ΕΡΓΑΣΤΗΡΙΟ 6

06/06/2021

ΣΤΟΙΧΕΙΑ MODEL CHECKING ΚΑΙ FORMAL VERIFICATION: ΟΜΟΜΟΡΦΙΚΟΙ ΓΡΑΦΟΙ

Ντουνέτας Δημήτρης
ΑΜ: 2016030141

Εισαγωγή

Σκοπός της έκτης άσκησης είναι η δημιουργία ενός προγράμματος για έλεγχο 2 αρχείων σε επίπεδο πυλών ώστε να μπορεί να ελεγχτεί η ισοδυναμία 2 μοντέλων. Αφού λάβουμε ως είσοδο ένα αρχείο με συνδεσμολογία πυλών και ένα αρχείο που γνωρίζουμε ότι είναι σωστό και έχει τη συνδεσμολογία που θέλουμε να ελέγξουμε, τα διαβάζουμε και βρίσκουμε αν τα αρχεία είναι ισοδύναμα. Αυτό σημαίνει ότι όλες οι πύλες τους και οι συνδέσεις τους είναι ταυτόσημες.

Λειτουργία του Προγράμματος

Το πρόγραμμα ξεκινάει διαβάζοντας τα 2 αρχεία και δημιουργώντας 2 πίνακες, έναν για το κάθε αρχείο με τις πληροφορίες που είναι γραμμένες στα αρχεία ώστε να μπορούμε να τις επεξεργαστούμε. Ξεκινάει διαβάζοντας το αρχείο του Golden Standard γραμμή προς γραμμή κάνοντας χρήση της συνάρτησης getline() και σπάει το αρχείο σε κομμάτια με χρήση της συνάρτησης strtok(). Όλα τα χρήσιμα μέρη αποθηκεύονται σε μεταβλητές για να χρησιμοποιηθούν αργότερα. Δημιουργείται ένας γράφος που χαρακτηρίζει το Netlist των πυλών που μας έχει δοθεί. Ύστερα γίνεται η ίδια διαδικασία και το αρχείο εισόδου. Στη συνέχεια γίνονται κάποιοι βασικοί απλοί έλεγχοι όπως η σύγκριση του αριθμού των πυλών του κάθε αρχείο και η σύγκριση του είδους των πυλών κάθε αρχείο. Με αυτό τον τρόπο μπορούμε να γλυτώσουμε περαιτέρω ελέγχους αφού οι γράφοι των 2 αρχείων αποκλείεται να είναι ομομορφικοί αν δεν περάσουν αυτούς τους ελέγχου. Τέλος εξάγουμε τα μονοπάτια των γράφων του κάθε αρχείο και ελέγχουμε πάλι αν ο αριθμός μονοπατιών των κυκλωμάτων είναι ίσος. Τέλος κωδικοποιούμε τα μονοπάτια και ελέγχουμε την ταύτιση τους. Αν βρούμε ταύτιση σε όλα τα μονοπάτια προσπαθούμε να βρούμε τις ισοδυναμίες μεταξύ πυλών και ακμών και ενημερώνουμε τον χρήστη.

Ανάλυση του Κώδικα

Αρχικοποιήσεις συναρτήσεων και μεταβλητών:

```
//Lab6 Functions
void getGoldenInfo(FILE* fpGolden,FILE *fpOut);//gets the info from the Golden Standard file
void startingChecks();
void mainChecking();
void DFS(int current, int node, int *path ,bool golden);
bool checkIfEnd(int current,bool golden);
int find_children(int current,int *children,bool golden);
void addPath(int *path, bool golden);
void EncodePaths();
void checkFilePaths();
void createReport(FILE* fpOut);

//Functions used
void checkStim(int argc, char *argv[]);//checks if a stim file is entered
void getCircuitinfo(FILE* fpIn,FILE* fpOut);//gets the info from the stim file
void printNetlist(FILE* fpOut);


//Memory allocations for Circuit Data.
int VCC,GND,numOfInputs,numOfOutputs,netlistRows,netlistCols;
int GoldenVCC,GoldenGND,GoldenNumOfInputs,GoldenNumOfOutputs,GoldenNetlistRows,GoldenNetlistCols;
int* inputs;
int* GoldenInputs;
int* outputs;
int* GoldenOutputs;
int **netlist;
int **GoldenNetlist;
int count,count1;
bool foundGate = false;
int NOTnum, NANDnum, NORnum, GoldenNOTnum, GoldenNANDnum, GoldenNORnum;
int paths[100][100];
int numOfpaths=0,GoldenNumOfPaths=0;
int GoldenPaths[100][100];
int Encpaths[100][100];
int GoldenEncPaths[100][100];
int EqualGates[100][2];
int numOfEqGates = 0;
```

Η main συνάρτηση:

```
int main(int argc, char *argv[])
{
    //Check if circuit file is present
    checkStim(argc, argv);
    //open circuit file
    FILE* fpIn = fopen(argv[1],"r");
    FILE* fpGolden = fopen(argv[2],"r");
    printf("Golden File for comparison is %s\n",argv[2]);
    if (fpIn == NULL)
    {
        printf("Could not find input file");
    }
    if (fpGolden == NULL)
    {
        printf("Could not find Golden Standard file");
    }
    char file1[64];
    char dot = '.';

    strcpy(file1,strtok(argv[1],&dot));
    strcat(file1,"_out1.txt");

    FILE *fp1 = fopen(file1,"w");
    //get Golden Standard File Info
    getGoldenInfo(fpGolden,fp1);
    //get circuit file info
    getCircuitinfo(fpIn,fp1);
    printNetlist(fp1);
    startingChecks(fp1);
    mainChecking(fp1);
    createReport(fp1);

    return 0;
}
```

Η συνάρτηση getCircuitInfo() δουλεύει με τον ίδιο τρόπο όπως στα προηγούμενα εργαστήρια , δηλαδή αποθηκεύει δεδομένα από το αρχείο εισόδου, η μόνη διαφορά σε αυτό το εργαστήριο είναι ότι προσαρμόστηκε ελαφρώς ώστε να διαβάζει αρχείο πυλών και όχι επίπεδο Netlist CMOS.

Επίσης προστέθηκε η συνάρτηση getGoldeninfo() που κάνει την ίδια διαδικασία για το Golden Standard αρχείο.

Ψήγμα κώδικα από τη συνάρτηση startingChecks():

```
//Simple checks for number of gates and gate types
void startingChecks(FILE * fpOut)
{
    //Check for correct number of Gates
    if (netlistRows != GoldenNetlistRows)
    {
        printf("\nThe 2 Models are not Equivalent!\nThe 2 Circuits have different number of Gates\n");
        fprintf(fpOut,"\nThe 2 Models are not Equivalent!\nThe 2 Circuits have different number of Gates\n");
        exit(1);
    }
    //Check that the types of the gates are similar
    NOTnum = 0;
    NANDnum = 0;
    NORnum = 0;
    GoldenNOTnum = 0;
    GoldenNANDnum = 0;
    GoldenNORnum = 0;
    int n=0;
    //Count Input file's Gate types
    // 0 = NOR_2
    // 1 = NAND_2
    // 2 = NOT
    for (int i = 0; i < netlistRows; i++)
    {
        n=netlist[i][1];
        switch (n)
        {
            case 0:
                NORnum++;
                break;
            case 1:
                NANDnum++;
                break;
            case 2:
                NOTnum++;
                break;
            default:
                break;
        }
    }
```

```
printf("\nNOR Gates: %d NAND Gates: %d NOT Gates: %d",NORnum,NANDnum,NOTnum);

//Count Golden Standard file's Gate types
// 0 = NOR_2
// 1 = NAND_2
// 2 = NOT
for (int i = 0; i < GoldenNetlistRows; i++)
{
    n=GoldenNetlist[i][1];
    switch (n)
    {
        case 0:
            GoldenNORnum++;
            break;
        case 1:
            GoldenNANDnum++;
            break;
        case 2:
            GoldenNOTnum++;
            break;
        default:
            break;
    }
}
printf("\nNOR Gates: %d NAND Gates: %d NOT Gates: %d",GoldenNORnum,GoldenNANDnum,GoldenNOTnum);

//Check if Gate types are similar
if (GoldenNORnum!= NORnum||GoldenNANDnum!= NANDnum||GoldenNOTnum!= NOTnum)
{
    printf("\nThe 2 Models are not Equivalent!\nThe Gate types are not the same!");
    fprintf(fpOut,"\nThe 2 Models are not Equivalent!\nThe 2 Circuits have different number of Gates\n");
    exit(1);
}
```

Η συγκεκριμένη συνάρτηση είναι υπεύθυνη για τους απλούς αρχικούς ελέγχους στα 2 αρχεία ώστε να σιγουρέψουμε ότι είναι σωστός ο αριθμός πυλών, ότι οι τύποι των πυλών είναι σωστοί και υπάρχει ο σωστός αριθμός από κάθε είδους πύλη.

Ψήγμα κώδικα από τη συνάρτηση mainChecking():

```
//Check to find if all paths of Golden standard are same with Input Circuit
void mainChecking(FILE *fpOut)
{
    int startingPoints[100];
    int GoldenStartingPoints[100];
    int startingPointNum = 0;
    int GoldenStartingPointNum = 0;
    int currPath[100];

    //Find Starting Points of the Path for Input Circuit
    for (int i = 0; i < netlistRows; i++)
    {
        for (int j = 0; j < numOfInputs; j++)
        {
            //Check if First Gate Input is Circuit Input
            if (netlist[i][2]==inputs[j])
            {
                startingPoints[startingPointNum]=i;
                startingPointNum++;
            }
            //Check if Second Gate Input is Circuit Input if the 2 inputs are not equal and the gate is not "NOT"
            if ((netlist[i][3]==inputs[j])&&(netlist[i][1]!=2)&&(netlist[i][3]!=netlist[i][2]))
            {
                startingPoints[startingPointNum]=i;
                startingPointNum++;
            }
        }
    }
    //Find Starting Points of the Path for Golden Standard Circuit
    for (int i = 0; i < GoldenNetlistRows; i++)
    {
        for (int j = 0; j < GoldenNumOfInputs; j++)
        {
            //Check if First Gate Input is Circuit Input
            if (GoldenNetlist[i][2]==GoldenInputs[j])
            {
                GoldenStartingPoints[GoldenStartingPointNum]=i;
                GoldenStartingPointNum++;
            }
            //Check if Second Gate Input is Circuit Input if the 2 inputs are not equal and the gate is not "NOT"
            if ((GoldenNetlist[i][3]==GoldenInputs[j])&&(GoldenNetlist[i][1]!=2)&&(GoldenNetlist[i][3]!=GoldenNetlist[i][2]))
            {
                GoldenStartingPoints[GoldenStartingPointNum]=i;
                GoldenStartingPointNum++;
            }
        }
    }
```

Η συγκεκριμένη συνάρτηση εκτελεί την βασική διαδικασία ελέγχου για τα 2 μοντέλα ώστε να σιγουρευτούμε ότι είναι ισοδύναμα. Ξεκινά βρίσκοντας σημεία έναρξης των μονοπατιών. Αυτό το κάνει ελέγχοντας όλες τις εισόδους των πυλών και στα 2 αρχεία και αποθηκεύει αυτές που έχουν ως είσοδο μια από τις εισόδους του κυκλώματος.

Ψήγμα κώδικα από τη συνάρτηση mainChecking():

```
for (int i = 0; i < startingPointNum; i++)
{
    // Start Depth First Search for Input file to find paths
    DFS(startingPoints[i],0,currPath,false);
}
for (int i = 0; i < GoldenStartingPointNum; i++)
{
    // Start Depth First Search for Golden Standard file to find paths
    DFS(GoldenStartingPoints[i],0,currPath,true);
}
```

Στη συνέχεια εκτελούμε Depth First Search για όλα τα σημεία έναρξης. Με αυτό τον τρόπο βρίσκουμε όλα τα μονοπάτια που διασχίζουν τα 2 κυκλώματα από την είσοδο ως την έξοδο τους.

Η συγκεκριμένη συνάρτηση εκτελεί τον έλεγχο για την εύρεση μια πύλης NAND. Αρχικά για να βρούμε πιο εύκολα μια πύλη NAND αναζητούμε τα 2 παράλληλα PMOS πρώτα. Οπότε αυτή η διαδικασία προχωράει όταν

Ψήγμα κώδικα από τη συνάρτηση mainChecking()

```
//Array sorting in descending order using bubbleshort for Input File's paths
int a=0;
for (int i = 0; i < numOfpaths-1; ++i){
    int len_j_plus1,len_j,j;
    int temp[100];
    for (j = 0; j < numOfpaths-i-1; j++){
        // Calculate current path length
        for (a = 0; paths[j][a]!= -1; a++)
        {
        }
        len_j = a;
        // Calculate next path length
        for (a = 0; paths[j+1][a]!= -1; a++)
        {
        }
        len_j_plus1 = a;
        // If next path is longer move up
        if (len_j_plus1 > len_j)
        {
            // Store the current path temporarily
            for (a = 0; paths[j][a]!= -1; a++)
            {
                temp[a]=paths[j][a];
            }
            temp[a] = -1;
            // Change the paths
            for (a = 0; paths[j+1][a]!= -1; a++)
            {
                paths[j][a] = paths[j+1][a];
            }
            paths[j][a] = -1;
            // Move the current path down
            for (a=0;temp[a]!=-1;a++){
                paths[j+1][a] = temp[a];
            }
            paths[j+1][a] = -1;
        }
    }
}
```

Αφού βρούμε όλα τα μονοπάτια και ελέγξουμε οτι ο αριθμός τους είναι ίσος στα 2 αρχεία χρησιμοποιούμε αλγόριθμο Bubble Short ώστε να ταξινομήσουμε τα μονοπάτια που βρήκαμε και για τα 2 κυκλώματα σε φθίνουσα σειρά μήκους μονοπατιού. Αυτό το κάνουμε για να μας βοηθήσει στον έλεγχο αργότερα να ελέγξουμε πρώτα τα μακρύτερα μονοπάτια που πιθανότερα θα μας δώσουν καλύτερη και πιο ακριβή πληροφορία σχετικά με την ισοδυναμία των πυλών των 2 κυκλωμάτων.

Ψήγμα κώδικα από τη συνάρτηση DFS()

```
// Depth First Search algorithm to find paths
void DFS(int current, int node, int *path ,bool golden)
   {
      //Check if current node is an Output node
      if (checkIfEnd(current,golden))
      {
         path[node] = current;
         path[node+1] = -1;
         // Store current path to paths arrays
         addPath(path,golden);
         return;
      }
      //Find all possible children of current node
      int children[10];
      int childrenNum = find_children(current,children,golden);
      //Add current node to the path
      path[node] = current;
      // Call DFS for all the children nodes
      for (int i = 0; i < childrenNum; i++)
      {
          node++;
          DFS(children[i],node,path,golden);
          path[node]=-1;
          node--;
      }
      return;
   }
```

Εκτελούμε τον γνωστό μας αλγόριθμο Depth First Search για να βρούμε όλα τα μονοπάτια που διασχίζονται στα 2 κυκλώματα. Αρχικά ελέγχουμε αν ο κόμβος που βρισκόμαστε είναι έξοδος του κυκλώματος και αν είναι αποθηκεύουμε το μονοπάτι που έχουμε βρει στο πίνακα με τα μονοπάτια.

Αλλιώς αναζητούμε τα παιδιά του συγκεκριμένου κόμβου και εκτελούμε ξανά αναδρομικά Depth First Search για όλα τα παιδιά του κόμβου που βρισκόμαστε.

Ψήγμα κώδικα από τη συνάρτηση EncodePaths()

```
// Encode paths in order to check them
void EncodePaths()
{
    int temp;
    //Encode Input File paths
    for (int i = 0; i < numOfpaths; i++)
    {
        int j;
        // Add -2 in the beggining of the path for checking purposes
        Encpaths[i][0]=-2;
        // For every gate add 0 if NOR_2 , 1 if NAND_2 or 2 if NOT
        for (j = 1; paths[i][j]!= -1; j++)
        {
            temp = paths[i][j];
            if (netlist[temp][1]==0)
            {
                Encpaths[i][j] = 0;
            }
            else if (netlist[temp][1]==1)
            {
                Encpaths[i][j] = 1;
            }
            else if (netlist[temp][1]==2)
            {
                Encpaths[i][j] = 2;
            }
        }
        Encpaths[i][j]=-1;
    }
    //Encode Golden Standard File paths
    for (int i = 0; i < GoldenNumOfPaths; i++)
    {
        int j;
        // Add -2 in the beggining of the path for checking purposes
        GoldenEncPaths[i][0] = -2;
        // For every gate add 0 if NOR_2 , 1 if NAND_2 or 2 if NOT
        for (j = 1; GoldenPaths[i][j]!= -1; j++)
        {
            temp = GoldenPaths[i][j];
            if (GoldenNetlist[temp][1]==0)
            {
                GoldenEncPaths[i][j] = 0;
            }
            else if (GoldenNetlist[temp][1]==1)
            {
                GoldenEncPaths[i][j] = 1;
            }
            else if (GoldenNetlist[temp][1]==2)
            {
                GoldenEncPaths[i][j] = 2;
            }
        }
        GoldenEncPaths[i][j] = -1;
    }
```

Η συγκεκριμένη συνάρτηση είναι υπεύθυνη για την κωδικοποίηση των μονοπατιών ώστε να μπορούμε στην συνέχεια να συγκρίνουμε τα κωδικοποιημένα μονοπάτια μεταξύ τους. Διαβάζει το μονοπάτι που μας έχει δοθεί από την DFS και αποτελείται από τις θέσεις των πυλών στον αρχικό πίνακα Netlist που τους αποθηκεύσαμε και το μετατρέπει σε ένα μονοπάτι από ήδη πυλών. Για κάθε πύλη NOR_2 βάζει τον αριθμό 0 ,για κάθε πύλη NAND_2 τον αριθμό 1 και για κάθε πύλη NOT τον αριθμό 2.

Για παράδειγμα αν ένα μονοπάτι αποτελείτε από τα παρακάτω:

Όπου 0 η θέση 0 του πίνακα πυλών , 2 η θέση 2 και ούτω καθεξής.

Αν θεωρήσουμε ότι στη θέση 0 βρίσκεται μια πύλη NAND_2
Στη θέση 2 μια πύλη NOR_2, στη θέση 4 μια πύλη NOT και στη θέση 10 μια πύλη NAND_2. Τότε θα έχουμε:

Path	0	2	4	10
Encoded	1	0	2	1

Αφού γίνει αυτό για όλες τις διαδρομές προχωράμε σε έλεγχο των μονοπατιών.

Ψήγμα κώδικα από τη συνάρτηση checkFilePaths()

```
// Check paths of files to determine gate equalities
void checkFilePaths()
{
    // Memory preping
    for (int i = 0; i < 100; i++)
    {
        EqualGates[i][0]=-1;
        EqualGates[i][1]=-1;
    }
    int GoldenPathLenght = 0;
    int pathLength = 0;
    for (int i = 0; i < GoldenNumOfPaths; i++)
    {
        bool same = true;
        int j;
        int k;
        for (j = 0; j < numOfpaths; j++)
        {
            same = true;
            if (Encpaths[j][0]==-3)
            {
                continue;
            }
            // Measure the path length of Golden Standard file's paths
            for (GoldenPathLenght = 0; GoldenPaths[i][GoldenPathLenght]!=-1; GoldenPathLenght++)
            {

            }
            // Measure the path length of Input file's paths
            for (pathLength = 0; paths[j][pathLength]!=-1; pathLength++)
            {

            }
            printf("\n%d  %d",GoldenPathLenght,pathLength);
            // Check if the 2 paths are equal
            if (GoldenPathLenght!=pathLength)
            {
                same = false;
                continue;
            }
            // Check if every gate type in the path is same
            for (k = 1; k < GoldenPathLenght; k++)
            {
                if (GoldenEncPaths[i][k]!=Encpaths[j][k])
                {
                    same = false;
                    break;
                }
            }
            // Reduce cpu time if find a match
            if (same)
            {
                break;
            }
        }
        // If found similar path mark them and create an equality Array
        if (same)
        {
            // Mark paths
            GoldenEncPaths[i][0]=-3;
            Encpaths[j][0]=-3;
            // Create equality array
            for (int gold = 1; GoldenPaths[i][gold]!=-1; gold++)
            {
                // For current path check all nodes of golden Standard if already in equality array dont add them
                for (k = 0; k < numOfEqGates; k++)
                {
                    if(GoldenPaths[i][gold]==EqualGates[k][0]||paths[j][gold]==EqualGates[k][1])
                    {
                        break;
                    }
                }
                // If not in array add them
                if (k==numOfEqGates)
                {
                    EqualGates[numOfEqGates][0]=GoldenPaths[i][gold];
                    EqualGates[numOfEqGates][1]=paths[j][gold];
                    // temp++;
                    numOfEqGates++;
                }
            }
        }
```

Η συνάρτηση αυτή , ελέγχει όλα τα μονοπάτια που έχουμε κωδικοποιήσει και βρίσκει αυτά που περνάνε από τους ίδιου τύπους πυλών. Πρέπει να βρει μονοπάτια με το ίδιο μήκος και όλες τις πύλες που περνάνε μια προς μια με την ίδια σειρά να είναι ίδιες. Αν μπορέσει να το κάνει για όλα τα μονοπάτια αποφασίζουμε ότι οι γράφοι είναι ομομορφικοί. Κάθε φορά που θα βρει ένα τέτοιο μονοπάτι το αποκωδικοποιούμε και ελέγχουμε τις πύλες που έχουμε εξισώσει αν υπάρχουν ήδη στον πίνακα ισοδυναμιών. Αν υπάρχουν τις προσπερνούμε αλλιώς τις προσθέτουμε. Και έτσι έχουμε δημιουργήσει ένα πίνακα από ισοδυναμίες πυλών που μας επιτρέπουν να καθορίσουμε όλες τις ισοδυναμίες μεταξύ των 2 κυκλωμάτων.

Ψήγμα κώδικα από τη συνάρτηση createReport()

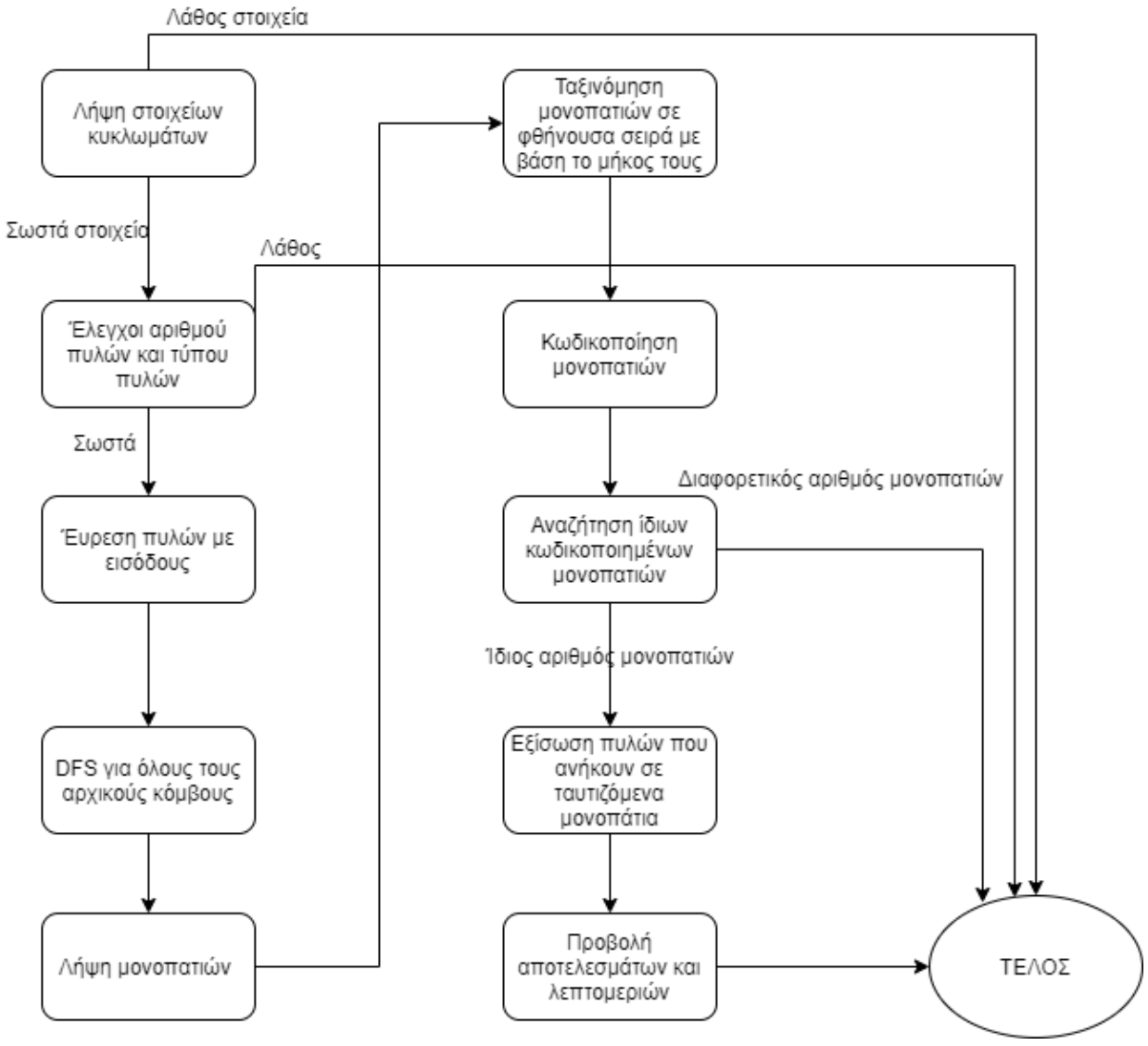
```
void createReport(FILE * fpOut)
{
    printf("\n");
    printf("The 2 Models are Equivalent \n\n");
    fprintf(fpOut,"The 2 Models are Equivalent \n\n");
    printf("Their gate Equalities are as described bellow\n\n");
    fprintf(fpOut,"Their gate Equalities are as described bellow\n\n");
    printf("Input File's Specs. Gate number(First Input,Second Input)-->Output == Golden Standards File's Specs. Gate number(First Input,Second Input)-->Output\n\n");
    fprintf(fpOut,"Input File's Specs. Gate number(First Input,Second Input)-->Output == Golden Standards File's Specs. Gate number(First Input,Second Input)-->Output\n\n");

    int difference = GoldenNetlistRows - numOfEqGates;
    if (difference==1)
    {
        // bool exist = true;
        for (int i = 0; i < GoldenNetlistRows; i++)
        {
            int j;
            for (j = 0; j < numOfEqGates; j++)
            {
                if (EqualGates[j][0] == i)
                {
                    break;
                }
            }
            if (j==numOfEqGates)
            {
                EqualGates[numOfEqGates][0]=i;
            }
        }
        for (int i = 0; i < netlistRows; i++)
        {
            int j;
            for (j = 0; j < numOfEqGates; j++)
            {
                if (EqualGates[j][1] == i)
                {
                    break;
                }
            }
            if (j==numOfEqGates)
            {
                EqualGates[numOfEqGates][1]=i;
            }
        }
        numOfEqGates++;
    }

    for (int i = 0; i < numOfEqGates; i++)
    {
        // Correct Print for NOT Gates
        if (netlist[EqualGates[i][1]][1]==2||GoldenNetlist[EqualGates[i][0]][1]==2)
        {
            printf("G%d(%d)-->%d == G%d(%d)-->%d\n",netlist[EqualGates[i][1]][0],netlist[EqualGates[i][1]][2],netlist[EqualGates[i][1]][4],GoldenNetlist[EqualGates[i][0]][0],G
            fprintf(fpOut,"G%d(%d)-->%d == G%d(%d)-->%d\n",netlist[EqualGates[i][1]][0],netlist[EqualGates[i][1]][2],netlist[EqualGates[i][1]][4],GoldenNetlist[EqualGates[i][
```

Τέλος δημιουργούμε την αναφορά που θα δει ο τελικός χρήστης. Πρώτα όμως ελέγχουμε αν έχουμε βρει την ισοδυναμία για όλες τις πύλες. Αν μας λείπει μια ισοδυναμία μπορούμε εύκολα να βρούμε την πύλη που λείπει για κάθε αρχείο και να τις εξισώσουμε.

Τελικά εκτυπώνουμε στον χρήστη αν τα 2 αρχεία είναι ισοδύναμα. Αν το πρόγραμμα έχει φτάσει ως αυτό το σημείο τότε τα αρχεία είναι ισοδύναμα και δείχνουμε το κατάλληλο μήνυμα καθώς και τις ισοδυναμίες πυλών και ακμών.

Ανάλυση της Διαδικασίας

Η διαδικασία είναι αυτή που περιγράφεται από το παρακάτω flowChart.



Ripple Carry Adder 1 bit

Το πρόγραμμα δοκιμάστηκε πλήρως με έναν RCA 1 bit και δημιουργήθηκα 2 σωστά αρχεία ανακατεμένα με διαφορετικούς τρόπους και 2 αρχεία λάθος το ένα με πλεονάζοντα πύλη και το άλλο με έλλειψη πύλης:

Στον φάκελο **Correct Test Inputs**:

CorrectTest1.txt : Το αρχείο εισόδου της άσκησης 6 με ανακατεμένες πύλες .

CorrectTest2.txt : Το αρχείο εισόδου της άσκησης 6 με ανακατεμένες πύλες .

Στον φάκελο **Correct Test Outputs**:

CorrectTest1_out.txt : Το αρχείο εξόδου της άσκησης 6 με ανακατεμένες πύλες .

CorrectTest2_out.txt : Το αρχείο εξόδου της άσκησης 6 με ανακατεμένες πύλες.

Στον φάκελο **Wrong Test Inputs**:

WrongTest1.txt : Το αρχείο εισόδου της άσκησης 6 πλεονάζουμε πύλη .

WrongTest2.txt : Το αρχείο εισόδου της άσκησης 6 με έλλειψη πύλης .

Στον φάκελο **Output Lab5**:

WrongTest1_out.txt : Το αρχείο εξόδου της άσκησης 6 πλεονάζουμε πύλη .

WrongTest2_out.txt : Το αρχείο εξόδου τ της άσκησης 6 με έλλειψη πύλης .

Συμπέρασμα

Όπως φαίνεται και από όλα τα παραπάνω αρχεία το πρόγραμμα λειτουργεί κανονικά και βρίσκει αν 2 κυκλώματα είναι ισοδύναμα ή όχι καθώς ενημερώνει και τον χρήστη για πότε υπάρχει κάτι λάθος και τι είναι αυτό. Επίσης όταν τα κυκλώματα είναι ισοδύναμα εκτυπώνει τις ισοδυναμίες πυλών και συρμάτων.