Σειρά Ασκήσεων 8:
Πρώτη Γνωριμία με την Γλώσσα Verilog

Προθεσμία έως Τετάρτη 24 Μαρτίου 2010, ώρα 23:59

[**Up:** Table of Contents] [printer version, in **PDF**]
[**Prev:** 7. Linked List] [9. Verilog Intro. 2 - **Next**]

8.1 Γλώσσες Περιγραφής Hardware (HDL):

Σήμερα, η σχεδίαση hardware σε βιομηχανικό επίπεδο στηρίζεται πρώτ' απ' όλα στις Γλώσσες Περιγραφής Hardware (Hardware Description Languages - HDL). Χρησιμοποιώντας αυτές, μπορεί κανείς να περιγράψει με ακρίβεια την επιθυμητή λειτουργία ή την υλοποίηση ενός ψηφιακού συστήματος, ή (συνήθως) και τα δύο. Αυτό είναι εξαιρετικά σημαντικό, πρώτον για την *προσομοίωση* (simulation) ενός σχεδίου, δηλαδή για να προσπαθήσει ένα πρόγραμμα υπολογιστή, ο "προσομοιωτής", να μιμηθεί τον αναμενόμενο τρόπο λειτουργίας του πραγματικού ψηφιακού συστήματος, δίνοντας μας έτσι τη δυνατότητα να βρούμε και να διορθώσουμε τα λάθη σχεδίασης, καθώς επίσης και να μελετήσουμε την επίδοση του υπό σχεδίαση συστήματος. Δεύτερον, όταν η περιγραφή του υπό σχεδίαση ψηφιακού συστήματος γίνει με κατάλληλο τρόπο, υπάρχουν εργαλεία αυτόματης σύνθεσης που μπορούν διαβάζοντας αυτή την περιγραφή να την μετατρέψουν σε άλλη μορφή, χαμηλότερου επιπέδου, κατάλληλη για αυτόματη κατασκευή του πραγματικού συστήματος (κάτι σαν compilation προγράμματος, δηλαδή).

Στο μάθημά μας θα χρησιμοποιήσουμε τη γλώσσα περιγραφής hardware *"Verilog"* που είναι η μία από τις δύο δημοφιλέστερες (παγκοσμίως) HDL --η άλλη είναι η "VHDL". Η Verilog είναι μεγάλη και περίπλοκη γλώσσα, με πολλές δυνατότητες και πολλούς τρόπους να κάνει κανείς το κάθε τι. Σε αυτό το μάθημα θα μάθετε και θα χρησιμοποιήσουμε ένα πολύ μικρό, μόνο, υποσύνολο της Verilog. Ο σκοπός των σημερινών ασκήσεων είναι η εξοικείωσή σας με την Verilog και τα συναφή εργαλεία:

- Θα δείτε πώς ένα πρώτο, απλούστατο κύκλωμα περιγράφεται σε Verilog, και πώς περιγράφουμε τις κυματομορφές εισόδου του.
- Θα προσομοιώσετε το κύκλωμα αυτό και θα δείτε γραφικά τις κυματομορφές εξόδου του κυκλώματος χρησιμοποιώντας το simulator ModelSim.

8.2 Μοντέλο Καθυστερήσεων:

Θα χρησιμοποιήσουμε βιβλιοθήκες δομικών στοιχείων (πυλών, κλπ) που ακολουθούν βασικά το παρακάτω μοντέλο καθυστερήσεων. Κάθε πύλη AND στις σημερινές ασκήσεις θα έχει μέγιστη καθυστέρηση 200 ps και ελάχιστη καθυστέρηση μηδέν. Αυτό σημαίνει ότι όταν αλλάζει μία είσοδος η οποία προκαλεί αλλαγή στην τιμή της εξόδου, η νέα τιμή στην έξοδο εμφανίζεται κάποια στιγμή μεταξύ 0 και 200 ps. Το γεγονός ότι δεν γνωρίζουμε ποιά ακριβώς στιγμή σταθεροποιείται η έξοδος το μοντελοποιούμε στη Verilog θέτοντας την έξοδο στη τιμή x ("άγνωστο") για το διάστημα από 0 μέχρι 200 ps. Η τιμή x στην έξοδο μίας πύλης μπορεί να είναι είσοδος σε μιαν άλλη πύλη.

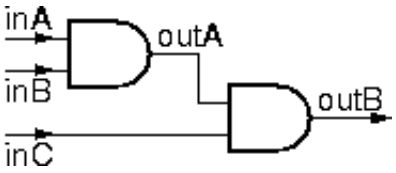
Όταν μια πύλη έχει εισόδους x, δηλαδή εισόδους με άγνωστη τιμή, τότε η τιμή εξόδου της προκύπτει ως εξής. Θεωρούμε όλους τους δυνατούς συνδυασμούς 1 και 0 σε όλες τις εισόδους με τιμή x. Για κάθε τέτοιο συνδυασμό εξετάζουμε τι τιμή θα είχε η έξοδος. Αν όλες αυτές οι τιμές είναι 0, τότε η τιμή εξόδου είναι 0. Αν όλες οι παραπάνω τιμές είναι 1, τότε η τιμή εξόδου είναι 1. Αν όμως μερικές από τις παραπάνω τιμές είναι 0 και μερικές είναι 1, τότε η τιμή εξόδου είναι x, που σημαίνει ότι η αβεβαιότητα που υπάρχει για τις τιμές των εισόδων που είναι x, επηρεάζει την έξοδο σε αυτή την περίπτωση, και κάνει την τιμή της να είναι επίσης αβέβαιη. Με βάση αυτή τη λογική, η τιμή εξόδου μιάς πύλης AND και μιάς πύλης OR, όταν οι είσοδοί τους είναι 0, 1, ή x, θα είναι:

```
AND: inA-> 0   1   x        OR: inA-> 0   1   x
     inB:                        inB:
      0    0   0   0             0    0   1   x
      1    0   1   x             1    1   1   1
```

```
x    0    x    x              x    x    1    x
```

Άσκηση 8.3: Περιγραφή Κυκλώματος σε Verilog

Στις σημερινές ασκήσεις θα περιγράψουμε και προσομοιώσουμε το απλούστατο κύκλωμα που φαίνεται στο σχήμα. Ο κώδικας που το περιγράφει σε Verilog δίδεται εδώ:




```
module top;

    wire outA, outB;
    reg  inA, inB, inC;

    // Instantiate two AND gates:
    //
    lib8_and and1(outA, inA,  inB);
    lib8_and and2(outB, outA, inC);

endmodule
```

Στην περιγραφή αυτή ορίζεται ένα "module" με το όνομα "top". Μέσα του, στην αρχή, δηλώνονται τα σύρματα (wire) outA και outB (που είναι έξοδοι πυλών), καθώς και οι είσοδοι πυλών inA, inB, και inC. Παρά το γεγονός ότι και αυτά τα σήματα (inA, inB, inC) είναι τρόπον τινά απλά σύρματα, πρέπει να δηλωθούν "reg", επειδή στο κύκλωμά μας δεν υπάρχει κανείς που να τα οδηγεί, και κατά συνέπεια, ο μόνος τρόπος να κρατάνε ό,τι τιμή τους βάζουμε είναι να δηλωθούν "reg" (κάτι σαν "καταχωρητής", αλλά όχι το ίδιο με τους δικούς μας καταχωρητές).

Στη συνέχεια, μέσα στο module "top", υλοποιούμε (instantiate) δύο αντίτυπα μιάς πύλης AND. Και τα δύο είναι αντίτυπα της ίδιας πύλης ονόματι "**lib8_and**" από μιά κατάλληλη βιβλιοθήκη που θα πούμε παρακάτω. Το πρώτο αντίτυπο (instance) είναι το "and1", και το δεύτερο είναι το "and2". Η πύλη lib8_and έχει οριστεί με τρείς "πόρτες", δηλαδή τρία σήματα επικοινωνίας με τον έξω (της) κόσμο: το πρώτο (σε σειρά) είναι η έξοδός της, και τα επόμενα δύο είναι οι είσοδοί της. Επομένως, βάσει αυτής της σειράς ορισμού, που πρέπει να μας την έχει δώσει αυτός που έφτιαξε τη βιβλιοθήκη, στο αντίτυπο and1 η έξοδος συνδέεται στο (τροφοδοτεί το) σήμα outA, ενώ οι είσοδοι συνδέονται στα (τροφοδοτούνται από τα) σήματα inA και inB. Στο αντίτυπο and2, η έξοδος πάει στο outB, ενώ είσοδοι είναι τα outA και inC.

Το module **top**, έτσι "σκέτο" όπως το ορίσαμε παραπάνω, αν περάσει από τον προσομοιωτή δεν θα βγάλει κανένα αποτέλεσμα, αφού κανένα ενδιαφέρον συμβάν (γεγονός - event) δεν γεννιέται από καμία πηγή: δεν υπάρχουν συγκεκριμένα σήματα εισόδου για να δώσουν εξόδους. Επομένως πρέπει να επεκτείνουμε το module έτσι ώστε να εφαρμόσουμε ακολουθίες εισόδου (test vectors) και να παρατηρήσουμε τις τιμές σε εξωτερικούς και εσωτερικούς κόμβους του κυκλώματος. Οι επεκτάσεις αυτές δεν αποτελούν τμήμα του κυκλώματος, αλλά το περιβάλλον ελέγχου ("test bench"). Για λόγους απλότητας, σε αυτή την άσκηση, θα τοποθετήσουμε την περιγραφή του κυκλώματος στο ίδιο αρχείο με το περιβάλλον ελέγχου:

```
`timescale 1ps / 1ps
module top;
    wire outA, outB;
    reg  inA, inB, inC;

    // Instantiate two AND gates:
    //
    lib8_and and1(outA, inA,  inB);
    lib8_and and2(outB, outA, inC);

    // Test Bench
    //
    initial
    begin
        // print the values of "outA" and "outB" at stdout
        // each time one of them changes
```

```
        //
        $monitor($time, ": outA=%b, outB=%b\n", outA, outB);

        // Test vectors
        //
        #100    inA = 1; inB = 1; inC = 1;
        #900    inA = 0;
        #1000   inA = 1;
        #1000   inB = 0; inC = 0;
        #1000   inA = 0;
        #1000   inA = 1;
        #1000   inB = 1; #100 inA = 0;
        #1000;
    end
endmodule
```

- Τα τμήματα των γραμμών μετά το "**//**" είναι σχόλια.
- Το περιβάλλον ελέγχου (test bench) βρίσκεται μέσα σε μιάν εντολή "**initial**" (που το block της καθορίζεται από το ζευγάρι begin-end). Η εντολή "initial" σημαίνει "κάνε αυτά που σου λέω μία φορά, αρχίζοντας από την αρχή του χρόνου (από το χρόνο 0)".
- Η οδηγία "**$monitor**" λέει στον προσομοιωτή να μας τυπώνει τον παρόντα χρόνο ακολουθούμενο από τις τιμές των σημάτων outA και outB, όποτε αλλάζει κάποιο από αυτά. Η σύνταξη της οδηγίας αυτής θυμίζει τη σύνταξη της printf στη C ("%b" σημαίνει "binary").
- Η εντολή timescale καθορίζει την μονάδα του χρόνου στην οποία ο simulator βλέπει ότι πραγματοποιούνται γεγονότα στων κώδικα μας. Εδώ ορίζουμε ότι η μονάδα του χρόνου είναι το 1ps.
- Η εντολή "***#αριθμός***", στην Verilog, σημαίνει "περίμενε να περάσει τόση ώρα όση λέει ο αριθμός, και μετά προχώρα σε ό,τι λέει η επόμενη εντολή". Έτσι, στο κομάτι με τις κυματομορφές εισόδου (test vectors), η πρώτη γραμμή (#100) λέει "περίμενε 100 ps" (μετά την αρχή του initial, άρα μετά το χρόνο 0ps, αφού η μονάδα μέτρησης μας είναι το 1ps), δηλαδή περίμενε να φτάσει ο χρόνος το 0 + 100 = 100 ps, και τότε κάνε το σήμα inA να γίνει 1, το σήμα inB να γίνει επίσης 1, και το ίδιο και το σήμα inC. (Ερώτηση: φροντίστε να διαπιστώσετε τι τιμή δίνει η iverilog σε αυτά τα σήματα όταν αρχίζει να δουλεύει ο προσομοιωτής, δηλαδή μεταξύ χρόνου 0 και 100 ps).
- Η επόμενη γραμμή λέει "περίμενε (άλλα) 900 ps", δηλαδή περίμενε να φτάσει ο χρόνος στα 100 + 900 = 1000 ps, και τότε κάνε το σήμα inA να γίνει 0 (εν τω μεταξύ, το σήμα αυτό κρατούσε την τιμή 1 που του είχαμε δώσει τελευταία, επειδή έχει δηλωθεί τύπου "reg"). Η επόμενη γραμμή λέει "περίμενε (άλλα) 1000 ps", δηλαδή περίμενε να φτάσει ο χρόνος στα 1000 + 1000 = 2000 ps, και τότε ξανακάνε το σήμα inA = 1, κ.ο.κ. Στο τέλος, περιμένουμε 1000 επιπλέον ps μετά την τελευταία αλλαγή εισόδου, προκειμένου να δούμε τις συνέπειες που είχε αυτή η τελευταία αλλαγή σε μετέπειτα χρονικές στιγμές.

Γράψτε το παραπάνω module στο αρχείο σας "**top.v**". Επίσης, αντιγράψτε στην περιοχή σας το αρχείο:

~hy225/verilog/lib/lib8.v

που περιέχει τα βοηθητικά modules για την πύλη AND και τους ορισμούς των καθυστερήσεων.

Άσκηση 8.4: Ελεγχος Κυματομορφών με το ModelSim

Κατεβάστε και εγκαταστήστε το ModelSim (προτιμήστε την έκδοση από Xilinx ή Altera στα οποία δεν χρειάζεται registration):
Xilinx version: **link** (for Windows)
Altera version: **link** (for Linux & Windows) (επιλέξτε το ModelSim-Altera Starter Edition)
ModelSim : **link** (for Windows but needs registration)
Για ένα σύντομο οδηγό του ModelSim μπορείτε να δείτε αυτό το **link**.

Τρέξτε το Modelsim και με την εντολή File -> Change Directory... μεταφερθείτε στο directory που βρίσκονται τα αρχεία σας (lib8.v, top.v).
Στη συνέχεια εκτελέστε από το Transcript window τις παρακάτω εντολές:

```
ModelSim> vlib work
ModelSim> vlog -work work lib8.v
ModelSim> vlog -work work top.v
ModelSim> vsim -novopt work.top
ModelSim> add wave *
```

```
ModelSim> run 10 ns
```

Η εντολή vlib δημιουργεί την βιβλιοθήκη που θα αποθηκεύονται τα modules.
Η εντολή vlog κάνει compile τον κώδικα verilog στην βιβλιοθήκη work.
Η εντολή vsim προσομοιώνει το module top.
Τέλος το add wave χρησιμοποιείται για να προσθέσετε όλα τα σήματα στο παράθυρο προσομοίωσης και το run χρησιμοποιείται για να δηλώσετε για πόσο χρόνο θέλετε να τρέξει η προσομοίωση.
Σε περίπτωση που θέλετε να τρέξετε ένα σύνολο από εντολές (script) μπορείτε να τις αποθηκεύσετε σε ένα αρχείο .do και στη συνέχεια να τρέξετε το script με την εντολη do. (πχ. `ModelSim> do run1.do`)

Όταν τελειώσει η προσομοίωση, παρατηρήστε τις κυματομορφές στο wave window και ελέγξτε ότι οι έξοδοι έχουν την αναμενόμενη τιμή, με βάση τις εισόδους που δώσαμε στο test bench.

Αφού τρέξει το γραφικό περιβάλλον του προγράμματος, ακολουθήστε τα εξής βήματα:

- Στην αριστερή πλευρά του παραθύρου φαίνεται το top level module, καθώς και τα υπο-modules, ιεραρχικά.
- Επιλέγοντας ένα από αυτά, εμφανίζονται απο κάτω όλα τα σήματα που περιέχονται στο module αυτό.
- Με τους φακούς με τα σύμβολα "+", "-" ακριβώς πάνω-αριστερά από την κυματομορφή μπορείτε να κάνετε zoom-in και zoom-out.
- Ολα τα σήματα που μας ενδιαφέρουν φαίνονται καθαρά. Ελέγξτε και πάλι τις τιμές τους.

Το ModelSim αναπαριστά τη τιμή x με κόκκινη μπάρα ανάμεσα στο επίπεδο 0 όσο και στο επίπεδο 1. Παρατηρήστε τη συμπεριφορά των εξόδων σε κάθε μία από τις χρονικές στιγμές 1ns, 2ns, 3ns, 6ns. Πόση ώρα μένει η έξοδος στην τιμή x σε κάθε περίπτωση και γιατί;

Παραδώστε, όπως και στις προηγούμενες ασκήσεις, ένα χαρακτηριστικό στιγμιότυπο από το ModelSim της άσκησης 8.4, "**`ask8_4.jpg`**", και τα σχόλια σας για τις τιμές στις εξόδους, "**`ask8_4.txt`**".
Χρησιμοποιήστε την ίδια διαδικασία όπως και στις προηγούμενες ασκήσεις.

[**Up:** Table of Contents] [printer version, in **PDF**]
[**Prev:** 7. Linked List] [9. Verilog Intro. 2 - **Next**]

Up to the Home Page of CS-225