Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών
ΗΥ-225: Οργάνωση Υπολογιστών
Δ. Πνευματικάτος

ΗΥ-225
Άνοιξη 2000
2000-03-30 – βδομάδα 9
φυλλάδιο 9

5η Σειρά Ασκήσεων
Εκμάθηση υποσύνολου της γλώσσας **Verilog**

(HDL – Γλώσσα Περιγραφής Hardware) και του Προσομοιωτή Veriwell

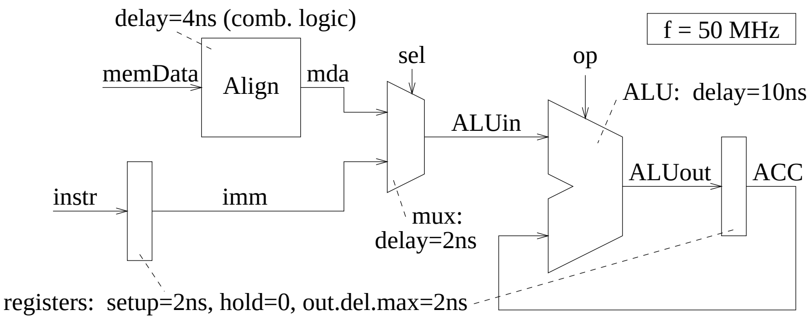
Προθεσμία έως Παρασκευή 14 Απριλίου (βδομάδα 10)

ΑΛΛΑ *ΔΕΝ* χρειαζεται να την παραδώσετε

Στην άσκηση αυτή θα μάθετε ένα μικρό υποσύνολο της γλώσσας ‘‘Verilog’’, που είναι η μία από τις δύο δημοφιλέστερες (παγκοσμίως) γλώσσες περιγραφής hardware (Hardware Description Languages – HDL) (η άλλη είναι η ‘‘VHDL’’). Θα μάθετε μόνο ένα μέρος των εντολών γιά την περιγραφή της συνδεσμολογίας των κυκλωμάτων (structural modeling), και όχι εντολές γιά περιγραφή transistors ή γιά περιγραφή συμπεριφοράς σε ψηλό επίπεδο (behavioral modeling). Επίσης θα μάθετε μερικές εντολές γιά την παροχή εισόδων στο κύκλωμά σας και γιά την παρακολούθηση των εξόδων του, προκειμένου να ελέγχετε τη σωστή λειτουργία του κατά την προσομοίωση.

Την προσομοίωση θα την κάνετε με τον προσομοιωτή ‘‘Veriwell’’ της εταιρείας Wellspring. (Η δημόσια διαθέσιμη έκδοση του προσομοιωτή αυτού την οποία έχουμε επιτρέπει την προσομοίωση συστημάτων μεγέθους μέχρι 1000 γραμμές μόνο). Ο προσομοιωτής είναι του τύπου event-driven, δηλαδή (επαν-) υπολογίζει τη (νέα) τιμή της εξόδου ενός στοιχείου όποτε (και μόνον όποτε) αλλάζει μία από τις εισόδους του στοιχείου αυτού.

Γιά να μάθετε τα παραπάνω θα περιγράψτε σε Verilog το μικρό κυκλωματάκι της 3ης σειράς ασκήσεων:



Γράψτε την περιγραφή σας σ’ ένα αρχείο, π.χ. το "askisi5.v", το οποίο θα είναι περίπου ως εξής:

```
‘define WORD_WIDTH 16
‘define CLK_PERIOD 30

module askisi6;
// Declarations:  (single-line comments start with // )
    reg  clk, sel;
    reg  [‘WORD_WIDTH-1 : 0] memdata, instr;
    wire [‘WORD_WIDTH-1 : 0] mda, imm, ALUin, ALUout, ACC;
    reg  [2:0] op;
```

```
// Module instances
    Reg   #(‘WORD_WIDTH) r_imm(imm, instr, clk);
    Reg   #(‘WORD_WIDTH) r_ACC(ACC, ALUout, clk);
    Align #(‘WORD_WIDTH) align_1(mda, memdata);
    Mux2  #(‘WORD_WIDTH) m_1(ALUin, imm, mda, sel);
    Alu   #(‘WORD_WIDTH) alu_1(ALUout, ALUin, ACC, op);

initial begin   // clock generator
    clk=0;
    forever #(‘CLK_PERIOD/2) clk=˜clk;
end

// initializations, stimulus, etc. -- these are only examples --
//  you will need to provide more!...
initial begin
  // specify which nodes to monitor:
    $monitor("t=%d: mda=%b, ALUout=%b", $time, mda, ALUout);
  // initialize registers:
    instr=   0;
    op=      3’b111;
  // provide stimulus at later times:
    #5 memdata = 16’b0111_0101_0011_0000;
    #(‘CLK_PERIOD) memdata=0; op=0;
    #(‘CLK_PERIOD) instr=10;
    $display("Example of message: at time %d, imm=%b", $time, imm);
  // tell when to stop:
    #(‘CLK_PERIOD) $stop;
end

endmodule
```

Στη συνέχεια, μεταφράστε και προσομοιώστε την περιγραφή σας ως εξής:

```
veriwell  ˜hy225/veriwell/lib/ask5_lib.v  askisi5.v
```

Προφανώς θα χρειαστεί να επεκτείνετε τον παραπάνω κώδικα, να διορθώσετε τυχόν σχεδιαστικά σφάλματα, και να δώσετε περισσότερες εισόδους για έλεγχο του κυκλώματος. Για να δείτε τι πράξεις κάνει η ALU και το κύκλωμα Align, δείτε τον κώδικα της βιβλιοθήκης.

Περισσότερες πληροφορίες γιά τη Verilog και τον προσομοιωτή θα βρείτε στο εγχειρίδιο που θα σας μοιραστεί (ή που μπορείτε να βρείτε στην Ιστοσελίδα του μαθήματος) και στα μέρη όπου αυτό παραπέμπει. Εδώ θα δώσουμε μόνο τα εξής λίγα σχόλια.

Οι κόμβοι "reg" είναι είσοδοι του κυκλώματος, και κρατάνε την τιμή που τους δίδεται μέχρι να τους δοθεί νέα τιμή. Οι κόμβοι "wire" αποτελούν εξόδους κυκλωμάτων και παίρνουν την τιμή που τους δίνουν αυτά τα κυκλώματα, όποτε τους την δίνουν.

Η βιβλιοθήκη που σας δίδεται γιά αυτή την άσκηση περιέχει 4 modules: ακμοπυροδότητο καταχωρητή (Reg), ένα module "Align" που απλά αλλάζει τη θέση μερικών bits, ένα πολυπλέκτη 2-σε-1 (Mux2), και μία ALU (Alu). Τα modules της βιβλιοθήκης είναι παραμετρικά: η πρώτη παράμετρος τους είναι το πλήθος των bits τους. Ειδικά γιά τον καταχωρητή (Reg), η δεύτερη παράμετρος είναι το output-delay ενώ για απλότητα θα αγνοήσουμε το set-up time. Οι πόρτες των modules της βιβλιοθήκης είναι: πρώτη η έξοδος, μετά οι είσοδοι δεδομένων, τέλος η είσοδος ρολογιού ή ελέγχου.

Στο "initial block" (π.χ. βλ. το τελευταίο τέτοιο block), το "#n" σημαίνει καθυστέρηση. Ο προσομοιωτής στο χρόνο 0 να εκτελεί τις εντολές τη μία μετά την άλλη, αλλά μόλις συναντά το πρώτο #5 περιμένει 5 μονάδες χρόνου, και στη συνέχεια εκτελεί το "memdata=...". Μετά, περιμένει άλλες CLK_PERIOD μονάδες χρόνου, κι έτσι στον χρόνο 5+30=35 εκτελεί το επόμενο "memdata=..." και "op=0".