ΑΝΑΠΤΥΞΗ ΕΡΓΑΛΕΙΩΝ CAD ΓΙΑ ΣΧΕΔΙΑΣΗ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ – ΗΡΥ 608 / 419
Εαρινό Εξάμηνο 2023

Προθεσμία: Παρασκευή 9/6/23 έως τα μεσάνυχτα, υποβολή στο eClass, χωρίς παράταση για το Project ή την αναφορά, και με εξέταση του Project την Δευτέρα 12/6/23

PROJECT Μαθήματος (Έκδοση 1.0 της εκφώνησης, μπορεί να υπάρχουν διορθώσεις)

ΠΛΗΡΗΣ ΠΡΟΣΟΜΟΙΩΣΗ ΛΟΓΙΚΟΥ ΚΥΚΛΩΜΑΤΟΣ

1.0 ΕΙΣΑΓΩΓΗ

Όπως έχει παρουσιαστεί στο μάθημα και έχει συζητηθεί εκτεταμένα, το project μας φέτος είναι η δημιουργία ενός πλήρους αλλά απλού ψηφιακού προσομοιωτή για τα λογικά κυκλώματα που επεξεργαστήκαμε στις Ασκέσεις 2-5. Προς τούτο:

- Θα αγνοήσουμε πλήρως την βιβλιοθήκη υποσυστημάτων
- Θα επεκτείνουμε κάπως το format της βιβλιοθήκης πυλών ώστε να περιλαμβάνει τον πίνακα αληθείας κάθε πύλης με ένα τρόπο πολύ εύκολο και συνοπτικό
- Θα έχουμε σαν είσοδο κυκλώματος το netlist από ένα λογικό κύκλωμα που θα απαρτίζεται μόνο από πύλες και την διασύνδεσή τους, με ονόματα είτε εσωτερικών κόμβων (U8, U11, κλπ.) είτε εξωτερικών σημάτων (A2, B3, κλπ.) για τις εισόδους των πυλών, ακολουθώντας το format που έχετε ήδη
- Θα έχουμε σαν είσοδο δεδομένων του κυκλώματος (testbench) ένα αρχείο με τα ονόματα των σημάτων και την λογική τους τιμή. Αυτό θα επιτρέπει το ίδιο κύκλωμα να προσομοιώνεται με διαφορετικές εισόδους αφού θα αλλάζει μόνο το αρχείο δεδομένων και όχι το αρχείο με το netlist του κυκλώματος. Το format θα επιτρέπει πολλές διαδοχικές τιμές, αλλά νοείται σαν συνδυαστικό κύκλωμα που κατασταλάζει κάθε φορά, όχι σαν ακολουθιακό κύκλωμα.

Επειδή ο χρόνος πιέζει, θα θεωρήσουμε σαν Project την παραπάνω προδιαγραφή. Η αναδιάταξη των εισόδων ή ο τρόπος υπολογισμών των τιμών των κόμβων με βάση την θεωρία γράφων δεν θα είναι παραδοτέο, αλλά για όποιον/όποιαν το κανει θα αποτελεί bonus. Προφανώς δεν μπορείτε να έχετε bonus αν δεν δουλεύει άψογα το πρόγραμμα προσομοίωσης, και ούτε το bonus αντικαθιστά το να έχετε καλούς και τεκμηριωμένους κώδικες και μία πάρα πολύ καλή αναφορά.

2.0 ΑΡΧΕΙΑ ΓΙΑ ΤΟ PROJECT

2.1 ΑΡΧΕΙΟ ΠΥΛΩΝ

Αλλάζουμε το **`COMPONENT LIBRARY`** ως εξής:

```
%%   THIS IS THE COMPONENT LIBRARY, COMPRISING OF GATES
**      COMPONENT LIBRARY
COMP NOT     ;    IN:P      ; 1, 0
COMP AND2    ;    IN:P,Q    ; 0, 0, 0, 1
COMP NAND2   ;    IN:P,Q    ; 1, 1, 1, 0
COMP OR2     ;    IN:P,Q    ; 0, 1, 1, 1
COMP NOR2    ;    IN:P,Q    ; 1, 0, 0, 0
COMP XOR2    ;    IN:P,Q    ; 0, 1, 1, 0
COMP XNOR2   ;    IN:P,Q    ; 1, 0, 0, 1
```

Το παραπάνω είναι το πλήρες αρχείο και δεν χρειάζεται τίποτα άλλο, μόνο πύλες έως δύο εισόδων θα χρησιμοποιήσουμε. Ο πίνακας αληθείας είναι ο προφανής, δηλαδή η έξοδος για τους συνδυασμούς εισόδων **00, 01, 10, 11** (οι είσοδοι είναι συμμετρικές οπότε δεν υπάρχει θέμα ποια είσοδος είναι ποια). Δεν θα υπάρχει subsystem library.

2.2 ΑΡΧΕΙΟ NETLIST ΚΥΚΛΩΜΑΤΟΣ

Το netlist του κυκλώματος είναι το γνωστό, και για να κάνουμε απλή την ζωή μας δεν αναφερόμαστε καν στο ποια σήματα είναι είσοδοι και ποιά έξοδοι, θεωρούμε δε ότι θα δώσουμε αρχεία (για προσομοίωση) που είναι ενδεχόμενα ελλιπή με την έννοια του να λείπει κάποιο σήμα εισόδου (και πρέπει να το βρείτε – δηλαδή να έχουμε τερματισμό προγράμματος χωρίς να έχουν διευθετηθεί ολα τα σήματα), αλλά δεν θα εμπεριέχει λάθη με την έννοια του να είναι δύο έξοδοι βραχυκυκλωμένες. Κάθε καταχώρηση θα έχει μία από τις δύο γνώριμες μορφές, όπως στο παρακάτω παράδειγμα:

```
U3 NAND2 U7,A6
```
ή
```
S = U11
```

Δηλαδή ή θα αρχίζει πάντα από **U** με έναν αριθμό έως το 99 κατόπιν και θα αφορά κάποια πύλη, στην ίδια γραμμή θα έχουμε το όνομα της πύλης και κατόπιν τα ονόματα των εισόδων που θα είναι είτε κάποιο εσωτερικό σήμα **Uxx** ή κάποιο εξωτερικό σήμα, π.χ. **A6** στο παραπάνω παράδειγμα. Τα εξωτερικά σήματα ΔΕΝ θα ορίζονται ως τέτοια. Θα υπάρχει όμως η δυνατότητα να ορίσουμε με όνομα την έξοδο κάποιας πύλης, όπως στον πολυπλέκτη του παρακάτω παραδείγματος που ορίζουμε με το σήμα **D** την έξοδο της πύλης **U4**:

```
U1 NOT C
U2 NAND2 A, U1
U3 NAND2 B, C
U4 NAND2 U2, U3
D = U4
```

Το netlist δεν θα έχει στοιχεία για το τι είναι το κύκλωμα, αλλά αν θέλετε να βάλετε δικά σας σχόλια σε γραμμές που θα αγνοούνται, μπορείτε να χρησιμοποιήσετε το format της Άσκησης 2.

2.3 ΑΡΧΕΙΟ TESTBENCH

Για να κρατήσουμε απλό το testbench, θα έχουμε δύο δεσμευμένες λέξεις **IN** και **OUT**. Κατόπιν, κάθε γραμμή στο πεδίο **IN** θα αρχίζει με το όνομα ενός σήματος που μετά από ένα κενό χαρακτήρα θα ακολουθείται από μία ή περισσότερες τιμές, χωρισμένες με κόμμα. Το πεδίο **OUT** θα ορίζεται μία φορά και θα έχει τα σήματα (ένα ή περισσότερα) στα οποία θέλουμε να δούμε την τιμή, ένα σε κάθε γραμμή (προσοχή: μπορεί να είναι και εσωτερικά σήματα ορισμένα σαν **Uxx**). Αν για παράδειγμα θέλουμε να δοκιμάσουμε όλες τις πιθανές εισόδους **A,B,C** στον παραπάνω πολυπλέκτη και μάλιστα με το να οργανώσουμε την πληροφορία με βάση την είσοδο ελέγχου **C**, τότε το testbench μας θα είναι:

```
IN
C 0, 0, 0, 0, 1, 1, 1, 1
A 0, 1, 0, 1, 0, 1, 0, 1
B 0, 0, 1, 1, 0, 0, 1, 1
OUT
D
```

3.0 ΔΙΕΞΑΓΩΓΗ ΠΡΟΣΟΜΟΙΩΣΗΣ

Η προσομοίωση θα γίνεται σε κόμβους `D` με την σειρά που εμφανίζονται στο αρχείο εισόδου και με double buffering όπως έχουμε περιγράψει στο μάθημα. Για κάθε σετ εισόδων θα ολοκληρώνεται η προσομοίωση και θα βγαίνει η έξοδος, κατόπιν δε θα πηγαίνει ο προσομοιωτής στην επόμενη τιμή. Ένα ενδεικτικό αρχείο εξόδου θα είναι, για το παραπάνω κύκλωμα και testbench είναι το εξής:

```
INPUTS  | OUTPUTS

C A B | D
0 0 0 | 0
0 0 1 | 0
0 1 0 | 1
0 1 1 | 1
1 0 0 | 0
1 0 1 | 1
1 1 0 | 0
1 1 1 | 1
```

Επί πλέον από την έξοδο, θέλουμε και κάποια χαρακτηριστικά του τρεξίματος του προγράμματος, όπως π.χ. αριθμό επαναλήψεων για να συγκλίνει, και χρόνο εκτέλεσης.

3.0 ΣΗΜΑΝΤΙΚΕΣ «ΛΕΠΤΟΜΕΡΕΙΕΣ»

Δοκιμάσετε ενδελεχώς τον κώδικά σας. Σίγουρα θα έχετε από εμένα για δοκιμή και κάποιους κώδικες (π.χ. adders κάποιων Bit ακρίβειας που όμως δεν θα φαίνονται τι είναι αφού έχουμε μόνο πύλες και όχι subsystems). Δοκιμάσετε τον κώδικά σας με «ανακατεμένες» τις πύλες και δείτε ότι λειτουργεί σωστά. Στο τελικό παραδοτέο περιλάβετε και *όλα* τα αρχεία εισόδου με τα οποία δοκιμάσατε τους κώδικές σας, μαζί με τα αρχεία εξόδου, και στην αναφορά. Εκτός από μεθοδολογία, κλπ. γράψετε και πως δοκιμάσατε τους κώδικές σας. Κατ' αρχήν η απόδοση (ταχύτητα) του κώδικα δεν βαθμολογείται (εκτός αν κάνετε κάτι πάρα πολύ χαζό), αλλά η καλή δομή και επιβεβαίωση λειτουργίας βαθμολογείται.

4.0 BONUS

Αυτό που είχαμε πει στο μάθημα σαν «Φάση 2» του Project βγαίνει εκτός, και γίνεται Bonus με αυτόνομη βαθμολόγηση, αλλά μόνο εφόσον γίνει σωστά το κανονικό project. Τι είναι αυτό; Το να αναδιατάξετε τις πύλες ή την σειρά υπολογισμού χρησιμοποιώντας θεωρία γράφων ώστε σε κάθε πέρασμα να έχουμε την μέγιστη δυνατή πληροφορία (προσοχή: αυτό δεν αναιρεί το double buffering, αλλά με τις κατάλληλες δομές μπορεί π.χ. να έχουμε λιγότερους κόμβους προς επίλυση από επανάληψη σε επανάληψη).

Αν υπάρχουν λάθη ή ασάφειες παρακαλώ στείλετέ μου e-mail στο adollas@tuc.gr ώστε να το διευθετήσουμε.

ΚΑΛΗ ΕΠΙΤΥΧΙΑ!!!