Εργαστήριο 4

Χειμερινό εξάμηνο 2016-2017

1. Στόχοι των επόμενων εργαστηρίων

Ο στόχος των επόμενων εργαστηρίων του μαθήματος της Οργάνωσης είναι ο σχεδιασμός, υλοποίηση και η επαλήθευση μέσω προσομοίωσης (simulation) μιας αρχιτεκτονικής που υλοποιεί ένα υποσύνολο των εντολών MIPS. Συγκεκριμένα, τα εργαστήρια θα υλοποιήσουν σταδιακά έναν επεξεργαστή MIPS ενός κύκλου και στην συνέχεια θα επεκτείνουν την αρχιτεκτονική πολλαπλών κύκλων με 5 στάδια διοχέτευσης (pipeline stages) με νέες εντολές. Εφόσον υπάρξει χρόνος και όρεξη από τους φοιτητές μας (που είμαι σίγουρος ότι υπάρχει) θα προχωρήσουμε για την υλοποίηση μέρους της ιεραρχίας μνήμης.
Για τον σκοπό αυτό θα χρησιμοποιήσετε την γλώσσα περιγραφής υλικού Verilog, και μια βιβλιοθήκη μονάδων (components). Η βιβλιοθήκη θα περιλαμβάνει και όλες τις απαραίτητες μονάδες για την υλοποίηση του τμήματος δεδομένων του επεξεργαστή, και συγκεκριμένα Μνήμη (memory), Αρχείο Καταχωρητών (Register File), και Αριθμητική-Λογική Μονάδα (ALU).

2. Υλοποίηση βασικών μονάδων του MIPS

Για εξάσκηση σε σχεδιασμό hardware με την Verilog θα ξεκινήσετε την δημιουργία της βιβλιοθήκης έτοιμων μονάδων υλοποιώντας 2 βασικές μονάδες που θα χρησιμοποιηθούν αργότερα στην υλοποίηση του MIPS.

a. Αριθμητική Λογική Μονάδα - ALU

```
module ALU (out, zero, inA, inB, op);
```

Αριθμητική/λογική μονάδα για χρήση με εντολές MIPS. Εκτελεί πρόσθεση ή αφαίρεση ακεραίων (με την χρήση συμπληρώματος ως προς δύο, 2's complement), την πράξη Set On Less (slt) και τις λογικές πράξεις AND/OR/NOR.

Παράμετροι - Θύρες:

N: Παράμετρος: το πλάτος της Αριθμητικής, Λογικής μονάδας σε bits.

out: Output (N bits): έξοδος δεδομένων.

zero: Output (1 bit): 1 όταν η έξοδος out είναι 0.

inA: Input (N bits): πρώτη είσοδος δεδομένων.

inB: Είσοδος (N bits): δεύτερη είσοδος δεδομένων

Op: Είσοδος (4 bits): είσοδος επιλογής πράξης. Οι παρακάτω αριθμοί είναι εκφρασμένοι στο δεκαδικό σύστημα.

- 0 bitwise AND: out = inA & inB.
- 1 bitwise OR: out = inA | inB.
- 2 πρόσθεση: out = inA + inB.
- 6 αφαίρεση: out = inA – inB
- 7 Set On Less Than: out = ((inA < inB)?1:0)
- 12 NOR: out = ~(inA | inB).

Παράδειγμα χρήσης ALU των 32 bits:

```
ALU #(parameter N=32) ALUInst (ALUOut, Zero, ALUinA,
ALUinB, ALUop);
```

Μην ξεχάσετε να ορίσετε την έξοδο **out** του module σε περίπτωση που η είσοδος **Op** πάρει κάποια τιμή που δεν βρίσκεται στον παραπάνω πίνακα. Σε αυτήν την περίπτωση η έξοδος μπορεί να είναι, για παράδειγμα, don't care X.

b. Αρχείο Καταχωρητών – Register File

```
module RegFile (clock, reset, raA, raB, wa, wen, wd, rdA, rdB)
```

Αρχείο 32 Καταχωρητών, μεγέθους 32-bit έκαστος, με δυο θύρες ανάγνωσης και μια εγγραφής.
Οι είσοδοι clock και reset θα πρέπει να εμφανίζονται σε όλα τα ακολουθιακά κυκλώματα όπως Flip-Flops, Καταχωρητές, Μηχανές Πεπερασμένων Καταστάσεων (Finite State Machines, FSMs), κοκ. Το ρολόι σε ακολουθιακά κυκλώματα αποτελεί τον τρόπο χρονισμού του κυκλώματος έτσι ώστε οτιδήποτε αλλαγές συμβαίνουν στο κύκλωμα να γίνονται μόνο την χρονική στιγμή κατά την οποία το ρολόι αλλάζει τιμή. Στην περίπτωση αυτή το κύκλωμα ονομάζεται ακμοπυροδότητο (edge-triggered), και όταν η αλλαγή στην έξοδο του κυκλώματος γίνεται κατά την μετάβαση του ρολογιού από 0 σε 1 ονομάζεται θετικά ακμοπυροδότητο (positive edge-triggered). Όταν η αλλαγή στην έξοδο του κυκλώματος γίνεται όταν το ρολόι αλλάζει από 1 σε 0, το κύκλωμα ονομάζεται αρνητικά ακμοπυροδότητο (negative edge-triggered). Σχεδόν σε όλες τις περιπτώσεις που θα εξετάσουμε το ακολουθιακό κύκλωμα είναι θετικά ακμοπυροδότητο. Τυχόν εξαιρέσεις θα επισημαίνονται ανά περίπτωση.

Το ασύγχρονο reset προκαλεί την **άμεση** μεταγωγή των Flip-Flops του ακολουθιακού κυκλώματος σε μια γνωστή αρχική κατάσταση (συνήθως 0), χωρίς να περιμένουμε την ακμή του ρολογιού. Για όσον χρονικό διάστημα το reset είναι 0 (active), όλοι οι καταχωρητές παραμένουν στο 0, ανεξάρτητα από τις ακμές του ρολογιού (reset is active low). Η κανονική λειτουργία του κυκλώματος αρχίζει όταν το reset γίνει inactive, ανεβεί δηλαδή στην τιμή 1.

Θύρες:

clock: Είσοδος. Ρολόι συγχρονισμού. Όλες οι εγγραφές σε καταχωρητές θα πρέπει να λαμβάνουν χώρα στην αρνητική ακμή του ρολογιού (negative edge).

reset: Είσοδος. Ασύγχρονο reset. Active low.

raA: Είσοδος (5 bits): διεύθυνση καταχωρητή προς ανάγνωση από την πρώτη θύρα.

raB: Είσοδος (5 bits): διεύθυνση καταχωρητή προς ανάγνωση από την δεύτερη θύρα.

wa: Είσοδος (5 bits): διεύθυνση εγγραφής.

wen: Είσοδος (1 bit): enable signal γιά εγγραφή δεδομένων.

wd: Είσοδος (32 bits): είσοδος δεδομένων προς εγγραφή.

rdA: Εξοδος (32 bits): έξοδος δεδομένων ανάγνωσης της πρώτης θύρας.

rdB: Εξοδος (32 bits): έξοδος δεδομένων ανάγνωσης της δεύτερης θύρας.

Το Αρχείο Καταχωρητών έχει ασύγχρονη προσπέλαση για ανάγνωση (read), όπου μετά την εφαρμογή του αριθμού του καταχωρητή (**raA** και **raB**) εμφανίζονται τα δεδομένα. Για την εγγραφή (write) στο Αρχείο Καταχωρητών χρειάζεται εκτός από τον αριθμό του καταχωρητή **wa** και την είσοδο δεδομένων **wd**, το σήμα enable **wen** και η αρνητική ακμή του ρολογιού clock. Επίσης το reset πρέπει να είναι 1 (inactive).

i. Εγγραφή στο Αρχείο Καταχωρητών

Η εγγραφή γίνεται βάση του σήματος wen, με την ακόλουθη διαδικασία:

- Το **wen** γίνεται 1.
- Η διεύθυνση εγγραφής, **wa** , και τα δεδομένα, **wd** , πρέπει να είναι ήδη σταθερά.
- Το ρολόι μεταβαίνει από 1 σε 0 (αρνητική ακμοπυροδότηση).

ii. Ανάγνωση από το Αρχείο Καταχωρητών

Η ανάγνωση γίνεται μόνο βάση της διεύθυνσης του καταχωρητή, χωρίς δηλαδή σήμα ενεργοποίησης της ανάγνωσης.

Κανονικά, σε ένα πραγματικό σύστημα, οι καταχωρητές θα πρέπει να αρχικοποιούνται μέσω του κώδικα μηχανής του προγράμματος. Για αυτό το εργαστήριο, οι καταχωρητές αρχικοποιούνται από το testbench όπως θα δούμε στην επόμενη παράγραφο.

c. Διασύνδεση των δύο στοιχείων

Το τελευταίο τμήμα του εργαστηρίου σας ζητάει να συνδέσετε την ALU και το αρχείο καταχωρητών ως εξής:

- οι δύο είσοδοι της ALU (inA και inB) θα πρέπει να συνδεθούν με τις δύο εξόδους δεδομένων ανάγνωσης του αρχείου καταχωρητών,
- η έξοδος της ALU (out) θα πρέπει να συνδεθεί με την είσοδο δεδομένων εγγραφής του αρχείου καταχωρητών,
- τα υπόλοιπα σήματα (op, wen, …) θα θέτονται από το testbench.

Με άλλα λόγια το αρχείο καταχωρητών τροφοδοτεί την ALU με εισόδους (operands) και δέχεται πίσω το αποτέλεσμα των πράξεων της ALU.

3. Προσομοίωση για Επαλήθευση Ορθής Λειτουργίας

Για την επαλήθευση του τμήματος δεδομένων του επεξεργαστή παρέχετε ένα πρότυπο σκελετού πλαισίου δοκιμής, το οποίο θα πρέπει να τροποποιήσετε (αρχείο testbench.v). Στο πρότυπο αυτό αρχείο εμπεριέχονται οδηγίες για την εμφάνιση της μονάδας του επεξεργαστή, την αρχικοποίηση του καταχωρητή, τον ορισμό του ρολογιού και την εφαρμογή των σημάτων ελέγχου ανά κύκλο. Η αρχικοποίηση του αρχείου καταχωρητών θα γίνει με απευθείας ανάθεση της τιμής σε κάθε καταχωρητή.

Για την επαλήθευση του κυκλώματος σας, σας προτείνεται να χρησιμοποιήσετε το παράθυρο κυματομορφών, ή και τις εντολές **$display**, και **$monitor** της Verilog, οι οποίες εμφανίζουν ή παρακολουθούν ενεργά την τιμή των σημάτων αντίστοιχα. Έτσι, μπορείτε να παρακολουθείτε όλες τις εξόδους του τμήματος δεδομένων του επεξεργαστή, ενώ μετά από

κάθε εντολή σας προτείνεται να τυπώνετε τις τιμές των καταχωρητών. Για παράδειγμα αν θέλουμε να τυπώσουμε τον καταχωρητή 2, μέσω της εντολής $display:

```
$display("Register %d : %x", i, regs.data[i]);
```

Όπου σε αυτό το παράδειγμα, regs είναι το όνομα της εμφάνισης του τμήματος δεδομένων του επεξεργαστή, registerfile είναι η εμφάνιση του αρχείου καταχωρητών μέσα στο testbench, και data[i] είναι η i λέξη στον εσωτερικό πίνακα των καταχωρητών στην βιβλιοθήκη (*0<=i<=31*).

Για να ελέγξετε την σωστή λειτουργία του κυκλώματός σας θα πρέπει να δημιουργήσετε ένα for-loop στο testbench το οποίο θα διαπερνάει όλους τους καταχωρητές και θα εκτελεί διαφορετικές πράξεις στην ALU. Δεν ψάχνουμε για κάτι συγκεκριμένο στο testbench, απλά έναν τρόπο για να ελέγξετε όσο το δυνατόν περισσότερες λειτουργίες του register file και της ALU.

4. Λογισμικό ανάπτυξης και προσομοίωσης hardware

Το *Modelsim* της Mentor Graphics είναι ένα πλήρες πακέτο προσομοίωσης και σύνθεσης κυκλωμάτων που υποστηρίζει την Γλώσσα Περιγραφής Υλικού (Hardware Description Language) Verilog. Το *Modelsim* θα είναι χρήσιμο για τις ασκήσεις που αφορούν την υλοποίηση, προσομοίωσης και επαλήθευση λειτουργίας επεξεργαστή τύπου MIPS στο υλικό σε γλώσσα Verilog. Το λογισμικό υποστηρίζει πλατφόρμες Linux και Windows και είναι εγκατεστημένο στους υπολογιστές Linux του τμήματος σαν κομμάτι ενός μεγαλύτερου πακέτου εργαλείων CAD που ονομάζεται *questa*. Μπορείτε επίσης να κατεβάσετε δωρεάν το *Modelsim student edition* από την σχετική ιστοσελίδα *http://www.mentor.com/company/higher_ed/modelsim-student-edition* για να εργάζεστε στο desktop/laptop σας.

Στην πλατφόρμα του Linux, η εκκίνηση του προγράμματος και η εμφάνιση των παραθύρων γίνεται με την εντολή *vsim*.

% vsim

Αφού τοποθετήσετε τα αρχεία Verilog που έχετε αναπτύξει σε κάποιο directory, έστω $HOME/ece232/labs/components, μπορείτε να μεταβείτε σε αυτό το directory από την κονσόλα χρησιμοποιώντας τις γνωστές εντολές του *Linux*.

% cd $HOME/ece232/labs/components

Στο directory αυτό θα πρέπει πρώτα να δημιουργήσετε το όνομα της βιβλιοθήκης που θα περιέχει τα binary files που χρησιμοποιεί το *Modelsim* για την αναπαράσταση του κυκλώματος. Στο *Modelsim* το όνομα της τρέχουσας βιβλιοθήκης είναι **work** (by default).

%vlib work

Την πράξη δημιουργίας της βιβλιοθήκης **work** χρειάζεται να την κάνετε μόνο μια φορά. Η εντολή *vlib* δημιουργεί το directory work κάτω από το τρέχων directory και το αρχικοποιεί με κάποια αρχεία. Μην κάνετε edit τα αρχεία αυτά με το χέρι, μιας και ότι αλλαγή υφίστανται θα πρέπει να γίνεται μόνο μέσω των επιμέρους εργαλείων του *Modelsim.*

Για να κάνετε compile τα αρχεία Verilog χρησιμοποιείστε την εντολή *vlog* του *Modelsim*, ή ακόμα καλύτερα, επιλέξτε *Compile* → *Compile* από το pop-down menu στο επάνω μέρος του παραθύρου και επιλέξτε τα αρχεία που θέλετε να γίνουν compile. Το compilation ελέγχει απλά την λεξική και συντακτική ορθότητα του προγράμματος σας, και όχι την συνδεσμολογία των επιμέρους υπομονάδων. Καλό θα ήταν να κάνετε συχνά compile το πρόγραμμά σας κατά την διάρκειά ανάπτυξης του και όχι μόνο στο τέλος.

Εφόσον όλα πάνε καλά με το compilation και όλα τα αρχεία Verilog είναι συντακτικά σωστά, θα μπορέσετε να φορτώσετε το κύκλωμα σας στο *Modelsim* για προσομοίωση

χρησιμοποιώντας την εντολή *vsim*. Στο pop-down menu: *Simulate → Start Simulation και* επιλέξτε το αρχείο *work/testbench.*

Στο σημείο αυτό γίνεται το λεγόμενο *elaboration*, η φάση κατά την οποία όπου τα επιμέρους modules διασυνδέονται και δημιουργείται η ιεραρχία τους. Συνηθισμένο λάθος στο στάδιο αυτό είναι η αναντιστοιχία (mismatch) στο μέγεθος μεταξύ διασυνδεδεμένων σημάτων σε διαφορετικά επίπεδα ιεραρχίας.

Εφόσον και το *elaboration* είναι σωστό και η όλη διαδικασία τερματιστεί χωρίς λάθη ή προειδοποιήσεις (warnings), μπορείτε να ξεκινήσετε το simulation χρησιμοποιώντας εντολές όπως :

run 100 ns // run design for 100 ns

restart // restart simulation from time=0

Ο καλύτερος τρόπος ελέγχου της σωστής λειτουργίας του κυκλώματος σας είναι παρατηρώντας τις κυματομορφές (waveforms).