ΠΟΛΥΤΕΧΝΕΙΟ
ΚΡΗΤΗΣ

Αναφορά μαθήματος ΗΡΥ302

Εργασία #1: Σχεδίαση επεξεργαστή ενός κύκλου

Γεώργιος Φραγγιάς 2018030086
gfrangias@isc.tuc.gr

Διδάσκων Καθηγητής:
Σωτήριος Ιωαννίδης

Υπεύθυνος Εργαστηρίου:
Κυπριανός Παπαδημητρίου

Πολυτεχνειο Κρητης

Τμημα Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Απρίλιος 2021

Σκοπός εργαστηριακής άσκησης

Ο σκοπός της συγκεκριμένης εργαστηριακής άσκησης είναι να κατανοηθεί, να σχεδιαστεί και να υλοποιηθεί ένας επεξεργαστής ενός κύκλου. Πέρα από τον επεξεργαστή, υλοποιήθηκε και ένα module RAM μνήμης το οποίο συνδέθηκε και εξετάστηκε μαζί με τον επεξεργαστή.
Λόγω της πολυπλοκότητας του εγχειρήματος ήταν εξ αρχής γνωστό πως ακόμη και τα πιο μικρά κομμάτια της σχεδίασης έπρεπε να έχουν ελεγχθεί εξαντλητικά, ώστε να μπορούν να συμπεριληφθούν στο σύνολο του επεξεργαστή. Έτσι η αποσφαλμάτωση της $3^{ης}$ φάσης της εργασίας ήταν πολύ πιο εύκολη. Επίσης, αναμενόταν ότι πολλές λειτουργίες του επεξεργαστή θα εκτελούνταν ασύγχρονα (single-cycle).

Προεργασία

Πριν από την υλοποίηση της εργασίας ήταν απαραίτητη η περαιτέρω εξοικείωση με το εργαλείο σχεδίασης Xilinx ISE(14.7) και με την VHDL. Χρειάστηκε εντρύφηση στο εργαστηριακό υλικό του μαθήματος «Προχωρημένη Λογική Σχεδίαση». Επιπλέον, ειδικότερα για την σχεδίαση και την καλύτερη κατανόηση των ενοτήτων `DATAPATH` και `PROC_SC` χρειάστηκε η σχεδίασή τους σε block diagrams με την βοήθεια του γνωστού ανοικτού λογισμικού DIA. Τα block diagrams παρατίθενται στην περιγραφή της άσκησης παρακάτω.

Περιγραφή

$1^η$ Φάση της Άσκησης 1

Στην $1^η$ φάση της άσκησης σχεδιάστηκε μία απλή Αριθμητική και Λογική Μονάδα(`ALU`) και ένα Αρχείο 32 Καταχωρητών(`RF`).
Η `ALU` δέχεται 3 εισόδους, δύο τελεστέους 32-bits(`A` και `B`) έναν τελεστή 4-bits(`Op`). Οι πράξεις των τελεστών περιγράφονται στην εκφώνηση. Οι έξοδοι της `ALU` είναι το αποτέλεσμα της πράξης(`Out`) και τα σήματα ελέγχου. Τα σήματα αυτά ενεργοποιούνται, μόνο για τις πράξεις της πρόσθεσης και της αφαίρεσης, και περιγράφονται στην εκφώνηση.
Ο `RF` περιέχει 32 καταχωρητές χωρητικότητας 32-bits ο καθένας. Ο `RF` έχει την δυνατότητα να «διαβάσει» δύο καταχωρητές ασύγχρονα και να «γράψει» έναν καταχωρητή σύγχρονα. Η επιλογή του καταχωρητή που θα γραφτεί γίνεται στο module `decoder_5to32` το οποίο λαμβάνει την διεύθυνση σε μορφή ακεραίου και εξάγει μια 32-bits γεμάτο μηδενικά και μόνο ένα '1' στην διεύθυνση αυτή. Ο καταχωρητής `R0` αρχικοποιείται στην τιμή '0' και είναι αδύνατον να πανωγραφτεί αργότερα. Τελικά, χρησιμοποιήθηκε η είσοδος `RST`, ώστε όταν ενεργοποιείται να αρχικοποιούνται στην πρώτη ακμή του ρολογιού όλοι οι καταχωρητές στο 0 και να μην έχουν άχρηστη πληροφορία.

$2^η$ Φάση της Άσκησης 1

Η $2^η$ φάση αποτελείται από 4 στάδια (`IFSTAGE, DECSTAGE, EXSTAGE, MEMSTAGE`).

Το στάδιο ανάκλησης εντολών(`IFSTAGE`) είναι υπεύθυνο για την ανάκληση των εντολών από την μνήμη. Αυτό γίνεται είτε σειριακά, αυξάνοντας κατά 4 την τιμή του μετρητή προγράμματος, είτε με διακλάδωση στην διεύθυνση που δίνει η άνευ προθέματος άμεση τιμή. Η μνήμη(`RAM`) δέχεται σαν είσοδο τα 12 έως 2 bits του μετρητή προγράμματος, ώστε να λάβη είσοδο με διεύθυνση λέξης και όχι byte.

Το στάδιο αποκωδικοποίησης εντολών(DECSTAGE) είναι υπεύθυνο για την κωδικοποίηση της κάθε εντολής και την χρήση της για εγγραφές και προσπελάσεις του αρχείου καταχωρητών, όπως και την επέκταση των άμεσων(immediate) τιμών. Η καταχώρηση γίνεται είτε από την μνήμη είτε από αποτέλεσμα της ALU. Η άμεση τιμή επεκτείνεται με 4 διαφορετικούς τρόπους οι οποίοι ορίστηκαν ως:

00	πλήρωση με μηδενικά
01	πλήρωση με μηδενικά και 16 ολισθήσεις αριστερά
10	επέκταση προσήμου
11	επέκταση προσήμου και 2 ολισθήσεις αριστερά

Το στάδιο εκτέλεσης(EXSTAGE) είναι υπεύθυνο για την εκτέλεση των εντολών που σχετίζονται με τις πράξεις της ALU. Εκεί εκτελείται μία πράξη μεταξύ του ενός καταχωρητή(RF_A) είτε με τον άλλο καταχωρητή(RF_B), είτε με την άμεση(immediate) τιμή.

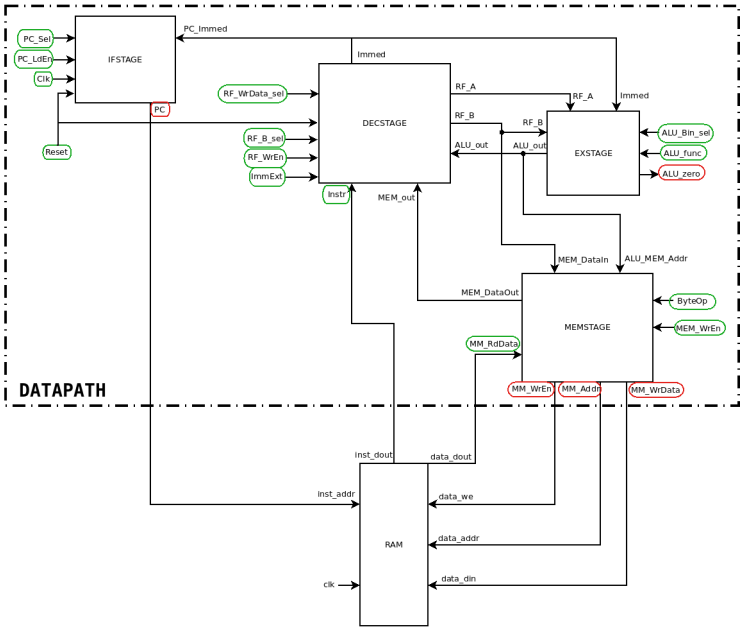
Το στάδιο μνήμης(MEMSTAGE) είναι υπεύθυνο για την καταχώρηση και την προσπέλαση της μνήμης(RAM). Στην εργασία γίνεται η παραδοχή πως η RAM χωρίζεται σε δύο τμήματα:

- τμήμα εντολών(0-1023 θέσεις λέξεων)
- τμήμα δεδομένων(1024-2047 θέσεις λέξεων)

Με αυτή την λογική η διεύθυνση που δέχεται το στάδιο μνήμης θα πρέπει να αυξηθεί κατά $(4096)_{dec}$ και όχι $(1024)_{dec}$. Έτσι αποθηκεύοντας στην θέση '0' θα γίνεται αποθήκευση στην θέση λέξης 1024 της μνήμης και αντίστοιχα η '4' στην θέση λέξης 1025 κ.ο.κ.

3η Φάση της Άσκησης 1

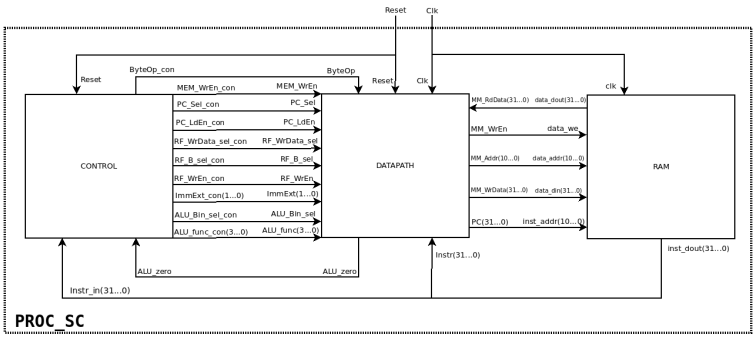
Η 3η φάση της άσκησης αποτελείται από 3 νέες ενότητες(DATAPATH, CONTROL, PROC_SC). Ο διάδρομος δεδομένων DATAPATH ενώνει όλα τα στάδια της 2ης φάσης μεταξύ τους όπως φαίνεται στο παρακάτω διάγραμμα βαθμίδων (είσοδοι-πράσινο, έξοδοι-κόκκινο):



Διάγραμμα βαθμίδων του DATAPATH μαζί με την RAM

Η ενότητα ελέγχου `CONTROL` λαμβάνει ως εισόδους την εκάστοτε εντολή και το σήμα ελέγχου `ALU_zero`(χρησιμεύει στον έλεγχο των εντολών `beq, bne`) και εξάγει σχεδόν όλα τα σήματα ελέγχου που χρειάζονται για την ορθή λειτουργία της ενότητας `DATAPATH`.

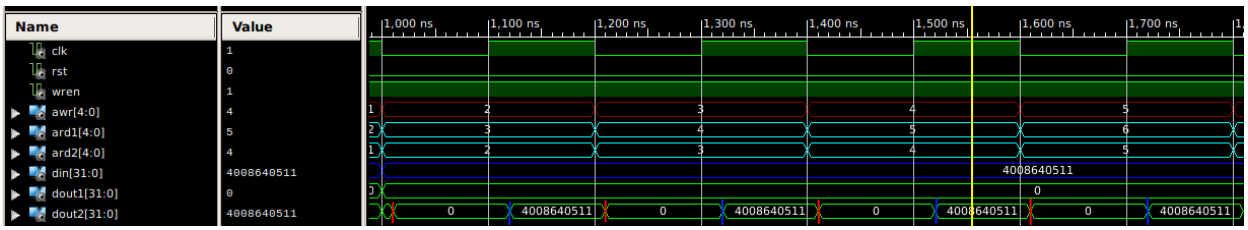
Ο επεξεργαστής `PROC_SC` είναι η ιεραρχικά υψηλότερη ενότητα και ενώνει τις ενότητες `DATAPATH, CONTROL` και `RAM`. Δέχεται σαν είσοδο μόνο τα σήματα επανεκκίνησης και ρολογιού. Η `RAM` αρχικοποιείται με εντολές από ένα αρχείο κειμένου το οποίο περιέχει σε σειρά όλες τις εντολές που θα εκτελεστούν. Η σύνδεση των ενοτήτων είναι φαίνεται στο παρακάτω διάγραμμα βαθμίδων:



Διάγραμμα βαθμίδων του PROC_SC

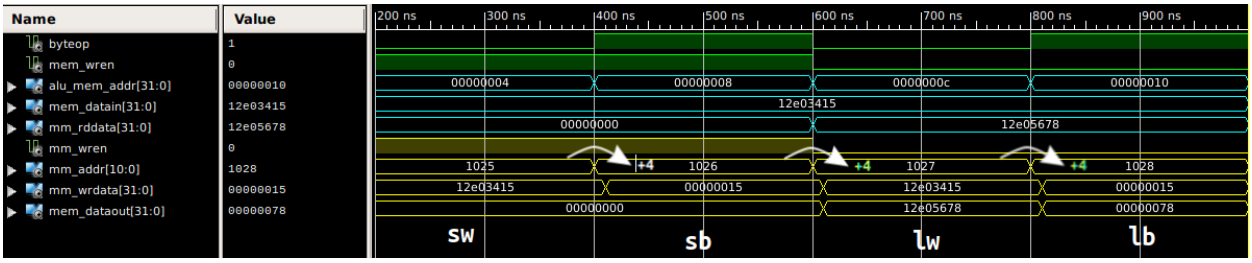
Κυματομορφές-Προσομοίωση

Στην παρακάτω κυματομορφή φαίνεται η βασική λειτουργία του αρχείου καταχωρητή. Φαίνεται πως ταυτόχρονα γράφεται και διαβάζεται(στην δεύτερη έξοδο) ένας καταχωρητής την φορά. Ο καταχωρητής που διαβάζεται στην πρώτη έξοδο δεν προλαβαίνει ποτέ να διαβάσει την τιμή που γράφει. Στο διάγραμμα με κόκκινες γραμμές φαίνονται οι χρονικές στιγμές που αρχίζει το διάβασμα(ασύγχρονα) και με μπλε γραμμές οι στιγμές του γραψίματος(σύγχρονα). Αυτό είναι πολύ σημαντικό για να εκτελούνται οι εντολές σε μόνο ένα κύκλο ρολογιού. Έτσι γράφεται το αποτέλεσμα μόνο στην θετική ακμή του ρολογιού όταν έχουν γίνει οι απαραίτητες πράξεις.



Κυματομορφή αρχείου καταχωρητών

Στην επόμενη κυματομορφή παρουσιάζεται ένα μέρος της προσομοίωσης του σταδίου μνήμης(`MEMSTAGE`). Φαίνεται πως εκτελούνται οι εντολές `sw, sb, lw` και `lb`. Παράλληλα αυξάνεται η μετατόπιση που δίνεται από την εντολή κατά 4 bytes με αποτέλεσμα η διεύθυνση λέξης να αυξάνεται κατά 1 σε κάθε εντολή.



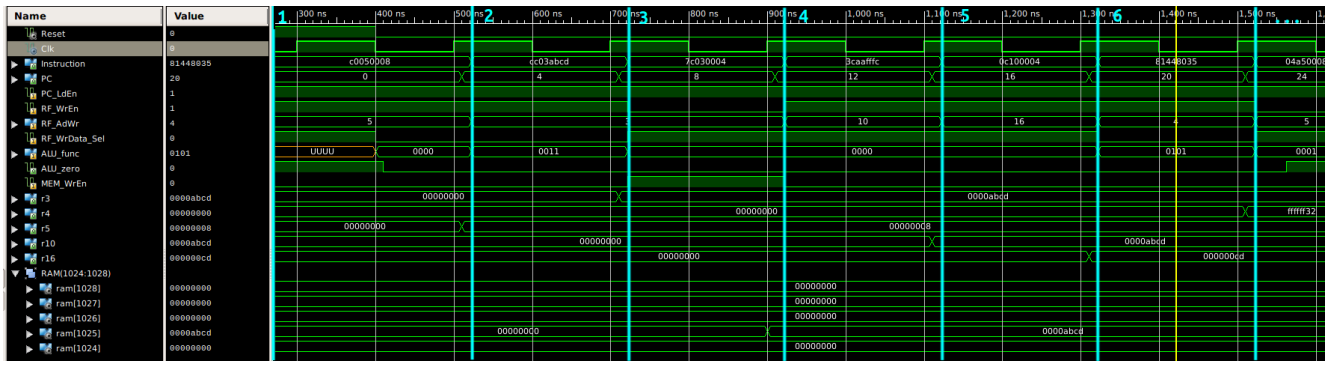
Κυματομορφή σταδίου μνήμης

Τέλος, παρουσιάζεται η κυματομορφή του επεξεργαστή ενός κύκλου `PROC_SC`. Σε ένα αρχείο κειμένου αντίστοιχο του `rom.data` το οποίο έχει ονομαστεί `report.data` έχουν γραφτεί οι εντολές του πρώτου και δεύτερου σετ εντολών. Στο πρώτο σετ εκτελούνται τα παρακάτω:

```
addi r5, r0, 8
ori r3, r0, ABCD
sw r3, 4(r0)
lw r10,-4(r5)
lb r16,4(r0)
nand r4,r10,r16
```

Στις εντολές παρατηρούνται τα εξής με την σειρά:

1. καταχωρείται το $(8)_{hex}$ στον `R5`
2. καταχωρείται το $(ABCD)_{hex}$ στον `R3`
3. καταχωρείται από τον `R3` το $(ABCD)_{hex}$ στην διεύθυνση λέξης 1025 της μνήμης
4. καταχωρείται στον `R10` η τιμή της διεύθυνσης 1025 δηλαδή $(ABCD)_{hex}$
5. καταχωρείται στον `R16` ένα byte από την διεύθυνση 1025 δηλαδή $(CD)_{hex}$
6. γίνεται `NAND` μεταξύ `R10` και `R16` και το αποτέλεσμα γράφεται στον `R4`

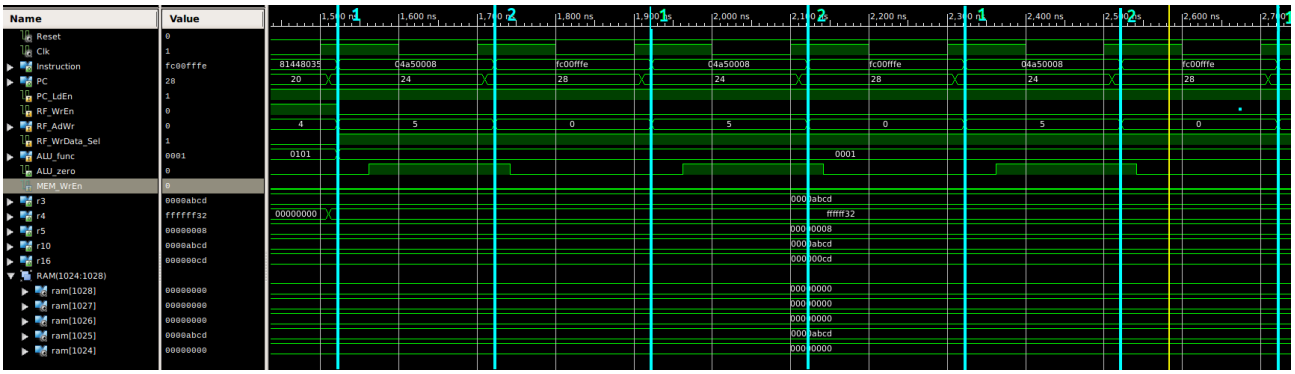


Κυματομορφή πρώτου σετ εντολών

Το δεύτερο σετ εντολών είναι το εξής:

```
bne r5,r5,8
b -2
addi r1,r0,1
```

Αυτό το σετ καταλήγει σε ατέρμονο βρόχο. Η πρώτη εντολή εκτελεί διακλάδωση στην θέση 8(τρίτη εντολή) μόνο όταν R5≠R5 το οποίο δεν συμβαίνει ποτέ. Άρα ο PC συνεχίζει με την εκτέλεση της δεύτερης εντολής. Αυτή κάνει διακλάδωση στην θέση $4+4+4\cdot(-2)=0$ δηλαδή στην πρώτη εντολή. Έτσι εκτελούνται αλλεπάλληλα οι εντολές 1 και 2 όπως φαίνεται στην κυματομορφή:



Κυματομορφή δεύτερου σετ εντολών

Συμπεράσματα

Μετά από την υλοποίηση αυτής της άσκησης έγιναν καλά κατανοητές οι διάφορες ενότητες του επεξεργαστή και ο ρόλος τους σαν οργανωμένο σύνολο. Αυτές οι ενότητες είχαν μελετηθεί σε βασικό επίπεδο στα μαθήματα «Προχωρημένη Λογική Σχεδίαση» και «Ψηφιακοί Υπολογιστές», αλλά για πρώτη φορά δημιουργήθηκαν εξ αρχής και ενώθηκαν σε ένα εργαστήριο. Ο επεξεργαστής μπορεί στην αρχή να φαίνεται περίπλοκος και δυσνόητος όμως με την μέθοδο του «διαίρει και βασίλευε» μπορεί να κατανοηθεί σε βάθος.

Κώδικας

Σε αυτό το σημείο παρατίθενται σημαντικά αποσπάσματα κώδικα από το σύνολο της άσκησης.

Το παρακάτω απόσπασμα είναι ο υπολογισμός της υπερχείλισης στην ALU. Γενικά ισχύει ότι αν γίνεται πρόσθεση μεταξύ δύο ομόσημων αριθμών και το αποτέλεσμα είναι διαφορετικού προσήμου έχει γίνει υπερχείλιση. Στην αφαίρεση, αν οι δύο τελεστέοι είναι ετερόσημοι και το αποτέλεσμα έχει διαφορετικό πρόσημο από τον πρώτο τελεστέο τότε έχει γίνει υπερχείλιση.

```
--Calculating the overflow value using the carry out
--and the signs of A and B
Ovf <= '1' after 10ns when (Op = x"0" and A(31) = B(31) and tmp33(31) /= A(31))
    or (Op = x"1" and A(31) /= B(31) and tmp33(31) /= A(31)) else '0' after 10ns;
```

Στο επόμενο απόσπασμα παρατίθεται η βασική λειτουργία του αποκωδικοποιητή του αρχείου καταχωρητών. Έχει χρησιμοποιηθεί μια τεχνική αριστερής ολίσθησης. Ξεκινώντας με ένα 32-bits διάνυσμα μηδενικών με ένα μόνο '1' στο LSB γίνεται μετά αριστερή λογική ολίσθηση όσες θέσεις υπαγορεύει ο ακέραιος **Awr**.

```
--A 32-bit vector with a '1' as the last bit
constant tmp_address: std_logic_vector(31 downto 0) := x"00000001";

begin

        --Shift left logical the '1' as many times as the unsigned value of the input
        Aout <= std_logic_vector(unsigned(tmp_address) sll to_integer(unsigned(Awr)));

end decoder;
```

Το επόμενο είναι από το στάδιο μνήμης και παρουσιάζεται η μετατροπή που υφίσταται η είσοδος-διεύθυνση **ALU_MEM_Addr** η οποία επεξηγήθηκε στην περιγραφή της άσκησης.

```
--add 1024 x 4 = 4096 to enter data part of RAM
added_1024 <= ALU_MEM_Addr + x"0000_1000";
--32-bit to 11-bit while multiplying by 4
MM_Addr <= added_1024(12 downto 2);
```

Είναι σημαντικό να παρατεθεί και ένα παράδειγμα από την ενότητα ελέγχου. Τα σήματα ελέγχου τα οποία διαχειρίζεται αυτή η ενότητα ελέγχονται ένα-ένα με βάση την εντολή η οποία ανακτάται από την μνήμη. Το συγκεκριμένο παράδειγμα είναι το σήμα **RF_WrEn** το οποίο είναι ανενεργό μόνο για τις εντολές **b, beq, bne, sb, sw**. Δηλαδή, είναι ανενεργό μόνο για τις εντολές που δεν χρειάζεται καταχώρηση στο αρχείο καταχωρητών.

```
--RF_WrEn is '0' only when there is no need to write in register RF[rd]
--So only for the instructions: 'b', 'beq', 'bne', 'sb' and 'sw'
RF_WrEn_con <= '0'
        when(sig_inst(31 downto 26) = "111111") or
                (sig_inst(31 downto 26) = "000000") or
                (sig_inst(31 downto 26) = "000001") or
                (sig_inst(31 downto 26) = "000111") or
                (sig_inst(31 downto 26) = "011111")
                else '1';
```