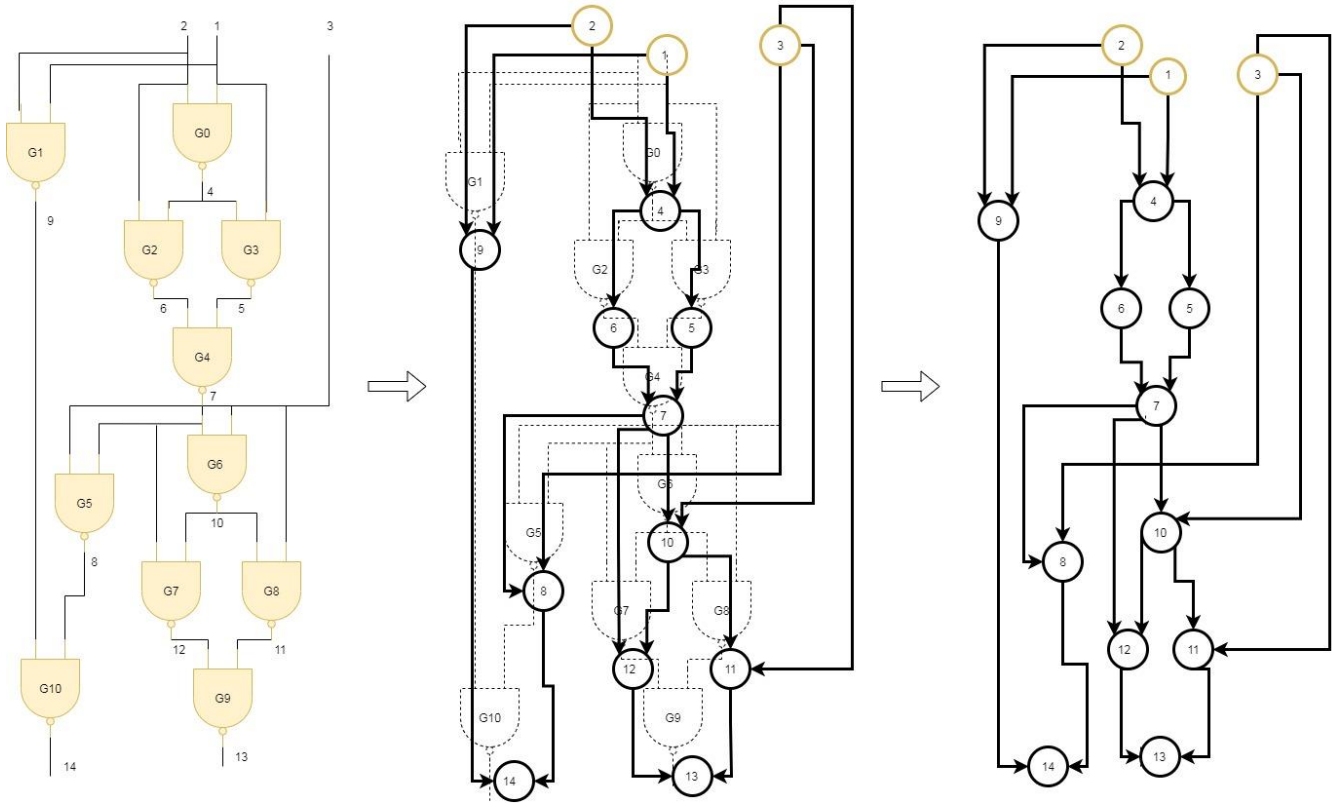
ΑΝΑΦΟΡΑ 6ου set

Κουρκουλος Αγγελος ΑΜ:2017030111

Σε αυτή την άσκηση σκοπός είναι η δημιουργία ενός αλγόριθμου που θα κάνει μια απλή μορφή formal verification . Ποιο συγκεκριμένα το πρόγραμμα δέχεται ως είσοδο ένα πρότυπο αρχείο που απεικονίζει ένα αθροιστή (FULL ADDER) και ένα δεύτερο αρχείο που επίσης απεικονίζει ένα αθροιστή υλοποιημένο με τις ίδιες πύλες αλλά διατεταγμένο με διαφορετική σειρά και με διαφορετική ονοματολογία των πυλών και των ακμών που τις ενώνουν . Έχοντας αναγνώσει, αποθηκεύσει την πληροφορία από αυτά τα 2 αρχεία στόχος του προγράμματος είναι η απόφαση για το κατά πόσο τα αρχεία απεικονίζουν το ίδιο η διαφορετικό λογικό κύκλωμα . Αν τα κυκλώματα είναι τα ίδια το πρόγραμμα παράγει ένα αρχείο εξόδου που επαληθεύει την ταύτιση τους και δείχνει την αντιστοίχηση μεταξύ πυλών διαφορετικά διατυπώνει την απόφαση της διαφοράς των κυκλωμάτων και δείχνει την μεγαλύτερη ταύτιση πυλών που μπορεί να βρει .

Για τη δημιουργία του προγράμματος ακολουθήθηκαν τα εξής βήματα :

1. Αρχικά χρησιμοποιήθηκε η συνάρτηση initialization() η οποία είχε υλοποιηθεί από προηγούμενες ασκήσεις η οποία διαβάζει το αρχείο εισόδου και αποθηκεύει την απαραίτητη πληροφορία σε στατικούς πίνακες ικανοποιητικού μεγέθους . Αυτή η συνάρτηση καλείται 2 φορές ώστε να διαβαστούν και το πρότυπο αρχείο και το αρχείο για την εξέταση της ταύτισης του με το πρότυπο .
2. Στη συνέχεια για καθένα από τα 2 αρχεία καλείται η συνάρτηση createDAG() που είναι μια παραλλαγή της αντίστοιχης συνάρτησης που είχε υλοποιηθεί σε προηγούμενες ασκήσεις η οποία δημιουργεί ένα κατευθυνόμενο γράφο με βάση το netlist του κυκλώματος που παίρνει ως είσοδο. Η δημιουργία του γράφου επιτυγχάνεται φτιάχνοντας ένα κόμβο για κάθε πύλη του κυκλώματος με αριθμό ίσο με τον αριθμό εξόδου της πύλης καθώς και ένα κόμβο για κάθε είσοδο του κυκλώματος με αριθμό ίδιο με αυτόν της εισόδου και ενώνοντας τους με κατευθυνόμενες ακμές .Ακολουθεί το παράδειγμα της απεικόνισης του FULL ADDER ως γράφο :



3. Μετά τη δημιουργία του κάθε γράφου κατασκευάζεται ένα array οπου τοποθετούνται ταξινομημένα όλοι οι κόμβοι του γράφου σύμφωνα με τη πύλη που αντιπροσωπεύουν το οποίο θα χρησιμοποιηθεί στη συνέχεια. Έπειτα καλείται η συνάρτηση που υλοποιήθηκε homomorphicGraphs() οι οποία παίρνει ως είσοδο τους 2 γράφους και τσεκάρει αν είναι ομομορφικοί-ισομορφικοί . Αυτό επιτυγχάνεται αρχικά κάνοντας τους χοντρικούς ελέγχους δηλαδή αν οι 2 γράφοι έχουν τον ίδιο αριθμό κόμβων που αντιπροσωπεύουν συγκεκριμένες πύλες και τον ίδιο αριθμό ακμών και ύστερα κάνοντας εξονυχιστικό έλεγχο σε κάθε κόμβο . Αυτός ο πλήρης έλεγχος γίνεται κοιτώντας τον κάθε κόμβο του εξεταζόμενου γράφου και αφού βρεθεί κάποιος όμοιος κόμβος του πρότυπου γράφου ,δηλαδή να έχει ίδιο βαθμό ακμών και ίδιο τύπο πύλης , ελέγχεται αν ταιριάζουν και οι γειτονικοί τους κόμβοι και οι γειτονικοί των γειτονικών κλπ. Αυτή η διαδικασία γίνεται με αναδρομικό τρόπο και σταματάει όταν επιβεβαιωθεί ότι οι υπογράφοι των 2 κόμβων που εξετάζουμε είναι πανομοιότυποι . Αν οι διαδικασία αποτύχει τότε γίνεται προσπάθεια για αντιστοίχηση του εξεταζόμενου κόμβου με άλλο κόμβο του πρότυπου γράφου . Σε αυτό το σημείο χρησιμοποιούνται και οι ταξινομημένοι πίνακας με κόμβους που βοηθάνε στην γρηγορότερη εύρεση κάποιου πιθανού ζευγαριού . Σε περίπτωση που η συνολική διαδικασία αποτύχει δηλαδή δεν βρεθεί κάποιος κόμβος του πρότυπου γράφου που να ταιριάζει με κάποιο κόμβο του εξεταζόμενου γράφου τότε ξέρουμε ότι οι 2 γράφοι δεν είναι ομομορφικοί-ισομοεφικοί και επιστρέφεται ο μεγαλύτερος υπογράφος που βρέθηκε.

4. Αφού βρεθεί η αντιστοιχία των κόμβων των 2 γράφων το επόμενο πράγμα που κάνει το πρόγραμμα είναι να αποθηκεύει τις πύλες που αντιπροσωπεύουν οι αντίστοιχοι κόμβοι των 2 γράφων σε αντίστοιχες θέση μέσα σε 2 πίνακες .
5. Στη συνέχεια καλείται η συνάρτηση sortMatchingGates() η οποία αναδιατάσσει τα κελιά των 2 ποινικών ώστε ο πίνακα που περιέχει τις πύλες του πρότυπου κυκλώματος να είναι αποθηκευμένες με τη σειρά κρατώντας όμως την αντιστοίχηση των πυλών σωστή αφού οι 2 πίνακες κάνουν τις ίδιες μεταθέσεις κελιών . Το βήμα αυτό δεν είναι αναγκαίο βοηθάει όμως στην ωραιότερη εκτύπωση του αρχείου εξόδου έτσι ώστε να είναι ευκολότερη η επαλήθευση της ορθότητας των αποτελεσμάτων
6. Τέλος καλείτε η συνάρτηση sortMatchingGates() που υλοποιήθηκε η οποία παίρνει ως είσοδο τους 2 πίνακες που εμπεριέχουν τις αντιστοιχισμένες πύλες, τα netlist των 2 κυκλωμάτων και την απόφαση για το αν οι γράφοι είναι ομομορφικοί-ισομοεφικοί δηλαδή αν τα 2 κυκλώματα είναι ίδια και δημιουργεί το αρχείο εξόδου το οποίο δείχνει τις αντιστοιχισμένες πύλες .

Η δομή του αλγορίθμου που υλοποιήθηκε είναι :

```
define the arrays we need to store the information from prototype file

initialization(…);                    //it stores the information from file in the appropriate array

define the arrays we need to store the information from Checking Files

initialization (…);                    //it stores the information from Library in the appropriate array

prototypeG r = createDAG(...) ;   //create the graph for the prototype circuit

checkingGr=createDAG(…) ;       //create the graph for the circuit we want to know if is the same with prototype

define the arrays we need to store the information for matching nodes

int ishomomorphic = homomorphicGraphs(prototypeGr,checkingGr,…);

int matchedGateProt[100]={};
int matchedGateCheck[100]={};
int i=0;
for(each PrototypeNode){    // loop that finds the matching gates from 2 files
            for(each checkingNode){
                  if(PrototypeNode is matched with checkingNode)  {   //finds matching gates from matching nodes
                                    matchedGateProt[i]= PrototypeNode->gate;
                                    matchedGateCheck [i]= checkingNode ->gate;
                                    i++;
                   }
            }
}

   //sorting the prototype matched gates and doing the corisponding swaps to the check matched gates
  sortMatchingGates(matchedGateProt,matchedGateCheck); // we doing it to make a better visualy output

createFileOfMatchingGates(matchedGateProt,matchedGateCheck,netlist,logicgate,netlistCheck,ishomomorphic);
```

Η δομή της συνάρτησης HomomorphicGraphs() που υλοποιήθηκε είναι :

```
if(numOfGatesProt!=numOfGatesCheck )//if the graphs have different number of gates they arent homomorphic
        return -1;
else if(numOfEdges!=numOfEdgesCheck)//if the graphs have different number of edges they arent homomorphic
        return -2;
else{
         int i=0;
         while(i!=numOfNodesCheck){       //loop for every node of cheking graph
               int j=0;
               int match=0;
               // NodeCheck and NodeProt are sorted by the type of gate.
              // while there is a possible match for the node NodeCheck [i]
              while(NodeCheck[i]-> Gate >= NodeProt[j]-> Gate && match=0){
                    // if nodes are possible matching
                    if(NodeCheck[i]-> Gate >= NodeProt[j]-> Gate&&NodeCheck[i]->degree>= NodeProt[j]->degree){
                          match=recursion(NodeCheck[i], NodeProt[j]); //see if neighbors are matching
                    }
                    j++;
               }
              // if there is a node which cant match with anyone then the 2 graphs are not homomorphic
              if(match ==0 ){return -3;}
              i++;
        }
}
```

Αρχικά η γίνεται η απλούστερη δοκιμή που τα 2 αρχεία εισόδου είναι τα ίδια και απεικονίζουν ένα FULL ADDER φτιαγμένο από πύλες NAND:

Πρότυπο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13,14
## NETLIST
G1,NAND_2 , IN,1,2,OUT,4
G2,NAND_2, IN,1,2,OUT,9
G3, NAND_2,  IN,2,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,5,6, OUT,7
G6, NAND_2,  IN,3,7, OUT,8
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND_2,  IN,12,11, OUT,13
G11, NAND_2,  IN,9,8, OUT,14
```

Εξεταζόμενο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13,14
## NETLIST
G1,NAND_2 , IN,1,2,OUT,4
G2,NAND_2, IN,1,2,OUT,9
G3, NAND_2,  IN,2,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,5,6, OUT,7
G6, NAND_2,  IN,3,7, OUT,8
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND_2,  IN,12,11, OUT,13
G11, NAND_2,  IN,9,8, OUT,14
```

Τα αποτελέσματα που παίρνουμε βάζοντας ως είσοδο τα παραπάνω αρχεία είναι :

```
THE 2 CIRUITS ARE THE SAME!!!

 ********THIS IS THE MATCH BETWEEN THE 2 CIRCUITS********

PROTOTYPE CIRCUIT          CHECKING CIRCUIT
G1 IN 1 ,2 OUT 4    ->    G1 IN 1 ,2 OUT 4
G2 IN 1 ,2 OUT 9    ->    G2 IN 1 ,2 OUT 9
G3 IN 2 ,4 OUT 6    ->    G3 IN 2 ,4 OUT 6
G4 IN 1 ,4 OUT 5    ->    G4 IN 1 ,4 OUT 5
G5 IN 5 ,6 OUT 7    ->    G5 IN 5 ,6 OUT 7
G6 IN 3 ,7 OUT 8    ->    G6 IN 3 ,7 OUT 8
G7 IN 7 ,3 OUT 10    ->    G7 IN 7 ,3 OUT 10
G8 IN 7 ,10 OUT 12    ->    G8 IN 7 ,10 OUT 12
G9 IN 3 ,10 OUT 11    ->    G9 IN 3 ,10 OUT 11
G10 IN 12 ,11 OUT 13    ->    G10 IN 12 ,11 OUT 13
G11 IN 9 ,8 OUT 14    ->    G11 IN 9 ,8 OUT 14
```

Η επόμενη δοκιμή είναι τα ίδια αρχεία εισόδου αλλά αφαιρώντας την τελευταία πύλη του εξεταζόμενου αρχείου και έχοντας αυτόματα μικρότερο αριθμό πυλών-κόμβων περιμένουμε να ενεργοποιηθεί ο απλός έλεγχος για διαφορετικό αριθμό κόμβων :

Πρότυπο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13,14
## NETLIST
G1,NAND_2 , IN,1,2,OUT,4
G2,NAND_2, IN,1,2,OUT,9
G3, NAND_2,  IN,2,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,5,6, OUT,7
G6, NAND_2,  IN,3,7, OUT,8
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND_2,  IN,12,11, OUT,13
G11, NAND_2,  IN,9,8, OUT,14
```

Εξεταζόμενο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13
## NETLIST
G1,NAND_2 , IN,1,2,OUT,4
G2,NAND_2, IN,1,2,OUT,9
G3, NAND_2,  IN,2,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,5,6, OUT,7
G6, NAND_2,  IN,3,7, OUT,8
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND 2,  IN,12,11, OUT,13
```

Τα αποτελέσματα που παίρνουμε βάζοντας ως είσοδο τα παραπάνω αρχεία είναι :

```
THE 2 CIRUITS ARE NOT THE SAME

 ********the number of GATES is different between the two circuits********
```

Αφού έχουμε ελέγξει της 2 απλούστερες περιπτώσεις εξετάζουμε την περίπτωση που τα 2 αρχεία είναι πανομοιότυπα αλλά έχουν διαφορετική σειρά αρίθμησης πυλών . Για αυτό το παράδειγμα εναλλάσσουμε την 11 πύλη του εξεταζόμενου αρχείου με την 5 πύλη κρατώντας ίδια την ονοματολογία των εισερχόμενων και εξερχόμενων ακμών :

Πρότυπο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13,14
## NETLIST
G1,NAND_2 , IN,1,2,OUT,4
G2,NAND_2, IN,1,2,OUT,9
G3, NAND_2,  IN,2,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,5,6, OUT,7
G6, NAND_2,  IN,3,7, OUT,8
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND_2,  IN,12,11, OUT,13
G11, NAND_2,  IN,9,8, OUT,14
```

Εξεταζόμενο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13,14
## NETLIST
G1,NAND_2 , IN,1,2,OUT,4
G2,NAND_2, IN,1,2,OUT,9
G3, NAND_2,  IN,2,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,9,8, OUT,14
G6, NAND_2,  IN,3,7, OUT,8
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND_2,  IN,12,11, OUT,13
G11, NAND_2,  IN,5,6, OUT,7
```

Τα αποτελέσματα που παίρνουμε βάζοντας ως είσοδο τα παραπάνω αρχεία είναι :

```
THE 2 CIRUITS ARE THE SAME!!!

********THIS IS THE MATCH BETWEEN THE 2 CIRCUITS********

PROTOTYPE CIRCUIT          CHECKING CIRCUIT
G1 IN 1 ,2 OUT 4     ->    G1 IN 1 ,2 OUT 4
G2 IN 1 ,2 OUT 9     ->    G2 IN 1 ,2 OUT 9
G3 IN 2 ,4 OUT 6     ->    G3 IN 2 ,4 OUT 6
G4 IN 1 ,4 OUT 5     ->    G4 IN 1 ,4 OUT 5
G5 IN 5 ,6 OUT 7     ->    G11 IN 5 ,6 OUT 7
G6 IN 3 ,7 OUT 8     ->    G6 IN 3 ,7 OUT 8
G7 IN 7 ,3 OUT 10     ->    G7 IN 7 ,3 OUT 10
G8 IN 7 ,10 OUT 12     ->    G8 IN 7 ,10 OUT 12
G9 IN 3 ,10 OUT 11     ->    G9 IN 3 ,10 OUT 11
G10 IN 12 ,11 OUT 13     ->    G10 IN 12 ,11 OUT 13
G11 IN 9 ,8 OUT 14     ->    G5 IN 9 ,8 OUT 14
```

Για την επόμενη δοκιμή, κρατώντας ίδια την εναλλαγή που έχει γίνει ,εξετάζεται η περίπτωση που τα αρχεία είναι ίδια αλλά έχουν διαφορετική αρίθμηση ακμών .Ενδεικτικά ανταλλάζονται οι ονοματολογίες των κόμβων 8 και 2:

Πρότυπο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13,14
## NETLIST
G1,NAND_2 , IN,1,2,OUT,4
G2,NAND_2, IN,1,2,OUT,9
G3, NAND_2,  IN,2,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,5,6, OUT,7
G6, NAND_2,  IN,3,7, OUT,8
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND_2,  IN,12,11, OUT,13
G11, NAND_2,  IN,9,8, OUT,14
```

Εξεταζόμενο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13,14
## NETLIST
G1,NAND_2 , IN,1,8,OUT,4
G2,NAND_2, IN,1,8,OUT,9
G3, NAND_2,  IN,8,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,9,2, OUT,14
G6, NAND_2,  IN,3,7, OUT,2
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND_2,  IN,12,11, OUT,13
G11, NAND_2,  IN,5,6, OUT,7
```

Τα αποτελέσματα που παίρνουμε βάζοντας ως είσοδο τα παραπάνω αρχεία είναι :

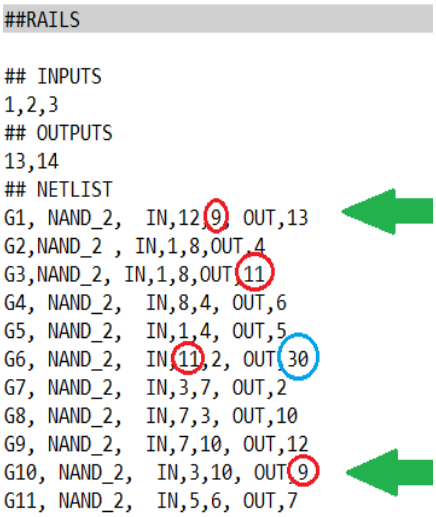
```
THE 2 CIRUITS ARE THE SAME!!!

 ********THIS IS THE MATCH BETWEEN THE 2 CIRCUITS********

PROTOTYPE CIRCUIT          CHECKING CIRCUIT
G1 IN 1 ,2 OUT 4    ->    G1 IN 1 ,8 OUT 4
G2 IN 1 ,2 OUT 9    ->    G2 IN 1 ,8 OUT 9
G3 IN 2 ,4 OUT 6    ->    G3 IN 8 ,4 OUT 6
G4 IN 1 ,4 OUT 5    ->    G4 IN 1 ,4 OUT 5
G5 IN 5 ,6 OUT 7    ->    G11 IN 5 ,6 OUT 7
G6 IN 3 ,7 OUT 8    ->    G6 IN 3 ,7 OUT 2
G7 IN 7 ,3 OUT 10    ->    G7 IN 7 ,3 OUT 10
G8 IN 7 ,10 OUT 12    ->    G8 IN 7 ,10 OUT 12
G9 IN 3 ,10 OUT 11    ->    G9 IN 3 ,10 OUT 11
G10 IN 12 ,11 OUT 13    ->    G10 IN 12 ,11 OUT 13
G11 IN 9 ,8 OUT 14    ->    G5 IN 9 ,2 OUT 14
```

Στην επόμενη δοκιμή για να επιβεβαιώσουμε ότι ο αλγόριθμος λειτουργεί πραγματικά σωστά κρατάμε τις 2 αλλαγές που έχουμε ήδη κάνει και κάνουμε ακόμα 3 ίδιου τύπου ώστε πλέον να έχουμε ένα αρκετά μπερδεμένο αρχείο προς εξέταση . Η αλλαγές που γίνονται είναι να ανταλλάξουμε τους αριθμούς των κόμβων 9 με 11 , να αλλάξουμε την τιμή του κόμβου 14 σε 30 και βάζουμε την πύλη νούμερο 10 στην πρώτη θέση αυξάνοντας τον αριθμό σειράς των πυλών 1-9 σε 2-10 :

Πρότυπο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13,14
## NETLIST
G1,NAND_2 , IN,1,2,OUT,4
G2,NAND_2, IN,1,2,OUT,9
G3, NAND_2,  IN,2,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,5,6, OUT,7
G6, NAND_2,  IN,3,7, OUT,8
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND_2,  IN,12,11, OUT,13
G11, NAND_2,  IN,9,8, OUT,14
```

Εξεταζόμενο αρχείο :



Τα αποτελέσματα που παίρνουμε βάζοντας ως είσοδο τα παραπάνω αρχεία είναι :

```
THE 2 CIRUITS ARE THE SAME!!!

 ********THIS IS THE MATCH BETWEEN THE 2 CIRCUITS********

PROTOTYPE CIRCUIT          CHECKING CIRCUIT
G1 IN 1 ,2 OUT 4    ->    G2 IN 1 ,8 OUT 4
G2 IN 1 ,2 OUT 9    ->    G3 IN 1 ,8 OUT 11
G3 IN 2 ,4 OUT 6    ->    G4 IN 8 ,4 OUT 6
G4 IN 1 ,4 OUT 5    ->    G5 IN 1 ,4 OUT 5
G5 IN 5 ,6 OUT 7    ->    G11 IN 5 ,6 OUT 7
G6 IN 3 ,7 OUT 8    ->    G7 IN 3 ,7 OUT 2
G7 IN 7 ,3 OUT 10    ->    G8 IN 7 ,3 OUT 10
G8 IN 7 ,10 OUT 12    ->    G9 IN 7 ,10 OUT 12
G9 IN 3 ,10 OUT 11    ->    G10 IN 3 ,10 OUT 9
G10 IN 12 ,11 OUT 13    ->    G1 IN 12 ,9 OUT 13
G11 IN 9 ,8 OUT 14    ->    G6 IN 11 ,2 OUT 30
```

Μετά από τις παραπάνω περιπτώσεις μπορούμε να πούμε με αρκετή σιγουριά ότι το πρόγραμμα βρίσκει τα κυκλώματα που είναι ίδια και τις σωστές αντιστοιχίσεις τους . Τέλοτς λοιπόν απομένει να εξετάσουμε την περίπτωση που το αρχείο δεν είναι ίδιο αλλα ούτε έχει διαφορετικό αριθμό κόμβων ή ακμών .

Αφερούμε τις αλλαγές που έγιναν στα προηγούμενα βήματα και η μόνη αλλαγή που γίνεται είναι να μπεί ως είσοδο στην $6^{η}$ πύλη η ακμή νουμερο 5 δηλαδή η έξοδος της πύλης 4 αντί για την ακμή 3 που είναι είσοδος του συστήματος.

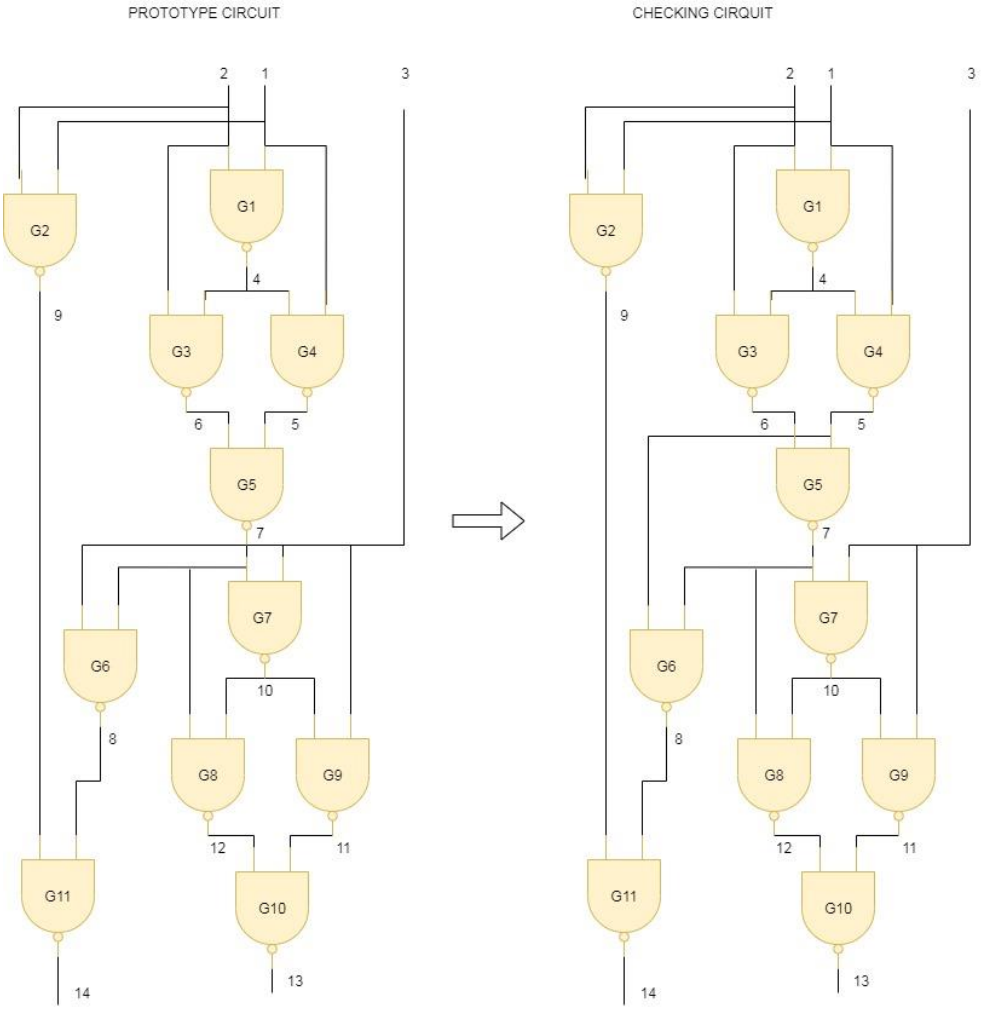
Πρότυπο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13,14
## NETLIST
G1,NAND_2 , IN,1,2,OUT,4
G2,NAND_2, IN,1,2,OUT,9
G3, NAND_2,  IN,2,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,5,6, OUT,7
G6, NAND_2,  IN,3,7, OUT,8
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND_2,  IN,12,11, OUT,13
G11, NAND_2,  IN,9,8, OUT,14
```

Εξεταζόμενο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13,14
## NETLIST
G1,NAND_2 , IN,1,2,OUT,4
G2,NAND_2, IN,1,2,OUT,9
G3, NAND_2,  IN,2,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,5,6, OUT,7
G6, NAND_2,  IN,3,7, OUT,8
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND_2,  IN,12,11, OUT,13
G11, NAND_2,  IN,9,8, OUT,14
```

Η απεικόνιση του κυκλώματος των παραπάνω αρχείων είναι :



Τα αποτελέσματα που παίρνουμε βάζοντας ως είσοδο τα παραπάνω αρχεία είναι :

```
THE 2 CIRUITS ARE NOT THE SAME

 ********THIS IS THE BIGGEST MATCH BETWEEN THE 2 CIRCUITS THAT COULD BE FOUND********

PROTOTYPE CIRCUIT          CHECKING CIRCUIT
G3 IN 2 ,4 OUT 6    ->    G3 IN 2 ,4 OUT 6
G5 IN 5 ,6 OUT 7    ->    G5 IN 5 ,6 OUT 7
G6 IN 3 ,7 OUT 8    ->    G6 IN 5 ,7 OUT 8
G7 IN 7 ,3 OUT 10    ->    G7 IN 7 ,3 OUT 10
G8 IN 7 ,10 OUT 12    ->    G8 IN 7 ,10 OUT 12
G9 IN 3 ,10 OUT 11    ->    G9 IN 3 ,10 OUT 11
G10 IN 12 ,11 OUT 13    ->    G10 IN 12 ,11 OUT 13
G11 IN 9 ,8 OUT 14    ->    G11 IN 9 ,8 OUT 14
```

Βλέπουμε ότι το πρόγραμμα δεν ταίριαξε την πύλη G4 καθώς στην έξοδο της βρίσκετε το λάθος του κυκλώματος και προσπάθησε να αντιστοιχίσει όσο το δυνατόν περισσότερες πύλες δίνοντας μας αυτό το αποτέλεσμα .

Γίνεται ακόμα μια δοκιμή διαφορετικών κυκλωμάτων αλλάζοντας την ακμή 9 που εισέρχεται στην πύλη 11 με την ακμή 11 :

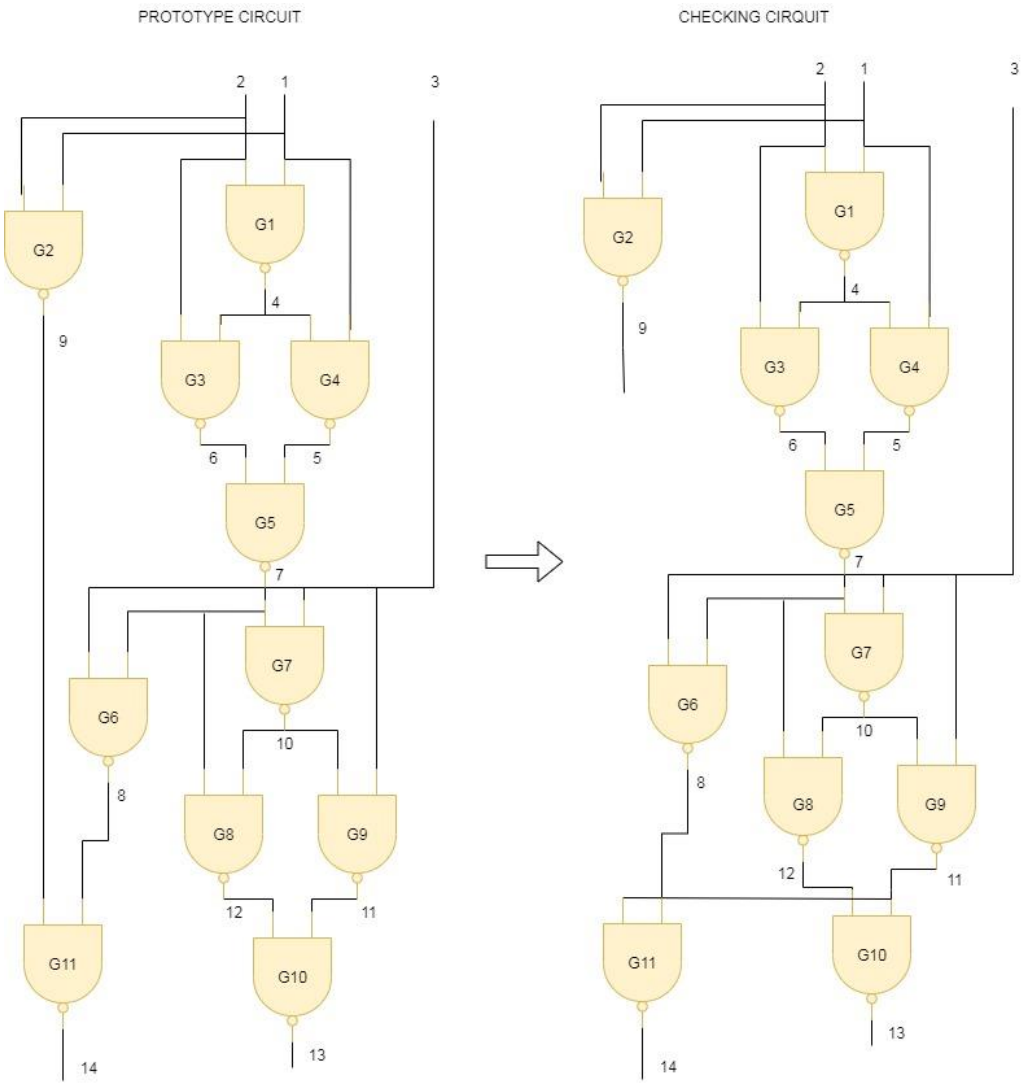
Πρότυπο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13,14
## NETLIST
G1,NAND_2 , IN,1,2,OUT,4
G2,NAND_2, IN,1,2,OUT,9
G3, NAND_2,  IN,2,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,5,6, OUT,7
G6, NAND_2,  IN,3,7, OUT,8
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND_2,  IN,12,11, OUT,13
G11, NAND_2,  IN,9,8, OUT,14
```

Εξεταζόμενο αρχείο :

```
##RAILS

## INPUTS
1,2,3
## OUTPUTS
13,14
## NETLIST
G1,NAND_2 , IN,1,2,OUT,4
G2,NAND_2, IN,1,2,OUT,9
G3, NAND_2,  IN,2,4, OUT,6
G4, NAND_2,  IN,1,4, OUT,5
G5, NAND_2,  IN,5,6, OUT,7
G6, NAND_2,  IN,3,7, OUT,8
G7, NAND_2,  IN,7,3, OUT,10
G8, NAND_2,  IN,7,10, OUT,12
G9, NAND_2,  IN,3,10, OUT,11
G10, NAND_2,  IN,12,11, OUT,13
G11, NAND_2,  IN,11,8, OUT,14
```

Η απεικόνιση του κυκλώματος των παραπάνω αρχείων είναι :



Τα αποτελέσματα που παίρνουμε βάζοντας ως είσοδο τα παραπάνω αρχεία είναι :

```
THE 2 CIRUITS ARE NOT THE SAME

 ********THIS IS THE BIGGEST MATCH BETWEEN THE 2 CIRCUITS THAT COULD BE FOUND********

PROTOTYPE CIRCUIT          CHECKING CIRCUIT
G6 IN 3 ,7 OUT 8    ->    G6 IN 3 ,7 OUT 8
G8 IN 7 ,10 OUT 12    ->    G8 IN 7 ,10 OUT 12
G10 IN 12 ,11 OUT 13    ->    G10 IN 12 ,11 OUT 13
G11 IN 9 ,8 OUT 14    ->    G11 IN 11 ,8 OUT 14
```

Φαίνεται και εδώ ότι η πύλη 9 που η έξοδος της πηγαίνει σε λάθος σημεία δεν εμπεριέχεται στις ταιριαζμένες πύλες του αρχείου εξόδου όπως και οι πύλες συνδέονται που με αυτή ή είναι πιο πίσο από αυτή . Αντίθετα οι πύλες 6,8,10,11 κατάφεραν να αντιστοιχιθούν .

Μετά τα παραπάνω αποτελέσματα μπορούμε να πούμε ότι ο αλγόριθμος λειτουργεί σχετικά καλά για κάθε είδους περίπτωση .

Ενδεικτικό flowchart :