ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

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

PROJECT 3 – ΜΕΤΑΤΡΟΠΗ NETLIST ΑΠΟ SUBSYSTEMS ΣΕ ΠΥΛΕΣ

ΑΝΑΦΟΡΑ

ΜΠΙΜΠΙΡΗΣ ΠΕΤΡΟΣ (Α.Μ.: 2019030135)

Εισαγωγή:

Το παρόν project αφορά την υλοποίηση ενός εργαλείου που δέχεται ως είσοδο ένα netlist (για παράδειγμα το netlist εξόδου της προηγούμενης άσκησης) και αντικαθιστά κάθε υποσύστημα με τις πύλες από τις οποίες αποτελείται. Η έξοδος του εργαλείου είναι ένα αρχείο στο οποίο περιέχονται τα ίδια (από πλευράς λειτουργικότητας) υποσυστήματα με το αρχείο εισόδου, υλοποιημένα μόνο με πύλες.

Χρήση του εργαλείου

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

```
$ ./translate_to_gates.exe
```

και εκτελεί την λειτουργία του χωρίς κάποια αλληλεπίδραση με τον χρήστη. Αν η εκτέλεση ολοκληρωθεί επιτυχώς, τυπώνεται αντίστοιχο μήνυμα και το πρόγραμμα τερματίζει. Σε αντίθετη περίπτωση, τυπώνεται μήνυμα λάθους που πληροφορεί τον χρήστη για το σφάλμα που προέκυψε.

Για να λειτουργήσει σωστά το εργαλείο αυτό πρέπει να «γνωρίζει» όλα τα υποσυστήματα (subsystems) και τις πύλες που χρησιμοποιούνται στο netlist εισόδου, οπότε είναι απαραίτητο να βρίσκονται στην ίδια τοποθεσία με το εκτελέσιμο και οι βιβλιοθήκες που τα περιέχουν.

Τα ονόματα των βιβλιοθηκών αυτών, καθώς και των netlist εισόδου και εξόδου είναι παραμετροποιήσιμα μέσω των σταθερών (constants) που ορίζονται στο αρχείο `translate_to_gates.c`.

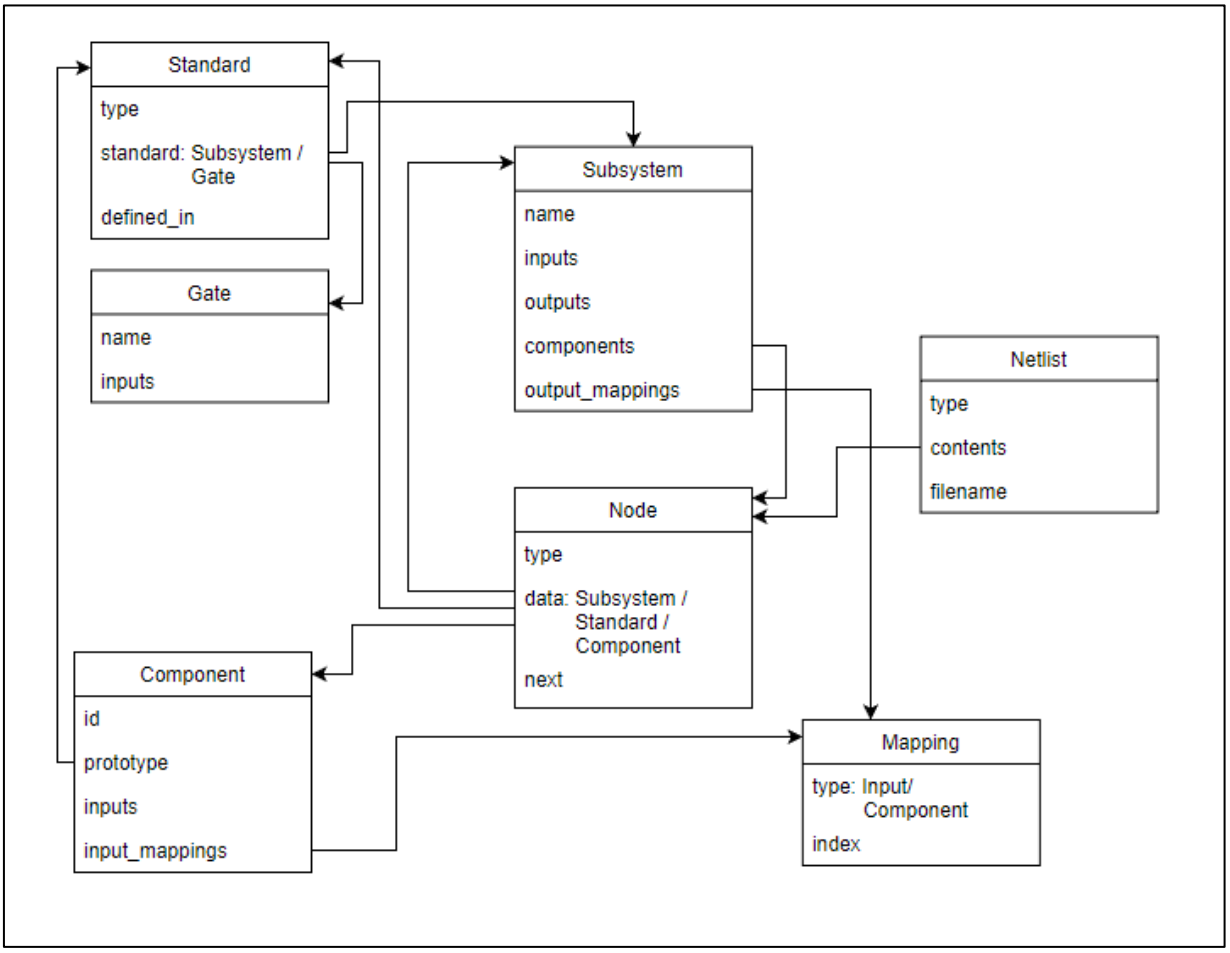
Δομή Υλοποίησης:

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

Χρησιμοποιούνται δομές και συναρτήσεις από την «βιβλιοθήκη» `netlist.h`, όπως και στην προηγούμενη άσκηση. Στην βιβλιοθήκη έχουν προστεθεί επιπλέον δομές και συναρτήσεις που υλοποιούν την επιπρόσθετη λειτουργικότητα που ήταν απαραίτητη για το παρόν project. Μια

σύντομη περιγραφή των δομών που χρησιμοποιήθηκαν καθώς και των συνδέσεων μεταξύ τους φαίνεται παρακάτω στο διάγραμμα 1.

Για τον ίδιο λόγο προστέθηκαν οι αντίστοιχες συναρτήσεις στην βιβλιοθήκη χειρισμού strings, `str_util.h`.



Διάγραμμα 1

Οι δομές που φαίνονται στο διάγραμμα 1 χρησιμοποιούνται ως εξής (για μεγαλύτερη λεπτομέρεια, ο κώδικας περιέχει εκτενή σχόλια):

- `Gate`: Αναπαριστά μια λογική πύλη. Οι λογικές πύλες μπορούν να έχουν οποιονδήποτε αριθμό εισόδων αλλά μόνο μια έξοδο. Οι πύλες που διαβάζονται από το component library μετατρέπονται σε δομές τέτοιου τύπου.

- `Standard`: Αναπαριστά ένα «πρότυπο» υποσύστημα ή πύλη, με βάση το οποίο δημιουργούνται νέα υποσυστήματα του ίδιου τύπου. Όταν μια πύλη ή ένα υποσύστημα διαβάζεται από μια βιβλιοθήκη, η δομή που το αναπαριστά ενθυλακώνεται σε ένα Standard.

- `Subsystem`: Αναπαριστά ένα υποσύστημα (οποιοδήποτε κύκλωμα με παραπάνω από μια εξόδους). Περιέχει πληροφορίες για τα «συστατικά» (components) που το υλοποιούν, τις εισόδους/εξόδους του υποσυστήματος και τις συνδέσεις μεταξύ τους.

- `Component`: Αναπαριστά ένα instance ενός υποσυστήματος ή πύλης ως συστατικό ενός μεγαλύτερου υποσυστήματος. Περιέχει το `Standard` το οποίο ακολουθεί/υλοποιεί, ένα μοναδικό (εντός του υποσυστήματος που το περιέχει) id, και πληροφορίες για τις εισόδους του.

- `Mapping`: Αναπαριστά μια σύνδεση μεταξύ 2 «σημείων» του κυκλώματος.
  Για παράδειγμα, αν έχουμε ένα υποσύστημα του οποίου η δεύτερη έξοδος συνδέεται με την έξοδο του πρώτου component του, τότε για αυτό το υποσύστημα
  $$output\_mappings[1] = \{type: Component, index: 0\}$$
  Ομοίως, αν η πρώτη είσοδος ενός component συνδέεται με την τρίτη είσοδο του υποσυστήματος στο οποίο ανήκει, τότε για αυτό το component
  $$input\_mappings[0] = \{type: Input, index: 2\}$$
  και αν η δεύτερη είσοδος του ίδιου component συνδέεται με την έξοδο του πρώτου component του υποσυστήματος στο οποίο ανήκει,
  $$input\_mappings[1] = \{type: Component, index: 0\}$$

- `Node`: Αναπαριστά έναν κόμβο σε οποιοδήποτε linked list. Το περιεχόμενό του είναι τύπου union, και προκειμένου να φαίνεται ποιο από τα μέλη του union χρησιμοποιείται υπάρχει η μεταβλητή type.

- `Netlist`: Αναπαριστά οποιοδήποτε netlist διαβάζεται από ένα αρχείο. Τα περιεχόμενα του είναι μια linked list από nodes, οπότε μπορεί να είναι υποσυστήματα, πύλες, ή ακόμα και `Standards` (στην περίπτωση που το αρχείο είναι βιβλιοθήκη).

Υλοποίηση – Μετατροπή Υποσυστημάτων σε Πύλες

Τα βήματα που ακολουθεί το πρόγραμμα προκειμένου να επιτευχθεί η επιθυμητή λειτουργικότητα περιγράφονται (συνοπτικά) παρακάτω:

1) Αρχικά διαβάζονται από το component library οι πύλες που θεωρούνται «γνωστές» και μπορούν να χρησιμοποιηθούν για να κατασκευαστούν πιο περίπλοκα υποσυστήματα. Η λειτουργία αυτή εκτελείται από την συνάρτηση `gate_lib_from_file()`, η οποία:
   - διαβάζει το αρχείο γραμμή – γραμμή,
   - δημιουργεί για κάθε γραμμή ένα μια μεταβλητή τύπου Gate, η οποία «τοποθετείται» μέσα σε μια μεταβλητή τύπου Standard
   - προσθέτει την μεταβλητή τύπου standard στην βιβλιοθήκη (μεταβλητή τύπου Netlist)

2) Στην συνέχεια καλείται η `subsys_lib_from_file()`, η οποία ακολουθεί παρόμοια διαδικασία, για να δημιουργήσει μια μεταβλητή τύπου Netlist που περιέχει ορισμούς (μεταβλητές τύπου standard) υποσυστημάτων. Οι κύριες διαφορές από την `gate_lib_from_file()` είναι οι εξής:
   - όταν βρεθεί γραμμή που ξεκινά με την λέξη COMP (δηλαδή γραμμή όπου δηλώνεται καινούριο subsystem), οι επόμενες γραμμές μέχρι και το END … NETLIST χρησιμοποιούνται για να οριστεί το πρότυπο ενός υποσυστήματος.
   - συγκεκριμένα στην δομή μεταφέρονται πληροφορίες:
     - για τις εισόδους/εξόδους του υποσυστήματος (ονόματα)

- για τα components του υποσυστήματος (ονόματα, Standards, input mappings)
- για τις συνδέσεις των εξόδων με τα components (output mappings)

3) Στην συνέχεια διαβάζεται με παρόμοιο τρόπο και αποθηκεύεται σε μια δομή τύπου Netlist (αυτή τη φορά τα περιεχόμενα είναι τύπου Subsystem) το αρχείο που περιέχει τον n-bit full adder

4) Στην συνέχεια τα components του υποσυστήματος μετατρέπονται σε πύλες με την εξής διαδικασία:
   - για κάθε component του αρχικού συστήματος (δηλαδή για κάθε single-bit full adder) δημιουργείται μια ενδιάμεση δομή τύπου subsystem. Με αυτόν τον τρόπο το κάθε component δεν περιέχει μόνο τις πύλες που το αποτελούν, **αλλά και τις συνδέσεις των εξόδων** του με αυτές. Γνωρίζουμε έτσι ότι κάθε έξοδος ενός component (π.χ. UXX_COUT) αντιστοιχεί στην έξοδο μιας πύλης (π.χ. UYY) στο εσωτερικό του. Το ενδιάμεσο subsystem δημιουργείται ως εξής:
     - με την βοήθεια των input mappings που περιέχει το component, βρίσκονται τα ονόματα των inputs του (π.χ. αν για ένα component έχουμε $input\_mappings[0] = \{type: Input, index: 2\}$ και το τρίτο input του n-bit full adder λέγεται '$A3$', το πρώτο input του component θα πάρει επίσης την τιμή '$A3$'.
     - αφού γίνουν resolve τα input mappings, καλείται η συνάρτηση `create_custom()`, η οποία δέχεται ως ορίσματα τα inputs και το standard και κατασκευάζει το ενδιάμεσο subsystem.
   - όταν ολοκληρωθεί η παραπάνω διαδικασία, δημιουργείται ένα τελικό subsystem που περιέχει ως components τα components όλων των ενδιάμεσων subsystems.

5) Τα output mappings του τελικού subsystem γίνονται επίσης resolve σε πύλες, σύμφωνα με τα output mappings του κάθε ενδιάμεσου υποσυστήματος (π.χ. αν έχουμε S1 = U1_COUT, το S1 θα συνδεθεί με την πύλη που είναι συνδεδεμένη με την έξοδο COUT στο ενδιάμεσο υποσύστημα που αντιστοιχεί στο component με id U1)

6) Τέλος εκτυπώνονται στο αρχείο εξόδου όλες οι πύλες (ομαδοποιημένες κατά 5 – όσες υλοποιούν τον single-bit full adder – για ευκολία ανάγνωσης) καθώς και οι συνδέσεις των εξόδων.

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». Για να αποφευχθεί αυτό, στο παραδοτέο zip περιλαμβάνεται και ένα statically compiled windows executable, το οποίο μπορεί να εκτελεστεί κατευθείαν (αρκεί να βρίσκεται στον ίδιο φάκελο με τα subsystem & component libraries).