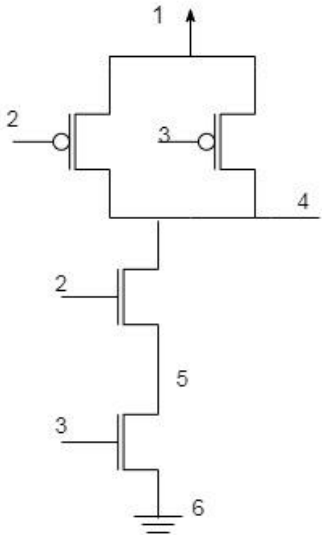
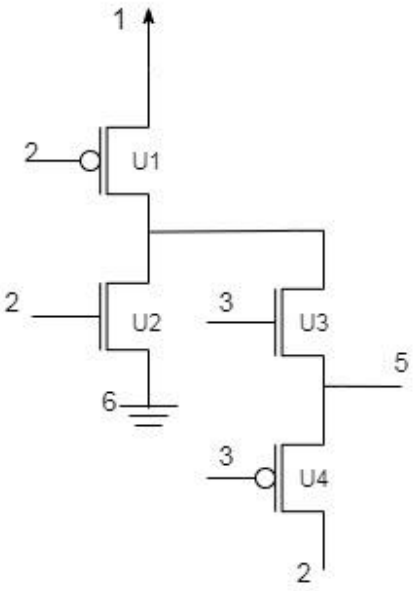
ΑΝΑΦΟΡΑ 3ου set

Κουρκουλος Αγγελος ΑΜ:2017030111

Η άσκηση αυτή ήταν μια επέκταση της προηγούμενης όπου είχε ως σκοπό τη μετατροπή ενός κυκλώματος από λογικές πύλες σε ένα ισοδύναμο κύκλωμα από συνδεδεμένα τρανζίστορ. Ποιό συγκεκριμένα είχαμε ως είσοδο ένα αρχείο όπως αυτό στην άσκηση 2 με τη διαφορά ότι στο nettlist του περιέγραφε ένα κυκλομα από συνδεδεμένες λογικές πύλες και είχε και μια αναφορά προς ένα αρχείο βιβλιοθήκης το οποίο περιείχε την πληροφορία για την μετατροπή κάθε λογικής πύλης που χρειάζεται σε κυκλομα από τρανζίστορ. Στη συγκεκριμένη βιβλιοθήκη που δημιούργησα με τίτλο MyLib.LIB συμπεριλαμβάνονται οι πύλες NOT , NOR_2 που μας δόθηκαν από την εκφώνηση καθώς και οι NAND_2 και XOR_2 που προσέθεσα με την παρακάτω συνδεσμολογία.

## GATE NAND_2 ## RAILS VCC 1 ; GND 6 ## INPUTS 2 ; 3 ## OUTPUTS 4 ## NETLIST U1 PMOS 2 1 4 U2 PMOS 3 1 4 U3 NMOS 2 4 5 U4 NMOS 3 5 6 ## END_GATE	

## GATE XOR_2 ## RAILS VCC 1 ; GND 6 ## INPUTS 2 ; 3 ## OUTPUTS 5 ## NETLIST U1 PMOS 2 1 4 U2 NMOS 2 4 6 U3 NMOS 3 4 5 U4 PMOS 3 5 2 ## END_GATE ##END LIBRARY	

Τι κύκλωμα που μας ζητήθηκε να δημιουργήσουμε και να προσομοιώσουμε είναι ενός αθριστη με διάσωση κρατούμενου RCA με 5 bit είσοδου (2 για κάθε full adder και Cin)και 3 bit έξοδου (S0 S1 Cout). Σε κάθε full adder το μέρος AND - OR γίνεται NAND-NAND λόγο De Morgan και η συνδεσμολογία του κυκλώματος καθώς και το αρχείο εισόδου (file2) φαίνονται παρακάτω:

file2:

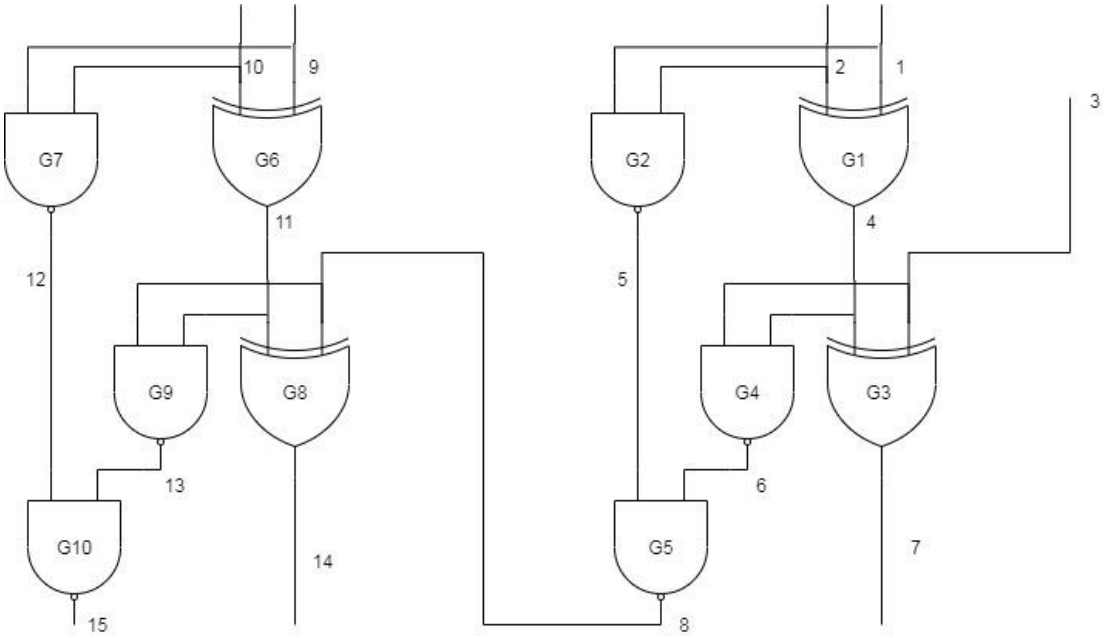
```
## LIBRARY
MyLib.LIB
##RAILS
## INPUTS
1,2,3,9,10
## OUTPUTS
7,14,15
## NETLIST
G1,XOR_2 ,IN,1,2,OUT,4
G2,NAND_2,IN,1,2,OUT,5
G3, XOR_2,  IN,3,4, OUT,7
G4,NAND_2,IN,3,4,OUT,6
G5,NAND_2,IN,6,5,OUT,8
G6,XOR_2 ,IN,9,10,OUT,11
G7,NAND_2,IN,9,10,OUT,12
G8, XOR_2,  IN,8,11, OUT,14
G9,NAND_2,IN,8,11,OUT,13
G10,NAND_2,IN,13,12,OUT,15

## TESTBENCH
## TEST_IN
10 ; 9 ; 2 ; 1 ; 3
## TEST_OUT
7 ; 14 ; 15
## TEST_VECTORS
0 ; 0 ; 0 ; 0 ; 0
## SIMULATE
```

```
## TEST_VECTORS
0 ; 0 ; 0 ; 0 ; 1
## SIMULATE
## TEST_VECTORS
0 ; 0 ; 0 ; 1 ; 0
## SIMULATE
## TEST_VECTORS
0 ; 0 ; 0 ; 1 ; 1
## SIMULATE
## TEST_VECTORS
0 ; 0 ; 1 ; 0 ; 0
## SIMULATE
## TEST_VECTORS
0 ; 0 ; 1 ; 0 ; 1
## SIMULATE
## TEST_VECTORS
0 ; 0 ; 1 ; 1 ; 0
## SIMULATE
## TEST_VECTORS
0 ; 0 ; 1 ; 1 ; 1
## SIMULATE
## TEST_VECTORS
0 ; 1 ; 1 ; 1 ; 1
## SIMULATE
## TEST_VECTORS
1 ; 0 ; 0 ; 0 ; 0
## SIMULATE
```

```
## TEST_VECTORS
1 ; 0 ; 0 ; 0 ; 1
## SIMULATE
## TEST_VECTORS
1 ; 0 ; 1 ; 0 ; 1
## SIMULATE
## TEST_VECTORS
1 ; 0 ; 1 ; 1 ; 0
## SIMULATE
## TEST_VECTORS
1 ; 1 ; 1 ; 1 ; 1
## SIMULATE
## END_TEST
## END_SIMULATION
```

Όπως βλέπουμε έχουμε πάρει αρκετά δείγματα εισόδων Test_Vectors συμπεριλαμβανομένων όλων των ακραίων περιπτώσεων ώστε να είμαστε σίγουροι για τη λειτουργία του κυκλώματος. Επίσης πρέπει να σημειωθεί ότι οι κόμβοι είσοδου Test_In που βρίσκονται στο αρχείο είναι σε διαφορετική σειρά από ότι είναι οι είσοδοι Inputs έτσι ώστε το test_vector να έχει το Cin και στη συνέχεια το list significant bit με τη σειρά ξεκινώντας από τα δεξιά του.

Το σχέδιο του κυκλώματος καθώς και οι κόμβοι του φαίνονται παρακάτω:



οι βασικές επεκτάσεις του αλγορίθμους είναι η συνάρτηση initializeLib() που φέρνει στο πρόγραμμα την πληροφορία από τη βιβλιοθήκη και οι συναρτήσεις createFinalNet() και createFinalNetAddMos() οι οποίες συνδυάζουν τις αποθηκευμένες πληροφορίες που έχουν αναγνωστεί από τα αρχεία εισόδου και δημιουργούν ένα τελικό netlist και τις υπόλοιπες απαραίτητες πληροφορίες που χρειάζονται για να περιγράψουν όλο το κύκλωμα σαν συνδεδεμένους κόμβους

Η δομη του αλγορίθμου που υλοποιήθηκε είναι :

```
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



while(flag==1){  //// loop we need in case that the gates are not with the corect order in the starting netlist

      flag=0;

        int c=0;

        while(netlist[c]!= 0){  ////Loop for every gate

              if(logicgate[c]==NOT){  //// for every case of Lgate call the func createFinalNet to add the gate to final
                        createFinalNet(…);                                                                    //// netlist
              }
             else if(logicgate[c]==NOR_2){
                        createFinalNet(…)
              }
              else if(logicgate[c]==NAND_2){
                        createFinalNet(…)
              }
              else if(logicgate[c]==XOR_2){
                        createFinalNet(…)
              }
              else if(logicgate[c]==NMOS || PMOS){ ///add the transistor to final netlist
                        createFinalNetAddMos(…)
              }
        }
}

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++;
}
```

Τα αποτελέσματα του αλγορίθμους για το αρχείο εισόδου που υπάρχει παραπάνω είναι:

```
For the 0 TEST_VECTORS  the output 7 is : 0
 the output 14 is : 0
 the output 15 is : 0
For the 1 TEST_VECTORS  the output 7 is : 1
 the output 14 is : 0
 the output 15 is : 0
For the 2 TEST_VECTORS  the output 7 is : 1
 the output 14 is : 0
 the output 15 is : 0
For the 3 TEST_VECTORS  the output 7 is : 0
 the output 14 is : 1
 the output 15 is : 0
For the 4 TEST_VECTORS  the output 7 is : 1
 the output 14 is : 0
 the output 15 is : 0
For the 5 TEST_VECTORS  the output 7 is : 0
 the output 14 is : 1
 the output 15 is : 0
For the 6 TEST_VECTORS  the output 7 is : 0
 the output 14 is : 1
 the output 15 is : 0
For the 7 TEST_VECTORS  the output 7 is : 1
 the output 14 is : 1
 the output 15 is : 0
For the 8 TEST_VECTORS  the output 7 is : 1
 the output 14 is : 0
 the output 15 is : 1
For the 9 TEST_VECTORS  the output 7 is : 0
 the output 14 is : 1
 the output 15 is : 0
For the 10 TEST_VECTORS  the output 7 is : 1
 the output 14 is : 1
 the output 15 is : 0
For the 11 TEST_VECTORS  the output 7 is : 0
 the output 14 is : 0
 the output 15 is : 1
For the 12 TEST_VECTORS  the output 7 is : 0
 the output 14 is : 0
 the output 15 is : 1
For the 13 TEST_VECTORS  the output 7 is : 1
 the output 14 is : 1
 the output 15 is : 1
```

Μπορούμε να δούμε ότι για τις άκρες περιπτώσεις:

Test_Vector0: 0 ; 0 ; 0 ; 0 ; 0 = B1;B0;A1;A0;Cin

Result : out 15 ; out 14 ; out 7 = 0 ; 0 ; 0 = Cout ; S1 ; S0

Test_Vector13: 1 ; 1 ; 1 ; 1 ; 1 = B1;B0;A1;A0;Cin

Result : out 15 ; out 14 ; out 7 = 1 ; 1 ; 1 = Cout ; S1 ; S0

Ενώ για μια μέση περιπτώσεις π.χ. :

Test_Vector7: 0 ; 0 ; 1 ; 1 ; 1 = B1;B0;A1;A0;Cin

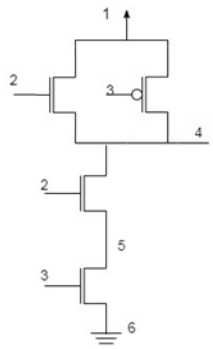
Result : out 15 ; out 14 ; out 7 = 0 ; 1 ; 1 = Cout ; S1 ; S0

Σε περίπτωση που υπάρχει κάποιο σφάλμα για παράδειγμα αλλάξουμε το πρώτο τρανζίστορ της πύλης NAND_2 της βιβλιοθήκη από PMOS σε NMOS :

```
For the 0 TEST_VECTORS  the output 7 is : 0
 the output 14 is : 1
 the output 15 is : 1
Node 5 of gate NAND_2 number 3 is short-circuited
Node 5 of gate NAND_2 number 6 is short-circuited
For the 1 TEST_VECTORS  the output 7 is : 1
 the output 14 is : 1
 the output 15 is : 1
Node 5 of gate NAND_2 number 3 is short-circuited
Node 5 of gate NAND_2 number 6 is short-circuited
For the 2 TEST_VECTORS  the output 7 is : 1
 the output 14 is : Z
 the output 15 is : 1
Node 5 of gate NAND_2 number 6 is short-circuited
For the 3 TEST_VECTORS  the output 7 is : 0
 the output 14 is : 1
 the output 15 is : 1
Node 5 of gate NAND_2 number 2 is short-circuited
Node 5 of gate NAND_2 number 3 is short-circuited
Node 5 of gate NAND_2 number 6 is short-circuited
For the 4 TEST_VECTORS  the output 7 is : 1
 the output 14 is : Z
 the output 15 is : 1
Node 5 of gate NAND_2 number 6 is short-circuited
For the 5 TEST_VECTORS  the output 7 is : 0
 the output 14 is : 1
 the output 15 is : 1
Node 5 of gate NAND_2 number 2 is short-circuited
Node 5 of gate NAND_2 number 6 is short-circuited
For the 6 TEST_VECTORS  the output 7 is : 0
 the output 14 is : 1
 the output 15 is : 1
Node 5 of gate NAND_2 number 1 is short-circuited
Node 5 of gate NAND_2 number 3 is short-circuited
Node 5 of gate NAND_2 number 6 is short-circuited
For the 7 TEST_VECTORS  the output 7 is : 1
 the output 14 is : 1
 the output 15 is : 1
Node 5 of gate NAND_2 number 1 is short-circuited
Node 5 of gate NAND_2 number 3 is short-circuited
Node 5 of gate NAND_2 number 6 is short-circuited
For the 8 TEST_VECTORS  the output 7 is : 1
 the output 14 is : 0
 the output 15 is : 1
Node 5 of gate NAND_2 number 1 is short-circuited
Node 5 of gate NAND_2 number 3 is short-circuited
Node 5 of gate NAND_2 number 5 is short-circuited
Node 5 of gate NAND_2 number 6 is short-circuited
```

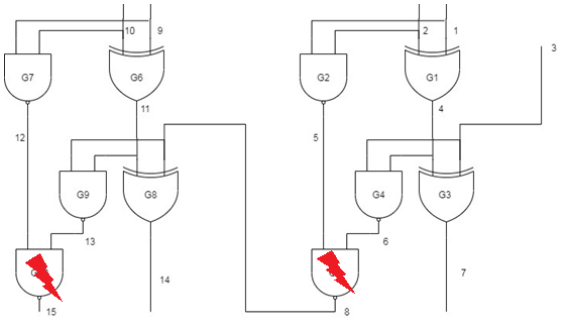
Παρατηρούμε ότι το πρόγραμμα βρίσκει πολλά βραχικικλοματα στο κύκλωμα μας οπότε καταλαβαίνουμε ότι το κυκλομα μας δε λειτουργεί σωστά.

Βρίσκει το πρόγραμμα τα βραχικικλοματα σε σωστά σημεία? Ας πάρουμε σαν παράδειγμα την πρώτη περίπτωση όπου έχουμε test_vector=0 ; 0 ; 0 ; 0



Βλέπουμε ότι η 1η (με τη σειρά που μπήκαν στο netlist) NAND έχει ως είσοδο το 0 ; 0 όπως και η 2 NAND αφού παίρνει ως είσοδο την έξοδο της 1ης XOR και το Cin που είναι 0 οπότε έχουν ως αποτέλεσμα στην έξοδο το 1 που τυχαία είναι σωστο. Από την άλλη η 3 NAND παίρνει σαν εισόδων τις εξόδους των 2 άλλων NAND δλδ 1 ; 1 οπότε παθαίνεις

βραχυκύκλωμα στον κόμβο 5 το οποίο είναι ένα από τα βραχικικλοματα που βρίσκει ο αλγοριθμος.



Βραχικικλοματα για το παραπάνω παράδειγμα .

Σημείωση: το file1 που υπάρχει στο παραδοτέο είναι το κύκλωμα and από μια NOR και 2 NOT που μας δίνεται από την εκφώνηση