ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
TECHNICAL UNIVERSITY OF CRETE

Ανάπτυξη Εργαλείων CAD για Σχεδιάση Ολοκληρωμένων Κυκλωμάτων

(ΗΡΥ 419)

Project

Κωνσταντίνος Νικολός 2019030096

Σκοπός της άσκησης:

Σκοπός της συγκεκριμένης άσκησης είναι η δημιουργία ενός εργαλείου που μπορεί να διαβάζει οποιοδήποτε σύστημα του δοθεί με συγκεκριμένη μορφή πυλών και να παράγει το αποτέλεσμα τους για όσες τιμές δοθούν ως είσοδοι του εργαλείου. Συγκεκριμένα δίνεται έμφαση στη γενικότητα του εργαλείου το οποίο πρέπει να μπορεί να επεξεργαστεί οποιοδήποτε αριθμό πυλών που έχουν δοθεί σε άκυρη σειρά αναγνωρίζοντας μερικές λέξεις κλειδιά.

Περιγραφή της άσκησης:

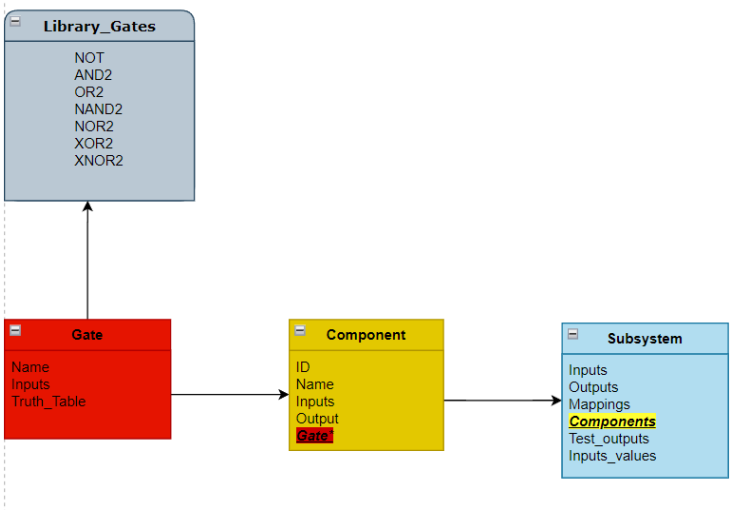
Αρχικά διαβάζεται μια βιβλιοθήκη πυλών η οποία χρησιμοποιείται για την αναγνώριση τους και τον υπολογισμό των τιμών με βάση των πίνακα αληθείας τους. Διαβάζεται ένα σύστημα αποτελούμενο από components εισόδους και εξόδους. Κάθε component περιέχει μία πύλη στην οποία αναφέρεται μέσω της βιβλιοθήκης. Διαβάζοντας ένα αρχείο testbench λαμβάνονται οι τιμές των εισόδων του συστήματος και εκτελούνται οι πράξεις ώστε κάθε component του συστήματος να καταλήξει σε μία τιμή εξόδου. Για την εκτέλεση των πράξεων χρησιμοποιούνται δύο buffers οι οποίοι εναλλάσσονται ως είσοδοι και έξοδοι των πράξεων αποφεύγοντας έτσι να χρειάζεται η διαρκής αποθήκευση των νέων τιμών. Αφού το κύκλωμα συγκλίνει οι τιμές των εξόδων αποθηκεύονται και όσες ζητήθηκαν από το αρχείο testbench τυπώνονται σε ένα αρχείο μαζί με τις τιμές των εισόδων που αποτελεί την έξοδο του προγράμματος.

Δομές και Λειτουργία τους:

Για την εκτέλεση του προγράμματος χρησιμοποιούνται 3 βασικές δομές gate,component και subysystem.

- **Gate** Αποτελεί μια απλή πύλη με το όνομα , τις εισόδους και τον πίνακα αληθείας της.
- **Component** Αποτελεί μία πύλη όπως διαβάζεται από το subsystem με όνομα , ID , εισόδους και έξοδο. Ακόμη περιέχει μία αναφορά σε μια πύλη με gate* ώστε να έχει πρόσβαση στον πίνακα αληθείας της.

- **Subsystem** Εδώ αποθηκεύεται η δομή όλης της άσκησης, οι είσοδοι, οι έξοδοι και τα components του συστήματος όπως διαβάζονται από το αρχείο. Η δομή του είναι δυναμική μπορεί να διαβαστεί οποιοσδήποτε αριθμός εισόδων εξόδων και components καθώς ο χώρος τους δεσμεύεται με την χρήση της συνάρτησης remalloc(). Ακόμη στη δομή αποθηκεύονται όλες τις τιμές των εισόδων που δίνονται από το testbench ώστε να μπορούν να υπολογιστούν οι έξοδοι για κάθε τιμή. Τέλος οι τιμές των εξόδων του subsystem και του testbench αποθηκεύονται εκεί ώστε να γραφτούν στο αρχείο εξόδου.



Συναρτήσεις:

- int read_line_from_file()
  Χρησιμοποιείται για να διαβαστεί μια γραμμή από ένα αρχείο
- char* split()
  Χρησιμοποιείται για να χωριστεί ένα string στο αναζητούμενο διαχωριστικό
- int checkStart()
  Χρησιμουποιείται για τον έλεγχο της αρχής ενός string

- getLibraryFromFile()

  Χρησιμοποιείται για να παραχθεί η βιβλιοθήκη από gates διαβάζοντας από το δοσμένο αρχείο.

- getSubSystemFromFile()

  Χρησιμοποιείται για να διαβαστεί ένα οποιοδήποτε σύστημα από το δοσμένο αρχείο ελέγχοντας πως κάθε component του αποτελεί μια έκτων πυλών από τη βιβλιοθήκη και συνδέοντας το με αυτή. Η μορφή του subystem είναι δυναμική ,μπορούν να διαβαστούν όσες είσοδοι, έξοδοι και components δοθούν. Δεν χρειάζεται να διαβαστούν με κάποια συγκεκριμένη σείρα ή τα components ids να αποτελούν μέρος κάποιας σειράς. Αρκεί μόνο να αναγνωρίζοντααι κάποιες λέξεις κλειδιά όπως το U για component

- readTestbench()
  Διαβάζει το δοσμένο testbench αρχείο και αποθηκεύει τις εισόδους μαζί με όλες τις τιμές τους και τις εξόδους στη δομή subsystem

- createBuffers()
  Δημιουργεί τους δύο buffers

- setInputs()
  Για κάθε component input αναζητάμε εάν είναι είσοδος του συστήματος και αποθυκεύουμε την τιμή του η εάν πρόκειται για έξοδο ενός άλλου component και παίρνουμε την τιμή του από τον input_buffer

- doTheMath()
  Εκτελεί την πράξη για κάθε component διαβάζοντας από τον buffer εισόδου τις τιμές , τον πίνακα αληθείας της πύλης και γράφει το αποτέλεσμα στον buffer εξόδου. Κρατάει counter των αλλαγών σε τιμές που γίνονται ώστε να γνωρίζει πότε έχει κατασταλάξει το κύκλωμα

- doubleBufferSystem()
  Εναλλάσσει τους 2 buffers μέχρι το κύκλωμα μας να συγκλίνει παρουσιάζοντας δύο φορές τις ίδιες τιμές. Για την υλοποίηση της doubleBufferSystem χρησιμοποιούνται και οι συναρτήσεις createBuffers(),saveLastValues(),setInputs(),doTheMath()

- findTheSubsOutputs()
  Για κάθε output του subsystem βρίσκει το αντίστοιχο component στο οποίο αναφέρεται και βάζει την τιμή του

- findTestBenchOutputs()
  Βρίσκει τις τελικές τιμές των εξόδων που ζητήθηκαν από το testbench αναζητώντας όλες τις εξόδους εισόδους και components του συστήματος. Μπορεί να είναι οποιοδήποτε από τα 3.

- doForEveryValue()
  Τελική συνάρτηση που εκτελεί όλες τις προηγούμενες ώστε να παραχθούν οι έξοδοι για κάθε τιμή εισόδων που δόθηκε και να γραφτούν σε ένα αρχείο εξόδου

<u>Περιγραφή λειτουργίας του προγράμματος:</u>

Αρχικά καλούνται οι συναρτήσεις getLibraryFromFIle() και getSubSystemFromFile()για να διαβαστεί η βιβλιοθήκη των πυλών και το κύκλωμα αντίστοιχα. Κάθε component του κυκλώματος αντιστοιχίζεται με μία πύλη της βιβλιοθήκης ώστε να έχει πρόσβαση στον πίνακα αληθείας της. Ακόμη καλείται η συνάρτηση readTestbench() για ν διαβαστεί το αρχείο testbench αποθυκέυοντας όλες τις τιμές των inputs σε ένα 2D int array καθώς και τα ονόματα τον εξώδων που ζητήθηκαν επίσης στη δομή του subsystem. Για την εκτέλεση όλου του προγράμματος καλείται η συνάρτηση doForEveryValue() η οποία με τη σειρά της καλεί την doubleBufferSystem(), findSubsOutputs() και findTestbenchOutputs() ώστε να βρεθούν οι τιμές των εξόδων για κάθε δοσμένη τιμή των εισόδων. Πιο συγκεκριμένα για τον υπολογισμό των εξόδων για κάθε δοσμένη τιμή των εισόδων καλείται η συνάρτηση doubleBufferSystem() για κάθε τιμή εισόδου. Δημιουργούνται δύο buffers με την συνάρτηση createBudders(), διαβάζονται οι τιμές των εισόδων για κάθε component του συστήματος καλώντας την setInputs() τα οποία μπορεί να είναι είτε είσοδοι του κυκλώματος είτε τιμές άλλων components οπότε και διαβάζεται η τιμή τους από τον input_buffer και εκτελούνται οι πράξεις για κάθε iteration καλώντας την συνάρτηση doTheMath().Η συνάρτηση doTheMath αφού εκτελέσει την πράξη διαβάζοντας τον πίνακα αληθείας της εκάστοτε πύλης ελέγχει εάν έχει αλλάξει η τιμή κάποιου component ώστε να γνωρίζουμε σε 2 διαδοχικές ίδιες τιμές όλων των components ότι το κύκλωμα έχει κατασταλάξει Για την τελική αποθήκευση των εξόδων είναι υπεύθυνες οι findSubsOutputs() που αποθηκεύει όλες τις εξόδους του συστήματος και η findTestbenchOutputs() που αποθηκεύει τις τιμές που ζητήθηκαν από το testbench . Η συνάρτηση doForEveryValue απλώς καλεί τις προηγούμενες για κάθε τιμή των inputs και τυπώνει τα αποτελέσματα σε ένα αρχείο εξόδου.

<u>Ακολουθεί ο τρόπος κλίσης των συναρτήσεων:</u>

```
doForEveryValue(){
        doubleBufferSystem(){
```

```
            setInputs(){}
            doTheMath(0{}
            saveLastValues(){}
        }
        findSubsOutputs(){}
        findTestbenchOutputs(){}
}
```

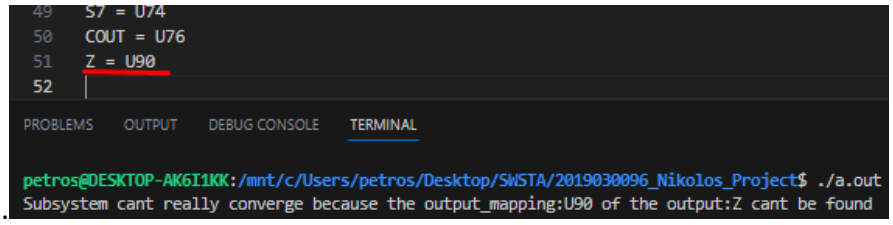
Running The Code

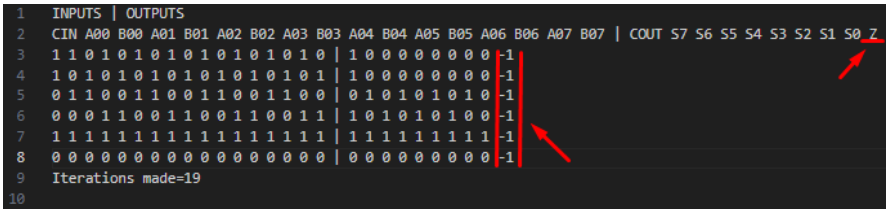
Αρκεί να τρέξουμε τις συναρτήσεις για το διάβασμα των 2 αρχείων getLibraryFromFIle() και getSubSystemFromFile() και την doForEveryValue(subsystem* sub)

Περιγραφή Αποτελέσματος:

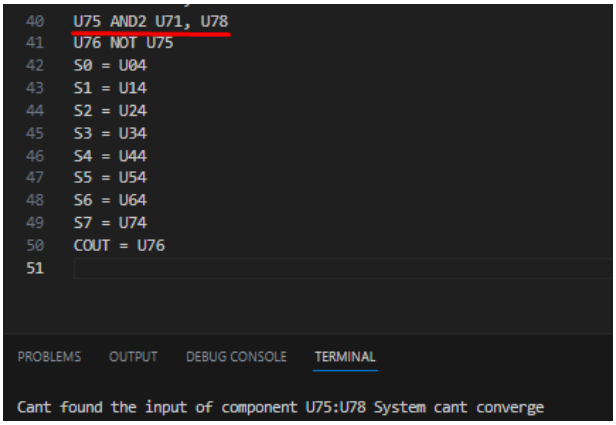
Στο αρχείο εξόδου του προγράμματος θα γραφτούν όλες οι είσοδοι και έξοδοι που δόθηκαν από το testbench. Προσοχή μία είσοδος η έξοδος του συστήματος που δεν ζητήθηκε από το testbench δεν θα εμφανιστεί στο αρχείο. Εμφανίζονται τα αποτελέσματα για όλες τις πιθανές τιμές των εισόδων. Έχουμε ορίσει την τιμή -1 ως άγνωστη (‘X’) και έχει αρχικοποιηθεί σε οποιαδήποτε είσοδο και έξοδο του κυκλώματος. Εάν βρεθεί κάποια τιμή εισόδου μιας πύλης να παραμένει -1 τότε και το αποτέλεσμα της πύλης θα παραμείνει -1. Δηλαδή σε περίπτωση που δεν δοθεί κάποια τιμή εισόδου από το testbench το κύκλωμα πιθανά δεν θα συγκλίνει και θα τυπωθεί ένα μήνυμα με το component στο οποίο δεν βρέθηκε το input θα παρατηρήσουμε επίσης αγνώστους -1 στις εξόδους. Ακόμη υπάρχει και η περίπτωση στο κύκλωμα μας να υπάρχει έξοδος που ζητάει τιμή κόμβου που δεν υπάρχει, τότε θα τυπωθεί μήνυμα ότι δεν μπορεί να βρεθεί η τιμή αυτής της εξόδου και στο αρχείο θα τυπωθεί ως -1

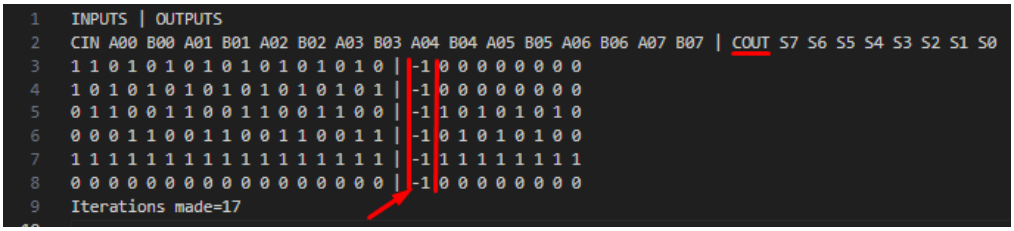
Στο παρακάτω παράδειγμα ζητείται η έξοδο Z χωρίς να υπάρχει κόμβος U90



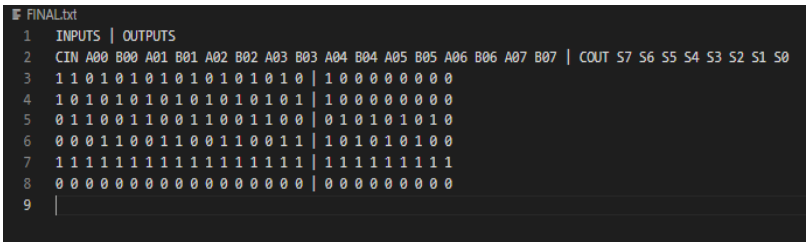


Στο παρακάτω παράδειγμα το component U75 παίρνει σαν είσοδο την έξοδο του component U78 που δεν υπάρχει ως αποτέλεσμα ένα μήνυμα τυπώνεται ότι δεν μπορεί να βρεθεί το input και στο testbench η τιμή του Cout που επηρεάζεται από το component U75 είναι -1 (άγνωστη).





Παρακάτω βλέπουμε ένα παράδειγμα ενός κυκλώματος που συγκλίνει.



Συμπεράσματα:

Έχουμε καταφέρει μέσω των ασκήσεων κατά τη διάρκεια του εξαμήνου και του συγκεκριμένου project να αποκτήσομε μία εμπειρία στη δημιουργία κατάλληλων δυναμικών δομών για την αποθήκευση διαφόρων μεγεθών συστημάτων και σαφώς την δημιουργία συναρτήσεων ικανών να διαβάσουν διαφορετικά υποσυστήματα με την προϋπόθεση ότι ακολουθούν μία κοινή δομή στον ορισμό τους στο αρχείο που δόθηκαν. Ακόμη είναι σημαντικό να αναφερθεί πως μέσα από την διαδικασία ελέγχου των εργασιών μας παρουσία του καθηγητή αποκτήθηκαν και γενικότερες γνώσεις προγραμματισμού όπως η δημιουργία κατάλληλων σχολίων και καλύτερων αναφορών.