18/4/2019

3ο Εργαστήριο Οργάνωσης Υπολογιστών
ΟΛΟΚΛΗΡΩΣΗ ΚΑΙ ΠΡΟΣΟΜΟΙΩΣΗ DATAPATH ΕΠΕΞΕΡΓΑΣΤΗ ΕΝΟΣ ΚΥΚΛΟΥ (ΧΩΡΙΣ ΤΟ CONTROLPATH)
22/3/2019

Ομάδα LAB31239632

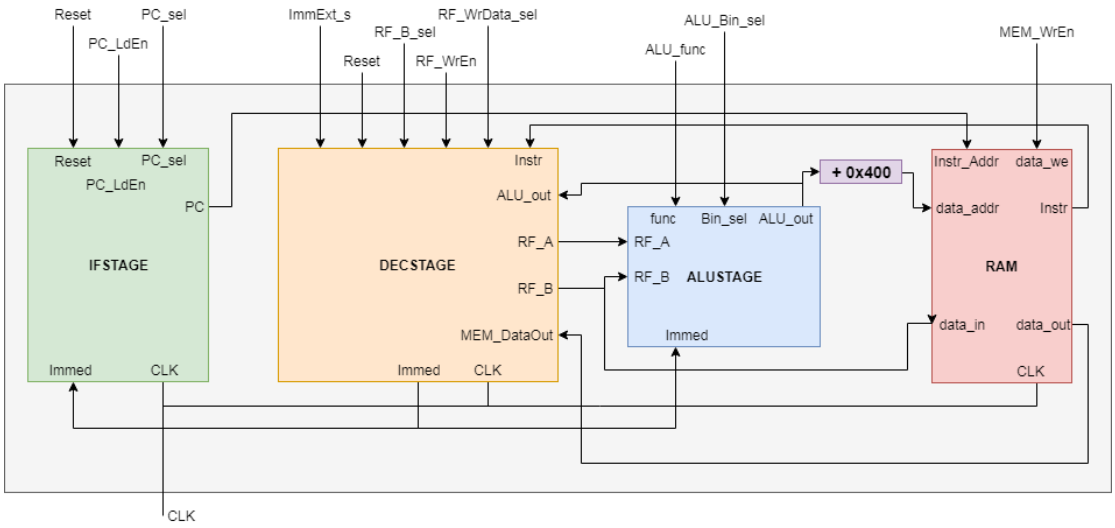
Αλέξανδρος Πουπάκης 2016030061
Βαγγέλης Κιούλος 2016030056

Σκοπός εργαστηριακής άσκησης

Στο εργαστήριο αυτό έπρεπε να ενώσουμε τα modules των προηγούμενων εργαστηρίων μεταξύ τους, δημιουργώντας έτσι το πλήρες και λειτουργικό datapath ενός απλού επεξεργαστή και να επιβεβαιώσουμε την ορθή λειτουργία του, εκτελώντας δυο απλά προγράμματα.

Προεργασία

Έχοντας έτοιμες τις επιμέρους βαθμίδες από τα προηγούμενα εργαστήρια, σχεδιάσαμε το παρακάτω block diagram που περιγράφει τη διασύνδεση τους.



Περιγραφή

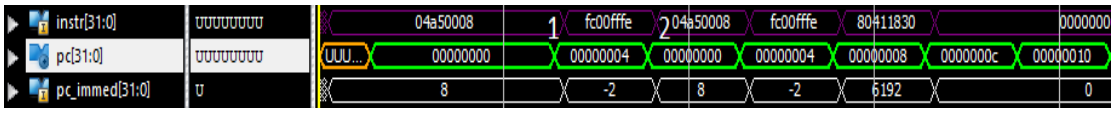
Υλοποιήσαμε την διασύνδεση των modules όπως φαίνεται παραπάνω σε ένα top module με όνομα Datapath. Για την πρόσθεση της σταθεράς 0x400 χρησιμοποιήθηκε από την βιβλιοθήκη IEEE.NUMERIC_STD ο τελεστής πρόσθεσης και συναρτήσεις μετατροπής από std logic vector σε unsigned integer και αντίστροφα.

Κυματομορφές-Προσομοίωση

Πρόγραμμα 1:

Παραπάνω, παρατηρούμε τις κυματομορφές που προέκυψαν από το πρώτο πρόγραμμα που εκτελέσαμε. Στην πρώτη σειρά (μωβ χρώμα), αναγράφονται οι εντολές πού εκτελούνται στο πρόγραμμα μας. Πρώτα, εκτελείται η εντολή “addi r5,r0,8”(c0050008), η οποία, τοποθετεί την τιμή “8”, στον καταχωρητή R5(1). Στην συνέχεια, εκτελούμε την εντολή “ori r3,r0,0xABCD” (cc03abcd), η εντολή αυτή, κάνει την λογική πράξη “or” με τελεστές το περιεχόμενο του καταχωρητή r0(δηλαδη το 0) και την τιμή “0xABCD” και αποθηκεύει το αποτέλεσμα(0000abcd) στον καταχωτητή r3. Στο παραπάνω σχήμα, με την κόκκινη γραμμή συμβολίζουμε τα περιεχόμενα του καταχωρητή και όπως βλέπουμε η τιμή “0000abcd”, έχει καταχωρηθεί στον καταχωρητή r3(2), μετά την εκτέλεση της εντολής. Έπειτα, εκτελείται η εντολή “sw r3,4(r0)”(7c030004) η οποία, αποθηκεύει την τιμή του καταχωρητή r3(δηλ. 0x0000abcd) στην διεύθυνση 0x404. Μετά την εκτέλεση της εντολής, παρατηρούμε ότι η τιμή “0000abcd” έχει αποθηκευθεί στην διεύθυνση 0x404(3). Στην συνέχεια, εκτελείται η εντολή “lw r10,-4(r5)”(3caaffcc) η οποία, διαβάζει από την διεύθυνση 0x404 και καταχωρεί την τιμή της στον καταχωρητή r10(4). Έπειτα, εκτελείται η εντολή “lb r16, 4(r0)”(0c100004), που διαβάζει ένα byte από την διεύθυνση 0x404 και την αποθηκεύει στον καταχωρητή r16. Παραπάνω, παρατηρούμε ότι μετά την εκτέλεση της εντολής, η τιμή “000000cd” έχει αποθηκευτεί στον καταχωρητή r16(5). Τέλος, εκτελείται η εντολή “nand r4,r16,r10” που εκτελεί την λογική πράξη “nand” με τελεστές τους καταχωρητές r16 και r10 και αποθηκεύει το αποτέλεσμα στον καταχωρητή r4(6).

Πρόγραμμα 2:



Στο παραπάνω σχήμα, βλέπουμε το αποτέλεσμα του δεύτερου προγράμματος. Στην πρώτη σειρά (μωβ χρώμα), εμφανίζονται οι εντολές που εκτελούνται με την σειρά που εκτελούνται. Πρώτα, εκτελείται η εντολή “bne r5,r5,8”(04a50008), η οποία συγκρίνει τον καταχωρητή r5 με τον εαυτό του και οι τιμές τους δεν είναι ίσες κάνει διακλάδωση στην διεύθυνση “8”. Στην συγκεκριμένη περίπτωση, η διακλάδωση θα είναι αποτυχημένη, οπότε, το πρόγραμμα θα εκτελέσει την ακριβώς επόμενη εντολή που άρα η τιμή του PC γίνεται “00000004” όπως βλέπουμε παραπάνω(1) και εκτελείται η εντολή “b -2”(fc00fff3) που κάνει διακλάδωση στην προηγούμενη εντολή, επομένως, η τιμή του PC γίνεται “00000000” (2). Με αυτό το πρόγραμμα, εκτελείτε ένα “infinite loop” οπότε, εκτελούνται μόνο αυτές οι δύο εντολές.

Παρατηρούμε ότι το πρόγραμμα δεν εκτελεί ατέρμονο loop όπως θα έπρεπε. Αυτό συμβαίνει διότι δεν υπάρχει το control για να δίνει συνεχώς τα κατάλληλα σήματα.

Συμπεράσματα

- Στα αρχεία ram.data, αποθηκεύουμε τα προγράμματα που θα εκτελέσει το datapath.
- Οι εντολές branch αλλάζουν την τιμή του καταχωρητή PC.
- Τα σήματα ελέγχου δίνονται μέσα από το testbench καθώς δεν υπάρχει το κομμάτι ελέγχου.

Κώδικας

Τέλος, προστέθηκε επιπλέων λογική για να γίνεται δυνατή η εκτέλεση των εντολών lb και sb. Η υλοποίηση φαίνεται στην παρακάτω εικόνα.

```
op_sb <= '1' when (std_match(op, "000111")) else '0';
rf_b_byte <= RF_B_raw and X"000000FF";

SB_MUX : MUX   Generic Map (N => 32)
               Port Map ( A => RF_B_raw,
                          B => rf_b_byte,
                          Sel => op_sb,
                          MUX_Out => RF_B);

op_lb <= '1' when (std_match(op, "000011")) else '0';
mem_out_byte <= MEM_out and X"000000FF";

LB_MUX : MUX   Generic Map (N => 32)
               Port Map ( A => MEM_out,
                          B => mem_out_byte,
                          Sel => op_lb,
                          MUX_Out => MEM_out_processed);
```

Για την υλοποίηση των εντολών, προστέθηκαν τα σήματα “op_sb” και “op_lb”, όταν τα σήματα αυτά είναι ‘1’, τότε θα αποθηκεύσουμε ή φορτώσουμε αντίστοιχα το τελευταίο byte της τιμής που θέλουμε. Η υλοποίηση προστέθηκε στην βαθμίδα DECSTAGE.