Αναφορά Τέταρτης εργαστηριακής άσκησης

ΗΡΥ 419 - ΗΡΥ 608 - Ανάπτυξη Εργαλείων CAD για Σχεδίαση Ολοκληρωμένων Κυκλωμάτων 2022- 2023

ΑΙΚΑΤΕΡΙΝΗ ΤΣΙΜΠΙΡΔΩΝΗ:2018030013

Το πρόγραμμα πού κατασκευάστηκε στην ουσία ενός φτιάχνει ένα netlist χρησιμοποιώντας τη γλώσσα προγραμματισμού C.
Το πρόγραμμα διαβάζει ένα αρχείο εισόδου που περιγράφει τις εισόδους και τις εξόδους του κύκλωματος(Αυθαίρετα εδώ θεωρήσαμε ότι είναι FULL_ADDER), επεξεργάζεται τις πληροφορίες και δημιουργεί ένα αρχείο netlist που αντιπροσωπεύει τις συνδέσεις του κυκλώματος.
Θα αναλυθούν οι δομές δεδομένων, οι λειτουργίες και οι αλγόριθμοι που χρησιμοποιούνται στο πρόγραμμα.

Δομές Δεδομένων:

Το πρόγραμμα χρησιμοποιεί διάφορες δομές δεδομένων για να αποθηκεύσει τις πληροφορίες που περιέχονται στο αρχείο εισόδου. Οι κύριες δομές που χρησιμοποιούνται είναι οι εξής:

1. Entity_list: Ένας πίνακας που αποθηκεύει τα στοιχεια που βρίσκονται στο αρχείο εισόδου.περιέχουν πληροφορίες για το όνομα, τον τύπο και τα χαρακτηριστικά(εισόδοι , έξοδοι).

2. Component_array: Ένας πίνακας που χρησιμοποιείται για να αποθηκεύσειτα στοιχεία που αντιπροσωπεύουν τα εξαρτήματα του κυκλώματος. (Για το αρχείο εξόδου).

3. OutputArray: Ένας πίνακας που χρησιμοποιείται για να αποθηκεύσειτις πληροφορίες που αφορούν τις εξόδους του κυκλώματος.

4. Netlist:Ένας πίνακας πού αποθηκεύει το Component_arraykαι το OutputArray.

Ο **αλγόριθμος** που χρησιμοποιήθηκε για τη δημιουργία των δομών είναι ο εξής ξεκινάμε από μία απλή δομή(**πχ. Iodeclaretion η οποία μας δυνατότητα να αποθηκεύσουμε μόνο το όνομά της εισόδου τον τύπο της εισόδου δηλαδή (IN ή OUT) και τα ονόματα των μεταβλητών(ΠΧ A0,A1,A2). Αφού ένα Port αποτελείται από πολλά Io declaration Φτιάχνω μία καινούργια δομή με πίνακα io declaration και την ονομάζουμε PORT.Το τελικό αποτέλεσμα όμως που θέλουμε εμείς είναι**

να φτιάξουν μία δομή Entity, ένα entity έχει μόνο ένα Port .'Ομως ενα entity περιέχει και μεταβλητές και μπορεί να περιέχει και παραπάνω από μία. Οπότε ακριβώς το ίδιο κάνουμε με μία δομή variable πού είναι η απλή δομή που περιέχει απλά το όνομα και την τιμή της μεταβλητής και μετά φτιάχνεται μία δομή που την ονομάζουμε variables_array και στην ουσία ένας πίνακας από variables. Κάπως έτσι συνδέεται η τελική δομή του ENTITY που αποτελείται από ένα PORT και πολλές μεταβλητές. Αυτό το ENTITY μπαίνει σε μία καινούργια δομή ENTITY_LIST σε περίπτωση που έχουμε παραπάνω από ένα entities που είναι και η κυρία δομή). Αυτός ακριβώς είναι ο αλγόριθμος που χρησιμοποιείται για την κατασκευή των δομών ξεκινώντας από μία απλή δομή που περιέχει την πληροφορία και αν χρειάζεται να πάρουμε περισσότερο από ένα πράγματα από αυτή την απλή δομή τότε κατασκευάζουμε πίνακα από αυτή και προσθέτουμε οποιαδήποτε μεταβλητή μας βοηθάει να επεξεργαστούμε καλύτερα τον πίνακα (πχ.numofEntity).

Ακολουθεί μια περιγραφή των **βασικών συναρτήσεων για τη διαχείριση αυτών των δομών**:

1. createEntityList(): Μια συνάρτηση που δημιουργεί και επιστρέφει μια νέα, Entity_list.

2. addEntity(): Μία συνάρτηση που φτιάχνει και προσθέτει ένα entity στην λίστα . Η συνάρτηση θα λαμβάνει την Entity_list, το όνομα της οντότητας, το variables_array και το PortMap ως ορίσματα.

3. createVariablesArray(): Μια συνάρτηση που δημιουργεί και επιστρέφει μια νέα δομή variables_array.

4. addVariable(): Μια συνάρτηση που προσθέτει μια μεταβλητή στο variables_array. Η συνάρτηση θα λαμβάνει το variables_array, το όνομα της μεταβλητής και την τιμή της ως ορίσματα.

5. createPort(): Μια συνάρτηση που δημιουργεί και επιστρέφει μια νέα δομή Port.
6. addIodeclaration(): Μια συνάρτηση που προσθέτει μια καταχώρηση Iodeclaretion στη δομή Port. Η συνάρτηση θα λαμβάνει το Port, το όνομα, την ετικέτα, το IOnames και το numofIOnames ως ορίσματα.

7. createComponentArray(): Μια συνάρτηση που δημιουργεί και επιστρέφει μια νέα δομή Component_array.

8. addComponent(): Μια συνάρτηση που προσθέτει ένα στοιχείο Component στη δομή Component_array. Η συνάρτηση θα λαμβάνει

το Component_array, το id, τον τύπο, τις εισόδους, την ετικέτα και το numofinputs ως ορίσματα.

9. createOutputArray(): Μια συνάρτηση που δημιουργεί και επιστρέφει μια νέα, κενή δομή OutputArray.

10. addOutputDec(): Μια συνάρτηση που προσθέτει μια καταχώρηση Output_Dec στη δομή OutputArray. Η συνάρτηση θα λαμβάνει το OutputArray, το Output, την origin και την ετικέτα ως ορίσματα.

11. createNetlist(): Μια συνάρτηση που δημιουργεί και επιστρέφει μια νέα δομή netlist.

12. setNetlistComponents(): Μια συνάρτηση που ορίζει το Component_array και το OutputArray της δομής netlist. Η συνάρτηση θα λαμβάνει το netlist, το Component_array και το OutputArray ως ορίσματα.

Βασικές συναρτήσεις για τη λειτουργία του προγράμματος:

1. ReadInputFile(): Αυτή η συνάρτηση θα είναι υπεύθυνη για την ανάγνωση του αρχείου εισόδου που περιέχει την περιγραφή του κυκλώματος. Κατά τη διάρκεια της ανάγνωσης, θα επεξεργάζεται τις πληροφορίες και θα τις αποθηκεύει στις κατάλληλες δομές δεδομένων (όπως Entity_list, variables_array, Component_array και OutputArray) χρησιμοποιώντας τις συναρτήσεις που περιγράφηκαν προηγουμένως.
2. makeOutputFile(): Αυτή η συνάρτηση θα είναι υπεύθυνη για τη δημιουργία του αρχείου εξόδου με βάση τις πληροφορίες που αποθηκεύονται στις δομές δεδομένων παραπανω . Θα διατρέχει τις δομές δεδομένων (όπως Entity_list, variables_array). Στην ουσία θα πάρει την δομή που κατασκευάσαμε στο ReadInputFile() και μία μορφής λογικής δηλαδή για παράδειγμα το ότι ta bit είναι MSB σε LSB θα κατασκευάσει δομές(component,Output) που θα περιέχουν μέσα της εισόδου στις εξόδους τα ονόματα και οποιαδήποτε άλλα χαρακτηριστικά χρειάζεται για να κατασκευαστεί ένα Netlist αυτές οι γραμμές θα αποθηκευτούν σε μία τελική δομή Netlist η οποία επιστρέφεται και από την συνάρτηση.
3. Αφού επιστρέφεται η δομή Netlist στην main κατασκευάζεται το τελικό αρχείο με fprintf.

Αποτελέσματα:

Το αρχείο εισόδου που χρησιμοποιήθηκε ήταν αυτό μας δόθηκε:

ENTITY adder IS
VAR N = 3 %% Number of Bits in the Adder
PORT(IN A:{ SO, YOU,THINK} %% A, MSB TO LSB
IN B:{ CAN, TELL,HEAVEN} %% B, MSB TO LSB
IN Cin:{FROM} %% Cin,
OUT S:{ HELL, BLUE,SKY} %% S, MSB TO LSB
OUT Cout:{PAIN} %% Cout,
)
END adder NETLIST

Το τελικό αποτέλεσμα που παίρνουμε στην έξοδο είναι τα παρακάτω:

```
1  U0 FULL_ADDER THINK,HEAVEN,FROM
2  U1 FULL_ADDER YOU,TELL,U0_Cout
3  U2 FULL_ADDER SO,CAN,U1_Cout
4  SKY= U0_S
5  BLUE= U1_S
6  HELL= U2_S
7  PAIN= U2_Cout
8
```

Που ήταν και το αναμενόμενο.
Μας ζητήθηκε επίσης να πάρουμε αυτό το πρόγραμμα και να δοκιμάσουμε αν δουλεύει στην άσκηση 3 και το αποτέλεσμα που πήραμε είναι αυτό:

```
U0 XOR2 THINK,HEAVEN
U1 XOR2 U0,FROM
U2 NAND2 FROM,U0
U3 NAND2 THINK,HEAVEN
U4 NOT U3
U5 NOT U2
U6 NOR2 U4,U5
U7 NOT U6
U8 XOR2 YOU,TELL
U9 XOR2 U8,U7
U10 NAND2 U8,U7
U11 NAND2 YOU,TELL
U12 NOT U11
U13 NOT U10
U14 NOR2 U12,U13
U15 NOT U14
U16 XOR2 SO,CAN
U17 XOR2 U16,U15
U18 NAND2 U16,U15
U19 NAND2 SO,CAN
U20 NOT U19
U21 NOT U18
U22 NOR2 U20,U21
```

```
U21 NOT U18
U22 NOR2 U20,U21
U23 NOT U22
SKY = U1
BLUE = U9
HELL = U17
PAIN = U23
```

Το όποιο λειτουργικά είναι σωστό.