ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Εργαλείων CAD για Σχεδίαση Ολοκληρωμένων Κυκλωμάτων (ΗΡΥ 419 - ΗΡΥ 608)

PROJECT ΕΞΑΜΗΝΟΥ – ΑΝΑΠΤΥΞΗ ΕΡΓΑΛΕΙΟΥ ΠΛΗΡΟΥΣ ΠΡΟΣΟΜΟΙΩΣΗΣ ΣΥΝΔΥΣΑΣΤΙΚΩΝ ΛΟΓΙΚΩΝ ΚΥΚΛΩΜΑΤΩΝ

ΑΝΑΦΟΡΑ

ΜΠΙΜΠΙΡΗΣ ΠΕΤΡΟΣ (Α.Μ.: 2019030135)

Εισαγωγή:

Το παρόν project ήταν η υλοποίηση ενός «πλήρους ψηφιακού προσομοιωτή για τα λογικά κυκλώματα που επεξεργαστήκαμε στις Ασκήσεις 2-5». Η προσομοίωση γίνεται με βάση πίνακες αληθείας των πυλών που αποτελούν τα κυκλώματα. Τα σήματα εισόδου δίνονται σε ένα αρχείο “testbench”, στο οποίο δηλώνονται και οι έξοδοι του κυκλώματος των οποίων οι τιμές παρουσιάζουν ενδιαφέρον. Ο προσομοιωτής γράφει σε ένα αρχείο εξόδου τις εισόδους και τα αποτελέσματα κάθε προσομοίωσης, μαζί με ορισμένες μετρήσεις που αφορούν την ταχύτητα με την οποία κατασταλάζει το κύκλωμα.

Χρήση του εργαλείου

Το πρόγραμμα καλείται από την γραμμή εντολών (command line) ως εξής:

```
$ ./simulate
```

και εκτελείται χωρίς κάποια αλληλεπίδραση με τον χρήστη. Υπάρχει η δυνατότητα να δοθούν τιμές για ορισμένες παραμέτρους του προγράμματος με την μορφή command line arguments. Οι επιλογές που δίνονται καθώς και η αντίστοιχη «σύνταξη» φαίνεται χρησιμοποιώντας το argument $-h$:

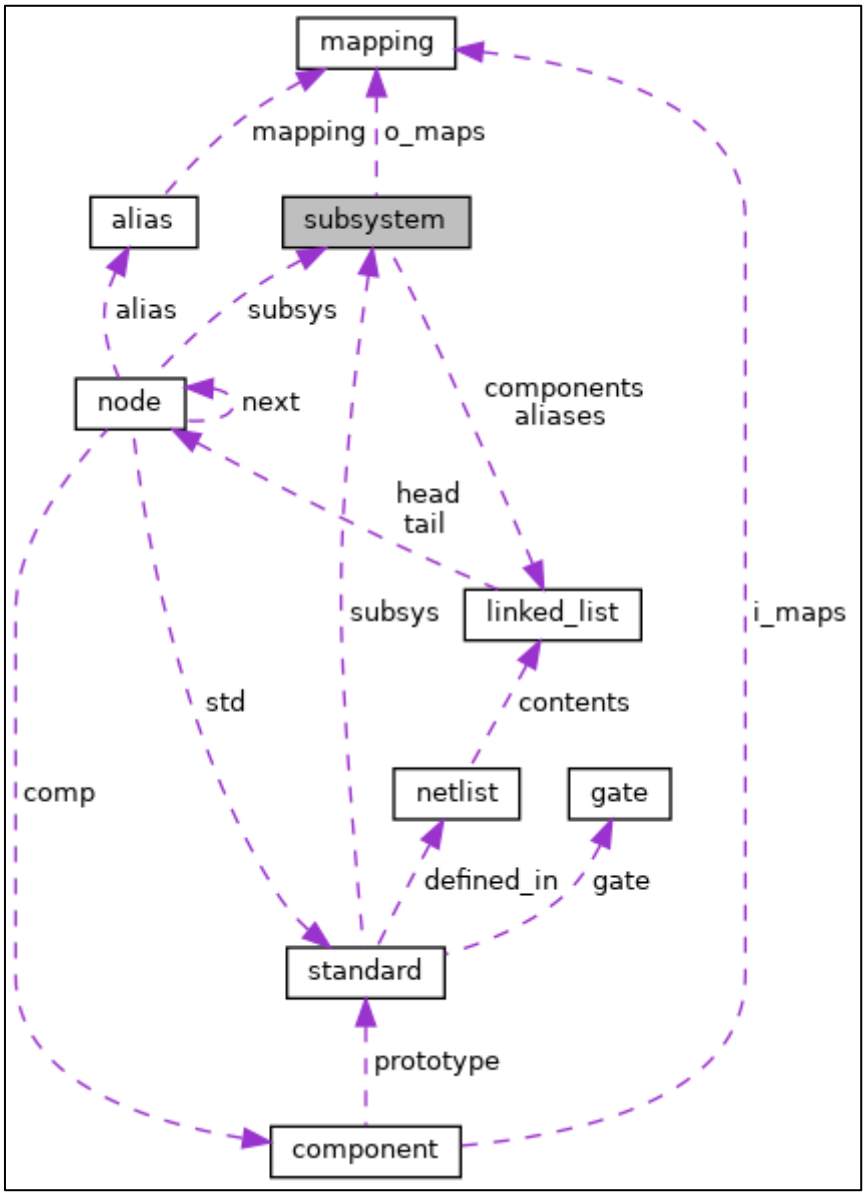
```
$ ./simulate -h
```

Δομή Υλοποίησης:

Ο κώδικας του εργαλείου βρίσκεται στο αρχείο `simulate.c`.

Όπως και στις προηγούμενες ασκήσεις, χρησιμοποιούνται δομές και συναρτήσεις από τις «βιβλιοθήκες» `netlist.h` και `string.h`, στις οποίες έχουν προστεθεί συναρτήσεις που διευκολύνουν τον χειρισμό των κυκλωμάτων, των υποσυστημάτων/πυλών, των πινάκων αληθείας, των testbench κ.α..

Οι κύριες δομές που χρησιμοποιούνται για τον χειρισμό των κυκλωμάτων και των υποσυστημάτων ορίζονται στο αρχείο `netlist.h`. Οι μεταξύ τους σχέσεις σκιαγραφούνται στην εικόνα 1:



Εικόνα 1: Σχέσεις εξάρτησης μεταξύ των βασικών δομών δεδομένων του προσομοιωτή

Αναλυτική περιγραφή για κάθε δομή, συνάρτηση, σταθερά και τύπο δεδομένων του προγράμματος περιέχεται στο documentation του εργαλείου, το οποίο είναι διαθέσιμο σε html μορφή. Για λόγους εξοικονόμησης χώρου δεν περιέχεται στο παρόν αρχείο .zip, αλλά μπορεί να κατασκευαστεί από τα σχόλια στον κώδικα με το Doxygen (https://www.doxygen.nl/). Για να γίνει αυτό αρκεί κανείς να εκτελέσει την εξής εντολή:

```
$ make doc
```

Με την εκτέλεση αυτής της εντολής δημιουργείται ένας φάκελος με όνομα ‘`doc`’. Το αρχείο ‘`./doc/html/index.html`’ αντιστοιχεί στην «αρχική σελίδα» του documentation.

Υλοποίηση – Μοντελοποίηση και εκτέλεση προσομοίωσης:

Η προσομοίωση βασίζεται κατά κύριο λόγο στους πίνακες αληθείας των πυλών που αποτελούν τα κυκλώματα και στα αρχεία testbench. Ακολουθεί μια high level περιγραφή της λογικής με την οποία υλοποιήθηκαν και λειτουργούν τα παραπάνω:

- **Πίνακες Αληθείας**: Κάθε πύλη ορίζεται σε μια βιβλιοθήκη μαζί με τον πίνακα αληθείας της. Αφού οι πύλες έχουν εξ' ορισμού μόνο μια έξοδο, ο πίνακας αληθείας τους μπορεί να αναπαρασταθεί ως μια σειρά από bits (bitstring) εκ των οποίων το καθένα αναπαριστά την έξοδο της πύλης με διαφορετική είσοδο. Ανάλογα με τις εισόδους της πύλης μπορούμε να κατασκευάσουμε ένα δεύτερο bit string το οποίο να συνδυαστεί με τον πίνακα αληθείας και να προκύψει η τιμή της εξόδου της πύλης για τις δεδομένες εισόδους.
  Η παρατήρηση αυτή μας επιτρέπει να υλοποιήσουμε τον πίνακα αληθείας κάθε πύλης ξοδεύοντας ελάχιστη μνήμη (το μέγεθος ενός ακεραίου είναι παραπάνω από αρκετό) και να υπολογίσουμε την έξοδο κάθε πύλης εξαιρετικά γρήγορα, χρησιμοποιώντας σχεδόν αποκλειστικά bitwise operations.

  Για παράδειγμα, έστω μια πύλη XOR 2 εισόδων. Ο πίνακας αληθείας της μπορεί να αναπαρασταθεί ως '0110', και για εισόδους 0 και 1 έχουμε: $'01'_{\text{bin}} = 1_{\text{dec}}$, άρα η έξοδος θα είναι το bit που βρίσκεται 1 θέση δεξιά από το MSB (ή αριστερά από το LSB, οι είσοδοι είναι συμμετρικές), δηλαδή το '1'.

- **Αρχεία Testbench**: Κάθε εκτέλεση προσομοίωσης βασίζεται σε ένα αρχείο testbench. Το αρχείο αυτό περιέχει τις τιμές των εισόδων που θα δοκιμαστούν κατά την προσομοίωση, καθώς και τα σήματα εξόδου των οποίων οι τιμές πρέπει να περιέχονται στο αρχείο εξόδου. Επιπλέον το αρχείο αυτό έχει συγκεκριμένο format (που ορίζεται στην εκφώνηση), οπότε η κατασκευή μιας δομής που να το αναπαριστά είναι εύκολη. Σε συνδυασμό με την δομή που αναπαριστά το υποσύστημα, παρέχουν όλα τα στοιχεία που χρειάζονται για την προσομοίωση.

Η διαδικασία της προσομοίωσης υλοποιείται με double buffering: δημιουργούνται 2 buffers που αποθηκεύουν τις λογικές τιμές των κόμβων του κυκλώματος, ο «παλιός» και ο «νέος». Επαναληπτικά υπολογίζονται οι τιμές των κόμβων (με εισόδους από τον «παλιό») και αντικαθίστανται στον («νεό») buffer σε περίπτωση που διαφέρουν από τις προηγούμενες. Ο «παλιός» buffer γίνεται «νέος», ο «νέος» γίνεται «παλιός», και η διαδικασία επαναλαμβάνεται μέχρι να περάσουν 2 επαναλήψεις (iterations) χωρίς αλλαγές, όποτε θεωρούμε ότι το κύκλωμα έχει κατασταλάξει στην τελική του κατάσταση και εμφανίζουμε την έξοδο.

Δοκιμή – Επαλήθευση Ορθής Λειτουργίας Προσομοιωτή

Η ορθή λειτουργία του προσομοιωτή επαληθεύτηκε μέσω προσομοιώσεων διαφορετικών κυκλωμάτων. Για λόγους εξοικονόμησης χώρου στο παρόν αρχείο .zip παρατίθενται μόνο ορισμένα από αυτά:

Κύκλωμα (αρχείο netlist)	Testbench	Αποτέλεσμα	Χρόνος Εκτέλεσης (max #iterations)
5-bit full adder	testbench.txt	Επιτυχής προσομοίωση	1.4 msec (14)
8-bit full adder	testbench_8.txt	Επιτυχής προσομοίωση	2.0 msec (20)
8-bit full adder (eclass)	testbench_eclass.txt	Επιτυχής προσομοίωση	1.9 msec (21)
MUX (2-to-1)	testbench_mux.txt	Επιτυχής προσομοίωση	0.5 msec (6)
MUX_ALIAS*	testbench_mux.txt	Επιτυχής προσομοίωση	0.5 msec (6)
5-bit full adder	testbench_error.txt	Κατάλληλο μήνυμα λάθους	-

Πίνακας 1: Αποτελέσματα δοκιμαστικών εκτελέσεων

Όλα τα κυκλώματα του πίνακα 1 βρίσκονται στην βιβλιοθήκη υποσυστημάτων subsystem.lib, και όλα τα αντίστοιχα testbench περιλαμβάνονται στο παραδοτέο zip.

*Ο πολυπλέκτης MUX_ALIAS είναι πανομοιότυπος με τον «απλό» πολυπλέκτη 2:1, με μόνη διαφορά ότι στο netlist του χρησιμοποιείται ένα ενδιάμεσο σήμα – alias.

Σημείωση:

Η ανάπτυξη του κώδικα έγινε σε περιβάλλον Linux (συγκεκριμένα Ubuntu 20.04), οπότε και το Makefile είναι σχεδιασμένο αντίστοιχα. Σε οποιοδήποτε περιβάλλον Linux η εντολή make είναι κατά πάσα πιθανότητα αρκετή για να γίνουν σωστά compile τόσο οι βιβλιοθήκες όσο και το εργαλείο καθ' αυτό. Σε windows ωστόσο το compilation process θα έπρεπε να γίνει «χειροκίνητα».