ΕΡΓΑΣΤΗΡΙΟ 5

20/05/2021

ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΓΙΑ FORMAL VERIFICATION ΕΞΑΓΩΓΗ ΛΟΓΙΚΩΝ ΔΟΜΩΝ ΠΥΛΩΝ ΑΠΟ ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΤΡΑΝΖΙΣΤΟΡ

Ντουνέτας Δημήτρης
ΑΜ: 2016030141

Εισαγωγή

Σκοπός της πέμπτης άσκησης είναι η δημιουργία ενός προγράμματος για εξαγωγή των λογικών δομών σε επίπεδο πυλών από μια συνδεσμολογία Transistor. Αφού λάβουμε ως είσοδο ένα αρχείο με συνδεσμολογία Transistor, το διαβάζουμε και βρίσκουμε τις πύλες που είναι υλοποιημένες καθώς και τη μεταξύ τους σύνδεση όπως δηλαδή ήταν η είσοδος του εργαστηρίου 3. Για την υλοποίηση της άσκησης λήφθηκε υπόψιν η παραδοχή που χρησιμοποιούμε και στα προηγούμενα εργαστήρια σχετικά με CMOS ώστε να πηγαίνει το ρεύμα από τη τάση στη γείωση.

Λειτουργία του Προγράμματος

Ο τρόπος λειτουργίας του προγράμματος είναι αρκετά απλός. Ξεκινάει διαβάζοντας το αρχείο εισόδου 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 επίπεδο Netlist που μας έχει δοθεί. Πάνω σε αυτόν τον γράφο που έχουμε εξάγει, κάνοντας χρήση της συνάρτησης findGates() που έχουμε δημιουργήσει βρίσκουμε τις πύλες που έχει εισάγει ο χρήστης σύμφωνα με τη σύνδεση των Transistor καθώς και τον τρόπο που είναι συνδεδεμένες αυτές οι πύλες. Μπορούμε να βρούμε τις πύλες με οποιονδήποτε τρόπο και αν είναι αυτές τοποθετημένες στο επίπεδο netlist, ακόμα και αν είναι σπασμένες σε διάφορα σημεία και έχουν ενδιάμεσα Transistor που αποτελούν άλλες πύλες. Στο τέλος, εκτυπώνουμε το Dataset των πυλών με τις μεταξύ του συνδέσεις, όπως αποτελούνταν η είσοδος του εργαστηρίου 3.

Ανάλυση του Κώδικα

Αρχικοποιήσεις συναρτήσεων και μεταβλητών:

```
//Lab5 Functions
void findGates(FILE* fpOut);
void checkNot(int currentTrans,int gateNum,FILE* fpOut);
void checkNand(int currentTrans,int gateNum,FILE* fpOut);
void checkNor(int currentTrans,int gateNum,FILE* fpOut);
void checkForUnusedTrans(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* inputs;
int* outputs;
int **netlist;
int count,count1;
bool foundGate = false;
```

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");
    if (fpIn == NULL)
    {
        printf("Could not find file");
    }
    char file1[64];
    char dot = '.';

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

    FILE *fp1 = fopen(file1,"w");
    //get circuit file info
    getCircuitinfo(fpIn,fp1);

    return 0;
}
```

Ψήγμα κώδικα από της συνάρτηση getCircuitInfo():

```
//Find Testbench values save them and do test
if(strcmp(line, "## TESTBENCH")==0)
{
    printNetlist(fpOut);
    findGates(fpOut);
    printf("END_SIMULATION");
    fprintf(fpOut,"END_SIMULATION");
    checkForUnusedTrans(fpOut);
}
```

Η συνάρτηση getCircuitInfo() δουλεύει με τον ίδιο τρόπο όπως στα προηγούμενα εργαστήρια , δηλαδή αποθηκεύει δεδομένα από το αρχείο εισόδου, εκτός από την περίπτωση του TESTBENCH όπου καλεί τις καινούργιες συναρτήσεις του εργαστηρίου 5.

Ψήγμα κώδικα από τη συνάρτηση findGates():

```
void findGates(FILE* fpOut)
{
    int gateNum=1;
    for (int i = 0; i < netlistRows; i++)
    {
        foundGate = false;
        if (netlist[i][4]!=0)
        {
            continue;
        }
        checkNot(i,gateNum,fpOut);
        checkNand(i,gateNum,fpOut);
        checkNor(i,gateNum,fpOut);
        if(foundGate)
        {
            gateNum++;
        }
    }
}
```

Η συγκεκριμένη συνάρτηση είναι υπεύθυνη για την εύρεση και την εξαγωγή του αρχείου των πυλών από το επίπεδο Netlist των Transistor. Ξεκινάει αρχικοποιώντας την μεταβλητή gateNum στον αριθμό 1. Ύστερα διαβάζει το netlist από την αρχή ως το τέλος και κάθε φορά κάνει τη μεταβλητή foundGate false. Αν διαβάσει ένα Transistor που έχουμε ήδη χρησιμοποιήσει το προσπερνά, αλλιώς καλεί τις συναρτήσεις checkNot(),checkNand(),checkNor() ώστε να ελέγξει αν το συγκεκριμένο Transistor ανήκει σε μία από αυτές τις πύλες. Αν βρεθεί συσχετισμός με πύλη αυξάνεται ο αριθμός gateNum και όλα τα συσχετιζόμενα Transistor με την συγκεκριμένη πύλη μαρκάρονται με ένα διακριτό αριθμός για κάθε πύλη.

Ψήγμα κώδικα από τη συνάρτηση checkNot():

```
void checkNot(int currentTrans,int gateNum,FILE* fpOut)
{
    for (int i = 0; i < netlistRows; i++)
    {
        if (netlist[i][4]!=0)
        {
            continue;
        }
        if ((netlist[currentTrans][0]==0)&&(netlist[i][0]==1))//Check if current transistor is PMOS and i transistor is NMOS
        {
            if (netlist[currentTrans][1]==netlist[i][1])//Check if transistors gates are equal
            {
                //Check if PMOS Transistor drain is equal to NMOS Transistor drain
                //And check if PMOS Source is VCC and NMOS Source is GND
                if ((netlist[currentTrans][3]==netlist[i][2])&&netlist[currentTrans][2]==VCC&&netlist[i][3]==GND)
                {
                    foundGate = true;
                    netlist[currentTrans][4]=gateNum;
                    netlist[i][4]=gateNum;
                    printf("G%d NOT ; IN %d ; OUT %d\n",gateNum,netlist[currentTrans][1],netlist[currentTrans][3]);
                    fprintf(fpOut,"G%d NOT ; IN %d ; OUT %d\n",gateNum,netlist[currentTrans][1],netlist[currentTrans][3]);
                    return;
                }
            }
        }
```

```
else if ((netlist[currentTrans][0]==1)&&(netlist[i][0]==0))
{
    if (netlist[currentTrans][1]==netlist[i][1])
    {
        if ((netlist[currentTrans][2]==netlist[i][3])&&netlist[currentTrans][3]==GND&&netlist[i][2]==VCC)
        {
            //Found all transistors and the gates is complete
            foundGate = true;
            netlist[currentTrans][4]=gateNum;
            netlist[i][4]=gateNum;
            printf("G%d NOT ; IN %d ; OUT %d\n",gateNum,netlist[currentTrans][1],netlist[currentTrans][3]);
            fprintf(fpOut,"G%d NOT ; IN %d ; OUT %d\n",gateNum,netlist[currentTrans][1],netlist[currentTrans][3]);
            return;
        }
```

Η συγκεκριμένη ελέγχει αν το Transistor που βρίσκεται υπό έλεγχο ανήκει σε πύλη Not. Για να γίνει αυτός ο έλεγχος προσπαθούμε να βρούμε όλα τα χαρακτηριστικά Transistor που αποτελούν την συγκεκριμένη πύλη λαμβάνοντας υπόψιν ότι ξεκινάμε από ένα συγκεκριμένο. Αν καταφέρουμε να ενώσουμε όλα τα κομμάτια σημαίνει ότι βρήκαμε τη πύλη και όλα τα Transistor που την αποτελούν, οπότε και τα μαρκάρουμε με τον ίδιο αριθμό. Στην συνέχεια εκτυπώνουμε την πύλη που βρήκαμε και τα IN και OUT της.

Για να βρούμε μια πύλη NOT αρχικά θεωρούμε ότι ξεκινάμε είτε από το PMOS της είτε από το NMOS της.

Αν έχουμε ξεκινήσει από το PMOS τότε αρκεί να ψάξουμε για ένα NMOS. Αφού βρούμε το NMOS ελέγχουμε αν έχει ίδια πύλη με το αρχικό μας. Τέλος ελέγχουμε αν το drain του PMOS είναι ίδιο με το Drain του NMOS και αν το Source του PMOS είναι το VCC και το Source του NMOS είναι το GND.

Αν ισχύουν όλα τα παραπάνω τα 2 Transistor που μας έμειναν αποτελούν μια πύλη NOT και η είσοδος της είναι το Gate του PMOS και η έξοδος της είναι το Drain του PMOS.

Με ακριβώς αντίστοιχο τρόπο ελέγχουμε στην περίπτωση που ξεκινάμε από το NMOS Transistor και ψάχνουμε το PMOS του για να ολοκληρωθεί η πύλη.

Ψήγμα κώδικα από τη συνάρτηση checkNand():

```
void checkNand(int currentTrans,int gateNum,FILE* fpOut)
{
    for (int i = 0; i < netlistRows; i++)
    {
        if (netlist[i][4]!=0||i==currentTrans)
        {
            continue;
        }
        //check if PMOS Transistors are in parallel and Sources are VCC
        if ((netlist[currentTrans][0]==0)&&(netlist[i][0]==0)&&(netlist[currentTrans][2]==VCC)&&(netlist[i][2]==VCC)&&(netlist[currentTrans][2]==netlist[i][2])&&(netlist[currentTrans][3]==netlist[i][3]))
        {
            //Find the NMOS transistor that is in series with the second PMOS Transistor
            for (int j = 0; j < netlistRows; j++)
            {
                if ((netlist[j][0]==1)&&(netlist[j][2]==netlist[i][3]))
                {
                    //Find the last NMOS Transistor in series with the previous NMOS Transistor with its source connected to GND
                    for (int k = 0; k < netlistRows; k++)
                    {
                        if ((netlist[k][0]==1)&&(netlist[k][2]==netlist[j][3])&&(netlist[k][3]==GND))
                        {
                            //Found all transistors and the gates is complete
                            foundGate = true;
                            netlist[currentTrans][4]=gateNum;
                            netlist[i][4]=gateNum;
                            netlist[j][4]=gateNum;
                            netlist[k][4]=gateNum;
                            printf("G%d NAND_2 ; IN %d,%d ; OUT %d\n",gateNum,netlist[currentTrans][1],netlist[i][1],netlist[j][2]);
                            fprintf(fpOut,"G%d NAND_2 ; IN %d,%d ; OUT %d\n",gateNum,netlist[currentTrans][1],netlist[i][1],netlist[j][2]);
                            return;
                        }
                    }
```

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 το αρχικό Transistor είναι PMOS. Αν αυτό ισχύει τότε ψάχνουμε σε όλο το netlist να βρούμε το παράλληλο του και ελέγχουμε ώστε τα 2 Sources τους να είναι VCC. Στην συνέχεια ψάχνουμε ξανά όλο το αρχείο για να βρούμε το NMOS transistor που είναι σε σειρά με με το δεύτερο PMOS. Άμα το βρούμε κι αυτό ψάχνουμε πάλι όλο το netlist για να βρούμε το τελευταίο Transistor NMOS που έχει Drain ίδιο με το Source του προηγούμενου NMOS και Source GND. Αφού έχει γίνει όλη η διαδικασία και έχουν βρεθεί και τα 4 Transistor τα μαρκάρουμε με έναν διακριτό αριθμό και εκτυπώνουμε στο χρήστη το είδος της πύλης και τις εισόδους και εξόδους τις.

Ψήγμα κώδικα από τη συνάρτηση checkNor()

```
void checkNor(int currentTrans,int gateNum,FILE* fpOut)
{
    for (int i = 0; i < netlistRows; i++)
    {
        if (netlist[i][4]!=0||i==currentTrans)
        {
            continue;
        }
        //check if NMOS Transistors are in parallel and Sources are GND
        if ((netlist[currentTrans][0]==1)&&(netlist[i][0]==1)&&(netlist[currentTrans][3]==GND)&&(netlist[i][3]==GND)&&(netlist[currentTrans][2]==netlist[i][2])&&(netlist[currentTrans][3]==netlist[i][3]))
        {
            //Find the PMOS transistor that is in series with the second NMOS Transistor
            for (int j = 0; j < netlistRows; j++)
            {
                if (netlist[j][4]!=0)
                {
                    continue;
                }
                if ((netlist[j][0]==0)&&(netlist[j][3]==netlist[i][2]))
                {
                    //Find the last PMOS Transistor in series with the previous PMOS Transistor with its source connected to VCC
                    for (int k = 0; k < netlistRows; k++)
                    {
                        if (netlist[k][4]!=0)
                        {
                            continue;
                        }
                        if ((netlist[k][0]==0)&&(netlist[k][3]==netlist[j][2])&&(netlist[k][2]==VCC))
                        {
                            //Found all transistors and the gates is complete
                            foundGate = true;
                            netlist[currentTrans][4]=gateNum;
                            netlist[i][4]=gateNum;
                            netlist[j][4]=gateNum;
                            netlist[k][4]=gateNum;
                            printf("G%d NOR_2 ; IN %d,%d ; OUT %d\n",gateNum,netlist[currentTrans][1],netlist[i][1],netlist[j][3]);
                            fprintf(fpOut,"G%d NOR_2 ; IN %d,%d ; OUT %d\n",gateNum,netlist[currentTrans][1],netlist[i][1],netlist[j][3]);
                            return;
                        }
```

Η συγκεκριμένη συνάρτηση λειτουργεί με τον ίδιο ακριβώς τρόπο όπως και η checkNand με τη διαφορά ότι ξεκινάμε ψάχνοντας τα 2 παράλληλα NMOS πρώτα και μετά προχωράμε ώστε να βρούμε το PMOS που βρίσκεται σε σειρά με το 1 από αυτά και τέλος βρίσκουμε το PMOS που βρίσκεται σε σειρά με το πρηγούμενο PMOS με το Source του συνδεδεμένο στο VCC. Αφού τα βρούμε όλα προχωράμε τη διαδικασία όπως στα προηγούμενα.

Η συνάρτηση checkForUnusedTrans():

```
void checkForUnusedTrans(FILE* fpOut)
{
    for (int i = 0; i < netlistRows; i++)
    {
        if (netlist[i][4]==0)
        {
            printf("\nTransistor U%d is not used in a NOT , NOR or NAND gate\n",i+1);
            fprintf(fpOut,"\nTransistor U%d is not used in a NOT , NOR or NAND gate\n",i+1);
        }
    }
}
```

Απλά ελέγχει όλο το netlist και αν βρει μη μαρκαρισμένο Transistor σημαίνει ότι αυτό το Transistor δεν ανήκει σε NOT, NAND ή NOR και εκτυπώνει το όνομα του.

Ανάλυση της Διαδικασίας

Η διαδικασία εύρεσης των λογικών δομών με Formal Verification είναι αρκετά απλή. Ουσιαστικά γνωρίζουμε τον τρόπο διασύνδεσης των Transistor που χαρακτηρίζουν μια συγκεκριμένη πύλη και χρησιμοποιούμε αυτή την πληροφορία ως μάσκα φιλτράροντας όλα τα υπόλοιπα μέχρι να μας μείνουν μόνο αυτά που ψάχνουμε.

Για παράδειγμα γνωρίζουμε ότι μια πύλη NOT αποτελείτε από 2 Transistor που το ένα είναι PMOS και το άλλο NMOS με κοινό Gate, κοινό Drain και το Source του PMOS συνδεδεμένο στην τάση ενώ το Source του NMOS συνδεδεμένο στην γείωση.

Έχοντας βρει αυτά τα Transistor με τις παραπάνω προδιαγραφές έχουμε βρει και μια πύλη NOT.

Ακολουθώντας την ίδια διαδικασία για τις πύλες NAND και NOR μπορούμε να βρούμε όλες τις πύλες όσο μπερδεμένες και αν είναι μέσα στο επίπεδο netlist του αρχείου εισόδου.

Ripple Carry Adder 1 bit

Το πρόγραμμα δοκιμάστηκε πλήρως με έναν RCA 1 bit και με κανονική διάταξη των Transistor, όπως οι πύλες είχαν χωρική τοπικότητα, καθώς και με διαφορετική διάταξη όπου κομμάτια της ίδιας πύλης βρίσκονταν διάσπαρτα μέσα στο αρχείο, χωρίς χωρική τοπικότητα.

Τα αρχεία εισόδου και εξόδου βρίσκονται στον ίδιο φάκελο με αυτό το έγγραφο με ονόματα:

Στον φάκελο **Input Lab5**:

RCA_1bit_RightOrder.txt : Το αρχείο εισόδου της άσκησης 5 με χωρική τοπικότητα .

RCA_1bit_WrongOrder.txt : Το αρχείο εισόδου της άσκησης 5 χωρίς χωρική τοπικότητα .

Στον φάκελο **Output Lab5**:

RCA_1bit_RightOrder_out.txt : Το αρχείο εξόδου της άσκησης 5 με χωρική τοπικότητα .

RCA_1bit_WrongOrder_out.txt : Το αρχείο εξόδου της άσκησης 5 χωρίς χωρική τοπικότητα .

Συμπέρασμα

Όπως φαίνεται και από όλα τα παραπάνω αρχεία το πρόγραμμα λειτουργεί κανονικά και χρωματίζει τα κομμάτια της κάθε πύλης σωστά με διαφορετικό αριθμό, καθώς επίσης εκτυπώνει και τα στοιχεία κάθε πύλης μαζί με την πληροφορία σχετικά με τη σύνδεση των πυλών μεταξύ τους.