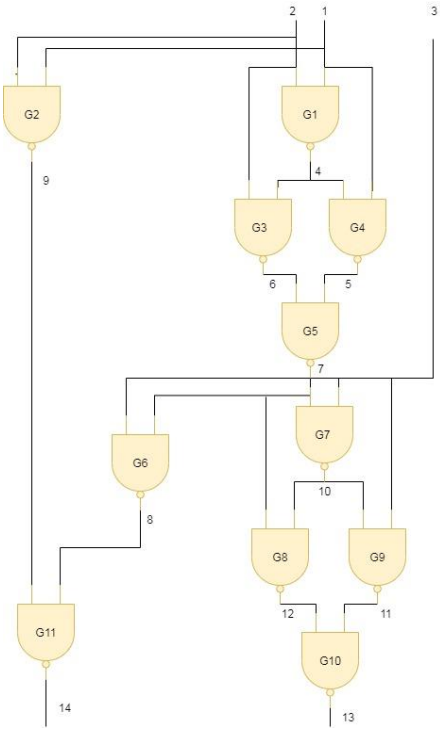
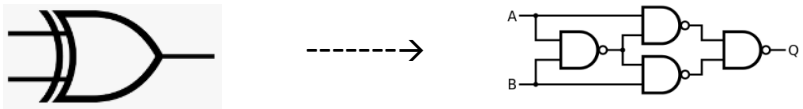
ΑΝΑΦΟΡΑ 5ου set

Κουρκουλος Αγγελος ΑΜ:2017030111

Στην προηγούμενη εργαστηριακή άσκηση παίρνοντας ως είσοδο το αρχείο ενός RCA επιτεύχθηκε, μετά την εύρεση του κατάλληλου σημείου διαχωρισμού, η διάσπαση του σε 2 ξεχωριστά αρχεία που περιείχαν την πληροφορία του μισού κυκλώματος το κάθε ένα . Σε αυτή την άσκηση λιπών σκοπός ήταν η χρήση ενός από τα αρχεία εξόδου της άσκησης 4 για τη δημιουργία ενός αρχείου διασυνδεδεμένων πυλών όπως το αρχείο εισόδου της άσκησης 3 . Επειδή στην προηγούμενη άσκηση χρησιμοποιήθηκε η πύλη Xor_2, που ήταν υλοποιημένη στη βιβλιοθήκη, για την δημιουργία του RCA αλλά σε αυτή την άσκηση η εκφώνηση ζητάει ρητά την χρήση μόνο πυλών NAND_2 , NOR_2 και NOT δημιουργήθηκε ένας καινούριος RCA 4 bit ,ο οποίος χρησιμοποιήθηκε για τη διάσπαση και έπειτα για τη δημιουργία του αρχείου των πυλών , όπου η υλοποίηση του κάθε Full Adder του έγινε μόνο με πύλες NAND_2 και η απεικόνιση τους φαίνεται παρακάτω :



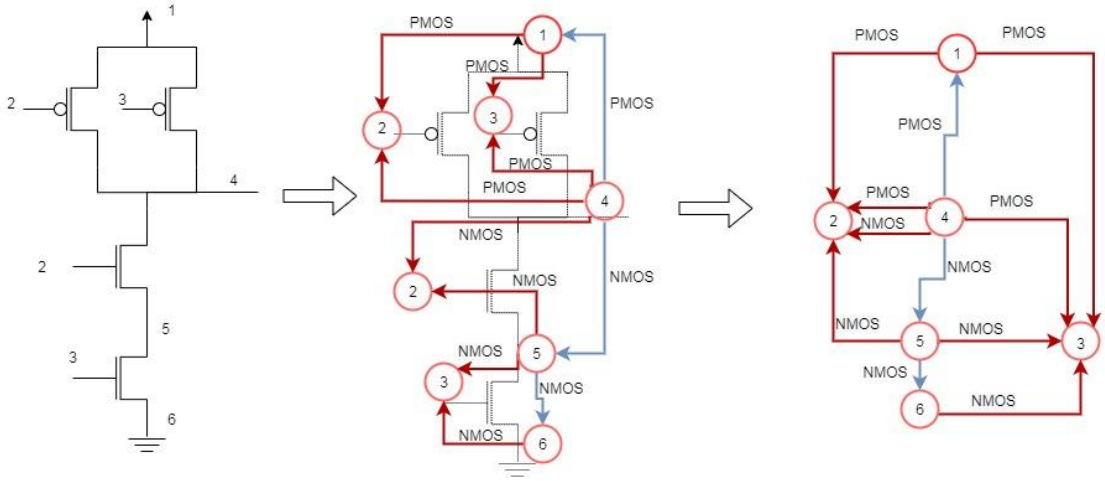
Μετατροπή Xor σε NANDs:



Για την περάτωση της παρούσας άσκησης χρησιμοποιήθηκε αυτούσιος ο κώδικας της άσκησης 4 ώστε να γίνουν σειριακά όλες οι απαραίτητες ενέργειες για τη διάσπαση του 4bitou RCA σε 2 2bitους και στη συνέχεια ,στο σημείο της συνάρτησης main που έχει επισημανθεί με σχόλια ,προστέθηκε ο κώδικας ο οποίος παίρνει ένα από τα παρεισαχθέντα αρχεία δημιουργεί το αρχείο διασυνδεδεμένων πυλών μετά από μια ακολουθία βημάτων και τέλος το προσομοιώνει φτιάχνοντας χειροκίνητα κάποια test_vector .

Τα βήματα του νέου κώδικα που δημιουργήθηκε για την επίλυση του προβλήματος μας είναι τα εξής :

1. Αρχικά καλείται συνάρτηση initialization που έχει υλοποιηθεί από προηγούμενες ασκήσεις δίνοντας ως είσοδο ένα από τα 2 αρχεία που παρήχθησαν από τη διάσπαση του 4bitou RCA ώστε να γίνει αποθήκευση της πληροφορίας που περιγράφει το αρχείο σε πίνακες και μεταβλητές που είναι απαραίτητοι για την αναπαράσταση του netlist , των Inputs ,των Outputs κτλ
2. Ακόμα καλείτε η συνάρτηση readLib() η οποία διαβάζει τη βιβλιοθήκη που έχει υλοποιηθεί από προηγούμενες ασκήσεις και αποθηκεύει σε πίνακες την απαραίτητη πληροφορία για την περιγραφή της δομής των γράφων που αντιπροσωπεύει κάθε πύλη.
3. Στη συνέχεια καλείτε η συνάρτηση createDAG() η οποία παίρνει το netlist και δημιουργεί ένα κατευθυνόμενο γράφο συνδέοντας τα source και drain των transistor με μια κατευθυνόμενη κόκκινη ακμή στα αντίστοιχα gates , καθώς συνδέει και τα drain με τα source με μια κατευθυνόμενη μπλε ακμή με κατεύθυνση εξαρτώμενη από το αν το transistor που βρίσκονται είναι PMOS η NMOS. Σε κάθε ακμή δίνεται βάρος ίσο με 1=NMOS αν το transistor που βρίσκονται είναι NMOS η 2=PMOS αν το transistor που βρίσκονται είναι PMOS. Πρακτικά οι μπλε ακμές είναι αυτές που συνδέουν τους ενδιάμεσους κόμβους μιας πύλης ενώ οι κόκκινες συνδέουν τις διαφορετικές πύλες μεταξύ τους . Ακολουθεί παράδειγμα για το πώς θα γινόταν ο γράφος μίας πύλης NAND



4. Έπειτα καλείται η συνάρτηση findPossibleBreak() ,που υλοποιήθηκε στην προηγούμενη άσκηση, η οποία μας βρίσκει όλα τα πιθανά σημεία διάσπασης δηλαδή τους κόμβους που είναι συνδεδεμένοι με τουλάχιστον ένα source ,ένα

drain και ένα gate . Στα σημεία αυτά βρίσκονται οι κόμβοι που συνδέουν την έξοδο της μια πύλης με την είσοδο μιας άλλης. Αν θεωρήσουμε λιπών ότι κάθε τέτοιο σημείο αντιπροσωπεύει την πύλη που έχει αυτό το σημείο ως έξοδο , έχουν βρεθεί όλε οι πύλες εκτός από αυτές που η έξοδος τους δεν πηγαίνει στην είσοδο καμίας άλλης δηλαδή τα outputs του κυκλώματος . Για να συμπεριλάβουμε αυτές τις πύλες προσθέτουμε στο array που έχει τα possibleBreaks τα outputs του κυκλώματος .

5. Το επόμενο βήμα είναι να βρεθεί τι πύλη αντιπροσωπεύει κάθε στοιχείο από το array με τα possibleBreaks. Το πρόβλημα αυτό λύνεται κοιτάζοντας τον αντίστοιχο κόμβο του γράφου για κάθε στοιχείο του array possibleBreaks και βλέποντας την τιμή που έχει κάθε ακμή αυτού του κόμβου . Αν οι 2 ακμές του κόμβου έχουν την τιμή PMOS και μία ακμή την τιμή NMOS τότε η πύλη είναι μια NAND , αν υπάρχουν 2 ακμές με τιμή NMOS και μια με τιμή PMOS τότε η πύλη είναι μια NOR ενώ αν υπάρχει μόνο μια τιμή NMOS και μια PMOS τότε είναι πύλη NOT. Μετά τον παραπάνω έλεγχο επαληθεύεται η ορθότητα της υλοποίησης της κάθε πύλης κοιτώντας αν οι εν σειρά μπλε ακμές με τιμή NMOS είναι ίσες με το πλήθος των κόκκινων ακμών με τιμή PMOS(δηλαδή αν τα εν σειρά NMOS είναι ίσα με τα εν παραλλήλω PMOS) και το αντίθετο. Κάνοντας αυτή τη διαδικασία για κάθε στοιχείο του possibleBreaks ,αποθηκεύεται σε ένα νέο πίνακα μια τιμή που δηλώνει την ταυτότητα της κάθε πύλης .
6. Μετά το πέρας της παραπάνω διαδικασίας ακολουθεί η εύρεση του netlist του κυκλώματος. Για κάθε στοιχείο του πίνακα possibleBreaks πηγαίνοντας στον αντίστοιχο κόμβο του γράφου η τιμές των εισόδων της κάθε πύλης είναι οι αριθμοί των διαφορετικών γειτονικών του κόμβων. Εκτελώντας την παραπάνω διαδικασία για κάθε τιμή του possibleBreaks , αποθηκεύονται οι τιμές εισόδου αλλά και η τιμή εξόδου (που είναι η τιμή του ίδιου του κόμβου) σε ένα νέο πίνακα .
7. Στη συνέχει αφού βρεθεί το netlist του κυκλώματος βρίσουμε τις διαφορετικές τιμές των κόμβων που συμμετέχουν για τη δημιουργία αυτού του netlist και δίνουμε σε κάθε έναν από αυτούς μια διαφορετική τιμή ξεκινώντας από το 1 και αυξάνοντας τροποποιώντας και τους αντίστοιχους κόμβους εισόδου και εξόδου. (το βήμα αυτό δεν είναι σημαντικό αλλά είναι χρήσιμο ώστε να μην υπάρχουν μεγάλα καινά ανάμεσα σε συνεχόμενους κόμβους του κυκλώματος και να γίνεται ευκολότερος έλεγχος για την ορθότητα του αρχείου εξόδου με το μάτι)
8. Έχοντας τελειώσει με την επεξεργασία της πληροφορίας καλείται η συνάρτηση που υλοποιήθηκε createFileOfGates () η οποία παίρνει αυτούς τους πίνακες που φτιάχτηκαν και με κατάλληλο τρόπο τους γράφει στο αρχείο εξόδου .
9. Τέλος καλείται η συνάρτηση convertInformation () που στην πραγματικότητα εκτελεί τον κώδικα αρχικοποίησης που είχε δημιουργηθεί στην άσκηση 3 ώστε να διαβαστεί ένα αρχείο που απεικονίζει διασυνδεδεμένες πύλες και διαβάζεται το αρχείο που μόλις δημιουργήθηκε. Μετά την αποθήκευση της πληροφορείς σε κατάλληλα arrays δημιουργείται ένα manualTest_vector και γίνεται η προσομοίωση του κυκλώματος ακριβώς όπως και στις προηγούμενες ασκήσεις .

Το αρχείο εισόδου που αντιπροσωπεύει τον 4bit RCA για την προιγούμενη άσκηση είναι το :

```
## LIBRARY
MyLib.LIB

##RAILS

## INPUTS
1,2,3,15,16,29,30,43,44
## OUTPUTS
13,27,41,55,56
## 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
G12,NAND_2 , IN,15,16,OUT,18
G13,NAND_2, IN,15,16,OUT,23
G14, NAND_2,  IN,16,18, OUT,20
G15, NAND_2,   IN,18,15, OUT,19
G16, NAND_2,  IN,20,19, OUT,21
G17, NAND_2,  IN,14,21, OUT,22
G18, NAND_2,  IN,21,14, OUT,24
G19, NAND_2,  IN,21,24, OUT,26
G20, NAND_2,  IN,24,14, OUT,25
G21, NAND_2,  IN,26,25, OUT,27
G22, NAND_2,  IN,23,22, OUT,28
G23,NAND_2 , IN,29,30,OUT,32
G24,NAND_2, IN,29,30,OUT,37
G25, NAND_2,  IN,30,32, OUT,34
G26, NAND_2,   IN,29,32, OUT,33
G27, NAND_2,  IN,33,34, OUT,35
G28, NAND_2,  IN,28,35, OUT,36
G29, NAND_2,  IN,35,28, OUT,38
G30, NAND_2,  IN,35,38, OUT,40
G31, NAND_2,  IN,28,38, OUT,39
G32, NAND_2,  IN,40,39, OUT,41
G33, NAND_2,  IN,37,36, OUT,42
G34,NAND_2 , IN,43,44,OUT,46
G35,NAND_2, IN,43,44,OUT,51
G36, NAND_2,  IN,44,46, OUT,48
G37, NAND_2,   IN,43,46, OUT,47
G38, NAND_2,  IN,47,48, OUT,49
G39, NAND_2,  IN,42,49, OUT,50
G40, NAND_2,  IN,49,42, OUT,52
G41, NAND_2,  IN,49,52, OUT,54
G42, NAND_2,  IN,42,52, OUT,53
G43, NAND_2,  IN,54,53, OUT,55
G44, NAND_2,  IN,51,50, OUT,56

## TESTBENCH
## TEST_IN
44 ; 43 ; 30 ; 29 ; 16 ; 15 ; 2 ; 1
## TEST_OUT
56 ; 55 ; 41     ; 27 ; 13
## TEST_VECTORS
0 ; 0 ; 0 ; 0 ; 0
## SIMULATE
## TEST_VECTORS
1 ; 0 ; 1 ; 0 ; 1
## SIMULATE
## TEST_VECTORS
1 ; 0 ; 1 ; 0 ; 1
## SIMULATE
## TEST_VECTORS
1 ; 1 ; 1 ; 1 ; 1
## SIMULATE
## END_TEST
## END_SIMULATION
```

Και το αρχείο εξόδου της προιγούμενη άσκηση που δέχθηκε ο νέος κώδικας σαν είσοδο είναι το:

NEWFILE1.txt

```
##RAILS
VCC 1 ; 7 ; 13 ; 19 ; 25 ; 31 ; 37 ; 43 ; 49 ; 55 ; 61 ; 67 ; 73 ; 79 ; 85 ; 91 ; 97 ; 103 ; 109 ; 115 ; 121 ; 127 ;
GND 6 ; 12 ; 18 ; 24 ; 30 ; 36 ; 42 ; 48 ; 54 ; 60 ; 66 ; 72 ; 78 ; 84 ; 90 ; 96 ; 102 ; 108 ; 114 ; 120 ; 126 ; 132
## INPUTS
2 ; 3 ; 32 ; 68 ; 69
## OUTPUTS
58 ; 124 ; 130
## NETLIST
U0 PMOS 2 1 4
U1 PMOS 3 1 4
U2 NMOS 2 4 5
U3 NMOS 3 5 6
U4 PMOS 2 7 10
U5 PMOS 3 7 10
U6 NMOS 2 10 11
U7 NMOS 3 11 12
U8 PMOS 3 13 16
U9 PMOS 4 13 16
U10 NMOS 3 16 17
U11 NMOS 4 17 18
U12 PMOS 2 19 22
U13 PMOS 4 19 22
U14 NMOS 2 22 23
U15 NMOS 4 23 24
U16 PMOS 22 25 28
U17 PMOS 16 25 28
U18 NMOS 22 28 29
U19 NMOS 16 29 30
U20 PMOS 32 31 34
U21 PMOS 28 31 34
U22 NMOS 32 34 35
U23 NMOS 28 35 36
U24 PMOS 28 37 40
U25 PMOS 32 37 40
U26 NMOS 28 40 41
U27 NMOS 32 41 42
U28 PMOS 28 43 46
U29 PMOS 40 43 46
U30 NMOS 28 46 47
U31 NMOS 40 47 48
U32 PMOS 32 49 52
U33 PMOS 40 49 52
U34 NMOS 32 52 53
U35 NMOS 40 53 54
U36 PMOS 46 55 58
U37 PMOS 52 55 58
U38 NMOS 46 58 59
U39 NMOS 52 59 60
U40 PMOS 10 61 64
U41 PMOS 34 61 64
U42 NMOS 10 64 65
U43 NMOS 34 65 66
U44 PMOS 68 67 70
U45 PMOS 69 67 70
U46 NMOS 68 70 71
U47 NMOS 69 71 72
U48 PMOS 68 73 76
U49 PMOS 69 73 76
U50 NMOS 68 76 77
U51 NMOS 69 77 78
U52 PMOS 69 79 82
U53 PMOS 70 79 82
U54 NMOS 69 82 83
U55 NMOS 70 83 84
U56 PMOS 70 85 88
U57 PMOS 68 85 88
U58 NMOS 70 88 89
U59 NMOS 68 89 90
U60 PMOS 82 91 94
U61 PMOS 88 91 94
U62 NMOS 82 94 95
U63 NMOS 88 95 96
U64 PMOS 64 97 100
U65 PMOS 94 97 100
U66 NMOS 64 100 101
U67 NMOS 94 101 102
U68 PMOS 94 103 106
U69 PMOS 64 103 106
U70 NMOS 94 106 107
U71 NMOS 64 107 108
U72 PMOS 94 109 112
U73 PMOS 106 109 112
U74 NMOS 94 112 113
U75 NMOS 106 113 114
U76 PMOS 106 115 118
U77 PMOS 64 115 118
U78 NMOS 106 118 119
U79 NMOS 64 119 120
U80 PMOS 112 121 124
U81 PMOS 118 121 124
U82 NMOS 112 124 125
U83 NMOS 118 125 126
U84 PMOS 76 127 130
U85 PMOS 100 127 130
U86 NMOS 76 130 131
U87 NMOS 100 131 132
## END_SIMULATION
```

Η δομη του αλγορίθμου που υλοποιήθηκε είναι :

```
//////////////////////////////////////////////old code////////////////////////////////////////////////////////////
define the arrays we need to store the information from file

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

define the arrays we need to store the information from Library

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

define the arrays we need to store the converted information we will create from Library and file arrays

convertInformation (…)       // converting information from file with gates to a flat netlist

define the arrays we need to store the information from Graph
```

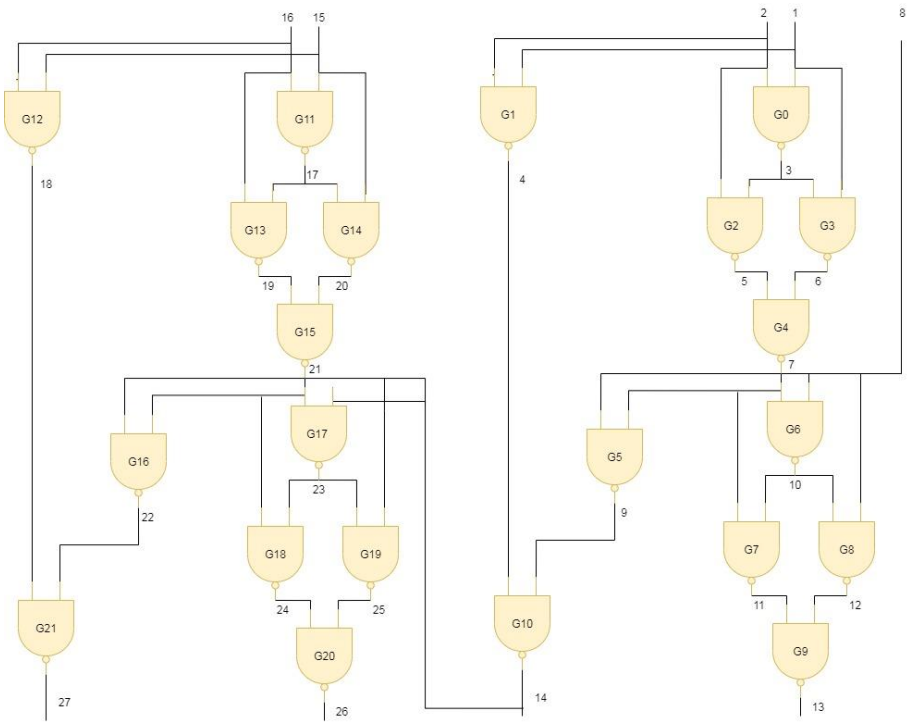
```
findPossibleBreak(…) ; ////finds the the nodes which are possible to be a good point to break the circuit

for(every possibleBreak){
    Graph gr=createGraph(possibleBreak);
    int validBreakCheck=colorGraph(gr)
    if(validBreakCheck==1){ //if coloures are not mixed
        validBreak[i]= possibleBreak
        i++;
    }
}

while(accuracyofvalidBreak[i]!=0){ // finds the best break point
     if(accuracyofvalidBreak[i]<min){
         min=accuracyofvalidBreak[i];
         bestbreak=validBreak[i];
     }
     i++;
    }

separateTheInpFileTo2(bestBreak)  //creates the 2 files
////////////////////////////////////////////old code////////////////////////////////////////////////////////////////////

////////////////////////////////////////////new code////////////////////////////////////////////////////////////////////

define the arrays we need to store the information from first separate file

initialization(…)                //it stores the information from 1st file we create in last lab to the appropriate arrays

// initialize the structure of every gate's graph (how many PMOS and NMOS are conected to every gate's output)
readLib(…);

Graph graph = createDAG(…);              //create the Directed Graph we need

findPossibleBreak(…);          // finde possible breaks for the fisrst file which are the divisions of the gates

// add the outputs of first file to possible brakes which are the division for the last gates
possibleBreakarray[]= possibleBreakarray[]  concatenate  outputs[]


/////////find how many PMOS and NMOS is conected to each PossibleBreak Node and check the number of serial
/////////an parallel PMOS and NMOS so we will know the identify of each gate
For(each possibleBreakearray[i]){
     int numofPMOS=0;
     int numofNMOS=0;
      for(each edge of graph(possibleBreakearray[i])){
            if(graph(possibleBreakearray[i])->edgeval==PMOS){ numofPMOS++;}
            if(graph(possibleBreakearray[i])->edgeval==NMOS){ numofNMOS++;}
      }

for(every gate from Lib){          // fill GatesOfBreakNode array
          if(possibleBreakearray[i] matches with gate ){
                    gateOfBreakNode[i] = gate;
          }
}
}/////////
```

```
For(each possibleBreakearray[i]){       // for every possibleBreak fill the array netofgates
      int tempArray[2]={};
      int position=0;
      for(each adjacent nod of graph(possibleBreakearray[i]))     // find the inputs of each gate
            if(adjacent ->number != tempArray[0]){
                  tempArray[position]= adjacent ->number
                  position++;
            }
      }
      // fill the Final netlist for each gate (netofgates)
      if(gatesOfBreakNode[i]==NAND_2 || gatesOfBreakNode[i]== NOR_2){
            netofgates[i][0]=tempArray[0];
            netofgates[i][1]=tempArray[1];
            netofgates[i][2]=possibleBreakarray[i];
     }
     else if(gatesOfBreakNode[i]==NOT){
            netofgates[i][0]=tempArray[0];
            netofgates[i][1]=tempArray[0];
           netofgates[i][2]=possibleBreakarray[i];
     }
}


//give a new value to every node of final netlist (netOfGates) . start from 1 and icreasing so we will not have big
//spaces between different nodes
ScaleDownNetOfGatesValues (…);

createFileOfGates(…) ;  // file

//(practicly the algorithm from lab 3) converting information from file with gates to a flat netlist
convertInformation()

///Simulate
     manual_test_vector={..};
     while (testvector[i] != 0){     //Loop for every testVector
            while(check==1){     //Loop that run until nothing changed so we done
                  check=0;
                  j=0;
                        while(transistor[j]!=0){  // Loop for every transistor

                          if(transistor[j]==PMOS && newNode[netlist[j][0]]==0 ){
                          …
                          check=1;
                           }
                          else if(transistor[j]==PMOS && newNode[netlist[j][0]]==1 ){
                         …
                         check=1;
                         }
                         else if(transistor[j]==NMOS && newNode[netlist[j][0]]==0 ){
                         …
                         check=1;
                         }
                         else if(transistor[j]==NMOS && newNode[netlist[j][0]]==1 ){
                         …
                         check=1;
                         }
```

```
                }
        }
i++;
}
```

Το αρχείο εξόδου που παρήχθησε :

```
## LIBRARY
MyLib.LIB
## RAILS
## INPUTS
1,2,8,15,16
## OUTPUTS
13,26,27
## NETLIST
G0,NAND_2,IN,1,2,OUT,3
G1,NAND_2,IN,1,2,OUT,4
G2,NAND_2,IN,2,3,OUT,5
G3,NAND_2,IN,1,3,OUT,6
G4,NAND_2,IN,6,5,OUT,7
G5,NAND_2,IN,8,7,OUT,9
G6,NAND_2,IN,7,8,OUT,10
G7,NAND_2,IN,7,10,OUT,11
G8,NAND_2,IN,8,10,OUT,12
G9,NAND_2,IN,11,12,OUT,13
G10,NAND_2,IN,4,9,OUT,14
G11,NAND_2,IN,15,16,OUT,17
G12,NAND_2,IN,15,16,OUT,18
G13,NAND_2,IN,16,17,OUT,19
G14,NAND_2,IN,17,15,OUT,20
G15,NAND_2,IN,19,20,OUT,21
G16,NAND_2,IN,14,21,OUT,22
G17,NAND_2,IN,21,14,OUT,23
G18,NAND_2,IN,21,23,OUT,24
G19,NAND_2,IN,23,14,OUT,25
G20,NAND_2,IN,24,25,OUT,26
G21,NAND_2,IN,18,22,OUT,27
```



Τα αποτελέσματα της προσομοίωσης για
test_vector={{0,0,0,0,0},{1,0,1,0,0},{1,0,1,0,1},{1,0,1,1,1},{0,0,1,0,1},{0,1,0,0,1},{1,1,1,1,1}}
που αντιστοιχούν σε {INP1,INP2,Cin,INP3,INP4} είναι :

```
For the 0 TEST_VECTORS the output 13 is : 0
the output 26 is : 0
the output 27 is : 0
For the 1 TEST_VECTORS the output 13 is : 0
the output 26 is : 1
the output 27 is : 0
For the 2 TEST_VECTORS the output 13 is : 0
the output 26 is : 0
the output 27 is : 1
For the 3 TEST_VECTORS the output 13 is : 0
the output 26 is : 1
the output 27 is : 1
For the 4 TEST_VECTORS the output 13 is : 1
the output 26 is : 1
the output 27 is : 0
For the 5 TEST_VECTORS the output 13 is : 1
the output 26 is : 1
the output 27 is : 0
For the 6 TEST_VECTORS the output 13 is : 1
the output 26 is : 1
the output 27 is : 1
```

Τα αποτελέσματα είναι αυτά που προβλέφθηκαν το οποίο αποδεικνύει την ορθότητα του αλγορίθμου .

Ενδεικτικό flowchart :