Αναφορά Project

ΗΡΥ 419 - ΗΡΥ 608 - Ανάπτυξη Εργαλείων CAD για Σχεδίαση Ολοκληρωμένων Κυκλωμάτων 2022- 2023

ΑΙΚΑΤΕΡΙΝΗ ΤΣΙΜΠΙΡΔΩΝΗ:2018030013

Το πρόγραμμα είναι ένα πρόγραμμα στη γλώσσα C που περιλαμβάνει διάφορες δομές δεδομένων και συναρτήσεις για την επεξεργασία με ψηφιακά κυκλώματα Ο στόχος του κώδικα είναι η εκτέλεση μιας δοκιμης ή ενός συνόλου δοκιμών (testbench) για ένα σύστημα. Ο κώδικας δέχεται εισόδους αρχεια , υπολογίζει τις αντίστοιχες εξόδους και αποθηκεύει τα αποτελέσματα.

Βιβλιοθήκες:

Το πρόγραμμα περιλαμβάνει αρκετές βιβλιοθήκες που παρέχουν απαραίτητες συναρτήσεις και τύπους δεδομένων:
<stdio.h>: Παρέχει συναρτήσεις εισόδου/εξόδου.
<stdlib.h>: Παρέχει συναρτήσεις ανάθεσης μνήμης και άλλες γενικής χρήσης συναρτήσεις.
<string.h>: Παρέχει συναρτήσεις για την επεξεργασία συμβολοσειρών.
<ctype.h>: Παρέχει συναρτήσεις για τη χειρισμό χαρακτήρων.
<math.h>: Παρέχει μαθηματικές συναρτήσεις.
<stdbool.h>: Παρέχει τον τύπο δεδομένων boolean και σταθερές.
<time.h>: Παρέχει συναρτήσεις για την εργασία με τον χρόνο.

Ορισμοί:

Ο κώδικας ορίζει αρκετές σταθερές χρησιμοποιώντας την προεπεξεργασία #define:
MAX_DEFINITION: Αντιπροσωπεύει το μέγιστο μήκος ενος ορισμού.
MAX_INPUTS: Αντιπροσωπεύει το μέγιστο αριθμό εισόδων.
MAX_CHARACTERS: Αντιπροσωπεύει το μέγιστο αριθμό χαρακτήρων.
MAX_COMPONENTS: Αντιπροσωπεύει το μέγιστο αριθμό συστατικών.
BILLION: Αντιπροσωπεύει την τιμή ενός δισεκατομμυρίου.

Δομές:

Ο κώδικας ορίζει αρκετές δομές για να αναπαραστήσει διάφορες πτυχές ενός ψηφιακού κυκλώματος:
DEC: Αναπαριστά ένα ψηφιακό στοιχείο και περιλαμβάνει το όνομά του, τις είσοδος θύρες, τον πίνακα αληθείας του και άλλες πληροφορίες όπως τον αριθμό των στοιχείων του πίνακα αλήθειας καθώς και τον αριθμό των εισόδων πού έχει ένα στοιχείο .
DEC_LIST: Αναπαριστά μια λίστα ψηφιακών στοιχείων .
Component: Αναπαριστά ένα στοιχειο στον κατάλογο δικτύου και περιλαμβάνει το ID του, τον τύπο του, τις είσοδος θύρες, τις συνδέσεις του και άλλες σχετικές πληροφορίες.
comp_array: Αναπαριστά έναν πίνακα των στοιχείων αυτών.
Output_Dec: Αναπαριστά την έξοδο ενός στοιχείου, περιλαμβάνοντας το όνομα της εξόδου και τον δείκτη του στοιχειου προέλευσης.
OutputArray: Αναπαριστά έναν πίνακα δηλώσεων εξόδου.
netsystemFile: Αναπαριστά το αρχείο συστήματος δικτύου και περιλαμβάνει έναν δείκτη προς έναν κατάλογο δικτύου και έναν πίνακα εξόδων.
testinputs: Αναπαριστά τις δοκιμαστικές εισόδους και περιλαμβάνει το όνομα της εισόδου, τις τιμές της και τον αριθμό των τιμών.

test_array: Αναπαριστά έναν πίνακα δοκιμαστικών εισόδων.
testbench: Αναπαριστά τον πίνακα δοκιμαστικών και περιλαμβάνει ένα δείκτη προς έναν πίνακα δοκιμαστικών, τον μέγιστο αριθμό τιμών πειραματικού εργαστηρίου, τα ονόματα θυρών εξόδου και τον αριθμό των εξόδων.
result: Αναπαριστά τα αποτελέσματα μιας δοκιμής και περιλαμβάνει πίνακες για τα ονόματα εισόδων θυρών, τις τιμές εισόδων, τα ονόματα εξόδων θυρών, τις τιμές εξόδων και τους αντίστοιχους μετρητές.
resultarray: Αναπαριστά έναν πίνακα αποτελεσμάτων.

Συναρτήσεις:

Συναρτήσεις που αφορούν τις δομές:

Μετά τους ορισμούς των δομών, ο κώδικας περιλαμβάνει διάφορες συναρτήσεις που εκτελούν λειτουργίες στις ορισμένες δομές. Αυτές οι λειτουργίες περιλαμβάνουν τις **createResult, addName, addNameOut, addResulttoArray, isLineEmpty, addComponent, addOutput, add_to_testinputs, add_to_test_array, add_to_testbench, addDEC** και **printResultArray.**

Η συνάρτηση **createResult** δυναμικά εκχωρεί μνήμη για μια δομή result και τους πίνακές της βάσει του καθορισμένου αριθμού εισόδων και εξόδων. Οι συναρτήσεις addName και addNameOut προσθέτουν ονόματα εισόδου και εξόδου στους αντίστοιχους πίνακες της δομής result.
Η συνάρτηση **addResulttoArray** προσθέτει μια δομή result στο resultarray.

Η συνάρτηση **addComponent** προσθέτει ένα εξάρτημα στη δομή **comp_array**. Ελέγχει αν ο πίνακας είναι έγκυρος και αν έχει επιτευχθεί το μέγιστο αριθμός εξαρτημάτων. Η συνάρτηση δημιουργεί έναν νέο δείκτη εξαρτήματος και αντιγράφει τα δεδομένα από την παράμετρο **new_component** σε αυτόν.

Η συνάρτηση **addOutput** προσθέτει μια δήλωση εξόδου στη δομή OutputArray. Ελέγχει αν ο πίνακας είναι έγκυρος και αν έχει επιτευχθεί το μέγιστο αριθμός εξόδων. Η συνάρτηση δημιουργεί ένα νέο δείκτη **Output_Dec**, εκχωρεί μνήμη για το πεδίο Output και αντιγράφει το όνομα και την προέλευση της εξόδου στην εκχωρημένη μνήμη.

Άλλες συναρτήσεις όπως **add_to_testinputs**, **add_to_test_array** και **add_to_testbench** προσθέτουν δοκιμαστικές εισόδους και πίνακες δοκιμών στις αντίστοιχες δομές.
Η συνάρτηση **addDEC** προσθέτει ένα νέο στοιχείο **DEC** στη δομή **DEC_LIST**. Ελέγχει αν η λίστα είναι έγκυρη και αν έχει φτάσει στη μέγιστη χωρητικότητά της πριν εκχωρηθεί δυναμικά μνήμη για το νέο στοιχείο DEC.

Τέλος, η συνάρτηση printResultArray εκτυπώνει το περιεχόμενο μιας δομής **resultarray**, εμφανίζοντας τον αριθμό των αποτελεσμάτων, τον αριθμό των εισόδων και των εξόδων για κάθε αποτέλεσμα, και τις αντίστοιχες τιμές εισόδου και εξόδου.

Αυτές ήταν οι βοηθητικές συναρτήσεις για την επεξεργασία των δομών τώρα θα παρατεθούν οι κυρίες συναρτήσεις όπου αφορούν την επίτευξη του project.

Συναντήσεις που αφορούν την ανάγνωση των αρχείων εισόδου και την εκχώρηση του σε δομές:

Η πρώτη συνάρτηση **CutandStoreComponentFile**(FILE* file) διαβάζει ένα αρχείο component.txt και αναλύει δηλώσεις στοιχείων(πυλών), αποθηκεύοντάς αυτες σε μια DEC_LIST. Εδώ. Κάθε γραμμή του αρχείου διαβάζεται, και αν η γραμμή ξεκινά με "COMP", θεωρείται ότι είναι δήλωση. Η δήλωση χωρίζεται από διακριτά σύμβολα χρησιμοποιώντας τον χαρακτήρα ";" ως διαχωριστικό. Το όνομα του στοιχείου, οι είσοδοι και οι τιμές του πίνακα αλήθειας εξάγονται από τα διακριτά σύμβολα και αποθηκεύονται σε μια δομή DEC, η οποία στη συνέχεια προστίθεται στη DEC_LIST. Αυτή η λειτουργία επιστρέφει μια δομη DEC_LIST.

Η δεύτερη συνάρτηση **trimWhitespace**(char* str) κόβει τα κενά πριν και μετά από μια συμβολοσειρά. Βρίσκει τα αρχικά και τα τελικά ευρήματα των μη κενών χαρακτήρων, μετακινεί αυτούς τους χαρακτήρες στην αρχή της συμβολοσειράς και τερματίζει με null τη συμβολοσειρά στο τέλος αυτών των χαρακτήρων.Χρησιμοποιείται στο διάβασμα των αρχείων ώστε να κόβει τα κενά.

Η τρίτη συνάρτηση **parseFileAndStoreNetsystem**(const char* filename) αναλύει ένα αρχείο και αποθηκεύει τις δηλώσεις του συστήματος σε μια δομή netsystemFile. Η λειτουργία ανοίγει το αρχείο για ανάγνωση και το διαβάζει γραμμή προς γραμμή. Αν μια γραμμή περιέχει μια δήλωση στοιχείου, αυτή αναλύεται και αποθηκεύεται ως δομή Component, η οποία στη συνέχεια προστίθεται στη λίστα υποσυνόλων (subnetlist) του netsystemFile. Αν μια γραμμή περιέχει μια δήλωση εξόδου, αυτή αναλύεται και αποθηκεύεται ως δομή Output_Dec, η οποία στη συνέχεια προστίθεται στον πίνακα εξόδων του netsystemFile. Αυτή η λειτουργία επιστρέφει μια δομή netsystemFile.

Η τέταρτη συνάρτηση **parseFileAndStoreTestBench**(const char* filename) διαβάζει ένα αρχείο και αναλύει τις δηλώσεις του testbench , αποθηκεύοντάς αυτες σε μια δομή testbench. Αυτή η λειτουργία πρώτα ανοίγει το αρχείο για ανάγνωση. Στη συνέχεια, διαβάζει το αρχείο γραμμή προς γραμμή, αναλύοντας και αποθηκεύοντας κάθε δήλωση του testbench στη δομή testbench. Αυτή η λειτουργία επιστρέφει μια δομή testbench.

Συναρτήσεις για την επεξεργασία της εξόδου:

Η συνάρτηση **makethearrayforInputs** δημιουργεί έναν πίνακα με τιμές εισόδου από τον πίνακα δοκιμών (testbench) για ένα συγκεκριμένο σετ δοκιμής.
Παράμετροι:tb: Το αντικείμενο του testbench.c: Ο δείκτης (index) της τιμής εισόδου που θέλουμε να εξάγουμε.
Αρχικά,κατανέμεται μνήμη για τον πίνακα array μεγέθους tb.tb->num_tests (το πλήθος των δοκιμών).Έπειτα ,γεμίζει ο πίνακας array με τιμές από τον πίνακα δοκιμών (tb.tb->tinput) για τον συγκεκριμένο δείκτη c.Επιστρέφει τον πίνακα array.
Ουσιαστικά, η συνάρτηση αυτή βοηθά στην εξαγωγή των τιμών εισόδου από τον πίνακα δοκιμών για ένα συγκεκριμένο σετ δοκιμής.

Η συνάρτηση **init_theOutputArray** αρχικοποιεί τον πίνακα nextBuffer με τιμές εισόδου του testbench και αναγνωριστικά στοιχείων (component IDs).

Παράμετροι:nextBuffer: Ο πίνακας nextBuffer που θα ενημερωθεί,i: Ο δείκτης (index) του στοιχείου που επεξεργάζεται.,nfile: Το αντικείμενο αρχείου δικτύου (network system file),dec_list: Το αντικείμενο λίστας αποφάσεων (decision list),tb: Το αντικείμενο του testbench,c: Ο δείκτης (index) της τιμής εισόδου που θέλουμε να εξάγουμε.
Δημιουργείται ο πίνακας array1 με τις τιμές εισόδου από τη συνάρτηση makethearrayforInputs για τον δείκτη c.Αποθηκεύεται το αναγνωριστικό στοιχείου (component ID) στην πρώτη στήλη του πίνακα nextBuffer.Για κάθε είσοδο του στοιχείου:Αν το στοιχείο δεν έχει συνδέσεις, αντιστοιχίζονται οι τιμές εισόδου του απευθείας.Αν το στοιχείο έχει όλες τις συνδέσεις, οι εισόδοι σημειώνονται ως -1.Αλλιώς αν το στοιχείο έχει κάποιες συνδέσεις, αντιστοιχίζεται η τιμή εισόδου αν υπάρχει αντιστοιχία, αλλιώς η είσοδος σημειώνεται ως -1.Επιστρέφεται ο ενημερωμένος πίνακας nextBuffer.
Η συνάρτηση **CompleteArray** συμπληρώνει τις τιμές συνδέσεων στον πίνακα currentBuffer βάσει της καθορισμένης τιμής.
Παράμετροι:currentBuffer: Ο πίνακας currentBuffer που θα ενημερωθεί,i: Ο δείκτης (index) του στοιχείου που επεξεργάζεται,value: Η τιμή που θα αντιστοιχιστεί στις απουσιάζουσες συνδέσεις,dec_list: Το αντικείμενο λίστας αποφάσεων (decision list),nfile: Το αντικείμενο αρχείου δικτύου (network system file).
Ψάχνει να βρει εάν κάποια σύνδεση αντιστοιχεί στο στοιχείο που επεξεργάζεται ο πίνακας currentBuffer.Εάν η αντίστοιχη είσοδος δεν είναι -1, αυξάνεται ο δείκτης k κατά 1 για να βρει την επόμενη είσοδο.Αν είναι - 1 τότε ενημερώνονται οι συνδέσεις με την καθορισμένη αυτη τιμή.Επιστρέφεται ο ενημερωμένος πίνακας currentBuffer.

Η συνάρτηση **takeallthathastodowithmissing** συλλέγει αναδρομικά όλα τα στοιχεία που έχουν συνδέσεις με το συγκεκριμένο στοιχείο.
Παράμετροι:array: Ο πίνακας για την αποθήκευση των αναγνωριστικών συστατικών (component IDs),i: Ο δείκτης (index) του στοιχείου που επεξεργάζεται.circuit: (circuit component array),counter: Η τρέχουσα τιμή του μετρητή.

Ψάχνει να βρει εάν καποια σύνδεση αντιστοιχεί στο στοιχείο που επεξεργάζεται ο πίνακας circuit:
Αποθηκεύεται το αναγνωριστικό του στοιχείου στον πίνακα array.Αυξάνεται ο μετρητής counter.
Κάνει αναδρομική κλήση για να χειριστεί τις ολες της συνδέσεις.
Επιστρέφεται η ενημερωμένη τιμή του μετρητή counter.

Η συνάρτηση **checkMissingConnections** ελέγχει εάν ένα συστατικό έχει ελλιπής συνδέσεις, δηλαδή αν κάποια σύνδεση του δεν έρχεται από πουθενά.
Παράμετροι:circuit: Ο πίνακας συστατικών κυκλώματος (circuit component array), componentIndex: Ο δείκτης (index) του συστατικού που ελέγχεται.
Ελέγχει αν οι συνδέσεις στο component με δείκτη index υπάρχουν μέσα στο κύκλωμα.Εάν βρεθεί η σύνδεση σε κάποιο άλλο συστατικό, η μεταβλητή connectionFound γίνεται True και διακόπτεται ο έλεγχος.Εάν η σύνδεση δεν βρεθεί σε κανένα άλλο συστατικό, επιστρέφεται False για να υποδείξει την απουσία σύνδεσης.

Η συνάρτηση **checkarray** επεξεργάζεται τον πίνακα **currentBuffer** για να ενημερώσει τα στοιχεία τα οποία δεν έχουν ενημερωθεί ακόμα λόγω του ότι μπορεί να λείπει κάποιο component από το κύκλωμα είτε το κύκλωμα δόθηκε να έχει ανακατεμένα τα components του.Αρχικά, η συνάρτηση δημιουργεί δύο μεταβλητές c και array2. Η μεταβλητή c

χρησιμοποιείται ως μετρητής για τον αριθμό των ελλιπων συνδέσεων(όταν δηλαδή κάποιο component δεν έχει σύνδεση επειδή λείπει το component που συνδέεται) ενώ ο πίνακας array2 χρησιμοποιείται για τη συλλογή των αναγνωριστικών των στοιχείων που έχουν αυτες τις συνδεσεις καθως και οποια αλλα συνδεονται με αυτα.
Η συνάρτηση επαναλαμβάνει για κάθε στοιχείο του δικτύου στον πίνακα subnetlist. Ελέγχει αν το στοιχείο στον πίνακα currentBuffer έχει ήδη επεξεργαστεί. Αν έχει, προχωράει στο επόμενο στοιχείο. Αν όχι, εξετάζει τις εισόδους του στοιχείου στον πίνακα currentBuffer.

Αν μια είσοδοι έχουν ήδη τιμή στον πίνακα currentBuffer, τότε αποθηκεύονται οι τιμες αυτες στον πίνακα array1 και αυξάνεται ο μετρητής counter. Αν η είσοδος δεν έχει τιμή (-1), αυξάνεται ο μετρητής counter1 που υπολογίζει τον αριθμό των απουσιάζουσων συνδέσεων για το εν λόγω στοιχείο.

Έπειτα, εκτελείται η συνάρτηση findtruthtableindex για να βρεθεί ο δείκτης στον πίνακα αληθείας για το εν λόγω στοιχείο, με βάση τις τιμές που έχουν εισαχθεί στον πίνακα array1 και τον μετρητή counter. Αν ο δείκτης έχει έγκυρη τιμή και ο μετρητής counter1 είναι μηδέν, τότε βρίσκεται η αντίστοιχη τιμή στον πίνακα αληθείας(dec_list.dec_list[k].truth_table[id->result]) και ενημερώνεται ο πίνακας currentBuffer με αυτήν την τιμή.Στη συνέχεια, η συνάρτηση ελέγχει αν υπάρχουν ελλιπής συνδέσεις για το εν λόγω στοιχείο. Αν δεν υπάρχουν, τίποτα δεν γίνεται. Αν υπάρχουν, τότε προστίθενται τα αναγνωριστικά των στοιχείων στον πίνακα array2 και εκτελείται η συνάρτησητakeallthathastodowithmissing για να συλλεχθούν όλα τα αντικείμενα που σχετίζονται με τις ελλιπής συνδέσεις.
Στη συνέχεια, γίνεται έλεγχος για κάθε στοιχείο του υποδικτύου. Αν το μήκος του πίνακας array2 έχει τιμή μεγαλύτερη του μηδενός, τότε ελέγχεται αν το αναγνωριστικό του στοιχείου υπάρχει στον πίνακα array2. Αν ναι, δεν γίνεται τίποτα. Αν όχι, γίνεται επεξεργασία των στοιχείων που έχουν απουσιάζουσες συνδέσεις. Η διαδικασία είναι αντίστοιχη με την προηγούμενη, αλλά εκτελείται μόνο για τα στοιχεία που δεν ανήκουν στον πίνακα array2.

Τέλος, η συνάρτηση επιστρέφει τον ενημερωμένο δείκτη idx.

Στην ουσία αυτή η συνάρτηση χρησιμοποιείται Αφού έχουμε τελειώσει τη βασική επεξεργασία για την έξοδο ώστε να κάνει έναν τελευταίο έλεγχο αν όλα τα στοιχεία μέσα στον πίνακα έχουν πάρει κάποια τιμή. Κι αν για κάποιο λόγο , είτε λόγω ελλειπων συνδέσεων, είτε λόγω ότι το κύκλωμα έχει δοθεί ανακατεμένο και κάποιες τιμές δεν έχουν επεξεργαστεί σωστά στο currentBuffer, η συνάρτηση αυτή φροντίζει να να μπουν οι σωστές τιμές στο current buffer και να ενημερωθεί.

Η συνάρτηση **update_theOutput** χρησιμοποιείται για να ενημερώσει τις τιμές εξόδου στον πίνακα currentBuffer βάσει των τιμών εισόδου και των πληροφοριών για τα στοιχεία που περιέχονται στη δομή netsystemFile και στη λίστα DEC_LIST.Αρχικά δημιουργείται ο μετρητής counter1 και αρχικοποιείται σε μηδέν.Επαναλαμβάνεται για κάθε στοιχείο της λίστας subnetlist στη δομή netsystemFile.Μεσα στο for δημιουργείται ένας δυναμικός πίνακας array1 μεγέθους ίσου με τον αριθμό των εισόδων του τρέχοντος στοιχείου.
Εάν η τιμή της εισόδου είναι -1, τότε ο βρόχος διακόπτεται και μεταβαίνουμε στο επόμενο στοιχείο.

Άρα αποθηκεύονται μόνο οι τιμές εισόδου που δεν είναι - 1 δηλαδή αυτές που έχουμε αρχικοποίησει με την **init_theOutputArray** αρχικά και ύστερα όλες αυτές οι τιμές που παίρνουν όταν καλούμε την **CompleteArray** λίγο πιο κάτω **.**Οι τιμές των εισόδων του τρέχοντος στοιχείου αποθηκεύονται στον πίνακα array1 και Καλείται η συνάρτηση findtruthtableindex για να βρεθεί ο δείκτης στον πίνακα αληθείας για τις τιμές εισόδου.
Εάν ο δείκτης id->result δεν είναι -1, τότε υπάρχει έγκυρη εγγραφή στον πίνακα αληθείας για τις τιμές εισόδου.
Τσεκάρει επίσης το όνομα του τρέχοντος στοιχείου της λίστας dec_list.dec_list ταιριάζει με τον τύπο του τρέχοντος στοιχείου της λίστας subnetlist, τότε βρέθηκε η αντίστοιχη εγγραφή στον πίνακα αληθείας.Η τιμή εξόδου αποθηκεύεται στο currentBuffer στη θέση currentBuffer[i][nfile.subnetlist->components[i].numofinputs + 1].

Καλείται η συνάρτηση CompleteArray για να ενημερώσει τον currentBuffer με βάση τη νέα τιμή εξόδου.Και αυτό γίνεται σε βρόγχο μέχρι να συμπληρωθεί ο πίνακας.
Τέλος,καλείται η συνάρτηση checkarray για να πραγματοποιηθούν επιπλέον υπολογισμοί στον currentBuffer αν χρειάζεται.

Αυτή είναι η βασική συνάρτηση η οποία είναι υπεύθυνη για τους υπολογισμούς της εξόδου καθώς και τη συμπλήρωση των εισόδων για τα στοιχεία που έχουνε συνδέσεις.

Η συνάρτηση **finalout** υλοποιεί τον αλγόριθμο για την τελική έξοδο των αποτελεσμάτων και χρησιμοποιεί την τεχνική του **double buffering**

Αρχικά, δημιουργούνται οι δύο πίνακες, στη συνέχεια, γίνεται η κλήση της συνάρτησης init_theOutputArray για κάθε στοιχείο του πίνακα currentBuffer. Αυτή η συνάρτηση χρησιμοποιείται για την αρχικοποίηση του πίνακα currentBuffer με τις αρχικές τιμές των εισόδων του testbench για την πρωτη επαναληψη γινεται μονο αυτο δηλαδη για το πρωτο set .Στη συνέχεια, εκτελείται η πρώτη επανάληψη της while . Όπου καλείται η init_theOutputArray για τον nextBuffer ώστε να πάρει τις αρχικές τιμές της επόμενης κατάστασης και ταυτόχρονα καλείται η update_theOutput για να τον currentBuffer επεξεργαστεί τις τιμές της τωρινής κατάστασης και να βγάλει το Output.Έτσι γίνονται δύο διαδικασίες και επιτρέπει τη χρήση ενός συνόλου δεδομένων ενώ συλλέγεται ένα άλλο σύνολο δεδομένων.Αμέσως μετά την ολοκλήρωση του υπολογισμού, ακολουθεί η αλλαγή των πινάκων με τη χρήση του "swap"Συγκεκριμένα, ο nextBuffer γίνεται ο currentBuffer και ο currentBuffer γίνεται ο nextBuffer.

Αυτή η διαδικασία επαναλαμβάνεται κατά τη διάρκεια των επόμενων επαναλήψεων της while Με κάθε επανάληψη, ο currentBuffer υπολογίζει τις τιμές του κυκλωματος και ο nextBuffer παίρνει τις αρχικές τιμές του επομενου , και στη συνέχεια οι πίνακες ανταλλάσσονται ώστε να ετοιμαστεί ο nextBuffer για το επόμενο βήμα.
Πριν τη διαδικασία του swap αποθηκεύεται το output σε ένα struct result array ώστε να μπορούμε να το τυπώσουμε ευκολότερα στην έξοδο. Τέλος ,εκτυπώνει τον συνολικο αριθμο υπολογισμών και του συνολικού χρόνου εκτέλεσης.

Αλγόριθμος για το πώς δουλεύει το double buffering:

1. Δημιουργήστε δύο σετ δεδομένων, Έστω "Currentset" και " Nextset". Το Currentset περιέχει τα δεδομένα που χρησιμοποιούνται για την επεξεργασία αποτελεσμάτων των πυλών, ενώ το Nextset χρησιμοποιείται για τη συλλογή νέων δεδομένων.

2. Αρχικά, αποθηκεύστε τα νεα δεδομένα στο Nextset, και τις αλλαγές που έχετε κάνει στο Currentset.

3. Ενεργοποιήστε το Currentset για να εφαρμόσετε τα δεδομένα στις πύλες και να εκτελέσετε το testbench.

4. Όταν ολοκληρωθεί η εκτέλεση του testbench και είστε έτοιμοι για την επόμενη επανάληψη, αντιγράψτε τα δεδομένα από το Nextset στο Currentset .

5. Επαναλάβετε τα βήματα 3 και 4 για να συνεχίσετε την ανανέωση των δεδομένων και την εκτέλεση του testbench.

Ο αλγόριθμος του double buffering επιτρέπει να συλλέξετε νέα δεδομένα ενώ εκτελείτε το testbench, ενώ ταυτόχρονα παραθέτουμε αποτελέσματα καθώς χρησιμοποιείτε το Currentset για τον υπολογισμο των πυλών.

Αποτελέσματα:

Κοινό αρχείο εισόδου για όλα όσο δοκιμάστηκαν είναι το component.txt:

```
%% THIS IS THE COMPONENT LIBRARY, COMPRISING OF GATES
** COMPONENT LIBRARY
COMP NOT ; IN:P ; 1, 0
COMP AND2 ; IN:P,Q ; 0, 0, 0, 1
COMP NAND2 ; IN:P,Q ; 1, 1, 1, 0
COMP OR2 ; IN:P,Q ; 0, 1, 1, 1
COMP NOR2 ; IN:P,Q ; 1, 0, 0, 0
COMP XOR2 ; IN:P,Q ; 0, 1, 1, 0
COMP XNOR2 ; IN:P,Q ; 1, 0, 0, 1
```

Για Full adder-1bit

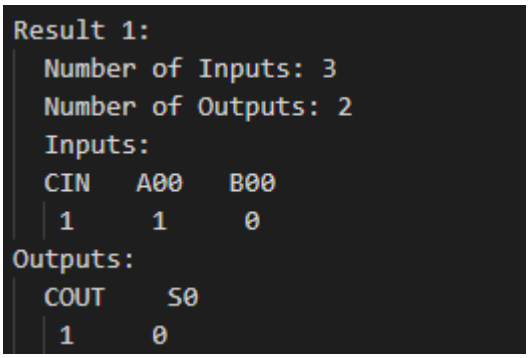
Το netlist.txt είναι :

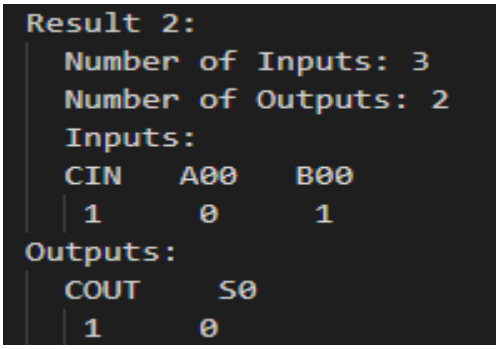
```
U01 NAND2 A00 B00
U02 XOR2 A00 B00
U03 NAND2 U02 CIN
U04 XOR2 U02 CIN
U05 NAND2 U01 U03
S0 = U04
COUT = U05
```

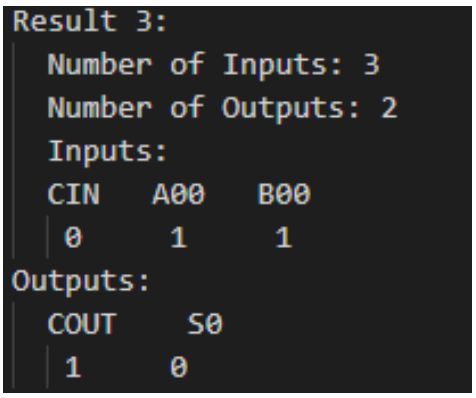
Ενώ το testbench.txt:

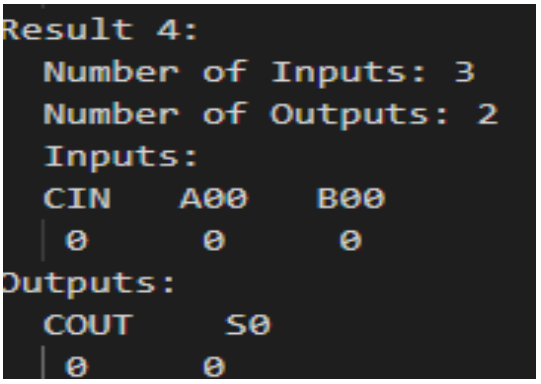
```
IN
CIN  1, 1, 0, 0, 1, 0
A00  1, 0, 1, 0, 1, 0
B00  0, 1, 1, 0, 1, 1
OUT
COUT
S0
```

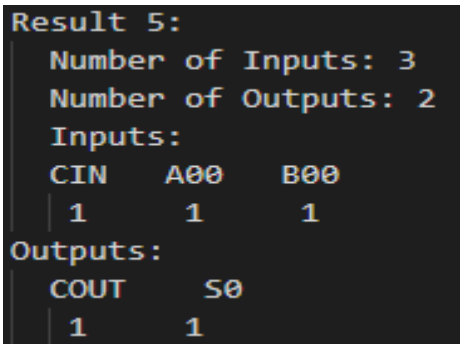
Αποτελέσματα του προγράμματος για αυτά τα αρχεία εισόδου είναι:

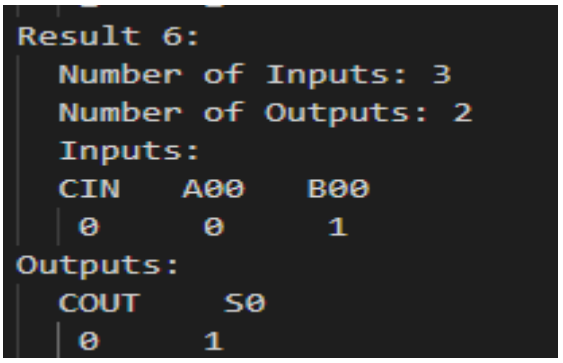












Όπως μπορούμε να δούμε και από το παρακάτω πίνακα αλήθειας τα αποτελέσματα για να σωστά.

A	B	C_in	S	C_out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Για Full adder-3bit
Το netlist.txt είναι :

U01 NAND2 A00 B00
U02 XOR2 A00 B00
U03 NAND2 U02 CIN
U04 XOR2 U02 CIN
U05 NAND2 U01 U03
U06 NAND2 A01 B01
U07 XOR2 A01 B01
U08 NAND2 U07 U05
U09 XOR2 U07 U05
U10 NAND2 U06 U09
U11 NAND2 A02 B02
U12 XOR2 A02 B02
U13 NAND2 U12 U10
U14 XOR2 U12 U10
U15 NAND2 U11 U13
S0 = U04
S1 = U09
S2 = U14
COUT = U15

Ενώ το testbench.txt:

IN
CIN 1, 1, 0, 0, 1, 0
A00 1, 0, 1, 0, 1, 0
B00 0, 1, 1, 0, 1, 0
A01 1, 0, 0, 1, 1, 0
B01 0, 1, 0, 1, 1, 0
A02 1, 0, 1, 0, 1, 0
B02 0, 1, 1, 0, 1, 0
OUT
COUT
S2
S1
S0

Αποτελέσματα για αυτά τα αρχεία είναι τα παρακάτω:

```
Inputs:
CIN   A00   B00   A01   B01   A02   B02
 1     1     0     1     0     1     0
Outputs:
COUT    S2    S1    S0
 1     0     0     0
```

```
  Inputs:
  CIN   A00   B00   A01   B01   A02   B02
   1     0     1     0     1     0     1
Outputs:
  COUT    S2    S1    S0
   1     0     0     0
```

```
  Inputs:
  CIN   A00   B00   A01   B01   A02   B02
   0     1     1     0     0     1     1
Outputs:
  COUT    S2    S1    S0
   1     0     1     0
```

```
Result 4:
  Number of Inputs: 7
  Number of Outputs: 4
  Inputs:
  CIN   A00   B00   A01   B01   A02   B02
   0     0     0     1     1     0     0
Outputs:
  COUT    S2    S1    S0
   0     1     0     0
```

```
  Number of Outputs: 4
  Inputs:
  CIN   A00   B00   A01   B01   A02   B02
   1     1     1     1     1     1     1
Outputs:
  COUT    S2    S1    S0
   1     1     1     1
```

```
  Inputs:
  CIN   A00   B00   A01   B01   A02   B02
   0     0     0     0     0     0     0
Outputs:
  COUT    S2    S1    S0
   0     1     0     0
Total calculations for all tests: 180
Total time taken to execute in seconds: 0.000185
```

```
C:\Users\Boss\Documents\Vstudio\Project1\Final_Project
Number of calculations: 30 for test: 1
Time taken to execute in seconds: 0.000050 for test: 1
Number of calculations: 30 for test: 2
Time taken to execute in seconds: 0.000030 for test: 2
Number of calculations: 30 for test: 3
Time taken to execute in seconds: 0.000028 for test: 3
Number of calculations: 30 for test: 4
Time taken to execute in seconds: 0.000027 for test: 4
Number of calculations: 30 for test: 5
Time taken to execute in seconds: 0.000025 for test: 5
Number of calculations: 30 for test: 6
```

```
Number of calculations: 30 for test: 6
Time taken to execute in seconds: 0.000025 for test: 6
```

Όπως φαίνεται και από τα αποτελέσματα και άμα χρησιμοποιήσουμε και τον πίνακα για τον πίνακα αλήθειας για τον που έχουμε παραπάνω με τη λογική ότι το Cout Θα μπαίνει ως είσοδος στο Cin. Βλέπουμε τα αποτελέσματα είναι σωστά.

Τέλος για το netist που μας δόθηκε για να το δοκιμάσουμε τα αποτελέσματα είναι τα παρακάτω:

```
Inputs:
CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
 1     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0
Outputs:
COUT    S7    S6    S5    S4    S3    S2    S1    S0
 1     0     0     0     0     0     0     0     0
```

```
CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
 1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1
Outputs:
COUT    S7    S6    S5    S4    S3    S2    S1    S0
 1     0     0     0     0     0     0     0     0
```

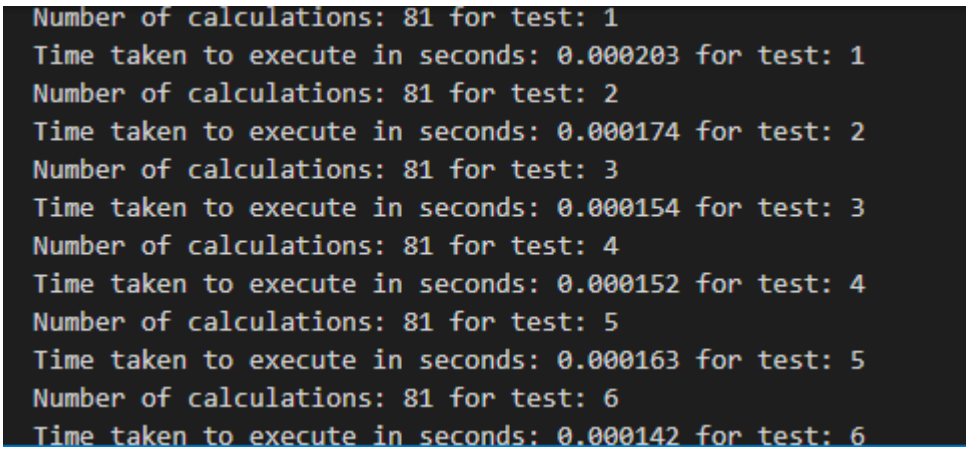
```
Inputs:
CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
 0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0
Outputs:
COUT    S7    S6    S5    S4    S3    S2    S1    S0
 0     1     0     1     0     1     0     1     0
```

```
Inputs:
CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
 0     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1
Outputs:
COUT    S7    S6    S5    S4    S3    S2    S1    S0
 1     0     1     0     1     0     1     0     0
```

```
Inputs:
CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
 1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
Outputs:
COUT    S7    S6    S5    S4    S3    S2    S1    S0
 1     1     1     1     1     1     1     1     1
```

```
Inputs:
CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
 0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
Outputs:
COUT    S7    S6    S5    S4    S3    S2    S1    S0
 0     0     0     0     0     0     0     0     0
```

```
Total calculations for all tests: 486
Total time taken to execute in seconds: 0.000989
```



Πάλι αν κάνουμε επαλήθευση με το με τον πίνακα αλήθειας που έχει δοθεί παραπάνω θα δούμε ότι τα αποτελέσματα βγαίνουν όπως πρέπει.

Τώρα για να επαληθεύσω ότι το κύκλωμα δουλεύει ακόμη και αν ανακατευτούν οι πύλες μετέτρεψα το Netflist μας δόθηκε.

U01 NAND2 A00 B00
U03 NAND2 U02 CIN
U04 XOR2 U02 CIN
U05 NAND2 U01 U03
U11 NAND2 A01 B01
U02 XOR2 A00 B00
U12 XOR2 A01 B01
U13 NAND2 U12 U05
U14 XOR2 U12 U05
U15 NAND2 U11 U13
U23 NAND2 U22 U15
U24 XOR2 U22 U15
U25 NAND2 U21 U23
U31 NAND2 A03 B03
U32 XOR2 A03 B03
U33 NAND2 U32 U25
U21 NAND2 A02 B02
U22 XOR2 A02 B02
U34 XOR2 U32 U25
U35 NAND2 U31 U33
U41 NAND2 A04 B04
U42 XOR2 A04 B04
U43 NAND2 U42 U35
U44 XOR2 U42 U35
U45 NAND2 U41 U43
U51 NAND2 A05 B05
U52 XOR2 A05 B05
U54 XOR2 U52 U45
U55 NAND2 U51 U53
U53 NAND2 U52 U45
U61 AND2 A06 B06
U62 XOR2 A06 B06

U63 AND2 U62 U55
U64 XOR2 U62 U55
U65 OR2 U61 U63
U71 NAND2 A07 B07
U72 XOR2 A07 B07
U73 NAND2 U72 U65
U76 NOT U75
U74 XOR2 U72 U65
U75 AND2 U71 U73
S0 = U04
S1 = U14
S2 = U24
S3 = U34
S4 = U44
S5 = U54
S6 = U64
S7 = U74
COUT = U76
Ενδεικτικά τα αποτελέσματα είναι τα ίδια παραθέτω μερικά για να το διαπιστώσετε:

```
  CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
   1     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0
Outputs:
  COUT    S7    S6    S5    S4    S3    S2    S1    S0
   1     0     0     0     0     0     0     0     0
```

```
  Inputs:
  CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
   0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0
Outputs:
  COUT    S7    S6    S5    S4    S3    S2    S1    S0
   0     1     0     1     0     1     0     1     0
```

```
  Inputs:
  CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
   0     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1
Outputs:
  COUT    S7    S6    S5    S4    S3    S2    S1    S0
   1     0     1     0     1     0     1     0     0
```

```
  CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
   0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
Outputs:
  COUT    S7    S6    S5    S4    S3    S2    S1    S0
   0     0     0     0     0     0     0     0     0
```

```
Total calculations for all tests: 594
Total time taken to execute in seconds: 0.000933
```

```
Number of calculations: 99 for test: 1
Time taken to execute in seconds: 0.000170 for test: 1
Number of calculations: 99 for test: 2
Time taken to execute in seconds: 0.000153 for test: 2
Number of calculations: 99 for test: 3
Time taken to execute in seconds: 0.000151 for test: 3
Number of calculations: 99 for test: 4
Time taken to execute in seconds: 0.000152 for test: 4
Number of calculations: 99 for test: 5
Time taken to execute in seconds: 0.000174 for test: 5
Number of calculations: 99 for test: 6
Time taken to execute in seconds: 0.000133 for test: 6
```

Και για το παρακάτω netlist έχουμε τα ίδια ακριβώς αποτελέσματα

U76 NOT U75
U75 AND2 U71 U73
U74 XOR2 U72 U65
U73 NAND2 U72 U65
U72 XOR2 A07 B07
U71 NAND2 A07 B07
U65 OR2 U61 U63
U64 XOR2 U62 U55
U63 AND2 U62 U55
U62 XOR2 A06 B06
U61 AND2 A06 B06
U55 NAND2 U51 U53
U54 XOR2 U52 U45
U53 NAND2 U52 U45
U52 XOR2 A05 B05
U51 NAND2 A05 B05
U45 NAND2 U41 U43
U44 XOR2 U42 U35
U43 NAND2 U42 U35
U42 XOR2 A04 B04
U41 NAND2 A04 B04
U35 NAND2 U31 U33
U34 XOR2 U32 U25
U33 NAND2 U32 U25
U32 XOR2 A03 B03
U31 NAND2 A03 B03
U25 NAND2 U21 U23
U24 XOR2 U22 U15
U23 NAND2 U22 U15
U22 XOR2 A02 B02
U21 NAND2 A02 B02

U15 NAND2 U11 U13
U14 XOR2 U12 U05
U13 NAND2 U12 U05
U12 XOR2 A01 B01
U11 NAND2 A01 B01
U05 NAND2 U01 U03
U04 XOR2 U02 CIN
U03 NAND2 U02 CIN
U02 XOR2 A00 B00
U01 NAND2 A00 B00
S0 = U04
S1 = U14
S2 = U24
S3 = U34
S4 = U44
S5 = U54
S6 = U64
S7 = U74
COUT = U76

```
 Inputs:
 CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
  1     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0
Outputs:
 COUT    S7    S6    S5    S4    S3    S2    S1    S0
  1     0     0     0     0     0     0     0     0
```

```
Inputs:
CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
 0     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1
utputs:
COUT    S7    S6    S5    S4    S3    S2    S1    S0
 1     0     1     0     1     0     1     0     0
```

```
Total calculations for all tests: 510
Total time taken to execute in seconds: 0.000989
```

```
Number of calculations: 85 for test: 1
Time taken to execute in seconds: 0.000180 for test: 1
Number of calculations: 85 for test: 2
Time taken to execute in seconds: 0.000176 for test: 2
Number of calculations: 85 for test: 3
Time taken to execute in seconds: 0.000159 for test: 3
Number of calculations: 85 for test: 4
Time taken to execute in seconds: 0.000160 for test: 4
Number of calculations: 85 for test: 5
Time taken to execute in seconds: 0.000177 for test: 5
Number of calculations: 85 for test: 6
Time taken to execute in seconds: 0.000138 for test: 6
```

Οπότε κάλυψε και την περίπτωση που οι πύλες είναι ανακατεμένες ή είσαι περίεργη διάταξη.

Και τώρα θα παραθέσω ένα παράδειγμα όταν κάποιο component λύπη πώς μοιάζει το τελικό netlist. Έστω ότι έχουμε netlist:

U01 NAND2 A00 B00
U02 XOR2 A00 B00
U03 NAND2 U02 CIN
U04 XOR2 U02 CIN
U05 NAND2 U01 U03
U11 NAND2 A01 B01
U12 XOR2 A01 B01
U13 NAND2 U12 U05
U14 XOR2 U12 U05
U21 NAND2 A02 B02
U22 XOR2 A02 B02
U23 NAND2 U22 U15
U24 XOR2 U22 U15
U25 NAND2 U21 U23
U31 NAND2 A03 B03
U32 XOR2 A03 B03
U33 NAND2 U32 U25
U34 XOR2 U32 U25
U35 NAND2 U31 U33
U41 NAND2 A04 B04
U42 XOR2 A04 B04
U43 NAND2 U42 U35
U44 XOR2 U42 U35
U45 NAND2 U41 U43
U51 NAND2 A05 B05
U52 XOR2 A05 B05
U53 NAND2 U52 U45
U54 XOR2 U52 U45
U55 NAND2 U51 U53
U61 AND2 A06 B06
U62 XOR2 A06 B06
U63 AND2 U62 U55
U64 XOR2 U62 U55
U65 OR2 U61 U63
U71 NAND2 A07 B07
U72 XOR2 A07 B07
U73 NAND2 U72 U65
U74 XOR2 U72 U65
U75 AND2 U71 U73
U76 NOT U75
S0 = U04
S1 = U14
S2 = U24
S3 = U34
S4 = U44
S5 = U54

S6 = U64
S7 = U74
COUT = U76
Και λείπει το U15

```
  Inputs:
  CIN   A00   B00   A01   B01   A02   B02   A03   B03   A04   B04   A05   B05   A06   B06   A07   B07
   1     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0
Outputs:
  COUT    S7    S6    S5    S4    S3    S2    S1    S0
   -1    -1     -1    -1     -1    -1    -1     0     0
Result 2:
```

θα υπολογίσει κανονικά τα δύο πρώτα και θα βγάλει χεις όλα τα υπόλοιπα εφόσον λείπει ένα σήμα που είναι βασικό για τον υπολογισμό των υπολοιπων.
Και για ένα κύκλωμα που δεν είναι full under δηλαδή έναν Mux:
U0 NOT C
U1 AND2 A U0
U2 AND2 B C
U3 OR2 U1 U2
D0 = U3

IN
C 1, 0
A 1, 1
B 0, 1
OUT
D0

```
  Inputs:
  C   A   B
   1     1     0
Outputs:
  D0
   0
```

```
  Inputs:
  C   A   B
   0     1     1
Outputs:
  D0
   1
```

Όπως φαίνεται από τα αποτελέσματα που δίνονται όταν οι πύλες είναι ανακατεμένες λίγες παραπάνω πράξεις το οποίο είναι και λογικό ενώ ο χρόνος παραμένει σχεδόν ο ίδιος για όλα τα κυκλώματα που έχουν τον ίδιο αριθμό bits το οποίο δείχνει ότι είναι λειτουργικός σχέση χρονικής πολυπλοκότητα.

BONUS:Στο bonus κομμάτι δεν πρόλαβα να επεξεργαστώ όλο το κύκλωμα λόγω χρονικό περιορισμό μου αλλά έφτιαξα μία συνάρτηση η οποία κάνει μία τοπολογική βελτίωση των components μέσα στο Netlist με αναζήτηση κατά βάθος (Depth-First Search). Είναι ένας αλγόριθμος που χρησιμοποιείται για την εξερεύνηση ή την αναζήτηση γράφων.
Συνάρτηση dfs:

Αυτή η συνάρτηση εκτελεί μια αναζήτηση πρώτα σε βάθος (DFS) στο κύκλωμα, ξεκινώντας από τον δοθέντα κόμβο. Χρησιμοποιείται για την καταγραφή της σειράς των στοιχείων σε έναν πίνακα sorted. Οι παράμετροι της συνάρτησης είναι:current: Ο τρέχων κόμβος που εξετάζεται,circuit: Η δομή comp_array που περιέχει το κύκλωμα,visited[]: Ένας πίνακας που καταγράφει αν ένας κόμβος έχει επισκεφθεί ή όχι,sorted[]: Ο πίνακας που καταγράφει τη σειρά των στοιχείων,index: Ένας δείκτης που χρησιμοποιείται για την τρέχουσα θέση στον πίνακα sorted.Η συνάρτηση dfs υλοποιείται με αναδρομή και περιλαμβάνει τις εξής ενέργειες,Σημειώνει τον τρέχοντα κόμβο ως επισκεπτόμενο,Επαναλαμβάνει για κάθε σύνδεση του τρέχοντος κόμβου,Βρίσκει τον επόμενο κόμβο με βάση το αναγνωριστικό σύνδεσης.Εάν ο επόμενος κόμβος δεν έχει επισκεφθεί, τότε καλείται αναδρομικά η dfs για τον επόμενο κόμβο.Αποθηκεύει τον τρέχοντα κόμβο στον πίνακα sorted και μειώνει τον δείκτη index.
Συνάρτηση topological_sort:
Αυτή η συνάρτηση εκτελεί μια τοπολογική ταξινόμηση των στοιχείων του κυκλώματος χρησιμοποιώντας την συνάρτηση dfs. Η τοπολογική ταξινόμηση είναι ένας τρόπος να ταξινομήσουμε τα στοιχεία ενός γράφου έτσι ώστε κάθε ακμή να οδηγεί από ένα στοιχείο προς ένα άλλο στοιχείο.

Συνολικά, οι δύο συναρτήσεις συνεργάζονται για να εκτελέσουν την τοπολογική ταξινόμηση των στοιχείων του κυκλώματος. Η dfs εκτελεί την αναζήτηση πρώτα σε βάθος, καταγράφοντας τη σειρά των στοιχείων, ενώ η topological_sort χρησιμοποιεί την dfs για να εκτελέσει την τοπολογική ταξινόμηση και να επιστρέψει τα στοιχεία σε ταξινομημένη μορφή. Κάνοντας μία επαλήθευση για αυτή τη συνάρτηση βλέπουμε ότι όταν ανακατεύουμε τις πύλες στο Netlist τότε η συνάρτηση αυτή δίνει βελτιωμένες πράξεις και χρονο εφόσον βάζει τις πύλες με τη σειρα.
Χωρίς τον αλγόριθμο δίνει το παρακάτω αποτέλεσμα:

```
Total calculations for all tests: 540
Total time taken to execute in seconds: 0.000925
```

Ενώ με τον αλγόριθμο δίνει αυτό το αποτέλεσμα:

```
Total calculations for all tests: 486
Total time taken to execute in seconds: 0.000864
```