατάλληλη ένωση των κυκλωμάτων που παράχθηκαν.

Φάση 2η

Στην 2η φάση υλοποιήθηκαν τα ολοκληρωμένα κυκλώματα:

- Η ολοκλήρωση του Datapath
- Η δημιουργία του Control
- Η ολοκλήρωση του Επεξεργαστή(Processor)

Datapath

Στόχος της βαθμίδας του Datapath, είναι η κατάλληλη ένωση όλων των κυκλωμάτων που υλοποιήθηκαν στην δεύτερη φάση, μεταξύ τους. Ειδικότερα, οι βαθμίδες DECSTAGE, EXSTAGE, MEMSTAGE και IFSTAGE ενώνονται με τον εξής τρόπο: Αρχικά, τα σήματα εξόδου (RF_A, RF_B, Immed) του DECSTAGE μεταφέρονται με την βοήθεια των εσωτερικών σημάτων 'RF_A_to_EX', 'RF_B_to_EX' και 'Immed_to_IF_EX' αντίστοιχα, στα σήματα εισόδου του EXSTAGE και το σήμα εξόδου 'RF_B' εισάγεται επίσης στην είσοδο 'MEM_DataIn' του MEMSTAGE. Κατόπιν, το σήμα εξόδου 'ALU_out' του EXSTAGE μεταφέρεται με την βοήθεια του εσωτερικού σήματος 'EX_ALU_out', στην είσοδο 'ALU_MEM_Addr' του MEMSTAGE και στην είσοδο 'ALU_out' του DECSTAGE. Επίσης, το σήμα εξόδου 'MEM_DataOut', που φέρει τα δεδομένα που εξέρχονται εκείνη τη στιγμή από την μνήμη, μεταφέρονται με το εσωτερικό σήμα 'ram', στο σήμα εισόδου 'MEM_out' του DECSTAGE. Για να οριστεί η τιμή του σήματος ελέγχου 'ByteOp' στο MEMSTAGE, δημιουργήθηκε ένας κωδικοποιητής (ByteOp_Encoder) ο οποίος σύμφωνα με τον κωδικό εντολής (Instruction(31-26)), αρχικοποιεί το σήμα 'ByteOp' με '1' για τις εντολές lb/sb και '0' για τις εντολές lw/sw. Η μόνη σύνδεση που έχει η βαθμίδα IFSTAGE με το υπόλοιπο κύκλωμα, είναι το ολισθημένο σήμα 'Immed_to_IF_EX' που εξέρχεται από το DECSTAGE και εισέρχεται στο σήμα εισόδου 'PC_Immed' του IFSTAGE. Έτσι, ολοκληρώνεται το Datapath χωρίς όμως τα σήματα ελέγχου που θα έρθουν από το Control στο τελικό στάδιο. Για την υλοποίηση της βαθμίδας του Datapath στο περιβάλλον του Xilinx, χρησιμοποιήθηκε η διαδικασία του Port Mapping για την κατάλληλη ένωση των κυκλωμάτων που παράχθηκαν.

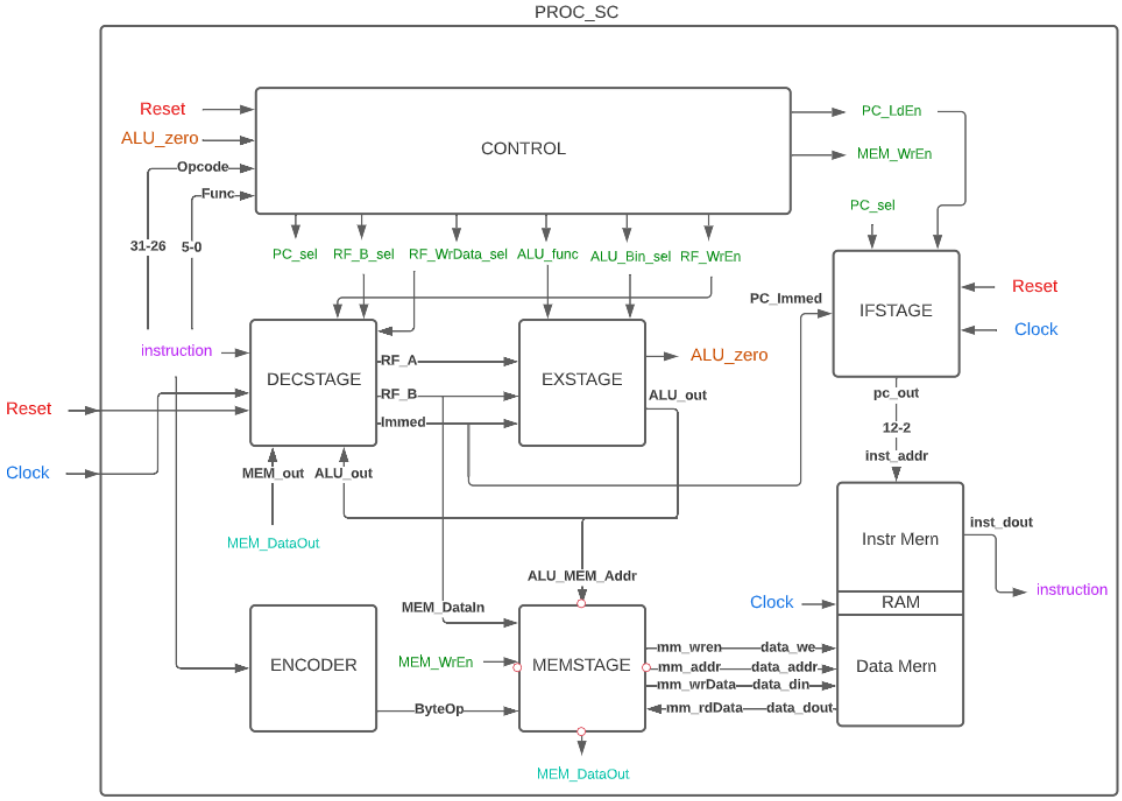
Control

Για το κομμάτι της βαθμίδας του Control βασιστήκαμε πάνω στον πίνακα που μας δόθηκε στη σελίδα 9 της εκφώνησης και θα αναλύθεί πως αξιοποιήθηκε. Για την υλοποίηση της βαθμίδας του Control

εκμεταλλευόμαστε τον κωδικό (Opcode) κάθε εντολής. Σύμφωνα με τον πίνακα, κάθε κωδικός (Opcode) είναι ξεχωριστός για κάθε εντολή ,εκτός της ειδικής περίπτωσης του κωδικού (Opcode) ’100000’ ,οποίος αναφέρεται στην πράξη που θα υλοποιήσει η μονάδα αριθμητικών και λογικών πράξεων (Arithmetic Logic Unit ή ALU) και έχει άλλο ένα σήμα ελέγχου 6-bits (Func) .Κάθε ξεχωριστό σήμα ελέγχου 6-bits (Func) πραγματοποιεί την αντίστοιχη πράξη στον πίνακα. Από κει και πέρα για να υλοποιηθεί κάθε εντολή χρειάζεται να γίνουν ορισμένες πράξεις ,όπως περιγράφει και ο πίνακας. Για τις πράξεις αυτές κάθε φορά πρέπει να ενεργοποιείται το κατάλληλο σήμα ελέγχου. Συμπερασματικά, αρχικά από τον κωδικό (Opcode) και στη συνέχεια από τα απαραίτητα σήματα ελέγχου πραγματοποιούνται όλες οι παραπάνω εντολές. Για την υλοποίηση της βαθμίδας του Control στο περιβάλλον του Xilinx χρησιμοποιήθηκε η τεχνική του case-then και του if-else .

Processor (PROC_SC)

Η βαθμίδα αυτή αποτελεί το top-Level της εργασίας και συνδέει το Datapath, το Control και την μνήμη που αποτελούν τον επεξεργαστή. Όσον αφορά την μνήμη RAM, το σήμα εισόδου 11bits διεύθυνσης εντολής συνδέεται με την έξοδο του IFSTAGE ‘pc_out’ η οποία όμως έχει 32bits πλάτος, οπότε επιλέγουμε τα 12-2 bits για να γίνει η σωστή διευθυνσιοδότηση της μνήμης. Οι άλλες τρεις εισόδοι της μνήμης, είναι η διεύθυνση των δεδομένων (‘data_addr’), η σημαία (flag) ενεργοποίησης εγγραφής στη μνήμη (’ data_we’) και τα δεδομένα που θα εγγραφούν στη μνήμη (‘data_din’), οι οποίες αρχικοποιούνται από τις εξόδους του MEMSTAGE. Οι έξοδοι της RAM ‘inst_dout’ και ‘data_dout’ μας δίνουν την εντολή (‘Instr’) και τα δεδομένα (‘MEM_DataOut’) που διαβάστηκαν από την μνήμη αντίστοιχα. Επίσης, η έξοδος ‘data_dout’ συνδέεται με την είσοδο ‘MM_RdData’ του MEMSTAGE. Όλες οι άγνωστες εισόδοι του DATAPATH πλέον δίνονται από το CONTROL, και το CONTROL με τη σειρά του δέχεται το σήμα ‘ALU_zero’ από το DATAPATH, και τα σήματα ‘Opcode’ και ‘Func’ από την εντολή που εξέρχεται από τη μνήμη RAM. Για την υλοποίηση της τελικής βαθμίδας του Επεξεργαστή(Processor) στο περιβάλλον του Xilinx χρησιμοποιήθηκε η διαδικασία του Port Mapping για την κατάλληλη ένωση των κυκλωμάτων που παράχθηκαν.



Πρόγραμμα αναφοράς #1:

00: addi r5, r0, 8

04: ori r3, r0, 0xABCB

08: sw r3, 4(r0) // γράφει στην διεύθυνση 0x4 => 0x404 την τιμή 0x0000ABCD

0C: lw r10, -4(r5) // διαβάζει από την διεύθυνση 0x4 => 0x404 την τιμή 0x0000ABCD

10: lb r16, 4(r0) // διαβάζει byte από την διεύθυνση 0x4 => 0x404 την τιμή 0xFFFFFFCD

14: nand r4, r10, r16

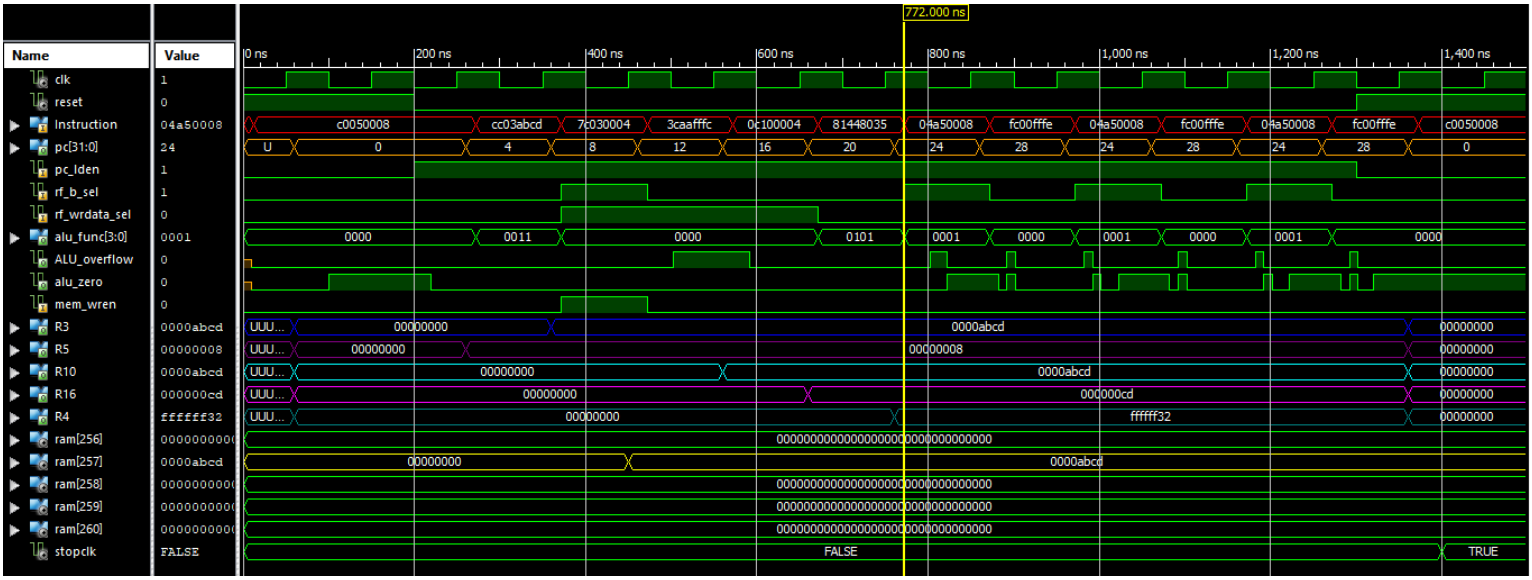
Πρόγραμμα αναφοράς #2:

00: bne r5, r5, 8 // αποτυχημένη διακλάδωση

04: b -2 // branch (PC=04 + 4 -2*4 = 00) infinite loop!

08: addi r1, r0, 1 // δεν θα εκτελεστεί

Η κυματομορφή που απεικονίζονται τα παραπάνω:



Στην αρχή ενεργοποιείται το σήμα Reset για δύο κύκλους και αφού απενεργοποιείται, αρχίζει να τρέχει το πρόγραμμα αναφοράς #1 που ξεκινάει με την εντολή Χ''C005_0008'' και τελειώνει με την εντολή Χ'' 8144_8035''. Από την εντολή Χ'' 04A5 0008'' και μετά τρέχει το πρόγραμμα αναφοράς #2 έως ότου να ενεργοποιηθεί ξανά το σήμα Reset ώστε να τερματίσει το Infinity loop. Στις περισσότερες εξόδους θα παρατηρήσουμε μία μικρή καθυστέρηση ή και την τιμή ''UU'' καθώς έχουμε χρησιμοποιήσει την εντολή after σε ορισμένα κυκλώματα ώστε να απεικονίσουμε τη ρεαλιστική λειτουργία του επεξεργαστή.