ΗΥ-225: Οργάνωση Υπολογιστών
Άνοιξη 2004

Τμ. Επ. Υπολογιστών
© Πανεπιστήμιο Κρήτης

Σειρά Ασκήσεων 10:
Το Datapath του Επεξεργαστή

Προθεσμία έως Κυριακή 2 Μαΐου, ώρα 23:59 (βδομάδα 8)

Σε αυτήν την άσκηση θα περιγράψετε και θα προσομοιώσετε σε Verilog το datapath του επεξεργαστή του μαθήματος για την υλοποίηση η οποία εκτελεί μία εντολή σε πολλαπλούς κύκλους ρολογιού. Η περιγραφή θα γίνει σε μορφή "structural", δηλαδή ενώνοντας μεταξύ τους έτοιμα δομικά στοιχεία που δίδονται στη βιβλιοθήκη. Το datapath δίδεται στο σχήμα που υπάρχει πιό κάτω --σε δύο κομάτια-- με όλα τα ονόματα των σημάτων που θα χρησιμοποιήσετε. Τα σήματα που έρχονται από επάνω είναι σήματα ελέγχου. Ο έλεγχος θα υλοποιηθεί στην επόμενη σειρά ασκήσεων --προς το παρόν, τα σήματα αυτά θα είναι απλώς είσοδοι στο δικό σας module.

Άσκηση 10.1: Περιγραφή του Datapath σε Verilog

Περιγράψτε σε Verilog το datapath που δίδεται στα σχήματα. Γράψτε την περιγραφή σας σαν ένα module, το "**`datapath`**", χρησιμοποιώντας τα ίδια ακριβώς ονόματα όπως στα σχήματα. Ονομάστε το αρχείο σας "**`datapath10.v`**". Οι είσοδοι και έξοδοι του module datapath θα πρέπει να είναι το ρολόι (clk), και όλα τα σήματα από και πρός τον έλεγχο που φαίνονται στο επάνω μέρος των σχημάτων. Δώστε ιδιαίτερη προσοχή στο σωστό ορισμό του πλάτους όλων των σημάτων, καθώς επίσης και στα άλλα θέματα που σημειώνονται πιό κάτω.

Τα έτοιμα δομικά στοιχεία που θα χρησιμοποιήσετε είναι καταχωρητές, πολυπλέκτες, ALU, μνήμη, και register file. Όλα αυτά ορίζονται στη βιβλιοθήκη που συνοδεύει αυτές τις ασκήσεις:

- Περιγραφή της βιβλιοθήκης δίδεται στο web.
- ο κώδικας της βιβλιοθήκης υπάρχει στο **`~hy225/verilog/lib/lib10.v`**

(α) Ορισμοί Συρμάτων:
Προσέξτε ότι κάθε καινούριο σύρμα πρέπει να ορίζεται σαφώς πριν χρησιμοποιηθεί. Αν π.χ. έχετε έναν πολυπλέκτη που η έξοδός του λέγεται "**ma**", η σύνταξη σε Verilog θα πρέπει να είναι:

```
wire [31:0] ma;
Mux2 #32 muxaddr (ma, pc, ALUout, IorD);
```

Αν δεν ορίσετε πρώτα το **ma**, μπορεί ο interpreter να μην παραπονεθεί, και να θεωρήσει το **ma** σύρμα 1 bit! Ομοίως, τα σύρματα που έρχονται από έξω απο το module σας πρέπει να οριστούν σαφώς με εντολές input ή output. Οι εντολές input και output ισοδυναμούν με τη wire. Δεν χρειάζεται να πείτε τίποτα παραπάνω από τα εξής για να ορίσετε τα σύρματα **IorD** και **op**:

```
input IorD;
output [5:0] op;
```

Στον πολυπλέκτη της δεύτερης εισόδου της ALU υπάρχει μία είσοδος, έστω "**const4**", που είναι ο σταθερός αριθμός 4. Παρ' ότι αυτή θα έπρεπε να μπορούσε να δηλωθεί μέσω wire/assign, όμως, λόγω κάποιου bug που σχετίζεται και με τη δική μας βιβλιοθήκη, δηλώστε την σαν (αρχικοποιημένο) reg:

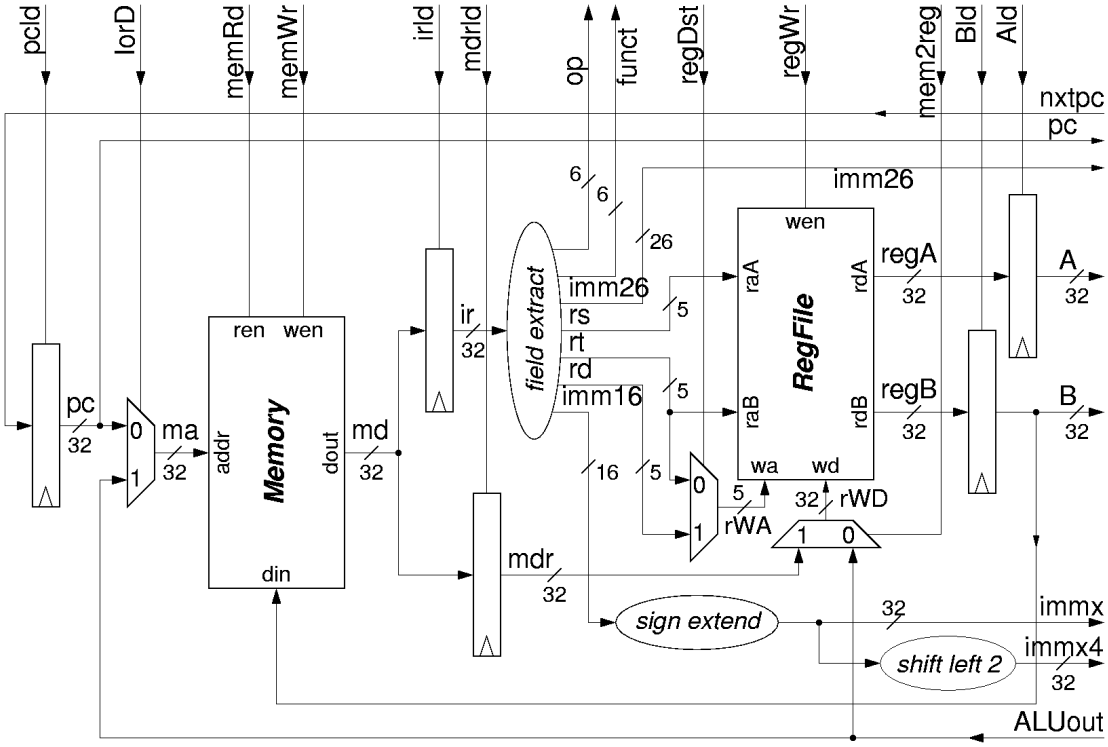
```
reg [31:0] const4;
initial const4 = 4;
```

(β) Εξαγωγή Πεδίων από bits:
Γιά το υποσύστημα "field extract" θα χρησιμοποιήσετε το μηχανισμό επιλογής bits της Verilog για να δημιουργήσετε καινούρια σύρματα με τα επιμέρους πεδία του **ir**. Υπενθυμίζεται ότι η σύνταξή είναι:

```
wire [4:0] rs;
assign rs = ir[25:21];
```

Αν ορίσετε πρώτα ένα σύρμα, με εντολή wire ή input ή output, θα το περιγράψετε μετά με μιάν εντολή assign όπως παραπάνω. Ειδικά στην περίπτωση της wire, όμως, μπορείτε να κάνετε και και τις δύο δουλειές με μία μόνο εντολή, ως εξής:

```
wire [4:0] rs = ir[25:21];
```



(γ) Επέκταση Προσήμου:
Το υποσύστημα "sign extend" κάνει επέκταση προσήμου, δηλαδή αντιγραφή του περισσότερο σημαντικού bit ενός αριθμού σε όλα τα επιπλέον bits ενός μεγαλύτερου (σε πλάτος bits) αριθμού. Παραδείγματος χάριν, ο αριθμός "`4'b0101`" γίνεται "`8'b00000101`", ενώ ο "`4'b1101`" γίνεται "`8'b11111101`". Για να το πετύχετε αυτό, θα χρησιμοποιήσετε τη συγκόλληση πεδίων (concatenation) της Verilog, που ενώνει μικρότερα σύρματα σε ένα μεγαλύτερο.

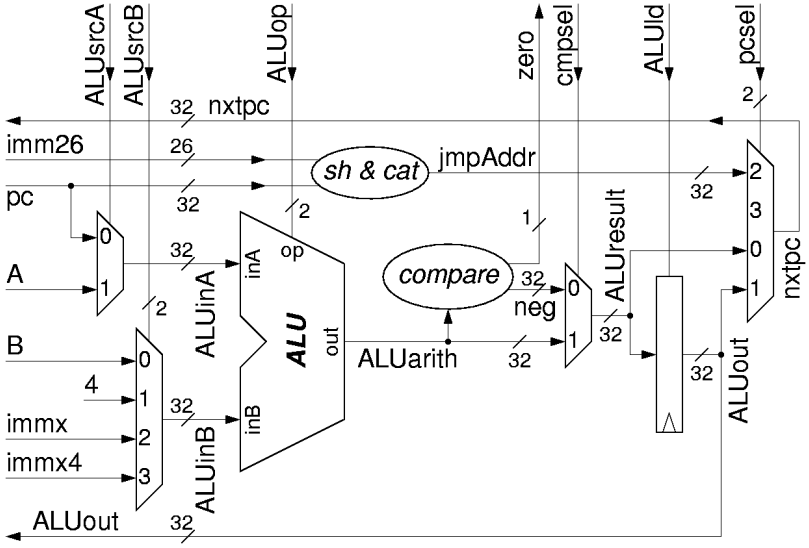
```
wire [3:0] unsigned;
wire [7:0] signed;
assign signed = {unsigned[3], unsigned[3], unsigned[3], unsigned[3],
                 unsigned[3:0]};
```

Έτσι θα γίνει η πράξη "hardwired", χωρίς καθυστερήσεις. Εναλλακτικά, μπορείτε να χρησιμοποιήσετε έναν πολυπλέκτη, που να ελέγχεται απο το περισσότερο σημαντικό bit του signed ("πληρώνοντας" την καθυστέρηση του πολυπλέκτη)....

(δ) Ολίσθηση:
Το υποσύστημα "shift left" που κάνει αριστερή ολίσθηση κατά σταθερό πλήθος bits (π.χ. 2 θέσεις), είναι πρακτικά η επιλογή όλων των λιγότερο σημαντικών bits του αριθμού εκτός από τα πρώτα δύο, και η δημιουργία ενός καινούριου σύρματος που τα έχει ως περισσότερο σημαντικά bits, και στα 2 LS bits του έχει 0:

```
wire [7:0] quantity;
wire [7:0] quantity4 = {quantity[5:0], 2'b00};
```

(ε) Συγκρίσεις:
Στην έξοδο της ALU υπάρχουν δύο μικρά κυκλώματα που αφορούν συγκρίσεις. Μία πύλη NOR 32 εισόδων (από τη βιβλιοθήκη) παράγει το σήμα "**zero**" που ενεργοποιείται όποτε η έξοδος της ALU είναι μηδενική. Το σήμα "**neg**" (negative, αρνητικό) είναι ένα σήμα 32 bits που ισούται με 000...001 όταν η έξοδος της ALU είναι αρνητική, αλλοιώς ισούται με μηδέν. Το σήμα αυτό θα το υλοποιείστε με επιλογές bits και συγκολλήσεις, χρησιμοποιώντας το bit προσήμου της εξόδου της ALU, δηλαδή το περισσότερο σημαντικό bit της.



Άσκηση 10.2: Πρώτος Ελεγχος του Datapath

Ελέγξτε το κύκλωμά σας ότι εκτελεί μερικές εντολές "με το χέρι". Για το σκοπό αυτό, θα ξεκινήστε με ένα test bench που βρίσκεται στην περιοχή του μαθήματος:

`~hy225/verilog/test/test10.v`

Το αρχείο αυτό δεν είναι έτοιμο να "τρέξει" --απλά περιέχει το "σκελετό" ενός test bench που θα φτιάξετε εσείς. Εκεί θα βρείτε οδηγίες γιά το πώς θα εκτελέσετε μερικούς κύκλους στο datapath σας, φορτώνοντας τη μνήμη με αρχικές τιμές και οδηγώντας τα σήματα ελέγχου για λίγους κύκλους.

Ο στόχος αυτής της σειράς ασκήσεων είναι να γράψετε το datapath, και να βεβαιωθείτε ότι δεν έχουν γίνει "τραγικά" λάθη -- **δεν** χρειάζεται να είναι τελείως σωστό, αφού αυτό θα είναι ένας από τους στόχους της επόμενης άσκησης.

Παραδώστε, όπως και στις προηγούμενες ασκήσεις, τον κώδικά σας "**datapath10.v**", το test bench "**test10.v**" όπως τελικά το διαμορφώσατε εσείς, και ένα χαρακτηριστικό στιγμιότυπο, "**signals10.jpg**", από το Signalscan της άσκησης 10.2 σε μορφή jpg, πακεταρισμένα στο αρχείο "**`ask10.tar`**", μέσω:

```
tar -cvf ask10.tar   datapath10.v test10.v signals10.jpg
~hy225/bin/submit 10
```

Up to the Home Page of CS-225

Last updated: 21 Apr. 2004, by M. Katevenis.